INDEX

week	Name of the Experiment	page	Date of Experiment	Date of Submission	Faculty Sign
1.	Start DevOps with a workflow that includes four phases: to do, in progress, code review, and done.		08/01/25	29/1/25	29/1
2.	Setup Eclipse for DevOps.		29/1/25	5/2/25	5/2
3.	Jenkins Setup on AWS.		5/2/25	12/2/25	
4.	Build WAR file in Devops		12/2/25	19/2/25	19/2
5.	Ansible Setup and SSH keys		19/2/25	12/3/25	
6.	Deploy the artifact on the test server .		12/3/25	19/3/25	19/3
7.	Perform automation using jenkins		19/3/25	26/3/25	
8.	Build and deploy a grid for chrome and Firefox based testing		26/3/25	9/4/25	9/4
9.	Create deployment resource using Kubernetes.		9/4/25	16/4/25	
10.	Create a docker image for any application using Docker file and push it to docker hub .		16/4/25	23/4/25	
11.	Setup Grafana for Devops		23/4/25	23/4/25	23/4
12.	Setup Prometheus for Devops		23/4/25	23/4/25	23/4

Experiment-1

Start DevOps with a Workflow that Includes Four Phases: To Do, in Progress, Code Review, and Done

Date: 08/01/25

PROCEDURE

Step-1: Define the Workflow

1. **Identify the Phases**

- **To Do:** Tasks that are planned but not yet started.
- **In Progress:** Tasks currently being worked on.
- **Code Review:** Completed tasks undergoing peer review or automated testing.
- **Done:** Fully completed and verified tasks.

2. **Map Workflow Rules**

- Define what criteria move a task from one phase to the next.
- **Example**
  - To Do → In Progress: A developer starts working on the task.
  - In Progress → Code Review: Development is complete; the code is ready for review.
  - Code Review → Done: All reviews and tests are approved.

Step-2: Set Up a Task Management Tool

1. **Choose a Tool**

- Popular options include **Jira**, **Trello**, **Azure DevOps**, **GitHub Projects**, or **GitLab**.

- Peer reviews are complete with no outstanding issues.
- Deployed to the production environment or ready for deployment.

2. **Move Task to Done**

- Transition the task to the **Done** column in the tool.
- Update any documentation or notify stakeholders, if needed.

Step-6: Continuously Improve Workflow

1. **Conduct Retrospectives**

- Regularly review the workflow to identify bottlenecks or inefficiencies.

2. **Optimize Automation**

- Add more automation to reduce manual effort, such as deployment pipelines.

3. **Enhance Communication**

- Use tools like Slack or Microsoft Teams for instant updates on task progress.

SOURCE CODE

PHASE 1: To Do

- Objective: Identify and prioritize tasks or features to be developed.
- Key Actions:
  - Define tasks clearly in a backlog.
  - Prioritize tasks based on impact, urgency and dependencies.
  - Assign owners or teams to each task.

PHASE 2: IN PROGRESS

- Objective: Actively work on tasks selected from the "To Do" phase.

- Key Actions:
  - Begin coding or configuring based on task requirements.
  - Update the task status to reflect ongoing work.
  - Ensure team members collaborate effectively (eg:- stand ups, pair programming)

- Best Practices:
  - Use branches in version control systems for individual tasks (e.g. git feature branches)
  - Write unit tests alongside development.

PHASE 3: CODE REVIEW

- Objective: Validate the quality, functionality and security of the code.
- Key Actions:
  - Submit pull requests for peer review
  - Review code for adherance to standards, logic and potential issues.
  - Approve or request changes.
- Tools:
  github Pull requests, github Merge Requests, Bitbucket
- Automation:
  Integrate CI/CD pipelines to run tests automatically during reviews.

PHASE 4: DONE

- Objective: Mark tasks as completed and deploy changes if necessary.

- Key Actions:
  - Merge the approved code into main branch.
  - Deploy to staging or production environments.
  - Monitor deployment and validate functionality.
- Post Completion:
  - Add documentation for the changes.
  - Gather feedback from stakeholders or users.

WORKFLOW VISUALIZATION

A Kanban board or similar visual representation can help track the status of tasks across the phases. For example:

i) To Do: Contains all pending tasks.

ii) In Progress: Tasks currently being worked for

iii) Code Review: Tasks awaiting review or approval.

iv) Done: Completed and Deployed tasks.

Tools:

Trello, Jira, Azure

A 2911

VIVA QUESTIONS

1. What is DevOps, and why is it important in software development?

Ans. DevOps is a set of practices that integrate development and operations to improve collaboration, automation and efficiency in software development. It helps in faster delivery, better quality and continuous development.

