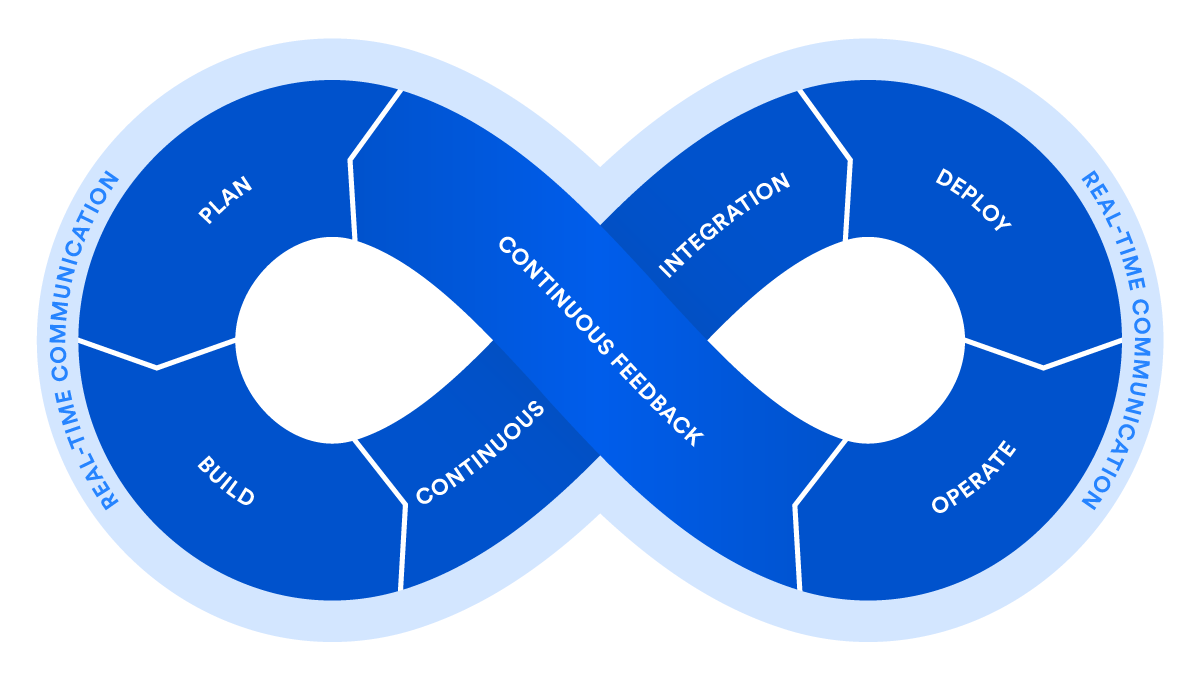
DevOps in DevonFrameWork

**What is DevOps?**

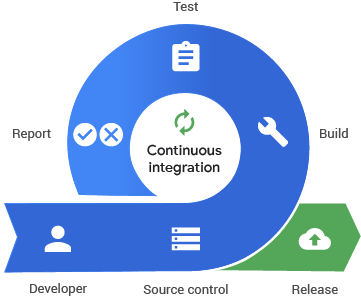


DevOps is a set of practices that automates the processes between software development and IT teams, in order that they can build, test, and release software faster and more reliably. The concept of DevOps is founded on building a culture of collaboration between teams that historically functioned in relative siloes. The promised benefits include increased trust, faster software releases, ability to solve critical issues quickly, and better manage unplanned work.

### **What is Continuous Integration?**

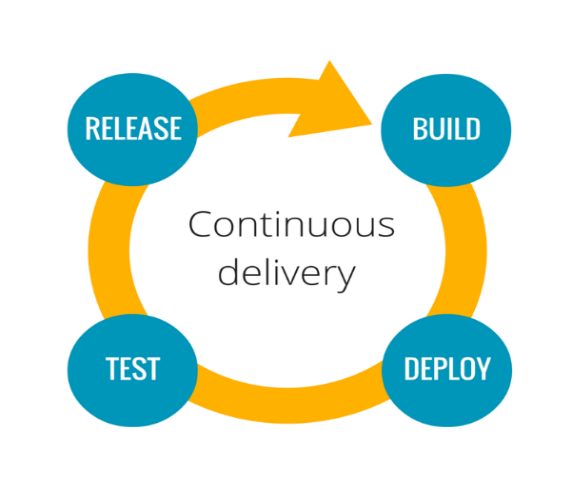
Continuous Integration (CI) is a development practice where developers integrate code into a shared repository frequently, preferably several times a day. Each integration can then be verified by an automated build and automated tests. While automated testing is not strictly part of CI it is typically implied.

Continuous Integration doesn’t get rid of bugs, but it does make them dramatically easier to find and remove.



**What is Continuous Delivery?**

## Continuous Delivery is the practice of keeping your codebase deployable at any point. Beyond making sure your application passes automated tests it has to have all the configuration necessary to push it into production. Many teams then do push changes that pass the automated tests into a test or production environment immediately to ensure a fast development loop.



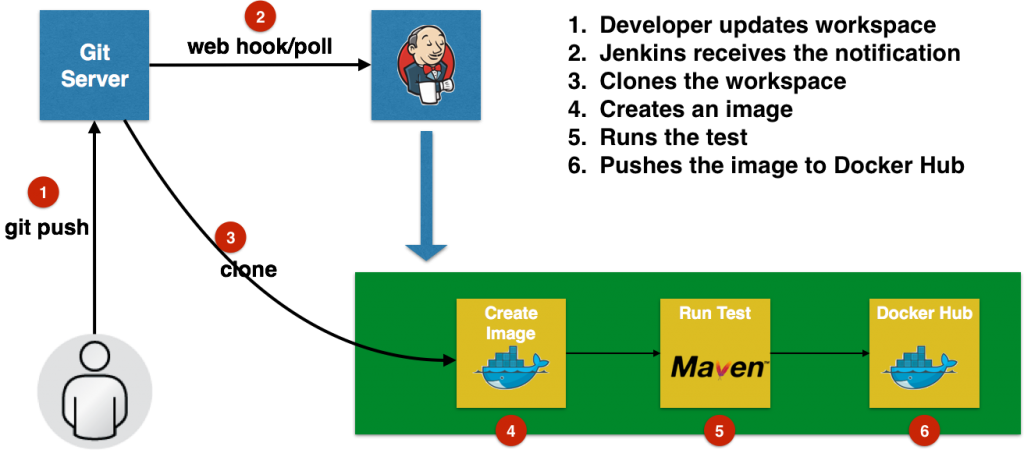
# What Is Git

Git is a free, open source distributed version control system tool designed to handle everything from small to very large projects with speed and efficiency.

