**Deploying Maven App to Tomcat Server**

This section will guide you to:

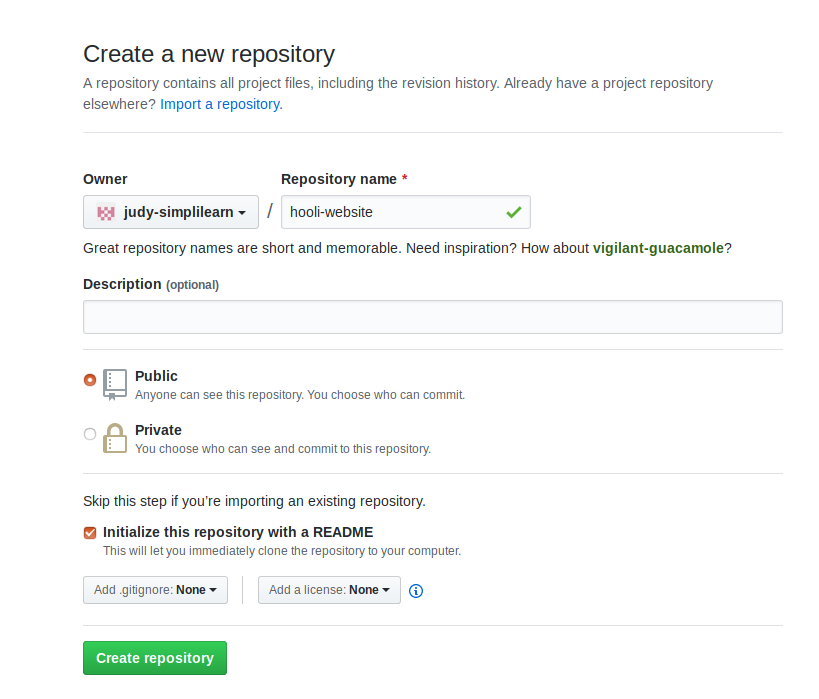
* Create an EC2 instance
* Create and run a Tomcat instance on an EC2 VM
* Create a Maven webapp
* Configure a Jenkins build job to compile and deploy a Maven app to a Tomcat server

**Step 1:** Creating a Git repository for the webapp

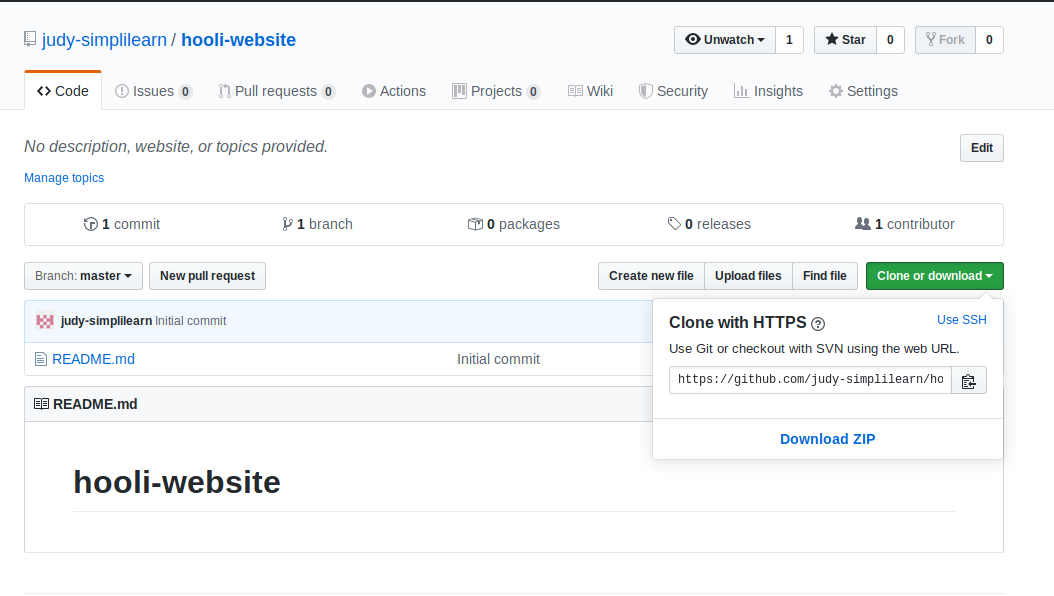
* Log in to your Github account.
* Click on the plus icon next to the profile picture and select *New repository* from the drop-down menu.



* Fill the required fields in the create repository form and give the name: hooli-website as shown below:



* Click on the **Create Repository** button.
* Click on the **Clone or download** button and copy the URL.



**Step 2:** Adding the code for the webapp to the repository

* Open the terminal
* Run **mvn archetype:generate -DgroupId=com.hooli.app -DartifactId=welcome-app -DarchetypeArtifactId=maven-archetype-webapp -DarchetypeVersion=1.4 -DinteractiveMode=false** to generate a maven project.
* Navigate inside the Maven app (**cd welcome-app/src/main/webapp**) and open the index.jsp file
* Add the following content to file and save it:

**<html>**

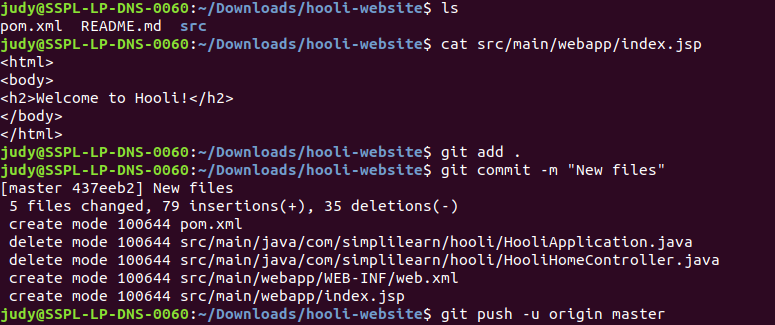
**<body>**

**<h2>Welcome to Hooli!</h2>**

**</body>**

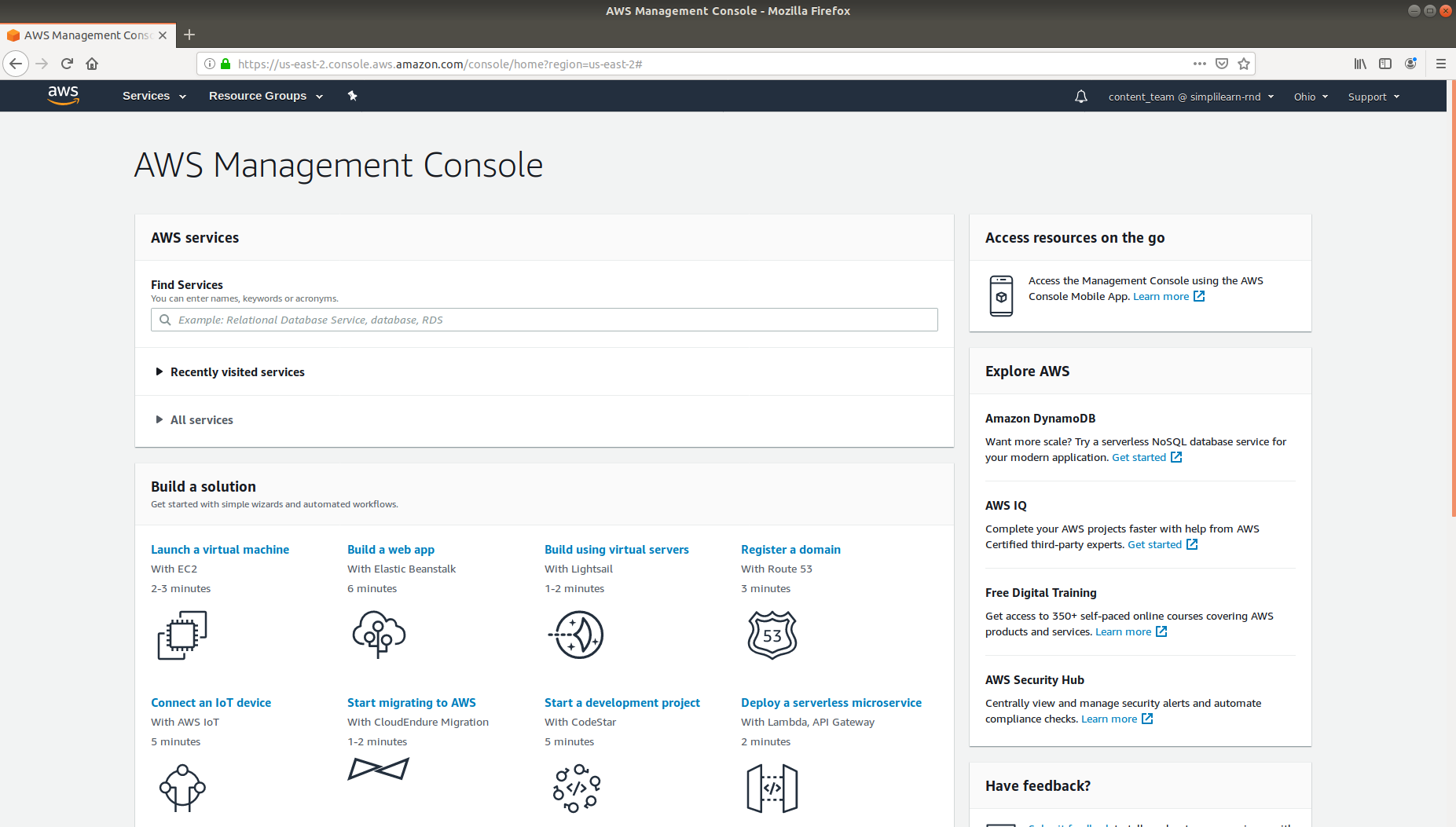
**</html>**

* Navigate outside the Maven directory.
* Run **git clone [URL]** to clone the repository.
* Move the Maven repository files inside the clone directory.
* Commit the changes to the remote SCM.
* Run **git add .**
* Run **git commit -m "Add new files"**
* Run **git push -u origin master**(**Please Note:** In case you get an error telling that “master” is not specified, try to execute **git push -u origin main**)