2. What are the key benefits of using phases like To Do, In Progress, Code Review, and Done?

Ans. These phases help track progress, improve task management, enhance collaboration, ensure code quality and streamline workflow efficiency.

3. What are the benefits of version control system?

Ans. It enables tracking changes, collaboration, backup, rollback and maintaining multiple versions of the code efficiently.

4. Can you explain the purpose of a feature branch in git?

Ans. A feature branch allows developers to work on new features independently without affecting the main codebase, ensuring safer integration.

5. What techniques can be used to improve automation in a DevOps environment?

Ans. Techniques include CI/CD pipelines, infrastructure as code (IaC), automated testing, containerization (Docker) and configuration management tools (Ansible, Terraform).

29/7

Experiment-2

Setup Eclipse for DevOps

Date: 29/1/25

PROCEDURE

Step-1: Download and Install Eclipse Enterprise Edition

Step-2: Configure Eclipse Workspace

Step-3: Create a Maven Project in Eclipse.

Step-4: Fix Initial Issues in Maven Project

1. Open the pom.xml file and add the following dependencies
2. Servlet API Dependency
3. Selenium Dependency
4. TestNG Dependency

Step-5: Add Compiler Plugin and Surefire Plugin

Step-6: Manually Create Missing Source Folders

Step-7: Install TestNG Plugin in Eclipse

Step-8: Configure Git Plugin in Eclipse

Step-9: Push Project to GitHub

Step-10: Verify Setup

SOURCE CODE

STEP-1 : Install Jdk-17 and set the java path in system environment

STEP-2 : Download eclipse zip file and extract the contents of the eclipse file.

STEP-3 : Create a maven project from eclipse as :
Click on File in left corner → click on new → Click on Maven Project and follow the below steps :-

a. Click Next in the New Maven Project dialog box that appears.
b. Search for org.apache.maven.archetypes and select webapp file.
c. In groupId you can type anything like your name and in ArtifactId type anything like your roll number. Select war as package
d. click Finish.
e. Type Y and press enter. You should see a Build succes message.

STEP-4 : Now open your pom.xml file and add your dependencies.

STEP-5 : Update your project once (Right click on Project → click on Maven → Click on Update project)

STEP-6 : Download Apache Tomcat V9 from official website

STEP-7 : After downloading the Apache Tomcat, extract the .zip file and paste your apache-tomcat 9.0.98 folder

STEP-8 : Now, click on your project option in Menu → Click on Properties → click on Targeted Runtime.

STEP-9 : Click on new.

STEP-10 : Select Apache Tomcat V9.0. Click on Next

STEP-11 : click on Browse and select your extracted file and then click on finish.

STEP-12 : Now click on Help Menu → click on Install new Software

STEP-13 : Click on Add and it will show a popup dialog of Add Repository.

In the place of Name type : TestNG

In the place of Location type : https://testng.org/testng-eclipse-update-site/

STEP-14 : Click on Add → It will load a testNG Dependencies → Select TestNG and then click Next. It will take 10 minutes to update TestNG in our project.

STEP-15 : After downloading all the dependencies, it will show some file select all and click on next.

STEP-16 : Accept Terms and Conditions and click on Finish.

STEP-17 : After finishing, it will show Restart option (Restart the project) otherwise just update once of your project.

STEP-18 : Now login your github Account.

STEP-19 : Create a New Repository and copy your repository and paste in Notepad.

STEP-20 : After that, click on your profile in right corner → click on setting.

STEP-21 : It will show a new page, scroll down and select the Developer setting → click on personal accen token → Select Token (classic) → click on generate new token and select generate new token (classic) → Write your token name and select repo option and scroll down and click on generate token.

STEP-22 : After generating the token, copy the token id and paste it in a Notepad.

STEP-23 : Now come on your project and right click on your project → click on Team → click on share project.

STEP-24: It will open a Dialog BOX for github setup, select the option use or create repository in parent folder of project → select your project and click on create Repository and click on Finish.

STEP-25: After that again, right-click on your project and select the Team → click on commit → and stage your all file → and write a comment (i.e., First commit) and click on Commit and push → After that it shows on error dialog → click OK → Now again click on Push Head button.

STEP-26: After than, again click on Push Head. It will show a dialog like of Push Branch master. Paste your Repository URL in URL section and type your github ~~user Id~~ and Password in User, password ~~section~~ → click on Preview → Again click on Preview.

STEP-27: Then, it will again show a user Id & password option → Just type your github id in user section and paste your token id in password token → click on push → one more time it will ask your user Id and password, just repeat your last step → Now check your repository on github, if your file is uploaded or not.

