Azure CICD K8S GitOps(ArgoCD)

Friday, April 18, 2025

11:22 PM

**Example #1:**

**Deploy microservices using azure CICD pipeline using ACR.**

**Steps:**

1. **Import project in azure repo**
2. **Create RG and Container registry**
3. **Create Self Hosted Agent (Linux VM and configure it)**

**Create VM using Portal**

**Login your vm using ssh from command line:**

**Ssh -I <yourkey.pem> azureuser@publicIp**

**Sudo apt update**

**Sudo apt install docker.io**

**Sudo systemctl restart docker**

**Sudo systectl status docker**

**Mkdir myagent**

**Cd myagent**

**Give docker permission to your VM.**

Sudo usermod -aG docker azureuser

**(azureuser is vm name)**

1. **Set up agent pool and agent on vm :**

**Go to agent pool in ADO and create a pool and then create an agent inside it.**

**Then go to your poolname then Agent and copy below url from next to "Download" button**

Wget <https://download.agent.dev.azure.com/agent/4.255.0/vsts-agent-linux-x64-4.255.0.tar.gz>

tar zxvf vsts-agent-linux-x64-4.255.0.tar.gz

1. **Configure your Agent on VM**

./config.sh

Enter (Y/N) Accept the Team Explorer Everywhere license agreement now? (press enter for N) > **Y**

**Enter your organization url (you will find in ADO)**

Enter server URL > [**https://dev.azure.com/KomalaDemOrg**](https://dev.azure.com/KomalaDemOrg)

**Enter your pool name that you created before (in my case azureagentpool )**

**Confirm your VM name, just enter**

**Your screen should like something like below:**

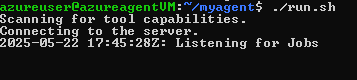


**Your Agent is configured successfully…now you can run it anytime you want use.**

1. **Now run your agent**

./run.sh

**You should see below:**



1. **Create Pipeline using docker registry template**

# Docker

# Build and push an image to Azure Container Registry

# <https://docs.microsoft.com/azure/devops/pipelines/languages/docker>

trigger:

 paths:

   include:

     - result/\*

resources:

- repo: self

variables:

  # Container registry service connection established during pipeline creation

  dockerRegistryServiceConnection: '43d95f3f-3689-4320-af06-d105c5c7fbbf'

  imageRepository: 'resultapp'

  containerRegistry: 'azurecicidregistry.azurecr.io'

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

  tag: '$(Build.BuildId)'

  # Agent VM image name

pool:

    name: 'default'

stages:

- stage: Build

  displayName: Build

  jobs:

  - job: Build

    displayName: Build

    steps:

    - task: Docker@2

      displayName: Build image to container registry

      inputs:

        containerRegistry: '$(dockerRegistryServiceConnection)'

        repository: '$(imageRepository)'

        command: 'build'

        Dockerfile: 'result/Dockerfile'

        tags: '$(tag)'

- stage: Push

  displayName: Push

  jobs:

  - job: Push

    displayName: Push

    steps:

    - task: Docker@2

      displayName: Push image to container registry

      inputs:

        containerRegistry: '$(dockerRegistryServiceConnection)'

        repository: '$(imageRepository)'

        command: 'push'

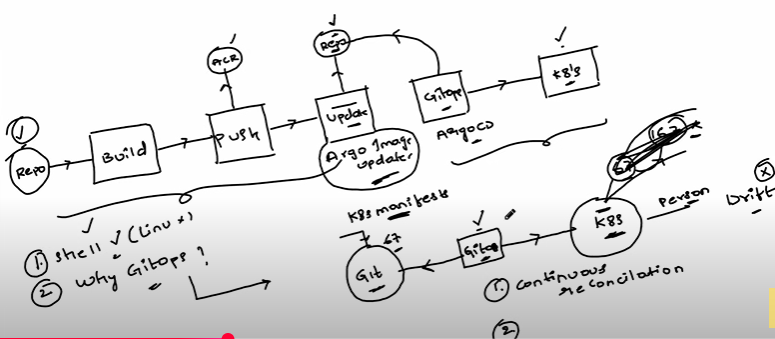
        Dockerfile: 'result/Dockerfile'

        tags: '$(tag)'

**Save and run your job.**

**Example #2:**

**Deploy microservices using azure CICD pipeline using ACR and AKS and GitOps(ArgoCD)**



1. **Get code in repo (Done in Example#1)**
2. **Setup pipelines for all microservices (Done in Example#1)**
3. **Prepare K8 Cluster**

**Name-**

Azuredevopsk8

East US

Standard\_D2ads\_v6

Zone 1, Zone 2, Zone 3

Ubuntu Linux

az aks get-credentials --name azuredevopsk8 --overwrite-existing --resource-group azurecicd

