

Deploy Nginx Ingress Controller on AWS EKS using Terraform and Helm

Overview

This document outlines the process of setting up an NGINX Ingress Controller on an AWS EKS cluster using **Terraform** and **Helm**. The NGINX Ingress Controller manages external access to services within the Kubernetes cluster and helps route traffic to the appropriate services based on defined rules.

Prerequisites

Before starting, ensure you have the following tools and services installed:

1. **Terraform** ($\geq 1.6.0$)
2. **AWS CLI** (configured with IAM permissions)
3. **kubectl** (Kubernetes CLI)
4. **Helm** (Package manager for Kubernetes)
5. **An AWS EKS Cluster** (already set up)

Installation Instructions

Follow these steps to install the required dependencies on an Ubuntu-based system.

1. Install Terraform

```
```bash
```

```
Update and upgrade system packages
```

```
sudo apt update && sudo apt upgrade -y
```

```
Install necessary dependencies
```

```
sudo apt install -y gnupg software-properties-common curl unzip
```

```
Add HashiCorp GPG key and repository
```

```
curl -fsSL https://apt.releases.hashicorp.com/gpg | sudo gpg --dearmor -o
/usr/share/keyrings/hashicorp-archive-keyring.gpg
```

```
echo "deb [signed-by=/usr/share/keyrings/hashicorp-archive-keyring.gpg]
https://apt.releases.hashicorp.com $(lsb_release -cs) main" | sudo tee
/etc/apt/sources.list.d/hashicorp.list
```

# Update package lists and install Terraform

```
sudo apt update && sudo apt install -y terraform
```

```
...
```

#### 2. Install kubectl

```
```bash
```

```
curl -LO "https://dl.k8s.io/release/$(curl -L -s  
https://dl.k8s.io/release/stable.txt)/bin/linux/amd64/kubectl"
```

```
sudo chmod +x kubectl
```

```
sudo mv kubectl /usr/local/bin/
```

```
sudo kubectl version --client
```

```
...
```

3. Install Helm

```
```bash
```

```
curl -fsSL https://raw.githubusercontent.com/helm/helm/main/scripts/get-helm-3 | bash
```

```
sudo helm version
```

```
...
```

#### 4. Install AWS CLI

```
```bash
```

```
curl "https://awscli.amazonaws.com/awscli-exe-linux-x86_64.zip" -o "awscliv2.zip"
```

```
unzip awscliv2.zip
```

```
sudo ./aws/install
```

```
```
```

To automate the installation, you can use the provided `scripts.sh` script.

```
Session ID: sandeep-7ehgc3hojz8t56dxt6gy Instance ID: i-06f5b27a2e4e078a

$ sudo su ubuntu
To run a command as administrator (user "root"), use "sudo <command>".
See "man sudo_root" for details.

