



Bootcamp-Project 2

DEPLOY A CONTAINERIZED APPLICATION TO AKS USING AZURE
DEVOPS USING HELM CHART

Mohanramrajan Erran Bothalraj

Table of Contents

Introduction	2
Project Objectives	2
Expected Outcome	2
Pre-Requisite	2
Solution Summary	3
Step 1 – Containerize the Application	3
Step 2 – Create the CI Pipeline	4
Step 3 – Validation	7
Step 4 – Azure Monitor with Prometheus and Grafana	8

Introduction

This project demonstrates a **complete CI/CD pipeline** setup to build, deploy, and monitor a containerized .net web application using **Azure DevOps, Azure Kubernetes Service (AKS), and Helm**. The pipeline automates the process from code commit to live deployment, ensuring fast, reliable, and scalable delivery. Additionally, **Azure Monitor and Prometheus** are integrated for observability and health monitoring of the deployed services.

Project Objectives

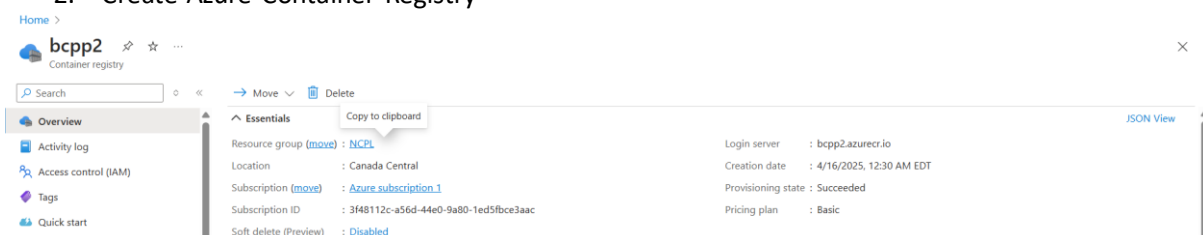
- 1. Containerize the Application:**
 - Use Docker to build a container image of the application.
 - Configure the Dockerfile for production-ready deployment.
- 2. Set Up a CI Pipeline (Build Pipeline):**
 - Automatically trigger on code commits.
 - Build the Docker image.
 - Tag and push the image to ACR.
- 3. Set Up a CD Pipeline (Release Pipeline):**
 - Use Helm charts to define Kubernetes resources.
 - Deploy the application to AKS using the latest image from ACR.
 - Update Kubernetes workloads seamlessly.
- 4. Monitoring and Observability:**
 - Enable Azure Monitor for real-time logs and metrics.
 - Deploy Prometheus (via Helm) to monitor application-specific metrics.

Expected Outcome

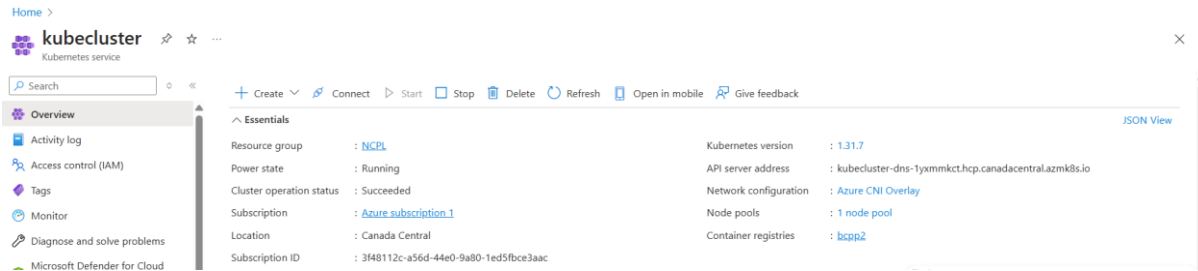
A fully automated CI/CD pipeline that builds, packages, and deploys a containerized application to **Azure Kubernetes Service (AKS) using Azure DevOps and Helm Chart**. The deployment is scalable, version-controlled, and monitored in real time using **Azure Monitor and Prometheus**—ensuring faster releases, improved reliability, and greater visibility into application health.

