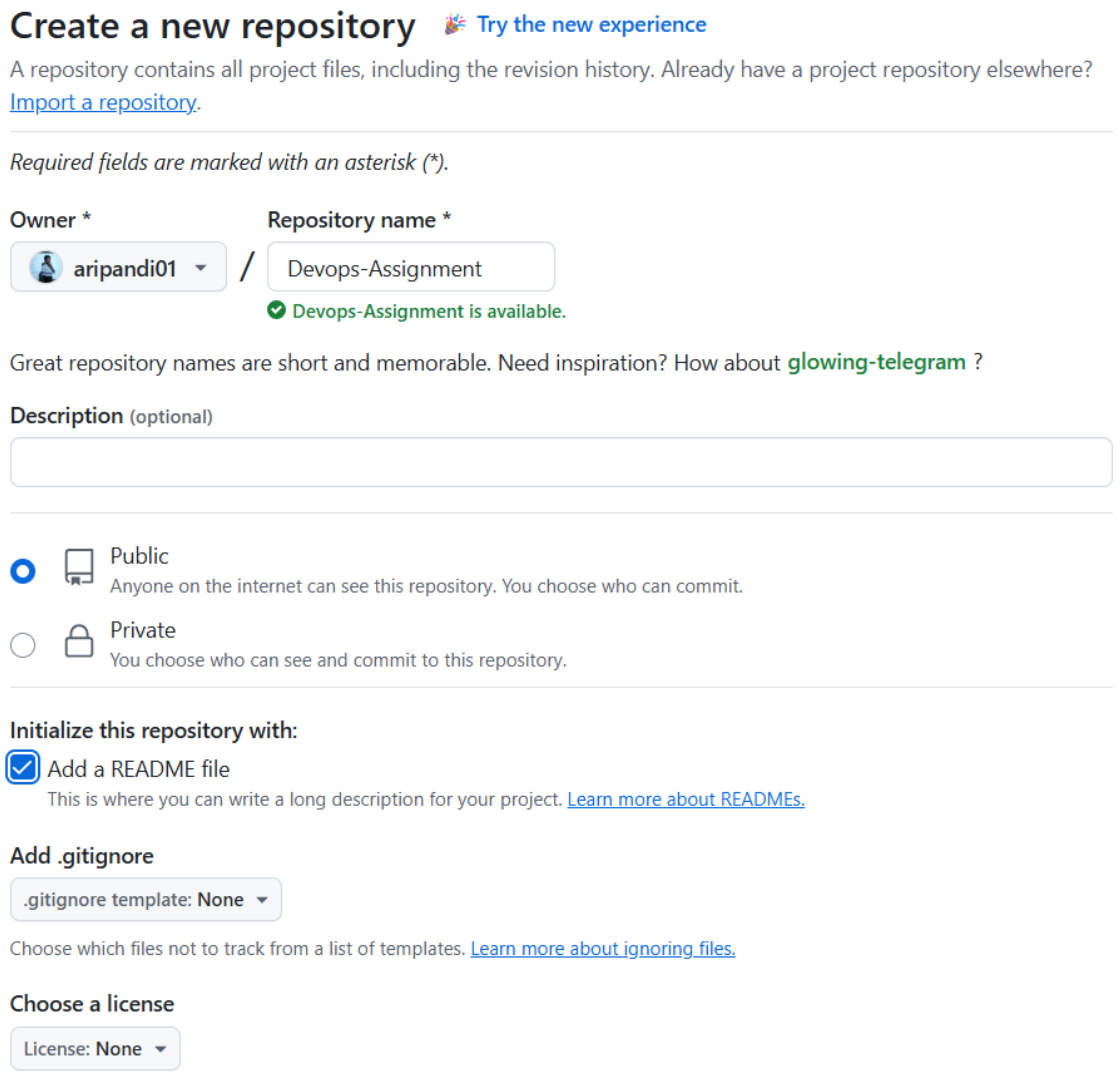
DevOps Interview Assignment – Documentation

