**ECS CloudFormation with Fluent Bit Logging Solution**

## **Introduction**

This document outlines the steps to deploy a containerized application on AWS ECS using CloudFormation, along with setting up Fluent Bit for log forwarding. Fluent Bit will parse and forward logs to Amazon OpenSearch Service and Amazon S3. Additionally, the setup includes a comprehensive VPC configuration for a 3-tier architecture.

## **Problem Statement**

Managing containerized applications involves monitoring and logging challenges. Traditional logging methods are insufficient for dynamic environments like ECS. There is a need for a robust solution to collect, parse, and forward logs to centralized systems for analysis and storage.

## **Solution**

This solution uses AWS CloudFormation to automate the deployment of an ECS service with Fluent Bit for log management. Fluent Bit is configured to parse logs using custom parsers and forward them to Amazon OpenSearch Service and S3. The deployment includes a custom VPC setup to ensure secure and isolated networking for the ECS service and OpenSearch instance.

## **Steps**

### **1. VPC Configuration**

**Create a VPC named ecs-fluentbit with the following structure:**

* **NAT Gateway: ecs-fluentbit**
* **Subnets:**
  + **Public:**
    - ecs-fluentbit-public-subnet-1
    - ecs-fluentbit-public-subnet-2
    - ecs-fluentbit-public-subnet-3
  + **Private:**
    - ecs-fluentbit-private-subnet-1
    - ecs-fluentbit-private-subnet-2
    - ecs-fluentbit-private-subnet-3
  + **Database:**
    - ecs-fluentbit-private-subnet-4
    - ecs-fluentbit-private-subnet-5
    - ecs-fluentbit-private-subnet-6
* **Route Tables:**
  + Public Route Table: ecs-fluentbit-public-routetable with an Internet Gateway (IGW)
  + Private Route Table: ecs-fluentbit-private-rt with a NAT Gateway

### 

### 

### **2. OpenSearch Service Configuration**

Deploy Amazon OpenSearch Service in the same VPC within private subnets. Ensure it has a security group allowing traffic from the ECS tasks.

### 

### 

### **3. ECS Configuration**

Use CloudFormation to deploy an ECS cluster with the necessary resources. Include the following files in the GitHub repository for the configuration:

#### **Parsers.conf**

| [PARSER]  Name nginx  Format regex  Regex ^(?<remote>[^ ]\*) (?<host>[^ ]\*) (?<user>[^ ]\*) \[(?<time>[^\]]\*)\] "(?<method>\S+)(?: +(?<path>[^\"]\*?)(?: +\S\*)?)?" (?<code>[^ ]\*) (?<size>[^ ]\*)(?: "(?<referer>[^\"]\*)" "(?<agent>[^\"]\*)")? \"-\"$  Time\_Key time  Time\_Format %d/%b/%Y:%H:%M:%S %z |
| --- |

**Imagedefinitions.json**

| **[{"name":"sample-app","imageUri":"785975698029.dkr.ecr.ap-south-1.amazonaws.com/ecs-package-repository"}]** |
| --- |

**fluent-bit.conf**

| [SERVICE]  Parsers\_File parsers.conf  Log\_Level debug  Daemon off  http\_listen 0.0.0.0  HTTP\_Server On  HTTP\_Listen 0.0.0.0  HTTP\_Port 80  [INPUT]  Name tail  Path /var/log/secure  Path\_Key filename  Refresh\_Interval 5  Tag system\_logs  [FILTER]  Name parser  Match \*\*  Parser nginx  Key\_Name log  [OUTPUT]  Name opensearch  Match \*  Host vpc-ecs-fluentbit-h6oh7moqra3nsi5yc4fhljh2bq.ap-south-1.es.amazonaws.com  Port 443  Index my\_index  AWS\_Auth On  AWS\_Region ap-south-1  tls On  Suppress\_Type\_Name On  [OUTPUT]  Name s3  Match \*  bucket aws-cloud-formation-test  region ap-south-1  total\_file\_size 250M  s3\_key\_format /$TAG[2]/$TAG[0]/%Y/%m/%d/%H/%M/%S/$UUID.gz  s3\_key\_format\_tag\_delimiters .- |
| --- |