**Jenkins:**

[Jenkins](http://jenkins-ci.org/) is an open-source [continuous integration](https://searchsoftwarequality.techtarget.com/definition/continuous-integration) software tool written in the [Java](https://www.theserverside.com/definition/Java) programming language for testing and reporting on isolated changes in a larger code base in real time. The software enables developers to find and solve defects in a code base rapidly and to automate testing of their builds.





**Docker:**

Docker is a containerization platform that packages your application and all its dependencies together in the form of Containers to ensure that your application works seamlessly in any environment.



As you can see in the diagram on the right, each application will run on a separate container and will have its own set of libraries and dependencies. This also ensures that there is process level isolation, meaning each application is independent of other applications, giving developers surety that they can build applications that will not interfere with one another.

As a developer, I can build a container which has different applications installed on it and give it to my QA team who will only need to run the container to replicate the developer environment.

## **Benefits of Docker**

Now, the QA team need not install all the dependent software and applications to test the code and this helps them save lots of time and energy. This also ensures that the working environment is consistent across all the individuals involved in the process, starting from development to deployment. The number of systems can be scaled up easily and the code can be deployed on them effortlessly.

**What is Docker Image**

Docker Image can be compared to a template which is used to create Docker Containers. They are the building blocks of a Docker Container. These Docker Images are created using the build command. These Read only templates are used for creating containers by using the run command. We will explore Docker commands in depth in the “Docker Commands blog”.

Docker lets people (or companies) create and share software through Docker images. Also, you don’t have to worry about whether your computer can run the software in a Docker image — a Docker container can always run it.

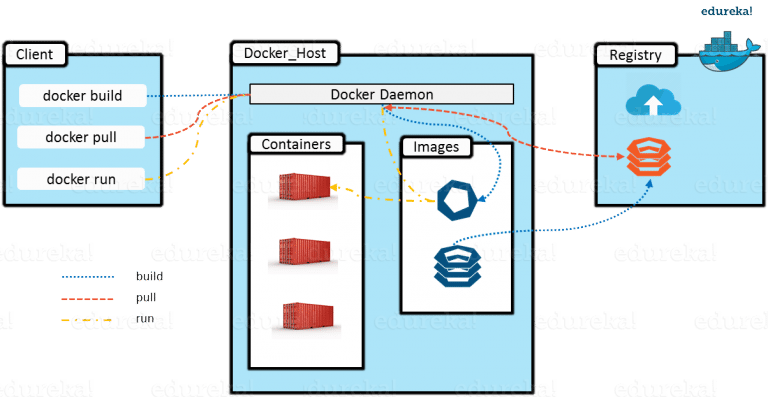
I can either use a ready-made docker image from docker-hub or create a new image as per my requirement. In the Docker Commands blog we will see how to create your own image.

## **What is Docker Container**

Containers are the ready applications created from Docker Images or you can say a Docker Container is a running instance of a Docker Image and they hold the entire package needed to run the application. This happens to be the ultimate utility of Docker.

## **What is Docker Registry**

Finally, Docker Registry is where the Docker Images are stored. The Registry can be either a user’s local repository or a public repository like a Docker Hub allowing multiple users to collaborate in building an application. Even with multiple teams within the same organization can exchange or share containers by uploading them to the Docker Hub. Docker Hub is Docker’s very own cloud repository similar to GitHub.



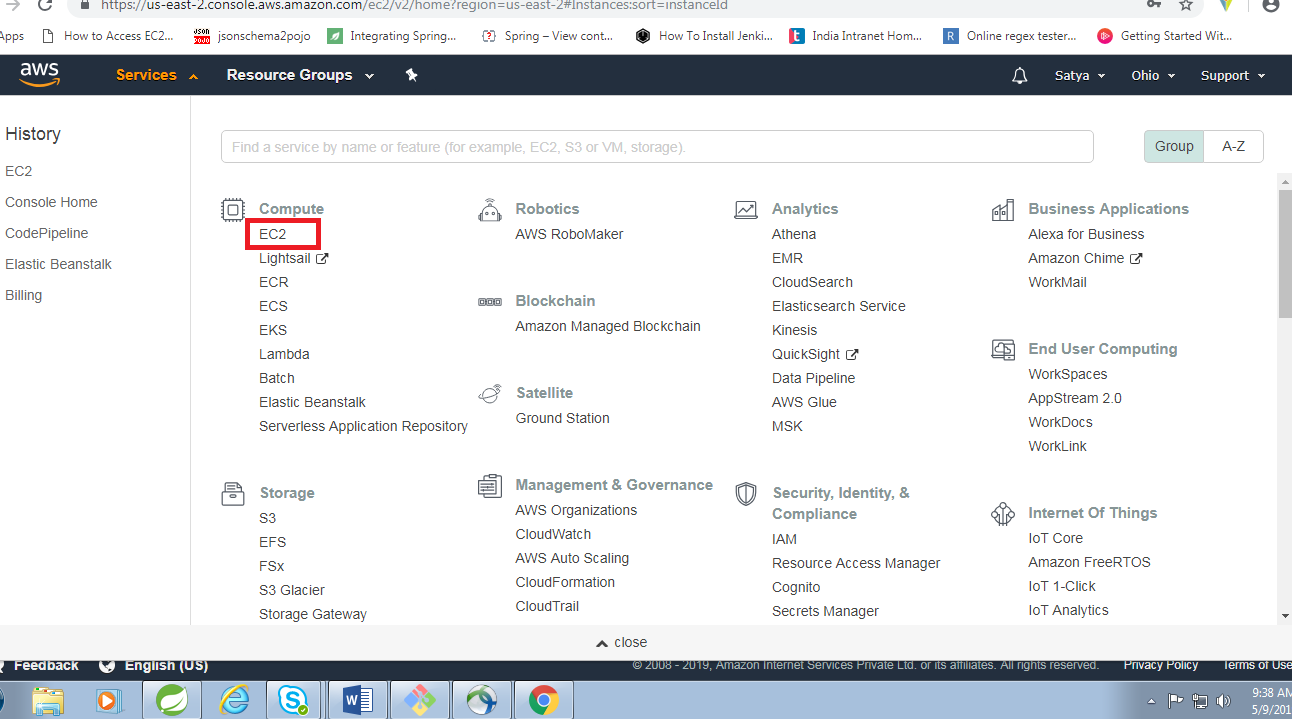
**What is AWS? – Amazon Web Services(AWS)** is a cloud service

