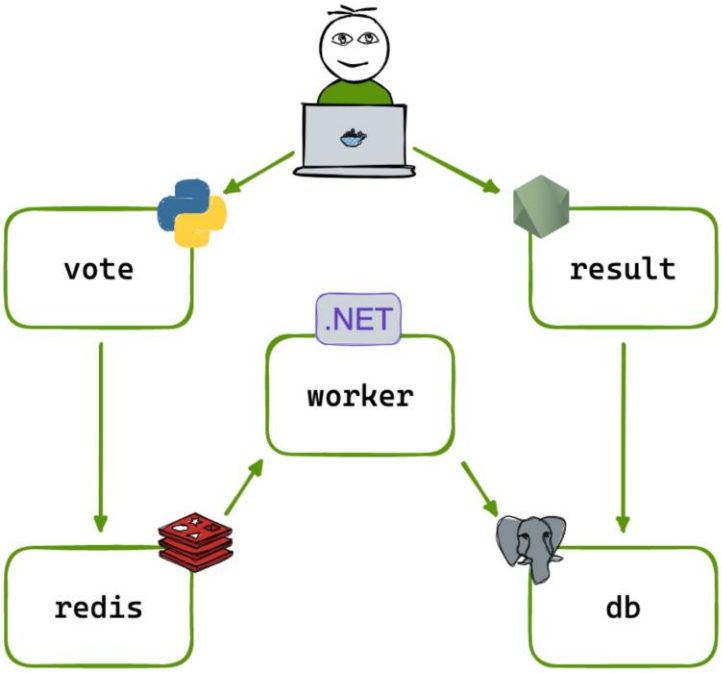
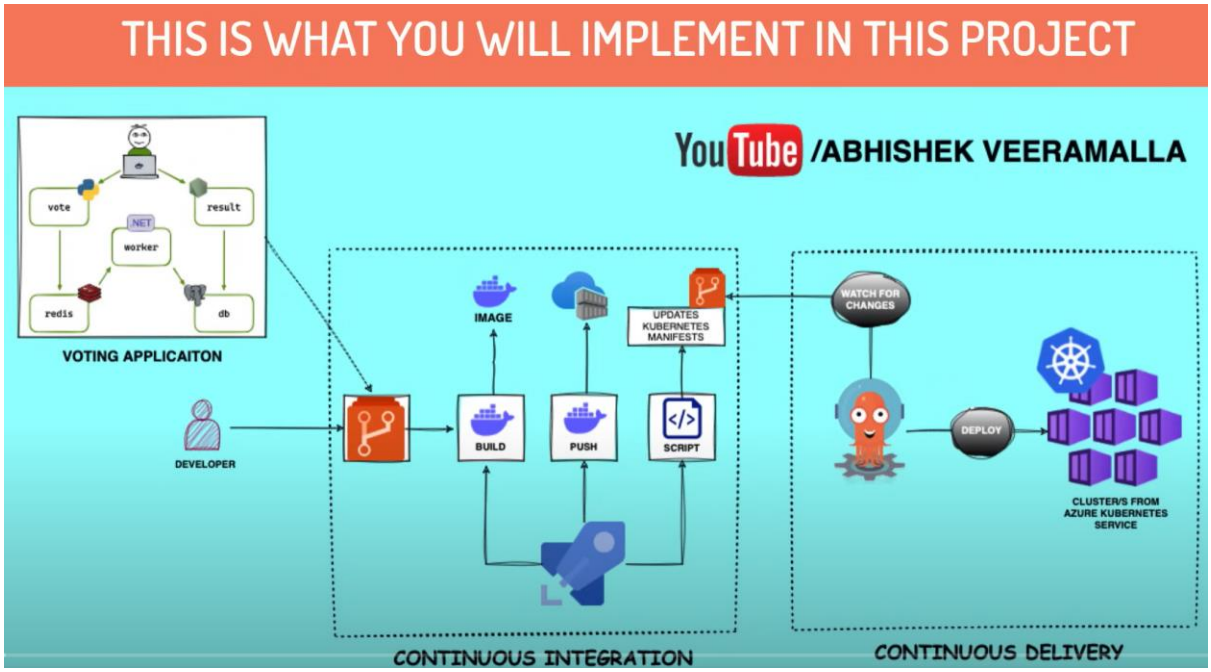
Day 14

App Repo: https://github.com/dockersamples/example-voting-app



Project Flow: Migrating the source code form GitHub repo to Azure DevOps Repo and the CICD using Azure DevOps Pipelines and Argo CD. During CI – Build using docker and push to Azure Container registry. And then updating the K8s manifest files in CI pipeline and deployment using Argo Cd into AKS Cluster.

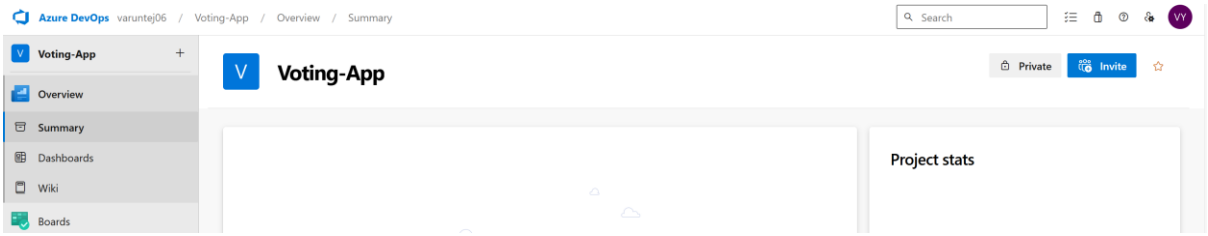
Prefer to do entire project in Ubuntu machine:



Let's do this application CICD in Azure DevOps:

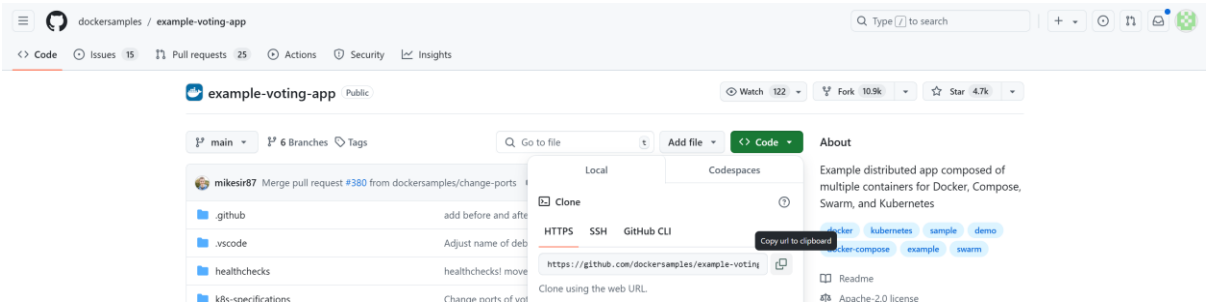
Let's first import the repo into **Azure DevOps Repos**:

Login to Azure DevOps and then create a new project with name Voting-App.



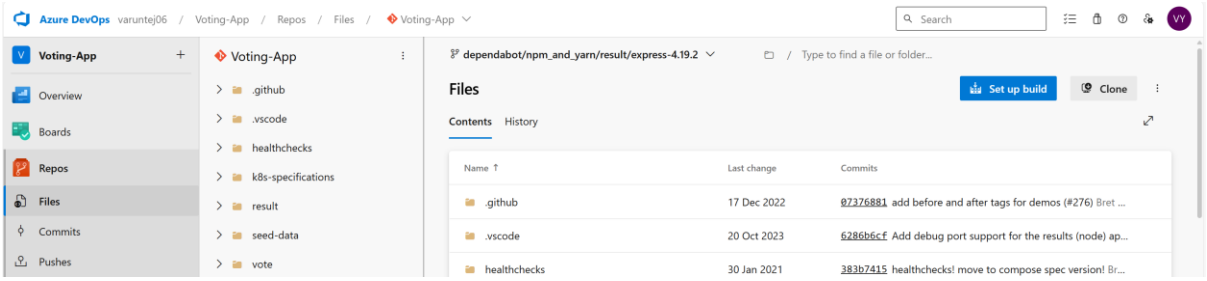
Go to Repo and add then “Import a Repository”

Github Repo URL from Dockersamples: https://github.com/dockersamples/example-voting-app

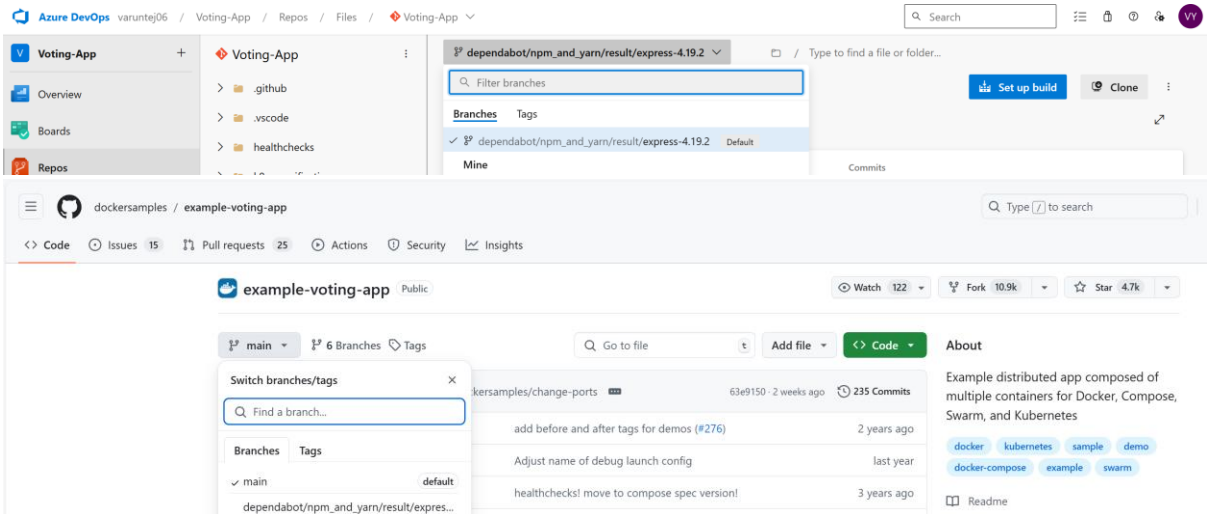


Copy the URL and give it in Azure DevOps.

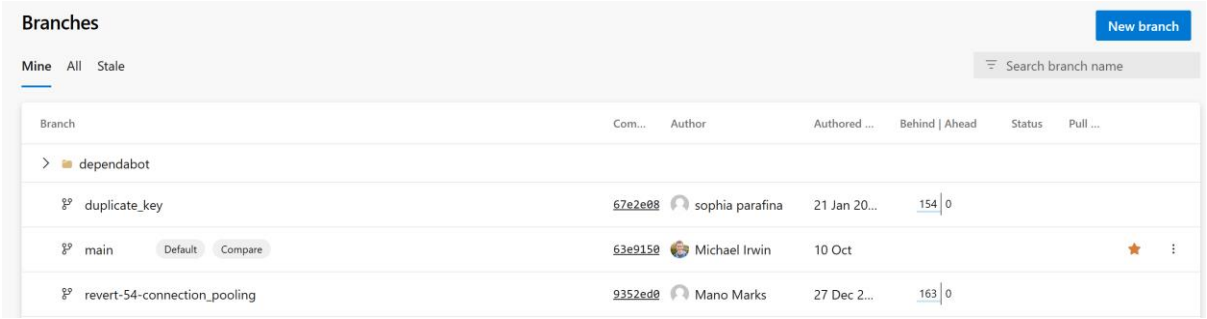
Entire Git repo is here now:



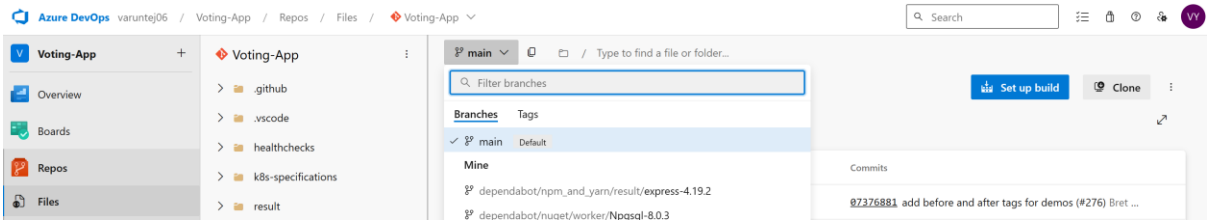
The default branch is different than GitHub repo default branch:



We have to change the default branch in Repos:



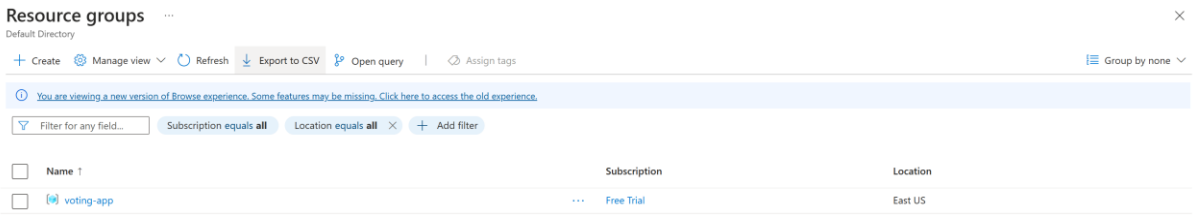
Go to Repo to check the default branch:



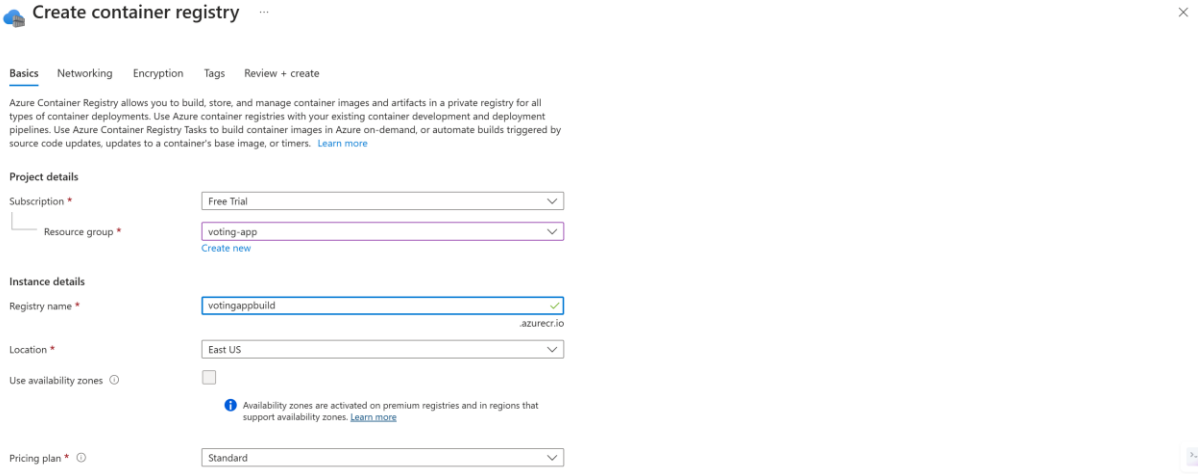
Now we have to create pipelines. We have to create three pipelines for Voting app, Result app and worker app. And whatever the build artifact we will receive will be placed in azure container registry in azure portal.

Lets create container registry first, and also resource group:

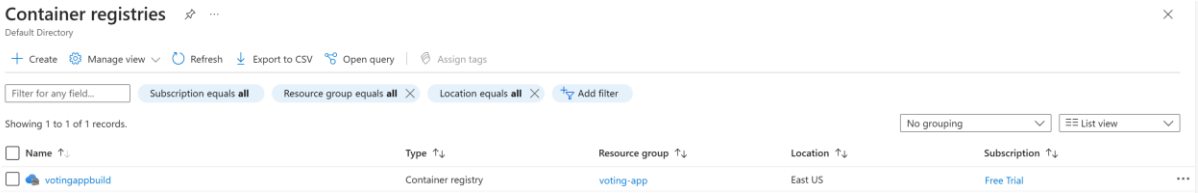
Login to Azure Portal and create a resource group.



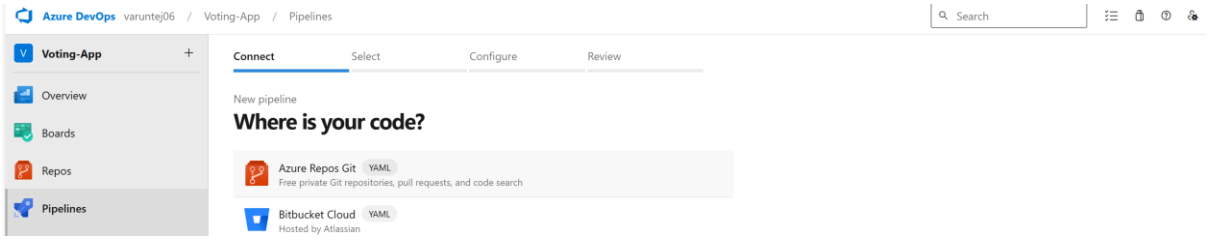
Go to Container Registry and create one registry:



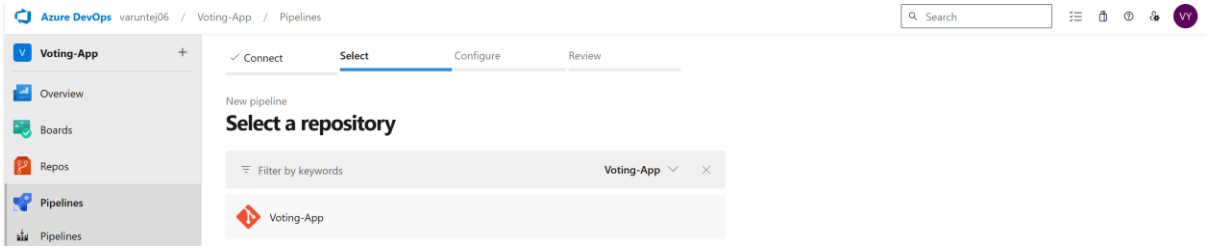
Click on Review + Create.



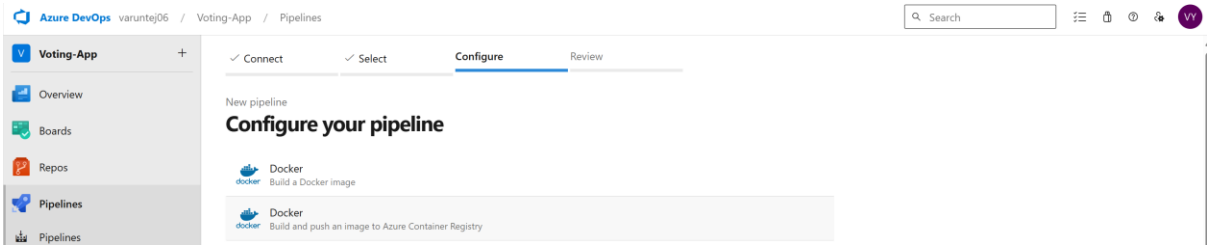
Now go to Azure Devops and create pipelines for Result MS first: Click on create pipeline.



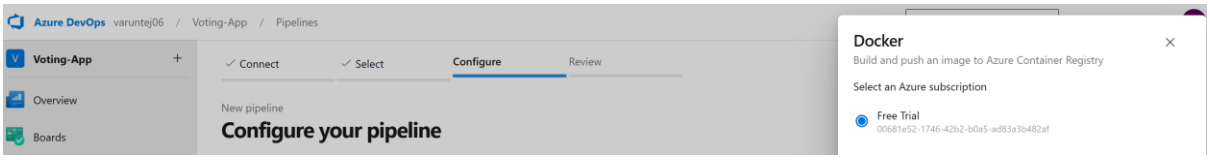
Select Azure Repos Git.



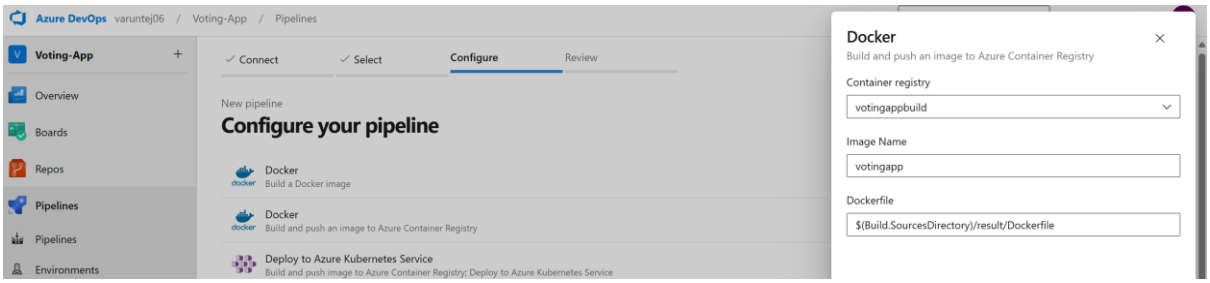
Select the repo.



