Pre-Requisites

1. Terraform

For Mac Os

- You can use Homebrew

$ brew install terraform

For Windows

- You can use Chocolatey (https://chocolatey.org/install)

$ choco install terraform

You can look for instructions in here:

https://learn.hashicorp.com/terraform/getting-started/install

2. AWS CLI

For MacOS

$ curl "https://awscli.amazonaws.com/AWSCLIV2.pkg" -o "AWSCLIV2.pkg"

$ sudo installer -pkg AWSCLIV2.pkg -target /

Verify Installation:

$ which aws

$ aws --version

For Windows

https://awscli.amazonaws.com/AWSCLIV2.msi

To confirm the installation, open the Start menu, search for cmd to open a command prompt window, and at the command prompt use the aws --version command.

3. AWS IAM Authenticator

For MacOS

$ brew install aws-iam-authenticator

For Windows

$ choco install aws-iam-authenticator

4. Kubectl

For MacOS

$ brew install kubernetes-cli

For Windows

$ choco install kubernetes-cli

Step 1.

After you've installed the AWS CLI, configure it by running aws configure.

When prompted, enter your AWS Access Key ID, Secret Access Key, region and output format.

$ aws configure

AWS Access Key ID [None]: **YOUR_AWS_ACCESS_KEY_ID**

AWS Secret Access Key [None]: **YOUR_AWS_SECRET_ACCESS_KEY**

Default region name [None]: **YOUR_AWS_REGION**

Default output format [None]: json

```
SJCMAC17JJHD4:EKS-Terraform lcastro$ aws configure
AWS Access Key ID [****************RDYC]: AKIARYHDXUF3ZDKJLZ2A
AWS Secret Access Key [****************VQ+a]: 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
Default region name [us-east-1]:
Default output format [json]:
```

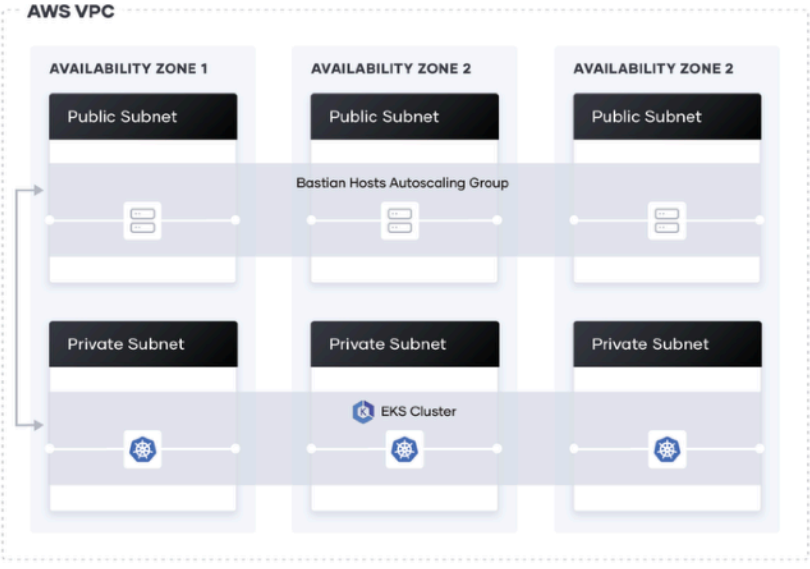
Step 2. Download the Terraform Files from GitHub

In your terminal, clone the following repository. It contains the example configuration used in this guide.

$ git clone https://github.com/lcastrose/EKS-Terraform.git

In here, you will find six files used to provision a VPC, security groups and an EKS cluster. The final product should be similar to this:

$ cd EKS-Terraform



Step 3. Inside the EKS-Terraform, modify the **vpc.tf** file according your specific data

Modify the file using **vi** or **vim**

```
SJCMAC17JJHD4:EKS-Terraform lcastro$ ls
README.md
eks-cluster.tf
kubernetes-dashboard-admin.rbac.yaml
outputs.tf
security-groups.tf
versions.tf
vpc.tf
```

