Creating an EKS cluster:

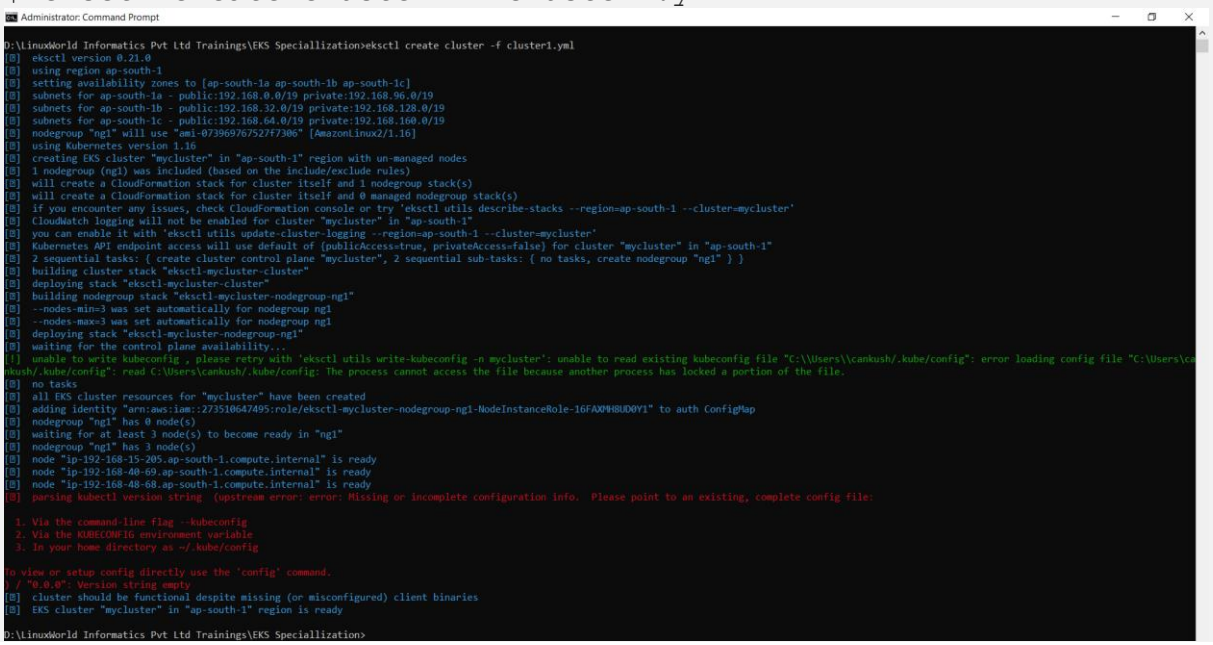
We will create a file named **cluster1.yml** which has all of the code required to create the EKS cluster. In this cluster, we will create three nodes(slaves) of the t2.micro instance. Adding ssh key so that later we can log in to these nodes for the management.

```yaml
apiVersion: eksctl.io/v1alpha5
kind: ClusterConfig

metadata:
  name: mycluster
  region: ap-south-1

nodeGroups:
  - name: ng1
    desiredCapacity: 3
    instanceType: t2.micro
    ssh:
      publicKeyName: cankush625
```

