Application DevOps Pipeline with Jenkins CI/CD and AKS Containerization

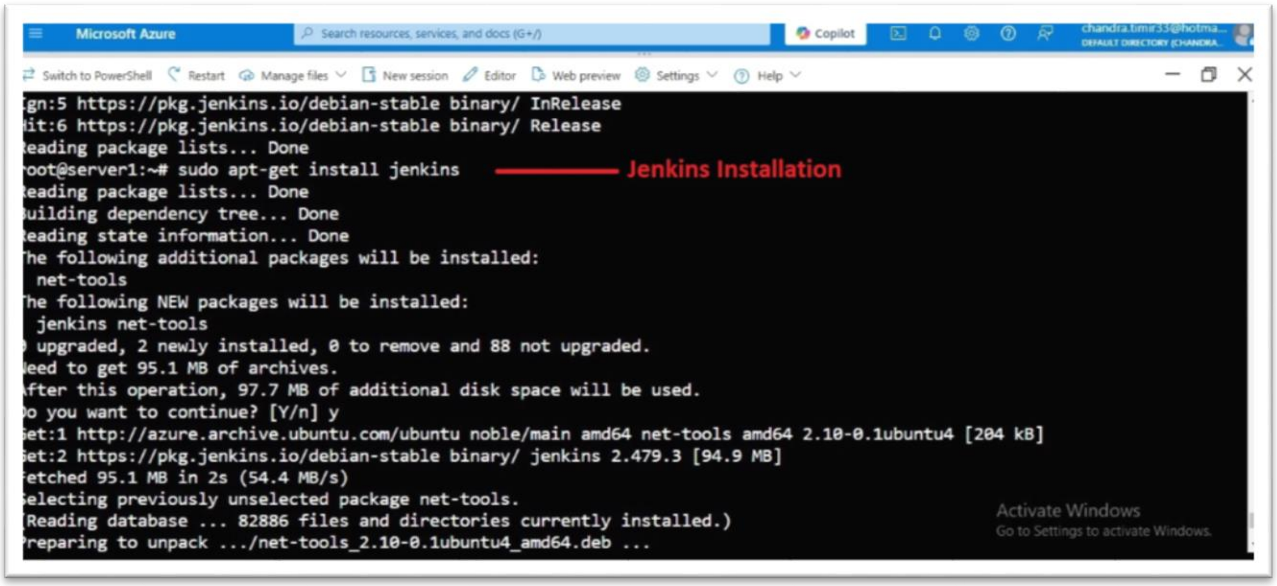
---

Table of Contents

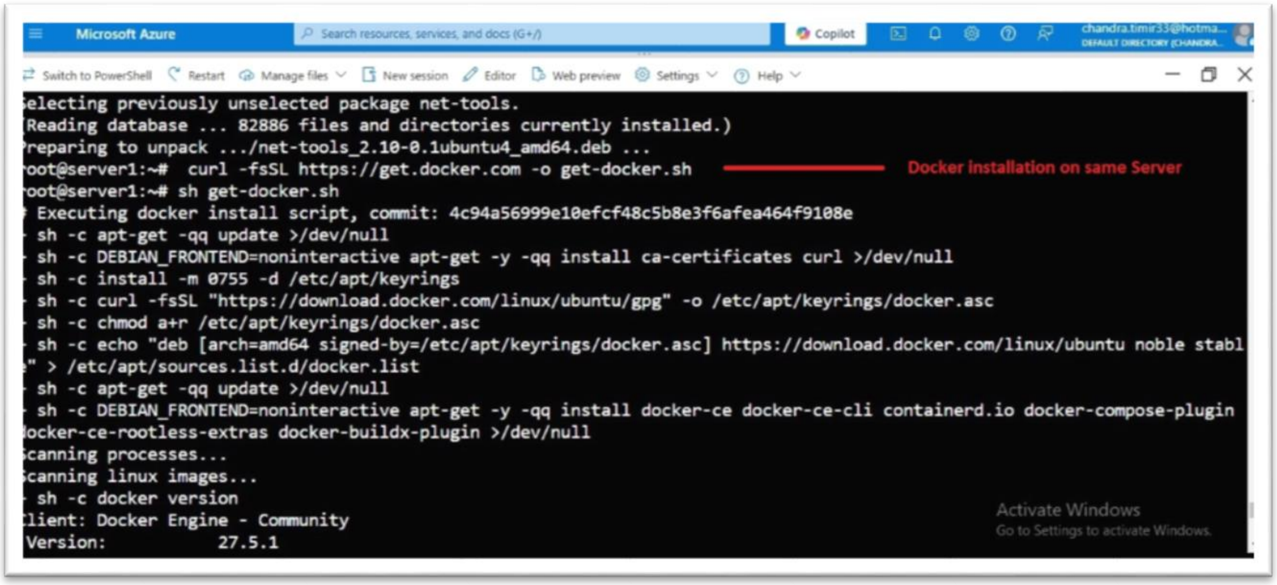
- ○ High-level architecture of the CI/CD pipeline
- ○ Stages of the pipeline:
    - ■ Code commit and version control with GitHub/GitLab
    - ■ Build and testing stage using Jenkins
    - ■ Containerization with Docker
    - ■ Deployment to AKS
- ○ Detailed workflow of each stage with diagrams
- ○ Monitoring and logging setup for pipeline feedback

Pipeline Implementation

- ○ Step-by-step setup and configuration:
    - ■ Install Jenkins and Docker on the same server

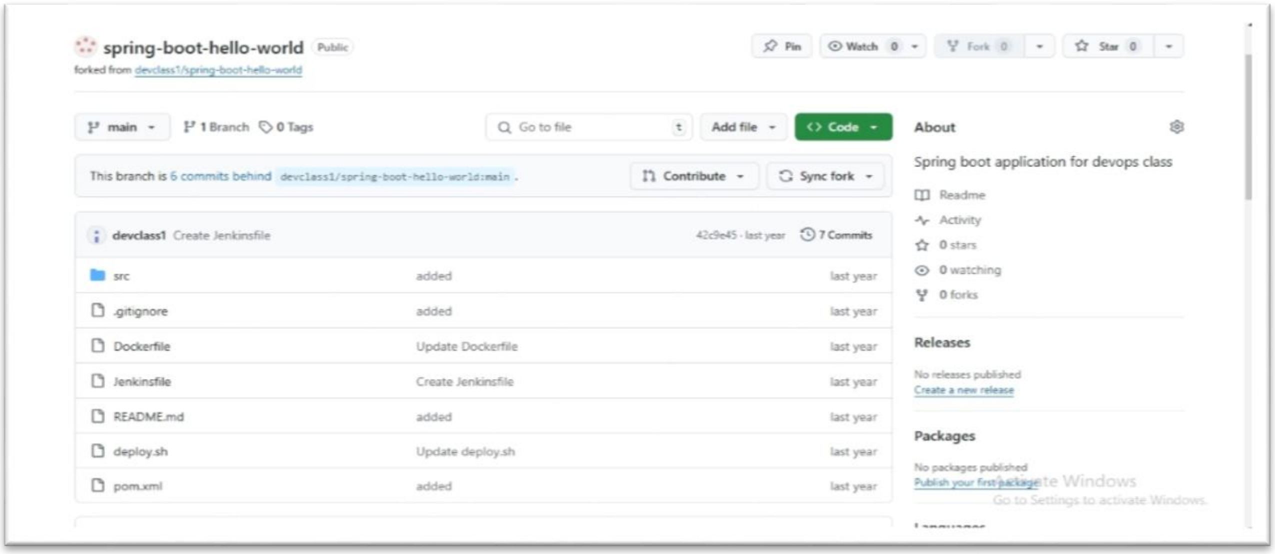


```
gn:5 https://pkg.jenkins.io/debian-stable binary/ InRelease
it:6 https://pkg.jenkins.io/debian-stable binary/ Release
eading package lists... Done
oot@server1:~# sudo apt-get install jenkins
eading package lists... Done
uilding dependency tree... Done
eading state information... Done
he following additional packages will be installed:
 net-tools
he following NEW packages will be installed:
 jenkins net-tools
 upgraded, 2 newly installed, 0 to remove and 88 not upgraded.
eed to get 95.1 MB of archives.
fter this operation, 97.7 MB of additional disk space will be used.
o you want to continue? [Y/n] y
et:1 http://azure.archive.ubuntu.com/ubuntu noble/main amd64 net-tools amd64 2.10-0.1ubuntu4 [204 kB]
et:2 https://pkg.jenkins.io/debian-stable binary/ jenkins 2.479.3 [94.9 MB]
etched 95.1 MB in 2s (54.4 MB/s)
electing previously unselected package net-tools.
(Reading database ... 82886 files and directories currently installed.)
reparing to unpack .../net-tools_2.10-0.1ubuntu4_amd64.deb ...
```



```
electing previously unselected package net-tools.
(Reading database ... 82886 files and directories currently installed.)
reparing to unpack .../net-tools_2.10-0.1ubuntu4_amd64.deb ...
oot@server1:~#  curl -fsSL https://get.docker.com -o get-docker.sh
oot@server1:~# sh get-docker.sh
 Executing docker install script, commit: 4c94a56999e10efcf48c5b8e3f6afea464f9108e
 sh -c apt-get -qq update >/dev/null
 sh -c DEBIAN_FRONTEND=noninteractive apt-get -y -qq install ca-certificates curl >/dev/null
 sh -c install -m 0755 -d /etc/apt/keyrings
 sh -c curl -fsSL "https://download.docker.com/linux/ubuntu/gpg" -o /etc/apt/keyrings/docker.asc
 sh -c chmod a+r /etc/apt/keyrings/docker.asc
 sh -c echo "deb [arch=amd64 signed-by=/etc/apt/keyrings/docker.asc] https://download.docker.com/linux/ubuntu noble stabl
" > /etc/apt/sources.list.d/docker.list
 sh -c apt-get -qq update >/dev/null
 sh -c DEBIAN_FRONTEND=noninteractive apt-get -y -qq install docker-ce docker-ce-cli containerd.io docker-compose-plugin
ocker-ce-rootless-extras docker-buildx-plugin >/dev/null
canning processes...
canning linux images...
 sh -c docker version
lient: Docker Engine - Community
Version:          27.5.1
```

- Git Repository name as a **spring-boot-hello-world** with Dockerfile


spring-boot-hello-world Public
forked from devclass1/spring-boot-hello-world

main · 1 Branch · 0 Tags · Go to file · Add file · Code

This branch is 6 commits behind devclass1/spring-boot-hello-world:main .

devclass1 Create Jenkinsfile · 42c9e45 · last year · 7 Commits

src	added	last year
.gitignore	added	last year
Dockerfile	Update Dockerfile	last year
Jenkinsfile	Create Jenkinsfile	last year
README.md	added	last year
deploy.sh	Update deploy.sh	last year
pom.xml	added	last year

About

Spring boot application for devops class

Readme · Activity · 0 stars · 0 watching · 0 forks

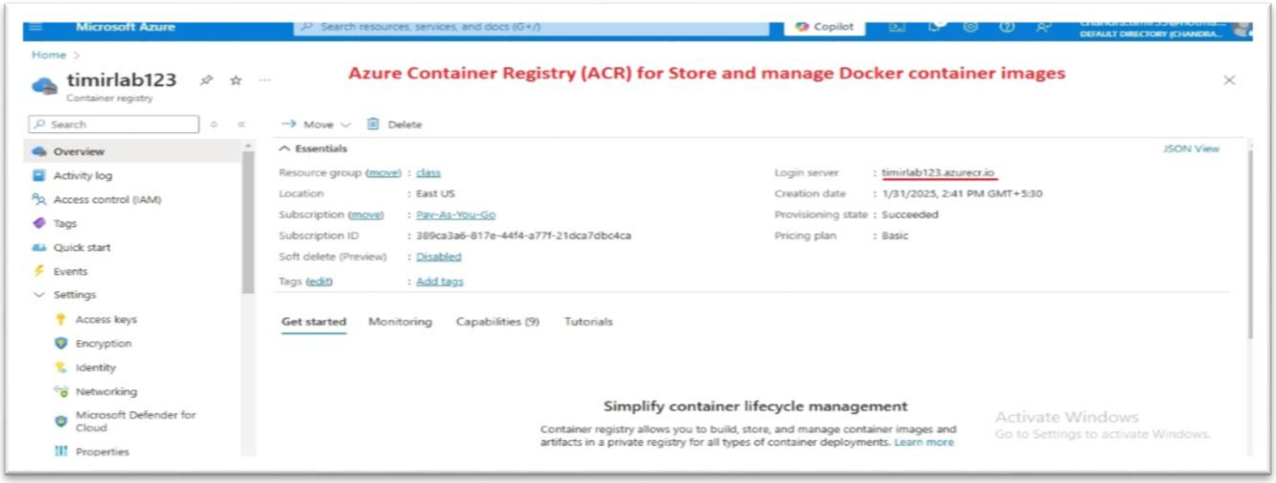
Releases

No releases published
Create a new release

Packages

No packages published
Publish your first package

- **Azure Container Registry (ACR) for Store and manage Docker container images**


