Jenkins Essentials: A Beginner's Guide to Fundamentals and Installation

What is Jenkins?

- **Jenkins** is a Continuous Integration (CI) tool that automates the process of pulling code from remote repositories and running tests on it.
- **Continuous Integration (CI)** refers to the practice of integrating development and testing activities together, ensuring smooth collaboration between developers and testers.

CI Process in Detail

1. Development and Testing Cycle:

- **Developers** push their code daily to a remote repository.
- The **DevOps team** builds the code overnight and provides a new build to testers for the next day.
- This cycle **repeats continuously**.

2. Automation in CI:

- **Automation testers** also update and push their test scripts to a remote repository daily.
- **Jenkins** automatically pulls the latest code from these repositories and executes the automated tests on it.

What is CD (Continuous Delivery/Deployment)?

1. Definition of Continuous Delivery (CD):

- **Continuous Delivery (CD)** is the process where software is continuously deployed to the production environment after testing.
- It ensures that once testing is complete, the code is released quickly to the customer.

2. Difference Between CI and CD:

- **CI (Continuous Integration)**: Focuses on continuous development and testing in local environments.
- **CD (Continuous Deployment)**: Focuses on deploying the code to the production environment after testing.

Role of Jenkins in CI/CD

Jenkins in Automation:

- **Developers and testers** each maintain their own GitHub repositories.
- **Jenkins** pulls the latest code from the **developer's GitHub repository** to create a build. This is typically the application code that needs to be tested.
- Once the build is created, Jenkins then pulls the **tester's GitHub repository**, which contains the **automated test scripts**.
- Jenkins executes these **automated tests** on the **testing environment**, ensuring that the code is continuously tested against the latest changes.

Packaging and Certification:

- Once Jenkins pulls the **latest code** from the developer's repository and creates a build, it then **packages** the software (preparing it for deployment).
- After that, Jenkins pulls the **automated test scripts** from the tester's repository and executes them on the build.
- If the **automation tests pass**, the build is considered **certified** and ready for deployment to the production environment.

Let's Understand What a Pipeline in Jenkins Is

A Jenkins pipeline automates the entire process of software development and testing. When developers make changes, Jenkins builds the code, while automation testers ensure their test scripts are executed. This cycle includes building, testing, packaging, and deployment. CI/CD ensures continuous integration and testing, making the process seamless and efficient for both development and testing teams.

Now, let’s look at how this pipeline brings developers and testers together:

- **The Developer's Role:** The developer pushes their latest code changes to a GitHub repository. Once that happens, Jenkins steps in and pulls the latest version of the code to begin building the application.
- **The Tester's Role:** Meanwhile, testers have their automated test scripts stored in a separate repository. After Jenkins finishes building the code, it retrieves these test scripts, runs the tests, and checks whether the new changes meet the required standards.

In this way, developers and testers collaborate effortlessly, ensuring that the software is continuously built, tested, and ready for deployment.

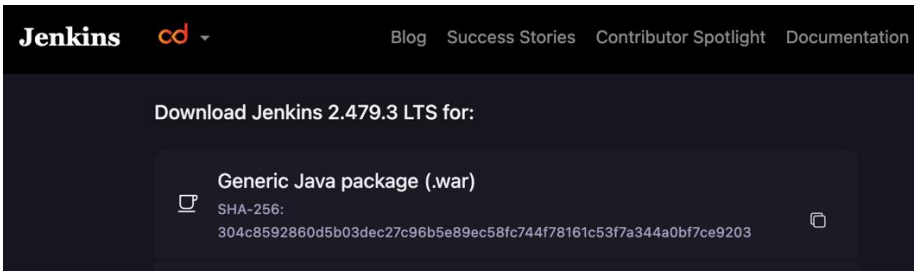
Jenkins' Headless Testing

- **Jenkins**, by default, follows **headless testing**, which means it runs tests without the need for a **GUI**.
- This is efficient for running **automation scripts** in a CI/CD pipeline.

