

Bootcamp-Project 5

ENTERPRISE JAVA CI/CD WITH JENKINS, GITOPS, AND
KUBERNETES OBSERVABILITY

Mohanramrajan Erran Bothalraj

Table of Contents

Introduction	2
Project Objectives	2
Expected Outcome	2
Pre-requisite:.....	2
Architecture Diagram	3
Solution Steps	3
1. Project Setup.....	3
1.1. Installation of Jenkins, SonarQube, Maven, and Docker	3
1.2. Configure Jenkins.....	4
1.3. Configure SonarQube.....	6
2. Continuous Integration	7
2.1. Jenkins configuration	7
2.2. Credential Management.....	7
2.3. Pipeline Creation and Execution	7
3. Argo CD.....	11
3.1. AKS Cluster Installation	11
3.2. Install ArgoCD Operator and controller on AKS cluster	11
3.3. Configure ArgoCD APP	12
Validation.....	13
Monitoring.....	14
Conclusion	15
APPENDIX	16

Introduction

In the modern software development landscape, speed, quality, and reliability are essential for delivering value to users. The "**Enterprise Java CI/CD with Jenkins, GitOps, and Kubernetes Observability**" project exemplifies a robust and scalable DevOps pipeline for a Java microservice. This end-to-end solution streamlines the software development lifecycle from code commit to production deployment while ensuring high code quality, automated delivery, and real-time system observability. By integrating leading tools such as **Jenkins, SonarQube, Argo CD, Prometheus, and Grafana**, the project adopts industry best practices in **Continuous Integration (CI), Continuous Deployment (CD), GitOps, and Kubernetes-native observability**.

Project Objectives

The key objectives of this project are:

1. **Automate the entire CI/CD pipeline** to ensure rapid, repeatable, and reliable deployments.
2. **Incorporate static code analysis** using SonarQube to enforce coding standards and improve maintainability.
3. **Build and containerize the Java application** using Docker to enable consistent deployment environments.
4. **Adopt GitOps practices** using Argo CD for declarative, version-controlled, and traceable deployments on Kubernetes.
5. **Enable observability and proactive monitoring** with Prometheus and Grafana for better system reliability and performance insights.

Expected Outcome

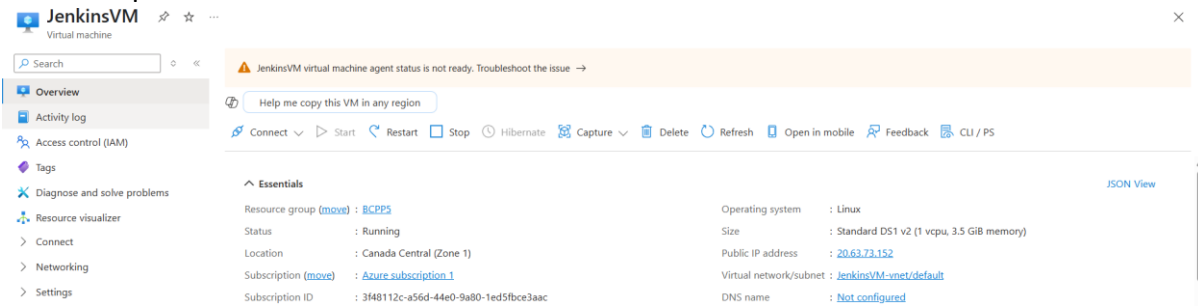
By the end of this project, the following outcomes are expected:

- A fully automated CI/CD pipeline using Jenkins, triggered by code commits and executing build, test, analysis, containerization, and deployment stages.
- Quality gates enforced through SonarQube to ensure code quality and maintainability before deployment.
- A production-ready Docker image of the Java application stored in a centralized container registry.
- A GitOps-managed deployment process via Argo CD, ensuring that the application state in the Kubernetes cluster matches the desired state defined in Git.
- A real-time observability stack using Prometheus and Grafana, with dashboards and alerts for key performance metrics such as response time, memory usage, and error rates.
- Improved deployment traceability, reduced manual effort, and faster feedback cycles for developers and operations teams.

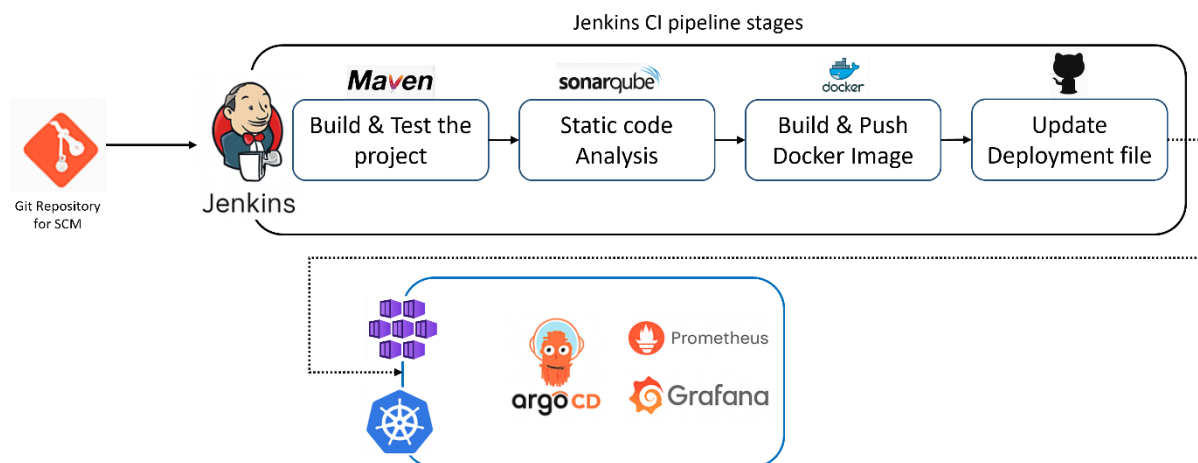
Pre-requisite:

1. Java application code available in GitHub repository:
<https://github.com/merranbo1989/BCP-P5.git>
Project Folder: java-maven-sonar-argocd-helm-k8s

2. Create an Ubuntu Virtual Machine to install Jenkins, SonarQube, Maven, and Docker dependencies.



Architecture Diagram



Solution Steps

1. Project Setup

1.1. Installation of Jenkins, SonarQube, Maven, and Docker

Use the following command to install the Jenkins in the VM created in pre-requisite

