Community Experience Distilled

DevOps for Web Development

Achieve continuous development and deployment of your web applications with ease

Mitesh Soni

[PACKT] PUBLISHING open source* community experience distilled

Table of Contents

1
Getting Started - DevOps Concepts, Tools, and Technology

The first rule of any technology used in a business is that automation applied to an efficient operation will magnify the efficiency. The second is that automation applied to an inefficient operation will magnify the inefficiency – Bill Gates

DevOps is not a tool or technology; it is an approach or culture that makes things better. This chapter describes in detail on how DevOps solves different problems of traditional application delivery cycle. It also describes how it can be used to make Development and Operations teams efficient and effective to make time to market faster by improving culture. It also explains key concepts essential for evolving DevOps culture.

Readers will learn about DevOps culture, its lifecycle and its key concepts, tools, technologies and platforms used for automating different aspects of application lifecycle management.

In this chapter, we will cover the following topics:

- Understanding DevOps movement
- DevOps Lifecycle-All about “Continuous”
- Continuous Integration
- Configuration Management
- Continuous Delivery / Continuous Deployment
- Continuous Monitoring
- Continuous Feedback

- Tools and Technologies
- Overview of Sample JEE Application

Understanding DevOps movement

Let's try to understand what DevOps is. Is it a real technical word? Answer is No and the reason for this is DevOps is not about only technical stuff. It is also not a technology nor an innovation. In simple terms, DevOps is a blend of complex terminologies. It can be considered as a concept, culture, development and operational philosophy or a movement.

To understand DevOps, let's revisit the old days of any IT organization. Consider there are multiple environments where application is deployed. Following sequence of events takes place when any new feature is implemented or bug is fixed:

1. The development team writes a code to implement a new feature or fixes a bug. New code is deployed in to development environment and generally tested by the development team.
2. New code is deployed in the QA environment where it is verified by the testing team.
3. New code is provided to the operations team for deploying it into production environment.
4. Operation team is responsible for managing and maintaining the code

Let's list out the possible issues in the above mentioned approach:

- Transition of current application build from development environment to production environment lasts over weeks or months
- Priorities of Development Team, QA Team and IT Operations Team are different in an organization and effective and efficient co-ordination becomes necessity for smooth operations
- Development team is focused on latest development release while Ops team cares about Stability of an Production environment
- Development and Operations team are not aware about each other's work and work culture
- Both teams work in different type of environments; there is a possibility where development team has resource constraints and hence they manage different kind of configuration. It may work on localhost or in Dev environment.
- Operations team work on production resources and thus there will be a

huge gap in configuration and deployment environment. It may not work where it needs to run – in production environment.

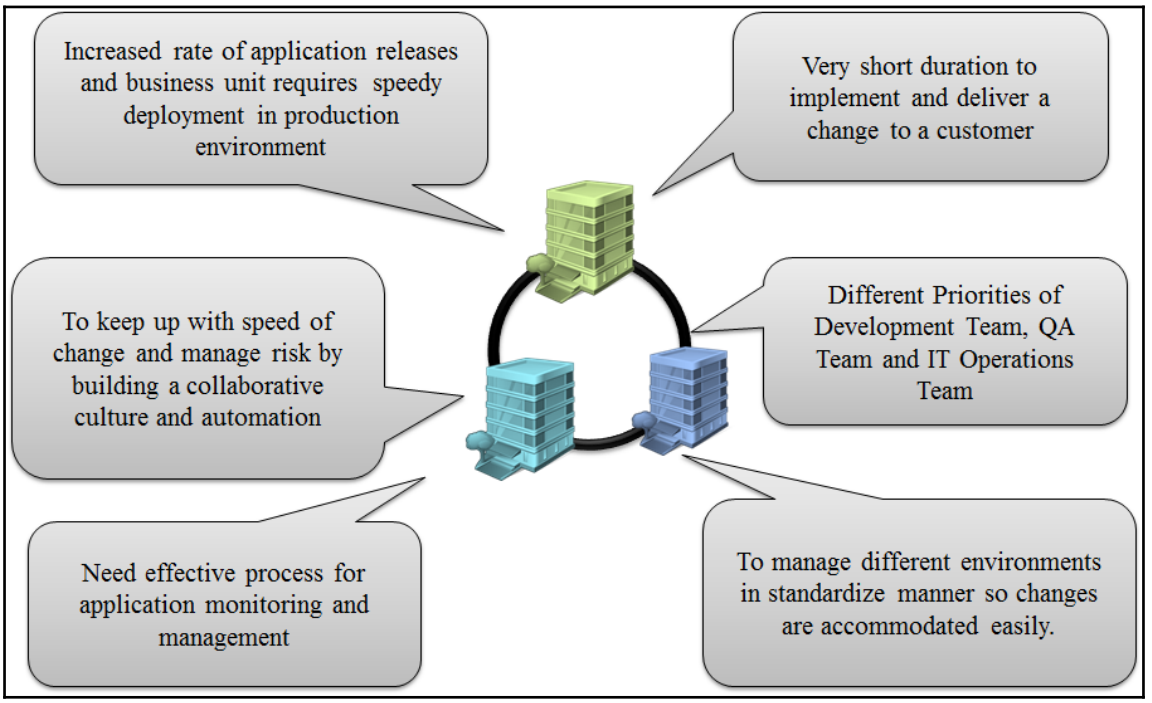
- Assumptions are key in such scenario and it is not possible that both team will work under same set of assumptions
- There is a manual work involved in the setting up runtime environment, configuration, and deployment activities. The biggest issue with manual application deployment process is its non-repeatability and manual process is error-prone.
- The development team has the installable files, configuration files, database scripts, and deployment documentation. They provide it to operations team. All these artifacts are verified in development environment and not in production or staging.
- Each team may take different approach for setting up runtime environment, configuration, and deployment activities considering resource constraints and resource availability.
- In addition, deployment process needs to be documented for future usage. Now, maintaining the documentation is a time-consuming task that requires collaboration between different stakeholders.
- Both teams work separately and hence there can be situation where both use different automation techniques
- Both teams are not aware about challenges faced by each other and hence they may not be able to visualize or understand an ideal scenario where application works
- While operations team is busy in deployment activities, development team may get another request for feature implementation or bug fix; in such case, if operations team faces any issues in deployment then they may try to consult development team who is already occupied in new implementation. It results in communication gaps and required collaboration may not happen.
- There is hardly any collaboration between the development team and the operations team. The poor collaboration that causes many issues in the application deployment to different environments that results into back and forth communication via mail, chat, calls, meetings, and so on and it often ends up with quick fixes.
- Challenges for Developers Team
    - Competitive Market creates pressure of on time delivery
    - Production ready Code Management and New feature Implementation
    - Release cycle is often long and hence development team

has to make assumptions before the application deployment finally takes place. In such scenario it takes more time to fix the issues occurred while deployment in staging or production environment.

- Challenges for Operations Team
  - Resource contention – Difficult to handle increasing demands of resources
  - Redesigning or tweaking is needed to run the application in Production environment
  - To diagnose and rectify the issues after application deployment in isolation

DevOps with changing times

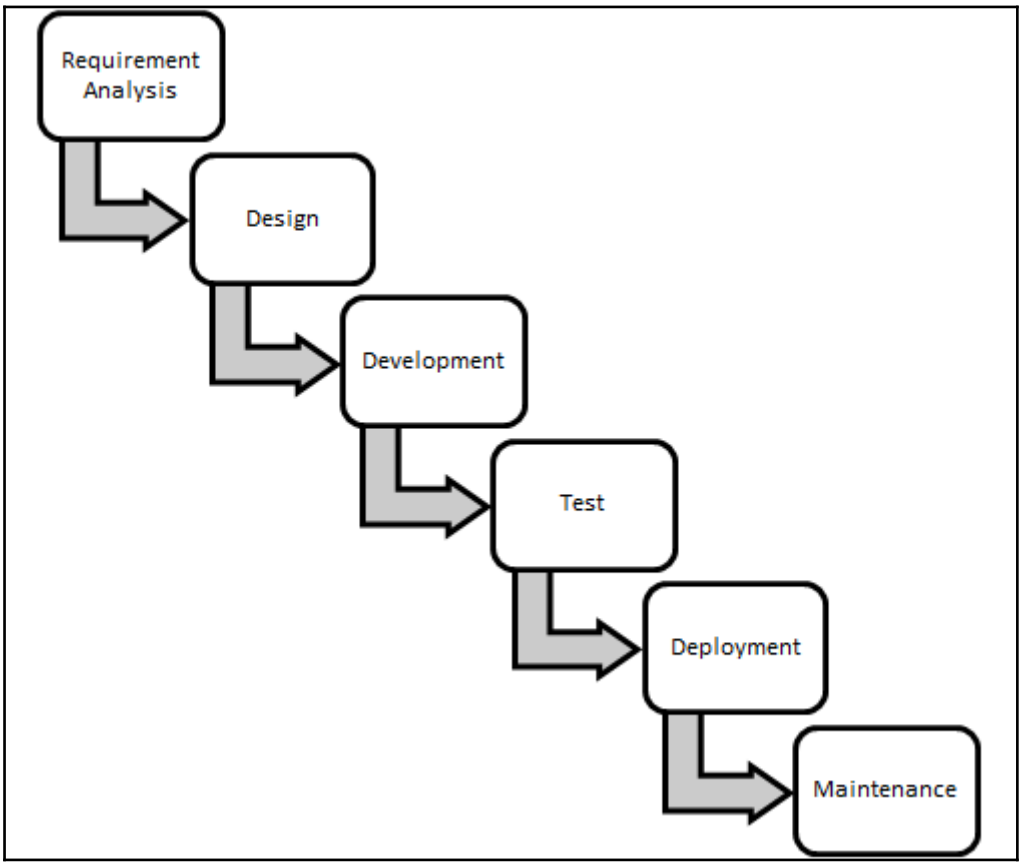
Time changes everything. In modern era, customers expect and demand extremely quick response, and we need to deliver new features continuously to stay in the business. Users and customers today have rapidly changing needs, they expect 24/7 connectivity and reliability, and access services over smart-phones, tablets and PCs. As software product vendors – irrespective of whether in development and / or operations – organizations need to push updates frequently to satisfy customers' needs to stay relevant. In short, organizations are facing following challenges:



Change in the behavior of customer or market demand affected the way development process takes place.

The waterfall model

Since long, Waterfall Model is used for software development.



It has its own advantages as follows:

- Easy to understand
- Easy to manage – Input and Output of each phase is defined
- Sequential process – Order is maintained
- Better control

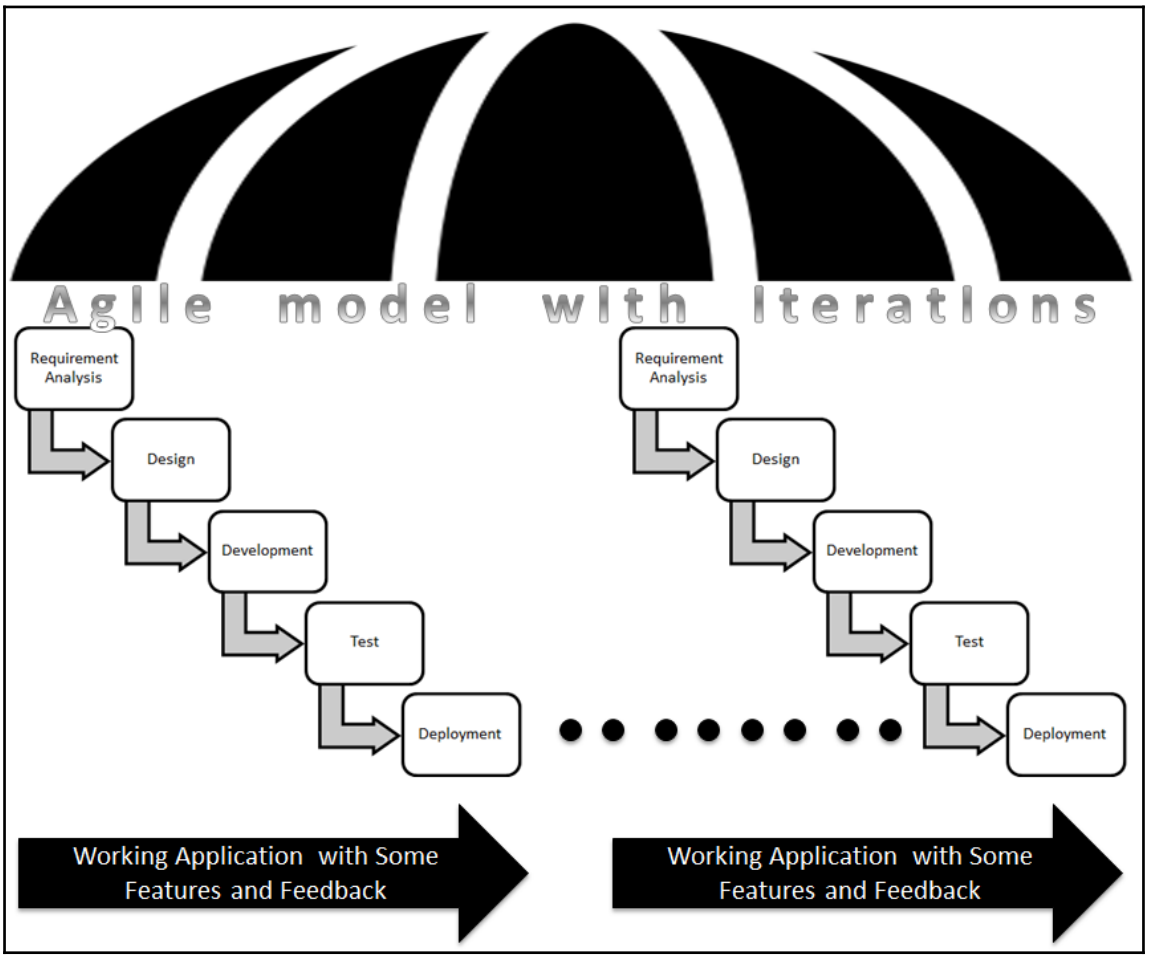
However, it was useful in the scenarios where requirements are predefined and fixed. As it is a rigid model with sequential process, we can't go back to any phase and change things. It has its own share of disadvantages as follows:

- No Revision
- No outcome or application package until all phases are completed

- Not possible to integrate feedback until all phases are completed
- Not suitable for changing requirements
- Not suitable for long term and complex projects

Agile Model

In Waterfall model, inefficient estimation, long time to market, and other issues led to a change. It is known as Agile Model. Agile development or agile methodology is a method of building an application by empowering individuals and encouraging interactions, by giving importance to working software, by customer collaboration – using feedback for improvement in next steps, and responding change in efficient manner. It emphasizes on customer satisfaction through continuous delivery in small interactions for specific features in short timelines or sprints.



One of the most attractive benefits of agile development is Continuous Delivery in short time frames or in agile terms – Sprints. Now, it is not a one-time deployment but it is the case of multiple deployments. Why? After each sprint, application with some feature is ready for showcase. It needs to be deployed in the specific environments for demo and thus deployment is no longer a one-time activity.

It is extremely essential from organization's perspective to meet changing demands of customers. To make it more efficient, communication and collaboration between all cross functional teams is essential. Many organizations have adopted agile methodology.

In such case, traditional manual deployment processes work as a speed breakers for

incremental deployments. Hence, it is necessary to change other processes also along with the change in application development methodology. One key can't be used for all the locks; similarly waterfall is not suitable in all projects. We need to understand that Agile is customer focused and feedback is vital. Based on customer feedback, changes happen and release cycles may increase. Just imagine a scenario where input is high but input processing is slow. Consider an example of a shoe company where one department prepares shoes and another department is working on final touches and packaging. What will happen if packaging process is slow and inefficient? It will be a shoe pile up in the packaging department. Now let's add a twist in this situation. What if shoe making department brings new machines and improve process of making shoe. It makes shoe making process 2 to 3 times faster. Now imagine a situation of packaging department. Similarly, Cloud computing and DevOps has gained momentum that increases speed of delivery and improve quality of end product. Thus, agile approach of application development, improvement in technology, and disruptive innovations and approaches has created a gap between development and operations team.

Collaboration

DevOps attempts to fill the gaps by developing a partnership between Development and Operations team. DevOps movement emphasizes communication, collaboration and integration between software developers and IT operations. DevOps promotes collaboration and collaboration is facilitated by automation and orchestration to improve processes. In other words, DevOps essentially extends the continuous development goals of the agile movement to continuous integration and release. DevOps is a combination of agile practices, processes leveraging the benefits of cloud solutions. Agile development and testing methodology help us to meet the goals of continuous integrate, develop, build, deploy, test, and release application. It provides mechanism for constant feedback from different teams and stakeholders. It also provides transparency, platform for collaboration across teams such as business analysts, developers and testers. In short, Agile and DevOps are compatible and increases value of each other.

One of the most popular saying is practice makes a man perfect. What if that saying is applied in production like environment? It will be much easier to repeat the entire process as there is no last minute surprises and most of issues in the deployment are already experienced and dealt with. The development team supports operational requirements such as deploy scripts, diagnostics, and load and performance testing from the beginning of the application delivery life cycle; and the operations team provides knowledgeable support and feedback before, during, and after deployment. The remedy is to integrate the testing, deployment, and release activities into the development process. By performing all activities multiple times and ongoing part of development so that by the time you are ready to release your system into production there is little to no risk, because deployment process

is already rehearsed it on many different environments in a progressively more production-like environments.

Cloud Computing: The Disruptive Innovation

One of the major challenges is to manage infrastructure for all environments. Virtualization and Cloud environment can help to get started with this. Cloud helps us to overcome this hurdle by providing flexible on demand resources and environments. It provides distributed access across the globe and helps in effective utilization of resources. Cloud provides repository of software, tools which can be used on-demand basis. We can clone environments, reproduce required versions as and when required. The entire development, test, and production environments can be monitored and managed using the facilities provided by the cloud providers. With the advent of Cloud computing, it is easy to re-create every piece of infrastructure used by application with the use of automation. That means operating systems, OS configuration, runtime environments, its configuration, infrastructure configuration, and so forth can all be managed. In this way, it is easy to recreate production environment exactly in an automated fashion. Thus DevOps on Cloud brings in the best of breed from both agile development as well as cloud solutions. It helps in providing Distributed Agile in Cloud, leading to Continuous Accelerated Delivery.

Why DevOps?

DevOps is effective because of new methodology, automation tools, agile resources by cloud service providers, and other disruptive innovations, practices, and technologies. However, it is not only about tools and technology. DevOps is more about culture than tools or technology alone.

> *Technology is just a tool. In terms of getting the kids working together and motivating them, the teacher is the most important-Bill Gates*

There is an urgent need of huge change the way development and operations team collaborates and communicates. Organizations need to have chnage in culture and have long term business goals that include DevOps in vision. It is important to establish pain points and obstacles experienced by different teams or business units and use that knowledge for refining business strategy and fix goals.

> *People always fear change. People feared electricity when it was invented, didn't they? People feared coal; they feared gas-powered engines… There will always be ignorance, and ignorance leads to fear. But with time, people will come to accept their silicon masters-Bill Gates*

If we identify common issues faced by different section of organization and change strategy to bring more value then it makes sense. It can be a stepping stone in the direction of DevOps. With same old values and objectives, it is difficult to adopt any new path. It is very important to align people with new process first. For example, team has to understand value of agile methodology else they will resist using it. They might resist it because they are comfortable with old process. Hence, it is important to make them realize the benefit and empower them also to bring the change.

> *Change is hard because people over estimate the value of what they have-and under estimate the value of what they may gain by giving that up-James Belasco and Ralph Stayer*

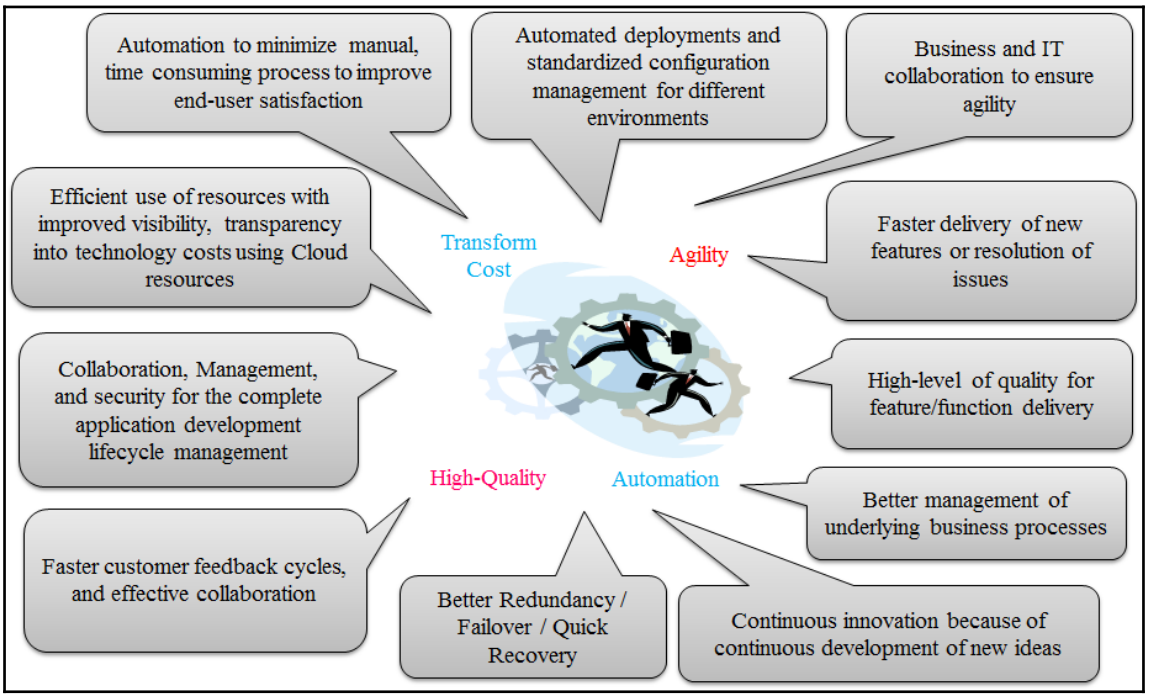
Self-dependent teams bring best out of them when they are empowered. We also need to understand that power comes with accountability and responsibility. Cross functional teams work together and enhance the quality by giving their expertise in the development process; however it is not isolated function. Communication and collaboration across teams makes quality way higher.

The end objective of DevOps culture is Continuous Improvement. We learn from mistakes and it becomes experience. Experience helps us to identify robust design patterns and minimize errors in the processes. This leads to enhancement of productivity and hence we achieve new heights with continuous innovations.

> *Software innovation, like almost every other kind of innovation, requires the ability to collaborate and share ideas with other people, and to sit down and talk with customers and get their feedback and understand their needs-Bill Gates*

Benefits of DevOps

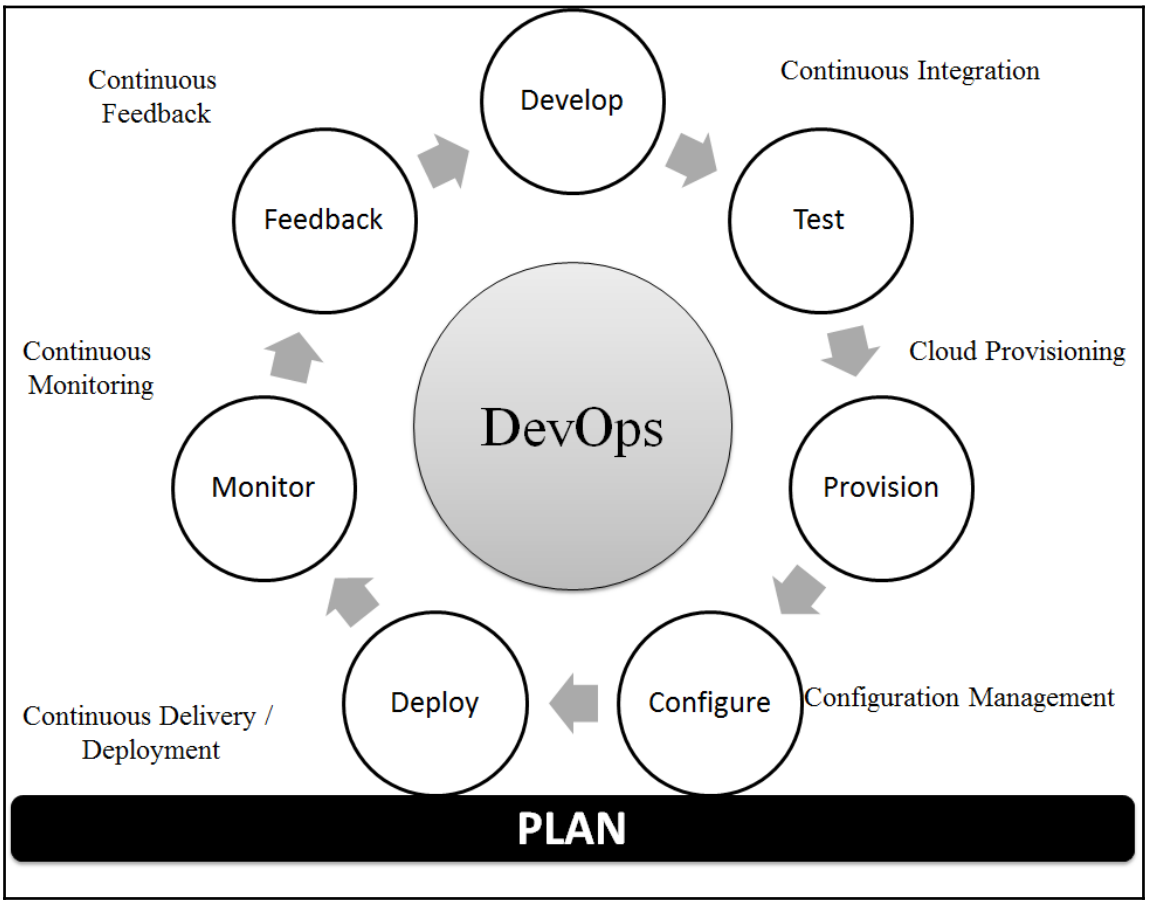
We will be covering all the benefits of DevOps in the following image:



Collaboration across different stakeholders brings many business and technical benefits that helps organizations to achieve their business goals.

DevOps lifecycle – all about "Continuous"

Continuous integration (CI), Continuous Testing (CT), and continuous delivery (CD) are significant part of DevOps culture. CI includes automation of build, unit test and package process while CD includes application delivery pipeline across different environments. CI and CD accelerates the application development process through automation across different phases such as build, test, code analysis and so on; and enables users to achieve end to end automation for application delivery lifecycle.

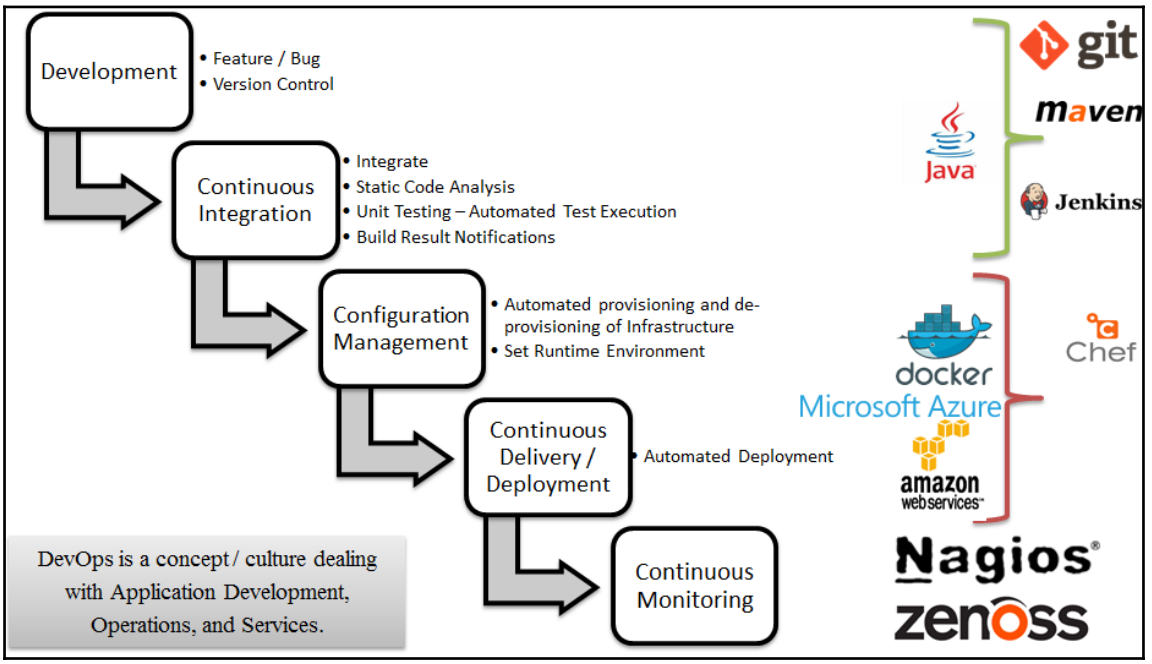


Continuous Integration and Continuous Delivery or Deployment is well supported by Cloud provisioning and Configuration Management. Continuous Monitoring helps to identify issues or bottlenecks in the end to end pipeline and helps to make pipeline effective.

Continuous Feedback is integral part of this pipeline which directs the stakeholders whether are near to the required outcome or going in the different direction.

> *Continuous effort — not strength or intelligence — is the key to unlocking our potential- Winston Churchill*

Following diagram shows mapping of different parts of Application delivery pipeline with toolset for Java Web application.



We will use Sample spring application throughout this book for demonstration purpose and hence toolset is related to Java technology.

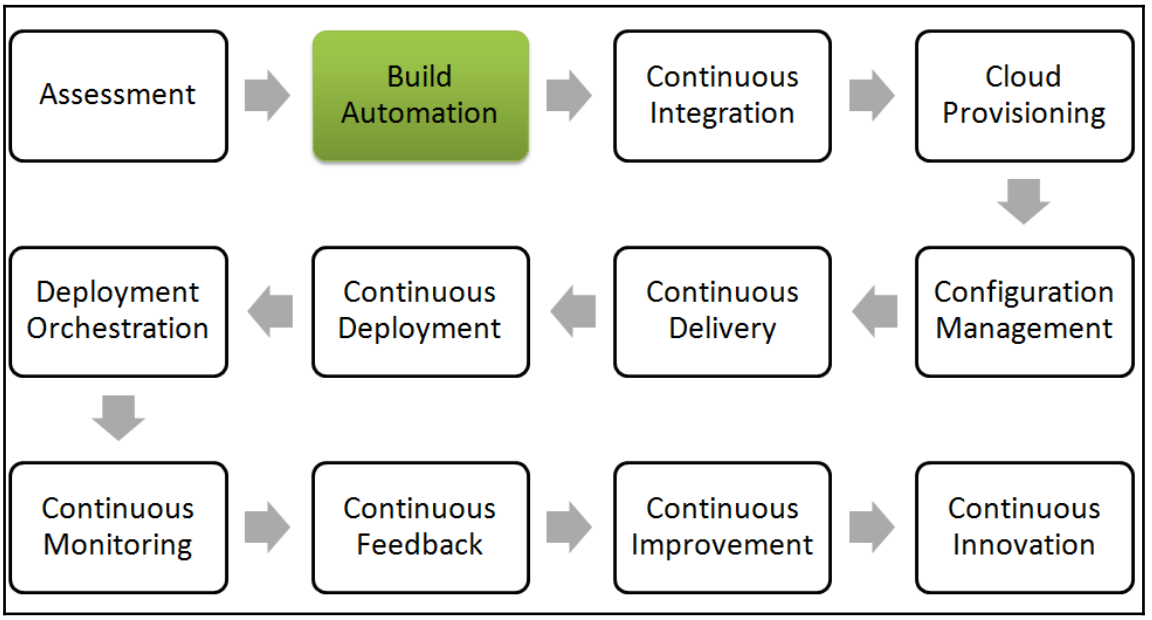
Build automation

Automated build helps to create application build using build automation tools such as Apache Ant, Apache Maven, and so on. Automated build process include following activities:

- Compile Source Code into Class files or Binary Files
- To provide reference to the third party library files
- To provide path of configuration files
- Packaging Class files or Binary Files into WAR files in case of Java
- To execute automated test cases
- To deploy WAR file into local or remote machine
- To reduce manual effort in creating WAR file

Apache Maven and Apache Ant automate build process and it makes build process simple,

repeatable, less error prone as it is a Create once Run Multiple times concept. Build automation is base of any automation in Application Delivery Pipeline.

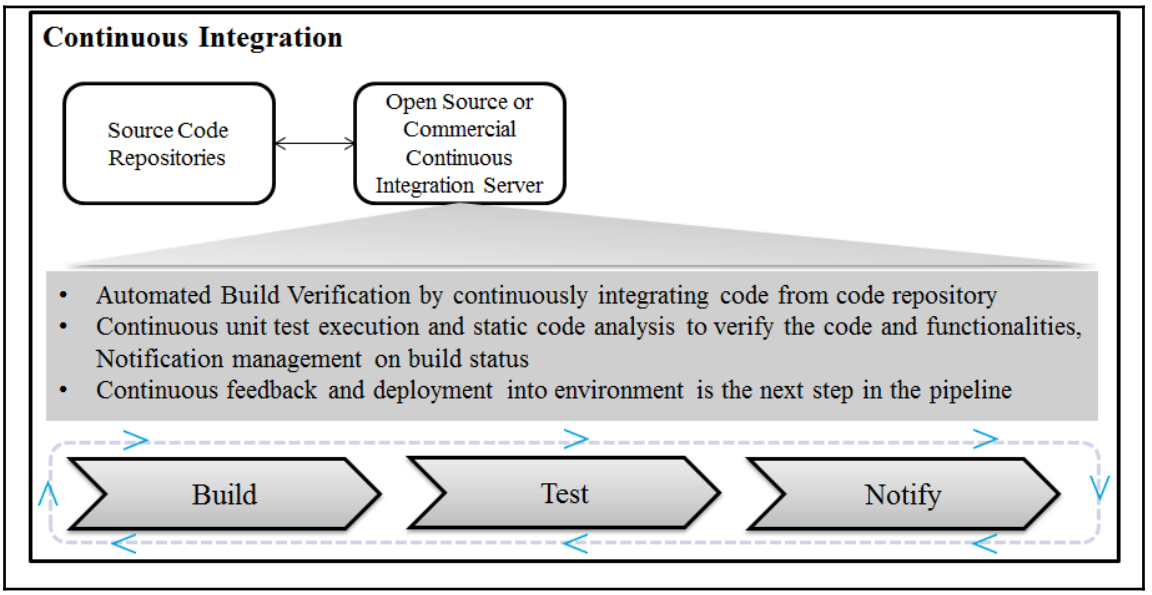


Build automation is essential for Continuous Integration and rest of the automation is effective only if build process is automated. All CI servers such as Jenkins, Atlassian Bamboo and so on are using build files for continuous integration and creating application delivery pipeline.

Continuous integration

What is Continuous Integration? In simple words, Continuous Integration (CI) is a software engineering practice where each check-in by a developer is verified by

- Pull mechanism: executing automated build at a scheduled time or
- Push mechanism: executing automated build when changes are saved in repository and
- Executing unit test against latest changes available in source code repository.



The main benefit of continuous integration is quick feedback based on the result of build execution. If it is successful then all is well else fix the responsibility on the developer whose commit has broken the build, notify all stakeholders and fix the issue.

Continuous Integration
http://martinfowler.com/articles/continuousIntegration.html

Why CI is needed or in other words, what is the requirement of it? Answer is, it makes things simple and identify bugs or errors in the code at very early stage of development and it is relatively easy to fix them. Just imagine if same scenario takes place after a long duration and there are too many dependencies and complexities we need to manage. In early stages it is far easier to cure and fix issues; consider health issues as an example and things will be more clear in that context.

Continuous Integration (CI) is a development practice that requires developers to integrate code into a shared repository several times a day. Each check-in is then verified by an automated build, allowing teams to detect problems early.

CI is a significant part and in fact a base of release management strategy of any organization that wants to develop DevOps culture.

Following are instant benefits of CI:

- Automated integration with Pull or Push mechanism
- Repeatable Process without any manual intervention
- Automated test case execution
- Coding standard verification
- Execution of scripts based on the requirement
- Quick feedback – Build status notification to stakeholders via mail
- Teams are focused on their work and not in managing processes

Jenkins, Apache Continuum, Buildbot, GitLabCI, and so on are some of the examples of open source CI Tools. AnthillPro, Atlassian Bamboo, TeamCity, Team Foundation Server, and so on are some of the examples of commercial CI Tools.

Best practices

We will now be looking at the best practices that can be useful while considering Continuous Integration implementation:

- Maintain a code repository such as Git or SVN
- Check-in third-party jars, build scripts, other artifacts and so on into Code repository
- It is advisable to execute builds fully from Code repository – Use clean build
- Automate the build using Maven or Ant for Java
- Make the build self-testing: Create unit tests
- Commit all changes at least once a day per feature
- Every commit should be built to verify the integrity of changes
- Authenticate users and enforce access control (Authentication and Authorization)
- Use alphanumeric characters for build names and avoid symbols
- Keep different build jobs to maintain granularity and managing operations in a better way. Single job for all task is difficult when we try to troubleshoot. It also helps to assign build execution to slave instances if that concept is supported by CI server
- Backup Home directory of CI server regularly as it contains archived builds and other artifacts too which may be useful in troubleshooting
- Make sure CI server has enough free disk space available as it store lot of details related to builds
- Better not to schedule multiple jobs to start at the same time or use master slave

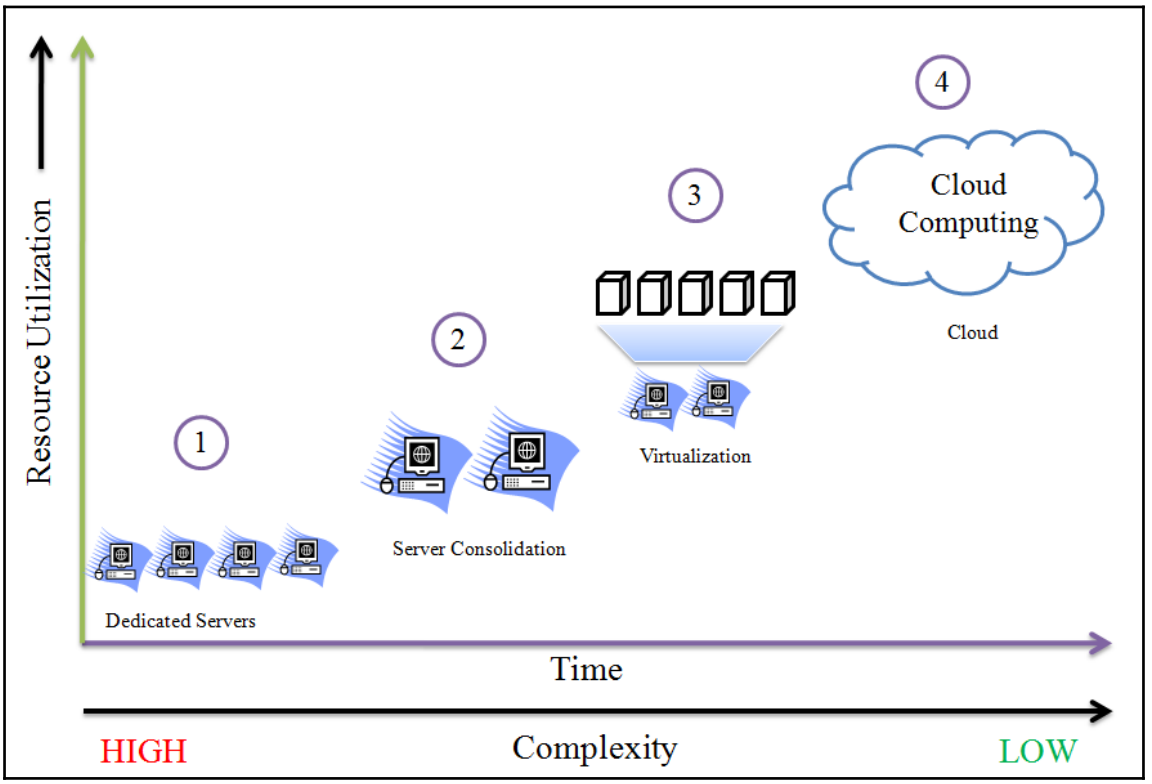
concept where specific jobs are assigned to slave instances so multiple build jobs can be executed at same time

- Set up Email, SMS or twitter notification to specific stakeholders of a project or an application. It is advisable to use customized mail to specific stakeholders
- It is advisable to use community plugins

Cloud computing

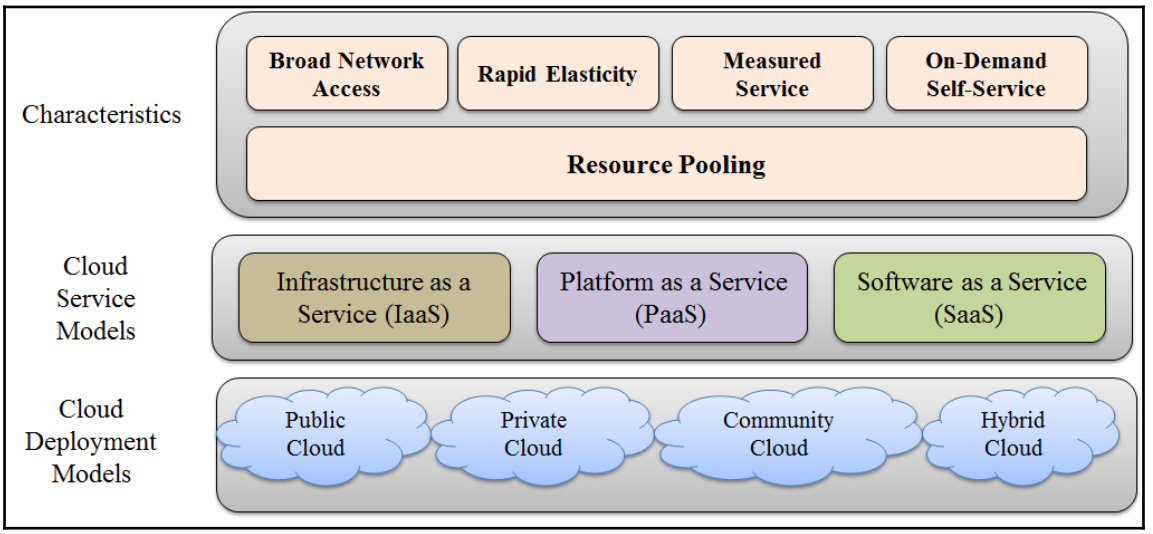
Cloud Computing is regarded as a ground breaking innovation in the recent years. It is reshaping the technology landscape. With breakthroughs made in appropriate service and business models, cloud computing has expanded its role as a backbone for IT services. Based on the experience, organizations improved from dedicated servers to consolidation, to virtualization to Cloud computing.

Cloud Computing provides elastic and unlimited resources which can be efficiently utilized in the time of peak load and normal load with pay per use pricing model. Pay as you go feature is a boon for development team which had faced resources scarcity since years. It is possible to automate provisioning resources and configuring resources based on requirements and that has reduced a lot of manual effort.



NIST SP 800-145, The NIST Definition of Cloud computing
http://csrc.nist.gov/publications/nistpubs/800-145/SP800-145.pdf

It has opened various opportunities in terms of availability of application deployment environments considering three service models and four deployment models.



There are four Cloud Deployment Models with that address specific requirements.

- Public Cloud: Cloud Infrastructure is available to general public
- Private Cloud: Cloud Infrastructure is operated for single organization
- Community Cloud: Cloud Infrastructure is shared by specific community that has shared concerns
- Hybrid Cloud: Cloud Infrastructure is composition of two or more cloud models

Cloud computing is pivotal components if we want to achieve our goals of automation to empower DevOps culture in any organization. Infrastructure can be considered as a code can be treated similar to code while creating resources, configuring them and managing resources with use of configuration management tools. Cloud resources play essential role in to successful adoption of DevOps culture. Elastic, scalable and pay as you go resource consumption allows organization to use same type of cloud resources in all different environments. The major problems in all the environments are inconsistency and limited capacity. Cloud computing solves this problem and that to with economic benefits.

Configuration management

Configuration Management (CM) manages changes in the system or to be more specific, in the server runtime environment. Let's consider an example where we need to manage multiple servers with same kind of configuration. For an example, we need to install tomcat

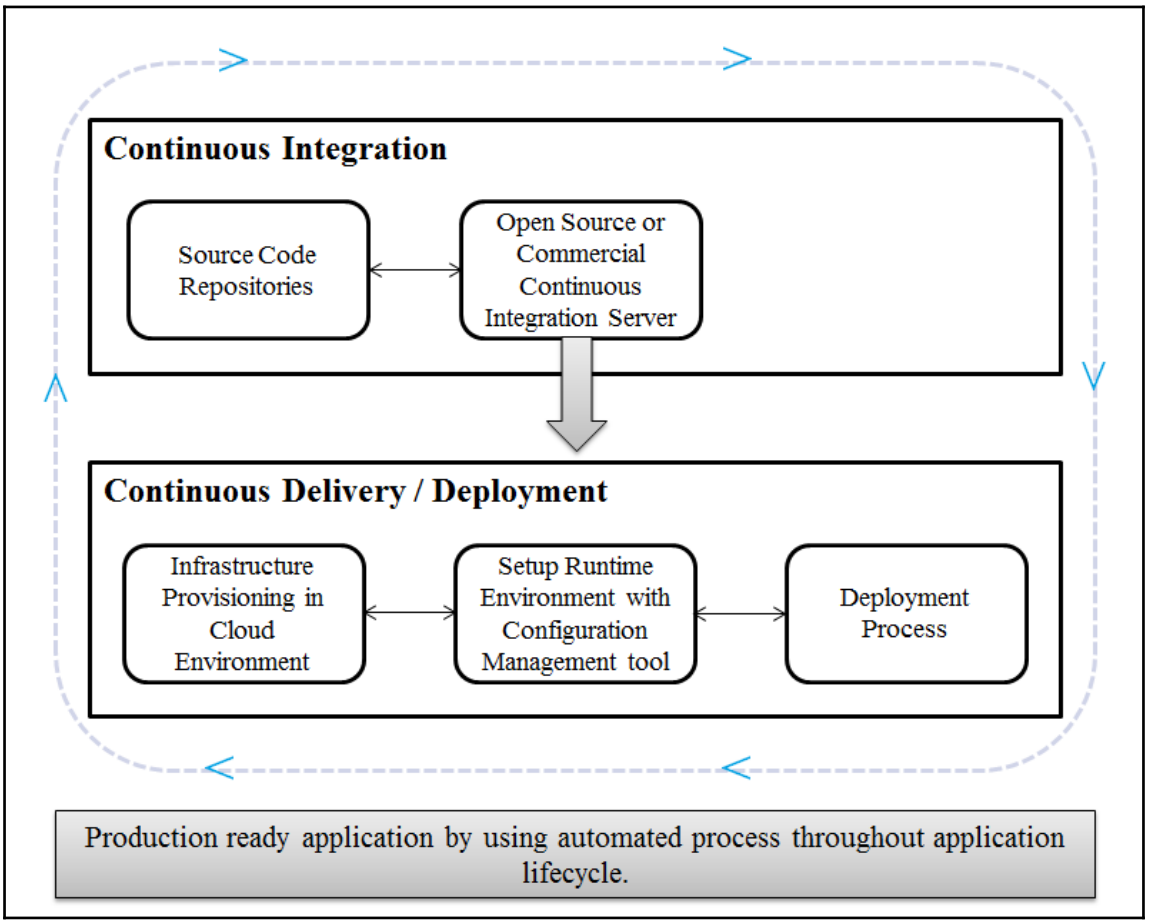
in each server. What if we need to change port in all servers or update some packages or provides rights to some users? Any kind of modifications in this scenario is a manual and error prone process if it is manual. As it is same configuration for all the servers, automation can be a key here. Automation of installation and modification in server runtime environment or permissions brings servers in desired state in effective manner.

CM is also about keeping track or versions of details related to state of specific nodes or servers. It is a far better situation when we need to change in many servers and we can push those changes to servers or all those server nodes can pull those changes and bring themselves into compliance of new policy. Centralized change can trigger this or nodes can communicate with CM server whether they need to update themselves or they are in a desired state already. CM tools makes process efficient when only changed behaviour is updated and not all installation and modification are applied again to server nodes.

There are many popular configuration management tools are available in the market such as Chef, Puppet, Ansible, Salt, and so on. Each tool are different in the way they work but characteristics and end goal is same – to bring standardized behaviour in state change of specific nodes without any errors.

Continuous delivery / continuous deployment

Continuous Delivery and Continuous Deployment are used interchangbly more often then not. However, there is a small difference between them. Continuous Delivery is a process to deploy application in any environment in automated fashion and continuous feedback to improve the quality of an application. Continuous Deployment on other hand is all about deploying application with latest changes to the production environment. In other words we can say that Continuous Deployment implies Continuous Delivery while Continuous Delivery doesn't imply Continuous Deployment.



Continuous Delivery is significant because of the incremental releases after short span of implementation or sprint in agile terms. To deploy feature ready application from development to testing may include multiple iterations in a sprint due to change in requirements or change in interpretation. However, at the end of sprint, final feature ready application is deployed into the production environment. As we discussed about multiple deployments in testing environment even in short span of time, it is advisable to have automated approach for it. Scripts to create infrastructure and runtime environment for all environments are useful. It is easier to provision resources in such environment.

For example, to deploy an application in Microsoft Azure environment we need following resources:

- Azure web app configured with specific types of resources
- Storage account to store BACPAC file to create database
- To create SQL Server to host database
- To import BACPAC file from Storage account to create a new database
- Deploy a web application into Microsoft Azure environment

In above scenario, we may consider to use configuration file for each environment with respect to naming conventions and paths. However, we need similar types of resources in each environment. It is possible that configuration of resources change according to environment but that can be managed in configuration file for each environment. Automation scripts can use configuration files based on the environment and create resources and deploy an application into it. Hence repetative steps can be easily managed by automated approach and it is helpful in Continuous Delivery and Continuous Deployment both.

Best practices for continuous delivery:

Following are some common practices we should follow to implement continuous delivery:

- Plan to automate everything in a application delivery pipeline: Consider a situation where a single commit only is required to deploy an application in the target environment. It should include compilation, unit test execution, code verification, notification, instance provisioning, setting up runtime environment, deployment of an application
    - Automate repetitive tasks
    - Automate difficult tasks
    - Automate manual tasks
        - Develop and Test newly implemented feature of bug fixing in a production like environment; it is possible now with pay per use resources provided by Cloud computing
            - Deploy frequently in Development and Test environment to gain experience and consistency

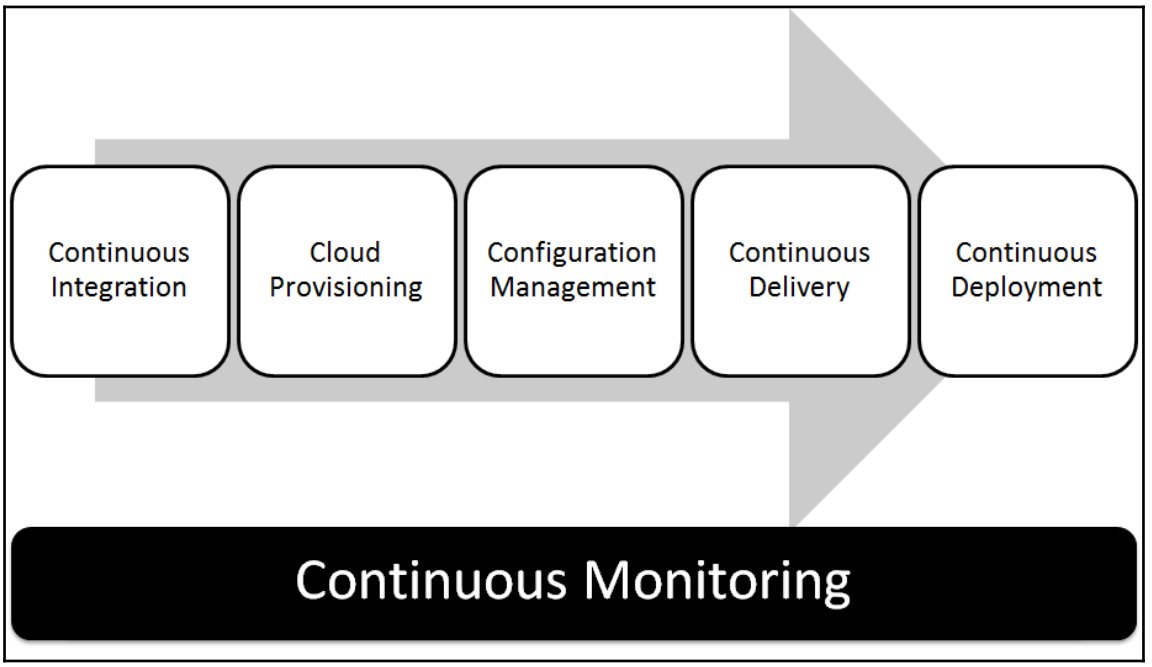
Continuous Delivery: Reliable Software Releases through Build, Test, and Deployment Automation
http://martinfowler.com/books/continuousDelivery.html

Continuous Delivery vs Continuous Deployment
http://continuousdelivery.com/2010/08/continuous-delivery-vs-continuous-deployment/

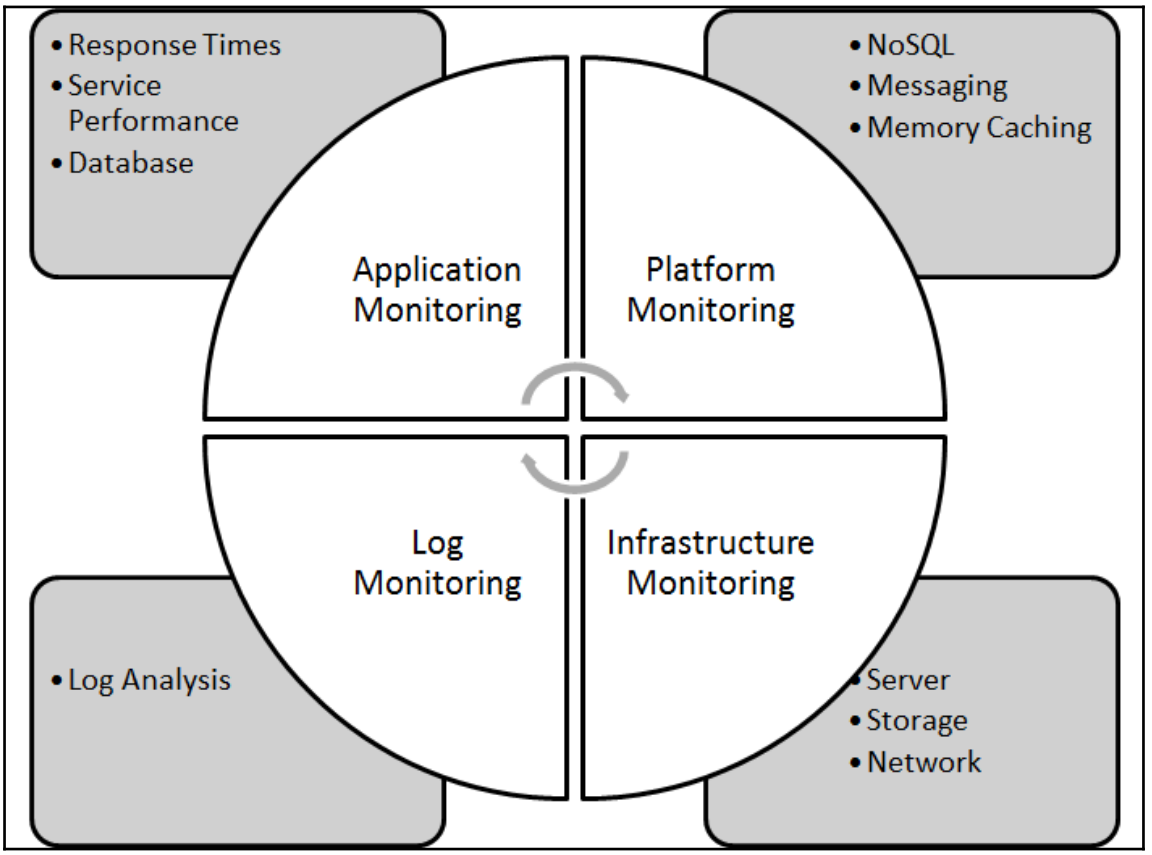
Continuous Delivery versus Continuous Deploy
http://devops.com/2015/10/30/continuous-delivery-versus-continuous-deploy/

Continuous monitoring

Continuous Monitoring is a backbone of end to end delivery pipeline and open source monitoring tools are like toppings on an ice cream scoop. It is desirable to have monitoring at almost every stage to have transparency about all the process as shown in below image. It also helps in the troubleshooting within quick time. Monitoring should be a well thought out implementation of plan that starts in the beginning itself.



There is a likely scenario where end to end deployment is implemented in automated fashion but issues arise due to coding problems, query related problems, infrastructure related issues and so on. We can consider different types of monitoring as shown in the figure.

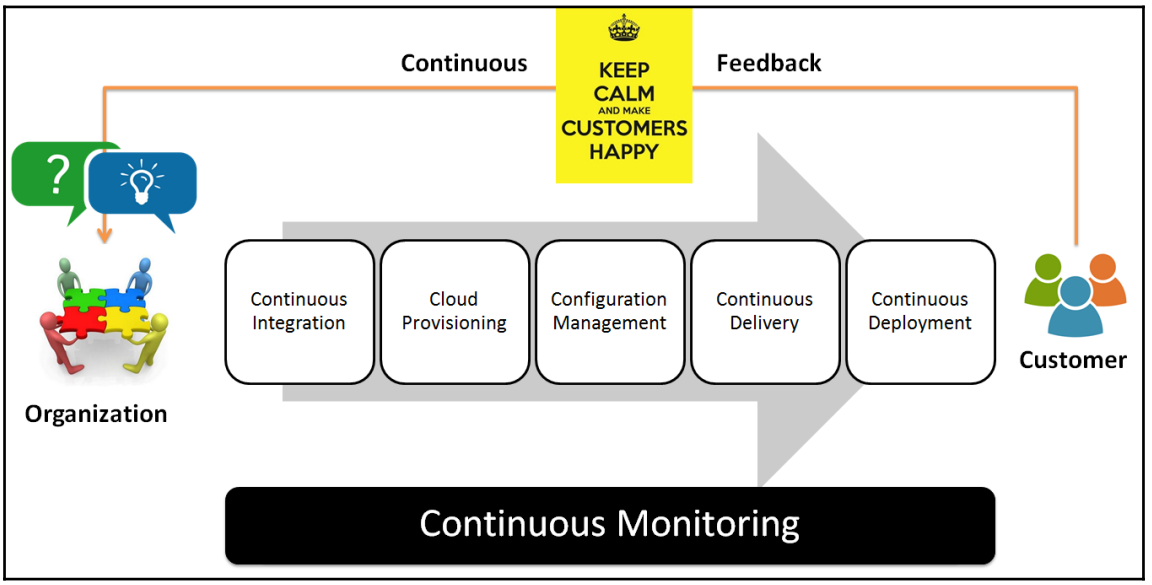


However, there is normal tendency to monitor only infrastructure resources. The question one must ask is whether it is enough or we must focus on other types of monitoring as well? To answer this question, we must have monitoring strategy in place in the planning stage only. It is always better to identify stakeholders, monitoring aspects, and so on based on culture and experience of an organization.

Continuous monitoring demystified
http://searchsecurity.techtarget.com/feature/Continuous-monitoring-demystified

Continuous feedback

Continuous feedback is the last important component in the DevOps culture that provides way for improvement and innovation. Feedback always provides way of improvement if it comes from the stakeholders who know what they need and how the outcome should be. Feedback from the customer after deployment activities can server to developers as inputs for improvement as shown in below figure and its correct integration make customer happy.



Here, we are considering situation where feature implementation is provided to the stakeholders and they provide their feedback. In waterfall model, feedback cycle is very long and hence developers may not be aware about whether the end product is what customer asked for or interpretation of what needs to be delivered is changed somewhere. In Agile or DevOps culture, shorter feedback cycle is major difference as stakeholders can actually see the end result of small implementation phase and hence outcome is verified multiple times. If customer is not satisfied then feedback is available at a stage where it is not much painful to change things. In waterfall model it was a disaster as feedback used to come very late. With time and dependencies, complexities increases and changes in such situation takes long time. In addition to it, none remembers what they wrote 2 months back. Hence, faster feedback cycle improves overall process and also connects end points as well as finding patterns in mistakes, learning lessons, and using improved patterns. However, continuous feedback not only improves technical aspects of implementation but it also

provides way to assess current features and whether they fits into overall scenario or still there is a room of improvement. It is important to realize that Continuous feedback plays significant role in making customers happy by providing improved experience.

Tools and technologies

Tools and technologies play important role in the DevOps culture however it is not the only part that needs attention. For all parts of application delivery pipeline, different tools, disruptive innovations, open source initiatives, community plugins and so on are required to keep the entire pipeline running to produce effective outcomes.

Code repositories – Git

Subversion is a version control system that is used to track all the changes made to files and folders. By this a track can be kept on the applications which are being built. The features added months ago can also be tracked using the version code. It is all about tracking the code. Whenever any new features are added or any new code is made, it is first tested and then it is committed by the developer. Now the code is sent to the repository to track the changes and a new version is given to it. A comment can also be made by the developer so that other developer can easily understand the changes that are made. Other developers only have to update their checkout to see the changes made.

Advantages

Following are some advantages of using source code repositories:

- No. of developers can work simultaneously on the same code
- If a computer crashes still the code can be recovered as it was committed in the server
- If a bug occurs the new code can be easily reverted back to the previous version

Git is an open source distributed version control system which is designed to handle from small to very large projects with speed and efficiency. It is easy to learn and has a good performance. It comprises of full-fledged repository and version control tracking capabilities independent of central server or a network access. It was developed and designed by Linux Torvalds in 2005.

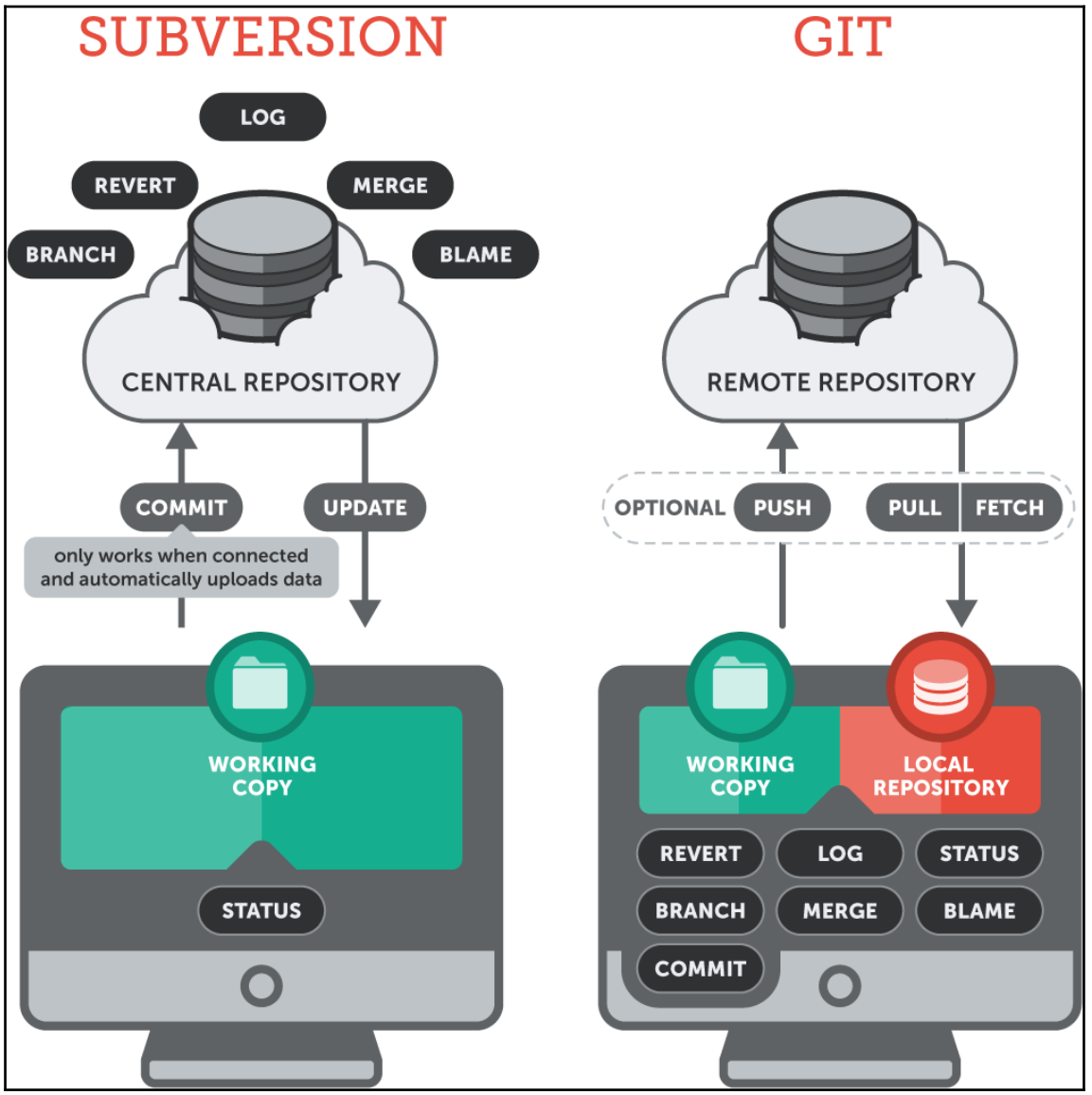
Characteristics

Following are some significant characteristics of Git:

- It provides support for non-linear development
- It is compatible with existing systems or protocols
- It ensures cryptographic authentication of history
- It has well designed pluggable merge strategies
- It consists of tool-kit based designs
- It supports various merging techniques such as Resolve, Octopus, and Recursive

Differences between SVN and Git

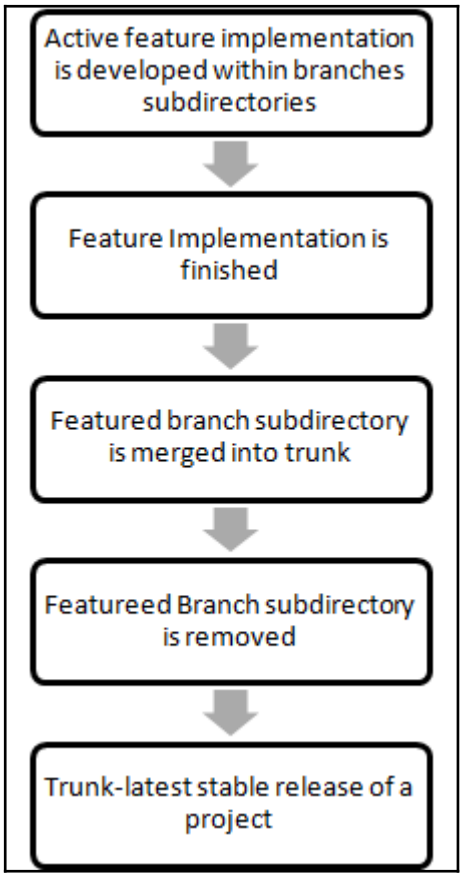
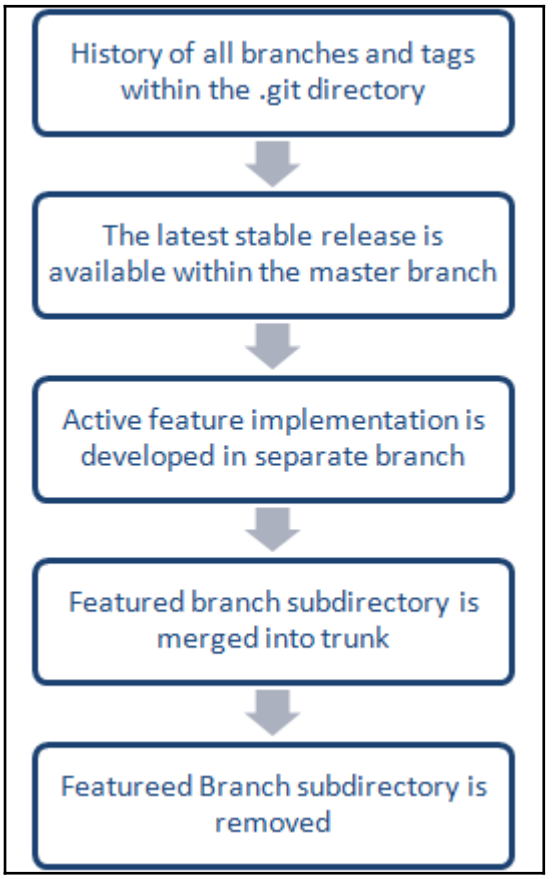
SVN and Git both are very popular source code repositories, however Git is getting more popular in recent times. Let's see the major differences between them both. The following figure shows visual difference between SVN and Git.



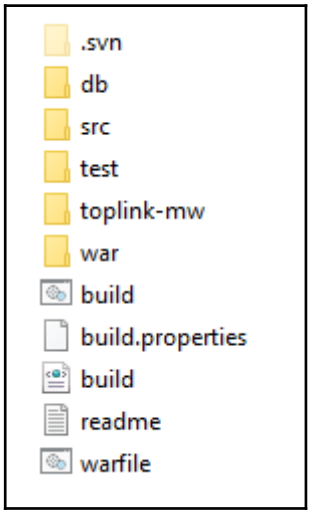
(Reference: https://www.git-tower.com/learn/git/ebook/en/mac/appendix/from-subversion-to-git)

Subversion	**Git**

Centralized Version Control System	Distributed Version Control System
Snapshot of a specific version of the project is available on developer's machine.	Complete clone of a full-fledged Repository is available on developer's machine.
Perform operations such as commit, merge, blame, revert and so on; verify branch and log from a central repository.	Perform operations such as commit, merge, blame, and so on; verify branch and log from a local repository. Pull and Push operation to remote repository if developer needs to share work with others.
URLs are used to trunk, branches, or tags Example of Repository URL: https://<URL/IP Address>/svn/trunk/AntExample1/	.git is the root of project and commands are used to address branches and not URLs Example of Repository URL: git@github.com:mitesh51/game-of-life.git

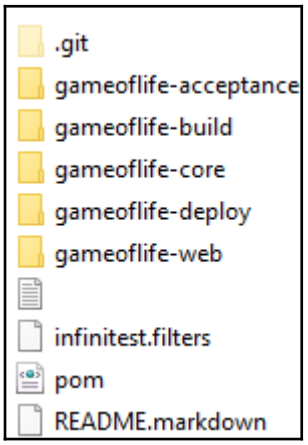
A SVN Workflow:	A Git Workflow:
	
B05561_01_16	B05561_01_17
File changes are included in the next commit.	File changes has to be marked explicitly and then only they are included in the next commit.
Committed work is directly transferred to central repository and hence direct connection to repository must be available.	Committed work is directly transferred to remote repository. It is committed to local repository. To share it with other developers, we need to push it to remote repository and in this case we need a connection to remote repository.
Each commit gets ascending revision number	Each commit gets commit hashes rather than ascending revision number

Application Directory:



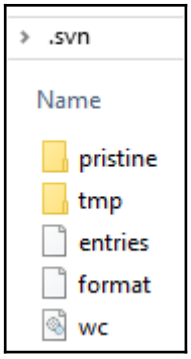
B05561_01_18

Application Directory:



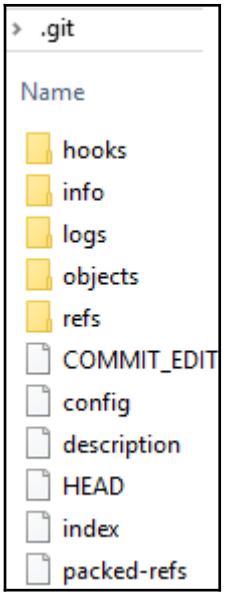
B05561_01_19

.svn directory structure:



B05561_01_20

.git directory structure:



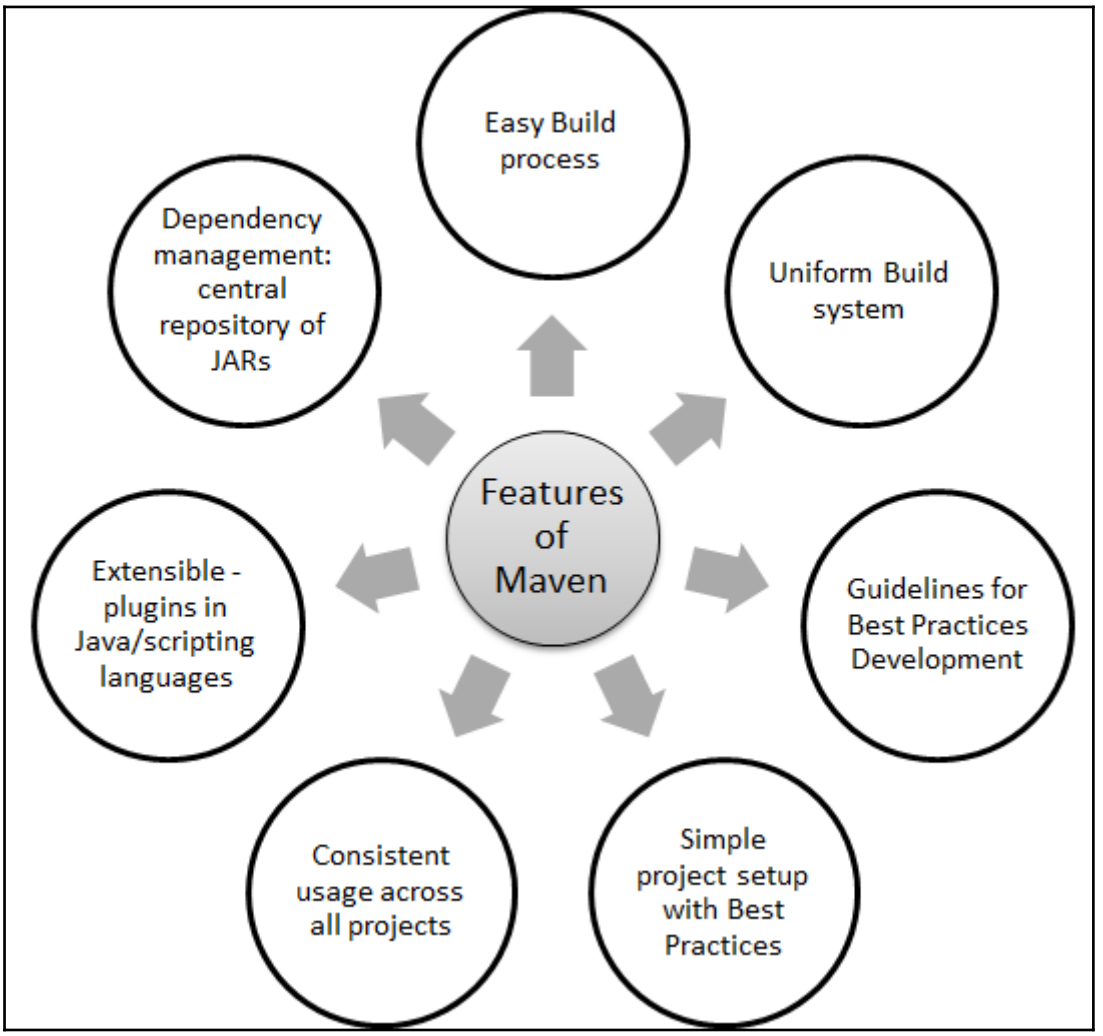
B05561_01_21

Short Learning Curve	Long Learning Curve

Build tools – Maven

Apache Maven is build tool having Apache License 2.0 license. It is used for Java projects and it can be used in cross platform environment. However it can be used for Ruby, Scala, C#, and other languages.

The following are the important features of Maven:



A Project Object Model (POM)- XML file, contains information on name of the application, owner information, how application distribution file can be created, how dependencies can be managed.

Example of POM.XML

POM.XML has pre-defined targets such as validate, generate-sources, process-sources, generate-resources, process-resources, compile, process-test-sources, process-test-resources,

test-compile, test, package, install, and deploy.

Following is an example of sample pom.xml file that is used in Maven:x

```
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
http://maven.apache.org/maven-v4_0_0.xsd">
    <modelVersion>4.0.0</modelVersion>
    <groupId>org.springframework.samples</groupId>
    <artifactId>spring-petclinic</artifactId>
    <version>1.0.0-SNAPSHOT</version>
    <name>petclinic</name>
    <packaging>war</packaging>
    <properties>

    <dependencies>
    <build>
        <defaultGoal>install</defaultGoal>
        <testResources>
          <testResource>
        <!-- Spring config files, next to corresponding JUnit test class  -->
            <directory>${project.basedir}/src/test/java</directory>
          </testResource>
          <testResource>
            <directory>${project.basedir}/src/test/resources</directory>
          </testResource>
        </testResources>
        <plugins>
    </build>
        <reporting>

    <url>demopetclinic</url>
</project>
```

Continuous integration tools – Jenkins

Jenkins is originally open source continuous integration software written in Java having MIT License. However, Jenkins 2 an open source automation server that focuses on any automation including continuous integration and continuous delivery.

Jenkins can be used across different platforms such as Windows Ubuntu/Debian, Red Hat/Fedora, Mac OS X, openSUSE, and FreeBSD. Jenkins enables user to utilize continuous

integration services for software development in agile environment. It can be used to build free style software project based on Apache Ant and Maven 2/ Maven 3 Project. It can also execute Windows batch commands and shell scripts.

It can be easily customized with the use of plug-ins. There are different kinds of plug-ins available to customize Jenkins based on specific needs. Categories of plug-ins include Source code management (that is, Git Plugin, CVS Plugin, Bazaar Plugin), build triggers (i.e. Accelerated Build Now Plugin, Build Flow Plug-in), Build reports (that is, CodeScanner Plug-in, Disk Usage Plug-in), Authentication and user management (i.e. Active Directory plug-in, Github OAuth Plug-in), Cluster management and distributed build (that is, Amazon EC2 Plugin, Azure Slave Plugin), and so on.

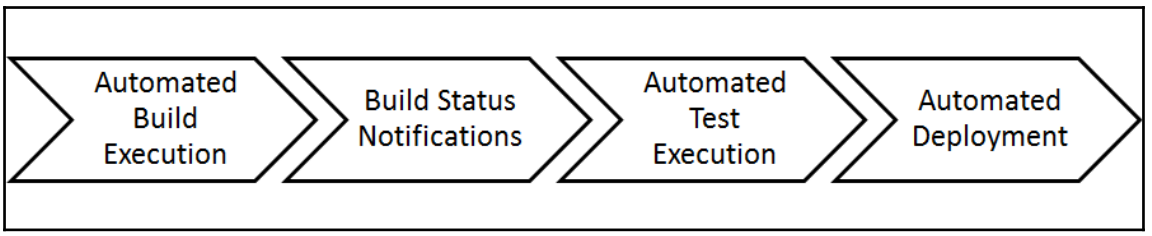
To know more about all plugins, visit
https://wiki.jenkins-ci.org/display/JENKINS/Plugins

To explore on how to create a new plugin, visit
https://wiki.jenkins-ci.org/display/JENKINS/Plugin+tutorial

To download different versions of plugins, visit
https://updates.jenkins-ci.org/download/plugins/

Continuous Integration Server: Jenkins http://jenkins.io/

Jenkins accelerates the software development process through automation



Key Features and Benefits

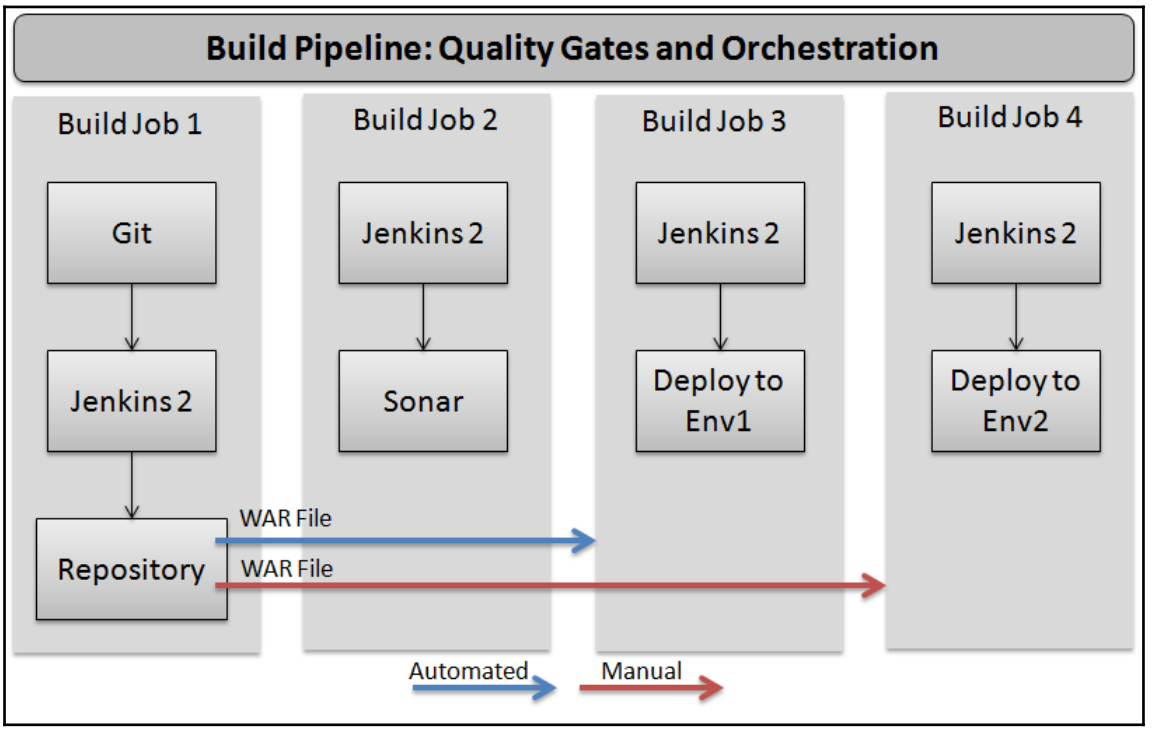
Following are some striking benefits of Jenkins:

- Easy install, easy upgrade, easy configuration
- Supported Platforms: Windows, Ubuntu/Debian, Red Hat/Fedora/CentOS, Mac OS X, openSUSE, FreeBSD, OpenBSD, Solaris, Gentoo
- Manages and controls development lifecycle processes
- Non java projects supported by Jenkins: .Net, Ruby, PHP, Drupal, Perl, C++, Node.js, Python, Android, Scala
- A development methodology of daily integrations verified by automated builds
- Every commit can trigger a build
- Jenkins is a fully featured technology platform that enables users to implement Continuous Integration (CI) and Continuous Delivery (CD)
- Use of Jenkins is not limited from continuous integration (CI) to continuous delivery (CD). It is possible to include model and orchestrate entire pipeline with the use of Jenkins as it supports shell and windows batch commands execution. Jenkins 2.0 supports delivery pipeline that uses a domain-specific language (DSL) for modeling entire deployment or delivery pipeline
- Pipeline as code provides a common language-DSL to help development and operations teams to collaborate in effective manner
- Jenkins 2 brings new GUI with stage view to observe the progress across delivery pipeline
- Jenkins 2.0 is fully backward compatible to Jenkins 1.x series of releases
- Jenkins 2 now requires Servlet 3.1 to run
- Use embedded Winstone-Jetty or use container that supports Servlet 3.1 (for example, Tomcat 8)
- GitHub, Collabnet, SVN, TFS code repositories, and so on are supported by Jenkins for Collaborative Development
- Continuous Integration: Automate build, test – automated testing (Continuous Testing), package, and static code analysis
- Supports common test frameworks such as HP ALM Tools, Junit, Selenium, MSTest, and so on
- For Continuous Testing, Jenkins has plugins for both; Jenkins slaves can execute the test suites on different platforms
- Jenkins supports static code analysis tools such as Code Verification support by Checkstyle and Findbug. It also integrates with Sonar
- Continuous Delivery and Continuous Deployment: automates the application

deployment pipeline; Integration with popular configuration management tools, and automated environment provisioning

- To achieve continuous delivery and deployment, Jenkins Supports Automatic Deployment; it provides plug-in for direct integration with IBM uDeploy
- Highly configurable tool-plugins-based architecture that provides support to many technology, repositories, build tools, and test tools; an Open-source CI server and provides 400+ plugins to achieve extensibility
- Supports Distributed builds: Jenkins supports the "master/slave" mode, where the workload of building projects are delegated to multiple "slave" nodes
- Machine-consumable remote access API to retrieve information from Jenkins for programmatic consumption, to trigger a new build, and so on
- Deliver better application faster by automating the application development lifecycle allowing faster delivery

Jenkins – Build Pipeline (Quality Gates) provides Build Pipeline View of upstream and downstream connected jobs: Chain of jobs each one subjecting build to quality assurance steps. It has the ability to define manual triggers for jobs that require intervention prior to execution such as an approval process outside of Jenkins.



Jenkins can be used with following tools in different categories.

Language	**Java**	**.Net**
Code Repositories	Subversion, Git, CVS, StarTeam	
Build Tools	Ant, Maven	NAnt, MS Build
Code Analysis Tools	Sonar, CheckStyle, Findbugs, Ncover, Visual Studio Code Metrics PowerTool	
Continuous Integration	Jenkins	

Continuous Testing	Jenkins Plugins (HP Quality Center 10.00, with the QuickTest Professional Add-in, HP Unified Functional Testing 11.5x and 12.0x, HP Service Test 11.20 and 11.50, HP LoadRunner 11.52 and 12.0x, HP Performance Center 12.xx, HP QuickTest Professional 11.00, HP Application Lifecycle Management 11.00, 11.52, and 12.xx, HP ALM Lab Management 11.50, 11.52, and 12.xx), Junit, MSTest, VsTest)
Infrastructure Provisioning	Configuration Management Tool – Chef
Virtualization / Cloud Service Provider	VMware, AWS – Amazon Web Services, Microsoft Azure (IaaS), Traditional Environment
Continuous Delivery / Deployment	Chef / Deployment Plugin / Shell Scripting / Powershell Scripts / Windows Batch Commands

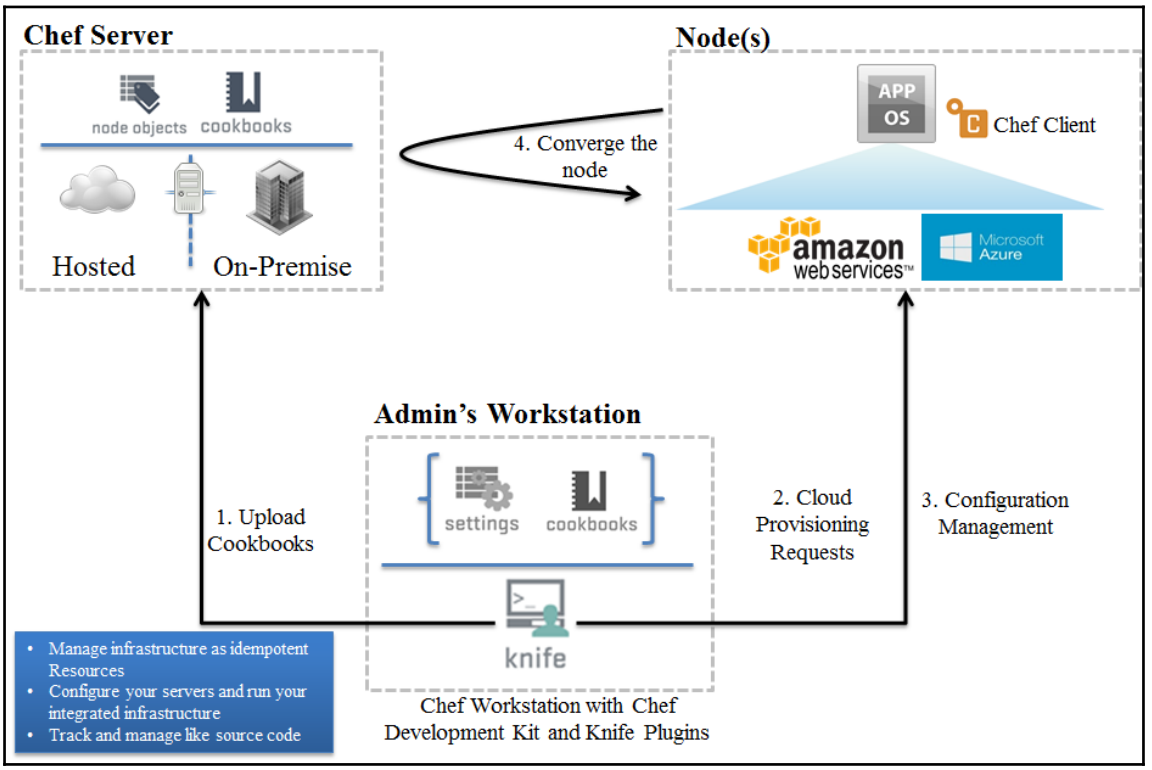
Configuration management tools – Chef

Software Configuration Management (SCM) is a software engineering discipline comprising of tools and techniques that an organizations used to manage the changes in software components. It includes technical aspects of the project, communication, and control of modifications changes to the projects during development phase. It also called as Software Control Management. It constitutes of practices for all software projects ranging from development, rapid prototyping, or ongoing maintenance. It enriches the reliability and quality of software.

