ATME COLLEGE OF ENGINEERING

13th KM Stone, Bannur Road, Mysore - 570028

Department of Computer Science & Engineering – Cyber Security

(ACADEMIC YEAR 2025-26)

LABORATORY MANUAL

SUBJECT: DEVOPS

SUBJECT CODE: BCSL657D

SEMESTER: VI

Outcome Based Education (OBE) and Choice Based Credit System (CBCS)

(Effective from the academic year 2025-26)

Composed by	Verified by	Approved by
PRADEEPU J	Ms. Sandhya G	Dr. NASREEN FATHIMA
PROGRAMMER	FACULTY CO-ORDINATOR	HOD, CSD

INSTITUTIONAL MISSION AND VISION

Objectives

- To provide quality education and groom top-notch professionals, entrepreneurs and leaders for different fields of engineering, technology and management.
- To open a Training-R & D-Design-Consultancy cell in each department, gradually introduce doctoral and postdoctoral programs, encourage basic & applied research in areas of social relevance, and develop the instituteas a center of excellence.
- To develop academic, professional and financial alliances with the industry as well as the academia at national and transnational levels.
- To cultivate strong community relationships and involve the students and the staff in local community service.
- To constantly enhance the value of the educational inputs with the participation of students, faculty, parents and industry.

Vision

- Development of academically excellent, culturally vibrant, socially responsible and globally competent human resources.

Mission

- To keep pace with advancements in knowledge and make the students competitive and capable at the global level.
- To create an environment for the students to acquire the right physical, intellectual, emotional and moral foundations and shine as torch bearers of tomorrow's society.
- To strive to attain ever-higher benchmarks of educational excellence.

DEPARTMENT MISSION AND VISION

Vision

To be a global leader in Computer Science and Design Engineering, striving for design excellence, cultural awareness, a profound commitment to environmental stewardship, and societal responsibility.

Mission

- To establish a technology-enabled experiential learning environment, prioritizing and cultivating problem-solving and design thinking skills among students.
- To foster collaboration with industries, research and development organizations, jointly addressing socially relevant challenges in Computer Science with a core emphasis on design.

Program Specific Outcomes:

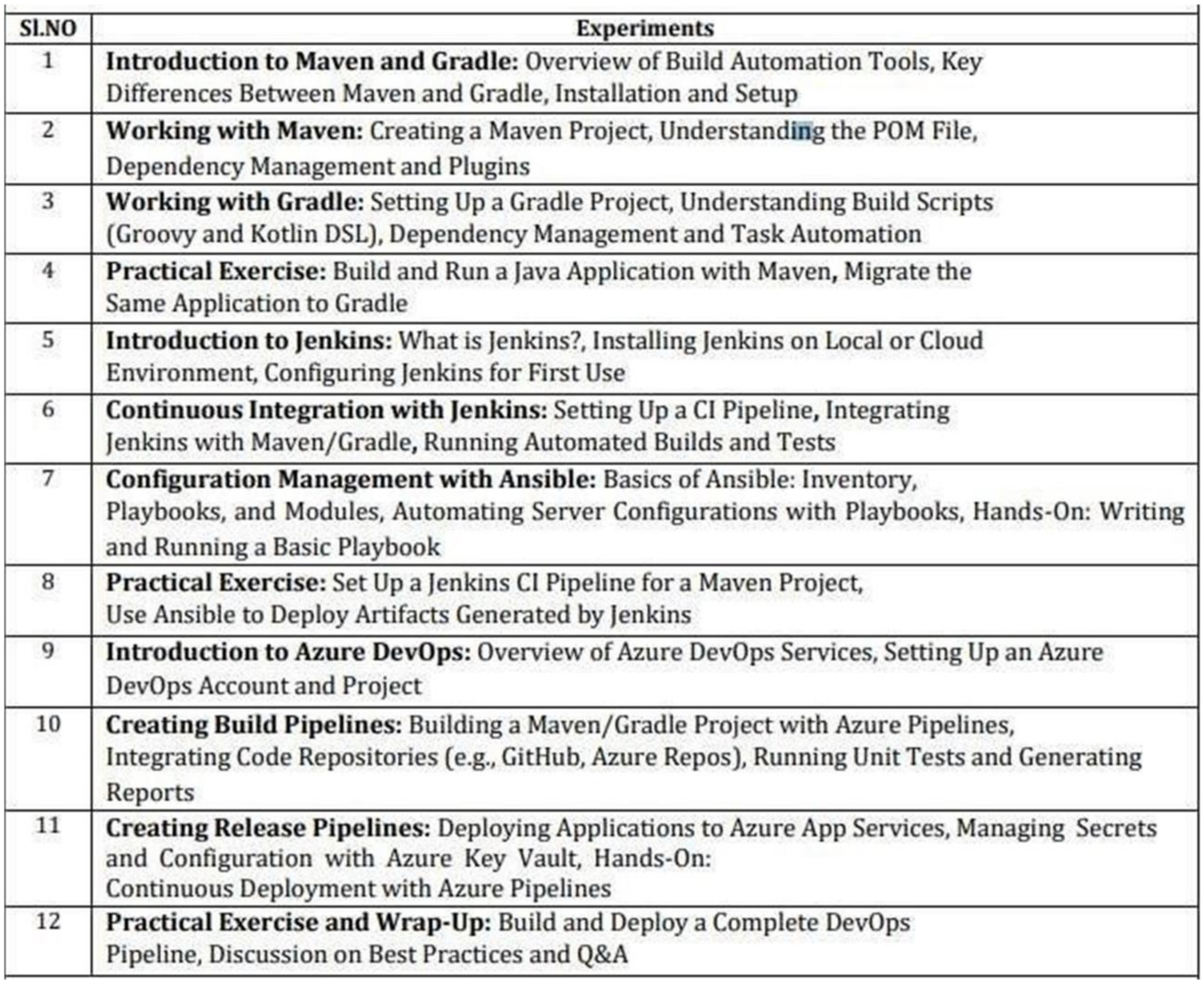
PSO1: To develop stand-alone, embedded, and web-based solutions with easy-to-operate interfaces using software engineering practices and contemporary programming languages.
PSO2: Design and develop computer-based systems in various areas of Multimedia, Graphics data visualization and computer vision.

DEVOPS

Subject Code	:	BCS657D	CIE Marks	:	50
Teaching Hours/Week (L:T:P: S)	:	0:0:2:0	SEE Marks	:	50
Credits	:	01	Total Marks	:	100
Examination type (SEE)		Theory/Practical	Exam Hours	:	02

Course objectives:

- To introduce DevOps terminology, definition & concepts
- To understand the different Version control tools like Git, Mercurial
- To understand the concepts of Continuous Integration/ Continuous Testing/ Continuous Deployment)
- To understand Configuration management using Ansible
- Illustrate the benefits and drive the adoption of cloud-based Devops tools to solve real world problems



Sl.NO	Experiments
1	**Introduction to Maven and Gradle:** Overview of Build Automation Tools, Key Differences Between Maven and Gradle, Installation and Setup
2	**Working with Maven:** Creating a Maven Project, Understanding the POM File, Dependency Management and Plugins
3	**Working with Gradle:** Setting Up a Gradle Project, Understanding Build Scripts (Groovy and Kotlin DSL), Dependency Management and Task Automation
4	**Practical Exercise:** Build and Run a Java Application with Maven, Migrate the Same Application to Gradle
5	**Introduction to Jenkins:** What is Jenkins?, Installing Jenkins on Local or Cloud Environment, Configuring Jenkins for First Use
6	**Continuous Integration with Jenkins:** Setting Up a CI Pipeline, Integrating Jenkins with Maven/Gradle, Running Automated Builds and Tests
7	**Configuration Management with Ansible:** Basics of Ansible: Inventory, Playbooks, and Modules, Automating Server Configurations with Playbooks, Hands-On: Writing and Running a Basic Playbook
8	**Practical Exercise:** Set Up a Jenkins CI Pipeline for a Maven Project, Use Ansible to Deploy Artifacts Generated by Jenkins
9	**Introduction to Azure DevOps:** Overview of Azure DevOps Services, Setting Up an Azure DevOps Account and Project
10	**Creating Build Pipelines:** Building a Maven/Gradle Project with Azure Pipelines, Integrating Code Repositories (e.g., GitHub, Azure Repos), Running Unit Tests and Generating Reports
11	**Creating Release Pipelines:** Deploying Applications to Azure App Services, Managing Secrets and Configuration with Azure Key Vault, Hands-On: Continuous Deployment with Azure Pipelines
12	**Practical Exercise and Wrap-Up:** Build and Deploy a Complete DevOps Pipeline, Discussion on Best Practices and Q&A

Course outcomes (Course Skill Set):

At the end of the course the student will be able to:

- Demonstrate different actions performed through Version control tools like Git.
- Perform Continuous Integration and Continuous Testing and Continuous Deployment using Jenkins by building and automating test cases using Maven & Gradle.
- Experiment with configuration management using Ansible.
- Demonstrate Cloud-based DevOps tools using Azure DevOps.

Assessment Details (both CIE and SEE)

The weightage of Continuous Internal Evaluation (CIE) is 50% and for Semester End Exam (SEE) is 50%. The minimum passing mark for the CIE is 40% of the maximum marks (20 marks out of 50) and for the SEE minimum passing mark is 35% of the maximum marks (18 out of 50 marks). A student shall be deemed to have satisfied the academic requirements and earned the credits allotted to each subject/ course if the student secures a minimum of 40% (40 marks out of 100) in the sum total of the CIE (Continuous Internal Evaluation) and SEE (Semester End Examination) taken together

Continuous Internal Evaluation (CIE):

CIE marks for the practical course are **50 Marks**.

The split-up of CIE marks for record/ journal and test are in the ratio **60:40**.

- Each experiment is to be evaluated for conduction with an observation sheet and record write-up. Rubrics for the evaluation of the journal/write-up for hardware/software experiments are designed by the faculty who is handling the laboratory session and are made known to students at the beginning of the practical session.
- Record should contain all the specified experiments in the syllabus and each experiment write-up will be evaluated for 10 marks.
- Total marks scored by the students are scaled down to **30 marks** (60% of maximum marks).
- Weightage to be given for neatness and submission of record/write-up on time.
- Department shall conduct a test of 100 marks after the completion of all the experiments listed in the syllabus.
- In a test, test write-up, conduction of experiment, acceptable result, and procedural knowledge will carry a weightage of 60% and the rest 40% for viva-voce.
- The suitable rubrics can be designed to evaluate each student's performance and learning ability.
- The marks scored shall be scaled down to **20 marks** (40% of the maximum marks).

The Sum of scaled-down marks scored in the report write-up/journal and marks of a test is the total CIE marks scored by the student.

Semester End Evaluation (SEE):

- SEE marks for the practical course are 50 Marks.
- SEE shall be conducted jointly by the two examiners of the same institute, examiners are appointed by the Head of the Institute.
- The examination schedule and names of examiners are informed to the university before the conduction of the examination. These practical examinations are to be conducted between the schedule mentioned in the academic calendar of the University.
- All laboratory experiments are to be included for practical examination.
- (Rubrics) Breakup of marks and the instructions printed on the cover page of the answer script to be strictly adhered to by the examiners. **OR** based on the course requirement evaluation rubrics shall be decided jointly by examiners.

- Students can pick one question (experiment) from the questions lot prepared by the examiners jointly.
- Evaluation of test write-up/ conduction procedure and result/viva will be conducted jointly by examiners.

General rubrics suggested for SEE are mentioned here, writeup-20%, Conduction procedure and result in -60%, Viva-voce 20% of maximum marks. SEE for practical shall be evaluated for 100 marks and scored marks shall be scaled down to 50 marks (however, based on course type, rubrics shall be decided by the examiners)
Change of experiment is allowed only once and 15% of Marks allotted to the procedure part are to be made zero.
The minimum duration of SEE is 02 hours

Suggested Learning Resources:

- https://www.geeksforgeeks.org/devops-tutorial/
- https://www.javatpoint.com/devops
- https://www.youtube.com/watch?v=2N-59wUIPVI
- https://www.youtube.com/watch?v=87ZqwoFeO88

CONTENTS

1. Introduction to Maven and Gradle: Overview of Build Automation Tools, Key Differences Between Maven and Gradle, Installation and Setup.

Introduction to Maven and Gradle
Overview of Build Automation Tools
Build automation tools help developers streamline the process of building, testing, and deploying software projects. They take care of repetitive tasks like compiling code, managing dependencies, and packaging applications, which makes development more efficient and error-free.
Two popular tools in the Java ecosystem are **Maven** and **Gradle**. Both are great for managing project builds and dependencies, but they have some key differences.

Maven

- **What is Maven?** Maven is a build automation tool primarily used for Java projects. It uses an XML configuration file called pom.xml (Project Object Model) to define project settings, dependencies, and build steps.
- **Main Features:**
    - Predefined project structure and lifecycle phases.
    - Automatic dependency management through Maven Central.
    - Wide range of plugins for things like testing and deployment.
    - Supports complex projects with multiple modules.

Gradle

- **What is Gradle?** Gradle is a more modern and versatile build tool that supports multiple programming languages, including Java, Groovy, and Kotlin. It uses a domain-specific language (DSL) for build scripts, written in Groovy or Kotlin.

- **Main Features:**
    - Faster builds thanks to task caching and incremental builds.
    - Flexible and customizable build scripts.
    - Works with Maven repositories for dependency management.
    - Excellent support for multi-module and cross-language projects.
    - Integrates easily with CI/CD pipelines.

Key Differences Between Maven and Gradle

Aspect	Maven	Gradle
Configuration	XML (pom.xml)	Groovy or Kotlin DSL
Performance	Slower	Faster due to caching
Flexibility	Less flexible	Highly customizable
Learning Curve	Easier to pick up	Slightly steeper
Script Size	Verbose	More concise
Dependency Management	Uses Maven Central	Compatible with Maven too
Plugin Support	Large ecosystem	Extensible and versatile

Installation and Setup & How to Install Maven:

1. **Download Maven:**
   Go to the Maven Download Page and download the latest binary ZIP file.
2. **Extract the ZIP File:**
   Right-click the downloaded ZIP file and select Extract All… or use any extraction tool like WinRAR or 7-Zip.
3. **Move the Folder:**
   After extraction, move the extracted Maven folder (usually named apache-maven-x.x.x) to a convenient directory like C:\Program Files\.
4. **Navigate to the bin Folder:**
   - Open the Maven folder, then navigate to the bin folder inside.
   - Copy the path from the File Explorer address bar(e.g., C:\Program Files\apache-maven-x.x.x\bin).
5. **Set Environment Variables:**
   - Open the Start Menu, search for Environment Variables, and select Edit the system environment variables.
   - Click Environment Variables.
   - Under System Variables:
     - Find the path, double click on it and click New.
     - Paste the full path to the bin folder of your Maven directory (e.g., C:\Program Files\apache-maven-x.x.x\bin).

6. **Save the Changes:**
    - Click OK to close the windows and save your changes.
7. **Verify the Installation:**
    - Open Command Prompt and run: mvn **-v** If Maven is correctly installed, it will display the version number.

How to install Gradle

1. **Download Gradle:**
   Visit the Gradle Downloads Page and download the latest binary ZIP file.
2. **Extract the ZIP File:**
   Right-click the downloaded ZIP file and select Extract All… or use any extraction tool like WinRAR or 7-Zip.
3. **Move the Folder:**
   After extraction, move the extracted Gradle folder (usually named gradle-x.x.x) to a convenient directory like C:\Program Files\.
4. **Navigate to the bin Folder:**
    - Open the Gradle folder, then navigate to the bin folder inside.
    - Copy the path from the File Explorer address bar (e.g., C:\Program Files\gradle-x.x\bin).
5. **Set Environment Variables:**
    - Open the Start Menu, search for Environment Variables, and select Edit the system environment variables.
    - Click Environment Variables.
    - Under System Variables:
        - Find the path, double click on it and click New.
        - Paste the full path to the bin folder of your Gradle directory (e.g., C:\Program Files\gradle-x.x.x\bin).
6. **Save the Changes:**
    - Click OK to close the windows and save your changes.
7. **Verify the Installation:**
    - Open a terminal or Command Prompt and run: gradle -v If it shows the Gradle version, the setup is complete.

Install the Java JDK

- If you haven't installed the **Java JDK** yet, you can follow the link below to download and install it. Download Java JDK from Oracle

2. Working with Maven: Creating a Maven Project, Understanding the POM File, Dependency Management and Plugins.

Working with Maven is a key skill for managing Java-based projects, particularly in the areas of build automation, dependency management, and project configuration. Below is a guide on creating a Maven project, understanding the POM file, and using dependency management and plugins:

Creating a Maven Project

There are a few ways to create a Maven project, such as using the command line, IDEs like IntelliJ IDEA or Eclipse, or generating it via an archetype.

Step 1: Create a New Maven Project:

Open IntelliJ IDEA.

Go to file -> new -> project

Select maven from the project types

Set the project name and location, then click Finish .

Understanding the POM File

The POM (Project Object Model) file is the heart of a Maven project. It is an XML file that contains all the configuration details about the project

Step 2: set up the pom.xml file

Add dependencies in the pom file

Add selenium java and TestNG dependencies by creating a tag called <dependencies> section

```
<dependencies>
    <dependency>
        <groupId>org.seleniumhq.selenium</groupId>
        <artifactId>selenium-java</artifactId>
        <version>3.141.59</version>
    </dependency>
    <dependency>
        <groupId>org.testng</groupId>
        <artifactId>testng</artifactId>
        <version>7.4.0</version>
        <scope>test</scope>
    </dependency>
</dependencies>
```

Dependency Management

Maven uses the <dependencies> tag in the pom.xml to manage external libraries or dependencies that your project needs. When Maven builds the project, it will automatically download these dependencies from a repository (like Maven Central).

- **Transitive Dependencies**

  Maven automatically resolves transitive dependencies. For example, if you add a library that depends on other libraries, Maven will also download those.

- **Scopes**
  - Dependencies can have different scopes that determine when they are available:
    - compile (default): Available in all build phases.
    - provided: Available during compilation but not at runtime (e.g., a web server container).
    - runtime: Needed only at runtime, not during compilation.
    - test: Required only for testing.

Plugins

Maven plugins are used to perform tasks during the build lifecycle, such as compiling code, running tests, packaging, and deploying. You can specify plugins within the <build> section of your pom.xml. The maven-compiler-plugin is used to compile Java code and specify the source and target JDK versions.