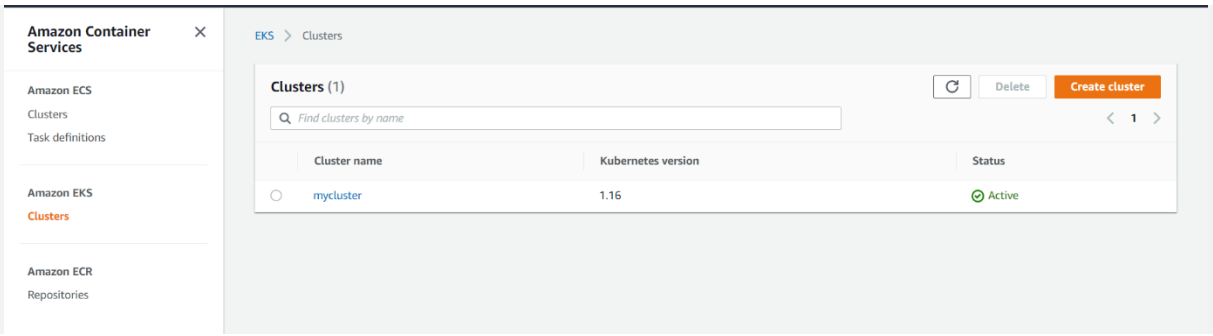
cluster1.yml

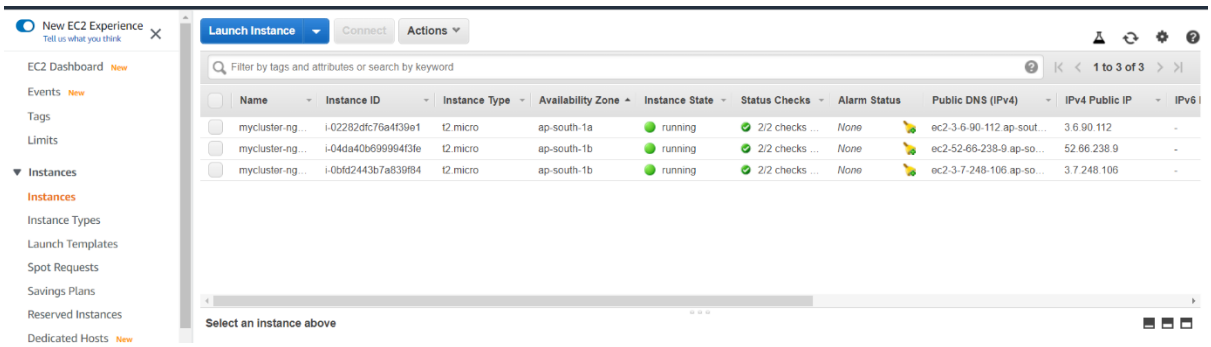
Run this command to run this file:

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



Create an EKS cluster



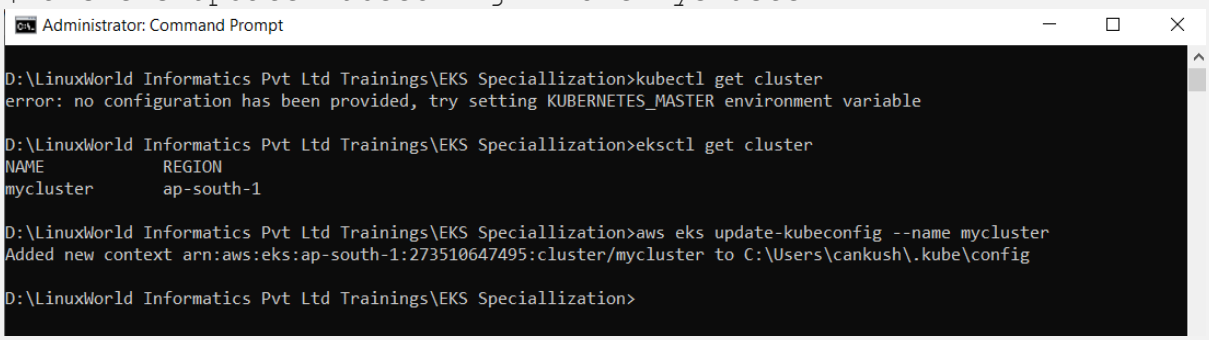


Now, the EKS cluster is created successfully.

Configuring kubectl command:

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

```
$ aws eks update-kubeconfig --name 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
```

