Sprint-2
Online Plant Nursery Application

1) Docker compose:

Steps to create a Dockerfile:

- The first line has to start with the **FROM** keyword. It tells docker, from which base image you want to base your image from. In our case, we are creating an image from the **openjdk:16**.
- The **EXPOSE** instruction informs Docker that the container listens on the specified network ports at runtime.
- The **ADD** instruction copies new files, directories or remote file URLs from source and adds them to the file system of the image at the path destination.
- The **ENTRYPOINT** instruction makes your container run as an executable. The executable command for java is: [“java”, “-jar”, “jar-filename.jar”].

docker-compose.yml | Dockerfile

```
1 FROM openjdk:16-alpine3.13
2 EXPOSE 9001
3 ADD target/onlineplant-nursery.jar app.jar
4 ENTRYPOINT ["java","-jar","/app.jar"]
```

Installation steps for docker:

- Download Docker:
  https://docs.docker.com/desktop/windows/install/
- Double –click Install Docker.
- Follow the install wizard: accept the license, authorize the installer, and proceed with the installation.
- Click finish to launch Docker.
- Docker starts automatically.

Steps to create a docker image:

- Open a terminal and go to the directory with the Dockerfile.
- Now build the container image using the **docker build** command:

```
$ docker build –t <image-name> .
```

```
C:\Users\Lenovo\Documents\Capgemini-STS-Workspace\onlineplant-nursery>docker build -t onlineplant-nursery .
[+] Building 18.7s (8/8) FINISHED
 => [internal] load build definition from Dockerfile
 => => transferring dockerfile: 31B
 => [internal] load .dockerignore
 => => transferring context: 2B
 => [internal] load metadata for docker.io/library/openjdk:8
 => [auth] library/openjdk:pull token for registry-1.docker.io
 => [internal] load build context
 => => transferring context: 47.01MB
 => CACHED [1/2] FROM docker.io/library/openjdk:8@sha256:697cbb23e53ea6ea20eee311af9d1e39e7bc1caec2ca8b5709f7581f0c514866
 => => resolve docker.io/library/openjdk:8@sha256:697cbb23e53ea6ea20eee311af9d1e39e7bc1caec2ca8b5709f7581f0c514866
 => [2/2] ADD target/onlineplant-nursery.jar app.jar
 => exporting to image
 => => exporting layers
 => => writing image sha256:e2b2e5e26bb1707e0fe6d4bfc61edec62321336bed041a427a1b4a5cfbb73bb1
 => => naming to docker.io/library/onlineplant-nursery

Use 'docker scan' to run Snyk tests against images to find vulnerabilities and learn how to fix them
```

Steps to create docker-compose file:

- At the root of the app project, create a file named **docker-compose.yml**.
- In the compose file, we'll start off by defining the schema version.

```
version: "3.7"
```

- Next, we'll define the list of services (or containers) we want to run as part of our application.

```
version: "3.7"

services:
```

And now, we'll start migrating a service at a time into the compose file.
This Compose file defines two services: app and postgresql

- First, let's define the service entry and the image for the container.
- Migrate the -p 9001:9001 part of the command by defining the ports for the service.
- We will first define the new service and name it postgresql and define the ports.
- Finally, we only need to specify the environment variables.

docker-compose.yml | Dockerfile

```
1  version: '3.7'
2  services:
3    app:
4      container_name: onlineplant-nursery
5      image: roshini111/onlineplant-nursery:0.0.1
6      ports:
7        - 9001:9001
8      depends_on:
9        - postgresqldb
10     links:
11       - postgresqldb:postgres
12   postgresqldb:
13     image: "postgres:latest"
14     ports:
15       - 5432:5432
16     environment:
17       POSTGRES_USER: postgres
18       POSTGRES_PASSWORD: postgres
19
```

Steps to push the image onto the docker hub:

- Login to the docker hub with the username.
- Tag the image using the **docker tag** command:

```
$ docker tag <image-name> username/image-name
```