1. **Common Plugins**
   - **maven-compiler-plugin**: Compiles Java code.
   - **maven-surefire-plugin**: Runs unit tests.
   - **maven-jar-plugin**: Packages the project as a JAR file.
   - **maven-clean-plugin**: Cleans up the target/ directory.
2. **Plugin Goals** Each plugin consists of goals, which are specific tasks to be executed. For example:
   - **mvn clean install:** This will clean the target directory and then install the package in the local repository.
   - **mvn compile:** This will compile the source code.
3. **mvn test:** This will run unit tests.

Step 3: To do test under the Main Create new package in Src-main-java-org.example.

1. Copy LoginTest Code in Org.Example

```
import org.openqa.selenium.By;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.chrome.ChromeDriver;
public class LoginTest {
    public static void main(String[] args) throws InterruptedException {
        WebDriver driver = new ChromeDriver();
        driver.get("https://www.saucedemo.com/");
        driver.manage().window().maximize();
        Thread.sleep(2000);
        driver.findElement(By.id("user-name")).sendKeys("standard_user");
        Thread.sleep(2000);
        driver.findElement(By.id("password")).sendKeys("secret_sauce");
        Thread.sleep(2000);
        driver.findElement(By.id("login-button")).click();
        Thread.sleep(2000);
        driver.quit();
    }
}
```

Step 4: Next To create the Docs file Copy this Plugine Code after Dependencies, which creates new directory Docs
Build /Plugin Code to create Docs file

```
<build>
        <plugins>
            <plugin>
                <groupId>org.apache.maven.plugins</groupId>
                <artifactId>maven-resources-plugin</artifactId>
                <version>3.2.0</version>
                <executions>
                    <execution>
                        <phase>prepare-package</phase>
                        <goals>
                            <goal>copy-resources</goal>
                        </goals>
                        <configuration>
                            <outputDirectory>${project.basedir}/docs</outputDirectory> <!-- Deploy to
/docs folder -->
                            <resources>
                                <resource>
                                    <directory>src/main/resources</directory>
                                    <includes>
                                        <include>*/</include> <!-- Copy all files in src/main/resources -->
                                    </includes>
                                </resource>
                            </resources>
                        </configuration>
                    </execution>
                </executions>
            </plugin>
        </plugins>
        </build>
```

Step 5: Create a Simple Website (HTML, CSS, and Logo):

Index.html:

```
 <!DOCTYPE html>
 <html lang="en">
 <head>
 <meta charset="UTF-8">
 <meta name="viewport" content="width=device-width, initial-scale=1.0">
 <title>My Simple Website</title>
 <link rel="stylesheet" href="style.css">
 </head>
 <body>
<header>
<img src="logo.png" alt="Logo">
</header>
<h1>Welcome to My Simple Website</h1>
</body>
</html>
```

Styles.css:

```
body {
background-color: #f4f4f4;
text-align: center;
}
header img {
    width: 100px;
}
```

Step 6:

Upload the Website to GitHub:

Initialize a Git repository in your project folder:

```
git init
```

Add your files and commit them:

```
git add .
git commit -m "Initial commit"
```

Create a GitHub repository and push the local project to GitHub:

```
git remote add origin <your-repository-url>
git push -u origin master
```

Step 7: Write a Simple Selenium Test with TestNG:

Create a new java class WebpageTest,java in the src/test/java directory.

```
package org.test;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.chrome.ChromeDriver;
import org.testng.Assert;
import org.testng.annotations.AfterTest;
import org.testng.annotations.BeforeTest;
import org.testng.annotations.Test;
import static org.testng.Assert.assertTrue;
public class WebpageTest {
private static WebDriver driver;
   @BeforeTest
   public void openBrowser() throws InterruptedException {
      driver = new ChromeDriver();
      driver.manage().window().maximize();
      Thread.sleep(2000);
      driver.get("");
   }
   @Test
   public void titleValidationTest(){
      String actualTitle = driver.getTitle();
      String expectedTitle = "Tripillar Solutions";
      Assert.assertEquals(actualTitle, expectedTitle);
   }
   @AfterTest
   public void closeBrowser() throws InterruptedException {
      Thread.sleep(1000);
      driver.quit();
   }
}
```

Run the Test:

In IntelliJ, right-click the 'WebPageTest' class and select Run 'WebPageTest' .

◦ The test will launch Chrome, open the webpage, and validate the title.

Deployment :

To deploy your Maven project to GitHub Pages using the /docs

Step 8: Create Java Archive file. Using that executes simple program

Plugins code to create Maven Jar files

```
<plugin>
            <groupId>org.apache.maven.plugins</groupId>
            <artifactId>maven-jar-plugin</artifactId>
            <version>3.1.0</version>
            <configuration>
               <!-- Specify the main class to be executed -->
               <archive>
                  <manifestEntries>
                     <Main-Class>org.example.Main</Main-Class>
<!-- Replace with your actual main class -->
                  </manifestEntries>
               </archive>
            </configuration>
         </plugin>
```

Step 9: Create new Java class called Main under
Src-main-java-org.example- Main

```
package org.example;

public class Main {
   public static void main(String[] args) {
      System.out.println("Hello from maven!");
   }
```

Step 10: To deploy site copy below code in POM file

Maven Site Plugin code

```
<plugin>
            <groupId>org.apache.maven.plugins</groupId>
            <artifactId>maven-site-plugin</artifactId>
            <version>3.12.1</version>  <!-- Use the latest version -->
         </plugin>
```

3. Working with Gradle: Setting Up a Gradle Project, Understanding Build Scripts (Groovy and Kotlin DSL), Dependency Management and Task Automation

Creating a Gradle Project in IntelliJ IDEA

Step 1: Open IntelliJ IDEA and Create a New Project
Click on "New Project".
Select "Gradle" (under Java).
Choose Groovy for the build script.
Set the Group ID (e.g., com.example).
Click Finish.

Step 2: Build and Run a Simple Java Application
Copy the Plugin code in build.gradle file

```
plugins {
    id 'application'
}
repositories {
    mavenCentral()
}
dependencies {
  testImplementation  'org.junit.jupiter:junit-jupiter:5.8.1'
 testImplementation 'org.seleniumhq.selenium:selenium-java:4.28.1' // use the latest stable version
 testImplementation 'org.testng:testng:7.4.0' // use the latest stable version
}
 test {
 useTestNG()
 }

application {
    mainClass = 'com.example.Main'
}
```

Step 3: Create new Package com.example
Src-main-java-com.example-Main(java class)

```
package org.example;

public class Main {
    public static void main(String[] args) {
        System.out.println("Hello from Gradle!");
    }
}
```

Step 4: Hosting Static Website on Github pages
Manually create new directory docs . copy and paste the index.html, logo and .css files.
Testing Website using selenium and TestNG dependencies in build.gradle

```
dependencies {
 testImplementation 'org.seleniumhq.selenium:selenium-java:4.28.1' // use the latest stable version
 testImplementation 'org.testng:testng:7.4.0' // use the latest stable version
 }

 test {
 useTestNG()
 }
```

Step 5: Then create New Package org.test
Src-test-org.test

Create new java WebpageTest file under org.test

```
package org.test;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.chrome.ChromeDriver;
import org.testng.Assert;
import org.testng.annotations.AfterTest;

import org.testng.annotations.BeforeTest;

import org.testng.annotations.Test;
import static org.testng.Assert.assertTrue;
public class WebpageTest {
   private static WebDriver driver;
   @BeforeTest
   public void openBrowser() throws InterruptedException {
      driver = new ChromeDriver();
      driver.manage().window().maximize();
      Thread.sleep(2000);
      driver.get("https://sauravsarkar-codersarcade.github.io/CA-MVN/");
   }

   @Test
   public void titleValidationTest(){
      String actualTitle = driver.getTitle();
      String expectedTitle = "Tripillar Solutions";
      Assert.assertEquals(actualTitle, expectedTitle);
      assertTrue(true, "Title should contain 'Tripillar'");
   }

   @AfterTest
   public void closeBrowser() throws InterruptedException {
      Thread.sleep(1000);
      driver.quit();
   }
}
```

Step 6:
To Create Gradle jar

```
plugins {
 id 'java'
id 'application'
}
 application {
mainClass = 'com.example.Main'
 }

jar {
 manifest {
attributes 'Main-Class': 'com.example.Main' // This tells Java where to start execution
 }
 }
```

4. Practical Exercise: Build and Run a Java Application with Maven, Migrate the Same Application to Gradle.

Step 1: Create a Maven Project in IntelliJ IDEA

1. Open IntelliJ IDEA
Launch IntelliJ IDEA and click on File → New → Project.

2. Select Maven
• In the New Project window, choose Maven from the options on the left.
• Check Create from archetype and select maven-archetype-quickstart.
• Click Next.

3. Enter Project Details
• GroupId: com.example
• ArtifactId: MVNGRDLDEMO
• Click Next and then Finish.

4. Wait for IntelliJ to Load Dependencies
IntelliJ will automatically download the Maven dependencies, so just relax for a moment.

Step 2: Update pom.xml to Add Build Plugin

To compile and package your project into a .jar file, you need to add the Maven Compiler and Jar plugins.

1. Open the pom.xml file.
2. Add the following inside the <project> tag:

```
<build>
 <plugins>
 <!-- Compiler Plugin -->
 <plugin>
 <groupId>org.apache.maven.plugins</groupId>
 <artifactId>maven-compiler-plugin</artifactId>
 <version>3.8.1</version>
 <configuration>
 <source>1.8</source>
 <target>1.8</target>
 </configuration>
 </plugin>

<!-- Jar Plugin -->
```

```
<plugin>
 <groupId>org.apache.maven.plugins</groupId>
 <artifactId>maven-jar-plugin</artifactId>
 <version>3.2.0</version>
 <configuration>
 <archive>
 <manifest>
 <mainClass>com.example.App</mainClass>
 </manifest>
 </archive>
 </configuration>
 </plugin>
 </plugins>
 </build>
```

Step 3: Build and Run the Maven Project

1. Open IntelliJ IDEA Terminal

Press Alt + F12 to open the terminal.

2. Compile and Package the Project

Run the following commands to build the project:

```
mvn clean compile
mvn package
```

Step 4: .Locate the JAR File

After running the above, your .jar file will be located at:

```
D:\Idea  Projects\MVNGRDLDEMO\target\MVNGRDLDEMO-1.0-SNAPSHOT.jar
```

Run the JAR File

To run the generated JAR file, use:

```
java -jar target\MVNGRDLDEMO-1.0-SNAPSHOT.jar
```

Part 2: Migrate Maven Project to Gradle

Step 1: Initialize Gradle in Your Project

1. Open Terminal in IntelliJ IDEA

Make sure you're in the project directory

2. Run Gradle Init Command

Execute the following command to migrate your Maven project to Gradle:

```
gradle init --type pom
```

This command will convert your Maven pom.xml into a Gradle build.gradle file.

Step 2: Review and Update

1. Open build.gradle in IntelliJ IDEA.

2. Ensure the following configurations are correct:

```
plugins {
id 'java'
}
group = 'com.example'
version = '1.0-SNAPSHOT'
repositories {
mavenCentral()
}
dependencies {
testImplementation 'junit:junit:4.13.2'
}
jar {
manifest {
attributes(
'Main-Class': 'com.example.App'
)
}
}
```

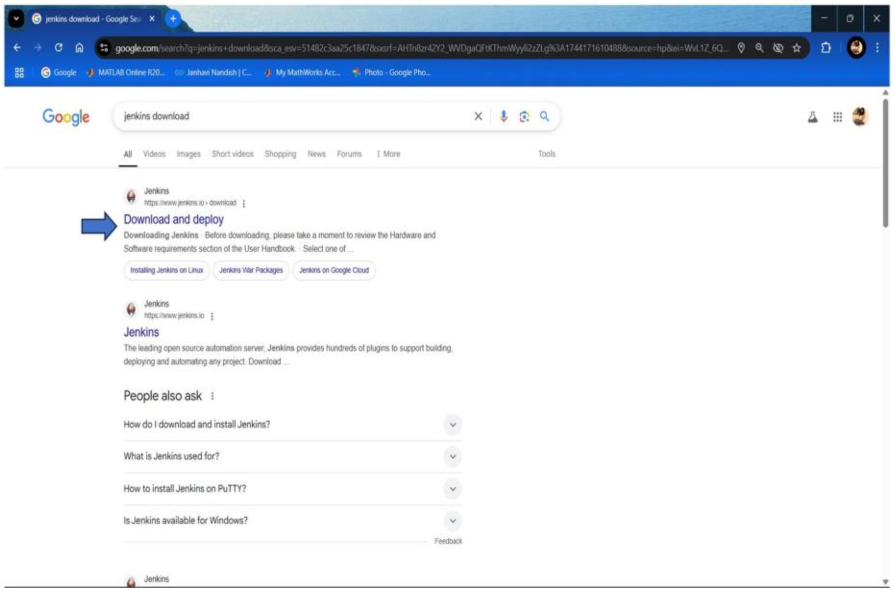
Step 3: Build and Run the Gradle Project

1. **Clean and Build the Project**
   To clean and build your Gradle project, run: gradle clean build
2. **Run the Generated JAR File**
   Now, run the generated JAR file using:

```
java -jar build/libs/MVNGRDLDEMO-1.0-SNAPSHOT.jar
```

5.Introduction to Jenkins: What is Jenkins?, Installing Jenkins on Local or Cloud Environment, Configuring Jenkins for First Use

Installing Jenkins and configuration

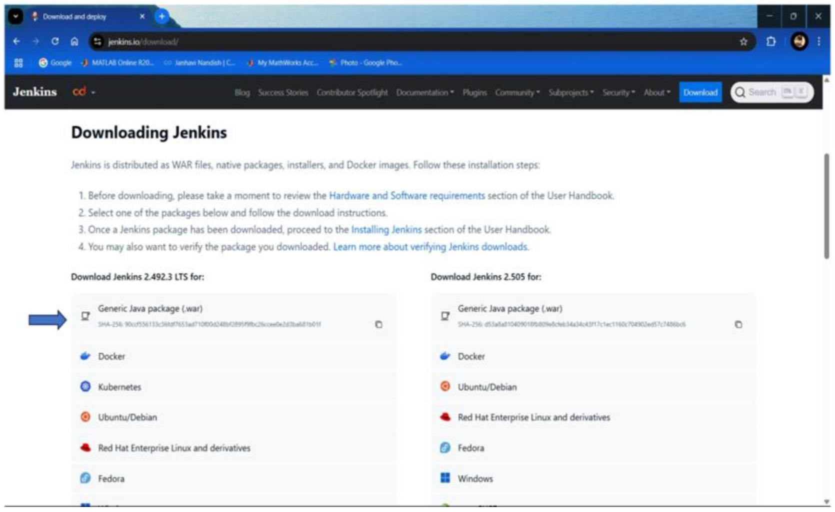

jenkins download

Jenkins
https://www.jenkins.io › download
Download and deploy
Downloading Jenkins · Before downloading, please take a moment to review the Hardware and Software requirements section of the User Handbook. · Select one of ...

Jenkins
https://www.jenkins.io
Jenkins
The leading open source automation server, Jenkins provides hundreds of plugins to support building, deploying and automating any project. Download ...

People also ask
How do I download and install Jenkins?
What is Jenkins used for?
How to install Jenkins on PuTTY?
Is Jenkins available for Windows?

Under download Jenkins LTS click on Generic Java package


jenkins.io/download/

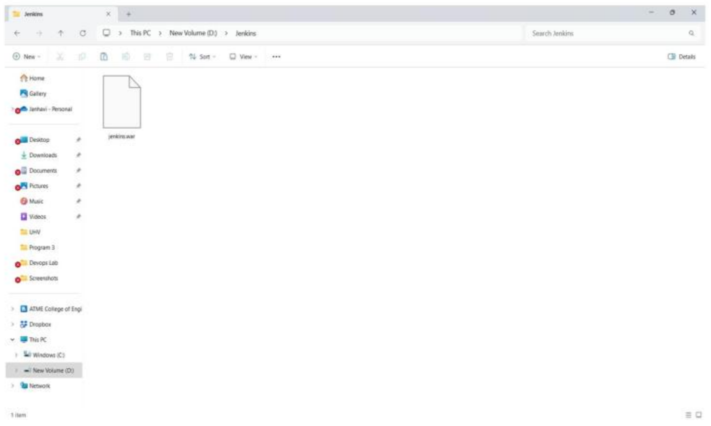
Downloading Jenkins

Jenkins is distributed as WAR files, native packages, installers, and Docker images. Follow these installation steps:

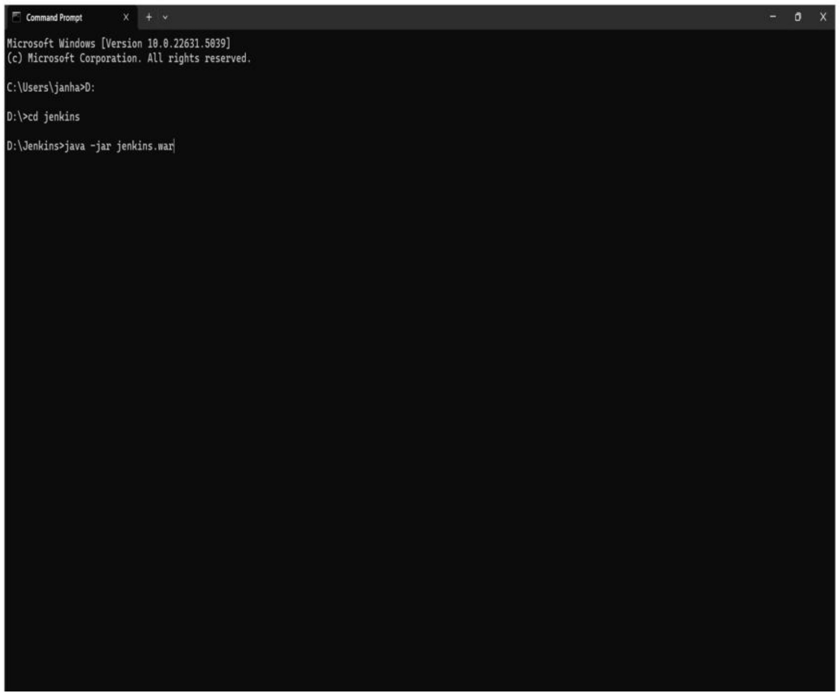
1. Before downloading, please take a moment to review the Hardware and Software requirements section of the User Handbook.
2. Select one of the packages below and follow the download instructions.
3. Once a Jenkins package has been downloaded, proceed to the Installing Jenkins section of the User Handbook.
4. You may also want to verify the package you downloaded. Learn more about verifying Jenkins downloads.

