

Mitesh Soni

# DevOps for Web Development

Achieve the Continuous Integration and Continuous Delivery of your web applications with ease



Packt

# **DevOps for Web Development**

Achieve the Continuous Integration and Continuous Delivery  
of your web applications with ease

**Mitesh Soni**



BIRMINGHAM - MUMBAI

# **DevOps for Web Development**

Copyright © 2016 Packt Publishing

All rights reserved. No part of this book may be reproduced, stored in a retrieval system, or transmitted in any form or by any means, without the prior written permission of the publisher, except in the case of brief quotations embedded in critical articles or reviews.

Every effort has been made in the preparation of this book to ensure the accuracy of the information presented. However, the information contained in this book is sold without warranty, either express or implied. Neither the author, nor Packt Publishing, and its dealers and distributors will be held liable for any damages caused or alleged to be caused directly or indirectly by this book.

Packt Publishing has endeavored to provide trademark information about all of the companies and products mentioned in this book by the appropriate use of capitals. However, Packt Publishing cannot guarantee the accuracy of this information.

First published: October 2016

Production reference: 1171015

Published by Packt Publishing Ltd.

Livery Place  
35 Livery Street  
Birmingham  
B3 2PB, UK.

ISBN 978-1-78646-570-2

[www.packtpub.com](http://www.packtpub.com)

# Credits

**Author**  
Mitesh Soni

**Copy Editor**  
Madhusudan Uchil

**Reviewer**  
Allan Espinosa

**Project Coordinator**  
Judie Jose

**Commissioning Editor**  
Pratik Shah

**Proofreader**  
Safis Editing

**Acquisition Editor**  
Smeet Thakkar

**Indexer**  
Pratik Shirodkar

**Content Development Editor**  
Amedh Gemraram Pohad

**Graphics**  
Kirk D'Penha

**Technical Editor**  
Vishal Kamal Mewada

**Production Coordinator**  
Deepika Naik

# About the Author

**Mitesh Soni** is an avid learner with 9 years' experience in the IT industry. He is an SCJP, SCWCD, VCP, and IBM Urbancode certified professional. He loves DevOps and cloud computing, and also has an interest in programming in Java. He finds design patterns fascinating. He occasionally contributes to <http://etutorialsworld.com>. He loves to play with kids, fiddle with his camera, and capture photographs at Indroda Park. He is addicted to taking good pictures without knowing many technical details. He lives in the capital of Mahatma Gandhi's home state.

*"I've missed more than 9000 shots in my career. I've lost almost 300 games. 26 times, I've been trusted to take the game-winning shot and missed. I've failed over and over and over again in my life. And that is why I succeed"*

– Michael Jordan

# Acknowledgments

To my...wife? (I am not married.)

And my...children? (Read the previous sentence.)

...without whom this book has been completed within 3-4 months. (Else it might have taken a year or two—pun intended!)

On a serious note, I would like to dedicate this book to the kid who taught me to live life freely. Shreyu (Shreyansh, my sister Jigisha's baby boy) showed me the power of innocence and smiles. I've had a completely different perspective of life since he has arrived.



Special thanks to Priyanka Agashe for supporting and encouraging me all the time. Please don't overrate me as a person (all sisters do that). Sorry for being *khadoos*. I would also like to dedicate this book to my father, who is an *avid* reader. He loves books so much that he reads these technical books and notes down all the quotes at the beginning of each chapter. I want to say thanks and share my gratitude for everything I've been blessed with.

I would like to thank my parents, Jigisha and Nitesh, *dada* and *dadi*, Vinay Kher, my teachers, friends, family members, Aakanksha "Akkus" Deshpande (thanks Mother India for always telling me "*koshish karne valo ki haar nahi hoti*"), Hemant and Priyanka, Mihir P and Anupama S, Yohan Wadia, Jyoti-Kanika Bhatia (you always remember special occasions *Jyotiben*), Rohini Gaonkar, Rohan C, Mayur Mothliya, Chintan Solaki, Navrang O, Dharmesh R, and Ashish B.

I am also thankful to Palak S, Subhrajyoti M, Siddharth B, Nirali Kotak, Sumukh, Bijal, Ragni, Beena, Arpan V, Parth S, Bibhas S, Paresh P, Nirav V, Vimal K, Paras Shah, Vishal R, Sharvil P, Sourabh M, Viral I, Vijay Y, Amit R, Manisha Y, Gowri, Saurabh S, Nishchal S, and Kushal V, who have always helped me and made my life easier at specific points in the past year or so. I'm not sure we will ever meet again in life, so I'm trying to thank all those who helped me, knowingly or unknowingly. Apologies if I have missed any names.

# About the Reviewer

**Allan Espinosa** is a DevOps practitioner living in Tokyo. He is an active open source contributor to various distributed systems tools, such as Docker and Chef. He maintains several Docker images for popular open source software that were popular even before the official release from the upstream open source groups themselves. In his career, Allan has worked on large distributed systems containing hundreds to thousands of servers in production. He has built scalable applications on various platforms, ranging from large supercomputing centers in the US to production enterprise systems in Japan. Allan can be contacted through his Twitter handle @AllanEspinosa. His personal website at <http://aespinoza.github.io> contains several blog posts on Docker and distributed systems in general.

*I would like to thank my wife, Kana, for her continuous support, which allowed me to spend significant time with this review project.*

# [www.PacktPub.com](http://www.PacktPub.com)

For support files and downloads related to your book, please visit [www.PacktPub.com](http://www.PacktPub.com).

Did you know that Packt offers eBook versions of every book published, with PDF and ePub files available? You can upgrade to the eBook version at [www.PacktPub.com](http://www.PacktPub.com) and as a print book customer, you are entitled to a discount on the eBook copy. Get in touch with us at [service@packtpub.com](mailto:service@packtpub.com) for more details.

At [www.PacktPub.com](http://www.PacktPub.com), you can also read a collection of free technical articles, sign up for a range of free newsletters and receive exclusive discounts and offers on Packt books and eBooks.



<https://www.packtpub.com/mapt>

Get the most in-demand software skills with Mapt. Mapt gives you full access to all Packt books and video courses, as well as industry-leading tools to help you plan your personal development and advance your career.

## **Why subscribe?**

- Fully searchable across every book published by Packt
- Copy and paste, print, and bookmark content
- On demand and accessible via a web browser

# Table of Contents

<b>Preface</b>	1
<b>Chapter 1: Getting Started – DevOps Concepts, Tools, and Technologies</b>	8
<b>Understanding the DevOps movement</b>	9
DevOps with the changing times	11
The waterfall model	12
The agile model	13
Collaboration	15
Cloud computing – the disruptive innovation	16
Why DevOps?	16
The benefits of DevOps	18
<b>The DevOps lifecycle – it's all about “continuous”</b>	19
Build automation	21
Continuous integration	22
Best practices	24
Cloud computing	25
Configuration management	27
Continuous delivery/continuous deployment	28
Best practices for continuous delivery	30
Continuous monitoring	30
Continuous feedback	32
<b>Tools and technologies</b>	33
Code repositories – Git	33
Advantages	34
Characteristics	34
Differences between SVN and Git	35
Build tools – Maven	38
Example pom.xml file	39
Continuous integration tools – Jenkins	40
Key features and benefits	41
Configuration management tools – Chef	44
Features	46
Cloud service providers	46
Container technology	47
Docker	48
Monitoring tools	49
Zenoss	49
Nagios	50

Deployment orchestration/continuous delivery – Jenkins	50
End-to-end orchestration: Jenkins plugins	51
The DevOps dashboard	52
<b>An overview of a sample Java EE application</b>	53
The list of tasks	55
<b>Self-test questions</b>	55
Summary	58
<b>Chapter 2: Continuous Integration with Jenkins 2</b>	59
<b>Introduction</b>	60
<b>Installing Jenkins</b>	61
Setting up Jenkins	63
<b>The Jenkins dashboard</b>	67
<b>Configuring Java and Maven in Jenkins</b>	69
Configuring Java	69
Configuring Maven	70
<b>Creating and configuring a build job for a Java application with Maven</b>	70
<b>Configuring and authenticating source code on GitHub</b>	72
<b>Configuring build job</b>	77
<b>Configuring JUnit</b>	81
<b>The Dashboard View plugin – overview and usage</b>	83
<b>Managing nodes</b>	86
Creating and configuring slave node in Jenkins 2	87
Configuring the build job for master and slave node	92
<b>Sending e-mail notifications based on build status</b>	94
<b>Integrating Jenkins and Sonar</b>	97
<b>Self-test questions</b>	107
Summary	108
<b>Chapter 3: Building the Code and Configuring the Build Pipeline</b>	109
<b>Creating built-in delivery pipelines</b>	110
<b>Creating scripts</b>	112
Example 1 – creating a Groovy script to build a job	113
Example 2 – creating a build step to publish test reports	114
Example 3 – archiving build job artifacts	115
Example 4 – running a build step on a node	116
Example 5 – marking the definite steps of a build job	117
<b>Creating a pipeline for compiling and executing test units</b>	118
<b>Using the Build Pipeline plugin</b>	127
<b>Integrating the deployment operation</b>	138
<b>Self-test questions</b>	145

<b>Summary</b>	146
<b>Chapter 4: Installing and Configuring Chef</b>	147
<b>Getting started with Chef</b>	148
<b>Overview of hosted Chef</b>	149
<b>Installing and configuring a Chef workstation</b>	156
<b>Converging a Chef node using a Chef workstation</b>	159
<b>Installing software packages using cookbooks</b>	167
<b>Creating a role</b>	170
<b>Self-test questions</b>	174
<b>Summary</b>	176
<b>Chapter 5: Installing and Configuring Docker</b>	177
<b>Overview of Docker containers</b>	178
<b>Understanding the difference between virtual machines and containers</b>	182
<b>Installing and configuring Docker on CentOS</b>	183
<b>Creating your first Docker container</b>	185
<b>Understanding the client-server architecture of Docker</b>	187
<b>Managing containers</b>	192
<b>Creating a Docker image from Dockerfile</b>	198
<b>Self-test questions</b>	204
<b>Summary</b>	204
<b>Chapter 6: Cloud Provisioning and Configuration Management with Chef</b>	205
<b>Chef and cloud provisioning</b>	206
<b>Installing knife plugins for Amazon Web Services and Microsoft Azure</b>	208
<b>Creating and configuring a virtual machine in Amazon EC2</b>	210
<b>Creating and configuring a virtual machine in Microsoft Azure</b>	218
<b>Docker containers</b>	226
<b>Self-test questions</b>	231
<b>Summary</b>	232
<b>Chapter 7: Deploying Application in AWS, Azure, and Docker</b>	233
<b>Prerequisites – deploying our application on Remote Server</b>	234
Setting up Tomcat server	236
<b>Deploying application in Docker container</b>	250
<b>Deploying Application in AWS</b>	254
<b>Deploying application in Microsoft Azure</b>	269
<b>Self-test questions</b>	279

<b>Summary</b>	280
<b>Chapter 8: Monitoring Infrastructure and Applications</b>	281
<b>Getting started – monitoring</b>	282
<b>Overview of Monitoring tools and Techniques</b>	282
<b>Nagios</b>	283
Quick start with Nagios	284
<b>Monitoring AWS Elastic Beanstalk</b>	303
<b>Monitoring Microsoft Azure Web App Service</b>	306
<b>Self-test questions</b>	341
<b>Summary</b>	342
<b>Chapter 9: Orchestrating Application Deployment</b>	343
<b>Creating build jobs for end-to-end automation</b>	344
<b>Configuring SSH authentication using a key</b>	348
<b>Configuring the build pipeline for build job orchestration</b>	366
<b>Executing the pipeline for application deployment automation</b>	383
<b>Hygieia – a DevOps dashboard</b>	386
<b>Self-test questions</b>	387
<b>Summary</b>	388
<b>Index</b>	389

# Preface

DevOps is part of almost every discussion in the project team, sales team, customer engagements, and so on. Yes, it is a Culture but customers are asking for Proof of Concepts of automation that can be utilized in the Application Life Cycle Management. Even though DevOps is in early stage and it is about changing the existing culture that invites resistance, still it is wise to follow what Socrates said:

*"The secret of change is to focus all your energy not on fighting the old, but on building the new."*

The reason behind the culture shift is to keep pace with evolution with ongoing revolution, innovations, and business demands in the highly dynamic and competitive market.

Main objective is to manage frequent releases effectively. The faster you fail, the faster you recover. To fail early is far better than to fail at the end of the phase where roll back is very difficult. By automating repetitive processes, you standardized the management of application lifecycle and avoid error prone manual processes.

In this book, we will cover all the key components of DevOps such as Continuous Integration, Cloud Computing, Configuration Management, Continuous Delivery, and Continuous Deployment; how to automate build integration, provision resources in cloud environment such as AWS and Microsoft Azure, use containers for application deployment, use Chef configuration management tool to set up runtime environment for application deployment; deploying web application into virtual machines configured with Chef, AWS Elastic Beanstalk, Microsoft Azure Web Apps, and Docker containers; application monitoring with Nagios, New Relic, and Native Cloud Monitoring features as well.

For Continuous Integration, we have used Jenkins 2. Orchestration of end to end automation is managed by Pipeline.

Jenkins 2 is aimed to claim Continuous Delivery space also. It brings a new setup experience and interesting UI improvements, and Pipeline as code while maintaining backward compatibility with existing Jenkins installations.

## What this book covers

Chapter 1, *Getting Started—DevOps Concepts, Tools, and Technologies*, gives insights into DevOps movement, challenges for developers team, challenges for operations team, challenges faced by organizations, waterfall and agile model, importance of collaboration, cloud computing, reason to go for DevOps, benefits of DevOps, DevOps lifecycle, build automation, continuous integration and its best practices, configuration management, continuous delivery and continuous deployment and its best practices, continuous monitoring, and continuous feedback. It also covers an overview of code repositories, Maven, Jenkins 2.0, Chef, AWS, Microsoft Azure, Docker, Nagios, Hygieia DevOps Dashboard, overview of Sample JEE application.

Chapter 2, *Continuous Integration with Jenkins 2*, describes in details on overview of continuous integration, Jenkins 2.0 installation, Java and Maven configuration in Jenkins, creating and configuring build job for Java application with Maven, Dashboard View plugin, managing nodes, email notifications based on build status, and Jenkins and Sonar integration

Chapter 3, *Building the Code and Configuring the Build Pipeline*, covers built-in delivery pipelines using a domain-specific language (DSL), Build Pipeline plugin, deploying a WAR file in the web server.

Chapter 4, *Installing and Configuring Chef*, gives insight on Chef configuration management tool, hosted Chef, installing and configuring Chef workstation, and converging Chef node using Chef workstation.

Chapter 5, *Installing and Configuring Docker*, covers overview of Docker container, understanding difference between virtual machines and containers, installation and configuration of Docker on CentOS, creating the first Docker container, and managing containers.

Chapter 6, *Cloud Provisioning and Configuration Management with Chef*, gives insight into Chef and cloud provisioning, installing knife plugins for Amazon Web Services and Microsoft Azure, and creating and configuring virtual machine in Amazon Web Services and Microsoft Azure.

Chapter 7, *Deploying Application in AWS, Azure, and Docker*, covers prerequisites—to deploy application on Remote Server, use tomcat manager app, deploying application in Tomcat Docker container, deploying application in AWS Elastic Beanstalk, and deploying application in Microsoft Azure web apps.

Chapter 8, *Monitoring Infrastructure and Applications*, provides overview of monitoring, Nagios monitoring tool and quick start on it, installation of Nagios, configuring monitoring of AWS EC2 instance, AWS Elastic Beanstalk monitoring, Microsoft Azure web app service monitoring, Microsoft Azure application insights, and monitoring web application and Tomcat server with New Relic.

Chapter 9, *Orchestrating Application Deployment*, describes in detail how to orchestrate different build jobs for continuous integration, configuration management, continuous delivery and so on. It will cover creating parameterized build jobs for end to end automation, configuring Build Pipeline for Orchestration of Build Job, executing Build Pipeline for Application Deployment Automation, Steps for Deployment in Amazon Elastic Beanstalk (Platform as a Service), Steps for Deployment in Microsoft Azure Web Apps (Platform as a Service), steps to implement end to end automation in Visual Studio Team Server and TFS online for Continuous Integration, Continuous Delivery and Continuous Deployment, and Steps for Deployment in Docker containers. It also gives a brief introduction on Hygieia—DevOps Dashboard and how to run it.

## What you need for this book

This book assumes that you are familiar with at least java programming language. Knowledge of core java and JEE is essential considering this book to gain better insight. Having a strong understanding of deployment of a web application in application server such as tomcat will help you to understand the flow quickly.

As application development lifecycle will cover lot of tools in general; it is essential to have some knowledge of repositories such as svn, git and so on. IDE tools such as Eclipse; build tools such as ant and maven. Knowledge of code analysis tools will make job easier in configuration and integration, however it is not extremely vital to perform exercises given in the book. Most of the configuration steps are mentioned clearly.

You will be walked through the steps required to install Jenkins 2, Chef Configuration Management tool. In order to be immediately successful, you will need administrative access to a host that runs a modern version of Linux; CentOS 6.x is what will be used for demonstration purposes. If you are a more experienced reader, then a recent release of almost any distribution will work just as well (but you may be required to do a little bit of extra work that is not outlined in the book). If you do not have access to a dedicated Linux host, a virtual host (or hosts) running inside of virtualization software such as VirtualBox or VMware workstation will work.

For AWS and Microsoft Azure, you can use the free trial and one-month free access respectively. Additionally, you will need access to the Internet to download plugins that you do not already have, as well as an installation of the Jenkins 2.

## Who this book is for

This book is especially aimed at technical readers. No prior experience with Continuous Integration, Cloud Computing, Configuration Management, Continuous Delivery, and Continuous Deployment is assumed. You may be novice or experienced with Continuous Integration tools such as Jenkins, Atlassian Bamboo, and so on. In any case, if you may want to bring the visualization of end to end automation to the reality and actually see:

- How to can you extend Continuous Integration to integrate with Configuration Management tools
- How to provision resources in AWS and Microsoft Azure Environment
- How to deploy Web Application in the different Cloud Environments

This book covers Continuous Integration, Cloud Computing, Configuration Management, Continuous Delivery, and Continuous Deployment for Sample Spring based application. The main objective is to see end to end automation and implement it one technology stack that can be extended further based on the understanding.

Additionally, different Cloud service models such as PaaS and IaaS of different Cloud Service Providers such as AWS and Microsoft Azure has been used. Docker containers are also used for application deployment. Infrastructure Monitoring with Nagios, Application Monitoring with New Relic, and native Monitoring features provided by AWS and Microsoft Azure are also covered.

## Conventions

In this book, you will find a number of text styles that distinguish between different kinds of information. Here are some examples of these styles and an explanation of their meaning.

Code words in text, database table names, folder names, filenames, file extensions, pathnames, dummy URLs, user input, and Twitter handles are shown as follows: "Now let's edit the `pom.xml` file."

A block of code is set as follows:

```
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"  
}
```

When we wish to draw your attention to a particular part of a code block, the relevant lines or items are set in bold:

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

New **terms** and **important words** are shown in bold. Words that you see on the screen, for example, in menus or dialog boxes, appear in the text like this: "Go to **Advanced Project Options**."

Warnings or important notes appear in a box like this.



Tips and tricks appear like this.



## Reader feedback

Feedback from our readers is always welcome. Let us know what you think about this book—what you liked or disliked. Reader feedback is important for us as it helps us develop titles that you will really get the most out of. To send us general feedback, simply e-mail [feedback@packtpub.com](mailto:feedback@packtpub.com), and mention the book's title in the subject of your message. If there is a topic that you have expertise in and you are interested in either writing or contributing to a book, see our author guide at [www.packtpub.com/authors](http://www.packtpub.com/authors).

## Customer support

Now that you are the proud owner of a Packt book, we have a number of things to help you to get the most from your purchase.

## Downloading the example code

You can download the example code files for this book from your account at <http://www.packtpub.com>. If you purchased this book elsewhere, you can visit <http://www.packtpub.com/support> and register to have the files e-mailed directly to you.

You can download the code files by following these steps:

1. Log in or register to our website using your e-mail address and password.
2. Hover the mouse pointer on the **SUPPORT** tab at the top.
3. Click on **Code Downloads & Errata**.
4. Enter the name of the book in the **Search** box.
5. Select the book for which you're looking to download the code files.
6. Choose from the drop-down menu where you purchased this book from.
7. Click on **Code Download**.

Once the file is downloaded, please make sure that you unzip or extract the folder using the latest version of:

- WinRAR / 7-Zip for Windows
- Zipeg / iZip / UnRarX for Mac
- 7-Zip / PeaZip for Linux

The code bundle for the book is also hosted on GitHub at <https://github.com/PacktPublishing/DevOps-for-Web-Development>. We also have other code bundles from our rich catalog of books and videos available at <https://github.com/PacktPublishing/>. Check them out!

## Downloading the color images of this book

We also provide you with a PDF file that has color images of the screenshots/diagrams used in this book. The color images will help you better understand the changes in the output. You can download this file from [https://www.packtpub.com/sites/default/files/downloads/DevOpsforWebDevelopment\\_ColorImages.pdf](https://www.packtpub.com/sites/default/files/downloads/DevOpsforWebDevelopment_ColorImages.pdf).

## Errata

Although we have taken every care to ensure the accuracy of our content, mistakes do happen. If you find a mistake in one of our books—maybe a mistake in the text or the code—we would be grateful if you could report this to us. By doing so, you can save other readers from frustration and help us improve subsequent versions of this book. If you find any errata, please report them by visiting <http://www.packtpub.com/submit-errata>, selecting your book, clicking on the **Errata Submission Form** link, and entering the details of your errata. Once your errata are verified, your submission will be accepted and the errata will be uploaded to our website or added to any list of existing errata under the Errata section of that title.

To view the previously submitted errata, go to <https://www.packtpub.com/books/content/support> and enter the name of the book in the search field. The required information will appear under the **Errata** section.

## Piracy

Piracy of copyrighted material on the Internet is an ongoing problem across all media. At Packt, we take the protection of our copyright and licenses very seriously. If you come across any illegal copies of our works in any form on the Internet, please provide us with the location address or website name immediately so that we can pursue a remedy.

Please contact us at [copyright@packtpub.com](mailto:copyright@packtpub.com) with a link to the suspected pirated material.

We appreciate your help in protecting our authors and our ability to bring you valuable content.

## Questions

If you have a problem with any aspect of this book, you can contact us at [questions@packtpub.com](mailto:questions@packtpub.com), and we will do our best to address the problem.

# 1

## Getting Started – DevOps Concepts, Tools, and Technologies

*“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 how DevOps solves different problems of the traditional application—delivery cycle. It also describes how it can be used to make development and operations teams efficient and effective in order to make time to market faster by improving culture. It also explains key concepts essential for evolving DevOps culture.

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

In this chapter, we will cover the following topics:

- Understanding the DevOps movement
- The DevOps lifecycle—it's all about “continuous”
- Continuous integration
- Configuration management

- Continuous delivery/continuous deployment
- Continuous monitoring
- Continuous feedback
- Tools and technologies
- Overview of a sample Java EE application

## **Understanding the DevOps movement**

Let's try to understand what DevOps is. Is it a real, technical word? No, because DevOps is not just about technical stuff. It is also neither simply 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 an application is deployed. The following sequence of events takes place when any new feature is implemented or bug fixed:

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

Let's list the possible issues in this approach:

- The transition of the current application build from the development environment to the production environment takes weeks or months.
- The priorities of the development team, QA team, and IT operations team are different in an organization and effective, and efficient co-ordination becomes a necessity for smooth operations.
- The development team is focused on the latest development release, while the operations team cares about the stability of the production environment.
- The development and operations teams are not aware of each other's work and work culture.

- Both teams work in different types of environments; there is a possibility that the development team has resource constraints and they therefore, use a different kind of configuration. It may work on the `localhost` or in the dev environment
- The operations team works on production resources and there will, therefore, be a huge gap in the configuration and deployment environments. It may not work where it needs to run – in the production environment.
- Assumptions are key in such a scenario, and it is improbable that both teams will work under the same set of assumptions.
- There is manual work involved in setting up the runtime environment and configuration and deployment activities. The biggest issue with the manual application-deployment process is its nonrepeatability and error-prone nature.
- The development team has the executable files, configuration files, database scripts, and deployment documentation. They provide it to the operations team. All these artifacts are verified on the development environment and not in production or staging.
- Each team may take a different approach for setting up the runtime environment and the configuration and deployment activities, considering resource constraints and resource availability.
- In addition, the 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 a situation where both use different automation techniques.
- Both teams are unaware of the challenges faced by each other and hence may not be able to visualize or understand an ideal scenario in which the application works.
- While the operations team is busy in deployment activities, the development team may get another request for a feature implementation or bug fix; in such a case, if the operations team faces any issues in deployment, they may try to consult the development team, who are already occupied with the new implementation request. This results in communication gaps, and the required collaboration may not happen.
- There is hardly any collaboration between the development team and the operations team. Poor collaboration causes many issues in the application's deployment to different environments, resulting in back-and-forth communication through e-mail, chat, calls, meetings, and so on, and it often ends in quick fixes.

- Challenges for the development team:
  - The competitive market creates on-time delivery pressure.
  - They have to take care of production-ready code management and new feature implementation.
  - The release cycle is often long and hence the development team has to make assumptions before the application deployment finally takes place. In such a scenario, it takes more time to fix the issues that occurred during deployment in the staging or production environment.
- Challenges for the operations team:
  - **Resource contention:** It's difficult to handle increasing resource demands
  - **Redesigning or tweaking:** This is needed to run the application in the production environment
  - **Diagnosing and rectifying:** They are supposed to diagnose and rectify issues after application deployment in isolation

## DevOps with the changing times

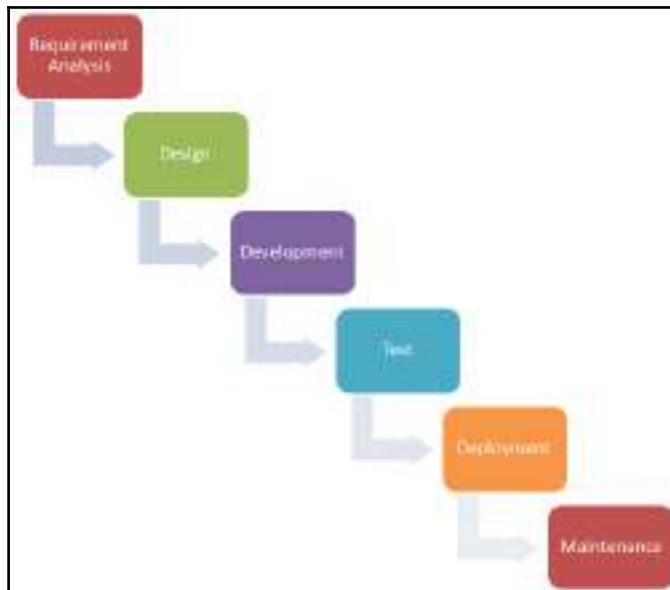
Time changes everything. In the modern era, customers expect and demand extremely quick response, and we need to deliver new features continuously to stay in business. Users and customers today have rapidly changing needs; they expect 24/7 connectivity and reliability and access services over smartphones,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



A change in the behavior of customers or market demand affects the development process.

## The waterfall model

The waterfall model follows sequential application design process for software development. It comes with good control but lacks revisions. It is a goal based development but without any scope of revision. The waterfall model has long been used for software development:



It has its advantages, as follows:

- Easy to understand
- Easy to manage—the input and output of each phase is defined
- Sequential process—order is maintained
- Better control

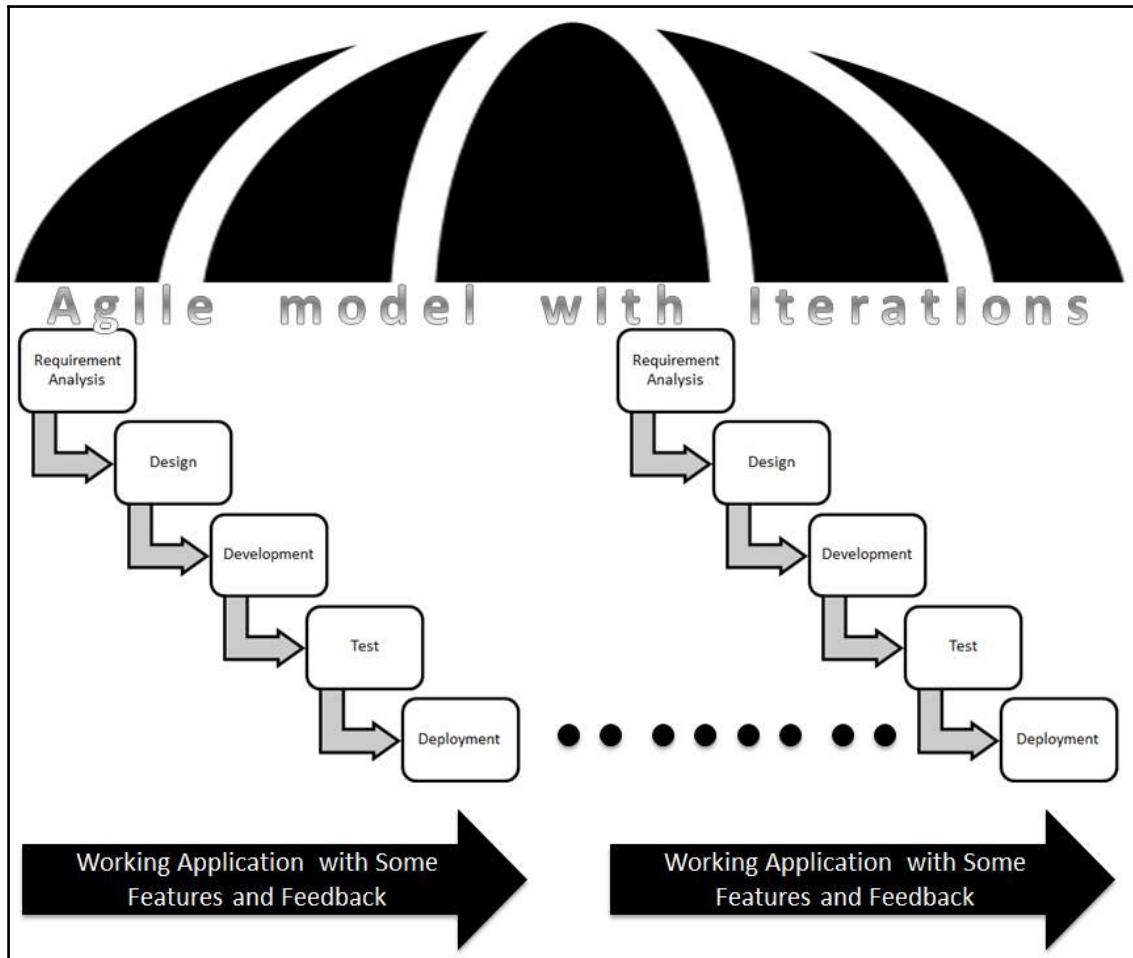
However, it is only useful in scenarios where requirements are predefined and fixed. As it is a rigid model with a sequential process, we can't go back to any phase and change things. It has its share of disadvantages, as follows:

- 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

## **The agile model**

Inefficient estimation, long time to market, and other issues led to a change in the waterfall model, resulting in the agile model. Agile development or the agile methodology is a method of building an application by empowering individuals and encouraging interactions, giving importance to working software, customer collaboration—using feedback for improvement in subsequent steps—and responding to change in an efficient manner. It emphasizes customer satisfaction through continuous delivery in small interactions for specific features in short timelines or **sprints**.

The following diagram illustrates the working mechanism of agile:



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 multiple deployments. Why? After each sprint, a version of the application with some features is ready for showcasing. It needs to be deployed in specific environments for demonstration, and thus, deployment is no longer a one-time activity.

It is very essential from an 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 the agile methodologies.

In such a case, traditional manual deployment processes work as speed barriers for incremental deployments. Hence, it is necessary to change other processes as well along with a change in the application development methodology. One key can't be used for all locks; similarly, the waterfall model is not suitable for all projects. We need to understand that agile is customer focused and feedback is vital. Changes happen based on customer feedback, and release cycles may increase. Just imagine a scenario where inputs are high but input processing is slow. Consider an example of a shoe company where one department prepares shoes and another department works on final touches and packaging. What would happen if the packaging process were slow and inefficient? Shoes would pile up in the packaging department. Now let's add a twist to this situation. What if the shoe-making department brings new machines and improves the process of making shoes? Let's say it makes the shoe-making process two to three times faster. Imagine the state of the packaging department. Similarly, cloud computing and DevOps have gained momentum, which increases the speed of delivery and improves the quality of the end product. Thus, the agile approach of application development, improvement in technology, and disruptive innovations and approaches have created a gap between development and operations teams.

## **Collaboration**

DevOps attempts to fill these gaps by developing a partnership between the development and operations teams. The DevOps movement emphasizes communication, collaboration, and integration between software developers and IT operations. DevOps promotes collaboration, and collaboration is facilitated by automation and orchestration in order 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 and processes leveraging the benefits of cloud solutions. Agile development and testing methodologies help us meet the goals of continuously integrating, developing, building, deploying, testing, and releasing applications. It provides a mechanism for constant feedback from different teams and stakeholders. It also provides transparency in the form of a platform for collaboration across teams, such as business analysts, developers, and testers. In short, agile and DevOps are compatible and increase each other's value.

One of the most popular sayings is that practice makes a man perfect. What if that saying were applied to a production-like environment? It is much easier to repeat the entire process as there are no last minute—surprises, and most of the issues in deployment have already been 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 lifecycle, 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. This is done by performing all activities multiple times and making them an 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 the deployment process has already been rehearsed on many different environments in progressively more production-like environments.

## **Cloud computing – the disruptive innovation**

A major challenge is managing the infrastructure for all environments. Virtualization and cloud environments can help you get started with this. The cloud helps us overcome this hurdle by providing flexible on-demand resources and environments. It provides distributed access across the globe and helps in the effective utilization of resources. The cloud provides a repository of software—tools that can be used on an on-demand basis. We can clone environments and reproduce required versions as and when required. The entire development, test, and production environments can be monitored and managed using the facilities provided by cloud providers. With the advent of cloud computing, it is easy to recreate every piece of infrastructure used by an application using automation. This means that operating systems, OS configuration, runtime environments and configuration, infrastructure configuration, and so forth can all be managed. In this way, it is easy to recreate the production environment exactly in an automated fashion. Thus, DevOps on cloud brings in the best-of-breed solution from both agile development and cloud solutions. It helps in providing a distributed agile environment in the cloud, leading to continuous accelerated delivery.

## **Why DevOps?**

DevOps is effective because of new methodologies, automation tools, agile resources of 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 a huge change in the way development and operations teams collaborate and communicate. Organizations need to have a change in culture and have long term business goals that include DevOps in their vision. It is important to establish the pain points and obstacles experienced by different teams or business units and use that knowledge for refining business strategy and fixing 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 the common issues faced by different sections of an organization and change the strategy to bring more value, then it makes sense. It can be a stepping stone in the direction of DevOps. With old values and objectives, it is difficult to adopt any new path. It is very important to align people with the new process first. For example, a team has to understand the value of the agile methodology; else, they will resist using it. They might resist it because they are comfortable with the old process. Hence, it is important to make them realize the benefit as well as empowering them to bring about the change.

*“Change is hard because people overestimate the value of what they have—and underestimate the value of what they may gain by giving that up.”*

*-James Belasco and Ralph Stayer*

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

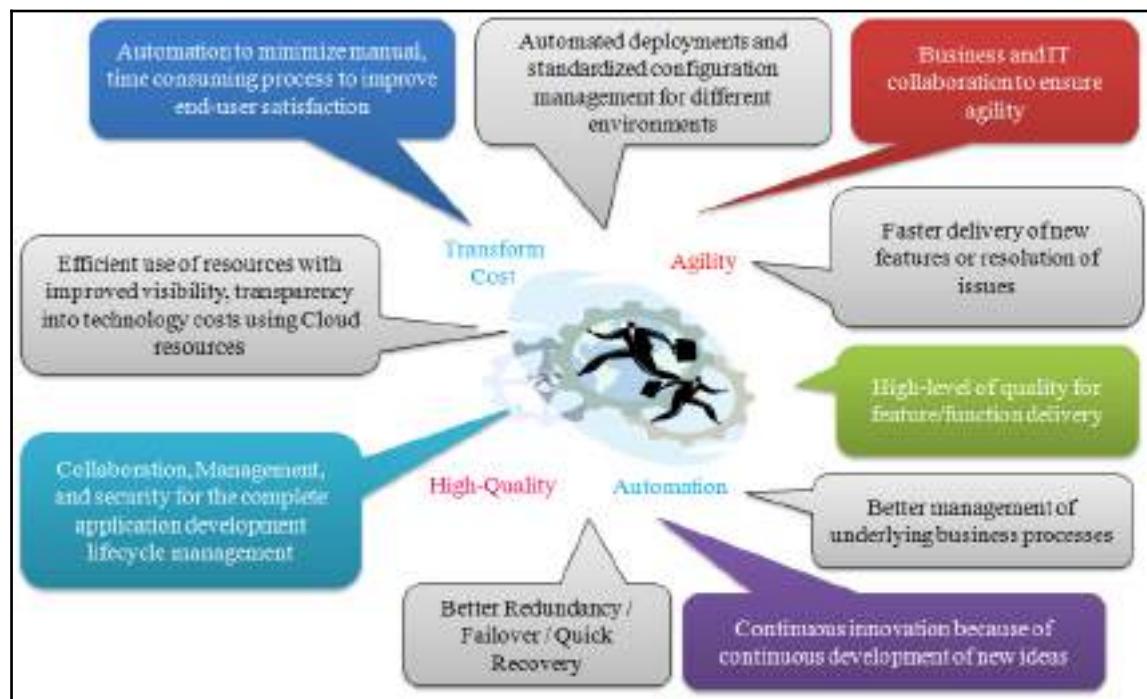
The end objective of the DevOps culture is continuous improvement. We learn from our mistakes, and it becomes experience. Experience helps us identify robust design patterns and minimize errors in processes. This leads to an enhancement of productivity, and hence, we achieve new heights with continuous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

## The benefits of DevOps

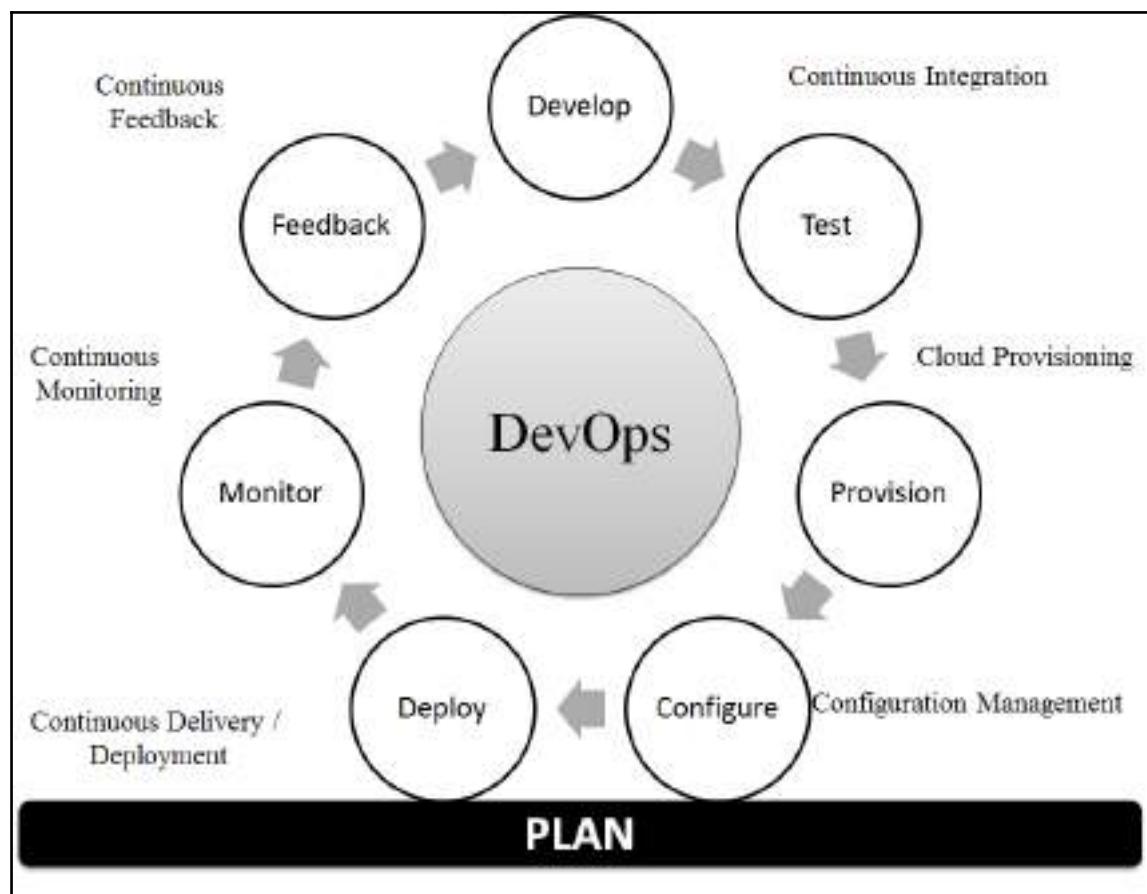
This diagram covers all the benefits of DevOps:



Collaboration among different stakeholders brings many business and technical benefits that help organizations achieve their business goals.

# The DevOps lifecycle – it's all about “continuous”

Continuous Integration (CI), Continuous Testing (CT), and Continuous Delivery (CD) are significant part of DevOps culture. CI includes automating builds, unit tests, and packaging processes while CD is concerned with the application delivery pipeline across different environments. CI and CD accelerate the application development process through automation across different phases, such as build, test, and code analysis, and enable users achieve end-to-end automation in the application delivery lifecycle:

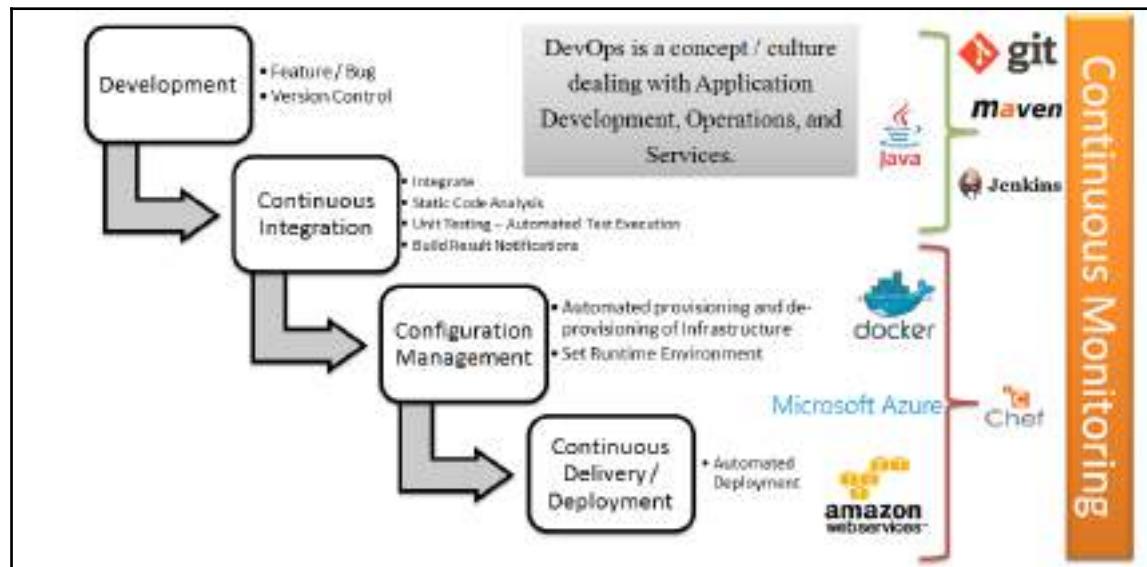


Continuous integration and continuous delivery or deployment are well supported by cloud provisioning and configuration management. Continuous monitoring helps identify issues or bottlenecks in the end-to-end pipeline and helps make the pipeline effective.

Continuous feedback is an integral part of this pipeline, which directs the stakeholders whether are close 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

The following diagram shows a mapping of different parts of an application delivery pipeline with the toolset for Java web applications:



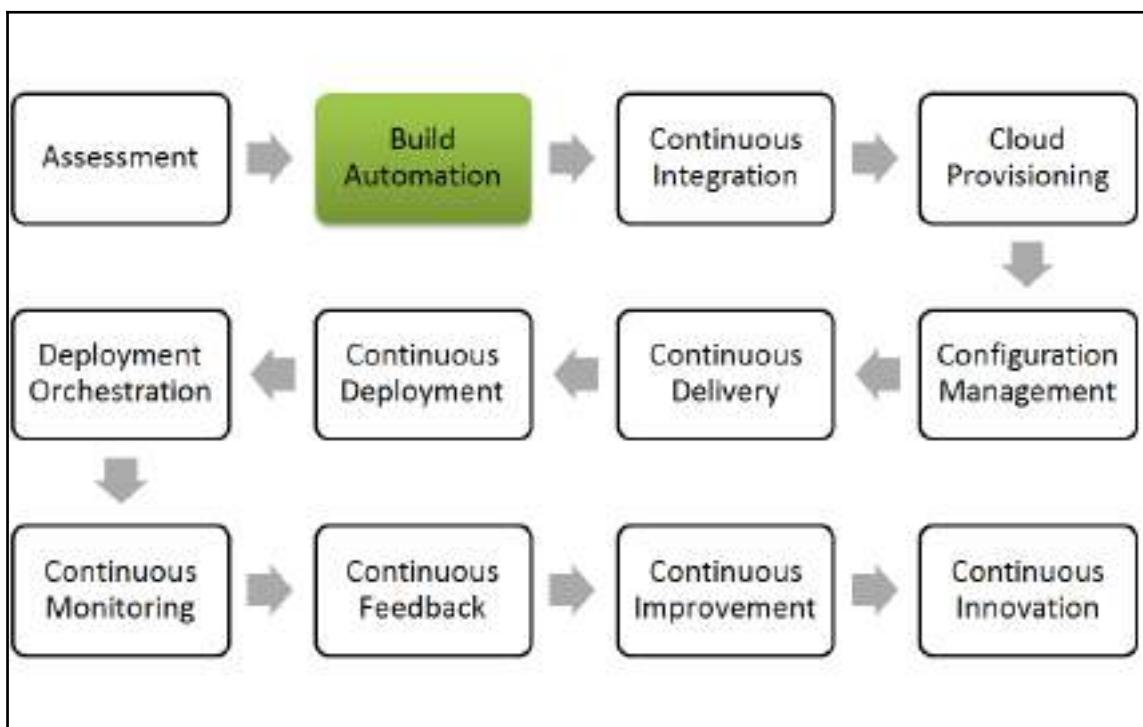
We will use a sample Spring application throughout this book for demonstration purposes, which is why the toolset is related to Java.

## Build automation

An automated build helps us create an application build using build automation tools such as Apache Ant and Apache Maven. An automated build process includes the following activities:

- Compiling source code into class files or binary files
- Providing references to third-party library files
- Providing the path of configuration files
- Packaging class files or binary files into WAR files in the case of Java
- Executing automated test cases
- Deploying WAR files on local or remote machines
- Reducing manual effort in creating the WAR file

Maven and Ant automate the build process and make it simple, repeatable, and less error prone as it is a create-once-run-multiple-times concept. Build automation is the base of any kind of automation in the application delivery pipeline:



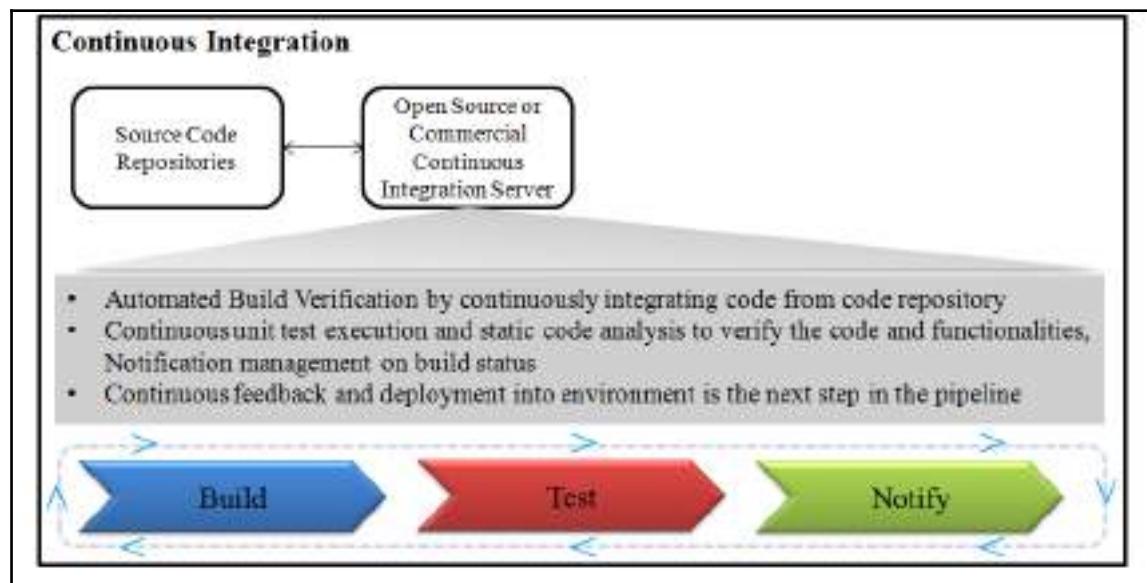
Build automation is essential for continuous integration and the rest of the automation is effective only if the build process is automated. All CI servers, such as Jenkins, Atlassian, and Bamboo use build files for continuous integration and creating their application-delivery pipeline.

## Continuous integration

What is continuous integration? In simple words, CI is a software engineering practice where each check-in made by a developer is verified by either of the following:

- **Pull mechanism:** Executing an automated build at a scheduled time
- **Push mechanism:** Executing an automated build when changes are saved in the repository

This step is followed by executing a unit test against the latest changes available in the source code repository:



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



Read more about CI at <http://martinfowler.com/articles/continuousIntegration.html>.

So why is CI needed? Because it makes things simple and helps us identify bugs or errors in the code at a very early stage of development, when it is relatively easy to fix them. Just imagine if the same scenario takes place after a long duration and there are too many dependencies and complexities we need to manage. In the early stages, it is far easier to cure and fix issues; consider health issues as an analogy, and things will be clearer in this context.

Continuous integration 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 for the release-management strategy of any organization that wants to develop a DevOps culture.

Following are immediate 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 requirement
- Quick feedback: build status notification to stakeholders via e-mail
- Teams focused on their work and not in the managing processes

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

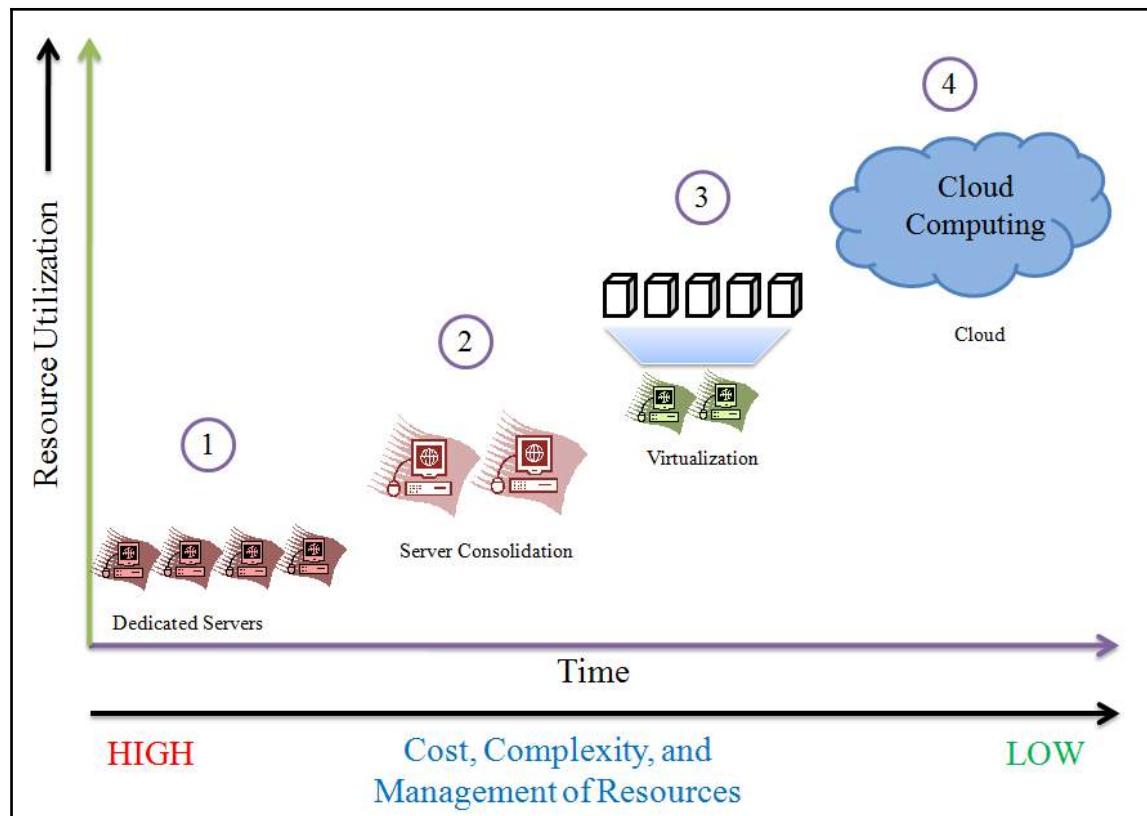
## Best practices

We will now be looking at best practices that can be useful when considering a continuous integration implementation:

- Maintain a code repository such as Git or SVN.
- Check-in third-party JAR files, build scripts, other artifacts, and so on into the code repository.
- Execute builds fully from the code repository: Use a 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 manage operations in a better way. A single job for all tasks is difficult when trying to troubleshoot. It also helps to assign build execution to slave instances, if that concept is supported by CI server.
- Backup the `home` directory of the CI server regularly as it contains archived builds and other artifacts too, which may be useful in troubleshooting.
- Make sure the CI server has enough free disk space available as it stores a lot of build-related details.
- Do not schedule multiple jobs to start at the same time, or use a master-slave concept, where specific jobs are assigned to slave instances so that multiple build jobs can be executed at the same time.
- Set up an e-mail, SMS, or Twitter notification to specific stakeholders of a project or an application. It is advisable to use customized e-mails for specific stakeholders.
- It is advisable to use community plugins.

## Cloud computing

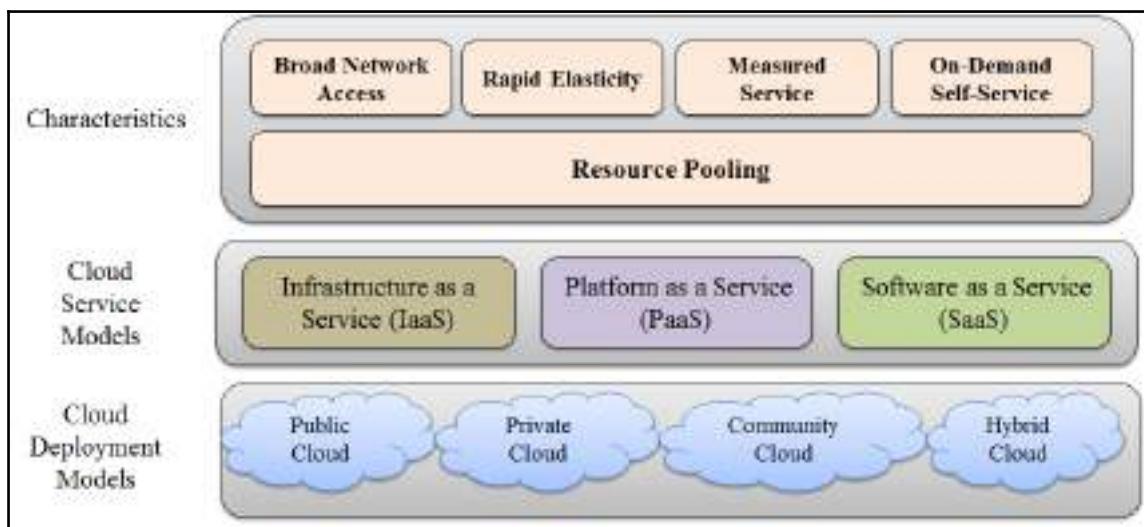
Cloud computing is regarded as a groundbreaking innovation of recent years. It is reshaping the technology landscape. With breakthroughs made in appropriate service and business models, cloud computing has expanded to its role as a backbone for IT services. Based on experience, organizations improved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





Cloud computing provides elastic and unlimited resources that can be efficiently utilized at the time of peak load and normal load with a pay-per-use pricing model. The pay-as-you-go feature is a boon for development teams that have faced resource scarcity for years. It is possible to automate resource provisioning and configuration based on your requirements, which has reduced a lot of manual effort. For more information, refer to *NIST SP 800-145, The NIST Definition of Cloud Computing* at <http://nvlpubs.nist.gov/nistpubs/Legacy/SP/nistspecialpublication8-145.pdf>.

It has opened various opportunities in terms of the availability of application—deployment environments, considering three service models and four deployment models as shown in the following diagram:



There are four cloud deployment models, each addressing specific requirements:

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

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

## Configuration management

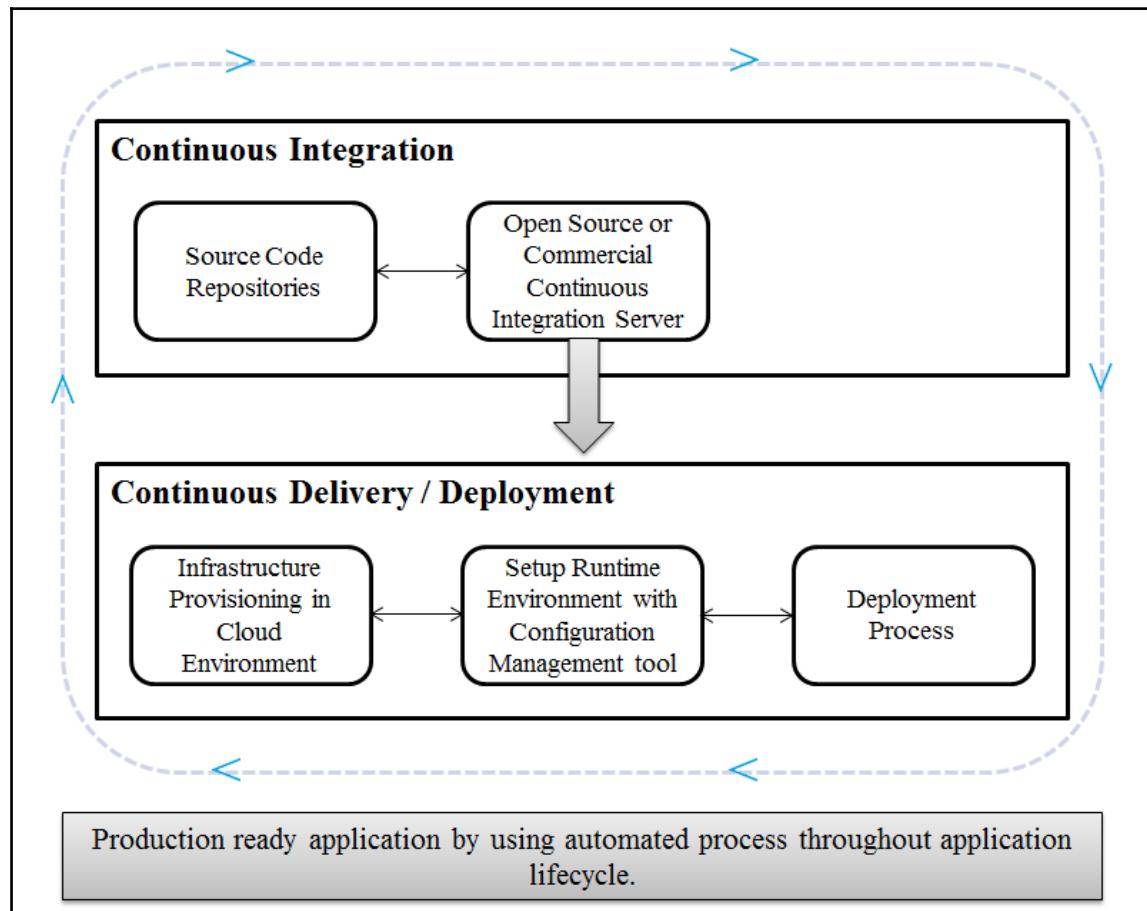
**Configuration management(CM)** manages changes in the system or, to be more specific, the server runtime environment. Let's consider an example where we need to manage multiple servers with same kind of configuration. For example, we need to install Tomcat on each server. What if we need to change the port on all servers or update some packages or provide rights to some users? Any kind of modification in this scenario is a manual and, if so, error-prone process. As the same configuration is being used for all the servers, automation can be useful here. Automating installation and modification in the server runtime environment or permissions brings servers up to spec effectively.

CM is also about keeping track or versions of details related to the state of specific nodes or servers. It is a far better situation when we need and update themselves. A centralized change can trigger this, or nodes can communicate with the CM server about whether they need to update themselves. CM tools make this process efficient when only changed behavior is updated, and the entire installation and modification isn't applied again to the server nodes.

There are many popular configuration management tools in the market, such as Chef, Puppet, Ansible, and Salt. Each tool is different in the way it works, but the characteristics and end goal are the same: to bring standardized behavior to the state changes of specific nodes without any errors.

## Continuous delivery/continuous deployment

Continuous delivery and continuous deployment are used interchangeably more often than not. However, there is a small difference between them. Continuous delivery is a process of deploying an application in any environment in an automated fashion and providing continuous feedback to improve its quality. Continuous deployment, on the other hand, is all about deploying an application with the latest changes to the production environment. In other words, we can say that continuous deployment implies continuous delivery, but the converse isn't true:



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

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

- The Azure web app configured with specific types of resources
- A storage account to store **BACPAC** files to create the database

Then, we need to follow these steps:

1. Create a SQL Server instance to host the database.
2. Import BACPAC files from the storage account to create a new database.
3. Deploy the web application to Microsoft Azure.

In this scenario, we may consider to use a 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 the configuration of resources changes according to the environment, but that can be managed in a 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, repetitive steps can be easily managed by an automated approach, and this is helpful both in continuous delivery and continuous deployment.

## Best practices for continuous delivery

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

- Plan to automate everything in an application delivery pipeline: Consider a situation where just 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, and deployment. You must remember to automate:
  - Repetitive tasks
  - Difficult tasks
  - Manual tasks
- Develop and test the newly implemented bug fixes in a production-like environment; it is possible now with pay-per-use resources provided by cloud computing.
- Deploy frequently in the development and test environments 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/2018/continuous-delivery-vs-continuous-deployment/>.

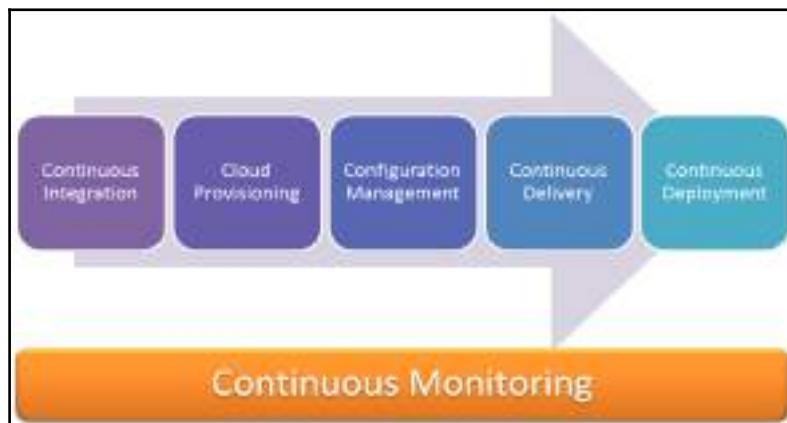
*Continuous Delivery versus Continuous Deploy:* <http://devops.com/2015/13/continuous-delivery-versus-continuous-deploy/>.



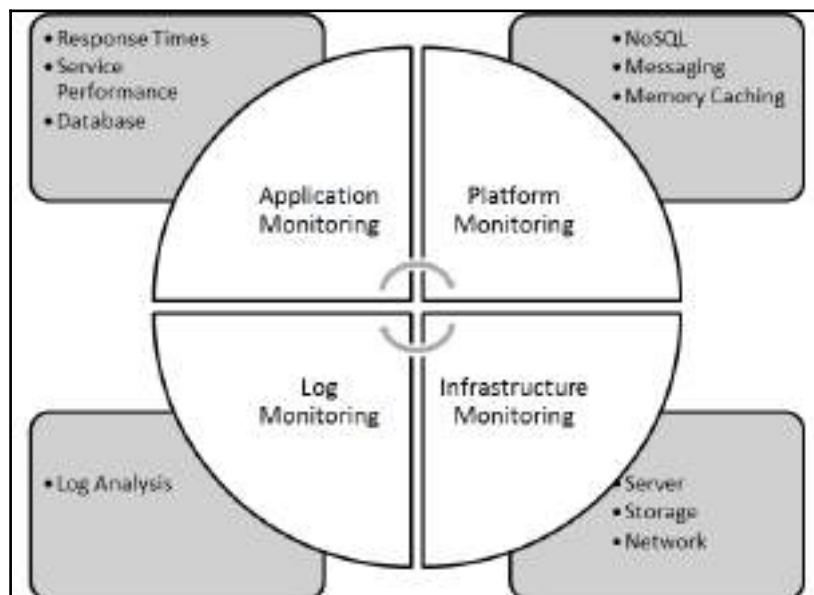
## Continuous monitoring

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 monitor at almost every stage in order to have transparency about all the processes, as shown in the following diagram. It also helps us troubleshoot quickly.

Monitoring should be a well thought-out implementation of a plan and it should be a part of each of the component mentioned in the following diagram. Consider monitoring practices for continuous integration to continuous delivery/deployment:



There is a likely scenario where end-to-end deployment is implemented in an 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 following diagram:



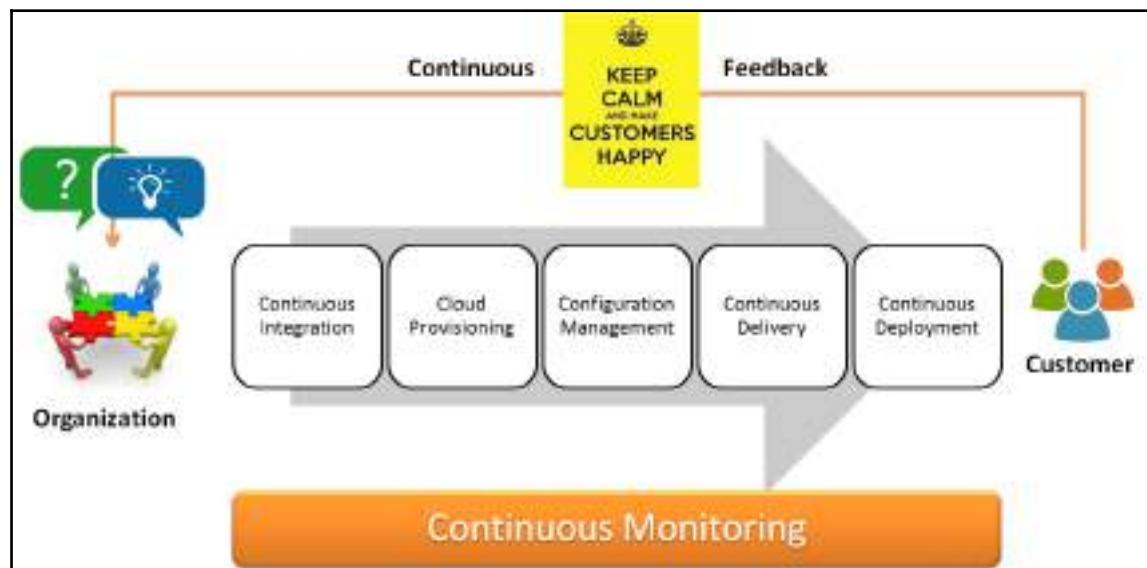
However, there is normally a tendency to monitor only infrastructure resources. The question one must ask is whether it is enough or whether we must focus on other types of monitoring as well. To answer this question, we must have a monitoring strategy in place in the planning stage itself. It is always better to identify stakeholders, monitoring aspects, and so on based on the 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 and provides a means of improvement and innovation. Feedback always provides improvement if it comes from stakeholders who know what they need and how the outcome should be. Feedback from the customer after deployment activities can serve as inputs to developers for improvement, as shown in the following diagram, and its correct integration will make the customer happy:



Here, we are considering a situation where a feature implementation is provided to the stakeholders and they provide their feedback. In the waterfall model, the feedback cycle is very long and hence developers may not be aware about whether the end product is what the customer asked for or whether the interpretation of what needs to be delivered was changed somewhere. In agile or DevOps culture, a shorter feedback cycle makes a major difference as stakeholders can actually see the result of a small implementation phase, and hence, the outcome is verified multiple times. If customers are not satisfied, then feedback is available at a stage where it is not very tedious to change things. In the waterfall model, this would've been a disaster as feedback used to be received very late. With time and dependencies, the complexity increases and changes in such situations takes a long time. In addition to this, no one remembers what they wrote 2 months back. Hence, a faster feedback cycle improves the overall process and connects endpoints as well as finding patterns in mistakes, learning lessons, and using improved patterns. However, continuous feedback not only improves the technical aspects of implementation but also provides a way to assess current features and whether they fit into the overall scenario or there is still room for improvement. It is important to realize that continuous feedback plays a significant role in making customers happy by providing an improved experience.

## **Tools and technologies**

Tools and technologies play an important role in the DevOps culture; however, it is not the only part that needs attention. For all parts of the 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. Using this, you can keep track of the applications being built. Features added months ago can also be tracked using the version code. It is all about tracking the code. Whenever any new features added or new code made, it is first tested and then committed by the developer. Then, 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 developers can easily understand changes that were made. Other developers only have to update their checkout to see the changes made.

## Advantages

The following are some advantages of using source code repositories:

- Many developers can work simultaneously on the same code
- If a computer crashes, the code can still be recovered as it had been committed in the server
- If a bug occurs, the new code can be easily reverted to the previous version

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

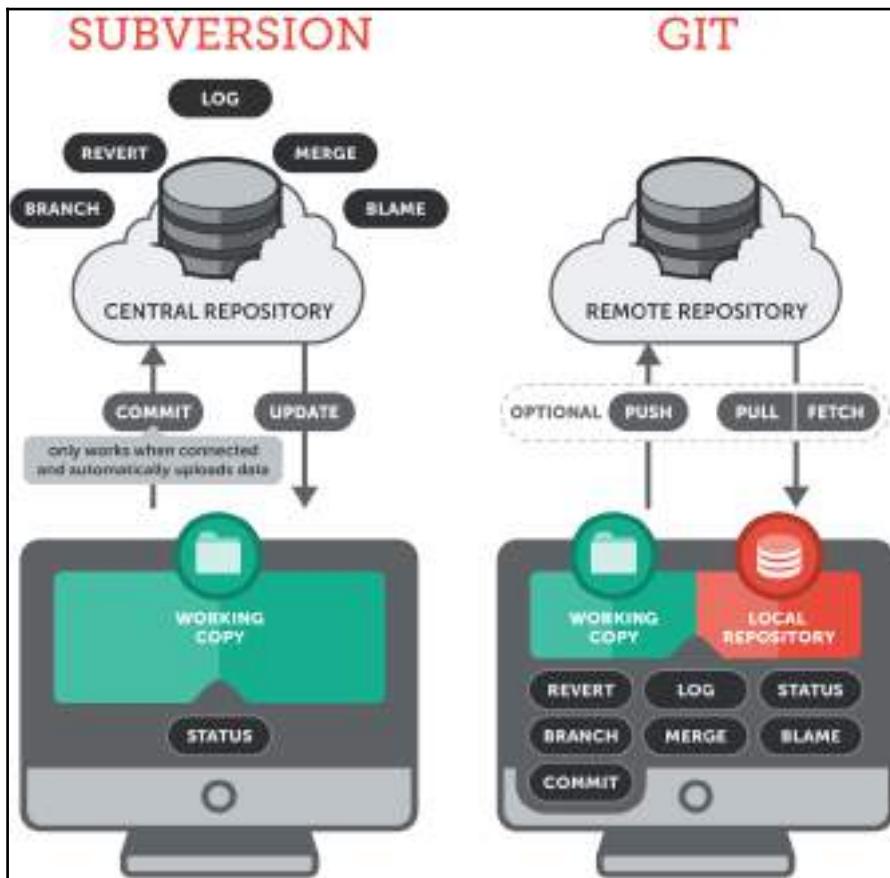
## Characteristics

The following are some significant characteristics of Git:

- It provides support for nonlinear development
- It is compatible with existing systems and protocols
- It ensures the cryptographic authentication of history
- It has well-designed pluggable merge strategies
- It consists of toolkit-based designs
- It supports various merging techniques, such as **resolve**, **octopus**, and **recursive**

## Differences between SVN and Git

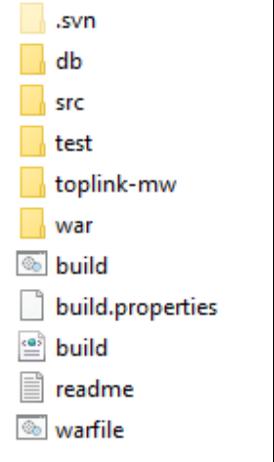
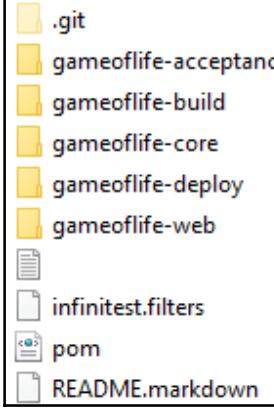
SVN and Git are both very popular source code repositories; however, Git is getting more popular in recent times. Let's look at the major differences between them:

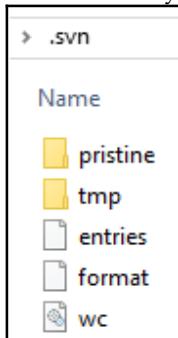
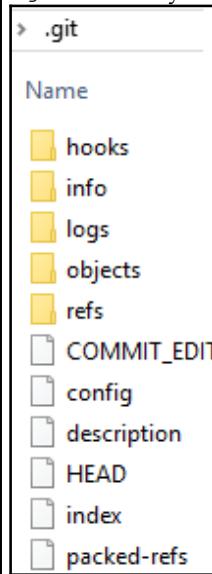


<https://www.git-tower.com/learn/git/ebook/en/mac/appendix/from-subversion-to-git>

Detailed description of Subversion and Git is illustrated in the following table:

Subversion	Git
Centralized version control system	Distributed version control system
Snapshot of a specific version of the project is available on the developer's machine	Complete clone of a full-fledged repository is available on the developer's machine
Perform operations such as commit, merge, blame, and revert and verifies branch and log from a central repository	Perform operations such as commit, merge, and blame and verifies branch and log from a local repository, along with pull and push operation to a remote repository if the developer needs to share work with others
URLs are used for trunks, branches, or tags: <code>https://&lt;URL/IP Address&gt;/svn/trunk/AntExample1/</code>	.git is the root of projects, and commands are used to address branches and not URLs: <code>git@github.com:mitesh51/game-of-life.git</code>
An SVN workflow:	A Git workflow:
<pre> graph TD     A[Active feature implementation is developed within branches subdirectories] --&gt; B[Feature implementation is finished]     B --&gt; C[Featured branch subdirectory is merged into trunk]     C --&gt; D[Featured Branch subdirectory is removed]     D --&gt; E[Trunk-latest stable release of a project]   </pre>	<pre> graph TD     A[History of all branches and tags within the .git directory] --&gt; B[The latest stable release is available within the master branch]     B --&gt; C[Active feature implementation is developed in separate branch]     C --&gt; D[Featured branch subdirectory is merged into trunk]     D --&gt; E[Featured Branch subdirectory is removed]   </pre>

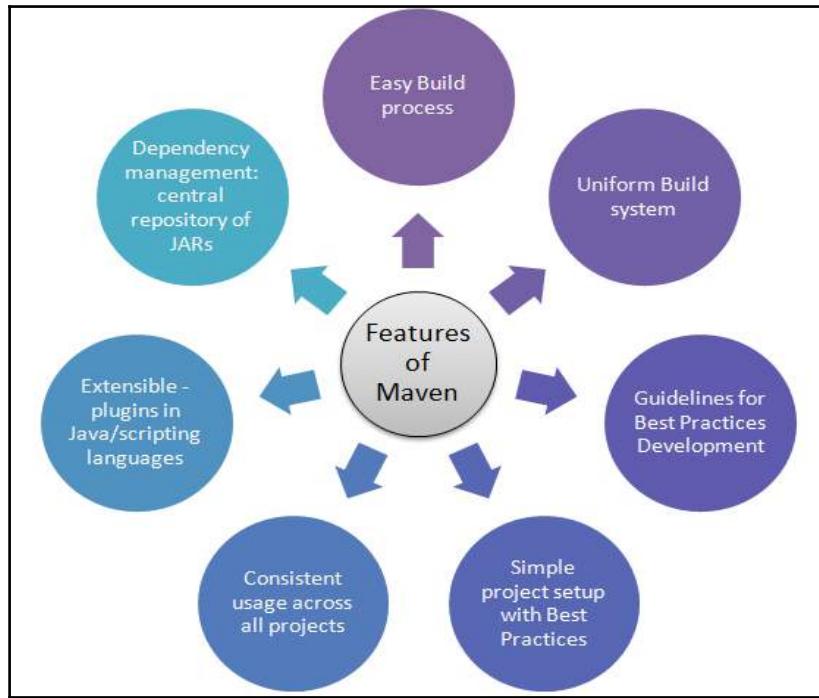
File changes are included in the next commit	File changes have to be marked explicitly and only then are they included in the next commit
Committed work is directly transferred to the central repository, and hence, direct connection to the repository must be available	Committed work is not directly transferred to the remote repository and committed to local repository, and to share it with other developers, we need to push it to the remote repository, in which case we need a connection to the remote repository
Each commit gets ascending revision numbers	Each commit gets commit hashes rather than ascending revision numbers
<b>Application directory:</b>  <ul style="list-style-type: none"> <li>.svn</li> <li>db</li> <li>src</li> <li>test</li> <li>toplink-mw</li> <li>war</li> <li>build</li> <li>build.properties</li> <li>build</li> <li>readme</li> <li>warfile</li> </ul>	<b>Application directory:</b>  <ul style="list-style-type: none"> <li>.git</li> <li>gameoflife-acceptance</li> <li>gameoflife-build</li> <li>gameoflife-core</li> <li>gameoflife-deploy</li> <li>gameoflife-web</li> <li>infinitest.filters</li> <li>pom</li> <li>README.markdown</li> </ul>

<p>.svn directory structure:</p>  <pre>› .svn   Name   ├── pristine   ├── tmp   ├── entries   ├── format   └── wc</pre>	<p>.git directory structure:</p>  <pre>› .git   Name   ├── hooks   ├── info   ├── logs   ├── objects   ├── refs   ├── COMMIT_EDIT   ├── config   ├── description   ├── HEAD   ├── index   └── packed-refs</pre>
Short learning curve	Long learning curve

## Build tools – Maven

Apache Maven is a build tool with the Apache 2.0 license. It is used for Java projects and can be used in a cross-platform environment. It can be also 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 about the name of the application, owner information, how the application distribution file can be created, and how dependencies can be managed.

## Example pom.xml file

The `pom.xml` file has predefined 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.

The following is an example of a sample pom.xml file used in Maven:

```
<!-- version 4.0.0-m19-SNAPSHOT -->
<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/xsd/maven-4.0.0.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>
    <dependencyManagementVersion>1.0.0-M19-SNAPSHOT</dependencyManagementVersion>
  </properties>
  <dependencies>
    <!-- Maven plugin versions are combined in order to guarantee the build reproducibility in the long term -->
    <build>
      <defaultGoal>install</defaultGoal>
      <testResources>
        <testResource>
          <!-- declared explicitly so Spring config files can be placed next to their corresponding class files since -->
          <directory>${project.basedir}/src/test/java/</directory>
        </testResource>
        <testResource>
          <directory>${project.basedir}/src/test/resources/</directory>
        </testResource>
      </testResources>
      <plugins>
        <plugin>
          <groupId>org.jacoco</groupId>
          <artifactId>jacoco-maven-plugin</artifactId>
          <version>0.8.0</version>
          <configuration>
            <includes>**/jacoco.xml</includes>
          </configuration>
          <executions>
            <execution>
              <phase>test</phase>
              <goals>
                <goal>prepare-agent</goal>
                <goal>report</goal>
              </goals>
            </execution>
          </executions>
        </plugin>
      </plugins>
    </build>
  </dependencies>
</project>
```

## Continuous integration tools – Jenkins

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

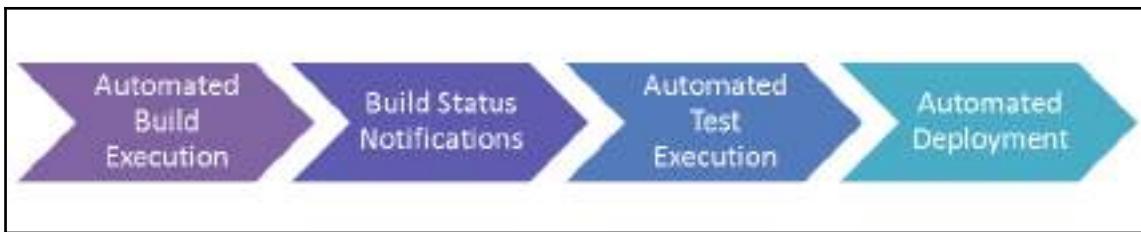
Jenkins can be used across different platforms, such as Windows, Ubuntu/Debian, Red Hat/Fedora, Mac OS X, openSUSE, and FreeBSD. Jenkins enables users to utilize continuous integration services for software development in an agile environment. It can be used to build freestyle software projects based on Apache Ant and Maven 2/Maven 3. It can also execute Windows batch commands and shell scripts.

It can be easily customized with the use of plugins. There are different kinds of plugins available for customizing Jenkins based on specific needs for setting up continuous integration. Categories of plugins include source code management (the Git, CVS, and Bazaar plugins), build triggers (the Accelerated Build Now and Build Flow plugins), build reports (the Code Scanner and Disk Usage plugins), authentication and user management (the Active Directory and GitHub OAuth plugins), and cluster management and distributed build (Amazon EC2 and Azure Slave plugins).



To know more about Jenkins please refer *Jenkins Essentials* <https://www.packtpub.com/application-development/jenkins-essentials>.

