Deploy a web application in AWS/Kubernetes

Amazon Elastic Kubernetes Service:

(Amazon EKS) is a fully managed Kubernetes service. Customers such as Intel, Snap, Intuit, GoDaddy, and Autodesk trust EKS to run their most sensitive and mission critical applications because of its security, reliability, and scalability.

Benefits of EKS:

1. High Availability
2. Serverless option
3. Secure
4. Build with the community

How does Amazon EKS work?

1. First, create an Amazon EKS cluster in the AWS Management Console or with the AWS CLI or one of the AWS SDKs.
2. Then, launch worker nodes that register with the Amazon EKS cluster. We provide you with an AWS CloudFormation template that automatically configures your nodes.
3. When your cluster is ready, you can configure your favorite Kubernetes tools (such as **kubectl**) to communicate with your cluster.
4. Deploy and manage applications on your Amazon EKS cluster the same way that you would with any other Kubernetes environment.

Pre-requisite for deploying web application on AWS EKS:

1. Install AWS CLI: First we need to install AWS CLI
2. Configure your AWS CLI credentials: Both eksctl and the AWS CLI require that you have AWS credentials configured in your environment. The **aws configure** command is the fastest way to set up your AWS CLI installation for general use.
3. Install eksctl: eksctl is a simple CLI tool for creating clusters on EKS
4. Install and configure kubectl: Kubernetes uses the **kubectl** command-line utility for communicating with the cluster API server.

Step 1:

In this step first we need to create IAM user with Administration Access.

Now we need to configure aws with the access key, secret key and region name to use aws from CLI.

```
C:\Users\himay>aws configure
AWS Access Key ID [****************REPZ]: AKIA3CGAB3HRRWTVREPZ
AWS Secret Access Key [****************UCab]: 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
Default region name [us-east-1]: us-east-1
Default output format [none]:
```

Step 2:

In this step we will create the aws eks cluster with the help of eksctl command**.**

Create a file name: cluster.yml

Code:

```
apiVersion: eksctl.io/v1alpha5

kind: ClusterConfig

metadata:

  name: mycluster

  region: us-east-1

nodeGroups:

  - name: ng-1

    instanceType: t2.micro

    desiredCapacity: 3

    ssh:

      publicKeyName: Webapp
```

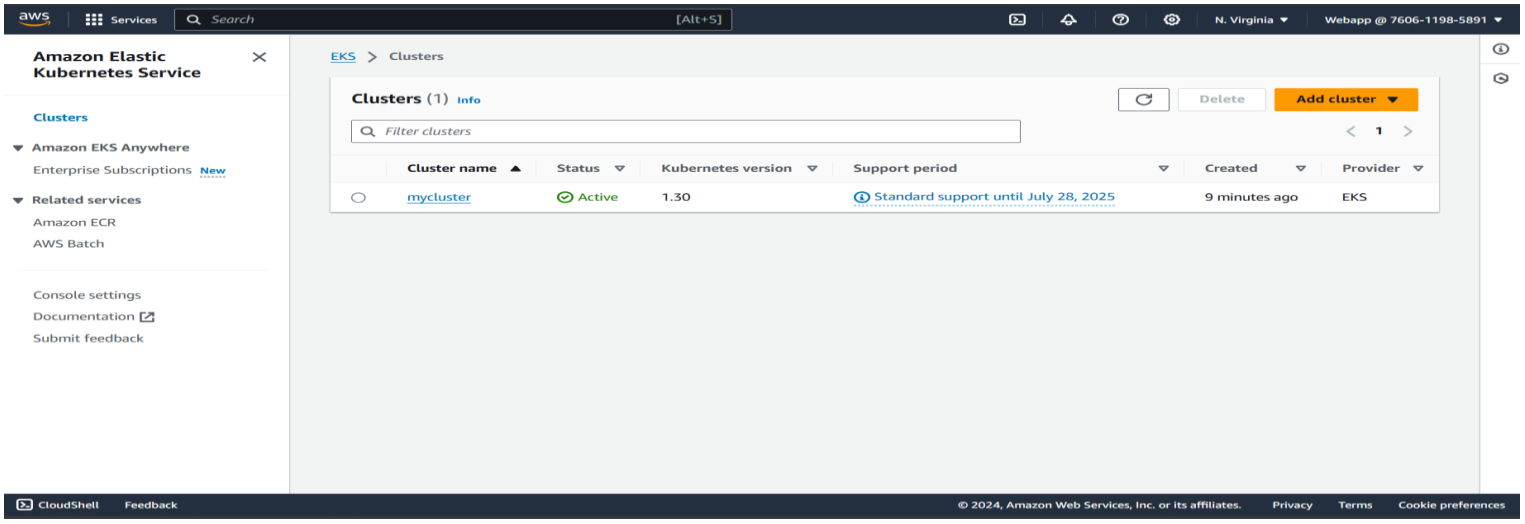
After this we need to run this command to create cluster.