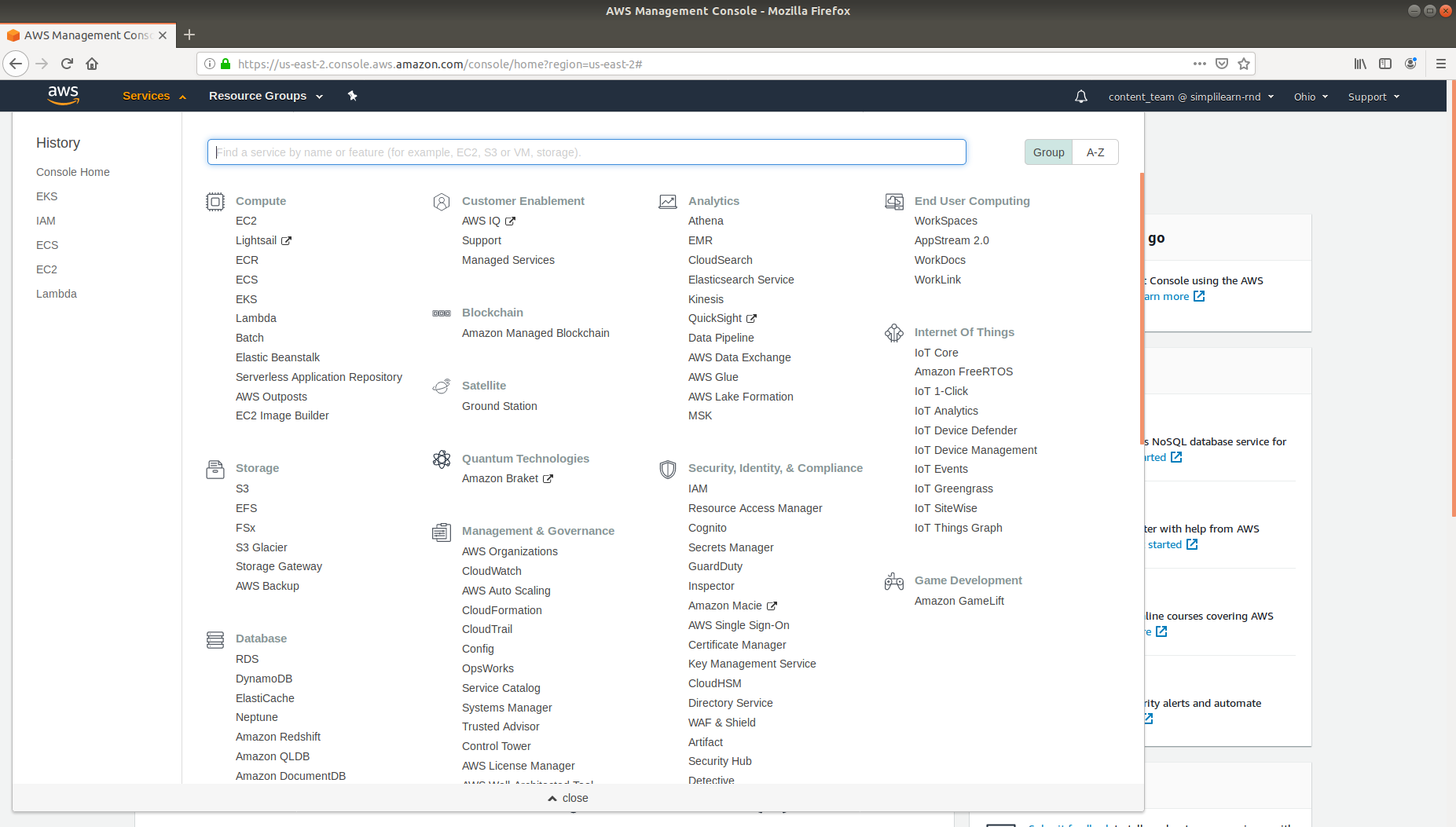


**Step 4:** Creating an EC2 instance

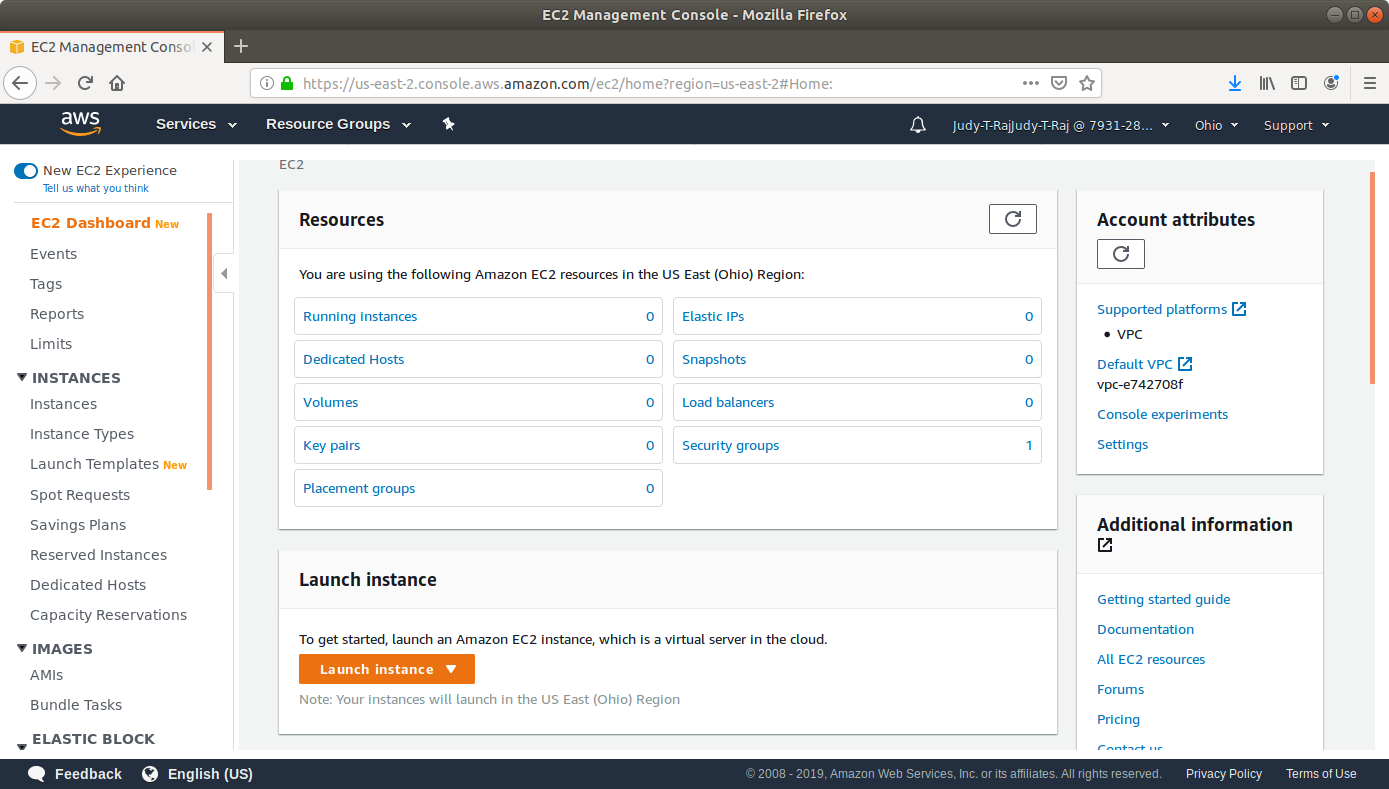
* Log in to the AWS lab account provided. You will then be able to see the following screen:



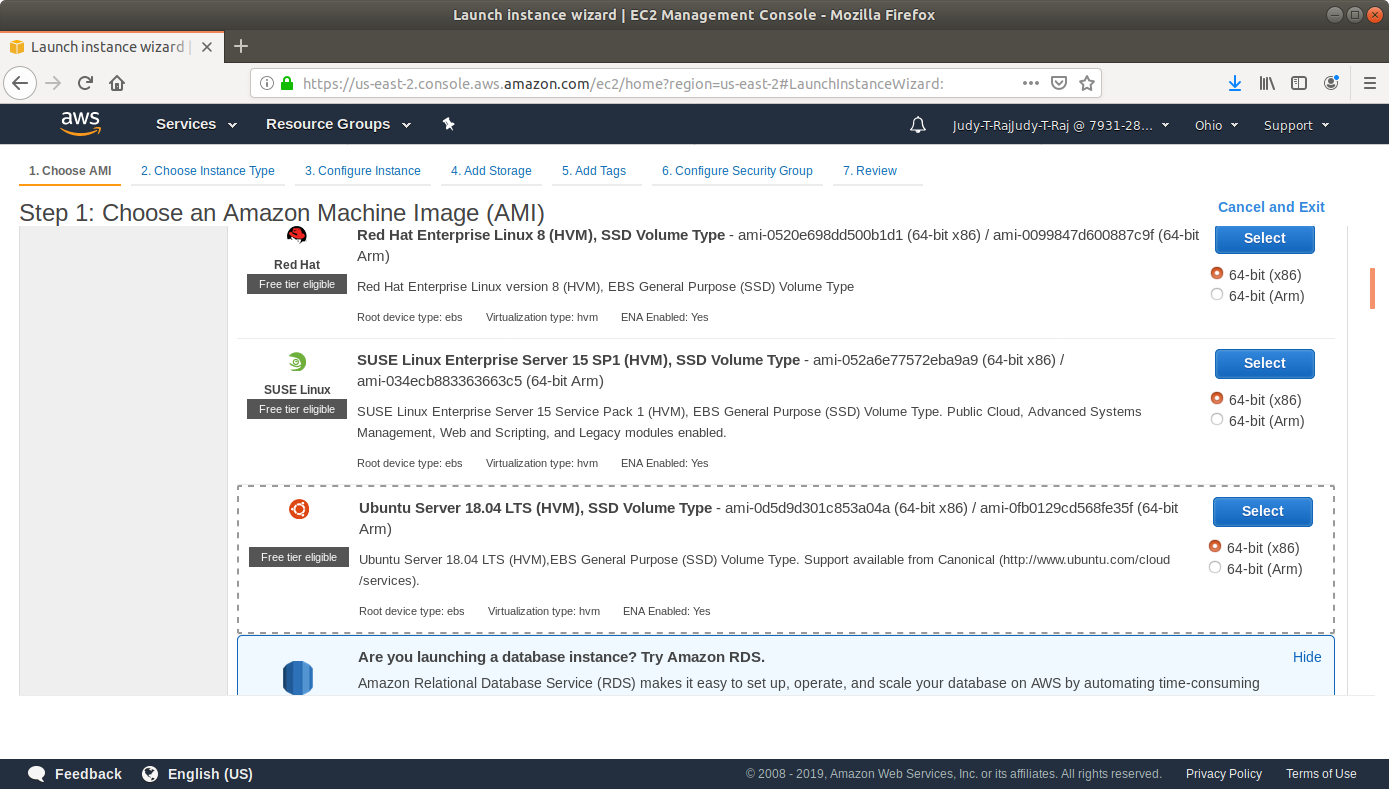
* Click on Services at the top left to view the drop-down list of resources.
* Click on EC2 under the Compute menu from the drop-down list.



