CSDO1010 Assignment 7

Paul Krznaric

# Repo:

<https://github.com/PaulKrznaric/07.Lab7-K8s>

# Proof:

## Terraform Apply

> terraform apply

Terraform used the selected providers to generate the following execution plan. Resource actions are indicated with the following symbols:

+ create

Terraform will perform the following actions:

# module.compute.aws\_instance.k8s-master will be created

+ resource "aws\_instance" "k8s-master" {

+ ami = (sensitive)

+ arn = (known after apply)

+ associate\_public\_ip\_address = true

+ availability\_zone = (known after apply)

+ cpu\_core\_count = (known after apply)

+ cpu\_threads\_per\_core = (known after apply)

+ get\_password\_data = false

+ host\_id = (known after apply)

+ id = (known after apply)

+ instance\_initiated\_shutdown\_behavior = (known after apply)

+ instance\_state = (known after apply)

+ instance\_type = "t2.micro"

+ ipv6\_address\_count = (known after apply)

+ ipv6\_addresses = (known after apply)

+ key\_name = "k8s"

+ outpost\_arn = (known after apply)

+ password\_data = (known after apply)

+ placement\_group = (known after apply)

+ primary\_network\_interface\_id = (known after apply)

+ private\_dns = (known after apply)

+ private\_ip = (known after apply)

+ public\_dns = (known after apply)

+ public\_ip = (known after apply)

+ secondary\_private\_ips = (known after apply)

+ security\_groups = (known after apply)

+ source\_dest\_check = true

+ subnet\_id = (known after apply)

+ tags = {

+ "Name" = "k8s\_master\_tf"

}

+ tags\_all = {

+ "Name" = "k8s\_master\_tf"

}

+ tenancy = (known after apply)

+ vpc\_security\_group\_ids = (known after apply)

+ capacity\_reservation\_specification {

+ capacity\_reservation\_preference = (known after apply)

+ capacity\_reservation\_target {

+ capacity\_reservation\_id = (known after apply)

}

}

+ ebs\_block\_device {

+ delete\_on\_termination = (known after apply)

+ device\_name = (known after apply)

+ encrypted = (known after apply)

+ iops = (known after apply)

+ kms\_key\_id = (known after apply)

+ snapshot\_id = (known after apply)

+ tags = (known after apply)

+ throughput = (known after apply)

+ volume\_id = (known after apply)

+ volume\_size = (known after apply)

+ volume\_type = (known after apply)

}

+ enclave\_options {

+ enabled = (known after apply)

}

+ ephemeral\_block\_device {

+ device\_name = (known after apply)

+ no\_device = (known after apply)

+ virtual\_name = (known after apply)

}

+ metadata\_options {

+ http\_endpoint = (known after apply)

+ http\_put\_response\_hop\_limit = (known after apply)

+ http\_tokens = (known after apply)

}

+ network\_interface {

+ delete\_on\_termination = (known after apply)

+ device\_index = (known after apply)

+ network\_interface\_id = (known after apply)

}

+ root\_block\_device {

+ delete\_on\_termination = (known after apply)

+ device\_name = (known after apply)

+ encrypted = (known after apply)

+ iops = (known after apply)

+ kms\_key\_id = (known after apply)

+ tags = (known after apply)

+ throughput = (known after apply)

+ volume\_id = (known after apply)

+ volume\_size = (known after apply)

+ volume\_type = (known after apply)

}

}

# module.compute.aws\_instance.k8s-node will be created

+ resource "aws\_instance" "k8s-node" {

+ ami = (sensitive)

+ arn = (known after apply)

+ associate\_public\_ip\_address = true

+ availability\_zone = (known after apply)

+ cpu\_core\_count = (known after apply)

+ cpu\_threads\_per\_core = (known after apply)

+ get\_password\_data = false

+ host\_id = (known after apply)

+ id = (known after apply)

+ instance\_initiated\_shutdown\_behavior = (known after apply)

+ instance\_state = (known after apply)

+ instance\_type = "t2.micro"

+ ipv6\_address\_count = (known after apply)

