Azure Migration Project

In this project we are going to migrate the on prem DevOps solution to Azure Cloud. This project holds the lowest budget architecture which uses minimal resources yet powerful and robust performance for a simple tier-3 dynamic portfolio website.

Process of Migration:

VM Details:

VM Size: Standard B2ms (2 vcpus, 8 GiB memory)
Operating system: Linux (ubuntu 24.04 LTS server)
Disk Size: 32 GiB
Storage Type: Premium SSD LRS

STEP -1 ---------- AZURE VM SSH LOGIN ----------

In this file, we are going to login to azure VM through SSH securely...

Pre-Requisites:
1) Azure VM (In running state)
2) Public IP enabled:
portal > Azure Virtual Machines > Your VM > Networking > Click on Network interface Ip/ Ip configuration (ipconfig1) > click on ipconfig1 > Check in this box: Public IP address settings > Click on Create a public IP address > Give name > Ok > Save > wait 2min your public IP will be shown in the VM.

Process:
In the login process we can follow two methods:

METHOD - 1: Temporary public IP allocation. We can create the public IP and assign to VM like stated above then after the logoff then we can delete the public IP to save the extra cost.
METHOD - 2: USING SSH Cloudflared tunnel:

a) First let's add our domain name in the azure: portal > Microsoft entra ID > Custom Domain Names > + Add Custom Domain name > give your domain name > In the cloudflared/whatever the domain hosted site, go to the site and in the dns records add the details shown in the azure portal as it is like alisa or hostname, message, ttl value: auto > Save.

b) At the time of VM creation a .pem file will be downloaded to your local device.

c) Login to the VM to setup cloudflared tunnel inside the VM:
ssh -i "C:\Users\CH Anil Kumar\Downloads\Personal-folio-VM.pem" vmadin@4.247.143.565 > type yes > enter

d) You are successfully logged in now, update the packages:
sudo apt update

e) adding the package url to the Linux kernel:
curl -fsSL https://pkg.cloudflare.com/cloudflare-main.gpg | sudo tee /usr/share/keyrings/cloudflare-main.gpg >/dev/null

f) Adding the gpg key:
echo "deb [signed-by=/usr/share/keyrings/cloudflare-main.gpg] https://pkg.cloudflare.com/cloudflared $(lsb_release -cs) main" | sudo tee /etc/apt/sources.list.d/cloudflared.list
deb [signed-by=/usr/share/keyrings/cloudflare-main.gpg] https://pkg.cloudflare.com/cloudflared noble main

g) again update the packages:
sudo apt update

h) install the cloudflared:
sudo apt install cloudflared -y

r) Check version:
cloudflared –version

j) login to the cloudflared:
cloudflared tunnel login

Output:
Please open the following URL and log in with your Cloudflare account:

https://dash.cloudflare.com/argotunnel?aud=&callback=https%3A%2F%2Flogin.cloudflareaccess.org%2FfgCi8JjQiTQQbNYD4nu42dDm1hA2K-nAZYHqBMeCWzo%3D – copy this url and paste it in the chrome/edge browser of the local computer > select the domain > Authorize.

Leave cloudflared running to download the cert automatically.
2025-12-29T16:44:42Z INF You have successfully logged in.
If you wish to copy your credentials to a server, they have been saved to:
/home/vmadin/.cloudflared/cert.pem

k) create cloudflared tunnel:
cloudflared tunnel create ARKS-Global-Asia-VM-ssh-tunnel

l) list the tunnels:

cloudflared tunnel list – to list tunnels

sudo ls /home/vmadin/.cloudflared/ - to see if the credentials file is there or not

m) create a cloudflared dir in the /etc/ folder of the VM

sudo mkdir -p /etc/cloudflared

n) Copy the credentials.json file to the /etc/cloudflared dir

sudo cp /home/vmadin/.cloudflared/15842aed-e191-4346-9sdfd-40fafbd956c3.json /etc/cloudflared/

o) Create and edit the config.yaml:

sudo vi /etc/cloudflared/config.yaml

|_ Add the content:

tunnel: <tunnel ID>

credentials-file: /etc/cloudflared/<tunnel ID>.json

ingress:

- hostname: <give your url>

service: ssh://localhost:22

- service: http_status:404

Click esc > :wq > enter > It will save the file.