STEP-28: create a simple java code in SRC file, so first open your project from file manager → open src → create folder in src → First name: java, ~~second~~ name: test → open test folder and create ~~two~~ more folders in test folder → come to your Eclipse IDE and update your project once → After ~~that~~ create a java class file with a statement "Hello World" in your src/test/java folder.

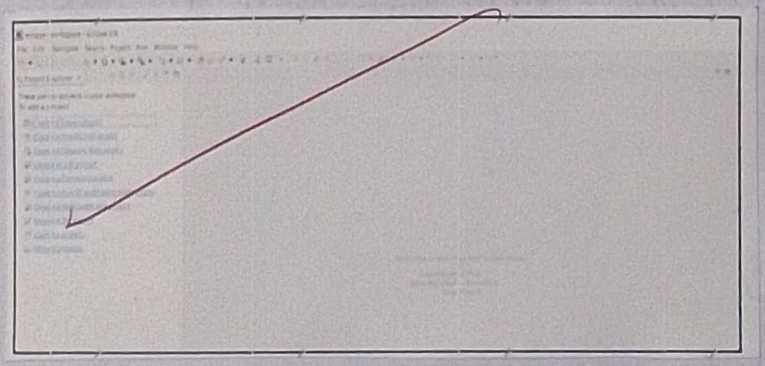
STEP-29: Now push again your all ~~mes~~tage files in your github Repository with different version or comment (its just for version control)

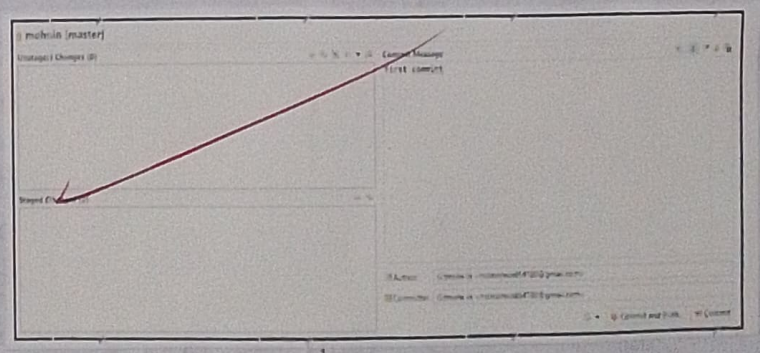
STEP-30: Now check your repository again to see whether your recent file is uploaded or not.

```
<dependencies>
    <dependency>
        <groupId> junit </groupId>
        <artifactId> junit </artifactId>
        <version> 3.8.1 </version>
        <scope> test </scope>
    </dependency>
    <dependency>
        <groupId> javax.servlet </groupId>
        <artifactId> javax.servlet-api </artifactId>
        <version> 4.0.1 </version>
        <scope> provided </scope>
    </dependency>
    <dependency>
        <groupId> org.seleniumhq.selenium </groupId>
        <artifactId> selenium-java </artifactId>
        <version> 4.27.0 </version>
    </dependency>
    <dependency>
        <groupId> org.testing </groupId>
        <artifactId> testing </artifactId>
        <version> 7.10.2 </version>
        <scope> test </scope>
    </dependency>
</dependencies>
<build>
  <plugins>
    <plugin>
      <groupId> org.apache.maven.plugins </groupId>
      <artifactId> maven-surefire-plugin </artifactId>
      <version> 3.5.2 </version>
    </plugin>
    <plugin>
      <groupId> org.apache.maven.plugins </groupId>
```

OUTPUT

```
<artifactId> maven-compiler-plugin </artifactId>
<version> 3.13.0 </version>
<configuration>
    <source> 17 </source>
    <target> 21 </target>
</configuration>
</plugin>
</plugins>
<finalName> 6229 </finalName>
</build>
</project>
```





A 5/2

VIVA QUESTIONS

1. Why do we use Maven in projects?

Ans. Maven automates build process, manages dependencies, and ensures a standardized project structure.

2. Explain the structure of a Maven project?

Ans. A Maven project follows a standard structure: src/main/java (source code), src/main/resources (config files), src/test/java (test code), pom.xml (dependency and build configuration)

3. What are Maven archetypes, and why did you select the maven-archetype-webapp?

Ans. Maven archetypes are project templates. The maven-archetype-webapp is used to create a basic web application structure.

4. What issues might arise during the creation of a Maven project in Eclipse, and how can you resolve them?

Ans. Issues may include missing dependencies, plugin errors, or network issues. Solutions: Update Maven, clean the project and check internet connectivity.

5. How did you find and add dependencies for Selenium and TestNG in Maven?

Ans. Search for dependencies on Maven Repository and add them to pom.xml.