Azure Container Registry (ACR) for Store and manage Docker container images

timirlab123
Container registry

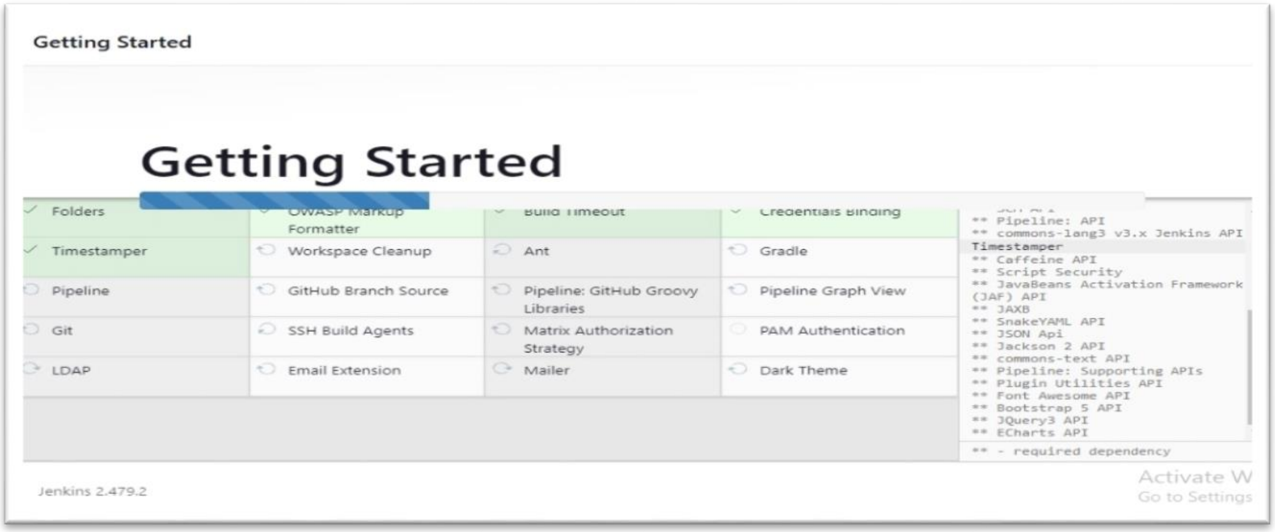
Essentials

Resource group (move)	: class	Login server	: timirlab123.azurecr.io
Location	: East US	Creation date	: 1/31/2025, 2:41 PM GMT+5:30
Subscription (move)	: Pay-As-You-Go	Provisioning state	: Succeeded
Subscription ID	: 389ca3a6-817e-44f4-a77f-21dca7dbc4ca	Pricing plan	: Basic
Soft delete (Preview)	: Disabled		
Tags (edit)	: Add tags		

Simplify container lifecycle management

Container registry allows you to build, store, and manage container images and artifacts in a private registry for all types of container deployments. Learn more

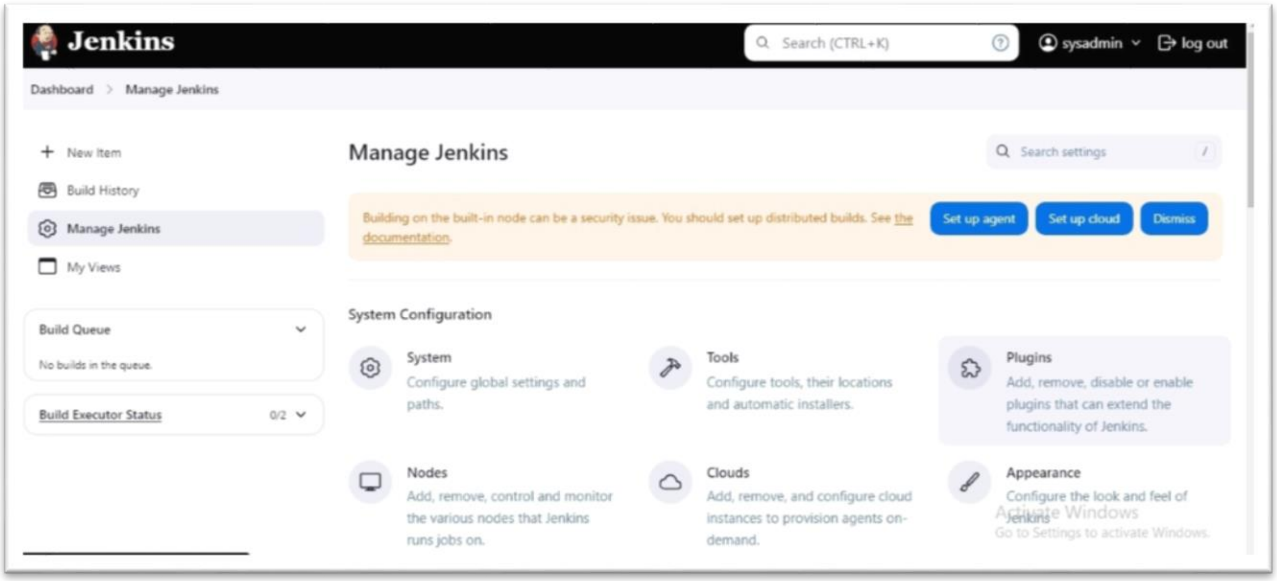
- Jenkins setup with relevant maven plugins & tools


Getting Started

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

```
** Pipeline: API
** commons-lang3 v3.x Jenkins API
Timestamper
** Caffeine API
** Script Security
** JavaBeans Activation Framework
(JAF) API
** JAXB
** SnakeYAML API
** JSON Api
** Jackson 2 API
** commons-text API
** Pipeline: Supporting APIs
** Plugin Utilities API
** Font Awesome API
** Bootstrap 5 API
** JQuery3 API
** ECharts API
** - required dependency
```

Jenkins 2.479.2


Jenkins | Search (CTRL+K) | sysadmin | log out

Dashboard > Manage Jenkins

New Item
Build History
Manage Jenkins
My Views

Build Queue
No builds in the queue.

Build Executor Status 0/2

Manage Jenkins

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

System Configuration

System
Configure global settings and paths.

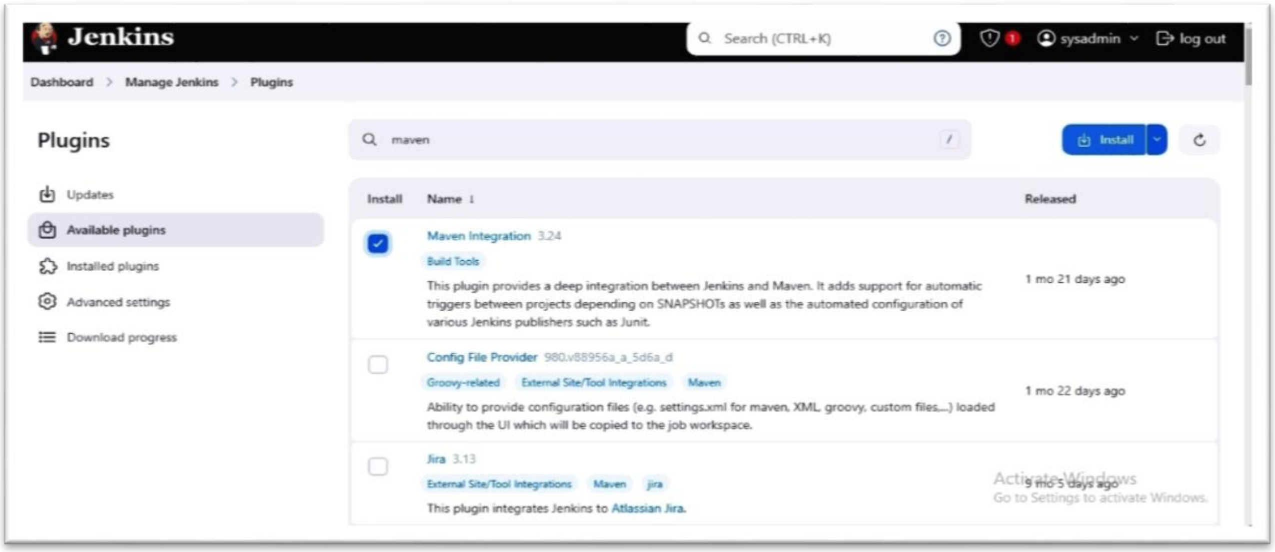
Tools
Configure tools, their locations and automatic installers.

Plugins
Add, remove, disable or enable plugins that can extend the functionality of Jenkins.

Nodes
Add, remove, control and monitor the various nodes that Jenkins runs jobs on.

Clouds
Add, remove, and configure cloud instances to provision agents on-demand.

Appearance
Configure the look and feel of Jenkins


Jenkins | Search (CTRL+K) | sysadmin | log out

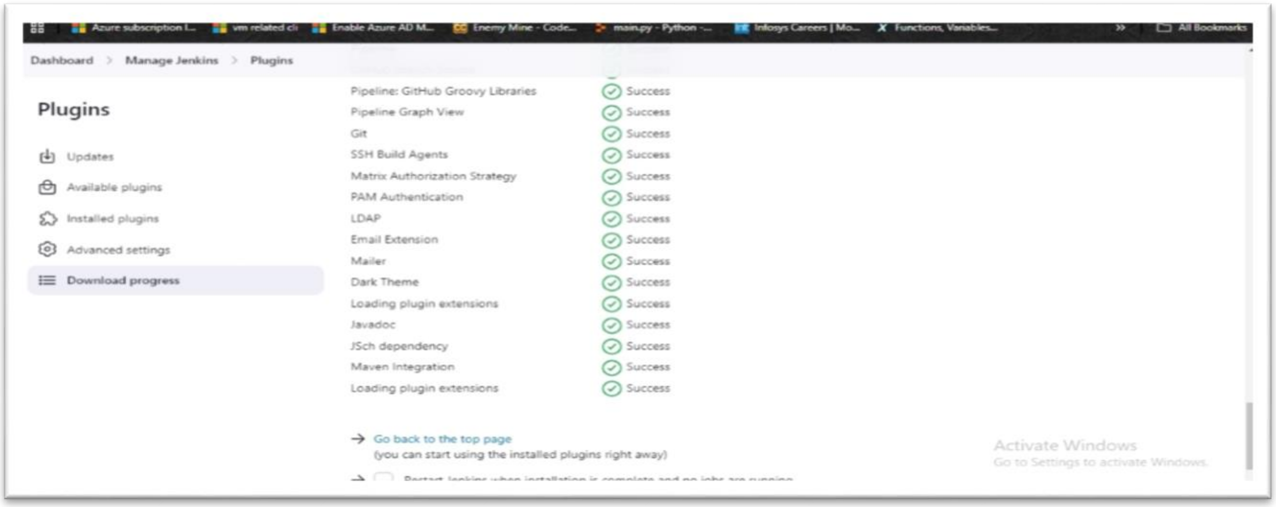
Dashboard > Manage Jenkins > Plugins

Plugins

Updates
Available plugins
Installed plugins
Advanced settings
Download progress

Search: maven | Install

Install	Name ↓	Released
✓	Maven Integration 3.24 — Build Tools — This plugin provides a deep integration between Jenkins and Maven. It adds support for automatic triggers between projects depending on SNAPSHOTs as well as the automated configuration of various Jenkins publishers such as Junit.	1 mo 21 days ago
☐	Config File Provider 980.v88956a_a_5d6a_d — Groovy-related, External Site/Tool Integrations, Maven — Ability to provide configuration files (e.g. settings.xml for maven, XML, groovy, custom files,...) loaded through the UI which will be copied to the job workspace.	1 mo 22 days ago
☐	Jira 3.13 — External Site/Tool Integrations, Maven, jira — This plugin integrates Jenkins to Atlassian Jira.	9 mo 5 days ago

