

# Course-End Project

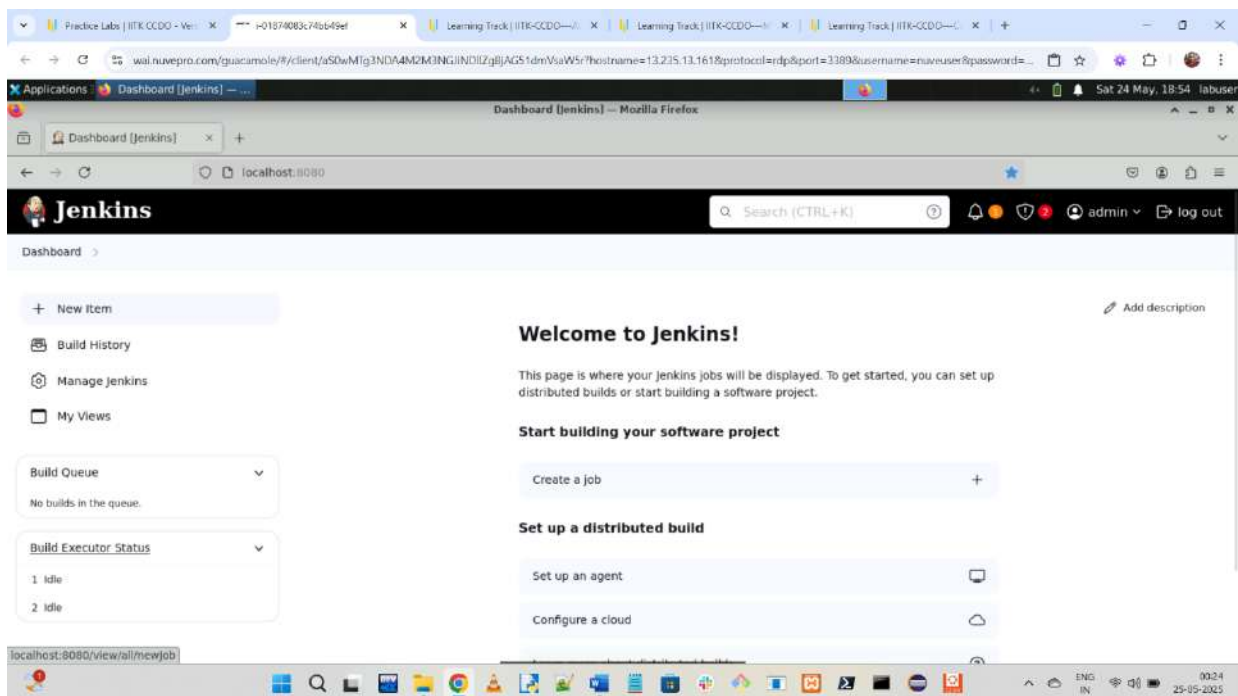
## Application Deployment to Multi-Cloud

### Steps to be followed:

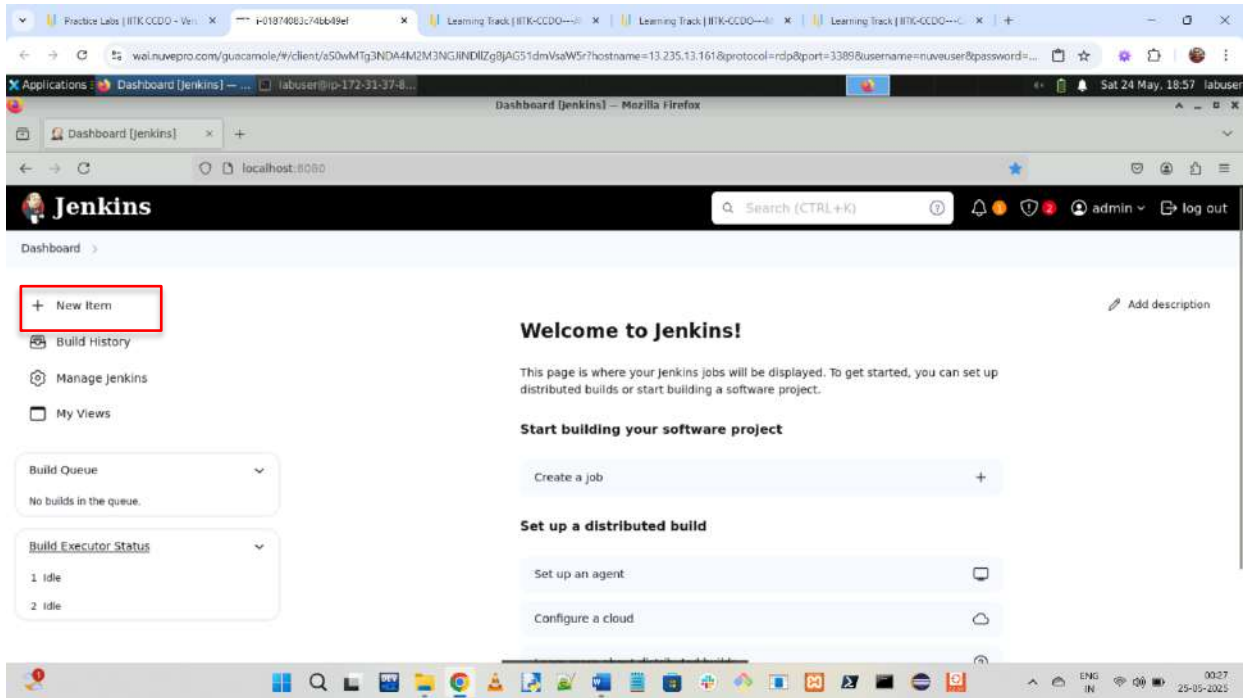
1. Create Jenkins pipeline for CI of Java application
2. Configure Ansible to connect AWS and Azure VMs
3. Configure Docker in Jenkins for CI/CD
4. Configure Ansible on Simplilearn lab
5. Validate Deployment

### Step 1: Create Jenkins pipeline for CI of Java application

- 1.1 Navigate to the DevOps lab where Jenkins is preinstalled, Open a browser and access Jenkins at <http://localhost:8080/>



- 1.2 Open Jenkins and click on **New Item** to create a new Jenkins job



Before proceeding further open terminal and then validate if ansible, docker and maven runtimes are installed on VM

- 1.3 Execute the following command to check the Docker version and confirm that Docker is correctly installed on your system

**docker version**

```
(base) labuser@ip-172-31-37-85:~$ docker version
Client: Docker Engine - Community
Version: 27.1.1
API version: 1.46
Go version: go1.21.12
Git commit: 6312585
Built: Tue Jul 23 19:59:47 2024
OS/Arch: linux/arm64
Context: default

Server: Docker Engine - Community
Engine:
Version: 27.1.1
API version: 1.46 (minimum version 1.24)
Go version: go1.21.12
Git commit: cc13195
Built: Tue Jul 23 19:59:47 2024
OS/Arch: linux/arm64
Experimental: false
containerd:
Version: 1.7.19
GitCommit: 2bf793ef6dc9a18e0cb12efb64355c2c9d5eb41
runc:
Version: 1.7.19
GitCommit: v1.1.13-0-g58aa920
docker-init:
Version: 0.19.0
GitCommit: de40ad0
(base) labuser@ip-172-31-37-85:~$ ansible --version
ansible [core 2.15.0]
```

- 1.4 Execute the following command to check the Ansible version and configuration details

## ansible version

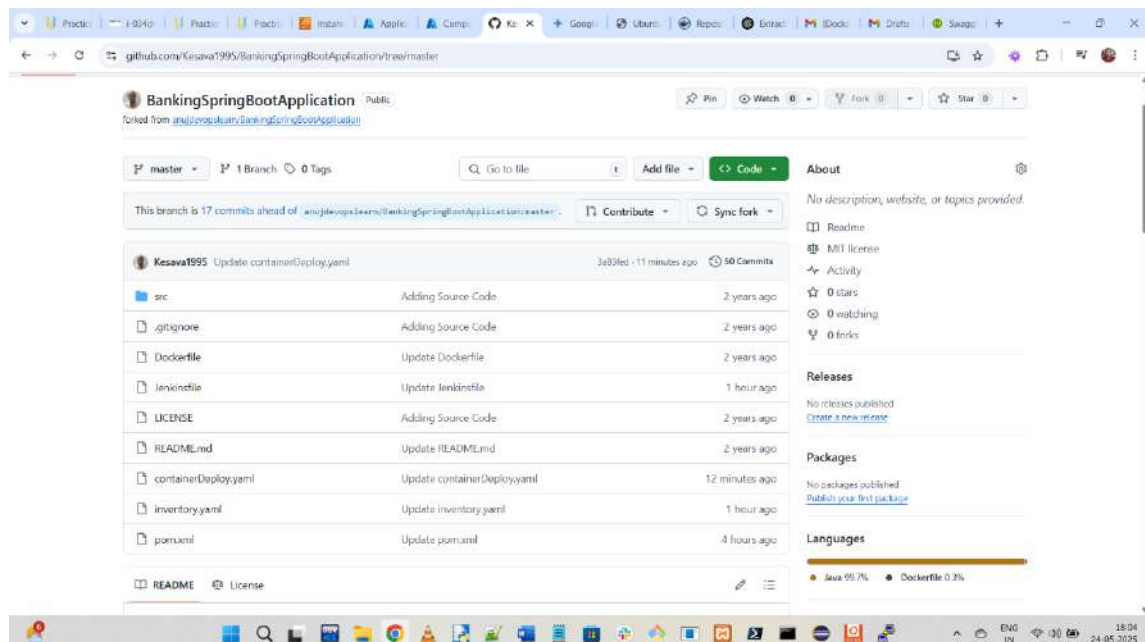


```
labuser@ip-172-31-37-85:~$ ansible --version
ansible [core 2.16.9]
  config file = /etc/ansible/ansible.cfg
  configured module search path = ['/home/labuser/.ansible/plugins/modules', '/usr/share/ansible/plugins/modules']
  ansible python module location = /usr/lib/python3/dist-packages/ansible
  ansible collection location = /home/labuser/.ansible/collections:/usr/share/ansible/collections
  executable location = /usr/bin/ansible
  python version = 3.10.12 (main, Jul 29 2024, 16:56:48) [GCC 11.4.0] (/usr/bin/python3)
  jinja version = 3.0.3
  libyaml = True
(base) labuser@ip-172-31-37-85:~$
```

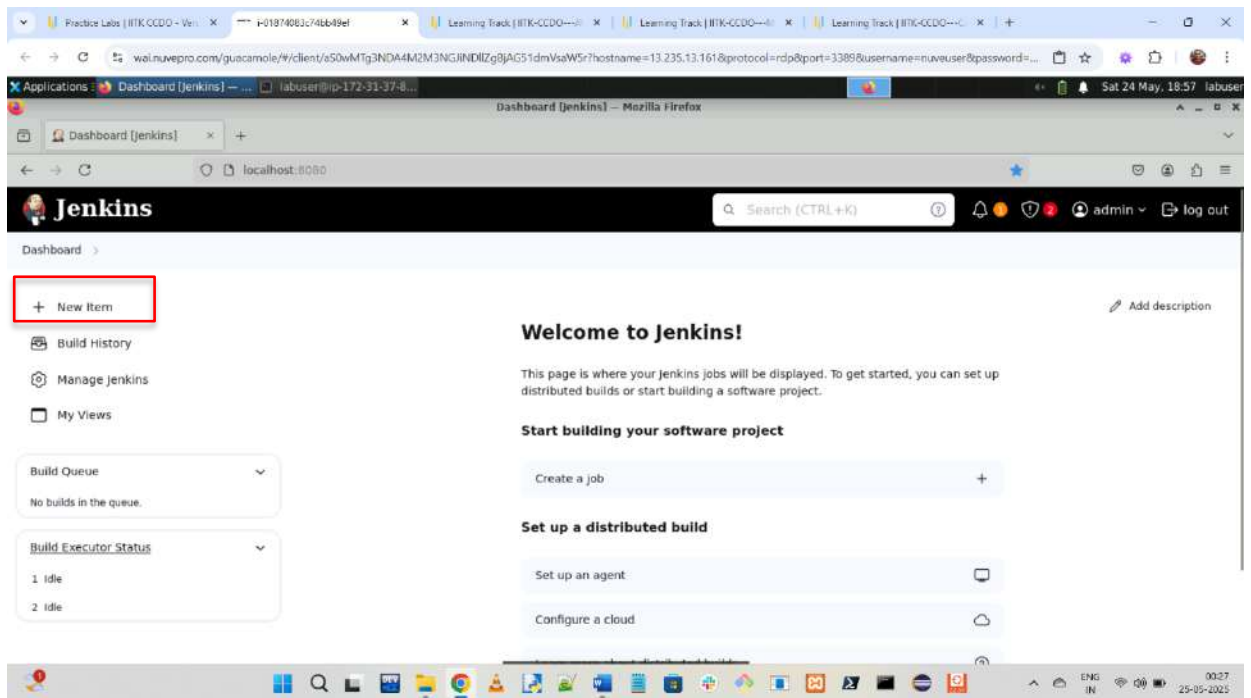
## Step 2: Configure Ansible to connect AWS and Azure VMs

