For VRT setup we need to follow below steps:

1. **Create EKS Cluster with below command:**

eksctl create cluster --name dmi-demo --vpc-cidr 10.50.0.0/16 --region ap-south-1 --nodegroup-name application-worker-node --node-type r5a.large --nodes 3 --nodes-min 3 --nodes-max 6 --node-volume-size 40 --node-volume-type gp3 --enable-ssm --managed --node-private-networking --ssh-access=true --ssh-public-key=demi-demo --profile dmi-eks

```
rupam.saini@scriptbox:~$ eksctl create cluster --name dmi-demo --vpc-cidr 10.50.0.0/16 --region ap-south-1 --nodegroup-name application-worker-node --node-type r5a.large --nodes 3 --nodes-min 3 --nodes-max 6 --node-volume-size 40 --node-volume-type gp3 --enable-ssm --managed --node-private-networking --ssh-access=true --ssh-public-key=demi-demo --profile dmi-eks
2022-12-19 16:05:52 [!]  SSM is now enabled by default; `ssh.enableSSM` is deprecated and will be removed in a future release
2022-12-19 16:05:53 [ℹ]  eksctl version 0.122.0
2022-12-19 16:05:53 [ℹ]  using region ap-south-1
2022-12-19 16:05:53 [ℹ]  setting availability zones to [ap-south-1a ap-south-1c ap-south-1b]
2022-12-19 16:05:53 [ℹ]  subnets for ap-south-1a - public:10.50.0.0/19 private:10.50.96.0/19
2022-12-19 16:05:53 [ℹ]  subnets for ap-south-1c - public:10.50.32.0/19 private:10.50.128.0/19
2022-12-19 16:05:53 [ℹ]  subnets for ap-south-1b - public:10.50.64.0/19 private:10.50.160.0/19
2022-12-19 16:05:53 [ℹ]  nodegroup "application-worker-node" will use "" [AmazonLinux2/1.23]
2022-12-19 16:05:53 [ℹ]  using EC2 key pair %!q(*string=<nil>)
2022-12-19 16:05:53 [ℹ]  using Kubernetes version 1.23
2022-12-19 16:05:53 [ℹ]  creating EKS cluster "dmi-demo" in "ap-south-1" region with managed nodes
2022-12-19 16:05:53 [ℹ]  will create 2 separate CloudFormation stacks for cluster itself and the initial managed nodegroup
2022-12-19 16:05:53 [ℹ]  if you encounter any issues, check CloudFormation console or try 'eksctl utils describe-stacks --region=ap-south-1 --cluster=dmi-demo
'
2022-12-19 16:05:53 [ℹ]  Kubernetes API endpoint access will use default of {publicAccess=true, privateAccess=false} for cluster "dmi-demo" in "ap-south-1"
2022-12-19 16:05:53 [ℹ]  CloudWatch logging will not be enabled for cluster "dmi-demo" in "ap-south-1"
2022-12-19 16:05:53 [ℹ]  you can enable it with 'eksctl utils update-cluster-logging --enable-types={SPECIFY-YOUR-LOG-TYPES-HERE (e.g. all)} --region=ap-south
-1 --cluster=dmi-demo'
2022-12-19 16:05:53 [ℹ]
2 sequential tasks: { create cluster control plane "dmi-demo",
    2 sequential sub-tasks: {
        wait for control plane to become ready,
        create managed nodegroup "application-worker-node",
    }
}
2022-12-19 16:05:53 [ℹ]  building cluster stack "eksctl-dmi-demo-cluster"
2022-12-19 16:05:54 [ℹ]  deploying stack "eksctl-dmi-demo-cluster"
2022-12-19 16:06:24 [ℹ]  waiting for CloudFormation stack "eksctl-dmi-demo-cluster"
2022-12-19 16:06:54 [ℹ]  waiting for CloudFormation stack "eksctl-dmi-demo-cluster"
2022-12-19 16:07:54 [ℹ]  waiting for CloudFormation stack "eksctl-dmi-demo-cluster"
2022-12-19 16:08:55 [ℹ]  waiting for CloudFormation stack "eksctl-dmi-demo-cluster"
```


```
2022-12-19 16:05:52 [!]  SSM is now enabled by default; `ssh.enableSSM` is deprecated and will be removed in a future release
2022-12-19 16:05:53 [ℹ]  eksctl version 0.122.0
2022-12-19 16:05:53 [ℹ]  using region ap-south-1
2022-12-19 16:05:53 [ℹ]  setting availability zones to [ap-south-1a ap-south-1c ap-south-1b]
2022-12-19 16:05:53 [ℹ]  subnets for ap-south-1a - public:10.50.0.0/19 private:10.50.96.0/19
2022-12-19 16:05:53 [ℹ]  subnets for ap-south-1c - public:10.50.32.0/19 private:10.50.128.0/19
2022-12-19 16:05:53 [ℹ]  subnets for ap-south-1b - public:10.50.64.0/19 private:10.50.160.0/19
2022-12-19 16:05:53 [ℹ]  nodegroup "application-worker-node" will use "" [AmazonLinux2/1.23]
2022-12-19 16:05:53 [ℹ]  using EC2 key pair %!q(*string=<nil>)
2022-12-19 16:05:53 [ℹ]  using Kubernetes version 1.23
2022-12-19 16:05:53 [ℹ]  creating EKS cluster "dmi-demo" in "ap-south-1" region with managed nodes
2022-12-19 16:05:53 [ℹ]  will create 2 separate CloudFormation stacks for cluster itself and the initial managed nodegroup
2022-12-19 16:05:53 [ℹ]  if you encounter any issues, check CloudFormation console or try 'eksctl utils describe-stacks --region=ap-south-1 --cluster=dmi-demo
'
2022-12-19 16:05:53 [ℹ]  Kubernetes API endpoint access will use default of {publicAccess=true, privateAccess=false} for cluster "dmi-demo" in "ap-south-1"
2022-12-19 16:05:53 [ℹ]  CloudWatch logging will not be enabled for cluster "dmi-demo" in "ap-south-1"
2022-12-19 16:05:53 [ℹ]  you can enable it with 'eksctl utils update-cluster-logging --enable-types={SPECIFY-YOUR-LOG-TYPES-HERE (e.g. all)} --region=ap-south
-1 --cluster=dmi-demo'
2022-12-19 16:05:53 [ℹ]
2 sequential tasks: { create cluster control plane "dmi-demo",
    2 sequential sub-tasks: {
        wait for control plane to become ready,
        create managed nodegroup "application-worker-node",
    }
}
2022-12-19 16:05:53 [ℹ]  building cluster stack "eksctl-dmi-demo-cluster"
2022-12-19 16:05:54 [ℹ]  deploying stack "eksctl-dmi-demo-cluster"
2022-12-19 16:06:24 [ℹ]  waiting for CloudFormation stack "eksctl-dmi-demo-cluster"
2022-12-19 16:06:54 [ℹ]  waiting for CloudFormation stack "eksctl-dmi-demo-cluster"
2022-12-19 16:07:54 [ℹ]  waiting for CloudFormation stack "eksctl-dmi-demo-cluster"
2022-12-19 16:08:55 [ℹ]  waiting for CloudFormation stack "eksctl-dmi-demo-cluster"
```

2. replace the EKS cluster name with dmi-demo on command and create Node group for custom application.

eksctl create nodegroup --cluster=dmi-demo --region=ap-south-1 --managed --name=custom-apps-nodegroup --instance-types=r5.large,t3.medium,t3a.large,t3a.xlarge --nodes-min=2 --nodes-max=8 --node-volume-type gp3 --node-volume-size 15 --asg-access --enable-ssm --managed --node-private-networking --ssh-access=true --ssh-public-key=demi-demo --profile dmi-eks

```
rupam.saini@scriptbox:~$ eksctl create nodegroup --cluster=dmi-demo --region=ap-south-1 --managed --name=custom-apps-nodegroup --instance-types=r5.large,t3.me
dium,t3a.large,t3a.xlarge --nodes-min=2 --nodes-max=8 --node-volume-type gp3 --node-volume-size 15 --asg-access --enable-ssm --managed --node-private-networki
ng --ssh-access=true --ssh-public-key=demi-demo --profile dmi-eks
2022-12-19 16:34:58 [ℹ]  will use version 1.23 for new nodegroup(s) based on control plane version
2022-12-19 16:34:58 [!]  SSM is now enabled by default; `ssh.enableSSM` is deprecated and will be removed in a future release
2022-12-19 16:34:59 [ℹ]  nodegroup "custom-apps-nodegroup" will use "" [AmazonLinux2/1.23]
2022-12-19 16:35:00 [ℹ]  using EC2 key pair %!q(*string=<nil>)
2022-12-19 16:35:00 [ℹ]  1 existing nodegroup(s) (application-worker-node) will be excluded
2022-12-19 16:35:00 [ℹ]  1 nodegroup (custom-apps-nodegroup) was included (based on the include/exclude rules)
2022-12-19 16:35:00 [ℹ]  will create a CloudFormation stack for each of 1 managed nodegroups in cluster "dmi-demo"
2022-12-19 16:35:00 [ℹ]
2 sequential tasks: { fix cluster compatibility, 1 task: { 1 task: { create managed nodegroup "custom-apps-nodegroup" } } }
}
2022-12-19 16:35:00 [ℹ]  checking cluster stack for missing resources
2022-12-19 16:35:00 [ℹ]  cluster stack has all required resources
2022-12-19 16:35:01 [ℹ]  building managed nodegroup stack "eksctl-dmi-demo-nodegroup-custom-apps-nodegroup"
2022-12-19 16:35:01 [ℹ]  deploying stack "eksctl-dmi-demo-nodegroup-custom-apps-nodegroup"
2022-12-19 16:35:01 [ℹ]  waiting for CloudFormation stack "eksctl-dmi-demo-nodegroup-custom-apps-nodegroup"
2022-12-19 16:35:32 [ℹ]  waiting for CloudFormation stack "eksctl-dmi-demo-nodegroup-custom-apps-nodegroup"
2022-12-19 16:36:26 [ℹ]  waiting for CloudFormation stack "eksctl-dmi-demo-nodegroup-custom-apps-nodegroup"
2022-12-19 16:37:29 [ℹ]  waiting for CloudFormation stack "eksctl-dmi-demo-nodegroup-custom-apps-nodegroup"
2022-12-19 16:38:10 [ℹ]  waiting for CloudFormation stack "eksctl-dmi-demo-nodegroup-custom-apps-nodegroup"
2022-12-19 16:39:08 [ℹ]  waiting for CloudFormation stack "eksctl-dmi-demo-nodegroup-custom-apps-nodegroup"
2022-12-19 16:39:08 [ℹ]  no tasks
2022-12-19 16:39:08 [✔]  created 0 nodegroup(s) in cluster "dmi-demo"
2022-12-19 16:39:08 [ℹ]  nodegroup "custom-apps-nodegroup" has 2 node(s)
2022-12-19 16:39:08 [ℹ]  node "ip-10-50-155-253.ap-south-1.compute.internal" is ready
2022-12-19 16:39:08 [ℹ]  node "ip-10-50-179-180.ap-south-1.compute.internal" is ready
2022-12-19 16:39:08 [ℹ]  waiting for at least 2 node(s) to become ready in "custom-apps-nodegroup"
2022-12-19 16:39:08 [ℹ]  nodegroup "custom-apps-nodegroup" has 2 node(s)
2022-12-19 16:39:08 [ℹ]  node "ip-10-50-155-253.ap-south-1.compute.internal" is ready
2022-12-19 16:39:08 [ℹ]  node "ip-10-50-179-180.ap-south-1.compute.internal" is ready
2022-12-19 16:39:08 [✔]  created 1 managed nodegroup(s) in cluster "dmi-demo"
2022-12-19 16:39:09 [ℹ]  checking security group configuration for all nodegroups
```

3. Create Tools stack nodegroup for Elasticsearch, logstash, Kafka.

eksctl create nodegroup --cluster=dmi-demo --region=ap-south-1 --managed --name=tools-elk --instance-types=r5a.large --nodes-min=2 --nodes-max=8 --node-volume-type gp3 --node-volume-size 15 --asg-access --enable-ssm --managed --node-private-networking --ssh-access=true --ssh-public-key=demi-demo --profile dmi-eks

```
rupam.saini@scriptbox:~$ eksctl create nodegroup --cluster=dmi-demo --region=ap-south-1 --managed --name=tools-elk --instance-types=r5a.large --nodes-min=2 --nodes-max=8 --node-volume-type gp3 --node-volume-size 15 --asg-access --enable-ssm --managed --node-private-networking --ssh-access=true --ssh-public-key=demi-demo --profile dmi-eks
2022-12-19 16:51:40 [ℹ]  will use version 1.23 for new nodegroup(s) based on control plane version
2022-12-19 16:51:40 [!]  SSM is now enabled by default; `ssh.enableSSM` is deprecated and will be removed in a future release
2022-12-19 16:51:41 [ℹ]  nodegroup "tools-elk" will use "" [AmazonLinux2/1.23]
2022-12-19 16:51:41 [ℹ]  using EC2 key pair %!q(*string=<nil>)
2022-12-19 16:51:42 [ℹ]  2 existing nodegroup(s) (application-worker-node,custom-apps-nodegroup) will be excluded
2022-12-19 16:51:42 [ℹ]  1 nodegroup (tools-elk) was included (based on the include/exclude rules)
2022-12-19 16:51:42 [ℹ]  will create a CloudFormation stack for each of 1 managed nodegroups in cluster "dmi-demo"
2022-12-19 16:51:42 [ℹ]
2 sequential tasks: { fix cluster compatibility, 1 task: { 1 task: { create managed nodegroup "tools-elk" } }
}
2022-12-19 16:51:42 [ℹ]  checking cluster stack for missing resources
2022-12-19 16:51:42 [ℹ]  cluster stack has all required resources
2022-12-19 16:51:43 [ℹ]  building managed nodegroup stack "eksctl-dmi-demo-nodegroup-tools-elk"
2022-12-19 16:51:43 [ℹ]  deploying stack "eksctl-dmi-demo-nodegroup-tools-elk"
2022-12-19 16:51:43 [ℹ]  waiting for CloudFormation stack "eksctl-dmi-demo-nodegroup-tools-elk"
2022-12-19 16:52:13 [ℹ]  waiting for CloudFormation stack "eksctl-dmi-demo-nodegroup-tools-elk"
2022-12-19 16:52:54 [ℹ]  waiting for CloudFormation stack "eksctl-dmi-demo-nodegroup-tools-elk"
2022-12-19 16:54:06 [ℹ]  waiting for CloudFormation stack "eksctl-dmi-demo-nodegroup-tools-elk"
2022-12-19 16:55:26 [ℹ]  waiting for CloudFormation stack "eksctl-dmi-demo-nodegroup-tools-elk"
2022-12-19 16:55:26 [ℹ]  no tasks
2022-12-19 16:55:26 [✓]  created 0 nodegroup(s) in cluster "dmi-demo"
2022-12-19 16:55:26 [ℹ]  nodegroup "tools-elk" has 2 node(s)
2022-12-19 16:55:26 [ℹ]  node "ip-10-50-107-80.ap-south-1.compute.internal" is ready
2022-12-19 16:55:26 [ℹ]  node "ip-10-50-132-35.ap-south-1.compute.internal" is ready
2022-12-19 16:55:26 [ℹ]  waiting for at least 2 node(s) to become ready in "tools-elk"
2022-12-19 16:55:26 [ℹ]  nodegroup "tools-elk" has 2 node(s)
2022-12-19 16:55:26 [ℹ]  node "ip-10-50-107-80.ap-south-1.compute.internal" is ready
2022-12-19 16:55:26 [ℹ]  node "ip-10-50-132-35.ap-south-1.compute.internal" is ready
2022-12-19 16:55:26 [✓]  created 1 managed nodegroup(s) in cluster "dmi-demo"
2022-12-19 16:55:27 [ℹ]  checking security group configuration for all nodegroups
2022-12-19 16:55:27 [ℹ]  all nodegroups have up-to-date cloudformation templates
```