6. Show how to add a new dependency to the pom.xml file and explain its impact.

Ans. Ex:-

```
<dependency>
    <groupId>org.seleniumhq.selenium</groupId>
    <artifactId>selenium-java</artifactId>
    <version>4.1.2</version>
</dependency>
```

Impact: Maven downloads and manages the required library automatically.

7. Walk through the process of pushing a code change to a GitHub repository from Eclipse.

Ans.
- Commit changes in Eclipse (Team → Commit)
- Push to Github using Team → Push to Upstream

Experiment-3

Jenkins Setup on AWS

Date: 5/2/25

PROCEDURE

Step-1: Create an AWS Account

Step-2: Launch an EC2 Instance

Step-3: Access Your EC2 Instance

Step-4: Configure the Instance

1. Switch to root user
2. Create a non-root user
3. Grant sudo privileges

Step-5: Install Required Tools

1. Update packages
2. Install Java
3. Install Git
4. Install Maven

Step-6: Install Jenkins

1. Add the Jenkins repository
2. Install Jenkins
3. Start Jenkins
4. Verify Jenkins is running

Step-7: Access Jenkins

1. Open your browser and navigate to
2. Retrieve the initial admin password
3. Copy the password and paste it into the Jenkins setup screen
4. Select **Install suggested plugins** during setup
5. Create an admin user

Step-8: Verify Jenkins Setup

SOURCECODE

Step1 : Search AWS FreeTier Account on any browser and create an account (If you already have an account, login)

Step2 : After creating the account, login as ROOT user.

Step3 : After Login it will show your account > click on EC2 option> click on Launch Instance

Step4: After clicking a launch Instance, it will ask Name and other things > Write Instance Name as Jenkins > Select application and IOS Image as Ubuntu > scroll down and come on Instance Type option and select t2 Medium. Now come in Key Pair (Login) section and click on create New Key pair > write your key pair name as Exp3 > select .pem> click on Create Key Pair.

Step5: After click on create Key Pair it will download a Exp3.pem file > After that on the right-side we have an option Number of instance select 1 > click on Launch Instance

Step6 : After that refresh the page > select the created instance > click on security.

Step7 : After clicking on security > It will show a blue link name as security Group > click on the blue link it will open a page> click on edit inbounds rule > click on Add Rule, Add one rule HTTP and source type Anywhere IPV4 > Add one more rule Custom TCP, Port range 8080 and source Anywhere IPV4 > click on save Rule.

Step 8: Now search Download Mobaxterm in your browser > click on download or first-Link > click on Home Edition or free Edition > click on Mobaxterm Portable Edition.

Step 9: After downloading the Mobaxterm.zip file right click and extract all and paste it in your DevOps folder or Anywhere.

Step 10: Now open Mobaxterm Extracted Folder and double click on .exe file of application type file > It will open Mobaxterm software

Step 11: Now click on Session (left upper corner) > click on SSH > It will ask remote host and specify username.

Step 12: In the place of Remote Host Place the Public IP of your instance > click on Advance SSH setting > select use private key > Browse the .pem file that you already downloaded during instance creation > After that click on OK > It will open a Linux Command Prompt.

Step 13: Now you have to setup Jenkins > search Jenkins Document in any browser > click on first Jenkins link > click on Debian / Ubuntu > After that it will show the Linux Command.

Step 14: Now come to your Mobaxterm Application > Type clear and Hit Enter key > It will clear your command screen > Now copy the Long term Support release code till /dev/null > Paste in your Linux terminal and hit the Enter key > It will download the Jenkins support file.

Code: sudo wget -O /usr/share/keyrings/jenkins/keyring-src\https://pkg.jenkins.io/debian-stable/jenkins.io-2023.key
echo "deb [signed-by=/usr/share/keyrings/jenkins-keyring arc]" https://pkg.jenkins.io/debian_stable binary/
sudo tee /etc/apt/sources.list.d/jenkins.list > /dev/null

Step 15: Clear your (section) screen > Now update your project using given code code: sudo apt-get update

Step 16: Clear your screen and check java version using command: java --version

Step 17: If it shows java Not Found then it shows the option install java. So install java using the command. It will ask Y/N, just type Y and hit enter. It will take some time to download java.

Code: sudo apt install fontconfig openjdk-17-jre

Step 18: Now Enable the Jenkins server using given command.
Command: sudo systemctl enable jenkins.

Step 19: After that enable the jenkins server start Jenkins server using the command: sudo systemctl start jenkins.