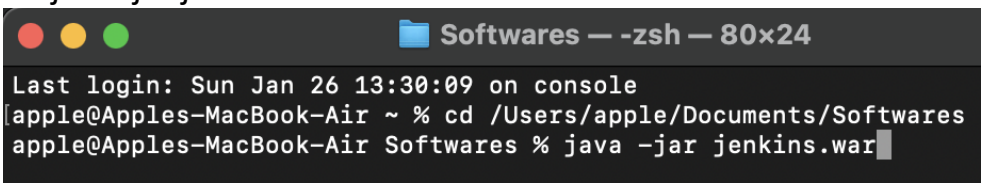
Jenkins Installation Guide

Steps:

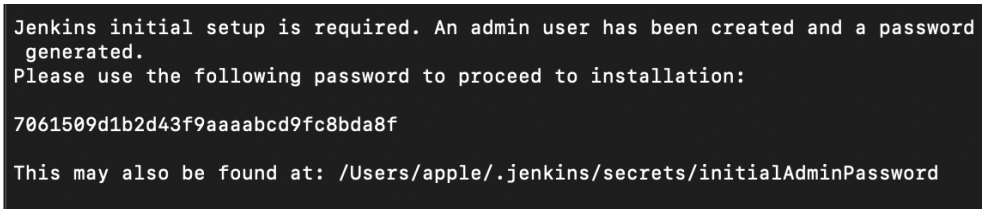
1. **Download Jenkins WAR File**:
   - Go to the Jenkins download page and download the **Jenkins WAR** file.



2. **Open Terminal and Navigate to Jenkins Location and Run Jenkins**:
   - Open **Terminal** and navigate to the folder where the **Jenkins.war** file is located.
   - Use the following command to start Jenkins:
   - java -jar jenkins.war



3. **Password Generation**:
   - After running the command for the first time, Jenkins will generate a temporary password in the console.
   - If you miss the password, you can retrieve it from the location mentioned in the console where it says, "This may also be found at...".



Note: The password is used to unlock Jenkins on your local system.

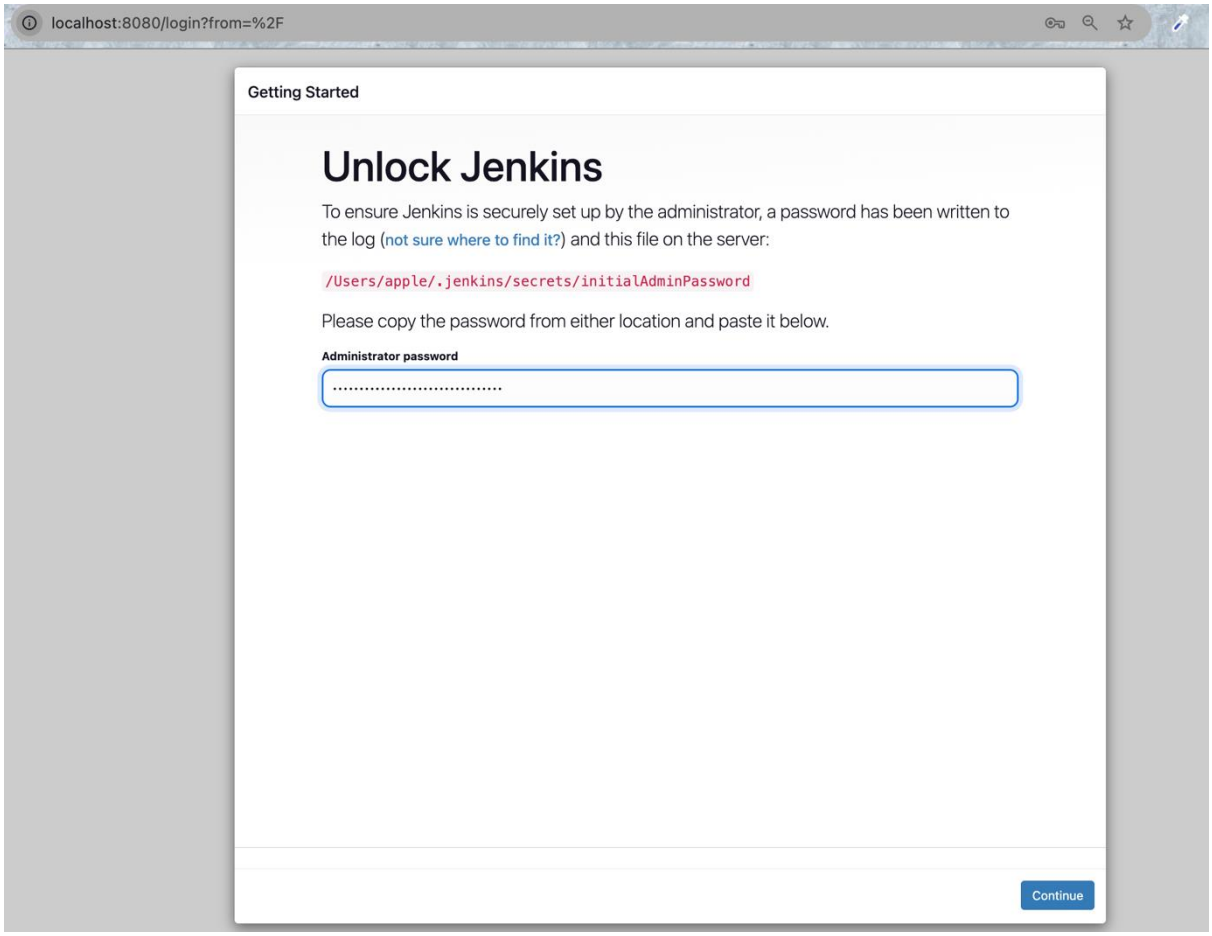
4. **Check If Jenkins Is Running**:
    - Once Jenkins is up and running, the terminal will display the message confirming it.

```
jenkins.InitReactorRunner$1#onAttained: Completed initialization
hudson.lifecycle.Lifecycle#onReady: Jenkins is fully up and running
h.m.DownloadService$Downloadable#load: Obtained the updated data file for hudson.tasks.Maven.MavenInstaller
hudson.util.Retrier#start: Performed the action check updates server successfully at the attempt #1
```