4. Create Node group for Consul service monitoring.

eksctl create nodegroup --cluster=dmi-demo --region=ap-south-1 --managed --name=consul-monitoring --instance-types=r5a.large --nodes-min=2 --nodes-max=8 --node-volume-type gp3 --node-volume-size 15 --asg-access --enable-ssm --managed --node-private-networking --ssh-access=true --ssh-public-key=demi-demo --profile dmi-eks

```
rupam.saini@scriptbox:~$ eksctl create nodegroup --cluster=dmi-demo --region=ap-south-1 --managed --name=consul-monitoring --instance-types=r5a.large --nodes-
min=2 --nodes-max=8 --node-volume-type gp3 --node-volume-size 15 --asg-access --enable-ssm --managed --node-private-networking --ssh-access=true --ssh-public-
key=demi-demo --profile dmi-eks
2022-12-19 17:00:44 [ℹ]  will use version 1.23 for new nodegroup(s) based on control plane version
2022-12-19 17:00:44 [!]  SSM is now enabled by default; `ssh.enableSSM` is deprecated and will be removed in a future release
2022-12-19 17:00:46 [ℹ]  nodegroup "consul-monitoring" will use "" [AmazonLinux2/1.23]
2022-12-19 17:00:46 [ℹ]  using EC2 key pair %!q(*string=<nil>)
2022-12-19 17:00:46 [ℹ]  3 existing nodegroup(s) (application-worker-node,custom-apps-nodegroup,tools-elk) will be excluded
2022-12-19 17:00:46 [ℹ]  1 nodegroup (consul-monitoring) was included (based on the include/exclude rules)
2022-12-19 17:00:46 [ℹ]  will create a CloudFormation stack for each of 1 managed nodegroups in cluster "dmi-demo"
2022-12-19 17:00:46 [ℹ]
2 sequential tasks: { fix cluster compatibility, 1 task: { 1 task: { create managed nodegroup "consul-monitoring" } }
}
2022-12-19 17:00:46 [ℹ]  checking cluster stack for missing resources
2022-12-19 17:00:47 [ℹ]  cluster stack has all required resources
2022-12-19 17:00:47 [ℹ]  building managed nodegroup stack "eksctl-dmi-demo-nodegroup-consul-monitoring"
2022-12-19 17:00:47 [ℹ]  deploying stack "eksctl-dmi-demo-nodegroup-consul-monitoring"
2022-12-19 17:00:47 [ℹ]  waiting for CloudFormation stack "eksctl-dmi-demo-nodegroup-consul-monitoring"
2022-12-19 17:01:18 [ℹ]  waiting for CloudFormation stack "eksctl-dmi-demo-nodegroup-consul-monitoring"
2022-12-19 17:02:14 [ℹ]  waiting for CloudFormation stack "eksctl-dmi-demo-nodegroup-consul-monitoring"
2022-12-19 17:03:33 [ℹ]  waiting for CloudFormation stack "eksctl-dmi-demo-nodegroup-consul-monitoring"
2022-12-19 17:05:09 [ℹ]  waiting for CloudFormation stack "eksctl-dmi-demo-nodegroup-consul-monitoring"
2022-12-19 17:05:09 [ℹ]  no tasks
2022-12-19 17:05:09 [✔]  created 0 nodegroup(s) in cluster "dmi-demo"
2022-12-19 17:05:09 [ℹ]  nodegroup "consul-monitoring" has 2 node(s)
2022-12-19 17:05:09 [ℹ]  node "ip-10-50-123-195.ap-south-1.compute.internal" is ready
2022-12-19 17:05:09 [ℹ]  node "ip-10-50-176-110.ap-south-1.compute.internal" is ready
2022-12-19 17:05:09 [ℹ]  waiting for at least 2 node(s) to become ready in "consul-monitoring"
2022-12-19 17:05:09 [ℹ]  nodegroup "consul-monitoring" has 2 node(s)
2022-12-19 17:05:09 [ℹ]  node "ip-10-50-123-195.ap-south-1.compute.internal" is ready
2022-12-19 17:05:09 [ℹ]  node "ip-10-50-176-110.ap-south-1.compute.internal" is ready
2022-12-19 17:05:09 [✔]  created 1 managed nodegroup(s) in cluster "dmi-demo"
2022-12-19 17:05:10 [ℹ]  checking security group configuration for all nodegroups
2022-12-19 17:05:10 [ℹ]  all nodegroups have up-to-date cloudformation templates
```

5. Create Node Group for redis tools.

eksctl create nodegroup --cluster=dmi-demo --region=ap-south-1 --managed --name=tools-redis --instance-types=r5a.large --nodes-min=2 --nodes-max=8 --node-volume-type gp3 --node-volume-size 15 --asg-access --enable-ssm --managed --node-private-networking --ssh-access=true --ssh-public-key=demi-demo --profile dmi-eks

```
rupam.saini@scriptbox:~$ eksctl create nodegroup --cluster=dmi-demo --region=ap-south-1 --managed --name=tools-redis --instance-types=r5a.large --nodes-min=2
--nodes-max=8 --node-volume-type gp3 --node-volume-size 15 --asg-access --enable-ssm --managed --node-private-networking --ssh-access=true --ssh-public-key=de
mi-demo --profile dmi-eks
2022-12-19 17:03:33 [ℹ]  will use version 1.23 for new nodegroup(s) based on control plane version
2022-12-19 17:03:33 [!]  SSM is now enabled by default; `ssh.enableSSM` is deprecated and will be removed in a future release
2022-12-19 17:03:34 [ℹ]  nodegroup "tools-redis" will use "" [AmazonLinux2/1.23]
2022-12-19 17:03:34 [ℹ]  using EC2 key pair %!q(*string=<nil>)
2022-12-19 17:03:35 [ℹ]  4 existing nodegroup(s) (application-worker-node,consul-monitoring,custom-apps-nodegroup,tools-elk) will be excluded
2022-12-19 17:03:35 [ℹ]  1 nodegroup (tools-redis) was included (based on the include/exclude rules)
2022-12-19 17:03:35 [ℹ]  will create a CloudFormation stack for each of 1 managed nodegroups in cluster "dmi-demo"
2022-12-19 17:03:35 [ℹ]
2 sequential tasks: { fix cluster compatibility, 1 task: { 1 task: { create managed nodegroup "tools-redis" } }
}
2022-12-19 17:03:35 [ℹ]  checking cluster stack for missing resources
2022-12-19 17:03:35 [ℹ]  cluster stack has all required resources
2022-12-19 17:03:36 [ℹ]  building managed nodegroup stack "eksctl-dmi-demo-nodegroup-tools-redis"
2022-12-19 17:03:36 [ℹ]  deploying stack "eksctl-dmi-demo-nodegroup-tools-redis"
2022-12-19 17:03:36 [ℹ]  waiting for CloudFormation stack "eksctl-dmi-demo-nodegroup-tools-redis"
2022-12-19 17:04:06 [ℹ]  waiting for CloudFormation stack "eksctl-dmi-demo-nodegroup-tools-redis"
2022-12-19 17:05:06 [ℹ]  waiting for CloudFormation stack "eksctl-dmi-demo-nodegroup-tools-redis"
2022-12-19 17:06:37 [ℹ]  waiting for CloudFormation stack "eksctl-dmi-demo-nodegroup-tools-redis"
2022-12-19 17:08:17 [ℹ]  waiting for CloudFormation stack "eksctl-dmi-demo-nodegroup-tools-redis"
2022-12-19 17:08:18 [ℹ]  no tasks
2022-12-19 17:08:18 [✔]  created 0 nodegroup(s) in cluster "dmi-demo"
2022-12-19 17:08:18 [ℹ]  nodegroup "tools-redis" has 2 node(s)
2022-12-19 17:08:18 [ℹ]  node "ip-10-50-137-122.ap-south-1.compute.internal" is ready
2022-12-19 17:08:18 [ℹ]  node "ip-10-50-182-210.ap-south-1.compute.internal" is ready
2022-12-19 17:08:18 [ℹ]  waiting for at least 2 node(s) to become ready in "tools-redis"
2022-12-19 17:08:18 [ℹ]  nodegroup "tools-redis" has 2 node(s)
2022-12-19 17:08:18 [ℹ]  node "ip-10-50-137-122.ap-south-1.compute.internal" is ready
2022-12-19 17:08:18 [ℹ]  node "ip-10-50-182-210.ap-south-1.compute.internal" is ready
2022-12-19 17:08:18 [✔]  created 1 managed nodegroup(s) in cluster "dmi-demo"
2022-12-19 17:08:19 [ℹ]  checking security group configuration for all nodegroups
2022-12-19 17:08:19 [ℹ]  all nodegroups have up-to-date cloudformation templates
```

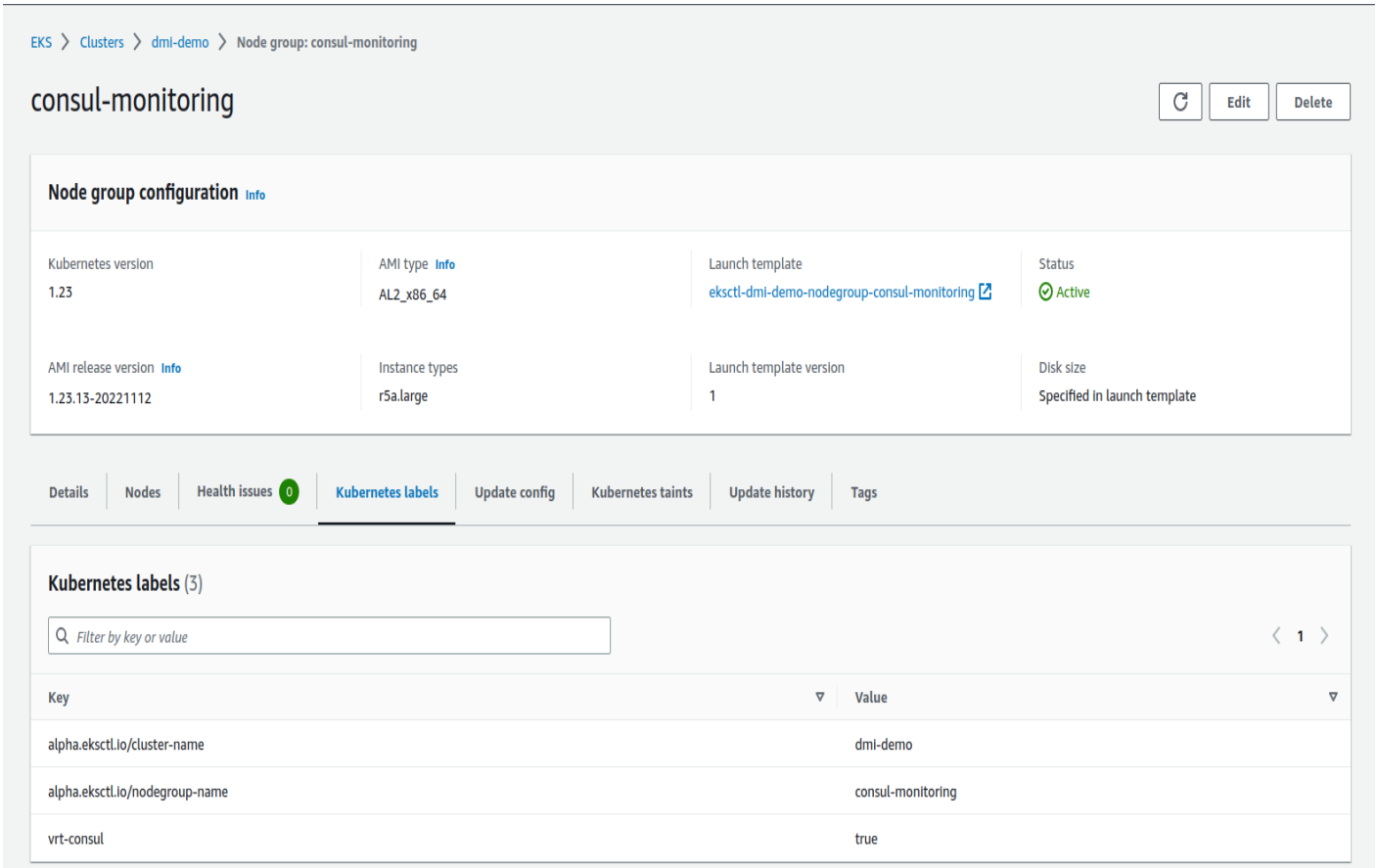
6. Create Node group for Nginx Ingress controller setup.

eksctl create nodegroup --cluster=dmi-demo --region=ap-south-1 --managed --name=tools-nginx --instance-types=r5a.large --nodes-min=2 --nodes-max=8 --node-volume-type gp3 --node-volume-size 15 --asg-access --enable-ssm --managed --node-private-networking --ssh-access=true --ssh-public-key=demi-demo --profile dmi-eks

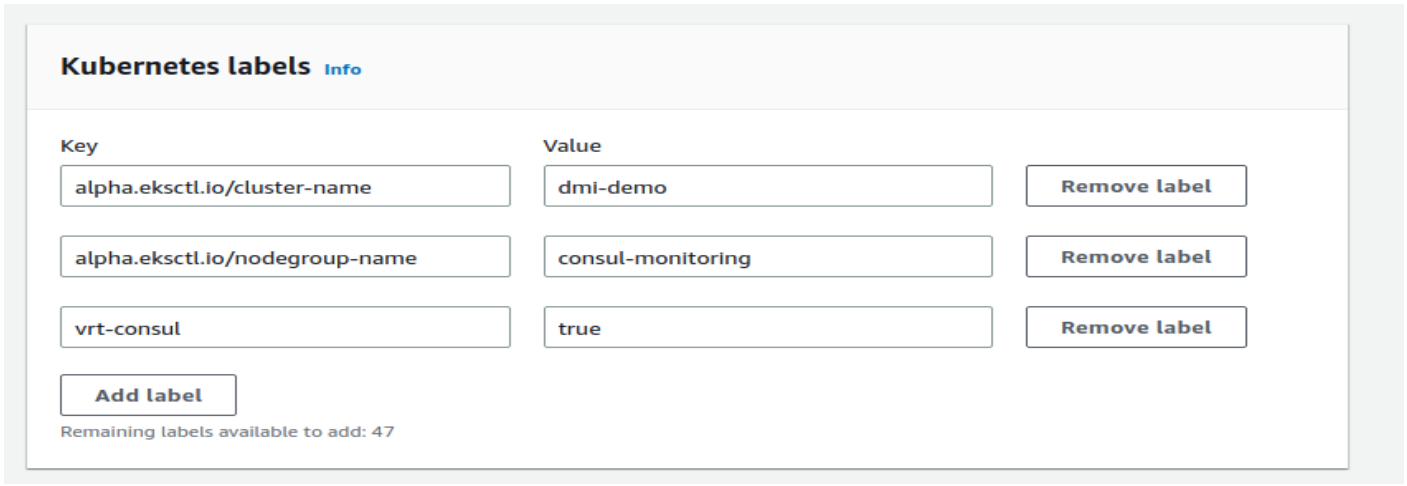
```
rupam.saini@scriptbox:~$ eksctl create nodegroup --cluster=dmi-demo --region=ap-south-1 --managed --name=tools-nginx --instance-types=r5a.large --nodes-min=2
--nodes-max=8 --node-volume-type gp3 --node-volume-size 15 --asg-access --enable-ssm --managed --node-private-networking --ssh-access=true --ssh-public-key=de
mi-demo --profile dmi-eks
2022-12-19 17:06:39 [ℹ]  will use version 1.23 for new nodegroup(s) based on control plane version
2022-12-19 17:06:39 [!]  SSM is now enabled by default; `ssh.enableSSM` is deprecated and will be removed in a future release
2022-12-19 17:06:40 [ℹ]  nodegroup "tools-nginx" will use "" [AmazonLinux2/1.23]
2022-12-19 17:06:40 [ℹ]  using EC2 key pair %!q(*string=<nil>)
2022-12-19 17:06:41 [ℹ]  5 existing nodegroup(s) (application-worker-node,consul-monitoring,custom-apps-nodegroup,tools-elk,tools-redis) will be excluded
2022-12-19 17:06:41 [ℹ]  1 nodegroup (tools-nginx) was included (based on the include/exclude rules)
2022-12-19 17:06:41 [ℹ]  will create a CloudFormation stack for each of 1 managed nodegroups in cluster "dmi-demo"
2022-12-19 17:06:41 [ℹ]
2 sequential tasks: { fix cluster compatibility, 1 task: { 1 task: { create managed nodegroup "tools-nginx" } }
}
2022-12-19 17:06:41 [ℹ]  checking cluster stack for missing resources
2022-12-19 17:06:42 [ℹ]  cluster stack has all required resources
2022-12-19 17:06:42 [ℹ]  building managed nodegroup stack "eksctl-dmi-demo-nodegroup-tools-nginx"
2022-12-19 17:06:43 [ℹ]  deploying stack "eksctl-dmi-demo-nodegroup-tools-nginx"
2022-12-19 17:06:43 [ℹ]  waiting for CloudFormation stack "eksctl-dmi-demo-nodegroup-tools-nginx"
2022-12-19 17:07:13 [ℹ]  waiting for CloudFormation stack "eksctl-dmi-demo-nodegroup-tools-nginx"
2022-12-19 17:08:13 [ℹ]  waiting for CloudFormation stack "eksctl-dmi-demo-nodegroup-tools-nginx"
2022-12-19 17:09:55 [ℹ]  waiting for CloudFormation stack "eksctl-dmi-demo-nodegroup-tools-nginx"
2022-12-19 17:11:19 [ℹ]  waiting for CloudFormation stack "eksctl-dmi-demo-nodegroup-tools-nginx"
2022-12-19 17:11:19 [ℹ]  no tasks
2022-12-19 17:11:19 [✔]  created 0 nodegroup(s) in cluster "dmi-demo"
2022-12-19 17:11:20 [ℹ]  nodegroup "tools-nginx" has 2 node(s)
2022-12-19 17:11:20 [ℹ]  node "ip-10-50-113-234.ap-south-1.compute.internal" is ready
2022-12-19 17:11:20 [ℹ]  node "ip-10-50-144-27.ap-south-1.compute.internal" is ready
2022-12-19 17:11:20 [ℹ]  waiting for at least 2 node(s) to become ready in "tools-nginx"
2022-12-19 17:11:20 [ℹ]  nodegroup "tools-nginx" has 2 node(s)
2022-12-19 17:11:20 [ℹ]  node "ip-10-50-113-234.ap-south-1.compute.internal" is ready
2022-12-19 17:11:20 [ℹ]  node "ip-10-50-144-27.ap-south-1.compute.internal" is ready
2022-12-19 17:11:20 [✔]  created 1 managed nodegroup(s) in cluster "dmi-demo"
2022-12-19 17:11:20 [ℹ]  checking security group configuration for all nodegroups
2022-12-19 17:11:20 [ℹ]  all nodegroups have up-to-date cloudformation templates
```

