**Jenkins**

**Jenkins:** It's a continuous integration automation server, used to automate repetitive tasks and also for continuous integration.

Jenkins is an open-source automation server that helps automate parts of the software development process, such as building, testing, and deploying code. It's widely used for continuous integration and continuous delivery (CI/CD).

**Essential Requirements for Installing Jenkins & Step-by-Step Guide to Installing Jenkins**

Run the commands below on your EC2 instance.

* **Update patches on Linux Machine:**

yum -y update

* **Install git:**

yum install -y git

* **Install Java:**

yum install -y java-21-amazon-corretto

yum install -y java-21-amazon-corretto-devel

[root@ip-172-31-41-161 ~]# java --version

openjdk 21.0.5 2024-10-15 LTS

OpenJDK Runtime Environment Corretto-21.0.5.11.1 (build 21.0.5+11-LTS)

OpenJDK 64-Bit Server VM Corretto-21.0.5.11.1 (build 21.0.5+11-LTS, mixed mode, sharing)

[root@ip-172-31-41-161 ~]#

[root@ip-172-31-41-161 ~]#

[root@ip-172-31-41-161 ~]# which java

/usr/bin/java

[root@ip-172-31-41-161 ~]#

[root@ip-172-31-41-161 ~]#

[root@ip-172-31-41-161 ~]# readlink -f /usr/bin/java

/usr/lib/jvm/java-21-amazon-corretto.x86\_64/bin/java

* **Install Maven:**

cd /opt

sudo wget https://dlcdn.apache.org/maven/maven-3/3.9.9/binaries/apache-maven-3.9.9-bin.tar.gz

sudo tar xvf apache-maven-3.9.9-bin.tar.gz

ls -ltr

cd

echo "export M2\_HOME=/opt/apache-maven-3.9.9" >> .bash\_profile

echo "export M2=$M2\_HOME/bin" >> .bash\_profile

echo "export PATH=$M2:$PATH" >> .bash\_profile

source ~/.bash\_profile

echo $M2\_HOME

* **Check versions of installed tools:**

git --version; java --version; /opt/apache-maven-3.9.9/bin/mvn --version

=============

* **Install Jenkins:**

sudo wget -O /etc/yum.repos.d/jenkins.repo \

https://pkg.jenkins.io/redhat-stable/jenkins.repo

sudo rpm --import https://pkg.jenkins.io/redhat-stable/jenkins.io-2023.key

sudo yum upgrade

sudo yum install -y jenkins

sudo systemctl daemon-reload

service jenkins start

chkconfig jenkins on

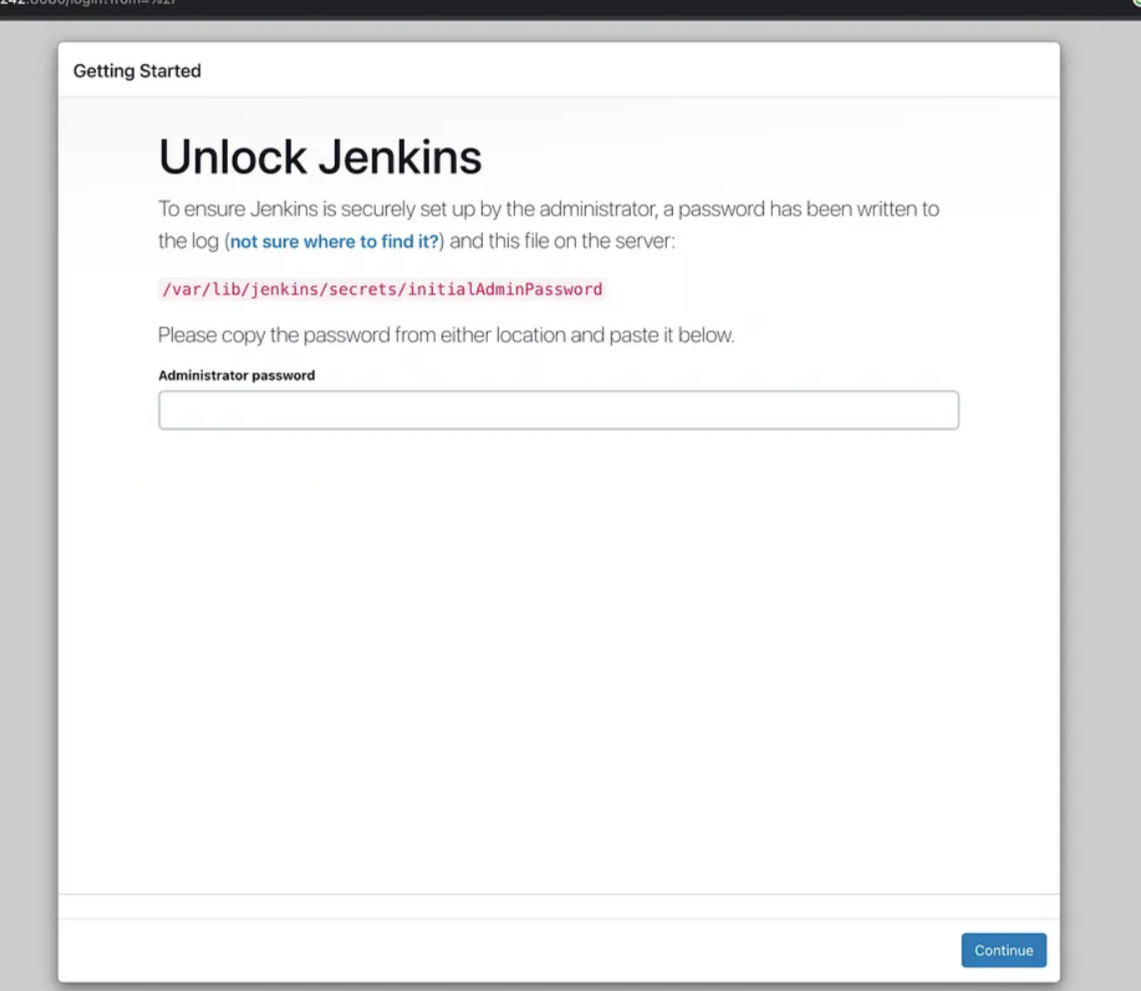
***Once the Jenkins installed on machine, follow the steps below to configure the Jenkins.***