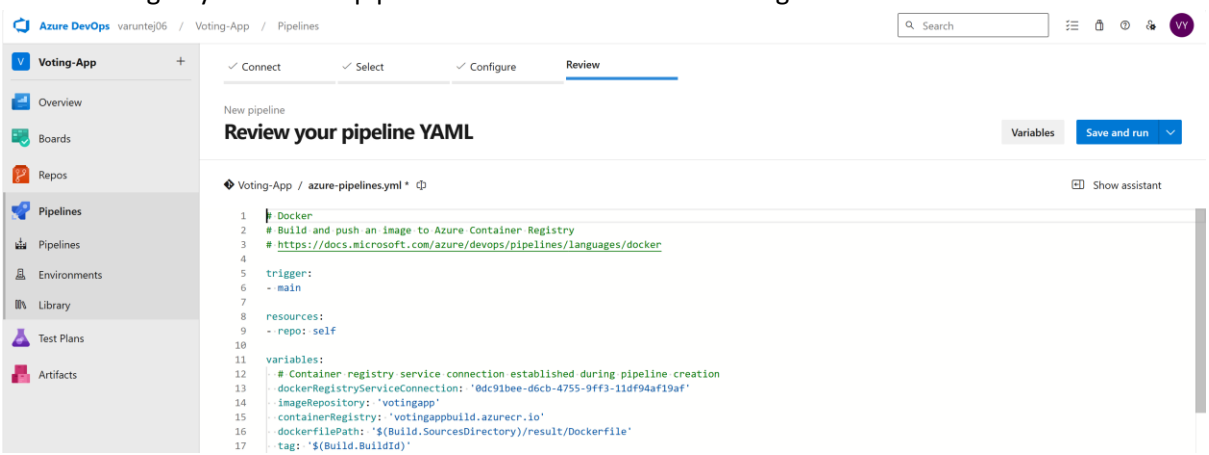
Select the template for CI pipeline. Select the Second one “Buidl and push the image to Azure Conatiner Registry”. Then it will load the subscription. Select free trail.



Then login with the Azure portal account.



Then it will show the Container registry we had in our account. It creates a connection to the Azure container registry with the CI pipeline. Click on Validate and configure.



Pipeline template with some code snippets is already written. Lets modify this:

Change the pipeline name:

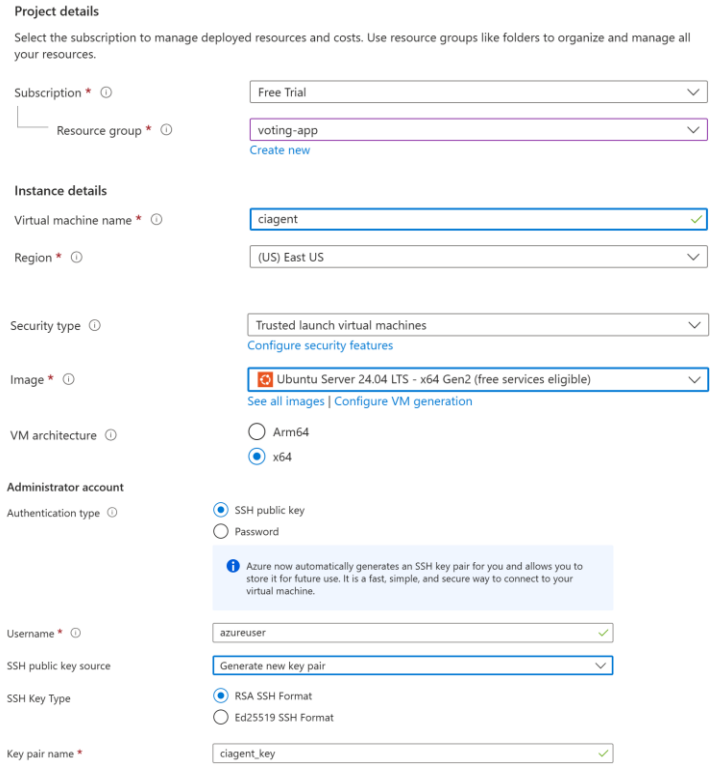
Update the pipeline:

```yaml
trigger:
  paths:
    include:
      - result/*
# any change inside the result folder will trigger this pipeline.

resources:
- repo: self
 variables:
  # Container registry service connection established during pipeline creation
  dockerRegistryServiceConnection: '0dc91bee-d6cb-4755-9ff3-11df94af19af'
  imageRepository: 'resultapp'
  containerRegistry: 'votingappbuild.azurecr.io'
  dockerfilePath: '$(Build.SourcesDirectory)/result/Dockerfile'
  tag: '$(Build.BuildId)'
```

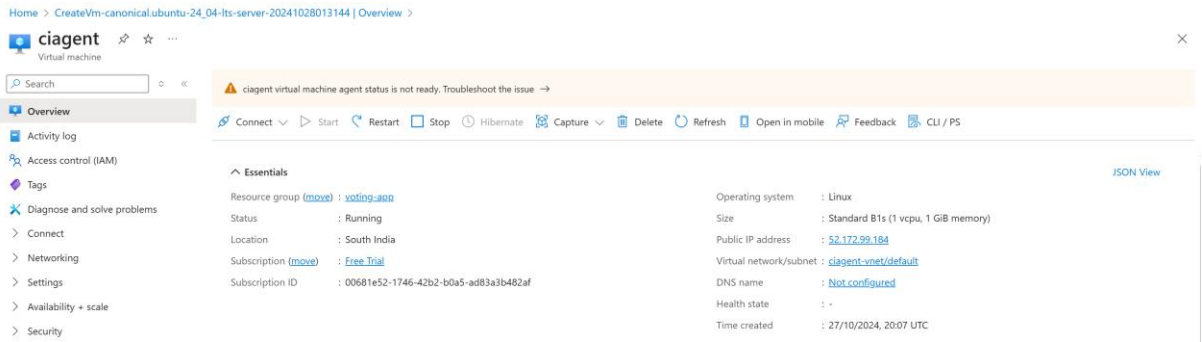
Azure doesn't provide an agent to run the pipeline in free tier account. Instaed we can create our own agent:

```yaml
    # Agent VM image name
  pool:
  name: 'ciagent'
  # should have an agent - VM with this name created in azure portal to use as agent to run this pipeline
```

Now we have to go back to azure and create a VM and this VM will acts as a Agent for our CI pipeline.



VM is created.



Now lets update the stages:

We will take only build in one stage. You can get help by clicking on settings and you can update in the right side popup menu and it will be added into the pipeline file.

Then click on Add button. Task automatically updates.

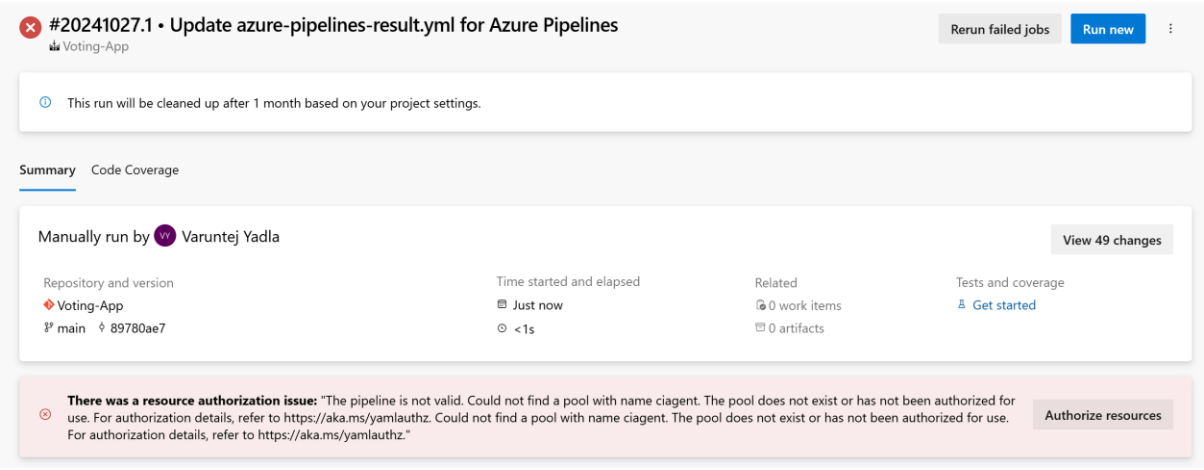
Copy the whole stage and paste it to add another stage for push.

```
- stage: Push
displayName: Push
jobs:
- job: Push
displayName: Push the Image
steps:
Settings
- task: Docker@2
displayName: Build
inputs:
containerRegistry: '$(dockerRegistryServiceConnection)'
repository: '$(imageRepository)'
command: 'build'
Dockerfile: 'result/Dockerfile'
tags: '$(tag)'
```

Click on settings button and update the command from build to push for Build task.

```
41  - stage: Push
42  displayName: Push
43  jobs:
44  - job: Push
45  displayName: Push
46  steps:
        Settings
47  - task: Docker@2
48  displayName: Push the Image
49  inputs:
50  containerRegistry: '$(dockerRegistryServiceConnection)'
51  repository: '$(imageRepository)'
52  command: 'push'
53  tags: '$(tag)'
```

Click on Save and Run:

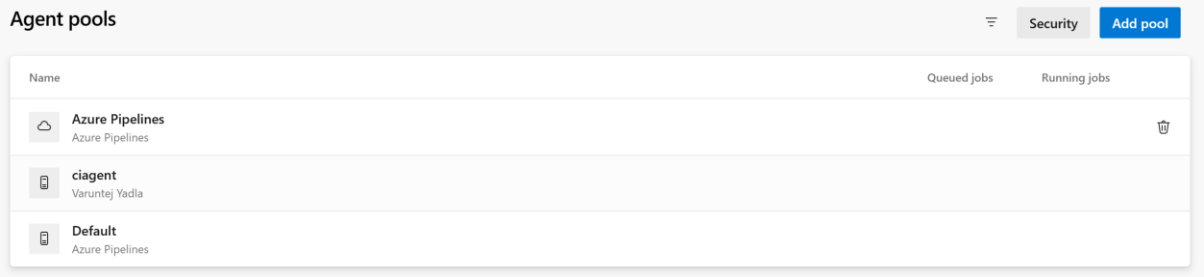


Now, we have to make a connection to the pipeline and VM we created.

Documentation: https://learn.microsoft.com/en-us/azure/devops/pipelines/agents/linux-agent?view=azure-devops

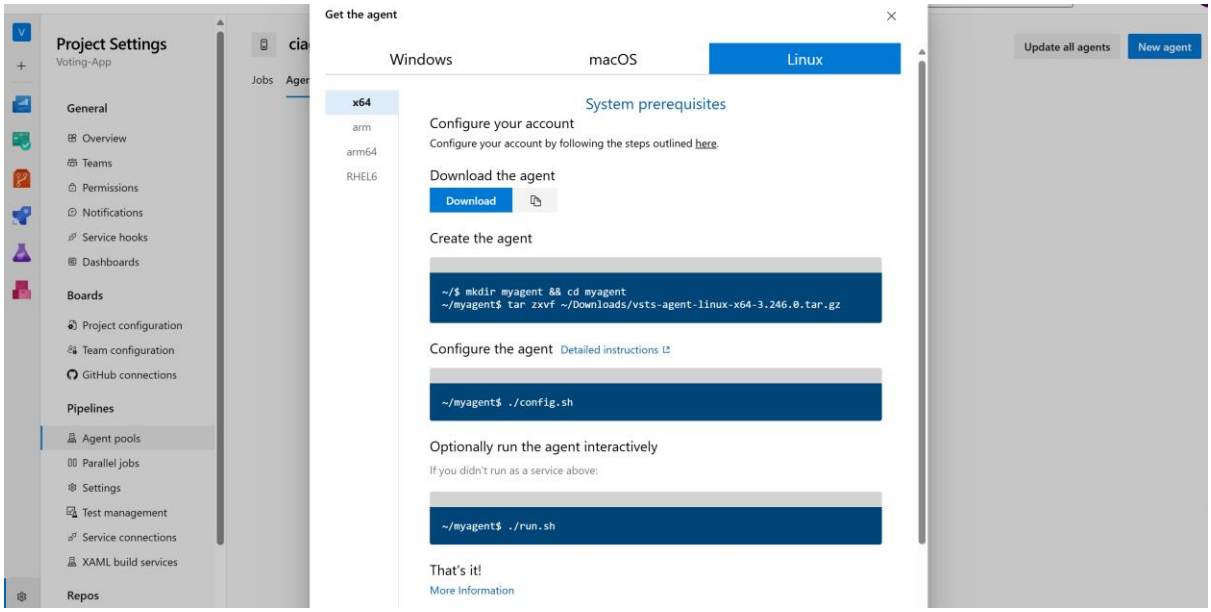
Follow this documentation to add the VM we created in the pool. For our Project we will go with Project settings in the bottom of the screen inside the project page (Voting-App) → Agent Pools → Click on add pool

Once pool is added it will look like this:



Click on the Agent → Again click on Agent and then click on New Agent

We need to copy these commands and paste in our VM we created as a agent.



Connect to the VM: Using Public IP when SSH.

```
yvaru@Tej MINGW64 ~
$ pwd
/c/Users/yvaru

yvaru@Tej MINGW64 ~
$ cd Downloads/

yvaru@Tej MINGW64 ~/Downloads
$ ls -al | grep 'cia'
-rw-r--r-- 1 yvaru 197121      2494 Oct 28 01:36 ciagent_key.pem

yvaru@Tej MINGW64 ~/Downloads
$ ssh -i ciagent_key.pem azureuser@52.172.99.184
The authenticity of host '52.172.99.184 (52.172.99.184)' can't be established.
ED25519 key fingerprint is SHA256:3gIeQhqcNhyK2TnSi64Q/2AX5tLmeD5NiT2wXKHbUUc.
This key is not known by any other names.
```

give **sudo apt update** command first.

```
azureuser@ciagent:~$ mkdir myagent && cd myagent
azureuser@ciagent:~/myagent$ wget https://vstsagentpackage.azureedge.net/agent/3.246.0/vsts-agent-linux-x64-3.246.0.tar.gz
--2024-10-27 20:30:31--  https://vstsagentpackage.azureedge.net/agent/3.246.0/vsts-agent-linux-x64-3.246.0.tar.gz
Resolving vstsagentpackage.azureedge.net (vstsagentpackage.azureedge.net)... 117.18.232.200, 2606:2800:147:120f:30c:1ba0:fc6:265a
Connecting to vstsagentpackage.azureedge.net (vstsagentpackage.azureedge.net)|117.18.232.200|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 146311562 (140M) [application/octet-stream]
Saving to: 'vsts-agent-linux-x64-3.246.0.tar.gz'

vsts-agent-linux-x64-3.246.0.tar.gz     100%[=============================================================================>] 139.53M  95.5MB/s    in 1.5s

2024-10-27 20:30:33 (95.5 MB/s) - 'vsts-agent-linux-x64-3.246.0.tar.gz' saved [146311562/146311562]

azureuser@ciagent:~/myagent$
```

Before giving the second command in the create agent, give wget URL (Copy it form Download the agent from Azure DevOps Portal in above image)

Then give the second command: Remove the part before vsts-agent……

```
azureuser@ciagent:~/myagent$ tar zxvf vsts-agent-linux-x64-3.246.0.tar.gz
```

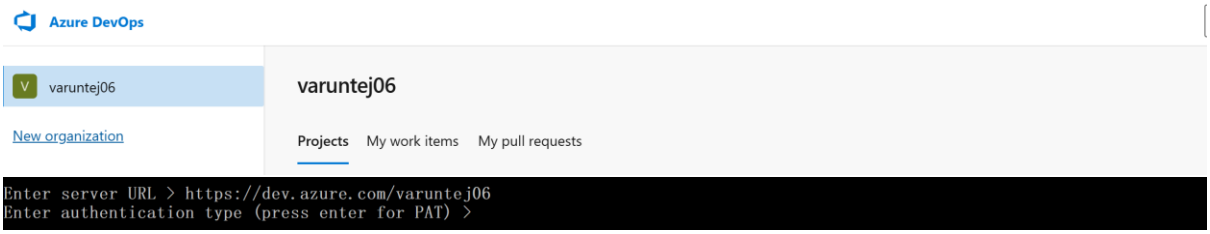
Then give the next command:

```
azureuser@ciagent:~/myagent$ ./config.sh

  ___                      ______ _            _ _
 / _ \                     | ___ (_)          | (_)
/ /_\ \_____   _ _ __ ___  | |_/ /_ _ __   ___| |_ _ __   ___  ___
|  _  |_  / | | | '__/ _ \ |  __/| | '_ \ / _ \ | | '_ \ / _ \/ __|
| | | |/ /| |_| | | |  __/ | |   | | |_) |  __/ | | | | |  __/\__ \
\_| |_/___|\__,_|_|  \___| \_|   |_| .__/ \___|_|_|_| |_|\___||___/
                                   | |
                                   |_|

       agent v3.246.0                (commit 59c8fe5)

>> End User License Agreements:

Building sources from a TFVC repository requires accepting the Team Explorer Everywhere End User License Agreement. This step is not required for building sou
rces from Git repositories.

A copy of the Team Explorer Everywhere license agreement can be found at:
  /home/azureuser/myagent/license.html

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

>> Connect:

Enter server URL >
```

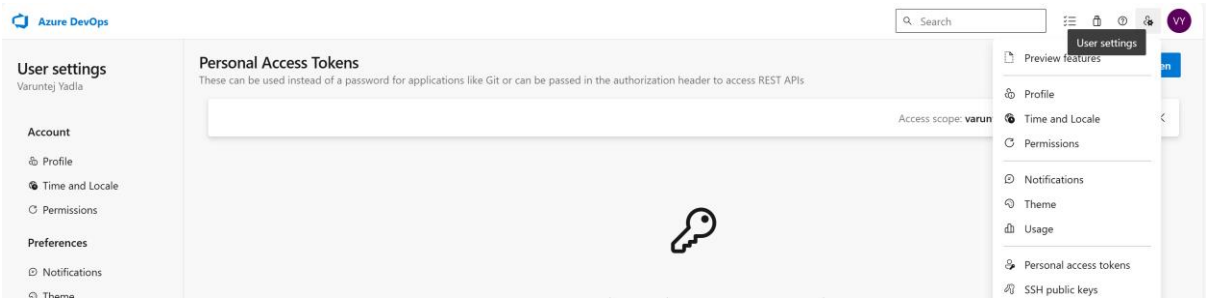
Type Y and then add server URL: Modify the Organisation name in the URL.

URL we can see what to give in the documentation.

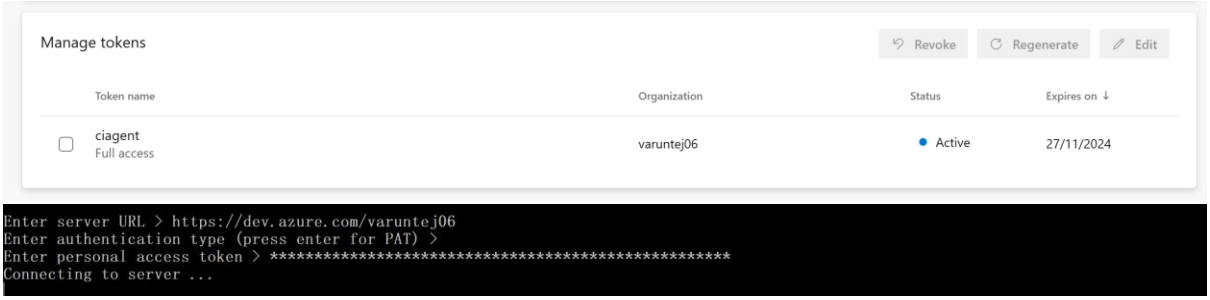
My organisation name:



For PAT: Click on User settings → PAT



Click on New Token and give full access to this token and make sure not to miss copying the code.



Hit Enter and then paste the PAT and then again hit enter key.

It is asking for agent pool: give the agent pool you created before:

It is asking for agent name. It is also same so press enter.

press enter.

Run the last command:

```
azureuser@ciagent:~/myagent$ ./run.sh
Scanning for tool capabilities.
Connecting to the server.
2024-10-27 20:49:39Z: Listening for Jobs
```

We are building our CI build from docker. So, we need to install docker in the VM agent. We missed it lets install it.

```
azureuser@ciagent:~/myagent$ sudo apt install docker.io
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following additional packages will be installed:
  bridge-utils containerd dns-root-data dnsmasq-base pigz runc ubuntu-fan
```

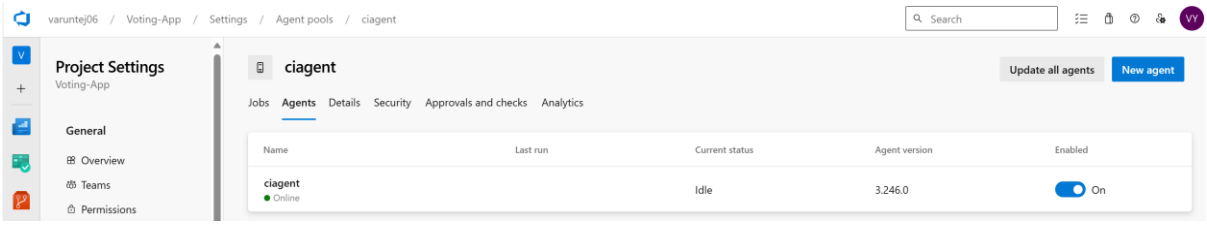
We are not using the root user for this VM. We configured the agent using this azureuser which is the default user for this machine. So, we have to run below commands.

```
azureuser@ciagent:~/myagent$ sudo usermod -aG docker azureuser
azureuser@ciagent:~/myagent$ sudo systemctl restart docker
```

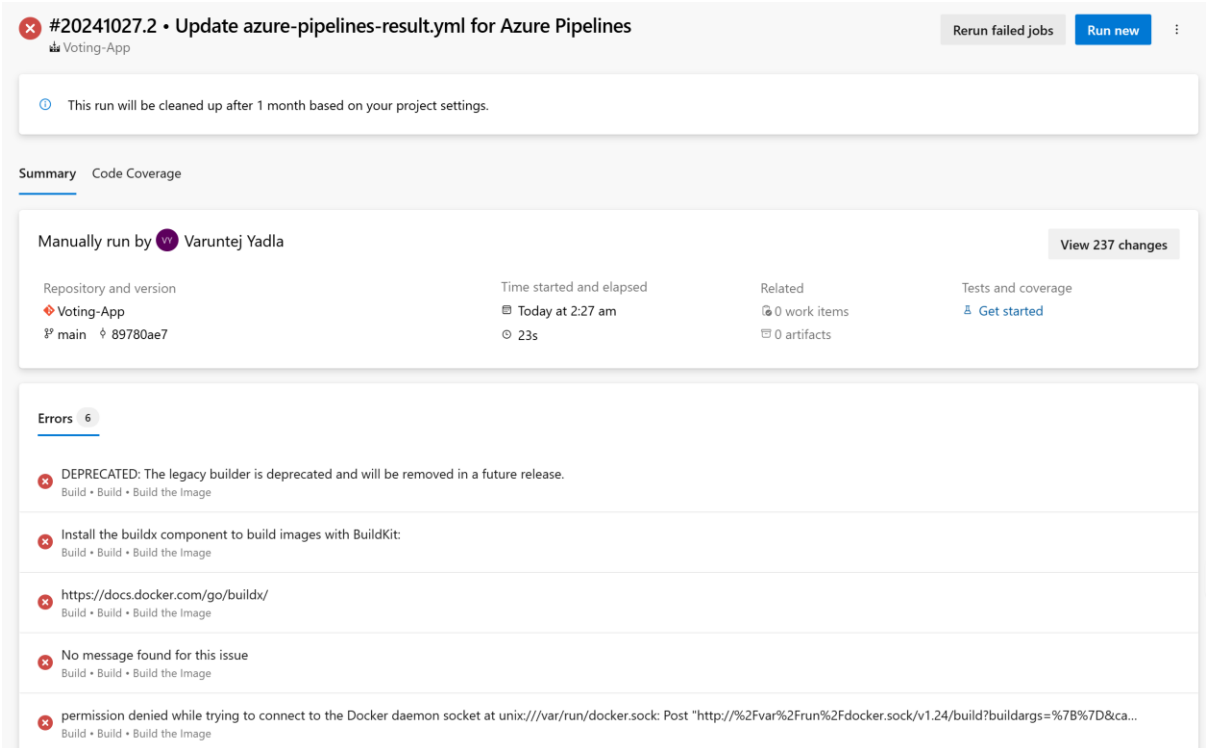
Now, start the agent again:

```
azureuser@ciagent:~/myagent$ ./run.sh
Scanning for tool capabilities.
Connecting to the server.
2024-10-27 20:55:45Z: Listening for Jobs
```

Agent is added in the agent pool:



Go to the pipeline: Run the pipline again.




```
azureuser@ciagent:~/myagent$ sudo systemctl restart docker
azureuser@ciagent:~/myagent$ ./run.sh
Scanning for tool capabilities.
Connecting to the server.
2024-10-27 20:55:45Z: Listening for Jobs
2024-10-27 20:57:37Z: Running job: Build
2024-10-27 20:57:58Z: Job Build completed with result: Failed
```

Build failed as it is not able to connetc to the Docker deamon.

