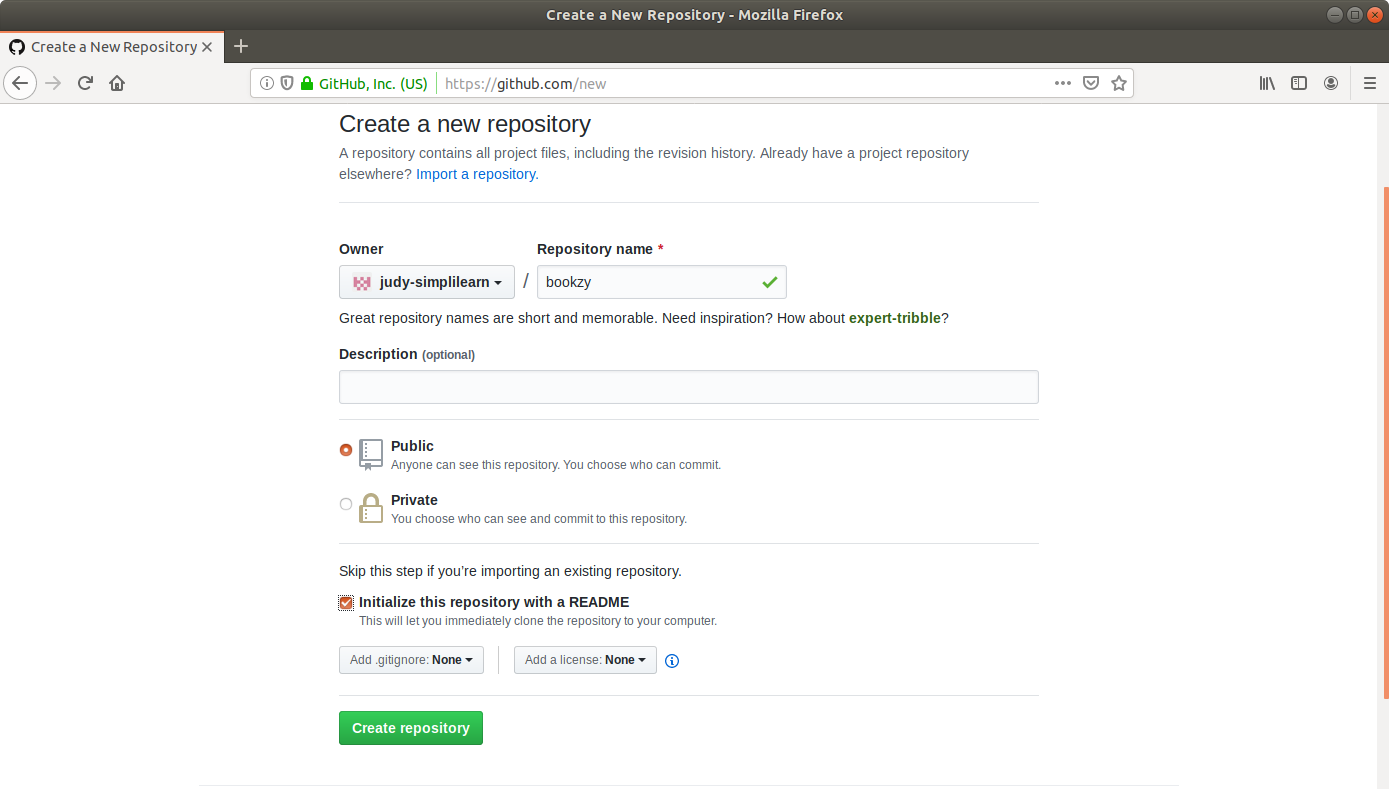
Building a CI/CD Pipeline with Jenkins

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

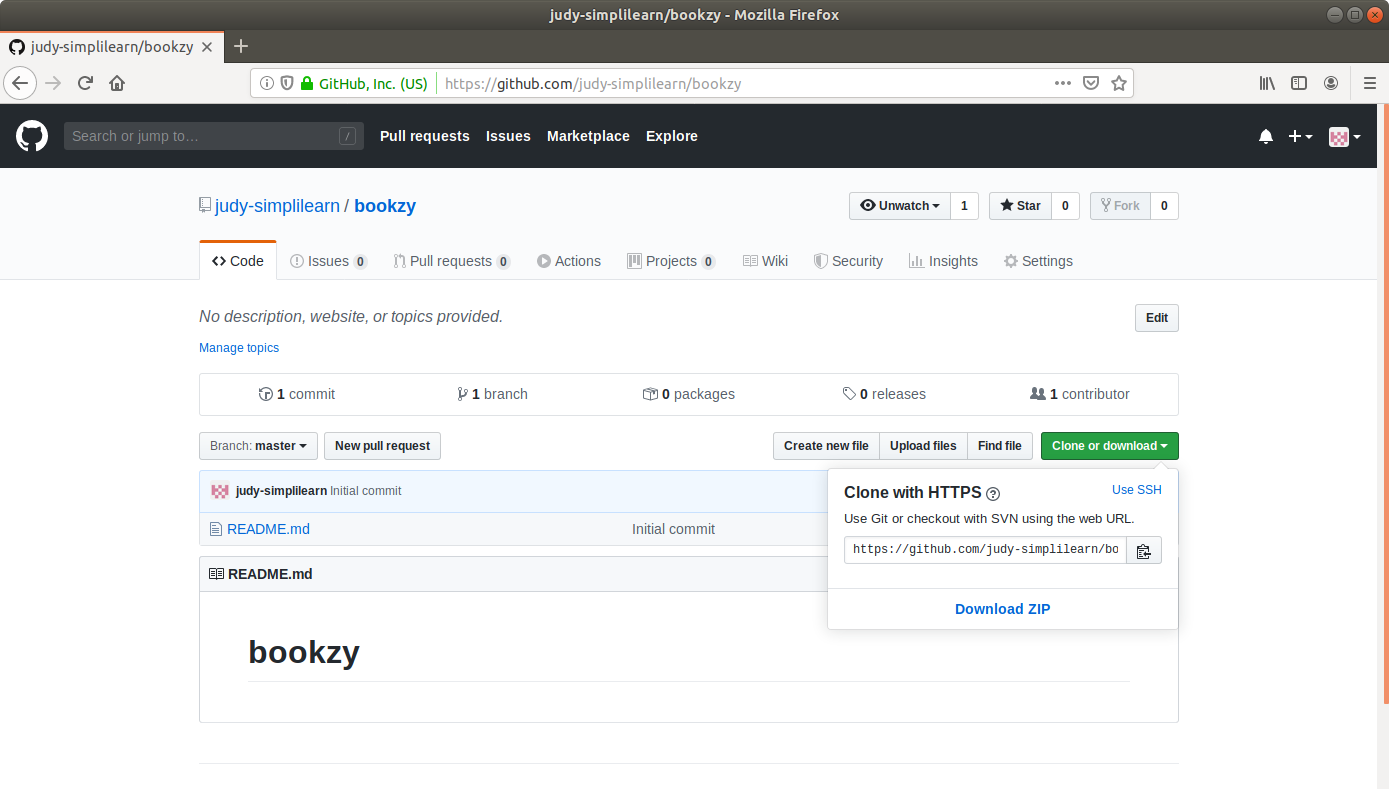
* Log in to your Github account.
* Click on the plus icon next to the profile picture and select *New repository* from the drop-down menu.



* Fill the required fields in the create repository form.

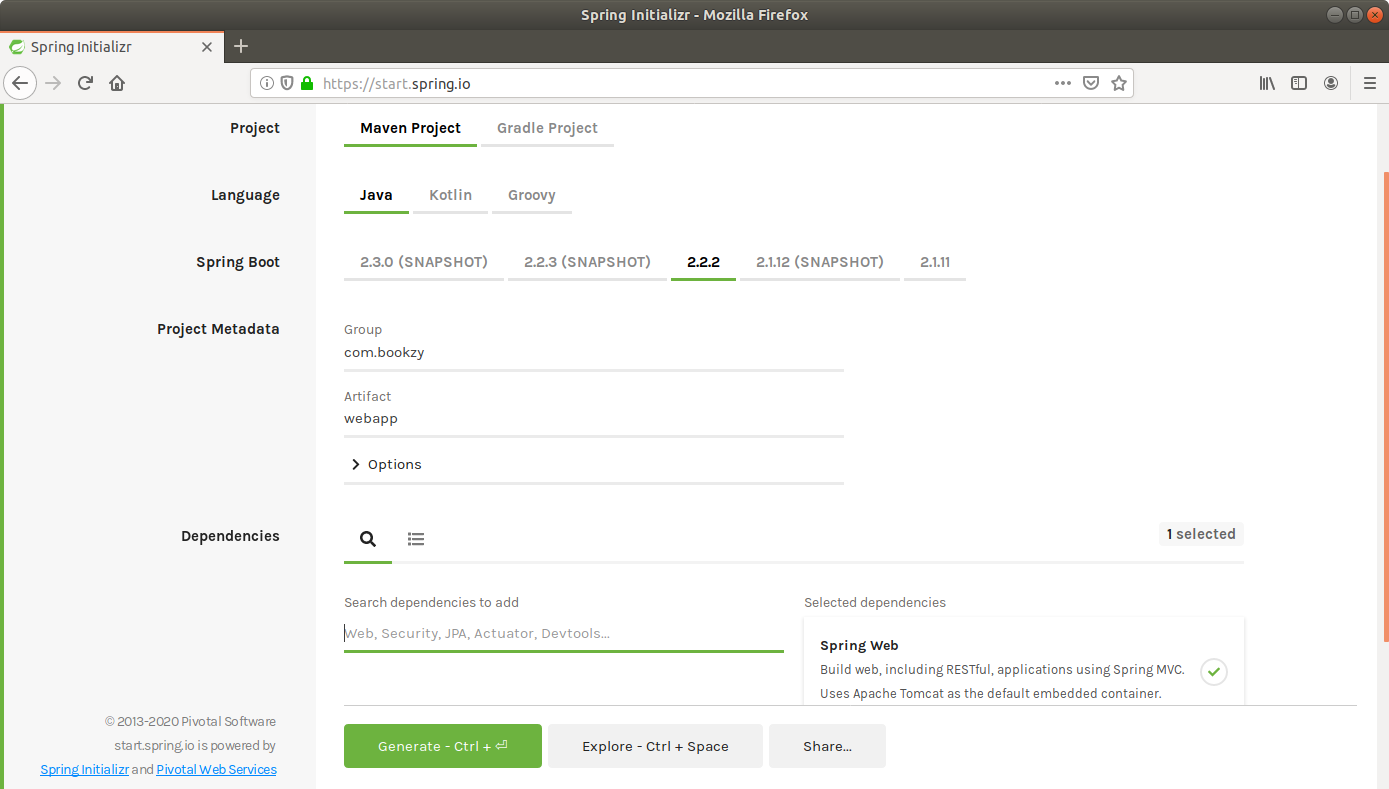


* Click on the **Create Repository** button.
* Click on the **Clone or download** button and copy the URL.



**Step 2:** Generating a spring boot project

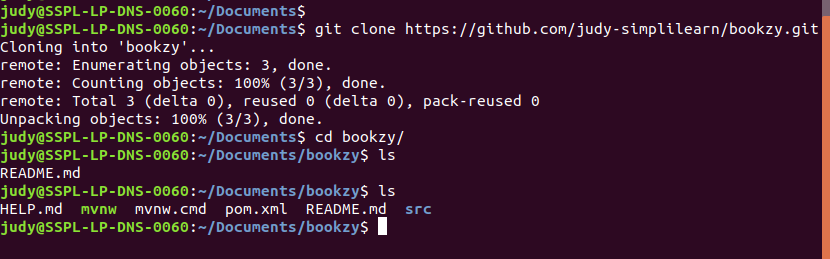
* Go to ​start.​spring.​io/​
* Select Maven as the project type.
* Fill Group and Artifact with appropriate values. For example, *com.Bookzy* and *webapp.*
* Click on **Generate Project.**

****

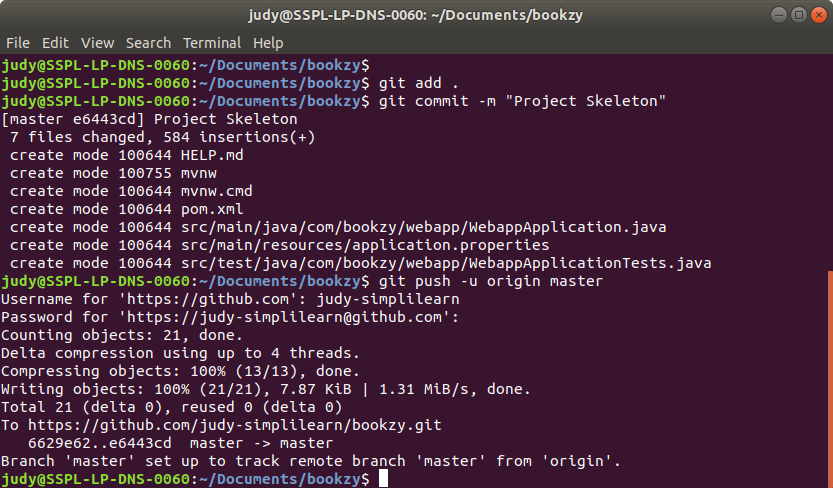
* The generated skeleton project should be downloaded as a zip file.

**Step 3:** Adding the code for the webapp to the repository

* Open the terminal and navigate to an appropriate location.
* Run **git clone [URL]** to clone the repository.
* Unzip the downloaded spring boot project to the cloned repository.

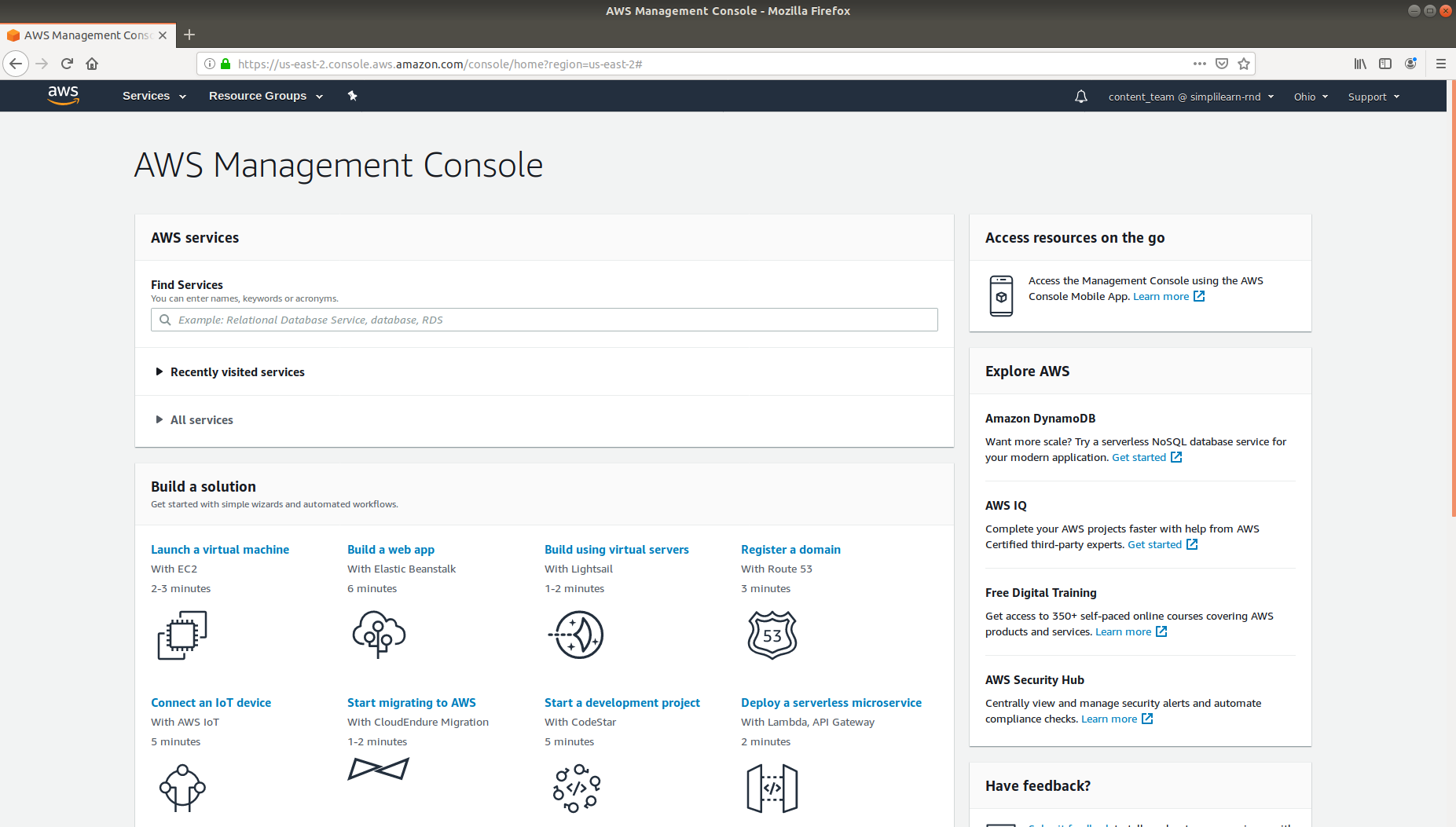


* Commit the changes to the remote SCM.
* Run **git add.**
* Run **git commit -m "Add project skeleton"**
* Run **git push -u origin master.**