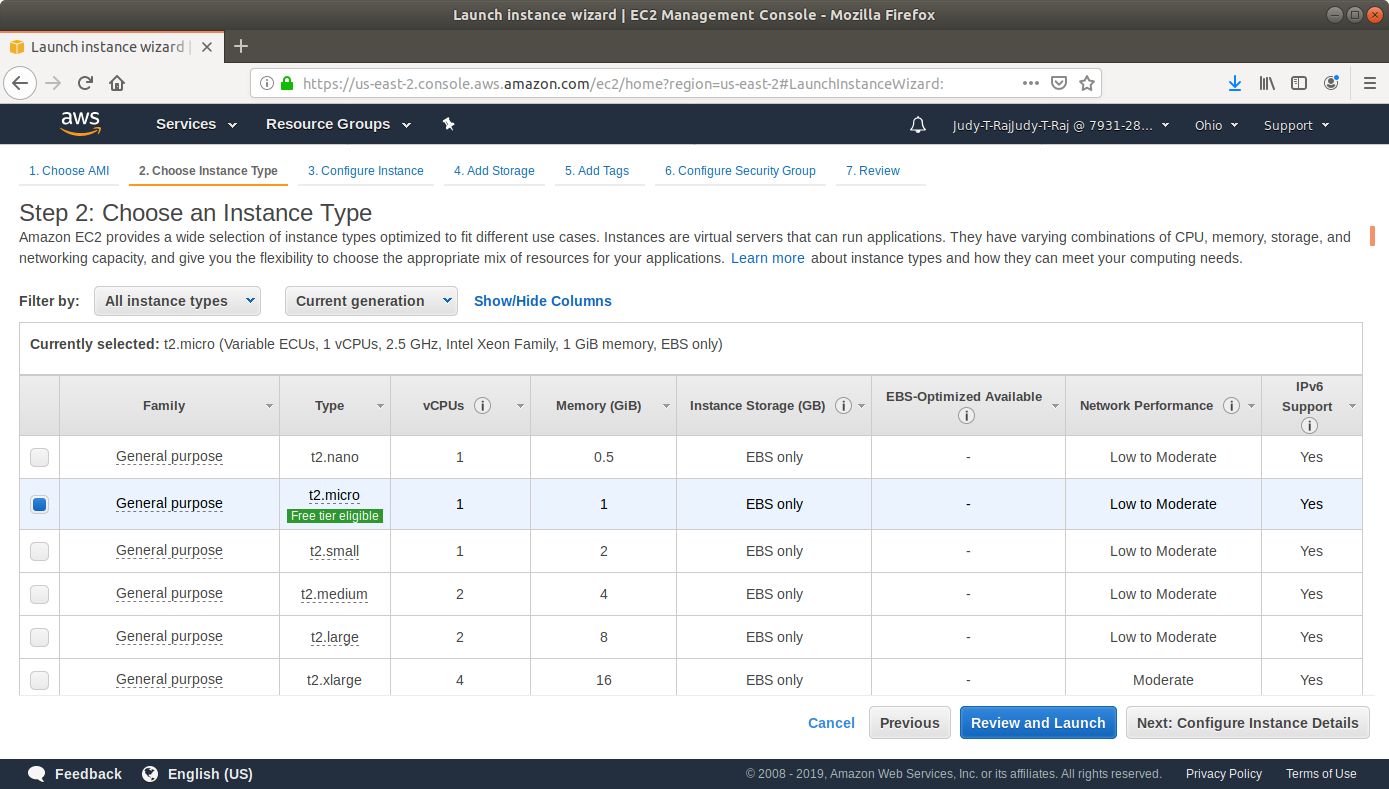
* Click on the Launch Instance button and select Launch Instance from the menu.



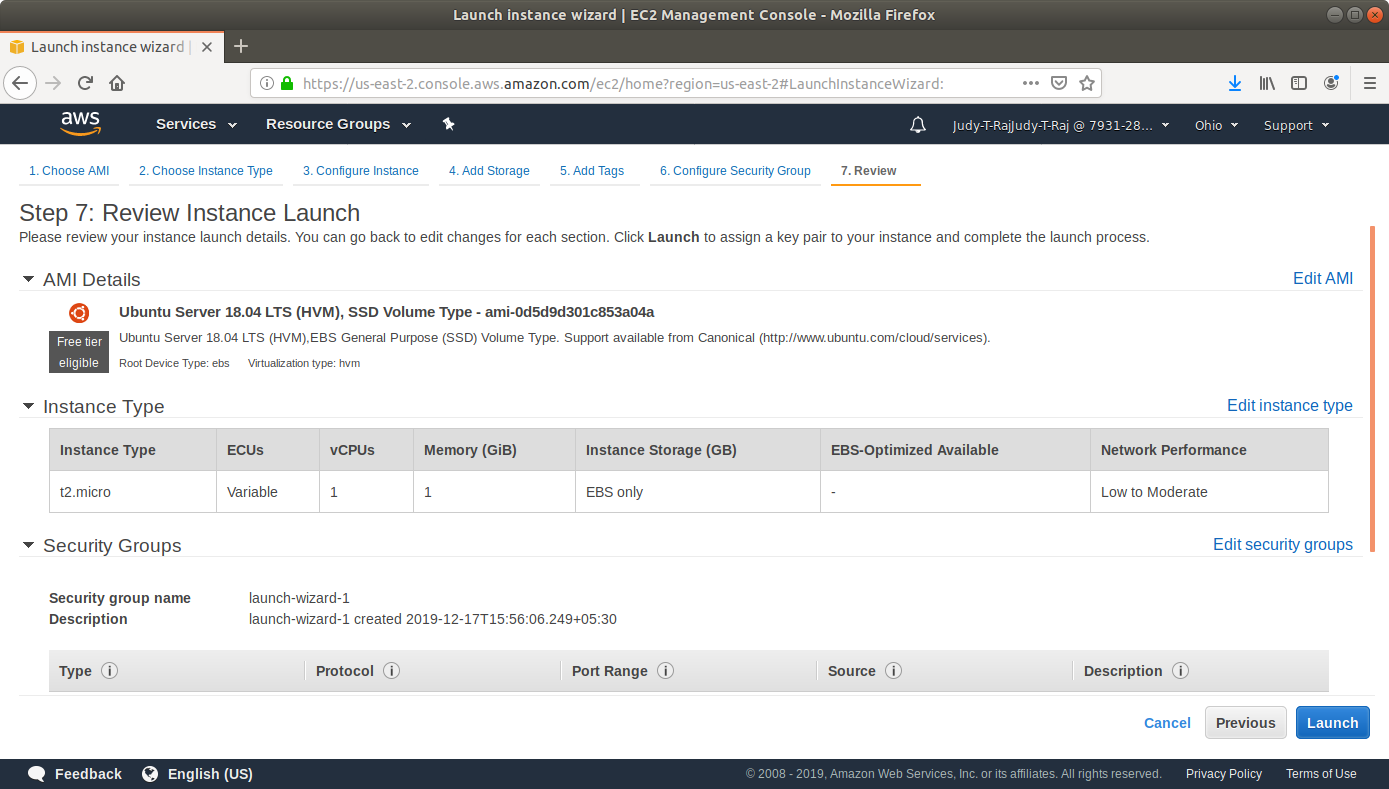
* Choose an Amazon Machine Image (AMI) from the list of AMIs and click on Select.



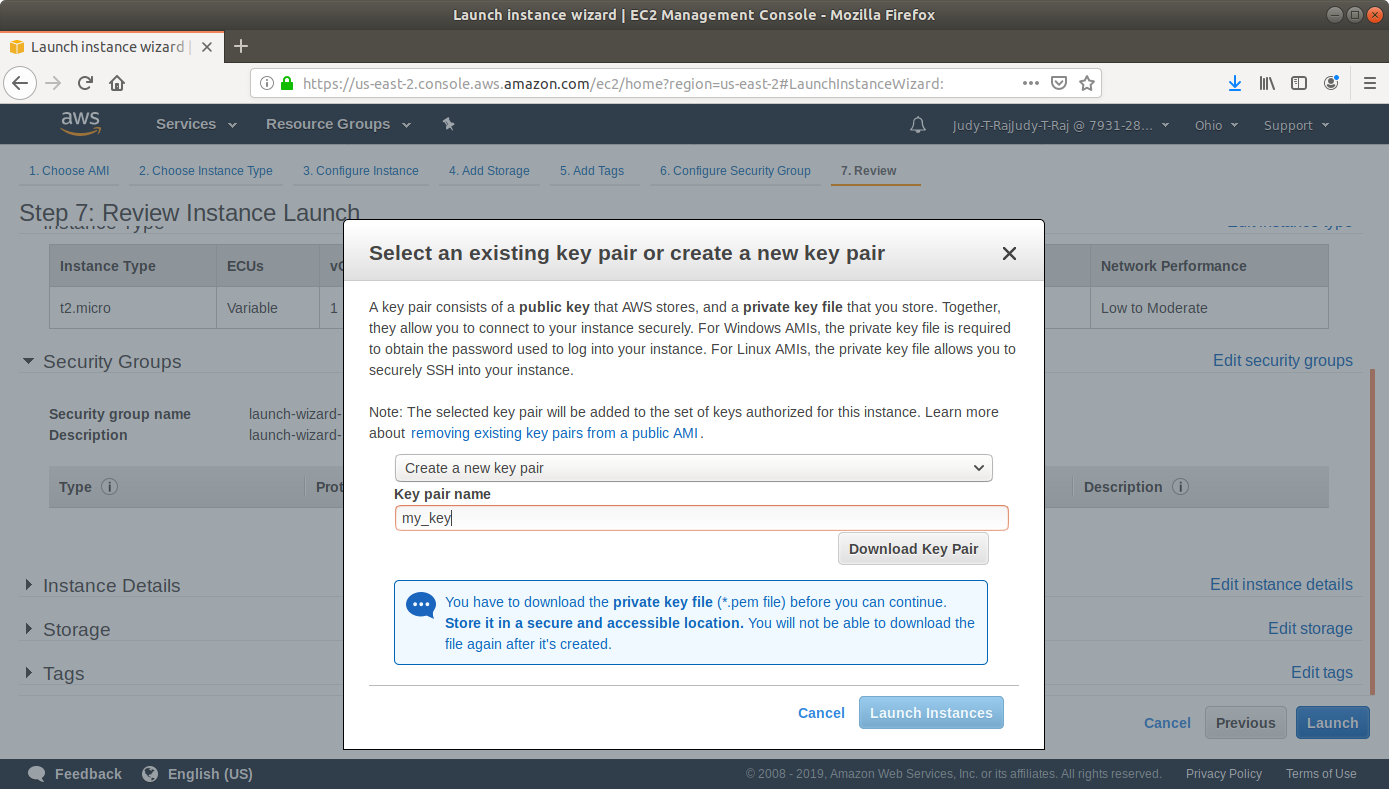
* Choose an Instance Type and click Review and Launch.