Chef is a configuration management tool which is used to transform the infrastructure into code. It automates building, deploying, and managing of the infrastructure. By using chef, infrastructure can be considered as a code. Concept behind chef is of reusability. It uses recipes to automate the infrastructure. Recipes are instructions required for configuring databases, web servers, and load balances. It describes every part of the infrastructure and how it should be configured, deployed, and managed. It uses building blocks known as resources. A resource describes parts of infrastructure such as template, package, and files to be installed.

This recipes and configuration data are stored in Chef Servers. Chef Client is installed on each node of the network. A node can be physical or virtual server.

The Chef client periodically checks the Chef server for the latest recipes and to see if the node is in compliance with the policy defined by the recipes. If it is out of date, the Chef client runs them on the node to bring it up to date.



Features

Following are some important features of Chef Configuration Management Tool:

- Chef Server
  - It manages a huge amount of the nodes
  - It maintains a blueprint of the infrastructure
    - Chef Client
      - It manages various operating systems such as Linux, Windows, Mac OS, Solaris, and FreeBSD
      - It provides integration with cloud providers.
        - It is easy to manage the

containers in a version-able, testable, and repeatable way

- Chef provides automation Platform to continuously define, build, and manage Cloud infrastructure that is used for deployment
- It enables resource provisioning and configuration of resources programmatically and it will help in the deployment pipeline to automate provisioning and configuration

Three basic concepts of Chef will enable organizations to quickly manage any infrastructure with Chef:

- Achieving desired state
- Centralized modeling of IT infrastructure
- Resource primitives that serve as building blocks

Chef Configuration Management tool https://www.chef.io/

Chef Management Console https://manage.chef.io/login

Cloud service providers

AWS and Microsoft Azure are popular public cloud providers in recent times. They provide cloud services in different areas and both have they strong areas. Based on the organization culture and past partnership anyone can be considered after detailed assessment based on requirements.

Followings are side by side comparison in terms of services:

	AWS	**Microsoft Azure**
Virtual Machines	Amazon EC2	Virtual Machine
PaaS	Elastic Beanstalk	Azure Web Apps
Container Services	Amazon EC2 Container Services	Azure Container Services
RDBMS	Amazon RDS	Azure SQL Database
NoSQL	DynamoDB	DocumentDB
BIG Data	Amazon EMR	HD Insight
Networking	Amazon VPC	Virtual Network
Cache	Amazon Elasticache	Azure RadisCache
Import/Export	Amazon Import/Export	Azure Import/Export
Search	Amazon CloudSearch	Azure Search
CDN	CloudFront	Azure CDN
Identity and Access Management	AWS IAM and Directory Services	Azure Active Directory
Automation	AWS OpsWorks	Azure Automation

Amazon Web Services http://aws.amazon.com/

Microsoft Azure https://azure.microsoft.com

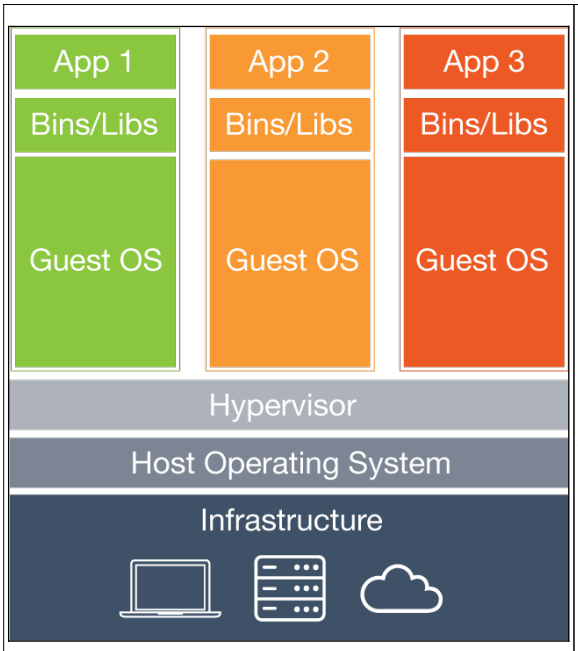
Container technology

Containers use OS level virtualization where kernel is shared between isolated user spaces. Docker and OpenVZ are popular open source example of operating system-level virtualization technology.

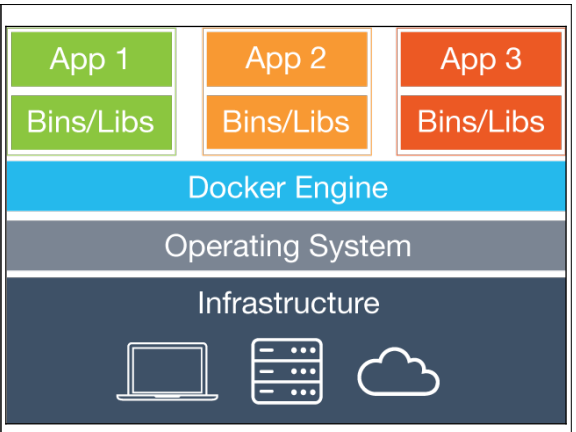
Docker

Docker is an open source initiative to wrap code, runtime environment, system tools, and libraries. Docker containers share the kernel they are running on and hence they start instantly and lightweight in nature. Docker containers runs on Microsoft operating systems and Linux distributions. It is important to understand how containers and virtual machines are different. Below is the comparison table of Virtual machines and Containers.

Virtual Machine	Docker Container



B05561_01_27 (www.docker.com)



B05561_01_28 (www.docker.com)

Virtual machines depend on the traditional virtualization. It can be considered as Hardware level Virtualization.	Container depends on containerization technique at a kernel level. It can be considered as Operating system level Virtualization.
Each virtual machine contains Guest operating system, Binaries, and library files and application itself.	Each container include application, Binaries, and library files but the major difference compared to virtual machine is the shared kernal; each container runs as isolated process in user space on Host OS.
Size of the each virtual machine is in GBs as each one runs on its own.	As each container runs as isolated process in user space on Host OS and separate operating system is not required for each container, size of each container is much less.
Each virtual machine has its own set of resources – better isolation and less sharing of resources.	Each container share kernel and hence more scope of sharing resources.
Virtual Machine can not run on Container.	Container can run on Virtual machine.

Docker – the open-source application container engine
https://github.com/docker/docker

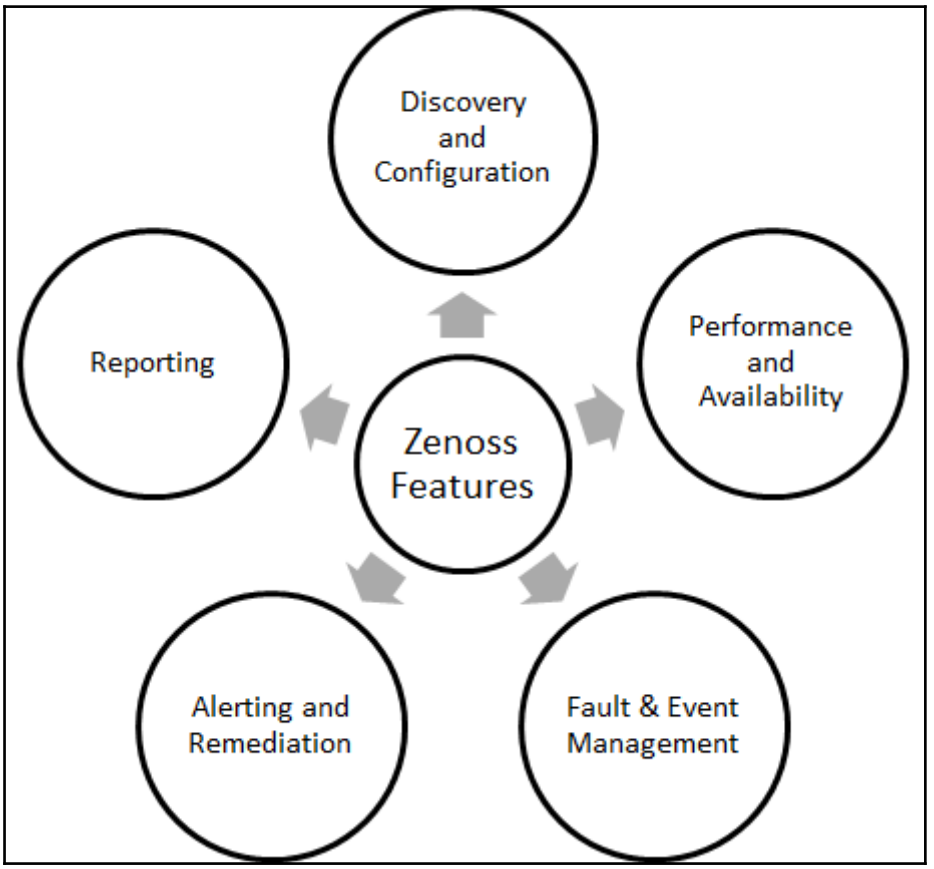
Monitoring tools

There are many open source tools are available for monitoring resources. Zenoss and Nagios are one of the most popular open source tools adopted by many organizations.

Zenoss

Zenoss is an agent less and open source management platform for application, server, and

network released under the GNU General Public License (GPL) version 2 based on the Zope application server. Zenoss Core consists of extensible programming language Python, object-oriented web server Zope Application server, Monitoring protocol Net, Graph and log time series data by RRDtool, MySQL, and event-driven networking engine Twisted. It provides easy to use web portal to monitor alerts, performance, configuration, and inventory.



Zenoss Core 5 http://www.zenoss.org/

Nagios

Nagios is a cross platform and open source monitoring tool for infrastructure and network. It monitors network services such as FTP, HTTP, SSH, and SMTP. It monitors resources, detects the problems, and alerts the stakeholders. Nagios can empower organizations and service providers to identify and resolve issues in a way that outages have minimal impact on IT infrastructure and processes hence highest compliance to SLAs. Nagios can monitor cloud resources such as compute, storage, and network.

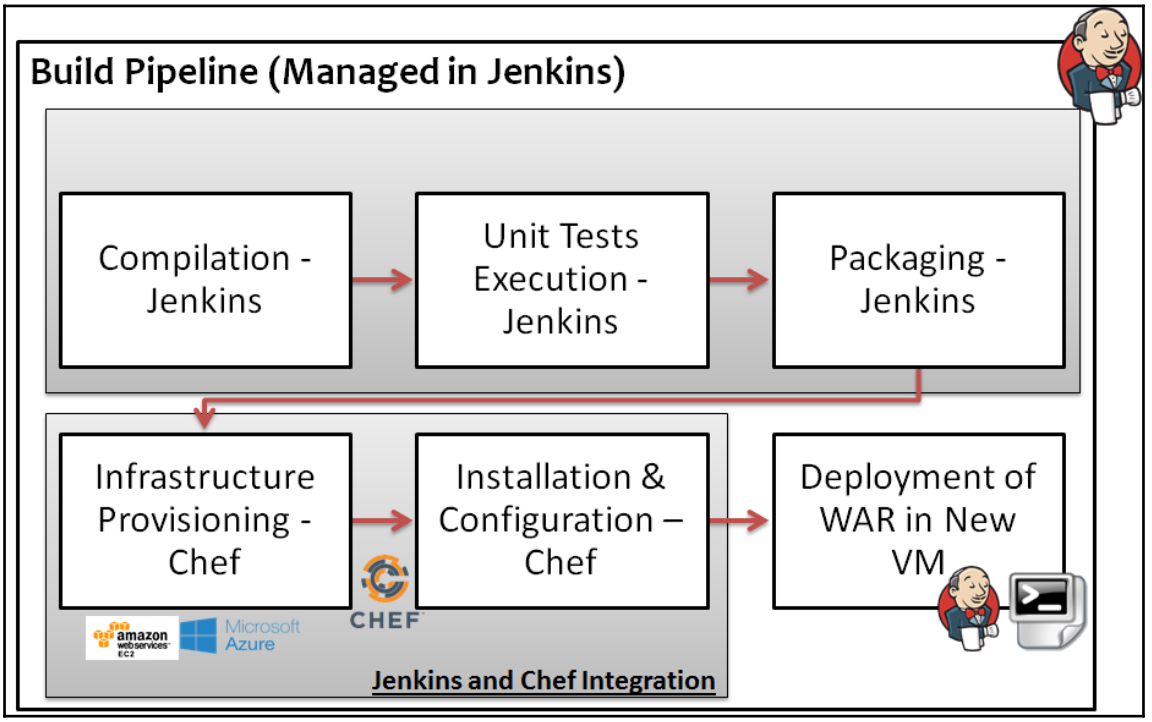
The Industry Standard In IT Infrastructure Monitoring
https://www.nagios.org/

Deployment Orchestration / Continuous Delivery – Jenkins

Build pipeline or Deployment Pipeline or Application Delivery pipeline, can be used to achieve end to end automation for all operations. Starting from Continuous Integration, Cloud Provisioning, Configuration Management, Continuous Delivery, Continuous Deployment, and Notifications. Jenkins plugins can be used for overall orchestration of all activities involved in end to end automation.

- Continuous Integration: Jenkins
- Configuration Management: Chef
- Cloud Service Providers: AWS, Microsoft Azure
- Container Technology: Docker
- Continuous Delivery / Deployment: ssh
- End to End Orchestration: Jenkins Plugins

Following is a sample representation of end to end automation using different tools:



Jenkins can be used to manage unit testing, code verification; Chef can be used for setting up runtime environment; knife plugins can be used for creating a virtual machine in AWS or in Microsoft Azure; Build pipeline or Deployment pipeline plugin in Jenkins can be used for managing deployment orchestration.

From a single pipeline dashboard, we can view status of all builds which are configured in pipeline. Each build in the pipeline is a kind of quality gate. If one build fails then execution won't go further. Another dimensions can be added such as notification based on compilation failures, unit test failure or for unsuccessful deployment. Final deployment can be based on some sort of permission from a specific stakeholder. We can consider a scenario for parameterized build or promoted build concept. What we will do? Wait for upcoming chapters and all secretes will be revealed.

DevOps Dashboard

One of the most desired components to get into DevOps culture is Dashboard functionality or GUI that provides combined status of all end to end activities. For automation tools, easy

to use web GUI is handful for management of resources. For end to end automation in application deployment activity, multiple open source or commercial tools are used. There is a high possibility where single product may not be used for all activities. For example, Git or SVN as repository, Jenkins as CI server, IBM Urbancode Deploy as deployment orchestration tool. In such scenario, it is easier if there is single pane of glass view where we can track multiple tools for a specific application.

Hygieia is an open source DevOps dashboard that provides way to track status of deployment pipeline. It basically tracks 6 different areas as of now including features (Jira, VersionOne), code repo (GitHub, Subversion), build (Jenkins, Hudson), quality (Sonar, Cucumber/Selenium), monitoring, and deployment (IBM UrbanCode Deploy)

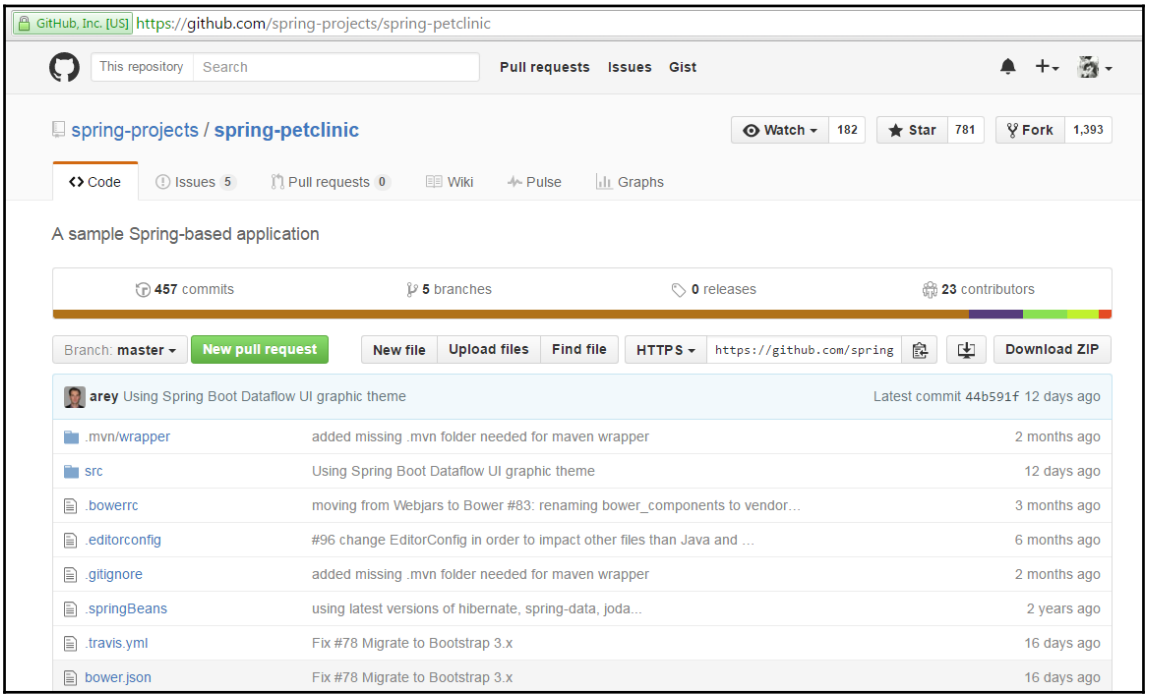
CapitalOne DevOps Dashboard https://github.com/capitalone/Hygieia

Overview of Sample JEE Application

We are going to use PetClinic Application available on GitHub.

A sample Spring-based application
https://github.com/spring-projects/spring-petclinic

The PetClinic sample application can be used to build simple and robust database-oriented applications to demonstrate the use of Spring's core functionality. It is accessible via web browser.



A few use cases:

- Add a new pet owner, a new pet, and information pertaining to a visit to the pet's visitation history to the system
- Update the information pertaining to a pet and a pet owner
- View a list of veterinarians and their specialties, a pet owner, a pet, and pet's visitation history

Once WAR file is created, we can deploy it in Tomcat or another web server and to verify it in localhost, visit http://localhost:8080/petclinic. We will see something like:

List of Tasks

Following are the tasks we will try to complete in rest of the chapters:

- Jenkins Installation, Configuration, UI Personalization
- Java configuration (JAVA_HOME) in Jenkins
- Maven or Ant configuration in Jenkins
- Plugins installation and configuration in Jenkins
- Security (Access Control, Authorization, Project based Security) in Jenkins
- Jenkins build configuration and Execution
- Email Notification Configuration
- Deployment of WAR file to Web Application Server
- Eclipse Integration with source code repository
- Create and configure a Build / Deployment Pipeline

- Install and Configure Chef – Configuration Management tool
- Install and Configure Docker
- Create and configure Virtual machine in AWS and Microsoft Azure and containers
- Deploy WAR file into Virtual Machine and Container
- Configure Infrastructure monitoring
- Orchestrate Application delivery pipeline using Jenkins plugins

Self-Test Questions

1. Which of the following statement is not related to Development team in Traditional Environment?

1. Competitive Market creates pressure of on time delivery of feature or bug fixing

2. Production ready Code Management and New feature Implementation

3. Release cycle is often long and hence development team has to make assumptions before the application deployment finally takes place

4. Redesigning or tweaking is needed to run the application in Production environment

1. Which of the following are benefits of DevOps?

5. Collaboration, Management, and security for the complete application development lifecycle management

6. Continuous innovation because of continuous development of new ideas

7. Faster delivery of new features or resolution of issues

8. Automated deployments and standardized configuration management for different environments

9. All of the above

1. Which of the following are parts of DevOps culture or Application Delivery Pipeline?

10. Continuous Integration

11. Cloud Provisioning
12. Configuration Management
13. Continuous Delivery / Deployment
14. Continuous Monitoring
15. Continuous Feedback

    1. Which of the following are by product of DevOps culture or Application Delivery Pipeline?

16. Continuous Integration
17. Continuous Delivery / Deployment
18. Continuous Monitoring
19. Continuous Feedback
20. Continuous Improvement
21. Continuous Innovation

    1. State whether following statement is True or False: Jenkins and Atlassian Bamboo are a build automation tool.

22. True
23. False

    1. State whether following statement is True or False: Apache Ant and Apache Maven are Continuous Integration Tools

24. True
25. False

    1. State whether following statement is True or False: Chef is a configuration management tool.

26. True
27. False

1. State whether following statement is True or False: Build automation is essential for Continuous Integration and rest of the automation is effective only if build process is automated.

28. True

29. False

1. State whether following statement is True or False: Subversion is a Distributed Version Control System.

30. True

31. False

1. State whether following statement is True or False: Git is a Centralized Version Control System.

32. True

33. False

1. Which of the followings are Cloud Deployment Models according to NIST's definition of Cloud Computing?

34. Public Cloud

35. Private Cloud

36. Community Cloud

37. Hybrid Cloud

38. All of the above

1. Which of the followings are Cloud Service Models according to NIST's definition of Cloud Computing?

39. Software as a Service

40. Platform as a Service

41. Infrastructure as a Service

42. All of the above

1. State whether following statement is True or False: AWS and Microsoft Azure are Public Cloud Service Providers

43. True

44. False

1. Which of the following are main component of Chef Installation?

45. Chef Server / Hosted Chef

46. Chef Workstation

47. Nodes

48. All of the Above

Summary

In this chapter, we have learnt about difficulties faced by development and operations team in traditional environment and how Agile helps in such scenario. What has changed after arrival of agile development methodology and what challenges it brought with its arrival? We have covered important aspects of DevOps culture including Continuous Integration and Continuous Delivery. We also covered details regarding Cloud computing and Configuration Management that enhances the processes and helps to adopt DevOps culture.

In terms of tools and technologies, we have covered brief overview of SVN, Git, Apache Maven, Jenkins, AWS, Microsoft Azure, Chef, Nagios, Zenoss, and Hygieia-DevOps Dashboard.

In the next chapter, we will see how to install and configure Jenkins and how to implement Continuous Integration best practices by using sample spring application available on Github.

It is a right time to quote Charles Darwin as it is relevant in the context of DevOps culture:

It is not the most intellectual or the strongest species that survives, but the species that survives is the one that is able to adapt to or adjust best to the changing environment in which it finds itself.

2
Continuous Integration with Jenkins 2

"The way to get started is to quit talking and begin doing." – *Walt Disney*

Jenkins 2 has already arrived. Jenkins 2 comes with Built-in support for delivery pipelines, Improved usability – a new setup experience and total backwards compatibility with existing Jenkins installations. For the last time, we are using Jenkins 2 in this book.

This chapter describes in detail how Jenkins plays an important role in Continuous Integration. It covers how to prepare runtime environment for application lifecycle management and configure it with Jenkins. It manages all aspects of running a build to create a distribution file or war file for deployment by integrating source code repository such as SVN / Git for Sample JEE application. Jenkins 2 is recently made available for general usage and we have used Jenkins 2 in this book.

Readers will learn how to install and configure Jenkins and they will be able to get end to end experience from build job creation, configuration of build job, static code analysis, notifications, Jenkins plugins etc. and details on what exactly the sample application is all about.

In this chapter, we will cover the following topics:

- Jenkins – Introduction
- Jenkins installation with plugins
- Java, Maven/Ant, configuration in Jenkins
- Create and configure build job for Java application with Maven
- Dashboard view plugin – overview and usage
- Email notifications based on build status

- Jenkins and Sonar integration

Introduction

We all know what **Continuous Integration (CI)** is, right? It is the first step in our journey.

> *"The journey of a thousand miles begins with one step." Lao Tzu father of Taoism*

In simple words, CI is a software engineering practice where each check-in by a developer is verified by

- Pull mechanism: executing automated build at a scheduled time or
- Push mechanism: executing automated build when changes are saved in repository and
- Executing unit test against latest changes available in source code repository.

Jenkins doesn't need an introduction; still it is an open source and one the most popular Continuous Integrations tools available in the market. It helps in automating repetitive task of continuous integration. Jenkins makes the process effective and transparent.

> *"We are what we repeatedly do. Excellence, then, is not an act, but a habit." – Aristotle*

Next question you may ask is what makes Jenkins so popular? I have already given one reason; can you recollect?

Yes, because it is open source; however, open source tools come with predefined notions but Jenkins community is different and Jenkins as a tool is quite different.

So, what are the other reasons for popularity of Jenkins? Let's have a look:

- Written in Java
- Extensibility: 400+ plugins for different integrations are available
    - Source code management
    - Build triggers
    - Build reports
    - Artifact uploaders
    - External site/tool integrations
    - UI plugins
    - Authentication and user management
    - Cluster management and distributed build

- Others
  - Supports Java, .NET, Ruby, Groovy, Grails, PHP, Android, iOS Applications
  - Easy to use
    - Easy Installation
    - Easy Configuration
      - Simple learning curve
      - User Interface was already simple, now improved after Jenkins 2 is available for General Availability

Installing Jenkins

Jenkins provides multiple ways to install it for all types of users. We can install Jenkins on following operating systems of platforms:

- Ubuntu/Debian
- Windows
- Mac OS X
- OpenBSD
- FreeBSD
- OpenSUSE
- Gentoo
- CentOS/Fedora/Red Hat

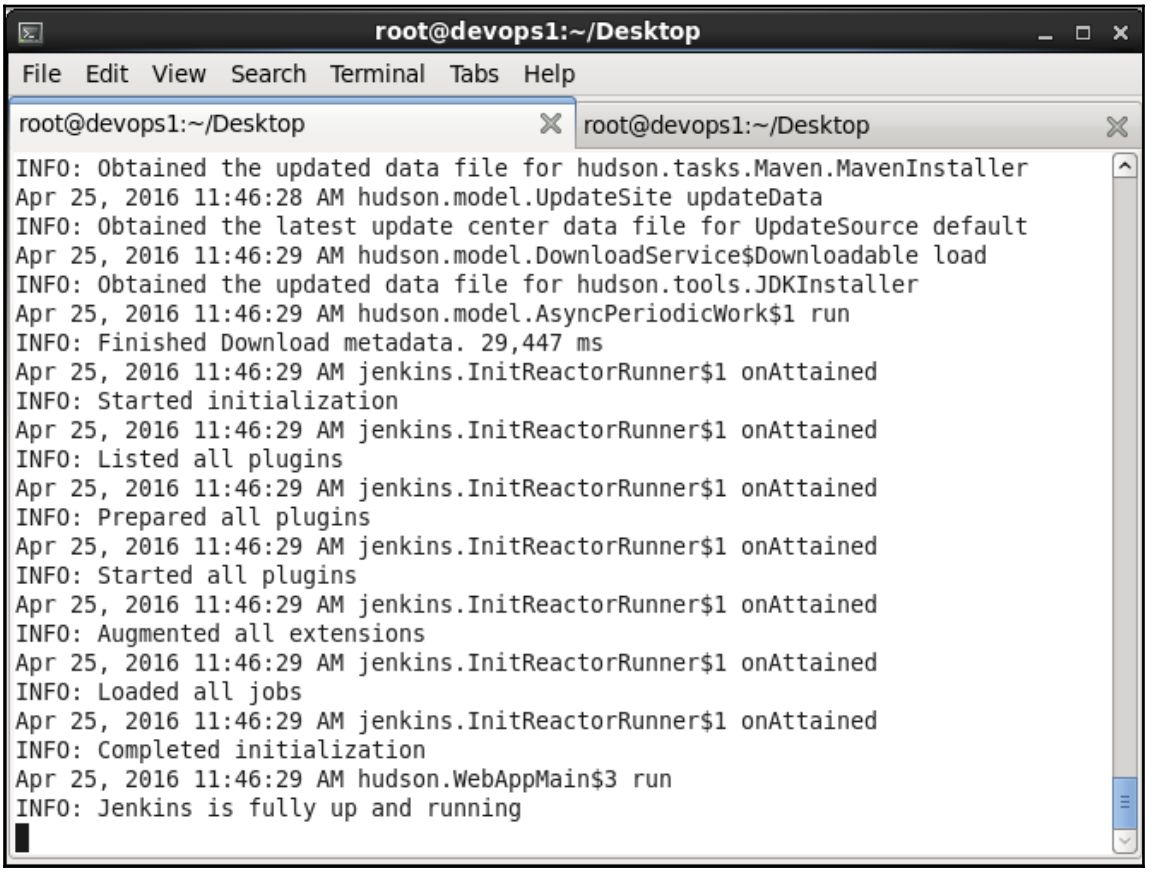
One of the easiest options I recommend is to use WAR file. WAR file can be used with container or Web application Server or without it. Java installation is must before we intend to use WAR file for Jenkins.

1. Download jenkins.war file from https://jenkins.io/
2. Open command prompt in Windows or Terminal in CentOS; go to the directory where Jenkins.war file is stored and execute following command in command

prompt or terminal:

```
java - jar jenkins.war
```

1. Once, Jenkins is fully up and running as shown in the screenshot below, explore Jenkins in the web browser by visiting http://localhost:8080.



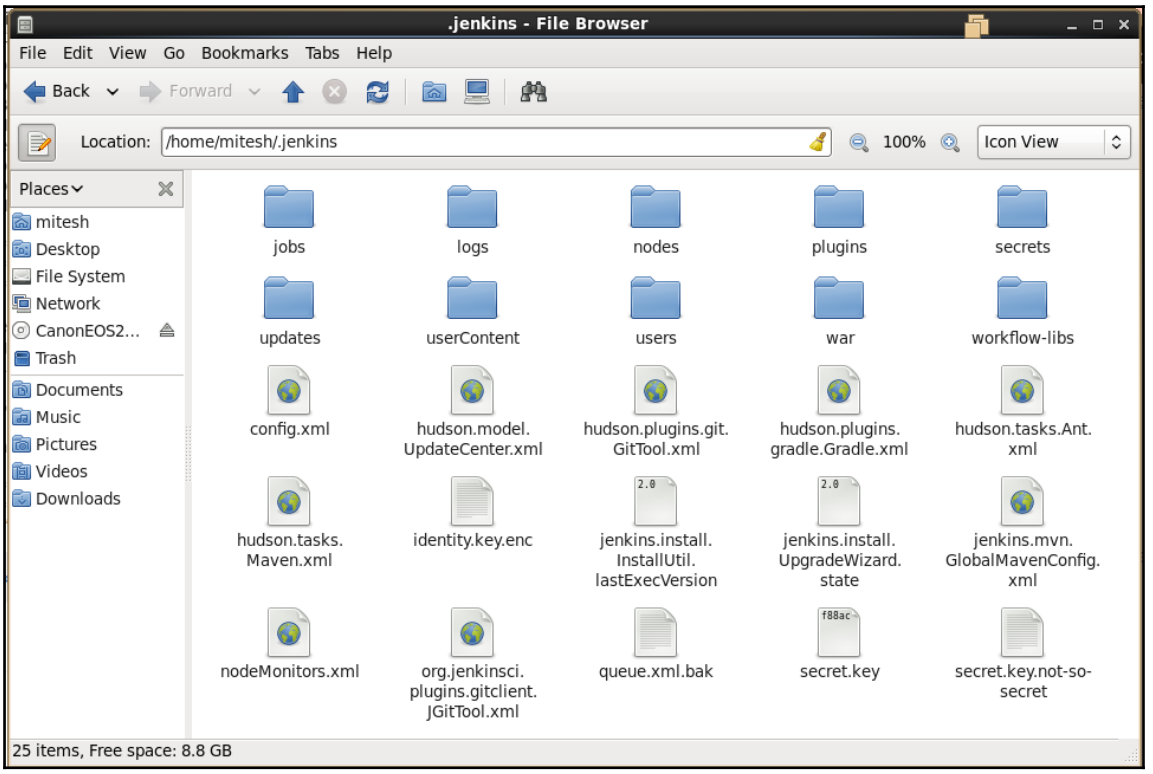
1. By default, Jenkins works on port 8080. Execute the following command at command prompt in Windows or Terminal in Linux:

```
java -jar jenkins.war --httpPort=9999
```

1. For https, use the following command at command prompt in Windows or Terminal in Linux:

```
java -jar jenkins.war --httpsPort=8888
```

1. Once Jenkins is running, visit Jenkins home directory. In our case we have installed Jenkins 2 on CentOS 6.7 virtual machine.
2. Go to /home/<username>/.jenkins directory as shown in the screenshot below. If .jenkins directory is not available then make sure that hidden files are visible. In CentOS, press ctrl+h to make hidden files visible.

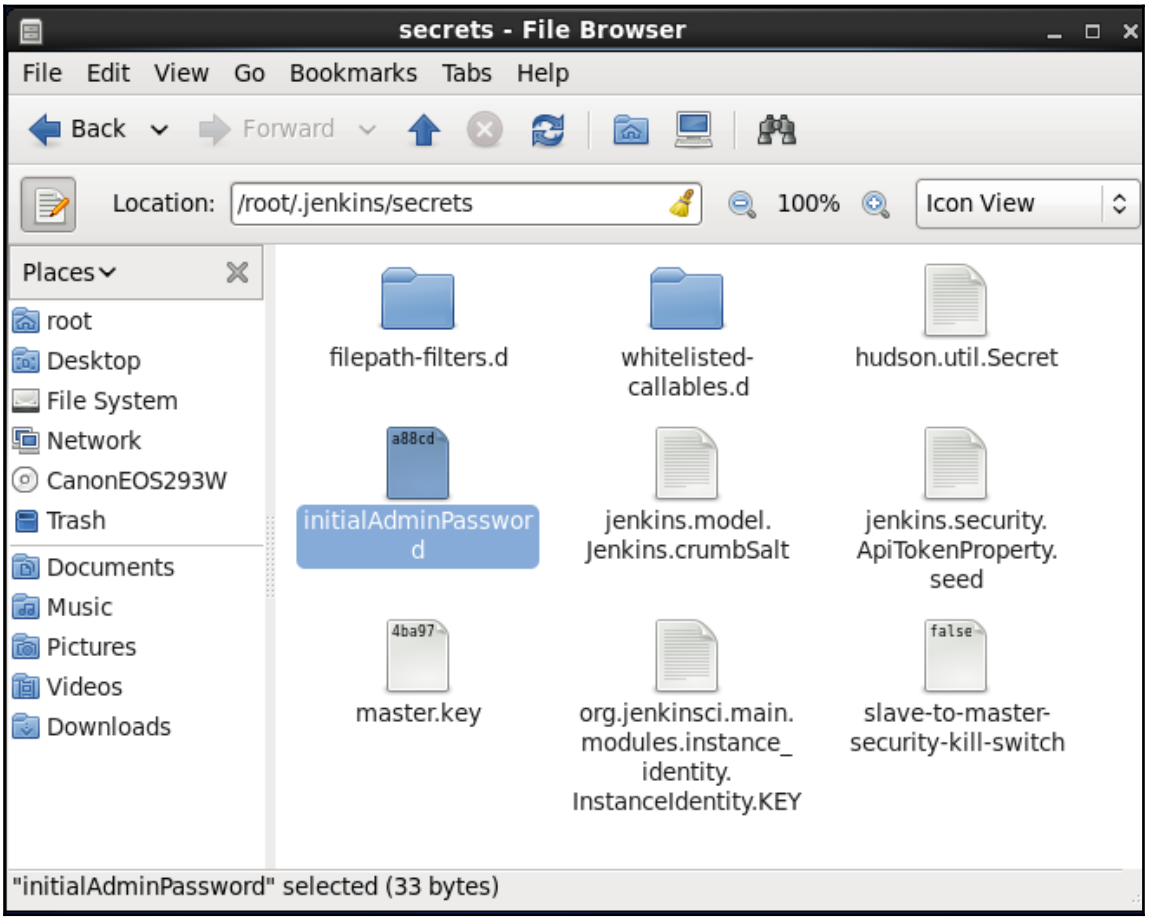


Setting up Jenkins

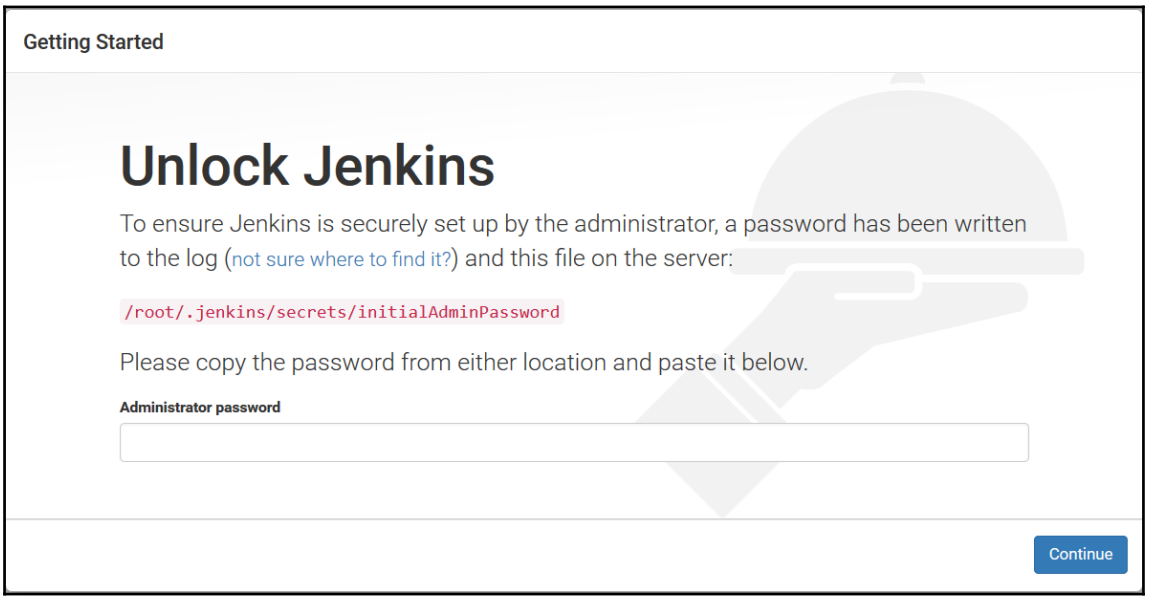
Now that we have installed Jenkins, let's verify whether Jenkins is running or not. Open any browser installed in your system and navigate to http://localhost:8080 or http://<IP_ADDRESS>:8080. If you have already used Jenkins earlier and recently downloaded Jenkins 2 WAR file, then it will ask for security setup.

To unlock Jenkins, we will follow these steps:

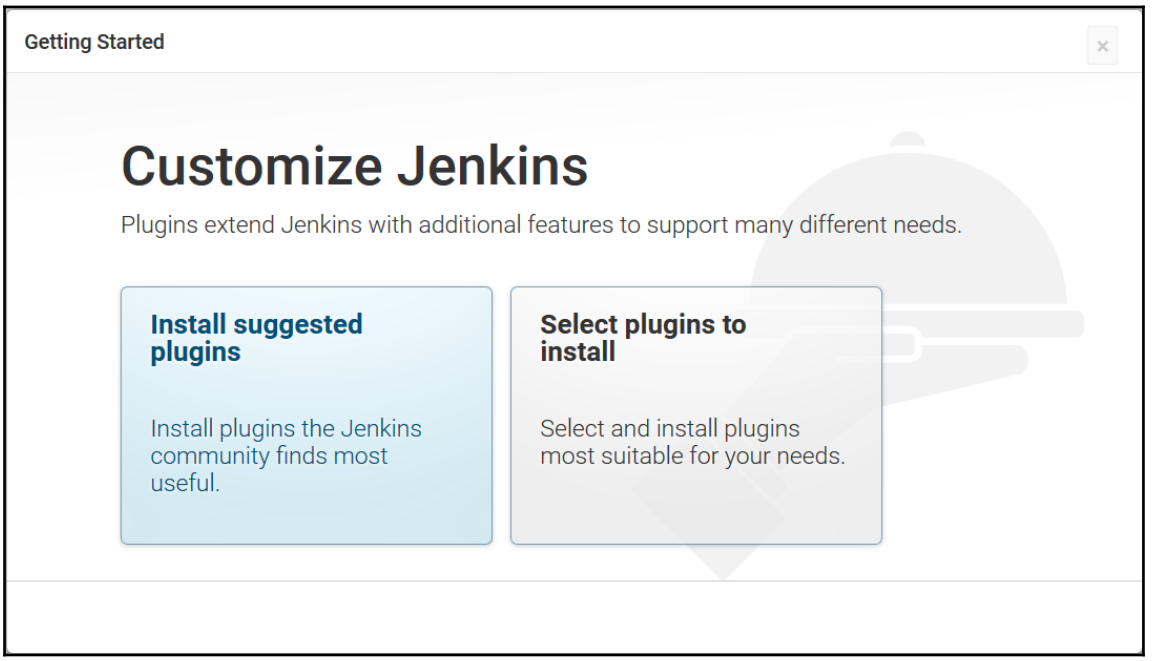
1. Go to .jenkins directory and open initialAdminPassword file from secrets subdirectory as shown below:



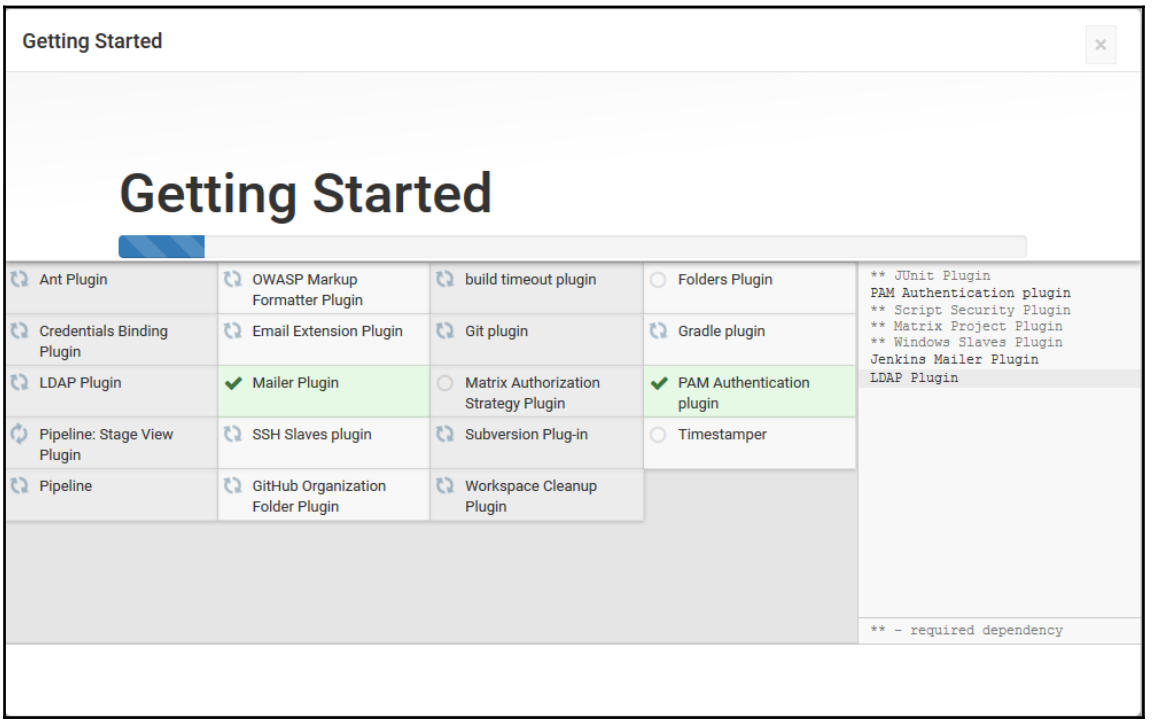
2. Copy password available in that file and paste it in **Administrator password** box and click on **Continue** as shown below:



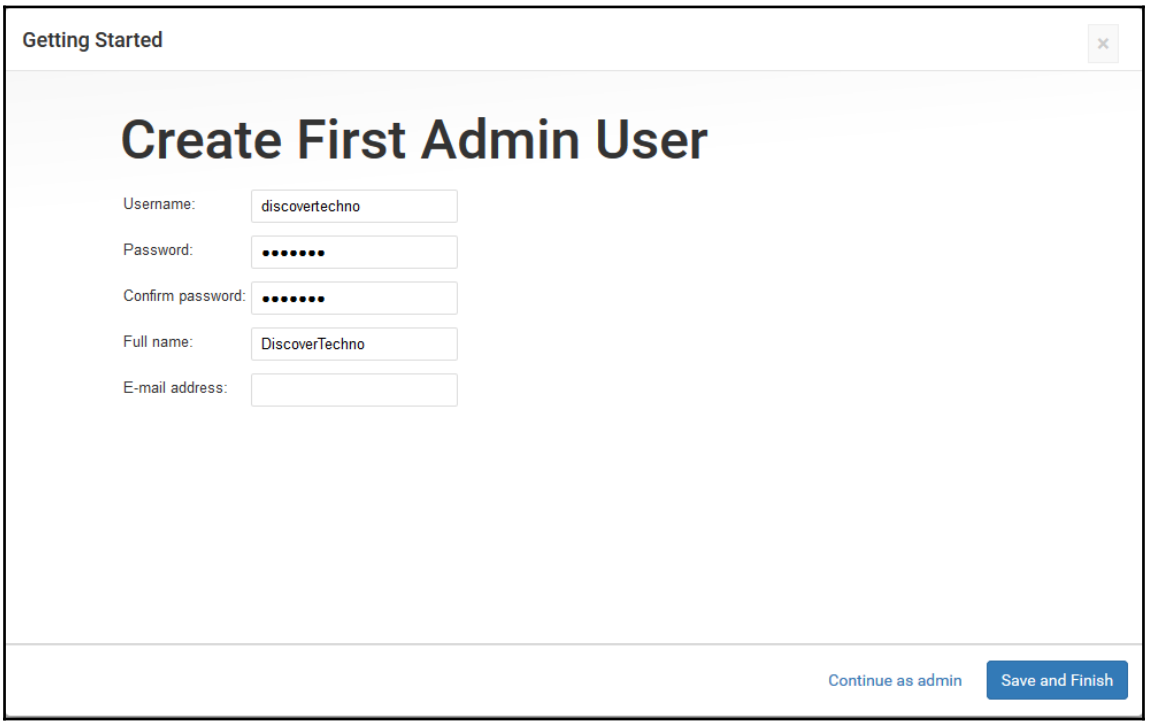
3. Clicking **Continue** will redirect you to **Customize Jenkins** page as shown below. Click on **Install suggested plugins**.



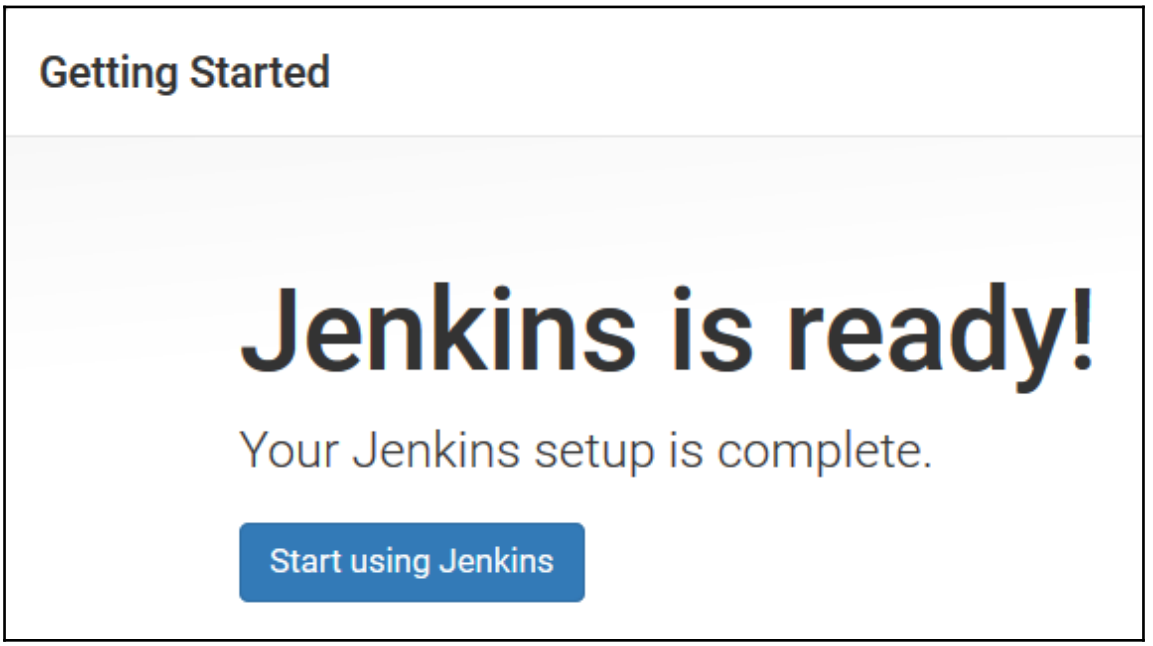
4. Installation of required plugins will start. Make sure that you have internet connection working.



5. Once all required plugins are installed; you will see **Create First Admin User** page. Provide required details and click on **Save and Finish**.



6. Jenkins is Ready! Our Jenkins setup is completed. Click on **Start using Jenkins**.



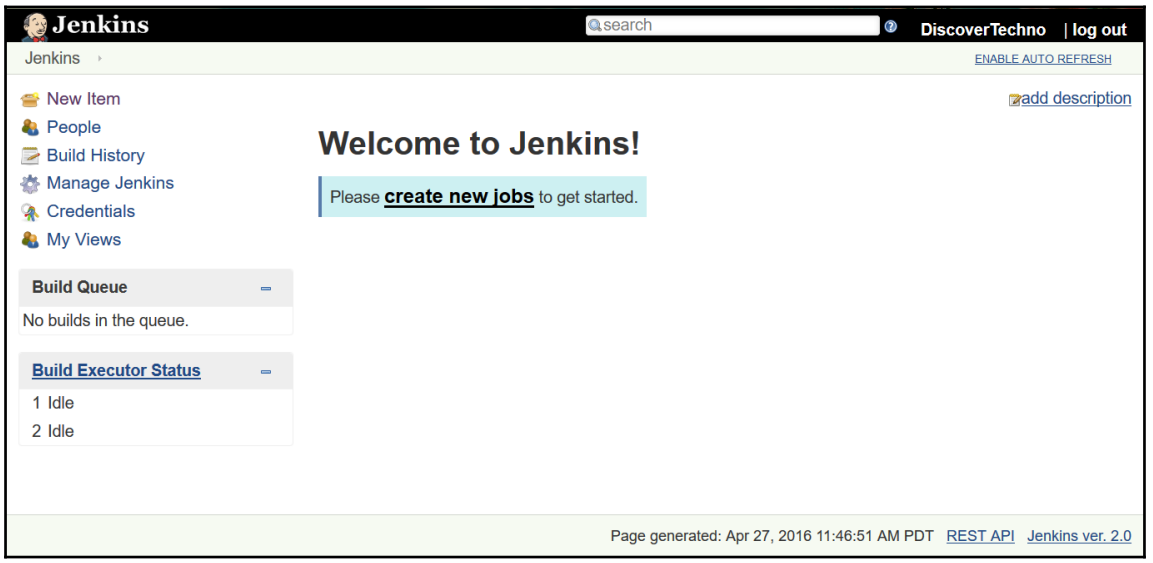
Get Jenkins plugins at:
https://wiki.jenkins-ci.org/display/JENKINS/Plugins

Jenkins dashboard

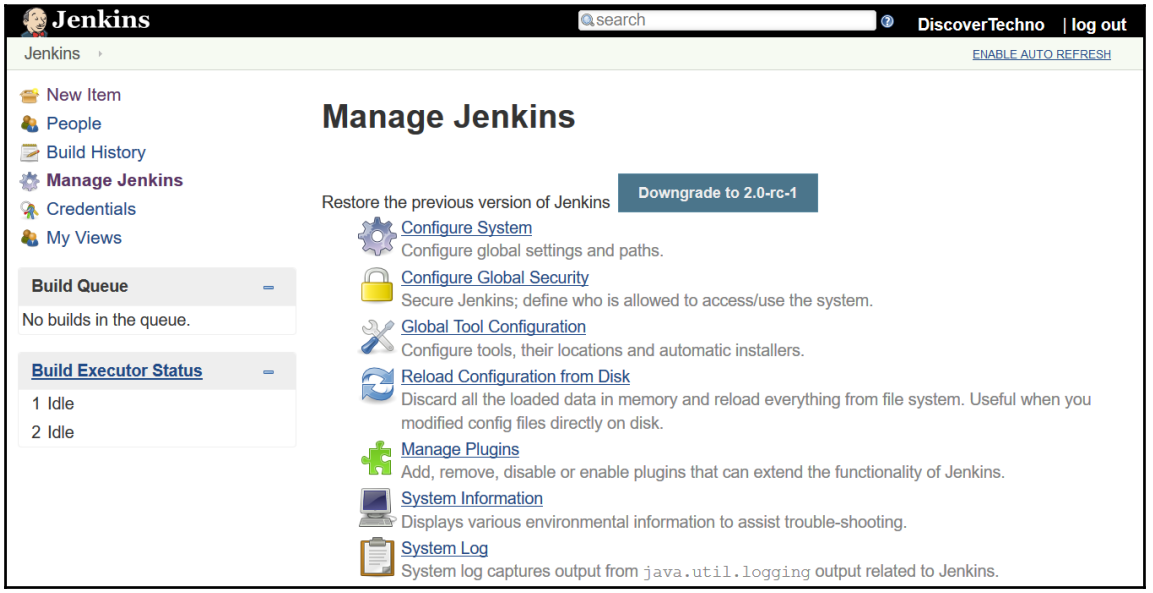
Jenkins dashboard is a simple and powerful place where we can manage all builds and hence we can manage application delivery pipeline too. Open http://<localhost or IP address>:8080 from browser. Log in with the user credentials which we have created earlier. It will direct us to dashboard of Jenkins.

Let's understand the dashboard parameters:

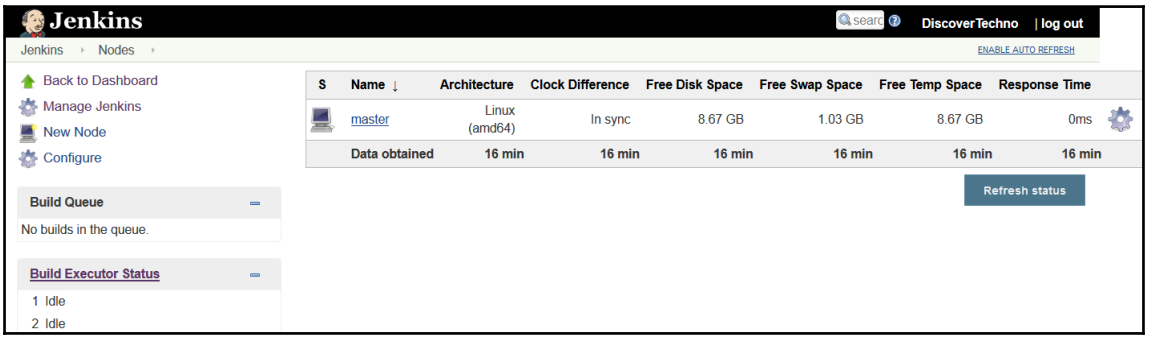
- **New Item:** To create new build job, pipeline or build flow in Jenkins 2.



- **Manage Jenkins:** Allows Jenkins 2 administrator to manage plugins, users, security, nodes, credentials, global tools configuration and so on.



- To know about existing nodes that are used for build execution, click on **Manage Nodes**. **master** node entry will be available. It is the node where Jenkins is installed. We can add multiple slave node to distribute the load the we will learn later in this chapter.



Once we have installed Jenkins and as we become familiar with the Jenkins dashboard, the next is to configure different tools that are used for build execution and create a base for Continuous Integration.

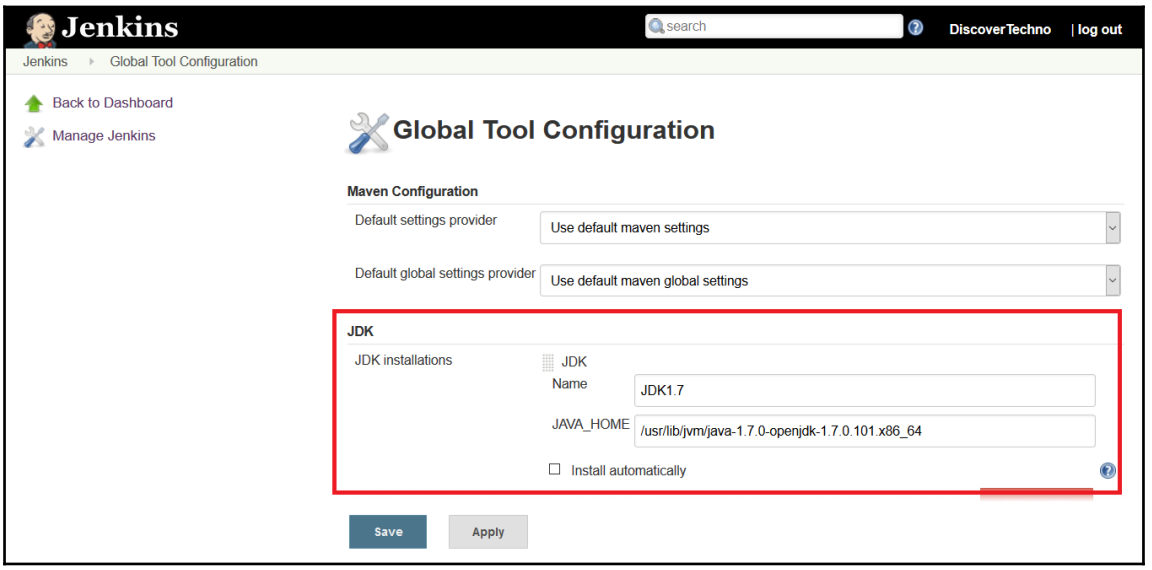
In next section, we will install or configure Java, Maven, Ant and so on.

Configuration Java, Maven/Ant in Jenkins

In Jenkins 2, **Global Tool Configuration** section is introduced and that is a good move. All major configurations related to external tools, their locations and automatic installers tools can be done in this section. Earlier, these configurations were part on **Configure System** which used to make that page bit cluttered.

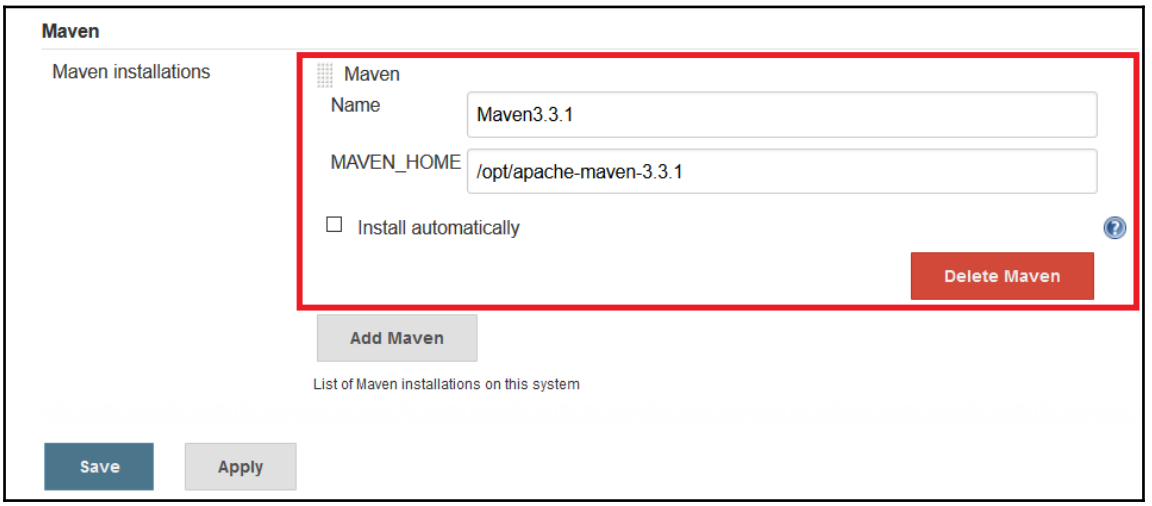
Configuring Java

To configureJava, provide **Name** and **JAVA_HOME** path or click on **Install automatically**.



Configuring Maven

To configure Maven, download installable files of Maven, extract it and keep in some directory of your Jenkins virtual machine. In **Global Tool Configuration** section, provide **Name** and **JAVA_HOME** path or click on **Install automatically**.



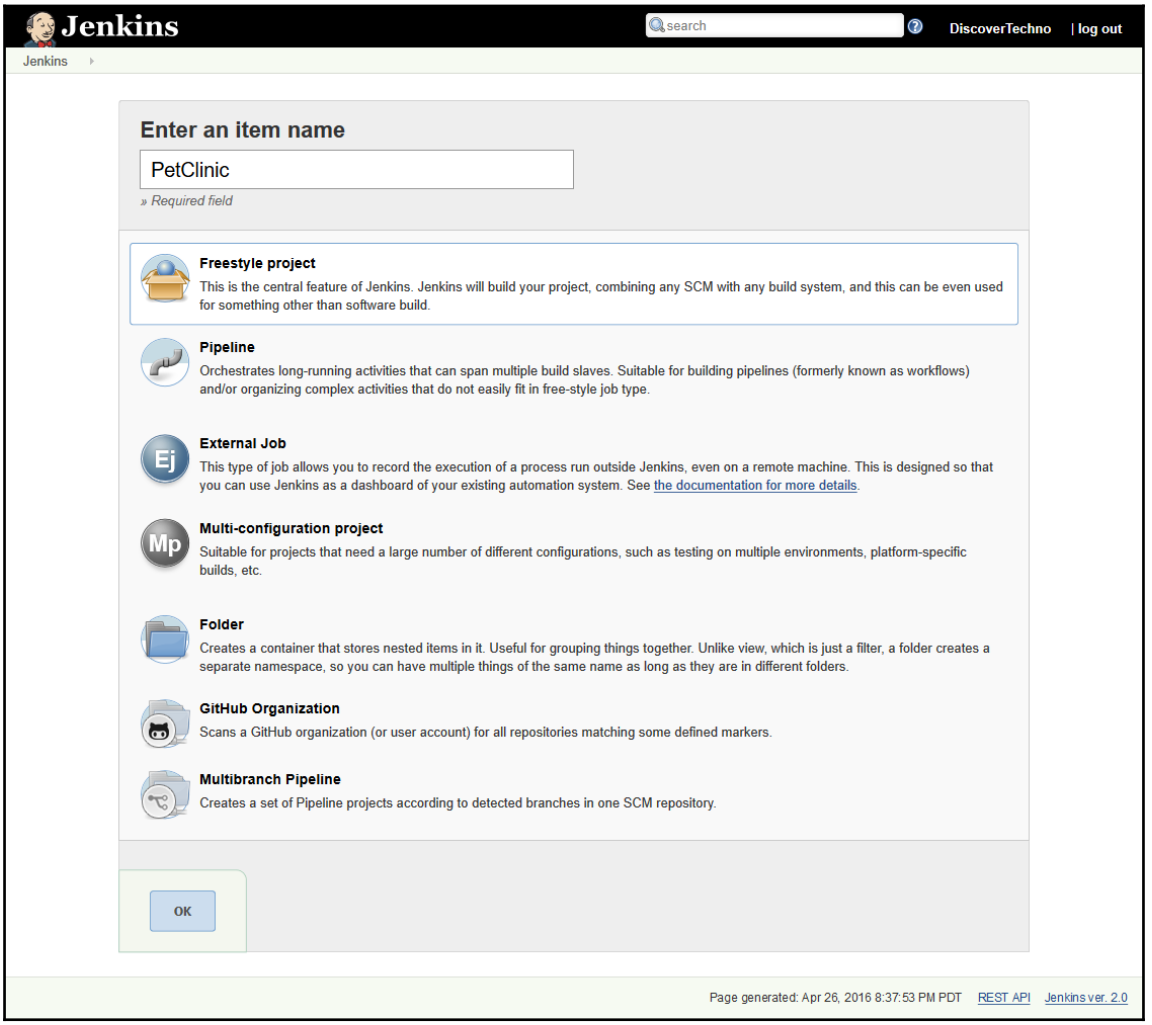
That's it! Our major configurations for running a simple build is done. Now let's go to the Home page of Jenkins dashboard to create and configure build job.

Creating and Configuring build job for Java application with Maven

Now, let's perform steps to create and configure a new build job. Go to Jenkins Dashboard and click on **New Item**.

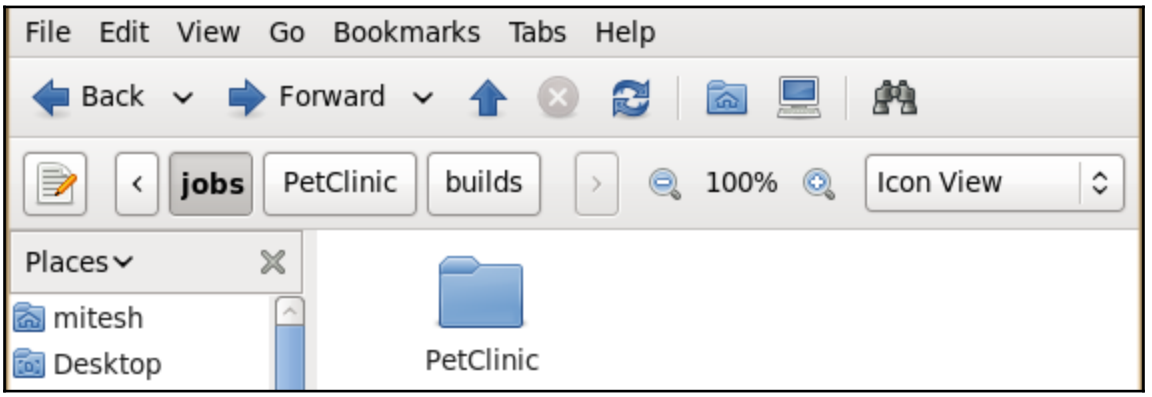
Go through all the options available of type of jobs we can create. In our case let's create a freestyle project for a demo purpose:

1. Enter an item name such as PetClinic.
2. Select **Freestyle project**.
3. Click on **OK**.



1. Let's verify what this operation does! Go to Jenkins home directory and navigate to **jobs** directory.

- We can see directory is created for newly created job with same name as shown below in the screenshot.



Next step is to configure source code repository with build job. We will use open source spring application that is hosted on GitHub as information is provided in Chapter 1,*Getting Started-DevOps Concepts, Tools, and Technology*.

1. Create a GitHub account and fork https://github.com/spring-projects/spring-petclinic.

2. After that we will get URL similar to: https://github.com/mitesh51/spring-petclinic.

- Install Git on virtual machine by using instruction available on Git documentation.

- Getting Started – Installing Git at: https://git-scm.com/book/en/v2/Getting-Started-Installing-Git

- Download for Windows at: https://git-scm.com/

3. Let's generate a new SSH key to use for authentication. Open terminal in CentOS virtual machine and make sure Git is installed in it.

4. Execute `ssh-keygen -t rsa -b 4096 -C "your_email@example.com"` command by substituting in GitHub email address.

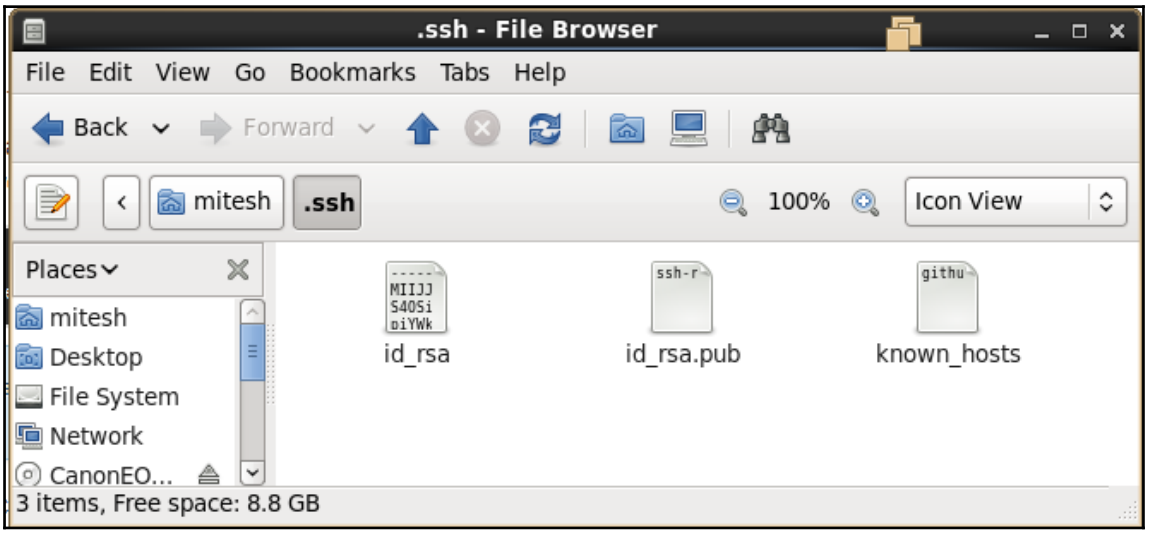
5. Press Enter when you are prompted for `Enter file in which to save the key`.

```
[mitesh@devops1 git]$ ssh-keygen -t rsa -b 4096 -C "[redacted]@gmail.com"
Generating public/private rsa key pair.
Enter file in which to save the key (/home/mitesh/.ssh/id_rsa):
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/mitesh/.ssh/id_rsa.
Your public key has been saved in /home/mitesh/.ssh/id_rsa.pub.
The key fingerprint is:
d5:48:73:9f:94:d8:02:32:75:5d:c8:08:da:33:2b:5d mitesh.soni83@gmail.com
The key's randomart image is:
+--[ RSA 4096]----+
|        o.*oo*.+.|
|         * *+o*. |
|        . * E.o  |
|         o =     |
|        S o      |
|         .       |
|                 |
|                 |
|                 |
+-----------------+
[mitesh@devops1 git]$
```

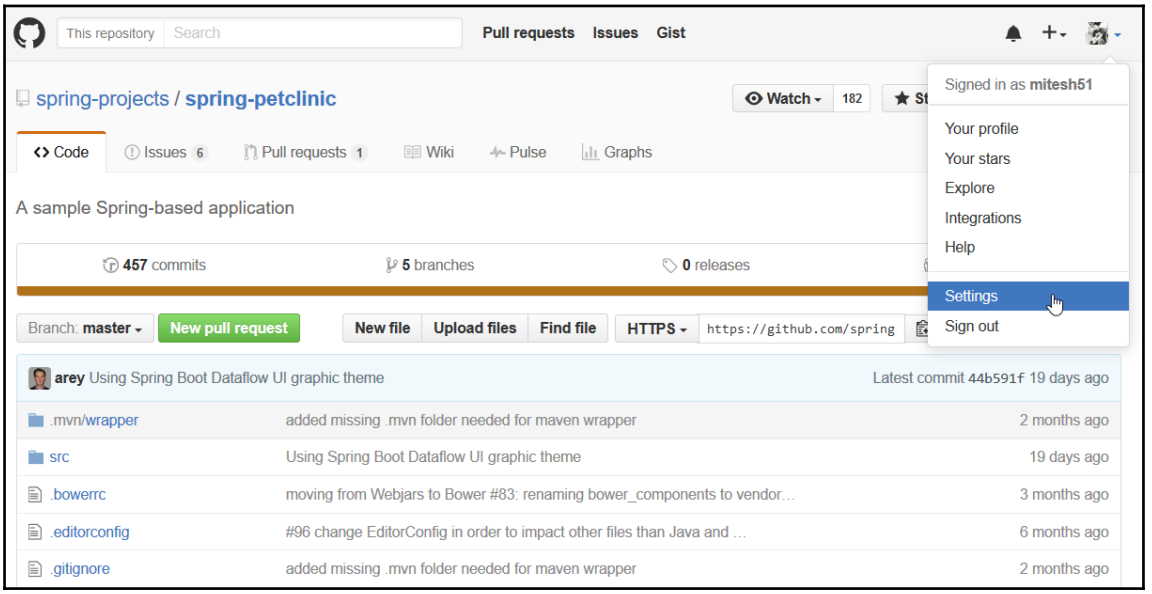
6. Add your SSH key to the ssh-agent.

```
[mitesh@devops1 git]$ ssh-add ~/.ssh/id_rsa
Identity added: /home/mitesh/.ssh/id_rsa (/home/mitesh/.ssh/id_rsa)
[mitesh@devops1 git]$
```

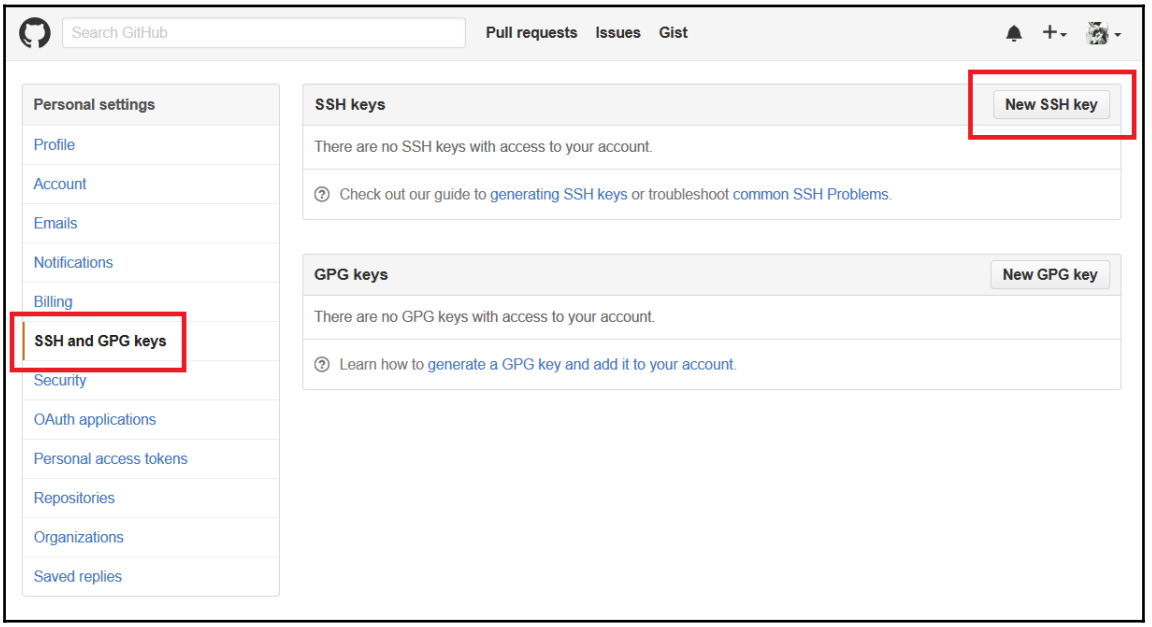
7. Verify newly generated keys in .ssh folder.



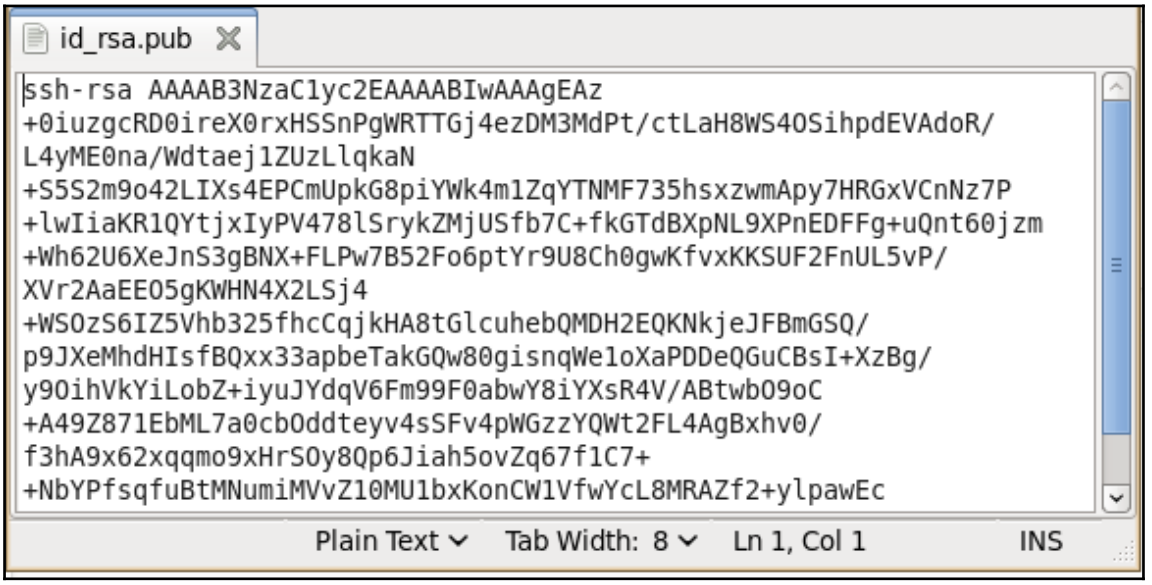
8. To configure GitHub account to use new SSH key, add it to your GitHub account. Go to https://github.com/mitesh51 and click on **Settings**.



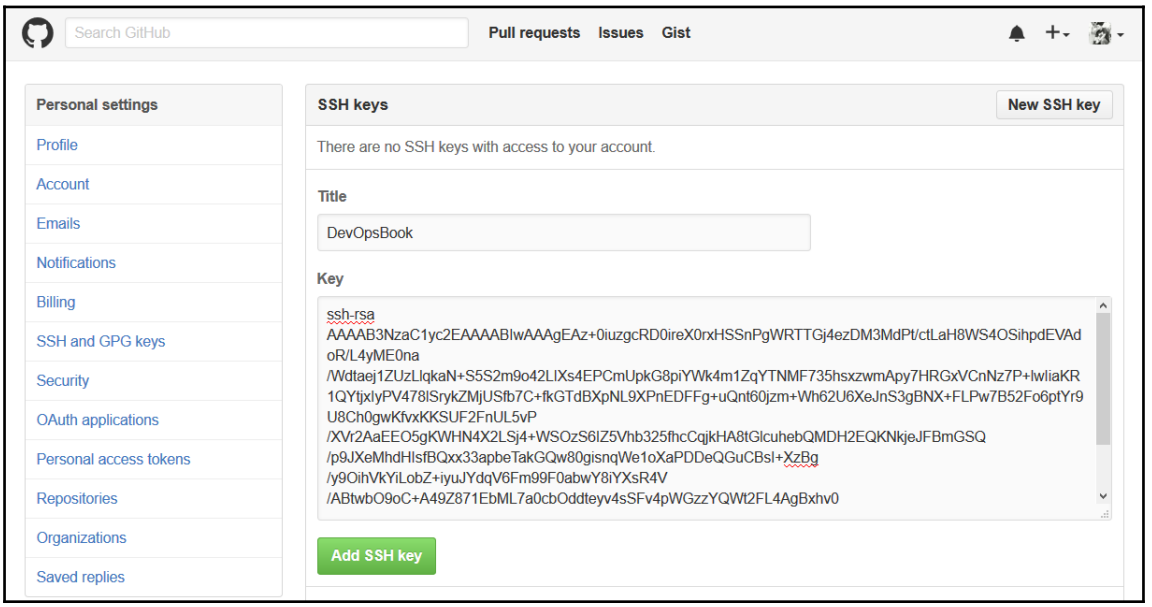
9. In the **Personal settings** sidebar, click **SSH and GPG keys**. Click **New SSH key.**



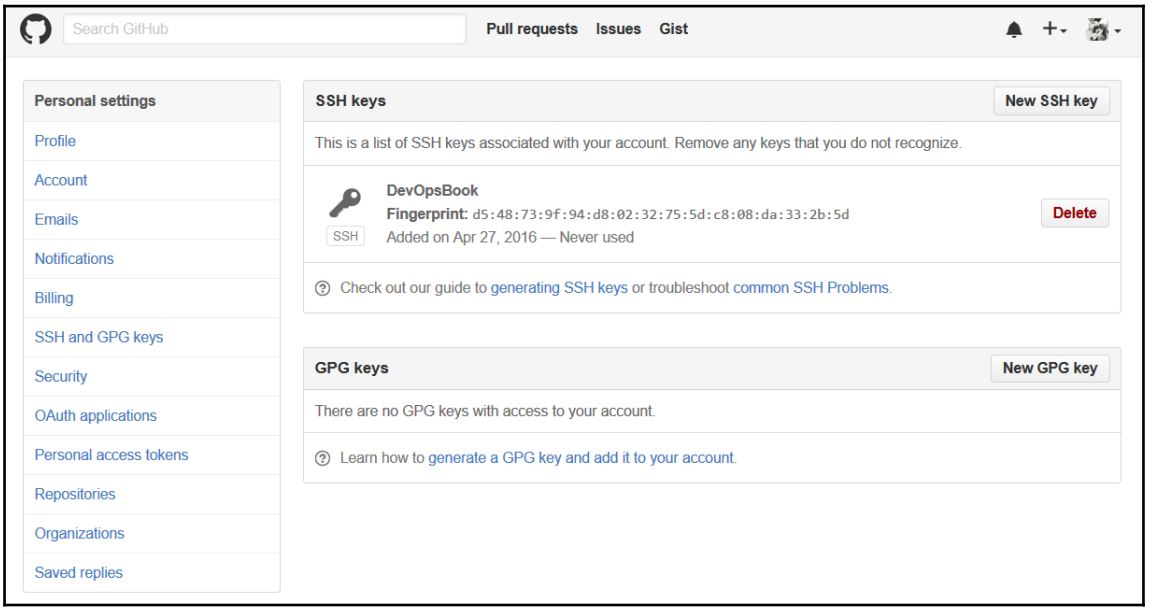
10. Open /.ssh/id_rsa.pub file in editor from CentOS virtual machine and copy the content.



11. In the **Title** field, add a descriptive label for the new key and paste the copied key content. Click on the **Add SSH key** as shown below:



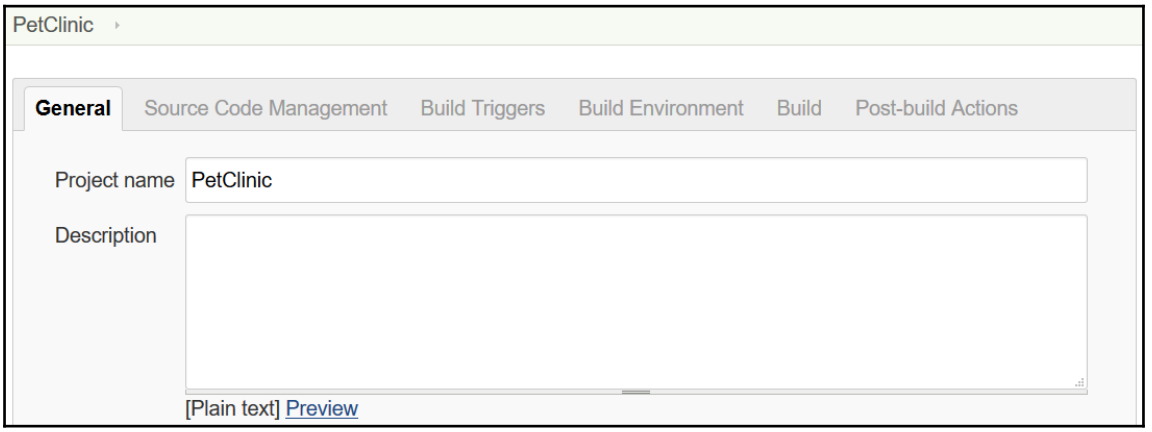
12. Verify the added SSH Key.



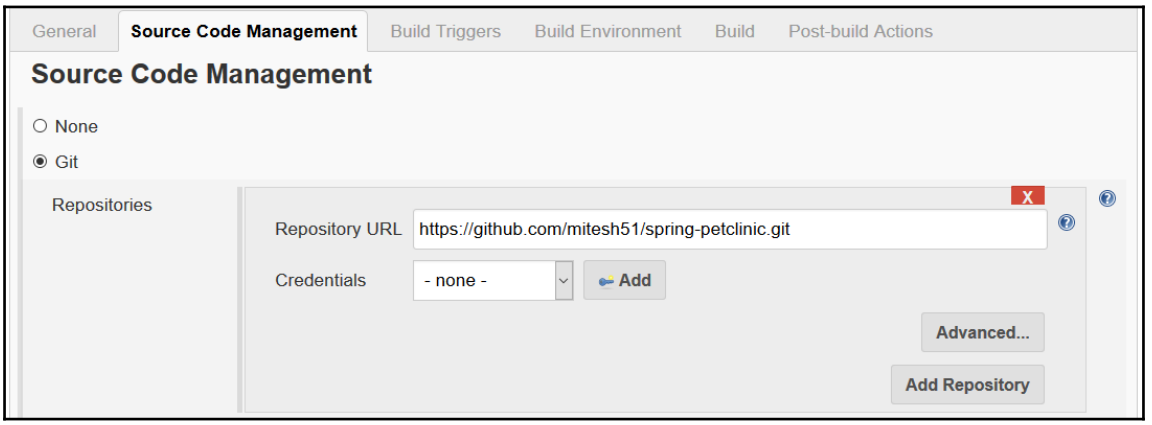
13. Now, let's verify authentication.

Once Git authentication is done, let's configure PetClinic build job.

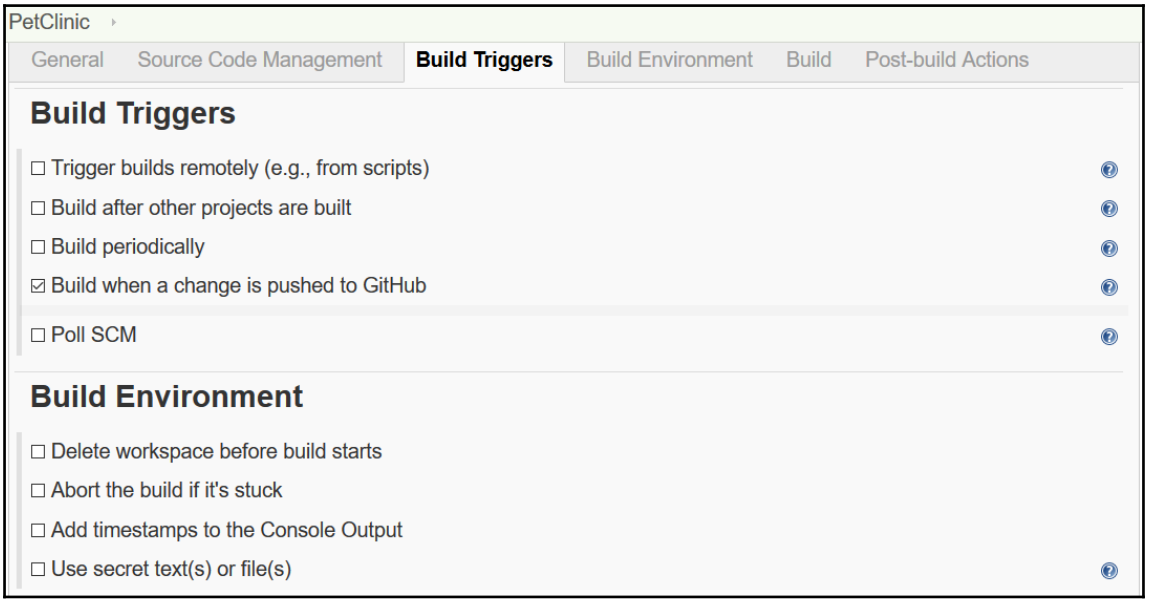
1. Click on the PetClinic build job in Jenkins dashboard. Click on **Configure** link of a PetClinic build job.