Lets try to diconnect the VM, connect again and restart the docker service and check.

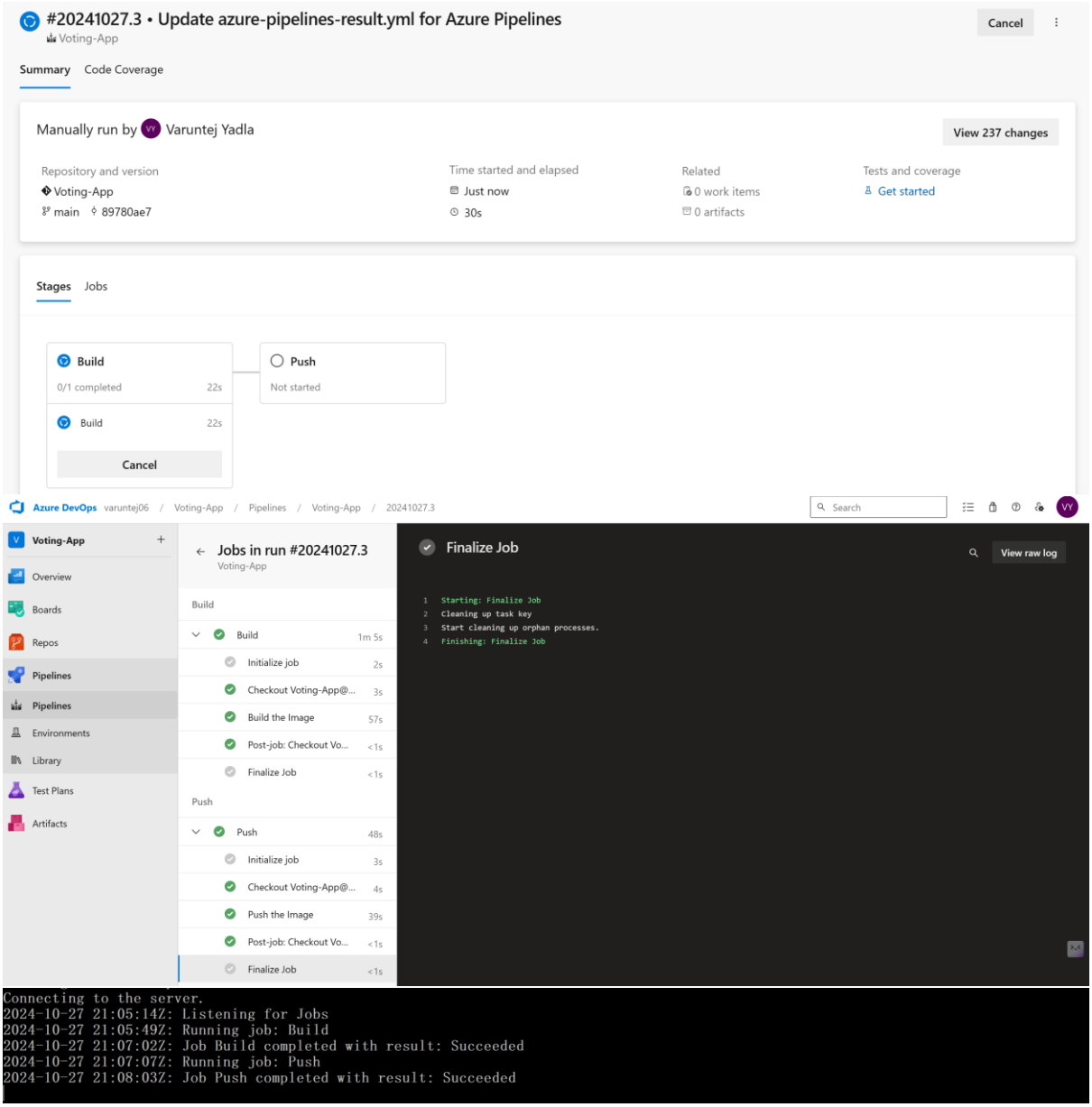
```
azureuser@ciagent:~/myagent$ logout
Connection to 52.172.99.184 closed.

yvaru@Tej MINGW64 ~/Downloads
$ ssh -i ciagent_key.pem azureuser@52.172.99.184
Welcome to Ubuntu 24.04.1 LTS (GNU/Linux 6.8.0-1016-azure x86_64)
```

It is able to connect with docker deamon now:

```
azureuser@ciagent:~$ ls
myagent
azureuser@ciagent:~$ cd
azureuser@ciagent:~$ cd myagent/
azureuser@ciagent:~/myagent$ ls
_diag  _work  bin  config.sh  env.sh  externals  license.html  reauth.sh  run-docker.sh  run.sh  svc.sh  vsts-agent-linux-x64-3.246.0.tar.gz
azureuser@ciagent:~/myagent$ sudo systemctl restart docker
azureuser@ciagent:~/myagent$ sudo docker pull hello-world
Using default tag: latest
latest: Pulling from library/hello-world
Digest: sha256:d211f485f2dd1dee407a80973c8f129f00d54604d2c90732e8e320e5038a0348
Status: Image is up to date for hello-world:latest
docker.io/library/hello-world:latest
azureuser@ciagent:~/myagent$ docker ps
CONTAINER ID   IMAGE     COMMAND   CREATED   STATUS    PORTS     NAMES
azureuser@ciagent:~/myagent$
```

```
azureuser@ciagent:~/myagent$ ./run.sh
Scanning for tool capabilities.
Connecting to the server.
2024-10-27 21:05:14Z: Listening for Jobs
```

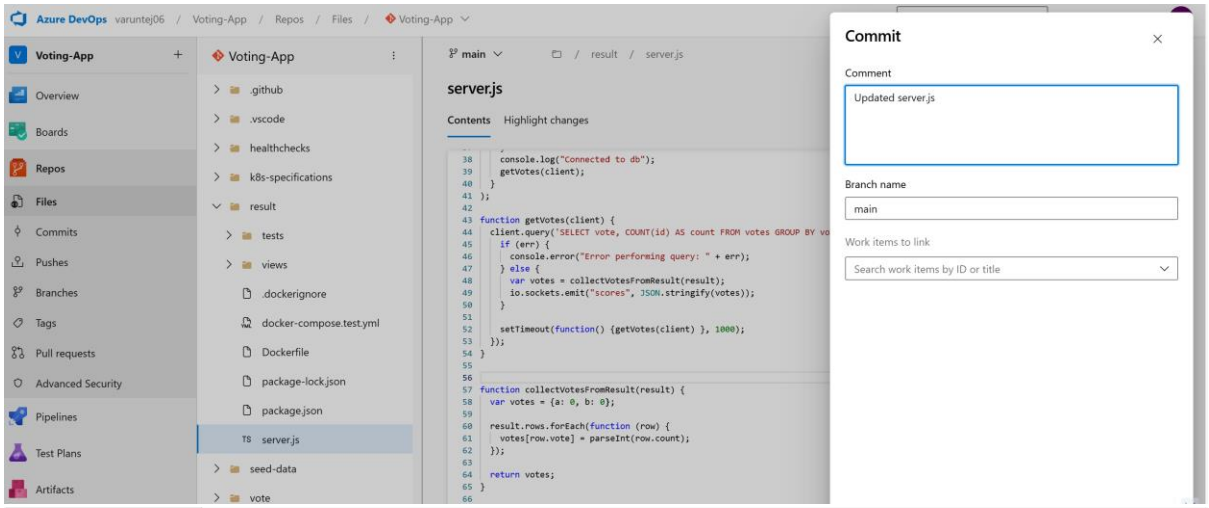
Now run the pipeline again:

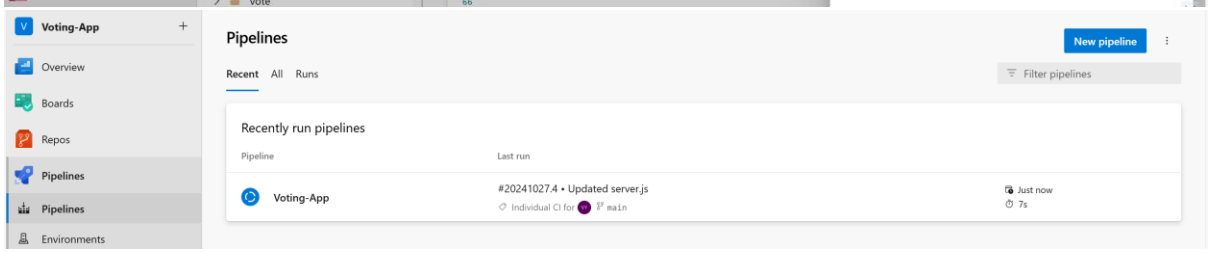



```
Connecting to the server.
2024-10-27 21:05:14Z: Listening for Jobs
2024-10-27 21:05:49Z: Running job: Build
2024-10-27 21:07:02Z: Job Build completed with result: Succeeded
2024-10-27 21:07:07Z: Running job: Push
2024-10-27 21:08:03Z: Job Push completed with result: Succeeded
```

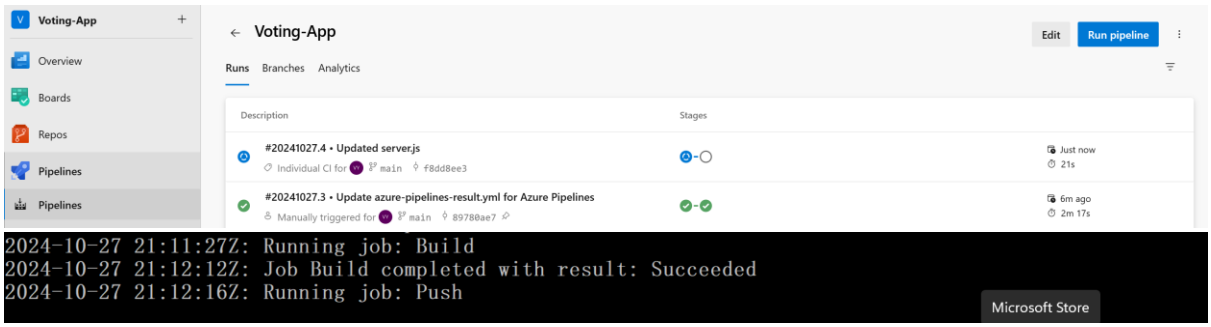
Now, you can modify some file in worker folder and check pipeline is triggering or not. It should not trigger, but if we change something in result folder it should trigger.

I added an empty line in server.js file:

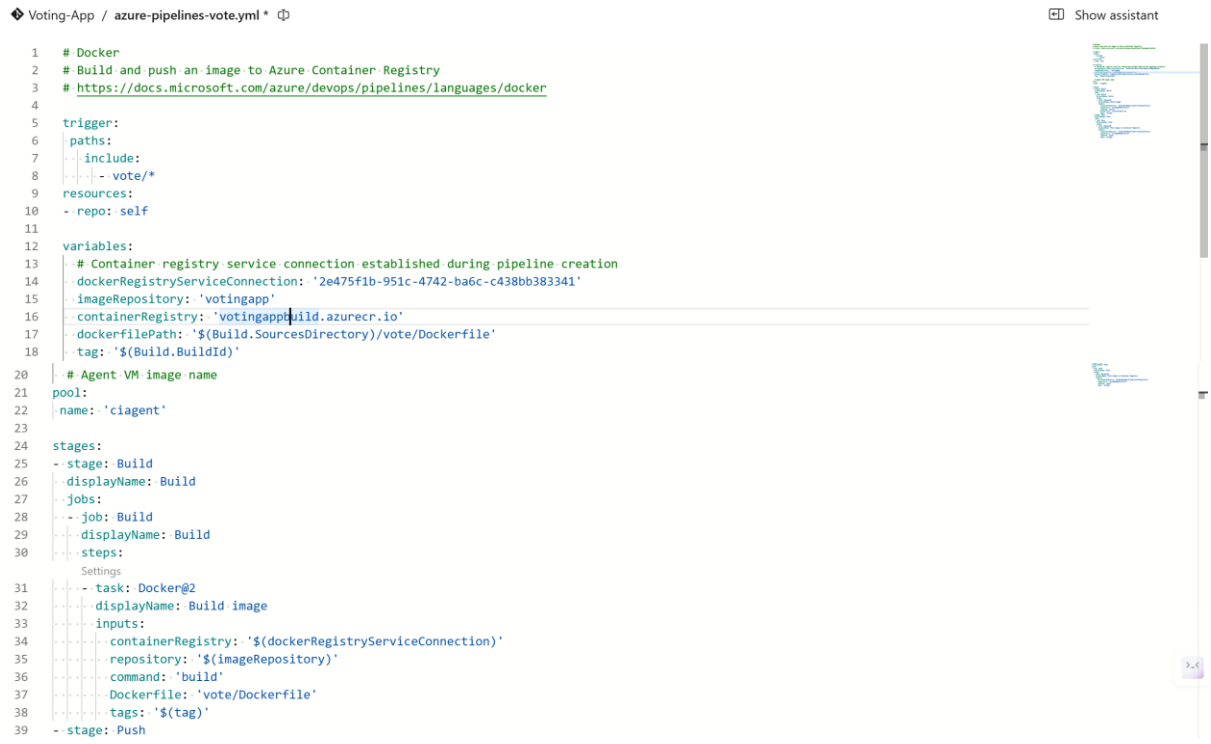




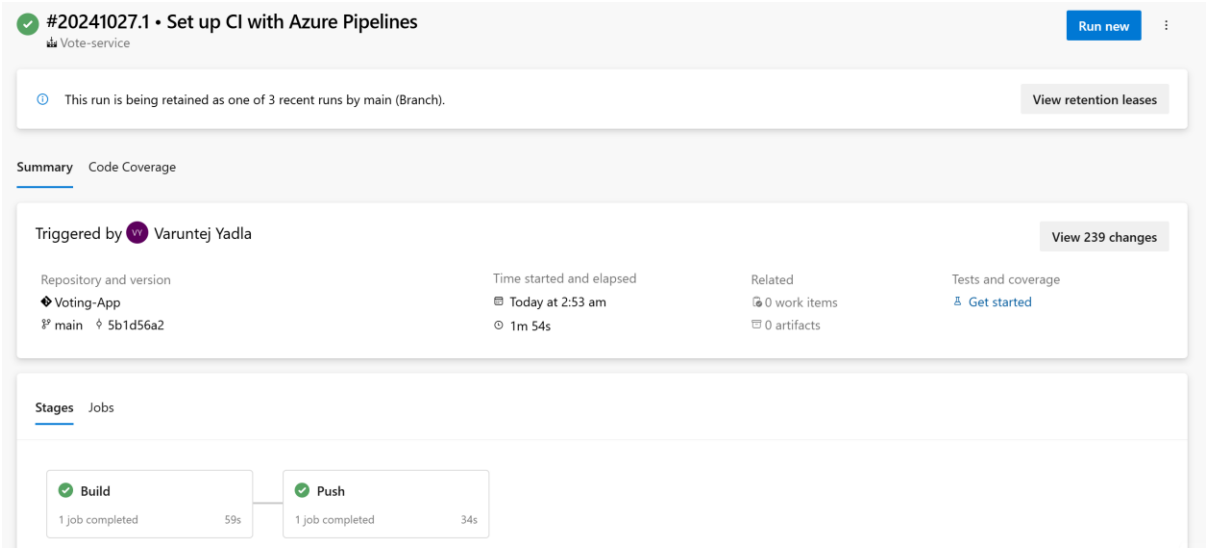
Run started:



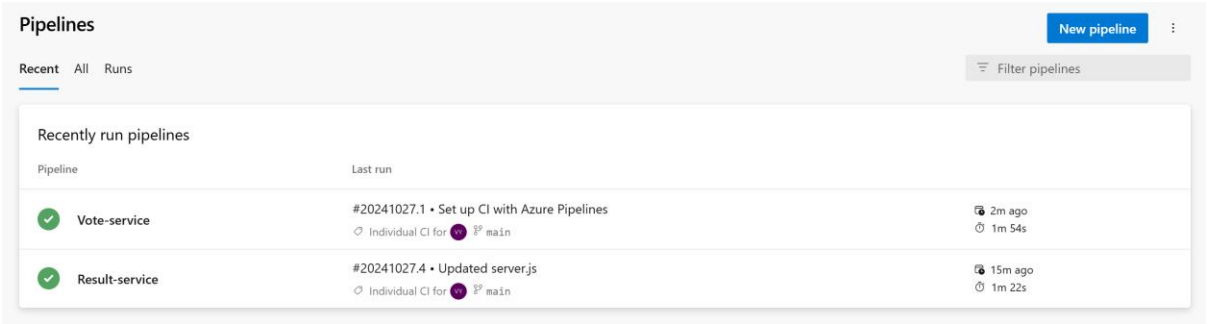
Now add pipeline for vote and worker as well in the same process:



```yaml
- stage: Push
  displayName: Push
  jobs:
  - job: Push
    displayName: Push
    steps:
    Settings
    - task: Docker@2
      displayName: Push image to Container Registry
      inputs:
        containerRegistry: '$(dockerRegistryServiceConnection)'
        repository: '$(imageRepository)'
        command: 'push'
        tags: '$(tag)'
```



Rename the pipelines: Click on three dots on the pipeline and select rename:



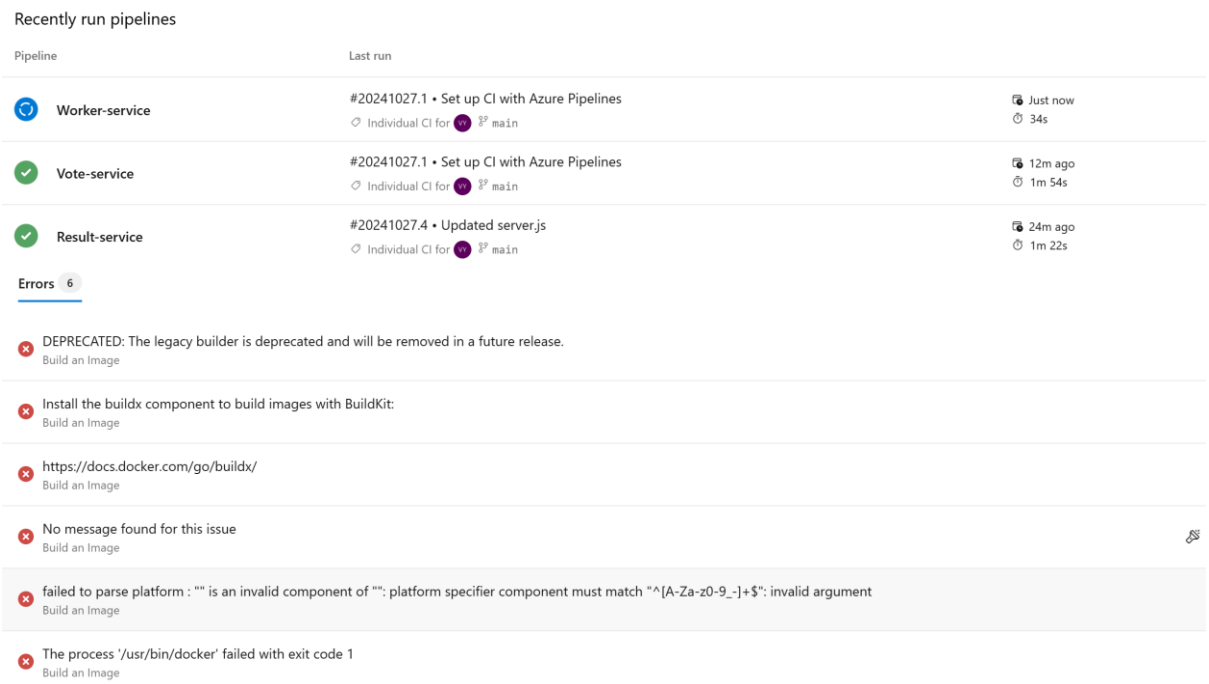
Creating pipeline for worker:

Voting-App / **azure-pipelines-worker.yml** *

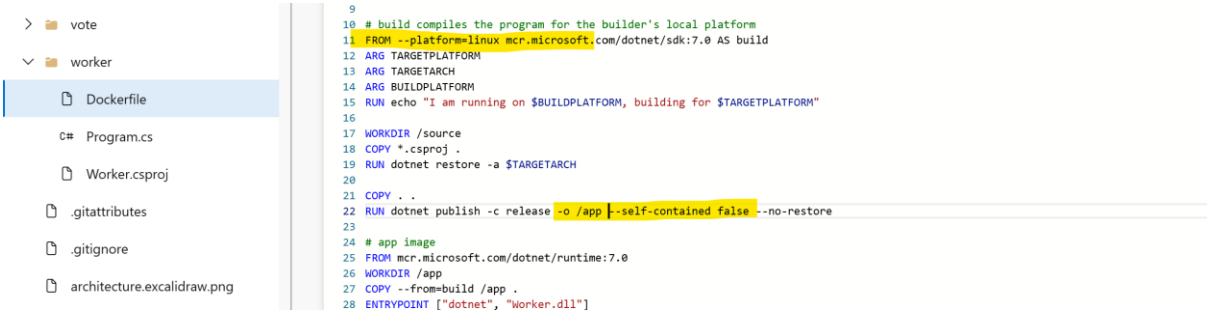
```yaml
# Docker
# Build and push an image to Azure Container Registry
# https://docs.microsoft.com/azure/devops/pipelines/languages/docker

trigger:
  paths:
    include:
      - worker/*
resources:
- repo: self

variables:
  # Container registry service connection established during pipeline creation
  dockerRegistryServiceConnection: 'ffaea5df-46d6-4d00-8c4b-c537f41f40db'
  imageRepository: 'votingapp'
  containerRegistry: 'votingappbuild.azurecr.io'
  dockerfilePath: '$(Build.SourcesDirectory)/worker/Dockerfile'
  tag: '$(Build.BuildId)'
```

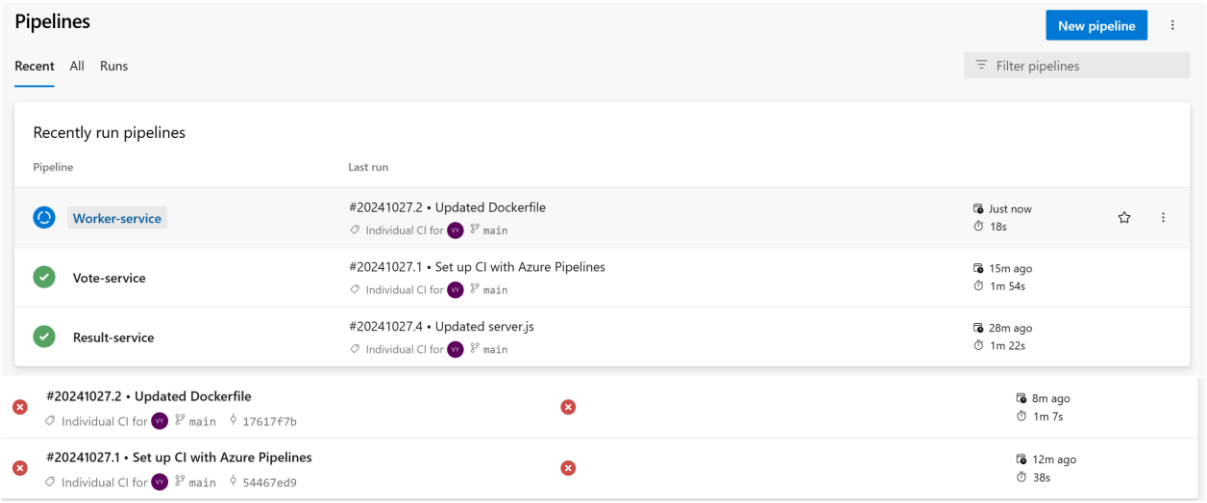
```yaml
    # Agent VM image name
pool:
  name: 'ciagent'
stages:
- stage: Build
  displayName: Build
  jobs:
  - job: Build
    displayName: Build
    steps:
      Settings
    - task: Docker@2
      displayName: Build an Image
      inputs:
        containerRegistry: '$(dockerRegistryServiceConnection)'
        repository: '$(imageRepository)'
        command: 'build'
        Dockerfile: 'worker/Dockerfile'
        tags: '$(tag)'
```

We added till build stage only. It will fail as there are some env variables defined in the docker file of this service.

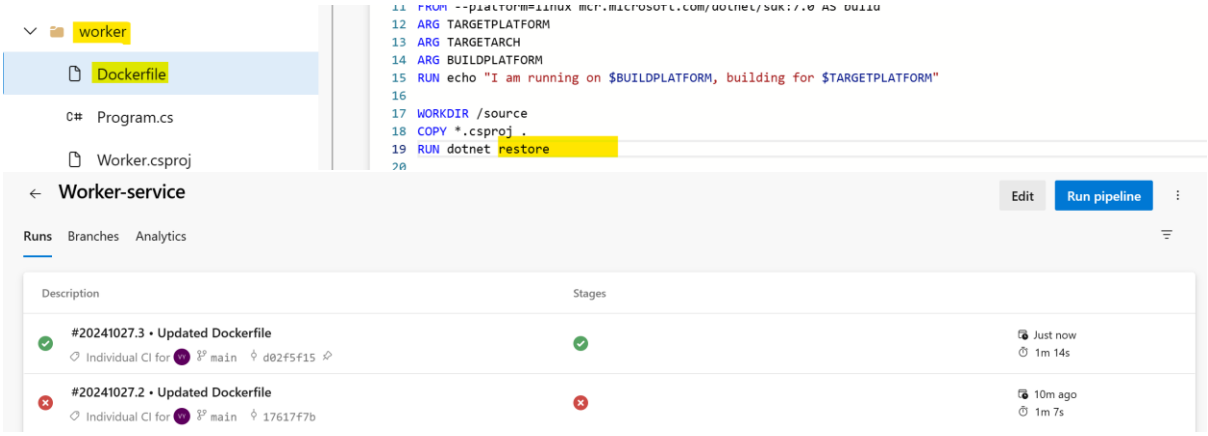


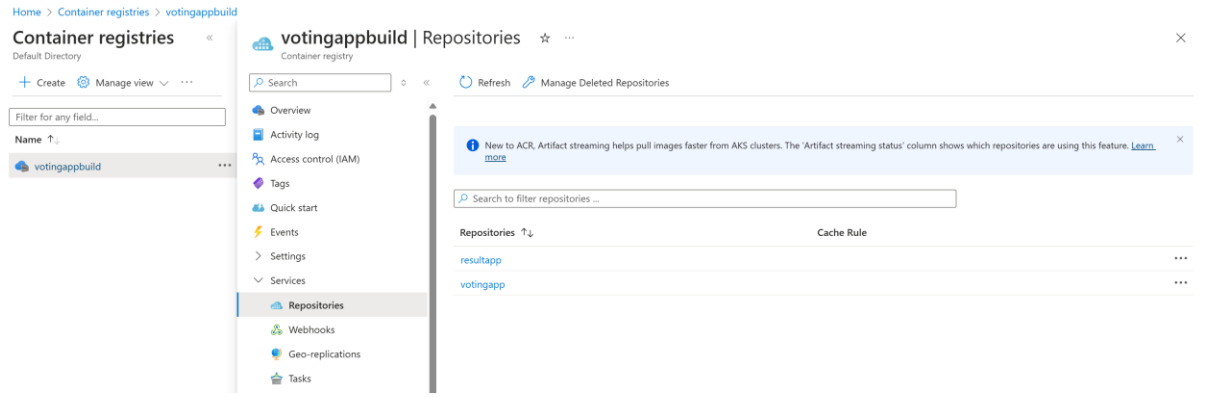
Do these two changes: Commit these changes and pipeline should auto trigger.



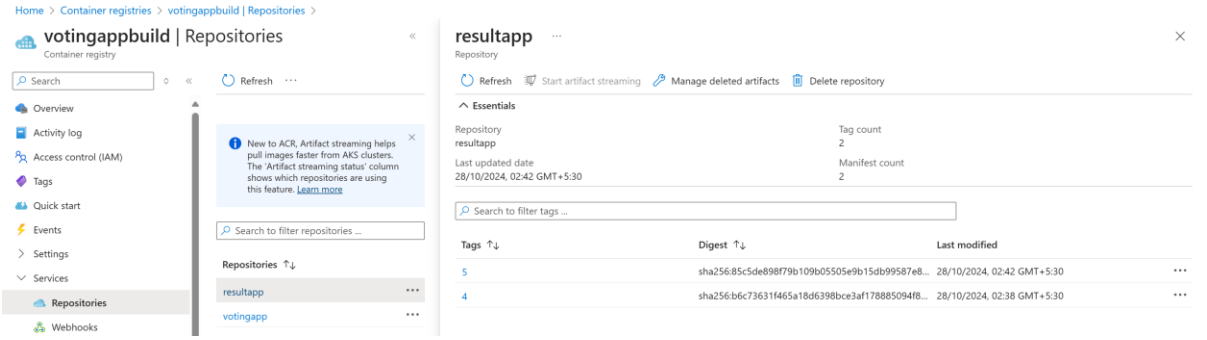


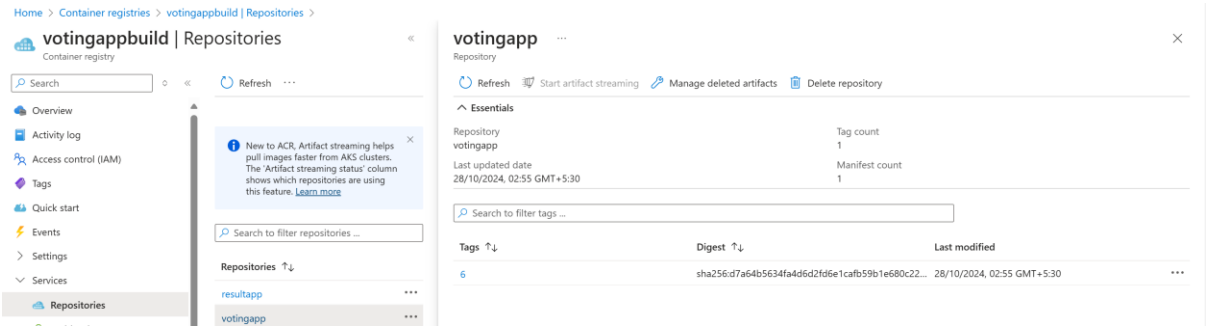
Failed: Remove everything after restore in line 19. Commit changes & pipeline triggers automatically.





We had two repositories in the container registry for two frontend services. We can see the images of those services.

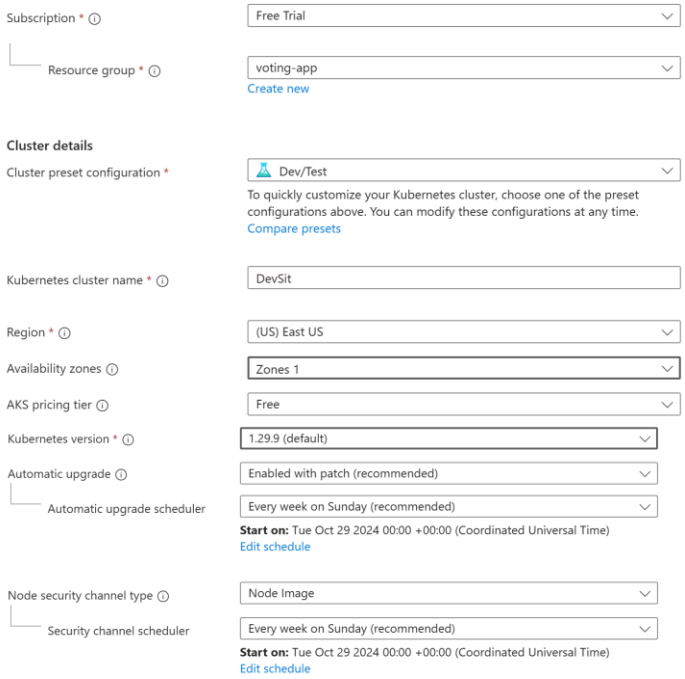




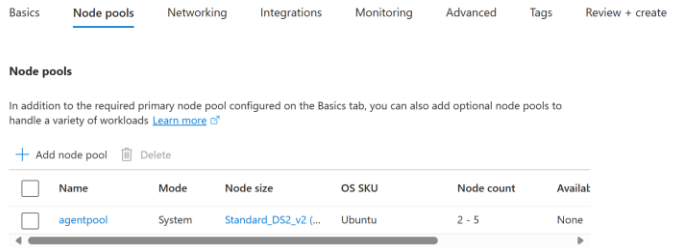
Day 15:

Create a Azure Kubernetes service:

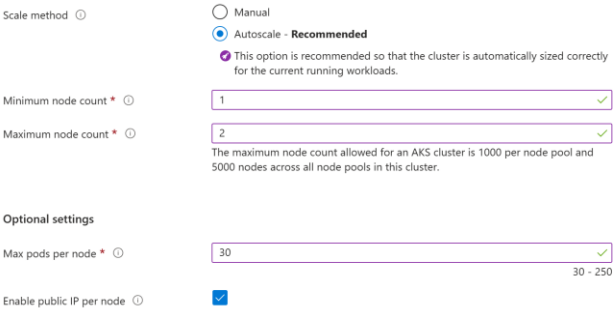
Go to Kubernetes service and click on create → Kubernetes cluster



We need agent pool to our AKS cluster. AKS is services managed by Azure but the pods that we have to run need nodes.

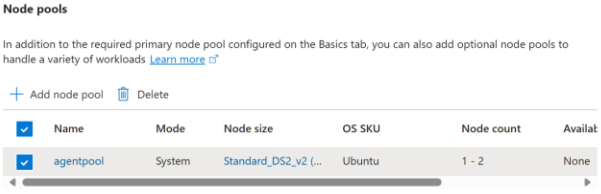


Click on agentpool and modify configurations:



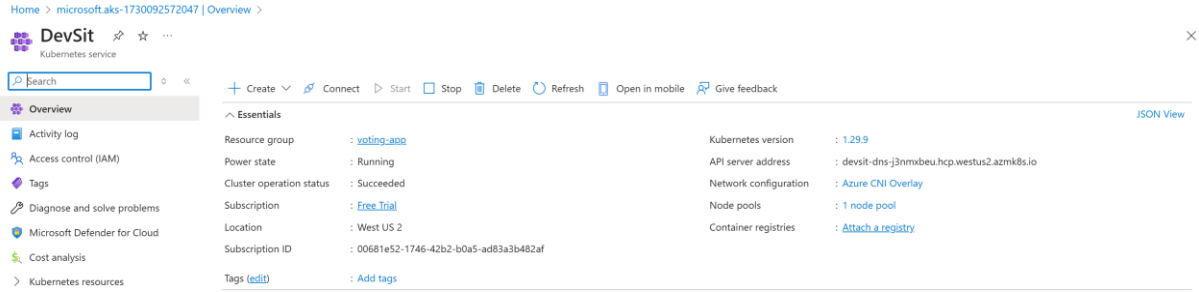
We don’t need much nodes ofr this project and 30 pods per node is also fine. Also enable the public IP per node. Then click on Update.

Select the agent pool:



We can also setup multiple agent pools. Then click on Review + create: Then click on create.

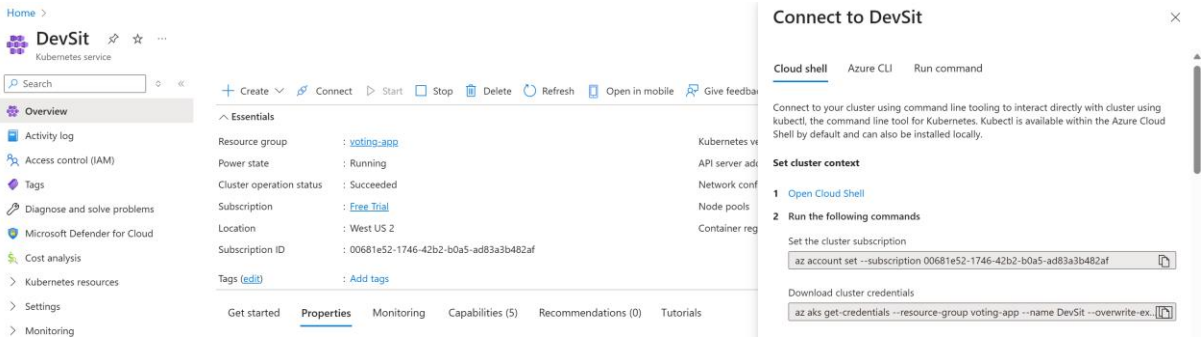
Faced VM size unavailability in us east region changed to us west region.



Connect to this cluster from terminal or Cloud shell:

Azure CLI and Kubectl to be installed in local PC if want to connect from local terminal.

We can connect through cloushell:



Connecting to AKS cluster from terminal:

```
yvaru@Tej MINGW64 ~
$ az aks get-credentials --resource-group voting-app --name DevSit --overwrite-existing
Merged "DevSit" as current context in C:\Users\yvaru\.kube\config

yvaru@Tej MINGW64 ~
$ kubectl version
Client Version: v1.31.2
Kustomize Version: v5.4.2
Server Version: v1.29.9
WARNING: version difference between client (1.31) and server (1.29) exceeds the supported minor version skew of +/-1

yvaru@Tej MINGW64 ~
$ kubectl get pods
No resources found in default namespace.

yvaru@Tej MINGW64 ~
$
```

Now, we need to install Argo CD in the cluster:

Documentation: https://argo-cd.readthedocs.io/en/stable/getting_started/

Installing Argo CD in argocd namespace where all Argo CD services and application resources will be live.

```
yvaru@Tej MINGW64 ~
$ kubectl create namespace argocd
kubectl apply -n argocd -f https://raw.githubusercontent.com/argoproj/argo-cd/stable/manifests/install.yaml
namespace/argocd created
customresourcedefinition.apiextensions.k8s.io/applications.argoproj.io created
customresourcedefinition.apiextensions.k8s.io/applicationsets.argoproj.io created
customresourcedefinition.apiextensions.k8s.io/appprojects.argoproj.io created
serviceaccount/argocd-application-controller created
serviceaccount/argocd-applicationset-controller created
```

Lot of argocd resources are created in the namespace.

```
yvaru@Tej MINGW64 ~
$ kubectl get pods -n argocd
NAME                                                READY   STATUS    RESTARTS   AGE
argocd-application-controller-0                     1/1     Running   0          3m4s
argocd-applicationset-controller-5b899f5459-h8mjf   1/1     Running   0          3m10s
argocd-dex-server-7f4f5476db-vwknn                  1/1     Running   0          3m9s
argocd-notifications-controller-75d8587699-4qbks    1/1     Running   0          3m8s
argocd-redis-5bc565bfc4-f2dkb                       1/1     Running   0          3m8s
argocd-repo-server-55c7764fd4-hj4vm                 1/1     Running   0          3m7s
argocd-server-5f6567647b-z2gsh                      1/1     Running   0          3m6s
```

Argo Cd in the cluster need to check the changes in K8s manifets files in Git Repo. So, we need to configure that. But before that we need to expose argocd server UI. And need to know the admin password of argocd.

```
yvaru@Tej MINGW64 ~
$ kubectl get secrets -n argocd
NAME                          TYPE     DATA   AGE
argocd-initial-admin-secret   Opaque   1      5m21s
argocd-notifications-secret   Opaque   0      5m49s
argocd-redis                  Opaque   1      5m23s
argocd-secret                 Opaque   5      5m48s

yvaru@Tej MINGW64 ~
$ kubectl edit secret argocd-initial-admin-secret -n argocd
Edit cancelled, no changes made.
```

```
# Please edit the object below. Lines beginning with a '#' will be ignored,
# and an empty file will abort the edit. If an error occurs while saving this file will be
# reopened with the relevant failures.
#
apiVersion: v1
data:
  password: MG9SNUNsc3FBb1J4LUVNUw==
kind: Secret
metadata:
  creationTimestamp: "2024-10-28T08:35:40Z"
  name: argocd-initial-admin-secret
  namespace: argocd
  resourceVersion: "46725"
  uid: 29e9ea7e-b85d-4ac2-bdb9-b69b6e9f4673
type: Opaque
```

Copy it. And decode it.

```
yvaru@Tej MINGW64 ~
$ echo MG9SNUNsc3FBb1J4LUVNUw== | base64 --decode
0oR5ClsqAoRx-EMS
yvaru@Tej MINGW64 ~
$
```

Copy this password. Now lets expose the argocd server to NodePort so that we can access in our browser. 0oR5ClsqAoRx-EMS

```
yvaru@Tej MINGW64 ~
$ kubectl get svc -n argocd
NAME                                       TYPE        CLUSTER-IP      EXTERNAL-IP   PORT(S)                      AGE
argocd-applicationset-controller           ClusterIP   10.0.139.152    <none>        7000/TCP,8080/TCP            9m36s
argocd-dex-server                          ClusterIP   10.0.147.22     <none>        5556/TCP,5557/TCP,5558/TCP   9m35s
argocd-metrics                             ClusterIP   10.0.182.46     <none>        8082/TCP                     9m34s
argocd-notifications-controller-metrics    ClusterIP   10.0.220.223    <none>        9001/TCP                     9m33s
argocd-redis                               ClusterIP   10.0.128.111    <none>        6379/TCP                     9m32s
argocd-repo-server                         ClusterIP   10.0.226.184    <none>        8081/TCP,8084/TCP            9m31s
argocd-server                              ClusterIP   10.0.227.106    <none>        80/TCP,443/TCP               9m30s
argocd-server-metrics                      ClusterIP   10.0.162.0      <none>        8083/TCP                     9m29s

yvaru@Tej MINGW64 ~
$ kubectl edit svc argocd-server -n argocd
```

```
    protocol: TCP
    targetPort: 8080
  selector:
    app.kubernetes.io/name: argocd-server
  sessionAffinity: None
  type: NodePort
status:
  loadBalancer: {}
```

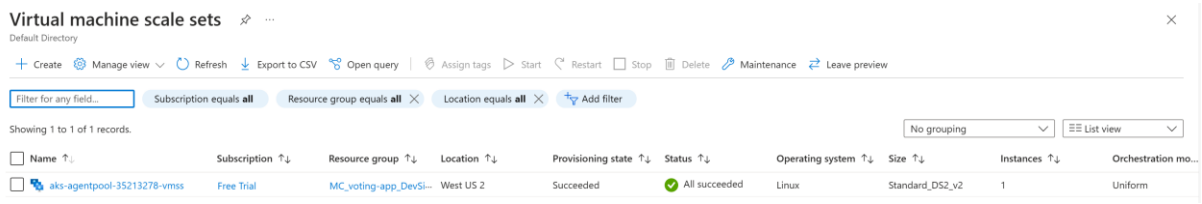
```
yvaru@Tej MINGW64 ~
$ kubectl get svc -n argocd
NAME                                       TYPE        CLUSTER-IP      EXTERNAL-IP   PORT(S)                      AGE
argocd-applicationset-controller           ClusterIP   10.0.139.152    <none>        7000/TCP,8080/TCP            11m
argocd-dex-server                          ClusterIP   10.0.147.22     <none>        5556/TCP,5557/TCP,5558/TCP   11m
argocd-metrics                             ClusterIP   10.0.182.46     <none>        8082/TCP                     11m
argocd-notifications-controller-metrics    ClusterIP   10.0.220.223    <none>        9001/TCP                     11m
argocd-redis                               ClusterIP   10.0.128.111    <none>        6379/TCP                     11m
argocd-repo-server                         ClusterIP   10.0.226.184    <none>        8081/TCP,8084/TCP            11m
argocd-server                              NodePort    10.0.227.106    <none>        80:31336/TCP,443:32317/TCP   11m
argocd-server-metrics                      ClusterIP   10.0.162.0      <none>        8083/TCP                     11m
```

So, now we need to know the node IP and port. Port we already knew 31336 from above pic.

