

Creating an EKS cluster:

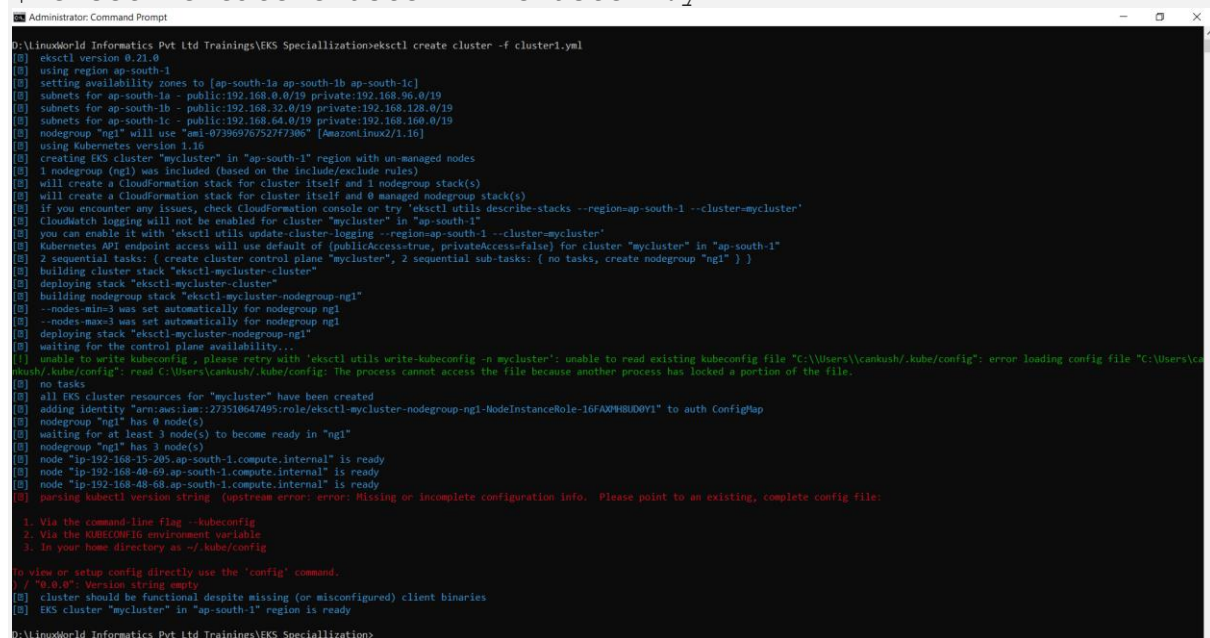
We will create a file named **cluster1.yml** which has all of the code required to create the EKS cluster. In this cluster, we will create three nodes(slaves) of the t2.micro instance. Adding ssh key so that later we can log in to these nodes for the management.

```
1  apiVersion: eksctl.io/v1alpha5
2  kind: ClusterConfig
3
4  metadata:
5    name: mycluster
6    region: ap-south-1
7
8  nodeGroups:
9    - name: ng1
10      desiredCapacity: 3
11      instanceType: t2.micro
12      ssh:
13        publicKeyName: cankush625
```