ubuntu@ip-172-31-11-161:~$ /var/snap/amazon-ssm-agent/9881$ cd
ubuntu@ip-172-31-11-161:~$
ubuntu@ip-172-31-11-161:~$ vi scripts.sh
ubuntu@ip-172-31-11-161:~$ vi scripts.sh
ubuntu@ip-172-31-11-161:~$ sh -x scripts.sh
sudo apt update
Get:1 http://eu-north-1.ec2.archive.ubuntu.com/ubuntu noble InRelease [126 kB]
Get:2 http://eu-north-1.ec2.archive.ubuntu.com/ubuntu noble-updates InRelease [126 kB]
Get:3 http://eu-north-1.ec2.archive.ubuntu.com/ubuntu noble-backports InRelease [126 kB]
Get:4 http://security.ubuntu.com/ubuntu noble-security InRelease [126 kB]
Get:5 http://eu-north-1.ec2.archive.ubuntu.com/ubuntu noble/universe amd64 Packages [15.0 MB]
Get:6 http://eu-north-1.ec2.archive.ubuntu.com/ubuntu noble/universe Translation-en [5982 kB]
Get:7 http://security.ubuntu.com/ubuntu noble-security/main amd64 Packages [670 kB]
Get:8 http://eu-north-1.ec2.archive.ubuntu.com/ubuntu noble/universe amd64 Components [3871 kB]
Get:9 http://eu-north-1.ec2.archive.ubuntu.com/ubuntu noble/universe amd64 c-n-f Metadata [301 kB]
Get:10 http://eu-north-1.ec2.archive.ubuntu.com/ubuntu noble/multiverse amd64 Packages [269 kB]
Get:11 http://eu-north-1.ec2.archive.ubuntu.com/ubuntu noble/multiverse Translation-en [118 kB]
Get:12 http://eu-north-1.ec2.archive.ubuntu.com/ubuntu noble/multiverse amd64 Components [35.0 kB]
Get:13 http://eu-north-1.ec2.archive.ubuntu.com/ubuntu noble/multiverse amd64 c-n-f Metadata [8328 B]
Get:14 http://eu-north-1.ec2.archive.ubuntu.com/ubuntu noble-updates/main amd64 Packages [921 kB]
Get:15 http://eu-north-1.ec2.archive.ubuntu.com/ubuntu noble-updates/main Translation-en [289 kB]
Get:16 http://eu-north-1.ec2.archive.ubuntu.com/ubuntu noble-updates/main amd64 Components [151 kB]
Get:17 http://eu-north-1.ec2.archive.ubuntu.com/ubuntu noble-updates/main amd64 c-n-f Metadata [13.4 kB]
Get:18 http://eu-north-1.ec2.archive.ubuntu.com/ubuntu noble-updates/universe amd64 Packages [1040 kB]
Get:19 http://eu-north-1.ec2.archive.ubuntu.com/ubuntu noble-updates/universe Translation-en [262 kB]
Get:20 http://eu-north-1.ec2.archive.ubuntu.com/ubuntu noble-updates/universe amd64 Components [364 kB]
Get:21 http://security.ubuntu.com/ubuntu noble-security/main Translation-en [130 kB]
Get:22 http://security.ubuntu.com/ubuntu noble-security/main amd64 Components [9004 B]
```

```
ubuntu@ip-172-31-11-161:~$ terraform version
Terraform v1.11.2
on linux_amd64
ubuntu@ip-172-31-11-161:~$ aws --version
aws-cli/2.24.26 Python/3.12.9 Linux/6.8.0-1021-aws exe/x86_64.ubuntu.24
ubuntu@ip-172-31-11-161:~$ sudo kubectl version --client
Client Version: v1.32.3
Kustomize Version: v5.5.0
ubuntu@ip-172-31-11-161:~$ sudo helm version
version.BuildInfo{Version:"v3.17.2", GitCommit:"cc0bbbd6d6276b83880042c1ecb34087e84d41eb", GitTreeState:"clean", GoVersion:"go1.23.7"}
ubuntu@ip-172-31-11-161:~$
```

### ### Configure AWS Credentials

Run the following command to configure AWS CLI:

```
```bash
```

```
aws configure
```

```
```
```

```
ubuntu@ip-172-31-11-161:~$ aws configure
AWS Access Key ID [None]:
AWS Secret Access Key [None]:
Default region name [None]: eu-north-1
Default output format [None]:
ubuntu@ip-172-31-11-161:~$
```

## ## Step 1: Create Terraform Configuration Files

### 1. \*\*Create Terraform Directory:\*\*

```
``bash
mkdir -p terraform
cd terraform/
``
```

### 2. \*\*Create Terraform Files:\*\*

- \*\*main.tf:\*\* Define the Virtual Private Cloud (VPC) setup.
- \*\*eks.tf:\*\* Define the AWS EKS cluster configuration.
- \*\*iam.tf:\*\* Define the IAM roles.
- \*\*outputs.tf:\*\* Define outputs for the configuration.
- \*\*variables.tf:\*\* Define variables for the configuration.

### 3. \*\*Initialize Terraform:\*\*