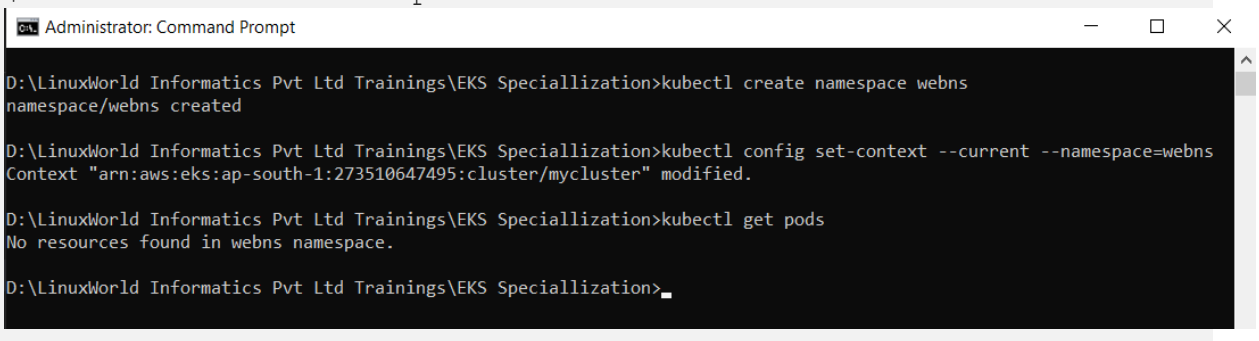

```
=================================================================================================
 Package                    Arch            Version                    Repository           Size
=================================================================================================
Installing:
 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
```

