

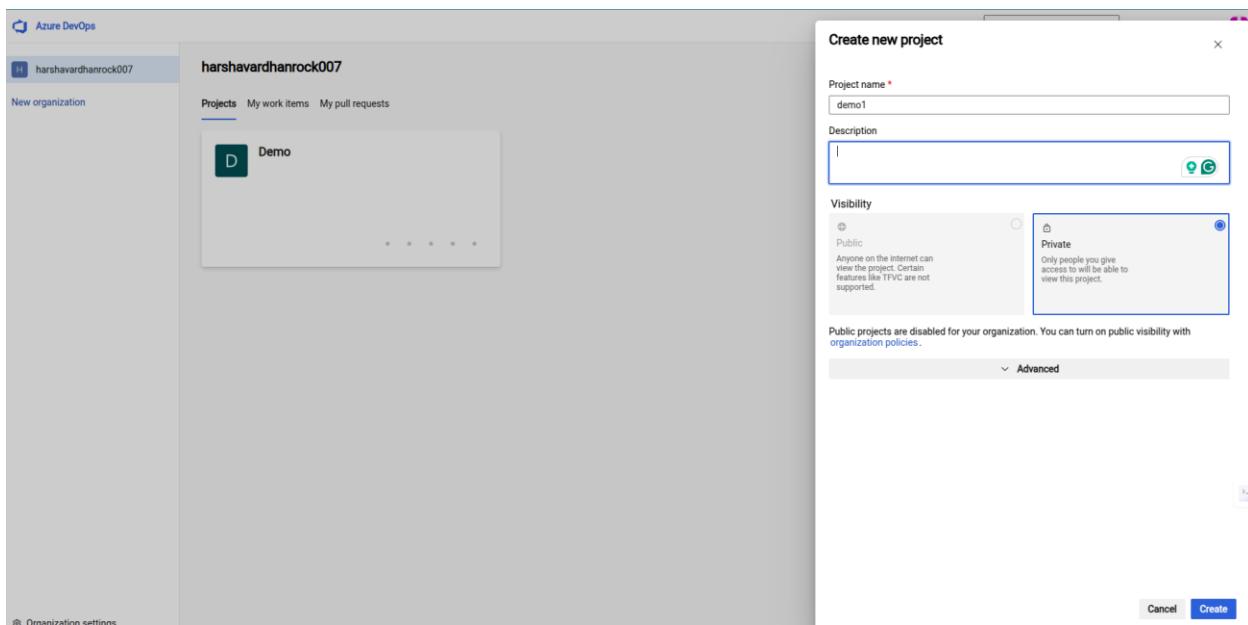
# Azure DevOps Project with Kubernetes & ArgoCD Integration

## 1. Azure Account Initialization

- Set up a valid **Azure Account** with an active subscription.
- Log into [Azure DevOps](#) using the **same credentials** as your Azure portal.

## 2. Create a New Azure DevOps Project

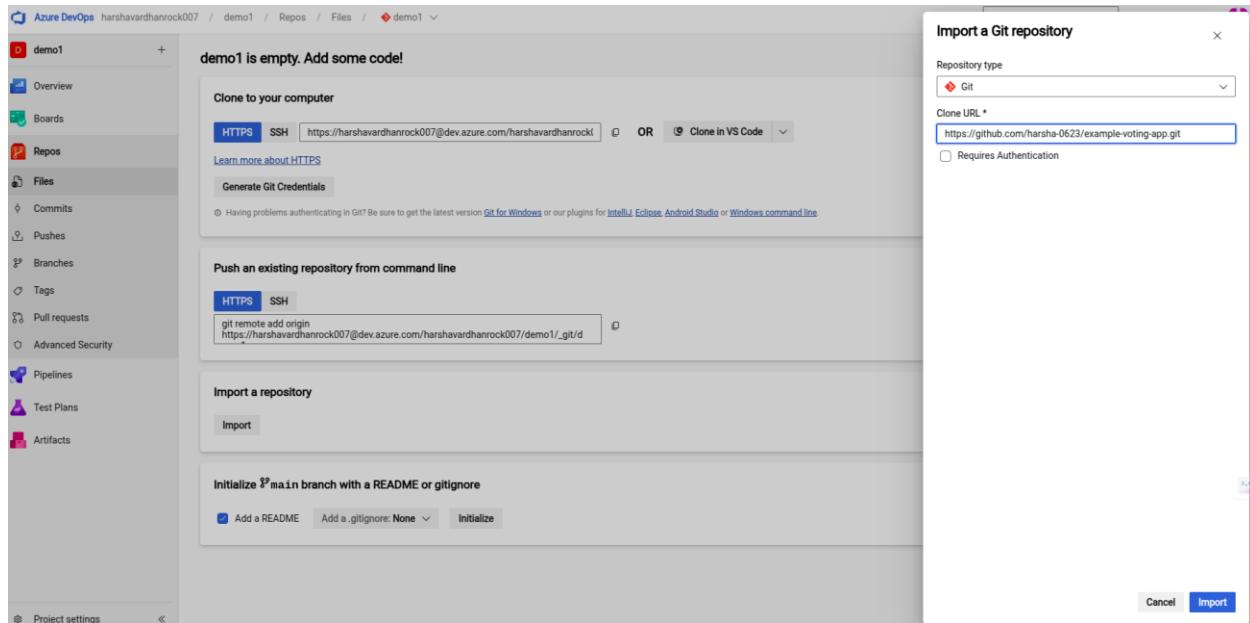
- Navigate to Azure DevOps Dashboard → Click on **New Project**.
- Name the project appropriately based on your objectives.