- 2.1 Fork the GitHub repository below to execute the Jenkins pipeline, and validate that Maven scripts are present in the code repository for source code validation

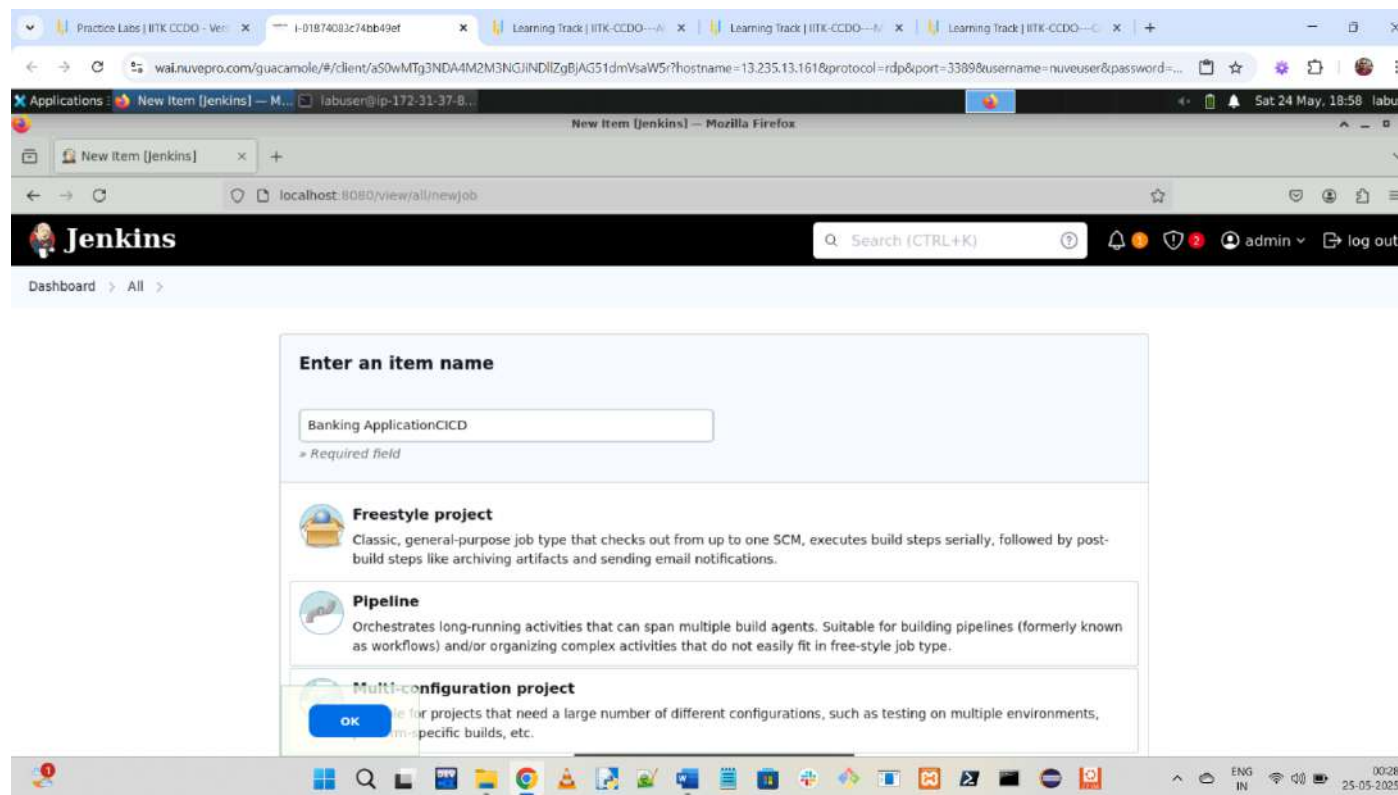
<https://github.com/Kesava1995/BankingSpringBootApplication>



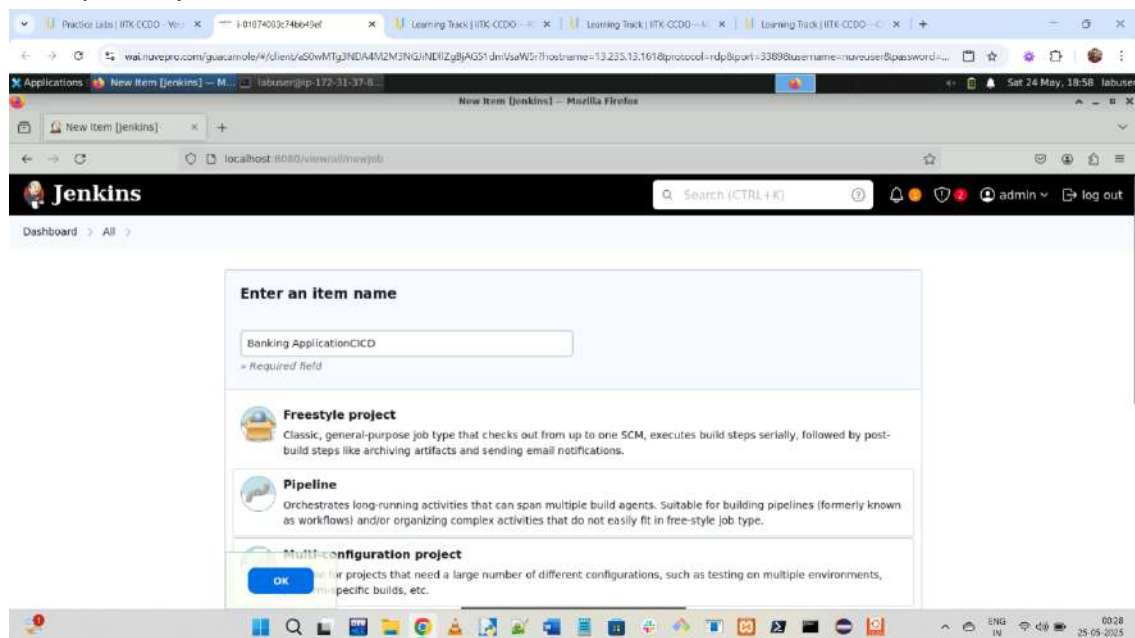
## 2.2 Access Jenkins application and click on **New Item** to create a new Jenkins job



## 2.3 Select desired Jenkins pipeline job type and fill in job name as per project requirement

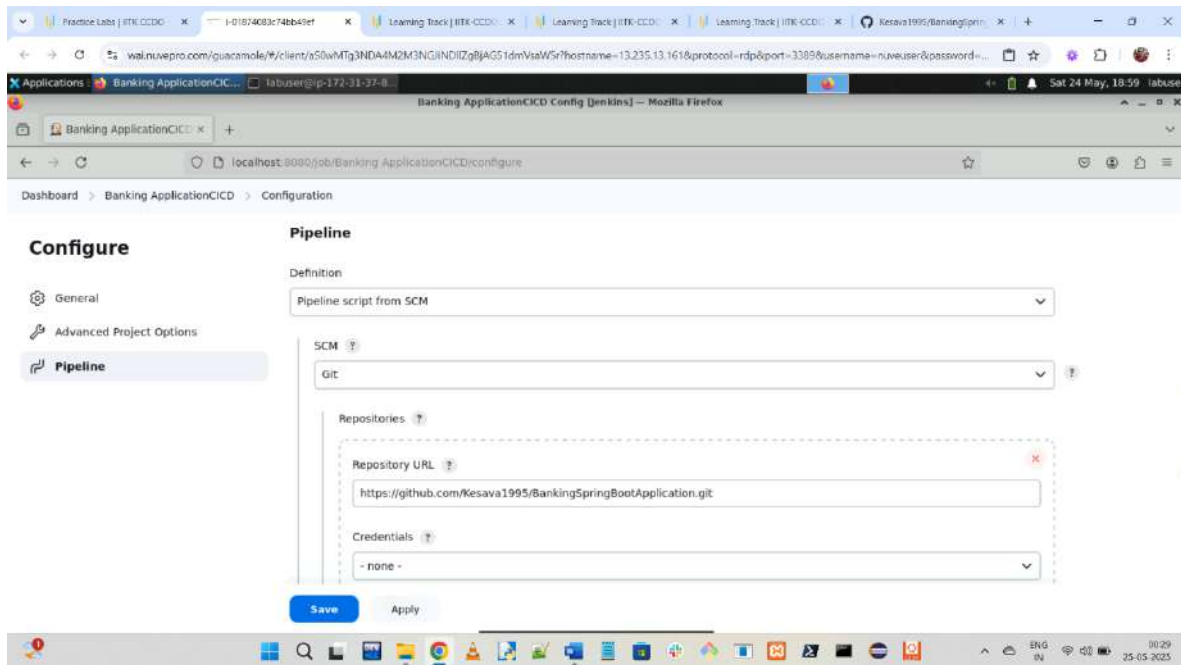


2.4 In your Jenkins project configuration, set the Pipeline Definition to **Pipeline script from SCM**, select SCM as **Git**, and enter the Repository URL where the pipeline script is stored. Once configured, click **Save** to apply the changes, enabling Jenkins to fetch the pipeline script directly from the specified Git repository



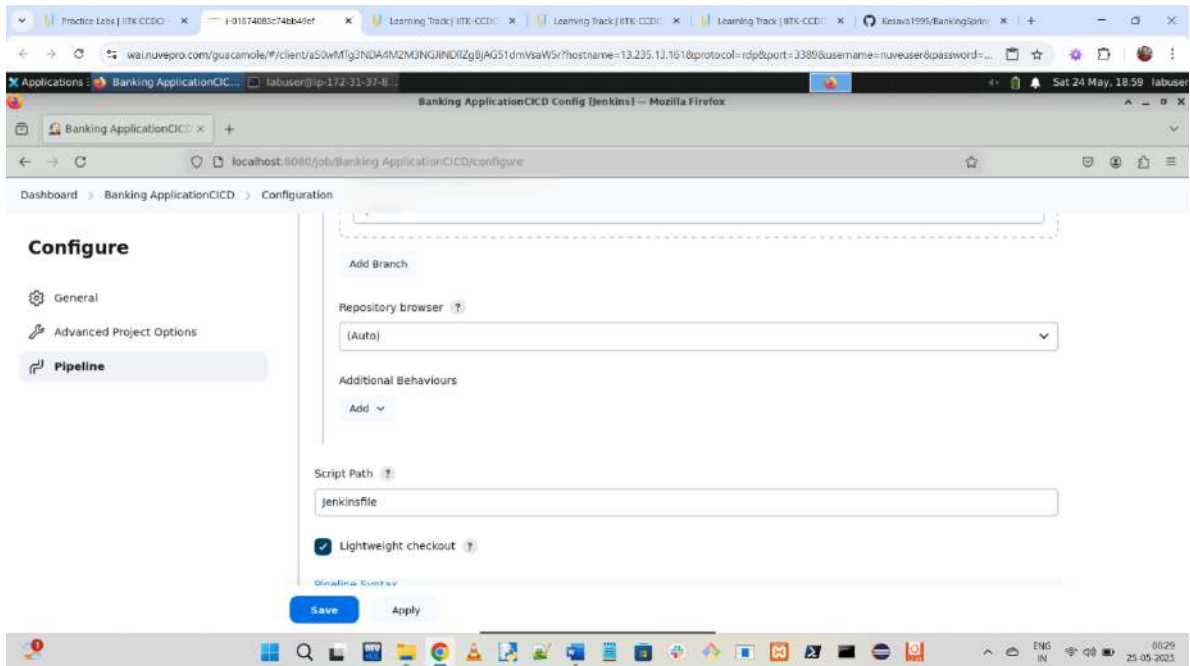
2.5 Configure below code repository or forked repository in Jenkins pipeline to checkout Pipeline script from Git version control system

<https://github.com/Kesava1995/BankingSpringBootApplication>



**Note:** In above code repository we are using default Jenkins pipeline name, but we can modify in case we are using a different pipeline name

2.6 Set the Script Path to **Jenkinsfile** and enable Lightweight checkout in your Jenkins pipeline configuration, then click **Save** to apply these settings and ensure Jenkins can locate and load the pipeline script efficiently