Jenkins accelerates the software development process through automation:



## Key features and benefits

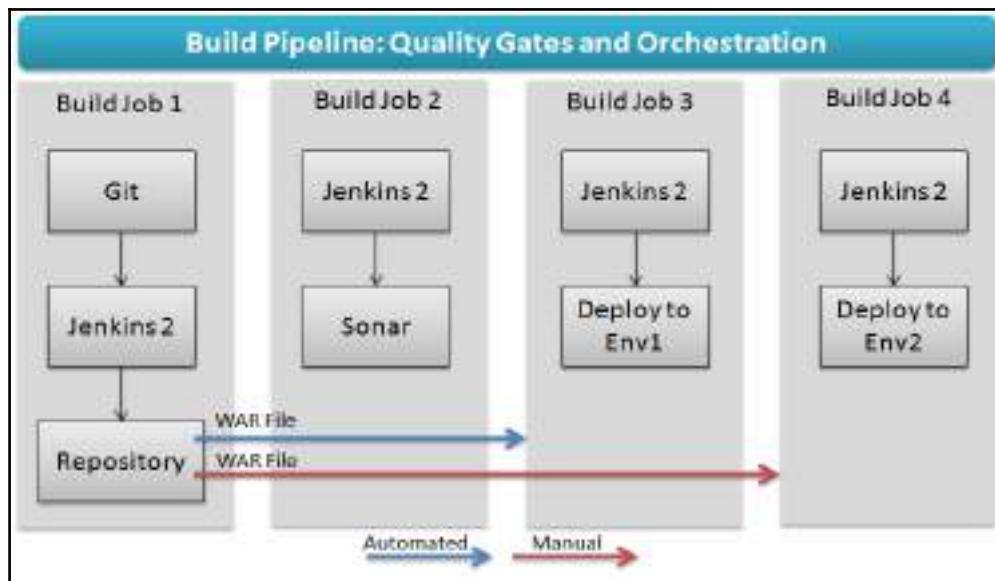
Here are some striking benefits of Jenkins:

- Easy install, upgrade, and configuration.
- **Supported platforms:** Windows, Ubuntu/Debian, Red Hat/Fedora/CentOS, Mac OS X, openSUSE, FreeBSD, OpenBSD, Solaris, and Gentoo.
- Manages and controls development lifecycle processes.
- **Non-Java projects supported by Jenkins:** Such as .NET, Ruby, PHP, Drupal, Perl, C++, Node.js, Python, Android, and 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 CI and CD.
- The use of Jenkins is not limited to CI and CD. It is possible to include a model and orchestrate the entire pipeline with the use of Jenkins as it supports shell and Windows batch command execution. Jenkins 2.0 supports a delivery pipeline that uses a **Domain-Specific Language (DSL)** for modeling entire deployments or delivery pipelines.
- The pipeline as code provides a common language-DSL-to help the development and operations teams to collaborate in an effective manner.
- Jenkins 2 brings a new GUI with stage view to observe the progress across the delivery pipeline.
- Jenkins 2.0 is fully backward compatible with the Jenkins 1.x series.
- Jenkins 2 now requires Servlet 3.1 to run.
- You can use embedded **Winstone-Jetty** or a container that supports Servlet 3.1 (such as Tomcat 8).
- GitHub, Collabnet, SVN, TFS code repositories, and so on are supported by Jenkins for collaborative development.
- **Continuous integration:** Automate build and the test automated testing (continuous testing), package, and static code analysis.
- Supports common test frameworks such as HP ALM Tools, JUnit, Selenium, and MSTest.
- For continuous testing, Jenkins has plugins for both; Jenkins slaves can execute test suites on different platforms.
- Jenkins supports static code analysis tools such as code verification by CheckStyle and FindBug. It also integrates with Sonar.
- **Continuous delivery and continuous deployment:** It automates the application deployment pipeline, integrates with popular configuration management tools, and automates environment provisioning.
- To achieve continuous delivery and deployment, Jenkins supports automatic deployment; it provides a plugin for direct integration with IBM uDeploy.
- **Highly configurable:** Plugins-based architecture that provides support to many technologies, repositories, build tools and test tools; it has an open source CI server and provides over 400 plugins to achieve extensibility.
- **Supports distributed builds:** Jenkins supports “master/slave” mode, where the workload of building projects is delegated to multiple slave nodes.

- It has a machine-consumable remote access API to retrieve information from Jenkins for programmatic consumption, to trigger a new build, and so on.
- It delivers a better application faster by automating the application development lifecycle, allowing faster delivery.

The Jenkins build pipeline (quality gate system) provides a build pipeline view of upstream and downstream connected jobs, as a chain of jobs, each one subjecting the 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. In the following diagram **Quality Gates and Orchestration of Build Pipeline** are illustrated:



Jenkins can be used with the following tools in different categories as shown here:

Language	Java	.Net
<b>Code repositories</b>	Subversion, Git, CVS, StarTeam	Subversion, Git, CVS, StarTeam
<b>Build tools</b>	Ant, Maven	NAnt, MS Build
<b>Code analysis tools</b>	Sonar, CheckStyle, FindBugs, NCover, Visual Studio Code Metrics, PowerTool	Sonar, CheckStyle, FindBugs, NCover, Visual Studio Code Metrics, PowerTool
<b>Continuous integration</b>	Jenkins	Jenkins

<b>Continuous testing</b>	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, and VsTest)	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, and VsTest)
<b>Infrastructure provisioning</b>	Configuration management tool-Chef	Configuration management tool-Chef
<b>Virtualization/cloud service provider</b>	VMware, AWS, Microsoft Azure (IaaS), traditional environment	VMware, AWS, Microsoft Azure (IaaS), traditional environment
<b>Continuous delivery/deployment</b>	Chef/deployment plugin/shell scripting/Powershell scripts/Windows batch commands	Chef/deployment plugin/shell scripting/Powershell scripts/Windows batch commands

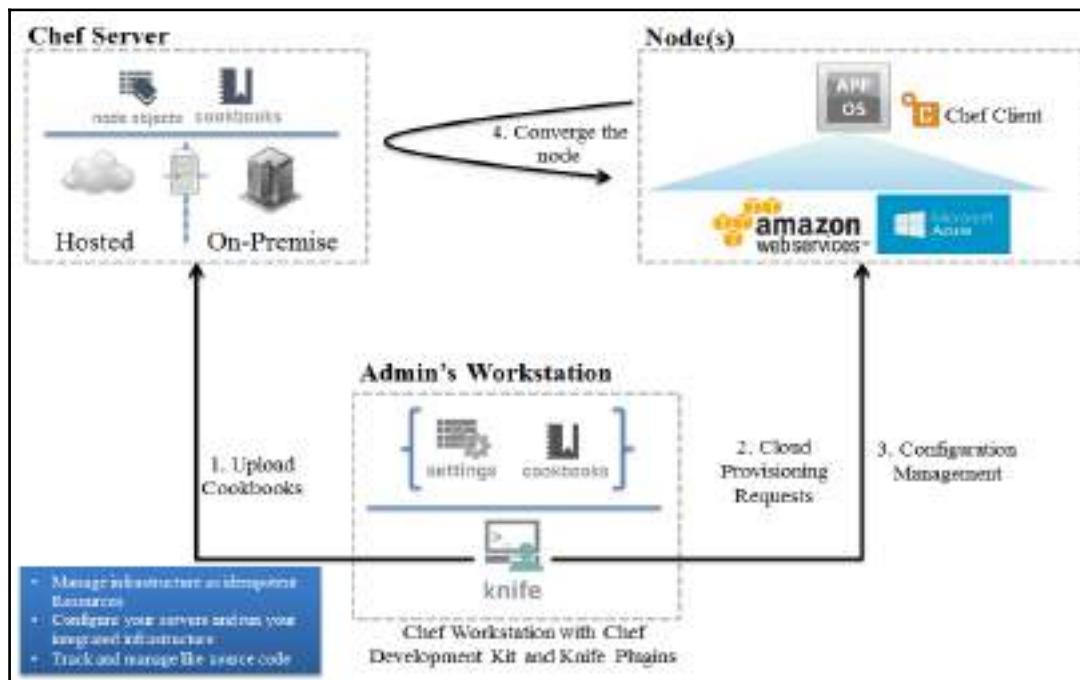
## Configuration management tools – Chef

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

Chef is a configuration management tool used to transform infrastructure into code. It automates the building, deploying, and managing of the infrastructure. By using Chef, infrastructure can be considered as code. The concept behind Chef is that 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 the infrastructure, such as the template, package, and files to be installed.

These recipes and configuration data are stored on Chef servers. The Chef client is installed on each node of the network. A node can be a physical or virtual server.

As shown in the following diagram, the Chef client periodically checks the Chef server for the latest recipes and to see whether 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

The following are some important features of the Chef configuration management tool:

- The Chef server:
  - It manages a huge number of nodes
  - It maintains a blueprint of the infrastructure
- The 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 versionable, testable, and repeatable way
  - Chef provides an automation platform to continuously define, build, and manage cloud infrastructure used for deployment
  - It enables resource provisioning and the configuration of resources programmatically, and it will help in the deployment pipeline in order to automate provisioning and configuration

The following three basic concepts of Chef will enable organizations to quickly manage any infrastructure:

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



To learn more about Chef refer *Learning Chef* <https://www.packtpub.com/networking-and-servers/learning-chef>.

## Cloud service providers

AWS and Microsoft Azure are popular public cloud providers right now. They provide cloud services in different areas, and both have their strong areas. Based on the organization's culture and past partnerships, either can be considered after a detailed assessment based on requirements.

The following is a side-by-side comparison:

	AWS	Microsoft Azure
<b>Virtual machines</b>	Amazon EC2	Virtual machine
<b>PaaS</b>	Elastic Beanstalk	Azure Web Apps
<b>Container services</b>	Amazon EC2 Container Services	Azure Container Services
<b>RDBMS</b>	Amazon RDS	Azure SQL Database
<b>NoSQL</b>	DynamoDB	DocumentDB
<b>BIG Data</b>	Amazon EMR	HD Insight
<b>Networking</b>	Amazon VPC	Virtual network
<b>Cache</b>	Amazon ElastiCache	Azure RedisCache
<b>Import/export</b>	Amazon import/export	Azure import/export
<b>Search</b>	Amazon CloudSearch	Azure Search
<b>CDN</b>	CloudFront	Azure CDN
<b>Identity and access management</b>	AWS IAM and Directory Services	Azure Active Directory
<b>Automation</b>	AWS OpsWorks	Azure Automation



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

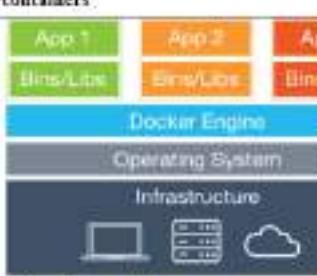
Microsoft Azure: <https://azure.microsoft.com>

## Container technology

Containers use OS-level virtualization, where the kernel is shared between isolated user-spaces. Docker and OpenVZ are popular open source examples of OS—level virtualization technologies.

## Docker

Docker is an open source initiative to wrap code, the runtime environment, system tools, and libraries. Docker containers share the kernel they are running on and hence start instantly and in a lightweight manner. Docker containers run on Windows as well as Linux distributions. It is important to understand how containers and virtual machines are different. Here is a comparison table of virtual machines and containers:

Virtual-machines	Docker-containers
 <a href="http://www.docker.com/">http://www.docker.com/</a>	 <a href="http://www.docker.com/">http://www.docker.com/</a>
Virtual machines depend on traditional virtualization. They can be considered hardware-level virtualization.	Containers depends on the containerization technique at the kernel level. They can be considered OS-level virtualization.
Each virtual machine contains a guest OS, binaries, library files, and the application itself.	Each container include application, binaries, and library files, but the major difference compared to virtual-machine is the shared kernel: each container runs as an isolated process in the user-space on the host OS.
The size of a virtual-machine is in the order of GBs, as each one runs on its own.	As each container runs as an isolated process in the user-space on the host OS and a separate OS is not required for each container, the size of each container is much smaller.
Each virtual-machine has its own set of resources, which results in better isolation and less sharing of resources.	Each container shares the kernel, and hence, there's more scope of sharing resources.
A virtual machine cannot run on a container	A container can run on a virtual-machine.



You can download Docker by visiting <https://github.com/docker/docker>.

## Monitoring tools

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

### Zenoss

**Zenoss** is an agentless and open source management platform for applications, servers, and networks released under the GNU **General Public License (GPL)** version 2 and is based on the Zope application server. Zenoss Core consists of the extensible programming language Python, object-oriented web server Zope, monitoring protocol network, graph and log time series data by RRD tool, MySQL, and event-driven networking engine Twisted. It provides an easy-to-use web portal to monitor alerts, performance, configuration, and inventory. In the following diagram, Zenoss features are illustrated:



You can visit Zenoss Core 5 website at <http://www.zenoss.org/>.



## Nagios

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



You can get more information by navigating to Nagios official website at <https://www.nagios.org/>.

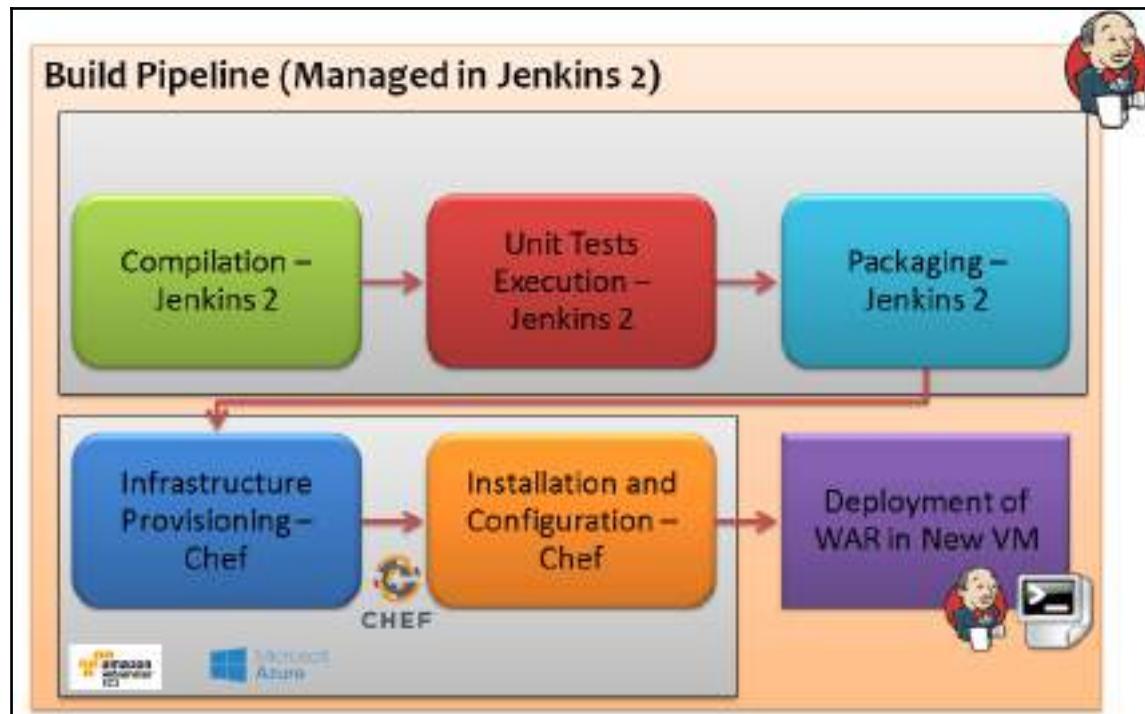
## Deployment orchestration/continuous delivery – Jenkins

The build pipeline, also called the **deployment** or **application delivery** pipeline, can be used to achieve end-to-end automation for all operations, including continuous integration, cloud provisioning, configuration management, continuous delivery, continuous deployment, and notifications. The following Jenkins plugins can be used for overall orchestration of all the 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

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



Jenkins can be used to manage unit testing and code verification; Chef can be used for setting up a runtime environment; Knife plugins can be used for creating a virtual machine in AWS or Microsoft Azure; the build pipeline or deployment pipeline plugins in Jenkins can be used for managing deployment orchestration.

From a single pipeline dashboard, we can view the status of all the builds that are configured in the pipeline. Each build in the pipeline is a kind of quality gate. If one build fails, then the execution won't go further. Additional dimensions can be added, such as notification based on compilation failures, unit test failures, or for unsuccessful deployment. The final deployment can be based on some sort of permission from a specific stakeholder. Consider a scenario for a parameterized build or promoted build concept-what should we do? All will be revealed in the chapters to follow!

## The DevOps dashboard

One of the most liked components of DevOps culture is the dashboard or GUI that provides a combined status of all end-to-end activities. For automation tools, an easy-to-use web GUI is handy for managing resources. For end-to-end automation in application deployment activities, multiple open source or commercial tools are used. There is a high possibility that a single product may not be used for all activities, for example, Git or SVN as the repository, Jenkins as the CI server, and IBM UrbanCode Deploy as the deployment orchestration tool. In such a scenario, it is easier if there is a single-pane-of-glass view where we can track multiple tools for a specific application.

**Hygieia** is an open source DevOps dashboard that provides a way to track the status of a deployment pipeline. It basically tracks six different areas as of now, including features (Jira, VersionOne), code repository (GitHub, Subversion), builds (Jenkins, Hudson), quality (Sonar, Cucumber/Selenium), monitoring, and deployment (IBM UrbanCode Deploy). Following is a sample image of configured DevOps dashboard:





Download Hygieia from here <https://github.com/capitalone/Hygieia>.

## An overview of a sample Java EE application

We are going to use PetClinic, available on GitHub. It is a sample spring application with JUnit test cases already written for it.



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:

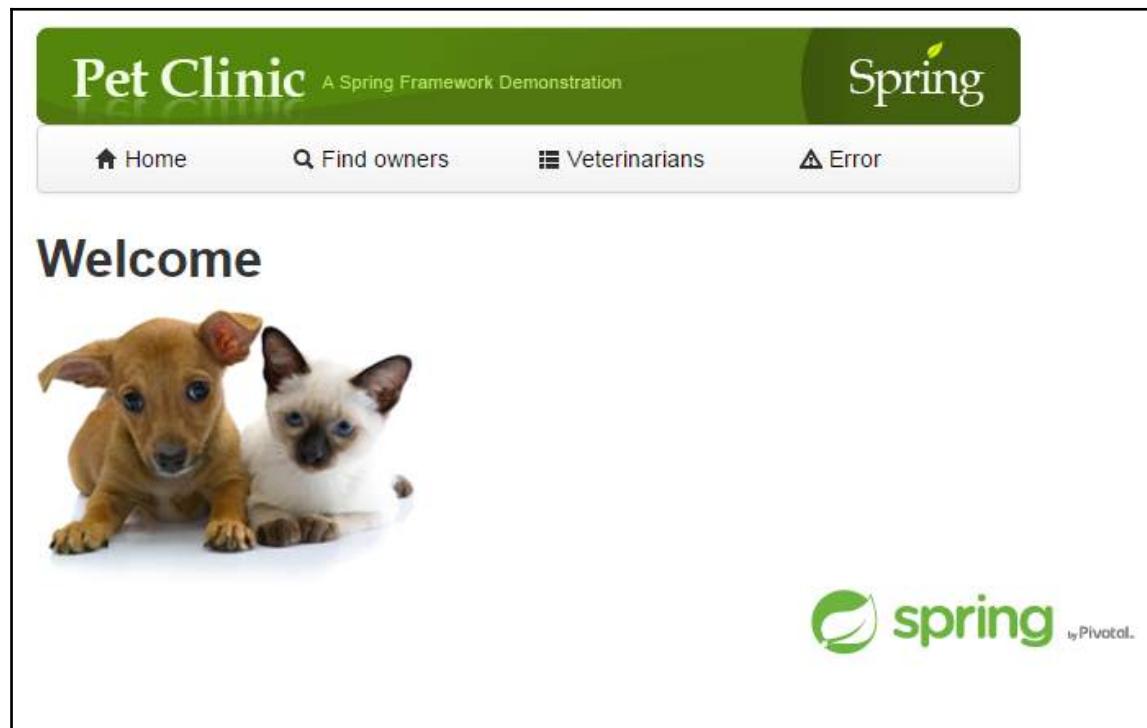
The screenshot shows the GitHub repository page for 'spring-projects/spring-petclinic'. The repository has 1,293 stars and 23 contributors. It contains 5 branches, 0 releases, and 437 commits. The master branch is selected. A green button labeled 'New pull request' is visible. Below the repository summary, there is a list of recent commits:

Author	Commit Message	Date
anyusing	anyusing Spring Boot Database in gradle work	2 months ago
anyusing	added missing .gradle wrapper for maven wrapper	2 months ago
MC	Using Spring Boot colormap UI gradle theme	12 days ago
anyusing	fixing tool imports to Dockerfile, renaming Dockerfile to vendor...	2 months ago
anyusing	fix change Embedding to embed to import other files than .java and...	2 months ago
anyusing	adding missing .gradle wrapper for maven wrapper	2 months ago
springteam	using latest version of liberman, spring-data-jpa...	2 years ago
travisgrf	Fix #29 migrate to Dockerfile:3.0	11 days ago
anyusing	Fix #29 migrate to Dockerfile:3.0	11 days ago

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 pet owner
- View a list of veterinarians and their specialties, a pet owner, a pet, and pet's visitation history

Once a WAR file is created, we can deploy it in Tomcat or another web server, and to verify it on the localhost, visit <http://localhost:8080/petclinic>. You will see something like this:



## The list of tasks

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

- Jenkins installation, configuration, UI personalization
- Java configuration (`JAVA_HOME`) in Jenkins
- Maven or Ant configuration in Jenkins
- Plugin installation and configuration in Jenkins
- Security (access control, authorization, and project-based security) in Jenkins
- Jenkins build configuration and execution
- Email notification configuration
- Deploying a WAR file to a web application server
- Creating and configuring a build/deployment pipeline
- Installing and configuring Chef
- Installing and configuring Docker
- Creating and configuring a virtual machine in AWS, Microsoft Azure, and containers
- Deploy a WAR file into a virtual machine and a container
- Configuring infrastructure monitoring
- Orchestrating the application delivery pipeline using Jenkins plugins

## Self-test questions

1. Which of the following statements is not related to the development team in a traditional environment?
  - A competitive market creates pressure of on-time delivery of feature or bug fixing
  - Production-ready code management and new feature implementation
  - The release cycle is often long and hence the development team has to make assumptions before the application deployment finally takes place
  - Redesigning or tweaking is needed to run the application in a production environment

2. Which of the following are benefits of DevOps?
  - Collaboration, management, and security for the complete application development lifecycle management
  - Continuous innovation because of continuous development of new ideas
  - Faster delivery of new features or resolution of issues
  - Automated deployments and standardized configuration management for different environments
  - All of these
  
3. Which of the following are parts of the DevOps culture or application delivery pipeline?
  - Continuous integration
  - Cloud provisioning
  - Configuration management
  - Continuous delivery/deployment
  - Continuous monitoring
  - Continuous feedback
  
4. Which of the following are by-products of the DevOps culture or application delivery pipeline?
  - Continuous integration
  - Continuous delivery/deployment
  - Continuous monitoring
  - Continuous feedback
  - Continuous improvement
  - Continuous innovation

5. State whether the following statements are true or false:
  - Jenkins and Atlassian Bamboo are build automation tools
  - Apache Ant and Apache Maven are continuous integration tools
  - Chef is a configuration management tool
  - Build automation is essential for continuous integration and the rest of the automation is effective only if the build process is automated
  - Subversion is a distributed version control system
  - Git is a centralized version control system
  - AWS and Microsoft Azure are public cloud service providers
6. Which of the following are cloud deployment models according to NIST's definition of cloud computing?
  - Public cloud
  - Private cloud
  - Community cloud
  - Hybrid cloud
  - All of these
7. Which of the followings are cloud service models according to NIST's definition of cloud computing?
  - Software as a Service
  - Platform as a Service
  - Infrastructure as a Service
  - All of these
8. Which of the following are major components of a Chef installation?
  - Chef server/hosted chef
  - Chef workstation
  - Nodes
  - All of these

## Summary

In this chapter, we learned about the difficulties faced by development and operations teams in a traditional environment and how agile development helps in such a scenario. What has changed after the arrival of agile development and what challenges has it brought with its arrival? We have covered the important aspects of the DevOps culture, including continuous integration and continuous delivery. We also covered details regarding cloud computing and configuration management that enhance the processes and help to adopt DevOps culture.

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

In the next chapter, we will see how to install and configure Jenkins 2.0 and implement continuous integration using a sample Spring application available on GitHub.

It is the 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 arrived. It comes with built-in support for delivery pipelines, improved usability, a new setup experience, and complete backward compatibility with existing Jenkins installations. We will be 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 a runtime environment for application lifecycle management and configure it with Jenkins. It manages all the aspects of running a build to create a distribution file or **Web application ARchive (WAR)** file for deployment by integrating with a source code repository such as SVN or Git for a sample Java EE application.

You will learn how to install and configure Jenkins, and you'll be able to get end-to-end experience in build job creation and configuration, static code analysis, notifications, Jenkins plugins, and so on as well as details on what exactly the sample application is all about.

In this chapter, we will cover the following topics:

- An introduction to Jenkins
- Installing Jenkins with plugins
- Configuring Java and Maven or Ant in Jenkins
- Creating and configuring a build job for a Java application with Maven
- The dashboard View plugin – overview and usage
- Sending e-mail notifications based on build status
- Integrating Jenkins and Sonar

## 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, the father of Taoism*

In simple words, CI is a software engineering practice where each check-in made by a developer is verified by either of the following:

- **Pull mechanism:** Executing an automated build at a scheduled time
- **Push mechanism:** Executing an automated build when changes are saved in the repository

This step is followed by executing a unit test against the latest changes available in the source code repository.

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

*“We are what we repeatedly do. Excellence, then, is not an act, but a habit.”*

*-Aristotle*

The next question you may ask is what makes Jenkins so popular. I already gave you one reason – can you recollect?

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

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

- It is written in Java
- It provides extensibility with over 400 plugins for different integrations, such as the following:
  - Source code management
  - Build triggers
  - Build reports
  - Artifact uploaders
  - External site/tool integrations
  - UI plugins
  - Authentication and user management
  - Cluster management and distributed build
- It supports Java, .NET, Ruby, Groovy, Grails, PHP, Android, and iOS applications
- It is easy to use:
  - It has a simple learning curve
  - The user interface was already simple, and it has now improved after Jenkins 2 has been made available to the general public
- Easy installation
- Easy configuration

## Installing Jenkins

Jenkins provides us with multiple ways to install it for all types of users. We can install it on at least the following operating systems:

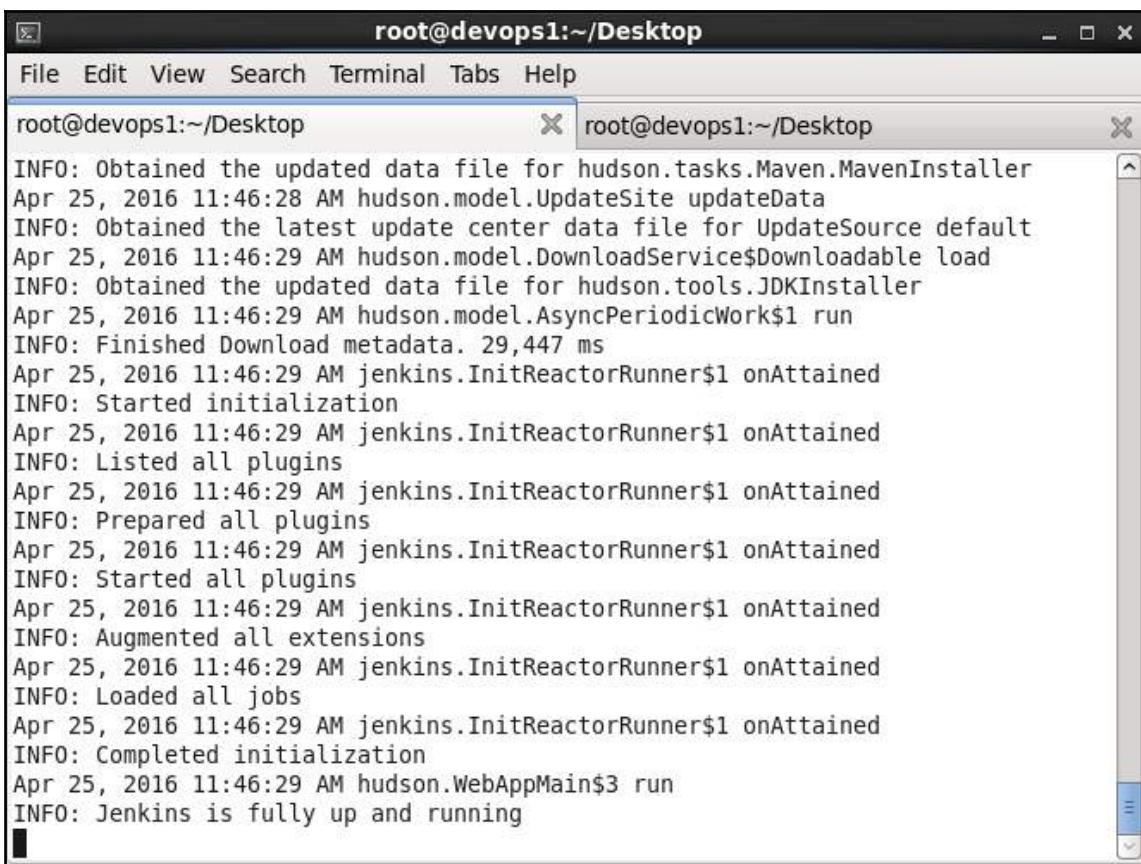
- Ubuntu/Debian
- Windows
- Mac OS X
- OpenBSD
- FreeBSD
- openSUSE
- Gentoo
- CentOS/Fedora/Red Hat

One of the easiest options I recommend is to use a WAR file. A WAR file can be used with or without a container or web application server. Having Java is a must before we try to use a WAR file for Jenkins, which can be done as follows:

1. Download the jenkins.war file from <https://jenkins.io/>.
2. Open command prompt in Windows or a terminal in Linux, go to the directory where the jenkins.war file is stored, and execute the following command:

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

3. Once Jenkins is fully up and running, as shown in the following screenshot, explore it in the web browser by visiting <http://localhost:8080>:



The screenshot shows a terminal window titled "root@devops1:~/Desktop". It has two tabs open, both labeled "root@devops1:~/Desktop". The left tab shows the Jenkins startup logs, which detail the process of obtaining update site data, downloading plugins, preparing extensions, loading jobs, and finally stating that Jenkins is fully up and running. The right tab is currently empty.

```
root@devops1:~/Desktop
File Edit View Search Terminal Tabs Help
root@devops1:~/Desktop          X root@devops1:~/Desktop X
INFO: Obtained the updated data file for hudson.tasks.Maven.MavenInstaller
Apr 25, 2016 11:46:28 AM hudson.model.UpdateSite updateData
INFO: Obtained the latest update center data file for UpdateSource default
Apr 25, 2016 11:46:29 AM hudson.model.DownloadService$Downloadable load
INFO: Obtained the updated data file for hudson.tools.JDKInstaller
Apr 25, 2016 11:46:29 AM hudson.model.AsyncPeriodicWork$1 run
INFO: Finished Download metadata. 29,447 ms
Apr 25, 2016 11:46:29 AM jenkins.InitReactorRunner$1 onAttained
INFO: Started initialization
Apr 25, 2016 11:46:29 AM jenkins.InitReactorRunner$1 onAttained
INFO: Listed all plugins
Apr 25, 2016 11:46:29 AM jenkins.InitReactorRunner$1 onAttained
INFO: Prepared all plugins
Apr 25, 2016 11:46:29 AM jenkins.InitReactorRunner$1 onAttained
INFO: Started all plugins
Apr 25, 2016 11:46:29 AM jenkins.InitReactorRunner$1 onAttained
INFO: Augmented all extensions
Apr 25, 2016 11:46:29 AM jenkins.InitReactorRunner$1 onAttained
INFO: Loaded all jobs
Apr 25, 2016 11:46:29 AM jenkins.InitReactorRunner$1 onAttained
INFO: Completed initialization
Apr 25, 2016 11:46:29 AM hudson.WebAppMain$3 run
INFO: Jenkins is fully up and running
```

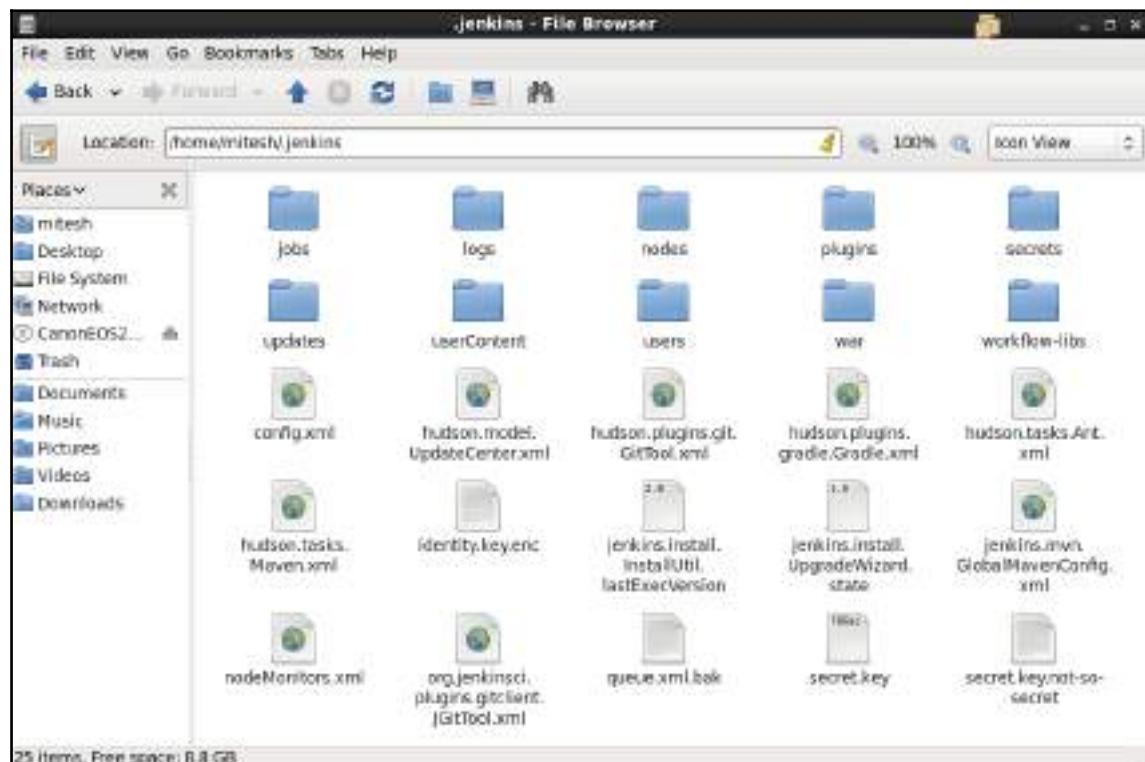
4. By default, Jenkins works on port 8080. Execute the following command from the command-line:

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

5. For HTTPS, use the following command:

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

6. Once Jenkins is running, visit the Jenkins home directory. In our case, we have installed Jenkins 2 on a CentOS 6.7 virtual machine.
7. Go to /home/<username>/ .jenkins, as shown in the following screenshot. If you can't see the .jenkins directory, make sure hidden files are visible. In CentOS, press *Ctrl + H* to make hidden files visible:

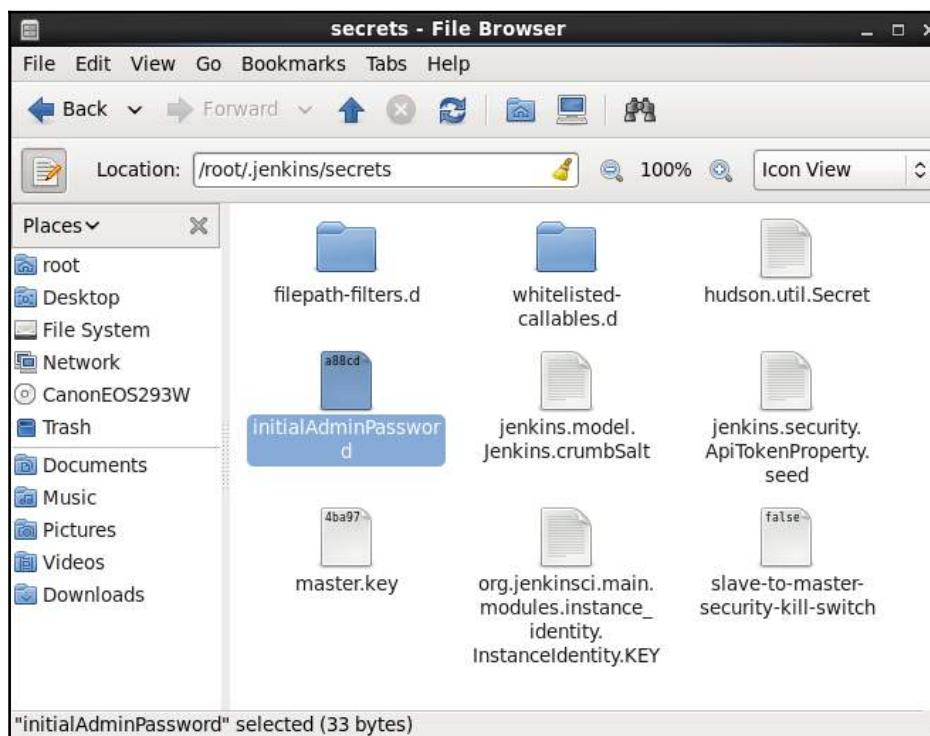


## Setting up Jenkins

Now that we have installed Jenkins, let's verify whether Jenkins is running. Open a browser and navigate to `http://localhost:8080` or `http://<IP_ADDRESS>:8080`. If you've used Jenkins earlier and recently downloaded the Jenkins 2 WAR file, it will ask for a security setup.

To unlock Jenkins, follow these steps:

1. Go to the `.Jenkins` directory and open the `initialAdminPassword` file from the `secrets` subdirectory:



2. Copy the password from that file, paste it in the **Administrator password** box, and click on **Continue**, as shown here:

The screenshot shows the 'Unlock Jenkins' step of the Jenkins setup process. At the top left, there's a 'Getting Started' link. The main title 'Unlock Jenkins' is displayed prominently. Below it, a note states: 'To ensure Jenkins is securely set up by the administrator, a password has been written to the log (not sure where to find it?) and this file on the server: /root/.jenkins/secrets/initialAdminPassword'. A red text link to this file is shown. Below this, a placeholder text 'Please copy the password from either location and paste it below.' is present. There is a text input field labeled 'Administrator password' with a placeholder 'Enter password here'. At the bottom right of the form area is a blue 'Continue' button.

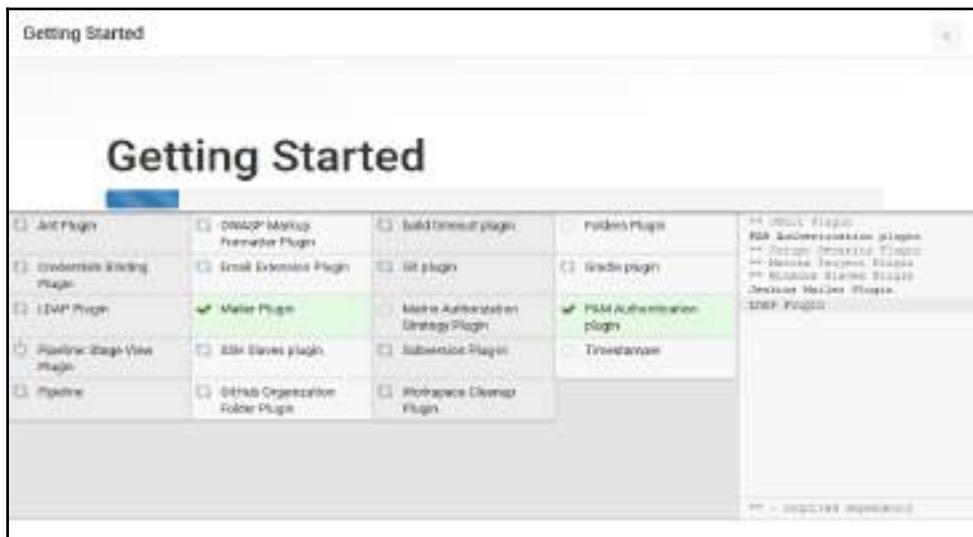
3. Clicking on **Continue** will redirect you to the **Customize Jenkins** page. Click on **Install suggested plugins**:

The screenshot shows the 'Customize Jenkins' configuration page. At the top left, there's a 'Getting Started' link. The main title 'Customize Jenkins' is displayed prominently. Below it, a note states: 'Plugins extend Jenkins with additional features to support many different needs.' Two options are presented in boxes:

- Install suggested plugins**: A light blue box containing the text 'Install plugins the Jenkins community finds most useful.'
- Select plugins to install**: A white box containing the text 'Select and install plugins most suitable for your needs.'

At the bottom right of the configuration area is a blue 'Continue' button.

4. The installation of the plugins will start. Make sure that you have a working Internet connection:



5. Once all the required plugins have been installed, you will see the **Create First Admin User** page. Provide the required details, and click on **Save and Finish**:

The screenshot shows the 'Create First Admin User' configuration page. It has two tabs: 'User name' and 'Description'. Under 'User name', the 'Username' field is filled with 'admin' and the 'Password' field contains '\*\*\*\*\*'. Under 'Description', the 'Full name' field is filled with 'DevOps Testbed' and the 'Email address' field is empty. At the bottom right, there are two buttons: 'Cancel or abort' and a blue 'Save and Finish' button.

6. Jenkins is ready! Our Jenkins setup is complete. Click on **Start using Jenkins**:



Get Jenkins plugins from <https://wiki.jenkins-ci.org/display/JENKINS/Plugins>.

## The Jenkins dashboard

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

Let's understand the dashboard parameters:

- **New Item:** It is used to create a new build job, pipeline, or build flow in Jenkins 2:

A screenshot of the Jenkins dashboard. At the top, there is a navigation bar with the Jenkins logo, a search bar, and user information for 'DiscoverTechno'. Below the bar, on the left, is a sidebar with links: 'New Item', 'People', 'Build History', 'Manage Jenkins', 'Credentials', and 'My Views'. The main content area has a heading 'Welcome to Jenkins!' with the subtext 'Please create new jobs to get started.' Below this are two expandable sections: 'Build Queue' (which shows 'No builds in the queue.') and 'Build Executor Status' (which shows '1 idle' and '2 idle'). At the bottom of the page, there is a footer with the text 'Page generated: Apr 27, 2018 11:48:51 AM PDT RECENT ACTIVITY Jenkins v2.0'.

- **Manage Jenkins:** It allows a Jenkins 2 administrator to manage plugins, users, security, nodes, credentials, global tool configuration, and so on:

The screenshot shows the Jenkins Manage Jenkins page. On the left, there's a sidebar with links: New Item, People, Build History, Manage Jenkins (which is selected), Credentials, and My Views. Below that are sections for Build Queue (No builds in the queue) and Build Executor Status (1 idle, 2 idle). The main content area is titled "Manage Jenkins" and contains several configuration links:
 

- Configure System:** Configure global settings and paths.
- Configure Global Security:** Secure Jenkins; define who is allowed to access/alter the system.
- Global Tool Configuration:** Configure tools, their locations and automatic installation.
- Reload Configuration from Disk:** Discard all the loaded data in memory and reload everything from file system. Useful when you modified config files directly on disk.
- Manage Plugins:** Add, remove, disable or enable plugin that can extend the functionality of Jenkins.
- System Information:** Displays various environmental information to assist trouble-shooting.
- System Log:** Streamline log capture output from jenkins.log and log4j.xml related to Jenkins.

 There's also a prominent blue button at the top right: "Upgrade to 2.8+ now!"

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

The screenshot shows the Jenkins Nodes page. On the left, there's a sidebar with links: Back to Dashboard, Manage Jenkins, New Node, and Configure. Below that are sections for Build Queue (No builds in the queue) and Build Executor Status (1 idle, 2 idle). The main content area is titled "Nodes" and contains a table of existing nodes:
 

S	Name	Architecture	Clock Difference	Free Disk Space	Free Swap Space	Free Temp Space	Response Time
1	master	Linux (parallel)	0 ms	8.07 GB	1.03 GB	8.07 GB	0 ms

 The table shows one node named "master". A "Refresh status" button is located at the bottom right of the table. The "Data obtained" timestamp is shown as "16 min" ago.

Now that we have installed Jenkins and become familiar with the Jenkins dashboard, the next step is to configure different tools that are used for build execution and create a base for continuous integration.

In the following sections, we will install and configure Java, Maven, and Ant.

## Configuring Java and Maven in Jenkins

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

## Configuring Java

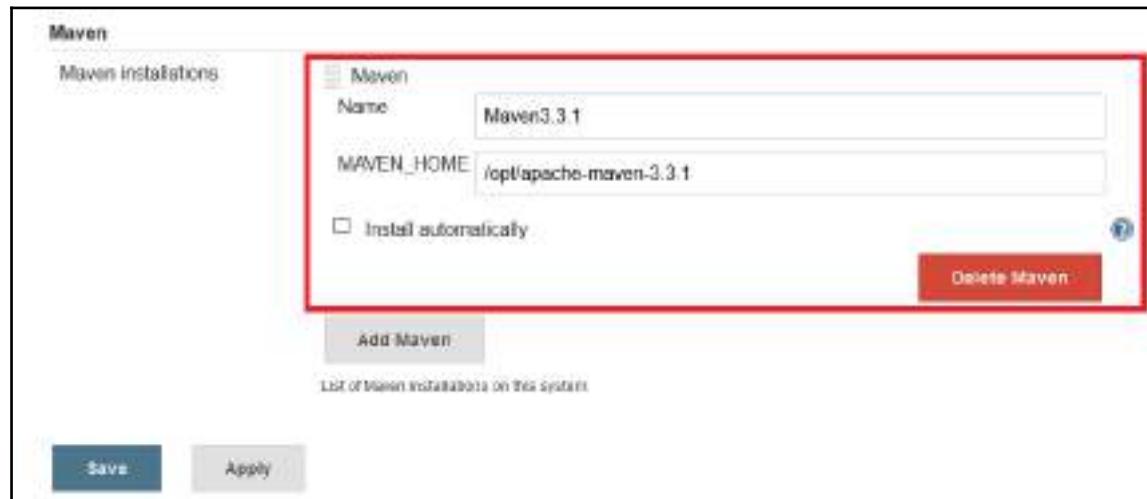
To configure Java, provide a **Name** and the **JAVA\_HOME** path, or check **Install automatically** checkbox:

The screenshot shows the Jenkins Global Tool Configuration interface. The 'JDK' section is highlighted with a red box. It contains a 'JDK installations' table with one entry. The entry shows 'Name' as 'jDK17' and 'JAVA\_HOME' as 'c:\Program Files\Java\jdk-17\openjdk-17.0.1\bin'. There is also an unchecked checkbox for 'Install automatically'. At the bottom are 'Save' and 'Apply' buttons.

JDK installations
<input checked="" type="checkbox"/> jDK17 JAVA_HOME: c:\Program Files\Java\jdk-17\openjdk-17.0.1\bin

## Configuring Maven

To configure Maven, download the Maven installer from <https://maven.apache.org/download.cgi>, and extract it to the directory on your Jenkins virtual machine. In the **Global Tool Configuration** section, provide the **Name** and **MAVEN\_HOME** path, or check **Install automatically** checkbox:



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

## Creating and configuring a build job for a Java application with Maven

Jenkins builds configured with Maven understands how Maven works and what is required in terms of execution. It uses `pom.xml` to set up and create package files from the source files.

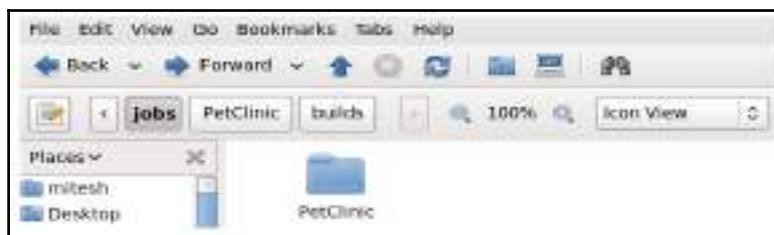
Now, let's perform steps to create and configure a new build job. Go to the Jenkins dashboard and click on **New Item**.

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

1. Enter an item name, such as PetClinic, then select **Freestyle project**. Now click on OK to continue:



2. Let's verify what this operation does. Go to the Jenkins home directory, and navigate to the jobs directory. We can see that the directory has been created for the newly created job with the same name, as shown in the following screenshot:



# Configuring and authenticating source code on GitHub

The next step is to configure a source code repository with the build job. We will use the open source Spring application hosted on GitHub, as explained in the previous chapter:

1. After that, we will get a URL similar to <https://github.com/mitesh51/spring-petclinic>.
2. Create a GitHub account and fork repository from <https://github.com/spring-projects/spring-petclinic>.

Install Git on a virtual machine using the instructions available in the documentation:



*Getting Started – Installing Git* (<https://git-scm.com/book/en/v2/Getting-Started-Installing-Git>).

To download a Windows application navigate to <https://git-scm.com/> and click on **Downloads for Windows**.

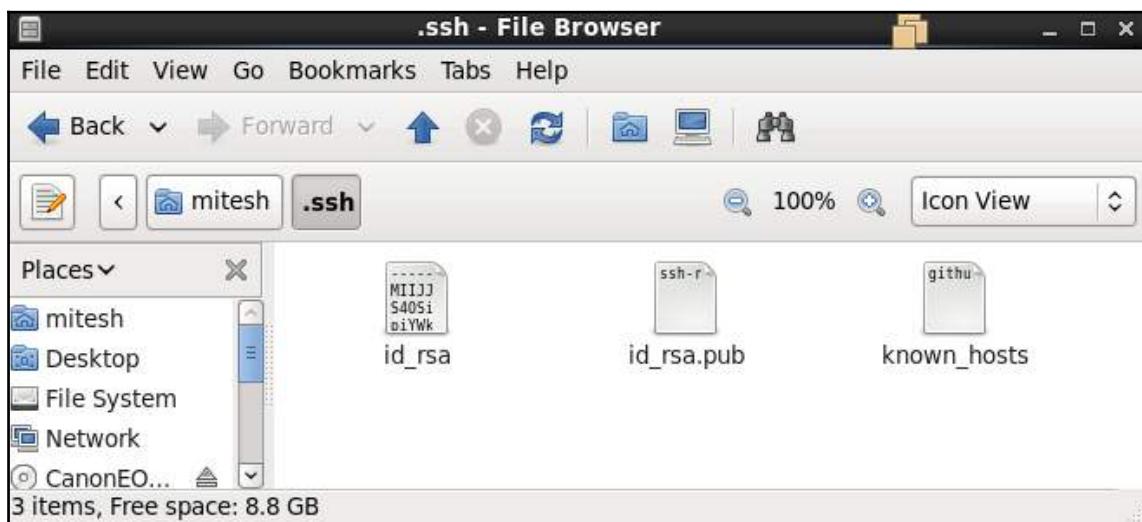
3. Let's generate a new SSH key to use for authentication. Open a terminal on a CentOS virtual machine with Git installed.
4. Run `ssh-keygen -t rsa -b 4096 -C "your_email@example.com"`, substituting your GitHub e-mail address.
5. Press *Enter* when you are prompted with **Enter file in which to save the key:**

```
[mithesh@devops1 git]$ ssh-keygen -t rsa -b 4096 -C "██████████@gmail.com"
Generating public/private rsa key pair.
Enter file in which to save the key (/home/mithesh/.ssh/id_rsa):
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/mithesh/.ssh/id_rsa.
Your public key has been saved in /home/mithesh/.ssh/id_rsa.pub.
The key fingerprint is:
d5:48:73:9f:94:d8:02:32:75:5d:c8:08:da:33:2b:5d mithesh.soni83@gmail.com
The key's randomart image is:
+-- [ RSA 4096] ---+
|   0.*oo*+.|
|   * *+o*.|
|   . * E.o |
|   o =     |
|   S o     |
|           .|
+-----+
[mithesh@devops1 git]$ █
```

6. Add your SSH key to ssh-agent:

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

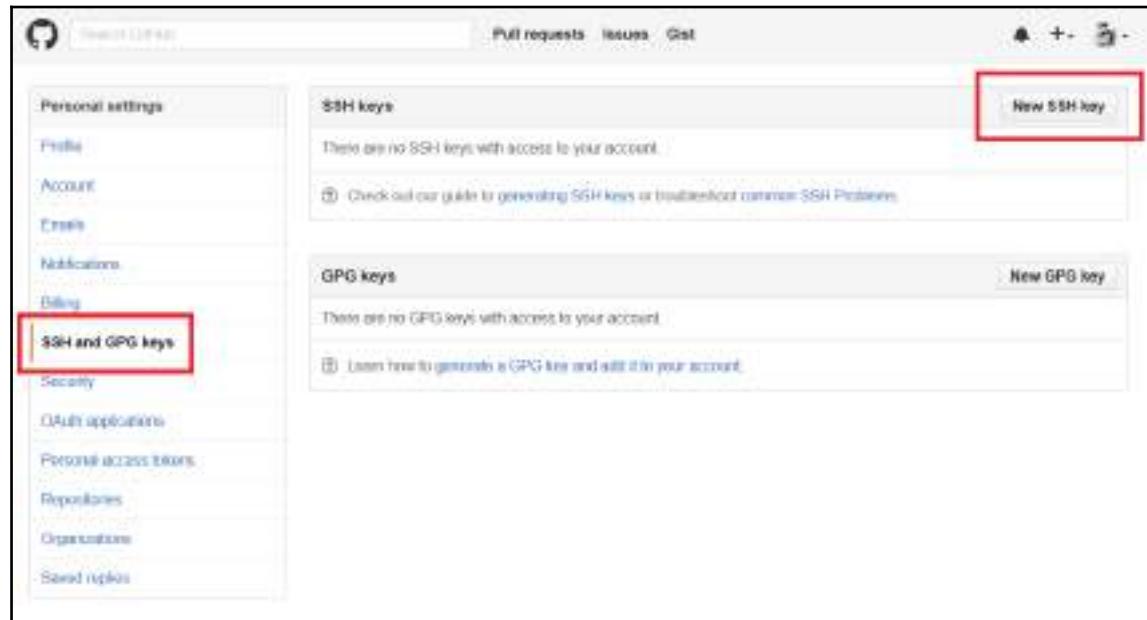
7. Verify the newly generated keys in the .ssh folder:



8. To configure your GitHub account to use the new SSH key, add it to your GitHub account. Visit your account page and click on **Settings**:



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

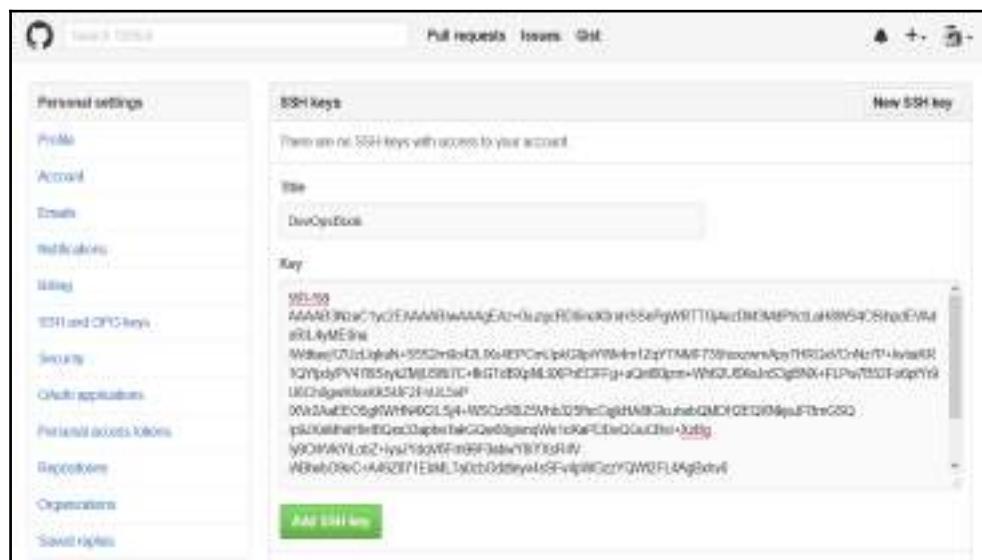


10. Open `/.ssh/id_rsa.pub` in a text editor your virtual machine, and copy the content:

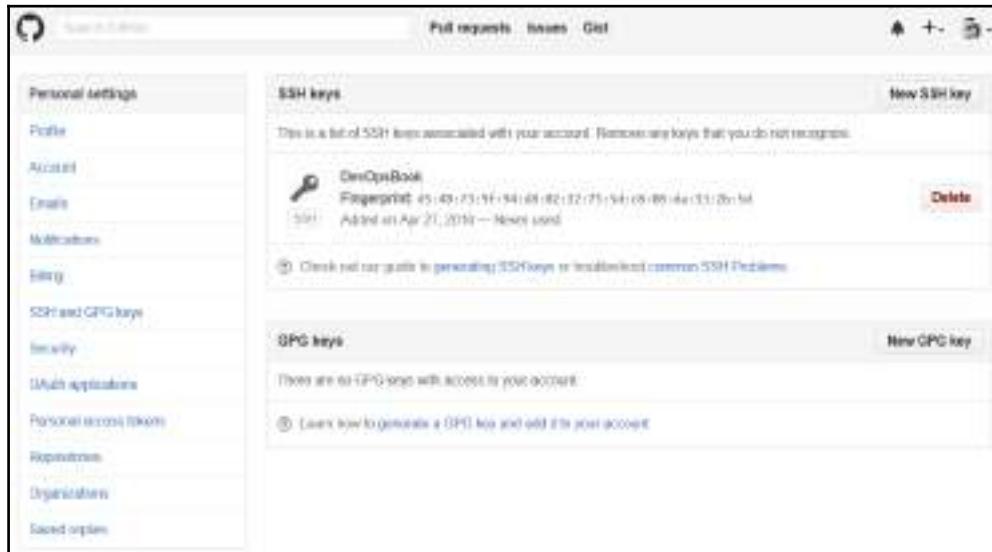
The screenshot shows a text editor window titled 'id\_rsa.pub'. The content of the file is a long string of characters representing an RSA public key. The text editor interface includes a toolbar at the top with icons for file operations, and a status bar at the bottom showing 'Plain Text', 'Tab Width: 8', 'Ln 1, Col 1', and 'INS'.

```
ssh-rsa AAAAB3NzaC1yc2EAAAABIwAAAgEAz
+0iuZgcRD0irex0rxHSSnPgWRRTGj4ezDM3MdPt/ctLaH8WS40SihpEVAdoR/
L4yME0na/Wdtaej1ZUzLlqkaN
+S5S2m9o42LIXs4EPiCmUpkG8piYWk4m1ZqYTNMF735hsxzwmApy7HRGxVCnNz7P
+lwIiaKR1QYtjxIyPV478lSrykZMjUSfb7C+fkGTdBXpNL9XPnEDFFg+uQnt60jzm
+Wh62U6XeJnS3gBNX+FLPw7B52Fo6ptYr9U8Ch0gwKfvxKKSUF2FnUL5vP/
XVr2AaEE05gKWHN4X2LSj4
+WS0zS6IZ5Vhb325fhcCqjkHA8tGlcuhebQMDH2E0KNkjeJFBmGSQ/
p9JXeMhdHIsfBQxx33apbeTakGQw80gisnqWe1oXaPDDeQGuCBsI+XzBg/
y90ihKyiLobZ+iyuJYdqV6Fm99F0abwY8iYXsR4V/ABtwb09oC
+A49Z871EbML7a0cb0ddteyv4sSFv4pWGzzYQWt2FL4AgBxhv0/
f3hA9x62xqqmo9xHrS0y8Qp6Jiah5ovZq67f1C7+
+NbYPfsqfuBtMNumiMVVz10MU1bxKonCW1VfwYcL8MRAZf2+ylpawEc
```

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



12. Verify the added SSH key:



13. Now, let's verify authentication. Open terminal and type `ssh -T git@github.com` and press *Enter*. If we get successfully authenticated then we can access Git repository without credentials:

The screenshot shows two terminal windows side-by-side. The left window has the title 'mitesh@devops1:~/Desktop' and contains the command '[mitesh@devops1 git]\$ ssh -T git@github.com'. The right window has the title 'mitesh@devops1:~/Desktop/git' and displays the message 'Hi mitesh51! You've successfully authenticated, but GitHub does not provide shell access.' followed by a prompt '[mitesh@devops1 git]\$'. Both windows have a standard Linux terminal interface with a blue vertical scroll bar on the right.

## Configuring build job

Now that Git authentication is done with, let's configure a PetClinic build job:

1. Click on the **PetClinic** build job on the Jenkins dashboard. Then, click on the **Configure** link. You'll see the following page as shown here:

The screenshot shows the Jenkins configuration page for the 'PetClinic' job. The top navigation bar includes tabs for General, Source Code Management, Build Triggers, Build Environment, Build, and Post-build Actions. The 'General' tab is selected. The configuration fields include:

- Project name:** PetClinic
- Description:** (empty text area)

At the bottom of the configuration section, there are links for [Plain text] and Preview.

2. Under **Source Code Management**, provide the GitHub URL for the sample Spring project we forked earlier, as shown in the following screenshot:

The screenshot shows the 'Source Code Management' configuration page for a Jenkins job. The 'Source Code Management' tab is selected. Under the 'Repositories' section, the 'Git' option is chosen. A single repository is configured with the URL `https://github.com/mibesh51/spring-petcclinic.git`. The 'Credentials' dropdown is set to 'none'. There are buttons for 'Advanced...', 'Add...', and 'Add Repository'.

3. We will configure **Build Triggers** and the **Build Environment** as shown here:

The screenshot shows the 'Build Triggers' and 'Build Environment' configuration pages for a Jenkins job named 'PetClinic'.  
**Build Triggers:**  
The 'Build Triggers' tab is selected. Configuration options include:

- Trigger builds remotely (e.g., from scripts)
- Build after other projects are built
- Build periodically
- Build when a change is pushed to GitHub
- Poll SCM

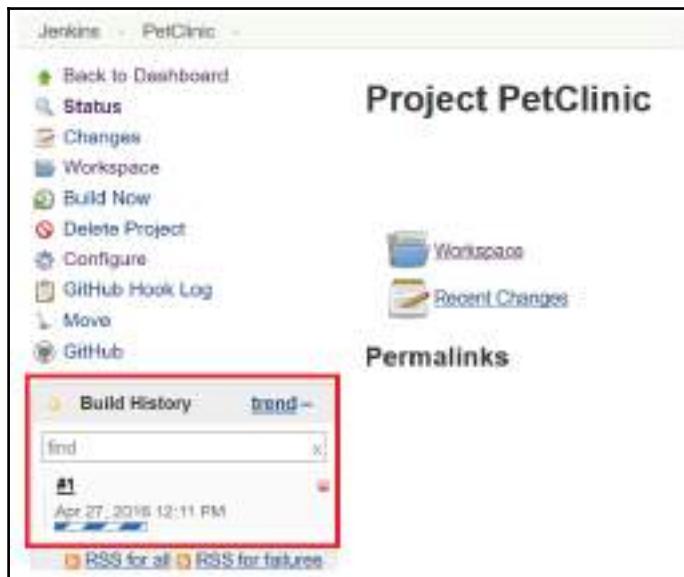
**Build Environment:**  
The 'Build Environment' tab is selected. Configuration options include:

- Delete workspace before build starts
- Abort the build if it's stuck
- Add timestamps to the Console Output
- Use secret text(s) or file(s)

- Under **Build**, click on **Add build step** and select **Invoke top-level Maven targets**. Select the **Maven Version** we configured in **Global Tools Configuration**. Enter the **Maven target** and click on **Save**:



- Let's manually trigger the build by clicking on **Build Now**. After the build is complete, you'll see this:



6. Click on the build number, the one with the # symbol. Open **Console Output**. Verify the Git operations executing before **Maven target** execution:

The screenshot shows the Jenkins interface for a build named 'PetClinic' (Build #1). The left sidebar has links for Back to Project, Status, Changes, and Console Output (which is selected). The main area is titled 'Console Output' and shows the command-line output of the build process. It starts with 'Started by user DiscoverTechno' and then details the git operations used to fetch code from GitHub. These include cloning the 'spring-petclinic' repository, fetching upstream changes, and checking out a specific revision. The output ends with commands to parse the checkout and perform a git checkout.

```

Started by user DiscoverTechno
Building in workspace /home/mitsuh/.jenkins/workspace/PetClinic
> git rev-parse --is-inside-work-tree # timeout=10
Fetching changes from the remote Git repository
> git config remote.origin.url https://github.com/spring-projects/spring-petclinic.git # timeout=10
Fetching upstream changes from https://github.com/spring-projects/spring-petclinic.git
> git --version # timeout=10
> git fetch --tags --progress https://github.com/spring-projects/spring-petclinic.git +refs/heads/*:refs/remotes/origin/*
> git rev-parse refs/remotes/origin/master^{commit} # timeout=10
> git rev-parse refs/remotes/origin/origin/master^{commit} # timeout=10
Checking out Revision 44b591f537aae6ebbe0399beab6d38ba8214c (refs/remotes/origin/master)
> git config core.sparsecheckout # timeout=10
> git checkout -f 44b591f537aae6ebbe0399beab6d38ba8214c

```

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

The screenshot shows the Jenkins interface for the same 'PetClinic' build (#1). The console output shows the Maven build process. It starts with '[INFO] --- maven-war-plugin:2.3:war {default-war} # spring-petclinic ---'. It then details the assembly of a webapp ('[INFO] Packaging webapp'), processing of the war project ('[INFO] Processing war project'), copying of webapp resources ('[INFO] Copying webapp resources'), and the successful assembly of the webapp ('[INFO] Webapp assembled in [13697 ms]'). The build then moves on to '[INFO] Building war: /home/mitsuh/.jenkins/workspace/PetClinic/target/petclinic.war'. It concludes with '[INFO] BUILD SUCCESS', a summary of the build time ('[INFO] Total time: 43s 14 min'), the finish time ('[INFO] Finished at: 2016-04-21T12:15:29-07:00'), final memory usage ('[INFO] Final Memory: 21M/214M'), and a final 'Finished: SUCCESS' message.

```

[INFO] --- maven-war-plugin:2.3:war {default-war} # spring-petclinic ---
[INFO] Packaging webapp
[INFO] Assembling webapp [spring-petclinic] in [/home/mitsuh/.jenkins/workspace/PetClinic/target/spring-petclinic-4.2.5-SNAPSHOT]
[INFO] Processing war project
[INFO] Copying webapp resources [/home/mitsuh/.jenkins/workspace/PetClinic/src/main/webapp]
[INFO] Webapp assembled in [13697 ms]
[INFO] Building war: /home/mitsuh/.jenkins/workspace/PetClinic/target/petclinic.war
[INFO] 
[INFO] BUILD SUCCESS
[INFO] 
[INFO] Total time: 43s 14 min
[INFO] Finished at: 2016-04-21T12:15:29-07:00
[INFO] Final Memory: 21M/214M
[INFO] 
Finished: SUCCESS

```

Page generated: Apr 27, 2016 12:12:15 PM PDT [RENT A PI](#) Jenkins ver. 2.0

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 the `.jenkins` folder under the specific build job:

The screenshot shows the Jenkins workspace for the 'PetClinic' project on the 'master' branch. The left sidebar includes links for Back to Dashboard, Status, Changes, Workspace (selected), Wipe Out Current Workspace, Build Now, Delete Project, Configure, GitHub Hook Log, Move, GitHub, and Build History. The workspace tree shows the following structure:

- jenkins
- HTTP://WATERSHED
- git
- gitignore
- gitmodules
- boom
- editorconfig
- gitignore
- springBeans
- root.xml
- bower.json
- mine
- mine.cmd
- pom.xml
- readme.md
- sonar-project.properties

File sizes and last modified times are listed next to each file. At the bottom, there are RSS feed links for all and failures, and a link to a full file in zip.

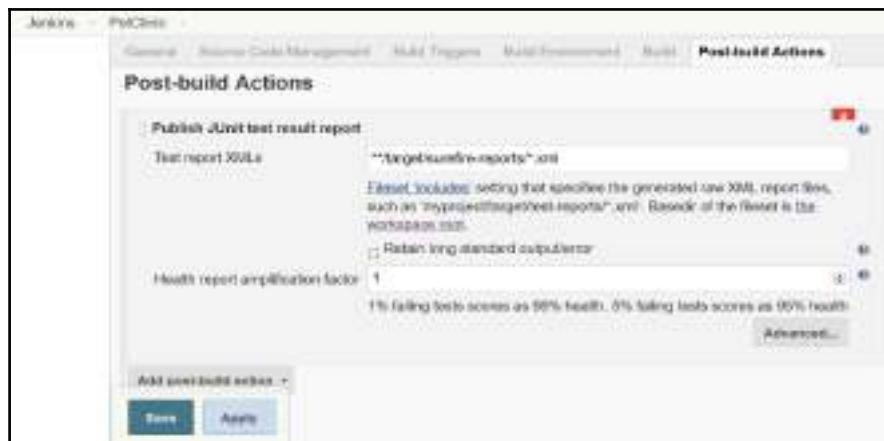
Page generated: Apr 27, 2016 12:16:52 PM PDT Jenkins ver. 2.0

## Configuring JUnit

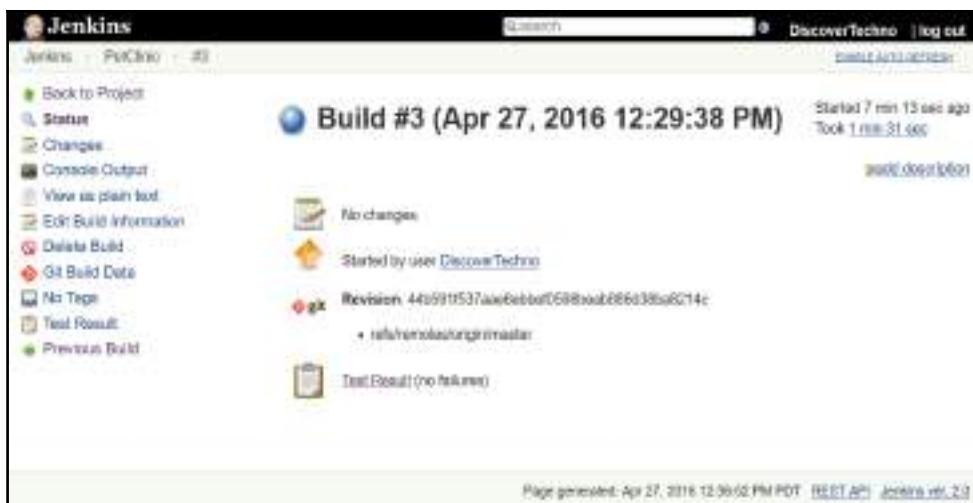
Our sample application has JUnit test cases, and to execute them, we need to configure JUnit-related settings in the build job configuration:

1. Under **Post-build Actions**, select **Publish JUnit test result report**.
2. Provide a path for **Test report XMLs** based on the workspace.

3. Click on **Apply** and then click on **Save**:



4. After you've configured the JUnit settings for the build, wait for a scheduled build execution, or click on **Build Now**.
5. Verify the build status on the Jenkins dashboard and you will see the **Test Result** link with a small summary. Click on **Test Result**:



6. Verify all test execution statuses package wise. The page also provides information related to duration and failed test cases:

The screenshot shows the Jenkins Test Result page for build #3. The left sidebar contains links like Back to Project, Status, Changes, Console Output, View as plain text, Edit Build Information, History, Git Build Data, No Tags, Test Result (which is selected), and Previous Build. The main content area has a title 'Test Result' and a summary bar showing 0 failures, 59 tests, and a duration of 17 sec. Below this is a section titled 'All Tests' with a table showing test results by package. The table has columns for Package, Duration, Fail, N/A, Skip, N/A, Pass, N/A, Total, and N/A. The data is as follows:

Package	Duration	Fail	N/A	Skip	N/A	Pass	N/A	Total	N/A
org.springframework.samples.petclinic.model	5.3 sec	0		0		1 +1		1	+1
org.springframework.samples.petclinic.services	4.8 sec	0		0		30 +30		30	+30
org.springframework.samples.petclinic.web	7.5 sec	0		0		28 +28		28	+28

At the bottom, it says 'Page generated: Apr 27, 2016 12:35:12 PM PDT | REST API | Jenkins ver. 2.0'.

In the next section, we will cover the **Dashboard View** plugin, which helps us customize the view for build jobs.

## The Dashboard View plugin – overview and usage

**Dashboard View** plugin provides a different view implementation, based on a portal kind of layout. We can select different build jobs to be included in a new view and configure different portlets for the view.

To configure it, follow these steps:

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

Install	Name	Version
<input type="checkbox"/> Dashboard View	Dashboard View	2.9.7
<input type="checkbox"/> Mission Control Plugin	Mission Control Plugin	0.6.3

[Install without restart](#)   [Download now and install after restart](#)

Update information obtained: 16 h

Page generated: Apr 27, 2016 12:38:24 PM PDT REST API Jenkins ver. 2.0

2. Once the plugin has been installed successfully, we can create a new view by clicking on the + sign on the Jenkins dashboard.
3. Enter a **View name**, select the view type, and click on **OK**:

**OK**

Page generated: Apr 27, 2016 12:47:25 PM PDT REST API Jenkins ver. 2.0

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

**Dashboard Portlets**

- Show standard Jenkins list at the top of the page
- Full screen view - hide standard Jenkins panels
- Set CSS custom parameters

**Portlets at the top of the page**

Add Dashboard Portlet to the top of the view

**Portlets in the left column**

<b>Test Statistics Chart</b>
Display name <input type="text" value="Test Statistics Chart"/>
<span style="float: right;">Delete</span>

5. Add different portlets based on your requirements into the view, and save it. Here's a sample view:

**Jenkins** DiscoverTechno | log out

**PetClinic-First**

S	W	Name	Last Success	Last Failure
8	W	PetClinic	7 min 37 sec · 55	N/A

Icon: B M L

Legend: RSS for all RSS for failure RSS for just latest builds

**Test Statistics Chart**

**Test Statistics Grid**

Job	Success #	%	Failed #	%	Skipped #	%	Total #
PetClinic	59	100%	0	0%	0	0%	59
Total	59	100%	0	0%	0	0%	59

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



Now, we'll look at one of the most popular features of Jenkins: **distributed builds**.

Consider a scenario where you want different Java applications that need different JDK versions to compile source files. How do you manage such a situation effectively? We'll find out in the next section.

## Managing nodes

Jenkins provides a master-slave concept for managing the aforementioned scenarios. We can assign different build jobs to different slaves in the build configuration and use the master-slave system to manage its overall lifecycle. The master node itself can execute the build if a slave node is not configured explicitly in the build job configuration.

There are quite a few reasons for using this feature:

- Build jobs require resources, and they compete for resource availability
- A different runtime environment is required for different build jobs
- It distributes the load across slave nodes

To make things clearer, we need not install Jenkins on the *slave nodes*. We only need to configure the slave nodes properly, which we will now cover.

The only requirements are the following:

- The configurations and runtime environment have to be available on the slave node
- The path needs to be configured correctly on the master node for the runtime environments or tools used by the slave node for execution.

## Creating and configuring slave node in Jenkins 2

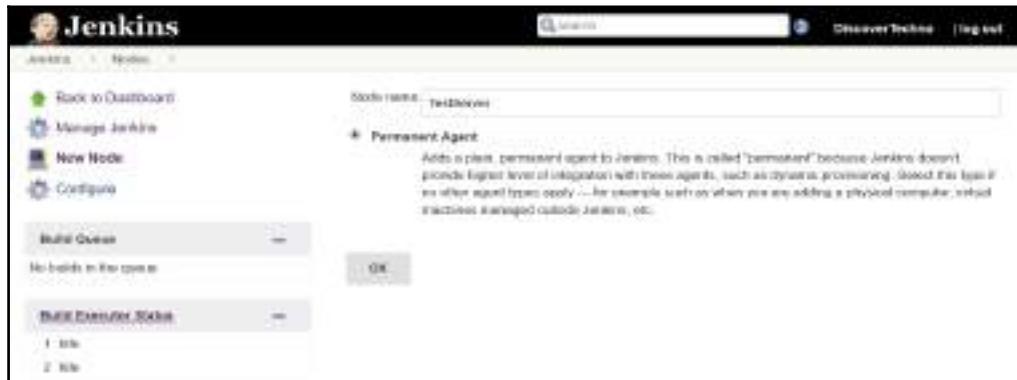
The following steps will guide you to create and configure a slave node in Jenkins 2:

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

The screenshot shows the Jenkins dashboard with the 'Manage Jenkins' link highlighted in blue. The dashboard includes sections for 'Build Queue' (empty), 'Build Executor Status' (1 idle, 2 busy), and a table for monitoring the master node's status.

#	Name	Architecture	Clock Difference	Free Disk Space	Free Swap Space	Free Temp Space	Response Time
1	master	Linux (parallel)	0 ms	8.07 GB	1.03 GB	8.87 GB	0 ms
	Data obtained	16 min	16 min	16 min	16 min	16 min	16 min

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



3. The next step is to configure the newly created node. Enter a **Remote root directory**, which will store details related to the build jobs on slave nodes. Give **Labels** to this node. Labels can be used to assign different build jobs to specific slave machines:



4. In Jenkins 2, on creating a slave node and configuring it, if you get the **slaveAgentPort.disabled** error, as shown in the following screenshot, you need to solve it before advancing to the next step:

The screenshot shows the Jenkins Node configuration page for 'Agent TestServer (TestServer)'. On the left, there's a sidebar with links: Back to List, Status, Delete Agent, Configure, Build History, Load Statistics, and Log. The main area displays the node details: 'Agent TestServer (TestServer)', 'Created by DiscoverTechno', 'Labels WindowsNode', and 'Projects tied to TestServer'. A prominent red error message 'slaveAgentPort.disabled' is present, with a link to the security configuration screen.

5. To solve it, go to the **Manage Jenkins** page, and click on the **Configure Global Security** link. Select **Enable security**, and select **Fixed** or **Random** for **TCP port for JNLP agents**, and save the configuration:

The screenshot shows the 'Configure Global Security' page. It has a yellow padlock icon and the title 'Configure Global Security'. Below the title, there are three radio buttons for 'TCP port for JNLP agents': 'Fixed' (unchecked), 'Random' (checked), and 'Disable' (unchecked). At the bottom, there is a checkbox for 'Disable remember me'.

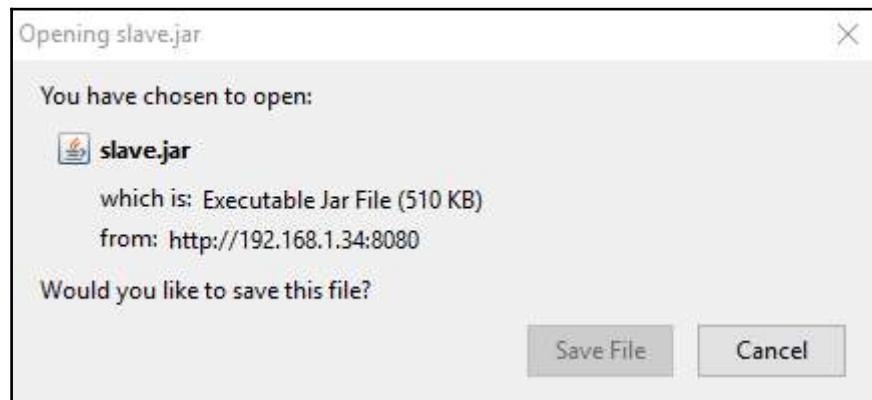
6. The next step is to connect the Jenkins slave with the Jenkins master. We will connect the agent to Jenkins using the command-line:

The screenshot shows the Jenkins interface with the 'Nodes' page selected. A specific node named 'TestServer' is highlighted. The main content area displays the 'Agent TestServer (TestServer)' configuration. It provides instructions for connecting the agent to Jenkins via a browser or command line. The command-line option is shown as:

```
java -jar slave.jar -jnlpUrl http://192.168.1.34:8080/computer/TestServer/slave-agent.jnlp -secret 6546e02c58c85b192883f7848ad2758408220bed2f3af715c01c9b01cb72f9b
```

Below this, there are sections for 'Labels' (WindowsNode) and 'Projects tied to TestServer' (None).

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



8. Execute the following code from the command-line on the slave node:

```
java -jar slave.jar -jnlpUrl  
http://192.168.1.34:8080/computer/TestServer/slave-agent.jnlp -secret  
6546e02c58c85b192883f7848ad2758408220bed2f3af715c01c9b01cb72f9b
```

```
C:\Users\MItesh\Downloads>java -jar slave.jar -jnlpUrl http://192.168.1.34:8080/computer/TestServer/slave-agent.jnlp -secret 65464e02c58c85b192883f7848ad2758408220bed2f3af715c01c9b01cb72f9b
May 04, 2016 5:38:44 PM hudson.remoting.jnlp.Main createEngine
INFO: Setting up slave: TestServer
May 04, 2016 5:38:44 PM hudson.remoting.jnlp.Main$CuiListener <init>
INFO: Jenkins agent is running in headless mode.
May 04, 2016 5:38:44 PM hudson.remoting.jnlp.Main$CuiListener status
INFO: Locating server among [http://192.168.1.34:8080/]
May 04, 2016 5:38:44 PM hudson.remoting.jnlp.Main$CuiListener status
INFO: Handshaking
May 04, 2016 5:38:44 PM hudson.remoting.jnlp.Main$CuiListener status
INFO: Connecting to 192.168.1.34:44559
May 04, 2016 5:38:44 PM hudson.remoting.jnlp.Main$CuiListener status
INFO: Trying protocol: JNLP3-connect
May 04, 2016 5:38:45 PM hudson.remoting.jnlp.Main$CuiListener status
INFO: Server didn't accept the handshake: Unknown protocol:Protocol:JNLP3-connect
May 04, 2016 5:38:45 PM hudson.remoting.jnlp.Main$CuiListener status
INFO: Connecting to 192.168.1.34:44559
May 04, 2016 5:38:45 PM hudson.remoting.jnlp.Main$CuiListener status
INFO: Trying protocol: JNLP2-connect
May 04, 2016 5:38:45 PM hudson.remoting.jnlp.Main$CuiListener status
INFO: Connected
```

9. Verify the status of the slave node from the Jenkins dashboard:



## Agent TestServer (TestServer)

Connected via JNLP agent.

Created by [DiscoverTechno](#)

**Labels**

[WindowsNode](#)

**Projects tied to TestServer**

None

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

The screenshot shows the Jenkins 'Nodes' page. It lists two nodes: 'master' (Linux (amd64)) and 'TestServer' (Windows 8 (amd64)). The table includes columns for Name, Architecture, Clock Difference, Free Disk Space, Free Swap Space, Free Temp Space, Response Time, and Last build. The 'master' node has 8.80 GB free disk space, 1.02 GB swap, and 0 ms response time. The 'TestServer' node has 0% free disk space, 3.56 GB swap, and 256ms response time. The last row shows 'Data obtained' at 8 min 25 sec.

S	Name	Architecture	Clock Difference	Free Disk Space	Free Swap Space	Free Temp Space	Response Time	ACTIONS
1	master	Linux (amd64)	In sync	8.80 GB	1.02 GB	0.00 GB	0ms	
2	TestServer	Windows 8 (amd64)	In sync	N/A	3.56 GB	103.27 GB	256ms	
	Data obtained	8 min 25 sec	8 min 25 sec	8 min 25 sec	8 min 22 sec	8 min 25 sec	8 min 25 sec	

## Configuring the build job for master and slave node

The following steps will guide you to configure build jobs for master and slave nodes:

1. To configure the build job to run on the master, open its build configuration, and in **General** section, select **Restrict where this project can be run**.
2. In **Label Expression**, enter the label of the Master node:

The screenshot shows the Jenkins 'Job Configuration' page for 'PetClinic'. The 'General' tab is selected. Under 'Project URL', the value is 'https://github.com/mihai5/spring-petclinic'. Below it, under 'Advanced...', there is a section titled 'Restrict where this project can be run' with a radio button labeled 'Master'. The 'Label Expression' field is set to 'Master'. A note below states 'Label is serviced by 1 node.'

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



4. To configure tools specific to the slave node, click on **Configure** in the **Manage Nodes** section. In **Node Properties**, configure **Tool Locations** for the slave node, as shown in the following screenshot:



In the next section, we will see how to configure e-mail notifications.