- Sudo apt update && sudo apt upgrade -y
- sudo apt install openjdk-21-jdk -y
- sudo apt install git -y
- sudo apt install maven -y

e. Jenkins Installation:

```
sudo wget -O /etc/apt/keyrings/jenkins-keyring.asc \
https://pkg.jenkins.io/debian-stable/jenkins.io-2023.key
echo "deb [signed-by=/etc/apt/keyrings/jenkins-keyring.asc] \
https://pkg.jenkins.io/debian-stable binary/" | sudo tee \
/etc/apt/sources.list.d/jenkins.list > /dev/null
sudo apt-get update
sudo apt-get install jenkins -y
```

f. Docker Installation:

- sudo apt install docker.io -y
- sudo systemctl enable docker
- sudo systemctl start docker
- sudo usermod -aG docker \$USER
- sudo usermod -aG docker jenkins
- g. Sudo systemctl restart docker
- h. Sudo reboot
- i. **Login to VM:**
 - ssh -i "key.pem" azureuser@<public-ip>
- j. **SonarQube Installation:**
 - docker volume create --name sonarqube_data
 - docker volume create --name sonarqube_logs
 - docker volume create --name sonarqube_extensions
 - docker run -d --name sonarqube \
 -p 9000:9000 \
 -v sonarqube_data:/opt/sonarqube/data \
 -v sonarqube_extensions:/opt/sonarqube/extensions \
 -v sonarqube_logs:/opt/sonarqube/logs \
 sonarqube:its-community

```

azureuser@jenkinsVM:~$ java --version
openjdk 21.0.6 2025-01-21
OpenJDK Runtime Environment (build 21.0.6+7-Ubuntu-122.04.1)
OpenJDK 64-Bit Server VM (build 21.0.6+7-Ubuntu-122.04.1, mixed mode, sharing)
azureuser@jenkinsVM:~$ mvn --version
Apache Maven 3.6.3
Maven home: /usr/share/maven
Java version: 21.0.6, vendor: Ubuntu, runtime: /usr/lib/jvm/java-21-openjdk-amd64
Default locale: en, platform encoding: UTF-8
OS name: "linux", version: "6.8.0-1027-azure", arch: "amd64", family: "unix"
azureuser@jenkinsVM:~$ jenkins --version
2.504.1
azureuser@jenkinsVM:~$ sudo systemctl status docker
● docker.service - Docker Application Container Engine
   Loaded: loaded (/lib/systemd/system/docker.service; enabled; vendor preset: enabled)
   Active: active (running) since Sat 2025-05-03 17:28:40 UTC; 2min 42s ago
     TriggeredBy: ● docker.socket
   Docs: https://docs.docker.com
   Main PID: 823 (dockerd)
   Tasks: 22
azureuser@jenkinsVM:~$ docker ps
CONTAINER ID   IMAGE                                COMMAND                  CREATED        STATUS        PORTS                               NAMES
25a4bb7c585e   sonarqube:its-community             "/opt/sonarqube/dock..." 2 minutes ago  Up 2 minutes  0.0.0.0:9000->9000/tcp, :::9000->9000/tcp  sonarqube

```

1.2. Configure Jenkins

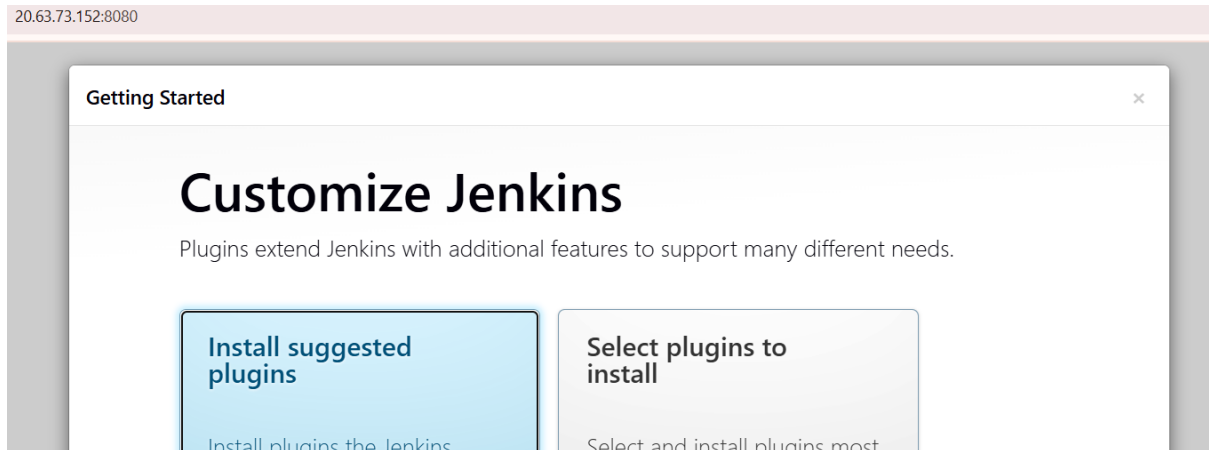
- a. Configure the Jenkins for the first time by launching the Jenkins URL: <http://<public-ip>:8080>



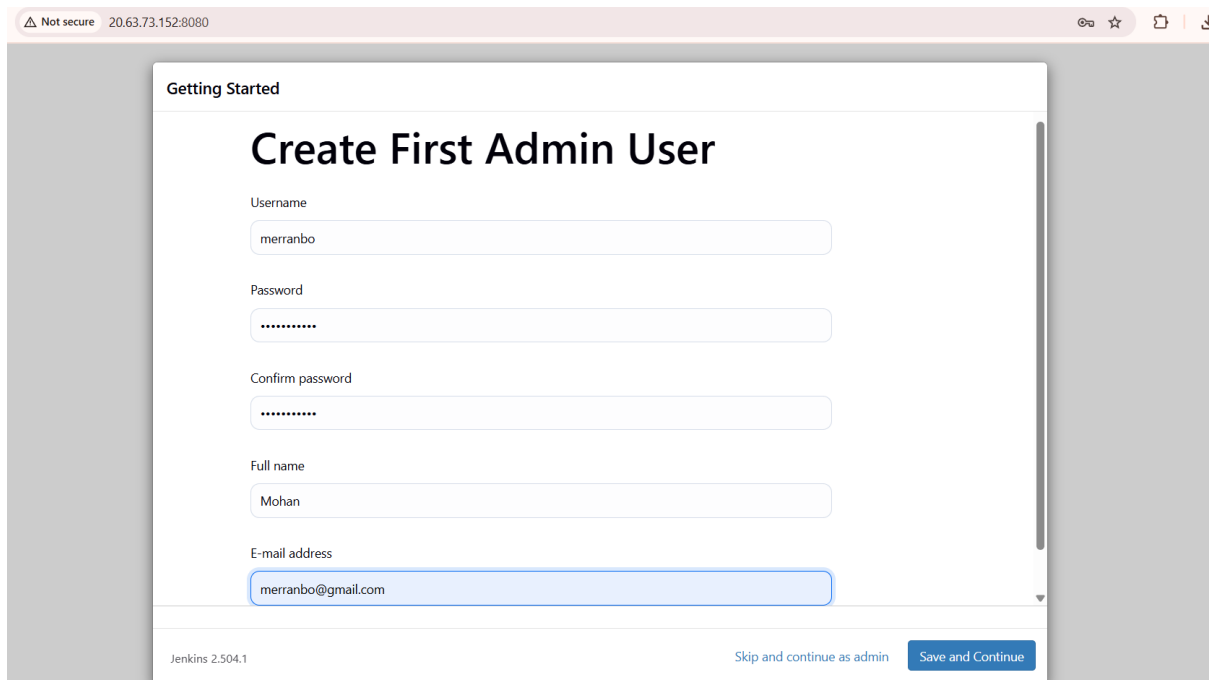
- b. Use the below command to retrieve the password for Jenkins
Command: `sudo cat /var/lib/jenkins/secrets/initialAdminPassword`
Password: 2ef14ed435934809b329f20813b77005

```
azureuser@JenkinsVM:~$ sudo cat /var/lib/jenkins/secrets/initialAdminPassword
2ef14ed435934809b329f20813b77005
azureuser@JenkinsVM:~$
```

