Chapter 4

Lab 4.1

Objective: Deploying a Java application with Chef infrastructure automation

Prerequisites:

In order to complete this lab, there will need to be a Ruby environment running on the workstation. If one is not installed, please refer to Chapter 2, Lab 2.2 for guidelines. A working Artifactory instance with a deployed **`petclinic.war`** file is also required; this should already exist if you have completed Chapter 3. The goal is to build a targeted set of Chef recipes to deploy the "PetClinic" application on a Ubuntu server environment. Because this is not intended to be a global installation, very specific code can be written that makes assumptions based on the platform. This will make the code easier to build, but more difficult to maintain, so you should make sure to take that into account when pushing code onto a server.

Introduction to Chef

ChefDK is the Chef Development Kit, which comes with the code generators and ease-of-life tools to help maintain Chef repositories. Chef itself is a development framework for configuration management code; so, like most web frameworks, the code needs to be structured in a very particular way. In this case, you will generate a standard Chef repository initialized as a Git repository. This will enable the use of the **knife** CLI tools, which will help with the use of "canned" code from the Chef Supermarket, an online repository of community-contributed "cookbooks".

In a larger production environment, you may wish to look into Chef Server and/or Hosted Chef - central management for Chef Nodes. This allows you to search against the Server for service discovery, as well as the cryptographically-signed delivery of cookbooks to authorized nodes. In this case, Chef Zero, a standalone solo node, will be used to run code on the target box.

Using Chef to Deploy Your Application

Step 1. Download and install ChefDK

1. Navigate to Chef's official site and download the current release of the Ubuntu ChefDK.

2. Use the **`dpkg`** command to install the **`.deb`** file:

```
$ dpkg -i <your_chefdk.deb>
```

Step 2. After installing Chef, build a Chef repository with the prerequisites needed to deploy PetClinic:

1. Create the repository with Chef:

   ```
   $ chef generate repo petclinic-chef
   ```

2. Change the directory to the new repository:

   ```
   $ cd petclinic-chef
   ```

3. Initialize the repository as a Git repository:

   ```
   $ git init
   ```

4. Add the code generated by the initialization to a local Git repository:

   ```
   $ git add
   ```

5. Commit the added code:

   ```
   $ git commit -m "initial commit"
   ```

6. Use the knife CLI to download cookbooks for both Java and Tomcat:

   ```
   $ knife cookbook site install java -o cookbooks/
   $ knife cookbook site install tomcat -o cookbooks/
   ```

7. Change to the `cookbook` directory:

   ```
   $ cd cookbooks
   ```

8. Create a `petclinic` cookbook:

   ```
   $ chef generate cookbook petclinic
   ```

9. Change the directory to the new `cookbook` location:

   ```
   $ cd petclinic
   ```

10. Add the following code to your newly created `petclinic` cookbook (`cookbooks/petclinic/recipes/default.rb`):

    ```
    tomcat_install "petclinicserver" do
     version '8.0.36'
    end
    ```

```
tomcat_service "petclinicserver" do
  action [:enable, :start]
  env_vars [{'CATALINA_BASE' =>
'/opt/tomcat_petclinicserver_8_0_36/'},{'CATALINA_PID' =>
'$CATALINA_BASE/bin/tomcat.pid'},{'JAVA_OPTS' =>
'-Djava.net.preferIPv4Stack=true -Djava.net.preferIPv4Addresses=true'}]
end

remote_file '/opt/tomcat_petclinicserver_8_0_36/webapps/petclinic.war' do
  source
'http://your.artifactory.server:8081/path/to/petclinic/pettclinic-1.0.0-SNAPS
HOT.war'
  owner 'tomcat_petclinicserver'
  group 'tomcat_petclinicserver'
end

directory '/home/tomcat_petclinicserver' do
  owner 'tomcat_petclinicserver'
  group 'tomcat_petclinicserver'
  mode 0755
end
```

Note: Please make sure that the path to the latest **`petclinic`** build in the Artifactory server is correct. This recipe file users resources provided by the Tomcat Chef community cookbook to download Tomcat and install it. The community cookbook does not create a home directory for the **`tomcat unix`** user, so we need to create a directory for it, so that `petclinic` has a place to store its working files.

Step 3. Create a Chef role to use for the `petclinic` install:

1. Using a text editor, create a new file in **`petclinic-chef/roles`** called **`petclinic.json`** with the following contents:

```
json
{
"run_list": [
"recipe[java]",
"recipe[tomcat]",
"recipe[petclinic]"
],
"default_attributes":{
"java": {
"jdk_version": "8",
"install_flavor": "oracle",
"accept_license_agreement": true,
"jce_enabled": true,
"oracle": {
"accept_oracle_download_terms": true
}
}
}
}
```

2. Add the default recipe and code you have modified in the **`petclinic-chef`** repository:

```
$ git add
```

3. Commit the added code:

```
$ git commit -m “Adding recipe commit”
```

4. Push your code to a remote repository, so that it can be cloned to the new VM in the next step:

```
$ git push
```

Note: You will need to create a new repository on GitHub. After you create the new repository without a **`README.md`** file, you will be given instructions on how to set up your remote repository and push.

Step 4. Create a new Ubuntu 16.04 virtual machine and clone the entire repository into the `admin` user's home directory. As the root user, run the following commands:

1. Install Chef onto the new virtual machine:

```
$ curl -L https://omnitruck.chef.io/install.sh | sudo bash
```

2. Change the directory to the newly cloned **`petclinic`** repository:

```
$ cd petclinic-chef
```

3. Run the **`petclinic`** installed from the **`petclinic.json`** run list:

```
$ sudo chef-client -z -r 'role[petclinic]'
```

4. Validate that Tomcat is running:

```
$ service tomcat_petclinicserver status
```

When the Chef converge run is complete, **`petclinic`** should run on the server's IP address at **`port 8080/petclinic`**.

Note: If you have other Vagrant boxes running, you may be using `port 8080` for something like Jenkins. In order to do so, you need to change the `Vagrantfile` of this box to map the `8080` client to another port locally on the host. As a result, you should be able to reach the application.

For instance, I remapped locally and my PetClinic URL was `http://localhost:8090/petclinic`, so I could keep Jenkins running on the default port.