Kubernetes Services, route53, ingress - Lab Assignment

Part 1: Recap of Previous Class

- Learned Kubernetes routing and internal networking
- Wrote custom YAML files for:
  - Application Pods
  - ReplicaSets (for scaling and high availability)

Part 2: Introduction to Kubernetes Services

What is a Service?
- A Service in Kubernetes is an abstraction that provides a stable IP and DNS name to access a set of ephemeral Pods.
- Services allow consistent network access even as pods are recreated or rescheduled.

**Types of Services: **
1. **ClusterIP**
   - Default service type
   - Accessible only within the cluster
   - Used for internal communication

2. **NodePort**
   - Exposes service on a static port on each node (range: 30000–32767)
   - Accessible externally via <NodeIP>:<NodePort>

3. **LoadBalancer**
   - Provisions a cloud provider load balancer (e.g., AWS ELB)
   - Suitable for production-grade apps needing external access

4. **ExternalName**
   - Maps the service to an external DNS name
   - Returns a CNAME record instead of proxying traffic

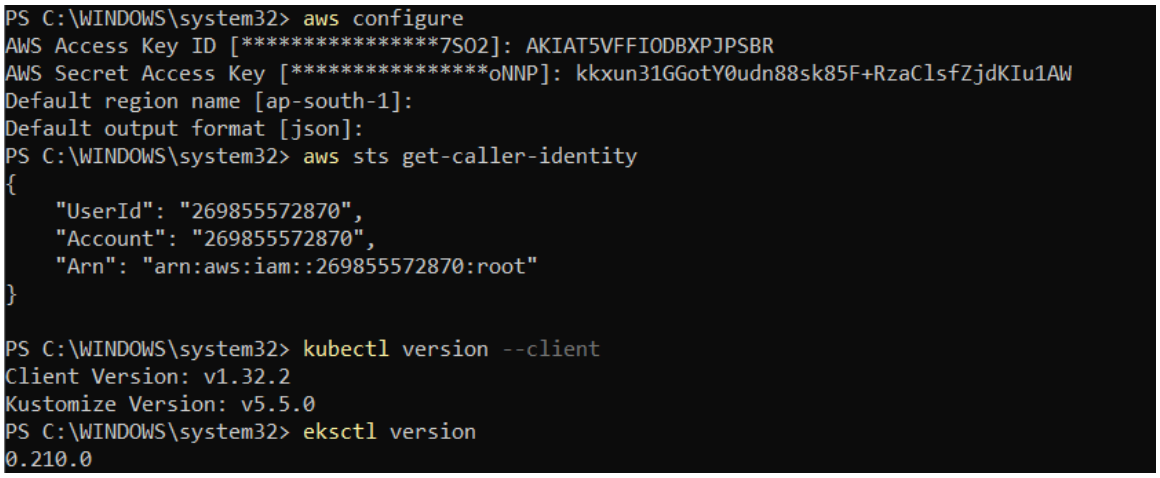
Part 3: Complete Lab Setup on AWS

Step 1: Set up AWS CLI
- Open PowerShell as administrator
- Run `aws configure`
  - Provide Access Key ID
  - Secret Access Key
  - Region: ap-south-1
  - Output format: json

To verify:
aws sts get-caller-identity

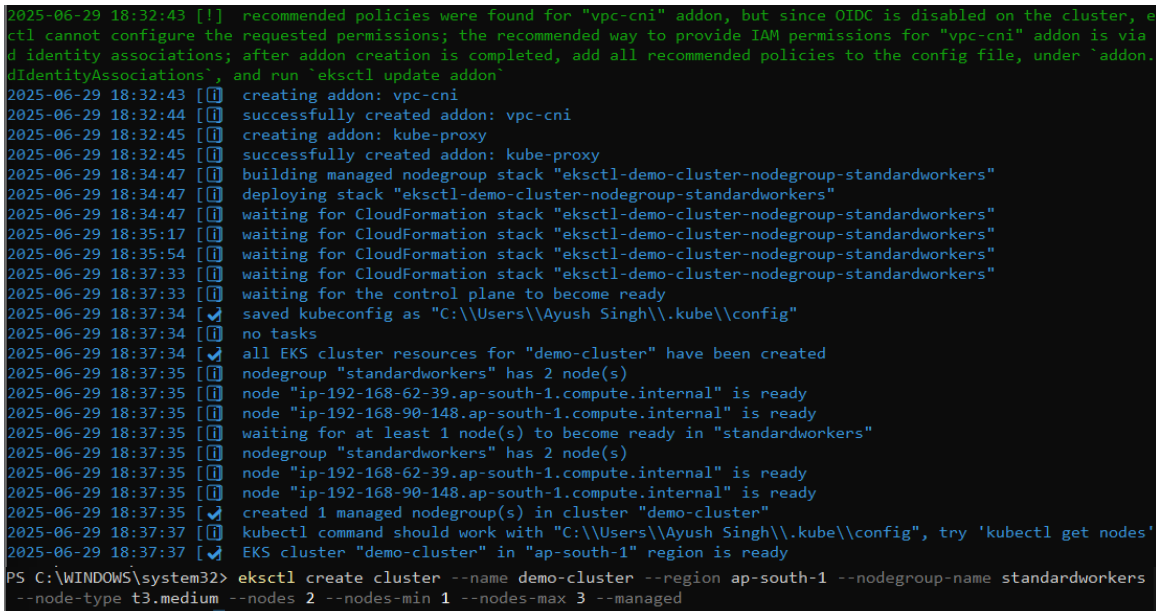
Step 2: Install Required Tools
- AWS CLI
- `kubectl`
- `eksctl`

