# Experiment 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

**1. Introduction to Ansible 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.
* **Orchestration:** Coordinate complex IT workflows and processes.

## 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: o **Hosts:** The target machines (or groups) on which the tasks should run. o **Tasks:** A list of actions (using modules) that should be executed.

o **Modules:** Reusable, standalone scripts that perform specific actions (e.g., installing packages, copying files, configuring services).

* **Modules:**

Ansible comes with a large collection of built-in modules (such as apt, yum, copy, service, etc.). These modules perform specific tasks on target hosts. You can also write custom modules if needed.

**Why Use Ansible?**

* **Agentless:** Ansible uses SSH to communicate with target hosts, so no agent needs to be installed on them.
* **Simplicity:** Playbooks use simple YAML syntax, making them easy to write and understand.
* **Idempotence:** Ansible tasks are idempotent, meaning running the same playbook multiple times yields the same result, ensuring consistency.
* **Scalability:** Ansible can manage a small number of servers to large infrastructures with hundreds or thousands of nodes.

# 2. Installing Ansible on Ubuntu

Before writing a playbook, you need to install Ansible on your control machine (your local Ubuntu system).

**Step 1: Update Your System** Open your terminal and run:

sudo apt update sudo apt upgrade -y **Step 2: Install Ansible** Install Ansible using apt:

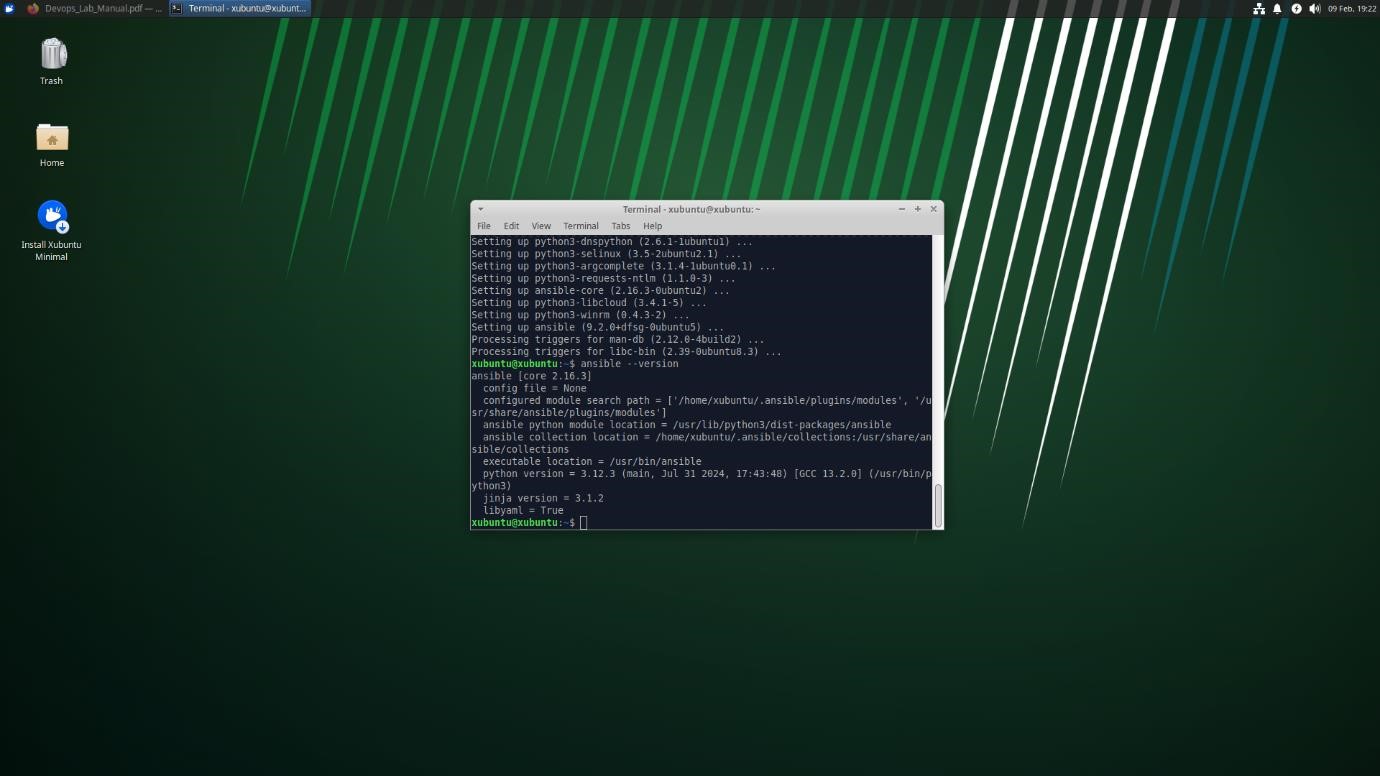
sudo apt install ansible -y

Verify the installation by checking the version:

ansible --version *Expected Output Example:*

ansible 2.9.x

config file = /etc/ansible/ansible.cfg ...

