**Manually Setting Up Jenkins onto an EC2, Creating a Build Project, and Deploying a Target Application to the EC2**

*Note 1: This method of setting up an EC2, running Jenkins builds, and deploying applications to the EC2 can be shorted by writing scripts to handle most of the Linux commands on the EC2 side. Instead of moving local files to the EC2, you could host the files in an S3 bucket and use* wget *to retrieve them for software installations on the EC2.*

*Note 2: PuTTy is also optional, as we can configure Git to allow for OpenSSH to be used on the Windows Command Prompt (much more convenient, and cuts out steps 4 through 6)*

1) Create a Java web application

2) Pushed the Java web app to a new GitHub repository (not a branch)

3) Created an EC2 instance on AWS

- AMI: ami-467ca739

- Instance Type: t2.micro

- Configure Instance Details: leave default

- Add Storage: leave default

- Add Tags: optionally add descriptive tags here

- Configure Security Group:

+ SSH; Port 22; Source = My IP

+ Custom TCP Rule; Port 8080; Source = Anywhere

- Generate a key pair (you get the private key)

4) Use PuTTy Gen to convert the .pem file we received from AWS into a .ppk file that will be used to allow us to SSH into our EC2 instance

5) Configured PuTTy to connect to our EC2 instance:

- Session

+ Host Name = Public DNS address of EC2 instance

+

- Connection - Data

+ Auto-login username: ec2-user

- Connection - Auth

+ Private key file for authentication: path to the .ppk file

6) Go back to "Session" in PuTTy and save the configuration to a session and then click "Open"

7) The EC2 ASCII logo should be displayed and the terminal will open within the home (`~`) directory of the EC2

8) Use pscp -i (if you are using PuTTY, otherwise use scp -i) (to move the Tomcat, Java, and Maven files over from our local machine to our EC2 instance

pscp -i <file path to private key> <file path to source-file> ec2@<public DNS address>:/tmp/<same file name as the source file>

9) Create the structure of our EC2

- From the root/usr directory (/usr):

ec + sudo mkdir java

+ sudo mkdir apache

- From the home directory (~):

+ sudo mkdir .git

+ sudo mkdir .m2

10) Give proper permissions to these files:

- From the root directory (/):cd

+ sudo chmod 777 /usr/java

11) Install software:

l- From /usr/java, execute:

+ sudo rpm -ivh /tmp/<target jdk .rpm file>

- From /usr/apache, execute:

+ sudo tar xvf /tmp/<target Maven .tar.gz file>

+ sudo tar xvf /tmp/<target Tomcat tar.gz file>

- From ~/.git, execute:

+ sudo yum install git

12) Permissions for new files

- From /usr/apache, execute:

+ sudo chmod 777 <name of Maven folder>

+ sudo chmod 777 <name of Tomcat folder>

13) Later we will need to get into the Tomcat bin and webapps directories, so add permissions for themc as well

- From /usr/apache, execute:

+ sudo chmod 777 <name of Tomcat folder>/bin

+ sudo chmod 777 <name of Tomcat folder>/webapps

14) Configure Tomcat with two roles, and two usernames (one for the 8080 login page, another for Maven and Tomcat to use)

- From /usr/apache/<name of Tomcat folder>, execute:

+ sudo vim conf/tomcat-users.xml

15) With VIM open change the roles and usernames to look like this:

<role rolename="manager-gui"/>

<role rolename="manager-script"/>

<user username="manager-gui" password="tomcat" roles="manager-gui"/>

<user username="manager-script" password="script" roles="manager-script"/>

16) Configure Tomcat to allow Jenkins to log into Tomcat’s Manager Console from an IP address that is different than Tomcat’s:

- From /usr/apache/<name of Tomcat folder>, execute:

+ sudo vim webapps/manager/META-INF/context.xml

17) With VIM open, comment out the element <Valve className=...>

18) Increase the file upload size from 50 MB to 134 MB:

- From /usr/apache/<name of Tomcat folder>`, execute:

+ sudo vim webapps/manager/WEB-INF/web.xml

19) With VIM open, change the values within the elements: <max-file-size> and <max-request-size> to the value = 134217728

20) Configure environment variables:

- From the home directory (~), execute:

+ sudo vim .bashrc

`21) With VIM open, add the following:

JAVA\_HOME=/usr/java/latest

export JAVA\_HOME

CATALINA\_HOME=/usr/apache/<name of Tomcat folder>

export CATALINA\_HOME

M2\_HOME=/usr/apache/<name of Maven folder>

export M2\_HOME

M2=$M2\_HOME/bin

export M2

export PATH=$M2:$PATH

22) Configure the location of the local Maven repository

- From /usr/apache/<name of Maven folder>, execute:

+ `sudo vim conf/settings.xml`

23) With VIM open, uncomment out the <localRepository> tags and put the path to your `.m2` directory: `/home/ec2-user/.m2`

24) Check to see that Tomcat is configured and working properly

- From /usr/apache/<name of Tomcat folder>/bin, execute:

+ sudo ./startup.sh

25) In the browser navigate to: `<Public DNS address of EC2>:8080 (You should see Tomcat)

26) In Tomcat (within the browser), click on the Manager App button and use the login credential we set in Step 15

- username: manager-gui

- password: tomcat

27) If you see the **Tomcat Web Application Manager** page, then you have correctly configured Tomcat.

