

PROGRAM - 01

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,

PROGRAM - 01

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 folders :

- After extraction, move the extracted Gradle folder (usually named gradle-x.x.x.x) to a

convenient directory like
c:\Program Files\.

(14) Navigate to the bin folder:

- Open the grade folder, then navigate to the bin folder inside.
- Copy the path from the file explorer address bar (e.g, c:\Program Files\grade - x.x\bin).

(15) 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 grade directory.
(e.g, c:\Program Files\grade - x.x\bin)

(16) Save the changes:

- Click OK to close the windows and save your changes.

Name of Experiment

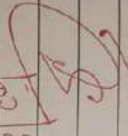
Experiment No.

Date

Page No. 03

(4) Verify the Installation :

• Open a terminal or command prompt and run : `gradle -v` If it shows the gradle version, the setup is complete.


03/02/25

PROGRAM - 02

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

Creating a Maven Project.
Working with Maven Project.

- Open command prompt
- mkdir program - this will create program folder.
- cd program - navigate program folder
- After then follow the below step to working with Maven project.

Using command line:

- To create a basic Maven project using the command line, you can use the following command:

mvn archetype:generate -DgroupId=com.lab2 -DartifactId=myapp -DarchetypeArtifactId=maven-archetype-quickstart -DinteractiveMode=false

- groupId: A unique identifier for the group (usually the domain name).
- artifactId: A unique name for the project artifact (your project).

- Open Java Code (Apptext.java) File

- Open a file -AppTest.java inside the src /test/ java /
dom / example / directory.
- After opening the -AppTest.java copy the below code
and paste it in that file then save it.

~~package~~ var. example;

import org.junit.Assert;
import org.junit.Test;

public class AppTest {

Q15

Public void testAdd() {

-App app \leq news -App();

```
int result = app.add(2, 3);
```

system.out.println("Running test : $2 + 3 =$ " + result);

Assert. assertEquals(5, result);

$$\frac{3}{3}$$

Building the Project

Compile the Project

`mvn compile`

Run the Unit Tests
`mvn test`

Package the project into a JAR
`mvn package`

Run the application (using JAR)

`java -cp target/
myapp-1.0-SNAPSHOT.jar
com.example.App`

The above command is used to run a Java application from the command line. There's a breakdown of each part:

- `java`: This is the Java runtime command used to run Java applications.
- `-cp`: This stands for classpath. and it specifies the location of the classes and resources that the JVM needs to run the application. In this case, it's pointing to the JAR file where your compiled

classes are stored.

- target/myapp-1.0-SNAPSHOT.jar: This is the JAR file (Java Archive) that contains the compiled Java classes and resources. It's located in the target directory, which Maven creates after you run mvn package.

- com.example.App: This is the main class that contains the main() method. When you run this command, Java looks for the main() method inside the App class located in the com.example package and executes it.

Q.

PROGRAM - 03

Working with Gradle: Setting up a Gradle project, understanding Build Scripts (Groovy and Kotlin DSL)

Step 1: Create your Project folder

- Open the command prompt or terminal.
- Create and Navigate to a New folder:

```
n> mkdir MyGradleProject
```

```
n> cd MyGradleProject
```

Step 2: Initialize the Gradle Project

Run the following command to initialize a Java application project:

```
n> gradle init --type java-application
```

① During initialization, Gradle will prompt you with questions such as:

- Which build script DSL to use: Groovy (creates build.gradle) or Kotlin (creates build.gradle.kts).
- The project type, test framework, etc.

Step 3: Explore the generated Project structure.
After running the init command, your project folder

should contain files similar to this :

```
MyGradleProject /
  build.gradle // (if you selected groovy DSL) or
  build.gradle.kts (if Kotlin DSL)
  settings.gradle // Project settings file.
  gradle / // Gradle wrapper files (if included)
  gradlew // Unix Gradle wrapper script.
  gradlew.bat // Windows.
  gradle wrapper script.
  src /
  main /
  java /
  App.java // your main
  Java file
  test /
  java /
  AppTest.java // A sample test file.
```

~~Example for groovy DSL (build.gradle)~~

Open the file build.gradle (located in the project root) in your favourite text editor. It might look like this :

plugins {
id 'application'

```
id 'java'
}
application {
    mainClassName = 'App'
}
repositories {
    mavenCentral()
}
dependencies {
    implementation 'org.apache.commons:commons-lang3:3.12.0'
    testImplementation 'junit:junit:4.13.2'
}
// A custom task example:
tasks.register('greet') {
    doLast {
        println 'Hello, this is my custom gradle task!'
    }
}
```