**Test if K8 in your system:**

kubectl get pods

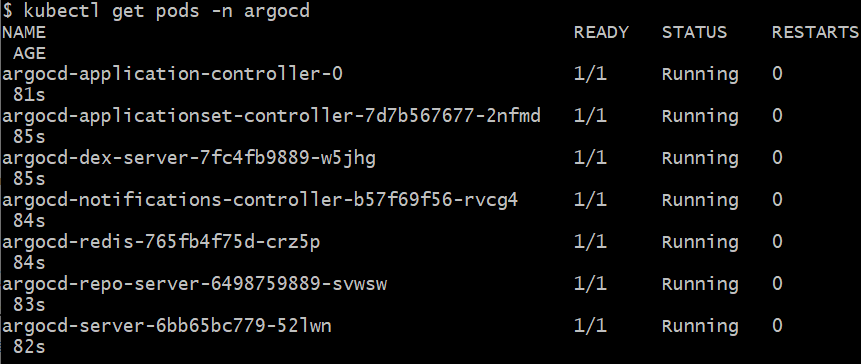
1. **Install GitOps (argoCD)**

**Got to argoCD official documentation > getting started**

kubectl create namespace argocd

kubectl apply -n argocd -f <https://raw.githubusercontent.com/argoproj/argo-cd/stable/manifests/install.yaml>

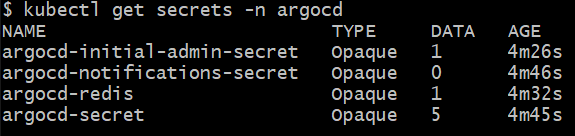
kubectl get pods -n argocd



1. **Configure argoCD with your Git repo**

**5.1 : Get argocd credentials**

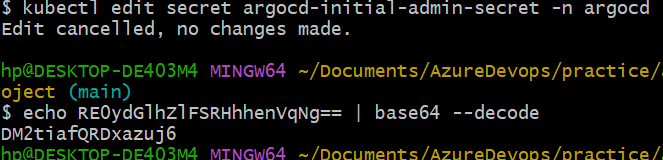
**kubectl get secrets -n argocd**



kubectl edit secret argocd-initial-admin-secret -n argocd

**Then decode password using:**

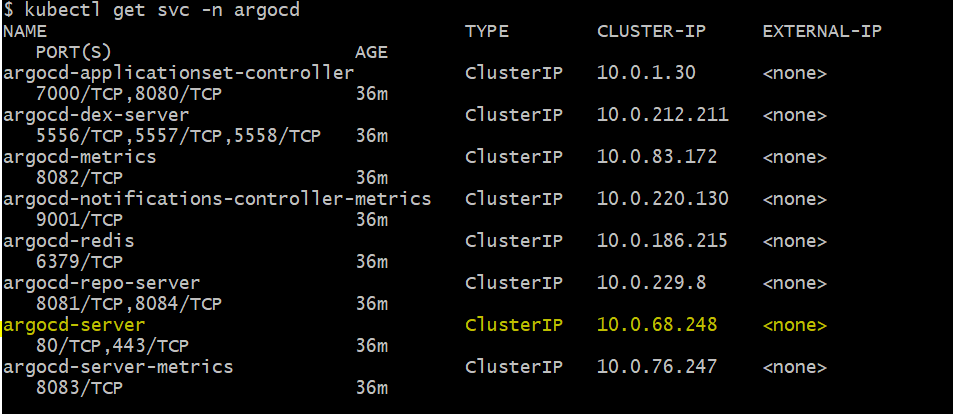
Echo <encryptedPassword> | base64 --decode



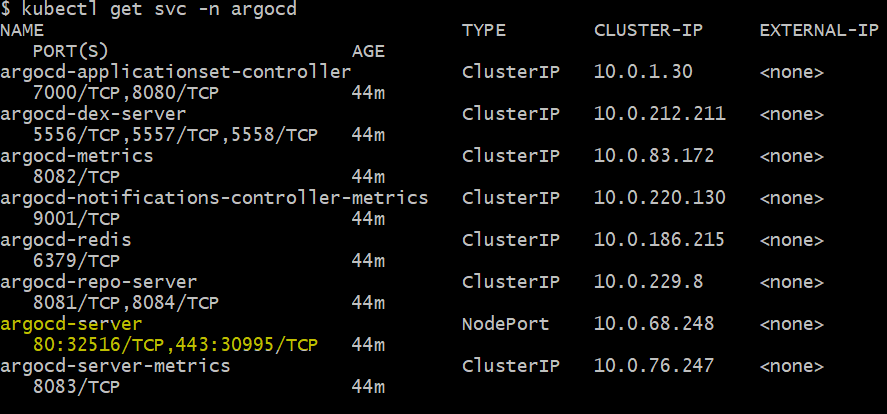
DM2tiafQRDxazuj6

**5.1 : Expose argocd in NodePort**

kubectl get svc -n argocd

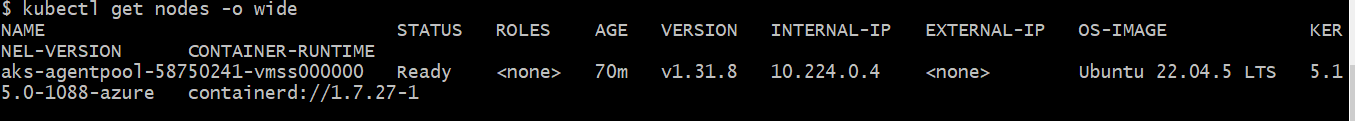


kubectl patch svc argocd-server -n argocd -p '{"spec": {"type": "NodePort"}}'



**Get node id:**

kubectl get nodes -o wide



**Expose argocd-server from cluster**

kubectl port-forward svc/argocd-server -n argocd 8080:443

Access using localhost:8080

Login with username - admin and password that we saved before.