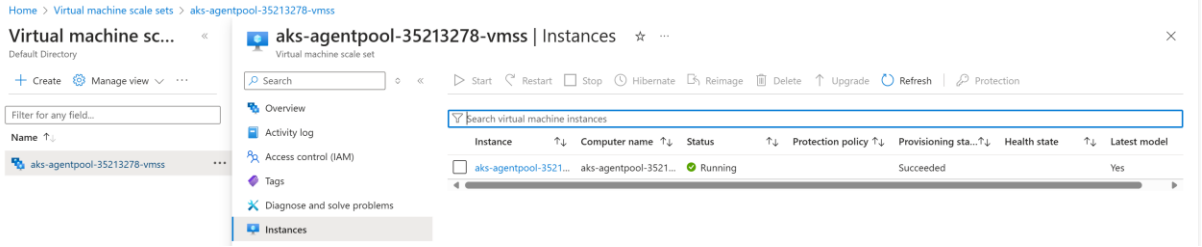
```
yvaru@Tej MINGW64 ~
$ kubectl get nodes -o wide
NAME                                STATUS   ROLES    AGE    VERSION   INTERNAL-IP    EXTERNAL-IP     OS-IMAGE             KERNEL-VERSION       CONTAINER-RUNTI
ME
aks-agentpool-35213278-vmss000000   Ready    <none>   3h29m  v1.29.9   10.224.0.4     51.143.91.198   Ubuntu 22.04.5 LTS   5.15.0-1073-azure    containerd://1.
7.22-1
```

Copy the External IP: 51.143.91.198: 31336 but we will not be able to access it. We need to open the ports for the AKS cluster node.

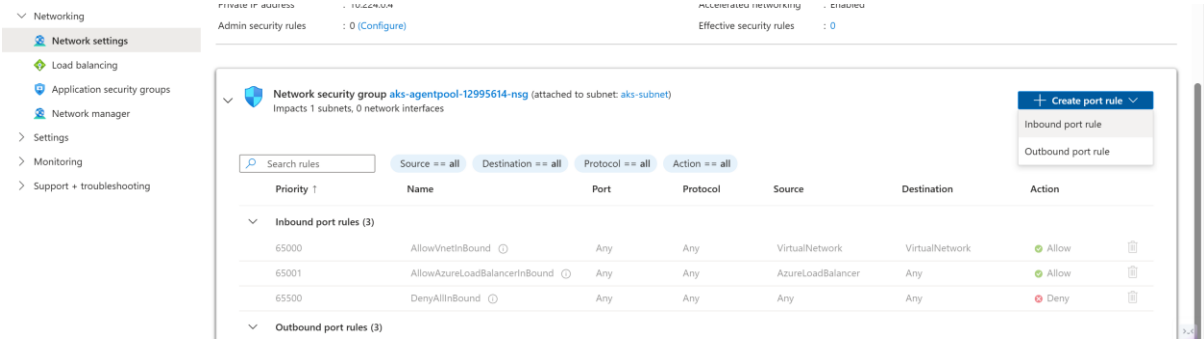
Go to azure portal and search for virtual machine scale sets:



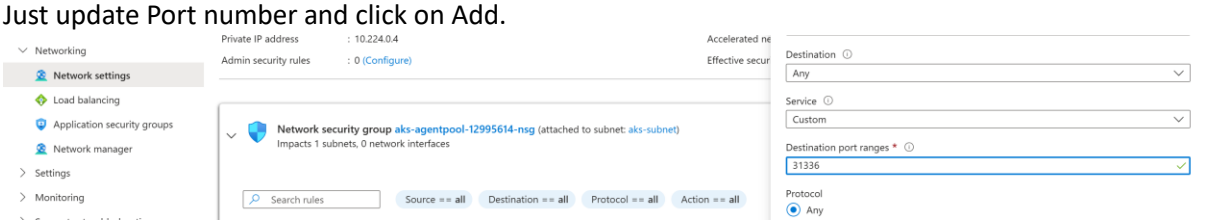
Click on Instances



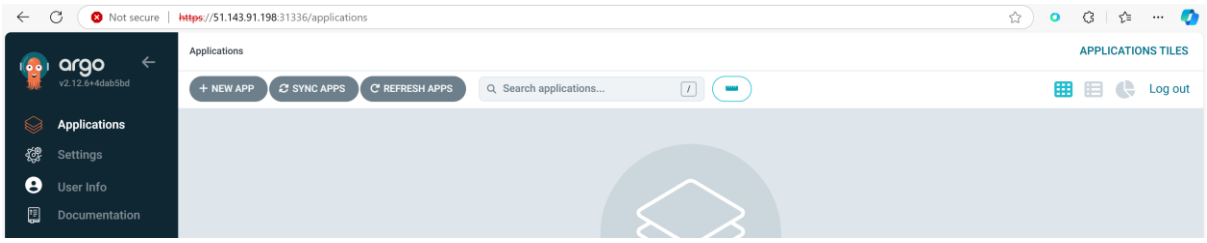
Now click on aks-agentpool: Click on network and open the port in inbound port rules:



Just update Port number and click on Add.



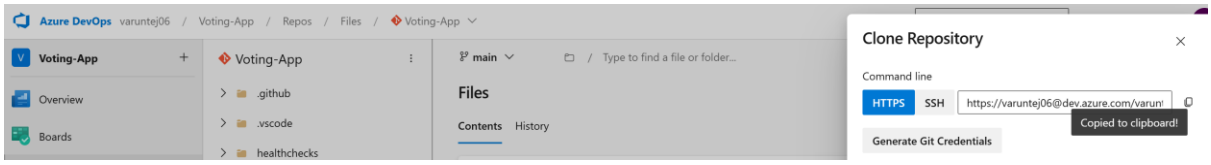
Now give the Node IP and port number: Give admin as user name and password of it.

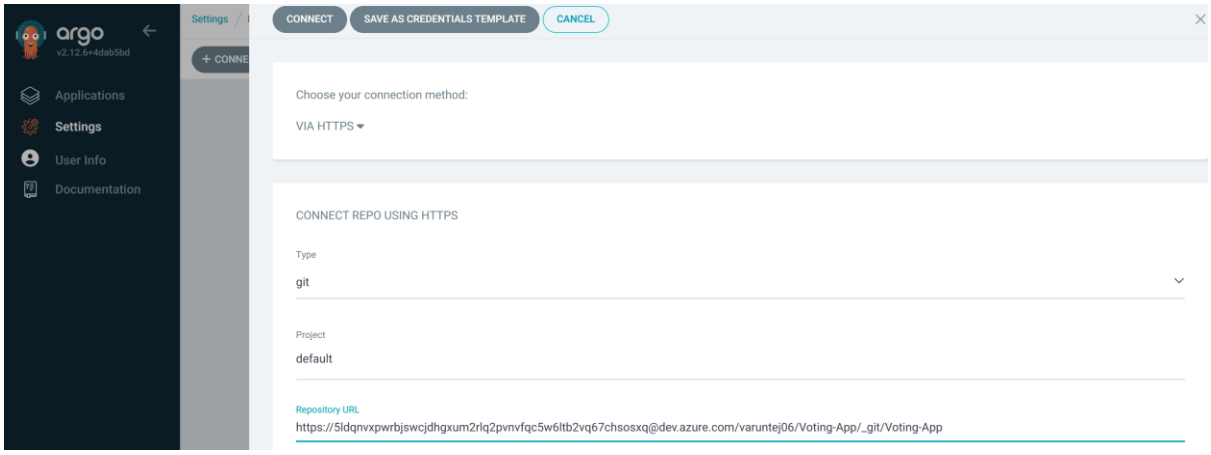


This Argocd can only read the changes in repos. So, we can create PAT that had read only permission while connecting this to repos.

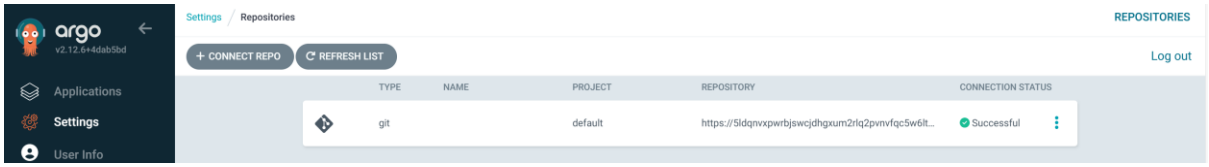
Now, create a PAT from azure DevOps and copy it.

Now in the ArgoCD click on Setting → Repositories → Connect Repo



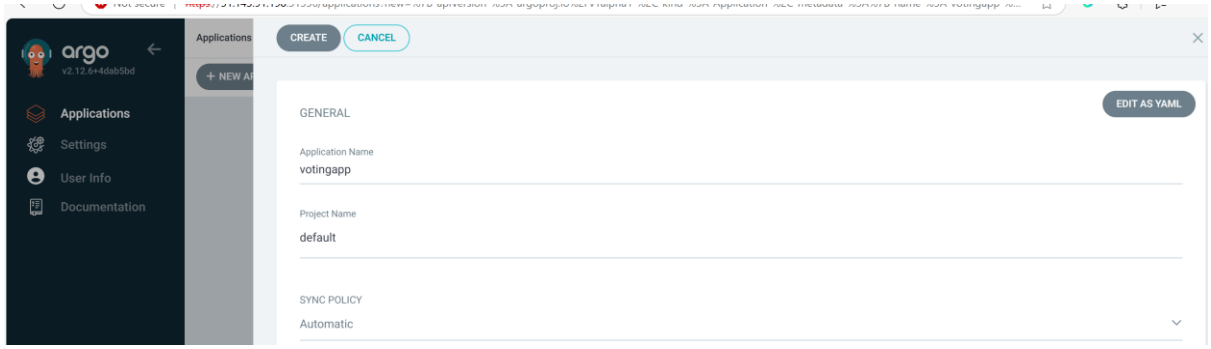


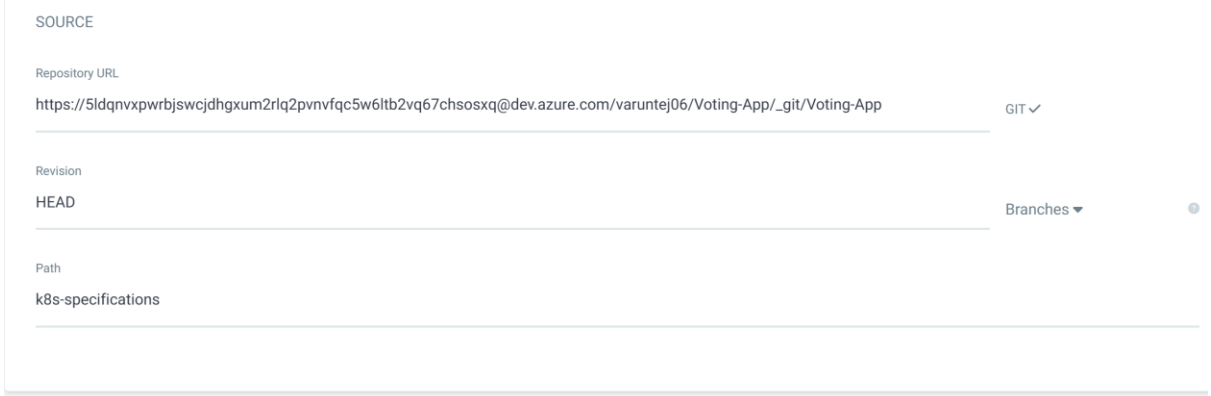
Add these details and in the repo URL you need to replace organisation name with access token.

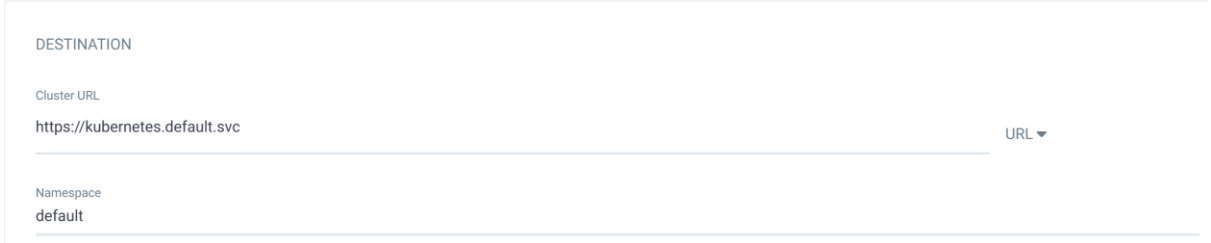


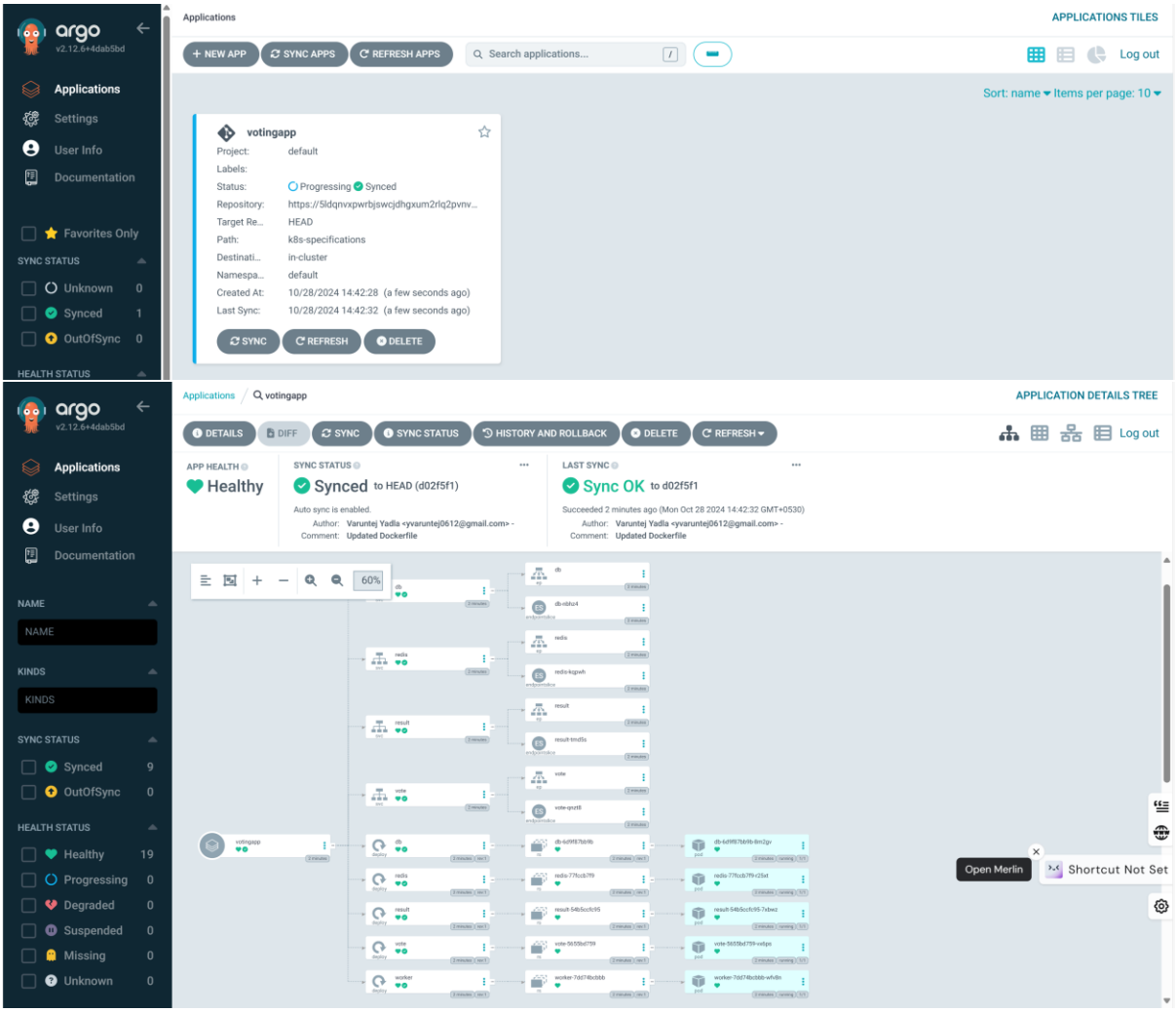
Now, we need to deploy the manifests to Cluster:

Go to applications → New Application → Application should be lowercase.









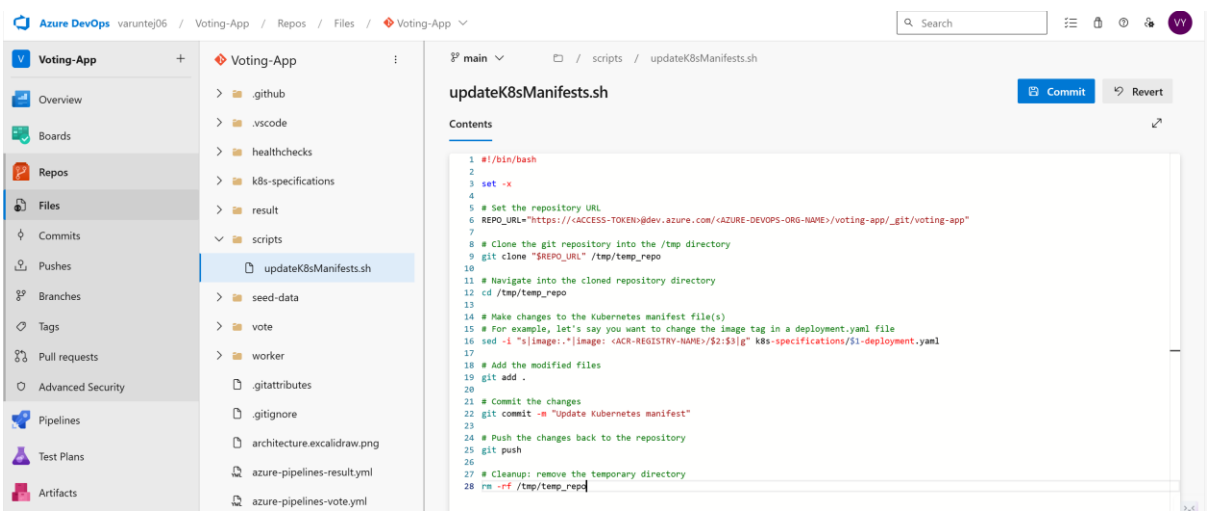
It deployed everything:

```
yvaru@Tej MINGW64 ~
$ kubectl get pods -n default
NAME                      READY   STATUS    RESTARTS   AGE
db-6d9f87bb9b-8m2gv       1/1     Running   0          3m38s
redis-77fccb7f9-r25xt     1/1     Running   0          3m38s
result-54b5ccfc95-7xbwz   1/1     Running   0          3m38s
vote-5655bd759-vx6ps      1/1     Running   0          3m38s
worker-7dd74bcbbb-wfv8n   1/1     Running   0          3m38s
```

It deployed the application, but it should also trigger deployments on changing something in repo. We need to implement CD now.

We will add a folder in the repo root level and add a script file which will update the manifest files on successful build.