+ ipv6\_addresses = (known after apply)

+ key\_name = "k8s"

+ outpost\_arn = (known after apply)

+ password\_data = (known after apply)

+ placement\_group = (known after apply)

+ primary\_network\_interface\_id = (known after apply)

+ private\_dns = (known after apply)

+ private\_ip = (known after apply)

+ public\_dns = (known after apply)

+ public\_ip = (known after apply)

+ secondary\_private\_ips = (known after apply)

+ security\_groups = (known after apply)

+ source\_dest\_check = true

+ subnet\_id = (known after apply)

+ tags = {

+ "Name" = "k8s\_node\_tf"

}

+ tags\_all = {

+ "Name" = "k8s\_node\_tf"

}

+ tenancy = (known after apply)

+ vpc\_security\_group\_ids = (known after apply)

+ capacity\_reservation\_specification {

+ capacity\_reservation\_preference = (known after apply)

+ capacity\_reservation\_target {

+ capacity\_reservation\_id = (known after apply)

}

}

+ ebs\_block\_device {

+ delete\_on\_termination = (known after apply)

+ device\_name = (known after apply)

+ encrypted = (known after apply)

+ iops = (known after apply)

+ kms\_key\_id = (known after apply)

+ snapshot\_id = (known after apply)

+ tags = (known after apply)

+ throughput = (known after apply)

+ volume\_id = (known after apply)

+ volume\_size = (known after apply)

+ volume\_type = (known after apply)

}

+ enclave\_options {

+ enabled = (known after apply)

}

+ ephemeral\_block\_device {

+ device\_name = (known after apply)

+ no\_device = (known after apply)

+ virtual\_name = (known after apply)

}

+ metadata\_options {

+ http\_endpoint = (known after apply)

+ http\_put\_response\_hop\_limit = (known after apply)

+ http\_tokens = (known after apply)

}

+ network\_interface {

+ delete\_on\_termination = (known after apply)

+ device\_index = (known after apply)

+ network\_interface\_id = (known after apply)

}

+ root\_block\_device {

+ delete\_on\_termination = (known after apply)

+ device\_name = (known after apply)

+ encrypted = (known after apply)

+ iops = (known after apply)

+ kms\_key\_id = (known after apply)

+ tags = (known after apply)

+ throughput = (known after apply)

+ volume\_id = (known after apply)

+ volume\_size = (known after apply)

+ volume\_type = (known after apply)

}

}

# module.compute.aws\_key\_pair.aws-key will be created

+ resource "aws\_key\_pair" "aws-key" {

+ arn = (known after apply)

+ fingerprint = (known after apply)

+ id = (known after apply)

+ key\_name = "k8s"

+ key\_pair\_id = (known after apply)

+ public\_key = "ssh-rsa  paulkrznaric@Pauls-MBP.local"

+ tags\_all = (known after apply)

}

# module.vpc.aws\_internet\_gateway.tf\_igw will be created

+ resource "aws\_internet\_gateway" "tf\_igw" {

+ arn = (known after apply)

+ id = (known after apply)

+ owner\_id = (known after apply)

+ tags = {

+ "Name" = "Terraform-Gateway"

}

+ tags\_all = {

+ "Name" = "Terraform-Gateway"

}

+ vpc\_id = (known after apply)

}

# module.vpc.aws\_route\_table.tf\_public\_route will be created

+ resource "aws\_route\_table" "tf\_public\_route" {

+ arn = (known after apply)

+ id = (known after apply)

+ owner\_id = (known after apply)

+ propagating\_vgws = (known after apply)

+ route = [

+ {

+ carrier\_gateway\_id = ""

+ cidr\_block = "0.0.0.0/0"

+ destination\_prefix\_list\_id = ""

+ egress\_only\_gateway\_id = ""

+ gateway\_id = (known after apply)

+ instance\_id = ""

+ ipv6\_cidr\_block = ""

+ local\_gateway\_id = ""

+ nat\_gateway\_id = ""

+ network\_interface\_id = ""

+ transit\_gateway\_id = ""

+ vpc\_endpoint\_id = ""

