

DevOps Project Report: Voting App

1. Project Overview

The *Example Voting App* is a distributed microservices-based web application that demonstrates fundamental DevOps practices. It showcases the orchestration and deployment of containerized services using Docker, Docker Swarm, and Kubernetes, with CI/CD pipelines implemented for automation.

This application is composed of five main components:

- A Python-based frontend voting app
- Redis message queue for vote handling
- A .NET Core worker service to process the votes
- A Postgres database for persistent storage
- A Node.js-based result app for real-time vote display

2. Technologies Used

- **Frontend:** Python (Vote App), Node.js (Results App)
- **Backend Worker:** .NET Core
- **Database:** PostgreSQL
- **Messaging Queue:** Redis
- **Containerization:** Docker, Docker Compose
- **Orchestration:** Docker Swarm & Kubernetes
- **CI/CD Tool:** GitHub Actions + ArgoCD
- **Cloud Platform:** Microsoft Azure (AKS Cluster)

3. Project Setup and Execution

3.1. Local Setup

- Docker Desktop installed with Compose.
- Project launched using:
\$docker compose up
- Applications accessible via:
 - Vote App: <http://localhost:8080>
 - Results App: <http://localhost:8081>

3.2. Swarm Mode

- Docker Swarm initialized using:

```
bash
```

 Copy  Edit

```
docker swarm init  
docker stack deploy --compose-file docker-stack.yml vote
```

4.1. YAML Configurations

The k8s-specifications folder contains deployment and service files for all components. The app was deployed using:

```
bash
```

 Copy  Edit

```
kubectl create -f k8s-specifications/
```

4.2. Access

- Vote App exposed on port **31000**
- Results App exposed on port **31001**

5. CI/CD Implementation

5.1. CI (Continuous Integration)

- The project includes a functional CI pipeline via Azure Pipelines .
- On code push, automated tests and container builds are triggered.
- The container Image then Pushed To Azure Container Registry

5.2. CD (Continuous Delivery)

- Azure Kubernetes Service (AKS) cluster created and connected.
- ArgoCD installed and configured for GitOps-based delivery.

Steps followed:

- Connect to cluster using Azure CLI:

bash

Copy Edit

```
az aks get-credentials --name <cluster-name> --resource-group <group-name>
```

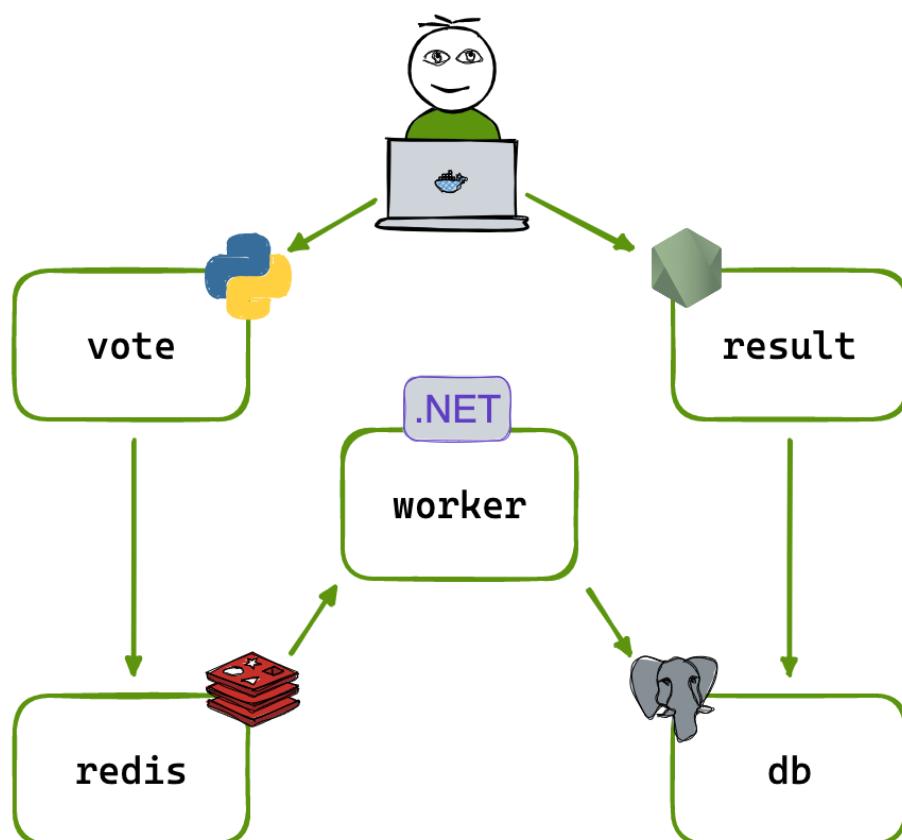
- Install ArgoCD and access via exposed service using:

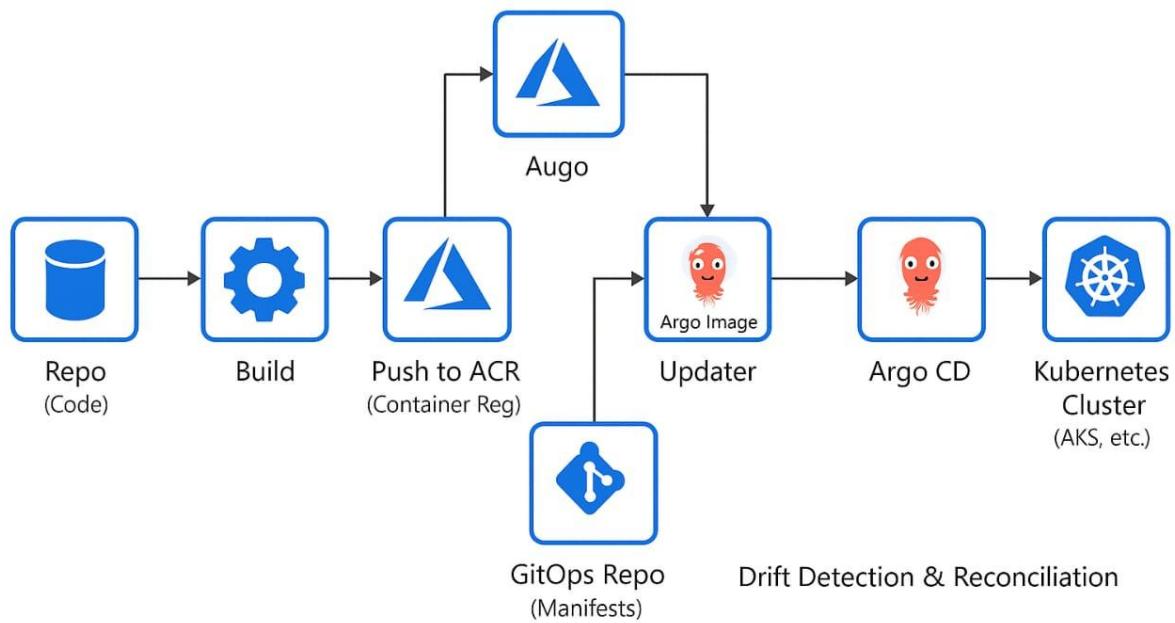
bash

Copy Edit

```
kubectl edit svc argocd-server -n argocd
```

6. Architecture Diagram





7. Screenshots of Project all Process

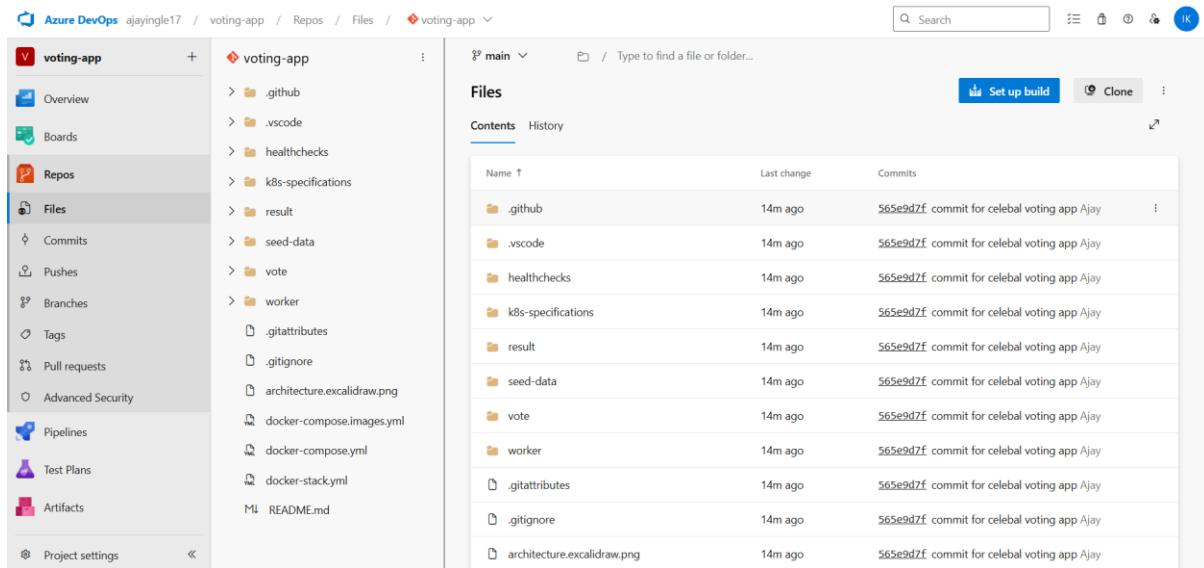
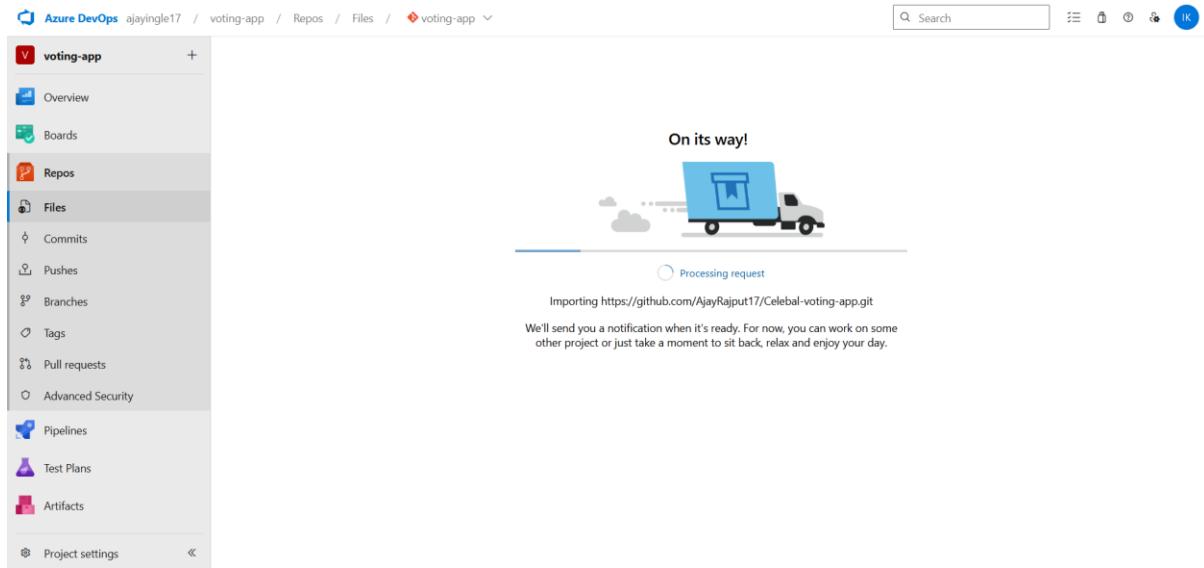
The image consists of two screenshots of the Azure DevOps interface.

Top Screenshot: Create a project to get started

This screenshot shows the process of creating a new project. The project name is "voting-app" and the description is "An Voting app for my celebal Intership Project". The visibility is set to "Private". A cartoon illustration of a person sitting on the ground with a dog is displayed in the background.

Bottom Screenshot: voting-app Overview

This screenshot shows the overview page for the "voting-app" project. The left sidebar includes links for Overview, Summary, Dashboards, Wiki, Boards, Repos, Pipelines, Test Plans, and Artifacts. The main area features a "Welcome to the project!" message and a "Project stats" section which states "No stats are available at this moment". The Members section shows one member. The top of the screen shows a Windows taskbar with various pinned icons and system status.



Microsoft Azure

Search resources, services, and docs (G+)

Copilot

202302040021@mstea... MIT ACADEMY OF ENGINEERING...