- c. Create a new user with new password and install the required plugins



- d. Verify the Jenkins dashboard page is displayed



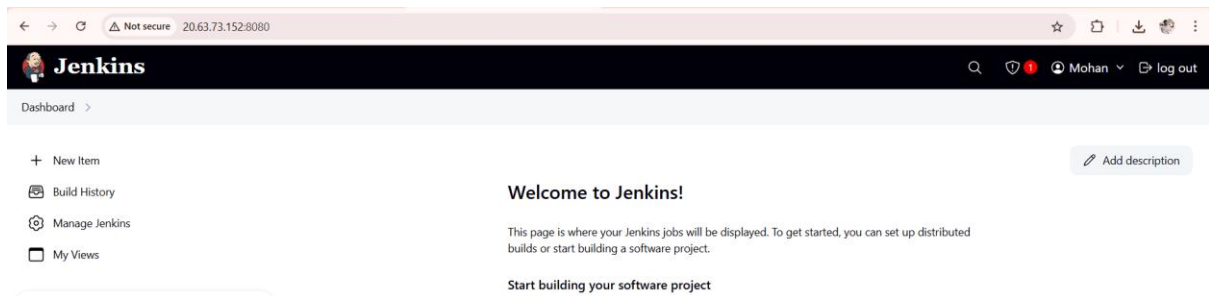
Instance Configuration

Jenkins URL:

`http://20.63.73.152:8080/`

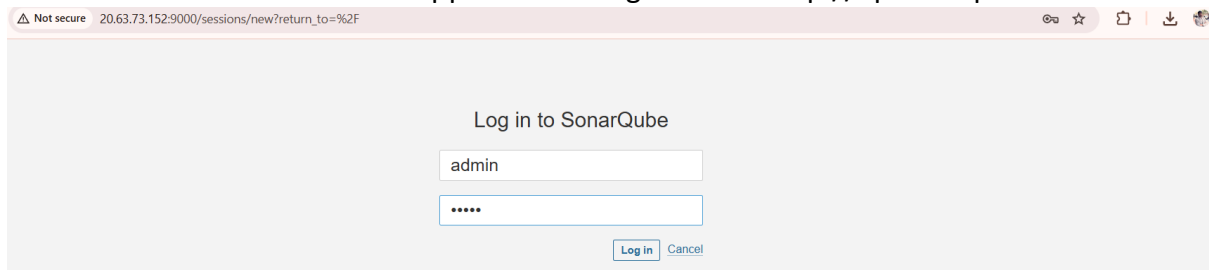
The Jenkins URL is used to provide the root URL for absolute links to various Jenkins resources. That means this value is required for proper operation of many Jenkins features including email notifications, PR status updates, and the `BUILD_URL` environment variable provided to build steps.

The proposed default value shown is **not saved yet** and is generated from the current request, if possible. The best practice is to set this value to the URI that users are expected to use. This will avoid confusion when sharing or viewing links.