```
``bash
terraform init
``
```

```

ubuntu@ip-172-31-11-161:~$ mkdir -p terraform
ubuntu@ip-172-31-11-161:~$ cd terraform/
ubuntu@ip-172-31-11-161:~/terraform$ vi provider.tf
ubuntu@ip-172-31-11-161:~/terraform$ vi vpc.tf
ubuntu@ip-172-31-11-161:~/terraform$ vi eks.tf
ubuntu@ip-172-31-11-161:~/terraform$ vi node-group.tf
ubuntu@ip-172-31-11-161:~/terraform$ vi outputs.tf
ubuntu@ip-172-31-11-161:~/terraform$ terraform init
Initializing the backend...
Initializing modules...
Downloading registry.terraform.io/terraform-aws-modules/eks/aws 20.0.0 for eks...
- eks in .terraform/modules/eks
- eks.eks_managed_node_group in .terraform/modules/eks/modules/eks-managed-node-group
- eks.eks_managed_node_group.user_data in .terraform/modules/eks/modules/_user_data
- eks.fargate_profile in .terraform/modules/eks/modules/fargate-profile
Downloading registry.terraform.io/terraform-aws-modules/kms/aws 2.1.0 for eks.kms...
- eks.kms in .terraform/modules/eks/kms
- eks.self_managed_node_group in .terraform/modules/eks/modules/self-managed-node-group
- eks.self_managed_node_group.user_data in .terraform/modules/eks/modules/_user_data
Downloading registry.terraform.io/terraform-aws-modules/eks/aws 20.34.0 for eks_managed_node_group...
- eks_managed_node_group in .terraform/modules/eks_managed_node_group/modules/eks-managed-node-group
- eks_managed_node_group.user_data in .terraform/modules/eks_managed_node_group/modules/_user_data
Downloading registry.terraform.io/terraform-aws-modules/vpc/aws 5.0.0 for vpc...
- vpc in .terraform/modules/vpc
Initializing provider plugins...
- Finding hashicorp/null versions matching ">= 3.0.0"...
- Finding hashicorp/aws versions matching ">= 4.33.0, >= 5.0.0, >= 5.34.0, >= 5.83.0"...
- Finding hashicorp/tls versions matching ">= 3.0.0"...
- Finding hashicorp/time versions matching ">= 0.9.0"...
- Finding hashicorp/cloudinit versions matching ">= 2.0.0"...
- Installing hashicorp/aws v5.91.0...
- Installed hashicorp/aws v5.91.0 (signed by HashiCorp)
- Installing hashicorp/tls v4.0.6...
- Installed hashicorp/tls v4.0.6 (signed by HashiCorp)
- Installing hashicorp/time v0.13.0...
- Installed hashicorp/time v0.13.0 (signed by HashiCorp)
- Installing hashicorp/cloudinit v2.3.6...
- Installed hashicorp/cloudinit v2.3.6 (signed by HashiCorp)
- Installing hashicorp/null v3.2.3...
- Installed hashicorp/null v3.2.3 (signed by HashiCorp)
Terraform has created a lock file .terraform.lock.hcl to record the provider
selections it made above. Include this file in your version control repository
so that Terraform can guarantee to make the same selections by default when
you run "terraform init" in the future.

Terraform has been successfully initialized!

```

#### 4. \*\*Plan and Apply the Configuration:\*\*