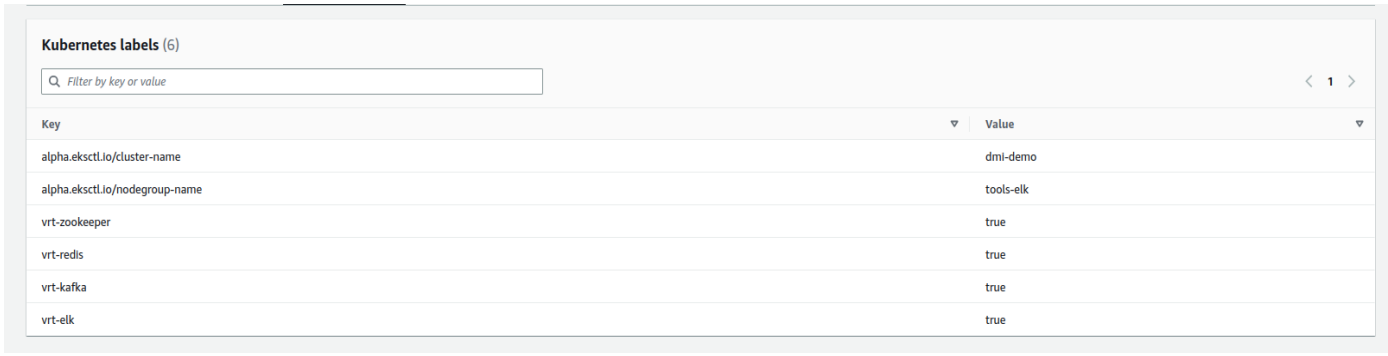
Once above EKS cluster and Node group created then add label to nodegroups:-

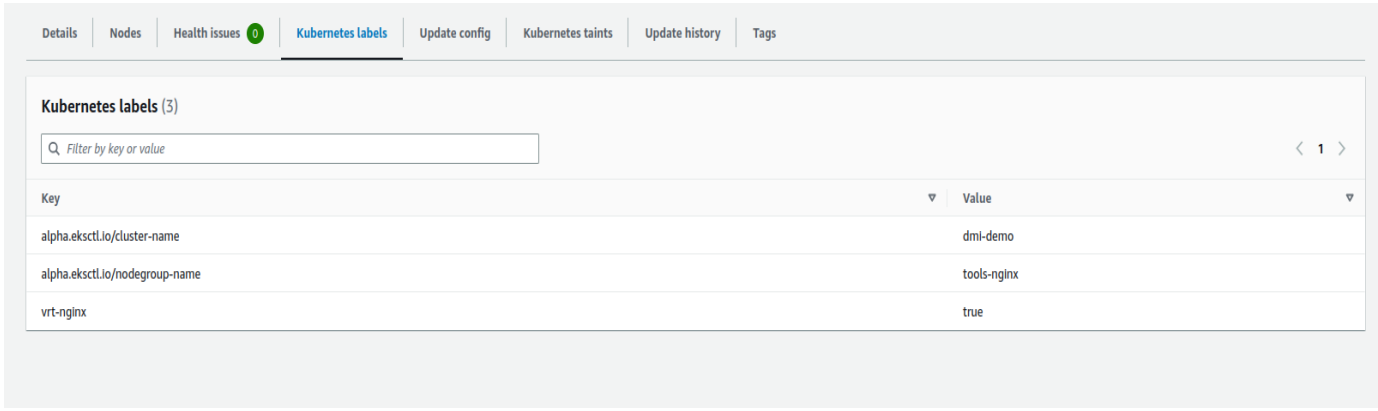
1.click on consul-monitoring nodegroup and click on edit button in right corner.
Key: **vrt-consul**
Value**: true**

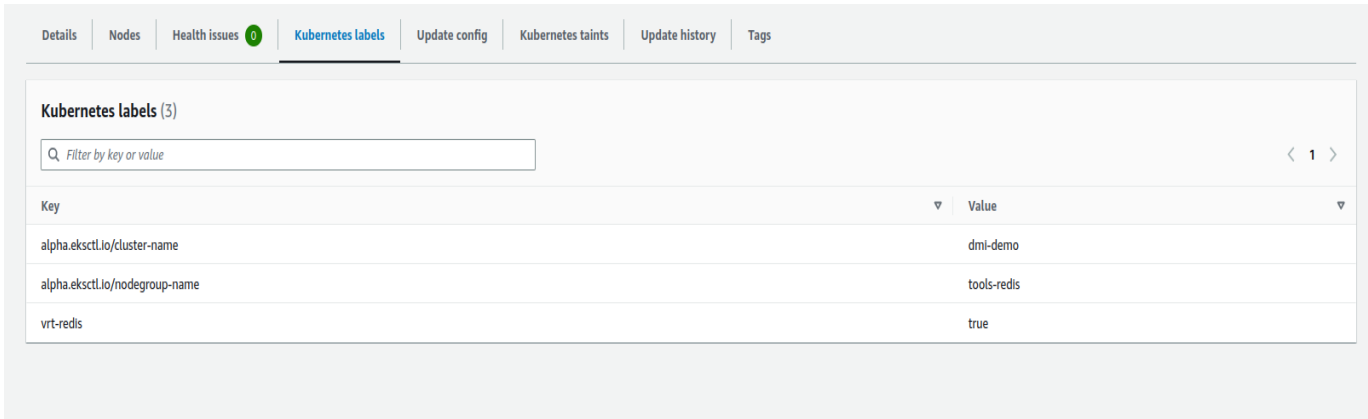


2. Go to the kubernetes labels section and add labels like in the image below. The label is vrt-consul you have to change the label name in another nodegroup. **For example, for an tools-elk node the label is vrt-elk.** Like below screenshots.

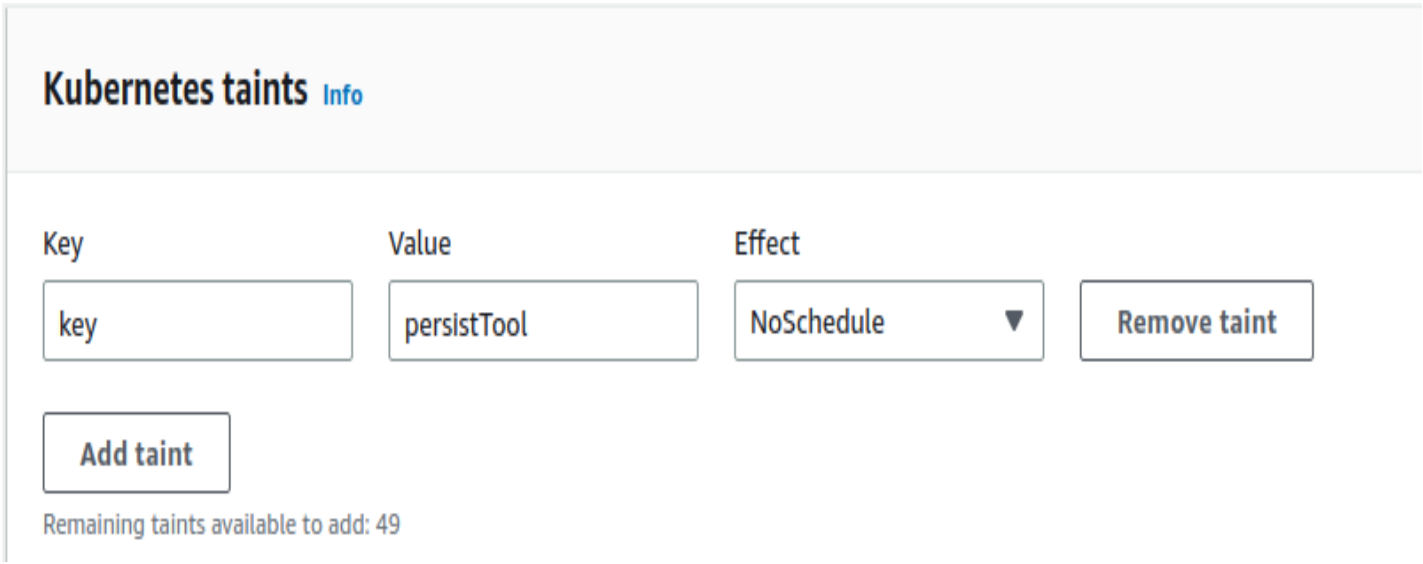

Kubernetes labels Info
Key
Value
alpha.eksctl.io/cluster-name
dmi-demo
Remove label
alpha.eksctl.io/nodegroup-name
consul-monitoring
Remove label
vrt-consul
true
Remove label
Add label
Remaining labels available to add: 47


Kubernetes labels (6)
Filter by key or value
1
Key
Value
alpha.eksctl.io/cluster-name
dmi-demo
alpha.eksctl.io/nodegroup-name
tools-elk
vrt-zookeeper
true
vrt-redis
true
vrt-kafka
true
vrt-elk
true


Details
Nodes
Health issues 0
Kubernetes labels
Update config
Kubernetes taints
Update history
Tags
Kubernetes labels (3)
Filter by key or value
1
Key
Value
alpha.eksctl.io/cluster-name
dmi-demo
alpha.eksctl.io/nodegroup-name
tools-nginx
vrt-nginx
true


Details
Nodes
Health issues 0
Kubernetes labels
Update config
Kubernetes taints
Update history
Tags
Kubernetes labels (3)
Filter by key or value
1
Key
Value
alpha.eksctl.io/cluster-name
dmi-demo
alpha.eksctl.io/nodegroup-name
tools-redis
vrt-redis
true

3. Go to the kubernetes taint section and add taints like in the image below and hit save button.

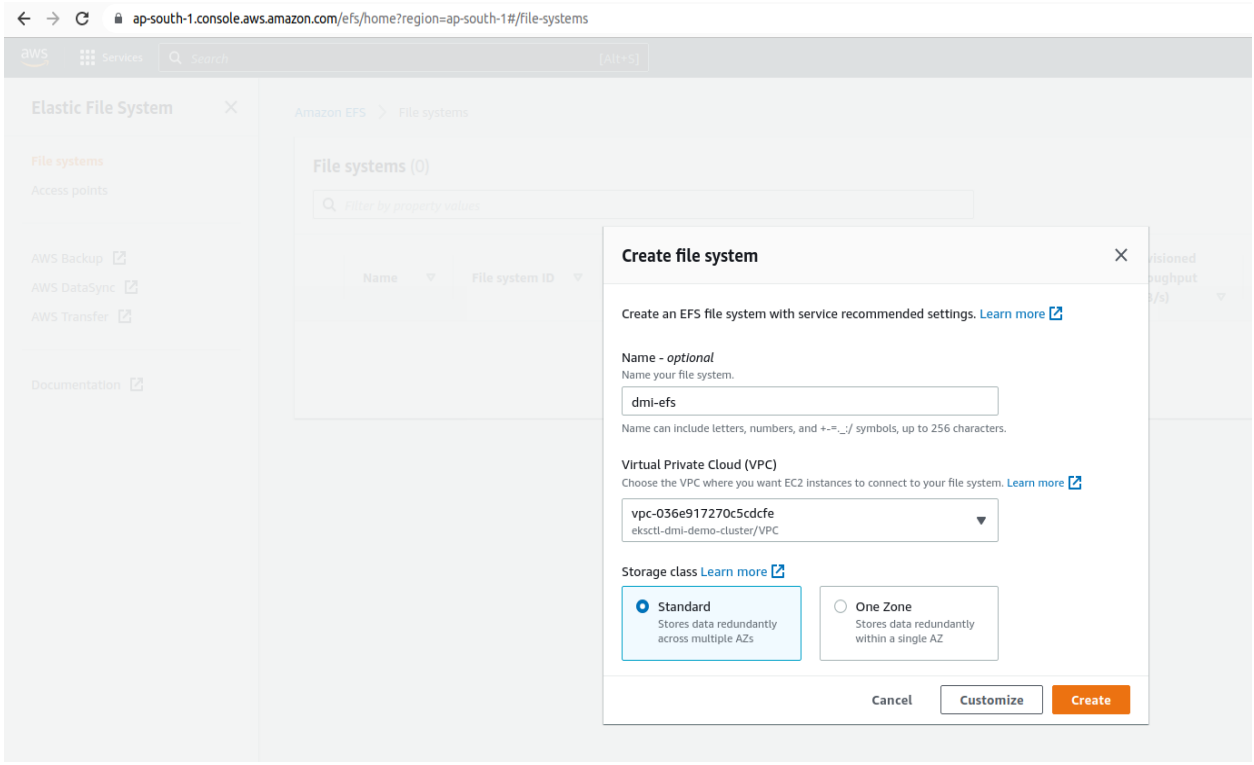


4. Follow this process for all node-groups except application node and custom application node.

Create EFS:

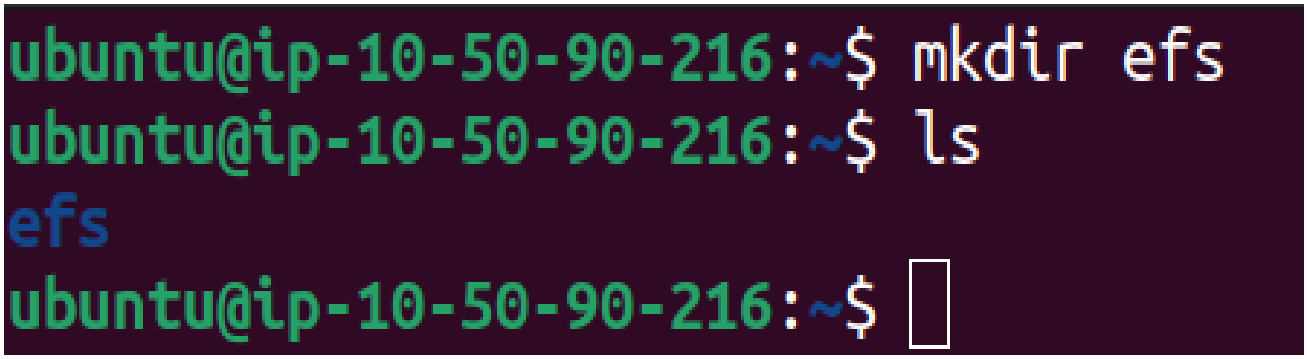
Stateful applications like (Elasticsearch, Logstash, Kafka, PostgreSQL, Consul, Zookeeper) store the data. In which data may be lost during restart pods.
To prevent this, We keeps data on NFS storage, Which is available on AWS service as a EFS.

1. EFS for persistence volume and persistent volume claim for deploying the tools.
2. Go to your aws console and search for EFS and create EFS and select your cluster vpc.