## 3. GitHub Repo Migration to Azure Repos

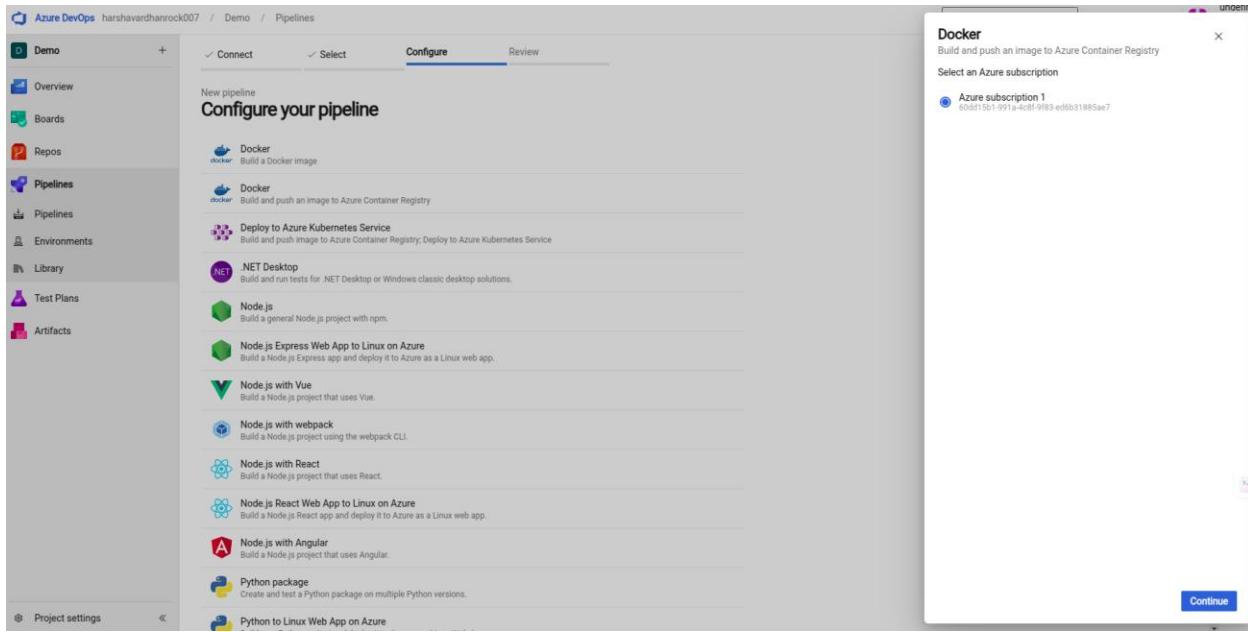
- Go to **Repos** → **Import Repository**.

- Paste your **GitHub HTTPS URL** to import all code into Azure Repos.



## 🔧 4. Build & Push Pipeline Configuration

1. Navigate to **Pipelines** → **New Pipeline**.
2. Select **Azure Repos Git** as the source.
3. Choose your required repository.
4. Under **Configure Pipeline**, select:
  - a. **Docker - Build and push an image to Azure Container Registry**.
5. Choose the **correct Azure subscription**.



## 5. Create Azure Container Registry (ACR)

- Go to [Azure Portal](#) → Search **Container Registry** → Create a new instance.
- Group all project services in a **standard Resource Group** for manageability.

