4

Installing and Configuring Chef

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

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

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

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

In this chapter, we will cover the following topics:

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

Getting started with Chef

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

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

There are three major components in Chef:

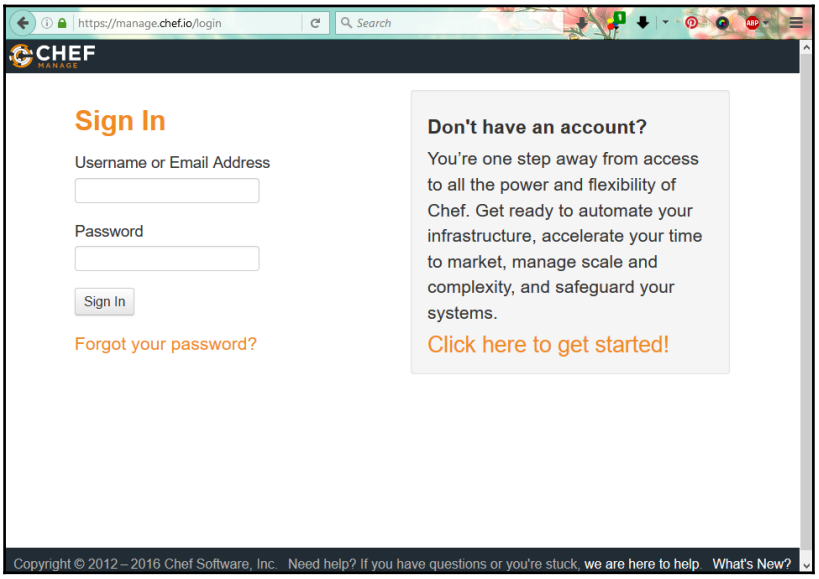
- **Open Source Chef Server or Hosted Chef**: Chef Server or Hosted Chef is the pivotal component that stores cookbooks and other important details of registered nodes. It is used to configure and manage Nodes with the use of Chef workstation.
- **Chef Workstation**: Chef Workstation works as local repository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

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

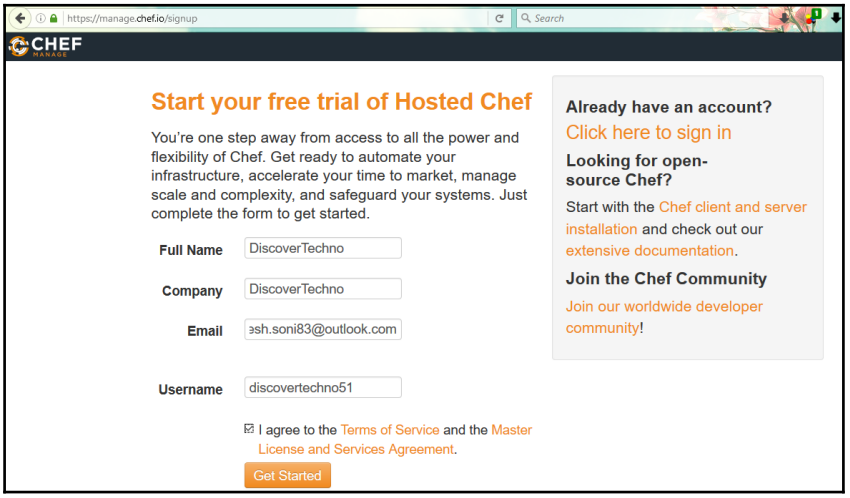
There are lot of details available of Chef and Cloud related integration and knife plugins too. We will create a Hosted Chef account in the section and configure it with local workstation. To go ahead, click on the **MANAGEMENT CONSOLE** link available in the right top corner of the Chef website.

Overview of Hosted Chef

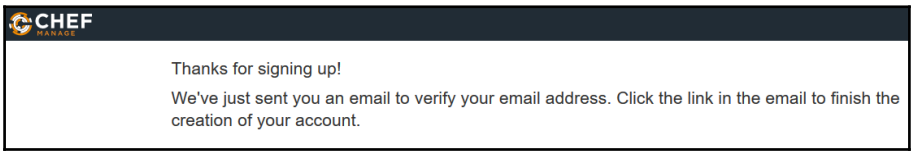
1. Click on **MANAGEMENT CONSOLE** or visit the URL https://manage.chef.io/login. We are going to start from scratch so click on **Click here to get started!**



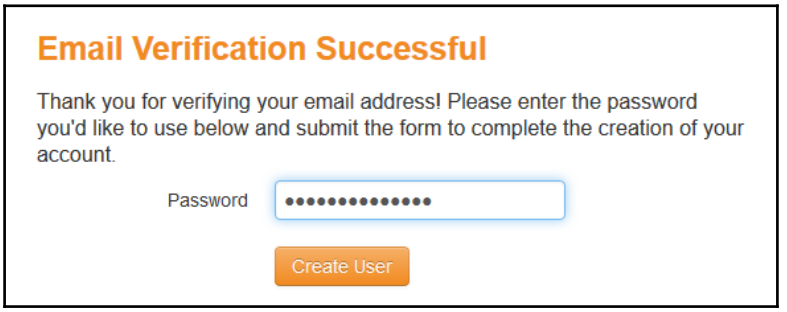
2. Enter **Full Name**, **Email**, and **Username** in the text boxes; check mark on **I agree to the Terms of Service and the Master License and Services Agreement**. Click on **Get Started** button.



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

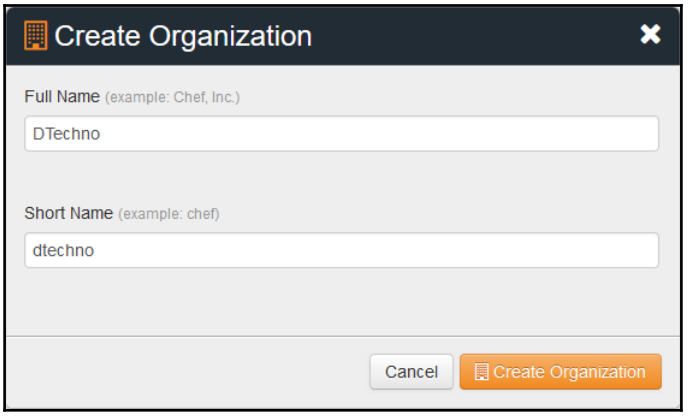


4. Open you Mail inbox and click on the verification link to complete the creation of Hosted Chef account. We will get **Email Verification Successful** message. Click on **Create User** button.

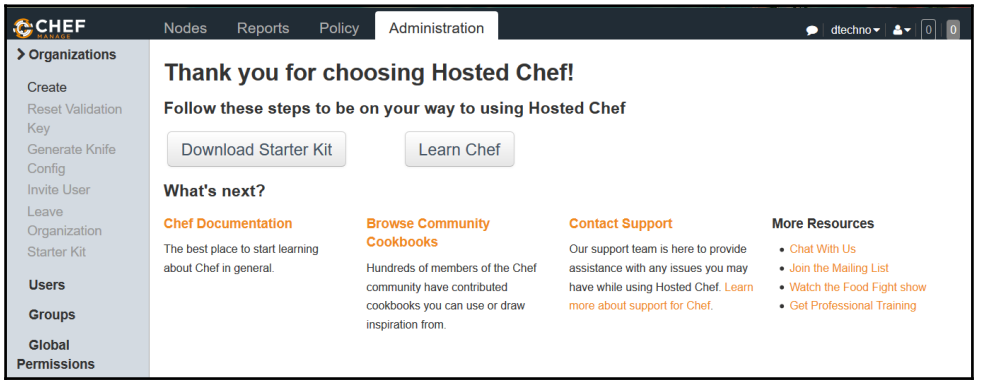


5. Next task is to create an organization. Click on **Create New Organization**.

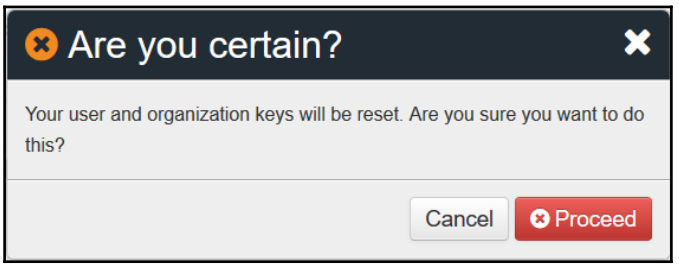
6. Provide **Full Name** and **Short Name** for the organization and click on the **Create Organization** button.



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

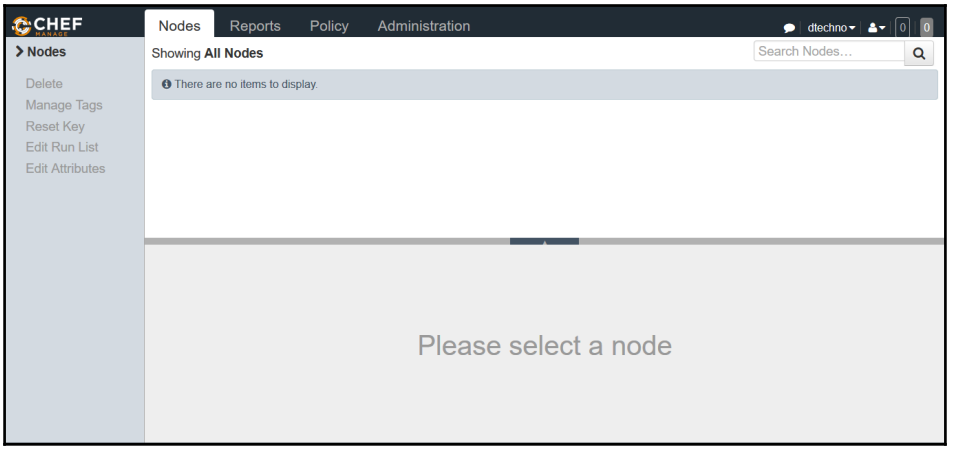


8. When we **Download Starter Kit**, User and Organization Keys will be reset. Make sure to keep it at safe place. Click on **Proceed**.

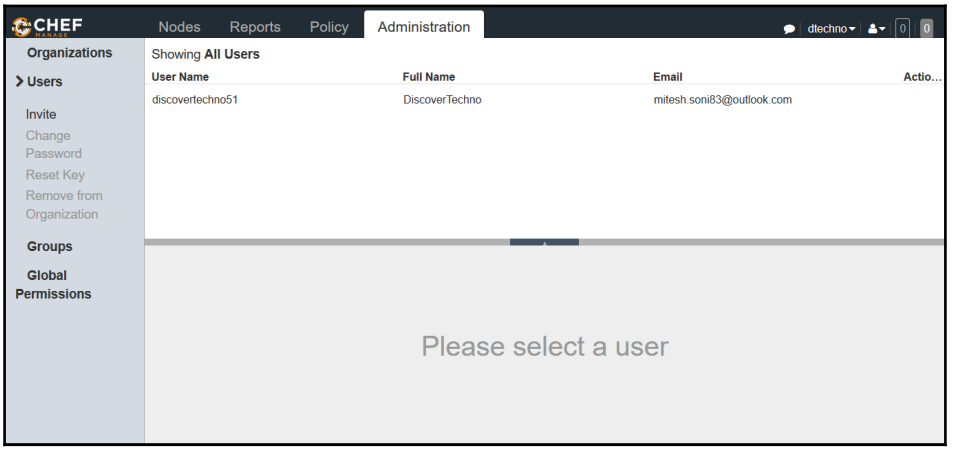


Let's have a quick walk through of the Hosted Chef portal or dashboard:

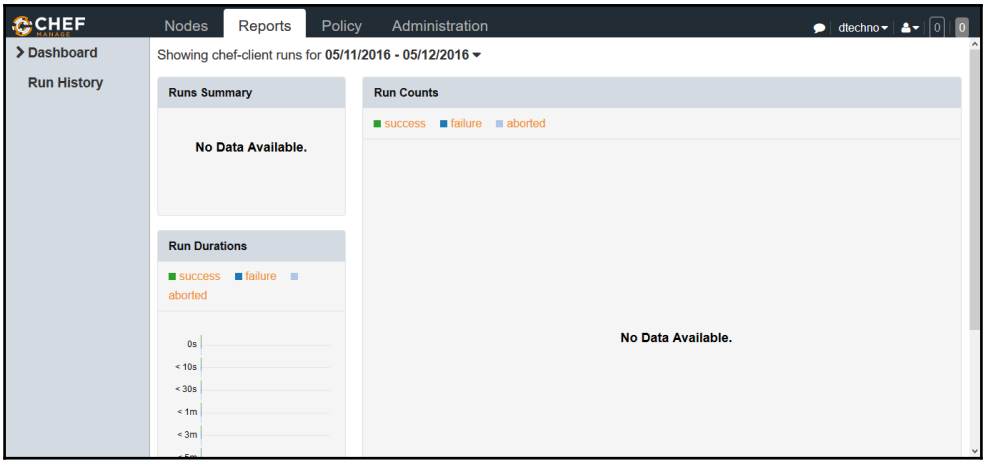
1. Click on the **Nodes** and it will show an empty list as no node is configured using Chef Server. Note this as we are going to see the same screen when we will configure a node.



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

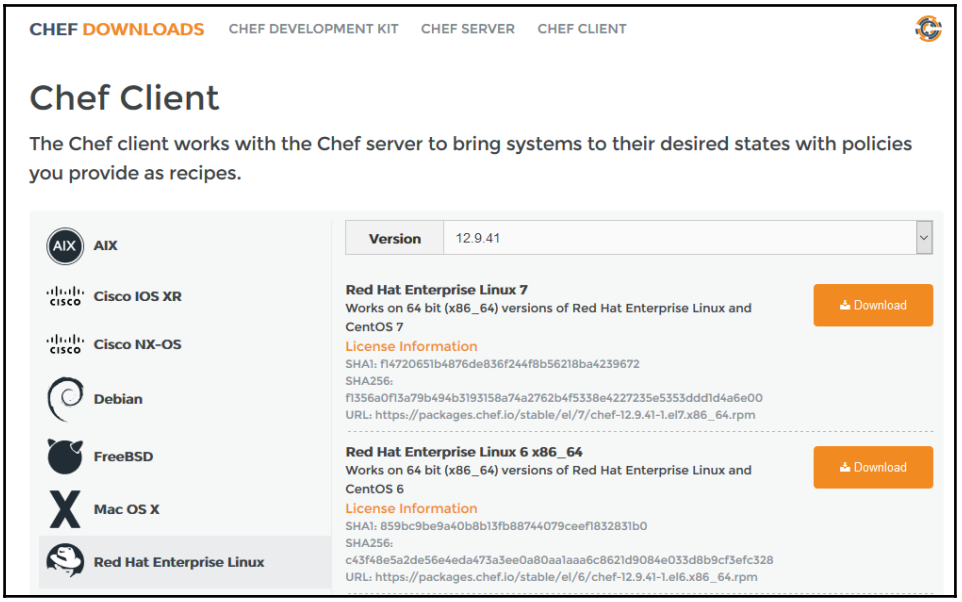


3. **Reports** tab is having no data as convergence process hasn't taken place and no success or failure data is available.

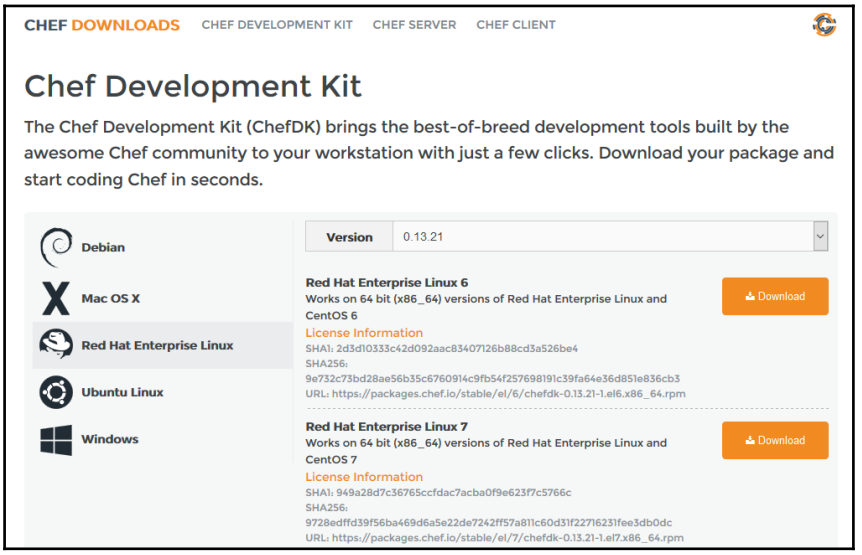


Once we have Hosted Chef account available; next step is to configure Chef workstation.

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



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



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

Installing and Configuring Chef Workstation

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

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

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

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

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

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

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

```
(Permission denied)
```

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

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

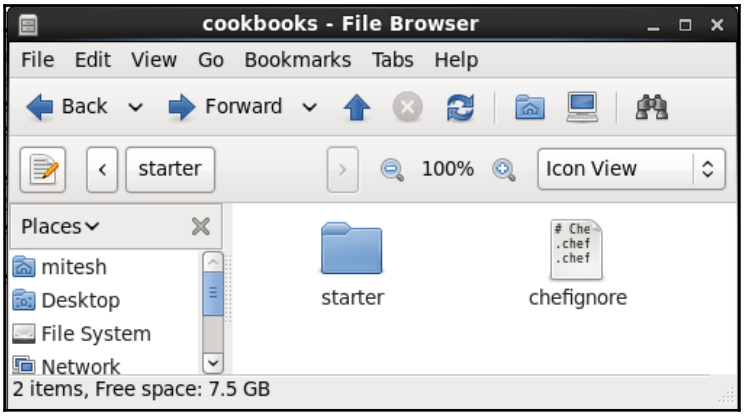
5. After successful installation, let's verify Chef client version.

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

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

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

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



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

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

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

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

Converging Chef node using Chef Workstation

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

10. We have removed chef client; verify again.

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

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

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

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

12. Verify JDK installation on node.

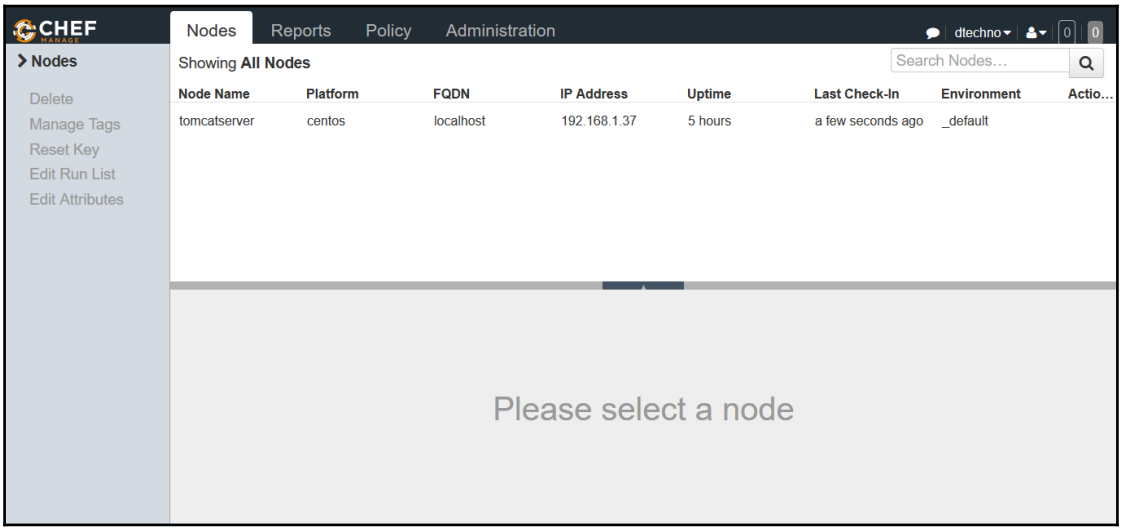
```
[root@localhost Desktop]# java -version
java version "1.7.0_75"
OpenJDK Runtime Environment (rhel-2.5.4.0.el6_6-x86_64 u75-b13)
OpenJDK 64-Bit Server VM (build 24.75-b04, mixed mode)
```