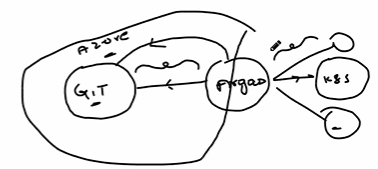
**NOTE: The above yellow steps I had to do because while create K8 I forgot to assign public ip to VMSS instance hence no EXTERNAL\_IP assigned here hence I had to do port forward and use locally. If you see EXTERNAL\_IP in your kubectl get svc -n argocd command response then on need this step.**

**Then go to Settings in argoCD UI (EXTERNAL\_IP:NodePort)**

Connect Repository

User http:<PAT>@repourl (Replace name with PAT)

**We have done GIT To Argo connection successfully**

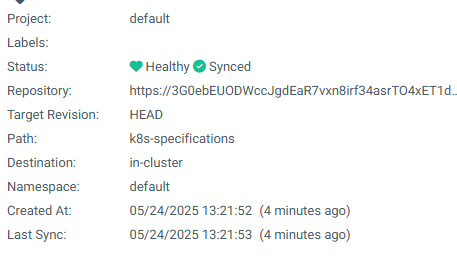


**Now time to Connect argo to K8s Cluster:**

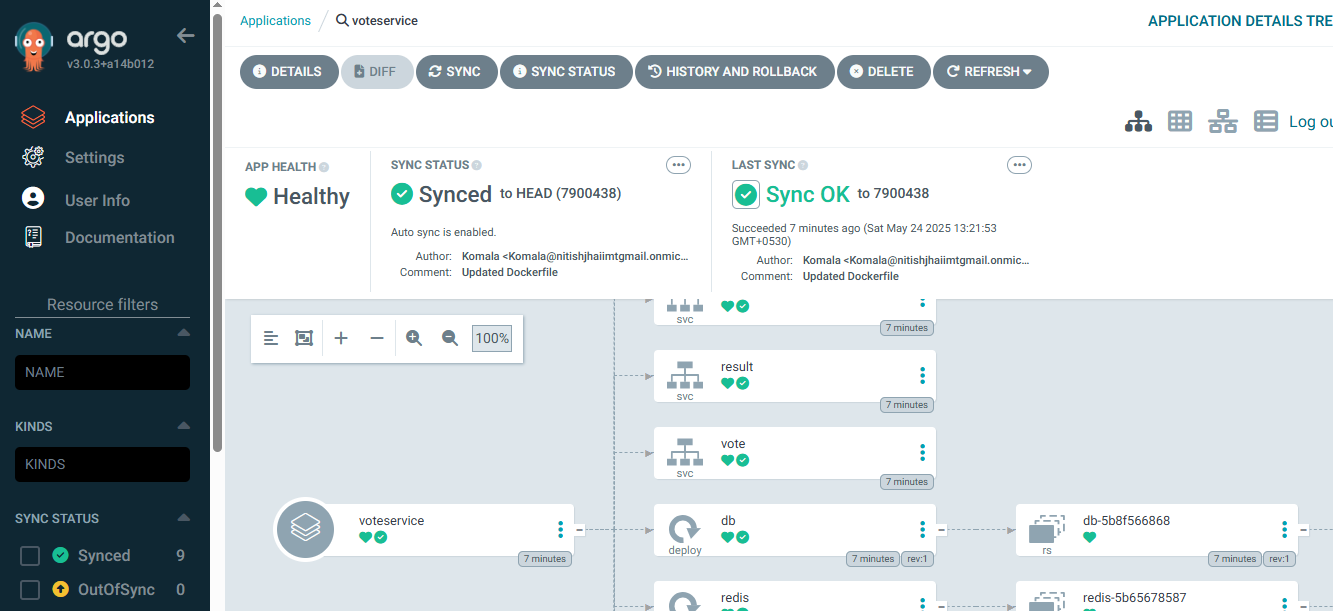
**Go to argoCD Application >> Create app**

* **Give app name e.g. votingapp**
* **Repo url will populate automatically and select the same.**
* **Target leave as HEAD**
* **Path give k8s-specifications because we want to deploy menifest file from our report to cluster**
* **Destination - select cluster, you may see multiple clusters so choose where you want to deploy**
* **Namespace - you can write your desired namespace, default in my case.**

**You should see below in argocd:**



**Click on above and you can see your git is synching up.**



1. **Update k8s specification files using shell script (updateK8sManifests.sh)**

We can use argocd image updater as well, give it a try.

1. **Test pipeline**

* Do some changes in project git push and see the magic,
* The pipline should run
* script should update respective file inside k8s-specification
* argoCD should trigger changes automatically and deploy those on k8 cluster.