    - Jenkins runs on **localhost:8080** by default.
5. **Access Jenkins in Browser**
    - Open your web browser and go to:
    - http://localhost:8080

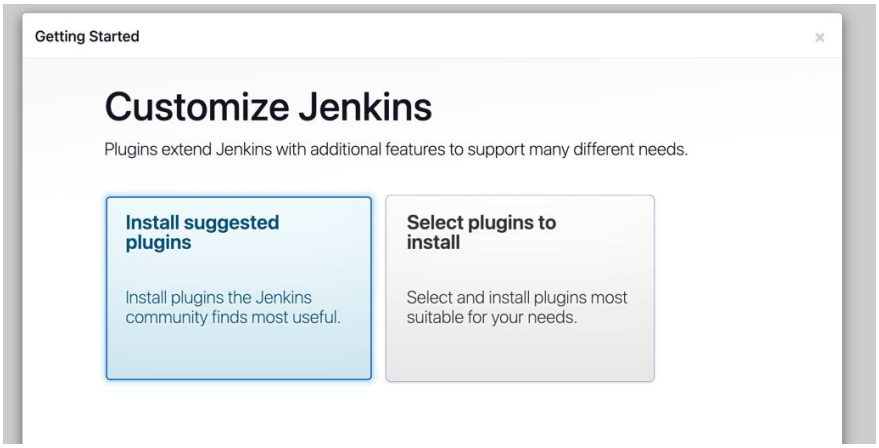
Unlock Jenkins

- You will be prompted to enter the password you found in the terminal.
- Provide the password to unlock Jenkins.

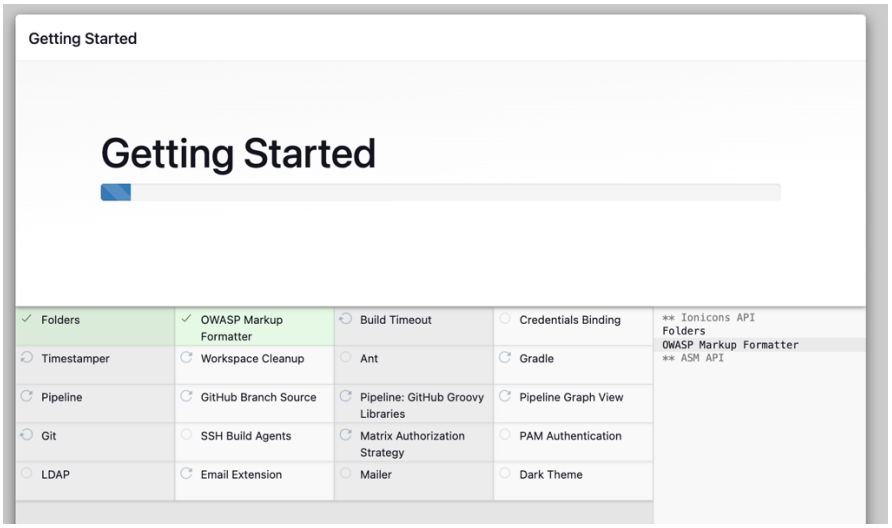


Install Suggested Plugins

- After unlocking Jenkins, click on **Install suggested plugins**.