# Sending e-mail notifications based on build status

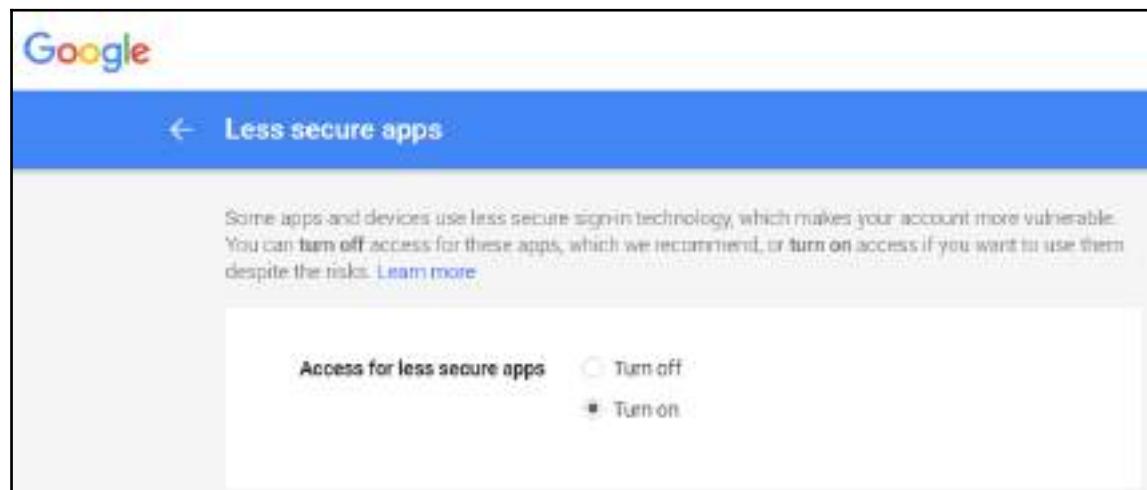
*“Failure is simply the opportunity to begin again, this time more intelligently.”*  
-Henry Ford

However, it is extremely vital to be aware of failure or at least to know when things fail so we can fix them and get rid of issues.

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

We will use G-mail configuration for setting up e-mail notifications. Follow these steps:

1. Go to <https://www.google.com/settings/u/1/security/lesssecureapps> and click on **Turn on Access for less secure apps**, as shown here, to send e-mail notifications from Jenkins 2:

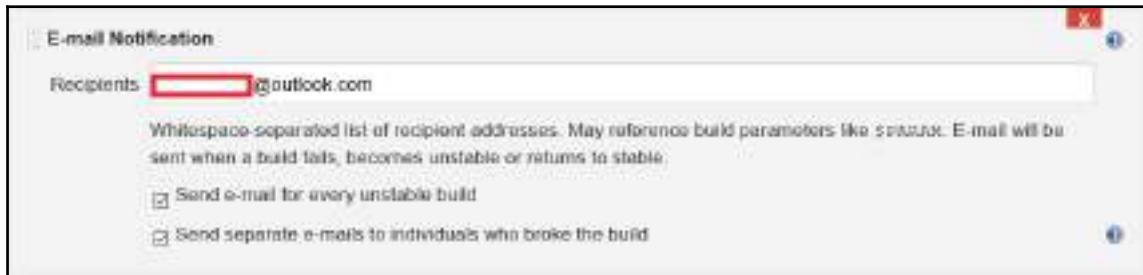


2. Do the following from the Jenkins dashboard:
  1. Click on **Manage Jenkins**, and go to the **Configure System** section.
  2. Go to the **E-mail Notification** subsection and enter the appropriate values for **SMTP Server** and **Default user e-mail suffix**.
  3. Check the **Use SMTP Authentication** box, and enter a **User Name** and **Password**.
  4. Check the **Use SSL** checkbox, and enter details for **SMTP Port** and **Reply-To Address**.
  5. Finally, select **Test configuration by sending test e-mail**. If you set everything up correctly, you will see a message saying **Email was successfully sent**:

The screenshot shows the 'E-mail Notification' configuration page within the Jenkins 'Configure System' section. The page includes fields for SMTP server (smtp.gmail.com), Default user e-mail suffix, Use SMTP Authentication (checked), User Name (redacted@gmail.com), Password (redacted), Use SSL (checked), SMTP Port (465), Reply-To Address (noreply@gmail.com), Charset (UTF-8), and a Test configuration by sending test e-mail checkbox (checked). The recipient for the test email is redacted@outlook.com. A success message 'Email was successfully sent' is displayed, along with a 'Test configuration' button.

E-mail Notification	
SMTP server	smtp.gmail.com
Default user e-mail suffix	
<input checked="" type="checkbox"/> Use SMTP Authentication	
User Name	redacted@gmail.com
Password	redacted
Use SSL	<input checked="" type="checkbox"/>
SMTP Port	465
Reply-To Address	noreply@gmail.com
Charset	UTF-8
<input checked="" type="checkbox"/> Test configuration by sending test e-mail	
Test e-mail recipient	redacted@outlook.com
Email was successfully sent	
<a href="#">Test configuration</a>	

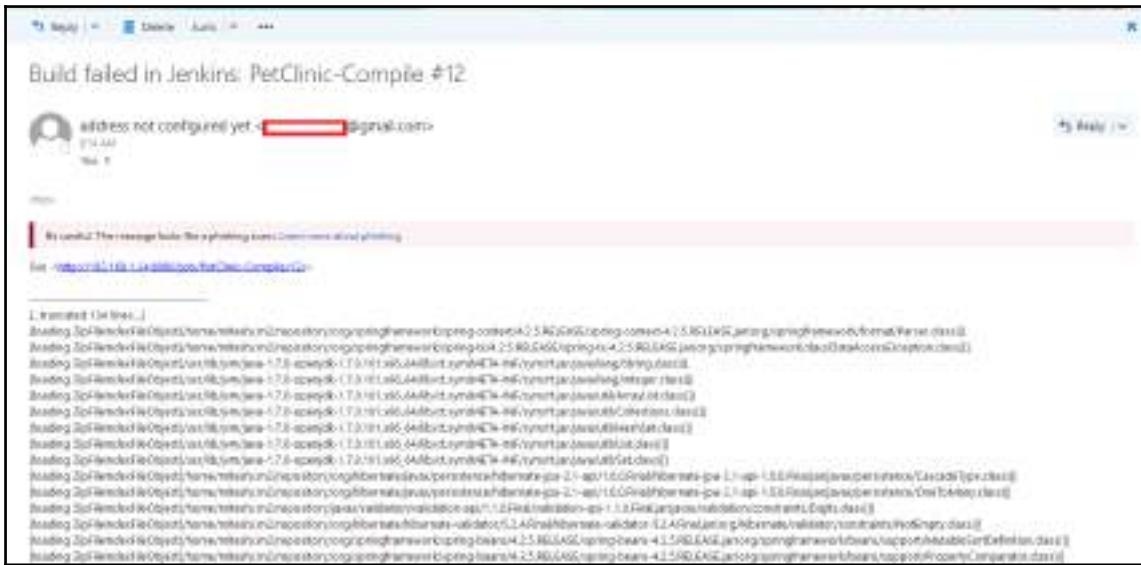
3. To verify e-mail notifications, simulate a failure in a build job. Open any build job and click on **Configure**.
4. In **Post-build Actions**, click on **Add Post-build Action** and configure it like this:
  1. Select **E-mail Notification**.
  2. Enter a list of **Recipients**.
  3. Select **Send e-mail for every unstable build** and **Send separate e-mails to individuals who broke the build**:



In our case, we execute a compile goal against the Maven build and we wanted to publish a JUnit test result to simulate a failure. We can see that the compilation of files is successful but the post-build action fails, and it triggers an e-mail 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: [REDACTED]@outlook.com  
Finished: FAILURE
```

The following is the e-mail received. It contains a stack trace of the execution:



Consider a scenario where you want to send customized content in the e-mail. How would you achieve that?

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



# Integrating Jenkins and Sonar

SonarQube is an open source tool for managing the code quality of an application. It manages seven axes of code quality, such as architecture and design, duplications, unit tests, potential bugs, complexities, coding rules, and comments. It covers programming languages and formats such as ABAP, C/C++, C#, COBOL, CSS, Erlang, Flex/ActionScript, Groovy, Java, JavaScript, JSON, Objective-C, PHP, PL/I, PL/SQL, Puppet, Python, RPG, Swift, VB.NET, Visual Basic 6, and XML. One of the most striking features is its extensibility. It is easy to cover new languages and add rule engines using an extension mechanism in the form of plugins.

To install the SonarQube plugin, follow these steps:

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

Install	Name	Version
<input type="checkbox"/>	<a href="#">CodeSonar Plugin</a>	1.0.1
<input checked="" type="checkbox"/>	<a href="#">SonarQube Plugin</a> This plugin allows easy integration of <a href="#">SonarQube</a> ™, the open source platform for continuous inspection of code quality.	2.4
<input type="checkbox"/>	<a href="#">Sonargraph Integration Jenkins Plugin</a> This plugin integrates <a href="#">Sonargraph</a> version 8 and newer into your build. Sonargraph allows to define an architecture for a software system and automatically checks how the code base conforms to it. For Sonargraph version 7 use <a href="#">Sonargraph Plugin</a> .	1.0.3
<input type="checkbox"/>	<a href="#">Sonargraph Plugin</a> This plugin integrates <a href="#">Sonargraph</a> version 7 into your build. Sonargraph allows to	1.8.4

**Install without restart**    **Download now and install after restart**    Last update information obtained

2. Download Sonar from <http://www.sonarqube.org/downloads/>.
3. Extract the installable directory from the ZIP file and go into the `bin` subdirectory.
4. Select the installable directory based on your OS, and run the `StartSonar.*` file, as shown here:

```
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\Installables\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:+UseConcMarkSweepGC -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.SearchServer C:\Users\MIteh\AppData\Local\Temp\sq-process700072261932287622properties
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.tmpdir=D:\##DevOps Book\Installables\sonarqube-5.4\temp -cp ./lib/common/*;./lib/server/*;D:\##DevOps Book\Installables\sonarqube-5.4\lib\jdbc\h2\h2-1.3.176.jar org.sonar.server.app.WebServer C:\Users\MIteh\AppData\Local\Temp\sq-process301913882364693273properties
jvm 1 | 2016.04.29 23:59:07 INFO app[o.s.p.m.Monitor] Process[web] is up
```

- Once Sonar is up and running, open a browser and visit `http://localhost:9000/` or `http://<IP_Address>:9000/`. You will get the Sonar dashboard:

The screenshot shows the SonarQube Administration interface. At the top, there's a navigation bar with links like Dashboards, Issues, Metrics, Rules, Quality Profiles, Quality Gates, Administrations, and more. On the right side of the header is a user icon labeled "Administrator". Below the header, the main area is titled "Administration" and has a sub-section for "Users". It says "Create and administer individual users." and includes a "Create User" button. There's also a search bar. The "USERS" section lists one user: "Administrator admin" with a green profile picture. The "SON ACCOUNTS" section is empty. The "GROUPS" section shows two groups: "sonar-administrators" and "sonar-users". The "TOKENS" section shows 0 tokens available. At the bottom, there's a footer with the text "1/1 pages".

An important step for Jenkins 2 and Sonar integration is the **security token**:

- Go to the **My Account** link in the top-right corner.
- Click on the Security tab and then on **Generate Tokens**:

The screenshot shows the "Tokens" page. The title is "Tokens". There's a table with columns "NAME" and "CREATED". A message "No tokens" is displayed. Below the table is a section titled "Generate Tokens" with a form. It has a text input field "Enter Token Name" and a blue "Generate" button. In the bottom right corner of the page, there's a blue "Done" button.

3. Enter a token name, and click on **Generate**. Copy the token value and click on **Done**:

The screenshot shows the Jenkins Tokens page. At the top, there is a table with one row containing a token named "ms9883" created on April 30, 2016. Below this is a "Generate Tokens" section with a text input field for "Enter Token Name" and a "Generate" button. A message box displays: "New token 'ms9883' has been created. Make sure you copy it now, you won't be able to see it again!" Below the message is a "Copy" button next to the token value "213862af16b6b71d846a00fa5045b9f2d4575fe5". At the bottom right is a "Done" button.

4. Verify the **TOKENS** column in the on dashboard:

The screenshot shows the SonarQube Administration - Users page. The user "Administrator [admin]" has a token listed in the "TOKENS" column: "213862af16b6b71d846a00fa5045b9f2d4575fe5".

5. Once we have a security token ready, we need to integrate Jenkins and Sonar:
6. In the **Manage Jenkins** section, click on **Configure System**, and add SonarQube servers. Here, provide a **Server URL** and **Server authentication token**, and save the settings:

**SonarQube servers**

Environment variables	<input type="checkbox"/> Enable injection of SonarQube server configuration as build environment variables <small>If checked, job administrators will be able to inject a SonarQube server configuration as environment variables in the build.</small>
SonarQube installations	Name Sonar5.4
	Server URL http://localhost:9000
	Server version Default is http://localhost:9000 5.3 or higher
	Server authentication token SonarQube authentication token. Mandatory when anonymous access is disabled. [REDACTED]

7. In **Global Tool Configuration**, configure **SonarQube Scanner installations** as well:

**SonarQube Scanner**

SonarQube Scanner installations	SonarQube Scanner Name: SonarQube Scanner <input checked="" type="checkbox"/> install automatically
	Install from Maven Central Version: SonarQube Scanner 2.5.1
	<a href="#">Delete Installer</a>

Once all Sonar-related installations and configurations are completed, we need to add a build step to execute **SonarQube Scanner**. Run the build job with these steps:

1. We need `sonar-project.properties` to configure Sonar with a specific application. In our sample application, the `sonar-project.properties` file is already available, as shown here:

```
# 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
```

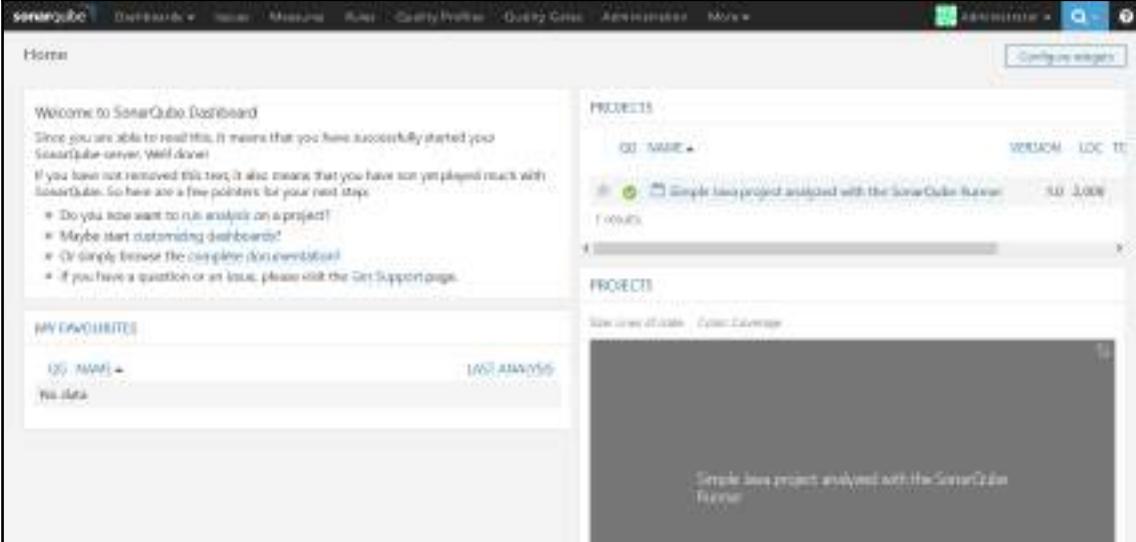
2. Verify the console output of a build job for Sonar execution:

```
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\MTesh\.sonar\cache
INFO: Load global repositories
INFO: Load global repositories (done) | time=1131ms
INFO: User cache: C:\Users\MTesh\.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\AbstractCl
inicServiceTests.java
    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
```

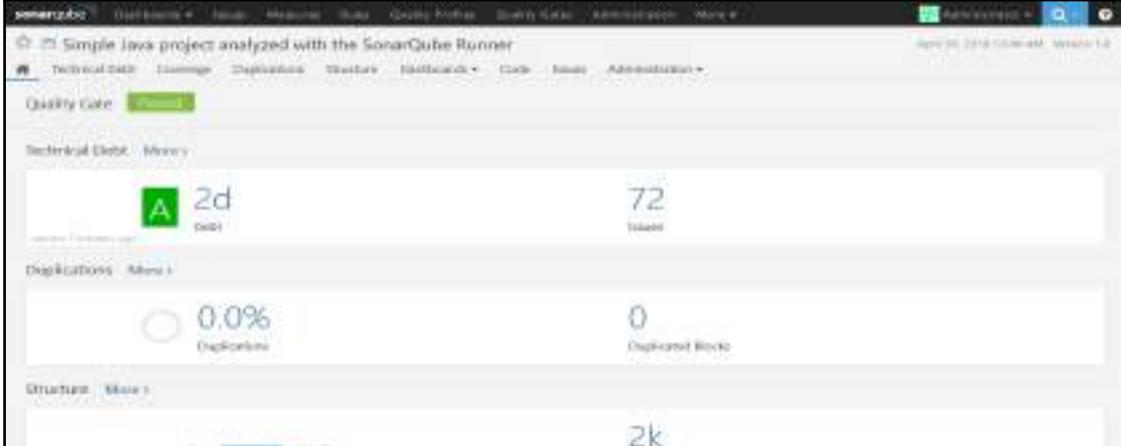
```
INFO: Java Test Files AST scan done: 1 ms
INFO: Sensor JavaSquidSensor (done) | time=15295ms
INFO: Sensor Lines Sensor
INFO: 56/56 source files have been analyzed
INFO: 0/0 source files have been analyzed
INFO: Sensor Lines Sensor (done) | time=28ms
INFO: Sensor QProfileSensor
INFO: Sensor QProfileSensor (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:



The screenshot shows the SonarQube Dashboard. On the left, there's a sidebar with 'Home', 'Welcome to SonarQube Dashboard', 'MY FAVORITES' (with 'QA NAME' and 'Project'), and 'LAST ANALYSIS'. The main area has a 'PROJECTS' header. Under it, there's a card for 'Simple Java project analyzed with the SonarQube Runner'. The card displays 'VERSION: 1.0.0-SNAPSHOT' and 'LOC: 11'. Below the card, there's a 'PROJECT' section with a large dark gray box containing the text 'Simple Java project analyzed with the SonarQube Runner'.

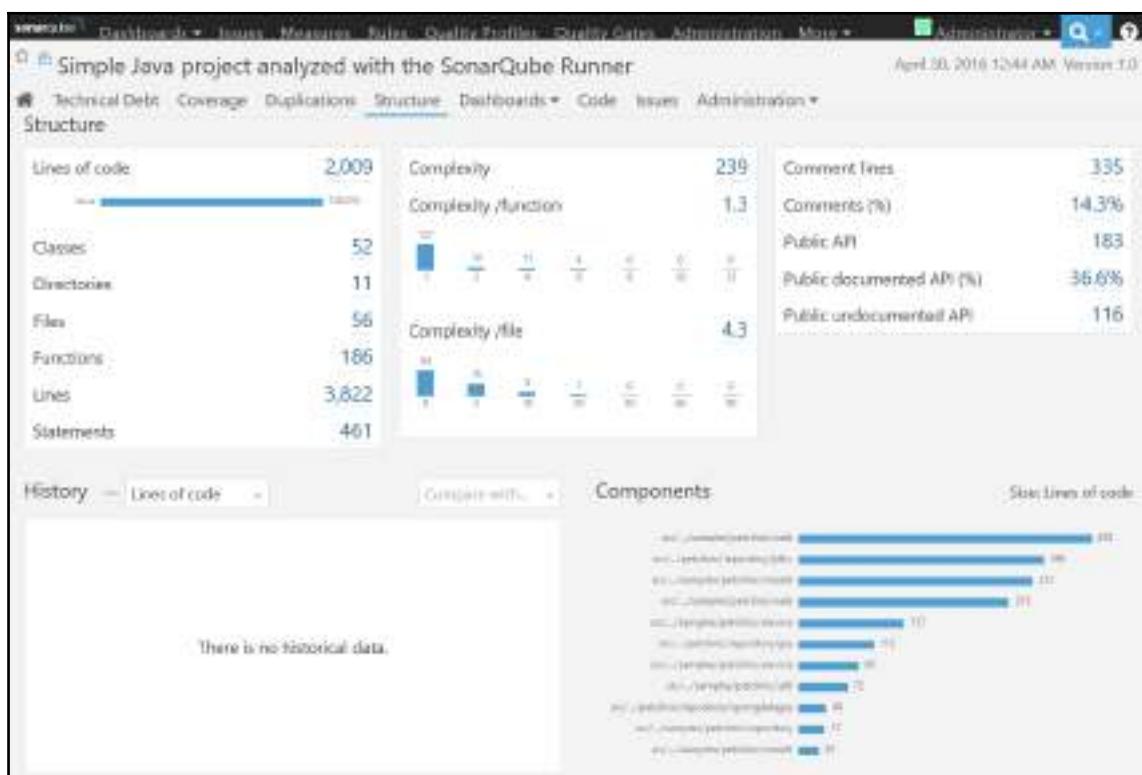
5. We can see the result of the analysis here. **Quality Gate** shows passed. It provides details about **Technical Debt**, **Duplications**, and **Structure** too:



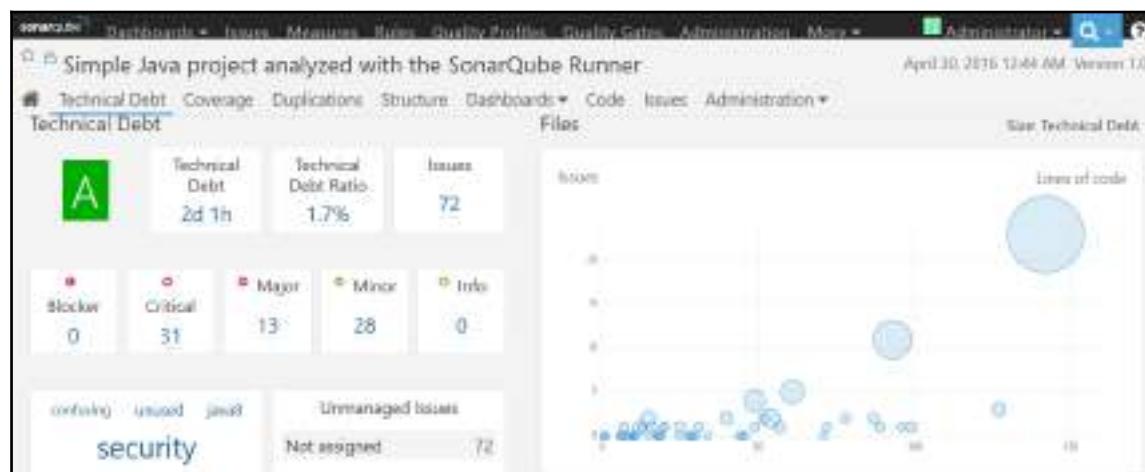
The screenshot shows the 'Simple Java project: analyzed with the SonarQube Runner' analysis page. At the top, there's a navigation bar with 'Dashboard', 'Issues', 'Measures', 'Rules', 'Quality Profile', 'Quality Gates', 'Administrators', 'More', and a date/time stamp 'April 30, 2018 11:06 AM - Version 1.0'. Below the navigation is a 'Quality Gate' section with a green 'Passed' button. Underneath are three main sections: 'Technical Debt - Money' (with a green 'A' icon, '2d debt', and a '72' score), 'Duplications' (with a yellow '0.0%' icon and '0 Duplicate Blocks'), and 'Structure' (with a blue '2k' icon). The bottom of the page has a footer with 'SonarQube' and 'SonarQube 5.6'.

6. Quality gates can be defined in the Sonar dashboard. We have used the default quality gate here:

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



8. To get more insight into issues in specific files, click on the **Technical Debt** tab, and click on the bubbles on the chart:



Sonar stores historical data in 24-hour slices.



## Self-test questions

1. Jenkins is written in Java.
  - True
  - False
2. On which of the following operating systems can Jenkins be installed?
  - Ubuntu/Debian
  - Windows
  - Mac OS X
  - CentOS/Fedora/Red Hat
  - All of these

3. Which of the following commands can be used to change the default port on which Jenkins runs?

- `java -jar jenkins.war --httpPort=9999`
- `java -jar jenkins.war --http=9999`
- `java -jar jenkins.war --https=9999`
- `java -jar jenkins.war --httpsPort=9999`

4. Sonar stores historical data in 22-hour slices.

- True
- False

## Summary

In this chapter, we learned about some new features in Jenkins 2, why Jenkins is so popular, and how to install it. We discussed the improvements with respect to security and plugin installations during setup and how to configure Java and Maven. We took a look at what happens in the background when we create a new job in Jenkins, how to authenticate with Git, and how to configure Git in Jenkins. We then performed a unit test execution in a sample Spring application and configured the Dashboard View plugin with different portlets for customized views. We then learned how to manage the master and slave nodes for load distribution and managing different environments as required, how to configure e-mail notifications for build status, and how to integrate Sonar and Jenkins.

In the next chapter, we will look at one of the most important aspects in terms of the orchestration of the end-to-end pipeline of application delivery. We will discuss the pipeline concept of Jenkins 2 and the build pipeline plugin.

It is the proper time to quote *Ralph Waldo Emerson*, as it is relevant in the context of failures during 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 the Build Pipeline

*“Start wide, expand further, and never look back.”*

*-Arnold Schwarzenegger*

It is always better to start early and visualize the things we want to achieve. That is the objective of this chapter. It will be easy to realize the importance of this chapter when we are at the last line of the final chapter of this book.

One of the highlights of Jenkins 2 is built-in support for delivery pipelines. We know that Jenkins is a continuous integration server, but what if we wanted to use it for continuous delivery or continuous deployment too? Automation and orchestration both are equally important while dealing with the application delivery pipeline.

This chapter describes in detail how to create the pipelines of different jobs for a sample **Java Enterprise Edition (Java EE)** application. It will also cover the deployment of an application to a local web or application server and the configuration of a build pipeline for the lifecycle of continuous integration. This way, Jenkins users can model application delivery pipelines as the code. Once we make it into code, we can store in a code repository and it can be managed in a better way. An important benefit is a 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 the lifecycle of continuous integration, including pulling code from a code repository, building the code, executing unit tests, and static code analysis using different jobs.

We will cover the following topics:

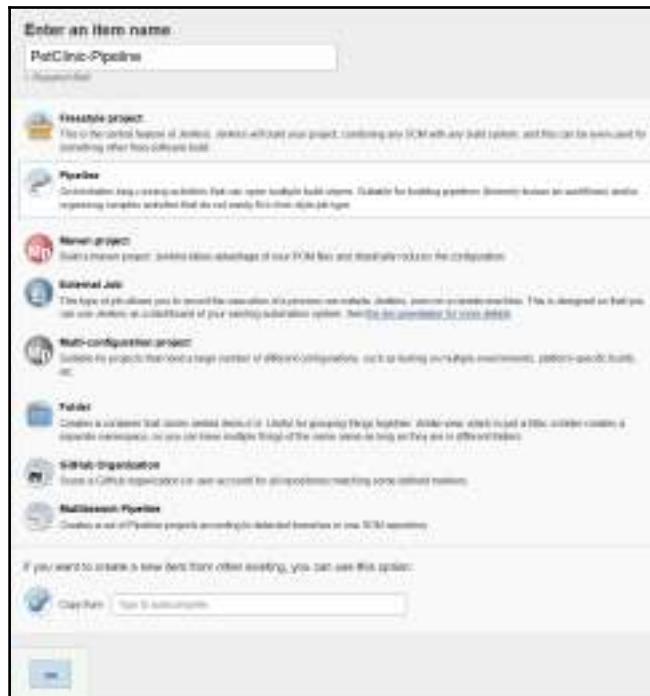
- The built-in delivery pipelines of Jenkins 2
- Build pipeline configuration for end-to-end automation for managing the lifecycle of continuous integration
- Deploying a WAR file from Jenkins to a local Tomcat server

## Creating built-in delivery pipelines

Jenkins 2 provides a way to create delivery pipelines using a **domain-specific language (DSL)**.

The steps for creating a built-in delivery pipeline are as follows:

1. Go to the Jenkins dashboard and click on **New Item**.
2. Enter an item name, say **PetClinic-Pipeline**, and select **Pipeline**, as shown in the following screenshot, and click on **OK**:



3. In case you have an existing pipeline available, you can create a new pipeline by copying from it.
4. Go to **Advanced Project Options**. For the purpose of learning, input echo 'Hello from Pipeline Demo' in the **Script** box.
5. Click on **Save** to save the configuration:



6. As we haven't created any stage, we will get a warning, as shown in the following screenshot. However, we can execute the pipeline for demo purposes:

A screenshot of the Jenkins Pipeline configuration page for 'PetClinic-Pipeline'. The title is 'Pipeline PetClinic-Pipeline'. Below the title, there is a 'Stage View' section with a yellow warning message: 'This Pipeline has run successfully, but does not define any stages. Please use the stage step to define some stages in this Pipeline.' Underneath this, there is a 'Recent Changes' link and a 'Permalinks' section containing a bulleted list of four build links: 'Last build (#1) 2 min 17 sec ago', 'Last stable build (#1) 2 min 17 sec ago', 'Last successful build (#1) 2 min 17 sec ago', and 'Last completed build (#1) 2 min 17 sec ago'.

---

7. Click on the **Build Now**. Verify the **Console Output**. We can see the script execution completing successfully:

The screenshot shows the Jenkins interface for the 'PetClinic-Pipeline' project, specifically build #2. On the left, there's a sidebar with links: 'Back to Project', 'Status', 'Changes', 'Console Output' (which is currently selected), 'View as plain text', 'Edit Build Information', 'Delete Build', 'Replay', 'Pipeline Steps', and 'Previous Build'. The main content area is titled 'Console Output' and displays the following log output:

```
Started by user DiscoverTechno
[Pipeline] echo
Hello from Pipeline Demo
[Pipeline] End of Pipeline
Finished: SUCCESS
```

## Creating scripts

Let's go step by step and learn how we can create a script. To make things easier, refer to the 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**.



*Pipeline DSL Reference:* <https://jenkins.io/doc/pipeline/steps/>  
*Snippet Generator:* <https://jenkins.io/doc/pipeline/#using-snippet-generator>

## Example 1 – creating a Groovy script to build a job

Here's how to create a Groovy script to build a job. It triggers a new downstream job to build:

Sample step	Parameters
<b>build: Build a job</b>	<b>Project to Build:</b> PetClinic-Compile <b>Parameters:</b> None <b>Other configurations:</b> Default

The screenshot shows the Jenkins job configuration interface. The top navigation bar includes tabs for General, Build Triggers, Advanced Project Options (which is selected), and Pipeline. Below the tabs, the 'Steps' section is visible. A single step is listed: 'build: Build a job'. This step has a dropdown menu icon to its right. Underneath the step, there are configuration options: 'Project to Build' set to 'PetClinic-Compile', a checkbox for 'Wait for completion' which is checked, and another for 'Propagate errors' which is unchecked. There is also a 'Quiet period' input field and a note about parameters stating 'PetClinic-Compile is not parameterized'. At the bottom of the steps section is a 'Generate Groovy' button. Below the steps section, a preview area shows the generated Groovy code: 'build 'PetClinic-Compile''. At the very bottom are 'Save' and 'Apply' buttons.

## Example 2 – creating a build step to publish test reports

Creating a build step is used to configure post-build actions or in general build steps that are pipeline compatible, based on the drop-down list:

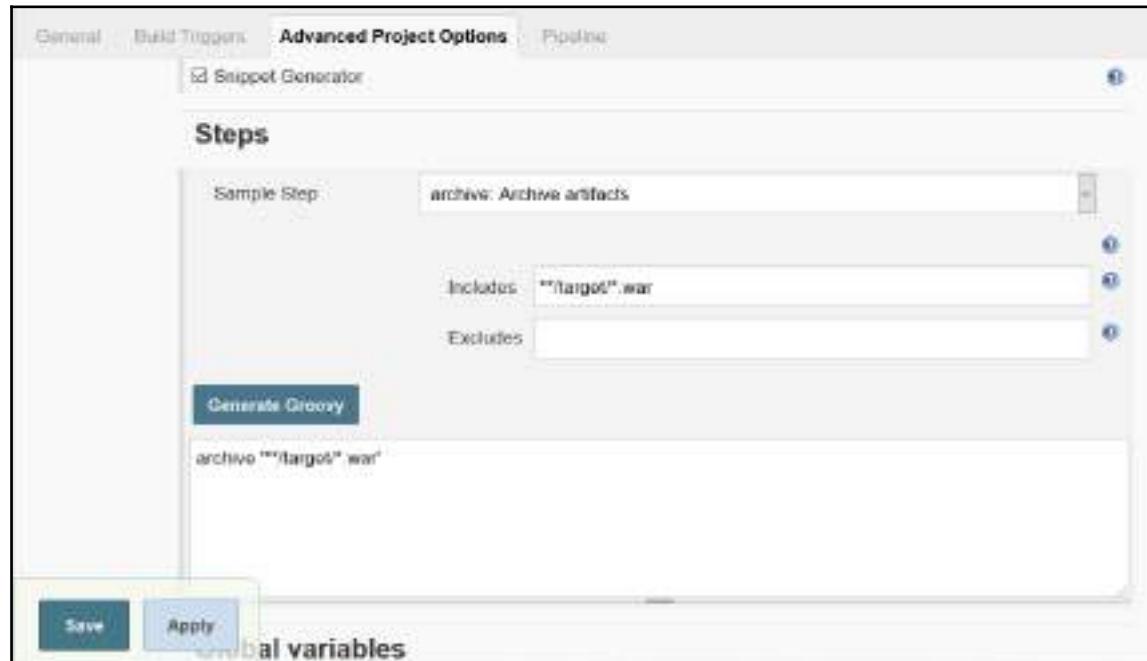
Sample step	Parameters
step: General Build Setup	<b>Build Step:</b> Publish JUnit test result report <b>Test Report XMLs:</b> **/target/surefire-reports/TEST-*.xml Other configurations: Default



## Example 3 – archiving build job artifacts

To archive build job artifacts, use the following parameters:

Sample step	Parameters
archive: Archive artifacts	<b>Includes:</b> This includes artifacts, using a comma separated list-matching Ant-style pattern for archiving artifacts <b>Excludes:</b> This excludes artifacts, using a comma separated list-matching Ant-style pattern for not archiving artifacts



## Example 4 – running a build step on a node

To run a build step on a specific node, we need to write a script. Use **Snippet Generator** and select a sample step node, and select the slave node label. Then, click on **Generate Groovy**:

Sample step	Parameters
<b>node: Allocate node</b>	<b>Label:</b> The label associated with slave node. Refer to Chapter 2, <i>Continuous Integration with Jenkins 2</i> . For more details on master-slave nodes in Jenkins 2

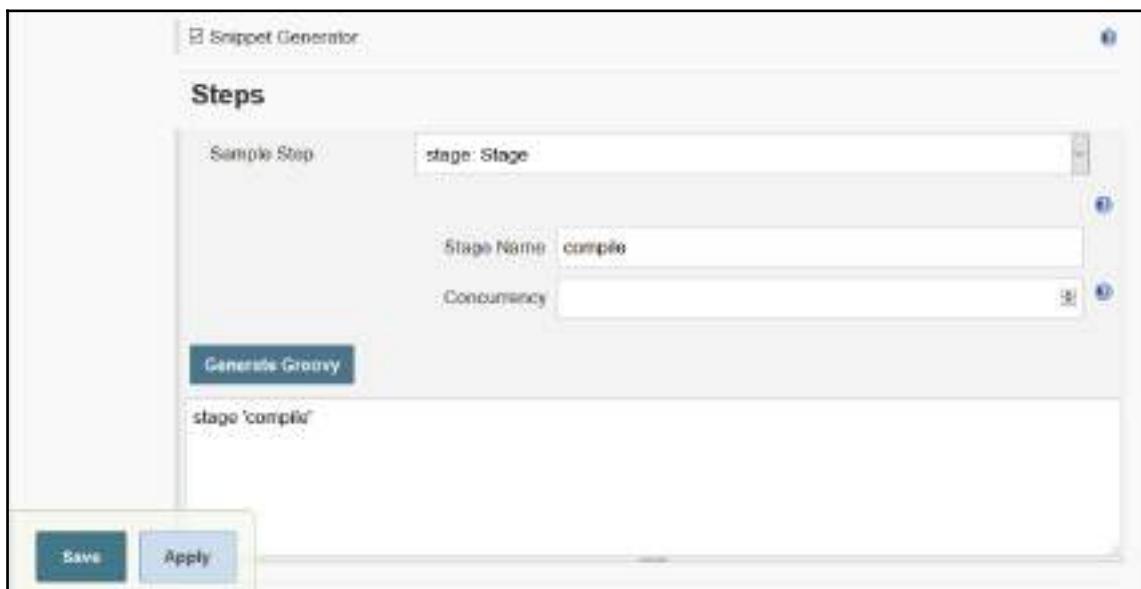
The screenshot shows the Jenkins Snippet Generator interface. In the 'Steps' section, there is a single step named 'node: Allocate node'. Below it, under 'Label', is the value 'WindowsNode'. A tooltip indicates that 'Label is serviced by 1 node'. At the bottom of the snippet editor, there is a 'Generate Groovy' button, and the generated Groovy code is displayed in a scrollable text area:

```
node('WindowsNode') {  
    // some block  
}
```

## Example 5 – marking the definite steps of a build job

We will now create a Groovy script to mark definite sections of a build job as being controlled by limited concurrency:

Sample step	Parameters
stage: Stage	<b>Stage Name:</b> Compile/Test/Deploy Other configurations: Default

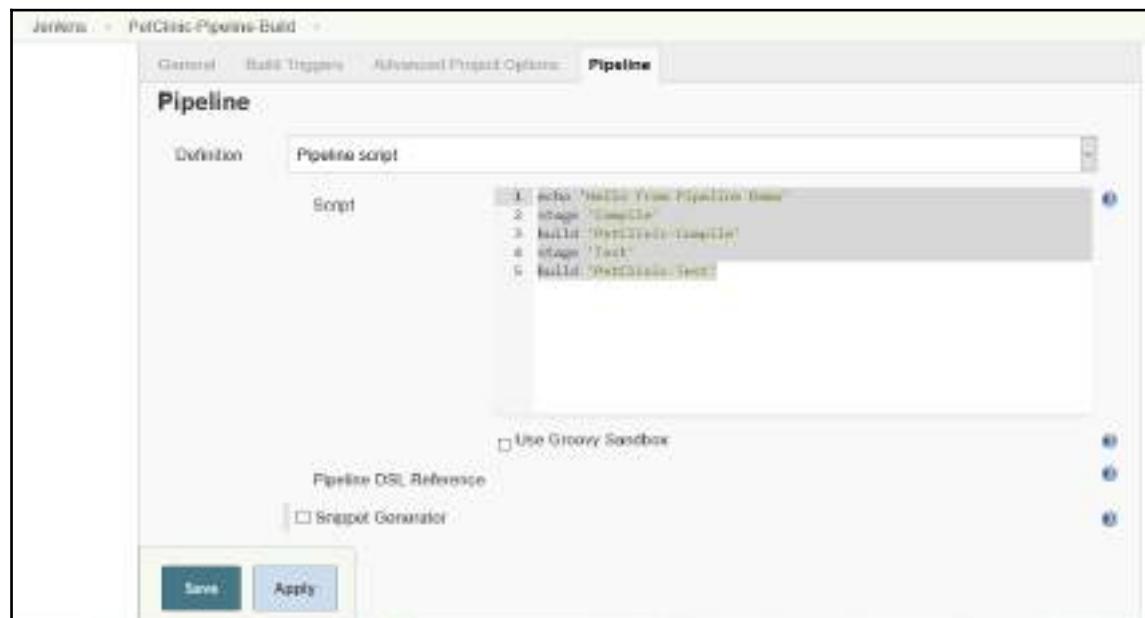


# Creating a pipeline for compiling and executing test units

For demonstration purposes, let's try a simple scenario to create a pipeline for compiling source files and executing unit test cases:

1. Let's use the following script in the **Script** box:

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



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

The screenshot shows the Jenkins interface for the 'PetClinic-Pipeline-Build' job. On the left, a sidebar lists options like Back to Project, Status, Changes, Console Output (which is selected), View as plain text, Edit Build Information, Delete Build, Reply, and Pipeline Steps. The main content area is titled 'Console Output' and displays the following log output:

```
started by user DiscoverTechno
[Tipeline] info
Hello from Pipeline Demo
[Tipeline] stage [COMPILE]
Entering stage Compile
Proceeding
[Tipeline] build [Building PetClinic-Compile]
Scheduling project: PetClinic-Compile
Starting building: PetClinic-Compile #14
[Tipeline] stage [TEST]
Entering stage Test
Proceeding
[Tipeline] build [Building PetClinic-Test]
Scheduling project: PetClinic-Test
Starting building: PetClinic-Test #10
[Tipeline] end of pipeline
Finished: SUCCESS
```

3. Go to the build job's main page. We can see **Stage View** here. Remember, we have created two stages: one is **Compile** and the other is **Test**. **Stage View** provides instant visualization. It provides details such as build completion time, the node on which the build has been executed, and whether the build has executed successfully or failed:

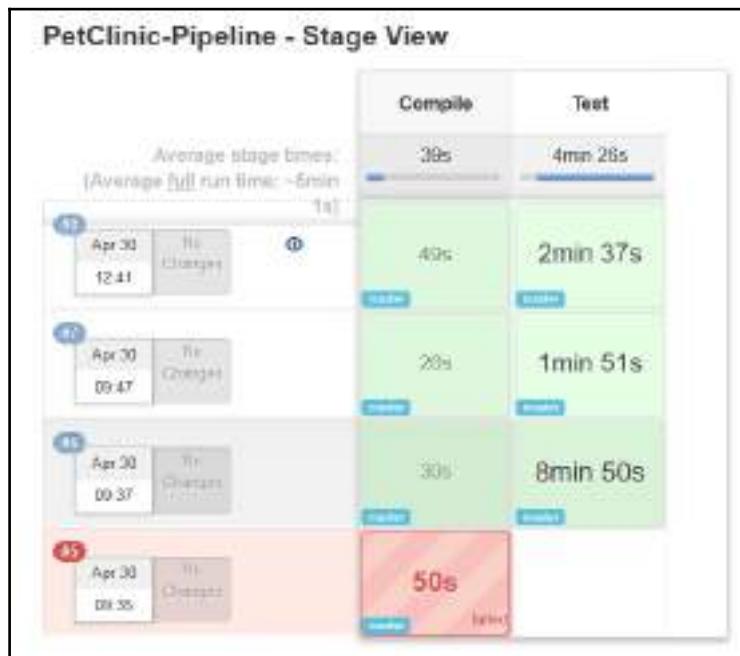


4. For a particular build execution, we can verify **Pipeline Steps** as well:

The screenshot shows the Jenkins interface for a pipeline named "PetClinic-Pipeline Build". The current build number is #1. On the left, there is a sidebar with links: Back to Project, Status, Changes, Console Output, Edit Build Information, Delete Build, Reply, and Pipeline Steps. The Pipeline Steps page lists the following steps:

Step	Status
Start of Pipeline	Success
Print Message	Success
Compile	Success
Builds PetClinic Compile	Success
Test	Success
Builds PetClinic Test	Success

5. Click on **Full Stage View** to get a full-screen view, as shown in the following screenshot:



6. To obtain details specific to a stage, mouse over a specific stage, and it will show you the status of that stage's execution as well as the **Logs** link:

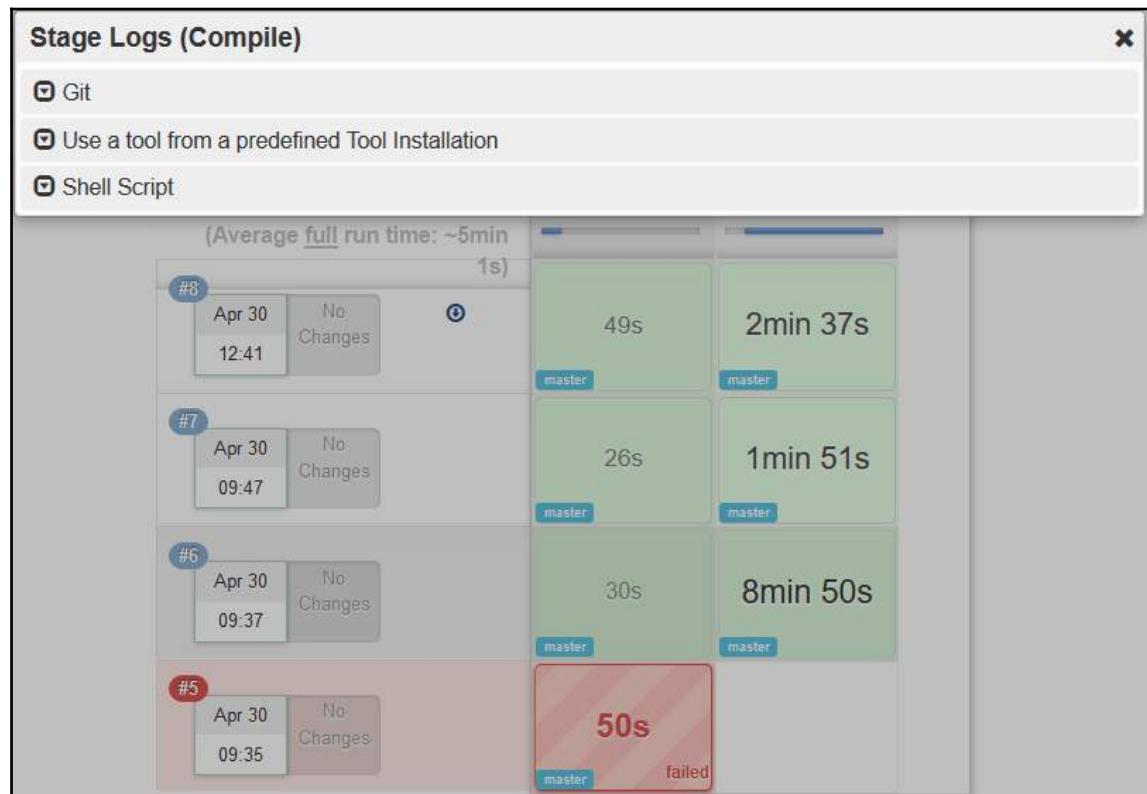
## PetClinic-Pipeline - Stage View

Average stage times:  
(Average full run time: ~5min)

#	Date	Time	Changes	Duration	Stage Status
#8	Apr 30	12:41	No Changes	1s	Success Compile <a href="#">Logs</a>
#7	Apr 30	09:47	No Changes	26s	master 2min 37s
#6	Apr 30	09:37	No Changes	30s	master 1min 51s
#5	Apr 30	09:35	No Changes	50s	master failed

The screenshot shows a Jenkins pipeline stage view for the PetClinic-Pipeline. It displays four stages (#5, #6, #7, #8) with their respective execution times and statuses. Stage #8 is highlighted with a light green background and has a tooltip showing 'Success' and 'Compile' status, along with a 'Logs' button which is being clicked by a cursor. Stage #5 is highlighted with a pink background and is labeled 'failed'. Stage #6 and #7 are also visible with their execution times and master status.

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



Now, let's consider a scenario where we want to execute different stages on different nodes.

1. Copy the following code and paste it in **Script** section:

```
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'
```

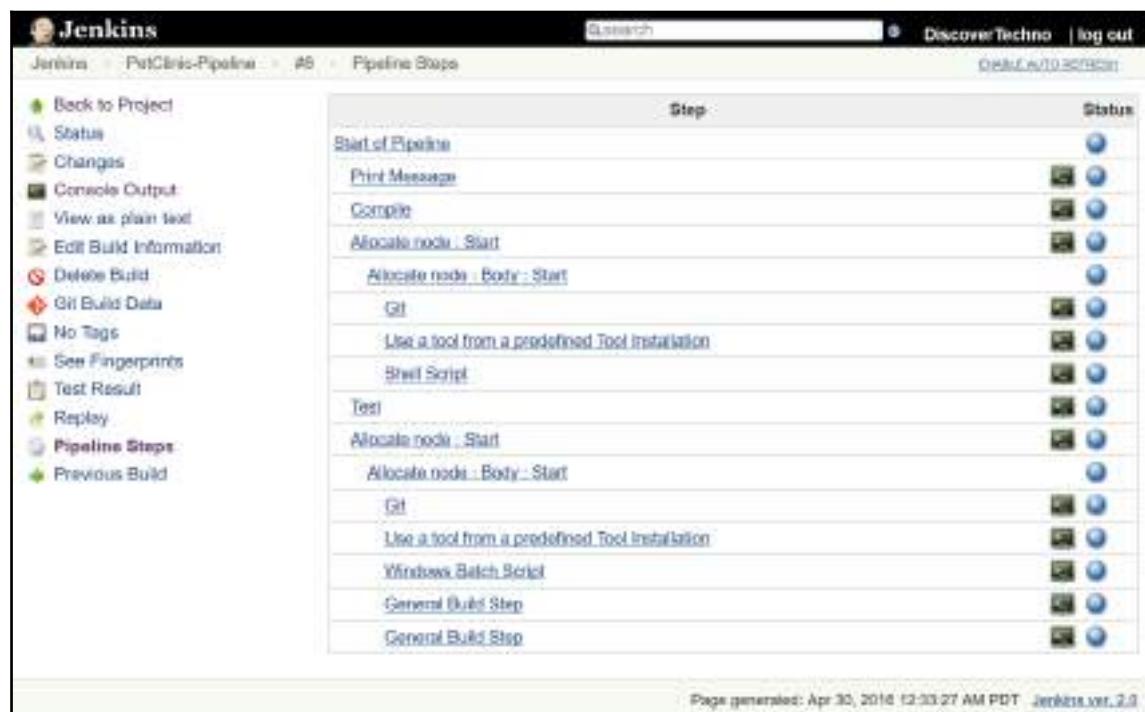
```
bat "${mvnHome}\bin\mvn -B verify"
step([$class: 'ArtifactArchiver', artifacts: '**/target/*.war',
fingerprint: true])
step([$class: 'JUnitResultArchiver', testResults:
 '**/target/surefire-reports/TEST-*.xml'])
}
```

2. Click on **Build Now** and verify the **Stage View**:

The screenshot shows the Jenkins interface for the 'Pipeline PetClinic-Pipeline' job. On the left, there's a sidebar with links like Back to Dashboard, Status, Changes, Build Now, Delete Pipeline, Configure, Move, Full Stage View, and GitHub. Below that is a 'Build History' section with a dropdown menu and a 'trend' button. The main area is titled 'Stage View' and displays a grid of build stages. Each stage has a timestamp, a 'No. Changes' link, and a duration. The stages are color-coded: light green for most recent ones, and one stage from April 30 at 09:35 is highlighted in red. To the right of the stage grid is a 'Test Result Trend' chart showing a single blue bar with a value of 40. At the bottom of the stage view grid, there are 'RSS for all' and 'RSS for failures' links.

Stage	Time	No. Changes	Duration
Apr 30, 2016 12:11 AM	0	0	39s
Apr 29, 2016	1	0	4min 26s
Apr 29, 2016	2	0	2min 37s
Apr 29, 2016	3	0	1min 51s
Apr 29, 2016	4	0	8min 50s
Apr 30, 2016 09:35	5	0	50s

3. **Pipeline Steps** describes drill-down details of execution, as shown in the following screenshot:



The screenshot shows the Jenkins Pipeline Steps configuration page for the 'PutClinic-Pipeline' project. The left sidebar contains links for Back to Project, Status, Changes, Console Output, View as plain text, Edit Build Information, Delete Build, Git Build Data, No Tags, See Fingerprints, Test Result, Replay, Pipeline Steps (which is selected and highlighted in blue), and Previous Build. The main content area is titled 'Pipeline Steps' and lists various steps with their status (green circle with a checkmark). The steps listed are: Start of Pipeline, Print Message, Compile, Allocate node\_Start, Allocate node\_Body\_Start, Git, Use a tool from a predefined Tool Installation, Shell Script, Test, Allocate node\_Start, Allocate node\_Body\_Start, Git, Use a tool from a predefined Tool Installation, Windows Batch Script, General Build Step, and General Build Step.

Step	Status
Start of Pipeline	Green (checkmark)
Print Message	Green (checkmark)
Compile	Green (checkmark)
Allocate node_Start	Green (checkmark)
Allocate node_Body_Start	Green (checkmark)
Git	Green (checkmark)
Use a tool from a predefined Tool Installation	Green (checkmark)
Shell Script	Green (checkmark)
Test	Green (checkmark)
Allocate node_Start	Green (checkmark)
Allocate node_Body_Start	Green (checkmark)
Git	Green (checkmark)
Use a tool from a predefined Tool Installation	Green (checkmark)
Windows Batch Script	Green (checkmark)
General Build Step	Green (checkmark)
General Build Step	Green (checkmark)

Page generated: Apr 30, 2016 12:33:27 AM PDT Jenkins v1.52

4. Let's verify stage logs for **Git** operation. Mouse over the **Compile** stage, and click on **logs**. Expand the **Git** dropdown, as shown in the following screenshot, to get more details:

The screenshot shows a window titled "Stage Logs (Compile)". The "Git" tab is selected. The log output is as follows:

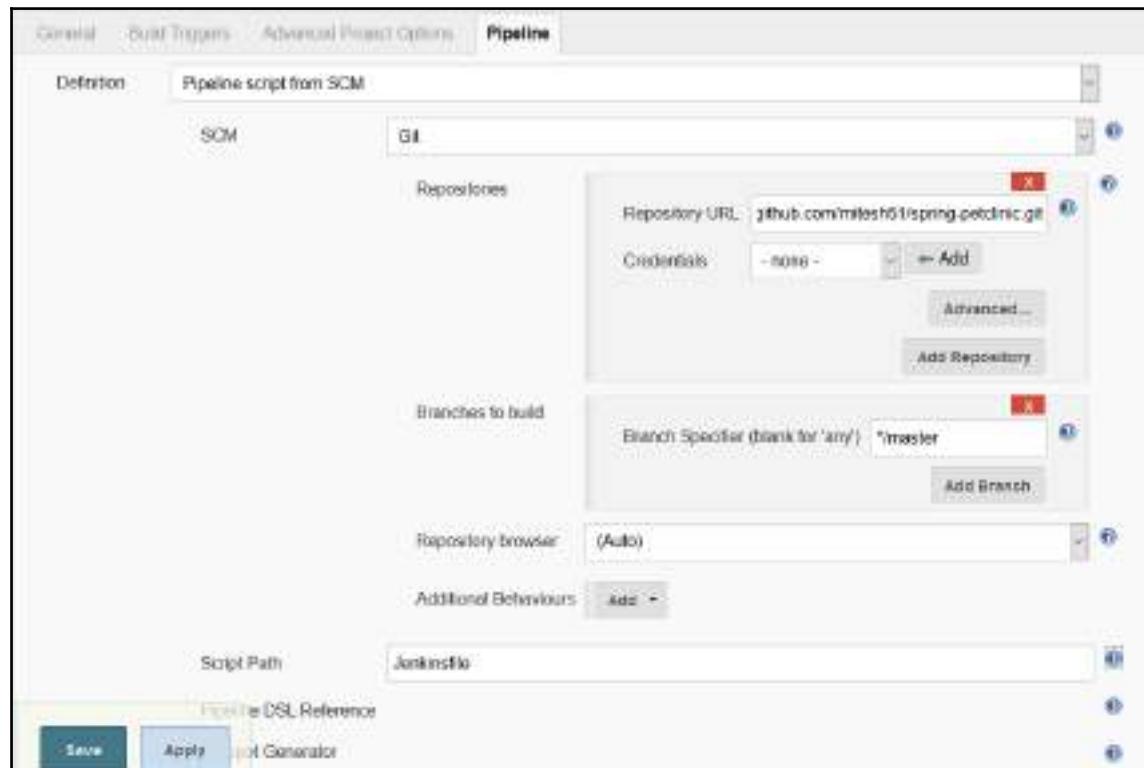
```
> git rev-parse --is-inside-work-tree # timeout=10
Fetching changes from the remote Git repository
> git config remote.origin.url https://github.com/mitesh51/spring-petclinic.git # tim
eout=10
Fetching upstream changes from https://github.com/mitesh51/spring-petclinic.git
> git --version # timeout=10
> git fetch --tags --progress https://github.com/mitesh51/spring-petclinic.git +refs/
heads/*:refs/remotes/origin/*
> git rev-parse refs/remotes/origin/master^{commit} # timeout=10
> git rev-parse refs/remotes/origin/origin/master^{commit} # timeout=10
Checking out Revision 44b591f537aae6ebbef0598beab886d38ba8214c (refs/remotes/origin/ma
ster)
> git config core.sparsecheckout # timeout=10
> git checkout -f 44b591f537aae6ebbef0598beab886d38ba8214c # timeout=10
> git branch -a -v --no-abbrev # timeout=10
> git branch -D master # timeout=10
```

Below the log, there are two checkboxes:

- Use a tool from a predefined Tool Installation
- Shell Script

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

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



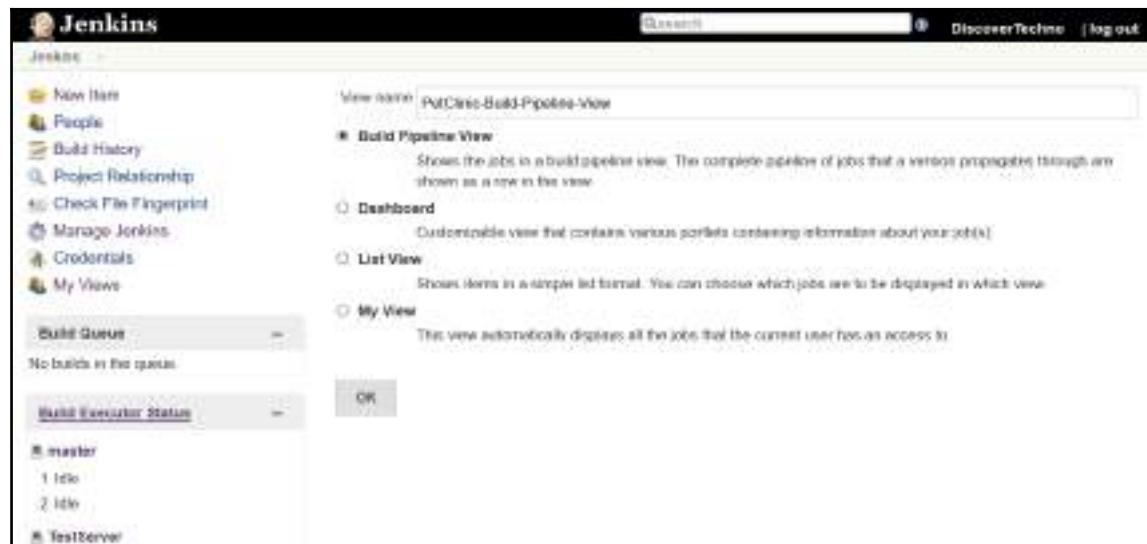
For more details visit *Getting Started with Pipeline* from the Jenkins documentation: <https://jenkins.io/doc/pipeline/>

# Using the Build Pipeline plugin

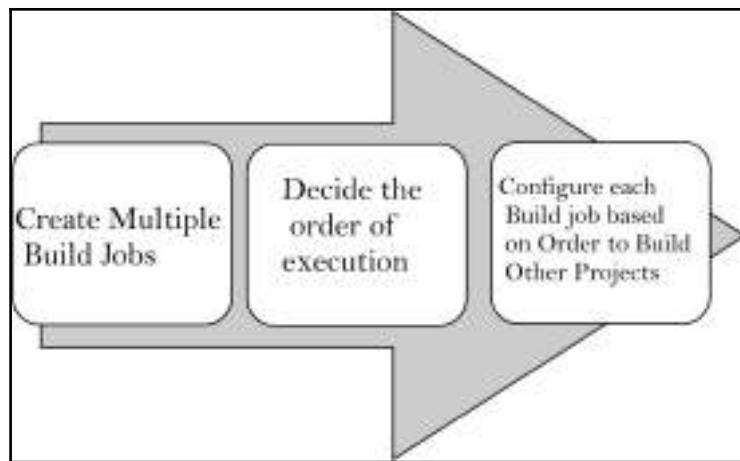
We have seen the built-in pipeline concept of Jenkins 2. It is a very flexible and powerful concept, but for that, we need to write a Groovy script. Another way that has an easy learning curve is to use the **Build Pipeline** plugin. It provides simple visualization of upstream and downstream build jobs. It also enables manual triggers for a situation where we need approval for executing a specific build. We can create a chain of jobs for end-to-end automation. Here, I'm assuming that you are aware of the concept of upstream and downstream build jobs.

To create a build pipeline, follow these steps:

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



3. It is important to configure upstream and downstream build jobs:



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

We have defined the order as well: if compilation is successful, the other two build jobs will be executed. In our case, they are PetClinic-Code and PetClinic-Test.

Follow these steps to configure the build jobs:

1. Go to the configuration page of the PetClinic-Compile build job.
2. Go to the **Post-build Actions** section.
3. Enter the name of the build jobs in the **Project to build** textbox. You can provide a comma-separated list here.

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

The screenshot shows the 'Post-build Actions' configuration screen for a Jenkins job. It includes sections for 'Build other projects' (with 'Projects to build' set to 'PetClinic-Code, PetClinic-Test' and 'Trigger only if build is stable' selected), 'E-mail Notification' (with 'Recipients' set to 'mitesh.soni@outlook.com' and two checkboxes for sending emails on unstable builds and broken builds), and an 'Add post-build action...' dropdown menu. At the bottom are 'Save' and 'Apply' buttons.

5. Verify the list of the **Downstream Projects** on the build job's main page:

The screenshot shows the main page for the Jenkins project 'PetClinic-Compile'. It features a sidebar with links like 'Back to Dashboard', 'Status', 'Changes', 'Workspace', 'Build Now', 'Delete Project', 'Configure', 'Github Hook Log', 'More', and 'Github'. The main content area is titled 'Project PetClinic-Compile' and includes sections for 'Workspace' (with a link to 'Recent Changes'), 'Downstream Projects' (listing 'PetClinic-Code' and 'PetClinic-Test'), and 'Permalinks' (listing several build history items). A 'Disable Project' button is visible in the top right corner.

6. The next step is to configure the **Build Pipeline View** we created earlier. Use this table:

Property name	Property description
<b>Name</b>	The name of the build pipeline.
<b>Description</b>	The description is displayed on the <b>Build Pipeline View</b> page. It can be used to display details such as the pipeline, resources, the objective of the pipeline, and the flow.
<b>Filter build queue</b>	Only jobs in this specific view will be shown in the queue.
<b>Filter build executors</b>	This is used to show build executors that could execute the jobs in this view.
<b>Build Pipeline View Title</b>	The build pipeline view title to display on the Jenkins dashboard.
<b>Layout</b>	<b>Based on the upstream/downstream relationship:</b> This layout mode derives the pipeline structure based on the upstream/downstream trigger relationship between jobs.
<b>Select Initial Job</b>	This sets the initial or parent job in the build pipeline view. The rest of the build job will be considered based on the upstream/downstream relationship.
<b>No Of Displayed Builds</b>	The number of build pipelines to display in the view.
<b>Restrict triggers to most recent successful builds</b>	This is used to restrict the display of a <b>Trigger</b> button to only the most recent successful build pipelines.
<b>Always allow manual trigger on pipeline steps</b>	This is used to execute a successful pipeline step again, using the same parameter values if the build is parameterized.
<b>Show pipeline project headers</b>	This is used to show the pipeline definition header in the pipeline view.
<b>Show pipeline parameters in project headers</b>	This is used to list the parameters used to run the latest successful job in the pipeline's project headers.
<b>Show pipeline parameters in revision box</b>	This is used to list the parameters used to run the first job in each pipeline's revision box.
<b>Refresh frequency (in seconds)</b>	This provides the frequency in seconds with which the <b>Build Pipeline Plugin</b> updates the build lightbox.

<b>URL for custom CSS files</b>	Used for a custom CSS files if any.
<b>Console Output Link Style</b>	You can choose from <b>Lightbox</b> , <b>New Window</b> , or <b>This Window</b> .

7. We have selected the **PetClinic-Compile** build job in the **Select Initial Job** section as the initial job, as you can see here:

The screenshot shows the Jenkins Pipeline View configuration interface. Key settings visible include:

- Name:** PetClinic-Build-Pipeline-View
- Description:** (empty)
- Plan (x) Preview:** (disabled)
- Filter build queue:** (disabled)
- Filter build executors:** (disabled)
- Build Pipeline View Title:** (empty)
- Layout:** Based on upstream/downstream relationship
- Select Initial Job:** PetClinic-Compile
- No Of Displayed Builds:** 1
- Restrict triggers to most recent successful builds:**  Yes  No
- Always allow manual trigger on pipeline steps:**  Yes  No
- Show pipeline project headers:**  Yes  No
- Show pipeline parameters in project headers:**  Yes  No
- Show pipeline parameters in tooltip box:**  Yes  No
- Refresh frequency (in seconds):** 3
- URL for custom CSS files:** (empty)
- Console Output Link Style:** Lightbox

At the bottom are two buttons: **OK** and **Apply**.

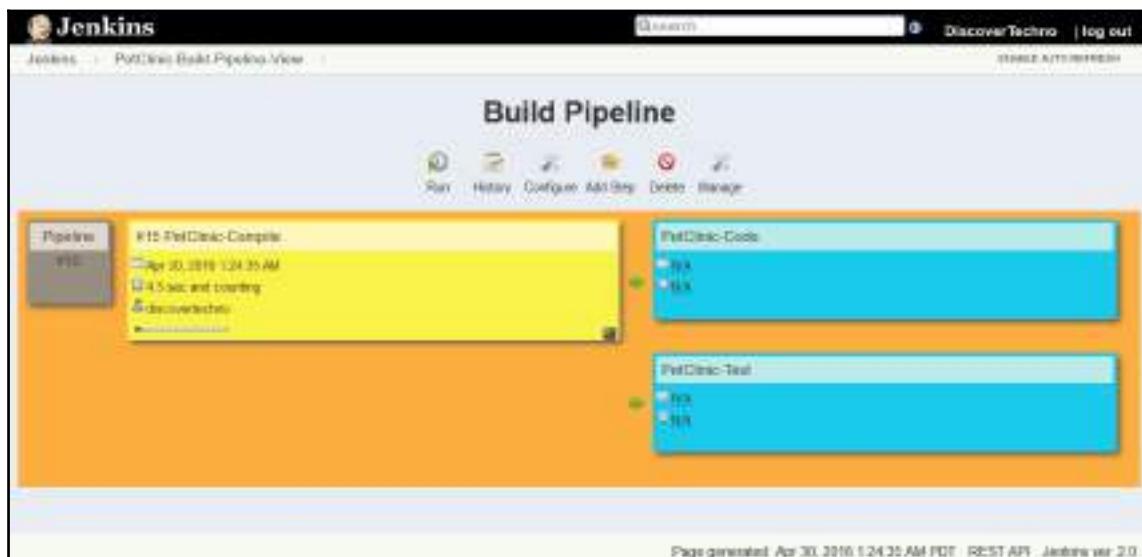
8. On the **PetClinic-Build-Pipeline-View** page, we can run the build pipeline by clicking on **Run**, view the history by clicking on **History**, configure the pipeline by clicking on **Configure**, and delete the pipeline using **Delete**. Click on **Run** to execute the build pipeline for the first time:



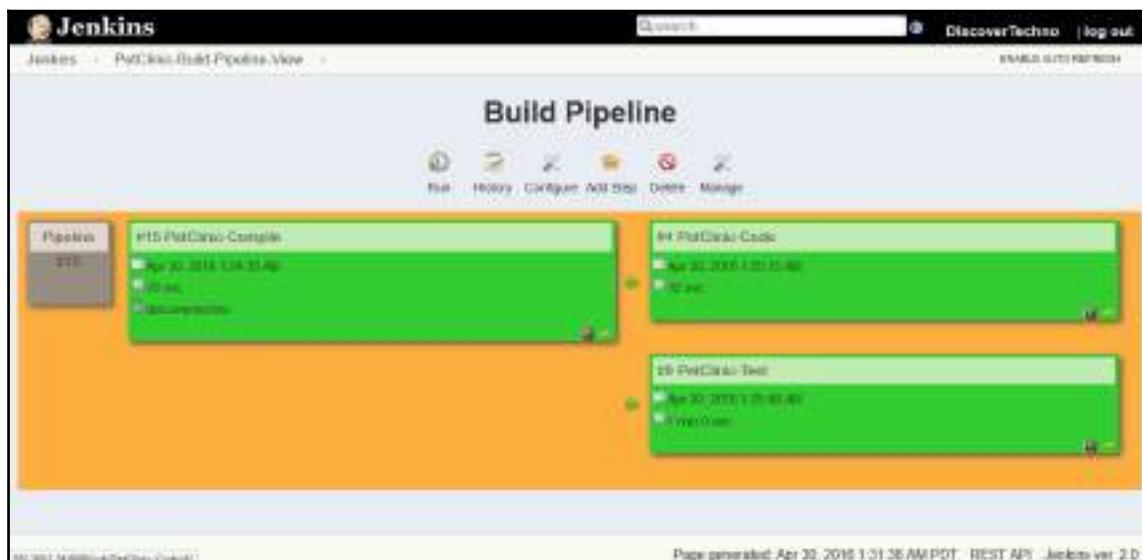
9. The following are the default color codes:

Color	Description
Red	Indicates failed execution of a build job
Green	Indicates successful execution of a build job
Blue	Indicates a build job that hasn't been executed
Yellow	Indicates a running build job

10. Now let's observe the execution of a build job in this pipeline, as shown in the following screenshot:



11. We can see all jobs in green as all the builds have been executed successfully:

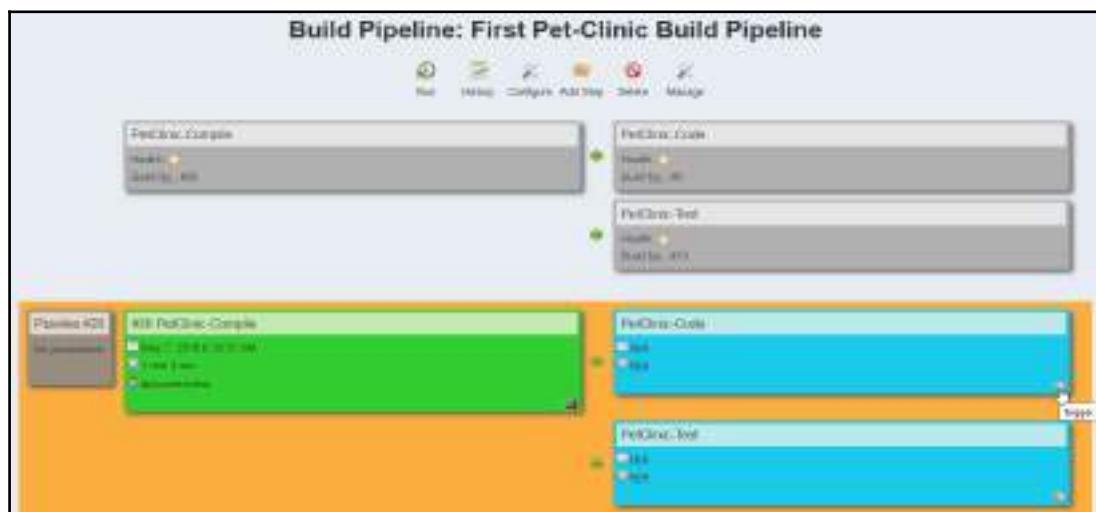


Manual trigger configuration makes sure that pipeline step doesn't execute automatically and requires manual intervention to execute next step. It can be very useful in the scenarios where you need to wait for the deployment in the specific environment or to wait for permission before deployment. Let's configure the build pipeline using a manual trigger:

- As shown in the following screenshot, select Yes for all the options:



- Let's save and verify the changes in the **Build Pipeline View** section. Verify the manual trigger and headers with the health details of each build job:



3. Verify the build history of **PetClinic-Build-Pipeline-View**, as shown in the following screenshot:

The screenshot shows the Jenkins interface for the 'PetClinic-Build-Pipeline-View'. At the top, there's a header with a calendar icon and the title 'Build History of PetClinic-Build-Pipeline-View'. Below the header is a timeline showing dates from May 5 to May 11. Under each date, there are four small icons representing different stages of the pipeline. Below the timeline is a table titled 'Export as plain XML' with columns for 'Build', 'Time Since', and 'Status'. The table contains four rows of build information:

Build	Time Since	Status
PetClinic-Code #10	1 day 4 hr	stable
PetClinic-Code #9	1 day 21 hr	stable
PetClinic-Test #13	1 day 21 hr	stable
PetClinic-Compile #19	1 day 21 hr	stable



the Build Pipeline plugin from <https://wiki.jenkins-ci.org/display/JENKINS/Build+Pipeline+Plugin>.

Deploying a WAR file. The most important thing in application life cycle management is the deployment of the packages. That is the business part of the whole exercise. The objective is to automate the process of deployment of the packages into web server or application server. Once the deployment process is automated, it can be easily integrated into end to end automation of application delivery.