* Open google chrome and type 15.206.165.149:8080 (Public IP address of your ec2 instance)

***Note : In your ec2 instance security group you have to whitelist the Port 8080 because Jenkins use Port 8080***

***Ensure your security groups are configured to allow traffic through port 8080. This is the default port Jenkins uses.***

* You will see a setup wizard and be prompted to enter the administrator password.

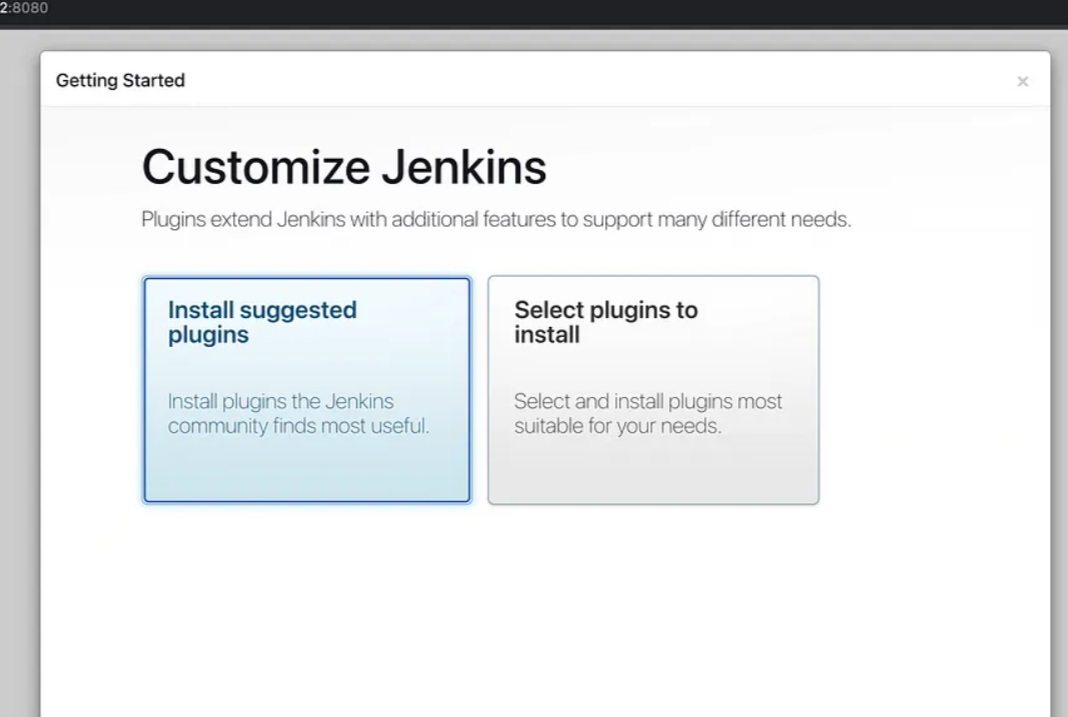


Retrieve the password with the following command:

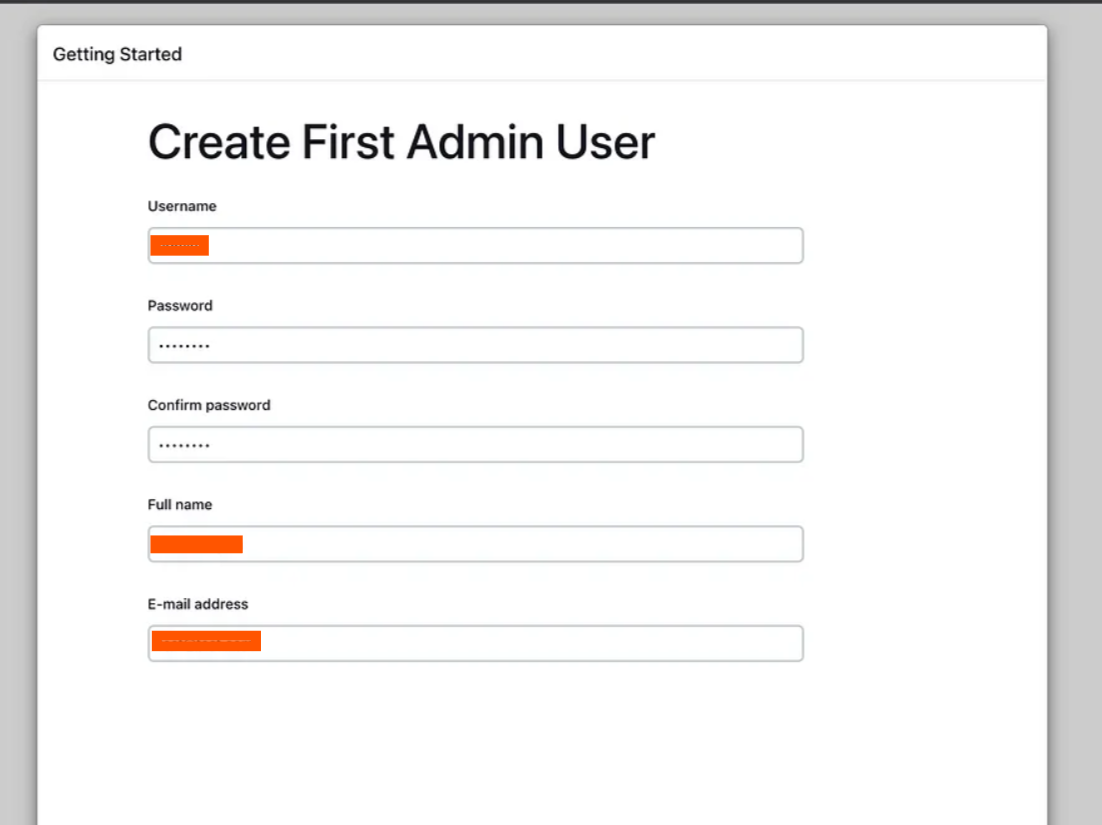
sudo cat /var/lib/jenkins/secrets/initialAdminPassword

Copy the password and paste it into the Jenkins setup wizard to continue.

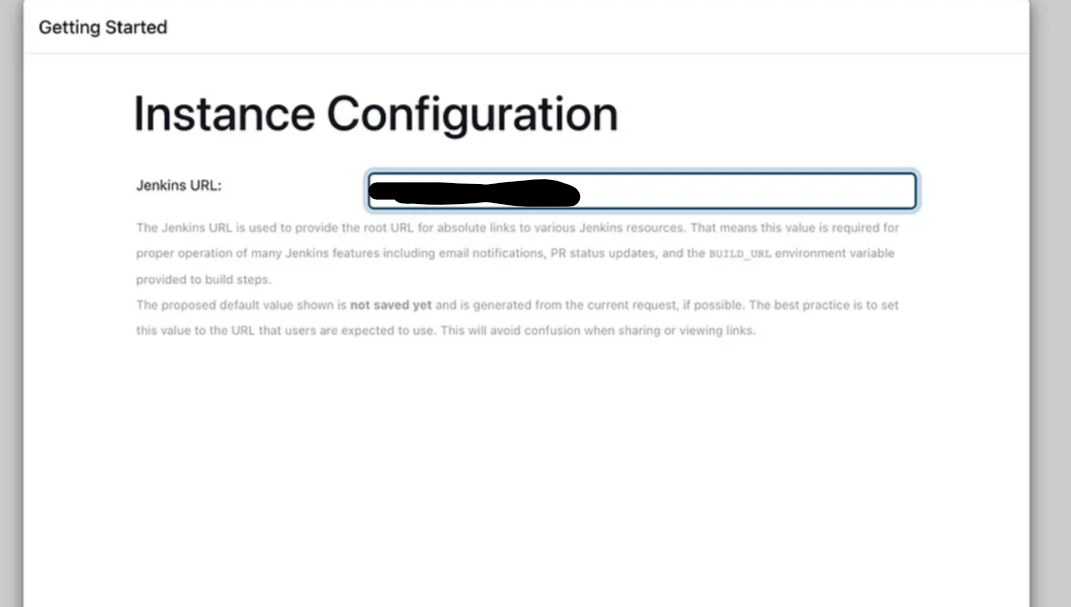
* Once done click on the install Plugins.