**Dockerfile**

| FROM amazon/aws-for-fluent-bit:latest  # Add your Fluent Bit configuration files  ADD fluent-bit.conf /fluent-bit/etc/  ADD parsers.conf /fluent-bit/etc/  # Expose ports (optional based on your application's requirements)  EXPOSE 2020  EXPOSE 80  # Define volumes for logs and other data  VOLUME /var/log  # Set the command to run when the container starts  CMD ["/fluent-bit/bin/fluent-bit", "-c", "/fluent-bit/etc/fluent-bit.conf"] |
| --- |

**buildspec.yml**

| version: 0.2  run-as: root  phases:  pre\_build:  commands:  - echo Logging in to Amazon ECR...  - aws ecr get-login-password --region ap-south-1 | docker login --username AWS --password-stdin 785975698029.dkr.ecr.ap-south-1.amazonaws.com  - REPOSITORY\_URI=785975698029.dkr.ecr.ap-south-1.amazonaws.com/ecs-package-repository  build:  commands:  - echo Build started on `date`  - echo Building the Docker image...  - docker build -t ecs-package-repository:latest -f dockerfile .  - docker tag ecs-package-repository:latest 785975698029.dkr.ecr.ap-south-1.amazonaws.com/ecs-package-repository:latest  post\_build:  commands:  - echo Build completed on `date`  - echo Pushing the Docker image to ECR...  - docker push 785975698029.dkr.ecr.ap-south-1.amazonaws.com/ecs-package-repository:latest  artifacts:  files:  - imagedefinitions.json  - appspec.yml |
| --- |

#### 

#### **appspec.yml**

| version: 0.0  Resources:  - TargetService:  Type: AWS::ECS::Service  Properties:  TaskDefinition: <TASK\_DEFINITION>  LoadBalancerInfo:  ContainerName: "sample-app"  ContainerPort: 2020 |
| --- |