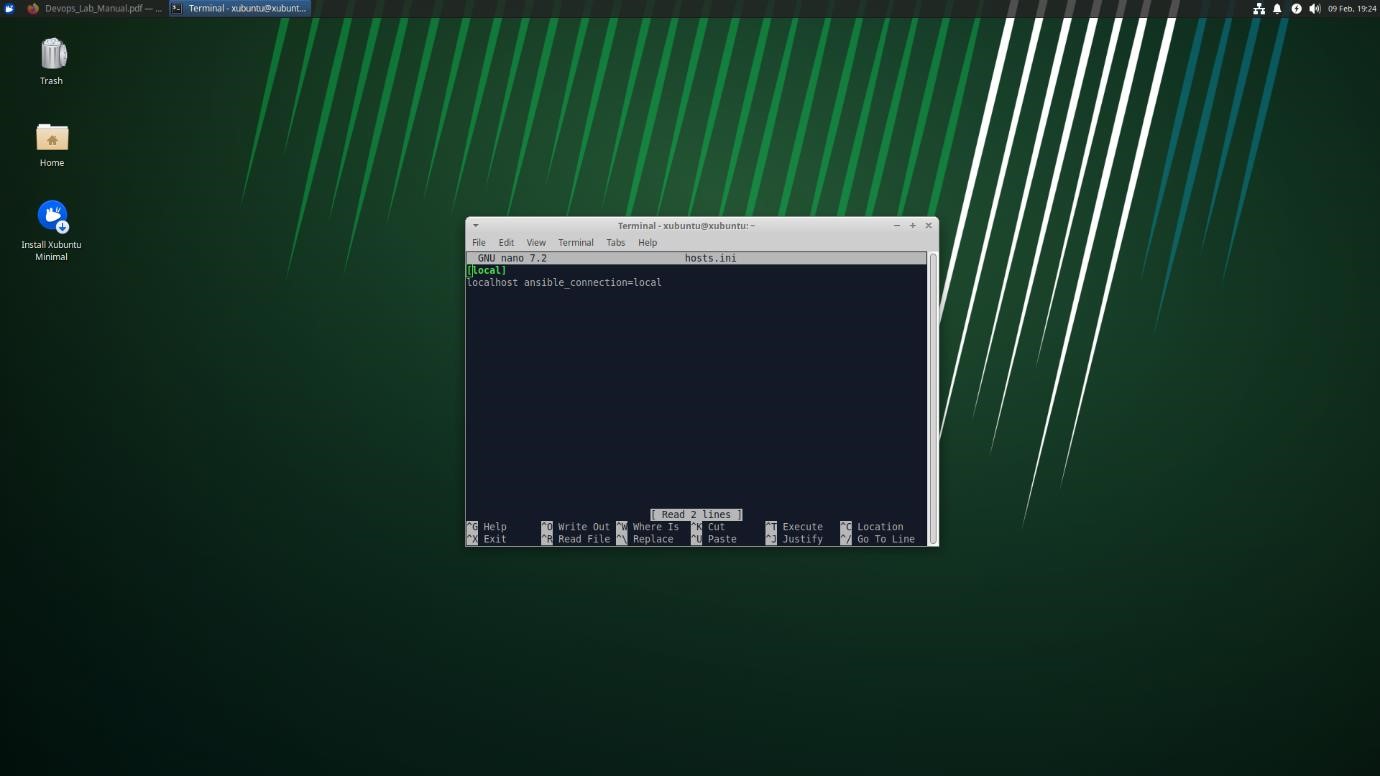


# 3. Creating an Ansible Inventory

An inventory file lists the hosts you want to manage. For this experiment, you can use the local host.

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



## 3. Automated Server Configurations

While our experiment covered the basics, here’s how you can extend it:

* **Configuring Services:**

Use modules like service to start, stop, or restart services. For example, you can automate the configuration of web servers (e.g., Apache or Nginx).

* **Managing Files and Templates:**

Use the copy or template modules to deploy configuration files across your servers.

This is useful for maintaining consistent configuration settings.

* **User and Group Management:**

The user and group modules allow you to create or modify user accounts, ensuring that the correct permissions and roles are applied automatically.

* **Advanced Orchestration:**