Create a resource group

Basics Tags Review + create

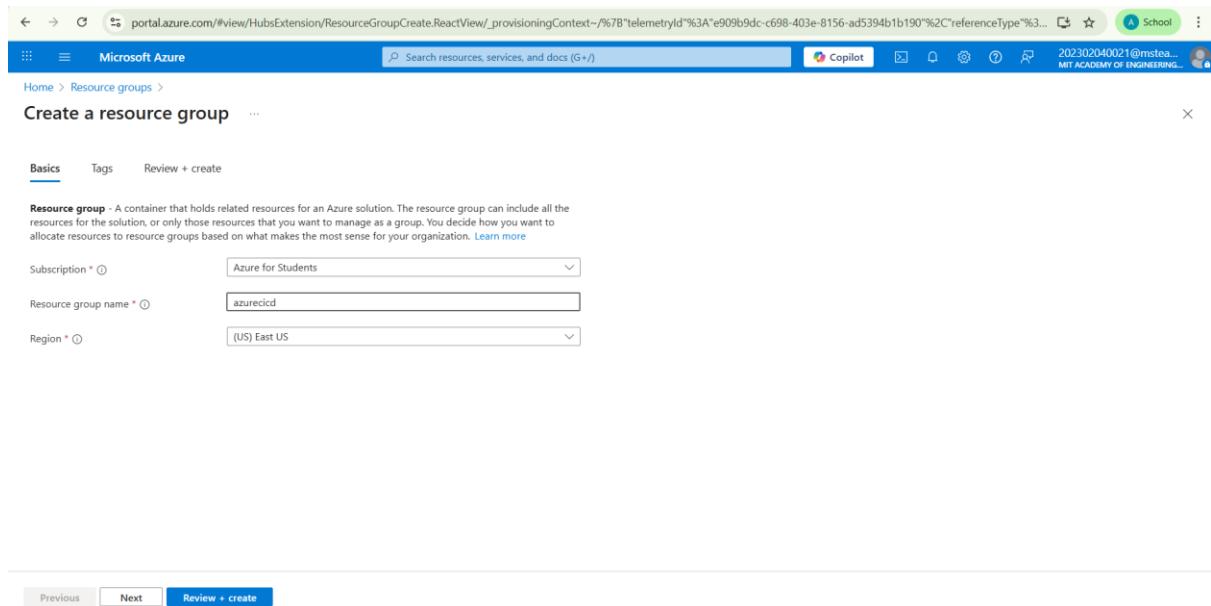
Resource group - A container that holds related resources for an Azure solution. The resource group can include all the resources for the solution, or only those resources that you want to manage as a group. You decide how you want to allocate resources to resource groups based on what makes the most sense for your organization. [Learn more](#)

Subscription *

Resource group name *

Region *

Previous Next Review + create



Microsoft Azure

Search resources, services, and docs (G+)

Copilot

202302040021@mstea... MIT ACADEMY OF ENGINEERING...

Create a resource group

Basics Tags Review + create

[Automation Link](#)

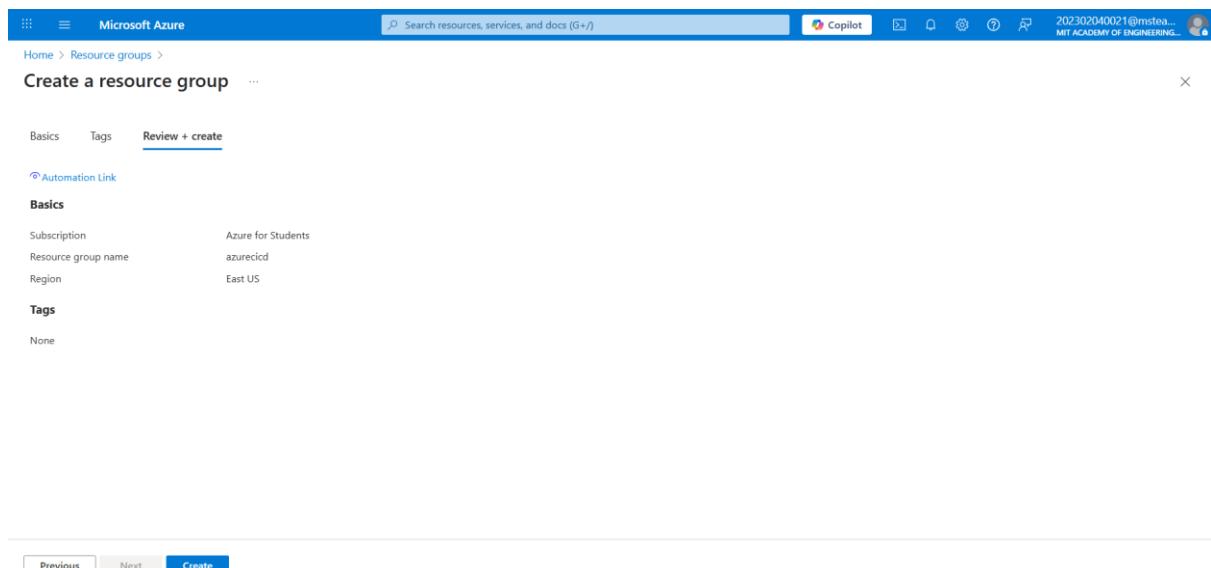
Basics

Subscription	Azure for Students
Resource group name	azurecid
Region	East US

Tags

None

Previous Next Create



Inbox (3,072) - 202302040021 | Azure DevOps | Microsoft Azure | azurecid - Microsoft Azure | Pipelines - Recent

portal.azure.com/#@msteams.mitaoe.ac.in/resource/subscriptions/0587b5e2-0375-4611-af56-ca8af4d38810c/resourceGroups/azurecid/overview

Microsoft Azure

Home > Resource groups >

Resource groups

MIT Academy of Engineering (MITAOE), Alandi, Pu...

+ Create Manage view ...

You are viewing a new version of Browse experience. Click here to access the old experience.

Name
azurecid
my_storage
NetworkWatcherRG

Search

Overview

Activity log

Access control (IAM)

Tags

Resource visualizer

Events

Settings

Cost Management

Monitoring

Automation

Help

Create

Manage view

Delete resource group

Refresh

Export to CSV

Open query

Group by none

JSON View

No resources match your filters

Try changing or clearing your filters.

+ Create Clear filters

Learn more

Add or remove favorites by pressing Ctrl+Shift+F

Showing 1 - 3 of 3. Display count: auto

Showing 1 - 0 of 0. Display count: auto

Give feedback

portal.azure.com/#browse/Microsoft.ContainerRegistry%2Fregistries

Microsoft Azure

Home >

Container registries

MIT Academy of Engineering (MITAOE), Alandi, Pune (msteams.mitaoe.ac.in)

+ Create Manage view ... Refresh Export to CSV Open query Assign tags

You are viewing a new version of Browse experience. Click here to access the old experience.

Subscription	Resource Group	Location
Subscription equals all	Resource Group equals all	Location equals all

Filter for any field...

No container registries to display

Build, store, secure, scan, replicate, and manage container images and artifacts with a fully managed, geo-replicated instance of OCI distribution. Connect across environments, including Azure Kubernetes Service and Azure Red Hat OpenShift, and across Azure services like App Service, Machine Learning, and Batch.

+ Create

Learn more

Showing 1 - 0 of 0. Display count: auto

Give feedback

Create container registry

Basics Networking Encryption Tags Review + create

Azure Container Registry allows you to build, store, and manage container images and artifacts in a private registry for all types of container deployments. Use Azure container registries with your existing container development and deployment pipelines. Use Azure Container Registry Tasks to build container images in Azure on-demand, or automate builds triggered by source code updates, updates to a container's base image, or timers. [Learn more](#)

Project details

Subscription * Azure for Students

Resource group * azurecid [Create new](#)

Instance details

Registry name * ajayazurecid.azurecr.io

Location * East US

Domain name label scope * Unsecure

Registry domain name ajayazurecid.azurecr.io

Use availability zones

[Review + create](#) [< Previous](#) [Next: Networking >](#)

Create container registry

Running final validation

Basics Networking Encryption Tags Review + create

Registry details

Registry name	ajayazurecid
Subscription	Azure for Students
Resource Group	azurecid
Location	East US
Domain name label scope	Unsecure
Availability zones	Disabled
Pricing plan	Basic
Role assignment permissions mode (Preview)	RBAC Registry Permissions

Networking

Public network access Yes

Encryption

[Create](#) [< Previous](#) [Next >](#) Download a template for automation

Microsoft Azure | Overview

Your deployment is complete

Deployment name : Microsoft.ContainerRegistry
Subscription : Azure for Students
Resource group : azurecid

Start time : 7/20/2025, 1:30:43 PM
Correlation ID : 706aa79f-25ad-4c2c-b540-ec16895900de

Cost management
Get notified to stay within your budget and prevent unexpected charges on your bill.
[Set up cost alerts >](#)

Microsoft Defender for Cloud
Secure your apps and infrastructure
[Go to Microsoft Defender for Cloud >](#)

Free Microsoft tutorials
[Start learning today >](#)

Work with an expert
Azure experts are service provider partners who can help manage your assets on Azure and be your first line of support.
[Find an Azure expert >](#)

Give feedback
[Tell us about your experience with deployment](#)

Add or remove favorites by pressing **Ctrl+Shift+F**

Azure DevOps | voting-app/_build

Pipelines

Create your first Pipeline

Automate your build and release processes using our wizard, and go from code to cloud-hosted within minutes.

Create Pipeline

Project settings

28°C Mostly cloudy

The screenshot shows the 'Connect' step of creating a new pipeline. The left sidebar lists project navigation options like Overview, Boards, Repos, Pipelines, Environments, Library, Test Plans, and Artifacts. The Pipelines option is selected. The main area displays a 'Where is your code?' section with three options: 'Azure Repos Git (YAML)', 'GitHub (YAML)', and 'Bitbucket Cloud (YAML)'. A 'More options' dropdown is visible.

The screenshot shows the 'Configure' step of creating a new pipeline. The left sidebar is identical to the previous screenshot. The main area displays a 'Configure your pipeline' section with various templates listed. A 'Docker' template is selected. A modal dialog titled 'Docker' is overlaid on the right, prompting the user to 'Build and push an image to Azure Container Registry'. It asks to 'Select an Azure subscription' and shows a single option: 'Azure for Students' (0587b5e2-0375-4611-af56-ca8a4d38810c). A 'Continue' button is at the bottom right of the dialog.

Screenshot of the Azure DevOps Pipelines configuration screen for a new pipeline.

The pipeline configuration is set to "Docker" mode, with the following settings:

- Container registry:** ajayazurecid
- Image Name:** votingapp
- Dockerfile:** \${Build.SourcesDirectory}/result/Dockerfile

The pipeline steps listed on the left include:

- Docker (Build a Docker image)
- Docker (Build and push an image to Azure Container Registry)
- Deploy to Azure Kubernetes Service (Build and push image to Azure Container Registry; Deploy to Azure Kubernetes Service)
- .NET Desktop (Build and run tests for .NET desktop or Windows classic desktop solutions)
- Node.js (Build a general Node.js project with npm)
- Node.js Express Web App to Linux on Azure (Build a Node.js Express app and deploy it to Azure as a Linux web app)
- Node.js with Vue (Build a Node.js project that uses Vue)
- Node.js with webpack (Build a Node.js project using the webpack CLI)
- Node.js with React (Build a Node.js project that uses React)

Buttons at the bottom right: Back, Validate and configure.

Screenshot of the Azure DevOps Pipelines review screen for a new pipeline.

The pipeline configuration is set to "Review" mode, showing the generated YAML file:

```
voting-app / 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: 'a422b892-bd22-4a7d-b297-5fe0cc166251'
14 - imageRepository: 'votingapp'
15 - containerRegistry: 'ajayazurecid.azurecr.io'
16 - dockerfilePath: '${Build.SourcesDirectory}/result/Dockerfile'
17 - tag: '${Build.BuildId}'
18
19 - # Agent VM image name
20 - vmImageName: 'ubuntu-latest'
21
22 stages:
23 - stage: Build
24   displayName: Build and push stage
```

Buttons at the top right: Variables, Save and run, Show assistant.

This screenshot shows the Microsoft Azure Compute infrastructure Virtual machines page. The left sidebar is collapsed, and the main area displays a message: "No virtual machines to display". It includes instructions to "Create a virtual machine that runs Linux or Windows. Select an image from the marketplace or use your own customized image." Below this are links to "Learn more about Windows virtual machines" and "Learn more about Linux virtual machines". At the bottom, there's a "Create" button and a "Give feedback" link.

This screenshot shows the Microsoft Azure Create a virtual machine page. The "Basics" tab is selected. The "Project details" section shows a subscription of "Azure for Students" and a resource group of "azurecid". The "Instance details" section shows a virtual machine name of "VM1" and a region of "(US) East US". At the bottom, there are navigation buttons: "< Previous", "Next : Disks >", and "Review + create".

The screenshot shows the Microsoft Azure portal with a deployment overview page. The deployment name is "CreateVm-canonical.ubuntu-24_04-lts-server-20250720184537". The status is "Your deployment is complete". Deployment details include a start time of 7/20/2025, 6:51:35 PM, and a correlation ID of 3d2972eb-adb8-4e5d-84f7-950328d9d70. There are sections for "Next steps" and "Give feedback". A sidebar on the right provides links to Cost Management, Microsoft Defender for Cloud, Free Microsoft tutorials, and Work with an expert.