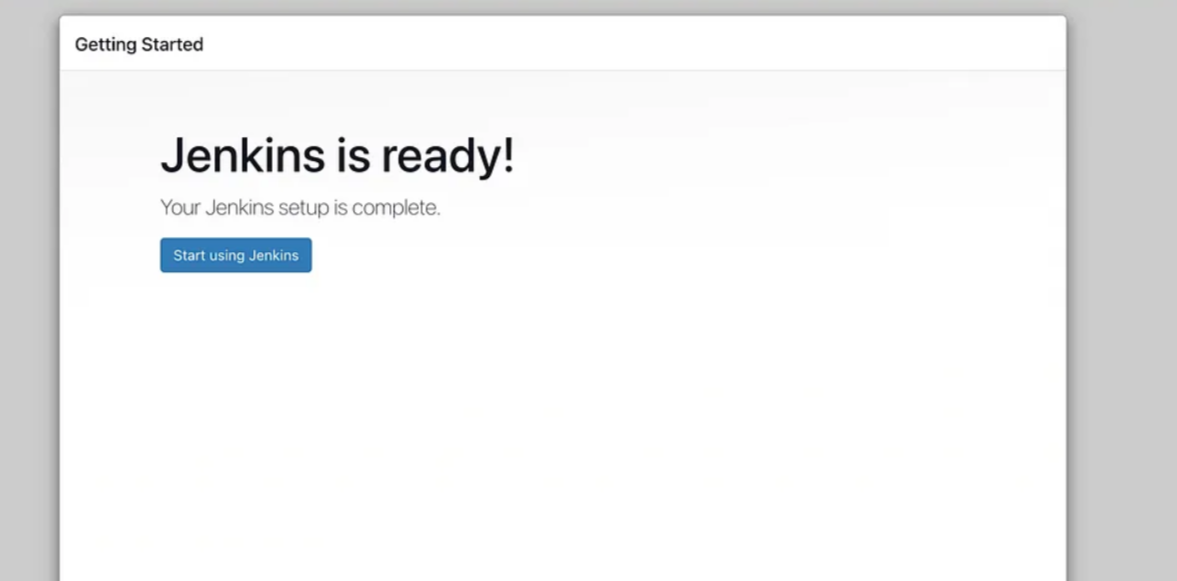
* Create the Admin user and click on save and continue.



* Confirm the Jenkins URL.  It should be set to your instance’s IP.



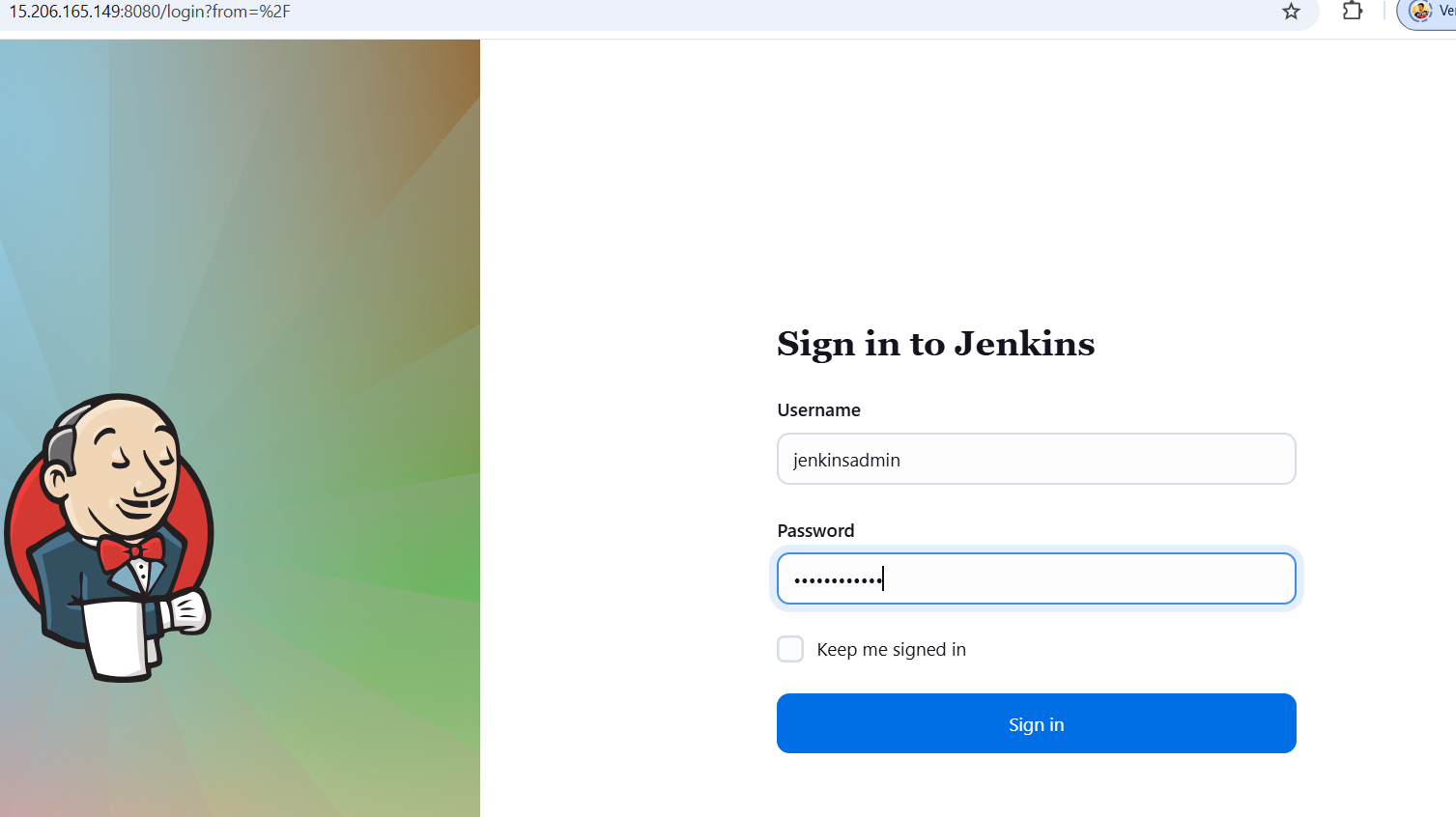
* Once done, start using Jenkins.