Pre-Requisite

1. Git/Azure Repos with Dockerfile configured for the .net application:
<https://github.com/merranbo1989/BCP-P2.git>
2. Create Azure Container Registry



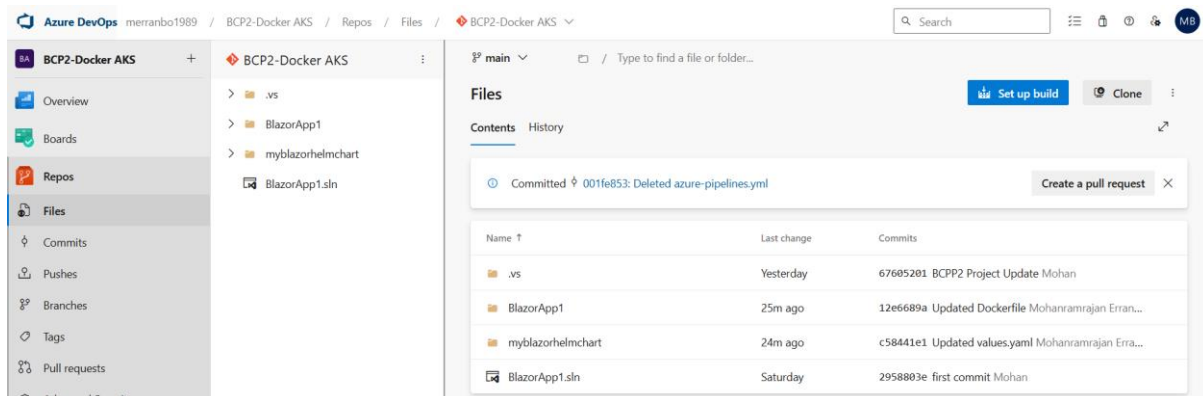
3. Create Azure Kubernetes Cluster



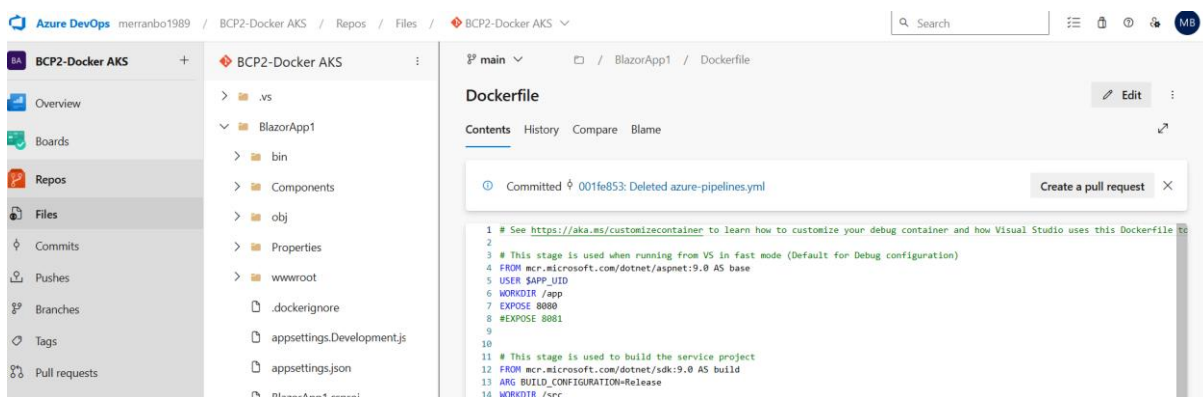
Solution Summary

Step 1 – Containerize the Application

A. Upload the .net application code into Azure Repos



B. Configure the Dockerfile for the .net application



***** Dockerfile Code Starts *****

```
FROM mcr.microsoft.com/dotnet/aspnet:9.0 AS base
USER $APP_UID
WORKDIR /app
EXPOSE 8080
#EXPOSE 8081
```

```
# This stage is used to build the service project
```

```
FROM mcr.microsoft.com/dotnet/sdk:9.0 AS build
ARG BUILD_CONFIGURATION=Release
WORKDIR /src
COPY ["BlazorApp1.csproj", "."]
RUN dotnet restore "./BlazorApp1.csproj"
COPY . .
WORKDIR "/src/"
RUN dotnet build "./BlazorApp1.csproj" -c $BUILD_CONFIGURATION -o /app/build
```

This stage is used to publish the service project to be copied to the final stage

```
FROM build AS publish
ARG BUILD_CONFIGURATION=Release
RUN dotnet publish "./BlazorApp1.csproj" -c $BUILD_CONFIGURATION -
o /app/publish /p:UseAppHost=false
```