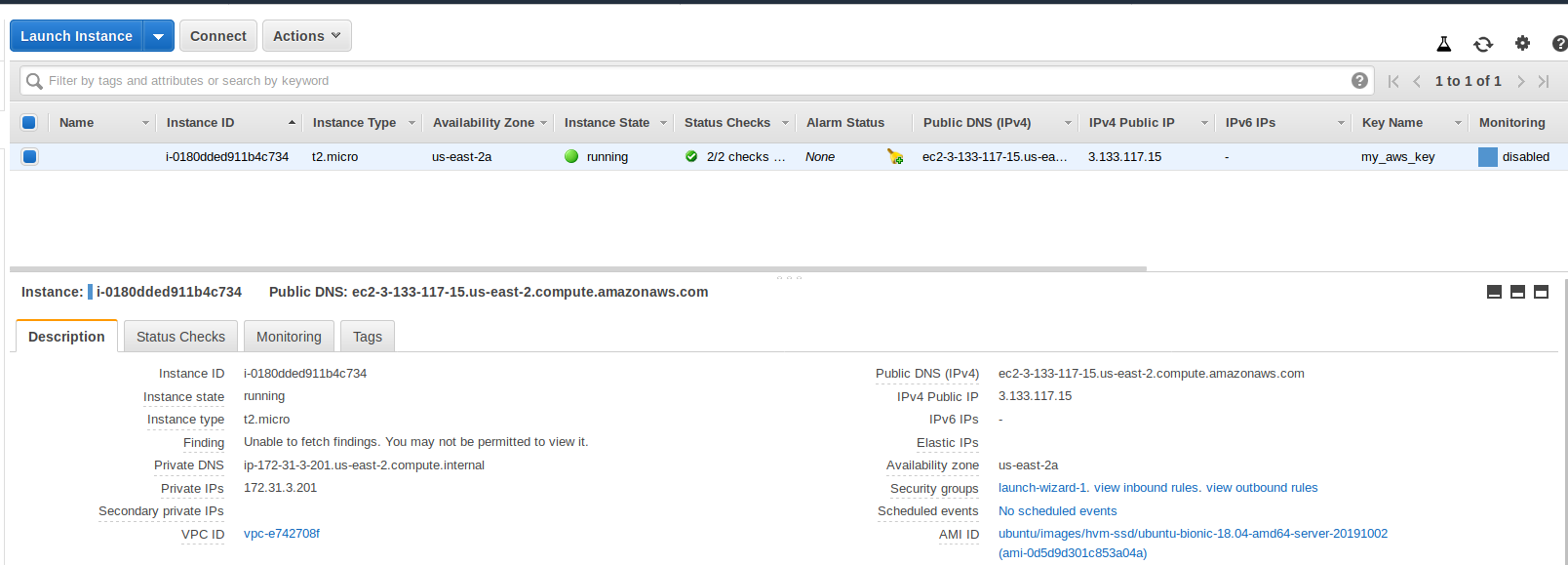
* Click on Launch.



* In the pop-up menu, select **Create a new key-value pair**.
* Click on **Download** Key. You’ll need this key to SSH to the VM later.



* Click on Launch.
* Navigate to the security groups console .



* Add a rule to the security group to which the instance belongs to allow SSH, with the following settings:

**Type:** SSH

**Protocol:** TCP

**Port Range:** 22

**Source:** Anywhere 0.0.0.0/0

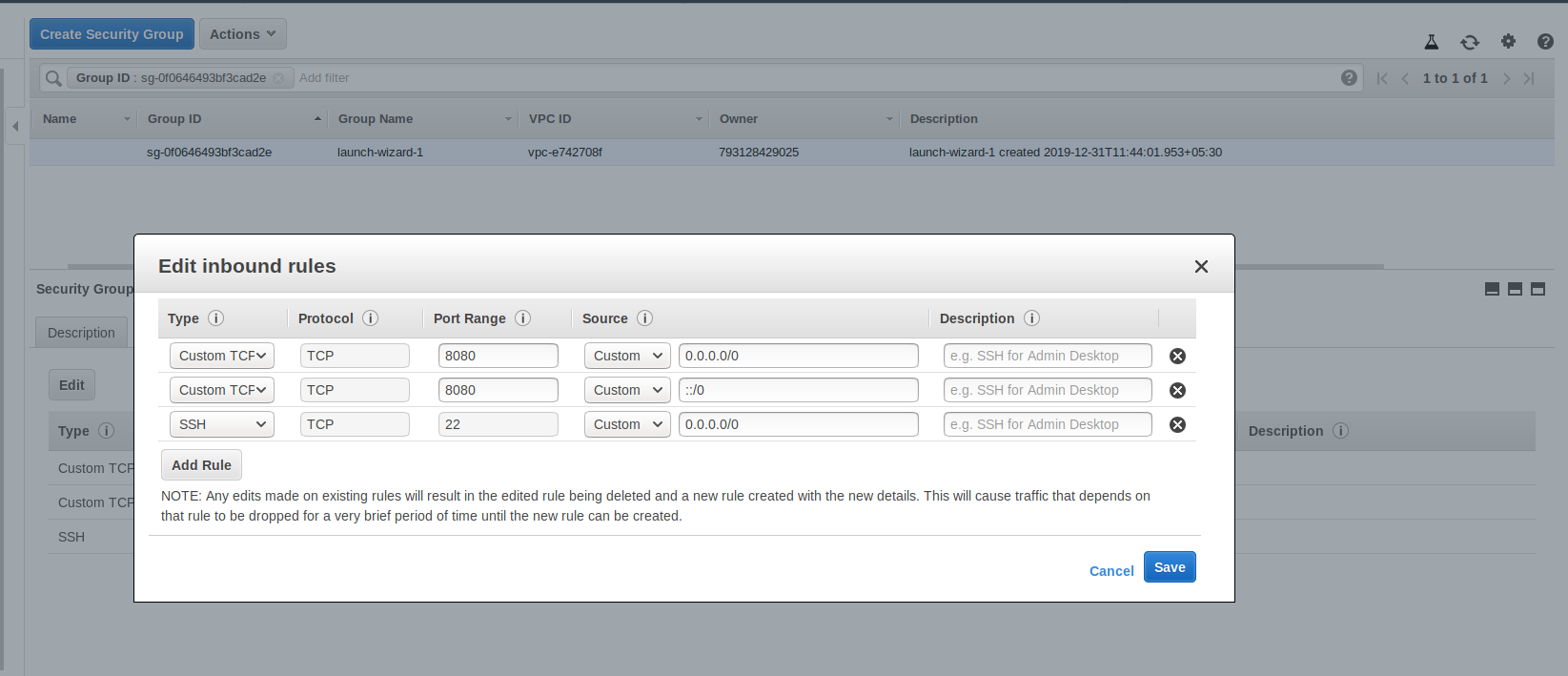
* Add a rule to the security group to which the instance belongs to allow http traffic to port 8081, with the following settings:

**Type:** Custom TCP Rule

**Protocol:** TCP

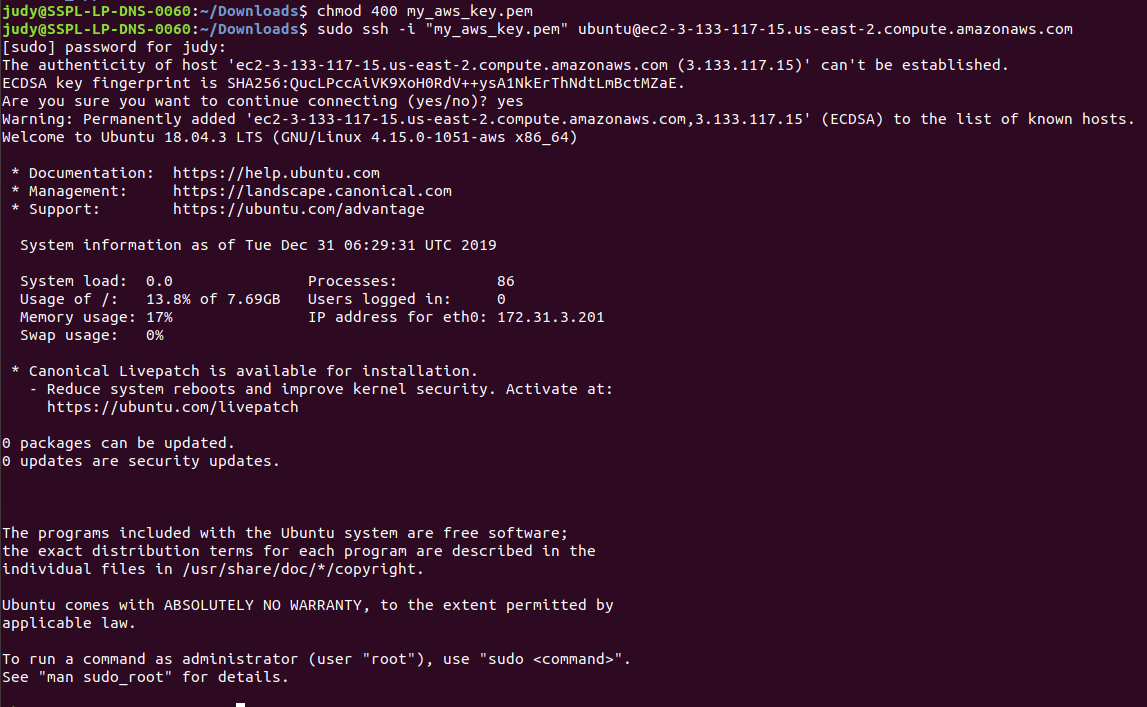
**Port Range:** 8081

**Source:** Anywhere 0.0.0.0/0



**Step 5:** Installing Tomcat on EC2

* Open the terminal.
* Navigate to the location where the AWS key is stored.
* Make the key file executable with the command **chmod 400 <key-name>.pem**
* SSH to the EC2 instance with the command **sudo ssh -i <key-name>.pem ubunutu@<public-dns>**



* Run the following commands to install Java and Tomcat and grant permissions to the Tomcat user:

**sudo apt-get update**

**sudo apt-get install default-jdk**

**sudo groupadd tomcat**

**sudo useradd -s /bin/false -g tomcat -d /opt/tomcat tomcat**

**cd /tmp**

**curl -O** [**http://mirrors.estointernet.in/apache/tomcat/tomcat-9/v9.0.41/bin/apache-tomcat-9.0.41.tar.gz**](http://mirrors.estointernet.in/apache/tomcat/tomcat-9/v9.0.41/bin/apache-tomcat-9.0.41.tar.gz)

**sudo mkdir /opt/tomcat**

**sudo tar xzvf apache-tomcat-9.0.41.tar.gz -C /opt/tomcat --strip-components=1**

**cd /opt/tomcat**

**sudo chgrp -R tomcat /opt/tomcat**

**sudo chmod -R g+r conf**

**sudo chmod g+x conf**

**sudo chown -R tomcat /opt/tomcat**

**sudo chown -R tomcat webapps/ work/ temp/ logs/**

* Find the path to Java with the following command:

**sudo update-java-alternatives -l**

* Open the tomcat.service file with the command **sudo vi /etc/systemd/system/tomcat.service**
* Add the following content to the file. Replace the JAVA\_HOME value with the value obtained in the previous step:

**Description=Apache Tomcat Web Application Container**

**After=network.target**

**[Service]**

**Type=forking**

**Environment=JAVA\_HOME=/usr/lib/jvm/java-1.11.0-openjdk-amd64**

**Environment=CATALINA\_PID=/opt/tomcat/temp/tomcat.pid**

**Environment=CATALINA\_HOME=/opt/tomcat**

**Environment=CATALINA\_BASE=/opt/tomcat**

**Environment='CATALINA\_OPTS=-Xms512M -Xmx1024M -server -XX:+UseParallelGC'**

**Environment='JAVA\_OPTS=-Djava.awt.headless=true -Djava.security.egd=file:/dev/./urandom'**

**ExecStart=/opt/tomcat/bin/startup.sh**

**ExecStop=/opt/tomcat/bin/shutdown.sh**

**User=tomcat**

**Group=tomcat**

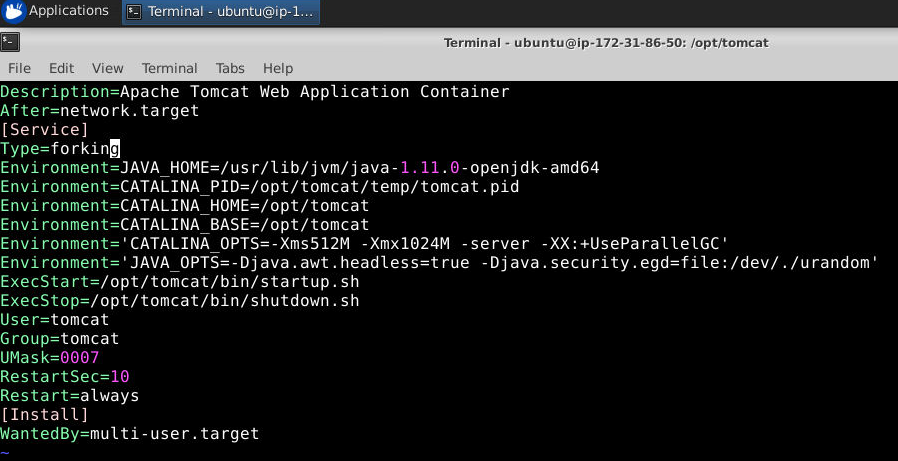
**UMask=0007**

**RestartSec=10**

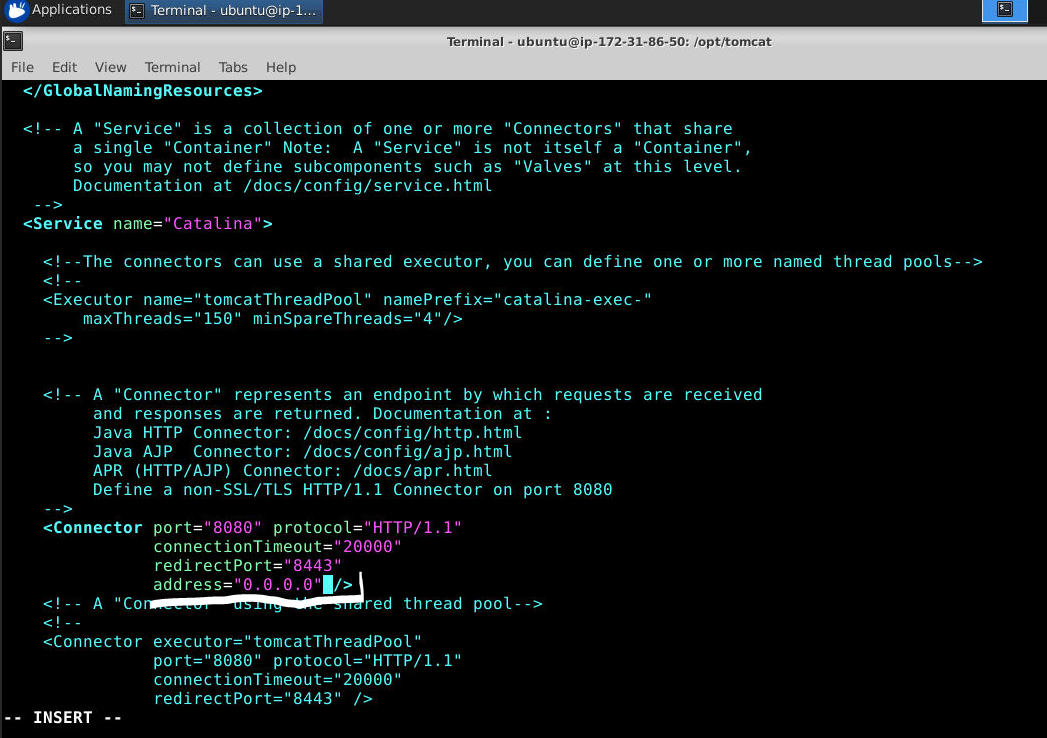
**Restart=always**

**[Install]**

**WantedBy=multi-user.target**

****

* Open the **server.xml** file with the command **sudo vi conf/server.xml**
* Add **address="0.0.0.0"** to connector and save the file.



* Open the users file with the following command:

**sudo vi /opt/tomcat/conf/tomcat-users.xml**

* Add the following lines right before the last line and save the file:

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

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

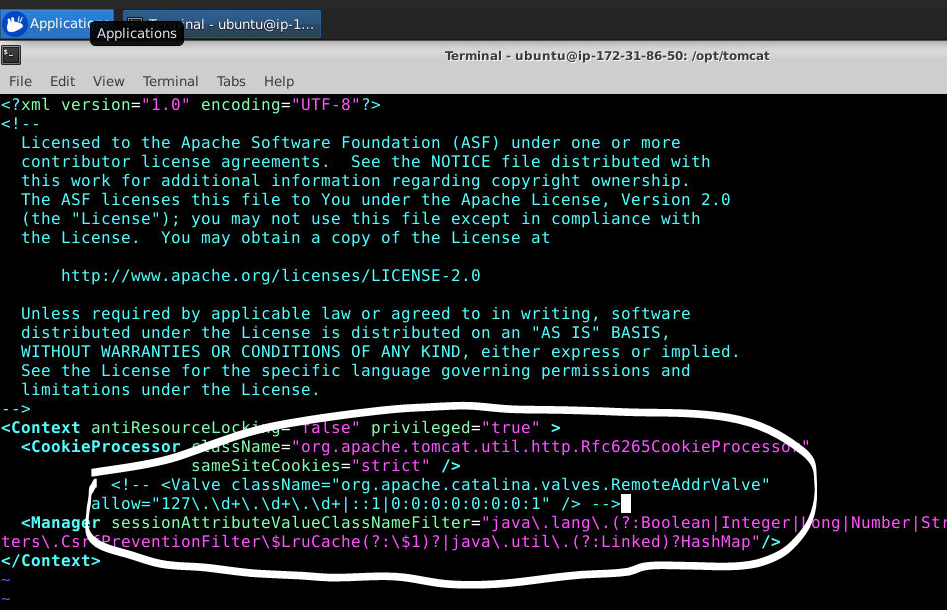
****

* By default, newer versions of Tomcat restrict access to the Manager and Host Manager apps to connections coming from the server itself. Since we are installing on a remote machine, you will need to remove or alter this restriction. To change the IP address restrictions on these, open the appropriate context.xml files with the following commands:

**sudo vi /opt/tomcat/webapps/manager/META-INF/context.xml** and

**sudo vi /opt/tomcat/webapps/host-manager/META-INF/context.xml**

* Comment out the IP address restriction **in both the above mentioned files** to allow connections from anywhere.

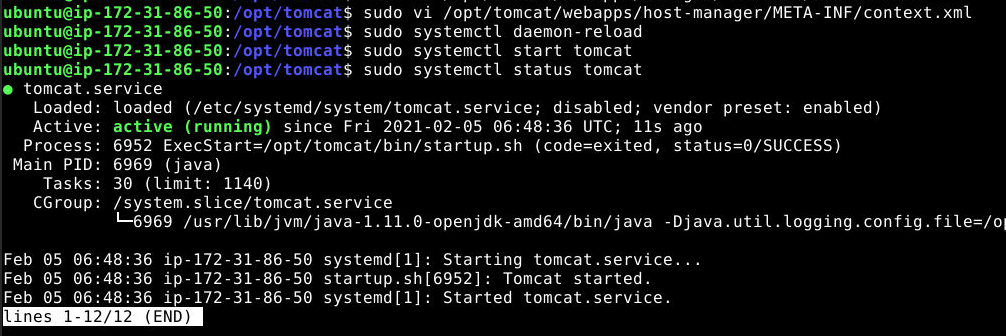
****

* Start and verify the Tomcat server with the following commands:

**sudo systemctl daemon-reload**

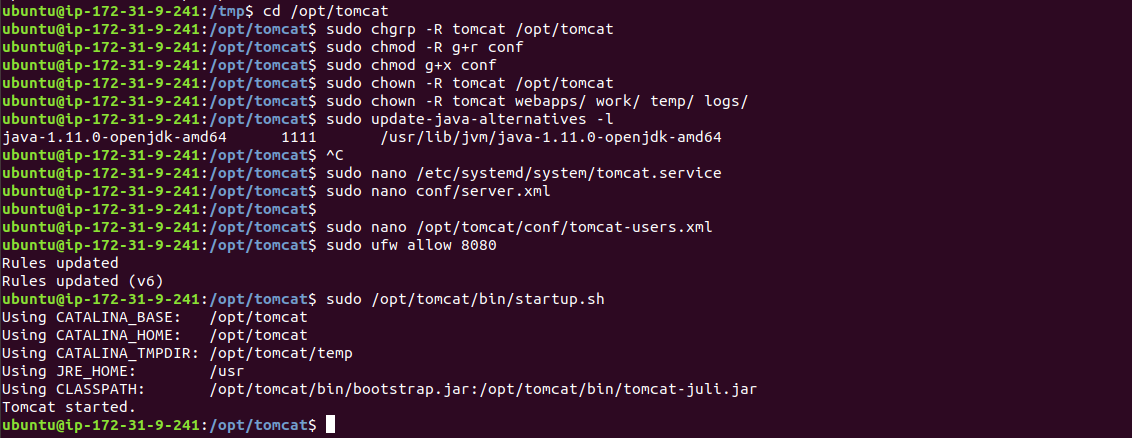
**sudo systemctl start tomcat**

**sudo systemctl status tomcat**

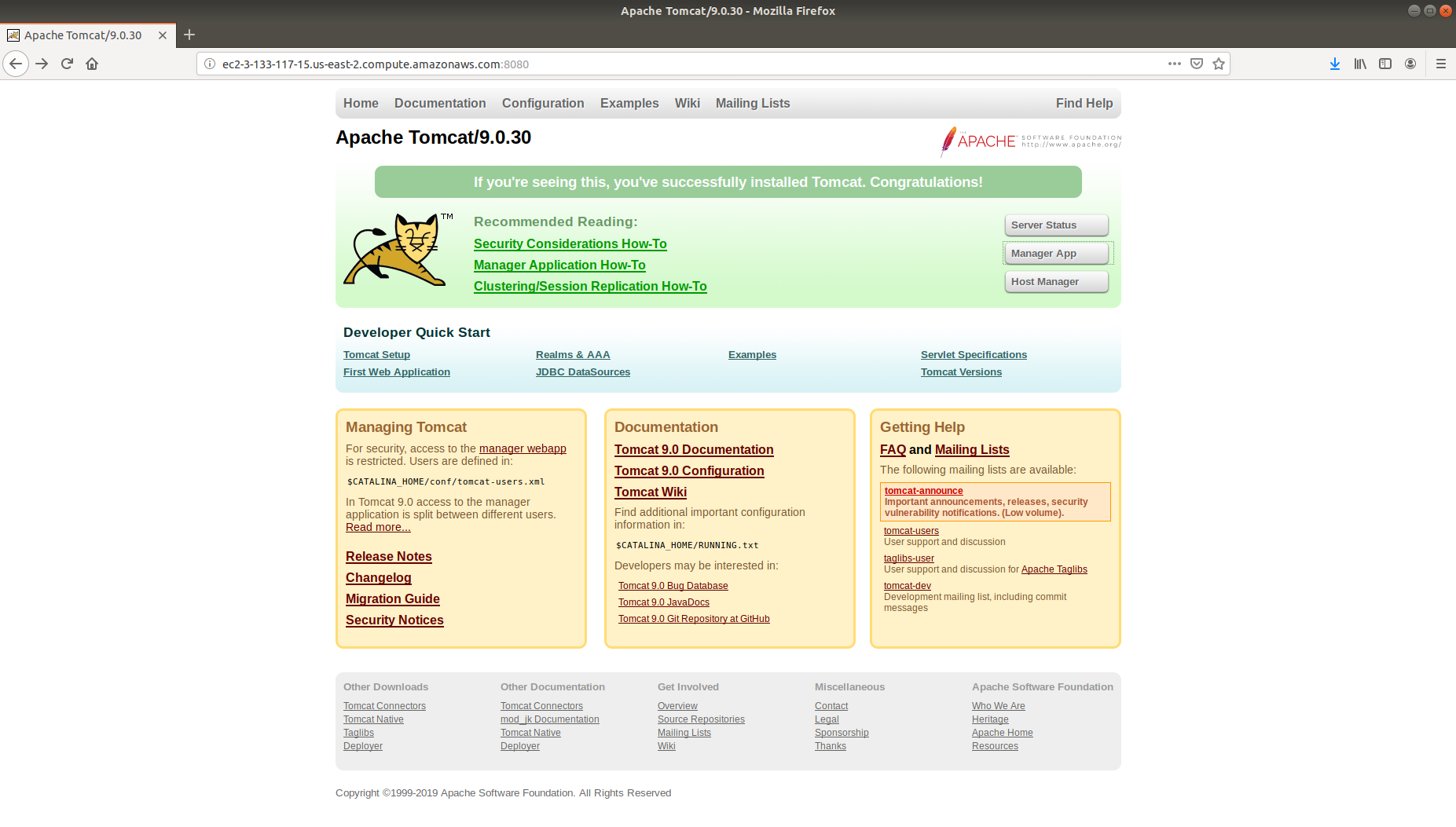


* Allow traffic at 8081 with the following command:

**sudo ufw allow 8081**

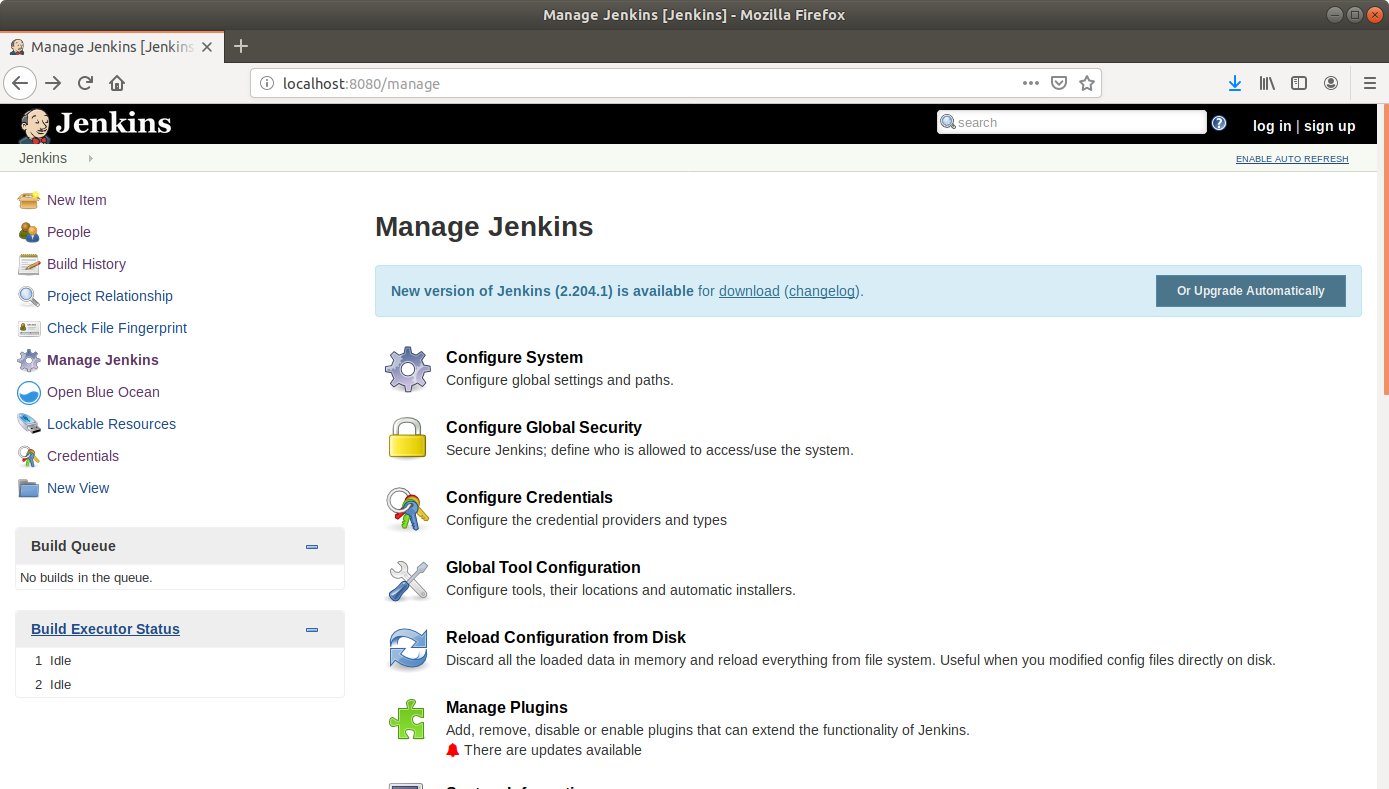
****

* Navigate to **http://<Public DNS (IPv4)>:8080** to view the tomcat server.

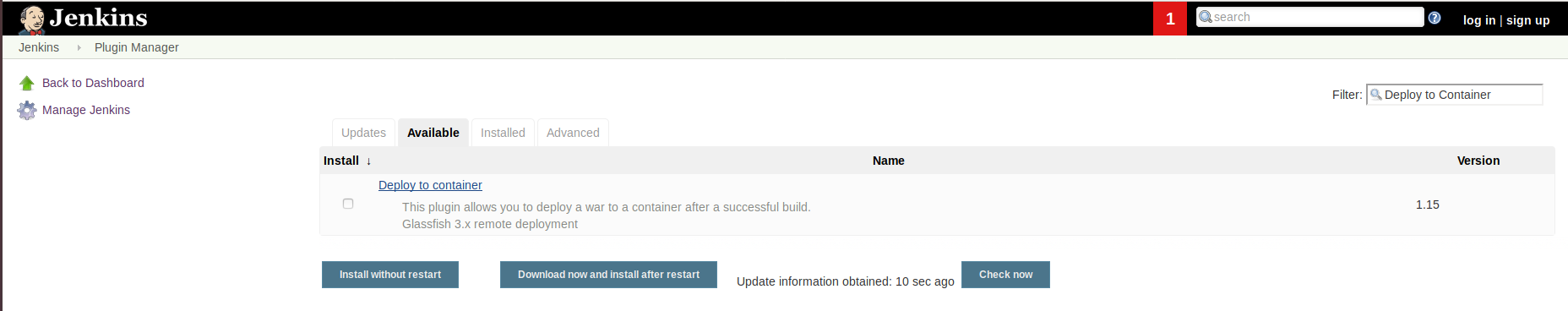
****

**Step 6:** Creating a deployment pipeline in Jenkins

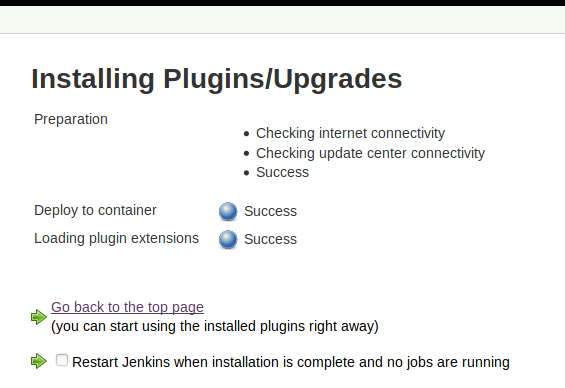
* Go to Jenkins dashboard.
* Click on *Manage Jenkins* and select *Manage Plugins.*