The screenshot shows the 'Container registries' blade in the Azure Portal. It lists one item:

Name	Type	Resource group	Location	Subscription
hanshacid	Container registry	azuredcid	East US	Azure subscription 1

## 6. YAML Pipeline Customization

- Modify the generated YAML to:
  - Replace default VM image with a **custom agent pool**.
  - Split the **build** and **push** stages.
  - Create separate pipelines for **worker**, **results**, and **vote** services.

The screenshot shows the Azure DevOps Pipelines interface for a project named 'Demo'. The pipeline is named 'vote' and is defined by the file 'azure-pipelines-vote.yml'. The code in the file is as follows:

```

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   dockerRegistryServiceConnection: '5c35154c-1e06-4bb2-9550-9395c909555f'
15   imageRepository: 'vote-app'
16   containerRegistry: 'harshacaid.azurecr.io'
17   dockerfilePath: '$(Build.SourcesDirectory)/result/Dockerfile'
18   tag: '$(Build.BuildId)'
19
20 pool:
21   name: 'harshaagent'
22
23 stages:
24   - stage: Build
25     displayName: Build
26     jobs:
27       - job: Build
28         displayName: Build
29
30       - steps:
31         - task: Docker@2
32           displayName: Build an image to container registry
33           inputs:
34             containerRegistry: '$(dockerRegistryServiceConnection)'
35             repository: '$(imageRepository)'
36             command: 'build'
37

```

On the right side of the interface, there is a 'Tasks' pane listing various build and deployment tasks, such as .NET Core, Android signing, Ant, App Center distribute, App Center test, Archive files, ARM template deployment, Azure App Configuration Export, Azure App Configuration Import, Azure App Configuration Snapshot, Azure App Service deploy, and Azure App Service manage.

The screenshot shows the Azure DevOps Pipelines interface for a project named 'Demo'. The pipeline is named 'results' and is defined by the file 'azure-pipelines-result.yml'. The code in the file is as follows:

```

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       - result/*
9
10 resources:
11   - repo: self
12
13 variables:
14   dockerRegistryServiceConnection: '7afe3083-49cc-4933-b342-264f921141b9'
15   imageRepository: 'result-app'
16   containerRegistry: 'harshacaid.azurecr.io'
17   dockerfilePath: '$(Build.SourcesDirectory)/result/Dockerfile'
18   tag: '$(Build.BuildId)'
19
20 # Agent VM image name
21 vmImageName: 'ubuntu-latest'
22
23 pool:
24   name: 'harshaagent'
25
26 stages:
27   - stage: Build
28     displayName: Build and push stage
29     jobs:
30       - job: Build
31         displayName: Build
32
33       - pool:
34         - steps:
35           - task: Docker@2
36             displayName: Build and push an image to container registry
37             inputs:
38               command: 'buildAndPush'
39               repository: '$(imageRepository)'


```

## 7. Setup Virtual Machine for Self-hosted Agent

- Create a VM and install:
  - Docker
  - Kubernetes CLI (kubectl)

- Go to **Project Settings > Agent Pools** → Create a **New Pool** (mark as **Self-hosted**).

The screenshot shows two windows side-by-side. On the left is the 'Virtual machines' blade, displaying a single VM named 'harshaagent' in the 'Azure subscription 1' resource group. On the right is the 'Agent pools' blade under 'Project Settings'. A modal window titled 'Add agent pool' is open, showing the configuration for a new self-hosted pool. The 'Name' field is set to 'test', and the 'Pool type' is selected as 'Self-hosted'. Other options like 'Managed DevOps Pool' and 'Azure virtual machine scale set' are also shown. At the bottom right of the modal is a blue 'Create' button.

## 8. Generate Personal Access Token (PAT)

- Go to **User Settings > Personal Access Tokens** → Generate a new token with relevant scopes.

The screenshot shows the Azure DevOps User settings page. On the left, there's a sidebar with options like Account, Preferences, and Security. Under Security, 'Personal access tokens' is selected. The main area shows a list of existing tokens: 'access-token' (Full access) and 'Git: https://dev.azure.com/harshavardhanrock007 on the website.' (Code (Read & write)). A modal window titled 'Create a new personal access token' is open on the right, prompting for a Name, Organization (set to 'All accessible organizations'), Expiration (set to '30 days' with a date of '09/05/2025'), Scopes (set to 'Custom defined'), and various permission levels for Work items, Code, Build, Release, and Test Management.