### 

### **4. Deployment**

**Use AWS CodePipeline to automate the build and deployment process. The pipeline will:**

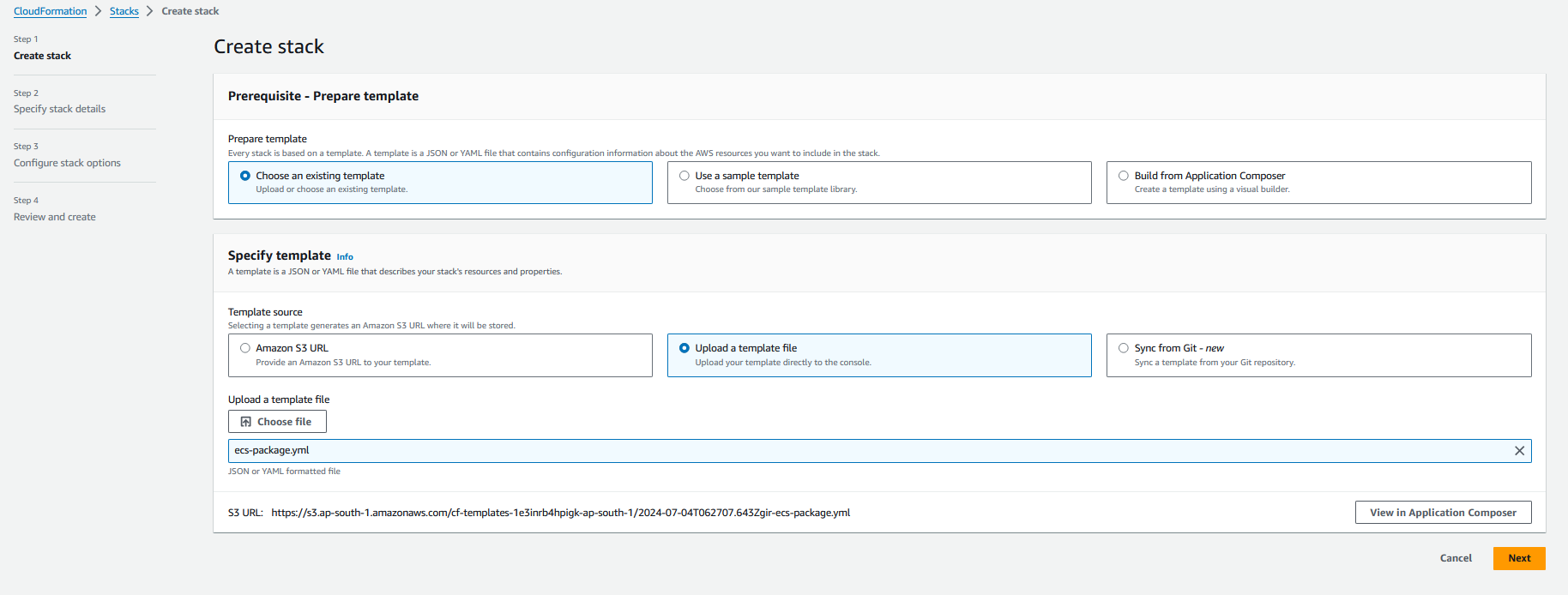
* Fetch the source code from GitHub
* Build the Docker image using CodeBuild
* Push the image to Amazon ECR
* Update the ECS service with the new image