```
``bash
```

```
terraform plan
```

```
terraform apply -auto-approve
```

```
```
```

```

ubuntu@ip-172-31-11-161:~/terraform$ terraform apply -auto-approve
module.eks.module.eks_managed_node_group["general"].data.aws_caller_identity.current: Reading...
module.eks.module.eks_managed_node_group["general"].data.aws_iam_policy_document.assume_role_policy[0]: Reading...
module.eks.module.eks_managed_node_group["general"].data.aws_partition.current: Reading...
module.eks.data.aws_caller_identity.current: Reading...
module.eks.data.aws_partition.current: Reading...
module.eks.module.kms.data.aws_caller_identity.current[0]: Reading...
module.eks.module.kms.data.aws_partition.current[0]: Reading...
module.eks.data.aws_iam_policy_document.assume_role_policy[0]: Reading...
module.eks.data.aws_iam_policy_document.assume_role_policy[0]: Read complete after 0s [id=27644486067]
module.eks.module.kms.data.aws_partition.current[0]: Read complete after 0s [id=aws]
module.eks.data.aws_partition.current: Read complete after 0s [id=aws]
module.eks.module.kms.data.aws_caller_identity.current[0]: Read complete after 0s [id=794038256791]
module.eks.module.eks_managed_node_group["general"].data.aws_partition.current: Read complete after 0s [id=aws]
module.eks.module.eks_managed_node_group["general"].data.aws_iam_policy_document.assume_role_policy[0]: Read complete after 0s [id=2560088296]
module.eks.module.eks_managed_node_group["general"].data.aws_caller_identity.current: Read complete after 0s [id=794038256791]

```

[illegible]

Step 2: Install NGINX Ingress Controller using Helm

1. ****Create a new directory for NGINX Ingress deployment:****

```
``bash
```

```
mkdir terraform-nginx-ingress && cd terraform-nginx-ingress
```

```
terraform init
```

```
terraform apply -auto-approve
```

///

```
+ set {
+   name = "controller.ingressClassResource.default"
+   value = "true"
+   # (1 unchanged attribute hidden)
+ }
+ set {
+   name = "controller.service.type"
+   value = "LoadBalancer"
+   # (1 unchanged attribute hidden)
+ }
}
```

Plan: 1 to add, 0 to change, 0 to destroy.

```
helm_release.nginx_ingress: Creating...
helm_release.nginx_ingress: Still creating... [10s elapsed]
helm_release.nginx_ingress: Still creating... [20s elapsed]
helm_release.nginx_ingress: Still creating... [30s elapsed]
helm_release.nginx_ingress: Creation complete after 40s [id=nginx-ingress]
```

Apply complete! Resources: 1 added, 0 changed, 0 destroyed.

2. ****Update the kubeconfig file to access the EKS cluster:****

```
```bash
```

```
aws eks update-kubeconfig --region eu-north-1 --name stylish-threads-cluster
```

```
```
```

The output will confirm the addition of a new context to the kubeconfig file:

```
```
```

Added new context arn:aws:eks:eu-north-1:794038256791:cluster/stylish-threads-cluster to /home/ubuntu/.kube/config

```
```
```

3. ****Get a token for accessing the EKS cluster:****

```
```bash
```

```
aws eks get-token --cluster-name stylish-threads-cluster
```

```
```
```

4. ****Verify Cluster Access:****

```
```bash
```

```
kubectl get nodes
```

```
kubectl get svc -n ingress-nginx
```

```
```
```

```
ubuntu@ip-172-31-11-161:~/project4/terraform/terraform-ingress-nginx$ kubectl get nodes
NAME                                STATUS    ROLES    AGE   VERSION
ip-10-0-1-155.eu-north-1.compute.internal Ready    <none>   14m   v1.20.15-eks-a5c579
ip-10-0-2-249.eu-north-1.compute.internal Ready    <none>   14m   v1.20.15-eks-a5c579
ubuntu@ip-172-31-11-161:~/project4/terraform/terraform-ingress-nginx$ kubectl get svc -n ingress-nginx
NAME                                TYPE                CLUSTER-IP      EXTERNAL-IP
nginx-ingress-ingress-nginx-controller LoadBalancer       172.20.234.217   a69714e10eaa84dba905578400eaa8fc-1289960659.eu-north-1.elb.amazonaws.com
nginx-ingress-ingress-nginx-controller-admission ClusterIP           172.20.17.150   <none>
```

At this point, the NGINX Ingress Controller service should be running with an external IP address.

Ingress Controller Service is running – The LoadBalancer has an external IP:

a69714a18eaa84dba905578400eaabfc-1289960659.eu-north-1.elb.amazonaws.com

Ports 80 and 443 are exposed.