cluster1.yml

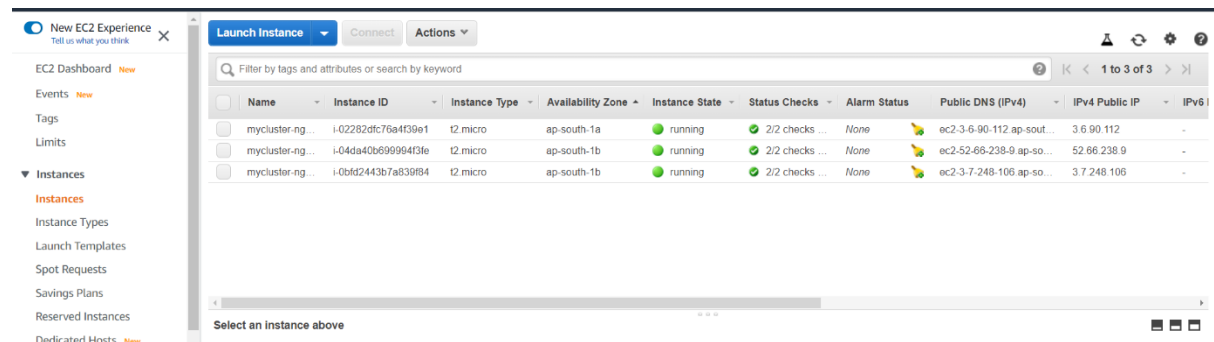
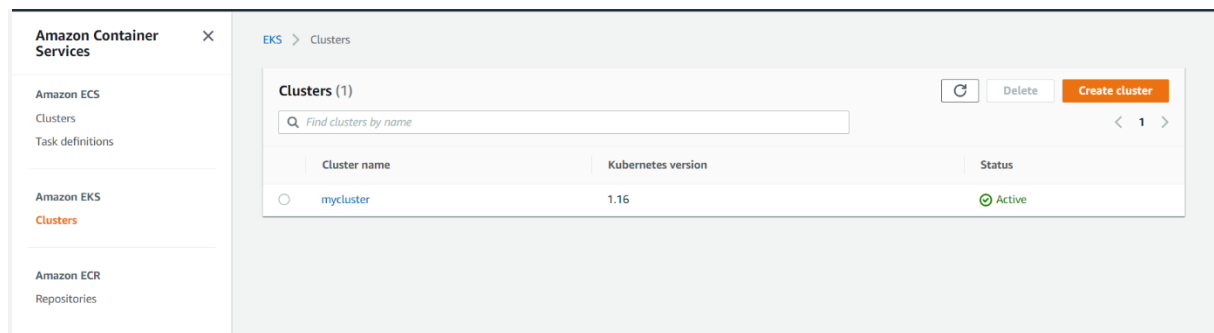
Run this command to run this file:

```
$ eksctl create cluster -f cluster1.yml
```



```
Administrator: Command Prompt
D:\LinuxWorld Informatics Pvt Ltd Trainings\EKS Specialization>eksctl create cluster -f cluster1.yml
[0] eksctl version 0.21.0
[0] using region ap-south-1
[0] setting availability zones to [ap-south-1a ap-south-1b ap-south-1c]
[0] subnets for ap-south-1a - public:192.168.0.0/19 private:192.168.96.0/19
[0] subnets for ap-south-1b - public:192.168.32.0/19 private:192.168.128.0/19
[0] subnets for ap-south-1c - public:192.168.64.0/19 private:192.168.160.0/19
[0] nodegroup "ng1" will use "ami-07396976/52747306" (AmazonLinux2/1.16)
[0] using Kubernetes version 1.16
[0] creating EKS cluster "mycluster" in "ap-south-1" region with un-managed nodes
[0] 1 nodegroup (ng1) was included (based on the include/exclude rules)
[0] will create a CloudFormation stack for cluster itself and 0 managed nodegroup stack(s)
[0] if you encounter any issues, check CloudFormation console or try 'eksctl utils describe-stacks --region=ap-south-1 --cluster=mycluster'
[0] CloudWatch logging will not be enabled for cluster "mycluster" in "ap-south-1"
[0] you can enable it with 'eksctl utils update-cluster-logging --region=ap-south-1 --cluster=mycluster'
[0] Kubernetes API endpoint access will use default of (publicAccess=true, privateAccess=false) for cluster "mycluster" in "ap-south-1"
[0] 2 sequential tasks: { create cluster control plane "mycluster", 2 sequential sub-tasks: { no tasks, create nodegroup "ng1" } }
[0] building cluster stack "eksctl-mycluster-cluster"
[0] deploying stack "eksctl-mycluster-cluster"
[0] building nodegroup stack "eksctl-mycluster-nodegroup-ng1"
[0] --nodes-min=3 was set automatically for nodegroup ng1
[0] --nodes-max=3 was set automatically for nodegroup ng1
[0] deploying stack "eksctl-mycluster-nodegroup-ng1"
[0] waiting for the control plane availability...
[0] unable to write kubeconfig, please retry with 'eksctl utils write-kubeconfig --n mycluster': unable to read existing kubeconfig file "C:\Users\cankush\kube/config": error loading config file "C:\Users\cankush\kube/config": read C:\Users\cankush\kube/config: The process cannot access the file because another process has locked a portion of the file.
[0] no tasks
[0] all EKS cluster resources for "mycluster" have been created
[0] adding identity "arn:aws:iam::273510647495:role/eksctl-mycluster-nodegroup-ng1-NodeInstanceRole-16FA0H8UD0Y1" to auth ConfigMap
[0] nodegroup "ng1" has 0 node(s)
[0] waiting for at least 3 node(s) to become ready in "ng1"
[0] nodegroup "ng1" has 3 node(s)
[0] node "ip-192-168-15-205.ap-south-1.compute.internal" is ready
[0] node "ip-192-168-40-69.ap-south-1.compute.internal" is ready
[0] node "ip-192-168-40-68.ap-south-1.compute.internal" is ready
[0] parsing kubectl version string (optional error: error: Missing or incomplete configuration info. Please point to an existing, complete config file:
1. Via the command-line flag --kubeconfig
2. Via the KUBECOW environment variable
3. In your home directory as ~/.kube/config
to view or setup config directly use the 'config' command.
/ "0.0.0": Version string empty
[0] cluster should be functional despite missing (or misconfigured) client binaries
[0] EKS cluster "mycluster" in "ap-south-1" region is ready
D:\LinuxWorld Informatics Pvt Ltd Trainings\EKS Specialization>
```

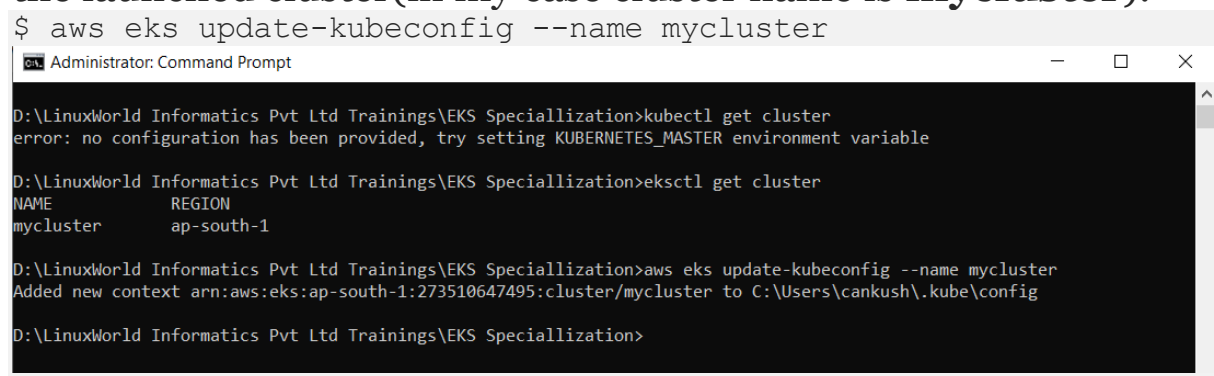
Create an EKS cluster



Now, the EKS cluster is created successfully.

