**Microservices Integrated with AKS Cluster:**

**Pre-requisites:**

1. Docker Installation
2. Kubectl Installation
3. EKSCTL installation
4. Azure CLI Installation
5. AKS Cluster Setup
6. ACR setup.

**Installation steps for above pre-requisites:**

**Docker Installation Steps:**

**https://www.digitalocean.com/community/tutorials/how-to-install-and-use-docker-on-ubuntu-20-04**

1 sudo apt update

2 sudo apt install apt-transport-https ca-certificates curl software-properties-common

3 curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo apt-key add -

4 sudo add-apt-repository "deb [arch=amd64] https://download.docker.com/linux/ubuntu focal stable"

5 apt-cache policy docker-ce

6 sudo apt install docker-ce

7 sudo systemctl status docker

8 sudo usermod -aG docker azureuser

9 su - azureuser

10 docker info

11 docker –version

**Intalling Azure CLI:**

[**https://documentation.ubuntu.com/azure/en/latest/azure-how-to/instances/install-azure-cli/**](https://documentation.ubuntu.com/azure/en/latest/azure-how-to/instances/install-azure-cli/)

13. sudo apt-get update -y

14 sudo apt-get install ca-certificates curl apt-transport-https lsb-release gnupg

15 mkdir -p /etc/apt/keyrings

16 curl -sL https://packages.microsoft.com/keys/microsoft.asc | gpg --dearmor | sudo tee /etc/apt/keyrings/microsoft.gpg > /dev/null

17 SUITE=$(lsb\_release -cs)

18 echo "deb [arch=amd64 signed-by=/etc/apt/keyrings/microsoft.gpg] https://packages.microsoft.com/repos/azure-cli/ $SUITE main" | sudo tee /etc/apt/sources.list.d/microsoft.list

19 cat << EOF | sudo tee /etc/apt/preferences.d/99-microsoft

20 sudo apt-get update && sudo apt-get install -y azure-cli

**OR**

1. sudo apt install azure-cli
2. az login

when you execute az login command you will get login URL and key use copy the URL and browse and enter the key. Then it will ask your credentials you provide those credentials then it will get login.

A screenshot of a computer

Description automatically generated

KUBECTL INSTALLATION:-

curl -LO [https://dl.k8s.io/release/**$(**curl -L -s https://dl.k8s.io/release/stable.txt**)**/bin/linux/amd64/kubectl](https://dl.k8s.io/release/$(curl%20-L%20-s%20https://dl.k8s.io/release/stable.txt)/bin/linux/amd64/kubectl)

curl -LO [https://dl.k8s.io/release/**$(**curl -L -s https://dl.k8s.io/release/stable.txt**)**/bin/linux/amd64/kubectl.sha256](https://dl.k8s.io/release/$(curl%20-L%20-s%20https://dl.k8s.io/release/stable.txt)/bin/linux/amd64/kubectl.sha256)

echo "**$(**cat kubectl.sha256**)** kubectl" | sha256sum –check

Valid output

kubectl: OK

Install Kubectl:\_

Refer install link :\_ https://kubernetes.io/docs/tasks/tools/install-kubectl-linux/