Step 20: After downloading java, clear your screen using clear command > Once again update your project using update command (step 14). Now install Jenkins using Jenkins command. It will ask Y/N, just type Y and hit Enter. After installing Jenkins once again update your project using update command. check java and jenkins version using (java --version and Jenkins --version) commands
code: sudo apt-get install jenkins.

step 21: After starting the jenkins server, Check jenkins server status using given command. It will show the status active running and shows the status of memory, CPU, etc.
Command: sudo systemctl status jenkins

Step 22: Now open any browser and search the jenkins server using instance IP and Custom Port Number.
(IP address followed by 8080)

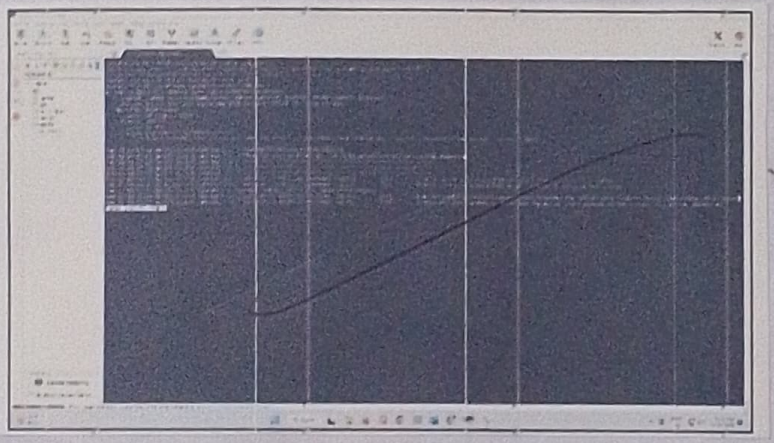
Step 23: It will start the jenkins server, show some path with red colour and ask the administrator password.

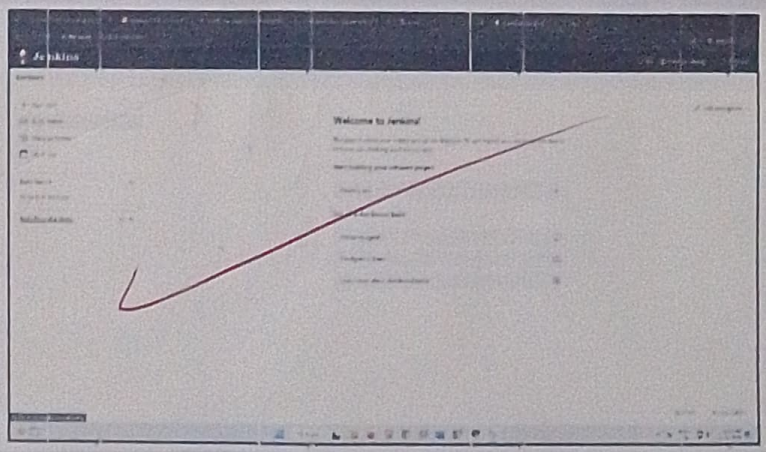
Step 24: Copy the red colour path and paste it in your linux terminal followed by sudo cat command and hit the Enter key. It will show 64-bit alpha numeric code > Copy that code and paste in the place of password in your browser > click on Continue.

Step 25: After clicking on continue option, it will show 2 options for installing the plugin, just click on install suggested plugin. After installing the plugins, it will ask creating an account using userid and password. Click on save and continue.

OUTPUT

Step 26: After clicking on save and continue, it will show more links and show the option save and Finish. Click on save and Finish. After that it shows Jenkins is Ready and shows an option start using Jenkins. Just click on it. It will open a Jenkins dashboard.





VIVA QUESTIONS

1. What is an Amazon Machine Image (AMI), and how did you select the CentOS AMI?

Ans. An AMI is a preconfigured VM template. CentOS AMI is selected based on version and compatibility.

2. How do you configure SSH access for an EC2 instance in MobaXterm or PuTTY?

Ans. Load the .pem key in MobaXterm or convert it to .ppk for PuTTY, then connect using the instances public IP address via SSH.

3. What is the difference between a private and a public IP address in an EC2 instance?

Ans. Public IP is internet-facing, allowing external access. Private IP is internal, enabling communication within the AWS network.

4. Why is setting the JAVA_HOME environment variable important?

Ans. It is important because it specifies the java installation path, ensuring tools like maven and IDE's locate the correct Java version.

5. How do you start and verify the Jenkins service on CentOS?

Ans. Start jenkins with systemctl start jenkins and verify with systemctl status jenkins or by accessing http://:8080 in a browser.