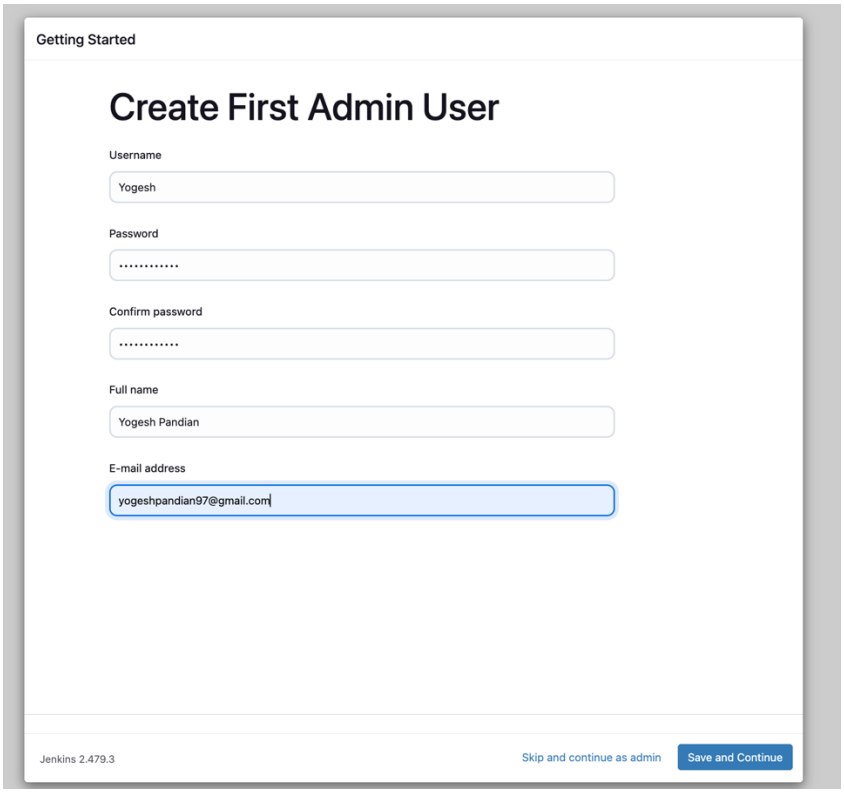


- Wait for Jenkins to install all the required basic plugins.



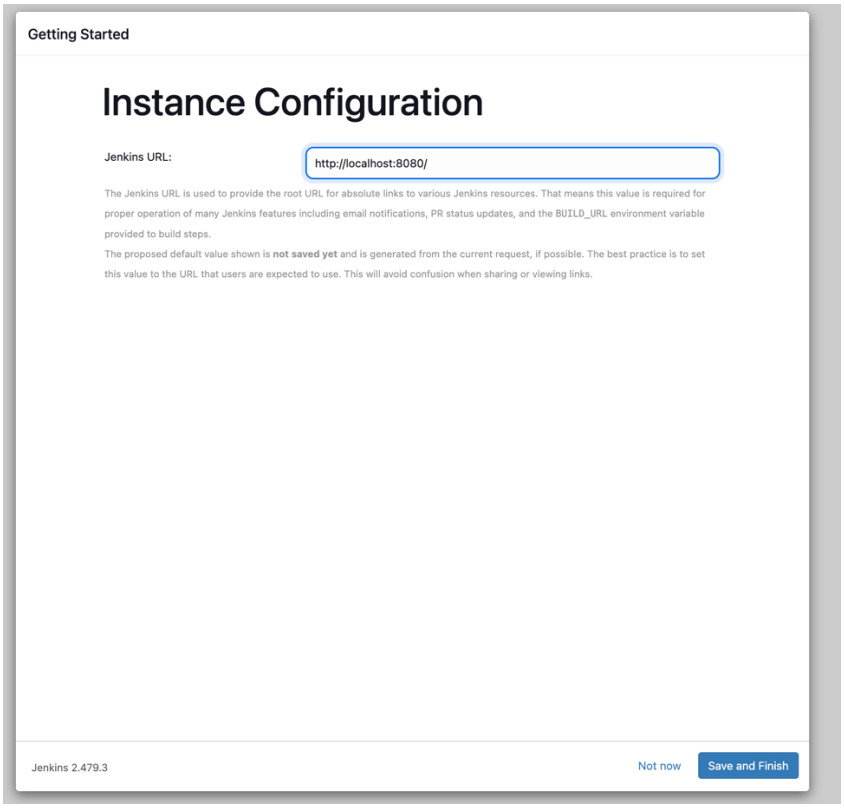
6. Set Up Jenkins Admin User

- After the plugins are installed, Jenkins will redirect you to a page where you need to set up your **username**, **password**, and **email address**.