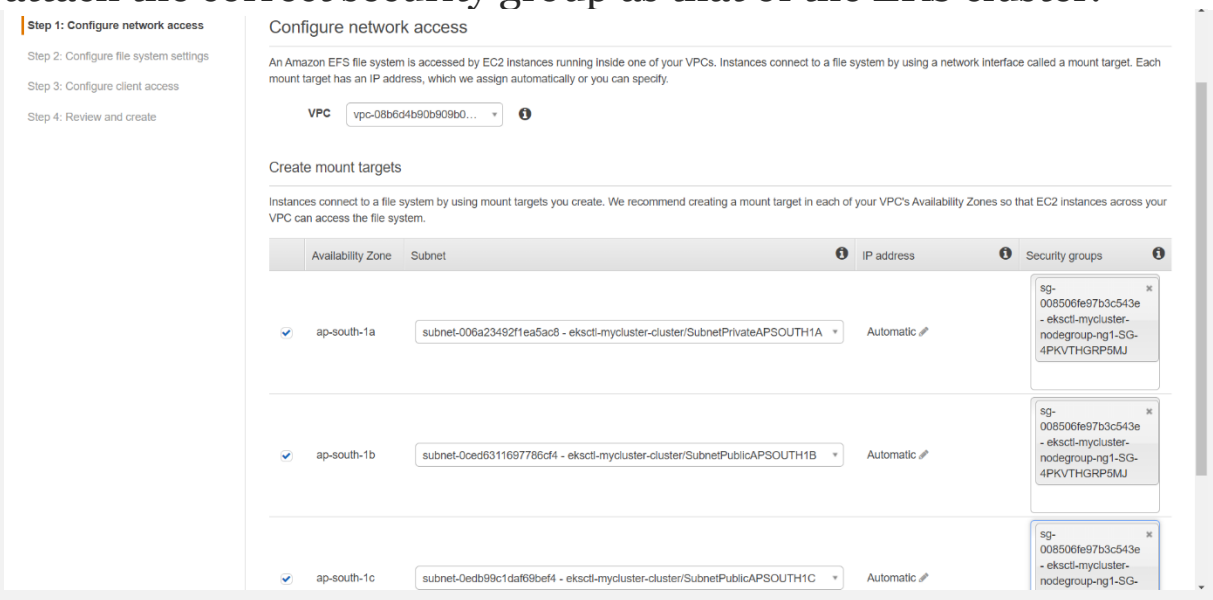


Create namespace

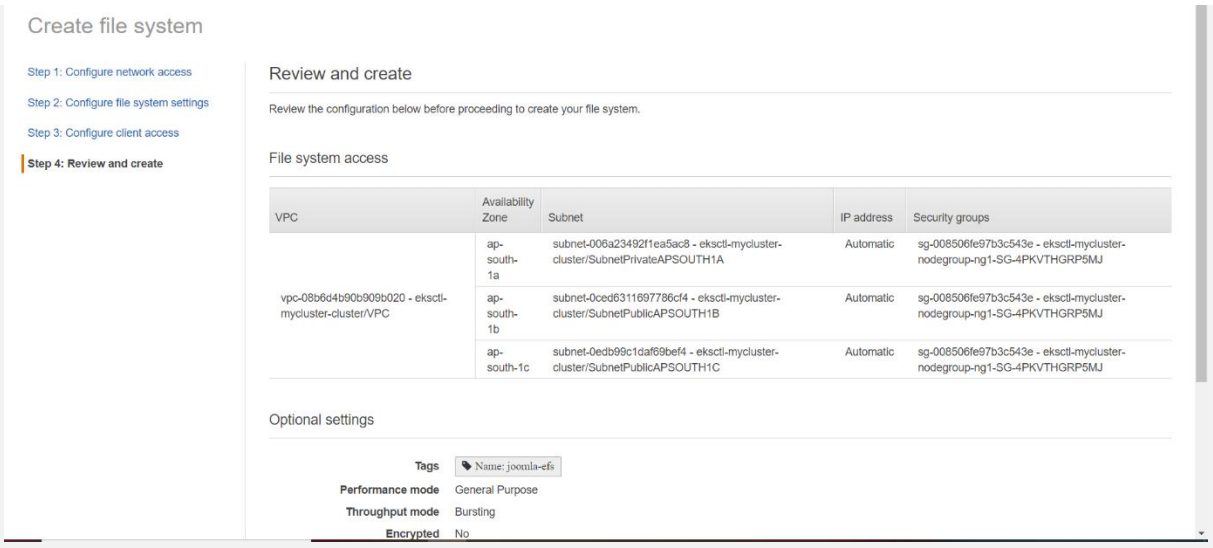
Create an EFS storage:

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.

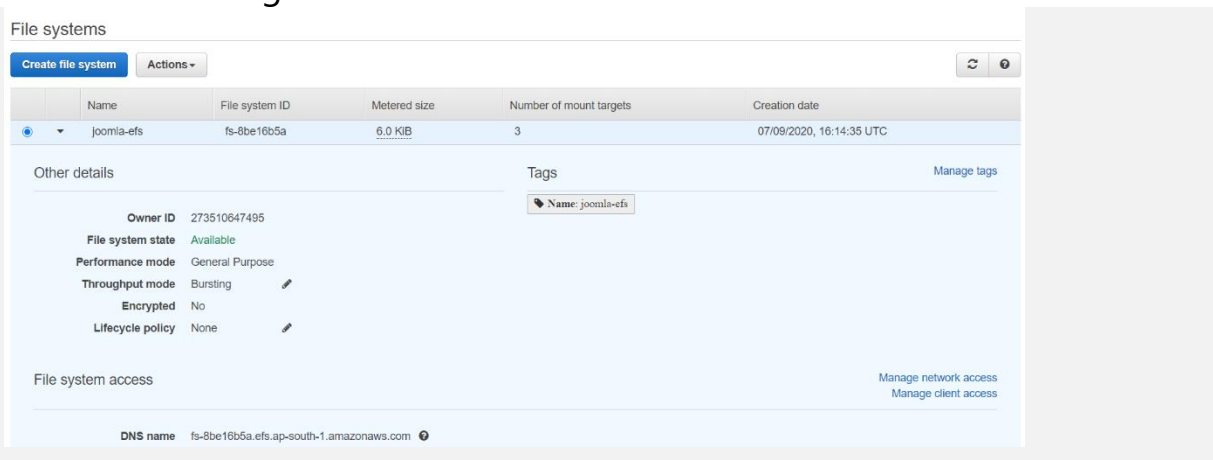
Create an EFS storage from the AWS console and make sure to attach the correct security group as that of the EKS cluster.



Create EFS storage



Create EFS storage



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.

```yaml
! create-efs-provisioner.yaml
kind: Deployment
apiVersion: apps/v1
metadata:
  name: efs-provisioner
spec:
  selector:
    matchLabels:
      app: efs-provisioner
  replicas: 1
  strategy:
    type: Recreate
  template:
    metadata:
      labels:
        app: efs-provisioner
    spec:
      containers:
        - name: efs-provisioner
          image: quay.io/external_storage/efs-provisioner:v0.1.0
          env:
            - name: FILE_SYSTEM_ID
              value: fs-8be16b5a
            - name: AWS_REGION
              value: ap-south-1
            - name: PROVISIONER_NAME
              value: aws-efs
          volumeMounts:
            - name: pv-volume
              mountPath: /persistentvolumes
      volumes:
        - name: pv-volume
          nfs:
            server: fs-8be16b5a.efs.ap-south-1.amazonaws.com
            path: /
```