Region, VPC Name, CIDR, Private Subnets, Public Subnets

```
variable "region" {
  default     = "us-east-2"
  description = "AWS region"
}

provider "aws" {
  version = ">= 2.28.1"
  region  = "us-east-2"
}

data "aws_availability_zones" "available" {}

locals {
  cluster_name = "training-eks-${random_string.suffix.result}"
}

resource "random_string" "suffix" {
  length  = 8
  special = false
}

module "vpc" {
  source  = "terraform-aws-modules/vpc/aws"
  version = "2.6.0"

  name                 = "training-vpc"
  cidr                 = "10.0.0.0/16"
  azs                  = data.aws_availability_zones.available.names
  private_subnets      = ["10.0.1.0/24", "10.0.2.0/24", "10.0.3.0/24"]
  public_subnets       = ["10.0.4.0/24", "10.0.5.0/24", "10.0.6.0/24"]
  enable_nat_gateway   = true
  single_nat_gateway   = true
  enable_dns_hostnames = true

  tags = {
    "kubernetes.io/cluster/${local.cluster_name}" = "shared"
  }

  public_subnet_tags = {
    "kubernetes.io/cluster/${local.cluster_name}" = "shared"
    "kubernetes.io/role/elb"                      = "1"
  }

  private_subnet_tags = {
    "kubernetes.io/cluster/${local.cluster_name}" = "shared"
    "kubernetes.io/role/internal-elb"             = "1"
  }
}
```

Step 4. Terraform Init

Once you have cloned the repository, initialize your Terraform workspace, which will download and configure the providers.

$ terraform init

```
SJCMAC17JJHD4:EKS-Terraform lcastro$ terraform init
Initializing modules...
Downloading terraform-aws-modules/eks/aws 12.0.0 for eks...
- eks in .terraform/modules/eks/terraform-aws-eks-12.0.0
- eks.node_groups in .terraform/modules/eks/terraform-aws-eks-12.0.0/modules/node_groups
Downloading terraform-aws-modules/vpc/aws 2.6.0 for vpc...
- vpc in .terraform/modules/vpc/terraform-aws-vpc-2.6.0

Initializing the backend...

Initializing provider plugins...
- Checking for available provider plugins...
- Downloading plugin for provider "aws" (hashicorp/aws) 2.64.0...
- Downloading plugin for provider "kubernetes" (hashicorp/kubernetes) 1.11.3...
- Downloading plugin for provider "random" (hashicorp/random) 2.2.1...
- Downloading plugin for provider "local" (hashicorp/local) 1.4.0...
- Downloading plugin for provider "null" (hashicorp/null) 2.1.2...
- Downloading plugin for provider "template" (hashicorp/template) 2.1.2...

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.
```