- Configure the agent on Virtual Machine

The screenshot shows the Azure DevOps interface for a project named "voting-app". Under "Project Settings", the "Agent pools" section is selected. A modal window titled "Get the agent" is open, specifically for the "Linux" tab. It shows "System prerequisites" and "Download the agent" (with a "Download" button). Below that is "Create the agent" with a terminal command:

```
~$ mkdir myagent && cd myagent  
~$ myagent$ tar zxf ~/downloads/vsts-agent-linux-x64-4.258.1.tar.gz
```

. Further down is "Configure the agent" with a terminal command:

```
~/myagent$ ./config.sh
```

. At the bottom, there's an optional step: " Optionally run the agent interactively" with a terminal command:

```
~/myagent$ ./run.sh
```

.

```
azureuser@azureagent:~/myagent
./bin/Microsoft.VisualStudio.Services.ArtifactServices.App.Shared.dll
./bin/System.Private.Xml.dll
./bin/Agent.Plugins.dll
./bin/System.Xml.Serialization.dll
./bin/Agent.Plugins.Log.TestResultParser.Contracts.dll
./bin/System.Xml.XPath.XDocument.dll
./bin/System.Security.Cryptography.ProtectedData.dll
./bin/System.Numerics.Vectors.dll
./bin/System.Security.Cryptography.Pkcs.dll
./bin/de/
./bin/de/System.Private.ServiceModel.resources.dll
./bin/Agent.Sdk.deps.json
./run.sh
./reauth.sh
./env.sh
./license.html
azureuser@azureagent:~/myagent$ config.sh
config.sh: command not found
azureuser@azureagent:~/myagent$ ./config.sh


agent v4.258.1 (commit 8292055)

>> 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 sources 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) >
```

26°C Mostly cloudy Search ENG IN 19:46 20-07-2025

```
azureuser@azureagent:~/myagent
config.sh: command not found
azureuser@azureagent:~/myagent$ ./config.sh


agent v4.258.1 (commit 8292055)

>> 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 sources 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 > https://dev.azure.com/ajayingle17
Enter authentication type (press enter for PAT) >
Enter personal access token > *****
Connecting to server ...
>> Register Agent:
Enter agent pool (press enter for default) > azureagent
Enter agent name (press enter for azureagent) > azureagent
Scanning for tool capabilities.
Connecting to the server.
Successfully added the agent
Testing agent connection.
Enter work folder (press enter for _work) > |
```

24°C Mostly cloudy Search ENG IN 20:30 20-07-2025

The screenshot shows the Azure DevOps portal with the URL dev.azure.com/ajayingle17/_settings/agentpools?poolId=10&view=agents. The page displays the 'Agents' tab for the 'azureagent' pool. A single agent, 'azureagent', is listed with the status 'Idle'. The portal interface includes a left sidebar with 'Organization Settings' and various navigation tabs like 'Jobs', 'Agents', 'Details', 'Security', 'Settings', 'Maintenance History', and 'Analytics'.

```
azureuser@azureagent:~/myagent$ ./config.sh
[AZUREPIPELINES]
agent v4.258.1
          (commit 8292055)

Error reported in diagnostic logs. Please examine the log for more details.
- /home/azureuser/myagent/_diag/Agent_20250720-150225-utc.log
Cannot configure the agent because it is already configured. To reconfigure the agent, run 'config.cmd remove' or './config.sh remove' first.
azureuser@azureagent:~/myagent$ ./run.sh
Scanning for tool capabilities.
Connecting to the server.
2025-07-20 15:03:10Z: Listening for jobs
```

The terminal window shows the configuration of an Azure Pipelines agent. It starts with the command `./config.sh`, which prints the agent version (v4.258.1) and commit hash (8292055). An error message follows, stating that the agent is already configured and cannot be reconfigured without removing it first. After this, the command `./run.sh` is run, which performs a tool scan and connects to the server, ending with the message "Listening for jobs". The terminal window has a dark background and uses white text for readability.

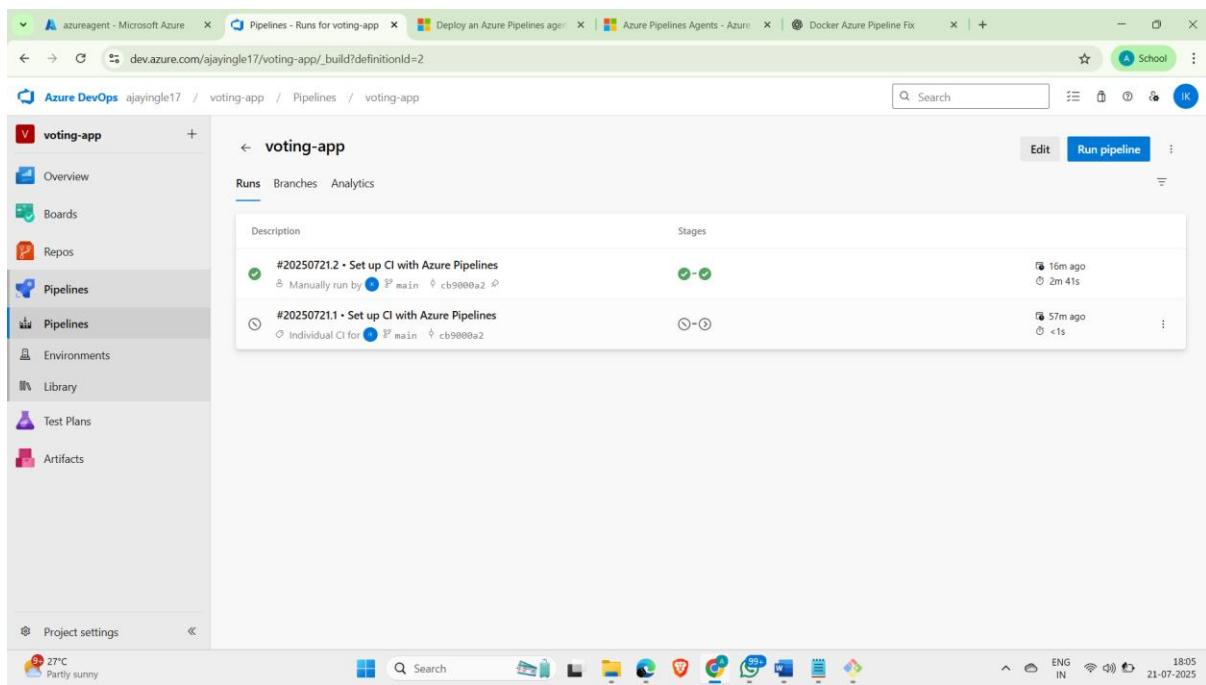
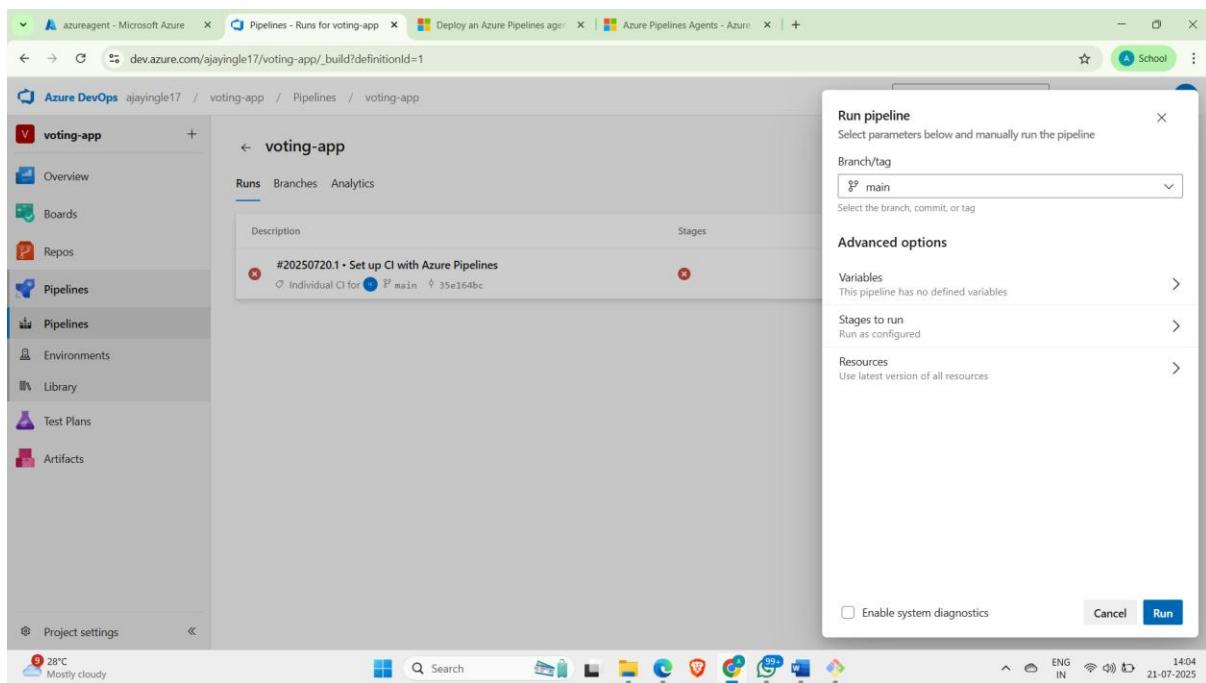
The screenshot shows the Azure DevOps interface for managing agent pools. On the left, there's a sidebar with 'Organization Settings' for 'ajayingle17'. The main area is titled 'azureagent' under 'Agent pools'. It has tabs for 'Jobs', 'Agents', 'Details', 'Security', 'Settings', 'Maintenance History', and 'Analytics'. The 'Agents' tab is selected, displaying a table with one row for 'azureagent'. The table columns are 'Name', 'Last run', 'Current status', 'Agent version', and 'Enabled'. The 'azureagent' row shows 'azureagent' in the Name column, 'Idle' in the Current status column, '4.258.1' in the Agent version column, and an 'On' toggle switch in the Enabled column.

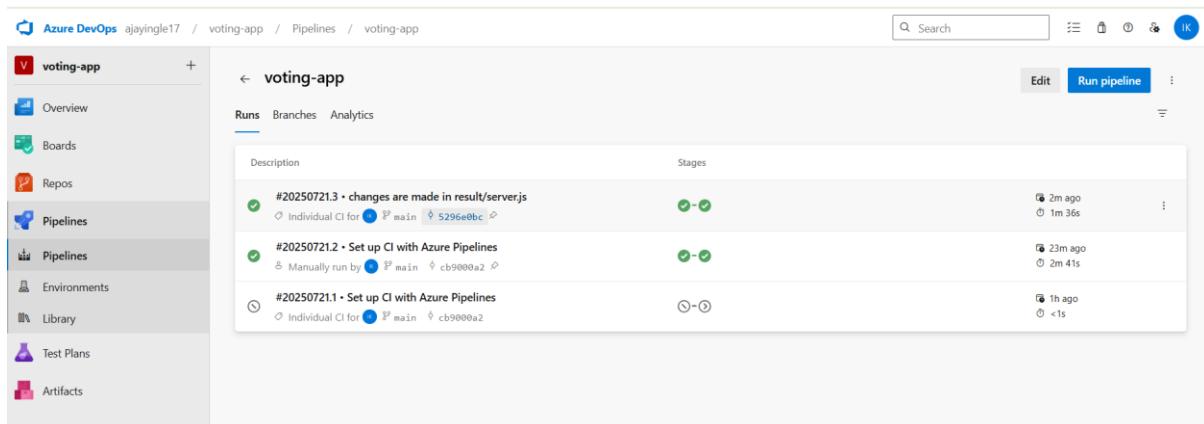
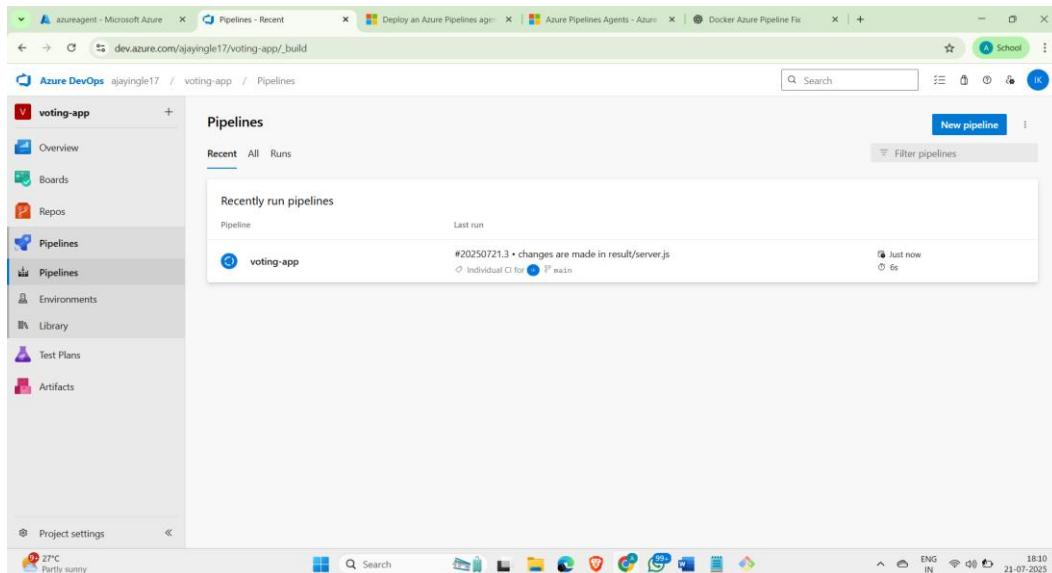
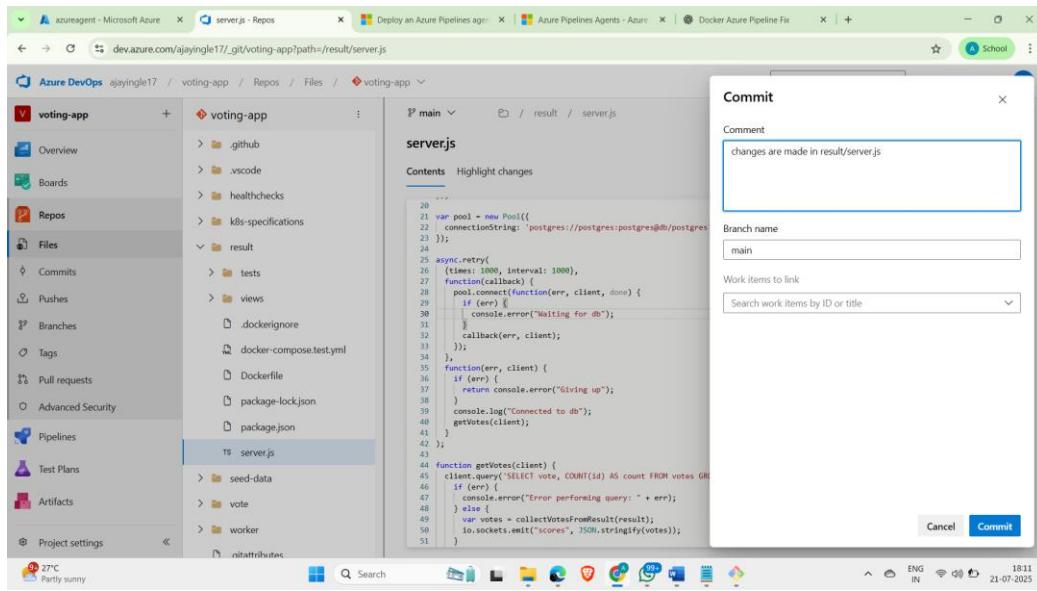
```
azureuser@azureagent:~/myagent
azureuser@azureagent:~$ ls
myagent
azureuser@azureagent:~$ cd myagent
azureuser@azureagent:~/myagent$ ./run.sh
Scanning for tool capabilities.
Connecting to the server.
2025-07-21 08:20:44Z: Listening for jobs
^CExiting...
azureuser@azureagent:~/myagent$ sudo apt install docker.io
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
```

```
azureuser@azureagent:~
azureuser@azureagent:~$ sudo usermod -aG docker azureuser
azureuser@azureagent:~$ sudo systemctl restart docker
azureuser@azureagent:~$ |
```