sudo install -o root -g root -m 0755 kubectl /usr/local/bin/kubectl

chmod +x kubectl

mkdir -p ~/.local/bin

mv ./kubectl ~/.local/bin/kubectl

*# and then append (or prepend) ~/.local/bin to $PATH*

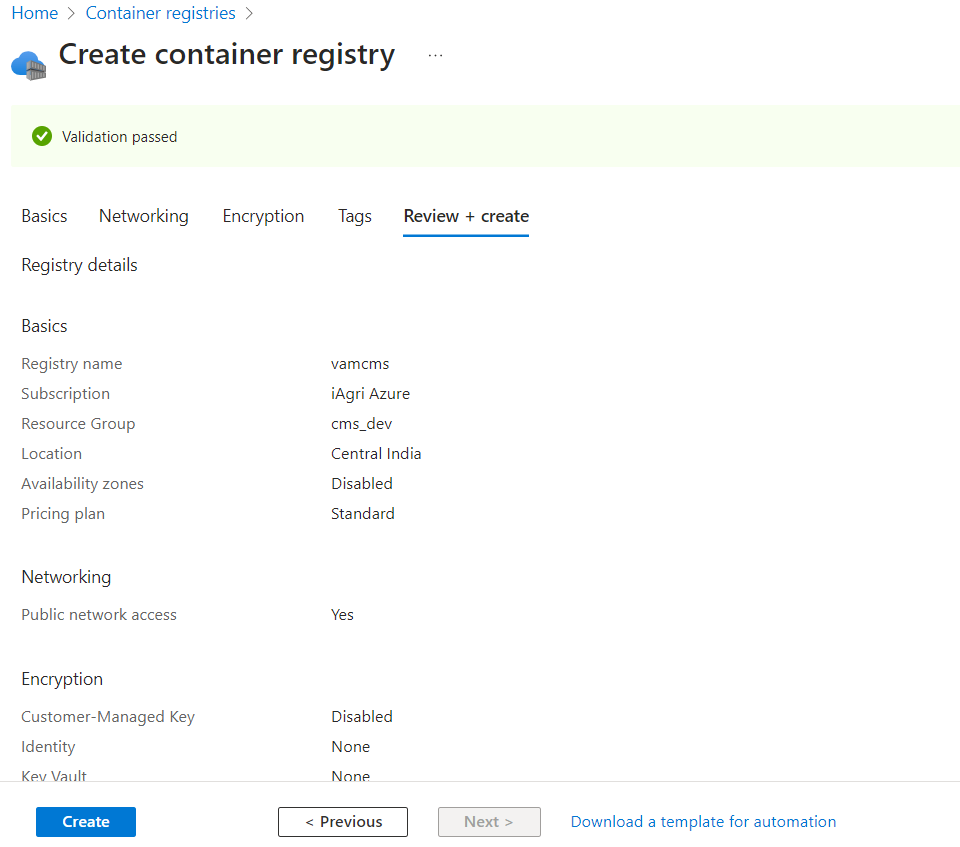
kubectl version –client

**ACR Setup:**

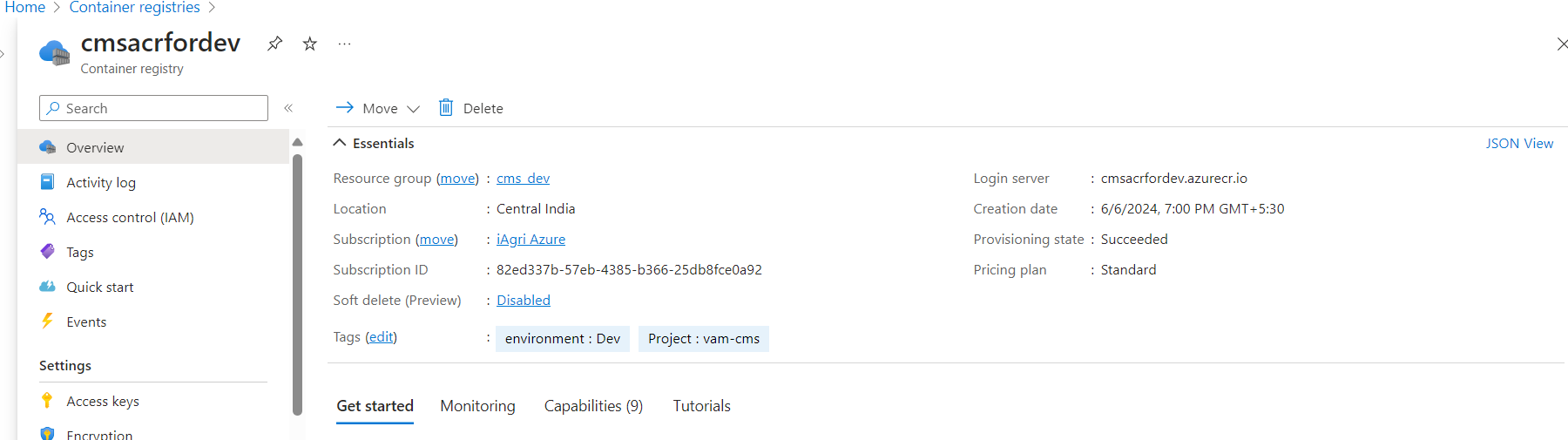
**Step:1** Go to Azure portal 🡪 Click on all services 🡪 Select container registries 🡪 Create New