Configuring kubectl command:

After that, we required to configure the **kubectl** command for the launched cluster(in my case cluster name is **mycluster**).



Configure kubectl command

Now, we required to install **Amazon EFS Utilities** on every node in the cluster using the following command:

```
$ yum install amazon-efs-utils
```

```
  _ | _ | )
  _ | ( _ | /
  _ | \ _ | _ |
Amazon Linux 2 AMI
```

<https://aws.amazon.com/amazon-linux-2/>

4 package(s) needed for security, out of 10 available

Run "sudo yum update" to apply all updates.

[root@ip-192-168-82-184 ~]# yum install amazon-efs-utils -y

Loaded plugins: priorities, update-motd

Resolving Dependencies

--> Running transaction check

--> Package amazon-efs-utils.noarch 0:1.26-2.amzn2 will be installed

--> Processing Dependency: stunnel >= 4.56 for package: amazon-efs-utils-1.26-2.amzn2.noarch

--> Running transaction check

--> Package stunnel.x86_64 0:4.56-6.amzn2.0.3 will be installed

--> Finished Dependency Resolution

Dependencies Resolved

| Package | Arch | Version | Repository | Size |
|------------------------------|--------|------------------|------------|-------|
| Installing: | | | | |
| amazon-efs-utils | noarch | 1.26-2.amzn2 | amzn2-core | 32 k |
| Installing for dependencies: | | | | |
| stunnel | x86_64 | 4.56-6.amzn2.0.3 | amzn2-core | 149 k |

Transaction Summary

Install 1 Package (+1 Dependent package)

Total download size: 181 k

Installed size: 416 k

Downloading packages:

| | | |
|---|--------|----------|
| (1/2): amazon-efs-utils-1.26-2.amzn2.noarch.rpm | 32 kB | 00:00:00 |
| (2/2): stunnel-4.56-6.amzn2.0.3.x86_64.rpm | 149 kB | 00:00:00 |

| Package | Arch | Version | Repository | Size |
|------------------------------|--------|------------------|------------|-------|
| Installing: | | | | |
| amazon-efs-utils | noarch | 1.26-2.amzn2 | amzn2-core | 32 k |
| Installing for dependencies: | | | | |
| stunnel | x86_64 | 4.56-6.amzn2.0.3 | amzn2-core | 149 k |

Transaction Summary

Install 1 Package (+1 Dependent package)

Total download size: 181 k

Installed size: 416 k

Downloading packages:

| | | |
|---|--------|----------|
| (1/2): amazon-efs-utils-1.26-2.amzn2.noarch.rpm | 32 kB | 00:00:00 |
| (2/2): stunnel-4.56-6.amzn2.0.3.x86_64.rpm | 149 kB | 00:00:00 |

Total 1.2 MB/s | 181 kB 00:00:00

Running transaction check

Running transaction test

Transaction test succeeded

Running transaction

| | |
|---|-----|
| Installing : stunnel-4.56-6.amzn2.0.3.x86_64 | 1/2 |
| Installing : amazon-efs-utils-1.26-2.amzn2.noarch | 2/2 |
| Verifying : stunnel-4.56-6.amzn2.0.3.x86_64 | 1/2 |
| Verifying : amazon-efs-utils-1.26-2.amzn2.noarch | 2/2 |

Installed:

amazon-efs-utils.noarch 0:1.26-2.amzn2

Dependency Installed:

stunnel.x86_64 0:4.56-6.amzn2.0.3

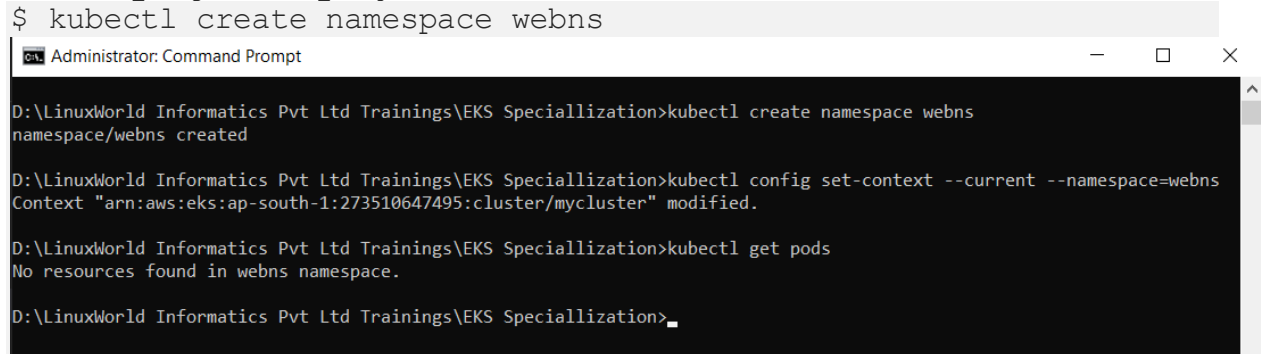
Complete!