from Amazon, which provides services in the form of building blocks, these building blocks can be used to create and deploy any type of application in the cloud.These services or building blocks are designed to work with each other, and result in applications which are sophisticated and highly scalable.

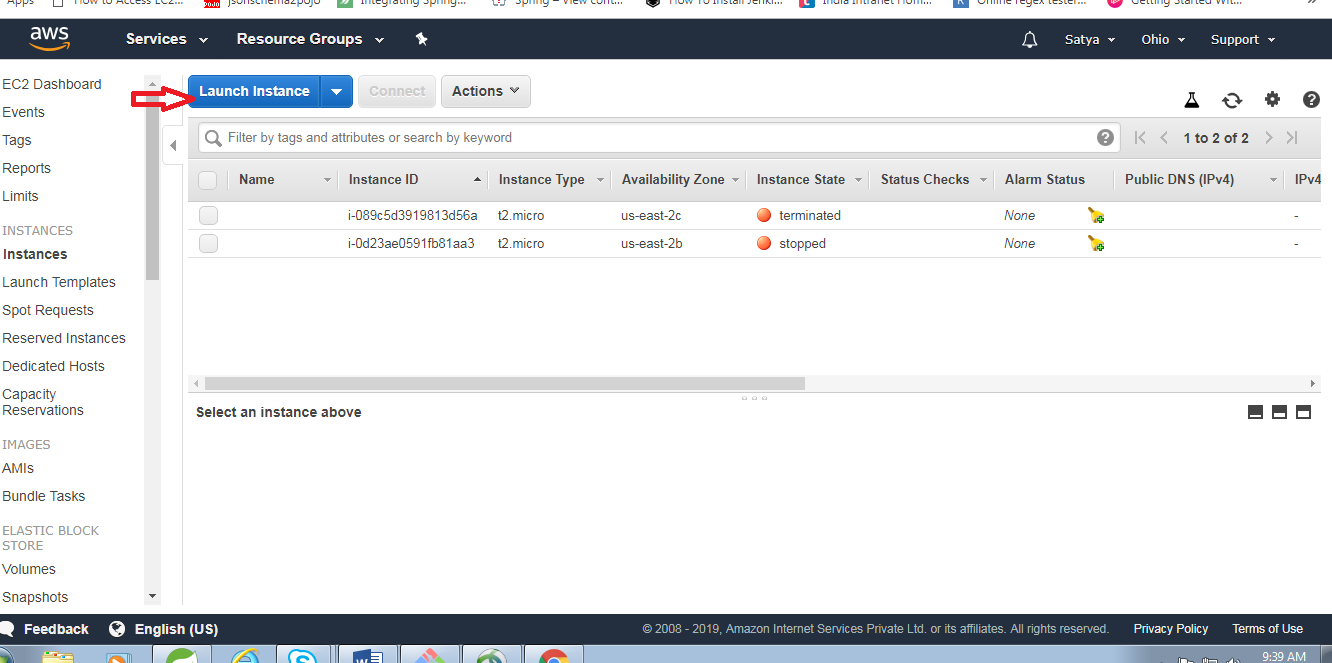
We should create account in AWS

To launch the instance in Amazon Web Services :

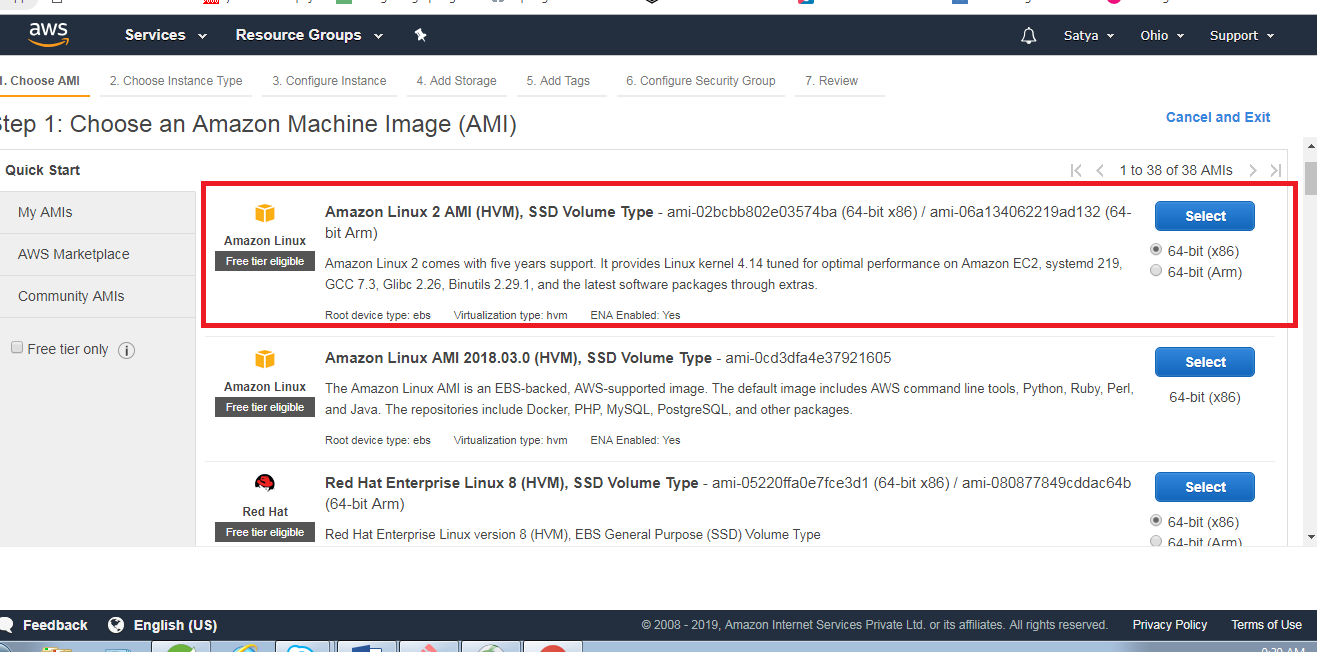
Step-1: Select the **EC2** from the services