For Maven and Tomcat integration, let's create an admin user. We will use admin user credentials to deploy an application to a Tomcat server:

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

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

2. Create a user with the name `admin`, and assign a password and roles, as shown here:

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

3. Now, we need to add this Tomcat admin user to the `Maven settings.xml` file:

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

4. Now let's edit the `pom.xml` file. Find the `Tomcat Plugin` block in `pom.xml`, and add following details. Make sure that the server name is the same one we provided in the `settings.xml` file 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>
```

5. We can verify execution from the command-line using `mvn tomcat7:deploy` command. Maven will deploy the WAR file to Tomcat 7 using the Manager app at `http://localhost:8080/manager/text`, to the `/petclinic` path.

6. In the case of any failures because of a preexisting WAR file in the 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 for **Goals** and Save it:



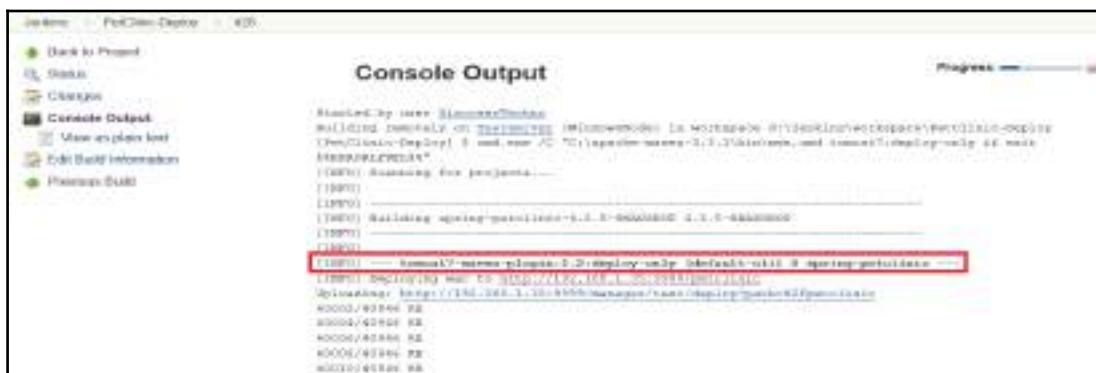
2. Execute the build by clicking on **Build Now**. Verify the deployment process in the console output:

```
[INFO] --- maven-war-plugin:2.3:war (default-war) @ spring-petclinic ---
[INFO] Packaging warapp
[INFO] Assembling webapp [spring-petclinic] in d:\jenkins\workspace\PetClinic-Deploy\target\spring-petclinic-4.2.5-SNAPSHOT
[INFO] Processing war project
[INFO] Copying war resources d:\jenkins\workspace\PetClinic-Deploy\src\main\webapp
[INFO] Webapp assembled in [969] needs
[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
```

- Once the WAR file has been uploaded, the build job will be completed successfully:

```
40940/40946 KB  
40942/40946 KB  
40944/40946 KB  
40946/40946 KB  
uploaded: http://192.168.1.15:8080/manager/text/deploy?path=%2fpetclinic&update=true (40946 KB at  
3024.6 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: 59.440 s  
[INFO] Finished at: 2016-05-07T23:41:13+05:30  
[INFO] Final Memory: 388/1523M  
[INFO] --  
[INFO] finished: success
```

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



# Integrating the deployment operation

Till now we have covered Pipeline or orchestration of different tasks and now let's integrate pipeline and deployment automation. By doing this, we will complete Continuous Integration and Continuous Delivery with orchestration. Let's try to integrate the deployment operation into the build pipeline.

We will need to perform the following tasks:

1. Compile source files.
2. Execute JUnit test cases.
3. Archive the artifact/WAR file: It is used to archive build artifacts, such as JAR files, WAR files, and ZIP files, so they can be downloaded later. Add **Post-build Actions** to PetClinic-Test to archive the artifact:



4. Execute the build job, as shown in the following screenshot, and verify whether it has been successfully archived or not. If you see **Finished: SUCCESS**, then the build job was successfully executed:

```
[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
```

5. We need to add a build step to copy artifacts from **PetClinic-Test**. Check the **Copy Artifact Plugin** checkbox, and then click on **Install without restart**:

The screenshot shows the Jenkins Plugins page with the 'Available' tab selected. There are two listed plugins:

- Artifact Deployer Plugin**: Version 0.33. Description: This plugin makes it possible to copy artifacts to remote locations.
- Copy Artifact Plugin**: Version 1.38. Description: Adds a build step to copy artifacts from another project.

At the bottom of the list are three buttons: 'Install without restart' (highlighted in blue), 'Download now and install after restart', and 'Check now'.

6. Configure the copy artifact plugin in the **PetClinic-Deploy** build job:

The screenshot shows the Jenkins Build configuration page for the 'PetClinic-Deploy' job. The 'Build' tab is selected. Under the 'Copy artifacts from another project' section, the configuration is as follows:

- Project name: PetClinic-Test
- Which build: Latest successful build
- Artifacts to copy: target/\*.war
- Artifacts not to copy: (empty)
- Target directory: (empty)
- Parameter filters: (empty)
- Advanced options: Flatten directories, Optional, Fingerprint Artifacts

7. Verify the workspace directory. Go to the **PetClinic-Test** target directory. If a WAR file is there from a past build, remove it:

New Volume (D:) > jenkins > workspace > PetClinic-Test > target		
Name	Date modified	Type
classes	4/28/2016 8:43 AM	File folder
generated-sources	4/28/2016 8:42 AM	File folder
generated-test-sources	4/28/2016 8:43 AM	File folder
maven-archiver	5/8/2016 12:27 AM	File folder
maven-status	4/28/2016 8:42 AM	File folder
spring-petclinic-4.2.5-SNAPSHOT	5/8/2016 12:27 AM	File folder
surefire-reports	4/28/2016 8:43 AM	File folder
test-classes	4/28/2016 8:43 AM	File folder

8. Verify the target directory of the **PetClinic-Deploy** folder. There is no WAR file:

New Volume (D:) > jenkins > workspace > PetClinic-Deploy > target		
Name	Date modified	Type
classes	5/7/2016 10:33 PM	File folder
generated-sources	5/7/2016 10:33 PM	File folder
generated-test-sources	5/7/2016 10:33 PM	File folder
maven-archiver	5/7/2016 10:46 PM	File folder
maven-status	5/7/2016 10:33 PM	File folder
spring-petclinic-4.2.5-SNAPSHOT	5/7/2016 10:46 PM	File folder
surefire-reports	5/7/2016 10:34 PM	File folder
test-classes	5/7/2016 10:33 PM	File folder

9. Add **PetClinic-Deploy** as a downstream project to **PetClinic-Test**. Then, run the build pipeline:



10. Verify the execution of the build pipeline. Click on the lightbox of any build job in the build pipeline. Verify the **PetClinic-Test** console output:

The screenshot shows the PetClinic-Test build console output. It displays a series of Maven build logs:

```
Tests run: 0, 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 war file contents [d:\jenkins\workspace\PetClinic-Test\target\spring-petclinic-4.2.5-SNAPSHOT]
[INFO] Webapp assembled in [2371] msec
[INFO] Building war: d:\jenkins\workspace\PetClinic-Test\target\spring-petclinic-4.2.5-SNAPSHOT.war
[INFO]
[INFO] success
[INFO]
[INFO] Total time: 38.605 s
[INFO] Finished at: 2016-05-09T00:53:30+05:30
[INFO] Final Memory: 250/211M
[INFO]
[INFO] Annotating artifacts
Recording test results
Warning: you have no plugins providing coverage control for builds; falling back to legacy behavior of permitting any downstream builds to be triggered
Triggering a new build of PetClinic-Deploy
finished: success
```

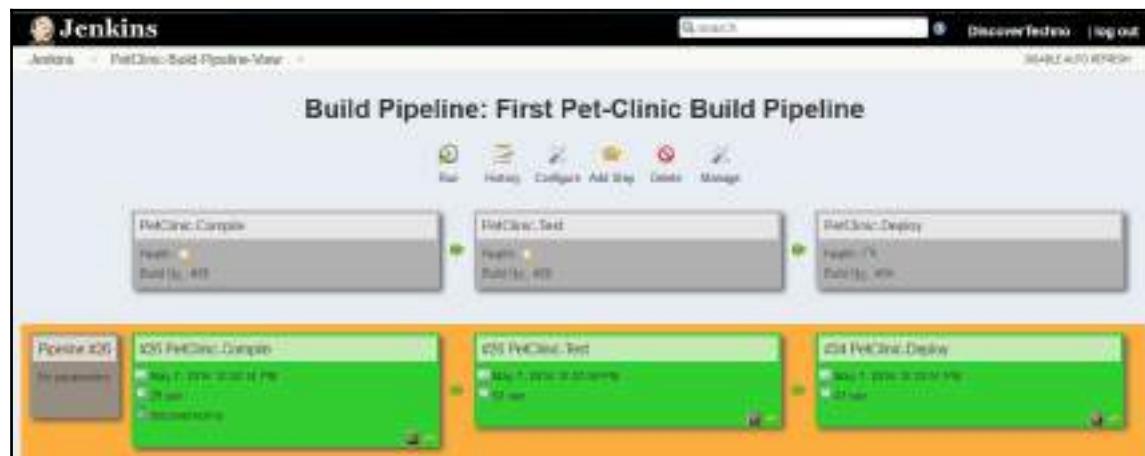
At the bottom, a green bar indicates the build was successful: "Page generated: May 7, 2016 12:24:56 PM PDT BEST\_AWS instance-20".

Now, we will copy the archive file from one build and use it for deployment in another build. Once the **PetClinic-Test** build job execution has completed, follow these steps:

1. Verify the target folder in the workspace. You will see the WAR file in the target directory, as shown in the following screenshot:

New Volume (D:) > jenkins > workspace > PetClinic-Test > target			
Name	Date modified	Type	Size
classes	4/28/2016 8:43 AM	File folder	
generated-sources	4/28/2016 8:42 AM	File folder	
generated-test-sources	4/28/2016 8:43 AM	File folder	
maven-archiver	5/8/2016 12:27 AM	File folder	
maven-status	4/28/2016 8:42 AM	File folder	
spring-petclinic-4.2.5-SNAPSHOT	5/8/2016 12:27 AM	File folder	
surefire-reports	4/28/2016 8:43 AM	File folder	
test-classes	4/28/2016 8:43 AM	File folder	
spring-petclinic-4.2.5-SNAPSHOT.war	5/8/2016 12:53 AM	WAR File	40,946 KB

2. Verify the execution of the **PetClinic-Deploy** build job:



3. Verify the build job's status in the Jenkins dashboard:

[Back to Project](#)

[Status](#)

[Changes](#)

[Console Output](#)

[View as plain text](#)

[Edit Build Information](#)

[Delete Build](#)

[See Fingerprint](#)

[Previous Build](#)

## Build #34 (May 7, 2016 12:23:51 PM)

Started 1 min 57 sec ago  
took 41 sec on [TestServer](#)

[Add description](#)

No changes. Changes in dependency

+ [PetClinic-Test @#19 → @620 \(detected\)](#)

Started by upstream project [PetClinic-Test](#) build number 20  
originally caused by

- Started by upstream project [PetClinic-Console](#) build number 18  
originally caused by
  - Started by user [DiscoverTechno](#)

[Promote Build](#)

### Upstream Builds

PetClinic-Test @#620

4. Click on the lightbox of **Build Pipeline View**; it will redirect you to the console output of a specific build job. Click on the **PetClinic-Deploy** lightbox.
  5. Verify the console output:

**Console Output**

```
started by upstream project "pcclinic-test" build number 20
originally caused by:
Started by upstream project "pcclinic-compile" build number 18
originally caused by:
    Started by user DiscoverTechno
Building remotely on Tenderloin (Windows10) in workspace /opt/jenkins/workspace/pcclinic-deploy
Copied 1 artifact from "pcclinic-test" build number 20
[pcClinic-Deploy] F.xml.war /D:/apache-tomcat-3.0.16/bin/www.war tomcat7:deploy-only 44-unit
[INFO]SCHEMELINKS
[INFO] missing fac:projects...
[INFO]
[INFO] -----
[INFO] building spring-pcclinic-4.2.5-warpack 4.2.5-snapshot
[INFO] -----
[INFO]
[INFO] -----
[INFO] --- tomcat7-maven-plugin:2.2:deploy-only (default-cli) @ spring-pcclinic ---
[INFO] Deploying war to http://192.168.1.10:8080/tomcat7
optimizing http://192.168.1.10:8080/tomcat7/...
2/40048: EB
4/40048: EB
6/40048: EB
8/40048: EB
```

6. Verify that the successfully uploaded file adheres to the configuration:

As an exercise, try to use the build flow plugin.



# Self-test questions

1. Which feature is one of the highlights of the Jenkins 2 release?
    - Built-in support for continuous integration
    - Built-in support for JUnit
    - Built-in support for delivery pipelines
    - Built-in support for Apache Maven

2. Which language is used to create delivery pipelines ?

- Java
- C++
- C#
- Domain-specific language

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

- Indicating failed execution of a build job
- Indicating successful execution of a build job
- Indicating a build job that hasn't been executed
- Indicating a running build job

## Summary

In this chapter, we covered one of the latest features of Jenkins 2 and one of its highlights: built-in support for delivery pipelines. We learned how to use it in detail. We covered a simple Groovy script to build a job, generate a build step, archive build job artifacts, run a build step on a specific node, mark definite sections of a build as being controlled by limited concurrency, and so on. We walked through a scenario where we want to execute different stages on different nodes. Another similar plugin was installed and configured with an example: the Build Pipeline plugin.

In the next chapter, we will discuss one of the important pillars of DevOps culture—configuration management—using Chef. First, we will see how to install Chef on a workstation and configure it with hosted Chef. We will look at installing Tomcat using community Tomcat installation cookbooks.

# 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 the application delivery lifecycle. Let's revisit the context. We want to create an end-to-end pipeline where the application source files are compiled, unit tests are executed, package file is created, a new virtual machine created, runtime environment is setup, and deployment is performed. Chef in our context plays a vital role, considering its many uses. We are going to use it for setting up our runtime environment and standardizing the process of configuration management rather than implementing a customized way to install tools using scripts. Centralized configuration management makes it easy to control and configure resources without complication.

This chapter describes in detail the configuration management tool Chef, the installation of its components and alternatives, and the configuration of components and convergence of a node for preparing a runtime environment for Java EE application using cookbooks. However, writing a cookbook and a detailed discussion of Chef components is out of the scope of this book as it will take up too much space.

You will learn how to install and configure Chef and converge a node based on cookbooks/roles.

We will cover the following topics:

- Getting started with Chef
- An overview of hosted Chef
- Installing and configuring a Chef workstation
- Converging a Chef node using a Chef workstation

## Getting started with Chef

The Chef is one of the most popular configuration tools in the open source world. We discussed Chef briefly in [Chapter 1, Getting Started-DevOps Concepts, Tools, and Technology](#).

Let's get hands-on with provisioning instances and configuration management. However, before that, we will need to understand the basics.

There are three major components of Chef:

- **The open source Chef server or hosted Chef:** The Chef server or hosted Chef is the pivotal component, which stores cookbooks and other important details of registered nodes. It is used to configure and manage nodes using Chef workstations.
- **Chef workstations:** A Chef workstation works as a local repository, and the `knife` plugin is installed on it. Knife is used to upload cookbooks to the Chef server and execute plugin commands.
- **Node:** A node is a physical or virtual machine in any environment where we need to configure runtime environments or perform operations using Chef configuration. The node communicates with the Chef server (open source or hosted), obtains configuration details related to itself, and then starts executing steps based on it. The Chef server can be installed on a physical machine or a virtual machine with an open source installable file, based on the operating system. Another, easier method to use is hosted Chef, where we need not install and configure a Chef server. We can use the SaaS offering from Chef. It allows up to five nodes. The biggest benefit is we need not manage a Chef server or upgrade it. Hence, we save ourselves from management and maintenance overhead.

Take a look at the Chef website <https://chef.io>. You will see the Chef homepage, as shown here:

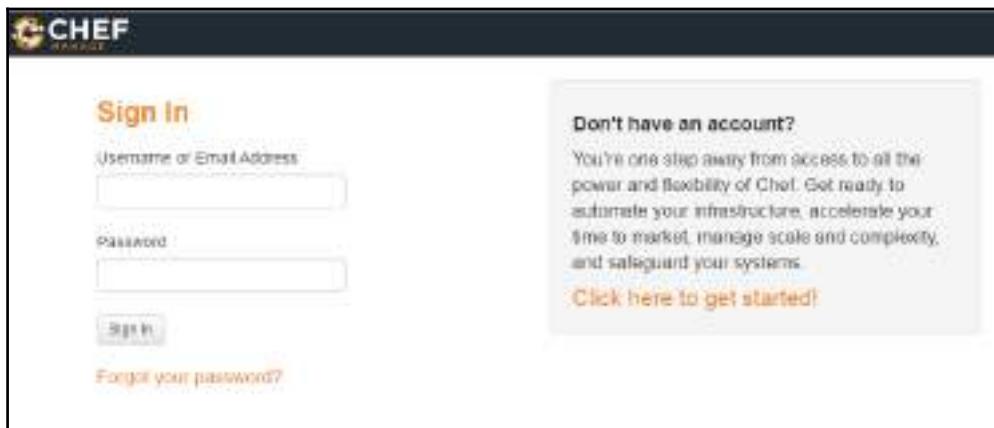


There are a lot of details available here about Chef and cloud-related integration as well as knife plugins. We will create a hosted Chef account in the next section and configure it with a local workstation. To proceed, click on the **MANAGEMENT CONSOLE** link in the top right corner of the Chef website.

## Overview of hosted Chef

We can use Chef server either by installing and managing Chef server on our own or we can use hosted Chef – SaaS offering to utilize in configuration management.

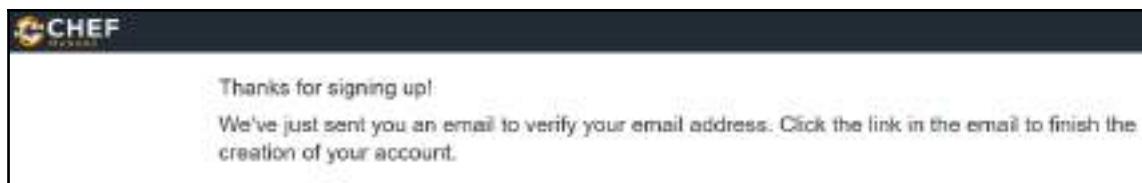
1. Click on **MANAGEMENT CONSOLE** or navigate to <https://manage.chef.io/> login. We are going to start from scratch, so click on **Click here to get started!**



2. Enter your **Full Name**, **Company** name, **Email** address, and **Username** in the respective text boxes and check the box that says **I agree to the Terms of Service and the Master License and Services Agreement**. Then, click on the **Get Started** button:

The screenshot shows the 'Start your free trial of Hosted Chef' page. It has fields for 'Full Name' (DiscoverTechno), 'Company' (DiscoverTechno), 'Email' (redacted@outlook.com), and 'Username' (discovertechno61). A checkbox at the bottom left states 'I agree to the [Terms of Service](#) and the [Master License and Services Agreement](#)'. On the right side, there are links for 'Already have an account?' (with a 'Click here to sign in' link) and 'Looking for open-source Chef?'. It also promotes the 'Chef client and server installation' and the 'Chef Community'.

3. You will then see this message:



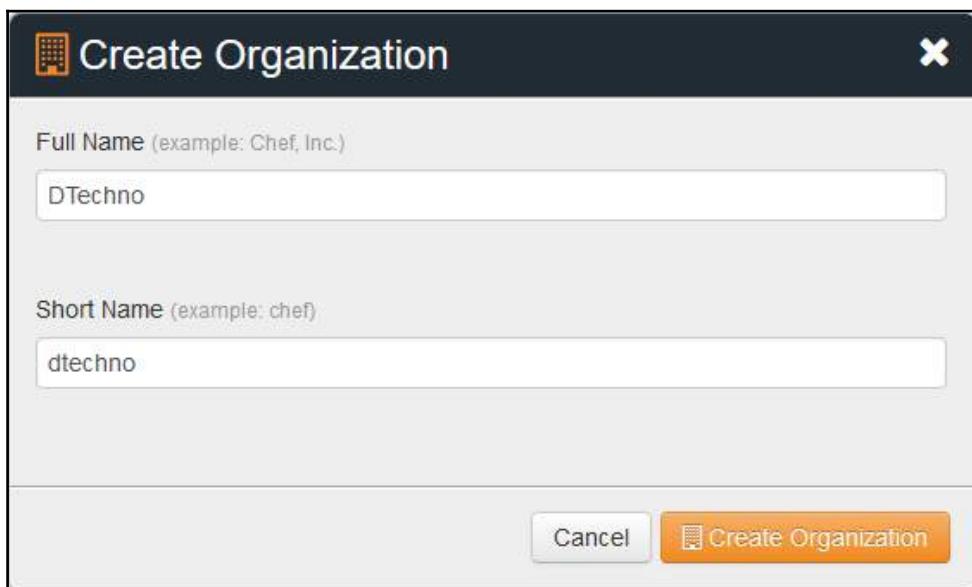
4. Open your e-mail inbox and click on the verification link to complete the creation of your hosted Chef account. You will get an **Email Verification Successful** message. After typing your password, click on **Create User** button:

The page title is "Email Verification Successful". The text says: "Thank you for verifying your email address! Please enter the password you'd like to use below and submit the form to complete the creation of your account." Below this is a password input field with masked text and a "Create User" button.

5. The next task is to create an organization. Click on **Create New Organization**:



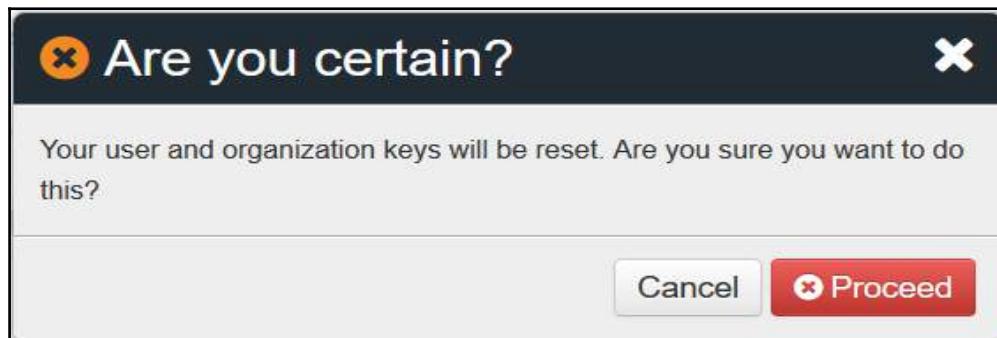
6. Provide the **Full Name** and **Short Name** of the organization, and click on the **Create Organization** button:



7. Bingo! You've just created your hosted Chef account, and you can now start using it. The next step is to download a starter kit:

The screenshot shows the Hosted Chef dashboard. The top navigation bar includes "Nodes", "Reports", "Policy", and "Administration". The left sidebar has sections for "Organizations" (Create, Plan Validator, Key, Generate Recipe, Config, Invite User, Learn, Organisation, Starter Kit), "Users", "Groups", and "Global Permissions". The main content area features a "Thank you for choosing Hosted Chef!" message, a "Follow these steps to be on your way to using Hosted Chef" section with "Download Starter Kit" and "Learn Chef" buttons, and a "What's next?" section with links to "Chef Documentation" (described as "The best place to start learning about Chef in general"), "Browse Community Cookbooks" (described as "Hundreds of members of the Chef community have contributed cookbooks you can use or draw inspiration from"), "Contact Support" (described as "Our support team is here to provide assistance with any issues you may have while using Hosted Chef. Learn more about support for Chef."), and "More Resources" (links to "Chef Mailing List", "Join the Mailing List", "Watch the Free Chef Flight show", and "Get Professional Training").

- When you click on **Download Starter Kit**, your user and organization keys will be reset. Make sure to keep them in a safe place. On the confirmation dialog, click on **Proceed**:



Let's have a quick walkthrough of the hosted Chef portal or dashboard

- Click on **Nodes**; you will be shown an empty list as no node has been configured using the Chef server. Note this as we are going to see the same screen when we configure a node later.

A screenshot of the Chef Nodes dashboard. The top navigation bar includes "CHEF", "Nodes", "Reports", "Policy", and "Administration". The "Nodes" tab is selected. The main content area is titled "Showing All Nodes" and displays the message "There are no items to display". A large, light-gray callout at the bottom center says "Please select a node". On the left side, there is a sidebar with links: "Create", "Manage Keys", "Reset Key", "Edit Node List", and "Edit Preferences".

2. Now, navigate to **Administration | Users** and verify the user account created at the time of registration:

The screenshot shows the Chef Administration interface with the 'Administration' tab selected. On the left, there's a sidebar with 'Organizations' and 'Users' sections. Under 'Users', it lists 'Invite', 'Change Password', 'Reset Password', 'Delete User', 'Previous User', and 'Permissions'. The main panel is titled 'Showing All Users' and displays a single user entry: 'User Name: duncanmcclure', 'Full Name: Duncan McClure', and 'Email: duncan.mcclure@redhat.com'. Below this, a large message says 'Please select a user'.

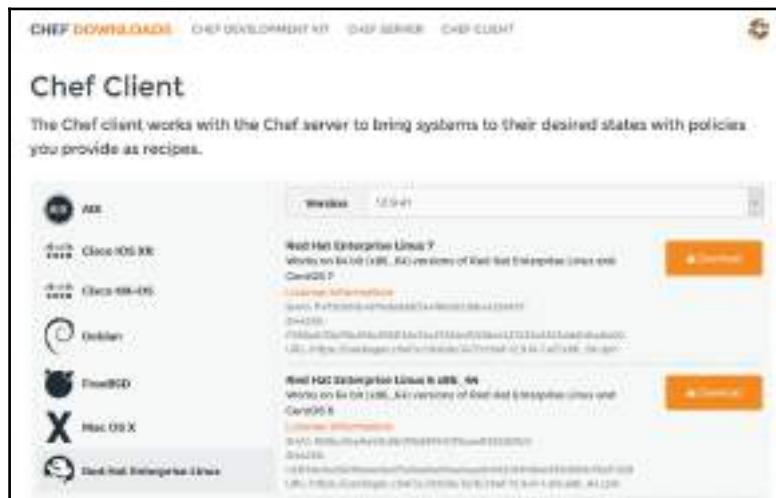
3. The **Reports** tab has no data as the convergence process hasn't taken place and no success or failure data is available:

The screenshot shows the Chef Administration interface with the 'Reports' tab selected. On the left, there's a sidebar with 'Dashboard' and 'Run History' sections. The main panel is titled 'Showing chef-client runs for 05/16/2016 - 05/12/2016'. It contains three sections: 'Run Summary' (which says 'No Data Available.'), 'Run Details' (which shows a table with columns 'Run ID', 'Run Date/Time', 'Run Type', and 'Status' with a single row '05-16-2016 10:45 AM Success'), and 'Run Counts' (which also says 'No Data Available.').

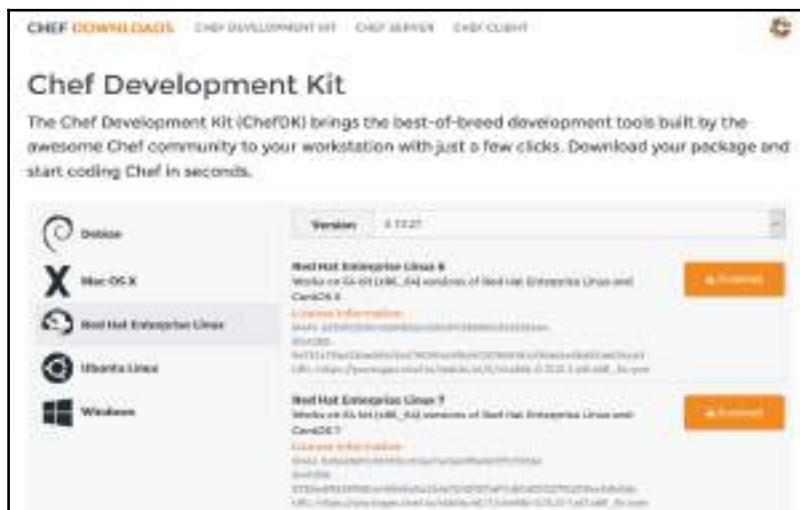
Once we have a hosted Chef account available, the next step is to configure a Chef workstation:

1. First, download the Red Hat version of the Chef client from <https://downloads.chef.io/chef-client/redhat/> as we are going to use a CentOS virtual machine to act as our workstation.

2. Select your operating system type, select the Chef client version, and download the installation files:



3. The Chef development kit is useful for installing development tools, and it can also be used to install knife plugins for AWS and Azure. Download it from <https://downloads.chef.io/chef-dk/>:



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

# Installing and configuring a Chef workstation

Before installing a Chef client for preparing a workstation, let's try to verify whether the Chef client has been installed:

1. Execute the `chef-client -version` command to verify whether the Chef client has been installed:

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

2. As you can see in the output of the previous command, the Chef client is not installed. Now, navigate to the directory where the Chef client installable is stored using the `cd` command:

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

3. Run the downloaded Chef client RPM using `rpm -ivh chef-<version>.rpm`:

```
[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 V4DSA/SHA1  
Signature, key ID 83ef826a: NOKEY  
error: can't create transaction lock on /var/lib/rpm/.rpm.lock  
(Permission denied)
```

4. Permission is denied, so 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 V4DSA/SHA1  
Signature, key ID 83ef826a: NOKEY  
Preparing...#####
1:chef ##### [100%]  
Thank you for installing Chef!
```

5. After successful installation, verify the Chef client version:

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

The next step is to use the Chef starter kit that we downloaded while creating an account in hosted Chef:

1. Extract the `chef-repo` compressed file, and verify its contents. Copy the `.chef` directory into the root or user folder:



2. Verify the `cookbooks` folder, available in the `chef-repo` directory:



3. In the `.chef` directory, open the `knife.rb` file, which contains various configurations. All the configurations you need are already available. Adjust the path of the `cookbooks` directory if needed:

```
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"]
```



For more information on knife's configuration options, visit [http://docs.chef.io/config\\_rb\\_knife.html](http://docs.chef.io/config_rb_knife.html).

4. With that, we've finished configuring our Chef workstation. The next step is using it to converge the node.

# Converging a Chef node using a Chef workstation

In this section, we will try to setup runtime environment in node using Chef workstation.

First of all, let's login to the Chef workstation which setup:

1. Open the terminal and verify the IP address using ifconfig:

```
[root@devops1 chef-repo]#ifconfig
eth3      Link encap:Ethernet HWaddr 00:0C:29:D9:30:7F
inetaddr:192.168.1.35 Bcast:192.168.1.255 Mask:255.255.255.0
inet6addr: 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
inetaddr:127.0.0.1 Mask:255.0.0.0
inet6addr: ::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:

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

3. The knife node list command is used to obtain the list of nodes served by the Chef server in our case, hosted Chef. As we haven't converged any nodes, the list will be empty.

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

4. Create a virtual machine using VMware Workstation or VirtualBox. Install CentOS. Once the VM is ready, find its IP address and note it down.

5. On your Chef workstation, open a terminal and, using ssh, try to connect to the node or VM we just created:

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

6. The authenticity of the host 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. We now have an SSH session on the node from the Chef workstation. If you verify the IP address, you'll know that you are accessing a different machine by remote (SSH)access:

```
[root@localhost ~]#ifconfig  
eth1      Link encap:Ethernet HWaddr 00:0C:29:44:9B:4B  
inetaddr:192.168.1.37 Bcast:192.168.1.255 Mask:255.255.255.0  
inet6addr: 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  
inetaddr:127.0.0.1 Mask:255.0.0.0  
inet6addr: ::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 the node virtual machine. In my case, the VM already had the Chef client installed, so executing rpm -qa \*chef\*gave me the following 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`:

```
[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_641/1
  Verifying  : chef-12.3.0-1.el6.x86_641/1
Removed:
  chef.x86_64 0:12.3.0-1.el6
Complete!
You have new mail in /var/spool/mail/root
```

10. We've removed the Chef client; to verify this, execute the following command:

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

11. Let's remove the Tomcat installation as well if it has been 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:
tomcat6x86_64      6.0.24-83.el6_6      @updates      188 k
Removing for dependencies:
tomcat6-admin-webappsx86_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_641/2
  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
```

12. Now, run the `yum remove tomcat6` command to verify whether Tomcat is still installed on the system:

```
[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
```

13. Check whether the Java Development Kit (JDK) has been installed on the node:

```
[root@localhost Desktop]# java -version
java version "1.7.0_75"
OpenJDK Runtime Environment (rhel-2.5.4.0.el6_6-x86_64u75-b13)
OpenJDK 64-Bit Server VM (build 24.75-b04, mixed mode)
```

14. Exit the SSH session of the node's virtual machine. We now have control of the Chef workstation machine, and we will try to converge the node VM we just accessed remotely.
15. Use knife to converge the node. Provide the IP address/DNS name, user, password, and name of the node.
16. 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 sha1859bc9be9a40b8b13fb88744079ceef1832831b0
192.168.1.37
sha256c43f48e5a2de56e4eda473a3ee0a80aa1aaa6c8621d9084e033d8b9cf3efc328
```

```
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 V4DSA/SHA1
Signature, key ID 83ef826a: NOKEY
192.168.1.37 Preparing...
#####
192.168.1.37 1:chef
#####
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
```

17. There was no run list or role associated with the knife command, but the convergence was successful.

18. Let's verify our hosted Chef account. We can see the **Node Name** and **IP Address** in the **Nodes** section of dashboard, so open it and verify the details:

The screenshot shows the Chef dashboard with the 'Nodes' tab selected. The main table header includes columns for Node Name, Platform, FQDN, IP Address, Uptime, Last Check-in, Environment, and Actions. A single node entry is present: 'bmcatserver' (Platform: centos, FQDN: bmcatserver, IP Address: 192.168.1.37, Uptime: 5 hours, Last Check-in: a few seconds ago, Environment: default). Below the table, a large message says 'Please select a node'.

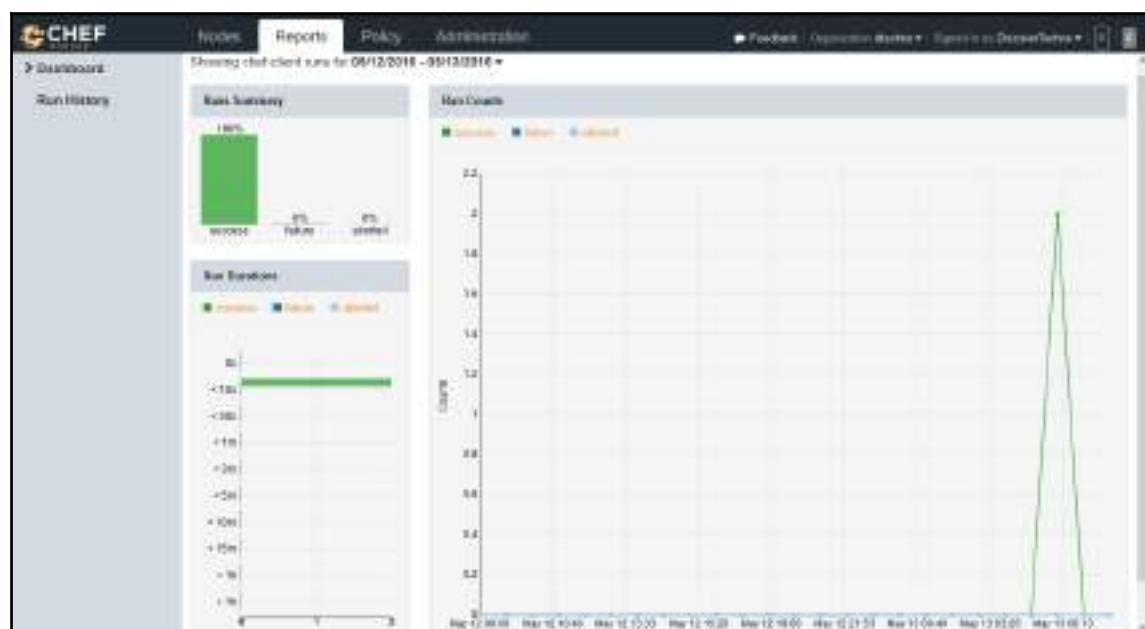
19. Select a node and click on the **Details** tab to get more information about the node, such as the **Attributes** associated with it and its **Permissions**, as shown in the following screenshot:

The screenshot shows the Chef dashboard with the 'Nodes' tab selected. The node 'bmcatserver' is selected in the table, highlighted by an orange row. A detailed modal window titled 'Node: bmcatserver' is open, showing three tabs: Details, Attributes, and Permissions. The 'Details' tab is active, displaying the following information: Last Check-in: An Hour Ago (2016-05-10 00:27:24), Uptime: 8 Hours (since 2016-05-10 00:27:24), Environment: default, Platform: centos, FQDN: bmcatserver, and IP Address: 192.168.1.37.

20. Verify the CPU attributes and other details of the node:

The screenshot shows the Chef web interface with the 'Nodes' tab selected. A search bar at the top right contains the text 'Search Nodes...'. Below the search bar, a table lists nodes: 'tomcatserver' (Platform: centos, FQDN: tomcatserver, IP Address: 192.168.1.17, Uptime: 3 days, Last Check-in: 30 hours ago, Environment: default). The main content area is titled 'Node: tomcatserver' and shows the 'Attributes' tab selected. The attribute tree under 'cpu' is expanded, showing 'model\_name: Intel(R) Core(TM) i5-3230M CPU @ 2.60GHz'.

21. The convergence was successful, and we can see that in the **Reports** section of the hosted Chef account:

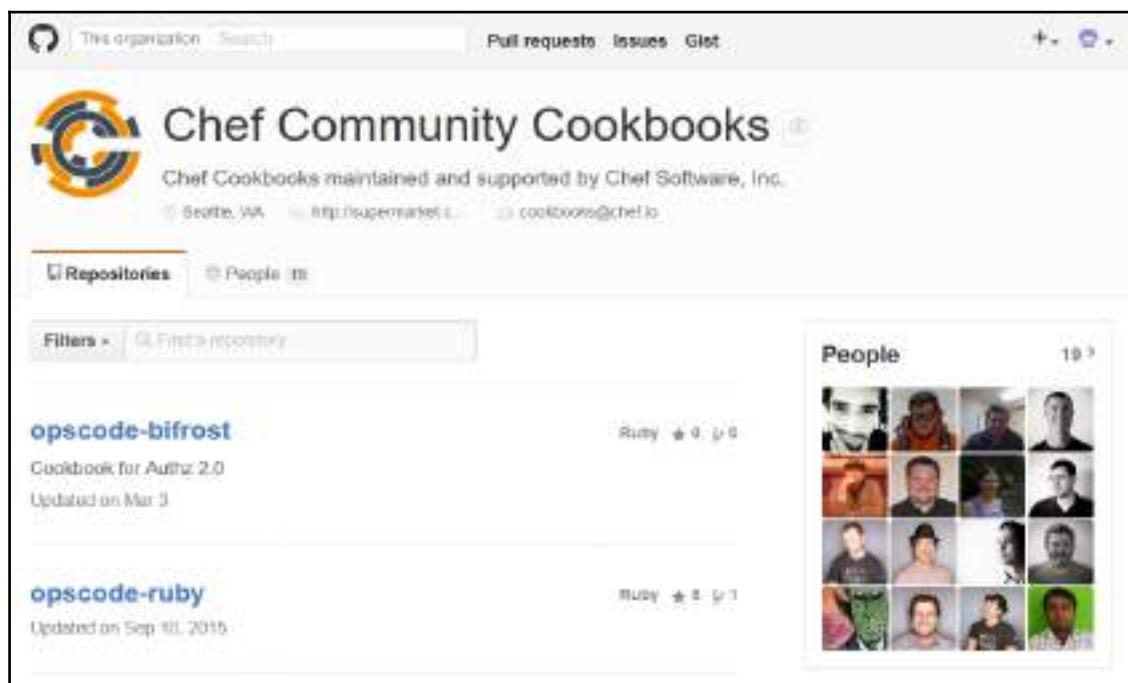


# Installing software packages using cookbooks

Until now, we've seen how to create a hosted Chef account, how to configure a Chef workstation, and how to converge a node.

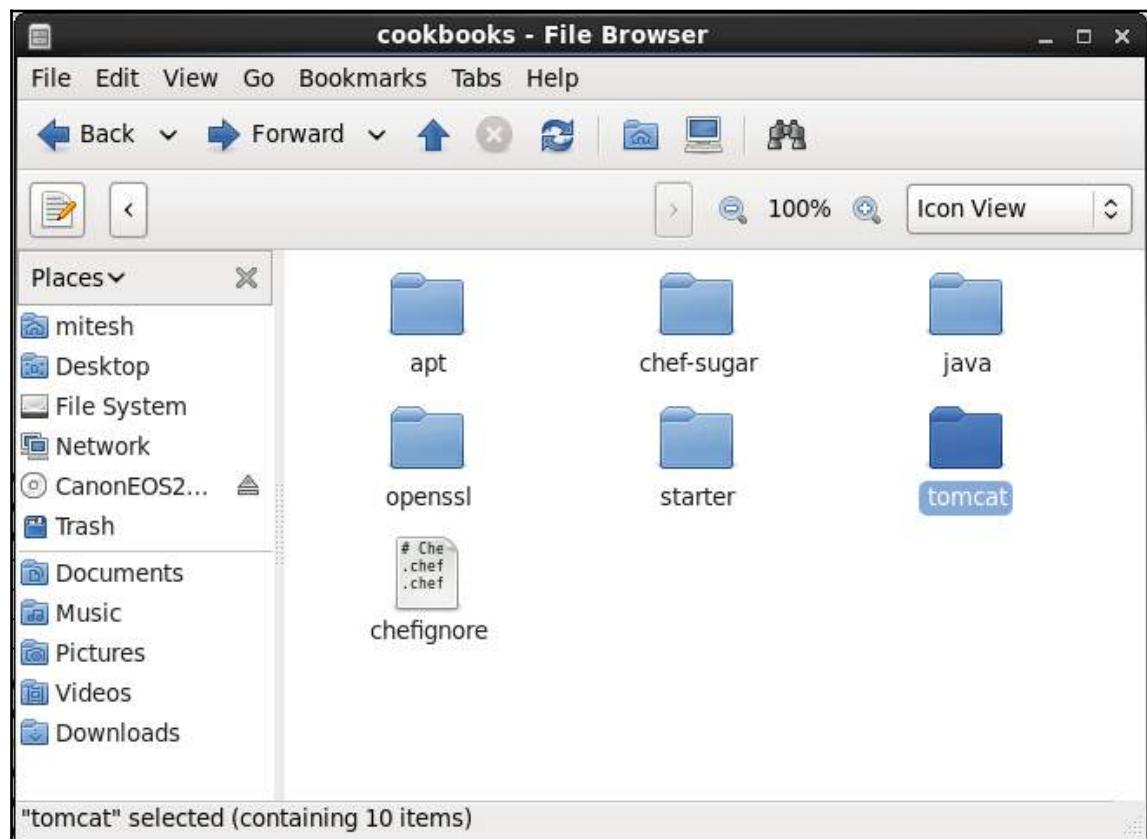
Now it is time to install software packages using cookbooks. To set up the runtime environment automatically, it's best to use the Chef community cookbooks:

1. Visit <https://github.com/chef-cookbooks> and find all the community cookbooks required to set up a runtime environment, as shown in the following screenshot:



2. We are using a sample Spring application, namely, PetClinic. We need to install Java and Tomcat to run a Java EE application such as this.
3. Download the Tomcat cookbook from <https://supermarket.chef.io/cookbooks/tomcat>, and navigate to the **Dependencies** section on that page. Without the dependencies uploaded to our Chef server, we can't upload the Tomcat cookbook 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. To install Java, download the cookbook from <https://supermarket.chef.io/cookbooks/java> and its dependency as well: <https://supermarket.chef.io/cookbooks/apt>. Extract all compressed files to the cookbooks directory:



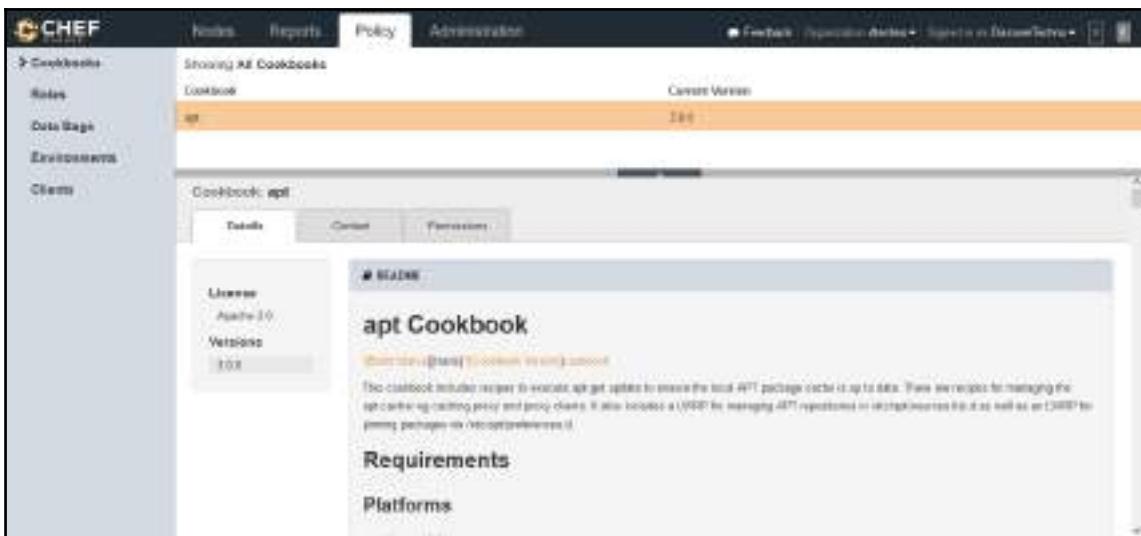
6. Go to cookbooks from the terminal and verify the subdirectories of the community cookbooks.

```
[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:

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

8. Verify from the **Cookbooks** section on the hosted Chef instance whether the **apt Cookbook** has been uploaded:



9. Make sure to upload all dependencies first, or it will give you an 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
Upgrading java           [1.39.0]
Uploaded 1 cookbook.
[root@devops1 chef-repo]# knife cookbook upload openssl
Upgrading openssl         [4.4.0]
```

```
Uploaded 1 cookbook.  
[root@devops1 chef-repo]# knife cookbook upload tomcat  
Uploading tomcat [0.17.0]  
Uploaded 1 cookbook.
```

10. Check whether all the cookbooks have been uploaded from the hosted Chef account:



## Creating a role

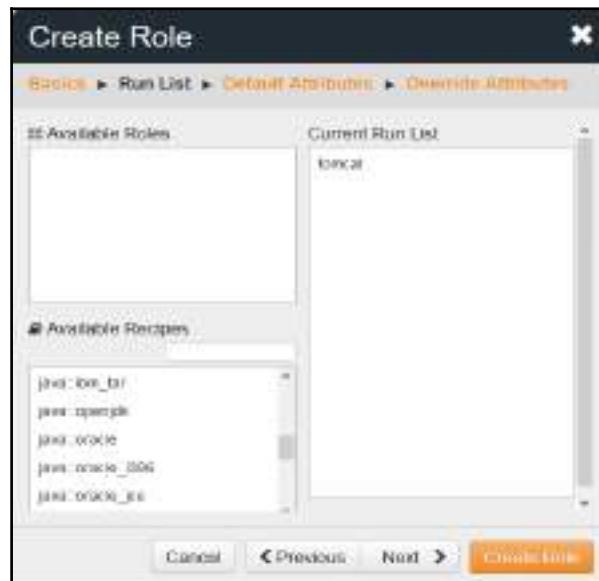
Once all cookbooks have been uploaded successfully, we need to create a role. A role is defined for a specific function and provides a path for different patterns and workflow processes.

For example, the web server role can consist of Tomcat server recipes and any custom attributes:

1. Go to **Policy | Roles | Create** to create a role. In the **Create Role** window, provide a **Name** and **Description** and then click on **Next**, as shown in the following screenshot:



2. A **Run List** keeps roles/recipes in a proper manner and order. We can say that it describes the specifications of a node. Select **tomcat** from the **Available Recipes** section, drag it to the **Current Run List** section, and click on **Create Role**:



3. Verify the newly added role details in the hosted Chef dashboard:

The screenshot shows the Chef Dashboard interface. The top navigation bar includes 'Nodes', 'Reports', 'Policy', and 'Administration'. The left sidebar has sections for 'Cookbooks', 'Roles' (selected), 'Data Bags', 'Environments', and 'Clients'. Under 'Roles', there is a table with columns 'Name', 'Description', and 'Environment'. One row is highlighted for 'v-tomcat' with the description 'Tomcat Installation on VM'. Below this, a modal window titled 'Role: v-tomcat' displays the 'Details' tab, which contains the description 'Tomcat Installation on VM.' and a 'Run List' section. The 'Run List' section shows a single item: 'tomcat'. A 'Verification' section at the bottom right shows a status of '0 17.09'.

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

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

5. The role has now been associated with the node, and the next time the Chef client runs on the node, it will check whether it is in sync with its assignment. If not, it will execute the steps to bring the status in 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...
```

```
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

. <!-- 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" />
- -->
+ -->

* 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
```

6. If you take a look at the output, we will come to know what exactly happens when convergence takes place.

7. Verify the **Reports** section in the hosted Chef account to obtain the latest details:



Now we know how to create a hosted Chef account, configure a workstation, and converge anode. This is the important piece, in the end-to-end automation as with the use of Chef configuration management tool, we have setup runtime environment that is required to run Java EE application.

## Self-test questions

1. Which of these categories does Chef fall into?
  - Continuous integration
  - Configuration management
  - Both of these
  - Neither of these

2. What are the three main components of a Chef installation?

- Chef server
- Chef workstation
- Chef Node
- Cookbooks
- All of these
- None of these

3. Which command can be used to check the version of a Chef client?

- chefclient -version
- chef-client -version
- chefclient --version
- chef-client --version
- None of these

4. What is the name of the configuration file in Chef?

- knife.java
- knife.py
- knife.rb
- knife.sh
- None of these

5. Which command is used for listing a node as available on a Chef server?

- knife node list
- knife client list
- knife node listing
- knife nodes list
- None of these

## Summary

In this chapter, we covered how we can create a hosted Chef account, configure a workstation, upload a community cookbook to a hosted Chef account, converge a node, use community cookbooks to install Tomcat, verify the convergence of a node on a hosted Chef account, and verify success and failure reports. Essentially, we are standardizing the process of setting up a runtime environment from a centralized location. Most of the configuration tools do almost similar things, and you can decide based on experience and other features which configuration management tool you want. Automating the repetitive process in any field is the key to increasing efficiency, and configuration management tools do exactly that in the end-to-end automation of application delivery. In this chapter, we automated installing tomcat and other runtime requirements for sample Java EE application so we can deploy the WAR file created by Continuous Integration process.

In the next chapter, we will discuss Docker, one of the most popular technologies 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 them, and some basics about the technology.

# 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 one of the disruptive innovations of recent times. Docker containers are isolated packages that contain the components required to run an application.

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

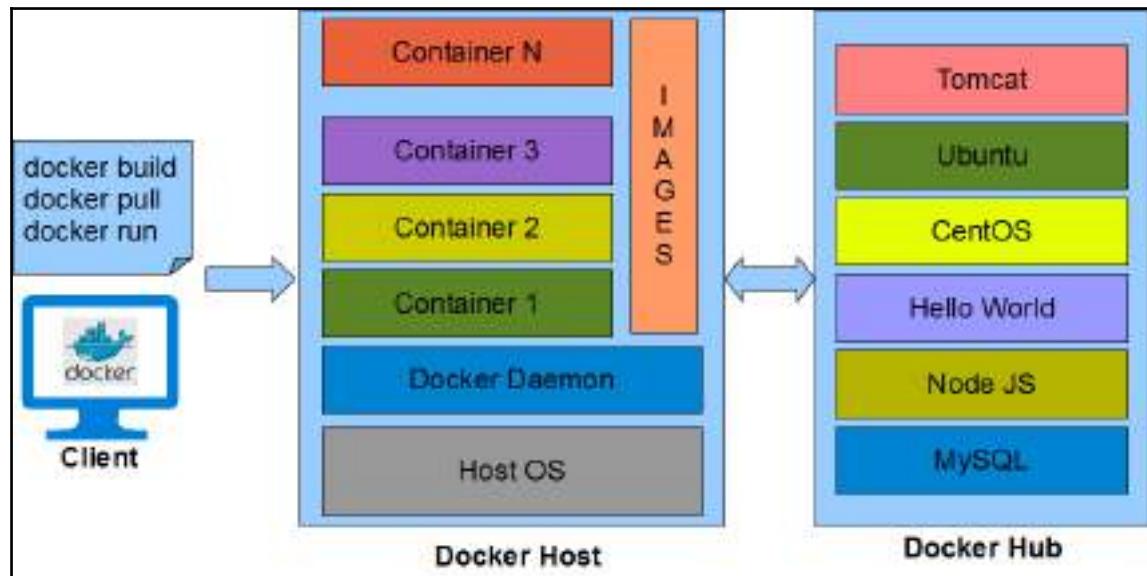
We will also cover Docker Hub and the basic architecture of Docker. We will see how to use the Tomcat image available on Docker Hub and create a sample image with a Java and Tomcat installation and a Dockerfile.

We will cover the following topics:

- Overview of Docker containers
- Understanding the difference between virtual machines and containers
- Installation and configuration of Docker on CentOS
- Creating your first Docker container
- Managing containers

## Overview of Docker containers

Docker is an open source initiative for OS virtualization that automates the deployment of applications inside software containers. It provides isolated user spaces and hence provides user-based processes, space, and filesystems. Behind the scenes, it shares the Linux host kernel. The following diagram illustrates the working mechanism of a Docker container:



Docker has two main components, with a client-server architecture:

- **The Docker host:** The Docker host contains the Docker daemon, containers, and images. The **Docker engine** is an important component that provides the core Docker technology. This core Docker technology enables images and containers. 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, use `docker run -i -t <image> /bin/bash`:
  - The `-i` flag initiates an interactive container
  - The `-t` flag creates a pseudo-terminal that attaches `stdin` and `stdout`
  - The `<image>` is a CentOS image
  - `/bin/bash` starts a shell

- When we run this command, it verifies whether the CentOS image is available locally. If it is not available, it will download the image from Docker Hub.
  - An image has a filesystem and parameter that can be used at runtime, while a container is an instance of an image with a state. It is simple to understand that containers change while images do 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 provided by Docker. As a user, we can use it to build and ship applications. It allows us to create a pipeline to integrate with code repositories and for collaboration, image discovery, and automation.
1. Let's navigate to <https://hub.docker.com> and sign up by providing a username, e-mail, and password:



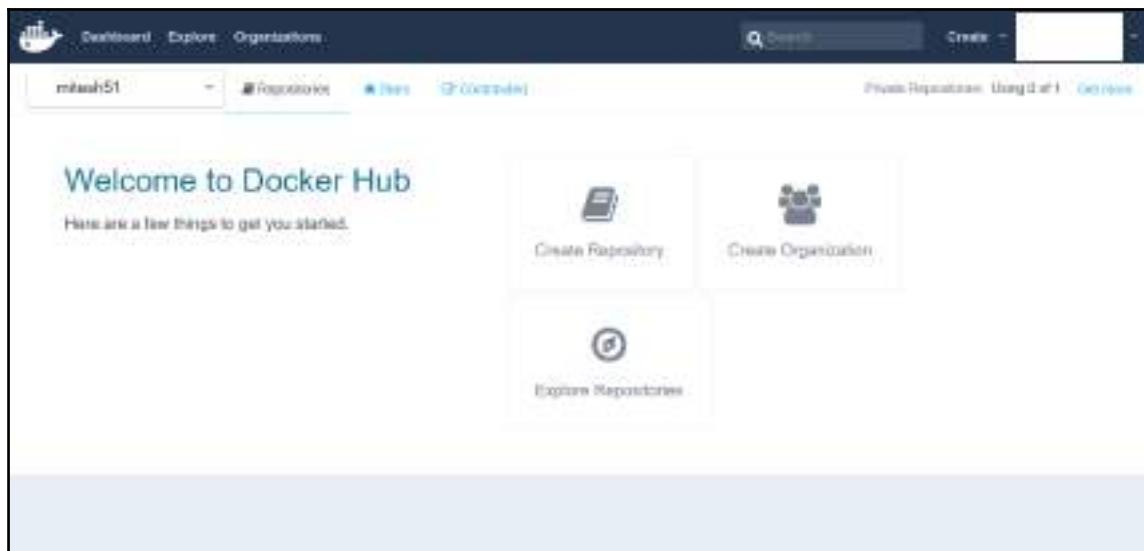
2. Activate your account by clicking on the activation link sent to your e-mail ID:



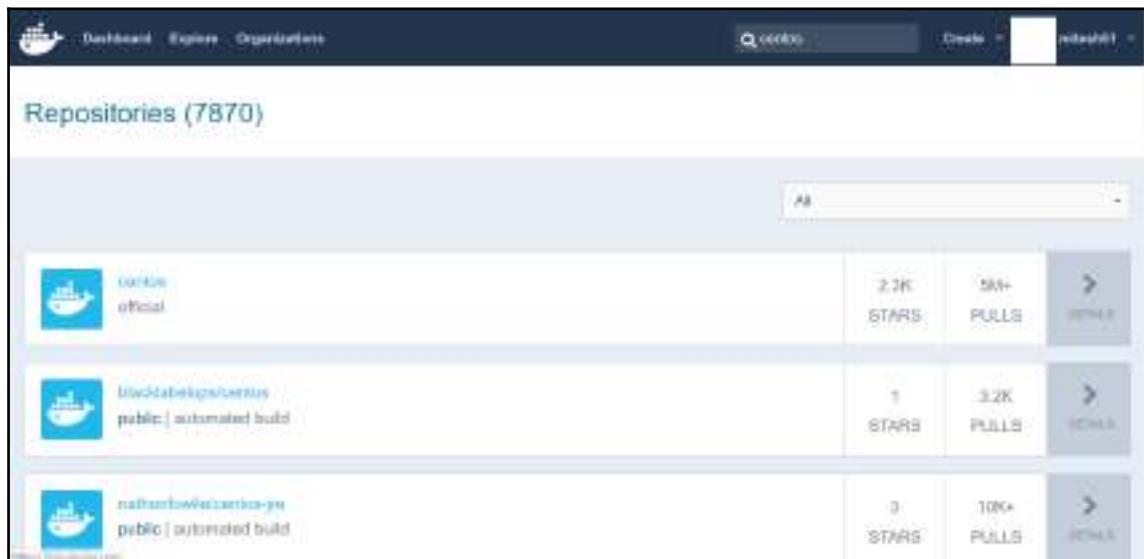
3. After successful activation, login to your Docker Hub account:



4. Following is the screenshot of the Docker **Dashboard**. Explore it as an exercise:



5. Click on **Repositories** to find images available in the public domain. Search for `centos`, and you will get a list of all CentOS images available on Docker Hub:

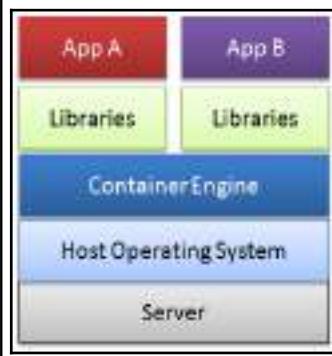


In the next section, we will see why containers are gaining so much attention by comparing them with virtual machines.

## Understanding the difference between virtual machines and containers

In recent times, cloud computing has become part of almost all technical discussions. Virtual machines have served a lot of people in utilizing resources efficiently. However, Docker containers have given them competition and, in fact, containers are more effective.

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

Virtual machines	Containers
<p>In the virtual machine, we need to install an operating system with the appropriate device drivers; hence, the footprint or size of a virtual machine is huge. A normal VM with Tomcat and Java installed may take up to 10 GB of drive space:</p>  <pre>graph TD; Server[Server] --&gt; HostOS[Host Operating System]; HostOS --&gt; Hypervisor[Hypervisor]; Hypervisor --&gt; GuestOS[Guest OS]; GuestOS --&gt; Libraries[Libraries]; Libraries --&gt; AppA[App A]; Libraries --&gt; AppB[App B]</pre>	<p>A container shares the operating system and device drivers of the host. Containers are created from images, and for a container with Tomcat installed, the size is less than 500 MB:</p>  <pre>graph TD; Server[Server] --&gt; HostOS[Host Operating System]; HostOS --&gt; ContainerEngine[Container Engine]; ContainerEngine --&gt; Libraries[Libraries]; ContainerEngine --&gt; AppA[App A]; ContainerEngine --&gt; AppB[App B]</pre>
<p>There's an overhead of memory management and device drivers. A VM has all the components a normal physical machine has in terms of operation.</p>	<p>Containers are small in size and hence effectively give faster and better performance.</p>
<p>In a VM, the hypervisor abstracts resources.</p>	<p>Containers abstract the operating system.</p>

In a 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 and Windows 2003.	A container runs as an isolated user space, with processes and filesystem in the user space on the host operating system itself, and it shares the kernel with other containers. Sharing and resource utilization are at their best in containers, and more resources are available due to less overhead. It works with very few required resources.
Cloud service providers use a hypervisor to provide a standard runtime environment for VMs. Hypervisors come in type 1 and type 2 categories.	Docker makes it efficient and easier to port applications across environments.

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

## Installing and configuring Docker on CentOS

To create a virtual machine using VMware Workstation or VirtualBox, install CentOS 6.6 or 6.7.

Follow these steps to use CentOS 6.7 to run Docker. In CentOS 6.x, there was a minor issue of a package name conflict with a system tray application and its executable, so 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 the 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. The sample image execution didn't complete successfully as the Docker service wasn't running. Let's verify the Docker installation:

1. 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
```

2. Verify the status of the Docker service:

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

So we have now successfully installed Docker and verified whether its services are running on a CentOS 6.7 virtual machine.

## Creating your 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.

The `hello-world` image is not available locally, so it will be fetched from 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 performs the following steps:

1. The Docker client communicates with the Docker daemon.
2. Then Docker daemon pulls the `hello-world` image from Docker Hub.
3. After that, the Docker daemon creates a new container from that image, which runs the executable that produces the output you are currently reading.
4. When the executable gets executed in a newly created container, the Docker daemon streams that output to the Docker client, which sends it to your terminal.

Let's try something more ambitious:

1. You can run an Ubuntu container with this command:

```
$ docker run -it ubuntu bash
You have new mail in /var/spool/mail/root
[root@localhost Desktop]#
```



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

2. Now we have one image available locally. Let's try to create an Ubuntu container and open its bash shell 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 the `docker images` command to verify that the existing images are available locally:

```
[root@localhost Desktop]# docker images
REPOSITORY      TAG      IMAGE ID      CREATED      VIRTUALSIZE
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 the client-server architecture of Docker using another example of a Tomcat container.

## Understanding the client-server architecture of Docker

Let's recollect our main objective: we want to deploy a sample Spring application named `Pet-clinic` on our Tomcat server. How it is different when we install tomcat in the virtual machine and use containers? In Container environment, host OS is installed and then it is used to host container layer. Container layer is used for provisioning container instances. Container instances are extremely lightweight and efficient as extra libraries or resources are needed for the operating system that is needed in the virtual machine while not in case of containers.

For that, in the rest of the section, we will try to use the existing Tomcat image and also create a sample image with a Tomcat installation:

1. Navigate to [https://hub.docker.com/\\_/tomcat/](https://hub.docker.com/_/tomcat/), and after you login, search for `tomcat` in the search section. Click on **tomcat**, and you will be presented with something like this:

The screenshot shows a web browser displaying the Docker Hub page for the 'tomcat' repository. The URL in the address bar is [https://hub.docker.com/\\_/tomcat/](https://hub.docker.com/_/tomcat/). The main heading is 'Supported tags and respective Dockerfile links'. Below it is a list of supported tags, each with a link to its Dockerfile. At the bottom left, there is a button labeled 'ImageLayers.io' with '0 B / 25 Layers'.

- 6.0.45-jre7 , 6.0-jre7 , 6-jre7 , 6.0.45 , 6.0 , 6 ([6/jre7/Dockerfile](#))
- 6.0.45-jre8 , 6.0-jre8 , 6-jre8 ([6/jre8/Dockerfile](#))
- 7.0.69-jre7 , 7.0-jre7 , 7-jre7 , 7.0.69 , 7.0 , 7 ([7/jre7/Dockerfile](#))
- 7.0.69-jre8 , 7.0-jre8 , 7-jre8 ([7/jre8/Dockerfile](#))
- 8.0.35-jre7 , 8.0-jre7 , 8-jre7 , jre7 , 8.0.35 , 8.0 , 8 , latest ([8.0/jre7/Dockerfile](#))
- 8.0.35-jre8 , 8.0-jre8 , 8-jre8 , jre8 ([8.0/jre8/Dockerfile](#))
- 8.5.2-jre8 , 8.5-jre8 , 8.5.2 , 8.5 ([8.5/jre8/Dockerfile](#))
- 9.0.0.M6-jre8 , 9.0.0-jre8 , 9.0-jre8 , 9-jre8 , 9.0.0.M6 , 9.0.0 , 9.0 , 9 ([9.0/jre8/Dockerfile](#))

ImageLayers.io 0 B / 25 Layers

2. Verify the images with `docker images`, and then try to run the Tomcat image. It will take a while.

- Once image is pulled completely, the container will be created and a bash shell will be available for command execution:

```
[root@localhost Desktop]# docker run -it tomcat bash
```

```
[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
499b5c54fied: 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
```

- Let's try to install Tomcat 8.0; you'll notice that the image is 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
.
.
.
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 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. The container is created successfully. Verify existing containers using the docker ps command:

[root@localhost Desktop]# docker ps			
CONTAINER ID	IMAGE	COMMAND	CREATED
STATUS	PORTS	NAMES	
c3fb72a1b35	tomcat:8.0	"catalina.sh run"	29 minutes ago
Up 29 minutes	8080/tcp	sad_pasteur	.

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

Use docker inspect with the container ID to find out the IP address of the container:

```
[root@localhost Desktop]# docker inspect c3fb72a1b35
[
{
  "Id": "c3fb72a1b35c6725606df726b5651cbd774b02d55bad6352c0e5205894b8b56",
  "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 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:

The screenshot shows a web browser window with the URL `172.17.0.3:8080`. The page title is "Apache Tomcat/8.0.36". At the top, there is a navigation bar with links to Home, Documentation, Configuration, Examples, Wiki, and Mailing Lists, along with a "Find Help" search bar. To the right of the navigation bar is the "The Apache Software Foundation" logo and the URL `http://www.apache.org/`.

The main content area features a green banner with the text "If you're seeing this, you've successfully installed Tomcat. Congratulations!" Below this, there is a cartoon cat icon and a section titled "Recommended Reading:" with links to "Security Considerations HOW-TO", "Manager Application HOW-TO", and "Clustering/Session Replication HOW-TO".

On the right side, there are three buttons: "Server Status", "Manager App", and "Host Manager".

Below the banner, there is a "Developer Quick Start" section with links to "Tomcat Setup", "First Web Application", "Roles & AAA", "JDBC Datasources", "Examples", "Server Specifications", and "Tomcat Versions".

The page is divided into three main columns:

- Managing Tomcat:** Discusses security access to the manager webapp and mentions that access is split between different users. It includes a "Read more..." link and a "Release Notes" section.
- Documentation:** Links to "Tomcat 8.0 Documentation", "Tomcat 8.0 Configuration", and "Tomcat Wiki". It also provides information about configuration files like `CATALINA_HOME/conf/tomcat-users.xml` and `CATALINA_HOME/running.txt`.
- Getting Help:** Includes links to "FAQ and Mailing Lists" and a list of available mailing lists: `tomcat-announcements`, `tomcat-dev`, `tomcat-user`, and `tomcat-interest`.

So finally, we are able to run a Tomcat container. In the 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 the Tomcat container as a background process.

1. It is best practice to run a Docker container as a background process to avoid accidentally stopping containers from the terminal:

2. Use the `-d` parameter:

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

3. Verify the container you just created:

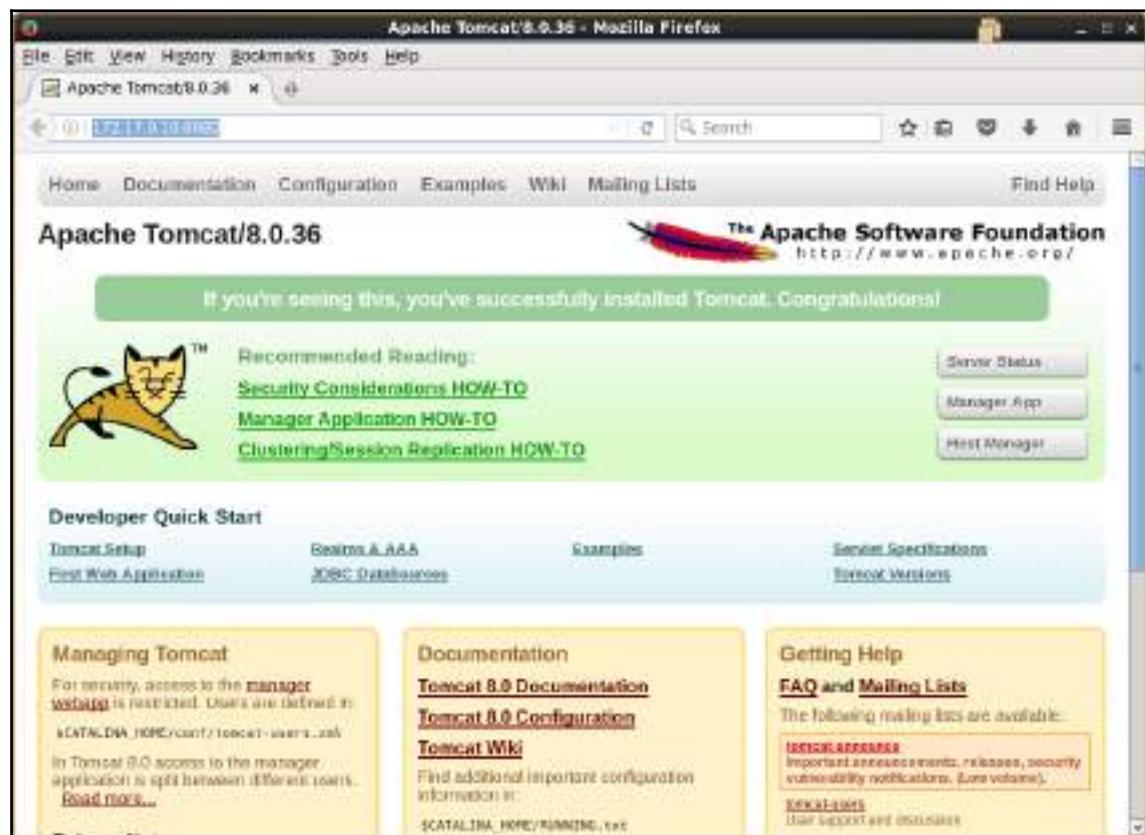
CREATED NAMES	CONTAINER ID	IMAGE	STATUS	COMMAND	PORTS
	68c6d1f7bc63	tomcat	"catalina.sh run" seconds ago desperate_hypatia	Up 11 seconds	8080/tcp
			You have new mail in /var/spool/mail/root		15

4. Get the IP address of the container with the `docker inspect` command along with the 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":  
    "5d4577339b146f4e71ddb267812213bcd1a612eeb48a5f3c95f105b7894a4a73",  
  "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
},
{
  "Image": "tomcat",
  "Volumes": null,
  "VolumeDriver": "",
  "WorkingDir": "/usr/local/tomcat",
  "Entrypoint": null,
  "NetworkDisabled": false,
  "MacAddress": "",
  "OnBuild": null,
  "Labels": {}
}
]
```

5. Note the IP address `http://172.17.0.10:8080/` and try to access it from the browser:



6. Now the obvious question is how to stop containers, right? To obtain the details of running containers, use `docker ps`.
7. Observe the last column, **Names**; you can see a strange name `desperate_hypatia` being automatically allocated to a container if it is not given a name explicitly:

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

8. Let's stop the container using this automatically assigned container name:

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

9. If we want to provide a custom name to the container, then we can rename it using the --name operator, as shown in the following command:

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

10. Can we see the list of all containers that have been stopped? Yes: use the docker ps -a command, to get the list of stopped containers:

```
[root@localhost Desktop]# docker ps -a  
CONTAINER ID        IMAGE               COMMAND             CREATED             STATUS              PORTS  
NAMES               STATUS              PORTS  
STATUS             COMMAND             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  
a6f402e7a2a8       ubuntu            "ls"              43 minutes  
ago    Exited (0)   43 minutes ago  
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
```

- A container's lifetime is limited to the existence of a parent process.



```
Let's run the container from the image with Tomcat to deploy application in it.  
[root@localhost Desktop]# docker run -p 8080:9090 -d --name devops_tomcat9 tomcat  
0f8c251929b2f316bac1d53c5b8d03a155d790dada1ce2fcf94f95844a3acfef
```

11. To get access to a terminal on the container, use the following command after creating the container:

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

12. Once you have access to the container console, verify its IP address using ip addr show eth0:

```
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#
```

13. Now, let's try to search for the Tomcat images available on Docker Hub. Try docker search tomcat command:

```
[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]  
.  
.  
.
```

```
davidcaste/debian-tomcat Yet another Debian Docker image for
Tomcat... 0 [OK]
```

14. 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
```

## Creating a Docker image from Dockerfile

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

Let's try with a simple CentOS image:

1. The Dockerfile contains the following two lines:

```
FROM centos
MAINTAINER mitesh <mitesh.sxxxxxxxxx@xxxxxxxxx.com>
```

2. Go to the same directory in the 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.sxxxxxxxxx@xxxxxxxxx.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 built a sample Docker image. Now, let's verify it by executing the following command:

```
[root@localhost Desktop]# docker images
```

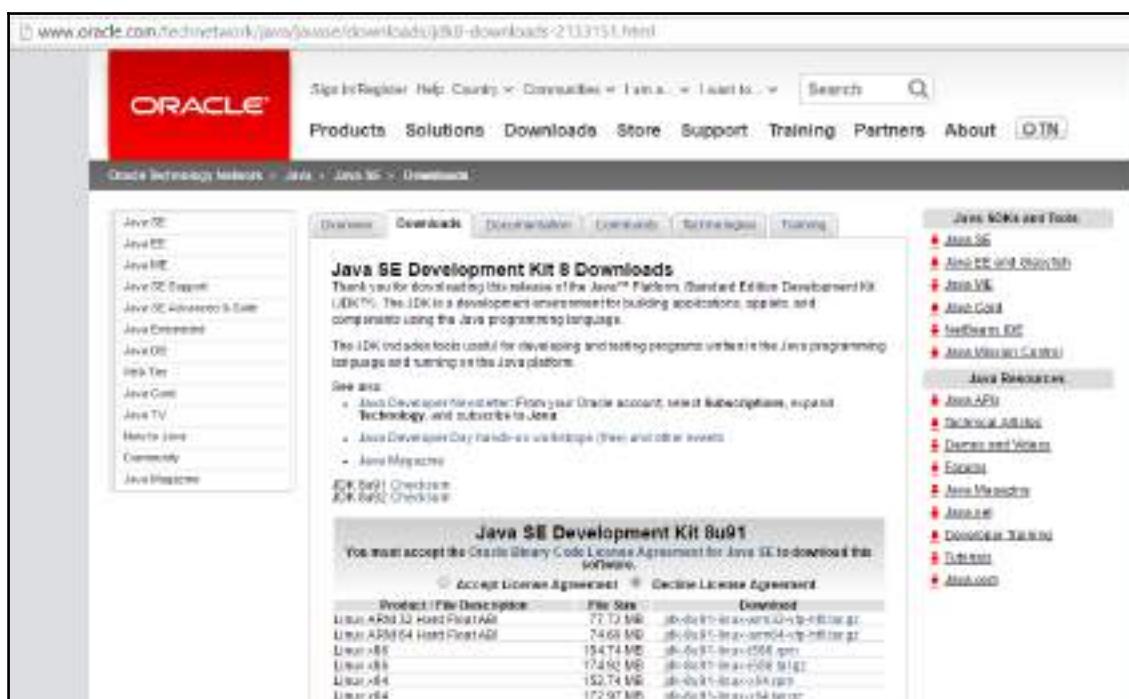
REPOSITORY	TAG	IMAGE ID	CREATED	VIRTUAL SIZE
<none>	<none>	b636e26a333a	16 seconds ago	196.7 MB
tomcat	8.0	5d4577339b14	7 days ago	359.2 MB
tomcat	latest	5d4577339b14	7 days ago	359.2 MB
centos	latest	2a332da70fd1	2 weeks ago	196.7 MB
ubuntu	latest	686477c12982	7 weeks ago	120.7 MB
hello-world	latest	f1d956dc5945	8 weeks ago	967 B

- Now, we will create an image with Java 8 and Tomcat 9 to understand how we can create a sample image. Verify whether you have a 32- or 64-bit operating system. Based on that, we will download the respective installable files:

```
[root@localhost Desktop]# uname -m  
x86_64
```

We have a 64-bit operating system, so for Java, we will use the 64-bit installer:

1. The download URL for Java is <http://www.oracle.com/technetwork/java/java8-downloads-jdk8-downloads-2133151.html>:



2. Download Tomcat from <http://apache-mirror.rbc.ru/pub/apache/tomcat/>:

The screenshot shows a web browser window with the URL <http://apache-mirror.rbc.ru/pub/apache/tomcat/> in the address bar. The page title is "Index of /pub/apache/tomcat". Below the title is a table listing various Tomcat releases and connectors. The columns are "Name", "Last modified", "Size", and "Description".

Name	Last modified	Size	Description
<a href="#">Parent Directory</a>		-	
<a href="#">maven-plugin/</a>	2015-02-17 21:55	-	
<a href="#">taglibs/</a>	2015-05-14 16:48	-	
<a href="#">tomcat-6/</a>	2016-04-12 10:39	-	
<a href="#">tomcat-7/</a>	2016-06-21 11:32	-	
<a href="#">tomcat-8/</a>	2016-06-13 18:53	-	
<a href="#">tomcat-9/</a>	2016-06-13 18:31	-	
<a href="#">tomcat-connectors/</a>	2015-02-17 21:55	-	

3. For a Java and Tomcat installation, we have the following Dockerfile:

```
FROM centos
MAINTAINER mitesh <mixxxx.xxxx@xxxxxx.com>
RUN yum -y update && yum -y install wget && yum -y install tar

# Set Environment Variables
ENV JAVA_HOME /usr/java
ENV CATALINA_HOME /usr/tomcat
ENV PATH $PATH:$JAVA_HOME/bin:$CATALINA_HOME/bin

# Download and Install Java 8 :

http://www.oracle.com/technetwork/java/javase/downloads/jdk8-downloads-
2133151.html
RUN wget --no-cookies --no-check-certificate --header "Cookie:
gpw_e24=http%3A%2F%2Fwww.oracle.com%2F; oraclelicense=accept-securebackup-
cookie"
"http://download.oracle.com/otn-pub/java/jdk/8u92-b14/jdk-8u92-linux-x64.t
ar.gz" && tar -xvf jdk-8u92-linux-x64.tar.gz && rm jdk-8u92-linux-x64.tar.gz
&& mv jdk* ${JAVA_HOME}

# Download and Install Tomcat 9 :
```

```
http://apache-mirror.rbc.ru/pub/apache/tomcat/
RUN wget
http://apache-mirror.rbc.ru/pub/apache/tomcat/tomcat-9/v9.0.0.M8/bin/apache-
tomcat-9.0.0.M8.tar.gz && tar -xvf apache-tomcat-9.0.0.M8.tar.gz && rm
apache-tomcat-9.0.0.M8.tar.gz && mv apache-tomcat*
${CATALINA_HOME}

WORKDIR /usr/tomcat

EXPOSE 8080
EXPOSE 8009
```

4. Let's run the Dockerfile and build an image out of it:

```
[root@localhost Desktop]# docker build -t devopstomcat .
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.soni@outlook.com>
--> Using cache
--> b636e26a333a
Step 2 : RUN yum -y update && yum -y install wget && yum -y
install
--> Using cache
--> 665ffbc90cba
Step 3 : ENV JAVA_HOME /usr/java
--> Using cache
--> 0be3176a4b86
Step 4 : ENV CATALINA_HOME /usr/tomcat
--> Using cache
--> 9c8cccd332f45
Step 5 : ENV PATH $PATH:$JAVA_HOME/bin:$CATALINA_HOME/bin
--> Using cache
--> 64f697c88093
Step 6 : RUN wget --no-cookies --no-check-certificate --header
"Cookie: gpw_e24=http%3A%2F%2Fwww.oracle.com%2F; oraclelicense=accept-
securebackup-
cookie""http://download.oracle.com/otn-pub/java/jdk/8u92-b14/jdk-8u92-linux-
-x64.tar.gz"&& tar -xvf jdk-8u92-linux-x64.tar.gz && rm jdk-8u92-linux-
x64.tar.gz && mv jdk* ${JAVA_HOME}
--> Running in 116b0e860348
--2016-06-23 19:48:41--
http://download.oracle.com/otn-pub/java/jdk/8u92-b14/jdk-8u92-linux-x64.tar
.gz
Resolving download.oracle.com (download.oracle.com)...
203.192.223.200, 203.192.223.202
Connecting to download.oracle.com
```

```
(download.oracle.com) |203.192.223.200|:80... connected.
    HTTP request sent, awaiting response... 302 Moved Temporarily
    Location:
https://edelivery.oracle.com/otn-pub/java/jdk/8u92-b14/jdk-8u92-linux-x64.t
ar.gz [following]
.

.

.

Connecting to download.oracle.com
(download.oracle.com) |203.192.223.200|:80... connected.
    HTTP request sent, awaiting response... 200 OK
    Length: 181389058 (173M) [application/x-gzip]
    Saving to: 'jdk-8u92-linux-x64.tar.gz'

    0% ..... 1.12M 2m35s
    50K ..... 4.88M 95s
    .
    .
    .
    177100K ..... 397K=3m22s
100% 2016-06-23 19:52:06 (878 KB/s) - 'jdk-8u92-linux-x64.tar.gz'
saved [181389058/181389058]

jdk1.8.0_92/
jdk1.8.0_92/javafx-src.zip
jdk1.8.0_92/bin/
jdk1.8.0_92/bin/jmc
jdk1.8.0_92/bin/serialver
.
.
.

jdk1.8.0_92/README.html
--> b025a8495f67
Removing intermediate container 116b0e860348
Step 7 : RUN wget
http://apache-mirror.rbc.ru/pub/apache/tomcat/tomcat-9/v9.0.0.M8/bin/apache-
tomcat-9.0.0.M8.tar.gz && tar -xvf apache-tomcat-9.0.0.M8.tar.gz && rm
apache-tomcat-9.0.0.M8.tar.gz && mv apache-tomcat* ${CATALINA_HOME}
--> Running in 485e2f6059b0
--2016-06-23 19:53:18--
http://apache-mirror.rbc.ru/pub/apache/tomcat/tomcat-9/v9.0.0.M8/bin/apache-
tomcat-9.0.0.M8.tar.gz
Resolving apache-mirror.rbc.ru (apache-mirror.rbc.ru)...
80.68.250.217
Connecting to apache-mirror.rbc.ru (apache-
mirror.rbc.ru)|80.68.250.217|:80... connected.
```

So, we have successfully built a sample image using a Dockerfile.

This was just a quick example to get started and familiar with Docker and its concepts.

## Self-test questions

State whether the following statements are true or false:

1. Has Docker a client-server architecture?
2. Docker has two main components: the Docker host and Docker Hub?
3. While creating a container, the image has to be available locally or the operation fails?
4. Is Docker Hub used to store and manage containers?
5. The overhead of memory management and device drivers is extremely high in Docker containers?
6. For CentOS6, the Docker RPM package is called `docker-io`?
7. The `docker ps -a` command is used to see the list of stopped containers?

## Summary

In this chapter, we had an overview of Docker containers, architecture details, and details of the main components of Docker, including a quick overview of Docker Hub. Based on the overview, we tried to compare virtual machines with Docker containers to gain a clear picture of why containers have recently been gaining traction.

After gaining some understanding of virtual machines and containers, we covered the process of installing Docker on a CentOS 6.x virtual machine. We created a `hello-world` container and Ubuntu and CentOS containers from the images available on Docker Hub.

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

On the subject of containers, this quote by Ted Engstrom 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.”*

In the next chapter, we will see how to use Chef to create a virtual machine in Amazon Web Services and Microsoft Azure and set up a runtime environment.

# 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 our goal was in the first chapter. Our main objective is to create an 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 the cloud and containers, continuous delivery, continuous deployment, continuous monitoring, continuous feedback, continuous improvement, and continuous innovation. We want to use an end-to-end pipeline for our sample Spring application, PetClinic. In [Chapter 4, Installing and Configuring Chef](#) and [Chapter 5, Installing and Configuring Docker](#), we covered the configuration management tool Chef and Docker containers in brief. Both could be topics for book on their own. Now we are at the stage where we understand the basics of configuration management and containers, so we can start with resource provisioning in a cloud environment using Chef and install the runtime environment required to run PetClinic. In this scenario, it will be an installation of Java and Tomcat.

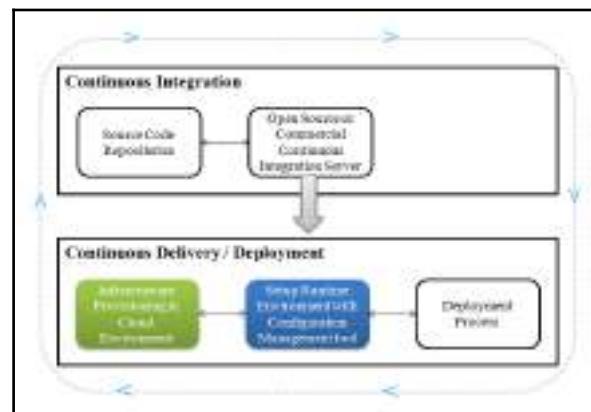
This chapter describes in detail how to install knife plugins used to manage cloud resources using Chef. It will cover creating instances in AWS and Azure using the **knife EC2** and **knife Azure** plugins. It will also cover how Chef is used to manage Docker containers.

We will explore the following topics:

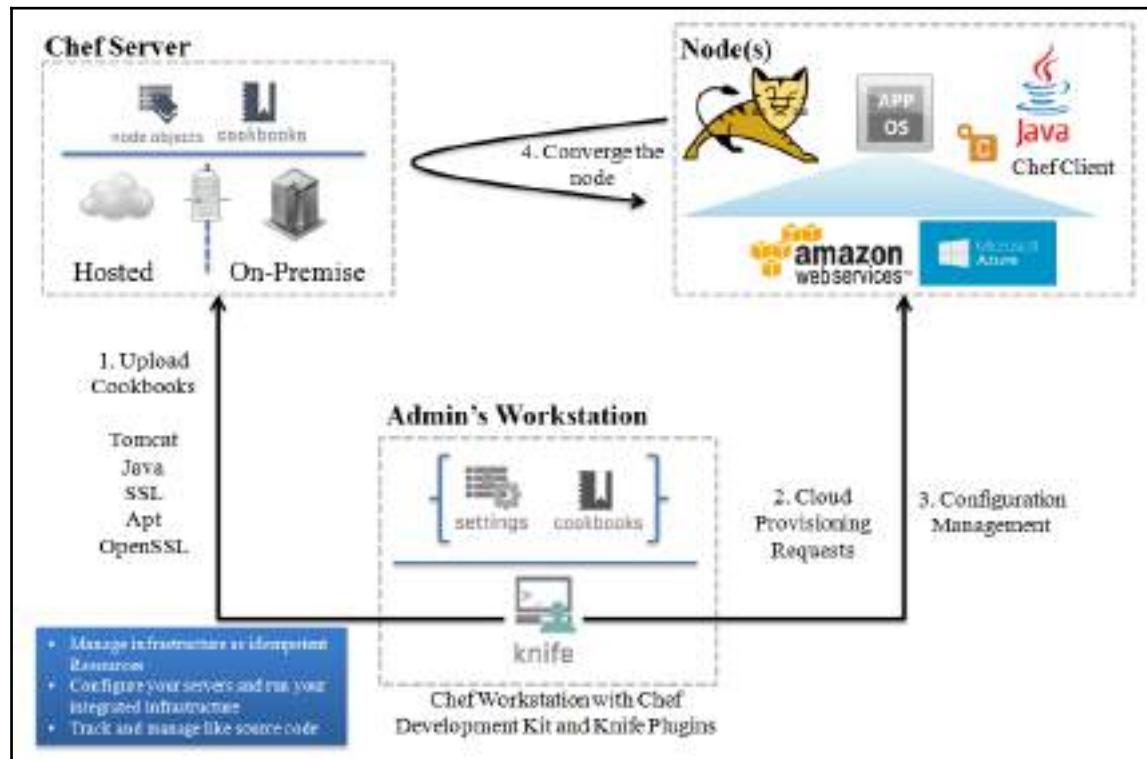
- Chef and cloud provisioning
- Installing knife plugins for Amazon EC2 and Microsoft Azure
- Creating and configuring a virtual machine in Amazon Web Services
- Creating and configuring a virtual machine in Microsoft Azure
- Managing Docker containers with Chef

## Chef and cloud provisioning

Chef is not only used for setting up runtime environments or configuration management, but it is also used for resource provisioning in cloud environments. It supports cloud service providers such as Microsoft Azure, Amazon Web Services, VMware, OpenStack, HP Cloud, and Google Compute Engine. Chef provides more flexibility to the concept of infrastructure as a code and brings configuration management into the picture as well. Knife plugins are used to manage or use different cloud service providers. With knife plugins, it is easier to provision and deprovision resources along with controlled and centralized configuration management. In this chapter we will focus on infrastructure provisioning in Cloud environment and setting up runtime environment as shown in below diagram:



We will specifically focus on infrastructure provisioning in a cloud environment and setting up a runtime environment with a configuration management tool:



We will provision resources in a public cloud environment using knife plugins using a Chef workstation. We configured a Chef workstation in Chapter 4, *Installing and Configuring Chef*. From the Chef workstation, we can execute knife commands to create instances (Chef nodes) in different cloud environments. In our case, we will provision resources in Amazon EC2 and Microsoft Azure. This is how the process will work:

1. Chef workstation to CSP: Create a new instance in your cloud environment.
2. CSP: OK...done! The new instance is up and running (the Chef node is available).
3. Chef node to Chef server: Hello!
4. Chef server to Chef node: Here is your task-download the Chef client.
5. Chef server <> Chef node: A secure handshake is made; the Chef server generates a security certificate. The security certificate is used to authenticate the new node's upcoming requests.

6. Chef server to Chef node: Here is the list of recipes you need to install.
7. Chef node to Chef server: Thank you; I've been updated!

Some of the major benefits we get through using Chef with different cloud platforms are as follows:

- Easy policy enforcement with centralized control
- Setup of a consistent runtime environment
- Building a repeatable infrastructure to avoid manual effort and errors
- Rapid deployment of new applications
- Easy restoration of environments
- Disaster recovery and business continuity
- Community-based cookbooks and recipes
- Faster time to market to remain in competition
- Support for 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

The **Chef Development Kit (ChefDK)** comes with development tools built by the Chef community. It makes the task of installing knife plugins easier.

Go to <https://downloads.chef.io/chef-dk/> and download the ChefDK for your platform. For our purposes, select **Red Hat Enterprise Linux** and select the version. Click on the Red Hat Enterprise Linux 6 **Download** button as it works on 64 bit (x86\_64) versions of Red Hat Enterprise Linux and CentOS 6:

```
[root@localhost Desktop]# sudo rpm -ivh chefdk-0.13.21-1.el6.x86_64.rpm
Preparing... #################################
[100%]      1:chefdk          #####
[100%]
Thank you for installing Chef Development Kit!
```

Once we have the ChefDK installed, we can use `chef gem install knife-ec2` to create, bootstrap, and manage EC2 instances. It is available at <https://github.com/chef/knife-ec2>.

```
[root@localhost Desktop]# chef gem install knife-ec2
Fetching: knife-ec2-0.13.0.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 knife-ec2-0.13.0
1 gem installed
```

Once `knife-ec2` has been installed successfully, we should verify the available EC2 commands:

```
[root@localhost Desktop]# knife ec2 --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 can configure Amazon EC2 credentials for `knife ec2` in the `knife.rb` file using `knife[:aws_access_key_id]` and `knife[:aws_secret_access_key]`, like this:

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

Once we have the ChefDK installed, we can use the `chef gem install knife-azure` plugin, which is used to create, delete, and enumerate Microsoft Azure resources to be managed by Chef. The Chef knife plugin for Microsoft Azure is available at <https://github.com/chef/knife-azure>.