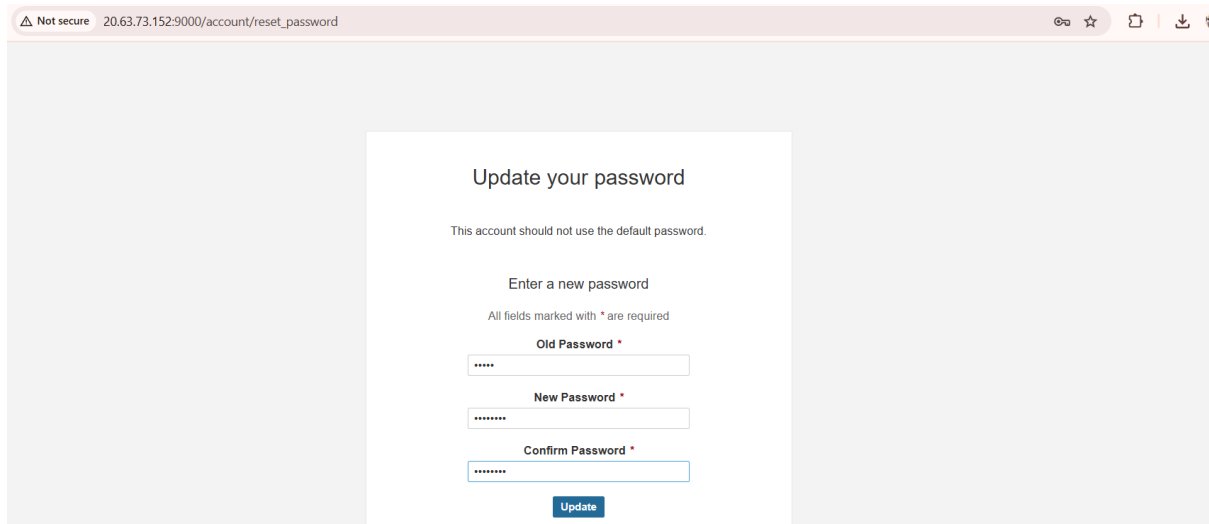


1.3.Configure SonarQube

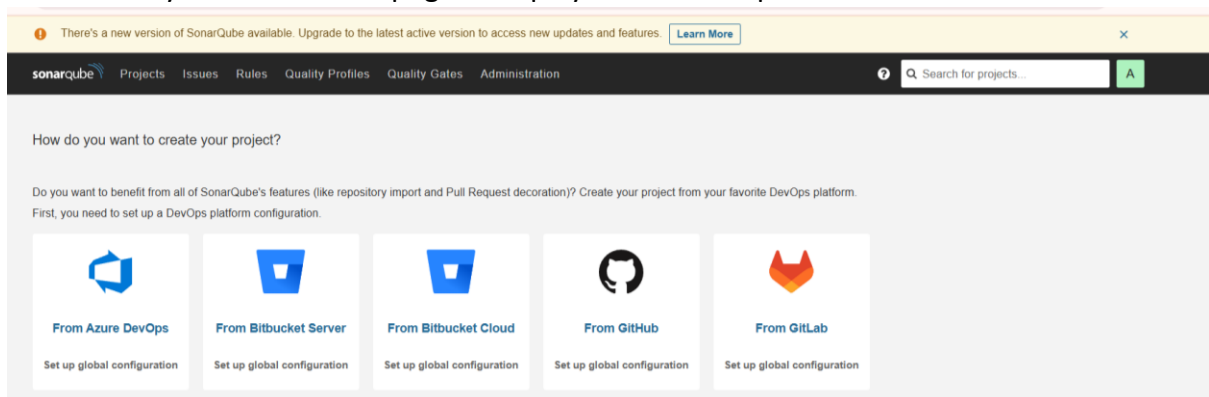
- Launch the SonarQube application using the URL: `http://<public-ip>:9000`



- Give the initial username and password as below:
Username: admin
Password: admin
- When prompted provide a new password and proceed.



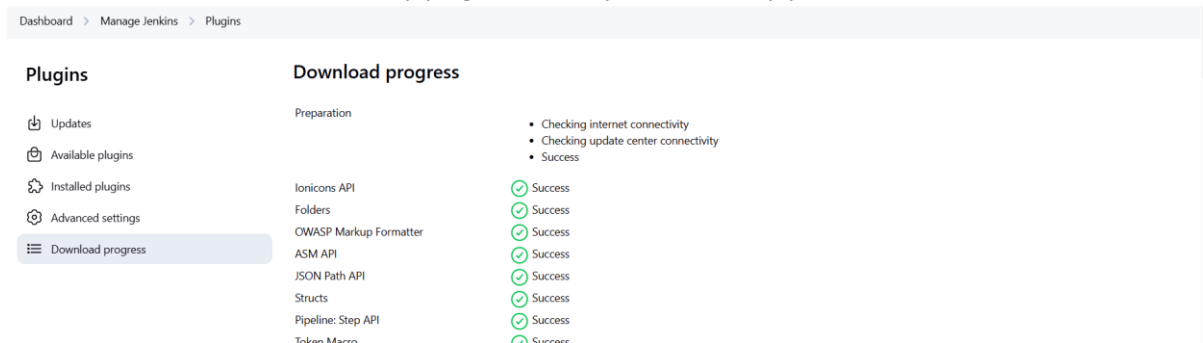
- Verify the dashboard page is displayed for sonarqube.



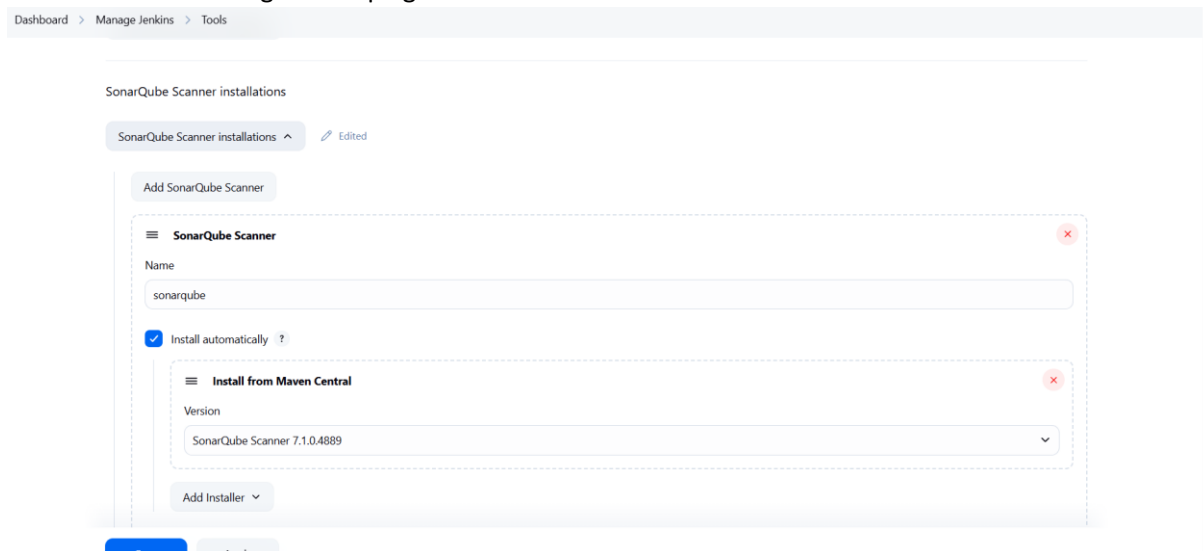
2. Continuous Integration

2.1.Jenkins configuration

a. Add the necessary plugins to set up the Jenkins pipeline

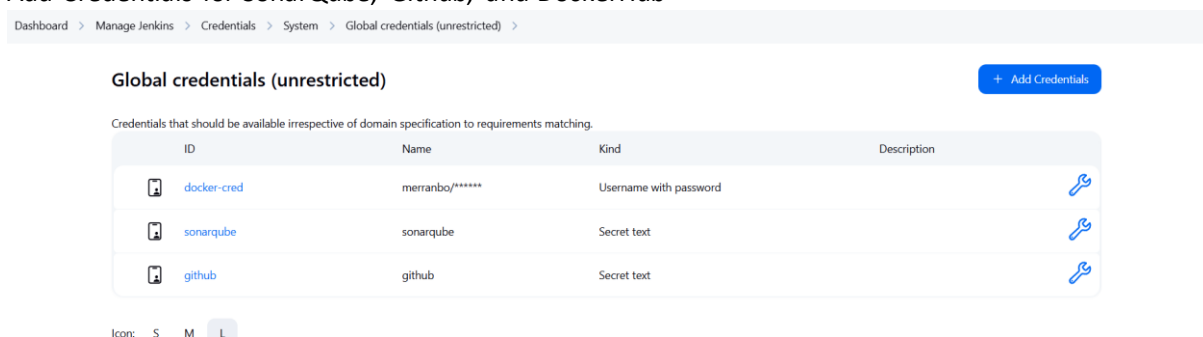


b. Configure the plugins



2.2.Credential Management

Add Credentials for SonarQube, Github, and DockerHub



2.3.Pipeline Creation and Execution

a. Create a New Pipeline project

Dashboard > BCPP5 > Configuration

Configure

- General
- Triggers
- Pipeline
- Advanced

General

Enabled ☒

Description

Pipeline for Continuous Integration

Plain text Preview

☐ Discard old builds ?

b. Use Pipeline from SCM option for pipeline script

Configure

General
Triggers
Pipeline
Advanced

Pipeline

Define your Pipeline using Groovy directly or pull it from source control.