- Push the image into the docker hub using the **docker push** command:

```
$ docker push username/image-name
```

```
C:\Users\Lenovo\Documents\Capgemini-STS-Workspace\onlineplant-nursery>docker tag onlineplant-nursery roshini111/onlineplant-nursery:0.0.1

C:\Users\Lenovo\Documents\Capgemini-STS-Workspace\onlineplant-nursery>docker push roshini111/onlineplant-nursery:0.0.1
The push refers to repository [docker.io/roshini111/onlineplant-nursery]
eb98cc6dc91a: Pushed
4d327f577a2b: Layer already exists
a6159ee91199: Layer already exists
a81f1846a0d2: Layer already exists
3b441d7cb46b: Layer already exists
d3710de04cb3: Layer already exists
91f7336bbfff: Layer already exists
e2e8c39e0f77: Layer already exists
0.0.1: digest: sha256:e5687da1062e417acb78d58301ba007d98667752bbb045a400b74b4b59eb5a44 size: 2007
```

- Start up the application stack using the **docker-compose up** command.

```
$ docker-compose up
```

```
C:\Users\Lenovo\Documents\Capgemini-STS-Workspace\onlineplant-nursery>docker-compose up
[+] Running 2/0
 - Container onlineplant-nursery_postgresqldb_1  Created                                                    0.0s
 - Container onlineplant-nursery                 Created                                                    0.0s
Attaching to onlineplant-nursery, postgresqldb_1
postgresqldb_1       |
postgresqldb_1       | PostgreSQL Database directory appears to contain a database; Skipping initialization
postgresqldb_1       |
postgresqldb_1       | 2021-12-20 16:55:23.121 UTC [1] LOG:  starting PostgreSQL 14.1 (Debian 14.1-1.pgdg110+1) on x86_64-pc-linux-gnu, compiled by gcc (Debian 10.2.1-6
) 10.2.1 20210110, 64-bit
postgresqldb_1       | 2021-12-20 16:55:23.133 UTC [1] LOG:  listening on IPv4 address "0.0.0.0", port 5432
postgresqldb_1       | 2021-12-20 16:55:23.133 UTC [1] LOG:  listening on IPv6 address "::", port 5432
postgresqldb_1       | 2021-12-20 16:55:23.143 UTC [1] LOG:  listening on Unix socket "/var/run/postgresql/.s.PGSQL.5432"
postgresqldb_1       | 2021-12-20 16:55:23.179 UTC [27] LOG:  database system shutdown was interrupted; last known up at 2021-12-20 11:34:45 UTC
postgresqldb_1       | 2021-12-20 16:55:23.787 UTC [27] LOG:  database system was not properly shut down; automatic recovery in progress
postgresqldb_1       | 2021-12-20 16:55:23.840 UTC [27] LOG:  redo starts at 0/16FABF8
postgresqldb_1       | 2021-12-20 16:55:23.915 UTC [27] LOG:  invalid record length at 0/1740EA8: wanted 24, got 0
postgresqldb_1       | 2021-12-20 16:55:23.915 UTC [27] LOG:  redo done at 0/1740E70 system usage: CPU: user: 0.01 s, system: 0.06 s, elapsed: 0.07 s
postgresqldb_1       | 2021-12-20 16:55:24.129 UTC [1] LOG:  database system is ready to accept connections
onlineplant-nursery  |
onlineplant-nursery  |   .   ____          _            __ _ _
onlineplant-nursery  |  /\\ / ___'_ __ _ _(_)_ __  __ _ \ \ \ \
onlineplant-nursery  | ( ( )\___ | '_ | '_| | '_ \/ _` | \ \ \ \
onlineplant-nursery  |  \\/  ___)| |_)| | | | | || (_| |  ) ) ) )
onlineplant-nursery  |   '  |____| .__|_| |_|_| |_\__, | / / / /
onlineplant-nursery  |  =========|_|==============|___/=/_/_/_/
onlineplant-nursery  |  :: Spring Boot ::                (v2.4.5)
onlineplant-nursery  |
onlineplant-nursery  | 2021-12-20 16:55:30.692  INFO 1 --- [           main] o.PlantNurseryApplicationRoshApplication : Starting PlantNurseryApplicationRoshApplication
v0.0.1-SNAPSHOT using Java 1.8.0_312 on a447d1ebfc5a with PID 1 (/app.jar started by root in /)
onlineplant-nursery  | 2021-12-20 16:55:30.728  INFO 1 --- [           main] o.PlantNurseryApplicationRoshApplication : No active profile set, falling back to default p
rofiles: default
onlineplant-nursery  | 2021-12-20 16:55:35.833  INFO 1 --- [           main] .s.d.r.c.RepositoryConfigurationDelegate : Bootstrapping Spring Data JPA repositories in DE
FAULT mode.
onlineplant-nursery  | 2021-12-20 16:55:36.035  INFO 1 --- [           main] .s.d.r.c.RepositoryConfigurationDelegate : Finished Spring Data repository scanning in 86 m
s. Found 0 JPA repository interfaces.
onlineplant-nursery  | 2021-12-20 16:55:43.167  INFO 1 --- [           main] o.s.b.w.embedded.tomcat.TomcatWebServer  : Tomcat initialized with port(s): 9001 (http)
onlineplant-nursery  | 2021-12-20 16:55:43.261  INFO 1 --- [           main] o.apache.catalina.core.StandardService   : Starting service [Tomcat]
onlineplant-nursery  | 2021-12-20 16:55:43.262  INFO 1 --- [           main] org.apache.catalina.core.StandardEngine  : Starting Servlet engine: [Apache Tomcat/9.0.45]
onlineplant-nursery  | 2021-12-20 16:55:44.319  INFO 1 --- [           main] o.a.c.c.C.[Tomcat].[localhost].[/]       : Initializing Spring embedded WebApplicationConte
xt
onlineplant-nursery  | 2021-12-20 16:55:44.320  INFO 1 --- [           main] w.s.c.ServletWebServerApplicationContext : Root WebApplicationContext: initialization compl
eted in 12926 ms
```

```
herefore, database queries may be performed during view rendering. Explicitly configure spring.jpa.open-in-view to disable this warning
onlineplant-nursery  | 2021-12-20 16:56:13.014  INFO 1 --- [           main] o.s.s.concurrent.ThreadPoolTaskExecutor  : Initializing ExecutorService 'applicationTaskExe
cutor'
onlineplant-nursery  | 2021-12-20 16:56:21.123  INFO 1 --- [           main] o.s.b.w.embedded.tomcat.TomcatWebServer  : Tomcat started on port(s): 9001 (http) with cont
ext path ''
onlineplant-nursery  | 2021-12-20 16:56:21.198  INFO 1 --- [           main] o.PlantNurseryApplicationRoshApplication : Started PlantNurseryApplicationRoshApplication i
n 52.597 seconds (JVM running for 56.172)
onlineplant-nursery  | Springboot Started...
```