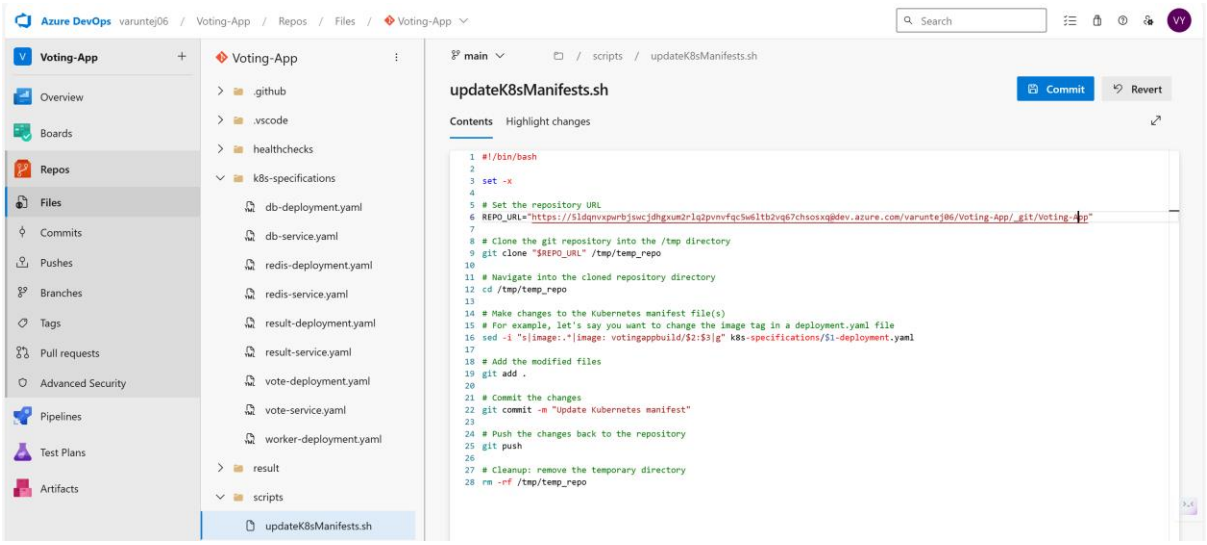
Script: https://github.com/VarunTej06/sampleScripts/blob/main/script.sh



Replace the access token in line 6. Use the same one that we used for argocd. Since we granted all

permissions to it. You can copy it from repositories from argocd where we added. Also change the organisation name in the same line. Alos update Project name properly at 2 places in the same line.

Now update the container registry name in line 16.



$1, $2, $3 are command line arguments passed to this script file from CI pipleine. Commit the file.

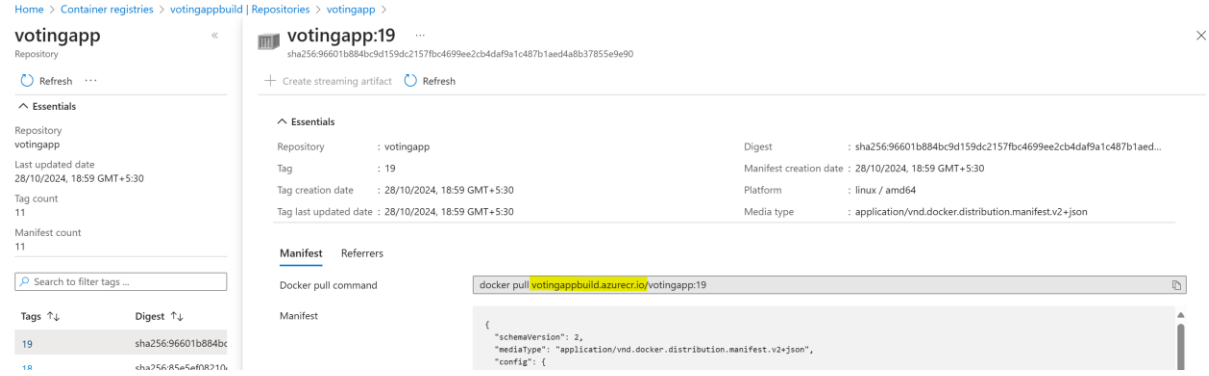
$1 is deployment file name, $2 is container registry repository name and $3 is tag name.

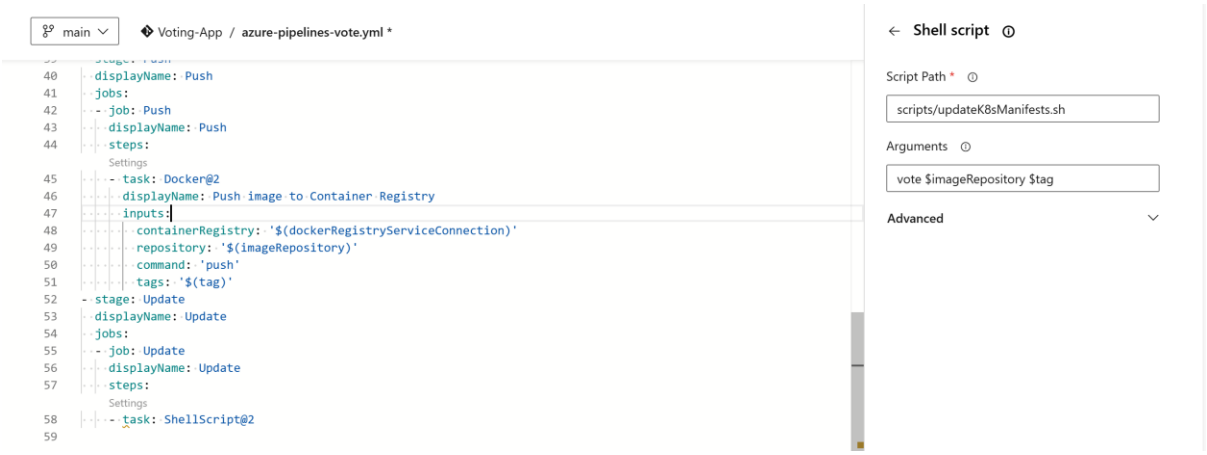
Lets try to Edit pipelines and add stage to trigger this script file from a new stage called update.

Note: This point added at the end of the project when faced an issue. Here we mentioned image name wrongly that needs to be updated.

```
13 # Make changes to the Kubernetes manifest file(s)
14 # For example, let's say you want to change the image tag in a deployment.yaml file
15 sed -i "s|image:.*|image: votingappbuild.azurecr.io/$2:$3|g" k8s-specifications/$1-deployment.yaml
16
17 # Add the modified files
```

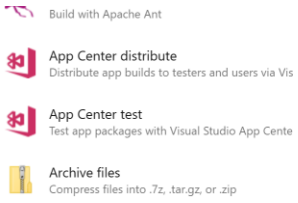
To know how we can give this correctly:





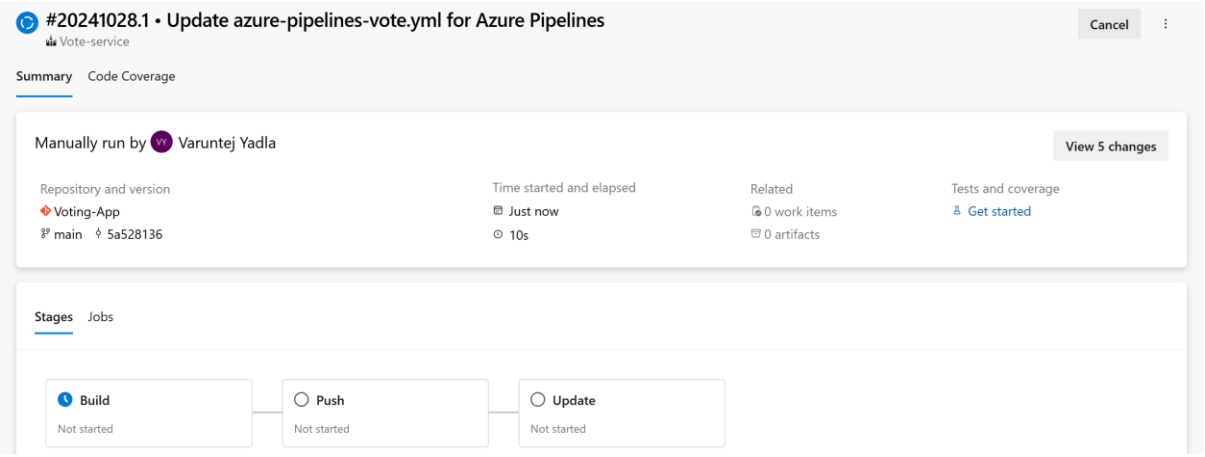
Passing the arguments here.

```
- stage: Update
  displayName: Update
  jobs:
  - job: Update
    displayName: Update
    steps:
    - task: ShellScript@2
      inputs:
        scriptPath: 'scripts/updateK8sManifests.sh'
        args: 'vote $(imageRepository) $(tag)'
```

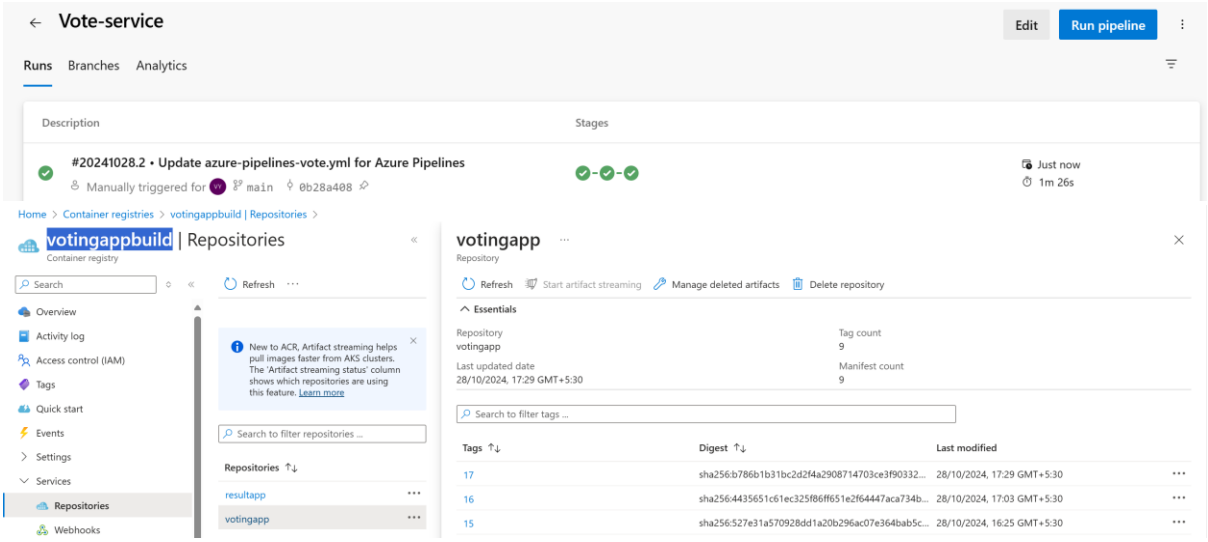


Click on validate and save. When you run the pipeline it will update the line:

```
spec:
  containers:
  - image: dockersamples/examplevotingapp_vote
    name: vote
    ports:
    - containerPort: 80
      name: vote
```



New stage is added and it should update the image name properly after the build push and update is success in the pipeline.



The latest image is 17. It should be updated in the vote-deployment.yaml

Updated. So, everytime this vote pipeline is triggered, it will automatically update the image version.

Usually argocd takes 3 mins time to sync. To get the changes done in manifest file. We can reduce it as well to 10 seconds.

```
yvaru@Tej MINGW64 ~
$ kubectl edit cm argocd-cm -n argocd
configmap/argocd-cm edited
```

Add last two lines (data:.....) at the end of the file:

```
  resourceVersion: "46410"
  uid: c4a54732-dbc7-458e-8253-f78a03df8f38
data:
  timeout.reconciliation: 10s
```

Vote Pod went into ImagePullBackOff issue:

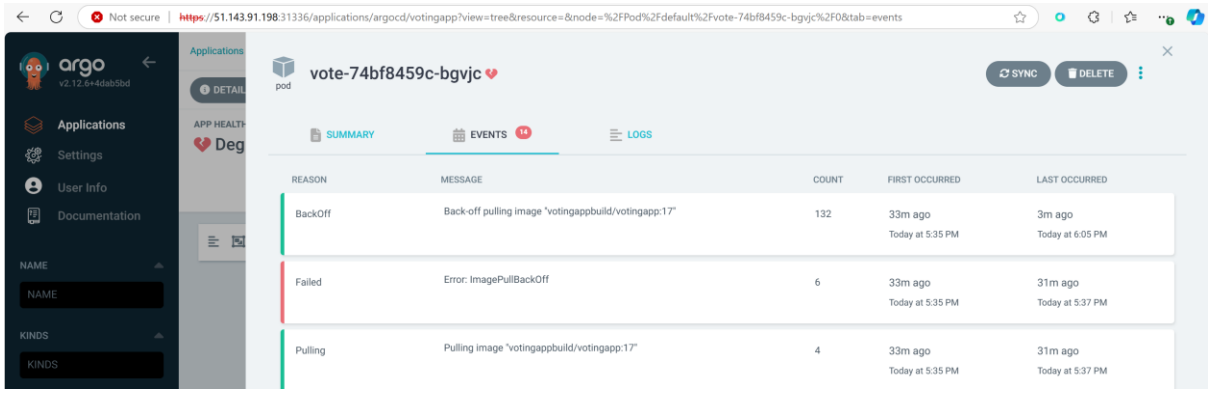
```
yvaru@Tej MINGW64 ~
$ kubectl get pods -n default
NAME                      READY   STATUS             RESTARTS   AGE
db-6d9f87bb9b-8m2gv       1/1     Running            0          3h13m
redis-77fccb7f9-r25xt     1/1     Running            0          3h13m
result-54b5ccfc95-7xbwz   1/1     Running            0          3h13m
vote-5655bd759-vx6ps      1/1     Running            0          3h13m
vote-74bf8459c-bgvjc      0/1     ImagePullBackOff   0          20m
worker-7dd74bcbbb-wfv8n   1/1     Running            0          3h13m
```

```
Events:
  Type     Reason     Age                  From               Message
  ----     ------     ----                 ----               -------
  Normal   Scheduled  21m                  default-scheduler  Successfully assigned default/vote-74bf8459c-bgvjc to aks-agentpool-35213278-vmss000000
  Normal   Pulling    19m (x4 over 21m)    kubelet            Pulling image "votingappbuild/votingapp:17"
  Warning  Failed     19m (x4 over 21m)    kubelet            Failed to pull image "votingappbuild/votingapp:17": failed to pull and unpack image "docker.io/votingappbuild/votingapp:17": failed to resolve reference "docker.io/votingappbuild/votingapp:17": pull access denied, repository does not exist or may require authorization: server message: insufficient_scope: authorization failed
  Warning  Failed     19m (x4 over 21m)    kubelet            Error: ErrImagePull
  Warning  Failed     19m (x6 over 21m)    kubelet            Error: ImagePullBackOff
  Normal   BackOff    77s (x87 over 21m)   kubelet            Back-off pulling image "votingappbuild/votingapp:17"
```

```
yvaru@Tej MINGW64 ~
$ kubectl edit pod vote-74bf8459c-bgvjc
Edit cancelled, no changes made.
```

Image updated in Cluster level also:

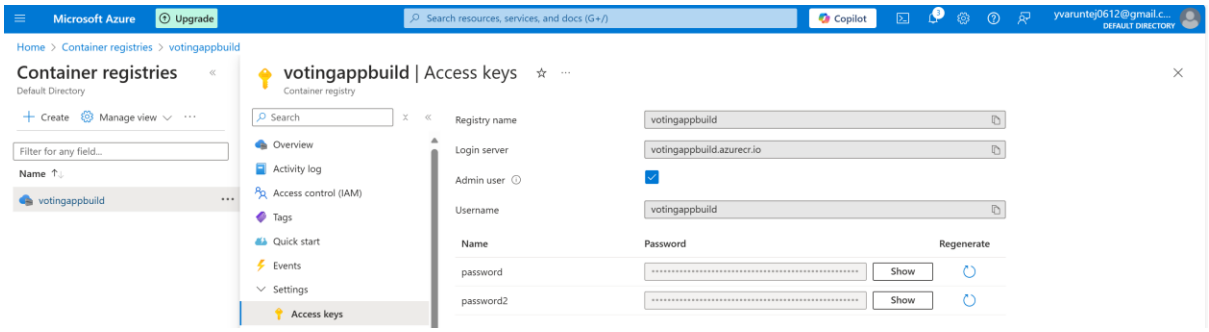
```
  resourceVersion: "101748"
  uid: a2663413-bed5-48ba-8a01-1d8bfb703f20
spec:
  containers:
  - image: votingappbuild/votingapp:17
    imagePullPolicy: Always
    name: vote
```



We are trying to pull the image from a private repository from azure container registry: We need to use a concept called ImagePullSecrets inorder to pull image from private repositories or private registeries. Go to Azure portal and container registries:

Click on Access keys:

Enable Admin User:



Then give this command in the cluster:

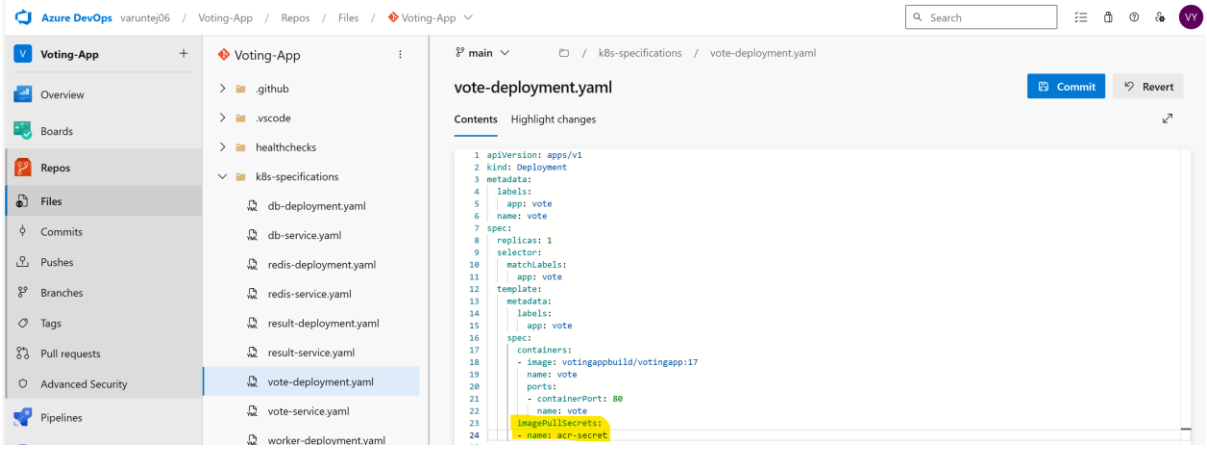
Command to create ACR ImagePullSecret

```
kubectl create secret docker-registry <secret-name> \
    --namespace <namespace> \
    --docker-server=<container-registry-name>.azurecr.io \
    --docker-username=<service-principal-ID> \
    --docker-password=<service-principal-password>
```

secret name: any; namespace: default; principal-ID: Username from access keys; principal password: password from access keys.

```
vvaru@Tej MINGW64 ~
$ kubectl create secret docker-registry acr-secret --namespace default --docker-server=votingappbuild.azurecr.io --docker-username=votingappbuild --docker-password=Rhxxh1Cyy+SLPzBXrb1M8j43FkK4ai2guUTXxF5OHM+ACRCFjOIK
secret/acr-secret created

vvaru@Tej MINGW64 ~
$
```

Now, we will go to deployment.yaml and add these lines:



Now, we will change the some liens of code in Vote directory so that vote pipeline triggers automatically. I updated dogs and cats to Docker and Kubernetes in app.py in Vote directory.

Once committed, pipeline should auto trigger.