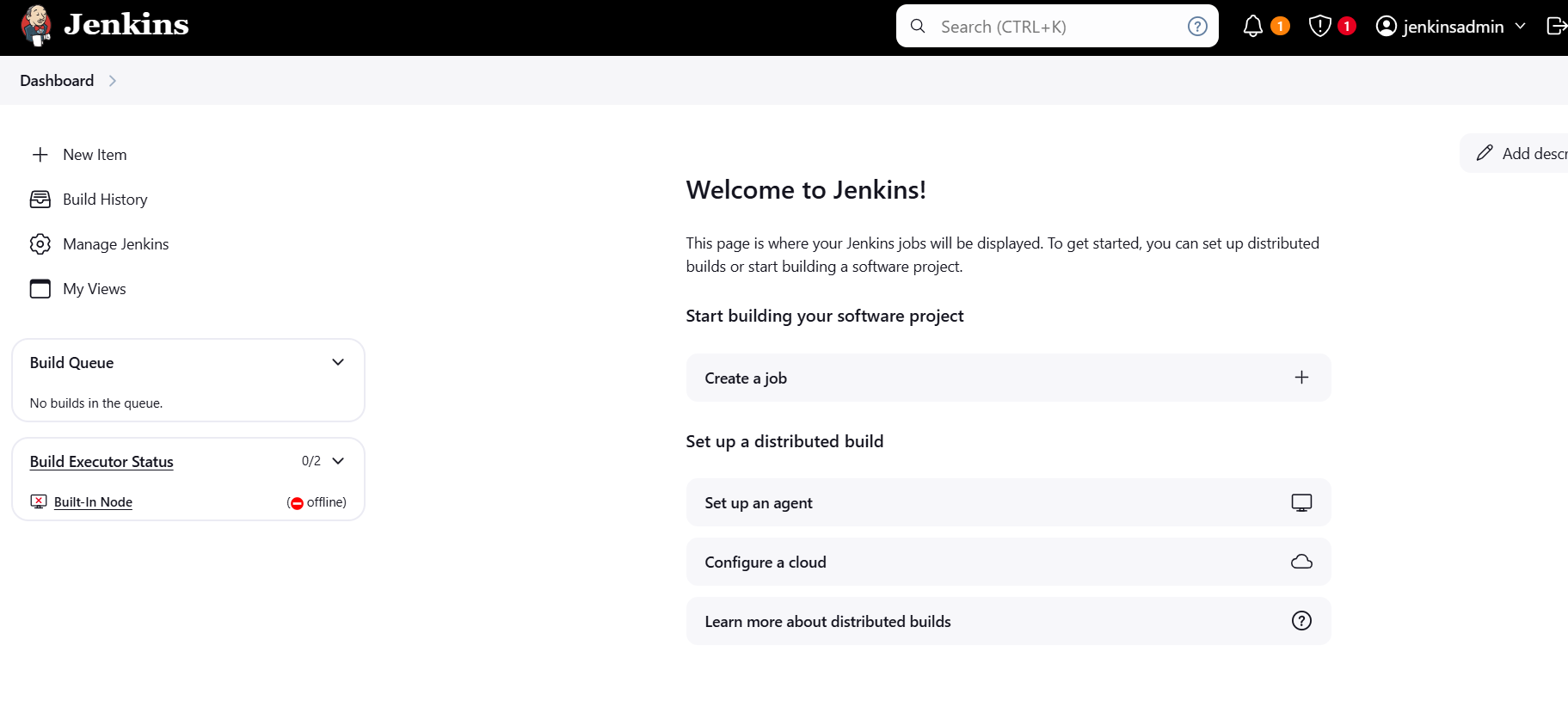
***Once the initial configuration done, you have to setup path for Java, Git and Maven in Jenkins.***