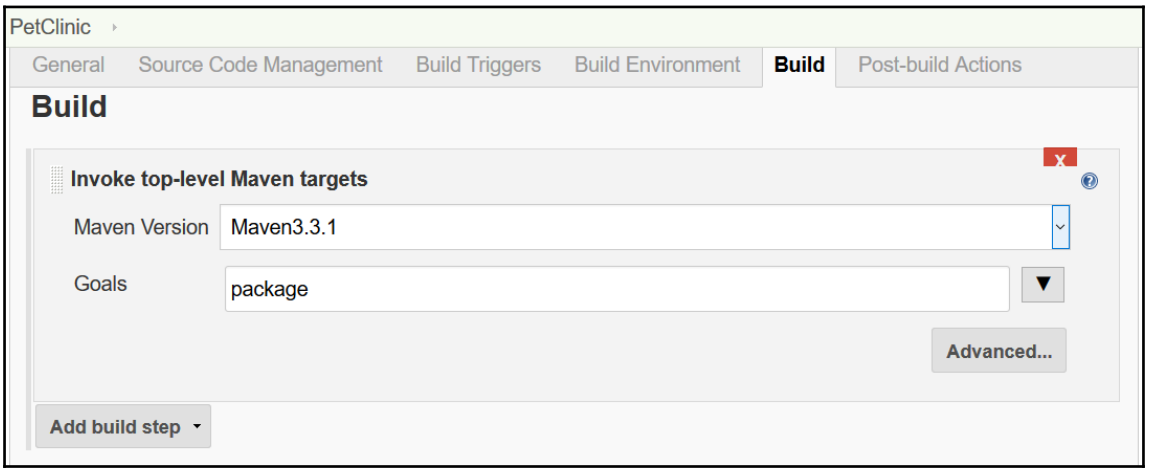
2. In **Source Code Management** provide Git URL for the Sample spring project we forked as shown below:



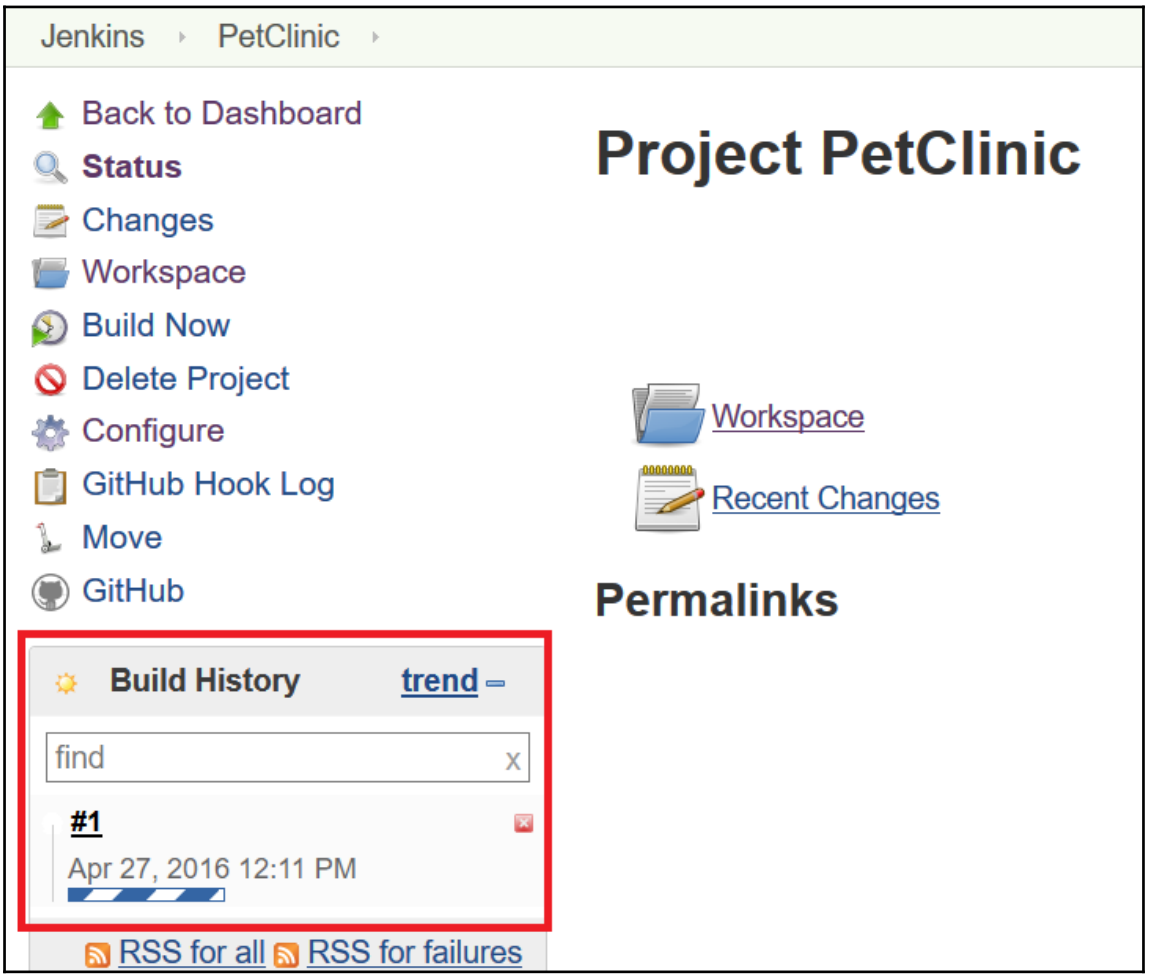
3. We will Configure **Build Triggers** and **Build Environment** as shown:



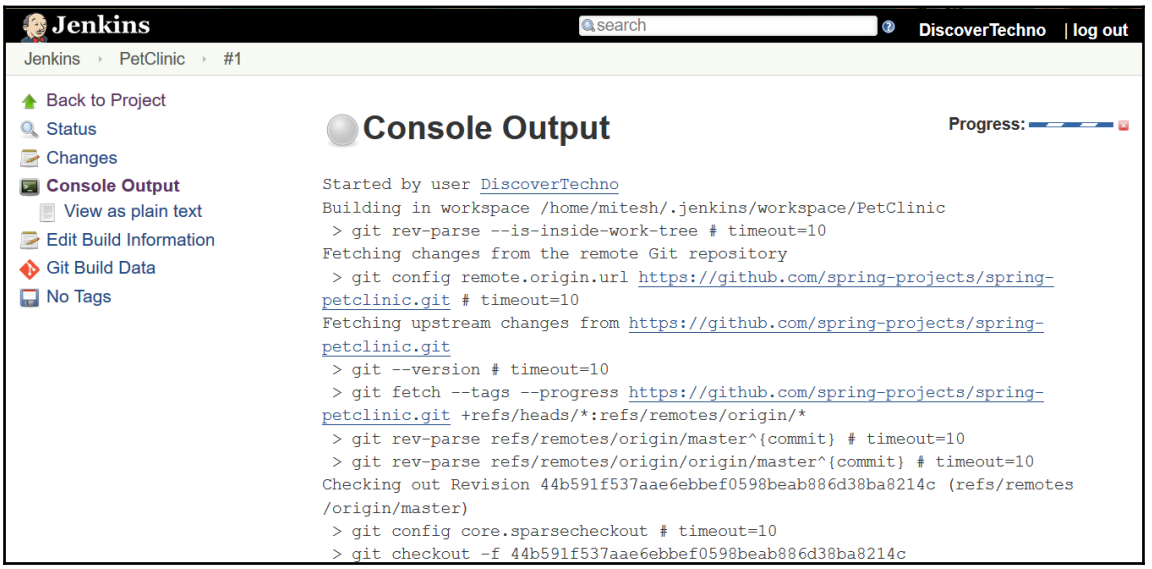
4. Click on **Add build step** and select **Invoke top-level Maven targets**. Select Maven version we configured in **Global Tools Configuration**. Enter Maven target and Click on **Save**.



5. Let's manually trigger the build by clicking on **Build Now**.



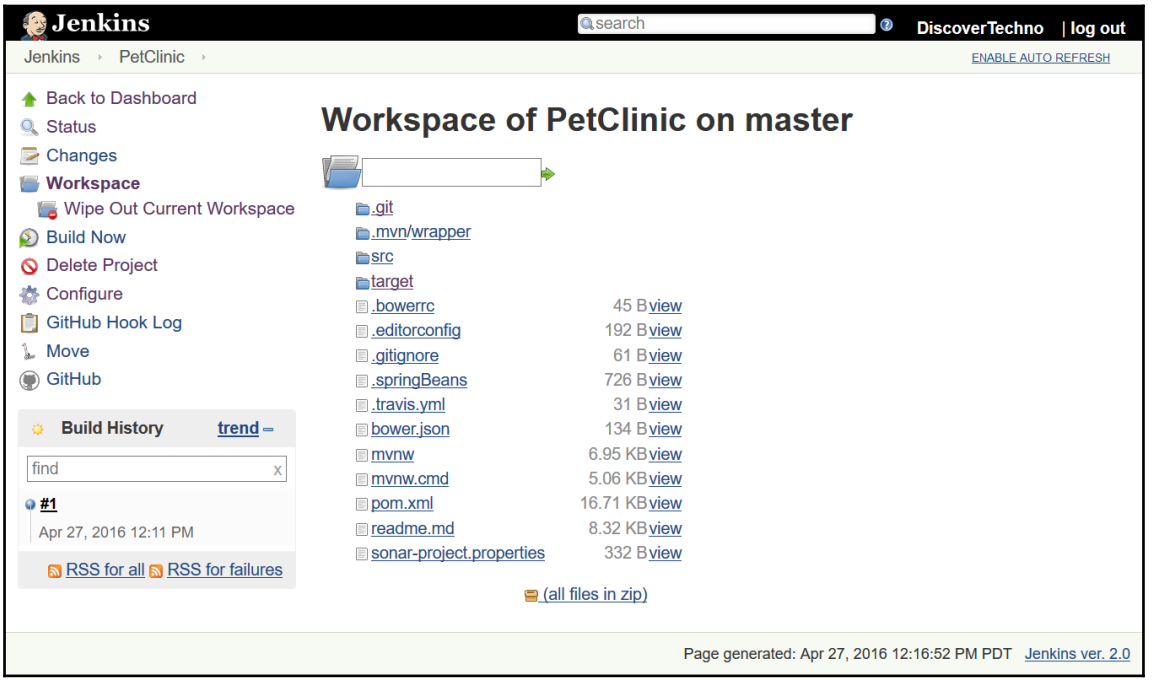
6. Click on the Build number with # sign. Open **Console Output**. Verify the Git operations executing before Maven target execution.



7. Once source code is available in the build job's workspace, Maven target will be executed and war file will be created. Verify the build status.

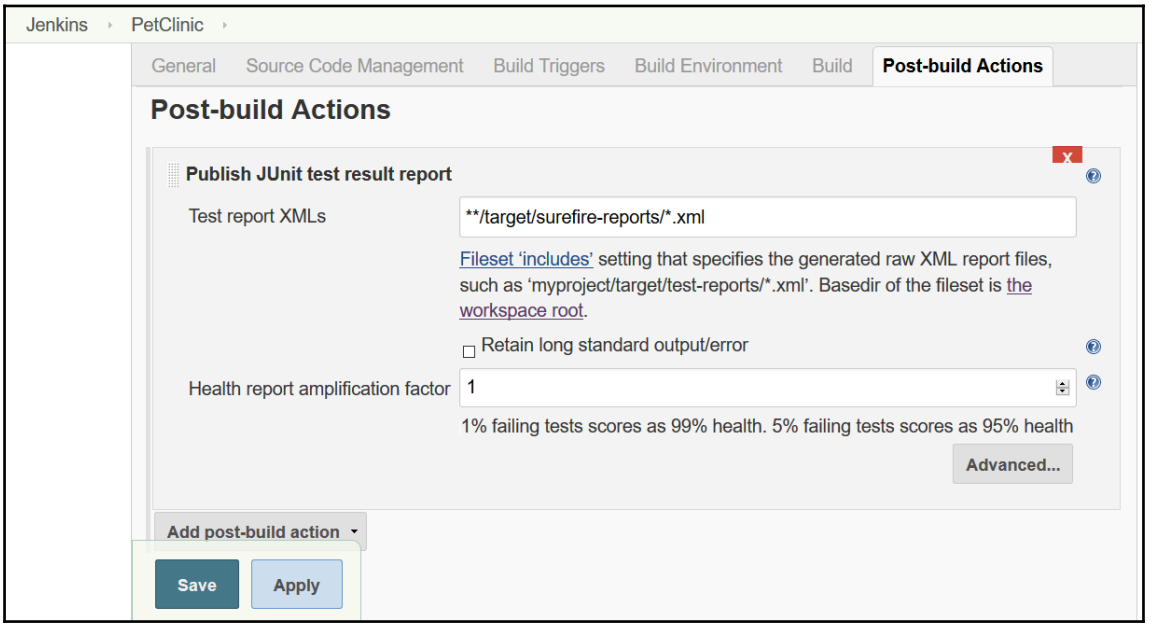
8. To verify the workspace of a build Job, click on the **Workspace** link. Verify all the

files available in the workspace. We can find these files in .jenkins folder under specific build job.



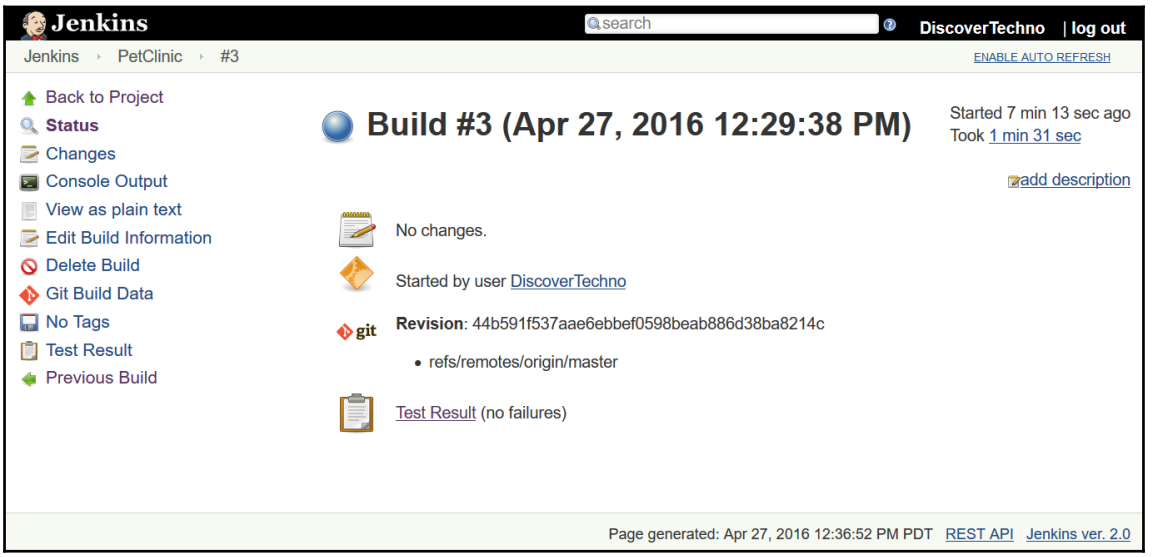
Our sample application has JUnit test cases and to execute them, we need to configure JUnit related settings in build job configuration.

1. In **Post-build Actions**, select **Publish JUnit test result report**.
2. Provide path for **Test Report XMLs** based on the workspace.
3. Click on **Apply** and then click on **Save**.

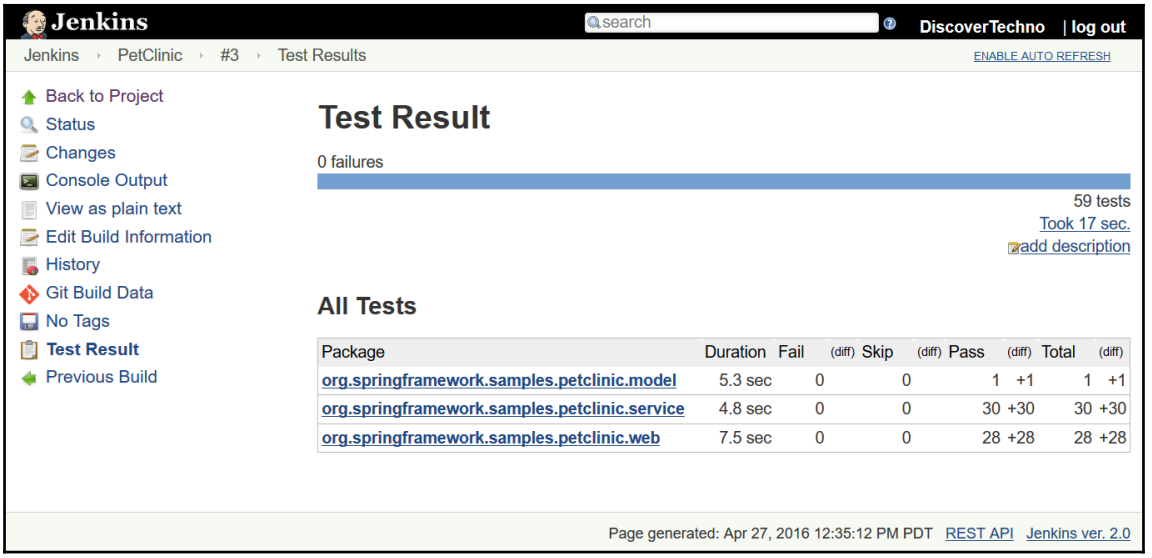


4. Once we configure JUnit settings in a build, wait for the build execution based on scheduling or click on the **Build Now** link.

5. Verify the build status on Jenkins dashboard and you will see **Test Result** link with small summary. Click on the **Test Result** link.



6. Verify all test execution status package wise. It also provides information related to duration, failed test cases.



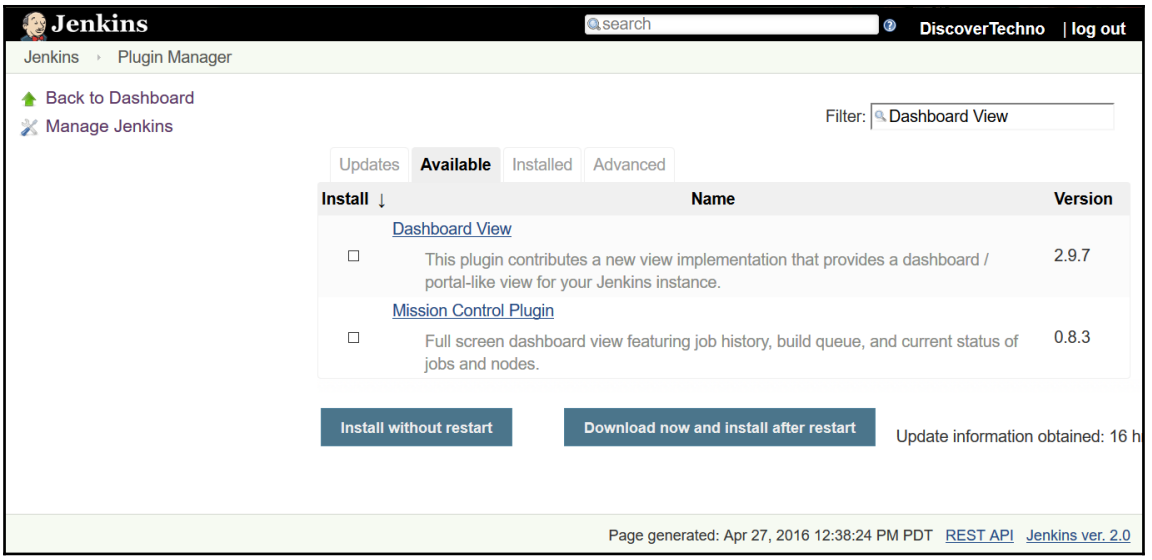
In the next section we will cover Dashboard View plugin to customize view for build jobs.

Dashboard view plugin – overview and usage

Dashboard View plugin provides different view implementation considering portal kind of layout. We can select different build jobs to be included in new view and configure different portlets for view.

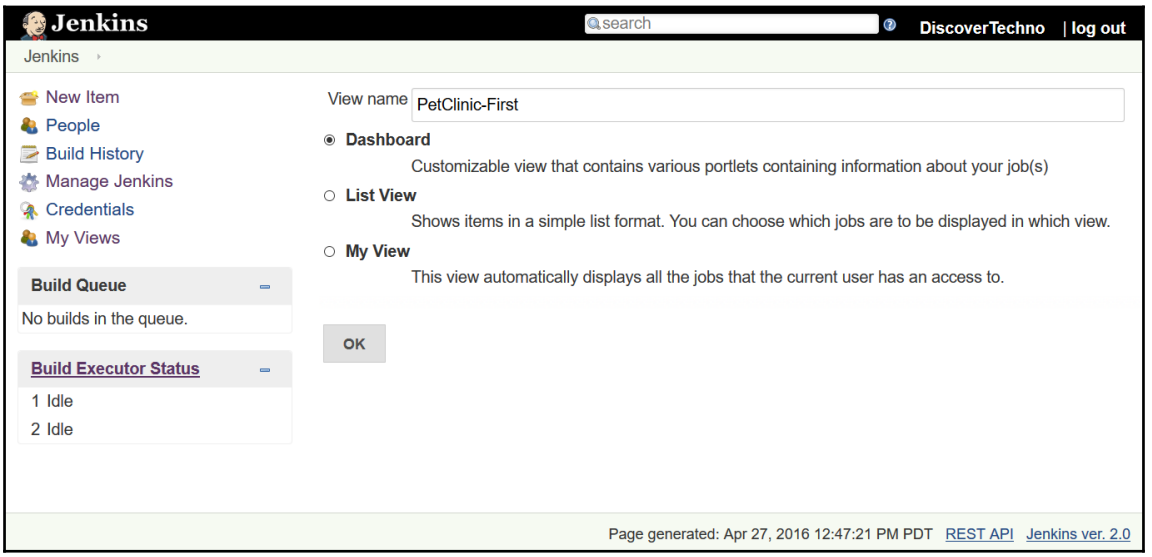
To configure:

1. Go to **Plugin Manager** from **Manage Jenkins**, and click on the **Available** tab. Search for Dashboard View plugin and click **Install without restart**.



2. Once installation of Dashboard View plugin is completed successfully, we can create a new view by clicking on the + sign on Jenkins dashboard.

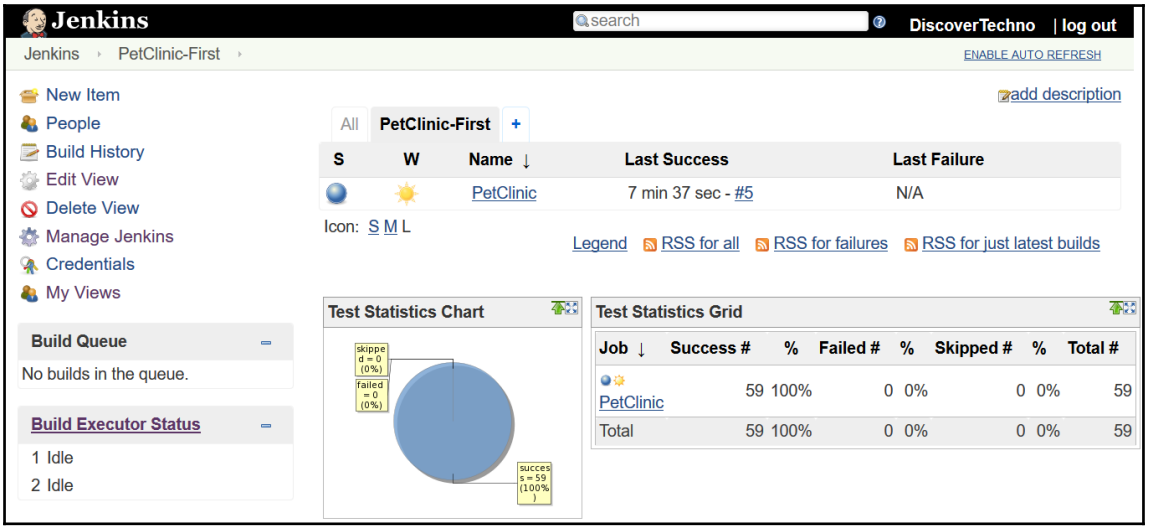
3. Enter **View name**, select view type and click on **OK**.



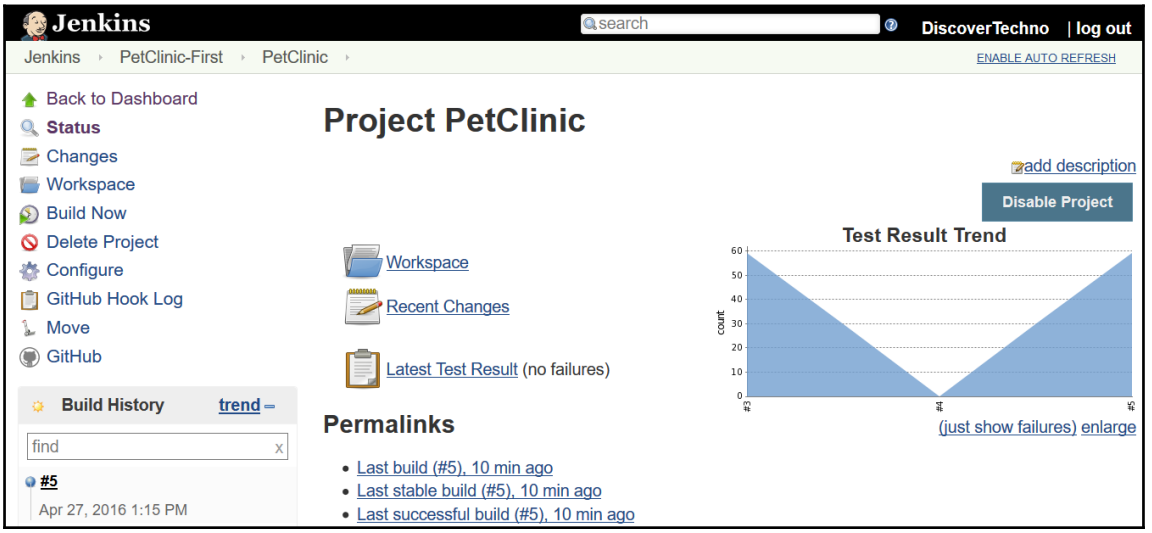
4. Click on **Edit** and configure **Dashboard Portlets** for top view, left column, right column, and bottom view. We can use different portlets such as **Test Statistics Chart**, **Trends**, and so on.

5. Add different portlets based on needs into the view and save it. Sample view is

given in the below screenshot:



6. Once we run the build job, we can find test result chart on the build job's dashboard as well.



In the next section, one of the most popular feature of Jenkins and that is distributed builds.

Consider a scenario where you want different Java applications that need different kind of JDK version to compile source files?

How to manage such situation in an effective manner? We will see answers in next section.

Managing Nodes

Jenkins provides Master-Slave concept to manage above mentioned scenarios. We can assign different build jobs to different slaves in build configuration and Master-Slave manage its overall lifecycle. Master node itself can execute the build if slave node is not configured explicitly in the build job configuration.

There are quite a few reasons why we should use this feature of Jenkins:

- Build job execution requires resources and they compete for resource availability.
- Different runtime environment for different build jobs.
- To distribute the load across slave nodes.

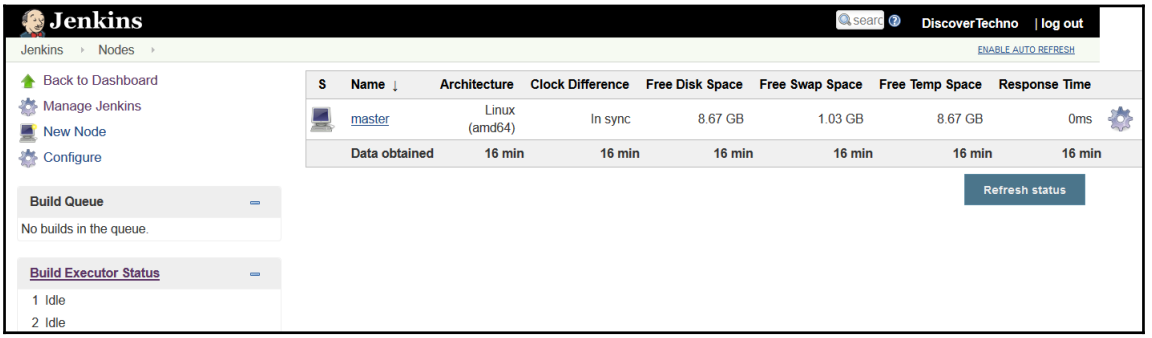
To make things more clear, we need not to install Jenkins in the slave nodes. We only need to configure slave node properly which we will demonstrate in this section.

The only requirements are:

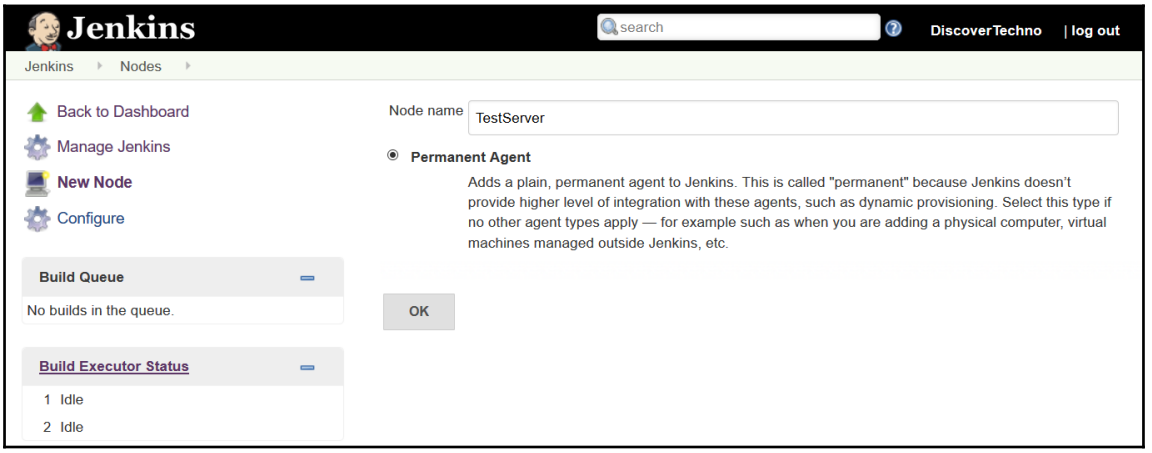
- Configurations and Runtime Environment has to be available on the slave node.
- Path needs to be configured correctly on Master node for Runtime. Environments or tools used by slave node for execution.

To create a slave node in Jenkins 2:

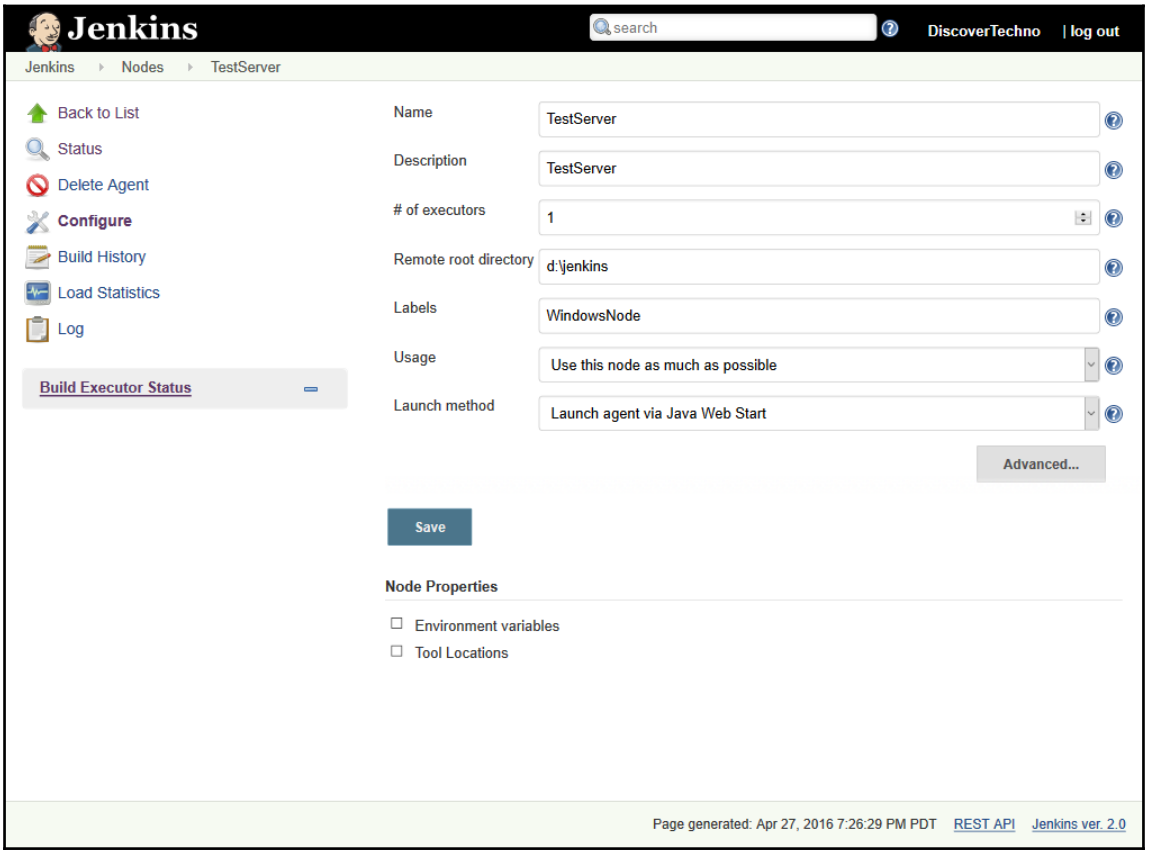
1. Click on **Manage Jenkins** link on Jenkins dashboard.



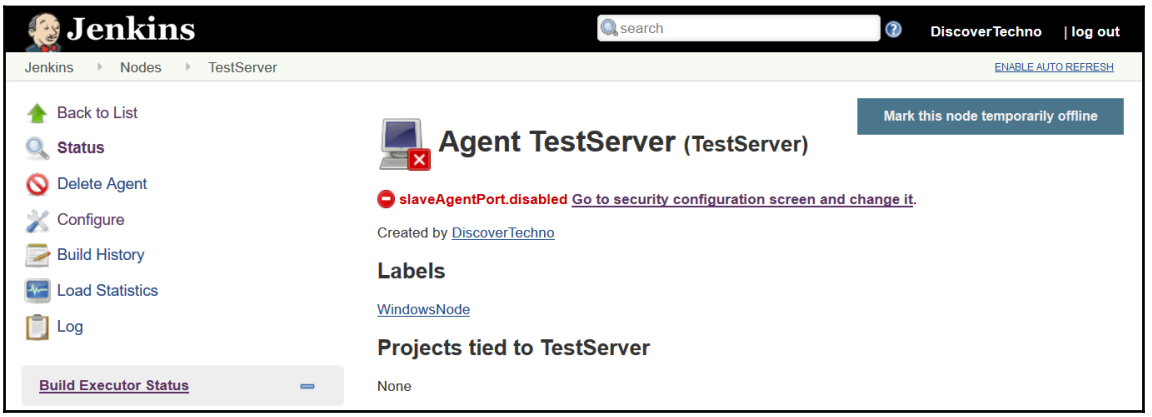
2. Verify that only Master node's entry is available. To add a new node, click on **New Node** in left sidebar. Enter node name in **Node name** field and click on **OK**.



3. Next step is to configure the newly created node. Enter **Remote root directory** that will store details related to build jobs on slave node. Give **Labels** to this node. Labels can be used to assign different build jobs to specific slave machine.

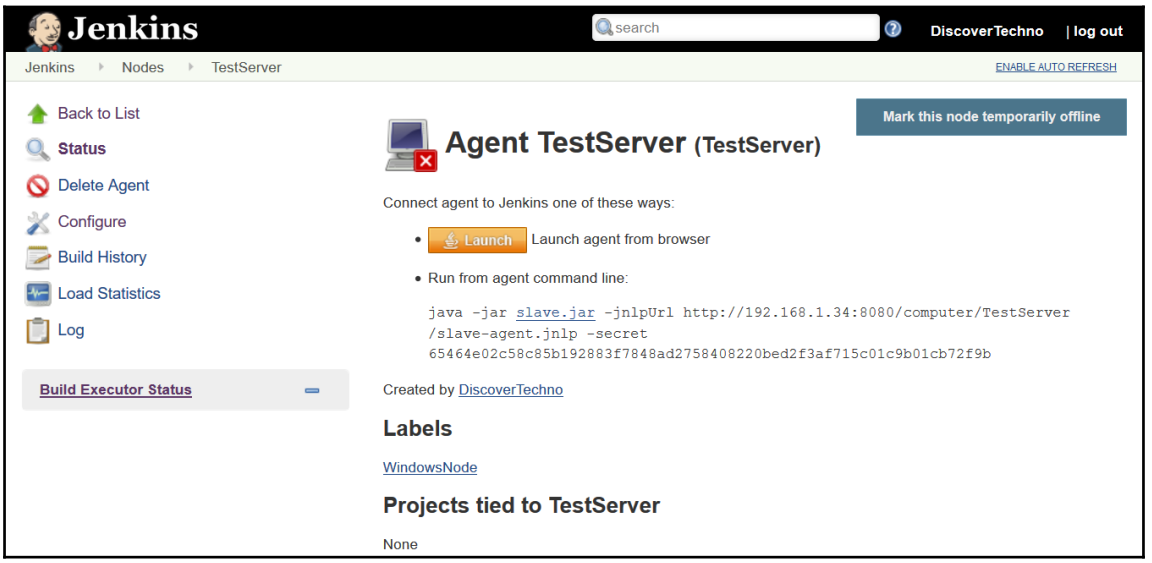


4. In Jenkins 2, after creating slave node and configuring it, if there is an error **slaveAgentPort.disabled** as shown in figure below, then we need to first solve it and then perform the further steps.

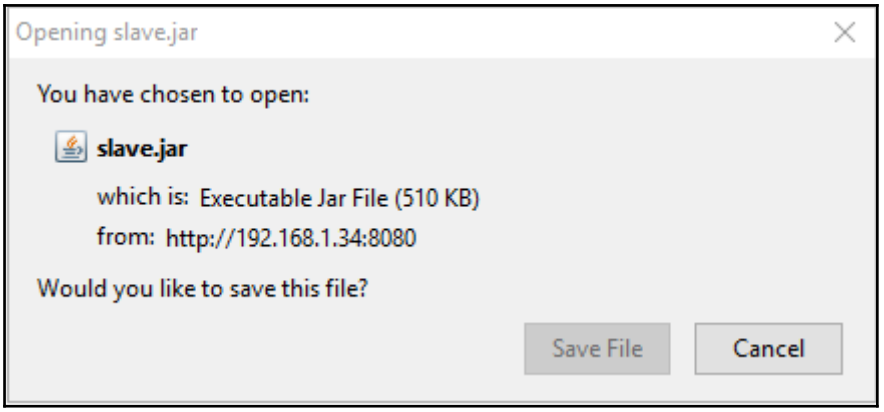


5. Go to **Manage Jenkins** page, and click on **Configure Global Security** link. Select **Enable security** and select **Fixed** or **RandomTCP port for JNLP agents** and save the configuration.

6. Next step is to connect Jenkins slave with Jenkins Master. We will connect agent to Jenkins by using command line.

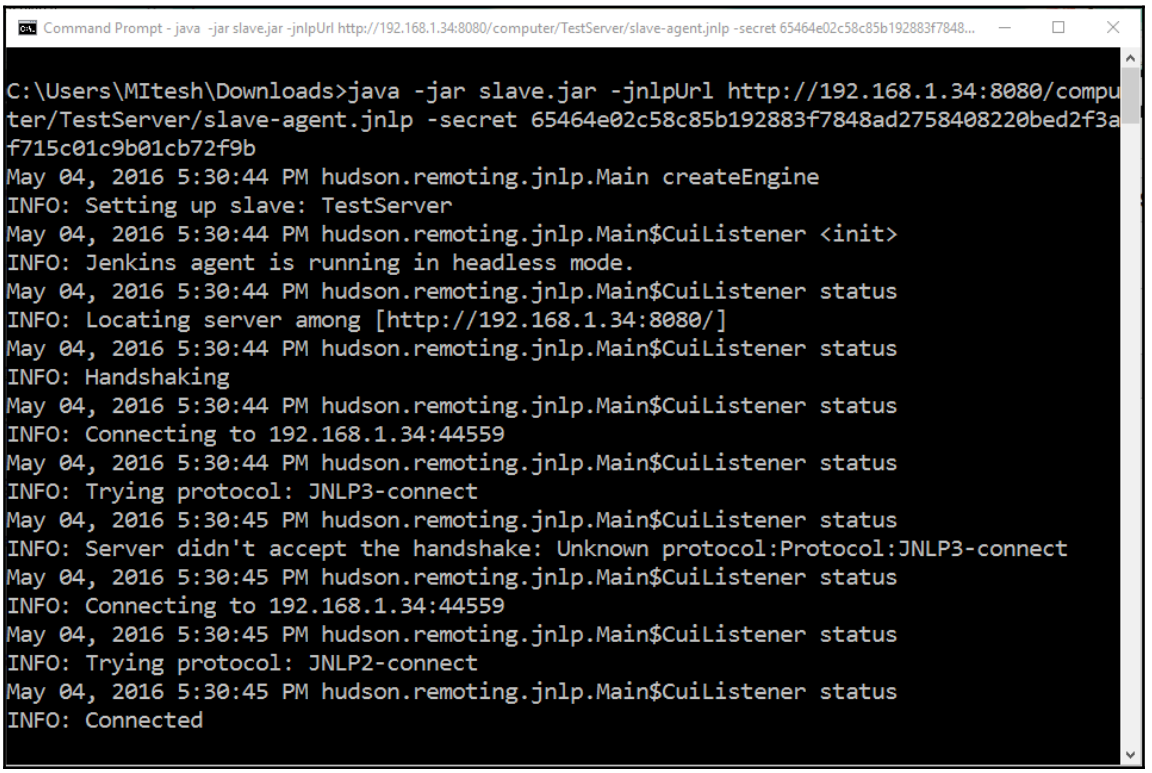


7. Download the slave.jar file and put it on slave node.



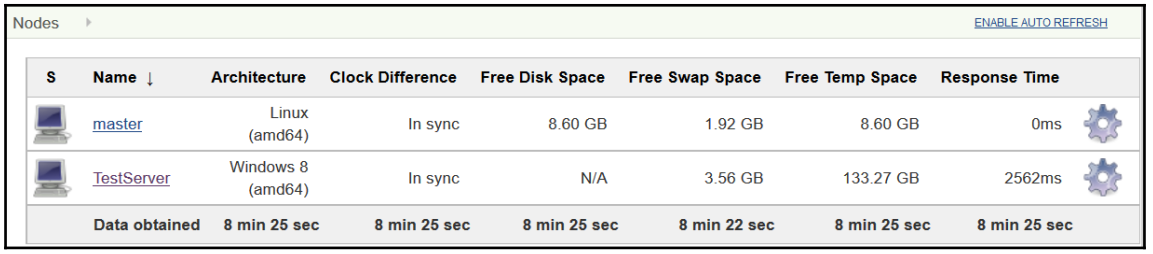
8. Execute following code in the terminal or command prompt based on the operating systems on the slave node:

java -jar slave.jar -jnlpUrl http://192.168.1.34:8080/computer/TestServer/slave-agent.jnlp -secret 65464e02c58c85b192883f7848ad2758408220bed2f3af715c01c9b01cb72f9b



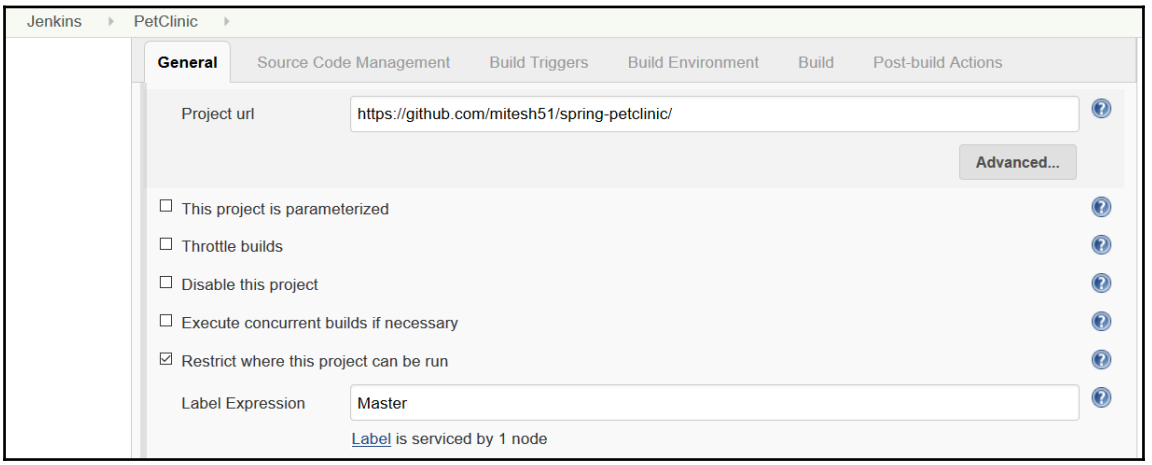
9. Verify the status of slave node in the Jenkins dashboard.

10. Now, we can see two nodes in Jenkins dashboard.

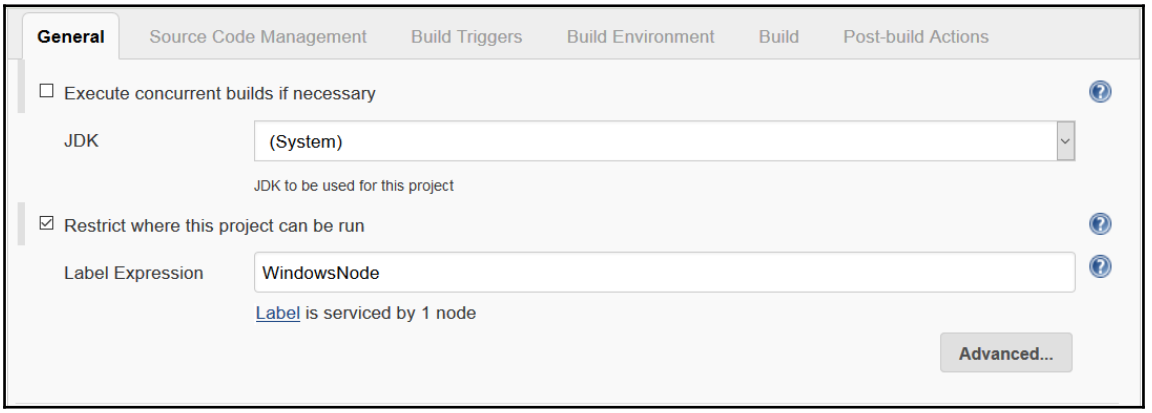


11. To configure build job to run on master, open build configuration and in **General** section select **Restrict where this project can be run**.

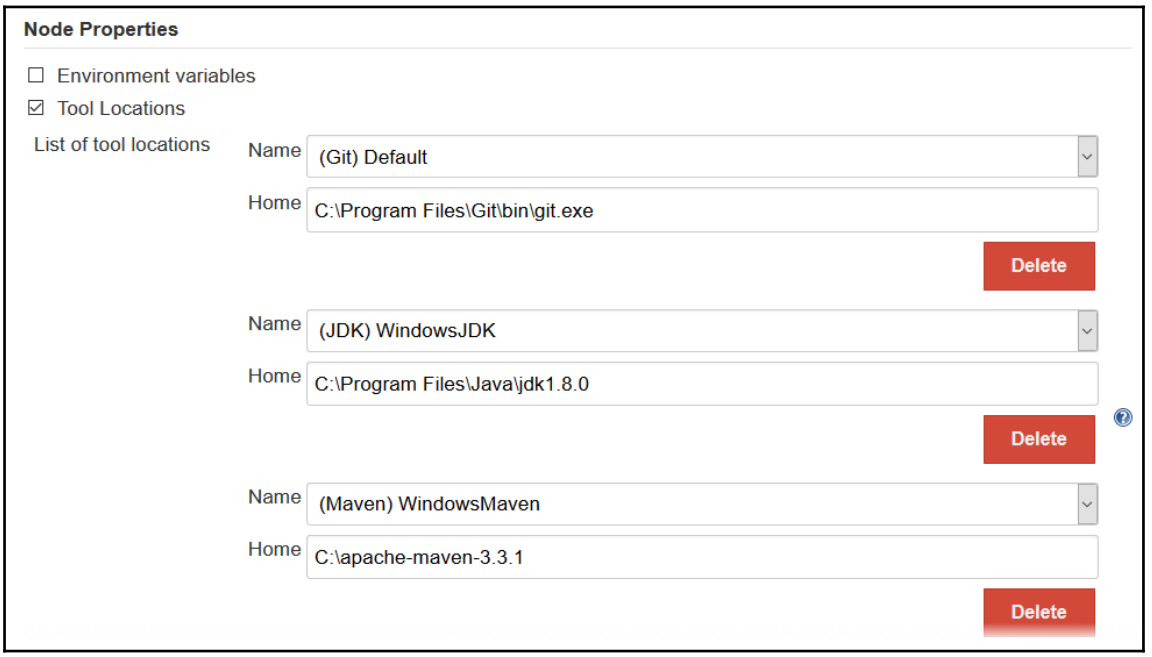
12. In **Label Expression**, enter label of the master node.



13. To configure build job to run on slave node, enter label of slave node in **Label Expression**. We can also configure **JDK** or other required path for build execution.



14. To configure tools specific to slave node, click on **Configure** in **Manage Nodes** section. In **Node Properties**, configure **Tool Locations** for slave node as shown in below image:



In the next section, we will see how to configure email notifications.

Email notifications based on build status

> *"Failure is simply the opportunity to begin again, this time more intelligently."—Henry Ford*

However, it is extremely vital to be aware about failure or at least to know when things fail to fix it and remove issues.

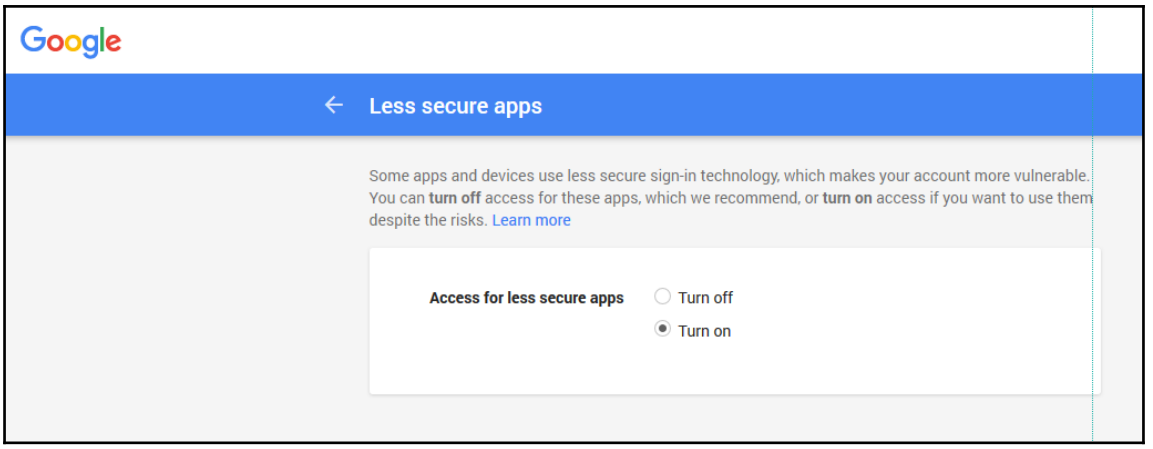
Notifications are always helpful in case of failures. Consider a scenario where build failure or test case failure has to be notified to specific set of stakeholders. In such situation it is desirable to have email notifications.

We will use Gmail configuration for setting up email notifications.

To make things work,

1. Go to: https://www.google.com/settings/u/1/security/lesssecureapps and

Turn onAccess for less secure apps as shown below to send email notifications from Jenkins 2.



2. In Jenkins dashboard:

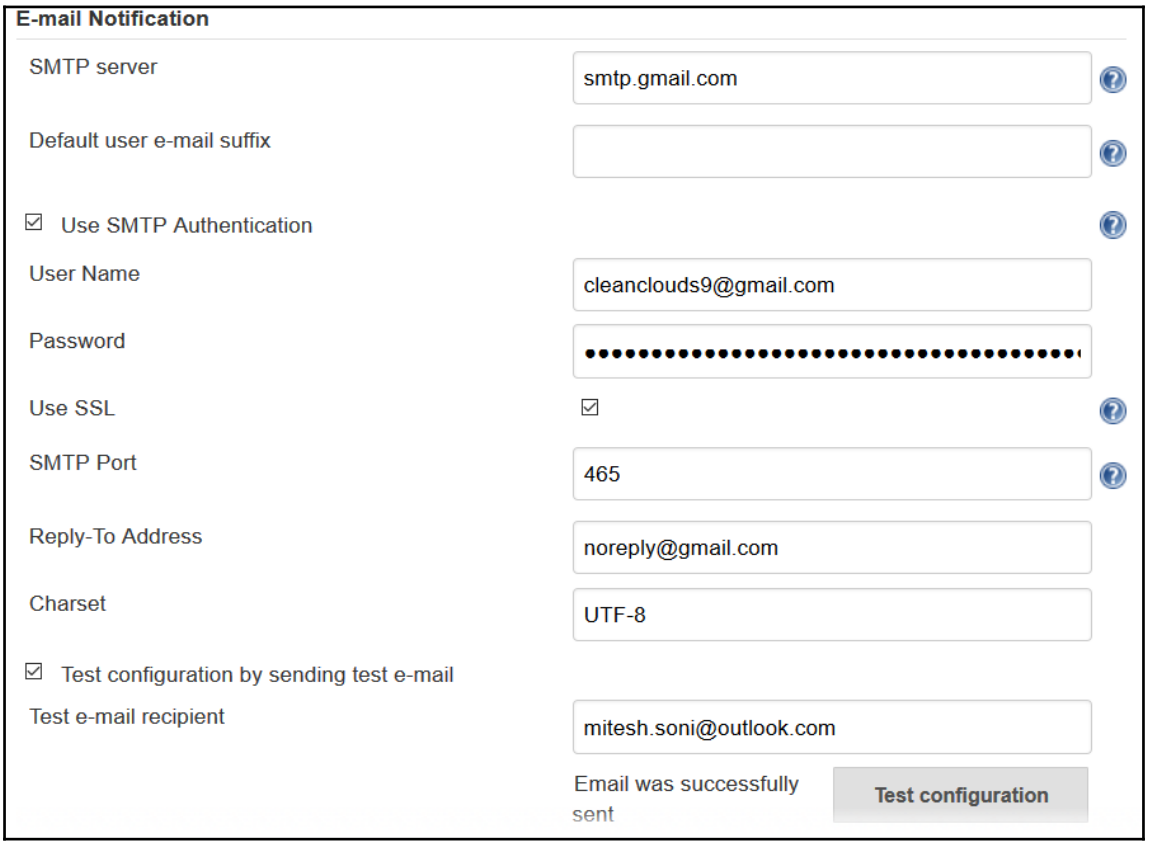
    1. Click on **Manage Jenkins** and go to **Configure System** section.

    2. Go to **E-mail Notification** sub section and enter values for **SMTP Server** and **Default user e-mail suffix**.

    3. Select **Use SMTP Authentication** checkbox, enter **User Name** and **Password**.

    4. Select **Use SSL** checkbox, enter **SMTP Port**, **Reply-To Address**.

    5. Finally select **Test configuration by sending test e-mail**. If Email configurations are correct then you will find a message **Email was successfully sent**.



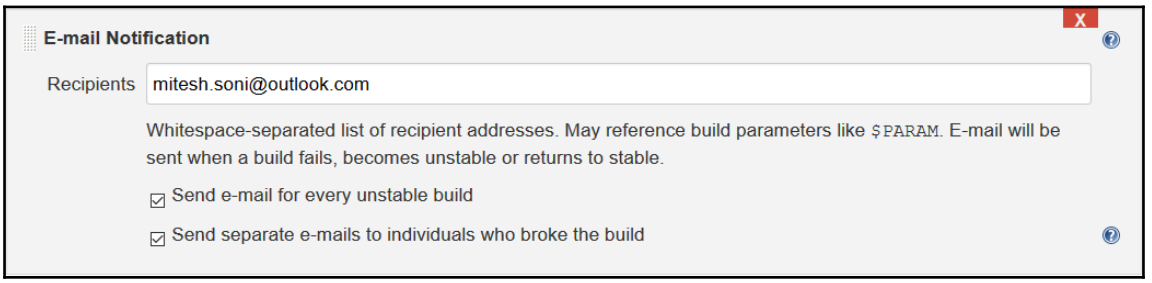
6. To verify Email notifications, simulate failure in one of the build job. Open any build job and click on **Configure**.

7. In **Post-build** Actions, click on **Add post-build** action

3. Select **E-mail Notification**.

4. Enter list of **Recipients**.

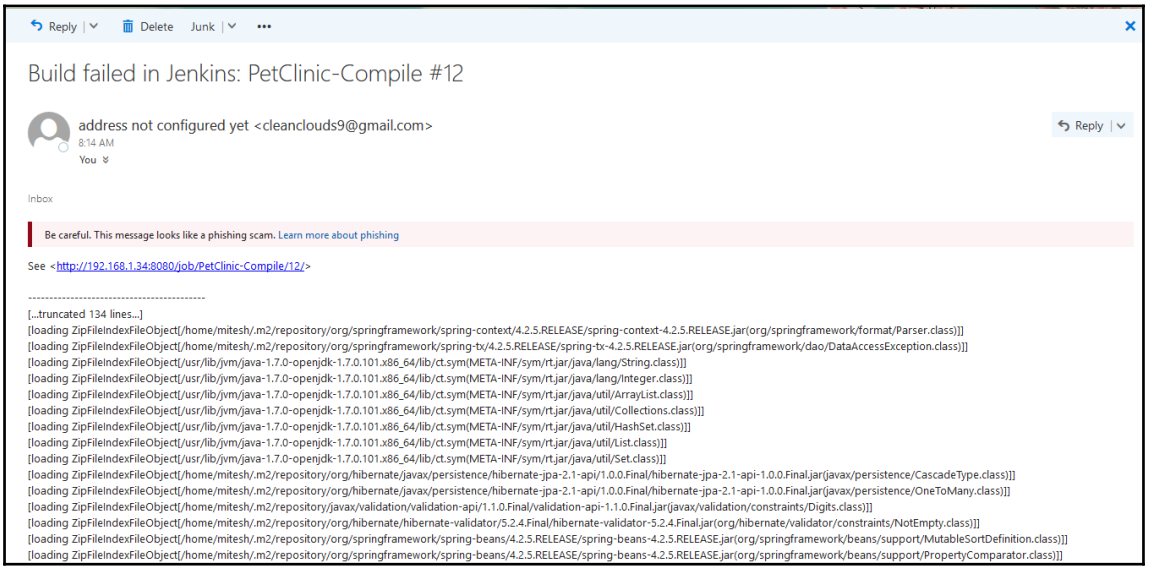
5. Select **Send e-mail for every unstable build** and **Send separate e-mails to individuals who broke the build**.



In our case, we execute compile goal against Maven build and we wanted to publish JUnit Test result to simulate failure. We can see that compilation of files are successful but Post-build action fails and it triggers Email notification based on the configuration.

```
[INFO] ------------------------------------------------------------------------
[INFO] BUILD SUCCESS
[INFO] ------------------------------------------------------------------------
[INFO] Total time: 21.332 s
[INFO] Finished at: 2016-04-28T19:44:26-07:00
[INFO] Final Memory: 25M/134M
[INFO] ------------------------------------------------------------------------
Recording test results
ERROR: Step 'Publish JUnit test result report' failed: No test report files were
found. Configuration error?
Sending e-mails to: mitesh.soni@outlook.com
Finished: FAILURE
```

Following is the Email received from Jenkins build job failure. It contains stack trace of the execution.



Let's consider a scenario where we want to send customized content in the mail. How to achieve that?

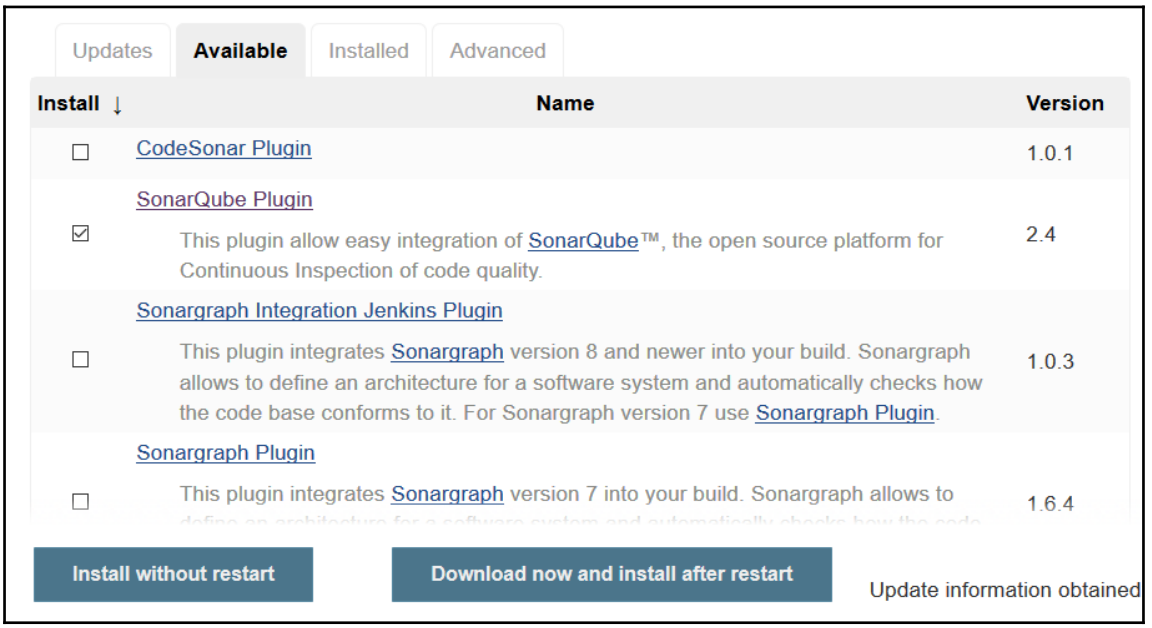
Hint: Configure Extended E-mail Notification. Try it as an exercise.

Jenkins and Sonar integration

SonarQube is an open source tool to manage code quality of an application. It manages seven axes of code quality such as Architecture & Design, Duplications, Unit Tests, Potential Bugs, Complexities, Coding Rules, and Comments. It covers programming languages such as ABAP, C/C++, C#, COBOL, CSS, Erlang, Flex / ActionScript, Groovy, Java, Java Properties, JavaScript, JSON, Objective-C, PHP, PL/I, PL/SQL, Puppet, Python, RPG, Swift, VB.NET, Visual Basic 6, Web, and XML. One of the striking feature is its extensibility. It is easy to cover new languages and adding rules engines using extension mechanism called plugins.

To install, SonarQube plugin,

1. Go to **Manage Jenkins**, click on **Manage Plugins**. Click on **Available** tab. Search SonarQube plugin and install it by clicking on **Install without restart**.



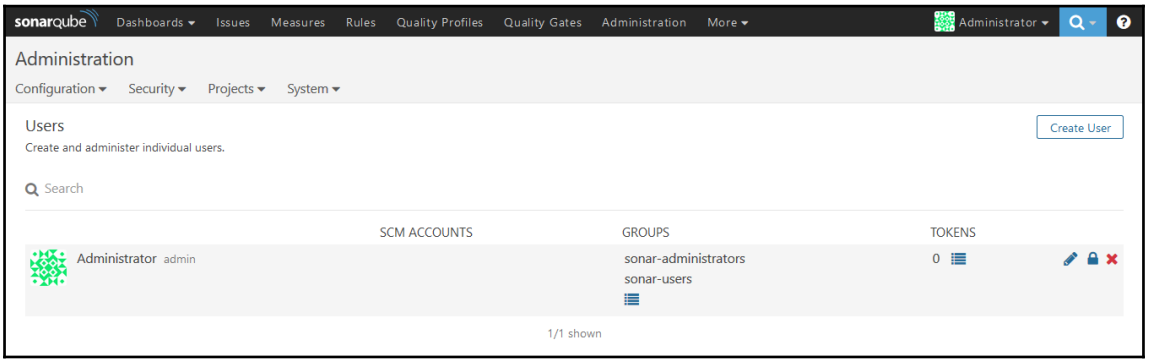
2. Download sonar from http://www.sonarqube.org/downloads/.

3. Extract installable directory from the zip file and go to bin sub-directory.

4. Based on the operating system, select the installable directory and run the StartSona.* file as shown in below image.

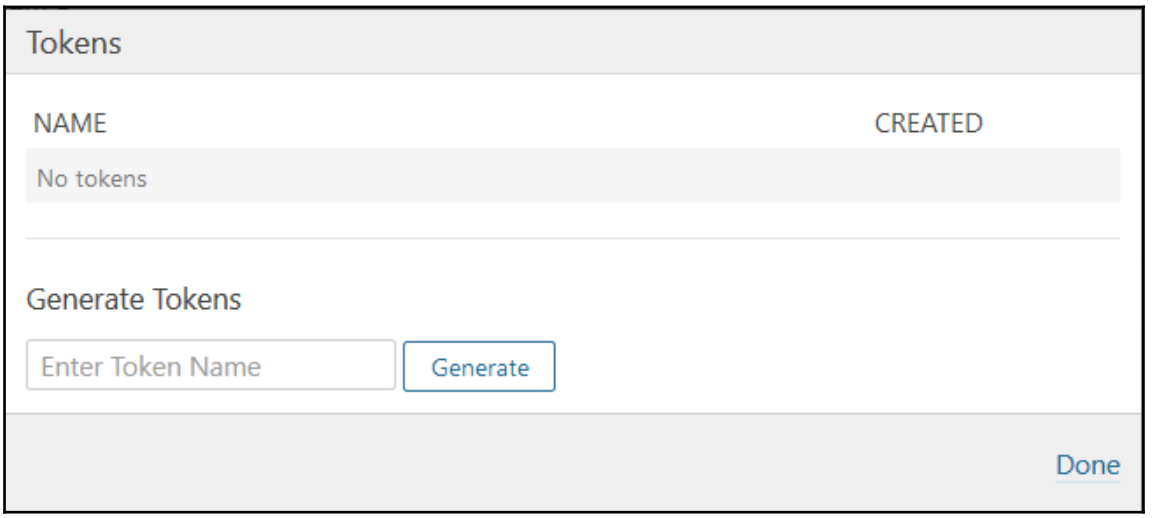
```
D:\##DevOps Book\Installables\sonarqube-5.4\bin\windows-x86-64>StartSonar.bat
wrapper  | --> Wrapper Started as Console
wrapper  | Launching a JVM...
jvm 1    | Wrapper (Version 3.2.3) http://wrapper.tanukisoftware.org
jvm 1    |   Copyright 1999-2006 Tanuki Software, Inc.  All Rights Reserved.
jvm 1    |
jvm 1    | 2016.04.29 23:57:37 INFO  app[o.s.a.AppFileSystem] Cleaning or creating temp directory D:\##DevOps Book\Install
ables\sonarqube-5.4\temp
jvm 1    | 2016.04.29 23:57:38 INFO  app[o.s.p.m.JavaProcessLauncher] Launch process[search]: C:\Program Files\Java\jre1.8
.0_45\bin\java -Djava.awt.headless=true -Xmx1G -Xms256m -Xss256k -Djava.net.preferIPv4Stack=true -XX:+UseParNewGC -XX:+Use
ConcMarkSweepGC -XX:CMSInitiatingOccupancyFraction=75 -XX:+UseCMSInitiatingOccupancyOnly -XX:+HeapDumpOnOutOfMemoryError -
Djava.io.tmpdir=D:\##DevOps Book\Installables\sonarqube-5.4\temp -cp ./lib/common/*;./lib/search/* org.sonar.search.Search
Server C:\Users\MItesh\AppData\Local\Temp\sq-process7000722619322287622properties
jvm 1    | 2016.04.29 23:57:49 INFO  app[o.s.p.m.Monitor] Process[search] is up
jvm 1    | 2016.04.29 23:57:49 INFO  app[o.s.p.m.JavaProcessLauncher] Launch process[web]: C:\Program Files\Java\jre1.8.0_
45\bin\java -Djava.awt.headless=true -Dfile.encoding=UTF-8 -Djruby.management.enabled=false -Djruby.compile.invokedynamic=
false -Xmx768m -Xms256m -XX:MaxPermSize=160m -XX:+HeapDumpOnOutOfMemoryError -Djava.net.preferIPv4Stack=true -Djava.io.tmp
dir=D:\##DevOps Book\Installables\sonarqube-5.4\temp -cp ./lib/common/*;./lib/server/*;D:\##DevOps Book\Installables\sonar
qube-5.4\lib\jdbc\h2\h2-1.3.176.jar org.sonar.server.app.WebServer C:\Users\MItesh\AppData\Local\Temp\sq-process3019138822
364693273properties
jvm 1    | 2016.04.29 23:59:07 INFO  app[o.s.p.m.Monitor] Process[web] is up
```

5. Once Sonar is up and running, Open browser and visit http://localhost:9000/ or http://<IP_Address>:9000/. We will get the Sonar dashboard.

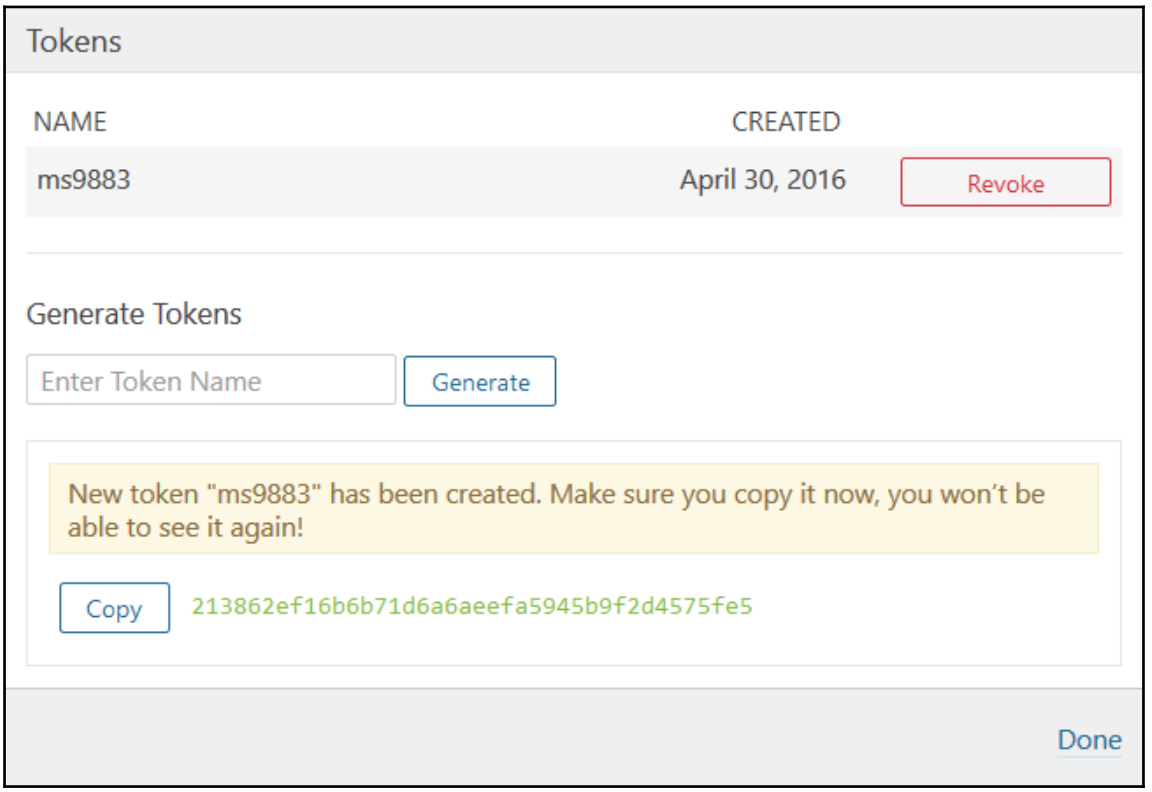


One of the important step for Jenkins 2 and Sonar integration is security token.

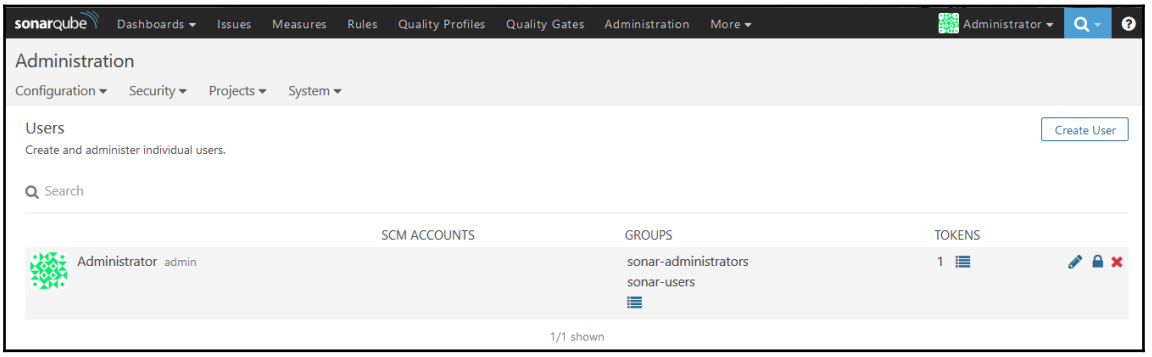
1. Go to **My Account** link on top right corner.
2. Click on **security** tab and **Generate Tokens**.



3. Enter Token name and click on **Generate**. Copy the token value and click on **Done**.

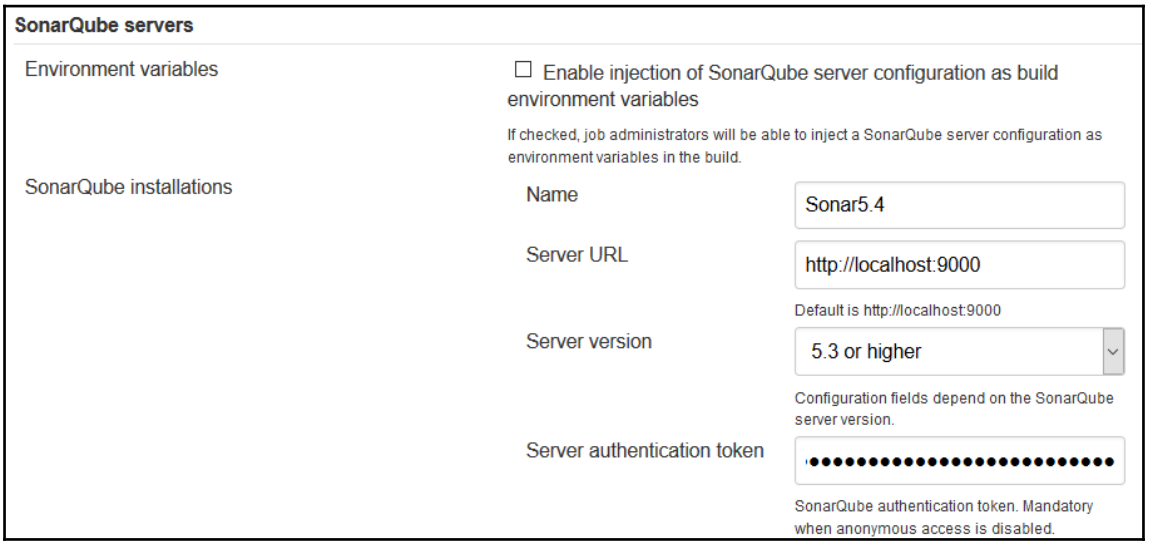


4. Verify the Tokens column in the Sonar dashboard.

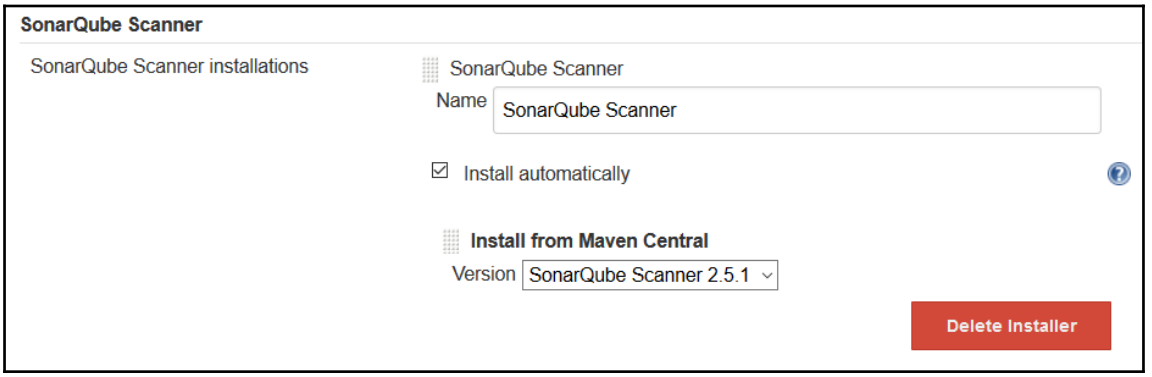


Once we have a security token ready, next step is to integrate Jenkins and Sonar.

1. In **Manage Jenkins** section, click on **Configure System** and add **SonarQube servers**. Here provide **Server URL** and security token and save the settings.



2. In **Global Tool Configuration**, configure **SonarQube Scanner** installations also.



Once all Sonar related installations and configurations are completed, we need to add Build step to execute SonarQube Scanner. Run the build Job.

1. We need sonar-project.properties for Sonar configuration with specific application. In our sample application, sonar-project.properties file is already available.

```
# Required metadata
sonar.projectKey=java-sonar-runner-simple
sonar.projectName=Simple Java project analyzed with the SonarQube Runner
sonar.projectVersion=1.0

# Comma-separated paths to directories with sources (required)
sonar.sources=src

# Language
sonar.language=java

# Encoding of the source files
sonar.sourceEncoding=UTF-8
```

2. Verify the console output of a build job for Sonar execution.

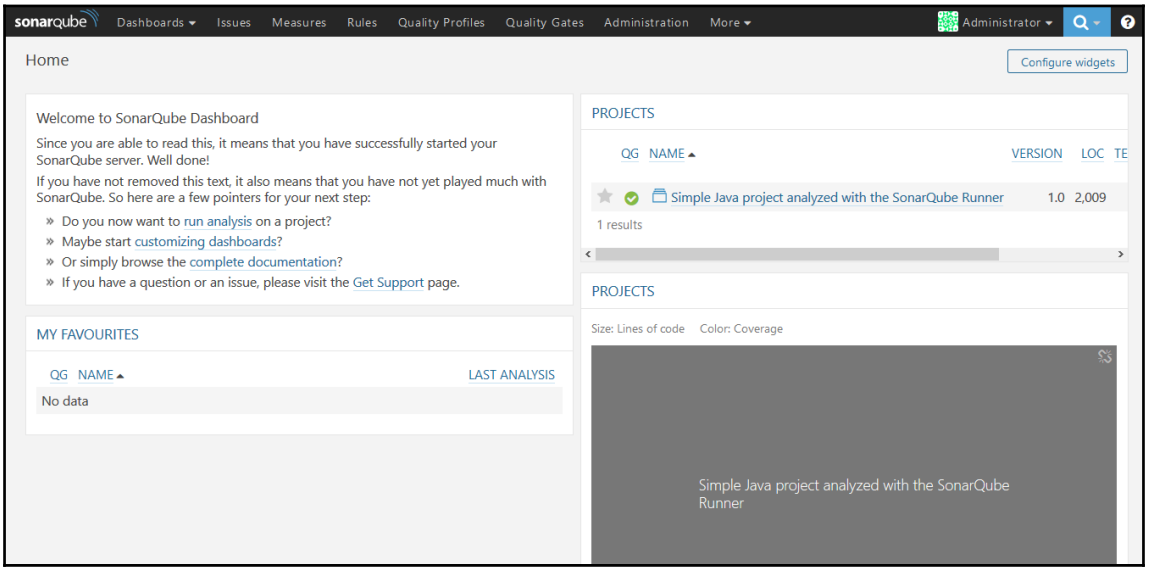
```
  D:\##DevOps Book\Installables\sonar-scanner-2.6
  INFO: Scanner configuration file: D:\##DevOps Book\Installables\sonar-
scanner-2.6\conf\sonar-scanner.properties
    INFO: Project root configuration file: d:\jenkins\workspace\PetClinic-Test\sonar-
project.properties
  INFO: SonarQube Scanner 2.6
  INFO: Java 1.8.0-ea Oracle Corporation (64-bit)
  INFO: Windows 8.1 6.3 amd64
  INFO: Error stacktraces are turned on.
  INFO: User cache: C:\Users\MItesh\.sonar\cache
  INFO: Load global repositories
  INFO: Load global repositories (done) | time=1131ms
  INFO: User cache: C:\Users\MItesh\.sonar\cache
  INFO: Load plugins index
  INFO: Load plugins index (done) | time=16ms
    INFO: Download sonar-csharp-plugin-4.4.jar
    INFO: Download sonar-java-plugin-3.10.jar
    INFO: Download sonar-scm-git-plugin-1.0.jar
    INFO: Download sonar-scm-svn-plugin-1.2.jar
    INFO: Download sonar-javascript-plugin-2.10.jar
    INFO: SonarQube server 5.4
  INFO: Default locale: "en_US", source code encoding: "UTF-8"
  INFO: Process project properties
  INFO: Load project repositories
  INFO: Load project repositories (done) | time=133ms
  INFO: Apply project exclusions
  INFO: Load quality profiles
  INFO: Load quality profiles (done) | time=927ms
  INFO: Load active rules
  INFO: Load active rules (done) | time=4068ms
  INFO: Publish mode
    INFO: ------------  Scan Simple Java project analyzed with the SonarQube Runner
```

```
    INFO: Language is forced to java
    INFO: Load server rules
    INFO: Load server rules (done) | time=656ms
    INFO: Base dir: d:\jenkins\workspace\PetClinic-Test
      INFO: Working dir: d:\jenkins\workspace\PetClinic-Test\.sonar
      INFO: Source paths: src
      INFO: Source encoding: UTF-8, default locale: en_US
      INFO: Index files
      INFO: 56 files indexed
    INFO: Quality profile for java: Sonar way
    INFO: JaCoCoSensor: JaCoCo report not found : d:\jenkins\workspace\PetClinic-
Test\target\jacoco.exec
    INFO: JaCoCoItSensor: JaCoCo IT report not found: d:\jenkins\workspace\PetClinic-
Test\target\jacoco-it.exec
    INFO: Sensor JavaSquidSensor
    INFO: Configured Java source version (sonar.java.source): none
    INFO: JavaClasspath initialization...
    INFO: Bytecode of dependencies was not provided for analysis of source files, you might
end up with less precise results. Bytecode can be provided using sonar.java.libraries property
    INFO: JavaClasspath initialization done: 1 ms
    INFO: JavaTestClasspath initialization...
    INFO: Bytecode of dependencies was not provided for analysis of test files, you might end
up with less precise results. Bytecode can be provided using sonar.java.test.libraries property
    INFO: JavaTestClasspath initialization done: 1 ms
      INFO: Java Main Files AST scan...
      INFO: 56 source files to be analyzed
    INFO: 46/56 files analyzed, current file: d:\jenkins\workspace\PetClinic-
Test\src\test\java\org\springframework\samples\petclinic\service\AbstractClinicServiceTests.ja
va
    INFO: Java Main Files AST scan done: 12107 ms
    INFO: Java bytecode has not been made available to the analyzer. The
org.sonar.java.bytecode.visitor.DependenciesVisitor@4f1150f5,
org.sonar.java.checks.unused.UnusedPrivateMethodCheck@3fba233d are disabled.
    INFO: Java Test Files AST scan...
    INFO: 0 source files to be analyzed
    INFO: Java Test Files AST scan done: 1 ms
    INFO: Sensor JavaSquidSensor (done) | time=15295ms
    INFO: Sensor Lines Sensor
      INFO: 56/56 source files have been analyzed
    INFO: 0/0 source files have been analyzed
    INFO: Sensor Lines Sensor (done) | time=28ms
    INFO: Sensor QProfileSensor
    INFO: Sensor QProfileSensor (done) | time=29ms
    INFO: Sensor SurefireSensor
    INFO: parsing d:\jenkins\workspace\PetClinic-Test\target\surefire-reports
    INFO: Sensor SurefireSensor (done) | time=531ms
    INFO: Sensor SCM Sensor
    INFO: SCM provider for this project is: git
```

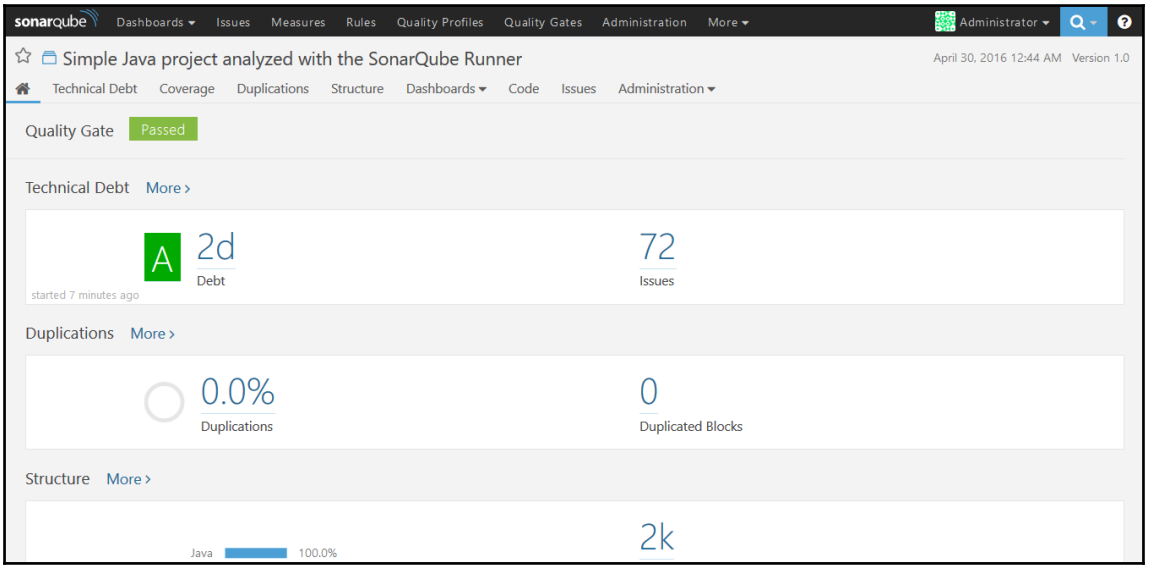
```
INFO: 56 files to be analyzed
INFO: 56/56 files analyzed
INFO: Sensor SCM Sensor (done) | time=3754ms
INFO: Sensor Code Colorizer Sensor
INFO: Sensor Code Colorizer Sensor (done) | time=9ms
INFO: Sensor CPD Sensor
INFO: JavaCpdIndexer is used for java
INFO: Sensor CPD Sensor (done) | time=303ms
INFO: Analysis report generated in 1055ms, dir size=294 KB
  INFO: Analysis reports compressed in 629ms, zip size=191 KB
  INFO: Analysis report uploaded in 524ms
  INFO: ANALYSIS SUCCESSFUL, you can browse
    http://localhost:9000/dashboard/index/java-sonar-runner-simple
  INFO: Note that you will be able to access the updated dashboard once the server has
processed the submitted analysis report
  INFO: More about the report processing at
    http://localhost:9000/api/ce/task?id=AVRjchhfszI1jSgY1AZe
INFO: ------------------------------------------------------------------------
  INFO: EXECUTION SUCCESS
INFO: ------------------------------------------------------------------------
INFO: Total time: 57.737s
INFO: Final Memory: 52M/514M
INFO: ------------------------------------------------------------------------
  Recording test results
  Finished: SUCCESS
```

3. Let's verify the Sonar UI at
http://localhost:9000/dashboard/index/java-sonar-runner-simple

4. In the **Projects** section, we can find project details available now. Click on the project name.

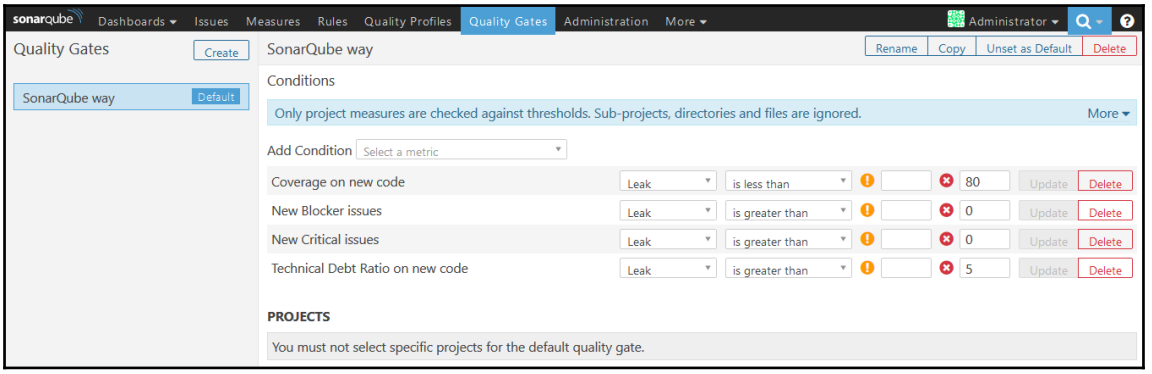


5. We can see the result of analysis here. Quality Gate is passed. It provides details about **Technical Debt**, **Duplications**, **and Structure** too.