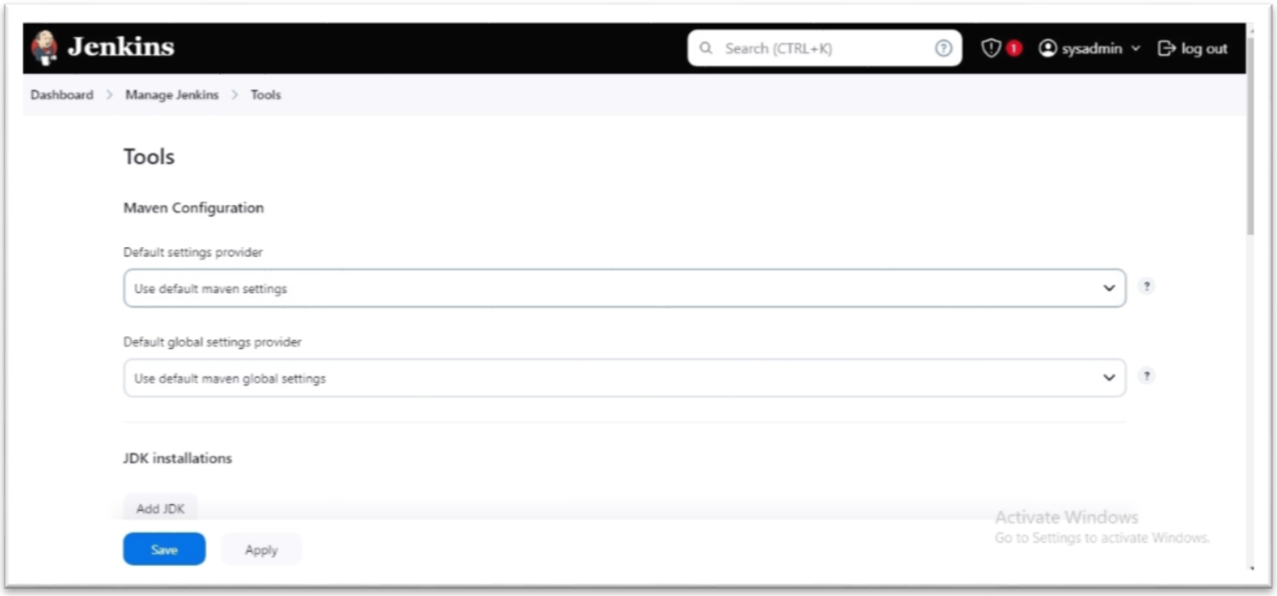

Dashboard > Manage Jenkins > Plugins

Plugins

Updates
Available plugins
Installed plugins
Advanced settings
Download progress

Pipeline: GitHub Groovy Libraries	✓ Success
Pipeline Graph View	✓ Success
Git	✓ Success
SSH Build Agents	✓ Success
Matrix Authorization Strategy	✓ Success
PAM Authentication	✓ Success
LDAP	✓ Success
Email Extension	✓ Success
Mailer	✓ Success
Dark Theme	✓ Success
Loading plugin extensions	✓ Success
Javadoc	✓ Success
JSch dependency	✓ Success
Maven Integration	✓ Success
Loading plugin extensions	✓ Success

→ Go back to the top page
(you can start using the installed plugins right away)


Jenkins
Search (CTRL+K)
sysadmin
log out
Dashboard > Manage Jenkins > Tools

Tools

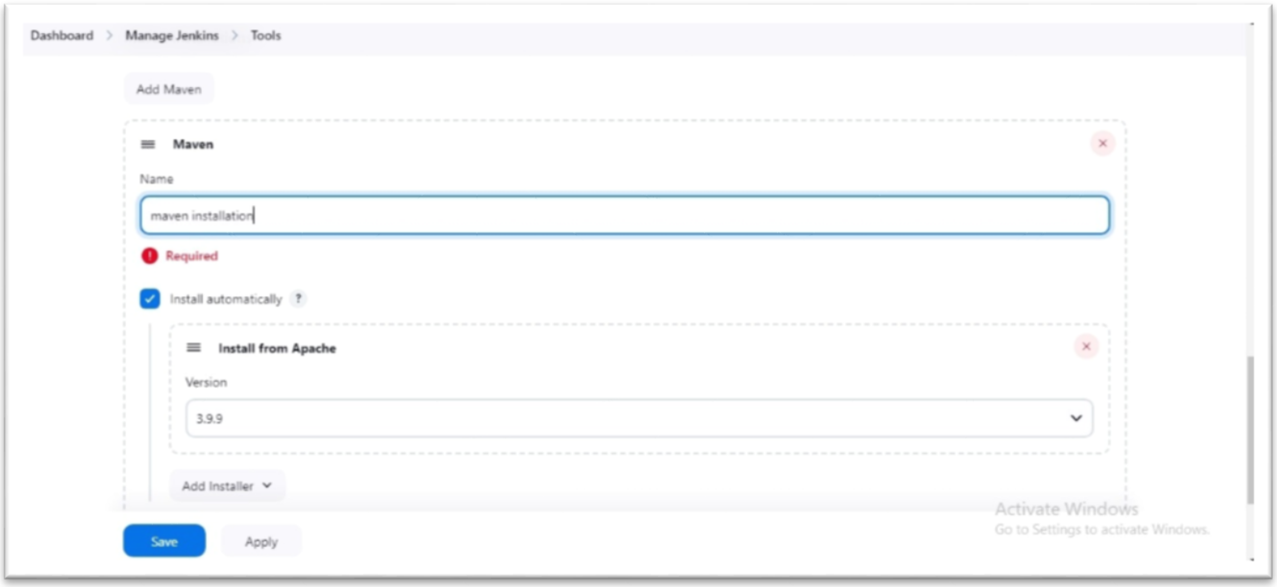
Maven Configuration

Default settings provider: Use default maven settings

Default global settings provider: Use default maven global settings

JDK installations

Add JDK
Save
Apply
Activate Windows
Go to Settings to activate Windows.


Dashboard > Manage Jenkins > Tools
Add Maven

Maven

Name: maven installation

Required

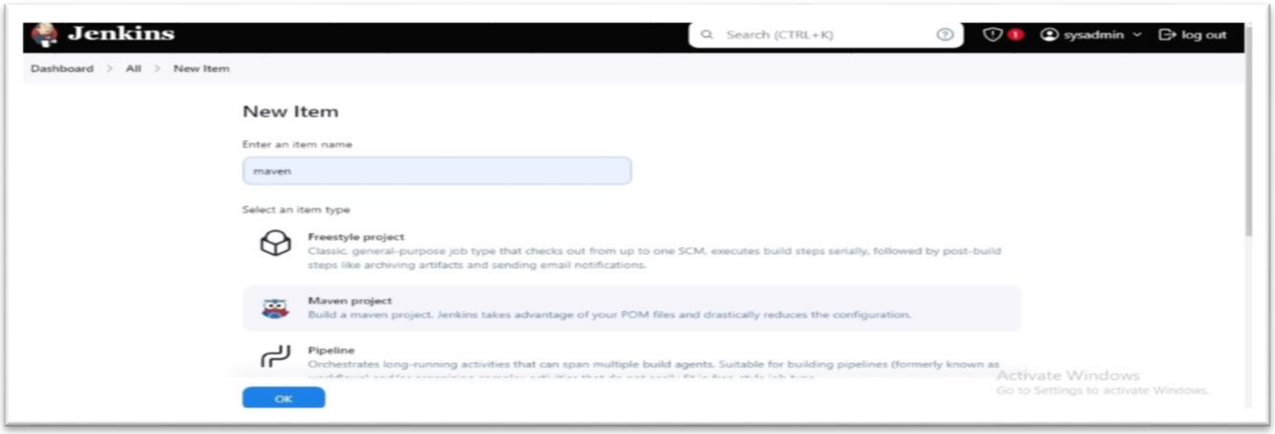
Install automatically [✓]

Install from Apache

Version: 3.9.9

Add Installer
Save
Apply
Activate Windows
Go to Settings to activate Windows.

- **Creating a Spring Boot project with the name 'Maven'**



New Item

Enter an item name: maven

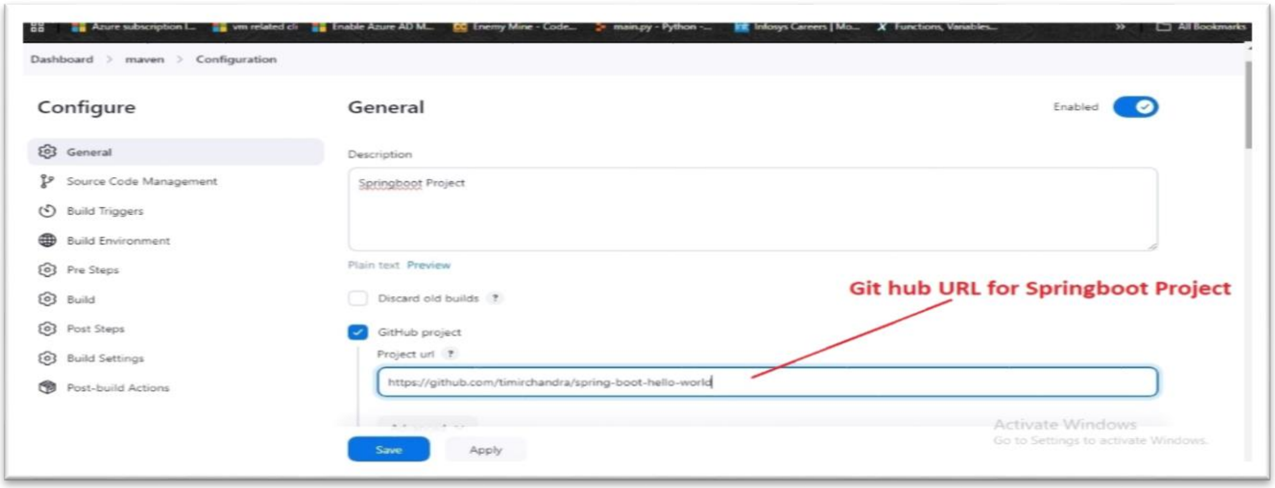
Select an item type

Freestyle project
Classic, general-purpose job type that checks out from up to one SCM, executes build steps serially, followed by post-build steps like archiving artifacts and sending email notifications.

Maven project
Build a maven project. Jenkins takes advantage of your POM files and drastically reduces the configuration.

Pipeline
Orchestrates long-running activities that can span multiple build agents. Suitable for building pipelines (formerly known as

- **Configuration General Setting**


Dashboard > maven > Configuration

Configure

General

Enabled

Description

Springboot Project

Plain text Preview

Discard old builds

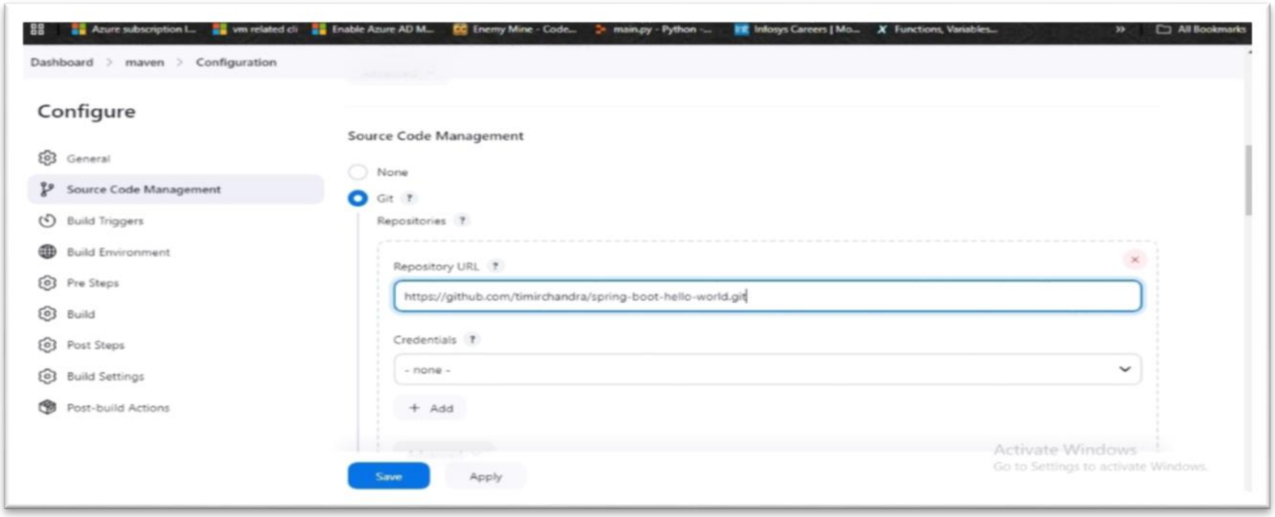
[✓] GitHub project

Project url

https://github.com/timirchandra/spring-boot-hello-world

Git hub URL for Springboot Project

- **Source Code Management**


Dashboard > maven > Configuration

Configure

Source Code Management

None

[✓ Git]