+ vpc\_peering\_connection\_id = ""

},

]

+ tags = {

+ "Name" = "Terraform-Public-RouteTable"

}

+ tags\_all = {

+ "Name" = "Terraform-Public-RouteTable"

}

+ vpc\_id = (known after apply)

}

# module.vpc.aws\_route\_table\_association.tf\_public1\_assoc will be created

+ resource "aws\_route\_table\_association" "tf\_public1\_assoc" {

+ id = (known after apply)

+ route\_table\_id = (known after apply)

+ subnet\_id = (known after apply)

}

# module.vpc.aws\_route\_table\_association.tf\_public2\_assoc will be created

+ resource "aws\_route\_table\_association" "tf\_public2\_assoc" {

+ id = (known after apply)

+ route\_table\_id = (known after apply)

+ subnet\_id = (known after apply)

}

# module.vpc.aws\_security\_group.tf\_public\_sg will be created

+ resource "aws\_security\_group" "tf\_public\_sg" {

+ arn = (known after apply)

+ description = "Used for access to the public instances"

+ egress = [

+ {

+ cidr\_blocks = [

+ "0.0.0.0/0",

]

+ description = ""

+ from\_port = 0

+ ipv6\_cidr\_blocks = []

+ prefix\_list\_ids = []

+ protocol = "-1"

+ security\_groups = []

+ self = false

+ to\_port = 0

},

]

+ id = (known after apply)

+ ingress = [

+ {

+ cidr\_blocks = [

+ "0.0.0.0/0",

]

+ description = ""

+ from\_port = 0

+ ipv6\_cidr\_blocks = []

+ prefix\_list\_ids = []

+ protocol = "-1"

+ security\_groups = []

+ self = false

+ to\_port = 0

},

]

+ name = "tf\_public\_sg"

+ name\_prefix = (known after apply)

+ owner\_id = (known after apply)

+ revoke\_rules\_on\_delete = false

+ tags = {

+ "Name" = "Terraform-SecurityGroup"

}

+ tags\_all = {

+ "Name" = "Terraform-SecurityGroup"

}

+ vpc\_id = (known after apply)

}

# module.vpc.aws\_subnet.tf\_public\_subnet1 will be created

+ resource "aws\_subnet" "tf\_public\_subnet1" {

+ arn = (known after apply)

+ assign\_ipv6\_address\_on\_creation = false

+ availability\_zone = "us-east-1a"

+ availability\_zone\_id = (known after apply)

+ cidr\_block = "10.0.1.0/24"

+ id = (known after apply)

+ ipv6\_cidr\_block\_association\_id = (known after apply)

+ map\_public\_ip\_on\_launch = false

+ owner\_id = (known after apply)

+ tags = {

+ "Name" = "Terraform-Subnet1"

}

+ tags\_all = {

+ "Name" = "Terraform-Subnet1"

}

+ vpc\_id = (known after apply)

}

# module.vpc.aws\_subnet.tf\_public\_subnet2 will be created

+ resource "aws\_subnet" "tf\_public\_subnet2" {

+ arn = (known after apply)

+ assign\_ipv6\_address\_on\_creation = false

+ availability\_zone = "us-east-1b"

+ availability\_zone\_id = (known after apply)

+ cidr\_block = "10.0.2.0/24"

+ id = (known after apply)

+ ipv6\_cidr\_block\_association\_id = (known after apply)

+ map\_public\_ip\_on\_launch = false

+ owner\_id = (known after apply)

+ tags = {

+ "Name" = "Terraform-Subnet2"

}

+ tags\_all = {

+ "Name" = "Terraform-Subnet2"

}

+ vpc\_id = (known after apply)

}

# module.vpc.aws\_vpc.tf\_vpc will be created

+ resource "aws\_vpc" "tf\_vpc" {

+ arn = (known after apply)

+ assign\_generated\_ipv6\_cidr\_block = false

+ cidr\_block = "10.0.0.0/16"

+ default\_network\_acl\_id = (known after apply)

+ default\_route\_table\_id = (known after apply)