Verify with:
kubectl version --client
eksctl version



```
PS C:\WINDOWS\system32> aws configure
AWS Access Key ID [****************7SO2]: AKIAT5VFFIODBXPJPSBR
AWS Secret Access Key [****************oNNP]: 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
Default region name [ap-south-1]:
Default output format [json]:
PS C:\WINDOWS\system32> aws sts get-caller-identity
{
    "UserId": "269855572870",
    "Account": "269855572870",
    "Arn": "arn:aws:iam::269855572870:root"
}

PS C:\WINDOWS\system32> kubectl version --client
Client Version: v1.32.2
Kustomize Version: v5.5.0
PS C:\WINDOWS\system32> eksctl version
0.210.0
```

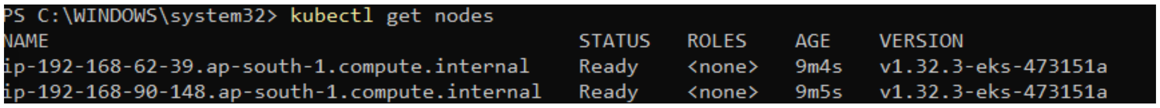
Step 3: Create an EKS Cluster
```bash
eksctl create cluster --name demo-cluster --region ap-south-1 --nodegroup-name
standardworkers --node-type t3.medium --nodes 2 --nodes-min 1 --nodes-max 3 --managed
```



```
2025-06-29 18:32:43 [!]  recommended policies were found for "vpc-cni" addon, but since OIDC is disabled on the cluster, e
ctl cannot configure the requested permissions; the recommended way to provide IAM permissions for "vpc-cni" addon is via
d identity associations; after addon creation is completed, add all recommended policies to the config file, under `addon.
dIdentityAssociations`, and run `eksctl update addon`
2025-06-29 18:32:43 [ℹ]  creating addon: vpc-cni
2025-06-29 18:32:44 [ℹ]  successfully created addon: vpc-cni
2025-06-29 18:32:45 [ℹ]  creating addon: kube-proxy
2025-06-29 18:32:45 [ℹ]  successfully created addon: kube-proxy
2025-06-29 18:34:47 [ℹ]  building managed nodegroup stack "eksctl-demo-cluster-nodegroup-standardworkers"
2025-06-29 18:34:47 [ℹ]  deploying stack "eksctl-demo-cluster-nodegroup-standardworkers"
2025-06-29 18:34:47 [ℹ]  waiting for CloudFormation stack "eksctl-demo-cluster-nodegroup-standardworkers"
2025-06-29 18:35:17 [ℹ]  waiting for CloudFormation stack "eksctl-demo-cluster-nodegroup-standardworkers"
2025-06-29 18:35:54 [ℹ]  waiting for CloudFormation stack "eksctl-demo-cluster-nodegroup-standardworkers"
2025-06-29 18:37:33 [ℹ]  waiting for CloudFormation stack "eksctl-demo-cluster-nodegroup-standardworkers"
2025-06-29 18:37:33 [ℹ]  waiting for the control plane to become ready
2025-06-29 18:37:34 [✔  saved kubeconfig as "C:\\Users\\Ayush Singh\\.kube\\config"
2025-06-29 18:37:34 [ℹ]  no tasks
2025-06-29 18:37:34 [✔  all EKS cluster resources for "demo-cluster" have been created
2025-06-29 18:37:35 [ℹ]  nodegroup "standardworkers" has 2 node(s)
2025-06-29 18:37:35 [ℹ]  node "ip-192-168-62-39.ap-south-1.compute.internal" is ready
2025-06-29 18:37:35 [ℹ]  node "ip-192-168-90-148.ap-south-1.compute.internal" is ready
2025-06-29 18:37:35 [ℹ]  waiting for at least 1 node(s) to become ready in "standardworkers"
2025-06-29 18:37:35 [ℹ]  nodegroup "standardworkers" has 2 node(s)
2025-06-29 18:37:35 [ℹ]  node "ip-192-168-62-39.ap-south-1.compute.internal" is ready
2025-06-29 18:37:35 [ℹ]  node "ip-192-168-90-148.ap-south-1.compute.internal" is ready
2025-06-29 18:37:35 [✔  created 1 managed nodegroup(s) in cluster "demo-cluster"
2025-06-29 18:37:37 [ℹ]  kubectl command should work with "C:\\Users\\Ayush Singh\\.kube\\config", try 'kubectl get nodes'
2025-06-29 18:37:37 [✔  EKS cluster "demo-cluster" in "ap-south-1" region is ready
PS C:\WINDOWS\system32> eksctl create cluster --name demo-cluster --region ap-south-1 --nodegroup-name standardworkers
 --node-type t3.medium --nodes 2 --nodes-min 1 --nodes-max 3 --managed
```

Step 4: Verify the Cluster

kubectl get nodes



```
PS C:\WINDOWS\system32> kubectl get nodes
NAME                                            STATUS   ROLES    AGE   VERSION
ip-192-168-62-39.ap-south-1.compute.internal    Ready    <none>   9m4s  v1.32.3-eks-473151a
ip-192-168-90-148.ap-south-1.compute.internal   Ready    <none>   9m5s  v1.32.3-eks-473151a
```

Step 5: Clone the Kubernetes Manifests Repository

git clone https://github.com/sibasish934/kubernetes-manifests.git

cd kubernetes-manifests

Part 4: Access Application Using ClusterIP

Apply Deployment and Service:

\`\`\`bash

kubectl apply -f deployment.yaml

kubectl apply -f service.yaml

\`\`\`

Verify:

\`\`\`bash

kubectl get pods

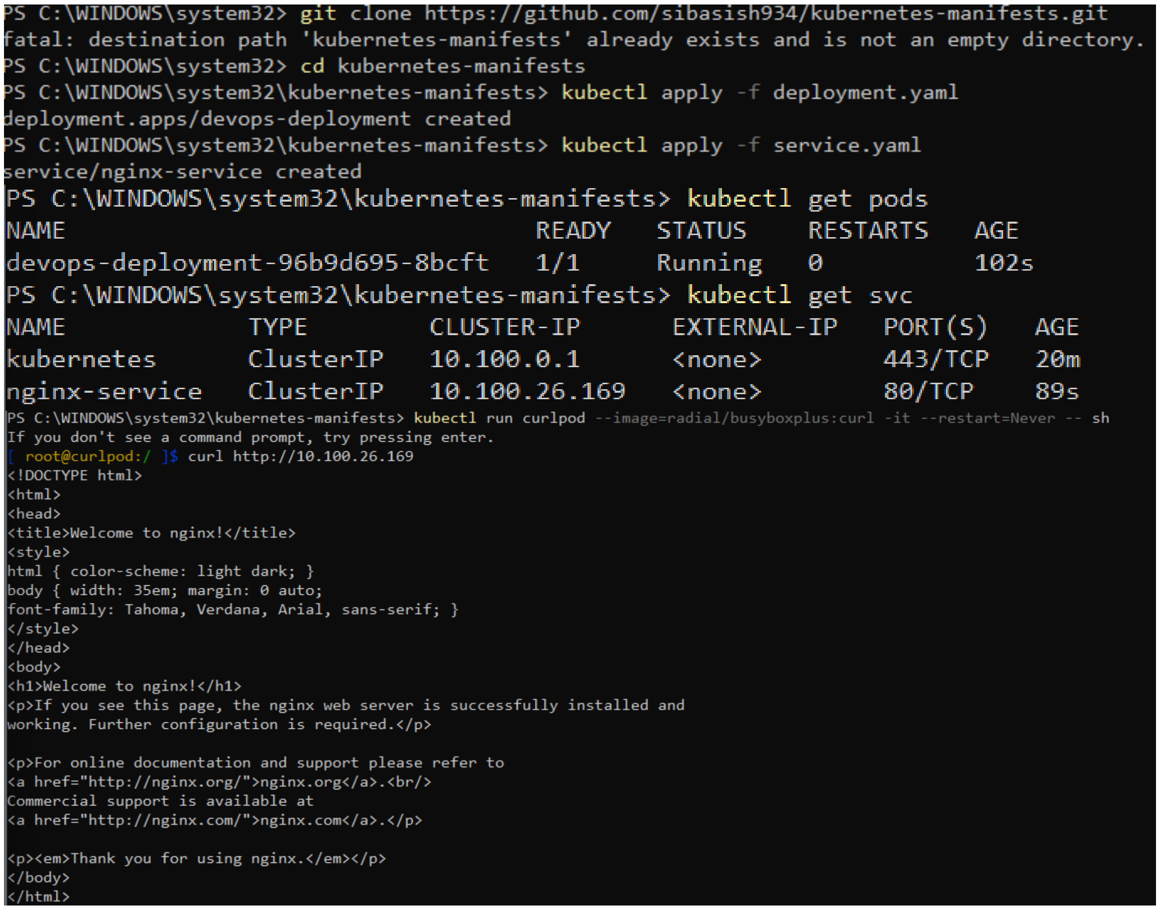
kubectl get svc

\`\`\`

Run a Temporary Curl Pod:

\`\`\`bash

kubectl run curlpod --image=radial/busyboxplus:curl -it --restart=Never -- sh

~~~
Then inside the pod:
```bash
curl http://<CLUSTER-IP>



~~~
PS C:\WINDOWS\system32> git clone https://github.com/sibasish934/kubernetes-manifests.git
fatal: destination path 'kubernetes-manifests' already exists and is not an empty directory.
PS C:\WINDOWS\system32> cd kubernetes-manifests
PS C:\WINDOWS\system32\kubernetes-manifests> kubectl apply -f deployment.yaml
deployment.apps/devops-deployment created
PS C:\WINDOWS\system32\kubernetes-manifests> kubectl apply -f service.yaml
service/nginx-service created
PS C:\WINDOWS\system32\kubernetes-manifests> kubectl get pods
NAME   READY   STATUS   RESTARTS   AGE
devops-deployment-96b9d695-8bcft   1/1   Running   0   102s
PS C:\WINDOWS\system32\kubernetes-manifests> kubectl get svc
NAME   TYPE   CLUSTER-IP   EXTERNAL-IP   PORT(S)   AGE
kubernetes   ClusterIP   10.100.0.1   <none>   443/TCP   20m
nginx-service   ClusterIP   10.100.26.169   <none>   80/TCP   89s
PS C:\WINDOWS\system32\kubernetes-manifests> kubectl run curlpod --image=radial/busyboxplus:curl -it --restart=Never -- sh
If you don't see a command prompt, try pressing enter.
[root@curlpod:/]$ curl http://10.100.26.169
<!DOCTYPE html>
<html>
<head>
<title>Welcome to nginx!</title>
<style>
html { color-scheme: light dark; }
body { width: 35em; margin: 0 auto;
font-family: Tahoma, Verdana, Arial, sans-serif; }
</style>
</head>
<body>
<h1>Welcome to nginx!</h1>
<p>If you see this page, the nginx web server is successfully installed and
working. Further configuration is required.</p>

<p>For online documentation and support please refer to
<a href="http://nginx.org/">nginx.org</a>.<br/>
Commercial support is available at
<a href="http://nginx.com/">nginx.com</a>.</p>

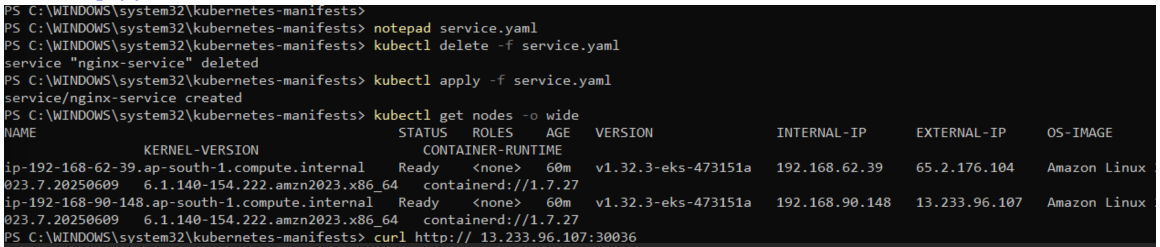
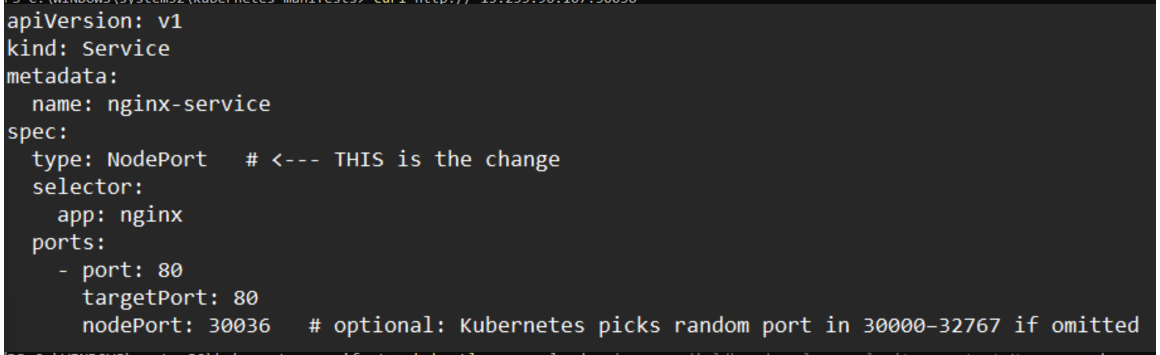
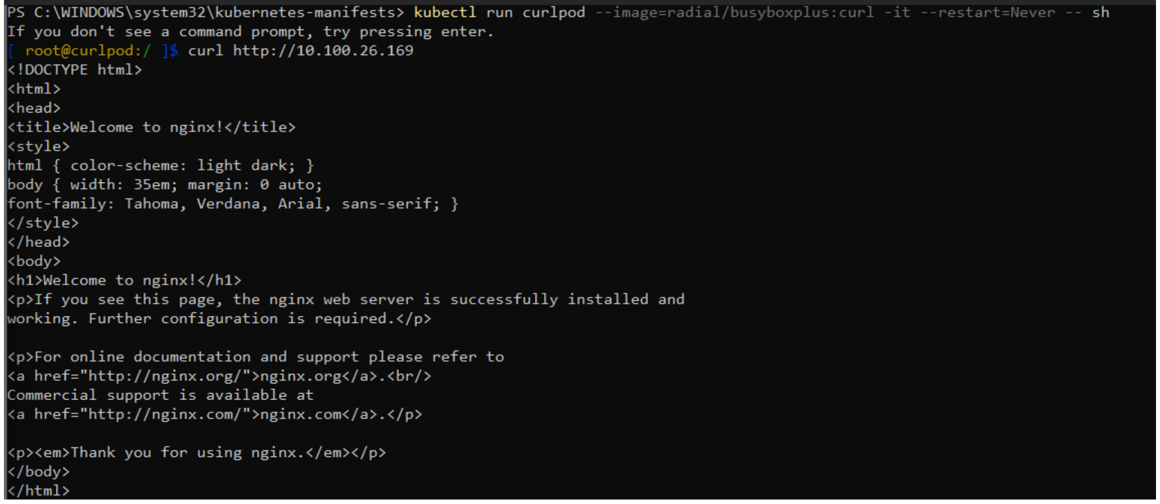
<p><em>Thank you for using nginx.</em></p>
</body>
</html>

Part 5: Access Application Using NodePort

Edit `service.yaml`:
```yaml
spec:
  type: NodePort
  ports:
    - port: 80
      targetPort: 80
      nodePort: 30036
```

Apply the changes:

kubectl delete -f service.yaml

kubectl apply -f service.yaml
```