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

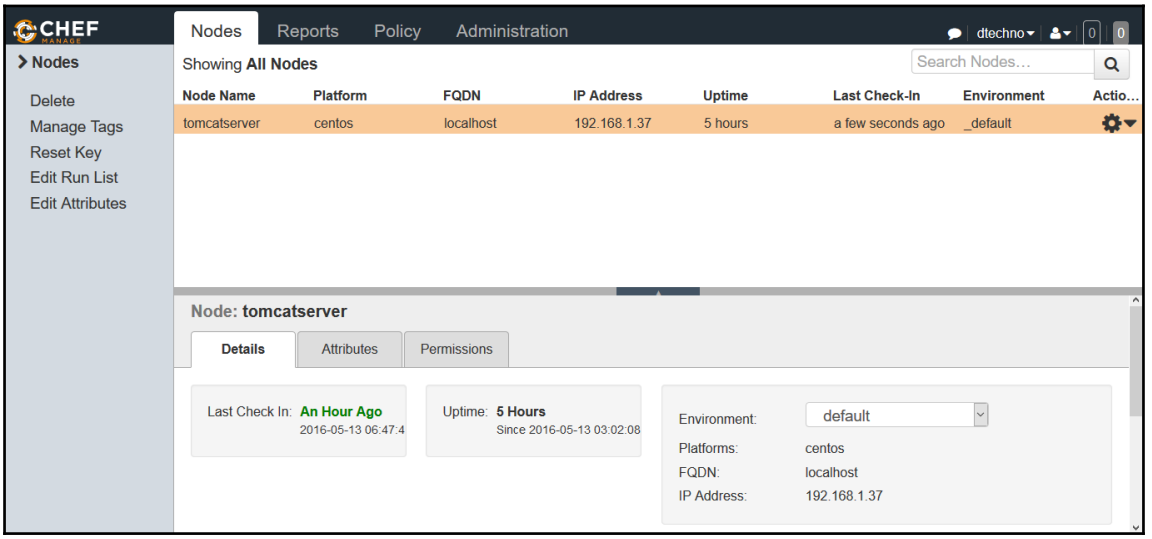
```
[root@devops1 chef-repo]# knife bootstrap 192.168.1.37 -x root -P cloud@123
 -N tomcatserver
Doing old-style registration with the validation key at /home/mitesh/chef-
 repo/.chef/dtechno-validator.pem...
Delete your validation key in order to use your user credentials instead
Connecting to 192.168.1.37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

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

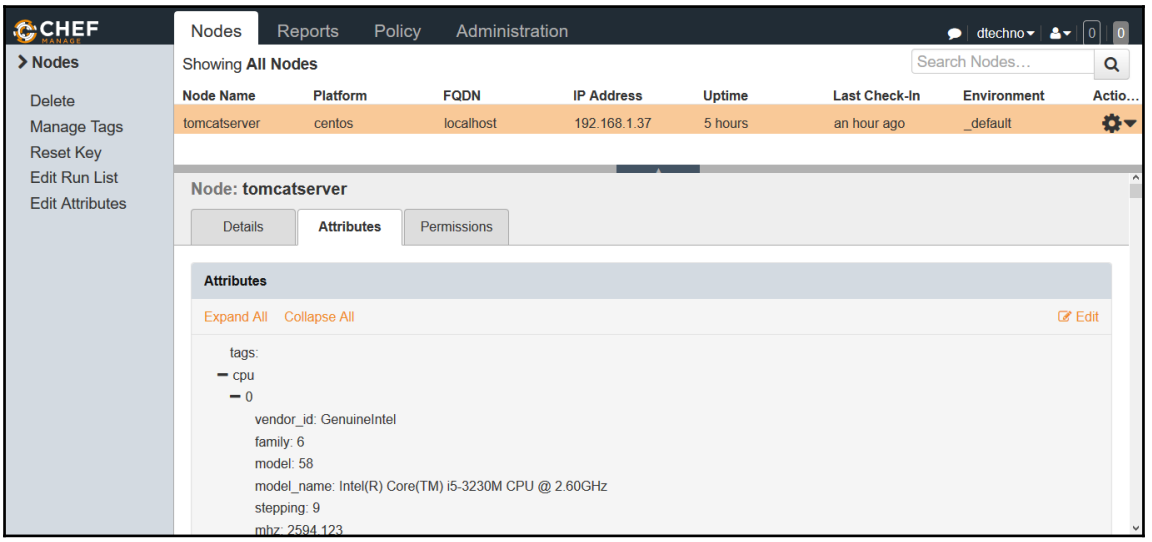
16. There was no run list or role associated with the knife command but convergence is successful.
17. Let's verify Hosted Chef account. We can see the **Node Name** and **IP Address** in the **Nodes** section of dashboard. Click on the **Nodes** and verify details:



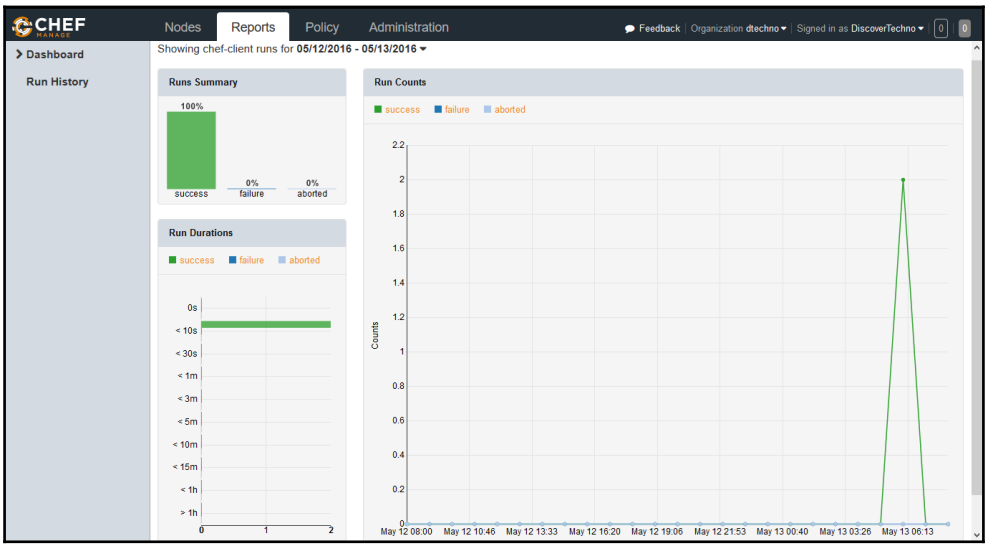
18. Select a node and check **Details**, **Attributes** associated with it, and **Permissions** as shown below:



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



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



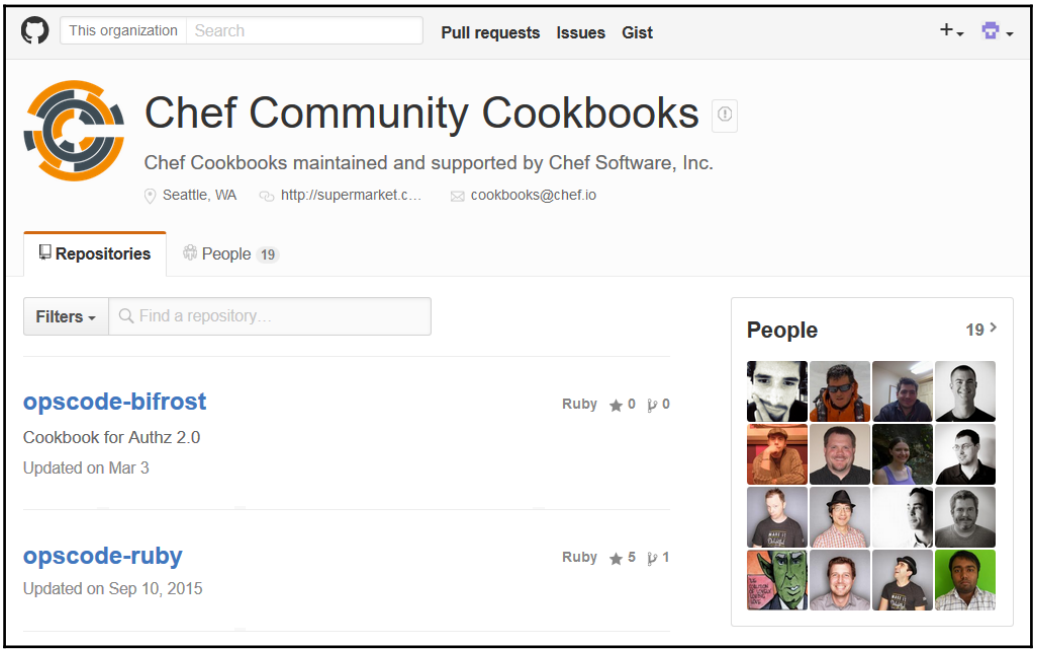
Till now, we have seen how to created Hosted Chef account, how to configure Chef workstation, and how to converge node.

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

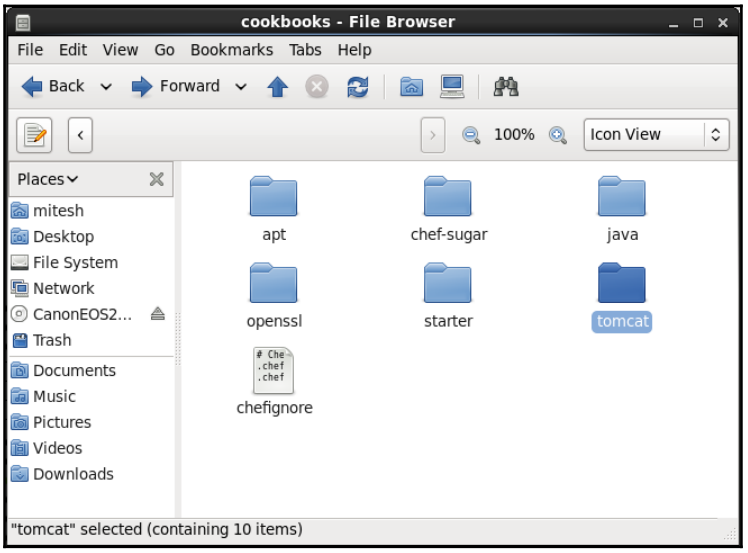
Let's revisit the context. We want to create an end to end pipeline where application source files are compiled, unit tests are executed, package file is created, new virtual machine is created, runtime environment is setup, and finally the deployment.

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

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



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



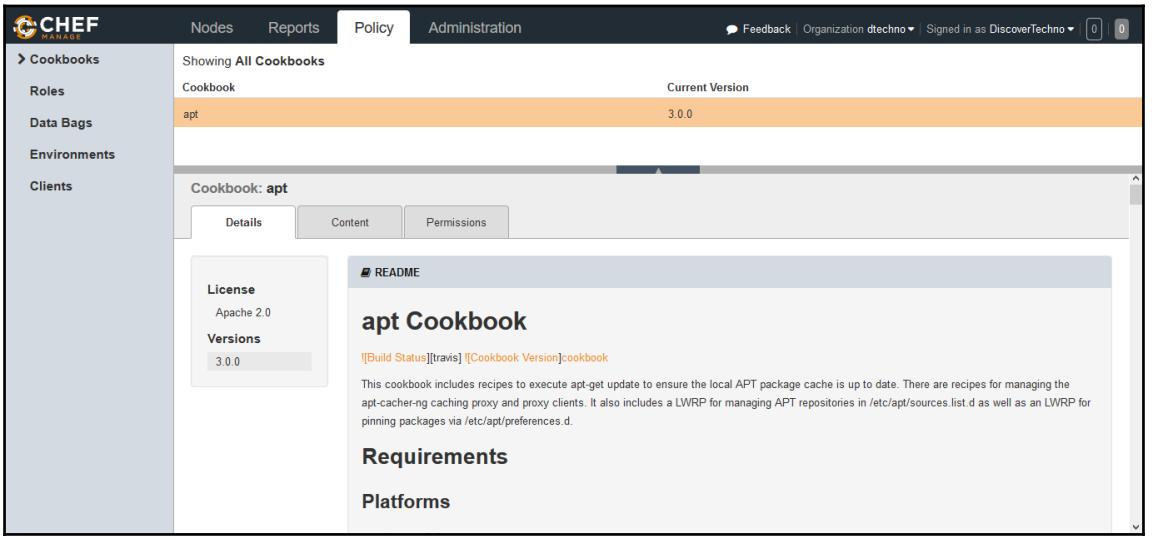
6. Go to cookbooks directory in the terminal and verify the sub-directories of community cookbooks.

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

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

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

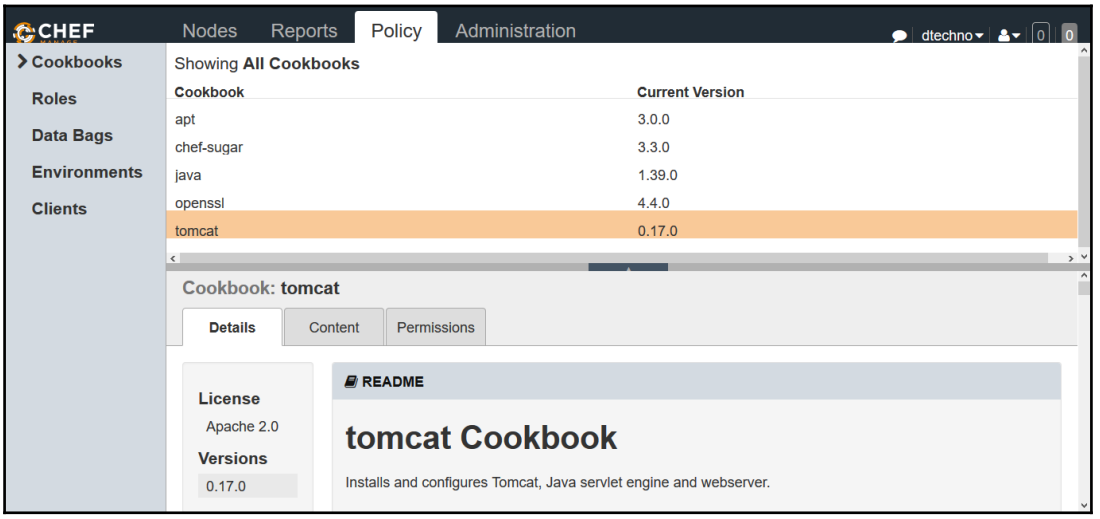
8. Verify the **Cookbooks** section in the Hosted Chef whether apt Cookbook is uploaded or not.



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

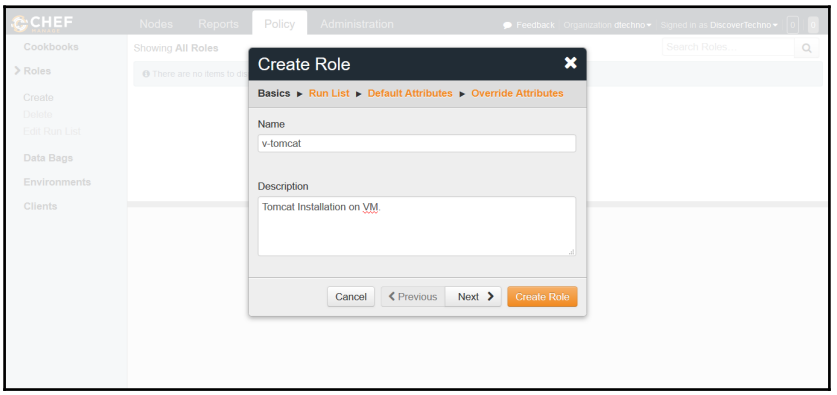
```
[root@devops1 chef-repo]# knife cookbook upload chef-sugar
Uploading chef-sugar     [3.3.0]
Uploaded 1 cookbook.
[root@devops1 chef-repo]# knife cookbook upload java
Uploading java           [1.39.0]
Uploaded 1 cookbook.
[root@devops1 chef-repo]# knife cookbook upload openssl
Uploading openssl  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

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



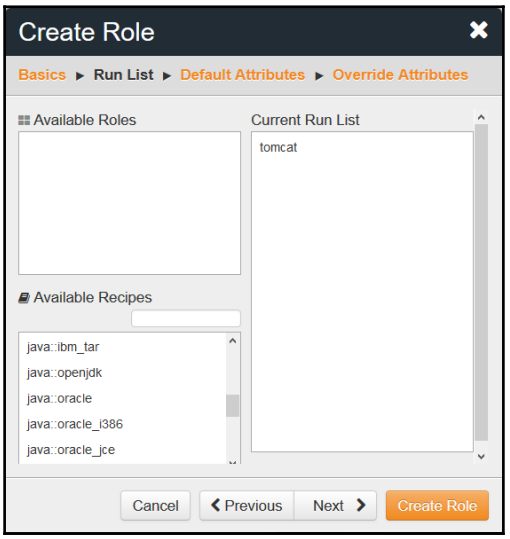
Once all cookbooks are uploaded successfully, we need to create a Role. A role is defined for a specific function and it provides a path for different patterns and workflow processes. For an example, the web server role can consist of Tomcat server recipes and any custom attributes.

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

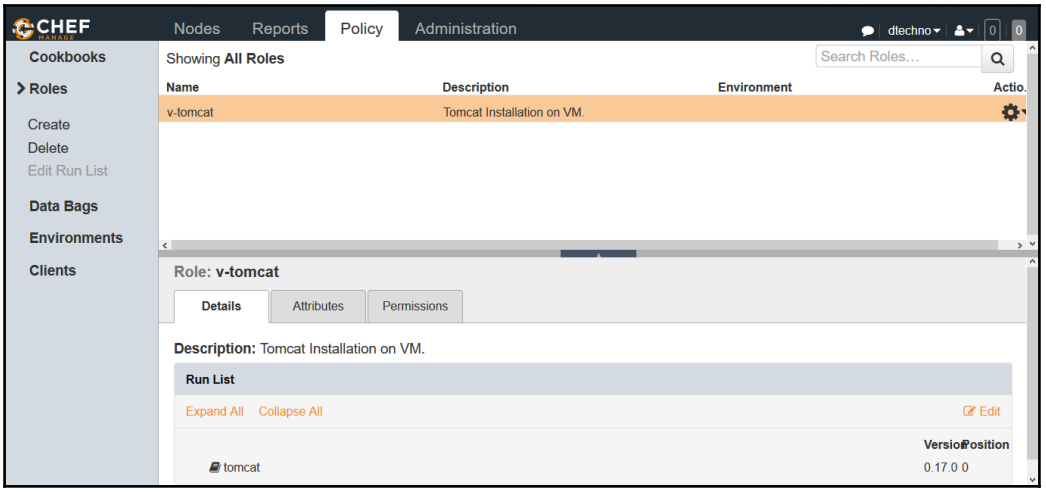


2. A **Run List** keeps roles/recipes in a proper manner and order. We can say that run-list describes the specification of a node. Select Tomcat from the **Available**

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



3. Verify Role details in Hosted Chef dashboard:



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

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

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

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

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

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

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

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

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

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

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

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

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

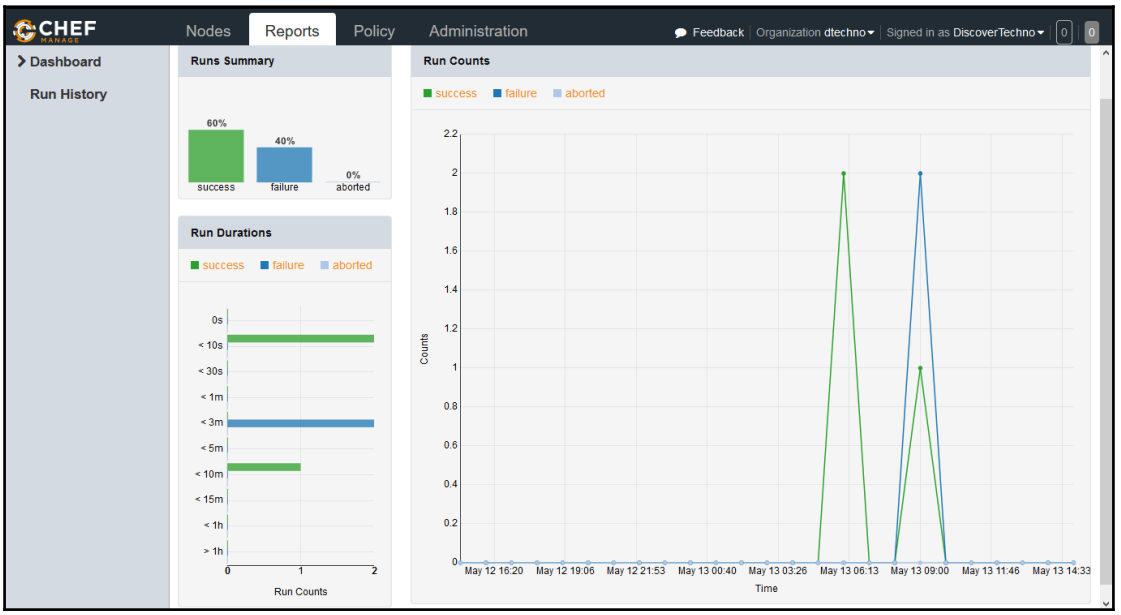
```
    <!-- An Engine represents the entry point (within Catalina) that
         processes every request.  The Engine implementation for Tomcat
         stand alone analyzes the HTTP headers included with the
         request, and passes them
@@ -97,16 +108,16 @@
         Documentation at /docs/config/engine.html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

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

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

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

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

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

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

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

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

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

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

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

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



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

Self-Test Questions

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

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

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

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

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

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

Summary

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

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

5
Installing and Configuring Docker

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

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

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

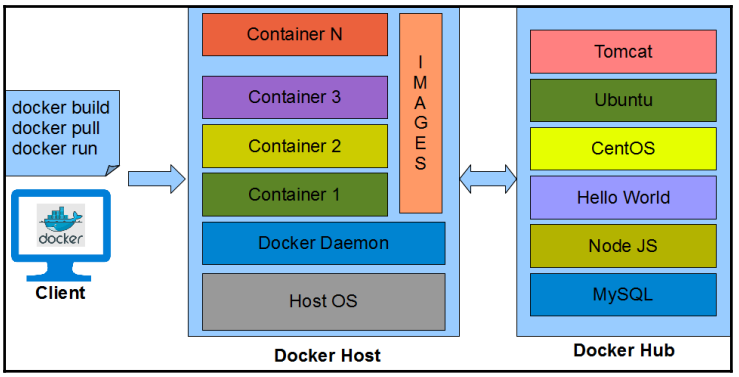
We will also cover Docker hub and basic architecture of Docker. In this chapter we will see how to use tomcat image available on Docker hub and then create a sample image with Java and tomcat installation with Dockerfile.

In this chapter, we will cover the following topics:

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

Overview of Docker Container

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



Docker has two main Components with Client Server Architecture:

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

  To run an interactive shell in the CentOS image:

  docker run -i -t ubuntu /bin/bash

  -i flag: Initiates an interactive container

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

  Image: Centos

  /bin/bash: Starts a shell

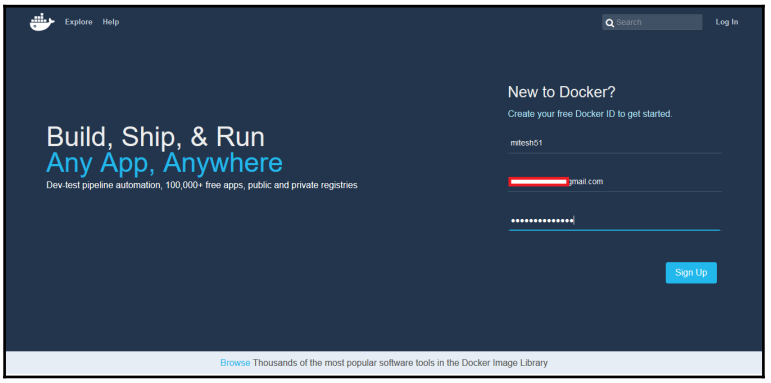
  When we run above command, it verifies whether the centos image is available locally or not. If it is not available, then it will download the image from the Docker Hub.

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

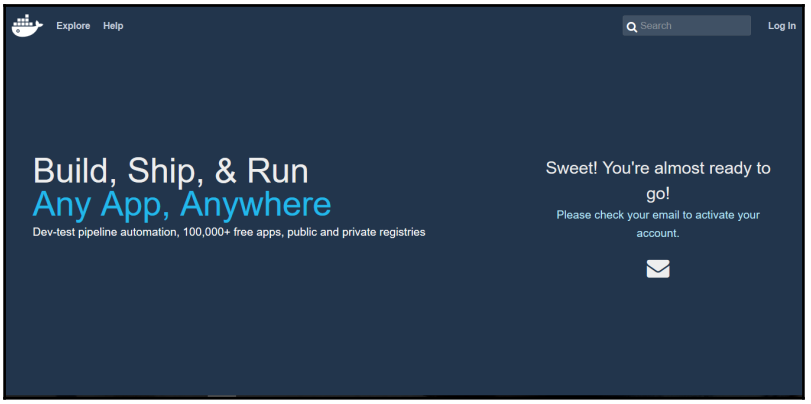
an instance of an Image with a state. It is simple to understand that container changes while Images not.

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

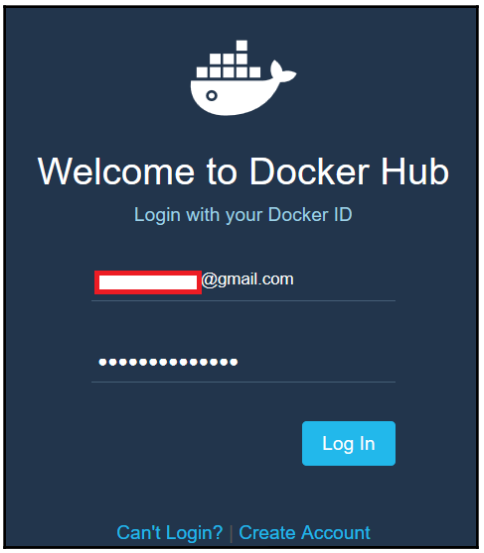
1. Let's navigate to https://hub.docker.com and sign up by providing username, email, and password details:



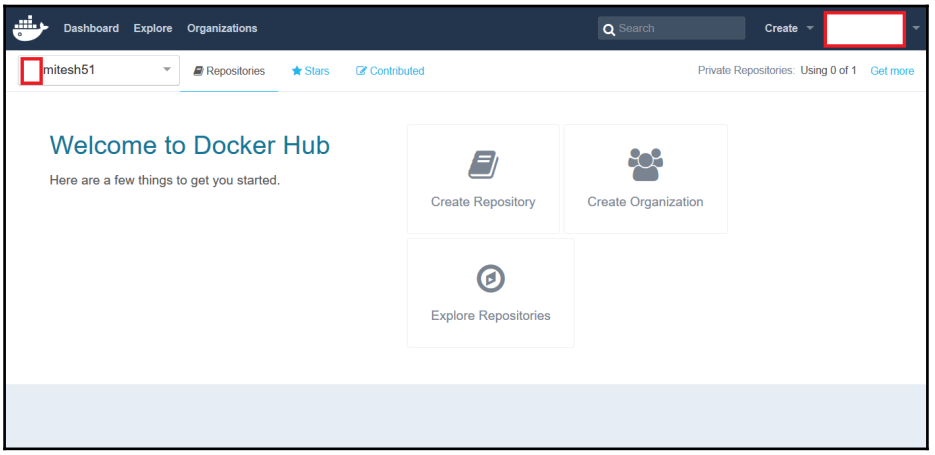
2. Activate account by clicking on the activation link sent to email id mentioned in the sign up process:



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

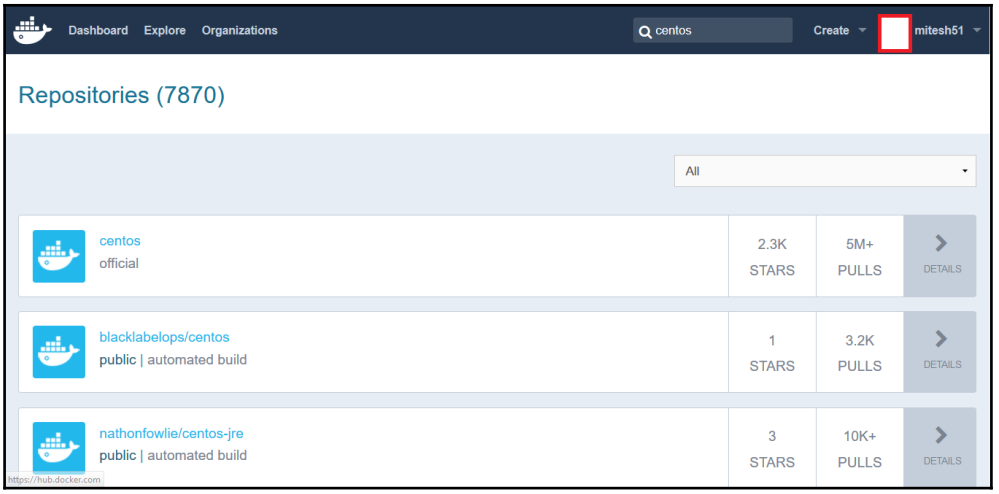


4. Following is the screenshot of Docker Dashboard. Try to explore Docker dashboard as a self exercise:



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

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



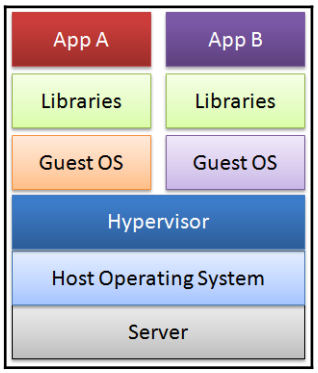
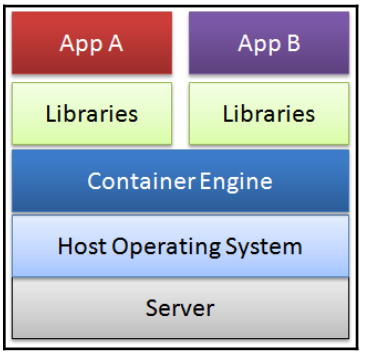
In the next section, we will see why Containers are gaining so much attraction by comparing them with Virtual Machines.

Understanding difference between Virtual Machines and Containers

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

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

Virtual Machine	**Docker**

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

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

Installing and Configuring Docker on CentOS

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

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

1. Let's install docker-io:

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

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

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

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

Install 6 Package(s)

Total download size: 5.0 M
Installed size: 20 M

Is this ok [y/N]: y

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

Running rpm_check_debug
Running Transaction Test
Transaction Test Succeeded

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

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

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

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

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

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

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

   First, start the Docker service:

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

   Verify status of Docker service:

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

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

Creating a first Docker container

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

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

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

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

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

To generate this message, Docker took the following steps:

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

Let's try something more ambitious:

1. You can run an Ubuntu container with:

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

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

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

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

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

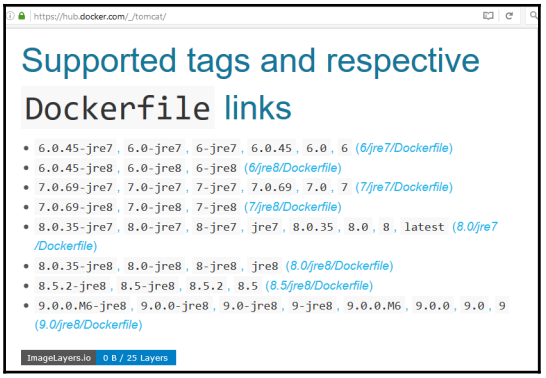
3. Use Docker images command to verify the existing images available locally:

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

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

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

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



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

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

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

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

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

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

Status: Downloaded newer image for tomcat:8.0

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

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

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

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

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

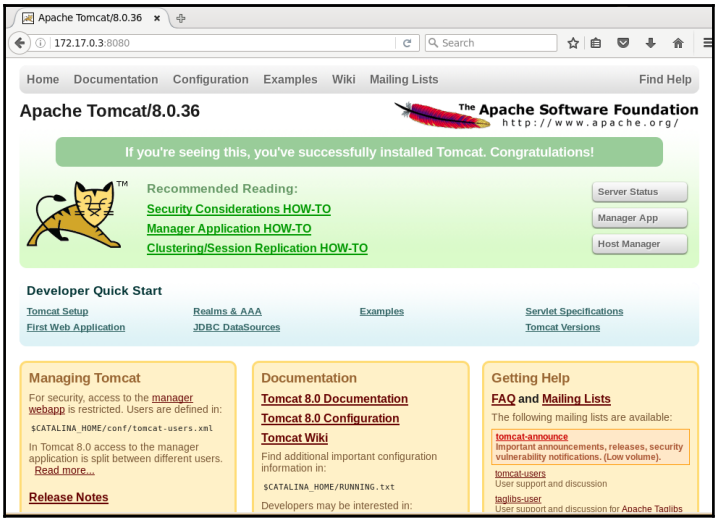
Once we have tomcat container ready, let's try to find out it's IP address so we can access the Tomcat using it.

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

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

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

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



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

Managing Containers

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

1. Use -d parameter:

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

2. Verify the container that is created recently:

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

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

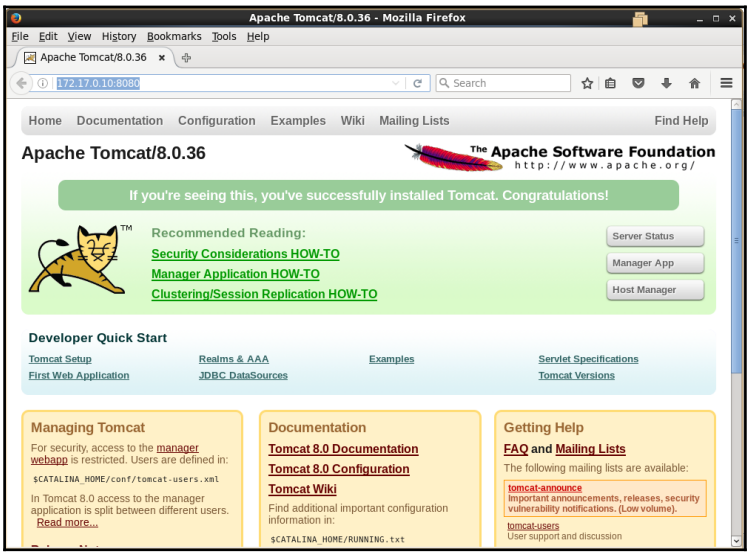
```
[root@localhost Desktop]# docker inspect 68c6d1f7bc63
[
{
    "Id": "68c6d1f7bc631613813ffb761cc833156a70e2063c2a743dd2729fe73b2873f9",
    "Created": "2016-06-21T18:25:20.73708668Z",
    "Path": "catalina.sh",
    "Args": [
        "run"
    ],
    "State": {
        "Running": true,
        "Paused":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