Example for Kotlin DSL
(build.gradle.kts)

If you chose Kotlin DSL, open build.gradle.kts. It might look like this

plugins {
 application

```
java.  
{  
  application {  
    mainClass.set("App")  
  }  
  repositories {  
    mavenCentral()  
  }  
  dependencies {  
    implementation ("org.apache.commons:commons-lang  
3:3.12.0")  
  }  
  testImplementation ("junit:junit:4.13.2")  
}  
// A custom task example:  
tasks.register("greet") {  
  doLast {  
    println("Hello, this is my custom gradle task!")  
  }  
}
```

Step 4:

- Open the appropriate build script file.
- Add the custom task code (as shown above) at the end of the file.
- Save the file.

Step 5: Build and Run your Project

1. Build the Project:

In your command prompt, ensure you're in your project directory and run:

```
~> gradle build.
```

This compiles your source code, runs tests, and packages your application.

Run the Application:

To run your Java application (using the main class defined in your build script), execute:

```
~> gradle run
```

~~Create the custom task:~~

If you added the custom task (named greet), run:

```
~> gradle greet
```

This will print the custom greeting message to the console.

Building and running the project ensures that your code is compiled correctly and that the configurations in your build script are applied.

Custom tasks help automate additional actions during your build process.

Summary.

1. Create and Navigate to your project folder.
2. Initialize your project:
gradle init --type java-application.
3. Review the generated Project Structure:
 - build.gradle / build.gradle.kts (in the project root)
 - src folder for your java code.
4. Understand and Edit the Build Script:
 - Modify plugins, repositories, dependencies, and tasks.
 - Add a custom task (example provided).
5. Build and Run:
 - Run gradle build to compile and test.
 - Run gradle run to execute the application.
 - Run gradle greet to see your custom task in action.

14/3/20

PROGRAM - 04

Build and Run a Java application with Maven,
Migrate the same application to Gradle.

Step 1: Creating a Maven Project

You can create a Maven project using the mvn command (or through your IDE, as mentioned earlier). But here, I'll give you the essential pom.xml and Java code.

I'm using command Line :

To create a basic Maven project using the command line, you can use the following command :

```
mvn archetype:generate -DgroupId=com.example -DartifactId=maven-example -DarchetypeArtifactId=maven-archetype-quickstart -DinteractiveMode=false
```

Step 2: Open Java Code (App.java) File.

- Open a file App.java inside the src/main/java/com/example/ directory.
- After opening the App.java copy the below code and paste it in that file then save it.


```
package com.example;
```

```
public class App {
```

```
    public static void main(String[] args) {
```

```
        System.out.println("Hello, Maven");
```

```
        System.out.println("This is the simple realworld  
example....");
```

```
        int a = 5;
```

```
        int b = 10;
```

```
        System.out.println("sum of " + a + " and " + b +  
                           " is " + sum(a, b));
```

```
    }
```

```
    public static int sum(int x, int y) {
```

```
        return x + y;
```

```
    }
```

```
}
```

Note: Before building the project make sure you are in the project folder if not navigate the project folder type command in your command prompt `cd maven-example`.

Step 3: Run the Project.

To build and run this project, follow these steps:

- Open the terminal in the project directory and run the following command to build the project.

mvn clean install

- Run the program with below command :

mvn exec:java -Dexec.mainClass = "com.example.App"

Step : Migrate the Maven Project to Gradle.

Initialize Gradle : Navigate to the project directory (gradle-example) and run :

gradle init

- It will ask found a Maven build. Generate a Gradle build from this ? (default: yes) [yes, no]
- Type yes.
- Select build script DSL :
 1. : Kotlin
 2. : GroovyEnter selection (default: Kotlin) [1, 2]
 - Type 2.
- Generate build using new APIs and behavior (some features may change in the next minor release) ? (default: no) [yes, no]
 - Type No.

Navigate the project folder and open build.gradle file then add the below code and save it.

```
plugins {  
    id 'java'  
}  
  
group = 'com.example'  
version = '1.0 - SNAPSHOT'  
  
repositories {  
    mavenCentral()  
}  
  
dependencies {  
    testImplementation 'junit:junit:4.12'  
}  
  
task run(type: JavaExec) {  
    main = 'com.example.App'  
    classpath = sourceSets.main.runtimeClasspath  
}
```

Step 4: Run the Gradle Project

- Build the project: In the project directory (gradle-example), run the below command to build the project:

Name of Experiment

Date:

Experiment No.