p) Change the owner of the folder cloudflared:

sudo chown root:root /etc/cloudflared/*

q) set the permissions for cloudflared folder:

sudo chmod 700 /etc/cloudflared/* (if you give 600 instead of 700 then you only able to rw and it is purely for root so use sudo to edit the file. To see what permission there use this command: **ls -l /etc/cloudflared**)

r) to edit the service file of the cloudflare-ssh tunnel:

sudo vi /etc/systemd/system/cloudflared-ssh.service

|_ Paste the following script:

[Unit]
Description=Cloudflare Tunnel (SSH)
After=network-online.target
Wants=network-online.target

[Service]
Type=simple
ExecStart=/usr/bin/cloudflared tunnel --config /etc/cloudflared/config.yaml run
Restart=always
RestartSec=5
User=root

[Install]
WantedBy=multi-user.target

Esc > :wq > enter > Save and Exit

To install cloudflared:
Sudo systemctl install cloudflared

sample output:
2025-12-29T17:18:26Z INF Using Systemd
2025-12-29T17:18:30Z INF Linux service for cloudflared installed successfully

s. To update the daemon:
sudo systemctl daemon-reexec

t) To reload the daemon:
sudo systemctl daemon-reload

u) To start the ssh tunnel service:
sudo systemctl start cloudflared-ssh

v) To enable ssh tunnel service on boot:
sudo systemctl enable cloudflared-ssh

w) To know the status:
sudo systemctl status cloudflared-ssh

sample output:

● cloudflared.service - cloudflared
Loaded: loaded (/etc/systemd/system/cloudflared.service; enabled; preset: enabled)
Active: active (running) since Mon 2025-12-29 17:18:30 UTC; 34s ago
Main PID: 2887 (cloudflared)
Tasks: 7 (limit: 449)
Memory: 32.2M (peak: 32.4M)
CPU: 356ms
CGroup: /system.slice/cloudflared.service
└─2887 /usr/bin/cloudflared --no-autoupdate --config /etc/cloudflared/config.yml tunnel run.

v) After running all the commands successfully, open another terminal, login to the vm using tunnel:
ssh -i "C:\Users\CH Anil Kumar\Downloads\VM-pub-key.pem" -o ProxyCommand="cloudflared access ssh --hostname %h" vmadin@<your URL> - in this we need to use the private key which was downloaded at the time of the VM Creation.

> this is will allow you to login to the VM no need of public IP now.

Note: Instead of using this big command **ssh -i "C:\Users\CH Anil Kumar\Downloads\VM-pub-key.pem" -o ProxyCommand="cloudflared access ssh --hostname %h" vmadin@<your url>**,
we can simply create a file in our local PC under the path:
C:\Users\CH Anil Kumar\.ssh\config

> add the following content to it:
Host <give a name ex: portfoio-vm>
HostName <your URL> <Domain name>
User vmadin <azure vm user>
IdentityFile C:/Users/CH Anil Kumar/Downloads/portlio-vm.pem <private key downloaded file path>
ProxyCommand cloudflared access ssh --hostname %h <Command used for login>

> Then save it after that login with this command:
ssh <name of the host. ex: portfoio-vm>

3) Afer the tunnel creation delete the public IP:

portal > Azure Virtual Machines > Your VM > Networking > Click on Network interface Ip/ Ip configuration (ipconfig1) > click on ipconfig1 > Check out this box: Public IP address settings > save

> When we get any error like this:

C:\Users\CH Anil Kumar>ssh AR-VM
@@@
@   WARNING: REMOTE HOST IDENTIFICATION HAS CHANGED!   @
@@@
IT IS POSSIBLE THAT SOMEONE IS DOING SOMETHING NASTY!
Someone could be eavesdropping on you right now (man-in-the-middle attack)!
It is also possible that a host key has just been changed.
The fingerprint for the ED25519 key sent by the remote host is
SHA256:lsDSfwqFEIHIeltjNz5pJsNw/4va+ZcnfeVKLQM.
Please contact your system administrator.
Add correct host key in C:\\Users\\CH Anil Kumar/.ssh/known_hosts to get rid of this message.
Offending ED25519 key in C:\\Users\\CH Anil Kumar/.ssh/known_hosts:7
Host key for <your url >has changed and you have requested strict checking.
Host key verification failed.

FIX:

Open config file:
notepad C:\Users\CH Anil Kumar\.ssh\known_hosts - remove the existing line like this:
<your url >ssh-ed25519
AAAAC3NzaC1lZDI1NTE5AAAAIAjgRMXsfsdfpOMMkyrPpEk3gst0PUa+KsFR9aoDywaI
> then try again, it will work!