Experiment-5

Build WAR file in DevOps

Date: 12/2/25

PROCEDURE

Setting Up Jenkins and Building a WAR File

Step-1: Verify Jenkins Server is Up

Step-2: Log into Jenkins

Step-3: Create a New Jenkins Job

Step-4: Install Maven Plugin

Step-5: Create a Maven Project

Step-6: Configure Source Code Management

Step-7: Configure Maven Settings

Step-8: Save and Run the Job

Step-9: Monitor the Job

1. Navigate to the **Console Output** of the job.
2. Check for the build process and ensure the following:
   - The WAR file is built successfully.
   - The file is located at Jenkins/workspace/Build_WAR_File/target/<your-file-name>.war

SOURCE CODE

Build war file:

- Connect dev terminal through git
- check java is installed or not. Install it if not already installed.
- Check if maven is installed or not. If not, install it.
- Check if jenkins is installed or not. If not, install it.

Maven download:

- Go to the root directory: sudo su -
- cd /opt
- Open the browser and type maven download.
- Right click on the apache-maven 3.9.9 bin.tar.gz and copy the link.
- Enter command: wget https://dlcdn.apache.org.maven.org/maven/maven-3/3.9.9/binaries/apache-maven-3.9.9-bin.tar.gz
- Unzip the maven file:

  tar -xvzf apache-maven-3.9.9-bin.tar.gz

- ll
- Rename apache_maven_3.9.9 to maven:

  mv apache_maven_3.9.9 maven

- ll
- cd maven
- ll
- Note down the maven path: /opt/maven
- Move to bin: cd bin
- ll
- Note down maven bin path: /opt/maven/bin
- pwd
- Now check if maven is installed or not:

  mvn --version

(or)

./mvn --version

- Now go back to the root directory: cd ~
- check maven version: mvn --versim
  It shows maven not found.
  so we need to create environment variable
- Go to root directory: cd ~
- pwd
- ll
  we can see .profile file
- Open the file: sudo vim .profile
- Go to insert mode (click on I) and give the maven, java home and m2 paths here.
- M2_HOME = /opt/maven
  M2 = /opt/maven/bin
  JAVA_HOME = /usr/lib/jvm/java-22-openjdk_amd64
  PATH = $PATH:$HOME/bin:$JAVA_HOME:$M2_HOME:$M2
  :wq
- To get java_home path: find / -name java-21*
- echo $PATH
  You can see the java and maven path above. So we need to restart the .profile file with the below command.
- source .profile
- Now we can see the java and maven paths.
- check maven version.
  mvn --version (we can see maven now)
- Go to jenkins dashboard. We need to install one plugin (maven integration)

Manage jenkins > plugins > available plugins > maven integration plugin

- Without this plugin, we can't see the maven project.
- Once installed, click on restart jenkins.
- Add java and maven paths in jenkins.
- Go to jenkins dashboard > manage jenkins > tools
  Add JDK and add MAVEN
- Uncheck install automatically and add maven.
- Click on apply and save
- Now we can create job for war file.
- Dashboard > new item > war (give any name)
  select maven project, click on OK.
- Go to github, select/ copy repo url with pom.xml file and paste in jenkins repo url.
- Select build goal and options, write the command clean install and click on apply and save.
- Click on Build Now.
- If job execution is taking or lot of time, then logout jenkins.

Stop the dev instance.