2.7 Create a **Jenkinsfile** in your code repository and add Code Checkout stage in the pipeline script

```
node{
```

```
    def tag, dockerHubUser, containerName, httpPort = ""
```

```
    stage('Prepare Environment'){
```

```
        echo 'Initialize Environment'
```

```
        tag="3.0"
```

```
        withCredentials([usernamePassword(credentialsId: 'dockerHubAccount', usernameVariable:
'dockerUser', passwordVariable: 'dockerPassword'))] {
```

```
            dockerHubUser="$dockerUser"
```

```
        }
```

```
        containerName="bankingapp"
```

```
        httpPort="8989"
```

```
    }
```

```
    stage('Code Checkout'){
```

```
        try{
```

```
            checkout scm
```

```
        }
```

```
        catch(Exception e){
```



```

        echo 'Exception occurred in Git Code Checkout Stage'
        currentBuild.result = "FAILURE"
    }
}

stage('Maven Build'){
    echo '--- Diagnosing Maven Build ---'
    echo '1. Current Working Directory:'
    sh 'pwd'
    echo '2. Contents of pom.xml:'
    sh 'cat pom.xml' // CRITICAL: This will show us what pom.xml Jenkins is actually using
    echo '3. Java Version on Jenkins Agent:'
    sh 'java -version' // CRITICAL: Confirm the JDK version
    echo '4. Running Maven with debug logging:'
    sh 'mvn clean package -X' // CRITICAL: This will provide verbose output
    echo '--- End Diagnosis ---'
}

stage('Docker Image Build'){
    echo 'Creating Docker image'
    sh "docker buildx build --platform linux/amd64 -t $dockerHubUser/$containerName:$tag --load ."
}

stage('Publishing Image to DockerHub'){
    echo 'Pushing the docker image to DockerHub'
    withCredentials([usernamePassword(credentialsId: 'dockerHubAccount', usernameVariable:
'dockerUser', passwordVariable: 'dockerPassword'))] {
        sh "docker login -u $dockerUser -p $dockerPassword"
        sh "docker push $dockerUser/$containerName:$tag"
        echo "Image push complete"
    }
}

stage('Ansible Playbook Execution'){
    withCredentials([string(credentialsId: 'ssh_password', variable: 'password')]) {
        sh "export ANSIBLE_HOST_KEY_CHECKING=False && ansible-playbook -i
inventory.yaml containerDeploy.yaml -e httpPort=$httpPort -e containerName=$containerName -e
dockerImageTag=$dockerHubUser/$containerName:$tag -e ansible_password=$password -e
key_pair_path=/var/lib/jenkins/server.pem --become"
    }
}

```



}

**Note:** The Jenkins file and all the other files are present in the GitHub repository

2.8 Below is the build automation stage which will invoke maven command to perform compilation, test execution and packaging

```
stage('Maven Build'){  
    sh "mvn clean package"  
}
```

**Note:** We need to integrate Docker image build process within your Jenkins pipeline, so we must create a Dockerfile build script which will help in building custom Docker image

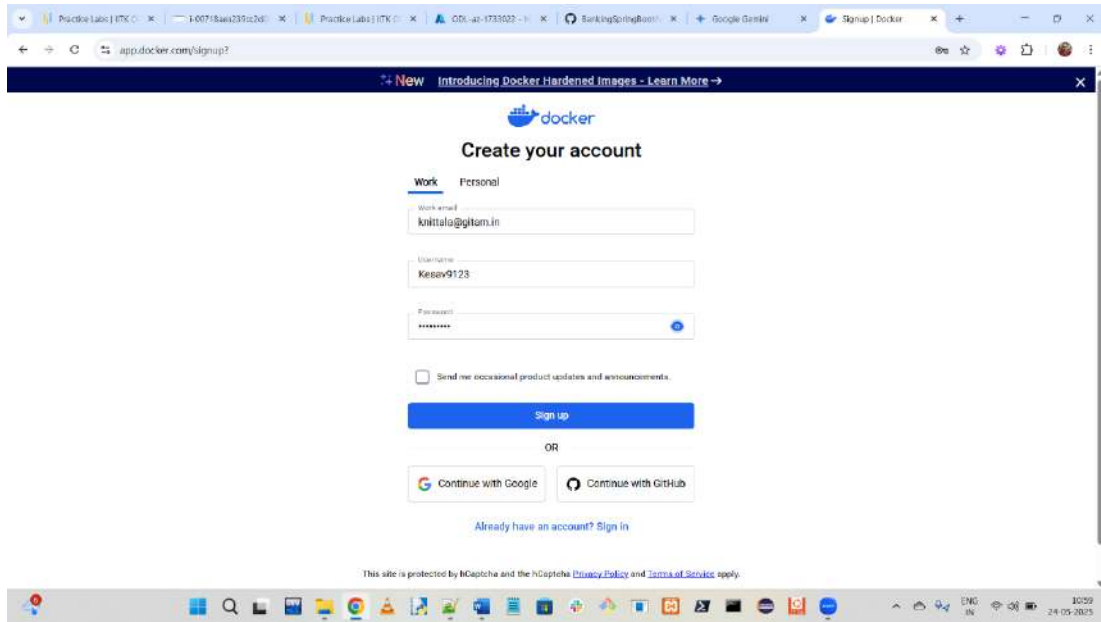
2.9 Create Dockerfile script for Backend:

```
FROM openjdk:11  
ARG JAR_FILE=target/*.jar  
COPY ${JAR_FILE} app.jar  
EXPOSE 8989  
ENTRYPOINT ["java","-jar","/app.jar"]
```

**Note:** Once Jenkins job configuration is done, we must create a Docker credential which will be used to push Docker image to Docker hub

2.10 Navigate to Docker hub and create a new free account and keep Docker hub username and password with you before configuring Jenkins credential

<https://hub.docker.com/>



### Step 3: Configure Docker in Jenkins for CI/CD

- 3.1 Once Docker hub account is created, navigate to **Manage Jenkins** and then **Credentials** and then select **global** domain


Practice Labs | IITK C... | -01874083C74b64b... | Learning Track | IITK... | Learning Track | IITK... | Learning Track | IITK... | Kevsa1995/Banking... | Docker Hub... | +

← → ↻ wai.nuavepro.com/guacamole/4/client/a50wMTg3NDAM2M3NGlNDlZgBjAGS1dmVsaWS?hostname=13.235.13.161&protocol=rdp&port=3389&username=naveuser&password=...

Applications Manage Jenkins [jenkins]... labuser@ip-172-31-37-8

Manage Jenkins [jenkins] — Mozilla Firefox

← → ↻ localhost:8080/manage/

 **Jenkins**

Search (CTRL+K) admin log out

Dashboard > Manage Jenkins

+ New Item

Build History

Manage Jenkins

My Views

Build Queue  
No builds in the queue.

Build Executor Status  
1 idle  
2 idle

Manage Jenkins

New version of Jenkins (2.504.1) is available for download (change log).

Building on the built-in node can be a security issue. You should set up distributed builds. See the documentation.

Warnings have been published for the following currently installed components:  
Jenkins 2.452.3 core and libraries:  
Multiple security vulnerabilities in Jenkins 2.470 and earlier, LTS 2.452.3 and earlier  
Multiple security vulnerabilities in Jenkins 2.503 and earlier, LTS 2.492.2 and earlier  
Denial of service vulnerability in bundled json-lib  
Multiple security vulnerabilities in Jenkins 2.499 and earlier, LTS 2.492.1 and earlier  
Multiple security vulnerabilities in Jenkins 2.478 and earlier, LTS 2.462.2 and earlier  
Fixes for all of these issues are available. Update Jenkins now.

Go to plugin manager Configure which of these warnings are shown

Set up agent Set up cloud Dismiss

https://www.jenkins.io/security/advisory/2024-08-07/


Practice Labs | IITK C... | -01874083C74b64b... | Learning Track | IITK... | Learning Track | IITK... | Learning Track | IITK... | Kevsa1995/Banking... | Docker Hub... | +

← → ↻ wai.nuavepro.com/guacamole/4/client/a50wMTg3NDAM2M3NGlNDlZgBjAGS1dmVsaWS?hostname=13.235.13.161&protocol=rdp&port=3389&username=naveuser&password=...

Applications Jenkins - Credentials [jenkins]... labuser@ip-172-31-37-8

Jenkins - Credentials [jenkins] — Mozilla Firefox

← → ↻ localhost:8080/manage/credentials/

 **Jenkins**

Search (CTRL+K) admin log out

Dashboard > Manage Jenkins > Credentials

T P Store ↓

Domain ID Name

Stores scoped to Jenkins

P Store ↓

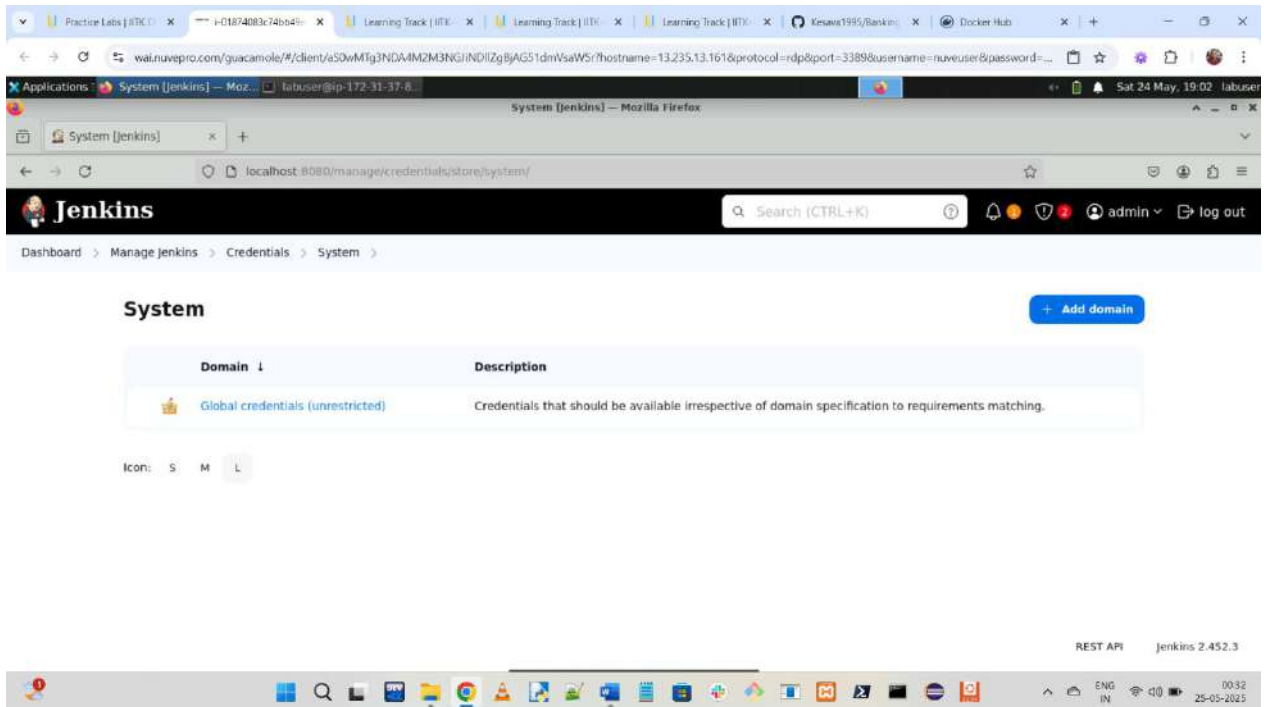
Domains

System (global)