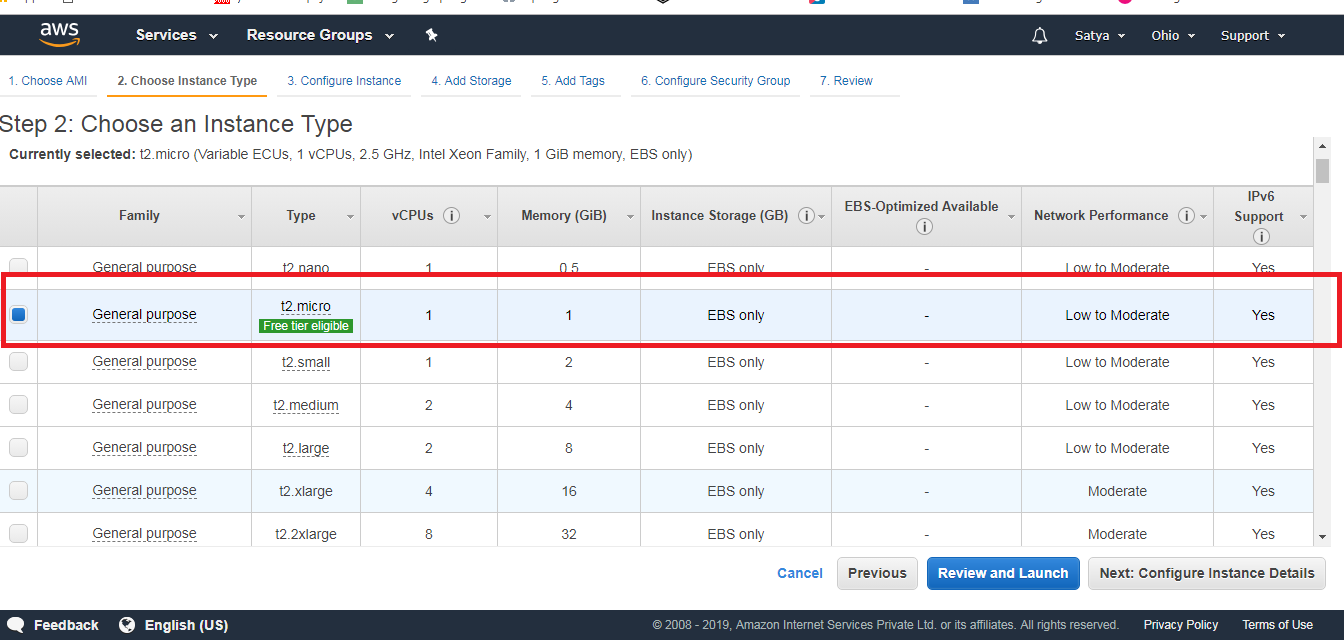
Step-2: Select **launch instance** from next page as follows



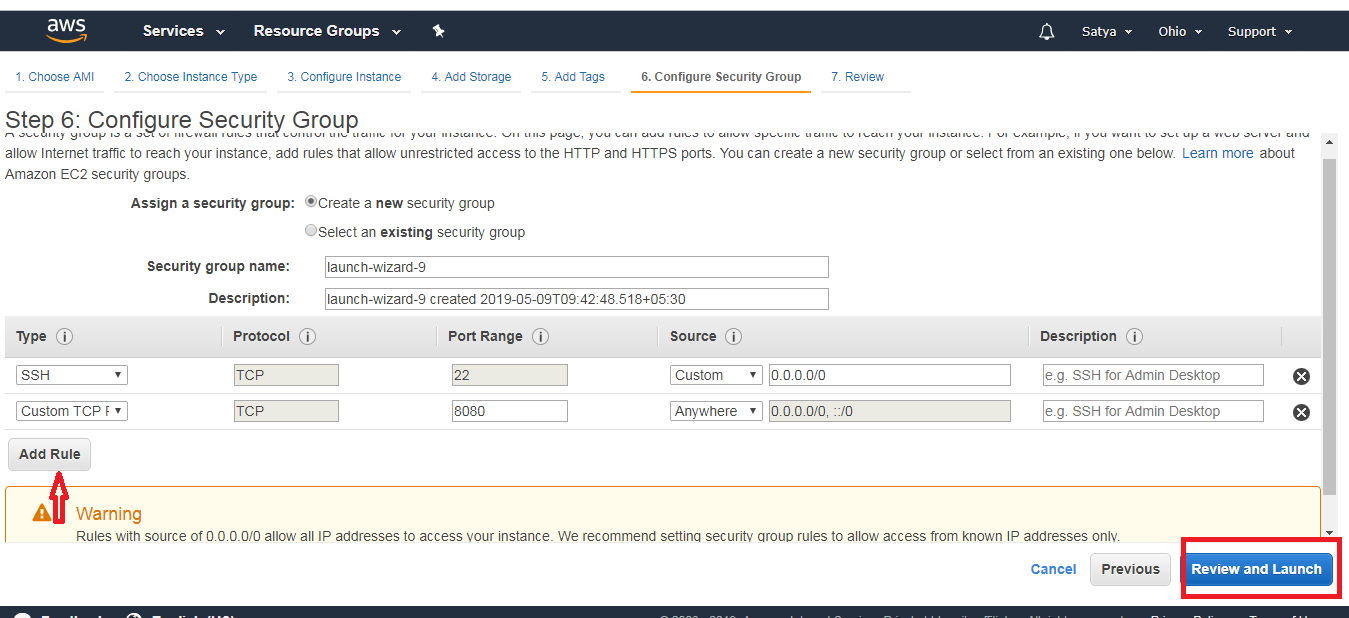
Step-3: Select **Amazon Linux 2** from the following.We can select any other according to our requirement.