+ default\_security\_group\_id = (known after apply)

+ dhcp\_options\_id = (known after apply)

+ enable\_classiclink = (known after apply)

+ enable\_classiclink\_dns\_support = (known after apply)

+ enable\_dns\_hostnames = true

+ enable\_dns\_support = true

+ id = (known after apply)

+ instance\_tenancy = "default"

+ ipv6\_association\_id = (known after apply)

+ ipv6\_cidr\_block = (known after apply)

+ main\_route\_table\_id = (known after apply)

+ owner\_id = (known after apply)

+ tags = {

+ "Name" = "Terraform-VPC"

}

+ tags\_all = {

+ "Name" = "Terraform-VPC"

}

}

Plan: 11 to add, 0 to change, 0 to destroy.

Changes to Outputs:

+ k8s-master-Public-IP = (known after apply)

+ k8s-node-Public-IP = (known after apply)

Do you want to perform these actions?

Terraform will perform the actions described above.

Only 'yes' will be accepted to approve.

Enter a value: yes

module.compute.aws\_key\_pair.aws-key: Creating...

module.vpc.aws\_vpc.tf\_vpc: Creating...

module.compute.aws\_key\_pair.aws-key: Creation complete after 1s [id=k8s]

module.vpc.aws\_vpc.tf\_vpc: Still creating... [10s elapsed]

module.vpc.aws\_vpc.tf\_vpc: Creation complete after 13s [id=vpc-00d9f22c6e275ea34]

module.vpc.aws\_internet\_gateway.tf\_igw: Creating...

module.vpc.aws\_subnet.tf\_public\_subnet2: Creating...

module.vpc.aws\_subnet.tf\_public\_subnet1: Creating...

module.vpc.aws\_security\_group.tf\_public\_sg: Creating...

module.vpc.aws\_subnet.tf\_public\_subnet2: Creation complete after 1s [id=subnet-084e2bc899094ddbb]

module.vpc.aws\_subnet.tf\_public\_subnet1: Creation complete after 1s [id=subnet-05df4ddd916f479f2]

module.vpc.aws\_internet\_gateway.tf\_igw: Creation complete after 1s [id=igw-0e33d33badd000d50]

module.vpc.aws\_route\_table.tf\_public\_route: Creating...

module.vpc.aws\_route\_table.tf\_public\_route: Creation complete after 1s [id=rtb-06c1bb74e49efb069]

module.vpc.aws\_route\_table\_association.tf\_public1\_assoc: Creating...

module.vpc.aws\_route\_table\_association.tf\_public2\_assoc: Creating...

module.vpc.aws\_route\_table\_association.tf\_public2\_assoc: Creation complete after 1s [id=rtbassoc-080d7af0e01c8fdbd]

module.vpc.aws\_route\_table\_association.tf\_public1\_assoc: Creation complete after 1s [id=rtbassoc-032f2d7fd4fd5e0ad]

module.vpc.aws\_security\_group.tf\_public\_sg: Creation complete after 4s [id=sg-04854b3a075770061]

module.compute.aws\_instance.k8s-node: Creating...

module.compute.aws\_instance.k8s-master: Creating...

module.compute.aws\_instance.k8s-node: Still creating... [10s elapsed]

module.compute.aws\_instance.k8s-master: Still creating... [10s elapsed]

module.compute.aws\_instance.k8s-node: Still creating... [20s elapsed]

module.compute.aws\_instance.k8s-master: Still creating... [20s elapsed]

module.compute.aws\_instance.k8s-master: Still creating... [30s elapsed]

module.compute.aws\_instance.k8s-node: Still creating... [30s elapsed]

module.compute.aws\_instance.k8s-node: Creation complete after 34s [id=i-0dc0247b246f14bea]

module.compute.aws\_instance.k8s-master: Still creating... [40s elapsed]

module.compute.aws\_instance.k8s-master: Creation complete after 44s [id=i-0b7da65944b315dc8]

Apply complete! Resources: 11 added, 0 changed, 0 destroyed.

