DevOps Lab

Assignment 4:

Aim: To understand Continuous Integration & Continuous Delivery with the help of tools like Jenkins and to also install Jenkins on our Operating System and Create a Simple Job to build a Java Project on Local Machine & also use a GitHub repository code to build it.

Theory:

In software engineering, **continuous integration (CI)** is the practice of merging all developers' working copies to a shared mainline several times a day.

Continuous delivery (CD) is a software engineering approach in which teams produce software in short cycles, ensuring that the software can be reliably released at any time and, when releasing the software, without doing so manually.

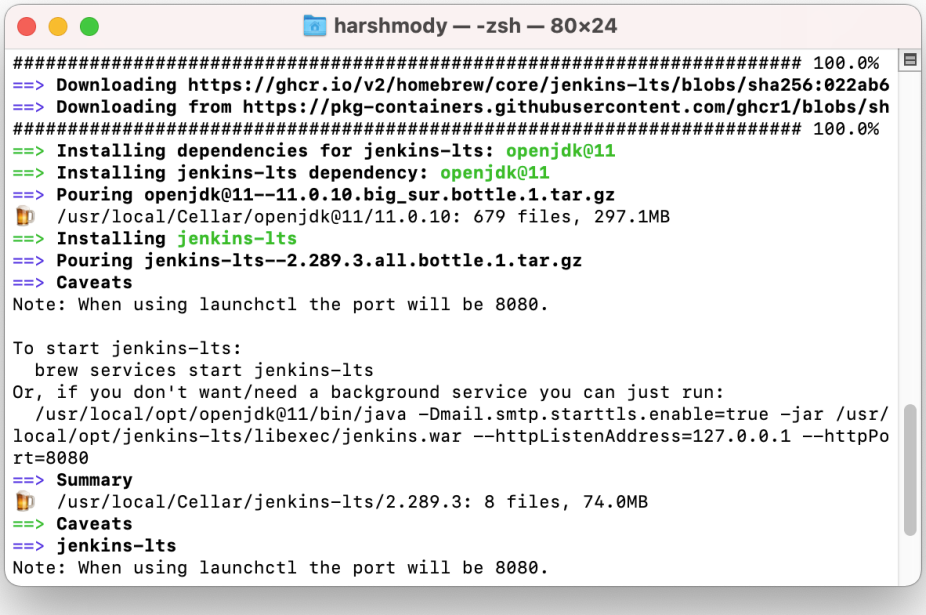
Jenkins is an open source automation server. With Jenkins, organizations can accelerate the software development process by automating it. Jenkins manages and controls software delivery processes throughout the entire lifecycle, including build, document, test, package, stage, deployment, static code analysis and much more.

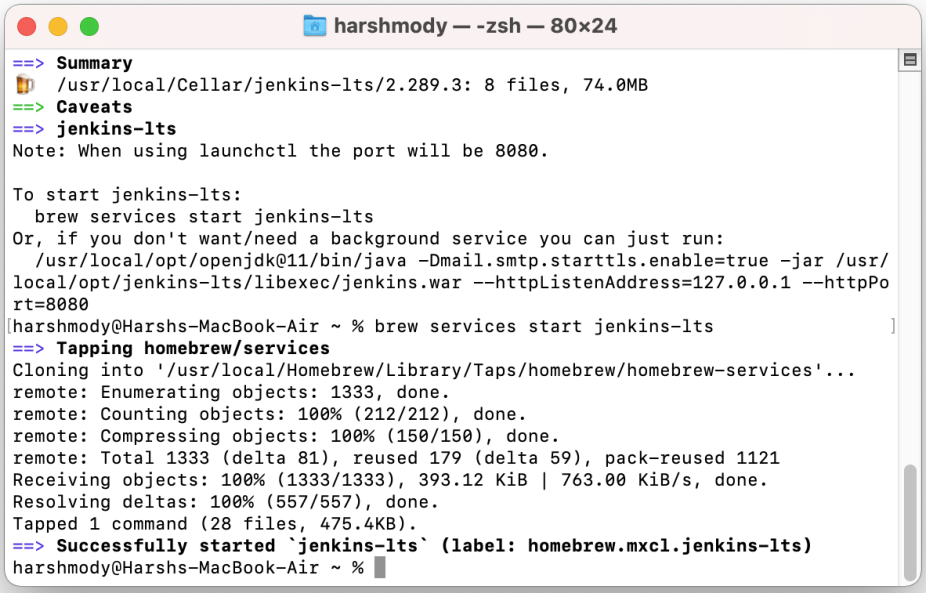
You can set up Jenkins to watch for any code changes in places like GitHub, Bitbucket or GitLab and automatically do a build a with tools like Maven and Gradle. You can utilize container technology such as Docker and Kubernetes, initiate tests and then take actions like rolling back or rolling forward in production.

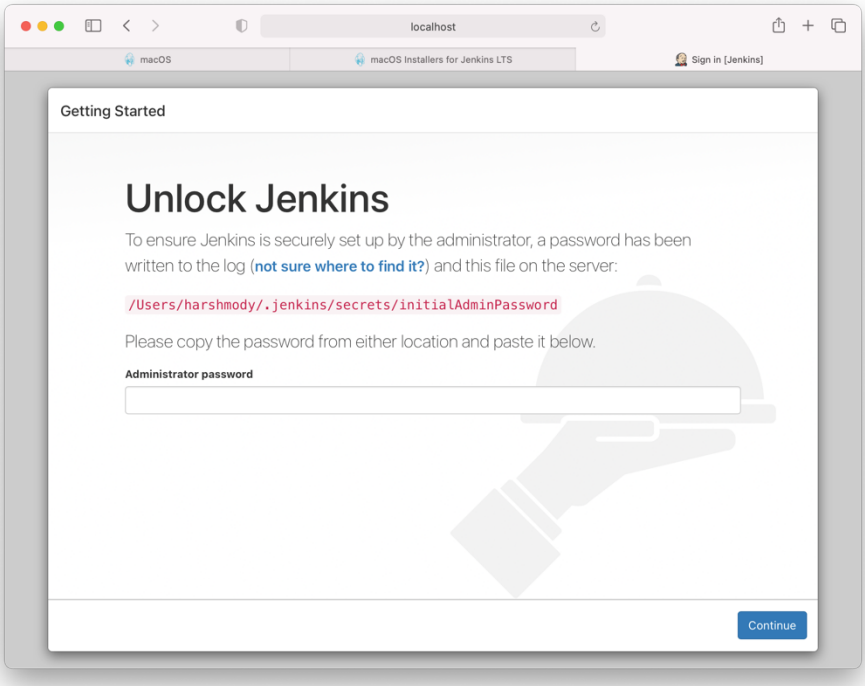
Originally developed by Kohsuke for continuous integration (CI), today Jenkins orchestrates the entire software delivery pipeline – called continuous delivery. For some organizations automation extends even further, to continuous deployment. Continuous delivery (CD), coupled with a DevOps culture, dramatically accelerates the delivery of software.