Get Node's Public IP:

kubectl get nodes -o wide

Access Application:

curl http://<EXTERNAL-IP>:30036



```
PS C:\WINDOWS\system32\kubernetes-manifests>
PS C:\WINDOWS\system32\kubernetes-manifests> notepad service.yaml
PS C:\WINDOWS\system32\kubernetes-manifests> kubectl delete -f service.yaml
service "nginx-service" deleted
PS C:\WINDOWS\system32\kubernetes-manifests> kubectl apply -f service.yaml
service/nginx-service created
PS C:\WINDOWS\system32\kubernetes-manifests> kubectl get nodes -o wide
NAME                                           STATUS   ROLES    AGE   VERSION               INTERNAL-IP      EXTERNAL-IP     OS-IMAGE
                 KERNEL-VERSION                   CONTAINER-RUNTIME
ip-192-168-62-39.ap-south-1.compute.internal   Ready    <none>   60m   v1.32.3-eks-473151a   192.168.62.39    65.2.176.104    Amazon Linux
023.7.20250609   6.1.140-154.222.amzn2023.x86_64   containerd://1.7.27
ip-192-168-90-148.ap-south-1.compute.internal  Ready    <none>   60m   v1.32.3-eks-473151a   192.168.90.148   13.233.96.107   Amazon Linux
023.7.20250609   6.1.140-154.222.amzn2023.x86_64   containerd://1.7.27
PS C:\WINDOWS\system32\kubernetes-manifests> curl http:// 13.233.96.107:30036
```



```
apiVersion: v1
kind: Service
metadata:
  name: nginx-service
spec:
  type: NodePort   # <--- THIS is the change
  selector:
    app: nginx
  ports:
    - port: 80
      targetPort: 80
      nodePort: 30036   # optional: Kubernetes picks random port in 30000-32767 if omitted
```



```
PS C:\WINDOWS\system32\kubernetes-manifests> kubectl run curlpod --image=radial/busyboxplus:curl -it --restart=Never -- sh
If you don't see a command prompt, try pressing enter.
[ root@curlpod:/ ]$ curl http://10.100.26.169
<!DOCTYPE html>
<html>
<head>
<title>Welcome to nginx!</title>
<style>
html { color-scheme: light dark; }
body { width: 35em; margin: 0 auto;
font-family: Tahoma, Verdana, Arial, sans-serif; }
</style>
</head>
<body>
<h1>Welcome to nginx!</h1>
<p>If you see this page, the nginx web server is successfully installed and
working. Further configuration is required.</p>

<p>For online documentation and support please refer to
<a href="http://nginx.org/">nginx.org</a>.<br/>
Commercial support is available at
<a href="http://nginx.com/">nginx.com</a>.</p>

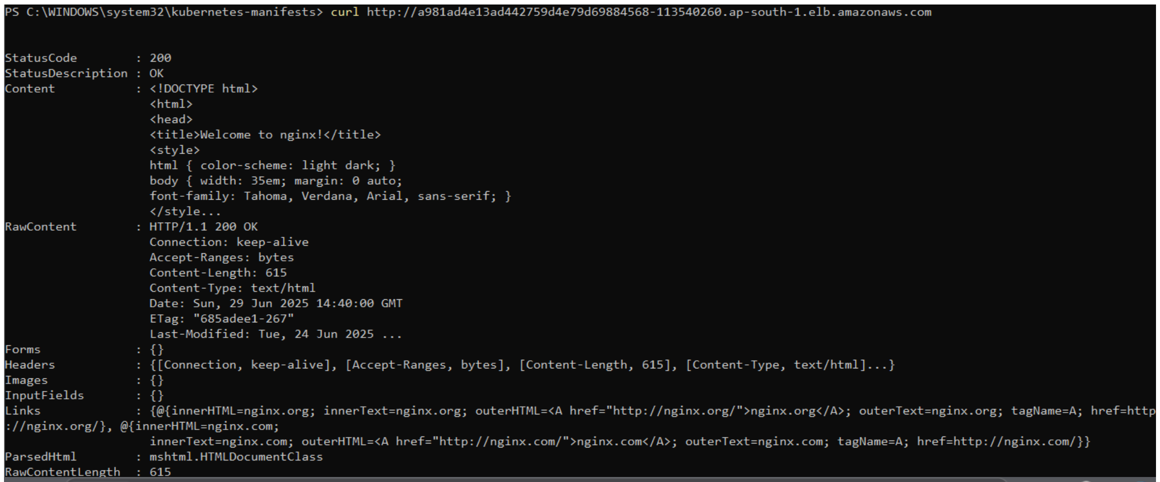
<p><em>Thank you for using nginx.</em></p>
</body>
</html>
~~~

## Part 6: Access Application Using LoadBalancer

**Update `service.yaml` to:**
```yaml
spec:
~~~

```
  type: LoadBalancer
  ports:
    - port: 80
      targetPort: 80
```

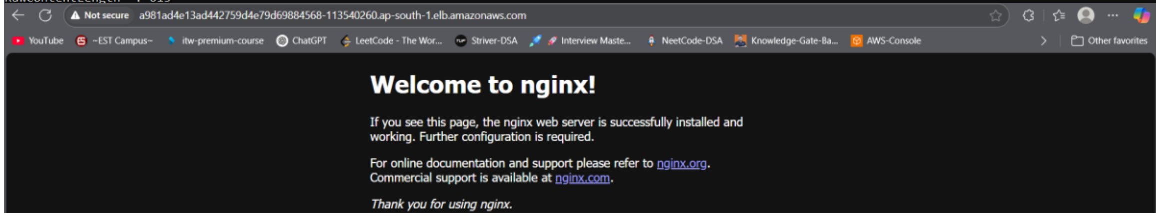
Apply the changes:
```bash
kubectl delete -f service.yaml
kubectl apply -f service.yaml
```