Download Jenkins 2.492.3 LTS for:	Download Jenkins 2.505 for:
Generic Java package (.war)	Generic Java package (.war)
Docker	Docker
Kubernetes	Ubuntu/Debian
Ubuntu/Debian	Red Hat Enterprise Linux and derivatives
Red Hat Enterprise Linux and derivatives	Fedora
Fedora	Windows

Create a new folder called Jenkins in a drive D
Copy the downloaded file to D drive in the Jenkins folder.

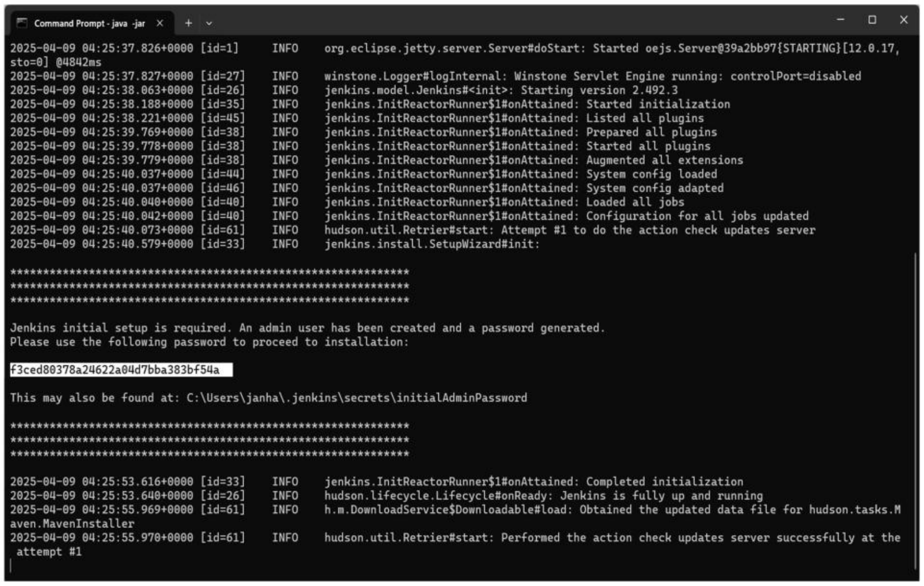


In the command prompt, type the following commands to run the server



```
Microsoft Windows [Version 10.0.22631.5039]
(c) Microsoft Corporation. All rights reserved.

C:\Users\janha>D:

D:\>cd jenkins

D:\Jenkins>java -jar jenkins.war
```

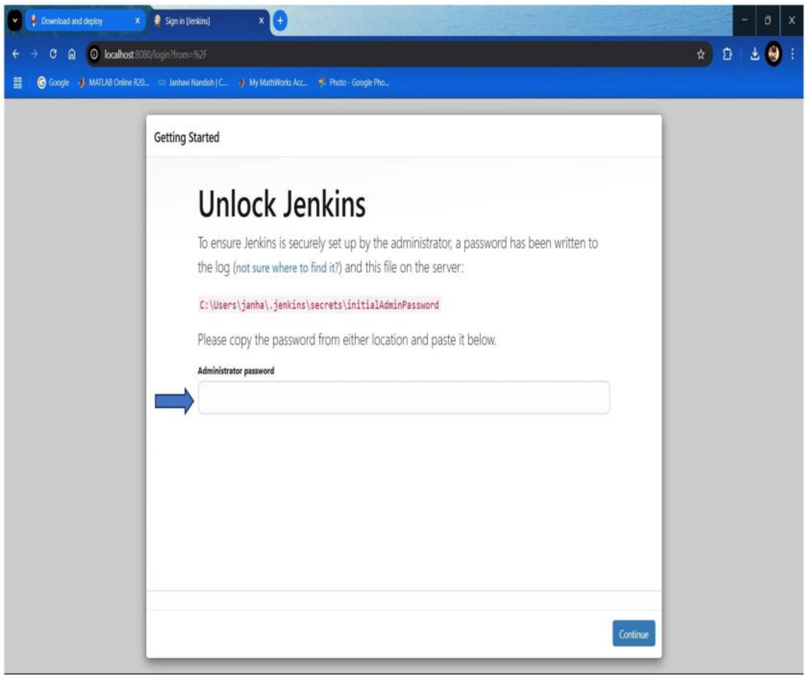
Copy the unique 24-character code for further use



```
2025-04-09 04:25:37.826+0000 [id=1]     INFO    org.eclipse.jetty.server.Server#doStart: Started oejs.Server@39a2bb97{STARTING}[12.0.17,
sto=0] @4842ms
2025-04-09 04:25:37.827+0000 [id=27]    INFO    winstone.Logger#logInternal: Winstone Servlet Engine running: controlPort=disabled
2025-04-09 04:25:38.063+0000 [id=26]    INFO    jenkins.model.Jenkins#<init>: Starting version 2.492.3
2025-04-09 04:25:38.188+0000 [id=35]    INFO    jenkins.InitReactorRunner$1#onAttained: Started initialization
2025-04-09 04:25:38.221+0000 [id=45]    INFO    jenkins.InitReactorRunner$1#onAttained: Listed all plugins
2025-04-09 04:25:39.769+0000 [id=38]    INFO    jenkins.InitReactorRunner$1#onAttained: Prepared all plugins
2025-04-09 04:25:39.778+0000 [id=38]    INFO    jenkins.InitReactorRunner$1#onAttained: Started all plugins
2025-04-09 04:25:39.779+0000 [id=38]    INFO    jenkins.InitReactorRunner$1#onAttained: Augmented all extensions
2025-04-09 04:25:40.037+0000 [id=44]    INFO    jenkins.InitReactorRunner$1#onAttained: System config loaded
2025-04-09 04:25:40.037+0000 [id=46]    INFO    jenkins.InitReactorRunner$1#onAttained: System config adapted
2025-04-09 04:25:40.040+0000 [id=40]    INFO    jenkins.InitReactorRunner$1#onAttained: Loaded all jobs
2025-04-09 04:25:40.042+0000 [id=40]    INFO    jenkins.InitReactorRunner$1#onAttained: Configuration for all jobs updated
2025-04-09 04:25:40.073+0000 [id=61]    INFO    hudson.util.Retrier#start: Attempt #1 to do the action check updates server
2025-04-09 04:25:40.579+0000 [id=33]    INFO    jenkins.install.SetupWizard#init:

*************************************************************
*************************************************************
*************************************************************

Jenkins initial setup is required. An admin user has been created and a password generated.
Please use the following password to proceed to installation:

f3ced80378a24622a04d7bba383bf54a

This may also be found at: C:\Users\janha\.jenkins\secrets\initialAdminPassword

*************************************************************
*************************************************************
*************************************************************

2025-04-09 04:25:53.616+0000 [id=33]    INFO    jenkins.InitReactorRunner$1#onAttained: Completed initialization
2025-04-09 04:25:53.640+0000 [id=26]    INFO    hudson.lifecycle.Lifecycle#onReady: Jenkins is fully up and running
2025-04-09 04:25:55.969+0000 [id=61]    INFO    h.m.DownloadService$Downloadable#load: Obtained the updated data file for hudson.tasks.M
aven.MavenInstaller
2025-04-09 04:25:55.970+0000 [id=61]    INFO    hudson.util.Retrier#start: Performed the action check updates server successfully at the
 attempt #1
```

On browser goto localhost:8080 and

Paste the 24-character here and click on continue



Unlock Jenkins

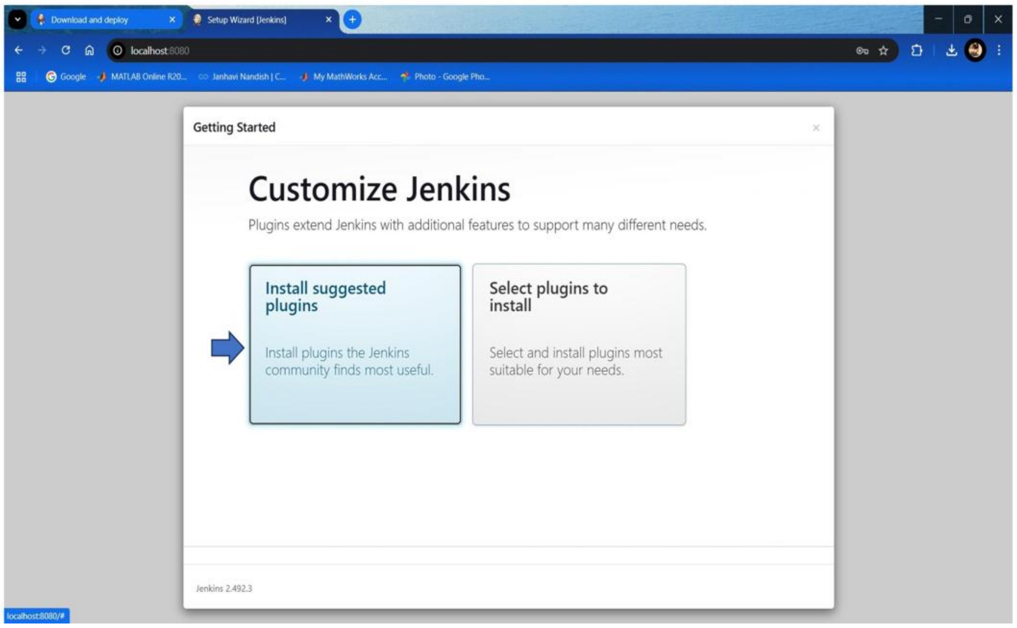
To ensure Jenkins is securely set up by the administrator, a password has been written to the log (not sure where to find it?) and this file on the server:

C:\Users\janha\.jenkins\secrets\initialAdminPassword

Please copy the password from either location and paste it below.

Administrator password

Select install suggested plugins



Getting Started

Customize Jenkins

Plugins extend Jenkins with additional features to support many different needs.

Install suggested plugins

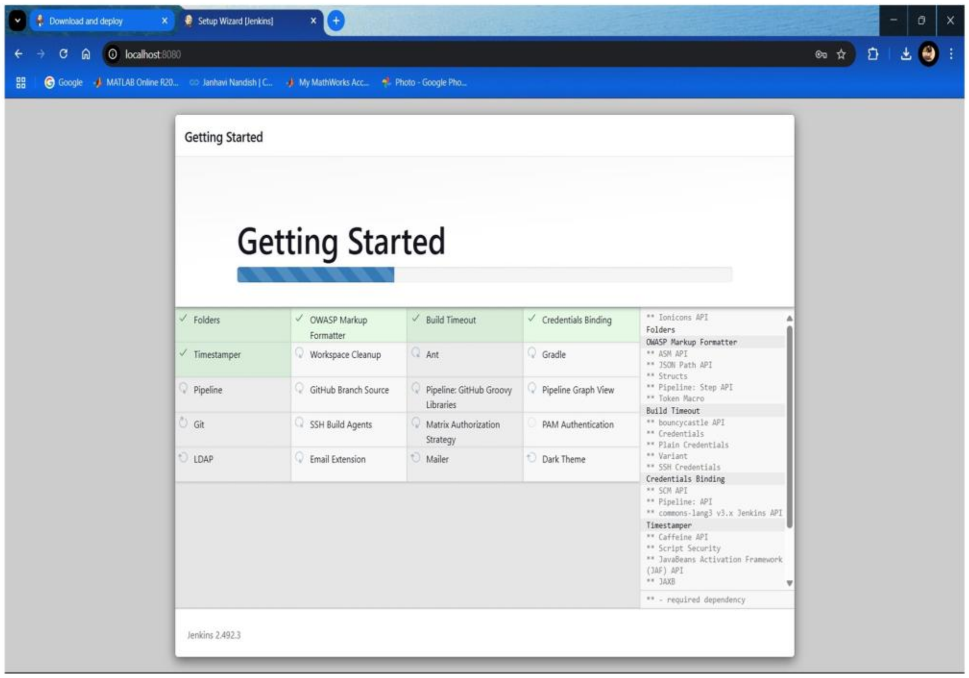
Install plugins the Jenkins community finds most useful.

Select plugins to install

Select and install plugins most suitable for your needs.

Jenkins 2.492.3

Download all the plugins



Getting Started

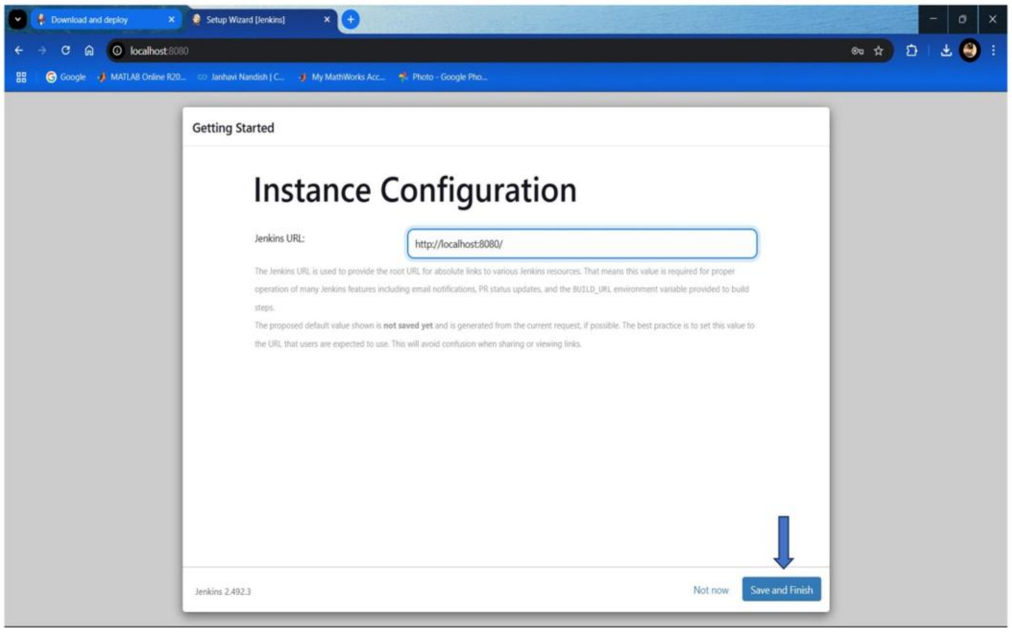
Getting Started

✓ Folders	✓ OWASP Markup Formatter	✓ Build Timeout	✓ Credentials Binding
✓ Timestamper	Workspace Cleanup	Ant	Gradle
Pipeline	GitHub Branch Source	Pipeline: GitHub Groovy Libraries	Pipeline Graph View
Git	SSH Build Agents	Matrix Authorization Strategy	PAM Authentication
LDAP	Email Extension	Mailer	Dark Theme

```
** Ionicons API
Folders
OWASP Markup Formatter
** ASM API
** JSON Path API
** Structs
** Pipeline: Step API
** Token Macro
Build Timeout
** bouncycastle API
** Credentials
** Plain Credentials
** Variant
** SSH Credentials
Credentials Binding
** SCM API
** Pipeline: API
** commons-lang3 v3.x Jenkins API
Timestamper
** Caffeine API
** Script Security
** JavaBeans Activation Framework
(JAF) API
** JAXB
** - required dependency
```

Jenkins 2.492.3

Click on save and finish



Getting Started

Instance Configuration

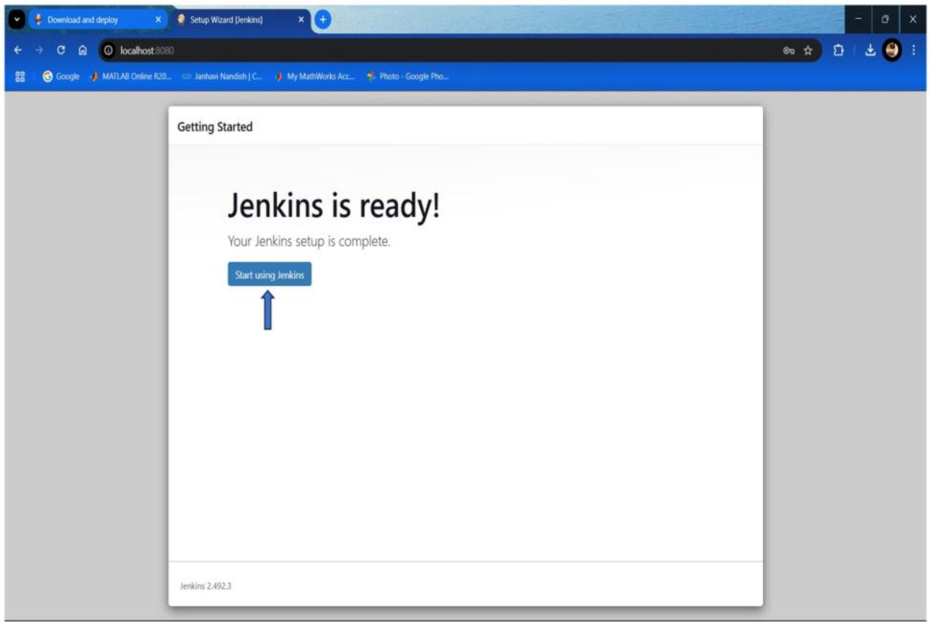
Jenkins URL: http://localhost:8080/

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 2.492.3

Jenkins is ready to use.



Getting Started

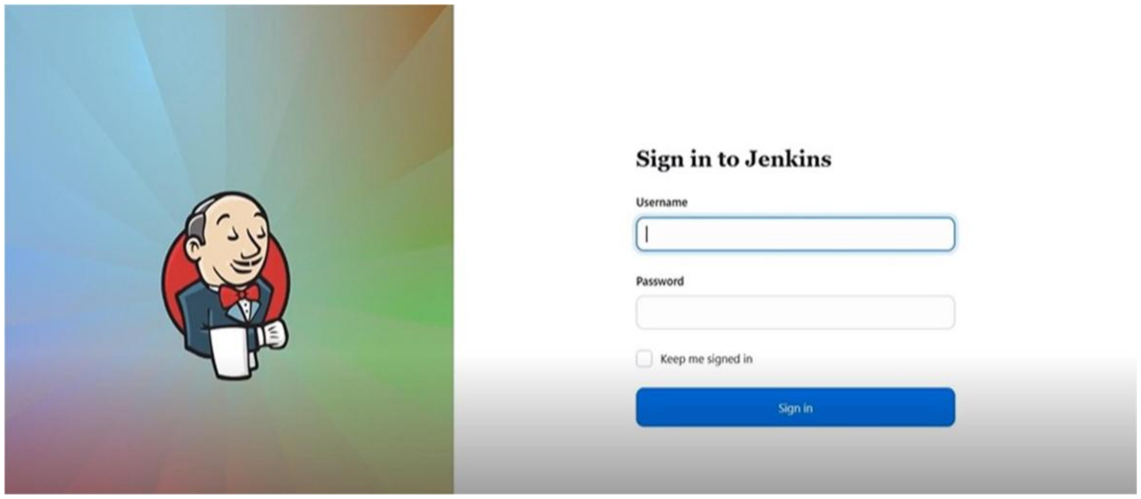
Jenkins is ready!