## 9. Connect VM to Azure DevOps as Agent

1. From **Agent Pools**, click on **New Agent**.
2. Follow the commands provided:

```
./config.sh # Enter server URL and PAT
./run.sh # Run agent in active state
```

The screenshot shows the 'Project Settings' page for a 'Demo' project. The 'Agent pools' section is selected. A modal window titled 'Get the agent' is open, specifically for a 'Linux' agent. It shows a list of existing agents (Job 10, Job 9, Job 8, Job 7, Job 6, Job 5, Job 4, Job 3, Job 2, Job 1) and a 'Create the agent' section with configuration steps: 'Configure your account' (using curl command), 'Download the agent' (with a 'Download' button), 'Create the agent' (using tar command), 'Configure the agent' (using ./config.sh), and 'Optionally run the agent interactively' (using ./run.sh). To the right, a table lists recent agent jobs with columns for Start time, Wait time, and Duration.

```

Enter personal access token >
Enter personal access token > Exiting...
azureuser@harshaagent:~/nyagent$ ./config.sh

agent v4.253.0
                (commit 5263f72)

-> 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/nyagent/license.html

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

-> Connect:

Enter server URL > https://dev.azure.com/harshavardhanrock007
Enter authentication type (press enter for PAT) > 71lBqZM4gyq6DVF2SCVxqTTzyMdFPEOXv2IAgKDMnzUsyB7mjXN3QJ99BDACAAAAAAAASAZD01u0V
Enter a valid value for authentication type.
Enter authentication type (press enter for PAT) > 71lBqZM4gyq6DVF2SCVxqTTzyMdFPEOXv2IAgKDMnzUsyB7mjXN3QJ99BDACAAAAAAAASAZD01u0V
Enter a valid value for authentication type.
Enter authentication type (press enter for PAT) >
Enter personal access token > *****
Connecting to server ...

-> Register Agent:

Enter agent pool (press enter for default) > harshaagent
Enter agent name (press enter for harshaagent) >
Scanning for tool capabilities.
Connecting to tools service...
Successfully added the agent
Testing agent connection...
Enter work folder (press enter for _work) >
2025-04-09 07:32:19Z: Settings Saved.
azureuser@harshaagent:~/nyagent$ ./config.sh

```

The screenshot shows the Azure DevOps interface for pipeline configuration. On the left, the navigation bar includes 'Overview', 'Boards', 'Repos', 'Pipelines' (selected), 'Environments', 'Library', 'Test Plans', and 'Artifacts'. The main area displays the 'worker' pipeline configuration.

**Pipeline Configuration:**

- YAML File:** Demo / 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   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: '63087ad1-b001-4d77-8dd4-aa94d5e1fab2'
16   imageRepository: 'worker-app'
17   containerRegistry: 'harshacid.azurecr.io'
18   dockerfilePath: '${Build.SourcesDirectory}/result/Dockerfile'
19   tag: '${Build.BuildId}'
20
21 pool:
22   name: 'harshaagent'
23
24 stages:
25   - stage: Build
26     displayName: Build
27     jobs:
28       - job: Build
29         displayName: Build
30       - steps:
31           - task: Docker@2
32             displayName: Build an image to container registry
33             inputs:
34               dockerRegistryServiceConnection: '$(dockerRegistryServiceConnection)'
35               repository: '$(imageRepository)'
36               command: 'build'
37               Dockerfile: 'worker/Dockerfile'
38
      
```
- Docker Task Configuration:**
  - Container Repository:** Container registry: \$(dockerRegistryServiceConnection), Container repository: \$(imageRepository)
  - Commands:** Command: build, Dockerfile: worker/Dockerfile
  - Build context:** \*\*
  - Tags:** \${tag}
  - Arguments:** (empty)

The screenshot shows the Azure DevOps Pipelines dashboard. The left sidebar includes 'Overview', 'Boards', 'Repos', 'Pipelines' (selected), 'Environments', 'Library', and 'Test Plans'.

**Recently run pipelines:**

Pipeline	Last run	Time
worker	#20250409.4 • Updated Dockerfile	30m ago 1m 56s
vote	#20250409.3 • Update azure-pipelines-vote.yml for Azure Pipelines	44m ago 1m 44s
results	#20250409.3 • Update azure-pipelines-result.yml for Azure Pipelines	2h ago 2m 14s

## 10. Create Kubernetes Cluster (AKS)

- In Azure Portal → Search **Kubernetes services** → Create a new AKS cluster.
- On the VM:

```
curl -sL https://aka.ms/InstallAzureCLIDeb | sudo bash
az login
az aks get-credentials --name azuredvops --resource-group azurecicd --overwrite-existing
sudo snap install kubectl --classic
kubectl get pods
```

## 11. Install & Configure ArgoCD

1. Check ArgoCD pods:

```
kubectl get pods -n argocd
```

2. Retrieve & decode initial admin password:

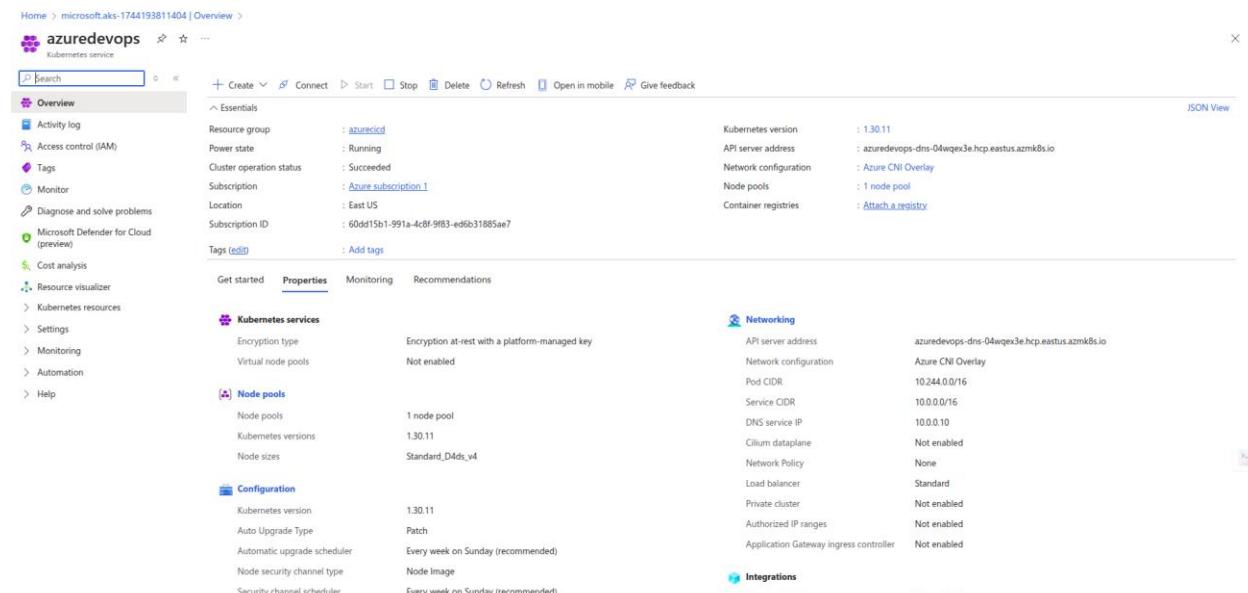
```
kubectl get secrets -n argocd
kubectl edit secret argocd-initial-admin-secret -n argocd
echo <base64-password> | base64 --decode
```

3. Edit NodePort:

```
kubectl edit svc argocd-server -n argocd
kubectl get nodes -o wide
```

4. Adjust config (like session timeout):

```
kubectl edit cm argocd-cm -n argocd
```



```
az user@harshaagent:~$ az login
To sign in, use a web browser to open the page https://microsoft.com/devicelogin and enter the code EHB4HRQSM to authenticate.

Retrieving tenants and subscriptions for the selection...
[Tenant and subscription selection]
No Subscription name Subscription ID Tenant
--- -----
[1] * Azure subscription 1 60dd15b1-991a-4c8f-9f83-ed6b31885ae7 Default Directory

The default is marked with an *; the default tenant is 'Default Directory' and subscription is 'Azure subscription 1' (60dd15b1-991a-4c8f-9f83-ed6b31885ae7).