1.Prerequisites before start the project

1.1 Setup the Github Repostrories and Check the required tools are installed.

GitHUB and setting up the Directories



```
Lenovo@Aripandi MINGW64 /d
$ git clone https://github.com/aripandi01/Devops-Assignment.git
Cloning into 'Devops-Assignment'...
remote: Enumerating objects: 3, done.
remote: Counting objects: 100% (3/3), done.
remote: Total 3 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)
Receiving objects: 100% (3/3), done.

Lenovo@Aripandi MINGW64 /d
$ cd Devops-Assignment/

Lenovo@Aripandi MINGW64 /d/Devops-Assignment (main)
$ mkdir terraform manifests ArgoCD

Lenovo@Aripandi MINGW64 /d/Devops-Assignment (main)
$ ll
total 1
drwxr-xr-x 1 Lenovo 197609  0 Jul  9 09:33 ArgoCD/
-rw-r--r-- 1 Lenovo 197609 19 Jul  9 09:16 README.md
drwxr-xr-x 1 Lenovo 197609  0 Jul  9 09:33 manifests/
drwxr-xr-x 1 Lenovo 197609  0 Jul  9 09:33 terraform/

Lenovo@Aripandi MINGW64 /d/Devops-Assignment (main)
$
```

1.2 Verify Necessary tools are present

```
PS D:\Devops-Assignment\terraform> terraform -v
Terraform v1.12.2
on windows_amd64
PS D:\Devops-Assignment\terraform> kubectl version --client
Client Version: v1.33.2
Kustomize Version: v5.6.0
PS D:\Devops-Assignment\terraform> helm version
version.BuildInfo{Version:"v3.18.2", GitCommit:"04cad4610054e5d546aa5c5d9c1b1d5cf68ec1f8", GitTreeState:"clean", GoVersion:"go1.24.3"}
PS D:\Devops-Assignment\terraform>
```

1.3 Create IAM user and AWS configure

```
Lenovo@Aripandi MINGW64 /d/Devops-Assignment/terraform (main)
$ aws configure
AWS Access Key ID [****************6MX6]: AKIA6IW7G2KUA6RDWNYV
AWS Secret Access Key [****************0yo7]: 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
Default region name [us-east-1]:
Default output format [json]:
```

Task 1: Provision AWS EKS Cluster using Terraform

Provision a production-ready AWS EKS cluster using Terraform and other AWS resources

Terraform Modules Used

- terraform-aws-modules/vpc/aws
- terraform-aws-modules/eks/aws

Terraform init

```
Terraform has been successfully initialized!

You may now begin working with Terraform. Try running "terraform plan" to see
any changes that are required for your infrastructure. All Terraform commands
should now work.

If you ever set or change modules or backend configuration for Terraform,
rerun this command to reinitialize your working directory. If you forget, other
commands will detect it and remind you to do so if necessary.
```

Terraform plan

```
Plan: 43 to add, 0 to change, 0 to destroy.

Changes to Outputs:
  + cluster_endpoint = (known after apply)
  + cluster_name     = "Devops-Assignment"
```

Terraform apply

```
Apply complete! Resources: 43 added, 0 changed, 0 destroyed.

Outputs:
                    Follow link (ctrl + click)
cluster_endpoint = "https://393CD6D50B22E69D0E03CBBD1068B657.gr7.us-east-1.eks.amazonaws.com"
cluster_name = "Devops-Assignment-Cluster"
```

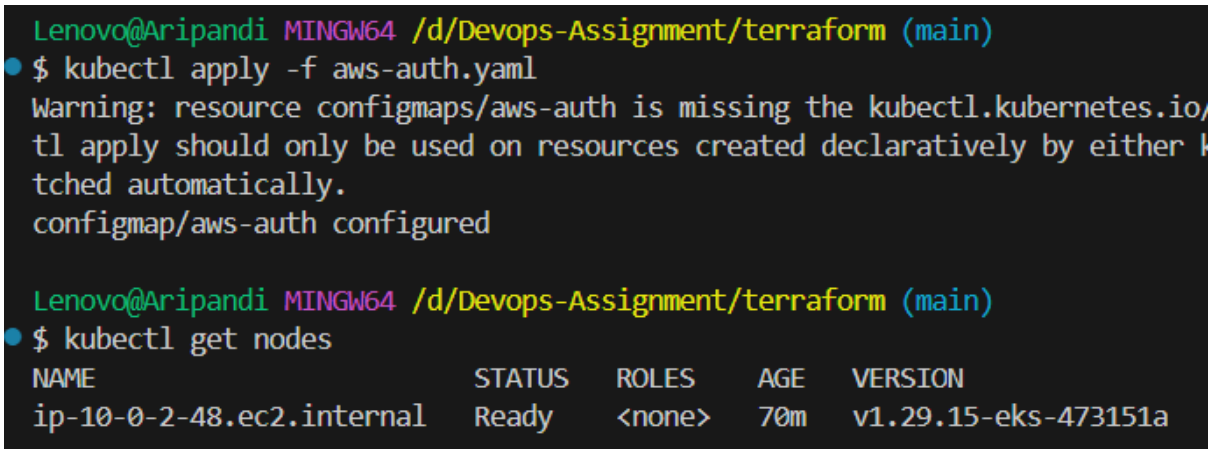
Update kubeconfig to connect to your EKS cluster

```
Lenovo@Aripandi MINGW64 /d/Devops-Assignment/terraform (main)
$ aws eks --region us-east-1 update-kubeconfig --name Devops-Assignment-Cluster
Added new context arn:aws:eks:us-east-1:980794397352:cluster/Devops-Assignment-Cluster to C:\Users\Lenovo\.kube\config
```