```
[root@localhost Desktop]# chef gem install knife-azure -v 1.5.2
Fetching: knife-azure-1.5.2.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 knife-azure-1.5.2
1 gem installed
```

Once `knife-azure` has been installed successfully, we should verify the available Azure commands:

```
[root@localhost Desktop]# knife azure --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)
```

Chef knife has support for VMware Workstation and allows deployments against a workstation. It is available at <https://github.com/chipx86/knife-wsfusion>:

```
[root@localhost Desktop]# chef gem install knife-wsfusion
Fetching: knife-wsfusion-0.1.1.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 knife-wsfusion-0.1.1
1 gem installed
You have new mail in /var/spool/mail/root
```

Once knife-wsfusion has been installed successfully, verify its available commands:

```
[root@localhost Desktop]# knife wsfusion --help
** WSFUSION COMMANDS **
knife wsfusion create (options)
```

Thus, we have installed the knife plugins required for AWS and Microsoft Azure.

In the next section, we will try to create a virtual machine using Amazon EC2.

## Creating and configuring a virtual machine in Amazon EC2

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

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

- After installing knife EC2 plugin, we can use `knife ec2 server create` command with following parameters to create new virtual machine:

Parameter	Value	Description
-I	ami-1ecae776	ID of the 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
```

- At this stage, the AWS EC2 instance has been created and 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 version12.9.41
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 1:chef-12.9.41-1.el6
#####
ec2-52-90-219-205.compute-1.amazonaws.com Thank you for installing
Chef!
```

3. At this stage, the Chef client has been installed on the AWS instance. It is ready for the initial 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. It is now 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. Our runtime environment is setup, and now it is time to start Tomcat services in our 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 Chef Client finished, 13/15
resources updated in 01 minutes 13 seconds
```

6. Here are the details of the newly created AWS instance:

```
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 log in with admin or **Identity and Access Management (IAM)** credentials:

You are using the following Amazon EC2 resources in the US East (N. Virginia) region:

1 Running Instances	0 Elastic IPs
0 Dedicated Hosts	0 Snapshots
1 Volumes	0 Load Balancers
2 Key Pairs	1 Security Groups
0 Placement Groups	

Build and run distributed, fault-tolerant applications in the cloud with Amazon Simple Workflow Service.

### Create Instance

To start using Amazon EC2 you will want to launch a virtual server, known as an Amazon EC2 instance.

[Launch Instance](#)

Now, your instance will launch in the US East (N. Virginia) region.

[Service Health](#) [Scheduled Events](#)

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

The screenshot shows the AWS Management Console with the EC2 service selected. The left sidebar lists navigation options like 'EC2 Dashboard', 'Events', 'Tags', 'Reports', 'Limits', 'Instances', 'Spot Requests', 'Reserved Instances', 'Scheduled Instances', and 'Dedicated Hosts'. The main content area displays a table of instances. The first instance, 'DevOpzVMonAWS', is highlighted with a red box around its row. It has an Instance ID of 'i-048d2de3' and a Public DNS of 'ec2-02-95-219-295.compute-1.amazonaws.com'. The second instance is also listed. Below the table, detailed information for the first instance is shown, including its state as 'running', instance type as 't2.micro', private IP as '10.17.31.17', and secondary private IP as '172.31.1.27'. The Public DNS is listed as 'ec2-02-95-219-295.compute-1.amazonaws.com'.

9. Now, let's go to the hosted Chef dashboard and log in. Click on **Nodes** and verify the newly created/converged node:

The screenshot shows the Chef Dashboard interface. On the left, there is a sidebar with options like 'Datacenter', 'Manage Tags', 'Reset Key', 'Edit Run List', and 'Edit Attributes'. The main area is titled 'Nodes' and shows a table with two rows. The first row is highlighted in orange and labeled 'DEVOPSVMONAWS'. The second row is labeled 'localhost'. The columns in the table are 'Node Name', 'Platform', 'FQDN', 'IP Address', 'Update', 'Last Check In', 'Environment', and 'Actions'. Below the table, a modal window is open for the node 'DEVOPSVMONAWS'. It has tabs for 'Details', 'Attributes', and 'Permissions'. Under 'Details', it shows 'Last Check In: 23 Minutes Ago' (2015-05-14 10:30:15 UTC), 'Update: 2 Minutes' (from 2015-05-14 10:30:15 UTC), 'Environment: default', 'Platform: amzn-2', 'FQDN: ip-172-31-1-27.ec2.internal', and 'IP Address: 172.31.1.27'. There are sections for 'Tags' (empty) and 'Run List' (empty). A note at the bottom says 'There are no run lists to display'.

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

This screenshot shows the 'Node: DevOpsVMonAWS' details page. At the top, there are tabs for 'Details', 'Attributes', and 'Permissions'. The 'Details' tab is selected. It displays 'Last Check In: 23 Minutes Ago' (2015-05-14 10:30:15 UTC), 'Update: 2 Minutes' (from 2015-05-14 10:30:15 UTC), 'Environment: default', 'Platform: amzn-2', 'FQDN: ip-172-31-1-27.ec2.internal', and 'IP Address: 172.31.1.27'. Below this, there are sections for 'Tags' (empty) and 'Run List' (empty). A note at the bottom says 'There are no run lists to display'.

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

Node: DevOpsVMonAWS

Actions: Details Attributes Permissions

Attributes

https  
prox  
sprox  
tomcat

tomcat\_port: 8080  
proxy\_port: 8080  
ssl\_port: 8443  
ssl\_proxy\_port: 8443  
http\_port: 8080  
shadow\_port: 8080  
catalina\_options: 'java -Xms1G -Xmx1G -Djava.awt.headless=true'  
Warning: This node has no run\_list

Everything seems to be nicely done with regard to the creation and configuration of the AWS instance and its registration on hosted Chef.

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

1. You'll see that **The connection has timed out**:



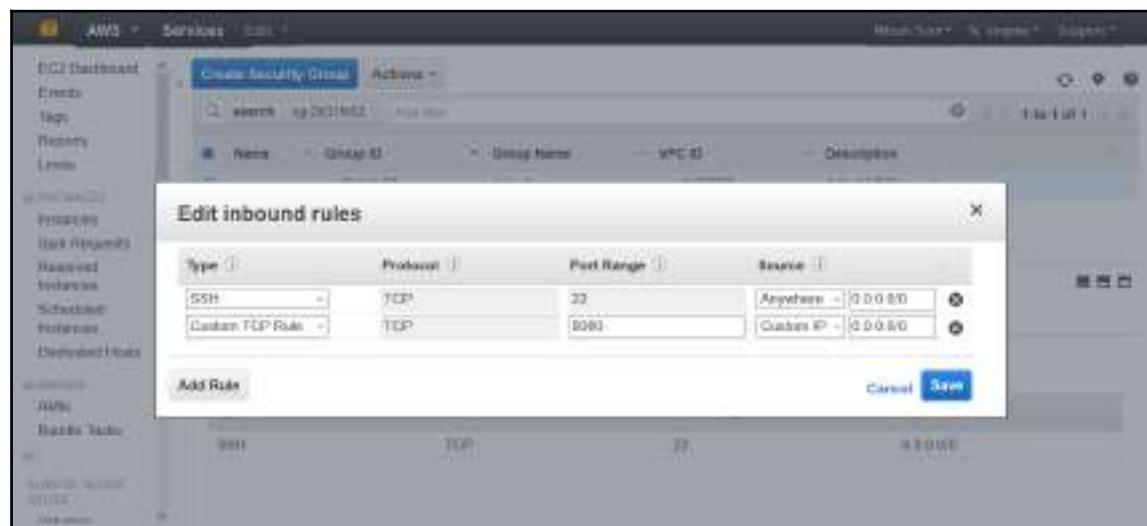
2. The reason for this is the restriction of security groups in AWS. Verify the **Security groups** the AWS instance belongs to:

The screenshot shows the AWS EC2 Instances dashboard. On the left, there's a sidebar with navigation links like AWS Dashboard, Events, Tags, Reports, Units, Instances, and more. The main area displays a table of instances. One instance, named "DevOpsVM01", is highlighted with a red box around its "Security Groups" field. A tooltip appears over this field, explaining: "The security groups to which the instance belongs. A security group is a collection of firewall rules that restrict the network traffic for the instance. Click View rules to see the rules for the specific group." Below the table, there are tabs for Description, Status-CWLogs, Monitoring, and Tags. Underneath the table, detailed information about the instance is shown, including Instance state (running), Instance type (t2.micro), Private DNS (ip-175-31-1-37.ec2.internal), Private IP (173.31.1.37), Secondary private IPs, VPC ID (vpc-6d099999), and Security groups (default). The tooltip is positioned over the "Security groups" row.

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

The screenshot shows the AWS Security Groups dashboard. The sidebar includes links for AWS Dashboard, Events, Tags, Reports, Units, Instances, and more. The main content area shows a table of security groups. One group, named "sg-2b31fe52" with Group Name "default" and VPC ID "vpc-6d099999", is selected and highlighted with a blue border. Below the table, there's a section for the selected security group: "Security Group: sg-2b31fe52". It shows tabs for Description, Inbound, Outbound, and Test. The Inbound tab is selected, displaying a table with one rule: Type (SSH), Protocol (TCP), Port Range (22), and Source (0.0.0.0/0).

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



5. Now, let's verify the URL, and we will get the Tomcat page on our AWS instance.

In the next section, we will see how to create and configure a virtual machine in Microsoft Azure.

## Creating and configuring a virtual machine in Microsoft Azure

For the `knife azure` plugin to communicate with Azure's REST API, we need to provide information to `knife` regarding our Azure account and credentials:

1. Sign in into 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 on the local filesystem and refer to this local file by creating an entry in `knife.rb`:

```
knife[:azure_publish_settings_file] = "~/<name>.publishsettings"
```

The screenshot shows the Microsoft Azure portal interface. At the top, there's a message: "Your subscription file is being generated, and the download will begin shortly." Below this, a note says: "This file contains secure credentials and additional information about subscriptions that you can use in your development environment. Click here if the download does not start automatically." The main content area is a numbered list:

- 1 Sign up for Windows Azure preview features.**  
Sign up for Windows Azure preview features that you are interested in.
- 2 Save a local copy of the publishSettings file.**  
Warning: This file contains an assigned management certificate. It has all your credentials to administer your subscriptions and related services. Store the file in a secure location or delete it after you use it.
- 3 Import the publishSettings file.**  
Run the following command:  
`azurerm account import`
- 4 Create a new Web app.**  
Run the following Azure PowerShell command to create a new web app that is initialized with a Git repository.  
`azurerm webapp create --name`

3. Here are the parameters used to create a virtual machine in Microsoft Azure:

Parameter	Value	Description
--azure-dns-name	dstechnodemo	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

After installing knife azure plugin, let's create virtual machine in Microsoft Azure:

```
[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.

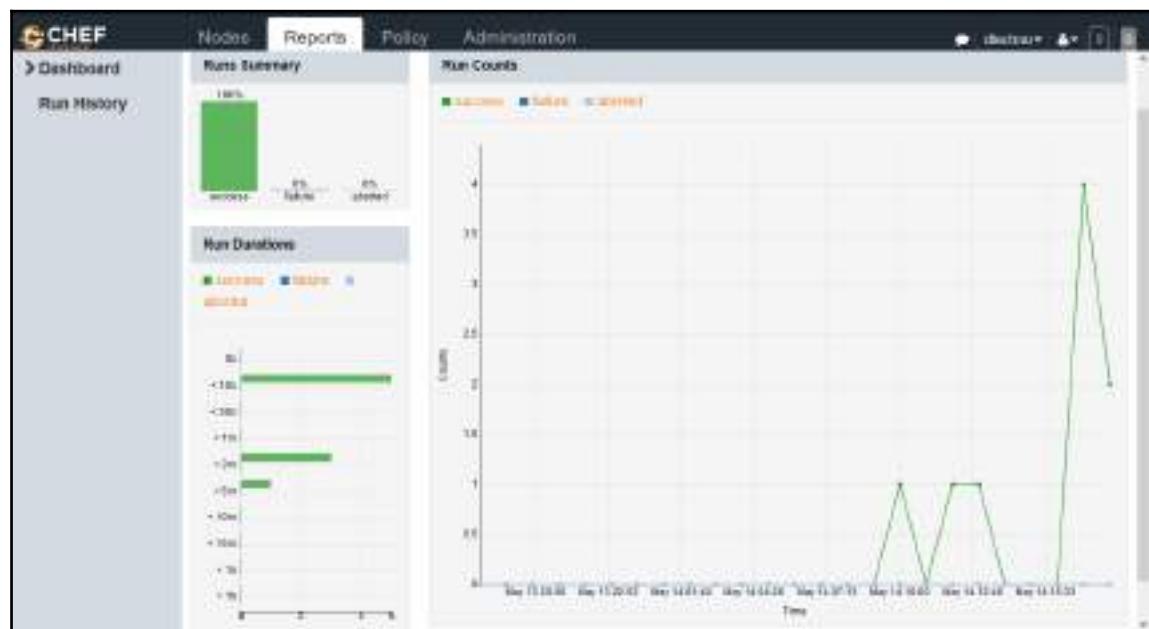
.
DNS Name: distechnodemo.cloudapp.net
VM Name: dtserver02
Size: Small
Azure Source Image: 5112500ae3b842c8b9c604889f8753c3__OpenLogic-
CentOS-67-20160310
Azure Service Location: Central US
Private Ip Address: 100.73.210.70
Environment: _default
Runlist: ["role[v-tomcat]"]
Resource provisioning is going 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]#
```

1. Go to the hosted Chef portal and click on **Nodes** to check whether the new node has been registered on the hosted Chef server:

The screenshot shows the Chef web interface. The top navigation bar includes links for Nodes, Reports, Policy, Administration, and Logout. On the left, there's a sidebar with options like Delete, Manage Tags, Root Key, Edit Run List, and Edit Attributes. The main area displays a table titled "Showing All Nodes" with columns: Node Name, Platform, FQDN, IP Address, Uptime, Last Check-in, Environment, and Action. Five nodes are listed: DevOpsVMonAzure1, DevOpsVMonAWS, sonarqube, DevOpsVMonAzure2, and DevOpsVMonAWS2. The row for DevOpsVMonAzure2 is highlighted with a yellow background. Below this, a modal window is open for the node "DevOpsVMonAzure2". The modal has tabs for Details, Attributes, and Permissions. Under Details, it shows Last Check-in: 34 Minutes Ago (2016-05-14 17:52:4), Uptime: 6 Minutes (Since 2016-05-14 16:50:42), Environment: default, Policies: centos, FQDN: dtserver02.distechnodemo.cloudapp.net, and IP Address: 100.73.210.70.

2. Click on the **Reports** section on the hosted Chef server and verify the graphs for **Runs Summary**, **Run Durations**, and **Run Counts**:

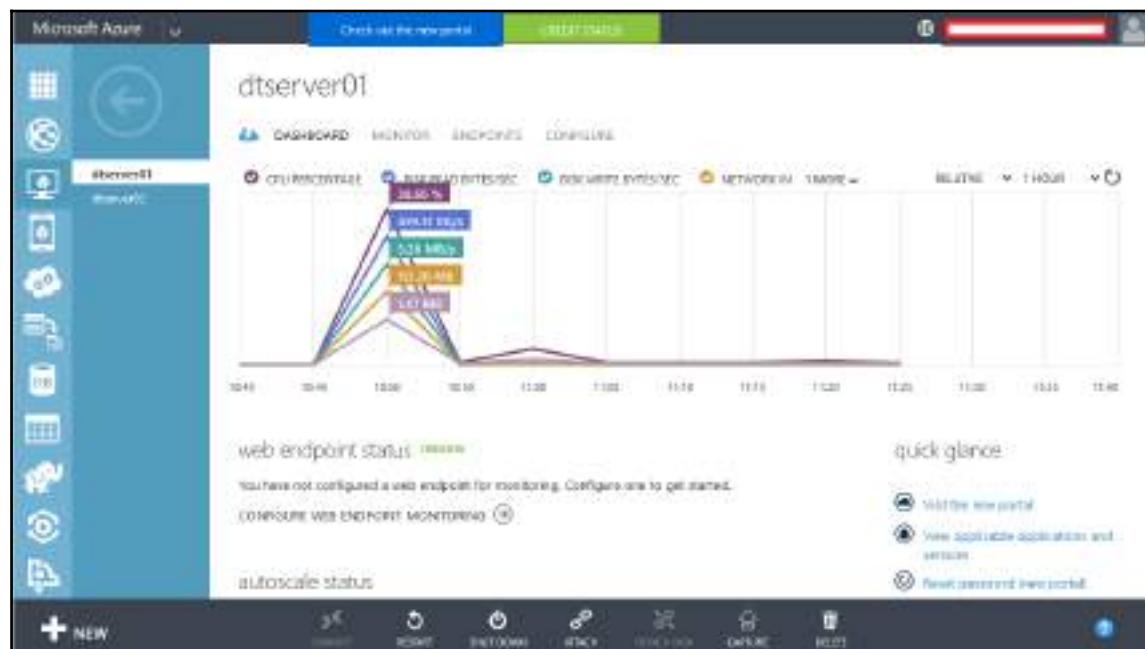


3. Now, let's go to the **classic Azure portal** and verify the newly created virtual machine:

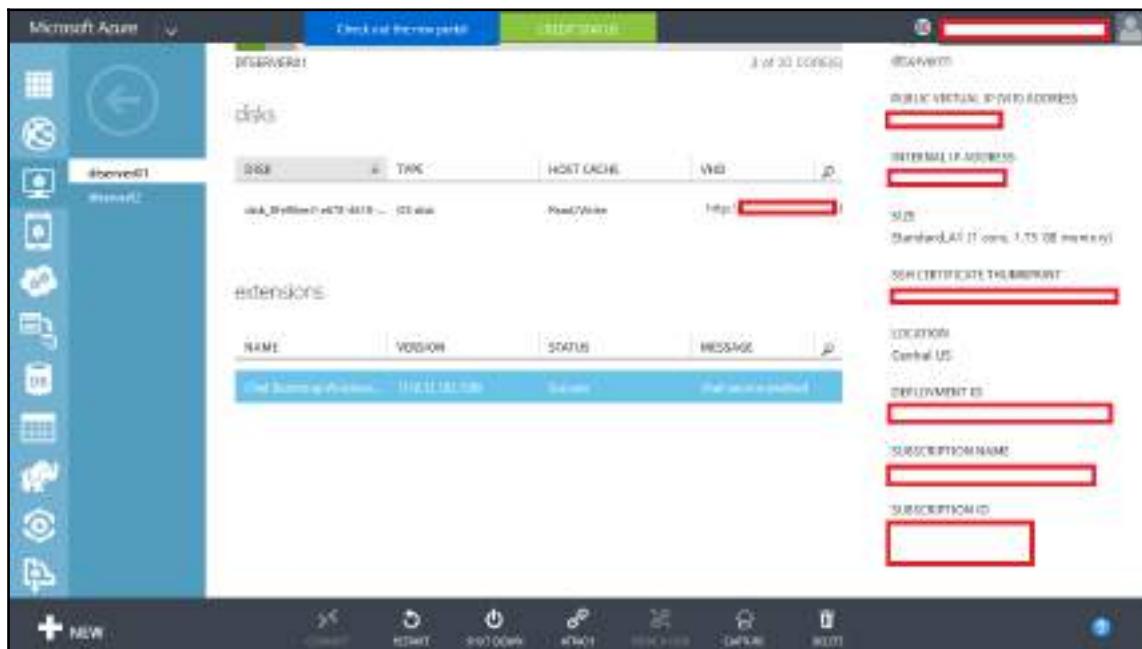
The screenshot shows the Microsoft Azure classic portal interface. On the left, there is a sidebar with various service icons: ALL ITEMS, WEB APPS, VIRTUAL MACHINES (selected), MOBILE SERVICES, CLOUD SERVICES, BATCH SERVICES, SQL DATABASES, STORAGE, HIGHLIGHT, MEDIA SERVICES, and SERVICE BUS. Below these is a '+ NEW' button. The main area is titled 'virtual machines' and contains a table with two rows of data. The columns are labeled NAME, STATUS, SUBSCRIPTION, LOCATION, and DNS NAME. The first row has a red box around the 'SUBSCRIPTION' column, which is empty. The second row has a green checkmark in the STATUS column and a red box around the 'DNS NAME' column, which shows 'dbserv01.dbs.devapp.net'. At the bottom of the table are several action buttons: RESTART, SHUTDOWN, RESTART, REATTACH, RECREATE, COMPILE, and DEPLOY.

NAME	STATUS	SUBSCRIPTION	LOCATION	DNS NAME
dbserv01	<span style="color: red;">!</span> Pending		Central US	dbserver01.dbs.devapp.net
dbserv02	<span style="color: green;">✓</span> Running		Central US	dbserv02.dbs.devapp.net

4. Click on **VIRTUAL MACHINES** in Microsoft Azure, and you'll get details about it:



5. At the bottom of the page, verify the extensions section and check whether it shows **chef-server enabled**:



Verify the Tomcat installation and virtual machine creation in VMware Workstation as an exercise, the way we did for the AWS instance.



For VMware Workstation, use <https://github.com/chipx86/knife-wsfc> for reference.

Just to reiterate, we are now close to our main objective, that is, the end-to-end automation of the application deployment pipeline. We have covered continuous integration, cloud provisioning, containers, and configuration management. What's left is the actual deployment, monitoring, and orchestration of all the activities involved in the end-to-end automation.

## Docker containers

Docker containers are extremely lightweight. We are going to use Tomcat as a web application server to deploy the PetClinic application. Docker Hub already has the Tomcat image, so we are not going to configure too many things except users for accessing the Tomcat manager app:

1. To Tomcat-users.xml, an add role and user, as follows:

```
<?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">

    <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 Docker Hub and add tomcat-users.xml to /usr/local/tomcat/conf/tomcat-users.xml. Create a Dockerfile, as shown here:

```
FROM tomcat:8.0
MAINTAINER Mitesh <mitesh.xxxx @xxxxxx.com>
COPY tomcat-users.xml /usr/local/tomcat/conf/tomcat-users.xml
```

3. Once everything is ready, use docker build 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 <YourEmailID@xyz.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
```

- The image has been successfully built. Let's verify using docker images:

```
[root@localhost mitesh]# docker images
REPOSITORY          TAG      IMAGE ID      CREATED
VIRTUAL SIZE
devopstomcatnew    latest   eb50c4ceefb5  10 seconds ago
ago                359.2 MB
devopstomcat8       latest   f3537165ebe7  10 minutes ago
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
```

- Create a container from the newly created Tomcat image. Verify existing containers using docker ps and docker ps -a:

```
[root@localhost mitesh]# docker ps
CONTAINER ID        IMAGE               COMMAND
STATUS             PORTS              NAMES
You have new mail in /var/spool/mail/root
[root@localhost mitesh]# docker ps -a
CONTAINER ID        IMAGE               COMMAND
STATUS             PORTS              NAMES
[root@localhost mitesh]# docker run -p 8180:8080 -d --name devopstomcat1 devopstomcatnew
b5f054ee4ac36d67279db10497fe7a780aecf2a72a7f52fa31ee80c618d98e4a
```

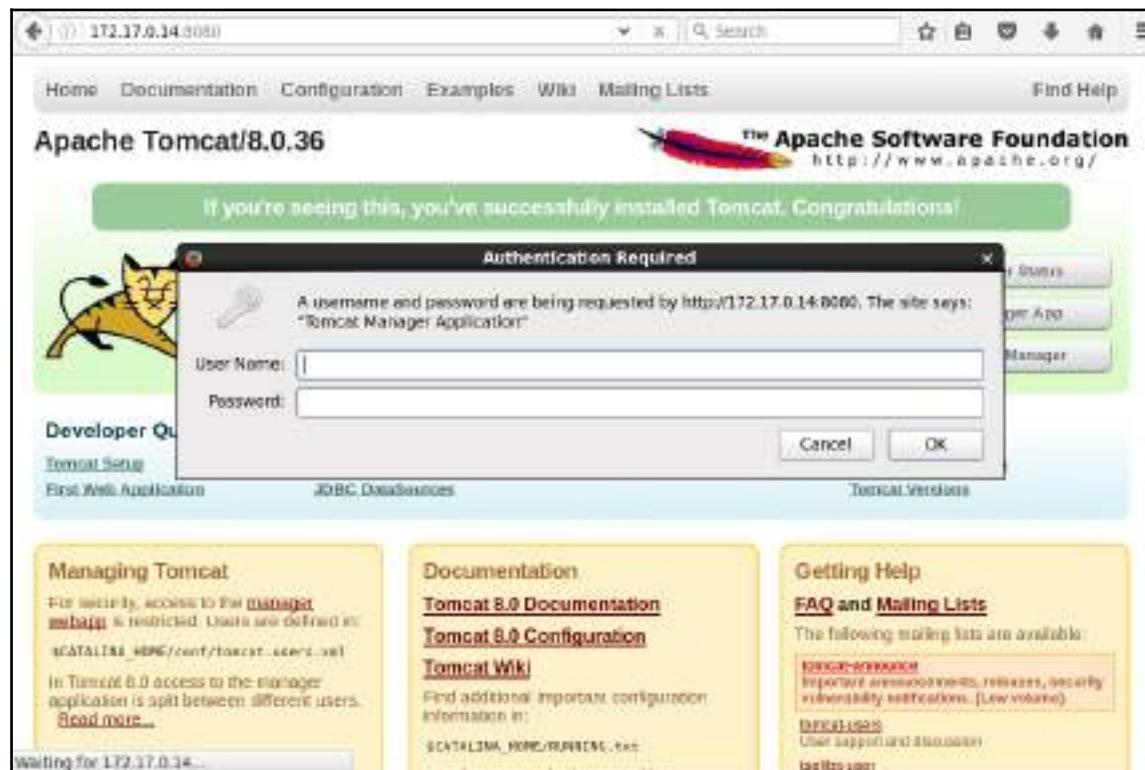
- Verify existing containers using docker ps and docker ps -a:

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

7. Use `docker inspect b5f054ee4ac3` to obtain the IP address, and browse the Tomcat web server using the IP address and port:

The screenshot shows a web browser displaying the Apache Tomcat 8.0.36 homepage. The URL in the address bar is `http://172.17.0.2:8080`. The page has a light green header with the Apache Software Foundation logo and a banner stating "If you're seeing this, you've successfully installed Tomcat. Congratulations!". Below the header, there's a yellow cat icon and a section titled "Recommended Reading" with links to "Security Considerations HOW-TO", "Manager Application HOW-TO", and "Clustering/Session Replication HOW-TO". The main content area is divided into several sections: "Developer Quick Start" (with links to "Tomcat Setup", "First Web Application", "Realm & AAA", "JDBC DataSources", "Examples", and "Server Specifications" like "Tomcat Webapps"), "Documentation" (with links to "Tomcat 8.0 Documentation", "Tomcat 8.0 Configuration", and "Tomcat Wiki"), and "Getting Help" (with links to "FAQ and Mailing Lists" and sections for "SECURITY", "ANNOUNCEMENTS", and "FORUMS").

8. Click on the **Manager App** button. It will ask for a **User Name** and **Password**. Type them in and click on **OK**:



- Now, we can access the Tomcat manager application:

The screenshot shows a web browser window with the URL `172.37.0.14:8080manager/html`. The page title is "Tomcat Web Application Manager". At the top, there's a banner for "The Apache Software Foundation" and a logo of a yellow cat. A message box says "Message: OK". Below the header is a navigation bar with tabs: "List Applications", "HTML Manager Help", "Manager Help", and "Server Status". The main content area is titled "Applications" and contains a table with three rows. The table columns are: Path, Version, Display Name, Running, Sessions, and Commands. The first row (Welcome to Tomcat) has a yellow background. The second row (Tomcat Documentation) has a green background. The third row (Servlet and JSP Examples) has a white background. Each row includes a set of buttons for Start, Stop, Reload, and Undeploy, and a link to "Expire sessions with idle ≥ 30 minutes".

Path	Version	Display Name	Running	Sessions	Commands
/	None specified	Welcome to Tomcat	true	0	<a href="#">Start</a> <a href="#">Stop</a> <a href="#">Reload</a> <a href="#">Undeploy</a> <a href="#">Expire sessions with idle ≥ 30 minutes</a>
docs	None specified	Tomcat Documentation	true	0	<a href="#">Start</a> <a href="#">Stop</a> <a href="#">Reload</a> <a href="#">Undeploy</a> <a href="#">Expire sessions with idle ≥ 30 minutes</a>
examples	None specified	Servlet and JSP Examples	true	0	<a href="#">Start</a> <a href="#">Stop</a> <a href="#">Reload</a> <a href="#">Undeploy</a> <a href="#">Expire sessions with idle ≥ 30 minutes</a>

We can use Tomcat manager application to deploy applications. Until now, we have looked at continuous integration, configuration management, containers, and cloud provisioning. Next, we will cover application deployment using different methods, monitoring, and end-to-end automation pipeline using **orchestration**.

## Self-test questions

1. Which of the following are benefits of Chef?
  - Easy policy enforcement with centralized control
  - Enables setup of consistent runtime environment
  - Enables easy restoration of environments
  - Enables disaster recovery and business continuity
  - Community-based cookbooks and recipes
  - All of these
  
2. Which two parameters are configured for Amazon EC2 credentials for `knife-ec2` in the `knife.rb` file?
  - `knife[:aws_access_key_id]` = “your AWS access key ID”
  - `knife[:aws_secret_access_key]` = “your AWS secret access key”
  - Both
  
3. Which of the following are `knife ec2` commands?
  - `knife ec2 flavor list (options)`
  - `knife ec2 server create (options)`
  - `knife ec2 server delete SERVER [SERVER] (options)`
  - `knife ec2 server list (options)`
  - All of the Above
  
4. True or false: The `rvm use` command is used to set the Ruby version.
  - True
  - False
  
5. Which of the following are `knife azure` commands?
  - `knife azure server create (options)`
  - `knife azure server delete SERVER [SERVER] (options)`
  - `knife azure server list (options)`
  - `knife azure image list (options)`
  - All of these

6. True or false: In the `knife ec2 server create` command, the `-I` parameter is used for the type of virtual machine.
  - True
  - False
  
7. True or false: In the `knife ec2 server create` command, the `-N` parameter is used for the name of the Chef Node.
  - True
  - False

## Summary

In this chapter, we covered how to provision resources in the cloud and configure them. We used the knife EC2 and knife Azure plugins to create virtual machines in AWS and Microsoft Azure, respectively. We used the Docker Hub Tomcat image to build a new image with the `tomcat-users.xml` file, which has roles and users configured to access the Tomcat manager web app.

In the next chapter, we will cover different methods to deploy an application in a Tomcat web container. Let's again reiterate the goal of the book: end-to-end automation using an 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?

It's because we want 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 to a remote Tomcat server. Once that is done, it can be used as common practice for all instances. This chapter describes in detail all the steps required to deploy our sample application to a different environment once the configuration management tool prepares it for the final deployment. We will also learn how to deploy the application in different environments, such as cloud or container-based ones.

This chapter will also cover on how to deploy an application on a PaaS model. We will deploy the application on AWS Elastic Beanstalk.

We will cover the following topics:

- Prerequisite – deploying our application on a remote server
- Deploying the application on AWS
- Deploying the application on Microsoft Azure
- Deploying the application in a Docker container

## Prerequisites – deploying our application on Remote Server

Our main objective is to deploy application in a web server. Web server and application server can be on local environment or remote environment. We will first deploy on a remote server. We will try to use Windows Agent for compilation and deployment to see how Agent-based architecture can be utilized. Follow these steps to deploy an application on a remote server:

1. First, let's start an agent on a Windows machine. Open command prompt and run the following command, given in the **Manage Nodes** section of the Jenkins dashboard. Change the URL accordingly:

```
java -jar slave.jar -jnlpUrl  
http://192.168.0.100:8080/computer/TestServer/slave-agent.jnlp -secret  
65464e02c58c85b192883f7848ad2758408220bed2f3af715c01c9b01cb72f9b  
  
.  
  
INFO: Trying protocol: JNLP2-connect  
Jul 06, 2016 8:57:16 PM hudson.remoting.jnlp.Main$CuiListener status  
INFO: Connected
```

2. Our agent is now connected to the **master**. Let's verify the status of the agent on the **master** node, where Jenkins is running:

S	Name	Architecture	Clock Difference	Free Disk Space	Free Swap Space	Free Temp Space	Response Time	
	master	Linux (amd64)	In sync	8.29 GB	1.32 GB	8.28 GB	0ms	
	TestServer	Windows 8.1 (amd64)	In sync	36.31 GB	5.16 GB	153.73 GB	2551ms	
	Data obtained	6 sec	6.9 sec	6.9 sec	5.2 sec	6.9 sec	6.9 sec	
<a href="#">Refresh status</a>								

3. Click on the **TestServer** agent, and you'll get all the details regarding projects tied to it, as shown in this screenshot:

The screenshot shows the Jenkins interface for the 'WindowsNode' slave. On the left, there's a sidebar with links: 'Back to Dashboard', 'Overview', 'Configure', and 'Load Statistics'. The main area has a title 'WindowsNode' with a pencil icon. Below it, there are two sections: 'Nodes' and 'Projects'. The 'Nodes' section shows one entry: 'TestServer'. The 'Projects' section lists three projects: 'PetClinic-Deploy' (red circle), 'PetClinic-Deploy' (blue circle), and 'PetClinic-Test' (yellow circle). Each project row includes columns for 'Last Success', 'Last Failure', and 'Last Duration'. At the bottom, there's a legend for build status: 'B8S.success' (green), 'B8S.inProgress' (orange), and 'B8S.failed/lastFailure' (red).

Now that we have the **agent** node ready, let's prepare a remote server by downloading and setting up Tomcat.

## Setting up Tomcat server

In our case, we need not to do it for cloud instances as they will be configured using Chef. The following is a more involved perspective on how we did it earlier and how all of the installation and other activities can be automated using Chef. Let's take a step-by-step tour:

1. Download Tomcat 7 from <https://tomcat.apache.org/download-7.cgi>. We are going to use the **Deploy** plugin from Jenkins, and it requires specific versions of Tomcat for deployment:

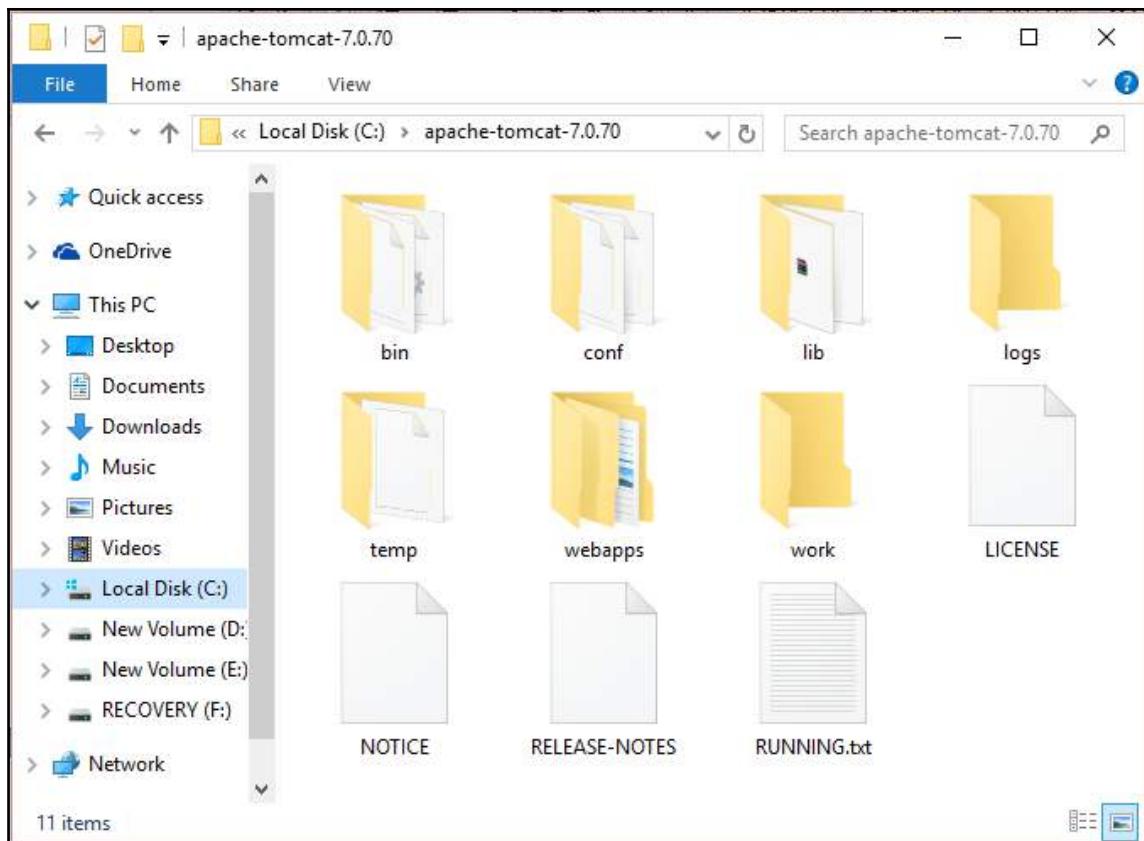
**7.0.70**

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 Tomcat:

```
C:\>cd apache-tomcat-7.0.70\bin
```

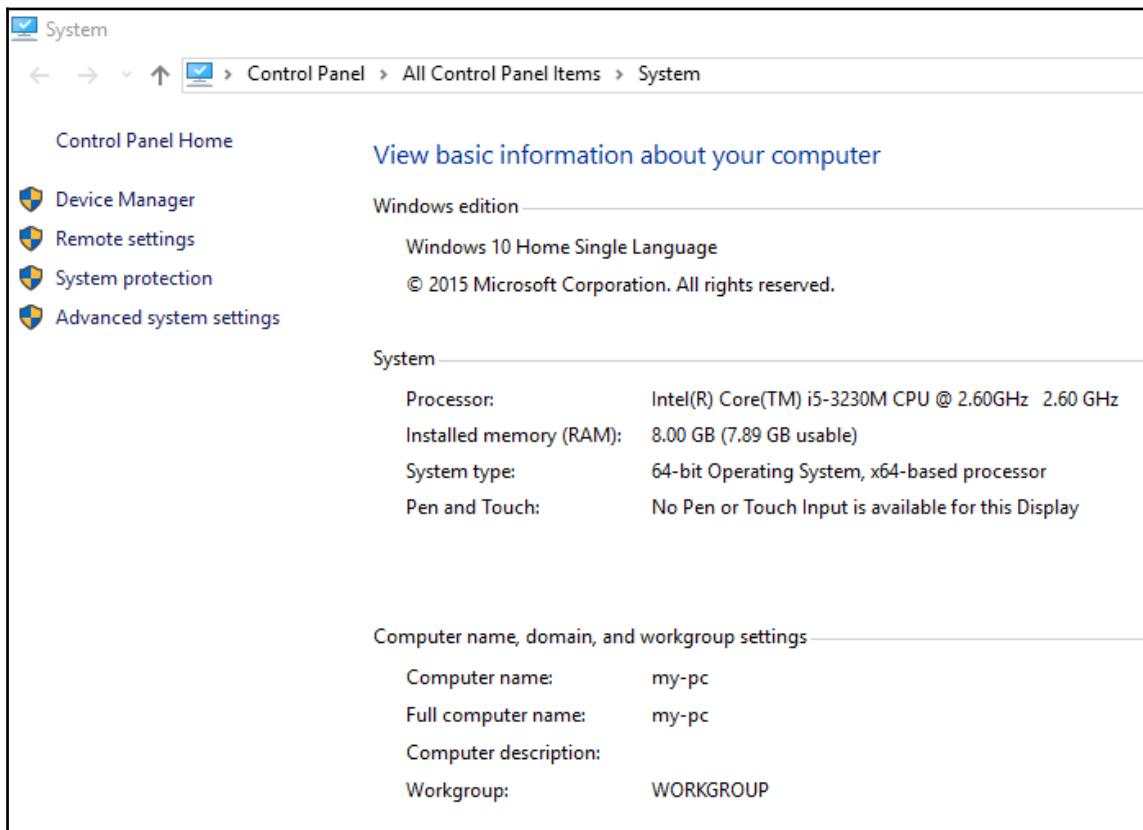
4. Run `startup.bat` from command prompt:

```
C:\apache-tomcat-7.0.70\bin>startup.bat
```

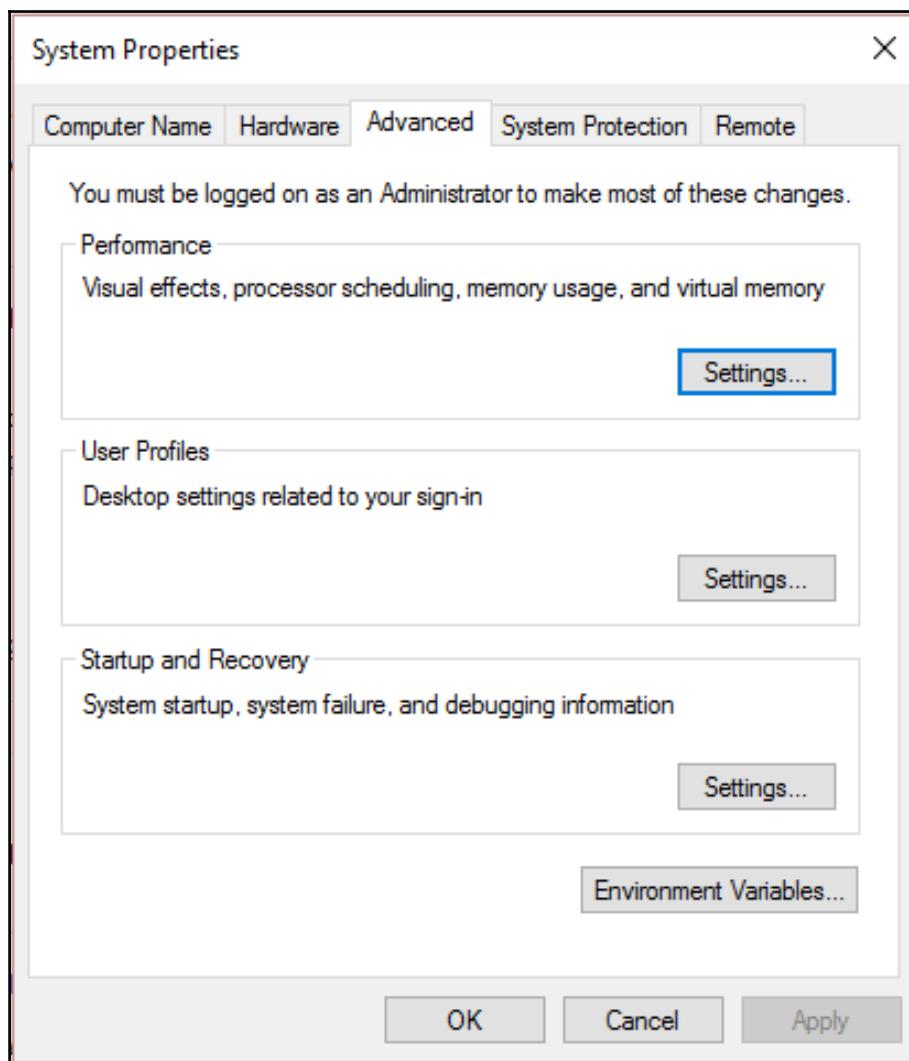
Neither the `JAVA_HOME` nor the `JRE_HOME` environment variable is defined. At least one of these environment variable is needed to run this program.

5. Oops! We need to set environment variables. Go to **Control Panel | All Control Panel Items | System**.

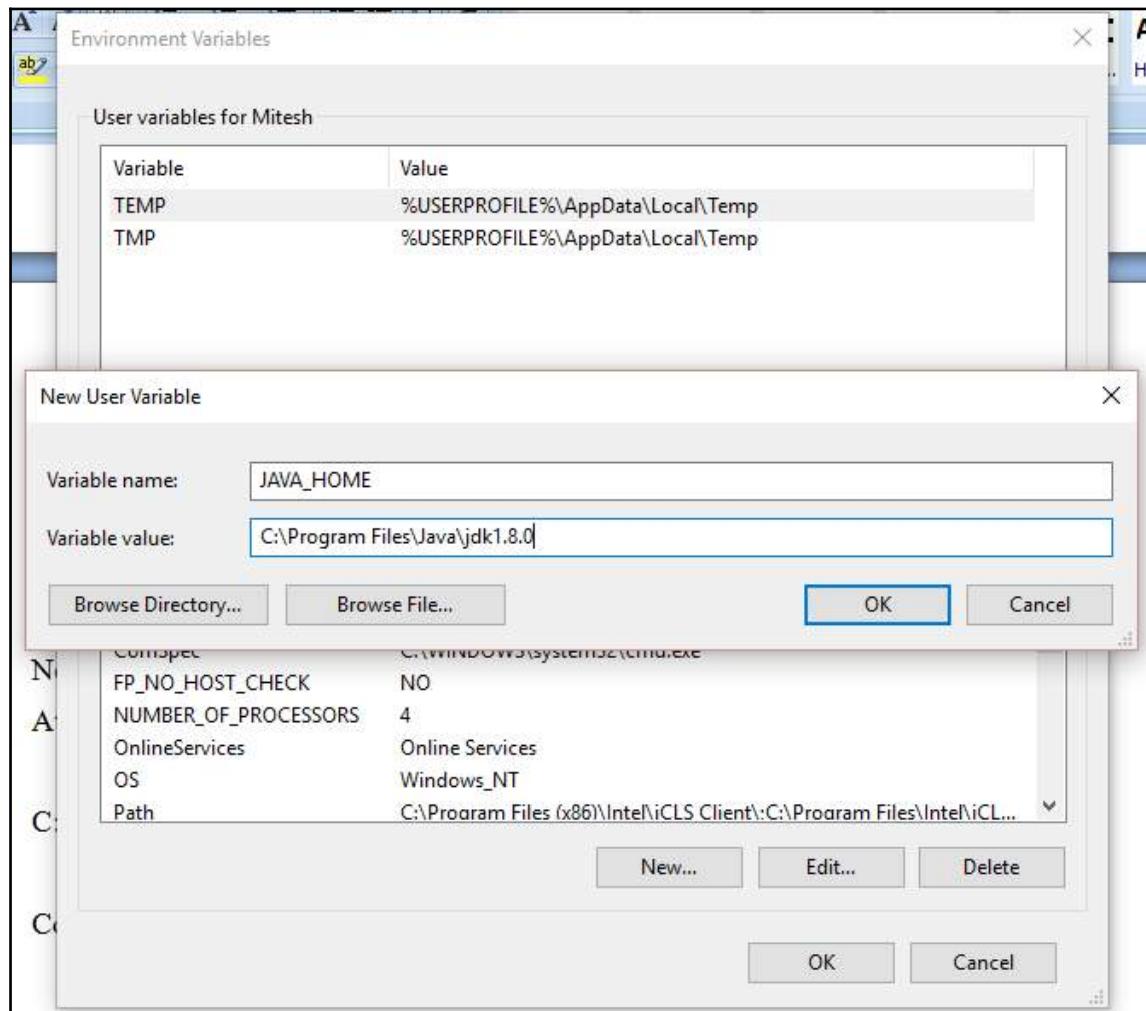
6. Click on **Advanced system settings**:



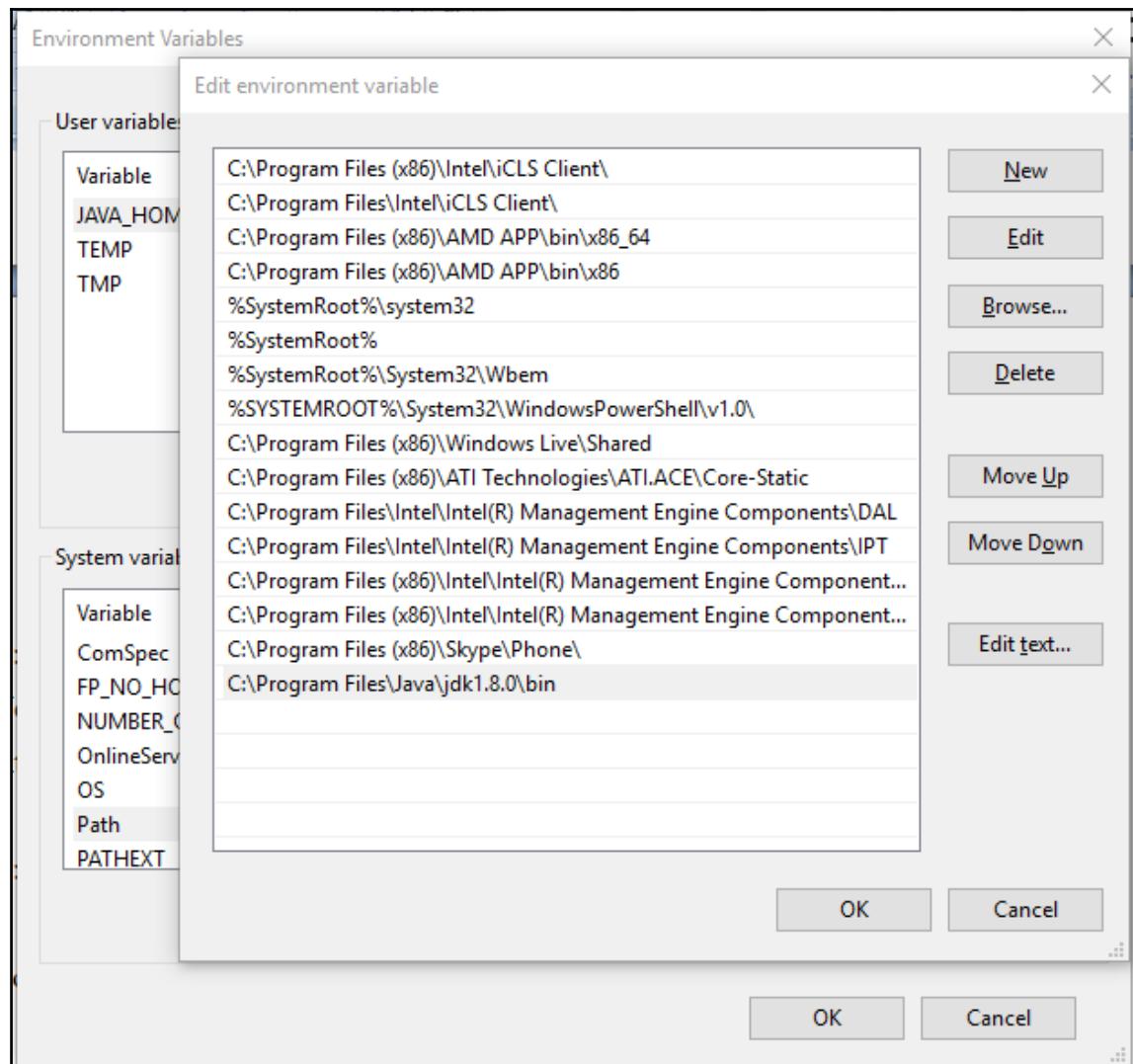
7. Click on **Environment Variables...** to set JAVA\_HOME:



8. Click on **New...** and create a new variable for **JAVA\_HOME** with the value **C:\Program Files\Java\jdk1.8.0**, and click on **OK**:



9. Click on **OK** once again to complete the process:



10. Open a new command prompt windows and verify the Java version by executing the following command:

```
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 and execute startup.bat:

```
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. Our Tomcat server is now running. It will have output similar to the following. Verify the 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 9:29:11 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 the proper IP address and port number combination to navigate to the Tomcat **Home** page, which looks like this:

The screenshot shows the Apache Tomcat 7.0.70 homepage. At the top, there's a navigation bar with links to Home, Documentation, Configuration, Examples, Wiki, and Mailing Lists. To the right is a "Find Help" search bar. Below the navigation is the Apache Software Foundation logo and link. A green banner says "If you're seeing this, you've successfully installed Tomcat. Congratulations!" On the left, there's a cartoon cat icon and a "Recommended Reading" section with links to Security Considerations HOW-TO, Manager Application HOW-TO, and Clustering/Session Replication HOW-TO. On the right, there are buttons for Server Status, Manager App, and Host Manager. The main content area is divided into three columns: "Developer Quick Start" (with links to Tomcat Setup, Best Web Applications, Roles & AAA, and JMX Dashboards), "Documentation" (with links to Tomcat 7.0 Documentation, Tomcat 7.0 Configuration, and Tomcat WIKI), and "Getting Help" (with links to FAQ and Mailing Lists, and descriptions of various mailing lists: tomcat-announce, tomcat-user, tomcat-dev, tomcat-jira, and tomcat-servlets).

14. Go to `conf` and then open `tomcat-users.xml` in your Tomcat installation directory and *uncomment* the `role` and `user` lines or rewrite them. Set `manager-gui` as the `rolename` for testing purposes. We need `manager-script` for deployment via the Deploy plugin:

```
<?xml version="1.0" encoding="UTF-8"?>
<!--
  $Id: tomcat-users.xml,v 1.12 2008-02-12 12:03:47 mrglavas Exp $
-->
<tomcat-users>
  <!--
    NOTE: The sample user and role entries below are intended for use with the
    example 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
    example 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="admin123" roles="manager-script"/>
</tomcat-users>
```

15. Click on the manager application link on the Tomcat **Home** page and enter the username and password you set in `tomcat-users.xml`. Now, we can access the management application:

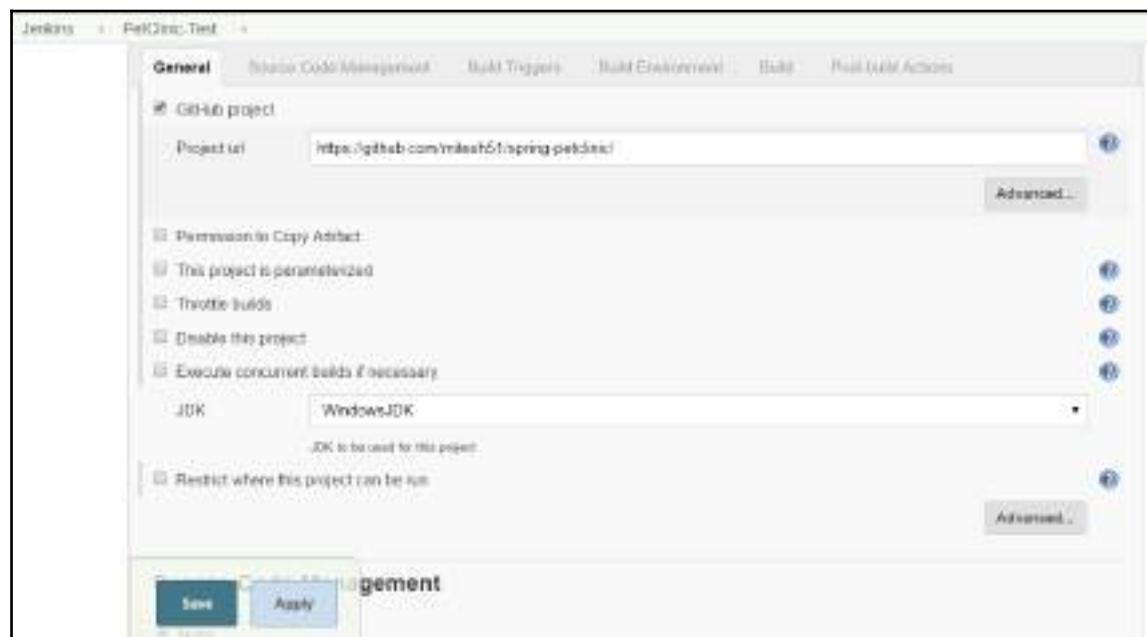
The screenshot shows a web browser window with the URL `localhost:8080/manager/text`. The title bar says "Tomcat Web Application Manager". The page has a header with the Apache Software Foundation logo and a yellow cat icon. Below the header is a navigation bar with tabs: "Manager", "List Applications", "HTML Manager Help", "Manager Help", and "Server Status". The main content area is titled "Applications" and contains a table with the following data:

Path	Version	Display Name	Running	Sessions	Commands
/	None specified	Welcome to Tomcat!	true	0	<a href="#">Start</a> <a href="#">Stop</a> <a href="#">Reload</a> <a href="#">Undeploy</a> <a href="#">Expire sessions with idle &gt; 30 minutes</a>
docs	None specified	Tomcat Documentation	true	0	<a href="#">Start</a> <a href="#">Stop</a> <a href="#">Reload</a> <a href="#">Undeploy</a> <a href="#">Expire sessions with idle &gt; 30 minutes</a>
examples	None specified	GuruB and JSP Examples	true	0	<a href="#">Start</a> <a href="#">Stop</a> <a href="#">Reload</a> <a href="#">Undeploy</a> <a href="#">Expire sessions with idle &gt; 30 minutes</a>

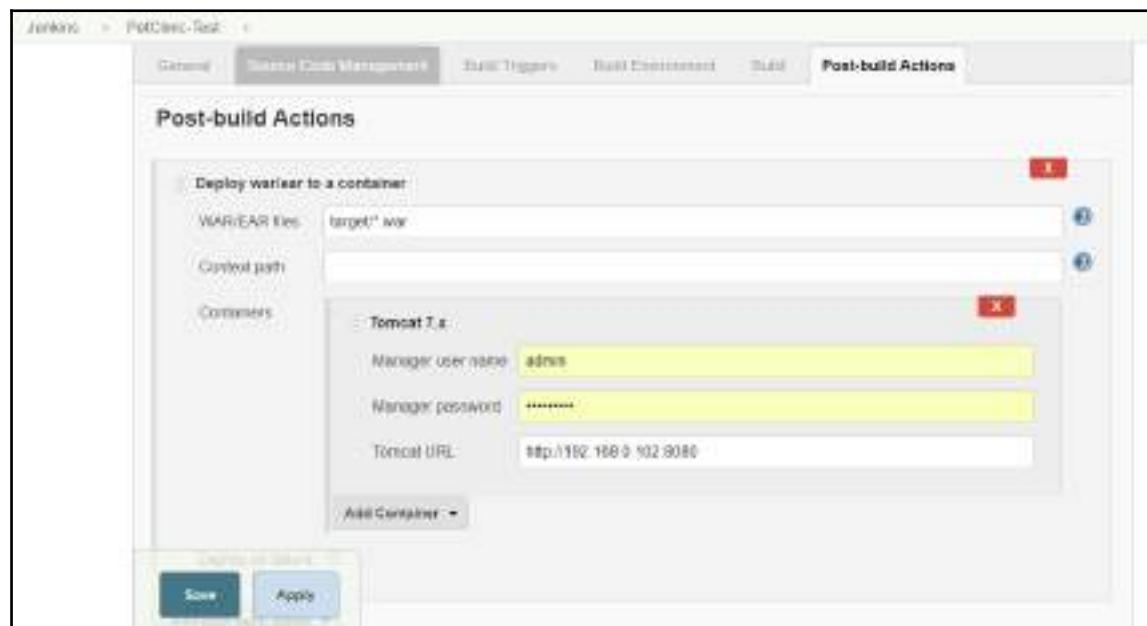
16. For the Jenkins Deploy plugin, change the rolename to `manager-script`.
17. Restart Tomcat and visit `http://<IP Address>:8080/manager/text/list`. You should see this output:

```
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 the Jenkins job build page and click on **Configure**. Select the proper JDK configuration for the Jenkins agent:



19. Under **Post-build Actions**, select **Deploy war/ear to a container**. Provide the location of the WAR file in the Jenkins workspace, the Tomcat manager credentials, and the **Tomcat URL** with the port:



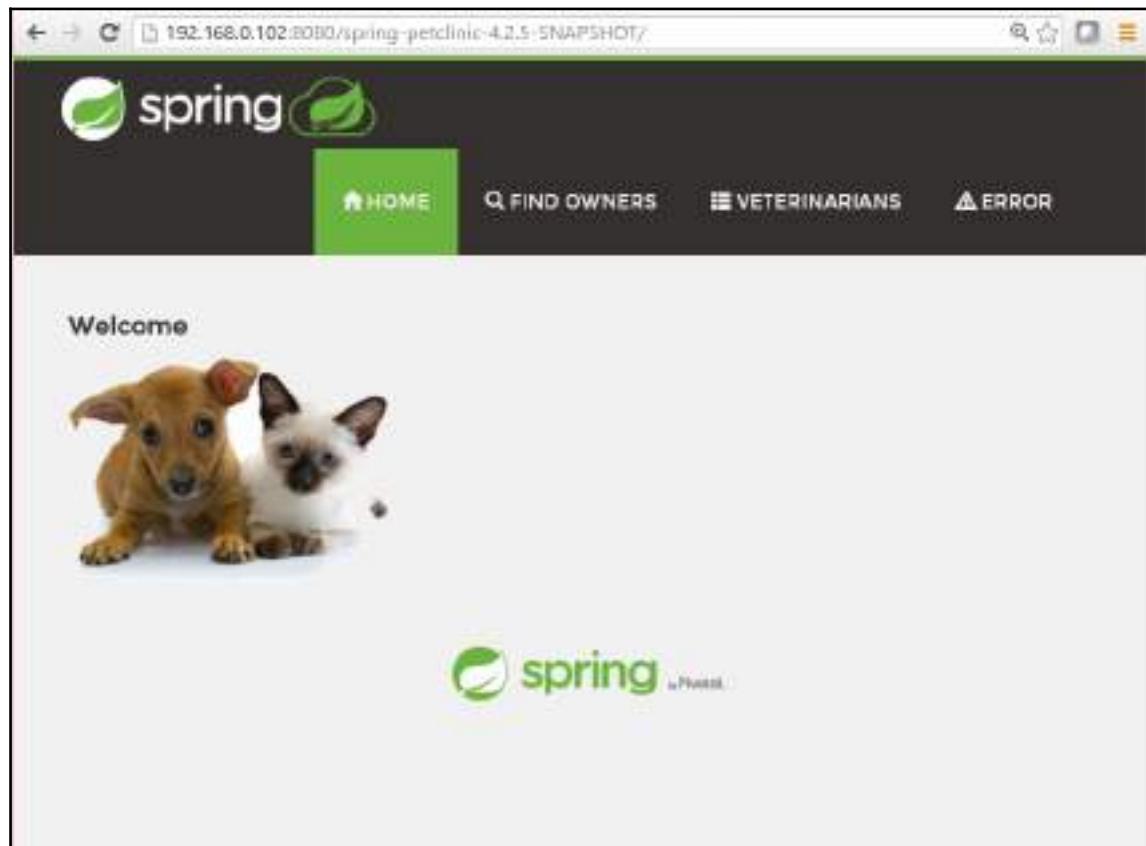
20. Click on **Apply** and **Save**. Click on **Build now** on the Jenkins build's page. Verify the console output as showing a fresh deployment:

```
Results : 

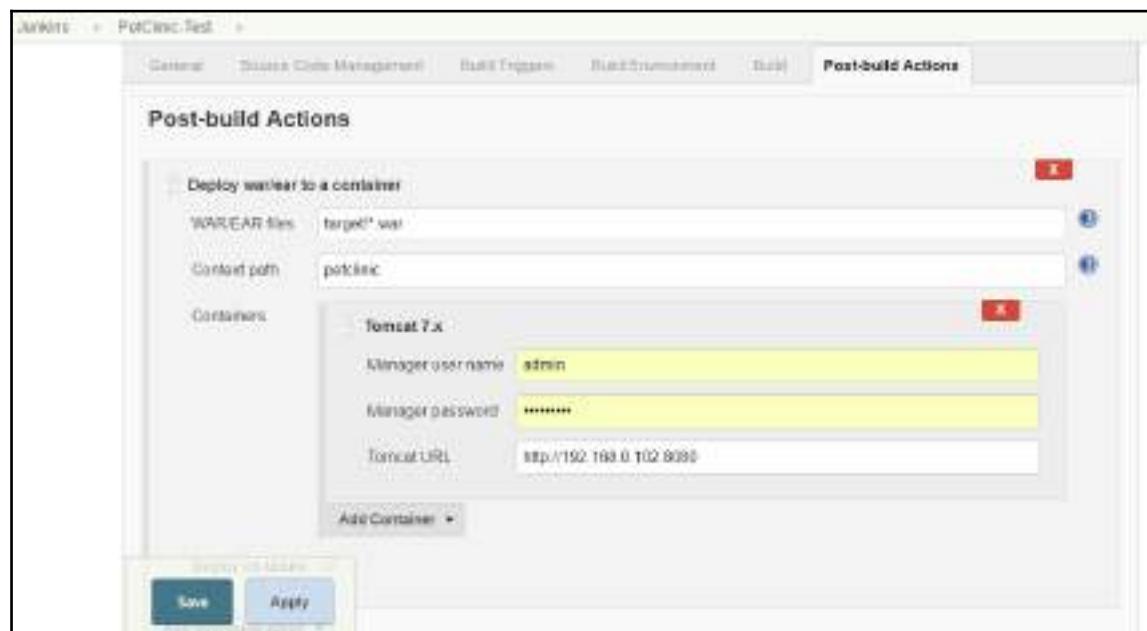
Tests run: 59, Failures: 0, Errors: 0, Skipped: 0

[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 [1659 ms]
[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 the build is successful, visit the URL from your browser and notice the context. It is similar to name of the application:



22. In **Post-build Actions**, provide a **Context path** and click on **Save**. Click on **Build now** again:



23. Verify the application URL by providing a new **Context path**.



For deployments where we can access the `tomcat-users.xml` file in cases where we use Tomcat as the application container, we will use the same method for deployment. If we don't have direct access to the Tomcat directory or can't change `tomcat-users.xml`, another approach can be to SSH the remote host and copy the file into the remote host's `webapps` file in the 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 containers in Chapter 5, *Installing and Configuring Docker*. To deploy an application with the Deploy plugin of Jenkins, we will modify `tomcat-users.xml`. Let's take it step by step:

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 the Dockerfile, we will copy `tomcat-users.xml` to the `/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 the 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 using the docker images command:

```
[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
```

5. Execute docker run 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 the dockers command:

```
[root@localhostmitesh]#dockers
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
```

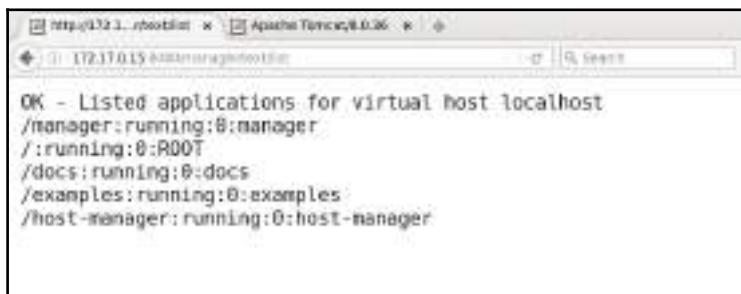
7. Use docker inspect 771bb7cb809d<container ID> to get an IP address.

8. Stop the iptables service for verification or opening ports in IP tables:

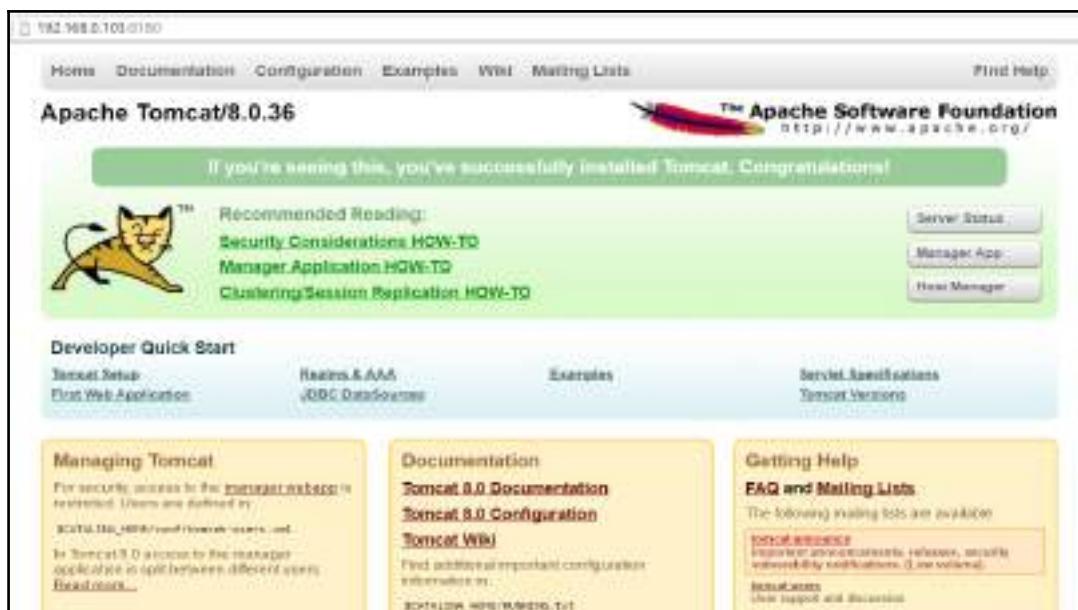
```
[root@localhostmitesh]# service iptables stop
iptables: Setting chains to policy ACCEPT: nat filter      [  OK  ]
```

```
iptables: Flushing firewall rules: [ OK ]
iptables: Unloading modules: [ OK ]
You have new mail in /var/spool/mail/root
```

9. Use the IP address to access the **Manager App** URL. Verify whether it is successful:



10. As we have mapped the port, use the host's IP address and verify your Tomcat installation:



11. Use the IP address of the host, and access the **Manager App URL**. Provide a **User Name** and **Password**:

The screenshot shows a web browser window with the URL `192.168.0.103:8080/manager/text/list`. The title bar says "Apache Tomcat/8.0.36". The main content area displays the "Developer Quick Start" section with links to "Tomcat Setup", "First Web Application", "Realm & AAA", "JDBC Datasources", "Examples", and "Server Specifications/Tomcat Versions". Below this, there are three yellow boxes: "Managing Tomcat" (with notes about manager vs admin users), "Documentation" (links to Tomcat 8.0 Documentation, Configuration, and Wiki), and "Getting Help" (links to FAQ and Mailing Lists, with sections for "SUBSCRIPTIONS" and "FORUMS"). A central modal dialog box titled "Authentication Required" asks for a "User name" (set to "admin") and "Password". To the right of the modal, a banner from the "Apache Software Foundation" with the URL `http://www.apache.org/` is displayed.

12. Check whether it is successful:

The screenshot shows a terminal window with the command `192.168.0.103:8080/manager/text/list` entered. The output shows the following listed applications for the virtual host localhost:

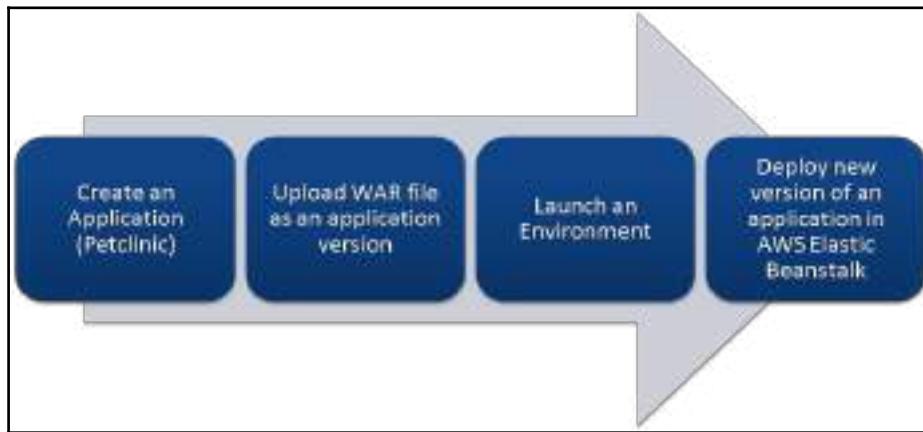
```
OK - Listed applications for virtual host localhost
/manager:running:0:manager
/:running:0:ROOT
/docs:running:0:docs
/examples:running:0:examples
/host-manager:running:0:host-manager
```

13. Once everything is working fine, use the Deploy plugin to deploy an application in a Docker container.

# Deploying Application in AWS

AWS Elastic Beanstalk is a **Platform as a Service (PaaS)** offering from Amazon. We will use it to deploy the 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.

These are the steps to deploy an application on AWS Elastic Beanstalk:



Elastic Beanstalk supports the following programming languages and platforms:

Programming Languages	Platforms
<ul style="list-style-type: none"><li>Java, PHP, Python, Ruby, Go</li></ul>	<ul style="list-style-type: none"><li>Go, Java SE, Java with Tomcat, .NET on Windows Server with IIS, Node.js, PHP, Python, Ruby</li></ul>

Let's create a sample application to understand how Elastic Beanstalk works and then use the Jenkins plugin to deploy an application:

1. Go to the AWS management console and verify whether we have a default **Virtual Private Cloud (VPC)**. If you've deleted the default VPC and subnet by accident, send a request to AWS customer support to recreate it:

The screenshot shows the AWS VPC Dashboard. The left sidebar lists various VPC components: Your VPCs, Subnets, Route Tables, Internet Gateways, DHCP Options Sets, Elastic IPs, Endpoints, NAT Gateways, Peering Connections, Security, and Network ACLs. The main content area is titled "Resources" and includes a "Start VPC Wizard" button and a "Launch EC2 Instances" button. A note states: "Note: Your Instances will launch in the US East (N. Virginia) region." Below this, it says: "You are using the following Amazon VPC resources in the US East (N. Virginia) region:" followed by a table of resource counts.

Resource Type	Count
1 VPC	1 Internet Gateway
4 Subnets	1 Route Table
1 Network ACL	0 Elastic IPs
0 VPC Peering Connections	0 Endpoints
0 Nat Gateways	1 Security Group
0 Running Instances	0 VPN Connections
0 Virtual Private Gateways	0 Customer Gateways

## VPN Connections

Amazon VPC enables you to use your own isolated resources within the AWS cloud, and then connect those resources directly to your own datacenter using industry-standard encrypted IPsec VPN connections.

2. Click on **Services** in the AWS management console and select **AWS Elastic Beanstalk**. Create a new application named **petclinic**. Select **Tomcat** as a **Platform** and select the **Sample application** radio button:

The screenshot shows the 'Create a web app' wizard in the AWS Elastic Beanstalk console. The application name is set to 'petclinic'. The platform is selected as 'Tomcat'. The 'App code' section is set to 'Sample application', which is described as a pre-built application with sample code. There is also an option to 'Upload your own code'. A note at the bottom indicates that the app code will be uploaded after initial setup.

Application name: petclinic  
Platform: Tomcat  
App code:  Sample application  
Creates with pre-built sample code to get you started quickly. You can upload your own source code for this application.  
 Upload your own code  
You can upload a file or provide a URL to your app code in Amazon S3.

3. Verify the sequence of events for the creation of a sample application:

The screenshot shows the 'Creating petclinic' status page in the AWS Elastic Beanstalk console. It displays a log of events during the deployment process. The log includes messages about launching EC2 instances, creating a security group, and setting up an S3 bucket. The status is currently 'Creating environment'.

Creating petclinic  
This will take a few minutes.

11:04am Waiting for EC2 instances to launch. This may take a few minutes.  
11:04am Created EC2-12-79-142-147  
11:04am Environment health has transitioned to Pending. Initialization in progress (waiting for 8 seconds). There are no instances.  
11:05am Created security group named: `aws-ebs-vmwvcn-elastic-aws-ELB-Security-Group-61678313450`  
11:05am Using elastic-beanstalk-us-west-1-68028087657 as Amazon S3 storage bucket for www content data.  
11:05am createEnvironment is starting.

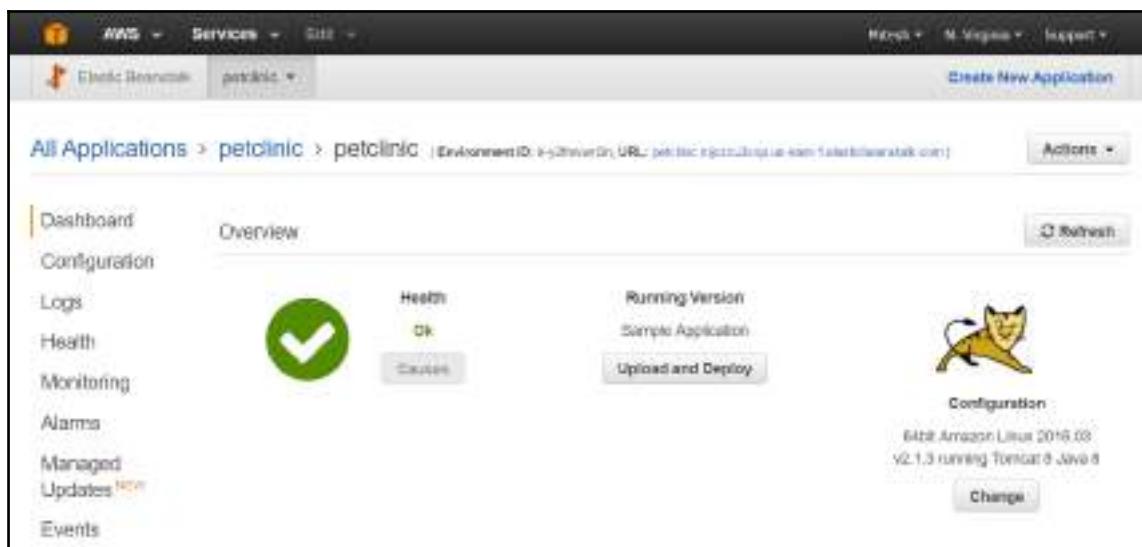
Learn More  
[Get started using Elastic Beanstalk](#)  
[Modify the code](#)  
[Create and connect to a database](#)  
[Add a custom domain](#)

Command Line Interface (v3)  
[Installing the AWS EB CLI](#)  
[EB CLI Command Reference](#)

4. It will take a while, and once the environment has been created, it will be highlighted in green, as shown here:



5. Click on the **petclinic** environment and verify the **Health** and **Running Version** in the dashboard:



6. Verify the environment ID and URL. Click on the URL and verify the default page:

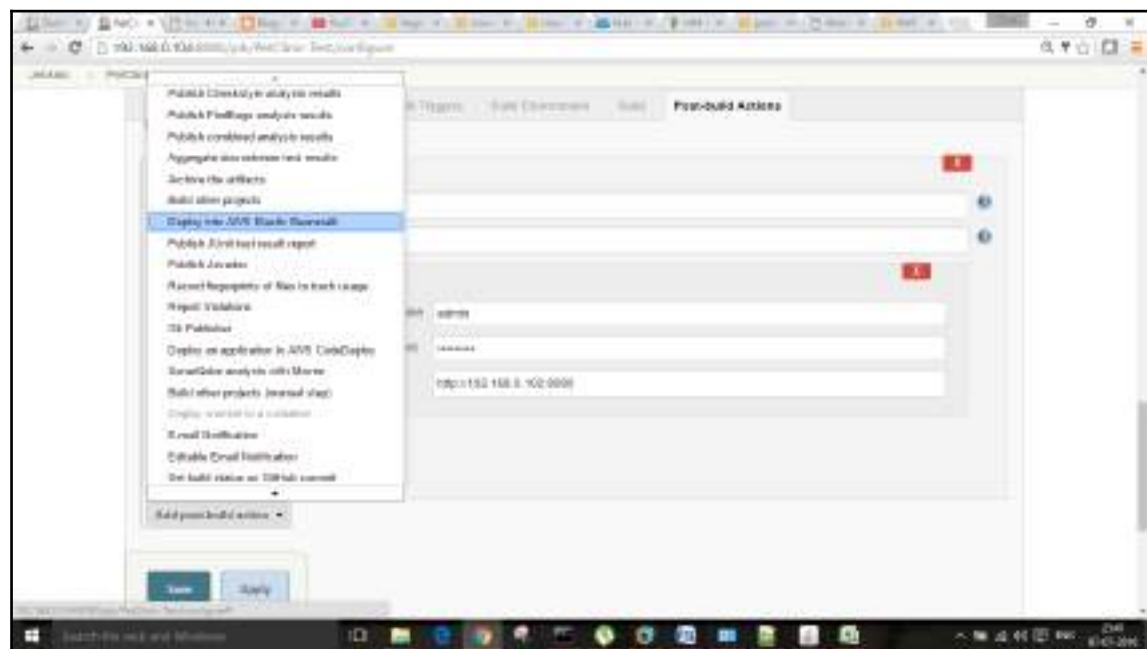


7. Install **AWS Elastic Beanstalk Publisher** plugin.

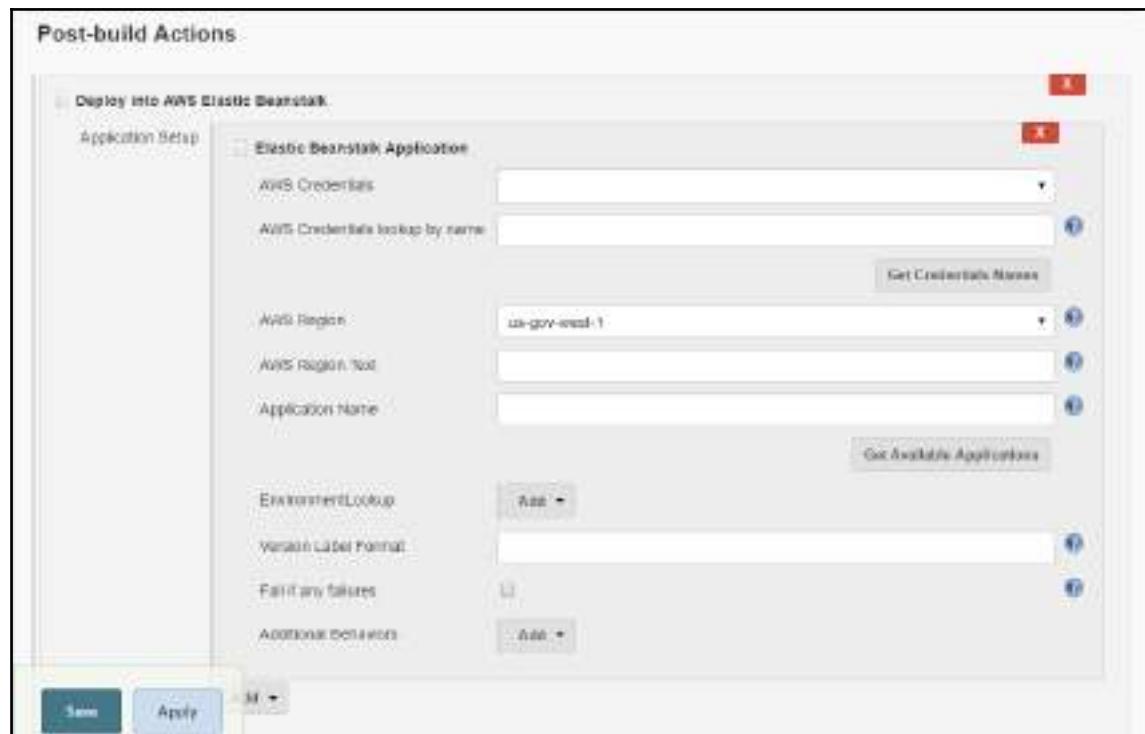


For more details, visit <https://wiki.jenkins-ci.org/display/JENKINS/AWS+Beanstalk+Publisher+Plugin>.

8. Open the Jenkins dashboard and go to **Build job**. Click on **Post-build Actions** and select **Deploy into AWS Elastic Beanstalk**.



9. A new section will come up in **Post-build Actions** for Elastic Beanstalk:



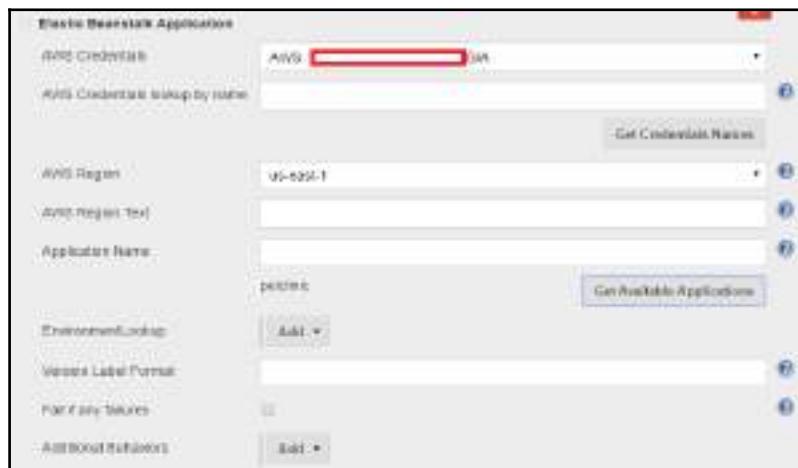
10. Click on the Jenkins dashboard and select **Credentials**; add your AWS credentials:



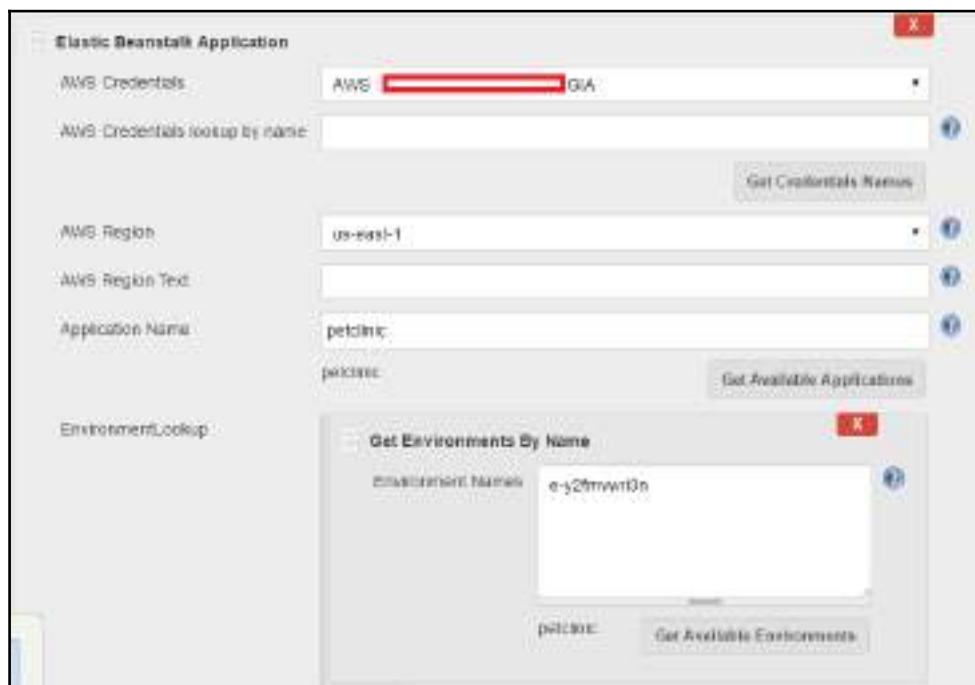
11. Go to your Jenkins build and select an **AWS Credential** that is set in the global configuration:



12. Select **AWS Region** from the list and click on **Get Available Applications**. As we have created a sample application, it will show up like this:



13. In **EnvironmentLookup**, provide an environment ID in the **Get Environments By Name** box and click on **Get Available Environments**:



14. Save the configuration and click on **Build now**.

Let's verify the AWS management console whether WAR file is being copied in Amazon S3 or not:

1. Go to **S3 Services** and check the available buckets:



2. Verify the build job's execution status in Jenkins. Some sections of the expected output follow:
3. Since the WAR file is large, it will take a while to upload to **Amazon S3**. Once it is uploaded, it will be available in the Amazon S3 bucket.
4. The test case execution and WAR file creation are 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 msec]
[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] -----
```

---

5. The execution of **AWSEB Deployment** plugin – post build action has been 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
```

6. The deployment activity with the new version label starts:

```
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'
```

7. The environment and health statuses are updated along with the 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.zip
Finished: SUCCESS
```

8. The build is successful. Now, check the AWS management console:

The screenshot shows the AWS S3 console interface. At the top, there are navigation links for 'AWS', 'Services', 'Edit', and dropdown menus for 'Region', 'Global', and 'Switch Region'. Below the header, there are buttons for 'Upload', 'Create Folder', and 'Actions'. A search bar with the placeholder 'Search by prefix' is followed by three tabs: 'None', 'Properties', and 'Transfers'. The main area displays a table titled 'All Buckets / elasticbeanstalk-us-east-1-685218287657'. The table has columns for 'Name', 'Storage Class', 'Size', and 'Last Modified'. It lists three objects: 'elasticbeanstak...', 'petclinic-jenkins-PetClinic-Test-39.zip', and 'readme.txt'. The 'elasticbeanstak...' object is a folder icon, 'petclinic-jenkins-PetClinic-Test-39.zip' is a file icon, and 'readme.txt' is a text file icon.