Select a subscription and tenant (Type a number or Enter for no changes):
Tenant: Default Directory
Subscription: Azure subscription 1 (60dd15b1-991a-4c8f-9f83-ed6b31885ae7)

[Announcements]
With the new Azure CLI login experience, you can select the subscription you want to use more easily. Learn more about it and its configuration 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 user@harshaagent:~$ az aks get-credentials --name azuredevops --resource-group azurecid --overwrite-existing
merged "azuredevops" as current context in /home/azuser/.kube/config
az user@harshaagent:~$ kubectl get pods
Container runtime not found, but can be installed with:
sudo snap install kubectl
az user@harshaagent:~$ sudo snap install kubectl --classic
2025-04-09T10:39:42Z INFO Waiting for automatic snapd restart...
Kubectl 1.32.3 from canonical\ installed
az user@harshaagent:~$ kubectl get pods
No resources found in default namespace.
```

## 12. Create Docker Secret for K8s

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

- Verify using:

```
kubectl get pods -w
```

## 🌐 13. Access ArgoCD UI

- Allow the ArgoCD NodePort in **Inbound NSG Rules**.
- Access the ArgoCD UI at: <http://<external-ip>:<node-port>>

The screenshot shows the Microsoft Azure portal interface. On the left, there's a sidebar with navigation links like Overview, Connect, Networking, Settings, Monitoring, and Support + troubleshooting. The main area shows the 'aks-agentpool-80838772-vmss\_0 | Network settings' page. In the center, there's a table with network interface details: Network interface (aks-agentpool-80838772-vmss), Virtual network / subnet (aks-vnet-24480629 / aks-subnet), Public IP address (172.172.176.198), Private IP address (10.224.0.4), and Admin security rules (0). To the right, there's a 'Load balancers' section and a 'Network security groups' section. A 'Network security group aks-agentpool-24480629-nsg (attached to subnet: aks-subnet)' is listed, showing it impacts 1 subnets, 0 network interfaces. Below this, there's a table for 'Inbound port rules' with three entries: 65000 (AllowVnetInBound), 65001 (AllowAzureLoadBalancerInBound), and 65500 (DenyAllInBound). A modal window titled 'Add inbound security rule' is open on the right, showing fields for Source (Any), Destination (Any), Service (Custom), Destination port ranges (30461), Protocol (Any), Action (Allow), Priority (100), Name (AllowAnyCustom30461Inbound), and Description. At the bottom of the modal are 'Add' and 'Cancel' buttons.

## ❖ 14. Configure ArgoCD Application

- In ArgoCD UI → Go to **Settings > Repositories** → Add Git Repo.
- **Create Application:**
  - Git Repo URL
  - Revision/Path
  - Cluster + Namespace

Dec 3 22:18

Resume scan results Devops architect

**Hard skills HIGH SCORE IMPACT**

Hard skills enable you to perform job-specific duties and responsibilities. You can learn hard skills in the classroom, training courses, and on the job. These skills are typically focused on teachable tasks and measurable abilities such as the use of tools, equipment, or software. Hard skills have a high impact on your match score.

**Tip:** Match the skills in your resume to the exact spelling in the job description. Prioritize skills that appear most frequently in the job description.

Skills Comparison			Highlighted Skills
Skill	Resume	Job Description	
Devops	6	8 <span style="background-color: black; color: white;">Required</span>	
SaaS	X	1	
docker	3	1	
puppet	X	1	
Tooling	X	1	
automation	2	1	
	X	1	
	X	1	
	X	1	
analytical skills	1	1	

Show more

Feedback 

**Settings / Repositories** REPOSITORIES Log out

TYPE	NAME	PROJECT	REPOSITORY	CONNECTION STATUS
git	default		https://7lBqZMe4gyq6DVF2SCVxqTTZyMdFPEOXv2lAgKDMmzUsyB7mjXNJJQQJ99BDACAAAAAAAASAZD01u0V@dev.azure.com/harshavardhanrock007/Der	<span style="color: green;">Successful</span>

**Applications** + NEW APP CANCEL

**SOURCE**

Repository URL: <https://7lBqZMe4gyq6DVF2SCVxqTTZyMdFPEOXv2lAgKDMmzUsyB7mjXNJJQQJ99BDACAAAAAAAASAZD01u0V@dev.azure.com/harshavardhanrock007/Der> GIT ✓

Revision: HEAD

Path: k8s-specifications

**DESTINATION**

Cluster URL: <https://kubernetes.default.svc> URL ▾

Namespace: default

**Directory**

DIRECTORY

argos v2.14.6+fe2a6e9

Applications Settings User Info Documentation

Favorites Only SYNC STATUS Unknown Synced OutOfSync 0 0 1

Progressing 0 Suspended 0 Healthy 0 Degraded 0 Missing 1 Unknown 0

Labels

Projects

PROJECTS

The screenshot shows a GitHub repository interface. At the top, it displays the path: main / scripts / updatek8sManifests.sh. Below this, there's a navigation bar with links for Contents, History, Compare, and Blame. An 'Edit' button is located in the top right corner.