2) Kubernetes Deployment:

Step-1 Change the application properties as the following:

```
spring.datasource.driverClassName=org.postgresql.Driver
spring.datasource.url=jdbc:postgresql://${DB_HOST}:5432/${DB_NAME}
spring.datasource.username=${POSTGRES_USER}
spring.datasource.password=${POSTGRES_PASSWORD}
spring.jpa.hibernate.ddl-auto=update
```

Step-2 Creating manifest files:

Defining a service:

- The specification creates a new Service object named "**plant-nursery-postgres**", which targets TCP port 9001 on any Pod with the **app=plant-nursery-postgres** label.
- The default protocol for Services is TCP.
- Kubernetes assigns this Service an IP address which is used by the service proxies.
- The controller for the Service selector continuously scans for Pods that match its selector, and then posts any updates to an Endpoint object also named **"plant-nursery-postgres"**.
- Port definitions in Pods have names, and you can reference these names in the targetPort attribute of a Service.

deployment.yaml

```yaml
kind: Service
apiVersion: v1
metadata:
  name: plant-nursery-postgres
  labels:
    name: plant-nursery-postgres
spec:
  ports:
    #- nodePort: 30163
    - port: 9001
      targetPort: 9001
      protocol: TCP
  selector:
    app: plant-nursery-postgres
  #type: NodePort

```

Defining a deployment:

- It creates a ReplicaSet to bring up three **plant-nursery-postgres** Pods.
- A deployment named **plant-nursery-postgres** is created, indicated by the **.metadata.name** field.
- The deployment creates three replicated Pods, indicated by the **.spec.replicas** field.
- The **.spec.selector** field defines how the Deployment finds which Pods to manage. In this case, you select a label that is defined in the Pod template (app: plant-nursery-postgres).
- The template field contains the following sub-fields:

- The Pods are labelled app: plant-nursery-postgres using the .**metadata.labels** field.
- The Pod template's specification, or .**template.spec field**, indicates that the Pods run one container, plant-nursery-postgres, which runs the plant- nursery-postgres DockerHub image.

```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: onlineplant-nursery
spec:
  selector:
    matchLabels:
      app: onlinepant-nursery
  replicas: 3
  template:
    metadata:
      labels:
        app: onlineplant-nursery
    spec:
      containers:
        - name: onlineplant-nursery
          image: roshini111/onlineplant-nursery:0.0.1
          ports:
            - containerPort: 9001
          env:   # Setting Enviornmental Variables
            - name: DB_HOST   # Setting Database host address from configMap
              valueFrom:
                configMapKeyRef:
                  name: postgres-conf  # name of configMap
                  key: host
            - name: DB_NAME  # Setting Database name from configMap
              valueFrom:
                configMapKeyRef:
                  name: postgres-conf
                  key: name
            - name: POSTGRES_USER  # Setting Database username from Secret
              valueFrom:
                secretKeyRef:
                  name: postgres-credentials # Secret Name
                  key: postgres_user
            - name: POSTGRES_PASSWORD # Setting Database password from Secret
              valueFrom:
                secretKeyRef:
                  name: postgres-credentials
                  key: postgres_password
```

Creating a ConfigMap file:

The ConfigMap configures the container(s) in Pod based on the data in the ConfigMap.

```yaml
apiVersion: v1
kind: ConfigMap
metadata:
  name: postgres-conf
data:
  host: postgres
  name: postgres
```

Creating a secret file:

- A Secret is an object that contains a small amount of sensitive data such as a password, a token, or a key.
- When creating a Pod, Kubernetes automatically creates a service account Secret and automatically modifies your Pod to use this Secret.
- When using this Secret type, the data field of the Secret must contain one of the following two keys:
    - username: the user name for authentication.
    - password: the password or token for authentication.

```
1 apiVersion: v1
2 kind: Secret
3 metadata:
4   name: postgres-credentials
5 data:
6   postgres_user: postgres
7   postgres_password: postgres
```

Step-3 Installation of minikube:

- Download the latest release of minikube from: https://minikube.sigs.k8s.io/docs/start/
- From a terminal with administrator access (but not logged in as root), run:

```
$ minikube start
```

```
C:\Users\Lavanya>minikube start
* minikube v1.24.0 on Microsoft Windows 10 Pro 10.0.19042 Build 19042
* Using the docker driver based on existing profile
* Starting control plane node minikube in cluster minikube
* Pulling base image ...
! Executing "docker container inspect minikube --format={{.State.Status}}" took an unusually long time: 3.5029163s
* Restarting the docker service may improve performance.
* Restarting existing docker container for "minikube" ...
* Preparing Kubernetes v1.22.3 on Docker 20.10.8 ...
* Verifying Kubernetes components...
  - Using image gcr.io/k8s-minikube/storage-provisioner:v5
* Enabled addons: storage-provisioner, default-storageclass
* Done! kubectl is now configured to use "minikube" cluster and "default" namespace by default
```

- Docker engine already provides kubectl pre-installed, so in order to check whether it is installed check for the version by using the following command:

```
$ kubectl version
```

```
C:\Users\Lenovo>kubectl version
Client Version: version.Info{Major:"1", Minor:"22", GitVersion:"v1.22.4", GitCommit:"b695d79d4f967c403a96986f1750a35eb75e75f1", GitTreeState:"clean", BuildDate:"2021-11
-17T15:48:33Z", GoVersion:"go1.16.10", Compiler:"gc", Platform:"windows/amd64"}
Server Version: version.Info{Major:"1", Minor:"21+", GitVersion:"v1.21.2-eks-06eac09", GitCommit:"5f6d83fe4cb7febb5f4f4e39b3b2b64ebbbe3e97", GitTreeState:"clean", Build
Date:"2021-09-13T14:20:15Z", GoVersion:"go1.16.5", Compiler:"gc", Platform:"linux/amd64"}
```

Step-4 Build and push the image to docker hub:

Build and push the image into the docker hub using the above mentioned commands.

```
plant-nursery-postgres                  latest    cabe133c334a   29 hours ago     573MB
roshini111/plant-nursery-postgres       latest    cabe133c334a   29 hours ago     573MB
plant-nursery                           latest    5ce86f6f0b61   2 days ago       371MB
roshini111/plant-nursery                latest    5ce86f6f0b61   2 days ago       371MB
employees-service-postgres              latest    2d9c91e2bcc8   3 days ago       565MB
roshini111/employees-postgres           latest    2d9c91e2bcc8   3 days ago       565MB
roshini111/employees-service-postgres   latest    2d9c91e2bcc8   3 days ago       565MB
employee-service-postgres               latest    4509ec39f4ea   3 days ago       565MB
<none>                                  <none>    68ee782825a8   4 days ago       565MB
employee-service-h2                     latest    744614bc4f23   5 days ago       567MB
roshini111/employee-service-h2          latest    744614bc4f23   5 days ago       567MB
employee-service                        latest    2905ecbc78c2   5 days ago       546MB
roshini111/employee-service             latest    2905ecbc78c2   5 days ago       546MB
```

Step-5 Creating yaml files using kubectl command:

- To create a file, the following command is used:

```
$ kubectl create –f <file-name>
```

```
D:\OnlinePlant-Nursery\OnlinePlant-Nursery\k8s>kubectl create -f deployment.yaml
service/plant-nursery-postgres created
deployment.apps/plant-nursery-postgres created

D:\OnlinePlant-Nursery\OnlinePlant-Nursery\k8s>
D:\OnlinePlant-Nursery\OnlinePlant-Nursery\k8s>kubectl create -f postgres-configmap.yaml
configmap/postgres-conf created

D:\OnlinePlant-Nursery\OnlinePlant-Nursery\k8s>
D:\OnlinePlant-Nursery\OnlinePlant-Nursery\k8s>kubectl create -f postgres-deployment.yaml
service/postgres created
persistentvolumeclaim/postgres-pv-claim created
deployment.apps/postgres created

D:\OnlinePlant-Nursery\OnlinePlant-Nursery\k8s>kubectl create -f postgres-credentials.yaml
secret/postgres-credentials created
```

- To view all the pods, deployments and services created, we use the following command:

```
$ kubectl get all
```

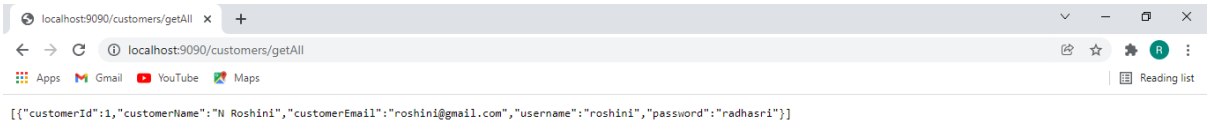
```
C:\Users\Lenovo>kubectl get all
NAME                                        READY   STATUS    RESTARTS      AGE
pod/plant-nursery-postgres-576d7c8764-6ff7k   1/1     Running   2 (13m ago)   25h
pod/plant-nursery-postgres-576d7c8764-jss89   1/1     Running   2 (13m ago)   25h
pod/plant-nursery-postgres-576d7c8764-vsggp   1/1     Running   2 (13m ago)   25h
pod/postgres-6f4cd8968f-dv5kv                 1/1     Running   2 (13m ago)   25h

NAME                              TYPE        CLUSTER-IP      EXTERNAL-IP   PORT(S)    AGE
service/kubernetes                ClusterIP   10.96.0.1       <none>        443/TCP    40h
service/plant-nursery-postgres    ClusterIP   10.111.34.193   <none>        9001/TCP   25h
service/postgres                  ClusterIP   None            <none>        5432/TCP   25h

NAME                                    READY   UP-TO-DATE   AVAILABLE   AGE
deployment.apps/plant-nursery-postgres   3/3     3            3           25h
deployment.apps/postgres                 1/1     1            1           25h

NAME                                              DESIRED   CURRENT   READY   AGE
replicaset.apps/plant-nursery-postgres-576d7c8764   3         3         3       25h
replicaset.apps/postgres-6f4cd8968f                 1         1         1       25h
```