### **5. Logging**

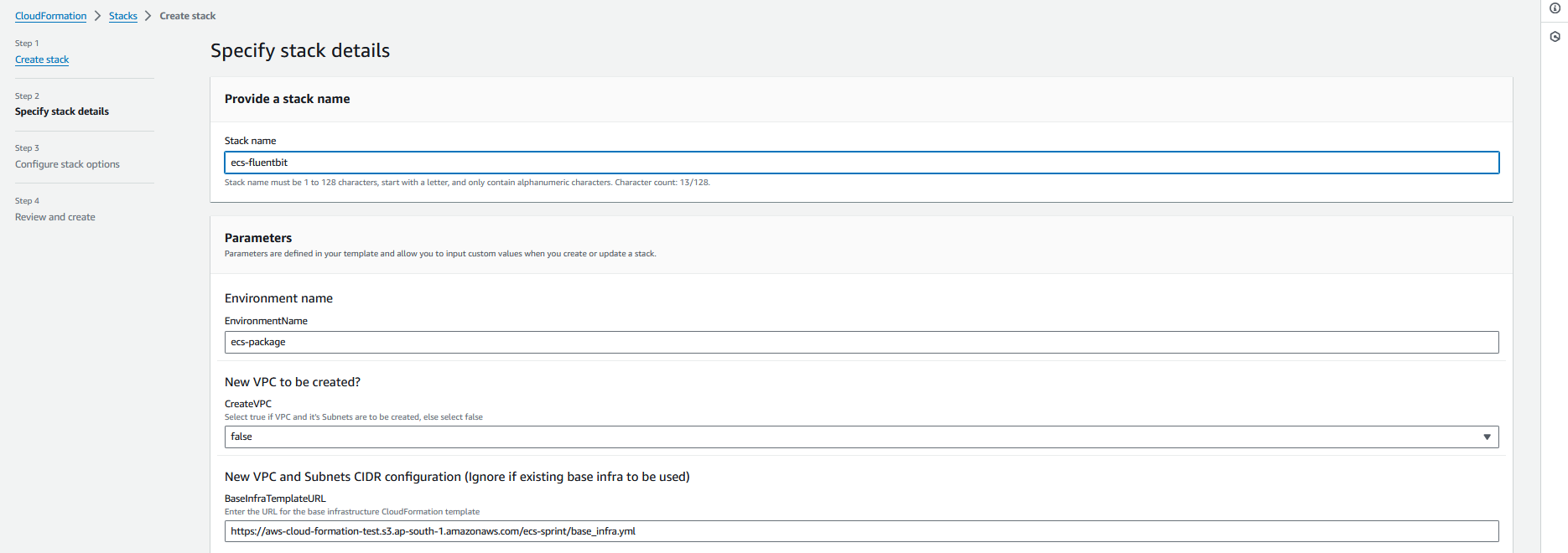
**Fluent Bit will be configured to:**

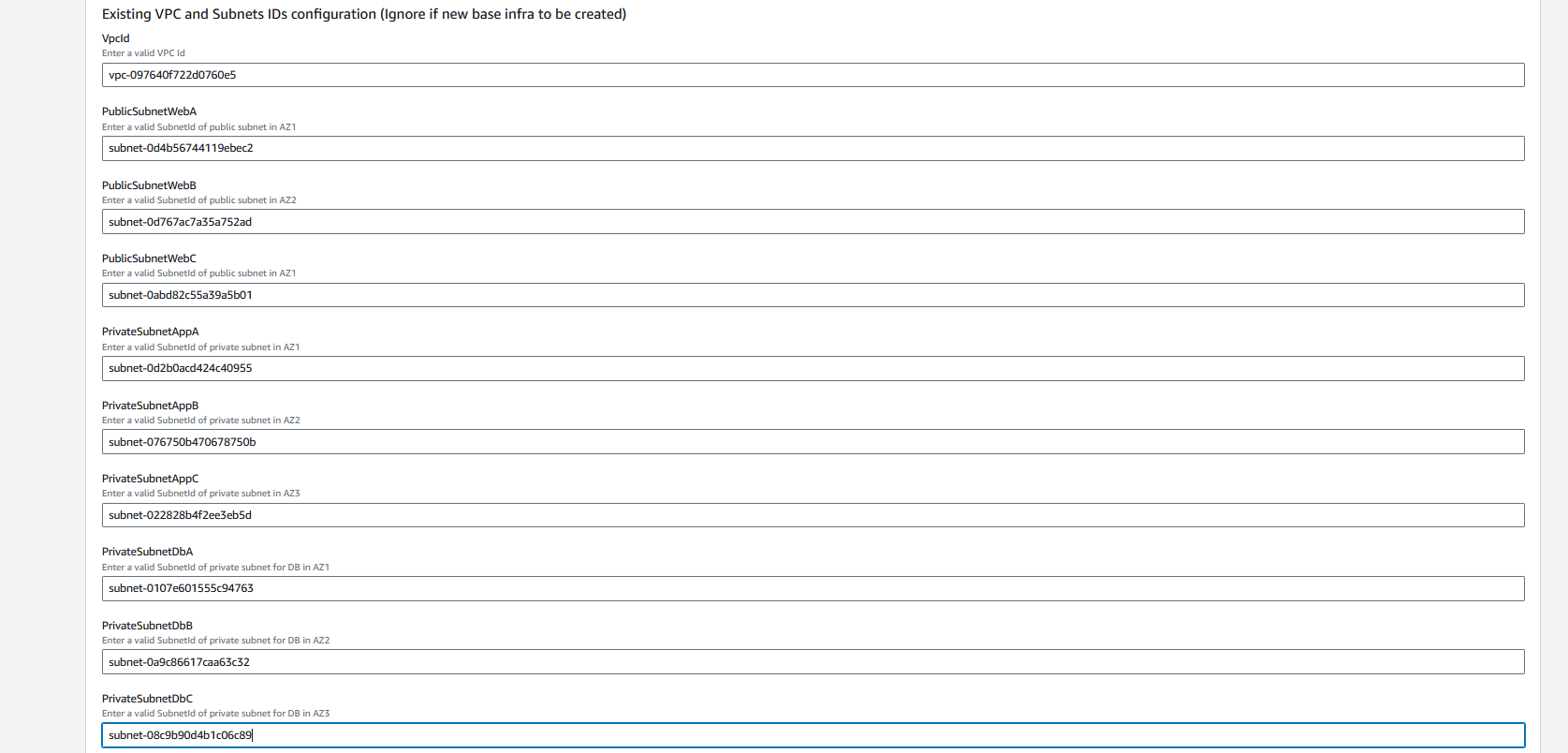
* Parse logs from /var/log/secure using the nginx parser defined in parsers.conf
* Forward logs to Amazon OpenSearch Service for real-time analysis
* Backup logs to Amazon S3 for long-term storage