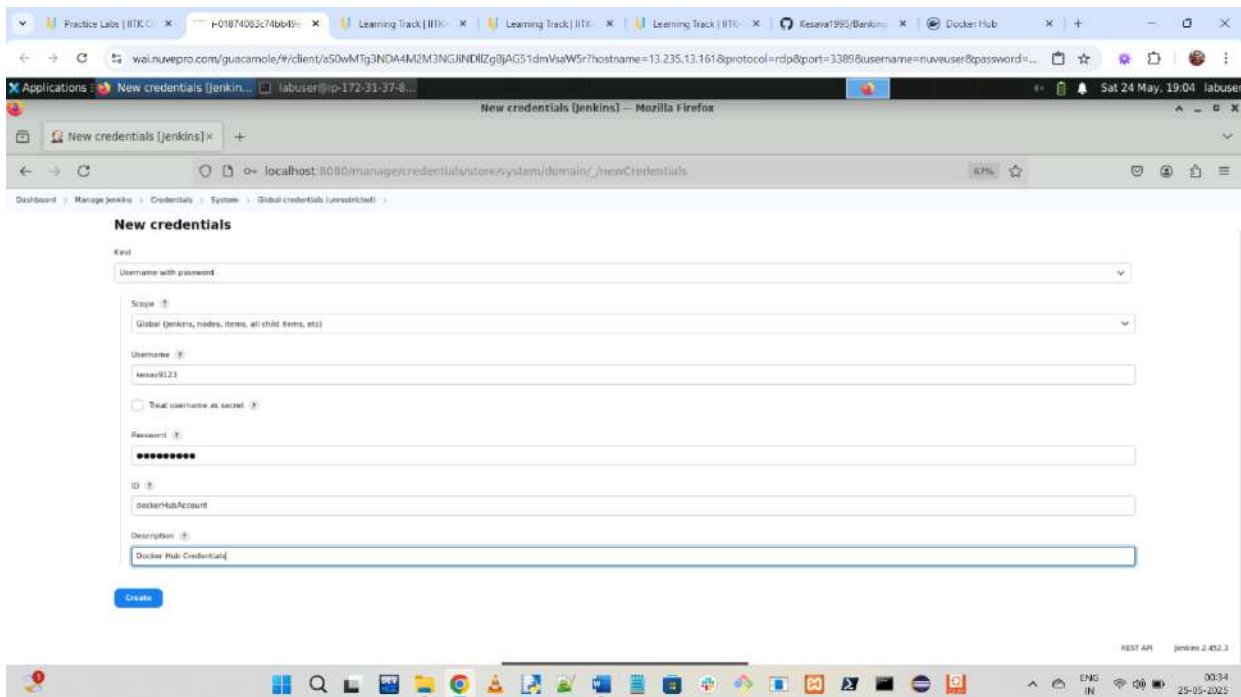
Icon: S M L

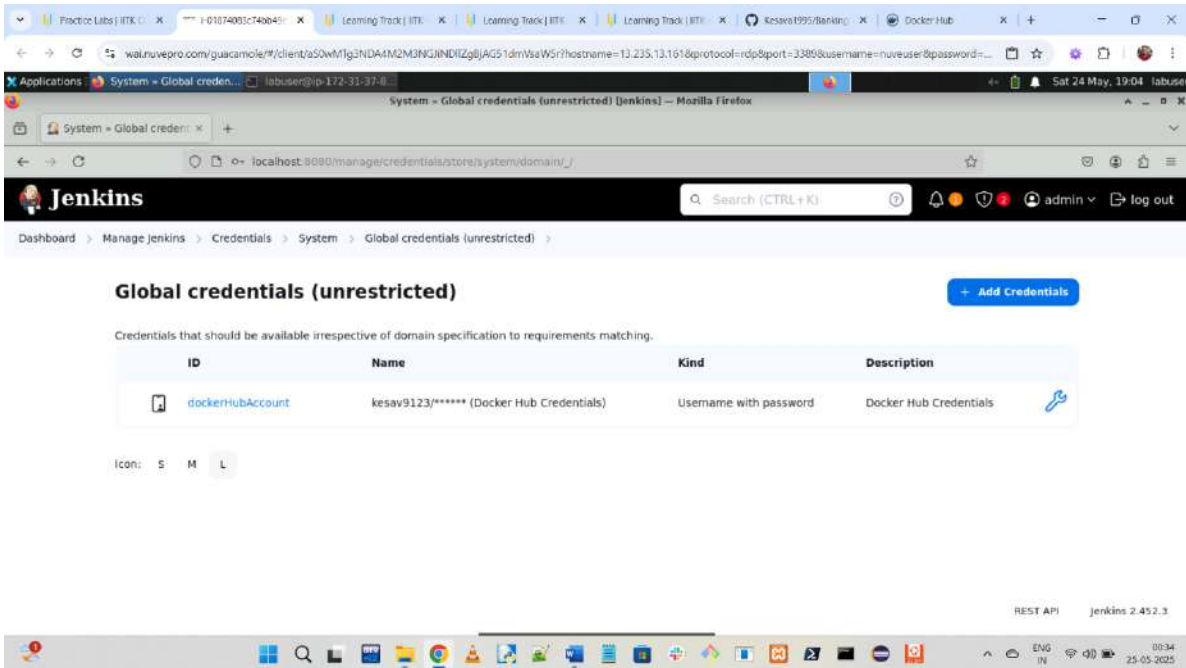
REST API Jenkins 2.452.3

00:32 25-05-2025



3.2 Click on **Add Credentials** to create new Docker Hub credentials with the details provided below and click on **Create**





**Note:** Credential id in below screenshot should match what you have mentioned in Jenkins pipeline

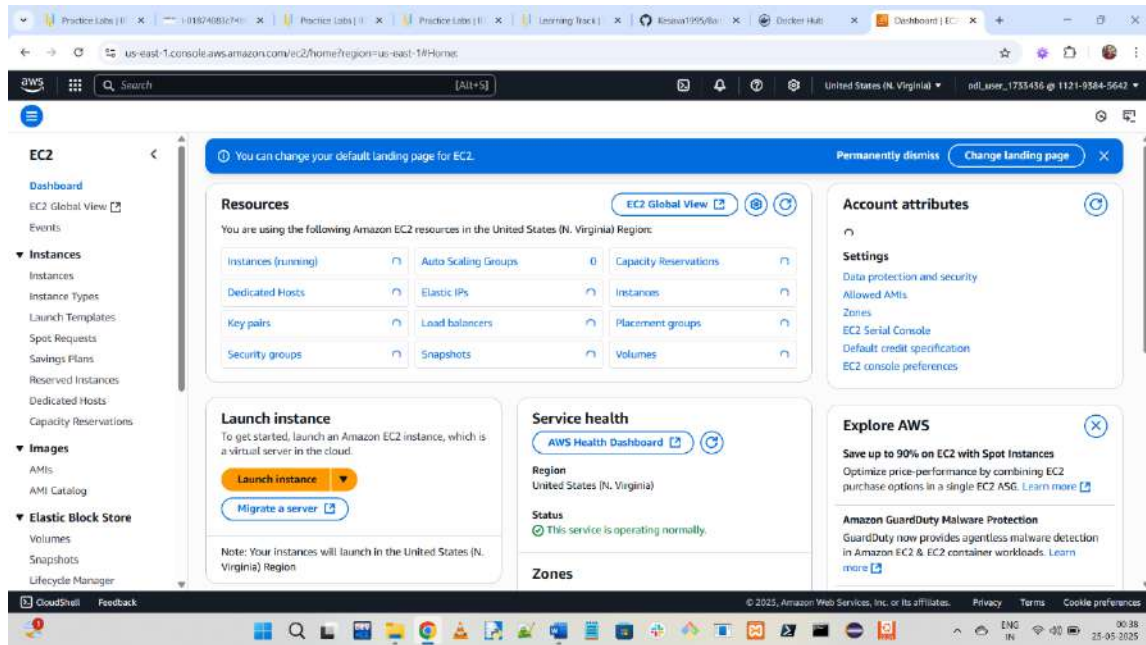
- 3.3 Once Docker image is built, we can integrate Docker image publish stage to publish Docker image to Docker hub registry

```
stage('Publishing Image to DockerHub'){
    echo 'Pushing the docker image to DockerHub'
    withCredentials([usernamePassword(credentialsId: 'dockerHubAccount', usernameVariable:
'dockerUser', passwordVariable: 'dockerPassword'))] {
        sh "docker login -u $dockerUser -p $dockerPassword"
        sh "docker push $dockerUser/$containerName:$tag"
        echo "Image push complete"
    }
}
```

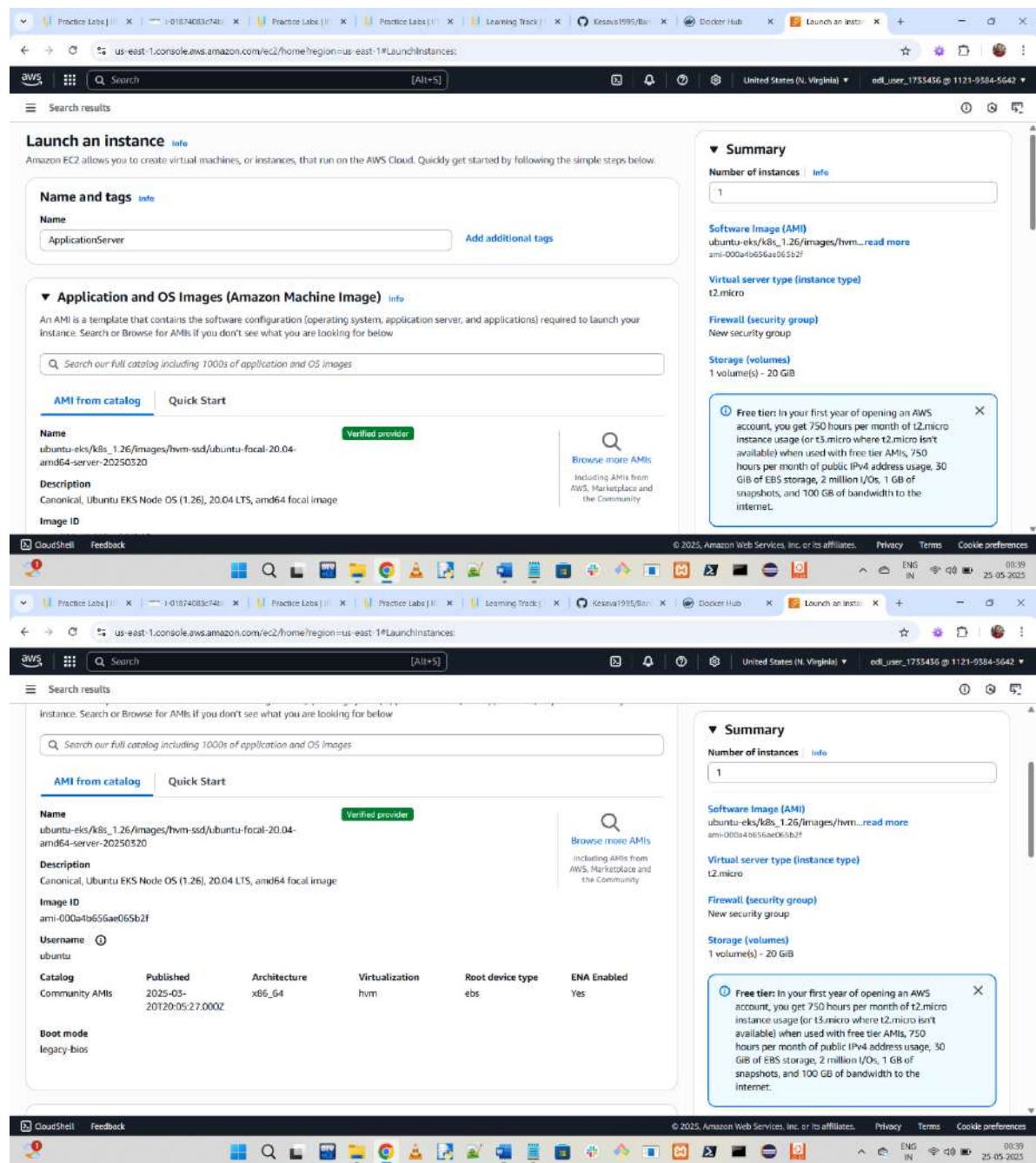
## Step 4: Configure Ansible on Simplilearn lab

**Note:** we need to configure Ansible Controller on Simplilearn Lab from where it would connect to both AWS and Azure VM