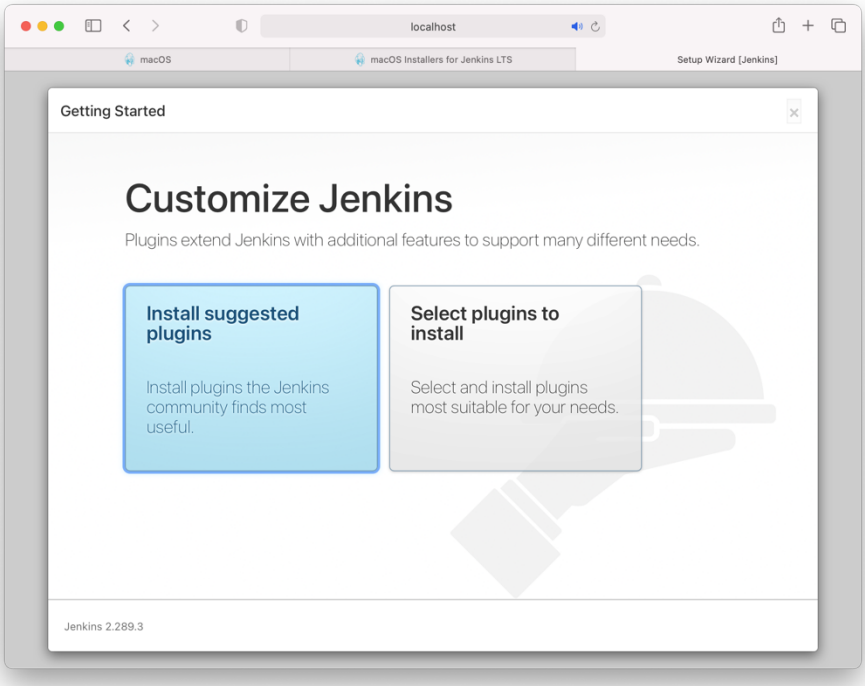
Jenkins is the most widely adopted solution for continuous delivery, thanks to its extensibility and a vibrant, active community. The Jenkins community offers more than 1,700 plugins that enable Jenkins to integrate with virtually any tool, including all of the best-of-breed solutions used throughout the continuous delivery process. Jenkins continues to grow as the dominant solution for software process automation, continuous integration and continuous delivery and, as of February 2018, there are more than 165,000 active installations and an estimated 1.65 million users around the world.

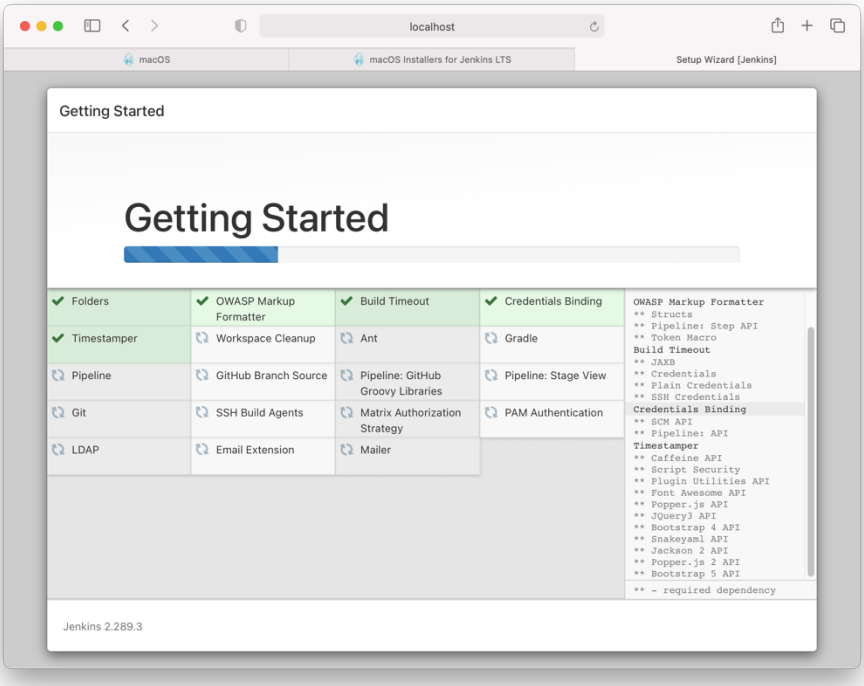
Installation of Jenkins using Homebrew on Mac:

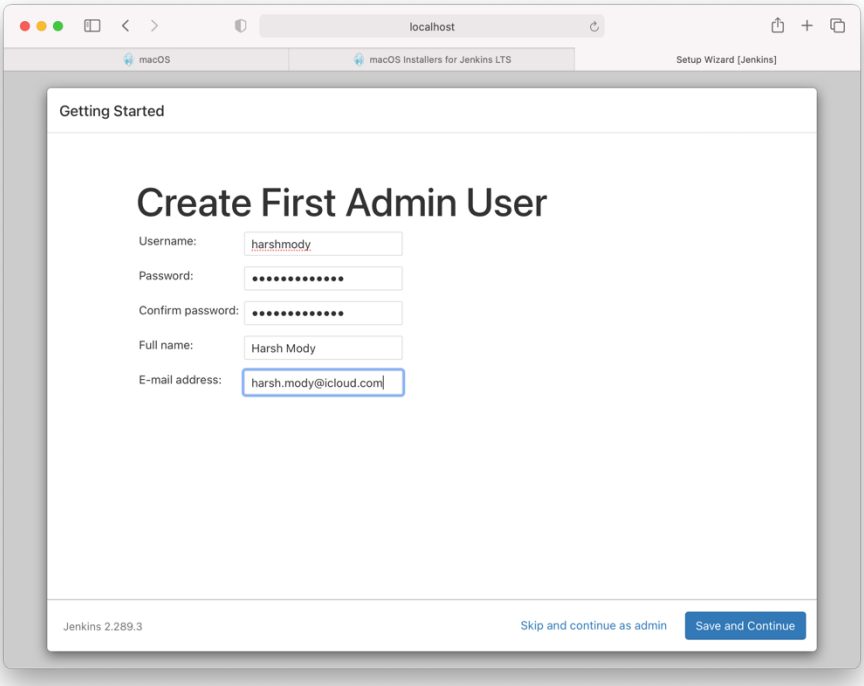

harshmody — -zsh — 80×24
100.0%
==> Downloading https://ghcr.io/v2/homebrew/core/jenkins-lts/blobs/sha256:022ab6
==> Downloading from https://pkg-containers.githubusercontent.com/ghcr1/blobs/sh
100.0%
==> Installing dependencies for jenkins-lts: openjdk@11
==> Installing jenkins-lts dependency: openjdk@11
==> Pouring openjdk@11--11.0.10.big_sur.bottle.1.tar.gz
/usr/local/Cellar/openjdk@11/11.0.10: 679 files, 297.1MB
==> Installing jenkins-lts
==> Pouring jenkins-lts--2.289.3.all.bottle.1.tar.gz
==> Caveats
Note: When using launchctl the port will be 8080.
To start jenkins-lts:
brew services start jenkins-lts
Or, if you don't want/need a background service you can just run:
/usr/local/opt/openjdk@11/bin/java -Dmail.smtp.starttls.enable=true -jar /usr/
local/opt/jenkins-lts/libexec/jenkins.war --httpListenAddress=127.0.0.1 --httpPo
rt=8080
==> Summary
/usr/local/Cellar/jenkins-lts/2.289.3: 8 files, 74.0MB
==> Caveats
==> jenkins-lts
Note: When using launchctl the port will be 8080.

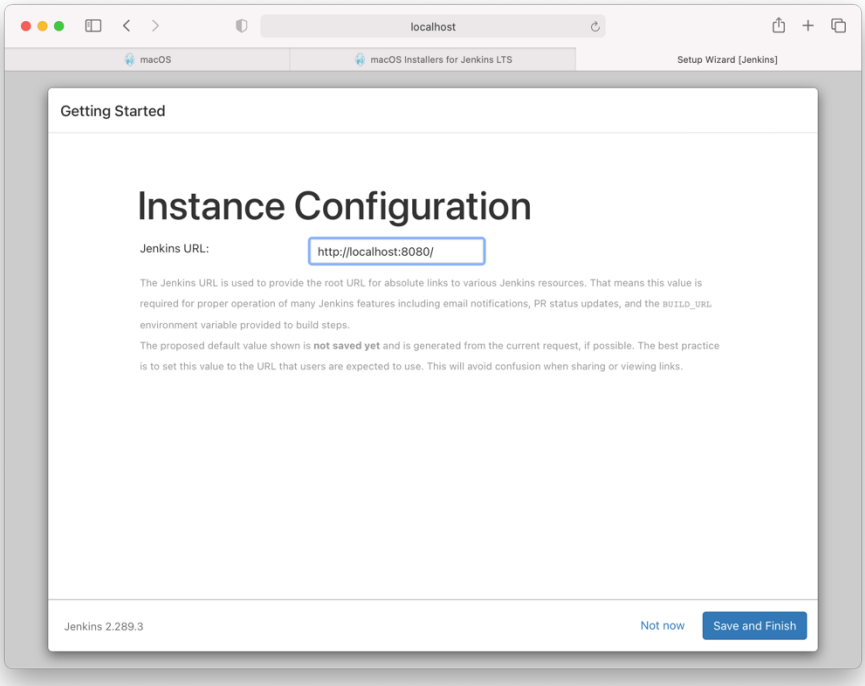

harshmody — -zsh — 80×24
==> Summary
/usr/local/Cellar/jenkins-lts/2.289.3: 8 files, 74.0MB
==> Caveats
==> jenkins-lts
Note: When using launchctl the port will be 8080.
To start jenkins-lts:
brew services start jenkins-lts
Or, if you don't want/need a background service you can just run:
/usr/local/opt/openjdk@11/bin/java -Dmail.smtp.starttls.enable=true -jar /usr/
local/opt/jenkins-lts/libexec/jenkins.war --httpListenAddress=127.0.0.1 --httpPo
rt=8080
harshmody@Harshs-MacBook-Air ~ % brew services start jenkins-lts
==> Tapping homebrew/services
Cloning into '/usr/local/Homebrew/Library/Taps/homebrew/homebrew-services'...
remote: Enumerating objects: 1333, done.
remote: Counting objects: 100% (212/212), done.
remote: Compressing objects: 100% (150/150), done.
remote: Total 1333 (delta 81), reused 179 (delta 59), pack-reused 1121
Receiving objects: 100% (1333/1333), 393.12 KiB | 763.00 KiB/s, done.
Resolving deltas: 100% (557/557), done.
Tapped 1 command (28 files, 475.4KB).
==> Successfully started `jenkins-lts` (label: homebrew.mxcl.jenkins-lts)
harshmody@Harshs-MacBook-Air ~ %