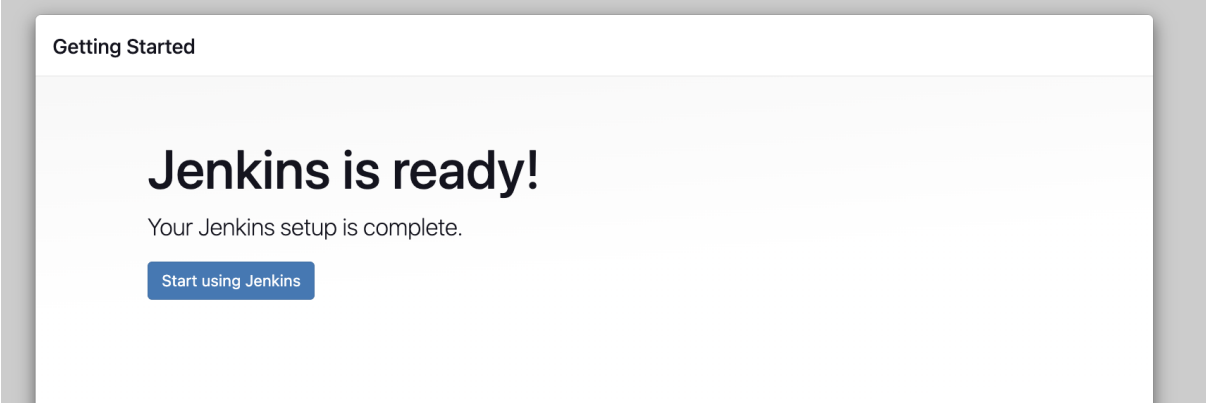


7. Complete Setup

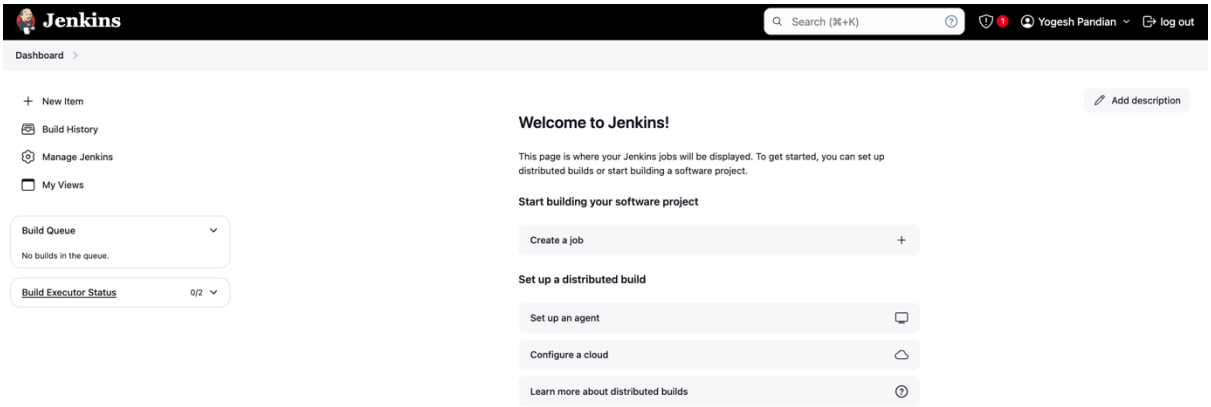
- After completing the setup, Jenkins will be ready to use.
- Bookmark the **localhost:8080** URL for easy access later.



Click on Start using Jenkins it will open the Jenkins Dashboard



Jenkins Dashboard



Only the first time you set up Jenkins, it will provide a password to unlock it. After that, you can start Jenkins by running the following command in the terminal from the directory where the jenkins.war file is located:

java -jar jenkins.war

Then, open the localhost server (usually http://localhost:8080) and provide your username and password to access Jenkins.

Using Jenkins with .war File

Every Time You Start Jenkins:

- Open your **Terminal** and go to the directory where the jenkins.war file is located.
- Run the following command:

```
java -jar jenkins.war
```

Note: If you close the terminal, the Jenkins process will stop. So, make sure the terminal stays open while you are working with Jenkins in this mode.

Using Jenkins with Executable File

Every Time You Start Jenkins:

- Double-click the Jenkins installer (executable file) on your system.
- Jenkins will automatically start and run in the background.

Note: Unlike the .war file, when using the executable file, Jenkins will continue running in the background even after closing the terminal or the command window.