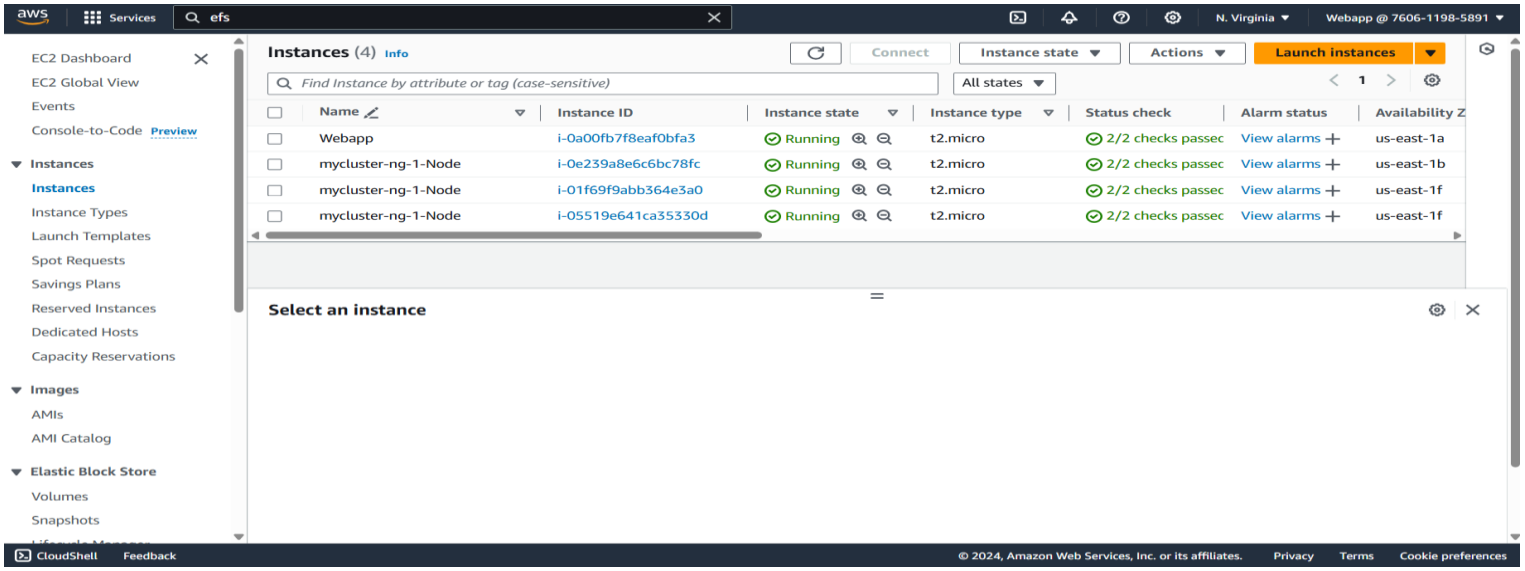
Command Prompt: eksctl create cluster -f cluster.yml

```
C:\Users\himay>aws eks list-clusters
{
    "clusters": []
}


C:\Users\himay>eksctl get clusters
No clusters found

C:\Users\himay>mkdir eks-task

C:\Users\himay>cd eks-task

C:\Users\himay\eks-task>notepad cluster.yml
```

```
C:\Users\himay\eks-task>eksctl create cluster -f cluster.yml
2024-07-21 22:48:14 [ℹ]  eksctl version 0.184.0
2024-07-21 22:48:14 [ℹ]  using region us-east-1
2024-07-21 22:48:20 [ℹ]  setting availability zones to [us-east-1f us-east-1b]
2024-07-21 22:48:20 [ℹ]  subnets for us-east-1f - public:192.168.0.0/19 private:192.168.64.0/19
2024-07-21 22:48:20 [ℹ]  subnets for us-east-1b - public:192.168.32.0/19 private:192.168.96.0/19
2024-07-21 22:48:22 [ℹ]  nodegroup "ng-1" will use "ami-07a876f98b5bdf972" [AmazonLinux2/1.30]
2024-07-21 22:48:22 [ℹ]  using EC2 key pair "Webapp"
2024-07-21 22:48:22 [ℹ]  using Kubernetes version 1.30
2024-07-21 22:48:22 [ℹ]  creating EKS cluster "mycluster" in "us-east-1" region with un-managed nodes
2024-07-21 22:48:22 [ℹ]  1 nodegroup (ng-1) was included (based on the include/exclude rules)
2024-07-21 22:48:22 [ℹ]  will create a CloudFormation stack for cluster itself and 1 nodegroup stack(s)
2024-07-21 22:48:22 [ℹ]  will create a CloudFormation stack for cluster itself and 0 managed nodegroup stack(s)
2024-07-21 22:48:22 [ℹ]  if you encounter any issues, check CloudFormation console or try 'eksctl utils describe-stacks --region=us-east-1 --cluster=myclust
er'
2024-07-21 22:48:22 [ℹ]  Kubernetes API endpoint access will use default of {publicAccess=true, privateAccess=false} for cluster "mycluster" in "us-east-1"
2024-07-21 22:48:22 [ℹ]  CloudWatch logging will not be enabled for cluster "mycluster" in "us-east-1"
2024-07-21 22:48:22 [ℹ]  you can enable it with 'eksctl utils update-cluster-logging --enable-types={SPECIFY-YOUR-LOG-TYPES-HERE (e.g. all)} --region=us-eas
t-1 --cluster=mycluster'
2024-07-21 22:48:22 [ℹ]  default addons vpc-cni, kube-proxy, coredns were not specified, will install them as EKS addons
2024-07-21 22:48:22 [ℹ]
2 sequential tasks: { create cluster control plane "mycluster",
    2 sequential sub-tasks: {
        2 sequential sub-tasks: {
            1 task: { create addons },
            wait for control plane to become ready,
        },
        create nodegroup "ng-1",
    }
}
2024-07-21 22:48:22 [ℹ]  building cluster stack "eksctl-mycluster-cluster"
2024-07-21 22:48:29 [ℹ]  deploying stack "eksctl-mycluster-cluster"
```