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

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

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

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



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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

11. Let's verify the existing images again:

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

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

Let's try with simple CentOS image:

1. Dockerfile contains following two lines:

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

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

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

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

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

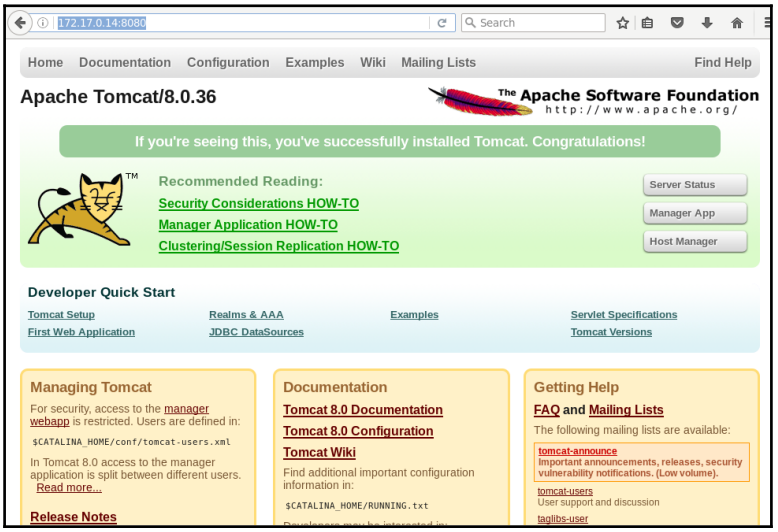
To run tomcat from the host IP address or from localhost:

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

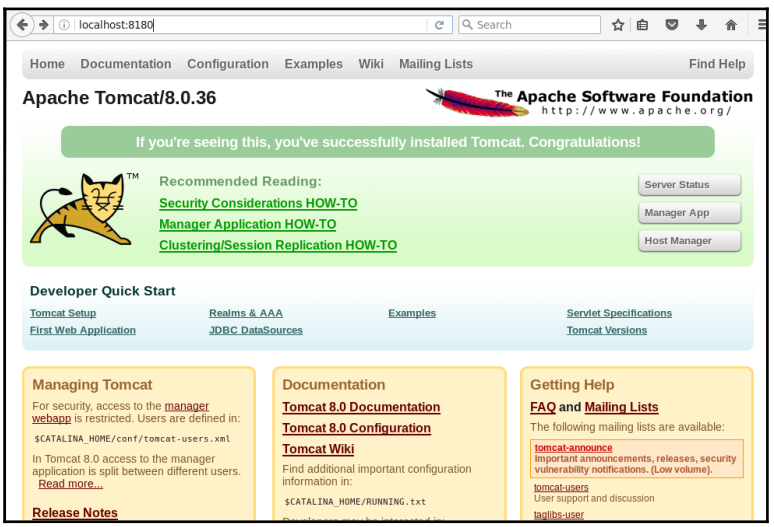
Here, 8081 is port for Host. Verify newly created container using dockerps command.

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

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



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



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

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

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

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

Self-Test Questions

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

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

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

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

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

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

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

Summary

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

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

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

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

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

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

6
Cloud Provisioning and Configuration Management with Chef

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

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

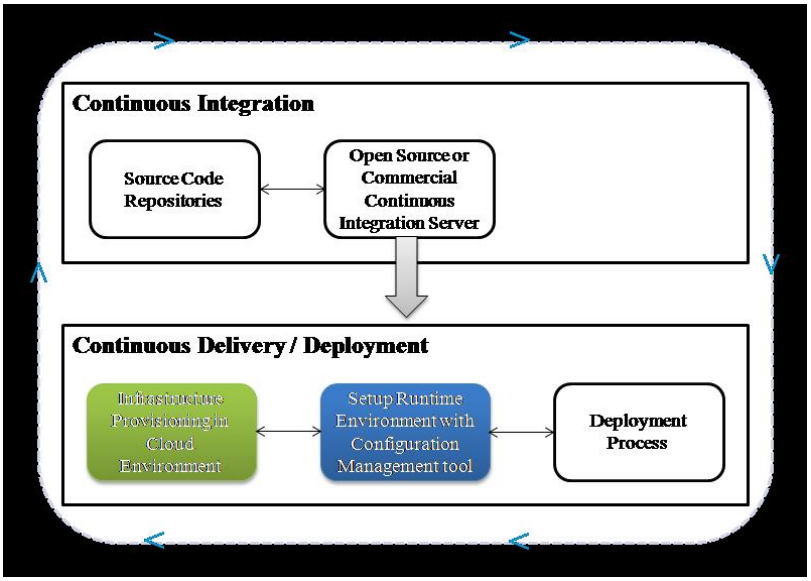
This chapter describes in detail how to install knife plugins that are used to manage cloud resources using Chef. It will cover creating instances in the AWS and Azure with the use of knife-EC2 and knife-azure plugins. It will also cover how Chef is used to manage Docker containers.

In this chapter, we will explore the following topics:

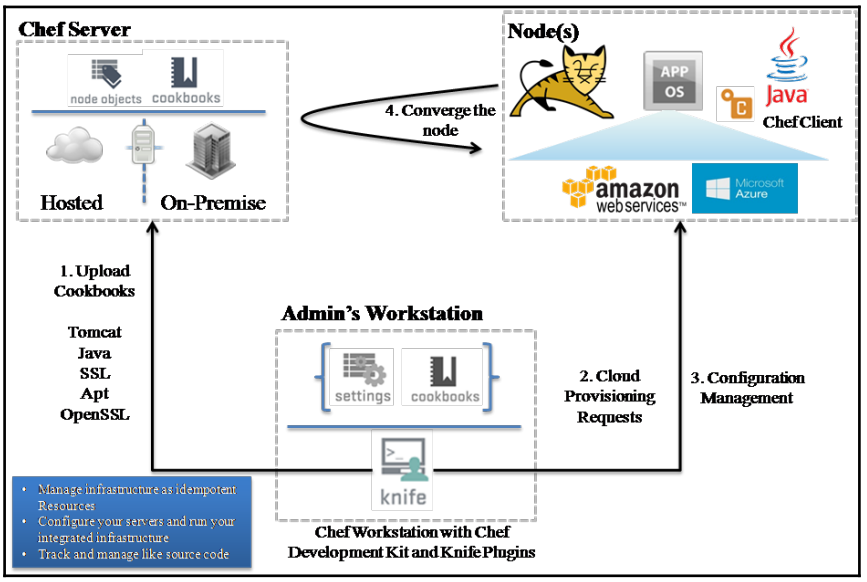
- Chef and Cloud Provisioning
- Installing Knife Plugins for Amazon EC2 and Microsoft Azure
- Creating and Configuring Virtual Machine in Amazon Web Services
- Creating and Configuring Virtual Machine in Microsoft Azure
- Manage Docker containers with Chef

Chef and Cloud Provisioning

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



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



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

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

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

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

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

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

Installing Knife Plugins for Amazon Web Services and Microsoft Azure

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

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

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

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

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

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

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

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

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

```
Chef: 12.9.41
```

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

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

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

4. Let's install additional Ruby dependencies:

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

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

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

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

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

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

6. Update the gems:

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

manage Microsoft Azure resources:

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

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

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

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

3. Install knife-wsfusion plugin:

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

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

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

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

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

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

Creating and Configuring Virtual Machine in Amazon EC2

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

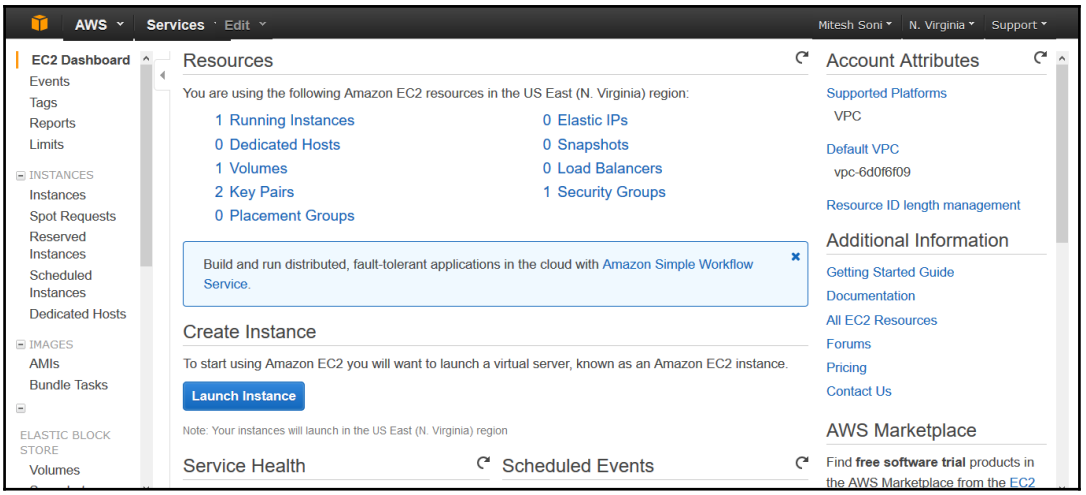
```
  ec2-52-90-219-205.compute-1.amazonaws.com
  ec2-52-90-219-205.compute-1.amazonaws.com  * service[tomcat6] action start
  ec2-52-90-219-205.compute-1.amazonaws.com    - start service service[tomcat6]
  ec2-52-90-219-205.compute-1.amazonaws.com  * execute[wait for tomcat6] action run
  ec2-52-90-219-205.compute-1.amazonaws.com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

6. Details of the newly created AWS instances:

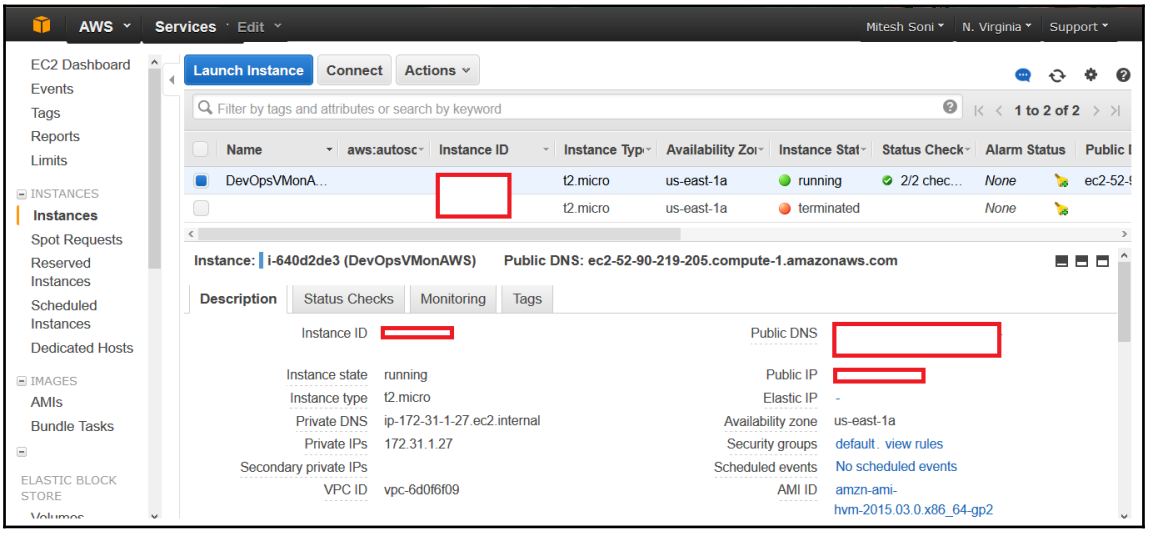
```
  Instance ID: i-640d2de3
  Flavor: t2.micro
  Image: ami-1ecae776
  Region: us-east-1
  Availability Zone: us-east-1a
  Security Groups: default
  Security Group Ids: default
  Tags: Name: DevOpsVMonAWS
  SSH Key: book
  Root Device Type: ebs
  Root Volume ID: vol-1e0e83b5
  Root Device Name: /dev/xvda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

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

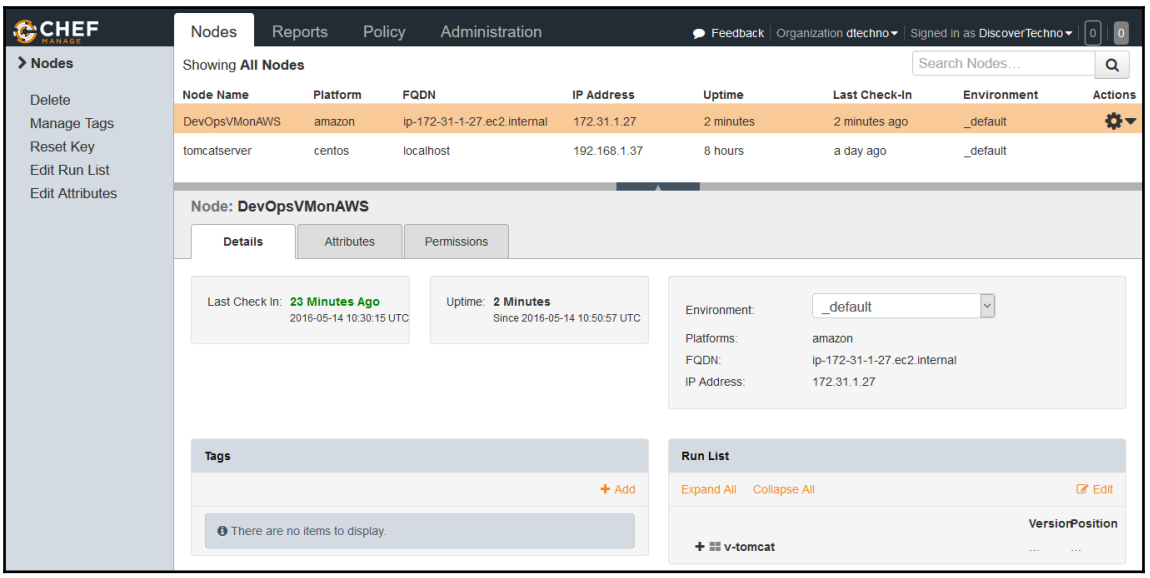
7. Go to https://aws.amazon.com/ and login with admin or IAM credentials:



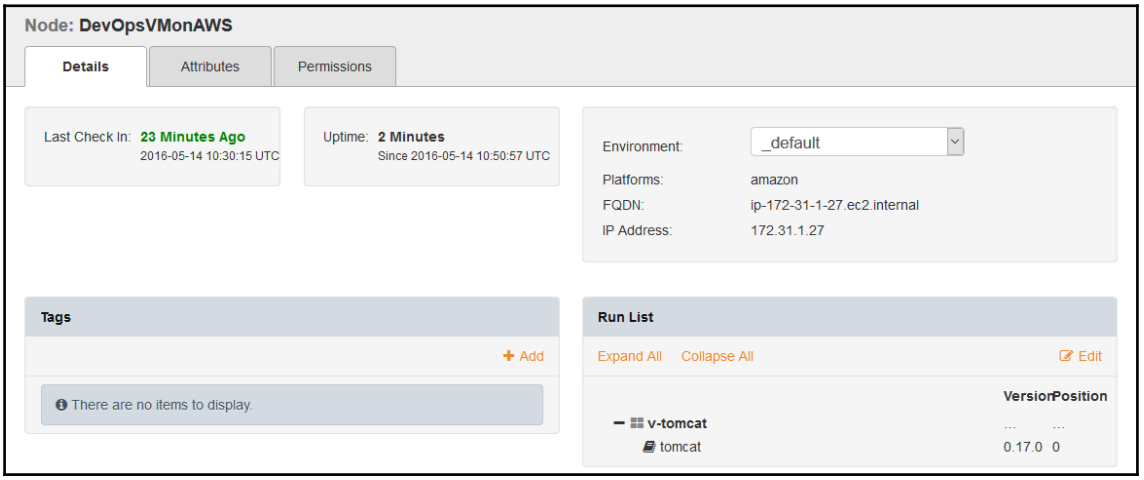
8. Click on the **Instances** in the left sidebar or **Running Instances** to get to the details about AWS instances. Verify **Name**, tag, **Public DNS** and other details that we get in the Chef client run:



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



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

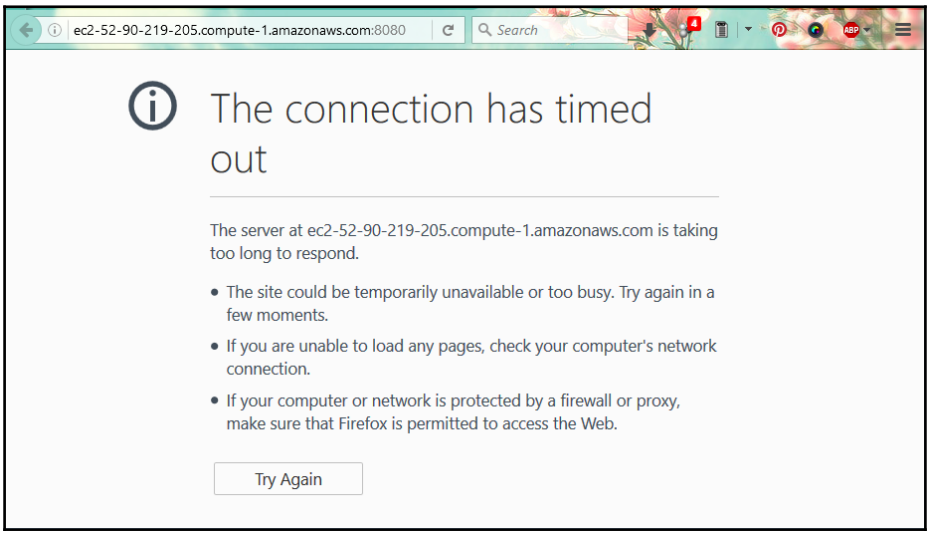


11. Check the **Attributes** section in Hosted Chef dashboard:

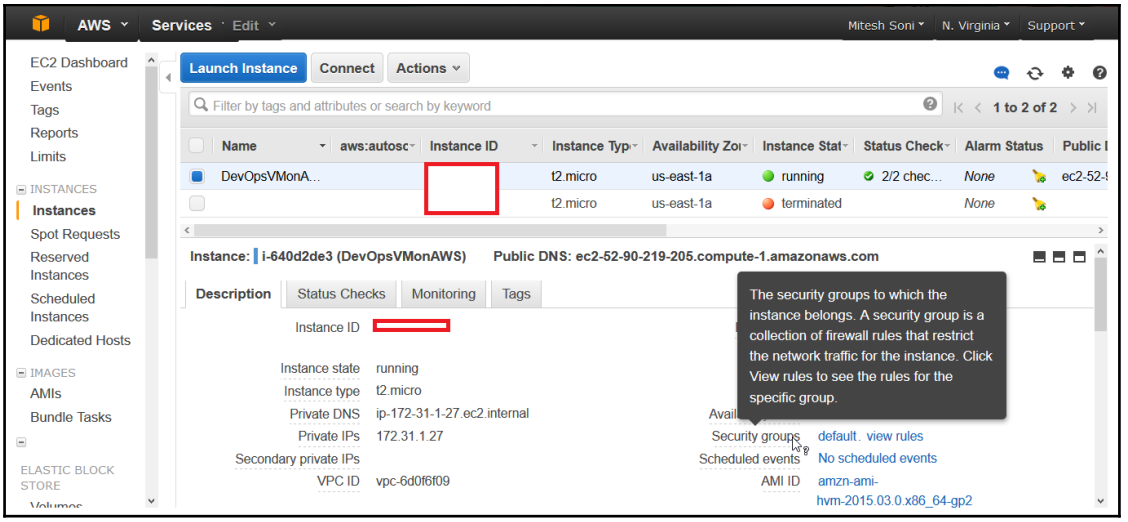
All seems to be nicely finished when it comes to creation and configuration of AWS instances and its registration on Hosted Chef.

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

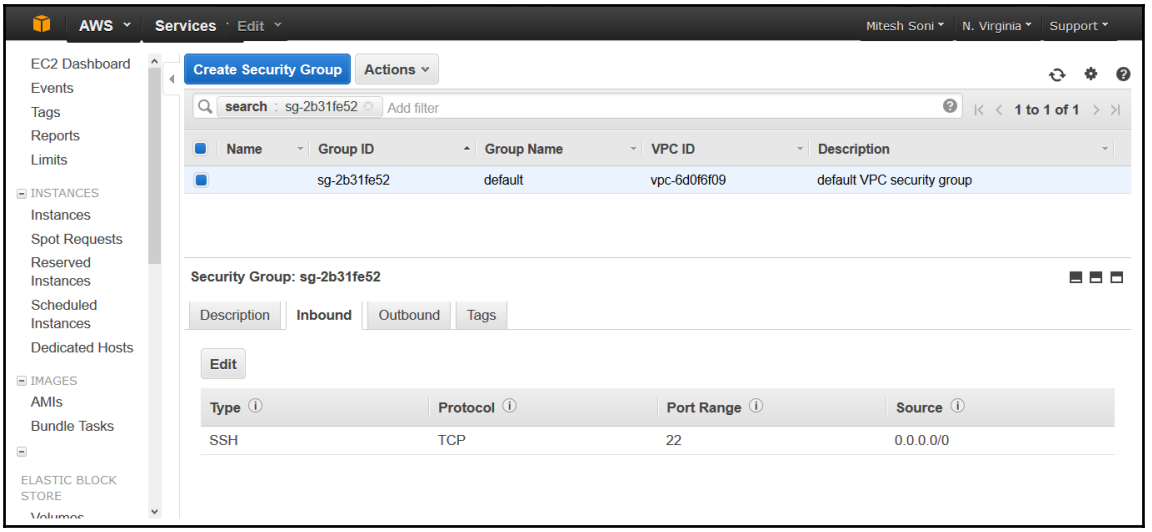
1. We get **The connection has timed out**:



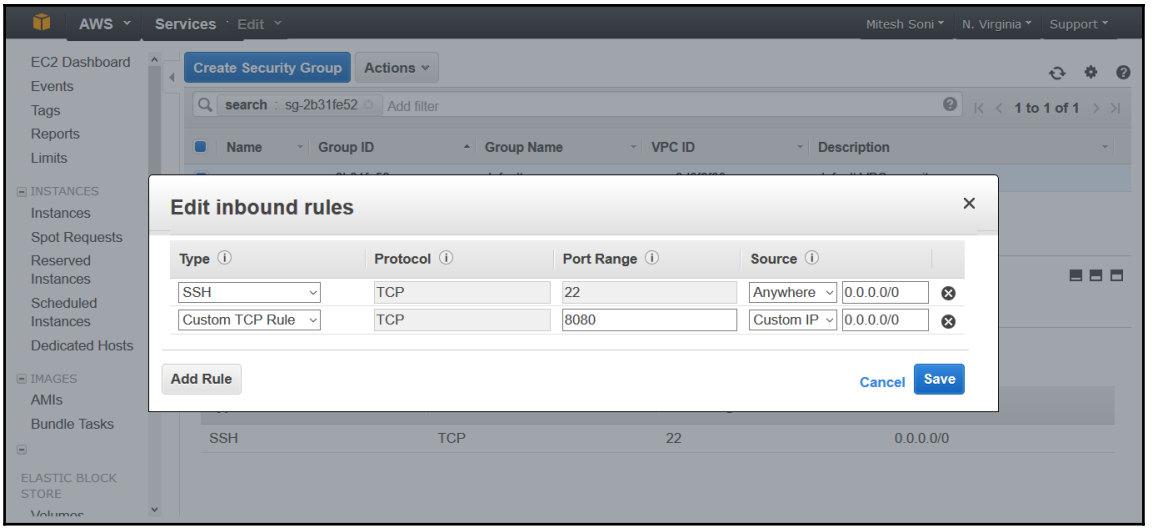
2. The reason for this is restriction of Security Groups in AWS. Verify the **security group** the AWS instance belongs to:



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



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



5. Now verify the URL and we will get the Tomcat page on AWS instance.

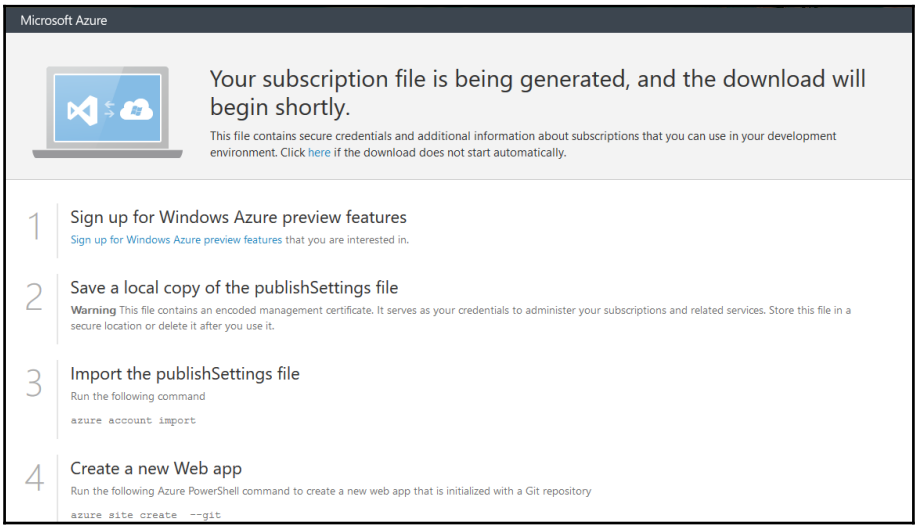
In the next section we will see how to create and configure virtual machine in Microsoft Azure.

