# Introduction to infrastructure/msa source code:

* 1. **Infrastructure source code:**

Clone the repository on https://github.com/namhoangHvN/sd1095\_msa.git

A screenshot of a computer

Description automatically generated

**The folders will contain these files and folder:**

* Folder backend, backend source code and docker file
* Folder frontend, frontend source code and docker file
* Folder mongodb, mongodb source code and docker file
* File azure-pipelines.yml contain pipeline for azure to deploy “CI part in CI/CD”
  1. **MSA source code:**

Clone the repository on https://github.com/namhoangHvN/sd1095\_\_azure\_infrastructure.git

A screenshot of a computer

Description automatically generated

**The folders will contain these files and folder:**

* Folder k8smanifest contains manifest to deploy backend, frontend, MongoDB from AKS
* Folder argocd contains the manifest file to create a project and app for argocd without interacting with the UI
* Folder terraform contains source code to create infrastructure on Azure
* File azure-pipelines.yml contain pipeline for azure to deploy “CD part in CI/CD”

# Setup infrastructure with Terraform:

* 1. **Initialize Terraform**

Run terraform init to initialize the Terraform deployment. This command downloads the Azure provider required to manage your Azure resources.

Command

terraform init -upgrade

A screenshot of a computer program

Description automatically generated

* 1. **Create a Terraform execution plan**

Run terraform plan to create an execution plan.

Command

terraform plan -out main.tfplan

A computer screen shot of a blue screen

Description automatically generated

* 1. **Apply a Terraform execution plan**

Run Terraform apply to apply the execution plan to your cloud infrastructure.

Command

terraform apply main.tfplan

A screenshot of a computer program

Description automatically generated

* 1. **Write down the cluster name and resource group**

A screenshot of a computer screen

Description automatically generated

* 1. **Verify the results:**

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

# Setup Azure DevOps pipelines

**Create new ACR connections:**

* 1. Go to project settings.

A screenshot of a computer

Description automatically generated A screenshot of a computer

Description automatically generated

* 1. Then go to the Service connections, choose new service connection

A screenshot of a computer

Description automatically generated

* 1. Then find the Docker Registry

A screenshot of a computer

Description automatically generated

* 1. Then choose your ACR, grant access permission to all pipelines, then save

A screenshot of a computer

Description automatically generated

* 1. Find and choose your new created connection

A screenshot of a computer

Description automatically generated

* 1. Then write down the resource ID on the url

A screenshot of a computer

Description automatically generated

* 1. Go to Pipelines and update those vars in dev-var-group:

A screenshot of a computer

Description automatically generated

**Vars:**

* azureResourceGroup: your azure resource group name
* azureSubscriptionConnection: your subscription ID
* containerRegistry: Your acr connection
* dockerRegistryServiceConnection: your acr connection resource ID
* kubernetesCluster: your azure k8s resource group name
  1. Then run msa pipeline for CI in CI/CD:

A screenshot of a computer

Description automatically generated

* 1. Verify result at pipeline:

A screenshot of a computer

Description automatically generated

* 1. Verify result at ACR:

A screenshot of a computer

Description automatically generated

* 1. Your will see backend and frontend repos at ACR

Choose backend or frontend:

A screenshot of a computer

Description automatically generated

Get the lastest tag (ex: 1 in the pics) and update to these files:

AKS setup

* 1. **Connect to your AKS:**

Command

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

* 1. **Show your pods in your AKS after deployed by the pipelines:**

Command

kubectl get pods

A blue screen with white text

Description automatically generated

* 1. **Install NGINX ingress controller:**

Command

kubectl apply -f <https://raw.githubusercontent.com/kubernetes/ingress-nginx/controller-v1.7.1/deploy/static/provider/cloud/deploy.yaml>