A screenshot of a computer

Description automatically generated



**Step:2** Click on create



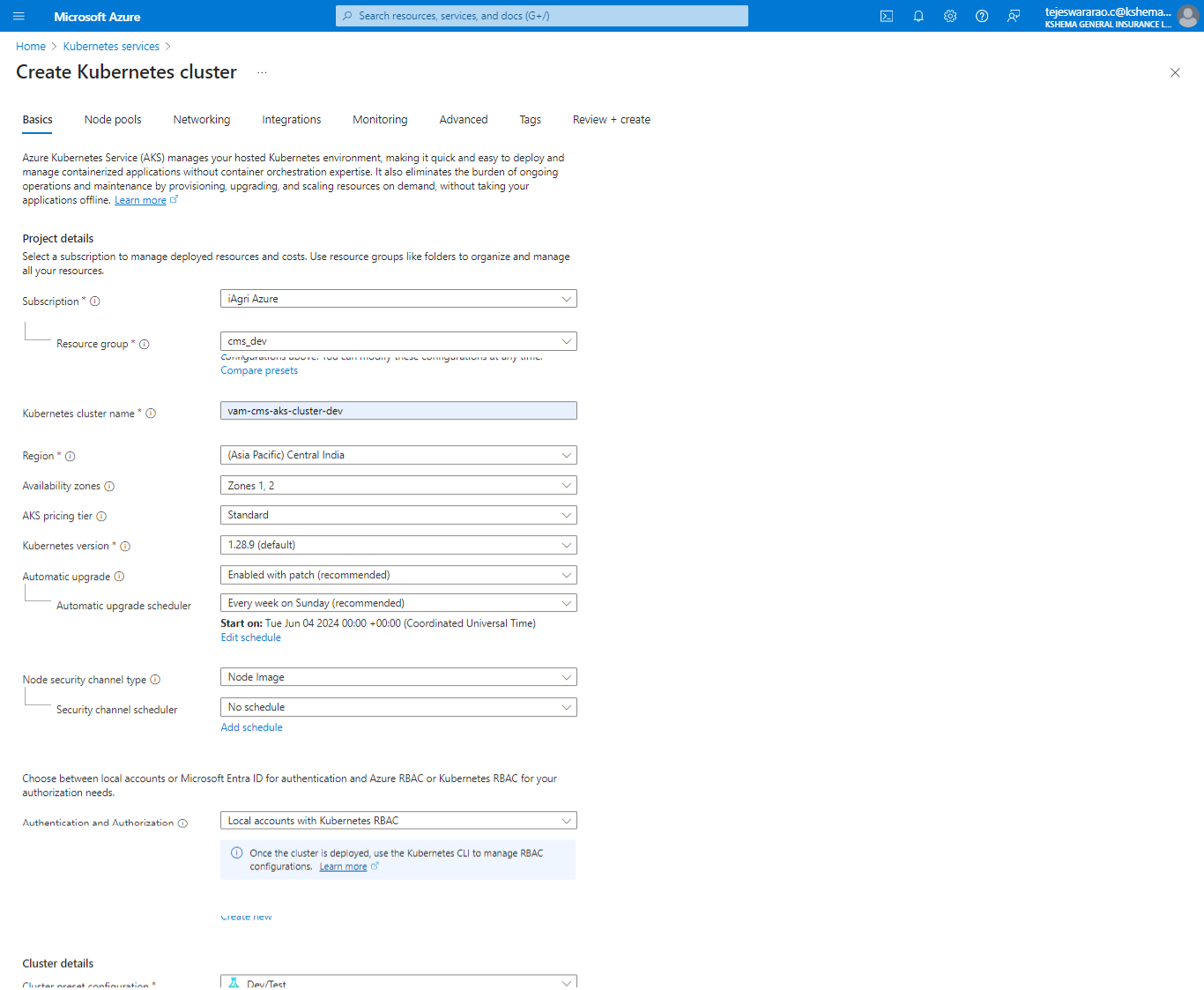
**AKS Cluster Setup:**

**Step:1** Go to Azure portal 🡪 Click on all services 🡪 Select Kubernetes Service 🡪 Create a Kubernetes Cluster

A screenshot of a computer

Description automatically generated

**Step:2** Go to Basics Field and fill the required information like below.



**Stpe:3** Go to Node pool Section and fill the required information of Node Size and Zones and OS type, min and max nodes.

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

**Step:4** Click on network tab and fill the networking related information as per your project requirements,

A screenshot of a computer

Description automatically generated

**Step:5** Go To Integration tab attach your ACR repository to your AKS Cluster.

A screenshot of a computer

Description automatically generated

**Step:6** Click on monitoring tab and fill the networking related information as per your project requirements.

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

**Step:7** Review and Create

A screenshot of a computer

Description automatically generated

**How to connect AKS Cluster:**

After installing the AKS cluster go to cluster and click on connect.

A screenshot of a computer

Description automatically generated

Az login then execute the below commands.

A screen shot of a computer

Description automatically generated

**Service Connection Creation For Docker Registry and Kubernetes.**

**Docker Registry Service Connection:**

**Step:1** Go toAzure DevOps portal 🡪 Login to your organization 🡪 Select Project 🡪 CMS 🡪 Project Settings 🡪 Select Service Connections 🡪 New Service Connection

A screenshot of a computer