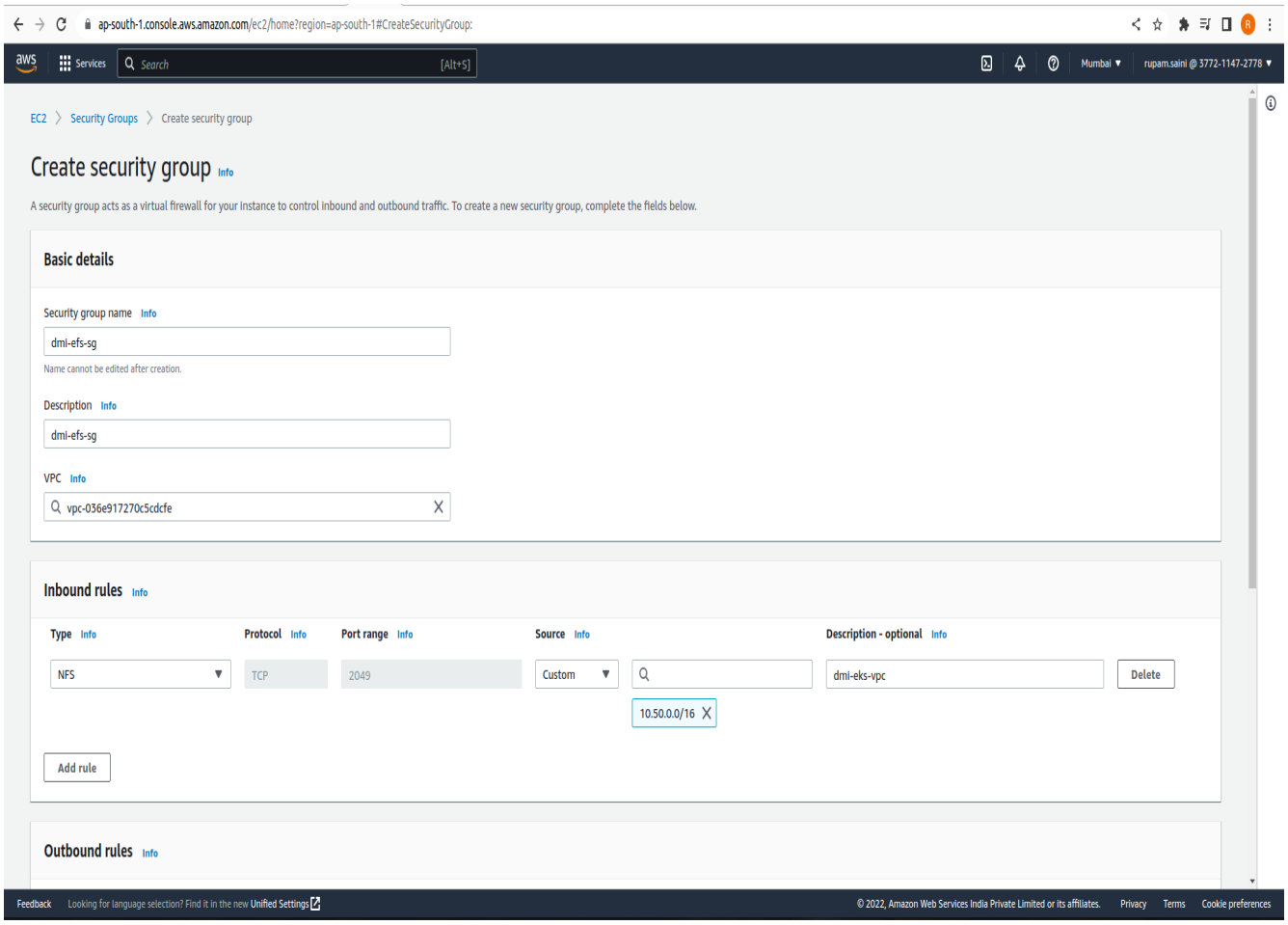


Jumpbox:

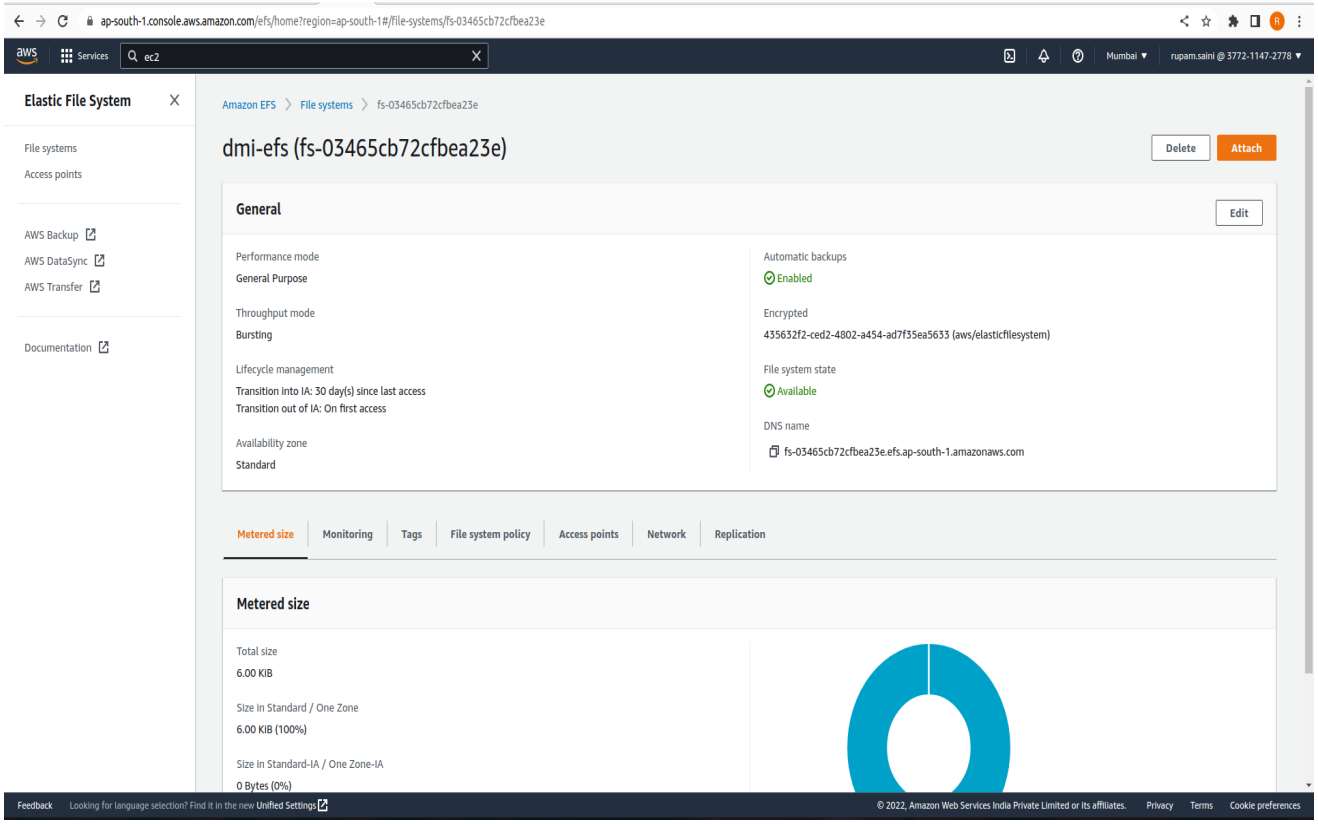
1. Create an ec2 instance into your cluster vpc public subnet.(search your vpc in cluster networking section)
2. Now login to your ec2 instance and create an efs folder on home directory.



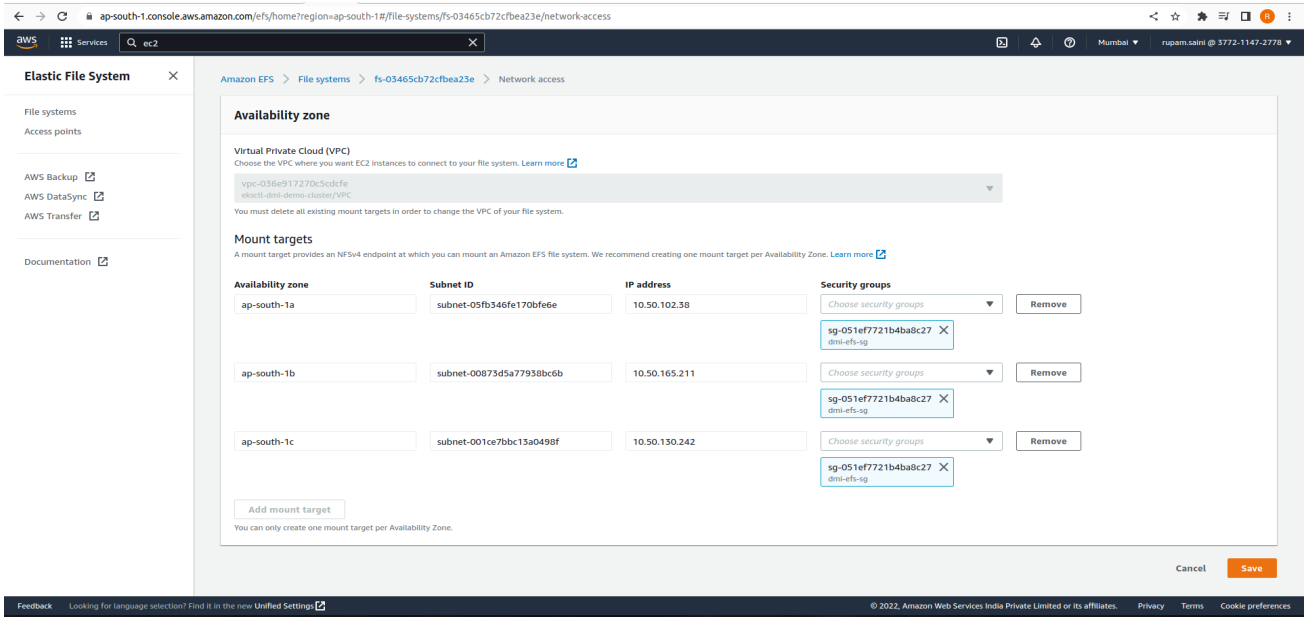
3. Now create a security group in your cluster vpc and allow the nfs inbound rule with your vpc cidr.



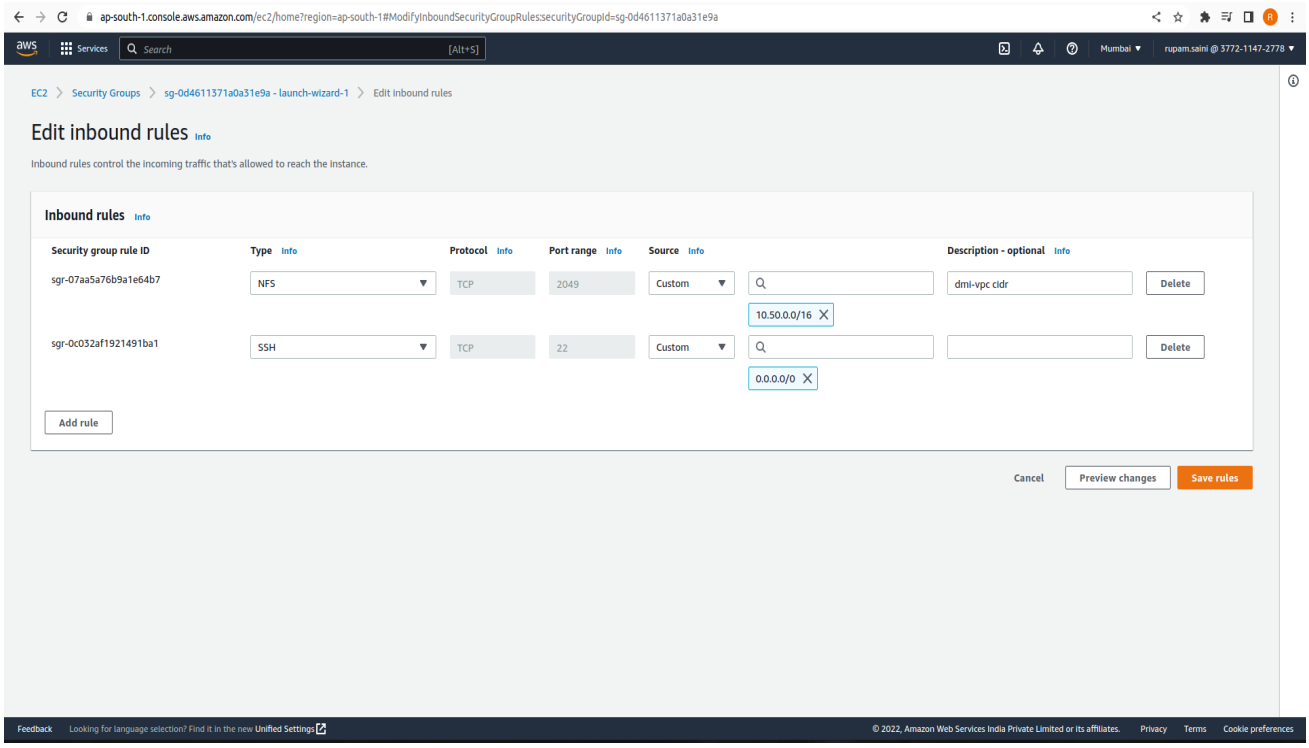
4. Now go to your aws console and go to the EFS dashboard.click on your EFS name. And go to the network section and add the security group which you created earlier.



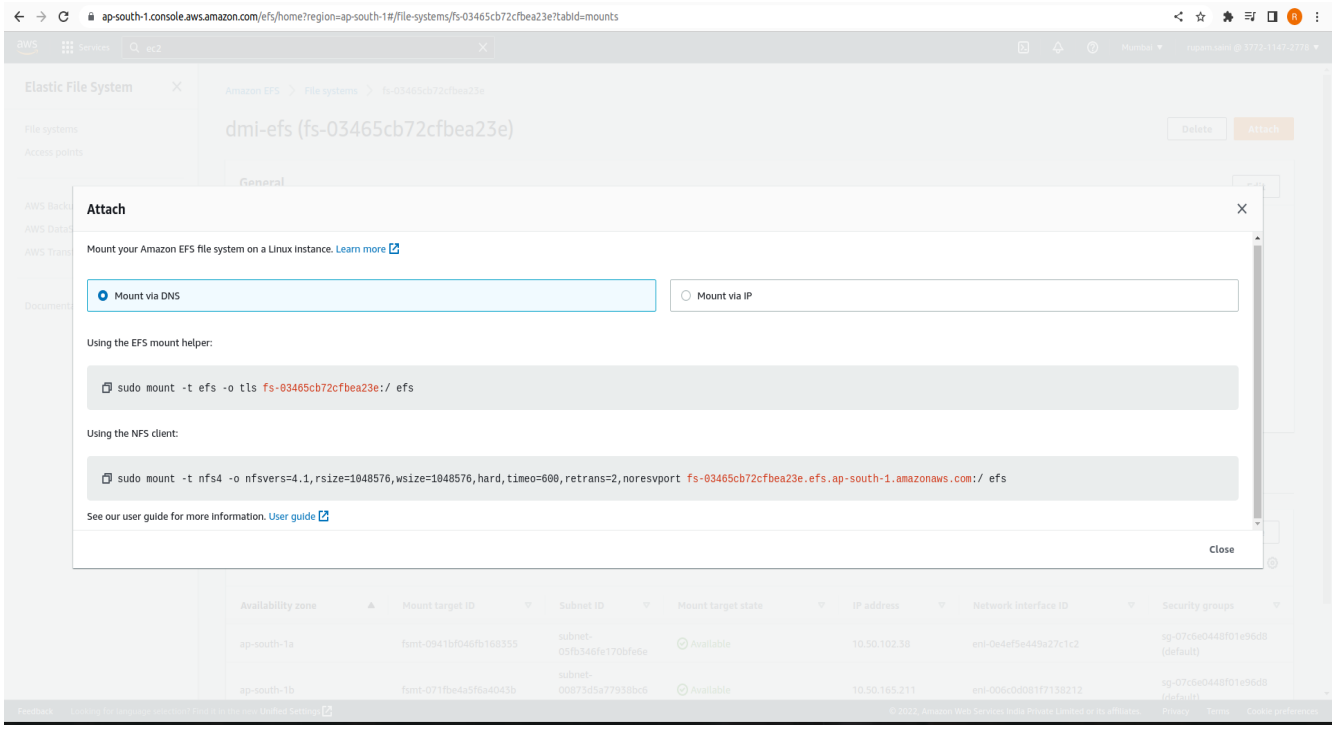
5. Add your security group in the networking section and click the save button.



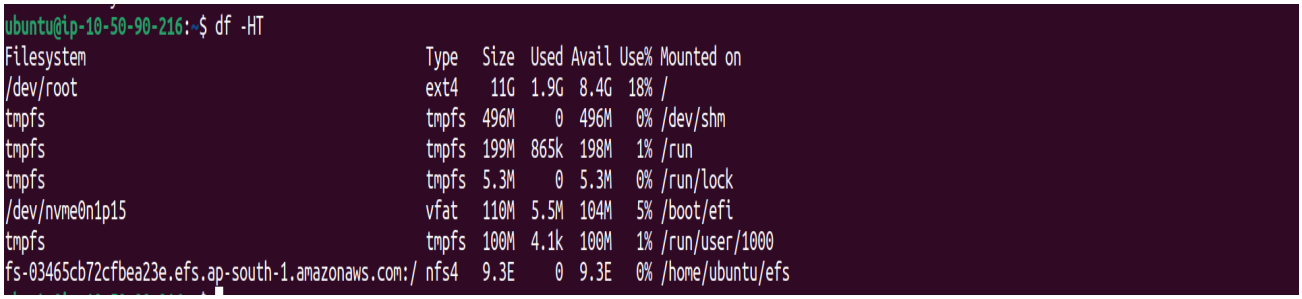
6. Now go to your jumpbox security group and allow NFS port 2049 on inbound rule with your EKS cluster VPC CIDR.