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.

```yaml
---
apiVersion: rbac.authorization.k8s.io/v1beta1
kind: ClusterRoleBinding
metadata:
  name: nfs-provisioner-role-binding
subjects:
  - kind: ServiceAccount
    name: default
    namespace: webns
roleRef:
  kind: ClusterRole
  name: cluster-admin
  apiGroup: rbac.authorization.k8s.io
```

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.

```yaml
create-storage.yaml
kind: StorageClass
apiVersion: storage.k8s.io/v1
metadata:
  name: aws-efs
provisioner: aws-efs
---
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
  name: efs-joomla
  annotations:
    volume.beta.kubernetes.io/storage-class: "aws-efs"
spec:
  accessModes:
    - ReadWriteMany
  resources:
    requests:
      storage: 10Gi
---
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
  name: efs-mysql
  annotations:
    volume.beta.kubernetes.io/storage-class: "aws-efs"
spec:
  accessModes:
    - ReadWriteMany
  resources:
    requests:
      storage: 10Gi
```

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

```yaml
apiVersion: v1
kind: Service
metadata:
  name: joomladb
  labels:
    app: joomla
spec:
  ports:
    - port: 3306
  selector:
    app: joomla
    tier: mysql
  clusterIP: None
---
apiVersion: apps/v1
kind: Deployment
metadata:
  name: joomladb
  labels:
    app: joomla
spec:
  selector:
    matchLabels:
      app: joomla
      tier: mysql
  strategy:
    type: Recreate
  template:
    metadata:
      labels:
        app: joomla
        tier: mysql
    spec:
      containers:
      - image: mysql:5.6
        name: mysql
        env:
        - name: MYSQL_ROOT_PASSWORD
```

deploy-mysql.yaml

```yaml
  selector:
    matchLabels:
      app: joomla
      tier: mysql
  strategy:
    type: Recreate
  template:
    metadata:
      labels:
        app: joomla
        tier: mysql
    spec:
      containers:
      - image: mysql:5.6
        name: mysql
        env:
        - name: MYSQL_ROOT_PASSWORD
          valueFrom:
            secretKeyRef:
              name: mydbsecret
              key: rootpass
        ports:
        - containerPort: 3306
          name: mysql
        volumeMounts:
        - name: mysql-persistent-storage
          mountPath: /var/lib/mysql
      volumes:
      - name: mysql-persistent-storage
        persistentVolumeClaim:
          claimName: efs-mysql
```

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

deploy-joomla.yaml

```yaml
apiVersion: v1
kind: Service
metadata:
  name: joomla
  labels:
    app: joomla
spec:
  ports:
    - port: 80
  selector:
    app: joomla
    tier: frontend
  type: LoadBalancer
---
apiVersion: apps/v1
kind: Deployment
metadata:
  name: joomla
  labels:
    app: joomla
spec:
  selector:
    matchLabels:
      app: joomla
      tier: frontend
  strategy:
    type: Recreate
  template:
    metadata:
      labels:
        app: joomla
        tier: frontend
    spec:
      containers:
      - image: joomla:3.4.8-apache
        name: joomla
        env:
        - name: JOOMLA_DB_HOST
```

deploy-joomla.yaml

```yaml
        app: joomla
        tier: frontend
    strategy:
      type: Recreate
    template:
      metadata:
        labels:
          app: joomla
          tier: frontend
      spec:
        containers:
        - image: joomla:3.4.8-apache
          name: joomla
          env:
          - name: JOOMLA_DB_HOST
            value: joomladb
          - name: JOOMLA_DB_PASSWORD
            valueFrom:
              secretKeyRef:
                name: mydbsecret
                key: rootpass
          ports:
          - containerPort: 80
            name: joomla
          volumeMounts:
          - name: joomla-persistent-storage
            mountPath: /var/www/html
        volumes:
        - name: joomla-persistent-storage
          persistentVolumeClaim:
            claimName: efs-joomla
```