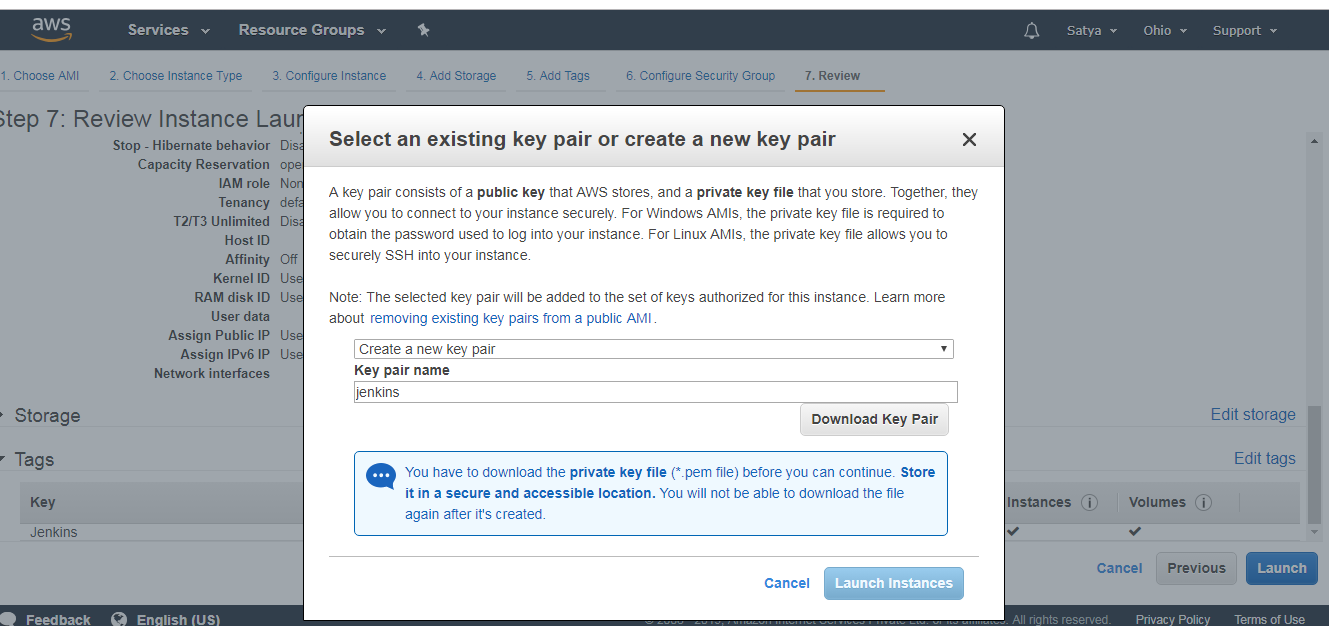
Step-4: Select the **free tier eligible** type as following



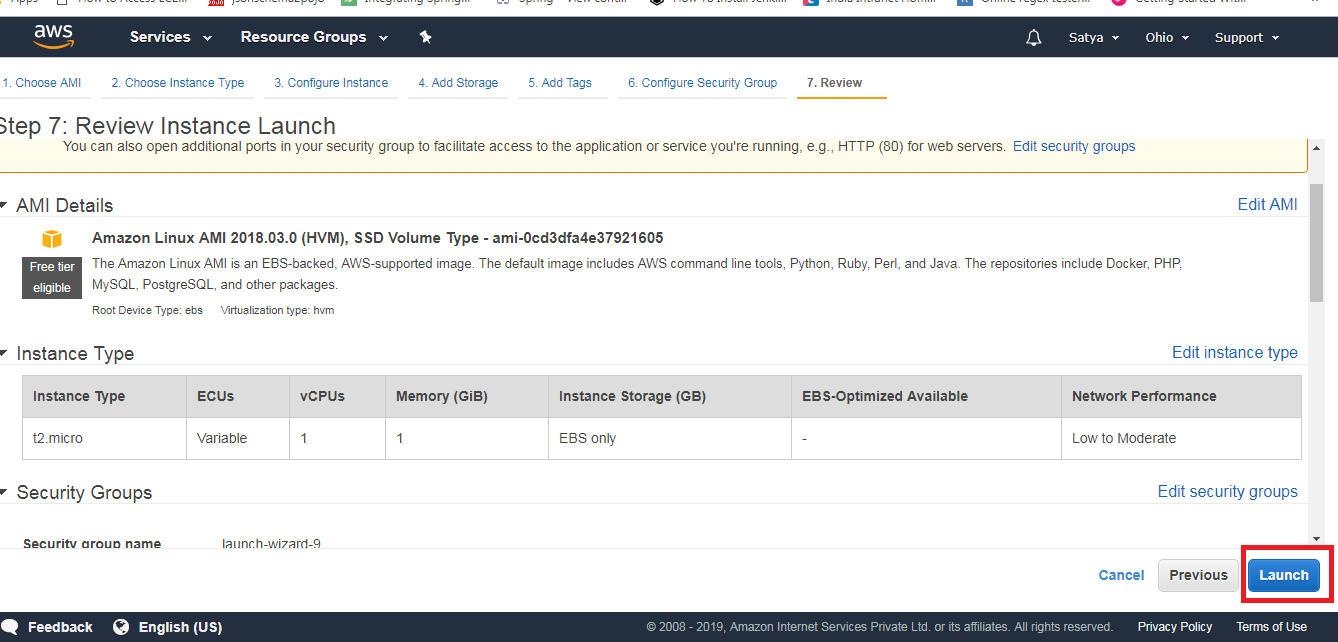
Step-5:click on **add Rule** to add the ports to run this instance and then click **Review and Launch.**