Repositories

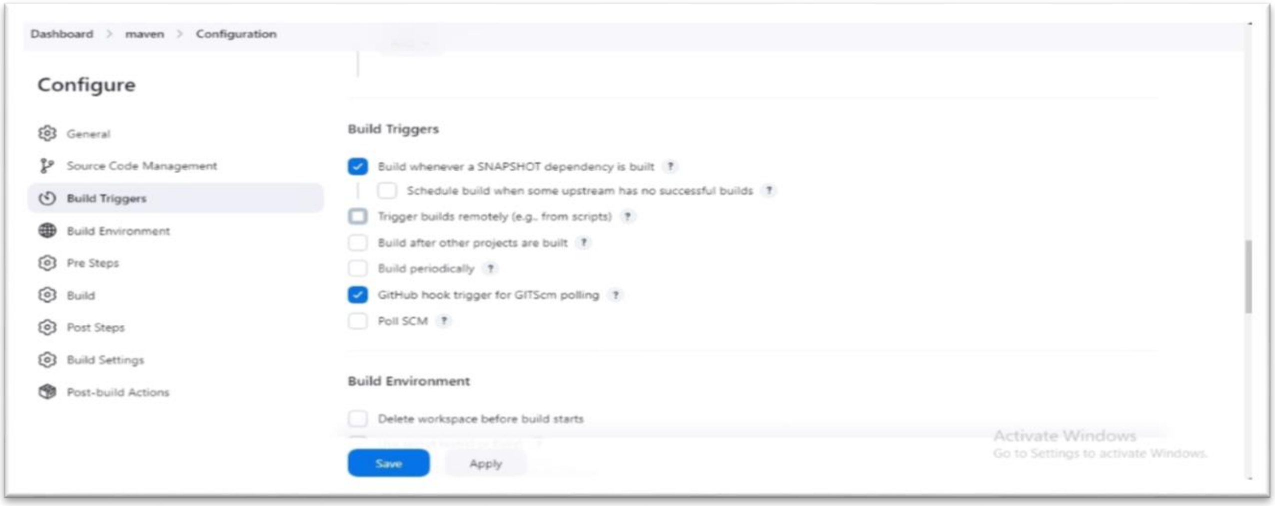
Repository URL

https://github.com/timirchandra/spring-boot-hello-world.git

Credentials

- none -

- **Build Triggers**


Dashboard > maven > Configuration

Configure

Build Triggers

[✓] Build whenever a SNAPSHOT dependency is built

Schedule build when some upstream has no successful builds

Trigger builds remotely (e.g., from scripts)

Build after other projects are built

Build periodically

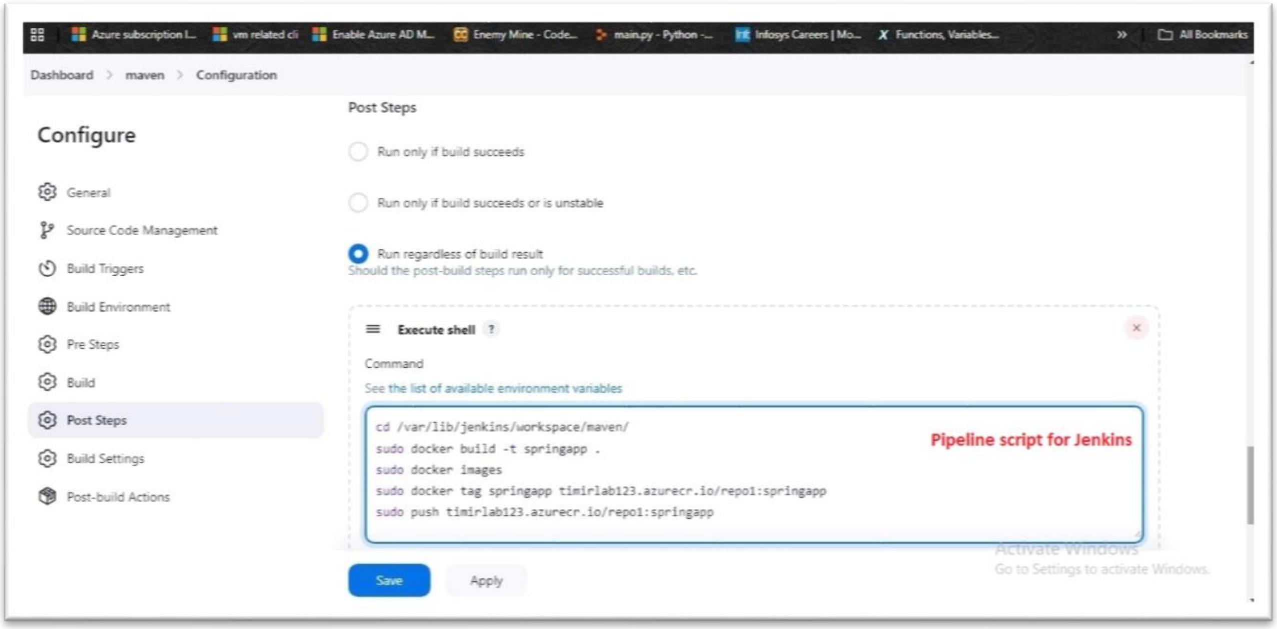
[✓] GitHub hook trigger for GITScm polling

Poll SCM

Build Environment

Delete workspace before build starts

- **Pipeline as Code: Jenkinsfile setup for CI/CD stages**
- **Write pipeline script for Jenkins, then save and apply it**



Post Steps

- Run only if build succeeds
- Run only if build succeeds or is unstable
- [✓ Run regardless of build result]

Should the post-build steps run only for successful builds, etc.

Execute shell

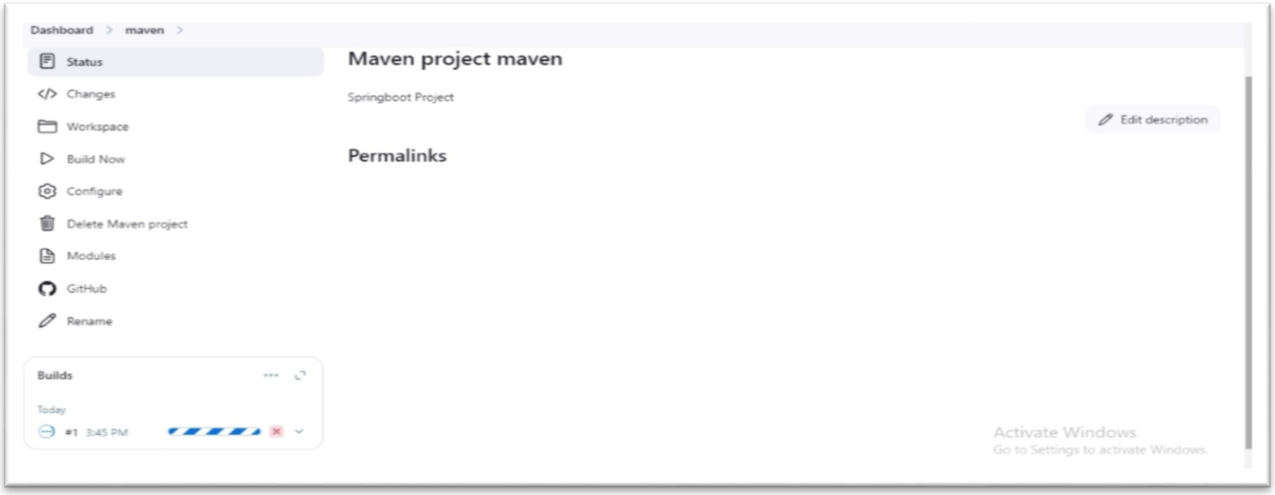
Command

See the list of available environment variables

```
cd /var/lib/jenkins/workspace/maven/
sudo docker build -t springapp .
sudo docker images
sudo docker tag springapp timirlab123.azurecr.io/repo1:springapp
sudo push timirlab123.azurecr.io/repo1:springapp
```

Pipeline script for Jenkins

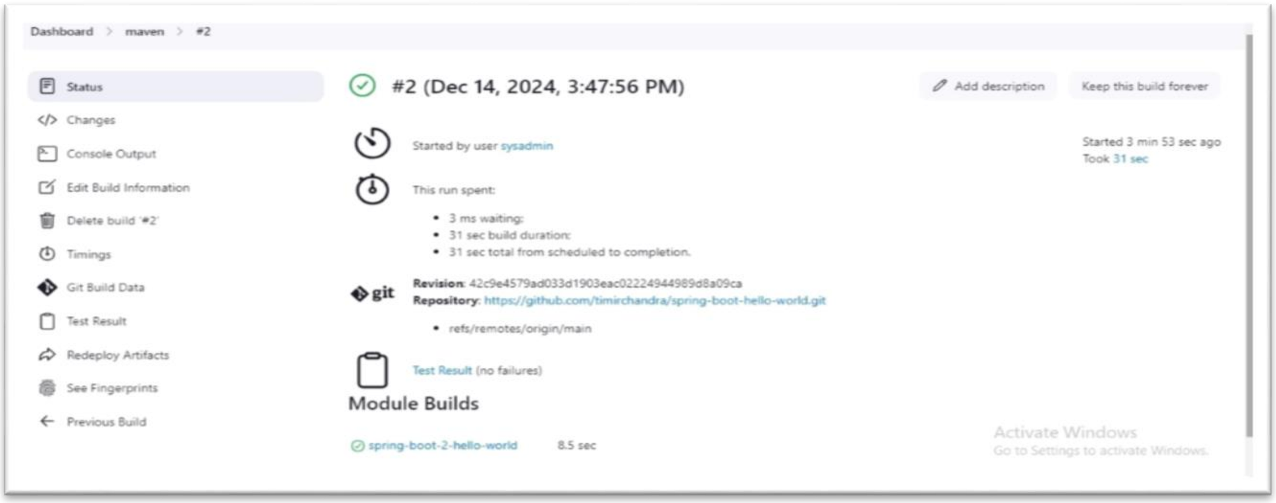
- **Then trigger the build**



Maven project maven

Springboot Project

Permalinks



#2 (Dec 14, 2024, 3:47:56 PM)

Started by user sysadmin

Started 3 min 53 sec ago
Took 31 sec

This run spent:

- 3 ms waiting;
- 31 sec build duration;
- 31 sec total from scheduled to completion.

Revision: 42c9e4579ad033d1903eac02224944989d8a09ca
Repository: https://github.com/timirchandra/spring-boot-hello-world.git

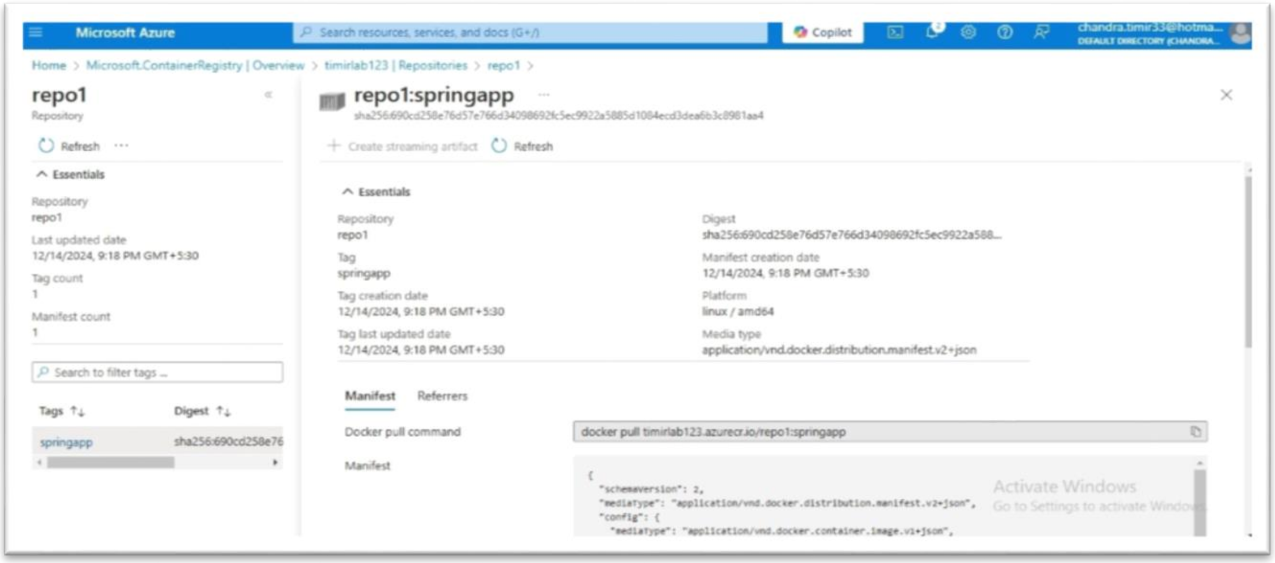
- refs/remotes/origin/main

Test Result (no failures)

Module Builds

spring-boot-2-hello-world 8.5 sec

- **After successfully running the script, the custom image is pushed to the Azure Container Registry (ACR)**



Microsoft Azure
Search resources, services, and docs (G+/)
Copilot
chandra.timir33@hotma...
DEFAULT DIRECTORY (CHANDRA...
Home > Microsoft.ContainerRegistry | Overview > timirlab123 | Repositories > repo1 >
Refresh
Create streaming artifact
Refresh
Essentials
Manifest
Referrers

repo1
Repository

Repository
repo1
Last updated date
12/14/2024, 9:18 PM GMT+5:30
Tag count
1
Manifest count
1