This stage is used in production or when running from VS in regular mode (Default when not using the Debug configuration)

```
FROM base AS final
WORKDIR /app
COPY --from=publish /app/publish .
ENTRYPOINT ["dotnet", "BlazorApp1.dll"]
```

***** Code ends here *****

Step 2 – Create the CI Pipeline

- A. Setup the CI/CD build to build & push the Docker Image to ACR

✓ Connect ✓ Select ✓ Configure **Review**

New pipeline

Review your pipeline YAML Variables Save and run

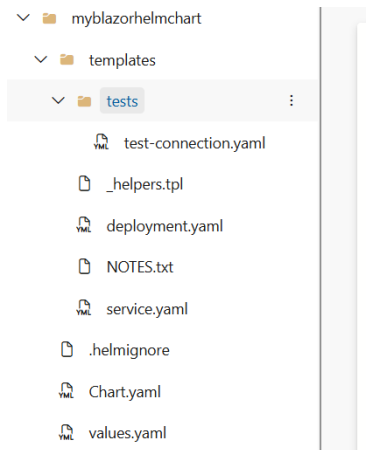
BCP2-Docker AKS / azure-pipelines.yml

```
1  # Docker
2  # Build and push an image to Azure Container Registry
3  # https://docs.microsoft.com/azure/devops/pipelines/languages/docker
4
5  trigger:
6  - main
7
8  resources:
9  - repo: self
10
11 variables:
12   - # Container registry service connection established during pipeline creation
13     - dockerRegistryServiceConnection: '68cf6555-47b6-49fc-a1e3-e0216fbc3c29'
14     - imageRepository: 'blazorapp1'
15     - containerRegistry: 'bcpp2.azurecr.io'
16     - dockerfilePath: '$(Build.SourcesDirectory)/BlazorApp1/Dockerfile'
```

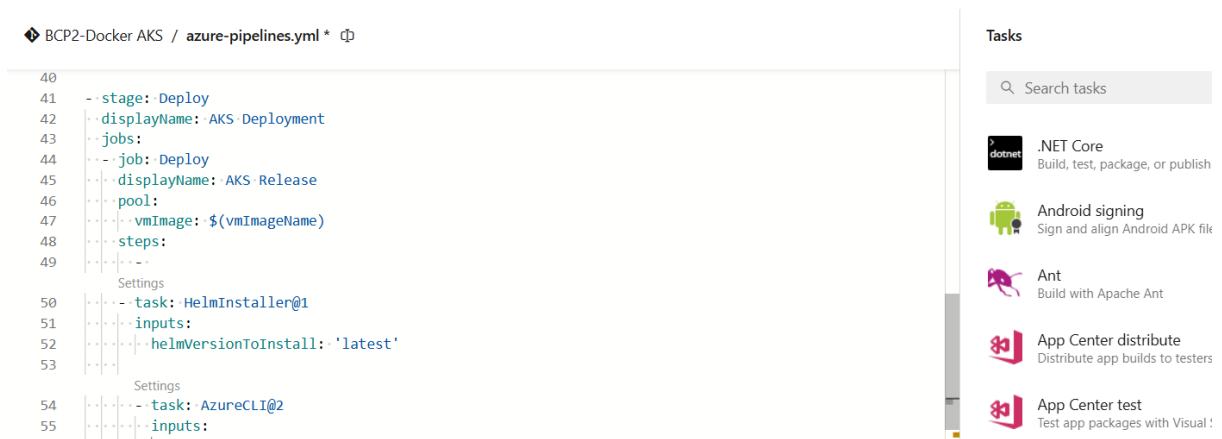
Tasks

- .NET Core
Build, test, package, or publish a dotnet application...
- Android signing
Sign and align Android APK files
- Ant
Build with Apache Ant
- App Center distribute
Distribute app builds to testers and users via Visu...

- B. Configure the Helm Chart to deploy into AKS.