This will create an eks cluster with three nodes.





Now we need to update the kubectl config file so that kubectl can connect to the eks cluster.

```
C:\Users\himay\eks-task>kubectl config view
apiVersion: v1
clusters:
- cluster:
    certificate-authority-data: DATA+OMITTED
    server: https://D003B377F79A6B757A55A1D01328309B.gr7.us-east-1.eks.amazonaws.com
  name: mycluster.us-east-1.eksctl.io
contexts:
- context:
    cluster: mycluster.us-east-1.eksctl.io
    user: Webapp@mycluster.us-east-1.eksctl.io
  name: Webapp@mycluster.us-east-1.eksctl.io
current-context: Webapp@mycluster.us-east-1.eksctl.io
kind: Config
preferences: {}
users:
- name: Webapp@mycluster.us-east-1.eksctl.io
  user:
    exec:
      apiVersion: client.authentication.k8s.io/v1beta1
      args:
      - eks
      - get-token
      - --output
      - json
      - --cluster-name
      - mycluster
      - --region
      - us-east-1
      command: aws
      env:
      - name: AWS_STS_REGIONAL_ENDPOINTS
        value: regional
      interactiveMode: IfAvailable
      provideClusterInfo: false
```

Now create a new namespace.

Command Prompt: kubectl get ns

Command Prompt: kubectl config set-context –current –namespace=ns

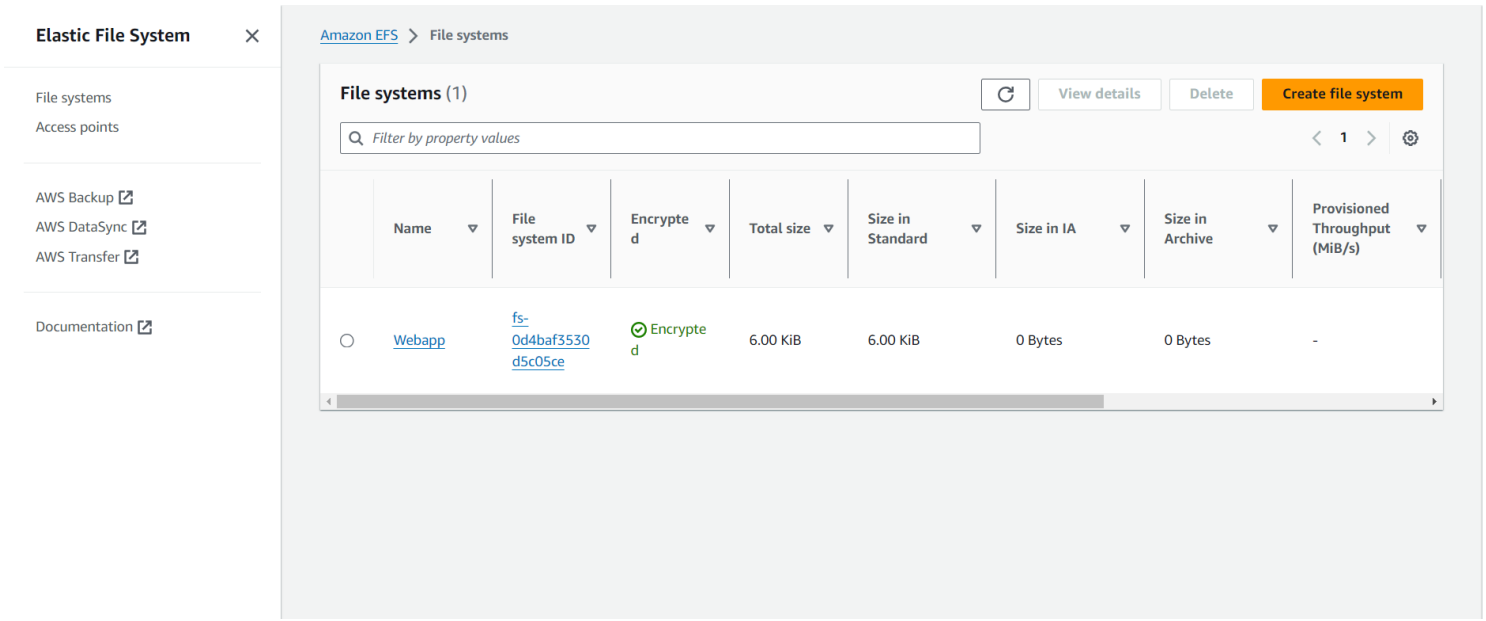
again

Command Prompt: kubectl get ns

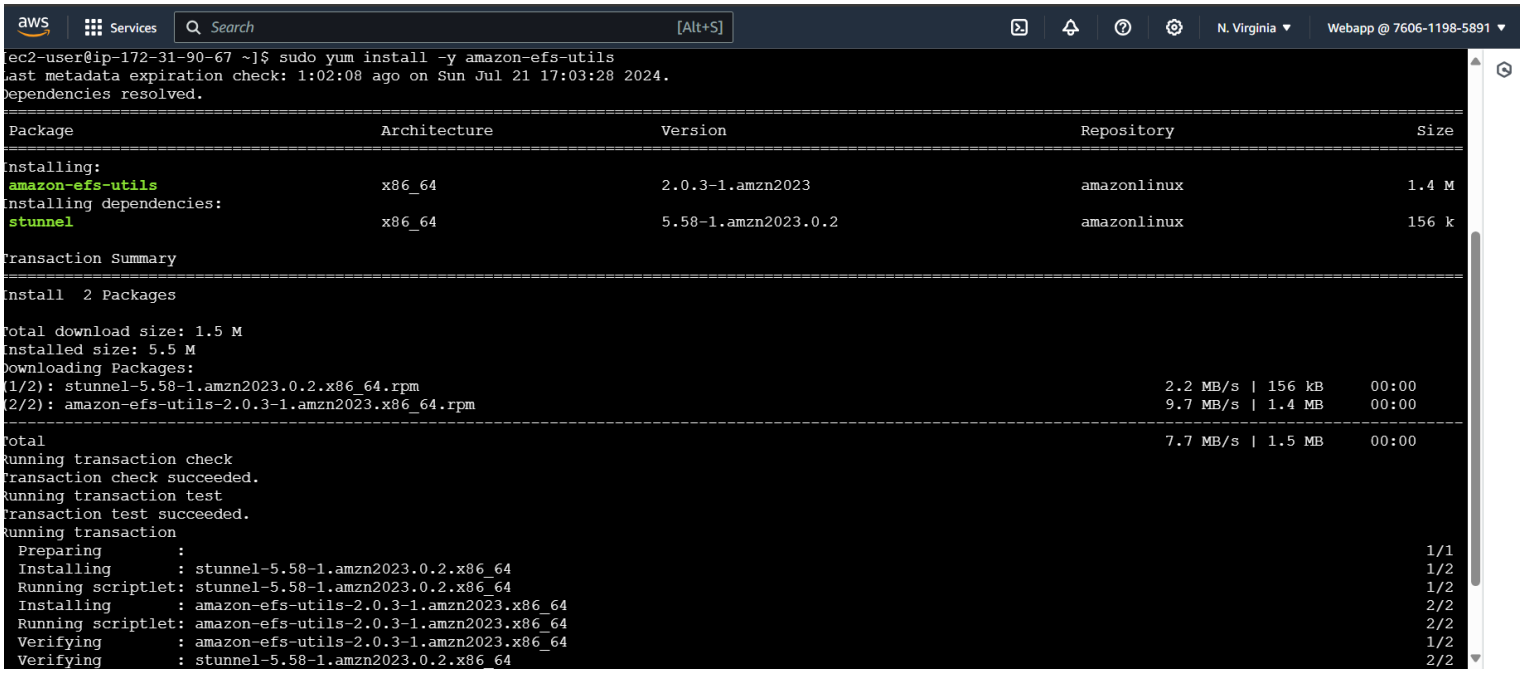
Step 3:

In this step we will create a VPC for our Web Application and for storage we are going to use EFS.

Firstly, we will create an EFS for storage.



Now for using EFS we need to install amazon-efs-utils on worker nodes of the cluster.



Step 4:

In this step we will create a few yml files for deploying our web application.

Firstly, we will create a provisioner for EFS.

Create a file name: provisioner.yml

Code:

apiVersion: apps/v1

kind: Deployment

```
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-0cfc1abfee4bf6829
            - name: AWS_REGION
              value: us-east-1
            - name: PROVISIONER_NAME
              value: eks-prov/aws-efs
          volumeMounts:
            - name: pv-volume
              mountPath: /persistentvolumes
      volumes:
        - name: pv-volume
          nfs:
            server: fs-0cfc1abfee4bf6829.efs.us-east-1.amazonaws.com
            path: /
```

Now we will create role binding.

Create a file name: rbac.yml

Code:

```
apiVersion: rbac.authorization.k8s.io/v1beta1
kind: ClusterRoleBinding
metadata:
  name: nfs-prov-role-binding
subjects:
  - kind: ServiceAccount
    name: default
    namespace: wp-MySQL
roleRef:
  kind: ClusterRole
  name: Cluster-admin
  apiGroup: rbac.authorization.k8s.io
```

Now we will create a storage class.

Create a file name: storage.yml

Code:

```
apiVersion: Storage.k8s.io/v1
kind: StorageClass
metadata:
  name: aws-efs
provisioner: eksprov/aws-efs
```

Now create MySQL deployment.

Create a file name: mysqldeploy.yml

Code:

```
apiVersion: v1
kind: Service
```

```
metadata:
  name: wordpress-mysql
  labels:
    app: wordpress
spec:
  ports:
    - port: 3306
  selector:
     app: wordpress
     tier: MySQL
     clusterIP: None
---
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
   name: efs-mysql
   annotations:
     volume.beta.kubernetes.io/Storage-class: "aws-efs"
spec:
  accessModes:
     - ReadWriteMany
  resources:
    requests:
      storage: 10Gi
---
apiVersion: apps/v1
kind: Deployment
metadata:
  name: wordpress-mysql
  labels:
    app: wordpress
spec:
```

```
selector:
  matchLabels:
    app: wordpress
    tier: MySQL
strategy:
  type: Recreate
template:
  metadata:
    labels:
      app:
      tier: MySQL
   spec:
     containers:
     - image: MySQL:5.6
       name: MySQL
       env:
       - name: MySQL_ROOT_PASSWORD
         valueFrom:
           sercetKeyRef:
             name: MySQL-pass
             key: password
       ports:
       - containerPort: 3306
         name: MySQL
       volumwMounts:
       - name: mysql-persistent-storage
         mountPath: /var/lib/MySQL
     volumes:
     - name: mysql-persistent-storage
       persistentVolumeClaim:
          claimName: efs-mysql
```