- 4.1 Navigate to AWS Console and EC2 service to launch VM using below steps

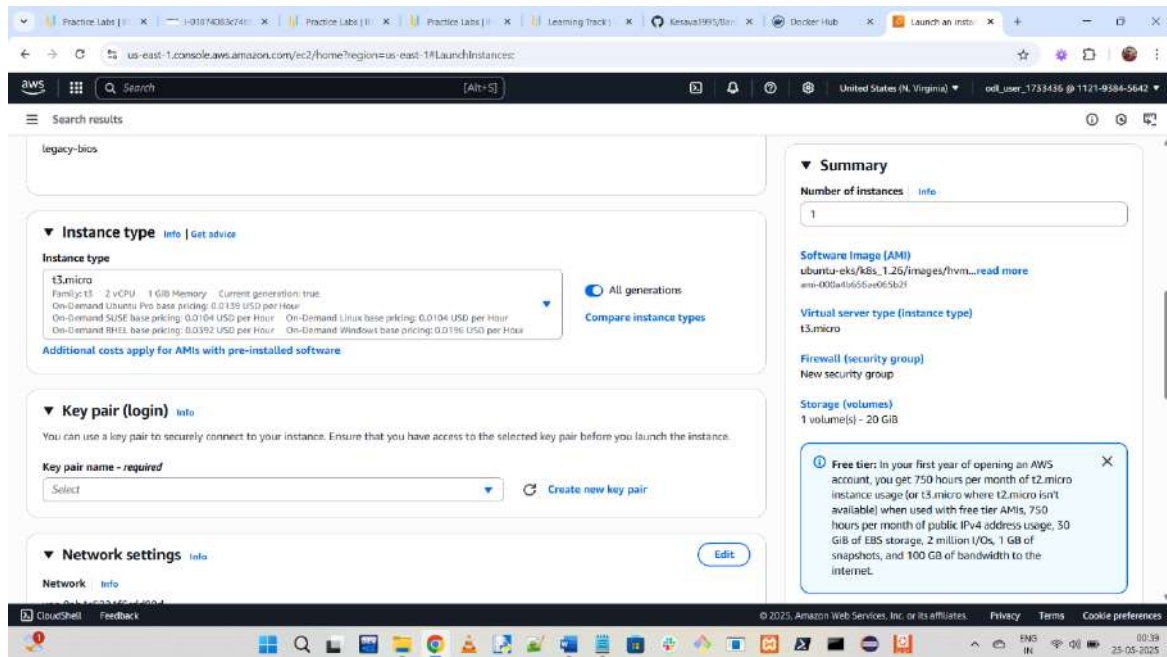


4.2 Provide name as **ApplicationServer** and Ubuntu 24.04 which will be used to launch VM



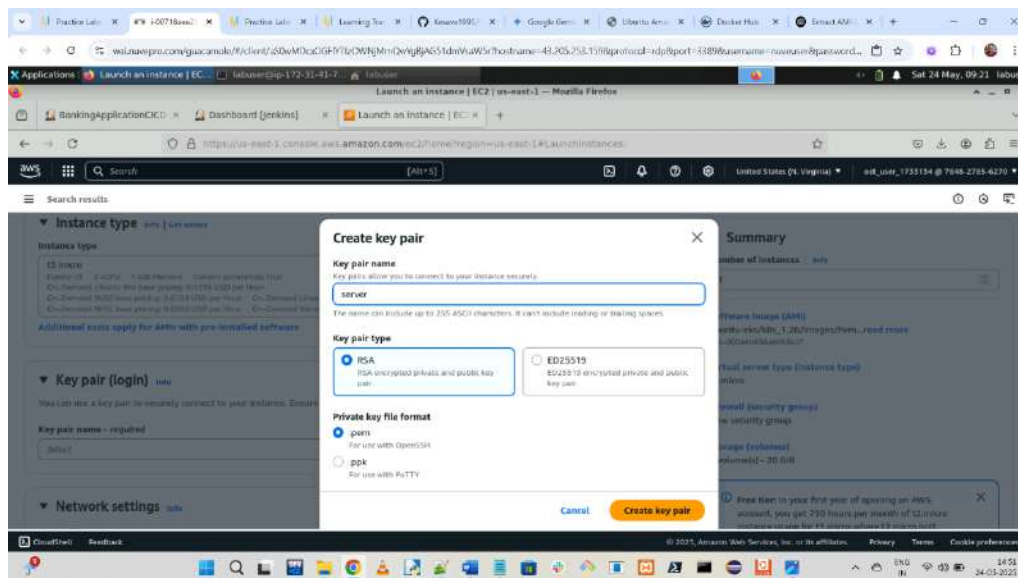
### 4.3 Select Instance type as **t3.micro**





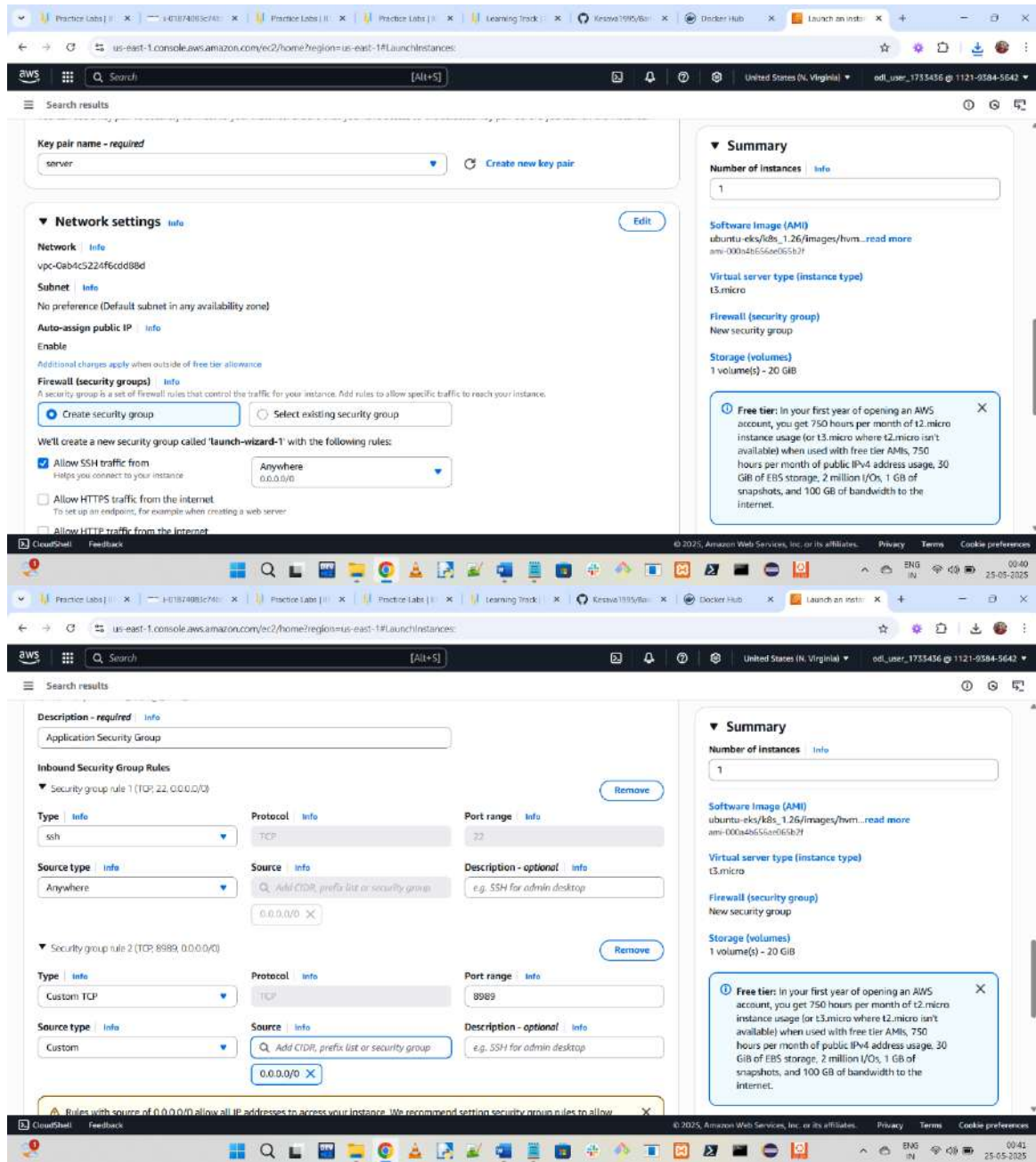
Next, we need to create key pair which will be used to perform SSH connectivity with server

4.4 Enter a key pair name as **server**, select the key pair type, choose the private key file format (.pem or .ppk), and click **Create key pair**. Save the private key securely for future instance connections.

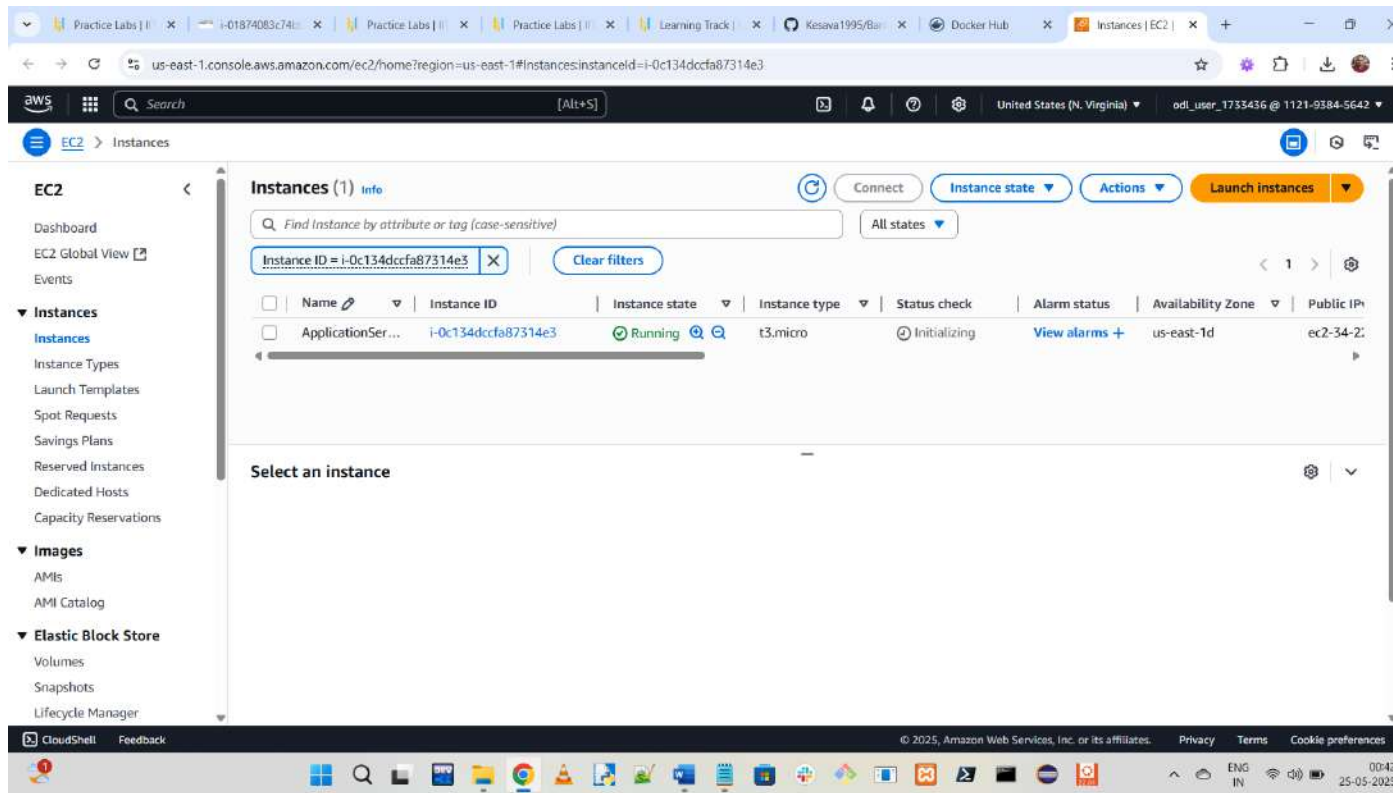


**Note:** This will download pem file locally, which we need to preserve since we would be using this key pair to establish connectivity from Ansible. Select this key pair once its created

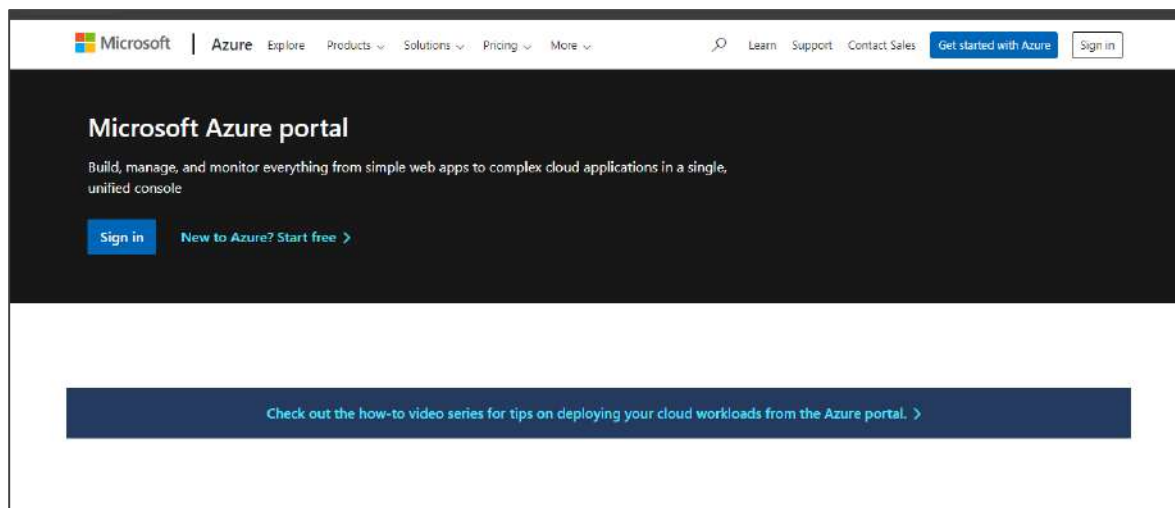
4.5 Click on Edit under network settings and add Application port in security group to allow that port.