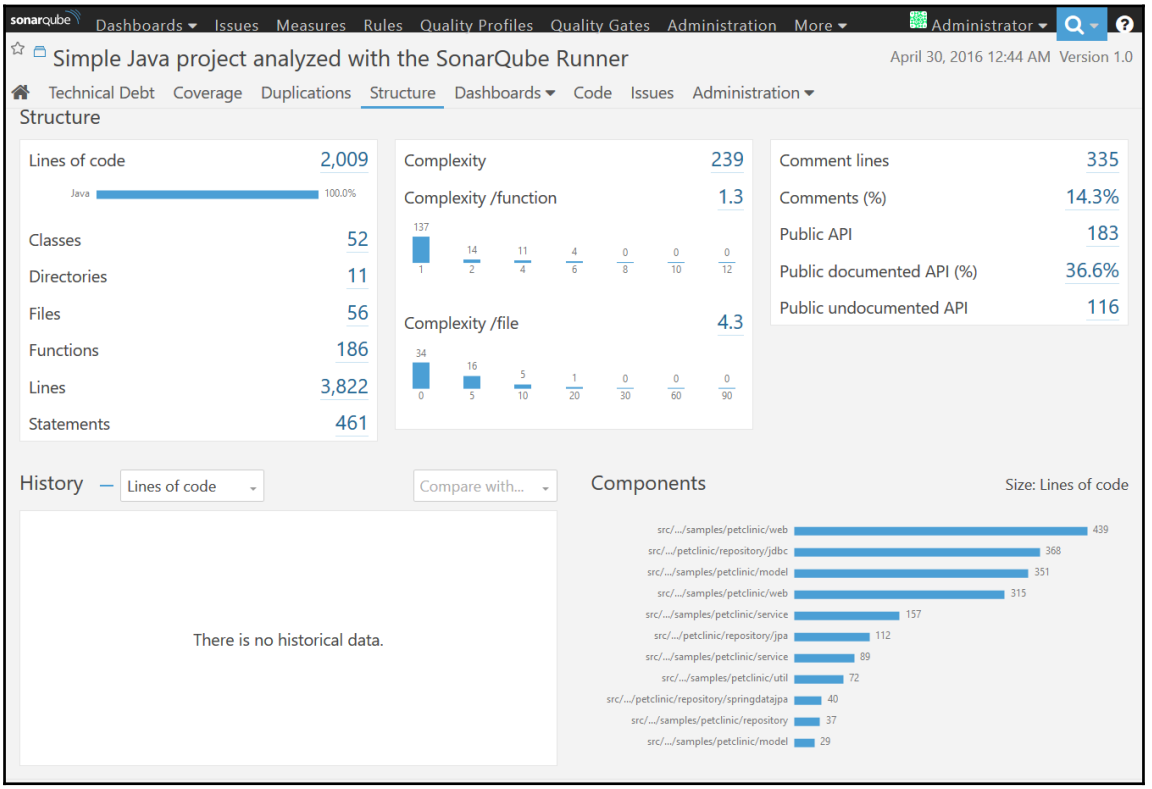


6. **Quality Gates** can be defined in the Sonar dashboard. We have used default

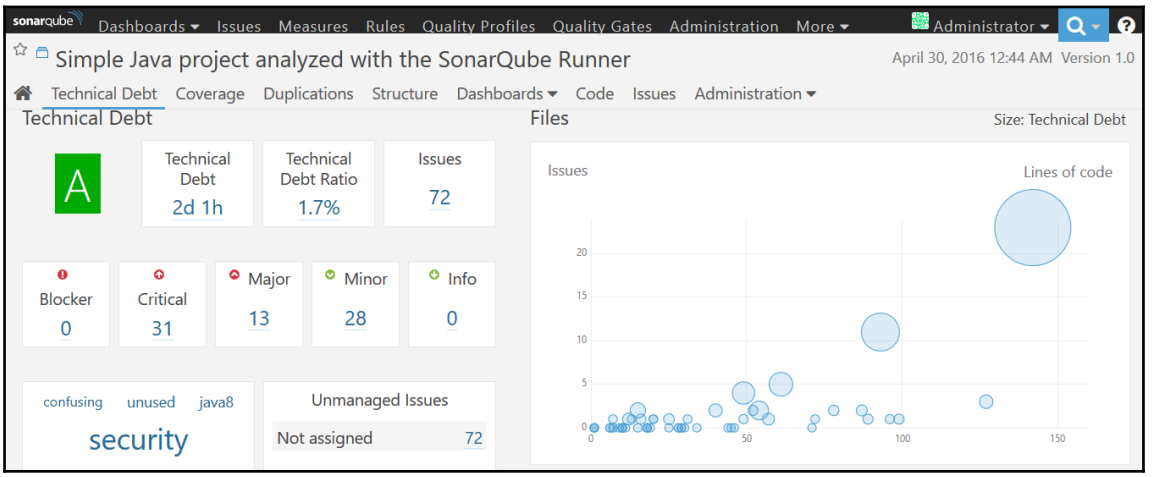
quality gate here.



7. To verify **Lines of code**, **Complexity**, and **Comment lines** click on **Structure** tab in Sonar dashboard.



8. To get more insights into issues in specific files, click on **Technical Debt** tab and click on Bubbles available in the chart.



- Sonar stores historical data in 24-hour slices.

Self-Test Questions

1. State whether following statement is True or False: Jenkins is written in Java.

    1. True
    2. False

2. On which of the following operating systems Jenkins can be installed?

    1. Ubuntu/Debian
    2. Windows
    3. Mac OS X
    4. CentOS/Fedora/Red Hat
    5. All of the above

3. Which of the following command can be used to change the default port on which Jenkins is running?

1. java -jar jenkins.war –httpPort=9999
2. java -jar jenkins.war –http=9999
3. java -jar jenkins.war –https=9999
4. java -jar jenkins.war –httpsPort=9999

4. State whether following statement is True or False: Sonar stores historical data in 22-hour slices

1. True
2. False

Summary

In this chapter, we have learnt about some new features in Jenkins 2, why Jenkins is so popular, how to install Jenkins, what are improvements with respect to security and plugin installations while setup, how to configure Java and Maven, what happens in the background when we create a new job in Jenkins, how to authenticate with Git, how to configure Git in Jenkins, unit test execution in sample spring application, how to configure dashboard view plugin with different portlets for customized view, how to manage master and slave node for load distribution and managing different environment as per need, how to configure E-mail notifications for build status, and how to integrate sonar and Jenkins.

In the next chapter, we will see one of the most important aspect in terms of orchestration of end to end pipeline of application delivery. We will discuss Pipeline concept of Jenkins 2 and Build Pipeline Plugin.

It is a right time to quote Ralph Waldo Emerson as it is relevant in the context of failures while build execution in the process of Continuous Integration:

> *"Our greatest glory is not in never failing, but in rising up every time we fail."*

3
Building the Code and Configuring Build Pipeline

"Start wide, expand further, and never look back."

– Arnold Schwarzenegger

It is always better to start early and visualize the things which we want to achieve. That is the objective of this chapter. It is easy to visualize the end or realize the importance of this chapter when we will be ending the last line of last chapter of this book. One of the Highlights of Jenkins 2 release is Built-in support for delivery pipelines. We know that Jenkins is a Continuous Integration server but what if we want to use it for Continuous Delivery or Continuous Deployment too? Automation and Orchestration both are equally important while dealing with application delivery pipeline.

This chapter describes in detail how to create pipeline of different jobs for a sample JEE application. It will also cover deployment of an application to local web/application server and configuration of Build pipeline for lifecycle of continuous integration. This way Jenkins users can model application delivery pipelines as code. Once we can make it as a code then we can store in code repository and it can be managed in a better way. One important benefit is collaboration. As it can be stored in version control, different teams can reuse it for different operations based on the environment.

Readers will learn how to manage lifecycle of continuous integration including pulling code from code repository, building the code, unit test execution, and static code analysis using different jobs.

In this chapter, we will cover the following topics:

- Built-in Delivery Pipelines of Jenkins 2

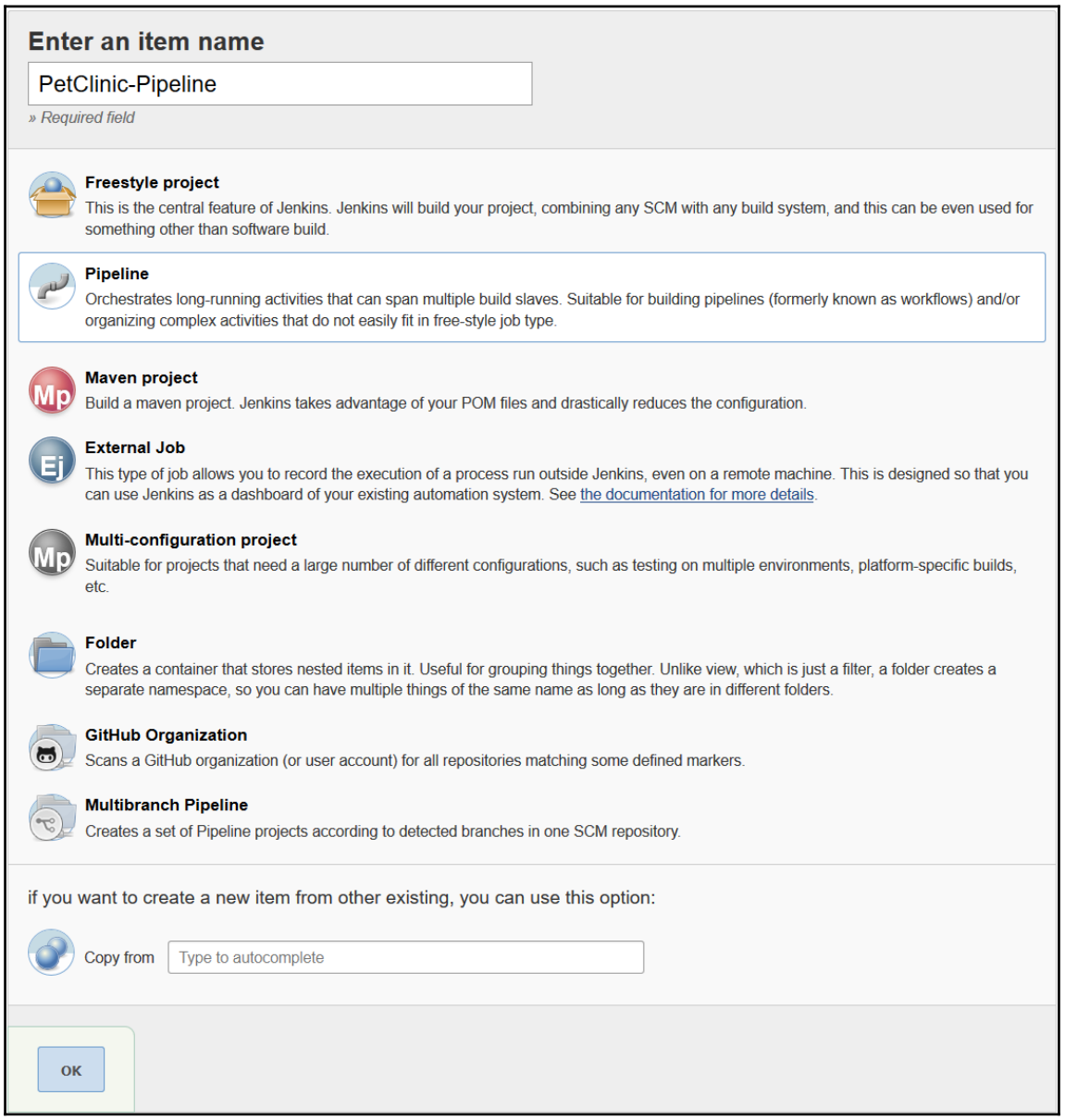
- Build Pipeline Configuration for End to End Automation to manage lifecycle of continuous integration
- Deploying a WAR file from Jenkins to Local Tomcat Server

Creating Built-in Delivery Pipelines

Jenkins 2 provides a way to create delivery pipelines using a Domain-Specific Language (DSL).

Steps for creating Built-in Delivery Pipeline are as follows:

1. Go to Jenkins dashboard and click on **New Item**.
2. Enter an item name and select **Pipeline** as shown in below image.
3. Click on **OK**.

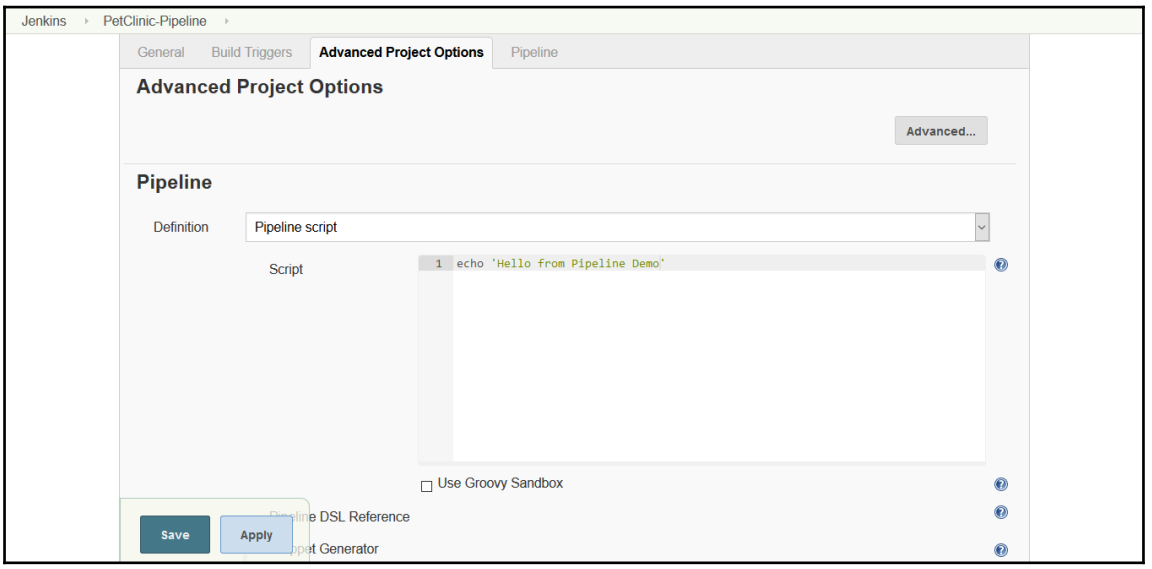


4. In case you have existing Pipeline available then you can create new pipeline by copying from it.

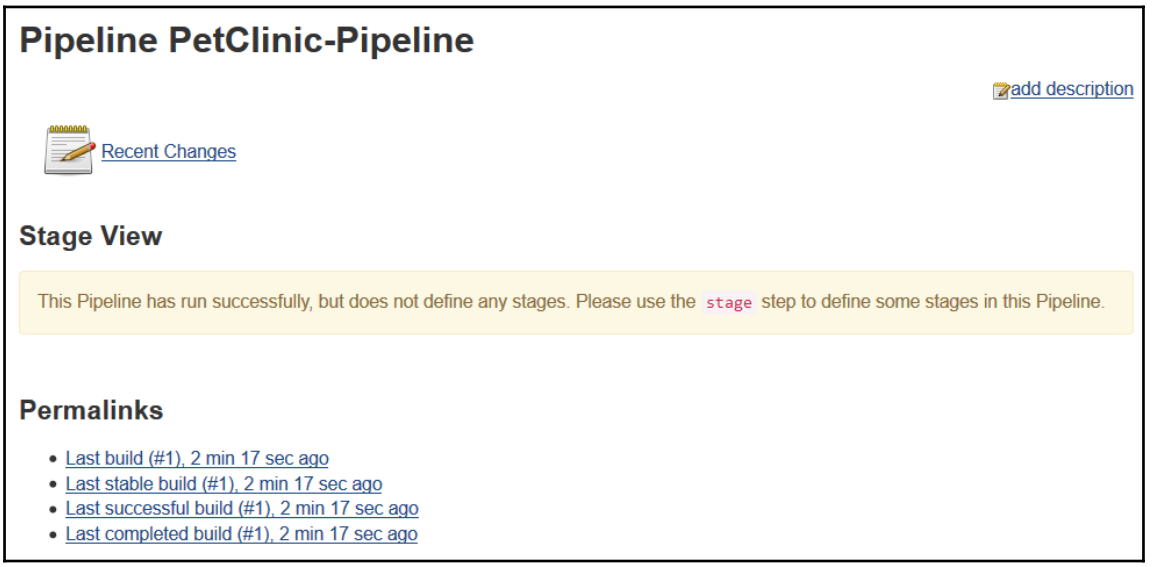
5. Go to **Advanced Project Options**. For the learning purpose, input echo

'Hello from Pipeline Demo' in the Script box.

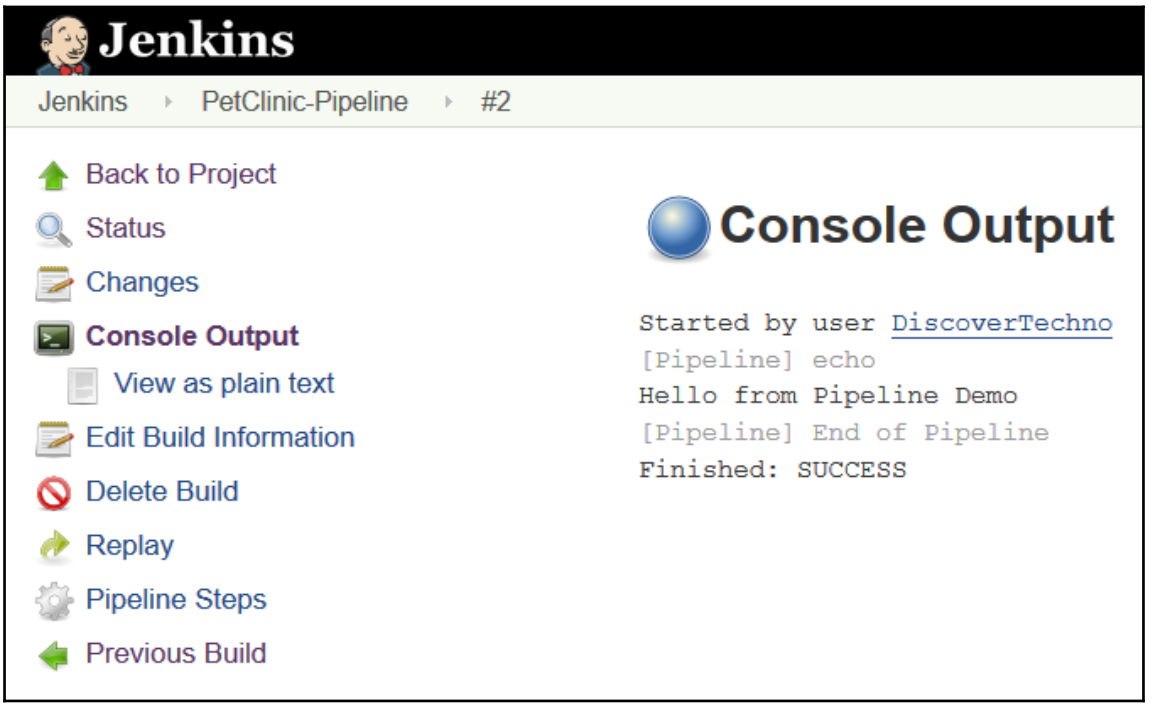
6. Click on **Save** to save the configuration.



7. As we have not created any stage, we will get warning as shown in the below image. However, we can execute the pipeline for demo purpose.



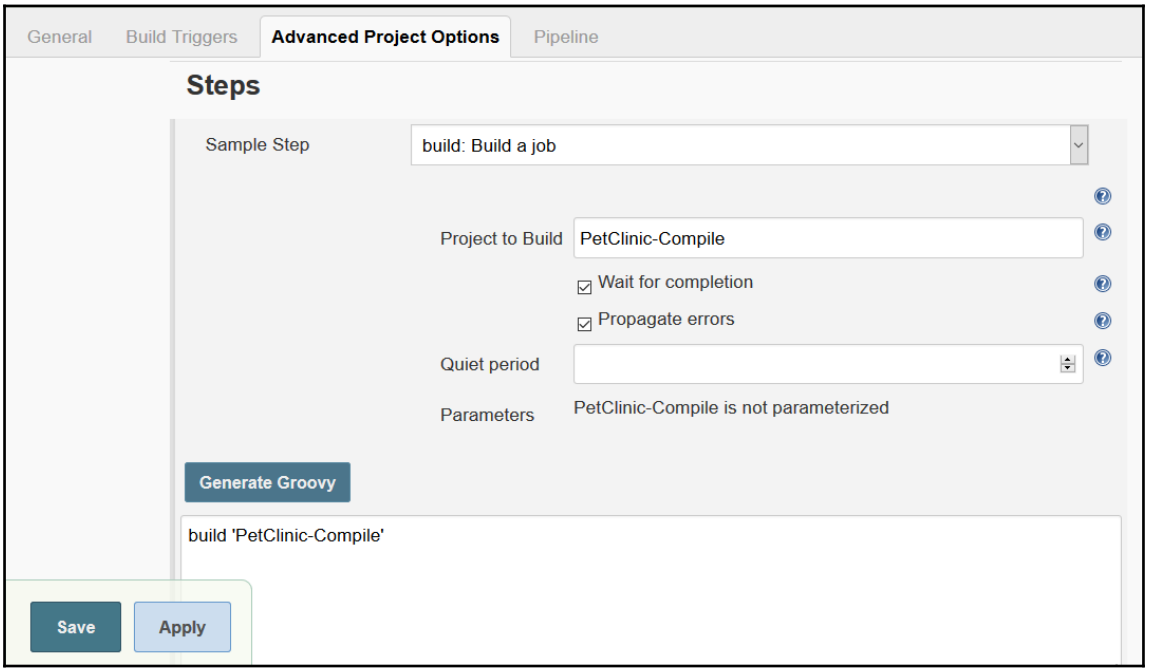
8. Click on the **Build Now**. Verify the **Console Output**. We can see the successful completion of script execution.



Let's go step by step and learn how we can create script. To make things easier refer Pipeline DSL Reference or use Snippet Generator. Select the checkbox and then select a **Sample Step**. Provide specific parameters required by the step and click on **Generate Groovy**.

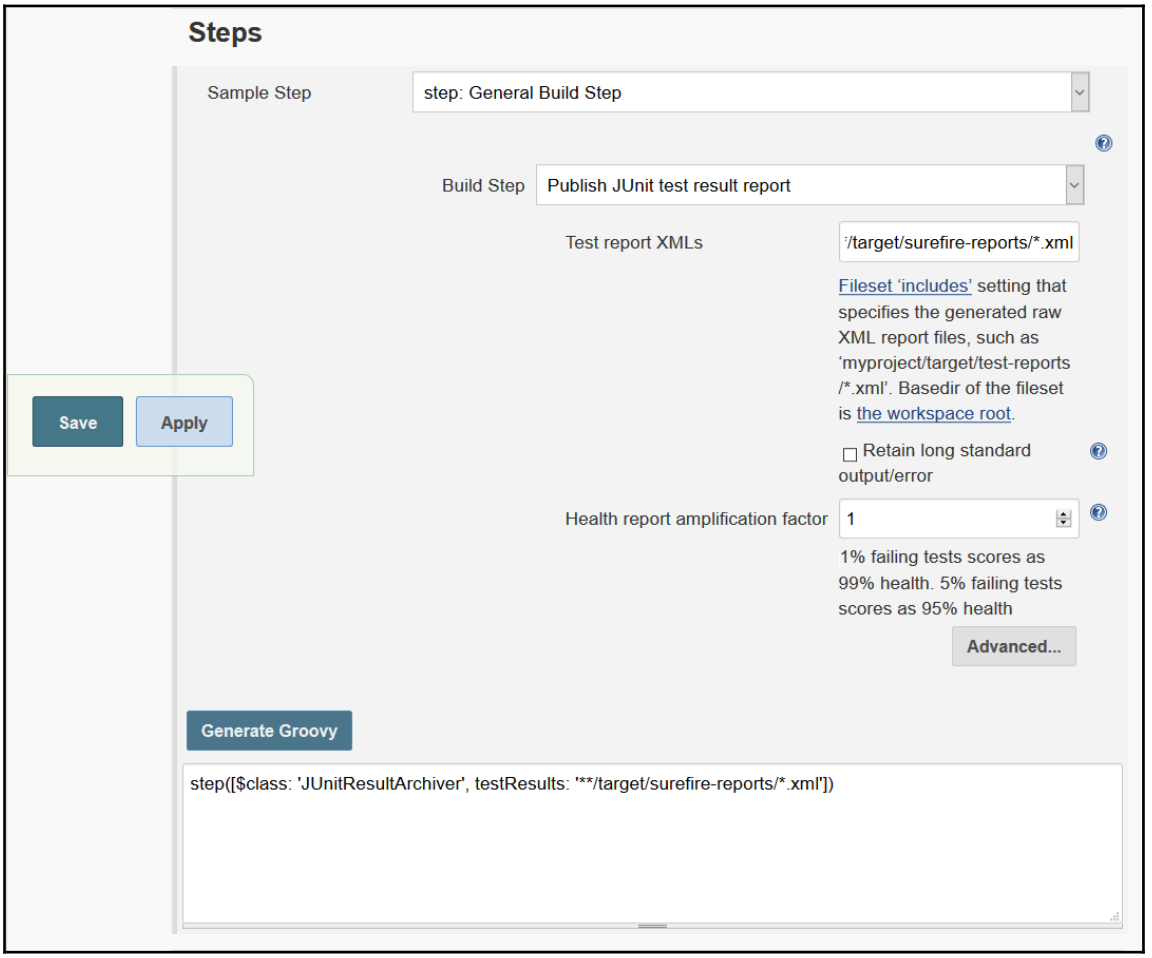
Example 1: Groovy script to build a job. It triggers a new downstream job to build.

Sample Step	**build: Build a Job**
Parameters	**Project to build**: PetClinic-Compile Parameters: None Other configurations: Default



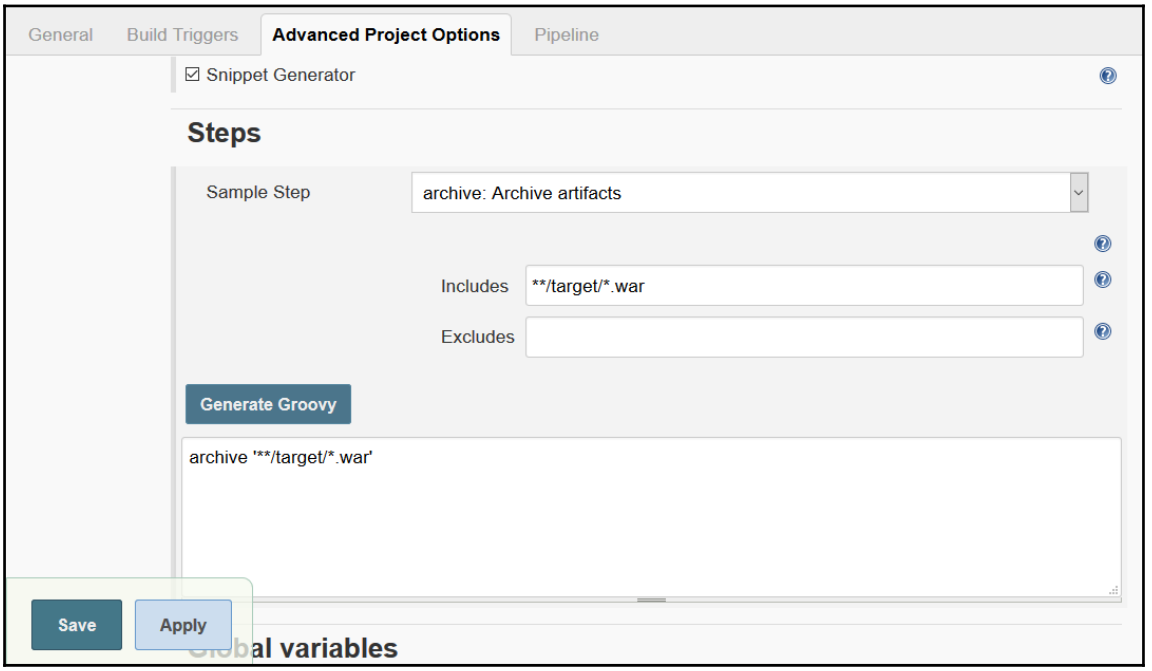
Example 2: Create a step – Generate a build step. It is used to configure post build actions or in general build step that are Pipeline-compatible based on the dropdown list.

Sample Step	**step: General Build Setup**
Parameters	**Build Step**: **Publish JUnit test result report** **Test Report XMLs**: **/target/surefire-reports/TEST-*.xml Other configurations: Default



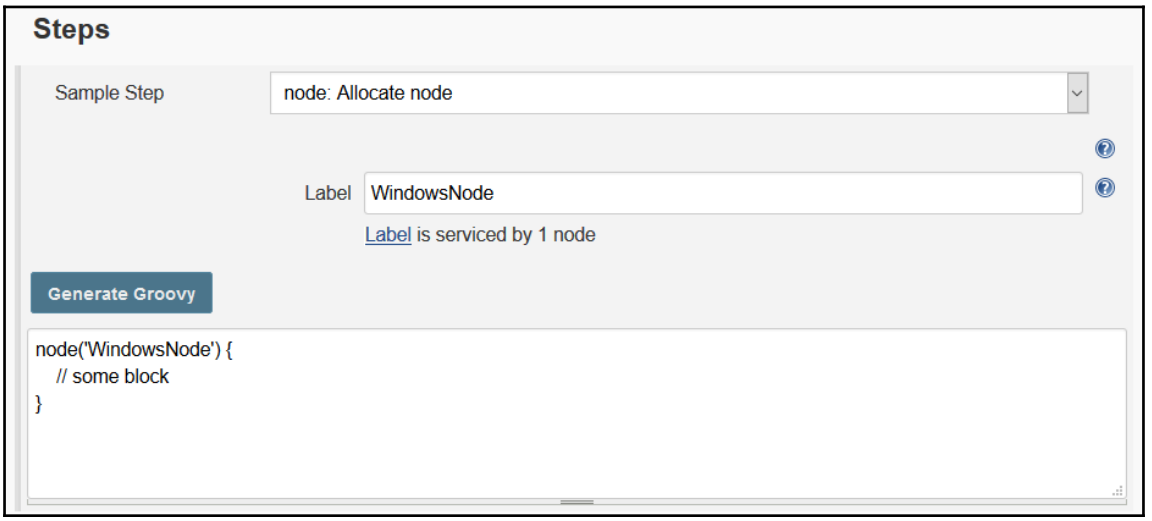
Example 3: To archive build job artifacts.

Sample Step	**archive: Archive artifacts**
Parameters	**Includes**: It includes artifacts using comma separated list matching Ant style pattern for archiving artifacts. **Excludes**: It excludes artifacts using comma separated list matching Ant-style pattern for not archiving artifacts.



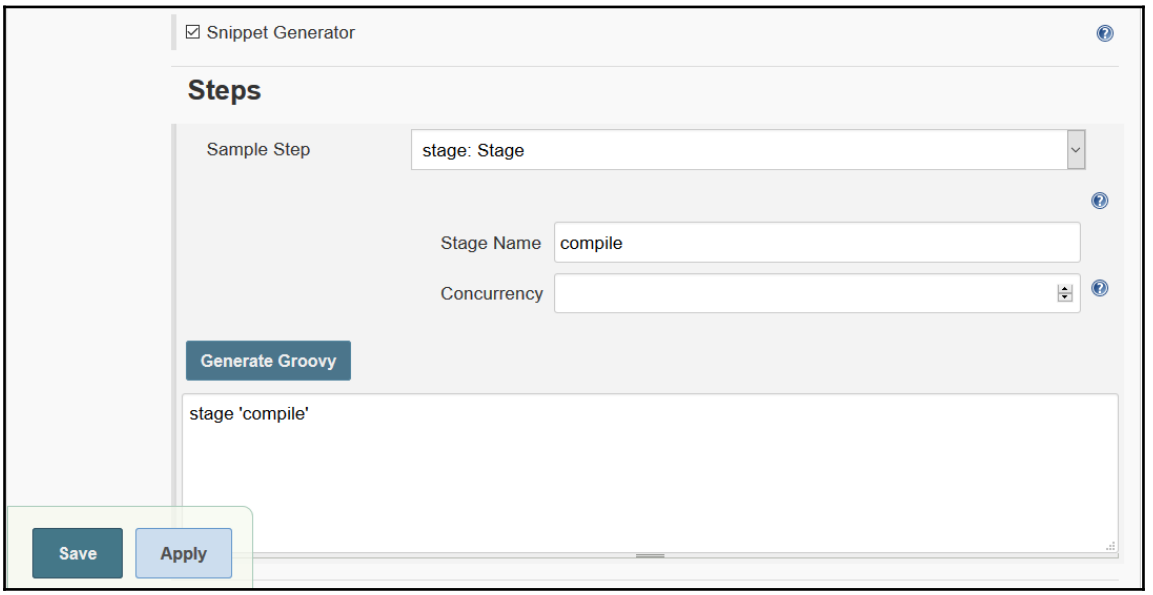
Example 4: For example, to run build step on a specific node, we need to write a script. Use Snippet Generator and select sample step node and select the slave node label. Click on **Generate Groovy**.

Sample Step	**node: Allocate node**
Parameters	**Label**: Label associated with slave node. Refer to Chapter 2, *Continuous Integration with Jenkins.* for more details on Master Slave nodes in Jenkins 2.



Example 5: Groovy script to mark definite sections of a build as being controlled by limited concurrency.

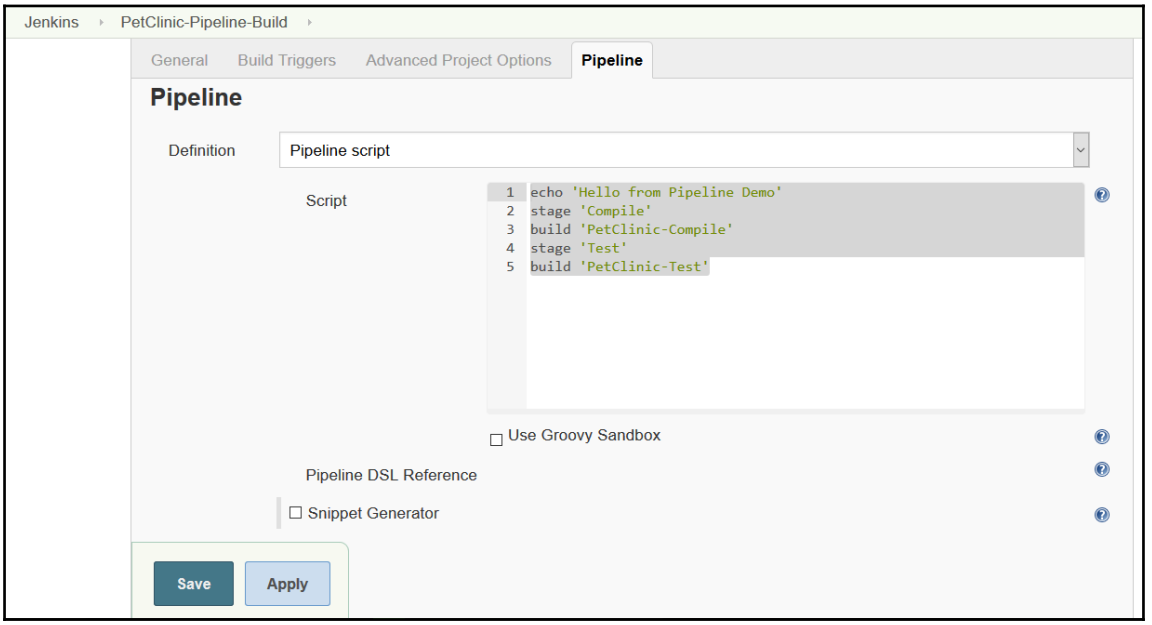
Sample Step	**stage: Stage**
Parameters	Stage Name: Compile / Test / Deploy Other configurations: Default



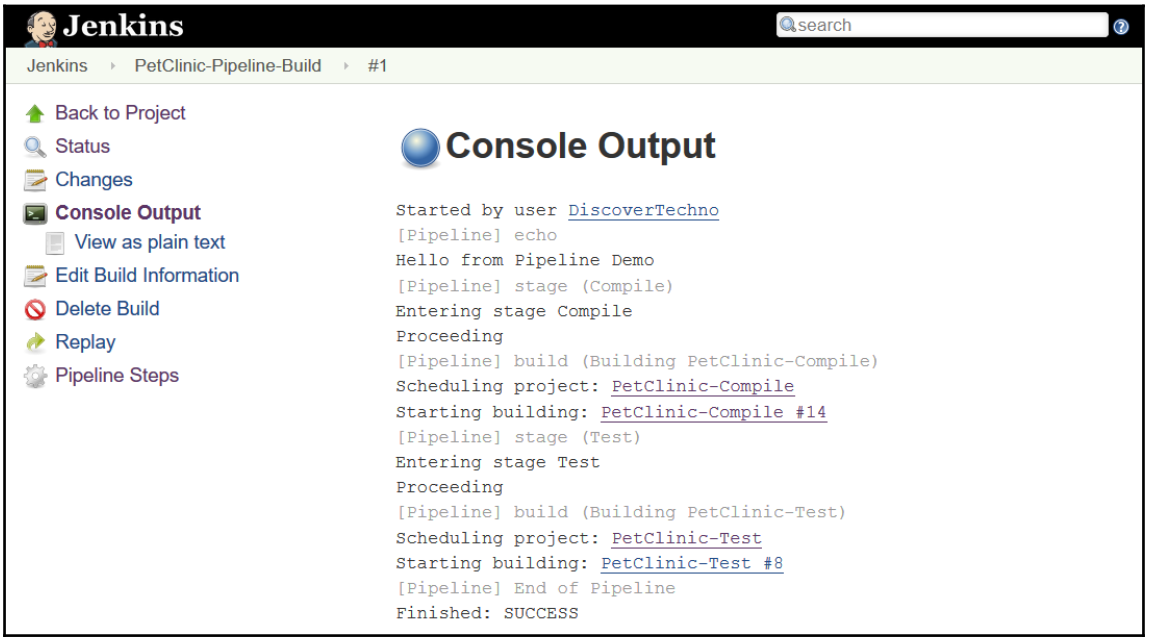
For test purpose let's try a simple scenario to create pipeline for compiling source files and executing unit test cases.

1. Let's write below script in the Script Box.

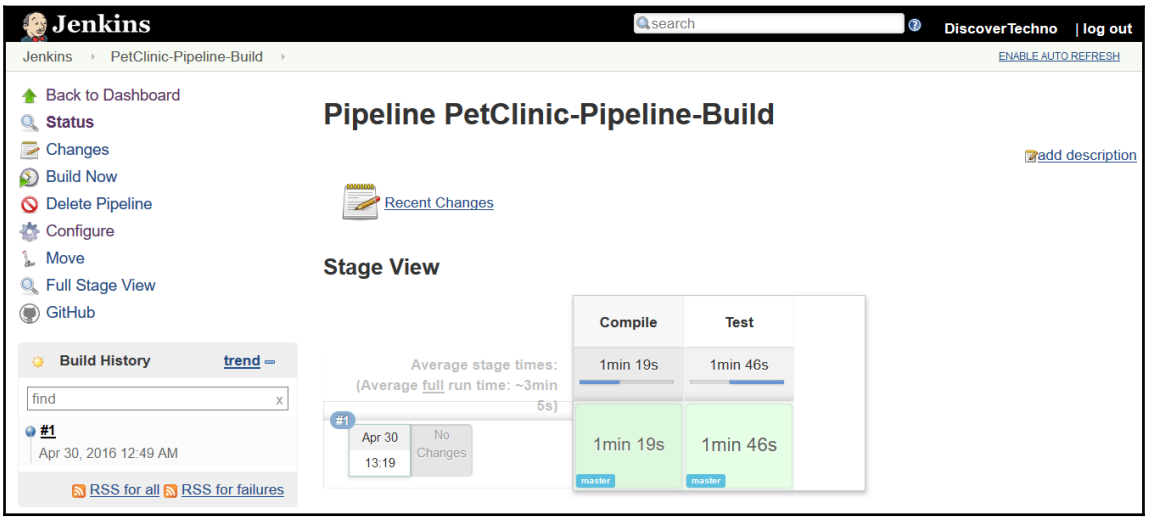
```
echo 'Hello from Pipeline Demo'
stage 'Compile'
build 'PetClinic-Compile'
stage 'Test'
build 'PetClinic-Test'
```



2. Click on the **Build Now** and go to **Console Output** to verify the execution process.

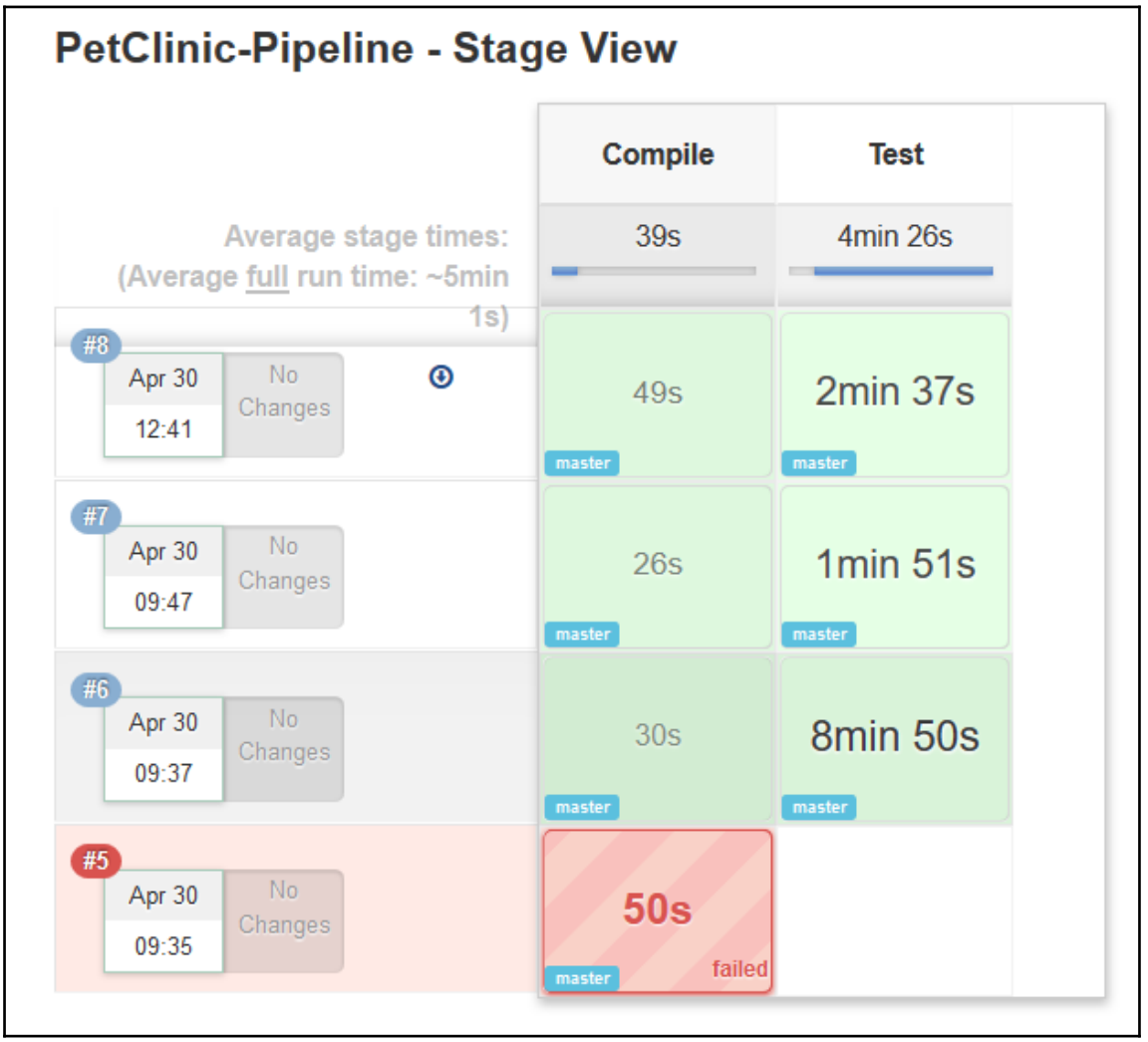


3. Go to Build Job's main page. We can see **Stage view** here. Remember, we have created two stages, one is compile and another is test. Stage view provides instant visualization. It provides details such as build completion time, on which node build has been executed, build has been failed or executed successfully.

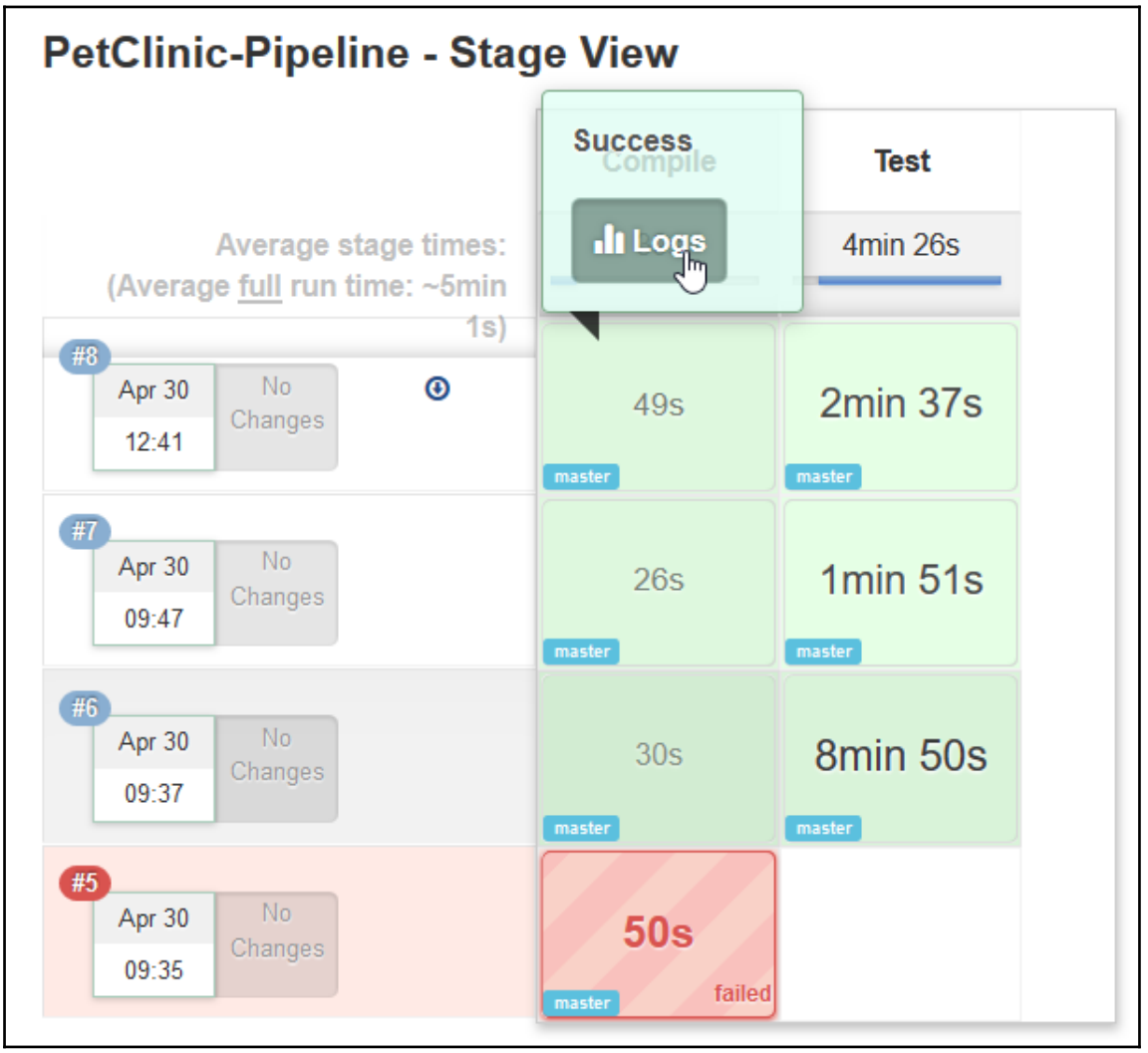


4. In the specific build execution, we can verify Pipeline Steps also.

5. Click on **Full stage view** to get full screen view as shown in below image:



6. To get details specific to stage, mouse over specific stage and it will show us status of that stage execution as well as **Logs** link.

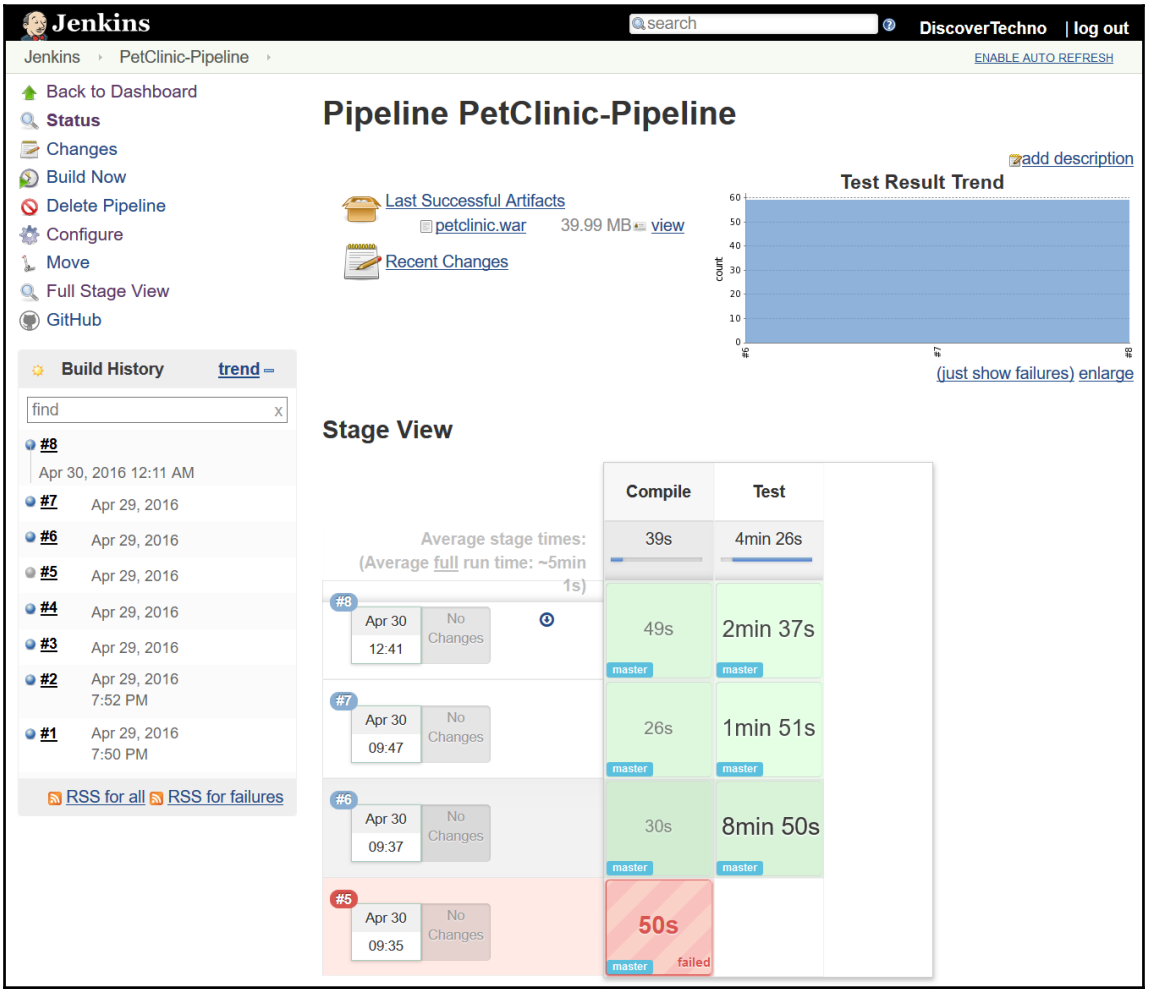


7. Click on the **Stage Logs** link and it will provide log details respective to stage. Click on dropdown to get more details about logs.

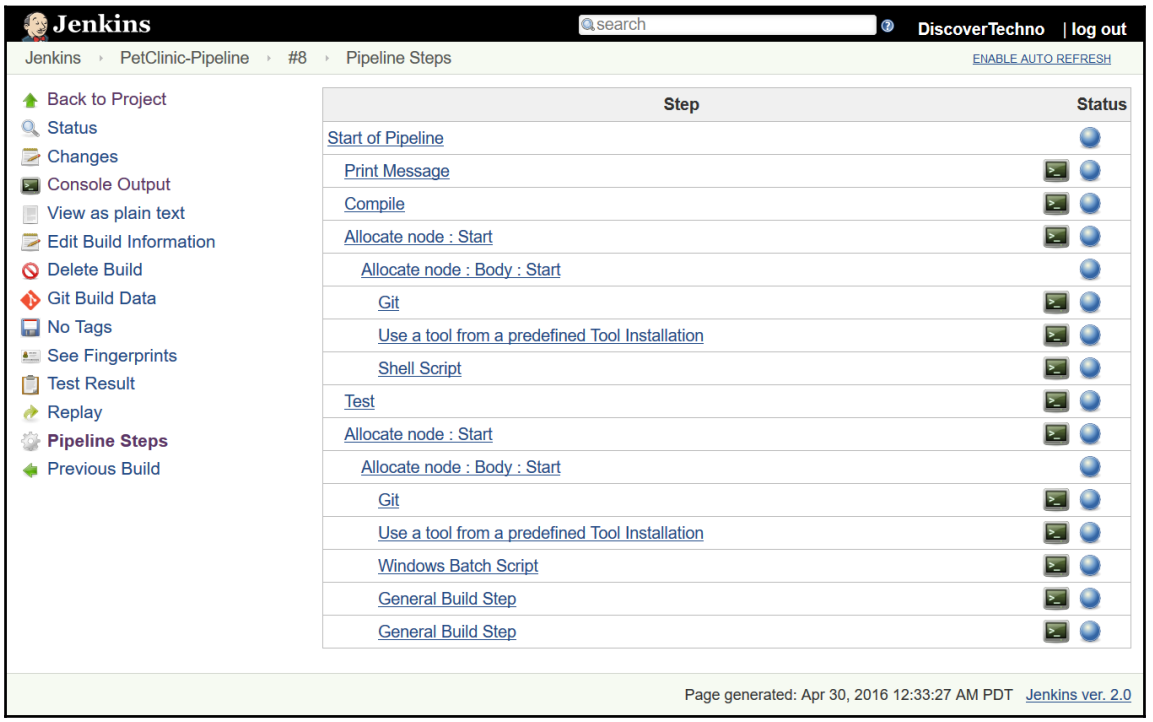
8. Let's consider a scenario where we want to execute different stages on different nodes.

```
echo 'Hello from Pipeline Demo'
stage 'Compile'
node {
  git url: 'https://github.com/mitesh51/spring-petclinic.git'
  def mvnHome = tool 'Maven3.3.1'
  sh "${mvnHome}/bin/mvn -B compile"
}
stage 'Test'
node('WindowsNode') {
  git url: 'https://github.com/mitesh51/spring-petclinic.git'
  def mvnHome = tool 'WindowsMaven'
  bat "${mvnHome}\\bin\\mvn -B verify"
  step([$class: 'ArtifactArchiver', artifacts: '**/target/*.war', fingerprint: true])
  step([$class: 'JUnitResultArchiver', testResults: '**/target/surefire-reports/TEST-*.xml'])
}
```

9. Click on **Build Now** and verify the **Stage View**.



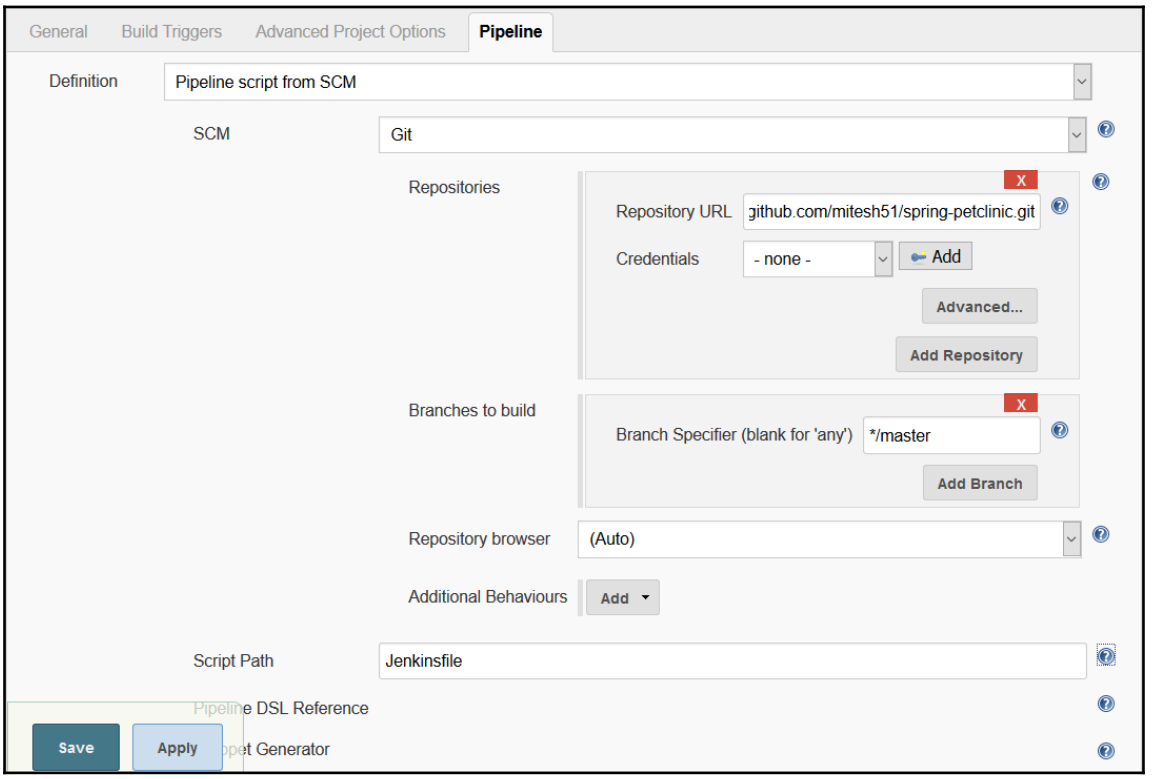
10. Pipeline steps describes drill down details of execution as shown below:



11. Let's verify stage logs for Git Operation. Mouse over the compile stage and click on **logs**. Expand **Git** dropdown as shown in the below image to get more details.

Can you guess what can be the potential issue with Groovy script for creating pipeline?

Yes, again it is a code. It becomes difficult to manage it over the time and hence it is always better to store them in repository. In Pipeline definition, there is an option available to load Pipeline script from SCM. We can select SCM from Git or Subversion and then we need to provide repository details and script file details.



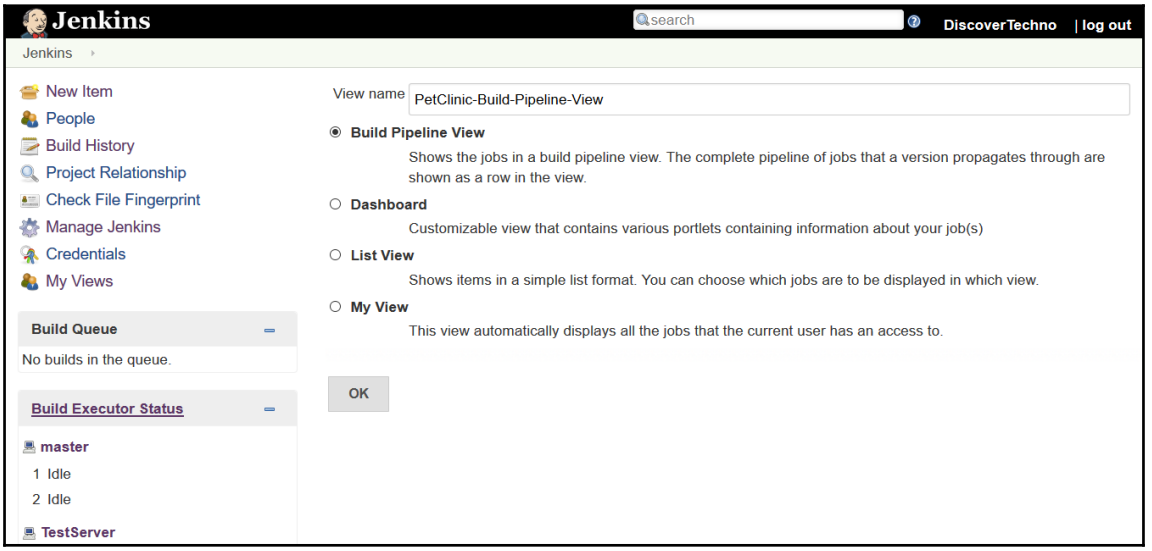
Getting Started with Pipeline at https://jenkins.io/doc/pipeline/

Building Pipeline plugin

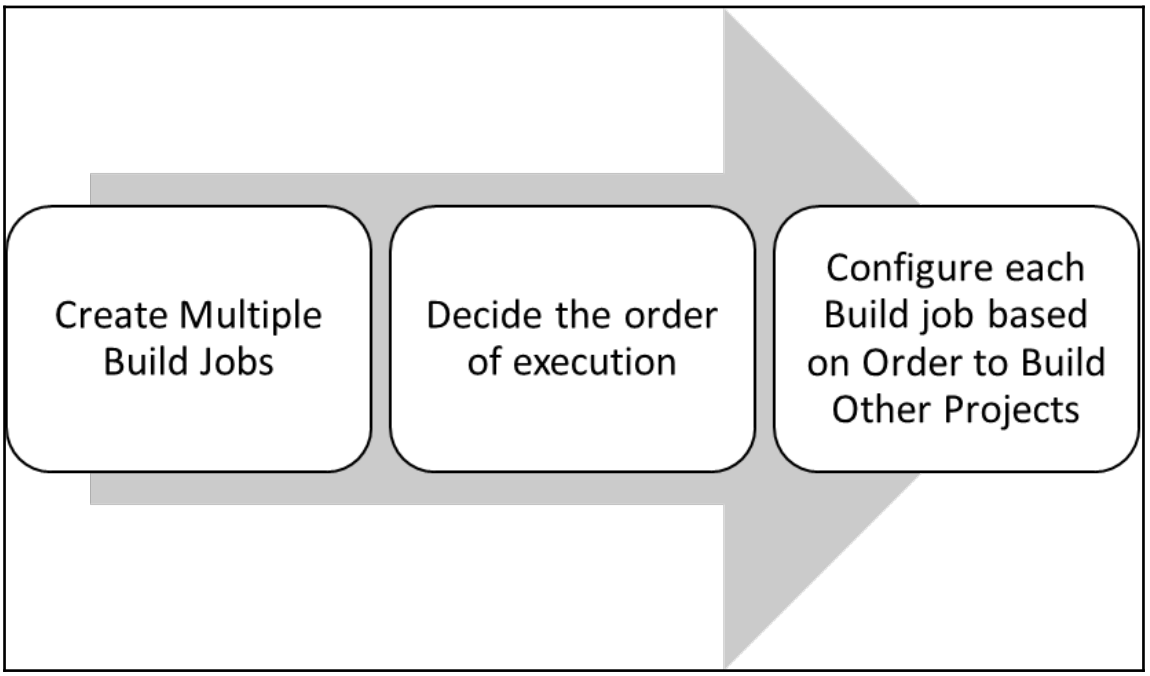
We have seen built-in pipeline concept of Jenkins 2. It is a very flexible and powerful concept but for that we need to write a groovy script. One another way that has easy learning curve is to use build pipeline plugin. It provides simple visualization upstream and downstream build jobs. It also allows manual triggers for a situation where we need approvals for executing specific build. We can create chain of jobs for end to end automation. Here we assume that reader is aware about concept of upstream and downstream build jobs.

To create a Build pipeline:

1. Install Build Pipeline plugin.
2. On Jenkins dashboard, click on plus sign that will open a page to create **Build Pipeline View**. Provide name for the build pipeline and click on **OK**.



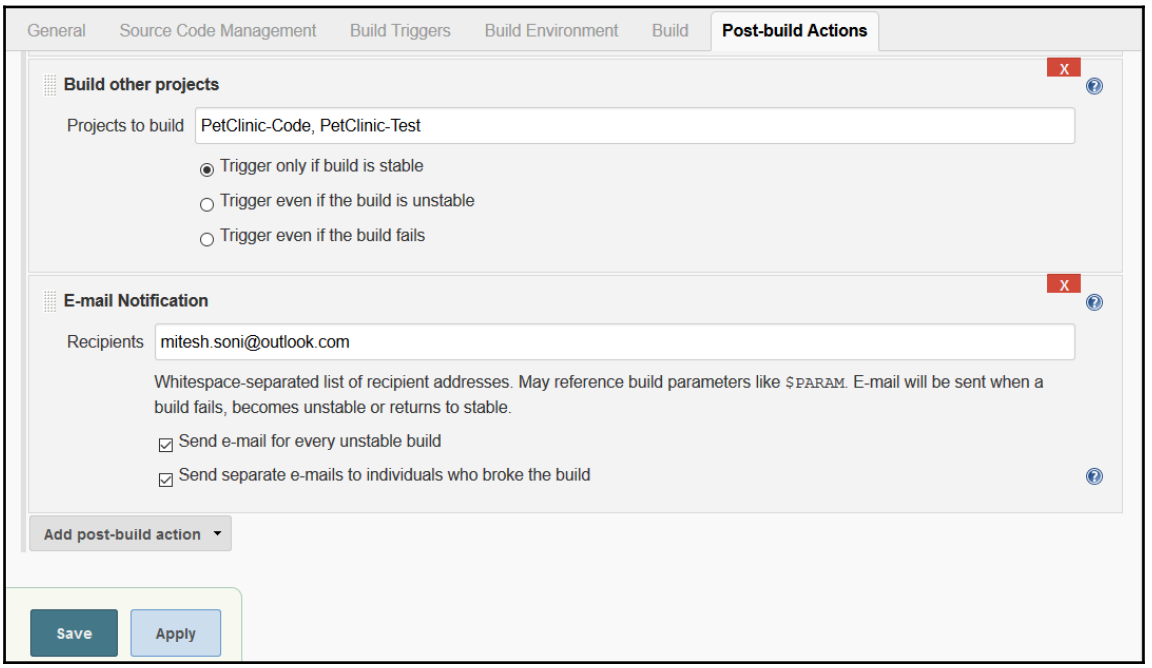
It is important to configure upstream and downstream build jobs.



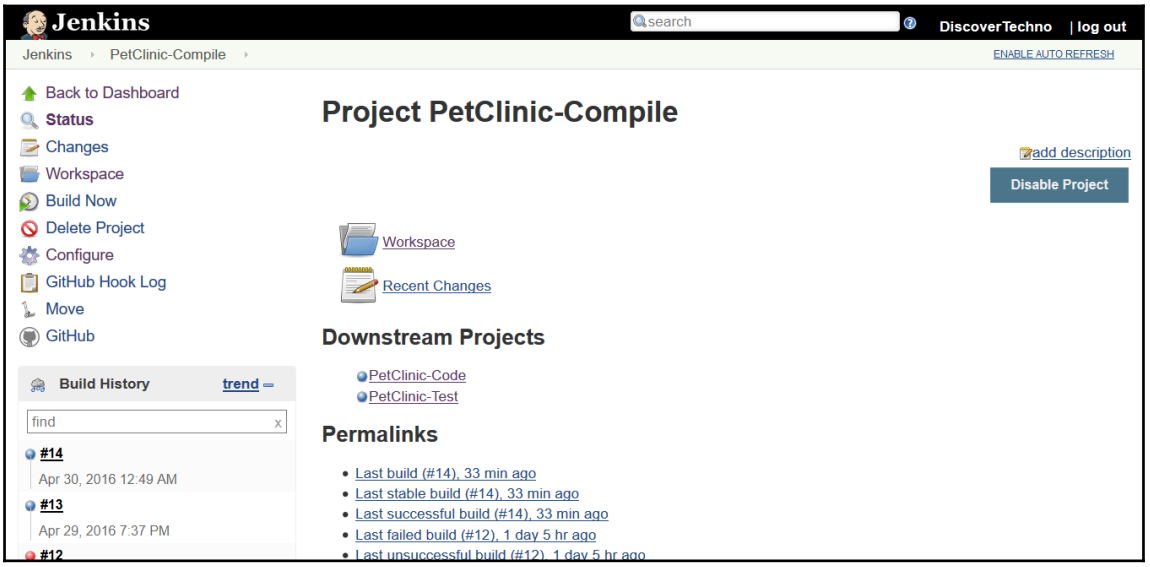
We have created multiple build jobs to compile the source code, to verify source code using Sonar, and to execute JUnit test cases.

We have defined the order also, if compilation is successful and then rest of the two build jobs will be executed. In our case it is PetClinic-Code and PetClinic-Test.

1. Go to configuration page of PetClinic-Compile build job.
2. Go to **Post-build Actions** section.
3. Enter name of the Build jobs in the **Project to build** box. We can provide a comma separated list here.
4. Click on **SAVE** to save the configuration.



5. Verify list of the **Downstream projects** on Build Job's main page.

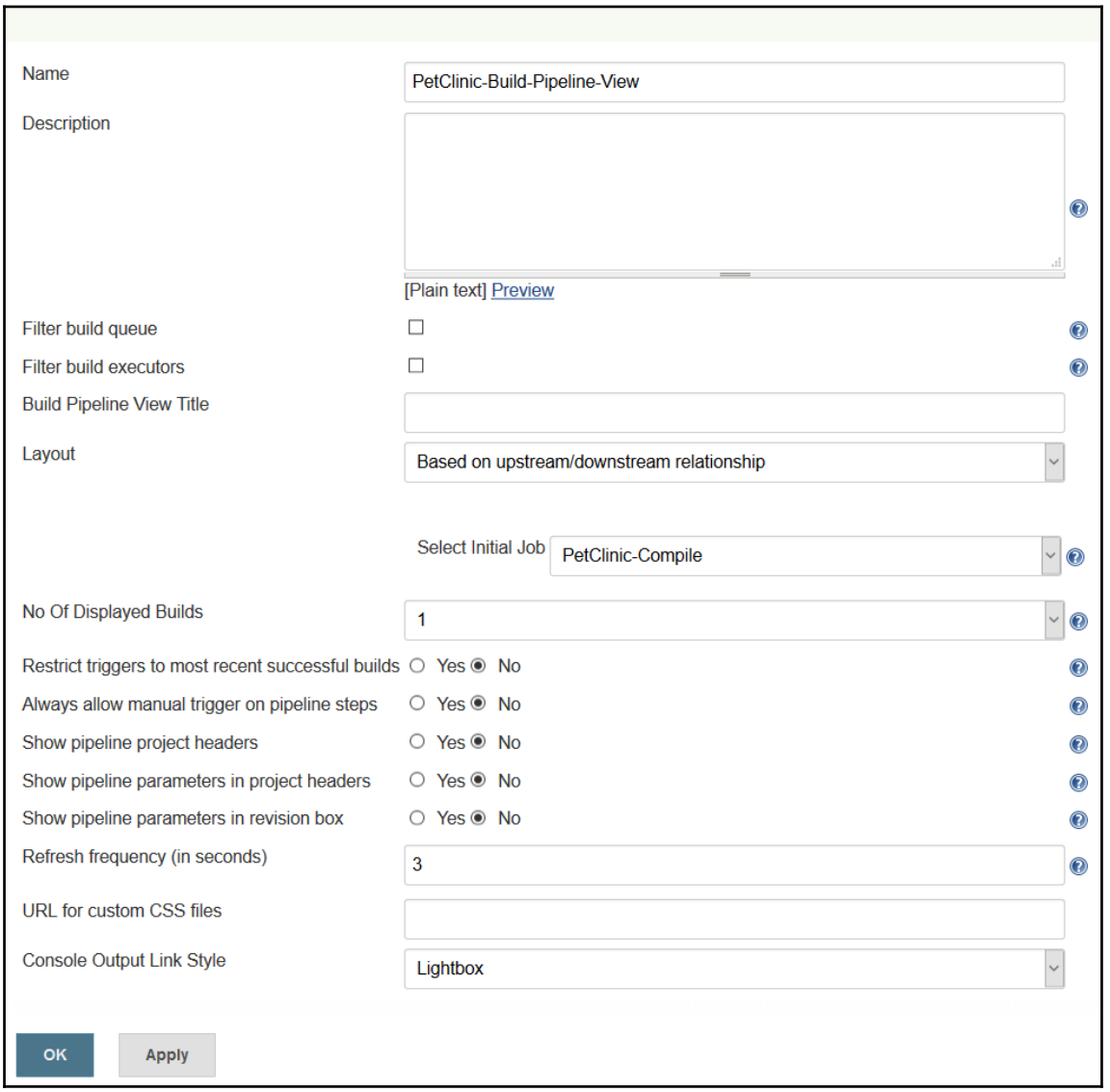


6. Now, the next step is to configure Build Pipeline view that we have created earlier.

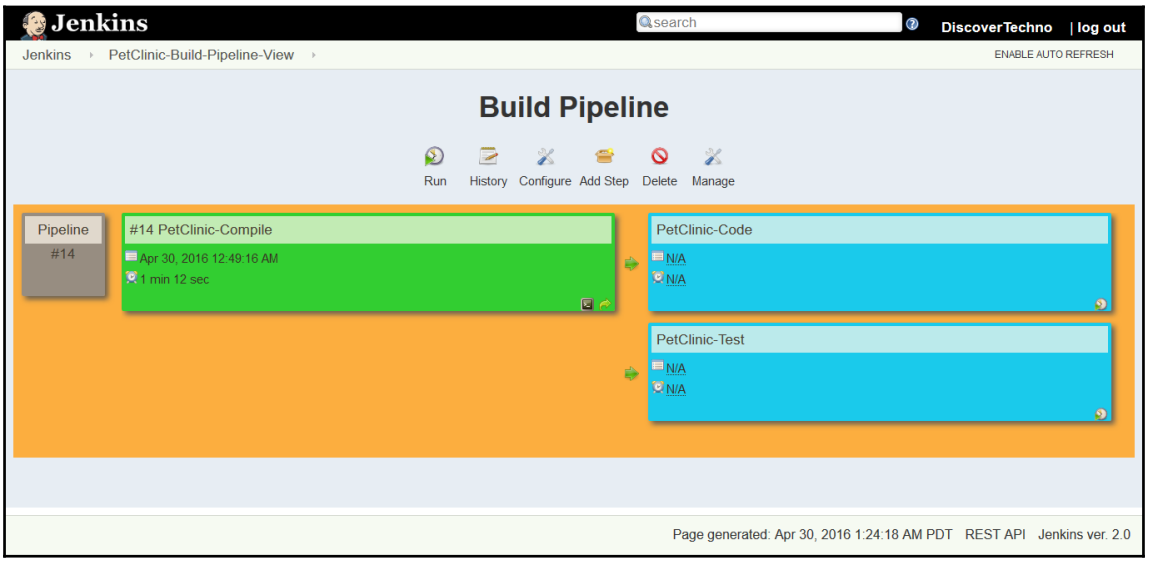
Name	**Name of the Build pipeline**
Description	Description is displayed on the Build Pipeline View Page. It can be used to display details such as Pipeline, resources, objective of the pipeline, flow, and so on.
Filter build queue	Only jobs in this specific view will be shown in the queue.
Filter build executors	To show build executors that could execute the jobs in this view.
Build Pipeline View Title	Build Pipeline View Title to display on the Jenkins Dashboard
Layout	Based on upstream/downstream relationship: This layout mode derives the pipeline structure based on the upstream/downstream trigger relationship between jobs.
Select Initial Job	Set the initial or parent Job in the build pipeline view. Rest of the Build Job will be considered based on upstream/downstream relationship.
No Of Displayed Builds	Number of build pipelines to display in the view.
Restrict triggers to most recent successful builds	To restrict the display of a Trigger button to only the most recent successful build pipelines.

Always allow manual trigger on pipeline steps	To execute again a successful pipeline step using the same parameter values if the build is parameterized.
Show pipeline project headers	To show the pipeline definition header in the pipeline view.
Show pipeline parameters in project headers	To list the parameters used to run the latest successful job in the pipeline's project headers.
Show pipeline parameters in revision box	To list the the parameters used to run the first job in each pipeline's revision box.
Refresh frequency (in seconds)	Provide frequency at which the Build Pipeline Plugin updates the build lightbox in seconds
URL for custom CSS files	Custom CSS file if any
Console Output Link Style	Lightbox, New Window, This Window

7. We have select PetClinic-Compile build job as Initial Job as shown in the below image:



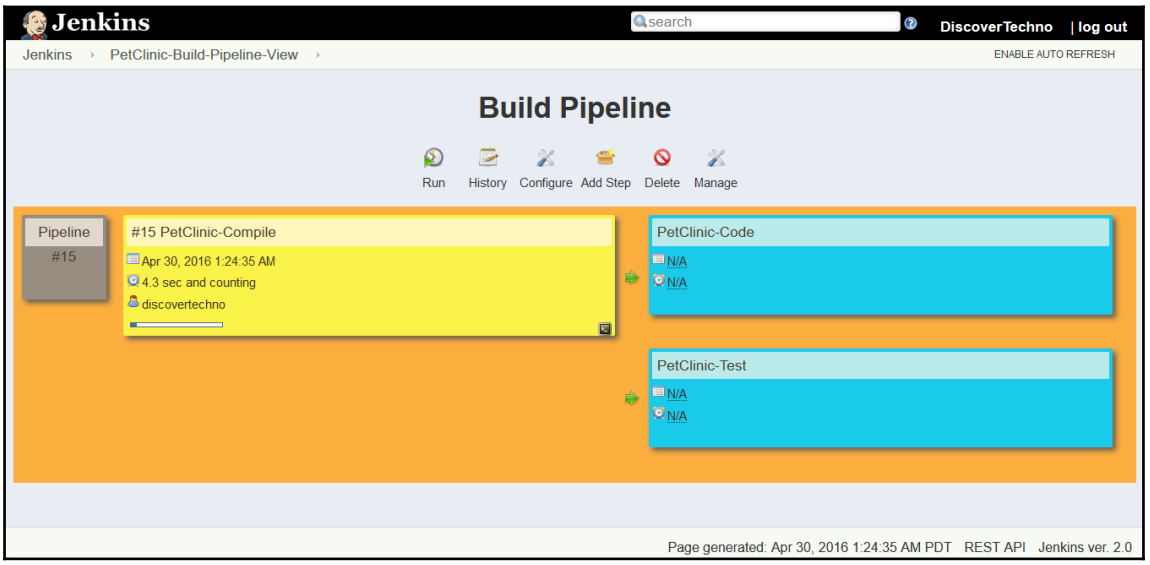
8. On the View page, we can run the build pipeline, view history, configure the pipeline, delete the pipeline, and so on. Click on **Run** to execute Build pipeline for the first time.



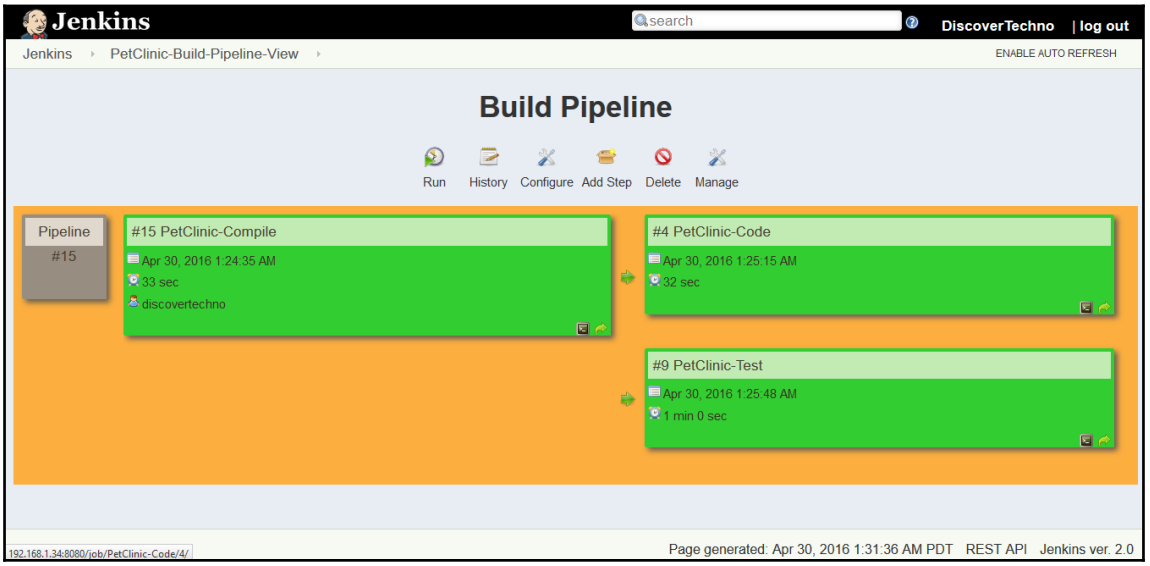
9. Following are color codes by default:

Color	**Description**
Red	Indicates Failed execution of Build Job
Green	Indicates Successful execution of Build Job
Blue	Indicates Build Job that hasn't been executed
Yellow	Indicates Running Build Job

10. Now just observe the execution of build job in this pipeline.

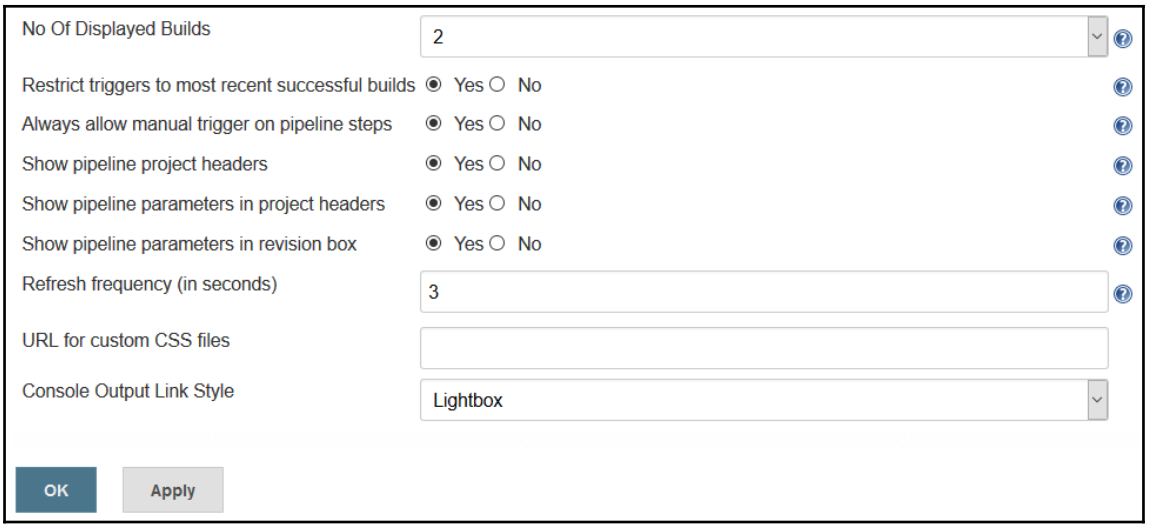


12. We can see all jobs in Green as all the builds have been executed successfully, as shown in the image below:

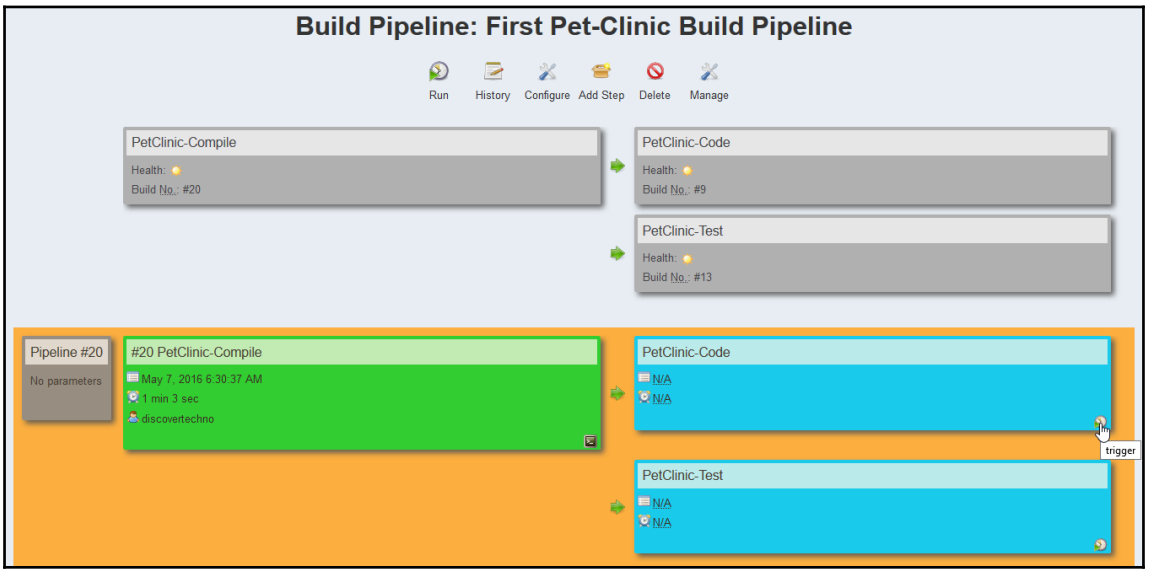


Let's configure Build pipeline using manual trigger:

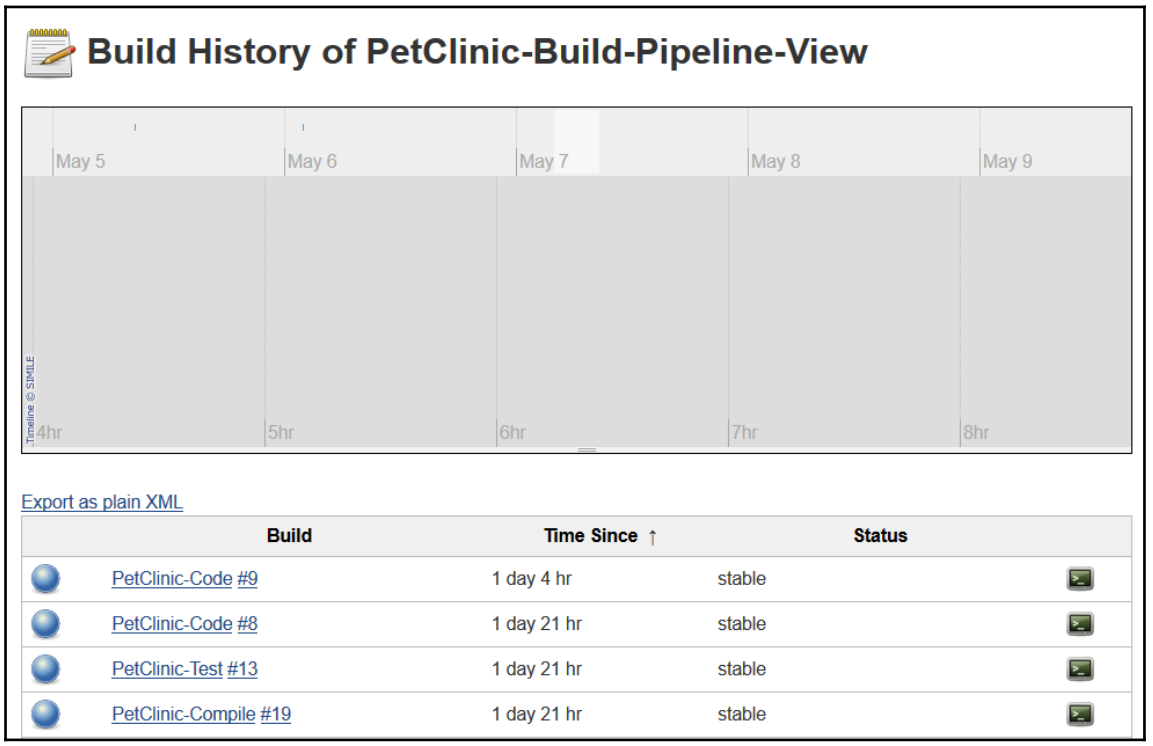
1. Show pipeline project headers, Show pipeline parameters in project headers, Show pipeline parameters in revision box, and so on.