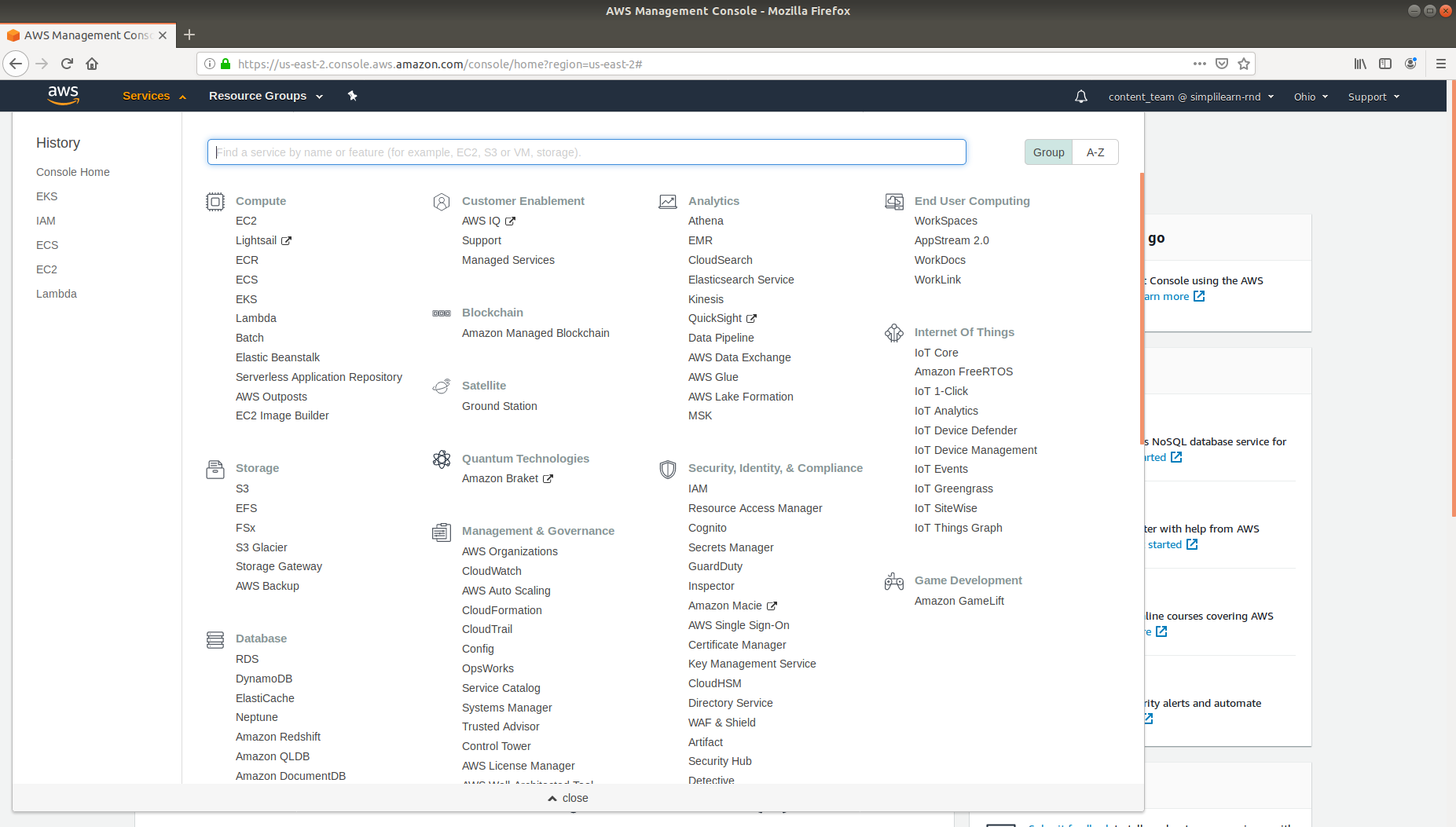
****

**Step 4:** Creating an EC2 instance

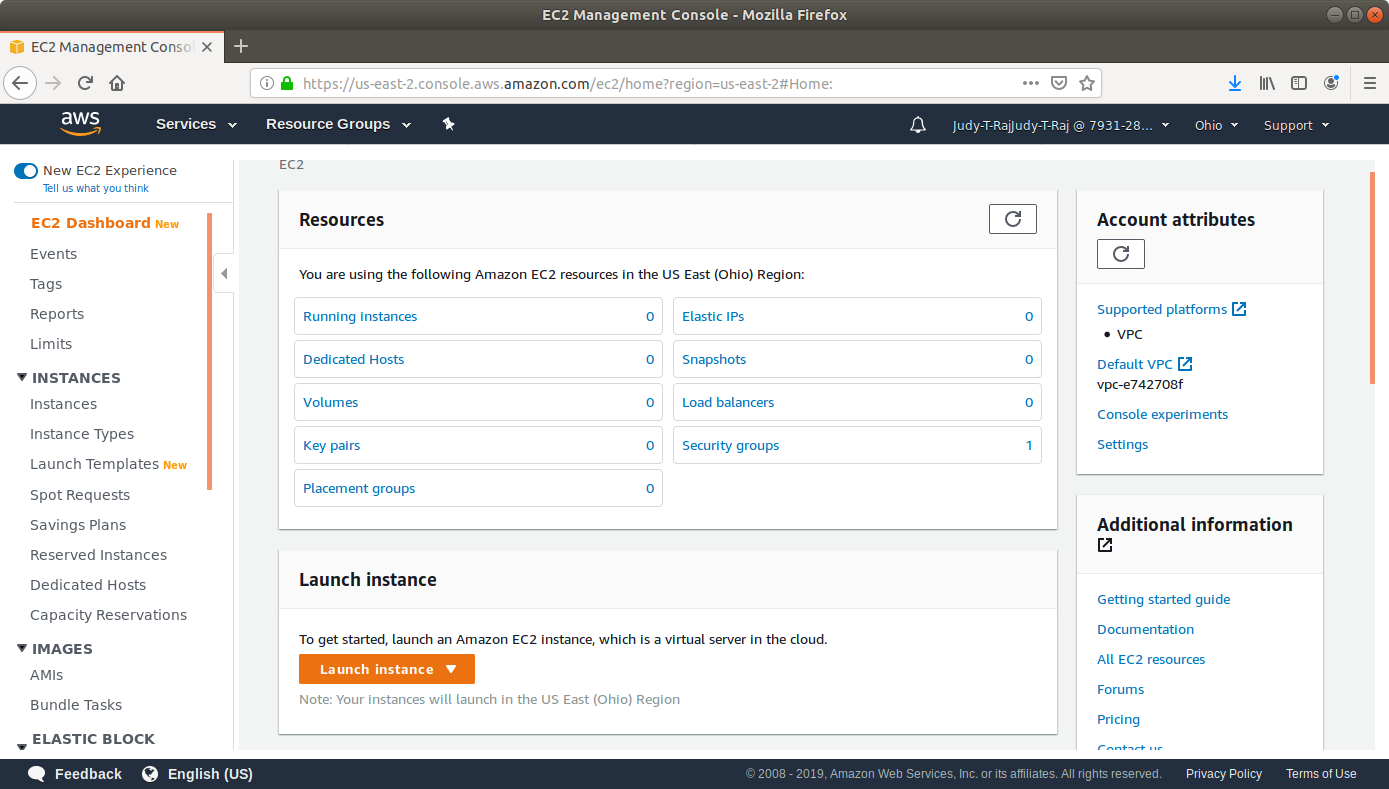
* Log in to the AWS lab account provided. You will then be able to see the following screen:



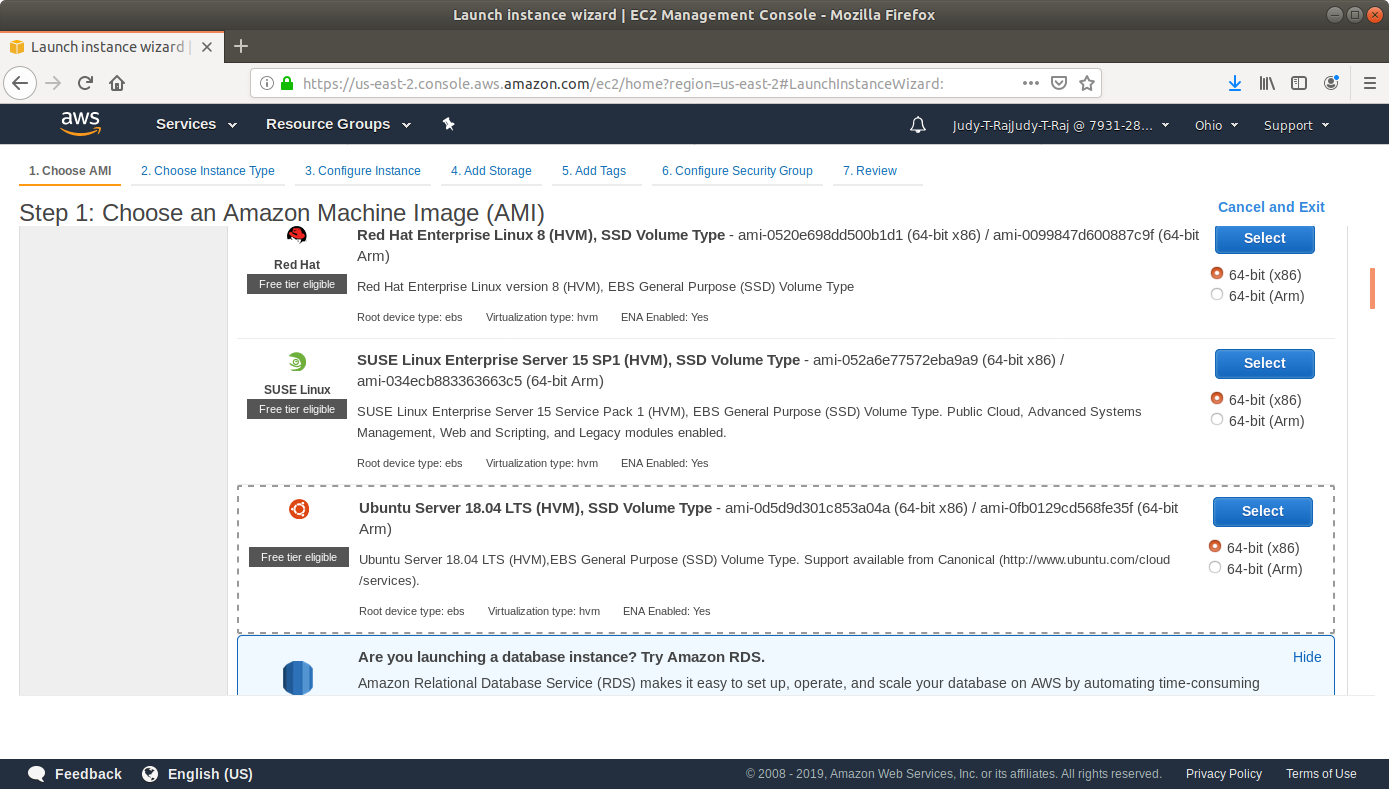
* Click on Services at the top left to view the drop-down list of resources.
* Click on EC2 under the Compute menu from the drop-down list.



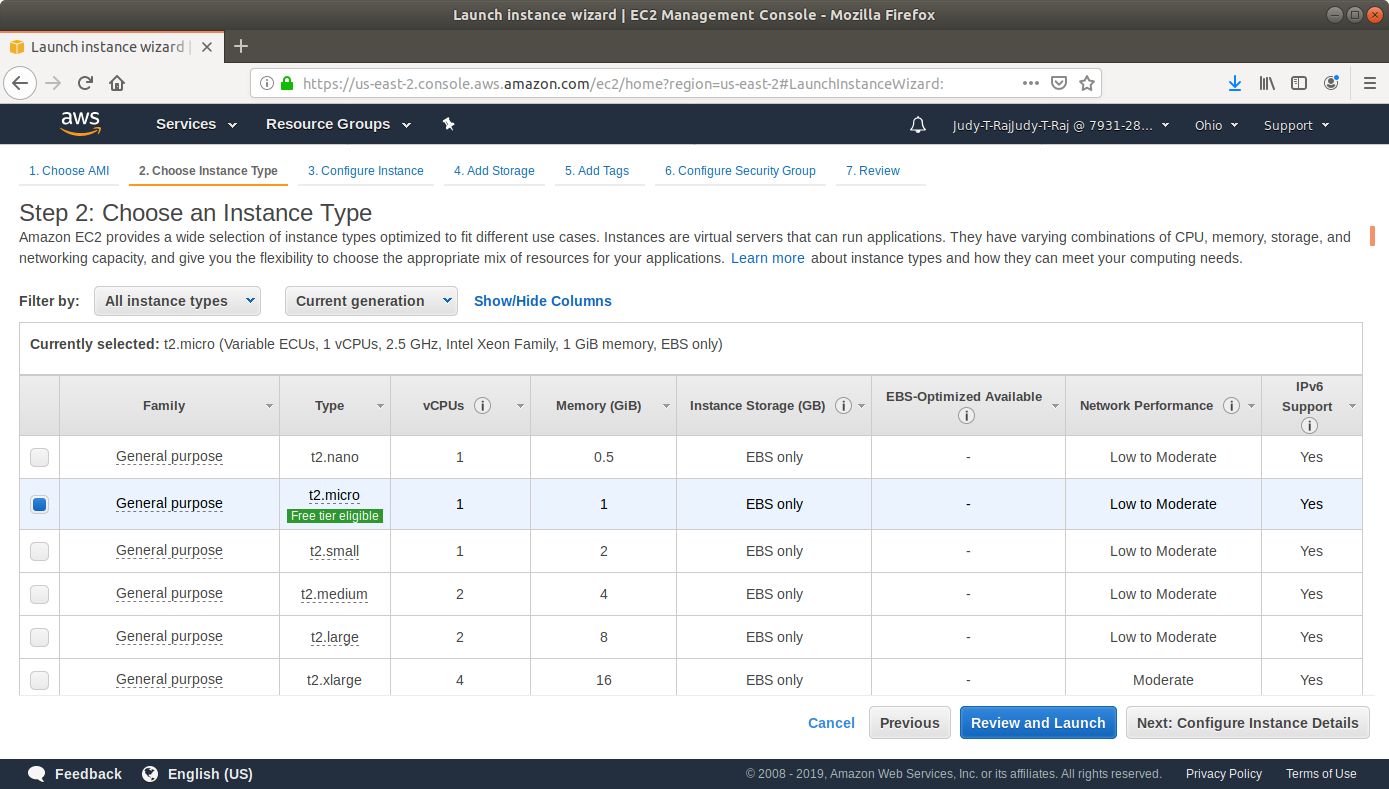
* Click on the **Launch Instance** button and select **Launch Instance** from the menu.