Now we will create WordPress deployment.

Create a file name: wpdeploy.yml

Code:

```
apiVersion: v1
kind: Service
metadata:
  name: wordpress
  labels:
    app: wordpress
 spec:
   ports:
     - port: 80
   selector:
     app: wordpress
     tier: frontend
   type: LoadBalancer
---
apiVerison: v1
kind: persistentVolumeClaim
metadata:
   name: efs-wordpress
   annotations:
     volume.beta.kubernetes.io/storage-class: "aws-efs"
spec:
  accessModes:
    - ReadWriteMany
  resources:
    requests:
      storage: 10Gi
---
 apiVersion: apps/v1
kind: Deployment
```

```
metadata:
  name: wordpress
  labels:
    app: wordpress
spec:
  selector:
    matchLabels:
      app: wordpress
      tier: frontend
  strategy:
    type: Recreate
  template:
    metadata:
      labels:
        app: wordpress
        tier: frontend
    spec:
      containers:
      - image: wordpress:4.8-apache
        name: wordpress
        env:
        - name: WORDPRESS_DB_HOST
          value: wordpress
        - name:  WORDPRESS_DB_PASSWORD
          valueFrom:
            secretKeyRef:
              name: mysql-pass
              key: password
        ports:
        - containersPort: 80
          name: wordpress
        volumeMounts:
```

```
      - name: wordpress-persistent-storage
        mountPath: /var/www/html
      volumes:
      - name: wordpress-persistent-storage
        persistentVolumeClaim:
          claimName: efs-wordpress
```

Now we will create a kustomization file.

Create a file name: kustomization.yml

Code:

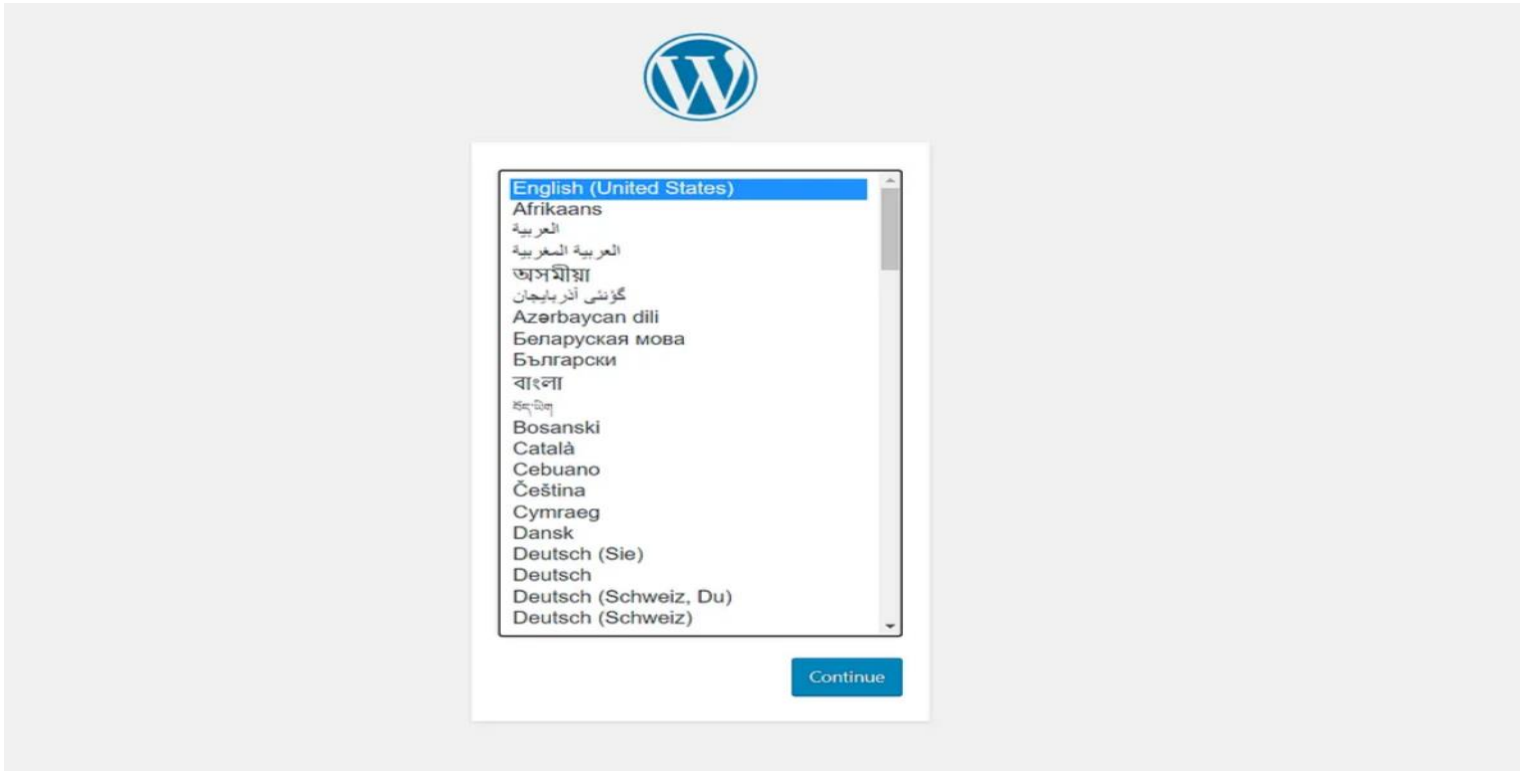
```
apiVersion: kustomize.config.k8s.io/v1beta1
kind: Kustomization
secretGenerator:
  - name: mysql-pass
    literals:
    - password=redhat
resources:
  - provisioner.yaml