2. Let's save and verify the changes in the Build pipeline view. Verify the manual trigger and Headers with health details of each build job.



3. Verify the History of the Build pipeline as shown in below image.



Download Build Pipeline Plugin at:
https://wiki.jenkins-ci.org/display/JENKINS/Build+Pipeline+Plugin

Deploying a WAR file

For Maven and Tomcat integration, lets create an admin user. We will use admin user credential to deploy an application into Tomcat server.

1. Open apache-tomcat-7.0.68\conf\tomcat-users.xml and add following statements into it.

- Here we define roles such as manager-gui, manager-script. For this deployment, we will use manager-script role.

- Create a user with name admin and assign password and roles as below:

```
<role rolename="manager-gui"/>
<role rolename="manager-script"/>
<user username="admin" password="cloud@123" roles="manager-script" />
```

1. Now, we need to add Tomcat's admin user that we created in the Maven setting file.

```
<servers>
<server>
  <id>tomcat-development-server</id>
  <username>admin</username>
  <password>password</password>
</server>
</servers>
```

2. Now let's edit pom.xml file. Find Tomcat Plugin block in Pom.xml and add following details. Make sure that Server Name is same that we provided in settings.xml of Maven as Id.

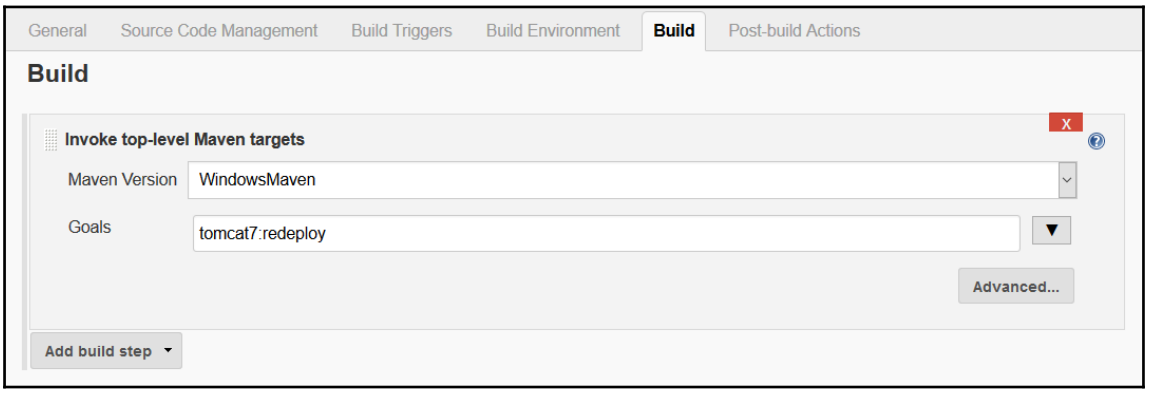
```
<plugin>
            <groupId>org.apache.tomcat.maven</groupId>
            <artifactId>tomcat7-maven-plugin</artifactId>
            <version>2.2</version>
            <configuration>
               <server>tomcat-development-server</server>
              <url>http://192.168.1.35:9999/manager/text</url>
              <warFile>target\petclinic.war</warFile>
              <path>/petclinic</path>
             </configuration>
 </plugin>
```

4. We can verify the execution from command line using mvn tomcat7:deploy command. Maven deploy the WAR file to Tomcat 7 using Manager App http://localhost:8080/manager/text, on path /petclinic.

5. In case of any failures because of already existing WAR file in Tomcat webapps folder, use tomcat7:redeploy.

Let's create a build job in Jenkins and add a build step to invoke top-level Maven targets:

1. Use tomcat7:redeploy as goals. Save the configuration.



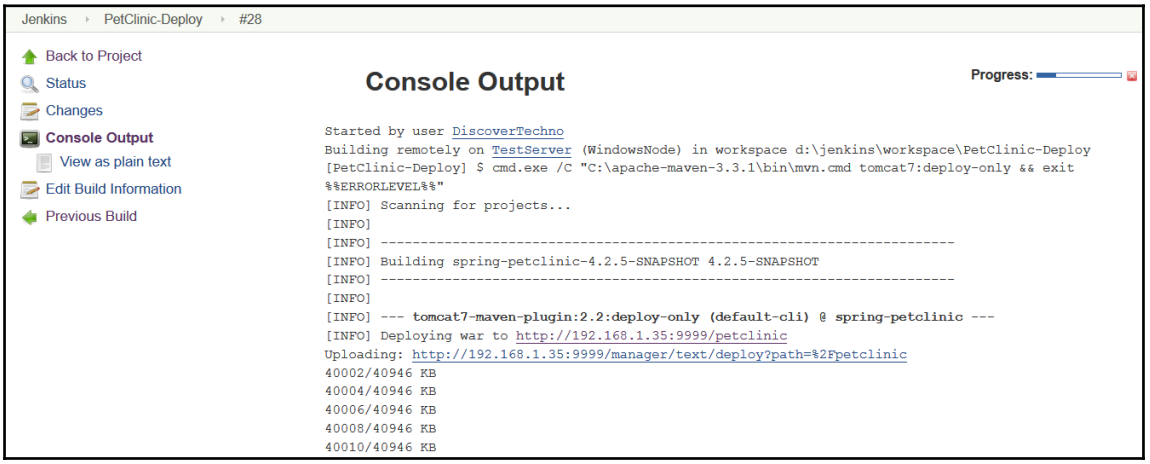
2. Execute the build by click on **Build Now**. Verify the deployment process in the **Console Output**.

```
[INFO] --- maven-war-plugin:2.3:war (default-war) @ spring-petclinic ---
[INFO] Packaging webapp
[INFO] Assembling webapp [spring-petclinic] in [d:\jenkins\workspace\PetClinic-Deploy\target\spring-
petclinic-4.2.5-SNAPSHOT]
[INFO] Processing war project
[INFO] Copying webapp resources [d:\jenkins\workspace\PetClinic-Deploy\src\main\webapp]
[INFO] Webapp assembled in [969 msecs]
[INFO] Building war: d:\jenkins\workspace\PetClinic-Deploy\target\spring-petclinic-4.2.5-SNAPSHOT.war
[INFO]
[INFO] <<< tomcat7-maven-plugin:2.2:redeploy (default-cli) < package @ spring-petclinic <<<
[INFO]
[INFO] --- tomcat7-maven-plugin:2.2:redeploy (default-cli) @ spring-petclinic ---
[INFO] Deploying war to http://192.168.1.35:9999/petclinic
Uploading: http://192.168.1.35:9999/manager/text/deploy?path=%2Fpetclinic&update=true
40002/40946 KB
40004/40946 KB
40006/40946 KB
40008/40946 KB
40010/40946 KB
40012/40946 KB
```

3. Once WAR file is uploaded successfully, Build Job will be completed successfully.

```
40940/40946 KB
40942/40946 KB
40944/40946 KB
40946/40946 KB
Uploaded: http://192.168.1.35:9999/manager/text/deploy?path=%2Fpetclinic&update=true (40946 KB at
9024.8 KB/sec)

[INFO] tomcatManager status code:200, ReasonPhrase:OK
[INFO] OK - Deployed application at context path /petclinic
[INFO] ------------------------------------------------------------------------
[INFO] BUILD SUCCESS
[INFO] ------------------------------------------------------------------------
[INFO] Total time: 58.469 s
[INFO] Finished at: 2016-05-07T23:41:13+05:30
[INFO] Final Memory: 38M/263M
[INFO] ------------------------------------------------------------------------
Finished: SUCCESS
```

When we use tomcat7:deploy or tomcat7:redeploy then it includes package lifecycle in the execution. If we want to only deploy the WAR file, then we can use tomcat7:deploy-only as shown in below console output.



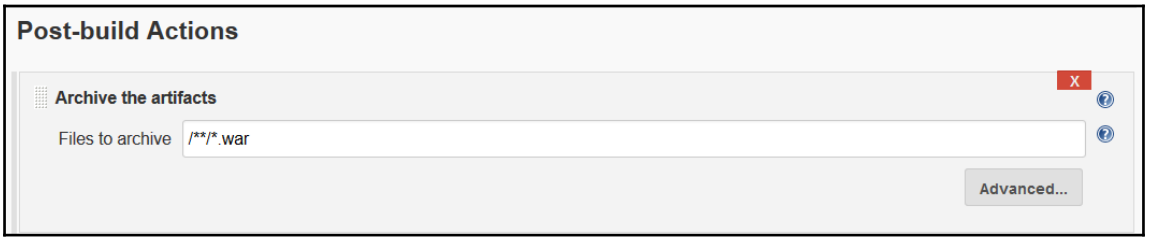
Let's try to integrate deploy operation in the build pipeline.

We need to do following things:

1. Compile source files.
2. Execute JUnit test cases.

3. Archive artifact / WAR file.
4. Copy artifact to deploy build job.

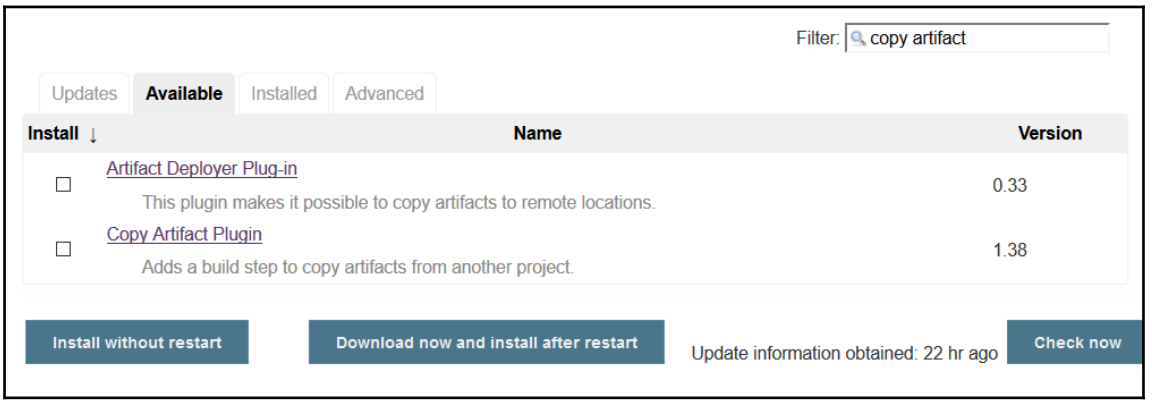
    - It is used to archive the build artifact such as jar files, war files, and zip files so it can be downloaded later. Add post build action in to PetClinic-Test file to archive artifact.



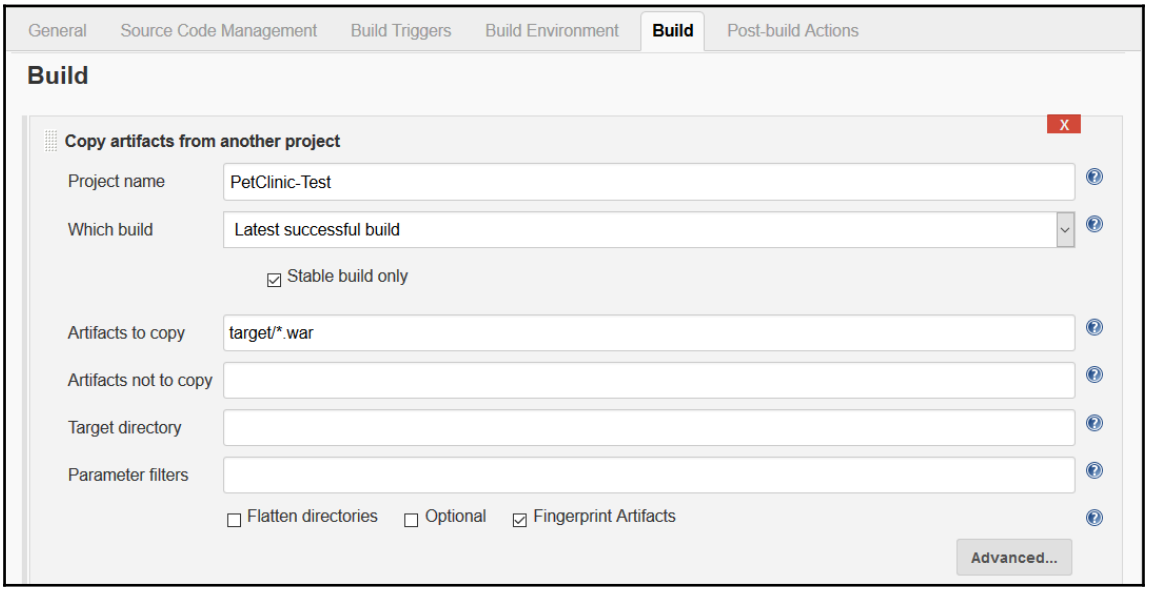
5. Execute the build job as shown below and verify whether it is successfully archived or not.

```
[INFO] --- maven-war-plugin:2.3:war (default-war) @ spring-petclinic ---
[INFO] Packaging webapp
[INFO] Assembling webapp [spring-petclinic] in [d:\jenkins\workspace\PetClinic-Test\target\spring-
petclinic-4.2.5-SNAPSHOT]
[INFO] Processing war project
[INFO] Copying webapp resources [d:\jenkins\workspace\PetClinic-Test\src\main\webapp]
[INFO] Webapp assembled in [2371 msecs]
[INFO] Building war: d:\jenkins\workspace\PetClinic-Test\target\spring-petclinic-4.2.5-SNAPSHOT.war
[INFO] ------------------------------------------------------------------------
[INFO] BUILD SUCCESS
[INFO] ------------------------------------------------------------------------
[INFO] Total time: 35.535 s
[INFO] Finished at: 2016-05-08T00:53:30+05:30
[INFO] Final Memory: 29M/271M
[INFO] ------------------------------------------------------------------------
Archiving artifacts
Recording test results
Warning: you have no plugins providing access control for builds, so falling back to legacy behavior
of permitting any downstream builds to be triggered
Triggering a new build of PetClinic-Deploy
Finished: SUCCESS
```

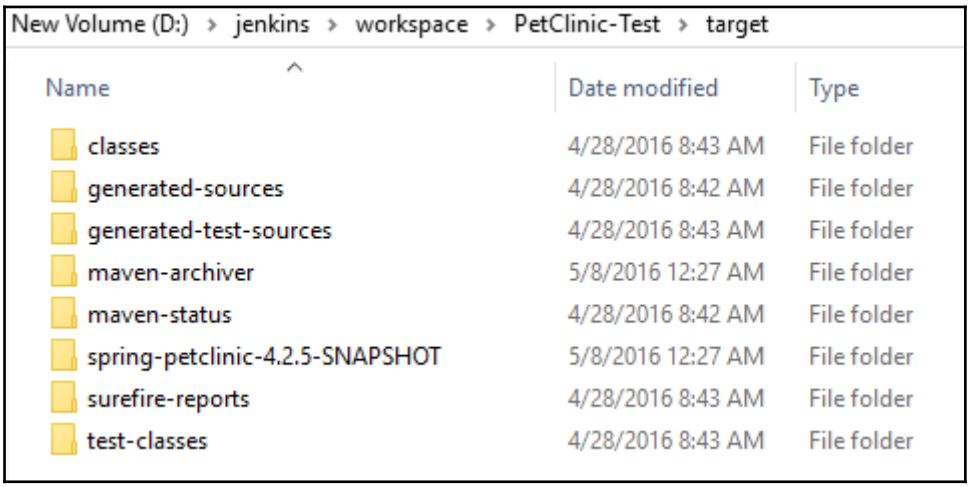
6. We need to add a build step to copy artifacts from PetClinic-Test. Install Copy Artifact Plugin.



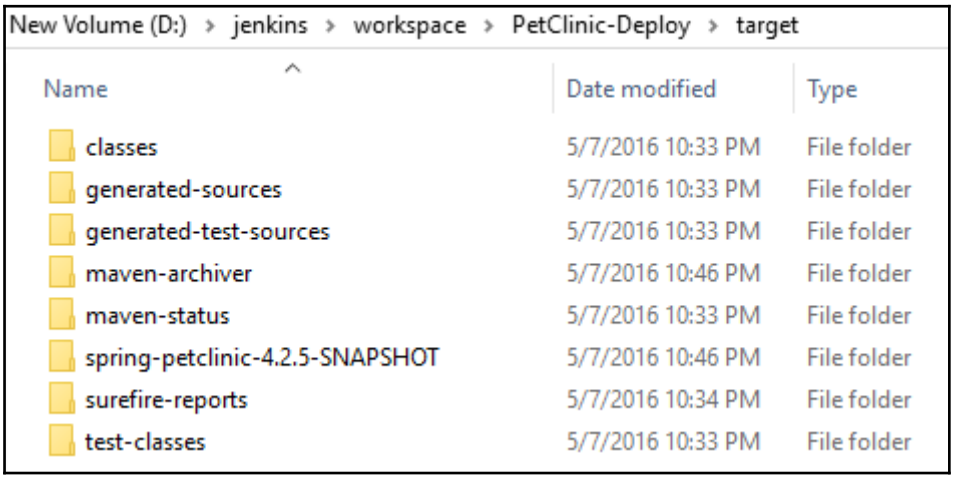
7. Configure Copy Artifact plugin in the PetClinic-Deploy Build job as shown in the below image.



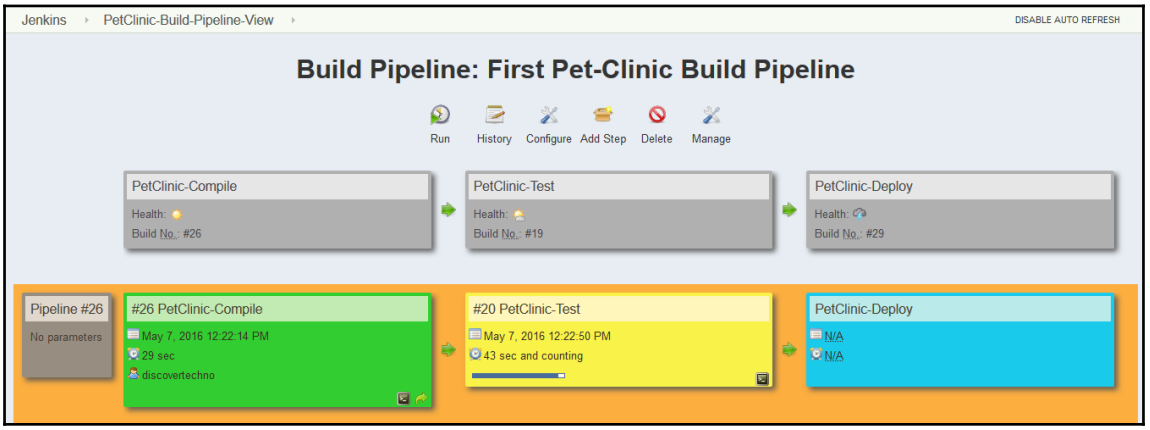
8. Verify the workspace directory. Go to PetClinic-Test's target directory. If war file is there from past build, then remove it.



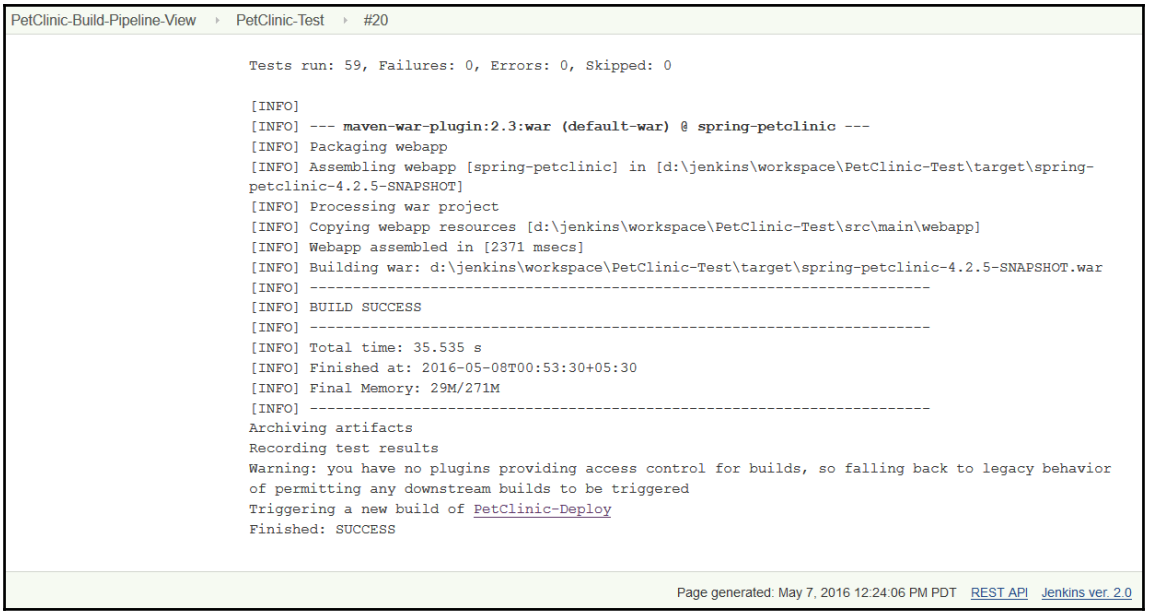
9. Verify the target directory of PetClinic-Deploy folder. No WAR file is available.



10. Add PetClinic-Deploy as Downstream project in the PetClinic-Test. Run the Build Pipeline.



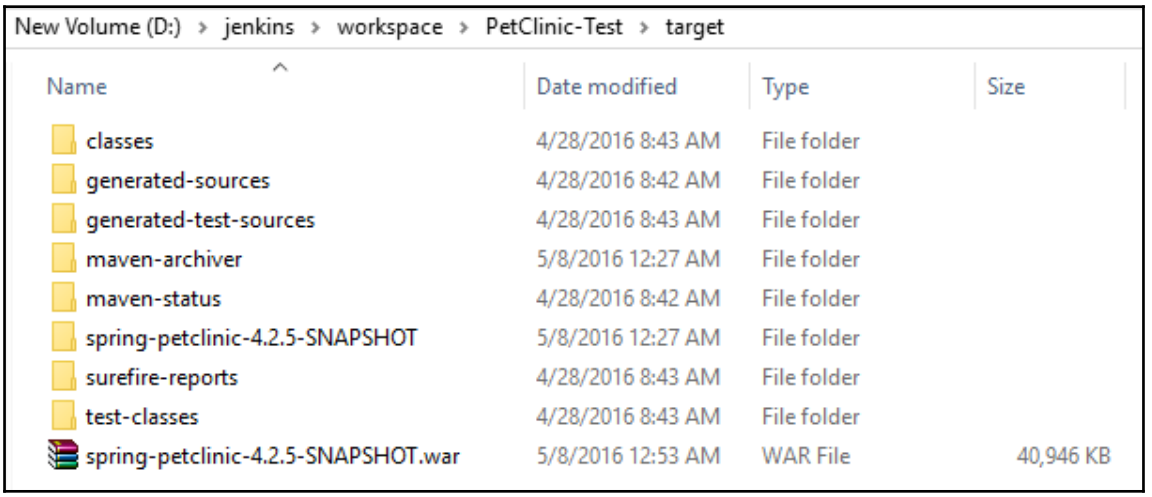
11. Verify the execution of Build Pipeline. Click on the Light box of any build job available in the Build Pipeline. Verify the PetClinic-Test console output.



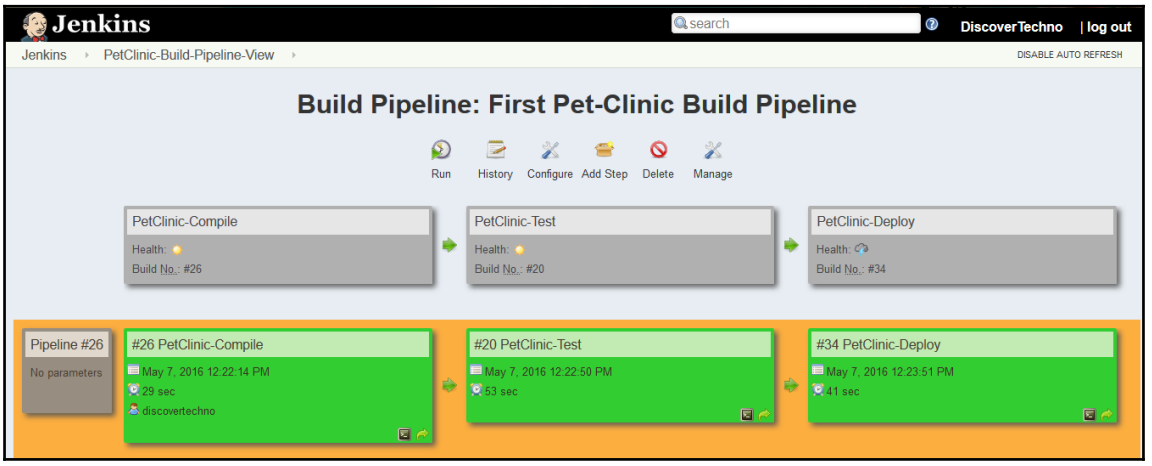
Once PetClinic-Test build job is execution is completed then:

- Verify the target folder in workspace.
- We will see WAR file in the target directory as shown in the below

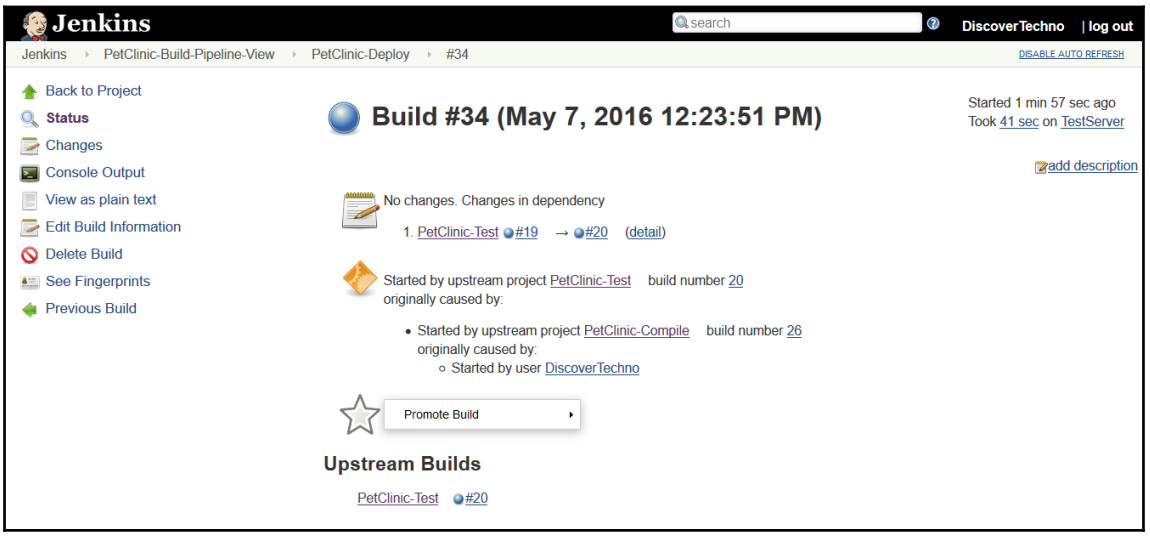
image.



1. Verify the execution of PetClinic-Deploy build job.

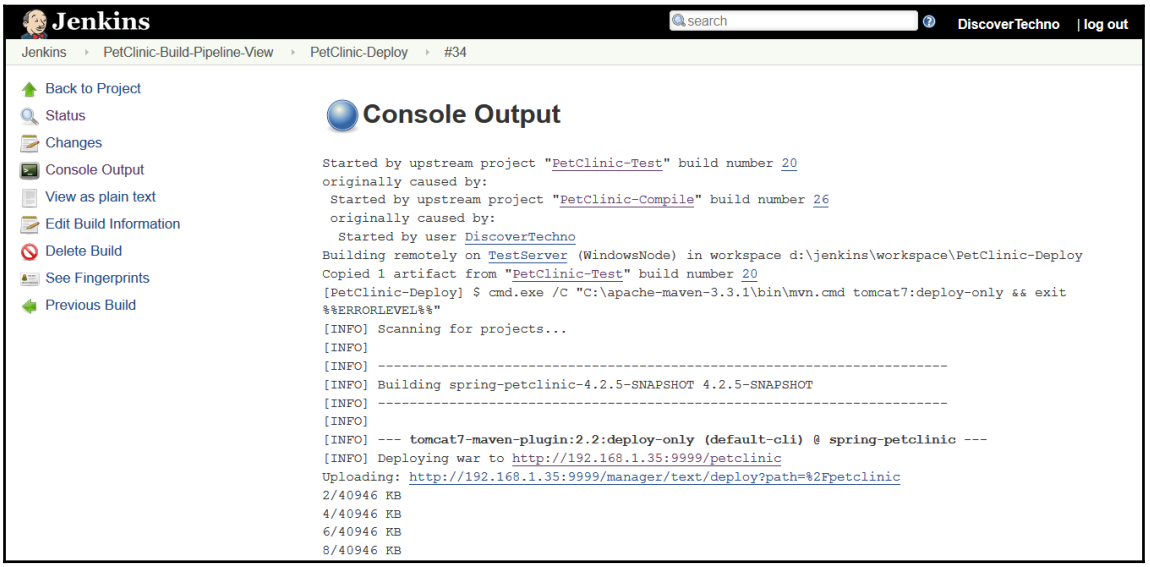


2. Verify the build job's status in the Jenkins dashboard.

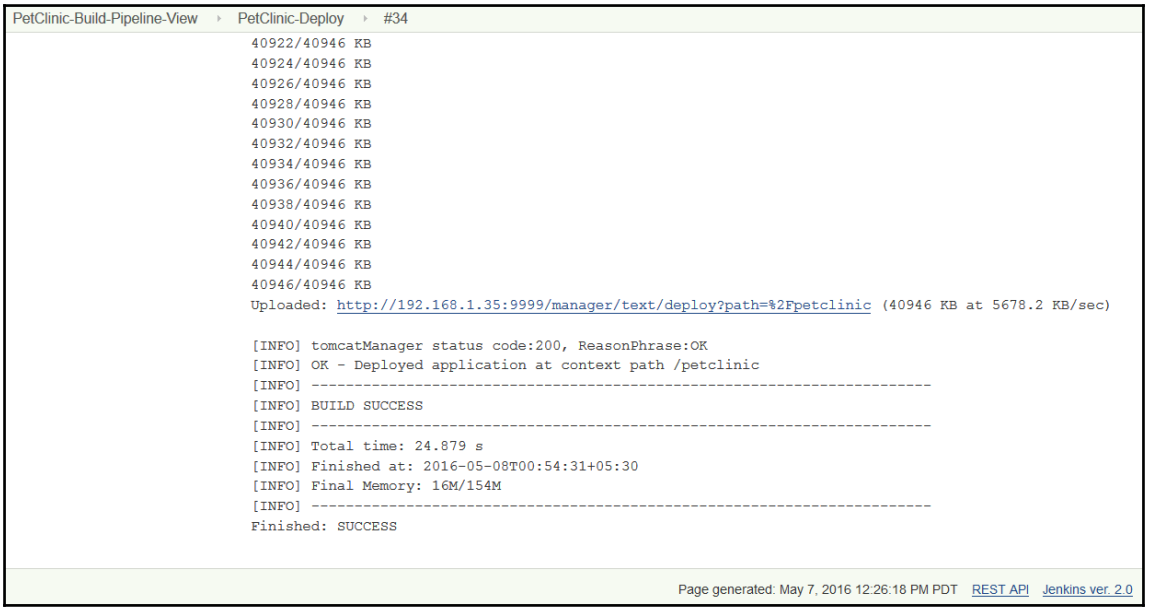


3. Click on the light box of Build pipeline view, it will direct us to console output of specific build job. Click on PetCLinic-Deploy lighbox.

4. Verify the Console output.



5. Verify the successfully uploaded file as per the configuration.



For the self exercise, try to use build flow plugin.

Self-Test Questions

1. Which feature is one of the Highlights of Jenkins 2 release?

a. Built-in support for continuous integration

b. Built-in support for JUnit

c. Built-in support for delivery pipelines

d. Built-in support for Apache Maven

2. Which language is used to create delivery pipelines ?

a. Java

b. C++

c. C#

d. Domain-specific language

3. In Build Pipeline plugin, what is the significance of Blue color?

a. Indicates Failed execution of Build Job

b. Indicates Successful execution of Build Job

c. Indicates Build Job that hasn't been executed

d. Indicates Running Build Job

Summary

In this chapter, we have covered latest feature of Jenkins 2 that is one of the Highlights of Jenkins 2 release – Built-in support for delivery pipelines. We have described in details how to use it. It has covered simple groovy script to build a job, to generate a build step, to archive build job artifacts, to run build step on a specific node, to mark definite sections of a build as being controlled by limited concurrency and so on. We have provided a scenario where we want to execute different stages on different nodes. The other similar type of plugin is installed and configured with example – Build Pipeline plugin.

In the next chapter, we will discuss about one of the important pillar of DevOps culture and that is Configuration Management using Chef. First we will see how to install Chef workstation and configure it with Hosted Chef. We will consider installing tomcat using community cookbooks of tomcat installation.

4

Installing and Configuring Chef

"Give me six hours to chop down a tree and I will spend the first four sharpening the axe."
– Abraham Lincoln

We are going to see how Chef is useful in end to end automation of Application delivery lifecycle. Chef in our context plays a vital role considering the usage of it. We are going to use it for setup of runtime environment and standardized the process of configuration management rather than implementing customized way to install tools using scripts. Centralized configuration management makes it easy to control and configure resources without complexities.

This chapter describes in detail about configuration management tool Chef, installation of its components and alternatives; configuration of components and convergence of node based on the cookbooks for preparing runtime environment for JEE application. However, writing cookbooks, and detailed description of Chef component is out of scope as it will take too much space.

You will learn how to install and configure Chef-configuration management tool and convergence of node based on cookbooks/role.

In this chapter, we will cover the following topics:

- Getting started with Chef
- Overview of Hosted Chef
- Installing and Configuring Chef Workstation
- Converging Chef node using Chef Workstation

Getting started with Chef

Chef is one of the most popular configuration tools in the open source world. We have discussed briefly about Chef in Chapter 1, *Getting Started-DevOps Concepts, Tools, and Technology*.

Let's try to get our hands on it for provisioning instances and configuration management. However, before that we will understand basics about it.

There are three major components in Chef:

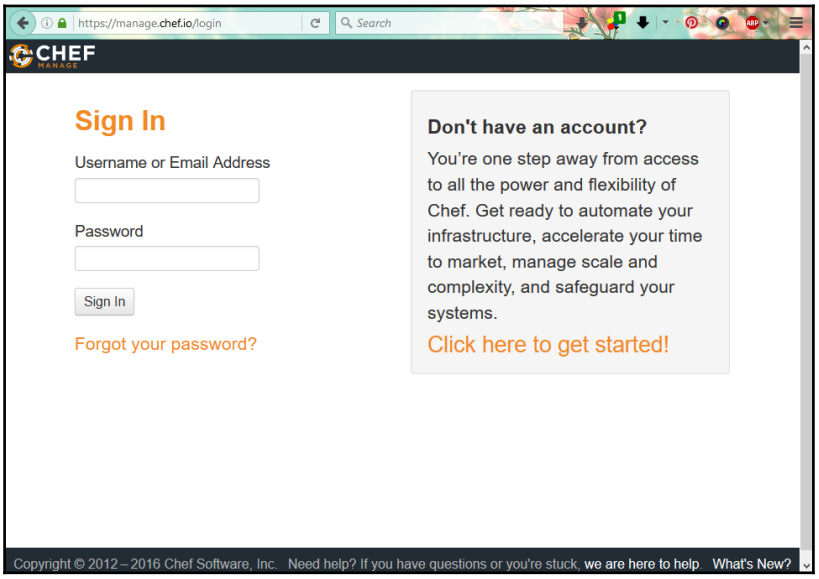
- **Open Source Chef Server or Hosted Chef**: Chef Server or Hosted Chef is the pivotal component that stores cookbooks and other important details of registered nodes. It is used to configure and manage Nodes with the use of Chef workstation.
- **Chef Workstation**: Chef Workstation works as local repository where Knife is installed. Knife is used to upload cookbooks to Chef server or execute plugins commands.
- **Node**: Node is a physical or virtual machine in any environment where we need to configure runtime environments or perform operations with the use of Chef configuration management tool. Node communicates with Chef server (Open source or hosted) and get configuration details related to itself and then start executing steps based on it. Chef Server can be installed on physical machine or on virtual machine with open source installable file based on the operating systems. Another easiest way to use is Hosted Chef where we need not to install and configure Chef server. We can use SaaS offering from Chef. It allows up to five nodes. The biggest benefit is we need not manage Chef server or upgrade it. Hence, we save ourselves from management and maintenance overhead.

To get a first look of the Chef website, visit https://chef.io. You will see a Chef home page as shown below:

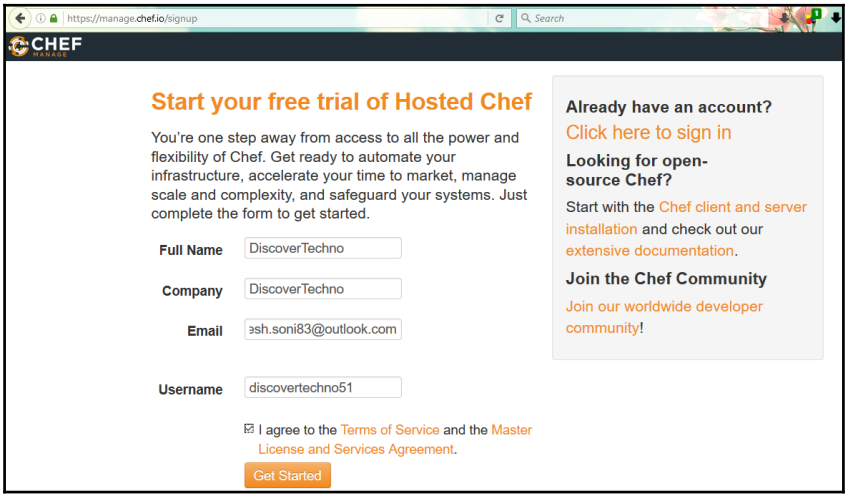
There are lot of details available of Chef and Cloud related integration and knife plugins too. We will create a Hosted Chef account in the section and configure it with local workstation. To go ahead, click on the **MANAGEMENT CONSOLE** link available in the right top corner of the Chef website.

Overview of Hosted Chef

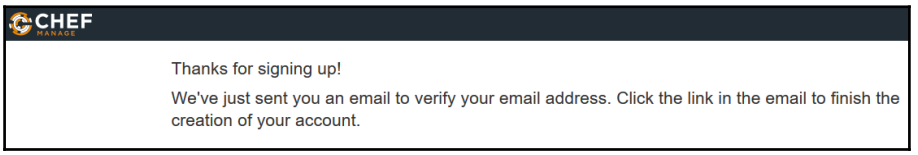
1. Click on **MANAGEMENT CONSOLE** or visit the URL https://manage.chef.io/login. We are going to start from scratch so click on **Click here to get started!**



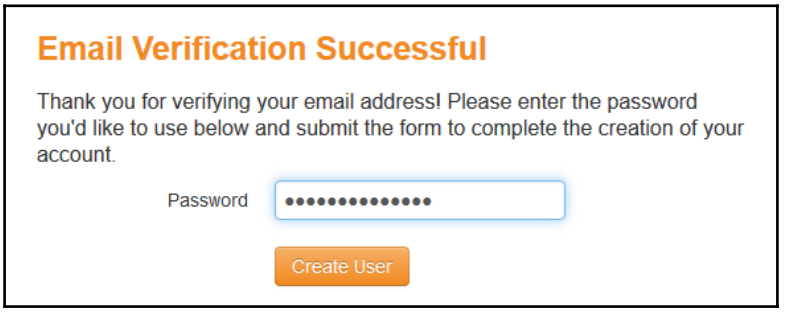
2. Enter **Full Name**, **Email**, and **Username** in the text boxes; check mark on **I agree to the Terms of Service and the Master License and Services Agreement**. Click on **Get Started** button.



3. We will get a message, **Thanks for signing up!**

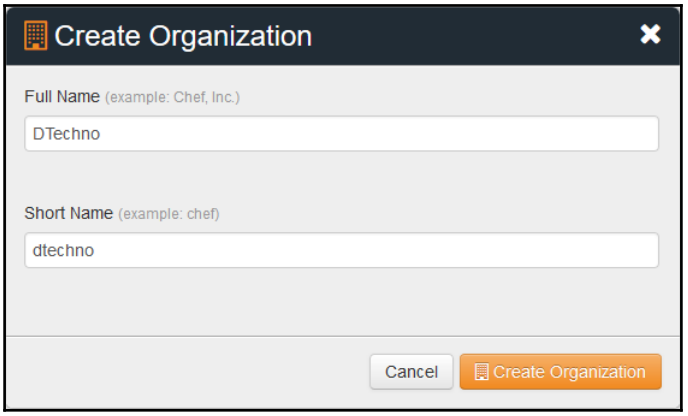


4. Open you Mail inbox and click on the verification link to complete the creation of Hosted Chef account. We will get **Email Verification Successful** message. Click on **Create User** button.

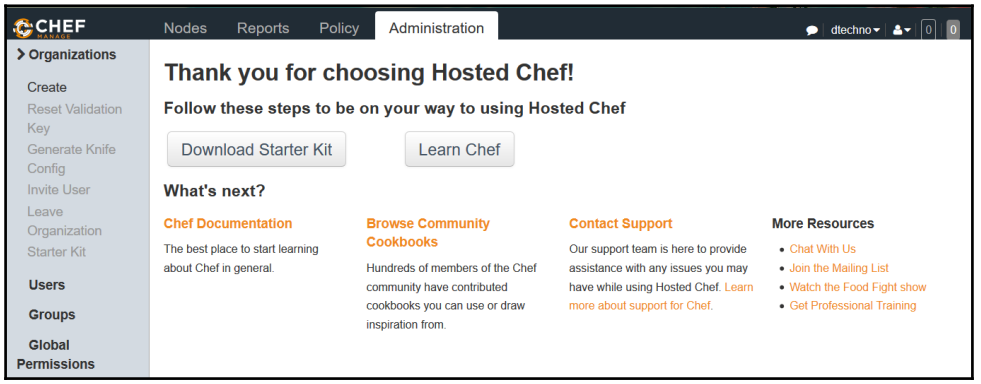


5. Next task is to create an organization. Click on **Create New Organization**.

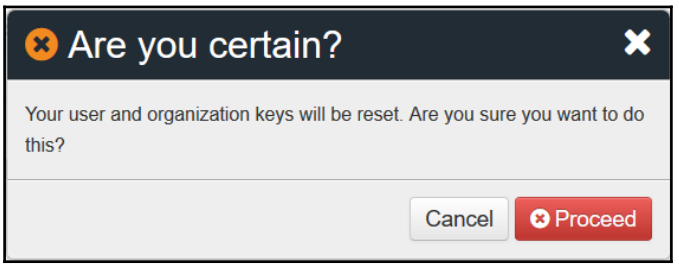
6. Provide **Full Name** and **Short Name** for the organization and click on the **Create Organization** button.



7. Bingo! We have created our hosted Chef account and now we can start using it. Next step is to download a starter kit.

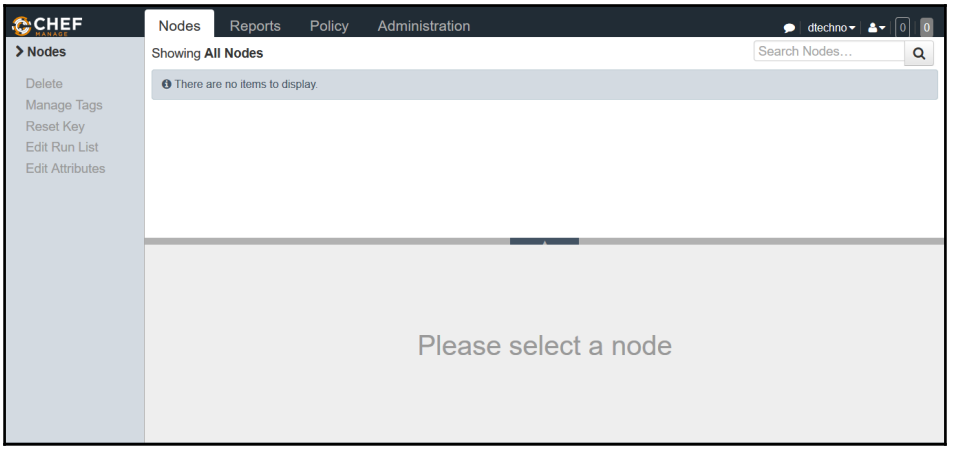


8. When we **Download Starter Kit**, User and Organization Keys will be reset. Make sure to keep it at safe place. Click on **Proceed**.

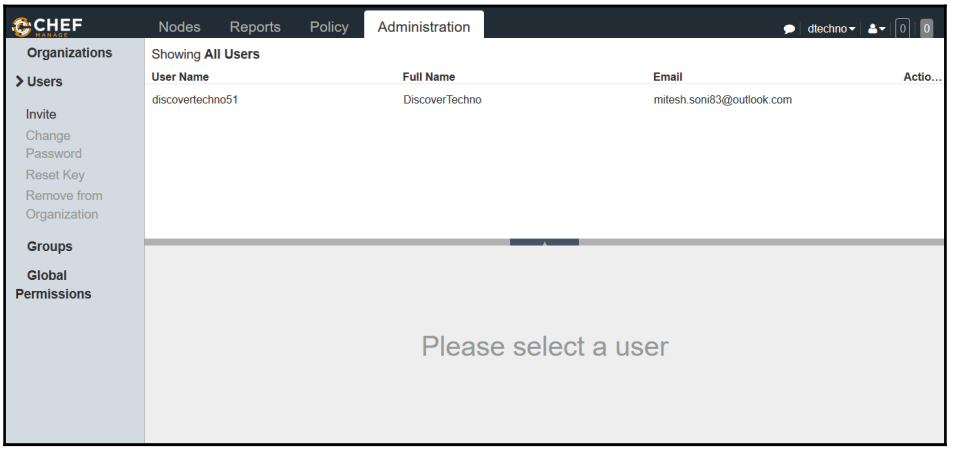


Let's have a quick walk through of the Hosted Chef portal or dashboard:

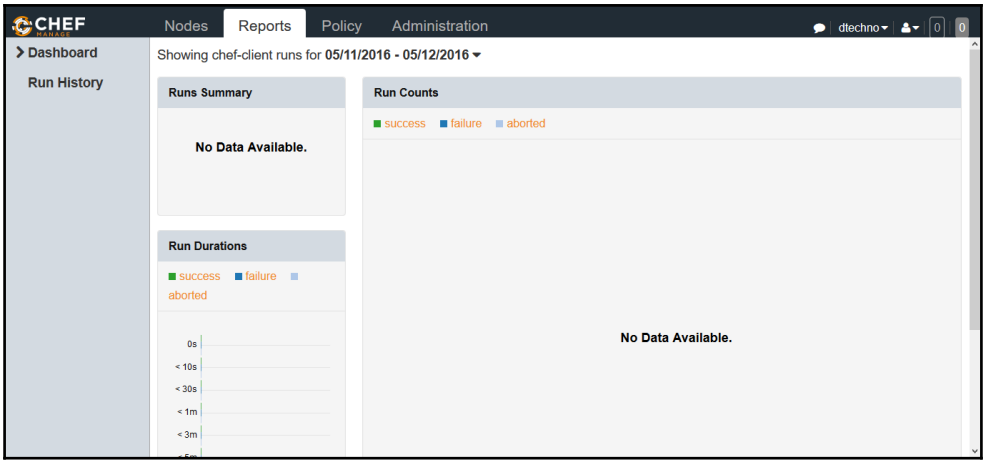
1. Click on the **Nodes** and it will show an empty list as no node is configured using Chef Server. Note this as we are going to see the same screen when we will configure a node.



2. Click on the **Administration** tab and verify the user created at the time of registration.

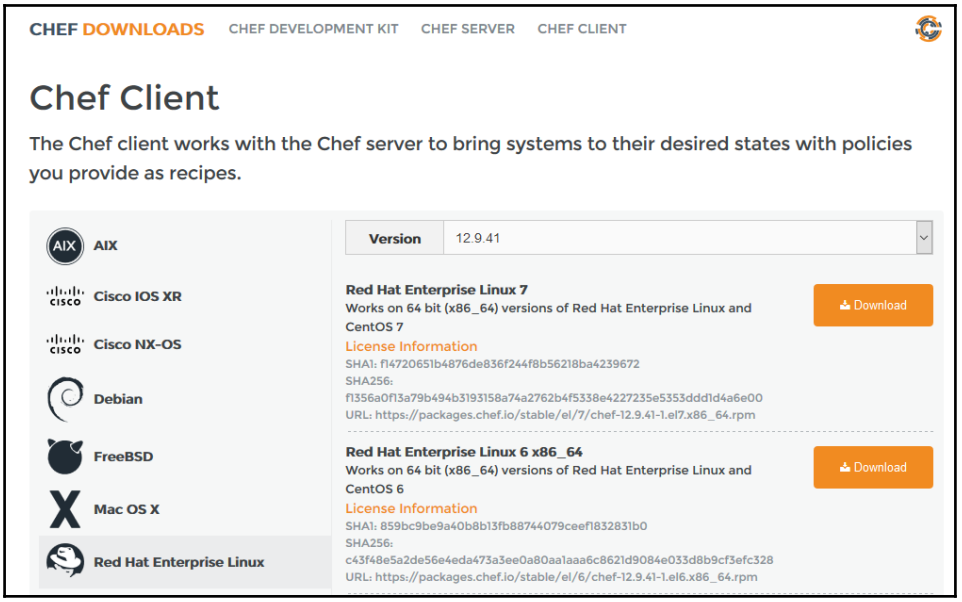


3. **Reports** tab is having no data as convergence process hasn't taken place and no success or failure data is available.

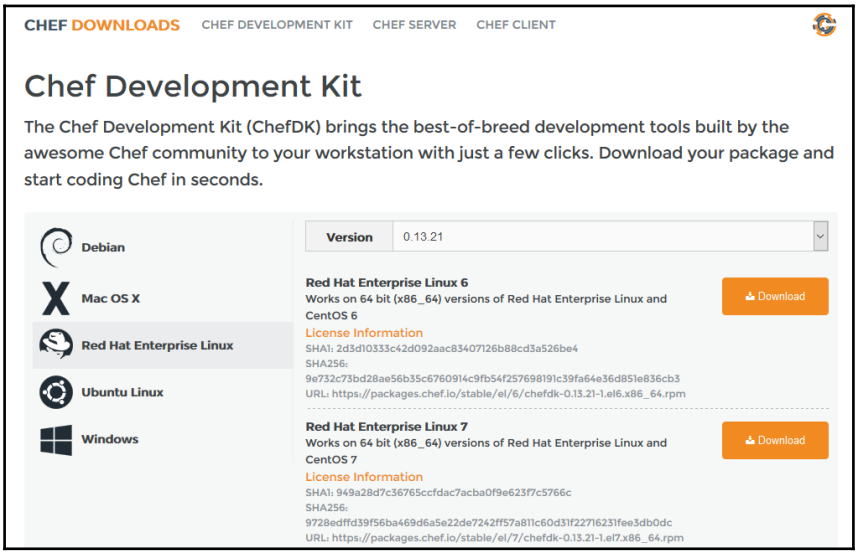


Once we have Hosted Chef account available; next step is to configure Chef workstation.

1. First, download Chef-client from https://downloads.chef.io/chef-client/redhat/ as we are going to use CentOS virtual machine to act as Workstation.
2. Select Operating System and select the Chef client version. Download the Chef client installation files as per the platform available.



3. Chef development kit installation is useful for installing development tools and it can be useful for knife plugins installations for AWS and Azure. Download Chef Development Kit from https://downloads.chef.io/chef-dk/.



In the next section we will see how to configure Chef workstation.

Installing and Configuring Chef Workstation

Before installing Chef-client for preparing workstation, let's try to verify whether Chef client is installed or not.

1. Execute chef-client -version to verify it.

```
[mitesh@devops1 Desktop]$ chef-client -version
bash: chef-client: command not found
```

2. Go to the directory where Chef client installable is stored.

```
[mitesh@devops1 Desktop]$ cd chef/
[mitesh@devops1 chef]$ ls
chef-12.9.41-1.el6.x86_64.rpm  chefdk-0.13.21-1.el6.x86_64.rpm
```

3. Run the downloaded Chef client rpm using rpm -ivh chef-<version>.rpm command

```
[mitesh@devops1 chef]$ rpm -ivh chef-12.9.41-1.el6.x86_64.rpm
warning: chef-12.9.41-1.el6.x86_64.rpm: Header V4 DSA/SHA1
 Signature, key ID 83ef826a: NOKEY
error: can't create transaction lock on /var/lib/rpm/.rpm.lock
```

```
(Permission denied)
```

4. Permission is denied to execute hence use sudo to run the command and verify the installation process.

```
[mitesh@devops1 chef]$ sudo rpm -ivh chef-12.9.41-
 1.el6.x86_64.rpm
[sudo] password for mitesh:
warning: chef-12.9.41-1.el6.x86_64.rpm: Header V4 DSA/SHA1
 Signature, key ID 83ef826a: NOKEY
Preparing...
 ########################################### [100%]
 1:chef
 ########################################### [100%]
Thank you for installing Chef!
```

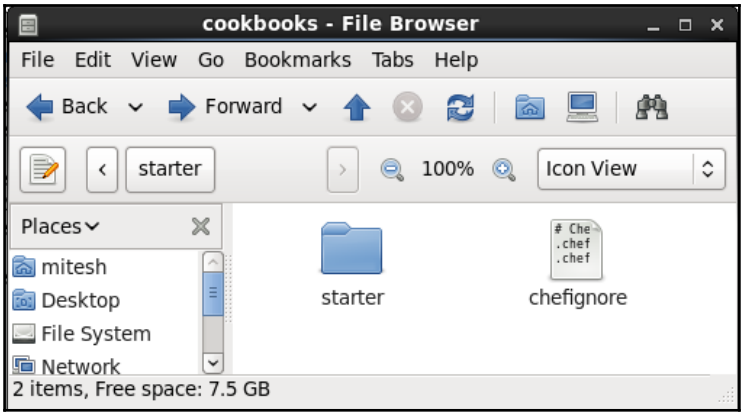
5. After successful installation, let's verify Chef client version.

```
[mitesh@devops1 chef]$ chef-client -version
Chef: 12.9.41
```

Now, next step is to use Starter Kit that we downloaded while creating account in Hosted Chef.

6. Extract the chef-repo compressed file and verify the content. Copy the .chef directory into root or user folder:

7. Verify the cookbooks folder available in chef-repo directory:



8. In .chef directory, open the knife.rb file that contains different configurations. All the configurations are already available. Adjust path of cookbooks directory if needed.

For more information on knife configuration options, visit at: http://docs.chef.io/config_rb_knife.html

```
current_dir = File.dirname(__FILE__)
log_level                :info
log_location             STDOUT
node_name                "discovertechno51"
client_key               "#{current_dir}/
                         discovertechno51.pem"
validation_client_name   "dtechno-validator"
validation_key           "#{current_dir}/dtechno-
                         validator.pem"
chef_server_url          "https://api.chef.io/
                         organizations/dtechno"
cookbook_path            ["#{current_dir}/../cookbooks"]
```

9. Chef Workstation configuration is completed. Next step is to converge the node using Chef workstation.

Converging Chef node using Chef Workstation

First of all, let's login to Chef Workstation which we have setup.

1. Open the terminal and verify the IP address with ifconfig command.

```
[root@devops1 chef-repo]# ifconfig
 eth3      Link encap:Ethernet  HWaddr 00:0C:29:D9:30:7F
           inet addr:192.168.1.35  Bcast:192.168.1.255  Mask:255.255.255.0
           inet6 addr: fe80::20c:29ff:fed9:307f/64 Scope:Link
           UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
           RX packets:841351 errors:0 dropped:0 overruns:0 frame:0
           TX packets:610551 errors:0 dropped:0 overruns:0 carrier:0
           collisions:0 txqueuelen:1000
           RX bytes:520196141 (496.0 MiB)  TX bytes:278125183 (265.2 MiB)
 lo        Link encap:Local Loopback
           inet addr:127.0.0.1  Mask:255.0.0.0
           inet6 addr: ::1/128 Scope:Host
           UP LOOPBACK RUNNING  MTU:65536  Metric:1
           RX packets:1680 errors:0 dropped:0 overruns:0 frame:0
           TX packets:1680 errors:0 dropped:0 overruns:0 carrier:0
           collisions:0 txqueuelen:0
           RX bytes:521152 (508.9 KiB)  TX bytes:521152 (508.9 KiB)
```

2. Verify the knife version installed on the Chef workstation with knife –version command.

```
[root@devops1 chef]# knife --version
Chef: 12.9.41
```

3. knife node list command is used to get list of nodes served by Chef server. In our case Hosted Chef. As we haven't converged any single node so list will be empty.

```
[root@devops1 chef-repo]# knife node list
```

4. Create a virtual machine using VMware workstation of Virtual box. Install CentOS. Once VM is ready, find out its IP address and note it.
5. In Chef Workstation, open terminal and try to take ssh of the node or VM created recently.

```
[root@devops1 chef-repo]# ssh root@192.168.1.37
```

6. The authenticity of host '192.168.1.37 (192.168.1.37)' can't be established.

```
RSA key fingerprint is 4b:56:28:62:53:59:e8:e0:5e:5f:54:08:c1:0c:1e:6c.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added '192.168.1.37' (RSA) to the list of known hosts.
root@192.168.1.37's password:
Last login: Thu May 28 10:26:06 2015 from 192.168.1.15
```

7. Now, we have taken ssh of node from Chef workstation. Verify IP address and we know we are accessing a different machine by remote access or ssh access.

```
[root@localhost ~]# ifconfig
 eth1      Link encap:Ethernet  HWaddr 00:0C:29:44:9B:4B
           inet addr:192.168.1.37  Bcast:192.168.1.255  Mask:255.255.255.0
           inet6 addr: fe80::20c:29ff:fe44:9b4b/64 Scope:Link
           UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
           RX packets:11252 errors:0 dropped:0 overruns:0 frame:0
           TX packets:6628 errors:0 dropped:0 overruns:0 carrier:0
           collisions:0 txqueuelen:1000
           RX bytes:14158681 (13.5 MiB)  TX bytes:466365 (455.4 KiB)
 lo        Link encap:Local Loopback
           inet addr:127.0.0.1  Mask:255.0.0.0
           inet6 addr: ::1/128 Scope:Host
           UP LOOPBACK RUNNING  MTU:65536  Metric:1
           RX packets:59513 errors:0 dropped:0 overruns:0 frame:0
           TX packets:59513 errors:0 dropped:0 overruns:0 carrier:0
           collisions:0 txqueuelen:0
           RX bytes:224567119 (214.1 MiB)  TX bytes:224567119 (214.1 MiB)
[root@localhost ~]#
```

8. Let's verify Node virtual machine; In my VM Chef client was already installed hence execution of rpm -qa *chef* command gave me result.

```
[root@localhost Desktop]# rpm -qa *chef*
chef-12.3.0-1.el6.x86_64
```

9. Let's remove the Chef client installation using yum remove command.

```
[root@localhost Desktop]# yum remove chef-12.3.0-1.el6.x86_64
Loaded plugins: fastestmirror, refresh-packagekit, security
Setting up Remove Process
Resolving Dependencies
--> Running transaction check
---> Package chef.x86_64 0:12.3.0-1.el6 will be erased
--> Finished Dependency Resolution
Dependencies Resolved
================================================================
 Package      Arch          Version             Repository       Size
================================================================
```

```
Removing:
 chef        x86_64        12.3.0-1.el6          installed        125 M
Transaction Summary
================================================================
Remove       1 Package(s)
Installed size: 125 M
Is this ok [y/N]: y
Downloading Packages:
Running rpm_check_debug
Running Transaction Test
Transaction Test Succeeded
Running Transaction
  Erasing    : chef-12.3.0-1.el6.x86_64                          1/1
  Verifying  : chef-12.3.0-1.el6.x86_64                          1/1
Removed:
  chef.x86_64 0:12.3.0-1.el6
Complete!
You have new mail in /var/spool/mail/root
```

10. We have removed chef client; verify again.

```
[root@localhost Desktop]# chef-client -version
bash: chef-client: command not found
```

11. Let's remove Tomcat installation also if it is installed on the node.

```
[root@localhost Desktop]# yum remove tomcat6
Loaded plugins: fastestmirror, refresh-packagekit, security
Setting up Remove Process
Resolving Dependencies
--> Running transaction check
---> Package tomcat6.x86_64 0:6.0.24-83.el6_6 will be erased
--> Processing Dependency: tomcat6 = 6.0.24-83.el6_6 for package: tomcat6-admin-
  webapps-6.0.24-83.el6_6.x86_64
--> Running transaction check
---> Package tomcat6-admin-webapps.x86_64 0:6.0.24-83.el6_6 will be erased
--> Finished Dependency Resolution
Dependencies Resolved
================================================================
 Package                Arch      Version              Repository    Size
================================================================
Removing:
 tomcat6                x86_64    6.0.24-83.el6_6      @updates     188 k
Removing for dependencies:
 tomcat6-admin-webapps  x86_64    6.0.24-83.el6_6      @updates      62 k
Transaction Summary
================================================================
Remove       2 Package(s)
```

```
Installed size: 250 k
Is this ok [y/N]: y
Downloading Packages:
Running rpm_check_debug
Running Transaction Test
Transaction Test Succeeded
Running Transaction
  Erasing    : tomcat6-admin-webapps-6.0.24-83.el6_6.x86_64              1/2
  Erasing    : tomcat6-6.0.24-83.el6_6.x86_64                            2/2
warning: /etc/tomcat6/server.xml saved as /etc/tomcat6/server.xml.rpmsave
warning: /etc/tomcat6/logging.properties saved as /etc/tomcat6/
 logging.properties.rpmsave
warning: /etc/sysconfig/tomcat6 saved as /etc/sysconfig/tomcat6.rpmsave
  Verifying  : tomcat6-admin-webapps-6.0.24-83.el6_6.x86_64              1/2
  Verifying  : tomcat6-6.0.24-83.el6_6.x86_64                            2/2
Removed:
 tomcat6.x86_64 0:6.0.24-83.el6_6
Dependency Removed:
 tomcat6-admin-webapps.x86_64 0:6.0.24-83.el6_6
Complete!
You have new mail in /var/spool/mail/root
[root@localhost Desktop]# yum remove tomcat6
Loaded plugins: fastestmirror, refresh-packagekit, security
Setting up Remove Process
No Match for argument: tomcat6
Loading mirror speeds from cached hostfile
 * base: centos.excellmedia.net
 * extras: centos.excellmedia.net
 * rpmforge: ftp.riken.jp
 * updates: centos.excellmedia.net
Package(s) tomcat6 available, but not installed.
No Packages marked for removal
```

12. Verify JDK installation on node.

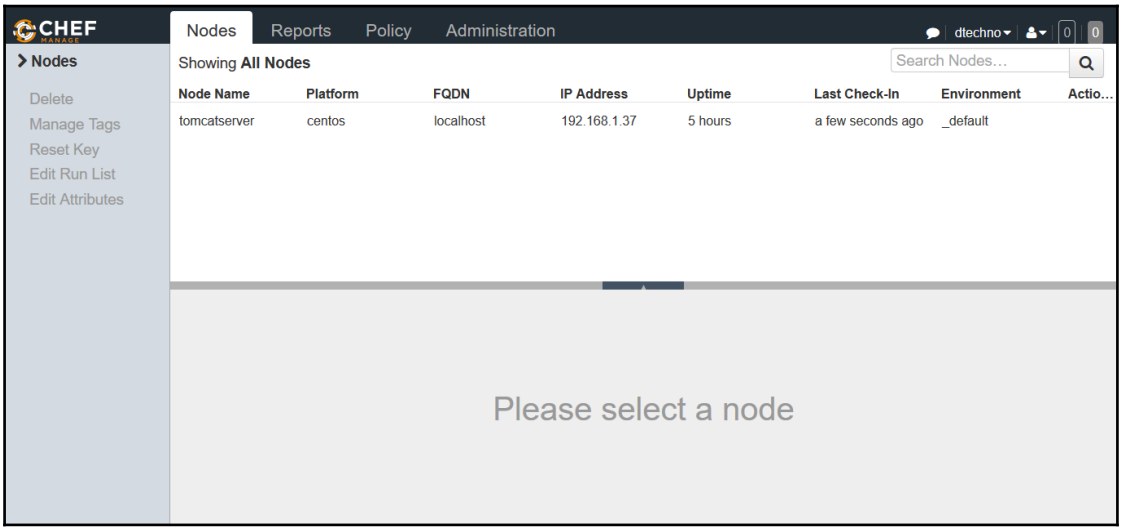
```
[root@localhost Desktop]# java -version
java version "1.7.0_75"
OpenJDK Runtime Environment (rhel-2.5.4.0.el6_6-x86_64 u75-b13)
OpenJDK 64-Bit Server VM (build 24.75-b04, mixed mode)
```

13. Exit from the SSH of node virtual machine. Now we are are having control of Chef workstation machine and we will try to converge the node VM we have recently accessed remotely.
14. Use knife command to converge the node. Give IP address / DNS name, user, password, and name of the node.
15. Verify the output:

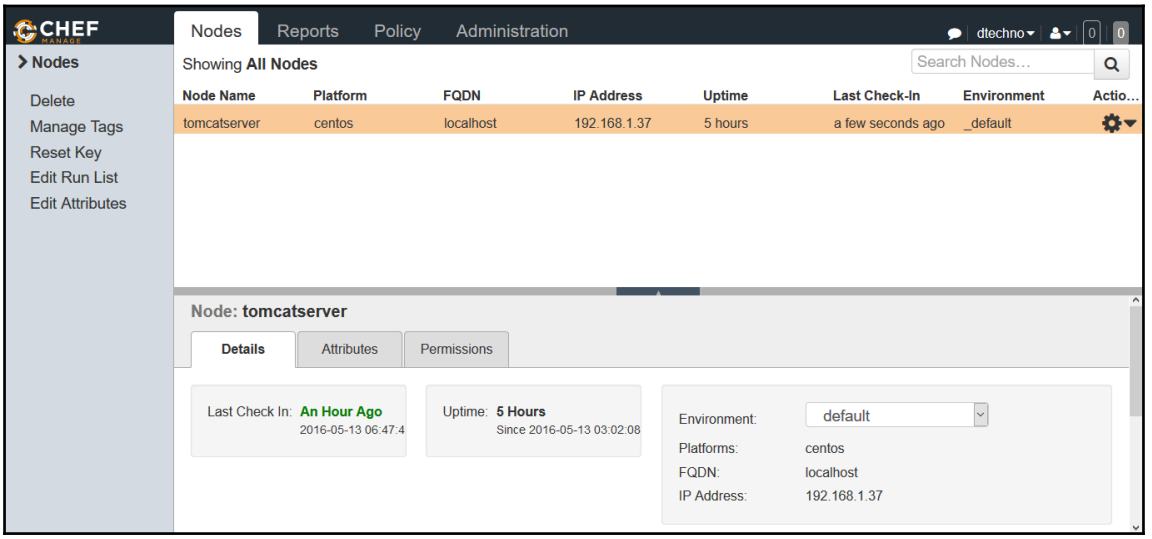
```
[root@devops1 chef-repo]# knife bootstrap 192.168.1.37 -x root -P cloud@123
 -N tomcatserver
Doing old-style registration with the validation key at /home/mitesh/chef-
 repo/.chef/dtechno-validator.pem...
Delete your validation key in order to use your user credentials instead
Connecting to 192.168.1.37
192.168.1.37 -----> Installing Chef Omnibus (-v 12)
192.168.1.37 downloading https://omnitruck-direct.chef.io/chef/install.sh
192.168.1.37 to file /tmp/install.sh.26574/install.sh
192.168.1.37 trying wget...
192.168.1.37 el 6 x86_64
192.168.1.37 Getting information for chef stable 12 for el...
192.168.1.37 downloading https://omnitruck-direct.chef.io/stable/chef/
 metadata?v=12&p=el&pv=6&m=x86_64
192.168.1.37 to file /tmp/install.sh.26586/metadata.txt
192.168.1.37 trying wget...
192.168.1.37 sha1 859bc9be9a40b8b13fb88744079ceef1832831b0
192.168.1.37 sha256 c43f48e5a2de56e4eda473a3ee0a80aa1aaa6c8621d90
 84e033d8b9cf3efc328
192.168.1.37 url https://packages.chef.io/stable/el/6/chef-12.9.41-
 1.el6.x86_64.rpm
192.168.1.37 version 12.9.41
192.168.1.37 downloaded metadata file looks valid...
192.168.1.37 downloading https://packages.chef.io/stable/el/6/chef-12.9.41-
 1.el6.x86_64.rpm
192.168.1.37 to file /tmp/install.sh.26586/chef-12.9.41-1.el6.x86_64.rpm
192.168.1.37 trying wget...
192.168.1.37 Comparing checksum with sha256sum...
192.168.1.37 Installing chef 12
192.168.1.37 installing with rpm...
192.168.1.37 warning: /tmp/install.sh.26586/chef-12.9.41-1.el6.x86_64.rpm:
 Header V4 DSA/SHA1 Signature, key ID 83ef826a: NOKEY
192.168.1.37 Preparing...
 ########################################### [100%]
192.168.1.37 1:chef
 ########################################### [100%]
192.168.1.37 Thank you for installing Chef!
192.168.1.37 Starting the first Chef Client run...
192.168.1.37 Starting Chef Client, version 12.9.41
192.168.1.37 Creating a new client identity for tomcatserver using the
 validator key.
192.168.1.37 resolving cookbooks for run list: []
192.168.1.37 Synchronizing Cookbooks:
192.168.1.37 Installing Cookbook Gems:
192.168.1.37 Compiling Cookbooks...
192.168.1.37 [2016-05-12T23:47:49-07:00] WARN: Node tomcatserver has an
 empty run list.
192.168.1.37 Converging 0 resources
```

```
192.168.1.37
192.168.1.37 Running handlers:
192.168.1.37 Running handlers complete
192.168.1.37 Chef Client finished, 0/0   resources updated in 37 seconds
```

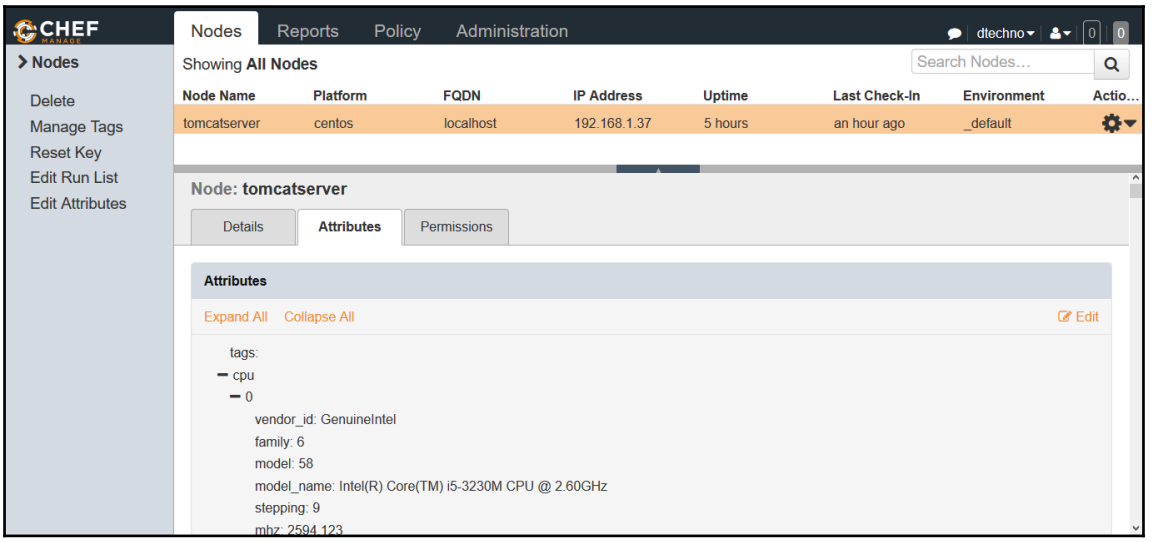
16. There was no run list or role associated with the knife command but convergence is successful.
17. Let's verify Hosted Chef account. We can see the **Node Name** and **IP Address** in the **Nodes** section of dashboard. Click on the **Nodes** and verify details:



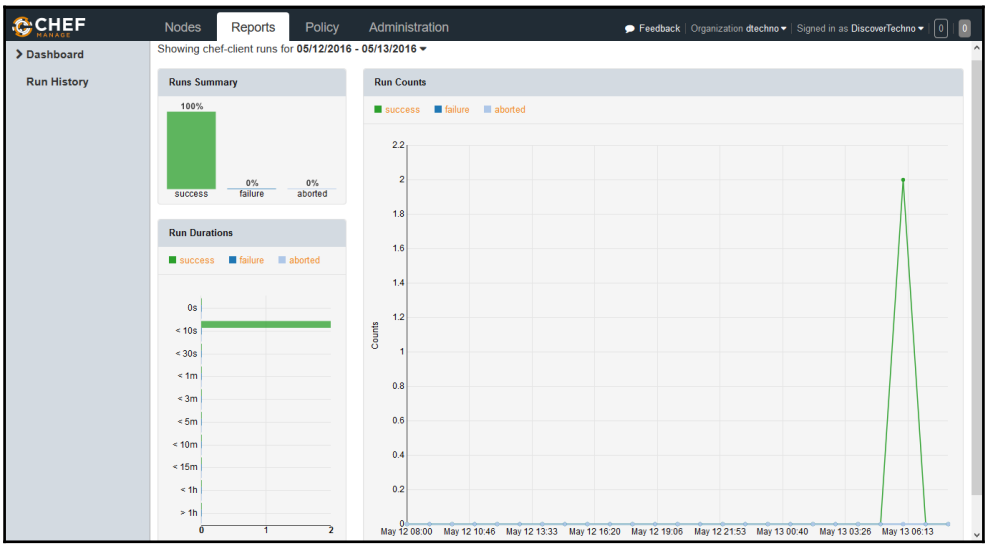
18. Select a node and check **Details**, **Attributes** associated with it, and **Permissions** as shown below:



19. Verify **cpu** attributes associated with the node and other details:



20. Convergence process was successful and we can see that in **Reports** section of the Hosted Chef account:



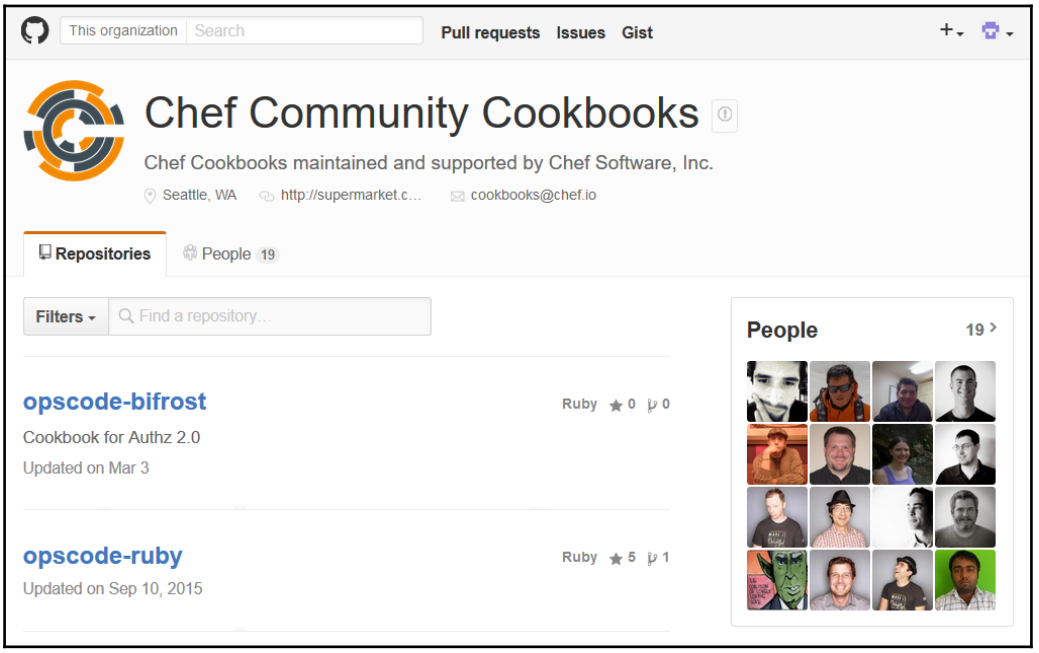
Till now, we have seen how to created Hosted Chef account, how to configure Chef workstation, and how to converge node.

Now it is time to install software packages using cookbooks. The question should be, why we want to do it?

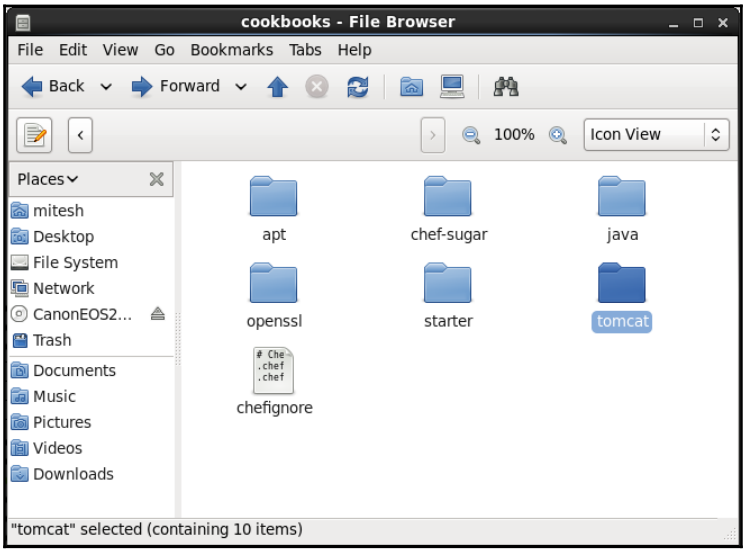
Let's revisit the context. We want to create an end to end pipeline where application source files are compiled, unit tests are executed, package file is created, new virtual machine is created, runtime environment is setup, and finally the deployment.

To set up runtime environment automatically, we would like to use Chef community cookbooks.

1. Visit https://github.com/opscode-cookbooks and find all community cookbooks which are required to setup runtime environment as shown below:



2. We are using Sample spring application that is PetClinic application. For JEE application, we need to install Java and Tomcat for running the application.
3. Download the tomcat cookbook from https://supermarket.chef.io/cookbooks/tomcat and go the **Dependencies** section on that page. Without dependencies uploaded on Chef server, we can't upload tomcat cookbook on the Chef server to use it.
4. Download OpenSSL and Chef Sugar from https://supermarket.chef.io/cookbooks/openssl and https://supermarket.chef.io/cookbooks/chef-sugar respectively.
5. For Java installation, download the cookbook https://supermarket.chef.io/cookbooks/java and its dependency as well https://supermarket.chef.io/cookbooks/apt. Extract all compressed file in the cookbooks directory:



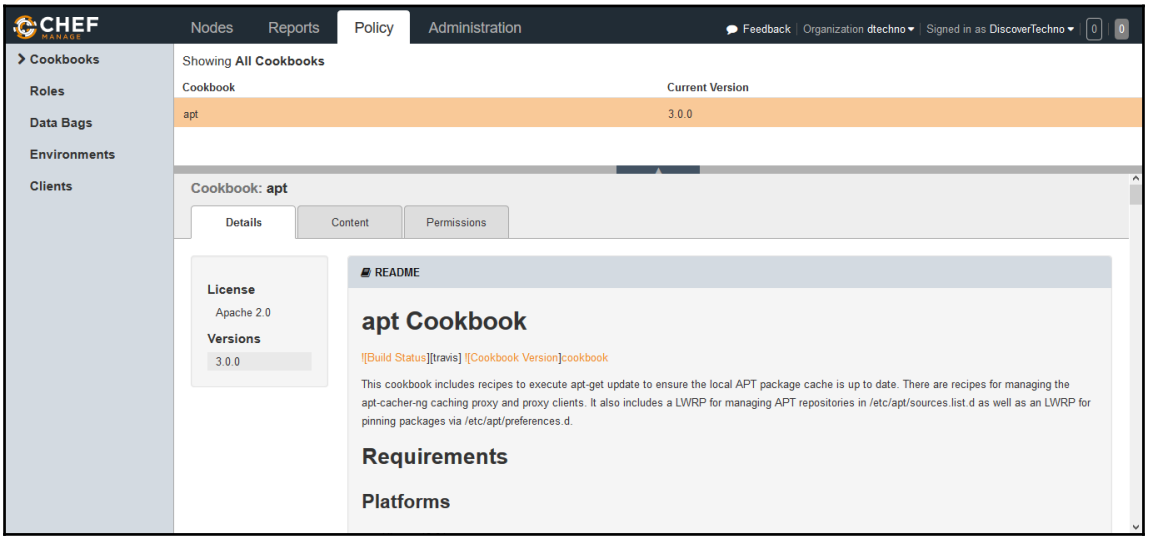
6. Go to cookbooks directory in the terminal and verify the sub-directories of community cookbooks.

```
[root@devops1 cookbooks]# ls
apt  chefignore  chef-sugar  java  openssl  starter  tomcat
[root@devops1 cookbooks]# cd ..
```

7. Upload the apt cookbook with knife cookbook upload apt command.

```
[root@devops1 chef-repo]# knife cookbook upload apt
Uploading apt          [3.0.0]
Uploaded 1 cookbook.
```

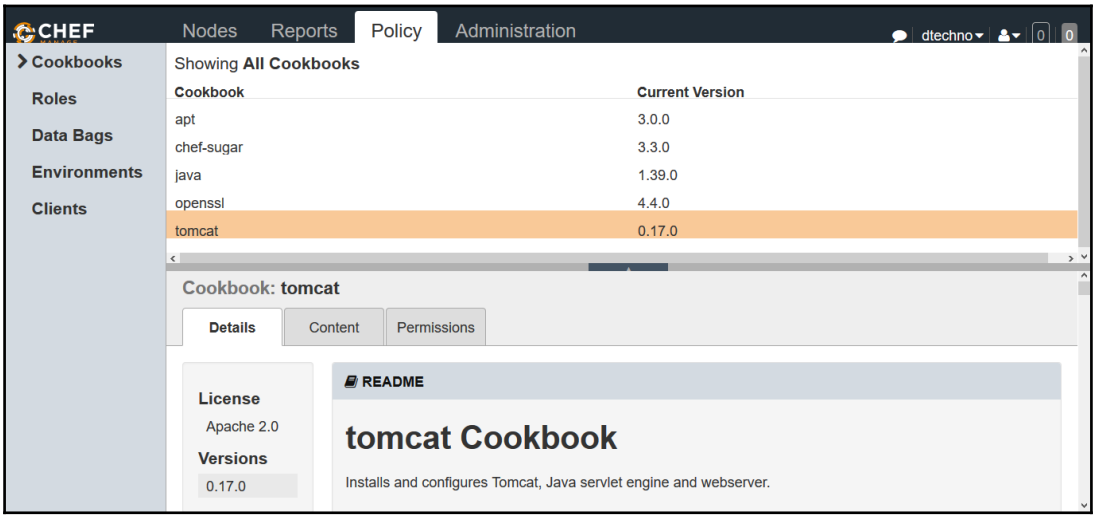
8. Verify the **Cookbooks** section in the Hosted Chef whether apt Cookbook is uploaded or not.