[root@ip-192-168-82-184 ~]#

Create a namespace for our project:

Let's create a namespace named **webns** where we will create all of our project deployments.

```
$ kubectl create namespace webns
```



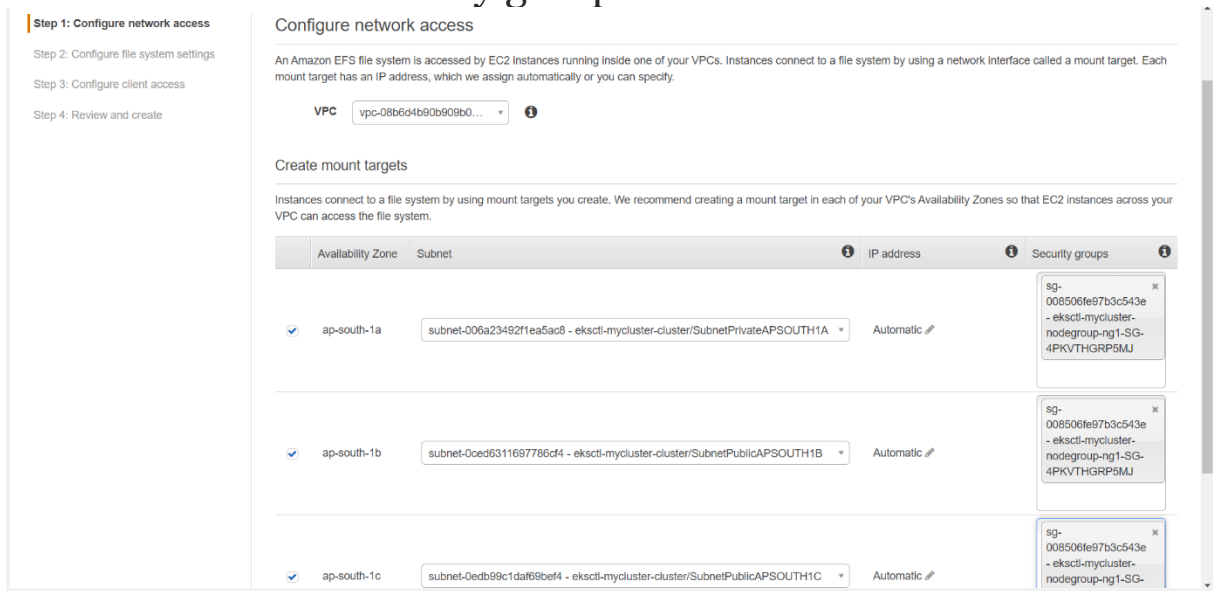
The screenshot shows a Windows Command Prompt window titled "Administrator: Command Prompt". The command prompt is running in the directory "D:\LinuxWorld Informatics Pvt Ltd Trainings\EKS Speciallization". The user has entered the command "kubectl create namespace webns", which has been executed successfully, resulting in the message "namespace/webns created". The user then enters "kubectl config set-context --current --namespace=webns", which is also executed successfully, resulting in the message "Context \"arn:aws:eks:ap-south-1:273510647495:cluster/mycluster\" modified.". Finally, the user enters "kubectl get pods", which returns the message "No resources found in webns namespace.". The prompt ends with "D:\LinuxWorld Informatics Pvt Ltd Trainings\EKS Speciallization>".

Create namespace

Create an EFS storage:

We have to store the data and configurations of our project permanently. For this purpose, AWS provides us one scalable and elastic storage known as EFS.

Create an EFS storage from the AWS console and make sure to attach the correct security group as that of the EKS cluster.



The screenshot shows the "Configure network access" page in the AWS Management Console. The page is divided into two main sections: "Configure network access" and "Create mount targets".

Configure network access:

- Step 1: Configure network access** (selected)
- Step 2: Configure file system settings
- Step 3: Configure client access
- Step 4: Review and create

Configure network access:

- An Amazon EFS file system is accessed by EC2 instances running inside one of your VPCs. Instances connect to a file system by using a network interface called a mount target. Each mount target has an IP address, which we assign automatically or you can specify.
- VPC:** vpc-08b6d4b90b909b0...