- Agent start listening for job

```
azureuser@azureagent:~/myagent
azureuser@azureagent:~$ cd myagent
azureuser@azureagent:~/myagent$ ./run.sh
Scanning for tool capabilities.
Connecting to the server.
2025-07-23 05:04:50Z: Listening for jobs
2025-07-23 05:35:35Z: Running job: Build
2025-07-23 05:36:15Z: Job Build completed with result: Succeeded
2025-07-23 05:36:19Z: Running job: Push
2025-07-23 05:36:54Z: Job Push completed with result: Succeeded
```





Azure DevOps ajayingle17 / voting-app / Pipelines

voting-service

```

main voting-app / azure-pipelines-vote.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 paths:
7 - include:
8   - vote
9
10 resources:
11 - repo: self
12
13 variables:
14 - # Container registry service connection established during pipeline creation
15 - dockerRegistryServiceConnection: '0559ca2f-34bb-4866-93f7-9fb6b1080a94'
16 - imageRepository: 'votingapp'
17 - containerRegistry: 'ajayazurecid.azurecr.io'
18 - dockerfilePath: '${Build.SourcesDirectory}/vote/Dockerfile'
19 - tag: '${Build.BuidId}'
20
21 pool:
22 - name: 'azureagent'
23
24 stages:
25 - stage: Build
26   displayName: Build
27   jobs:
28     - job: Build

```

Tasks

- .NET Core
- Android signing
- Ant
- App Center distribute
- App Center test
- Archive files
- ARM template deployment
- Azure App Configuration Export

Azure DevOps ajayingle17 / voting-app / Pipelines / voting-service / 20250721.1

#20250721.1 • Set up CI with Azure Pipelines

This run is being retained as one of 3 recent runs by main (Branch).

Summary **Code Coverage**

Individual CI by INGLE AJAY KAILAS

Repository and version	Time started and elapsed	Related	Tests and coverage
voting-app	Today at 6:36 PM 2m 28s	0 work items	Get started
main		0 artifacts	

Stages **Jobs**

```

graph LR
    Build[Build] --> Push[Push]
    subgraph Stages [Stages]
        Build
        Push
    end
    subgraph Jobs [Jobs]
        Build
        Push
    end

```

Azure DevOps ajayingle17 / voting-app / Pipelines

Review your pipeline YAML

New pipeline

```

voting-app / azure-pipelines-worker.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 paths:
7 - include:
8   - worker
9
10 resources:
11 - repo: self
12
13 variables:
14 - # Container registry service connection established during pipeline creation
15 - dockerRegistryServiceConnection: '4301eeef-e4a1-4a0e-8262-a73ec22670ce'
16 - imageRepository: 'workerapp'
17 - containerRegistry: 'ajayazurecid.azurecr.io'
18 - dockerfilePath: '${Build.SourcesDirectory}/worker/Dockerfile'
19 - tag: '${Build.BuidId}'
20
21 pool:
22 - name: 'azureagent'
23
24 stages:

```

Variables **Save and run**

Tasks

- .NET Core
- Android signing
- Ant
- App Center distribute
- App Center test
- Archive files
- ARM template deployment

Screenshot of the Azure DevOps Pipelines interface showing recently run pipelines for the 'voting-app' project.

The pipeline list includes:

- result-service**: Last run #20250722.1 • Updated server.js (Individual CI for main) - 12h ago (2m 7s)
- voting-service**: Last run #20250722.1 • changes are made in vote app.py (Individual CI for main) - 12h ago (1m 54s)
- Worker Services**: Last run #20250722.2 • changes arwe made in worker Dockerfile (Individual CI for main) - 12h ago (6m 43s)

Project settings and system status are visible at the bottom.

Pipeline	Last run	Time Ago	Duration
result-service	#20250722.1 • Updated server.js Individual CI for main	12h ago	2m 7s
voting-service	#20250722.1 • changes are made in vote app.py Individual CI for main	12h ago	1m 54s
Worker Services	#20250722.2 • changes arwe made in worker Dockerfile Individual CI for main	12h ago	6m 43s

• Create the K8 cluster

Azure Kubernetes Service (AKS) manages your hosted Kubernetes environment, making it quick and easy to deploy and manage containerized applications without container orchestration expertise. It also eliminates the burden of ongoing operations and maintenance by provisioning, upgrading, and scaling resources on demand, without taking your applications offline. [Learn more](#)

Project details
Select a subscription to manage deployed resources and costs. Use resource groups like folders to organize and manage all your resources.

Subscription *

Resource group * [Create new](#)

Cluster details
Cluster preset configuration *
To quickly customize your Kubernetes cluster, choose one of the preset configurations above. You can modify these configurations at any time.

[Previous](#) [Next](#) [Review + create](#) [Give feedback](#)

azuredevops Kubernetes service

Overview

Properties

Resource group : [azurecid](#)

Power state : Running

Cluster operation status : Succeeded

Subscription : Azure for Students

Location : Australia Central

Subscription ID : 0587b5e2-0375-4611-af56-ca8a4d38810c

Fleet Manager : Click here to assign

Tags (edit) : Add tags

Get started Properties Monitoring Recommendations

Kubernetes services

Encryption type : Encryption at-rest with a platform-managed key

Virtual node pools : Not enabled

Node pools

Node pools : 1 node pool

Kubernetes versions : 1.32.5

Connect to azureddevops

Cloud shell Azure CLI Run command

Connect to your cluster using command line tooling to interact directly with cluster using kubectl, the command line tool for Kubernetes. Kubectl is available within the Azure Cloud Shell by default and can also be installed locally.

Prerequisites

1. [Install Azure CLI](#)
2. [Install kubectl](#)

Set cluster context

1. Open terminal
2. Run the following commands

Login to your azure account
`az login`

Set the cluster subscription
`az account set --subscription 0587b5e2-0375-4611-af56-ca8a4d38810c`

Download cluster credentials
`az aks get-credentials --resource-group azurecid --name azureddevops --overwrite`

Sample commands

- Connect the cluster using azure cli

```

MINGW64/c/Users/ajayi
guration at https://go.microsoft.com/fwlink/?linkid=2271236
If you encounter any problem, please open an issue at https://aka.ms/azclibug
[warning] The login output has been updated. Please be aware that it no longer displays the full list of available subscriptions by default.

$ az aks install-cli
The detected architecture of current device is "amd64", and the binary for "amd64" will be downloaded. If the detection is wrong, please download and install the binary corresponding to the appropriate architecture.
No version specified, will get the latest version of kubectl from "https://dl.k8s.io/release/stable.txt"
Downloading client to "C:\Users\ajayi\.azure-kubectl\kubectl.exe" from "https://dl.k8s.io/release/v1.33.3/bin/windows/amd64/kubectl.exe"
The installation directory "C:\Users\ajayi\.azure-kubectl" has been successfully appended to the user path, the configuration will only take effect in the new command sessions. Please re-open the command window.
No version specified, will get the latest version of kubelogin from "https://api.github.com/repos/Azure/kubelogin/releases/latest"
Download client to "C:\Users\ajayi\AppData\Local\Temp\tmp6f7wfk69\kubelogin.zip" from "https://github.com/Azure/kubelogin/releases/download/v0.2.10/kubelogin.zip"
Moving binary to "C:\Users\ajayi\.azure-kubelogin\kubelogin.exe" from "C:\Users\ajayi\AppData\Local\Temp\tmp6f7wfk69\bin\windows_amd64\kubelogin.exe"
The installation directory "C:\Users\ajayi\.azure-kubelogin" has been successfully appended to the user path, the configuration will only take effect in the new command sessions. Please re-open the command window.

$ az aks get-credentials --resource-group azurecid --name azuredevops --overwrite-existing
Merged "azuredevops" as current context in C:/users/ajayi/.kube/config

$ az aks list -o table
+-----+-----+-----+-----+-----+-----+
| Name | Location | ResourceGroup | KubernetesVersion | CurrentKubernetesVersion | ProvisioningState | Fqdn |
+-----+-----+-----+-----+-----+-----+
| azuredevops | australiacentral | azurecid | 1.32.5 | 1.32.5 | Succeeded | azuredevops-dns-3.cwhqfkt.hcp.australiacentral.azmk8s.io |
+-----+-----+-----+-----+-----+-----+
$ |

```

The screenshot shows a Windows terminal window titled 'MINGW64/c/Users/ajayi'. It displays the output of several Azure CLI commands. The first command, 'az aks install-cli', installs the kubectl and kubelogin clients. The second command, 'az aks get-credentials', sets the current context to 'azuredevops'. The third command, 'az aks list -o table', shows a table of clusters, indicating that 'azuredevops' is running in the 'australiacentral' region with version 1.32.5 and a provisioning state of 'Succeeded'. The terminal window also shows the system tray with weather information (24°C, mostly cloudy) and system status icons.

- Install the ARGOCD

The screenshot shows a web browser window with the URL 'argo-cd.readthedocs.io/en/stable/getting_started/'. The page title is 'Getting Started'. On the left, there's a sidebar with navigation links for Argo CD - Declarative and GitOps CD for Kubernetes, including Overview, Understand The Basics, Core Concepts, Getting Started, Operator Manual, User Guide, Developer Guide, FAQ, Security Considerations, Support, Roadmap, and Releases. A 'Monetize your audience: Fund an OSS project or website with EthicalAds.' banner is visible. The main content area starts with a 'Requirements' section listing: 'Installed kubectl command-line tool.', 'Have a kubeconfig file (default location is ~/.kube/config).', and 'CoreDNS. Can be enabled for microk8s by microk8s enable dns & microk8s stop & microk8s start'. Below this is a '1. Install Argo CD' section with a 'Tip' box containing the command 'kubectl create namespace argocd' and 'kubectl apply -n argocd -f https://raw.githubusercontent.com/argoproj/argo-cd/stable/manifests/install.yaml'. A 'Warning' box notes that ClusterRoleBinding resources reference the 'argocd' namespace. A 'Table of contents' sidebar on the right lists steps from 1 to 7, such as 'Install Argo CD' and 'Sync (Deploy) The Application'. The bottom of the page includes a 'stable' dropdown, a footer with a '24°C Mostly cloudy' icon, and a system tray.

```

ajayi@ajay 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.apirextensions.k8s.io/applications.argoproj.io created
customresourcedefinition.apirextensions.k8s.io/applicationsets.argoproj.io created
customresourcedefinition.apirextensions.k8s.io/appprojects.argoproj.io created
serviceaccount/argocd-application-controller created
serviceaccount/argocd-applicationset-controller created
serviceaccount/argocd-dex-server created
serviceaccount/argocd-notifications-controller created
serviceaccount/argocd-redis created
serviceaccount/argocd-repo-server created
serviceaccount/argocd-server created
role.rbac.authorization.k8s.io/argocd-application-controller created
role.rbac.authorization.k8s.io/argocd-applicationset-controller created
role.rbac.authorization.k8s.io/argocd-dex-server created
role.rbac.authorization.k8s.io/argocd-notifications-controller created
role.rbac.authorization.k8s.io/argocd-redis created
role.rbac.authorization.k8s.io/argocd-server created
clusterrole.rbac.authorization.k8s.io/argocd-application-controller created
clusterrole.rbac.authorization.k8s.io/argocd-applicationset-controller created
clusterrole.rbac.authorization.k8s.io/argocd-server created
rolebinding.rbac.authorization.k8s.io/argocd-application-controller created
rolebinding.rbac.authorization.k8s.io/argocd-dex-server created
rolebinding.rbac.authorization.k8s.io/argocd-notifications-controller created
rolebinding.rbac.authorization.k8s.io/argocd-redis created
rolebinding.rbac.authorization.k8s.io/argocd-server created
clusterrolebinding.rbac.authorization.k8s.io/argocd-application-controller created
clusterrolebinding.rbac.authorization.k8s.io/argocd-applicationset-controller created
clusterrolebinding.rbac.authorization.k8s.io/argocd-server created
configmap/argocd-cm created
configmap/argocd-cmd-params-cm created
configmap/argocd-gpg-keys-cm created
configmap/argocd-notifications-cm created
configmap/argocd-rbac-cm created
configmap/argocd-ssh-known-hosts-cm created