9. Make sure to upload all dependencies first, else it will give error. Upload all other cookbooks in order.

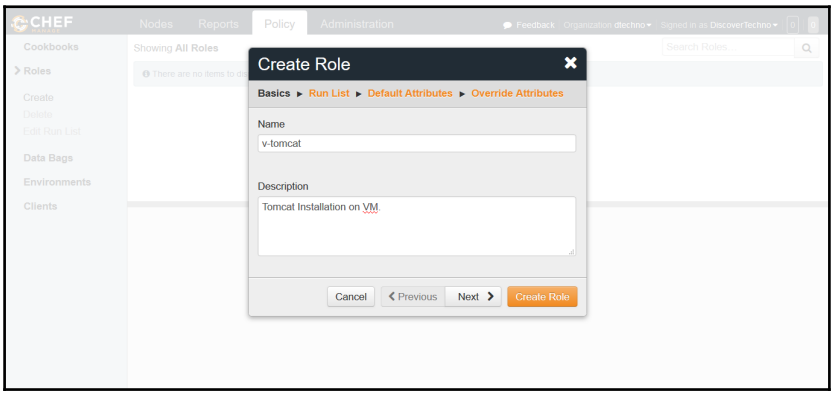
```
[root@devops1 chef-repo]# knife cookbook upload chef-sugar
Uploading chef-sugar     [3.3.0]
Uploaded 1 cookbook.
[root@devops1 chef-repo]# knife cookbook upload java
Uploading java           [1.39.0]
Uploaded 1 cookbook.
[root@devops1 chef-repo]# knife cookbook upload openssl
Uploading openssl        [4.4.0]
Uploaded 1 cookbook.
[root@devops1 chef-repo]# knife cookbook upload tomcat
Uploading tomcat         [0.17.0]
Uploaded 1 cookbook.
```

10. Once all cookbooks are uploaded, verify in Hosted Chef account:



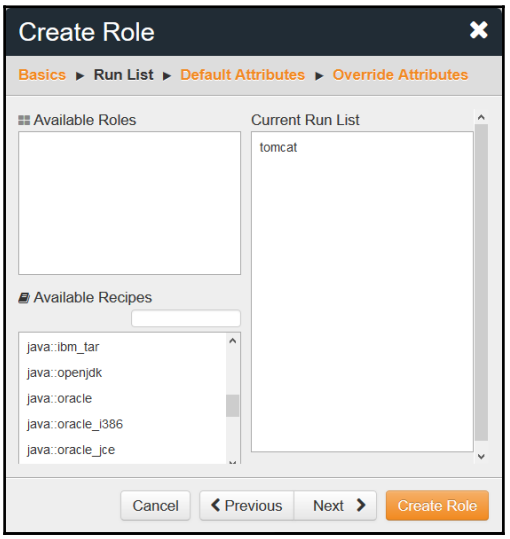
Once all cookbooks are uploaded successfully, we need to create a Role. A role is defined for a specific function and it provides a path for different patterns and workflow processes. For an example, the web server role can consist of Tomcat server recipes and any custom attributes.

1. Go to **policy** section and create a role. Provide **Name** and **Description** and click on **Next** as shown:

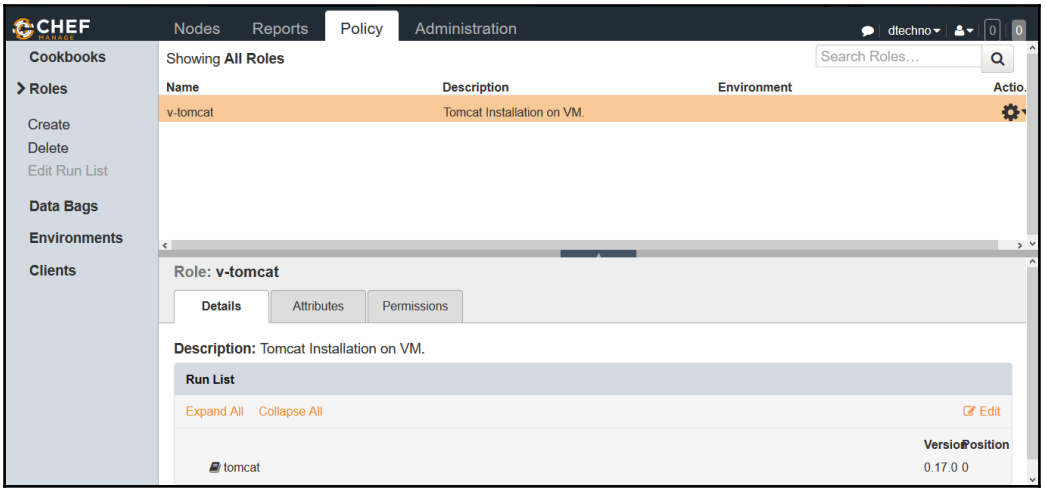


2. A **Run List** keeps roles/recipes in a proper manner and order. We can say that run-list describes the specification of a node. Select Tomcat from the **Available**

Recipes section and drag it to **Current Run List** section and click on **Create Role**.



3. Verify Role details in Hosted Chef dashboard:



4. Now we are ready to associate role while converging the node. Add role to the node with knife node run_list add tomcatserver "role[v-tomcat]" command

```
[root@devops1 chef-repo]# knife node run_list add tomcatserver "role[v-tomcat]"
tomcatserver:
 run_list: role[v-tomcat]
[root@devops1 chef-repo]#
```

5. Role is associated with the node now and next time chef client will run on the node, it will see whether it is in the sync with its assignment or not. If not then it will execute the steps to bring the status in the compliance with the role assigned.

```
[root@localhost Desktop]# chef-client
Starting Chef Client, version 12.9.41
resolving cookbooks for run list: ["tomcat"]
Synchronizing Cookbooks:
  - tomcat (0.17.0)
  - chef-sugar (3.3.0)
  - java (1.39.0)
  - apt (3.0.0)
  - openssl (4.4.0)
Installing Cookbook Gems:
Compiling Cookbooks...
[2016-05-13T02:46:48-07:00] WARN:
Chef::Provider::AptRepository already exists!
Cannot create deprecation class for LWRP provider apt_repository from
cookbook apt
[2016-05-13T02:46:48-07:00] WARN: AptRepository already exists!  Deprecation
class overwrites Custom resource apt_repository from cookbook apt
Converging 3 resources
Recipe: tomcat::default
  * yum_package[tomcat6] action install
    - install version 6.0.24-94.el6_7 of package tomcat6
  * yum_package[tomcat6-admin-webapps] action install
    - install version 6.0.24-94.el6_7 of package tomcat6-admin-webapps
  * tomcat_instance[base] action configure (up to date)
  * directory[/usr/share/tomcat6/lib/endorsed] action create (up to date)
  * template[/etc/sysconfig/tomcat6] action create
    - update content in file /etc/sysconfig/tomcat6 from 10e169 to d7a9c0
    --- /etc/sysconfig/tomcat6 2016-03-22 14:33:38.000000000 -0700
    +++ /etc/sysconfig/.chef-tomcat620160513-38410-1ok6v3f    2016-05-13
2:56:00.766994188 -0700
    @@ -1,3 +1,9 @@
    +#
    +# Dynamically generated by Chef on localhost
    +#
    +# Local modifications will be overwritten by Chef.
    +#
    +# Service-specific configuration file for tomcat6. This will be sourced
       by the SysV init script after the global configuration file
     # /etc/tomcat6/tomcat6.conf, thus allowing values to be overridden in
```

```
@@ -15,29 +21,28 @@
 #
 # Where your java installation lives
-#JAVA_HOME="/usr/lib/jvm/java"
+JAVA_HOME=
 # Where your tomcat installation lives
-#CATALINA_BASE="/usr/share/tomcat6"
-#CATALINA_HOME="/usr/share/tomcat6"
-#JASPER_HOME="/usr/share/tomcat6"
-#CATALINA_TMPDIR="/var/cache/tomcat6/temp"
+CATALINA_BASE="/usr/share/tomcat6"
+CATALINA_HOME="/usr/share/tomcat6"
+JASPER_HOME="/usr/share/tomcat6"
+CATALINA_TMPDIR="/var/cache/tomcat6/temp"

 # You can pass some parameters to java here if you wish to
-#JAVA_OPTS="-Xminf0.1 -Xmaxf0.3"
+JAVA_OPTS="-Xmx128M -Djava.awt.headless=true"

 # Use JAVA_OPTS to set java.library.path for libtcnative.so
 #JAVA_OPTS="-Djava.library.path=/usr/lib64"
 # What user should run tomcat
-#TOMCAT_USER="tomcat"
-#TOMCAT_GROUP="${TOMCAT_GROUP:-`id -gn $TOMCAT_USER`}"
-#
+TOMCAT_USER="tomcat"
+
 # You can change your tomcat locale here
 #LANG="en_US"

 # Run tomcat under the Java Security Manager
-#SECURITY_MANAGER="false"
+SECURITY_MANAGER="false"

 # Time to wait in seconds, before killing process
 #SHUTDOWN_WAIT="30"
@@ -48,8 +53,11 @@
 # Set the TOMCAT_PID location
 #CATALINA_PID="/var/run/tomcat6.pid"

-# Connector port is 8080 for this tomcat6 instance
-#CONNECTOR_PORT="8080"
+# JVM parameters passed only for start and run commands
+CATALINA_OPTS=""
+
+# Endorse .jar files in this directory
+JAVA_ENDORSED_DIRS="/usr/share/tomcat6/lib/endorsed"
```

```
  # If you wish to further customize your tomcat environment,
  # put your own definitions here
 - change mode from '0664' to '0644'
 - restore selinux security context
* template[/etc/tomcat6/server.xml] action create
 - update content in file /etc/tomcat6/server.xml from 178c5e to 71d23a
 --- /etc/tomcat6/server.xml 2016-03-22 14:31:26.000000000 -0700
 +++ /etc/tomcat6/.chef-server.xml20160513-38410-wjv3fl    2016-05-13
 2:56:01.693994187 -0700
 @@ -1,5 +1,9 @@
  <?xml version='1.0' encoding='utf-8'?>
  <!--
 +  Dynamically generated by Chef on localhost
 +  Local modifications will be overwritten by Chef.
 +-->
 +<!--
    Licensed to the Apache Software Foundation (ASF) under one or more
    contributor license agreements.  See the NOTICE file distributed with
    this work for additional information regarding copyright ownership.
 @@ -22,7 +26,9 @@
  <Server port="8005" shutdown="SHUTDOWN">
    <!--APR library loader. Documentation at /docs/apr.html -->
 +  <!--
    <Listener className="org.apache.catalina.core.AprLifecycleListener"
     SSLEngine="on" />    +  -->

    <!--Initialize Jasper prior to webapps are loaded. Documentation at
     /docs/jasper-howto.html -->
    <Listener className="org.apache.catalina.core.JasperListener"/>
    <!-- Prevent memory leaks due to use of particular java/javax APIs-->
 @@ -46,19 +52,18 @@
    </GlobalNamingResources>
    <!-- A "Service" is a collection of one or more "Connectors" that
      share a single "Container" Note:  A "Service" is not itself a
      "Container", + a single "Container" Note:  A "Service" is not
      itself a "Container", so you may not define subcomponents such as
      "Valves" at this level.
     Documentation at /docs/config/service.html
     -->
    <Service name="Catalina">
 -
 +  <!--The connectors can use a shared executor, you can define one or
     more named thread pools-->
    <!--
 -   <Executor name="tomcatThreadPool" namePrefix="catalina-exec-"
 +   <Executor name="tomcatThreadPool" namePrefix="catalina-exec-"
     maxThreads="150" minSpareThreads="4"/>
     -->
```

```
-
-
+   <!-- A "Connector" represents an endpoint by which requests are
     received and responses are returned. Documentation at :
  Java HTTP Connector: /docs/config/http.html (blocking & non-blocking)
@@ -66,30 +71,36 @@
         APR (HTTP/AJP) Connector: /docs/apr.html
         Define a non-SSL HTTP/1.1 Connector on port 8080
    -->
-   <Connector port="8080" protocol="HTTP/1.1"
-              connectionTimeout="20000"
-              redirectPort="8443" />
+   <Connector port="8080" protocol="HTTP/1.1"
+              connectionTimeout="20000"
+              URIEncoding="UTF-8"
+              redirectPort="8443"
+             />
    <!-- A "Connector" using the shared thread pool-->
    <!--
    <Connector executor="tomcatThreadPool"
-              port="8080" protocol="HTTP/1.1"
-              connectionTimeout="20000"
+              port="8080" protocol="HTTP/1.1"
+              connectionTimeout="20000"
              redirectPort="8443" />
-   -->
+    -->
    <!-- Define a SSL HTTP/1.1 Connector on port 8443
-
+        This connector uses the JSSE configuration, when using APR,
        the connector should be using the OpenSSL style configuration
        described in the APR documentation -->
-   <!--
    <Connector port="8443" protocol="HTTP/1.1" SSLEnabled="true"
+              keystoreFile="/etc/tomcat6/keystore.jks"
+              keystorePass="DBtN03iR_YligSPG5zW4"
+              keystoreType="jks"
+              truststorePass="DBtN03iR_YligSPG5zW4"
              maxThreads="150" scheme="https" secure="true"
              clientAuth="false" sslProtocol="TLS" />
-   -->
    <!-- Define an AJP 1.3 Connector on port 8009 -->
-   <Connector port="8009" protocol="AJP/1.3" redirectPort="8443" />
+   <Connector port="8009"
+              protocol="AJP/1.3"
+              tomcatAuthentication="true"
+              redirectPort="8443" />
-
```

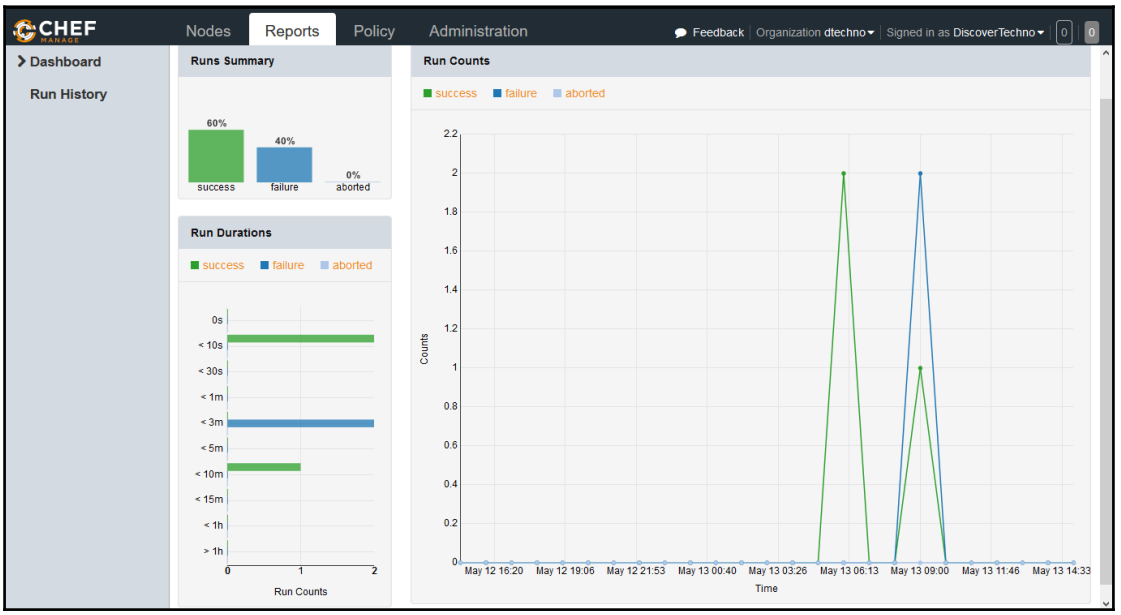
```
    <!-- An Engine represents the entry point (within Catalina) that
         processes every request.  The Engine implementation for Tomcat
         stand alone analyzes the HTTP headers included with the
         request, and passes them
@@ -97,16 +108,16 @@
         Documentation at /docs/config/engine.html -->

    <!-- You should set jvmRoute to support load-balancing via AJP ie :
-   <Engine name="Catalina" defaultHost="localhost"
     jvmRoute="jvm1">
-   -->
-   <Engine name="Catalina" defaultHost="localhost">
+   <Engine name="Catalina" defaultHost="localhost" jvmRoute="jvm1">
+   -->
+   <Engine name="Catalina" defaultHost="localhost" >

    <!--For clustering, please take a look at documentation
        at: /docs/cluster-howto.html  (simple how to)
        /docs/config/cluster.html (reference documentation) -->
    <!--
    <Cluster className="org.apache.catalina.ha.tcp.SimpleTcpCluster"/>
-    -->
+    -->

    <!-- The request dumper valve dumps useful debugging information
         about the request and response data received and sent by
         Tomcat.
@@ -127,7 +138,8 @@
     -->
    <Host name="localhost"  appBase="webapps"
     unpackWARs="true" autoDeploy="true"
-          xmlValidation="false" xmlNamespaceAware="false">
+          xmlValidation="false" xmlNamespaceAware="false"
+          >
     <!-- SingleSignOn valve, share authentication between web
          applications
          Documentation at: /docs/config/valve.html -->

@@ -138,7 +150,7 @@
     <!-- Access log processes all example.
          Documentation at: /docs/config/valve.html -->
     <!--
-    <Valve className="org.apache.catalina.valves.AccessLogValve"
      directory="logs"
+    <Valve className="org.apache.catalina.valves.AccessLogValve"
      directory="logs"
      prefix="localhost_access_log." suffix=".txt" pattern="common"
      resolveHosts="false"/>
```

```
      -->
  - change mode from '0664' to '0644'
  - restore selinux security context
* template[/etc/tomcat6/logging.properties] action create
  - update content in file /etc/tomcat6/logging.properties from fb8198 to
    d3364b
  --- /etc/tomcat6/logging.properties     2016-03-22 14:31:26.000000000
      -0700
  +++ /etc/tomcat6/.chef-logging.properties20160513-38410-1jqpw7h  2016-
      05-13 02:56:02.086994187 -0700
 @@ -13,10 +13,12 @@
  # See the License for the specific language governing permissions and
  # limitations under the License.
 -handlers = 1catalina.org.apache.juli.FileHandler,
  2localhost.org.apache.juli.FileHandler,
  3manager.org.apache.juli.FileHandler, 4host-
  manager.org.apache.juli.FileHandler, java.util.logging.ConsoleHandler
 +handlers = 1catalina.org.apache.juli.FileHandler,
  2localhost.org.apache.juli.FileHandler,
  java.util.logging.ConsoleHandler
  .handlers = 1catalina.org.apache.juli.FileHandler,
  java.util.logging.ConsoleHandler
 +.level = INFO
 +
  ############################################################
  # Handler specific properties.
  # Describes specific configuration info for Handlers.
 @@ -30,18 +32,9 @@
 2localhost.org.apache.juli.FileHandler.directory = ${catalina.base}/logs
 2localhost.org.apache.juli.FileHandler.prefix = localhost.

 -3manager.org.apache.juli.FileHandler.level = FINE
 -3manager.org.apache.juli.FileHandler.directory = ${catalina.base}/logs
 -3manager.org.apache.juli.FileHandler.prefix = manager.
 -
 -4host-manager.org.apache.juli.FileHandler.level = FINE
 -4host-manager.org.apache.juli.FileHandler.directory =
  ${catalina.base}/logs
 -4host-manager.org.apache.juli.FileHandler.prefix = host-manager.
 -
  java.util.logging.ConsoleHandler.level = FINE
  java.util.logging.ConsoleHandler.formatter =
  ava.util.logging.SimpleFormatter

 -
  ############################################################
  # Facility specific properties.
  # Provides extra control for each logger.
```

```
@@ -49,17 +42,4 @@

 org.apache.catalina.core.ContainerBase.[Catalina].[localhost].level =
 INFO
org.apache.catalina.core.ContainerBase.[Catalina].[localhost].handlers =
2localhost.org.apache.juli.FileHandler
-
-org.apache.catalina.core.ContainerBase.[Catalina].[localhost].
 /manager].level = INFO
-org.apache.catalina.core.ContainerBase.[Catalina].[localhost].
 [/manager].handlers = 3manager.org.apache.juli.FileHandler
-
-org.apache.catalina.core.ContainerBase.[Catalina].[localhost].[/host-
 manager].level = INFO
-org.apache.catalina.core.ContainerBase.[Catalina].[localhost].[/host-
manager].handlers = 4host-manager.org.apache.juli.FileHandler
-
-# For example, set the com.xyz.foo logger to only log SEVERE
-# messages:
-#org.apache.catalina.startup.ContextConfig.level = FINE
-#org.apache.catalina.startup.HostConfig.level = FINE
-#org.apache.catalina.session.ManagerBase.level = FINE
-#org.apache.catalina.core.AprLifecycleListener.level=FINE
 - change mode from '0664' to '0644'
 - restore selinux security context
* execute[Create Tomcat SSL certificate] action run (up to date)
* service[tomcat6] action start
 - start service service[tomcat6]
* execute[wait for tomcat6] action run
 - execute sleep 5
* service[tomcat6] action enable
 - enable service service[tomcat6]
* execute[wait for tomcat6] action run
 - execute sleep 5
* execute[wait for tomcat6] action nothing (skipped due to action
 :nothing)
* service[tomcat6] action restart
 - restart service service[tomcat6]
* execute[wait for tomcat6] action run
 - execute sleep 5

Running handlers:
Running handlers complete
Chef Client finished, 11/15 resources updated in 09 minutes 59 seconds
You have new mail in /var/spool/mail/root
[root@localhost Desktop]# service tomcat6 status
tomcat6 (pid 39782) is running...                          [  OK  ]
You have new mail in /var/spool/mail/root
```

Observe the above output and we will come to know what exactly happens when convergence takes place.

6. Verify the **Reports** section in the Hosted Chef account to get latest details.



Now we know how to create Hosted Chef account, configure workstation and how to converge the node.

Self-Test Questions

1. In which category Chef falls in from following?
2. Continuous Integration
3. Configuration Management
4. All of the Above
5. None of the Above

1. What are the 3 main components of Chef installation?
2. Chef Server
3. Chef Workstation

4. Chef Node
5. All of the Above
6. None of the Above

1. Which command can be used to check the version of Chef client?
2. chefclient -version
3. chef-client -version
4. chefclient –version
5. chef-client –version
6. None of the Above

1. What is the name of the configuration file in Chef?
2. knife.java
3. knife.py
4. knife.rb
5. knife.sh
6. None of the Above

1. Which command is used for listing node available in Chef server?
2. knife node list
3. knife client list
4. knife node listing
5. knife nodes list
6. None of the Above

Summary

In this chapter, we have covered how we can create Hosted Chef account, how to configure a workstation, how to upload a community cookbook to Hosted Chef account, how to converge a node, how to use community cookbooks to install tomcat, how to verify the convergence of node on Hosted Chef account and how to verify success and failure Reports. Essentially, we are standardizing process of setting up runtime environment from a centralized location. Most of the configuration tools do almost similar things and it can be decided based on experience and other features on the selection of Configuration Management tool. Automating the repetitive process in any field is the key to increase the efficiency and configuration management tools do exactly that in end to end automation of application delivery.

In the next chapter, we will discuss about Docker, one of the most popular and recent buzz word in recent times. It is also one of the most disruptive innovations. We will see how Docker containers are different from Virtual machines, how to install it and some basics of it.

5
Installing and Configuring Docker

"If you cannot do great things, do small things in a great way."
- Napoleon Hill

Docker, yes one of the hot topics of technical discussions in recent times. It is an open source container based technology and considered as one of the disruptive innovations of recent times. Docker containers are isolated packages that contains enough or required components to run an application.

This chapter describes in detail container technology and how it is different from virtual machines by comparing benefits of both. It will cover overview of Docker, its installation and configuration details; it will also cover how to create CentOS container for application deployment.

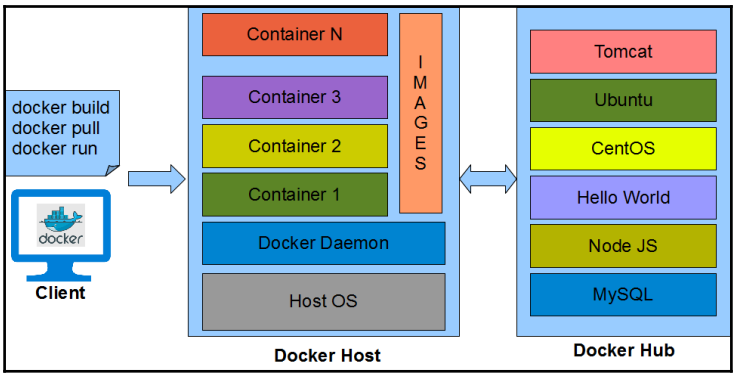
We will also cover Docker hub and basic architecture of Docker. In this chapter we will see how to use tomcat image available on Docker hub and then create a sample image with Java and tomcat installation with Dockerfile.

In this chapter, we will cover the following topics:

- Overview of Docker Container
- Understanding difference between Virtual Machines and Containers
- Installation and Configuration of Docker on CentOS
- Creating a First Docker Container
- Managing Containers

Overview of Docker Container

Docker is an open-source initiative for OS Virtualization that automates the deployment of applications inside software containers. It provides isolated user space and hence provides user based processes, space, and file system. Behind the scene it shares Linux Host Kernel.



Docker has two main Components with Client Server Architecture:

- **Docker Host**: Docker host contains Docker daemon, containers, and images. Docker Engine is an important component that provides the core Docker technology. This core Docker technology enables images and containers concepts. When we install Docker successfully, we run a simple command. In our case we will consider CentOS for the container.

  To run an interactive shell in the CentOS image:

  docker run -i -t ubuntu /bin/bash

  -i flag: Initiates an interactive container

  -t flag: Creates a pseudo-TTY that attaches stdin and stdout

  Image: Centos

  /bin/bash: Starts a shell

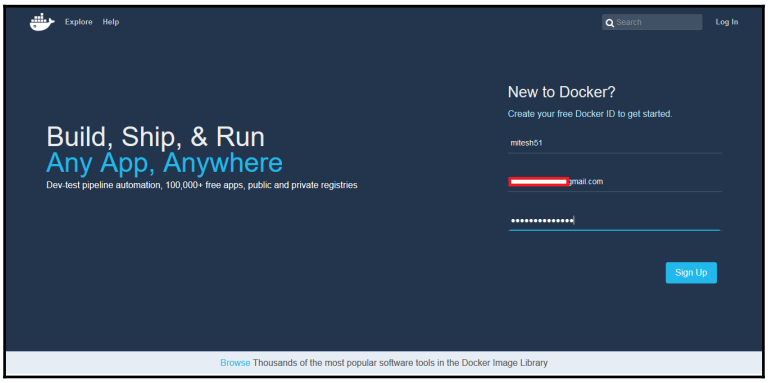
  When we run above command, it verifies whether the centos image is available locally or not. If it is not available, then it will download the image from the Docker Hub.

  Image has filesystem and parameter that can be used at runtime while container is

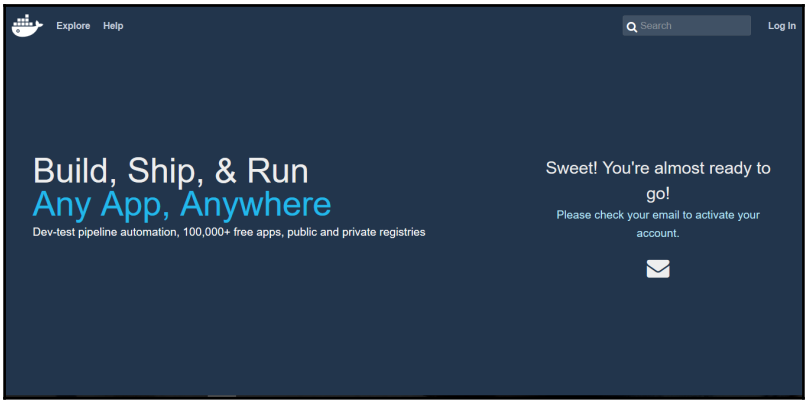
an instance of an Image with a state. It is simple to understand that container changes while Images not.

- **Docker Hub**: Docker hub is a **Software as a Service (SaaS)** for sharing and managing Docker containers. It is a kind of centralized registry service. As a user, we can use it to build and ship applications. It allows to create pipeline to integrate with code repositories, collaboration, image discovery, and automation.

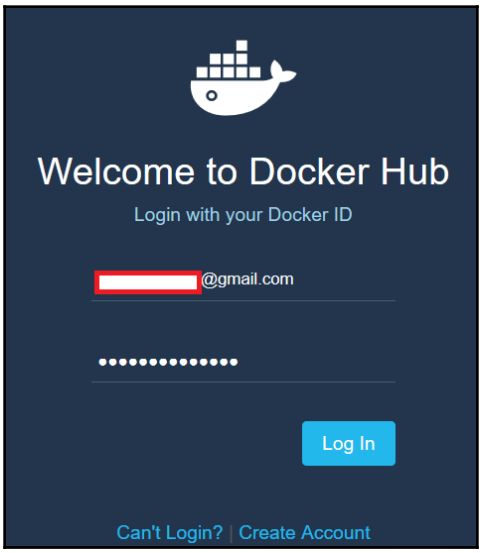
1. Let's navigate to https://hub.docker.com and sign up by providing username, email, and password details:



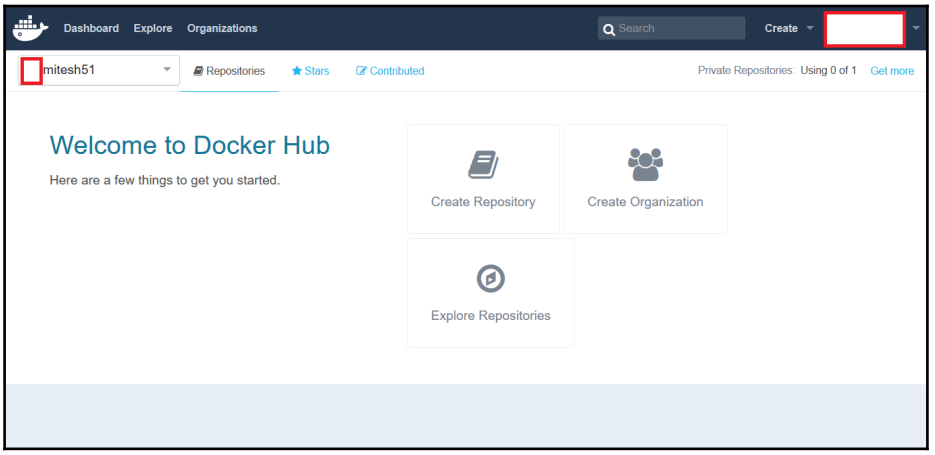
2. Activate account by clicking on the activation link sent to email id mentioned in the sign up process:



3. After successful activation link, login to Docker hub account:

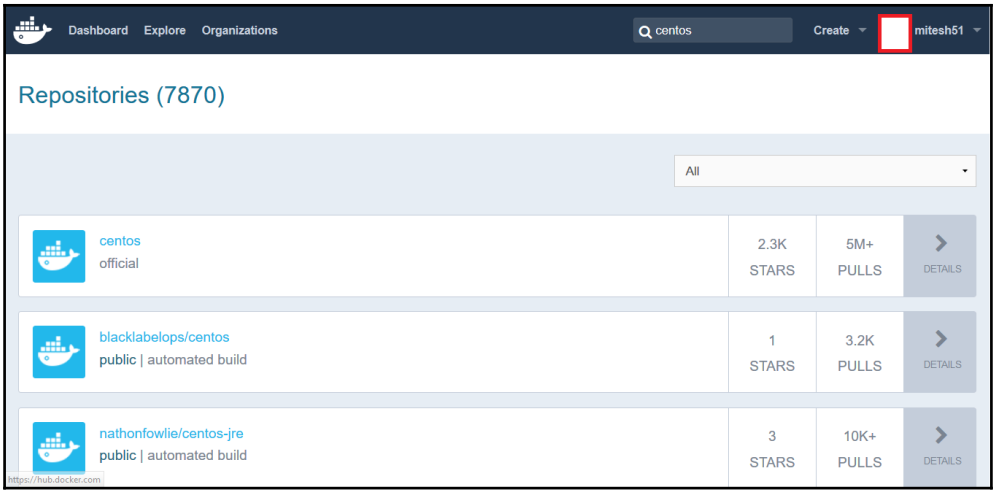


4. Following is the screenshot of Docker Dashboard. Try to explore Docker dashboard as a self exercise:



5. Click on the **Repositories** to find images available in public domain. Search

CentOS image available in the Docker hub and you will get list of all CentOS images available in the Docker hub.



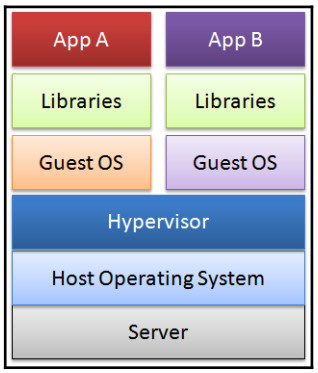
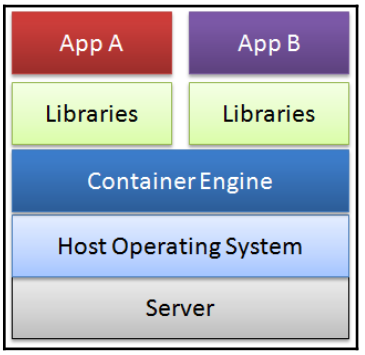
In the next section, we will see why Containers are gaining so much attraction by comparing them with Virtual Machines.

Understanding difference between Virtual Machines and Containers

In the recent times, Cloud computing is part of almost all technical discussions. Usages of virtual machines have served a lot in utilizing resources efficiently. However, Docker containers have given them competition and in fact containers are more effective.

Let's find out basic differences between both and find out the reason behind popularity of containers:

Virtual Machine	**Docker**

In Virtual Machine, we need to install operating system with the related device drivers and hence footprint or size of the virtual machine is huge. For a normal VM with Tomcat and Java installed, it may take up to 10 GB. 	It shares the operating system and device drivers of the host. Containers are created from the images and for tomcat installed container, size is less than 500 MB. 
Overhead of memory management and device drivers. VM is having all the components which a normal physical machine has in terms of operations.	Containers are small in size and hence effectively gives faster and better performance.
In VM, hypervisor abstracts resources.	Containers abstract the operating system.
In VM, the package includes not only the application but also the necessary binaries and libraries, and an entire guest operating system. For example: CentOS 6.7, Windows 2003, and so on.	Containers runs as an isolated user space, processes, and file system in user space on the host operating system itself, and it shares the kernel with other containers. Sharing and resource utilization are at its best in containers and now extra overhead is available. It works with minimum required resources.
Cloud service providers use hypervisor to provide a standard runtime environment for VMs. Hypervisor comes in type 1 and type ii category.	Docker makes it efficient and easier to port applications across environments

In the next section, we will install and configure Docker on CentOS virtual machine.

Installing and Configuring Docker on CentOS

To create a Virtual machine using VMware Workstation or Virtual box, Install CentOS 6.6 or CentOS 6.7.

We are using CentOS 6.7 to run Docker. For CentOS-6, there is a minor issue of package name conflict with a system tray application and its executable, hence the Docker RPM package was called docker-io:

1. Let's install docker-io:

```
[root@localhost Desktop]# yum install docker-io
Loaded plugins: fastestmirror, refresh-packagekit, security
Setting up Install Process
Loading mirror speeds from cached hostfile
 * epel: ftp.riken.jp
Resolving Dependencies
--> Running transaction check
---> Package docker-io.x86_64 0:1.7.1-2.el6 will be installed
--> Processing Dependency: lxc for package: docker-io-1.7.1-2.el6.x86_64
--> Running transaction check
---> Package lxc.x86_64 0:1.0.8-1.el6 will be installed
--> Processing Dependency: lua-lxc(x86-64) = 1.0.8-1.el6 for package: lxc-1.0.8-
    1.el6.x86_64
--> Processing Dependency: lua-alt-getopt for package: lxc-1.0.8-1.el6.x86_64
--> Processing Dependency: liblxc.so.1()(64bit) for package: lxc-1.0.8-
    1.el6.x86_64
--> Running transaction check
---> Package lua-alt-getopt.noarch 0:0.7.0-1.el6 will be installed
---> Package lua-lxc.x86_64 0:1.0.8-1.el6 will be installed
--> Processing Dependency: lua-filesystem for package: lua-lxc-1.0.8-
    1.el6.x86_64
---> Package lxc-libs.x86_64 0:1.0.8-1.el6 will be installed
--> Running transaction check
---> Package lua-filesystem.x86_64 0:1.4.2-1.el6 will be installed
--> Finished Dependency Resolution
Dependencies Resolved
```

Package	Arch	Version	Repository	Size
Installing:				
docker-io	**x86_64**	**1.7.1-2.el6**	**epel**	**4.6 M**
Installing for dependencies:				

lua-alt-getopt	noarch	0.7.0-1.el6	epel	6.9 k
lua-filesystem	x86_64	1.4.2-1.el6	epel	24 k
lua-lxc	x86_64	1.0.8-1.el6	epel	16 k
lxc	x86_64	1.0.8-1.el6	epel	122 k
lxc-libs	x86_64	1.0.8-1.el6	epel	255 k

```
Transaction Summary
================================================================

Install 6 Package(s)

Total download size: 5.0 M
Installed size: 20 M

Is this ok [y/N]: y

Downloading Packages:
(1/6): docker-io-1.7.1-2.el6.x86_64.rpm | 4.6 MB 04:32
(2/6): lua-alt-getopt-0.7.0-1.el6.noarch.rpm | 6.9 kB 00:01
(3/6): lua-filesystem-1.4.2-1.el6.x86_64.rpm | 24 kB 00:01
(4/6): lua-lxc-1.0.8-1.el6.x86_64.rpm | 16 kB 00:01
(5/6): lxc-1.0.8-1.el6.x86_64.rpm | 122 kB 00:03
(6/6): lxc-libs-1.0.8-1.el6.x86_64.rpm | 255 kB 00:11
----------------------------------------------------Total 17 kB/s | 5.0 MB 05:02

Running rpm_check_debug
Running Transaction Test
Transaction Test Succeeded

Running Transaction
 Installing : lxc-libs-1.0.8-1.el6.x86_64 1/6
 Installing : lua-filesystem-1.4.2-1.el6.x86_64 2/6
 Installing : lua-lxc-1.0.8-1.el6.x86_64 3/6
 Installing : lua-alt-getopt-0.7.0-1.el6.noarch 4/6
 Installing : lxc-1.0.8-1.el6.x86_64 5/6
 Installing : docker-io-1.7.1-2.el6.x86_64 6/6
 Verifying : lxc-libs-1.0.8-1.el6.x86_64 1/6
 Verifying : lua-lxc-1.0.8-1.el6.x86_64 2/6
 Verifying : lxc-1.0.8-1.el6.x86_64 3/6
 Verifying : docker-io-1.7.1-2.el6.x86_64 4/6
 Verifying : lua-alt-getopt-0.7.0-1.el6.noarch 5/6
 Verifying : lua-filesystem-1.4.2-1.el6.x86_64 6/6

Installed:
 docker-io.x86_64 0:1.7.1-2.el6
```

```
Dependency Installed:
 lua-alt-getopt.noarch 0:0.7.0-1.el6 lua-filesystem.x86_64 0:1.4.2-1.el6 lua-
  lxc.x86_64 0:1.0.8-1.el6 lxc.x86_64 0:1.0.8-1.el6
  lxc-libs.x86_64 0:1.0.8-1.el6

Complete!
You have new mail in /var/spool/mail/root
```

2. Let's try to run Sample Hello World Image of Docker:

```
[root@localhost Desktop]# docker run hello-world
Post http:///var/run/docker.sock/v1.19/containers/create: dial unix
 /var/run/docker.sock: no such file or directory. Are you trying to connect to a
 TLS-enabled daemon without TLS?
You have new mail in /var/spool/mail/root
```

3. Sample image execution didn't complete successfully as Docker service was not running. Let's verify the Docker installation:

   First, start the Docker service:

```
[root@localhost Desktop]# service docker start
Starting cgconfig service: [  OK  ]
Starting docker:   [  OK  ]
You have new mail in /var/spool/mail/root
```

   Verify status of Docker service:

```
[root@localhost Desktop]# service docker status
docker (pid  12340) is running...
```

So we have successfully installed Docker and verified whether its services are running or not on CentOS 6.7 virtual machine.

Creating a first Docker container

Just to get a feel of Docker, let's run a sample hello-world container which we tried to do earlier without success.

hello-world image is not available locally so it will fetch it from the Docker hub:

```
[root@localhost Desktop]# docker run hello-world
Unable to find image 'hello-world:latest' locally
latest: Pulling from hello-world
d59cd4c39e50: Pull complete
```

```
f1d956dc5945: Pull complete
Digest: sha256:4f32210e234b4ad5cac92efacc0a3d602b02476c754f13d517e1ada048e5a8ba
  Status: Downloaded newer image for hello-world:latest
Hello from Docker.
```

This message shows that your installation appears to be working correctly.

To generate this message, Docker took the following steps:

1. The Docker client contacted the Docker daemon.
2. The Docker daemon pulled the "hello-world" image from the Docker Hub.
3. The Docker daemon created a new container from that image which runs the executable that produces the output you are currently reading.
4. The Docker daemon streamed that output to the Docker client, which sent it to your terminal.

Let's try something more ambitious:

1. You can run an Ubuntu container with:

   ```
   $ docker run -it ubuntu bash
   ```

   Share images, automate workflows, and more with a free Docker Hub account at: https://hub.docker.com
   For more examples and ideas, visit at: https://docs.docker.com/engine/userguide/

   ```
   You have new mail in /var/spool/mail/root
   [root@localhost Desktop]#
   ```

2. Now we have one image available locally. Let's try to create an Ubuntu container and open its bash command directly:

   ```
   [root@localhost Desktop]# docker run -it ubuntu bash
   Unable to find image 'ubuntu:latest' locally
   latest: Pulling from ubuntu
   dd25ab30afb3: Pull complete
   a83540abf000: Pull complete
   630aff59a5d5: Pull complete
   cdc870605343: Pull complete
   686477c12982: Pull complete
   Digest:
    sha256:5718d664299eb1db14d87db7bfa6945b28879a67b74f36da3e34f5914866b71c
   Status: Downloaded newer image for ubuntu:latest
   ```

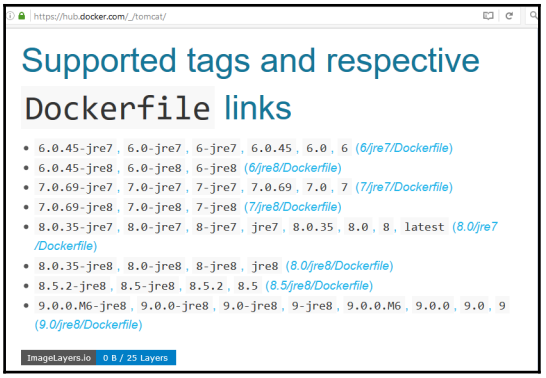
3. Use Docker images command to verify the existing images available locally:

```
[root@localhost Desktop]# docker images
REPOSITORY   TAG      IMAGE ID      CREATED       VIRTUAL SIZE
ubuntu       latest   686477c12982  5 weeks ago      120.7 MB
hello-world  latest   f1d956dc5945  6 weeks ago        967 B
```

After these two examples, let's try to understand client server architecture of Docker using another example of tomcat container.

Let's recollect our main objective. We want to deploy sample spring application named Pet-clinic in tomcat server. For that in the rest of the section we will try to use existing tomcat image and also create sample image with tomcat installation.

1. Go to Docker hub and find the tomcat container. Verify the supported tomcat installations on the same web page in Docker hub:



2. Verify the images with Docker images command and then try to run tomcat image. It will take some time.
3. Once image is pulled completely, container will be created and bash shell will be available for command execution:

```
[root@localhost Desktop]# docker images
REPOSITORY          TAG                 IMAGE ID            CREATED             VIRTUAL SIZE
centos              latest              2a332da70fd1        2 weeks ago         196.7 MB
ubuntu              latest              686477c12982        6 weeks ago         120.7 MB
hello-world         latest              f1d956dc5945        7 weeks ago         967 B
[root@localhost Desktop]# docker run -it tomcat bash
Unable to find image 'tomcat:latest' locally
latest: Pulling from tomcat

7d7852532044: Downloading [====================>                              ] 20.97 MB/51.35 MB
435cb21051b6: Download complete
4c76b3c13563: Download complete
35e170305690: Download complete
14fa7ed0654b: Download complete
02dec3806bda: Download complete
b50599b96e33: Download complete
ec7e4967fab4: Download complete
499b5c54f1ed: Download complete
cc5b39d4a8b7: Downloading [===========>                                       ] 18.37 MB/77.64 MB
290876b830ae: Download complete
30167fbc73d4: Download complete
3a80d45737ff: Download complete
d4c89486429f: Download complete
4513ebd4451d: Download complete
4d3f030833b5: Download complete
9b29824628e2: Download complete
91fa6d6b4e7a: Download complete
aa3cd4ef3986: Download complete
1e96877e40eb: Download complete
fa9f8e22fb74: Download complete
```

4. Let's try to install tomcat 8.0 and we will notice that image will be pulled from Docker hub. However, most of the parts are already available locally:

```
[root@localhost Desktop]# docker run -it --rm tomcat:8.0
Unable to find image 'tomcat:8.0' locally
8.0: Pulling from tomcat
7d7852532044: Already exists
435cb21051b6: Already exists
4c76b3c13563: Already exists
35e170305690: Already exists
14fa7ed0654b: Already exists
02dec3806bda: Already exists
b50599b96e33: Already exists
ec7e4967fab4: Already exists
499b5c54f1ed: Already exists
cc5b39d4a8b7: Already exists
290876b830ae: Already exists
30167fbc73d4: Already exists
3a80d45737ff: Already exists
d4c89486429f: Already exists
4513ebd4451d: Already exists
4d3f030833b5: Already exists
9b29824628e2: Already exists
91fa6d6b4e7a: Already exists
aa3cd4ef3986: Already exists
1e96877e40eb: Already exists
fa9f8e22fb74: Already exists
1f2d29d5c90e: Already exists
```

```
56fec8c9f483: Already exists
7245ac6b1b71: Already exists
5d4577339b14: Already exists
Digest: sha256:2af935d02022b22717e41768dc523a62d4c78106997ff467d652a506b70bc860

Status: Downloaded newer image for tomcat:8.0

Using CATALINA_BASE:   /usr/local/tomcat
Using CATALINA_HOME:   /usr/local/tomcat
Using CATALINA_TMPDIR: /usr/local/tomcat/temp
Using JRE_HOME:        /usr/lib/jvm/java-7-openjdk-amd64/jre
Using CLASSPATH:       /usr/local/tomcat/bin/bootstrap.jar:/usr/local/tomcat/bin/tomcat-juli.jar
19-Jun-2016 10:54:03.230 INFO [main] org.apache.catalina.startup.VersionLoggerListener.log
Server version:        Apache Tomcat/8.0.36
19-Jun-2016 10:54:03.233 INFO [main] org.apache.catalina.startup.VersionLoggerListener.log
Server built:          Jun 9 2016 13:55:50 UTC
19-Jun-2016 10:54:03.233 INFO [main] org.apache.catalina.startup.VersionLoggerListener.log
Server number:         8.0.36.0
19-Jun-2016 10:54:03.234 INFO [main] org.apache.catalina.startup.VersionLoggerListener.log
OS Name:               Linux
19-Jun-2016 10:54:03.234 INFO [main] org.apache.catalina.startup.VersionLoggerListener.log
OS Version:            2.6.32-573.26.1.el6.x86_64
19-Jun-2016 10:54:03.234 INFO [main] org.apache.catalina.startup.VersionLoggerListener.log
Architecture:          amd64
19-Jun-2016 10:54:03.235 INFO [main] org.apache.catalina.startup.VersionLoggerListener.log
Java Home:             /usr/lib/jvm/java-7-openjdk-amd64/jre
19-Jun-2016 10:54:03.235 INFO [main] org.apache.catalina.startup.VersionLoggerListener.log
JVM Version:           1.7.0_101-b00
19-Jun-2016 10:54:03.236 INFO [main] org.apache.catalina.startup.VersionLoggerListener.log
JVM Vendor:            Oracle Corporation
19-Jun-2016 10:54:03.236 INFO [main] org.apache.catalina.startup.VersionLoggerListener.log
CATALINA_BASE:         /usr/local/tomcat
19-Jun-2016 10:54:03.236 INFO [main] org.apache.catalina.startup.VersionLoggerListener.log
CATALINA_HOME:         /usr/local/tomcat
19-Jun-2016 10:54:03.238 INFO [main] org.apache.catalina.startup.VersionLoggerListener.log
Command line argument: -
Djava.util.logging.config.file=/usr/local/tomcat/conf/logging.properties
19-Jun-2016 10:54:03.238 INFO [main] org.apache.catalina.startup.VersionLoggerListener.log
Command line argument: -Djava.util.logging.manager=org.apache.juli.ClassLoaderLogManager
19-Jun-2016 10:54:03.238 INFO [main] org.apache.catalina.startup.VersionLoggerListener.log
Command line argument: -Djdk.tls.ephemeralDHKeySize=2048
19-Jun-2016 10:54:03.239 INFO [main] org.apache.catalina.startup.VersionLoggerListener.log
Command line argument: -Djava.endorsed.dirs=/usr/local/tomcat/endorsed
19-Jun-2016 10:54:03.239 INFO [main] org.apache.catalina.startup.VersionLoggerListener.log
Command line argument: -Dcatalina.base=/usr/local/tomcat
19-Jun-2016 10:54:03.240 INFO [main] org.apache.catalina.startup.VersionLoggerListener.log
Command line argument: -Dcatalina.home=/usr/local/tomcat
19-Jun-2016 10:54:03.240 INFO [main] org.apache.catalina.startup.VersionLoggerListener.log
```

```
Command line argument: -Djava.io.tmpdir=/usr/local/tomcat/temp
19-Jun-2016 10:54:03.241 INFO [main]
org.apache.catalina.core.AprLifecycleListener.lifecycleEvent Loaded APR based Apache
Tomcat Native library 1.2.7 using APR version 1.5.1.
19-Jun-2016 10:54:03.241 INFO [main]
org.apache.catalina.core.AprLifecycleListener.lifecycleEvent APR capabilities: IPv6 [true],
sendfile [true], accept filters [false], random [true].
19-Jun-2016 10:54:03.258 INFO [main]
org.apache.catalina.core.AprLifecycleListener.initializeSSL OpenSSL successfully initialized
(OpenSSL 1.0.2h  3 May 2016)
19-Jun-2016 10:54:03.408 INFO [main] org.apache.coyote.AbstractProtocol.init Initializing
ProtocolHandler ["http-apr-8080"]
19-Jun-2016 10:54:03.446 INFO [main] org.apache.coyote.AbstractProtocol.init Initializing
ProtocolHandler ["ajp-apr-8009"]
19-Jun-2016 10:54:03.453 INFO [main] org.apache.catalina.startup.Catalina.load Initialization
processed in 822 ms
19-Jun-2016 10:54:03.520 INFO [main] org.apache.catalina.core.StandardService.startInternal
Starting service Catalina
19-Jun-2016 10:54:03.520 INFO [main] org.apache.catalina.core.StandardEngine.startInternal
Starting Servlet Engine: Apache Tomcat/8.0.36
19-Jun-2016 10:54:03.533 INFO [localhost-startStop-1]
org.apache.catalina.startup.HostConfig.deployDirectory Deploying web application directory
/usr/local/tomcat/webapps/examples
19-Jun-2016 10:54:04.649 INFO [localhost-startStop-1]
org.apache.catalina.startup.HostConfig.deployDirectory Deployment of web application
directory /usr/local/tomcat/webapps/examples has finished in 1,115 ms
19-Jun-2016 10:54:04.649 INFO [localhost-startStop-1]
org.apache.catalina.startup.HostConfig.deployDirectory Deploying web application directory
/usr/local/tomcat/webapps/host-manager
19-Jun-2016 10:54:04.684 INFO [localhost-startStop-1]
org.apache.catalina.startup.HostConfig.deployDirectory Deployment of web application
directory /usr/local/tomcat/webapps/host-manager has finished in 34 ms
19-Jun-2016 10:54:04.684 INFO [localhost-startStop-1]
org.apache.catalina.startup.HostConfig.deployDirectory Deploying web application directory
/usr/local/tomcat/webapps/docs
19-Jun-2016 10:54:04.709 INFO [localhost-startStop-1]
org.apache.catalina.startup.HostConfig.deployDirectory Deployment of web application
directory /usr/local/tomcat/webapps/docs has finished in 25 ms
19-Jun-2016 10:54:04.709 INFO [localhost-startStop-1]
org.apache.catalina.startup.HostConfig.deployDirectory Deploying web application directory
/usr/local/tomcat/webapps/ROOT
19-Jun-2016 10:54:04.739 INFO [localhost-startStop-1]
org.apache.catalina.startup.HostConfig.deployDirectory Deployment of web application
directory /usr/local/tomcat/webapps/ROOT has finished in 30 ms
19-Jun-2016 10:54:04.739 INFO [localhost-startStop-1]
org.apache.catalina.startup.HostConfig.deployDirectory Deploying web application directory
/usr/local/tomcat/webapps/manager
19-Jun-2016 10:54:04.801 INFO [localhost-startStop-1]
```

```
org.apache.catalina.startup.HostConfig.deployDirectory Deployment of web application
directory /usr/local/tomcat/webapps/manager has finished in 61 ms
19-Jun-2016 10:54:04.817 INFO [main] org.apache.coyote.AbstractProtocol.start Starting
ProtocolHandler ["http-apr-8080"]
19-Jun-2016 10:54:04.828 INFO [main] org.apache.coyote.AbstractProtocol.start Starting
ProtocolHandler ["ajp-apr-8009"]
19-Jun-2016 10:54:04.830 INFO [main] org.apache.catalina.startup.Catalina.start Server startup
in 1376 ms
19-Jun-2016 12:05:22.546 INFO [Thread-3] org.apache.coyote.AbstractProtocol.pause Pausing
ProtocolHandler ["http-apr-8080"]
19-Jun-2016 12:05:22.580 INFO [Thread-3] org.apache.coyote.AbstractProtocol.pause Pausing
ProtocolHandler ["ajp-apr-8009"]
19-Jun-2016 12:05:22.582 INFO [Thread-3]
org.apache.catalina.core.StandardService.stopInternal Stopping service Catalina
19-Jun-2016 12:05:22.626 INFO [Thread-3] org.apache.coyote.AbstractProtocol.stop Stopping
ProtocolHandler ["http-apr-8080"]
19-Jun-2016 12:05:22.688 INFO [Thread-3] org.apache.coyote.AbstractProtocol.stop Stopping
ProtocolHandler ["ajp-apr-8009"]
19-Jun-2016 12:05:22.743 INFO [Thread-3] org.apache.coyote.AbstractProtocol.destroy
Destroying ProtocolHandler ["http-apr-8080"]
19-Jun-2016 12:05:22.745 INFO [Thread-3] org.apache.coyote.AbstractProtocol.destroy
Destroying ProtocolHandler ["ajp-apr-8009"]
You have new mail in /var/spool/mail/root
```

5. Container is created successfully, verify existing containers by using `docker ps` command:

```
[root@localhost Desktop]# docker ps
CONTAINER ID        IMAGE               COMMAND             CREATED
STATUS              PORTS               NAMES
c3fbd72a1b35        tomcat:8.0          "catalina.sh run"   29 minutes ago
Up 29 minutes       8080/tcp            sad_pasteur
```

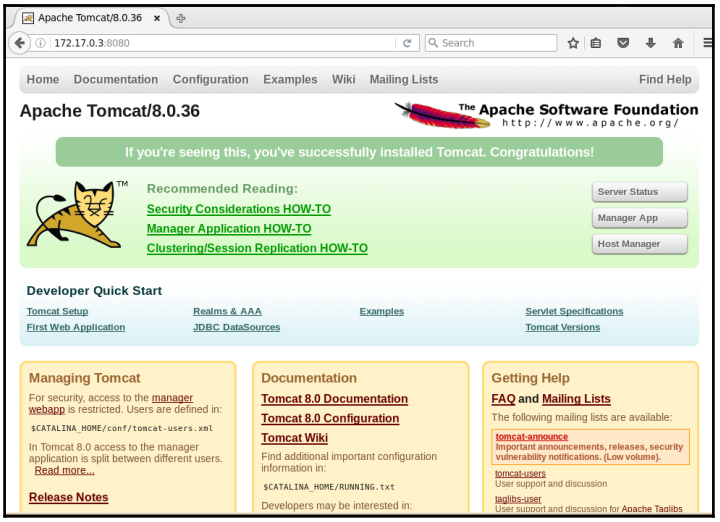
Once we have tomcat container ready, let's try to find out it's IP address so we can access the Tomcat using it.

Use `docker inspect` command with container id to find out the IP address of the container:

```
[root@localhost Desktop]# docker inspect c3fbd72a1b35
[
{
    "Id": "c3fbd72a1b35c6725606df726b5651cbd774b02d55bad6352c0e5205894b8b56",
    "Created": "2016-06-19T10:54:01.330825881Z",
    "Path": "catalina.sh",
    "Args": [
        "run"
    ],
    "State": {
        "Running": true,
        "Paused": false,
        "Restarting": false,
        "OOMKilled": false,
        "Dead": false,
        "Pid": 6293,
        "ExitCode": 0,
        "Error": "",
        "StartedAt": "2016-06-19T10:54:02.250775469Z",
        "FinishedAt": "0001-01-01T00:00:00Z"
    },
    "Image": "5d4577339b146f4e71ddb267812213bdc1a612eeb48a5f3c95f105b7894a4a73",
    "NetworkSettings": {
        "Bridge": "",
        "EndpointID": "a88792ad6a30316dbf8ad50c565d2c2c5951a040f4909f97418405142c7224e8",
        "Gateway": "172.17.42.1",
        "GlobalIPv6Address": "",
        "GlobalIPv6PrefixLen": 0,
        "HairpinMode": false,
        "IPAddress": "172.17.0.3",
```

Docker networking is a different concept itself and it is not in the scope of this book so we are not going to cover it.

However, let's verify whether the tomcat container is running properly or not:



So finally, we are able to run Tomcat container. In next section we will try to cover some basic but useful commands and try to build an image.

Managing Containers

Let's try to run tomcat container as background process. It is best practice to run Docker container as a background process to avoid stopping containers accidently from terminal:

1. Use -d parameter:

```
[root@localhost Desktop]# docker run -d tomcat
68c6d1f7bc631613813ffb761cc833156a70e2063c2a743dd2729fe73b2873f9
```

2. Verify the container that is created recently:

```
[root@localhost Desktop]# docker ps
CONTAINER ID        IMAGE               COMMAND             CREATED
 STATUS              PORTS               NAMES
68c6d1f7bc63        tomcat              "catalina.sh run"   15 seconds ago
 Up 11 seconds       8080/tcp            desperate_hypatia
You have new mail in /var/spool/mail/root
```

3. Get the IP address of the container with docker inspect command and providing container id:

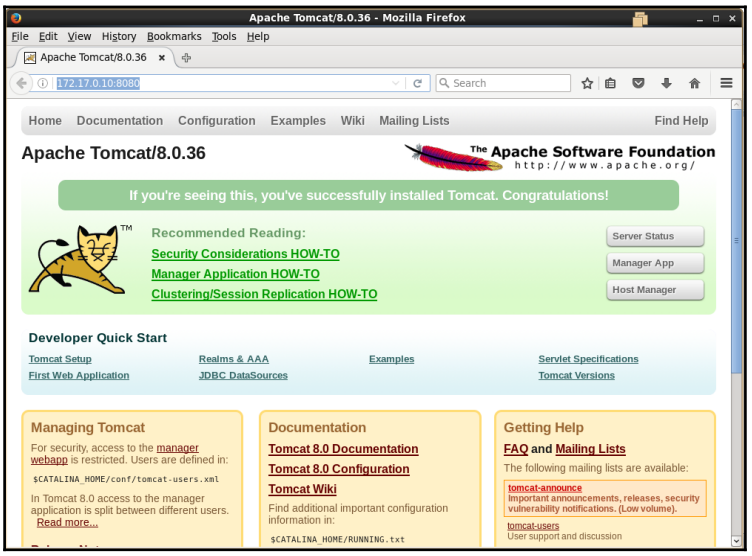
```
[root@localhost Desktop]# docker inspect 68c6d1f7bc63
[
{
    "Id": "68c6d1f7bc631613813ffb761cc833156a70e2063c2a743dd2729fe73b2873f9",
    "Created": "2016-06-21T18:25:20.73708668Z",
    "Path": "catalina.sh",
    "Args": [
        "run"
    ],
    "State": {
        "Running": true,
        "Paused": false,
        "Restarting": false,
        "OOMKilled": false,
        "Dead": false,
        "Pid": 20448,
        "ExitCode": 0,
        "Error": "",
        "StartedAt": "2016-06-21T18:25:23.086757711Z",
        "FinishedAt": "0001-01-01T00:00:00Z"
    },
    "Image": "5d4577339b146f4e71ddb267812213bdc1a612eeb48a5f3c95f105b7894a4a73",
    "NetworkSettings": {
        "Bridge": "",
        "EndpointID": "7ef4f440a137222ad96c20bd53330875ec8192499419f8d5d9c9a337c6044f9f",
```

```
        "Gateway": "172.17.42.1",
        "GlobalIPv6Address": "",
        "GlobalIPv6PrefixLen": 0,
        "HairpinMode": false,
        "IPAddress": "172.17.0.10",
        "IPPrefixLen": 16,
        "IPv6Gateway": "",
        "LinkLocalIPv6Address": "",
        "LinkLocalIPv6PrefixLen": 0,
        "MacAddress": "02:42:ac:11:00:0a",
        "NetworkID": "c5d8d33430092901b8f643f96f9d0fee2d70b45db782bd405a10a38b8cb12447",
        "PortMapping": null,
        "Ports": {
           "8080/tcp": null
        },
        "SandboxKey": "/var/run/docker/netns/68c6d1f7bc63",
        "SecondaryIPAddresses": null,
        "SecondaryIPv6Addresses": null
    },
    "ResolvConfPath":
    "/var/lib/docker/containers/68c6d1f7bc631613813ffb761cc833156a70e2063c2a743
    dd2729fe73b2873f9/resolv.conf",
    "HostnamePath": "/var/lib/docker/containers/68c6d1f7bc631613813ffb761cc833156a70e2063c
     2a743dd2729fe73b2873f9/hostname",
    "HostsPath": "/var/lib/docker/containers/68c6d1f7bc631613813ffb761cc833156a70e2063
     c2a743dd2729fe73b2873f9/hosts",
    "LogPath": "/var/lib/docker/containers/68c6d1f7bc631613813ffb761cc833156a70e2063c2
a743dd2729fe73b2873f9/68c6d1f7bc631613813ffb761cc833156a70e2063c2a743dd2729fe73b287
3f9-
     json.log",
    "Name": "/desperate_hypatia",
    "RestartCount": 0,
    "Driver": "devicemapper",
    "ExecDriver": "native-0.2",
    "MountLabel": "",
    "ProcessLabel": "",
    "Volumes": {},
    "VolumesRW": {},
    "AppArmorProfile": "",
    "ExecIDs": null,
    "HostConfig": {
        "Binds": null,
        "ContainerIDFile": "",
        "LxcConf": [],
        "Memory": 0,
        "MemorySwap": 0,
        "CpuShares": 0,
        "CpuPeriod": 0,
```

```
            "CpusetCpus": "",
            "CpusetMems": "",
            "CpuQuota": 0,
            "BlkioWeight": 0,
            "OomKillDisable": false,
            "Privileged": false,
            "PortBindings": {},
            "Links": null,
            "PublishAllPorts": false,
            "Dns": null,
            "DnsSearch": null,
            "ExtraHosts": null,
            "VolumesFrom": null,
            "Devices": [],
            "NetworkMode": "bridge",
            "IpcMode": "",
            "PidMode": "",
            "UTSMode": "",
            "CapAdd": null,
            "CapDrop": null,
            "RestartPolicy": {
                "Name": "no",
                "MaximumRetryCount": 0
            },
            "SecurityOpt": null,
            "ReadonlyRootfs": false,
            "Ulimits": null,
            "LogConfig": {
                "Type": "json-file",
                "Config": {}
            },
            "CgroupParent": ""
        },
        "Config": {
            "Hostname": "68c6d1f7bc63",
            "Domainname": "",
            "User": "",
            "AttachStdin": false,
            "AttachStdout": false,
            "AttachStderr": false,
            "PortSpecs": null,
            "ExposedPorts": {
                "8080/tcp": {}
            },
            "Tty": false,
            "OpenStdin": false,
            "StdinOnce": false,
            "Env": [
```

```
            "PATH=/usr/local/tomcat/bin:/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin",
            "LANG=C.UTF-8",
            "JAVA_HOME=/usr/lib/jvm/java-7-openjdk-amd64/jre",
            "JAVA_VERSION=7u101",
            "JAVA_DEBIAN_VERSION=7u101-2.6.6-2~deb8u1",
            "CATALINA_HOME=/usr/local/tomcat",
            "OPENSSL_VERSION=1.0.2h-1",
            "TOMCAT_MAJOR=8",
            "TOMCAT_VERSION=8.0.36",
            "TOMCAT_TGZ_URL=https://www.apache.org/dist/tomcat/tomcat-8/v8.0.36/bin/apache-
             tomcat-8.0.36.tar.gz"
        ],
        "Cmd": [
            "catalina.sh",
            "run"
        ],
        "Image": "tomcat",
        "Volumes": null,
        "VolumeDriver": "",
        "WorkingDir": "/usr/local/tomcat",
        "Entrypoint": null,
        "NetworkDisabled": false,
        "MacAddress": "",
        "OnBuild": null,
        "Labels": {}
    }
}
]
```

4. Note the IP address: http://172.17.0.10:8080/ and try to access it in the browser:



Obvious question will be, how to stop containers, right? To get details of running containers, use command docker ps:

Observer last column that is Names and we can see some strange name desperate_hypatia that is automatically allocated to a container if it is not given explicitly;

```
[root@localhost Desktop]# docker ps
  CONTAINER ID       IMAGE              COMMAND           CREATED
STATUS             PORTS              NAMES
  68c6d1f7bc63       tomcat               "catalina.sh run"   15 minutes ago
Up 15 minutes      8080/tcp           desperate_hypatia
```

5. Let's stop the container using container name that is automatically assigned.

```
[root@localhost Desktop]# docker stop desperate_hypatia
desperate_hypatia
```

6. If we want to give custom name to the container, then we can give it by using --name operator as shown below:

```
[root@localhost Desktop]# docker run -d --name devops_tomcat tomcat
cf2c1d19070fab73b840f94009391ad211f010044a7763fe201a115b0bc6a4b8
You have new mail in /var/spool/mail/root
[root@localhost Desktop]# docker ps
  CONTAINER ID       IMAGE              COMMAND           CREATED
```

```
STATUS          PORTS           NAMES
 cf2c1d19070f      tomcat          "catalina.sh run"   10 seconds ago
Up 9 seconds     8080/tcp        devops_tomcat
```

7. Can we see the list of all containers which are stopped? Yes, we can. Use `docker ps -a` command as shown below to get the list of stopped containers:

```
[root@localhost Desktop]# docker ps -a
CONTAINER ID      IMAGE           COMMAND          CREATED          STATUS
PORTS           NAMES
68c6d1f7bc63      tomcat          "catalina.sh run"   16 minutes ago      Exited (143) 47
seconds ago                desperate_hypatia
51e055a3414b      ubuntu          "ls -l"          43 minutes ago      Exited (0) 43 minutes ago
sick_meitner
a6f402e7a2a8      ubuntu          "ls"             43 minutes ago      Exited (0) 43 minutes ago
naughty_hopper
a4699613f112      ubuntu          "bash"           47 minutes ago      Exited (127) 46 minutes
ago               backstabbing_bardeen
66a04d9137d8      ubuntu          "/bin/bash"      47 minutes ago      Exited (0) 47 minutes
ago               hungry_mcclintock
a27b460778e6      ubuntu          "pwd"            48 minutes ago      Exited (0) 48 minutes
ago               dreamy_yonath
You have new mail in /var/spool/mail/root
```

Container's life time is limited to the existence of parent process.

```
[root@localhost Desktop]# docker run -p 8080:9090 -d --name devops_tomcat9
 tomcat
0f8c251929b2f316bac1d53c5b8d03a155d790dada1ce2fcf94f95844a3acfef
```

8. To get the access of terminal of the container, use below command after creation of the container:

```
[root@localhost Desktop]# docker exec -it devops_tomcat9 bash
```

9. Once we have an access to console of the container, verify IP address via `ip addr show eth0` command:

```
root@0f8c251929b2:/usr/local/tomcat# ip addr show eth0
57: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue state UP
    link/ether 02:42:ac:11:00:14 brd ff:ff:ff:ff:ff:ff
    inet 172.17.0.20/16 scope global eth0
    inet6 fe80::42:acff:fe11:14/64 scope link
      valid_lft forever preferred_lft forever
```

```
root@0f8c251929b2:/usr/local/tomcat# ip route
172.17.0.0/16 dev eth0  proto kernel  scope link  src 172.17.0.20
default via 172.17.42.1 dev eth0
root@0f8c251929b2:/usr/local/tomcat#
```

10. Now, let's try to search Docker images available in the Docker hub. Try docker search command to find tomcat images available in Docker hub:

```
[root@localhost Desktop]# docker search tomcat
NAME      DESCRIPTION                                     STARS     OFFICIAL   AUTOMATED
tomcat                Apache Tomcat is an open source implementa...   750          [OK]
dordoka/tomcat            Ubuntu 14.04, Oracle JDK 8 and Tomcat 8 ba...   19       [OK]
consol/tomcat-7.0         Tomcat 7.0.57, 8080, "admin/admin"              16           [OK]
consol/tomcat-8.0         Tomcat 8.0.15, 8080, "admin/admin"              14           [OK]
cloudesire/tomcat         Tomcat server, 6/7/8                            8            [OK]
davidcaste/alpine-tomcat  Apache Tomcat 7/8 using Oracle Java 7/8 wi...   7        [OK]
andreptb/tomcat           Debian Jessie based image with Apache Tomc...   4        [OK]
fbrx/tomcat               Minimal Tomcat image based on Alpine Linux      2        [OK]
openweb/oracle-tomcat     A fork off of Official tomcat image with O...   2        [OK]
kieker/tomcat                                                             2        [OK]
dreaminsun/tomcat         optimized tomcat                                1            [OK]
chrisipa/tomcat           Tomcat docker image based on Debian Jessie...   1            [OK]
abzcoding/tomcat-redis    a tomcat container with redis as session m...   1            [OK]
cirit/tomcat              Tomcat Docker Image with collectd               1            [OK]
ericogr/tomcat            Tomcat 8, 8080, "docker/docker"                 1            [OK]
jtech/tomcat              Latest Tomcat production distribution on l...   1            [OK]
nicescale/tomcat          Tomcat service for NiceScale. http://nices...   1            [OK]
mccoder/tomcat            Tomcat with APR                                 0            [OK]
foobot/tomcat                                                             0            [OK]
bitnami/tomcat            Bitnami Tomcat Docker Image                     0            [OK]
stakater/tomcat           Tomcat based on Ubuntu 14.04 and Oracle Java    0        [OK]
tb4mmaggots/tomcat        Apache Tomcat micro container                   0            [OK]
cheewai/tomcat            Tomcat and Oracle JRE in docker                 0            [OK]
inspectit/tomcat          Tomcat with inspectIT                           0            [OK]
davidcaste/debian-tomcat  Yet another Debian Docker image for Tomcat...   0    [OK]
```

11. Let's verify the existing images again:

```
[root@localhost Desktop]# docker images
REPOSITORY   TAG    IMAGE ID      CREATED        VIRTUAL SIZE
tomcat       8.0    5d4577339b14  7 days ago         359.2 MB
tomcat       latest 5d4577339b14  7 days ago         359.2 MB
centos       latest 2a332da70fd1  2 weeks ago        196.7 MB
ubuntu       latest 686477c12982  7 weeks ago        120.7 MB
hello-world  latest f1d956dc5945  8 weeks ago          967 B
You have new mail in /var/spool/mail/root
```

12. Our next step is to create a sample image file. We can build Docker image using Dockerfile. It provides step by step instructions to build images.

Let's try with simple CentOS image:

1. Dockerfile contains following two lines:

```
FROM centos
MAINTAINER mitesh <mitesh.soxxxxxx@xxxxxxxx.com>
```

2. Go to the same directory in terminal and use docker build . to build an image:

```
[root@localhost Desktop]# docker build .
Sending build context to Docker daemon 681.6 MB
Sending build context to Docker daemon
Step 0 : FROM centos
---> 2a332da70fd1
Step 1 : MAINTAINER mitesh < mitesh.soxxxxxx@xxxxxxxx.com >
---> Running in 305e8da05500
---> b636e26a333a
Removing intermediate container 305e8da05500
Successfully built b636e26a333a
You have new mail in /var/spool/mail/root
```

3. We have successfully build sample Docker image. Verify it:

```
[root@localhost Desktop]# docker images
REPOSITORY   TAG     IMAGE ID     CREATED         VIRTUAL SIZE
<none>      <none>   b636e26a333a  16 seconds ago     196.7 MB
tomcat       8.0     5d4577339b14  7 days ago         359.2 MB
tomcat      latest   5d4577339b14  7 days ago         359.2 MB
centos      latest   2a332da70fd1  2 weeks ago        196.7 MB
ubuntu      latest   686477c12982  7 weeks ago        120.7 MB
hello-world latest   f1d956dc5945  8 weeks ago          967 B
```

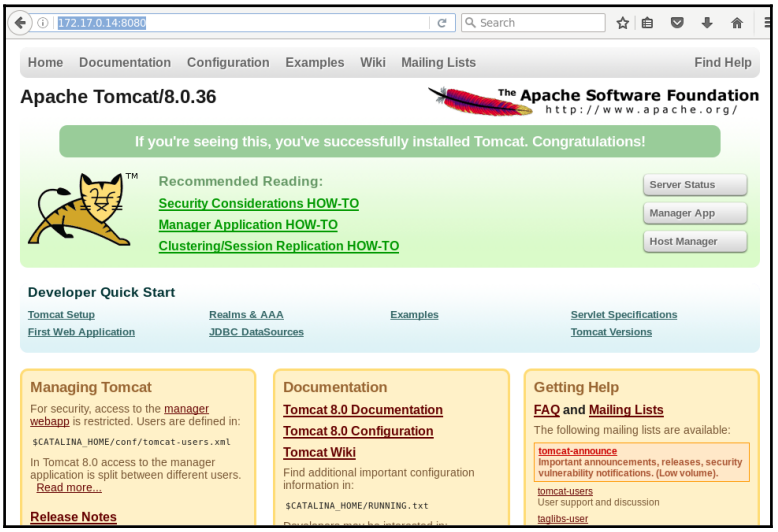
To run tomcat from the host IP address or from localhost:

```
[root@localhostmitesh]#docker run -p 8180:8080 -d --name
 devopstomcat1devopstomcatnew
b5f054ee4ac36d67279db10497fe7a780aecf2a72a7f52fa31ee80c618d98e4a
```

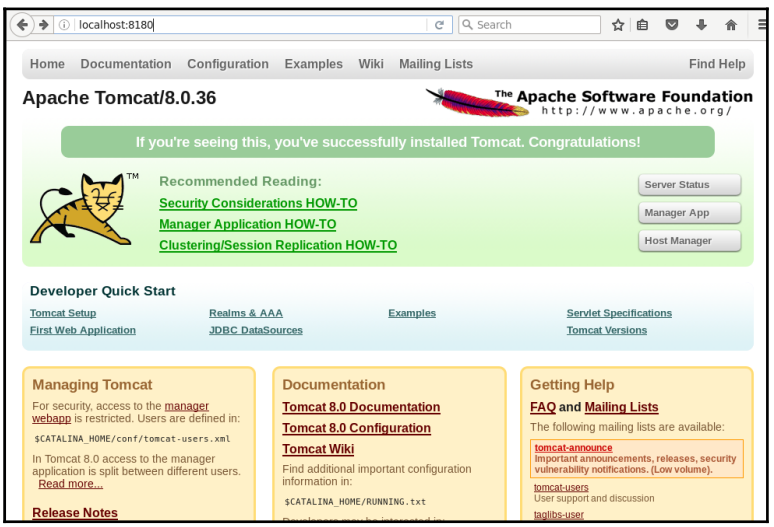
Here, 8081 is port for Host. Verify newly created container using dockerps command.

```
[root@localhostmitesh]#dockerps
CONTAINER ID       IMAGE              COMMAND           CREATED           STATUS
PORTS                  NAMES
b5f054ee4ac3   devopstomcatnew    "catalina.sh run"    21 seconds ago     Up 20 seconds
0.0.0.0:8180->8080/tcp    devopstomcat1
```

Use docker inspect command to get the IP address of the container. Browse http://172.17.0.14:8080/ from the Host Virtual machine:



Browse http://localhost:8180/ from the Host virtual machine; observe the Port number here.



To copy file from Container to Host virtual machine use dockercp command.

```
[root@localhostmitesh]#dockercp43f71c5d2ac0:/usr/local/tomcat/conf/tomcat-users.xml
/root/Desktop/
```

Here 43f71c5d2ac0 is a container ID followed by colon, source path on container and destination path on Host virtual machine.

Till now we have covered basics of Docker, it architecture, some basic operations and so on. This will essentially helps us while doing end to end orchestration as well as performing Docker related operations.

Self-Test Questions

1. State True or False: Docker has a Client Server architecture.
2. True
3. False

1. State True or False: Docker has two main components – Docker Host and Docker hub
2. True
3. False

1. State True or False: While creating a container, image has to be available locally else operation fails.
2. True
3. False

1. State True or False: Docker Hub is used to store and manage containers.
2. True
3. False

1. State True or False: Overhead of memory management and device drivers is extremely high in Docker containers
2. True
3. False

1. State True or False: For CentOS-6, Docker RPM package is called docker-io.
2. True
3. False

1. State True or False: docker ps -a command is used to see the list of stopped containers.
2. True
3. False

Summary

In this chapter, we have covered Overview of Docker Container, architecture details, details of main components of Docker including quick overview of Docker hub. Based on the overview, we tried to compare virtual machines with Docker containers to gain clear picture why containers are gaining traction in recent times.

After gaining some understanding on virtual machines and containers, we have covered process of Docker installation on CentOS 6.x virtual machine. We created `hello-world` container, ubuntu and CentOS containers from the images available in Docker hub.

Our main aim is to use tomcat container for deploying sample spring application so we used tomcat image and created container from it for verification. To gain more understanding, we used Dockerfile to build an image with Java and Tomcat.

In the context of container, Ted Engstrom's below quote is quite suitable:

> *"Anything that is wasted effort represents wasted time. The best management of our time thus becomes linked inseparably with the best utilization of our efforts."*
> *- Ted Engstrom*

In the next chapter, we will see how to create a virtual machine in Amazon Web Services and Microsoft Azure using Chef and how to setup runtime environment using Chef configuration management tool.

6
Cloud Provisioning and Configuration Management with Chef

"You may delay, but time will not."
- Benjamin Franklin

Let's revisit what we have covered till now and what was our goal in the first chapter. Our main objective is to create end to end automated pipeline for application deployment. We considered source code repositories, build tools, continuous integration, configuration management to setup runtime environment, resource provisioning in cloud and containers, continuous delivery, continuous deployment, continuous monitoring, continuous feedback, continuous improvement, and continuous innovation. We want to use end to end pipeline for sample spring application petclinic. In Chapter 4, *Installing and Configuring Chef* and Chapter 5, *Installing and Configuring Docker* we have covered Chef configuration management tool and Docker containers in brief manner. Both are the topic for a book in its own self. Now we are at the stage where we understand basics of configuration management and containers so we can go for resource provisioning in Cloud environment using Chef and install runtime environment required to run Petclinic. In this scenario, it will be an installation of Java and Tomcat.

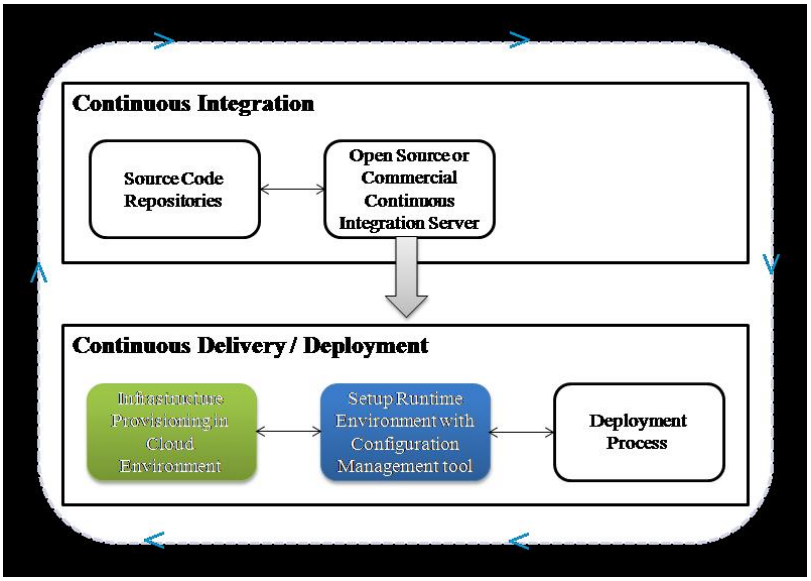
This chapter describes in detail how to install knife plugins that are used to manage cloud resources using Chef. It will cover creating instances in the AWS and Azure with the use of knife-EC2 and knife-azure plugins. It will also cover how Chef is used to manage Docker containers.

In this chapter, we will explore the following topics:

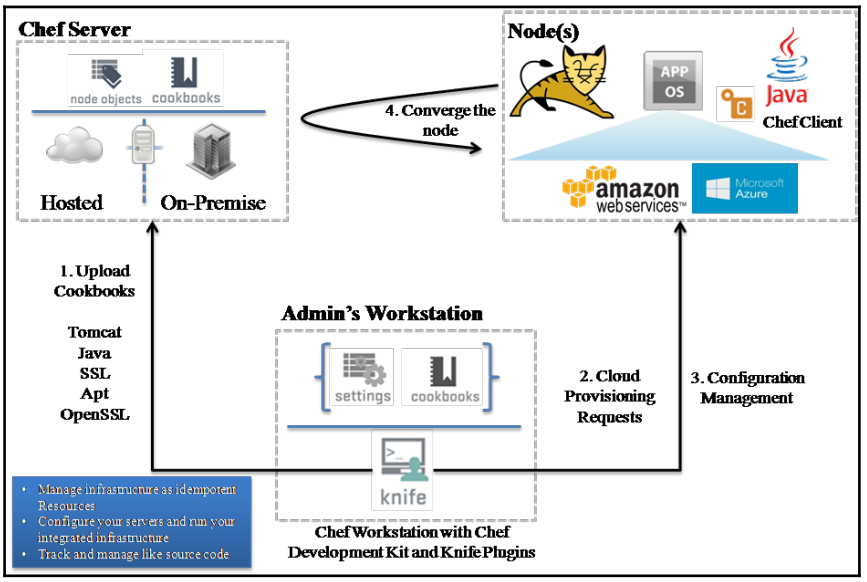
- Chef and Cloud Provisioning
- Installing Knife Plugins for Amazon EC2 and Microsoft Azure
- Creating and Configuring Virtual Machine in Amazon Web Services
- Creating and Configuring Virtual Machine in Microsoft Azure
- Manage Docker containers with Chef

Chef and Cloud Provisioning

Chef is not only used for setting up runtime environment or configuration management but it is used for resource provisioning in cloud environment. It supports Cloud service providers such as Microsoft Azure, Amazon Web Services, VMware, OpenStack, HP Cloud, Google Compute Engine and so on. Chef provides more flexibilities to the concept of infrastructure as a code and brings configuration management also into picture. Knife plugins are used to manage or use different Cloud service providers. With knife plugins, it is easier to provision and de-provision resources along with controlled and centralized configuration management.



We will specifically focus on Infrastructure Provisioning in Cloud Environment and Setup Runtime Environment with Configuration Management tool.



We will provision resources in public cloud environment using knife plugins with the use of Chef workstation. We have configured Chef workstation in Chapter 4, *Installing and Configuring Chef*. From Chef workstation, we can execute knife commands to create instances (Chef Node) in different cloud environments. In our case, we will provision resources in Amazon EC2 and Microsoft Azure:

1. Chef Workstation to CSP: Create new instance in your Cloud environment
2. CSP: Ok ... Done! New instance is up and Running. (Chef Node is available)
3. Chef Node to Chef Server: Hello!
4. Chef Server to Chef Node: Here is your task... Download Chef Client
5. Chef Server <-> Chef Node: A secure handshake; Chef server generates a security certificate. Security certificate is used to authenticate the new node's upcoming requests
6. Chef Server to Chef Node: Here is your list of recipes that you need to install.
7. Chef Node to Chef Server: Thank you, I am updated!

Some of the major benefits we get through Chef configuration management tool's usage with different Cloud platforms are:

- Easy policy enforcement with centralized control
- Enable setup of consistent runtime environment

- Build Repeatable Infrastructure to avoid manual effort and errors
- Enable rapid deployment of new applications
- Enable easy restoration of environments
- Enable disaster recovery and business continuity
- Community-based cookbooks and recipes
- Faster time to market to remain in Competition
- Supports major Cloud service providers through Plugins

In the next section, we will install knife plugins for some popular cloud platforms.

Installing Knife Plugins for Amazon Web Services and Microsoft Azure

Chef can be used to automate AWS services with the use of knife plugins. Knife EC2 is Chef knife plugin for Amazon EC2 that allows us to create and manage instances in the Amazon EC2.

For more details on the Knife EC2 plugin visit at:https://github.com/chef/knife-ec2.

Documentation for Knife EC2 plugin is available at: https://github.com/chef/knife-ec2/blob/master/README.md.

We can configure Amazon EC2 credentials for knife-EC2 in knife.rb file using knife[:aws_access_key_id] and knife[:aws_secret_access_key] as shown below:

```
knife[:aws_access_key_id] = "Your AWS Access Key ID"
knife[:aws_secret_access_key] = "Your AWS Secret Access Key"
```

1. Let's verify whether ruby is installed or not. If not then we need to install it along with gems for installing knife plugins:

   ```
   knife-ec2[root@devops1 Desktop]# ruby -v
   ruby 1.8.7 (2013-06-27 patchlevel 374) [x86_64-linux]
   ```

2. The version ruby 1.8.7 is old so we need to install ruby > 2.0. Verify the Chef client is installed or not? As this is a workstation we installed and configured, Chef version will be available.