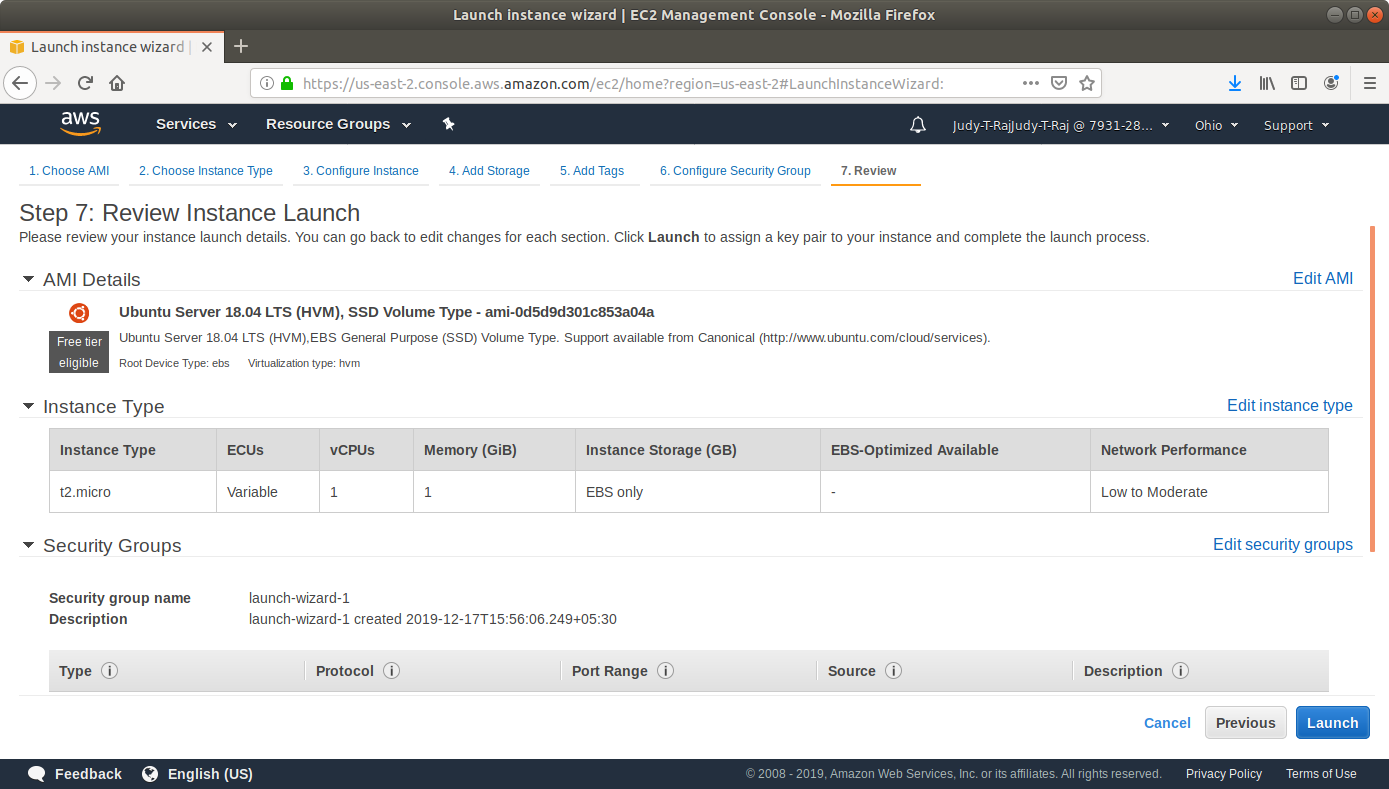
* Choose an Amazon Machine Image (AMI) from the list of AMIs and click on Select.



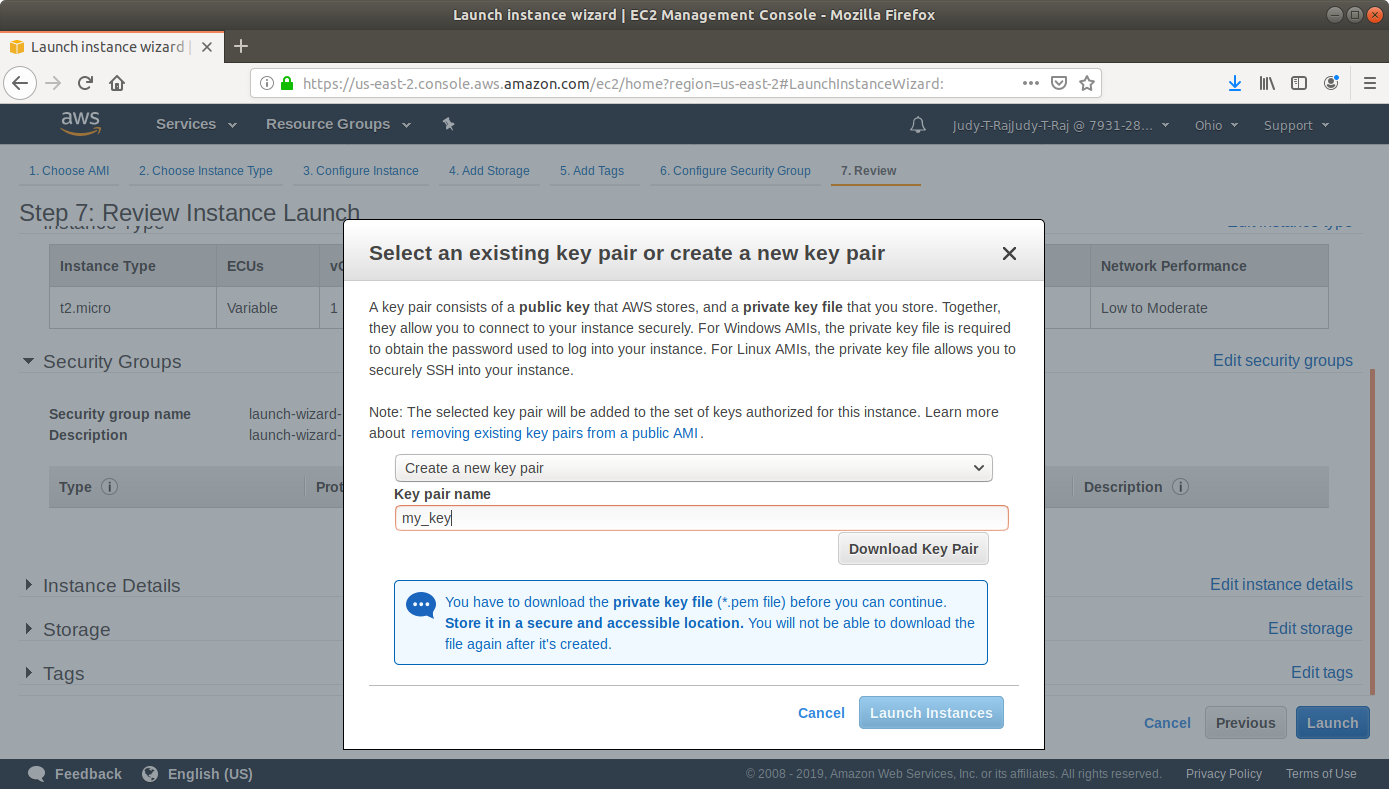
* Choose an Instance Type and click Review and Launch.



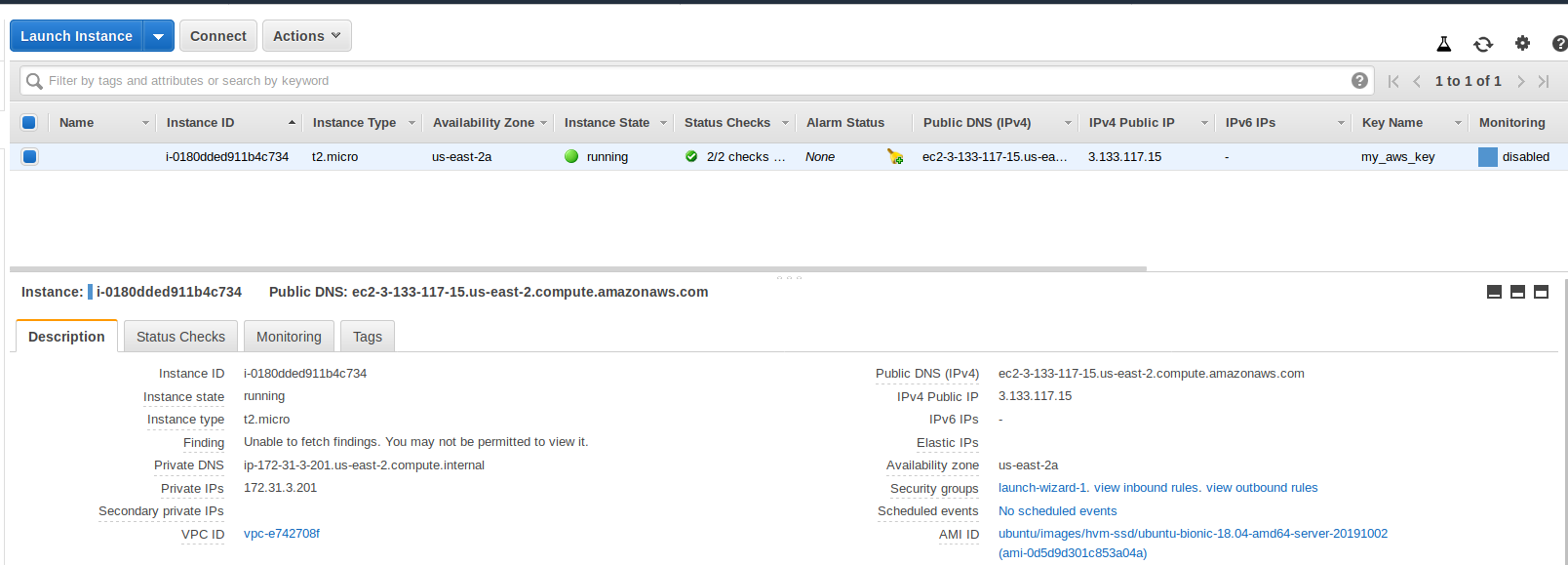
* Click on Launch.



* 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

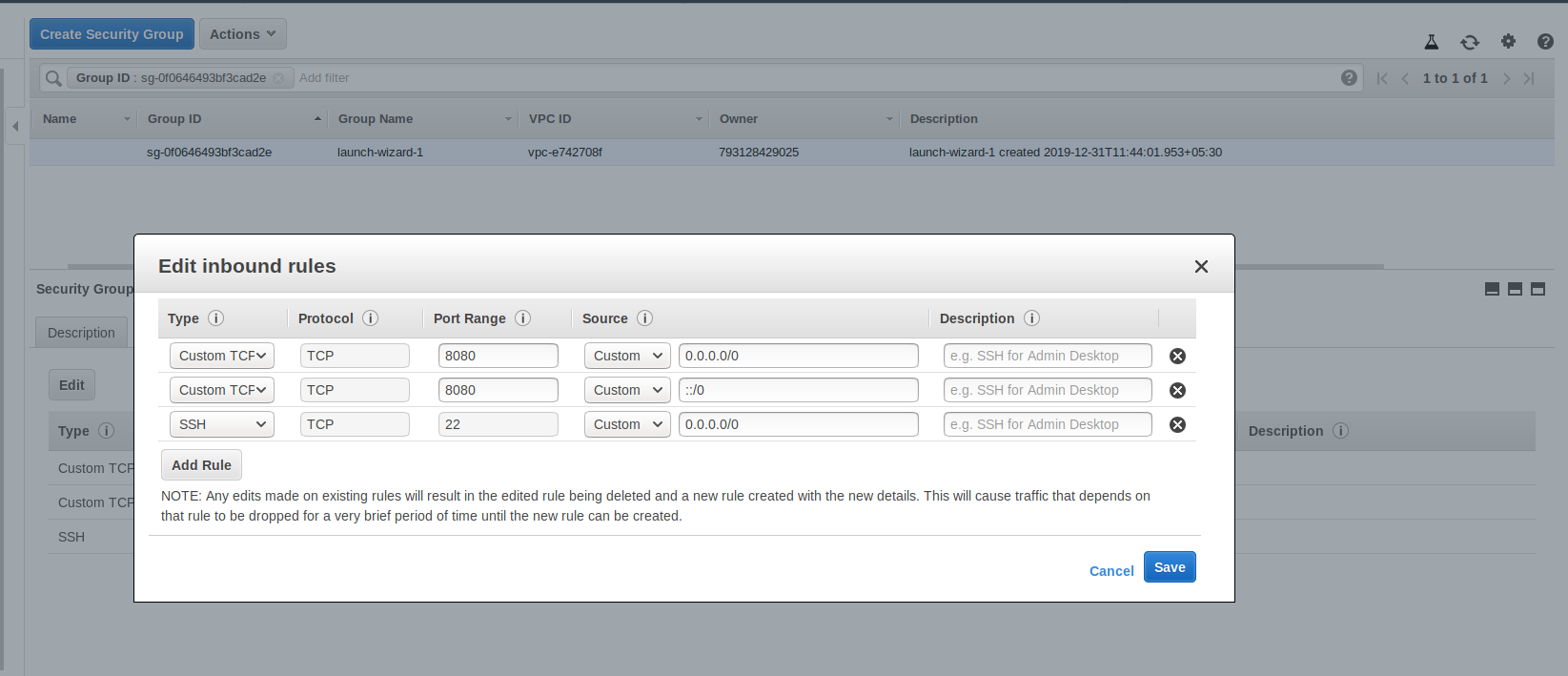
* Add a rule to the security group to which the instance belongs to allow http traffic to port 8081, with the following settings:

**Type:** Custom TCP Rule

**Protocol:** TCP

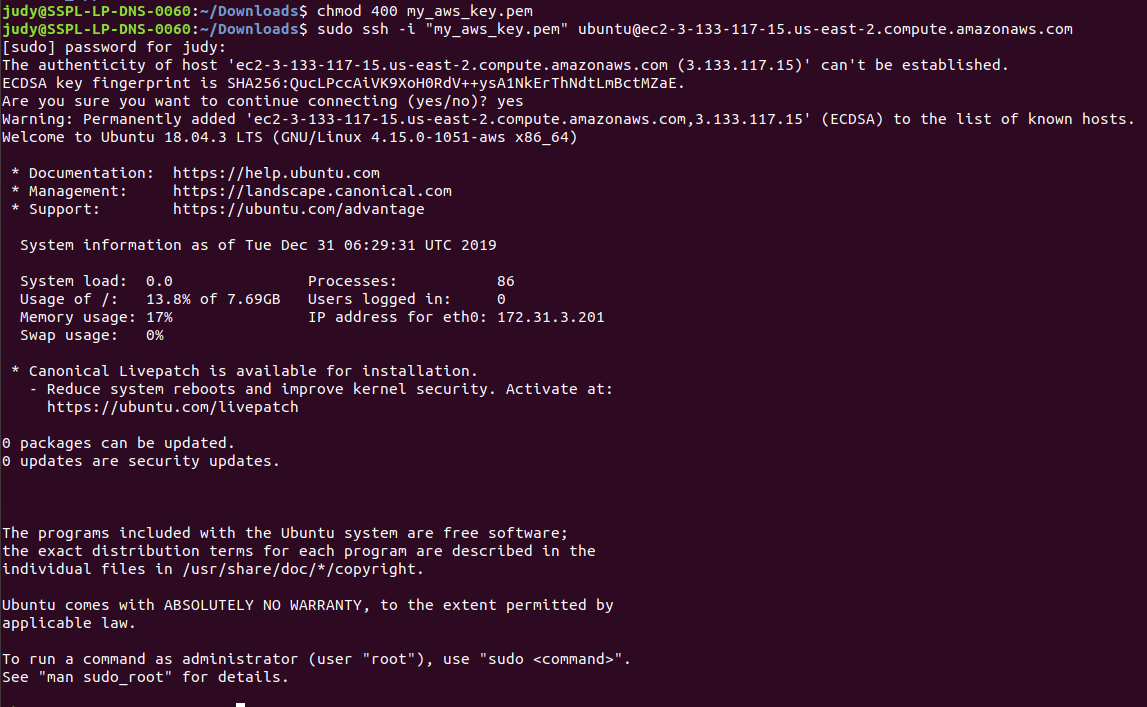
**Port Range:** 8081

**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/v9.0.30/bin/apache-tomcat-9.0.30.tar.gz**

**sudo mkdir /opt/tomcat**

**sudo tar xzvf apache-tomcat-9\*tar.gz -C /opt/tomcat --strip-components=1**

**cd /opt/tomcat**

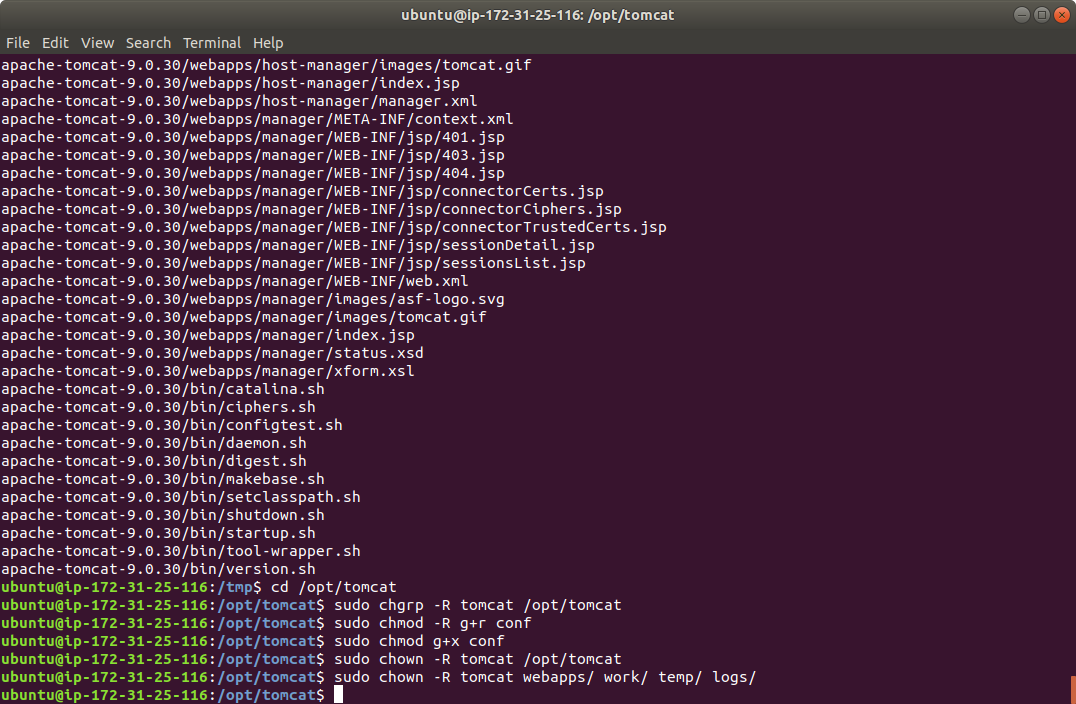
**sudo chgrp -R tomcat /opt/tomcat**

**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 nano /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/jre**