Kubectl get nodes

```
Lenovo@Aripandi MINGW64 /d/Devops-Assignment/terraform (main)
$ kubectl get nodes
E0709 10:59:52.406754   14812 memcache.go:265] "Unhandled Error" err="couldn't get current server API group list: Get \"https://393CD6D50B22E69D0E03CBBD10
68B657.gr7.us-east-1.eks.amazonaws.com/api?timeout=32s\": dial tcp [64:ff9b::a00:1b0]:443: i/o timeout"
E0709 11:00:22.413155   14812 memcache.go:265] "Unhandled Error" err="couldn't get current server API group list: Get \"https://393CD6D50B22E69D0E03CBBD10
68B657.gr7.us-east-1.eks.amazonaws.com/api?timeout=32s\": dial tcp [64:ff9b::a00:1b0]:443: i/o timeout"
E0709 11:00:52.416455   14812 memcache.go:265] "Unhandled Error" err="couldn't get current server API group list: Get \"https://393CD6D50B22E69D0E03CBBD10
68B657.gr7.us-east-1.eks.amazonaws.com/api?timeout=32s\": dial tcp [64:ff9b::a00:2d7]:443: i/o timeout"
E0709 11:01:22.419730   14812 memcache.go:265] "Unhandled Error" err="couldn't get current server API group list: Get \"https://393CD6D50B22E69D0E03CBBD10
68B657.gr7.us-east-1.eks.amazonaws.com/api?timeout=32s\": dial tcp [64:ff9b::a00:2d7]:443: i/o timeout"
E0709 11:01:52.424678   14812 memcache.go:265] "Unhandled Error" err="couldn't get current server API group list: Get \"https://393CD6D50B22E69D0E03CBBD10
68B657.gr7.us-east-1.eks.amazonaws.com/api?timeout=32s\": dial tcp [64:ff9b::a00:1b0]:443: i/o timeout"
Unable to connect to the server: dial tcp [64:ff9b::a00:1b0]:443: i/o timeout
```

Error message found on AWS Console: Your IAM principal doesn't have Kubernetes RBAC access
Root cause: Terraform doesn't auto-configure aws-auth
Fix : Manually apply aws-auth.yaml ConfigMap



Troubleshooting Faced & Solved

Error	Resolution
Subnets must be in 2 AZs	Used `us-east-1a`, `us-east-1b`
IAM user lacked Kubernetes access	Patched `aws-auth` ConfigMap
KMS or Log group already exists	Renamed cluster or deleted existing resources manually

Task Conclusion

- A custom VPC with two subnets across two availability zones was created.

- An EKS cluster and a managed node group were provisioned.

- IAM roles and service integrations (IRSA) were configured.

- Kubernetes access was established via `aws-auth` ConfigMap and verified using `kubectl`.

- The infrastructure is modular, reproducible, and fully managed via Terraform.

- All components are functional, and the EKS cluster is ready for workload deployment.