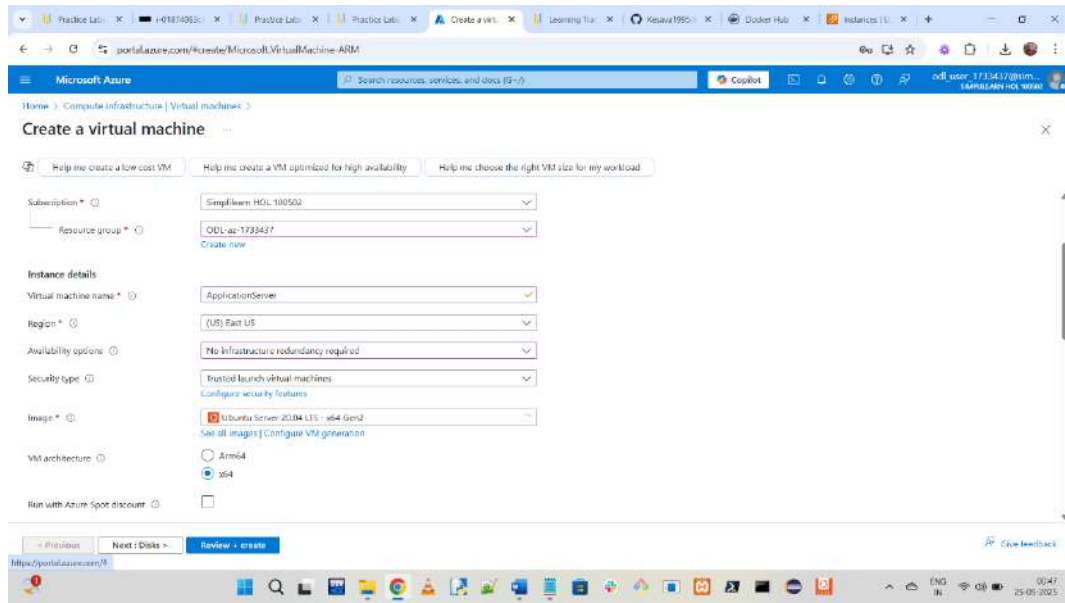
4.6 Create a security group named **sg\_application**, set the description to **Application Security Group**, and add inbound rules for **SSH** on port **22** from **Anywhere** and **Custom TCP** on port **8989** from **0.0.0.0/0**. Finally, click **Launch instance** to start the instance with these settings.



4.7 Navigate to Azure Console and launch a new VM which we will be connecting using Ansible Controller



4.8 Navigate to Virtual Machine service in Azure and proceed with launching new ubuntu 24.04 VM. Click on Create virtual machine and provide details as per below screenshot



#### 4.9 Select Ubuntu 24.04 image, VM size

The image consists of two screenshots of the Microsoft Azure portal, showing the process of creating a virtual machine (VM).

The top screenshot shows the "Select an image" page. The left sidebar lists "Other Items" (My images, Shared images, Community images, Direct Shared Images (PREVIEW)), "Marketplace" (All, Recently created, Private products), and "Categories" (Compute (466), Developer Tools (491), IT & Management Tools (283), DevOps (218)). The main area displays a grid of VM images. The first row includes "Ubuntu 20.04", "Ubuntu Minimal 20.04 LTS", "Ubuntu Server 20.04 LTS", "Ubuntu Server 20.04 LTS" (by Derek Coleman & Associates Co.), and "Ubuntu 20.04 LTS" (by Nitgral Inc.). The second row includes "Data Science Virtual Machine - Ubuntu 20.04", "Ubuntu Pro 20.04 LTS", "Ubuntu Server 20.04 LTS - x64 Gen 2", "Ubuntu Server 20.04 LTS - x64 Gen 1", "Ubuntu Server 20.04 LTS - Arm64", "PostgreSQL Ubuntu 20.04", and "Ubuntu 20.04 with NginX".

The bottom screenshot shows the "Create a virtual machine" page. The left sidebar lists "Help me create a low cost VM", "Help me create a VM optimized for high availability", and "Help me choose the right VM size for my workload". The main area shows the configuration steps: "Image" (Ubuntu Server 20.04 LTS - x64 Gen2), "VM architecture" (x64), "Run with Azure Spot discount" (unchecked), "Size" (Standard\_B2ms - 2 vcpus, 8 GiB memory), "Enable Hibernation" (unchecked), "Administrator account" (SSH public key), and "Authentication type" (SSH public key). The bottom navigation bar shows "Previous", "Next: Disks", and "Review + create".

4.10 For authentication provide username and password which we will be using with Anisble to authenticate



Microsoft Azure

Home > Compute infrastructure > Virtual machines >

## Create a virtual machine

Help me create a low cost VM | Help me create a VM optimized for high availability | Help me choose the right VM size for my workload

Hibernate does not currently support Trusted Launch and Confidential virtual machines for Linux images. [Learn more](#)

**Administrator account**

Authentication type: ☐ SSH public key ☒ Password

Username:

Password:

Confirm password:

**Inbound port rules**

Select which virtual machine network ports are accessible from the public internet. You can specify more limited or granular network access on the Networking tab.

Public inbound ports: ☐ None ☒ Allow selected ports

Select inbound ports:

This will allow all IP addresses to access your virtual machine. This is only

< Previous | Next > Disks | Review & create

Give feedback

Microsoft Azure

Home > Compute infrastructure > Virtual machines >

## Create a virtual machine

Help me create a low cost VM | Help me create a VM optimized for high availability | Help me choose the right VM size for my workload

**VM disk encryption**

Azure disk storage encryption automatically encrypts your data stored on Azure managed disks (OS and data disks) at rest by default when persisting it to the cloud.

Encryption at host: ☐

Encryption at host is not registered for the selected subscription. [Learn more](#)

**OS disk**

OS disk size:

OS disk type:

The selected VM size supports premium disks. We recommend Premium SSD for high IOPS workloads. Virtual machines with Premium SSD disks qualify for the 99.9% connectivity SLA.

Delete with VM: ☒

Key management:

Enable Ultra Disk compatibility: ☐

Ultra disk is supported in Availability Zones 1,2,3 for the selected VM size Standard\_B2ms.

< Previous | Next > Networking | Review & create

Give feedback

4.11 Click on Review and create to launch VM. Once VM is launched we must open **8989** ports from firewall rules to access application

The image consists of two screenshots of the Microsoft Azure portal interface.

The top screenshot shows the "Create a virtual machine" wizard, specifically the "Review + create" tab. The URL in the address bar is `portal.azure.com/#create/Microsoft.VirtualMachine-ARM`. The page displays a "Validation passed" message and three buttons: "Help me create a low cost VM", "Help me create a VM optimized for high availability", and "Help me choose the right VM size for my workload". Below these are tabs for "Basics", "Disks", "Networking", "Management", "Monitoring", "Advanced", "Tags", and "Review + create". The "Price" section shows "1 X Standard B2ms by Microsoft" with links to "Terms of use" and "Privacy policy". The "TERMS" section contains a legal disclaimer. The form fields are: "Name" (ODL\_User\_1733437), "Preferred e-mail address" (odl\_user\_1733437@simpilearn.hol100502.onmicrosoft.com), and "Preferred phone number" (4253580297). At the bottom are "Previous", "Next", and "Create" buttons, along with links for "Download a template for automation" and "Give feedback".

The bottom screenshot shows the "Overview" page of a completed deployment. The URL is `portal.azure.com/#view/HubsExtension/DeploymentDetailsBlade/~/overview/id/%2Fsubscriptions%2Fb655f245-a440-4ae5-b65a-7ea1bf155fc2%2FresourceGroups%2FODL.../deployments/0001-com-ubuntu-server-focal-2-20250525004708`. The page title is "CreateVm-canonical.0001-com-ubuntu-server-focal-2-20250525004708 | Overview". The left sidebar has tabs for "Overview", "Inputs", "Outputs", and "Template". The main content area shows a green checkmark and the text "Your deployment is complete". Deployment details include: "Deployment name: CreateVm-canonical.0001-com-ubuntu-server-f...", "Subscriptions: Simpilearn HOL 100502", "Resource group: ODL-az-1733437", "Start time: 25/5/2025, 12:48:30 am", and "Correlation ID: 38f69cc5-6f5c-479b-87b0-4b54a1d049b8". Under "Next steps", there are three recommended actions: "Setup auto-shutdown", "Monitor VM health, performance and network dependencies", and "Run a script inside the virtual machine". At the bottom are "Go to resource" and "Create another VM" buttons, and a "Give feedback" link.

The right sidebar contains three sections: "Cost Management" (Get notified to stay within your budget and prevent unexpected charges on your bill. Set up cost alerts >), "Microsoft Defender for Cloud" (Secure your apps and infrastructure. Go to Microsoft Defender for Cloud >), and "Free Microsoft tutorials" (Start learning today >). At the bottom, it says "Work with an expert" (Azure experts are service provider partners who can help manage your assets on Azure and be your first line of support. Find an Azure expert >).

4.12 Navigate to Virtual Machine service, select desired VM and modify networking configuration as per below screenshot



#### 4.13 Create inbound port rule for 8989 as shown in the screenshot below:

The first screenshot shows the 'Add inbound security rule' dialog box in the Azure portal. The rule is being added to the 'ApplicationServer-nsg' network security group. The configuration is as follows:

Field	Value
Source	Any
Source port ranges	*
Destination	Any
Service	Custom
Destination port ranges	8989
Protocol	Any
Action	Allow

The second screenshot shows the 'ApplicationServer' network settings page after the rule has been added. The 'Inbound port rules' table now includes the new rule for port 8989.

Priority	Name	Port	Protocol	Source	Destination	Action
800	SSH	22	TCP	Any	Any	Allow
65000	AllowAzureInbound	Any	Any	Any	Any	Allow
65001	AllowAzureLoadBalancerInbound	Any	Any	Any	Any	Allow
65500	DenyAllInbound	Any	Any	Any	Any	Deny

#### Step 5: Validate Deployment

**Note:** Once both AWS and Azure VM is launched, proceed with configuring Ansible inventory file to perform connectivity between Ansible and VMs

5.1 Create **inventory.yaml** in GitHub repository with below content

```
[all:children]
```

```
aws
```

```
azure
```

```
[aws]
```

```
34.229.191.94
```

```
[aws:vars]
```

```
ansible_ssh_user=ubuntu
```

```
ansible_ssh_private_key_file="{{ key_pair_path }}"
```

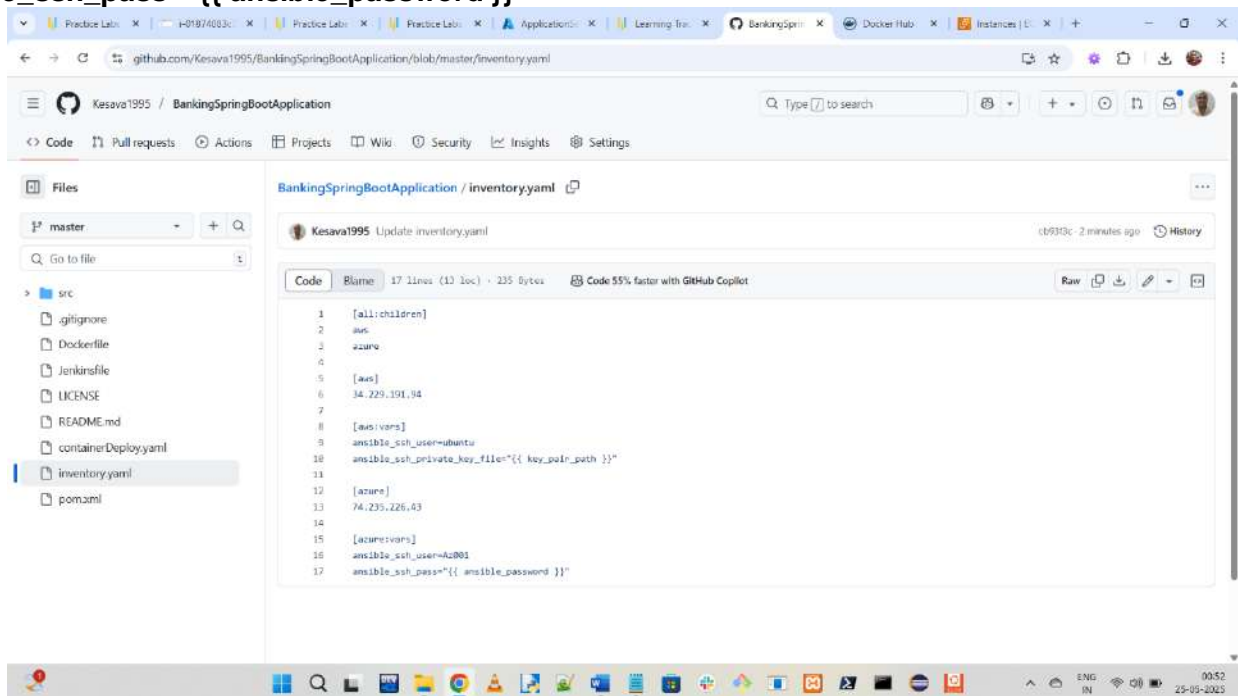
```
[azure]
```

```
74.235.226.43
```

```
[azure:vars]
```

