



BATCH :

LESSON :

DATE :

SUBJECT :

TERRAFORM

**MODULES &
DEMO PROJECT 2**



techproeducation



techproeducation



techproeducation



techproeducation



techproedu



techproeducation.com



info@techproeducation.com



+1 (917) 768-7466



MODULES

- **Create “webserver” module**
- Oluşturduğumuz webserver klasörünün içini modüllerle dolduralım.
- entry-script.sh I modules/webserver in içine taşıyalım.
- weserver/variables.tf i aç ve şu variable ları içine oluşturalım.

```
variable vpc_id{}  
variable my_ip{}  
variable env_prefix{}  
variable image_name{}  
variable public_key_location{}  
variable instance_type{}  
variable subnet_id{}  
variable avail_zone{}  
variable env_prefix{}
```



MODULES

- Create “webserver” module
- terraform.tfvars a aşağıdaki resimdeki gibi kodu ekleyelim:

```
image_name = "amzn-ami-hvm-*x86_64-gp2"
```

- Root output.tf ‘ı aşağıdaki gibi güncelleyelim:

```
output "ec2_public_ip" {  
    value = module.myapp-server.instance.public_ip  
}
```



MODULES

- Create “webserver” module
- modules/webserver output.tf i aşağıdaki gibi güncelleyelim:

```
output "instance" {  
    value = value = aws_instance.myapp-server  
}
```



DEMO PROJECT-2

Terraform & AWS EKS

- Bu kısımda Terraform kullanarak EKS Cluster oluşturacağız. Bunun için

1. Control Planı oluşturma.

- * `mycluster.eks.aws.amazon.com`
- * Master Nodes



Amazon EKS

2. VPC Oluşturma

- * Worker Node ların çalışması için VPC leri oluşturuyoruz.
 - * EC2 Instances c. Node Group
- Cluster ı spesifik bir region da oluşturmamız gerekiyor(Bizim durumumuzda us-east-1)



DEMO PROJECT-2

- VPC
- vpc.tf isimli bir dosya oluşturalım:

```
# Değişkenler
variable vpc_cidr_block{}
variable private_subnet_cidr_blocks{}
variable public_subnet_cidr_blocks{}

module "myapp-vpc" {
  source = "terraform-aws-modules/vpc/aws"
  version = 3.11.0
  # insert the 21 required variables here

  name = "myapp-vpc"
  cidr = var.vpc_cidr_block
  # Best Practice: Her bir availability zone için bir private bir de public subnet oluşturun;
  private_subnets = var.private_subnet_cidr_blocks
  public_subnets = var.public_subnet_cidr_blocks
}
```



DEMO PROJECT-2

- **VPC**
- terraform.tfvars isimli bir dosya daha oluşturalım:

```
vpc_cidr_block = "10.0.0.0/16"
```

- Region'umuzda kaç Availability Zone(AZ) varsa hepsine bir private bir de public subnet cidr block atamamız gerekiyor. Bizim durumumuzda 6 adet AZ var. Bu noktada 6 private 6 public olmak üzere toplamda 12 adet subnet cidr block oluşturmamız AZ ye bu subnet leri dağıtmamız gerekiyor.Hatta bunu dinamic olarakta tanımlayabiliriz. Bunun için:

```
# (private ve public) subnet_cidr_blocks vpc_cidr_blocks 'un parçası olmalı  
private_subnet_cidr_blocks = ["10.0.1.0/24", "10.0.2.0/24", "10.0.3.0/24", "10.0.4.0/24", "10.0.5.0/24", "10.0.6.0/24"]  
public_subnet_cidr_blocks = ["10.0.7.0/24", "10.0.8.0/24", "10.0.9.0/24", "10.0.10.0/24", "10.0.11.0/24", "10.0.12.0/24"]
```



DEMO PROJECT-2

- VPC

```
provider "aws" { # YENİ
  # data 'nın bağımlı olduğu provider
  region = "us-east-2" # YENİ
} # YENİ

variable vpc_cidr_block{}
variable private_subnet_cidr_blocks{}
variable public_subnet_cidr_blocks{}

data "aws_availability_zones" "azs" {} # bu data AZ leri query leyecek. İsmi azs.
# "aws_availability_zones" "aws" provider ına bağlı olduğu için "aws" providerını da belirtmeliyiz.

module "myapp-vpc" {
  source = "terraform-aws-modules/vpc/aws"
  version = "3.11.0"

  name = "myapp-vpc" # Resource daki Name Sütunu
  cidr = var.vpc_cidr_block # Burada hard codingtense variable ları kullanacağız. cidr block umuzu tanımlamıştık zaten önceden.
```