***Follow the steps below for the same.***

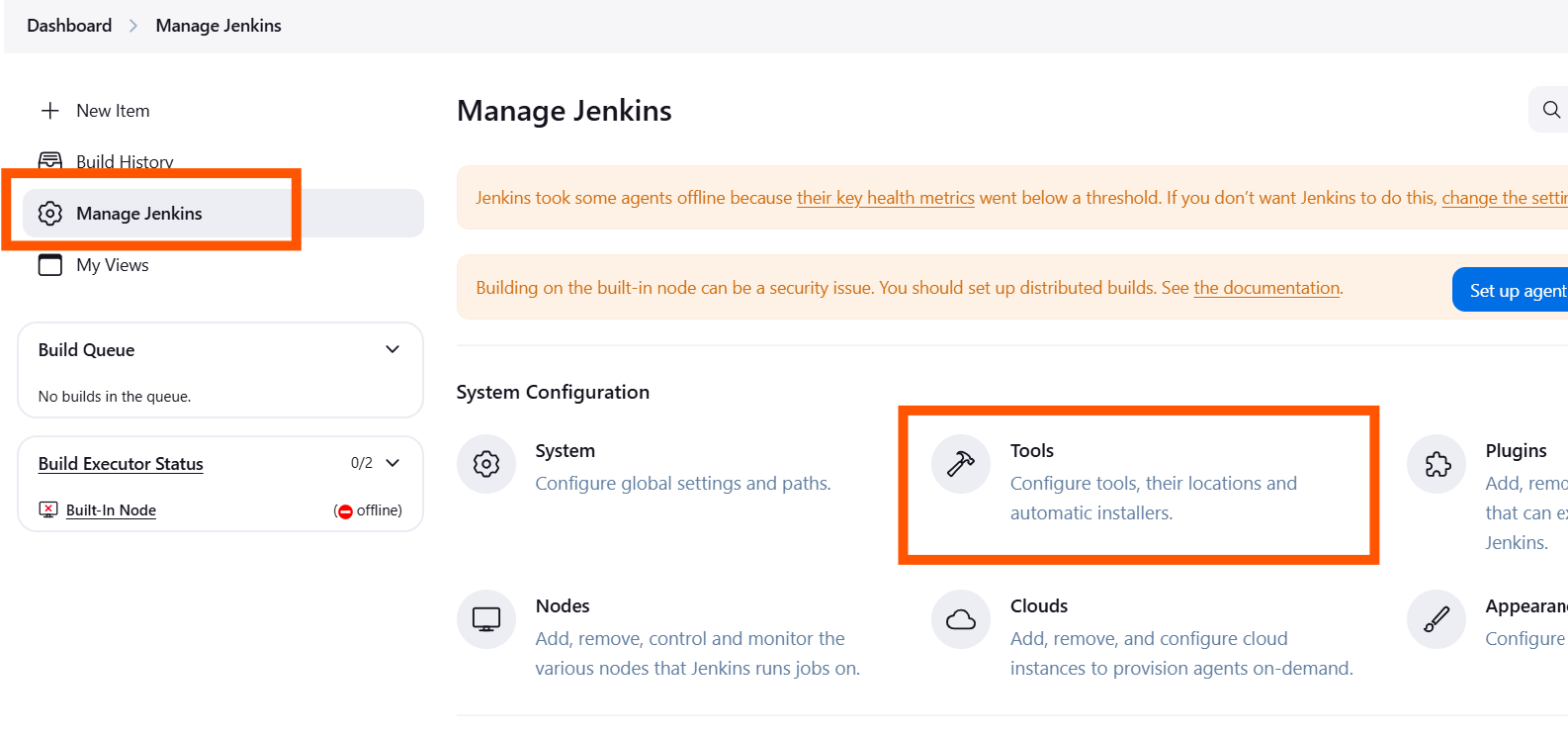
* **Login to Jenkins with username and password.**