```

```

ajayi@ajay MINGW64 ~
$ kubectl get pods -n argocd
NAME                      READY   STATUS    RESTARTS   AGE
argocd-application-controller-0   1/1    Running   0          110s
argocd-applicationset-controller-655cc58ff8-fgl48  1/1    Running   0          113s
argocd-dex-server-7d9dfb4fb8-f5494  1/1    Running   0          113s
argocd-notifications-controller-6c6848bc4c-lwz85  1/1    Running   0          112s
argocd-redis-656c79549c-k85rj   1/1    Running   0          112s
argocd-repo-server-856b768fd9-hwlzf  1/1    Running   0          111s
argocd-server-99c485944-5c58n   1/1    Running   0          111s

```

- Configure ARGOCD

\$ kubectl get secrets -n argocd

```

ajayi@Ajay MINGW64 ~
$ kubectl get secrets -n argocd
NAME              TYPE      DATA  AGE
argocd-initial-admin-secret  Opaque    1    5m12s
argocd-notifications-secret  Opaque    0    5m49s
argocd-redis        Opaque    1    5m18s
argocd-secret       Opaque    5    5m49s

```

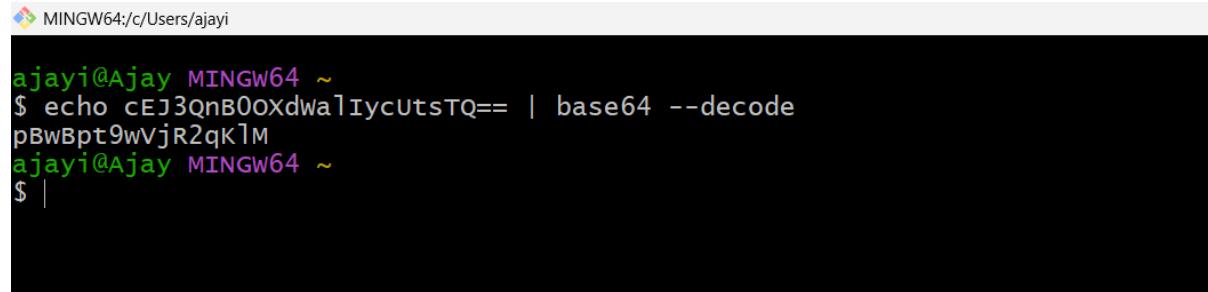
\$ kubectl edit argocd-initial-admin-secret -n argocd

```

# 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: cEJ3QnB00XdWalIycUtsTQ==
kind: Secret
metadata:
  creationTimestamp: "2025-07-22T18:16:03Z"
  name: argocd-initial-admin-secret
  namespace: argocd
  resourceVersion: "24418"
  uid: b21e8b54-adae-4ad2-b5af-a9e3194894db
type: Opaque

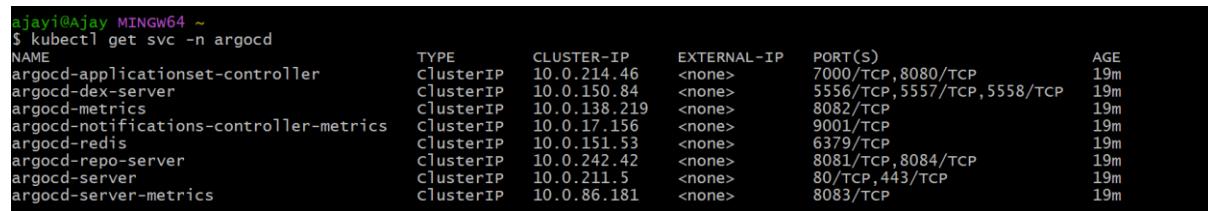
```

```
$ echo cEJ3QnB0OXdWallycUtsTQ== | base64 --decode
```



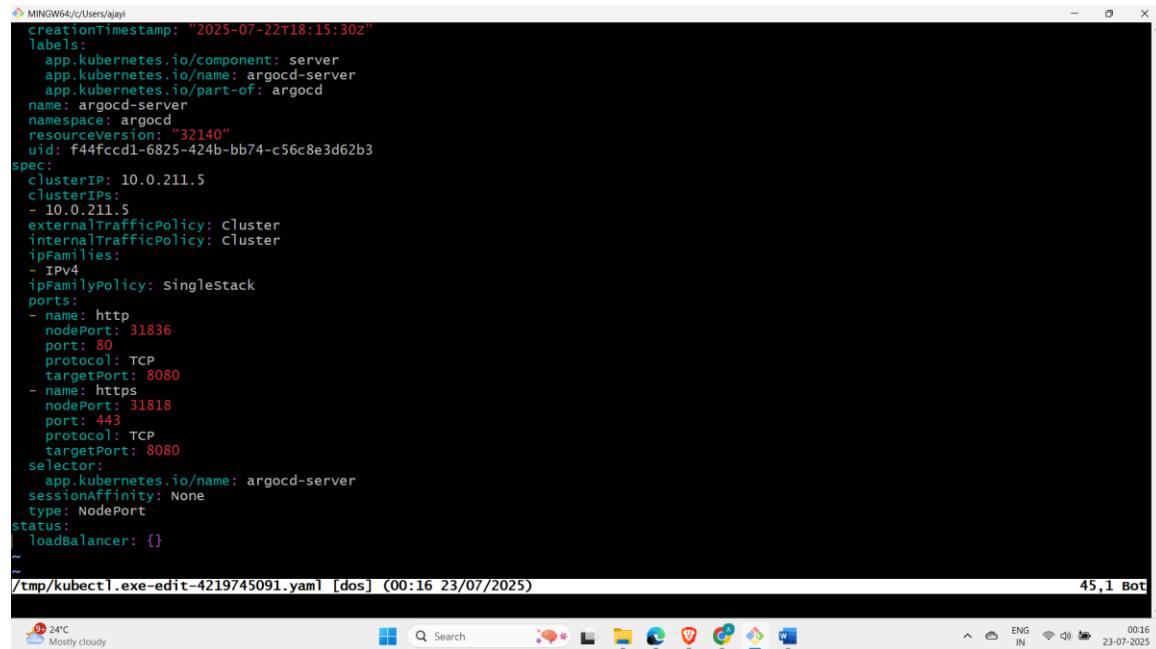
```
ajayi@Ajay MINGW64 ~
$ echo cEJ3QnB0OXdWallycUtsTQ== | base64 --decode
pBwBpt9wVjR2qKlM
ajayi@Ajay MINGW64 ~
$ |
```

```
$ kubectl get svc -n argocd
```



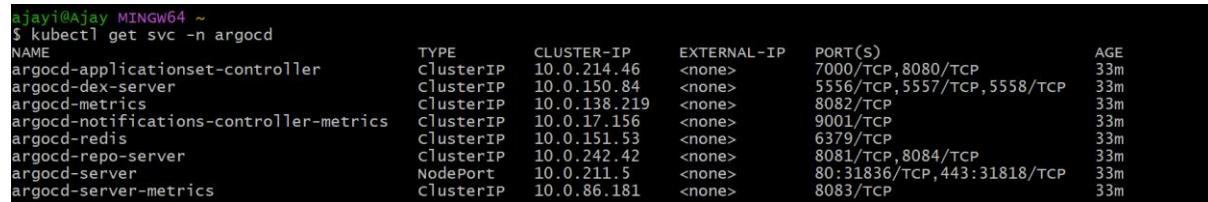
```
ajayi@Ajay MINGW64 ~
$ kubectl get svc -n argocd
NAME           TYPE      CLUSTER-IP   EXTERNAL-IP  PORT(S)          AGE
argocd-applicationset-controller  ClusterIP  10.0.214.46 <none>        7000/TCP,8080/TCP  19m
argocd-dex-server    ClusterIP  10.0.150.84 <none>        5556/TCP,5557/TCP,5558/TCP  19m
argocd-metrics     ClusterIP  10.0.138.219 <none>       8082/TCP          19m
argocd-notifications-controller-metrics ClusterIP  10.0.17.156 <none>       9001/TCP          19m
argocd-redis       ClusterIP  10.0.151.53 <none>        6379/TCP          19m
argocd-repo-server  ClusterIP  10.0.242.42 <none>        8081/TCP,8084/TCP  19m
argocd-server      ClusterIP  10.0.211.5  <none>        80/TCP,443/TCP   19m
argocd-server-metrics ClusterIP  10.0.86.181 <none>        8083/TCP          19m
```

```
$ kubectl edit svc argocd-server -n argocd
```



```
creationTimestamp: "2025-07-22T18:15:30Z"
labels:
  app.kubernetes.io/component: server
  app.kubernetes.io/name: argocd-server
  app.kubernetes.io/part-of: argocd
  name: argocd-server
  namespace: argocd
  resourceversion: "32140"
  uid: f44fcfd1-6825-424b-bb74-c56c8e3d62b3
spec:
  clusterIP: 10.0.211.5
  clusterIPs:
  - 10.0.211.5
  externalTrafficPolicy: Cluster
  internalTrafficPolicy: Cluster
  ipFamilies:
  - IPv4
  ipFamilyPolicy: SingleStack
  ports:
  - name: http
    nodePort: 31836
    port: 80
    protocol: TCP
    targetPort: 8080
  - name: https
    nodePort: 31818
    port: 443
    protocol: TCP
    targetPort: 8080
  selector:
    app.kubernetes.io/name: argocd-server
    sessionAffinity: None
    type: NodePort
  status:
    loadBalancer: {}
~
```

45,1 Bot



```
ajayi@Ajay MINGW64 ~
$ kubectl get svc -n argocd
NAME           TYPE      CLUSTER-IP   EXTERNAL-IP  PORT(S)          AGE
argocd-applicationset-controller  ClusterIP  10.0.214.46 <none>        7000/TCP,8080/TCP  33m
argocd-dex-server    ClusterIP  10.0.150.84 <none>        5556/TCP,5557/TCP,5558/TCP  33m
argocd-metrics     ClusterIP  10.0.138.219 <none>       8082/TCP          33m
argocd-notifications-controller-metrics ClusterIP  10.0.17.156 <none>       9001/TCP          33m
argocd-redis       ClusterIP  10.0.151.53 <none>        6379/TCP          33m
argocd-repo-server  ClusterIP  10.0.242.42 <none>        8081/TCP,8084/TCP  33m
argocd-server      NodePort   10.0.211.5  <none>        80:31836/TCP,443:31818/TCP  33m
argocd-server-metrics ClusterIP  10.0.86.181 <none>        8083/TCP          33m
```

```

ajayi@Ajay MINGW64 ~
$ kubectl get svc -n argocd
NAME                      TYPE        CLUSTER-IP      EXTERNAL-IP   PORT(S)          AGE
argocd-applicationset-controller   ClusterIP  10.0.214.46    <none>       7000/TCP,8080/TCP  33m
argocd-dex-server             ClusterIP  10.0.150.84   <none>       5556/TCP,5557/TCP,5558/TCP  33m
argocd-metrics               ClusterIP  10.0.138.219   <none>       8082/TCP          33m
argocd-notifications-controller-metrics ClusterIP  10.0.17.156   <none>       9001/TCP          33m
argocd-redis                 ClusterIP  10.0.151.53   <none>       6379/TCP          33m
argocd-repo-server            ClusterIP  10.0.242.42   <none>       8081/TCP,8084/TCP  33m
argocd-server                NodePort    10.0.211.5    <none>       80:31836/TCP,443:31818/TCP  33m
argocd-server-metrics         ClusterIP  10.0.86.181   <none>       8083/TCP          33m

ajayi@Ajay MINGW64 ~
$ kubectl get nodes -o wide
NAME           STATUS   ROLES   AGE     VERSION   INTERNAL-IP   EXTERNAL-IP   OS-IMAGE          KERNEL-VERSION
ON            CONTAINER-RUNTIME
aks-agentpool-68424352-vmss000000 Ready    <none>   118m   v1.32.5   10.224.0.4    20.248.120.233   Ubuntu 22.04.5 LTS   5.15.0-1091-
azure         containerd://1.7.27-1