Description automatically generated

**Step:2** Search For New Service Connection For Docker Registry.

A screenshot of a computer

Description automatically generated

**Step:3** fill therequired details for new service connection

A screenshot of a computer

Description automatically generated

**Step:4** Give Permissions to all pipelines and click on create.

**Kubernetes Service Connection:**

**Step:1** Go toAzure DevOps portal 🡪 Login to your organization 🡪 Select Project 🡪 CMS 🡪 Project Settings 🡪 Select Service Connections 🡪 New Service Connection

A screenshot of a computer

Description automatically generated

**Step:2** Search For New Service Connection For Kubernetes.

A screenshot of a computer

Description automatically generated

**Step:3** fill therequired details for new service connection

A screenshot of a computer

Description automatically generated

**Step:4** Click on save

**Project Setup:**

**Azurepipeline.yaml**

trigger:

- Test

pool:

  name: 'Cms\_Dev\_Agentpool'

variables:

  dockerRegistryServiceConnection: 'cms-dev-docker-service-connection'

  imageRepository: 'rulevalidation'

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

  tag: '$(Build.BuildId)'

  secretName: 'cmssecret'

  JAVA\_HOME: '/usr/lib/jvm/java-21-openjdk-amd64'

  JAVA\_HOME\_21\_X64: '/usr/lib/jvm/java-21-openjdk-amd64'

stages:

- stage: build

  jobs:

  - job:

    steps:

    - script: |

        echo "JAVA\_HOME=$JAVA\_HOME"

        echo "JAVA\_HOME\_21\_X64=$JAVA\_HOME\_21\_X64"

        java -version

      displayName: 'Check Java Environment Variables'