Name	Storage Class	Size	Last Modified
elasticbeanstak...	Standard	0 bytes	Thu Jul 07 01:03:15 GMT+05:30 2016
petclinic-jenkins-PetClinic-Test-39.zip	Standard	56.9 MB	Fri Jul 08 01:03:04 GMT+05:30 2016
readme.txt	-	-	-

9. Go to **Services**, click on **AWS Elastic Beanstalk**, and verify the environment. The previous version was **Sample Application**. Now, the version is updated as given in **Version Label Format** in the Jenkins build job configuration:

The screenshot shows the AWS Elastic Beanstalk console. In the top navigation bar, 'AWS' is selected, followed by 'Services', 'Edit', and 'Hitesh - N. Virginia'. The 'Elastic Beanstalk' icon is highlighted, and the application name 'petclinic' is shown. A 'Create New Application' button is visible. On the left, there's a sidebar with links like 'Learn More', 'Get started using Elastic Beanstalk', 'Modify the code', 'Create and connect to a database', 'Add a custom domain', 'Command Line Interface (v3)', 'Installing the AWS EB CLI', and 'EB CLI Command Reference'. The main area is titled 'All Applications' and shows 'petclinic' with a green background. Below it, the environment details are listed: Environment type: Web Server, Running versions: Jenkins-PetClinic-Test-39, Last modified: 2016-07-08 01:04:41 UTC+0530, and URL: petclinic.uptempo.us-east-1.amazonaws.com.

10. Go to the dashboard and verify **Health** and **Running Version** again:

The screenshot shows the AWS Elastic Beanstalk application dashboard for 'petclinic'. The top navigation bar includes 'AWS', 'Services', 'Edit', 'Hitesh - N. Virginia', and 'Report'. The 'Elastic Beanstalk' icon is highlighted, and the application name 'petclinic' is shown. A 'Create New Application' button is present. The main area shows the 'petclinic' application with a green background. On the left, a sidebar lists 'Dashboard', 'Configuration', 'Logs', 'Health', 'Monitoring', 'Alarms', 'Managed Updates', and 'Events'. The 'Overview' tab is selected. In the center, there's a large green circle with a white checkmark, labeled 'Health' and 'OK'. Below it is a 'Changes' button. To the right, the 'Running Version' is listed as 'jenkins-PetClinic-Test-39' with a 'Upload and Deploy' button. Further right is a yellow cat icon and the 'Configuration' section, which includes '64-bit Amazon Linux 2016.09 v2.1.3 running Tomcat 8 Java 8' and a 'Change' button.

11. Click on the **Configuration** link on the Elastic Beanstalk dashboard and verify **Scaling, Instances, Notifications, Software Configuration, Updates and Deployments, Health**, and so on:

The screenshot shows the AWS Elastic Beanstalk Configuration dashboard under the Web Tier tab. On the left, there's a sidebar with links: Dashboard, Configuration (which is selected), Logs, Health, Monitoring, Alarms, Managed Updates, Events, and Tags. The main area is divided into several sections: Scaling (Environment type: Single instance, Custom Availability Zones: Any), Instances (Instance type: t1.micro, Availability Zones: Any), Notifications (Notifications: 0), Software Configuration (Log publication: Off, Initial JVM heap size: 256m, JVM command-line options: -Djava.net.preferIPv4Stack=true, Maximum JVM heap size: 256m, Maximum JVM permanent generation size: 64m), Updates and Deployments (Rolling updates are disabled), and Health (Application health check URL: http://petclinic.us-west-1.elasticbeanstalk.com, Health reporting: Enhanced).

12. Click on **Logs** to download the log files for Elastic Beanstalk:

The screenshot shows the AWS Elastic Beanstalk Logs section for the petclinic application. The sidebar on the left has links: Dashboard, Configuration, Logs (which is selected), Health, Monitoring, Alarms, Managed Updates, and Events. The main area shows a table with one row: Log file: Download, Time: 2016-07-08 01:15:33 UTC+0600, EC2 instance: i-0761fed0979c2fa02, and Type: Last 100 Lines. There are also buttons for Request Logs and Refresh.

13. Go to the Enhanced Health Overview and check the Status:

The screenshot shows the AWS CloudWatch Applications interface. On the left, a sidebar lists navigation options: Dashboard, Configuration, Logs, **Health**, Monitoring, Alarms, Managed Updates (10), and Events. The main content area is titled "Enhanced Health Overview" for the "petclinic" application. It includes a table with columns for Instance ID, Status, Running, Dep. ID, Rate, 2xx, 3xx, 4xx, 5xx, P99, P90, and P75. A status bar at the bottom indicates "Overall: Ok". Below the table, there are buttons for Total (1), OK (1), Pending (0), Info (0), Unknown (0), No data (0), Warning (0), Degraded (0), and Severe (0). The URL shown is <https://console.aws.amazon.com/cloudwatch/applications/home?region=us-east-1&applicationId=petclinic>.

14. Click on **Monitoring** for extensive monitoring details in the form of CPU utilization and health of an application:

The screenshot shows the AWS CloudWatch Applications interface, specifically the Monitoring section for the "petclinic" application. The sidebar on the left is identical to the previous screenshot. The main area displays monitoring metrics: CPU Utilization (12.5%), Max Network In (42MB), and Max Network Out (556KB). Below these metrics is a "Monitoring" section with two line charts. The first chart, "Environment Health by health status", shows a single spike from "Ok" to "Degraded" around 10:00. The second chart, "CPU Utilization - 1 minute", shows a fluctuating line graph with several peaks, notably around 10:00, 14:00, and 18:00. The URL shown is <https://console.aws.amazon.com/cloudwatch/applications/home?region=us-east-1&applicationId=petclinic>.

15. Click on **Events** to get list of all the events of the Elastic Beanstalk application lifecycle:

Events			
	Severity	Date	Type
Logs	TRACE	2016-05-20 01:10:58 UTC+0530	Paged log for environment
Health	INFO	2016-07-08 01:10:58 UTC+0530	Environment: >E757fa3007fb27a02 Successfully flushed tailing T3 log(s)
Monitoring	INFO	2016-07-08 01:10:58 UTC+0530	requestsPerSecondRate is 0.0000
Alarms	INFO	2016-07-08 01:10:58 UTC+0530	Environment health has transitioned from Info to Ok. Application update completed 45 seconds ago and took 19 seconds.
Managed Updates <span style="color: #f08080;">14+</span>	INFO	2016-07-08 01:10:58 UTC+0530	Environment update completed successfully
Events	INFO	2016-07-08 01:10:40 UTC+0530	New application version was deployed to running EC2 instances
Tags	INFO	2016-07-08 01:10:42 UTC+0530	Environment health has transitioned from Ok to Info. Application update is progress (running for 14 seconds)
	INFO	2016-07-08 01:10:59 UTC+0530	Deploying new version to instance(s).
	INFO	2016-07-08 01:10:59 UTC+0530	Environment update is starting.

16. Once everything has been verified, click on the URL for the environment, and our PetClinic application is live:



- Once the application deployment is successful, terminate the environment:

The screenshot shows the AWS Elastic Beanstalk console. In the top navigation bar, 'AWS' is selected under 'Services'. The region is set to 'US East (N. Virginia)'. A 'Create New Application' button is visible in the top right. The main page displays the 'petclinic' application under 'All Applications'. A prominent message box says 'Basic Beanstalk is terminating your environment.' with a 'View Events' link. On the left, a sidebar menu includes 'Dashboard', 'Configuration', 'Logs', 'Health', 'Monitoring', 'Alarms', 'Managed Updates', 'Events', and 'Tags'. The 'Health' section shows a green icon with 'Info' and 'caused by' status. The 'Running Version' is listed as 'petclinic-PetClinic-1.0.0'. To the right, there's a yellow cat icon representing the application logo, and detailed configuration information: '64bit Amazon Linux 2010.03 v2.1.3 running Tomcat 8 Java 8'.

We have thus successfully deployed our application on Elastic Beanstalk.

## Deploying application in Microsoft Azure

Microsoft Azure app services is a PaaS. In this section, we will look at the Azure web app and how we can deploy our PetClinic application:

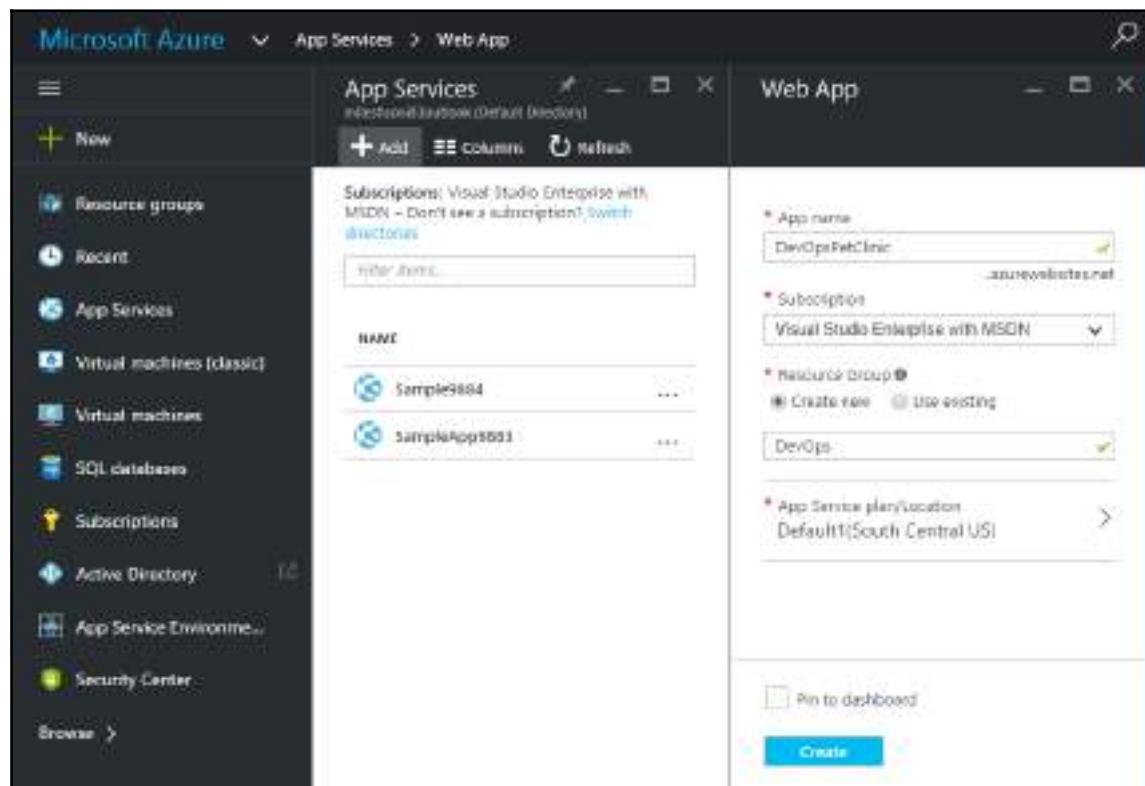
- Let's install the **Publish Over FTP** plugin in Jenkins. We will use the Azure web app's FTP details to publish the PetClinic WAR file:

The screenshot shows the Jenkins plugin manager interface. The 'Available' tab is selected. Under the 'Install' section, the 'FTP publisher plugin' is listed with a brief description: 'This plugin can be used to upload project artifacts and whole directories to an ftp server'. Below it, the 'Publish Over FTP' plugin is listed with its version 1.12. Further down, the 'Publish Over SSH' and 'SSH2 Easy Plugin' are listed with their respective versions 1.14 and 1.4.

- Once the plugin has been installed successfully, restart Jenkins:



- Go to Microsoft Azure portal at <https://portal.azure.com>. Click on **App Services** and then on **Add**. Provide values for **App Name**, **Subscription**, **Resource Group**, and **App Service plan/Location**. Click on **Create**:



- Once the Azure web app is created, see whether it shows up in Azure portal:

The screenshot shows the Microsoft Azure portal interface. On the left, there is a sidebar with various service icons: Resource groups, Recent, App Services (selected), Virtual machines (classic), Virtual machines, SQL databases, Subscriptions, Active Directory, App Service Environments, and Security Center. The main content area is titled "App Services" and shows a table of existing web apps. The table has columns for NAME, STATUS, APP TYPE, APP SERVICE PLAN, and LOCATION. Three entries are listed:

NAME	STATUS	APP TYPE	APP SERVICE PLAN	LOCATION
DevOpsPetClinic	Running	Web app	Default1	South Central US
Sample9884	Running	Web app	Default1	South Central US
SampleApp9883	Running	Web app	Default1	South Central US

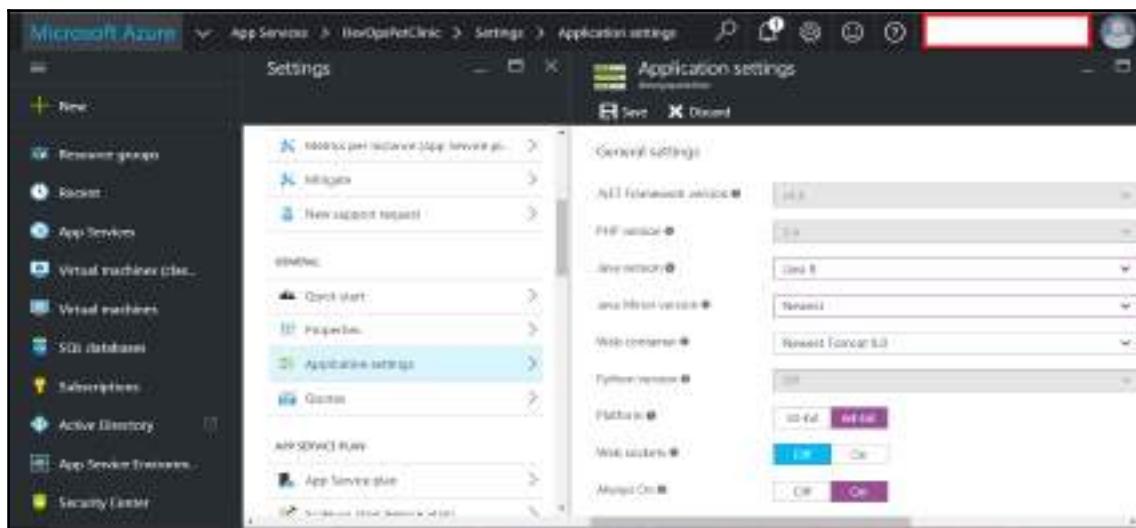
- Click on **DevOpsPetClinic** to obtain details related to the URL, Status, Location, and so on:

The screenshot shows the "Settings" blade for the "DevOpsPetClinic" web app. The left sidebar is identical to the previous screenshot. The main content area is titled "DevOpsPetClinic" and shows the "Essentials" section. It displays the following details:

- URL: <http://devopspetclinic.azurewebsites.net>
- App Service plan using the "Default1" tier (S1 Small)
- FRT deployment slot name: DevOpsPetClinicSlot1251068
- FRT endpoints: <http://www.pet-clinic-001-009.ftp.azurewebsites.net>, <http://www.pet-clinic-001-010.ftp.azurewebsites.net>
- Subscription: Visual Studio Enterprise with MSDN – Don't see a subscription? [Switch directories](#)
- Subscription ID: [View settings](#)

On the right side, there is a "SUPPORT + TROUBLESHOOTING" panel with links to Troubleshoot, Activity Log, Resource Health, Live HTTP traffic, AppLogs, Diagnostics 05-8 Telemetry, Metrics per instance (App Service plan), and Metrics.

6. Click on **All Settings**, go to the **GENERAL** section, and click on **Application settings** to configure the Azure web app for Java web application hosting. Select the **Java version**, **Java Minor version**, **Web container**, and **Platform**, and click on **Always On**:



7. Visit the URL of an Azure web app from your browser and verify that it is ready for hosting our sample Spring application, **PetClinic**:



8. Let's go to the Jenkins dashboard. Click on **New Item** and select **Freestyle project**:

The screenshot shows the Jenkins dashboard with the title 'Jenkins'. In the top right corner, there is a search bar, a 'DiscoverTechno' link, and a 'log out' button. Below the title, there is a breadcrumb navigation: 'Jenkins > All >'. A central dialog box is open, prompting the user to 'Enter an item name'. Inside the dialog, the text 'PetClinic-Deploy-Azure' is entered into a text input field, which is highlighted with a red border. Below the input field, a small note says '» Required field'. The dialog lists four project types with their icons and descriptions:

- Freestyle project**: This is the central feature of Jenkins. Jenkins will build your project, combining any SCM with any build system, and this can be even used for something other than software build.
- Pipeline**: Orchestrates long-running activities that can span multiple build slaves. Suitable for building pipelines (formerly known as workflows) and/or organizing complex activities that do not easily fit in free-style job type.
- Maven project**: Build a maven project. Jenkins takes advantage of your POM files and drastically reduces the configuration.
- Build Flow**: Build Flow can manage job orchestration as a dedicated entity, in a centralized way with complex orchestration and without polluting the jobs with various plugins to handle the upstream-downstream chain.

A large 'OK' button is visible at the bottom left of the dialog.

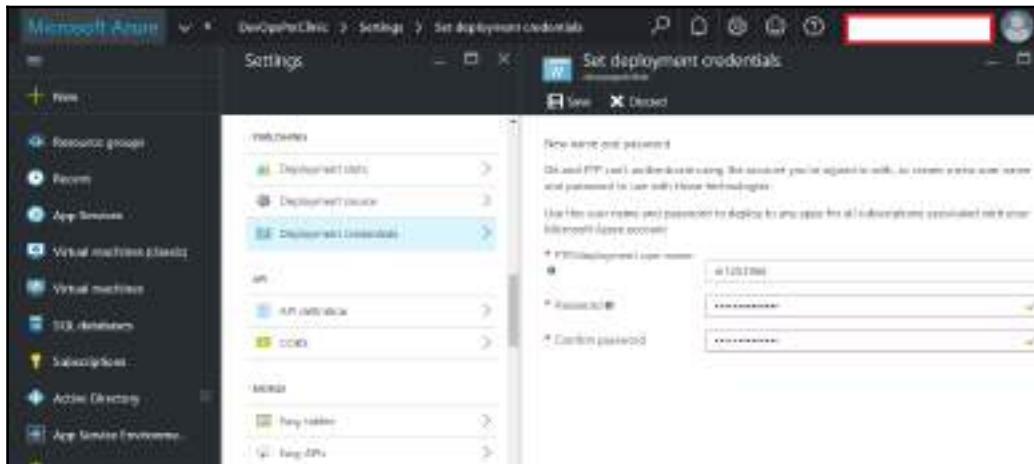
9. Copy the general configuration from another build so we don't need to repeat the configuration work in the newly created job:

The screenshot shows the Jenkins 'New Item' creation dialog. At the top, there are three main options listed:

- Folder**: Creates or container that stores nested items in it. Useful for grouping things together. Unlike view, which is just a filter, a folder creates a separate namespace, so you can have multiple things of the same name existing as they are in different folders.
- GitHub Organization**: Scans a GitHub organization (or user account) for all repositories matching some defined markers.
- Multibranch Pipeline**: Creates a set of Pipeline projects according to selected branches in one SCM repository.

Below these options, there is a note: 'If you want to create a new item from other existing, you can use this option:'. A 'Copy from' field is present, containing the text 'PetClinic-Deploy'. At the bottom of the dialog is a large blue 'OK' button.

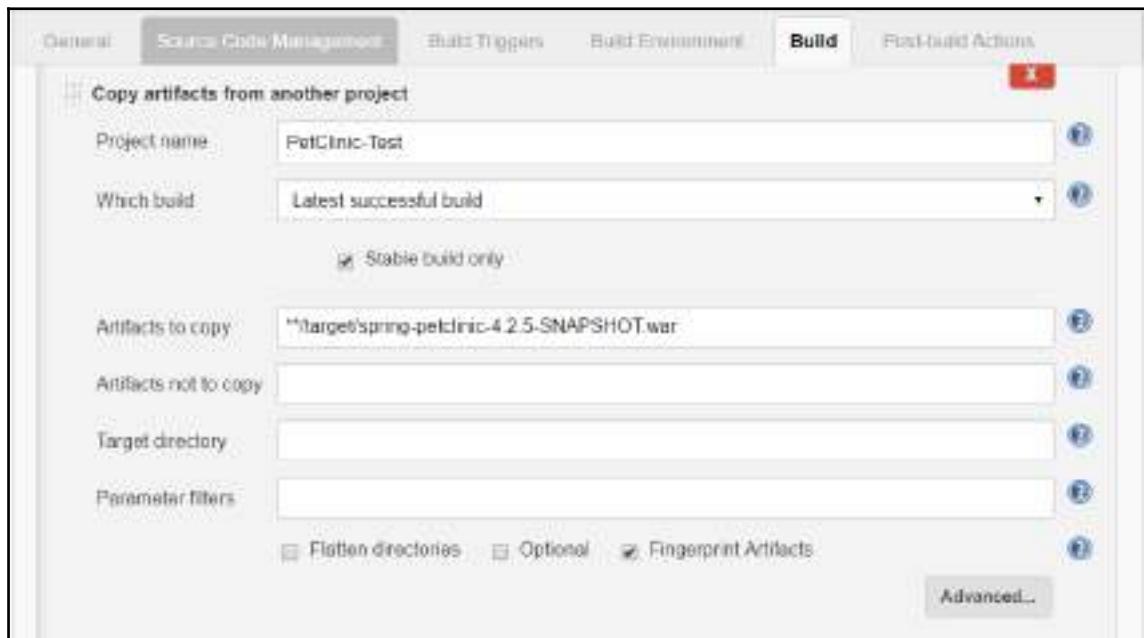
10. Click on **All Settings**, and go to **Deployment credentials** in the **PUBLISHING** section. Provide a username and password, and save your changes:



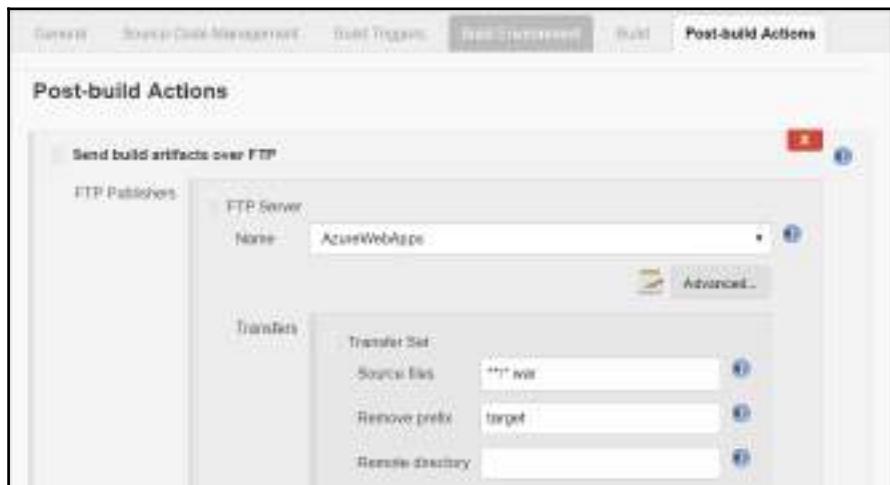
11. In Jenkins, go to **Manage Jenkins** and click on **Configure | Configure FTP settings**. Provide a **Hostname**, **Username**, and **Password**, available in Azure portal.
12. Go to `devopspetclinic.scm.azurewebsites.net` and download the **Kudu console**. Navigate to the different options and find the `site` directory and `webapps` directory. Click on **Test Configuration**, and once you get a **Success** message, you are ready to deploy the PetClinic application:



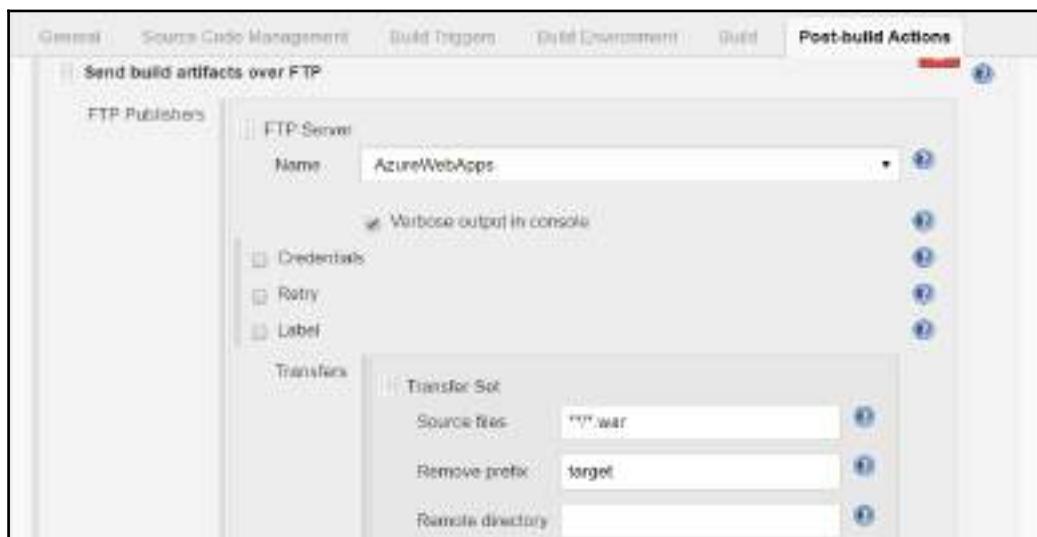
13. In the build job we created, go to the **Build** section and configure **Copy artifacts from another project**. We will copy the WAR file to a specific location on a virtual machine:



14. In **Post-build Actions**, click on **Send build artifacts over FTP**. Select the FTP server name configured in Jenkins. Configure **Source files** and the **Remove prefix** accordingly for deployment of an Azure web app:



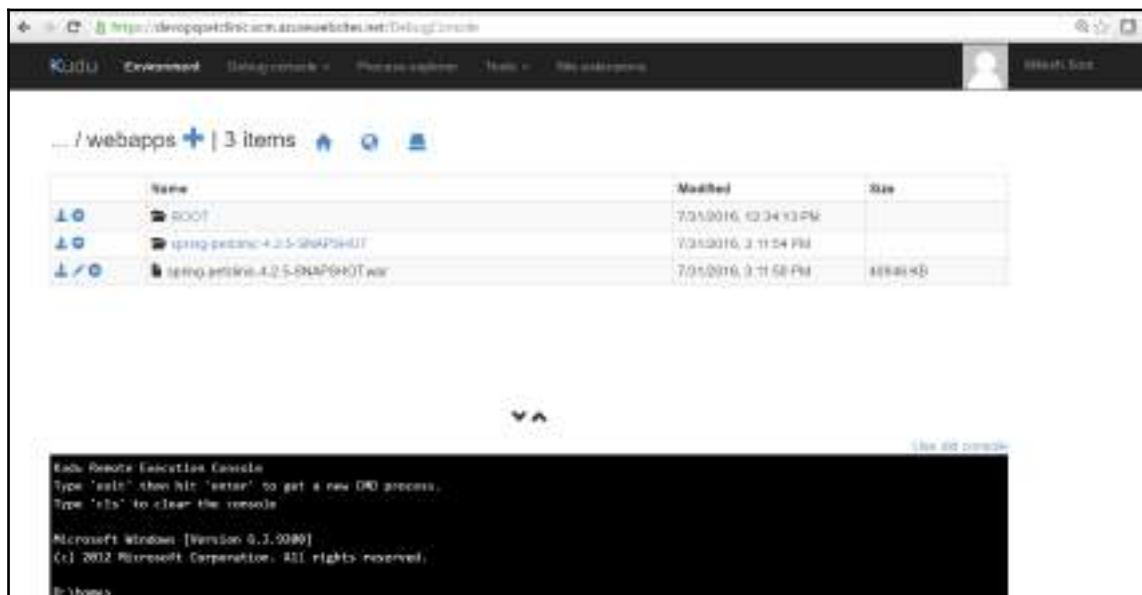
15. Tick **Verbose output in console**:



16. Click on **Build now** and see what happens behind the scene:

```
Started by user discoverTechie
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 the **Kudu** console, click on **Debug console**, and go to **Powershell**. Go to site | wwwroot | webapps. Check whether the WAR file has been copied:



18. Visit the Azure web app URL in the browser with the context of an application:

The screenshot shows a web browser window with the URL `devopspetclinic.azurewebsites.net/spring-petclinic-4.2.5-SNAPSHOT/vets.html`. The page has a header with the Spring logo and navigation links for HOME, FIND OWNERS, VETERINARIANS, and LOGOUT. A search bar is also present. The main content area is titled "Veterinarians" and displays a table with the following data:

Name	Specialties
Helen Lavery	veterinology
Henry Stevens	veterinology
James Carter	none
Linda Douglas	dentistry, surgery
Rafael Ortega	surgery
Sharon Jenkins	veterinology

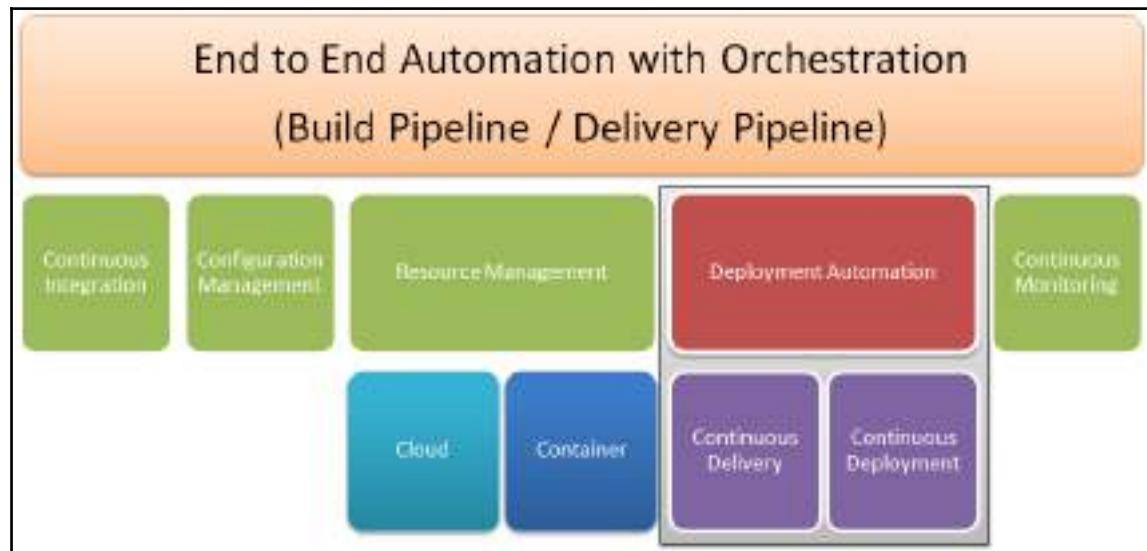
At the bottom of the table, there is a link "View all Vets/Veterinarians (20)". The footer of the page features the Spring logo.

Now we have an application deployed on Azure web apps.



It is important to note that the FTP username has to be with the domain. In our case, it can be Sample9888\m1253966. Using the username without the web app name won't work.

All these different ways of deployment to AWS IaaS, AWS PaaS, Microsoft Azure PaaS, and Docker container can be used in the final end-to-end automation:



We have covered four phases so far, and we will next discuss about continuous monitoring. In the final chapter, we will manage the entire end-to-end automation with pipeline or orchestration.

## Self-test questions

State whether the following statements are true or false:

1. Role and users in Tomcat can be created in `tomcat-users.xml` to access Manager web app?
  - True
  - False
  
2. To access the **Tomcat Manager** app, the `manager-script` role is required?
  - True
  - False

3. To deploy application in Tomcat container using the Deploy plugin in Jenkins, the manager-script role is required?
  - True
  - False
4. AWS Elastic Beanstalk and Azure app services are PaaS offerings from Amazon and Microsoft respectively?
  - True
  - False
5. Which of the following are steps for application deployment on AWS Elastic Beanstalk?
  - Create an application (PetClinic)
  - Upload WAR file as an application version
  - Launch an environment
  - Deploy the new version of the application on AWS Elastic Beanstalk
  - All of these

## 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.

# 8

# Monitoring Infrastructure and Applications

*“A lot of times, people don’t know what they want until you show it to them.”*  
- Steve Jobs

Cloud provides agility, scalability, pay as you go resources, and so on. Based on the Cloud service models, roles and responsibilities of Cloud-service providers and Cloud consumers are different. Having said that, it is equally important to know the status of the cloud resources irrespective of the Cloud deployment model including Private Cloud or Public Cloud. It is advisable to have detailed perspective of cloud resources to maintain and manage high availability and reputation.

An important thing to note here is that all resources are interdependent and if one resource is not in the sync of overall picture than main objective of providing good service and high availability is difficult to achieve. This is a scenario irrespective of the type of environment including physical, virtualized, or cloud.

This chapter describes the need of continuous monitoring and its significance in the end to end automation process in the context of DevOps culture development. It covers different aspects of monitoring such as cloud resources, application server and application monitoring to increase services, and application availability.

In this chapter, we will cover the following topics:

- Getting started-monitoring
- Installation and configuration open source monitoring tools
- Monitoring AWS, Azure resources
- Monitoring web application and Tomcat server with New Relic

## **Getting started – monitoring**

Let's start with a simple definition of monitoring and then we will gradually move towards cloud monitoring. It is about observing progress of some operation or activities performed and to make sure that end goal or performance objectives are achieved as desired. It expands its area from observing to analysis, from analysis to detection, and from detection to notifications to respective stakeholders for corrective measures.

In general, what kind of monitoring we do in traditional environment? Let's start considering component of Cloud service Model to understand it in a better perspective. In Cloud computing, there are three service models:

1. Infrastructure as a Service
2. Platform as a Service
3. Software as a Service

To get a clue from it, it is extremely important to monitor Infrastructure as well as platform running web application. Let's take a case of AWS EC2 instance that can be used to deploy our sample Spring Application – PetClinic. We need to ensure that Instance is running properly, tomcat server is performing well, memory consumption, CPU utilization, and so on. The next level is to monitor application itself. Application security monitoring and performance monitoring is extremely critical to make an application highly available and avoid failures.

Monitoring strategy has to be end to end to be effective. It should be in place at the time of design and not the afterthought. Prevention is always better than cure and it avoids cost implications, rollback to last successful builds, and last minute headaches.

## **Overview of Monitoring tools and Techniques**

In this section, we will cover Nagios monitoring tool, Azure Web Apps Monitoring, and AWS Elastic Beanstalk.

## Nagios

Nagios Core is an open source application written in C and PHP to monitor servers, networks, and infrastructure.

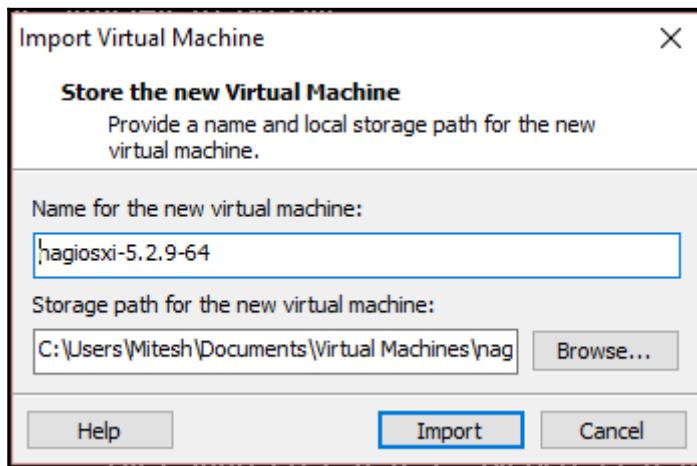
Below are some important features of Nagios:

Nagios supports monitoring of host resources such as processor and disk usage, network services,	Supports Auto Discovery to automatically discovers hosts or network devices Zenoss is connected to	Support IP SLAs Reports
Support User-defined groups to organize hosts or devices Zenoss monitors.	Tracking information related to data over duration and also to predict network statistics in Nagios XI	Support SNMP statistics, reports on Syslogs
Supports secure remote monitoring through SSH or SSL encrypted tunnels		Simple and Easy to use web interface for analysing current network status, notifications, and so on.

Nagios XI uses Nagios Core as the back-end and provides an extended interface for monitoring resources. Nagios XI is supported in CentOS and RedHat. Let's have a quick tour of Nagios XI.

## Quick start with Nagios

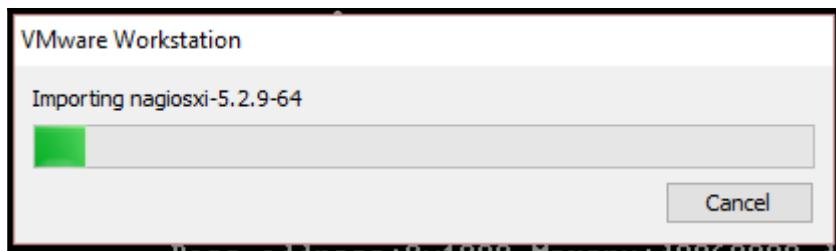
1. Go to <https://www.nagios.com/downloads/nagios-xi/> and download VMware 64 bit virtual machine so we can use the image in VMware workstation. Open the OVF in the VMware workstation:



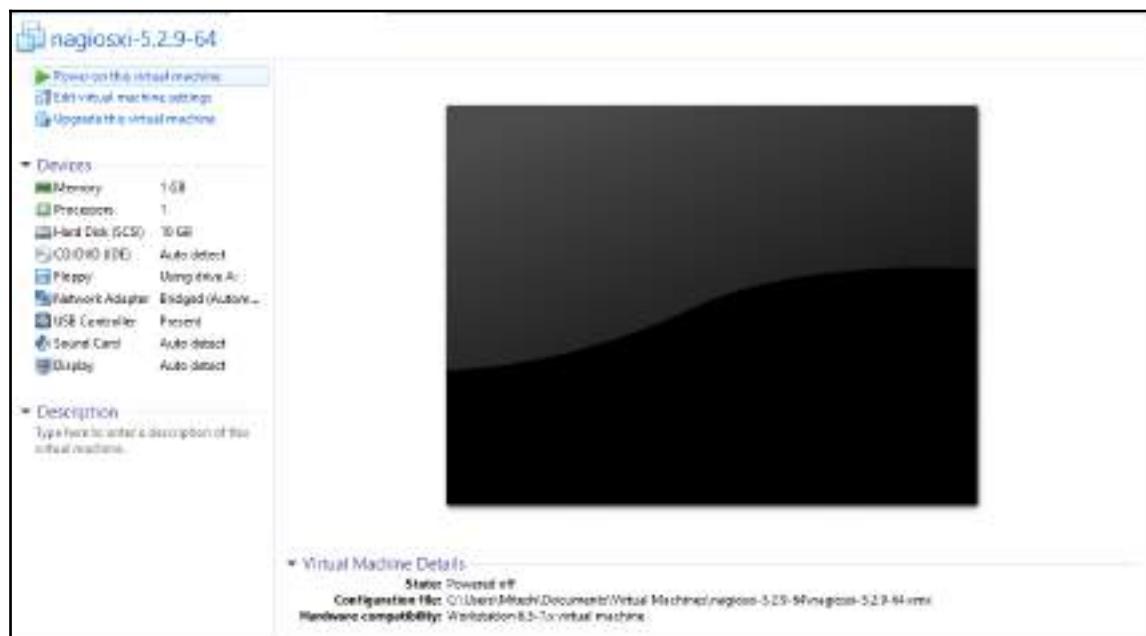
2. Importing Nagios OVF will take some time:



3. Wait for the completion of the process:



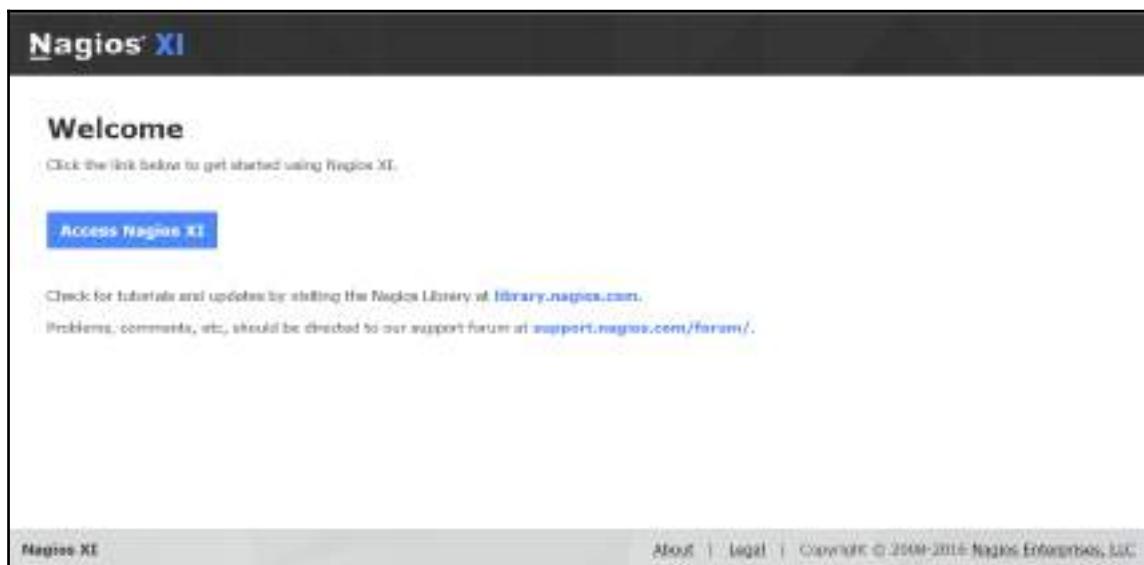
4. Once importing process is completed, click on the **Power on this virtual Machine**:



- Once the virtual machine is ready, login as root with password nagiosxi. Note the URL to access Nagios dashboard:

```
(B) Nagios XI™  
Access: Nagios XI at 192.168.51.10  
Welcome to the Nagios XI command-line interface.  
CentOS release 6.8 (Final).  
Kernel 2.6.32-642.1.1.el6.x86_64 on an x86_64  
localhost login: root:  
Password:  
Last login: Fri Apr 5 11:07:12 from 192.168.5.15  
root@localhost:~$ ls  
nagiosxi.cfg install.log install.log.syslog scripts  
root@localhost:~$
```

- Open the Nagios URL from the browser. Click on the **Access Nagios XI**:



7. Note the username and password with other details, select the **timezone** and click on **Install**:

The screenshot shows the 'Nagios XI Installer' interface. At the top, it says 'Welcome to the Nagios XI installation. Just answer a few simple questions and you'll be ready to go.' Below this, there's a section titled 'General Program Settings' with five input fields:

Program URL:	http://192.168.1.38/nagiosxi/
Administrator Name:	Nagios Administrator
Administrator Email Address:	root@localhost
Administrator Username:	nagiosadmin
Administrator Password:	5kerpt

Below these settings is a 'Timezone Settings' section with a dropdown menu set to 'America/New\_York'. At the bottom left is a blue 'Install >' button.

- Once installation is complete, click on the **Login to Nagios XI**:



Now, we have installed Nagios. Next step will be to configure Nagios to monitor resources from Cloud platforms.

Now, to monitor Cloud resources with Nagios, we will follow a simple scenario to monitor an AWS instance using Nagios:

- Login to Nagios dashboard using admin credentials:

A screenshot of the Nagios XI login interface. On the left, there's a 'Login' form with fields for 'username' and 'password', and a 'Login' button. Below it is a link 'Forgot your password?'. On the right, there's a large banner with the Nagios XI logo and a background image of network nodes. Below the banner, there's an 'About Nagios XI' section with a brief description and a link to the product page. Further down, there's a 'Nagios Learning Opportunities' section with links to training and certification information, and a 'Contact Us' section with a message about contacting support. At the bottom, there's a footer with links for 'About', 'Legal', and 'Copyright © 2009-2016 Nagios Enterprises, LLC'.

## 2. Accept License Agreement:



3. Here is the Nagios dashboard and we can configure our AWS instance to monitor with it:



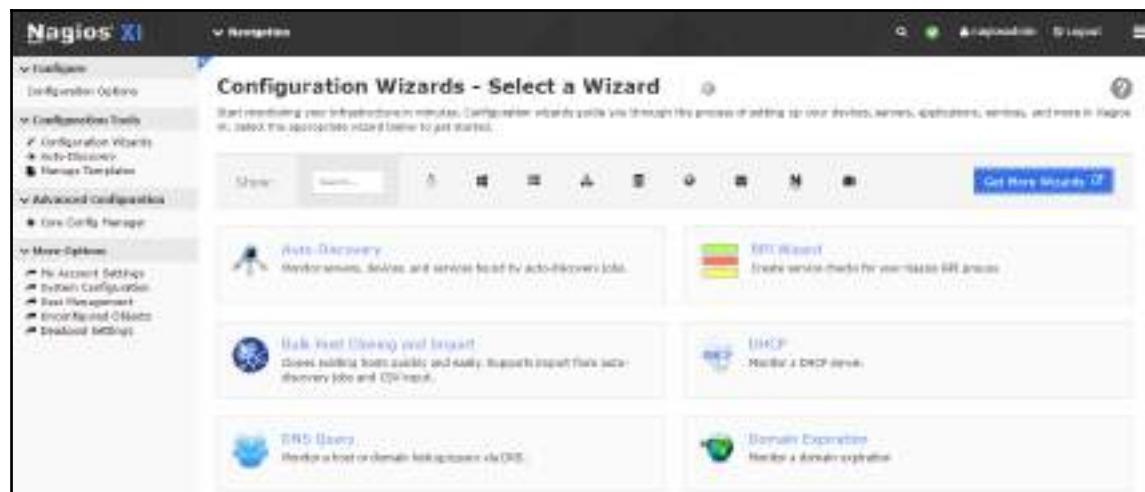
4. From the **Navigation** menu, got to **Configure** section for Auto Discovery Job:

The screenshot shows the Nagios XI web interface. The left sidebar has a 'Configure' section with 'Configuration Options' expanded, showing 'Auto-Discovery' under 'Configuration Tasks'. The main content area is titled 'New Auto-Discovery Job' with the sub-instruction 'Use this form to configure an auto-discovery job.' It contains fields for 'Scan Target' (set to '192.168.1.254'), 'Exclude IP's' (empty), 'Schedule' (set to 'FirstRun'), and 'Edit Auto-Discover Options' (link). At the bottom are 'Submit' and 'Cancel' buttons.

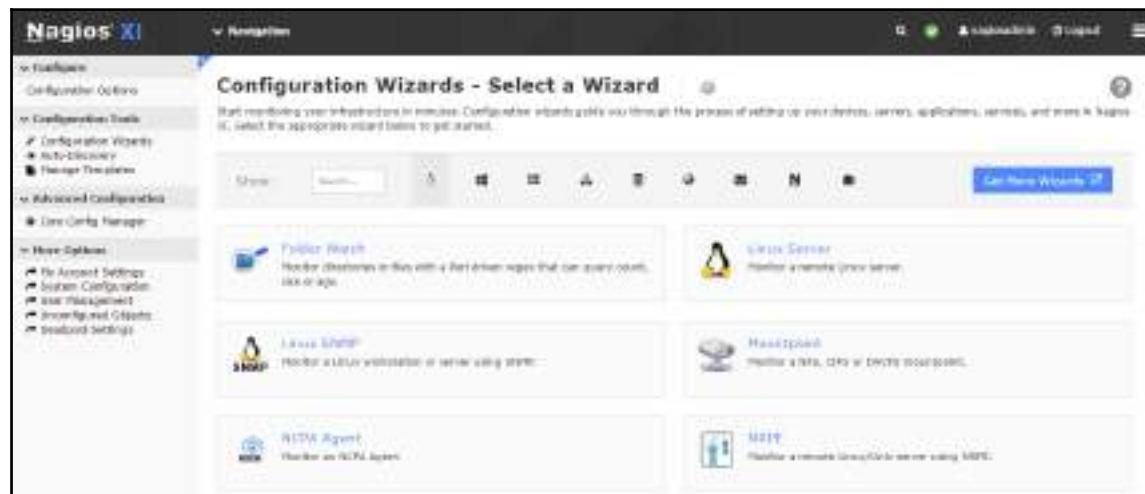
5. There are different options also available to monitor in the configure section:

The screenshot shows the 'Configuration Options' page. The left sidebar has a 'Configure' section with 'Configuration Options' expanded, showing 'Auto-Discovery' under 'Configuration Tasks'. The main content area has four sections: 'Start Monitoring Now' (with a link to 'Start Monitoring Now'), 'Auto-Discovery' (with a link to 'Create New Discovery Job'), 'Advanced Configuration' (with a link to 'Go to Nagios XI's Config Manager'), and 'Manage Account Settings' (with a link to 'Edit your profile settings').

6. We can also configure monitoring using **Configuration wizards**:



7. We have created AWS Linux instance and we would like to monitor it using Nagios. Select a **Linux Server**:



- Get the IP address of AWS instance and input it here. Select the Linux Distribution as Other. Click on Next:

The screenshot shows the Nagios XI configuration interface. On the left, there's a sidebar with navigation links like 'Configuration', 'Configuration Tools' (which is currently selected), 'Advanced Configuration', and 'More Options'. The main area is titled 'Configuration Wizard: Linux Server - Step 1' and features a penguin icon. It's asking for 'Linux Server Information'. The 'IP Address' field contains '52.91.225.24' and the 'Linux Distribution' dropdown is set to 'Other'. Below the form, a note says 'The Linux distribution running on the server you'd like to monitor.' At the bottom are 'Back' and 'Next >' buttons.

- Verify the IP address in the next step:

The screenshot shows the 'Configuration Wizard: NRPE - Step 1' page. It has a note at the top: 'You have been sent here from the **Linux Server Wizard** because you selected **Other** as your linux type. The NRPE Wizard is a similar wizard with more customizability. **Don't forget! You will need to install NRPE just like you would in the Linux Server Wizard.**' Below this, there's a 'Server Information' section with an 'IP Address' field containing '52.91.225.24' and an 'Operating System' dropdown set to 'Linux - Other'. A note below the OS dropdown says 'The operating system running on the server you'd like to monitor.' At the bottom are 'Back' and 'Next >' buttons.

10. Provide Host name for the EC2 instance:

 **Configuration Wizard: NRPE - Step 2** | 

**Server Details**

**IP Address:**

**Operating System:**   
Linux

**Host Name:**   
The name you'd like to have associated with this host.

**NRPE Agent**

Specify options that should be used to communicate with the remote NRPE agent.

**SSL Encryption:**   
Determines whether or not data between the Nagios XI server and NRPE agent is encrypted.  
**Note:** Legacy NRPE installations may require that SSL support be disabled.

**Server Metrics**

Specify which services you'd like to monitor for the server.

11. Configure the basic parameters for monitoring. Click on **Next**:

The screenshot shows the third step of the Configuration Wizard for NRPE. The title is "Configuration Wizard: NRPE - Step 3". The section is titled "Monitoring Settings" with the sub-instruction "Define basic parameters that determine how the host and service(s) should be monitored." Below this, under "Under normal circumstances:", there is a field "Monitor the host and service(s) every [5] minutes." Under "When a potential problem is first detected:", there is a field "Re-check the host and service(s) every [1] minutes up to [5] times before generating an alert." At the bottom are three buttons: "Back" (disabled), "Next >" (highlighted in blue), and "Finish" (disabled).

12. Configure Notification Settings:

The screenshot shows the fourth step of the Configuration Wizard for NRPE. The title is "Configuration Wizard: NRPE - Step 4". The section is titled "Notification Settings" with the sub-instruction "Define basic parameters that determine how notifications should be sent for the host and service(s)." Under "When a problem is detected:", there are three options: "Don't send any notifications" (radio button), "Send a notification immediately" (radio button, selected), and "Wait [15] minutes before sending a notification". Under "If problems persist:", there is a field "Send a notification every [60] minutes until the problem is resolved." Under "Send alert notifications to:", there are two checked checkboxes: "Myself (adjust my settings)" and "Other individual contacts". A dropdown menu below shows "Default Contact (x1\_default\_contact)".

13. Configure the **Host Groups**:

**Configuration Wizard: NRPE - Step 5**

Host Groups

Define which hostgroup(s) the monitored host should belong to (if any).

Linux Servers (linux-servers)

14. Click **Apply** to finish the configuration:

**Configuration Wizard: NRPE - Final Step**

Final Settings

Click **Apply** to add your new configuration.

[◀ Back](#)   [\*\*✓ Apply\*\*](#)   [\*\*Save as Template\*\*](#)

15. Within few minutes, monitoring of EC2 instance will start:

The screenshot shows the Nagios XI Home Dashboard. On the left, there's a navigation sidebar with sections like 'Quick View', 'Hosts', 'Services', 'Metrics', 'Metrics', 'Metrics', and 'Logs'. The main area has three main sections: 'Getting Started Guide' (with tasks like 'Change user account settings' and 'Configure user notifications'), 'Host Status Summary' (with a table showing 1 Up, 1 Down, 0 Unreachable, 0 Pending; status: Unhandled, Problems, All), and 'Service Status Summary' (with a table showing 1 OK, 1 Warning, 0 Unknown, 1 Critical, 0 Pending; status: Unhandled, Problems, All). At the bottom, there's a footer with links like 'Help', 'Logout', and 'Copyright © 2010-2011 Nagios Enterprises, LLC'.

16. Click on the EC2 instance in Nagios dashboard. It is showing the status as **Down**:

The screenshot shows the 'Host Status' page under the 'Hosts' section. It displays a 'Host Status Summary' table and a 'Service Status Summary' table. Below these are search and filter fields ('Search...', 'Q', 'Filters: Host=Down'). A message says 'Showing 1-1 of 1 total records'. A table at the bottom lists one host entry: 'ec2' with status 'Down' (indicated by a red background), duration '1h 20m 46s', attempt '5/5', last check '2010-07-15 13:24:20', and status information 'CRITICAL - 52.91.225.24:80: name, load 100%'. There are also pagination controls ('Page: 1 of 1', '15 Per Page', 'Go').

17. Verify the Host status in detail. Click on the **Ping this host**:

**Host Status Detail**

**ec2**  
Alias: ec2  
Hostgroups: linux-servers

**CRITICAL - 52.91.225.24: eti Iran, lost 100%**

**Address:** 52.91.225.24

**Status Details**

Host State:	● Down
Duration:	1h 27m 30s
Host Stability:	Unchanging (stable)
Last Check:	2016-07-16 13:25:04
Next Check:	2016-07-16 13:29:32

**Quick Actions**

- Acknowledge this problem
- Disable notifications
- Force an immediate check
- Ping this host
- Connect to ec2
- Traceroute to this host

18. We will get packet loss message:

102.168.1.38/nagiosxi/includes/components/pingaction/ping.php?host=52.91.2...  
192.168.1.38/nagiosxi/includes/components/pingaction/ping.php?host=52.91.2...

**Ping Output:** [Close This Window](#)

```
PING 52.91.225.24 (52.91.225.24) 56(84) bytes of data.  
--- 52.91.225.24 ping statistics ---  
6 packets transmitted, 0 received, 100% packet loss, time 5999ms
```

[Ping another host](#)

19. Go to AWS Management console and change in the Inbound rules of security group assigned to EC2 instance to all ICMP traffic. Verify Ping this Host again and we can see Ping requests are successful:

```
PING 52.91.225.24 (52.91.225.24) 56(84) bytes of data.  
64 bytes from 52.91.225.24: icmp_seq=1 ttl=240 time=243 ms  
64 bytes from 52.91.225.24: icmp_seq=2 ttl=240 time=242 ms  
64 bytes from 52.91.225.24: icmp_seq=3 ttl=240 time=241 ms  
64 bytes from 52.91.225.24: icmp_seq=4 ttl=240 time=741 ms  
64 bytes from 52.91.225.24: icmp_seq=5 ttl=240 time=930 ms  
  
--- 52.91.225.24 ping statistics ---  
5 packets transmitted, 5 received, 0% packet loss, time 4932ms  
rtt min/avg/max/mdev = 241.612/479.757/930.316/296.754 ms
```

[Ping another host](#)

20. Verify the Host Availability:



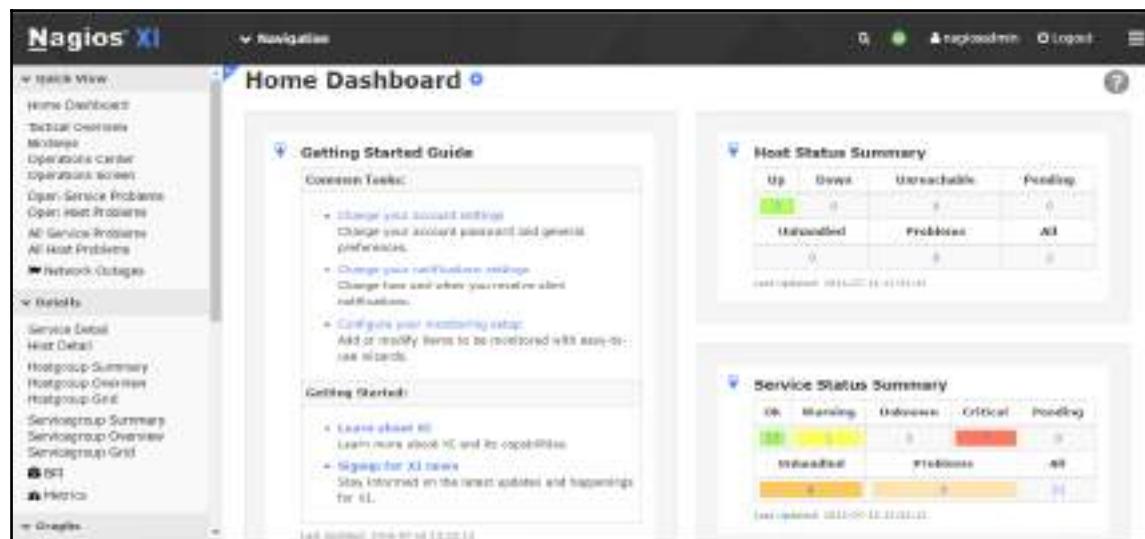
21. Click on the graph and get details on **Host Data** and **Service Data**:

Host Data					
Host	UP	Down	Unreachable		
ec2	92.344%	7.656%	0%		

Service Data					
Host	Service	OK	Warning	Unknown	Critical
ec2	Current Load	90.662%	0%	0%	9.338%
	Current Users	90.676%	0%	0%	9.324%
	Ping	92.419%	0%	0%	7.581%
	Total Processes	90.792%	0%	0%	9.208%
	Average	91.132%	0.000%	0.000%	8.863%

22. Verify the Nagios dashboard and all three configured instances are **Up**:



23. Verify the **Host Status Detail** again:

The screenshot shows the 'Host Status Detail' page for a host named 'ec2'. At the top, there is a penguin icon, the host name 'ec2', its alias 'ec2', and its hostgroup 'linux-servers'. Below this is a toolbar with icons for Overview, Graph, Add, Settings, Metrics, and Log. A status message 'OK - 52.91.225.24: rta 1022.218ms, lost 20%' is displayed. The address of the host is listed as 52.91.225.24. The 'Status Details' section contains the following table:

Host State:	Up
Duration:	6m 52s
Host Stability:	Unchanging (stable)
Last Check:	2016-07-16 13:35:32
Next Check:	2016-07-16 13:40:42

The 'Quick Actions' section includes links to Disable notifications, Force an immediate check, Ping this host, Connect to ec2, and Traceroute to this host.

24. From the **Incident Management** section, click on the **Latest Alerts** to get details on latest notifications or alerts:

The screenshot shows the 'Latest Alerts' section of the Nagios XI interface. On the left, there is a sidebar with navigation links for Quick View, Details, Graphs, Maps, Incident Management (Latest Alerts, Acknowledgements, Scheduled Downtime, Mass Acknowledge, Recurring Downtime, Notifications), and Monitoring Process (Process Info, Performance, Event Log). The main area displays the 'Latest Alerts' table:

Source	Latest Alert	Alerts
localhost	2016-07-16 13:17:36	SERVICE Monit - httpd is Warning
192.168.1.24	2016-07-16 11:16:10	HTTPS_MBEDTLS are Critical
m2	2016-07-16 11:12:22	Current Load, Current Users, Total Processes are Critical

25. Click on the **ec2**, and get all the notification for EC2 instance:

The screenshot shows the Nagios XI interface under the 'Notifications' section. The left sidebar includes links for 'Quick View', 'Metrics', 'Graphs', 'Maps', 'Incident Management', 'Latest Alerts', 'Administrative Groups', 'Scheduled Downtime', 'Mass Acknowledge', 'Recurring Downtime', and 'Notifications'. The main content area displays a table titled 'Notifications' with the following details:

Date / Time	Host	Service	Status	Escalated	State	Contact	Dispatcher	Description
2016-07-18 13:34:38	ec2	Current status	Service Problem	No	CRITICAL	responsive	Nagios XI	EC2_9999: socket timeout after 10 seconds.
2016-07-18 13:34:57	ec2	HTTP - Processes	Service Problem	No	CRITICAL	responsive	Nagios XI	EC2_9999: socket timeout after 10 seconds.
2016-07-18 13:39:13	ec2	Current status	Service Problem	No	CRITICAL	responsive	Nagios XI	CHICKEN_NRPE: Socket timeout after 10 seconds.
2016-07-18 13:39:08	192.168.1.34, http/80	Service Problem	Service Problem	No	CRITICAL	responsive	Nagios XI	connect: address: 192.168.1.34:443 port: 139 connection refused
2016-07-18 13:29:57	ec2	Host Recovery	Pending	No	UP	responsive	Nagios XI	OK - 0.00 220.341 ms 251.803 ms free 0%
2016-07-18 13:29:37	localhost	Service Status - ec2	Service Problem	No	WARNING	responsive	Nagios XI	http dead but pid file exists

26. We can verify Service Status from Nagios Dashboard:

The screenshot shows the Nagios XI Service Status dashboard. It features two summary sections: 'Host Status Summary' and 'Service Status Summary', both displaying counts for Up, Down, Unreachable, and Pending states. Below these are two tables:

- Host Status:** Shows the status of various hosts. One host, 'ec2', is listed as 'Down' with a 'Critical' status due to a socket timeout after 10 seconds. Other hosts like 'Ganesha' and 'Pkg' are shown as 'Up' or 'OK'.
- Service Status:** Shows the status of services across hosts. A service on 'ec2' is listed as 'Critical' due to a socket timeout after 10 seconds. Other services like 'http/80' and 'nrpe' are shown as 'OK'.

27. Verify all hosts' status from Nagios dashboard.

The screenshot shows the Nagios Host Status dashboard. At the top, there are two summary tables: 'Host Status Summary' and 'Service Status Summary'. Below these are search and filter fields, and navigation links for page 1 of 1 with 15 per page. The main area displays a table of hosts, each with a status icon (green for up, red for down), name, status, duration, attempt, last check, and status information. The table shows three hosts: 192.168.1.34 (up), web (up), and localhost (up). The status information for 192.168.1.34 indicates OK - 192.168.1.34: ita 2.099ms, lost 0%. The bottom of the screen shows the date and time as 2016-07-18 13:48:05.

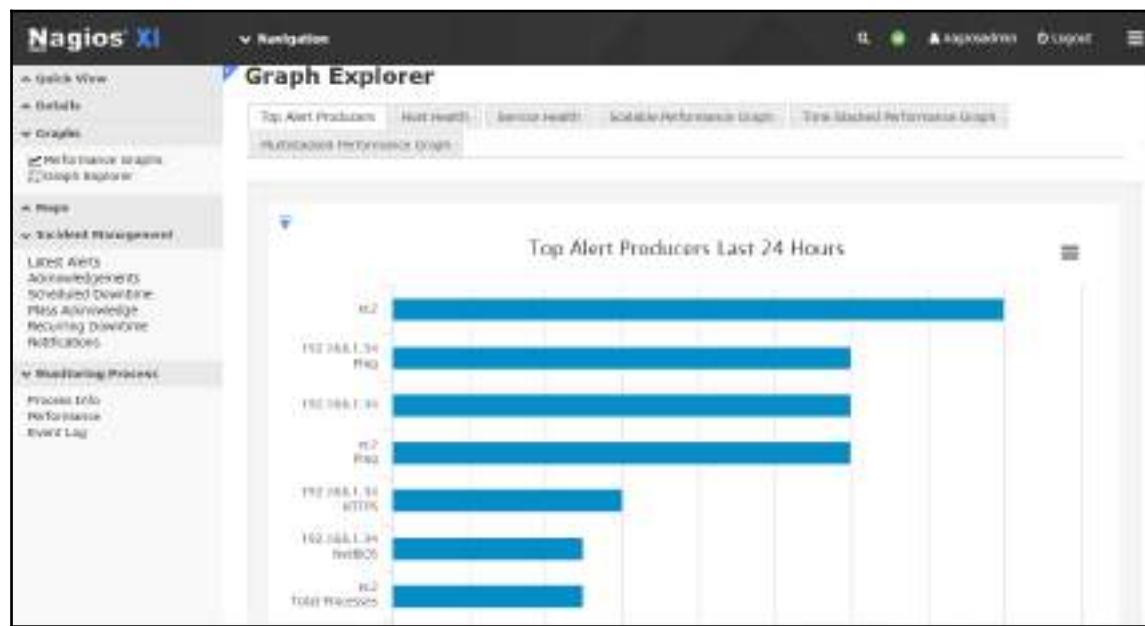
Host	Status	Duration	Attempt	Last Check	Status Information
192.168.1.34	Up	1h 27m 51s	1/5	2016-07-18 13:45:08	OK - 192.168.1.34: ita 2.099ms, lost 0%
web	Up	17m 29s	1/5	2016-07-18 13:45:32	OK - 52.91.225.24: ita 0.01.80ms, lost 0%
localhost	Up	2h 52m 30s	1/10	2016-07-18 13:45:31	OK - 127.0.0.1: ita 0.031ms, lost 0%

28. We can also verify **Host Group Status** from the Nagios Dashboard:

The screenshot shows the Nagios Host Group Status dashboard. It features two summary tables: 'Host Status Summary' and 'Service Status Summary'. Below these are navigation links for page 1 of 1 with 15 per page. The main area displays a table for the 'Linux Servers (linux-servers)' group. The table includes columns for Host, Status, and Services. It lists three hosts: 192.168.1.34 (up), web (down), and localhost (up). The services listed for each host include various monitoring metrics like CPU load, memory usage, and process counts. The bottom of the screen shows the date and time as 2016-07-18 13:48:03.

Host	Status	Services
192.168.1.34	Up	cpu, memory, disk, http, ssh, ntp, log, cron, users, httpd, ntpd, root, httpd, service status, cron, service status, http, service status, mem, service status, memory, service status, ntp, service status, httpd, service status, http, httpd, swap usage, top, processes
web	Down	cpu, memory, disk, http, ssh, ntp, log, cron, users, httpd, ntpd, root, httpd, service status, cron, service status, http, service status, mem, service status, memory, service status, ntp, service status, httpd, service status, http, httpd, swap usage, top, processes
localhost	Up	cpu, memory, disk, http, ssh, ntp, log, cron, users, httpd, ntpd, root, httpd, service status, cron, service status, http, service status, mem, service status, memory, service status, ntp, service status, httpd, service status, http, httpd, swap usage, top, processes

29. In the **Graphs** section, click on **Graph Explorer** to get graphical details on top alerts produced in the last 24 hours:



In the next section we will use Nagios to monitor Tomcat instance.

## Monitoring AWS Elastic Beanstalk

We have deployed PetClinic Application in to AWS Elastic Beanstalk as well with the use of Jenkins plugin. In AWS Elastic Beanstalk, health status of an environment is determined by Grey, Green, Yellow, and Red color. Grey indicates that environment is in the process of updation. Green indicates successful health check status in recent times. Yellow indicates that environment has failure of one or more Health checks. Red indicates that environment has failure of three or more Health checks.

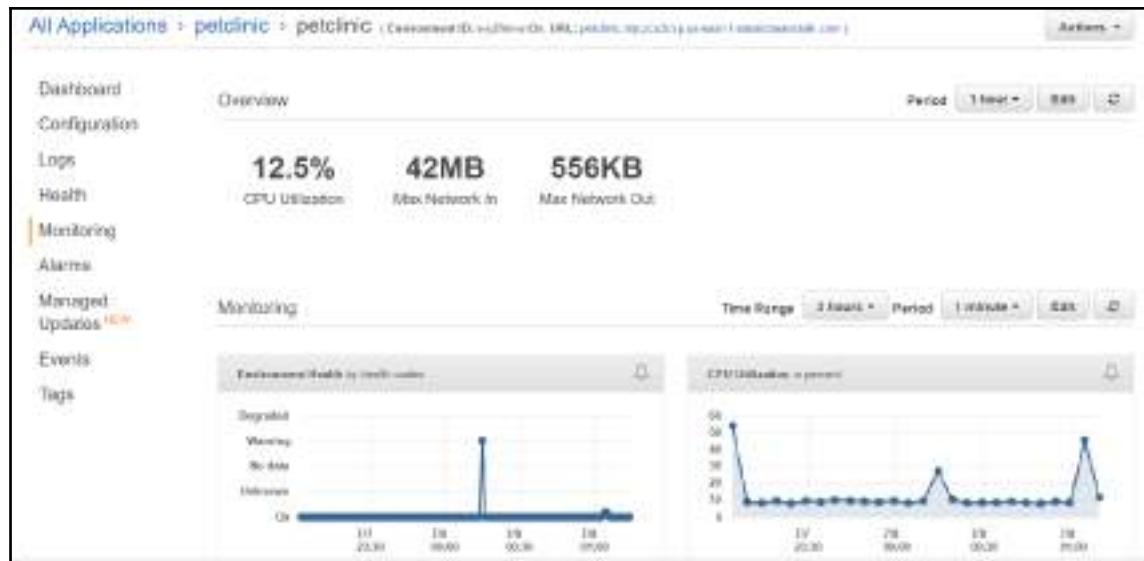
Health status is based on the response of an application running in the **Environments**:

The screenshot shows the AWS Elastic Beanstalk interface. In the top navigation bar, 'AWS Services' is selected, and 'petclinic' is chosen from the dropdown. On the left, a sidebar menu includes 'Environments', 'Application Versions', and 'Saved Configurations'. The main content area displays the 'petclinic' environment, which is highlighted with a green background. The details shown are: Environment type: Web Server, Running version: Simple Application, Last modified: 2016-01-07 20:05:59 UTC+0500, and URL: petclinic.us-east-1.elasticbeanstalk.com.

On the Environment dashboard in AWS Elastic Beanstalk, we get basic details such as **Health** as well as configuration of instances:

The screenshot shows the AWS Elastic Beanstalk environment dashboard for 'petclinic'. The top navigation bar shows 'AWS Services' and 'petclinic'. The left sidebar lists 'Dashboard', 'Configuration', 'Logs', 'Health', 'Monitoring', 'Alarms', 'Managed Updates', and 'Events'. The main dashboard has tabs for 'Overview' and 'Actions'. Under 'Overview', there is a large green circle with a checkmark icon, labeled 'Health' and 'Ok'. To its right, it shows the 'Running Version' as 'Simple Application' and a 'Upload and Deploy' button. To the right of the version information is a yellow cat icon and a 'Configuration' section. The configuration section includes the text 'EB@Amazon Linux 2016.08 v2.1.3 running Tomcat 8 Java 8' and a 'Change' button. A 'Refresh' button is also present at the top right of the dashboard.

Click on the **Monitoring** for extensive monitoring details in form of **CPU Utilization** and **Health** of an application. We can change **Time Range** to get more details on **Monitoring**:

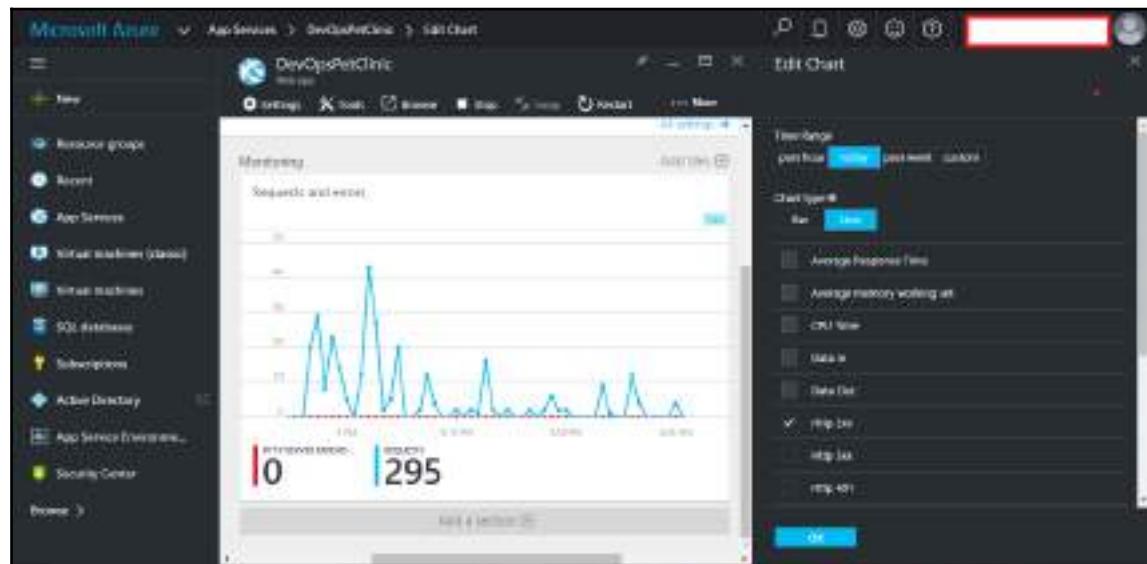


For more information on monitoring AWS Elastic Beanstalk, visit <http://docs.aws.amazon.com/elasticbeanstalk/latest/dg/environments-health.html>

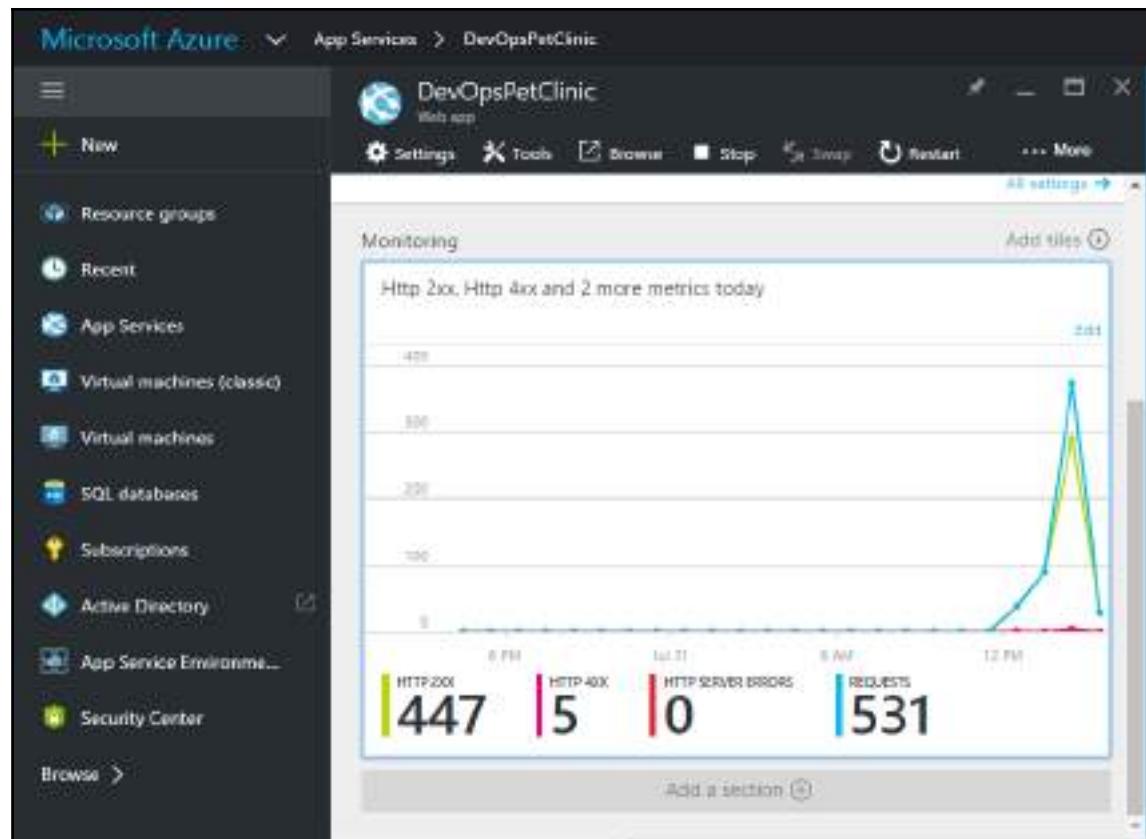
# Monitoring Microsoft Azure Web App Service

In Chapter 6, *Cloud Provisioning and Configuration Management* we deployed PetClinic Application in the Azure Web Apps. Once deployment is successful, monitoring Web App is an essential activity and Azure Portal itself provide many ways to monitor it.

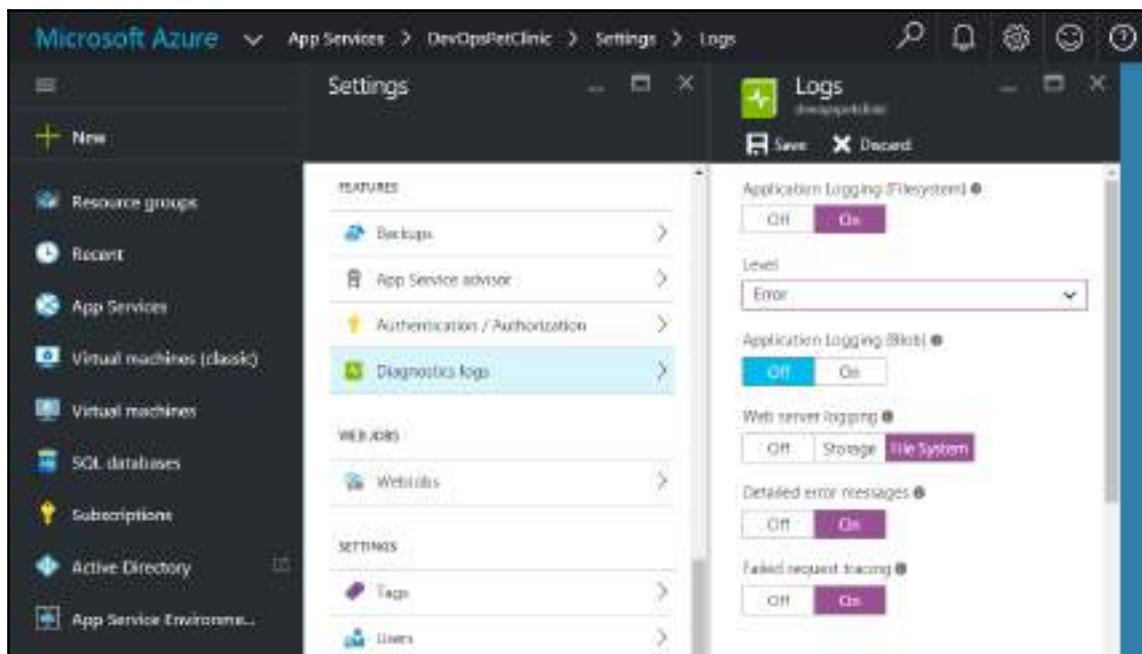
1. In Azure web app find **Monitoring** section below Application details. Click on **edit** to change the time range and chart type:



2. Verify the updated graph with more details based on selection:



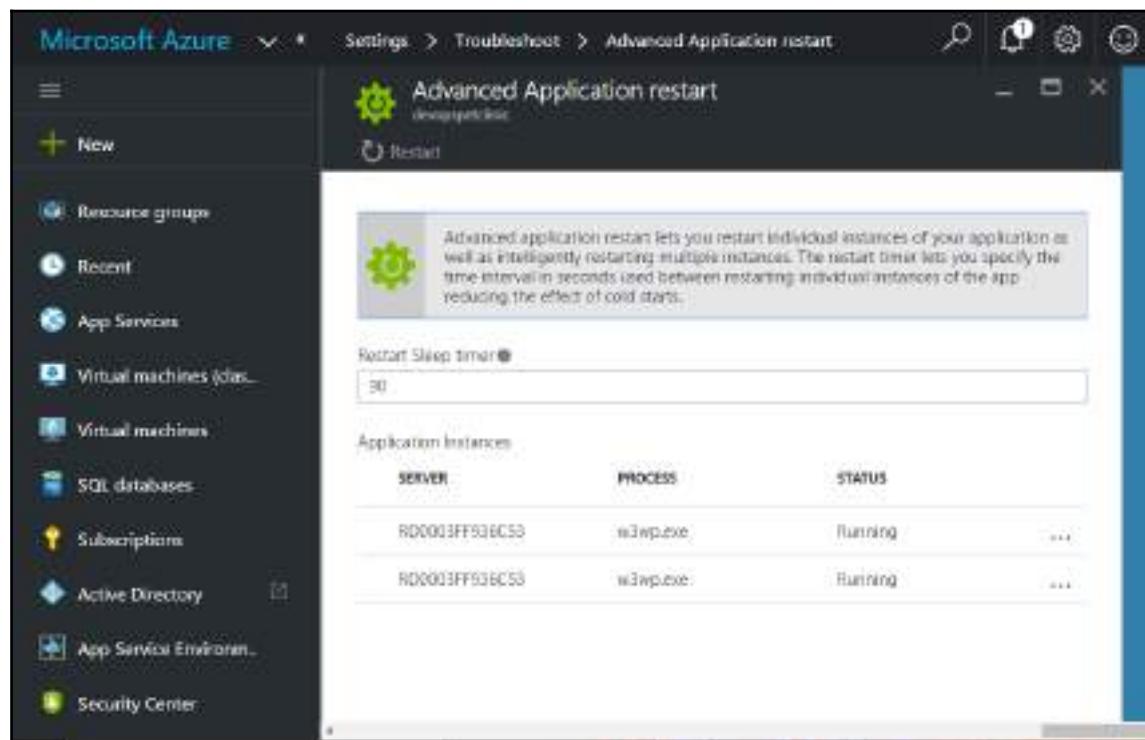
3. In Application, go to **Settings** and navigate to Features section. Click on **Diagnostics logs**. Here we can select **Application Logging (Filesystem)**, **Level**, **Application Logging (Blob)**, **Web server logging**, **Detailed error messages**, and **failed request tracing**:



4. In Application, go to **Settings** and navigate to **SUPPORT + TROUBLESHOOTING** section. Click on **Troubleshoot**, verify **RESOURCEHEALTH**. There are common solutions available here in case common problems such **5xx errors**:

The screenshot shows the Microsoft Azure portal interface for troubleshooting an App Service application named 'DevOpsWithCloud'. The left sidebar lists various Azure services like Resource groups, Storage, App Services, etc. The main area is titled 'Troubleshoot' under 'App Services > DevOpsWithCloud > Settings'. It displays the 'RESOURCE HEALTH' status as 'Available' (green) with a note: 'The Web app is running normally' (Last checked: 17/11/2018, 8:48:01 PM). Below this is a section for 'COMMON SOLUTIONS' with links: 'My App is returning HTTP 500 error', 'Verify Custom Domain Name configuration', 'My App is performing slowly', and 'How do I Setup SSL certificate?'. At the bottom, there's a 'NEED HELP?' link with the instruction: 'If you need assistance solving your issue, please open a support request'.

5. In case application is not accessible, we can restart the application or also do **Advanced Application restart**:



6. We can also verify activities performed on an application by filtering **Audit logs** from **Settings**:

The screenshot shows the Microsoft Azure Audit logs interface. The left sidebar lists various Azure services: Resource groups, Key Vault, App Services, Virtual machines (classic), Virtual machines, SQL database, Subscriptions, Active Directory, App Service Environments, and Security Center. The main pane is titled "Audit logs" and includes filters for Subscription, Resource group, Resource type, Operation type, and Event initiated by. A search bar at the top right contains the text "Logs (Last 24 hours) | Total log count: 10 | Log ingestion: [over] | Last log: [ ]". Below the filters is a table with columns: OPERATION NAME, STATUS, TIME, TIME STAMP, SUBSCRIPTION, and EVENT INITIATOR. One row is visible: "Update web site config" with STATUS "Succeeded", TIME "9 min ago", TIME STAMP "Sun, Jul 11, 2021", SUBSCRIPTION "Visual Studio Enterprise with MSDN", and EVENT INITIATOR "info@xyz.outlook.com".

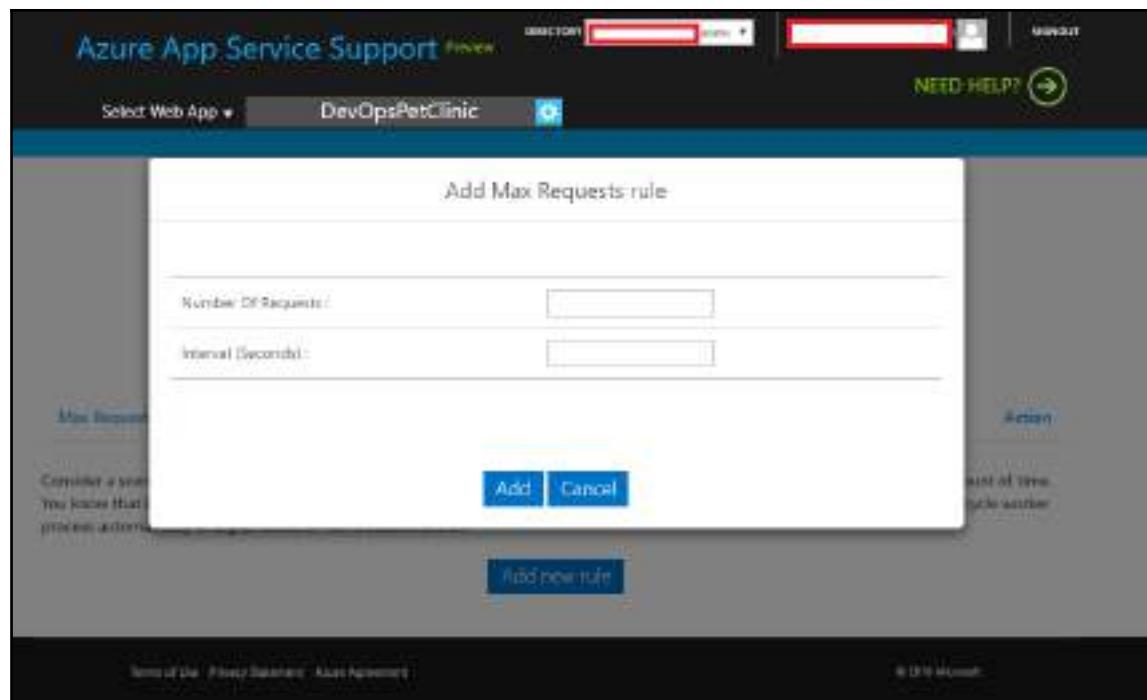
7. To automatically fix the issues in Azure Web App, go to **Settings** and navigate to **SUPPORT + TROUBLESHOOTING** section. Click on **Mitigate**:

The screenshot shows the Microsoft Azure Settings interface for an App Service named "DevelopersClinic". The left sidebar is identical to the previous screenshot. The main pane is titled "Settings" and lists several troubleshooting sections: SUPPORT + TROUBLESHOOTING, Troubleshoot, Activity logs, Resource health, Live HTTP traffic, Alerts, Diagnose & solve, Metrics per instance (App), Metrics per instance (App Service plan), and Mitigate. A modal window titled "Mitigate" is open on the right, with the sub-tittle "Create a rule to automatically restart and fix problems." It contains a "Create rule" button.

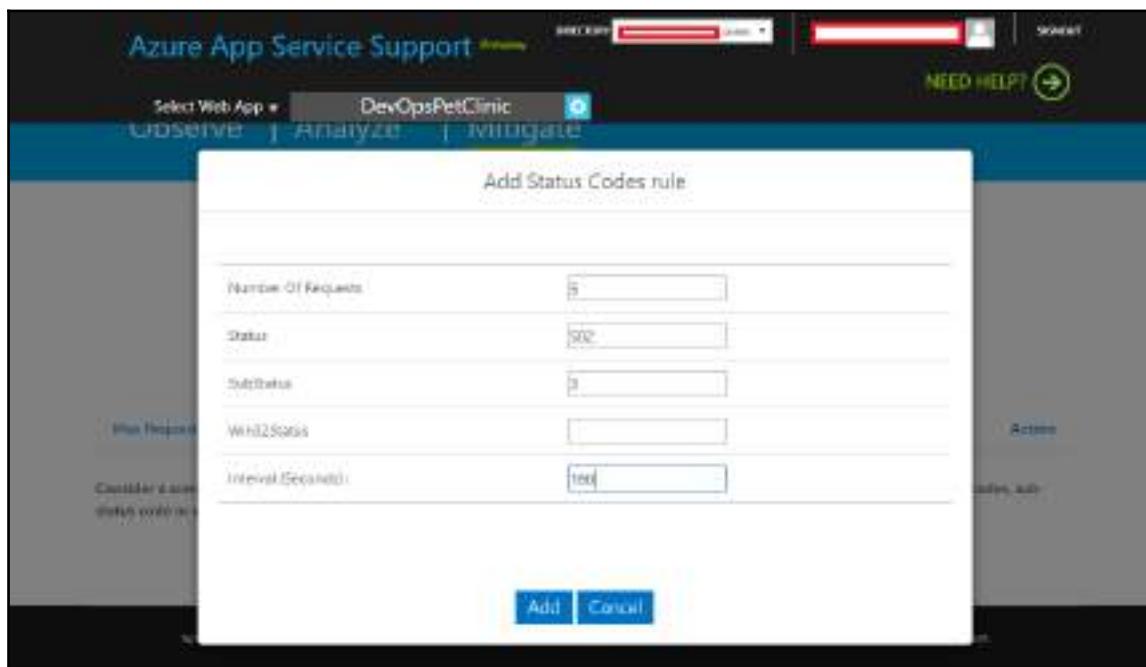
8. Select the Web App and enable **Autoheal**. Here we can configure settings to recover from application issues:

The screenshot shows the Azure App Service Support interface for the 'DevOpsPetClinic' web app. At the top, there are tabs for 'Observe', 'Analyze', and 'Mitigate'. The 'Mitigate' tab is currently selected. Below it, there is a large green button labeled 'Update' with a plus sign. Underneath the update button is a switch labeled 'Autoheal' which is set to 'ON'. At the bottom of the screen, there is a section titled 'Consider a scenario where you have a need to recycle your application automatically after it has served X number of requests in Y amount of time. You know that it just doesn't scale well after huge influx of requests in short amount of time. You want to detect this condition and recycle worker process automatically or log an event or run a custom action.' followed by a blue 'Add new rule' button.

9. We can configure Max Requests per specific Interval:



10. Status Code related monitoring. For example, when 5xx errors occur multiple times:



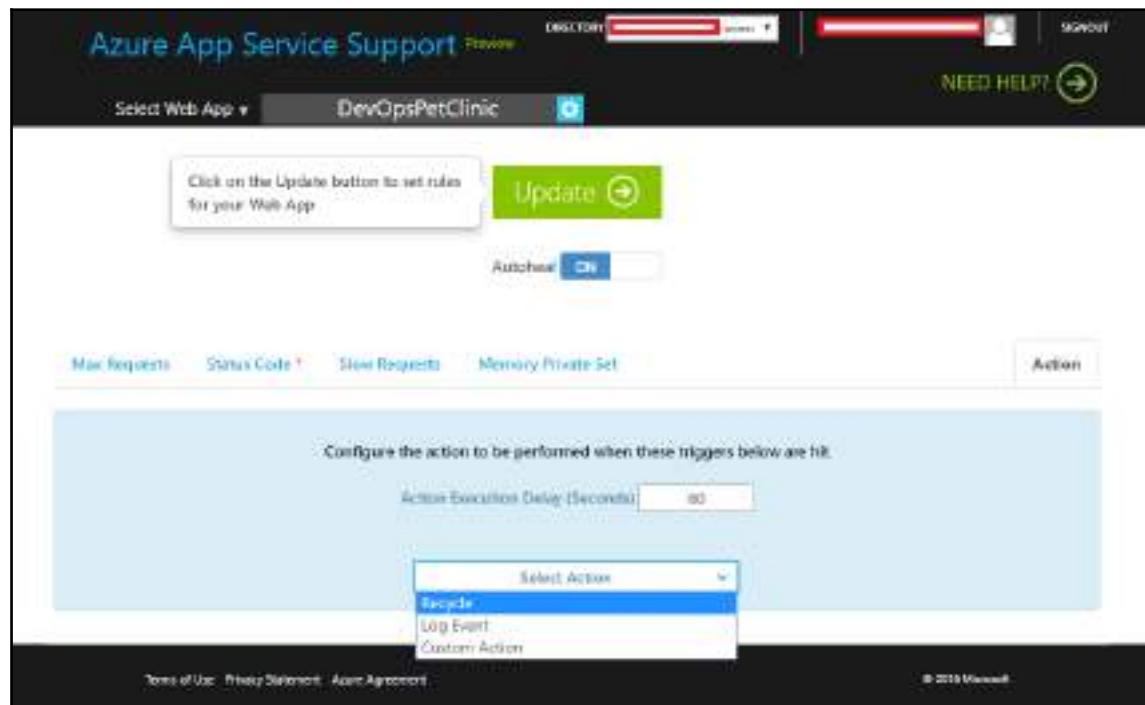
11. Verify newly added details of the Rule:

The screenshot shows the Azure App Service Support portal for the DevOpsPetClinic web app. At the top, there's a message: "Click on the Update button to set rules for your Web App." Below it is a green "Update" button with a plus sign. A toggle switch labeled "Autobuild" is set to "ON". To the right, a note says: "When you add new rule, check if action is set". At the bottom, there's a table with four rows: "Number Of Requests" (E), "Status" (502), "Substatus" (3), and "Win32Status" (0). An "Add new rule" button is located above the table.

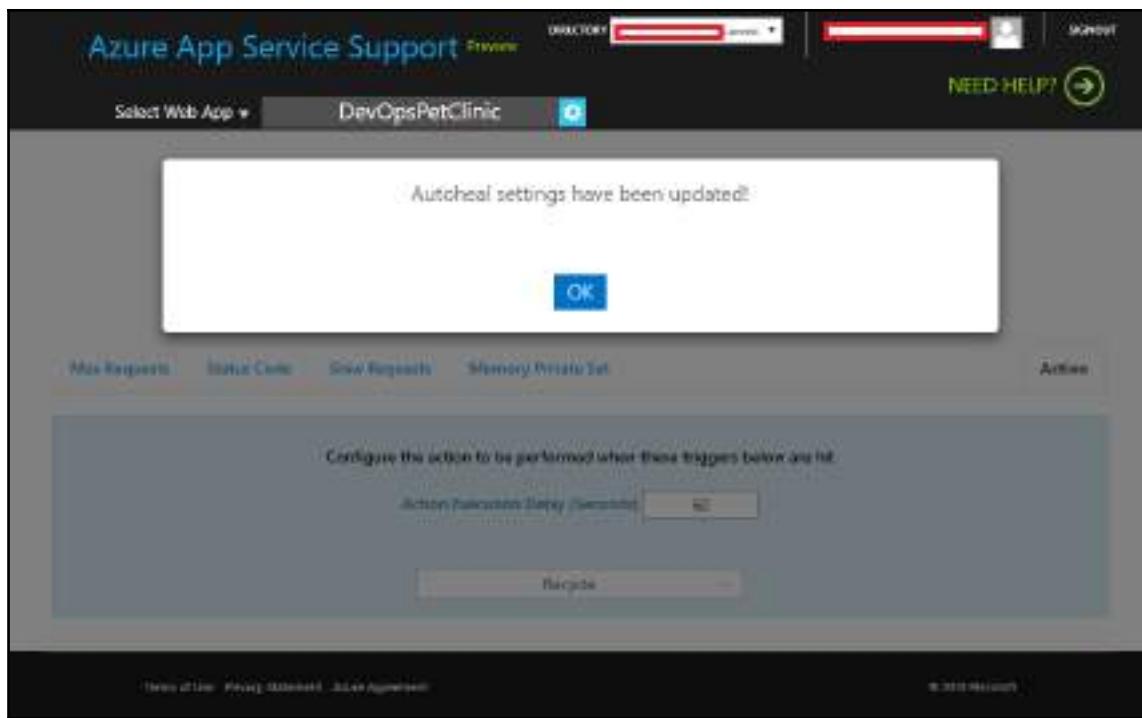
Number Of Requests	E
Status	502
Substatus	3
Win32Status	0

Once, everything is configured, click on **Action**. Here we can configure actions on the specific situation we have configured earlier. We can set recycle to auto recover from the issues.

1. Click on the **Update** button to set the rules for your web app:



2. You'll see the following message:



Go to KUDU Console by visiting  
[https://<application\\_name>.scm.azurewebsites.net/](https://<application_name>.scm.azurewebsites.net/). In our case, it will be <https://devopspetclinic.scm.azurewebsites.net>:

3. Click on **Debug console** and from drop down menu, select **CMD**. Click on **LogFiles**:

The screenshot shows the Kudu interface with the 'LogFiles' directory selected. The table lists the following files:

Name	Modified	Size
DetailedErrors	7/31/2016, 9:58:04 PM	
http	7/31/2016, 9:58:25 PM	
kudu	7/31/2016, 12:34:34 PM	
W3SVC13444584857	7/31/2016, 9:58:04 PM	
catalina.2016-07-31.log	7/31/2016, 3:58:46 PM	27 KB
eventlog.err	7/31/2016, 4:10:28 PM	9 KB
hostmanager.2016-07-31.log	7/31/2016, 12:34:37 PM	
... -		

At the bottom, there is a 'Kudu Remote Execution Console' section with the following text:

```
Type 'exit' then hit 'enter' to get a new CMD process.  
Type 'cls' to clear the console.
```

4. Go to **DetailedErrors** folder and verify what kind of errors have occurred:

The screenshot shows the Kudu interface with the 'DetailedErrors' directory selected. The table lists the following files:

Name	Modified	Size
ErrorPage000001.htm	7/31/2016, 3:57:49 PM	5 KB
ErrorPage000002.htm	7/31/2016, 3:57:51 PM	5 KB
ErrorPage000003.htm	7/31/2016, 3:57:50 PM	0 KB
ErrorPage000004.htm	7/31/2016, 3:57:53 PM	5 KB
ErrorPage000005.htm	7/31/2016, 3:58:04 PM	5 KB

At the bottom, there is a 'Kudu Remote Execution Console' section with the following text:

```
Type 'exit' then hit 'enter' to get a new CMD process.  
Type 'cls' to clear the console.
```

5. We can also Detailed Error logs from **FREB logs** in the portal itself:

FREE logs				
	Created	Verb	App Pool	
	Date ↴	URI		
	Created	URI		
	27/01/2018	https://1DevOpenPnPClient:8080/api/v1/Logs/GetAllEntries/EntryPage/100005.htm 11:18:23	GET	-1DevOpenPnPClient
	27/01/2018	https://1DevOpenPnPClient:8080/api/v1/Logs/GetAllEntries/102	DELETE	-1DevOpenPnPClient
	27/01/2018	https://1DevOpenPnPClient:8080	GET	DevOpenPnPClient
	27/01/2018	https://1DevOpenPnPClient:8080/api/v1/Logs/GetAllEntries/4-3-5-SAP/S4H/TwoDaysOldQuery- 11:34:14	GET	DevOpenPnPClient
	27/01/2018	https://1DevOpenPnPClient:8080/api/v1/Logs/GetAllEntries/4-3-5-SAP/S4H/TwoDaysOldQuery- 11:34:14	GET	DevOpenPnPClient

6. Open the Error Page in browser and we will get more details about the error:

The screenshot shows an error page with the following content:

**HTTP Error 502.3 - Bad Gateway**

**The specified CGI application encountered an error and the server terminated the process.**

**Most likely causes:**

- The CGI application did not return a valid set of HTTP errors.
- A server acting as a proxy or gateway was unable to process the request due to an error in a parent process.

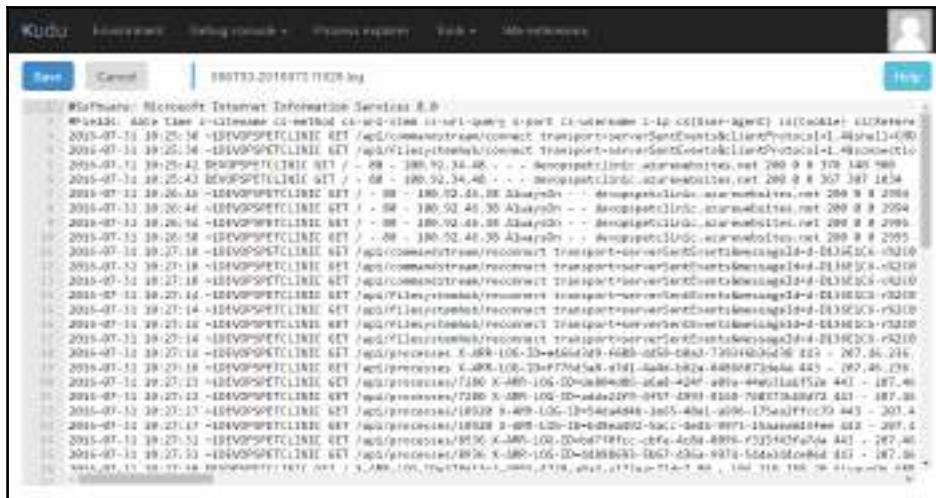
**Things you can try:**

- Use Debugging to troubleshoot the CGI application.
- Determine if a proxy or gateway is responsible for this error.

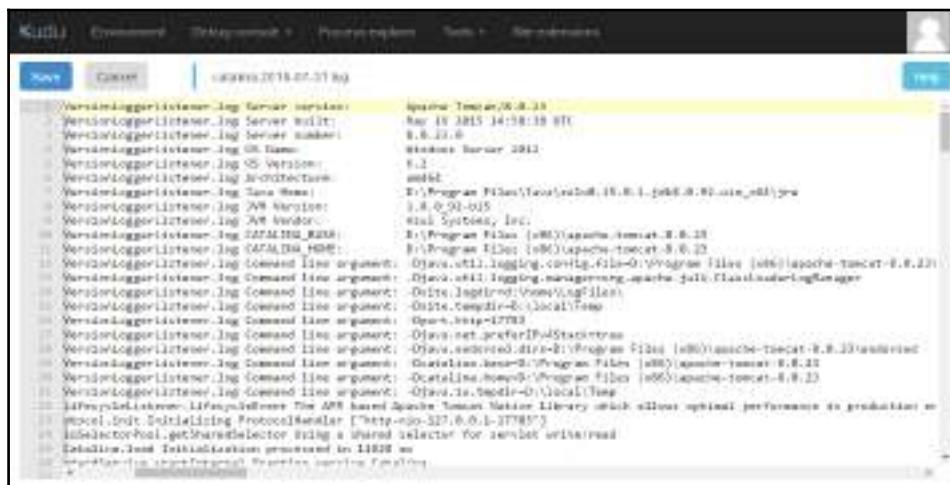
**Detailed Error Information:**

Module	httpPlatformHandler	Requested URL	http://localhost:4941/
Notification	endRequestHandler	Physical Path	D:\inetpub\wwwroot
Handler	httpPlatformModule	Logon Method	Anonymous
Error Code	0x80004012	Logon User	Anonymous

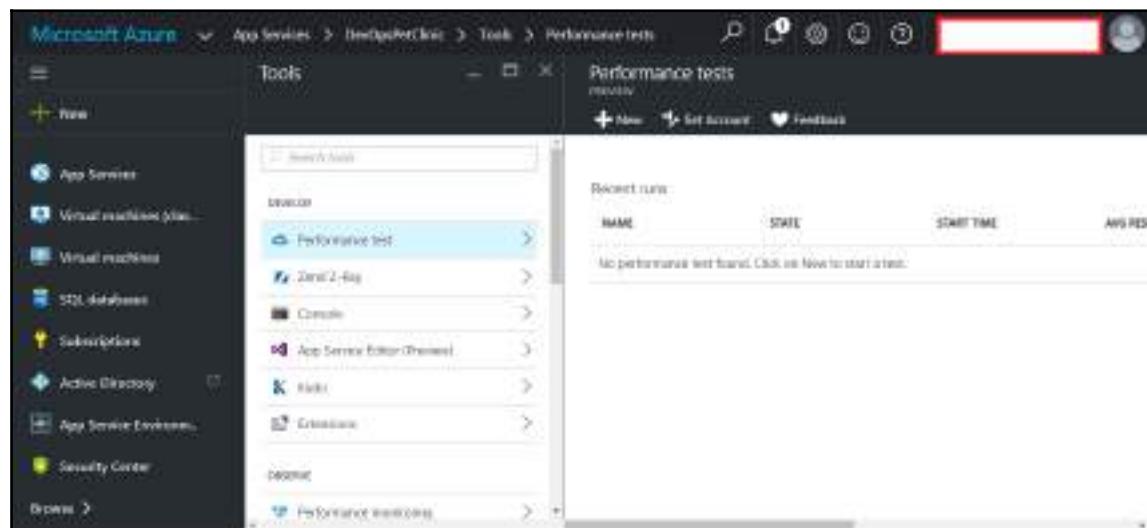
7. Go to **http | RawLogs | Open log files** to monitor all logs related to HTTP:



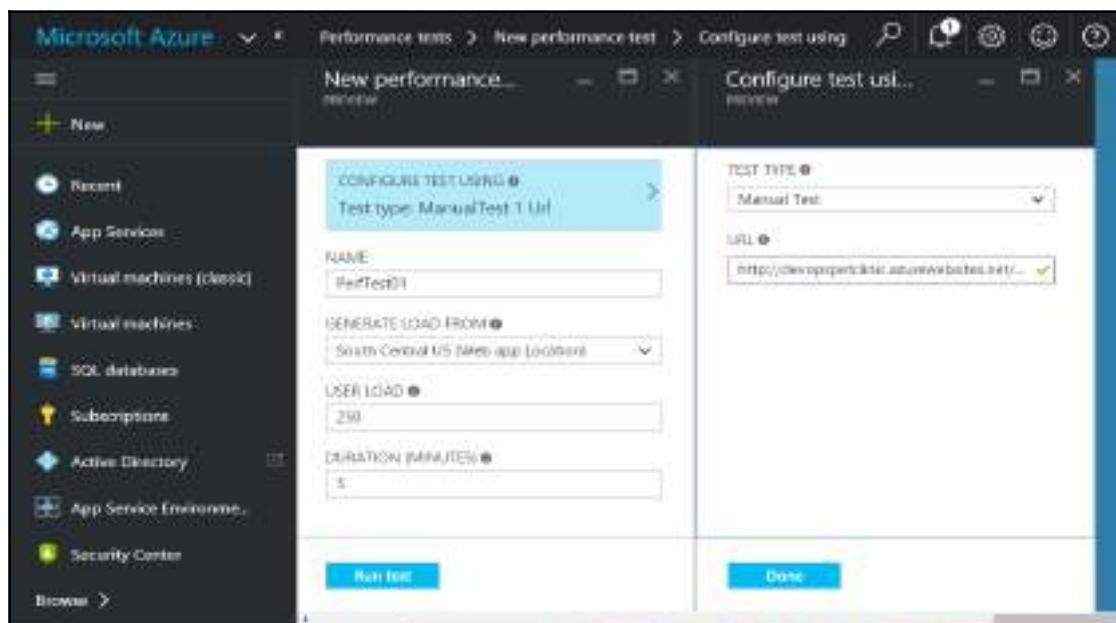
8. Catalina logs are also available and we can get all details about tomcat server and execution in it:



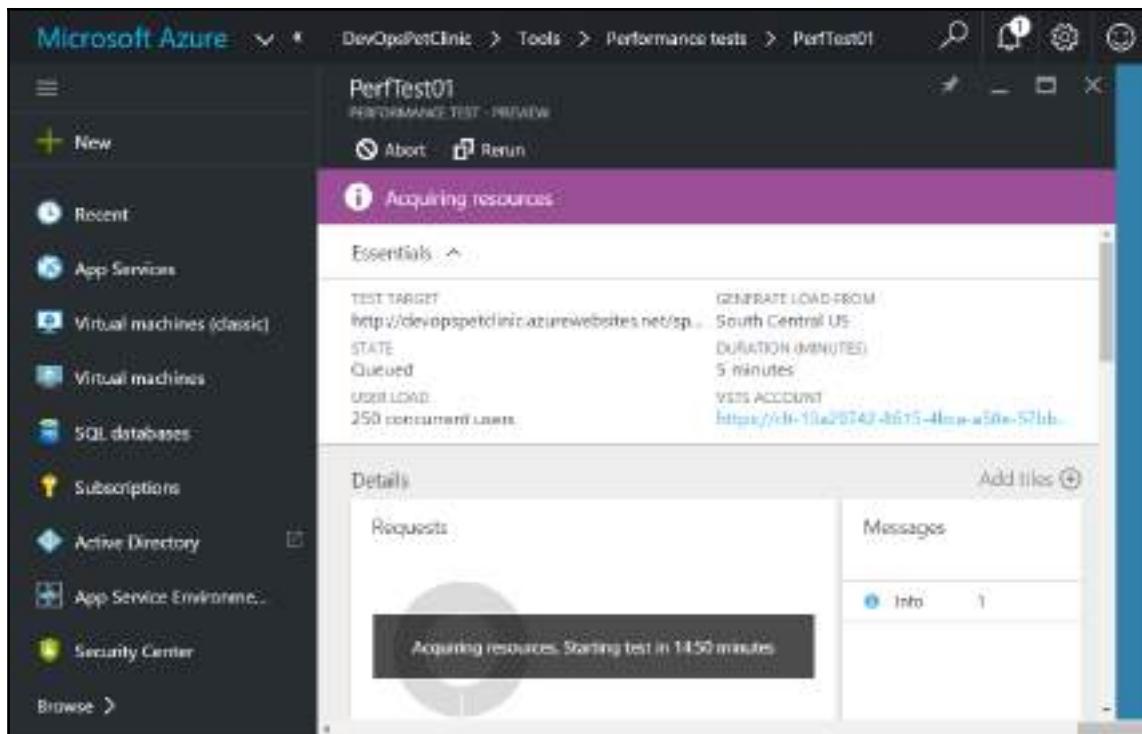
9. In Azure Web App, go to Application blade and click on **Tools**. In **DEVELOP** section, we get the feature to test the performance of an application. We need to have VSTS account for this feature. Click on **New**:



10. Configure test using **TEST TYPE**, **URL**, **USER LOAD**, Duration of Test:



11. It will take around 15 minutes to acquire resources:



12. Once performance test is over, verify the details in Azure Portal:

The screenshot shows the Azure Performance Test preview interface for a test named 'PerfTest01'. The top navigation bar includes 'App Services > DevOpsPetClinic > Tools > Performance tests > PerfTest01'. Below the title 'PerfTest01' and 'PERFORMANCE TEST - PREVIEW', there are two buttons: 'Abort' and 'Rerun'. The 'Essentials' section displays the following configuration details:

TEST TARGET	GENERATE LOAD FROM
<a href="http://devopspetclinic.azurewebsites.net">http://devopspetclinic.azurewebsites.net/sp...</a>	South Central US
STATE	DURATION (MINUTES)
Completed	5 minutes
USER LOAD	VSTS ACCOUNT
250 concurrent users	<a href="https://clt-13a29742-8615-4bca-a50e-57bb...">https://clt-13a29742-8615-4bca-a50e-57bb...</a>

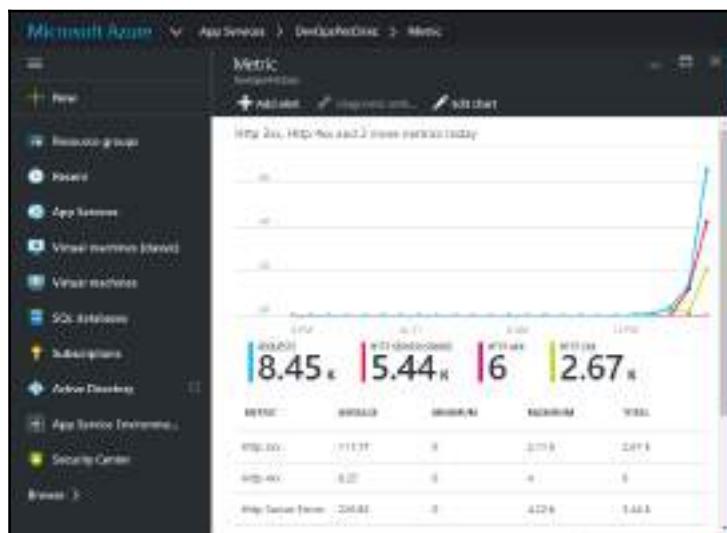
The 'Details' section contains a 'Requests' card with a donut chart showing the distribution of successful and failed requests. The chart indicates 2332 successful requests (30.35%) and 5351 failed requests (69.65%). To the right of the chart is a 'Messages' card showing 4 info messages.

13. Verify the Request Failures:

The screenshot shows the Microsoft Azure portal interface. The left sidebar navigation bar includes 'Portal', 'Resource groups', 'Record', 'App Services', 'Virtual machines', 'SQL databases', 'Table storage', 'Active Directory', 'App Service Environment', and 'Security Center'. The main content area is titled 'Performance' under 'App Services > Dev/TestWebApp > Performance' and shows 'Request Details'. A large circular chart indicates '2332' successful requests and '5351' failed requests. Below the chart, the text 'Performance counter load' is visible. To the right, a table titled 'Request Failures' lists various metrics: 'Request', 'Type', 'Status', and 'Count'. The table contains eight rows with the following data:

Request	Type	Status	Count
Health	HealthCheck	Success	1
Errors	BadRequestError	Success	1
HttpCode	HttpCode400	Success	1000
RequestMessage	Initial	Success	0
RequestMessage	Initial	Warning	0
RequestMessage	Warning	Success	0
RequestMessage	Warning	Warning	1

14. Go to Monitoring section of the Web App and check the recent results based on customized parameters:



15. Visual Studio Application Insights is in PREVIEW version and it has a capacity to Detect and diagnose issues in web apps and services. Click on **Browse** in Azure Portal and select **Application Insights**:

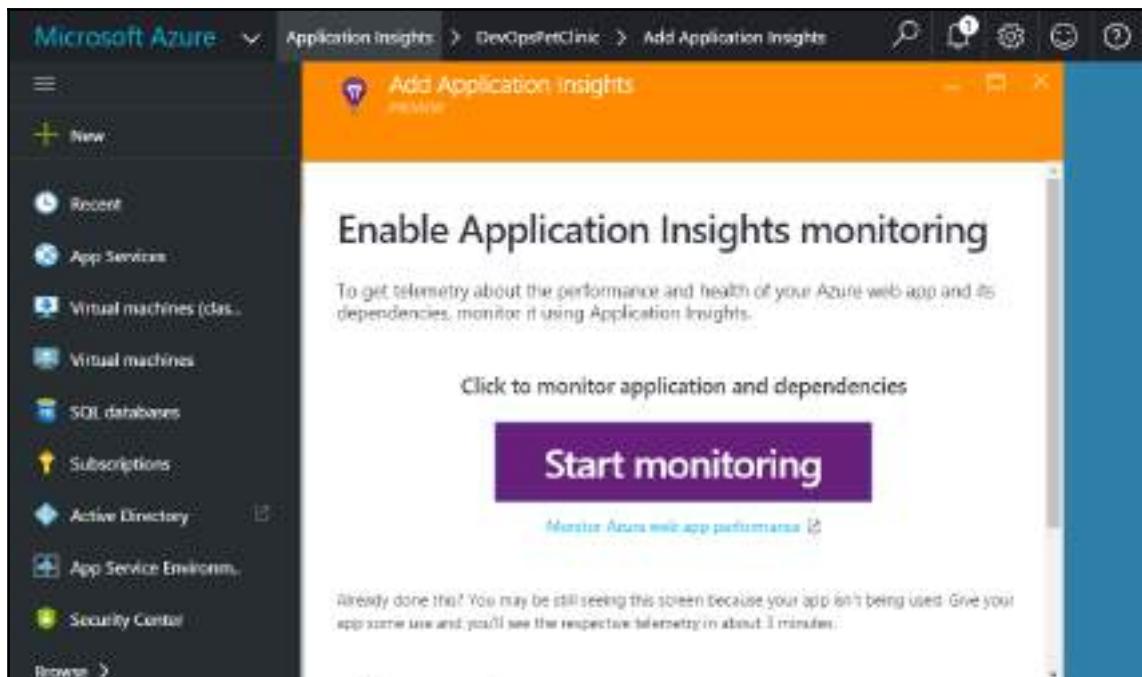
The screenshot shows the Microsoft Azure portal interface. On the left, there is a navigation sidebar with various service icons: Network, App Services, Virtual machines (classic), Virtual machines, SQL databases, Subscriptions, Active Directory, App Service Environments, and Security Center. Below this is a 'Known' section. The main content area is titled 'Application Insights' and displays a table of existing subscriptions:

NAME	RESOURCE GROUP	LOCATION	SIZE
DevOpsPetClinic	EnvOSS	Central US	W1
Suspetitas	EnvDF	Central US	W1

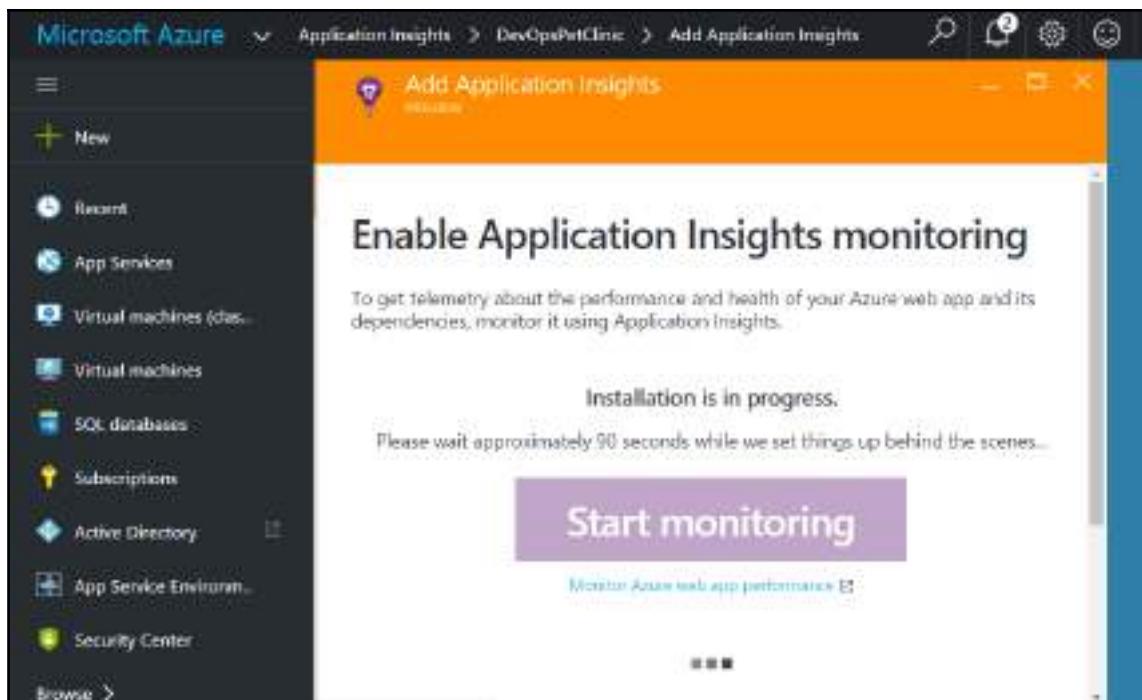
16. Select the application we have created and **Enable Application Insights** to start collecting telemetry:

The screenshot shows the Microsoft Azure portal interface, specifically the 'Settings and Diagnostics' blade for the 'DevOpsPetClinic' application insights instance. The left sidebar is identical to the previous screenshot. The main content area shows a summary card with the message 'Enable Application Insights to start collecting telemetry.' It includes sections for 'Essentials' (0 users, 0 proactive detection dashboards, 0 failed health checks, 0 app maps) and 'Application health' (Overview timeline). To the right, there is a sidebar with links for 'Ingestion', 'Support + Troubleshooting' (Activity logs, New support request), 'Instrumentation' (Application map, Log Metrics Stream, Patterns, Performance, Sensors, Browser), and a 'Known' section.

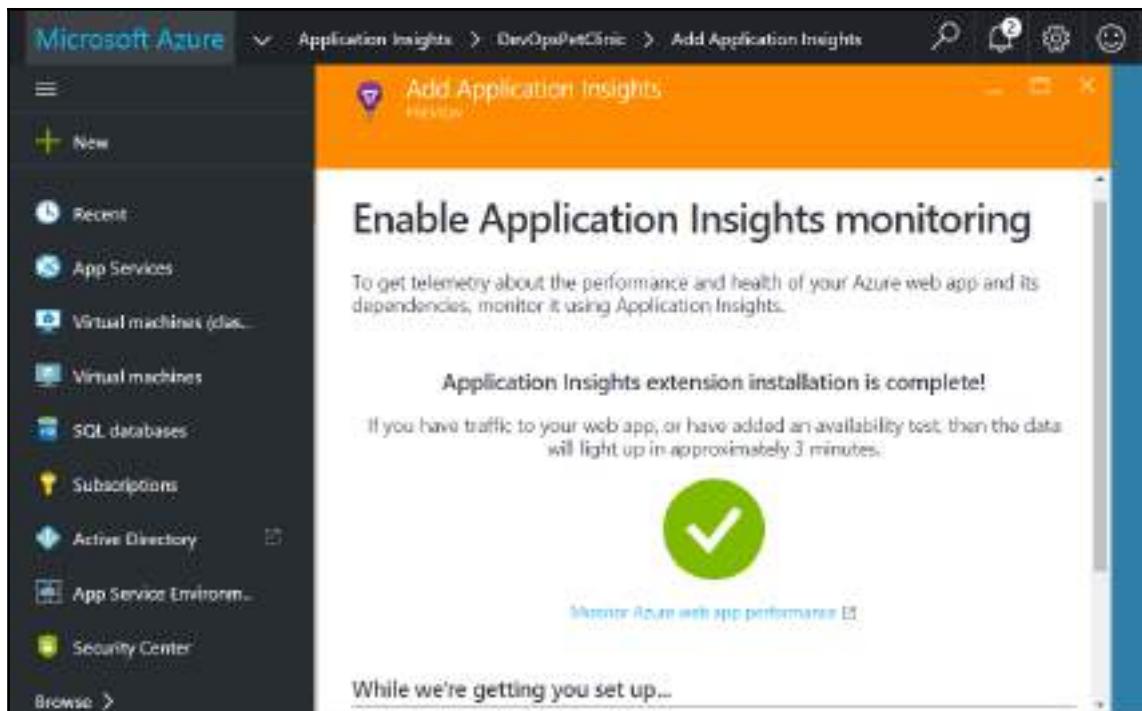
17. Click on **Start monitoring:**



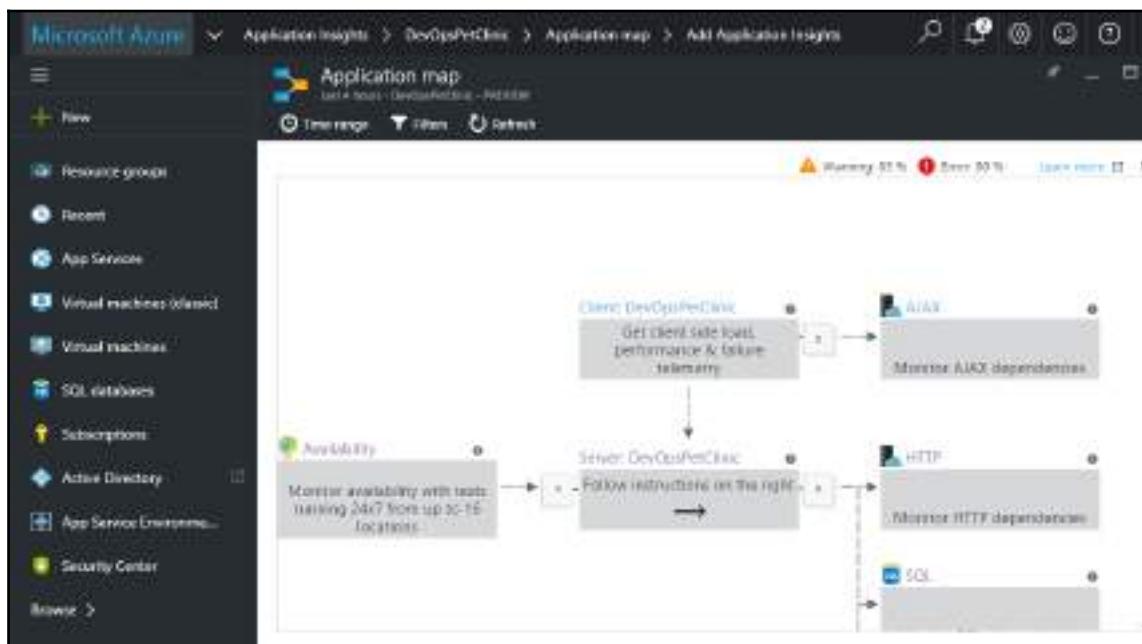
18. Wait till installation is completed.



19. Verify the successful installation in portal:



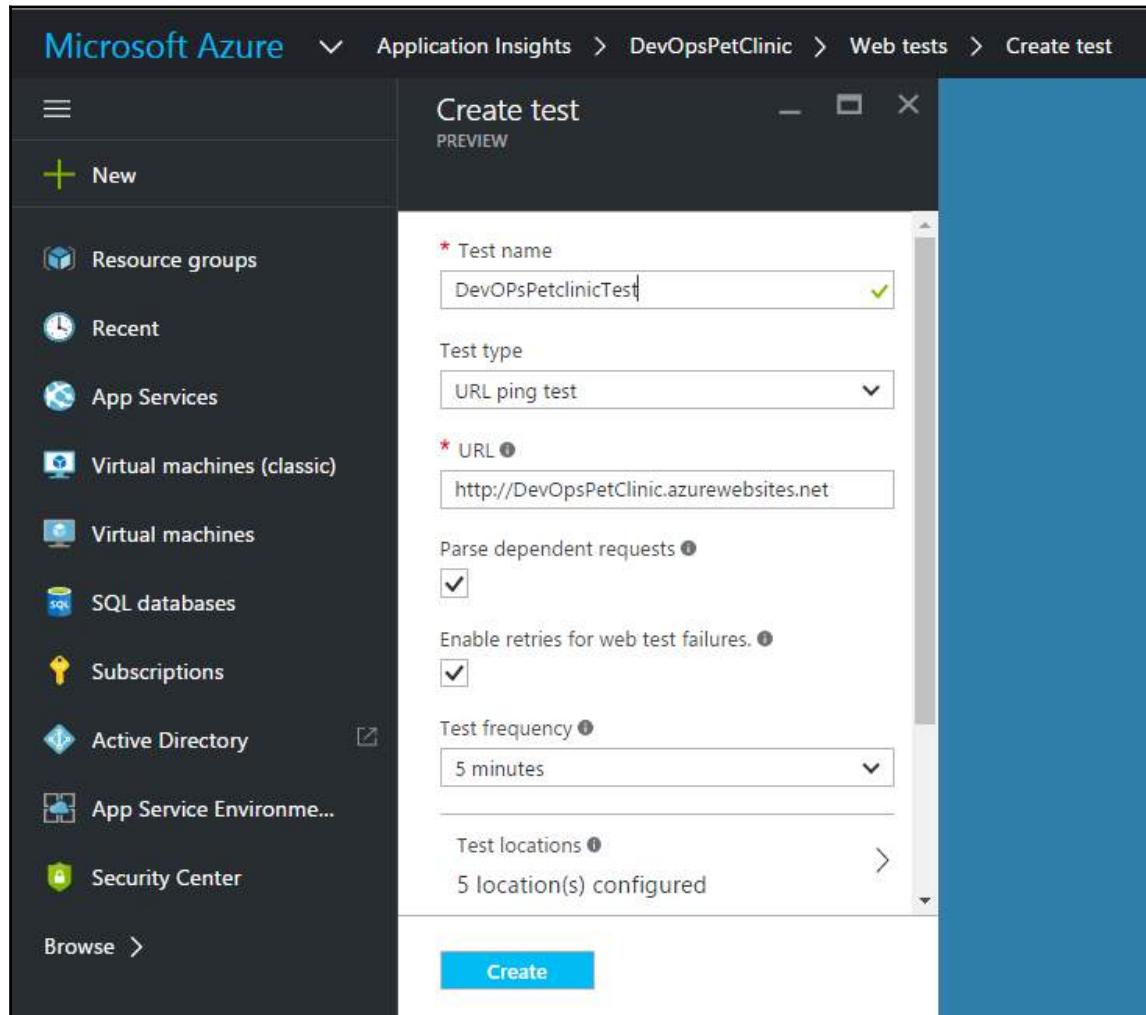
20. In **Application Insights**, we can also monitor**Application map** as shown in below figure:



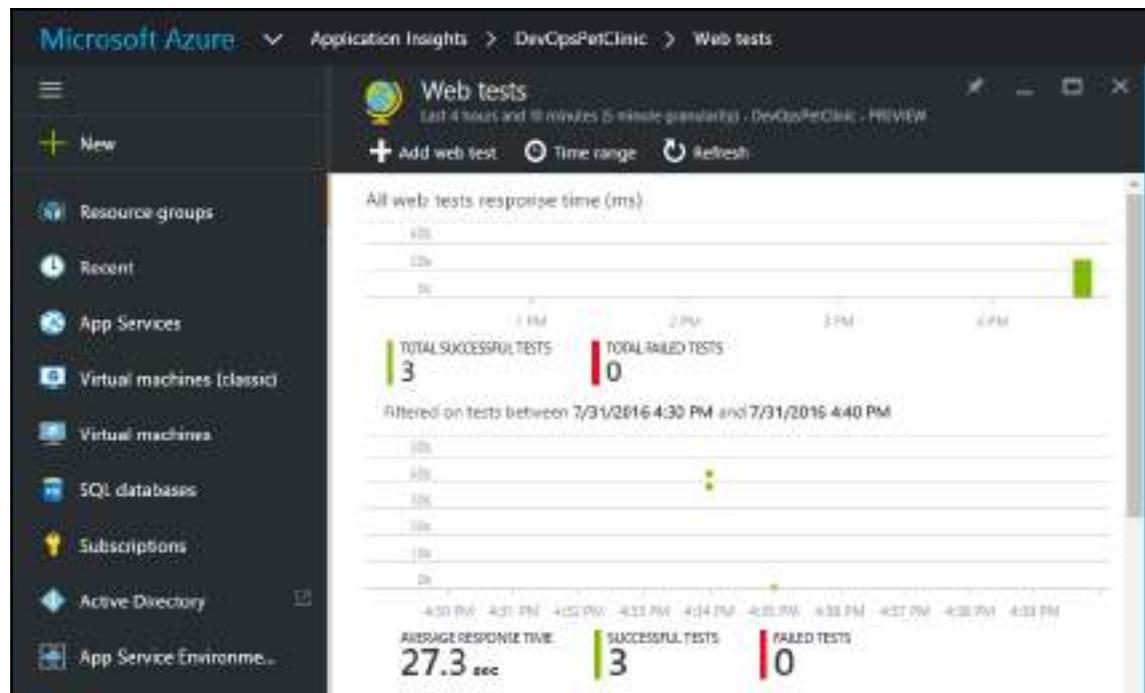
21. In Application Insights, We can configure Web tests to check availability of an application from different locations.

The screenshot shows the Microsoft Azure Application Insights interface. On the left, there's a sidebar with various service links: Resource groups, Recent, App Services, Virtual machines (classic), Virtual machines, SQL databases, Subscriptions, Active Directory, App Service Environments, and Security Center. Below that is a 'Browse' link. The main area is titled 'Web tests' and shows a chart for 'All web tests response time (ms)'. The chart has four time ranges: 1 MIN, 1 H, 1 D, and 1 W. A message box states 'There are no web test results in this time period.' At the bottom, there's a section for 'All web tests' with a 'WEB TEST' button and time range filters for 20 MIN, 1 H, 24 H, and 72 H.

22. We can configure Availability tests from different regions to verify whether it is available or not. Configure regions based on priority:



23. Create a Web test and within some time we will get results on Azure Portal:



We have covered most of the monitoring features available with Azure Web Apps. In the next section, we will cover how to use New Relic tool to monitor PetClinic Spring Application.

### Monitoring Web Application and Tomcat Server with New Relic

New Relic is a Software as a Service in the context of Cloud service models. New Relic monitors applications in real time and that also in any environment such as on premise or in cloud. We can install New Relic in the Application server's root directory and within minutes it starts providing monitoring and it reflects in the New Relic portal. It allows free trial.

New Relic supports monitoring for web applications developed in Java, .Net, PHP, Python, Node.js, Ruby and so on.

1. Create an account in New Relic. New Relic has Java installer for JBoss, Tomcat, Jetty, and Glassfish.
2. Login to New Relic.
3. Go to Account Settings and Download the Agent for the specific platform. In our case, we will install a Java Agent.
4. In the downloaded zip file, there will be two important files that are needed for monitoring:
  - `newrelic.jar`: that contains agent class files and
    - `newrelic.yml`: to configure license details that will be available on New Relic dashboard even for a free trial.
5. Extract the files from the zip folder and put the directory into Tomcat installation directory.
6. Open `newrelic.yml` in notepad and find the placeholder for License key and replace it with actual license key.
7. It will be in `common: &default_settings` as shown below:

```
common: &default_settings
# =====LICENSE KEY =====
# You must specify the license key associated with your New Relic
# account. For example, if your license key is 12345 use this:
# license_key: '12345'
# The key binds your Agent's data to your account in the New Relic service.
license_key: '12345'
```

8. Run Tomcat. Once Tomcat is up and running, open a command prompt and go to the Directory of New Relic in the Tomcat root directory.
9. Execute `java -jar newrelic.jar install` in the command prompt.

- We are trying to monitor an application that is deployed on a local environment in Tomcat 7. We can do similar installation and monitoring for virtual machine available in cloud or virtualized environment.

```
Microsoft Windows [Version 10.0.10586]
(c) 2015 Microsoft Corporation. All rights reserved.

C:\Users\Mitesh>cd \

C:\>cd apache-tomcat-7.0.70\newrelic

C:\apache-tomcat-7.0.70\newrelic>java -jar newrelic.jar install
***** ( ( o)) New Relic Java Agent Installer

***** Installing version 3.30.1 ...

* Backed up start script to C:\apache-tomcat-7.0.70\bin\catalina.bat.20160724_180719
* Added agent switch to start script C:\apache-tomcat-7.0.70\bin\catalina.bat

* No need to create New Relic configuration file because:
A config file already exists: C:\apache-tomcat-7.0.70\newrelic\newrelic.yml

***** Install successful

***** Next steps:
You're almost done! To see performance data for your app:

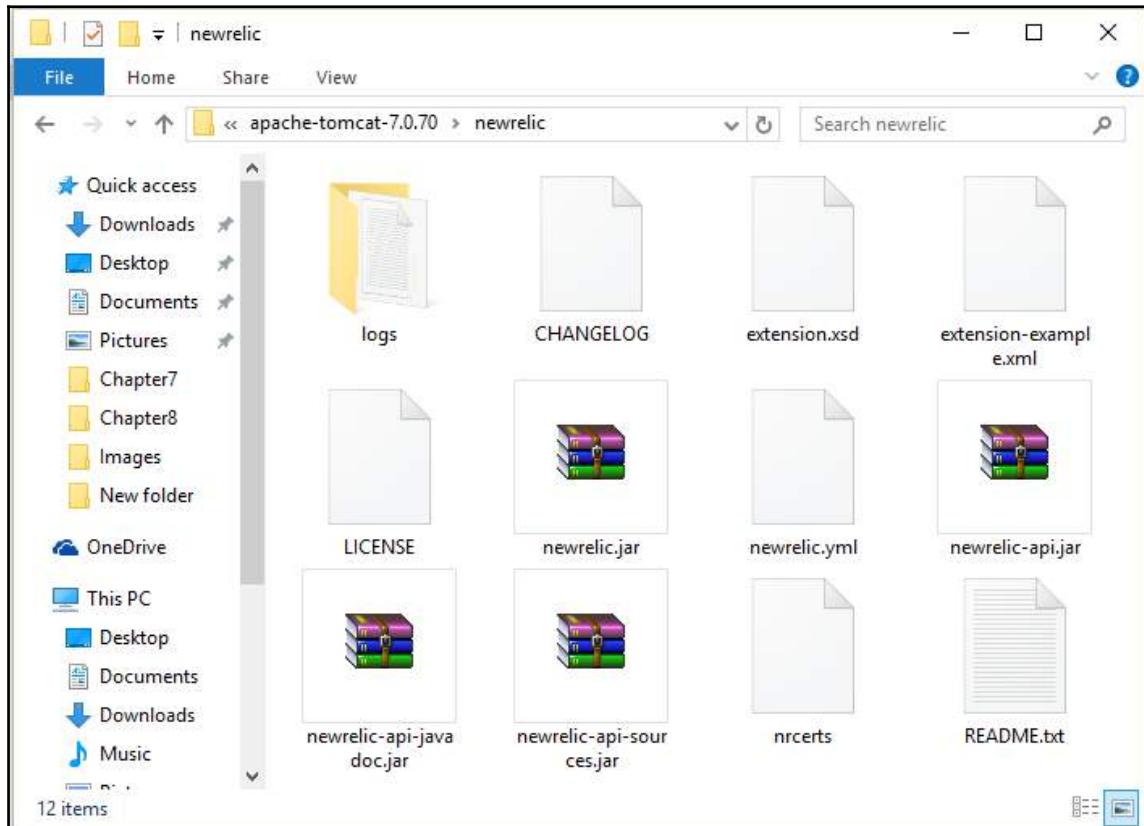
1) Restart your app server
2) Exercise your app
3) Log into http://rpm.newrelic.com

Within two minutes, your app should show up, ready to monitor and troubleshoot.
If app data doesn't appear, check newrelic/logs/newrelic_agent.log for errors.

C:\apache-tomcat-7.0.70\newrelic>
```

10. Once the installation is successful, restart the Tomcat server.

11. Verify the newrelic directory under Tomcat installation directory. New log folder will be created as shown in the following screenshot:



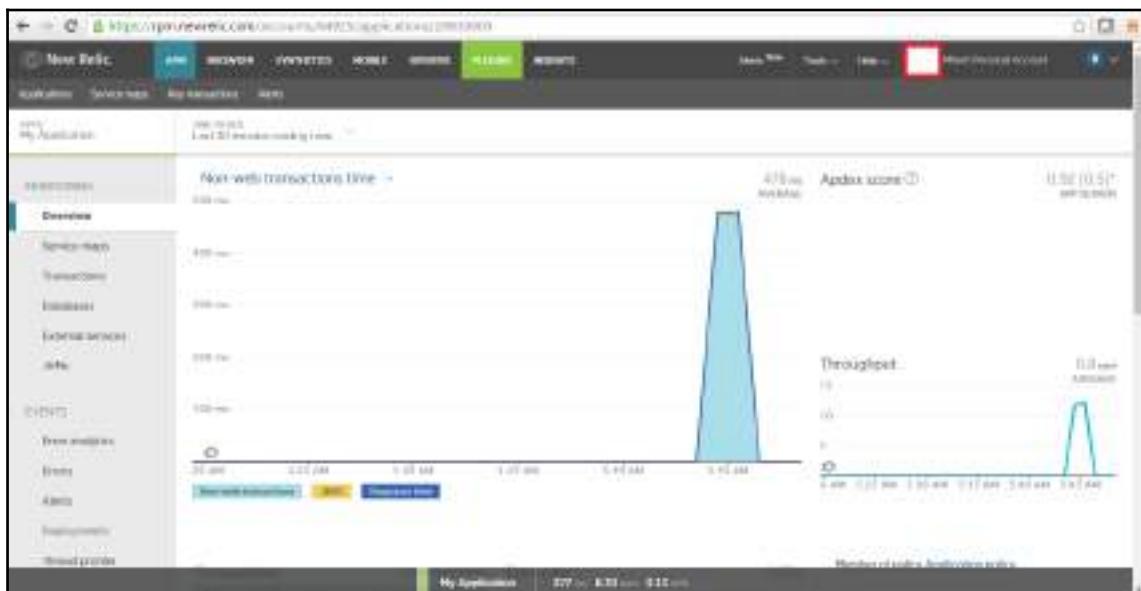
12. Open newrelic\_agent.log file in notepad.

13. Verify the reporting to line:

```
Jul 24, 2016 18:14:50 +0530 [6920 27] com.newrelic INFO: Reporting to:  
https://rpm.newrelic.com/accounts/64925/applications/20830005
```

```
Jul 24, 2016 18:14:51 +0530 [6920 7] com.newrelic INFO: Instrumentation  
com.newrelic.instrumentation.hibernate-2.5 is enabled. Loading.  
Jul 24, 2016 18:14:51 +0530 [6920 11] com.newrelic.agent.MetricServiceManagerImpl INFO: Configured  
to connect to New Relic at collector.newrelic.com:443  
Jul 24, 2016 18:14:52 +0530 [6920 1] com.newrelic INFO: Setting wadit mode to false  
Jul 24, 2016 18:14:52 +0530 [6920 11] com.newrelic INFO: Setting protocol to "https".  
Jul 24, 2016 18:14:52 +0530 [6920 11] com.newrelic.agent.config.ConfigServiceImpl INFO:  
Configuration file is C:\apache-tomcat-7.0.70\newrelic\newrelic.yml  
Jul 24, 2016 18:14:52 +0530 [6920 1] com.newrelic INFO: New Relic Agent v3.30.1 has started.  
Jul 24, 2016 18:14:52 +0530 [6920 1] com.newrelic INFO: Agent <class loader:  
sun.misc.Launcher$AppClassLoader@590644d4>  
Jul 24, 2016 18:14:52 +0530 [6920 1] com.newrelic INFO: Framework startup complete in 6,36ms.  
Jul 24, 2016 18:14:52 +0530 [6920 1] com.newrelic INFO: Server Info: Apache Tomcat/7.0.70  
Jul 24, 2016 18:14:54 +0530 [6920 27] com.newrelic INFO: Display host name is my-pc for  
application My Application  
Jul 24, 2016 18:14:49 +0530 [6920 27] com.newrelic INFO: Collector redirection to  
collector-216.newrelic.com:443  
Jul 24, 2016 18:14:50 +0530 [6920 27] com.newrelic INFO: Agent run id: 09464200173042674  
Jul 24, 2016 18:14:50 +0530 [6920 27] com.newrelic INFO: Agent 4920@my-pc/My Application  
connected to collector.newrelic.com:443  
Jul 24, 2016 18:14:50 +0530 [6920 27] com.newrelic INFO: Reporting to:  
https://rpm.newrelic.com/accounts/64925/applications/20830005  
Jul 24, 2016 18:14:50 +0530 [6920 27] com.newrelic INFO: Using RUM version 363 for application  
"My Application".  
Jul 24, 2016 18:14:50 +0530 [6920 27] com.newrelic INFO: Real user monitoring is enabled with  
auto instrumentation for application "My Application".
```

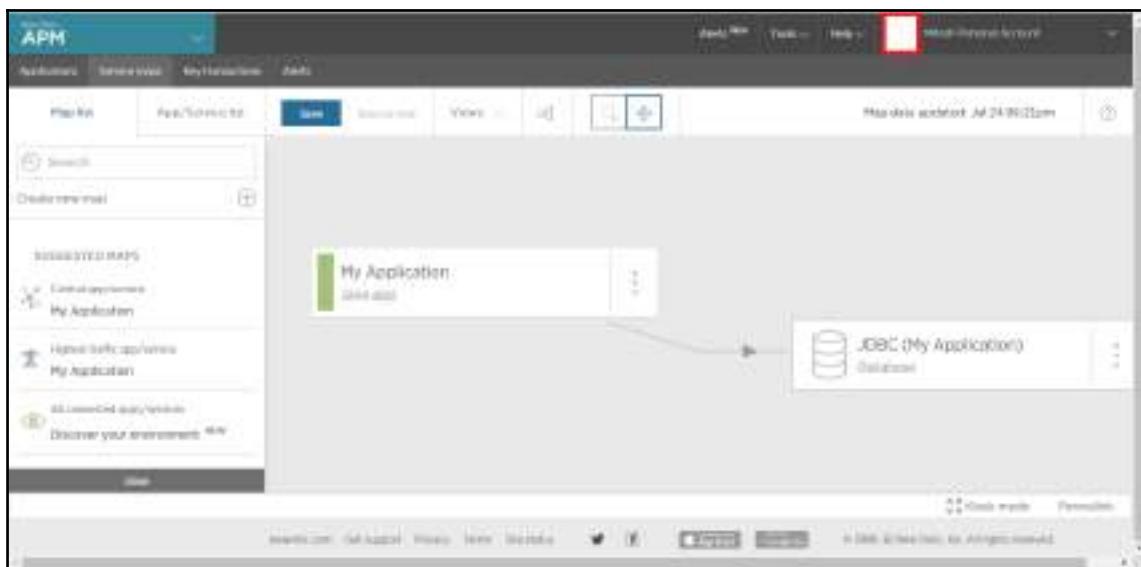
14. Copy the URL and open it in browser and observe the **Overview** section with graphs:



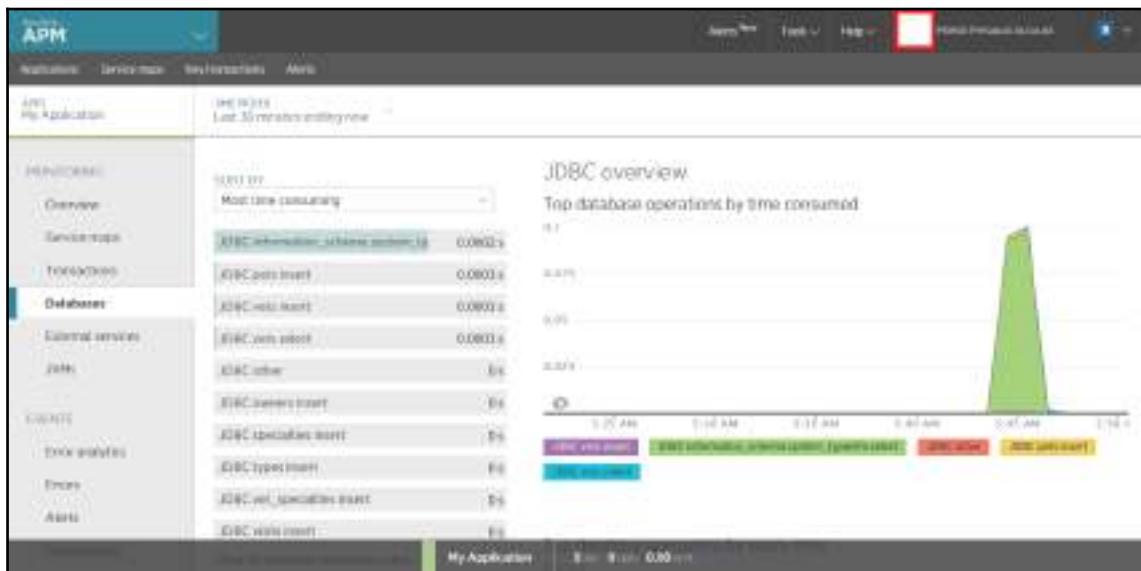
15. Click on the Applications link to get list of Applications:

The screenshot shows the APM application list interface. At the top, there's a navigation bar with tabs for 'APM', 'Metrics', 'Logs', 'Events', and 'Help'. Below the navigation is a search bar labeled 'Filter applications' and a 'Recent events' section with a message 'No Events In The Last 3 Days'. The main area displays a table titled 'Applications' with one entry: 'My Application'. The table has columns for 'Name', 'Container', 'Pod item', 'App service', 'Deployment', and 'Version'. The 'Name' column shows 'My Application' with a green status icon. The 'Version' column shows '0.0.1'. At the bottom of the table, it says '1 - 1 of 1'. The footer contains links for 'www.apm.com', 'Get Started', 'Watch', 'Learn', 'Metrics', and 'Logs'.

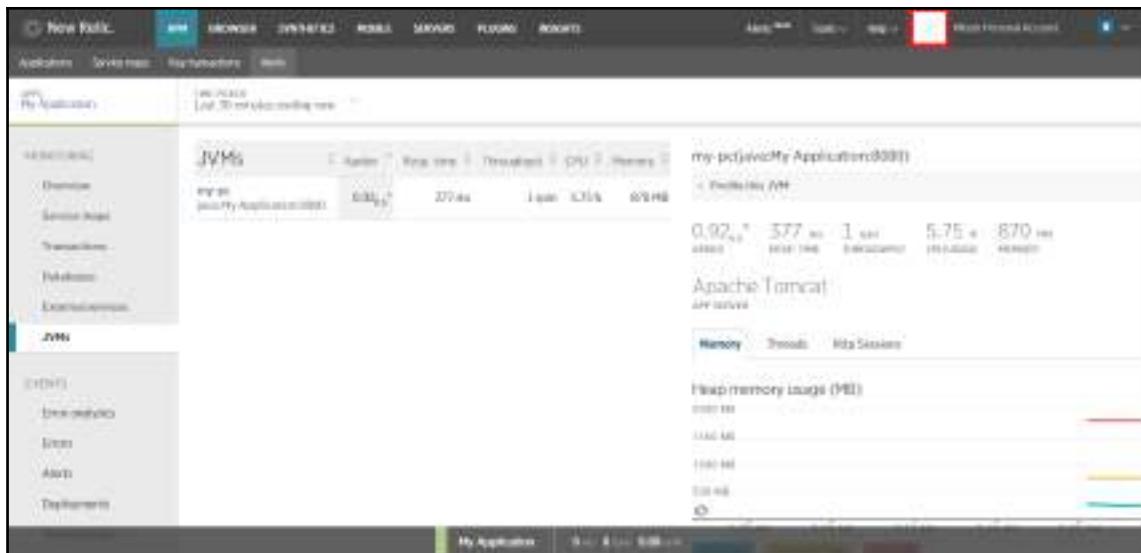
16. Service maps provides view of relationships between different components:



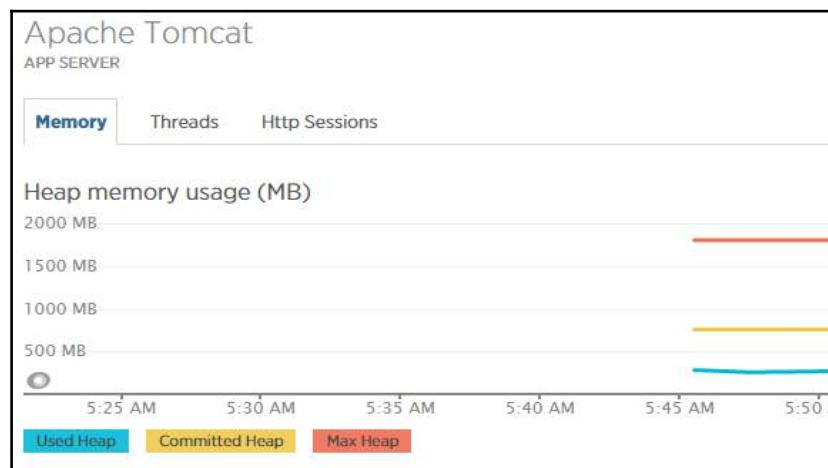
17. In the MONITORING Section of left sidebar, click on the **Databases** to get details of top database operations based on the time:



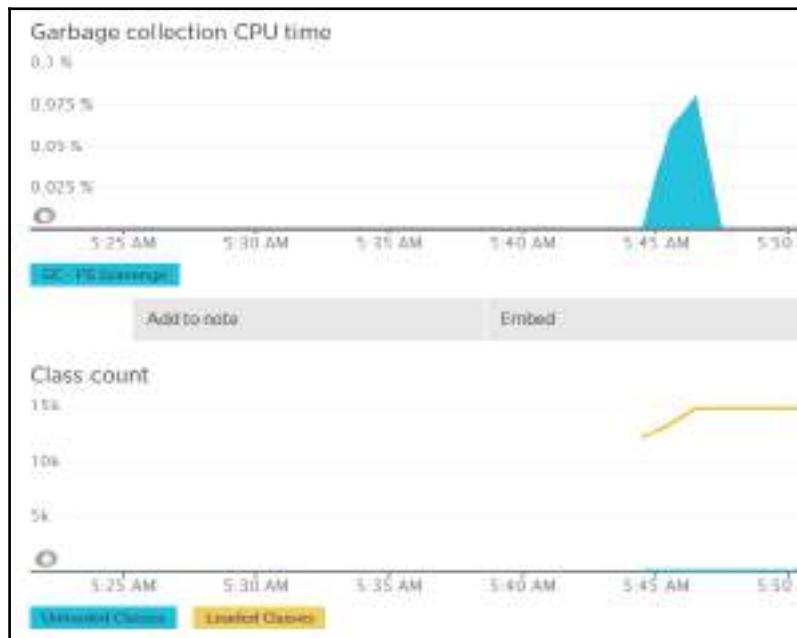
18. Click on **JVMs** to get details on the JVM and Tomcat:



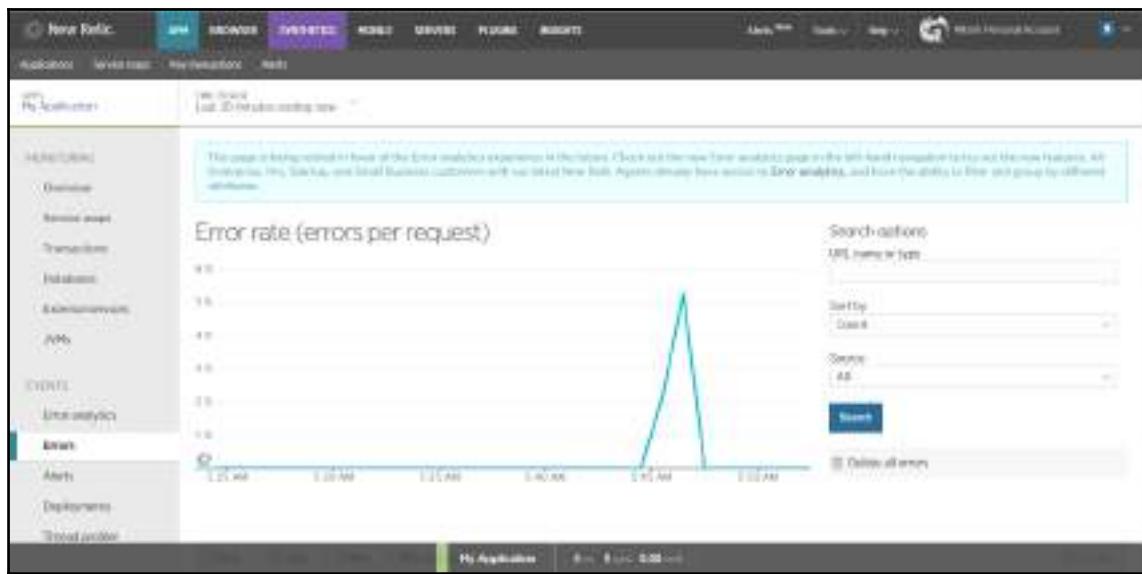
19. Verify the **Apache Tomcat** Section for details related to **Memory**, **Threads**, and **Http Sessions**:



20. In the same page Garbage Collection and Class count related details are also available:



21. Errors section will display graph of errors per request:

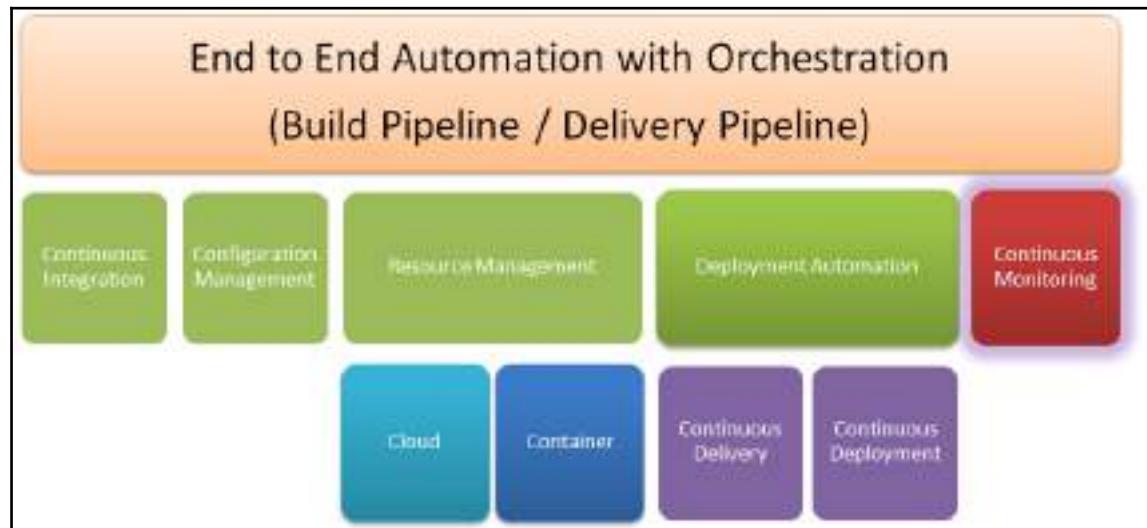


This is just a simple overview of New Relic for our PetClinic application deployed on premise.

So till now we have covered Overview of installation and configuration of Nagios; monitoring of AWS Elastic Beanstalk environment, monitoring of Azure Web Apps, and Application monitoring with New Relic.

Monitoring itself is a huge topic and to cover things in detail is out of scope of this book so we have only covered some portion to give a glimpse of Monitoring of resources.

We have covered five phases till now and now we will discuss about end to end automation with pipeline and orchestration:



## Self-test questions

State true or false:

1. Nagios Core, is an open source application written in C and PHP to monitor servers, networks, and infrastructure.
  - True
  - False
2. Nagios XI is supported in CentOS and RedHat.
  - True
  - False
3. In AWS Elastic Beanstalk, health status of an environment is determined by Grey, Green, Yellow, and Red color.
  - True
  - False

4. In AWS Elastic Beanstalk, Yellow color of health status indicates that environment has failure of one or more Health checks.
  - True
  - False
5. New Relic supports monitoring for web applications developed in Java, .Net, PHP, Python, Node.js, and Ruby
  - True
  - False

## **Summary**

In this chapter, we have seen quick overview of instance monitoring with Nagios, basic monitoring with AWS Elastic Beanstalk, Azure Web Apps monitoring, and Java web application monitoring with New Relic.

The constant monitoring of each event and interaction may look like very complex and not required but it is a need of an hour in the competitive environment where users are more demanding and hence availability of an application is extremely important.

In the next chapter we will see end to end automation with orchestration of activities such as Continuous Integration, Cloud Provisioning, Configuration Management, Continuous Delivery or Continuous Deployment.

# 9

# Orchestrating Application Deployment

*“Success is a lousy teacher. It seduces smart people into thinking they can’t lose. It’s fine to celebrate success but it is more important to heed the lessons of failure”*

This chapter describes in detail how to orchestrate different build jobs for continuous integration, configuration management, continuous delivery, and so on. It will cover how the build pipeline plugin and pipeline feature of Jenkins 2.0 can be used to orchestrate an end-to-end automation process for application deployment.

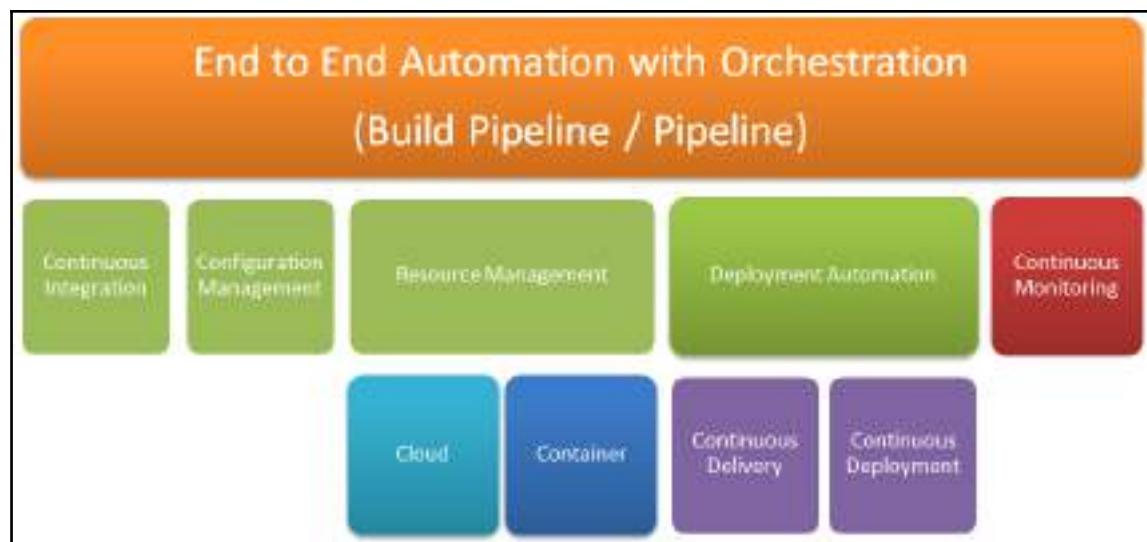
Until now, we have covered continuous integration, cloud provisioning using Chef, configuration management, and continuous delivery. Each was configured in a unique build job. Now we are only going to manage all those build jobs in a manner that the checkout or execution of the build pipeline will result in the checkout, compilation, unit test execution, installation of the Linux instance on Amazon EC2, installation of the runtime environment, configuration of permissions in the newly created instance, and deployment of the WAR file.

In this chapter, we will cover the following topics:

- Creating parameterized build jobs for end-to-end automation
- Configuring a build pipeline for the orchestration of a build job
- Executing the build pipeline for application deployment automation

# Creating build jobs for end-to-end automation

Before we configure end-to-end automation for build job execution, let's understand it graphically:



We will configure it using upstream and downstream job configuration in the case of the Build Pipeline plugin, while in the case of the Jenkins 2.0 pipeline, we will use a script.

Configuration management depends on the environment we use for deployment. We have covered the following deployment environments in this book:

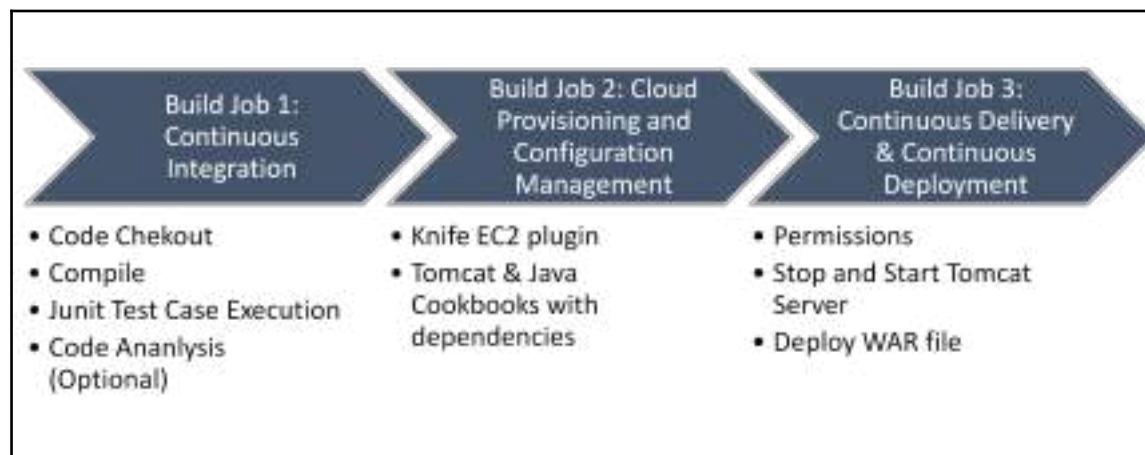
- PetClinic Spring application deployment on a Tomcat server (on-premise environment/personal laptop or desktop)
- PetClinic Spring application deployment on a Tomcat server on Amazon EC2 (IaaS)
- PetClinic Spring application deployment on a Tomcat server on a Microsoft Azure virtual machine (IaaS)

- PetClinic Spring application deployment on a Tomcat server in a Docker image (container)
- PetClinic Spring application deployment on a Tomcat server on Amazon Elastic Beanstalk (PaaS)
- PetClinic Spring application deployment on a Tomcat server in Microsoft Azure web apps (PaaS)

Based on the deployment environment, we need configuration management. In IaaS, we need to install a runtime environment, while in the case of PaaS and Docker containers, we only need minor modifications of the addition of a file or similar types of smaller changes.

Considering the deployment environment, we need to introduce build jobs for end-to-end automation.

In the case of the PetClinic Spring application deployment on a Tomcat server on Amazon EC2 (IaaS), we need the following flow:



Let's try to implement previous steps in Jenkins to achieve end-to-end automation that includes in the previously mentioned steps.

1. Let's visit the Jenkins dashboard. Click on **Manage Jenkins**. In the following screenshot, we can see that a new version of Jenkins is available. Click on **Or Upgrade Automatically** to update the application:

The screenshot shows the Jenkins Manage Jenkins page. On the left, there is a sidebar with links like 'New User', 'People', 'Build History', 'Manage Jenkins' (which is highlighted in blue), 'Create Jobs', and 'My Views'. Below that are sections for 'Build Queue' (empty) and 'Build Executor Status' (1 idle, 2 busy). The main content area has a heading 'Manage Jenkins'. It displays a message: 'A newer version of Jenkins (2.7.1) is available for download (checkboxed).'. There is a button 'Or Upgrade automatically' with a checkmark. Below this, it says 'Run the previous version of Jenkins' with a button 'Run Jenkins 2.7.0-1'. A list of available plugins is shown, each with a small icon and a brief description: 'Cloudbees Jenkins', 'Continuous Delivery', 'Configure Global Security', 'Global Tool Configuration', 'Hudson Surefire Plugin', 'Jenkins Shared Library', 'Manage Plugins', and 'System Information'.

2. It will start installing jenkins.war:

The screenshot shows the Jenkins 'Installing Plugins/Upgrades' page. The left sidebar has 'Manage Jenkins' selected. The main content area has a heading 'Installing Plugins/Upgrades'. It says 'Progress:' with three items: 'Checking network connectivity', 'Checking update center connectivity', and 'Resolving'. Below that, it says 'Status after: Upgrading'. There are two green bullet points: 'Go back to the homepage' (you can start using the installed plugin right away) and 'Percent: Jenkins server installation is complete and no jobs are running'. At the bottom, it says 'Page generated: 2020-07-20 10:30 AM PDT - 9522.2B - 192.168.1.20'.

- Once installation is successful, restart Jenkins from the terminal:

## Installing Plugins/Upgrades

Preparation

- Checking internet connectivity
- Checking update center connectivity
- Success

jenkins.war    Success

➔ [Go back to the top page](#)  
(you can start using the installed plugins right away)

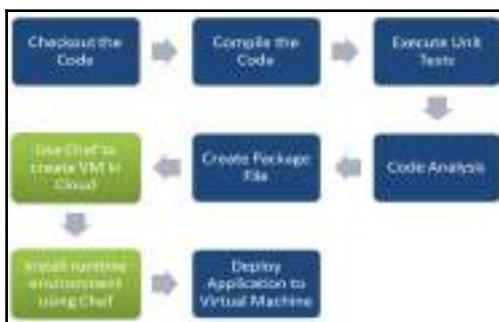
➔  Restart Jenkins when installation is complete and no jobs are running

- Refresh the Jenkins dashboard and check whether the new version has been installed properly:
- Verify the version number in the status bar:



# Configuring SSH authentication using a key

Before starting with end-to-end automation and orchestration, we need to configure SSH authentication using a key. The objective behind it is to allow the Jenkins VM to connect to the Chef workstation. Then, we can issue SSH commands from the Jenkins dashboard on the Chef workstation VM to create an instance in AWS or Azure cloud and install a runtime environment on it to deploy the PetClinic application:



If we try to access the SSH Chef workstation from Jenkins, it won't work as we still need to configure password less configuration for security:



Let's configure virtual machine where Jenkins is installed to access virtual machine where Chef Workstation is installed.

1. Open a terminal in Jenkins. Use ssh-keygen to create a new key:



```
root@devops1:~/Desktop
File Edit View Search Terminal Help
[root@devops1 Desktop]# ssh-keygen
Generating public/private rsa key pair.
Enter file in which to save the key (/root/.ssh/id_rsa):
/root/.ssh/id_rsa already exists.
Overwrite (y/n)? y
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /root/.ssh/id_rsa.
Your public key has been saved in /root/.ssh/id_rsa.pub.
The key fingerprint is:
11:c8:f7:98:7f:6a:24:00:9b:98:c4:e0:18:35 root@devops1
The key's randomart image is:
+---[ RSA 2048]---+
|          . |
|          + |
|          . 0 |
|          o |
|         S 0   |
|        - + |
|       .+ .+ |
|      .++ .+ |
|     .++ + |
|    .++ ++ |
+---+
[root@devops1 Desktop]#
```

2. Verify the newly generated key on the local filesystem:



3. Copy the key to the remote host using `ssh-copy-id`:

```
File Edit View Search Terminal Help
[root@devops1 Desktop]# ssh-copy-id -i ~/.ssh/id_rsa.pub 192.168.0.106
Agent admitted failure to sign using the key.
root@192.168.0.106's password:
Now try logging into the machine, with "ssh '192.168.0.106'", and check in:
.ssh/authorized_keys
to make sure we haven't added extra keys that you weren't expecting.