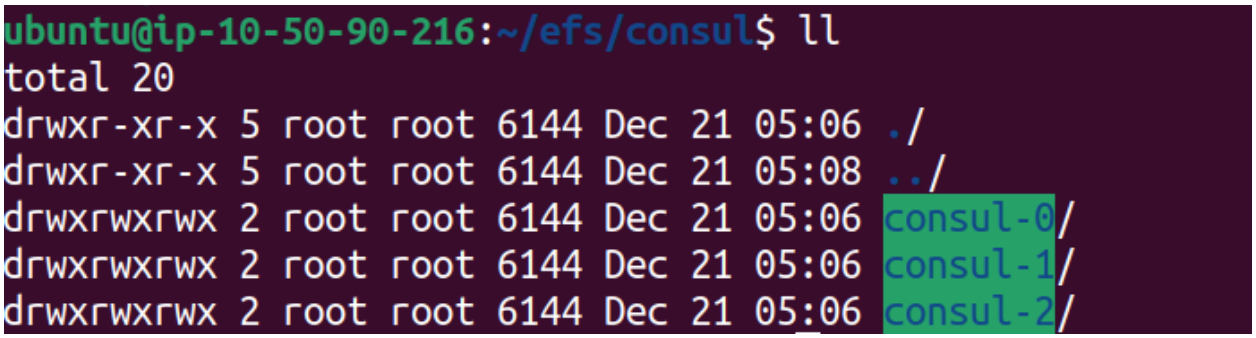
7. Now again go to your efs and click on your efs and in the right corner click on the **Attach** option.



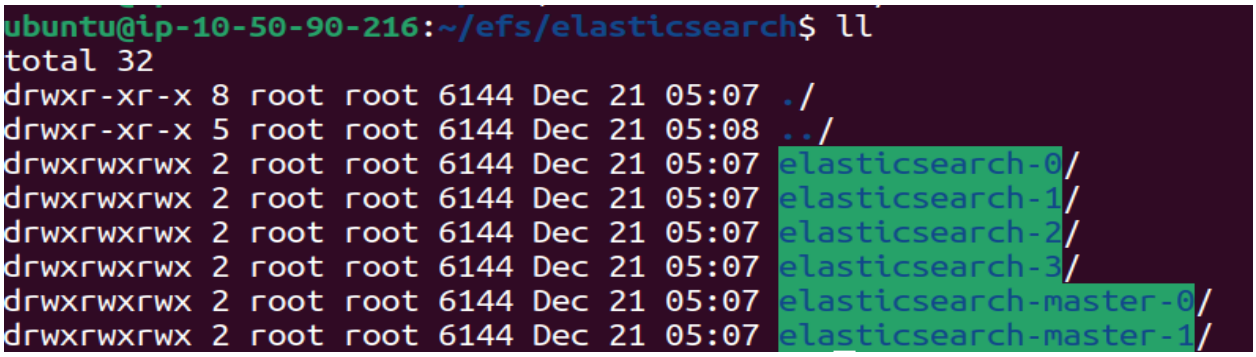
8. Now copy the command “using the nfs” client and paste on your jumpbox home directory and hit enter.
9. If you find “**mount: /home/ubuntu/efs: bad option; for several filesystems (e.g. nfs, cifs) you might need a /sbin/mount.<type> helper program.**”
   this error then runs below command and tries again.
   sudo apt update
   sudo apt install nfs-common
10. After this process you can verify your mount volume using the “**df -HT**” command and output should be like this.



11. You can check your efs id at the end of output in above image.
12. Now we have to create directories which are used for PV or PVC.
13. Create directories for consul like in image below and assign permission to then “sudo chmod 777”.



14. Create all directories like this.



```
ubuntu@ip-10-50-90-216:~/efs/kafka$ ll
total 24
drwxr-xr-x 6 root root 6144 Dec 21 05:12 ./
drwxr-xr-x 5 root root 6144 Dec 21 05:08 ../
drwxrwxrwx 2 root root 6144 Dec 21 05:12 kafka-0/
drwxrwxrwx 2 root root 6144 Dec 21 05:12 kafka-1/
drwxrwxrwx 2 root root 6144 Dec 21 05:12 kafka-2/
drwxrwxrwx 2 root root 6144 Dec 21 05:12 kafka-3/
```

```
ubuntu@ip-10-50-90-216:~/efs/logging-agent$ ll
total 20
drwxr-xr-x 5 root root 6144 Dec 21 05:13 ./
drwxr-xr-x 6 root root 6144 Dec 21 05:13 ../
drwxrwxrwx 2 root root 6144 Dec 21 05:13 logging-agent-0/
drwxrwxrwx 2 root root 6144 Dec 21 05:13 logging-agent-1/
drwxrwxrwx 2 root root 6144 Dec 21 05:13 logging-agent-2/
```

```
ubuntu@ip-10-50-90-216:~/efs/postgres$ ll
total 12
drwxr-xr-x 3 root root 6144 Dec 21 05:15 ./
drwxr-xr-x 8 root root 6144 Dec 21 05:17 ../
drwxrwxrwx 2 root root 6144 Dec 21 05:15 postgres-0/
```

```
ubuntu@ip-10-50-90-216:~/efs/redis$ ll
total 20
drwxr-xr-x 5 root root 6144 Dec 21 05:18 ./
drwxr-xr-x 8 root root 6144 Dec 21 05:17 ../
drwxrwxrwx 2 root root 6144 Dec 21 05:18 redis-0/
drwxrwxrwx 2 root root 6144 Dec 21 05:18 redis-1/
drwxrwxrwx 2 root root 6144 Dec 21 05:18 redis-2/
```

15. To install kubectl on jumpbox . follow this Document.
16. Then you have to install awscli using this command. “**sudo apt install awscli** “.
17. Now configure your aws credentials using “aws configure” command.
18. Alfer configure your aws credentials you have to update your cluster in jumpbox using below command.
    aws eks update-kubeconfig --name “your-cluster-name” --region “aws-region”
    Replace “your-cluster-name” and “aws-region” with your requirements.
    And your output is like the image below.

```
ubuntu@ip-10-50-90-216:~$ aws eks update-kubeconfig --name dmi-demo  --region ap-south-1
Added new context arn:aws:eks:ap-south-1:377211472778:cluster/dmi-demo to /home/ubuntu/.kube/config
```

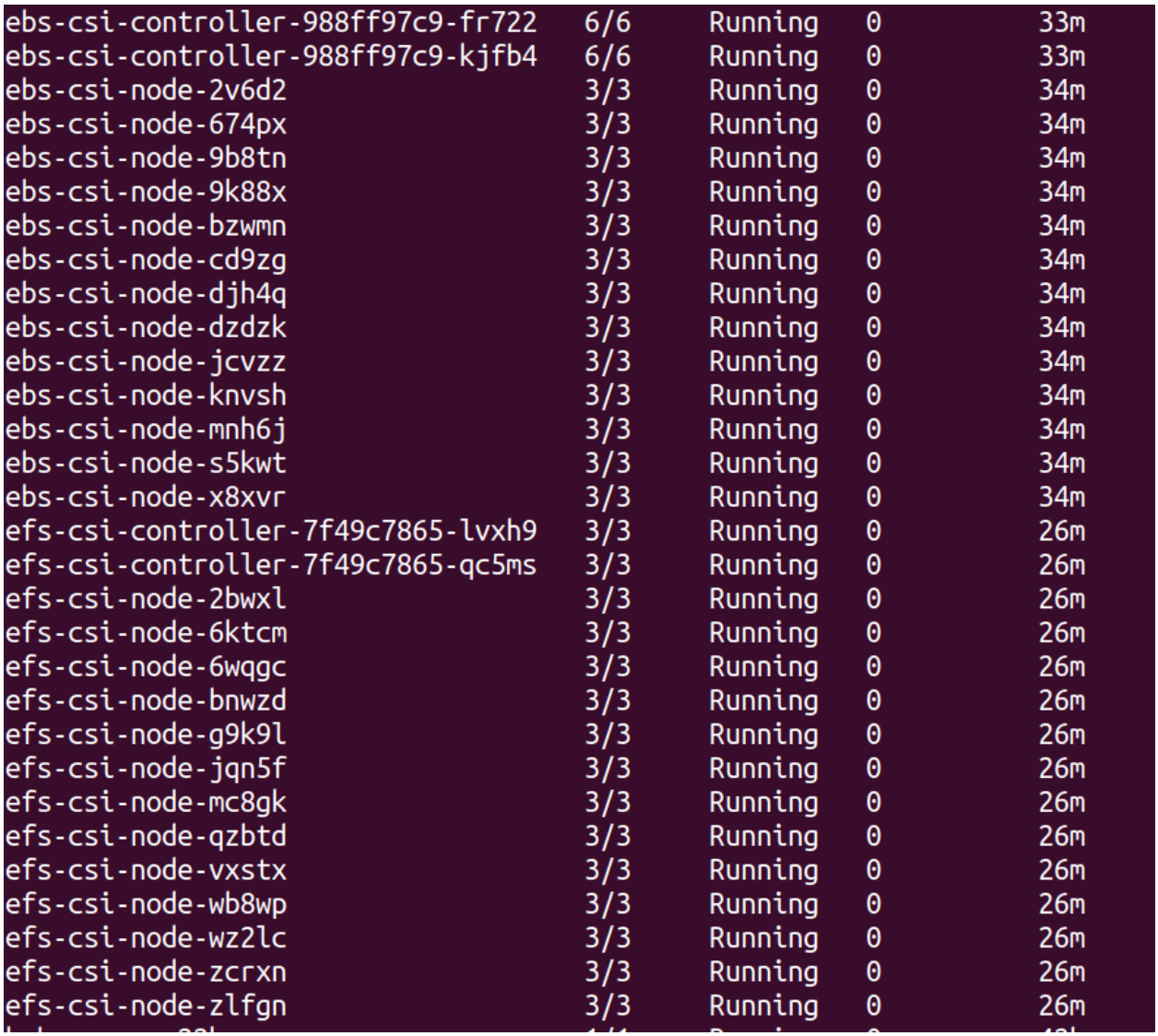
19. After these steps you can verify your cluster with “kubectl get ns” command.

```
ubuntu@ip-10-50-90-216:~$ kubectl get ns
NAME              STATUS   AGE
default           Active   42h
kube-node-lease   Active   42h
kube-public       Active   42h
kube-system       Active   42h
```

20. Now create a new namespace with any name you want.
kubectl create ns “namespace-name”

```
ubuntu@ip-10-50-90-216:~$ kubectl get ns
NAME              STATUS   AGE
default           Active   43h
dmi               Active   8s
kube-node-lease   Active   43h
kube-public       Active   43h
kube-system       Active   43h
```

21. Now you have to install aws CSI and EBS driver using this link.
22. After completing this step you can verify your driver using.
“**kubectl get pods -n kube-system**” and output should be like this.
23. Check these pods on your output .



```
ebs-csi-controller-988ff97c9-fr722   6/6     Running   0          33m
ebs-csi-controller-988ff97c9-kjfb4   6/6     Running   0          33m
ebs-csi-node-2v6d2                   3/3     Running   0          34m
ebs-csi-node-674px                   3/3     Running   0          34m
ebs-csi-node-9b8tn                   3/3     Running   0          34m
ebs-csi-node-9k88x                   3/3     Running   0          34m
ebs-csi-node-bzwmn                   3/3     Running   0          34m
ebs-csi-node-cd9zg                   3/3     Running   0          34m
ebs-csi-node-djh4q                   3/3     Running   0          34m
ebs-csi-node-dzdzk                   3/3     Running   0          34m
ebs-csi-node-jcvzz                   3/3     Running   0          34m
ebs-csi-node-knvsh                   3/3     Running   0          34m
ebs-csi-node-mnh6j                   3/3     Running   0          34m
ebs-csi-node-s5kwt                   3/3     Running   0          34m
ebs-csi-node-x8xvr                   3/3     Running   0          34m
efs-csi-controller-7f49c7865-lvxh9   3/3     Running   0          26m
efs-csi-controller-7f49c7865-qc5ms   3/3     Running   0          26m
efs-csi-node-2bwxl                   3/3     Running   0          26m
efs-csi-node-6ktcm                   3/3     Running   0          26m
efs-csi-node-6wqgc                   3/3     Running   0          26m
efs-csi-node-bnwzd                   3/3     Running   0          26m
efs-csi-node-g9k9l                   3/3     Running   0          26m
efs-csi-node-jqn5f                   3/3     Running   0          26m
efs-csi-node-mc8gk                   3/3     Running   0          26m
efs-csi-node-qzbtd                   3/3     Running   0          26m
efs-csi-node-vxstx                   3/3     Running   0          26m
efs-csi-node-wb8wp                   3/3     Running   0          26m
efs-csi-node-wz2lc                   3/3     Running   0          26m
efs-csi-node-zcrxn                   3/3     Running   0          26m
efs-csi-node-zlfgn                   3/3     Running   0          26m
```

24. After verifying these you have to deploy the pv and pvc form attach file. With **kubectl apply -f <filename> -n <namespace>**

```
ubuntu@ip-10-50-90-216:~/vrt-manifest-yaml/vrt-pvc$ kubectl apply -f . -n dmi
persistentvolume/efs-consul-0 created
persistentvolumeclaim/data-consul-0 created
persistentvolume/efs-consul-1 created
persistentvolumeclaim/data-consul-1 created
persistentvolume/efs-consul-2 created
persistentvolumeclaim/data-consul-2 created
persistentvolume/efs-consul-3 created
persistentvolumeclaim/data-consul-3 created
persistentvolume/efs-consul-4 created
persistentvolumeclaim/data-consul-4 created
persistentvolume/efs-elasticsearch-data-0 created
persistentvolumeclaim/data-elasticsearch-data-0 created
persistentvolume/efs-elasticsearch-data-1 created
persistentvolumeclaim/data-elasticsearch-data-1 created
persistentvolume/efs-elasticsearch-data-2 created
persistentvolumeclaim/data-elasticsearch-data-2 created
persistentvolume/efs-elasticsearch-data-3 created
persistentvolumeclaim/data-elasticsearch-data-3 created
persistentvolume/efs-elasticsearch-master-0 created
persistentvolumeclaim/data-elasticsearch-master-0 created
persistentvolume/efs-elasticsearch-master-1 created
persistentvolumeclaim/data-elasticsearch-master-1 created
persistentvolume/efs-kafka-0 created
persistentvolumeclaim/data-kafka1-0 created
persistentvolume/efs-kafka-1 created
persistentvolumeclaim/data-kafka1-1 created
persistentvolume/efs-kafka-2 created
persistentvolumeclaim/data-kafka1-2 created
persistentvolume/efs-kafka-3 created
persistentvolumeclaim/data-kafka1-3 created
persistentvolume/efs-kafka-4 created
```

Helm chart:-

1. Now you have to deploy the helm chart from the given yaml file and below commands.
   “helm repo add bitnami https://charts.bitnami.com/bitnami”
   “helm install consul -f consul.yaml bitnami/consul -n dmi --version 10.7.7”
   Run these commands where you store the yamls files .
   Your output should be like this.

```
ubuntu@ip-10-50-90-216:~/vrt-manifest-yaml/helm-charts$ helm install consul -f consul.yaml bitnami/consul -n vrt --version 10.7.7 -n dmi
NAME: consul
LAST DEPLOYED: Wed Dec 21 07:23:22 2022
NAMESPACE: dmi
STATUS: deployed
REVISION: 1
TEST SUITE: None
NOTES:
CHART NAME: consul
CHART VERSION: 10.7.7
APP VERSION: 1.12.3

  ** Please be patient while the chart is being deployed **

  Consul can be accessed within the cluster on port 8300 at consul-headless.dmi.svc.cluster.local

In order to access to the Consul Web UI:

1. Get the Consul URL by running these commands:

    kubectl port-forward --namespace dmi svc/consul-ui 80:80
    echo "Consul URL: http://127.0.0.1:80"

2. Access ASP.NET Core using the obtained URL.

Please take into account that you need to wait until a cluster leader is elected before using the Consul Web UI.

In order to check the status of the cluster you can run the following command:

    kubectl exec -it consul-0 -- consul members

Furthermore, to know which Consul node is the cluster leader run this other command:
```