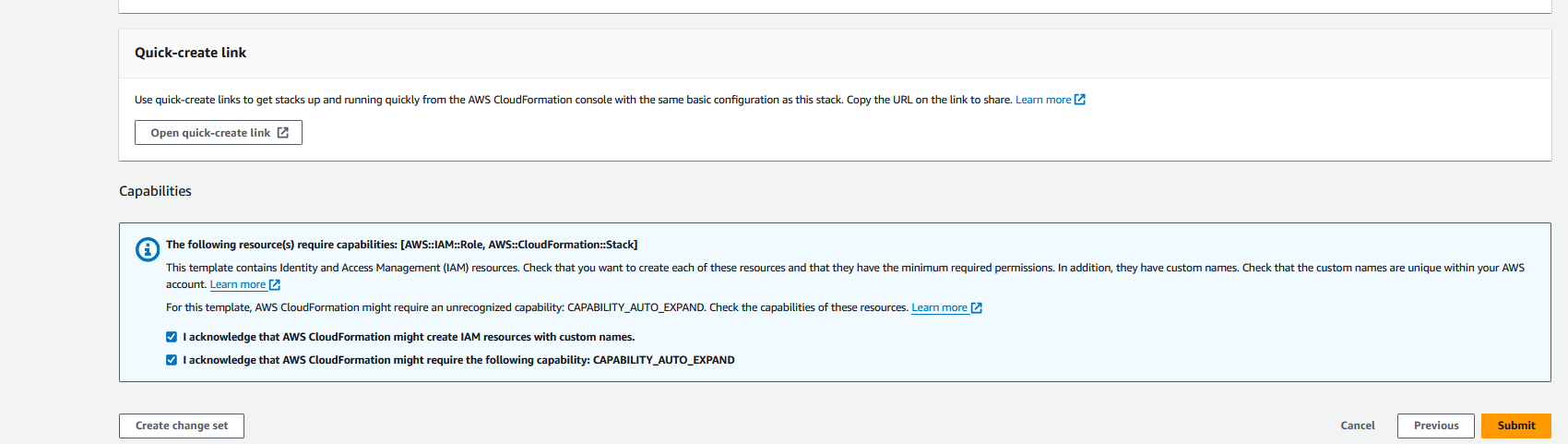
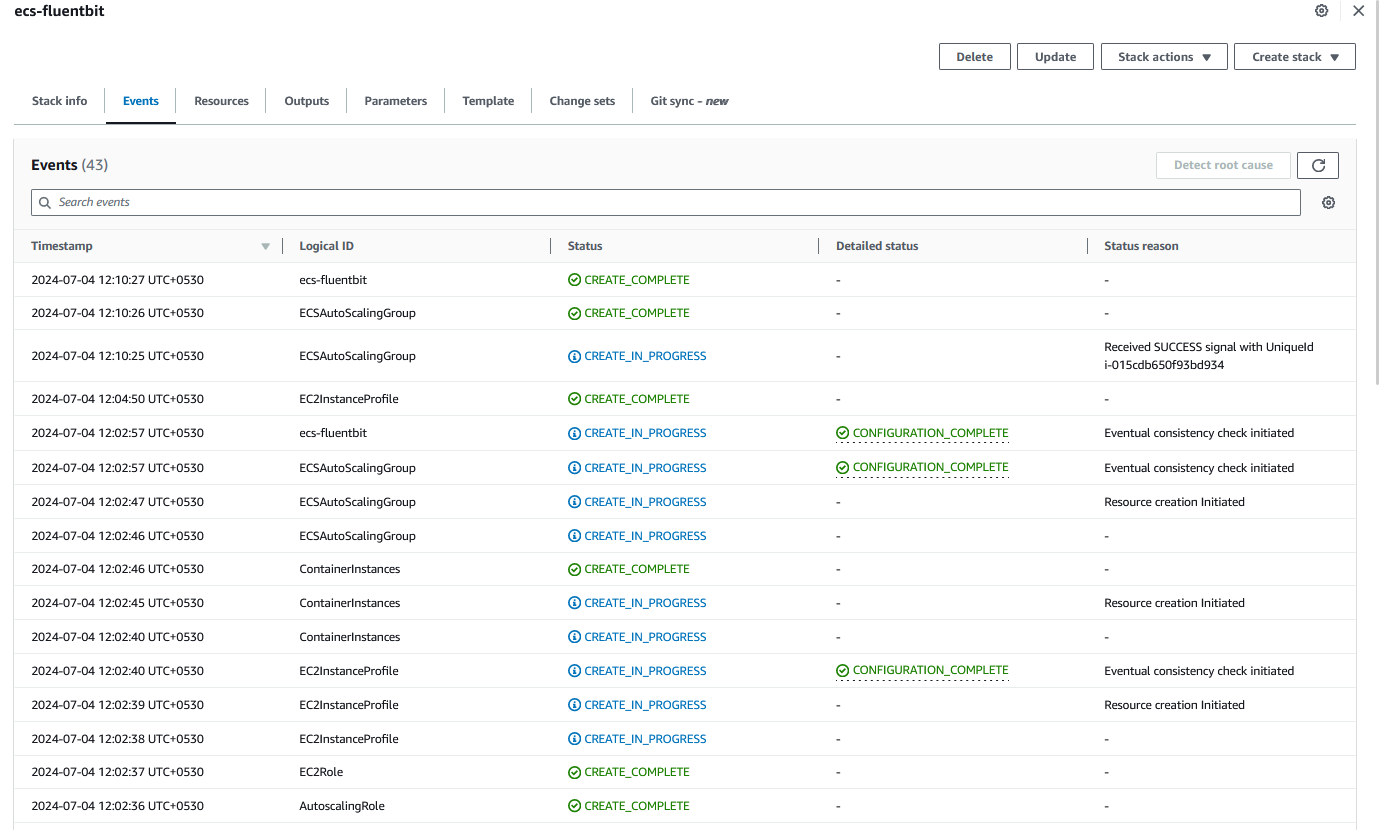
**CloudFormation Steps:** 1) Upload the ecs-package.yml file