Your Jenkins setup is complete.

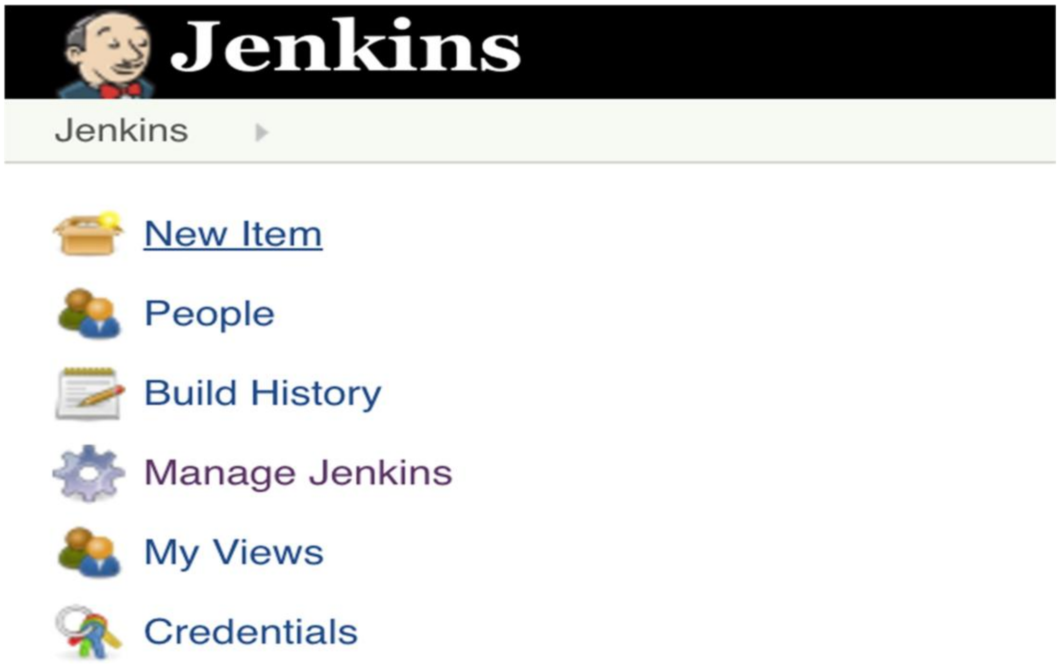
Jenkins 2.492.3

5. Introduction to Jenkins: What is Jenkins?, Installing Jenkins on Local or Cloud Environment, Configuring Jenkins for First Use

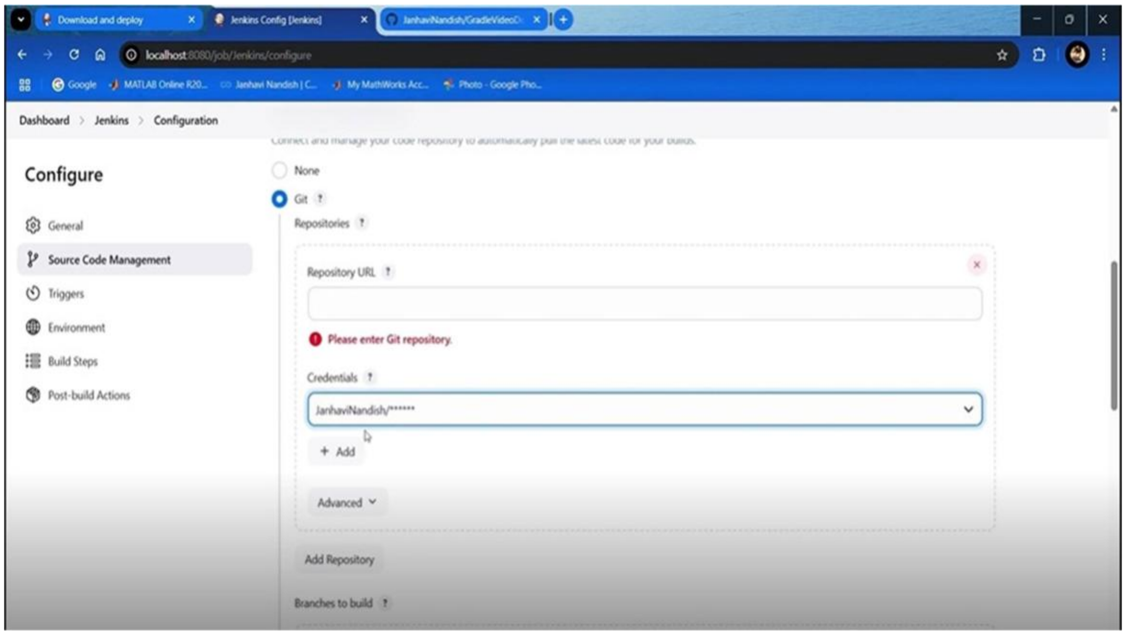
- Login into Jenkins by giving credentials



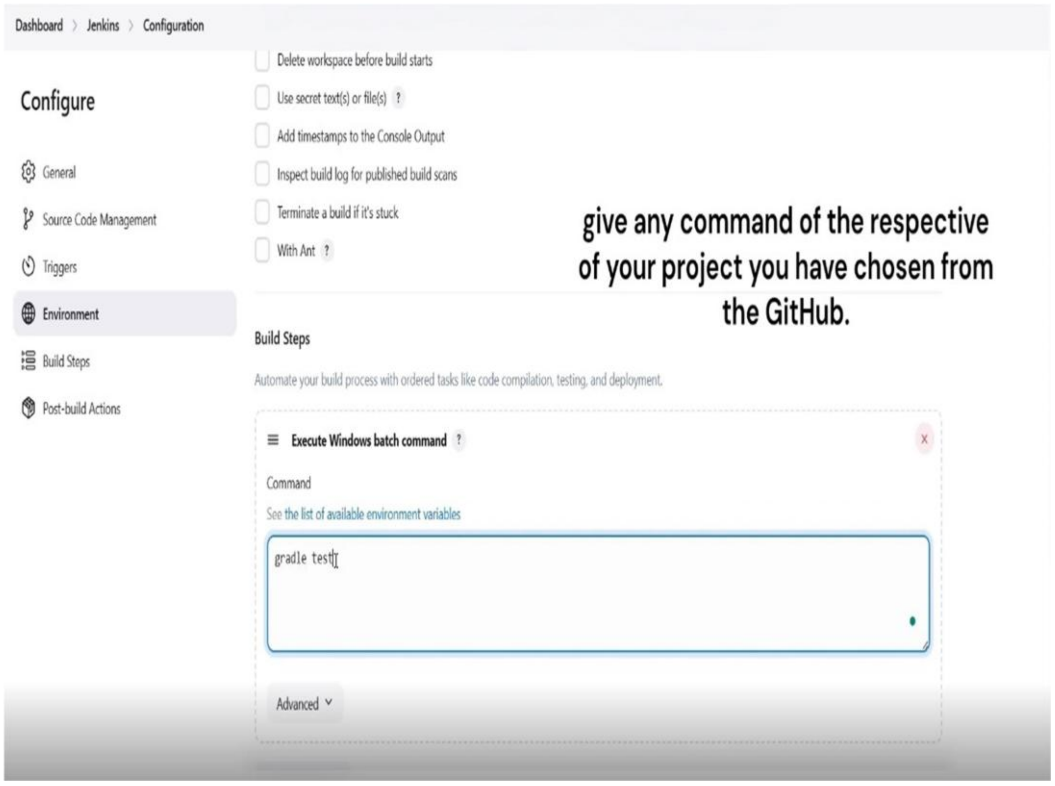
- Select new item



- Add github credentials

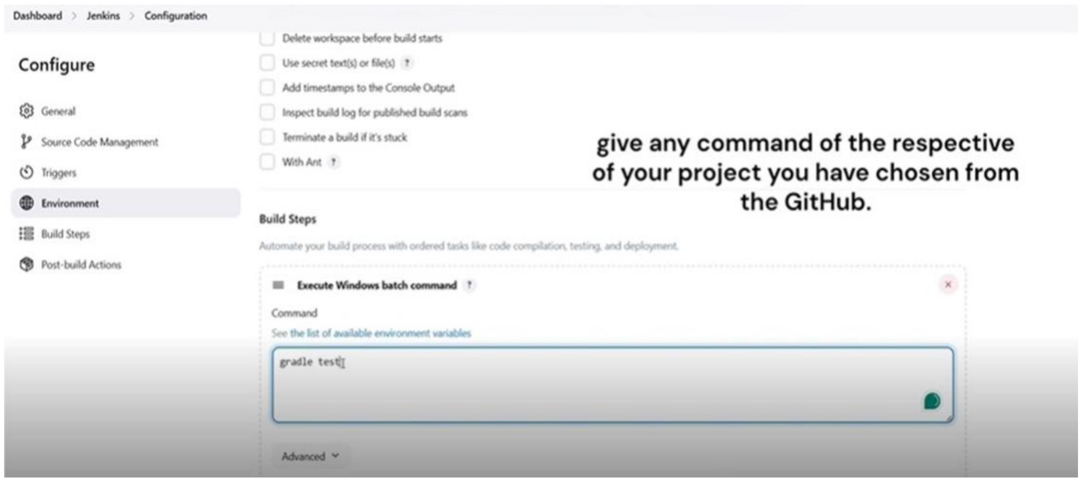


- Select any maven /gradle project


Dashboard > Jenkins > Configuration
Configure
General
Source Code Management
Triggers
Environment
Build Steps
Post-build Actions
Delete workspace before build starts
Use secret text(s) or file(s)
Add timestamps to the Console Output
Inspect build log for published build scans
Terminate a build if it's stuck
With Ant
give any command of the respective of your project you have chosen from the GitHub.
Build Steps
Automate your build process with ordered tasks like code compilation, testing, and deployment.
Execute Windows batch command
Command
See the list of available environment variables
Advanced

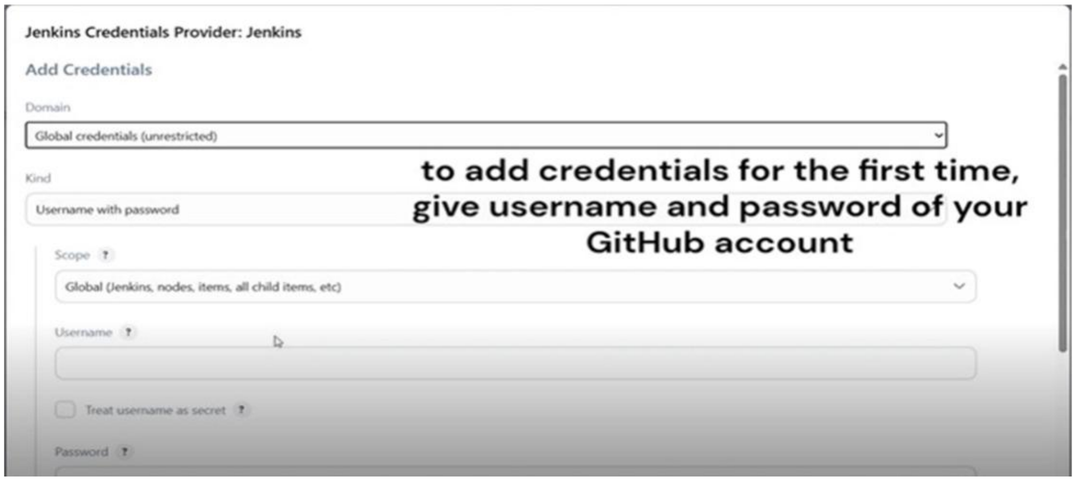
```
gradle test
```

- Give command to run with respect to the project


Dashboard > Jenkins > Configuration
Configure
General
Source Code Management
Triggers
Environment
Build Steps
Post-build Actions
Delete workspace before build starts
Use secret text(s) or file(s)
Add timestamps to the Console Output
Inspect build log for published build scans
Terminate a build if it's stuck
With Ant
Build Steps
Automate your build process with ordered tasks like code compilation, testing, and deployment.
Execute Windows batch command
Command
See the list of available environment variables
Advanced

give any command of the respective of your project you have chosen from the GitHub.

```
gradle test
```

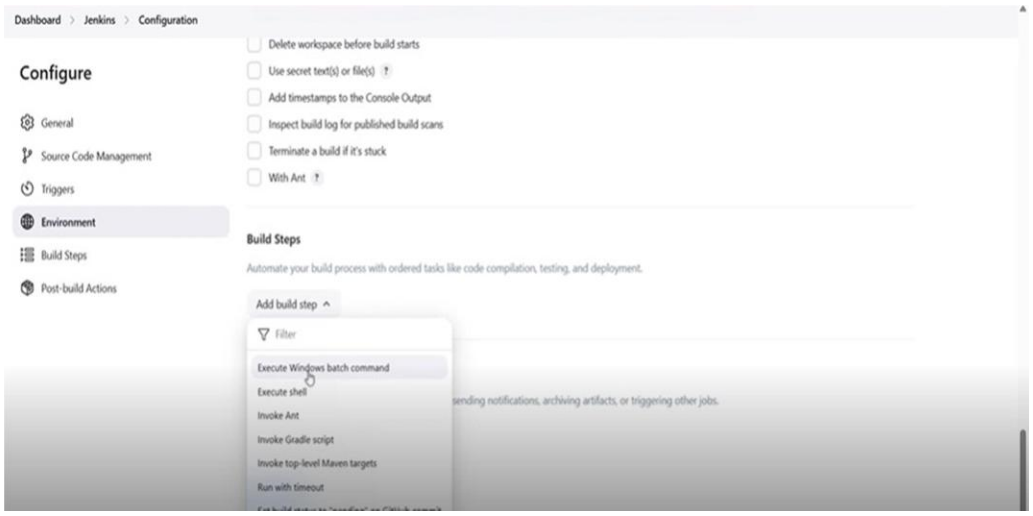


Domain: Global credentials (unrestricted)

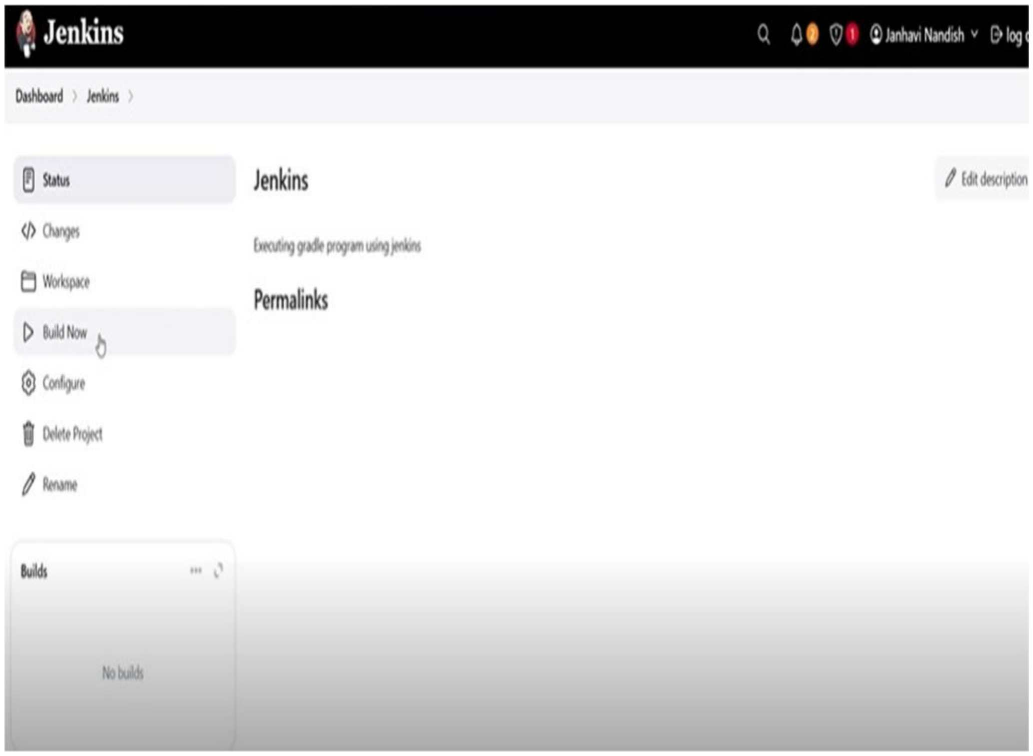
Kind: Username with password

Scope: Global (Jenkins, nodes, items, all child items, etc)

to add credentials for the first time, give username and password of your GitHub account


Dashboard > Jenkins > Configuration
Configure
General
Source Code Management
Triggers
Environment
Build Steps
Post-build Actions
Delete workspace before build starts
Use secret text(s) or file(s)
Add timestamps to the Console Output
Inspect build log for published build scans
Terminate a build if it's stuck
With Ant
Build Steps
Automate your build process with ordered tasks like code compilation, testing, and deployment.
Add build step
Filter
Execute Windows batch command
Execute shell
Invoke Ant
Invoke Gradle script
Invoke top-level Maven targets
Run with timeout
sending notifications, archiving artifacts, or triggering other jobs.