Definition

Pipeline script from SCM

SCM ?

Git

Repositories ?

Repository URL ?

https://github.com/merranbo1989/BCP-P5

P.S. use the below repo <https://github.com/merranbo1989/BCP-P5.git> and the script path as “java-maven-sonar-argocd-helm-k8s/spring-boot-app/JenkinsFile”

***** **JenkinsFile code starts here** *****

```

pipeline {
  agent any
  stages {
    stage('Checkout') {
      steps {
        // Checkout code from Git repository
        checkout scmGit(branches: [[name: '*/main']], extensions: [], userRemoteConfigs: [[url:
'https://github.com/merranbo1989/BCP-P5.git']])
      }
    }
    stage('Build and Test') {
      steps {
        sh 'ls -ltr'
        // build the project and create a JAR file
        sh 'cd java-maven-sonar-argocd-helm-k8s/spring-boot-app && mvn clean package'
      }
    }
    stage('Static Code Analysis') {
      environment {
        SONAR_URL = "http://20.63.73.152:9000/"
      }
      steps {
        withCredentials([string(credentialsId: 'sonarqube', variable: 'SONAR_AUTH_TOKEN')]) {
          sh 'cd java-maven-sonar-argocd-helm-k8s/spring-boot-app && mvn sonar:sonar -
Dsonar.login=$SONAR_AUTH_TOKEN -Dsonar.host.url=${SONAR_URL}'
        }
      }
    }
  }
}

```

```

}
stage('Build and Push Docker Image') {
  environment {
    DOCKER_IMAGE = "merranbo/bcpp5:${BUILD_NUMBER}"
    // DOCKERFILE_LOCATION = "java-maven-sonar-argocd-helm-k8s/spring-boot-app/Dockerfile"
    REGISTRY_CREDENTIALS = credentials('docker-cred')
  }
  steps {
    script {
      sh 'cd java-maven-sonar-argocd-helm-k8s/spring-boot-app && docker build -t
${DOCKER_IMAGE} .'
      def dockerImage = docker.image("${DOCKER_IMAGE}")
      docker.withRegistry('https://index.docker.io/v1/', "docker-cred") {
        dockerImage.push()
      }
    }
  }
}
stage('Update Deployment File') {
  environment {
    GIT_REPO_NAME = "BCP-P5"
    GIT_USER_NAME = "merranbo1989"
  }
  steps {
    withCredentials([string(credentialsId: 'github', variable: 'GITHUB_TOKEN')]) {
      sh '''
        git config user.email "merranbo1989@gmail.com"
        git config user.name "merranbo1989"
        BUILD_NUMBER=${BUILD_NUMBER}
        sed -i "s/replaceImageTag/${BUILD_NUMBER}/g" java-maven-sonar-argocd-helm-
k8s/spring-boot-app-manifests/deployment.yml
        git add java-maven-sonar-argocd-helm-k8s/spring-boot-app-manifests/deployment.yml
        git commit -m "Update deployment image to version ${BUILD_NUMBER}"
        git push
        https://${GITHUB_TOKEN}@github.com/${GIT_USER_NAME}/${GIT_REPO_NAME} HEAD:main
        '''
    }
  }
}
}
}
}
}
}
***** Code Ends Here *****

```

- c. Click on “Build now” option to run the pipeline

Dashboard > BCPP5 >

Status **BCPP5** Edit description

</> Changes
 ▶ Build Now
 ⚙️ Configure
 🗑️ Delete Pipeline
 🔍 Full Stage View
 📁 Stages
 ✎️ Rename
 ⓘ Pipeline Syntax
 📖 GitHub Hook Log

Pipeline for Continuous Integration

Stage View

Average stage times: (full run time: ~1min 14s)

	Declarative: Checkout SCM	Checkout	Build and Test	Static Code Analysis	Build and Push Docker Image	Update Deployment File
May 03 18:11	2s	793ms	18s	27s	18s	1s

Builds

Filter

Today

✓ #1 10:11 PM

Permalinks

- Last build (#1), 9 min 1 sec ago
- Last stable build (#1), 9 min 1 sec ago
- Last successful build (#1), 9 min 1 sec ago
- Last completed build (#1), 9 min 1 sec ago

d. Check the SonarQube for the code analysis result

sonarqube Projects Issues Rules Quality Profiles Quality Gates Administration

Search for projects...

My Favorites All

Search by project name or key

1 project(s) Perspective: Overall Status Sort by: Name

spring-boot-demo Passed Last analysis: 1 minute ago

Bugs	Vulnerabilities	Hotspots Reviewed	Code Smells	Coverage	Duplications	Lines
0	0	-	0	0.0%	0.0%	79

1 of 1 shown

e. Check the Docker Hub to see if the Image is created

docker:desktop PERSONAL

Images Give feedback

View and manage your local and Docker Hub images. Learn more

Local Docker Hub repositories

merranbo Search

Tags	OS	Vulnerabilities	Last pushed	Size
merranbo/bcpp5	1	Inactive	2 minutes ago	70.71 MB

View Docker Scout dashboard

View in Docker Hub Pull

f. Verify that the “Kubernetes” Manifest file “deployment.yml” in the github path

<https://github.com/merranbo1989/BCP-P5/tree/main/java-maven-sonar-argocd-helm-k8s/spring-boot-app-manifests>

BCP-P5 / java-maven-sonar-argocd-helm-k8s / spring-boot-app-manifests /

Add file

merranbo1989 Update deployment image to version 1 e397f46 · 2 minutes ago History