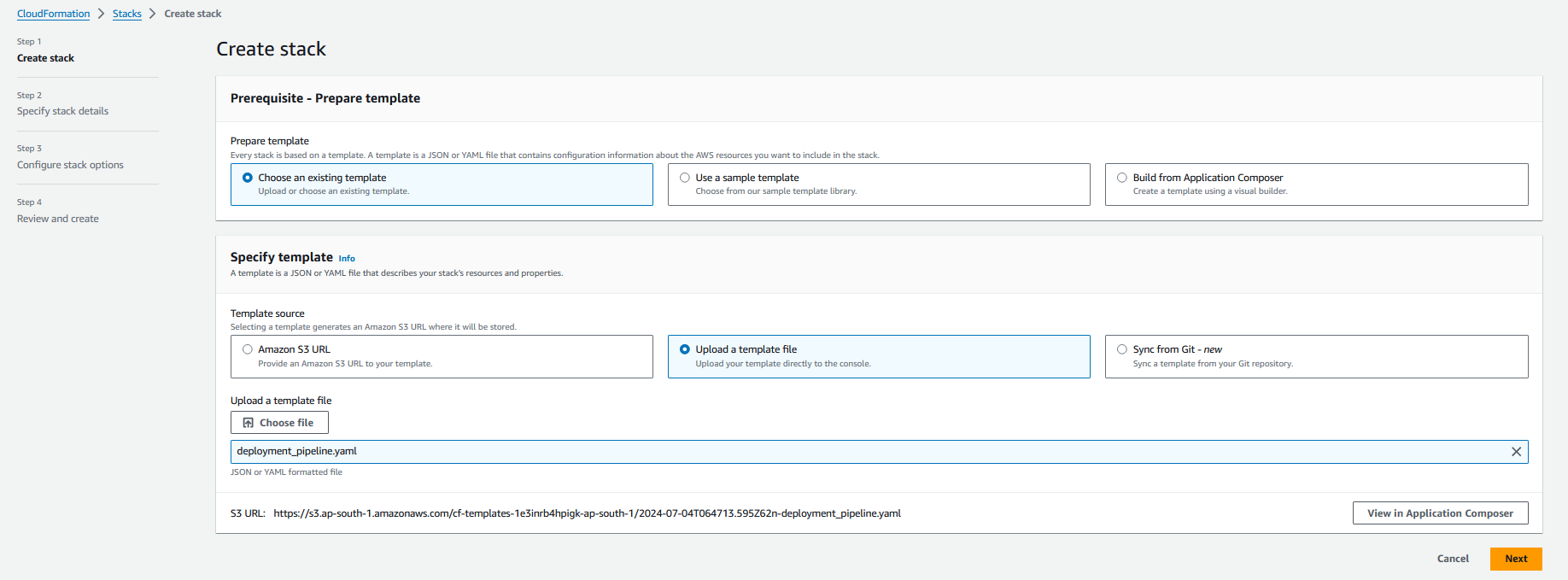
****

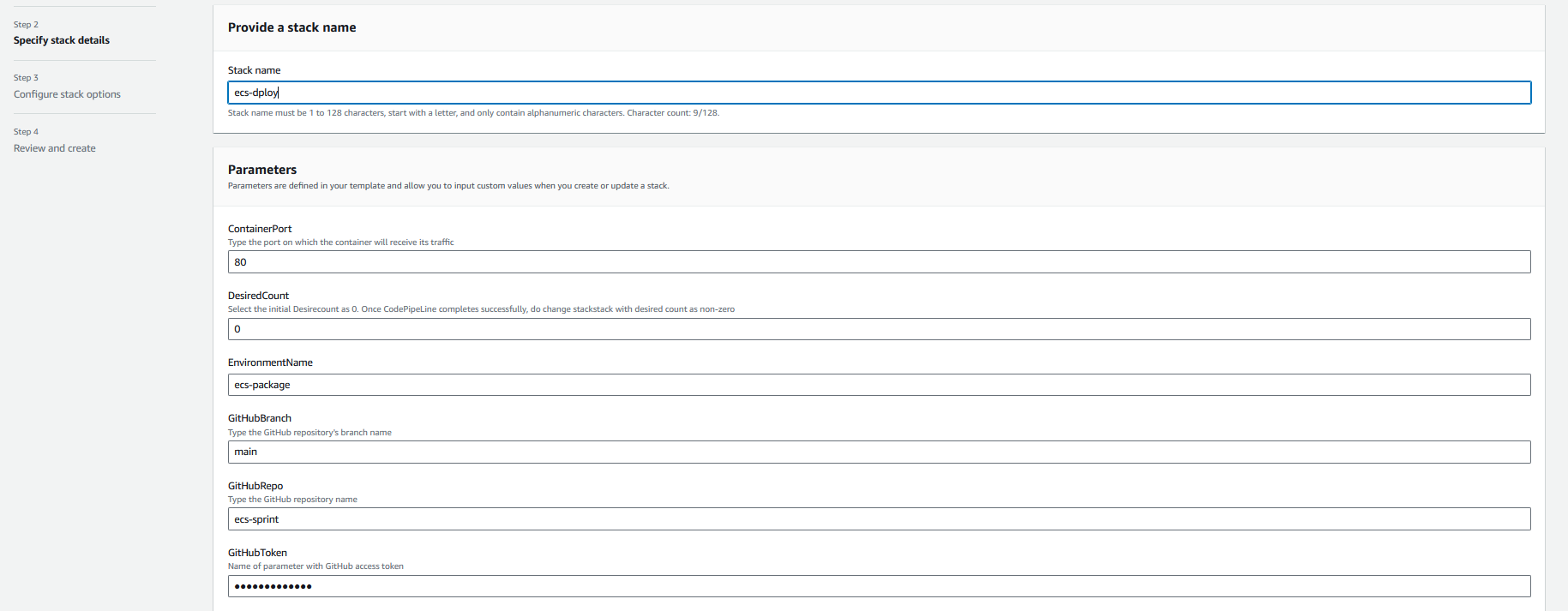
2) Fill up the required details accordingly