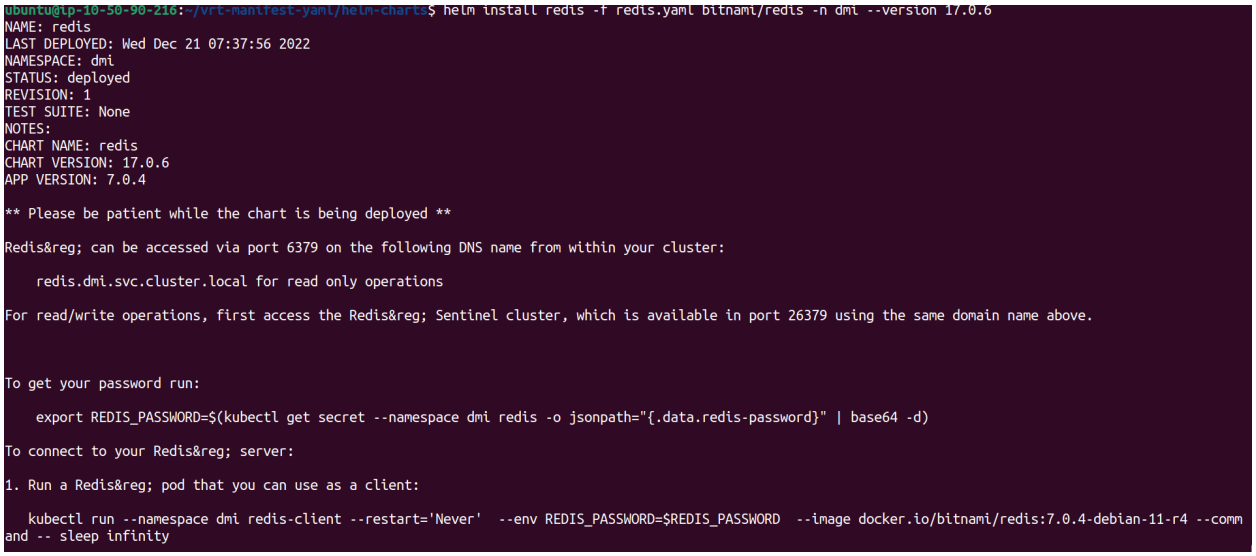
helm install zokeeper -f zookeeper.yaml bitnami/zookeeper -n dmi --version 10.0.1

```
ubuntu@ip-10-50-90-216:~/vrt-manifest-yaml/helm-charts$ helm install zokeeper -f zookeeper.yaml bitnami/zookeeper -n dmi --version 10.0.1
NAME: zokeeper
LAST DEPLOYED: Wed Dec 21 07:34:30 2022
NAMESPACE: dmi
STATUS: deployed
REVISION: 1
TEST SUITE: None
NOTES:
CHART NAME: zookeeper
CHART VERSION: 10.0.1
APP VERSION: 3.8.0

** Please be patient while the chart is being deployed **

ZooKeeper can be accessed via port 2181 on the following DNS name from within your cluster:

    zoo1.dmi.svc.cluster.local

To connect to your ZooKeeper server run the following commands:

    export POD_NAME=$(kubectl get pods --namespace dmi -l "app.kubernetes.io/name=zookeeper,app.kubernetes.io/instance=zokeeper,app.kubernetes.io/component=zo
okeeper" -o jsonpath="{.items[0].metadata.name}")
    kubectl exec -it $POD_NAME -- zkCli.sh

To connect to your ZooKeeper server from outside the cluster execute the following commands:

    kubectl port-forward --namespace dmi svc/zoo1 2181:2181 &
    zkCli.sh 127.0.0.1:2181
```

helm install kafka -f kafka.yaml bitnami/kafka -n dmi --version 18.0.3

```
ubuntu@ip-10-50-90-216:~/vrt-manifest-yaml/helm-charts$ helm install kafka -f kafka.yaml bitnami/kafka -n dmi --version 18.0.3
NAME: kafka
LAST DEPLOYED: Wed Dec 21 07:36:03 2022
NAMESPACE: dmi
STATUS: deployed
REVISION: 1
TEST SUITE: None
NOTES:
CHART NAME: kafka
CHART VERSION: 18.0.3
APP VERSION: 3.2.0

** Please be patient while the chart is being deployed **

Kafka can be accessed by consumers via port 9092 on the following DNS name from within your cluster:

    kafka1.dmi.svc.cluster.local

Each Kafka broker can be accessed by producers via port 9092 on the following DNS name(s) from within your cluster:

    kafka1-0.kafka1-headless.dmi.svc.cluster.local:9092
    kafka1-1.kafka1-headless.dmi.svc.cluster.local:9092
    kafka1-2.kafka1-headless.dmi.svc.cluster.local:9092

To create a pod that you can use as a Kafka client run the following commands:

    kubectl run kafka1-client --restart='Never' --image docker.io/bitnami/kafka:3.2.0-debian-11-r12 --namespace dmi --command -- sleep infinity
    kubectl exec --tty -i kafka1-client --namespace dmi -- bash

    PRODUCER:
        kafka-console-producer.sh \
```

helm install redis -f redis.yaml bitnami/redis -n dmi --version 17.0.6



helm install elasticsearch -f elasticsearch.yaml bitnami/elasticsearch -n dmi --version 18.2.6

Now you have to install aws alb ingress controller with below documentation.

1. Setup IAM for ServiceAccount

   Create IAM OIDC provider

```
eksctl utils associate-iam-oidc-provider \
  --region <aws-region> \
  --cluster <your-cluster-name> \
  --approve
```

2. Create an IAM policy called AWSLoadBalancerControllerIAMPolicy

```
aws iam create-policy \

  --policy-name AWSLoadBalancerControllerIAMPolicy \

  --policy-document file://iam-policy.json
```

```
ubuntu@ip-10-50-90-216:~$ aws iam create-policy --policy-name AWSLoadBalancerControllerIAMPolicy --policy-document file://iam-policy.json
POLICY  arn:aws:iam::[redacted]:policy/AWSLoadBalancerControllerIAMPolicy     0       2023-01-04T07:23:20+00:00       v1      True    /       0
        ANPAVPU4OJOFPNSJPWQRP   AWSLoadBalancerControllerIAMPolicy      2023-01-04T07:23:20+00:00
ubuntu@ip-10-50-90-216:~$ eksctl create iamserviceaccount \
> --cluster=dmi-demo \
> --namespace=dmi \
> --name=aws-load-balancer-controller \
> --attach-policy-arn=arn:aws:iam::[redacted]:policy/AWSLoadBalancerControllerIAMPolicy \
> --approve
2023-01-04 07:28:07 [ℹ]  1 iamserviceaccount (dmi/aws-load-balancer-controller) was included (based on the include/exclude rules)
2023-01-04 07:28:07 [!]  serviceaccounts that exist in Kubernetes will be excluded, use --override-existing-serviceaccounts to override
2023-01-04 07:28:07 [ℹ]  1 task: {
    2 sequential sub-tasks: {
        create IAM role for serviceaccount "dmi/aws-load-balancer-controller",
        create serviceaccount "dmi/aws-load-balancer-controller",
    } }2023-01-04 07:28:07 [ℹ]  building iamserviceaccount stack "eksctl-dmi-demo-addon-iamserviceaccount-dmi-aws-load-balancer-controller"
2023-01-04 07:28:08 [ℹ]  deploying stack "eksctl-dmi-demo-addon-iamserviceaccount-dmi-aws-load-balancer-controller"
2023-01-04 07:28:08 [ℹ]  waiting for CloudFormation stack "eksctl-dmi-demo-addon-iamserviceaccount-dmi-aws-load-balancer-controller"
2023-01-04 07:28:38 [ℹ]  waiting for CloudFormation stack "eksctl-dmi-demo-addon-iamserviceaccount-dmi-aws-load-balancer-controller"
2023-01-04 07:29:23 [ℹ]  waiting for CloudFormation stack "eksctl-dmi-demo-addon-iamserviceaccount-dmi-aws-load-balancer-controller"
2023-01-04 07:29:23 [ℹ]  created serviceaccount "dmi/aws-load-balancer-controller"
ubuntu@ip-10-50-90-216:~$ k9s
ubuntu@ip-10-50-90-216:~$ k9s
ubuntu@ip-10-50-90-216:~$
ubuntu@ip-10-50-90-216:~$ helm install aws-load-balancer-controller eks/aws-load-balancer-controller --set clusterName=dmi-demo -n dmi --set serviceA
ccount.create=false --set serviceAccount.name=aws-load-balancer-controller
NAME: aws-load-balancer-controller
LAST DEPLOYED: Wed Jan  4 07:30:32 2023
NAMESPACE: dmi
STATUS: deployed
REVISION: 1
TEST SUITE: None
NOTES:
AWS Load Balancer controller installed!
ubuntu@ip-10-50-90-216:~$ k9s
```

3. Download IAM policy for the AWS Load Balancer Controller

curl -o iam-policy.json https://raw.githubusercontent.com/kubernetes-sigs/aws-load-balance

```
ubuntu@ip-10-50-90-216:~$ curl -o iam-policy.json https://raw.githubusercontent.com/kubernetes-sigs/aws-load-balancer-controller/main/docs/install/ia
m_policy.json
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100  7617  100  7617    0     0  15840      0 --:--:-- --:--:-- --:--:-- 15868
```

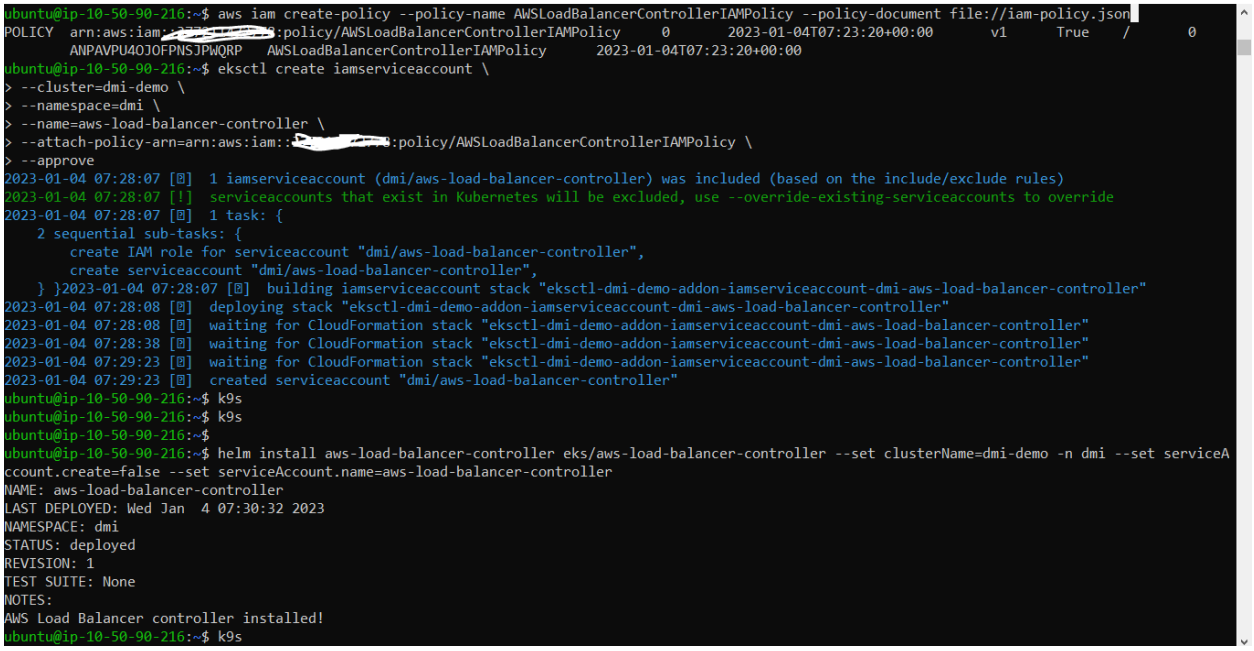
Take note of the policy ARN that is returned

4. Create a IAM role and ServiceAccount for the Load Balancer controller, use the ARN from the step above

```
eksctl create iamserviceaccount \
--cluster=<cluster-name> \
--namespace=kube-system \
--name=aws-load-balancer-controller \
--attach-policy-arn=arn:aws:iam::<AWS_ACCOUNT_ID>:policy/AWSLoadBalancerControllerIAMPolicy \
--approve
```

5. Now Install Helm chart for ALB Ingress controller:

helm install aws-load-balancer-controller eks/aws-load-balancer-controller --set clusterName=dmi-demo -n dmi --set serviceAccount.create=false --set serviceAccount.name=aws-load-balancer-controller



And your output should be like above images

Ingres:

Now you have to create an ingress rule with the given ingres.yaml file. But if you want your custom ingres rule you have to add rules in ingres file.

```
- path: /<path>/
  pathType: Prefix
  backend:
    service:
      name: <name>
      port:
        number: <portnumber>
```

Please change the values of <path> <name> <portnumber> according to your need. And you have to change annotation in ingres.yaml file

```
  annotations:
    kubernetes.io/ingress.class: alb
    alb.ingress.kubernetes.io/listen-ports: '[{"HTTP": 80},{"HTTPS": 443}]'
    alb.ingress.kubernetes.io/action.ssl-redirect: '{"Type": "redirect", "RedirectConfig":
{"Protocol": "HTTPS", "Port": 443, "StatusCode": "HTTP 301"}}'
```

alb.ingress.kubernetes.io/scheme: internet-facing
alb.ingress.kubernetes.io/target-type: ip
alb.ingress.kubernetes.io/subnetes: <subnetsids>
alb.ingress.kubernetes.io/certificate-arn: <certificatearn>
alb.ingress.kubernetes.io/ssl-policy: ELBSecurityPolicy-TLS-1-2-Ext-2018-06

Replace these two lines **<subnetsids>** lines with **<certificatearn>** with your subnet and certificate arn. And apply these yaml on jumpbox.

Kubectl apply -f ingres.yaml -n <namespace>

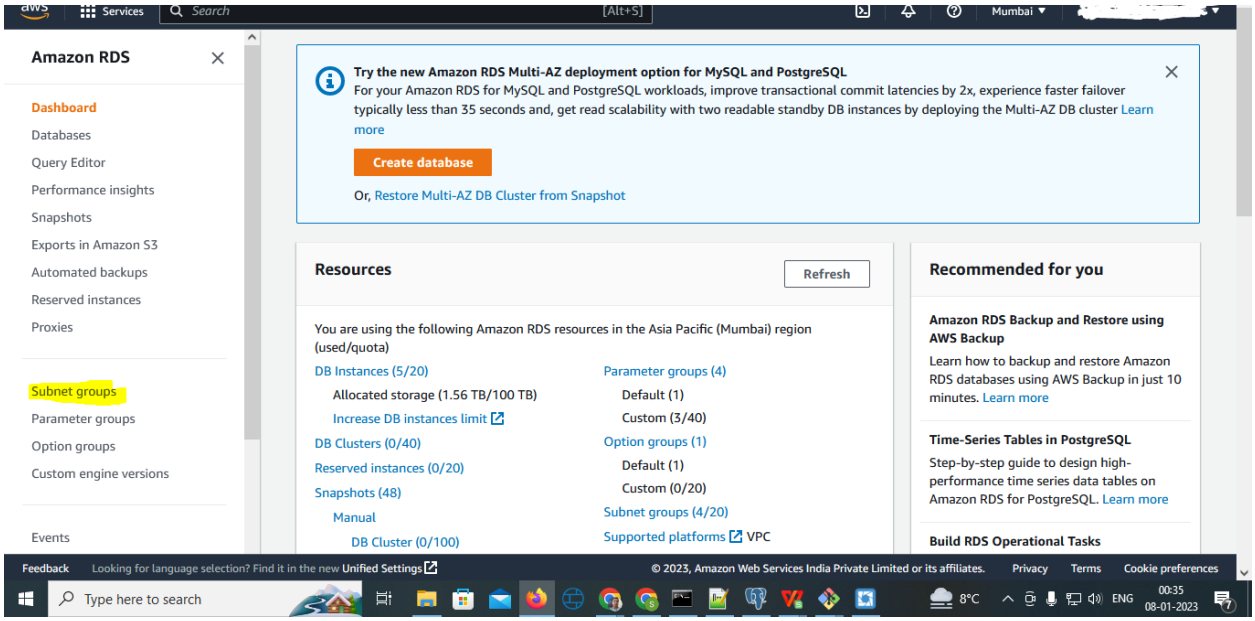
And check the load balancer in your aws account.if it is created or not.

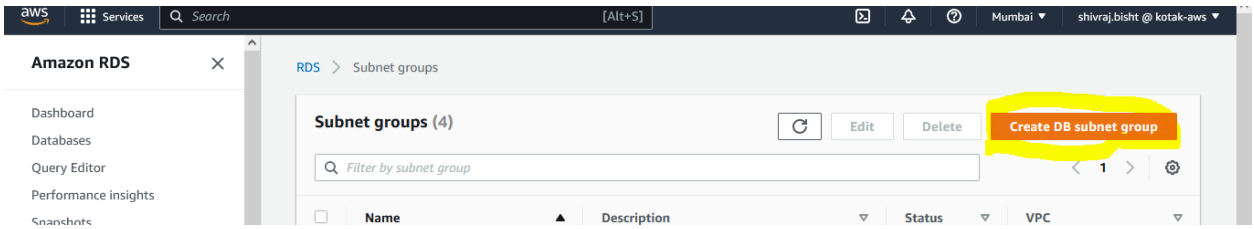
Create

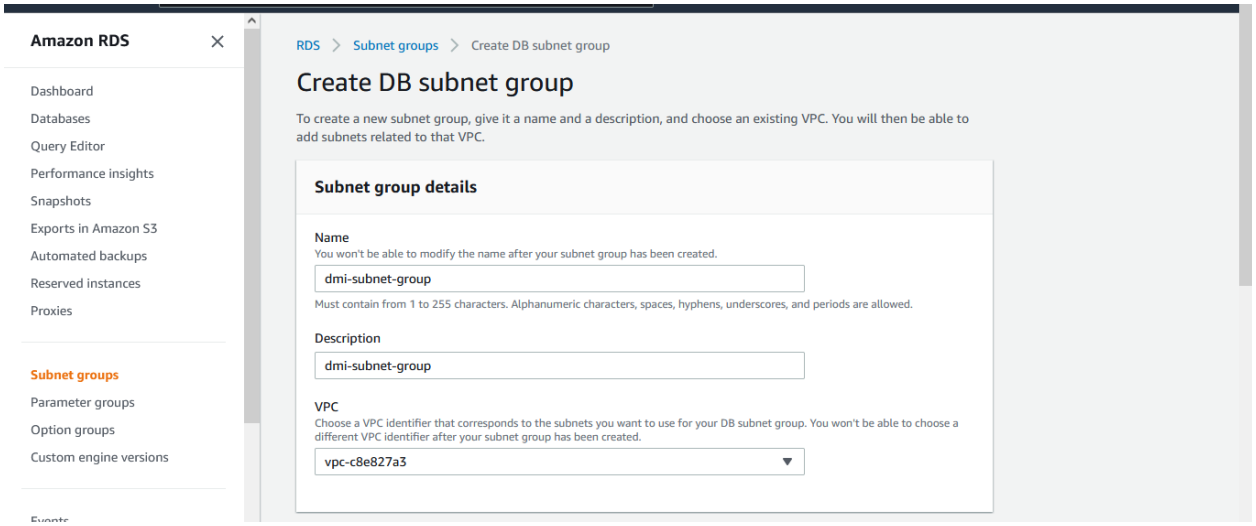
RDS Setup:

Now you have to create rds for our custom applications.please find below steps to create rds.