Wait for External IP:
```bash
kubectl get svc
```
Visit in browser:

http://<EXTERNAL-IP>



```
PS C:\WINDOWS\system32\kubernetes-manifests> curl http://a981ad4e13ad442759d4e79d69884568-113540260.ap-south-1.elb.amazonaws.com


StatusCode        : 200
StatusDescription : OK
Content           : <!DOCTYPE html>
                    <html>
                    <head>
                    <title>Welcome to nginx!</title>
                    <style>
                    html { color-scheme: light dark; }
                    body { width: 35em; margin: 0 auto;
                    font-family: Tahoma, Verdana, Arial, sans-serif; }
                    </style...
RawContent        : HTTP/1.1 200 OK
                    Connection: keep-alive
                    Accept-Ranges: bytes
                    Content-Length: 615
                    Content-Type: text/html
                    Date: Sun, 29 Jun 2025 14:40:00 GMT
                    ETag: "685adee1-267"
                    Last-Modified: Tue, 24 Jun 2025 ...
Forms             : {}
Headers           : {[Connection, keep-alive], [Accept-Ranges, bytes], [Content-Length, 615], [Content-Type, text/html]...}
Images            : {}
InputFields       : {}
Links             : {@{innerHTML=nginx.org; innerText=nginx.org; outerHTML=<A href="http://nginx.org/">nginx.org</A>; outerText=nginx.org; tagName=A; href=http
://nginx.org/}, @{innerHTML=nginx.com;
                    innerText=nginx.com; outerHTML=<A href="http://nginx.com/">nginx.com</A>; outerText=nginx.com; tagName=A; href=http://nginx.com/}}
ParsedHtml        : mshtml.HTMLDocumentClass
RawContentLength  : 615
```



Welcome to nginx!

If you see this page, the nginx web server is successfully installed and working. Further configuration is required.

For online documentation and support please refer to nginx.org.
Commercial support is available at nginx.com.

Thank you for using nginx.

Part 7: Installing AWS Load Balancer Controller with Helm

Install Helm (if not done):
```bash
choco install kubernetes-helm
helm version
```

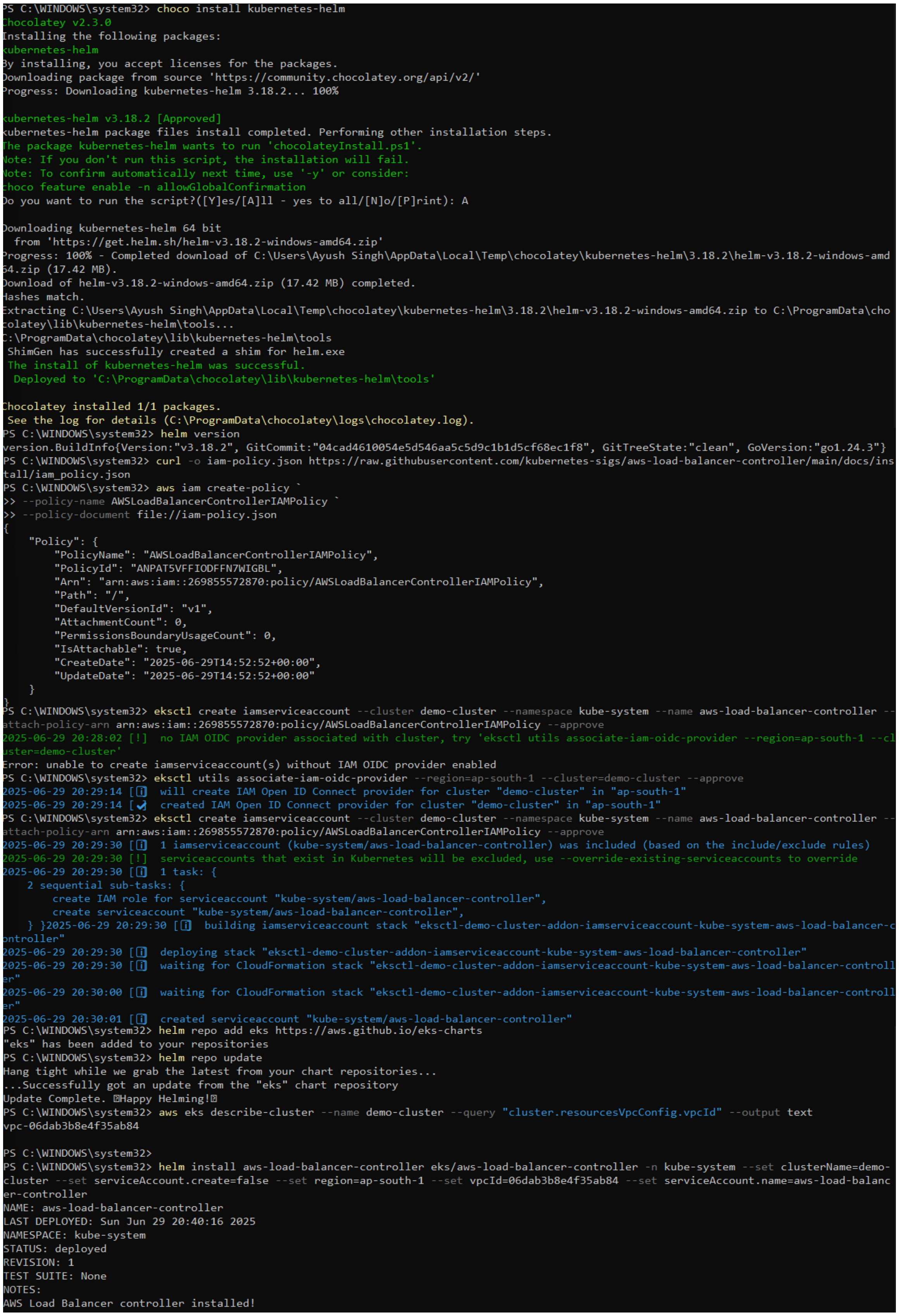
Create IAM Policy:
```bash
curl -o iam-policy.json https://raw.githubusercontent.com/kubernetes-sigs/aws-load-balancer-controller/main/docs/install/iam_policy.json
aws iam create-policy --policy-name AWSLoadBalancerControllerIAMPolicy --policy-document file://iam-policy.json
```