   ```
   [root@devops1 Desktop]# knife -v
   ```

```
Chef: 12.9.41
```

3. Install RVM using \curl -sSL https://get.rvm.io | bash. It allows to install and manage multiple environments in simple manner.

```
[root@devops1 Desktop]# \curl -sSL https://get.rvm.io | bash
Downloading https://github.com/rvm/rvm/archive/master.tar.gz
Creating group 'rvm'
Installing RVM to /usr/local/rvm/
stat: cannot stat `/<gconf>/ a<entry name='login_shell' mtime='1463163726'
 type='bool' value='true'/>': No such file or directory
stat: cannot stat `/<gconf>/ a<entry name='login_shell' mtime='1463163726'
 type='bool' value='true'/>': No such file or directory
Installation of RVM in /usr/local/rvm/ is almost complete:
  * First you need to add all users that will be using rvm to 'rvm' group,
    and logout - login again, anyone using rvm will be operating with `umask
    u=rwx,g=rwx,o=rx`.
  * To start using RVM you need to run `source /etc/profile.d/rvm.sh`
    in all your open shell windows, in rare cases you need to reopen all shell
    windows.
# Administrator,
#
#   Thank you for using RVM!
#   We sincerely hope that RVM helps to make your life easier and more
      enjoyable!!!
#
# ~Wayne, Michal & team.
```

In case of problems, visit at:
https://rvm.io/help and https://twitter.com/rvm_io

4. Let's install additional Ruby dependencies:

```
[root@devops1 Desktop]#  yum install gcc g++ make automake autoconf curl-devel openssl-
 devel zlib-devel httpd-devel apr-devel apr-util-devel sqlite-devel
Loaded plugins: fastestmirror, refresh-packagekit, security
Setting up Install Process
Loading mirror speeds from cached hostfile
 * base: centos.excellmedia.net
 * extras: centos.excellmedia.net
 * updates: centos.excellmedia.net
Package gcc-4.4.7-16.el6.x86_64 already installed and latest version
No package g++ available.
Package 1:make-3.81-20.el6.x86_64 already installed and latest version
Package automake-1.11.1-4.el6.noarch already installed and latest version
```

```
Package autoconf-2.63-5.1.el6.noarch already installed and latest version
Package libcurl-devel-7.19.7-46.el6.x86_64 already installed and latest version
Package openssl-devel-1.0.1e-42.el6_7.4.x86_64 already installed and latest version
Package zlib-devel-1.2.3-29.el6.x86_64 already installed and latest version
Package sqlite-devel-3.6.20-1.el6_7.2.x86_64 already installed and latest version
Resolving Dependencies
--> Running transaction check
---> Package apr-devel.x86_64 0:1.3.9-5.el6_2 will be installed
---> Package apr-util-devel.x86_64 0:1.3.9-3.el6_0.1 will be installed
--> Processing Dependency: openldap-devel for package: apr-util-devel-1.3.9-
    3.el6_0.1.x86_64
--> Processing Dependency: db4-devel for package: apr-util-devel-1.3.9-3.el6_0.1.x86_64
---> Package httpd-devel.x86_64 0:2.2.15-47.el6.centos.4 will be installed
--> Running transaction check
---> Package db4-devel.x86_64 0:4.7.25-20.el6_7 will be installed
--> Processing Dependency: db4-cxx = 4.7.25-20.el6_7 for package: db4-devel-4.7.25-
    20.el6_7.x86_64
--> Processing Dependency: libdb_cxx-4.7.so()(64bit) for package: db4-devel-4.7.25-
    20.el6_7.x86_64
---> Package openldap-devel.x86_64 0:2.4.40-7.el6_7 will be installed
--> Processing Dependency: cyrus-sasl-devel >= 2.1 for package: openldap-devel-2.4.40-
    7.el6_7.x86_64
--> Running transaction check
---> Package cyrus-sasl-devel.x86_64 0:2.1.23-15.el6_6.2 will be installed
---> Package db4-cxx.x86_64 0:4.7.25-20.el6_7 will be installed
--> Finished Dependency Resolution
Dependencies Resolved
================================================================================
 Package            Arch       Version                   Repository     Size
================================================================================
Installing:
 apr-devel          x86_64     1.3.9-5.el6_2             base          176 k
 apr-util-devel     x86_64     1.3.9-3.el6_0.1           base           69 k
 httpd-devel        x86_64     2.2.15-47.el6.centos.4    updates       155 k
Installing for dependencies:
 cyrus-sasl-devel   x86_64     2.1.23-15.el6_6.2         base          303 k
 db4-cxx            x86_64     4.7.25-20.el6_7           updates       588 k
 db4-devel          x86_64     4.7.25-20.el6_7           updates       6.6 M
 openldap-devel     x86_64     2.4.40-7.el6_7            updates       1.1 M
Transaction Summary
================================================================================
Install       7 Package(s)
Total download size: 8.9 M
Installed size: 33 M
Is this ok [y/N]: y
Downloading Packages:
(1/7): apr-devel-1.3.9-5.el6_2.x86_64.rpm                | 176 kB     00:00
(2/7): apr-util-devel-1.3.9-3.el6_0.1.x86_64.rpm         |  69 kB     00:00
```

```
(3/7): cyrus-sasl-devel-2.1.23-15.el6_6.2.x86_64.rpm          | 303 kB     00:01
(4/7): db4-cxx-4.7.25-20.el6_7.x86_64.rpm                     | 588 kB     00:02
(5/7): db4-devel-4.7.25-20.el6_7.x86_64.rpm                   | 6.6 MB     00:32
(6/7): httpd-devel-2.2.15-47.el6.centos.4.x86_64.rpm          | 155 kB     00:00
(7/7): openldap-devel-2.4.40-7.el6_7.x86_64.rpm               | 1.1 MB     00:05
--------------------------------------------------------------------------------
Total                                       196 kB/s | 8.9 MB     00:46
Running rpm_check_debug
Running Transaction Test
Transaction Test Succeeded
Running Transaction
Warning: RPMDB altered outside of yum.
  Installing : apr-devel-1.3.9-5.el6_2.x86_64                              1/7
  Installing : db4-cxx-4.7.25-20.el6_7.x86_64                              2/7
  Installing : db4-devel-4.7.25-20.el6_7.x86_64                            3/7
  Installing : cyrus-sasl-devel-2.1.23-15.el6_6.2.x86_64                   4/7
  Installing : openldap-devel-2.4.40-7.el6_7.x86_64                        5/7
  Installing : apr-util-devel-1.3.9-3.el6_0.1.x86_64                       6/7
  Installing : httpd-devel-2.2.15-47.el6.centos.4.x86_64                   7/7
  Verifying  : db4-devel-4.7.25-20.el6_7.x86_64                            1/7
  Verifying  : apr-devel-1.3.9-5.el6_2.x86_64                              2/7
  Verifying  : httpd-devel-2.2.15-47.el6.centos.4.x86_64                   3/7
  Verifying  : openldap-devel-2.4.40-7.el6_7.x86_64                        4/7
  Verifying  : apr-util-devel-1.3.9-3.el6_0.1.x86_64                       5/7
  Verifying  : cyrus-sasl-devel-2.1.23-15.el6_6.2.x86_64                   6/7
  Verifying  : db4-cxx-4.7.25-20.el6_7.x86_64                              7/7
Installed:
  apr-devel.x86_64 0:1.3.9-5.el6_2              apr-util-devel.x86_64 0:1.3.9-3.el6_0.1
  httpd-devel.x86_64 0:2.2.15-47.el6.centos.4
Dependency Installed:
  cyrus-sasl-devel.x86_64 0:2.1.23-15.el6_6.2       db4-cxx.x86_64 0:4.7.25-20.el6_7
  db4-devel.x86_64 0:4.7.25-20.el6_7            openldap-devel.x86_64 0:2.4.40-7.el6_7
Complete!
```

5. We have successfully installed Ruby and its dependencies. Now let's install rubygems:

```
[root@devops1 Desktop]#  yum install rubygems
Loaded plugins: fastestmirror, refresh-packagekit, security
Setting up Install Process
Loading mirror speeds from cached hostfile
 * base: centos.excellmedia.net
 * extras: centos.excellmedia.net
 * updates: centos.excellmedia.net
Resolving Dependencies
--> Running transaction check
---> Package rubygems.noarch 0:1.3.7-5.el6 will be installed
--> Finished Dependency Resolution
```

```
  Dependencies Resolved
  ================================================================
   Package            Arch            Version              Repository        Size
  ================================================================
  Installing:
   rubygems           noarch          1.3.7-5.el6          base              207 k
  Transaction Summary
  ================================================================Install       1
Package(s)
  Total download size: 207 k
  Installed size: 713 k
  Is this ok [y/N]: y
  Downloading Packages:
  rubygems-1.3.7-5.el6.noarch.rpm                          | 207 kB     00:01
  Running rpm_check_debug
  Running Transaction Test
  Transaction Test Succeeded
  Running Transaction
    Installing : rubygems-1.3.7-5.el6.noarch                                1/1
    Verifying  : rubygems-1.3.7-5.el6.noarch                                1/1
  Installed:
    rubygems.noarch 0:1.3.7-5.el6
  Complete!
```

6. Update the gems:

```
[root@devops1 Desktop]# gem update
Updating installed gems
Updating fog
Fetching: fog-xenserver-0.2.3.gem (100%)
Successfully installed fog-xenserver-0.2.3
Fetching: trollop-2.1.2.gem (100%)
Successfully installed trollop-2.1.2
Fetching: rbvmomi-1.8.2.gem (100%)
Successfully installed rbvmomi-1.8.2
.
.
.
Parsing documentation for rake-11.1.2
Done installing documentation for rake after 3 seconds
Updating rdoc
Fetching: rdoc-4.2.2.gem (100%)
Depending on your version of ruby, you may need to install ruby rdoc/ri data:

<= 1.8.6 : unsupported
 = 1.8.7 : gem install rdoc-data; rdoc-data --install
 = 1.9.1 : gem install rdoc-data; rdoc-data --install
>= 1.9.2 : nothing to do! Yay!
```

```
Successfully installed rdoc-4.2.2
Parsing documentation for rdoc-4.2.2
Installing ri documentation for rdoc-4.2.2
Installing darkfish documentation for rdoc-4.2.2
(eval):3: warning: string literal in condition
(eval):2: warning: string literal in condition
Done installing documentation for rdoc after 56 seconds
Parsing documentation for rdoc-4.2.2
Done installing documentation for rdoc after 34 seconds
Updating test-unit
Fetching: test-unit-3.1.8.gem (100%)
Successfully installed test-unit-3.1.8
Parsing documentation for test-unit-3.1.8
Installing ri documentation for test-unit-3.1.8
Installing darkfish documentation for test-unit-3.1.8
Done installing documentation for test-unit after 12 seconds
Parsing documentation for test-unit-3.1.8
Done installing documentation for test-unit after 7 seconds
Gems updated: fog fog-aliyun fog-cloudatcost fog-dynect fog-google fog-openstack fog-
rackspace fog-vsphere fog-xenserver rbvmomi trollop xml-simple mini_portile2 minitest
power_assert rake rdoc test-unit
```

7. Let's install ruby with version 2.1.0 using ruby version manager:

```
  [root@devops1 Desktop]# rvm install 2.1.0
  Searching for binary rubies, this might take some time.
  Found remote file
https://rvm_io.global.ssl.fastly.net/binaries/centos/6/x86_64/ruby-2.1.0.tar.bz2
  Checking requirements for centos.
  Requirements installation successful.
  ruby-2.1.0 - #configure
  ruby-2.1.0 - #download
    % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                   Dload  Upload   Total   Spent    Left  Speed
  100 20.1M  100 20.1M    0     0   147k      0  0:02:19  0:02:19 --:--:--  143k
  ruby-2.1.0 - #validate archive
  ruby-2.1.0 - #extract
  ruby-2.1.0 - #validate binary
  ruby-2.1.0 - #setup
  ruby-2.1.0 - #gemset created /usr/local/rvm/gems/ruby-2.1.0@global
  ruby-2.1.0 - #importing gemset /usr/local/rvm/gemsets/global.gems.....................................
  ruby-2.1.0 - #generating global wrappers........
  ruby-2.1.0 - #gemset created /usr/local/rvm/gems/ruby-2.1.0
  ruby-2.1.0 - #importing gemsetfile /usr/local/rvm/gemsets/default.gems evaluated to empty
gem list
  ruby-2.1.0 - #generating default wrappers........
```

8. Use the latest version of the ruby in terminal for command execution:

```
[root@devops1 Desktop]# rvm use 2.1.0
Using /usr/local/rvm/gems/ruby-2.1.0
```

9. Verify the overall system with gem update –system:

```
[root@devops1 Desktop]# gem update --system
Updating rubygems-update
Fetching: rubygems-update-2.6.4.gem (100%)
Successfully installed rubygems-update-2.6.4
Parsing documentation for rubygems-update-2.6.4
Installing ri documentation for rubygems-update-2.6.4
Installing darkfish documentation for rubygems-update-2.6.4
Done installing documentation for rubygems-update after 5 seconds
Installing RubyGems 2.6.4
RubyGems 2.6.4 installed
Parsing documentation for rubygems-2.6.4
Installing ri documentation for rubygems-2.6.4
=== 2.6.3 / 2016-04-05
------------------------------------------------------------------------------
RubyGems installed the following executables:
    /usr/local/rvm/rubies/ruby-2.1.0/bin/gem
Ruby Interactive (ri) documentation was installed. ri is kind of like man
pages for ruby libraries. You may access it like this:
  ri Classname
  ri Classname.class_method
  ri Classname#instance_method
If you do not wish to install this documentation in the future, use the
--no-document flag, or set it as the default in your ~/.gemrc file. See
'gem help env' for details.
RubyGems system software updated
```

10. Now everything is updated and working fine. Let's install rails:

```
[root@devops1 Desktop]# gem install rails
Fetching: rack-1.6.4.gem (100%)
Successfully installed rack-1.6.4
Fetching: concurrent-ruby-1.0.2.gem (100%)
Successfully installed concurrent-ruby-1.0.2
Fetching: sprockets-3.6.0.gem (100%)
Successfully installed sprockets-3.6.0
.
.
.
Parsing documentation for mail-2.6.4
Installing ri documentation for mail-2.6.4
Parsing documentation for actionmailer-4.2.6
Installing ri documentation for actionmailer-4.2.6
Parsing documentation for rails-4.2.6
```

```
Installing ri documentation for rails-4.2.6
Done installing documentation for rack, concurrent-ruby, sprockets, thread_safe, tzinfo, minitest, i18n, activesupport, mini_portile2, nokogiri, loofah, rails-html-sanitizer, rails-deprecated_sanitizer, rails-dom-testing, rack-test, erubis, builder, actionview, actionpack, sprockets-rails, thor, railties, arel, activemodel, activerecord, globalid, activejob, mime-types-data, mime-types, mail, actionmailer, rails after 1204 seconds
32 gems installed
```

11. Finally, execute /opt/chefdk/embedded/bin/gem install knife-ec2 to install knife ec2 plugin:

```
[root@devops1 Desktop]# /opt/chefdk/embedded/bin/gem install knife-ec2
WARNING:  You don't have /root/.chefdk/gem/ruby/2.1.0/bin in your PATH,
      gem executables will not run.
Successfully installed rubyntlm-0.6.0
Successfully installed nori-2.6.0
Successfully installed multi_json-1.12.0
Successfully installed little-plugger-1.1.4
Successfully installed logging-2.1.0
Successfully installed httpclient-2.8.0
Successfully installed gyoku-1.3.1
Building native extensions.  This could take a while...
Successfully installed ffi-1.9.10
Successfully installed gssapi-1.2.0
Successfully installed winrm-1.8.1
Successfully installed knife-windows-1.4.1
.
.
.
Fetching: fog-1.29.0.gem (100%)
Successfully installed fog-1.29.0
Fetching: knife-ec2-0.12.0.gem (100%)
Successfully installed knife-ec2-0.12.0
38 gems installed
```

12. Verify whether knife ec2 commands are available for execution or not:

```
[root@devops1 Desktop]# knife ec2 --help
FATAL: Cannot find subcommand for: 'ec2 --help'
Available ec2 subcommands: (for details, knife SUB-COMMAND --help)
** EC2 COMMANDS **
knife ec2 amis ubuntu DISTRO [TYPE] (options)
knife ec2 flavor list (options)
knife ec2 server create (options)
knife ec2 server delete SERVER [SERVER] (options)
knife ec2 server list (options)
```

We have successfully installed knife ec2 plugin. Let's install knife azure plugin to create and

manage Microsoft Azure resources:

1. Here, we have mentioned version of a plugin as well because at the time of writing there was some issues with the latest version of the plugin:

```
[root@devops1 Desktop]# /opt/chefdk/embedded/bin/gem install knife-azure -v 1.5.2
WARNING:  You don't have /root/.chefdk/gem/ruby/2.1.0/bin in your PATH,
    gem executables will not run.
Successfully installed rubyntlm-0.6.0
Successfully installed nori-2.6.0
Successfully installed multi_json-1.12.0
Successfully installed little-plugger-1.1.4
Successfully installed logging-2.1.0
Successfully installed httpclient-2.8.0
Successfully installed gyoku-1.3.1
Building native extensions.  This could take a while...
Successfully installed ffi-1.9.10
Successfully installed gssapi-1.2.0
Successfully installed winrm-1.8.1
Successfully installed knife-windows-1.4.1
Successfully installed knife-azure-1.5.2
12 gems installed
```

2. Let's try to install plugin for VMware workstation. Use ruby version 2.1.0:

```
[root@devops1 Desktop]# rvm use 2.1.0
Using /usr/local/rvm/gems/ruby-2.1.0
```

3. Install knife-wsfusion plugin:

```
[root@devops1 Desktop]# /opt/chefdk/embedded/bin/gem install knife-wsfusion
Fetching: uuidtools-2.1.5.gem (100%)
WARNING:  You don't have /root/.chefdk/gem/ruby/2.1.0/bin in your PATH,
    gem executables will not run.
Successfully installed uuidtools-2.1.5
Fetching: syslog-logger-1.6.8.gem (100%)
Successfully installed syslog-logger-1.6.8
Fetching: sfl-2.2.gem (100%)
Successfully installed sfl-2.2
Fetching: net-telnet-0.1.1.gem (100%)
Successfully installed net-telnet-0.1.1
Fetching: net-ssh-3.1.1.gem (100%)
.
.
.
Fetching: chef-zero-4.6.2.gem (100%)
Successfully installed chef-zero-4.6.2
Fetching: bundler-1.12.3.gem (100%)
```

```
Successfully installed bundler-1.12.3
Fetching: chef-12.9.41.gem (100%)
Successfully installed chef-12.9.41
Fetching: knife-wsfusion-0.1.1.gem (100%)
Successfully installed knife-wsfusion-0.1.1
42 gems installed
```

4. Now, knife-wsfusion plugin is installed. Let's verify all the require knife plugins available now:

```
[root@devops1 Desktop]# knife --help
Usage: knife sub-command (options)
    -s, --server-url URL             Chef Server URL
        --chef-zero-host HOST        Host to start chef-zero on
        --chef-zero-port PORT        Port (or port range) to start chef-zero on.  Port
                                     ranges like 1000,1010 or 8889-9999 will try all given
                                     ports until one works.
    -k, --key KEY                    API Client Key
        --[no-]color                 Use colored output, defaults to enabled
    -c, --config CONFIG              The configuration file to use
        --defaults                   Accept default values for all questions
    -d, --disable-editing            Do not open EDITOR, just accept the data as is
    -e, --editor EDITOR              Set the editor to use for interactive commands
    -E, --environment ENVIRONMENT    Set the Chef environment (except for in searches,
                                     where this will be flagrantly ignored)
        --[no-]fips                  Enable fips mode
    -F, --format FORMAT              Which format to use for output
        --[no-]listen                Whether a local mode (-z) server binds to a port
    -z, --local-mode                 Point knife commands at local repository instead of
                                     server
    -u, --user USER                  API Client Username
        --print-after                Show the data after a destructive operation
    -V, --verbose                    More verbose output. Use twice for max verbosity
    -v, --version                    Show chef version
    -y, --yes                        Say yes to all prompts for confirmation
    -h, --help                       Show this message
Available subcommands: (for details, knife SUB-COMMAND --help)
** AZURE COMMANDS **
knife azure ag create (options)
knife azure ag list (options)
knife azure image list (options)
knife azure internal lb create (options)
knife azure internal lb list (options)
knife azure server create (options)
knife azure server delete SERVER [SERVER] (options)
knife azure server list (options)
knife azure server show SERVER [SERVER]
knife azure vnet create (options)
```

```
knife azure vnet list (options)
.
.
.
** EC2 COMMANDS **
knife ec2 amis ubuntu DISTRO [TYPE] (options)
knife ec2 flavor list (options)
knife ec2 server create (options)
knife ec2 server delete SERVER [SERVER] (options)
knife ec2 server list (options)
.
.
.
** WSFUSION COMMANDS **
knife wsfusion create (options)
** WSMAN COMMANDS **
knife wsman test QUERY (options)
[root@devops1 Desktop]#
```

We have successfully installed knife plugins and in the next section we will try to create virtual machine in the Amazon EC2.

Creating and Configuring Virtual Machine in Amazon EC2

Before creating and configuring virtual machine in Amazon EC2, let's verify existing nodes converged by Chef. Local virtual machine is only configured using Chef:

```
[root@devops1 Desktop]# knife node list
tomcatserver
```

1. To provision a new virtual machine require following parameters with knife ec2 server create command:

Parameter	Value	Description
-I	**ami-1ecae776**	Id of Amazon Machine Image
-f	**t2.micro**	Type of Virtual Machine
-N	**DevOpsVMonAWS**	Name of the Chef Node
–aws-access-key-id	Your Access Key ID	AWS Account Access Key ID
–aws-secret-access-key	Your Secret Access Key	AWS Account Secret Access Key

-S	Book	SSH Key
–identity-file	book.pem	.PEM File
–ssh-user	ec2-user	User for AWS Instance
-r	role[v-tomcat]	Chef Role

```
[root@devops1 Desktop]# knife ec2 server create -I ami-1ecae776 -f t2.micro -N
DevOpsVMonAWS --aws-access-key-id '< Your Access Key ID >' --aws-secret-access-key '<
Your Secret Access Key >' -S book --identity-file book.pem --ssh-user ec2-user -r role[v-
tomcat]
  Instance ID: i-640d2de3
  Flavor: t2.micro
  Image: ami-1ecae776
  Region: us-east-1
  Availability Zone: us-east-1a
  Security Groups: default
  Tags: Name: DevOpsVMonAWS
  SSH Key: book
  Waiting for EC2 to create the instance......
  Public DNS Name: ec2-52-90-219-205.compute-1.amazonaws.com
  Public IP Address: 52.90.219.205
  Private DNS Name: ip-172-31-1-27.ec2.internal
  Private IP Address: 172.31.1.27
```

2. At this stage AWS EC2 instance is created and it is waiting for sshd access to become available:

```
  Waiting for sshd access to become available.....done
  Creating new client for DevOpsVMonAWS
  Creating new node for DevOpsVMonAWS
  Connecting to ec2-52-90-219-205.compute-1.amazonaws.com
  ec2-52-90-219-205.compute-1.amazonaws.com -----> Installing Chef Omnibus (-v 12)
  ec2-52-90-219-205.compute-1.amazonaws.com downloading
https://omnitruck-direct.chef.io/chef/install.sh
  ec2-52-90-219-205.compute-1.amazonaws.com   to file /tmp/install.sh.2311/install.sh
  ec2-52-90-219-205.compute-1.amazonaws.com trying wget...
  ec2-52-90-219-205.compute-1.amazonaws.com el 6 x86_64
  ec2-52-90-219-205.compute-1.amazonaws.com Getting information for chef stable 12 for el...
  ec2-52-90-219-205.compute-1.amazonaws.com downloading
https://omnitruck-direct.chef.io/stable/chef/metadata?v=12&p=el&pv=6&m=x86_64
  ec2-52-90-219-205.compute-1.amazonaws.com   to file /tmp/install.sh.2316/metadata.txt
  ec2-52-90-219-205.compute-1.amazonaws.com trying wget...
  ec2-52-90-219-205.compute-1.amazonaws.com sha1
859bc9be9a40b8b13fb88744079ceef1832831b0
  ec2-52-90-219-205.compute-1.amazonaws.com sha256
c43f48e5a2de56e4eda473a3ee0a80aa1aaa6c8621d9084e033d8b9cf3efc328
  ec2-52-90-219-205.compute-1.amazonaws.com url
```

```
https://packages.chef.io/stable/el/6/chef-12.9.41-1.el6.x86_64.rpm
  ec2-52-90-219-205.compute-1.amazonaws.com version 12.9.41
  ec2-52-90-219-205.compute-1.amazonaws.com downloaded metadata file looks valid...
  ec2-52-90-219-205.compute-1.amazonaws.com downloading
https://packages.chef.io/stable/el/6/chef-12.9.41-1.el6.x86_64.rpm
  ec2-52-90-219-205.compute-1.amazonaws.com   to file
/tmp/install.sh.2316/chef-12.9.41-1.el6.x86_64.rpm
  ec2-52-90-219-205.compute-1.amazonaws.com trying wget...
  ec2-52-90-219-205.compute-1.amazonaws.com Comparing checksum with sha256sum...
  ec2-52-90-219-205.compute-1.amazonaws.com Installing chef 12
  ec2-52-90-219-205.compute-1.amazonaws.com installing with rpm...
  ec2-52-90-219-205.compute-1.amazonaws.com warning:
/tmp/install.sh.2316/chef-12.9.41-1.el6.x86_64.rpm: Header V4 DSA/SHA1 Signature, key ID
83ef826a: NOKEY
  ec2-52-90-219-205.compute-1.amazonaws.com Preparing...
########################################## [100%]
  ec2-52-90-219-205.compute-1.amazonaws.com Updating / installing...
  ec2-52-90-219-205.compute-1.amazonaws.com    1:chef-12.9.41-1.el6
########################################## [100%]
  ec2-52-90-219-205.compute-1.amazonaws.com Thank you for installing Chef!
```

3. At this stage, Chef client is installed on AWS instance. It is ready for the very first Chef Client run with version 12.9.41:

```
  ec2-52-90-219-205.compute-1.amazonaws.com Starting the first Chef Client run...
  ec2-52-90-219-205.compute-1.amazonaws.com Starting Chef Client, version 12.9.41
```

4. Now, it is ready to resolve cookbooks based on the role and install runtime environments:

```
  ec2-52-90-219-205.compute-1.amazonaws.com resolving cookbooks for run list: ["tomcat"]
  ec2-52-90-219-205.compute-1.amazonaws.com Synchronizing Cookbooks:
  ec2-52-90-219-205.compute-1.amazonaws.com   - tomcat (0.17.0)
  ec2-52-90-219-205.compute-1.amazonaws.com   - java (1.39.0)
  ec2-52-90-219-205.compute-1.amazonaws.com   - apt (3.0.0)
  ec2-52-90-219-205.compute-1.amazonaws.com   - openssl (4.4.0)
  ec2-52-90-219-205.compute-1.amazonaws.com   - chef-sugar (3.3.0)
  ec2-52-90-219-205.compute-1.amazonaws.com Installing Cookbook Gems:
  ec2-52-90-219-205.compute-1.amazonaws.com Compiling Cookbooks...
  .
  .
  .
  ec2-52-90-219-205.compute-1.amazonaws.com Converging 3 resources
  ec2-52-90-219-205.compute-1.amazonaws.com Recipe: tomcat::default
  ec2-52-90-219-205.compute-1.amazonaws.com   * yum_package[tomcat6] action install
  ec2-52-90-219-205.compute-1.amazonaws.com     - install version 6.0.45-1.4.amzn1 of
package tomcat6
  ec2-52-90-219-205.compute-1.amazonaws.com   * yum_package[tomcat6-admin-webapps]
```

```
action install
  ec2-52-90-219-205.compute-1.amazonaws.com    - install version 6.0.45-1.4.amzn1 of
package tomcat6-admin-webapps
  ec2-52-90-219-205.compute-1.amazonaws.com  * tomcat_instance[base] action configure
(up to date)
  .
  .
  .
```

5. Runtime environment is setup and now it is time to start the tomcat services in AWS instance:

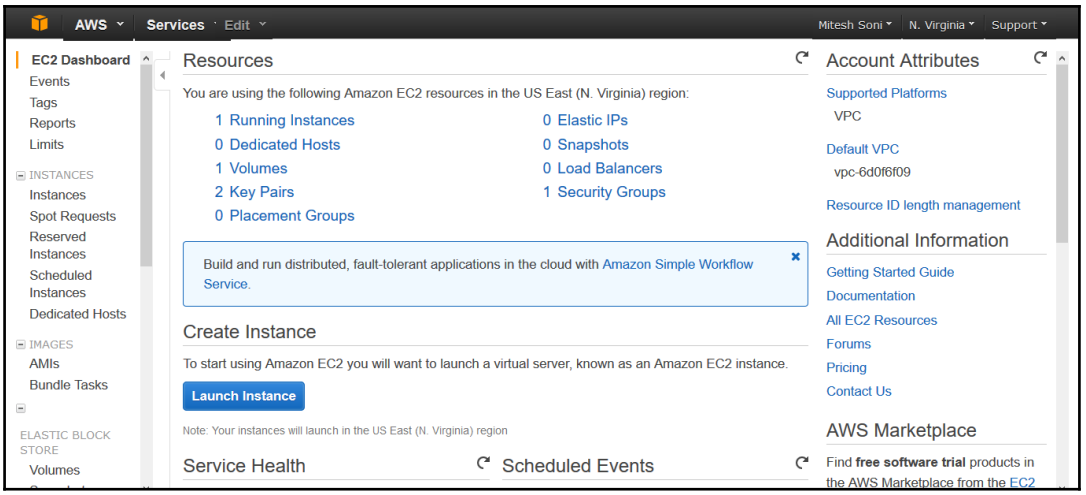
```
  ec2-52-90-219-205.compute-1.amazonaws.com
  ec2-52-90-219-205.compute-1.amazonaws.com  * service[tomcat6] action start
  ec2-52-90-219-205.compute-1.amazonaws.com    - start service service[tomcat6]
  ec2-52-90-219-205.compute-1.amazonaws.com  * execute[wait for tomcat6] action run
  ec2-52-90-219-205.compute-1.amazonaws.com    - execute sleep 5
  ec2-52-90-219-205.compute-1.amazonaws.com  * service[tomcat6] action enable
  ec2-52-90-219-205.compute-1.amazonaws.com    - enable service service[tomcat6]
  ec2-52-90-219-205.compute-1.amazonaws.com  * execute[wait for tomcat6] action run
  ec2-52-90-219-205.compute-1.amazonaws.com    - execute sleep 5
  ec2-52-90-219-205.compute-1.amazonaws.com  * execute[wait for tomcat6] action nothing
(skipped due to action :nothing)
  ec2-52-90-219-205.compute-1.amazonaws.com  * service[tomcat6] action restart
  ec2-52-90-219-205.compute-1.amazonaws.com    - restart service service[tomcat6]
  ec2-52-90-219-205.compute-1.amazonaws.com  * execute[wait for tomcat6] action run
  ec2-52-90-219-205.compute-1.amazonaws.com    - execute sleep 5
  ec2-52-90-219-205.compute-1.amazonaws.com
  ec2-52-90-219-205.compute-1.amazonaws.com Running handlers:
  ec2-52-90-219-205.compute-1.amazonaws.com Running handlers complete
  ec2-52-90-219-205.compute-1.amazonaws.com Chef Client finished, 13/15 resources updated
in 01 minutes 13 seconds
```

6. Details of the newly created AWS instances:

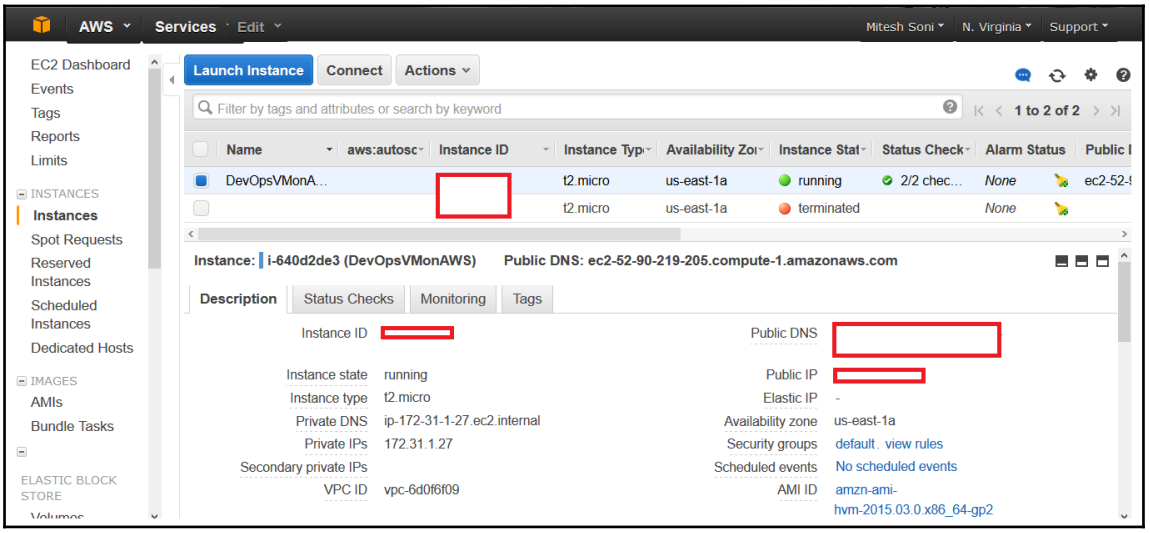
```
  Instance ID: i-640d2de3
  Flavor: t2.micro
  Image: ami-1ecae776
  Region: us-east-1
  Availability Zone: us-east-1a
  Security Groups: default
  Security Group Ids: default
  Tags: Name: DevOpsVMonAWS
  SSH Key: book
  Root Device Type: ebs
  Root Volume ID: vol-1e0e83b5
  Root Device Name: /dev/xvda
  Root Device Delete on Terminate: true
```

```
Block devices
==========================
Device Name: /dev/xvda
Volume ID: vol-1e0e83b5
Delete on Terminate: true
==========================
Public DNS Name: ec2-52-90-219-205.compute-1.amazonaws.com
Public IP Address: 52.90.219.205
Private DNS Name: ip-172-31-1-27.ec2.internal
Private IP Address: 172.31.1.27
Environment: _default
Run List: role[v-tomcat]
You have new mail in /var/spool/mail/root
[root@devops1 Desktop]#
```

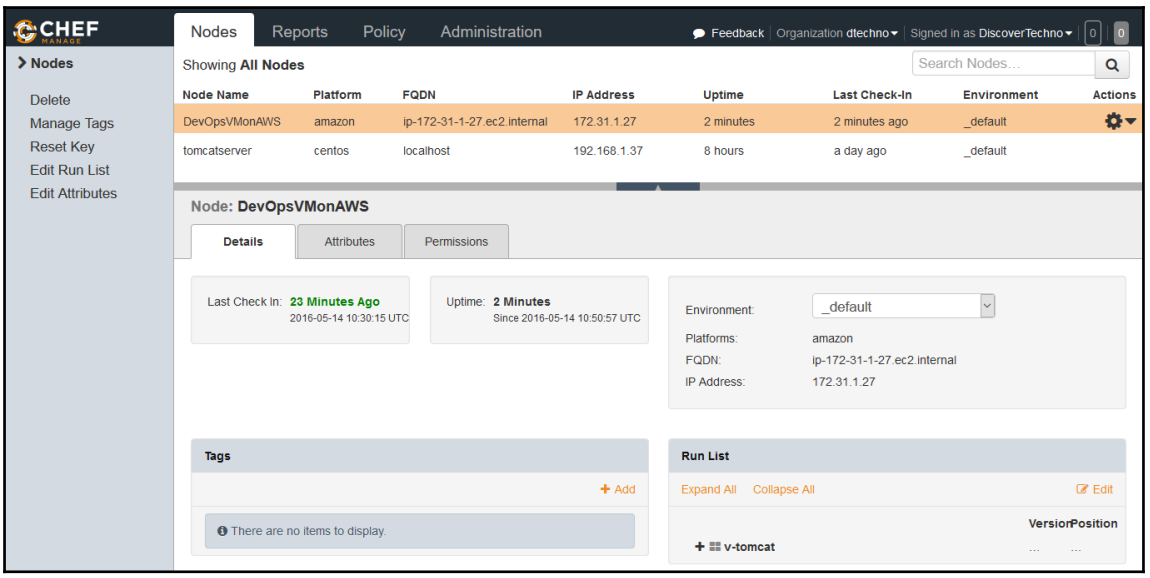
7. Go to https://aws.amazon.com/ and login with admin or IAM credentials:



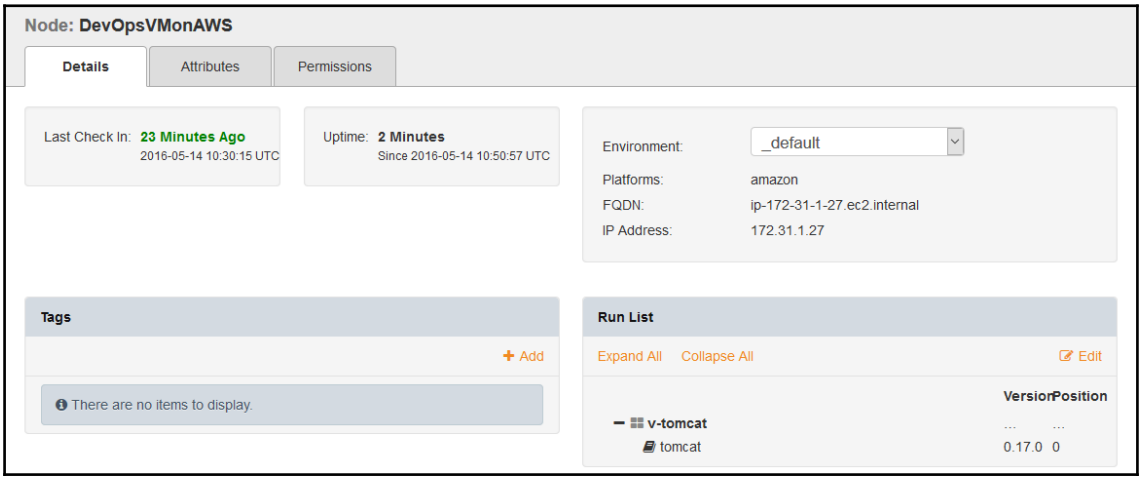
8. Click on the **Instances** in the left sidebar or **Running Instances** to get to the details about AWS instances. Verify **Name**, tag, **Public DNS** and other details that we get in the Chef client run:



9. Now let's go to Hosted Chef dashboard and login. Click on the **Nodes** and verify the newly created / converged node:



10. Verify Instance details and **Run List**:

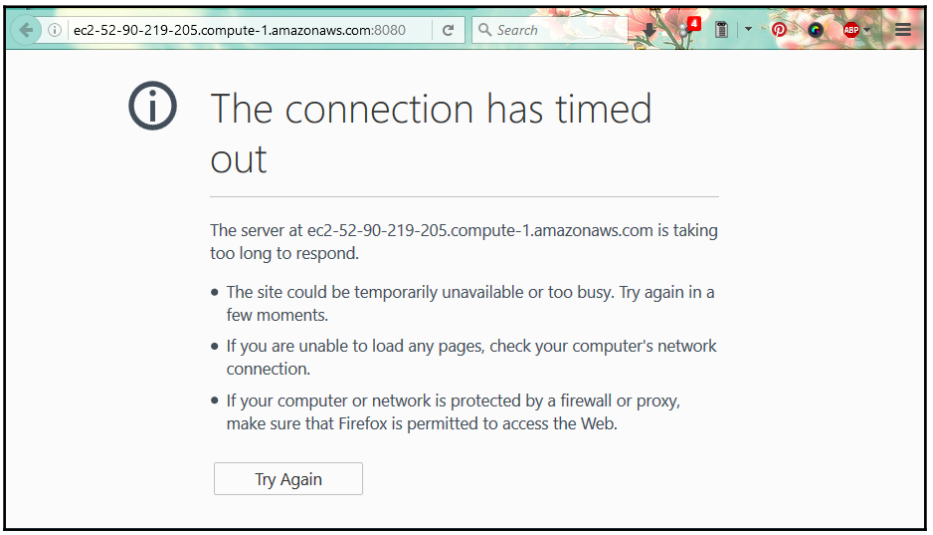


11. Check the **Attributes** section in Hosted Chef dashboard:

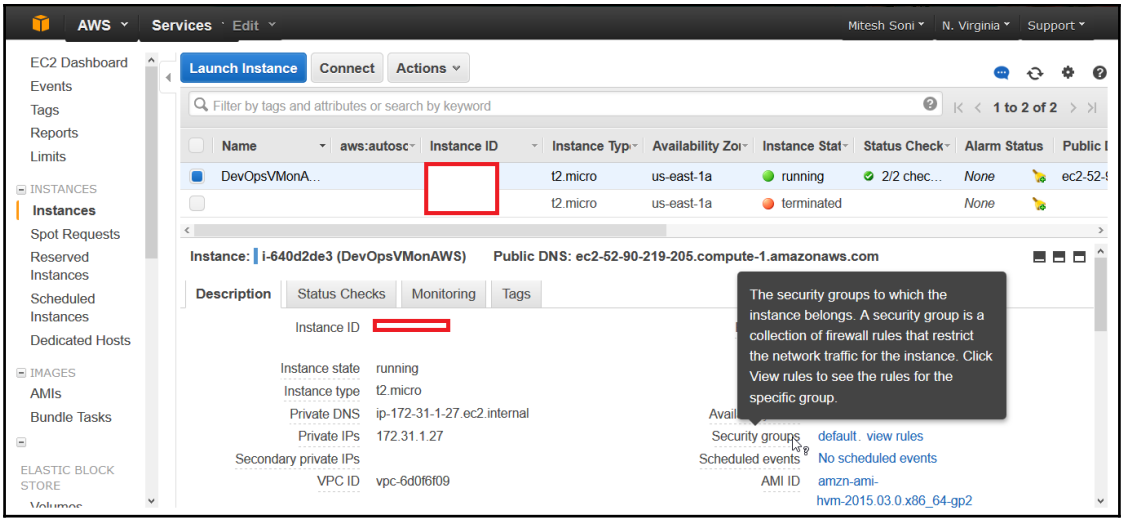
All seems to be nicely finished when it comes to creation and configuration of AWS instances and its registration on Hosted Chef.

Let's try to access the tomcat server installed on newly created AWS instance:

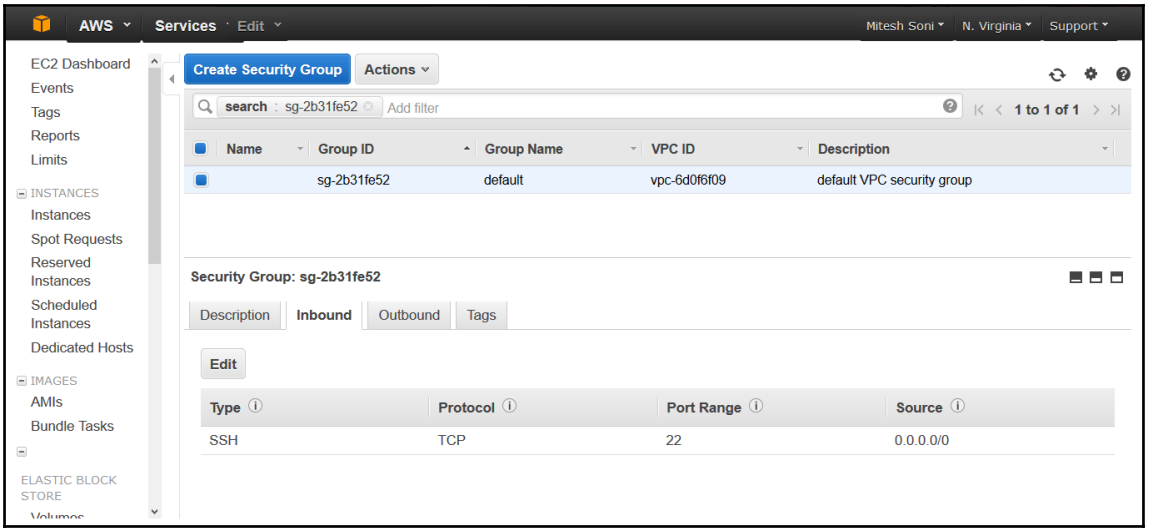
1. We get **The connection has timed out**:



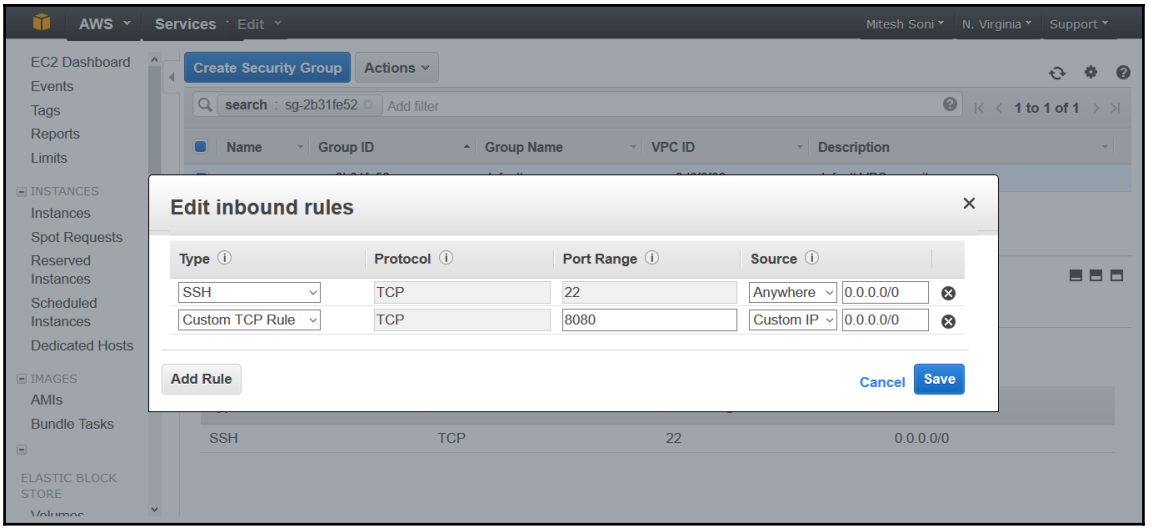
2. The reason for this is restriction of Security Groups in AWS. Verify the **security group** the AWS instance belongs to:



3. Go to **Security groups** section on AWS dashboard. Select the **default** security group and verify **Inbound** rules. We can see only SSH rule is available:



4. Let's edit new custom rule with port 8080:



5. Now verify the URL and we will get the Tomcat page on AWS instance.

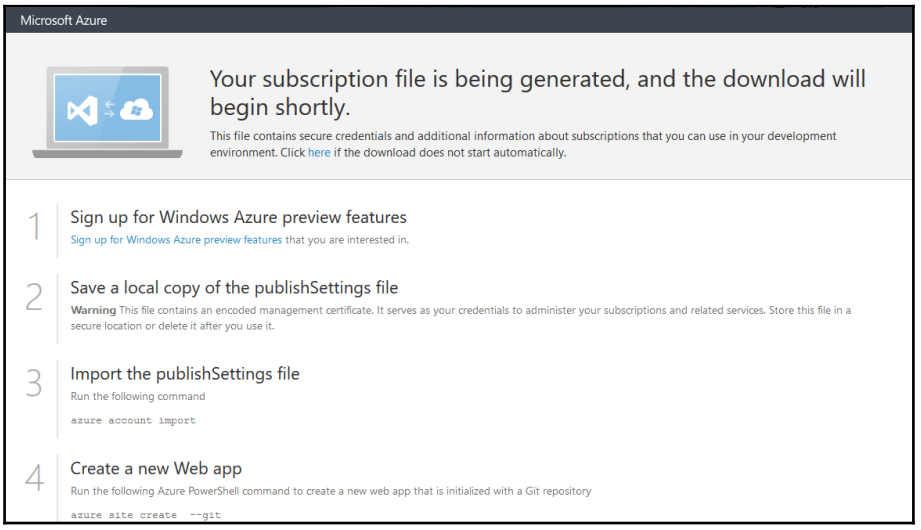
In the next section we will see how to create and configure virtual machine in Microsoft Azure.

Creating and Configuring Virtual Machine in Microsoft Azure

For knife azure plugin to communicate with Azure's REST API, we need to give Knife information regarding our Azure account and credentials:

1. Sign in to the Azure portal and download a publish settings file by visiting https://manage.windowsazure.com/publishsettings/index?client=xplat.
2. Store it on a CHef workstation in to a local file system and refer this local file by doing an entry in knife.rb:

```
knife[:azure_publish_settings_file] = "~/<name>.publishsettings"
```



3. Following are the parameters used to create a virtual machine in Microsoft Azure:

Parameter	Value	Description
–azure-dns-name	distechnodemo	DNS Name
–azure-vm-name	dtserver02	Virtual Machine Name
–azure-vm-size	Small	Virtual Machine Size
-N	DevOpsVMonAzure2	Name of the Chef Node
–azure-storage-account	classicstorage9883	Azure Storage Account
–bootstrap-protocol	cloud-api	Bootstrap Protocol
–azure-source-image	5112500ae3b842c8b9c604889f8753c3__OpenLogic-CentOS-67-20160310	Name of the Azure Source Image
–azure-service-location	Central US	Azure location to host Virtual Machine
–ssh-user	dtechno	SSH User
–ssh-password	**<Your Password>**	SSH Password
-r	role[v-tomcat]	Role
–ssh-port	22	SSH Port

[root@devops1 Desktop]# knife azure server create --azure-dns-name 'distechnodemo' --azure-vm-name 'dtserver02' --azure-vm-size 'Small' -N DevOpsVMonAzure2 --azure-storage-account 'classicstorage9883' --bootstrap-protocol 'cloud-api' --azure-source-image '5112500ae3b842c8b9c604889f8753c3__OpenLogic-CentOS-67-20160310' --azure-service-location 'Central US' --ssh-user 'dtechno' --ssh-password 'cloud@321' -r role[v-tomcat] --ssh-port 22

.....Creating new client for DevOpsVMonAzure2

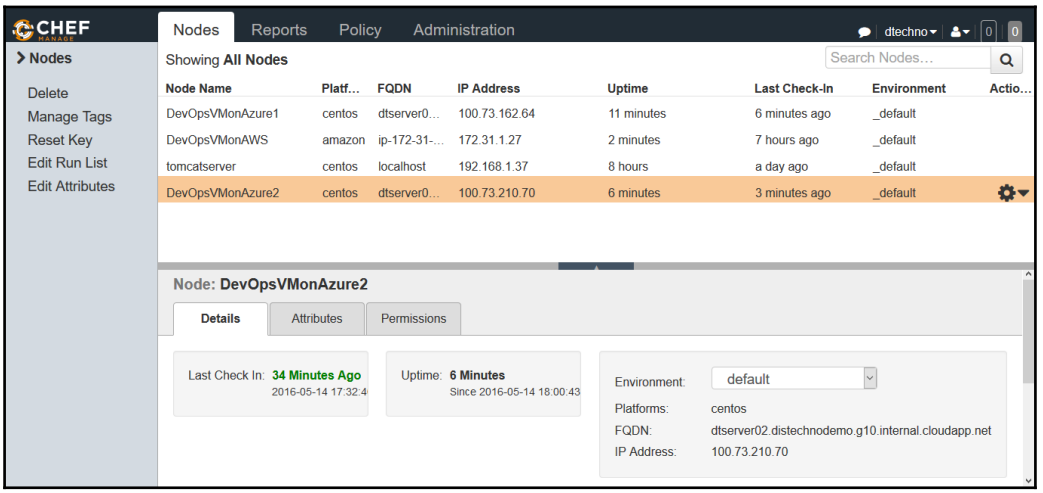
Creating new node for DevOpsVMonAzure2

.....

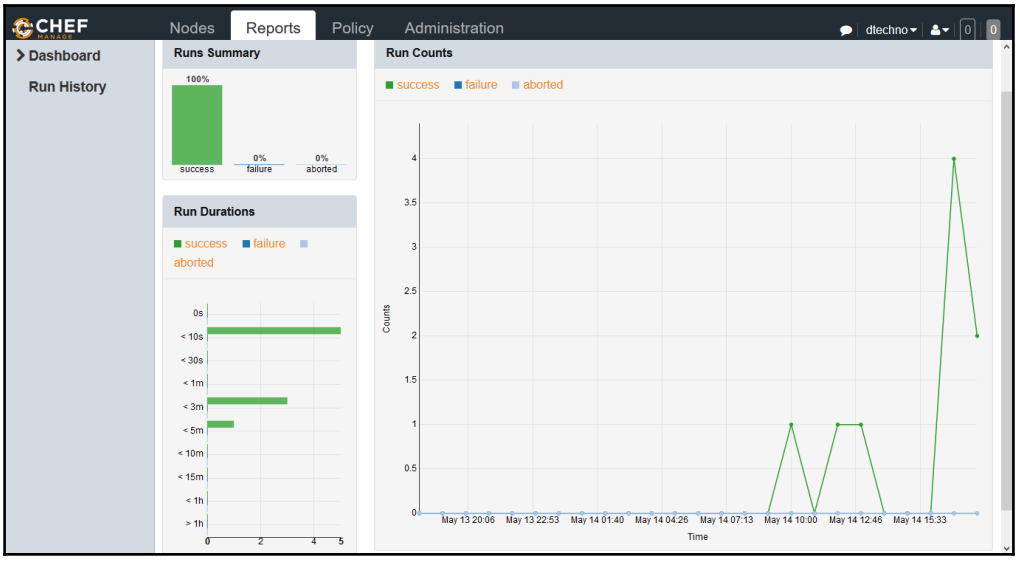
Waiting for virtual machine to reach status 'provisioning'.....vm state 'provisioning' reached after 2.47 minutes.

```
  ..
  DNS Name: distechnodemo.cloudapp.net
  VM Name: dtserver02
  Size: Small
  Azure Source Image: 5112500ae3b842c8b9c604889f8753c3__OpenLogic-
CentOS-67-20160310
  Azure Service Location: Central US
  Private Ip Address: 100.73.210.70
  Environment: _default
  Runlist: ["role[v-tomcat]"]
  Resource provisioning is going to start.
  Waiting for Resource Extension to reach status 'wagent provisioning'.....Resource extension
state 'wagent provisioning' reached after 0.17 minutes.
  Waiting for Resource Extension to reach status 'installing'.....Resource extension
state 'installing' reached after 2.21 minutes.
  Waiting for Resource Extension to reach status 'provisioning'.....Resource extension state
'provisioning' reached after 0.19 minutes.
  ..
  DNS Name: distechnodemo.cloudapp.net
  VM Name: dtserver02
  Size: Small
  Azure Source Image: 5112500ae3b842c8b9c604889f8753c3__OpenLogic-
CentOS-67-20160310
  Azure Service Location: Central US
  Private Ip Address: 100.73.210.70
  Environment: _default
  Runlist: ["role[v-tomcat]"]
  [root@devops1 Desktop]#
```

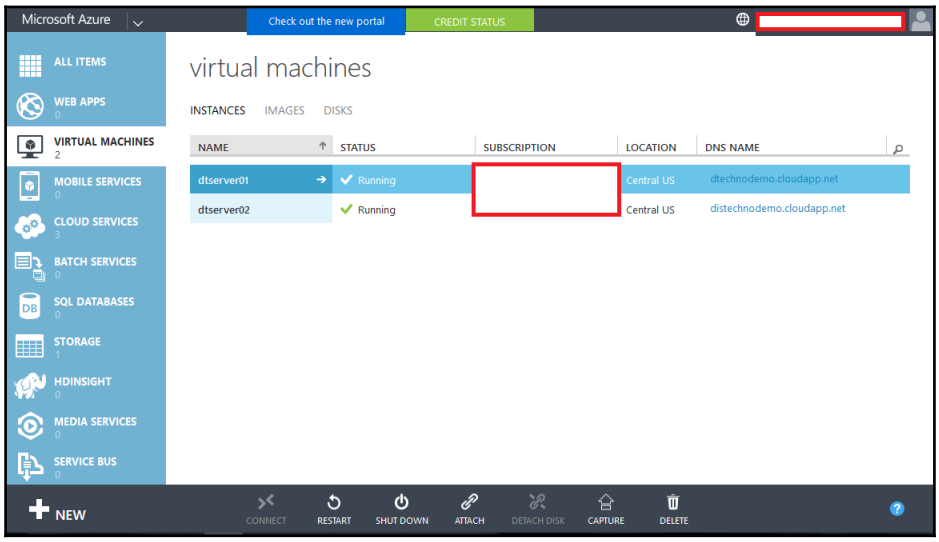
4. Go to Hosted Chef portal and click on the **Nodes** to verify whether new node is registered in the Hosted Chef server or not:



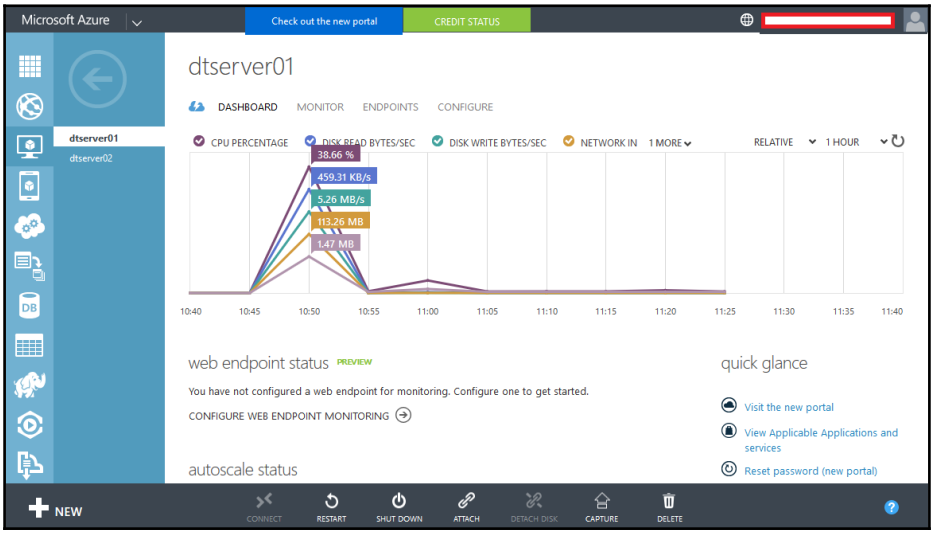
5. Click on the **Reports** section on Hosted Chef Server and verify the graphs for **Runs Summary**, **Run Durations**, and **Run Counts**:



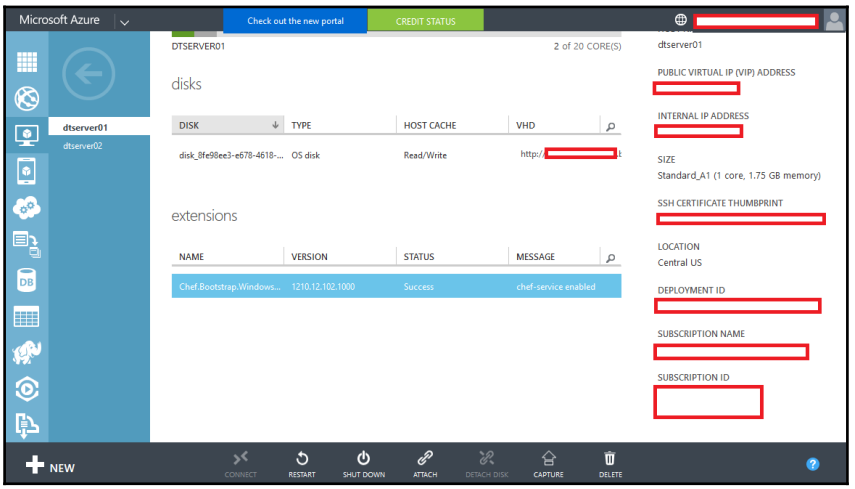
6. Now let's go to Azure Classic Portal and verify the newly created Virtual machine:



7. Click on the **VIRTUAL MACHINES** in Microsoft Azure and get details on it:



8. On the bottom of the page, verify the extensions section and see the chef-server enabled:



Verify the tomcat installation and creating virtual machine in the VMware Workstation as a self-exercise the way we did it for AWS instance.

For VMware workstation, use https://github.com/chipx86/knife-wsfusion for reference.

Just to remind again, we are now close to our main objective and that is end to end automation of application deployment pipeline. We have covered Continuous Integration, Cloud Provisioning, Containers, and Configuration Management. Remaining is actual deployment, monitoring, and orchestration of all activities involved in the end to end automation.

Docker Container

Docker containers are extremely lightweight. We are going to use Tomcat as a web application server to deploy Petclinic application. Docker Hub already have the tomcat image so we are not going to configure too many things except users for accessing Tomcat Manager:

1. In Tomcat-users.xml add role and user as shown in below section:

```
<?xml version='1.0' encoding='utf-8'?>
<!–
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements. See the NOTICE file distributed with this work for additional information regarding copyright ownership.
The ASF licenses this file to You under the Apache License, Version 2.0 (the “License”); you may not use this file except in compliance with the License. You may obtain a copy of the License at
http://www.apache.org/licenses/LICENSE-2.0
Unless required by applicable law or agreed to in writing, software distributed under the License is distributed on an “AS IS” BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
See the License for the specific language governing permissions and limitations under the License.
–>
<tomcat-users xmlns=”http://tomcat.apache.org/xml”
xmlns:xsi=”http://www.w3.org/2001/XMLSchema-instance”
xsi:schemaLocation=”http://tomcat.apache.org/xml tomcat-users.xsd”
version=”1.0”>
<!–
NOTE: By default, no user is included in the “manager-gui” role required to operate the “/manager/html” web application. If you wish to use this app, you must define such a user – the username and password are arbitrary. It is strongly recommended that you do NOT use one of the users in the commented out section below since they are intended for use with the examples web application.
–>
<!–
NOTE: The sample user and role entries below are intended for use with the examples web application. They are wrapped in a comment and thus are ignored when reading this file. If you wish to configure these users for use with the examples web application, do not forget to remove the <!.. ..> that surrounds them. You will also need to set the passwords to something appropriate.
–>
<role rolename=”manager-gui”/>
<user username=”admin” password=”admin@123” roles=”manager-gui”/>
</tomcat-users>
```

2. Now we are going to use the image available in the Docker hub and add the tomcat-sers.xml to /usr/local/tomcat/conf/tomcat-users.xml.Create a Dockerfile as shown below:

```
FROM tomcat:8.0
MAINTAINER Mitesh <mitesh.xxxx @xxxxx.com>
COPY tomcat-users.xml /usr/local/tomcat/conf/tomcat-users.xml
```

3. Once everything is ready, use docker build command to build a new image:

```
[root@localhost mitesh]# docker build -t devopstomcatnew .
Sending build context to Docker daemon 8.192 kB
Sending build context to Docker daemon
Step 0 : FROM tomcat:8.0
 ---> 5d4577339b14
Step 1 : MAINTAINER Mitesh <mitesh.soni@outlook.com>
 ---> Running in 9430cac12c4c
 ---> c63f90db4c14
Removing intermediate container 9430cac12c4c
Step 2 : COPY tomcat-users.xml /usr/local/tomcat/conf/tomcat-users.xml
 ---> eb50c4ceefb5
Removing intermediate container 7f31aed05097
Successfully built eb50c4ceefb5
You have new mail in /var/spool/mail/root
```

4. Image is successfully built. Let's verify using docker images command:

```
[root@localhost mitesh]# docker images
REPOSITORY        TAG              IMAGE ID          CREATED           VIRTUAL SIZE
devopstomcatnew   latest           eb50c4ceefb5      10 seconds ago    359.2 MB
devopstomcat8     latest           f3537165ebe7      10 minutes ago    344.6 MB
devopstomcat      latest           400f097677e9      9 days ago        658.4 MB
tomcat6           latest           400f097677e9      9 days ago        658.4 MB
tomcat            9.0              ce07000625c6      2 weeks ago       344.6 MB
centos            latest           2a332da70fd1      4 weeks ago       196.7 MB
ubuntu            latest           686477c12982      8 weeks ago       120.7 MB
hello-world       latest           f1d956dc5945      9 weeks ago       967 B
```

5. Create a container from the newly created tomcat image.
6. Verify existing containers using docker ps and docker ps -a command:

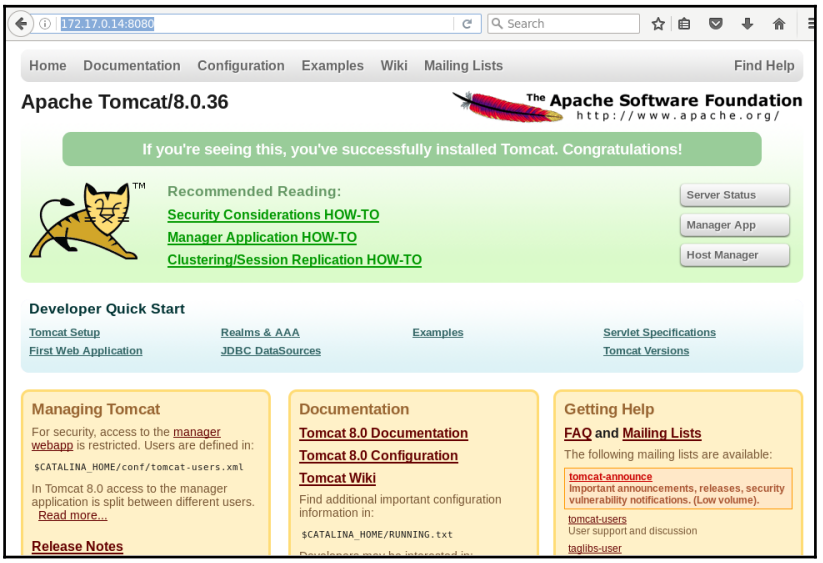
```
[root@localhost mitesh]# docker ps
CONTAINER ID      IMAGE          COMMAND          CREATED
 STATUS           PORTS          NAMES
You have new mail in /var/spool/mail/root
[root@localhost mitesh]# docker ps -a
CONTAINER ID      IMAGE          COMMAND          CREATED
 STATUS           PORTS          NAMES
[root@localhost mitesh]# docker run -p 8180:8080 -d --name devopstomcat1
 devopstomcatnewb5f054ee4ac36d67279db10497fe7a780aecf2a72a7f52fa31ee80
```

```
c618d98e4a
```

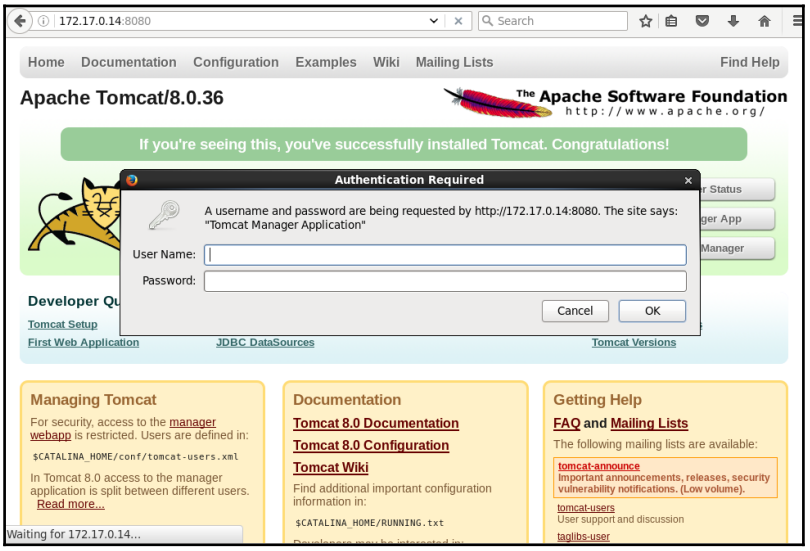
7. Verify existing containers using docker ps and docker ps -a command:

```
[root@localhost mitesh]# docker ps
CONTAINER ID        IMAGE               COMMAND             CREATED
 STATUS              PORTS               NAMES
b5f054ee4ac3        devopstomcatnew     "catalina.sh run"   21 seconds ago
 Up 20 seconds       0.0.0.0:8180->8080/tcp   devopstomcat1
```

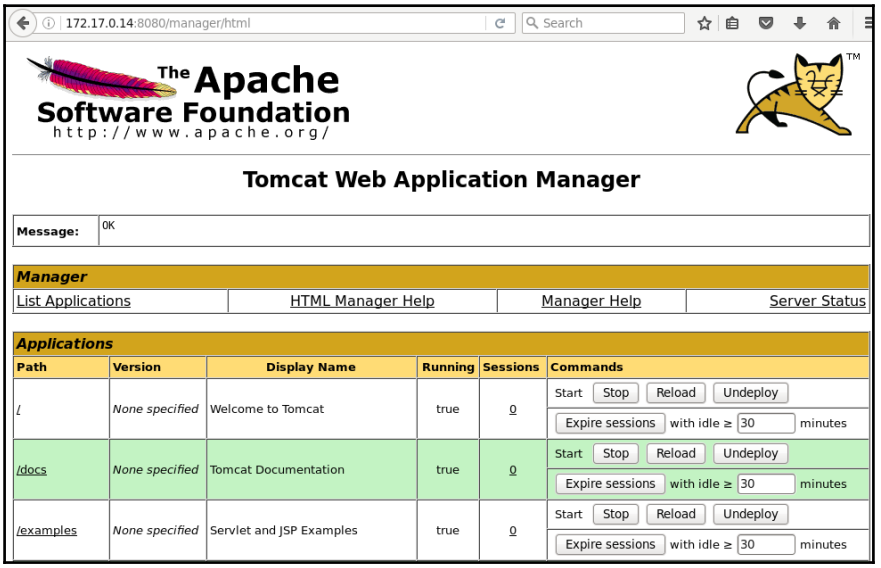
8. Use docker inspect b5f054ee4ac3 command to get the IP address and browse tomcat web server using IP address and Port:



9. Click on the **Manager App** button. It will ask for **User Name** and **Password**. Give Inputs and click on **OK**:



10. Now we can access the tomcat manager application:



We can use **Tomcat Manager Application** to deploy application. Till now we have seen Continuous Integration, Configuration Management, Containers, and Cloud Provisioning.

Next, we will see application deployment using different methods, Monitoring, and End to end automation pipeline using Orchestration.

Self-Test Questions

1. Which of the followings are benefits of Chef Configuration Management?
2. Easy policy enforcement with centralized control
3. Enable setup of consistent runtime environment
4. Enable easy restoration of environments
5. Enable disaster recovery and business continuity
6. Community-based cookbooks and recipes
7. All of the Above

1. Which two parameters are configured for Amazon EC2 credentials for knife-ec2 in knife.rb file?
2. knife[:aws_access_key_id] = “Your AWS Access Key ID”
3. knife[:aws_secret_access_key] = “Your AWS Secret Access Key”
4. Both a and b

1. Which of the followings are knife EC2 commands?
2. knife ec2 flavor list (options)
3. knife ec2 server create (options)
4. knife ec2 server delete SERVER [SERVER] (options)
5. knife ec2 server list (options)
6. All of the Above

1. State True or False: rvm use command is used to set the Ruby version.
2. True
3. False

1. Which of the followings are knife Azure commands?
2. knife azure server create (options)
3. knife azure server delete SERVER [SERVER] (options)
4. knife azure server list (options)
5. knife azure image list (options)
6. All of the Above

1. State True or False: In knife ec2 server create command -I parameter is used for Type of Virtual Machine
2. True
3. False

1. State True or False: In knife ec2 server create command -N parameter is used for Name of the Chef Node
2. True
3. False

Summary

In this chapter, we have covered how to provision resources in Cloud and configure them. We used knife ec2 and knife azure plugin to create virtual machine in AWS and Microsoft Azure. We used Docker Hub Tomcat image to build a new image with `tomcat-users.xml` file which has `role` and `user` configured to access Tomcat Manager web app.

In the next Chapter, we will cover different methods to deploy an application in Tomcat web container. Just to revisit the end goal of the book: End to End automation using application deployment pipeline.

7
Deploying Application in AWS, Azure, and Docker

Ultimate automation… will make our modern industry as primitive and outdated as the stone age man looks to us today.
- Albert Einstein

Finally, we are at the *Business* end of the book and our focus is on deployment automation, monitoring, and orchestration.

Why?

Answer is to achieve **End to End Application lifecycle Automation** or **End to End Deployment Automation**.

First we will go step by step to deploy our Petclinic application into remote tomcat server. Once that is achieved, it can be used as common practice for all. This chapter describes in detail all steps required to deploy sample application into different environment once configuration management tool prepare it for the final deployment. We will also learn how to deploy application in different environments such as cloud or container based environment.

This chapter will also cover on how to Deploy Application on Platform as a Service model. We will deploy application in AWS Elastic Beanstalk.

In this chapter, we will cover the following topics:

- Pre-requisites – To deploy application on Remote Server
- Deploying Application in AWS
- Deploying Application in Microsoft Azure

- Deploying Application in Docker Container

Pre-requisites – To deploy application on Remote Server

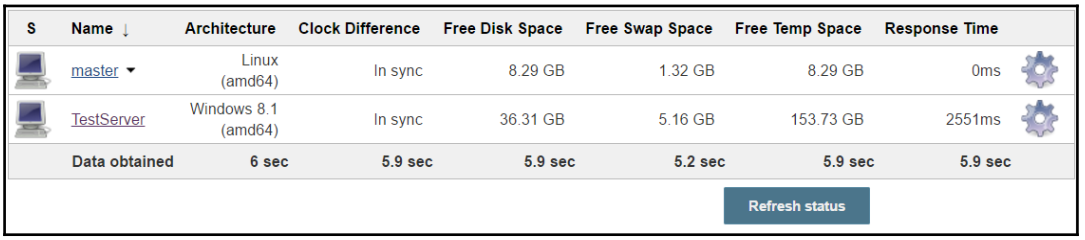
To deploy an application on remote server, let's take the following steps:

1. First, let's start an agent on Windows machine, open command prompt and run the command as it is given in **Manage Nodes** of Jenkins dashboard. Change URL appropriately:

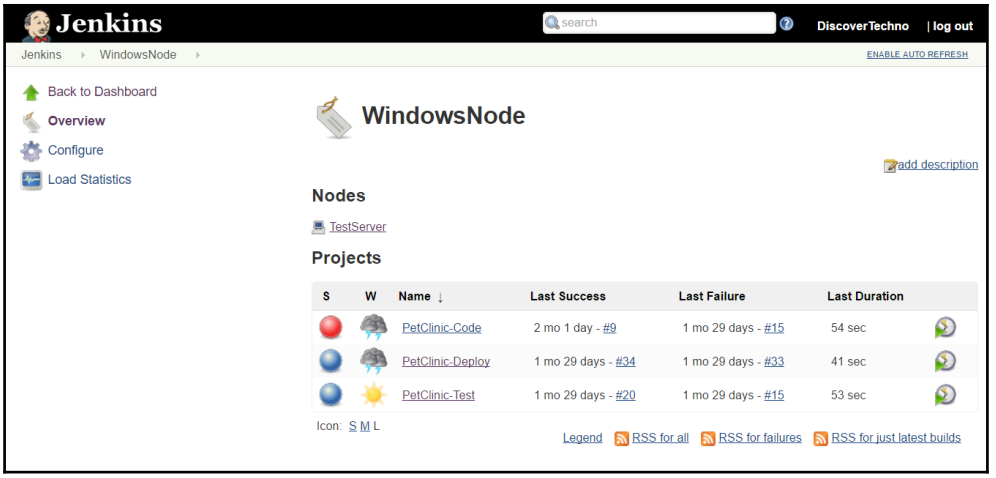
```
java -jar slave.jar -jnlpUrl
http://192.168.0.100:8080/computer/TestServer/slave-agent.jnlp -secret
65464e02c58c85b192883f7848ad2758408220bed2f3af715c01c9b01cb72f9b
```

```
Jul 06, 2016 8:56:54 PM hudson.remoting.jnlp.MaincreateEngine
INFO: Setting up slave: TestServer
Jul 06, 2016 8:56:54 PM hudson.remoting.jnlp.Main$CuiListener<init>
INFO: Jenkins agent is running in headless mode.
Jul 06, 2016 8:56:54 PM hudson.remoting.jnlp.Main$CuiListener status
INFO: Locating server among [http://192.168.1.34:8080/,
http://192.168.0.100:8080/]
Jul 06, 2016 8:57:15 PM hudson.remoting.jnlp.Main$CuiListener status
INFO: Handshaking
Jul 06, 2016 8:57:15 PM hudson.remoting.jnlp.Main$CuiListener status
INFO: Connecting to 192.168.0.100:33903
Jul 06, 2016 8:57:15 PM hudson.remoting.jnlp.Main$CuiListener status
INFO: Trying protocol: JNLP3-connect
Jul 06, 2016 8:57:16 PM hudson.remoting.jnlp.Main$CuiListener status
INFO: Server didn't accept the handshake: Unknown
protocol:Protocol:JNLP3-connect
Jul 06, 2016 8:57:16 PM hudson.remoting.jnlp.Main$CuiListener status
INFO: Connecting to 192.168.0.100:33903
Jul 06, 2016 8:57:16 PM hudson.remoting.jnlp.Main$CuiListener status
INFO: Trying protocol: JNLP2-connect
Jul 06, 2016 8:57:16 PM hudson.remoting.jnlp.Main$CuiListener status
INFO: Connected
```

2. Now our Agent is connected to Master. Let's verify the status of Agent on the Master Node where Jenkins is running.:



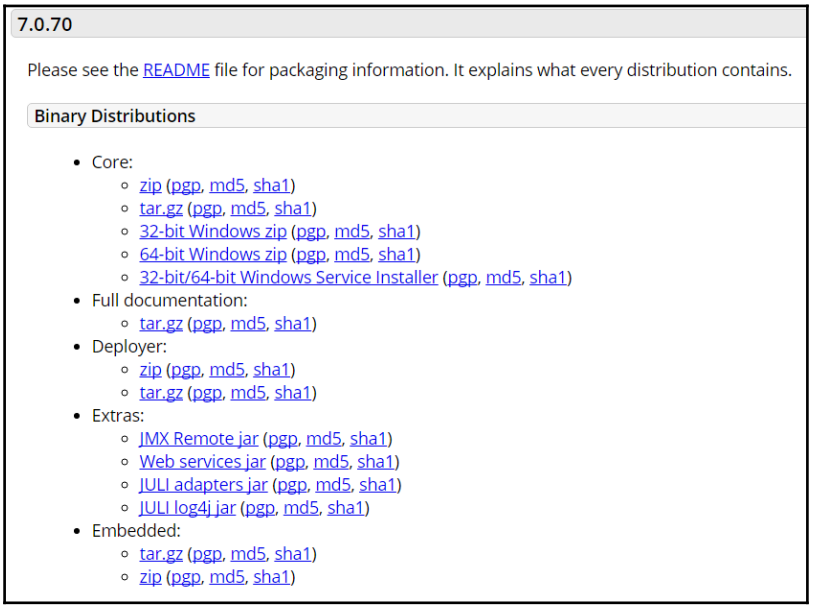
3. Click on the Agent **TestServer** and get all the details regarding projects tied to the Agents as shown in below screenshot:



Once we have Agent node ready, let's prepare a remote server ready by downloading and setting up tomcat server.

In our case we need not to do it for Cloud instances as they will be configured using Chef configuration management tool. This is more understanding perspective on how we used to do it earlier and how all installation and other activities can be automated using Chef. Let's take a step-by-step tour:

1. Download Tomcat 7 version from https://tomcat.apache.org/download-70.cgi. We are going to use Deploy plugin from Jenkins and it requires specific versions of Tomcat for deployment.



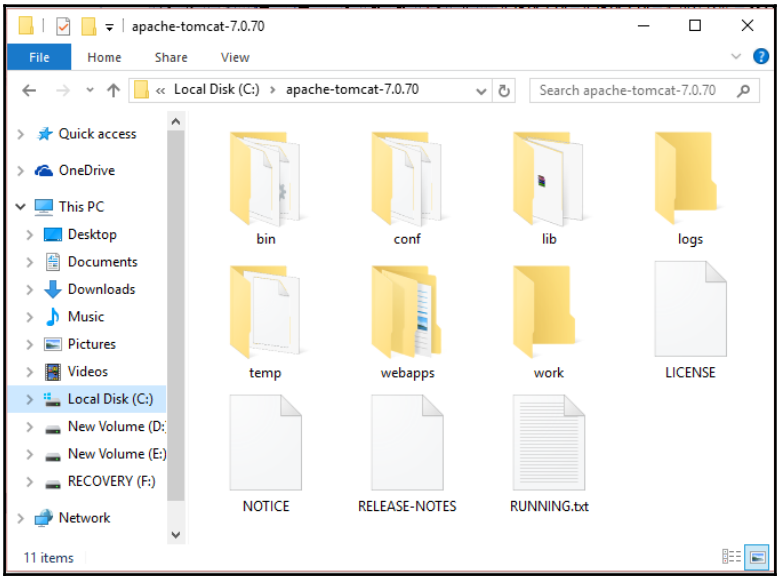
7.0.70

Please see the README file for packaging information. It explains what every distribution contains.

Binary Distributions

- Core:
  - zip (pgp, md5, sha1)
  - tar.gz (pgp, md5, sha1)
  - 32-bit Windows zip (pgp, md5, sha1)
  - 64-bit Windows zip (pgp, md5, sha1)
  - 32-bit/64-bit Windows Service Installer (pgp, md5, sha1)
- Full documentation:
  - tar.gz (pgp, md5, sha1)
- Deployer:
  - zip (pgp, md5, sha1)
  - tar.gz (pgp, md5, sha1)
- Extras:
  - JMX Remote jar (pgp, md5, sha1)
  - Web services jar (pgp, md5, sha1)
  - JULI adapters jar (pgp, md5, sha1)
  - JULI log4j jar (pgp, md5, sha1)
- Embedded:
  - tar.gz (pgp, md5, sha1)
  - zip (pgp, md5, sha1)

2. Extract the tomcat installation files:



3. Open command prompt and go to the bin directory to start the Tomcat.

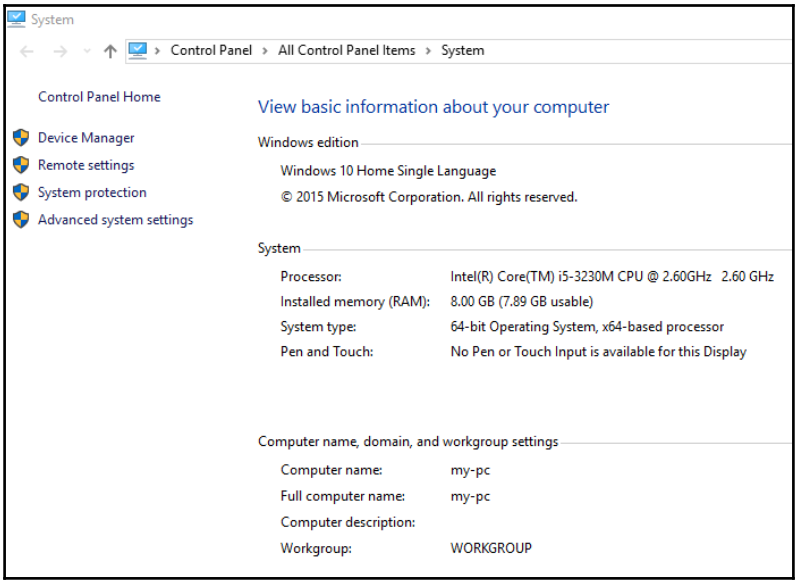
```
C:\>cd apache-tomcat-7.0.70\bin
```

4. Run startup.bat file in the command prompt.

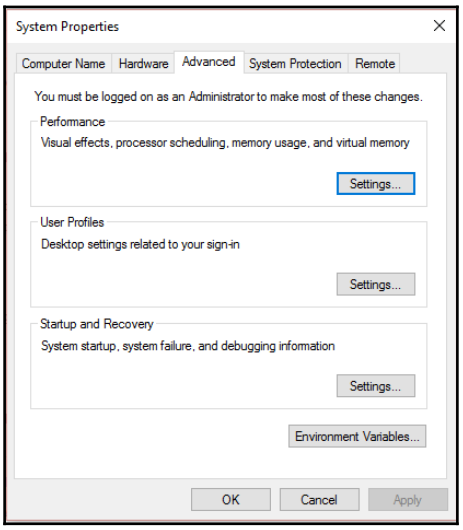
```
C:\apache-tomcat-7.0.70\bin>startup.bat
```

Neither the JAVA_HOME nor the JRE_HOME environment variable is defined. At least one of these environment variable is needed to run this program

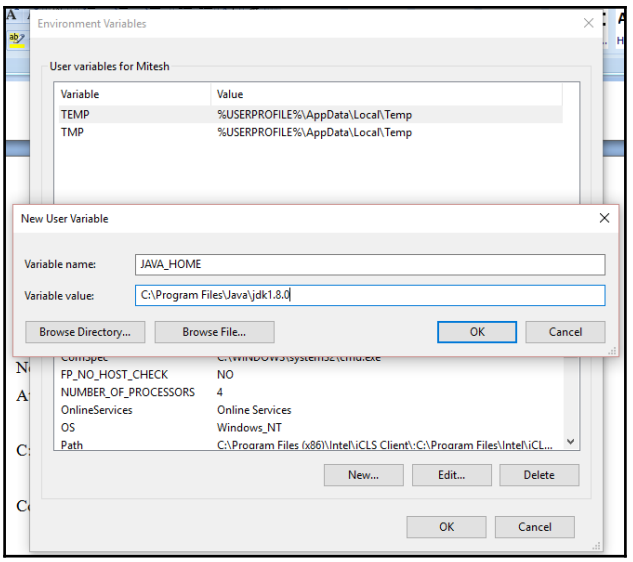
5. Oops! We need to set environment variables. Go to **Control Panel | All Control Panel Items | System**
6. Click on **Advanced system settings**:



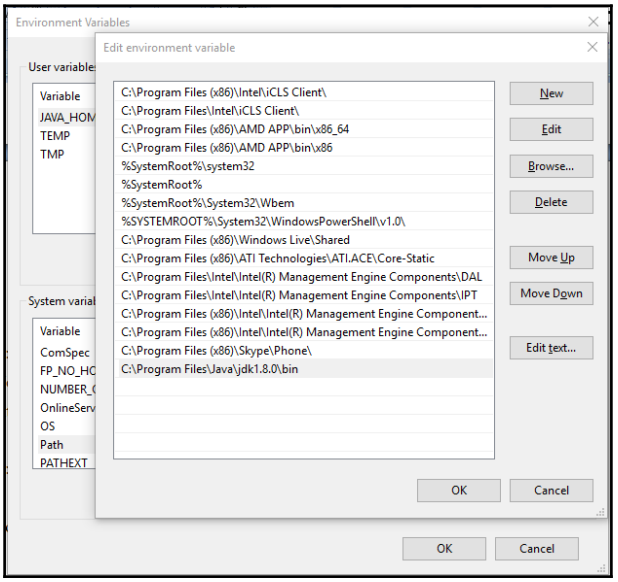
7. Click on the **Environment Variables...** to set JAVA_ HOME:



8. Click on **New...** and create a new variable for JAVA_HOME with value C:\Program Files\Java\jdk1.8.0 and click **OK**:



9. Click **OK** once again:



10. Open new command prompt and verify the Java Version:

```
C:\>java -version
java version "1.8.0-ea"
Java(TM) SE Runtime Environment (build 1.8.0-ea-b115)
Java HotSpot(TM) 64-Bit Server VM (build 25.0-b57, mixed mode)
```

11. Now go to tomcat\bin directory and execute startup.bat file:

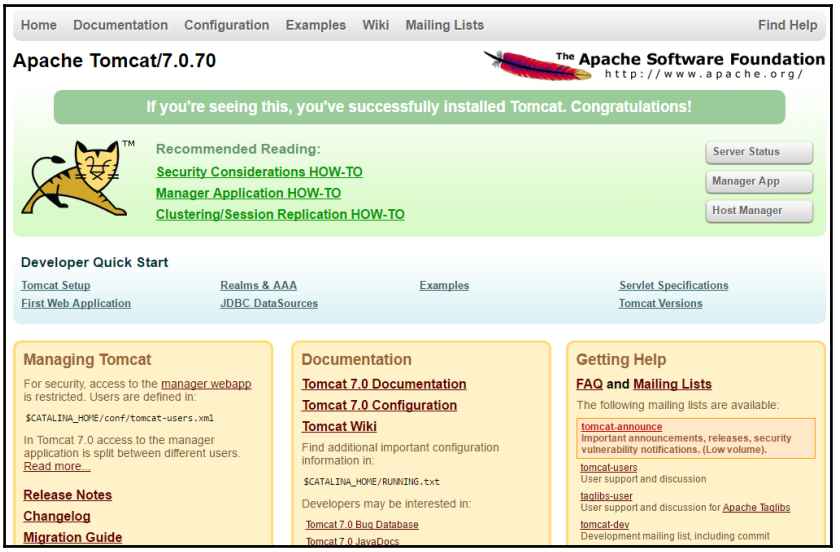
```
C:\apache-tomcat-7.0.70\bin>startup.bat
Using CATALINA_BASE:   "C:\apache-tomcat-7.0.70"
Using CATALINA_HOME:   "C:\apache-tomcat-7.0.70"
Using CATALINA_TMPDIR: "C:\apache-tomcat-7.0.70\temp"
Using JRE_HOME:        "C:\Program Files\Java\jdk1.8.0"
Using CLASSPATH:       "C:\apache-tomcat-7.0.70\bin\bootstrap.jar;C:\apache-
                        tomcat-7.0.70\bin\tomcat-juli.jar"
C:\apache-tomcat-7.0.70\bin>
```

12. Now our Tomcat is running. It may have similar type of output as given below. Verify Server startup message:

```
INFO: Starting Servlet Engine: Apache Tomcat/7.0.70
Jul 06, 2016 9:29:07 PM
org.apache.catalina.startup.HostConfigdeployDirectory
INFO: Deploying web application directory
C:\apache-tomcat-7.0.70\webapps\docs
Jul 06, 2016 9:29:08 PM
org.apache.catalina.util.SessionIdGeneratorBasecreateSecureRandom
INFO: Creation of SecureRandom instance for session ID generation using
[SHA1PRNG] took [331] milliseconds.
Jul 06, 2016 9:29:09 PM
org.apache.catalina.startup.HostConfigdeployDirectory
INFO: Deployment of web application directory
C:\apache-tomcat-7.0.70\webapps\docs has finished in 1,887 ms
Jul 06, 2016 9:29:09 PM
org.apache.catalina.startup.HostConfigdeployDirectory
INFO: Deploying web application directory
C:\apache-tomcat-7.0.70\webapps\examples
Jul 06, 2016 9:29:11 PM
org.apache.catalina.startup.HostConfigdeployDirectory
INFO: Deployment of web application directory
C:\apache-tomcat-7.0.70\webapps\examples has finished in 2,474 ms
Jul 06, 2016 9:29:11 PM
org.apache.catalina.startup.HostConfigdeployDirectory
INFO: Deploying web application directory
C:\apache-tomcat-7.0.70\webapps\host-manager
Jul 06, 2016 9:29:11 PM
org.apache.catalina.startup.HostConfigdeployDirectory
INFO: Deployment of web application directory
C:\apache-tomcat-7.0.70\webapps\host-manager has finished in 140 ms
Jul 06, 2016 9:29:11 PM
org.apache.catalina.startup.HostConfigdeployDirectory
INFO: Deploying web application directory
C:\apache-tomcat-7.0.70\webapps\manager
Jul 06, 2016 9:29:11 PM
org.apache.catalina.startup.HostConfigdeployDirectory
INFO: Deployment of web application directory
C:\apache-tomcat-7.0.70\webapps\manager has finished in 160 ms
Jul 06, 2016 9:29:11 PM
org.apache.catalina.startup.HostConfigdeployDirectory
INFO: Deploying web application directory
C:\apache-tomcat-7.0.70\webapps\ROOT
```

```
Jul 06, 2016 9:29:11 PM
org.apache.catalina.startup.HostConfigdeployDirectory
INFO: Deployment of web application directory
C:\apache-tomcat-7.0.70\webapps\ROOT has finished in 79 ms
Jul 06, 2016 9:29:11 PM org.apache.coyote.AbstractProtocol start
INFO: Starting ProtocolHandler ["http-apr-8080"]
Jul 06, 2016 9:29:11 PM org.apache.coyote.AbstractProtocol start
INFO: Starting ProtocolHandler ["ajp-apr-8009"]
Jul 06, 2016 9:29:11 PM org.apache.catalina.startup.Catalina start
INFO: Server startup in 5172 ms
```

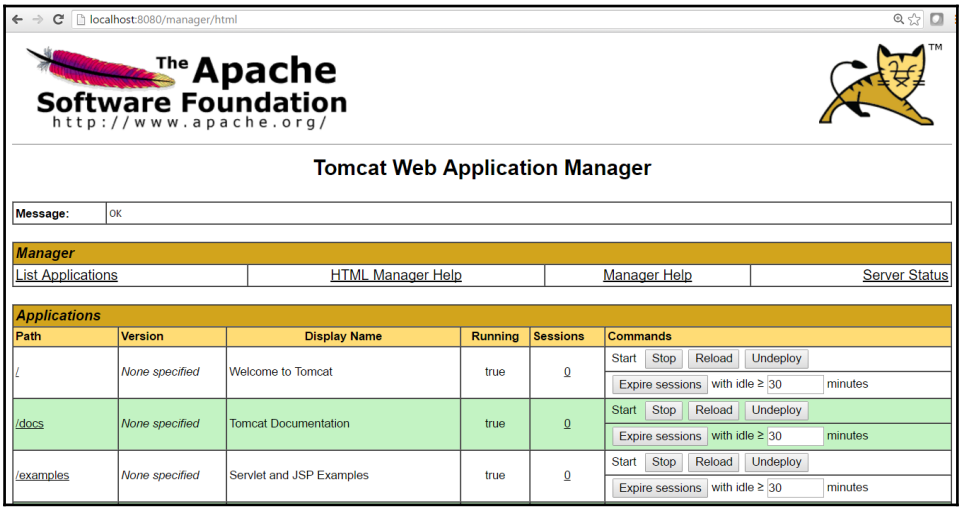
13. Use IP address and port number combination to navigate to tomcat Home page.