Create mount targets:

Instances connect to a file system by using mount targets you create. We recommend creating a mount target in each of your VPC's Availability Zones so that EC2 instances across your VPC can access the file system.

| Availability Zone | Subnet | IP address | Security groups |
|---|--|------------|---|
| <input checked="" type="checkbox"/> ap-south-1a | subnet-006a23492f1ea5ac8 - eksctl-mycluster-cluster/SubnetPrivateAPSOUTH1A | Automatic | sg-008506fe97b3c543e - eksctl-mycluster-nodegroup-ng1-SG-4PKVTHGRP5MJ |
| <input checked="" type="checkbox"/> ap-south-1b | subnet-0ced6311697786cf4 - eksctl-mycluster-cluster/SubnetPublicAPSOUTH1B | Automatic | sg-008506fe97b3c543e - eksctl-mycluster-nodegroup-ng1-SG-4PKVTHGRP5MJ |
| <input checked="" type="checkbox"/> ap-south-1c | subnet-0edb99c1da168b6f4 - eksctl-mycluster-cluster/SubnetPublicAPSOUTH1C | Automatic | sg-008506fe97b3c543e - eksctl-mycluster-nodegroup-ng1-SG- |

Create EFS storage

Create file system

Step 1: Configure network access
Step 2: Configure file system settings
Step 3: Configure client access
Step 4: Review and create

Review and create

Review the configuration below before proceeding to create your file system.

File system access

| VPC | Availability Zone | Subnet | IP address | Security groups |
|--|-------------------|--|------------|---|
| vpc-08b6d4b90b909b020 - eksctl-mycluster-cluster/VPC | ap-south-1a | subnet-006a23492f1ea5ac8 - eksctl-mycluster-cluster/SubnetPrivateAPSOUTH1A | Automatic | sg-006506fe97b3c543e - eksctl-mycluster-nodegroup-ng1-SG-4PKVTHGRP5MJ |
| | ap-south-1b | subnet-0ced6311697786cf4 - eksctl-mycluster-cluster/SubnetPublicAPSOUTH1B | Automatic | sg-006506fe97b3c543e - eksctl-mycluster-nodegroup-ng1-SG-4PKVTHGRP5MJ |
| | ap-south-1c | subnet-0edb99c1daf69bef4 - eksctl-mycluster-cluster/SubnetPublicAPSOUTH1C | Automatic | sg-006506fe97b3c543e - eksctl-mycluster-nodegroup-ng1-SG-4PKVTHGRP5MJ |

Optional settings

Tags

Name: joomla-efs

Performance mode
General Purpose

Throughput mode
Bursting

Encrypted
No

Create EFS storage

File systems

Create file system

Actions

| | Name | File system ID | Metered size | Number of mount targets | Creation date |
|-------------------------|------------|----------------|--------------|-------------------------|--------------------------|
| <div></div> <div></div> | joomla-efs | fs-8be16b5a | 6.0 KiB | 3 | 07/09/2020, 16:14:35 UTC |

Other details

Owner ID

273510647495

File system state

Available

Performance mode

General Purpose

Throughput mode

Bursting

Encrypted

No

Lifecycle policy

None

Tags

Name: joomla-efs

File system access

Manage network access

Manage client access

DNS name

fs-8be16b5a.efs.ap-south-1.amazonaws.com

EFS storage is created

EFS storage is created!

Create an EFS provisioner:

After that, we have to create **EFS provisioner** so that the EFS storage can be used by the resources in the EKS cluster.

The code for creating an EFS provisioner is written in a **create-efs-provisioner.yaml** file. Just replace the EFS file system ID at line no. 22 and replace the nfs server at line no. 33 with the values of your EFS file system.

```

! create-efs-provisioner.yaml
1  kind: Deployment
2  apiVersion: apps/v1
3  metadata:
4    name: efs-provisioner
5  spec:
6    selector:
7      matchLabels:
8        app: efs-provisioner
9    replicas: 1
10   strategy:
11     type: Recreate
12   template:
13     metadata:
14       labels:
15         app: efs-provisioner
16     spec:
17       containers:
18         - name: efs-provisioner
19           image: quay.io/external_storage/efs-provisioner:v0.1.0
20           env:
21             - name: FILE_SYSTEM_ID
22               value: fs-8be16b5a
23             - name: AWS_REGION
24               value: ap-south-1
25             - name: PROVISIONER_NAME
26               value: aws-efs
27       volumeMounts:
28         - name: pv-volume
29           mountPath: /persistentvolumes
30       volumes:
31         - name: pv-volume
32           nfs:
33             server: fs-8be16b5a.efs.ap-south-1.amazonaws.com
34             path: /
35

```

create-efs-provisioner.yaml

Run the **create-efs-provisioner.yaml** file in a webns namespace to create an EFS provisioner.

```
$ kubectl create -f create-efs-provisioner.yaml -n webns
```

Create RBAC:

Now, we required to create **Role-Based access control(RBAC)**. We are giving the role as cluster admin.

```
! create-rbac.yaml
1  ---
2  apiVersion: rbac.authorization.k8s.io/v1beta1
3  kind: ClusterRoleBinding
4  metadata:
5    name: nfs-provisioner-role-binding
6  subjects:
7    - kind: ServiceAccount
8      name: default
9      namespace: webns
10 roleRef:
11   kind: ClusterRole
12   name: cluster-admin
13   apiGroup: rbac.authorization.k8s.io
14
```

create-rbac.yaml

This can be done by running the **create-rbac.yaml** file in webns namespace.

```
$ kubectl create -f create-rbac.yaml -n webns
```

Create Storage Class, PVC for Joomla and PVC for MySQL:

We are creating the storage class and taking the storage from EFS storage we had created earlier.

```

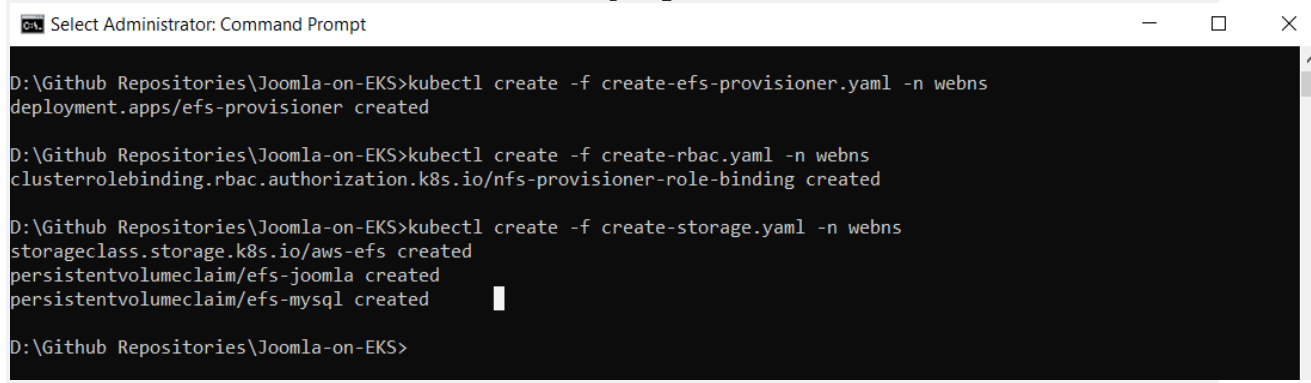
! create-storage.yaml
1  kind: StorageClass
2  apiVersion: storage.k8s.io/v1
3  metadata:
4  |   name: aws-efs
5  provisioner: aws-efs
6  ---
7  kind: PersistentVolumeClaim
8  apiVersion: v1
9  metadata:
10 |   name: efs-joomla
11 |   annotations:
12 |     volume.beta.kubernetes.io/storage-class: "aws-efs"
13 spec:
14 |   accessModes:
15 |     - ReadWriteMany
16 |   resources:
17 |     requests:
18 |       storage: 10Gi
19 ---
20 kind: PersistentVolumeClaim
21 apiVersion: v1
22 metadata:
23 |   name: efs-mysql
24 |   annotations:
25 |     volume.beta.kubernetes.io/storage-class: "aws-efs"
26 spec:
27 |   accessModes:
28 |     - ReadWriteMany
29 |   resources:
30 |     requests:
31 |       storage: 10Gi
32

```

create-storage.yaml

Run the **create-storage.yaml** file to create the Storage Class, PVC for Joomla, and PVC for MySQL.

```
$ kubectl create -f create-storage.yaml -n webns
```



```

Select Administrator: Command Prompt

D:\Github Repositories\Joomla-on-EKS>kubectl create -f create-efs-provisioner.yaml -n webns
deployment.apps/efs-provisioner created

D:\Github Repositories\Joomla-on-EKS>kubectl create -f create-rbac.yaml -n webns
clusterrolebinding.rbac.authorization.k8s.io/nfs-provisioner-role-binding created

D:\Github Repositories\Joomla-on-EKS>kubectl create -f create-storage.yaml -n webns
storageclass.storage.k8s.io/aws-efs created
persistentvolumeclaim/efs-joomla created
persistentvolumeclaim/efs-mysql created

D:\Github Repositories\Joomla-on-EKS>

```

Create EFS provisioner, RBAC, and storage

Now, all of the resources required to run our Content Management System(Joomla) are created. As a final step, we required to create the **MySQL** database and launch the **Joomla**.

Create a **deploy-mysql.yaml** file that contains all of the configurations to deploy the MySQL database.

```
! deploy-mysql.yaml
1  apiVersion: v1
2  kind: Service
3  metadata:
4    name: joomldb
5    labels:
6      app: joomla
7  spec:
8    ports:
9      - port: 3306
10   selector:
11     app: joomla
12     tier: mysql
13   clusterIP: None
14   ---
15   apiVersion: apps/v1
16   kind: Deployment
17   metadata:
18     name: joomldb
19     labels:
20       app: joomla
21   spec:
22     selector:
23       matchLabels:
24         app: joomla
25         tier: mysql
26     strategy:
27       type: Recreate
28     template:
29       metadata:
30         labels:
31           app: joomla
32           tier: mysql
33       spec:
34         containers:
35           - image: mysql:5.6
36             name: mysql
37             env:
38               - name: MYSQL_ROOT_PASSWORD
```

```

! deploy-mysql.yaml
22   selector:
23     matchLabels:
24       app: joomla
25       tier: mysql
26   strategy:
27     type: Recreate
28   template:
29     metadata:
30       labels:
31         app: joomla
32         tier: mysql
33   spec:
34     containers:
35     - image: mysql:5.6
36       name: mysql
37       env:
38       - name: MYSQL_ROOT_PASSWORD
39         valueFrom:
40           secretKeyRef:
41             name: mydbsecret
42             key: rootpass
43       ports:
44       - containerPort: 3306
45         name: mysql
46       volumeMounts:
47       - name: mysql-persistent-storage
48         mountPath: /var/lib/mysql
49     volumes:
50     - name: mysql-persistent-storage
51       persistentVolumeClaim:
52         claimName: efs-mysql
53