Creating and Configuring Virtual Machine in Microsoft Azure

For knife azure plugin to communicate with Azure's REST API, we need to give Knife information regarding our Azure account and credentials:

1. Sign in to the Azure portal and download a publish settings file by visiting https://manage.windowsazure.com/publishsettings/index?client=xplat.
2. Store it on a CHef workstation in to a local file system and refer this local file by doing an entry in knife.rb:

```
knife[:azure_publish_settings_file] = "~/<name>.publishsettings"
```



3. Following are the parameters used to create a virtual machine in Microsoft Azure:

Parameter	Value	Description
–azure-dns-name	distechnodemo	DNS Name
–azure-vm-name	dtserver02	Virtual Machine Name
–azure-vm-size	Small	Virtual Machine Size
-N	DevOpsVMonAzure2	Name of the Chef Node
–azure-storage-account	classicstorage9883	Azure Storage Account
–bootstrap-protocol	cloud-api	Bootstrap Protocol
–azure-source-image	5112500ae3b842c8b9c604889f8753c3__OpenLogic-CentOS-67-20160310	Name of the Azure Source Image
–azure-service-location	Central US	Azure location to host Virtual Machine
–ssh-user	dtechno	SSH User
–ssh-password	**<Your Password>**	SSH Password
-r	role[v-tomcat]	Role
–ssh-port	22	SSH Port

[root@devops1 Desktop]# knife azure server create --azure-dns-name 'distechnodemo' --azure-vm-name 'dtserver02' --azure-vm-size 'Small' -N DevOpsVMonAzure2 --azure-storage-account 'classicstorage9883' --bootstrap-protocol 'cloud-api' --azure-source-image '5112500ae3b842c8b9c604889f8753c3__OpenLogic-CentOS-67-20160310' --azure-service-location 'Central US' --ssh-user 'dtechno' --ssh-password 'cloud@321' -r role[v-tomcat] --ssh-port 22

.....Creating new client for DevOpsVMonAzure2

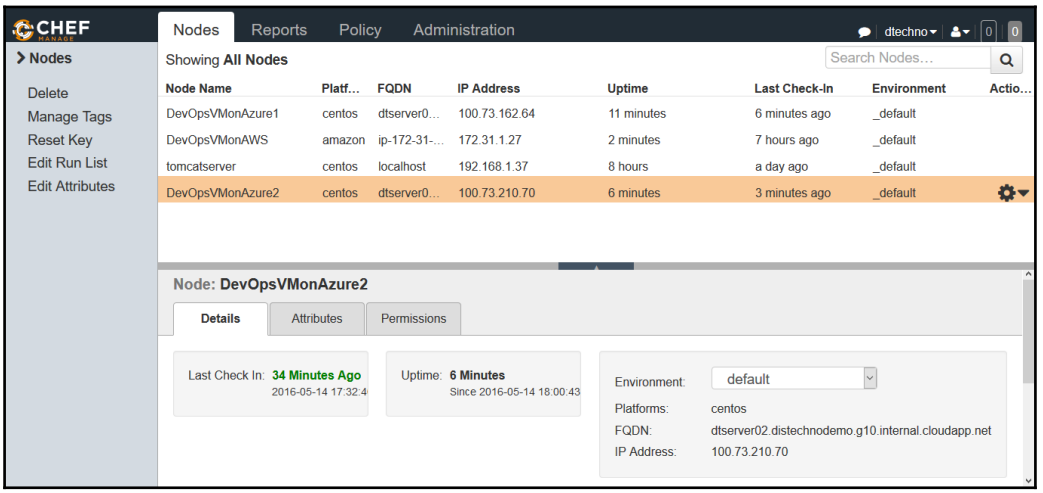
Creating new node for DevOpsVMonAzure2

.....

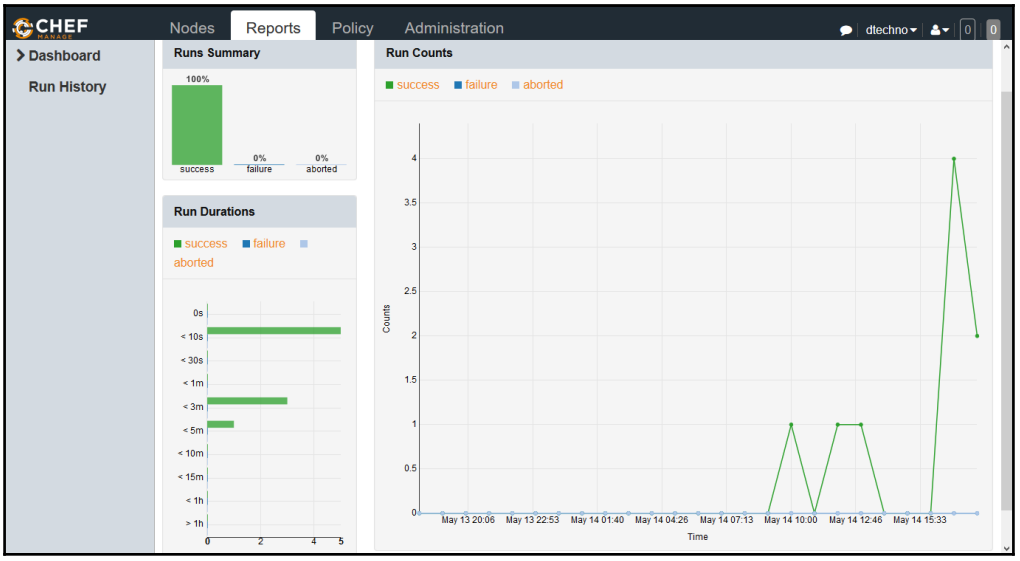
Waiting for virtual machine to reach status 'provisioning'.....vm state 'provisioning' reached after 2.47 minutes.

```
  ..
  DNS Name: distechnodemo.cloudapp.net
  VM Name: dtserver02
  Size: Small
  Azure Source Image: 5112500ae3b842c8b9c604889f8753c3__OpenLogic-
CentOS-67-20160310
  Azure Service Location: Central US
  Private Ip Address: 100.73.210.70
  Environment: _default
  Runlist: ["role[v-tomcat]"]
  Resource provisioning is going to start.
  Waiting for Resource Extension to reach status 'wagent provisioning'.....Resource extension
state 'wagent provisioning' reached after 0.17 minutes.
  Waiting for Resource Extension to reach status 'installing'.....Resource extension
state 'installing' reached after 2.21 minutes.
  Waiting for Resource Extension to reach status 'provisioning'.....Resource extension state
'provisioning' reached after 0.19 minutes.
  ..
  DNS Name: distechnodemo.cloudapp.net
  VM Name: dtserver02
  Size: Small
  Azure Source Image: 5112500ae3b842c8b9c604889f8753c3__OpenLogic-
CentOS-67-20160310
  Azure Service Location: Central US
  Private Ip Address: 100.73.210.70
  Environment: _default
  Runlist: ["role[v-tomcat]"]
  [root@devops1 Desktop]#
```

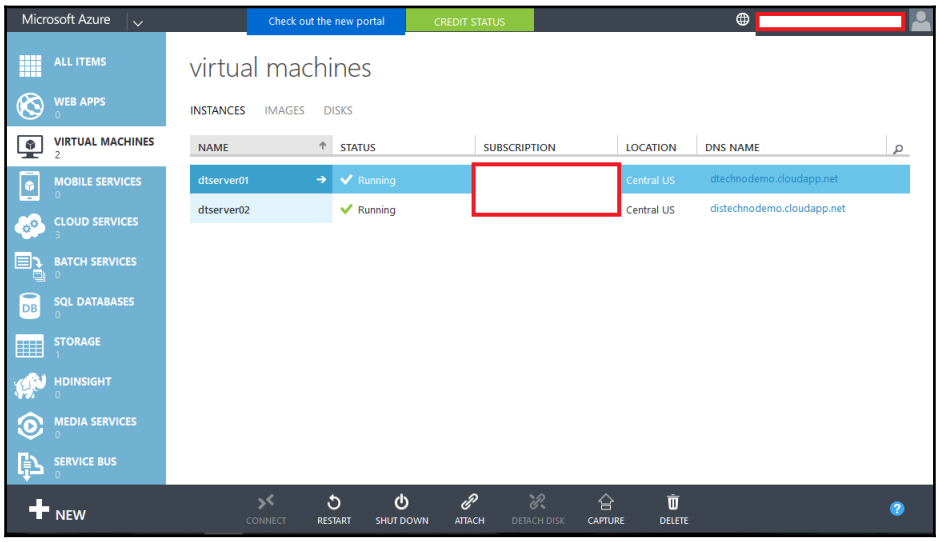
4. Go to Hosted Chef portal and click on the **Nodes** to verify whether new node is registered in the Hosted Chef server or not:



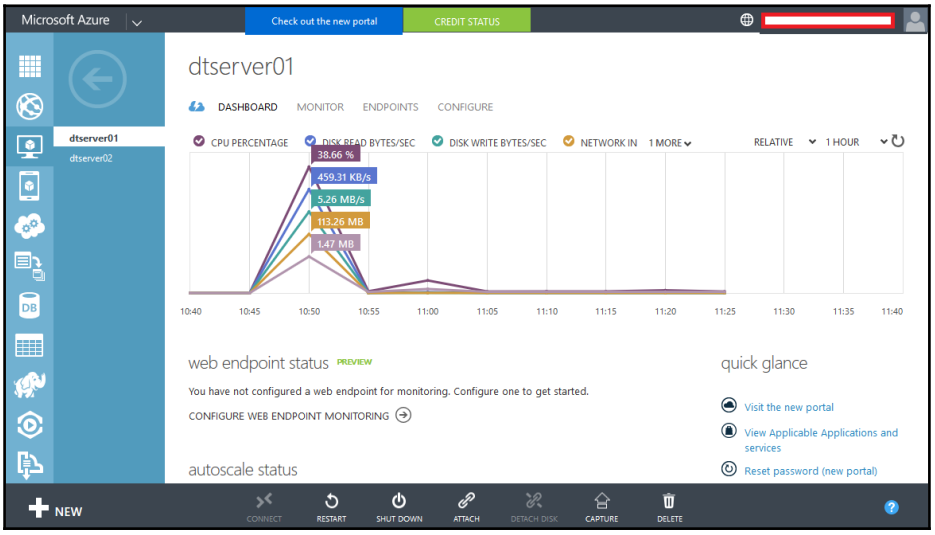
5. Click on the **Reports** section on Hosted Chef Server and verify the graphs for **Runs Summary**, **Run Durations**, and **Run Counts**:



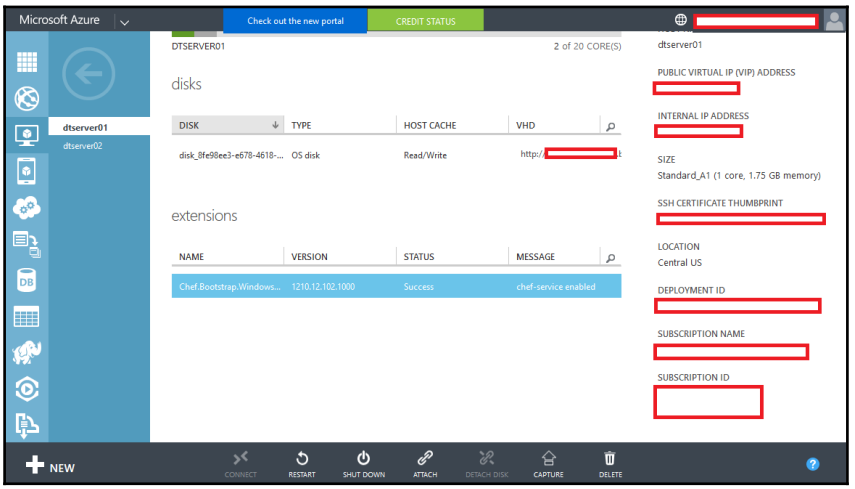
6. Now let's go to Azure Classic Portal and verify the newly created Virtual machine:



7. Click on the **VIRTUAL MACHINES** in Microsoft Azure and get details on it:



8. On the bottom of the page, verify the extensions section and see the chef-server enabled:



Verify the tomcat installation and creating virtual machine in the VMware Workstation as a self-exercise the way we did it for AWS instance.

For VMware workstation, use https://github.com/chipx86/knife-wsfusion for reference.

Just to remind again, we are now close to our main objective and that is end to end automation of application deployment pipeline. We have covered Continuous Integration, Cloud Provisioning, Containers, and Configuration Management. Remaining is actual deployment, monitoring, and orchestration of all activities involved in the end to end automation.

Docker Container

Docker containers are extremely lightweight. We are going to use Tomcat as a web application server to deploy Petclinic application. Docker Hub already have the tomcat image so we are not going to configure too many things except users for accessing Tomcat Manager:

1. In Tomcat-users.xml add role and user as shown in below section:

```
<?xml version='1.0' encoding='utf-8'?>
<!–
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements. See the NOTICE file distributed with this work for additional information regarding copyright ownership.
The ASF licenses this file to You under the Apache License, Version 2.0 (the “License”); you may not use this file except in compliance with the License. You may obtain a copy of the License at
http://www.apache.org/licenses/LICENSE-2.0
Unless required by applicable law or agreed to in writing, software distributed under the License is distributed on an “AS IS” BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
See the License for the specific language governing permissions and limitations under the License.
–>
<tomcat-users xmlns=”http://tomcat.apache.org/xml”
xmlns:xsi=”http://www.w3.org/2001/XMLSchema-instance”
xsi:schemaLocation=”http://tomcat.apache.org/xml tomcat-users.xsd”
version=”1.0”>
<!–
NOTE: By default, no user is included in the “manager-gui” role required to operate the “/manager/html” web application. If you wish to use this app, you must define such a user – the username and password are arbitrary. It is strongly recommended that you do NOT use one of the users in the commented out section below since they are intended for use with the examples web application.
–>
<!–
NOTE: The sample user and role entries below are intended for use with the examples web application. They are wrapped in a comment and thus are ignored when reading this file. If you wish to configure these users for use with the examples web application, do not forget to remove the <!.. ..> that surrounds them. You will also need to set the passwords to something appropriate.
–>
<role rolename=”manager-gui”/>
<user username=”admin” password=”admin@123” roles=”manager-gui”/>
</tomcat-users>
```

2. Now we are going to use the image available in the Docker hub and add the tomcat-sers.xml to /usr/local/tomcat/conf/tomcat-users.xml.Create a Dockerfile as shown below:

```
FROM tomcat:8.0
MAINTAINER Mitesh <mitesh.xxxx @xxxxx.com>
COPY tomcat-users.xml /usr/local/tomcat/conf/tomcat-users.xml
```

3. Once everything is ready, use docker build command to build a new image:

```
[root@localhost mitesh]# docker build -t devopstomcatnew .
Sending build context to Docker daemon 8.192 kB
Sending build context to Docker daemon
Step 0 : FROM tomcat:8.0
 ---> 5d4577339b14
Step 1 : MAINTAINER Mitesh <mitesh.soni@outlook.com>
 ---> Running in 9430cac12c4c
 ---> c63f90db4c14
Removing intermediate container 9430cac12c4c
Step 2 : COPY tomcat-users.xml /usr/local/tomcat/conf/tomcat-users.xml
 ---> eb50c4ceefb5
Removing intermediate container 7f31aed05097
Successfully built eb50c4ceefb5
You have new mail in /var/spool/mail/root
```

4. Image is successfully built. Let's verify using docker images command:

```
[root@localhost mitesh]# docker images
REPOSITORY        TAG              IMAGE ID          CREATED           VIRTUAL SIZE
devopstomcatnew   latest           eb50c4ceefb5      10 seconds ago    359.2 MB
devopstomcat8     latest           f3537165ebe7      10 minutes ago    344.6 MB
devopstomcat      latest           400f097677e9      9 days ago        658.4 MB
tomcat6           latest           400f097677e9      9 days ago        658.4 MB
tomcat            9.0              ce07000625c6      2 weeks ago       344.6 MB
centos            latest           2a332da70fd1      4 weeks ago       196.7 MB
ubuntu            latest           686477c12982      8 weeks ago       120.7 MB
hello-world       latest           f1d956dc5945      9 weeks ago       967 B
```

5. Create a container from the newly created tomcat image.
6. Verify existing containers using docker ps and docker ps -a command:

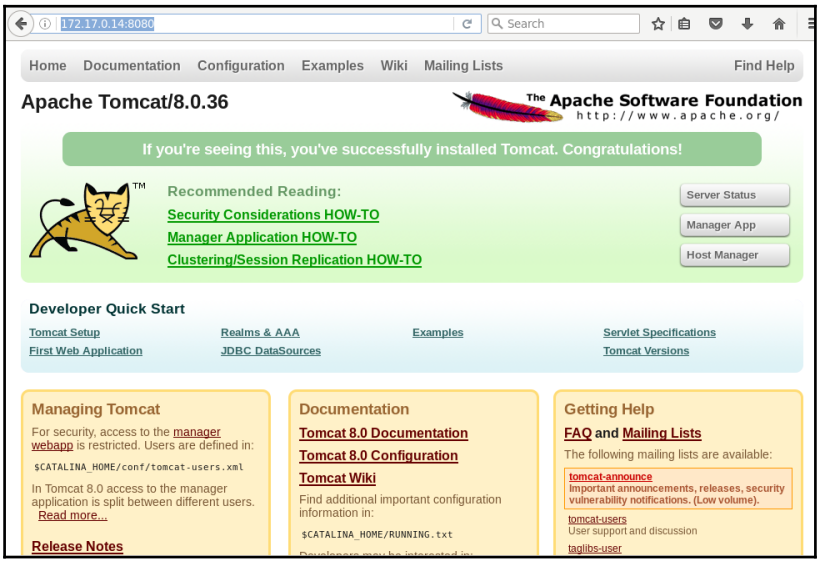
```
[root@localhost mitesh]# docker ps
CONTAINER ID      IMAGE          COMMAND          CREATED
 STATUS           PORTS          NAMES
You have new mail in /var/spool/mail/root
[root@localhost mitesh]# docker ps -a
CONTAINER ID      IMAGE          COMMAND          CREATED
 STATUS           PORTS          NAMES
[root@localhost mitesh]# docker run -p 8180:8080 -d --name devopstomcat1
 devopstomcatnewb5f054ee4ac36d67279db10497fe7a780aecf2a72a7f52fa31ee80
```

```
c618d98e4a
```

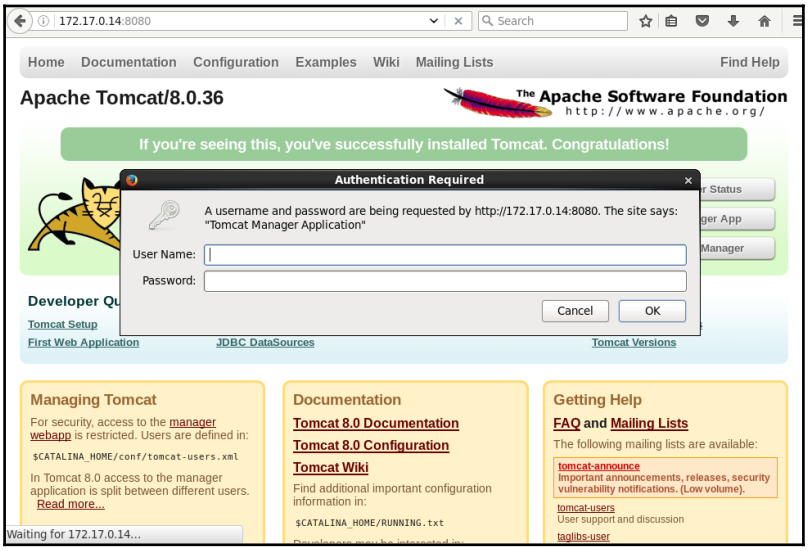
7. Verify existing containers using docker ps and docker ps -a command:

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

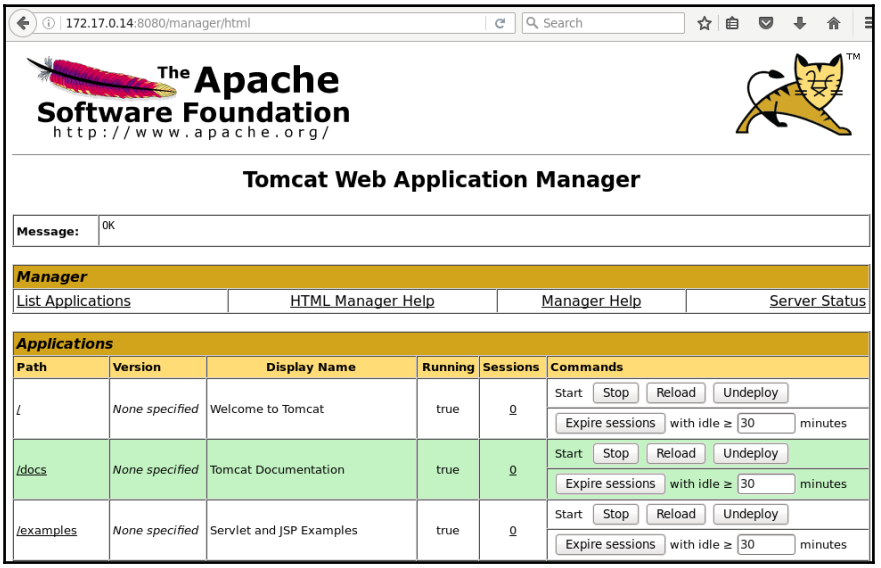
8. Use docker inspect b5f054ee4ac3 command to get the IP address and browse tomcat web server using IP address and Port:



9. Click on the **Manager App** button. It will ask for **User Name** and **Password**. Give Inputs and click on **OK**:



10. Now we can access the tomcat manager application:



We can use **Tomcat Manager Application** to deploy application. Till now we have seen Continuous Integration, Configuration Management, Containers, and Cloud Provisioning.

Next, we will see application deployment using different methods, Monitoring, and End to end automation pipeline using Orchestration.

Self-Test Questions

1. Which of the followings are benefits of Chef Configuration Management?
2. Easy policy enforcement with centralized control
3. Enable setup of consistent runtime environment
4. Enable easy restoration of environments
5. Enable disaster recovery and business continuity
6. Community-based cookbooks and recipes
7. All of the Above

1. Which two parameters are configured for Amazon EC2 credentials for knife-ec2 in knife.rb file?
2. knife[:aws_access_key_id] = “Your AWS Access Key ID”
3. knife[:aws_secret_access_key] = “Your AWS Secret Access Key”
4. Both a and b

1. Which of the followings are knife EC2 commands?
2. knife ec2 flavor list (options)
3. knife ec2 server create (options)
4. knife ec2 server delete SERVER [SERVER] (options)
5. knife ec2 server list (options)
6. All of the Above

1. State True or False: rvm use command is used to set the Ruby version.
2. True
3. False

1. Which of the followings are knife Azure commands?
2. knife azure server create (options)
3. knife azure server delete SERVER [SERVER] (options)
4. knife azure server list (options)
5. knife azure image list (options)
6. All of the Above

1. State True or False: In knife ec2 server create command -I parameter is used for Type of Virtual Machine
2. True
3. False

1. State True or False: In knife ec2 server create command -N parameter is used for Name of the Chef Node
2. True
3. False

Summary

In this chapter, we have covered how to provision resources in Cloud and configure them. We used knife ec2 and knife azure plugin to create virtual machine in AWS and Microsoft Azure. We used Docker Hub Tomcat image to build a new image with `tomcat-users.xml` file which has `role` and `user` configured to access Tomcat Manager web app.

In the next Chapter, we will cover different methods to deploy an application in Tomcat web container. Just to revisit the end goal of the book: End to End automation using application deployment pipeline.

7
Deploying Application in AWS, Azure, and Docker

Ultimate automation… will make our modern industry as primitive and outdated as the stone age man looks to us today.
- Albert Einstein

Finally, we are at the *Business* end of the book and our focus is on deployment automation, monitoring, and orchestration.

Why?

Answer is to achieve **End to End Application lifecycle Automation** or **End to End Deployment Automation**.

First we will go step by step to deploy our Petclinic application into remote tomcat server. Once that is achieved, it can be used as common practice for all. This chapter describes in detail all steps required to deploy sample application into different environment once configuration management tool prepare it for the final deployment. We will also learn how to deploy application in different environments such as cloud or container based environment.

This chapter will also cover on how to Deploy Application on Platform as a Service model. We will deploy application in AWS Elastic Beanstalk.