14. Go to Tomcat Installation Directory -> conf -> tomcat-users.xml and uncomment Role and User related line or rewrite. Give manager-gui as a **rolename** for testing purpose. We need **manager-script** for deployment via deploy plugin:

```
<?xml version='1.0' encoding='utf-8'?>
<!--
<tomcat-users>
<!--
<!--
  NOTE:  The sample user and role entries below are intended for use with the
  examples web application. They are wrapped in a comment and thus are ignored
  when reading this file. If you wish to configure these users for use with the
  examples web application, do not forget to remove the <!.. ..> that surrounds
  them. You will also need to set the passwords to something appropriate.
-->
  <role rolename="manager-script"/>
  <user username="admin" password="admin@123" roles="manager-script"/>
</tomcat-users>
```

15. Click on the Manager App link on the Tomcat home page and give user name and password given in the tomcat-users.xml. Now we can access Manager App:

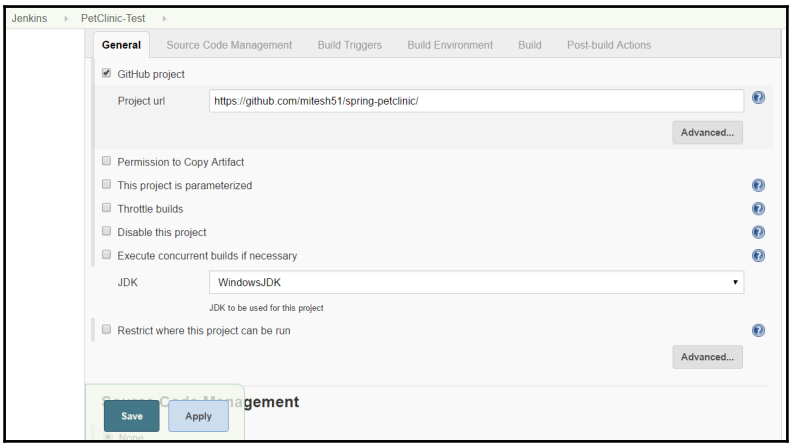


16. For Jenkins Deploy plgin, change the rolename to manager-script.
17. Restart the tomcat and visit http://<IP Address>:8080/manager/text/list

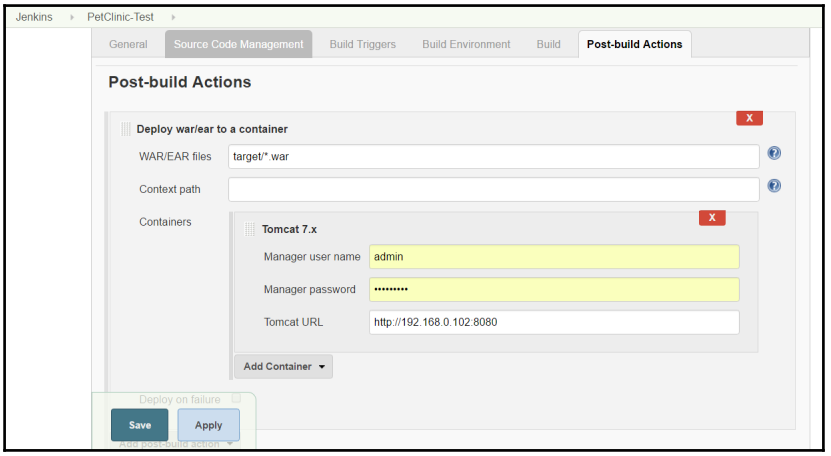
OK - Listed applications for virtual host localhost
/:running:0:ROOT
/petclinic:running:1:petclinic
/examples:running:0:examples
/host-manager:running:0:host-manager
/manager:running:0:manager

/docs:running:0:docs

18. Go to Jenkins Build Job and click on **Configure**. Select the proper JDK configuration for Jenkins agent:



19. In the **Post-build Action**, select **Deploy war/ear to a container**. Provide location of the war file in the Jenkins workspace, Tomcat manager credentials, and Tomcat URL with port:

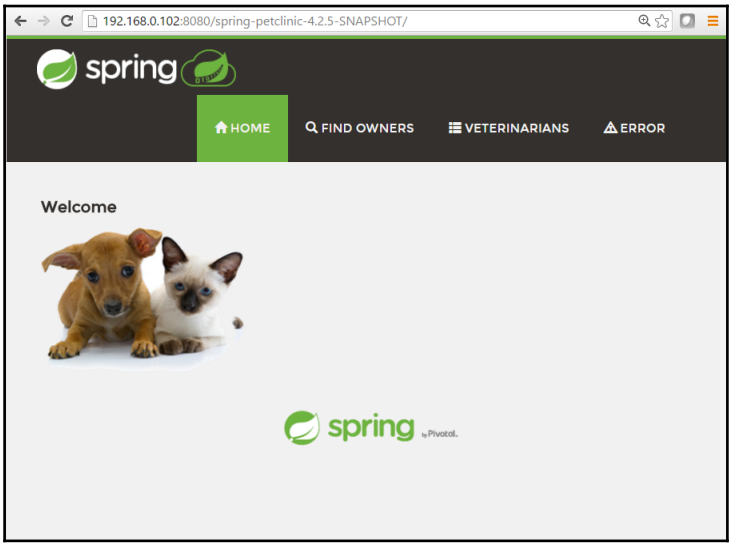


20. Click on **Apply** and **Save**. Click on **Build now** on Jenkins Build specific page.

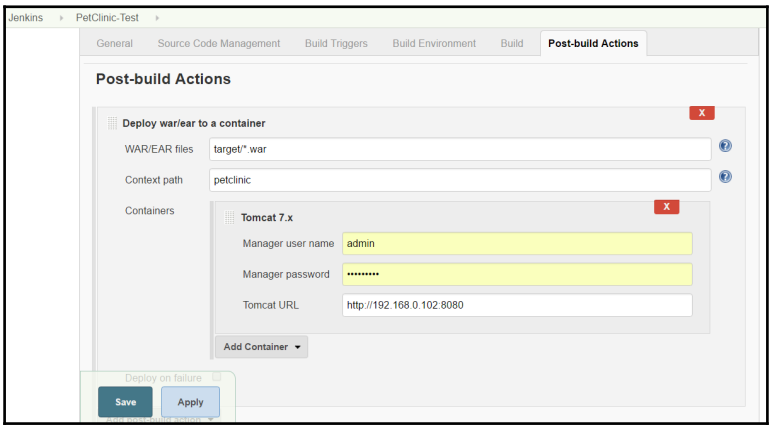
Verify the console output for fresh deployment:

```
Results :

Tests run: 59, Failures: 0, Errors: 0, Skipped: 0

[INFO]
[INFO] --- maven-war-plugin:2.3:war (default-war) @ spring-petclinic ---
[INFO] Packaging webapp
[INFO] Assembling webapp [spring-petclinic] in [d:\jenkins\workspace\PetClinic-Test\target\spring-
petclinic-4.2.5-SNAPSHOT]
[INFO] Processing war project
[INFO] Copying webapp resources [d:\jenkins\workspace\PetClinic-Test\src\main\webapp]
[INFO] Webapp assembled in [1669 msecs]
[INFO] Building war: d:\jenkins\workspace\PetClinic-Test\target\spring-petclinic-4.2.5-SNAPSHOT.war
[INFO] ------------------------------------------------------------------------
[INFO] BUILD SUCCESS
[INFO] ------------------------------------------------------------------------
[INFO] Total time: 28.772 s
[INFO] Finished at: 2016-07-06T22:59:37+05:30
[INFO] Final Memory: 29M/261M
[INFO] ------------------------------------------------------------------------
Deploying d:\jenkins\workspace\PetClinic-Test\target\spring-petclinic-4.2.5-SNAPSHOT.war to container
Tomcat 7.x Remote
  [d:\jenkins\workspace\PetClinic-Test\target\spring-petclinic-4.2.5-SNAPSHOT.war] is not deployed.
Doing a fresh deployment.
  Deploying [d:\jenkins\workspace\PetClinic-Test\target\spring-petclinic-4.2.5-SNAPSHOT.war]
Finished: SUCCESS
```

21. Once build is successful, visit the URL in browser and notice the context. It is similar to name of the Application:



22. In the **Post-build Actions**, give **Context path** and **Save**. Click on build now again:



23. Verify the Application URL by giving new context path.

For deployments, where we can access tomcat-users.xml file in case where we use Tomcat as application container, we will use the same method for deployment. If we don't have direct access to tomcat directory or can't change tomcat-users.xml in such case, another approach can be to ssh the remote host and copy the file into remote host's webapps file of tomcat directory. All ssh commands can be used directly from the build job.

Deploying Application in Docker Container

We have already covered how to use Tomcat with Docker container in Chapter 5, *Installing and Configuring Docker*. To deploy an application with Deploy plugin of Jenkins, we will change tomcat-users.xml. Let's take a step-by-step tour:

1. Change rolename to manager-script in tomcat-users.xml:

```
<?xml version='1.0' encoding='utf-8'?>
<tomcat-users xmlns=”http://tomcat.apache.org/xml”
xmlns:xsi=”http://www.w3.org/2001/XMLSchema-instance”
xsi:schemaLocation=”http://tomcat.apache.org/xml tomcat-users.xsd”
version=”1.0”>
<!–
```

NOTE: The sample user and role entries below are intended for use with the examples web application. They are wrapped in a comment and thus are ignored when reading this file. If you wish to configure these users for use with the examples web application, do not forget to remove the <!....> that surrounds them. You will also need to set the passwords to something appropriate.
–>
<role rolename="manager-script"/>
<user username="admin" password="admin@123" roles="manager-script"/>
</tomcat-users>

2. In Dockerfile, we will copy tomcat-users.xml to /usr/local/tomcat/conf/ directory:

FROM tomcat:8.0
MAINTAINER Mitesh<mitesh.soni@outlook.com>
COPY tomcat-users.xml /usr/local/tomcat/conf/tomcat-users.xml

3. Execute docker build command to create an image:

[root@localhostmitesh]#docker build -t devops_tomcat_sc .
Sending build context to Docker daemon 8.192 kB
Sending build context to Docker daemon
Step 0 : FROM tomcat:8.0
—> 5d4577339b14
Step 1 : MAINTAINER Mitesh<mitesh.soni@outlook.com>
—> Using cache
—> c63f90db4c14
Step 2 : COPY tomcat-users.xml /usr/local/tomcat/conf/tomcat-users.xml
—> aebbcf634f64
Removing intermediate container 7a528d1c8e3b
Successfully built aebbcf634f64
You have new mail in /var/spool/mail/root

4. Verify the newly created image by using docker images command in terminal:

[root@localhostmitesh]#docker images
REPOSITORY TAG IMAGE ID CREATED VIRTUAL SIZE
devops_tomcat_sc latest aebbcf634f64 2 minutes ago 359.2 MB
devopstomcatnew latest eb50c4ceefb5 5 days ago 359.2 MB
devopstomcat8 latest f3537165ebe7 5 days ago 344.6 MB

```
tomcat6 latest 400f097677e9 2 weeks ago 658.4 MB
devopstomcat latest 400f097677e9 2 weeks ago 658.4 MB
centos latest 2a332da70fd1 5 weeks ago 196.7 MB
ubuntu latest 686477c12982 9 weeks ago 120.7 MB
hello-world latest f1d956dc5945 10 weeks ago 967 B
You have new mail in /var/spool/mail/root
```

5. Execute docker run command to create a container:

```
[root@localhostmitesh]#docker run -p 8180:8080 -d –name
devopstomcatscdevops_tomcat_sc
771bb7cb809dabe9323d65579e98077eaec146db4fc38d2ace1d75577144002d
You have new mail in /var/spool/mail/root
```

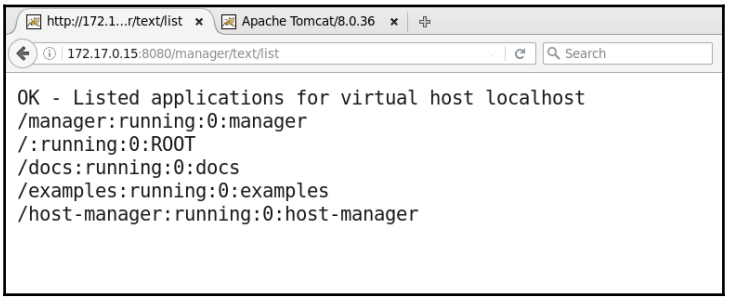
6. Verified the new container with dockerps command:

```
[root@localhostmitesh]#dockerps
CONTAINER ID IMAGE COMMAND CREATED STATUS PORTS
NAMES
771bb7cb809ddevops_tomcat_sc “catalina.sh run” 7 seconds ago Up 6
seconds 0.0.0.0:8180->8080/tcpdevopstomcatsc
```

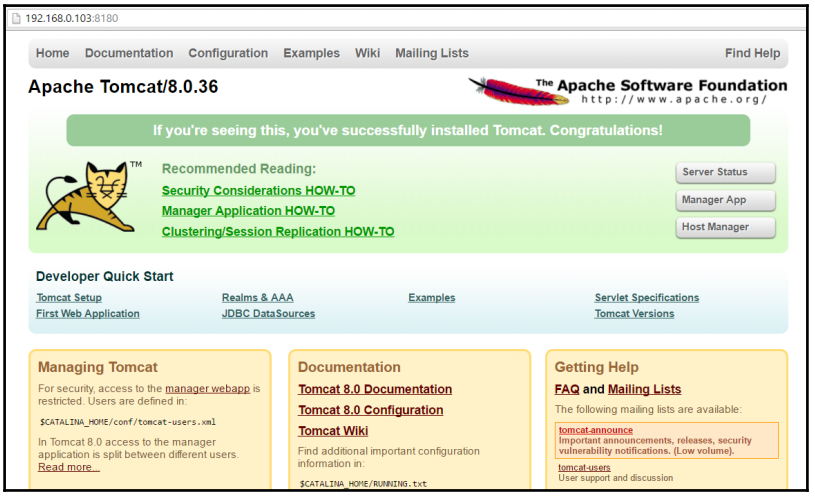
7. Use docker inspect 771bb7cb809d (container id) to get an IP address.
8. Stop IP tables for verification or open ports in IP tables:

```
[root@localhostmitesh]# service iptables stop
iptables: Setting chains to policy ACCEPT: nat filter [ OK ]
iptables: Flushing firewall rules: [ OK ]
iptables: Unloading modules: [ OK ]
You have new mail in /var/spool/mail/root
```

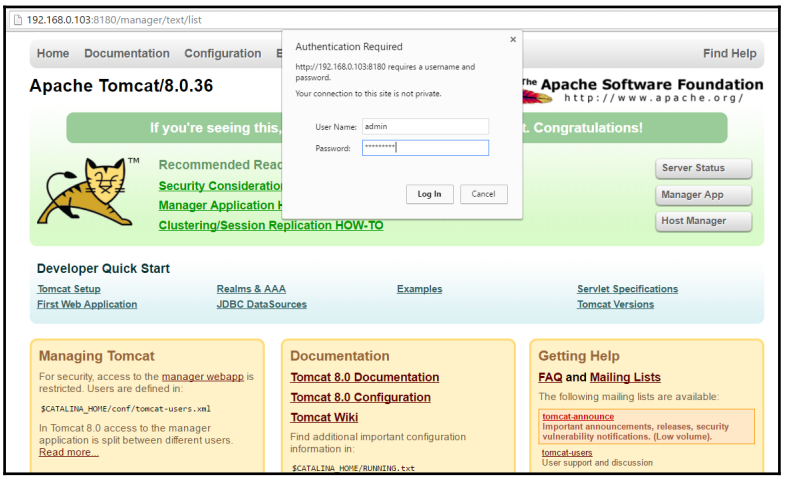
9. Use the IP address and access the manager app URL. Verify whether it is successful or not:



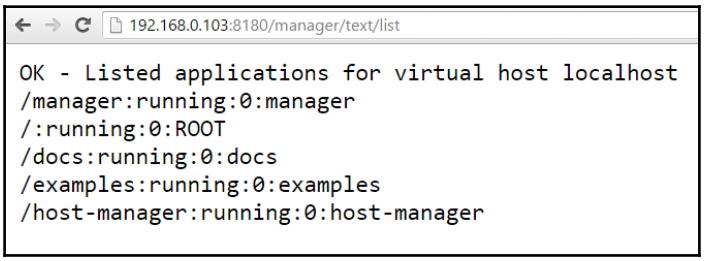
10. As we have mapped port, use host's IP address and verify Tomcat installation:



11. Use the IP address of the host and access the manager app URL. Provide **Username** and **Password**.



12. Verify whether it is successful or not:

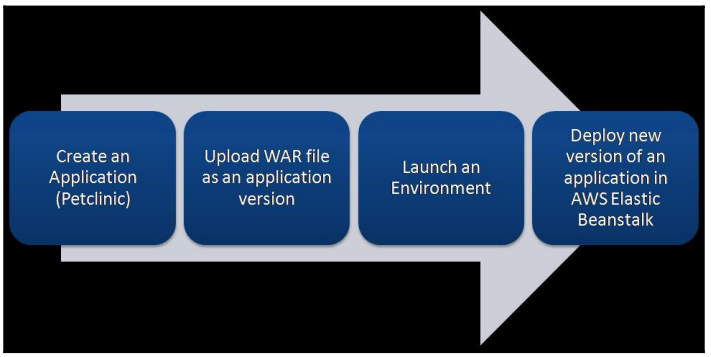


13. Once everything is working fine, use deploy plugin to deploy an application into Docker container.

Deploying Application in AWS

AWS Elastic Beanstalk is a **Platform as a Service (PaaS)** offering from Amazon. We will use AWS Elastic Beanstalk to deploy Petclinic application on the AWS Platform. The good part is we need not to manage infrastructure or even platform as it is a PaaS offering. We can configure scaling and other details.

Following are the steps to deploy an application in AWS Elastic Beanstalk:

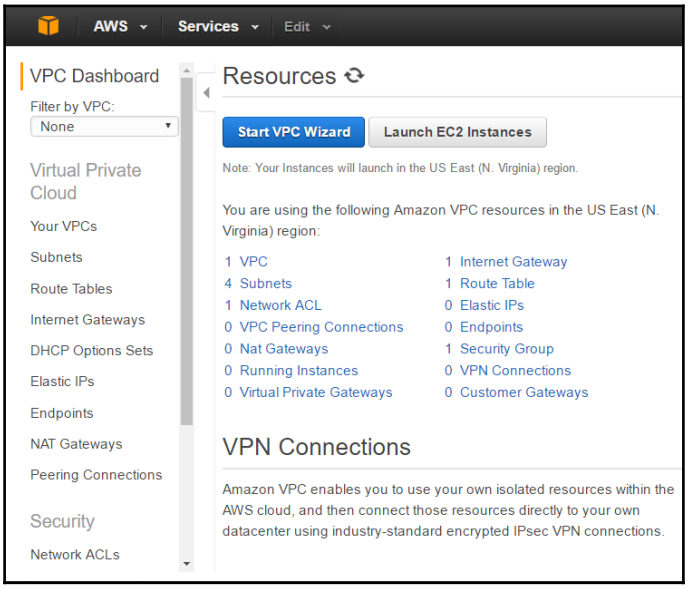


AWS Elastic Beanstalk supports following Programming languages and platforms:

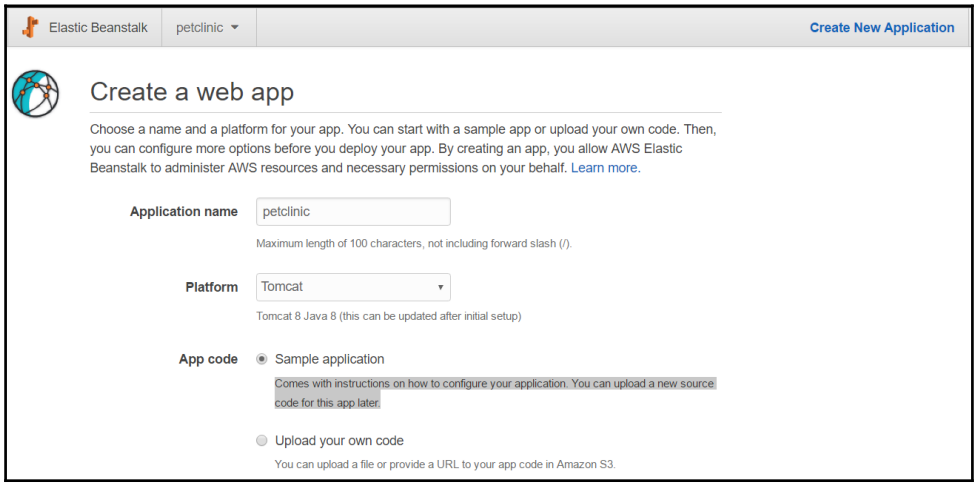
Programming Languages	Platforms
• Java, PHP, Python, Ruby, Go	• Go, Java SE, Java with Tomcat, .NET on Windows Server with IIS, Node.js, PHP, Python, Ruby

Let's create a sample application to understand how AWS Elastic Beanstalk works and then use Jenkins plugin to deploy an application:

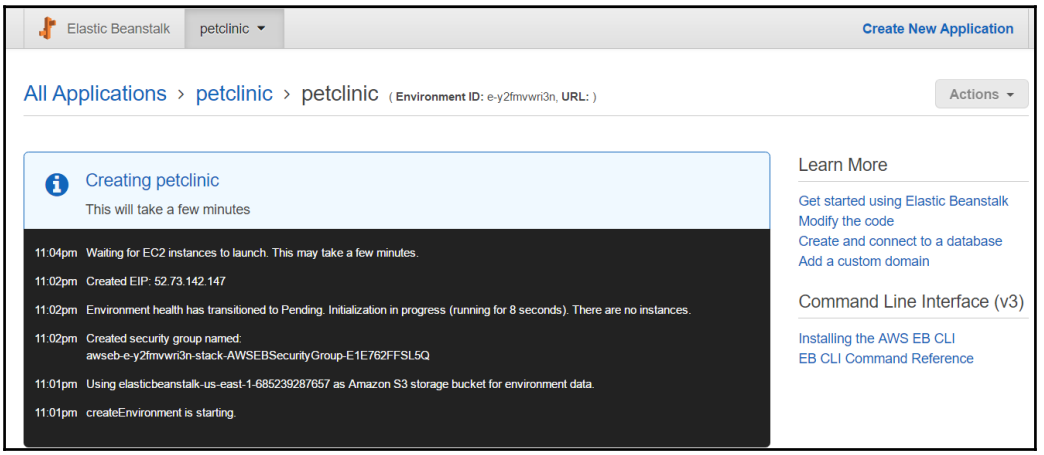
1. Go to AWS Management console and verify whether we have a default VPC or not. If by mistake you have deleted default VPC and subnet, then send request to AWS Customer support to recreate it:



2. Click on the **Services** in AWS Management Console and select **AWS Elastic Beanstalk**. Create a new application named petclinic. Select Tomcat as a platform Sample Application:

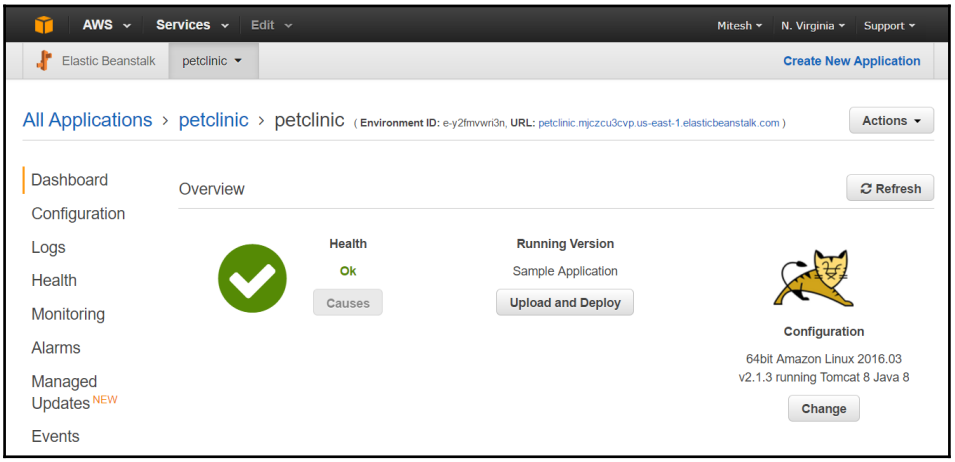


3. Verify the sequence of events for the creation of sample application:

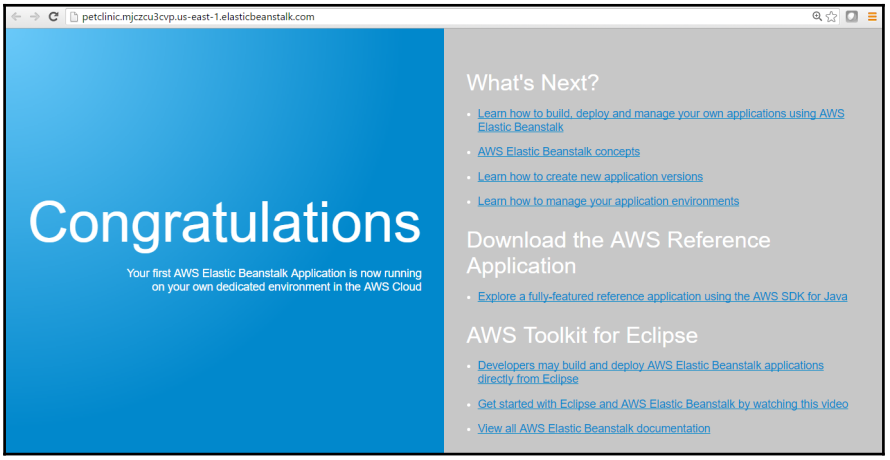


4. It will take some time and once Environment is created it will be in green color as shown below:

5. Click on the petclinic environment and verify the **Health** and **Running Version** on the Dashboard:



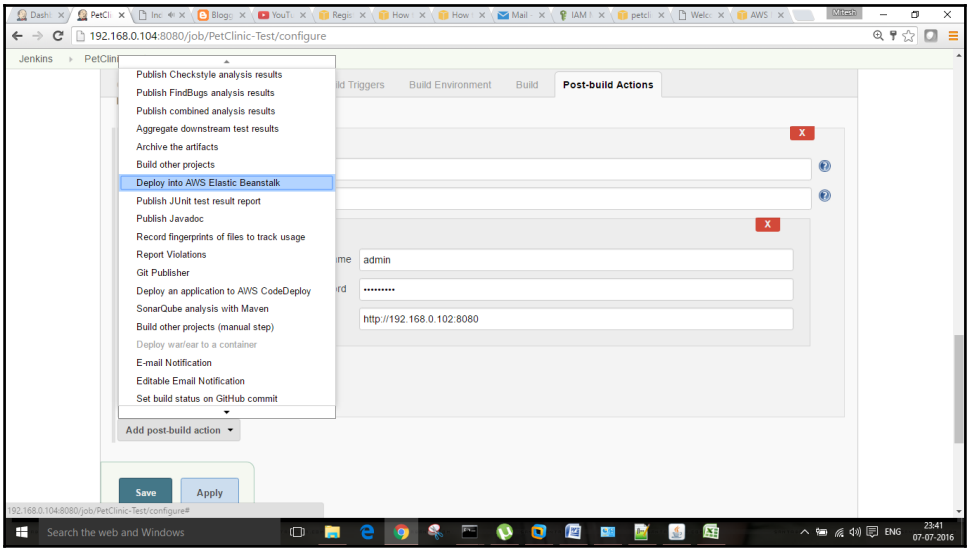
6. Verify the Environment ID and URL. Click on the URL and verify the default page:



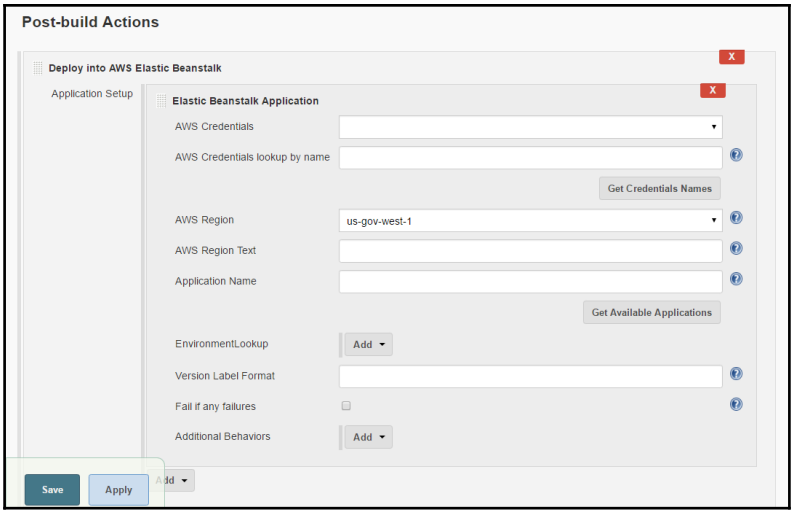
7. Install Amazon Web Services Elastic Beanstalk Publisher.

For more details, visit https://wiki.jenkins-ci.org/display/JENKINS/AWS+Beanstalk+Publisher+Plugin.

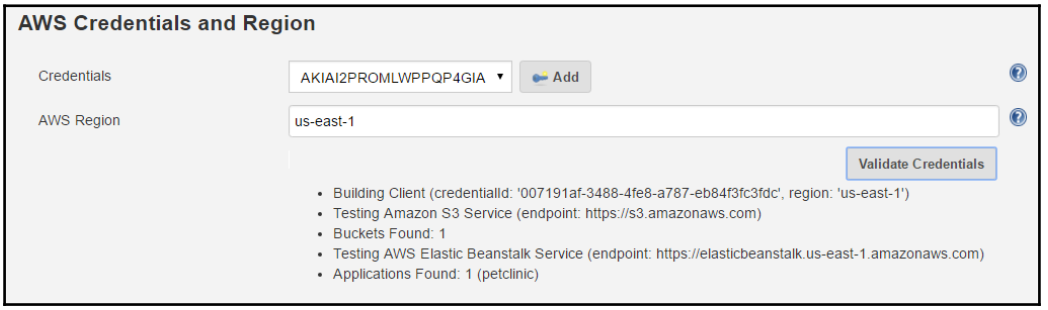
8. Open the Jenkins Dashboard and go to the **Build job**. Click on the **Post build action** and select **Deploy into AWS Elastic Beanstalk**.



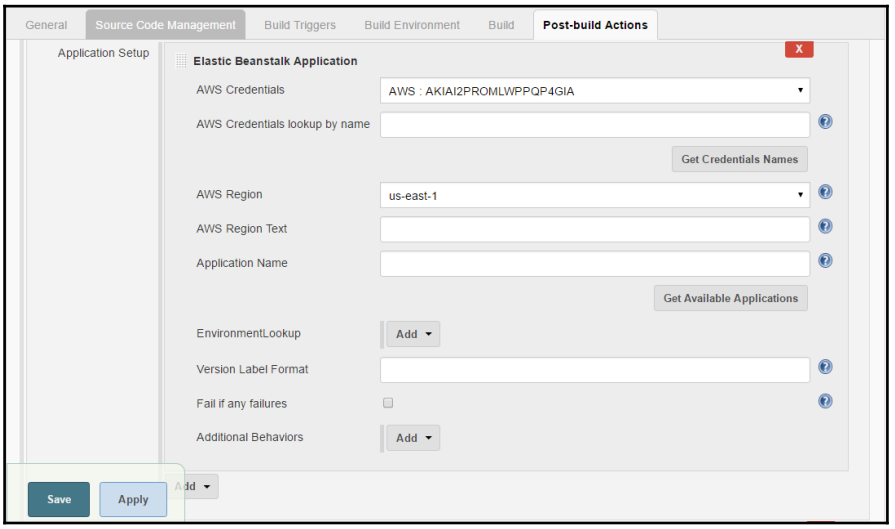
9. New section comes up in the **Post-build Actions** for AWS Elastic Beanstalk:



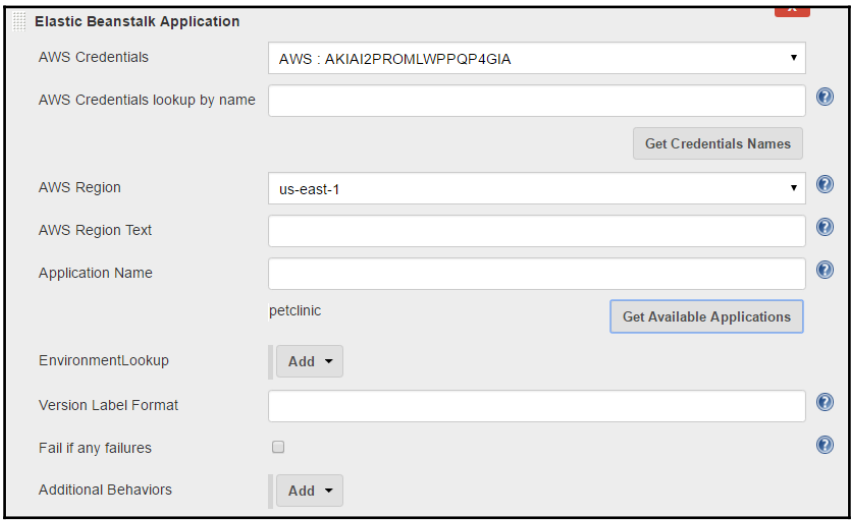
10. Click on Jenkins Dashboard and select **Credentials** and Add AWS credentials:



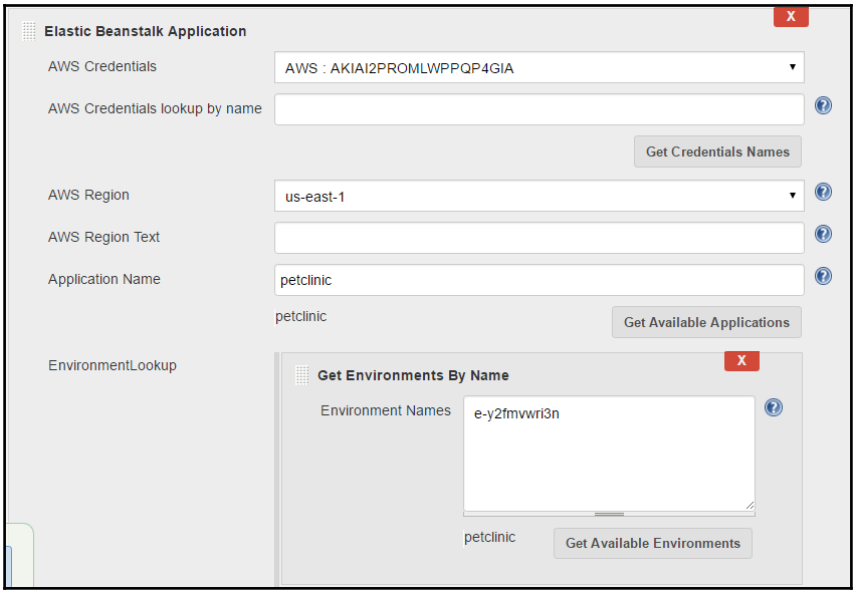
11. Go to Jenkins build and select an **AWS credentials** which is set in the global configuration:



12. Select **AWS Region** from the list and click on the **Get Available Applications**. As we have created a sample application, it will show up:



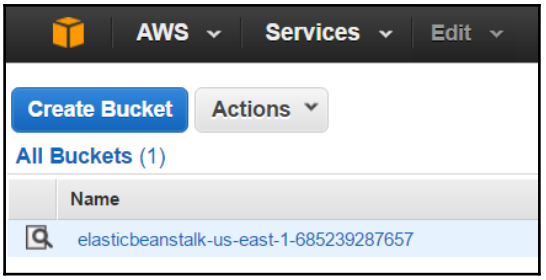
13. In the **EnvironmentLookup**, provide Environment ID in the Get Environments By Name and click on the **Get Available Environments**.



14. Save the configuration and click on Build now.

Let's verify the AWS Management Console:

1. Go to S3 services and verify the available buckets:



As WAR file is having large size, it will take some time to upload on the Amazon S3. Once it is uploaded, it will be available in the Amazon S3 bucket.

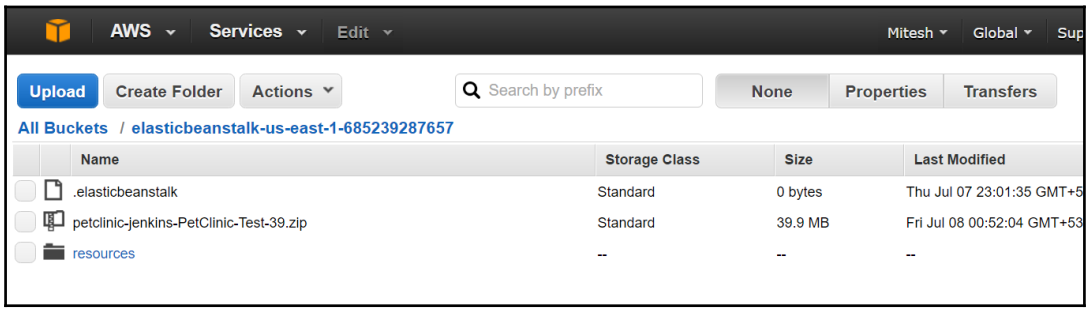
2. Verify the Build job execution status in Jenkins. Some section of output is given below with explanation.

Test case execution and WAR file creation is successful:

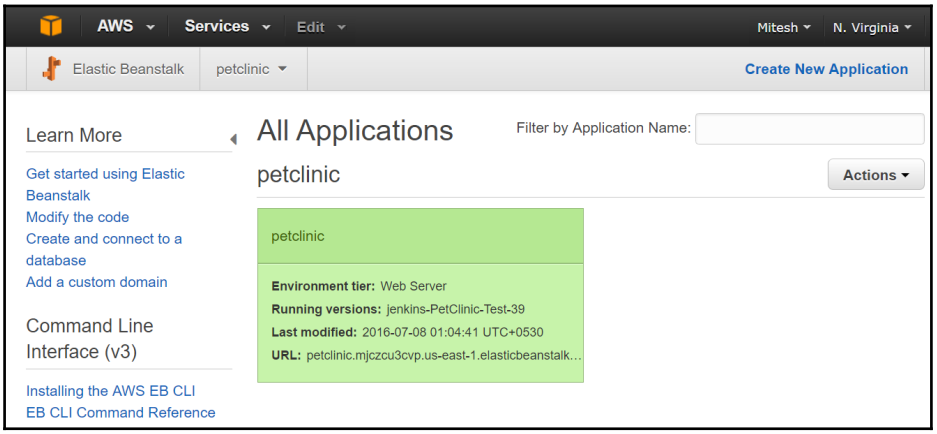
```
Tests run: 59, Failures: 0, Errors: 0, Skipped: 0

[INFO]
[INFO] — maven-war-plugin:2.3:war (default-war) @ spring-petclinic —
[INFO] Packaging webapp
[INFO] Assembling webapp [spring-petclinic] in
[d:\jenkins\workspace\PetClinic-Test\target\spring-petclinic-4.2.5-
SNAPSHOT]
[INFO] Processing war project
[INFO] Copying webapp resources [d:\jenkins\workspace\PetClinic-
Test\src\main\webapp]
[INFO] Webapp assembled in [1539 msecs]
[INFO] Building war: d:\jenkins\workspace\PetClinic-
Test\target\spring-petclinic-4.2.5-SNAPSHOT.war
[INFO] ————————————————————————————————
[INFO] BUILD SUCCESS
[INFO] ————————————————————————————————
[INFO] Total time: 30.469 s
```

```
[INFO] Finished at: 2016-07-08T00:51:52+05:30
[INFO] Final Memory: 29M/258M
[INFO] ————————————————————————————————
Execution of AWSEB Deployment Plugin / Post build action is started:
AWSEB Deployment Plugin Version 0.3.10
Root File Object is a file. We assume its a zip file, which is okay.
bucketName not set. Calling createStorageLocation
Using s3 Bucket 'elasticbeanstalk-us-east-1-685239287657'
Uploading file awseb-5081374840514488317.zip as s3://elasticbeanstalk-us-
east-1-685239287657/petclinic-jenkins-PetClinic-Test-39.zip
Deployment activity with new Version Label will start:
Creating application version jenkins-PetClinic-Test-39 for application
petclinic for path s3://elasticbeanstalk-us-east-1-685239287657/petclinic-
jenkins-PetClinic-Test-39.zip
Created version: jenkins-PetClinic-Test-39
Using environmentId 'e-y2fmvwri3n'
No pending Environment Updates. Proceeding.
Checking health/status of environmentId e-y2fmvwri3n attempt 1/30
Environment Status is 'Ready'. Moving on.
Updating environmentId 'e-y2fmvwri3n' with Version Label set to
'jenkins-PetClinic-Test-39'
Environment status is updated and Health status is updated along with
Deployment status:
Fri Jul 08 01:03:10 IST 2016 [INFO] Environment update is starting.
Checking health/status of environmentId e-y2fmvwri3n attempt 1/30
Versions reported: (current=jenkins-PetClinic-Test-39, underDeployment:
jenkins-PetClinic-Test-39). Should I move on? false
Environment Status is 'Ready' and Health is 'Green'. Moving on.
Deployment marked as 'successful'. Starting post-deployment cleanup.
Cleaning up temporary file
C:\Users\Mitesh\AppData\Local\Temp\awseb-5081374840514488317.zi
p
Finished: SUCCESS
```

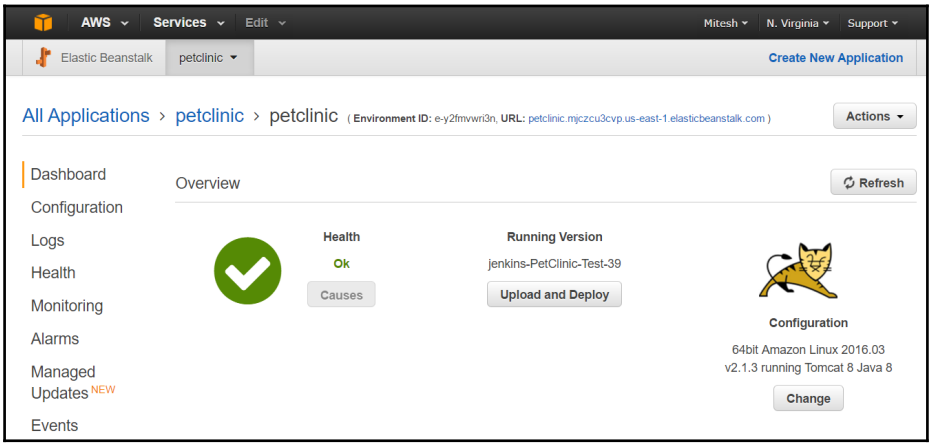
3. Build is successful and now verify the AWS Management console:



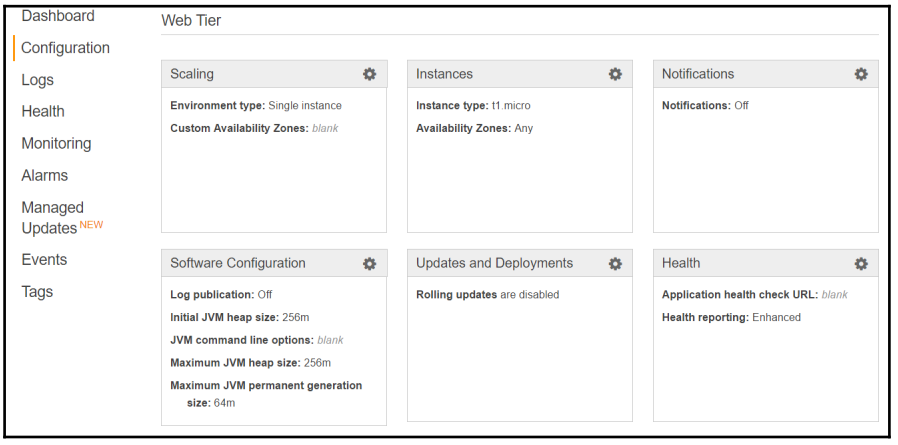
4. Go **to Services**, click on **AWS Elastic Beanstalk** and verify the Environment. Earlier Running versions was Sample Application, now the version is updated as given in Version Label Format in Jenkins build job configuration:



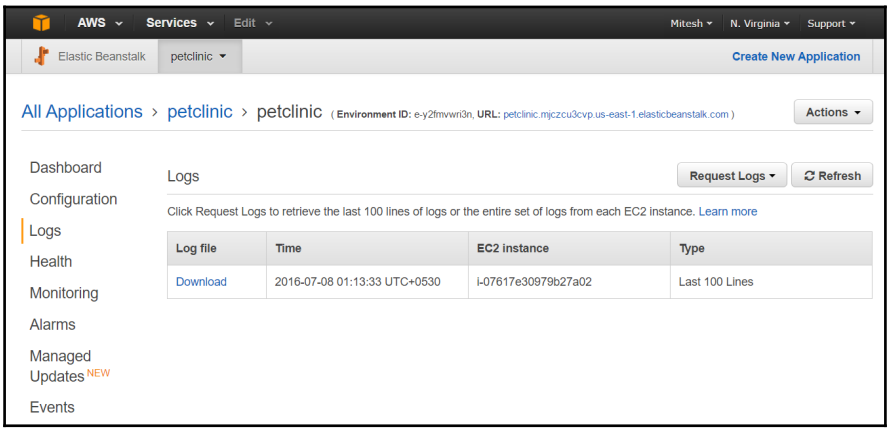
5. Go to Dashboard and verify **Health** and **Running Version** again:



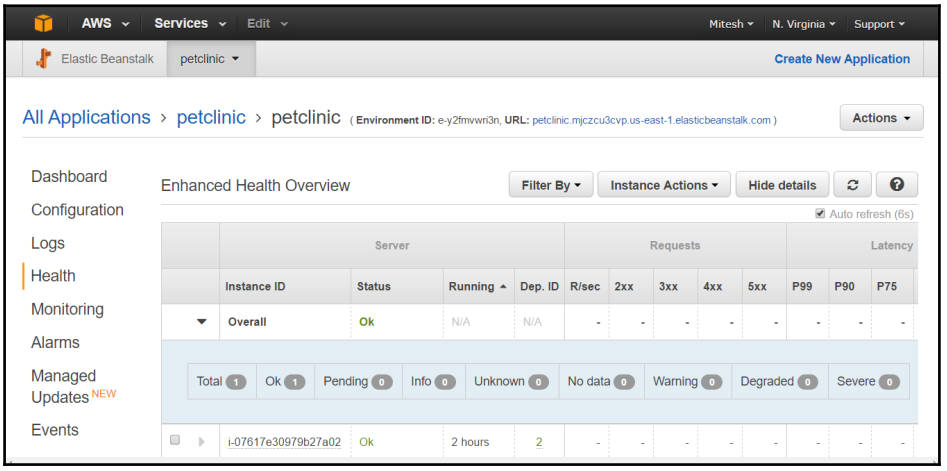
6. Click on the **Configuration** link on AWS Elastic Beanstalk Dashboard and verify **Scaling**, **Instances**, **Notifications**, **Software Configuration**, **Updates and Deployments**, **Health** and so on.



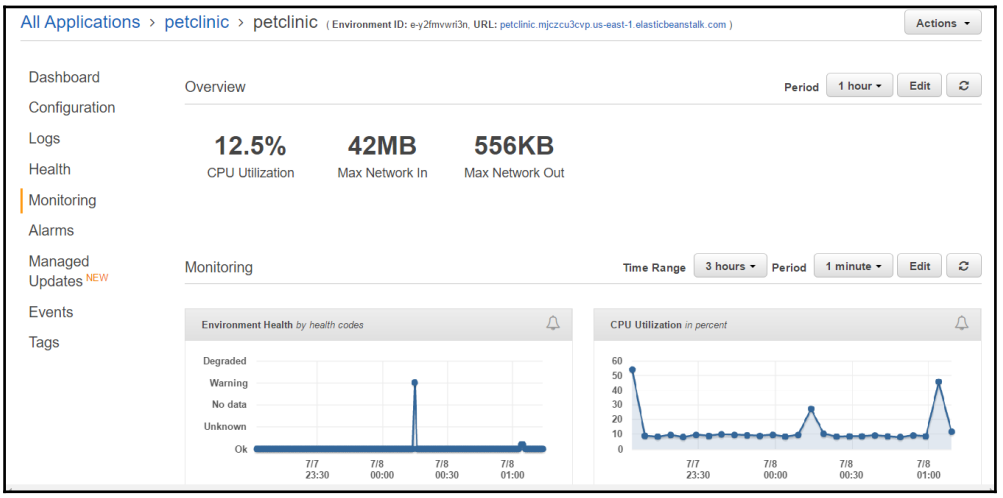
7. Click on **Logs** to download the log files for AWS Elastic Beanstalk application:



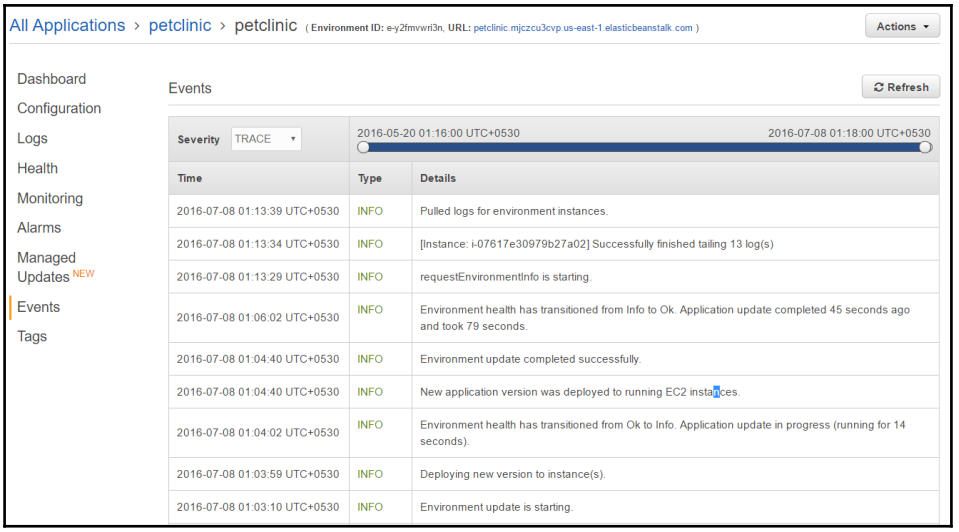
8. Verify the **Enhanced Health Overview** and check the status:



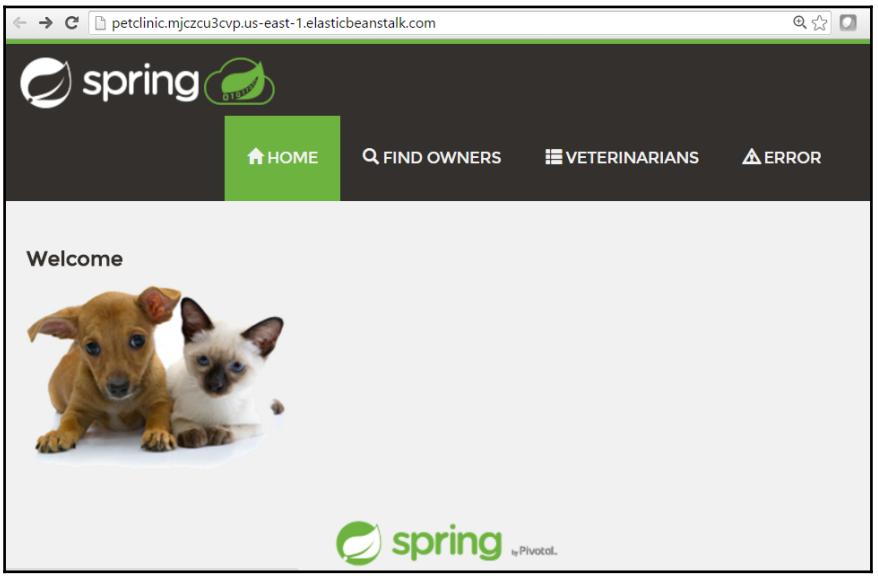
9. Click on the **Monitoring** for extensive monitoring details in form of CPU Utilization and Health of an application:



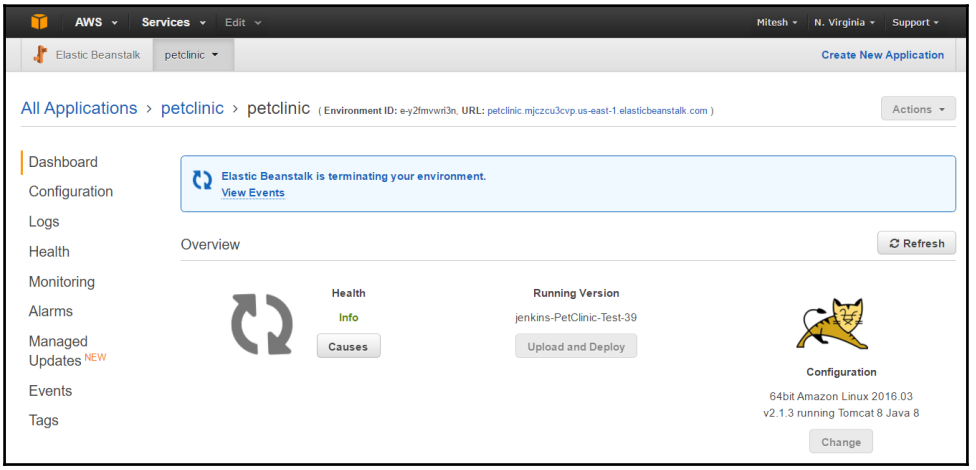
10. Click on **Events** to get list of all events of AWS Elastic Beanstalk application lifecycle:



11. Once, all is verified, click on the URL for the environment and our Petclinic Application is live:



12. Once application deployment is successful then terminate the environment:



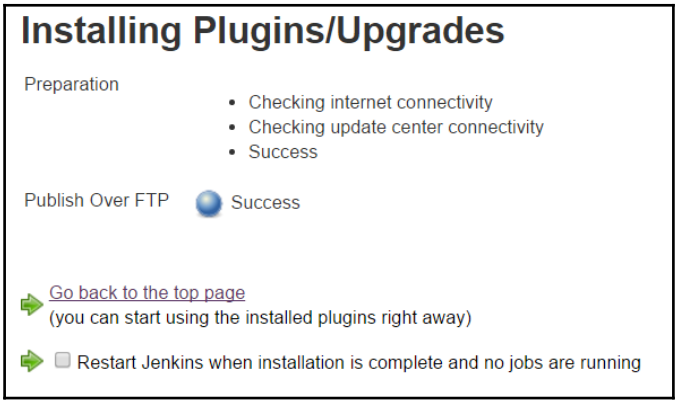
Thus, we have successful application deployment in AWS Elastic Beanstalk.

Deploying Application in Microsoft Azure

Microsoft Azure App Services is a Platform as a Service. In this section we will introduce Azure Web App and how we can deploy Petclinic application:

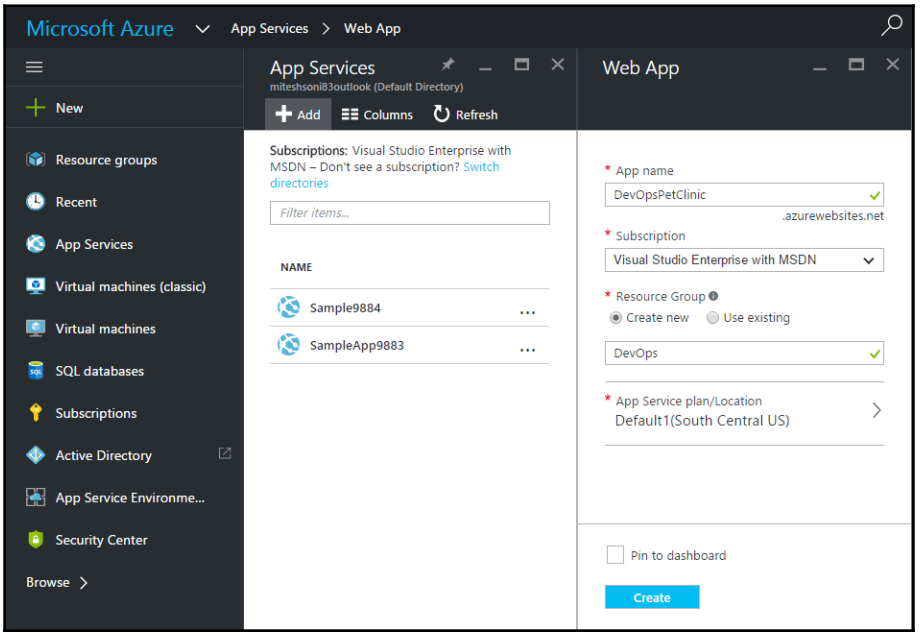
1. Let's install Publish Over FTP plugin in Jenkins. We will use Azure Web App's FTP details to publish Petclinic war file:

2. The plugin is installed successfully in the restart the Jenkins:

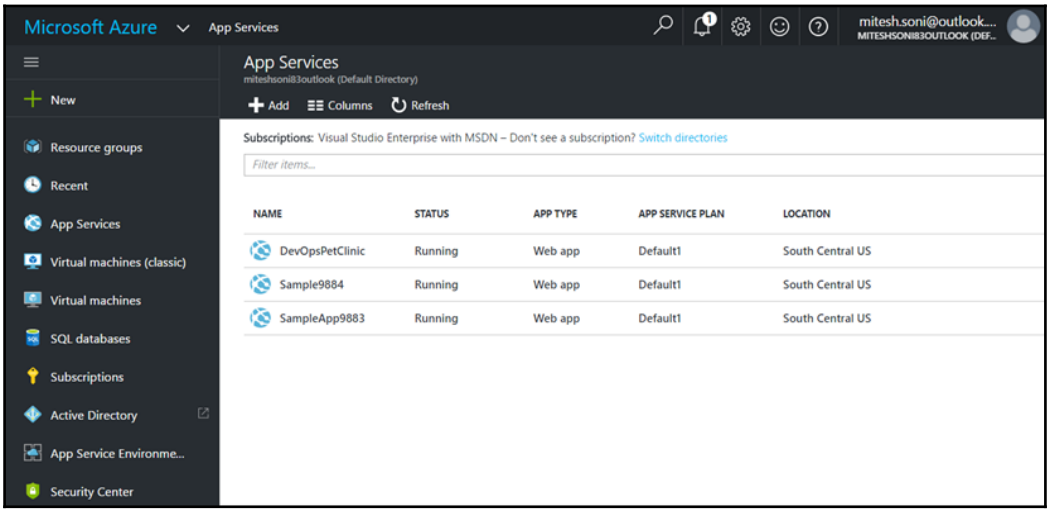


3. Go to Microsoft Azure Portal at https://portal.azure.com. Click on the **App Services** and click on the **Add**. Provide inputs for Name of Azure Web App,

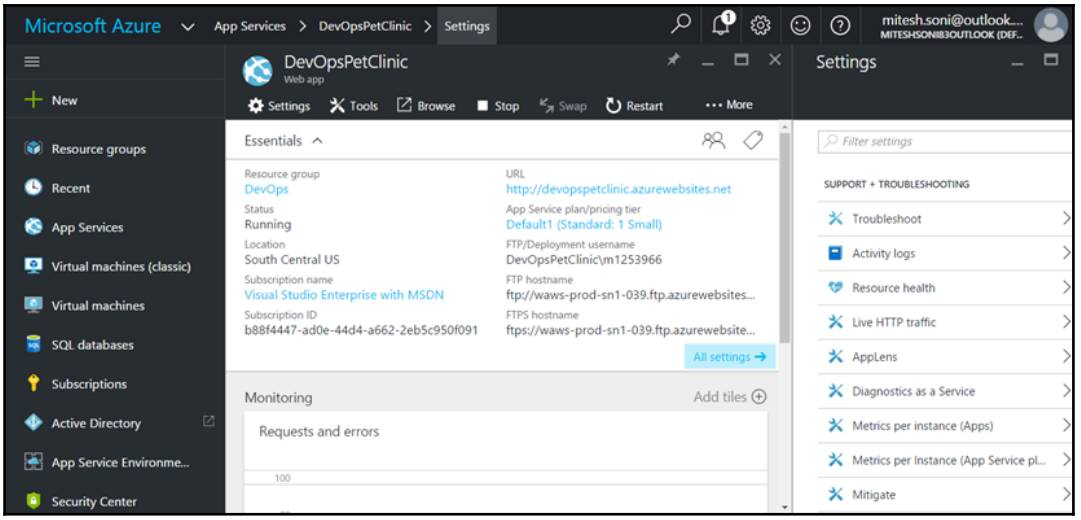
Subscription, **Resource Group**, and **App Service plan/Location**. Click on **Create**:



4. Once Azure Web App is created, verify it in Azure Portal:

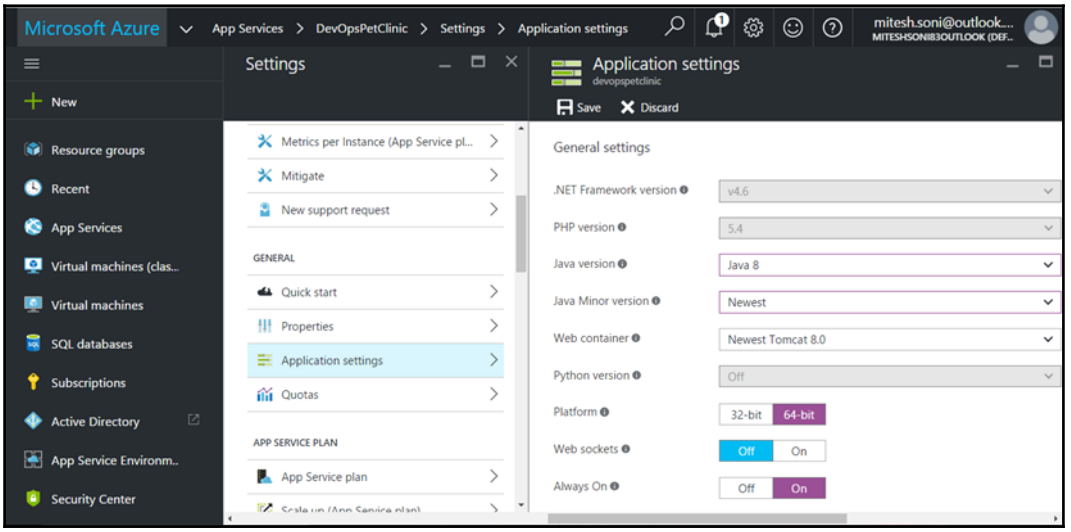


5. Click on the **DevOpsPetClinic**, get the details related to URL, Status, Location, and so on:

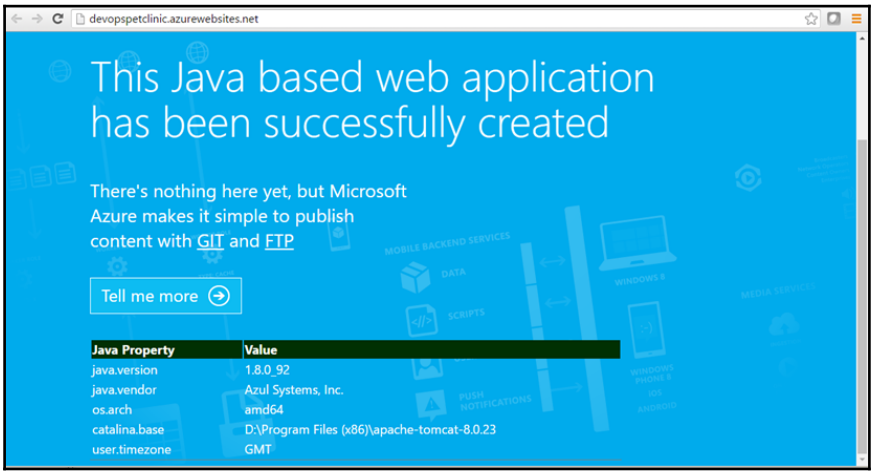


6. Click on All Settings, go to **GENERAL** Section and click on **Application settings** to configure Azure Web App for Java Web Application hosting. Select the **Java**

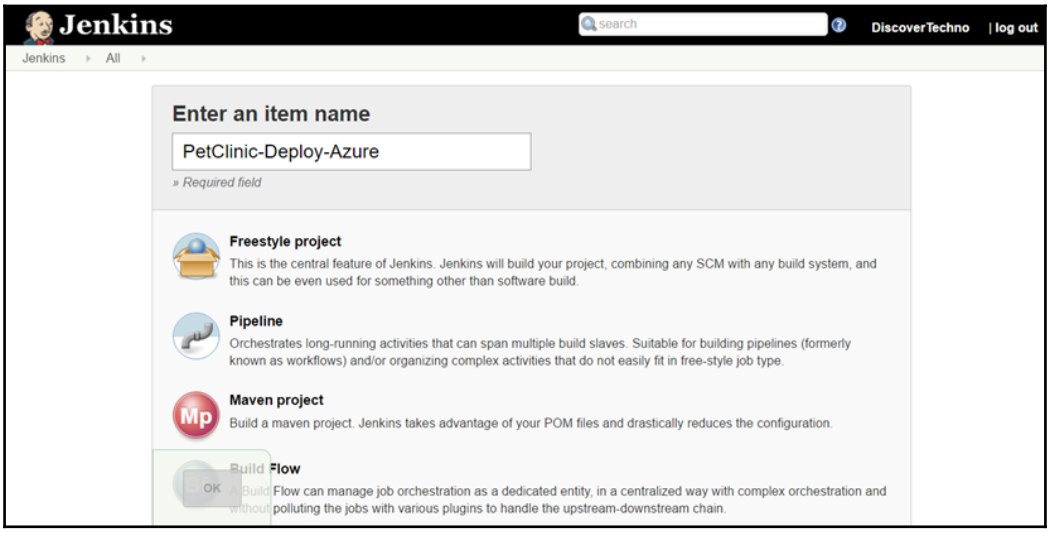
version, **Java Minor version**, **Web container**, **Platform**, and click on **Always On**:



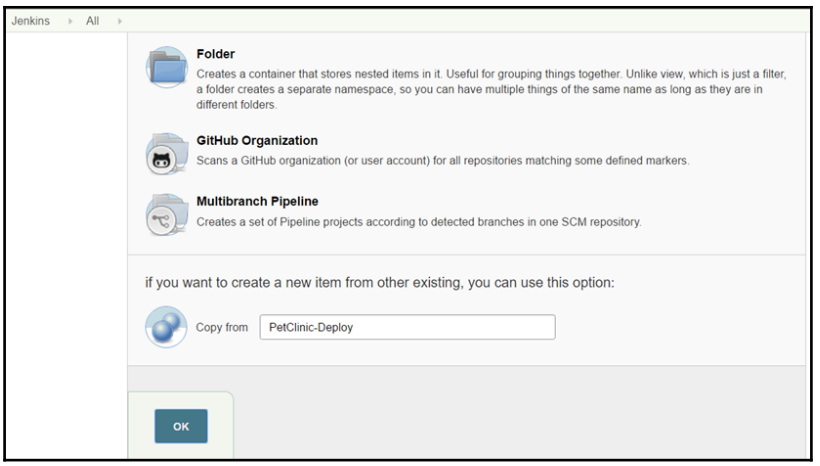
7. Visit the URL of an Azure Web App in the browser and verify whether it is ready for hosting Sample Spring application that is PetClinic.



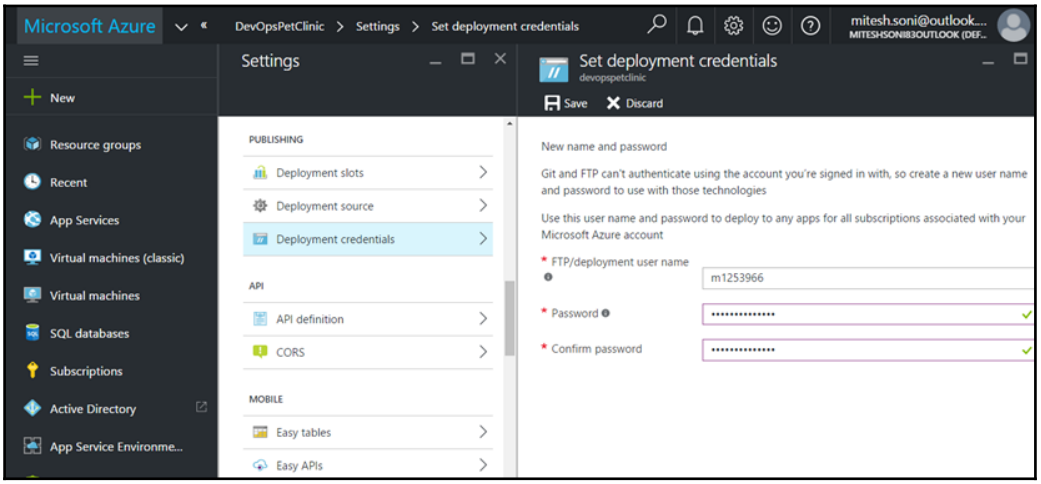
8. Let's go to Jenkins dashboard. Click on **New Item** and select **Freestyle project**:



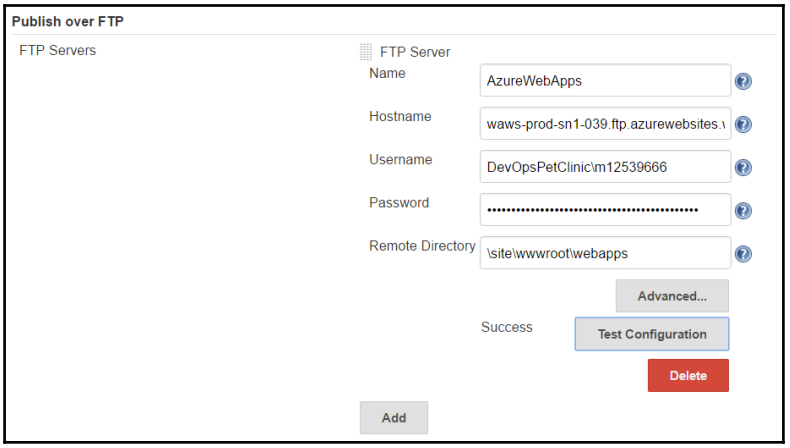
9. Copy general configuration from another build so we need not to repeat the configuration work in newly created job:



10. Click on All Settings, go to **Deployment credentials** in **PUBLISHING** section. Give user name and password. Save it:

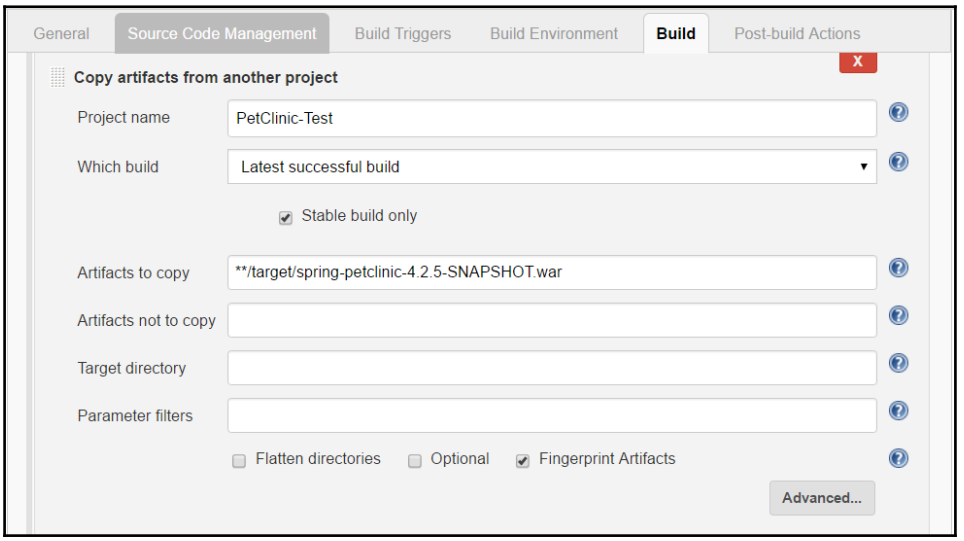


11. In Jenkins, go to **Manage Jenkins** and click on **Configure**. Configure FTP settings. Provide Hostname, Username and Password available in Azure Portal.
12. Go to devopspetclinic.scm.azurewebsites.net and get the Kudu console. Navigate to different options and find the site directory and webapps directory. Click on the **Test Configuration** and once you get Success message, we are ready to deploy our PetClinic application:

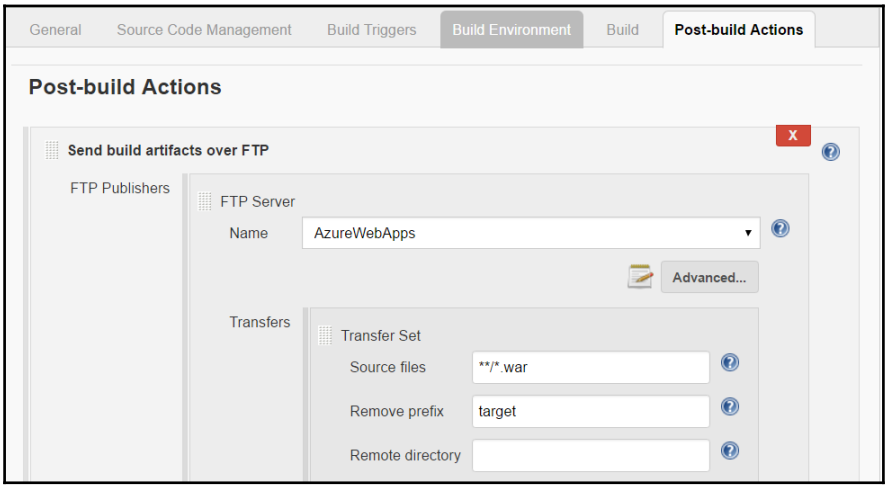


13. In the build job we created, go to **Build** section and configure **Copy artifacts from**

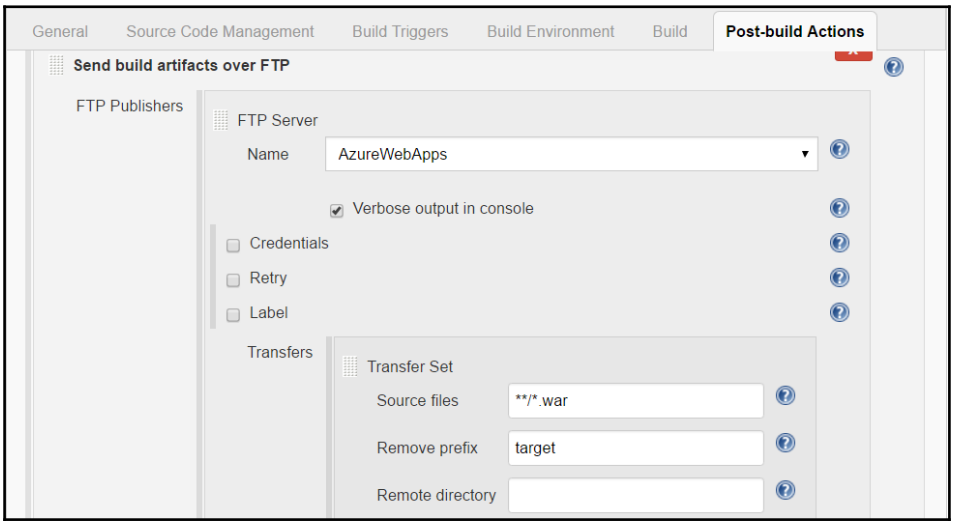
another project. We will copy war file into specific location on a virtual machine:



14. In **Post-build Actions**, click on **Send build artifacts over FTP**. Select **FTP server** name configured in Jenkins. Configure **Source files** and suffix to remove while deployment of an application in Azure Web App:



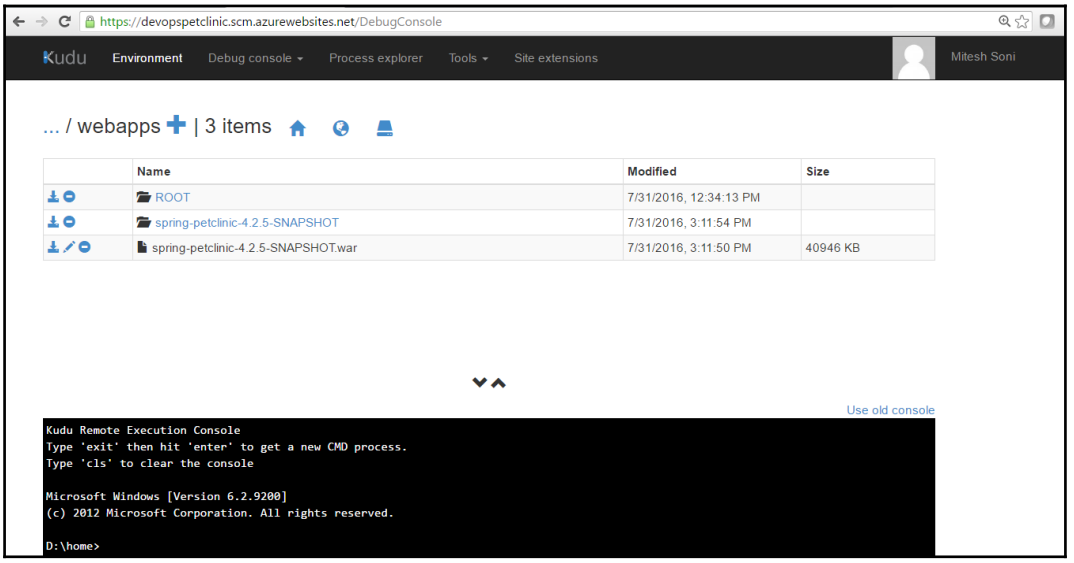
15. Click on the **Verbose output in console**:



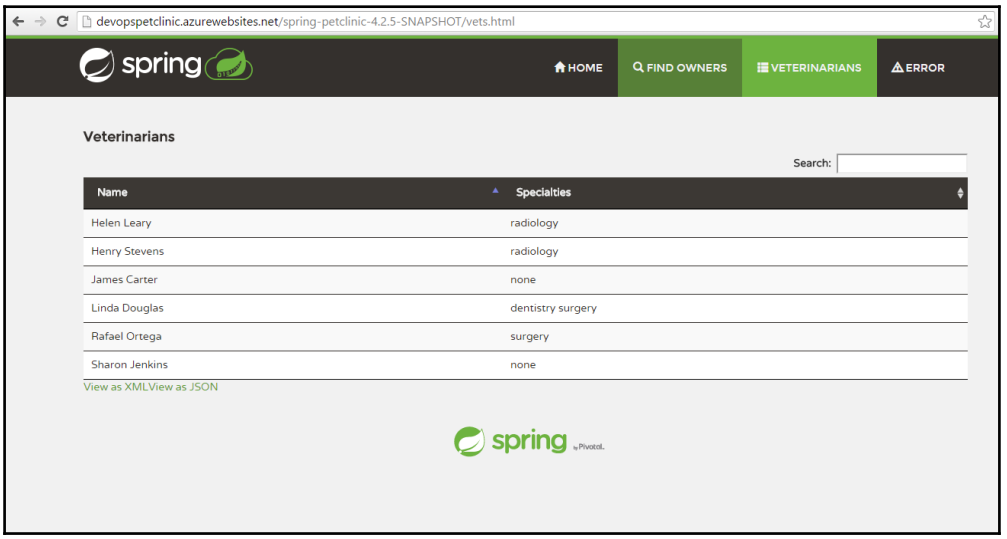
16. Click on **build now** and see what happens behind the seen:

```
Started by user DiscoverTechno
Building on master in workspace /home/mitesh/.jenkins/workspace/PetClinic-Deploy-Azure
Copied 1 artifact from "PetClinic-Test" build number 55
FTP: Connecting from host [devops1]
FTP: Connecting with configuration [AzureWebApps] ...
220 Microsoft FTP Service
FTP: Logging in, command printing disabled
FTP: Logged in, command printing enabled
CWD \site\wwwroot\webapps
250 CWD command successful.
TYPE I
200 Type set to I.
CWD \site\wwwroot\webapps
250 CWD command successful.
PASV
227 Entering Passive Mode (104,210,159,39,39,189).
STOR spring-petclinic-4.2.5-SNAPSHOT.war
125 Data connection already open; Transfer starting.
FTP: Disconnecting configuration [AzureWebApps] ...
```

17. Go to Kudu console, click on **Debug console** and go to **Powershell**. Go to **site | wwwroot | webapps**. Verify whether war file is copied or not:



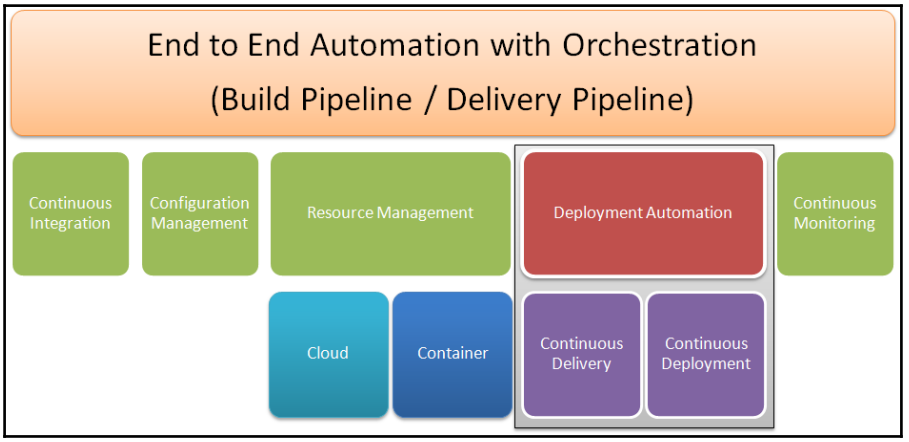
18. Visit the Azure Web App URL in the browser with the context of an application:



So we have an application deployed on Azure Web Apps.

It is important to note that FTP user name has to be with the domain. In our case, it can be Sample9888\m1253966. Direct user name without Web App name won't work.

All this different ways of deployment into AWS IaaS, AWS PaaS, Microsoft Azure PaaS, and Docker container can be used in final end to end automation:



We have covered four phases till now and now we will discuss about Continuous Monitoring and in the last Chapter we will manage all end to end automation with pipeline or orchestration.

Self-Test Questions

1. State True or False: Role and Users in Tomcat can be created in tomcat-users.xml to access Manager Web App
2. True
3. False
4. State True or False: To access Tomcat Manager App GUI Manager-script role is required.
5. True
6. False

1. State True or False: To deploy application in Tomcat container using Deploy

Plugin in Jenkins, Manager-script role is required.
2. True
3. False

1. State True or False: AWS Elastic Beanstalk and Azure App Services are a Platform as a Service (PaaS) offering from Amazon and Microsoft respectively.
2. True
3. False

1. Which of the following are steps for application deployment in AWS Elastic Beanstalk?
2. Create an Application (Petclinic)
3. Upload WAR file as an application version
4. Launch an Environment
5. Deploy new version of an application in AWS Elastic Beanstalk
6. All of the above

Summary

In this chapter, we have covered how to deploy an application in Tomcat using Tomcat Manager Application by setting Role and Users in tomcat-users.xml. We can use same deployment method where we can configure or edit tomcat-users.xml. Same approach was used for Petclinic application deployment in the Docker container.

It is a suitable approach in Infrastructure as a Service. We have also deployed Petclinic application in Platform as a Service such as AWS Elastic Beanstalk and Microsoft Azure Web App.

We have also verified what topics we have covered till now for end to end deployment for Petclinic application.

In the next chapter, we will discuss about Continuous Monitoring for Infrastructure and Application.