- To use clusterIP, we need to use port-forward command:

```
$ kubectl port-forward svc/image-name 9090:9001
```

```
C:\Users\Lenovo>kubectl port-forward svc/plant-nursery-postgres 9090:9001
Forwarding from 127.0.0.1:9090 -> 9001
Forwarding from [::1]:9090 -> 9001
Handling connection for 9090
Handling connection for 9090
```

Browser Screenshot:



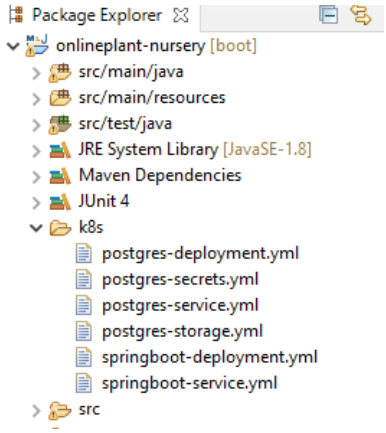
3) EKS Deployment:

Step-1 Change the application properties as the following:

```
spring.datasource.driverClassName=org.postgresql.Driver
spring.datasource.url=jdbc:postgresql://${POSTGRES_HOST}:5432/postgres
spring.datasource.username=${POSTGRES_USER}
spring.datasource.password=${POSTGRES_PASSWORD}
spring.jpa.hibernate.ddl-auto=update
```

Step-2 Creating manifest files:

- The manifest files also known as the “yaml” files are created just like the way these files are created in the kubernetes deployment.
- These files are as the following:
    - postgres-storage.yml
    - postgres-secrets.yml
    - postgres-service.yml
    - postgres-deployment.yml
    - springboot-deployment.yml
    - springboot-service.yml



Step-3 Installation of AWS CLI:

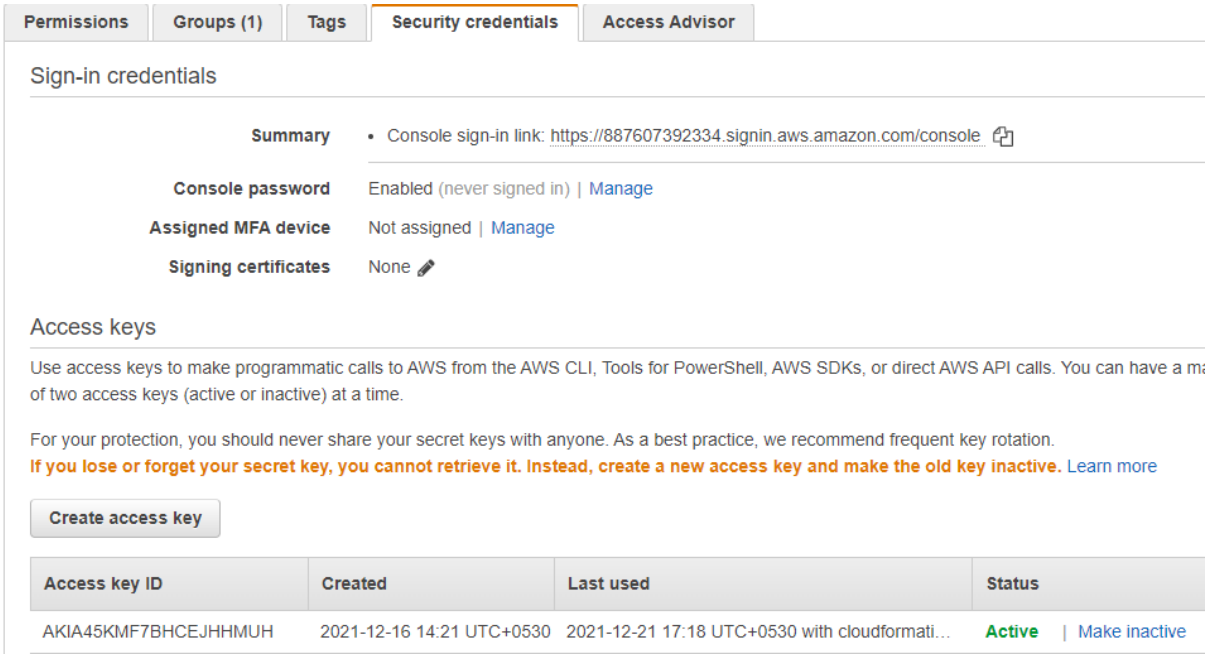
- Download and run the AWS CLI MSI installer for Windows (64-bit)