C. Setup the CD pipeline to deploy the Docker image from ACR to AKS



***** Code for azure-pipeline.yml *****

trigger:

- master

resources:

- repo: self

variables:

Container registry service connection established during pipeline creation

dockerRegistryServiceConnection: '1da2fadb-236f-4fe3-9b9d-1257a81b20d0'

azureSubscription: '3f48112c-a56d-44e0-9a80-1ed5fbce3aac'

imageRepository: 'blazorapp1'

containerRegistry: 'bcpp2.azurecr.io'

dockerfilePath: '\$(Build.SourcesDirectory)/BlazorApp1/Dockerfile'

tag: 'latest'

Agent VM image name

vmImageName: 'ubuntu-latest'

stages:

- stage: Build

displayName: ACR Docker Build

jobs:

```

- job: Build
  displayName: Build Docker Image
  pool:
    vmImage: $(vmImageName)
  steps:

- task: Docker@2
  inputs:
    containerRegistry: 'bcpp2'
    repository: 'blazorapp1'
    command: 'buildAndPush'
    Dockerfile: '**/Dockerfile'
    tags: $(tag)
- stage: Deploy
  displayName: AKS Deployment
  jobs:
  - job: Deploy
    displayName: Deploy to AKS
    pool:
      vmImage: $(vmImageName)
    steps:
      - task: HelmInstaller@1
        inputs:
          helmVersionToInstall: 'latest'

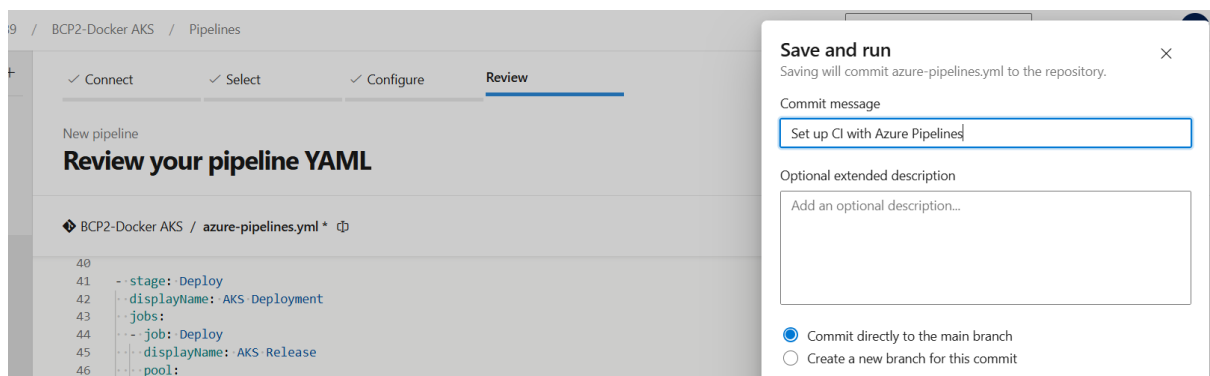
      - task: AzureCLI@2
        inputs:
          azureSubscription: 'Azure subscription 1(3f48112c-a56d-44e0-9a80-1ed5fbce3aac)'
          scriptType: 'bash'
          scriptLocation: 'inlineScript'
          inlineScript: |

            echo "Getting AKS credentials..."
            az aks get-credentials --resource-group NCPL --name kubecuster --overwrite-existing
            echo "Deploying with Helm..."
            helm upgrade --install myblazorapp1 ./myblazorhelmchart

```

***** Code ends here *****

D. Trigger the Pipeline



Azure DevOps merranbo1989 / BCP2-Docker AKS / Pipelines / BCP2-Docker AKS / 20250416.1

#20250416.1 • Update azure-pipelines.yml for Azure Pipelines

Summary Code Coverage

Triggered by Mohanramrajan Erran Bothalraj

Repository and version: BCP2-Docker AKS main 65ddb279

Time started and elapsed: Just now 21s

Related: 0 work items 0 artifacts

Tests and coverage: [Get started](#)

View 12 changes

Warnings 1

See <https://aka.ms/azdo-ubuntu-24.04> for changes to the ubuntu-24.04 image. Some tools (e.g. Mono, NuGet, Terraform) are not available on the image. Therefore some tasks...