- Build the project

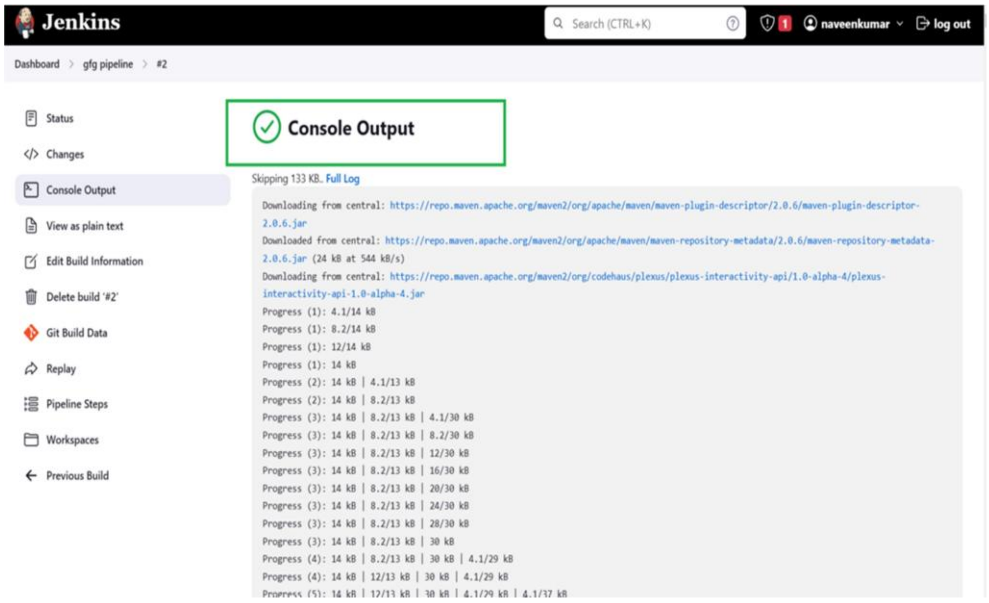


Jenkins

Executing gradle program using jenkins

Permalinks

- At console output you can see successfully



Console Output

Skipping 133 KB.. Full Log

```
Downloading from central: https://repo.maven.apache.org/maven2/org/apache/maven/maven-plugin-descriptor/2.0.6/maven-plugin-descriptor-
2.0.6.jar
Downloaded from central: https://repo.maven.apache.org/maven2/org/apache/maven/maven-repository-metadata/2.0.6/maven-repository-metadata-
2.0.6.jar (24 kB at 544 kB/s)
Downloading from central: https://repo.maven.apache.org/maven2/org/codehaus/plexus/plexus-interactivity-api/1.0-alpha-4/plexus-
interactivity-api-1.0-alpha-4.jar
Progress (1): 4.1/14 kB
Progress (1): 8.2/14 kB
Progress (1): 12/14 kB
Progress (1): 14 kB
Progress (2): 14 kB | 4.1/13 kB
Progress (2): 14 kB | 8.2/13 kB
Progress (3): 14 kB | 8.2/13 kB | 4.1/30 kB
Progress (3): 14 kB | 8.2/13 kB | 8.2/30 kB
Progress (3): 14 kB | 8.2/13 kB | 12/30 kB
Progress (3): 14 kB | 8.2/13 kB | 16/30 kB
Progress (3): 14 kB | 8.2/13 kB | 20/30 kB
Progress (3): 14 kB | 8.2/13 kB | 24/30 kB
Progress (3): 14 kB | 8.2/13 kB | 28/30 kB
Progress (3): 14 kB | 8.2/13 kB | 30 kB
Progress (4): 14 kB | 8.2/13 kB | 30 kB | 4.1/29 kB
Progress (4): 14 kB | 12/13 kB | 30 kB | 4.1/29 kB
Progress (5): 14 kB | 12/13 kB | 30 kB | 4.1/29 kB | 4.1/37 kB
```

6. Configuration Management with Ansible: Basics of Ansible: Inventory, Playbooks, and Modules, Automating Server Configurations with Playbooks, Hands-On: Writing and Running a Basic Playbook

What Is Ansible?

Ansible is an open-source IT automation and configuration management tool. It allows you to manage multiple servers and perform tasks such as:

- **Configuration Management:** Automate the configuration of servers.
- **Application Deployment:** Deploy applications consistently.

Key Concepts in Ansible

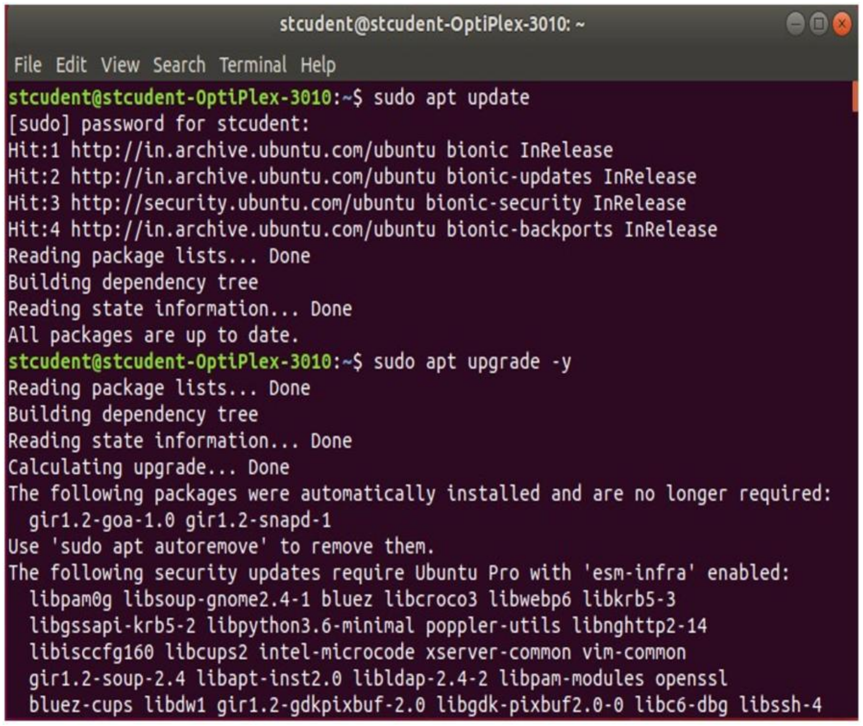
- **Inventory:**
  An inventory is a file (usually in INI or YAML format) that lists the hosts (or groups of hosts) you want to manage. It tells Ansible which machines to target.
- **Playbook:**
  A playbook is a YAML file that defines a set of tasks to be executed on your target hosts. It is the heart of Ansible automation. In a playbook, you specify:

Installing Ansible in UBUNTU

Step 1: Update Your System

Open your terminal and run:

```
sudo apt update
sudo apt upgrade -y
```



```
stcudent@stcudent-OptiPlex-3010:~$ sudo apt update
[sudo] password for stcudent:
Hit:1 http://in.archive.ubuntu.com/ubuntu bionic InRelease
Hit:2 http://in.archive.ubuntu.com/ubuntu bionic-updates InRelease
Hit:3 http://security.ubuntu.com/ubuntu bionic-security InRelease
Hit:4 http://in.archive.ubuntu.com/ubuntu bionic-backports InRelease
Reading package lists... Done
Building dependency tree
Reading state information... Done
All packages are up to date.
stcudent@stcudent-OptiPlex-3010:~$ sudo apt upgrade -y
Reading package lists... Done
Building dependency tree
Reading state information... Done
Calculating upgrade... Done
The following packages were automatically installed and are no longer required:
  gir1.2-goa-1.0 gir1.2-snapd-1
Use 'sudo apt autoremove' to remove them.
The following security updates require Ubuntu Pro with 'esm-infra' enabled:
  libpam0g libsoup-gnome2.4-1 bluez libcroco3 libwebp6 libkrb5-3
  libgssapi-krb5-2 libpython3.6-minimal poppler-utils libnghttp2-14
  libisccfg160 libcups2 intel-microcode xserver-common vim-common
  gir1.2-soup-2.4 libapt-inst2.0 libldap-2.4-2 libpam-modules openssl
  bluez-cups libdw1 gir1.2-gdkpixbuf-2.0 libgdk-pixbuf2.0-0 libc6-dbg libssh-4
```

Remove the lock file when process is not running

- sudo rm /var/lib/apt/lists/lock
- sudo rm /var/cache/apt/archives/lock
- sudo rm /var/lib/dpkg/lock*

Step 2: Install Ansible

Install Ansible using apt:

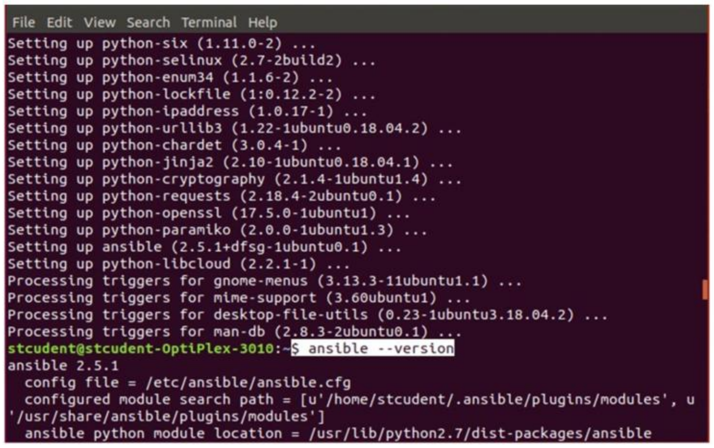
```
sudo apt install ansible -y
```

Verify the installation by checking the version:

```
ansible --version
```

Expected Output Example:

```
ansible 2.9.x
  config file = /etc/ansible/ansible.cfg
  ...
```



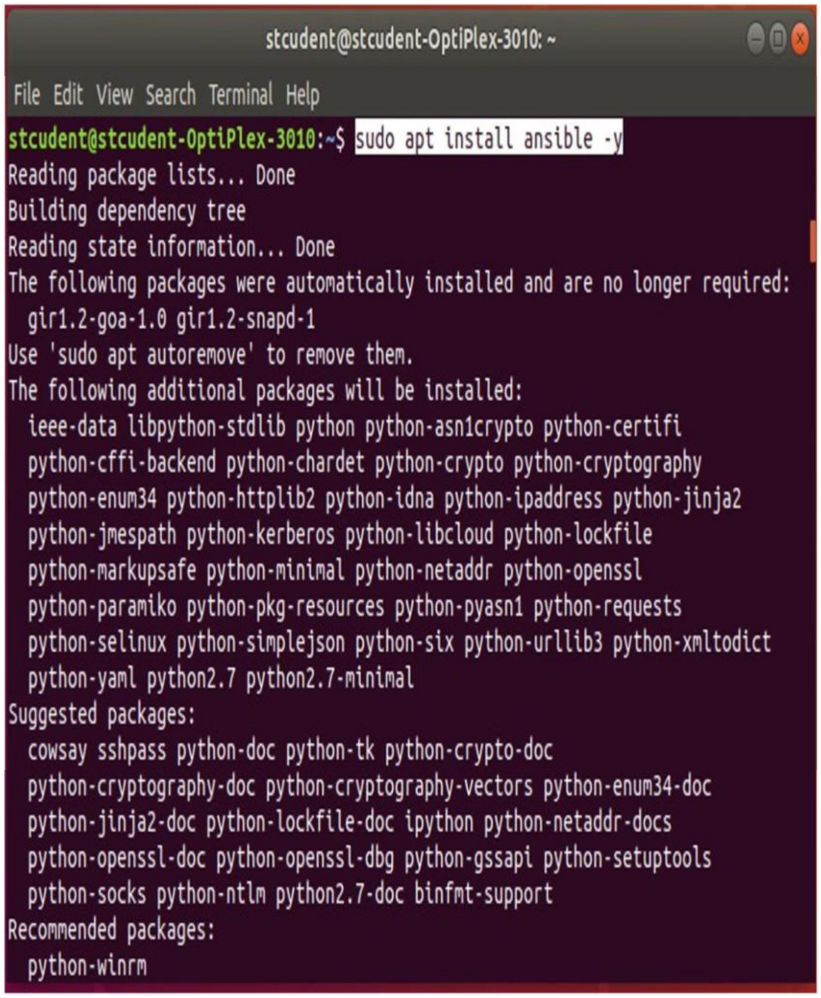
```
Setting up python-six (1.11.0-2) ...
Setting up python-selinux (2.7-2build2) ...
Setting up python-enum34 (1.1.6-2) ...
Setting up python-lockfile (1:0.12.2-2) ...
Setting up python-ipaddress (1.0.17-1) ...
Setting up python-urllib3 (1.22-1ubuntu0.18.04.2) ...
Setting up python-chardet (3.0.4-1) ...
Setting up python-jinja2 (2.10-1ubuntu0.18.04.1) ...
Setting up python-cryptography (2.1.4-1ubuntu1.4) ...
Setting up python-requests (2.18.4-2ubuntu0.1) ...
Setting up python-openssl (17.5.0-1ubuntu1) ...
Setting up python-paramiko (2.0.0-1ubuntu1.3) ...
Setting up ansible (2.5.1+dfsg-1ubuntu0.1) ...
Setting up python-libcloud (2.2.1-1) ...
Processing triggers for gnome-menus (3.13.3-11ubuntu1.1) ...
Processing triggers for mime-support (3.60ubuntu1) ...
Processing triggers for desktop-file-utils (0.23-1ubuntu3.18.04.2) ...
Processing triggers for man-db (2.8.3-2ubuntu0.1) ...
stcudent@stcudent-OptiPlex-3010:~$ ansible --version
ansible 2.5.1
  config file = /etc/ansible/ansible.cfg
  configured module search path = [u'/home/stcudent/.ansible/plugins/modules', u
'/usr/share/ansible/plugins/modules']
  ansible python module location = /usr/lib/python2.7/dist-packages/ansible
```

Creating an Ansible Inventory

Step 1: Create an Inventory File

1. Open your text editor to create a file called `hosts.ini`:
2. `nano hosts.ini`
3. Add the following content to define the local host:
4. `[local]`
5. `localhost ansible_connection=local`

Save the file by pressing **Ctrl+O** then **Enter**, and exit with **Ctrl+X**.



stcudent@stcudent-OptiPlex-3010: ~
File Edit View Search Terminal Help

```
stcudent@stcudent-OptiPlex-3010:~$ sudo apt install ansible -y
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following packages were automatically installed and are no longer required:
  gir1.2-goa-1.0 gir1.2-snapd-1
Use 'sudo apt autoremove' to remove them.
The following additional packages will be installed:
  ieee-data libpython-stdlib python python-asn1crypto python-certifi
  python-cffi-backend python-chardet python-crypto python-cryptography
  python-enum34 python-httplib2 python-idna python-ipaddress python-jinja2
  python-jmespath python-kerberos python-libcloud python-lockfile
  python-markupsafe python-minimal python-netaddr python-openssl
  python-paramiko python-pkg-resources python-pyasn1 python-requests
  python-selinux python-simplejson python-six python-urllib3 python-xmltodict
  python-yaml python2.7 python2.7-minimal
Suggested packages:
  cowsay sshpass python-doc python-tk python-crypto-doc
  python-cryptography-doc python-cryptography-vectors python-enum34-doc
  python-jinja2-doc python-lockfile-doc ipython python-netaddr-docs
  python-openssl-doc python-openssl-dbg python-gssapi python-setuptools
  python-socks python-ntlm python2.7-doc binfmt-support
Recommended packages:
  python-winrm
```

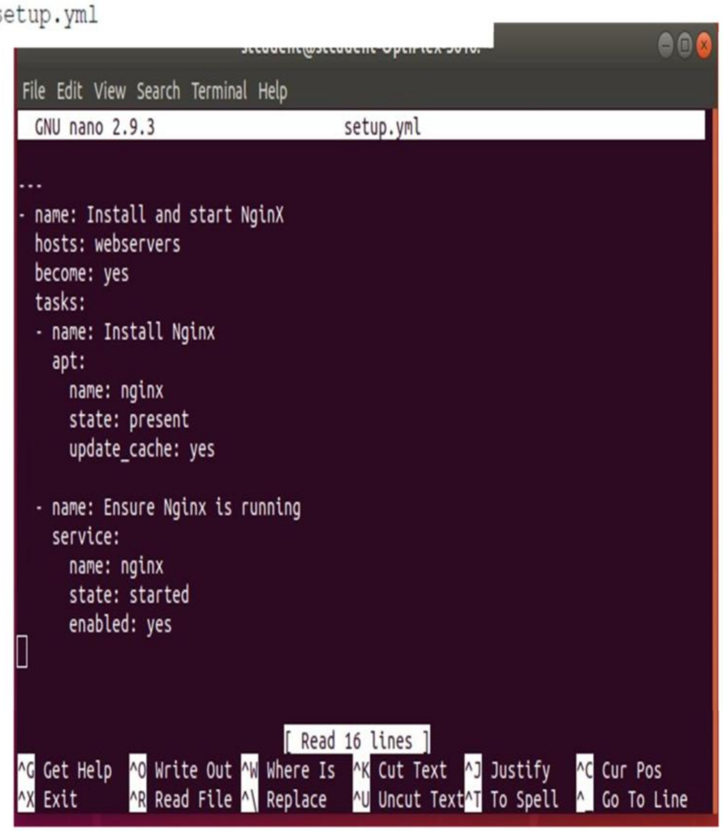
Writing a Basic Ansible Playbook

You will now create a simple playbook that performs two common tasks:

- Updating the apt cache
- Installing a package

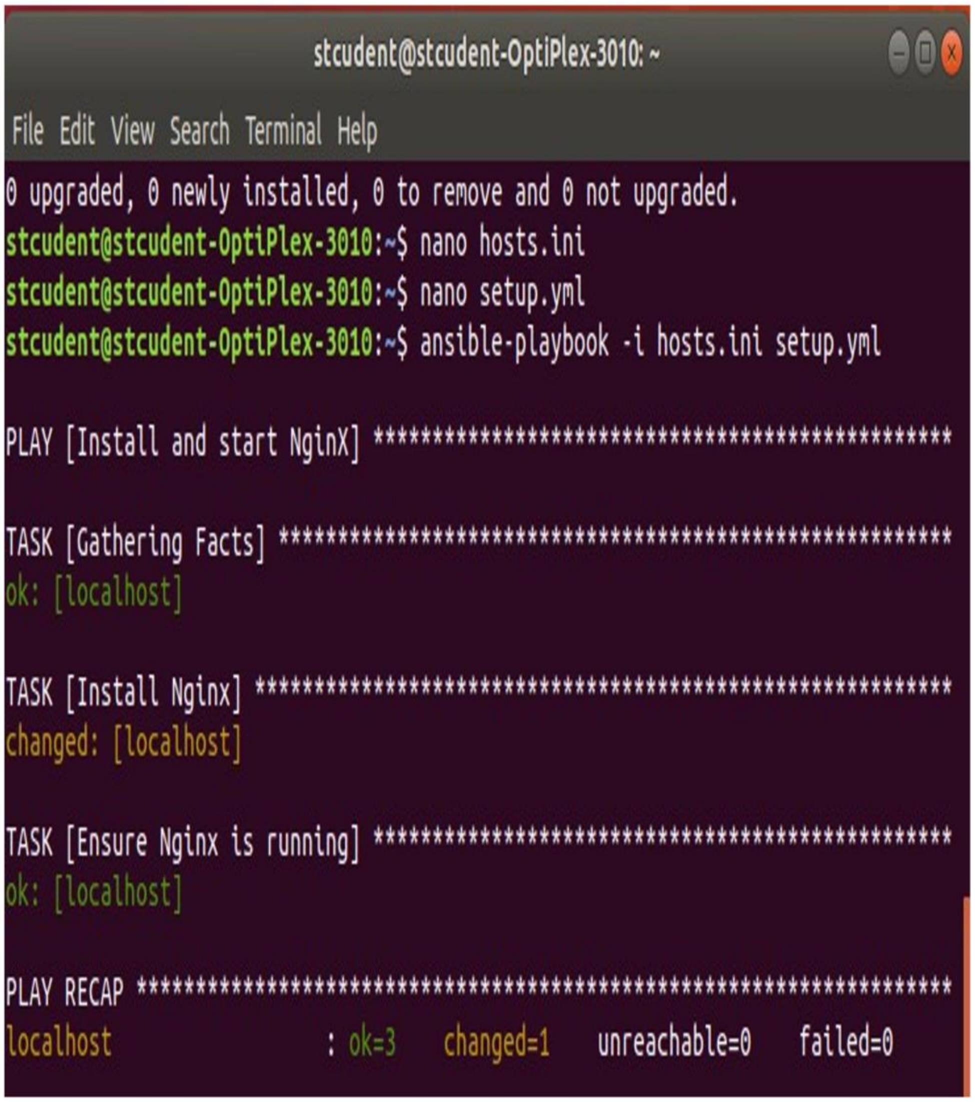
Step 1: Create the Playbook File

1. Open your text editor to create a file called `setup.yml`:
2. `nano setup.yml`



```
  GNU nano 2.9.3                    setup.yml

---
- name: Install and start NginX
  hosts: webservers
  become: yes
  tasks:
  - name: Install Nginx
    apt:
      name: nginx
      state: present
      update_cache: yes

  - name: Ensure Nginx is running
    service:
      name: nginx
      state: started
      enabled: yes

                              [ Read 16 lines ]
^G Get Help  ^O Write Out ^W Where Is  ^K Cut Text  ^J Justify   ^C Cur Pos
^X Exit      ^R Read File ^\ Replace   ^U Uncut Text^T To Spell  ^_ Go To Line
```

Output:


stcudent@stcudent-OptiPlex-3010: ~
File Edit View Search Terminal Help

```
0 upgraded, 0 newly installed, 0 to remove and 0 not upgraded.
stcudent@stcudent-OptiPlex-3010:~$ nano hosts.ini
stcudent@stcudent-OptiPlex-3010:~$ nano setup.yml
stcudent@stcudent-OptiPlex-3010:~$ ansible-playbook -i hosts.ini setup.yml

PLAY [Install and start NginX] *************************************************

TASK [Gathering Facts] *********************************************************
ok: [localhost]

TASK [Install Nginx] ***********************************************************
changed: [localhost]

TASK [Ensure Nginx is running] *************************************************
ok: [localhost]

PLAY RECAP *********************************************************************
localhost                  : ok=3    changed=1    unreachable=0    failed=0
```

9.Introduction to Azure DevOps: Overview of Azure DevOps Services, Setting Up an Azure DevOps Account and Project

Topics Covered:

- Overview of Azure DevOps Services
- Setting Up an Azure DevOps Account and Project

1. Overview of Azure DevOps

Azure DevOps is a comprehensive suite of cloud-based services that supports the entire software development lifecycle. It provides tools for planning, developing, testing, delivering, and monitoring applications. The primary services offered include:

1.1 Azure Repos

Version control tools that allow you to host Git repositories or use Team Foundation Version Control (TFVC). Features include:

- Pull requests
- Branch policies
- Code reviews

1.2 Azure Pipelines

A CI/CD service that automates builds, tests, and deployments. It supports:

- Multiple programming languages
- Multiple platforms (Linux, Windows, macOS)
- Integration with popular build systems

1.3 Azure Boards

A work tracking system to manage:

- Work items
- Sprints and backlogs
- Kanban boards and agile reports

1.4 Azure Test Plans

A solution for managing and executing tests. Provides:

- Manual and exploratory testing tools

- Defect tracking
- Test case management

1.5 Azure Artifacts

A package management service that enables you to:

- Create, host, and share packages (e.g., Maven, npm, NuGet, Python)
- Integrate with CI/CD pipelines

These services are integrated with each other and with third-party tools, providing a cohesive and extensible DevOps ecosystem.

2. Setting Up an Azure DevOps Account

Before using Azure DevOps, you must create an account and set up your organization.

Step 1: Sign Up for an Azure DevOps Account

1. **Open Your Web Browser:**
   Navigate to https://dev.azure.com
2. **Sign In or Create a Microsoft Account:**
   - If you already have a Microsoft account (e.g., Outlook, Hotmail, Office 365), click **"Sign in"**
   - If not, click **"Create one!"** and follow the instructions
3. **Accept Terms and Conditions:**
   - If prompted, review and accept Microsoft's terms of service

Step 2: Create an Azure DevOps Organization

1. **Create a New Organization:**
   - After signing in, you'll be prompted to create a new organization
   - Enter a unique name (e.g., MyPersonalOrg or YourCompanyDevOps)
   - Select a **Region** closest to your location
   - Click **"Continue"** or **"Create"**
2. **Review Your Organization's Dashboard:**
   - After creation, you'll land on the dashboard with navigation to:
     - **Repos**

    - **Pipelines**
    - **Boards**
    - **Test Plans**
    - **Artifacts**

3. Creating an Azure DevOps Project

A project in Azure DevOps is a container for all your source code, pipelines, work items, and other DevOps resources.

Step 1: Create a New Project

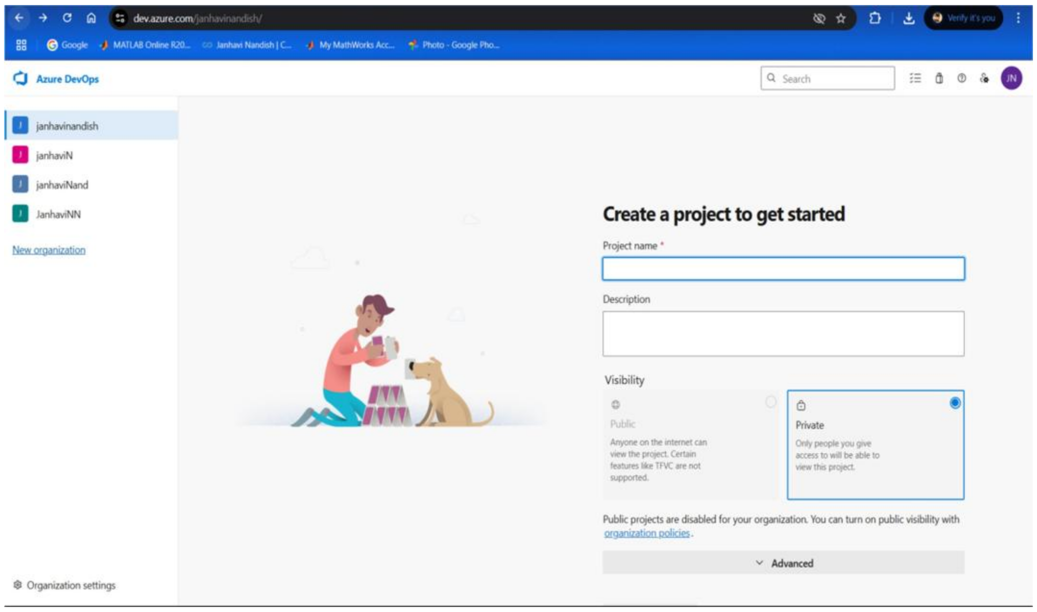
1. **Navigate to "New Project":**
    - From your organization's dashboard, click **"New Project"**
2. **Configure Your Project:**
    - **Project Name:** Enter a descriptive name (e.g., HelloDevOps)
    - **Description:** Optionally provide a short summary
    - **Visibility:**
        - Choose **"Private"** to restrict access
        - Choose **"Public"** if it can be viewed by anyone
    - **Advanced Options:**
        - Choose a version control system (default: Git)
        - Select a work item process (Agile, Scrum, Basic, CMMI – Agile recommended for beginners)
    - Click **"Create"**

Step 2: Explore Your Project Dashboard

1. **Project Overview:**
   After creation, you'll see a dashboard with access to:
    - **Repos:** Manage source code
    - **Pipelines:** Set up CI/CD
    - **Boards:** Track and plan work
    - **Test Plans:** Manage tests

- **Artifacts:** Package management

2. **Familiarize Yourself with the Interface:**
   - Click through each section (e.g., Repos, Pipelines, Boards) to understand its features



Create a project to get started

Project name *

Description

Visibility

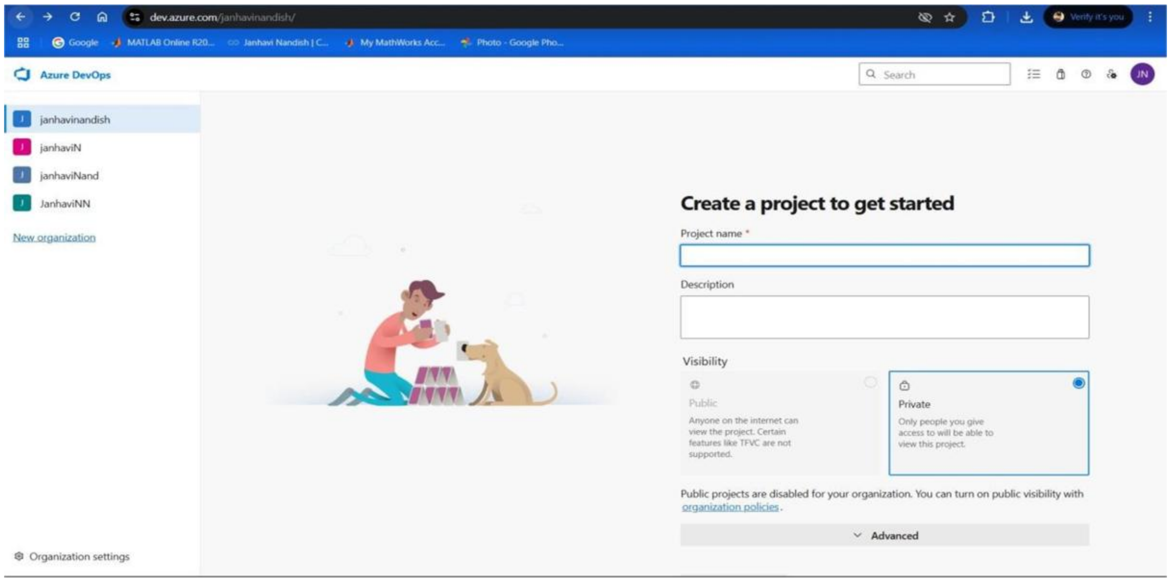
Public
Anyone on the internet can view the project. Certain features like TFVC are not supported.

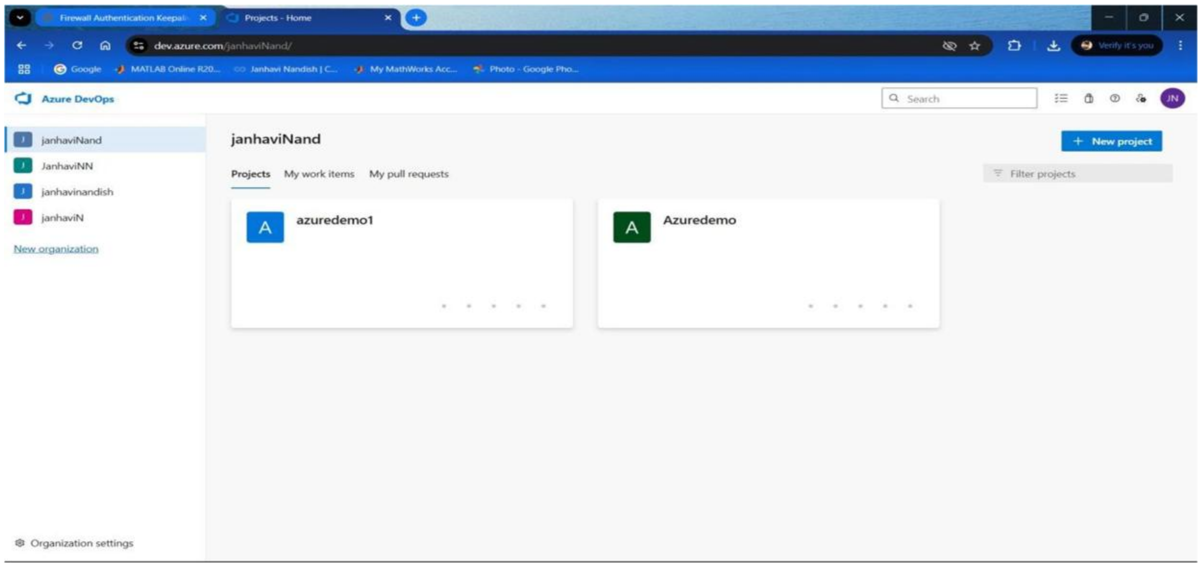
Private
Only people you give access to will be able to view this project.

Public projects are disabled for your organization. You can turn on public visibility with organization policies.

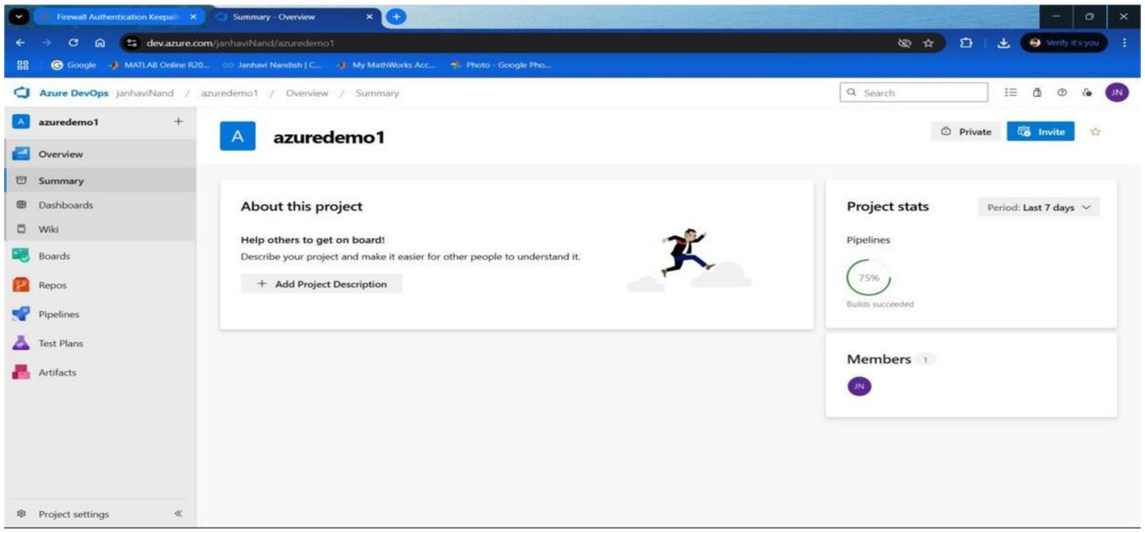
10. Building a Maven/Gradle Project with Azure Pipelines, Integrating Code Repositories (e.g.GitHub, Azure Repos), Running Unit Tests and Generating Reports.

Step 1: After logging into **Azure Devops**, click your organization name (top-left)

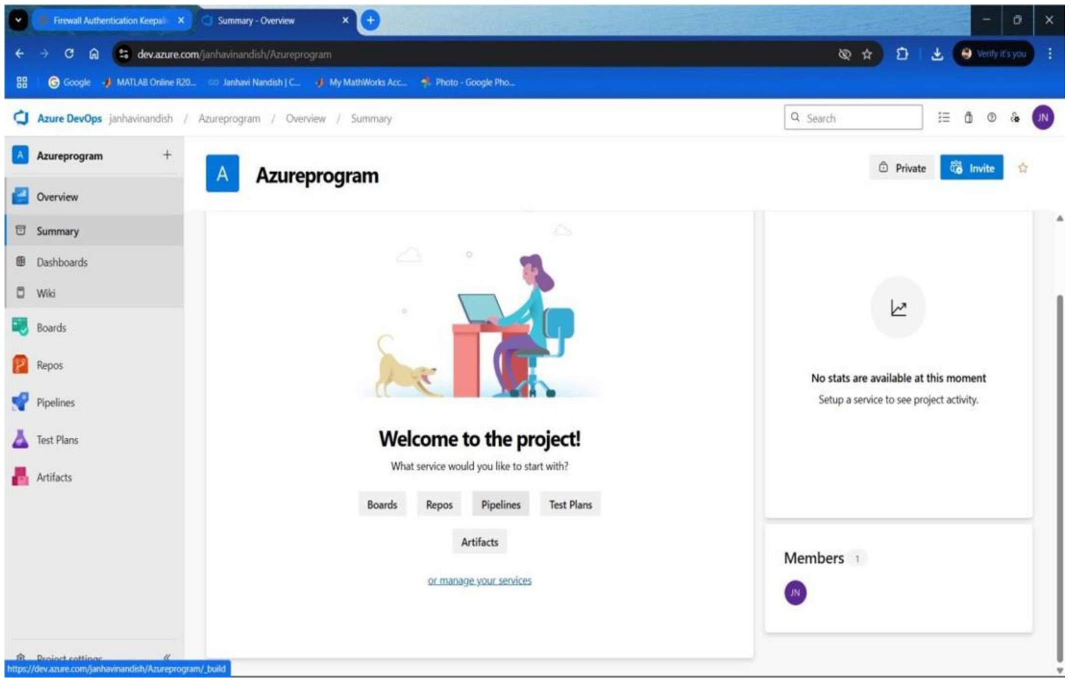

dev.azure.com/janhavinandish/
Azure DevOps
Search
janhavinandish
janhaviN
janhaviNand
JanhaviNN
New organization
Organization settings
Create a project to get started
Project name *
Description
Visibility
Public
Anyone on the internet can view the project. Certain features like TFVC are not supported.
Private
Only people you give access to will be able to view this project.
Public projects are disabled for your organization. You can turn on public visibility with organization policies.
Advanced



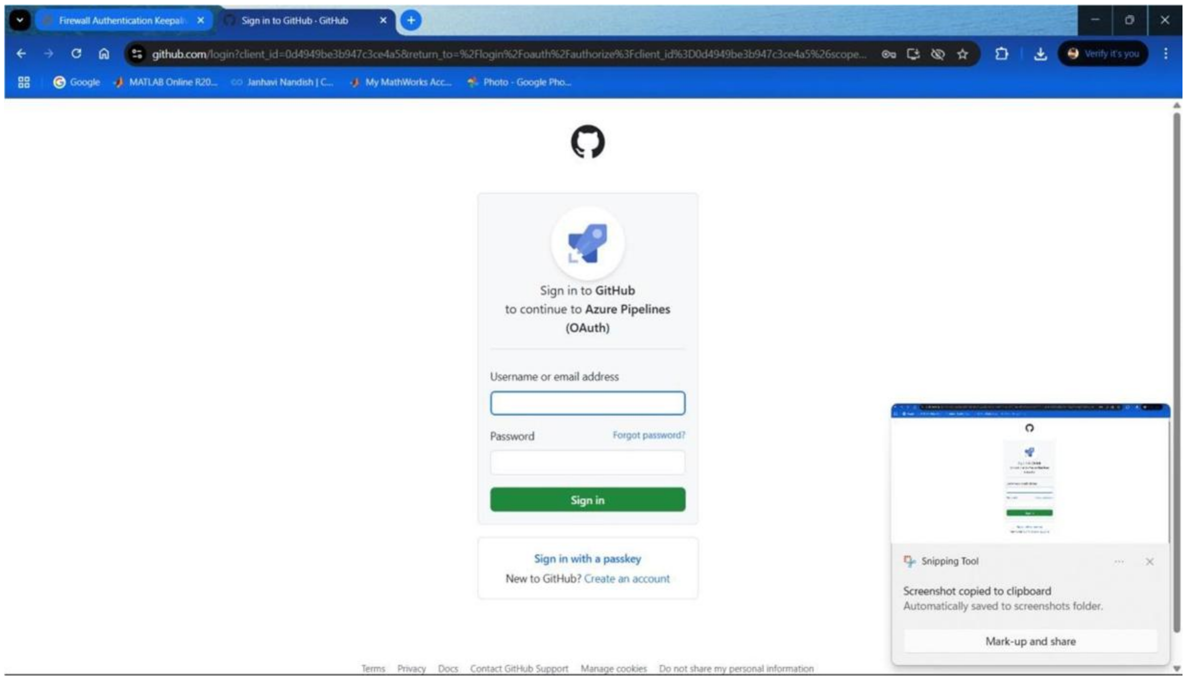
Step 2: From your Azure Devops homepage, create a new project (e.g., demo1). This is where your pipeline and linked services will reside.