In this chapter, we will cover the following topics:

- Pre-requisites – To deploy application on Remote Server
- Deploying Application in AWS
- Deploying Application in Microsoft Azure

- Deploying Application in Docker Container

Pre-requisites – To deploy application on Remote Server

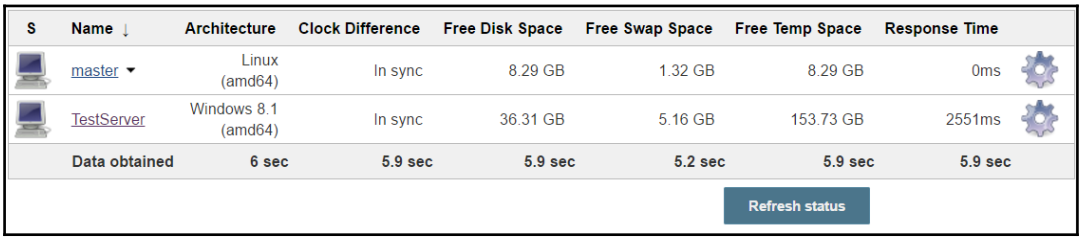
To deploy an application on remote server, let's take the following steps:

1. First, let's start an agent on Windows machine, open command prompt and run the command as it is given in **Manage Nodes** of Jenkins dashboard. Change URL appropriately:

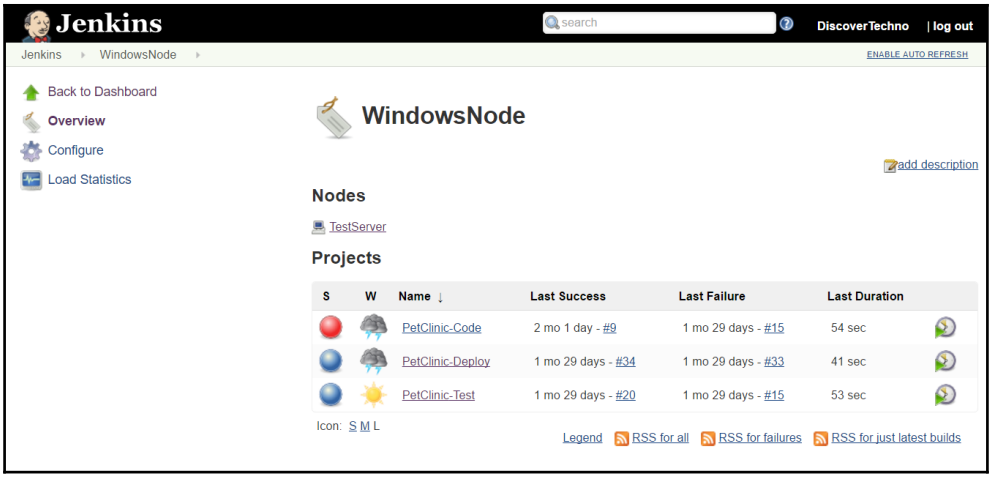
```
java -jar slave.jar -jnlpUrl
http://192.168.0.100:8080/computer/TestServer/slave-agent.jnlp -secret
65464e02c58c85b192883f7848ad2758408220bed2f3af715c01c9b01cb72f9b
```

```
Jul 06, 2016 8:56:54 PM hudson.remoting.jnlp.MaincreateEngine
INFO: Setting up slave: TestServer
Jul 06, 2016 8:56:54 PM hudson.remoting.jnlp.Main$CuiListener<init>
INFO: Jenkins agent is running in headless mode.
Jul 06, 2016 8:56:54 PM hudson.remoting.jnlp.Main$CuiListener status
INFO: Locating server among [http://192.168.1.34:8080/,
http://192.168.0.100:8080/]
Jul 06, 2016 8:57:15 PM hudson.remoting.jnlp.Main$CuiListener status
INFO: Handshaking
Jul 06, 2016 8:57:15 PM hudson.remoting.jnlp.Main$CuiListener status
INFO: Connecting to 192.168.0.100:33903
Jul 06, 2016 8:57:15 PM hudson.remoting.jnlp.Main$CuiListener status
INFO: Trying protocol: JNLP3-connect
Jul 06, 2016 8:57:16 PM hudson.remoting.jnlp.Main$CuiListener status
INFO: Server didn't accept the handshake: Unknown
protocol:Protocol:JNLP3-connect
Jul 06, 2016 8:57:16 PM hudson.remoting.jnlp.Main$CuiListener status
INFO: Connecting to 192.168.0.100:33903
Jul 06, 2016 8:57:16 PM hudson.remoting.jnlp.Main$CuiListener status
INFO: Trying protocol: JNLP2-connect
Jul 06, 2016 8:57:16 PM hudson.remoting.jnlp.Main$CuiListener status
INFO: Connected
```

2. Now our Agent is connected to Master. Let's verify the status of Agent on the Master Node where Jenkins is running.:



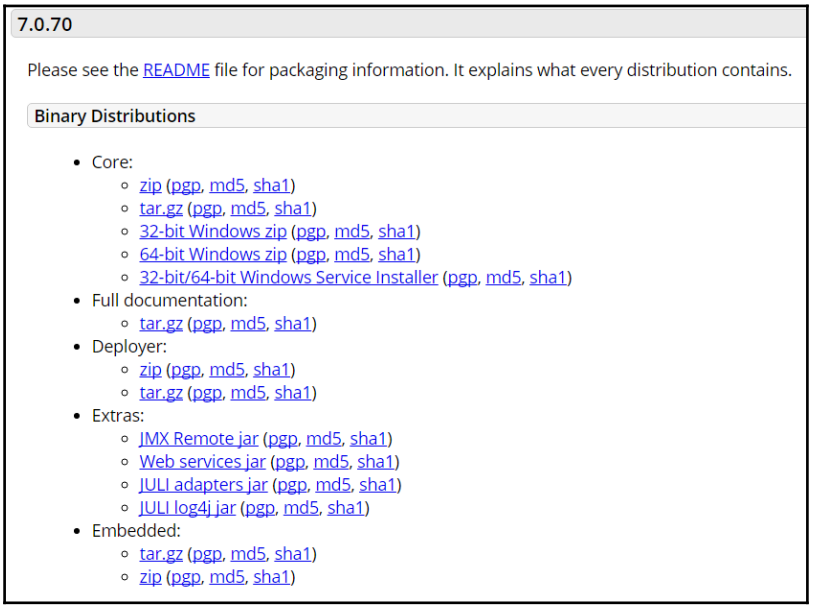
3. Click on the Agent **TestServer** and get all the details regarding projects tied to the Agents as shown in below screenshot:



Once we have Agent node ready, let's prepare a remote server ready by downloading and setting up tomcat server.

In our case we need not to do it for Cloud instances as they will be configured using Chef configuration management tool. This is more understanding perspective on how we used to do it earlier and how all installation and other activities can be automated using Chef. Let's take a step-by-step tour:

1. Download Tomcat 7 version from https://tomcat.apache.org/download-70.cgi. We are going to use Deploy plugin from Jenkins and it requires specific versions of Tomcat for deployment.



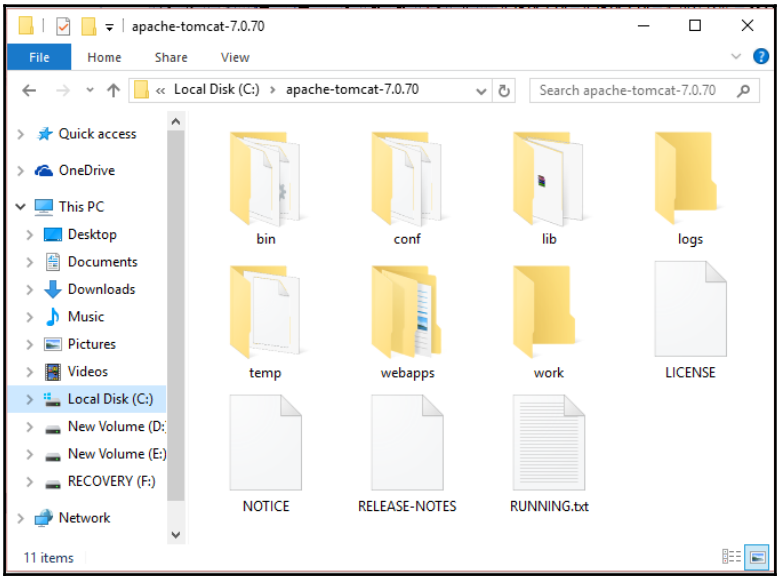
7.0.70

Please see the README file for packaging information. It explains what every distribution contains.

Binary Distributions

- Core:
  - zip (pgp, md5, sha1)
  - tar.gz (pgp, md5, sha1)
  - 32-bit Windows zip (pgp, md5, sha1)
  - 64-bit Windows zip (pgp, md5, sha1)
  - 32-bit/64-bit Windows Service Installer (pgp, md5, sha1)
- Full documentation:
  - tar.gz (pgp, md5, sha1)
- Deployer:
  - zip (pgp, md5, sha1)
  - tar.gz (pgp, md5, sha1)
- Extras:
  - JMX Remote jar (pgp, md5, sha1)
  - Web services jar (pgp, md5, sha1)
  - JULI adapters jar (pgp, md5, sha1)
  - JULI log4j jar (pgp, md5, sha1)
- Embedded:
  - tar.gz (pgp, md5, sha1)
  - zip (pgp, md5, sha1)

2. Extract the tomcat installation files:



3. Open command prompt and go to the bin directory to start the Tomcat.

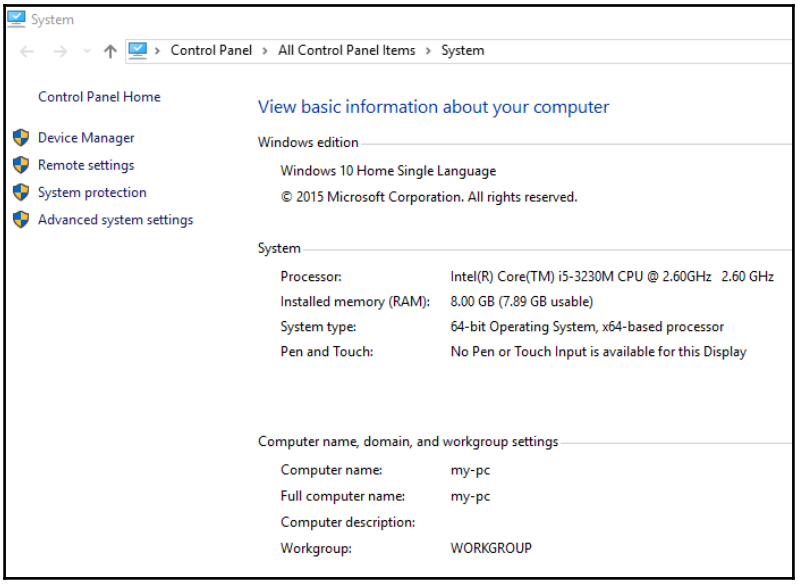
```
C:\>cd apache-tomcat-7.0.70\bin
```

4. Run startup.bat file in the command prompt.

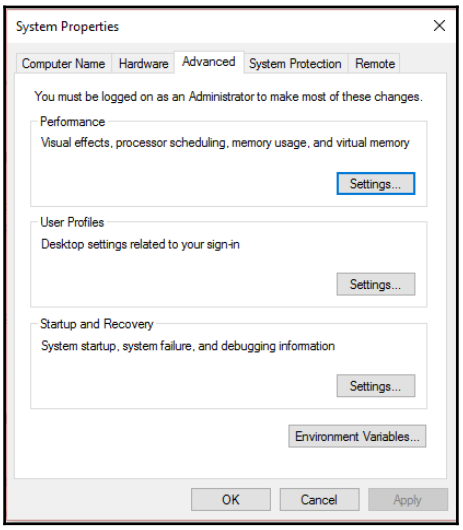
```
C:\apache-tomcat-7.0.70\bin>startup.bat
```

Neither the JAVA_HOME nor the JRE_HOME environment variable is defined. At least one of these environment variable is needed to run this program

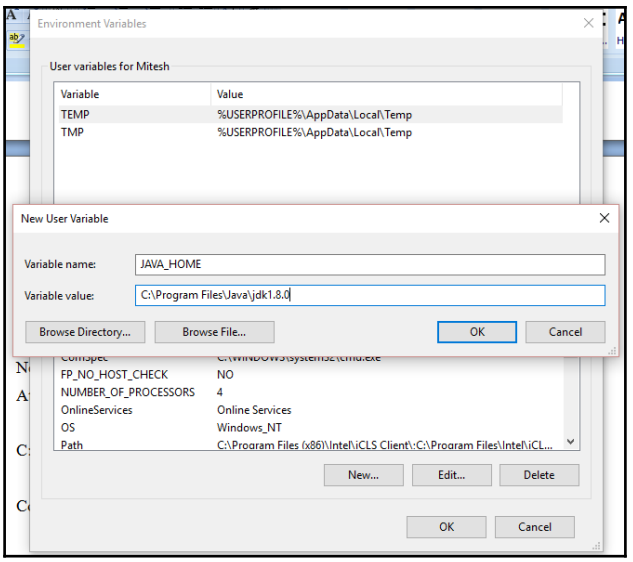
5. Oops! We need to set environment variables. Go to **Control Panel | All Control Panel Items | System**
6. Click on **Advanced system settings**:



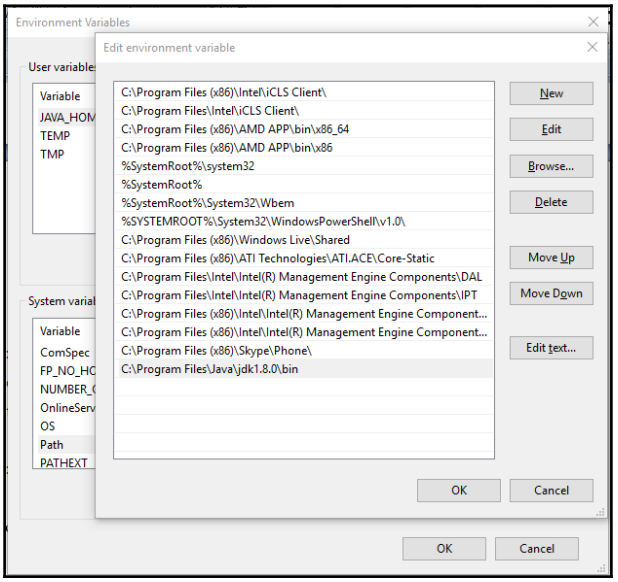
7. Click on the **Environment Variables...** to set JAVA_ HOME:



8. Click on **New...** and create a new variable for JAVA_HOME with value C:\Program Files\Java\jdk1.8.0 and click **OK**:



9. Click **OK** once again:



10. Open new command prompt and verify the Java Version:

```
C:\>java -version
java version "1.8.0-ea"
Java(TM) SE Runtime Environment (build 1.8.0-ea-b115)
Java HotSpot(TM) 64-Bit Server VM (build 25.0-b57, mixed mode)
```

11. Now go to tomcat\bin directory and execute startup.bat file:

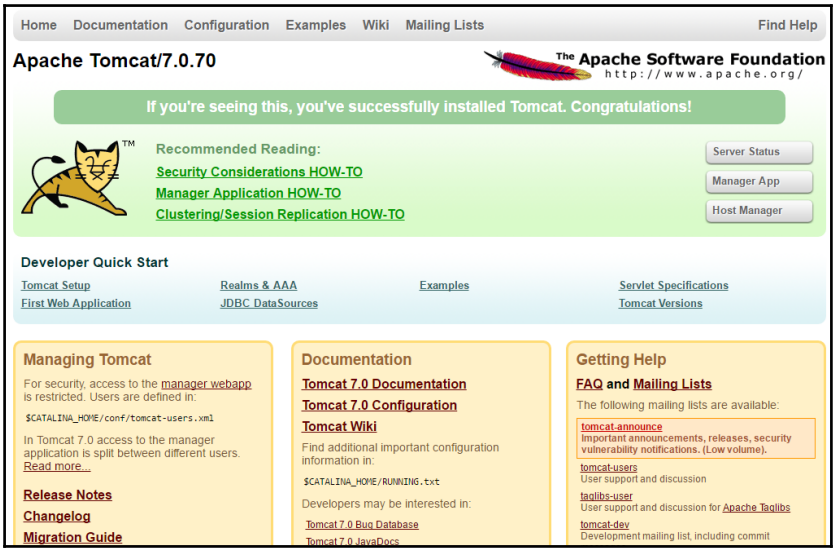
```
C:\apache-tomcat-7.0.70\bin>startup.bat
Using CATALINA_BASE:   "C:\apache-tomcat-7.0.70"
Using CATALINA_HOME:   "C:\apache-tomcat-7.0.70"
Using CATALINA_TMPDIR: "C:\apache-tomcat-7.0.70\temp"
Using JRE_HOME:        "C:\Program Files\Java\jdk1.8.0"
Using CLASSPATH:       "C:\apache-tomcat-7.0.70\bin\bootstrap.jar;C:\apache-
                        tomcat-7.0.70\bin\tomcat-juli.jar"
C:\apache-tomcat-7.0.70\bin>
```

12. Now our Tomcat is running. It may have similar type of output as given below. Verify Server startup message:

```
INFO: Starting Servlet Engine: Apache Tomcat/7.0.70
Jul 06, 2016 9:29:07 PM
org.apache.catalina.startup.HostConfigdeployDirectory
INFO: Deploying web application directory
C:\apache-tomcat-7.0.70\webapps\docs
Jul 06, 2016 9:29:08 PM
org.apache.catalina.util.SessionIdGeneratorBasecreateSecureRandom
INFO: Creation of SecureRandom instance for session ID generation using
[SHA1PRNG] took [331] milliseconds.
Jul 06, 2016 9:29:09 PM
org.apache.catalina.startup.HostConfigdeployDirectory
INFO: Deployment of web application directory
C:\apache-tomcat-7.0.70\webapps\docs has finished in 1,887 ms
Jul 06, 2016 9:29:09 PM
org.apache.catalina.startup.HostConfigdeployDirectory
INFO: Deploying web application directory
C:\apache-tomcat-7.0.70\webapps\examples
Jul 06, 2016 9:29:11 PM
org.apache.catalina.startup.HostConfigdeployDirectory
INFO: Deployment of web application directory
C:\apache-tomcat-7.0.70\webapps\examples has finished in 2,474 ms
Jul 06, 2016 9:29:11 PM
org.apache.catalina.startup.HostConfigdeployDirectory
INFO: Deploying web application directory
C:\apache-tomcat-7.0.70\webapps\host-manager
Jul 06, 2016 9:29:11 PM
org.apache.catalina.startup.HostConfigdeployDirectory
INFO: Deployment of web application directory
C:\apache-tomcat-7.0.70\webapps\host-manager has finished in 140 ms
Jul 06, 2016 9:29:11 PM
org.apache.catalina.startup.HostConfigdeployDirectory
INFO: Deploying web application directory
C:\apache-tomcat-7.0.70\webapps\manager
Jul 06, 2016 9:29:11 PM
org.apache.catalina.startup.HostConfigdeployDirectory
INFO: Deployment of web application directory
C:\apache-tomcat-7.0.70\webapps\manager has finished in 160 ms
Jul 06, 2016 9:29:11 PM
org.apache.catalina.startup.HostConfigdeployDirectory
INFO: Deploying web application directory
C:\apache-tomcat-7.0.70\webapps\ROOT
```

```
Jul 06, 2016 9:29:11 PM
org.apache.catalina.startup.HostConfigdeployDirectory
INFO: Deployment of web application directory
C:\apache-tomcat-7.0.70\webapps\ROOT has finished in 79 ms
Jul 06, 2016 9:29:11 PM org.apache.coyote.AbstractProtocol start
INFO: Starting ProtocolHandler ["http-apr-8080"]
Jul 06, 2016 9:29:11 PM org.apache.coyote.AbstractProtocol start
INFO: Starting ProtocolHandler ["ajp-apr-8009"]
Jul 06, 2016 9:29:11 PM org.apache.catalina.startup.Catalina start
INFO: Server startup in 5172 ms
```

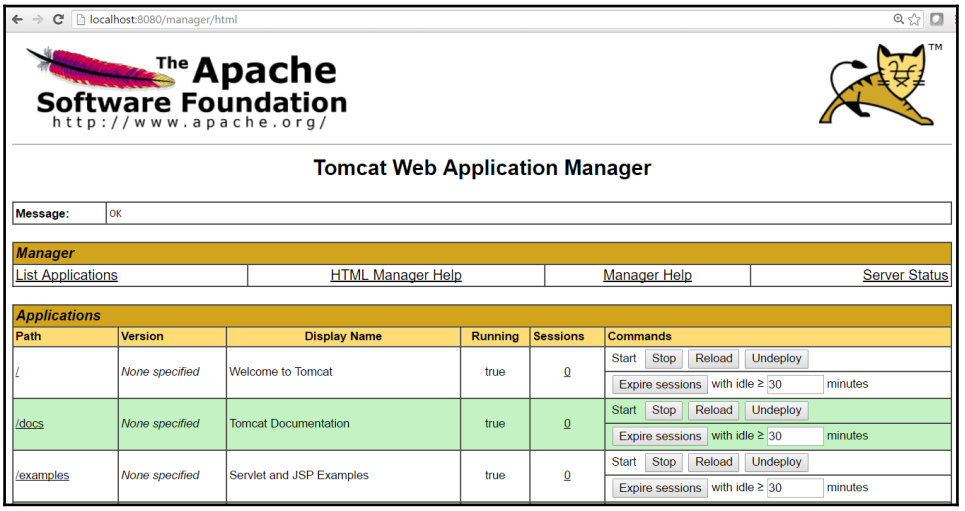
13. Use IP address and port number combination to navigate to tomcat Home page.



14. Go to Tomcat Installation Directory -> conf -> tomcat-users.xml and uncomment Role and User related line or rewrite. Give manager-gui as a **rolename** for testing purpose. We need **manager-script** for deployment via deploy plugin:

```
<?xml version='1.0' encoding='utf-8'?>
<!--
<tomcat-users>
<!--
<!--
  NOTE:  The sample user and role entries below are intended for use with the
  examples web application. They are wrapped in a comment and thus are ignored
  when reading this file. If you wish to configure these users for use with the
  examples web application, do not forget to remove the <!.. ..> that surrounds
  them. You will also need to set the passwords to something appropriate.
-->
  <role rolename="manager-script"/>
  <user username="admin" password="admin@123" roles="manager-script"/>
</tomcat-users>
```

15. Click on the Manager App link on the Tomcat home page and give user name and password given in the tomcat-users.xml. Now we can access Manager App:

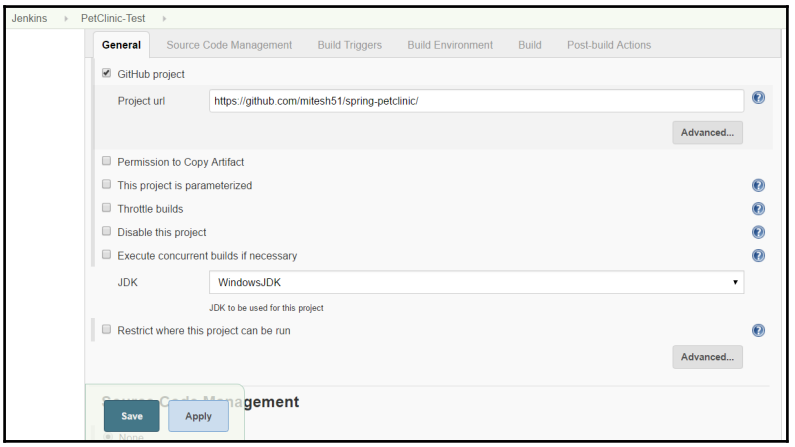


16. For Jenkins Deploy plgin, change the rolename to manager-script.
17. Restart the tomcat and visit http://<IP Address>:8080/manager/text/list

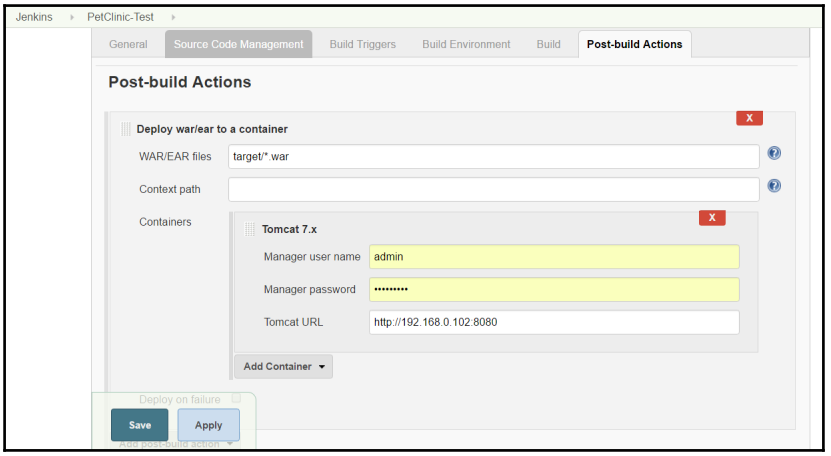
OK - Listed applications for virtual host localhost
/:running:0:ROOT
/petclinic:running:1:petclinic
/examples:running:0:examples
/host-manager:running:0:host-manager
/manager:running:0:manager

/docs:running:0:docs

18. Go to Jenkins Build Job and click on **Configure**. Select the proper JDK configuration for Jenkins agent:



19. In the **Post-build Action**, select **Deploy war/ear to a container**. Provide location of the war file in the Jenkins workspace, Tomcat manager credentials, and Tomcat URL with port:

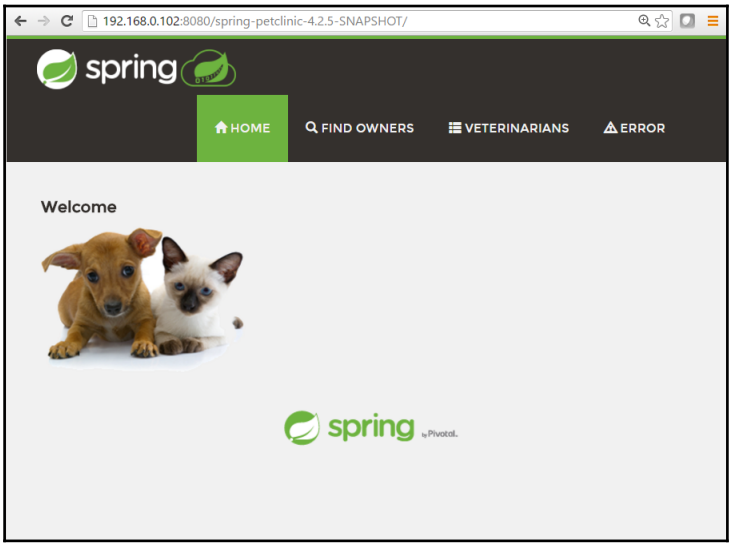


20. Click on **Apply** and **Save**. Click on **Build now** on Jenkins Build specific page.

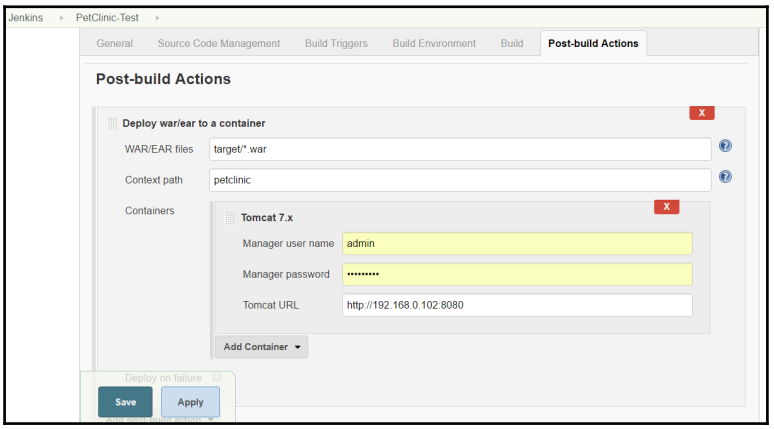
Verify the console output for fresh deployment:

```
Results :

Tests run: 59, Failures: 0, Errors: 0, Skipped: 0

[INFO]
[INFO] --- maven-war-plugin:2.3:war (default-war) @ spring-petclinic ---
[INFO] Packaging webapp
[INFO] Assembling webapp [spring-petclinic] in [d:\jenkins\workspace\PetClinic-Test\target\spring-
petclinic-4.2.5-SNAPSHOT]
[INFO] Processing war project
[INFO] Copying webapp resources [d:\jenkins\workspace\PetClinic-Test\src\main\webapp]
[INFO] Webapp assembled in [1669 msecs]
[INFO] Building war: d:\jenkins\workspace\PetClinic-Test\target\spring-petclinic-4.2.5-SNAPSHOT.war
[INFO] ------------------------------------------------------------------------
[INFO] BUILD SUCCESS
[INFO] ------------------------------------------------------------------------
[INFO] Total time: 28.772 s
[INFO] Finished at: 2016-07-06T22:59:37+05:30
[INFO] Final Memory: 29M/261M
[INFO] ------------------------------------------------------------------------
Deploying d:\jenkins\workspace\PetClinic-Test\target\spring-petclinic-4.2.5-SNAPSHOT.war to container
Tomcat 7.x Remote
  [d:\jenkins\workspace\PetClinic-Test\target\spring-petclinic-4.2.5-SNAPSHOT.war] is not deployed.
Doing a fresh deployment.
  Deploying [d:\jenkins\workspace\PetClinic-Test\target\spring-petclinic-4.2.5-SNAPSHOT.war]
Finished: SUCCESS
```

21. Once build is successful, visit the URL in browser and notice the context. It is similar to name of the Application:



22. In the **Post-build Actions**, give **Context path** and **Save**. Click on build now again:



23. Verify the Application URL by giving new context path.

For deployments, where we can access tomcat-users.xml file in case where we use Tomcat as application container, we will use the same method for deployment. If we don't have direct access to tomcat directory or can't change tomcat-users.xml in such case, another approach can be to ssh the remote host and copy the file into remote host's webapps file of tomcat directory. All ssh commands can be used directly from the build job.