STEP -2 ------------------------------ Servers tunnel Creation ----------------------------------

a) Create the tunnel for apps:
   Cloudflared tunnel create <name of the tunnel>
b) Copy the credentials file (.json) file from .cloudflared dir to /etc/cloudflared dir
   Sudo cp /home/vmadin/.cloudflared/<tunnel ID>.json /etc/cloudflared/
c) Create a config file in /etc/cloudflared/
   Sudo vi /etc/cloudflared/apps-config.yaml
   |_ Add the following content:

tunnel: <tunnel ID>
credentials-file: /etc/cloudflared/<tunnel ID>.json

ingress:
- hostname: <Jenkins url>
service: ssh://localhost:8080
- hostname: <sonarqube url>
service: ssh://localhost:9000
- hostname: <argocd url > ex: argocd.xyz.com
service: ssh://localhost:8081

- service: http_status:404

Esc > :wq > Enter > save and exit

d) Change the owner of the folder cloudflared:
sudo chown root:root /etc/cloudflared/*

e) set the permissions for cloudflared folder:
sudo chmod 700 /etc/cloudflared/* (if you give 600 instead of 700 then you only able to rw and it is purely for root so use sudo to edit the file. To see what permission there use this command: **ls -l /etc/cloudflared**)

f) to edit the service file of the cloudflare-ssh tunnel:
sudo vi /etc/systemd/system/cloudflared-apps.service
|_ Paste the following script:

```
[Unit]
Description=Cloudflare Tunnel (SSH)
After=network-online.target
Wants=network-online.target

[Service]
Type=simple
ExecStart=/usr/bin/cloudflared tunnel --config /etc/cloudflared/apps-config.yaml run
Restart=always
RestartSec=5
User=root

[Install]
```

WantedBy=multi-user.target

Esc > :wq > enter > Save and Exit

g) To update the daemon:
sudo systemctl daemon-reexec
h) To reload the daemon:
sudo systemctl daemon-reload
i) To start the ssh tunnel service:
sudo systemctl start cloudflared-apps
j) To enable ssh tunnel service on boot:
sudo systemctl enable cloudflared-apps
k) To know the status:
sudo systemctl status cloudflared-apps

Now you can ssh to the VM using simple cmd: **ssh <name of the host>** and use the servers like Jenkins, sonarqube, argocd… using <Jenkins URL> and connect.

Step – 3---------- To install Docker ----------
We can install the docker using:

a) To update the linux repository:
sudo apt update
b) To upgrade the services if any need upgrade:
sudo apt upgrade -y
c) To install Docker:
curl -fsSL https://get.docker.com | sudo sh
d) To enable Docker:
sudo systemctl enable docker
e) To start Docker:
sudo systemctl start docker
f) To add the user:
sudo usermod -aG docker $USER
g) To Check the version:
docker version

Step -4 ---------- To install K3S (light weight) ----------
k3s = **lightweight Kubernetes**, perfect for our budget.

a) To install K3S
   curl -sfL https://get.k3s.io | sh -
b) To check the nodes as sudo
   sudo kubectl get nodes
c) Configure Kubectl for normal user:
   1) **mkdir -p ~/.kube**
   2) **sudo cp /etc/rancher/k3s/k3s.yaml ~/.kube/config**
   3) **sudo chown $USER:$USER ~/.kube/config**
   4) **export KUBECONFIG=~/.kube/config**
   5) **export KUBECONFIG=$HOME/.kube/config**
   6) **echo 'export KUBECONFIG=$HOME/.kube/config' >> ~/.bashrc**
   7) **source ~/.bashrc**
d) To verify pods as normal user:
   kubectl get pods -A