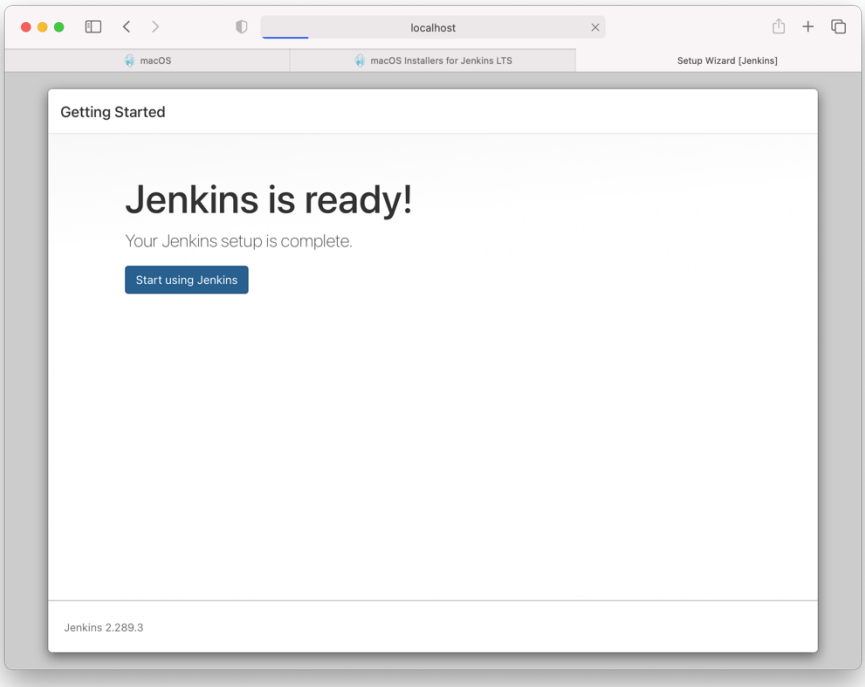

localhost
macOS
macOS Installers for Jenkins LTS
Sign in [Jenkins]
Getting Started
Unlock Jenkins
To ensure Jenkins is securely set up by the administrator, a password has been written to the log (not sure where to find it?) and this file on the server:
/Users/harshmody/.jenkins/secrets/initialAdminPassword
Please copy the password from either location and paste it below.
Administrator password
Continue

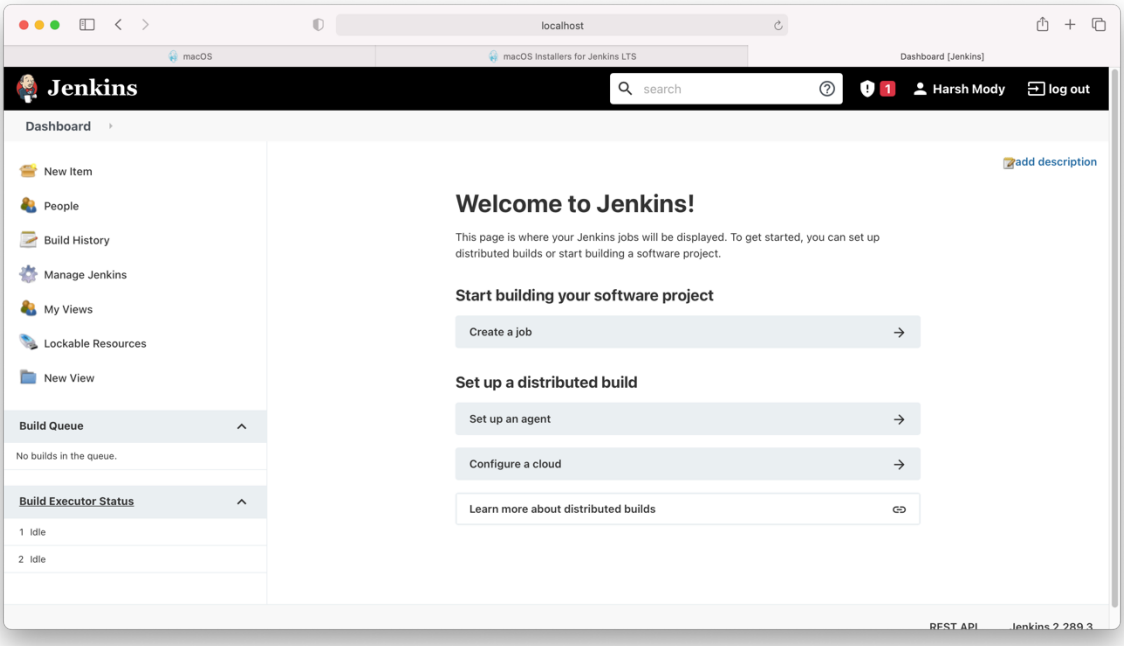

localhost
macOS
macOS Installers for Jenkins LTS
Setup Wizard [Jenkins]
Getting Started
Customize Jenkins
Plugins extend Jenkins with additional features to support many different needs.
Install suggested plugins
Install plugins the Jenkins community finds most useful.
Select plugins to install
Select and install plugins most suitable for your needs.
Jenkins 2.289.3


localhost
macOS
macOS Installers for Jenkins LTS
Setup Wizard [Jenkins]
Getting Started
Getting Started
Folders
OWASP Markup Formatter
Build Timeout
Credentials Binding
Timestamper
Workspace Cleanup
Ant
Gradle
Pipeline
GitHub Branch Source
Pipeline: GitHub Groovy Libraries
Pipeline: Stage View
Git
SSH Build Agents
Matrix Authorization Strategy
PAM Authentication
LDAP
Email Extension
Mailer
OWASP Markup Formatter
** Structs
** Pipeline: Step API
** Token Macro
Build Timeout
** JAXB
** Credentials
** Plain Credentials
** SSH Credentials
Credentials Binding
** SCM API
** Pipeline: API
Timestamper
** Caffeine API
** Script Security
** Plugin Utilities API
** Font Awesome API
** Popper.js API
** JQuery3 API
** Bootstrap 4 API
** Snakeyaml API
** Jackson 2 API
** Popper.js 2 API
** Bootstrap 5 API
** - required dependency
Jenkins 2.289.3


localhost
macOS
macOS Installers for Jenkins LTS
Setup Wizard [Jenkins]
Getting Started
Create First Admin User
Username:
harshmody
Password:
Confirm password:
Full name:
Harsh Mody
E-mail address:
harsh.mody@icloud.com
Jenkins 2.289.3
Skip and continue as admin
Save and Continue


localhost
macOS
macOS Installers for Jenkins LTS
Setup Wizard [Jenkins]
Getting Started
Instance Configuration
Jenkins URL:
http://localhost:8080/
The Jenkins URL is used to provide the root URL for absolute links to various Jenkins resources. That means this value is required for proper operation of many Jenkins features including email notifications, PR status updates, and the BUILD_URL environment variable provided to build steps.
The proposed default value shown is not saved yet and is generated from the current request, if possible. The best practice is to set this value to the URL that users are expected to use. This will avoid confusion when sharing or viewing links.
Jenkins 2.289.3
Not now
Save and Finish


localhost
macOS
macOS Installers for Jenkins LTS
Setup Wizard [Jenkins]
Getting Started
Jenkins is ready!
Your Jenkins setup is complete.
Start using Jenkins
Jenkins 2.289.3