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.

```yaml
kustomization.yaml
apiVersion: kustomize.config.k8s.io/v1beta1
kind: Kustomization
secretGenerator:
- name: mydbsecret
  literals:
  - rootpass=ankushroot
resources:
  - deploy-mysql.yaml
  - deploy-joomla.yaml
```

kustomization.yaml

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

```
$ kubectl create -k .
```

```
Administrator: Command Prompt

D:\Github Repositories\Joomla-on-EKS>kubectl create -k .
secret/mydbsecret-4c8868mchc created
service/joomladb created
service/joomla created
deployment.apps/joomladb 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/joomladb-5688bc4cd6-9w7ht           1/1     Running             0          24s

NAME                TYPE           CLUSTER-IP     EXTERNAL-IP
    PORT(S)         AGE
service/joomla      LoadBalancer   10.100.2.178   aa303455663e74696aefa595ea4eaf8c-2002097864.ap-south-1.elb.amazonaws.co
m   80:31054/TCP    25s
service/joomladb    ClusterIP      None           <none>
    3306/TCP        25s

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/joomladb            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/joomladb-5688bc4cd6            1         1         1       25s
```

kubectl create -k .

Check if all of the resources get launched.

```
$ kubectl get all
```

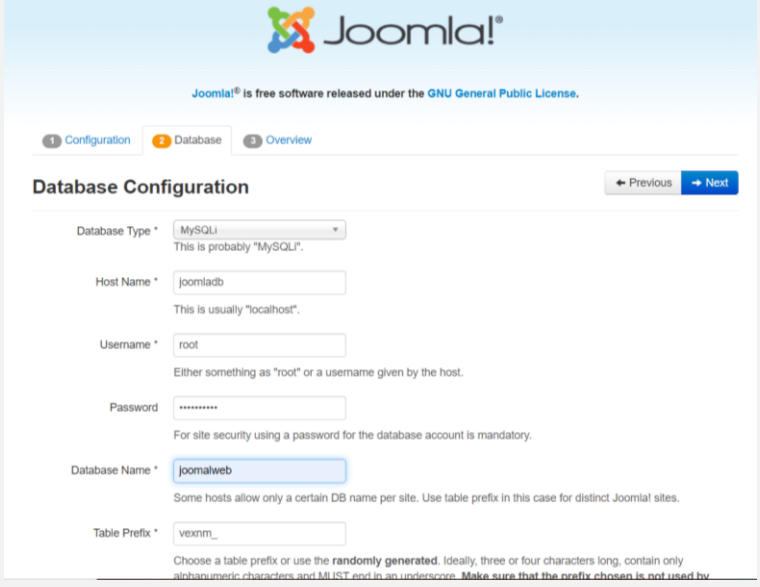
```
Administrator: Command Prompt
D:\Github Repositories\Joomla-on-EKS>kubectl get all
NAME                                    READY   STATUS    RESTARTS   AGE
pod/efs-provisioner-66b79c6f68-q6drn    1/1     Running   0          4m7s
pod/joomla-57cd8f7787-mn9mr             1/1     Running   0       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

NAME                TYPE           CLUSTER-IP     EXTERNAL-IP
    PORT(S)         AGE
service/joomla      LoadBalancer   10.100.2.178   aa303455663e74696aefa595ea4eaf8c-2002097864.ap-south-1.elb.amazonaws.co
m   80:31054/TCP    99s
service/joomladb    ClusterIP      None           <none>
    3306/TCP        99s

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/joomladb            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/joomladb-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!®

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

1 Configuration | 2 Database | 3 Overview

Finalisation

← Previous | → Install

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: Yes | No

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

Main Configuration

Site Name	ankush

Database Configuration

Database Type	mysqli

Joomla!®

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

Site Name	ankush
Site Offline	No
Administrator Email	ankush@gmail.com
Administrator Username	ankush
Administrator Password	***

Database Configuration

Database Type	mysqli
Host Name	joomladb
Username	root
Password	***
Database Name	joomlaweb
Table Prefix	vxxnm_
Old Database Process	Backup

Pre-Installation Check

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
Zlib Compression Support	Yes
XML Support	Yes
Database Support: (sqlite, pdo, mysqli)	Yes
INI Parser Support	Yes
JSON Support	Yes
configuration.php Writeable	Yes

Recommended settings:

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	Off
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.