```

1 #!/bin/bash
2
3 set -x
4
5 # Set the repository URL
6 REPO_URL="https://711ba2Me4gyq6DVF25CVxqTTzyMdFPE0Xv21AgKDMmzUsyB7mjXNJQJ99BDACAAAAAAAASAZD01u0V@dev.azure.com/harshavardhanrock007/Demo/_git/Demo"
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: $|image: harshacid/$2:$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

```

Below the code editor, there's a section titled 'vote' with a 'Run' button. To the left of the run button, there's a 'Variables' tab. On the right, there's a 'Shell script' configuration panel with fields for 'Script Path' (scripts/updatek8sManifests.sh) and 'Arguments' (vote \${imageRepository} \${tag}).

## 15. Add Automation Scripts

- In your repo → Create folder scripts/
- Add necessary .sh files
- In YAML, add a new stage update to run scripts.

Azure DevOps harshavardhanrock007 / Demo / Pipelines

← vote

Validate and save

Validate and commit azure-pipelines-vote.yml to the repository.

Validation

Pipeline is valid.

Commit message

Update azure-pipelines-vote.yml for Azure Pipelines

Optional extended description

Add an optional description...

Commit directly to the main branch

Create a new branch for this commit

Cancel Save

```
43 jobs:
44   - job: Push
45     displayName: Push
46
47   steps:
48     - task: Docker@2
49       displayName: Push an image to container registry
50       inputs:
51         containerRegistry: '$(dockerRegistryServiceConnection)'
52         repository: '$imageRepository'
53         command: 'push'
54         tags: '${tag}'
55
56   stage: Update
57   displayName: Update
58   jobs:
59     - job: Update
60       displayName: Update
61
62     steps:
63       - task: ShellScript@2
64         inputs:
65           scriptPath: 'scripts/updatek8sManifests.sh'
66           args: 'vote ${imageRepository} ${tag}'
```

Azure DevOps harshavardhanrock007 / Demo / Pipelines

← vote

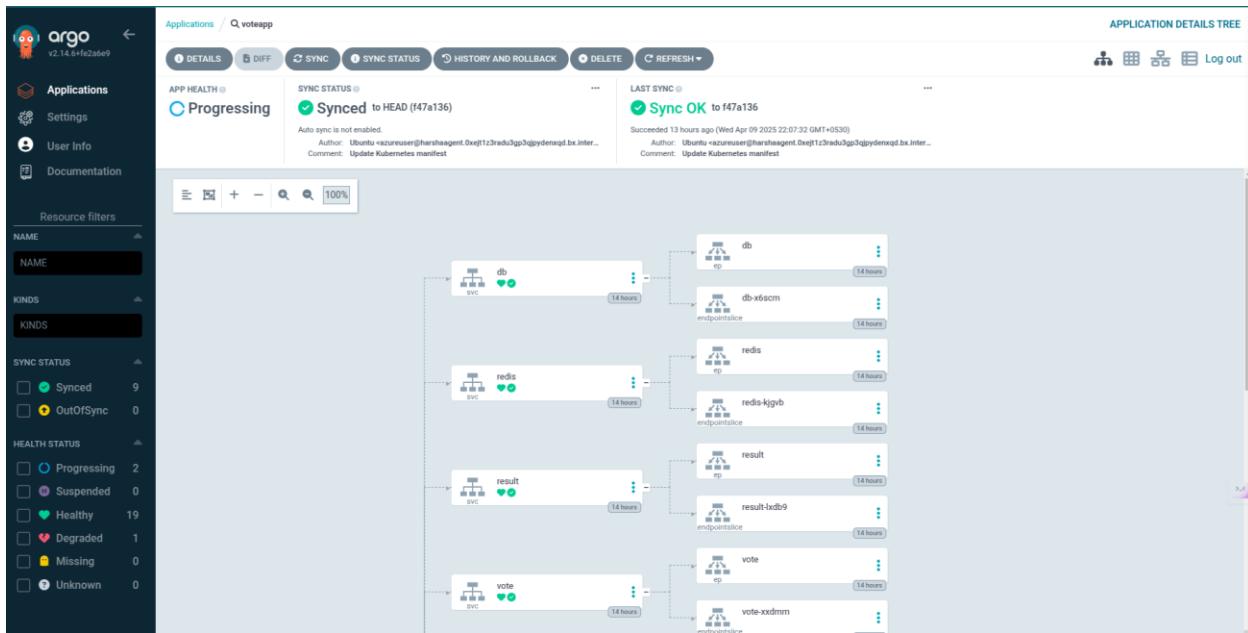
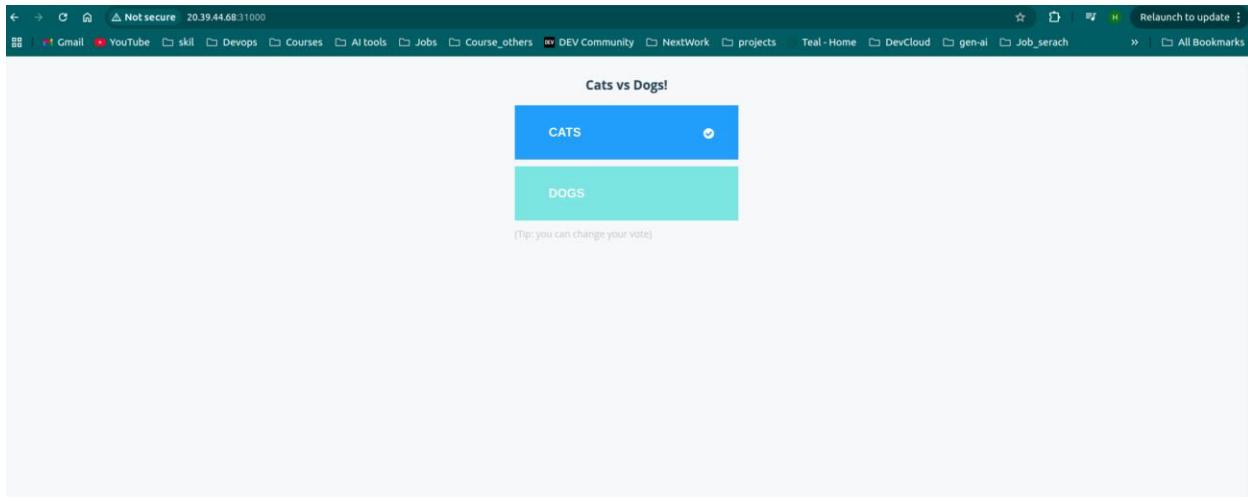
```
47   steps:
48     - task: Docker@2