Associate OIDC Provider (if needed):
```bash
eksctl utils associate-iam-oidc-provider --region=ap-south-1 --cluster=demo-cluster --approve
```

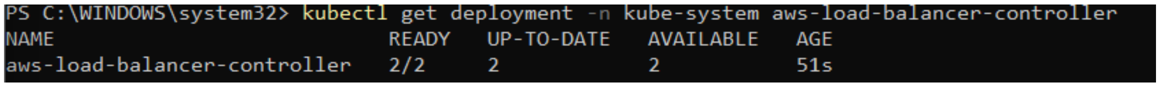
Create IAM Role + Service Account:
```bash
eksctl create iamserviceaccount --cluster demo-cluster --namespace kube-system --name aws-load-balancer-controller --attach-policy-arn arn:aws:iam::<AWS_ACCOUNT_ID>:policy/AWSLoadBalancerControllerIAMPolicy --approve
```

Add Helm Repo & Install Controller:
```bash
helm repo add eks https://aws.github.io/eks-charts
helm repo update
aws eks describe-cluster --name demo-cluster --query "cluster.resourcesVpcConfig.vpcId" --output text
helm install aws-load-balancer-controller eks/aws-load-balancer-controller -n kube-system --set clusterName=demo-cluster --set serviceAccount.create=false --set region=ap-south-1 --set vpcId=<VPC_ID> --set serviceAccount.name=aws-load-balancer-controller
```