```

Create a **deploy-joomla.yaml** file that contains all of the configurations to deploy the Joomla site.

```
! deploy-joomla.yaml
1  apiVersion: v1
2  kind: Service
3  metadata:
4    name: joomla
5    labels:
6      app: joomla
7  spec:
8    ports:
9      - port: 80
10   selector:
11     app: joomla
12     tier: frontend
13   type: LoadBalancer
14   ---
15   apiVersion: apps/v1
16   kind: Deployment
17   metadata:
18     name: joomla
19     labels:
20       app: joomla
21   spec:
22     selector:
23       matchLabels:
24         app: joomla
25         tier: frontend
26     strategy:
27       type: Recreate
28     template:
29       metadata:
30         labels:
31           app: joomla
32           tier: frontend
33       spec:
34         containers:
35           - image: joomla:3.4.8-apache
36             name: joomla
37             env:
38               - name: JOOMLA_DB_HOST
```

```

! deploy-joomla.yaml
24   app: joomla
25   tier: frontend
26   strategy:
27     type: Recreate
28   template:
29     metadata:
30       labels:
31         app: joomla
32         tier: frontend
33     spec:
34       containers:
35       - image: joomla:3.4.8-apache
36         name: joomla
37         env:
38         - name: JOOMLA_DB_HOST
39           value: joomladb
40         - name: JOOMLA_DB_PASSWORD
41           valueFrom:
42             secretKeyRef:
43               name: mydbsecret
44               key: rootpass
45         ports:
46         - containerPort: 80
47           name: joomla
48         volumeMounts:
49         - name: joomla-persistent-storage
50           mountPath: /var/www/html
51       volumes:
52       - name: joomla-persistent-storage
53         persistentVolumeClaim:
54           claimName: efs-joomla
55

```

Finally, we will create **kustomization.yaml** file that contains the sequence to launch the MySQL and Joomla configuration files. Also, this file will create secrets for us.

```

! kustomization.yaml
1  apiVersion: kustomize.config.k8s.io/v1beta1
2  kind: Kustomization
3  secretGenerator:
4  - name: mydbsecret
5    literals:
6    - rootpass=ankushroot
7  resources:
8  - deploy-mysql.yaml
9  - deploy-joomla.yaml

```

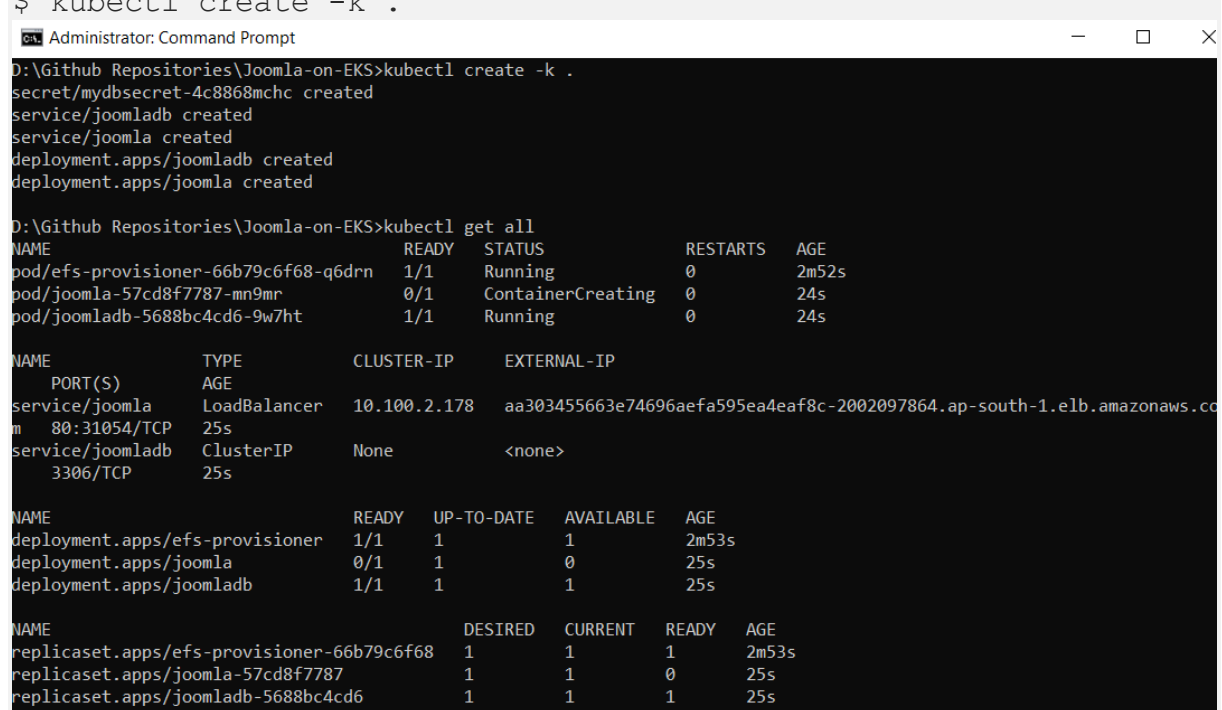
kustomization.yaml

Run this file to deploy the MySQL and Joomla site on the EKS cluster.

```

$ kubectl create -k .