    - task: JavaToolInstaller@0

      inputs:

        versionSpec: '21'

        jdkArchitectureOption: 'x64'

        jdkSourceOption: 'PreInstalled'

    - task: Maven@4

      inputs:

        mavenPomFile: 'ruleValidation/pom.xml'

        publishJUnitResults: true

        testResultsFiles: '\*\*/surefire-reports/TEST-\*.xml'

        javaHomeOption: 'Path'

        jdkDirectory: '$(JAVA\_HOME)'  # Explicitly use the JAVA\_HOME variable

        #javaHomeOption: 'JDKVersion'

        jdkVersionOption: 'Default'

        mavenVersionOption: 'Default'

        mavenOptions: '-Xmx3072m'

        mavenAuthenticateFeed: false

        effectivePomSkip: false

        #sonarQubeRunAnalysis: false

        goals: 'clean package'

    - task: PublishTestResults@2

      inputs:

        testResultsFormat: 'JUnit'

        testResultsFiles: '\*\*/surefire-reports/TEST-\*.xml'

    - task: CopyFiles@2

      inputs:

        SourceFolder: '$(system.defaultworkingdirectory)'

        Contents: '\*\*/target/\*.jar'

        TargetFolder: '$(build.artifactstagingdirectory)'

    - task: PublishBuildArtifacts@1

      inputs:

        PathtoPublish: '$(Build.ArtifactStagingDirectory)'

        ArtifactName: 'drop'

        publishLocation: 'Container'

    - task: DockerInstaller@0

      inputs:

        dockerVersion: '26.1.3'

    - task: Docker@2

      displayName: Build and publish image to Azure Container Registry

      inputs:

        command: buildAndPush

        containerRegistry: $(dockerRegistryServiceConnection)

        repository: $(imageRepository)

        dockerfile: $(dockerfilePath)

        tags: |

          $(tag)

- stage: Deploy

  displayName: Deploy to AKS

  dependsOn: Build

  condition: succeeded()

  jobs:

  - deployment: Deploy

    displayName: Deploy to AKS

    pool:

      name: 'Cms\_Dev\_Agentpool'

    environment: 'Dev'  # Update with your AKS environment name

    strategy:

      runOnce:

        deploy:

          steps:

          - checkout: self

          - script:

              cd $(Build.SourcesDirectory) && ls

          - task: KubernetesManifest@0

            inputs:

              action: 'createSecret'

              kubernetesServiceConnection: 'Dev-AKS-Service-Connection'

              namespace: 'dev'

              secretType: 'dockerRegistry'

              secretName: 'cmssecret'

              dockerRegistryEndpoint: 'cms-dev-docker-service-connection'

          - script: |

              ls $(Build.SourcesDirectory)

              sed -i 's/{{TAG\_PLACEHOLDER}}/$(Build.BuildId)/g' $(System.DefaultWorkingDirectory)/deployment.yaml

            displayName: 'Replace tag placeholder'

          - task: KubernetesManifest@0

            displayName: 'Deploy to AKS'

            inputs:

              action: 'deploy'

              kubernetesServiceConnection: 'Dev-AKS-Service-Connection'

              namespace: 'dev'

              manifests: |

                #$(Build.SourcesDirectory)/deployment.yaml

                $(System.DefaultWorkingDirectory)/deployment.yaml

              # imagePullSecrets: |

              #   - name: 'cmssecret'

              # containers: |

              #    $(containerRegistry)/$(imageRepository):$tag

**Dockerfile:**

FROM amazoncorretto:21

RUN yum install -y shadow-utils

RUN mkdir -p /Claim\_Ingestion/

WORKDIR /Claim\_Ingestion

COPY Claim\_Ingestion .