**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**

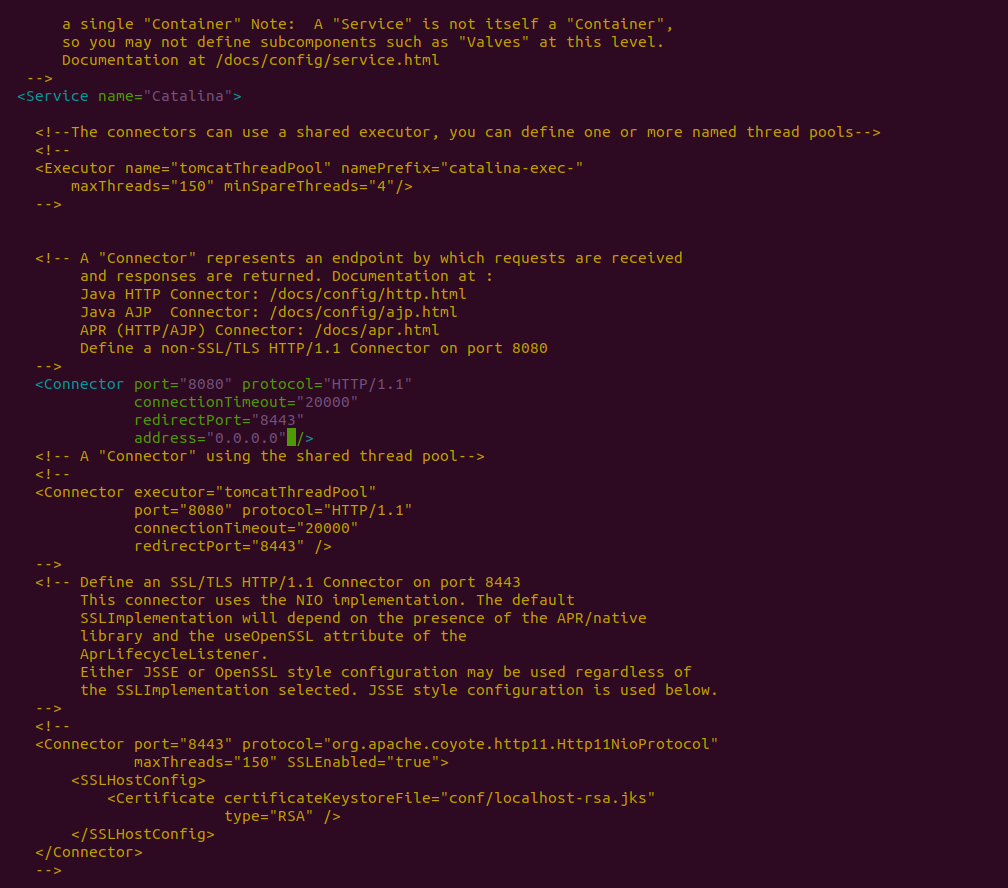
**Restart=always**

**[Install]**

**WantedBy=multi-user.target**



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



* Open the user’s file with the following command: **sudo nano /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"/>**

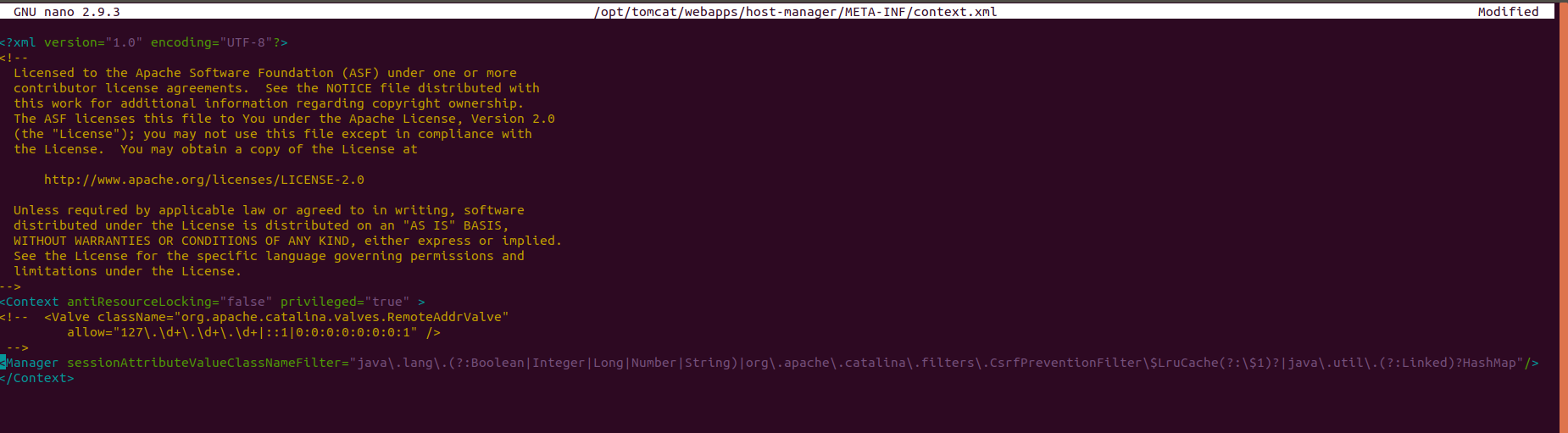
****

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

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

* Comment out the IP address restriction to allow connections from anywhere.



\

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

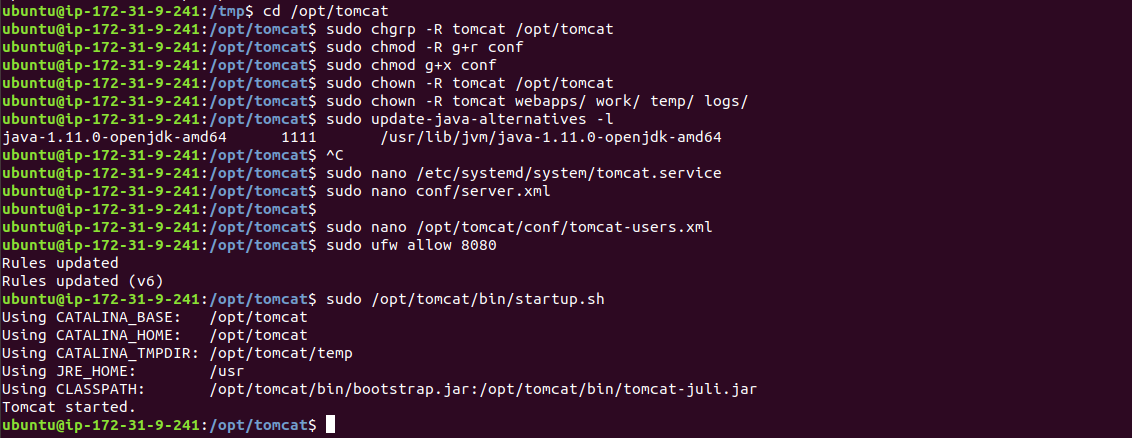
**sudo systemctl daemon-reload**

**sudo /opt/tomcat/bin/startup.sh**

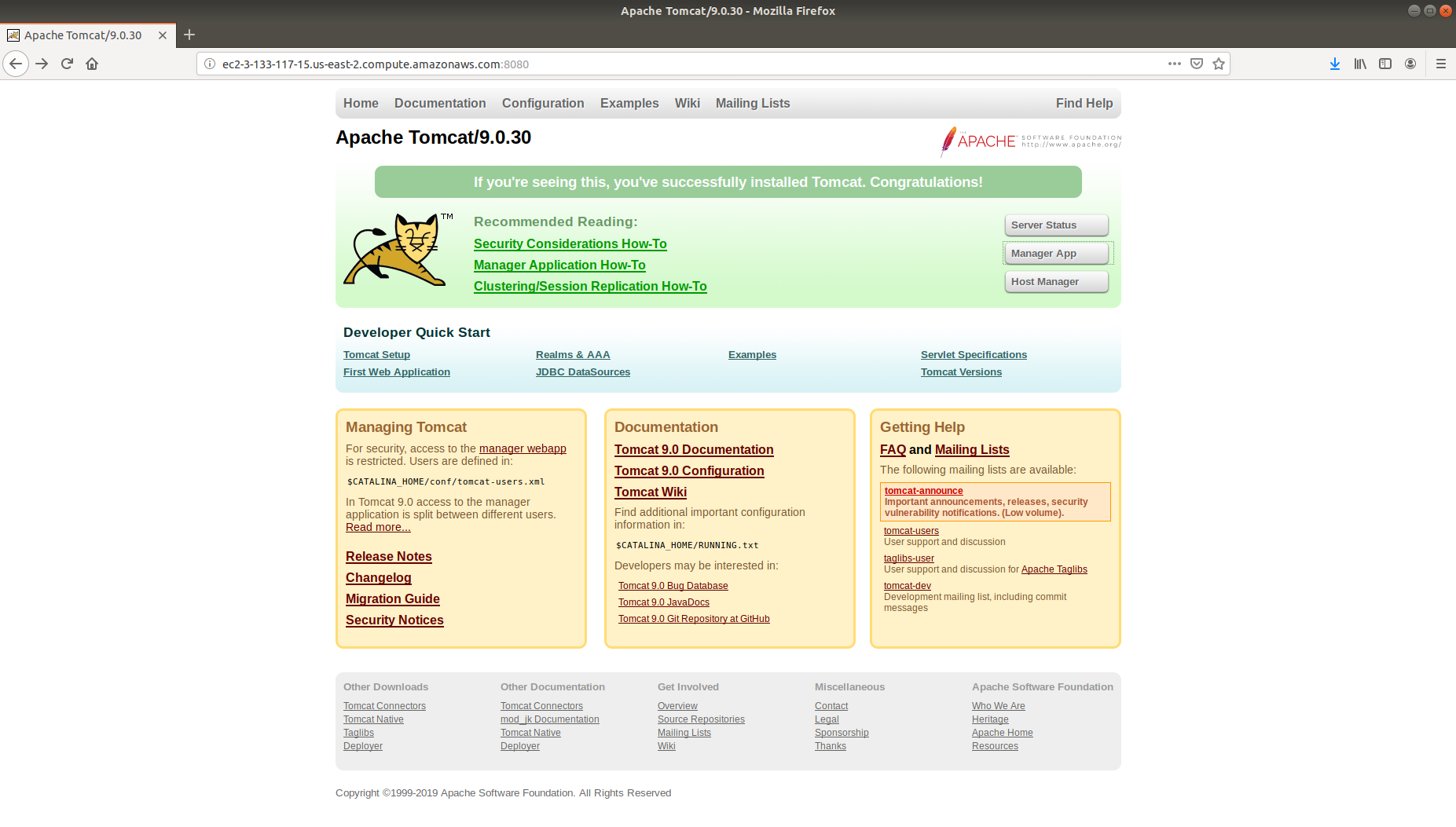
**sudo systemctl status tomcat**

* Allow traffic at 8081 with the following command:

**sudo ufw allow 8081**

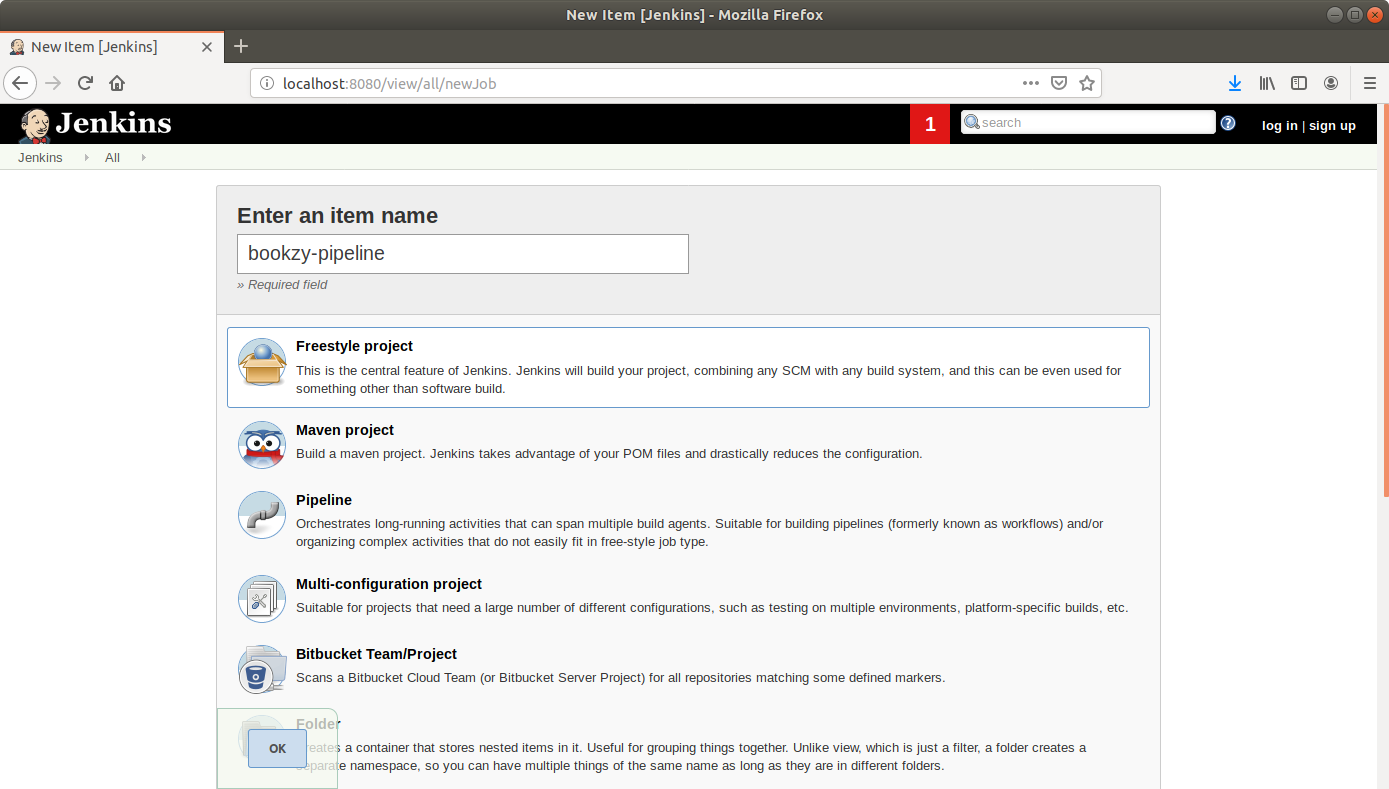
****

* Navigate to http://<Public DNS (IPv4)>:8081 to view the tomcat server.

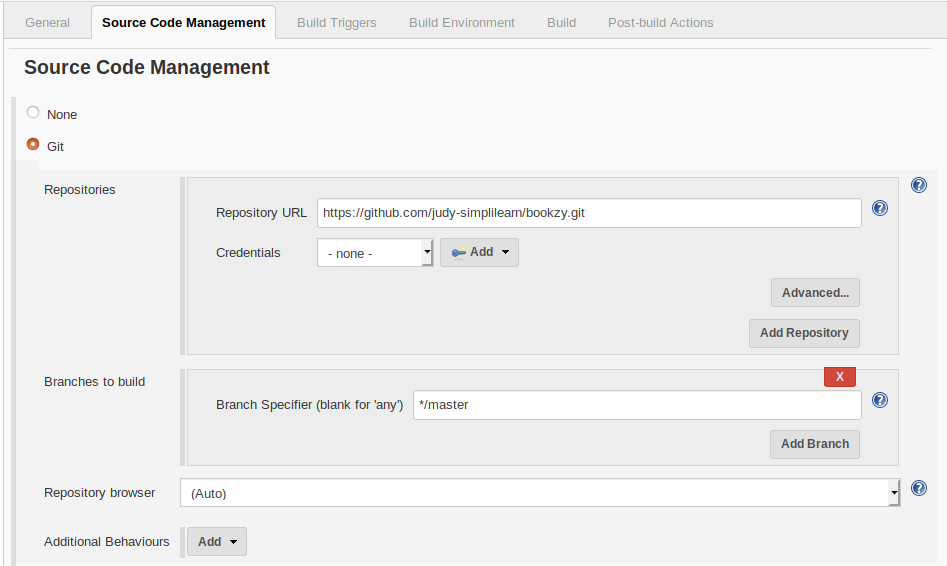
****

**Step 6:** Creating a freestyle job for compilation in Jenkins

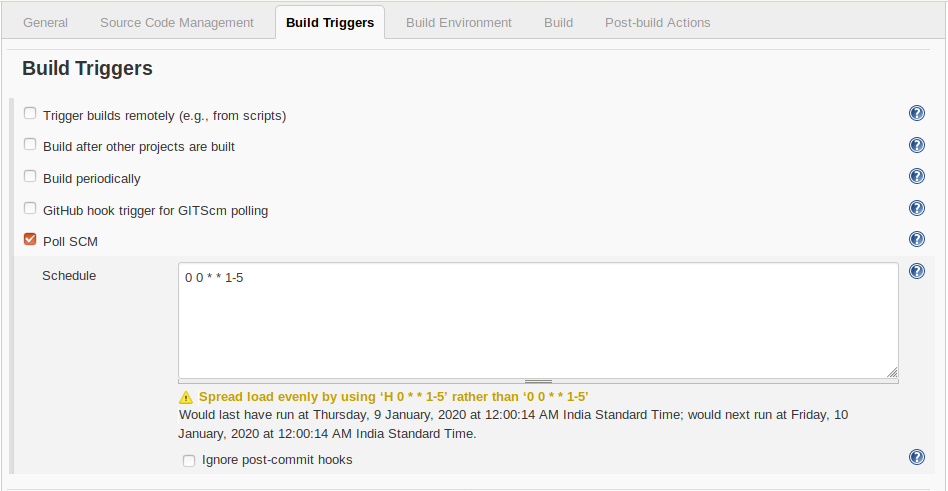
* Go to the Jenkins dashboard.
* Click on *New Item*.
* Enter a name for your build job.
* Select *Freestyle project* as the build job type.