  https://awscli.amazonaws.com/AWSCLIV2.msi

- To confirm the installation, open the **Start** menu, search for `cmd` to open a command prompt window, and at the command prompt use the `aws --version` command.

```
C:\> aws --version
aws-cli/2.3.7 Python/3.8.8 Windows/10 exe/AMD64 prompt/off
```

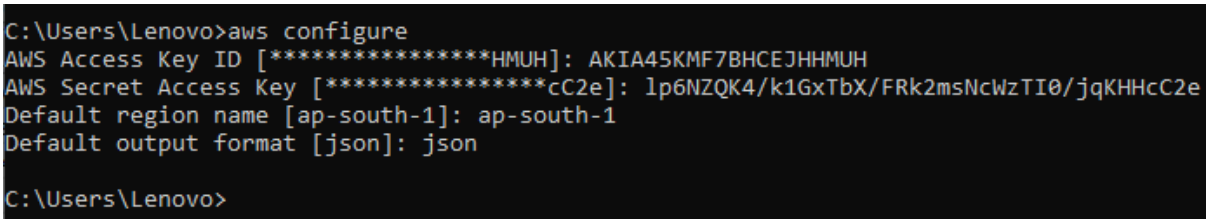
- We need secret keys from AWS IAM account. Go to IAM in AWS and generate access key by going into the security credentials section in users.



- Download the access key generated.
- Now, in cmd configure the AWS by using the following command:

```
$ aws configure
```

Then enter the access key id, secret access key, region name and the output format.



Step-4 Installation of eksctl:

- For installing the eksctl, chocolatey has to be installed first.
- In order to install Chocolatey, first, ensure that you are using an **administrative shell.**
- Copy the text specific to your command shell **- cmd.exe**.
- Paste the copied text into your shell and press Enter.
  @"%SystemRoot%\System32\WindowsPowerShell\v1.0\powershell.exe" -NoProfile -InputFormat None -ExecutionPolicy Bypass -Command "[System.Net.ServicePointManager]::SecurityProtocol = 3072; iex ((New-Object System.Net.WebClient).DownloadString('https://community.chocolatey.org/install.ps1'))" && SET "PATH=%PATH%;%ALLUSERSPROFILE%\chocolatey\bin"
- Wait a few seconds for the command to complete.

- After installing eksctl, run the commands as shown in the attached screenshot.

To install or upgrade `eksctl` on Windows using Chocolatey

1. If you do not already have Chocolatey installed on your Windows system, see Installing Chocolatey.
2. Install or upgrade `eksctl`.
   - Install the binaries with the following command:

```
choco install -y eksctl
```

   - If they are already installed, run the following command to upgrade:

```
choco upgrade -y eksctl
```

3. Test that your installation was successful with the following command.

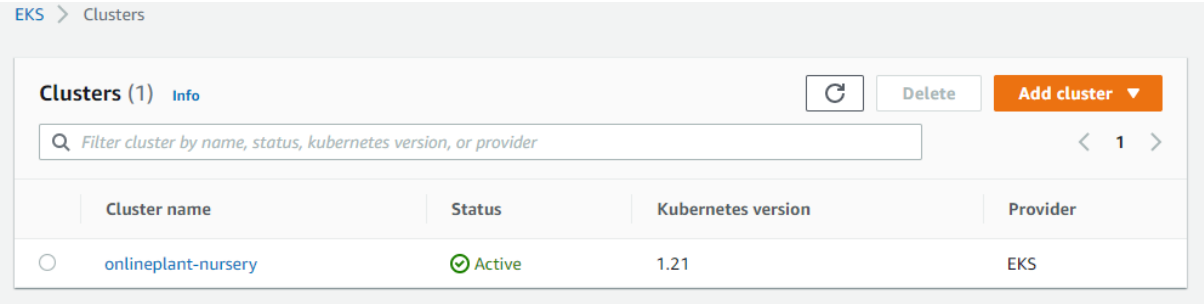
```
eksctl version
```

```
C:\Users\Lenovo>eksctl version
0.76.0
```