****

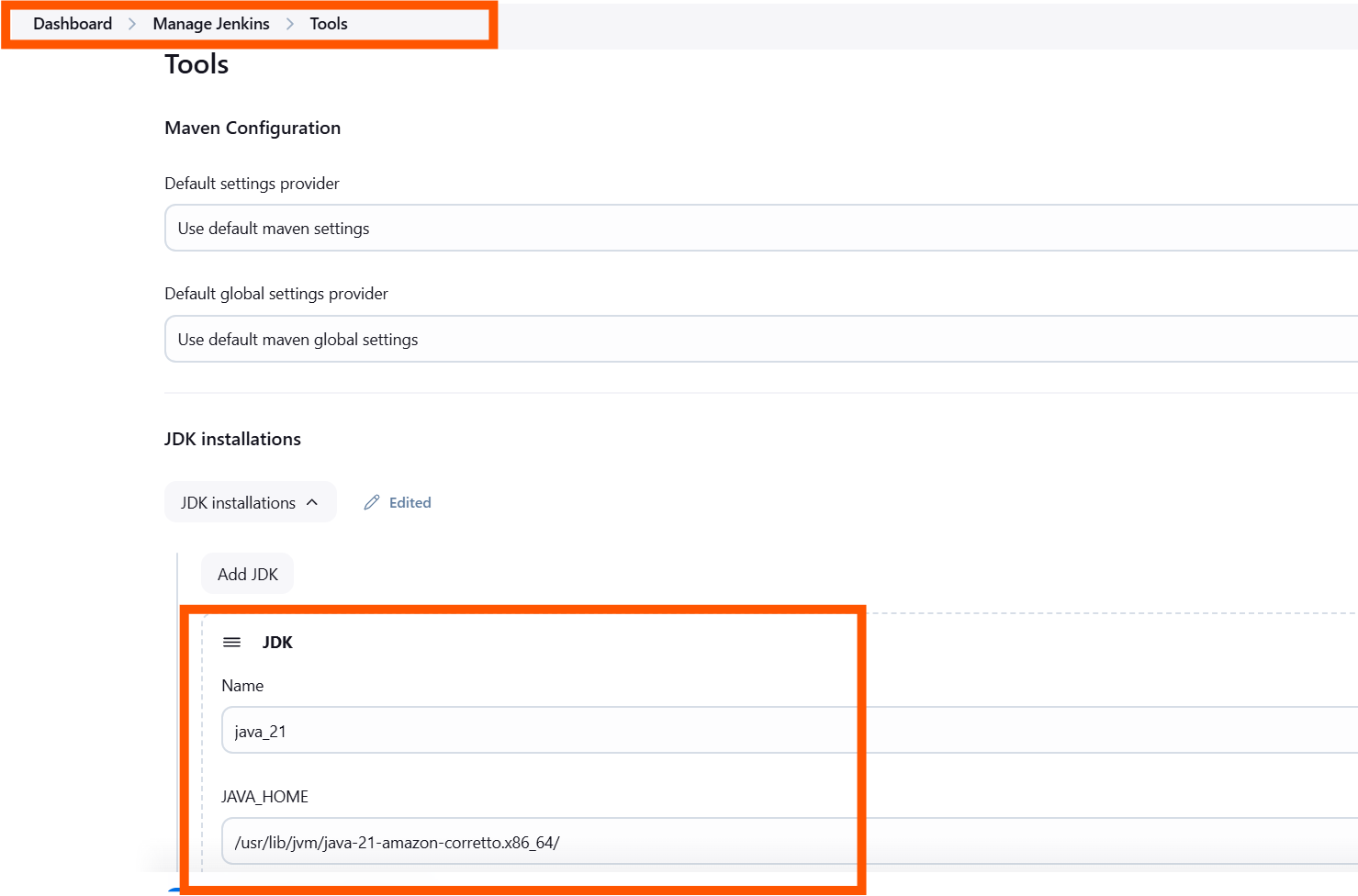
* Once you log in, Jenkins starting page will appear.

****

* Click on manage Jenkins and in System Configuration, Click on Tools.



* You need to specify the Java, Git, Maven installation path in tool.