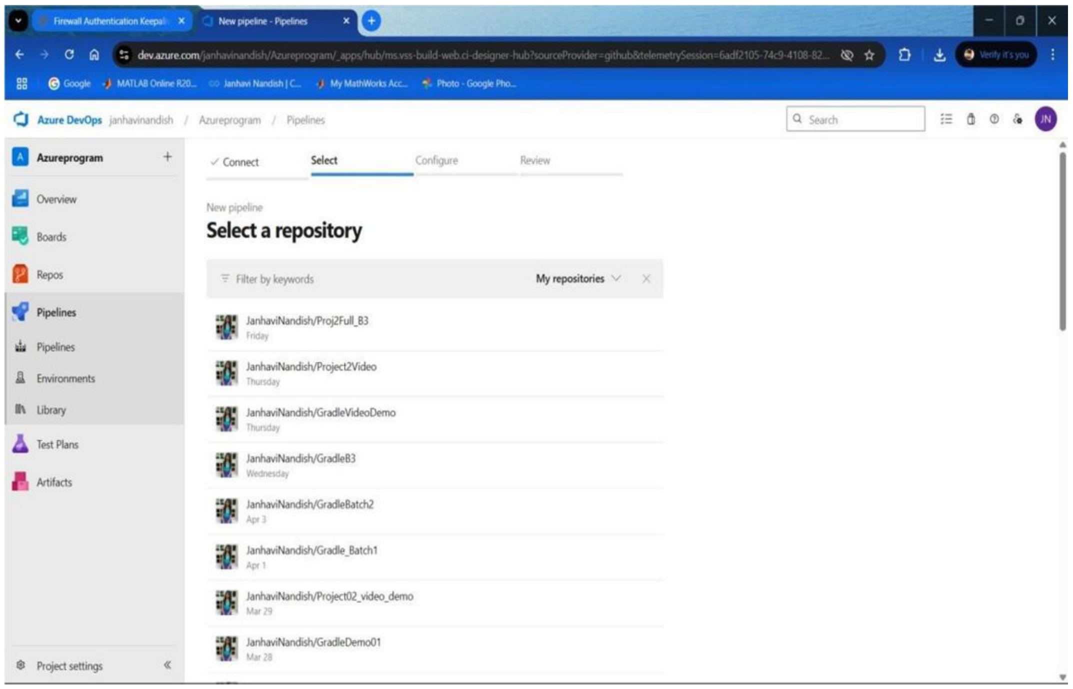
Step 3: Inside the project, go to **Pipelines > Pipelines** and click **Create Pipeline** to start the CI/CD setup wizard.



Step 4: Connect to Your GitHub Repository: When prompted for your code source, select **GitHub**. You'll be redirected to authenticate and allow access

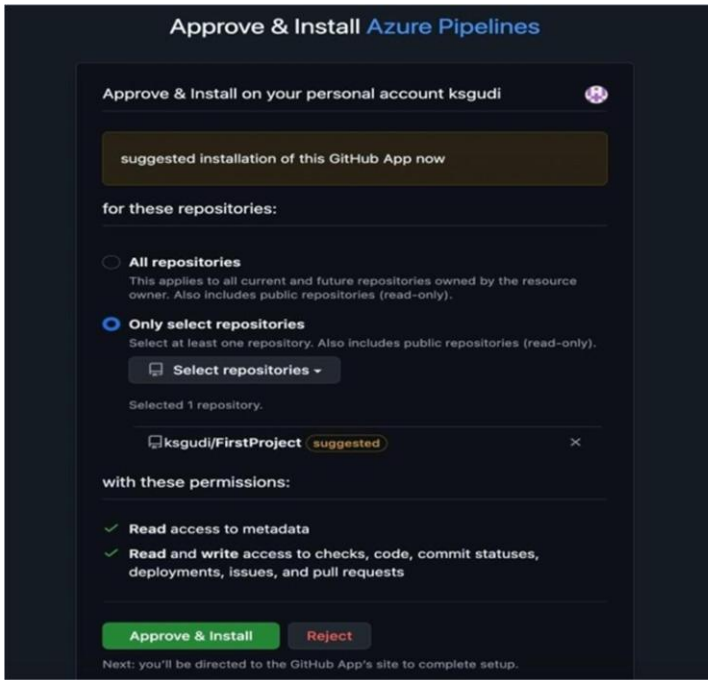

Sign in to GitHub
to continue to Azure Pipelines (OAuth)
Username or email address
Password
Forgot password?
Sign in
Sign in with a passkey
New to GitHub? Create an account
Snipping Tool
Screenshot copied to clipboard
Automatically saved to screenshots folder.
Mark-up and share

Step 5: Choose Your Repository : Select the Maven project repository you want to build, for .



JanhaviNandish/Proj2Full_B3 Friday
JanhaviNandish/Project2Video Thursday
JanhaviNandish/GradleVideoDemo Thursday
JanhaviNandish/GradleB3 Wednesday
JanhaviNandish/GradleBatch2 Apr 3
JanhaviNandish/Gradle_Batch1 Apr 1
JanhaviNandish/Project02_video_demo Mar 29
JanhaviNandish/GradleDemo01 Mar 28

Steps:6 Approve GitHub Integration: Approve access for Azure Pipelines to your GitHub repository. Choose "Only select repositories" and confirm access for First Project.



Approve & Install Azure Pipelines

Approve & Install on your personal account ksgudi

suggested installation of this GitHub App now

for these repositories:

All repositories
This applies to all current and future repositories owned by the resource owner. Also includes public repositories (read-only).

Only select repositories
Select at least one repository. Also includes public repositories (read-only).

Select repositories

Selected 1 repository.

ksgudi/FirstProject suggested

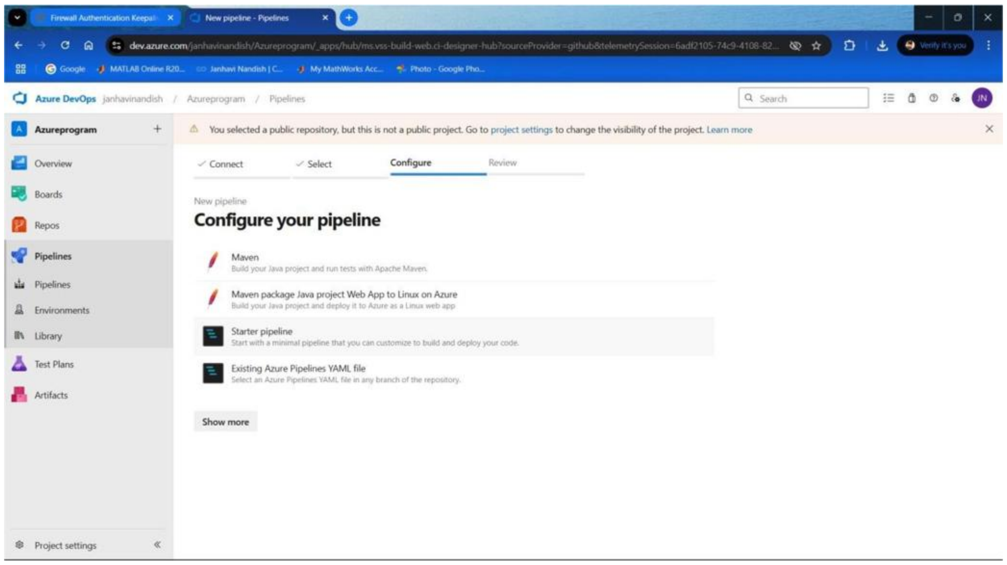
with these permissions:

✓ Read access to metadata
✓ Read and write access to checks, code, commit statuses, deployments, issues, and pull requests

Approve & Install Reject

Next: you'll be directed to the GitHub App's site to complete setup.

Step 7: Select Maven Pipeline Template: When prompted to configure the pipeline, choose **Maven/Gradle** to auto-generate a pipeline template suitable for building and testing Maven/Gradle project.



dev.azure.com/janhavinandish/Azureprogram/_apps/hub/ms.vss-build-web.ci-designer-hub?sourceProvider=github&telemetrySession=6adf2105-74c9-4108-82...

You selected a public repository, but this is not a public project. Go to project settings to change the visibility of the project. Learn more

✓ Connect ✓ Select Configure Review

New pipeline

Configure your pipeline

Maven
Build your Java project and run tests with Apache Maven.

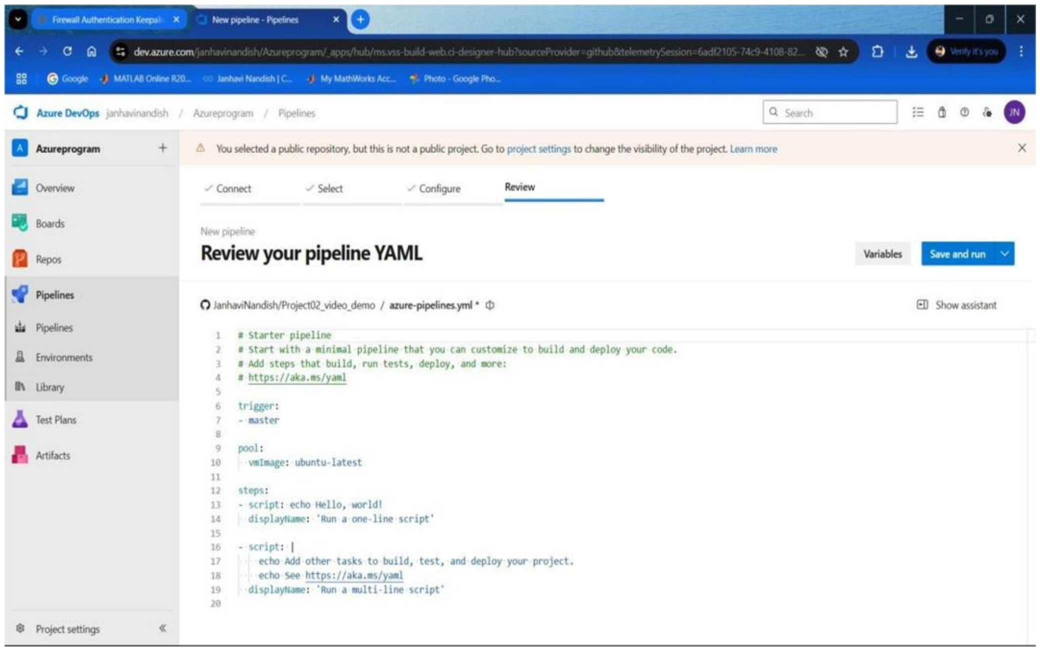
Maven package Java project Web App to Linux on Azure
Build your Java project and deploy it to Azure as a Linux web app

Starter pipeline
Start with a minimal pipeline that you can customize to build and deploy your code.

Existing Azure Pipelines YAML file
Select an Azure Pipelines YAML file in any branch of the repository.

Step 8: Review and Customize YAML: Review the auto-generated azure-pipelines.yml.

Ensure it contains the Maven/Gradle task with appropriate goals like clean install and test result file paths.



You selected a public repository, but this is not a public project. Go to project settings to change the visibility of the project. Learn more

New pipeline

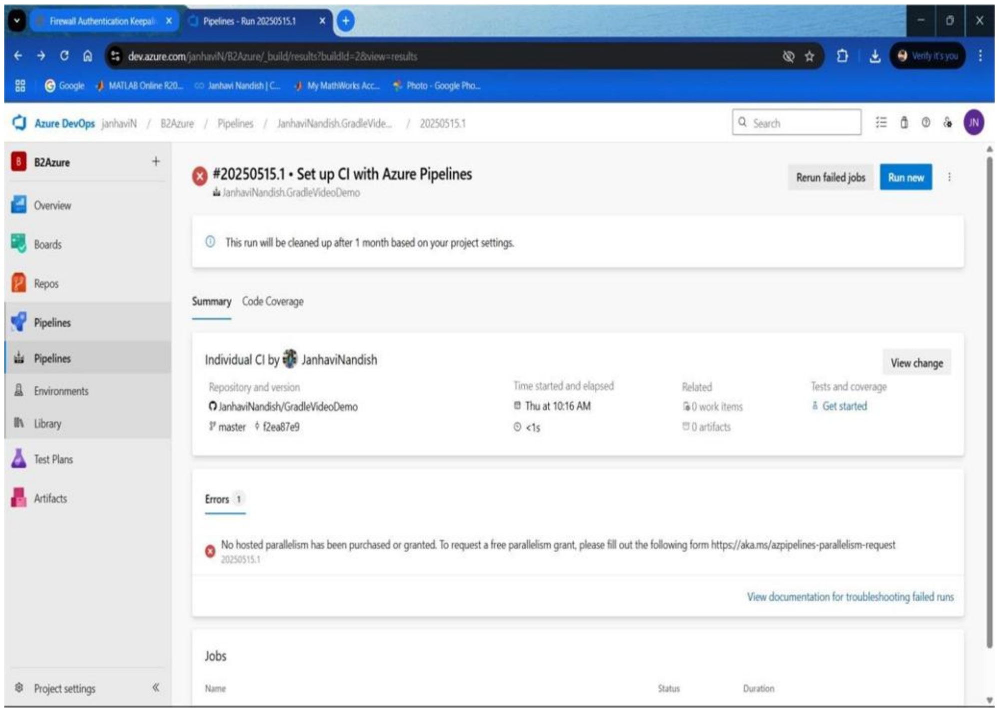
Review your pipeline YAML

JanhaviNandish/Project02_video_demo / azure-pipelines.yml *

```yaml
# Starter pipeline
# Start with a minimal pipeline that you can customize to build and deploy your code.
# Add steps that build, run tests, deploy, and more:
# https://aka.ms/yaml

trigger:
- master

pool:
  vmImage: ubuntu-latest

steps:
- script: echo Hello, world!
  displayName: 'Run a one-line script'

- script: |
    echo Add other tasks to build, test, and deploy your project.
    echo See https://aka.ms/yaml
  displayName: 'Run a multi-line script'
```

Step 9: Click **Save and Run** to trigger the pipeline.

⚠️ If the run fails with a message like "No hosted parallelism has been purchased," you must request free hosted agent access.



#20250515.1 • Set up CI with Azure Pipelines

JanhaviNandish.GradleVideoDemo

This run will be cleaned up after 1 month based on your project settings.

Individual CI by JanhaviNandish

Repository and version	Time started and elapsed	Related	Tests and coverage
JanhaviNandish/GradleVideoDemo	Thu at 10:16 AM	0 work items	Get started
master f2ea87e9	<1s	0 artifacts	

Errors 1

No hosted parallelism has been purchased or granted. To request a free parallelism grant, please fill out the following form https://aka.ms/azpipelines-parallelism-request
20250515.1

View documentation for troubleshooting failed runs

Jobs

Name Status Duration

Azure DevOps is **refusing to run your pipeline** because:

- Your organization **doesn't have permission** to use Microsoft-hosted agents (virtual machines that run your pipeline).
- By default, **new organizations created after certain limits** must **request access to free parallel jobs**, even for public projects.