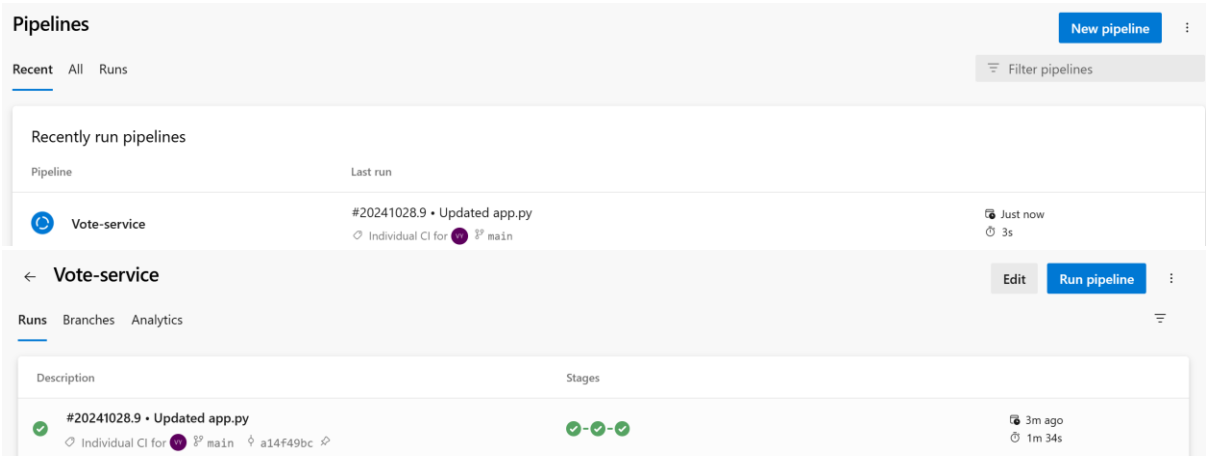
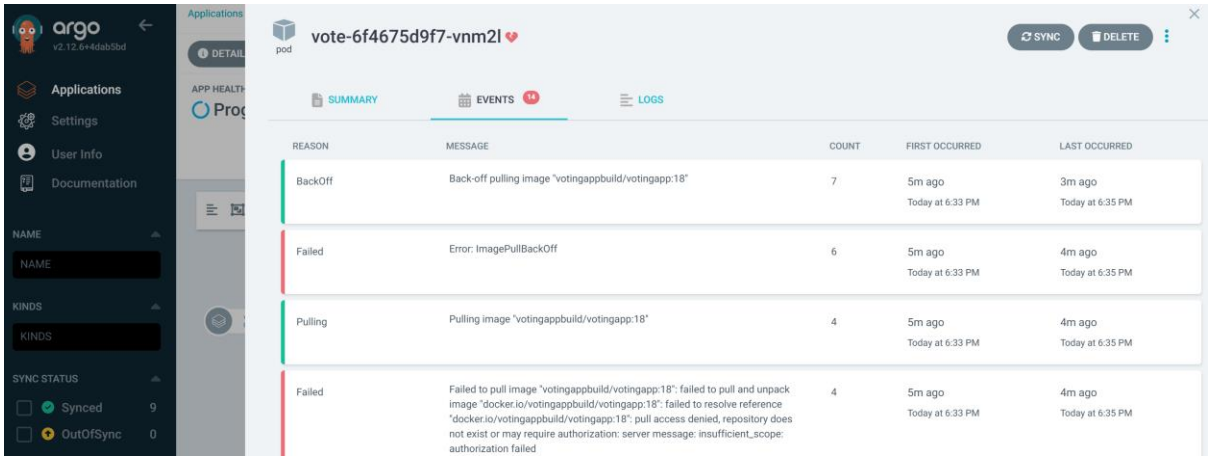


Image updated:

Bt still the pod is failing:

```
vvaru@Tej MINGW64 ~
$ kubectl get pods
NAME                      READY   STATUS         RESTARTS   AGE
db-6d9f87bb9b-8m2gv       1/1     Running        0          3h52m
redis-77fccb7f9-r25xt     1/1     Running        0          3h52m
result-54b5ccfc95-7xbwz   1/1     Running        0          3h52m
vote-5655bd759-vx6ps      1/1     Running        0          3h52m
vote-6f4675d9f7-vnm2l     0/1     ErrImagePull   0          38s
worker-7dd74bcbbb-wfv8n   1/1     Running        0          3h52m
                          node.kubernetes.io/unreachable:NoExecute op=Exists for 300s
Events:
  Type     Reason     Age                From               Message
  ----     ------     ----               ----               -------
  Normal   Scheduled  55s                default-scheduler  Successfully assigned default/vote-6f4675d9f7-vnm2l to aks-agentpool-35213278-vmss000000
  Normal   Pulling    13s (x3 over 55s)  kubelet            Pulling image "votingappbuild/votingapp:18"
  Warning  Failed     13s (x3 over 54s)  kubelet            Failed to pull image "votingappbuild/votingapp:18": failed to pull and unpack image "docker.io/votingappbuild/votingapp:18": failed to resolve reference "docker.io/votingappbuild/votingapp:18": pull access denied, repository does not exist or may require authorization: server message: insufficient_scope: authorization failed
  Warning  Failed     13s (x3 over 54s)  kubelet            Error: ErrImagePull
  Normal   BackOff    1s (x3 over 53s)   kubelet            Back-off pulling image "votingappbuild/votingapp:18"
  Warning  Failed     1s (x3 over 53s)   kubelet            Error: ImagePullBackOff
```



Let’s check both pod and deployment got secret added or not.

```
yvaru@Tej MINGW64 ~
$ kubectl edit pod vote-6f4675d9f7-vnm2l
```

```
spec:
  containers:
  - image: votingappbuild/votingapp:18
    imagePullPolicy: Always
    name: vote
    ports:
    - containerPort: 80
      name: vote
      protocol: TCP
    resources: {}
    terminationMessagePath: /dev/termination-log
    terminationMessagePolicy: File
    volumeMounts:
    - mountPath: /var/run/secrets/kubernetes.io/serviceaccount
      name: kube-api-access-6rqhm
      readOnly: true
  dnsPolicy: ClusterFirst
  enableServiceLinks: true
  imagePullSecrets:
  - name: acr-secret
  nodeName: aks-agentpool-35213278-vmss000000
```

```
yvaru@Tej MINGW64 ~
$ kubectl get deploy
NAME     READY   UP-TO-DATE   AVAILABLE   AGE
db       1/1     1            1           4h1m
redis    1/1     1            1           4h1m
result   1/1     1            1           4h1m
vote     1/1     1            1           4h1m
worker   1/1     1            1           4h1m

yvaru@Tej MINGW64 ~
$ kubectl edit deploy vote
```

```
# Please edit the object below. Lines beginning with a '#' will be ignored,
# and an empty file will abort the edit. If an error occurs while saving this file will be
# reopened with the relevant failures.
#
apiVersion: apps/v1
kind: Deployment
metadata:
  annotations:
    deployment.kubernetes.io/revision: "4"
    kubectl.kubernetes.io/last-applied-configuration: |

{"apiVersion":"apps/v1","kind":"Deployment","metadata":{"annotations":{},"labels":{"app":"vote","app.kubernetes.io/instance":"votingapp"},"name":"vote","namespace":"default"},"spec":{"replicas":1,
"selector":{"matchLabels":{"app":"vote"}},"template":{"metadata":{"labels":{"app":"vote"}},"spec":{"containers":[{"image":"votingappbuild/votingapp:18","name":"vote","ports":[{"containerPort":80,"
name":"vote"}]}],"imagePullSecrets":[{"name":"acr-secret"}]}}}
  creationTimestamp: "2024-10-28T09:12:32Z"
  generation: 4
  labels:
    app: vote
    app.kubernetes.io/instance: votingapp
```

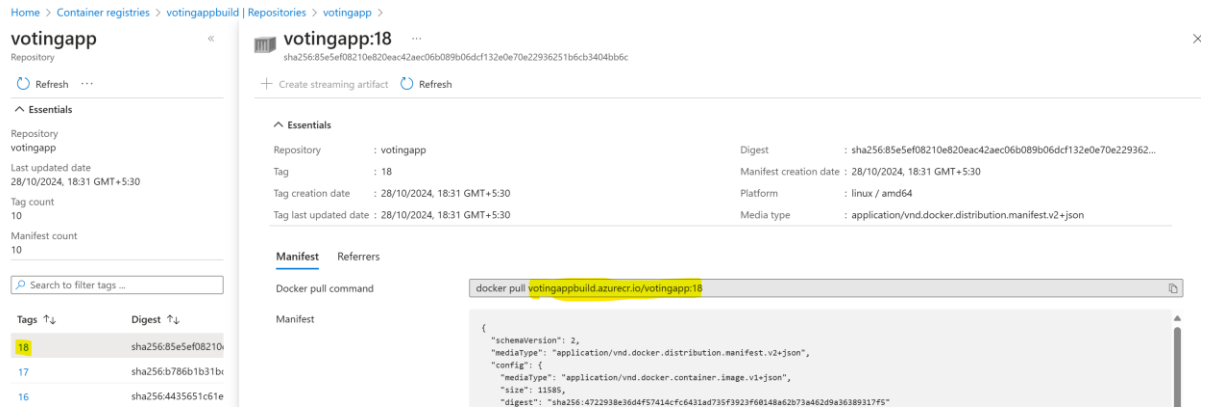
Secret is also created:

```
yvaru@Tej MINGW64 ~
$ kubectl get secret
NAME          TYPE                             DATA   AGE
acr-secret    kubernetes.io/dockerconfigjson   1      24m
```

Still in pod is having issues to pull image:

```
yvaru@Tej MINGW64 ~
$ kubectl get pods -w
NAME                      READY   STATUS             RESTARTS   AGE
db-6d9f87bb9b-8m2gv       1/1     Running            0          4h7m
redis-77fccb7f9-r25xt     1/1     Running            0          4h7m
result-54b5ccfc95-7xbwz   1/1     Running            0          4h7m
vote-5655bd759-vx6ps      1/1     Running            0          4h7m
vote-6f4675d9f7-jkqgz     0/1     ErrImagePull       0          12s
worker-7dd74bcbbb-wfv8n   1/1     Running            0          4h7m
vote-6f4675d9f7-jkqgz     0/1     ImagePullBackOff   0          15s
vote-6f4675d9f7-jkqgz     0/1     ErrImagePull       0          26s
vote-6f4675d9f7-jkqgz     0/1     ImagePullBackOff   0          40s
vote-6f4675d9f7-jkqgz     0/1     ErrImagePull       0          56s
vote-6f4675d9f7-jkqgz     0/1     ImagePullBackOff   0          67s
```

This is because: We mentioned the image pull command wrongly in the scripting file:



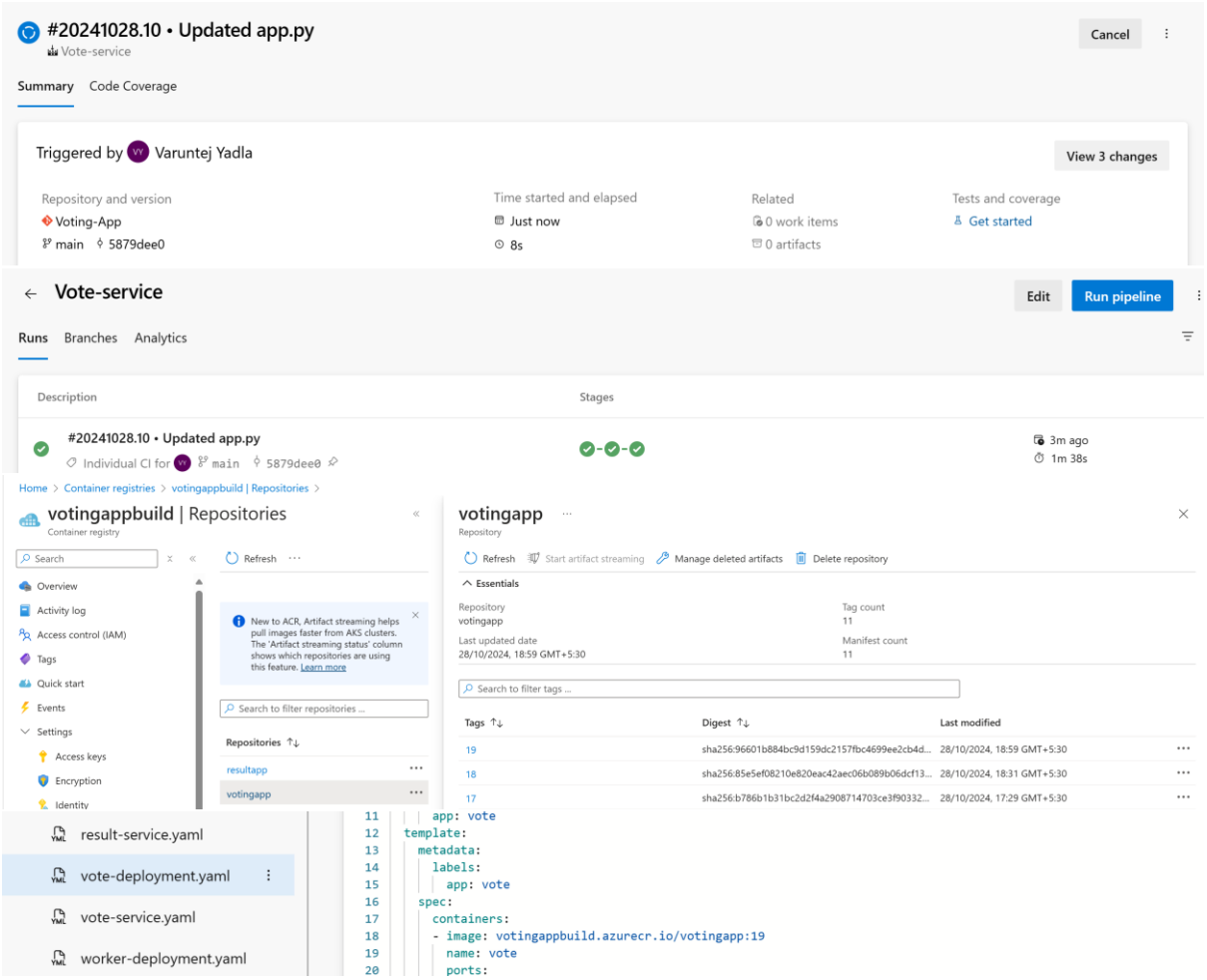
When we write script to update the Image name in the deployment.yaml file we haven't mentioned correctly:

```
13 # Make changes to the Kubernetes manifest file(s)
14 # For example, let's say you want to change the image tag in a deployment.yaml file
15 sed -i "s|image:.*|image: votingappbuild/$2:$3|g" k8s-specifications/$1-deployment.yaml
16
17 # Add the modified files
18 git add .
```

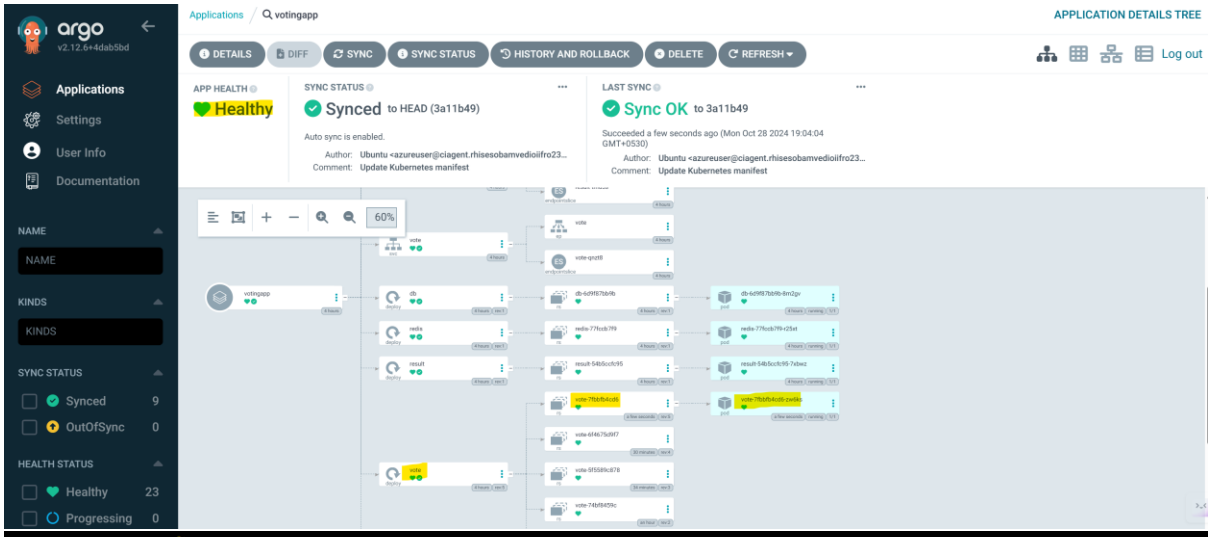
This is wrong. Update it correctly:

```
13 # Make changes to the Kubernetes manifest file(s)
14 # For example, let's say you want to change the image tag in a deployment.yaml file
15 sed -i "s|image:.*|image: votingappbuild.azurecr.io/$2:$3|g" k8s-specifications/$1-deployment.yaml
16
17 # Add the modified files
```

Now do a change in app.py again and check the pipeline run and deployment and pod.



Allo Argo CD to pull the change in manifest file. If still failed delete the pod and wait for sometime it will be up and running.

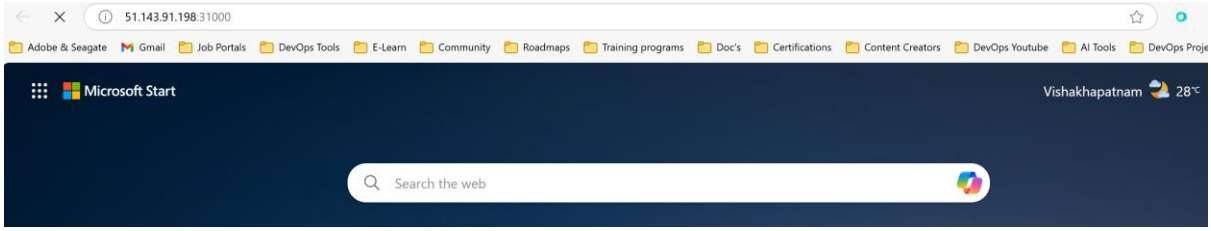



```
yvaru@Tej MINGW64 ~
$ kubectl get pods
NAME                      READY   STATUS    RESTARTS   AGE
db-6d9f87bb9b-8m2gv       1/1     Running   0          4h22m
redis-77fccb7f9-r25xt     1/1     Running   0          4h22m
result-54b5ccfc95-7xbwz   1/1     Running   0          4h22m
vote-7fbbfb4cd6-zw6ks     1/1     Running   0          78s
worker-7dd74bcbbb-wfv8n   1/1     Running   0          4h22m
```

Let’s try to access Vote app:

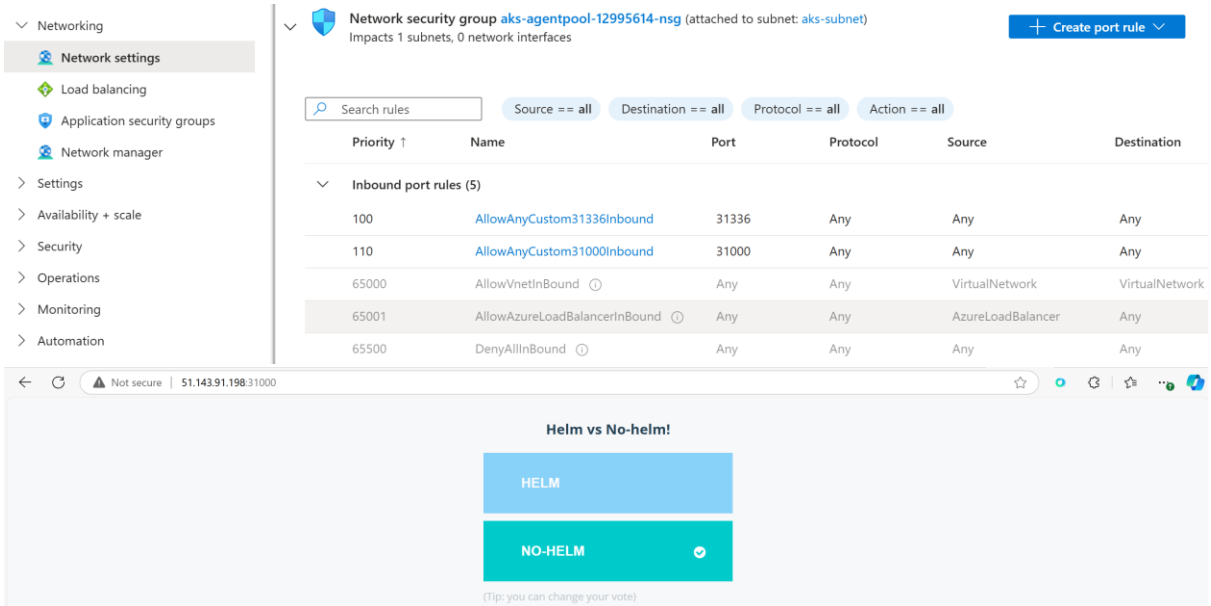
```
yvaru@Tej MINGW64 ~
$ kubectl get svc
NAME         TYPE        CLUSTER-IP     EXTERNAL-IP   PORT(S)          AGE
db           ClusterIP   10.0.209.116   <none>        5432/TCP         4h31m
kubernetes   ClusterIP   10.0.0.1       <none>        443/TCP          8h
redis        ClusterIP   10.0.104.117   <none>        6379/TCP         4h31m
result       NodePort    10.0.71.247    <none>        8081:31001/TCP   4h31m
vote         NodePort    10.0.88.30     <none>        8080:31000/TCP   4h31m
```

```
yvaru@Tej MINGW64 ~
$ kubectl get node -o wide
NAME                                STATUS   ROLES    AGE   VERSION   INTERNAL-IP    EXTERNAL-IP     OS-IMAGE             KERNEL-VERSION      CONTAINER-RUNTIME
aks-agentpool-35213278-vmss000000   Ready    <none>   8h    v1.29.9   10.224.0.4     51.143.91.198   Ubuntu 22.04.5 LTS   5.15.0-1073-azure   containerd://1.7.
22-1
```



Not opening because we need to allow the port number for this node pool in inbound rules:

Go to Azure portal → Virtual machine scale stes → Netwrok Settings:



Now we have to follow the process from Day 15 for result and worker (No Nodeport – Build satge not written – take care of this).

Result service:

Added new stage to the pipeline:

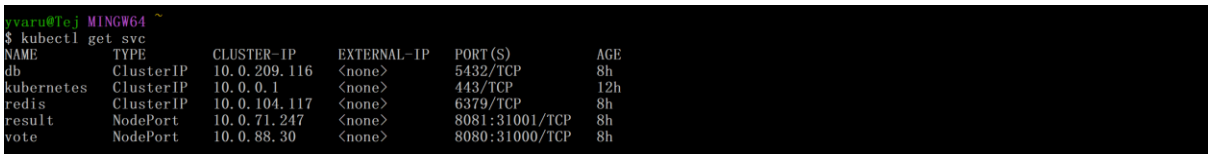
← **Result-service**

main / Voting-App / azure-pipelines-result.yml *

```yaml
- stage: Update
  displayName: Update
  jobs:
  - job: Update
    displayName: Update
    steps:
    Settings
    - task: ShellScript@2
      inputs:
        scriptPath: 'scripts/updateK8sManifests.sh'
        args: 'result $(imageRepository) $(tag)'
```

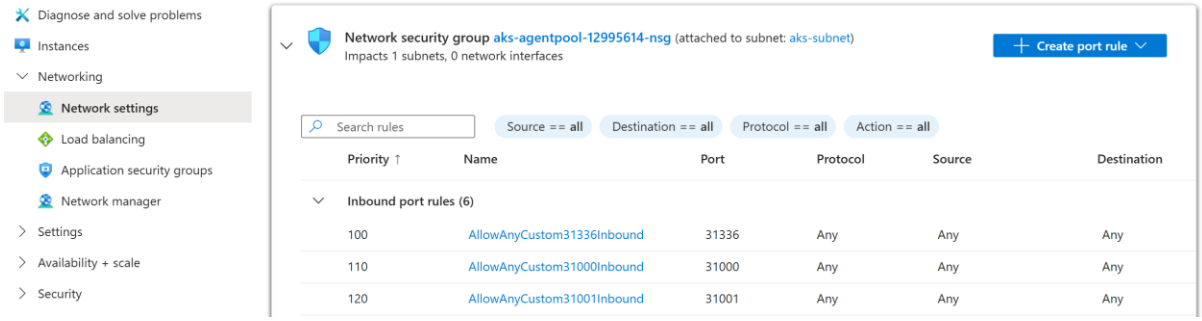
result-deployment.yaml file: adding imagePullSecrets

Result Pod shold be NodePort:

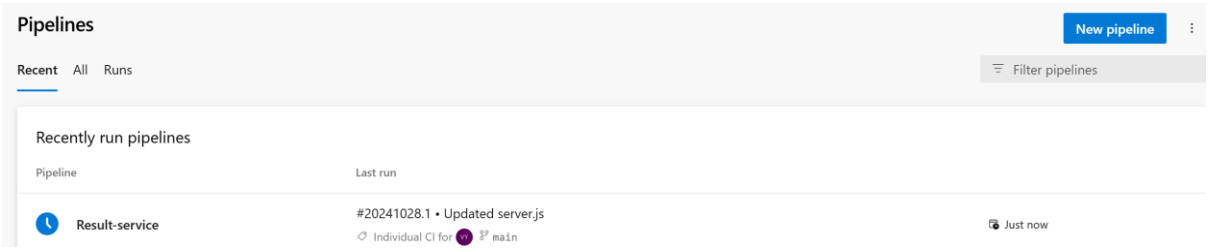


Already NodePort:

Need to allow port 31001 in Inbound rules: Update from Virtual machine scale sets.