Outputs:

k8s-master-Public-IP = "3.90.110.241"

k8s-node-Public-IP = "54.87.74.149"

Text

Description automatically generated

## Sudo kubeadm init –config kube-config.yml –ignore preflight-errors=all

Text

Description automatically generated

## [ec2-user@ip-10-0-1-184 ~]$ sudo kubeadm init --config kube-config.yml --ignore-preflight-errors=all

## [init] Using Kubernetes version: v1.22.0

## [preflight] Running pre-flight checks

## [WARNING NumCPU]: the number of available CPUs 1 is less than the required 2

## [WARNING Mem]: the system RAM (982 MB) is less than the minimum 1700 MB

## [WARNING FileExisting-tc]: tc not found in system path

## [WARNING KubeletVersion]: the kubelet version is higher than the control plane version. This is not a supported version skew and may lead to a malfunctional cluster. Kubelet version: "1.23.3" Control plane version: "1.22.0"

## [preflight] Pulling images required for setting up a Kubernetes cluster

## [preflight] This might take a minute or two, depending on the speed of your internet connection

## [preflight] You can also perform this action in beforehand using 'kubeadm config images pull'

## [certs] Using certificateDir folder "/etc/kubernetes/pki"

## [certs] Generating "ca" certificate and key

## [certs] Generating "apiserver" certificate and key

## [certs] apiserver serving cert is signed for DNS names [ip-10-0-1-184.ec2.internal kubernetes kubernetes.default kubernetes.default.svc kubernetes.default.svc.cluster.local] and IPs [10.96.0.1 10.0.1.184]

## [certs] Generating "apiserver-kubelet-client" certificate and key

## [certs] Generating "front-proxy-ca" certificate and key

## [certs] Generating "front-proxy-client" certificate and key

## [certs] Generating "etcd/ca" certificate and key

## [certs] Generating "etcd/server" certificate and key

## [certs] etcd/server serving cert is signed for DNS names [ip-10-0-1-184.ec2.internal localhost] and IPs [10.0.1.184 127.0.0.1 ::1]

## [certs] Generating "etcd/peer" certificate and key

## [certs] etcd/peer serving cert is signed for DNS names [ip-10-0-1-184.ec2.internal localhost] and IPs [10.0.1.184 127.0.0.1 ::1]

## [certs] Generating "etcd/healthcheck-client" certificate and key

## [certs] Generating "apiserver-etcd-client" certificate and key

## [certs] Generating "sa" key and public key

## [kubeconfig] Using kubeconfig folder "/etc/kubernetes"

## [kubeconfig] Writing "admin.conf" kubeconfig file

## [kubeconfig] Writing "kubelet.conf" kubeconfig file

## [kubeconfig] Writing "controller-manager.conf" kubeconfig file

## [kubeconfig] Writing "scheduler.conf" kubeconfig file

## [kubelet-start] Writing kubelet environment file with flags to file "/var/lib/kubelet/kubeadm-flags.env"

## [kubelet-start] Writing kubelet configuration to file "/var/lib/kubelet/config.yaml"

## [kubelet-start] Starting the kubelet

## [control-plane] Using manifest folder "/etc/kubernetes/manifests"

## [control-plane] Creating static Pod manifest for "kube-apiserver"

## [control-plane] Creating static Pod manifest for "kube-controller-manager"

## [control-plane] Creating static Pod manifest for "kube-scheduler"

## [etcd] Creating static Pod manifest for local etcd in "/etc/kubernetes/manifests"

## [wait-control-plane] Waiting for the kubelet to boot up the control plane as static Pods from directory "/etc/kubernetes/manifests". This can take up to 4m0s

## [apiclient] All control plane components are healthy after 26.003382 seconds

## [upload-config] Storing the configuration used in ConfigMap "kubeadm-config" in the "kube-system" Namespace

## [kubelet] Creating a ConfigMap "kubelet-config-1.22" in namespace kube-system with the configuration for the kubelets in the cluster

## NOTE: The "kubelet-config-1.22" naming of the kubelet ConfigMap is deprecated. Once the UnversionedKubeletConfigMap feature gate graduates to Beta the default name will become just "kubelet-config". Kubeadm upgrade will handle this transition transparently.