Java : /usr/lib/jvm/java-21-amazon-corretto.x86\_64/

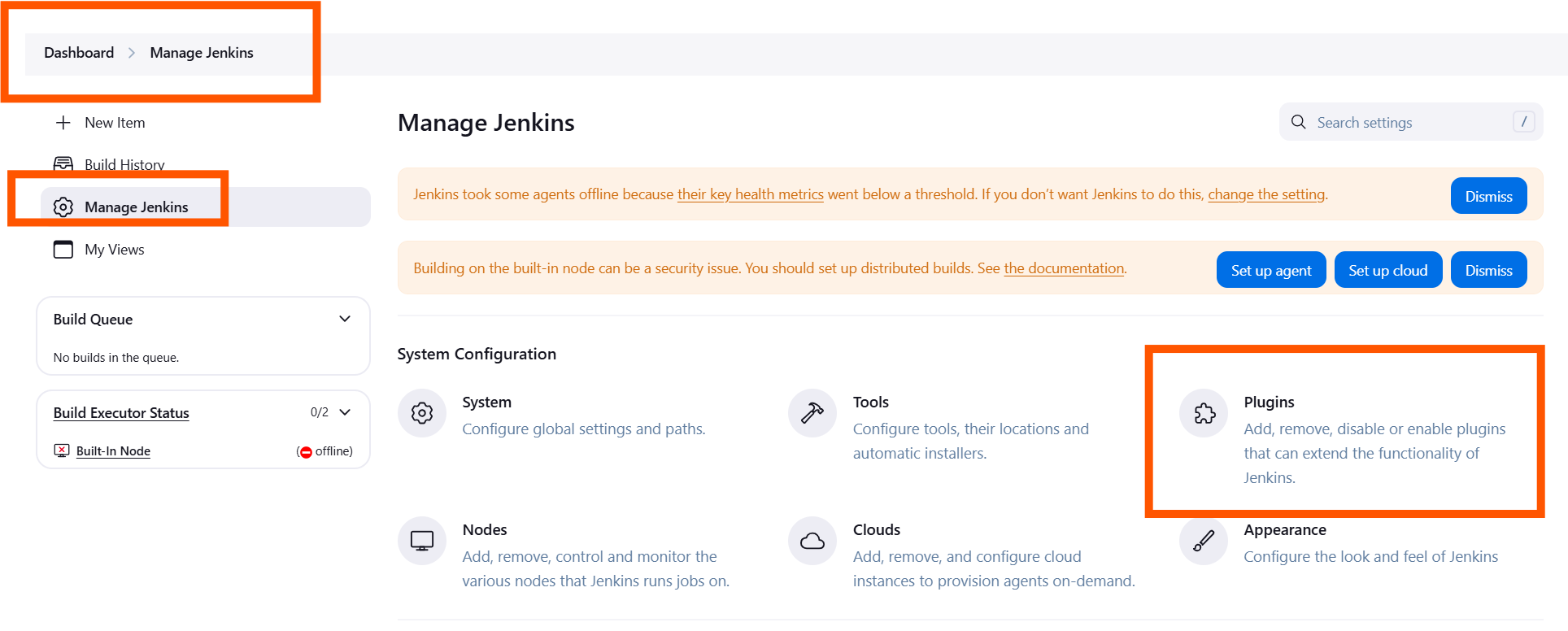
Git : /usr/bin/git

Maven : /opt/apache-maven-3.9.9

Once done click on save button.

* Install Plugins:

You can also install plugin in Jenkins as per your requirements.

****

**CI: (Continuous Integration):**

Continuous Integration (CI) in DevOps is the practice of automatically integrating code changes into a shared repository multiple times a day. Developers commit changes to their **local repository** and then **push** the changes to the **remote repository** (e.g., GitHub, GitLab). A **webhook** is set up in the remote repo to trigger a **Jenkins** job when code is pushed. Jenkins then pulls the latest code, and based on the **pom.xml** configuration, it runs a build to create a **JAR** or **WAR** file. This helps ensure that code is frequently tested and built, catching issues early in the development process.

**Example:**

Let's suppose I am a developer. I will create a feature or local branch from the dev branch, pull the latest changes from the dev branch, and start writing code in my local branch. Once I'm done writing the code, I will compile, package, and test it on my local system using maven if I am writing java code. Once testing is complete, I will push the code to the central repository (GitHub) from my local system.

Whenever I commit the code, the Jenkins pipeline will automatically trigger. It will clone the source code and fetch the code from the central repository (GitHub), then perform CPR (code compile, package, and run the code). Once done, an artifact will be created based on what we’ve specified in the pom.xml (either WAR or JAR). If we have specified a WAR file to be created in the pom.xml, then a WAR artifact will be created and stored in the artifact repository (such as Nexus or JFrog).

**Benefits of CI (Continuous Integration):**

 Early Bug Detection:  
Since code is integrated frequently, CI helps identify issues early in the development cycle, reducing the cost and time needed to fix them later.

 Improved Code Quality:  
Automated builds and tests ensure that code is always in a working state, leading to fewer defects in production and better overall code quality.

 Faster Development Cycle:  
Continuous integration speeds up the development process by automating repetitive tasks like builds, testing, and deployment. This allows developers to focus more on writing code and less on manual processes.

 Faster Feedback:  
Developers receive quick feedback on their changes, which helps them fix issues quickly and ensures smoother collaboration across teams.

 Consistent Environments:  
CI ensures that all code is built and tested in the same automated environment, eliminating the "it works on my machine" problem and providing more consistency across development, staging, and production environments.

 Automated Deployment:  
CI often integrates with automated deployment tools, reducing the time and effort required for manual deployment processes and enabling more frequent and reliable releases.

 Increased Collaboration:  
With frequent code merges and automated testing, team members are encouraged to work together more often, improving collaboration and communication.