Name	Last commit message	Last commit date
..		
deployment.yml	Update deployment image to version 1	2 minutes ago
service.yml	First Commit	18 hours ago

```

15     app: spring-boot-app
16   spec:
17     containers:
18     - name: spring-boot-app
19       image: merranbo/bcpp5:1
20     ports:
21     - containerPort: 8080

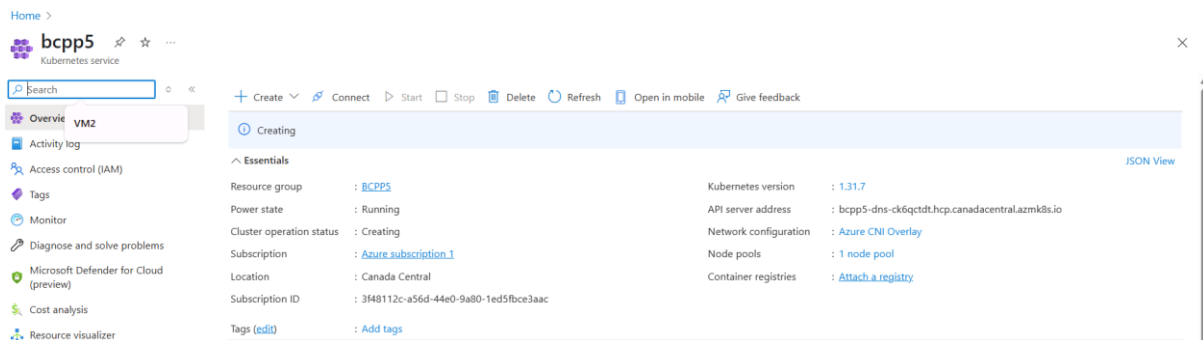
```

P.S. The updated image name will serve as the input for the CD stage using Argo CD

3. Argo CD

3.1.AKS Cluster Installation

a. Create an AKS cluster in Azure Portal



3.2.Install ArgoCD Operator and controller on AKS cluster

a. Follow the below commands to install the ArgoCD on AKS cluster

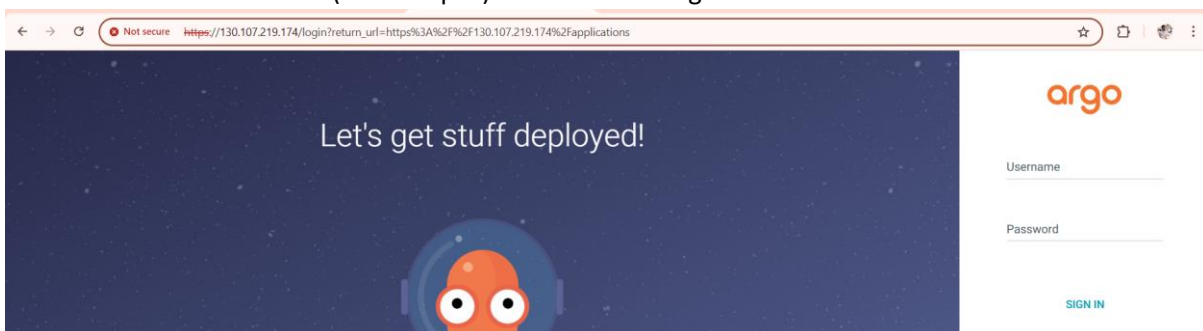
- i. az account set --subscription "<subscription name>"
- ii. az aks get-credentials --resource-group <resource group name> --name <AKS cluster name>
- iii. kubectl create namespace argocd
- iv. kubectl apply -n argocd -f https://raw.githubusercontent.com/argoproj/argo-cd/stable/manifests/install.yaml
- v. kubectl patch svc argocd-server -n argocd -p '{"spec": {"type": "LoadBalancer"}}'
- vi. kubectl get svc argocd-server -n argocd

```

mohanramrajan [ ~ ]$ kubectl get svc argocd-server -n argocd
NAME          TYPE          CLUSTER-IP    EXTERNAL-IP    PORT(S)                                     AGE
argocd-server LoadBalancer  10.0.230.148   130.107.219.174  80:31092/TCP,443:32071/TCP                4h59m
mohanramrajan [ ~ ]$

```

b. Use the external IP (from step vi) to launch the ArgoCD UI



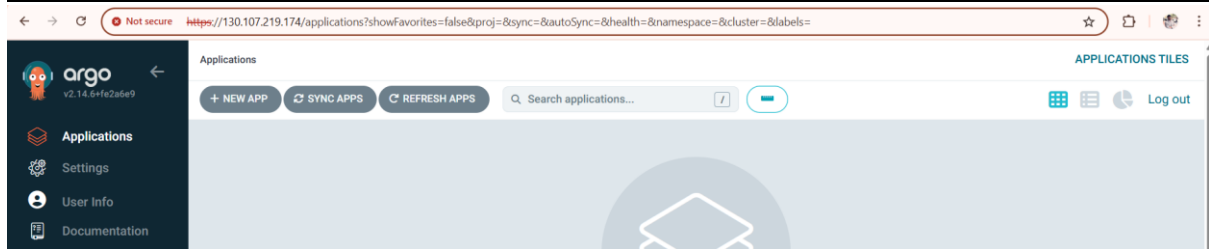
c. To get the initial password use the below command

- i. kubectl get secret argocd-initial-admin-secret -n argocd -o jsonpath="{.data.password}" | base64 --decode

Username: admin

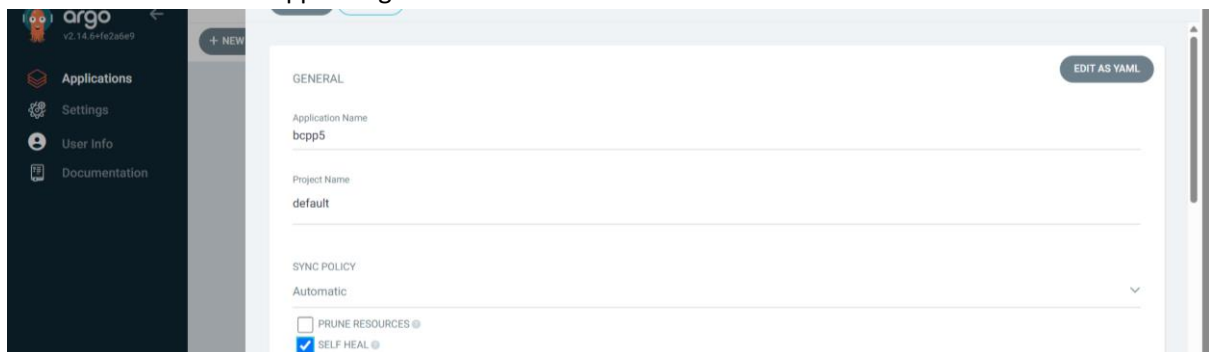
Password: DI97OjBRtuhhvChg

```
mohanramrajan [ ~ ]$ kubectl get secret argocd-initial-admin-secret -n argocd -o jsonpath="{.data.password}" | base64 --decode  
DI97OjBRtuhhvChgmohanramrajan [ ~ ]$
```