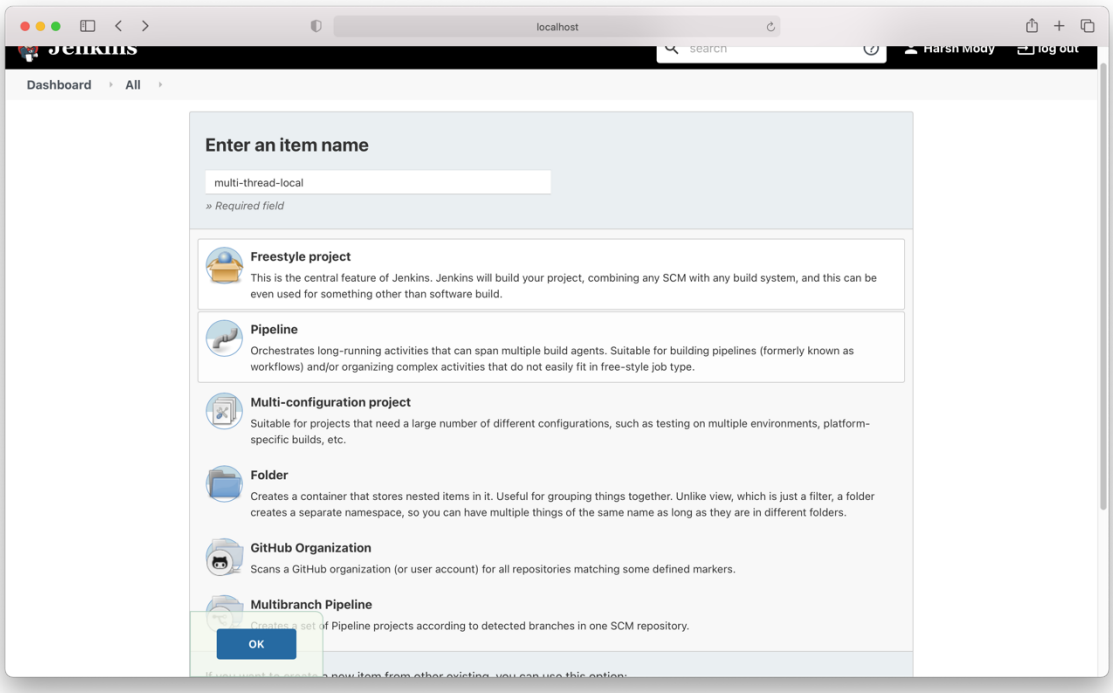


Firstly we need to write a simple Java Program to Build & Run in Jenkins Automation Environment as shown below:

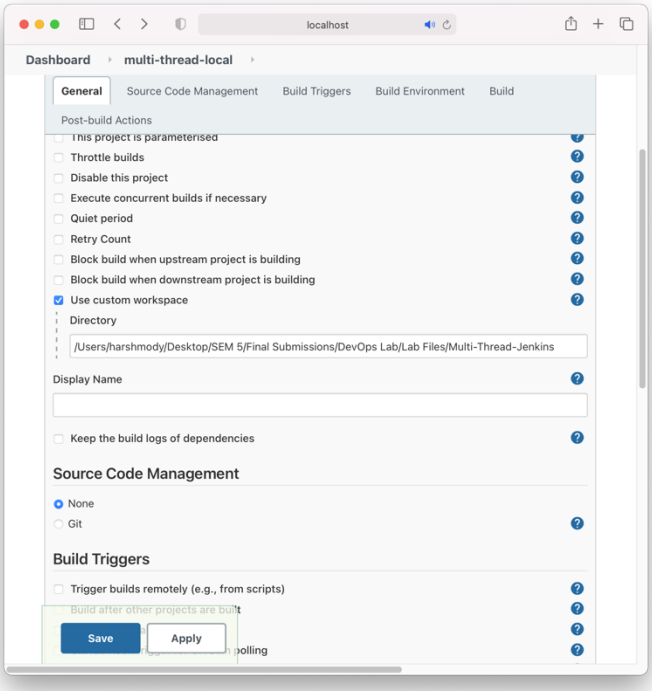
```
import java.util.concurrent.Phaser; public
class MultiThread
{
    public static void main(String[] args) throws InterruptedException
    {
        Phaser phaser = new Phaser();
        phaser.register();
        System.out.println("Printing Table of 3, 4, 6");
new MultiThread().PhaserRegister(phaser,200, 3);          new
MultiThread().PhaserRegister(phaser,600, 4);          new
MultiThread().PhaserRegister(phaser,800, 6);
phaser.arriveAndDeregister();
        Thread.sleep(15000);
        System.out.println("Done Printing Table ");
        System.exit(0);
    }     public static int i =
1;     public static int count =
0;
    void PhaserRegister(Phaser phaser,int sleepTime, int no)
    {
        phaser.register();
new Thread()
        {
            @Override
public void run()
            {
```

```
                for(count = 1; count <= 3 && phaser.arriveAndAwaitAdvance() > 0 &&
i <= 12; count++)                    {                        try                        {
                        phaser.arriveAndAwaitAdvance();
                        Thread.sleep(sleepTime);
                    }
                    catch (InterruptedException e)
                    {
                        e.printStackTrace();
break;
                    }
                    System.out.print(no + " X " + i + " = " + no* i+ "\t\t\t");
if (count % 3 == 0)
                    {
                        count = 0;
                        System.out.println();
i += 1;
                    }
                }
            }
        }.start();
    } }
```