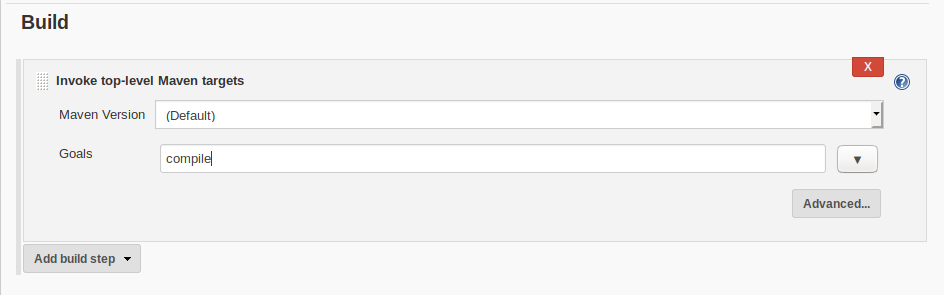
* Click OK.
* On the configuration page, scroll down to the Source Code Management section.
* Select *Git in SCM.*
* Add the repository URL.



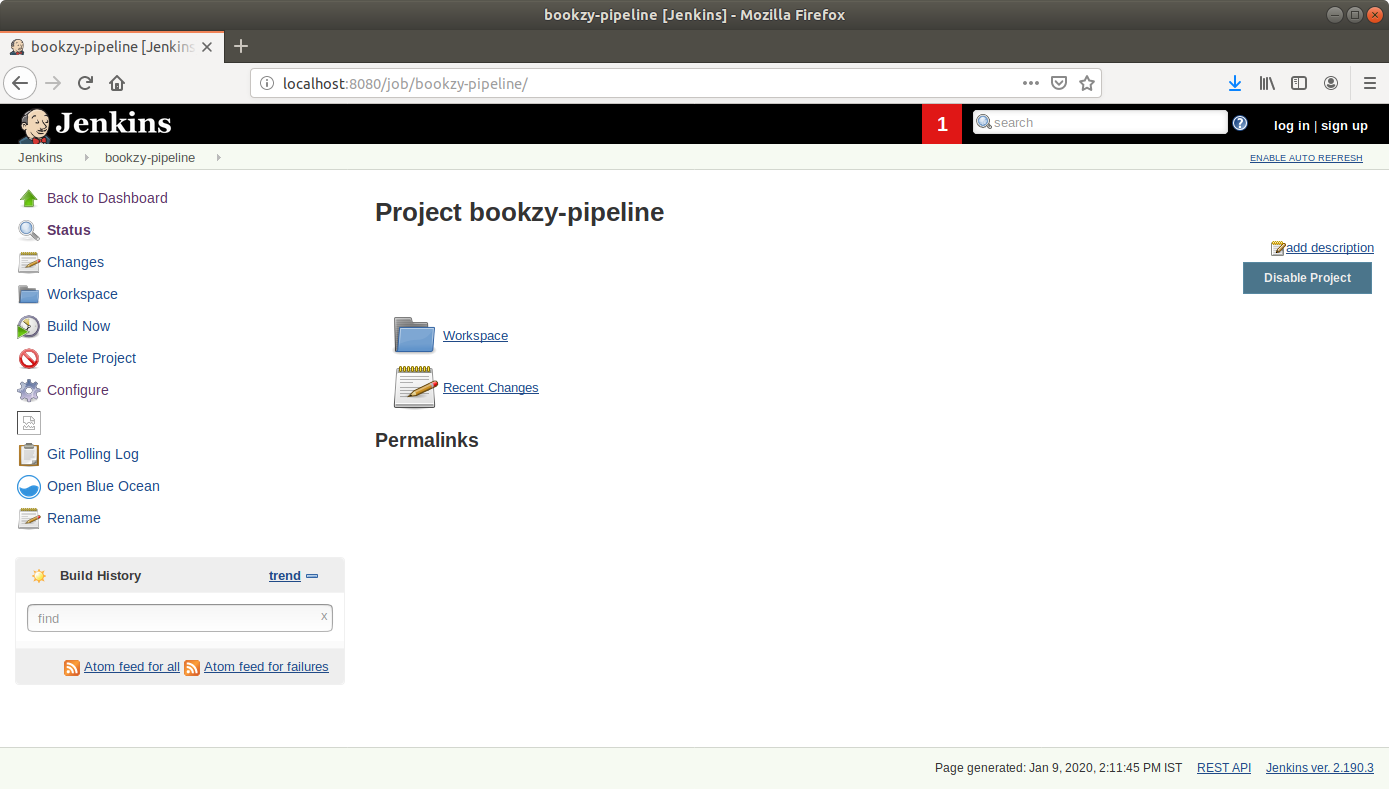
* Scroll down to the *Build Trigger* and pick the poll SCM option. Enter the expression for nightly builds.



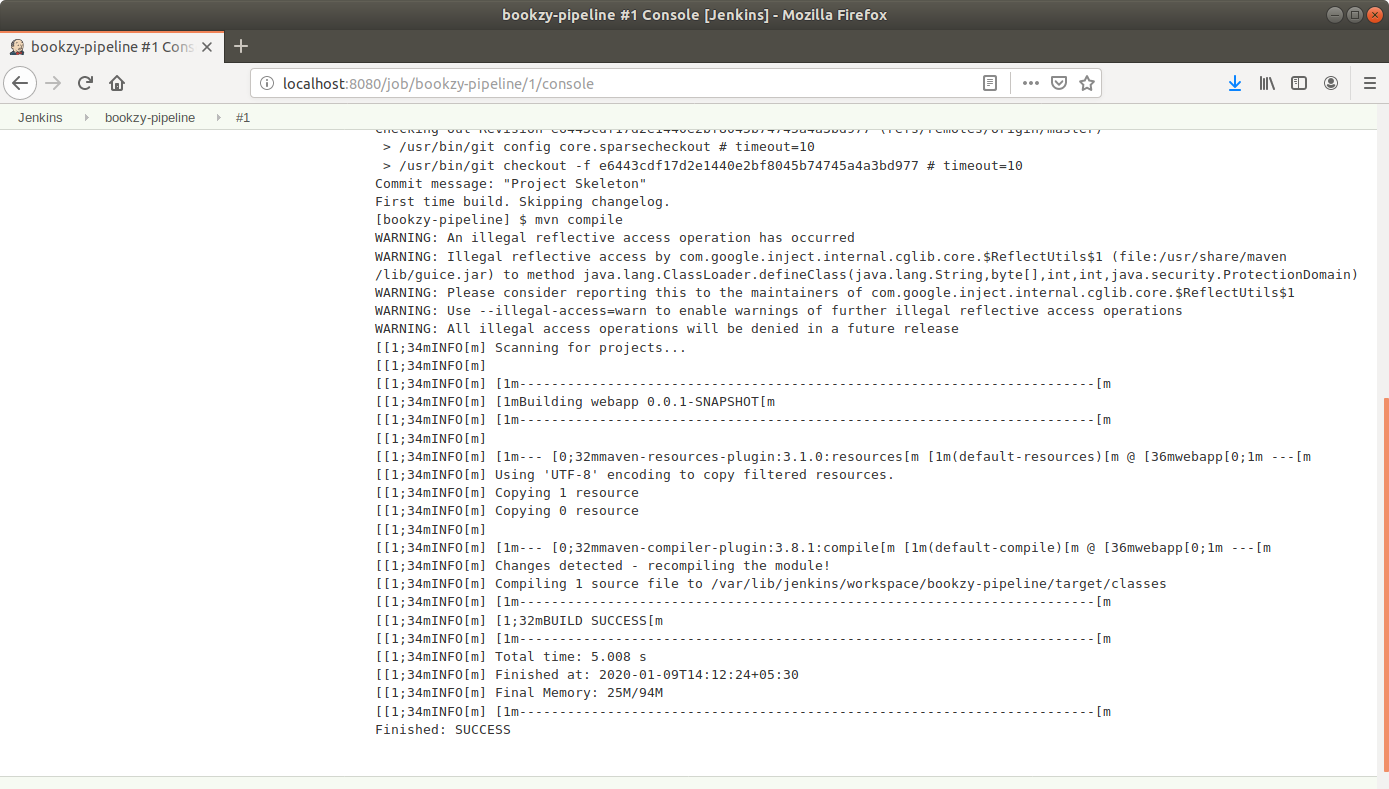
* Scroll down to the *Build* section.
* Click on the *Add build step* button.
* On the available options, click on the *Invoke top level Maven target* and enter ***compile*** as goal.



* Click on Save.

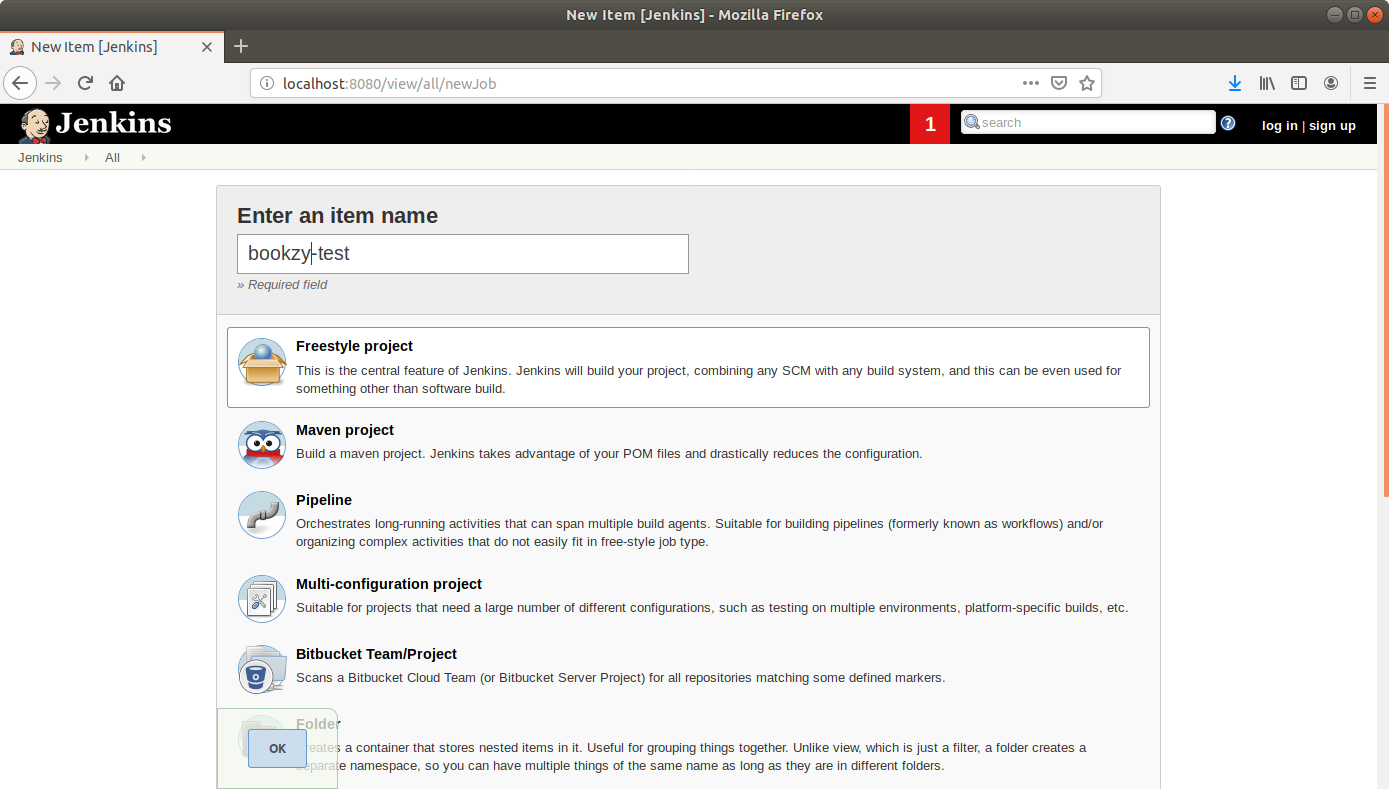


* Click on *Build Now* in the project window to test the job.

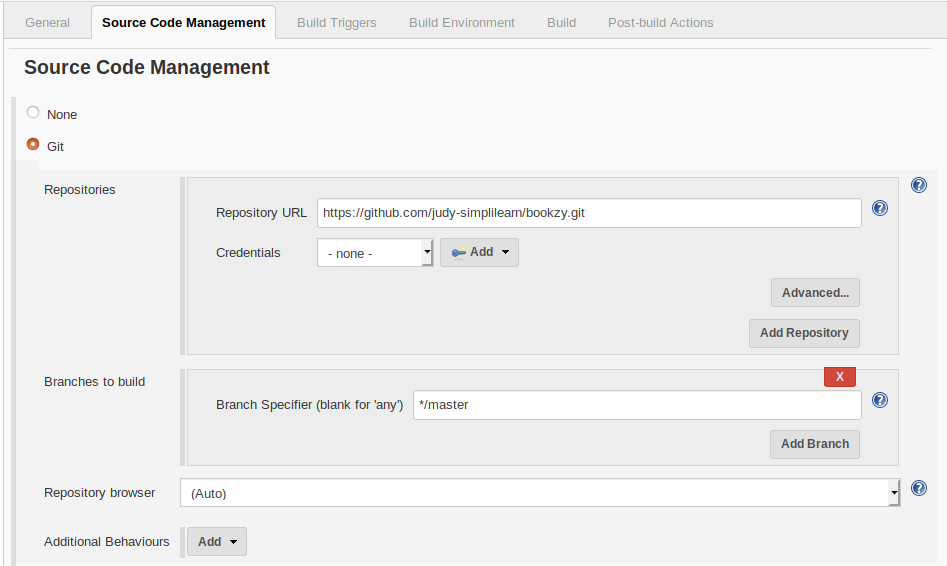


**Step 7:** Creating a freestyle job for testing in Jenkins

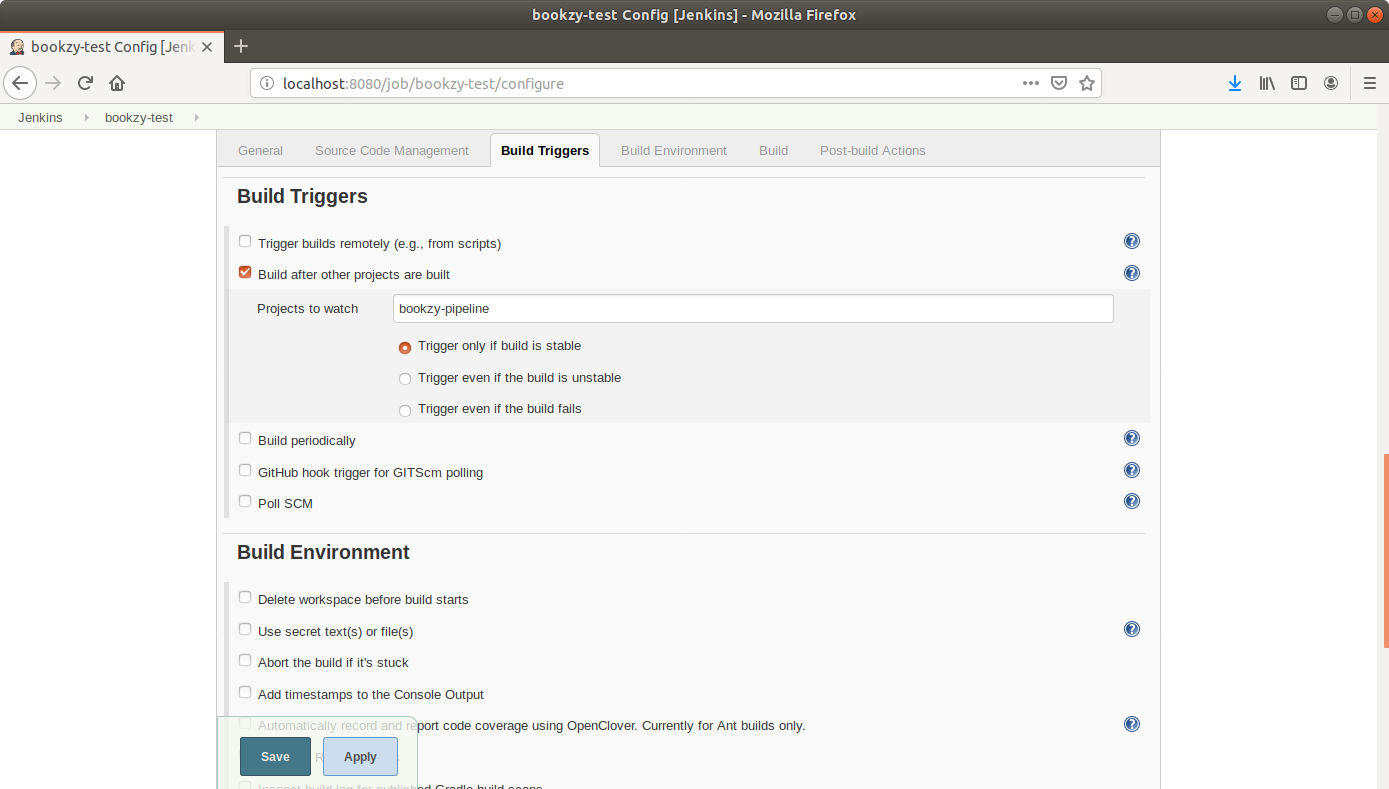
* Go to the Jenkins dashboard.
* Click on *New Item*.
* Enter a name for your build job.
* Select *Freestyle project* as the build job type.