Ansible playbooks can include conditionals, loops, and error handling to manage more complex setups, ensuring idempotence and consistency across your infrastructure.

## Example: Automating a Web Server Configuration

If you wanted to automate the configuration of an Nginx server, your playbook might include tasks such as:

* name: Configure Nginx Web Server hosts: webservers become: yes tasks:
* name: Update apt cache apt:

update\_cache: yes

* name: Install Nginx apt:

name: nginx state: present

* name: Copy Nginx configuration file template:

src: nginx.conf.j2 dest: /etc/nginx/nginx.conf notify:

* Restart Nginx

handlers:

* name: Restart Nginx service: name: nginx state: restarted

# 5. Writing a Basic Ansible Playbook

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

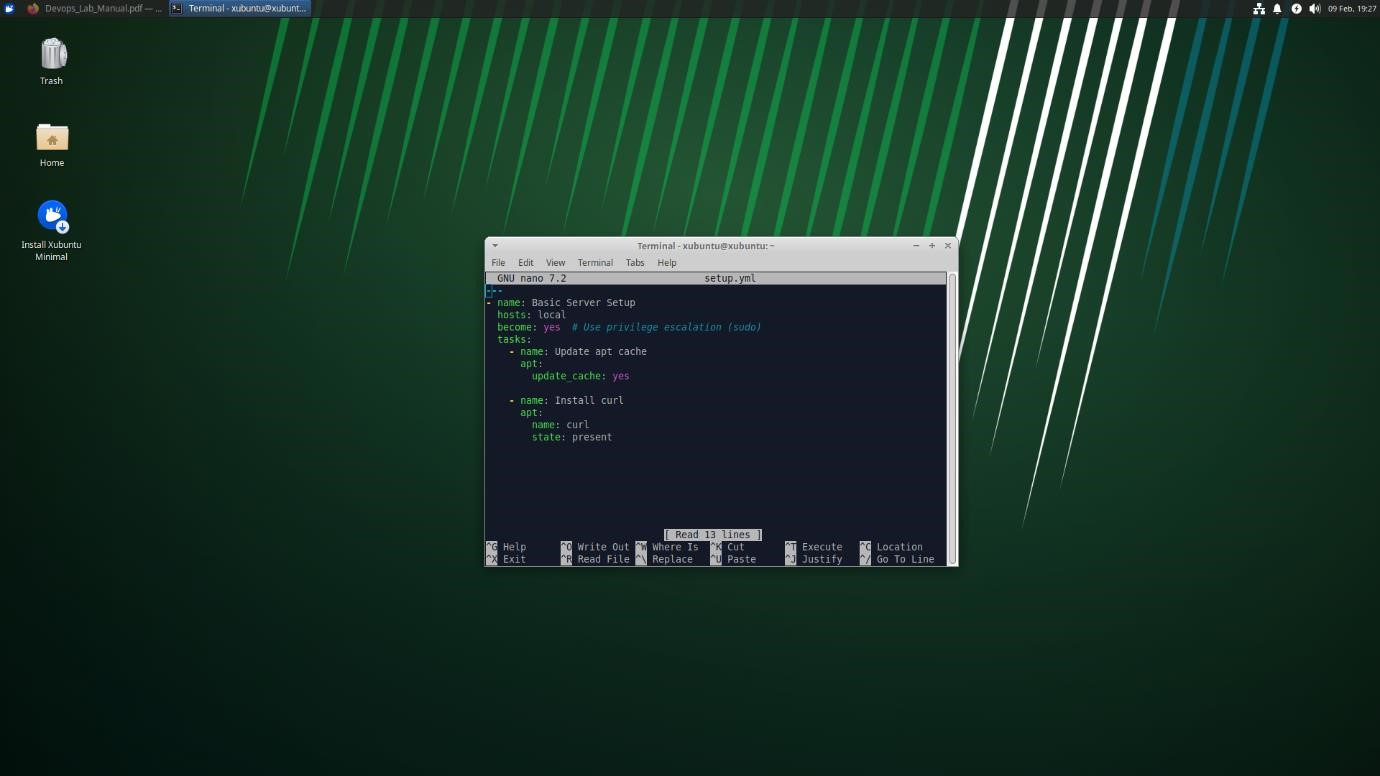
* **Updating the apt cache**
* **Installing a package (e.g., curl)**