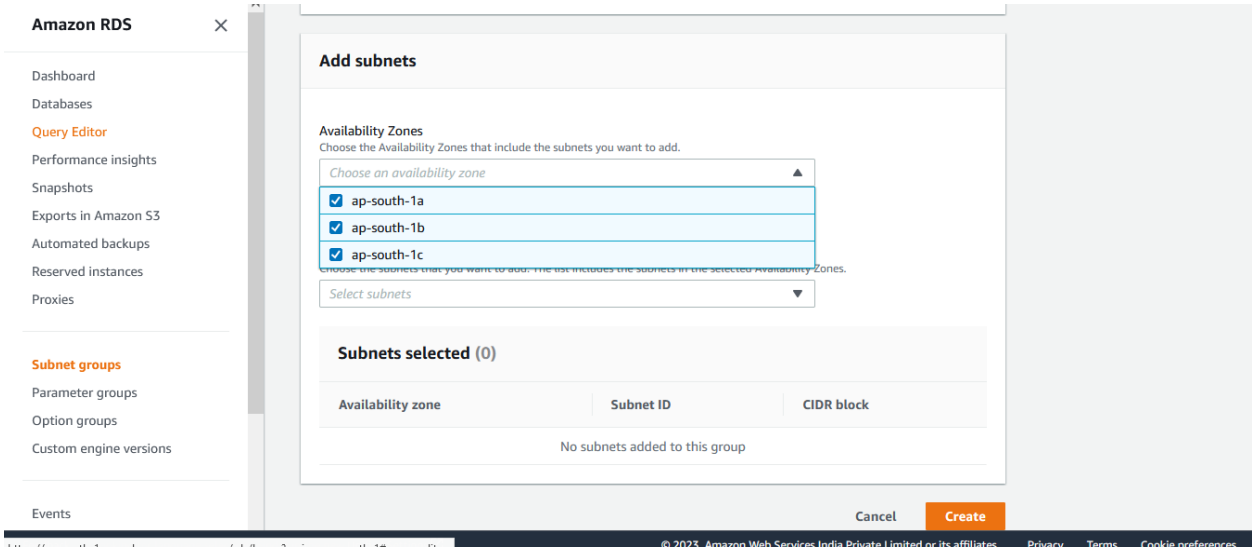
1.Sign in to the AWS Management Console and open the Amazon RDS.
2. Create RDS Subnet group.

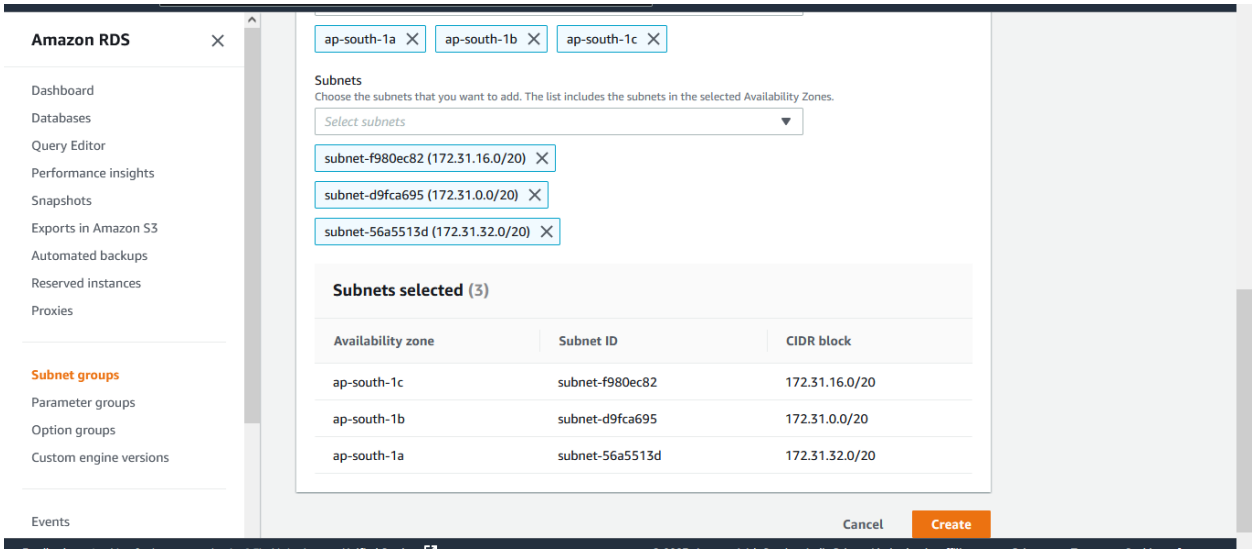






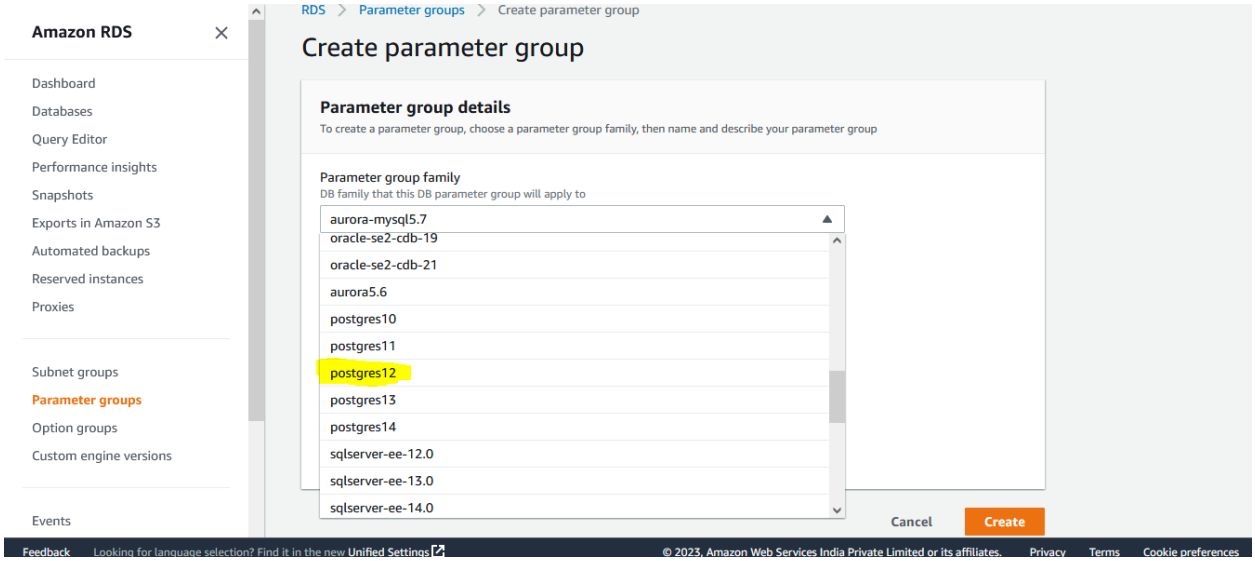
Add subnet and Availability Zones and choose private subnets of EKS cluster based VPC.

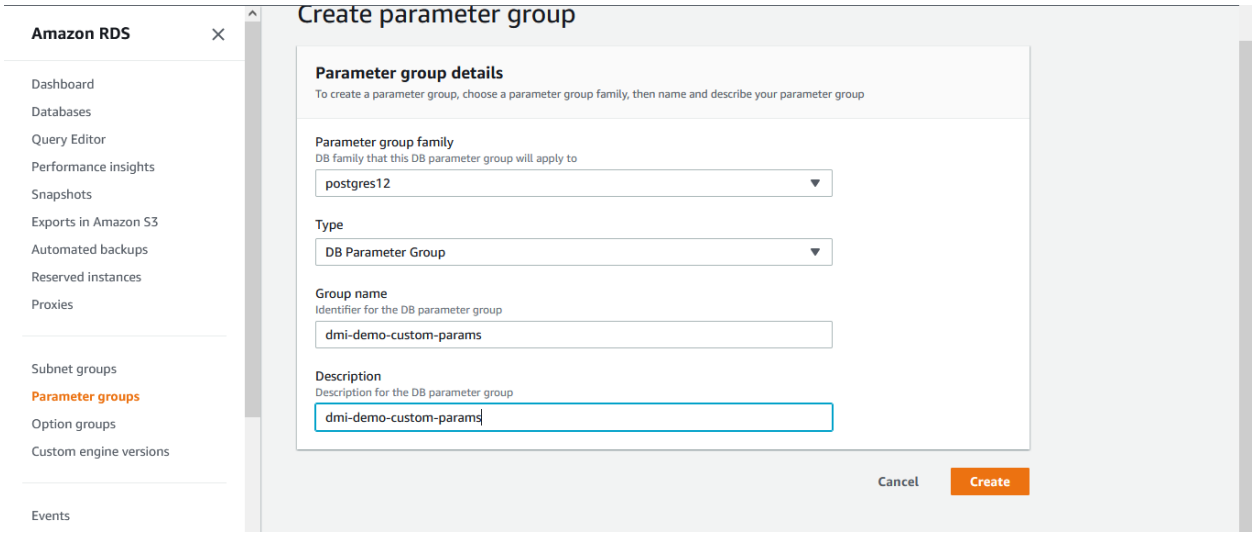




3. Create a Parameter group for postgreSQL database to customize your DB parameters according to your requirements.

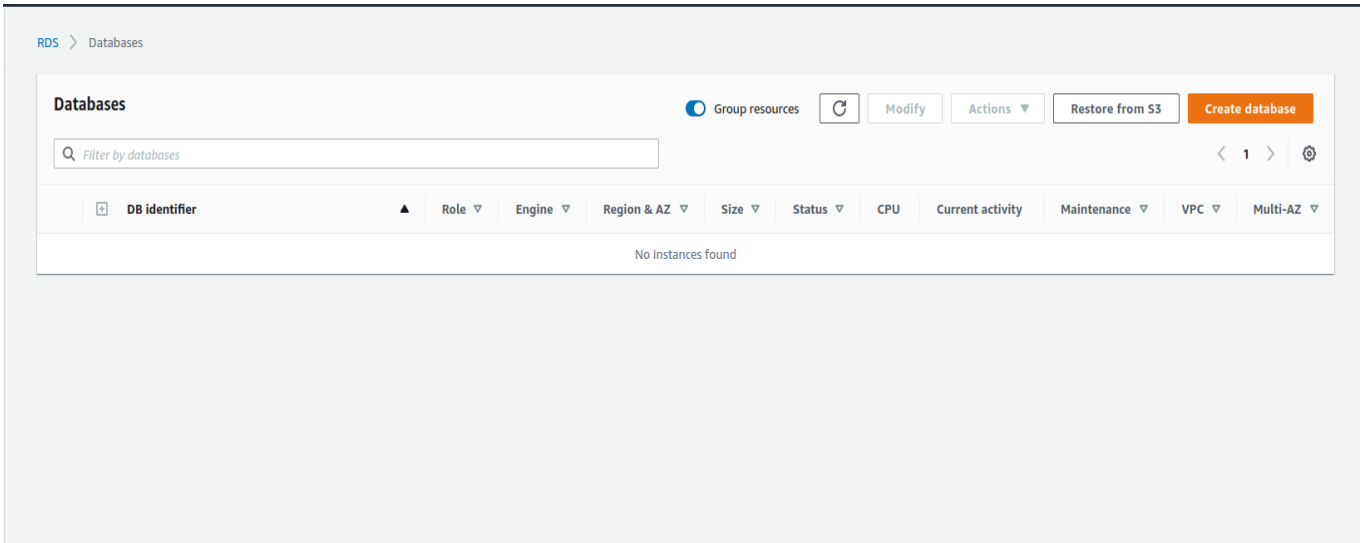
Select the Postgres as a DB Engine and version with 12.



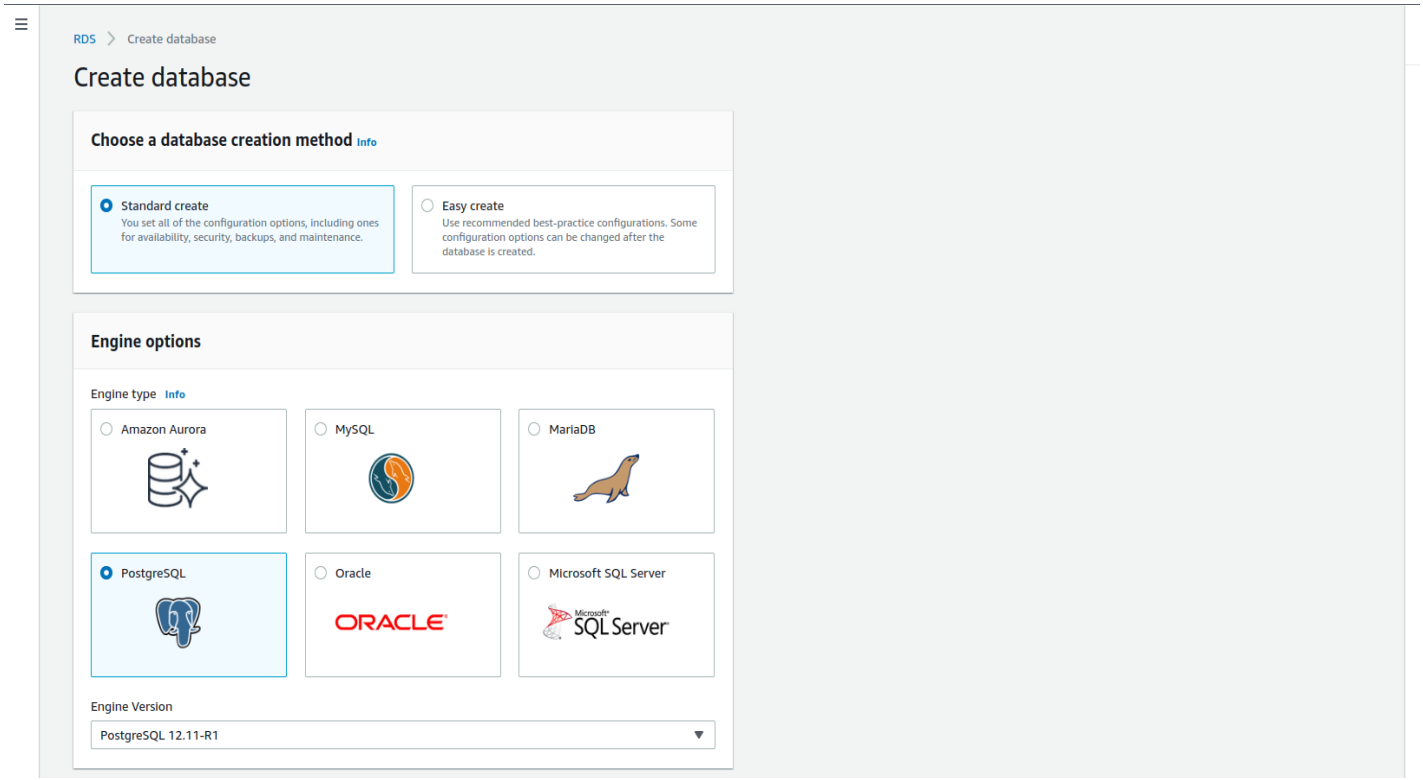


Once you select the all option click on Create for parameter group.