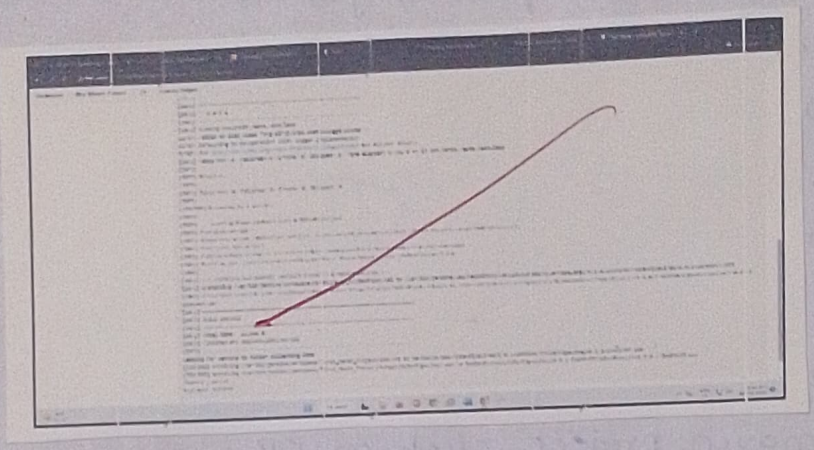
restart the dev instance

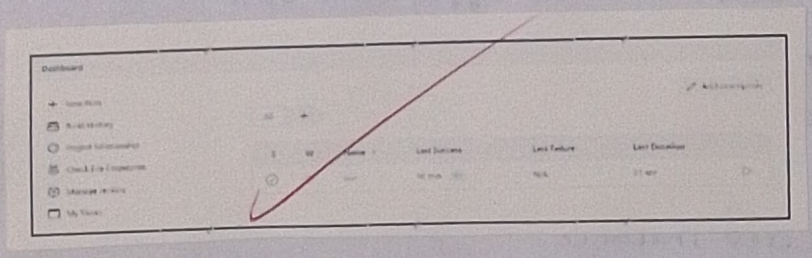
start the jenkins server

Run the job

Now your job will be executed.

OUTPUT





VIVA QUESTIONS

1. What is a WAR file, and where is it used?

Ans. A WAR file is a compressed archive used to deploy and package Java web applications on servers.

2. What steps would you take if the Jenkins server is not up?

Ans. Check Jenkins service status, review logs verify open ports, ensure efficient resources and restart the service.

3. What is the difference between "Install without restart" and "Install and restart" in plugin installation?

Ans. "Install without restart" install the plugin without restarting Jenkins. "Install and restart" install the plugin and immediately restarts Jenkins to apply changes.

Experiment-A5

Ansible Setup and SSH Keys

Date: 19/2/25

PROCEDURE

Step-1: Launch Jenkins Server

Step-2: Create Ansible Controller Node

1. Launch the Ansible Controller Node
2. Rename the Instance
3. Connect to the Ansible Controller
4. Create a Non-Root User
5. Grant Sudo Privileges
6. Enable Password Authentication

Step-3: Install Ansible

Step-4: Launch Ansible Managed Nodes

Step-5: Configure Ansible Hosts File

Step-6: Establish SSH Key-Based Authentication

Step-7: Verify Connectivity

Step-8: Build and Push Docker Image Using Jenkins

1. Create a Jenkins Job
2. Push Image to Docker Hub

SOURCECODE

Create 3 AWS ec2 instances in ubuntu.

First Instance - Ansible
second instance - server1
Third instance - server2

Login To Ansible ec2 instance and use these commands

- switch as root: sudo su -
- update packages: apt update -y
- run the following command to include the official projects PPA (Personal package archive) in your systems List of source:
  apt -add -repository ppa: ansible/ansible
- Next, refresh your systems package index so that it is aware of the packages available in the newly included PPA: apt update
- Following this update, you can install the ansible software with:
  apt install ansible -y
- Check ansible version: ansible --version
- Go to hosts and add your server1 and server2 nano /etc/hosts
  Add: 15.168.39.192 server1
  13.208.141.45 server2
- Generate ssh key from ansible server
  ssh -keygen -t rsa (and) press → enter → enter → enter
  you can see ssh keys of public key and private key.
- Copy the public key (id_rsa.pub) and paste it in authorize-key on server1 and server2.
  cat id_rsa.pub
- Go to server1 and server2.
- Login server1 and paste this public key in .ssh/authorize/key
  nano .ssh/authorize_keys
  (save it and come out from the shell)

- Login server2 and paste this public key in .ssh/authorize_key
  nano .ssh/authorize_keys
  (save it and come out from the shell)
- Return to the ansible server and check if ping is working on server1 and server2.
  ping server1
  ping server2
- Create a directory in the name of ansible
  mkdir ansible
- Get in the ansible directory
  cd ansible
- Create an inventory file and then add these lines hosts.
  nano inventory
  [webseavers]
  server1
  server2
  (save it and come out from shell)
- Create ansible.cfg file and add these lines
  nano ansible.cfg
  [default]
  inventory = /root/ansible/inventory
  remote_user = ubuntu
  ask_pass = false
  (save and come out of it)
- For testing purpose, we need to install nginx in server1 and apache in server2 from ansible server.
- Create a yml file for install nginx and apache in server1 and server2.
  nano install_webservers.yml

  - name : Install web servers
    host : webservers
    become : true
    tasks :