Task 2: Deploy an NGINX Application Using Kubernetes Manifest

Deploy a simple NGINX web application on the EKS cluster using Kubernetes YAML manifests

Files Created:

1. manifests/nginx-deployment.yaml
2. manifests/nginx-service.yaml

Deployment Steps:

```
Lenovo@Aripandi MINGW64 /d/Devops-Assignment (main)
$ kubectl apply -f manifests/nginx-deployment.yaml
deployment.apps/nginx-deployment created

Lenovo@Aripandi MINGW64 /d/Devops-Assignment (main)
$ kubectl apply -f manifests/nginx-service.yaml
service/nginx-service created
```

Verification Commands:

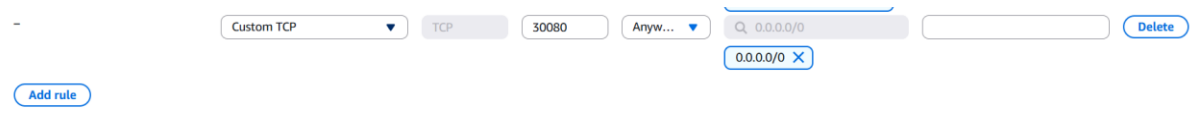
```
$ kubectl get pods
NAME                               READY   STATUS    RESTARTS   AGE
nginx-deployment-7667bf9c77-drgm9   1/1     Running   0          21s
nginx-deployment-7667bf9c77-s8bqp   1/1     Running   0          21s

Lenovo@Aripandi MINGW64 /d/Devops-Assignment (main)
$ kubectl get svc
NAME            TYPE        CLUSTER-IP       EXTERNAL-IP   PORT(S)        AGE
kubernetes      ClusterIP   172.20.0.1       <none>        443/TCP        101m
nginx-service   NodePort    172.20.111.100   <none>        80:30080/TCP   17s

Lenovo@Aripandi MINGW64 /d/Devops-Assignment (main)
$ kubectl get nodes -o wide
NAME                         STATUS   ROLES    AGE   VERSION               INTERNAL-IP   EXTERNAL-IP   OS-IMAGE         KERNEL-VERSION
NTAINER-RUNTIME
ip-10-0-2-48.ec2.internal    Ready    <none>   94m   v1.29.15-eks-473151a   10.0.2.48     54.87.41.20   Amazon Linux 2   5.10.238-231.953.amzn2.x86_64
ntainerd://1.7.27
```

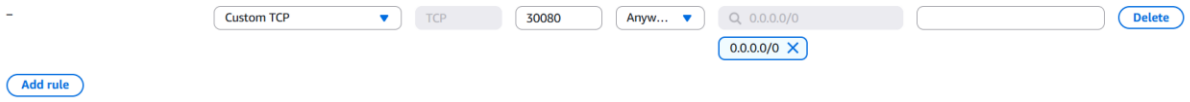
Application Access

- **Public EC2 Node IP: 54.87.41.20**
- **NodePort: 30080**



Troubleshooting Done:

- **Opened TCP:30080 in EC2 node's Security Group**
- **Ensured pods were Running**
- **Confirmed node had a public IP**



Task 3: Set Up ArgoCD on EKS and Deploy NGINX via GitOps

Created Namespace for ArgoCD

```
Lenovo@Aripandi MINGW64 /d/Devops-Assignment (main)
$ kubectl create namespace argocd
namespace/argocd created
```

Installed ArgoCD

```
Lenovo@Aripandi MINGW64 /d/Devops-Assignment (main)
$ kubectl apply -n argocd -f https://raw.githubusercontent.com/argoproj/argo-cd/stable/manifests/instal
customresourcedefinition.apiextensions.k8s.io/applications.argoproj.io created
customresourcedefinition.apiextensions.k8s.io/applicationsets.argoproj.io created
customresourcedefinition.apiextensions.k8s.io/appprojects.argoproj.io created
serviceaccount/argocd-application-controller created
serviceaccount/argocd-applicationset-controller created
serviceaccount/argocd-dex-server created
```