Step -5 ---------- Configuring Jenkins in Docker----------

a) To create a volume in docker named Jenkins_home
   docker volume create jenkins_home
b) To Run the Jenkins image with root access

```
docker run -d \
  --name jenkins \
  --user root \
  -p 8080:8080 \
  -p 50000:50000 \
  -v jenkins_home:/var/jenkins_home \
  -v /var/run/docker.sock:/var/run/docker.sock \
  --restart unless-stopped \
  jenkins/jenkins:lts
```

c) To enter into Jenkins container:
   docker exec -it -u 0 jenkins bash
   |_ To install docker cli inside the container:
   | **docker version || apt update && apt install -y docker.io**
   |_ To install apt install -y libatomic1
   | **apt install -y libatomic1**
   |_ To add Jenkins user to Docker group
   | **usermod -aG docker Jenkins**

|_ To install ArgoCD inside container:

a) **curl -sSL -o argocd https://github.com/argoproj/argo-cd/releases/latest/download/argocd-linux-amd64**
b) **install -m 555 argocd /usr/local/bin/argocd**
c) **rm argocd**
d) **argocd version --client**

|_ To install Kubectl inside the container:

a) **apt update**
b) **apt install -y curl ca-certificates**
c) **curl -LO "https://dl.k8s.io/release/$(curl -Ls https://dl.k8s.io/release/stable.txt)/bin/linux/amd64/kubectl"**
d) **install -o root -g root -m 0755 kubectl /usr/local/bin/kubectl**

d. initial login:

|_ To get initial admin password:
| **docker exec jenkins cat /var/jenkins_home/secrets/initialAdminPassword**
|_ Login with <Jenkins url>(since we created the cloudflared tunnel)
| |_ Copy paste the initial admin Password
|_ proceed with the initial setup
|_ Install Additional Plugins: Manage Jenkins > Plugins > Available plugins
|_ Generic Webhook Trigger Plugin, Git client plugin, Git plugin, Git server Plugin, GitHub API Plugin, GitHub Authentication plugin, GitHub Branch Source Plugin, GitHub Integration Plugin, GitHub plugin, Job and Stage monitoring Plugin, Pipeline: GitHub, Pipeline: GitHub Groovy Libraries, Quality Gates Plugin, Sonar Quality Gates Plugin, Sonargraph Integration Jenkins Plugin, SonarQube Scanner for Jenkins, Docker API Plugin, Docker Commons Plugin, Docker Pipeline, Docker plugin, docker-build-step, Kubernetes :: Pipeline :: DevOps Steps, Kubernetes CLI Plugin, Kubernetes Client API Plugin, Kubernetes Credentials Plugin, Kubernetes Credentials Provider, Kubernetes plugin, Job and Stage monitoring Plugin, Pipeline Aggregator View, Pipeline: Stage Step, Pipeline: Stage View Plugin, NodeJS Plugin, Role-based Authorization Strategy
|
|_ Configure the Global tools: manage Jenkins > tools
| |_ Git installations:
| Give a name
| path to git executable: git
| Check in install automatically.
| |_SonarQube Scanner installations:
| | Name: sonar-scanner
| | Check in install automatically.
| |_ NodeJS installations (Since our application is based on NodeJS and nextJS)
| | Name: nodeJS 18
| | Check in install automatically
|

|_ Global System Configuration:

|_ SonarQube servers

Name: sonarqube

server URL: <Sonarqube url>

Credentials: in sonarqube go to My account > Security > Enter token name > select type: Project Analysis token > Set expiration > Generate.

|_ Copy that token and come to Jenkins : manage Jenkins > Credentials > Add credentials > Secret text > Paste the token in secret field > Give ID and Description > Add

|_ Add K8S cloud: Manage Jenkins > Clouds > Add new cloud:

|_ Give name > select Kubernetes > Create

|_ Kubernetes URL: https://<IP>:<Port> (it is VM private URL > get it using **ip a**)

|_ Kubernetes Namespace: anil-folio

|_ credentials: Add > Jenkins > select secret file (Here we need to choose kubeconfig file)

|_ Give ID and Description > Add

|_ Check in WebSocket

- Test connection.

Generate a KUBECONFIG file:

|_ **kubectl create sa jenkins -n anil-folio** – to create sa

|_ **kubectl create rolebinding jenkins-admin --clusterrole=admin --serviceaccount=anil-folio:jenkins -n anil-folio –** To create a rolebinding

|_ **kubectl create token jenkins -n anil-folio --duration=1000000h** – To create token

|_ **kubectl config view –raw** - To get raw info about k8s cluster

|_ **sudo mkdir /home/vmadin/kubeconfig/** - To create kubeconfig dir

|_ **sudo vi /home/vmadin/kubeconfig/kubeconfig** – To open new editor by creating file