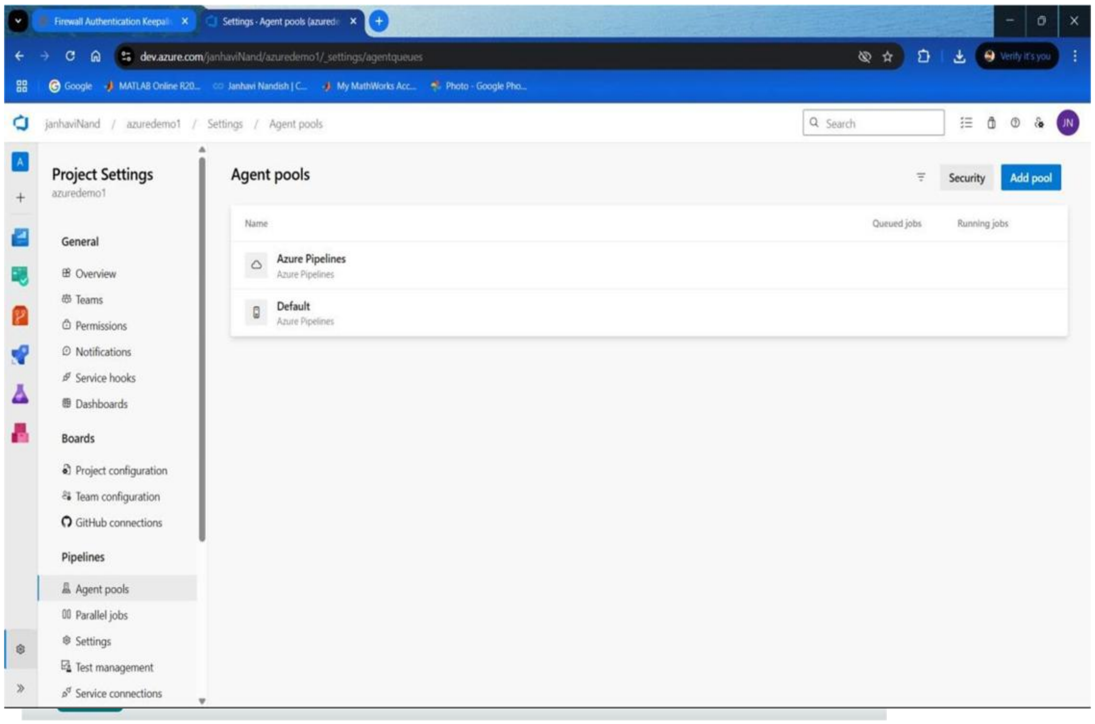
Solution:

👉 **Request Free Parallelism**

1. Go to this Microsoft form:

https://aka.ms/azpipelines-parallelism-request

2. Fill in the required details:

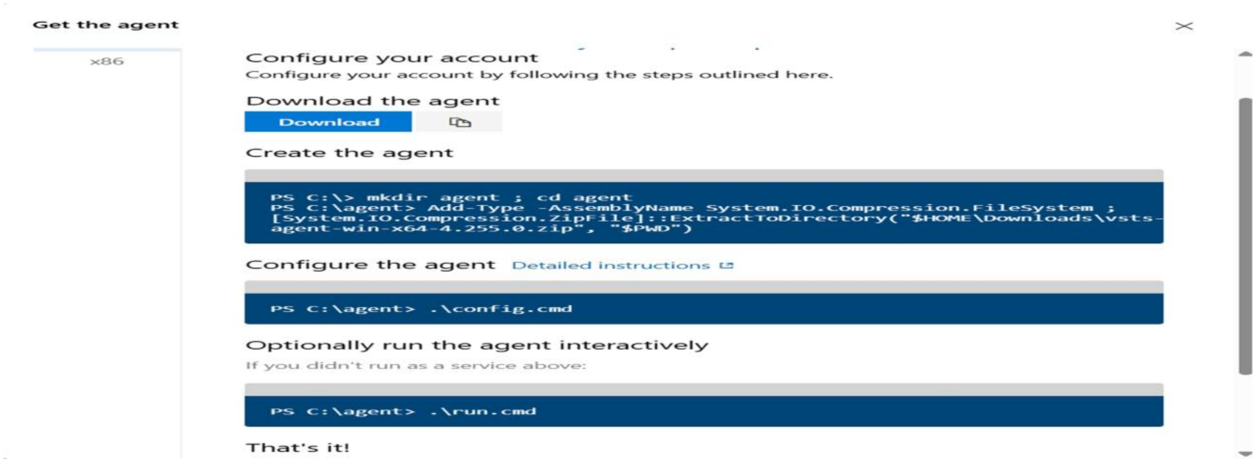


Name	Queued jobs	Running jobs
Azure Pipelines (Azure Pipelines)		
Default (Azure Pipelines)		

3. Wait for approval (usually within 2-5 business days).

Step 10:

1. Go to project settings on the left
2. Go to agent pools, under this select the default agent, and click on the new agent to create a new agent
3. Download agents and copy the downloaded zip file from downloads and paste it to C drive



Get the agent

x86

Configure your account

Configure your account by following the steps outlined here.

Download the agent

Create the agent

```
PS C:\> mkdir agent ; cd agent
PS C:\agent> Add-Type -AssemblyName System.IO.Compression.FileSystem ;
[System.IO.Compression.ZipFile]::ExtractToDirectory("$HOME\Downloads\vsts-
agent-win-x64-4.255.0.zip", "$PWD")
```

Configure the agent Detailed instructions

```
PS C:\agent> .\config.cmd
```

Optionally run the agent interactively

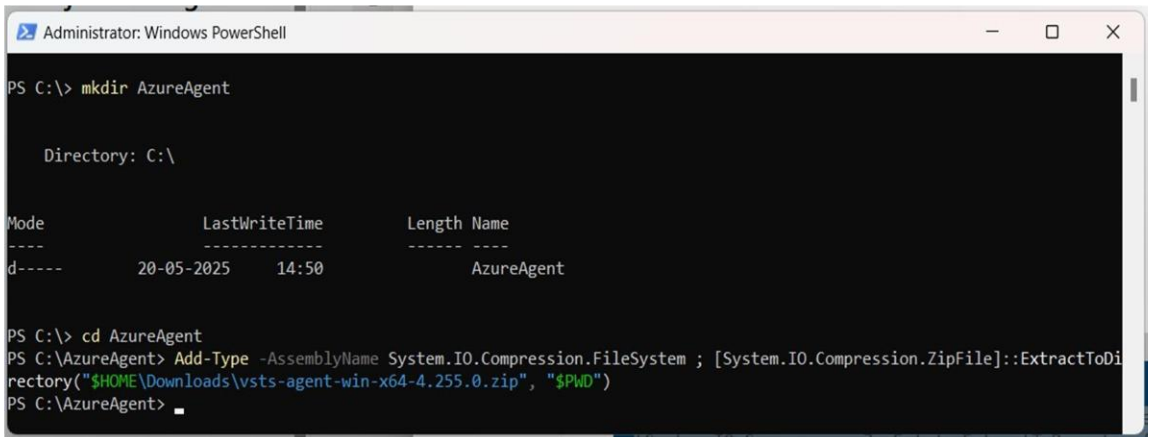
If you didn't run as a service above:

```
PS C:\agent> .\run.cmd
```

That's it!

4. Go to windows power shell to execute the agent

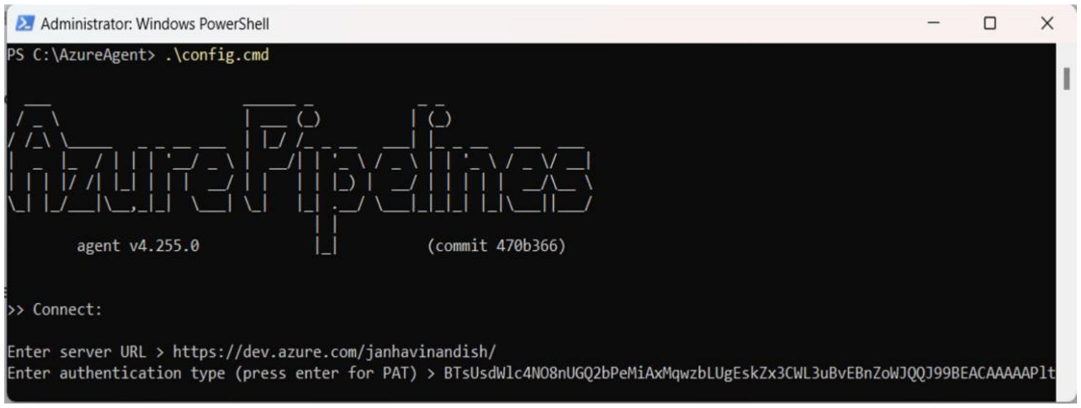
5. Give the following commands to create a directly calledagent and extract the download file.



Administrator: Windows PowerShell

```
PS C:\> mkdir AzureAgent


    Directory: C:\


Mode                 LastWriteTime         Length Name
----                 -------------         ------ ----
d-----        20-05-2025     14:50                AzureAgent


PS C:\> cd AzureAgent
PS C:\AzureAgent> Add-Type -AssemblyName System.IO.Compression.FileSystem ; [System.IO.Compression.ZipFile]::ExtractToDi
rectory("$HOME\Downloads\vsts-agent-win-x64-4.255.0.zip", "$PWD")
PS C:\AzureAgent> _
```

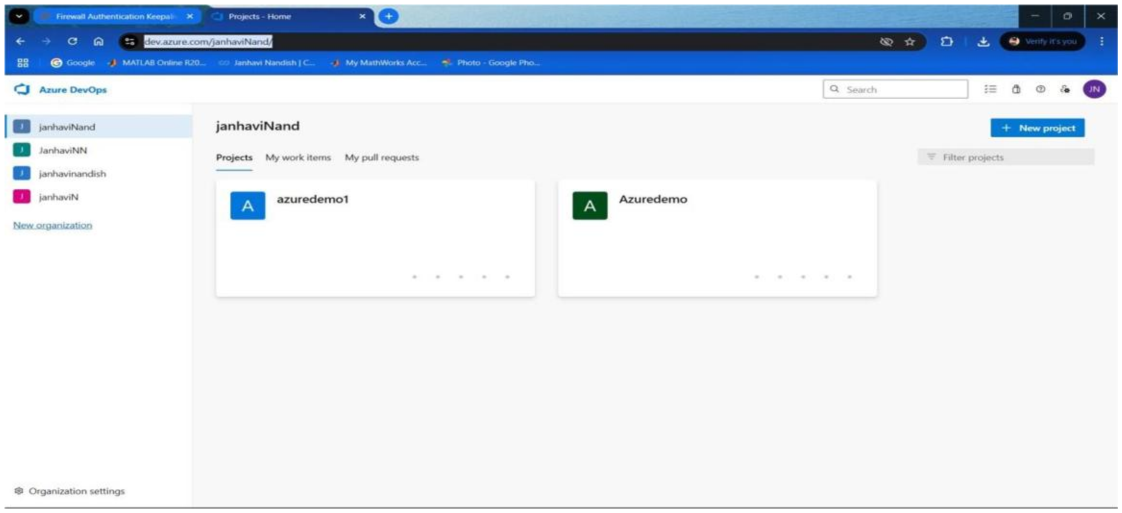
6. Configure the agent by .\config.cmd



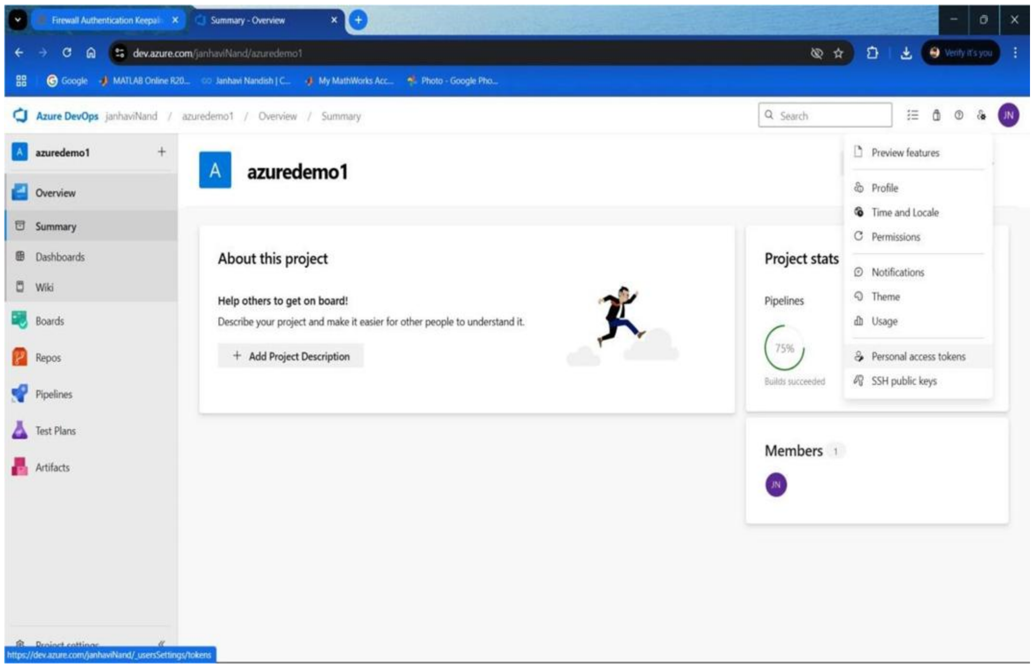
Administrator: Windows PowerShell

```
PS C:\AzureAgent> .\config.cmd

      agent v4.255.0                         (commit 470b366)


>> Connect:

Enter server URL > https://dev.azure.com/janhavinandish/
Enter authentication type (press enter for PAT) > BTsUsdWlc4NO8nUGQ2bPeMiAxMqwzbLUgEskZx3CWL3uBvEBnZoWJQQJ99BEACAAAAAPlt
```

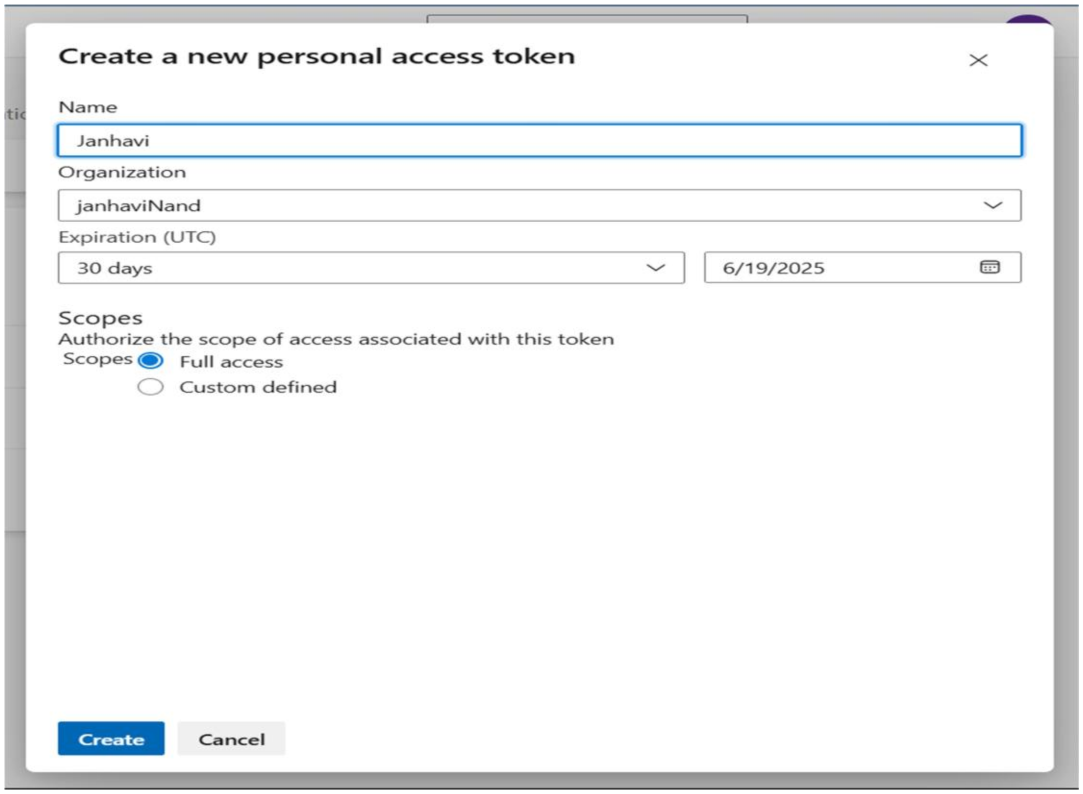
7. Give the organization URL for the server URL



8. For Personal Access Token, go to user settings, select PAT and clink on new token



9. Under PAT, provide the username and select full access, and create



Create a new personal access token

Name: Janhavi

Organization: janhaviNand

Expiration (UTC): 30 days 6/19/2025

Scopes

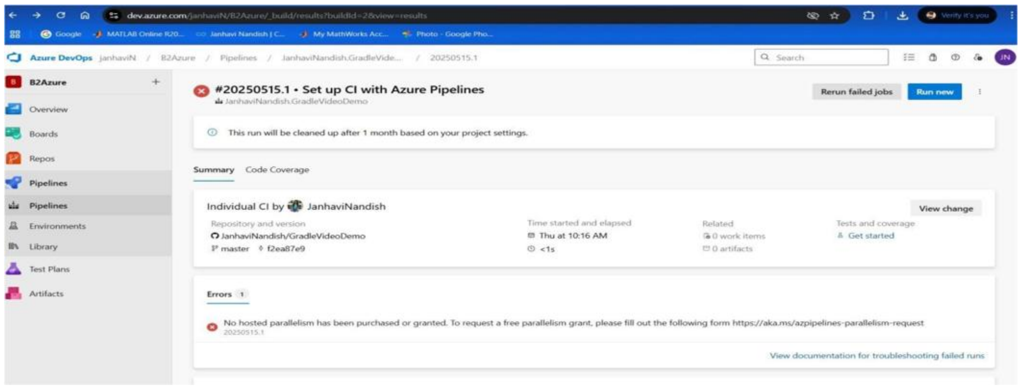
Authorize the scope of access associated with this token

Scopes [✓ Full access] Custom defined

10. Copy the PAT number and paste it on the terminal

    1. After the configuration of the agent, the agent will be in offline. To activate the agent, enter the command .\run.cmd
    2. It will connect to the server, now the agent is online and listening to the Job.

Step 11: After accepting the parallelism request, rerun the previously build job.



dev.azure.com/janhaviN/B2Azure/_build/results?buildId=2&view=results

Azure DevOps janhaviN / B2Azure / Pipelines / JanhaviNandish.GradleVide... / 20250515.1

#20250515.1 • Set up CI with Azure Pipelines

JanhaviNandish.GradleVideoDemo

This run will be cleaned up after 1 month based on your project settings.

Individual CI by JanhaviNandish

Repository and version: JanhaviNandish/GradleVideoDemo, master f2ea87e9

Time started and elapsed: Thu at 10:16 AM, <1s

Related: 0 work items, 0 artifacts

Tests and coverage: Get started

Errors 1

No hosted parallelism has been purchased or granted. To request a free parallelism grant, please fill out the following form https://aka.ms/azpipelines-parallelism-request

20250515.1