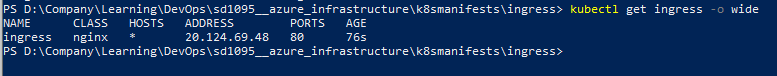
A screenshot of a computer program

Description automatically generated

* 1. **Get the address:**

Command

kubectl get ingress -o wide



* 1. **Access the address and verify result:**

A screenshot of a computer

Description automatically generated

# Setup ArgoCD

* 1. **Create namespace**

Command

kubectl create namespace argocd

A blue screen with white text

Description automatically generated

* 1. **Setup ArgoCD:**

Command

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

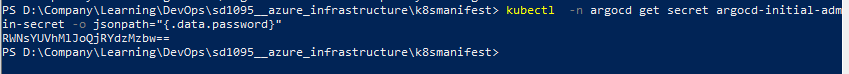
A screenshot of a computer screen

Description automatically generated

* 1. **Get account password:**

Command

kubectl -n argocd get secret argocd-initial-admin-secret -o jsonpath=”{.data.password}”



* 1. **Decode it and get the value before root with the command on linux wsl:**

Command

echo RWNsYUVhMlJoQjRYdzMzbw==| base64 --decode

A screenshot of a computer

Description automatically generated

* 1. **Port forward it and access it local with the command:**

Command

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

A screenshot of a computer

Description automatically generated

Access it with:

username: admin

password: the decoded secret above

* 1. **Verify result:**

A screenshot of a computer

Description automatically generated

* 1. **Setup project**

Command

kubectl apply -f argoproject.yml

* 1. **Verify result:**

A computer screen shot of a blue screen

Description automatically generated

A screenshot of a computer

Description automatically generated

* 1. **Setup app:**

Command

kubectl apply -f argoapp.yml

A blue screen with white text

Description automatically generated

A screenshot of a computer

Description automatically generated

* 1. **Verify result:** A screenshot of a computer

     Description automatically generated
  2. **Test your ArgoCD:**

Look we got 3 frontend pods and 2 pods Mongodb so let’s fix that:

* + 1. **Frontend:**

Change replicas value from 1 to 4

A screenshot of a computer

Description automatically generated

* + 1. **MongoDB:**

Change replicas value from 1 to 2

A screenshot of a computer program

Description automatically generated

* + 1. **Verify the result you will see Mongodb, frontend pods will be in terminating status:**

A screenshot of a computer

Description automatically generated

# Setup Argo image updater

* 1. **Setup Argo image updater with command:**

Command

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

A computer screen shot of a blue screen

Description automatically generated

* 1. **Edit Config**

Command

kubectl edit configmaps --namespace argocd argocd-image-updater-config

A screenshot of a computer

Description automatically generated

* 1. **Add data config for it to pull repo from ACR:**

data:

log.level: debug

registries.conf: |

registries:

name: <your ACR name>

prefix: <your ACR connection (ex yourACRname.azurecr.io)>

api\_url: https://<your ACR connection (ex https://yourACRname.azurecr.io)>

credentials: pullsecret:argocd/argocd-acr-secret

* 1. **Result:**

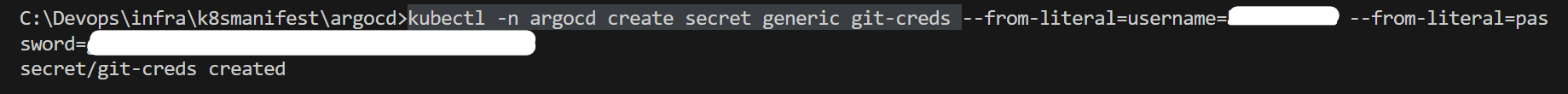
A screenshot of a computer

Description automatically generated

* 1. **Create secret for argo image updater access to GITHUB**

Command

kubectl -n argocd create secret generic git-creds --from-literal=username=<github username> --from-literal=password=<personal access token>



* 1. **Go to you ACR and navigate to access key to get username and password**

A screenshot of a computer

Description automatically generated

Then noted it