[root@devops1 Desktop]# ssh-copy-id -i ~/.ssh/id_rsa.pub mitesh@192.168.0.106
mitesh@192.168.0.106's password:
Now try logging into the machine, with "ssh 'mitesh@192.168.0.106'", and check in:
.ssh/authorized_keys
to make sure we haven't added extra keys that you weren't expecting.
```

4. Now try to access the Chef workstation using the Jenkins build job:



5. If it fails, then try to access it from the Jenkins VM using a terminal. If you get the Agent admitted failure to sign in using key message, then use ssh-add to fix the issue:

```
[mitesh@devops1 Desktop]$ ssh-copy-id -i ~/.ssh/id_rsa.pub root@192.168.0.103
root@192.168.0.103's password:
Now try logging into the machine, with "ssh 'root@192.168.0.103'", and check in:
  .ssh/authorized_keys

to make sure we haven't added extra keys that you weren't expecting.

[mitesh@devops1 Desktop]$ ssh -t root@192.168.0.103
Agent admitted failure to sign using the key.
root@192.168.0.103's password:

[mitesh@devops1 Desktop]$ ssh-add
Identity added: /home/mitesh/.ssh/id_rsa (/home/mitesh/.ssh/id_rsa)
[mitesh@devops1 Desktop]$ ssh -t root@192.168.0.103
Last login: Thu Jul 28 12:21:56 2016 from 192.168.0.106
[root@devops1 ~]# ifconfig
eth5      Link encap:Ethernet HWaddr 00:0C:29:91:3F:2F
          inet addr:192.168.0.103 Bcast:192.168.0.255 Mask:255.255.255.0
          inet6 addr: fe80::20c:29ff:fe91:3f2f/64 Scope:Link
            UP BROADCAST RUNNING MULTICAST MTU:1500 Metric:1
            RX packets:2664 errors:0 dropped:0 overruns:0 frame:0
            TX packets:1727 errors:0 dropped:0 overruns:0 carrier:0
            collisions:0 txqueuelen:1000
            RX bytes:716002 (699.2 KiB) TX bytes:197090 (192.4 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:50663 errors:0 dropped:0 overruns:0 frame:0
            TX packets:50663 errors:0 dropped:0 overruns:0 carrier:0
            collisions:0 txqueuelen:0
```

6. Now, our SSH connection is successful using not a password but a key:

7. Let's try to create an instance in AWS using the Jenkins build job and Chef workstation:

Jenkins > PetClinic-CloudProvisioning

**General**

Project name: PetClinic-CloudProvisioning

Description:

Discard old builds  
 Github project

8. Add a **Build** step, select **Execute shell**, and add paste the command mentioned here. We have already discussed `knife ec2` commands in Chapter 6, *Cloud Provisioning and Configuration Management with Chef*.

```
ssh -t -t root@192.168.1.36 "ifconfig; rvm use 2.1.0; knife ec2 server
create -I ami-1ecae776 -f t2.micro -N DevOpsVMonAWS1 --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]"
```



9. Click on **Save**. Click on the **Build Now** link to execute the build job.

10. Go to **Console Output** to check the progress:

A screenshot of a Jenkins job's console output. The title bar shows the Jenkins logo and the job name "PetClinic-CloudProvisioning". The left sidebar has links for "Back to Project", "Status", "Changes", "Console Output", "View as plain text", "Edit Build Information", and "Previous Build". The main area is titled "Console Output" and shows the following log entries:

```
Started by user DiscoverTechie
Building on master in workspace /home/ec2-user/.jenkins/workspace/PetClinic-CloudProvisioning
[PetClinic-CloudProvisioning] $ /bin/sh -e /tmp/jenkins49212136215942591.sh
+ ssh -t -t root@192.168.8.103 '$config; rm -rf .LLB; knife ec2 server ensure -I ami-locus707 -f
t2.micro -R us-west-2a -a ami-access-key-3d "[REDACTED]" --ami-secret-access-key
"[REDACTED]" --s3-bucket --identity-file .key.pem --ssh-user ec2-user
-r config-format'
register: Invalid argument
#055 Link encap:Ethernet Brdaddr:00:0C:29:81:97:1F
inet addr:192.168.8.103 Bcast:192.168.8.255 Mask:255.255.255.0
inet6 addr: fe80::20c:29ff:fe81:97ff/64 Scope:Link
UP BROADCAST NOFORWARD MTU:1500 Metric:1
RX packets:3379 errors:0 dropped:0 overruns:0 frame:0
TX packets:2376 errors:0 dropped:0 overruns:0 carrier:0
collisions:0 txqueuelen:1000
RX bytes:300000 (300.0 KB) TX bytes:272500 (272.5 KB)
```

11. AWS instance creation has started:

```
Jenkins -> PetClinic-CloudProvisioning -> #0  
  
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:69654 errors:0 dropped:0 overruns:0 frame:0  
TX packets:69654 errors:0 dropped:0 overruns:0 carrier:0  
collisions:0 txqueuelen:0  
RX bytes:31966874 (11.4 MiB) TX bytes:31966874 (11.4 MiB)  
  
[35m[{"id": "i-075fa6bd8e6af814", "image_id": "ami-0ccae776", "region": "us-east-1", "availability_zone": "us-east-1b", "security_groups": [{"name": "DevOpvWhizAHS"}], "tags": [{"key": "book"}]}, {"id": "i-075fa6bd8e6af814", "image_id": "ami-0ccae776", "region": "us-east-1", "availability_zone": "us-east-1b", "security_groups": [{"name": "DevOpvWhizAHS"}], "tags": [{"key": "book"}]}] Waiting for EC2 to create the instance[0m...  
[36mPublic DNS Name[0m: ec2-52-23-172-228.compute-1.amazonaws.com  
[36mPublic IP Address[0m: 52.23.172.228  
[36mPrivate DNS Name[0m: ig-172-31-56-252.ec2.internal  
[36mPrivate IP Address[0m: 172.31.56.252
```

## 12. Verify it in the AWS management console:

The screenshot shows the AWS EC2 Instances page. The left sidebar lists various EC2-related options like Instances, Spot Requests, Reserved Instances, etc. The main content area shows a table of instances. One instance, 'DevOpenVmwareAWS', is highlighted. Its details are shown in a modal window at the bottom right:

Description	Instance ID	InstanceState	Public IP
instance-state	i-0735e69c9e5d8014	running	ec2-43-23-172-223.compute-1.amazonaws.com
instance-type	t2.micro		52.21.152.228
private-dns	ip-172-01-56-252.ec2.internal		
private-ip	172.01.56.252		
secondary-private-ip			
VPC ID	vpc-640e1f62		

## 13. Before its execution can go further, check whether the AWS security group has an entry for SSH access:

The screenshot shows the AWS EC2 Instances page. The left sidebar lists various EC2-related options like Instances, Spot Requests, Reserved Instances, etc. The main content area shows a table of instances. One instance, 'DevOpenVmAWS1', is highlighted. Its details are shown in a modal window at the bottom right:

**Security Groups associated with i-082fac10306822ef4**

Port	Protocol	Source	Default
22	TCP	default	

14. Once SSH access is available, it will start Chef client installations:

```
[32mUsing /usr/local/rvm/gems/ruby-2.1.0[0m
[36mInstance ID[0m: i-024d3bf83022b89e4
[36mFlavor[0m: t2.micro
[36mImage[0m: ami-1ecae776
[36mRegion[0m: us-east-1
[36mAvailability Zone[0m: us-east-1d
[36mSecurity Groups[0m: default
[36mTags[0m: Name: DevOpsVMonAWS
[36mSSH Key[0m: book

[35mWaiting for EC2 to create the instance[0m.....
[36mPublic DNS Name[0m: ec2-52-23-215-193.compute-1.amazonaws.com
[36mPublic IP Address[0m: 52.23.215.193
[36mPrivate DNS Name[0m: ip-172-31-31-133.ec2.internal
[36mPrivate IP Address[0m: 172.31.31.133

[35mWaiting for sshd access to become available[0m.....done
Creating new client for DevOpsVMonAWS
Creating new node for DevOpsVMonAWS
Connecting to [1mecc2-52-23-215-193.compute-1.amazonaws.com[0m
[36mec2-52-23-215-193.compute-1.amazonaws.com[0m ----> Installing Chef Omnibus (-v 12)

[36mec2-52-23-215-193.compute-1.amazonaws.com[0m downloading https://omnitruck-direct.chef.io/chef/install.sh
[36mec2-52-23-215-193.compute-1.amazonaws.com[0m      to file /tmp/install.sh.2313/install.sh

[36mec2-52-23-215-193.compute-1.amazonaws.com[0m trying wget...
[36mec2-52-23-215-193.compute-1.amazonaws.com[0m el 6 x86_64
```

15. In our case, it will start downloading the Chef client and installing it on the AWS instance:

```
[36mecl2-52-23-215-193.compute-1.amazonaws.com]# On Getting information for chef stable 12 for x86_64...  
[36mecl2-52-23-215-193.compute-1.amazonaws.com]# In downloading https://omnitruck-direct.chef.io/stable/chef/metadata?x=12&p=x86_64  
[36mecl2-52-23-215-193.compute-1.amazonaws.com]# In trying wget...  
[36mecl2-52-23-215-193.compute-1.amazonaws.com]# In sha256  
9c6455bd38568c639e19485837baed97972c8e9f5cc3831fb4bc415bed24ed  
[36mecl2-52-23-215-193.compute-1.amazonaws.com]# In url https://packages.chef.io/stable/0/5/chef-12.12.15-x86_64.rpm  
[36mecl2-52-23-215-193.compute-1.amazonaws.com]# In version 12.12.15  
[36mecl2-52-23-215-193.compute-1.amazonaws.com]# In downloaded metadata file looks valid...  
[36mecl2-52-23-215-193.compute-1.amazonaws.com]# In downloading https://packages.chef.io/stable/0/5/chef-12.12.15-x86_64.rpm  
[36mecl2-52-23-215-193.compute-1.amazonaws.com]# In trying wget...  
[36mecl2-52-23-215-193.compute-1.amazonaws.com]# In Comparing checksum with sha256sum...  
[36mecl2-52-23-215-193.compute-1.amazonaws.com]# In Installing chef 12
```

16. Verify the Chef installation process on the console:

```
[36ec2-52-23-215-193.compute-1.amazonaws.com]#m Installing chef ii  
[36ec2-52-23-215-193.compute-1.amazonaws.com]#m installing with rpm...  
[36ec2-52-23-215-193.compute-1.amazonaws.com]#m warning: /tmp/install.sh.2318/chef-12.12.15-1.el6.x86_64.rpm: Header  
VI RSA/SHA1 Signature, key ID 83ef826a: NOKEY  
[36ec2-52-23-215-193.compute-1.amazonaws.com]#m Preparing...  
(100%)# (100%)## (100%)### (100%)#### (100%)##### (100%)#####  
(100%)### (100%)## (100%)### (100%)## (100%)## (100%)## (100%)##  
(100%)#### (100%)#### (100%)#### (100%)#### (100%)#### (100%)#### (100%)####  
(100%)#### (100%)#### (100%)#### (100%)#### (100%)#### (100%)#### (100%)####  
(100%)#### (100%)#### (100%)#### (100%)#### (100%)#### (100%)#### (100%)####  
(100%)#### (100%)#### (100%)#### (100%)#### (100%)#### (100%)#### (100%)####  
(100%)#### (100%)#### (100%)#### (100%)#### (100%)#### (100%)#### (100%)####  
(100%)#### (100%)#### (100%)#### (100%)#### (100%)#### (100%)#### (100%)####  
[36ec2-52-23-215-193.compute-1.amazonaws.com]#m updating / installing...  
[36ec2-52-23-215-193.compute-1.amazonaws.com]#m 1:chef-12.12.15-1.el6  
( 1%)# ( 4%)## ( 7%)### ( 10%)#### ( 13%)##### ( 16%)#####  
( 19%)#### ( 22%)##### ( 25%)##### ( 28%)##### ( 31%)##### ( 34%)##### ( 37%)#####  
( 40%)##### ( 43%)##### ( 46%)##### ( 49%)##### ( 52%)##### ( 55%)##### ( 58%)#####  
( 61%)##### ( 64%)##### ( 67%)##### ( 70%)##### ( 73%)##### ( 76%)##### ( 79%)#####  
( 82%)##### ( 85%)##### ( 88%)##### ( 91%)##### ( 94%)##### ( 97%)##### ( 100%)####
```

17. Once the Chef client is installed on the AWS instance, it will start its first Chef client execution.

18. Observe the run list and synchronizing cookbooks. It will converge and start installing packages:

```
[36mac2-52-13-215-193.compute-1.amazonaws.com]# Thank you for installing Chef!  
[36mac2-52-13-215-193.compute-1.amazonaws.com]# Starting the first Chef Client ran...  
[36mac2-52-13-215-193.compute-1.amazonaws.com]# Starting Chef Client, version 12.12.18  
[36mac2-52-13-215-193.compute-1.amazonaws.com]# resolving cookbooks for run list: ["tomcat"]  
[36mac2-52-13-215-193.compute-1.amazonaws.com]# Synchronizing Cookbooks:  
[36mac2-52-13-215-193.compute-1.amazonaws.com]#   - tomcat (0.17.0)  
[36mac2-52-13-215-193.compute-1.amazonaws.com]#   - java (1.30.0)  
[36mac2-52-13-215-193.compute-1.amazonaws.com]#   - apt (3.0.0)  
[36mac2-52-13-215-193.compute-1.amazonaws.com]#   - chef-sugar (2.2.0)  
[36mac2-52-13-215-193.compute-1.amazonaws.com]#   - openssl (4.4.0)  
[36mac2-52-13-215-193.compute-1.amazonaws.com]# Installing Cookbook Gems:  
[36mac2-52-13-215-193.compute-1.amazonaws.com]# Compiling Cookbooks...  
[36mac2-52-13-215-193.compute-1.amazonaws.com]# [2015-07-28T20:42:33+00:00] WARN: Chef::Provider::AptRepository already exists! Cannot create deprecation class for LWRP provider apt_repository from cookbook apt  
[36mac2-52-13-215-193.compute-1.amazonaws.com]# [2015-07-28T20:42:33+00:00] WARN: AptRepository already exists! Deprecation class overwrites Custom resource apt_repository from cookbook apt  
[36mac2-52-13-215-193.compute-1.amazonaws.com]# Converging 3 resources  
[36mac2-52-13-215-193.compute-1.amazonaws.com]# Recipe: tomcat::default  
[36mac2-52-13-215-193.compute-1.amazonaws.com]#   * yum_package[tomcat6] action install
```

19. Verify the package installations:

```
[36ec2-52-23-215-193.compute-1.amazonaws.com:0m Recipe: tomcat::default
[36ec2-52-23-215-193.compute-1.amazonaws.com:0m      * yum_package[tomcat6] action install
[36ec2-52-23-215-193.compute-1.amazonaws.com:0m      - install version 6.0.45-1.5.201101 of package tomcat6
[36ec2-52-23-215-193.compute-1.amazonaws.com:0m      * yum_package[tomcat6-admin-webapps] action install
[36ec2-52-23-215-193.compute-1.amazonaws.com:0m      - install version 6.0.45-1.5.201101 of package tomcat6-admin-
webapps
[36ec2-52-23-215-193.compute-1.amazonaws.com:0m      * tomcat_instance[base] action configure (up to date)
[36ec2-52-23-215-193.compute-1.amazonaws.com:0m      * directory[/usr/share/tomcat6/lib/endorsed] action create
[36ec2-52-23-215-193.compute-1.amazonaws.com:0m      - create new directory /usr/share/tomcat6/lib/endorsed
[36ec2-52-23-215-193.compute-1.amazonaws.com:0m      - change mode from '' to "0755"
[36ec2-52-23-215-193.compute-1.amazonaws.com:0m      * template[/etc/sysconfig/tomcat6] action create
[36ec2-52-23-215-193.compute-1.amazonaws.com:0m      - update content in File /etc/sysconfig/tomcat6 from 32gbal to
7ab378
[36ec2-52-23-215-193.compute-1.amazonaws.com:0m      --- /etc/sysconfig/tomcat6 2016-07-18 22:03:48.000000000 +0000
[36ec2-52-23-215-193.compute-1.amazonaws.com:0m      +++ /etc/sysconfig/.chef-tomcat620160718-2391-pszhdu 2016-
07-28 20:43:24.765025585 +0000
[36ec2-52-23-215-193.compute-1.amazonaws.com:0m      @@ -1,3 +1,9 @@
[36ec2-52-23-215-193.compute-1.amazonaws.com:0m      +#
[36ec2-52-23-215-193.compute-1.amazonaws.com:0m      ## Dynamically generated by Chef or sp-172-31-31-
133.ec2.internal
```

20. It will also display `conf.xml`, where port-related details can be verified based on the configuration:

```
[36ec2-52-23-215-193.compute-1.amazonaws.com:80] <!-- A "Connector" using the shared thread pool-->
[36ec2-52-23-215-193.compute-1.amazonaws.com:80] <!--
[36ec2-52-23-215-193.compute-1.amazonaws.com:80] <Connector executor="tomcatThreadPool"
[36ec2-52-23-215-193.compute-1.amazonaws.com:80]     port="8080" protocol="HTTP/1.1"
[36ec2-52-23-215-193.compute-1.amazonaws.com:80]     connectionTimeout="20000"
[36ec2-52-23-215-193.compute-1.amazonaws.com:80] +     port="8080" protocol="HTTP/1.1"
[36ec2-52-23-215-193.compute-1.amazonaws.com:80] +     connectionTimeout="20000"
[36ec2-52-23-215-193.compute-1.amazonaws.com:80]             redirectPort="8443" />
[36ec2-52-23-215-193.compute-1.amazonaws.com:80] -     -->
[36ec2-52-23-215-193.compute-1.amazonaws.com:80] +     -->
[36ec2-52-23-215-193.compute-1.amazonaws.com:80] <!-- Define a SSL HTTP/1.1 Connector on port 8443
[36ec2-52-23-215-193.compute-1.amazonaws.com:80] -     this connector uses the JSSE configuration, when using
[APR, the
[36ec2-52-23-215-193.compute-1.amazonaws.com:80] +     This connector uses the JSSE configuration, when using
[APR, the
[36ec2-52-23-215-193.compute-1.amazonaws.com:80]             connector should be using the OpenSSL style
[36ec2-52-23-215-193.compute-1.amazonaws.com:80]             described in the APR documentation -->
```

21. Once the package installation is finished, it will start service management:

```
[36mec2-52-23-215-193.compute-1.amazonaws.com[0m      * service[tomcat6] action start
[36mec2-52-23-215-193.compute-1.amazonaws.com[0m      - start service service[tomcat6]
[36mec2-52-23-215-193.compute-1.amazonaws.com[0m      * execute[wait for tomcat6] action run
[36mec2-52-23-215-193.compute-1.amazonaws.com[0m      - execute sleep 5
[36mec2-52-23-215-193.compute-1.amazonaws.com[0m      * service[tomcat6] action enable
[36mec2-52-23-215-193.compute-1.amazonaws.com[0m      - enable service service[tomcat6]
[36mec2-52-23-215-193.compute-1.amazonaws.com[0m      * execute[wait for tomcat6] action run
[36mec2-52-23-215-193.compute-1.amazonaws.com[0m      - execute sleep 5
[36mec2-52-23-215-193.compute-1.amazonaws.com[0m      * execute[wait for tomcat6] action nothing
:nothing)
[36mec2-52-23-215-193.compute-1.amazonaws.com[0m      * service[tomcat6] action restart
[36mec2-52-23-215-193.compute-1.amazonaws.com[0m      - restart service service[tomcat6]
[36mec2-52-23-215-193.compute-1.amazonaws.com[0m      * execute[wait for tomcat6] action run
[36mec2-52-23-215-193.compute-1.amazonaws.com[0m      - execute sleep 5
[36mec2-52-23-215-193.compute-1.amazonaws.com[0m
[36mec2-52-23-215-193.compute-1.amazonaws.com[0m Running handlers:
[36mec2-52-23-215-193.compute-1.amazonaws.com[0m Running handlers complete
```

22. Now, the Chef client execution has finished, and it will display related information for the AWS instance we created:

```
[36mec2-52-23-215-193.compute-1.amazonaws.com[0m

[36mec2-52-23-215-193.compute-1.amazonaws.com[0m Chef Client finished, 13/15 resources
seconds

[36mInstance ID[0m: i-024d3bf83022b89e4
[36mFlavor[0m: t2.micro
[36mImage[0m: ami-1ecae776
[36mRegion[0m: us-east-1
[36mAvailability Zone[0m: us-east-1d
[36mSecurity Groups[0m: default
[36mSecurity Group IDs[0m: default
[36mTags[0m: Name: DevOpsVMonAWS
[36mSSH Key[0m: book
[36mRoot Device Type[0m: ebs
[36mRoot Volume ID[0m: vol-00aae3951d7ed88bb
[36mRoot Device Name[0m: /dev/xvda
[36mRoot Device Delete on Terminate[0m: true

[35mBlock devices[0m
[35m=====
[36mDevice Name[0m: /dev/xvda
[36mVolume ID[0m: vol-00aae3951d7ed88bb
[36mDelete on Terminate[0m: true

[35m=====
[36mPublic DNS Name[0m: ec2-52-23-215-193.compute-1.amazonaws.com
[36mPublic IP Address[0m: 52.23.215.193
[36mPrivate DNS Name[0m: ip-172-31-31-133.ec2.internal
[36mPrivate IP Address[0m: 172.31.31.133
[36mEnvironment[0m: _default
[36mRun List[0m: role[v-tomcat]
Connection to 192.168.0.103 closed.
Finished: SUCCESS
```

23. Check the AWS management console for the successful status.
24. We have used a different agent node for some build jobs. To keep it ready, make it active:

```
C:\Users\Mitesh\Downloads>java -jar slave.jar -jnlpUrl  
http://192.168.1.35:8080/computer/TestServer/slave-agent.jnlp -secret  
65464e02c58c85b192883f7848ad2758408220bed2f3af715c01c9b01cb72f9b
```

```
C:\Users\Mitesh\Downloads>java -jar slave.jar -jnlpUrl http://192.168.1.35:8080/compu  
ter/TestServer/slave-agent.jnlp -secret 65464e02c58c85b192883f7848ad2758408220bed2f3a  
f715c01c9b01cb72f9b  
Jul 30, 2016 11:21:00 AM hudson.remoting.jnlp.Main createEngine  
INFO: Setting up slave: TestServer  
Jul 30, 2016 11:21:00 AM hudson.remoting.jnlp.Main$CuiListener <init>  
INFO: Jenkins agent is running in headless mode.  
Jul 30, 2016 11:21:00 AM hudson.remoting.jnlp.Main$CuiListener status  
INFO: Locating server among [http://192.168.1.34:8080/, http://192.168.1.35:8080/]  
Jul 30, 2016 11:21:01 AM hudson.remoting.jnlp.Main$CuiListener status  
INFO: Handshaking  
Jul 30, 2016 11:21:01 AM hudson.remoting.jnlp.Main$CuiListener status  
INFO: Connecting to 192.168.1.35:33337  
Jul 30, 2016 11:21:01 AM hudson.remoting.jnlp.Main$CuiListener status  
INFO: Trying protocol: JNLP3-connect  
Jul 30, 2016 11:21:02 AM hudson.remoting.jnlp.Main$CuiListener status  
INFO: Server didn't accept the handshake: Unknown protocol:Protocol:JNLP3-connect  
Jul 30, 2016 11:21:02 AM hudson.remoting.jnlp.Main$CuiListener status  
INFO: Connecting to 192.168.1.35:33337  
Jul 30, 2016 11:21:02 AM hudson.remoting.jnlp.Main$CuiListener status  
INFO: Trying protocol: JNLP2-connect  
Jul 30, 2016 11:21:02 AM hudson.remoting.jnlp.Main$CuiListener status  
INFO: Connected
```

25. Go to the Jenkins dashboard, and click on **Manage Jenkins**. Navigate to **Manage Nodes** and verify the status of both the master and agent:

S.	Name	Architecture	Clock Difference	Free Disk Space	Free Swap Space	Free Temp Space	Response Time	
	master	Linux (amd64)	In sync	7.88 GB	1.66 GB	7.88 GB	0ms	
	TestServer	Windows 8.1 (amd64)	In sync	31.89 GB	2.47 GB	120.38 GB	3087ms	
	Data obtained	58 sec	58 sec	1 min 0 sec	58 sec	58 sec	1 min 0 sec	
<a href="#">Refresh status</a>								

26. Verify hosted Chef for registered nodes:

The screenshot shows the Chef Management interface. On the left, there's a sidebar with links like 'Nodes', 'Manage Tags', 'Reset Key', 'Edit Run List', and 'Edit Attributes'. The main area has tabs for 'Nodes', 'Reports', 'Policy', and 'Administration'. Under 'Nodes', it says 'Showing All Nodes' and lists two nodes: 'DataOpMachine' and 'tomcatserver'. Each node row includes columns for 'Node Name', 'Pattern', 'FOONI', 'IP Address', 'Update', 'Last Checkin', 'Environment', and 'Actions'. Below this, there's a detailed view for the 'tomcatserver' node, showing 'Details', 'Attributes', and 'Permissions' tabs. The 'Details' tab displays 'Last Checkin: 3 Months Ago (2016-01-15 16:46)', 'Update: 6 Hours (2016-01-29 11:46:15)', 'Environment: \_default', and 'Attributes: FOONI IP Address: 192.168.1.37'.

Now, we have all the resources ready to configure the build pipeline.

# Configuring the build pipeline for build job orchestration

Now, it is time to integrate all the work in a way that continuous integration, cloud provisioning, configuration management, and continuous delivery is orchestrated in a sequence:

1. In the Jenkins dashboard, go to **PetClinic-Build-Pipeline-View**:

The screenshot shows the Jenkins configuration page for the 'PetClinic-Build-Pipeline-View'. The page has several input fields and options:

- Name:** PetClinic-Build-Pipeline-View
- Description:** (Empty)
- [Plain text] Preview:** (Empty)
- Filter build queue:** (checkbox)
- Filter build executors:** (checkbox)
- Build Pipeline View Title:** First Pet-Clinic Build Pipeline
- Layout:** Based on upstream/downstream relationship

At the bottom are two buttons: **OK** (dark blue) and **Apply** (light gray).

- Click on **Configure** to view settings or modify them:

**Build Pipeline View Title**: First Pet-Clinic Build Pipeline

**Layout**: Based on upstream/downstream relationship

The layout mode derives the pipeline structure based on the upstream/downstream trigger relationship between jobs.

**Select Initial Job**: PetClinic-Compile

**No Of Displayed Builds**: 2

**Restrict triggers to most recent successful builds**:  Yes  No

**Always allow manual trigger on pipeline steps**:  Yes  No

**Show pipeline project headers**:  Yes  No

**Show pipeline parameters in project headers**:  Yes  No

**Show pipeline parameters in revision box**:  Yes  No

**Refresh frequency (in seconds)**: 3

**URI for custom CSS file**:

**Console Output Link Style**: Lightbox

**OK** **Apply**

- Once we click on **OK**, changes are saved and we see the configuration, as shown in the following screenshot. It is the output of the upstream and downstream job configuration:



To create complete build pipeline mentioned in previous screenshot, we need to configure post build actions of each build job that we want to execute in specific sequence. We have different build job for continuous integration, configuration management and continuous delivery so we will configure them as upstream and downstream jobs to make pipeline. Let's look at each build job configuration step by step:

1. Click on **PetClinic-Compile | Configure**. Go to **Post-build Actions**. In the **Build other project** section, we have configured the build to be executed after **PetClinic-Compile** has been completed successfully:

The screenshot shows the Jenkins configuration interface for the 'PetClinic-Build-Pipeline-View' project, specifically the 'PetClinic-Compile' job. The 'Post-build Actions' tab is selected. Under the 'Build other projects' section, the 'Projects to build' field contains 'PetClinic-Test'. Three trigger options are listed: 'Trigger only if build is stable' (selected), 'Trigger even if the build is unstable', and 'Trigger even if the build fails'. At the bottom of the screen, there are 'Save' and 'Apply' buttons.

2. Save it and verify **Downstream Projects** on the Jenkins dashboard:

The screenshot shows the Jenkins dashboard for the project "PetClinic-Compile". On the left, there's a sidebar with links like Back to Dashboard, Status, Changes, Workspace, Build Now, Delete Project, Configure, Mine, and Create. Below that is a "Build History" section with two entries: #26 (Jul 30, 2018 12:42 PM) and #25 (Jul 30, 2018 12:05 PM). The main area is titled "Project PetClinic-Compile" and contains sections for "Analysis results" (with links to "BuildLog" and "Review Changes") and "Downstream Projects" (listing "PetClinic-Test"). A "Static Analysis Trend" chart is also present.

3. Click on **PetClinic-Test | Configure**. Go to **Post-build Actions**. In **Build other projects**, we have configured the build to be executed after PetClinic-Test has been completed successfully.
4. Configure the archive artifacts so we can copy it for deployment:

The screenshot shows the "Configure" screen for the "PetClinic-Test" job in Jenkins. The "Post-build Actions" tab is selected. It displays two sections: "Archive the artifacts" (with a "target" dropdown set to "war") and "Build other projects" (with "PetClinic-CloudProvisioning" selected as the target). Under "Build other projects", three options are available: "Trigger only if build is stable", "Trigger even if the build is unstable", and "Trigger even if the build fails". At the bottom, there are "Save" and "Apply" buttons.

5. Verify **Upstream** and **Downstream** projects for the **PetClinic-Test** build job in the Jenkins dashboard:

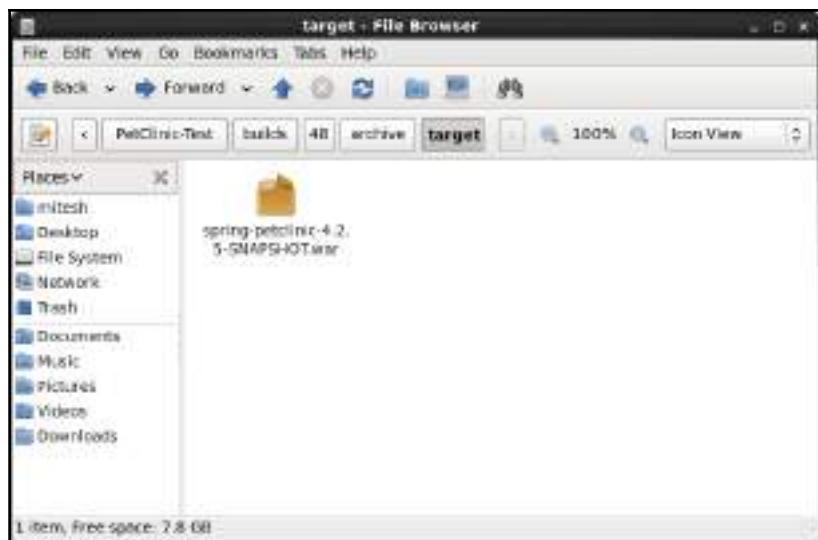
The screenshot shows the Jenkins interface for the 'PetClinic-Test' project. On the left, there's a sidebar with links like 'Back to Dashboard', 'Status', 'Changes', 'Workspace', 'Build Now', 'Delete Project', 'Configure', 'GitHub Hook Log', 'Move', and 'GitHub'. Below this is a 'Build History' section with a dropdown menu set to 'last 5' builds. The main area is titled 'Project PetClinic-Test'. It contains sections for 'Workspace' (with a link to 'Last Successful Action'), 'Recent Changes' (with a link to 'Recent Changes'), 'Upstream Projects' (listing 'PetClinic-Compile'), and 'Downstream Projects' (listing 'PetClinic-CloudProvisioning'). At the bottom right, there are 'Edit' and 'Delete Project' buttons.

6. Execute the build independently to check whether the artifact or WAR file is archived:

```
[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-4.2.0-SNAPSHOT]
[INFO] Processing war project
[INFO] Copying webapp resources [d:\jenkins\workspace\PetClinic-Test\src\main\webapp]
[INFO] Webapp assembled in [3129 ms]
[INFO] Building war: d:\jenkins\workspace\PetClinic-Test\target\spring-petclinic-4.2.0-SNAPSHOT.war
[INFO] 
[INFO] BUILD SUCCESS
[INFO] 
[INFO] -----
[INFO] Total time: 41.817 s
[INFO] Finished at: 2016-07-30T12:00:44+05:30
[INFO] Final Memory: 139/263M
[INFO] 
[INFO] Archiving artifacts
[INFO] Warning: you have no plugins providing access control for builds, so falling back to legacy behavior of permitting any downstream builds to be triggered
[INFO] Triggering a new build of PetClinic-CloudProvisioning
[INFO] Finished: SUCCESS
```

The screenshot shows the Jenkins build log for the 'PetClinic-Test' job. The log output details the execution of the Maven build, specifically the 'war' phase of the 'maven-war-plugin'. It shows the plugin assembling the webapp from the 'src/main/webapp' directory into a war file named 'spring-petclinic-4.2.0-SNAPSHOT.war'. The build is successful, taking approximately 41.817 seconds. The log also includes a warning about access control and triggers a new build for the 'PetClinic-CloudProvisioning' downstream project. At the bottom, it says 'Finished: SUCCESS'.

7. Verify the archived artifact in the Jenkins home directory:



8. At this stage, the build pipeline will look like this:



9. Click on **PetClinic-CloudProvisioning | Configure**. Go to **Post-build Actions**. In the **Build other projects** section, we have the configured the build to be executed after **PetClinic-CloudProvisioning** has been completed successfully:

The screenshot shows the Jenkins configuration interface for the 'PetClinic-CloudProvisioning' project. The top navigation bar includes 'General', 'Source Code Management', 'Build Triggers', 'Build Environment', 'Build', and 'Post-build Actions'. The 'Post-build Actions' tab is selected. Below it, the 'Post-build Actions' section is displayed. Under the heading 'Build other projects (manual step)', there is a dropdown menu labeled 'Downstream Project Names' containing 'PetClinic-Deploy'. A 'Save' button is visible at the bottom left, and an 'Apply' button is visible at the bottom right. The footer of the page indicates 'Page generated: Aug 4, 2016 10:10:21 PM PDT'.

- Verify **Upstream** and **Downstream** projects for the **PetClinic-CloudProvisioning** build job in the Jenkins dashboard:

The screenshot shows the Jenkins dashboard for the 'PetClinic-CloudProvisioning' project. On the left, there's a sidebar with links like 'Back to Dashboard', 'States', 'Changes', 'Workspace', 'Build Now', 'Delete Project', 'Configure', and 'New'. The main area is titled 'Project PetClinic-CloudProvisioning'. It shows 'Upstream Projects' with a link to '@PetClinic-Test'. Below that is 'Downstream Projects' with a link to '@PetClinic-Deploy'. Under 'Permalinks', there are two entries: 'B12' (Build #12, Jul 31, 2018 12:07 PM) and 'B13' (Build #13, Jul 31, 2018 12:07 PM).

- Click on **PetClinic-Deploy | Configure**. Our WAR file was ready in the **PetClinic-Test** build job, so let's copy it to a common location and configure it as a build step:

The screenshot shows the 'Build' configuration screen for the 'PetClinic-Deploy' job. The top navigation bar includes 'Overview', 'Internal Code Management', 'Edit Triggers', 'Edit Environment', 'Build' (which is selected), and 'Post-build Actions'. The 'Build' section has a heading 'Copy artifacts from another project'. It shows 'Project name: PetClinic-Test' and 'Which build: Latest successful build'. There's a checkbox for 'Stable build only'. Under 'Artifacts to copy', the value is '/target/spring-petclinic-4.2.5-SNAPSHOT.war'. The 'Target directory' is set to '/home/nitish/'. Below that, 'Target directories' is set to '/home/nitish/' and 'Optional' is checked. At the bottom are 'Save' and 'Apply' buttons.

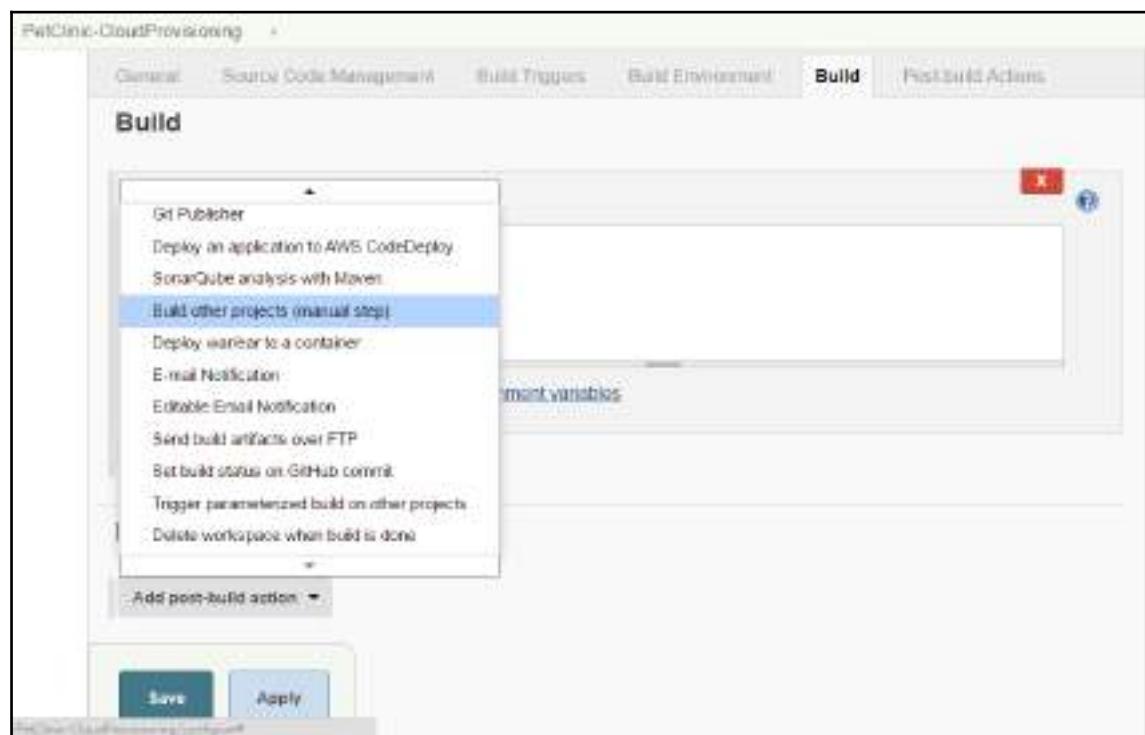
12. Configure the build step to copy the artifact and verify it by executing the build independently:

The screenshot shows the Jenkins interface for a job named "FatClinic-Deploy". On the left, there's a sidebar with options like "Back to Project", "Status", "Changes", "Console Output" (which is selected), "View as plain text", "Edit Build Information", "Delete Build", "See Fingerprints", and "Previous Build". The main area is titled "Console Output" and displays the following log entries:  
Started by user DiscoverTechno  
Building on master in workspace /home/jenkins/.jenkins/workspace/FatClinic-Deploy  
Copied 1 artifact from "FatClinic-Test" build number 48  
Finished: SUCCESS

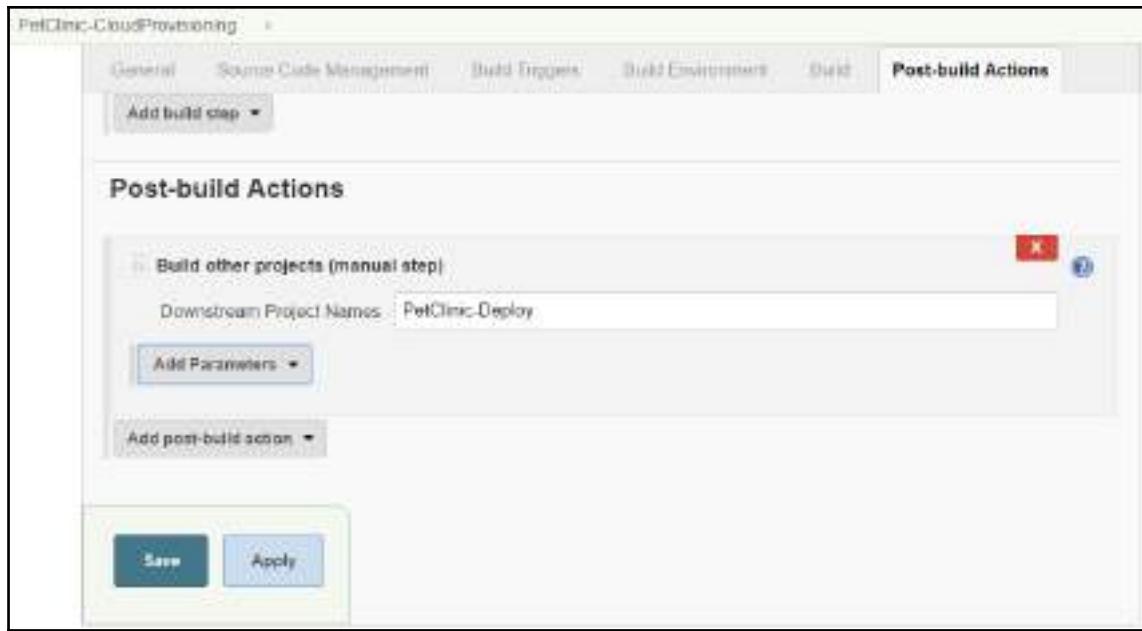
13. Once the artifact copy operation is verified, configure the build job so we can deploy it as a manual operation. We will create a job with the **String Parameter** of a newly created instance's domain name or IP address:

The screenshot shows the "General" configuration page for a Jenkins job named "FatClinic-Deploy". Under the "Parameters" section, there is a "String Parameter" entry with the following details:  
Name: AWSDNS  
Default Value: ec2-54-88-73-50.compute-1.amazonaws.com  
Description: Plain text|Percent

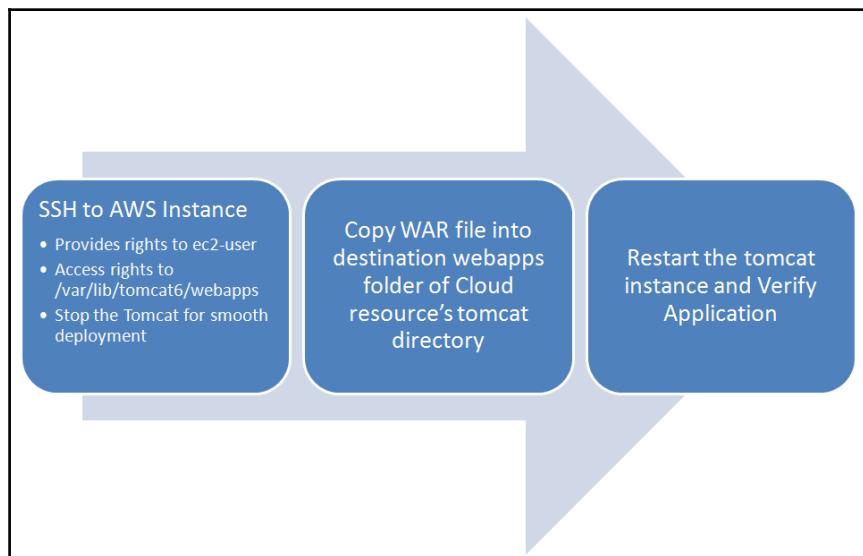
14. Go to the **PetClinic-CloudProvisioning** build job and check whether we have added **Build other projects (manual step)** for the **PetClinic-Deploy** build job:



15. Once you've verified this, let's move to executing the build pipeline:

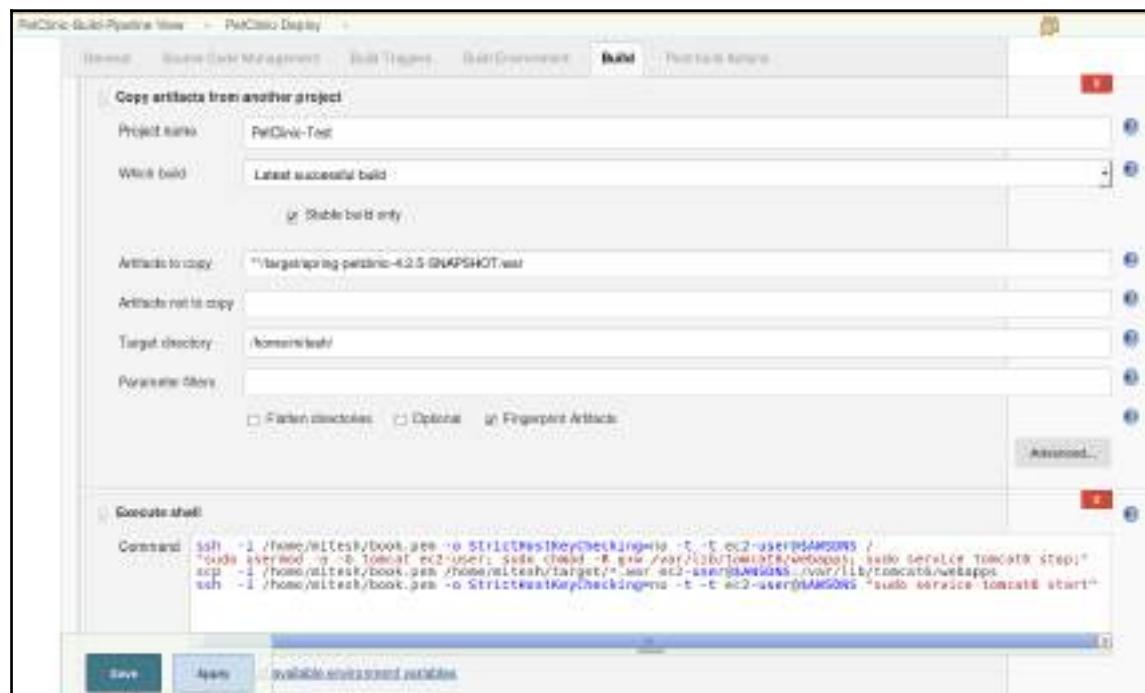


16. Once our artifact is ready to deploy, we need to perform following steps:



Let's configure the build job to execute deployment of WAR file in AWS instance by executing following command:

```
ssh -i /home/mitesh/book.pem -o StrictHostKeyChecking=no -t -t ec2-user@ec2-52-90-116-36.compute-1.amazonaws.com "sudo usermod -a -G tomcat ec2-user; sudo chmod -R g+w /var/lib/tomcat6/webapps; sudo service tomcat6 stop;"  
scp -i /home/mitesh/book.pem /home/mitesh/target/*.war ec2-user@ec2-52-90-116-36.compute-1.amazonaws.com:/var/lib/tomcat6/webapps  
ssh -i /home/mitesh/book.pem -o StrictHostKeyChecking=no -t -t ec2-user@ec2-52-90-116-36.compute-1.amazonaws.com "sudo service tomcat6 start"
```



1. Save the build job configuration. Verify the **Upstream Projects**:

The screenshot shows the Jenkins interface for the 'Project PetClinic-Deploy'. On the left, there's a sidebar with links like 'Back to Dashboard', 'Status', 'Changes', 'Workspaces', 'Build with Parameters', 'Delete Project', 'Configure', 'Move', and 'Orchestrator'. The main area is titled 'Project PetClinic-Deploy'. Below the title, there are two sections: 'Workspace' (with a link to 'Recent changes') and 'Upstream Projects' (with a link to 'PetClinic-CloudProvisioning'). Under 'Upstream Projects', it says 'Last build (M7E), 3 days 8 hr ago'. At the bottom, there's a 'Build History' table with one entry: 'Last build (M7E), 3 days 8 hr ago'. On the right side, there are buttons for 'Edit description' and 'Delete Project'.

2. It says **PetClinic-CloudProvisioning**, so once instance provisioning in the cloud is completed, the deployment process will start:

The screenshot shows the 'Post-build Actions' configuration screen for the 'PetClinic-CloudProvisioning' build job. The top navigation bar includes 'General', 'Source Code Management', 'Build Triggers', 'Build Environment', 'Build', and 'Post-build Actions'. The 'Post-build Actions' tab is selected. A single action is listed: 'Build other projects (manual-step)'. The 'Downstream Project Names' field is set to 'PetClinic-Deploy'. There are buttons for 'Add Parameters' and 'Add post-build action'. At the bottom, there are 'Save' and 'Apply' buttons. The footer of the page indicates it was generated on Aug 4, 2016 at 10:10:21 PM PDT.

3. Make sure to configure **Downstream Projects** in the **PetClinic-CloudProvisioning** build job.

- The key downloaded from AWS must have proper permissions. If it doesn't, the shell command gives an error saying the permissions for the key are too open:

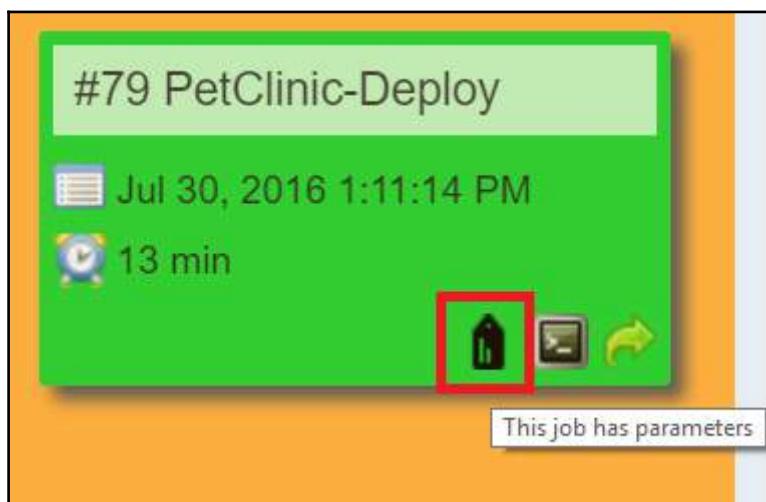
**Console Output**

```
Started by user DiscoverTechno
Building on master in workspace /home/mitsash/.jenkins/workspace/PetClinic-Deploy
Copied 1 artifact from "PetClinic-Test" build number 48
[PetClinic-Deploy] $ /bin/sh -e /tmp/hudson573163256020132324.sh
+ ssh -o StrictHostKeyChecking=no -t -t ec2-user@52.90.116.36 -i /home/mitsash/book.pem & cat config
Warning: Permanently added '52.90.116.36' (RSA) to the list of known hosts.
=====
#      WARNING: UNPROTECTED PRIVATE KEY FILE!
=====
Permissions 0644 for '/home/mitsash/book.pem' are too open.
It is required that your private key files are NOT accessible by others.
This private key will be ignored.
bad permissions: ignore key: /home/mitsash/book.pem
Permission denied (publickey).
Build step 'Execute shell' marked build as failure
Finished: FAILURE
```

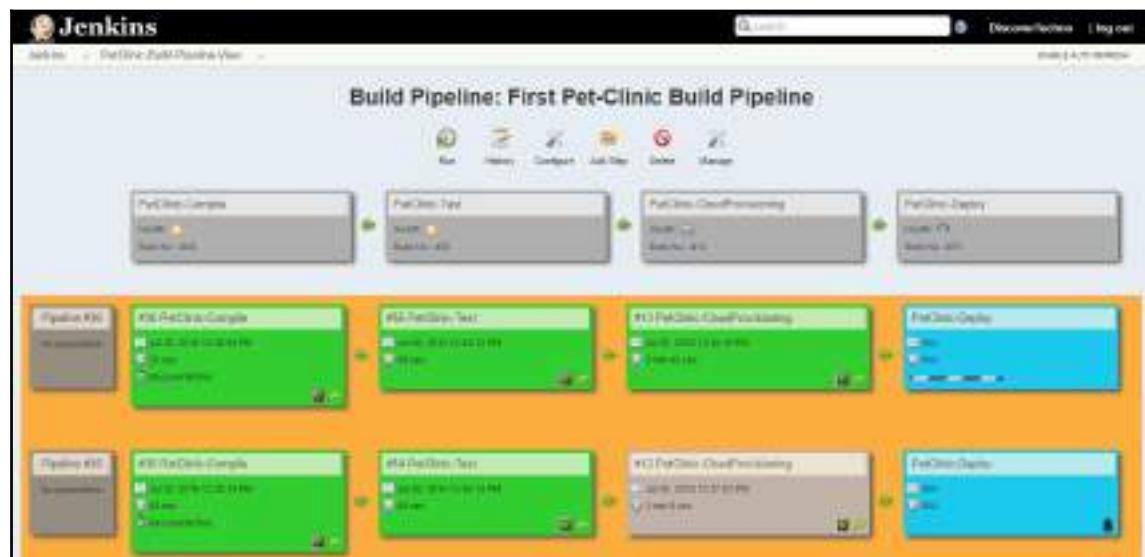
- To fix it, use `chmod 600` to change the permission for the given file, and execute the command.
- Once all build jobs have been verified as running individually, run the build pipeline:



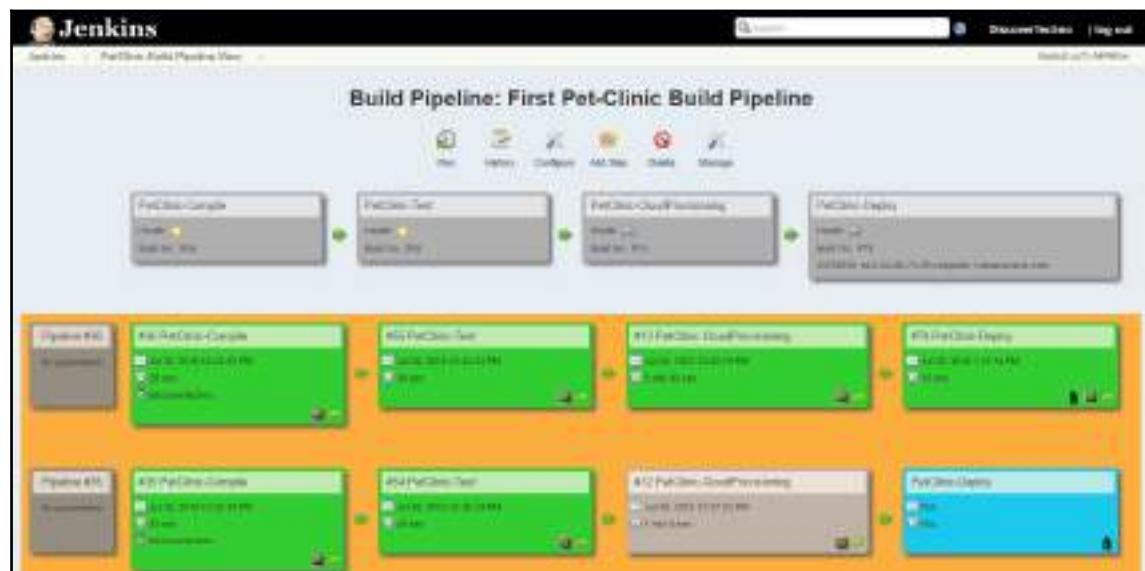
- Once all three build jobs have been executed successfully, we need to manually execute the last build job for deployment:



- Wait for the build job's result:

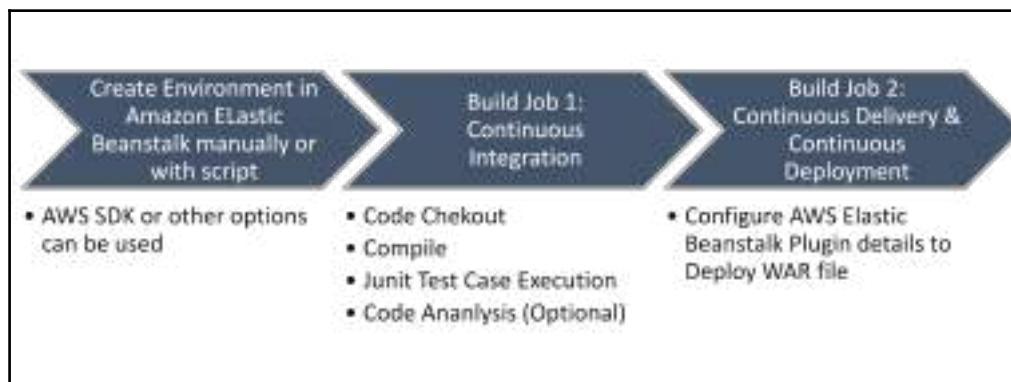


- Once the application deployment is successful, we have all the successful build jobs in the build pipeline:



- Check whether the application is running properly and whether it is configured in hosted Chef.

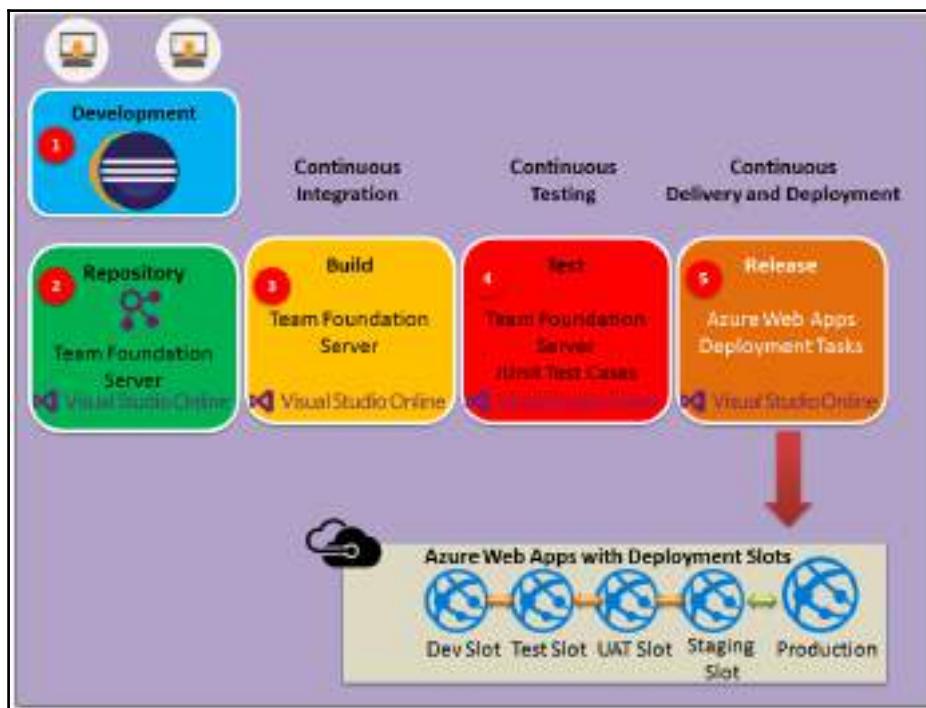
To deploy the PetClinic Spring application in Amazon Elastic Beanstalk (PaaS), we need the following flow:



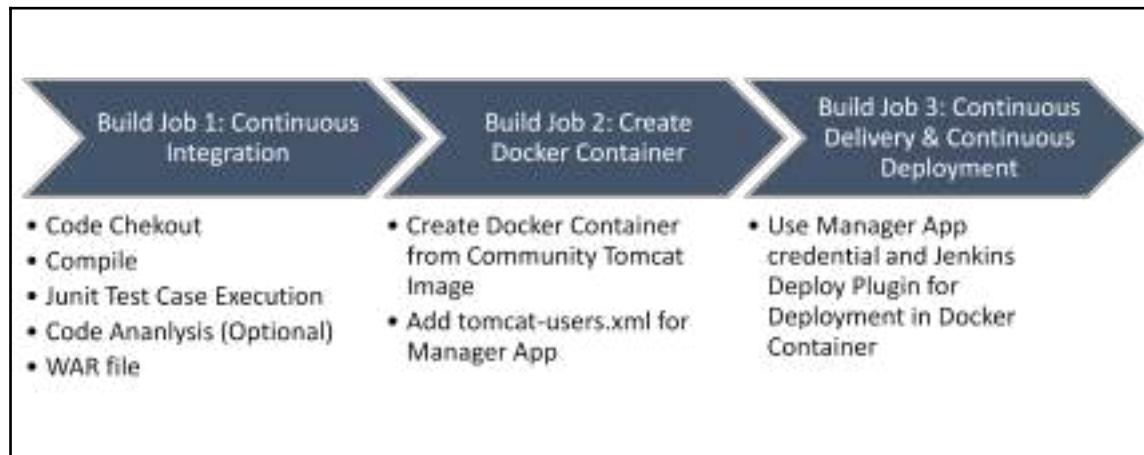
To deploy the PetClinic Spring application in Microsoft Azure web apps (PaaS), we need the following flow:



In Microsoft Azure's case, there is an alternative as well. We can use Visual Studio Team Server and TFS online for continuous integration, continuous delivery, and continuous deployment:



To deploy the PetClinic Spring application in a Docker container, we need the following flow:



In the next section, we will see how to use the pipeline feature of Jenkins 2.0 in brief.

## Executing the pipeline for application deployment automation

The pipeline feature in Jenkins 2.0 also provides features to orchestrate end-to-end automation for application deployment.

To give you an overview, here's a script that achieves checkout, continuous integration, cloud provisioning, and configuration management:

```
node('Master') {
    // Mark the code checkout 'stage'
    stage 'Checkout'

    // Get code for PetClinic Application from a GitHub repository
    git url: 'https://github.com/mitesh51/spring-petclinic.git'

    // Get the maven tool.
    // This ' Maven3.3.1' maven tool must be configuredin the global
    configuration.
    def mvnHome = tool 'Maven3.3.1'

    // Mark the code Compile'stage'....
```

```
stage 'Compile'
// Run the maven build
sh "${mvnHome}/bin/mvn clean compile"

// Mark the code for Unit test execution and package 'stage'.....
stage 'Test&Package'
sh "${mvnHome}/bin/mvn clean package"

// Mark the code Cloud provisioning 'stage' where instance is allocated
in Amazon EC2
// Once Instance is available, Chef will be used for Configuration
Management
// knife ec2 plugin will be used for instance provisioning in the AWS cloud
stage 'Cloud Provisioning'
sh "ssh -t -t root@192.168.1.39 'ifconfig; rvm use 2.1.0; knife ec2
server create -I ami-1ecae776 -f t2.micro -N DevOpsVMonAWS9 --aws-access-
key-id XXXXXXXXXXXXXXXXXXXXXXXXX --aws-secret-access-key
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXX -S book --identity-file book.pem --ssh-
user ec2-user -r role[v-tomcat]'"
}
```

1. Create a new item in the Jenkins dashboard, and select **Pipeline**:



2. In the **Pipeline** section, write the previous script, and make necessary changes:

```

1+ node('Master') {
2   // Have the code checkout "stage".....
3   stage("Checkout")
4
5   // Get some code from a GitHub repository
6   git url: "https://github.com/mashkil/spring-petclinic"
7
8   // Fetch the maven tool.
9   // *** NOTE: This 'maven' tool must be configured
10  // *** in the global configuration.
11  def mavenHome = tool 'Maven_3_1'
12
13  // Now the code build "stage"....
14  stage("Compile")
15  // Run the Maven build
16  sh "${mavenHome}/bin/mvn clean compile"
17

```

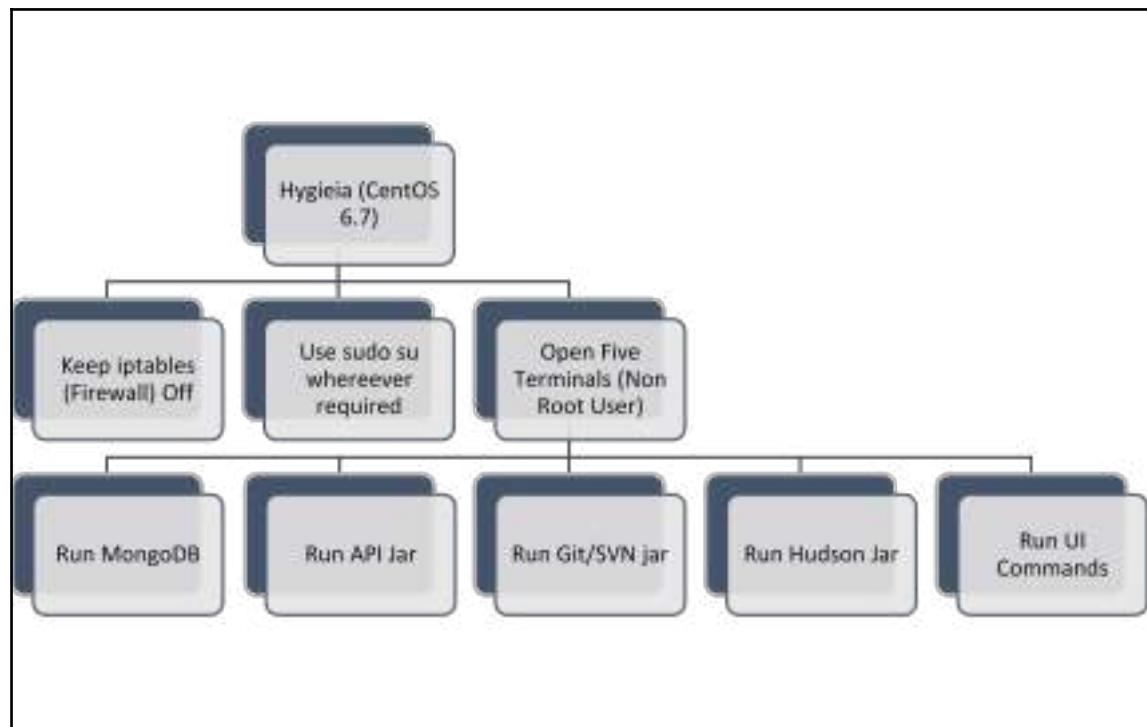
3. Once the script is modified and saved in the build job, click on the **Build Now** link and verify the execution:

Stage	Duration
Checkout	20
Compile	6
Test&Package	30
Clean Provisioning	3min 38s

Check the DSL reference and use the script to perform all the operations. In the next section, we will get an overview of DevOps dashboards.

## Hygieia – a DevOps dashboard

DevOps dashboards are also an emerging need. The point of having them is to have a view of all the tools in a single dashboard. **Hygieia** is an open source initiative to provide a unified, configurable, and easy-to-use DevOps dashboard for an end-to-end application delivery pipeline:



For the installation, visit <https://github.com/capitalone/Hygieia/blob/master/Setup.md>.

Once installation and configuration is successfully completed, we can create a DevOps dashboard that might look something like this:



<https://github.com/capitalone/Hygieia>

We can configure SVN, Git, Sonar, Jenkins, and IBM UrbanCode Deploy in a Hygieia dashboard.

## Self-test questions

State whether the following statements are true or false:

- AWS security groups need to have port 22 as an inbound rule to go further for configuration management
- A deployment build job is configured for manual execution with a parameterized job

# Summary

In this chapter, we covered how to use a Jenkins build job to execute SSH commands to be executed on a Chef workstation for configuration management on an AWS instance, and we also covered setting up permissions for application deployment.

We used the build pipeline plugin as well as the pipeline feature of Jenkins 2.0 for end-to-end automation. This chapter provided a brief overview of all deployment methods covered in this book and how they are different from each other. It also provides a brief overview of deployment methods with respect to the build pipeline.

Finally, we covered the Hygieia DevOps dashboard as it provides an end-to-end unified view of tools used in the automation process.

*“Every new beginning comes from some other beginning’s end.”*

– Seneca

# Index

## A

agent node 235  
agile model 13, 14, 15  
Amazon EC2  
    virtual machine, configuring in 210, 212, 213, 214, 215, 216, 218  
    virtual machine, creating in 210, 212, 213, 214, 215, 216, 217  
Amazon Web Services  
    PetClinic application, deploying in 254, 255, 256, 257, 258, 259, 260, 261, 262, 263, 265, 266, 267, 268, 269  
    used, for installing knife plugins 208, 209  
Apache Maven build tool  
    about 38, 39  
    features 39  
    pom.xml file 39  
Apache Maven  
    configuring 70  
    configuring, in Jenkins 69  
    URL, for downloading 70  
    used, for configuring Jenkins builds in Java application 70, 71  
    used, for creating Jenkins builds in Java application 70, 71  
AWS Elastic Beanstalk Publisher plugin  
    about 258  
    URL 258  
AWS Elastic Beanstalk, monitoring  
    reference link 305  
AWS Elastic Beanstalk  
    monitoring 303, 304, 305

**B**

BACPAC files 29  
build automation tool 21, 22

## build jobs

    configuring, master node used 92, 93  
    configuring, slave node used 92, 93  
    creating, for end-to-end automation 344, 345, 346, 347

## build pipeline plugin

    using 127, 128, 129, 130, 132, 133, 134, 135, 136, 137, 138

## build pipeline view

    configuring, build job orchestration used 366, 367, 368, 369, 370, 371, 372, 373, 374, 375, 376, 377, 378, 379, 380, 381, 382, 383  
    properties 130

## built-in delivery pipelines

    creating 110, 111, 112  
    script, creating 112

## C

### Chef

    169  
    Chef client  
        reference link 154  
    Chef configuration management tool  
        about 45  
        features 46  
    Chef Development Kit (ChefDK) 208  
    Chef development kit  
        URL, for downloading 155  
    chef provisioning 206, 207, 208  
    Chef  
        about 148, 149  
        Chef workstations 148  
        hosted Chef 148  
        node 148  
        open source Chef server 148  
        reference link 46  
        URL 149  
        workstation, configuring 156, 157, 158

workstation, installing 156, 157, 158  
workstation, used for converging Chef node 159, 160, 161, 165, 166

**ChefDK**  
URL, for downloading 208

classic Azure portal 223

cloud computing 25, 26, 27

cloud deployment models  
  community cloud 26  
  hybrid cloud 26  
  private cloud 26  
  public cloud 26

cloud provisioning 206, 207, 208

cloud service providers 46, 47

collaboration 15, 16

configuration management (CM) 27

container technology 47  
  Docker containers 48

continuous delivery (CD)  
  about 19, 28, 29  
  best practices 30  
  reference link 30

continuous deployment  
  about 28, 29

continuous feedback 32, 33

continuous integration (CI) 19  
  about 22, 23, 60  
  advantages 23  
  best practices 24  
  pull mechanism 22, 60  
  push mechanism 22, 60  
  reference link 23

continuous monitoring 30, 32  
  reference link 32

continuous testing (CT) 19

cookbooks  
  references 167  
  used, for installing software packages 167, 168, 169

**D**

Dashboard View plugin  
  overview 83, 84, 85, 86  
  usage 83, 84, 85, 86

Deploy plugin 236

deployment operation  
  integrating 138, 139, 140, 141, 142, 143, 144, 145

DevOps lifecycle  
  about 19, 20  
  build automation tool 21, 22  
  cloud computing 25, 27  
  configuration management (CM) 27  
  continuous delivery (CD) 28, 29  
  continuous deployment 28, 29  
  continuous feedback 32, 33  
  continuous integration (CI) 22, 23  
  continuous monitoring 30, 31, 32

DevOps movement  
  about 9, 10  
  advantages 18  
  agile model 13, 14, 15  
  cloud computing 16  
  collaboration 15, 16  
  determining 16, 17  
  time, changing 11  
  waterfall model 12, 13

DevOps  
  Apache Maven 38  
  Apache Maven build tool 39  
  Chef configuration management tool 44, 45  
  cloud service providers 46, 47  
  container technology 47  
  dashboard 52  
  Git repositories 33  
  Jenkins 41  
  Jenkins continuous integration tool 40  
  Jenkins plugins 50  
  monitoring tools 49  
  tools and technologies 33

distributed builds  
  about 86  
  features 86

Docker container  
  about 48, 178, 179, 180, 181, 226, 227, 228, 229, 230  
  client-server architecture 187, 188, 189, 191, 192  
  configuring, on CentOS 183, 185  
  creating 185, 186, 187

Docker host 178  
Docker Hub 179  
installing, on CentOS 183, 185  
managing 192, 193, 195, 197  
PetClinic application, deploying in 250, 253  
versus, virtual machines 182  
Docker engine 178  
Docker Hub  
  URL 179  
Docker image  
  creating, from Dockerfile 198, 201  
domain-specific language (DSL) 110

## E

e-mail notifications  
  sending, on build status 94, 95, 96, 97

## G

General Public License (GPL) 49  
Git repositories  
  about 33  
  advantages 34  
  characteristics 34  
  versus Subversion 35  
  versus, Subversion 36

## H

hosted Chef  
  overview 149, 150, 151, 152, 153, 154, 155  
Hygieia 386  
Hygieia-DevOps dashboard 386, 387  
  reference link 387

## I

Identity and Access Management (IAM) 214

## J

Java Development Kit (JDK) 163  
Java EE application  
  overview 53, 54  
  tasks 55  
Java Enterprise Edition (Java EE) application 109  
Java  
  configuring 69

URL, for downloading 199  
Jenkins builds  
  configuring, for Java application Apache Maven  
    used 70, 71  
  creating, for Java application Apache Maven  
    used 70, 71  
Jenkins continuous integration tool  
  about 40, 41  
  features 41, 42, 43  
  reference link 41  
Jenkins plugins  
  about 50  
  end-to-end orchestration 51  
  reference link 66  
jenkins.war file  
  URL, for downloading 62  
Jenkins  
  Apache Maven, configuring in 69  
  dashboard 67, 68, 69  
  installing 61, 62, 63  
  integrating 97  
  Java, configuring in 69  
  setting up 64, 65, 66  
JUnit  
  configuring 81, 82, 83

## K

knife Azure plugin 205  
knife EC2 plugin 205  
knife plugin 148

## M

Microsoft Azure Web App Service  
  monitoring 306, 307, 308, 309, 310, 311, 312, 313, 314, 315, 316, 317, 318, 319, 320, 321, 322, 323, 324, 325, 326, 327, 328, 329, 330, 331, 332  
Microsoft Azure  
  PetClinic application, deploying in 269, 270, 271, 272, 273, 274, 275, 276, 277, 278, 279  
  URL 270  
  used, for installing knife plugin 208, 209  
  virtual machine, configuring in 218, 219, 222, 223, 224, 225  
  virtual machine, creating in 218, 219, 222, 223,

224, 225  
monitoring 282  
monitoring techniques  
  overview 282  
monitoring tool  
  Nagios 50  
monitoring tools  
  about 49  
  Nagios 283  
  overview 282  
  Zenoss 49

## N

Nagios 50  
  about 283, 284, 285, 286, 287, 288, 289, 290,  
    291, 292, 293, 294, 295, 296, 297, 298, 299,  
    300, 301, 302, 303  
  features 283  
New Relic  
  used, for monitoring Tomcat server 332, 333,  
    334, 335, 336, 337, 338, 339, 340  
  used, for monitoring web application 332, 333,  
    334, 335, 336, 337, 338, 339, 340

## O

operations team, challenges  
  diagnosing 11  
  rectifying 11  
  redesigning 11  
  resource contention 11  
  tweaking 11

## P

PetClinic application  
  deploying, in Amazon Web Services 254, 255,  
    256, 257, 258, 259, 260, 261, 262, 263, 265,  
    266, 267, 268, 269  
  deploying, in Docker container 250, 253  
  deploying, in Microsoft Azure 269, 270, 271,  
    272, 273, 274, 275, 276, 277, 278, 279  
  deploying, on remote server 234, 235  
  Tomcat server, setting up 236, 237, 238, 239,  
    240, 241, 242, 243, 244, 245, 246, 247, 248,  
    249  
PetClinic build job

configuring 77, 78, 79, 80, 81  
Pipeline DSL reference  
  URL 112  
pipeline  
  creating, for compiling unit test 118, 119, 120,  
    121, 122, 124, 125, 126  
  creating, for executing unit test 118, 119, 120,  
    121, 122, 124, 125, 126  
  executing, application deployment automation  
    used 383, 385, 386  
Platform as a Service (PaaS) 254

pom.xml file

  about 39  
  example 40

Project Object Model (POM) XML file 39

## R

role  
  creating 170, 172, 174

## S

script  
  build job artifacts, archiving 115  
  build job, definite steps marking 117  
  build step, creating to publish test reports 114  
  build step, executing on node 116  
  creating 112  
  Groovy script, creating to build job 113  
security token 99  
slave node  
  build jobs, configuring master node used 92, 93  
  configuring, in Jenkins 2 87, 88, 89, 90, 91, 92  
  creating, in Jenkins 2 87, 88, 89, 90, 91, 92  
  managing 86, 87  
Snippet Generator  
  URL 112  
Software as a Service (SaaS) 179  
Software Configuration Management (SCM) 44  
software packages  
  installing, cookbooks used 167, 168  
SonarQube  
  integrating 97, 98, 99, 100, 101, 102, 105, 106,  
    107  
  plugin, installing 98, 99  
  URL, for downloading 98

**source code repository**  
authenticating, on GitHub 72, 73, 74, 75, 76, 77  
configuring, on GitHub 72, 73, 74, 75, 76, 77  
**SSH authentication**  
configuring, key used 348, 349, 350, 351, 352, 353, 354, 355, 356, 357, 358, 359, 360, 361, 362, 363, 364, 365  
**Subversion**  
versus, Git repositories 35, 36

## T

**Tomcat 7**  
URL, for downloading 236  
**Tomcat server**  
monitoring, New Relic used 332, 333, 334, 335, 336, 337, 338, 339, 340  
setting up 236, 237, 238, 239, 240, 241, 242, 243, 244, 245, 246, 247, 248, 249  
**Tomcat**  
URL, for downloading 200

## V

**Version Label Format** 265  
virtual machine

configuring, in Amazon EC2 210, 211, 213, 214, 216, 217  
configuring, in Microsoft Azure 218, 220, 222, 223, 224, 225  
creating, in Amazon EC2 210, 211, 213, 214, 215, 216, 217  
creating, in Microsoft Azure 218, 220, 222, 223, 224, 225  
versus, Docker containers 182  
Virtual Private Cloud (VPC) 255  
VMware 64 bit  
URL, for downloading 284

## W

**waterfall model**  
about 12, 13  
advantages 13  
**Web application ARchive (WAR) file** 59  
**web application**  
monitoring, New Relic used 332, 333, 334, 335, 336, 337, 338, 339, 340

## Z

**Zenoss** 49