ACR Build • Build

Stages Jobs

ACR Build 0/1 completed 19s

Build 19s

AKS Deployment Not started

Azure DevOps merranbo1989 / BCP2-Docker AKS / Pipelines / BCP2-Docker AKS / 20250416.1

Jobs in run #20250416.1 BCP2-Docker AKS

ACR Build

Build 1m 7s

AKS Deployment

AKS Release 18s

Initialize job 1s

Checkout BCP2-Docker ... 1s

HelmInstaller 2s

AzureCLI 12s

Post-job: Checkout BC... <1s

Finalize Job <1s

AKS Release

```

1 ##[warning]See https://aka.ms/azdo-ubuntu-24.04 for changes to the ubuntu-24.04 image. Some tools (e.g. Mono, NuGet, Terraform) are not
2 Pool: Azure Pipelines
3 Image: ubuntu-latest
4 Queued: Just now [manage_parallel_jobs]
5 Agent: Hosted Agent
6 Started: Just now
7 Duration: 18s
8
9 Acquiring an agent from the cloud:
10 This agent cloud does not support more specific status information.
11 Job preparation parameters
12 Job live console data:
13 Starting: AKS Release
14 Async Command Start: DetectDockerContainer
15 Async Command End: DetectDockerContainer
16 Async Command Start: DetectDockerContainer
17 Async Command End: DetectDockerContainer
18 Finishing: AKS Release

```

Step 3 – Validation

A. Check the ACR in Azure to see if the image is available

Home > Container registries > bcpp2 | Repositories

bcpp2 | Repositories

Repository

blazorapp1

Refresh Start artifact streaming Manage deleted artifacts Delete repository

Essentials

Repository: blazorapp1 Tag count: 1

Last updated date: 4/16/2025, 12:55 AM EDT Manifest count: 1

Search to filter tags ...

Tags	Digest	Last modified
latest	sha256:4a45377ddd10e4877ef2355a362fe4ad...	4/16/2025, 12:55 AM EDT

B. Check the AKS to see if the deployment is available

Access control (IAM)

Filter by namespace: All namespaces Service name: All Add label filter

Name	Namespace	Status	Type	Cluster IP	External IP	Ports	Age
kubernetes	default	Ok	ClusterIP	10.0.0.1		443/TCP	26 minutes
kube-dns	kube-system	Ok	ClusterIP	10.0.0.10		53/UDP,53/TCP	25 minutes
metrics-server	kube-system	Ok	ClusterIP	10.0.77.99		443/TCP	25 minutes
azure-wi-webhook-webhook-servi...	kube-system	Ok	ClusterIP	10.0.87.43		443/TCP	24 minutes
ama-metrics-kam	kube-system	Ok	ClusterIP	10.0.111.148		8080/TCP	21 minutes
ama-metrics-operator-targets	kube-system	Ok	ClusterIP	10.0.136.48		80/TCP	21 minutes
network-observability	kube-system	Ok	ClusterIP	10.0.243.209		10093/TCP	21 minutes
myblazorapp1-myblazorhelmchart	app1	Ok	LoadBalancer	10.0.79.155	130.107.170.217	8080:32098/TCP	4 minutes

- C. Launch the Azure CLI and verify if the nodes, pods, and services are running and available in the namespace

```
Switch to PowerShell Restart Manage files New session Editor Web preview Settings Help
mohanramrajan [ ~ ]$ kubectl get all -n app1
NAME                                READY   STATUS    RESTARTS   AGE
pod/myblazorapp1-myblazorhelmchart-59d4b885c5-tm45b  1/1     Running   0          5m9s

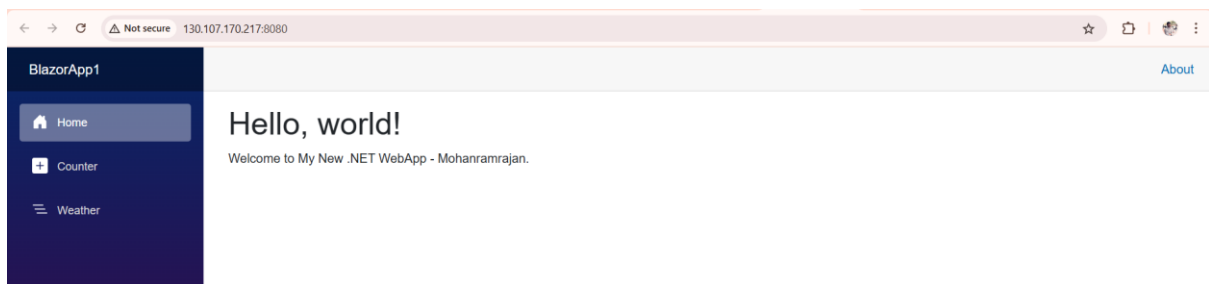
NAME                                TYPE          CLUSTER-IP   EXTERNAL-IP   PORT(S)          AGE
service/myblazorapp1-myblazorhelmchart  LoadBalancer 10.0.79.155   130.107.170.217 8080:32098/TCP   5m9s

NAME                                READY   UP-TO-DATE   AVAILABLE   AGE
deployment.apps/myblazorapp1-myblazorhelmchart  1/1     1            1          5m9s

NAME                                DESIRED   CURRENT   READY   AGE
replicaset.apps/myblazorapp1-myblazorhelmchart-59d4b885c5  1         1         1       5m9s
mohanramrajan [ ~ ]$
```