|_ Add the following content:

```
apiVersion: v1
kind: Config
clusters:
- cluster:
    insecure-skip-tls-verify: true
    server: https:<private IP>:port
  name: my-cluster
contexts:
- context:
    cluster: my-cluster
    namespace: anil-folio
    user: jenkins
  name: jenkins-context
current-context: jenkins-context
users:
- name: jenkins
```

user:

token:

eyJhbGciOiJSUzI1NiIsImtpZCI6IlpnNjIxT0FUeV85c0RqdWRHN05YSlowTGxncXdTX21vYj
lQMkVISERQUncifQ.eyJhdWQiOlsiaHR0cHM6Ly9rdWJlcm5ldGVzLmRlZmF1bHQuc3ZjL
mNsdXN0ZXIubG9jYWwiLCJrM3MiXSwiZXhwIjo1MzY3MTg2NTE0LCJpYXQiOjE3Njcx
ODY1MTQsImlzcyI6Imh0dHBzOi8va3ViZXJuZXRlcy5kZWZhdWx0LnN2Yy5jbHVzdGVyL
mxvY2FsIiwianRpIjoiOGE0NhMGI0YjItMDZmOC00NDc1LWIzODMtZTg3MGNkNDk5Y
WZiIn19LCJuYmYiOjE3NjcxODY1MTQsInN1YiI6InN5c3RlbTpzZXJ2aWNlYWNjb3VudD
phbmlsLXBvcnRmb2xpbzpqZW5raW5zIn0.S9lev6AbU7VGN8ge_qFwmpHWJxLFtYFeBuaM
UeKpIE7MCklC9yz5knfGn6IlhL7j2wFYU0VYRWOLUeVYYr0t2jE5DZByEbRLAgbLqwZpL
_2lKyDKq7hoRVSeMuCWWj7Z31k4d6ygQcj3ChXmSM02i3Q6qDx2ptoTJCjK_QFxuqmxo-
H_B19KcZcnvoiEix17Ewe-
EmVOqDhSbjnXKfaMXeXN9uoWBIb_Qn6UTOrmTurBR3UIextLPaZ9BwgnND4b2QmW
OOHajjLBE-ZLkYmmLSRQj-
R8Qmpv9_zUBr2O9Hm2cqJdPi21hZNgV_V7eJkmysQfsur1u9fsLFOZ_O3MCA

save the kubeconfig file.

- Here this is in VM we created this kubeconfig file but we are accessing the Jenkins GUI at <Jenkins URL >from our local computer… to copy kubeconfig file from VM to local computer

  |_ scp -i "C:\Users\CH Anil Kumar\Downloads\personal-folio-VM_key.pem" vmadin@<Public IP>:/home/vmadin/kubeconfig/kubeconfig "C:/Users/CH Anil Kumar/Desktop/Anil-personal/kubeconfig-file/"
- The authenticity of host <pub IP>(<Pub IP>)' can't be established.
- ED25519 key fingerprint is SHA256:lsDSfwqFEIHIeltjNzWV4O5pJsNw/4va+ZcnfeVKLQM.
  This host key is known by the following other names/addresses:
  C:\Users\CH Anil Kumar/.ssh/known_hosts:6: 12.124.64.235
  C:\Users\CH Anil Kumar/.ssh/known_hosts:9: dfsas
  C:\Users\CH Anil Kumar/.ssh/known_hosts:10: 325.546.235.12
  C:\Users\CH Anil Kumar/.ssh/known_hosts:11: 123.10.90.123
  Are you sure you want to continue connecting (yes/no/[fingerprint])?: **type yes**

Credentials to create:

1) GitHub Credentials:
   |_ github > Developer settings > personal access tokens (Classic) > Generate new token (classic) > Give name > Expiration > give permission to repo, write:packages > Generate token > Copy it> Go to Jenkins > Manage Jenkins > Add Credentials > User Name with password > give GitHub user name > password: paste the PAT token > give ID & description > create.

2) Docker Credentials:
   |_ Go to Jenkins > Manage Jenkins > Add Credentials > User Name with Password > Give Docker User name and password > give ID & description > Create
3) K8S credentials:
   |_ K8S kubeconfig file creation is already described below
4) Argocd Credentials:
   |_ Go to Jenkins > Manage Jenkins > Add Credentials > User Name with Password > Give Docker User name and password > give ID & description > Create
5) SonarQube Credentials:
   |_ SonarQube > My Account > Security > Enter Token Name > Type: project Analysis Token > set expiration > Generate > Jenkins > manage Jenkins > Credentials > Add Credentials > Secret text > paste the token in secret field > Give ID & Description > Add

Important Note:

- Whatever the credential ID's you are giving throughout the Jenkins configuration all the credential IDs and Tools and system configuration should match else will face the errors while pipeline execution.