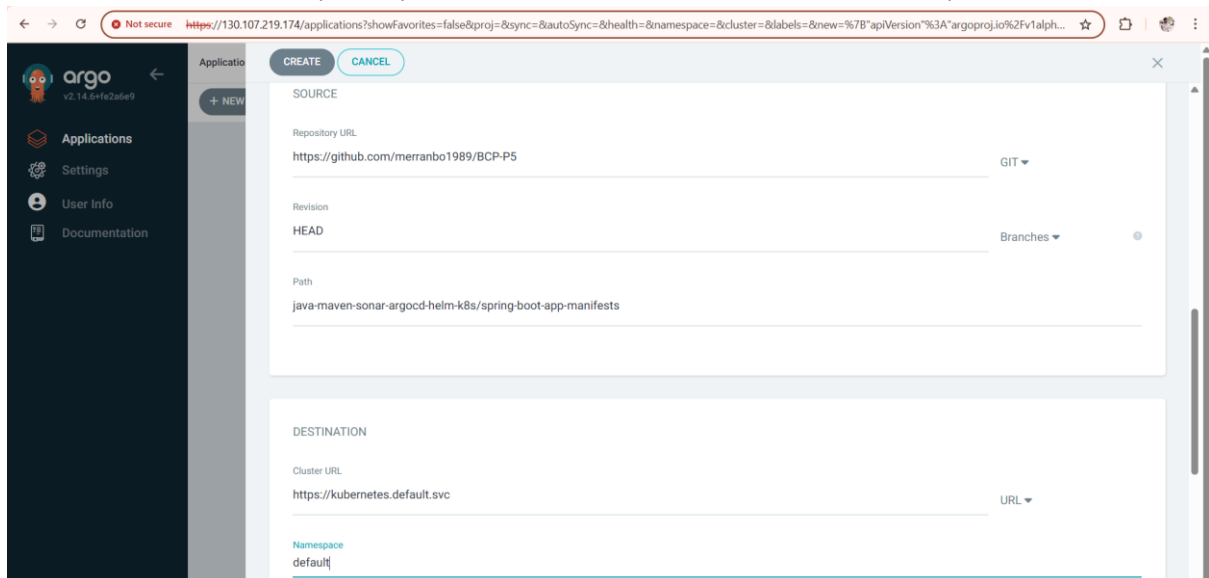


3.3. Configure ArgoCD APP

a. Create a New App in ArgoCD UI.



b. Provide the Git Repository details where the Kubernetes Manifests are placed



c. Click on Create option

The top screenshot shows the Argo CD application details for 'bcpp5'. The application is in a 'Progressing' state, synced to the HEAD of the repository. The repository is 'https://github.com/merranbo1989/BCP-...'. The target is 'HEAD'. The path is 'java-maven-sonar-argocd-helm-k8s/spri...'. The destination is 'in-cluster'. The namespace is 'default'. The application was created on 05/04/2025 at 17:13:28 and last synced on 05/04/2025 at 17:13:29.

The bottom screenshot shows the Argo CD application details tree for 'bcpp5'. The application is 'Healthy' and 'Synced to HEAD (8284014)'. The last sync was successful on Sun May 04 2025 17:13:29 GMT+0400. The application details tree shows the deployment of 'spring-boot-app' to 'spring-boot-app-service' and 'spring-boot-app-service-sg5pb'.

d. Use the below command to check if the pods are running

- Kubectl get nodes
- Kubectl get pods
- Kubectl get svc

```
mohanramrajan [ ~ ]$ kubectl get nodes
NAME                                STATUS    ROLES    AGE   VERSION
aks-agentpool-29289524-vmss000000  Ready    <none>   5h20m v1.31.7
aks-agentpool-29289524-vmss000001  Ready    <none>   5h20m v1.31.7
mohanramrajan [ ~ ]$ kubectl get pods
NAME                                READY    STATUS    RESTARTS   AGE
spring-boot-app-6d7f59694c-45bq1    1/1      Running   0           111s
spring-boot-app-6d7f59694c-k6vwk    1/1      Running   0           111s
mohanramrajan [ ~ ]$ kubectl get svc
NAME            TYPE          CLUSTER-IP    EXTERNAL-IP    PORT(S)          AGE
kubernetes      ClusterIP     10.0.0.1      <none>         443/TCP          5h21m
spring-boot-app-service  LoadBalancer 10.0.255.52   130.107.225.95  80:30452/TCP     119s
mohanramrajan [ ~ ]$
```

Validation

Use the External-IP to launch the application

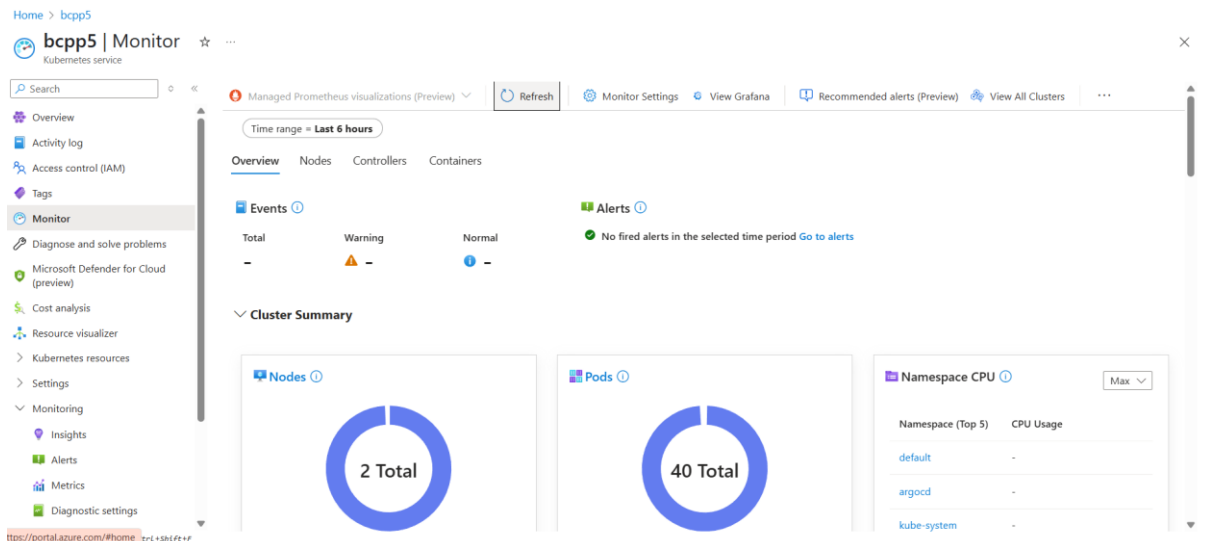


I have successfully built a sprint boot application using Maven

This application is deployed on to Kubernetes (AKS Cluster) using Argo CD and Jenkins pipeline