Step-5 Create a cluster:

- In order to create a cluster, the following command is used:

```
$ eksctl create cluster --name <cluster-name>  --version 1.21 --region <region-name>
--nodegroup-name <node-group-name> --node-type t2.micro –nodes 2
```



- To create or update kubeconfig for our cluster:

```
$ aws eks --region <region-code> update-kubeconfig --name <cluster-name>
```

- Now, create files using the kubectl command:

```
$ kubectl apply –f <file-name>
```

```
C:\Users\Lenovo\Documents\Capgemini-STS-Workspace\onlineplant-nursery\k8s>aws eks --region ap-south-1 update-kubeconfig --name onlineplant-nursery
Added new context arn:aws:eks:ap-south-1:887607392334:cluster/onlineplant-nursery to C:\Users\Lenovo\.kube\config

C:\Users\Lenovo\Documents\Capgemini-STS-Workspace\onlineplant-nursery\k8s>kubectl get all
NAME                 TYPE        CLUSTER-IP   EXTERNAL-IP   PORT(S)   AGE
service/kubernetes   ClusterIP   10.100.0.1   <none>        443/TCP   10m

C:\Users\Lenovo\Documents\Capgemini-STS-Workspace\onlineplant-nursery\k8s>kubectl apply -f postgres-deployment.yml
deployment.apps/postgres created

C:\Users\Lenovo\Documents\Capgemini-STS-Workspace\onlineplant-nursery\k8s>kubectl apply -f postgres-secrets.yml
secret/postgres-secrets created

C:\Users\Lenovo\Documents\Capgemini-STS-Workspace\onlineplant-nursery\k8s>kubectl apply -f postgres-service.yml
service/postgres created

C:\Users\Lenovo\Documents\Capgemini-STS-Workspace\onlineplant-nursery\k8s>kubectl apply -f postgres-storage.yml
persistentvolume/postgres-pv-volume created
persistentvolumeclaim/postgres-pv-claim created

C:\Users\Lenovo\Documents\Capgemini-STS-Workspace\onlineplant-nursery\k8s>ku_
```

- To view all the pods, deployments and services use the following kubectl command:

```
$ kubectl get all
```

```
C:\Users\Lenovo\Documents\Capgemini-STS-Workspace\onlineplant-nursery\k8s>aws eks --region ap-south-1 update-kubeconfig --name onlineplant-nursery
Updated context arn:aws:eks:ap-south-1:887607392334:cluster/onlineplant-nursery in C:\Users\Lenovo\.kube\config

C:\Users\Lenovo\Documents\Capgemini-STS-Workspace\onlineplant-nursery\k8s>kubectl get all
NAME                                     READY   STATUS    RESTARTS   AGE
pod/onlineplant-nursery-85574d4886-bh4kb   1/1     Running   0          24h
pod/postgres-5bdb4fc5f9-9cg64              1/1     Running   0          24h

NAME                           TYPE           CLUSTER-IP       EXTERNAL-IP                                                              PORT(S)          AGE
service/kubernetes             ClusterIP      10.100.0.1       <none>                                                                   443/TCP          25h
service/onlineplant-nursery    LoadBalancer   10.100.180.203   a2269844e5c8e4a2dae717d59e4ca3bd-705566216.ap-south-1.elb.amazonaws.com   9001:31494/TCP   24h
service/postgres               NodePort       10.100.56.9      <none>                                                                   5432:32751/TCP   24h

NAME                                   READY   UP-TO-DATE   AVAILABLE   AGE
deployment.apps/onlineplant-nursery    1/1     1            1           24h
deployment.apps/postgres               1/1     1            1           24h

NAME                                              DESIRED   CURRENT   READY   AGE
replicaset.apps/onlineplant-nursery-85574d4886    1         1         1       24h
replicaset.apps/postgres-5bdb4fc5f9               1         1         1       24h
```

- Now, check in the browser by pasting the IP address in the browser: *a2269844e5c8e4a2dae717d59e4ca3bd-705566216.ap-south-1.elb.amazonaws.com:9001/customers/getAll*