DEMO PROJECT-2

- VPC

```
# Best Practice: Her bir availability zone için bir private bir de public subnet oluşturun;  
# myapp-vpc modülünün içinde subnetler zaten çoktan tanımlı. Biz kaç tane subnet oluşturulacağını, hangi subnetlerin oluşturulacağını;  
# EKS için de best practice her bir Availability Zone için bir adet private bir adette public subnet oluşturmak.  
# Bizim regionumuzda buradan da(https://console.aws.amazon.com/ec2/v2/home?region=us-east-1#Home) görüleceği üzere 6 adet Availability Zone var.  
private_subnets = var.private_subnet_cidr_blocks  
public_subnets = var.public_subnet_cidr_blocks  
# Variable'ları yukarıya tanımladık.  
# Tanımladığımız variable ların değerlerini de terraform.tfvars a atadık.  
azs = data.aws_availability_zones.azs.names # dinamik tanımlama için variablelar ile module "myapp-vpc" nin arasına "azs" isimli variable ekledik.  
# data.aws_availability_zones un names attribute u olduğunu nereden biliyoruz peki ? Buradan arkadaşlar: https://registry.terraform.io/providers/hashicorp/aws/latest/docs/data-sources/availability\_zones  
  
enable_nat_gateway = true # default değeri her bir subnet için bir NAT gateway i. Transparency purposes için true ya set ettik.  
single_nat_gateway = true # bütün private subnetler single_nat_gateway aracılığıyla internet trafiklerini yönlendirirler.  
enable_dns_hostnames = true # bununla da oluşturacağımız EC2 instance ını public & private dns hostnames e de atayacağız.  
  
# Tag lerimizi de ekleyelim:  
# Tag lerimiz kolay okunma faydasının yanı sıra Cloud Controller Manager Identifier'a hangi resource ile bağlantıda olması gere
```



DEMO PROJECT-2

- VPC

```
# Aşağıdaki tag leri tanımlamak zorunlu
tags = { # VPC için
# myapp-eks-cluster cluster adı olsun.
"kubernetes.io/cluster/myapp-eks-cluster" = "shared" # value = shared
}

public_subnet_tags = { # public_subnets için
# myapp-eks-cluster cluster adı olsun.
"kubernetes.io/cluster/myapp-eks-cluster" = "shared" # value = shared
"kubernetes.io/role/elb" = 1 # elb = elastic load balancer
# Kubernetes te load balancer service oluşturduğumuzda service'e cloud native load balancer atar.
}

private_subnet_tags = { # private_subnets için
# myapp-eks-cluster cluster adı olsun.
"kubernetes.io/cluster/myapp-eks-cluster" = "shared" # value = shared
"kubernetes.io/role/internal-elb" = 1 # Private subnet internete kapalıdır.
}
}
```



DEMO PROJECT-2

- **Terraform & AWS EKS**
- eks-cluster.tf isimli yeni bir dosya oluşturalım. Bu sefer eks module u kullanacağız. <https://registry.terraform.io/modules/terraform-aws-modules/eks/aws/latest> a gidelim ve Dependencies'i açalımç



Amazon EKS

- Bu adımda kubernetes provider l yapılandırmalıyız. Gördüğünüz gibi bu module kubernetese bağımlı.Bunu da takip eden slaytlarda yapacağız:



DEMO PROJECT-2

- Terraform & AWS EKS

• eks-cluster.tf

```
provider "kubernetes" { # 8)

    host = data.aws_eks_cluster.myapp-cluster.endpoint # (Aşağısı) Endpoint of K8 cluster(API Server) host'umuza query lerek ulaşalım

    # Actual Authentication Data
    token = data.aws_eks_cluster_auth.myapp-cluster.token # token ları içeren objectleri istiyoruz burada
    cluster_ca_certificate = base64decode(data.aws_eks_cluster.myapp-cluster.certificate_authority.0.data) # certificate_authortity.0
    # base64decode u kullandık çünkü certificate_authority encoded(https://registry.terraform.io/providers/hashicorp/aws/latest/docs/)
}

data "aws_eks_cluster" "myapp-cluster" { # 9)
    name = module.eks.cluster_id # host'a queryleyrek ulaşmak için. Modülümüzü refer ettik.
}

data "aws_eks_cluster_auth" "myapp-cluster" { # 10) token'ı içeren object'i verir bize bu
    name = module.eks.cluster_id # actual authentication data için kullanırız. Aynı methodla query le ama bize bu sefer aws_eks_clust
}

module "eks" {
    source = "terraform-aws-modules/eks/aws"
    version = "17.24.0"
    # insert the 7 required variables here

    cluster_name = "myapp-eks-cluster" # 1) cluster 'ın ismi. Bunu zaten vpc.tf deki tags'te tanımlamıştık.
    cluster_version = "1.21" # 2) kubernetes version

    # Subnet Attributes. Worker Noda ların başlamasını istediğimiz subnetler
    subnets = module.myapp-vpc.private_subnets # 3) Burada myapp-vpc modulündeki private_subnets'e refer ediyoruz. Hatırlayalım ki priv
    vpc_id = module.myapp-vpc.vpc_id # 7) Burada da myapp-vpc deki vpc_id ye refer ediyoruz.
}
```



DEMO PROJECT-2

- Terraform & AWS EKS

```
# 4) tags. eks cluster için gerekli taglarımız ok, yani tanımlamazsak hata almayız.
tags = {
  environment = "deployment" # 5)
  application = "myapp" # 6) hangi application için
}

# Worker Nodes. Burada da hangi tür worker node larla çalışmak istediğimizi belirtelim.
# 8
worker_groups = [ #Takes array of worker node config objects as an input. Yani içerisine birden fazla worker node tanımlayabiliriz.
  {
    instance_type = "t2.small" # t2.small türünde
    name = "worker-group-1" # isimli
    asg_desired_capacity = 2 # 2 adet worker nodes istiyoruz.
  }, # ne zaman bir eks cluster oluştursak, AWS bize oluşturduğumuz her bir cluster için ayrı bir fatura çıkartır. - Saatlik 10 c
  {
    instance_type = "t2.medium" # t2.medium türünde
    name = "worker-group-2" # isimli
    asg_desired_capacity = 1 # 1 adet worker node istiyoruz.
  }
]
}
```