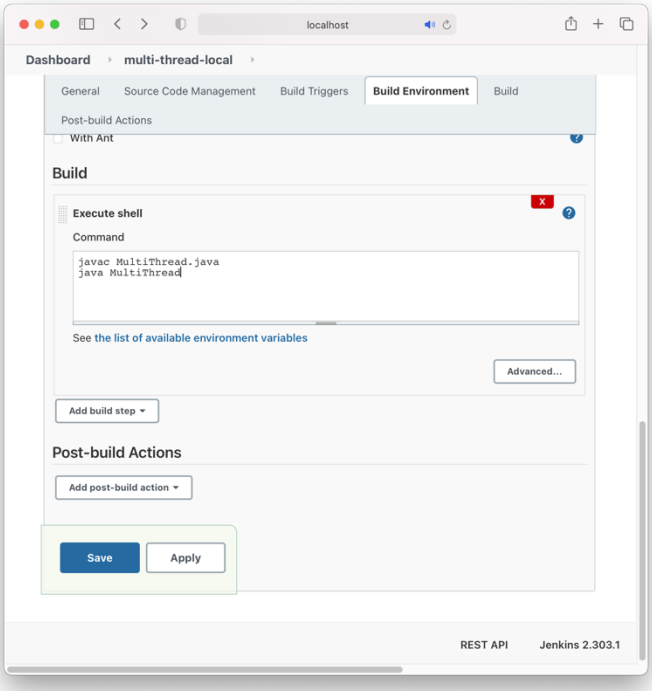
We then open http://localhost:8080 and login to Jenkins Dashboard and Create a New Item:



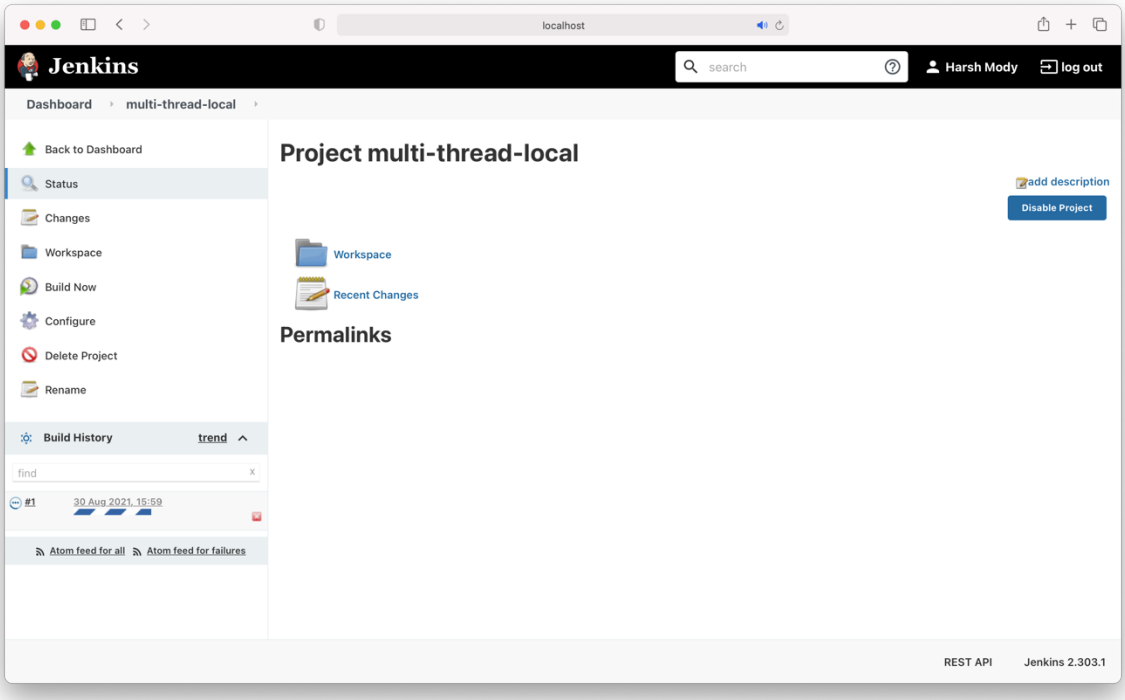
Then we, click on Advance and Select Custom Workspace. Here, add the folder with the Java Code.