Step -6 ---------- Configuring SonarQube in Docker ----------

a) To setup volume in docker
   |_ **docker volume create sonarqube_data**

b) To run sonarqube as container
   |_ **docker run -d **
   **--name sonarqube **
   **-p 9000:9000 **
   **-v sonarqube_data:/opt/sonarqube/data **
   **--restart unless-stopped **
   sonarqube:community

- **Access the sonarqube at** <Sonar qube URL>
- Initial credentilas – user – admin & password – admin

Step-7 ---------- Installing Argo CD in K3S cluster ----------

a) Create a namespace
   |_**kubectl create namespace argocd**

b) Configuring ArgoCD:

|_**kubectl apply -n argocd **

-f https://raw.githubusercontent.com/argoproj/argo-cd/stable/manifests/install.yaml

c) To get pods :

|_ **kubectl get pods -n argocd**

d) To patch the argocd-server service:

```
kubectl patch svc argocd-server -n argocd -p '{
  "spec": {
    "type": "NodePort",
    "ports": [
      {
        "name": "http",
        "port": 80,
        "targetPort": 3450,
        "nodePort": 30080
      }
    ]
  }
}'
```

➢ It will allow us to directly connect to argocd instead of port-forwarding.

e. To get the initial password for argocd login:

|_ **kubectl get secret argocd-initial-admin-secret **

-n argocd -o jsonpath="{.data.password}" | base64 -d

f) Since we are using cloudflared tunnel we need to change the config map:

|_ **kubectl edit configmap argocd-cmd-params-cm -n argocd**

|_ Add following:

|_ data:

server.insecure: "true"

g) To restart argocd-server:

|_**kubectl rollout restart deployment argocd-server -n argocd**

h) To edit the tunnel configuration to add the argocd service point:

|_ **sudo vi /etc/cloudflared/apps-config.yaml**

- hostname: <Argocd URL>

service: https://<private iP>:<port> (private Ip and port)

i) To get the permission to generate API token in argocd:

|_ **kubectl edit configmap argocd-cm -n argocd**

|_ Add the content:

|_ **data:**

accounts.admin: apiKey, login

- This will avoid this error:
  This page isn't working <Argocd URL> redirected you too many times. Try deleting your cookies. ERR_TOO_MANY_REDIRECTS
- Why this error happens:
  Browser
  ↓ HTTPS
  Cloudflare
  ↓ HTTP
  cloudflared
  ↓ HTTP
  ArgoCD (forces HTTPS again ❌)

- To Install ArgoCD CLI and Add repos:
  |_ **curl -sSL -o argocd https://github.com/argoproj/argo-cd/releases/latest/download/argocd-linux-amd64**
  |_ Make it executable
  chmod +x argocd
  |_ Move it to path
  |_ sudo mv argocd /usr/local/bin/
  |_ To check the ArgoCD version:
  |_ argocd version –client
  |_ To login ArgoCD
  |_ argocd login <Argocd URL> --grpc-web –insecure
  |_ User: Admin
  |_ Password: <Give password>
  |_ To generate ssh
  |_ ssh-keygen (The ssh pub key is in /home/vmadin/.ssh/id_ed25519**)**

  |_ To get the content of the ssh key
  |_ cat /home/vmadin/.ssh/id_ed25519.pub
  |_ To add the repository
  |_ argocd repo add <config repo url>--ssh-private-key-path "/home/vmadin/.ssh/id_ed25519"
  |_ argocd repo add <values repo url>git --ssh-private-key-path "/home/vmadin/.ssh/id_ed25519"

  |_ TO Create applications in ArgoCD:
  |_ Create these two role and rolebinding:
  1) cat <<EOF > jenkins-argocd-role.yaml
  apiVersion: rbac.authorization.k8s.io/v1
  kind: ClusterRole
  metadata:

```
  name: jenkins-argocd-app-manager
rules:
  - apiGroups: ["argoproj.io"]
    resources: ["applications"]
    verbs: ["get", "list", "watch", "create", "update", "patch", "delete"]
EOF

2) cat <<EOF > jenkins-argocd-binding.yaml
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRoleBinding
metadata:
  name: jenkins-argocd-app-manager-binding
subjects:
  - kind: ServiceAccount
    name: jenkins
    namespace: anil-folio
roleRef:
  kind: ClusterRole
  name: jenkins-argocd-app-manager
  apiGroup: rbac.authorization.k8s.io
EOF
```

|_ Apply the above files:

kubectl apply -f jenkins-argocd-role.yaml

kubectl apply -f jenkins-argocd-binding.yaml

|_ Check if we can create apps:

**kubectl auth can-i create applications.argoproj.io **

**--as system:serviceaccount:anil-folio:jenkins **

-n argocd

After successful image creation there will be an error in the deployment i.e., ImagePullBackOff error. To fix this we need to add a pull secret:

|_ Generate a PAT token in docker:

|_ Go to https://app.docker.com/accounts/anil188/settings/personal-access-tokens

|_ Generate new token

|_ Give name, Expiration Date, Access permission: Read, write, Delete

Create a K8S secret:

**kubectl create secret docker-registry argo-docker **

**--docker-server=https://index.docker.io/v1/ **

**--docker-username=anil188 **

--docker-password=dckr_pat_sFbkyH5Td0hE (paste the token) ****

**--docker-email=anilsa2345@gmail.com **

-n anil-folio

|_ Patch the Service account:

**kubectl patch serviceaccount jenkins **
**-n anil-folio **
-p '{"imagePullSecrets":[{"name":"argo-docker"}]}'

|_ Restart the deployment:

kubectl rollout restart deployment my-folio-n anil-folio

After all completion of the setps, go to Jenkins > New Item > Give name > pipeline > Ok > In pipeline section: select pipeline script from SCM > Select SCM: Git > paste the repo url > Give Cred name > Save > Build Now

Jenkins file for VM:

```
pipeline {
    agent any
        tools {

            nodejs 'nodeJS 18'
        }

    stages {
        stage('Git AppRepo Checkout') {
            steps {
                git branch: 'main', credentialsId: 'github-login-cred', url: <App repo url>'
            }
        }
        stage('Install Dependencies') {
            steps {
                // Use the named NodeJS installation for executing npm commands
                nodejs(nodeJSInstallationName: 'nodeJS 18') {
                    sh 'npm install --force'
                }
            }
        }
        stage('SonarQube Analysis') {
            environment {
                SCANNER_HOME = tool 'sonar-scanner'
            }
```

```
    steps {
        withSonarQubeEnv('sonarqube') {
            sh """
                ${SCANNER_HOME}/bin/sonar-scanner \
                -Dsonar.projectKey=my-folio\
                -Dsonar.projectName=my-folio\
                -Dsonar.sources=src \
                -Dsonar.sourceEncoding=UTF-8 \
                -Dsonar.exclusions=**/node_modules/**,**/.next/**,**/*.d.ts
            """
        }
    }
}

stage('Docker Build and Push') {
    steps {
        script {
            withDockerRegistry(
                credentialsId: 'docker-jenkins-cred',
                url: 'https://index.docker.io/v1/'
            ) {
                sh """
                    docker build -t anil188/my-portfolio:${BUILD_NUMBER} .
                    docker push anil188/my-portfolio:${BUILD_NUMBER}
                """
            }
        }
    }
}

stage('Git Values Repo Checkout') {
    steps {
        git branch: 'master', credentialsId: 'github-login-cred', url: <values repo>'
    }
}

stage('Update Image Tag in values.yml') {
    steps {
        withCredentials([
            usernamePassword(
                credentialsId: 'github-login-cred',
                usernameVariable: 'GIT_USERNAME',
                passwordVariable: 'GIT_TOKEN'
```

```
                )
            ]) {
                script {
                    sh 'ls -l'

                    def valuesFile = readFile('values.yml')
                    def newTag = "my-portfolio:${env.BUILD_NUMBER}"
                    def updatedContent = valuesFile.replaceAll(/my-portfolio:.*/, newTag)
                    writeFile(file: 'values.yml', text: updatedContent)

                    echo "✅ values.yaml updated with tag: ${newTag}"

                    sh """
                        git config user.name "AnilKumar"
                        git config user.email "anilsa3424@gmail.com"
                        git add values.yml
                        git commit -m "Update image tag to ${BUILD_NUMBER}" || echo "No
changes to commit"
                        git push https://x-access-token:${GIT_TOKEN}@github.com/Anil-
KumarCH/values-repo.git master

                    """
                }
            }
        }
    }


    stage('Check and Create Namespace') {
        steps {
            withCredentials([file(credentialsId: 'k8s-jenkins-file-cred', variable: 'KUBECONFIG')])
{
                script {
                    def nsStatus = sh(script: "kubectl get namespace anil-folio", returnStatus: true)
                    if (nsStatus == 0) {
                        echo "Namespace 'anil-folio' already exists. Skipping creation."
                    } else {
                        sh "kubectl create namespace anil-folio"
                        echo "Namespace 'anil-folio' created successfully."
                    }
                }
            }
        }
```

```
        }

        stage('Create ArgoCD Application') {
            steps {
                withCredentials([
                    file(credentialsId: 'k8s-jenkins-file-cred', variable: 'KUBECONFIG'),
                    usernamePassword(
                        credentialsId: 'jenkins-argocd-cred',
                        usernameVariable: 'ARGOCD_USERNAME',
                        passwordVariable: 'ARGOCD_PASSWORD'
                    )
                ]) {
                    script{
                        sh '''
                            argocd login <Argocd URL> \
                            --username "$ARGOCD_USERNAME" \
                            --password "$ARGOCD_PASSWORD" \
                            --insecure \
                            --grpc-web
                        '''
                        // Check if app exists
                        def appExists = sh(
                            script: 'argocd app get my-folio>/dev/null 2>&1',
                            returnStatus: true
                        )
                        if (appExists == 0) {
                            echo "ArgoCD application 'my-portfolio' already exists. Skipping creation."
                        } else {
                            sh '''
                                echo "Creating ArgoCD application 'my-portfolio'..."
                                kubectl apply -f application.yml -n argocd
                                echo "ArgoCD application 'my-portfolio' created successfully."
                            '''
                        }

                    }
                }
            }
        }
    }
}
```