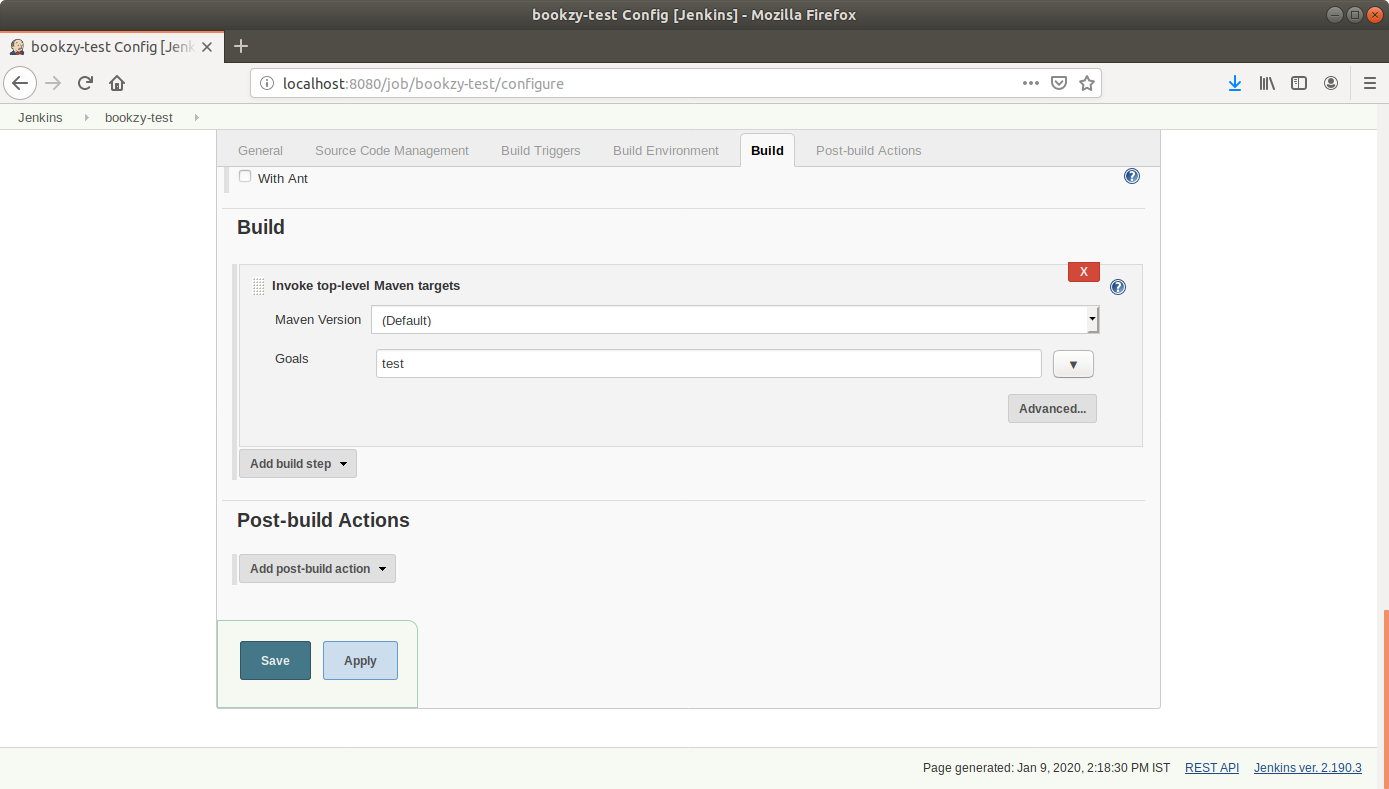
* Click OK.
* On the configuration page, scroll down to the Source Code Management section.
* Select *Git in SCM.*
* Add the repository URL.



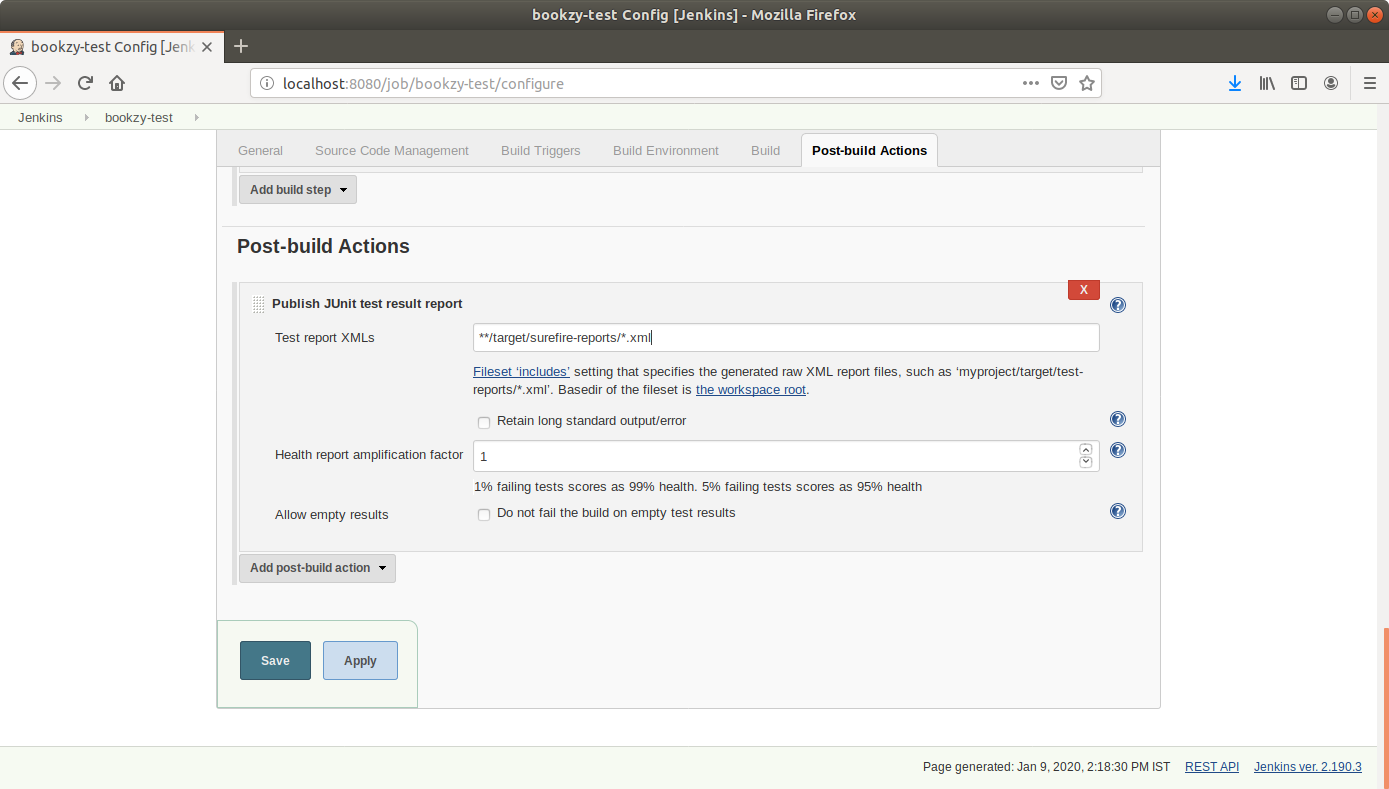
* In the *Build Trigger* tab, click on *build after other projects are built*.
* Type the name of the previous project and select the *Trigger only if the build is stable* option.



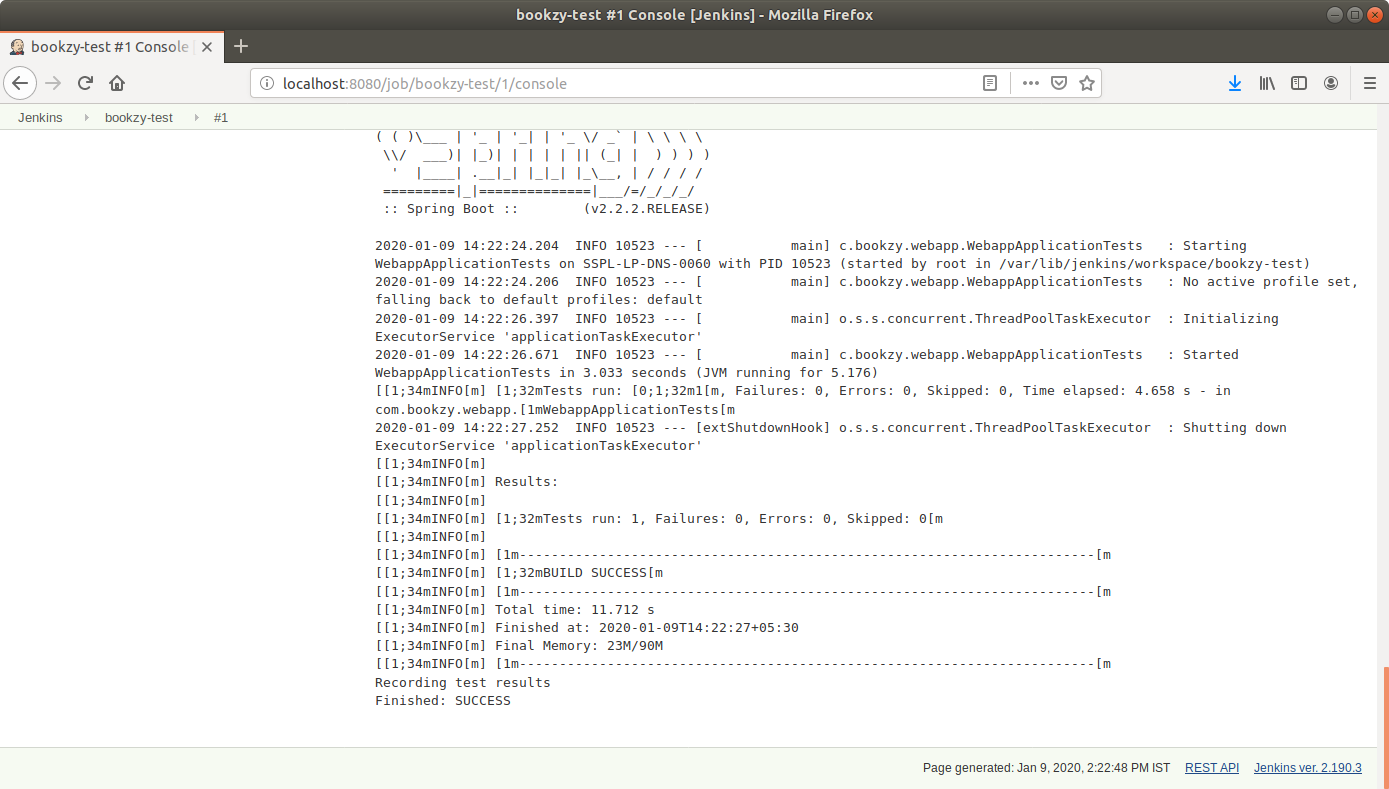
* Scroll down to the *Build* section.
* Click on the *Add build step* button.
* On the available options, click on the *Invoke top level Maven target* and enter ***test*** as the goal.



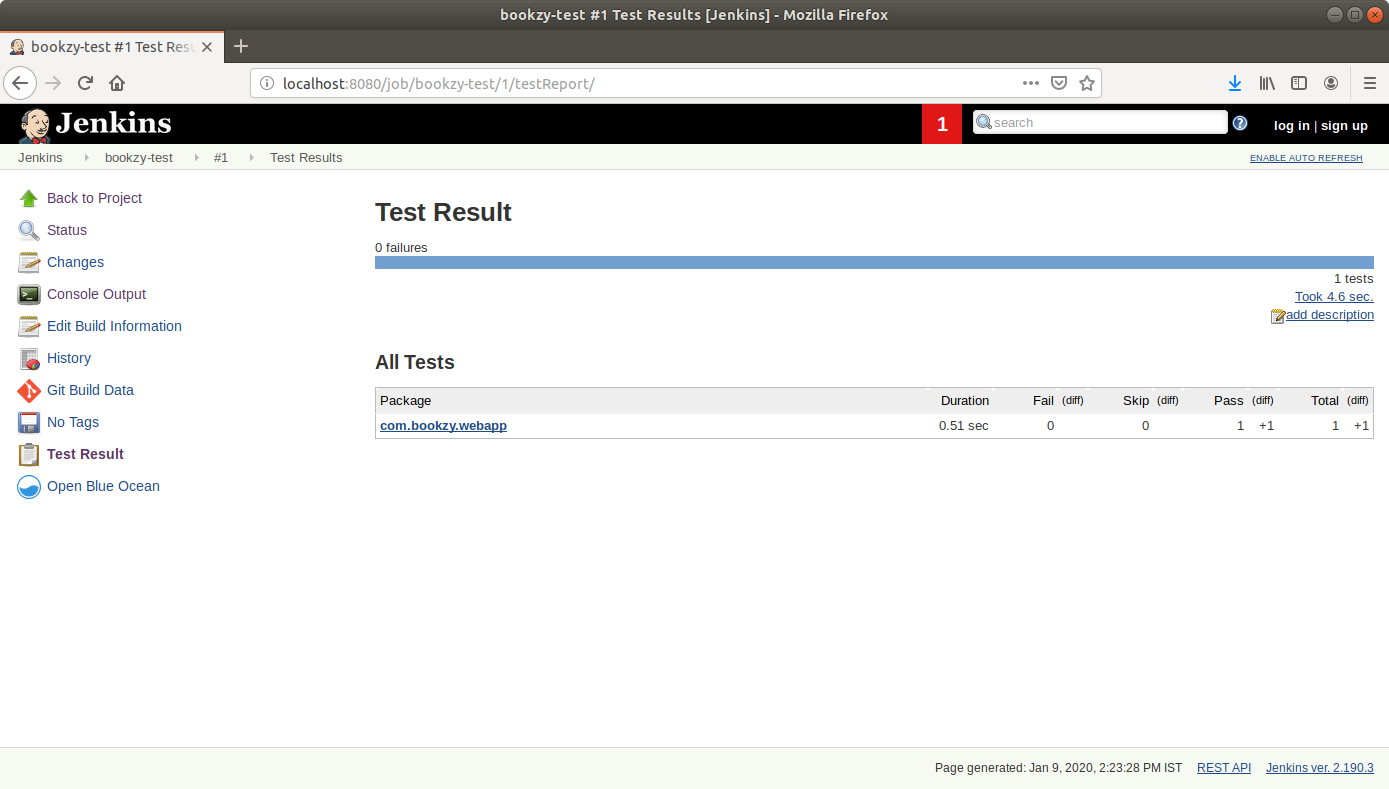
* Scroll down to the *Post-build Actions* section.
* Click on *Add post-build action* and select P*ublish JUnit test result report*. Fill the fields with appropriate values.