Page No. 22

gradlew build.

- Run the Application: Once the build is successful, run the application using below command:

gradlew run.

18/05/2024
10/10

PROGRAM - 05

Introduction to Jenkins, installation.

Introduction to Jenkins.

What is Jenkins?

Jenkins is an open-source automation server widely used in the field of continuous integration (CI) and continuous Delivery (CD).

- * It allows developers to automate the building, testing and deployment of software project, making the development process more efficient and reliable.

Key features of Jenkins:

- * CI/CD: Jenkins supports continuous Integration and Continuous Deployment, allowing developers to integrate code changes frequently and automate the deployment of application.
- * Plugins: Jenkins has a vast library of plugins that can extend its capabilities. These plugins integrate Jenkins with version control system (like git), build tools (like Maven or Gradle), testing frameworks, deployment tools and much more.

* Pipeline as code : Jenkins allows the creation of pipelines using groovy-based DSL scripts or YAML files, enabling version-controlled and repeatable pipelines.

* Cross-platform : Jenkins can run on various platforms such as windows, Linux, macOS, and others.

Installing Jenkins.

Jenkins can be installed on local machines on a cloud environment or even in containers. Here's how you can install Jenkins in window local system environments:

1) Installing Jenkins Locally.

Step-by-step guide (window)

i) Prerequisites

* Ensure that Java (JDK) is installed on your system. Jenkins requires Java 8.

* You can check if java is installed by running java-version in the terminal.

2) Install Jenkins on Window system.

- * Download the Jenkins windows installer from the official Jenkins website.
- * Run the installer and follow the on-screen instructions. while installing choose login system: run service as local system.
- * After then use default port or you can configure you own port like I'm using port 8080 then click on test and next.
- * After then change the directory and choose java jdk-21 path look like C:\Program Files\Java\jdk-21.
- * After then click next, next and then it will ask permission click on yes and it will start installing.
- * After successfully installed, Jenkins will be running on port either default port or chosen port like choose port 8080 by default (you can access it in your browser at <http://localhost:8080>)

3) Jenkins setup in browser :

- * After opening browser by visiting your local address the browser should look like below screenshot.

Name of Experiment

Date:.....

Experiment No.....

Page No.....26.....

- * It will ask administrator password so you have to navigate the above highlighted path and open that initial admin password in notepad or any software to see the password.
- * Just copy that password and paste it and click on continue.
- * It will ask to customize Jenkins so click on install suggested plugin it will automatically install all required plugin.
- * After then create admin profile by filling all details then click on save and continue after then save and finish after then click on start using Jenkin.

13/05/2024

10/10

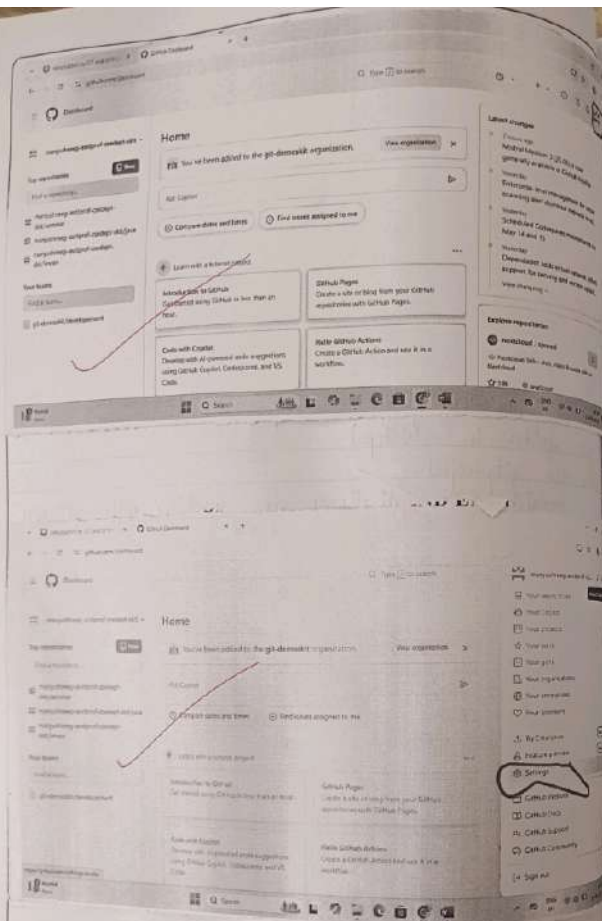
PROGRAM - 06

Commands to push an file from local repo to remote repo using git and github.