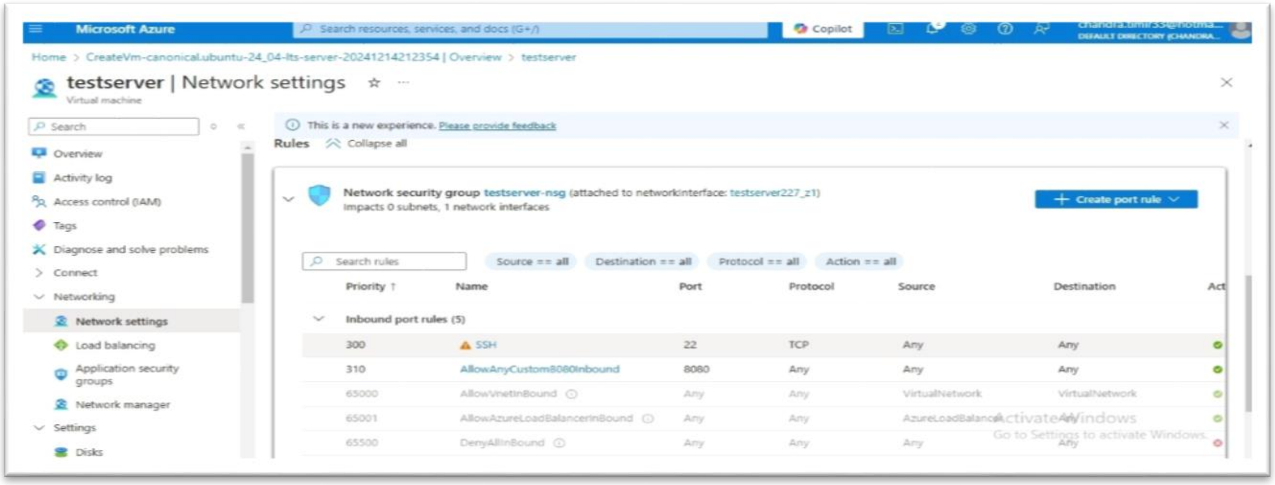
Tags	Digest
springapp	sha256:690cd258e76

repo1:springapp
sha256:690cd258e76d57e766d34098692fc5ec9922a5885d1084ecd3dea6b3c8981aa4

Repository
repo1
Tag
springapp
Tag creation date
12/14/2024, 9:18 PM GMT+5:30
Tag last updated date
12/14/2024, 9:18 PM GMT+5:30
Digest
sha256:690cd258e76d57e766d34098692fc5ec9922a588...
Manifest creation date
12/14/2024, 9:18 PM GMT+5:30
Platform
linux / amd64
Media type
application/vnd.docker.distribution.manifest.v2+json

Docker pull command
```
docker pull timirlab123.azurecr.io/repo1:springapp
```
Manifest
```
{
  "schemaVersion": 2,
  "mediaType": "application/vnd.docker.distribution.manifest.v2+json",
  "config": {
    "mediaType": "application/vnd.docker.container.image.v1+json",
```

- After successfully pushing the image to ACR, we need to test the custom image to ensure it was built properly.
- For testing purposes, we have launched a test server where Docker is already installed and running on port 8080



testserver | Network settings
Virtual machine

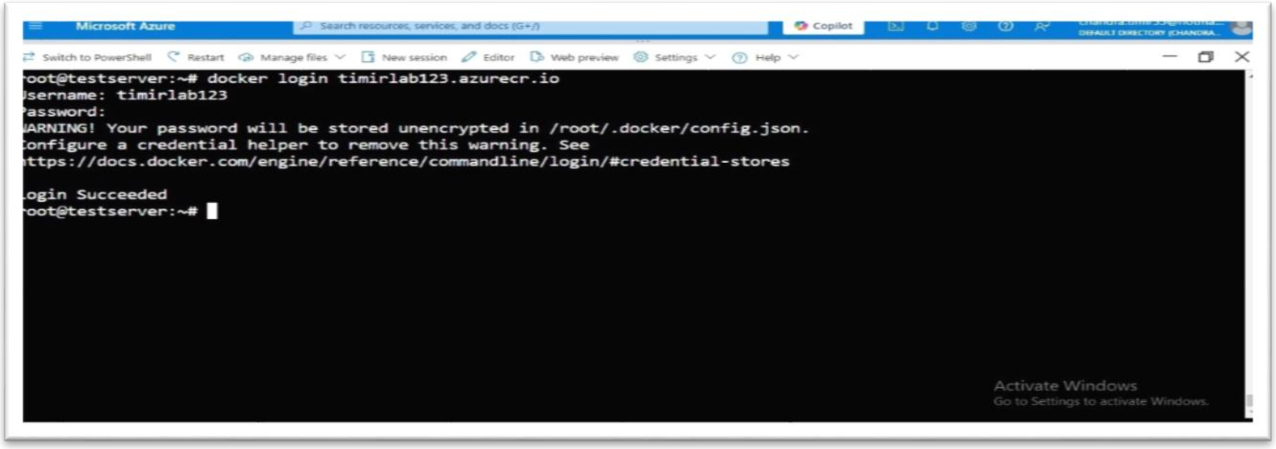
This is a new experience. Please provide feedback

Network security group testserver-nsg (attached to networkinterface: testserver227_z1)
Impacts 0 subnets, 1 network interfaces

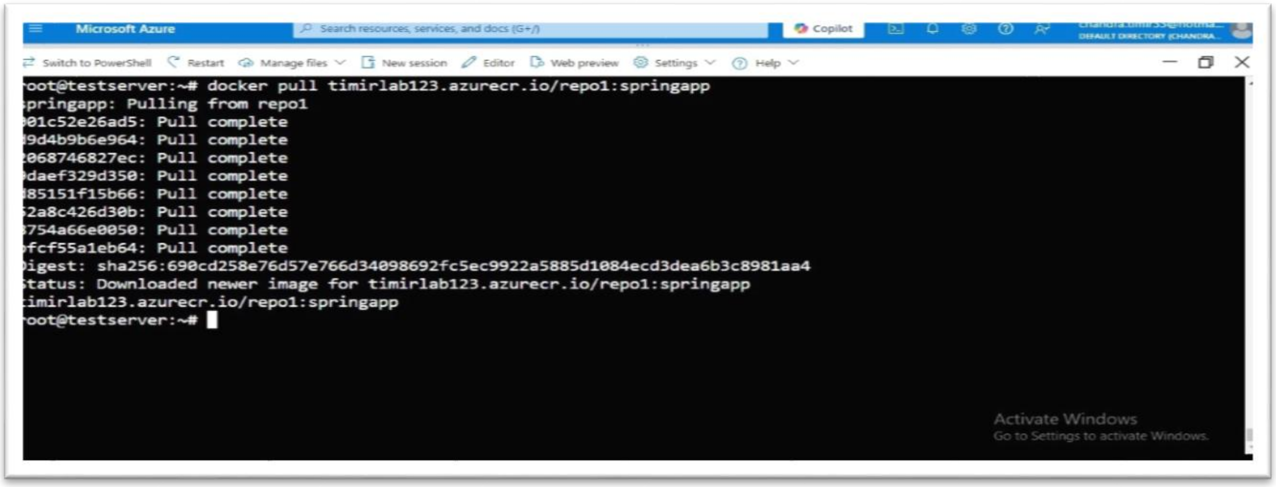
Inbound port rules (5)

Priority ↑	Name	Port	Protocol	Source	Destination	Act
300	SSH	22	TCP	Any	Any	●
310	AllowAnyCustom8080Inbound	8080	Any	Any	Any	●
65000	AllowVnetInBound	Any	Any	VirtualNetwork	VirtualNetwork	●
65001	AllowAzureLoadBalancerInBound	Any	Any	AzureLoadBalancer	Any	●
65500	DenyAllInBound	Any	Any	Any	Any	●

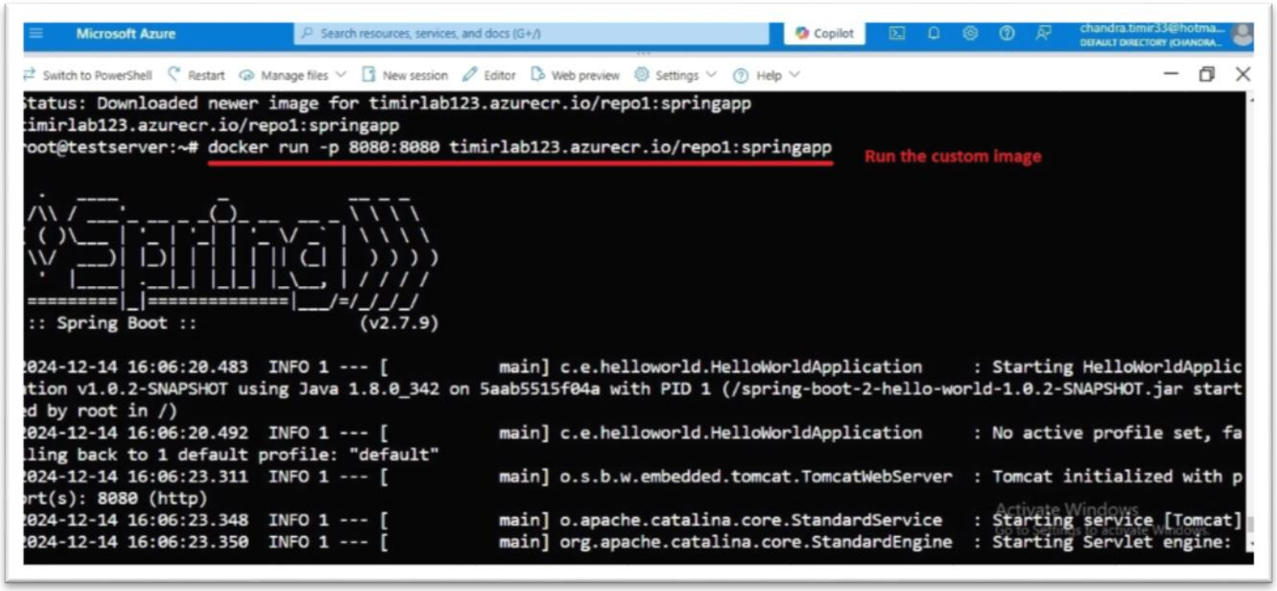
- Attach the test server to ACR (timirlab123) for image pull



```
oot@testserver:~# docker login timirlab123.azurecr.io
Jsername: timirlab123
assword:
ARNING! Your password will be stored unencrypted in /root/.docker/config.json.
onfigure a credential helper to remove this warning. See
ttps://docs.docker.com/engine/reference/commandline/login/#credential-stores

ogin Succeeded
oot@testserver:~# █
```

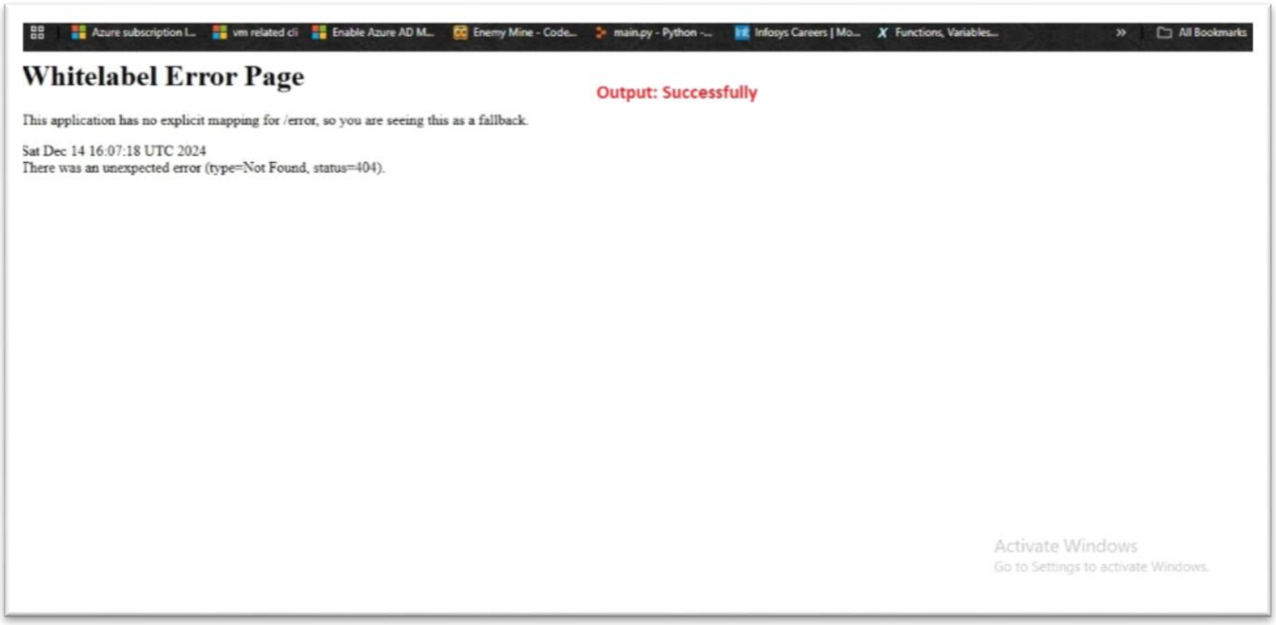


```
root@testserver:~# docker pull timirlab123.azurecr.io/repo1:springapp
springapp: Pulling from repo1
001c52e26ad5: Pull complete
19d4b9b6e964: Pull complete
2068746827ec: Pull complete
9daef329d350: Pull complete
d85151f15b66: Pull complete
52a8c426d30b: Pull complete
3754a66e0050: Pull complete
bfcf55a1eb64: Pull complete
Digest: sha256:690cd258e76d57e766d34098692fc5ec9922a5885d1084ecd3dea6b3c8981aa4
Status: Downloaded newer image for timirlab123.azurecr.io/repo1:springapp
timirlab123.azurecr.io/repo1:springapp
root@testserver:~# 
```



```
tatus: Downloaded newer image for timirlab123.azurecr.io/repo1:springapp
imirlab123.azurecr.io/repo1:springapp
oot@testserver:~# docker run -p 8080:8080 timirlab123.azurecr.io/repo1:springapp

  .   ____          _            __ _ _
 /\\ / ___'_ __ _ _(_)_ __  __ _ \ \ \ \
( ( )\___ | '_ | '_| | '_ \/ _` | \ \ \ \
 \\/  ___)| |_)| | | | | || (_| |  ) ) ) )
  '  |____| .__|_| |_|_| |_\__, | / / / /
 =========|_|==============|___/=/_/_/_/
 :: Spring Boot ::                (v2.7.9)

024-12-14 16:06:20.483  INFO 1 --- [           main] c.e.helloworld.HelloWorldApplication     : Starting HelloWorldApplic
tion v1.0.2-SNAPSHOT using Java 1.8.0_342 on 5aab5515f04a with PID 1 (/spring-boot-2-hello-world-1.0.2-SNAPSHOT.jar start
d by root in /)
024-12-14 16:06:20.492  INFO 1 --- [           main] c.e.helloworld.HelloWorldApplication     : No active profile set, fa
lling back to 1 default profile: "default"
024-12-14 16:06:23.311  INFO 1 --- [           main] o.s.b.w.embedded.tomcat.TomcatWebServer  : Tomcat initialized with p
rt(s): 8080 (http)
024-12-14 16:06:23.348  INFO 1 --- [           main] o.apache.catalina.core.StandardService   : Starting service [Tomcat]
024-12-14 16:06:23.350  INFO 1 --- [           main] org.apache.catalina.core.StandardEngine  : Starting Servlet engine:
```