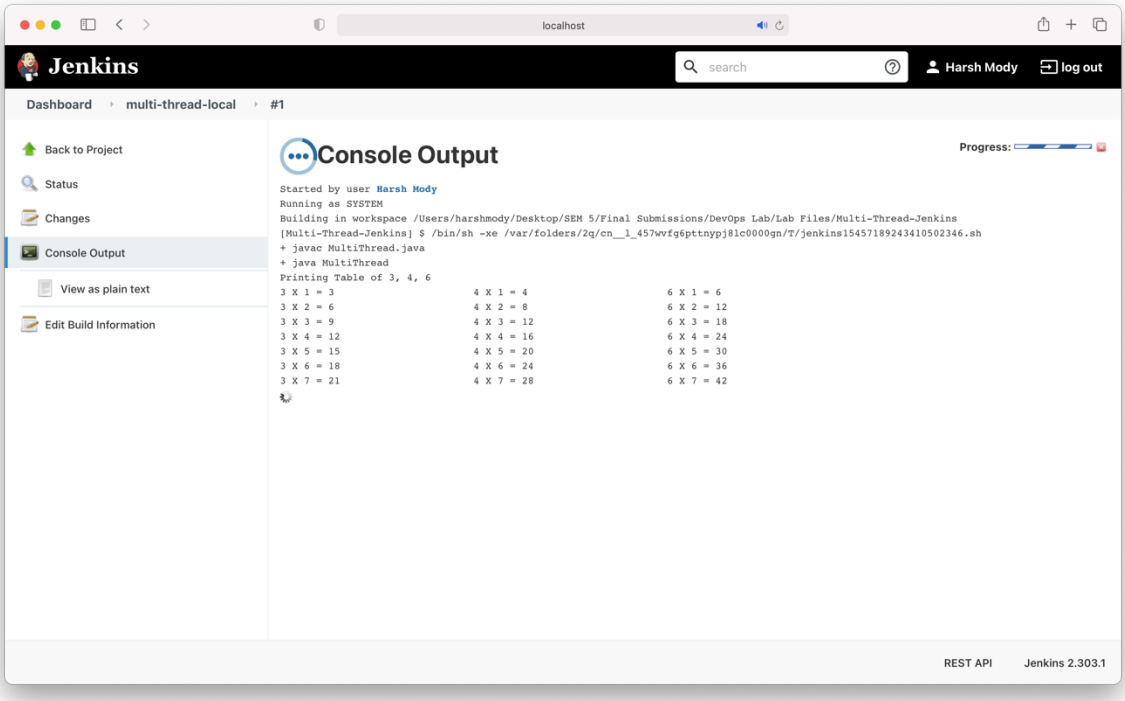


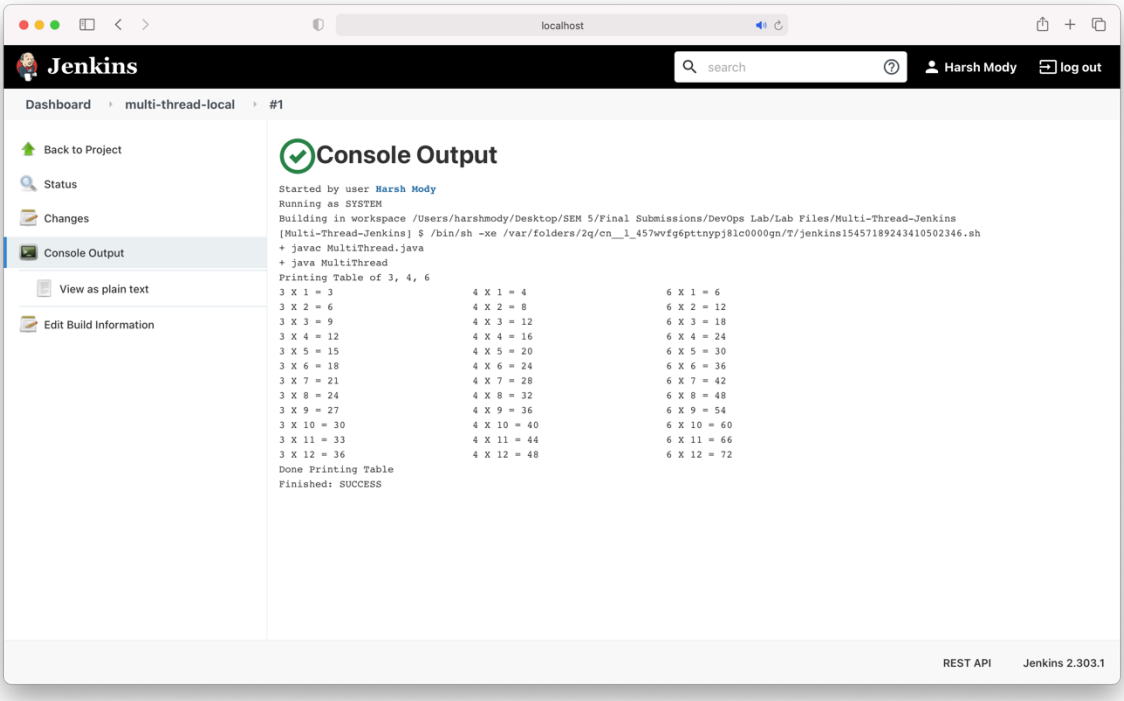
In build, select the Shell and add the command javac FileName.java and java Filename to automate the process of building and running Java Code and Click Apply and Save.



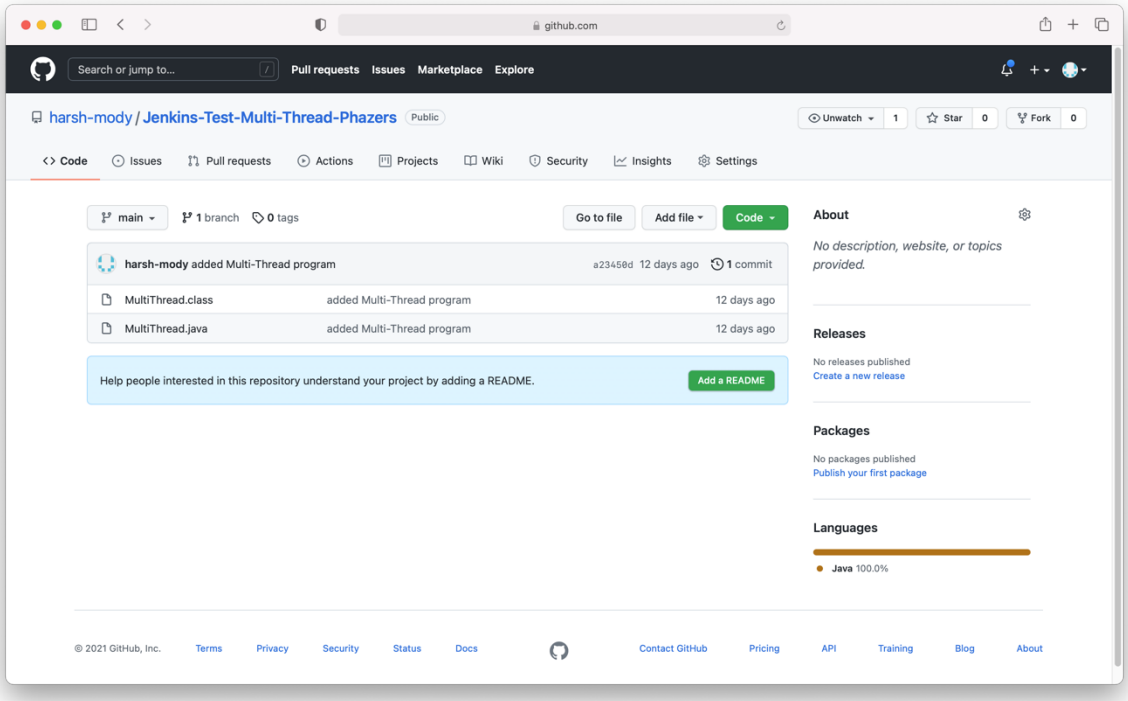
Now, we need to click on Build Now to build our Jenkins Project. And then, we click on the execution to view the console output. If we see SUCCESS and our code output, code execution was successful.