- Pwd
present working directory - to check where we are now.
- cd desktop
change directory to desktop (we need to create an git folder in desktop)
- mk dir gitproject
creating an directory for working with git you can give any name of your choice.
- cd gitproject
coming inside that git folder
- git init
initializing git software { we need to execute git commands inside this folder only }

Now we are using vi editor to write source code.

- vi files.sh
press i to insert after writing press esc and :wq (to save and quit)
- ls
to check the list of files



Name of Experiment _____ Date: _____
 Experiment No. _____ Page No. 28

• git add
 adding to staging area (*..will add all the files to staging area).

• git commit -m "commit message"
 adding to local repo

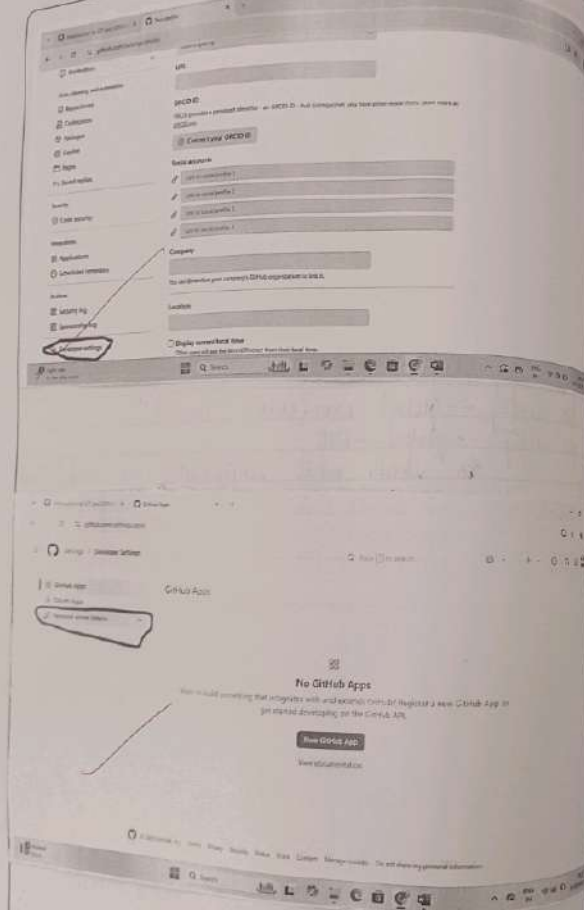
To do configuration we need to give our user name and email id of our git hub account.

- git config --global user.name "username"
- git config --global user.email "emailid"
- git config --global -list
 to check what configuration we have added.
- git remote add alias name url
 to add repository url
- git remote -v
 to check which repo is mapped.

To push the file to remote repo.. master is the default branch which will be created when we create an repository.

- git push alias name master.
 alias name is the name which is short form or nick name which we had given while adding remote repo. Once we execute git push cmd.

SKIT B'lore



Name of Experiment _____ Date: _____
 Experiment No. _____ Page No. 22

when we are executing for first time we will get and small window poped up asking pat (personal access token) or you can give your email and password.

• pat

Steps to generate PAT

In your git hub account

go to settings... developer settings... personal access tokens... classic... generate... give the duration of time which it will be active.

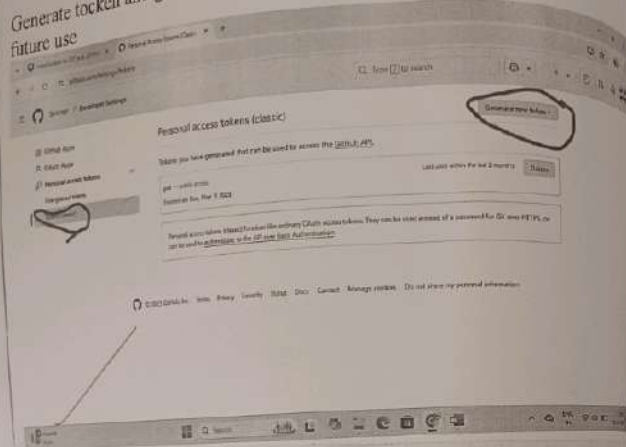
1. click this icon you will get settings.

go to settings

Select developer settings.