Verify Installation:
```bash
kubectl get deployment -n kube-system aws-load-balancer-controller
```



```
PS C:\WINDOWS\system32> choco install kubernetes-helm
hocolatey v2.3.0
Installing the following packages:
kubernetes-helm
By installing, you accept licenses for the packages.
Downloading package from source 'https://community.chocolatey.org/api/v2/'
Progress: Downloading kubernetes-helm 3.18.2... 100%

kubernetes-helm v3.18.2 [Approved]
kubernetes-helm package files install completed. Performing other installation steps.
The package kubernetes-helm wants to run 'chocolateyInstall.ps1'.
Note: If you don't run this script, the installation will fail.
Note: To confirm automatically next time, use '-y' or consider:
choco feature enable -n allowGlobalConfirmation
Do you want to run the script?([Y]es/[A]ll - yes to all/[N]o/[P]rint): A

Downloading kubernetes-helm 64 bit
  from 'https://get.helm.sh/helm-v3.18.2-windows-amd64.zip'
Progress: 100% - Completed download of C:\Users\Ayush Singh\AppData\Local\Temp\chocolatey\kubernetes-helm\3.18.2\helm-v3.18.2-windows-amd
64.zip (17.42 MB).
Download of helm-v3.18.2-windows-amd64.zip (17.42 MB) completed.
Hashes match.
Extracting C:\Users\Ayush Singh\AppData\Local\Temp\chocolatey\kubernetes-helm\3.18.2\helm-v3.18.2-windows-amd64.zip to C:\ProgramData\cho
colatey\lib\kubernetes-helm\tools...
C:\ProgramData\chocolatey\lib\kubernetes-helm\tools
 ShimGen has successfully created a shim for helm.exe
 The install of kubernetes-helm was successful.
  Deployed to 'C:\ProgramData\chocolatey\lib\kubernetes-helm\tools'

Chocolatey installed 1/1 packages.
 See the log for details (C:\ProgramData\chocolatey\logs\chocolatey.log).
PS C:\WINDOWS\system32> helm version
version.BuildInfo{Version:"v3.18.2", GitCommit:"04cad4610054e5d546aa5c5d9c1b1d5cf68ec1f8", GitTreeState:"clean", GoVersion:"go1.24.3"}
PS C:\WINDOWS\system32> curl -o iam-policy.json https://raw.githubusercontent.com/kubernetes-sigs/aws-load-balancer-controller/main/docs/ins
tall/iam_policy.json
PS C:\WINDOWS\system32> aws iam create-policy `
>> --policy-name AWSLoadBalancerControllerIAMPolicy `
>> --policy-document file://iam-policy.json
{
    "Policy": {
        "PolicyName": "AWSLoadBalancerControllerIAMPolicy",
        "PolicyId": "ANPAT5VFFIODFFN7WIGBL",
        "Arn": "arn:aws:iam::269855572870:policy/AWSLoadBalancerControllerIAMPolicy",
        "Path": "/",
        "DefaultVersionId": "v1",
        "AttachmentCount": 0,
        "PermissionsBoundaryUsageCount": 0,
        "IsAttachable": true,
        "CreateDate": "2025-06-29T14:52:52+00:00",
        "UpdateDate": "2025-06-29T14:52:52+00:00"
    }
}
PS C:\WINDOWS\system32> eksctl create iamserviceaccount --cluster demo-cluster --namespace kube-system --name aws-load-balancer-controller --
attach-policy-arn arn:aws:iam::269855572870:policy/AWSLoadBalancerControllerIAMPolicy --approve
2025-06-29 20:28:02 [!]  no IAM OIDC provider associated with cluster, try 'eksctl utils associate-iam-oidc-provider --region=ap-south-1 --cl
uster=demo-cluster'
Error: unable to create iamserviceaccount(s) without IAM OIDC provider enabled
PS C:\WINDOWS\system32> eksctl utils associate-iam-oidc-provider --region=ap-south-1 --cluster=demo-cluster --approve
2025-06-29 20:29:14 [ℹ]  will create IAM Open ID Connect provider for cluster "demo-cluster" in "ap-south-1"
2025-06-29 20:29:14 [✔]  created IAM Open ID Connect provider for cluster "demo-cluster" in "ap-south-1"
PS C:\WINDOWS\system32> eksctl create iamserviceaccount --cluster demo-cluster --namespace kube-system --name aws-load-balancer-controller --
attach-policy-arn arn:aws:iam::269855572870:policy/AWSLoadBalancerControllerIAMPolicy --approve
2025-06-29 20:29:30 [ℹ]  1 iamserviceaccount (kube-system/aws-load-balancer-controller) was included (based on the include/exclude rules)
2025-06-29 20:29:30 [!]  serviceaccounts that exist in Kubernetes will be excluded, use --override-existing-serviceaccounts to override
2025-06-29 20:29:30 [ℹ]  1 task: {
    2 sequential sub-tasks: {
        create IAM role for serviceaccount "kube-system/aws-load-balancer-controller",
        create serviceaccount "kube-system/aws-load-balancer-controller",
    } }2025-06-29 20:29:30 [ℹ]  building iamserviceaccount stack "eksctl-demo-cluster-addon-iamserviceaccount-kube-system-aws-load-balancer-c
ontroller"
2025-06-29 20:29:30 [ℹ]  deploying stack "eksctl-demo-cluster-addon-iamserviceaccount-kube-system-aws-load-balancer-controller"
2025-06-29 20:29:30 [ℹ]  waiting for CloudFormation stack "eksctl-demo-cluster-addon-iamserviceaccount-kube-system-aws-load-balancer-controll
er"
2025-06-29 20:30:00 [ℹ]  waiting for CloudFormation stack "eksctl-demo-cluster-addon-iamserviceaccount-kube-system-aws-load-balancer-controll
er"
2025-06-29 20:30:01 [ℹ]  created serviceaccount "kube-system/aws-load-balancer-controller"
PS C:\WINDOWS\system32> helm repo add eks https://aws.github.io/eks-charts
"eks" has been added to your repositories
PS C:\WINDOWS\system32> helm repo update
Hang tight while we grab the latest from your chart repositories...
...Successfully got an update from the "eks" chart repository
Update Complete. ⎈Happy Helming!⎈
PS C:\WINDOWS\system32> aws eks describe-cluster --name demo-cluster --query "cluster.resourcesVpcConfig.vpcId" --output text
vpc-06dab3b8e4f35ab84

PS C:\WINDOWS\system32>
PS C:\WINDOWS\system32> helm install aws-load-balancer-controller eks/aws-load-balancer-controller -n kube-system --set clusterName=demo-
cluster --set serviceAccount.create=false --set region=ap-south-1 --set vpcId=06dab3b8e4f35ab84 --set serviceAccount.name=aws-load-balanc
er-controller
NAME: aws-load-balancer-controller
LAST DEPLOYED: Sun Jun 29 20:40:16 2025
NAMESPACE: kube-system
STATUS: deployed
REVISION: 1
TEST SUITE: None
NOTES:
AWS Load Balancer controller installed!
```



```
PS C:\WINDOWS\system32> kubectl get deployment -n kube-system aws-load-balancer-controller
NAME                           READY   UP-TO-DATE   AVAILABLE   AGE
aws-load-balancer-controller   2/2     2            2           51s
```

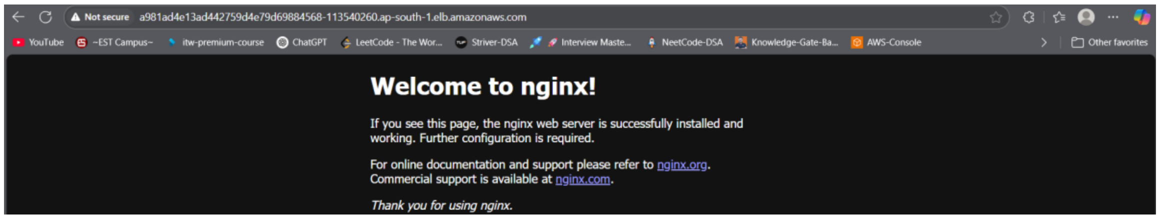
Part 8: Optional – Ingress with Custom Domain

Ingress YAML Example:
```yaml
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  name: nginx-ingress
  annotations:
    alb.ingress.kubernetes.io/scheme: internet-facing
spec:
  rules:
    - host: yourdomain.in
      http:
        paths:
          - path: /
            pathType: Prefix
            backend:
              service:
                name: nginx-service
                port:
                  number: 80
```

Apply Ingress:
```bash
kubectl apply -f ingress.yaml
kubectl get ingress
```

DNS Configuration (GoDaddy or Route 53):
- Add CNAME pointing to ALB hostname (shown in `kubectl get ingress`)

Verify in browser:
```
http://yourdomain.in
```



Welcome to nginx!

If you see this page, the nginx web server is successfully installed and working. Further configuration is required.

For online documentation and support please refer to nginx.org.
Commercial support is available at nginx.com.

Thank you for using nginx.

--x---