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

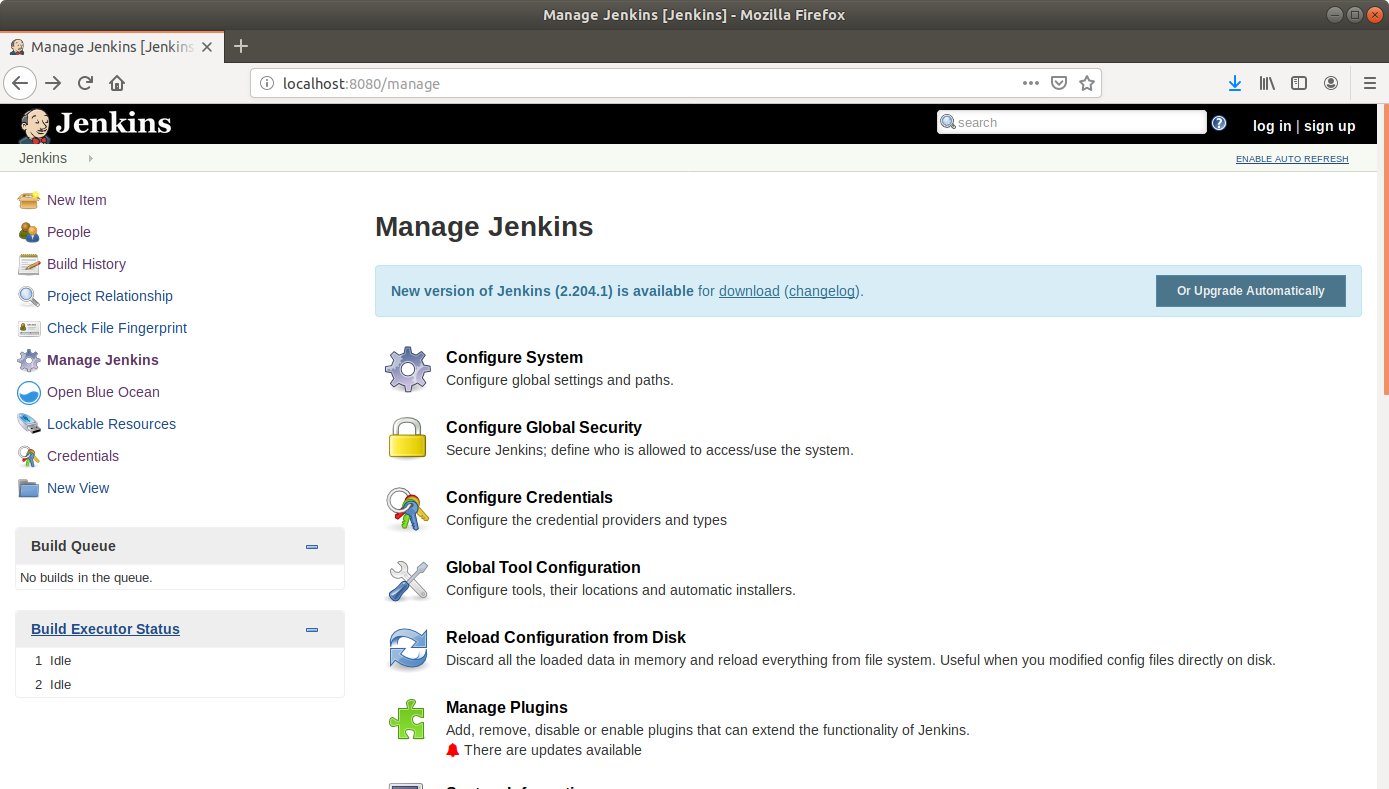


* The JUnit test reports can be viewed under the *Test Result* tab.

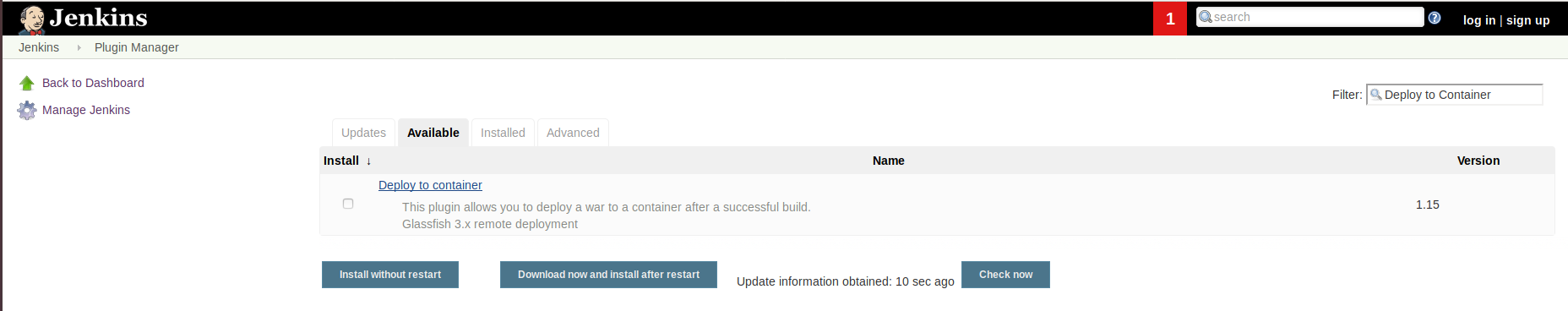


**Step 8:** Creating a freestyle job for deployment in Jenkins

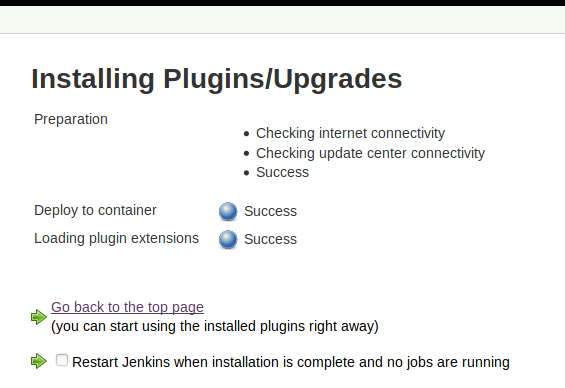
* Go to the Jenkins dashboard.
* Click on *Manage Jenkins* and select *Manage Plugins.*

**

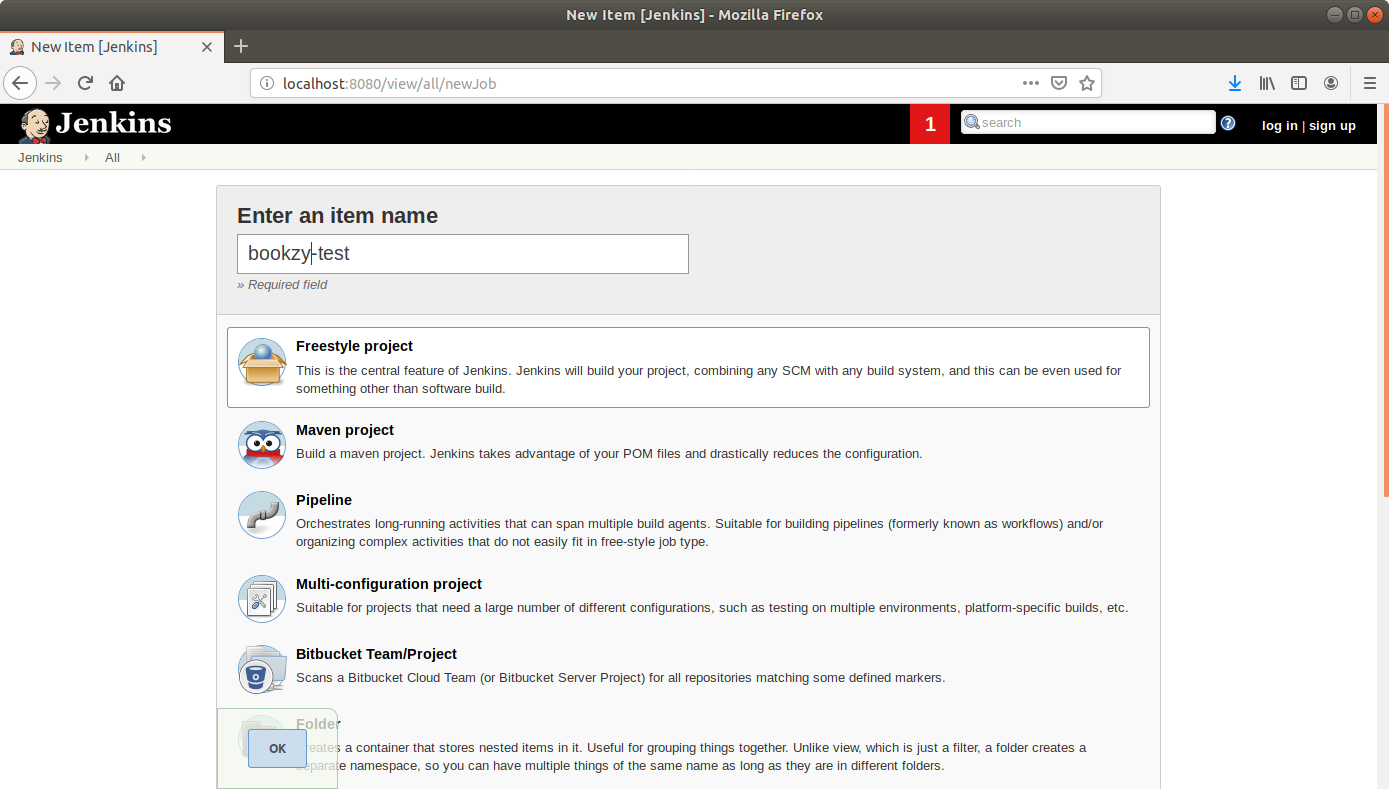
* From the available plugins, install *Deploy to container.*

**

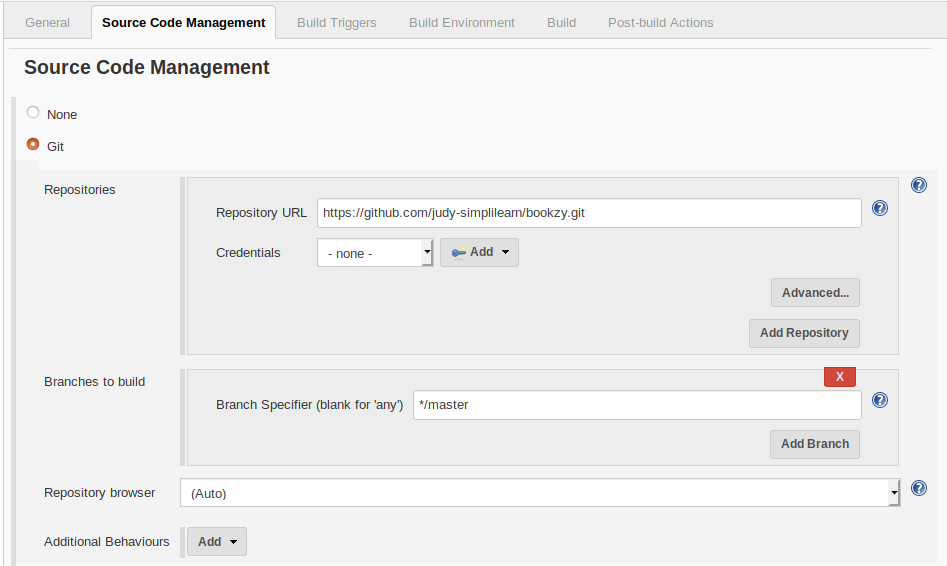
* Click on *Install without restart.*

**

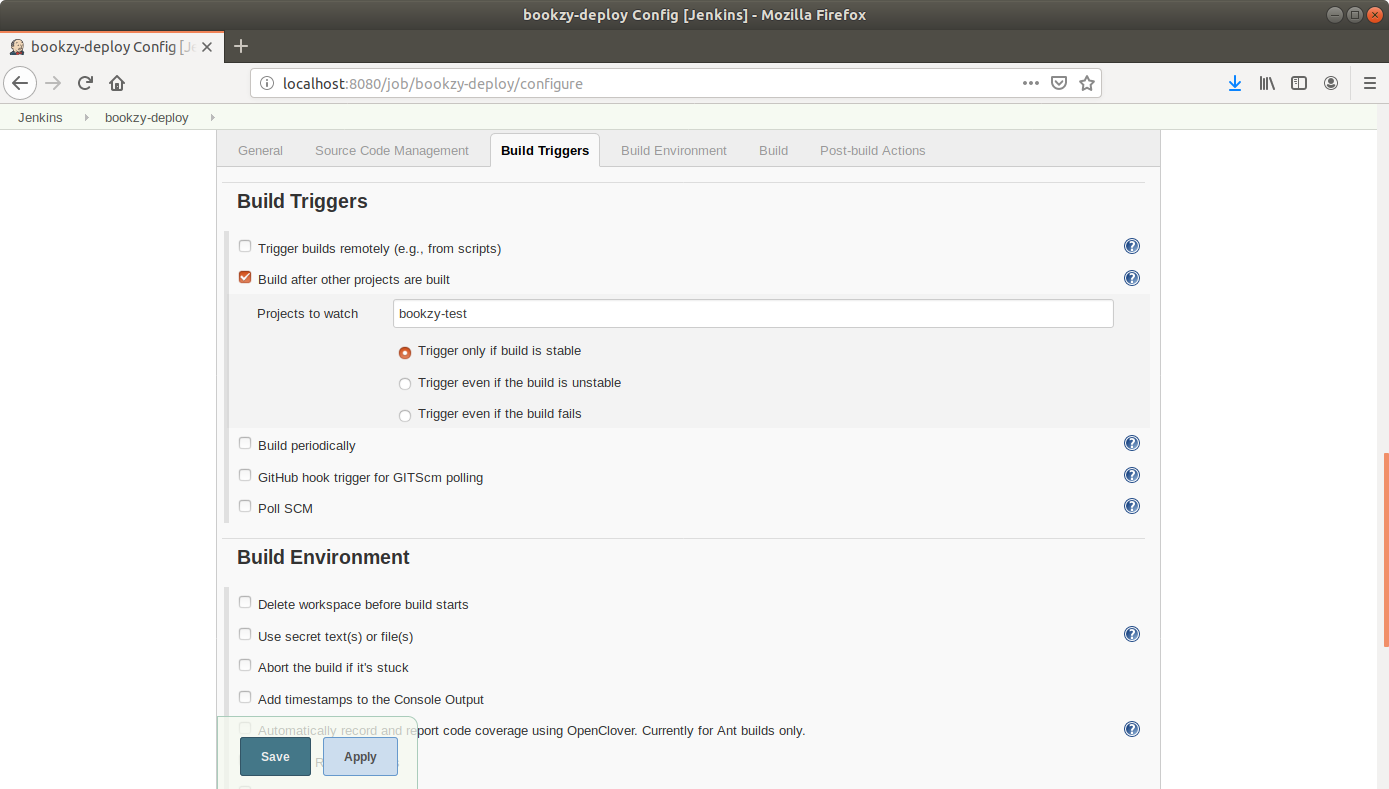
* Go to the Jenkins dashboard.
* Click on *New Item*.
* Enter a name for your build job.
* Select *Freestyle project* as the build job type.