## Step 1: Create the Playbook File

1. Open your text editor to create a file called setup.yml:
2. nano setup.yml
3. Add the following YAML content:
4. ---
5. - name: Basic Server Setup
6. hosts: local
7. become: yes # Use privilege escalation (sudo)
8. tasks:
9. - name: Update apt cache
10. apt:
11. update\_cache: yes

12.

1. - name: Install curl
2. apt:
3. name: curl
4. state: present o **Explanation:**
   * name: Provides a descriptive name for the play.
   * hosts: Specifies the group or hosts from the inventory file.
   * become: yes: Uses sudo to perform tasks that require elevated privileges.
   * **Tasks Section:**
   * **Update apt cache:** Uses the apt module to update the package cache.
   * **Install curl:** Uses the apt module to install the curl package if it isn’t already installed.
5. Save and exit the file.



# 6. Running the Ansible Playbook

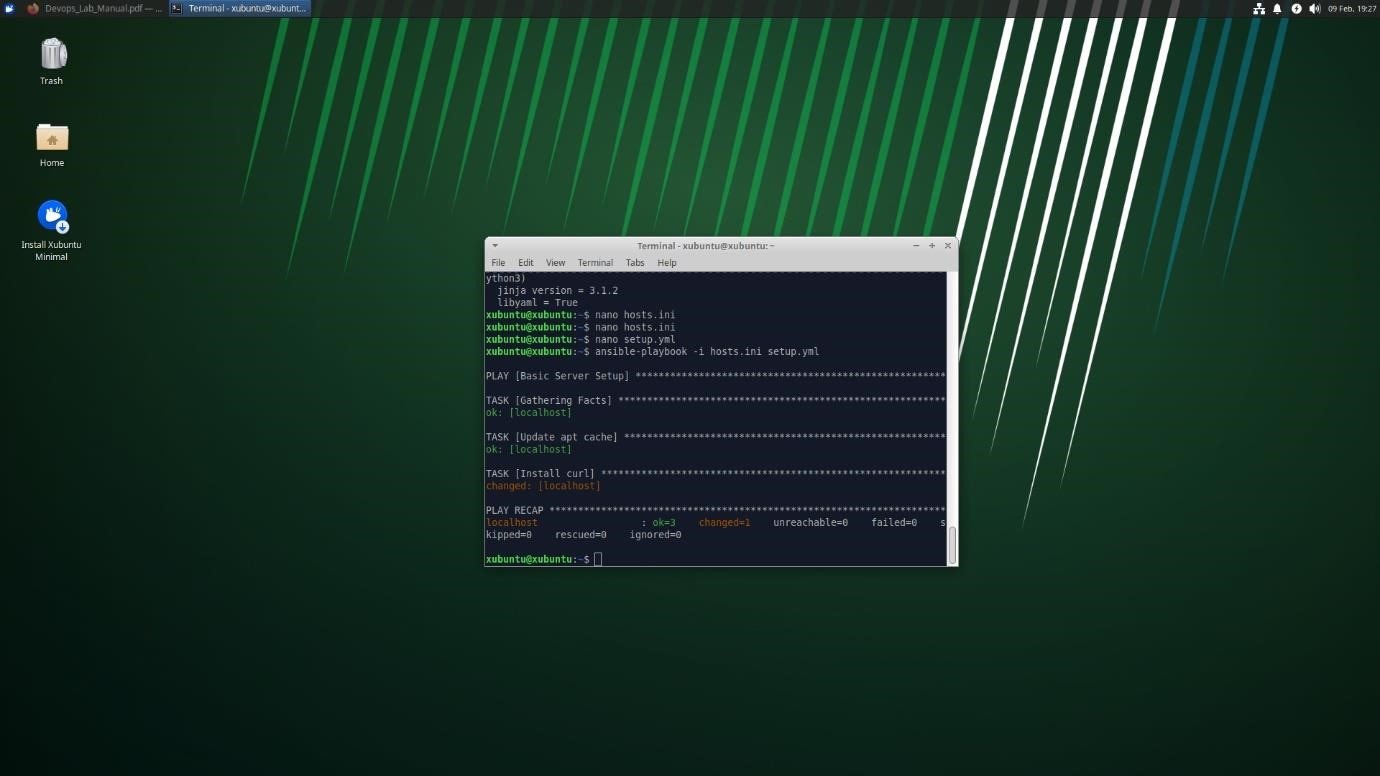
## Step 1: Execute the Playbook

In your terminal, run the following command: ansible-playbook -i hosts.ini setup.yml

* **Explanation:**
  + ansible-playbook: The command to run an Ansible playbook.
  + -i hosts.ini: Specifies the inventory file.
  + setup.yml: The playbook file you just created.

*Expected Output:*

* An output that details each task:
  + For example, a task summary might show “ok=2 changed=1” (if the apt cache was updated and curl was installed).



**Experiment 8: Practical Exercise: Set Up a Jenkins CI Pipeline for a Maven Project, Use Ansible to Deploy Artifacts Generated by Jenkins**

# 1. Overview

In this experiment, you will:

* Set up a Jenkins job to automatically build a Maven project from source control.
* Archive the build artifact (a JAR file) produced by Maven.
* Integrate an Ansible deployment step within Jenkins (using a post-build action) to deploy the artifact to a target location.
* Verify that the artifact is deployed successfully.

This exercise demonstrates how Continuous Integration (CI) and automated configuration management can work together to streamline the build-and-deploy process.

# 2. Prerequisites

Before you begin, ensure that:

* **Jenkins** is installed, running, and accessible (locally or on the cloud).
* **Maven Project:** You have a Maven project available in a Git repository (or stored locally). For this example, we will assume you are using the “HelloMaven” project generated in Experiment 2/4.
* **Git Repository:** The Maven project is committed to a Git repository (e.g., on GitHub) so Jenkins can pull the latest code.
* **Ansible Installed:** Ansible is installed on your control machine (or the Jenkins server) and you have created a basic inventory file (e.g., hosts.ini).

*Tip:* Verify installations and repository access before starting this exercise.

# 3. Step 1: Preparing the Maven Project

1. **Ensure Your Maven Project Is in Version Control:**

If your “HelloMaven” project is not already in a Git repository, navigate to its root and initialize Git:

1. cd path/to/HelloMaven
2. git init
3. git add .
4. git commit -m "Initial commit of HelloMaven project"

Then push it to your repository (GitHub, GitLab, etc.).