Deploying Application in Docker Container

We have already covered how to use Tomcat with Docker container in Chapter 5, *Installing and Configuring Docker*. To deploy an application with Deploy plugin of Jenkins, we will change tomcat-users.xml. Let's take a step-by-step tour:

1. Change rolename to manager-script in tomcat-users.xml:

```
<?xml version='1.0' encoding='utf-8'?>
<tomcat-users xmlns=”http://tomcat.apache.org/xml”
xmlns:xsi=”http://www.w3.org/2001/XMLSchema-instance”
xsi:schemaLocation=”http://tomcat.apache.org/xml tomcat-users.xsd”
version=”1.0”>
<!–
```

NOTE: The sample user and role entries below are intended for use with the examples web application. They are wrapped in a comment and thus are ignored when reading this file. If you wish to configure these users for use with the examples web application, do not forget to remove the <!....> that surrounds them. You will also need to set the passwords to something appropriate.
–>
<role rolename="manager-script"/>
<user username="admin" password="admin@123" roles="manager-script"/>
</tomcat-users>

2. In Dockerfile, we will copy tomcat-users.xml to /usr/local/tomcat/conf/ directory:

FROM tomcat:8.0
MAINTAINER Mitesh<mitesh.soni@outlook.com>
COPY tomcat-users.xml /usr/local/tomcat/conf/tomcat-users.xml

3. Execute docker build command to create an image:

[root@localhostmitesh]#docker build -t devops_tomcat_sc .
Sending build context to Docker daemon 8.192 kB
Sending build context to Docker daemon
Step 0 : FROM tomcat:8.0
—> 5d4577339b14
Step 1 : MAINTAINER Mitesh<mitesh.soni@outlook.com>
—> Using cache
—> c63f90db4c14
Step 2 : COPY tomcat-users.xml /usr/local/tomcat/conf/tomcat-users.xml
—> aebbcf634f64
Removing intermediate container 7a528d1c8e3b
Successfully built aebbcf634f64
You have new mail in /var/spool/mail/root

4. Verify the newly created image by using docker images command in terminal:

[root@localhostmitesh]#docker images
REPOSITORY TAG IMAGE ID CREATED VIRTUAL SIZE
devops_tomcat_sc latest aebbcf634f64 2 minutes ago 359.2 MB
devopstomcatnew latest eb50c4ceefb5 5 days ago 359.2 MB
devopstomcat8 latest f3537165ebe7 5 days ago 344.6 MB

```
tomcat6 latest 400f097677e9 2 weeks ago 658.4 MB
devopstomcat latest 400f097677e9 2 weeks ago 658.4 MB
centos latest 2a332da70fd1 5 weeks ago 196.7 MB
ubuntu latest 686477c12982 9 weeks ago 120.7 MB
hello-world latest f1d956dc5945 10 weeks ago 967 B
You have new mail in /var/spool/mail/root
```

5. Execute docker run command to create a container:

```
[root@localhostmitesh]#docker run -p 8180:8080 -d –name
devopstomcatscdevops_tomcat_sc
771bb7cb809dabe9323d65579e98077eaec146db4fc38d2ace1d75577144002d
You have new mail in /var/spool/mail/root
```

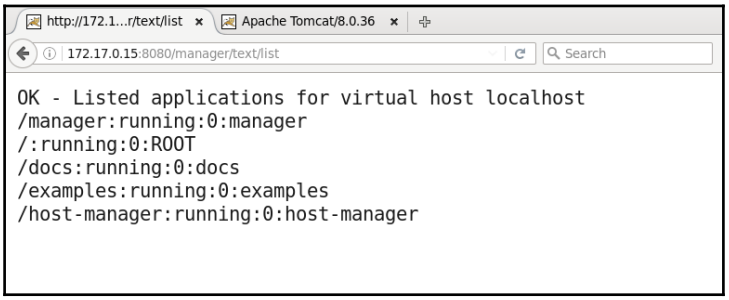
6. Verified the new container with dockerps command:

```
[root@localhostmitesh]#dockerps
CONTAINER ID IMAGE COMMAND CREATED STATUS PORTS
NAMES
771bb7cb809ddevops_tomcat_sc “catalina.sh run” 7 seconds ago Up 6
seconds 0.0.0.0:8180->8080/tcpdevopstomcatsc
```

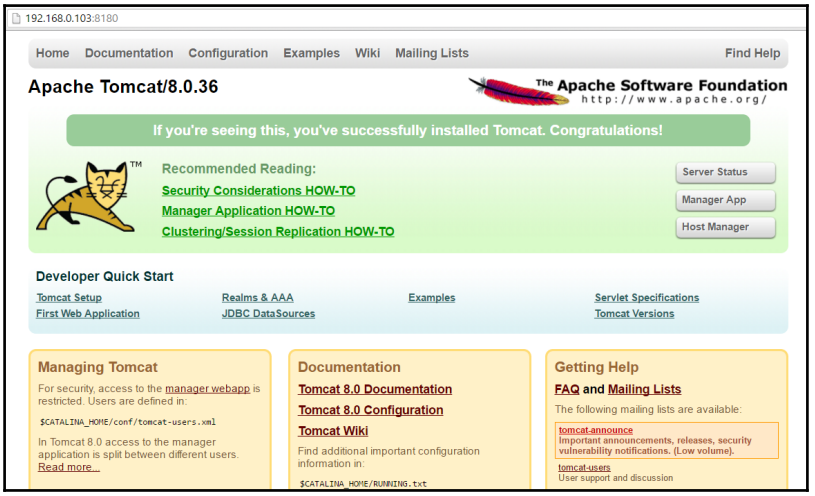
7. Use docker inspect 771bb7cb809d (container id) to get an IP address.
8. Stop IP tables for verification or open ports in IP tables:

```
[root@localhostmitesh]# service iptables stop
iptables: Setting chains to policy ACCEPT: nat filter [ OK ]
iptables: Flushing firewall rules: [ OK ]
iptables: Unloading modules: [ OK ]
You have new mail in /var/spool/mail/root
```

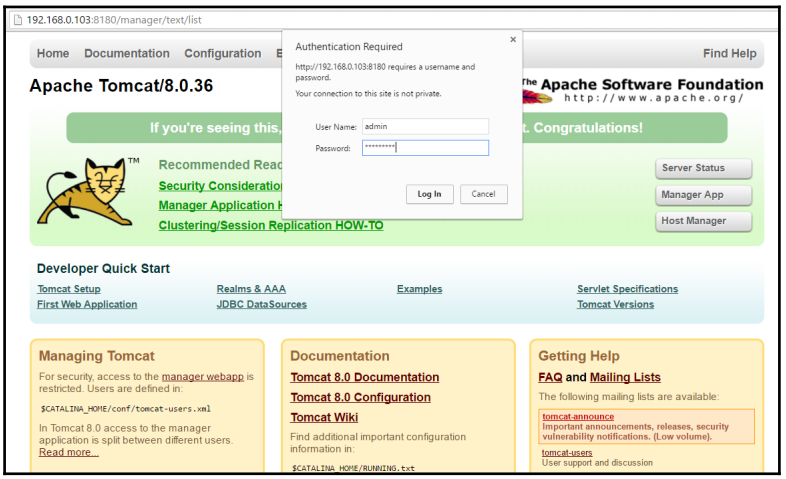
9. Use the IP address and access the manager app URL. Verify whether it is successful or not:



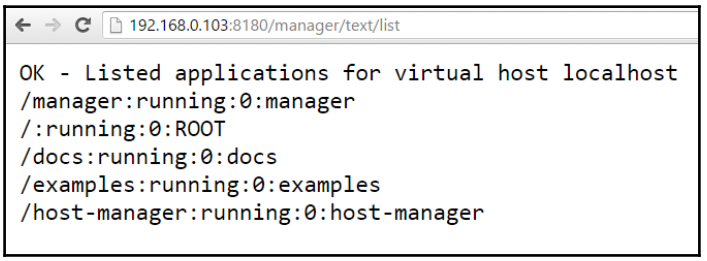
10. As we have mapped port, use host's IP address and verify Tomcat installation:



11. Use the IP address of the host and access the manager app URL. Provide **Username** and **Password**.



12. Verify whether it is successful or not:

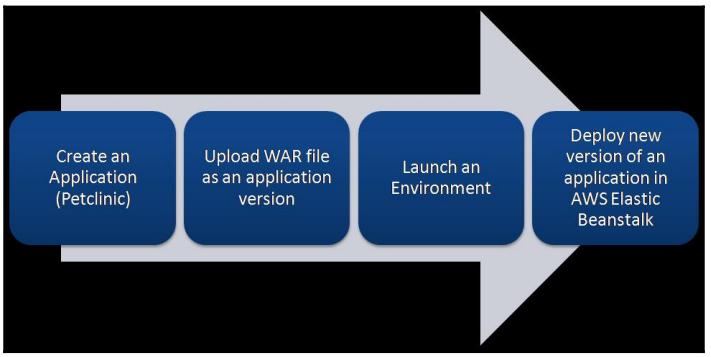


13. Once everything is working fine, use deploy plugin to deploy an application into Docker container.

Deploying Application in AWS

AWS Elastic Beanstalk is a **Platform as a Service (PaaS)** offering from Amazon. We will use AWS Elastic Beanstalk to deploy Petclinic application on the AWS Platform. The good part is we need not to manage infrastructure or even platform as it is a PaaS offering. We can configure scaling and other details.

Following are the steps to deploy an application in AWS Elastic Beanstalk:

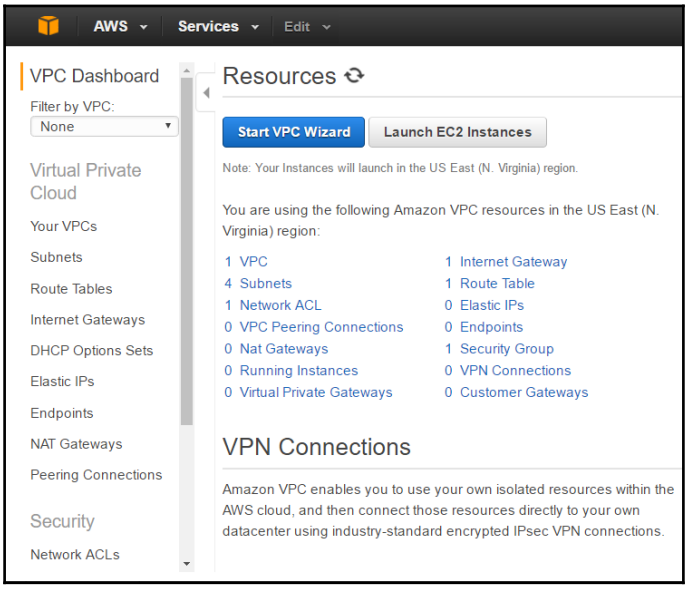


AWS Elastic Beanstalk supports following Programming languages and platforms:

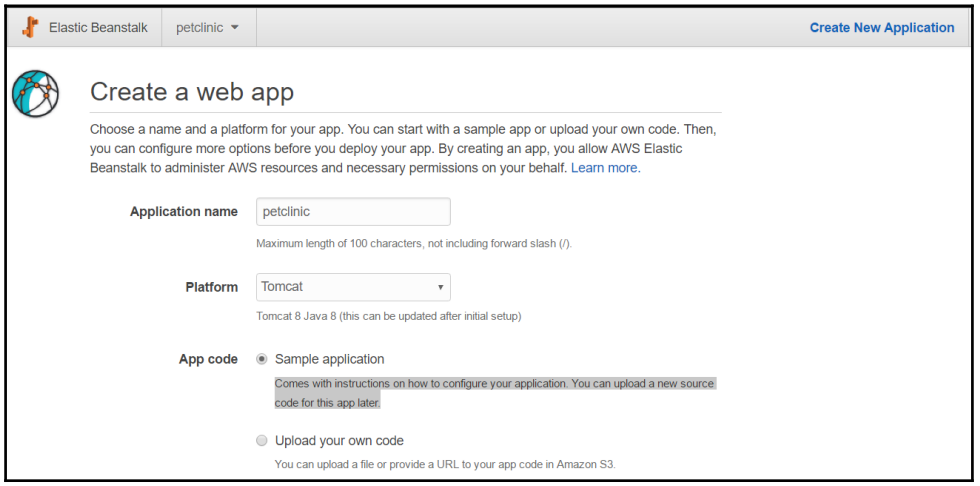
Programming Languages	Platforms
• Java, PHP, Python, Ruby, Go	• Go, Java SE, Java with Tomcat, .NET on Windows Server with IIS, Node.js, PHP, Python, Ruby

Let's create a sample application to understand how AWS Elastic Beanstalk works and then use Jenkins plugin to deploy an application:

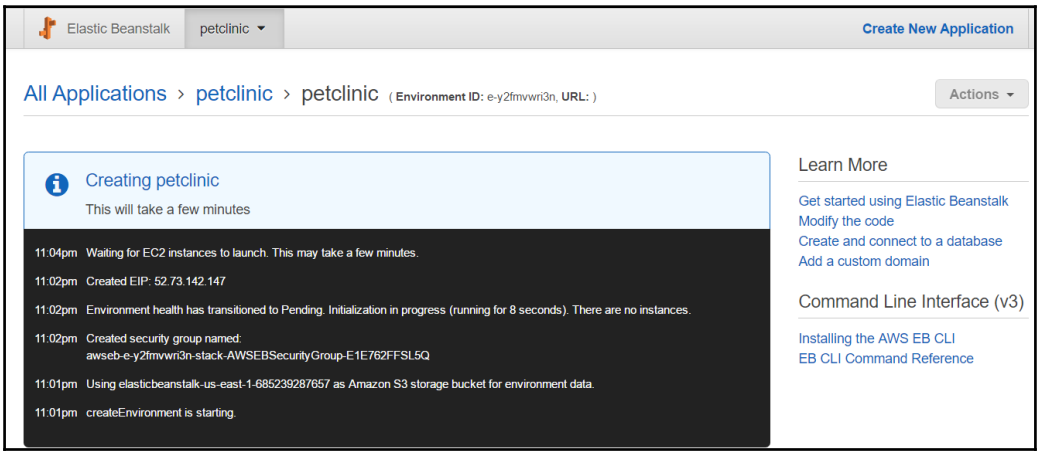
1. Go to AWS Management console and verify whether we have a default VPC or not. If by mistake you have deleted default VPC and subnet, then send request to AWS Customer support to recreate it:



2. Click on the **Services** in AWS Management Console and select **AWS Elastic Beanstalk**. Create a new application named petclinic. Select Tomcat as a platform Sample Application:

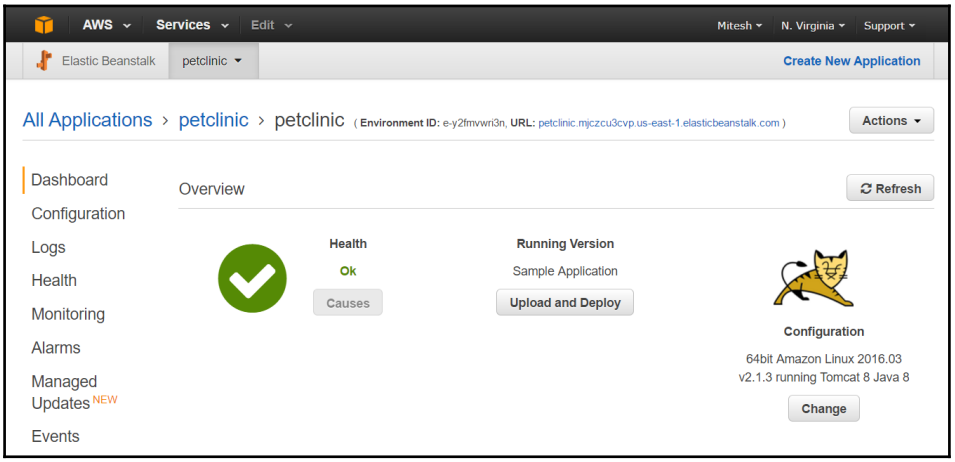


3. Verify the sequence of events for the creation of sample application:

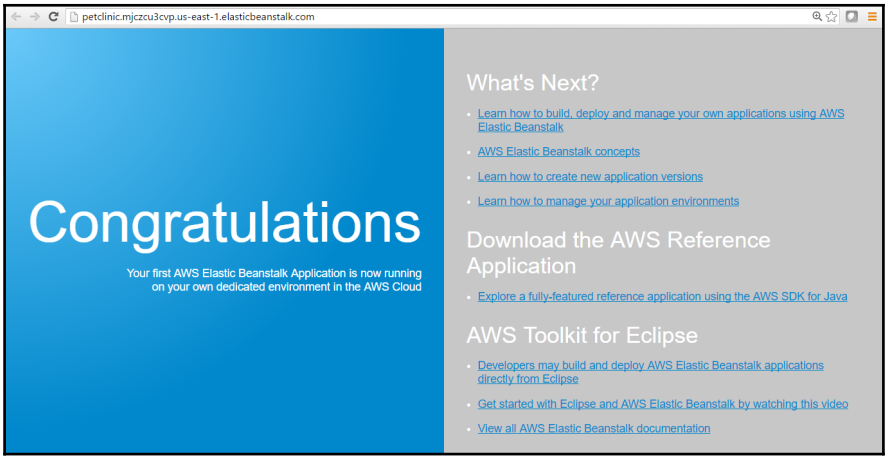


4. It will take some time and once Environment is created it will be in green color as shown below:

5. Click on the petclinic environment and verify the **Health** and **Running Version** on the Dashboard:



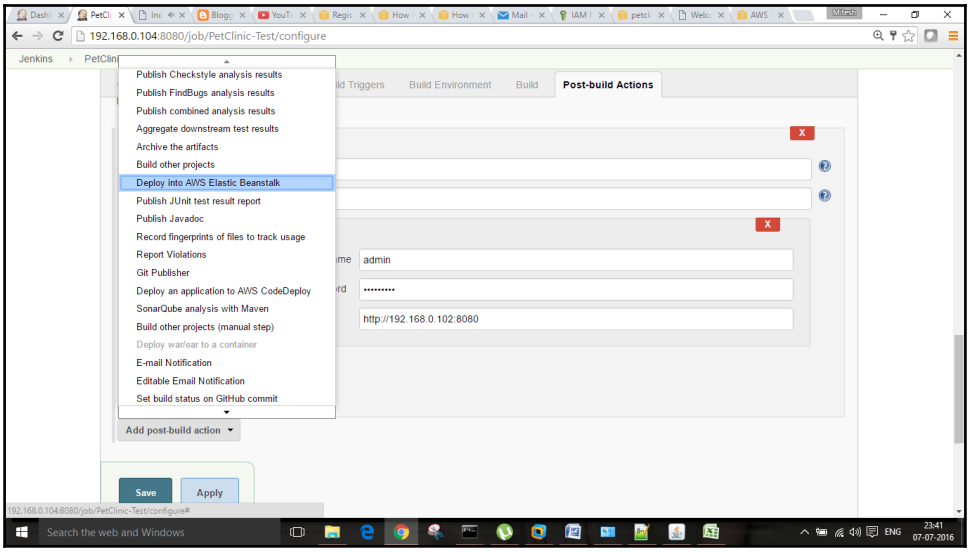
6. Verify the Environment ID and URL. Click on the URL and verify the default page:



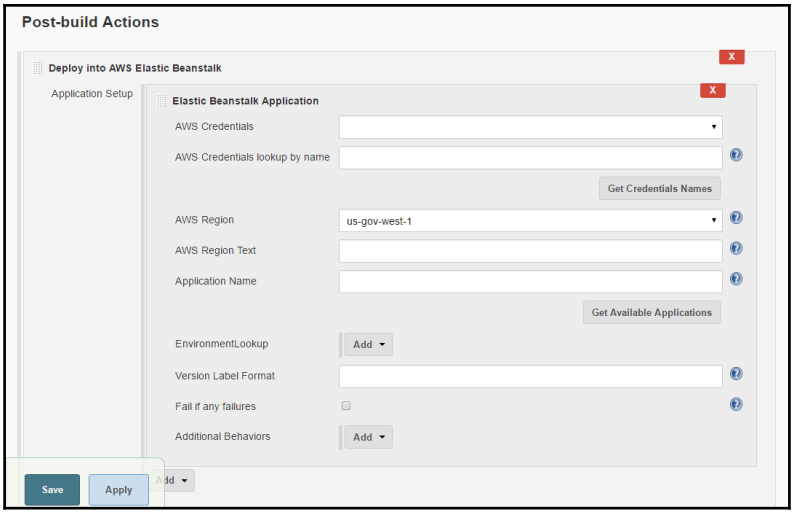
7. Install Amazon Web Services Elastic Beanstalk Publisher.

For more details, visit https://wiki.jenkins-ci.org/display/JENKINS/AWS+Beanstalk+Publisher+Plugin.

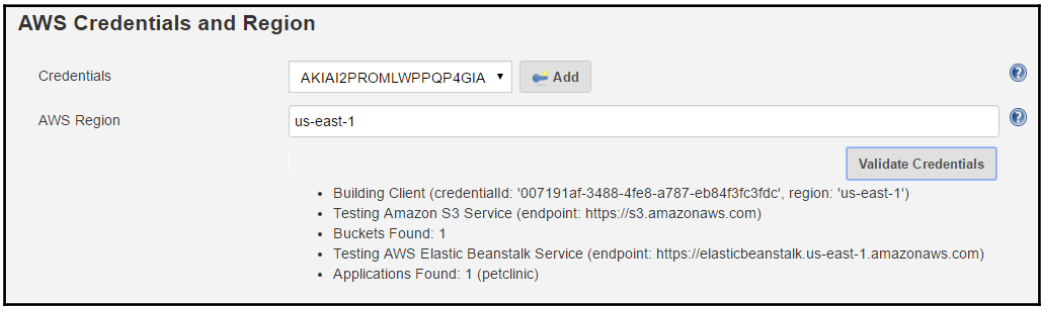
8. Open the Jenkins Dashboard and go to the **Build job**. Click on the **Post build action** and select **Deploy into AWS Elastic Beanstalk**.