Monitoring

- Enable Prometheus in Azure AKS cluster



- Enable Grafana

CIArmTemplateOnboardingDeployment-50d01293-0346-41b5-81b9-2756c0 | Overview

Deployment

arch

Delete Cancel Redeploy Download Refresh

Overview

Deployment is in progress

Deployment name : CIArmTemplateOnboardingDeployment-50d01293-0... Start time : 5/4/2025, 10:01:21 PM

Subscription : Azure subscription 1 Correlation ID : 5c97d11b-8392-494c-bfa3-ab767b6ee086

Resource group : bcpp5

Microsoft Defender for Cloud

Secure your apps and infras

Home > bcpp5

bcpp5 | Monitor

Managed Prometheus visualizations (Preview)

Time range = Last 6 hours

Overview Nodes Controllers Containers

Events

Total Warning Normal

Cluster Summary

Nodes

2 Total

Configure Azure monitor for containers

Managed Prometheus

Managed Prometheus provides a highly available, scalable and secure metrics platform to monitor your containerized workloads. Learn more

Azure Monitor workspace

defaultazuremonitorworkspace-cca

Managed Grafana

Selecting a fully managed instance of Grafana to visualize your managed Prometheus data stored in your Azure Monitor workspace. Learn more about pricing

Enable Grafana

A Grafana Instance (grafana-20254022054-cca) will be created automatically.

Container logs

Container logs collect container logs, Kubernetes events, and cluster inventory data with the Azure Monitor agent to enable debugging your applications. Learn more

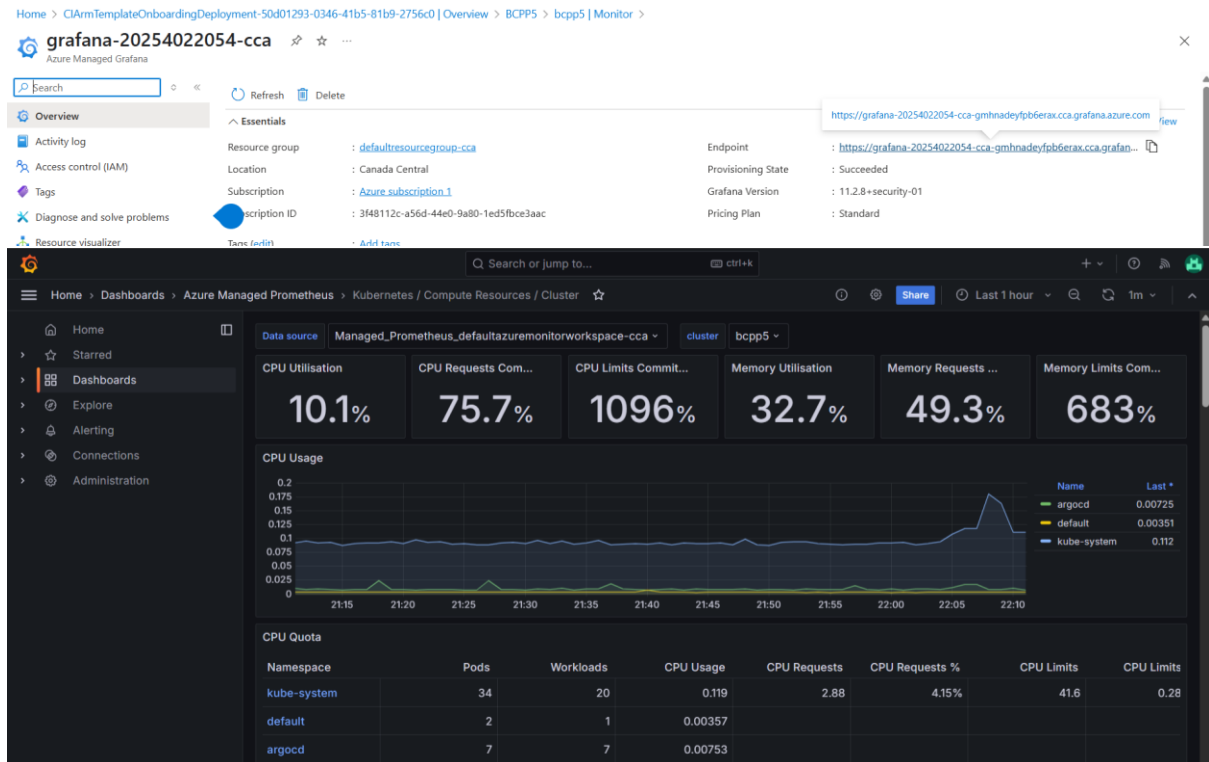
Enable container logs

A Log Analytics workspace (defaultworkspace-3f48112c-a56d-44e0-9a80-1ed5fbce3aac-cca) will be created automatically.

Advanced settings

Configure Cancel

- View Grafana chart



Conclusion

This project successfully demonstrates a modern, cloud-native DevOps workflow tailored for enterprise Java applications. By combining CI/CD automation, GitOps deployment, and Kubernetes observability, it addresses critical challenges in software delivery—speed, reliability, and visibility. The architecture and tooling are designed to scale across teams and environments, making this solution well-suited for organizations embracing DevOps and Kubernetes. Moreover, the implementation of proactive monitoring ensures system health is continuously assessed, allowing teams to detect and resolve issues before they impact users. This project sets a strong foundation for future enhancements such as security scanning, centralized logging, and policy-as-code integrations.

APPENDIX

A. Create WebHooks in Github Repository to automatically trigger the Jenkins pipeline job

URL: <https://github.com/merranbo1989/BCP-P5>

General

Access

Collaborators

Moderation options

Code and automation

Branches

Tags

Rules

Actions

Webhooks

Environments

Codespaces

Pages

Security

Webhooks / Manage webhook

SettingsRecent Deliveries

We'll send a POST request to the URL below with details of any subscribed events. You can also specify which data format you'd like to receive (JSON, x-www-form-urlencoded, etc). More information can be found in [our developer documentation](#).

Payload URL *
http://20.220.33.14:8080/github-webhook/

Content type *
application/x-www-form-urlencoded

There is currently a secret configured for this webhook. If you've lost or forgotten this secret, you can change it, but be aware that any integrations using this secret will need to be updated.

Change secret

General

Access

Collaborators

Moderation options

Code and automation

Branches

Webhooks

Add webhook

Webhooks allow external services to be notified when certain events happen. When the specified events happen, we'll send a POST request to each of the URLs you provide. Learn more in our [Webhooks Guide](#).

✓ http://20.220.33.14:8080/github-we... (push)

EditDelete

Last delivery was successful.