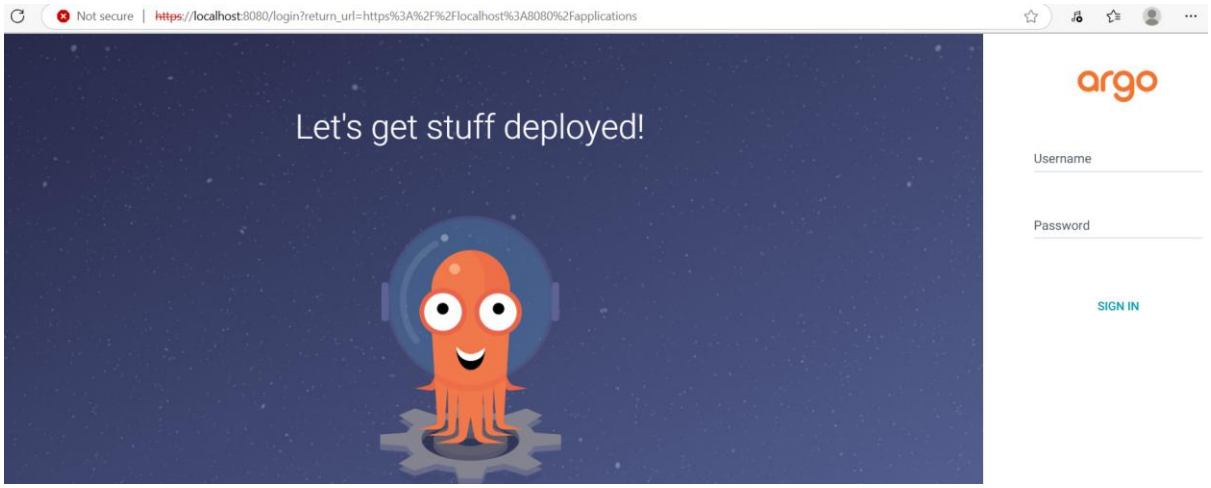
Verified

```
Lenovo@Aripandi MINGW64 /d/Devops-Assignment (main)
$ kubectl get pods -n argocd
NAME                                                READY   STATUS    RESTARTS      AGE
argocd-application-controller-0                     1/1     Running   0             57s
argocd-applicationset-controller-77cb676b6c-4tfsr   1/1     Running   0             65s
argocd-dex-server-69d76976fd-26krd                  1/1     Running   2 (44s ago)   63s
argocd-notifications-controller-7776574f76-cc7xl    1/1     Running   0             61s
argocd-redis-6dd5986579-6pvqv                       1/1     Running   0             60s
argocd-repo-server-575bbf7949-7t4fh                 1/1     Running   0             59s
argocd-server-6769964c47-zkrxq                      1/1     Running   0             58s
```

Exposed ArgoCD UI (Port-Forward Method)

```
Lenovo@Aripandi MINGW64 /d/Devops-Assignment (main)
$ kubectl port-forward svc/argocd-server -n argocd 8080:443
Forwarding from 127.0.0.1:8080 -> 8080
Forwarding from [::1]:8080 -> 8080
```

Accessed ArgoCD UI at: https://localhost:8080

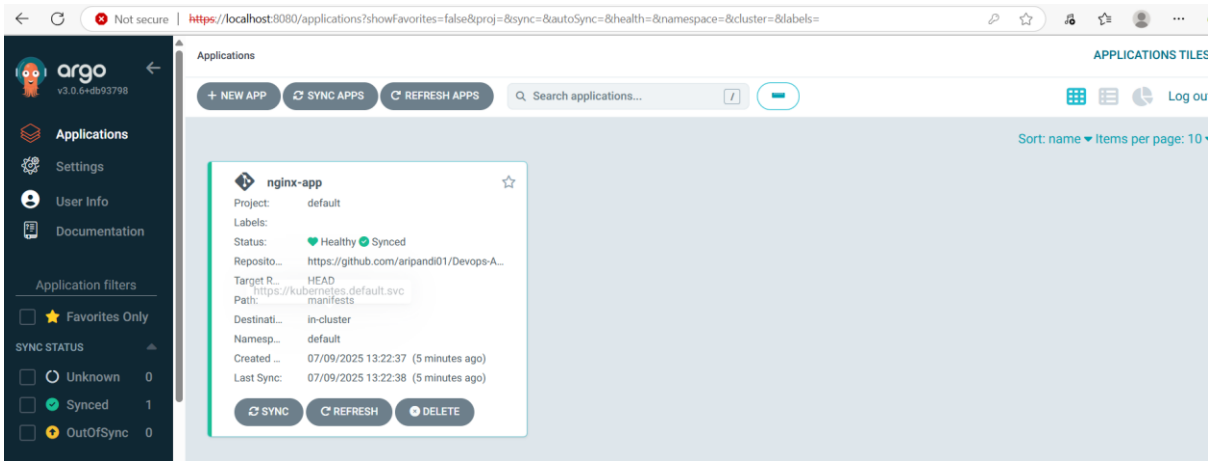



```
Lenovo@Aripandi MINGW64 /d/Devops-Assignment (main)
$ kubectl get secret argocd-initial-admin-secret -n argocd -o jsonpath="{.data.password}" | base64 -d
1eGtxdrSnZ7992Ze
```