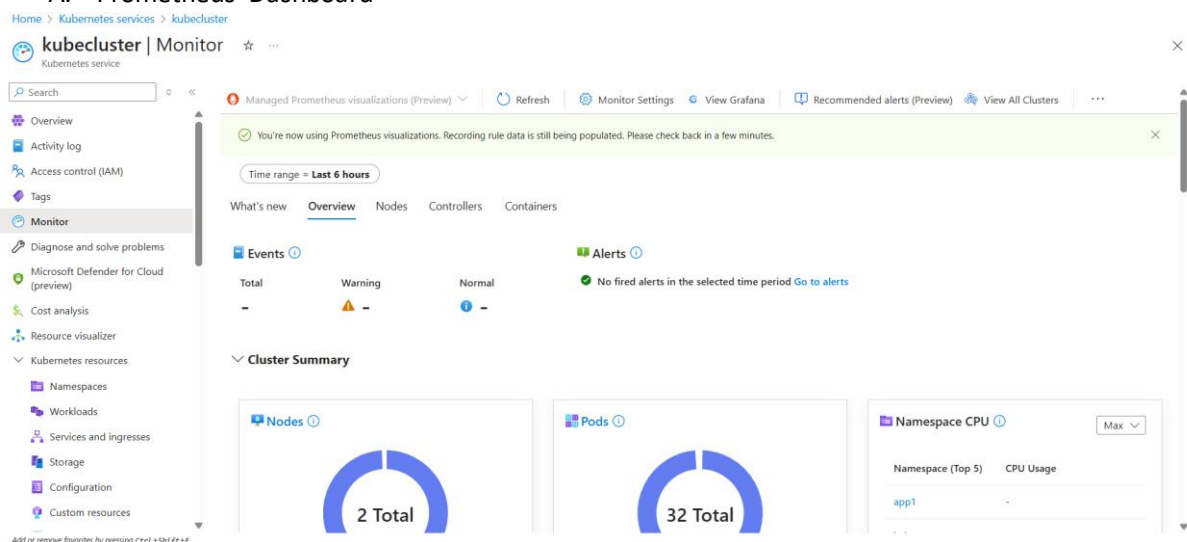
- D. Use the Load Balancer IP address and verify that the application is working as expected.