To install Ingress:

1) To install Ingress:
   |_ **kubectl apply -f https://raw.githubusercontent.com/kubernetes/ingress-nginx/main/deploy/static/provider/cloud/deploy.yaml**
2) To get the ingress service IP and port:
   |_ **kubectl get svc -n ingress-nginx**
3) **Sample output:**

NAME TYPE CLUSTER-IP EXTERNAL-IP PORT(S) AGE
ingress-nginx-controller LoadBalancer 10.43.206.82 <pending> 80:30744/TCP,443:30596/TCP 17s
ingress-nginx-controller-admission ClusterIP 10.43.179.237 <none> 443/TCP 17s

Cloudflared Tunnel Creation:

1) To create tunnel
   |_ **cloudflared tunnel create folio-tunnel**
2) To copy cred file:
   |_ **sudo cp /home/vmadin/.cloudflared/7f5541e7-0c-8eb7-1a27fbeff2e2.json /etc/cloudflared/**
3) To create and edit file:
   |_**sudo vi /etc/cloudflared/portfolio-config.yaml**

```
tunnel: 7f5541e7-0c-8eb7-1a27fbeff2e2
credentials-file: /etc/cloudflared/7f5541e7-0c-8eb7-1a27fbeff2e2.json
ingress:
  - hostname: arksglobalasia.cloud <your domain>
    service: <k8s url>: <port>
  - service: http_status:404
```

   Esc > :wq > Enter : save and exit
4) To change the owner
   |_**sudo chown root:root /etc/cloudflared/***
5) To set the permissions
   |_**sudo chmod 600 /etc/cloudflared/***
6) To create service file:
   |_ **sudo vi /etc/systemd/system/folio-tunnel.service**

```
[Unit]
Description=Cloudflare Tunnel (SSH)
After=network-online.target
Wants=network-online.target

[Service]
Type=simple
```

ExecStart=/usr/bin/cloudflared tunnel --config /etc/cloudflared/folio-config.yaml run
Restart=always
RestartSec=5
User=root

[Install]
WantedBy=multi-user.target

7) To load daemon
   |_**sudo systemctl daemon-reexec**
8) To reload daemon
   |_ **sudo systemctl daemon-reload**
9) To start the service
   |_**sudo systemctl start folio-tunnel**
10) To enable service so it will start on boot
   |_**sudo systemctl enable folio-tunnel**
11) To know the status of the service
   |_sudo systemctl status portfolio-tunnel
12) To add dns record
   |_ **cloudflared tunnel route dns folio-tunnel arksglobalasia.cloud**