28) Back in PuTTy, shutdown Tomcat:cd

- From /usr/apache/<name of Tomcat folder>/bin, execute:

+ sudo ./shutdown.sh

29) Deploy Jenkins by using pscp -i to move the jenkins.war file from the local machine to the EC2 instance:

- pscp -i <file path to private key> <file path to local Jenkins war> ec2-user@<EC2 Public DNS address>:/tmp/jenkins.war

30) Move the `jenkins.war` file from tdhe `/tmp` folder to the `webapps` folder of `CATALINA\_HOME`:

- From /usr/apache/<name of Tomcat folder>/webapps, execute:

+ sudo mv /tmp/jenkins.war .

31) Go back to the bin directory of CATALINA\_HOME and execute:

- sudo ./startup.sh

32) The Jenkins splash screen should appear, in PuTTy navigate to the folder on the page to get the secret initial password to login to Jenkins

- First give permission to access root and .jenkins:

+ sudo chmod 777 /root

+ sudo chmod 777 /root/.jenkins --recursive

33) Print the value of the initialAdminPassword to the console:

- From the root directory (/), execute:

+ cat root/.jenkins/secrets/initialAdminPassword

34) Once in, choose the Install Suggested Plugins option, and create an admin user on the following page. (REMEMBER THESE CREDENTIALS)

35) From your Jenkins dashboard, go to Manage Jenkins >> Manage Plugins and install the following plugins:

- GitHub Integration

- Maven Integration

- Slack Notification

- Deploy to Container

36) Once the plugins are installed, go back to the Jenkins Dashboard and navigate to Manage Jenkins >> Global Tool Configuration

- Tell Jenkins where it can find the settings.xml for Maven

+ Default Settings Provider = Settings in file system

+ File Path = /usr/apache/<name of Maven folder>/conf/settings.xml

- Tell Jenkins where it can find the JDK

+ Name = jdk-8u171

+ JAVA\_HOME = /usr/java/latest

- Tell Jenkins where to find MAVEN\_HOME

+ Name = maven-3.5.3

+ MAVEN\_HOME = /usr/apache/<name of Maven folder>

- Tell Jenkins where to find Git

+ Name = jenkins-git

+ Path to Git executable = git

37) In GitHub, go to your repository >> Settings

- Choose Integrations and Services

+ Add the Jenkins (GitHub plugin) service

\* Jenkins hook URL = http://<EC2 Public DNS>:8080/jenkins/github-webhook

38) In your GitHub user settings go to Developer Settings >> Personal Access Tokens

- Choose to generate a new user access token

- Give it a description (jenkins-demo), and assign it repo scope.

- Click Generate Token and copy the alphanumeric string which is generated.

39) Back in the Jenkins Dashboard navigate to Manage Jenkins and choose Configure System

- Under GitHub, choose Add GitHub Server

+ Name = jenkins-demo

+ API\_URL = https://api.github.com

+ Credentials

\* Click Add >> Jenkins

= Kind: Secret Text

= Secret: <Personal Access Token from GitHub>

= ID: jenkins-predemo-token

40) Test the connection to your repository by clicking Test Connection.

41) Configure Slack to allow Jenkins to notify us of build successes/failures

- In your Slack channel, click the settings icon and choose Add an app

42) This will bring you to the browser, from here click Add Configuration and copy the Base URL and Integration Token. Go to the bottom of the page and click Save

43) Back in Jenkins, still in the Configure System menu:

- In the Global Slack Notifier Settings section:

+ Base URL = <Base URL from Slack>

+ Integration Token = <Integration Token from Slack

+ Channel = <the channel you want notifications in>

44) Click Test Connection to check for success

45) Create new Jenkins Project

- Back in the Dashboard click New Item

+ Choose Maven Project >> OK

46) In the project setup:

- Select GitHub hook trigger for GitSCM polling for the build trigger.

- Source Code Management

+ Repository URL: <URL to your GitHub repo>

+ Branch Specifier: master

- In the Build Section:

(This might be wrong, `pom.xml` worked for me, but `jenkins-predemo/pom.xml` didn’t) + Room POM = jenkins-predemo/pom.xml

+ Goals and Options = clean package

- Post-build Actions

+ Deploy war/ear to a container

\* WAR/EAR files: \*\*/\*.war

(This might be wrong, it worked for me without the “jenkins-predemo” but not with it)\* Context path: jenkins-predemo

\* Click Add To Container

* Add >> Credentials
* Username: manager-script
* Password: script
* URL: http://localhost:8080

- Slack Notifications

+ Check: Notify Failure

+ Check: Notify Success

47) Save the project, and build! If the build target is correctly configured, then Jenkins will pull it from GitHub, build it, and deploy the application to our EC2 instance.