Run the custom image



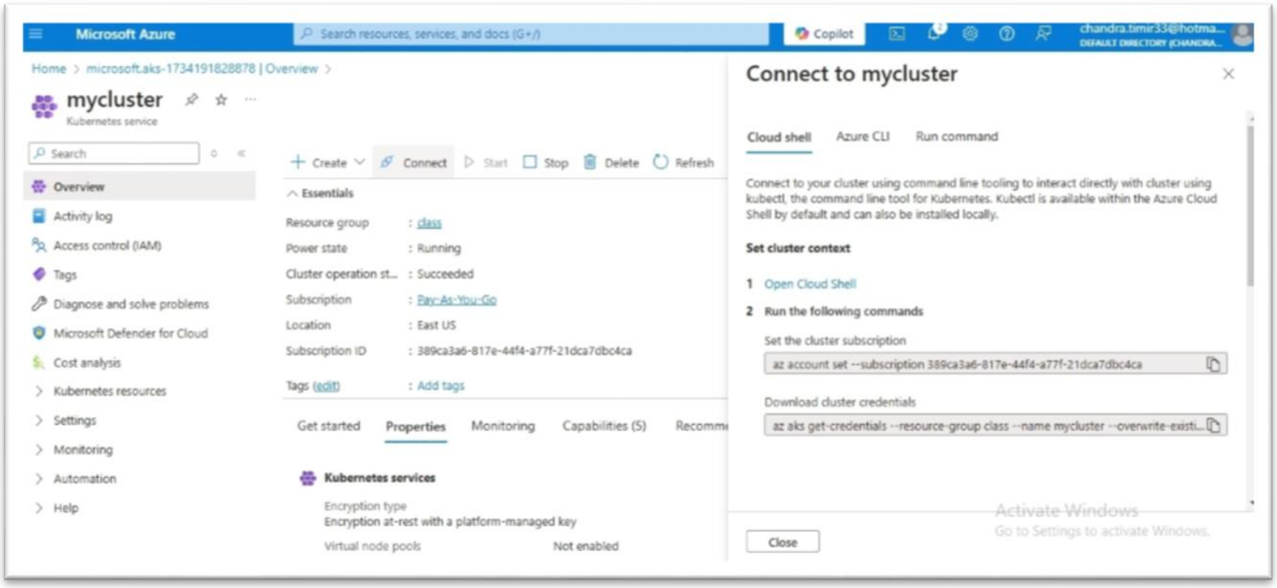
Whitelabel Error Page

Output: Successfully

This application has no explicit mapping for /error, so you are seeing this as a fallback.

Sat Dec 14 16:07:18 UTC 2024
There was an unexpected error (type=Not Found, status=404).

- After successfully passing the testing, we will proceed to the production phase. For this purpose, we need to launch AKS (e.g., the cluster name is mycluster).



Essentials

Resource group	: class
Power state	: Running
Cluster operation st...	: Succeeded
Subscription	: Pay-As-You-Go
Location	: East US
Subscription ID	: 389ca3a6-817e-44f4-a77f-21dca7dbc4ca
Tags (edit)	: Add tags

Kubernetes services

Encryption type
Encryption at-rest with a platform-managed key

Virtual node pools Not enabled

Connect to mycluster

Connect to your cluster using command line tooling to interact directly with cluster using kubectl, the command line tool for Kubernetes. Kubectl is available within the Azure Cloud Shell by default and can also be installed locally.

Set cluster context

1 Open Cloud Shell

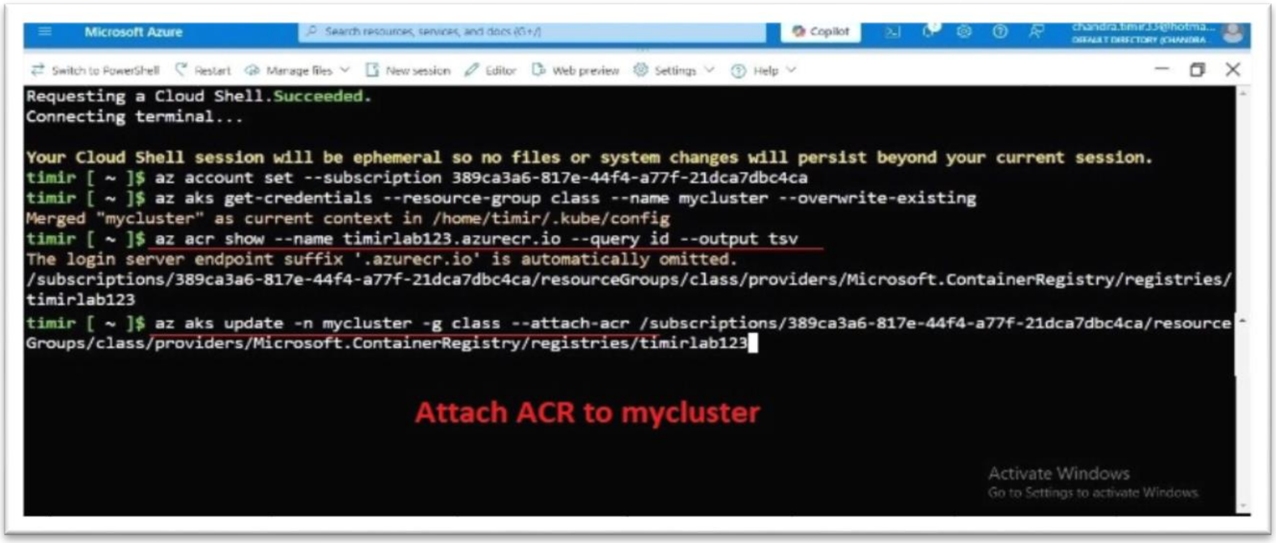
2 Run the following commands

Set the cluster subscription

```
az account set --subscription 389ca3a6-817e-44f4-a77f-21dca7dbc4ca
```

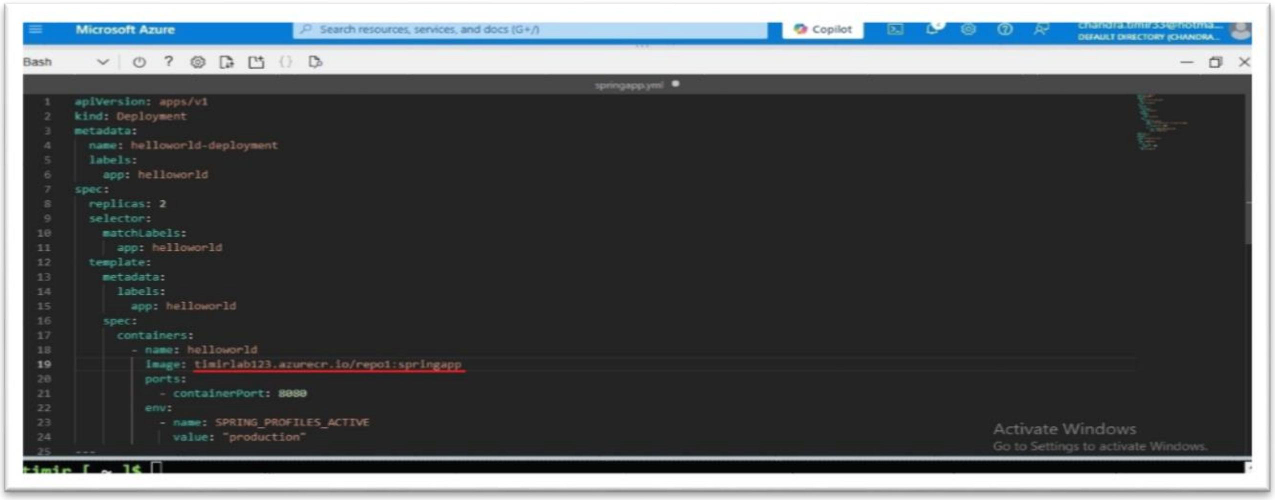
Download cluster credentials

```
az aks get-credentials --resource-group class --name mycluster --overwrite-existi...
```

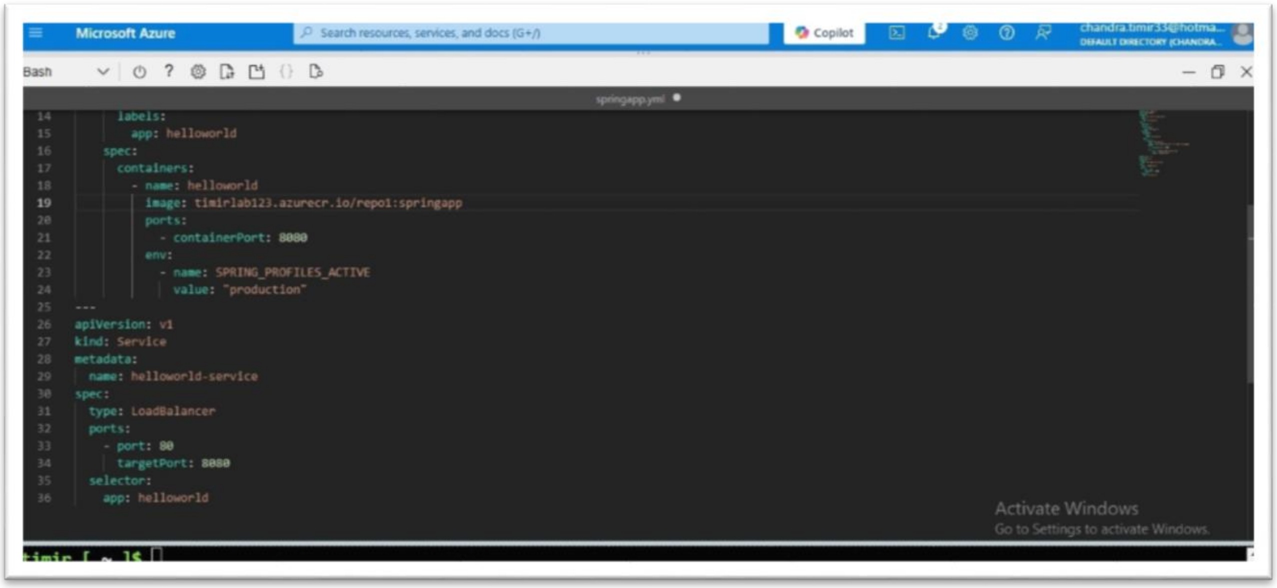


```
Requesting a Cloud Shell.Succeeded.
Connecting terminal...

Your Cloud Shell session will be ephemeral so no files or system changes will persist beyond your current session.
timir [ ~ ]$ az account set --subscription 389ca3a6-817e-44f4-a77f-21dca7dbc4ca
timir [ ~ ]$ az aks get-credentials --resource-group class --name mycluster --overwrite-existing
Merged "mycluster" as current context in /home/timir/.kube/config
timir [ ~ ]$ az acr show --name timirlab123.azurecr.io --query id --output tsv
The login server endpoint suffix '.azurecr.io' is automatically omitted.
/subscriptions/389ca3a6-817e-44f4-a77f-21dca7dbc4ca/resourceGroups/class/providers/Microsoft.ContainerRegistry/registries/
timirlab123
timir [ ~ ]$ az aks update -n mycluster -g class --attach-acr /subscriptions/389ca3a6-817e-44f4-a77f-21dca7dbc4ca/resource
Groups/class/providers/Microsoft.ContainerRegistry/registries/timirlab123
```

- YAML file for application deployment.