  - rbac.yaml
  - storage.yaml
  - mysqldeploy.yaml
  - wpdeploy.yaml
```

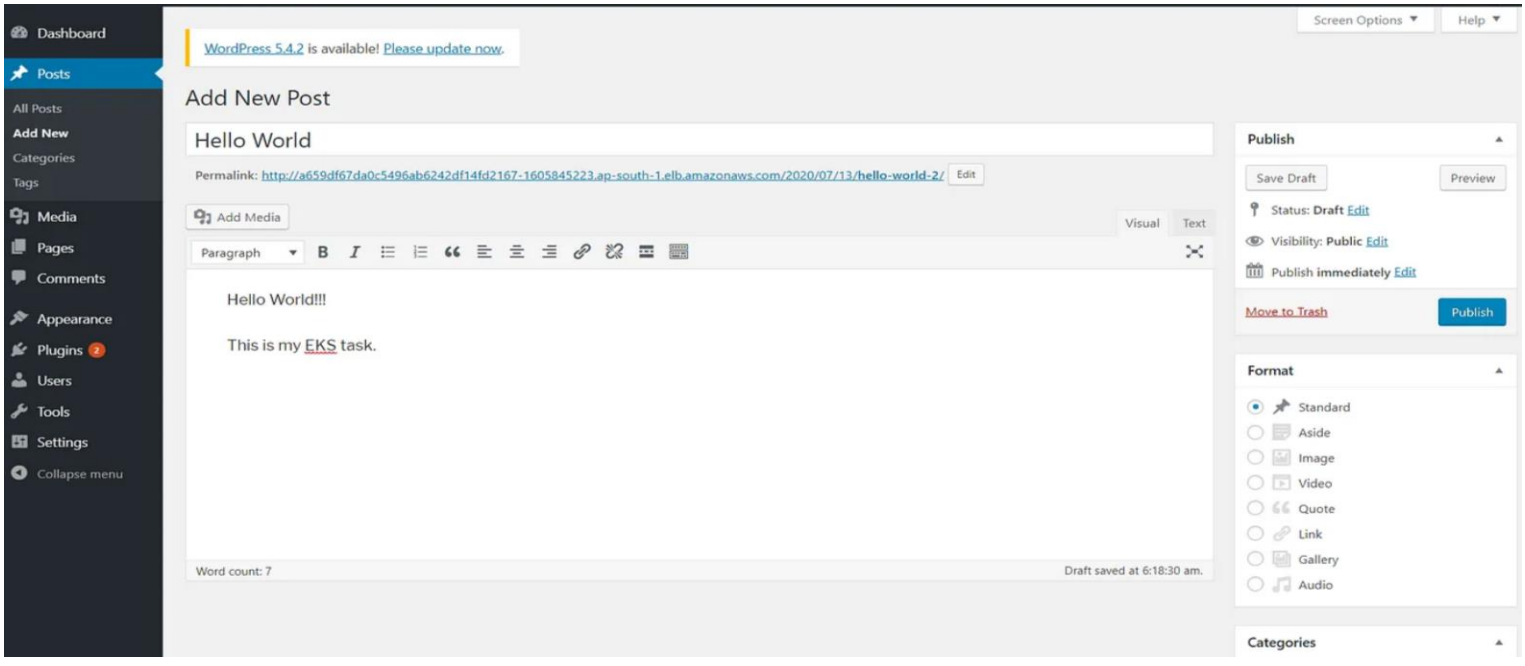
Now finally we will apply the kustomization file to create the complete setup.