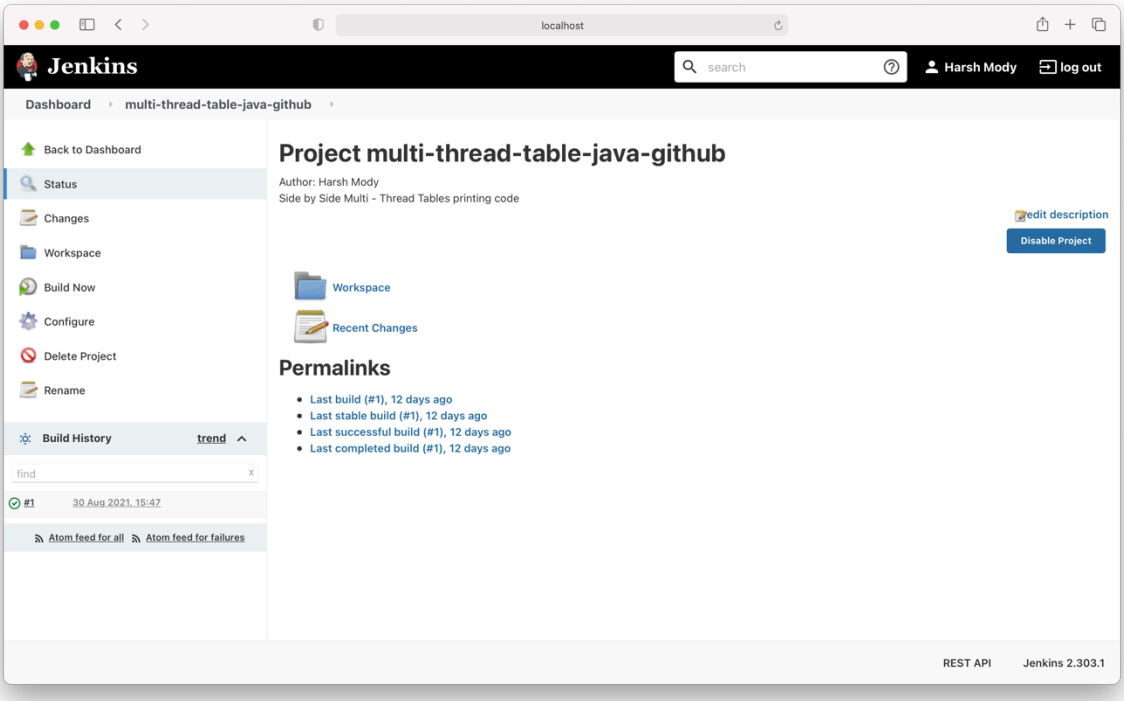


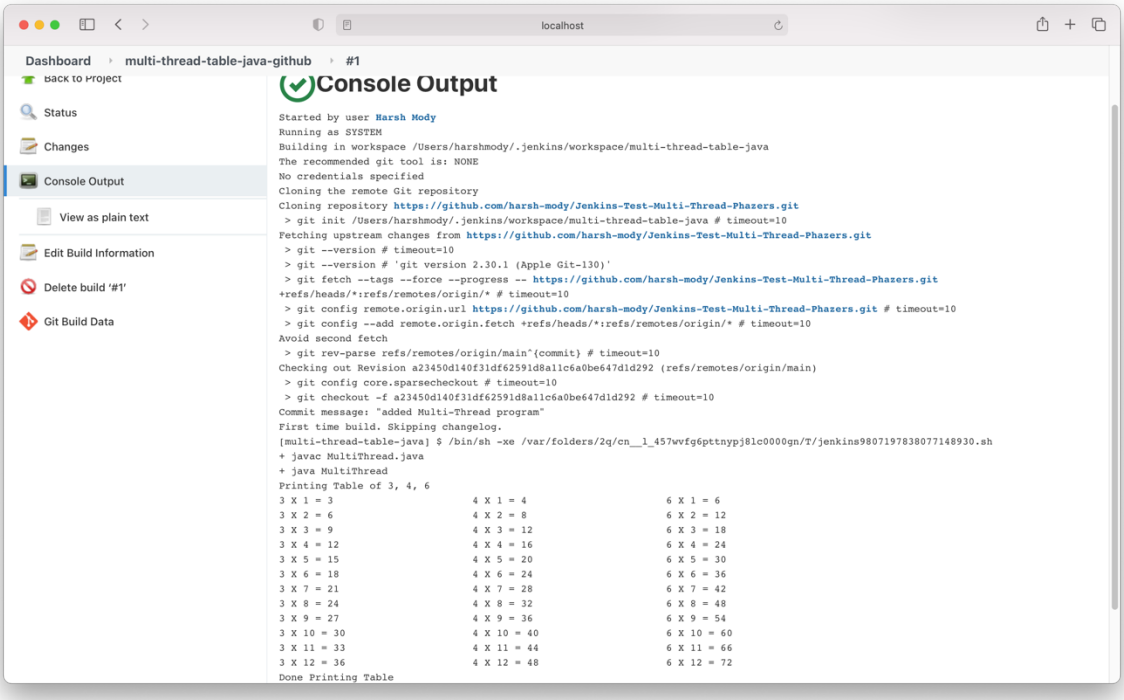




Now, we create a GitHub Repository for configure it too just like local code and then we click Build Now to build our Jenkins Project.







Conclusion: Thus, successfully understood the importance of Continuous Integration and Continuous Delivery Systems like Jenkins and also installed Jenkins on Mac successfully and also implemented a simple Build & Run Job for a Java Program on our local Machine & also using GitHub Repository.