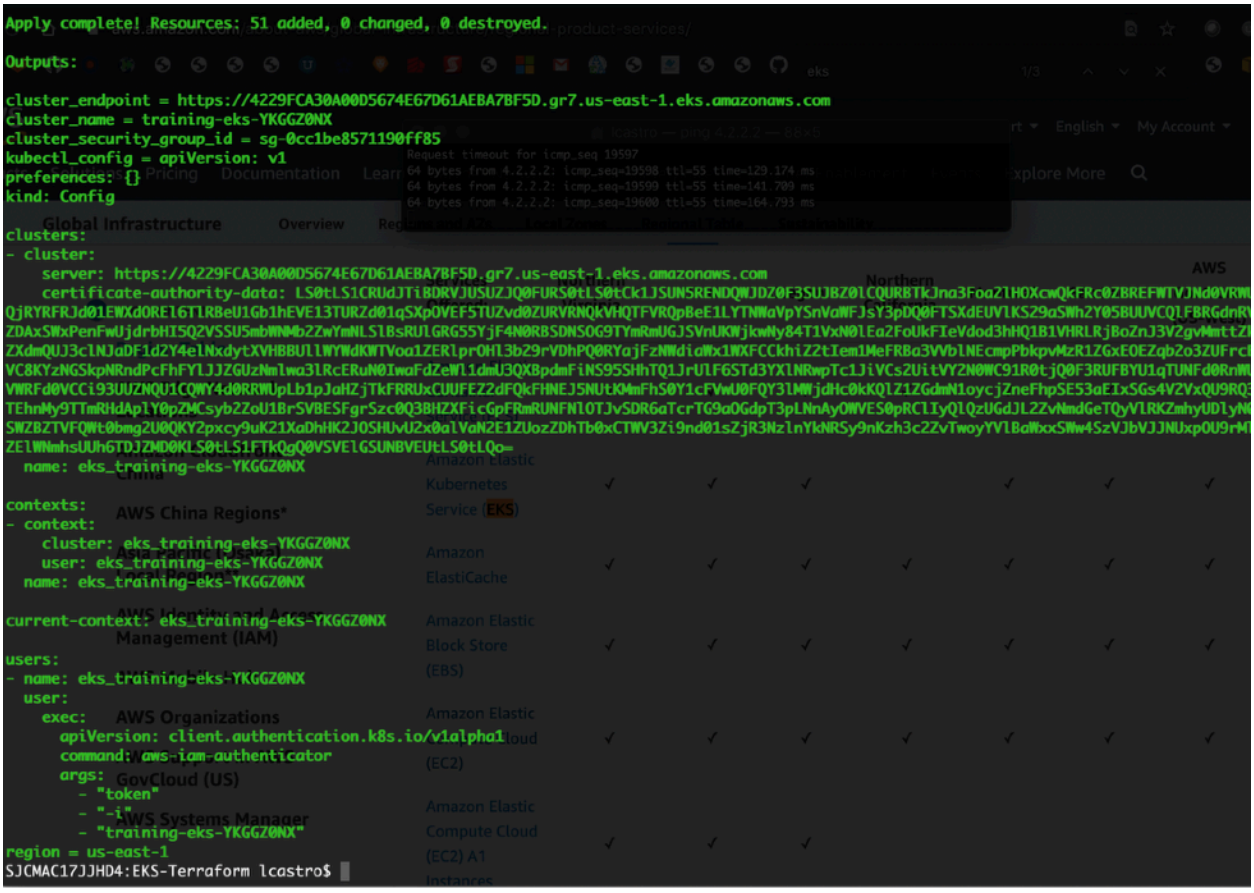
Step 4. Provision the EKS cluster (**It will take around 10 minutes to be deployed**)

In your initialized directory, run terraform apply and review the planned actions.

Your terminal output should indicate the plan is running and what resources will be created.

$ terraform apply

You can see this terraform apply will provision a total of 51 resources (VPC, Security Groups, AutoScaling Groups, EKS Cluster, etc...). If you're comfortable with this, confirm the run with a yes.



Step 5. Configure Kubectl

Now that you've provisioned your EKS cluster, you need to configure kubectl. Customize the following command with your cluster name and region, the values from Terraform's output. It will get the access credentials for your cluster and automatically configure kubectl.

$ aws eks --region *us-east-2* update-kubeconfig --name *training-eks-sR8eLIil*

The Kubernetes cluster name and region correspond to the output variables showed after the successful Terraform run.

```
SJCMAC17JJHD4:EKS-Terraform lcastro$ aws eks --region us-east-1 update-kubeconfig --name training-eks-YKGGZ0NX
Added new context arn:aws:eks:us-east-1:120736686455:cluster/training-eks-YKGGZ0NX to /Users/lcastro/.kube/config
```

Step 6. Validate running Nodes

$ kubectl get nodes

```
[SJCMAC17JJHD4:EKS-Terraform lcastro$ kubectl get nodes
NAME                           STATUS   ROLES    AGE   VERSION
ip-10-20-1-206.ec2.internal    Ready    <none>   15m   v1.16.8-eks-e16311
ip-10-20-1-56.ec2.internal     Ready    <none>   15m   v1.16.8-eks-e16311
ip-10-20-3-142.ec2.internal    Ready    <none>   15m   v1.16.8-eks-e16311
```

Step 7. Destroy EKS Cluster (DO NOT PERFORM THIS STEP UNTIL REQUIRED)

(It takes about 10 minutes to destroy all resources)

$ terraform destroy

```
[SJCMAC17JJHD4:EKS-Terraform lcastro$ terraform destroy
random_string.suffix: Refreshing state... [id=YKGGZ0NX]
module.eks.data.aws_caller_identity.current: Refreshing state...
module.eks.data.aws_ami.eks_worker_windows: Refreshing state...
data.aws_availability_zones.available: Refreshing state...
module.eks.data.aws_partition.current: Refreshing state...
```