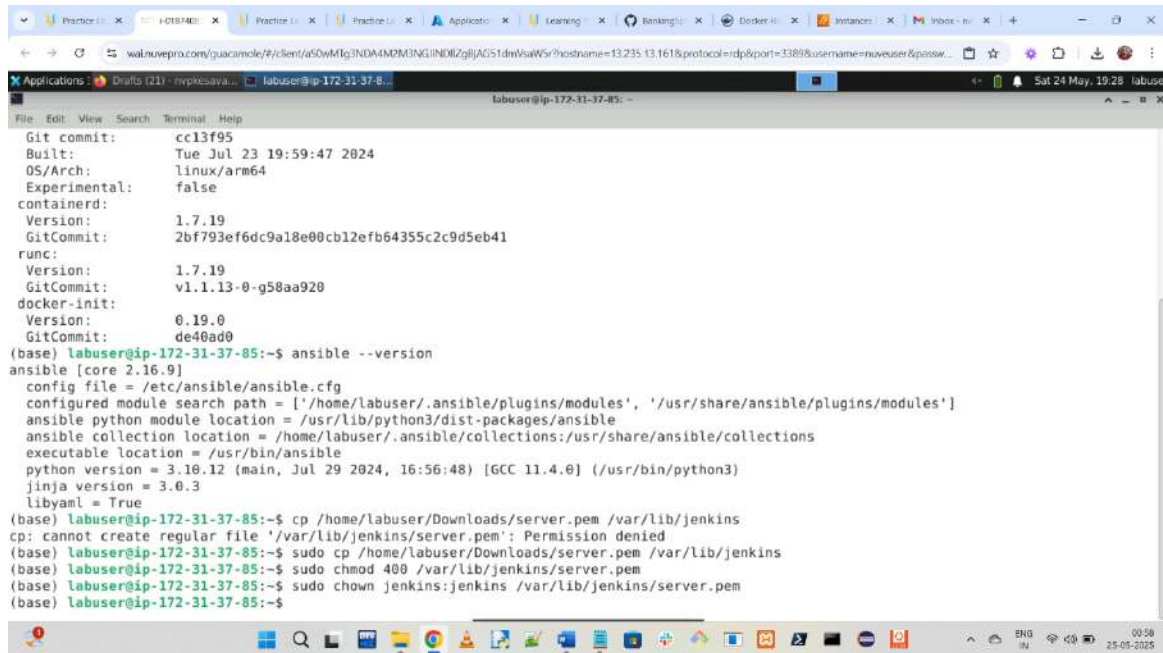
```
ansible_ssh_user=Az001
```

```
ansible_ssh_pass="{{ ansible_password }}"
```



We need to create key pair on DevOps lab which was downloaded while launching EC2 instance

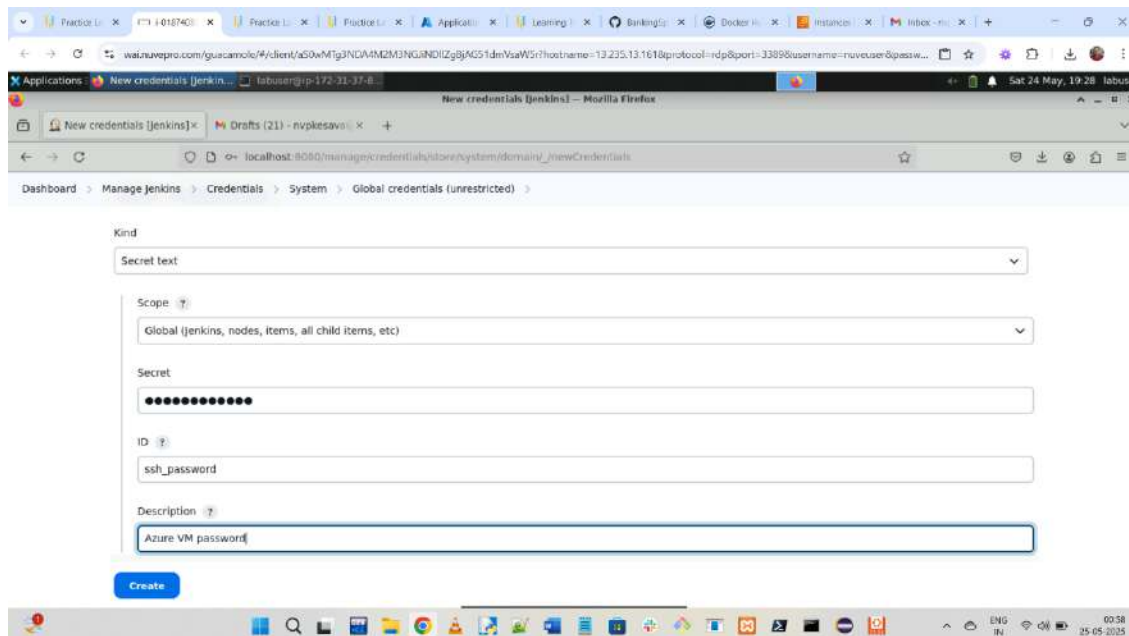
5.2 Navigate to **/var/lib/jenkins**, open the private key file **server.pem** using **vi**, verify its permissions with **ls -lart**, update the permissions to read-only using **chmod 400 server.pem**, and confirm the change with **ls -lart**, ensuring the file is secure with read-only access for the Jenkins user



The terminal window shows the following output:

```
Git commit: cc13f95
Built: Tue Jul 23 19:59:47 2024
OS/Arch: linux/arm64
Experimental: false
containerd:
  Version: 1.7.19
  GitCommit: 2bf793ef6dc9a18e00cb12efb64355c2c9d5eb41
runc:
  Version: 1.7.19
  GitCommit: v1.1.13-0-g58aa920
docker-init:
  Version: 0.19.0
  GitCommit: de40ad0
(base) labuser@ip-172-31-37-85:~$ ansible --version
ansible [core 2.16.9]
  config file = /etc/ansible/ansible.cfg
  configured module search path = ['/home/labuser/.ansible/plugins/modules', '/usr/share/ansible/plugins/modules']
  ansible python module location = /usr/lib/python3/dist-packages/ansible
  ansible collection location = /home/labuser/.ansible/collections:/usr/share/ansible/collections
  executable location = /usr/bin/ansible
  python version = 3.10.12 (main, Jul 29 2024, 16:56:48) [GCC 11.4.0] (/usr/bin/python3)
  jinja version = 3.0.3
  libyaml = True
(base) labuser@ip-172-31-37-85:~$ cp /home/labuser/Downloads/server.pem /var/lib/jenkins
cp: cannot create regular file '/var/lib/jenkins/server.pem': Permission denied
(base) labuser@ip-172-31-37-85:~$ sudo cp /home/labuser/Downloads/server.pem /var/lib/jenkins
(base) labuser@ip-172-31-37-85:~$ sudo chmod 400 /var/lib/jenkins/server.pem
(base) labuser@ip-172-31-37-85:~$ sudo chown jenkins:jenkins /var/lib/jenkins/server.pem
(base) labuser@ip-172-31-37-85:~$
```

We need to create a Jenkins credential with Azure password for VM connectivity



The screenshot shows the 'New credentials' form in Jenkins. The fields are filled as follows:

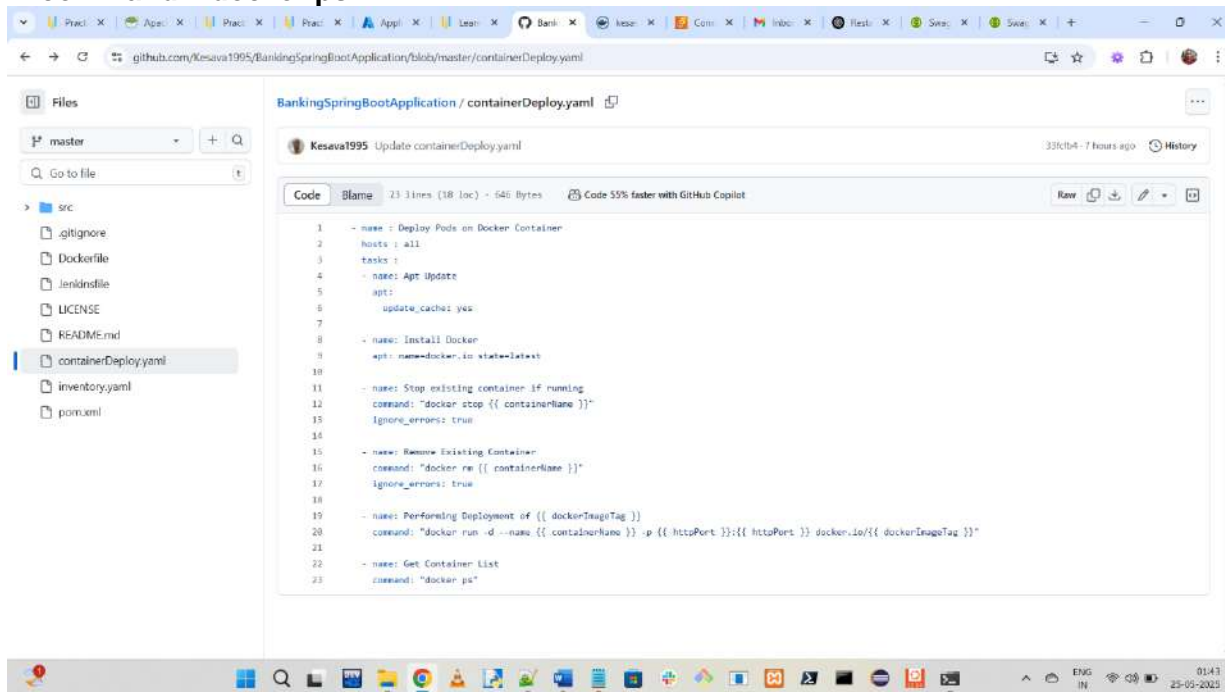
- Kind: Secret text
- Scope: Global (jenkins, nodes, items, all child items, etc)
- Secret: (masked with dots)
- ID: ssh\_password
- Description: Azure VM password

The 'Create' button is visible at the bottom left of the form.