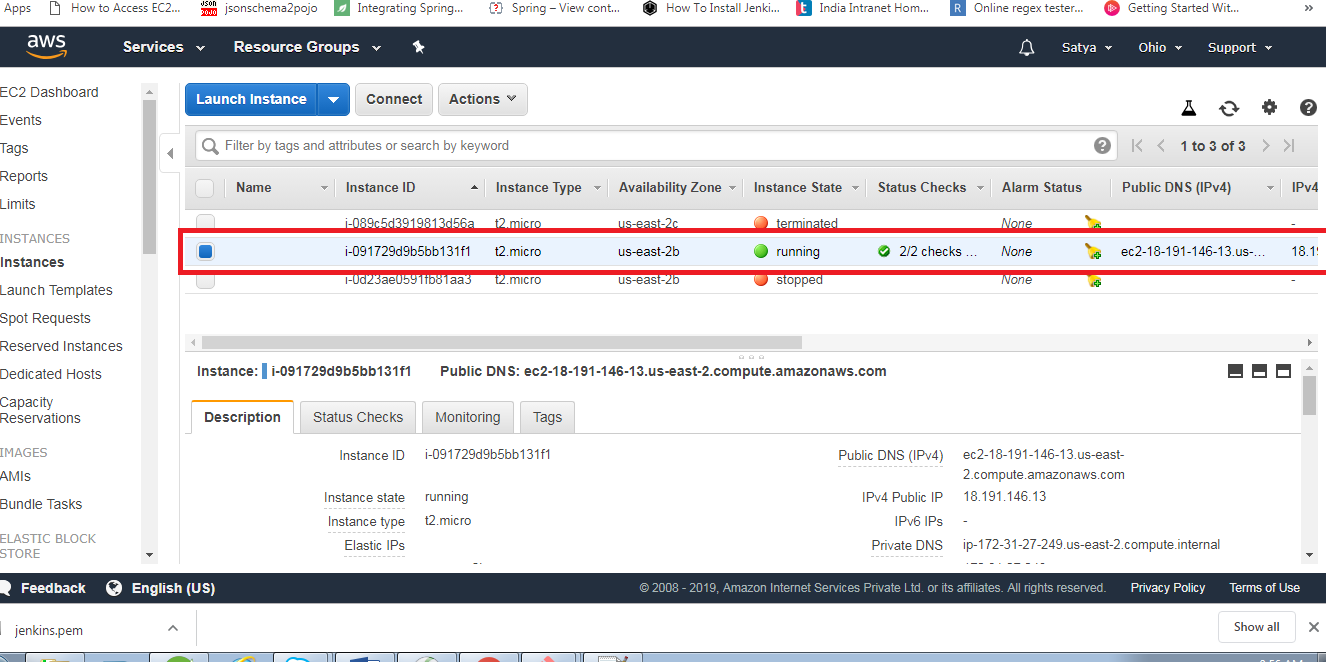
Step-6:create the **new key pair** and select download keyPair and save it to any folder (You have to download the **private key file** (\*.pem file) before you can continue. **Store it in a secure and accessible location.** You will not be able to download the file again after it's created.).



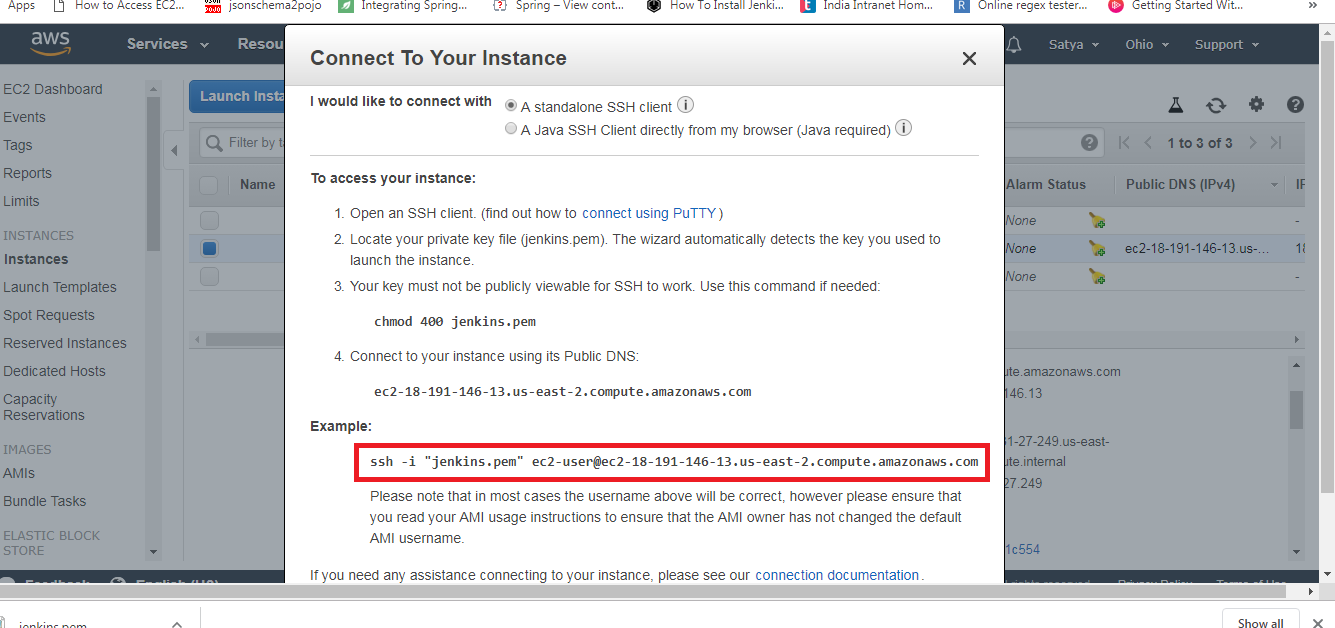
Step-5: We can check the details of our instance and click **Launch.**