URL: <http://130.107.170.217:8080>

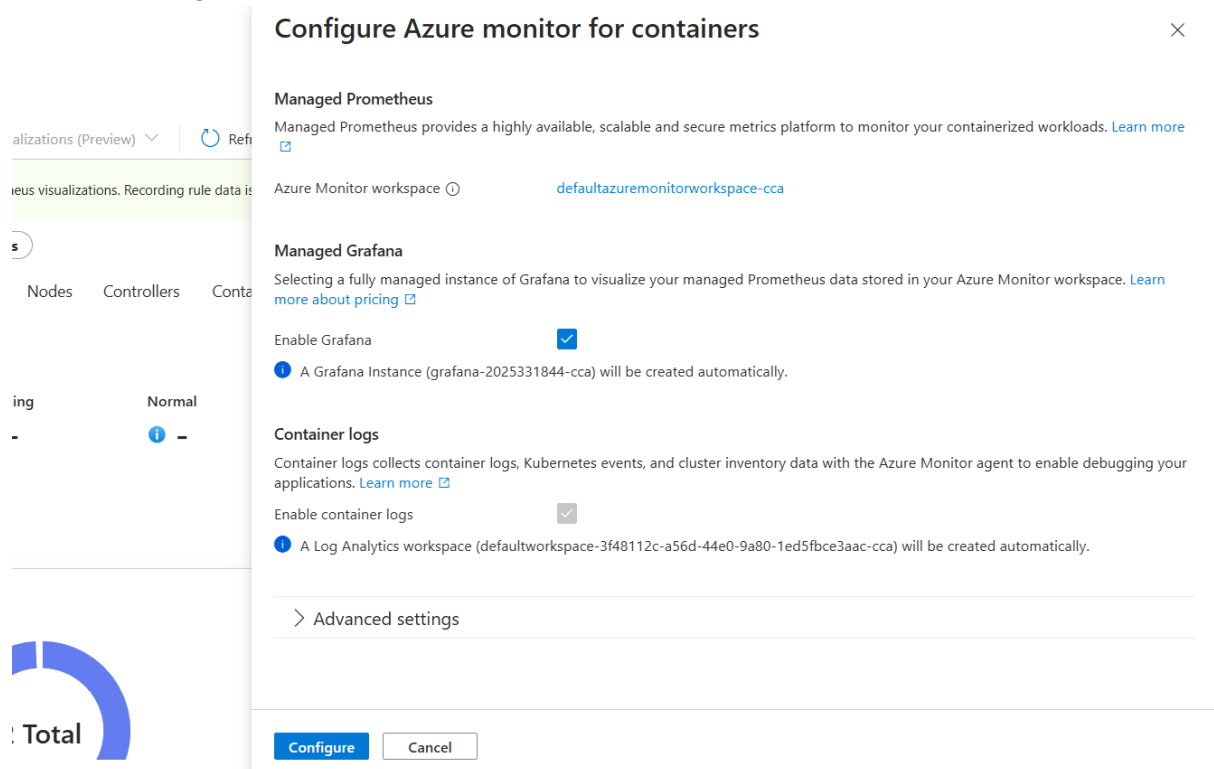


Step 4 – Azure Monitor with Prometheus and Grafana

A. Prometheus Dashboard



B. Enabling Grafana



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-2025331844-cca) will be created automatically.

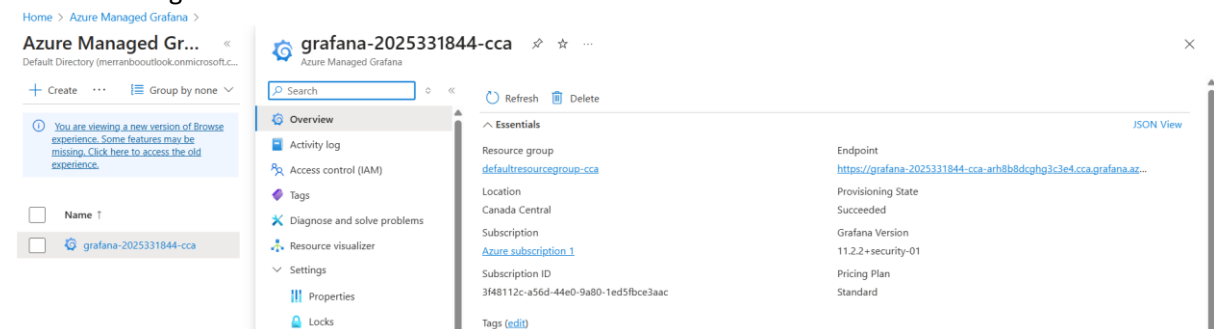
Container logs
Container logs collects container logs, Kubernetes events, and cluster inventory data with the Azure Monitor agent to enable debugging your applications. [Learn more](#)

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

> Advanced settings

Configure **Cancel**

C. Configure Grafana Dashboard



Home > Azure Managed Grafana >

Azure Managed Grafana

Default Directory (merranbootlook.onmicrosoft.c...)

+ Create ... Group by none

You are viewing a new version of Browse experience. Some features may be missing. Click here to access the old experience.

☐ Name ↑

☒ grafana-2025331844-cca

grafana-2025331844-cca

Overview

Activity log

Access control (IAM)

Tags

Diagnose and solve problems

Resource visualizer

Settings

Properties

Locks

Essentials

Resource group: [defaultresourcegroup-cca](#)

Location: Canada Central

Subscription: [Azure subscription 1](#)

Subscription ID: 3f48112c-a56d-44e0-9a80-1ed5fbce3aac

Tags (edit)

Endpoint: [https://grafana-2025331844-cca-arth8b8dcbhg3c3e4-cca.grafana.az...](#)

Provisioning State: Succeeded

Grafana Version: 11.2.2+security-01

Pricing Plan: Standard

D. View the Grafana Dashboard