Created ArgoCD Application YAML

```
$ kubectl apply -f ArgoCD/nginx-app.yaml
application.argoproj.io/nginx-app created
```

Logged in to ArgoCD



Troubleshoot faced

Issue	Solution
no matches for kind "Application"	**Waited for ArgoCD CRDs to install properly**
ArgoCD UI not opening	**Restarted kubectl port-forward**

The NGINX application's accessibility **(Task 4)** was also verified as part of this task. The service was exposed via NodePort, and the application was successfully accessed using the EC2 public IP and NodePort. Hence, Task 4 is considered completed within this step.

Task 5: Expose NGINX via Ingress + Custom Domain

Installed NGINX Ingress Controller using Helm

```
Lenovo@Aripandi MINGW64 /d/Devops-Assignment (main)
$ kubectl create namespace ingress-nginx
namespace/ingress-nginx created

Lenovo@Aripandi MINGW64 /d/Devops-Assignment (main)
$ helm repo add ingress-nginx https://kubernetes.github.io/ingress-nginx
"ingress-nginx" has been added to your repositories

Lenovo@Aripandi MINGW64 /d/Devops-Assignment (main)
$ helm repo update
Hang tight while we grab the latest from your chart repositories...
...Successfully got an update from the "ingress-nginx" chart repository
Update Complete. ⎈Happy Helming!⎈
```

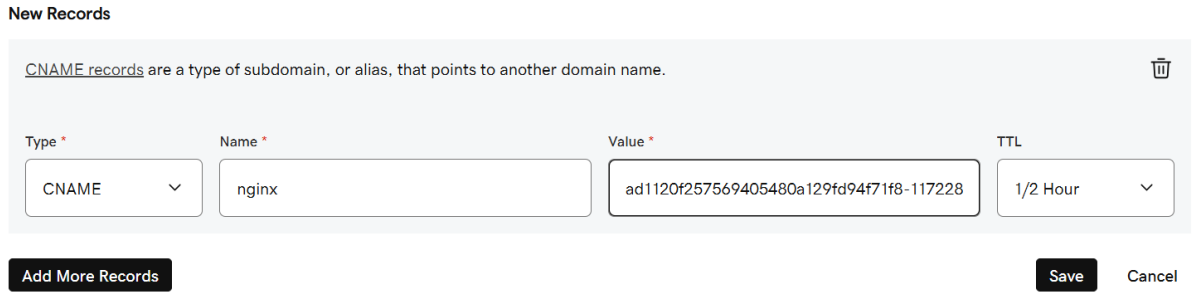
```
Lenovo@Aripandi MINGW64 /d/Devops-Assignment (main)
$ helm install ingress-nginx ingress-nginx/ingress-nginx \
  --namespace ingress-nginx \
  --set controller.service.type=LoadBalancer
NAME: ingress-nginx
LAST DEPLOYED: Wed Jul  9 13:44:29 2025
NAMESPACE: ingress-nginx
STATUS: deployed
```

```
Lenovo@Aripandi MINGW64 /d/Devops-Assignment (main)
$ kubectl get svc -n ingress-nginx
NAME                                 TYPE           CLUSTER-IP       EXTERNAL-IP                                                               PORT(S)
                AGE
ingress-nginx-controller             LoadBalancer   172.20.136.236   ad1120f257569405480a129fd94f71f8-1172287157.us-east-1.elb.amazonaws.com   80:32311/TC
P,443:30817/TCP   90s
ingress-nginx-controller-admission   ClusterIP      172.20.238.155   <none>                                                                    443/TCP
                90s
```

Created Ingress Resource for the NGINX App

```
Lenovo@Aripandi MINGW64 /d/Devops-Assignment (main)
$ kubectl apply -f manifests/nginx-ingress.yaml
ingress.networking.k8s.io/nginx-ingress created
```

Configured DNS in GoDaddy



Got 404 found and changed the host in nginx-ingress.yaml and applied again

Accessed the NGINX App via Domain (http://nginx.productivitypro.xyz)