SKIT B'lore

Generate token and give life span for that token and save in notepad for future use



Name of Experiment

Date

Experiment No.

Page No. 30

Select personal access token... Then classic... then generate token.

generate token and give life span for that token and save in notepad for future use.

generate token and give life span for that token and save in notepad for future use.

10/10

SKIT B'lore

PROGRAM - 07

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

ansible - playbooks / apache - install.yml

Steps:

1. sudo apt update.
2. sudo apt upgrade -y
3. sudo apt install ansible -y
4. ansible --version.

Steps: Create an Inventory File.

1. Open your text editor to create a file called hosts.ini:
2. Add localhost:
3. Add the following content to define the local host:
[local]
localhost ansible_connection=local.
- Explanation:
[local] is a group name.

- localhost is the target host.
- ansible-connection = local tells Ansible to execute commands on the local machine without SSH.

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

Step 2: Create the Playbook file.

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

-name: Basic Server Setup

hosts: local

become: true # Optional: if your tasks need root access.

tasks:

-name: Example task

debug:

msg: "Hello, this is a basic setup"

Step 3: Execute the Playbook

In your terminal, run the following command:

`ansible-playbook -i hosts.ini setup.yml`

`ansible-playbook -i hosts.ini setup.yml --ask-become-pass`

PROGRAM - 09.

Introduction to Azure Devops: Overview of Azure Devops services, setting up an azure Devops account and Project.

Overview of Azure Devops.

Azure Devops is a comprehensive suite of cloud based services designed to support the entire software development lifecycle. It provides tools for planning, developing, testing, delivering and monitoring applications. Here are the primary services offered:

• Azure Repos :

A set of version control tools that allow you to host git repositories or use Team foundation Version control (TFVC). It offers collaboration features such as pull requests, branch policies, and code reviews.

• Azure Pipelines :

A CI/CD Service that helps automate builds, tests and deployments. It supports multiple languages, platforms and can run on Linux, Windows or macOS agents.

• Azure Boards :

A work tracking system that helps teams manage

work items, sprints, backlogs and Kanban boards. It facilitates agile planning and reporting.

• Azure Test Plans:

Provides a solution for managing and executing tests, capturing data about defects and tracking quality.

• Azure Artifacts:

Allows you to create, host and share packages (such as Maven, npm, NuGet, and Python packages) with your team, integrating package management into your CI/CD pipelines.

These services integrate with each other and with popular third-party tools to create a cohesive DevOps ecosystem.

Step 1: Sign up for an Azure DevOps account

1. Open your Web Browser:

• Navigate to the Azure DevOps website: <https://dev.azure.com>

2. Sign In or Create a Microsoft Account:

• If you already have a Microsoft account (such as Outlook, Hotmail, or office 365), click "Sign in".

• If you do not have a Microsoft account, click "Create one!" and follow the instructions to create a new Microsoft account.

Step 2: Create an Azure DevOps organization.

1. Click on azure devops organization or search it on search bar.
- Click on my azure devops organizations.
- Click on create a New Organization and click on continue.
- Enter a unique name for your organization (e.g. Your Company or your name DevOps or MyPersonalOrg).
- Select a Region: India. (it is automatically shows india).
- Enter the characters you see.
- Click "Continue" or "Create".

Step 3: Creating an Azure DevOps Project.

1. Configure your project:
 - Project Name: Enter a descriptive name for your project (e.g. HelloDevOps).
 - Description: Optionally, provide a brief description (e.g. "A sample project to demonstrate Azure DevOps services").
 - Visibility:
 - Choose "Private" if you want to restrict access to your project.
 - Choose "Public" if you are okay with the project being accessible to anyone.
 - Advanced Options (Optional): You can choose a version control system (git is the default) and a work

item process (Agile, Scrum, or Basic). For most beginners, the defaults are recommended. Click "Create".

Step 4 : Explore your project Dashboard.

1. Project Overview:

Once your project is created, you will be directed to the project dashboard. Here you will see navigation options for:

- Repos : Where your code is stored.
- Pipelines : For build and release automation.
- Boards : For work tracking and agile planning.
- Test plans : For managing and running tests.
- Artifacts : For hosting packages.

2. Familiarize yourself with the Interface:

Click through each section (eg. Repos, Pipelines, Boards) to get a sense of the available features.

(Signature)
26/5/25