**Verify the Project Structure:**

Your project should have a standard Maven layout:

1. HelloMaven/
2. ├── pom.xml
3. └── src
4. ├── main/java/com/example/App.java
5. └── test/java/com/example/AppTest.java

# 4. Step 2: Configuring Jenkins to Build the Maven Project

## A. Create a New Jenkins Job (Freestyle Project)

1. **Log into Jenkins:**

Open your browser and navigate to your Jenkins URL (e.g.,

http://localhost:8080).

1. **Create a New Job:**

o Click **“New Item”** on the Jenkins dashboard. o **Enter an Item Name:** e.g., HelloMaven-CI. o **Select “Freestyle project”** and click **“OK”**.

## B. Configure Source Code Management (SCM)

1. **Scroll to the “Source Code Management” Section:**

* Select **“Git”**.
* **Repository URL:** Enter your repository URL (e.g.,

https://github.com/yourusername/HelloMaven.git).

* **Credentials:** If the repository is private, click **“Add”** and provide the necessary credentials.
* **Branch Specifier:** (e.g., \*/main or \*/master).

## C. Add a Maven Build Step

1. **Scroll Down to the “Build” Section:**
   * Click **“Add build step”** and select **“Invoke top-level Maven targets”**.
2. **Configure the Maven Build:**
   * **Goals:** Type: o clean package

This command cleans any previous builds, compiles the code, runs tests, and packages the application into a JAR file.

* + **POM File:** (Leave it as default if your pom.xml is in the root directory).

# 5. Step 3: Archiving the Artifact

After the Maven build completes, you need to archive the generated artifact so that it can be used later by the deployment process.

1. **Scroll Down to the “Post-build Actions” Section:**
   * Click **“Add post-build action”** and select **“Archive the artifacts”**.
2. **Configure Artifact Archiving:**
   * **Files to Archive:** Type: o target/\*.jar

This pattern tells Jenkins to archive any JAR file found in the target directory. **6. Step 4: Integrating Ansible Deployment in Jenkins**

Now, integrate an Ansible deployment step into the Jenkins job. You can do this as a post-build action that executes a shell command.

1. **Add Another Post-build Action:** o Click **“Add post-build action”** and select **“Execute shell”**.
2. **Configure the Shell Command:**
   * In the command box, add a command to trigger your Ansible playbook. For example:
   * ansible-playbook -i /path/to/hosts.ini /path/to/deploy.yml

**Note:**

* + - Replace /path/to/hosts.ini with the full path to your Ansible inventory file.
    - Replace /path/to/deploy.yml with the full path to your Ansible deployment playbook.
  + This command will run after a successful build, deploying the artifact using Ansible.

1. **Save the Jenkins Job:**
   * Click **“Save”** at the bottom of the configuration page.

# 7. Step 5: Writing an Ansible Playbook for Deployment

Create an Ansible playbook that deploys the Maven artifact (the JAR file) generated by Jenkins to a target directory.

## A. Create an Inventory File

If you haven’t already, create an inventory file (e.g., hosts.ini) that targets the deployment machine. For a local deployment, use:

[local]

localhost ansible\_connection=local **B. Create the Deployment Playbook**

1. **Open Your Text Editor** and create a file called deploy.yml:
2. nano deploy.yml
3. **Enter the Following YAML Content:**
4. ---
5. - name: Deploy Maven Artifact
6. hosts: local
7. become: yes
8. tasks:
9. - name: Copy the artifact to the deployment directory
10. copy:
11. src: "/var/lib/jenkins/workspace/HelloMaven-

CI/target/HelloMaven-1.0-SNAPSHOT.jar"

1. dest: "/opt/deployment/HelloMaven.jar" **Explanation:**
   * **hosts:** local means the playbook runs on the local machine. Adjust this if deploying to a remote server.
   * **become: yes:** Uses sudo privileges to write to system directories.
   * **src:** The path should point to the archived artifact in the Jenkins workspace.

(Adjust the path if your Jenkins workspace is different.) o **dest:** The target directory where you want the artifact deployed (ensure this directory exists or modify accordingly).