OUTPUT

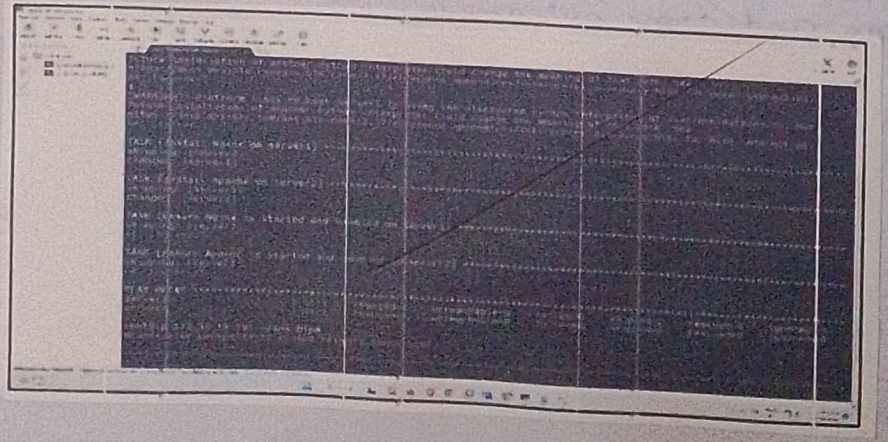
```
- name: Install Nginx on server1
  apt:
    name: nginx
    state: present
  when: inventory_hostname == 'server1'
- name: Install Apache on server2
  apt:
    name: apache2
    state: present
  When: inventory_hostname == 'server2'
- name: Ensure Nginx is started and enable on server1
  service:
    name: nginx
    state: started
    enabled: yes
  when: inventory_hostname == 'server1'
- name: Ensure Apache is started and enable on server2
  service:
    name: apache2
    state: started
    enabled: yes
  when: inventory_hostname == 'server2'
```

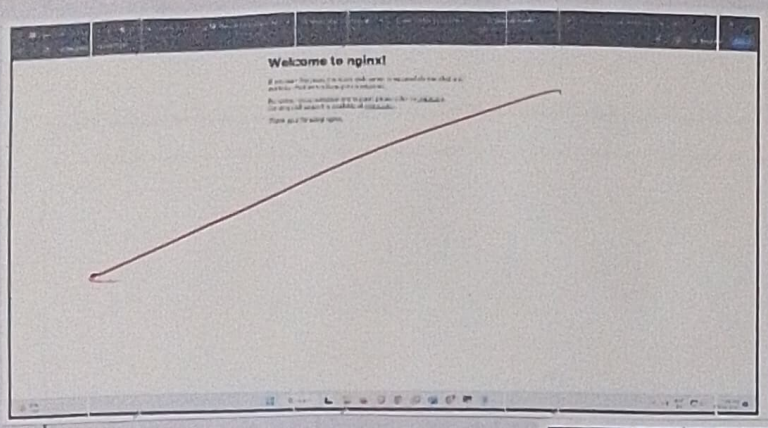
(save and come out from shell)

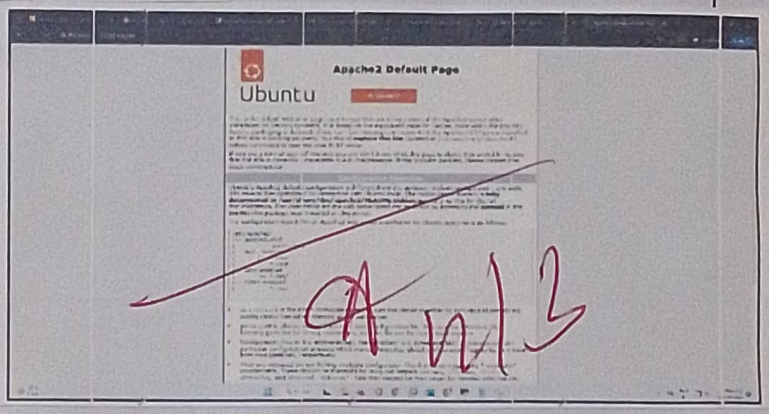
- Run ansible yml file using the following command.

ansible-playbook -i /root/ansible/inventory install-webservers.yml

Here you can see installing nginx and apache each servers and you can test by copy each server ip and paste it in the browser.







VIVA QUESTIONS

1. What is the difference between a Jenkins server and an Ansible controller?

Ans. Jenkins: automate CI/CD pipelines, managing build, test, deployment.
Ansible: automate infrastructure, management and configuration using declarative playbooks.

2. Why is Jenkins often integrated with tools like Ansible in DevOps pipelines?

Ans. To automate infrastructure provisioning, configuration management, and deployment ensuring efficient DevOps flows.

3. How do you create an Ansible controller node on AWS?

Ans. Launch an EC2 instance, install Ansible using yum or apt, configure SSH access, and define inventory and playbooks.

4. How can you test if the Ansible controller can communicate with the managed nodes?

Ans. Run ansible all -m ping to check connectivity between the ansible controller and managed nodes or using ping.

5. How does Jenkins push Docker images to Docker Hub?

Ans. Jenkins builds image, logs into Docker Hub, tags the image, and pushed it using docker push.
<repository>:<tag>