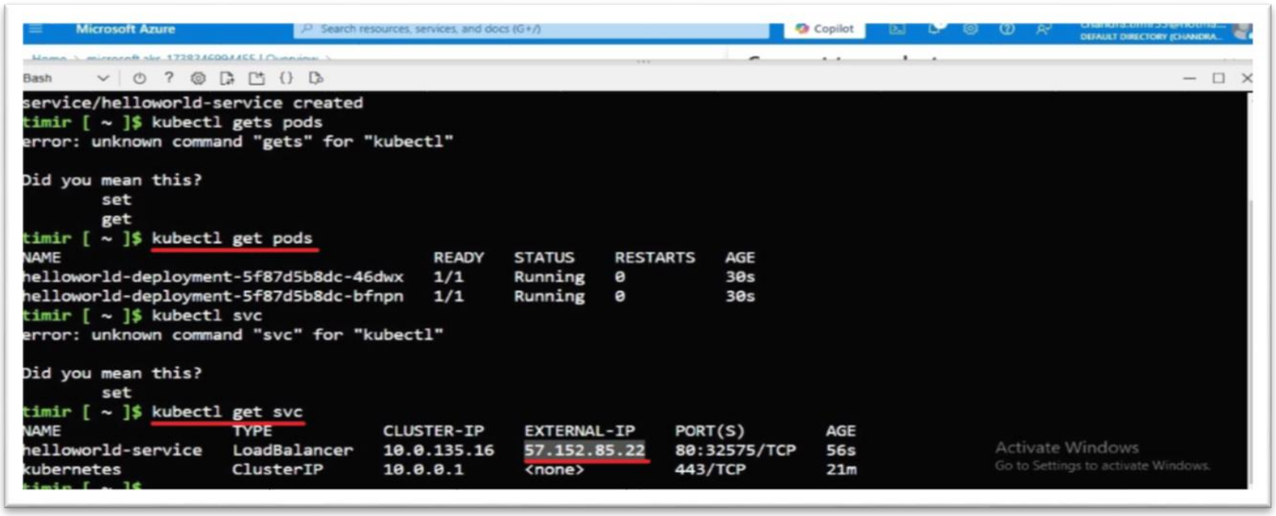

Microsoft Azure
Search resources, services, and docs (G+/)
Copilot
Bash
springapp.yml
Activate Windows
Go to Settings to activate Windows.

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: helloworld-deployment
  labels:
    app: helloworld
spec:
  replicas: 2
  selector:
    matchLabels:
      app: helloworld
  template:
    metadata:
      labels:
        app: helloworld
    spec:
      containers:
        - name: helloworld
          image: timirlab123.azurecr.io/repo1:springapp
          ports:
            - containerPort: 8080
          env:
            - name: SPRING_PROFILES_ACTIVE
              value: "production"
---
timir [ ~ ]$
```

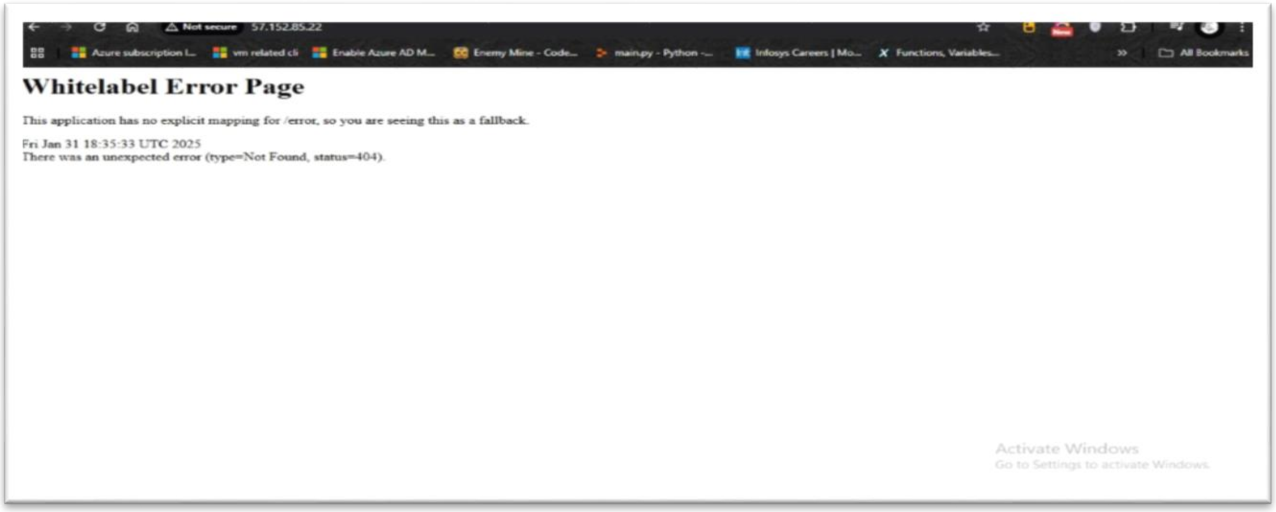


```
        labels:
          app: helloworld
      spec:
        containers:
          - name: helloworld
            image: timirlab123.azurecr.io/repo1:springapp
            ports:
              - containerPort: 8080
            env:
              - name: SPRING_PROFILES_ACTIVE
                value: "production"
---
apiVersion: v1
kind: Service
metadata:
  name: helloworld-service
spec:
  type: LoadBalancer
  ports:
    - port: 80
      targetPort: 8080
  selector:
    app: helloworld
```

- Deployment scripts for AKS using kubectl
- Code snippets for each stage of the pipeline



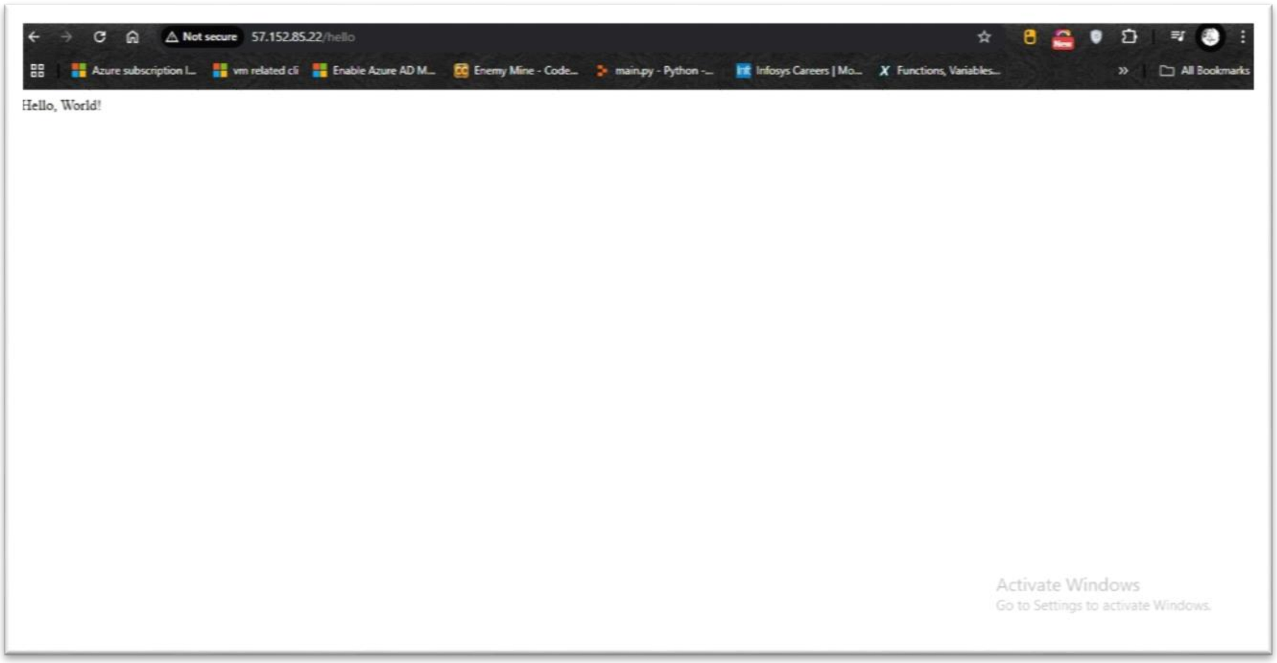
```
service/helloworld-service created
timir [ ~ ]$ kubectl gets pods
error: unknown command "gets" for "kubectl"

Did you mean this?
        set
        get
timir [ ~ ]$ kubectl get pods
NAME                                     READY   STATUS    RESTARTS   AGE
helloworld-deployment-5f87d5b8dc-46dwx   1/1     Running   0          30s
helloworld-deployment-5f87d5b8dc-bfnpn   1/1     Running   0          30s
timir [ ~ ]$ kubectl svc
error: unknown command "svc" for "kubectl"

Did you mean this?
        set
timir [ ~ ]$ kubectl get svc
NAME                 TYPE           CLUSTER-IP     EXTERNAL-IP    PORT(S)        AGE
helloworld-service   LoadBalancer   10.0.135.16    57.152.85.22   80:32575/TCP   56s
kubernetes           ClusterIP      10.0.0.1       <none>         443/TCP        21m
```



Whitelabel Error Page

This application has no explicit mapping for /error, so you are seeing this as a fallback.

Fri Jan 31 18:35:33 UTC 2025
There was an unexpected error (type=Not Found, status=404).



Hello, World!

Security and Compliance

- Security considerations in the DevOps pipeline
- Securing Jenkins with authentication and access control
- Container security best practices (e.g., image scanning)
- Kubernetes security and AKS-specific security practices
- Compliance with best practices and industry standards

Testing and Quality Assurance

- Overview of testing strategy for CI/CD
- Unit, integration, and functional testing
- Performance and load testing on AKS

Monitoring and Logging

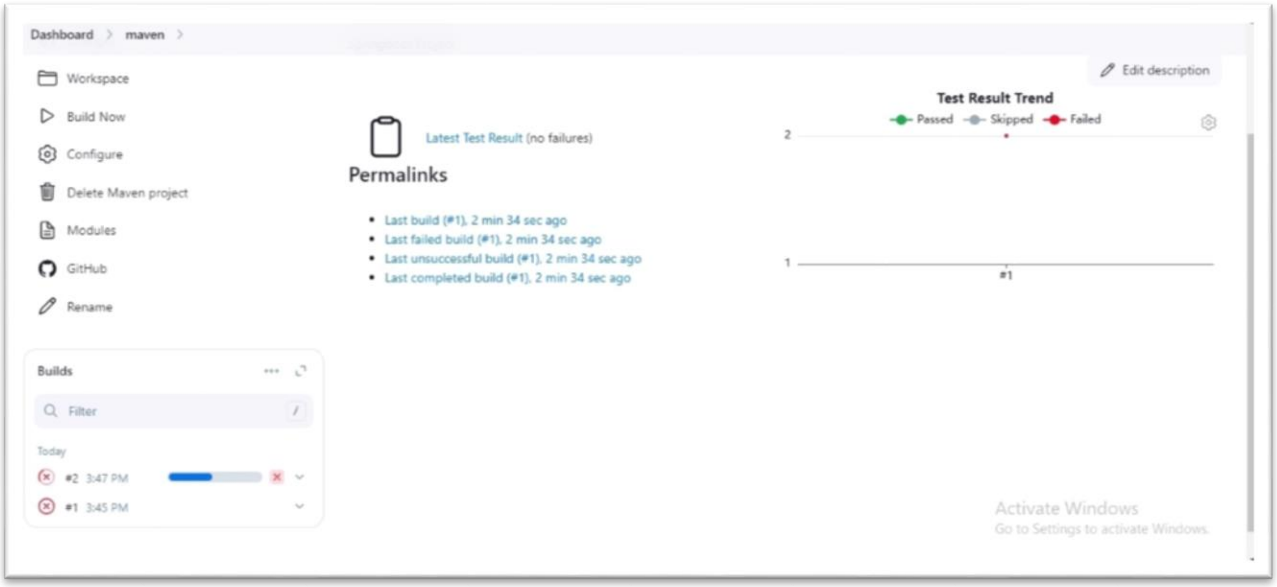
- Monitoring tools and techniques for the Jenkins pipeline
- Application performance monitoring on AKS
- Logging for error tracking and troubleshooting
- Integrating monitoring with Jenkins and AKS

Challenges and Solutions

- ○ Technical and procedural challenges in setting up the pipeline failed during the image build or push stages due to incorrect configurations, missing dependencies.

- ○ Authentication with the Azure Container Registry (ACR) failed, causing the push to the registry to be rejected.
- ○ **Solutions implemented to overcome challenges:** After reviewing the error logs, we identified the specific stage where the pipeline was failing and focused on resolving the issue step by step. We addressed issues related to missing files or incorrect paths in the `Dockerfile` and added the necessary files and dependencies to the Docker image build context
- ○ **Lessons learned during implementation** :

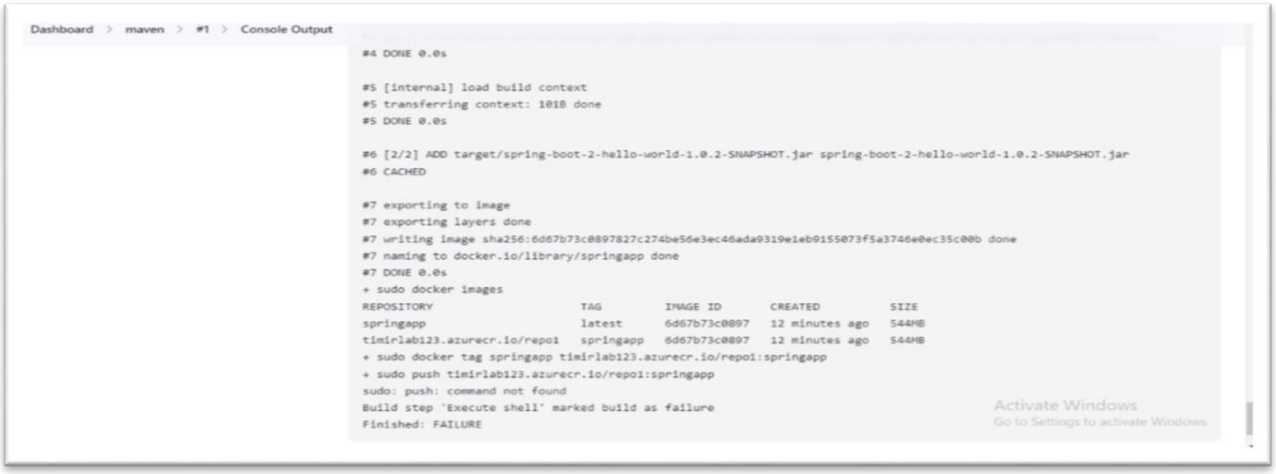
    - ■ **Importance of Detailed Error Logs:** Always analyze error messages in detail. Often, the error logs provide direct hints about the root cause of the issue (e.g., authentication failures, missing files, incorrect paths)



Latest Test Result (no failures)

Permalinks

- Last build (#1), 2 min 34 sec ago
- Last failed build (#1), 2 min 34 sec ago
- Last unsuccessful build (#1), 2 min 34 sec ago
- Last completed build (#1), 2 min 34 sec ago



Dashboard > maven > #1 > Console Output
Activate Windows
Go to Settings to activate Windows.