EXPOSE 8090

RUN groupadd -r appusergroup && useradd -r -g appusergroup appuser

RUN chown -R appuser:appusergroup /Claim\_Ingestion

USER appuser

CMD ["java", "-jar", "/Claim\_Ingestion/target/Claim\_Ingestion-0.0.1-SNAPSHOT.jar"]

**Deployment.yaml**

---

apiVersion: v1

kind: Service

metadata:

  name: cms-svc

  namespace: dev

spec:

  ports:

  - port: 80

    targetPort: 8100

    protocol: TCP

    name: http

  selector:

    app: cms

---

apiVersion: apps/v1

kind: Deployment

metadata:

  name: cms-deployment

  namespace: dev

spec:

  replicas: 1

  selector:

    matchLabels:

      app: cms

  template:

    metadata:

      labels:

        app: cms

    spec:

      # imagePullSecrets:

      # - name: cmssecret

      containers:

      - name: cms

        #image: vamcmsqaacr.azurecr.io/spring-boot-3-rest-api-example-master:#{Build.BuildId}#

        #image: vamcmsqaacr.azurecr.io/vamcmsqaacr:${Build.BuildId}

        image: cmsacrfordev.azurecr.io/cmsacrfordev:{{TAG\_PLACEHOLDER}}

        #image: cmsacrfordev.azurecr.io/cms:$tag

        imagePullPolicy: Always

        resources:

          requests:

            memory: "512Mi"

            cpu: "0.3"

          limits:

            memory: "1024Mi"

            cpu: "0.5"

        ports:

        - containerPort: 8100

---

apiVersion: autoscaling/v2

kind: HorizontalPodAutoscaler

metadata:

  name: cms-deployment

  namespace: dev

spec:

  scaleTargetRef:

    apiVersion: apps/v1

    kind: Deployment

    name: cms-deployment

  minReplicas: 1

  maxReplicas: 3

  metrics:

  - type: Resource

    resource:

      name: cpu

      target:

        type: Utilization

        averageUtilization: 70

Finally run Azurepipeline to deploy microservice into AKS Cluster.

We will get output like below.

A screenshot of a computer

Description automatically generated

Go to Virtual machine where you have integrated AKS cluster configuration commands.

Execute the commands to check deployment of microservice into AKS Cluster.

**Kubectl get pods -n dev**

A screenshot of a computer program

Description automatically generated

**Kubectl get svc -n dev**

A screenshot of a computer screen

Description automatically generated

**Kubectl get deploy -n dev**

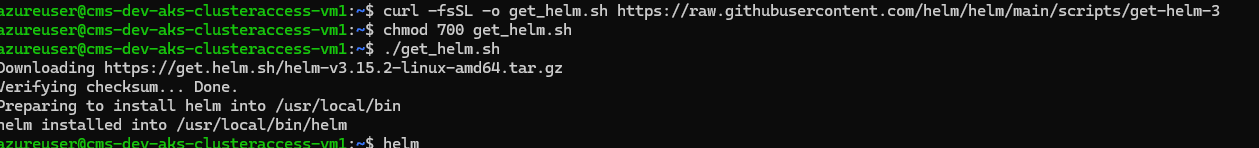
A screen shot of a computer

Description automatically generated

If you want configure ingress to your AKS cluster. Use the below commands to set up ingress controller into your AKS cluster. Install helm and execute the below commands.

HELM INSTALLATION: -

Refer Doc:- <https://helm.sh/docs/intro/install/>



**Installing HAProxy Ingress-Controller:**

**helm repo add haproxytech** [**https://haproxytech.github.io/helm-charts**](https://haproxytech.github.io/helm-charts)

**helm repo update**

**helm install haproxy-kubernetes-ingress haproxytech/kubernetes-ingress --create-namespace --namespace haproxy-controller --set controller.service.type=ClusterIP --set controller.service.annotations."service\.beta\.kubernetes\.io/azure-load-balancer-health-probe-request-path"="/healthz"**

A black screen with white text

Description automatically generated

after installing the Ingress Controller we have to configure ingress resource to AKS Cluster.