Step 3: Test the Ingress Controller

Test the NGINX Ingress Controller by sending an HTTP request to the external IP address.

```
``bash
```

```
curl -I http://a69714a18eaa84dba905578400eaabfc-1289960659.eu-north-1.elb.amazonaws.com
```

```
``
```

You should see an HTTP 404 (which is normal since no ingress rules are defined yet).

Session ID: sandeep-nu8qayxjgs8y0j466ykegoqcy

Instance ID: i-06f5b827a2e4e078a

```
shuntu@ip-172-31-11-161:~/Project4/terraform/terraform-ingress-nginx$ curl -I http://a69714a18eaa84dba905578400eaabfc-1289960659.eu-north-1.elb.amazonaws.com
HTTP/1.1 404 Not Found
Date: Wed, 19 Mar 2025 14:23:57 GMT
Content-Type: text/html
Content-Length: 146
Connection: keep-alive
shuntu@ip-172-31-11-161:~/Project4/terraform/terraform-ingress-nginx$
```

Step 4: Deploy an NGINX Application Behind the Ingress Controller

1. Deploy NGINX Application

Create a deployment and service for the NGINX application.

```
**nginx-deployment.yaml:**
```

Apply the YAML file to create the deployment and service:


```
```bash
```

```
kubectl apply -f nginx-deployment.yaml
```

```
```
```

2. Create an Ingress Resource

Now, create an Ingress resource to route traffic to the NGINX service.

```
**nginx-ingress.yaml:**
```

Apply the Ingress resource:

```
```bash
```

```
kubectl apply -f nginx-ingress.yaml
```

```
```
```

```
ubuntu@ip-172-31-11-161:~/Project4/terraform$ vi nginx.yaml
ubuntu@ip-172-31-11-161:~/Project4/terraform$ kubectl apply -f nginx-deployment.yaml
error: the path "nginx-deployment.yaml" does not exist
ubuntu@ip-172-31-11-161:~/Project4/terraform$ kubectl apply -f nginx.yaml
deployment.apps/nginx-app created
service/nginx-service created
ubuntu@ip-172-31-11-161:~/Project4/terraform$ vi nginx-ingress.yaml
ubuntu@ip-172-31-11-161:~/Project4/terraform$ kubectl apply -f nginx-ingress.yaml
ingress.networking.k8s.io/nginx-ingress created
ubuntu@ip-172-31-11-161:~/Project4/terraform$
```

Step 5: Verify Deployment

To verify the deployment, you can use the following commands:

1. ****Test the Ingress:****

```
```bash
```

```
curl -I http://a69714a18eaa84dba905578400eaabfc-1289960659.eu-north-1.elb.amazonaws.com/
```

...

```
ubuntu@ip-172-31-11-161:~$ kubectl get ingress
NAME CLASS HOSTS ADDRESS PORTS AGE
nginx-ingress nginx * a69714a18eaa84dba905578400eaabfc-1289960659.eu-north-1.elb.amazonaws.com 80 18m
ubuntu@ip-172-31-11-161:~$ curl -I http://a69714a18eaa84dba905578400eaabfc-1289960659.eu-north-1.elb.amazonaws.com/
HTTP/1.1 200 OK
Date: Wed, 19 Mar 2025 14:47:15 GMT
Content-Type: text/html
Content-Length: 615
Connection: keep-alive
Last-Modified: Wed, 05 Feb 2025 11:06:32 GMT
ETag: "Grs34638-267"
Accept-Ranges: bytes
ubuntu@ip-172-31-11-161:~$
```

## 2. \*\*Access via Browser:\*\*

Open a web browser and navigate to the external IP address:

...

<http://a69714a18eaa84dba905578400eaabfc-1289960659.eu-north-1.elb.amazonaws.com/>

...

You should see the default NGINX welcome page if everything is set up correctly.



## ## Conclusion

This guide has provided detailed instructions to deploy the NGINX Ingress Controller on an AWS EKS cluster using Terraform and Helm. With the Ingress Controller successfully deployed, you can now manage external access to your Kubernetes services and route traffic based on defined Ingress rules.

...