```

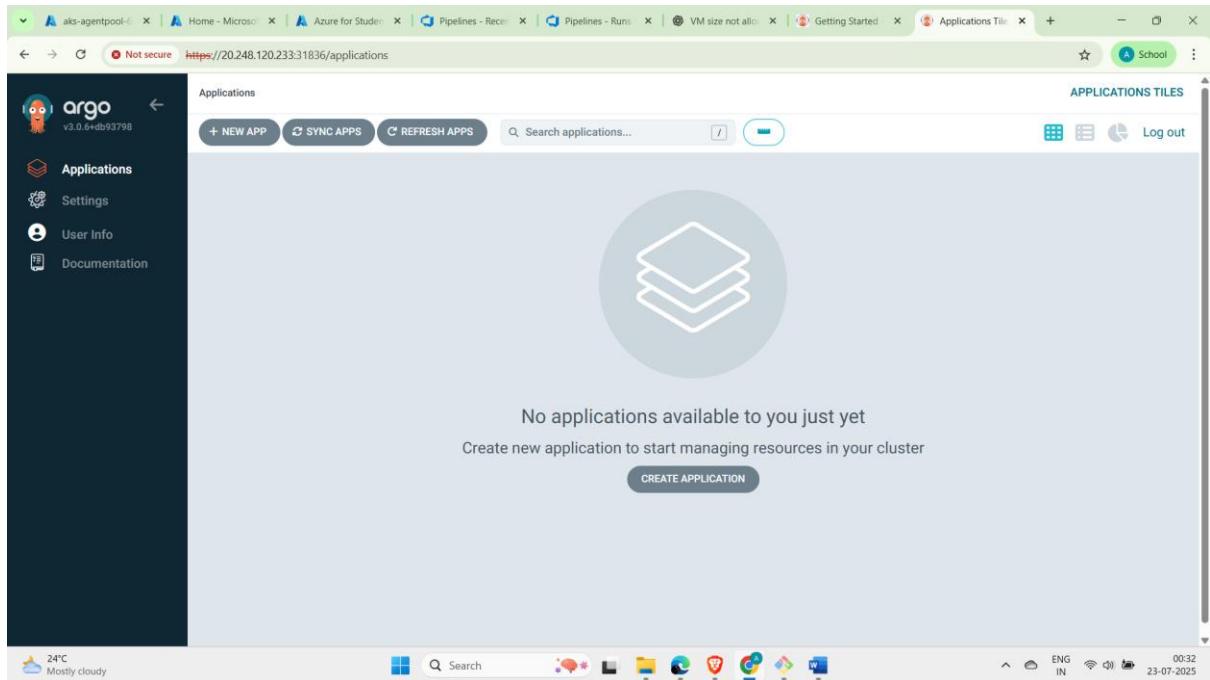
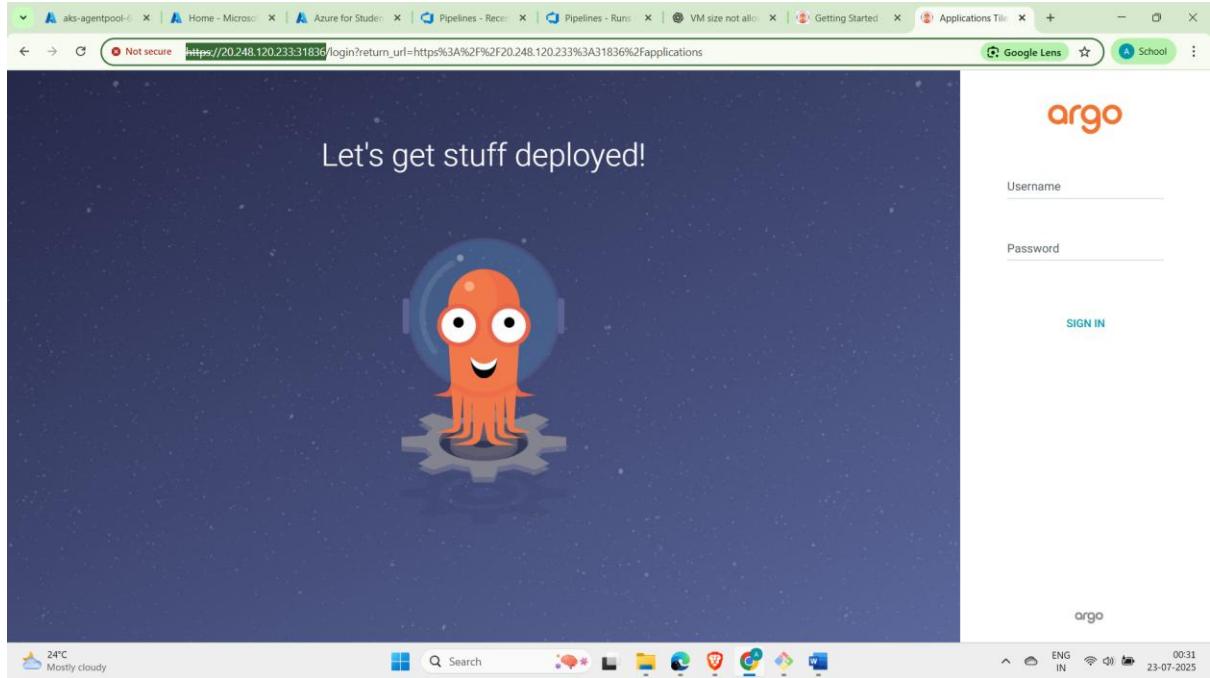
- Copy the node ip and argocd-server port and past in browser
- Before that expose the port in azure k8 to external world

The screenshot shows the Microsoft Azure Compute Infrastructure blade for a Virtual Machine Scale Set (VMSS). The main area displays the 'Instances' tab for the scale set 'aks-agentpool-68424352-vmss'. It shows one instance named 'aks-agentpool-68424352-vmss_0' in a 'Running' state. The left sidebar includes options like Overview, All resources, Infrastructure, and Virtual machines.

The screenshot shows the Microsoft Azure Compute Infrastructure blade for the 'aks-agentpool-68424352-vmss_0' instance's network settings. The 'Add inbound security rule' dialog is open, showing fields for Source (Any), Destination (Any), and Destination port ranges (31836). The dialog also includes sections for Protocol (Any), Action (Allow), and an 'Add' button.

- Now enter the ip of node followed by port in browser

<https://20.248.120.233:31836/>



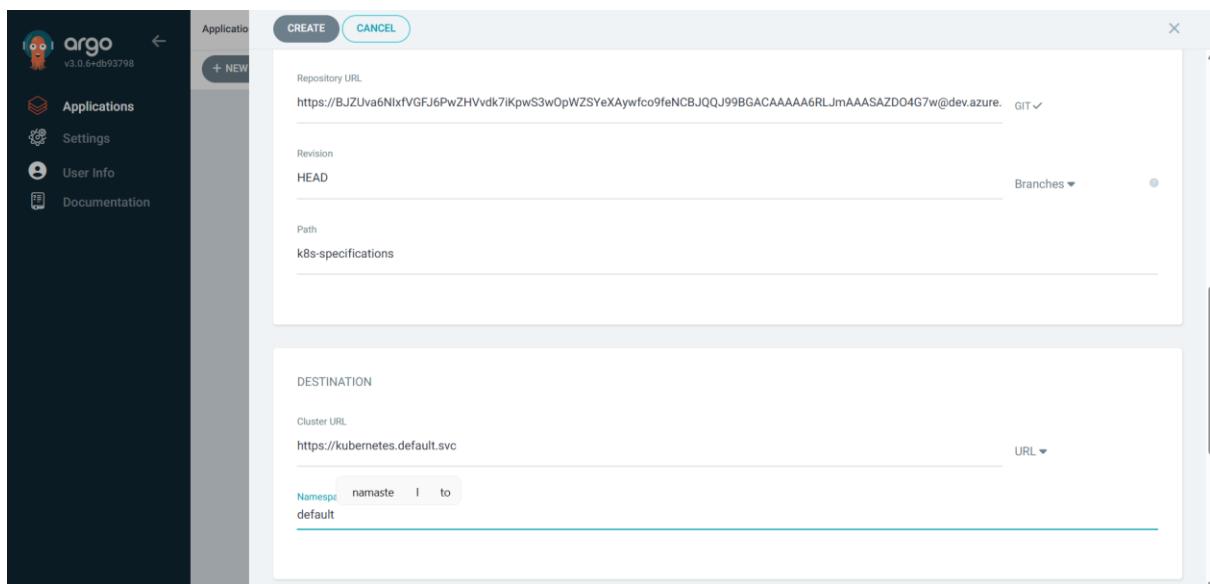
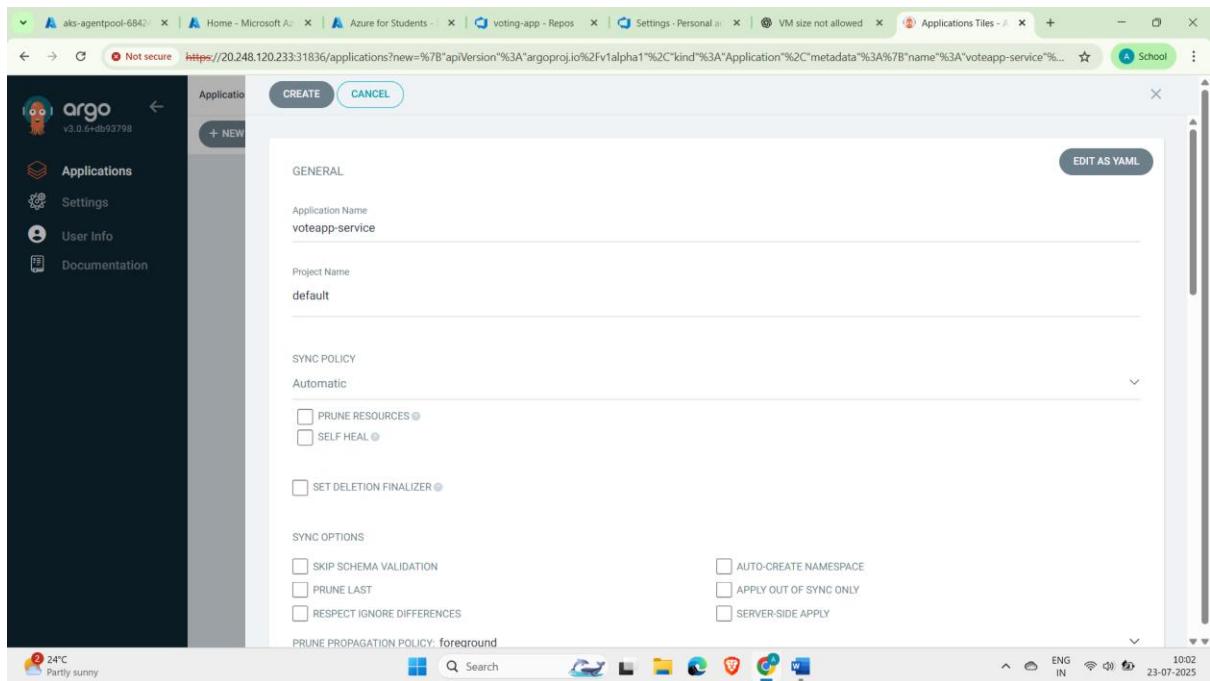
- Get The Access Token from Azure DevOps to connect the repositories

The screenshot shows the 'Personal Access Tokens' section of the Azure DevOps settings. A new token is being created with the name 'ajayingle17'. The token has custom-defined scopes: 'Work Items', 'Code', and 'Read & write'. The token will expire in 30 days on 8/22/2025.

- Go to the setting of argocd and click on connect
- Get the repo link and paste in repo link section
- Inside the repo link replace the organization name with the access token

The screenshot shows the 'Repositories / Settings' page in argocd. A new repository is being connected using the 'VIA HTTP/HTTPS' method. The type is set to 'git'. The repository URL is provided as `https://BJZUva6NixfVGfJ6PwZHvdk7iKpwS3wOpWzSYeAywfco9feNCBJQQJ99BGACAAAAA6RLJmAAASAZDO4G7w@dev.azure.com/ajayingle17/voting-app/_git`. The connection status is marked as 'Successful'.