**Ingress Resource.yaml:**

apiVersion: networking.k8s.io/v1

kind: Ingress

metadata:

name: claims-workflow-ingress

namespace: dev

spec:

ingressClassName: haproxy

rules:

- host: cms.dev.aks.kshema.co

http:

paths:

- path: "/"

pathType: Prefix

backend:

service:

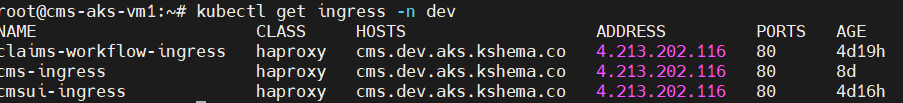
name: cmsui-svc

port:

number: 80

after configuring the ingress resource we can check ingress status using below command.

Kubectl get ingress -n dev



Using Ingress We can expose the service with outside world. The out put like below.



**Self Hosted Agent Installation Steps:**

When ever you configure selfhost agent. You should install required dependencies. Then only self hosted agent will work properly without compatibility issues.

**Pre-requisites:**

1. Java Installation (Compatible version with your project).
2. Maven Installation (Java Project)
3. Npm installation
4. Nodejs Installation

**Java Installation:**

sudo apt update

apt install openjdk-21 -y

**Maven Installation:**

sudo apt update

wget https://apache.osuosl.org/maven/maven-3/3.9.8/binaries/apache-maven-3.9.8-bin.tar.gz

sudo tar xf apache-maven-3.9.8-bin.tar.gz -C /opt

cd /opt

sudo mv apache-maven-3.9.8 /opt/maven

sudo nano /etc/profile.d/maven.sh

export MAVEN\_HOME=/opt/maven

export PATH=$MAVEN\_HOME/bin:$PATH

source /etc/profile.d/maven.sh

mvn -version

**Npm and Nodejs installation:**

sudo apt-get update

curl -fsSL <https://deb.nodesource.com/setup_lts.x> | sudo -E bash -  
sudo apt-get install -y nodejs