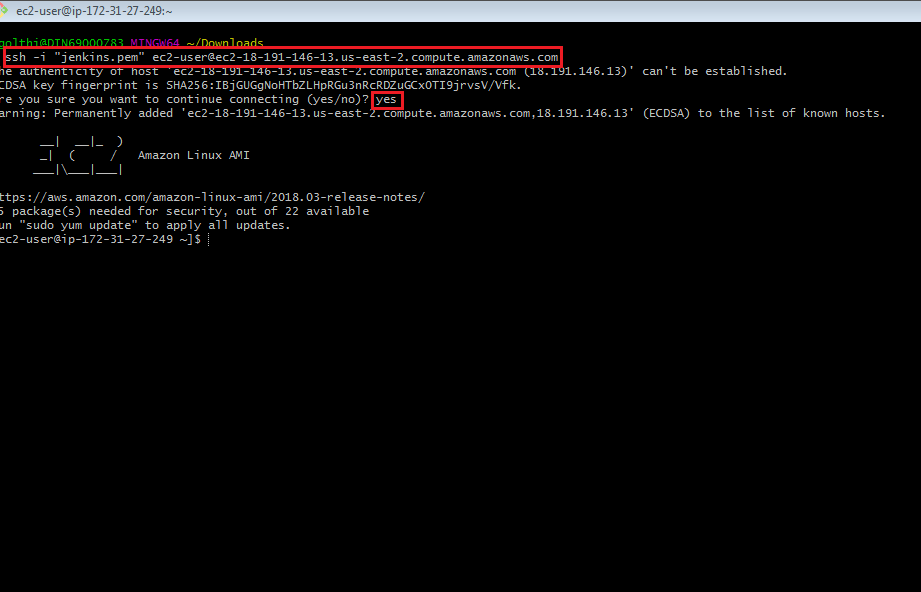
Step-6:Click on **view instances** to view the running instance. We can see the running instances



Step-8:Click **Connect** and copy the link selected in picture



Step-9:Open the gitbash in folder where we saved our key pair.



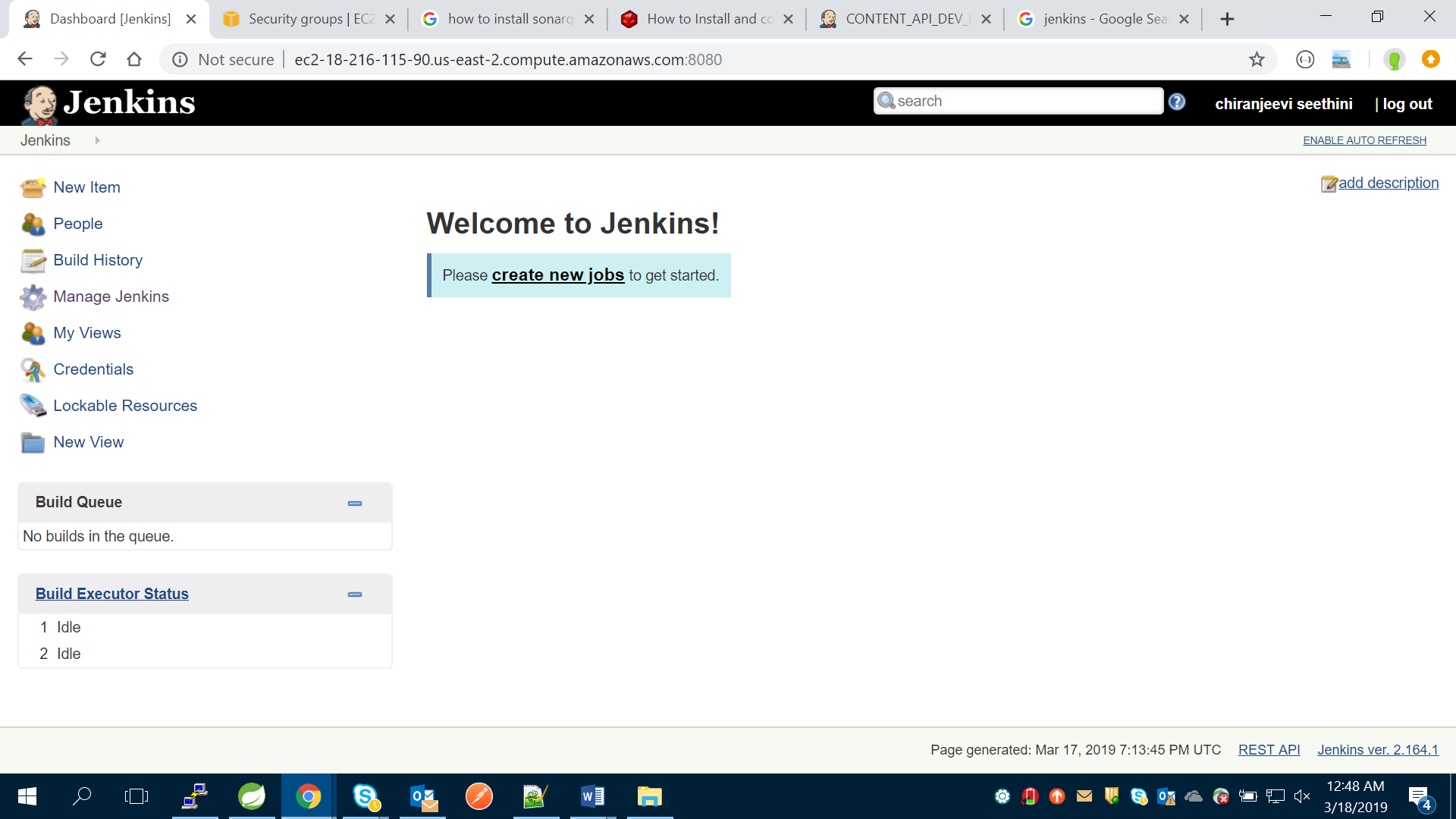
**Getting started with Jenkins**

Jenkins is an open source automation tool written in **Java** with plugins built for **Continuous** Integration purpose. Jenkins is used to build and test your software projects continuously making it easier for developers to integrate changes to the project, and making it easier for users to obtain a fresh build.