- Now create the argocd Application

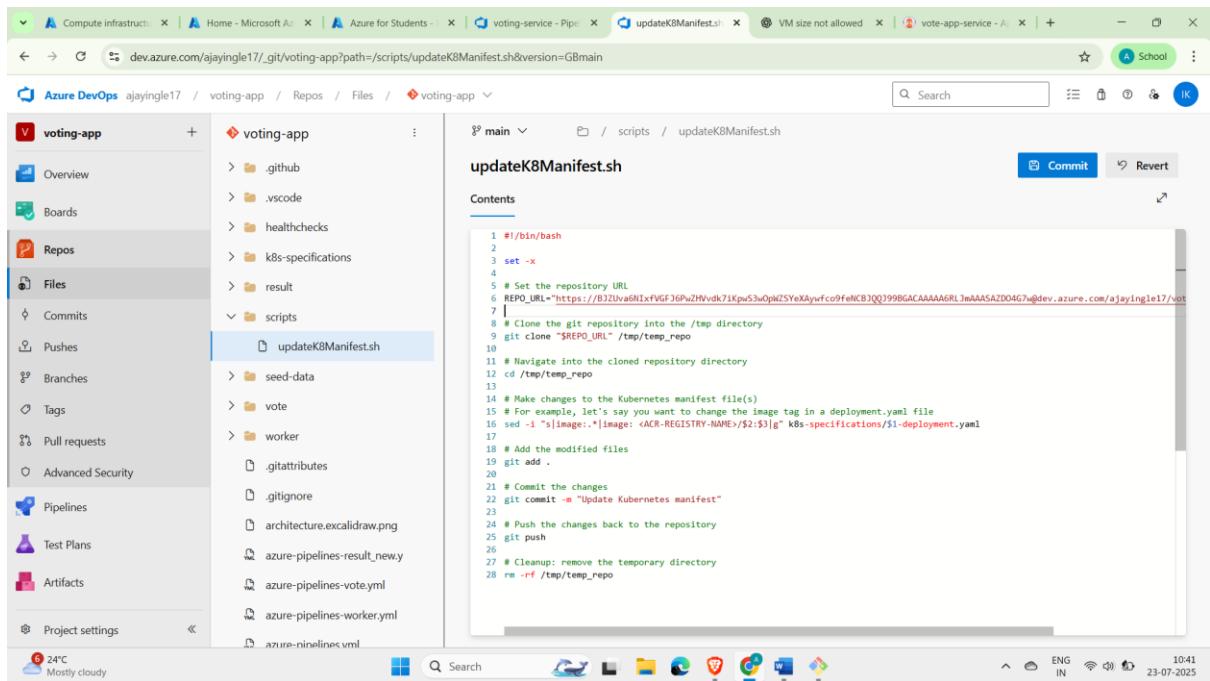


The screenshot shows the Argo UI interface. On the left, there's a sidebar with navigation links: Applications, Settings, User Info, Documentation, Application filters (Favorites Only), SYNC STATUS (Unknown: 0, Synced: 1, OutOfSync: 0), and HEALTH STATUS (Progressing: 0, Suspended: 0, Healthy: 1, Degraded: 0, Missing: 0). The main area displays the 'vote-app-service' application details. It shows the project is 'default', the status is 'Healthy & Synced', and the repository is 'https://BJJUva6NlxVGfJ6PwZHvdk7l...'. The target ref is 'HEAD', path is 'k8s-specifications', destination is 'in-cluster', namespace is 'argocd', and it was created and last synced at 07/23/2025 10:13:27 (a minute ago). Below this are buttons for SYNC, REFRESH, and DELETE. At the bottom right, there's a weather widget showing 24°C Mostly cloudy, and a system tray with icons for search, file, browser, and power.

- Now argocd Application is configured

This screenshot is similar to the previous one but shows more detailed application details. It includes tabs for DETAILS, DIFF, SYNC, SYNC STATUS, HISTORY AND ROLLBACK, DELETE, and REFRESH. The SYNC STATUS shows 'Synced' to 'Be7e5bb' with a note about auto sync being enabled. The APPLICATION DETAILS TREE section shows a hierarchical deployment of services: db, redis, result, vote, and worker, each with their own deployment logs and pod status.

- Create the updateK8Manifest.sh script to replace the new image name



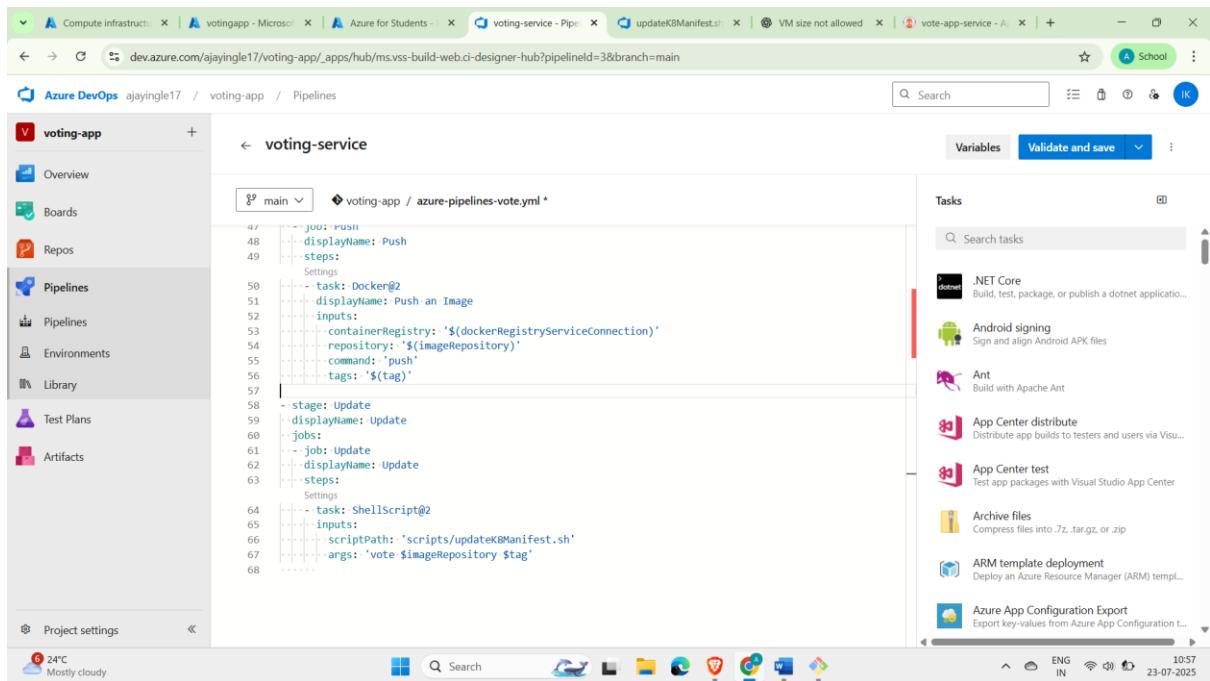
The screenshot shows the Azure DevOps interface for a repository named 'voting-app'. The 'Scripts' folder contains an 'updateK8Manifest.sh' file, which is selected and displayed in the code editor. The code is a bash script that clones a GitHub repository, navigates to its directory, makes changes to a Kubernetes manifest file, adds modified files, commits them with a message, pushes the changes back to the repository, and finally removes the temporary directory.

```

1 #!/bin/bash
2
3 set -x
4
5 # Set the repository URL
6 REPO_URL="https://B2Uva6H1xFVGfJ6PwHVvdk71Kpu53uOpkZSYeXayrfco9feNCBjQQJ99BGACAAAAA6RJmAAASAZD04G7u@dev.azure.com/ajayingle17/vot"
7
8 # Clone the git repository into the /tmp directory
9 git clone "$REPO_URL" /tmp/temp_repo
10
11 # Navigate into the cloned repository directory
12 cd /tmp/temp_repo
13
14 # Make changes to the Kubernetes manifest file(s)
15 # For example, let's say you want to change the image tag in a deployment.yaml file
16 sed -i "s|image: <ACR-REGISTRY-NAME>:52:$3|g" k8s-specifications/$1-deployment.yaml
17
18 # Add the modified files
19 git add .
20
21 # Commit the changes
22 git commit -m "Update Kubernetes manifest"
23
24 # Push the changes back to the repository
25 git push
26
27 # Cleanup: remove the temporary directory
28 rm -rf /tmp/temp_repo

```

- Add the update stage in every Pipelines



The screenshot shows the Azure DevOps pipeline editor for a pipeline named 'voting-service'. The configuration file 'azure-pipelines-vote.yml' is open, showing a 'push' stage and an 'update' stage. The 'push' stage uses a Docker task to push an image to a container registry. The 'update' stage uses a ShellScript task to run the 'updateK8Manifest.sh' script. The pipeline editor also displays a sidebar with various tasks like .NET Core, Android signing, and App Center distribute.

```

47 - job: Push
48   displayName: Push
49   steps:
50     - task: Docker@2
51       displayName: Push an Image
52       inputs:
53         containerRegistry: '$(dockerRegistryServiceConnection)'
54         repository: '$(imageRepository)'
55         command: 'push'
56         tags: '$(tag)'
57
58 - stage: Update
59   displayName: Update
60   jobs:
61     - job: Update
62       displayName: Update
63       steps:
64         - task: ShellScript@2
65           inputs:
66             filePath: 'scripts/updateK8Manifest.sh'
67             args: 'vote $imageRepository $tag'
68

```

Azure DevOps Pipelines - Run 20250723.1

This run is being retained as one of 3 recent runs by main (Branch).

Manually run by INGLE AJAY KAILAS

Repository and version: voting-app@main · 65e2a8cf

Time started and elapsed: Today at 11:05 AM · 1m 57s

Related: 0 work items · 0 artifacts

Tests and coverage: Get started

View 3 changes

Warnings: 1

No data was written into the file /home/azureuser/myagent/_work/_temp/task_outputs/build_1753248966802.txt

Build → Build → Build an Image

Stages: Jobs

Build (1 job completed) → Push (1 job completed) → Update (1 job completed)

Microsoft Azure

Container registries > ajayazurecid

You are viewing a new version of Browse experience. Click here to access the old experience.

Name: ajayazurecid

Registry name: ajayazurecid

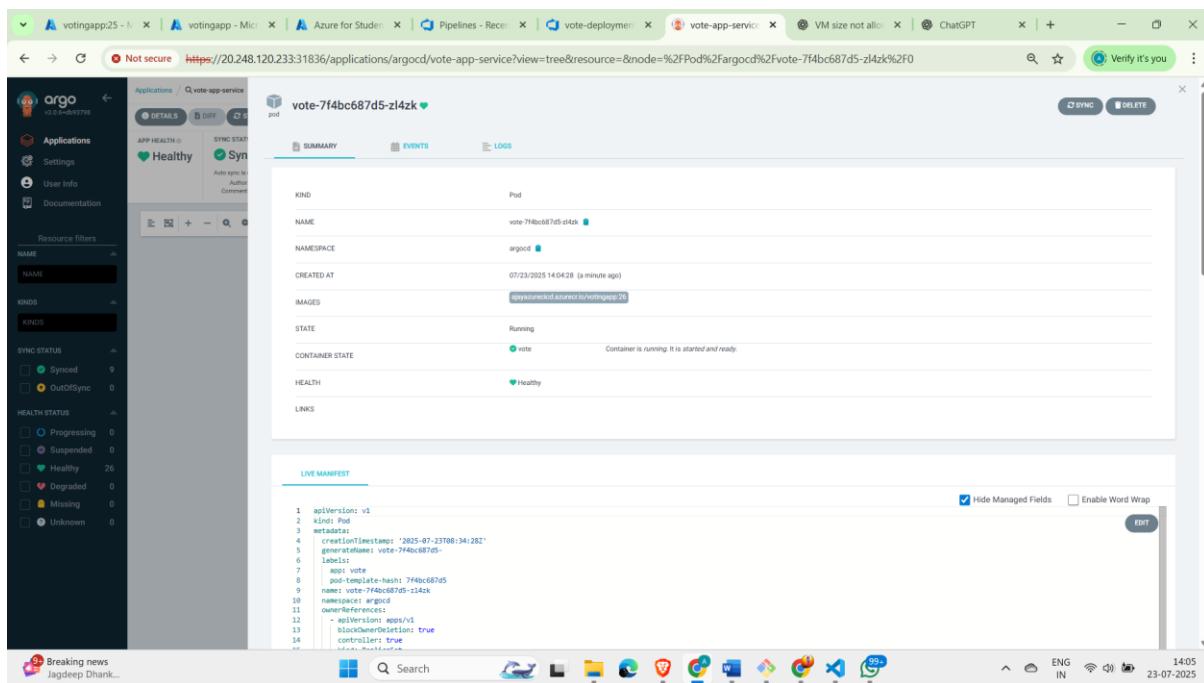
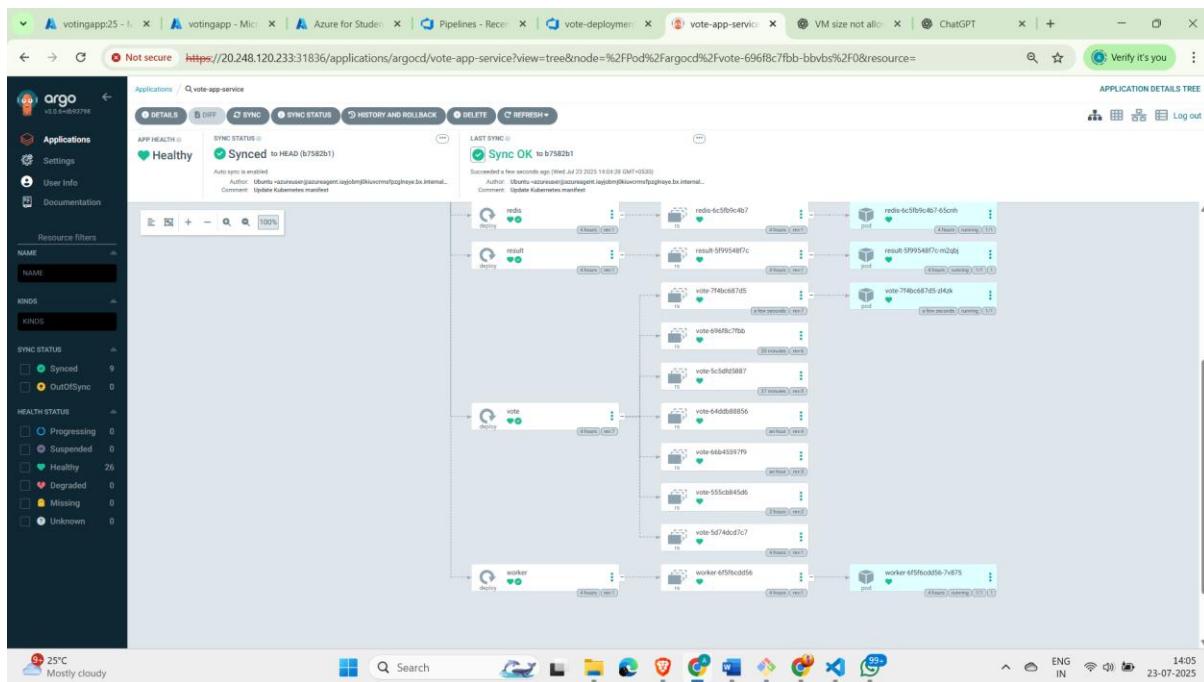
Login server: ajayazurecid.azurecr.io