1. **Save and Exit the File**.

# 8. Step 6: Testing the Complete Pipeline

1. **Trigger a Build in Jenkins:**
   * Navigate to your Jenkins job (HelloMaven-CI) and click **“Build Now”**.
   * Monitor the build history. Once the build completes, click the build number (e.g., #1) and check the **Console Output**.
   * Look for messages indicating that:
     + The Maven build ran successfully.
     + The artifact was archived.
     + The shell command executed the Ansible playbook.

*Screenshot Tip:* Capture the console output showing the full pipeline execution, including the deployment step.

1. **Verify Deployment:**
   * Log into your target machine (or check locally) and verify that the artifact has been copied to the destination directory (e.g.,

/opt/deployment/HelloMaven.jar). o For example, run: o ls -l /opt/deployment/ o The output should list the deployed JAR file.

**Experiment 9: 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:**
  1. 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:**
  1. 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:**
  1. 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.

**Setting Up an Azure DevOps Account**

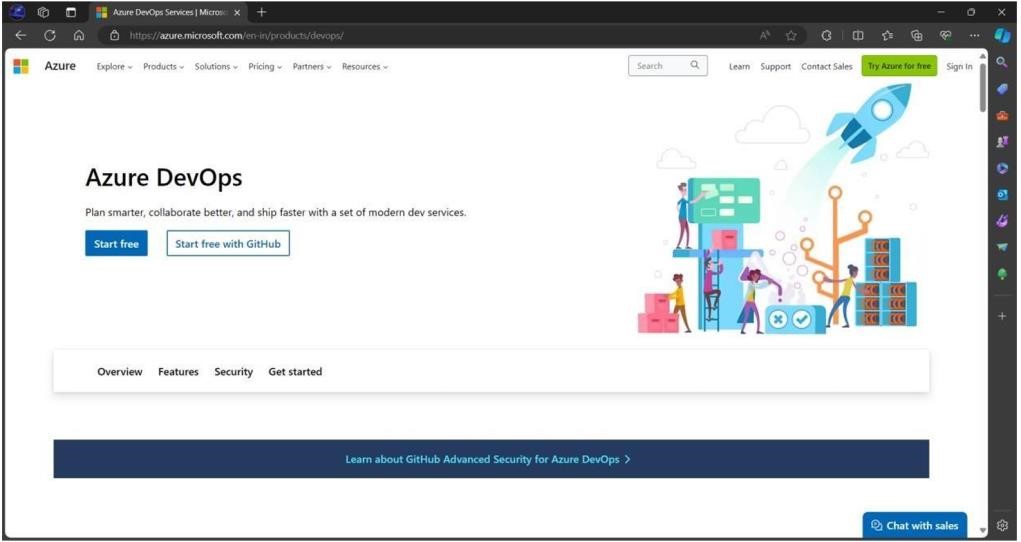
Before you can start using Azure DevOps services, you need to set up an account and create an organization. Follow these steps:

## Step 1: Sign Up for an Azure DevOps Account

1. **Open Your Web Browser:**
   * Navigate to the Azure DevOps website: [https://dev.azure.com.](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.