Command Prompt: kubectl create -k .

```
storageclass.storage.k8s.io/aws-efs created
clusterrolebinding.rbac.authorization.k8s.io/nfs-prov-role-binding created
secret/mysql-pass-ctm2f4889c created
service/wordpress-mysql created
service/wordpress created
deployment.apps/efs-provisioner created
deployment.apps/wordpress-mysql created
deployment.apps/wordpress created
persistentvolumeclaim/efs-mysql created
persistentvolumeclaim/efs-wordpress created
```

```
NAME                                    READY   STATUS    RESTARTS   AGE
pod/efs-provisioner-577f7d4d59-kcswd    1/1     Running   0          5m22s
pod/wordpress-88cb86b9b-cdjnd           1/1     Running   0          5m22s
pod/wordpress-mysql-66b4cc9ccb-s7q6h    1/1     Running   0          5m22s

NAME                      TYPE           CLUSTER-IP       EXTERNAL-IP                                                                PORT(S)        AGE
service/kubernetes        ClusterIP      10.100.0.1       <none>                                                                     443/TCP        23m
service/wordpress         LoadBalancer   10.100.228.141   a659df67da0c5496ab6242df14fd2167-1605845223.ap-south-1.elb.amazonaws.com   80:32369/TCP   5m23s
service/wordpress-mysql   ClusterIP      None             <none>                                                                     3306/TCP       5m23s

NAME                              READY   UP-TO-DATE   AVAILABLE   AGE
deployment.apps/efs-provisioner   1/1     1            1           5m22s
deployment.apps/wordpress         1/1     1            1           5m22s
deployment.apps/wordpress-mysql   1/1     1            1           5m22s

NAME                                         DESIRED   CURRENT   READY   AGE
replicaset.apps/efs-provisioner-577f7d4d59   1         1         1       5m22s
replicaset.apps/wordpress-88cb86b9b          1         1         1       5m22s
replicaset.apps/wordpress-mysql-66b4cc9ccb   1         1         1       5m22s
```





Welcome

Welcome to the famous five-minute WordPress installation process! Just fill in the information below and you'll be on your way to using the most extendable and powerful personal publishing platform in the world.

Information needed

Please provide the following information. Don't worry, you can always change these settings later.

Site Title Hello World

Username

Usernames can have only alphanumeric characters, spaces, underscores, hyphens, periods, and the @ symbol.

Password •••••••••• Show

Weak

Important: You will need this password to log in. Please store it in a secure location.

Confirm Password Confirm use of weak password

Your Email

Double-check your email address before continuing.

Search Engine Visibility Discourage search engines from indexing this site

It is up to search engines to honor this request.

Install WordPress

POSTS

Hello World

Hello World!!!

This is my EKS task.

Hello world!

Welcome to WordPress. This is your first post. Edit or delete it, then start writing!

Search …

RECENT POSTS

Hello World

Hello world!

RECENT COMMENTS

A WordPress Commenter on Hello world!

ARCHIVES