Admin user:

Access keys

Copy to clipboard

```
ajayi@Ajay MINGW64 ~
$ kubectl create secret docker-registry acr-secret \
    --namespace argocd \
    --docker-server=ajayazurecid.azurecr.io \
    --docker-username=ajayazurecid \
    --docker-password=WYATBRFQy87XobHc3FW8APjvz2LwPbz8hA2+dAR6D1+ACRC4T/xR -n argocd
secret/acr-secret created
```



- The Pod are running now
- Accesss the ip and port and paste in browsers

```
ajayi@Ajay MINGW64 ~
$ kubectl get svc -n argocd
NAME           TYPE        CLUSTER-IP   EXTERNAL-IP   PORT(S)          AGE
argocd-applicationset-controller   ClusterIP  10.0.214.46  <none>        7000/TCP,8080/TCP 15h
argocd-dex-server     ClusterIP  10.0.150.84   <none>        5556/TCP,5557/TCP,5558/TCP 15h
argocd-metrics      ClusterIP  10.0.138.219  <none>        8082/TCP          15h
argocd-notifications-controller-metrics ClusterIP  10.0.17.156  <none>        9001/TCP          15h
argocd-redis       ClusterIP  10.0.151.53   <none>        6379/TCP          15h
argocd-repo-server   ClusterIP  10.0.242.42  <none>        8081/TCP,8084/TCP 15h
argocd-server       NodePort    10.0.211.5    <none>        80:31836/TCP,443:31818/TCP 15h
argocd-server-metrics ClusterIP  10.0.86.181   <none>        8083/TCP          15h
db                ClusterIP  10.0.170.24   <none>        5432/TCP          4h45m
redis              ClusterIP  10.0.144.203  <none>        6379/TCP          4h45m
result             NodePort    10.0.203.191  <none>        8081:31001/TCP    4h45m
vote               NodePort    10.0.208.10   <none>        8080:31000/TCP    4h45m

ajayi@Ajay MINGW64 ~
$ kubectl get node -o wide
NAME            STATUS  ROLES   AGE   VERSION   INTERNAL-IP   EXTERNAL-IP   OS-IMAGE          KERNEL-VERSION
aks-agentpool-68424352-vmss000000 Ready   <none>  16h   v1.32.5   10.224.0.4   20.248.120.233  Ubuntu 22.04.5 LTS   5.15.0-1091-a
zure containerd://1.7.27-1
```

- Allow the port 31000 by editing inbound rule

The screenshot shows the Azure portal interface for managing a VM scale set named 'aks-agentpool-68424352-vmss_0'. The 'Networking' section is selected, specifically the 'Network settings' tab. A network security group named 'aks-agentpool-24480629-rsg' is attached to the subnet 'aks-subnet'. The inbound port rules table lists five rules:

Priority	Name	Port	Protocol	Source	Destination	Action
100	AllowAnyCustom31836Inbound	31836	Any	Any	Any	Allow
110	AllowAnyCustom31000Inbound	31000	Any	Any	Any	Allow
65000	AllowVnetInBound	Any	Any	VirtualNetwork	VirtualNetwork	Allow
65001	AllowAzureLoadBalancerInBound	Any	Any	AzureLoadBalancer	Any	Allow
65500	DenyAllInBound	Any	Any	Any	Any	Deny

The rule 'AllowAnyCustom31000Inbound' is highlighted, indicating it has been edited. The browser address bar shows the URL for the Azure portal page.

The screenshot shows the Azure DevOps Pipelines interface for the 'voting-app' project. The left sidebar has 'Pipelines' selected. The main area displays 'Recently run pipelines' with three entries:

Pipeline	Last run	
Worker Services	#20250723.1 • Updated Dockerfile for check CI/CD Individual CI for 1st main	14m ago 3m 28s
result-service	#20250723.1 • Updated server.js for check CI/CD Individual CI for 1st main	15m ago 3m 38s
voting-service	#20250723.1 • Updated app.py for check CI/CD Individual CI for 1st main	1h ago 2m 7s

At the top right, there is a 'New pipeline' button and a 'Filter pipelines' search bar.

The screenshot shows the Argo CD application details for the 'vote-app-service'. The left sidebar has 'Applications' selected. The main area shows the 'vote-app-service' application with the following details:

- SYNC STATUS:** Synced to HEAD (Se40fa4)
- LAST SYNC:** Sync OK to Se40fa4 (Succeeded 2 minutes ago)
- PODS:** A grid of pods including db-74574d66dd, db-74685b959f, redis-6b86569d44, result-766ff7b8f, vote-7bb4b48976, and worker-7d4d66d6c, each with a green heart icon indicating healthy status.

At the bottom left, there is a weather widget showing 'Light rain At night'.

```

ajayi@Ajay MINGW64 ~
$ kubectl get svc -n argocd
NAME                   TYPE        CLUSTER-IP      EXTERNAL-IP   PORT(S)          AGE
argocd-applicationset-controller   ClusterIP   10.0.214.46   <none>       7000/TCP,8080/TCP   15h
argocd-dex-server                ClusterIP   10.0.150.84    <none>       5556/TCP,5557/TCP,5558/TCP   15h
argocd-metrics                  ClusterIP   10.0.138.219   <none>       8082/TCP   15h
argocd-notifications-controller-metrics   ClusterIP   10.0.17.156    <none>       9001/TCP   15h
argocd-redis                     ClusterIP   10.0.151.53    <none>       6379/TCP   15h
argocd-repo-server               ClusterIP   10.0.242.42    <none>       8081/TCP,8084/TCP   15h
argocd-server                    NodePort    10.0.211.5     <none>       80:31836/TCP,443:31818/TCP   15h
argocd-server-metrics            ClusterIP   10.0.170.24    <none>       5432/TCP   5h20m
db                                ClusterIP   10.0.144.203   <none>       6379/TCP   5h20m
redis                            NodePort    10.0.203.191   <none>       8081:31001/TCP   5h20m
result                           NodePort    10.0.208.10     <none>       8080:31000/TCP   5h20m

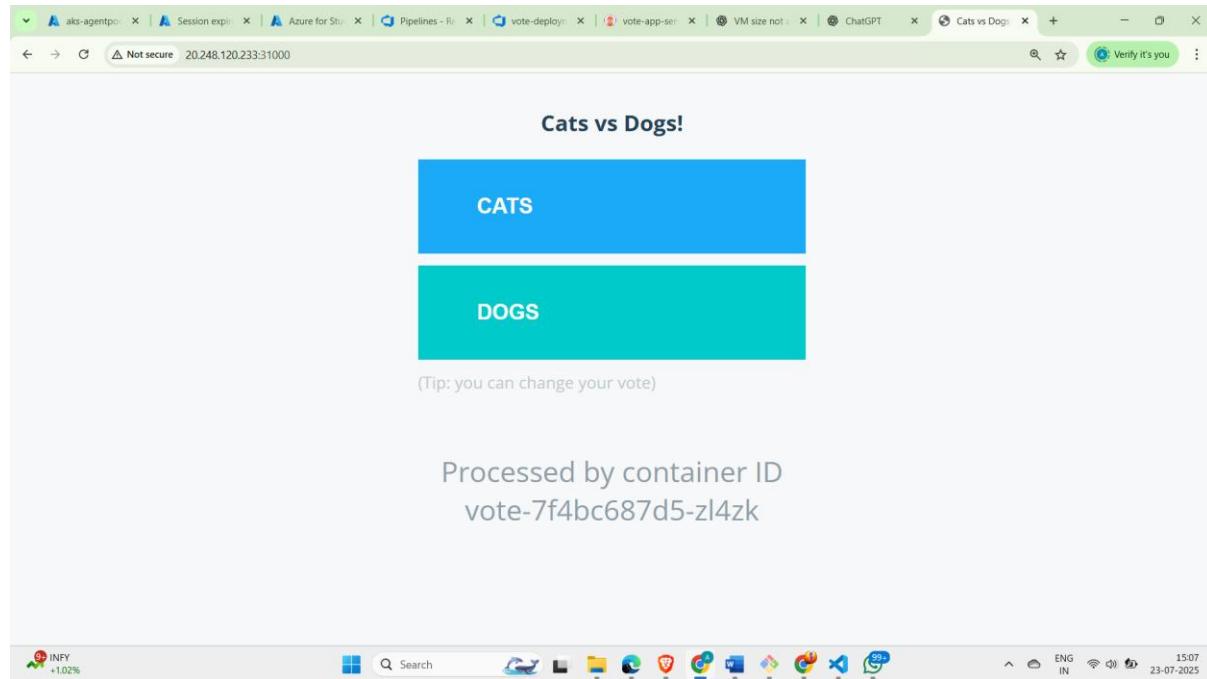
ajayi@Ajay MINGW64 ~
$ kubectl get node -o wide
NAME           STATUS  ROLES   AGE   VERSION INTERNAL-IP   EXTERNAL-IP   OS-IMAGE          KERNEL-VERSION
aks-agentpool-68424352-vmss000000   Ready   <none>  17h   v1.32.5  10.224.0.4   20.248.120.233  Ubuntu 22.04.5 LTS  5.15.0-1091-a
azure containerd://1.7.27-1

ajayi@Ajay MINGW64 ~
$ |

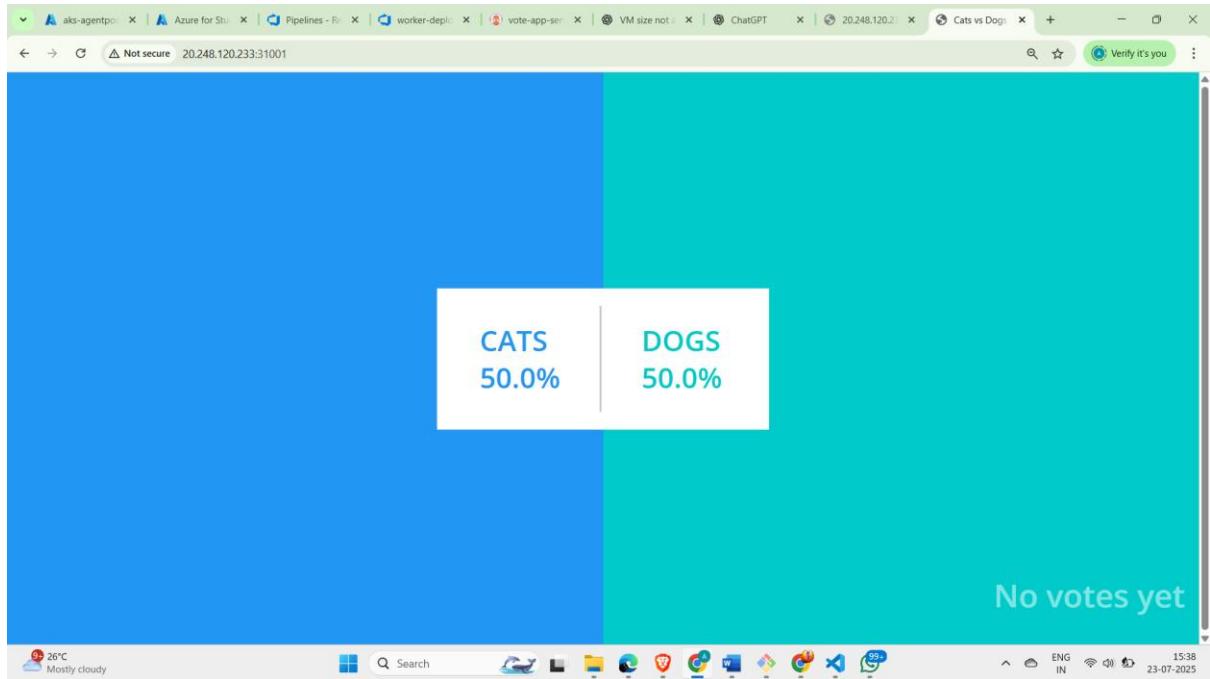
```

Now its Live

<http://20.248.120.233:31000/>



<http://20.248.120.233:31001/>



9. Conclusion

This project demonstrates the complete lifecycle of a microservices-based application, including development, containerization, orchestration, and deployment on cloud infrastructure. It also integrates modern DevOps practices such as CI/CD pipelines and GitOps via ArgoCD.