```
#4 DONE 0.0s

#5 [internal] load build context
#5 transferring context: 101B done
#5 DONE 0.0s

#6 [2/2] ADD target/spring-boot-2-hello-world-1.0.2-SNAPSHOT.jar spring-boot-2-hello-world-1.0.2-SNAPSHOT.jar
#6 CACHED

#7 exporting to image
#7 exporting layers done
#7 writing image sha256:6d67b73c0897827c274be56e3ec46ada9319e1eb9155073f5a3746e0ec35c00b done
#7 naming to docker.io/library/springapp done
#7 DONE 0.0s
+ sudo docker images
REPOSITORY                     TAG         IMAGE ID       CREATED          SIZE
springapp                      latest      6d67b73c0897   12 minutes ago   544MB
timirlab123.azurecr.io/repo1   springapp   6d67b73c0897   12 minutes ago   544MB
+ sudo docker tag springapp timirlab123.azurecr.io/repo1:springapp
+ sudo push timirlab123.azurecr.io/repo1:springapp
sudo: push: command not found
Build step 'Execute shell' marked build as failure
Finished: FAILURE
```

        - **Check Dockerfile Context:** Verify the `Dockerfile` and project directory structure. Any missing dependencies or incorrect paths in the build context can cause unexpected failures.

Results and Analysis

- Analysis of the outcomes from the pipeline
- Benefits of the automated pipeline for development and deployment
- Performance metrics and improvements observed with containerization and AKS

Conclusion

- Summary of findings and project outcomes
- Future recommendations for enhancing the pipeline
- Final thoughts on the role of DevOps, CI/CD, and AKS in modern application development

References

- List of references for tools, techniques, and theories discussed in the report

Appendices

- Additional diagrams, code snippets, and configuration files

---

Introduction

- ## Overview of DevOps and CI/CD

DevOps is a set of practices that integrates software development (Dev) and IT operations (Ops) to enhance collaboration, automation, and efficiency throughout the software lifecycle. The goal is to deliver high-quality software quickly and reliably. Continuous Integration (CI) involves automatically integrating code changes into a shared repository, where tests are run to identify issues early. Continuous Deployment (CD) automates the process of deploying code to production after successful testing, ensuring fast and reliable delivery of features. Together, CI/CD streamline workflows, reduce errors, and enable rapid, consistent delivery of applications with improved collaboration and faster time to market.

● Role of Jenkins in Continuous Integration and Continuous Deployment (CI/CD)

Jenkins plays a crucial role in Continuous Integration (CI) and Continuous Deployment (CD) by automating the build, test, and deployment processes in the software development lifecycle. In CI, Jenkins facilitates the integration of code from multiple developers into a shared repository by running automated tests on each commit, ensuring that errors are detected early. This reduces integration problems and improves collaboration among teams.

For CD, Jenkins automates the deployment of applications to different environments (e.g., staging, production) after successful testing. By integrating with various plugins, Jenkins can trigger deployments to cloud platforms like Azure, AWS, or Kubernetes environments, making the process seamless and reliable. The pipeline can include stages like code quality checks, unit tests, and deployment, ensuring that the application is always in a deployable state. Jenkins thus accelerates software delivery while maintaining quality, reducing manual intervention, and improving overall efficiency in DevOps practices.

● Importance of Kubernetes and AKS (Azure Kubernetes Service) in containerized applications

Kubernetes is an open-source container orchestration platform that automates the deployment, scaling, and management of containerized applications. It helps in managing clusters of containers across multiple machines, ensuring high availability, load balancing, and efficient resource utilization. Kubernetes is crucial for large-scale, distributed applications, providing features like self-healing, auto-scaling, and rollbacks.

Azure Kubernetes Service (AKS) is a managed Kubernetes service provided by Microsoft Azure, simplifying the process of deploying, managing, and scaling containerized applications. AKS automates tasks such as patching, upgrading, and monitoring, reducing operational overhead. It seamlessly integrates with Azure's ecosystem, offering enhanced security, scalability, and cost management. AKS enables businesses to leverage the power of Kubernetes while focusing on application development rather than infrastructure management, making it a critical tool for cloud-native applications.

● Project objectives and expected outcomes

The primary objective of this project is to design and implement a robust DevOps pipeline using Jenkins for Continuous Integration (CI) and Continuous Deployment (CD), coupled with Azure Kubernetes Service (AKS) for container orchestration. The pipeline will automate the entire software delivery process, ensuring faster, reliable, and scalable deployment of applications.

Key objectives include:

1. **CI/CD Automation:** Automate the build, test, and deployment processes to streamline the software development lifecycle and reduce manual intervention. This involves

integrating Jenkins with Git repositories and configuring pipelines for automatic code integration, testing, and deployment.

2. **Containerization:** Use Docker to containerize applications, making them portable across different environments while ensuring consistency and scalability.
3. **Deployment on AKS:** Set up an Azure Kubernetes Service cluster for container orchestration, ensuring efficient scaling, load balancing, and high availability of applications in production environments.
4. **Security and Monitoring:** Implement security best practices and monitoring tools to ensure the safety, stability, and performance of the application during deployment and scaling.

Expected outcomes of the project include a fully automated CI/CD pipeline that reduces development cycles, enhances code quality through continuous testing, and ensures consistent, fast, and secure deployment of applications on scalable cloud infrastructure. This will significantly improve the speed and reliability of application releases.

Literature Review

- ### Concepts of DevOps and its evolution

DevOps is a set of practices that integrates software development (Dev) and IT operations (Ops) to improve collaboration, automation, and efficiency throughout the software delivery lifecycle. It aims to shorten development cycles, increase deployment frequency, and ensure higher-quality software. The core principles of DevOps include automation, continuous integration (CI), continuous delivery (CD), collaboration, monitoring, and feedback loops.

Evolution of DevOps:

- **Pre-DevOps (1990s):** Software development and IT operations were siloed, leading to slow deployments, miscommunication, and quality issues.
- **Agile Development (Early 2000s):** Agile methodologies emerged, emphasizing iterative development, fast feedback, and adaptability. However, the separation between development and operations remained.
- **DevOps Emergence (2007-2010):** The term "DevOps" was coined as a response to challenges faced by teams working in isolated silos. DevOps focuses on collaboration between development and operations teams, creating a culture of shared responsibility and continuous improvement.
- **DevOps Adoption (2010s):** The rise of cloud computing, automation tools, and containerization technologies (like Docker and Kubernetes) accelerated DevOps adoption. CI/CD pipelines, infrastructure as code (IaC), and microservices became fundamental to modern DevOps practices.

Timeline:

1. **1990s:** Separate Dev and Ops roles.

2. **Early 2000s:** Agile principles emerge.
3. **2007-2010:** DevOps concept introduced.
4. **2010s:** Widespread DevOps adoption with cloud and automation tools.

Today, DevOps continues to evolve with AI/ML integration, serverless architectures, and enhanced security measures.

- ## Overview of CI/CD principles and tools

Continuous Integration (CI) and Continuous Deployment (CD) are fundamental principles in modern software development, aiming to automate and streamline the process of building, testing, and deploying applications.

Continuous Integration (CI) involves frequently integrating code changes from multiple developers into a shared repository. This process triggers automated build and testing pipelines to ensure that any issues are identified and addressed early, enhancing collaboration and code quality.

Continuous Deployment (CD) extends CI by automating the deployment process. After passing automated tests, code is automatically deployed to production, ensuring that new features and updates are delivered rapidly and reliably. CD minimizes manual intervention and reduces the risk of human error in deployments.

Key **CI/CD tools** include:

2. **Jenkins:** A widely-used open-source automation server that orchestrates CI/CD pipelines.
3. **GitLab CI/CD:** A built-in tool for CI/CD automation in GitLab.
4. **CircleCI:** A cloud-based CI/CD tool focused on speed and efficiency.
5. **Travis CI:** A CI/CD service used with GitHub repositories.
6. **Azure DevOps:** A suite of tools for end-to-end DevOps automation.

- ## Jenkins as a CI/CD tool and its integration capabilities

Jenkins is a popular open-source automation tool for Continuous Integration (CI) and Continuous Deployment (CD) that facilitates the automation of building, testing, and deploying applications. It helps streamline the software development process by integrating code changes frequently and ensuring early detection of errors. Jenkins supports a wide range of plugins, allowing it to integrate with version control systems like Git, build tools such as Maven and Gradle, testing frameworks like JUnit, and deployment platforms like Kubernetes and cloud services.

Jenkins' integration capabilities extend to various DevOps tools, enabling end-to-end automation across the development lifecycle. It can trigger automated tests, initiate deployments, and send notifications, making it a central tool in DevOps pipelines for accelerating software delivery while maintaining high quality and consistency.

- ## Containerization and orchestration with Kubernetes

Containerization is the process of packaging an application and its dependencies into a lightweight, portable container that can run consistently across different environments. Docker is the most widely used containerization platform, providing isolation and scalability.

Kubernetes, an open-source container orchestration platform, manages the deployment, scaling, and operation of containerized applications. It automates tasks such as container scheduling, load balancing, and self-healing, ensuring high availability and reliability. Kubernetes supports features like automatic scaling, rolling updates, and service discovery, allowing organizations to efficiently manage large, complex applications in distributed environments, often in cloud-based infrastructures.

- ## Azure Kubernetes Service (AKS) and its advantages for deploying scalable applications

Azure Kubernetes Service (AKS) is a managed Kubernetes service provided by Microsoft Azure, designed to simplify the deployment, management, and scaling of containerized applications. AKS abstracts much of the complexity of Kubernetes management, such as cluster setup, patching, and maintenance, enabling developers to focus on building applications rather than managing infrastructure.

Key advantages of AKS include:

1. **Scalability:** AKS enables automatic scaling of containerized applications, adjusting resources based on demand.
2. **High Availability:** AKS ensures fault tolerance by distributing containers across multiple nodes and availability zones.
3. **Integrated Azure Services:** Seamless integration with Azure services like Active Directory, monitoring, and security.
4. **Cost Efficiency:** Pay-as-you-go pricing for infrastructure, reducing operational costs.
5. **Security:** Built-in security features like Azure Active Directory integration and network policies ensure robust protection.

AKS simplifies container orchestration, enabling organizations to deploy highly scalable, reliable, and secure applications on the cloud.