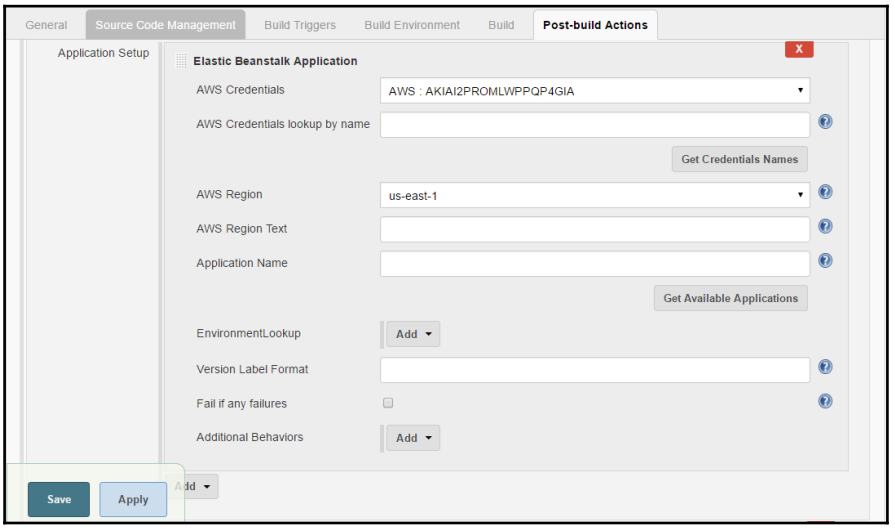
9. New section comes up in the **Post-build Actions** for AWS Elastic Beanstalk:



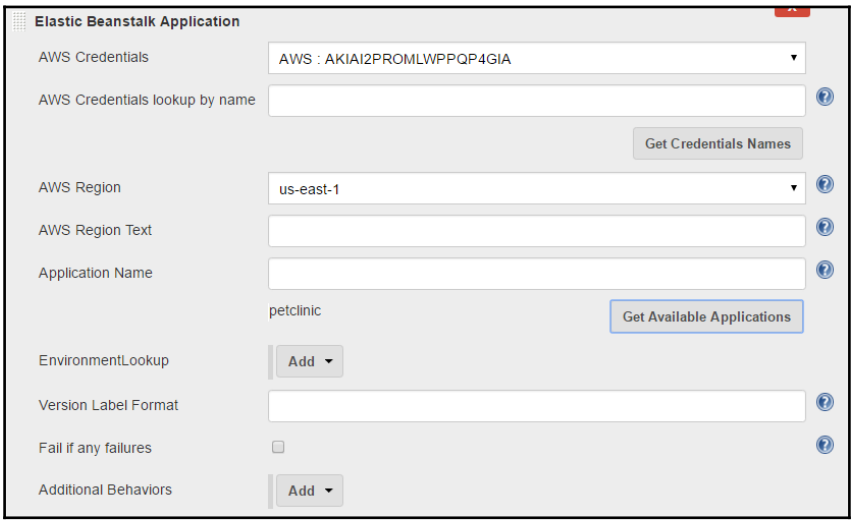
10. Click on Jenkins Dashboard and select **Credentials** and Add AWS credentials:



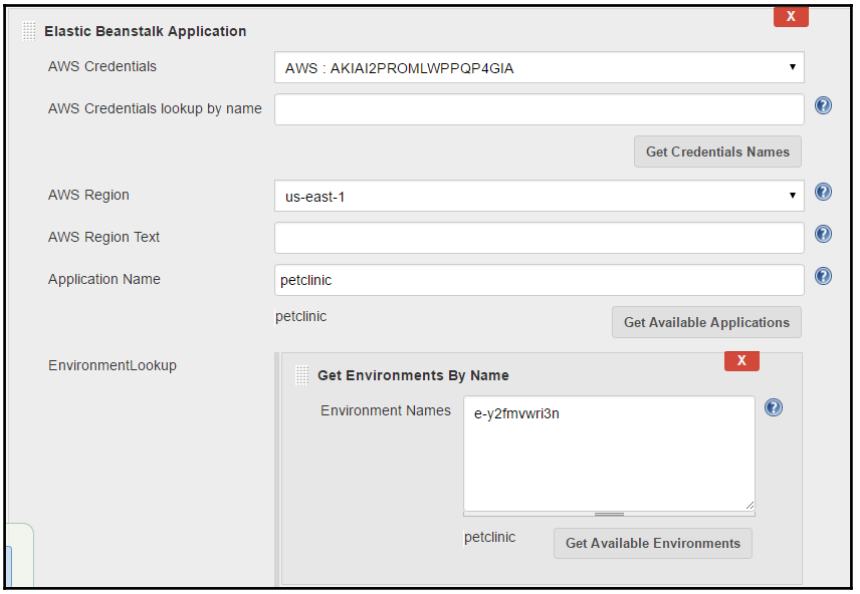
11. Go to Jenkins build and select an **AWS credentials** which is set in the global configuration:



12. Select **AWS Region** from the list and click on the **Get Available Applications**. As we have created a sample application, it will show up:



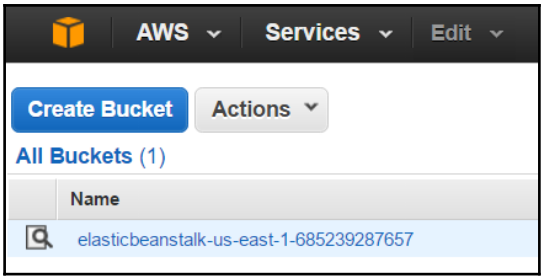
13. In the **EnvironmentLookup**, provide Environment ID in the Get Environments By Name and click on the **Get Available Environments**.



14. Save the configuration and click on Build now.

Let's verify the AWS Management Console:

1. Go to S3 services and verify the available buckets:



As WAR file is having large size, it will take some time to upload on the Amazon S3. Once it is uploaded, it will be available in the Amazon S3 bucket.

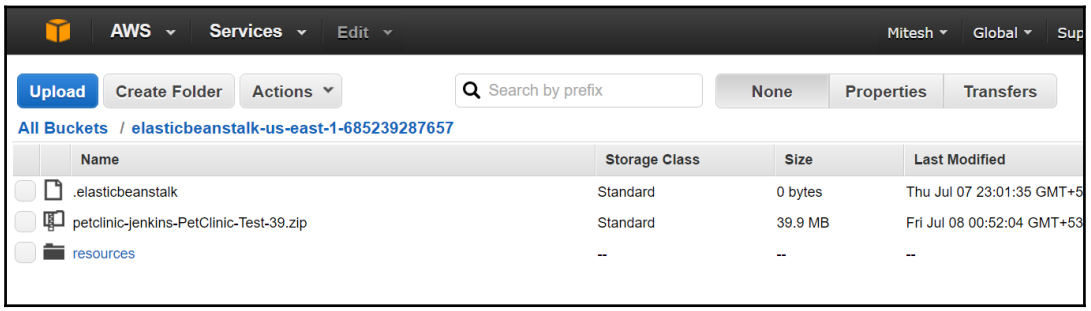
2. Verify the Build job execution status in Jenkins. Some section of output is given below with explanation.

Test case execution and WAR file creation is successful:

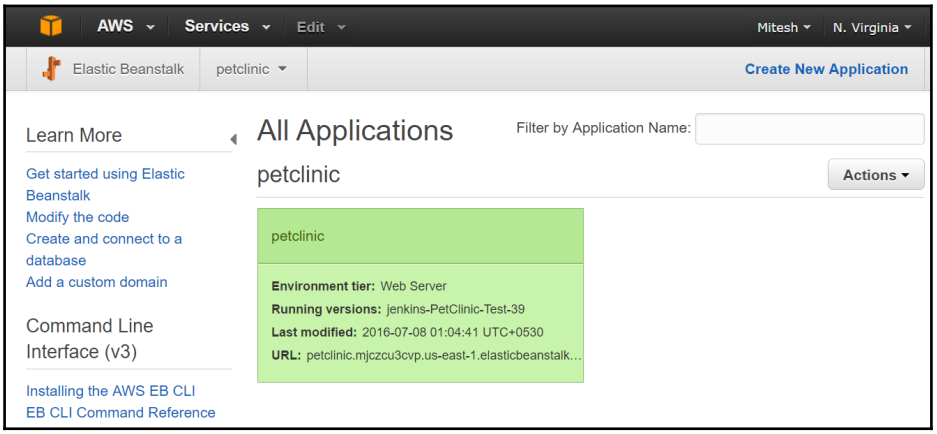
```
Tests run: 59, Failures: 0, Errors: 0, Skipped: 0

[INFO]
[INFO] — maven-war-plugin:2.3:war (default-war) @ spring-petclinic —
[INFO] Packaging webapp
[INFO] Assembling webapp [spring-petclinic] in
[d:\jenkins\workspace\PetClinic-Test\target\spring-petclinic-4.2.5-
SNAPSHOT]
[INFO] Processing war project
[INFO] Copying webapp resources [d:\jenkins\workspace\PetClinic-
Test\src\main\webapp]
[INFO] Webapp assembled in [1539 msecs]
[INFO] Building war: d:\jenkins\workspace\PetClinic-
Test\target\spring-petclinic-4.2.5-SNAPSHOT.war
[INFO] ————————————————————————————————
[INFO] BUILD SUCCESS
[INFO] ————————————————————————————————
[INFO] Total time: 30.469 s
```

```
[INFO] Finished at: 2016-07-08T00:51:52+05:30
[INFO] Final Memory: 29M/258M
[INFO] ————————————————————————————————
Execution of AWSEB Deployment Plugin / Post build action is started:
AWSEB Deployment Plugin Version 0.3.10
Root File Object is a file. We assume its a zip file, which is okay.
bucketName not set. Calling createStorageLocation
Using s3 Bucket 'elasticbeanstalk-us-east-1-685239287657'
Uploading file awseb-5081374840514488317.zip as s3://elasticbeanstalk-us-
east-1-685239287657/petclinic-jenkins-PetClinic-Test-39.zip
Deployment activity with new Version Label will start:
Creating application version jenkins-PetClinic-Test-39 for application
petclinic for path s3://elasticbeanstalk-us-east-1-685239287657/petclinic-
jenkins-PetClinic-Test-39.zip
Created version: jenkins-PetClinic-Test-39
Using environmentId 'e-y2fmvwri3n'
No pending Environment Updates. Proceeding.
Checking health/status of environmentId e-y2fmvwri3n attempt 1/30
Environment Status is 'Ready'. Moving on.
Updating environmentId 'e-y2fmvwri3n' with Version Label set to
'jenkins-PetClinic-Test-39'
Environment status is updated and Health status is updated along with
Deployment status:
Fri Jul 08 01:03:10 IST 2016 [INFO] Environment update is starting.
Checking health/status of environmentId e-y2fmvwri3n attempt 1/30
Versions reported: (current=jenkins-PetClinic-Test-39, underDeployment:
jenkins-PetClinic-Test-39). Should I move on? false
Environment Status is 'Ready' and Health is 'Green'. Moving on.
Deployment marked as 'successful'. Starting post-deployment cleanup.
Cleaning up temporary file
C:\Users\Mitesh\AppData\Local\Temp\awseb-5081374840514488317.zi
p
Finished: SUCCESS
```

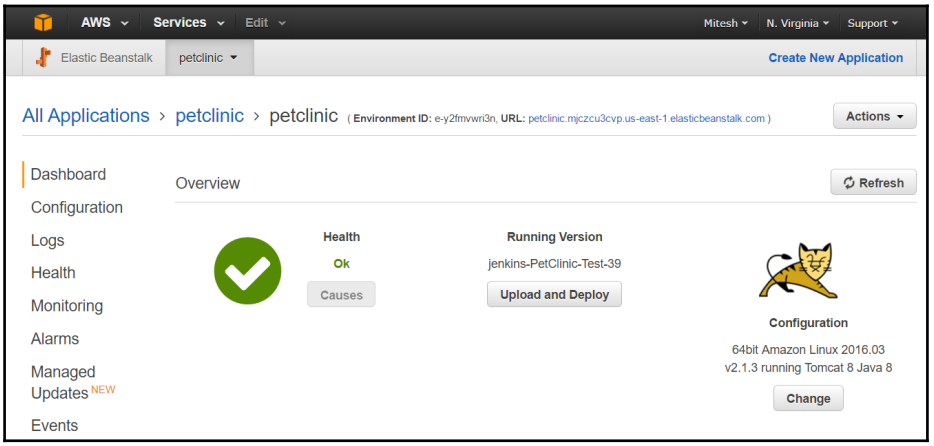
3. Build is successful and now verify the AWS Management console:



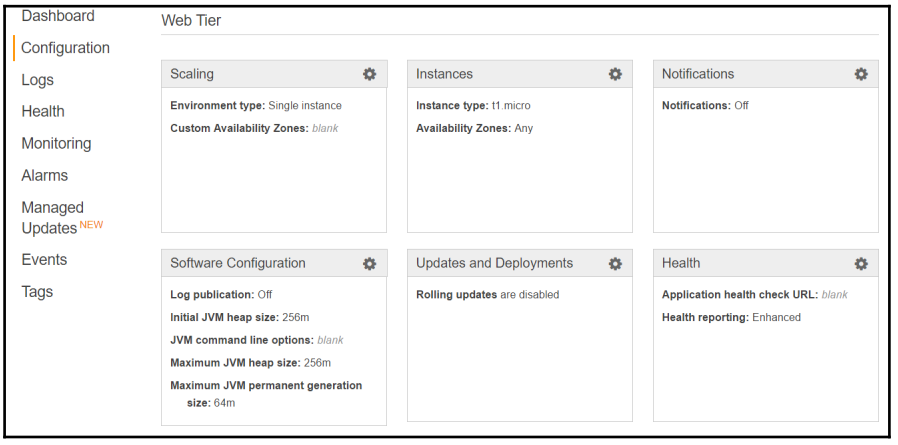
4. Go **to Services**, click on **AWS Elastic Beanstalk** and verify the Environment. Earlier Running versions was Sample Application, now the version is updated as given in Version Label Format in Jenkins build job configuration:



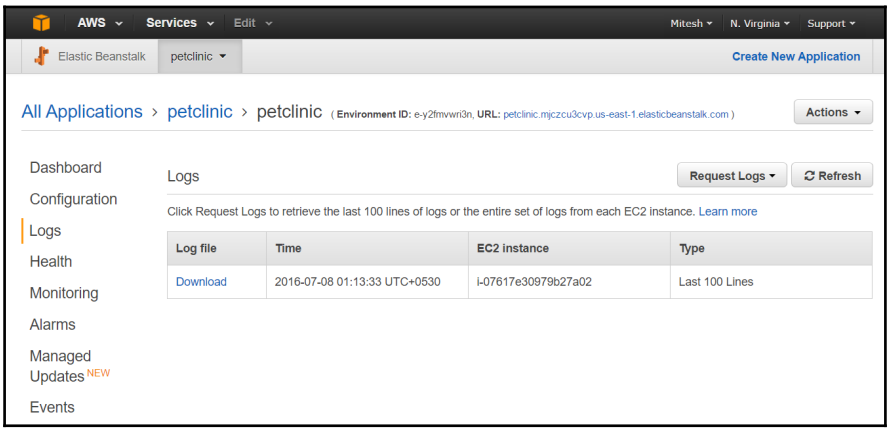
5. Go to Dashboard and verify **Health** and **Running Version** again:



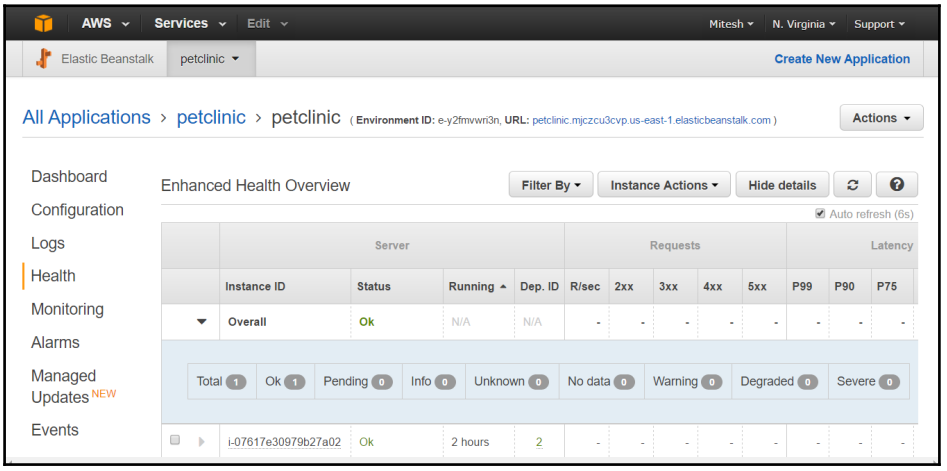
6. Click on the **Configuration** link on AWS Elastic Beanstalk Dashboard and verify **Scaling**, **Instances**, **Notifications**, **Software Configuration**, **Updates and Deployments**, **Health** and so on.



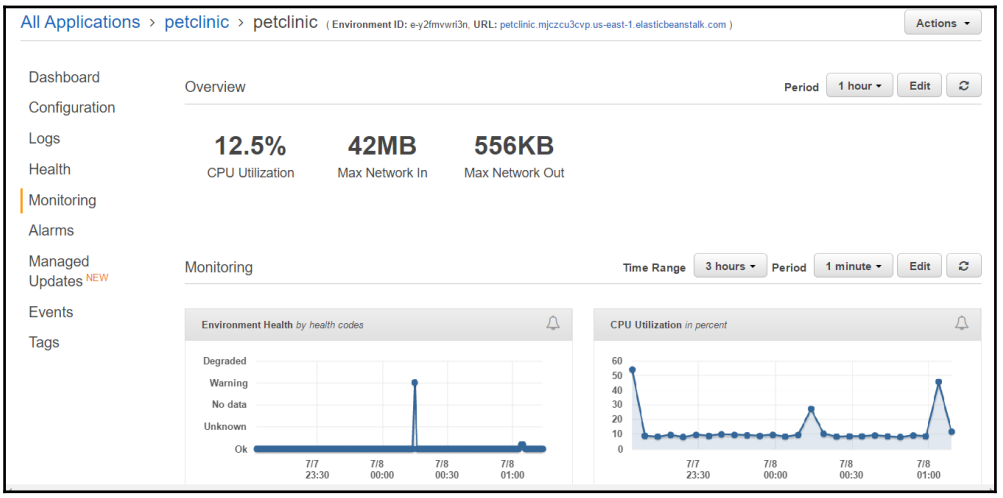
7. Click on **Logs** to download the log files for AWS Elastic Beanstalk application:



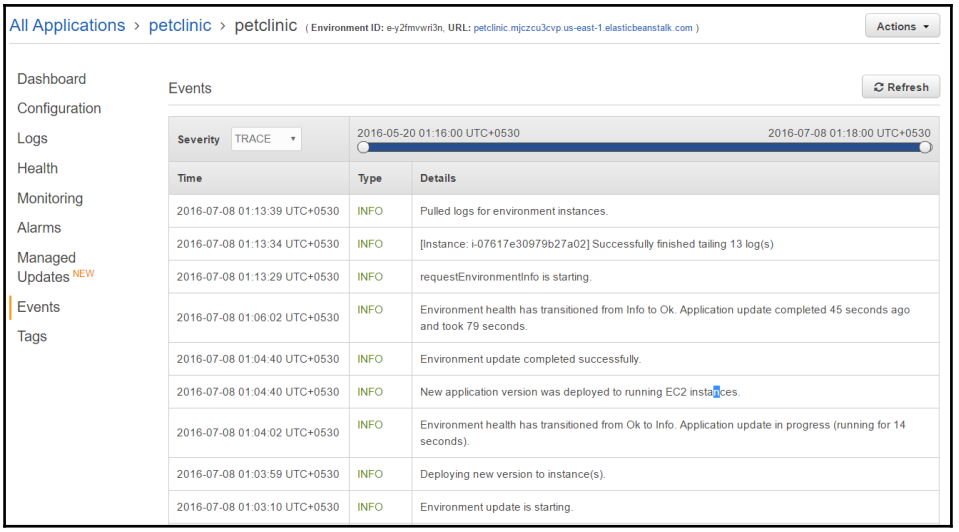
8. Verify the **Enhanced Health Overview** and check the status:



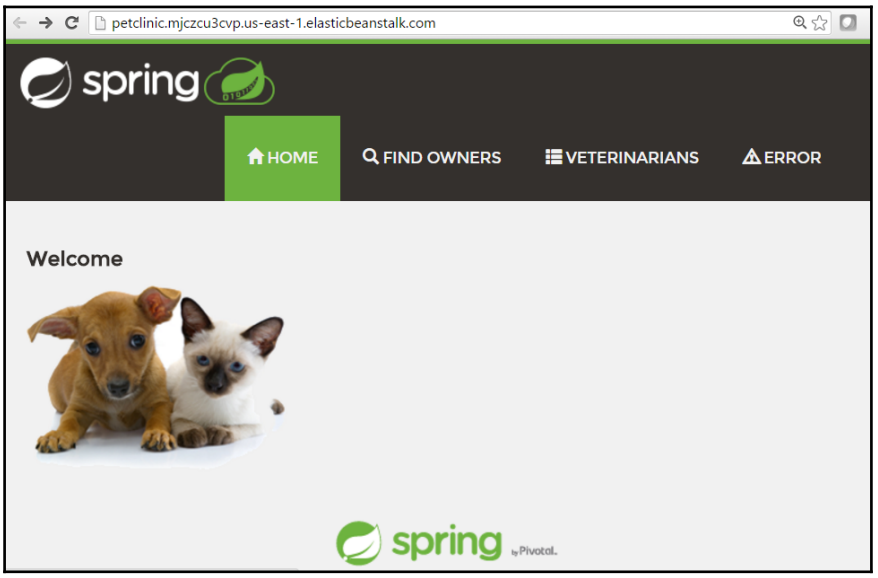
9. Click on the **Monitoring** for extensive monitoring details in form of CPU Utilization and Health of an application:



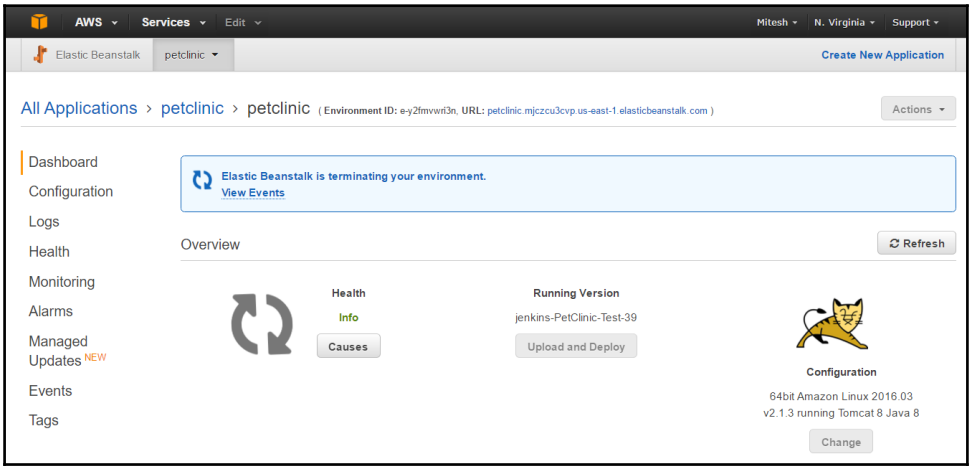
10. Click on **Events** to get list of all events of AWS Elastic Beanstalk application lifecycle:



11. Once, all is verified, click on the URL for the environment and our Petclinic Application is live:



12. Once application deployment is successful then terminate the environment:



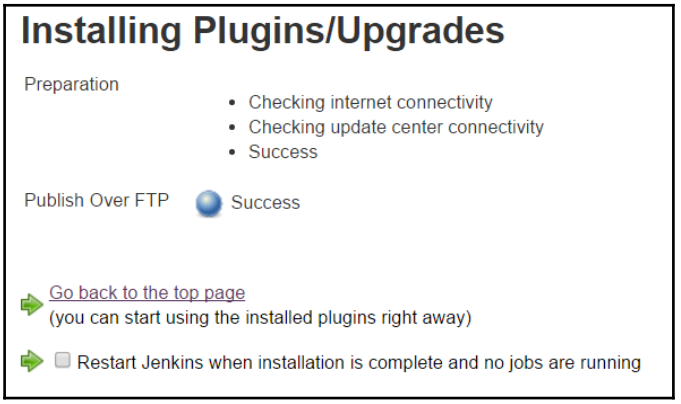
Thus, we have successful application deployment in AWS Elastic Beanstalk.