## [upload-certs] Skipping phase. Please see --upload-certs

## [mark-control-plane] Marking the node ip-10-0-1-184.ec2.internal as control-plane by adding the labels: [node-role.kubernetes.io/master(deprecated) node-role.kubernetes.io/control-plane node.kubernetes.io/exclude-from-external-load-balancers]

## [mark-control-plane] Marking the node ip-10-0-1-184.ec2.internal as control-plane by adding the taints [node-role.kubernetes.io/master:NoSchedule]

## [bootstrap-token] Using token: 5futhu.09y2a4c9fgn4h0gq

## [bootstrap-token] Configuring bootstrap tokens, cluster-info ConfigMap, RBAC Roles

## [bootstrap-token] configured RBAC rules to allow Node Bootstrap tokens to get nodes

## [bootstrap-token] configured RBAC rules to allow Node Bootstrap tokens to post CSRs in order for nodes to get long term certificate credentials

## [bootstrap-token] configured RBAC rules to allow the csrapprover controller automatically approve CSRs from a Node Bootstrap Token

## [bootstrap-token] configured RBAC rules to allow certificate rotation for all node client certificates in the cluster

## [bootstrap-token] Creating the "cluster-info" ConfigMap in the "kube-public" namespace

## [kubelet-finalize] Updating "/etc/kubernetes/kubelet.conf" to point to a rotatable kubelet client certificate and key

## [addons] Applied essential addon: CoreDNS

## [addons] Applied essential addon: kube-proxy

## Your Kubernetes control-plane has initialized successfully!

## To start using your cluster, you need to run the following as a regular user:

## mkdir -p $HOME/.kube

## sudo cp -i /etc/kubernetes/admin.conf $HOME/.kube/config

## sudo chown $(id -u):$(id -g) $HOME/.kube/config

## Alternatively, if you are the root user, you can run:

## export KUBECONFIG=/etc/kubernetes/admin.conf

## You should now deploy a pod network to the cluster.

## Run "kubectl apply -f [podnetwork].yaml" with one of the options listed at:

## https://kubernetes.io/docs/concepts/cluster-administration/addons/

## Then you can join any number of worker nodes by running the following on each as root:

## kubeadm join 10.0.1.184:6443 --token 5futhu.09y2a4c9fgn4h0gq \

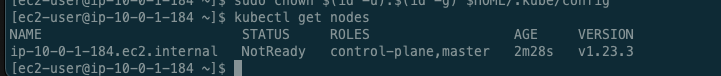
## --discovery-token-ca-cert-hash sha256:dc45c6743bf0a0b233ab60a6a024a1a52250c9eda38bba51feebb4c84b62def3

## Kubectl get nodes

[ec2-user@ip-10-0-1-184 ~]$ kubectl get nodes

NAME STATUS ROLES AGE VERSION

ip-10-0-1-184.ec2.internal NotReady control-plane,master 2m28s v1.23.3



[ec2-user@ip-10-0-1-184 ~]$ kubectl get nodes

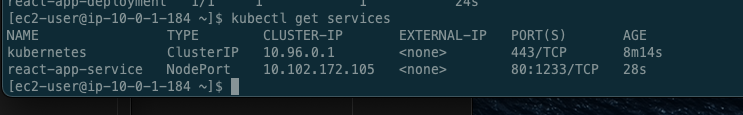
NAME STATUS ROLES AGE VERSION

ip-10-0-1-184.ec2.internal Ready control-plane,master 4m33s v1.23.3

A screenshot of a computer

Description automatically generated with medium confidence

## Kubectl get services



[ec2-user@ip-10-0-1-184 ~]$ kubectl get services

NAME TYPE CLUSTER-IP EXTERNAL-IP PORT(S) AGE

kubernetes ClusterIP 10.96.0.1 <none> 443/TCP 8m14s

react-app-service NodePort 10.102.172.105 <none> 80:1233/TCP 28s

## Port 1233

### Master IP

Text

Description automatically generated

### Worker IP

Text

Description automatically generated