**

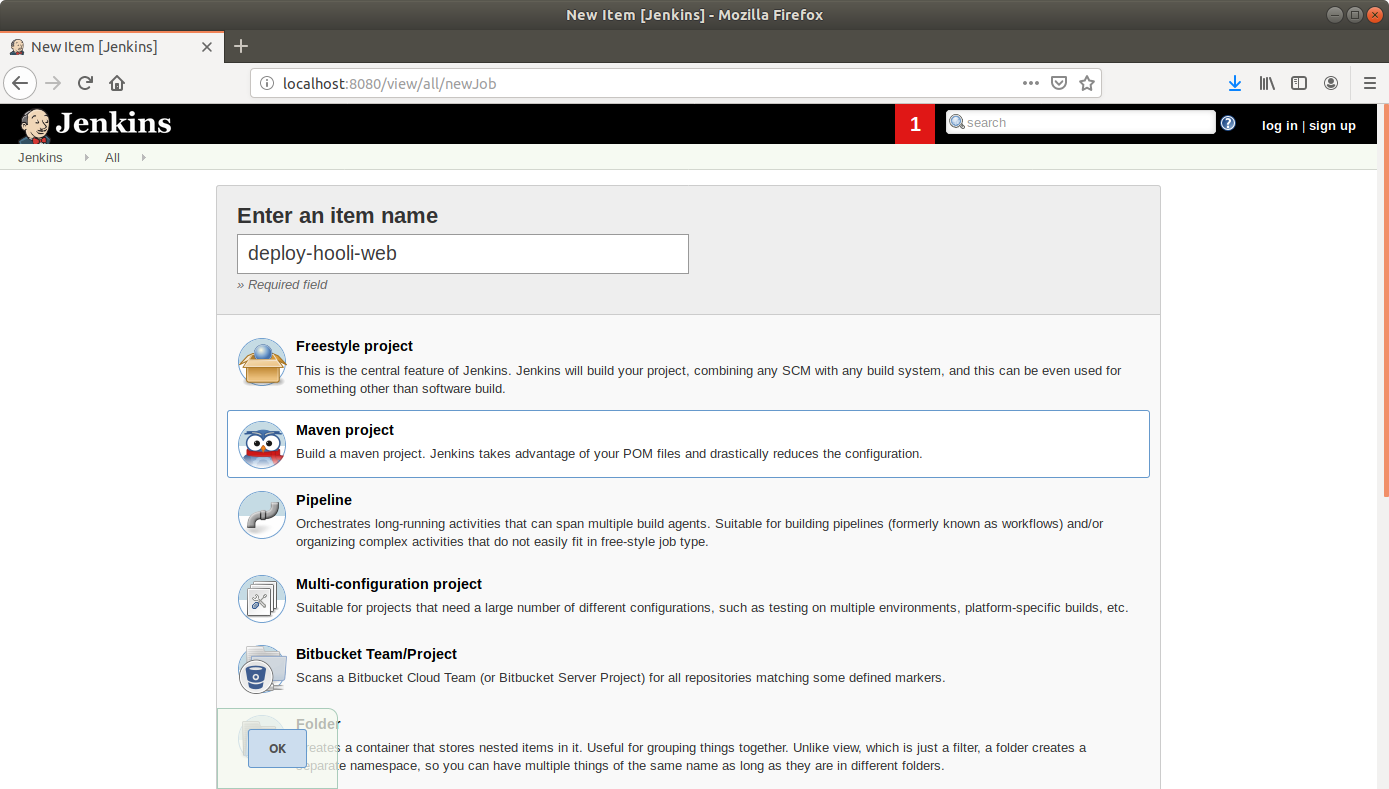
* From the available plugins, install ***Deploy to container****.*

**

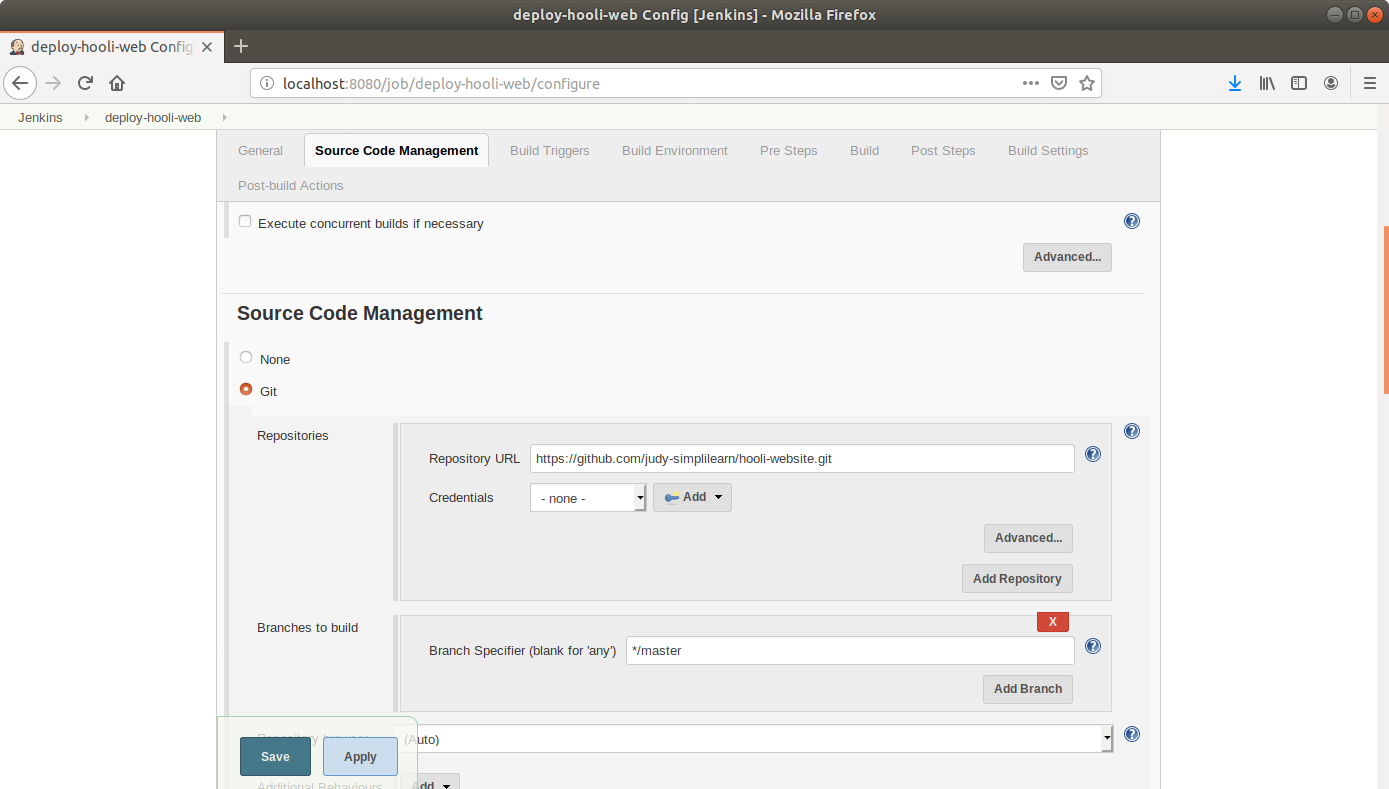
* Click on ***Install without restart****.*

**

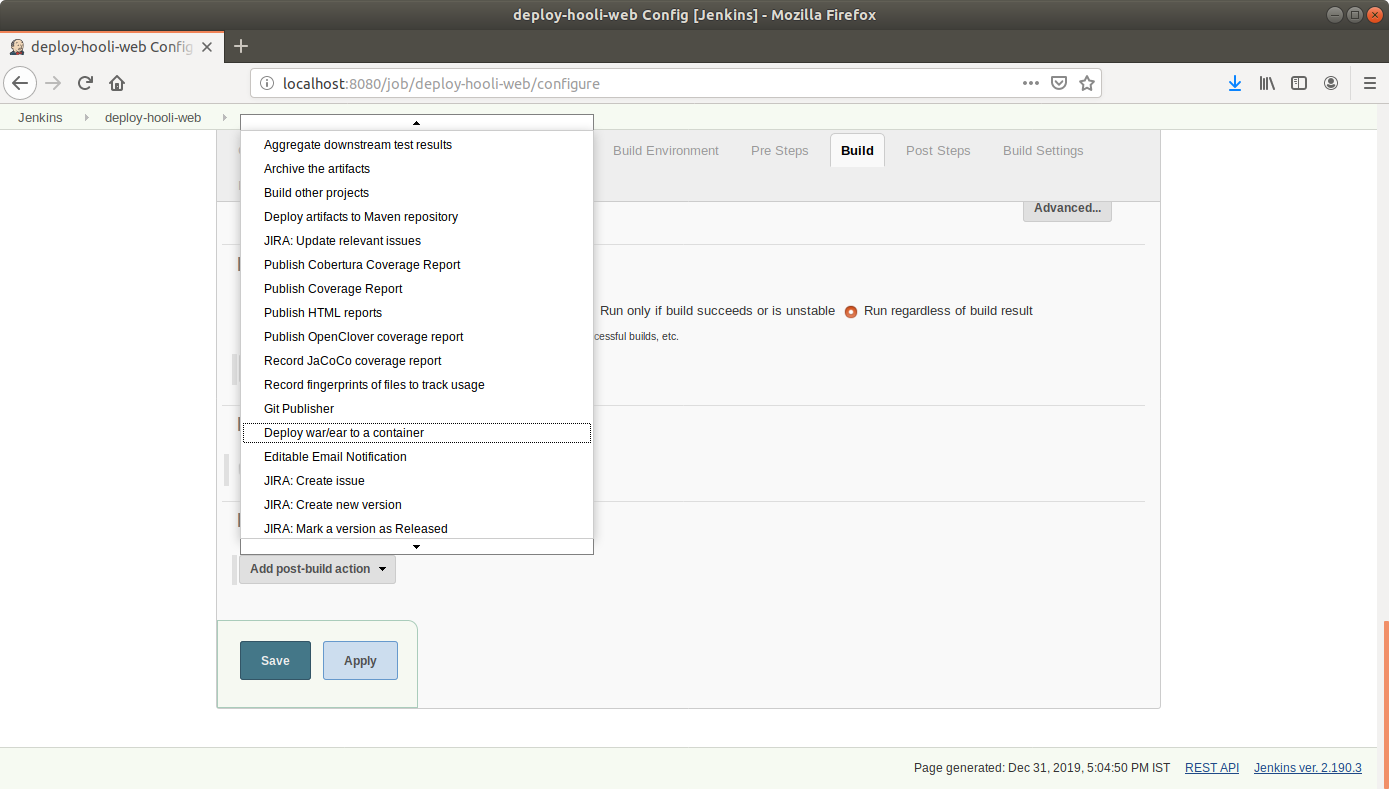
* Click on ***New Item***.
* Enter a **name** for your build job.
* Select ***Maven Project***as the build job type.