**ZIP Installation:**

sudo apt-get update

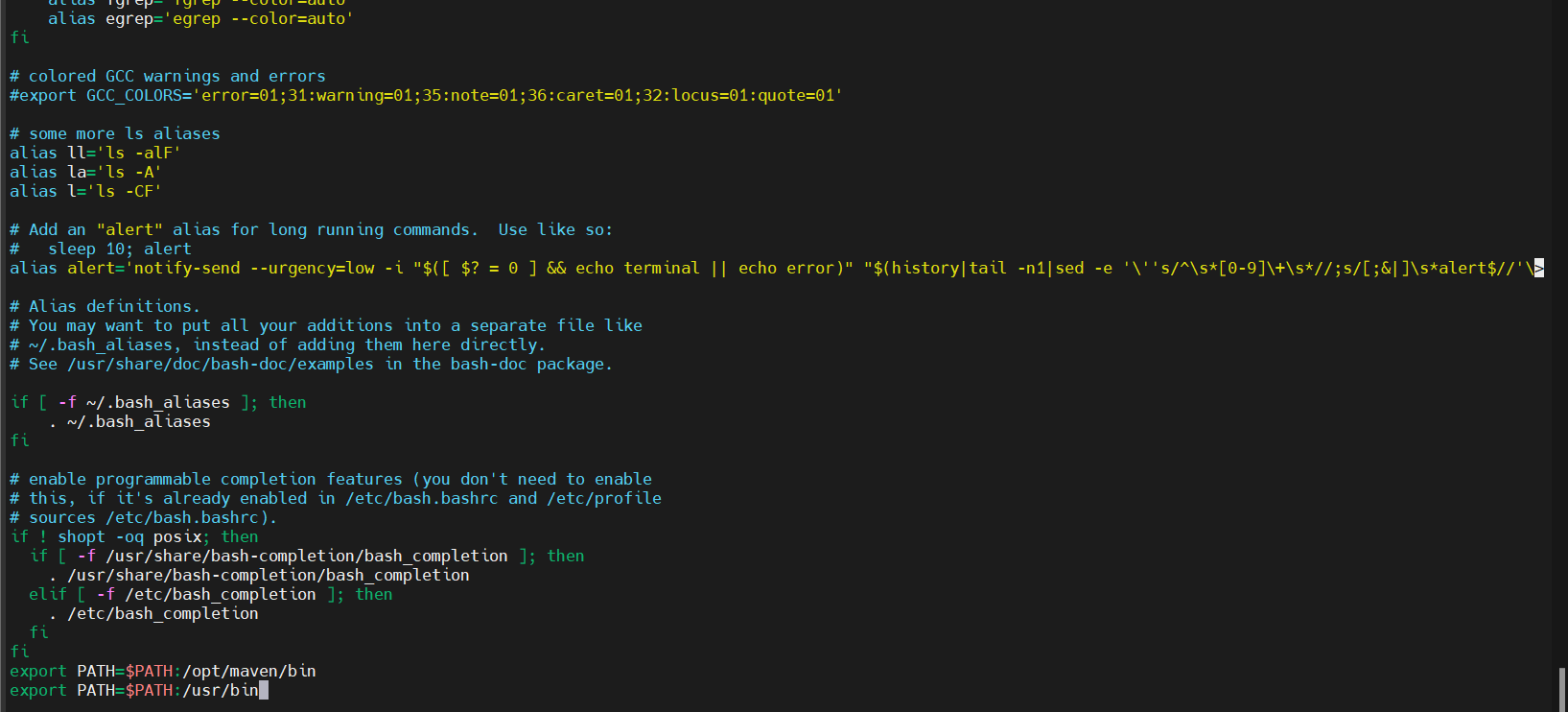
sudo apt-get install zip -y

zip –version

we have to update PATH in environment variables.

nano ~/.bashrc

export PATH=$PATH:/usr/bin



After installing the required dependencies. Configure self-hosted agent

**Step1:**  Go to project settings 🡪 select agentpool 🡪 select new agentpool 🡪give name

A screenshot of a computer

Description automatically generated

A screenshot of a computer

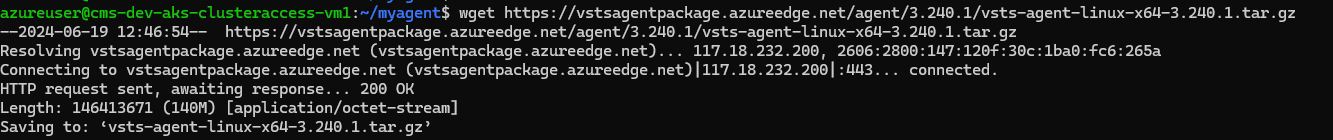
Description automatically generated

A screenshot of a computer

Description automatically generated

Open the powershell or command prompt and execute above commands:







A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer program

Description automatically generated

**Private AKS Cluster Service Connection:**

**Step:1** Go toAzure DevOps portal 🡪 Login to your organization 🡪 Select Project 🡪 CMS 🡪 Project Settings 🡪 Select Service Connections 🡪 New Service Connection

A screenshot of a computer

Description automatically generated

**Step:2** Search For New Service Connection For Kubernetes.

A screenshot of a computer

Description automatically generated

**Step:3** select the kubeconfig for new connection and fill the required details for new service connection

A screenshot of a computer

Description automatically generated

To view the kubeconfig file. Execute the below commands.

az aks get-credentials -n cms-dev-aks-cluster1 -g cms\_dev

cat ~/.kube/config (copy the kubeconfig file and update in service connection)

**Step:4**  click on give permissions to all pipelines and Click on save.