```



```

D:\Github Repositories\Joomla-on-EKS>kubectl create -k .
secret/mydbsecret-4c8868mhc created
service/joomlab created
service/joomla created
deployment.apps/joomlab created
deployment.apps/joomla created

D:\Github Repositories\Joomla-on-EKS>kubectl get all
NAME                                READY   STATUS              RESTARTS   AGE
pod/efs-provisioner-66b79c6f68-q6drn 1/1     Running             0           2m52s
pod/joomla-57cd8f7787-mn9mr          0/1     ContainerCreating   0           24s
pod/joomlab-5688bc4cd6-9w7ht         1/1     Running             0           24s

NAME                                PORT(S)    TYPE          CLUSTER-IP    EXTERNAL-IP
service/joomla                      80:31054/TCP LoadBalancer  10.100.2.178   aa303455663e74696aefa595ea4eaf8c-2002097864.ap-south-1.elb.amazonaws.com
service/joomlab                     3306/TCP   ClusterIP     None           <none>

NAME                                READY   UP-TO-DATE   AVAILABLE   AGE
deployment.apps/efs-provisioner      1/1     1             1           2m53s
deployment.apps/joomla              0/1     1             0           25s
deployment.apps/joomlab             1/1     1             1           25s

NAME                                DESIRED   CURRENT   READY   AGE
replicaset.apps/efs-provisioner-66b79c6f68 1         1         1       2m53s
replicaset.apps/joomla-57cd8f7787          1         1         0       25s
replicaset.apps/joomlab-5688bc4cd6         1         1         1       25s

```

kubectl create -k .

Check if all of the resources get launched.

```
$ kubectl get all
```

```
Administrator: Command Prompt
D:\Github Repositories\Joomla-on-EKS>kubectl get all
NAME                                READY   STATUS    RESTARTS   AGE
pod/efs-provisioner-66b79c6f68-q6drn 1/1     Running   0           4m7s
pod/joomla-57cd8f7787-mn9mr          1/1     Running   0           99s
pod/joomldb-5688bc4cd6-9w7ht         1/1     Running   0           99s

NAME                                TYPE                CLUSTER-IP      EXTERNAL-IP
service/joomla                      LoadBalancer        10.100.2.178     aa303455663e74696aefa595ea4eaf8c-2002097864.ap-south-1.elb.amazonaws.com
service/joomldb                      ClusterIP            None             <none>

NAME                                READY   UP-TO-DATE   AVAILABLE   AGE
deployment.apps/efs-provisioner      1/1     1             1           4m8s
deployment.apps/joomla               1/1     1             1           100s
deployment.apps/joomldb              1/1     1             1           100s

NAME                                DESIRED   CURRENT   READY   AGE
replicaset.apps/efs-provisioner-66b79c6f68 1         1         1       4m8s
replicaset.apps/joomla-57cd8f7787          1         1         1       100s
replicaset.apps/joomldb-5688bc4cd6         1         1         1       100s

D:\Github Repositories\Joomla-on-EKS>
```

kubectl get all

As we can see in the above picture, all of our resources are launched and all of the pods are up and running.


Access the launched site:

To access the launched site open the EXTERNAL-IP provided by the service/joomla in the browser to view the deployed joomla site.



Joomla site is launched

Do the installation.



Joomla!® is free software released under the GNU General Public License.

1 Configuration2 Database3 Overview

Finalisation

PreviousInstall

Install Sample Data

☒ None (Required for basic native multilingual site creation)

☐ Blog English (GB) Sample Data

☐ Brochure English (GB) Sample Data

☐ Default English (GB) Sample Data

☐ Learn Joomla English (GB) Sample Data

☐ Test English (GB) Sample Data

Installing sample data is strongly recommended for beginners.
This will install sample content that is included in the Joomla! Installation package.

Overview

Email Configuration

YesNo

Send configuration settings to **ankush@gmail.com** by email after installation.

Main Configuration


Database Configuration

Site Name

ankush

Database Type

mysql



Joomla!® is free software released under the GNU General Public License.

Installing ...

Backing up old database tables

Creating database tables

Creating configuration File

Main Configuration

Database Configuration

Site Name

ankush

Site Offline

No

Administrator Email

ankush@gmail.com

Administrator Username

ankush

Administrator Password

Database Type

mysql

Host Name

joomlaadb

Username

root

Password

Database Name

joomlaweb

Table Prefix

josnm_

Old Database Process

Backup

Pre-Installation Check

Recommended settings:

If any of these items are not supported (marked as **No**) then please take actions to correct them.
You can't install Joomla! until your setup meets the requirements below.

PHP Version >= 5.3.10

Yes

Magic Quotes GPC Off

Yes

Register Globals Off

Yes

Zip Compression Support

Yes

XML Support

Yes

Database Support:
(mysqli, pdo, mysql)

Yes

INI Parser Support

Yes

JSON Support

Yes

configuration.php Writable

Yes

These settings are recommended for PHP in order to ensure full compatibility with Joomla.
However, Joomla! will still operate if your settings do not quite match the recommended configuration.

Directive

Recommended

Actual

Safe Mode

Off

Off

Display Errors

Off

On

File Uploads

On

On

Magic Quotes Runtime

Off

On

Output Buffering

Off

Off

Session Auto Start

Off

Off

Native ZIP support

On

Off



And finally, our Joomla Content Management System is successfully launched using the EKS cluster.