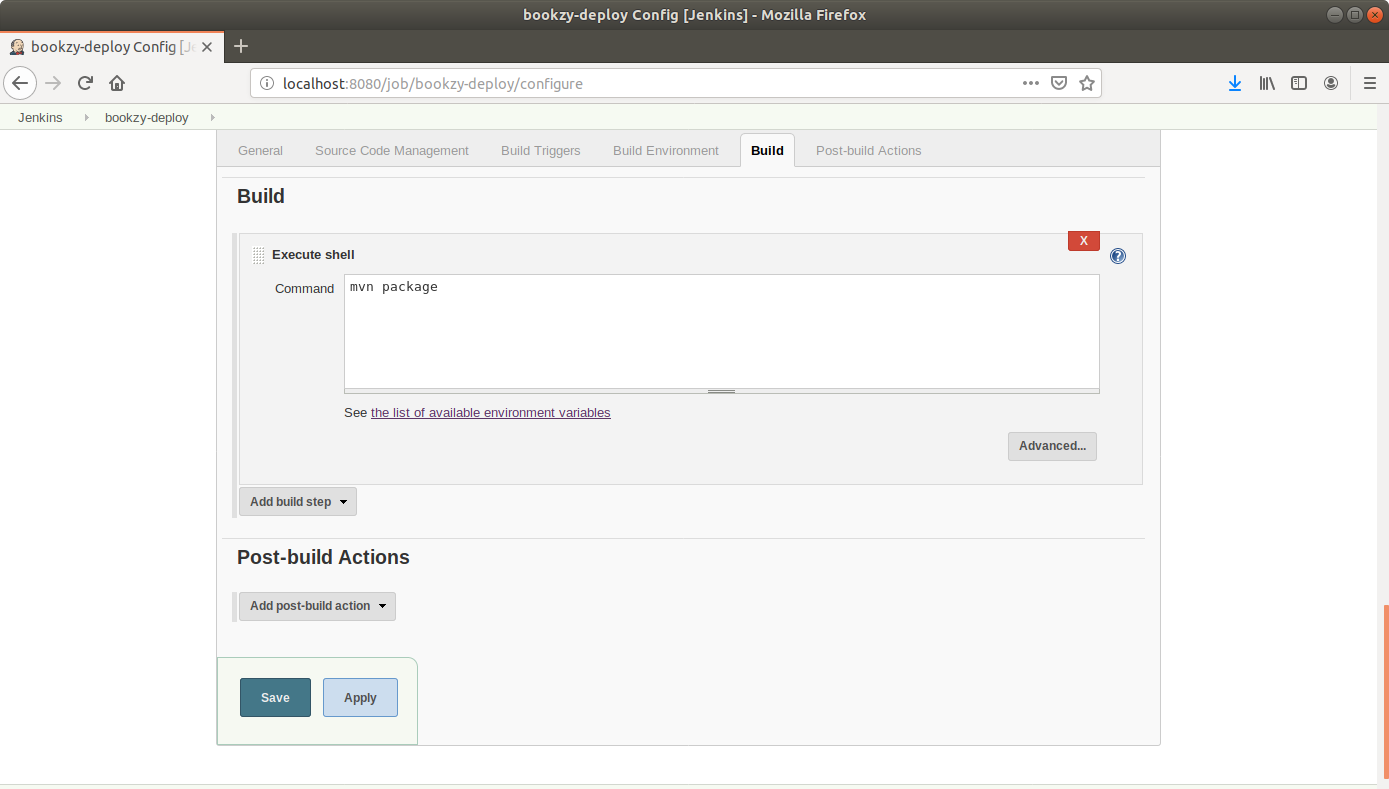
* Click OK.
* On the configuration page, scroll down to the Source Code Management section.
* Select *Git in SCM.*
* Add the repository URL.



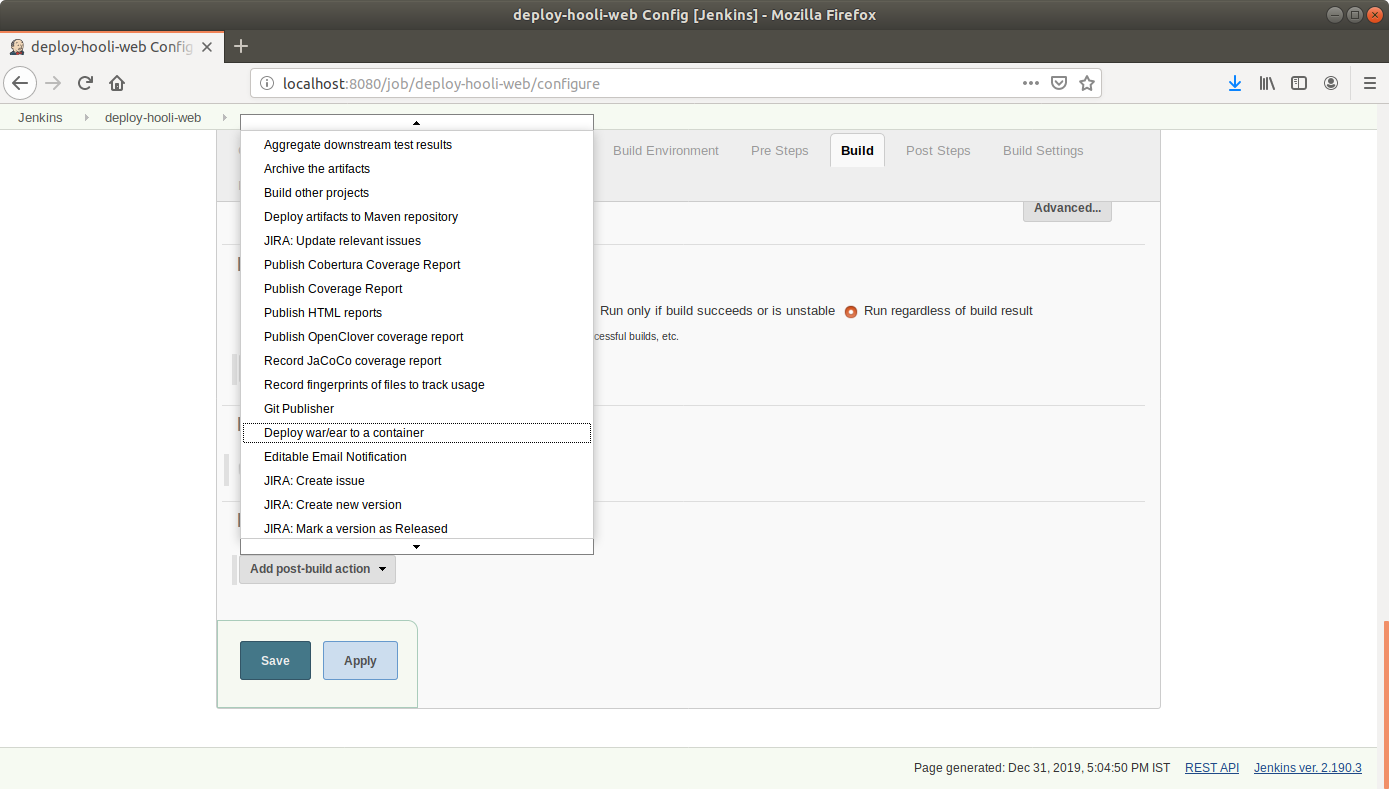
* In the *Build Trigger* tab click on the *build after other projects are built*.
* Type the name of the previous project and select the *Trigger only if the build is stable* option.



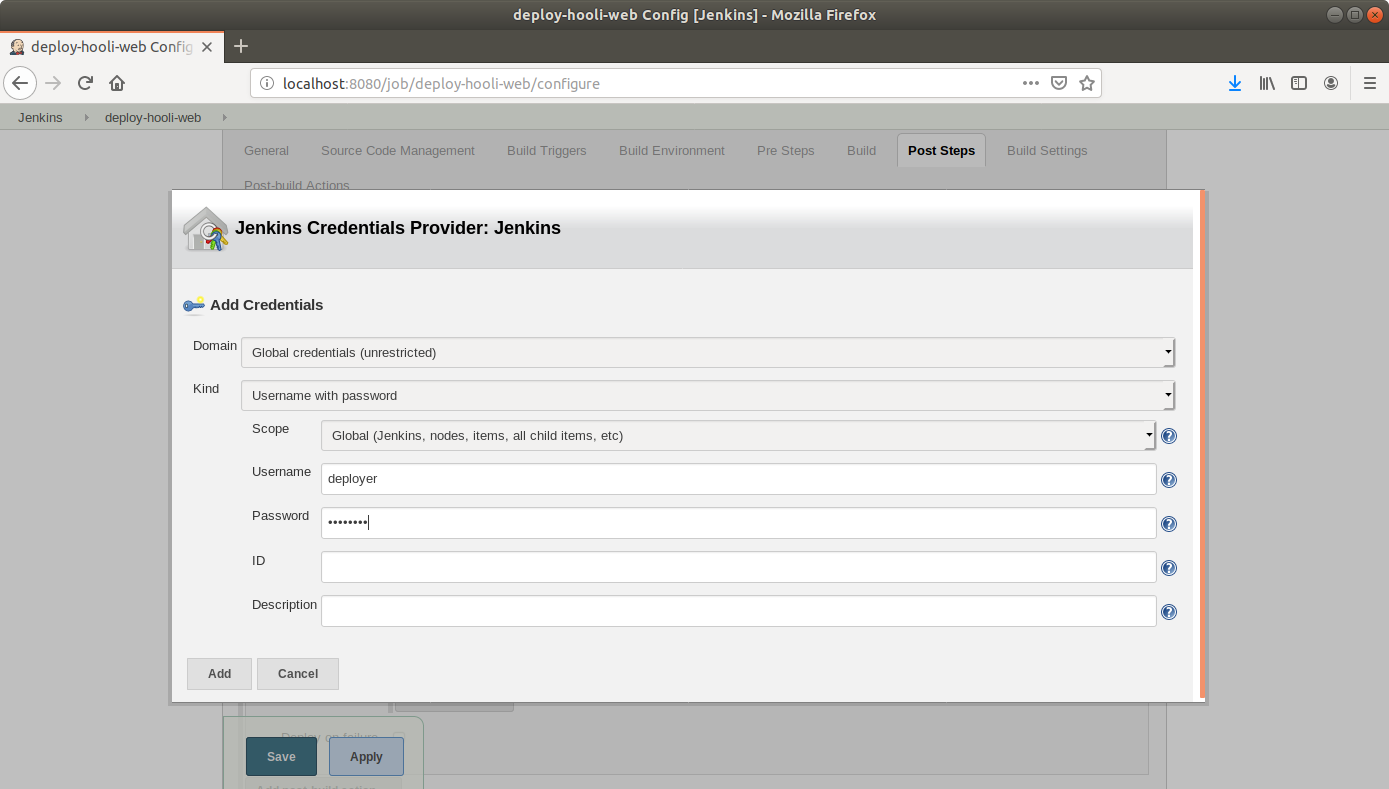
* Scroll down to the *Build* section.
* Click on the *Add build step* button.
* On the available options, click on the *Execute shell* and enter ***mvn package*** as command.



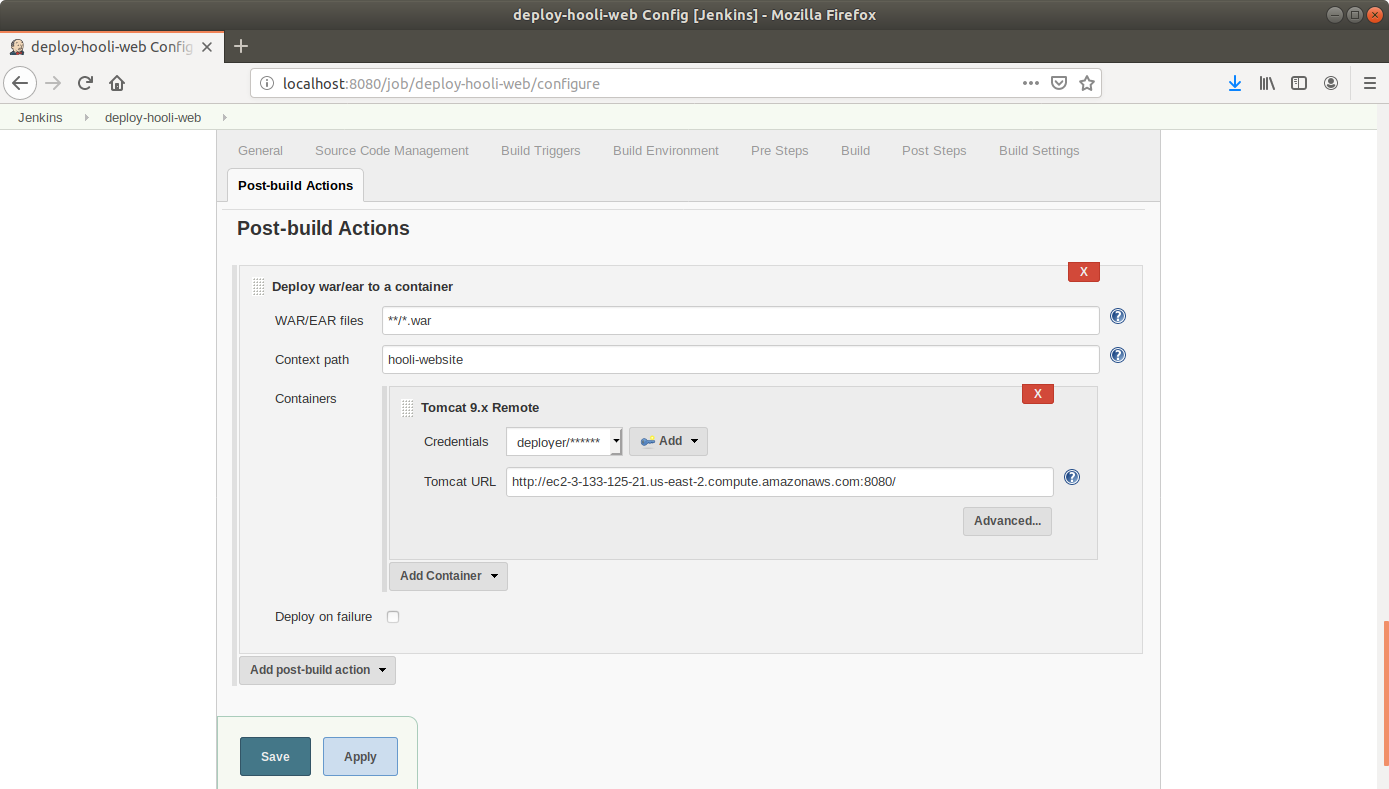
* Scroll down to the *Post-build Actions* section.
* Click on the *Add post-build action* button.
* On the available options, click on the *Deploy war/ear to container*.



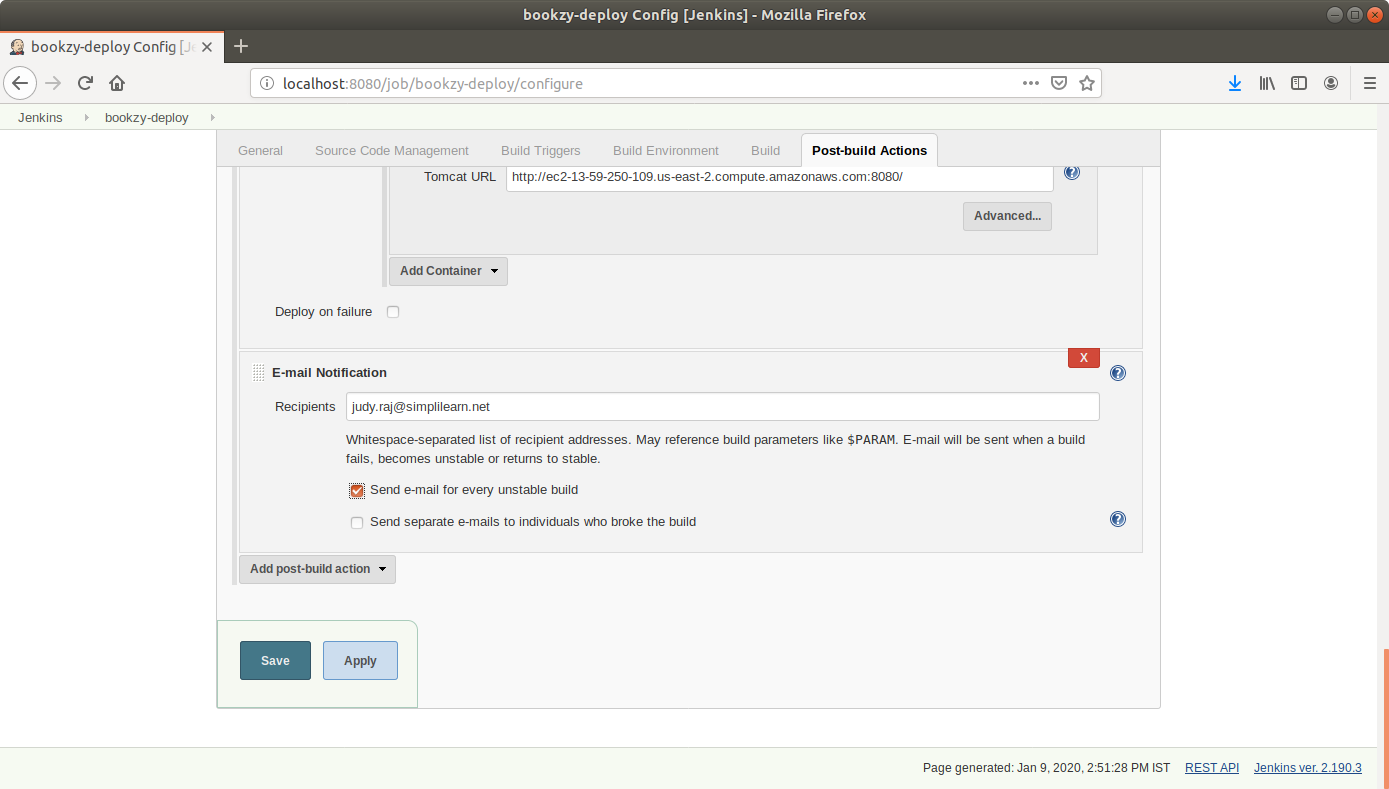
* Add the credentials.



* 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 on the *Add post-build action* and select *Sent Email Notification*. Fill the fields with appropriate values.



* Click on Save.
* Click on *Build Now* in the project window.
* Jenkins will now build your job and output the logs.
* Navigate to the URL on your browser to view your webapp.
* Trigger the first freestyle job to start the pipeline.