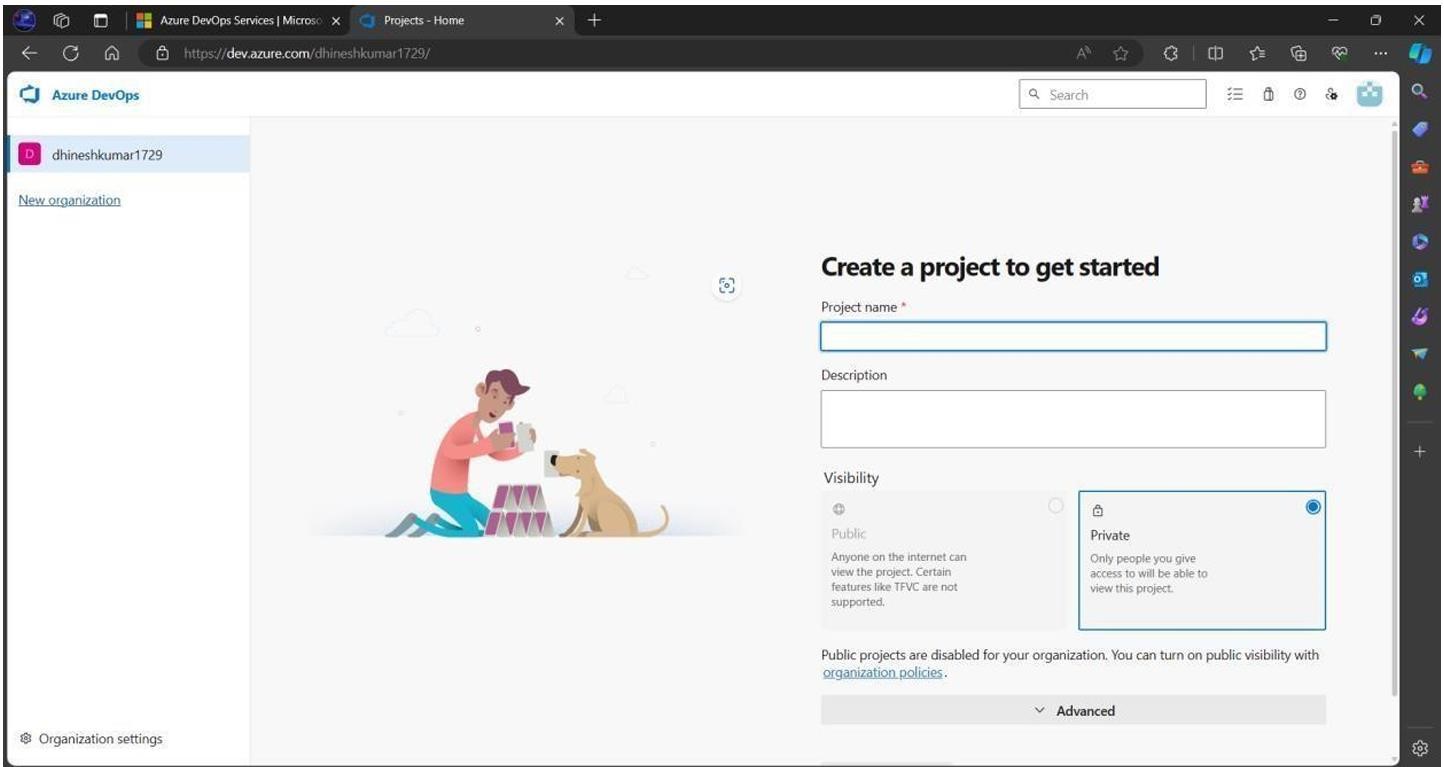
1. **Accept the Terms and Conditions:**
   * Review and accept the terms if prompted.

## Step 2: Create an Azure DevOps Organization

1. **Create a New Organization:**
   * Once signed in, you will be prompted to create an Azure DevOps organization.
   * Enter a unique name for your organization (e.g., YourCompanyDevOps or

MyPersonalOrg).

* + **Select a Region:** Choose the geographic region where your data will be stored (select the one closest to you for optimal performance). o Click **“Continue”** or **“Create”**.



1. **Review Your Organization’s Dashboard:**
   * Once created, you will see an overview dashboard for your organization. This dashboard provides navigation links to Repos, Pipelines, Boards, Test Plans, and Artifacts.

# 3. Creating an Azure DevOps Project

After setting up your organization, the next step is to create a project. A project in Azure DevOps is a container for all your source code, pipelines, boards, and other resources.

## Step 1: Create a New Project

1. **Navigate to “New Project”:**
   * On your organization’s dashboard, click the **“New Project”** button.
2. **Configure Your Project:**
   * **Project Name:** Enter a descriptive name for your project (e.g., HelloDevOps). o **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. o Click **“Create”**.

## Step 2: 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 (e.g., Repos, Pipelines, Boards) to get a sense of the available features.