Deploying Application in Microsoft Azure

Microsoft Azure App Services is a Platform as a Service. In this section we will introduce Azure Web App and how we can deploy Petclinic application:

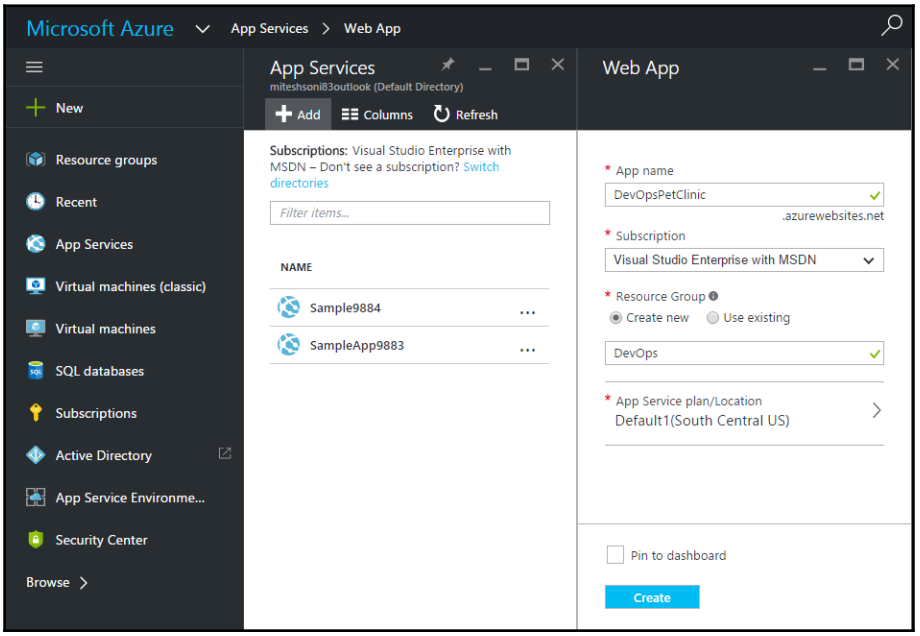
1. Let's install Publish Over FTP plugin in Jenkins. We will use Azure Web App's FTP details to publish Petclinic war file:

2. The plugin is installed successfully in the restart the Jenkins:

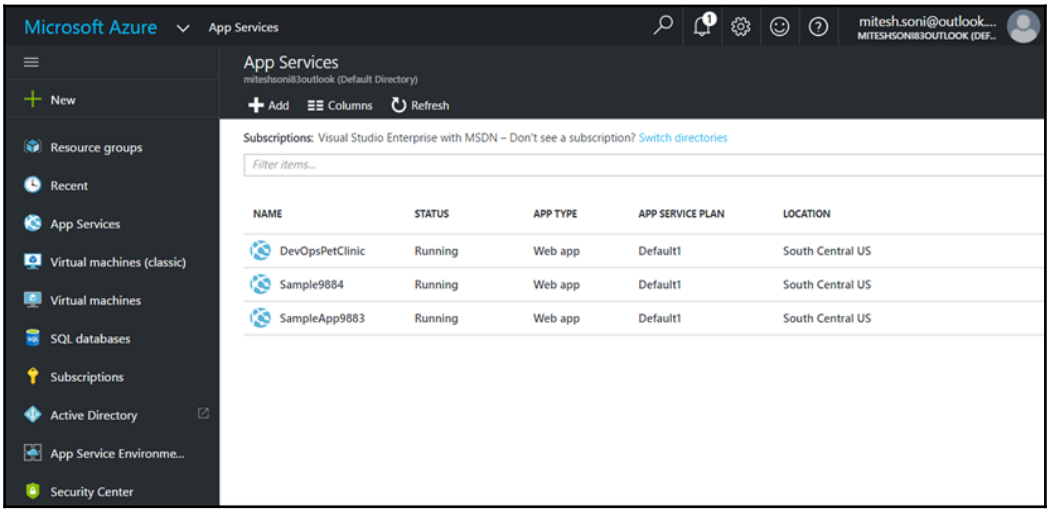


3. Go to Microsoft Azure Portal at https://portal.azure.com. Click on the **App Services** and click on the **Add**. Provide inputs for Name of Azure Web App,

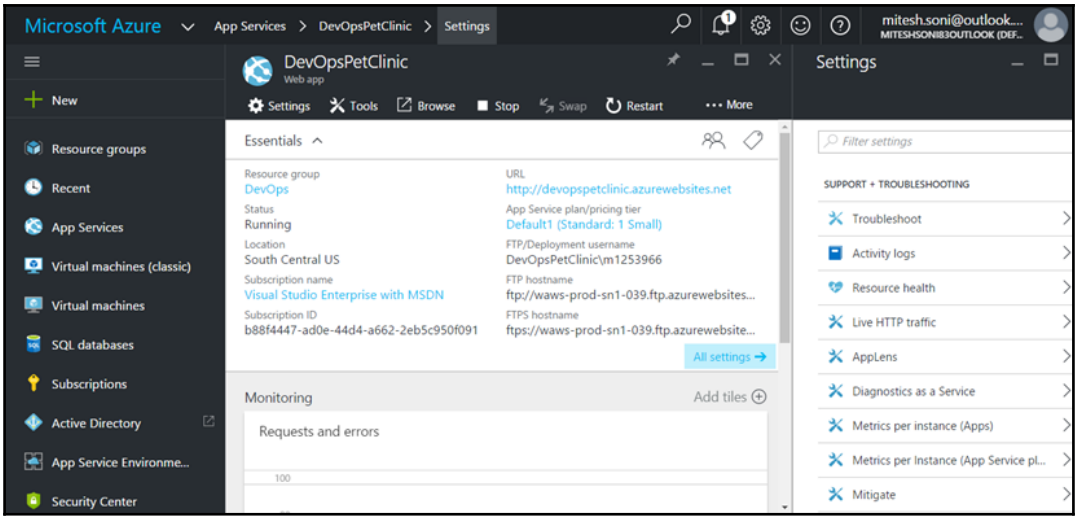
Subscription, **Resource Group**, and **App Service plan/Location**. Click on **Create**:



4. Once Azure Web App is created, verify it in Azure Portal:

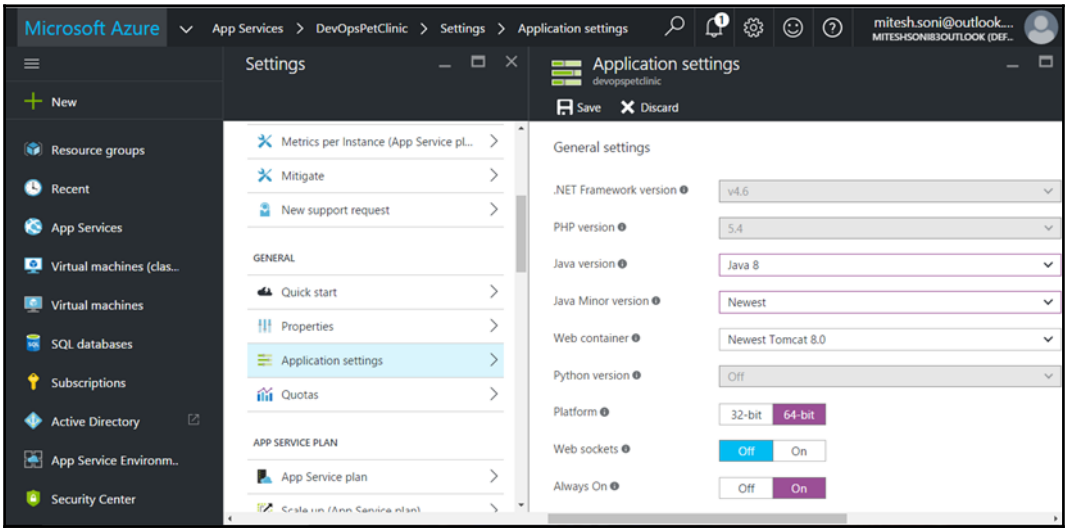


5. Click on the **DevOpsPetClinic**, get the details related to URL, Status, Location, and so on:

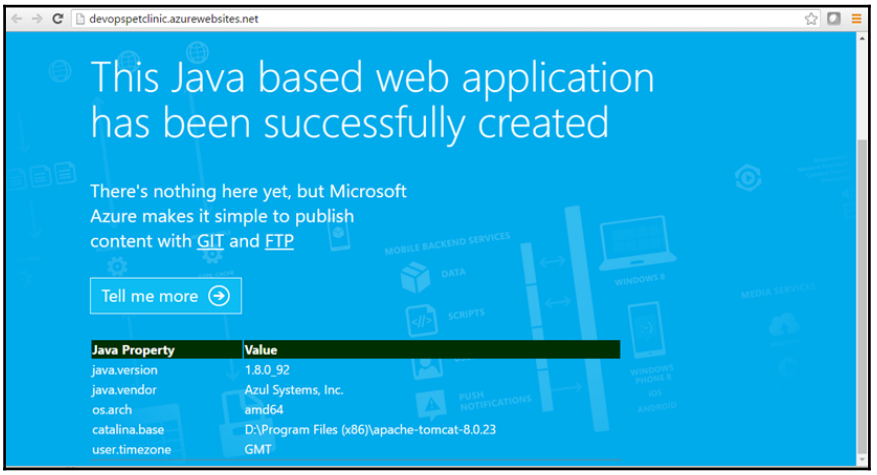


6. Click on All Settings, go to **GENERAL** Section and click on **Application settings** to configure Azure Web App for Java Web Application hosting. Select the **Java**

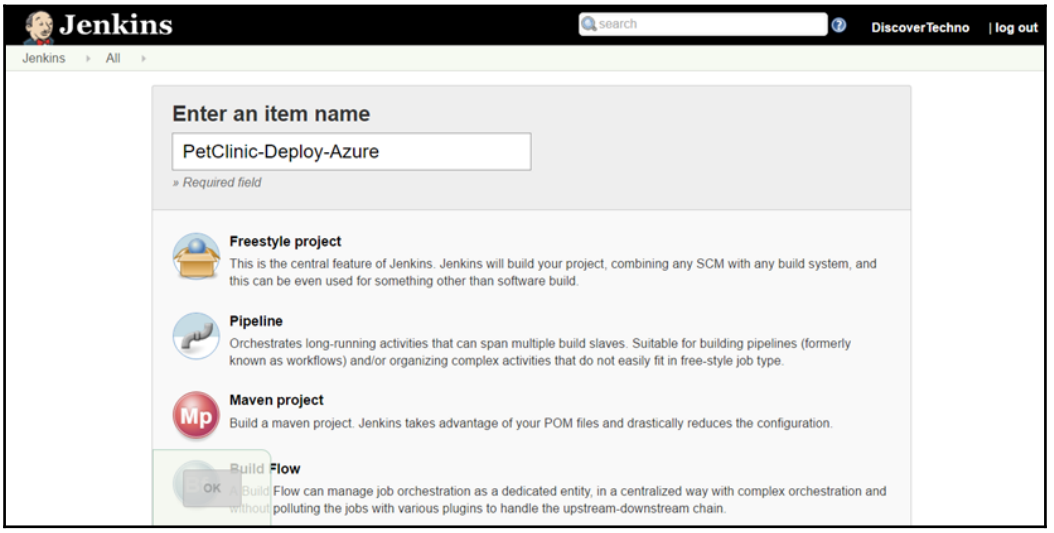
version, **Java Minor version**, **Web container**, **Platform**, and click on **Always On**:



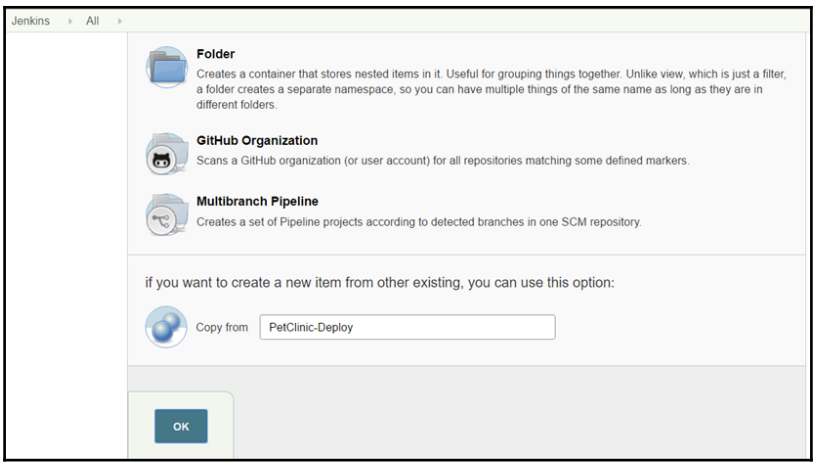
7. Visit the URL of an Azure Web App in the browser and verify whether it is ready for hosting Sample Spring application that is PetClinic.



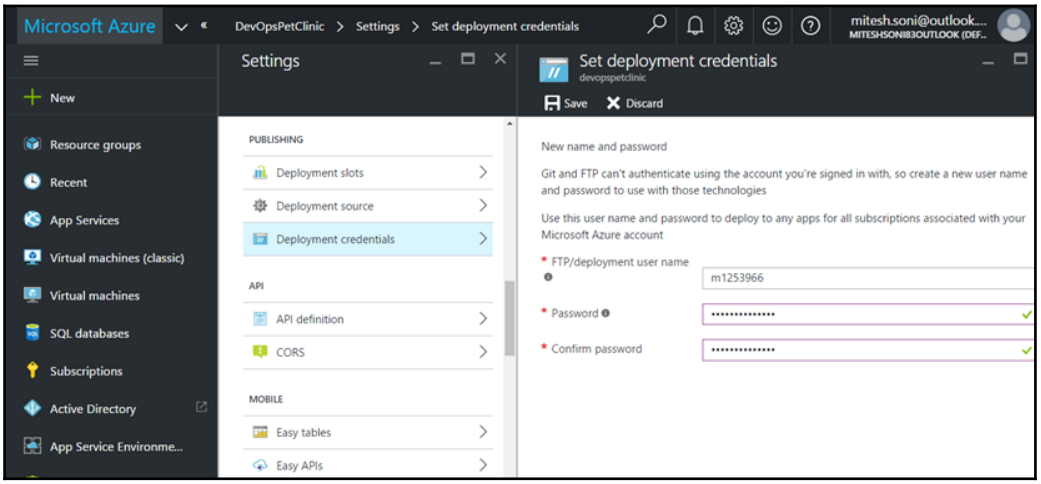
8. Let's go to Jenkins dashboard. Click on **New Item** and select **Freestyle project**:



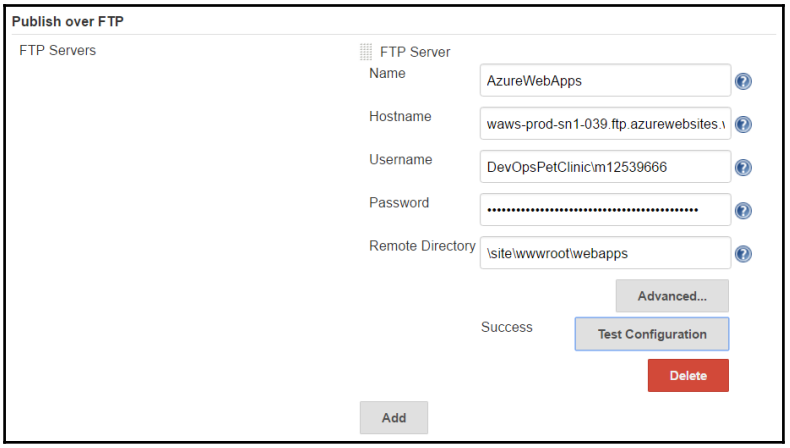
9. Copy general configuration from another build so we need not to repeat the configuration work in newly created job:



10. Click on All Settings, go to **Deployment credentials** in **PUBLISHING** section. Give user name and password. Save it:

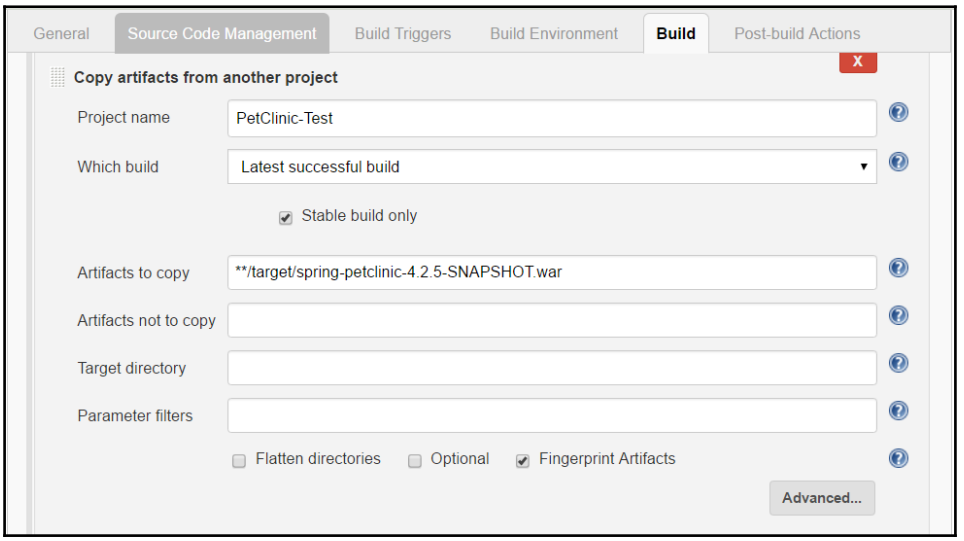


11. In Jenkins, go to **Manage Jenkins** and click on **Configure**. Configure FTP settings. Provide Hostname, Username and Password available in Azure Portal.
12. Go to devopspetclinic.scm.azurewebsites.net and get the Kudu console. Navigate to different options and find the site directory and webapps directory. Click on the **Test Configuration** and once you get Success message, we are ready to deploy our PetClinic application:

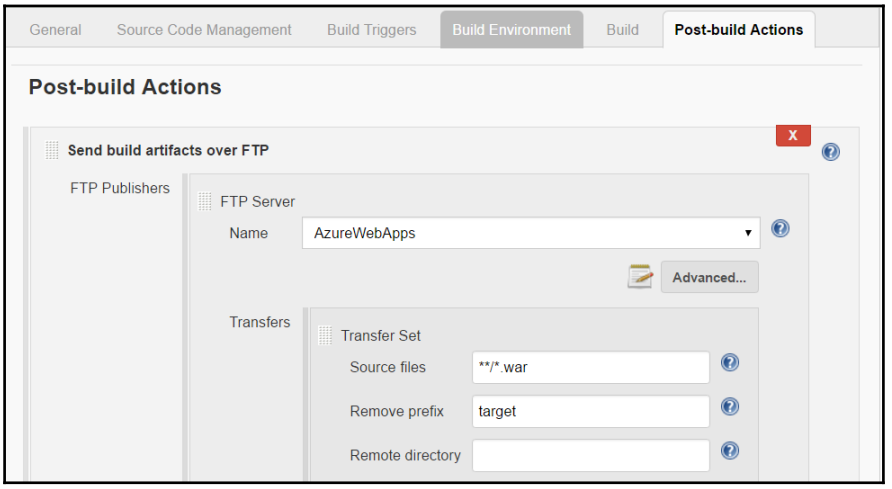


13. In the build job we created, go to **Build** section and configure **Copy artifacts from**

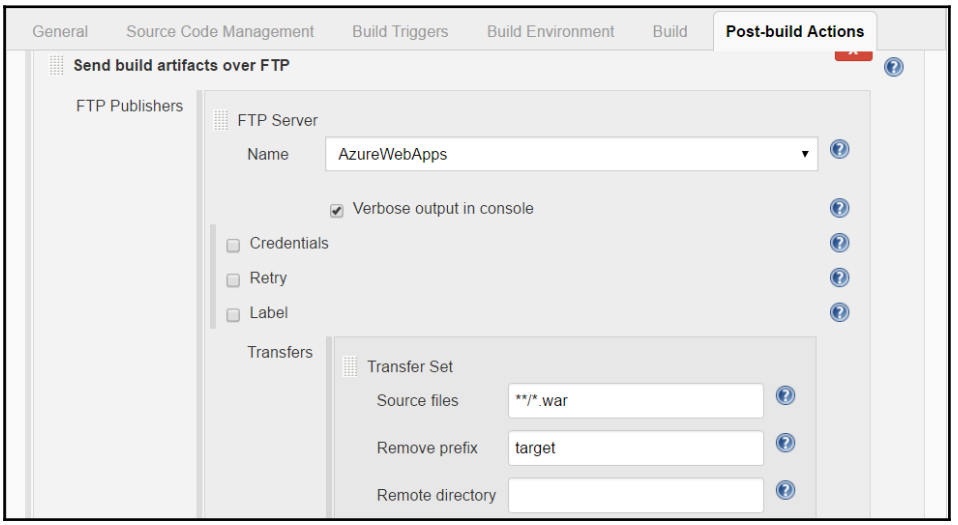
another project. We will copy war file into specific location on a virtual machine:



14. In **Post-build Actions**, click on **Send build artifacts over FTP**. Select **FTP server** name configured in Jenkins. Configure **Source files** and suffix to remove while deployment of an application in Azure Web App:



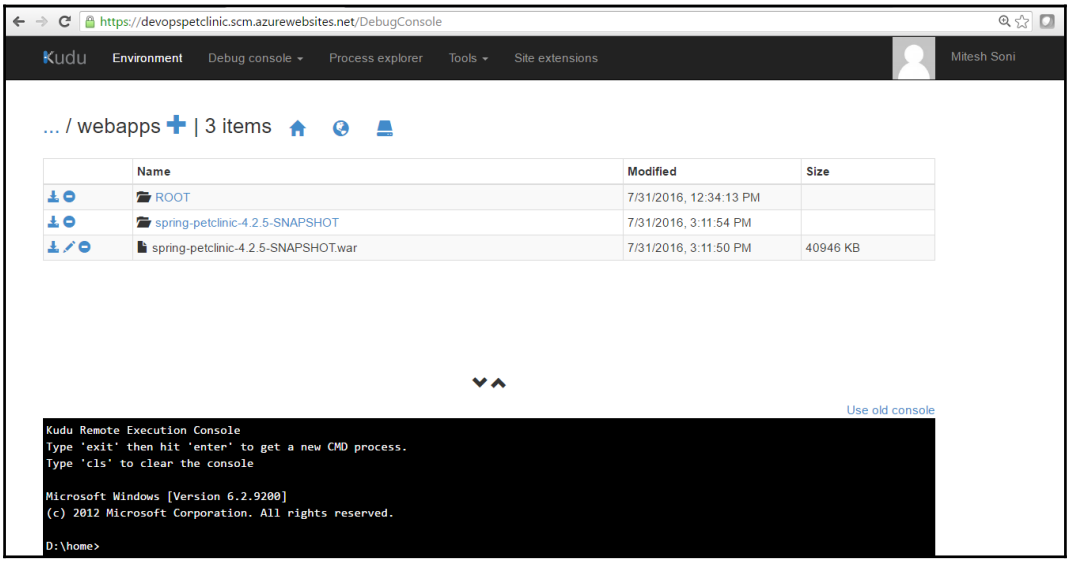
15. Click on the **Verbose output in console**:



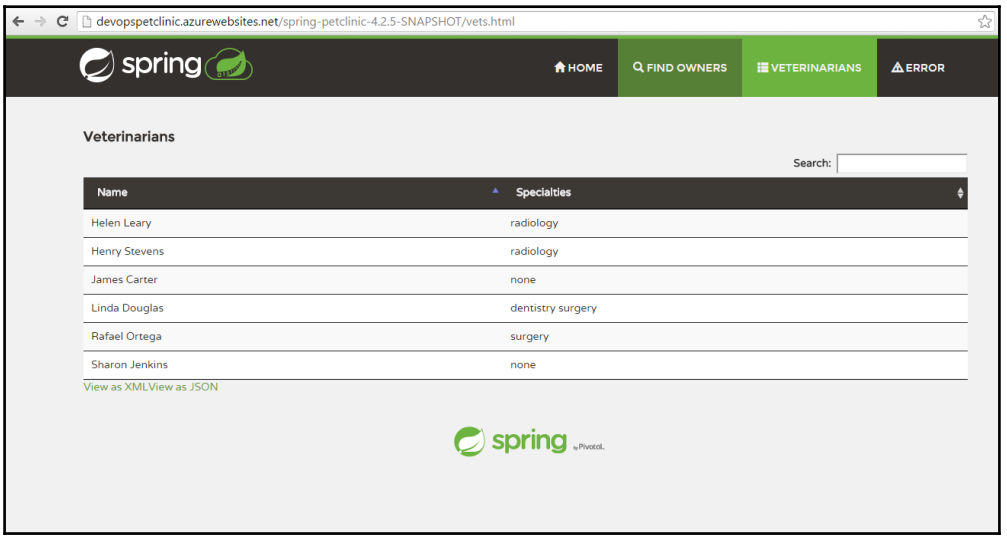
16. Click on **build now** and see what happens behind the seen:

```
Started by user DiscoverTechno
Building on master in workspace /home/mitesh/.jenkins/workspace/PetClinic-Deploy-Azure
Copied 1 artifact from "PetClinic-Test" build number 55
FTP: Connecting from host [devops1]
FTP: Connecting with configuration [AzureWebApps] ...
220 Microsoft FTP Service
FTP: Logging in, command printing disabled
FTP: Logged in, command printing enabled
CWD \site\wwwroot\webapps
250 CWD command successful.
TYPE I
200 Type set to I.
CWD \site\wwwroot\webapps
250 CWD command successful.
PASV
227 Entering Passive Mode (104,210,159,39,39,189).
STOR spring-petclinic-4.2.5-SNAPSHOT.war
125 Data connection already open; Transfer starting.
FTP: Disconnecting configuration [AzureWebApps] ...
```

17. Go to Kudu console, click on **Debug console** and go to **Powershell**. Go to **site | wwwroot | webapps**. Verify whether war file is copied or not:



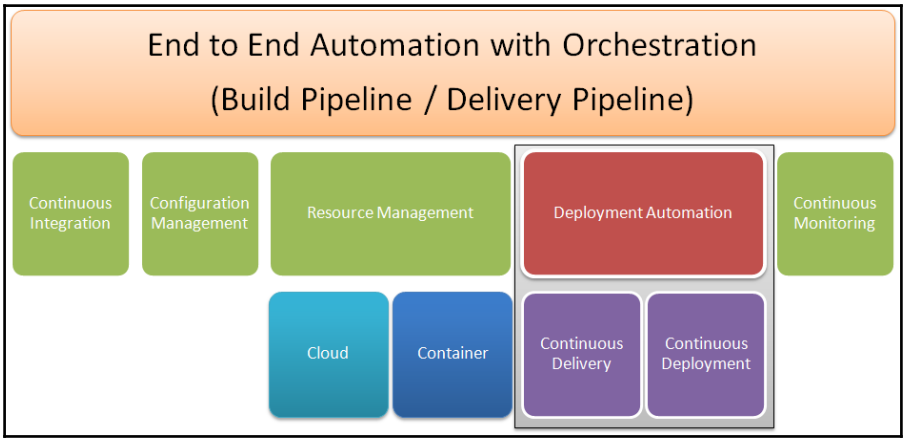
18. Visit the Azure Web App URL in the browser with the context of an application:



So we have an application deployed on Azure Web Apps.

It is important to note that FTP user name has to be with the domain. In our case, it can be Sample9888\m1253966. Direct user name without Web App name won't work.

All this different ways of deployment into AWS IaaS, AWS PaaS, Microsoft Azure PaaS, and Docker container can be used in final end to end automation:



We have covered four phases till now and now we will discuss about Continuous Monitoring and in the last Chapter we will manage all end to end automation with pipeline or orchestration.

Self-Test Questions

1. State True or False: Role and Users in Tomcat can be created in tomcat-users.xml to access Manager Web App
2. True
3. False
4. State True or False: To access Tomcat Manager App GUI Manager-script role is required.
5. True
6. False

1. State True or False: To deploy application in Tomcat container using Deploy

Plugin in Jenkins, Manager-script role is required.
2. True
3. False

1. State True or False: AWS Elastic Beanstalk and Azure App Services are a Platform as a Service (PaaS) offering from Amazon and Microsoft respectively.
2. True
3. False

1. Which of the following are steps for application deployment in AWS Elastic Beanstalk?
2. Create an Application (Petclinic)
3. Upload WAR file as an application version
4. Launch an Environment
5. Deploy new version of an application in AWS Elastic Beanstalk
6. All of the above

Summary

In this chapter, we have covered how to deploy an application in Tomcat using Tomcat Manager Application by setting Role and Users in tomcat-users.xml. We can use same deployment method where we can configure or edit tomcat-users.xml. Same approach was used for Petclinic application deployment in the Docker container.

It is a suitable approach in Infrastructure as a Service. We have also deployed Petclinic application in Platform as a Service such as AWS Elastic Beanstalk and Microsoft Azure Web App.

We have also verified what topics we have covered till now for end to end deployment for Petclinic application.

In the next chapter, we will discuss about Continuous Monitoring for Infrastructure and Application.