Do a small change in result directory (like adding empty line or removing empty line) and it should trigger CI pipeline automatically:



It is in queue for very long check the agent is running or not. If not then restart the agent.

```
yvaru@Tej MINGW64 ~/Downloads
$ ssh -i ciagent_key.pem azureuser@52.172.99.184
Welcome to Ubuntu 24.04.1 LTS (GNU/Linux 6.8.0-1016-azure x86_64)
```

```
azureuser@ciagent:~$ cd
.cache/  .ssh/    myagent/
azureuser@ciagent:~$ cd myagent/
azureuser@ciagent:~/myagent$ ls
_diag  bin        env.sh     license.html  run-docker.sh  svc.sh
_work  config.sh  externals  reauth.sh     run.sh         vsts-agent-linux-x64-3.246.0.tar.gz
azureuser@ciagent:~/myagent$ ./run.sh
Scanning for tool capabilities.
Connecting to the server.
2024-10-28 18:09:41Z: Listening for Jobs
2024-10-28 18:09:44Z: Running job: Build
```

After starting the agent, CI pipeline started first Job.

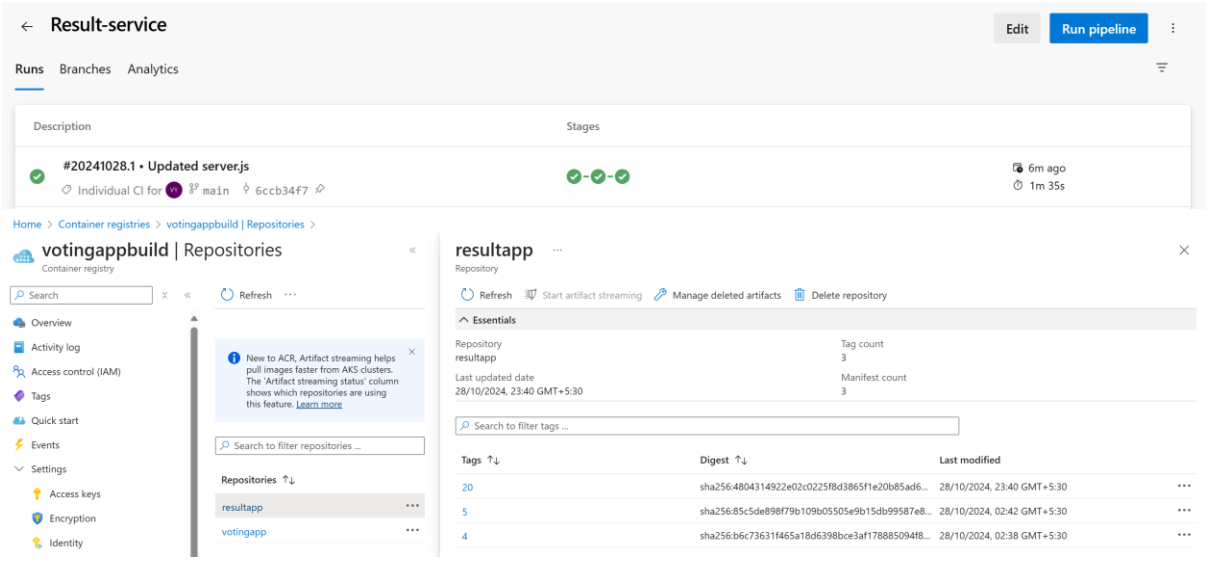
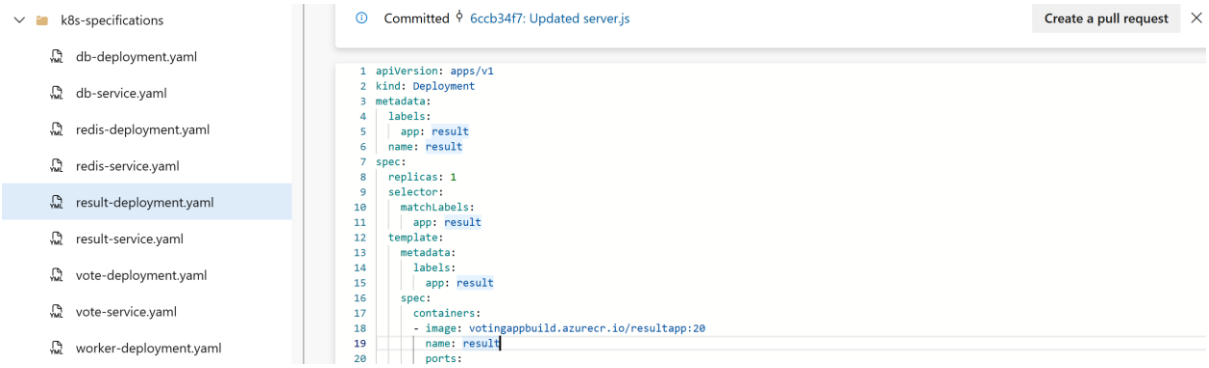
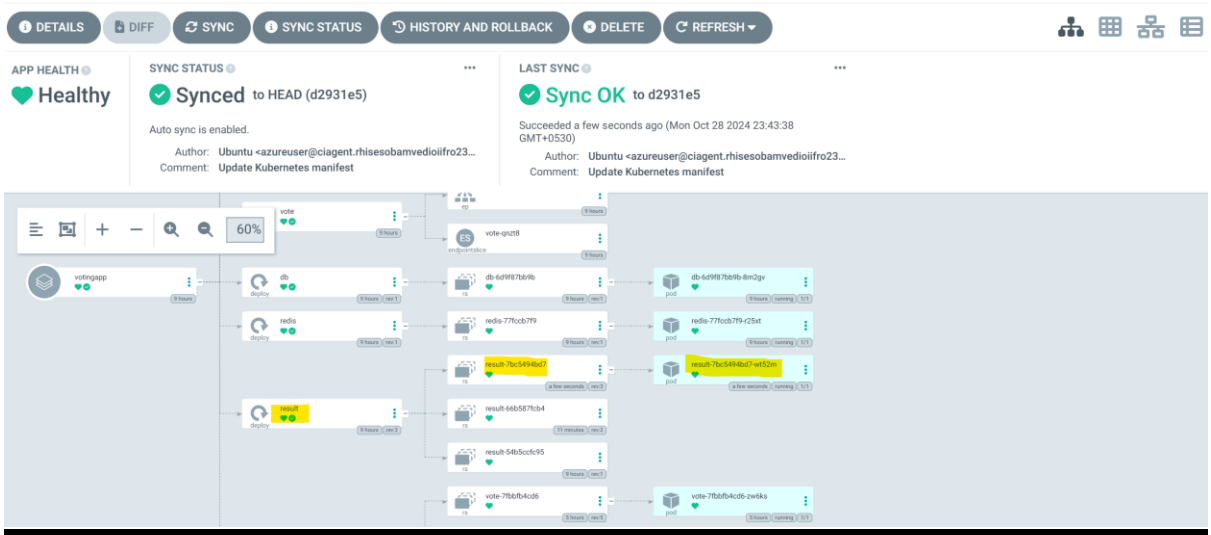


Image also updated in manifest file:



Wait for ArgoCd to pick up the changes in manifest file. Or refresh the voting app application in argocd.



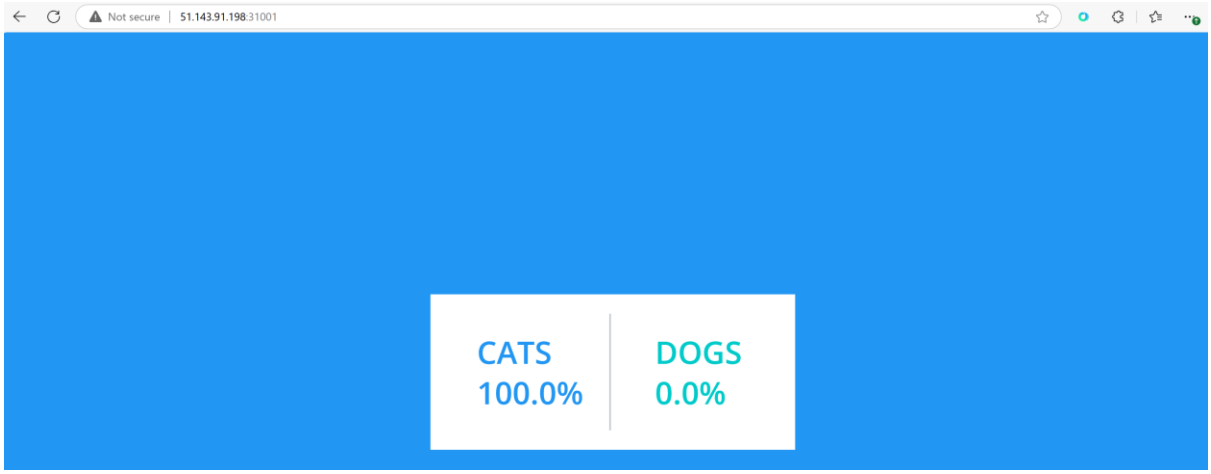

```
yvaru@Tej MINGW64 ~
$ kubectl get pods -w
NAME                      READY   STATUS              RESTARTS   AGE
db-6d9f87bb9b-8m2gv       1/1     Running             0          9h
redis-77fccb7f9-r25xt     1/1     Running             0          9h
result-66b587fcb4-9cftx   1/1     Running             0          10m
vote-7fbbfb4cd6-zw6ks     1/1     Running             0          4h39m
worker-7dd74bcbbb-wfv8n   1/1     Running             0          9h
result-7bc5494bd7-wt52m   0/1     Pending             0          0s
result-7bc5494bd7-wt52m   0/1     Pending             0          0s
result-7bc5494bd7-wt52m   0/1     ContainerCreating   0          0s
result-7bc5494bd7-wt52m   1/1     Running             0          8s
result-66b587fcb4-9cftx   1/1     Terminating         0          11m
result-66b587fcb4-9cftx   0/1     Terminating         0          11m
result-66b587fcb4-9cftx   0/1     Terminating         0          11m
result-66b587fcb4-9cftx   0/1     Terminating         0          11m
```

The new pod created first then old one got terminated. Same can be seen from argo CD UI as well.

Try to access Result service also:

```
yvaru@Tej MINGW64 ~
$ kubectl get svc
NAME         TYPE        CLUSTER-IP     EXTERNAL-IP   PORT(S)          AGE
db           ClusterIP   10.0.209.116   <none>        5432/TCP         9h
kubernetes   ClusterIP   10.0.0.1       <none>        443/TCP          12h
redis        ClusterIP   10.0.104.117   <none>        6379/TCP         9h
result       NodePort    10.0.71.247    <none>        8081:31001/TCP   9h
vote         NodePort    10.0.88.30     <none>        8080:31000/TCP   9h
```

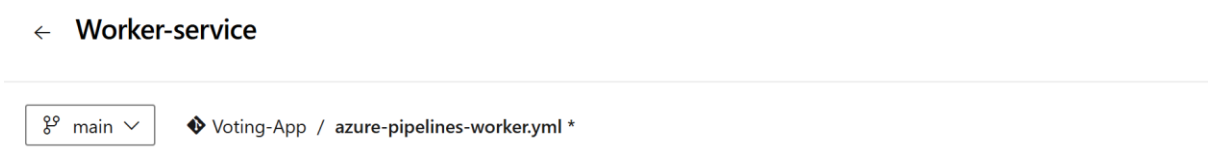
```
yvaru@Tej MINGW64 ~
$ kubectl get node -o wide
NAME                                STATUS   ROLES    AGE   VERSION    INTERNAL-IP   EXTERNAL-IP     OS-IMAGE             KERNEL-VERSION          CONTAINER-RUNTIME
aks-agentpool-35213278-vmss000000   Ready    <none>   12h   v1.29.9    10.224.0.4    51.143.91.198   Ubuntu 22.04.5 LTS   5.15.0-1073-azure       containerd://1.7.
22-1
```



It is working eventough we haven't completed the process for worker service. Because worker image was taken from dockerhub. So it is working. We can create it and check it should be working.

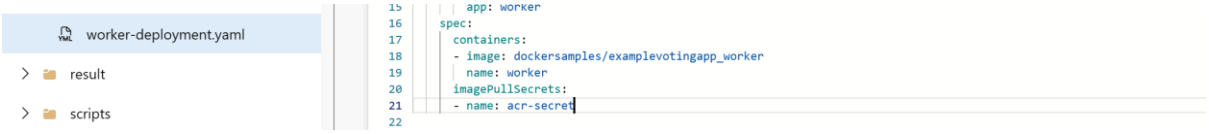
Worker Service:

Added new stage for Push and Update:

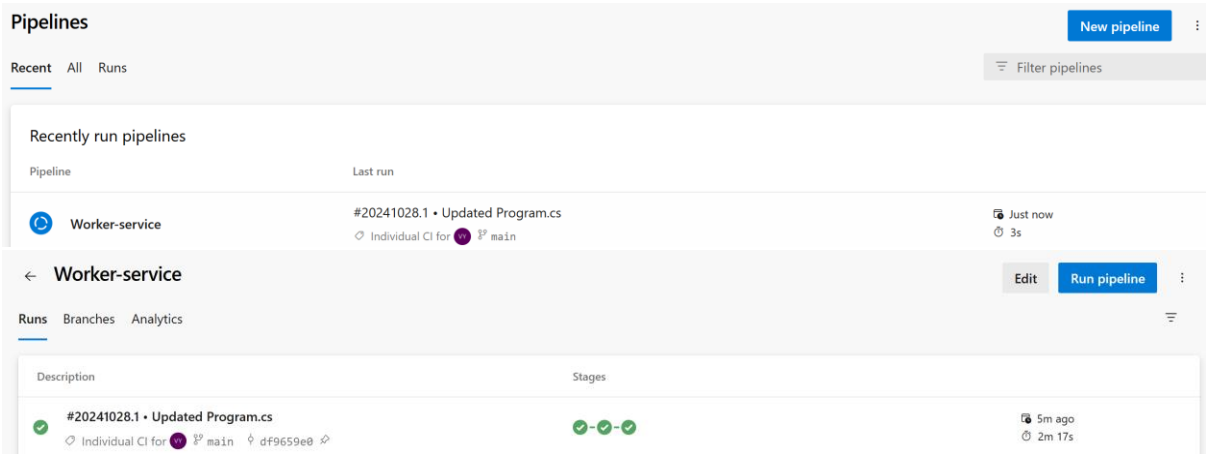


```yaml
variables:
  # Container registry service connection established during pipeline creation
  dockerRegistryServiceConnection: 'ffaea5df-46d6-4d00-8c4b-c537f41f40db'
  imageRepository: 'workerapp'
  containerRegistry: 'votingappbuild.azurecr.io'
  dockerfilePath: '$(Build.SourcesDirectory)/worker/Dockerfile'
  tag: '$(Build.BuildId)'
- stage: Push
  displayName: Push
  jobs:
  - job: Push
    displayName: Push
    steps:
      Settings
    - task: Docker@2
      displayName: Push the Image
      inputs:
        containerRegistry: '$(dockerRegistryServiceConnection)'
        repository: '$(imageRepository)'
        command: 'push'
        tags: '$(tag)'
- stage: Update
  displayName: Update
  jobs:
  - job: Update
    displayName: Update
    steps:
      Settings
    - task: ShellScript@2
      inputs:
        scriptPath: 'scripts/updateK8sManifests.sh'
        args: 'worker $(imageRepository) $(tag)'
```

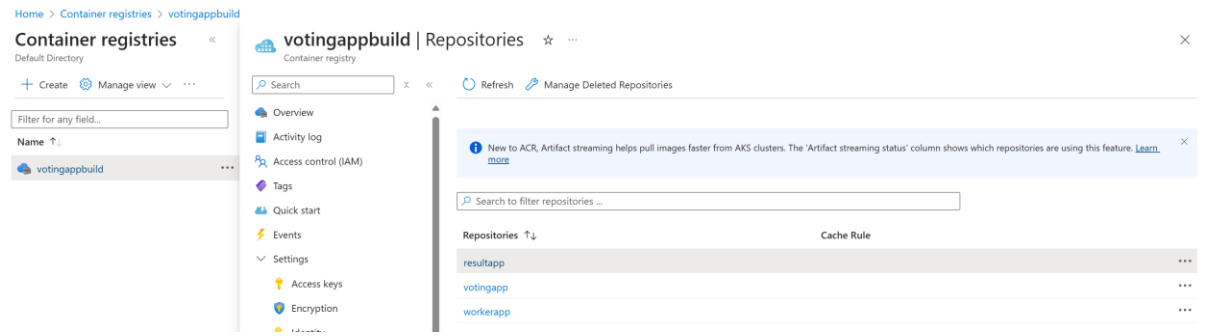
We haven't created the repository for the worker image that will be build during CI. New repo in Container registry with exact same name will be created when Pipeline strats and build and push stage is successful.

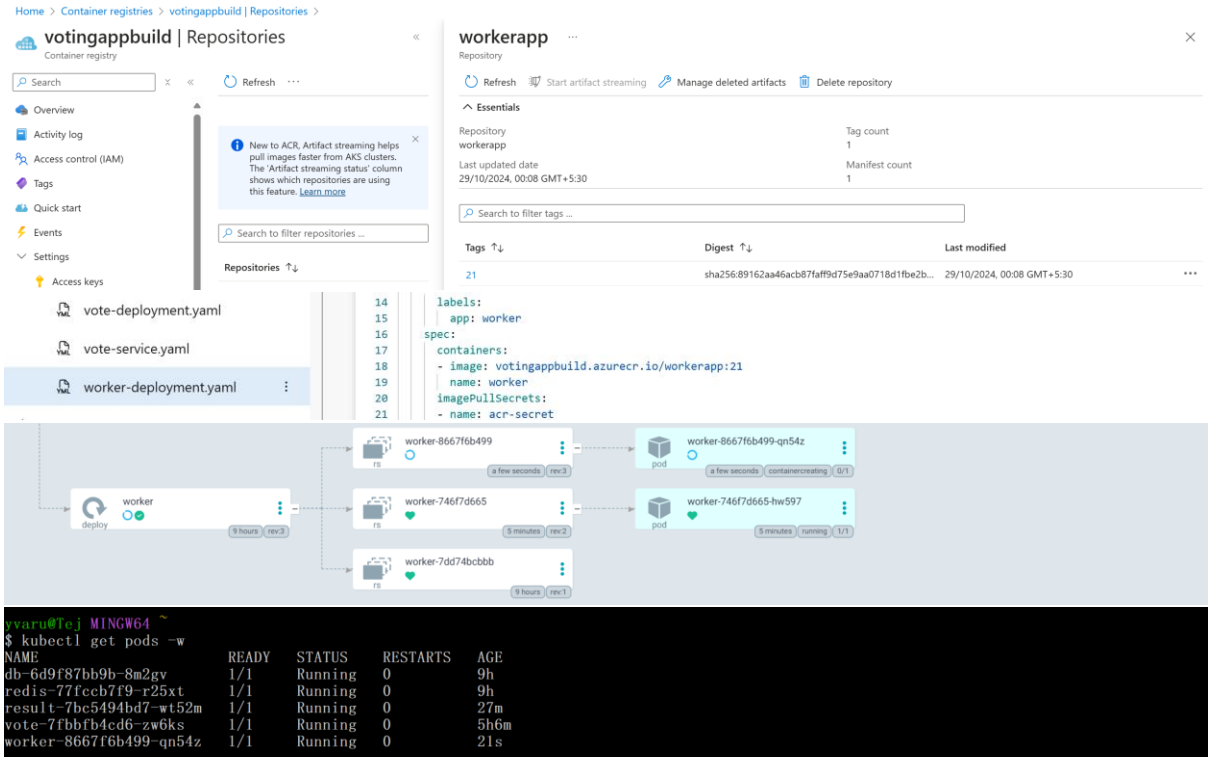


Update something in worker directory:



worker app repository is created in container registery:






```
yvaru@Tej MINGW64 ~
$ kubectl get pods -w
NAME                      READY   STATUS    RESTARTS   AGE
db-6d9f87bb9b-8m2gv       1/1     Running   0          9h
redis-77fceb7f9-r25xt     1/1     Running   0          9h
result-7bc5494bd7-wt52m   1/1     Running   0          27m
vote-7fbbfb4cd6-zw6ks     1/1     Running   0          5h6m
worker-8667f6b499-qn54z   1/1     Running   0          21s
```

Try to update the vote in vote app and check in result app. It should work fine.

Project is done. Will delete resources created form azure portal:

It will delete AKS, Container registeries, Agent pool, ArgoCd inside AKS.

Need to do all this process if you want to run this application again. Rest of the application source code, manifest files and scripts will be in azure DevOps.

Files changed in source code are: Manifest files for Vote, result and worker (Deployment.yaml)

Added Scriptting file for updating Image name in deployment manifest files.

Create CI pipelines for Vote, Result and Worker.