To start using Jenkins you need to install Jenkins

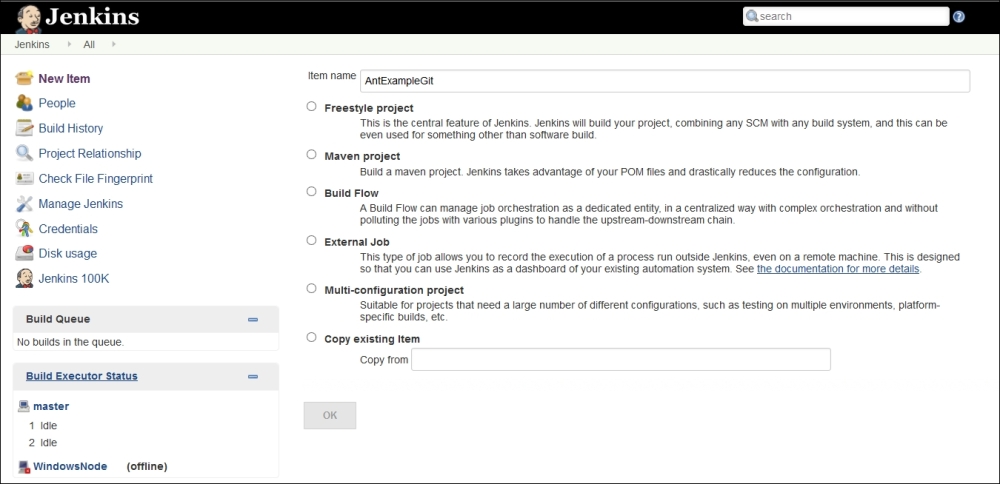
Jenkins by default run on port number 8080.

To open Jenkins the url will be http://<host\_name/ip >:8080 jenkins home page will looks like this when you launces it prior to configure.

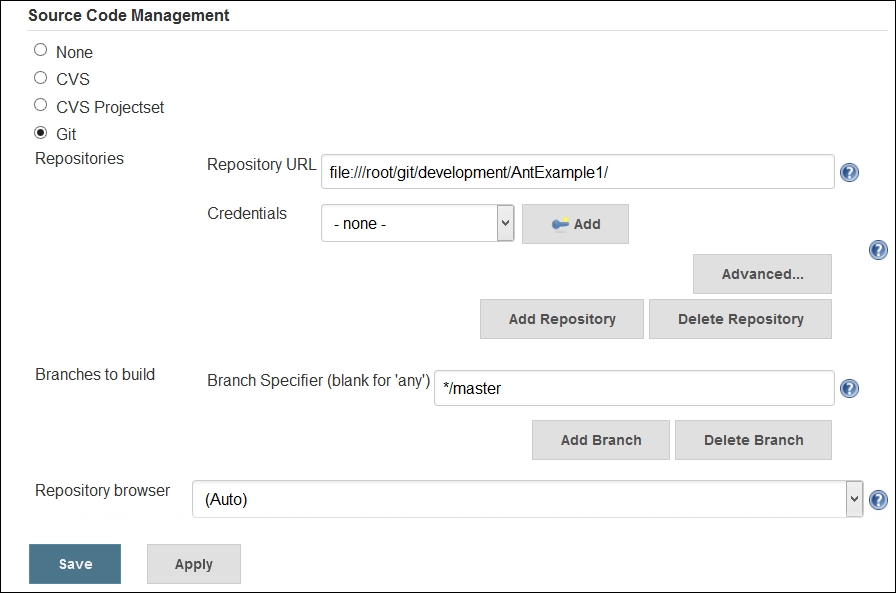


Creating job in Jenkins with git.

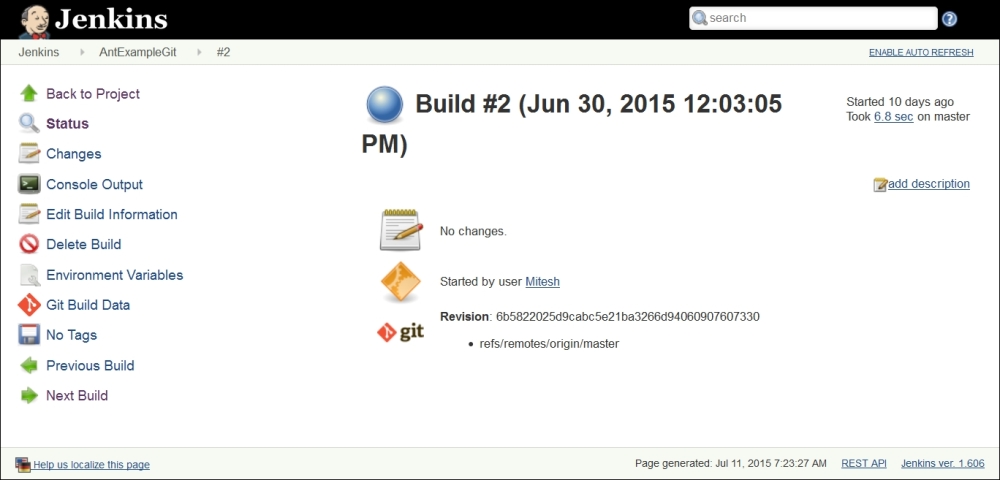
1. Installing the required plugin in Jenkins dashboard
   1. On the Jenkins dashboard, click on **Manage Jenkins** and select **Manage Plugins**. Click on the **Available** tab and write github plugin in the search box.
   2. Click the checkbox and click on the button, **Download now and install after restart**.
   3. Restart Jenkins.
2. Click on new item from the left side navigation, choose free style project and provide a name for your job and then click ok.



1. Check the Git hub project checkbox and provide cloning URL of your project code from the git hub.



1. From the **source code management** tab select Git and provide details like project repository URL, branch and credentials.
2. You can select build tool as well from the build section.
3. Click on save button.
4. Execute the build.



1. Click on Console Output to see the progress of the build.