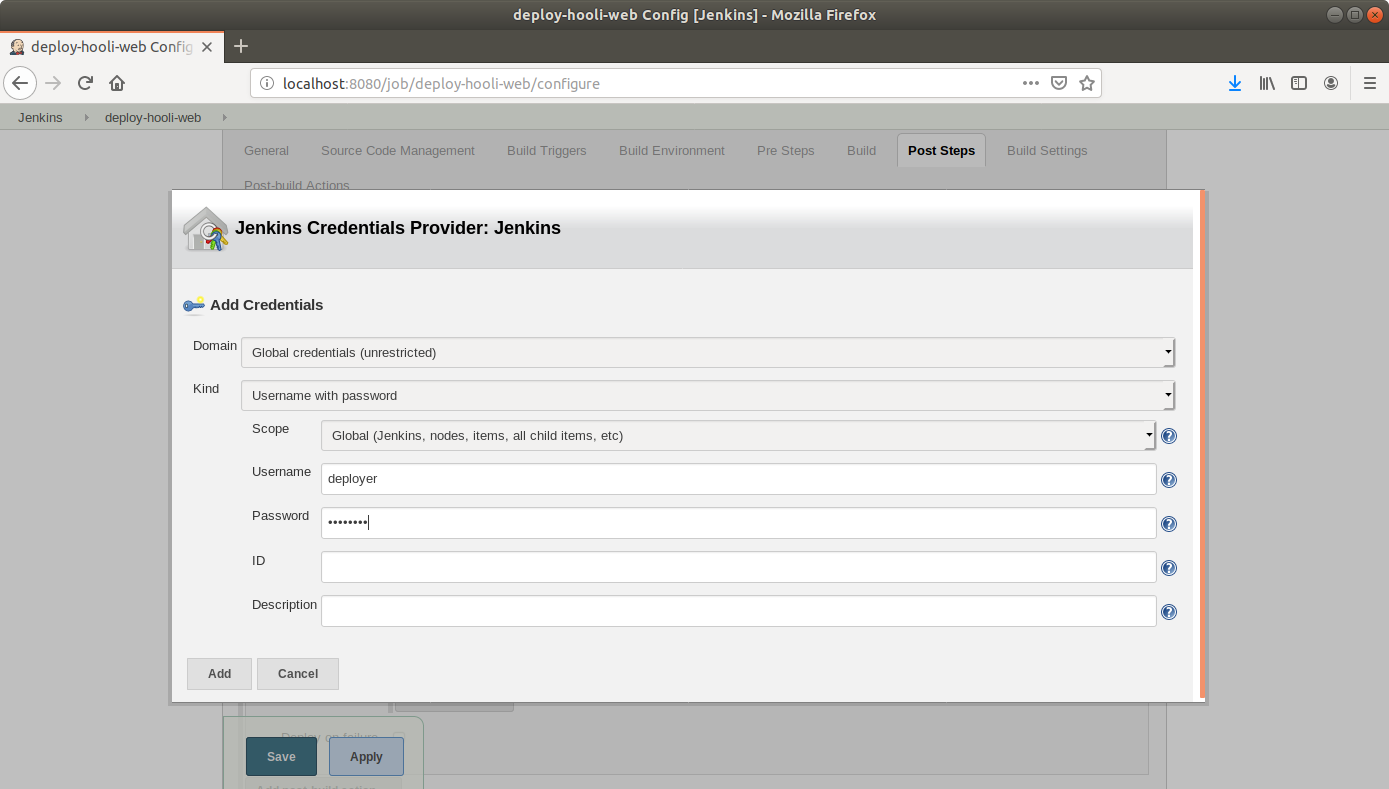
* Click OK.
* On the configuration page, scroll down to the Source Code Management section.
* Select *Git in SCM.*
* Add the repository URL
* Also clearly mention the branch to build (master or main) depending on your repository structure.



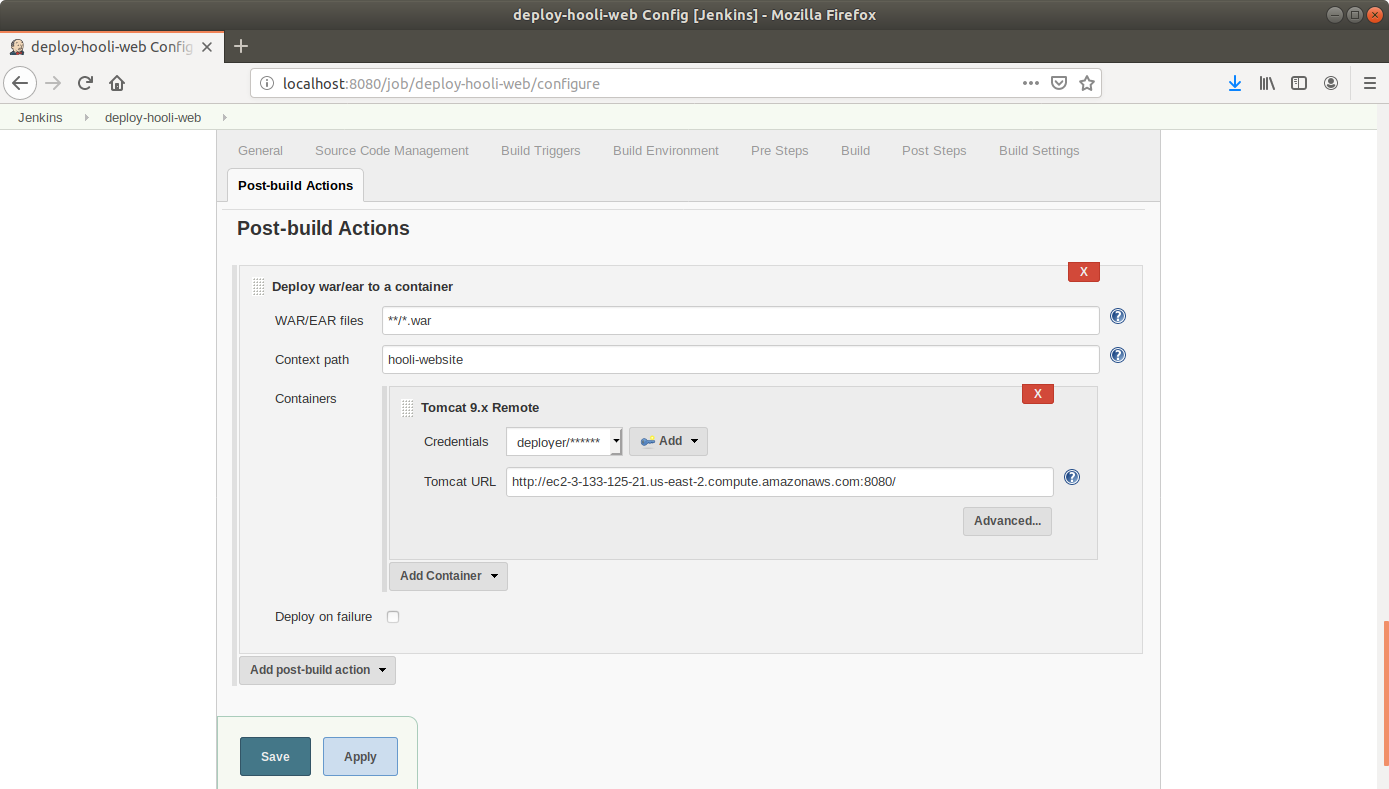
* Drag to the bottom and go to the ***Post-build Actions*** section.
* Click on ***Add post-build*** *action* button.
* On the available options click on the ***Deploy war/ear to container***.



* Add the credentials.



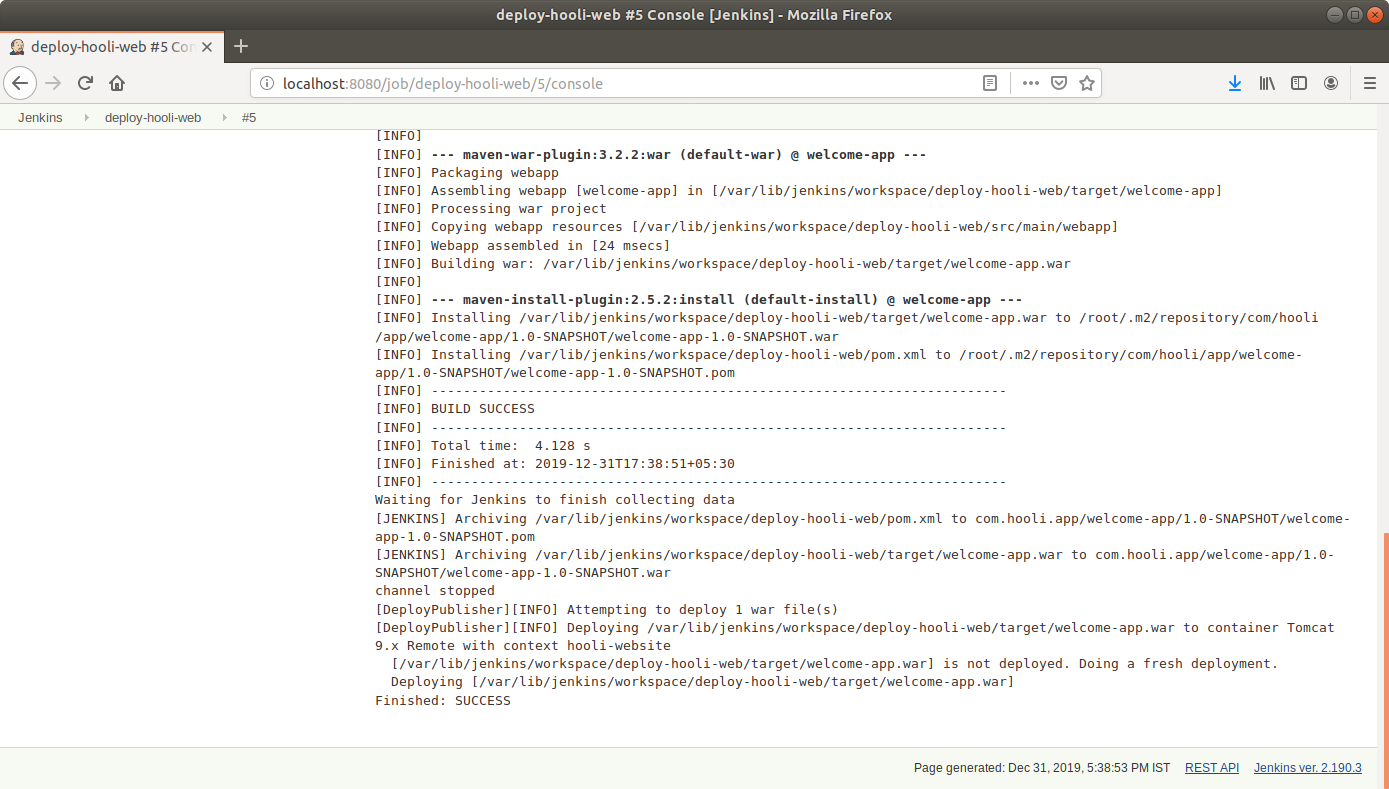
* Fill the required parameters for the plugin. Use the following screenshot as a reference:



* Choose the Context Path in which the application should be installed. It would rename the WAR file before deploying to the server and thereby the application context root would be changed.
* Click Save.

**Step 7:** Running a deployment pipeline in Jenkins

* Click on *Build Now* in the project window.
* Jenkins will now build your pipeline and output the logs.



* Navigate to the URL on your browser to view your webapp.