49       displayName: Push an image to container registry
50       inputs:
51         containerRegistry: '$(dockerRegistryServiceConnection)'
52         repository: '$imageRepository'
53         command: 'push'
54         tags: '${tag}'
55
56   stage: Update
57   displayName: Update
58   jobs:
59     - job: Update
60       displayName: Update
61
62   steps:
63     - task: ShellScript@2
64       inputs:
65         scriptPath: 'scripts/updatek8sManifests.sh'
66         args: 'vote ${imageRepository} ${tag}'
```

The screenshot shows two windows side-by-side. The left window is the Microsoft Azure Container Registry interface for a registry named 'harshacidd'. It displays the 'Access keys' section where two sets of credentials are listed: 'password' and 'password2'. The 'password' field contains a long, complex string of characters. The right window is an Azure DevOps repository for a project named 'Demo'. It shows a file named 'vote-deployment.yaml' which contains YAML configuration for a Kubernetes deployment. The file specifies an image pull secret named 'az-secret'.

```
1 apiVersion: apps/v1
2 kind: Deployment
3 metadata:
4   labels:
5     name: vote
6   name: vote
7 spec:
8   replicas: 1
9   selector:
10    matchLabels:
11      app: vote
12   template:
13     metadata:
14       labels:
15         app: vote
16     spec:
17       containers:
18         image: harshacidd/vote-app:13
19         name: vote
20       ports:
21         - containerPort: 80
22       name: vote
23       imagePullSecrets:
24         - name: az-secret
25
```

## 16. Redeploy and Validate

- Redeploy the pods
- Access the ArgoCD UI and validate your deployment visually



## Visual Breakdown

- Include diagrams showing worker, vote, and results jobs.
- Screenshots of ArgoCD UI, pipeline execution, and VM agent.

## Final Thoughts

This setup enables a complete **CI/CD workflow** using **Azure DevOps**, **Kubernetes**, and **ArgoCD**. You now have a scalable, secure, and automated deployment pipeline with modern DevOps best practices.