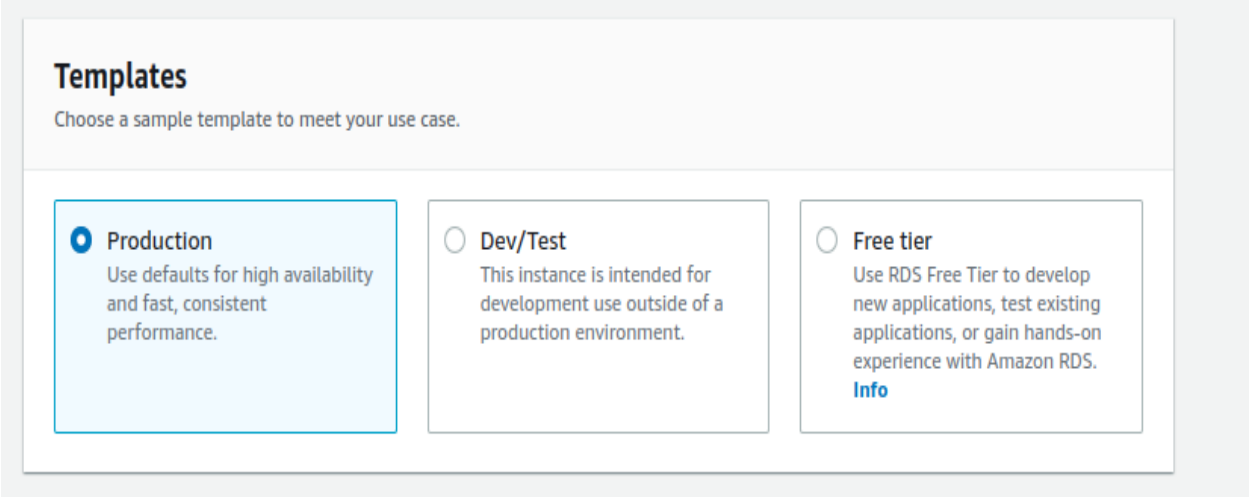
4.In the upper-right corner of the Amazon RDS console, choose the AWS Region in which you want to create the DB instance.
5.Now click on the create option on the right side .



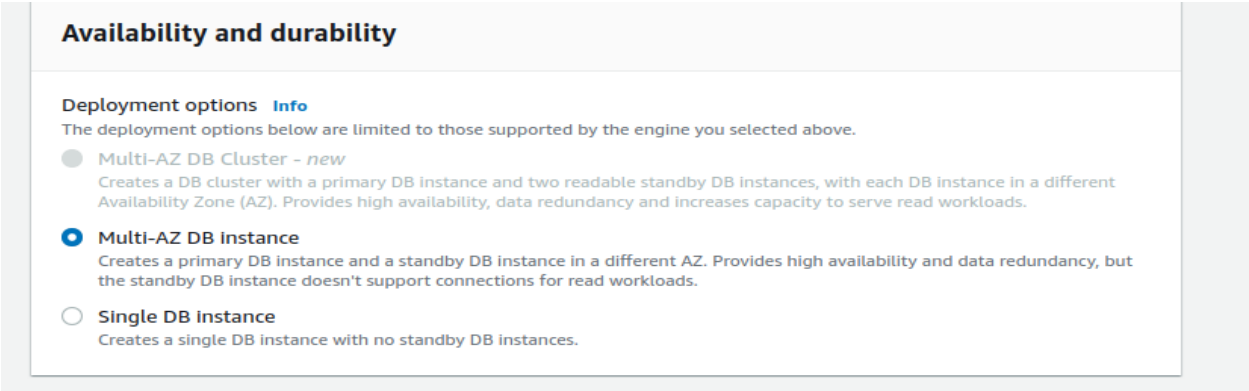
6. .And select these options like in the image below.



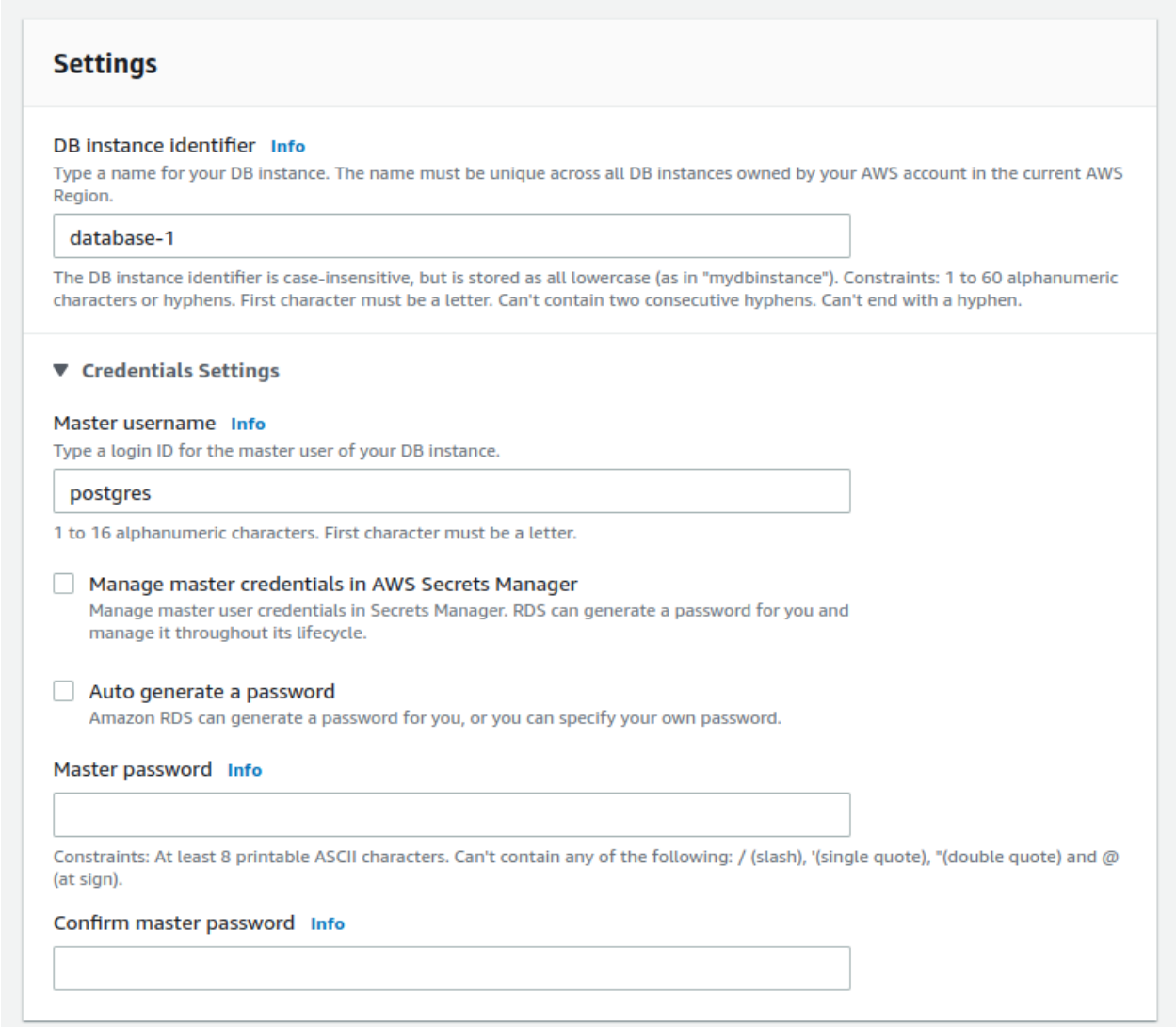
7. .And select templates option according to your need.



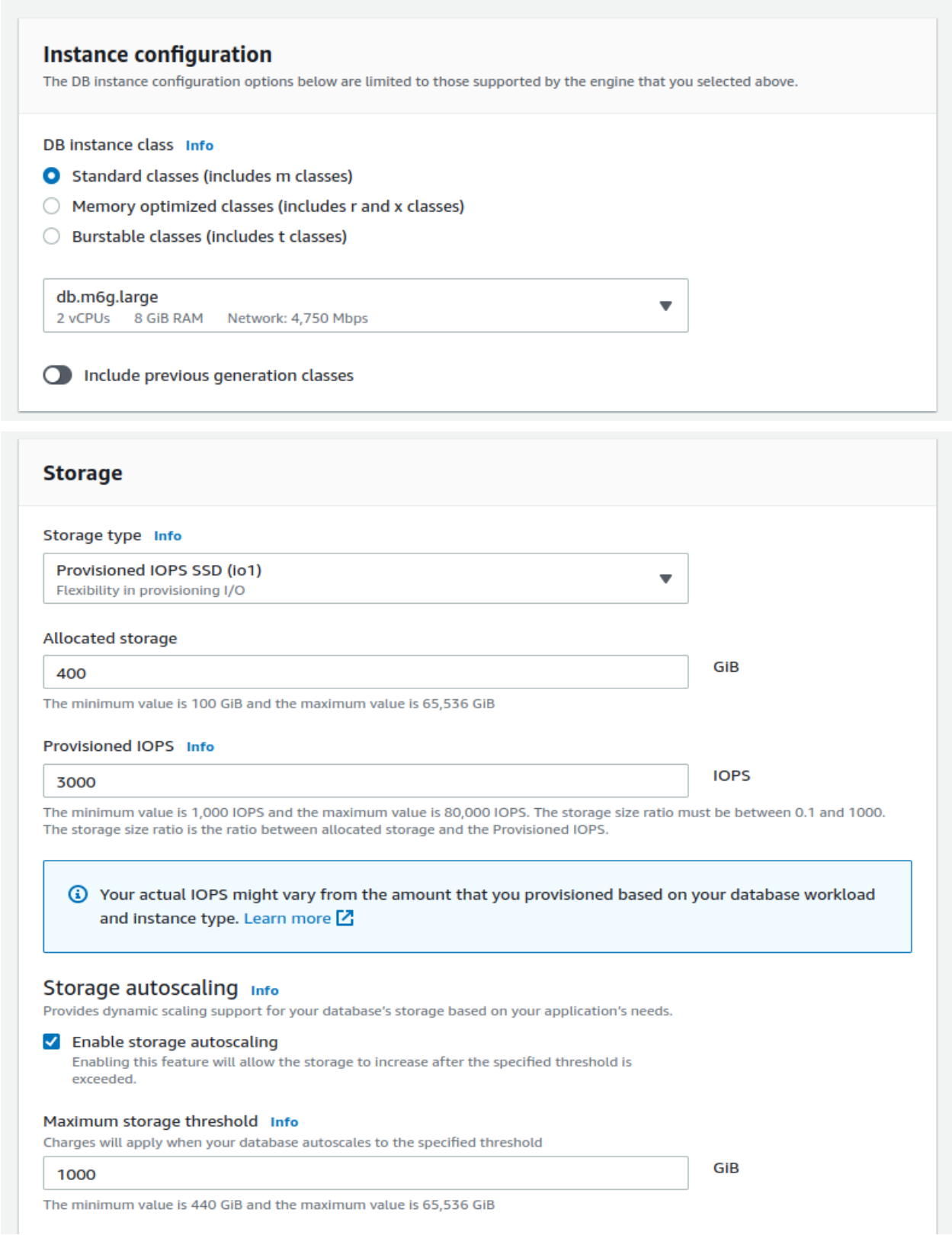
8.select multi A-Z instance.



9.Now you have to select a master username or password for login.



10.select instance configuration and storage according to your need.



Instance configuration

The DB instance configuration options below are limited to those supported by the engine that you selected above.

DB instance class Info

- (•) Standard classes (includes m classes)
- ( ) Memory optimized classes (includes r and x classes)
- ( ) Burstable classes (includes t classes)

db.m6g.large
2 vCPUs 8 GiB RAM Network: 4,750 Mbps

Include previous generation classes

Storage

Storage type Info

Provisioned IOPS SSD (io1)
Flexibility in provisioning I/O

Allocated storage

400 GiB

The minimum value is 100 GiB and the maximum value is 65,536 GiB

Provisioned IOPS Info

3000 IOPS

The minimum value is 1,000 IOPS and the maximum value is 80,000 IOPS. The storage size ratio must be between 0.1 and 1000. The storage size ratio is the ratio between allocated storage and the Provisioned IOPS.

Your actual IOPS might vary from the amount that you provisioned based on your database workload and instance type. Learn more

Storage autoscaling Info

Provides dynamic scaling support for your database's storage based on your application's needs.

[x] Enable storage autoscaling
Enabling this feature will allow the storage to increase after the specified threshold is exceeded.

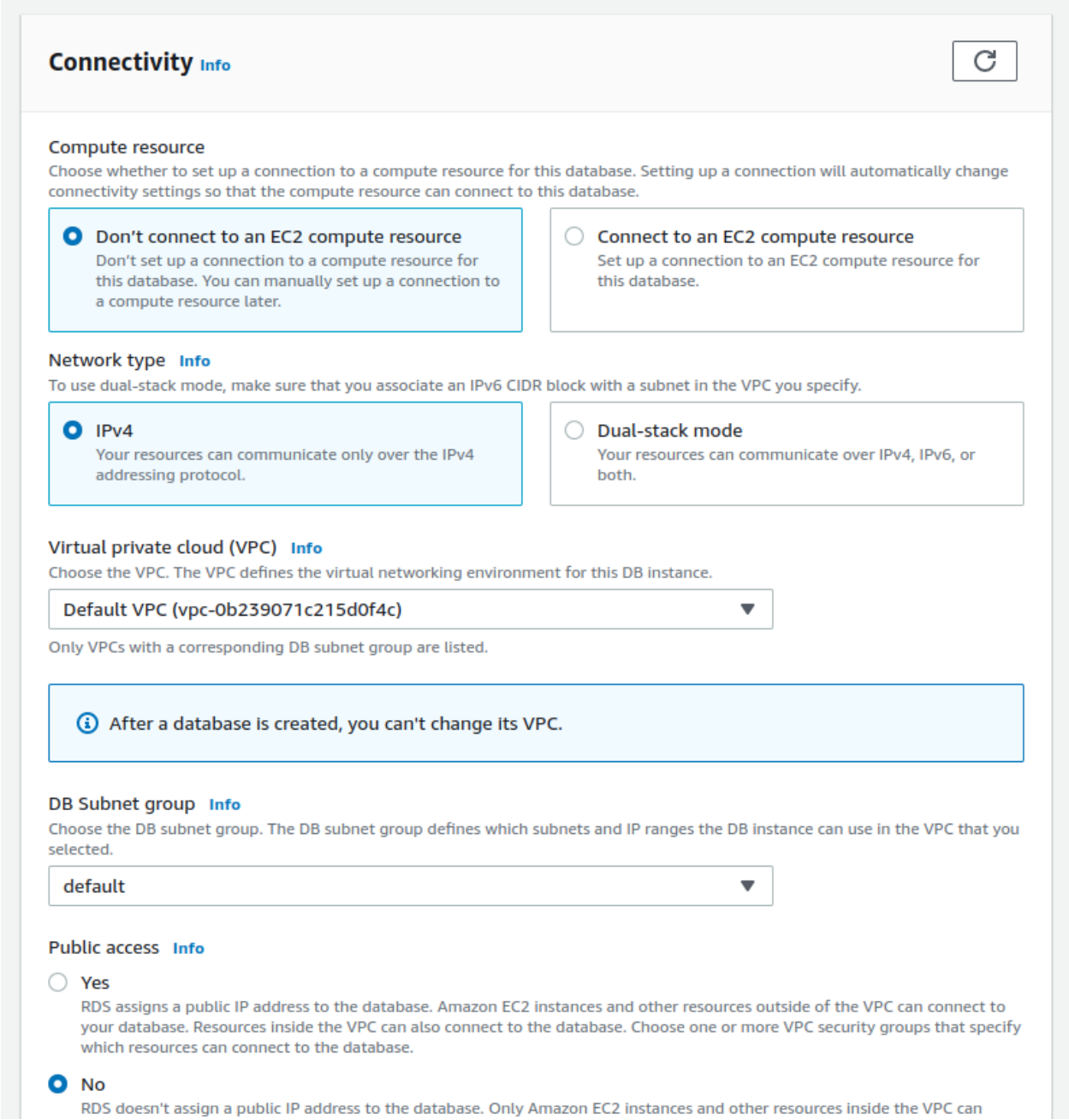
Maximum storage threshold Info

Charges will apply when your database autoscales to the specified threshold

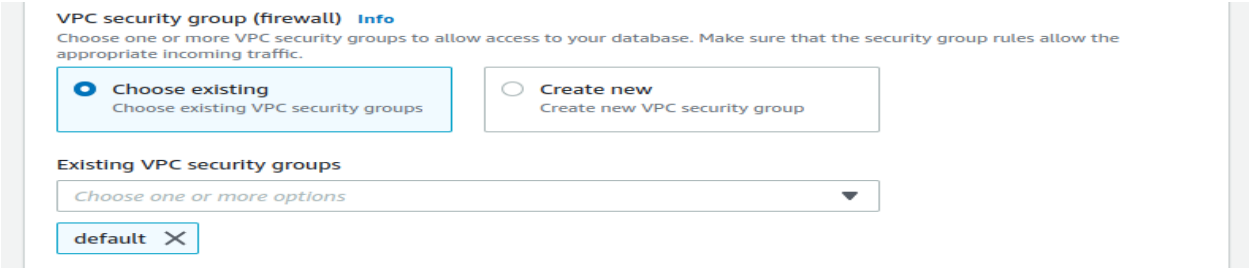
1000 GiB

The minimum value is 440 GiB and the maximum value is 65,536 GiB

11. .In the next connectivity section you have to select your vpc which is your eks vpc and subnets. Please no in public access.



12.Create a security group for your RDS.



13. Now click on the create RDS option.