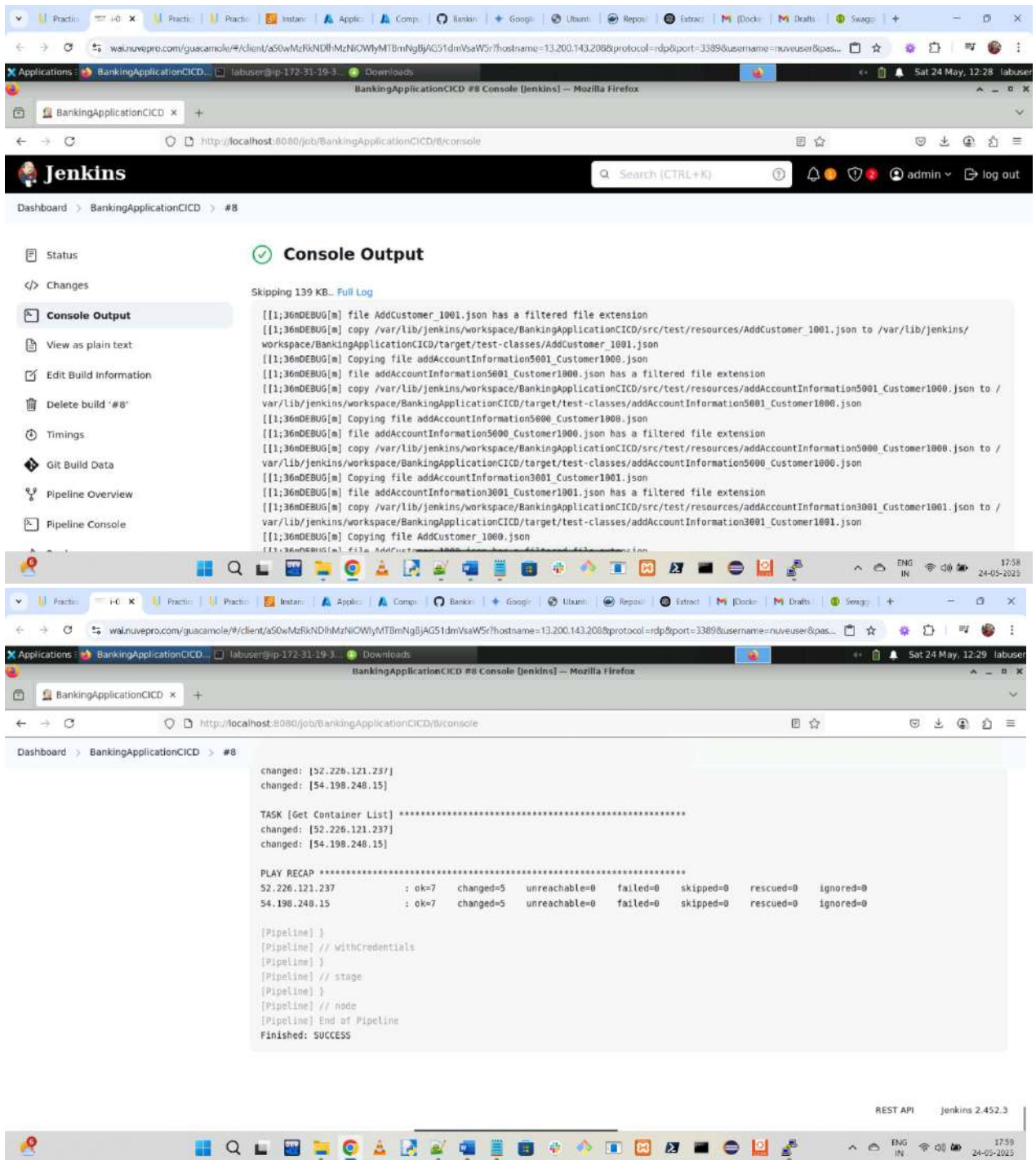
5.3 Now, we need to create Ansible Playbook file as per below content

- name : Deploy Pods on Docker Container
- hosts : all
- tasks :
- name: Apt Update
- apt:
- update\_cache: yes

- name: Install Docker  
apt: name=docker.io state=latest
- name: Stop existing container if running  
command: "docker stop {{ containerName }}"  
ignore\_errors: true
- name: Remove Existing Container  
command: "docker rm {{ containerName }}"  
ignore\_errors: true
- name: Performing Deployment of {{ dockerImageTag }}  
command: "docker run -d --name {{ containerName }} -p {{ httpPort }}:{{ httpPort }} docker.io/{{ dockerImageTag }}"
- name: Get Container List  
command: "docker ps"



5.4 Navigate to Jenkins job created and click on Build now to start running build for Jenkins job created



The screenshot displays the Jenkins web interface for a pipeline named 'BankingApplicationCICD'. The 'Console Output' tab is selected, showing a series of debug messages from a Java application. These messages indicate the copying of various JSON files to the Jenkins workspace, such as 'AddCustomer\_1001.json' and 'addAccountInformation5001\_Customer1000.json'. The output is truncated with 'Skipping 139 KB.. Full Log'.

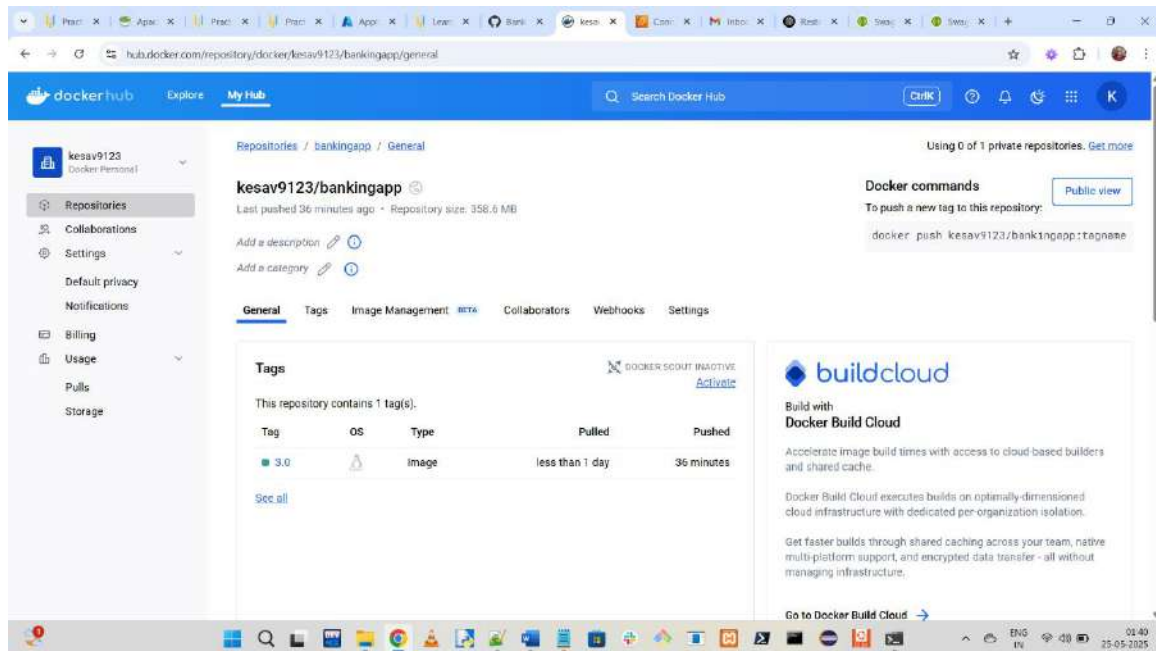
Below the console output, the 'Pipeline Recap' section provides a summary of the pipeline's execution. It shows the IP addresses of the nodes involved: 52.226.121.237 and 54.198.248.15. A table summarizes the status of each node:

Node	ok	changed	unreachable	failed	skipped	rescued	ignored
52.226.121.237	7	5	0	0	0	0	0
54.198.248.15	7	5	0	0	0	0	0

The pipeline concludes with the message 'Finished: SUCCESS'. The bottom of the image shows the Jenkins REST API endpoint and version (2.452.3).

5.5 Navigate to Docker hub to validate if Docker image gets published or not

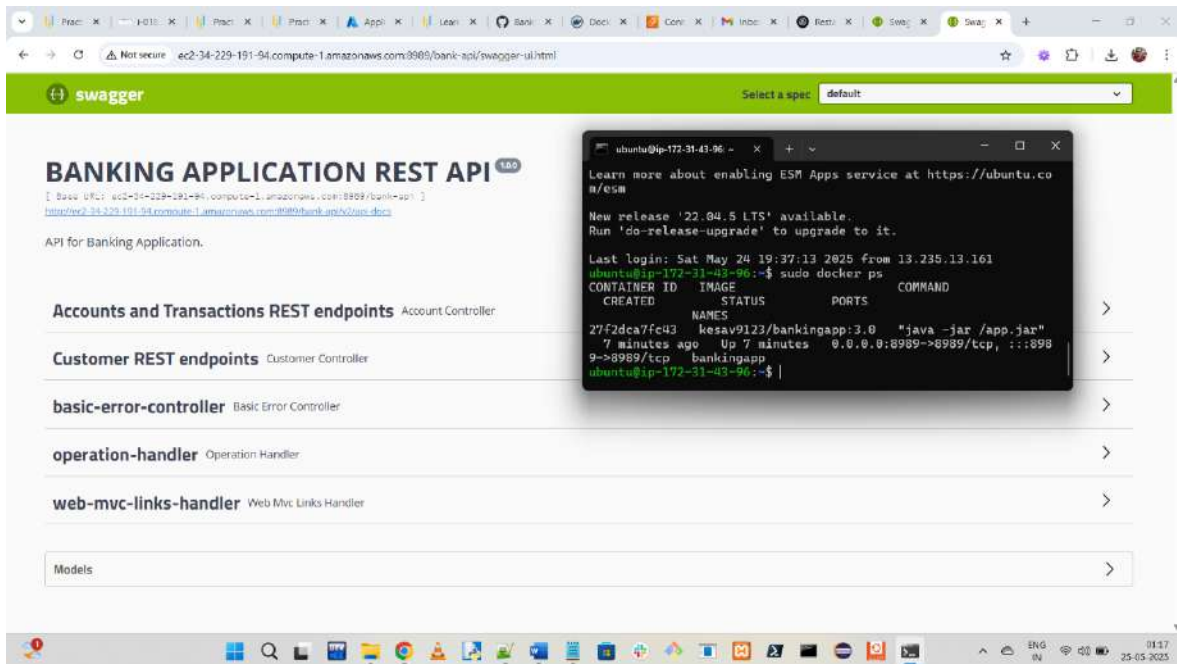




**Note:** First time build may take more time than expected since lot of maven dependencies will be downloaded and cache locally first time. Subsequent executions will not take much time later.

5.6 Once Application container is deployed on Docker host you can manually validate if containers are really running or not, connect to AWS VM and running below command to validate

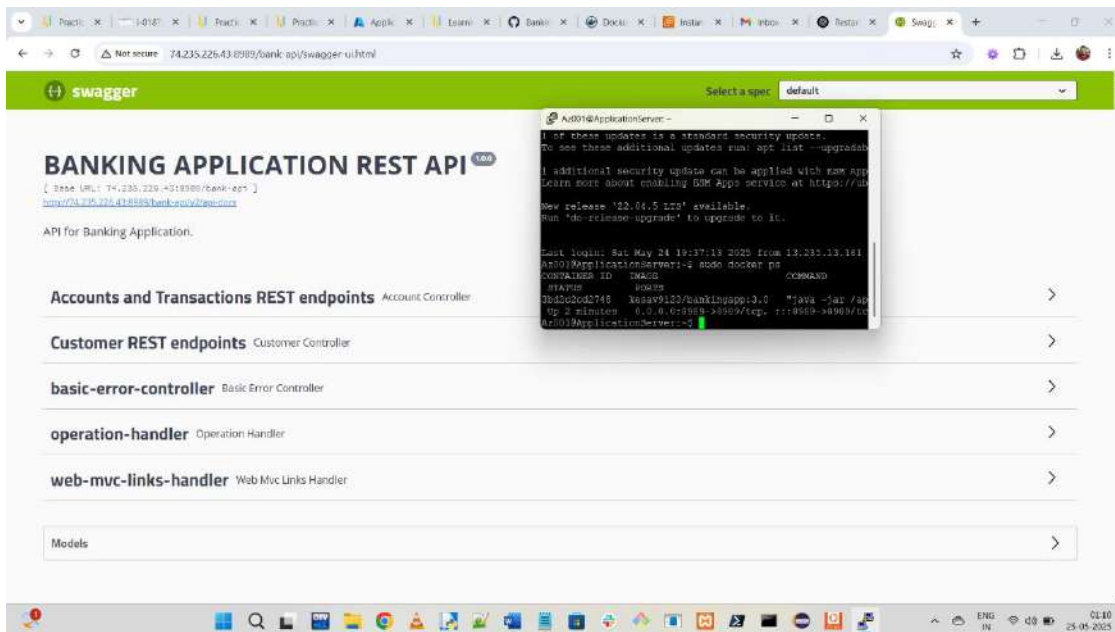
**ec2-34-229-191-94.compute-1.amazonaws.com:8989/bank-api/swagger-ui.html**



5.7 Navigate to Azure VM and validate in case Docker container is deployed there or not, use system default browser to access application deployed on Docker Host. We will be getting Swagger Interface using which you can validate your Rest APIs.

**http://<SERVER-IP>:8989/bank-api/swagger-ui.html**

**http:// 74.235.226.43:8989/bank-api/swagger-ui.html**





By following the above steps, you have successfully implemented a multi-cloud deployment pipeline using Jenkins, Ansible, and Docker, enabling seamless integration and automation for the Java application across AWS and Azure environments. This setup ensures efficient CI/CD processes, robust containerized deployments, and enhanced application availability, validated through Swagger API testing.