****

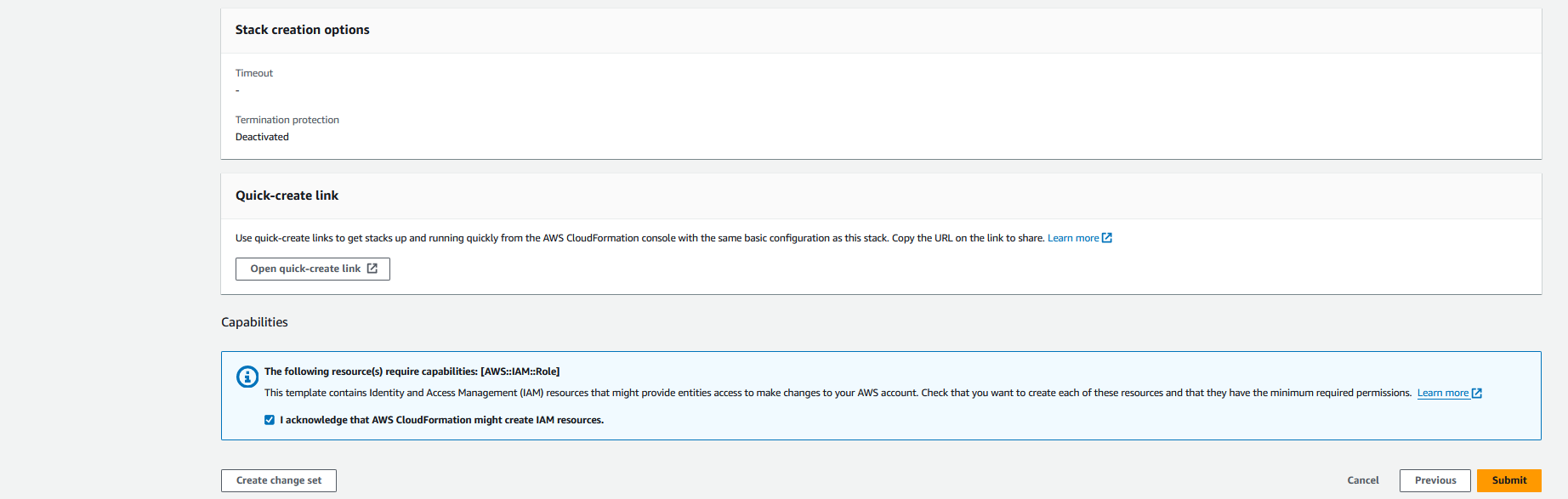
****

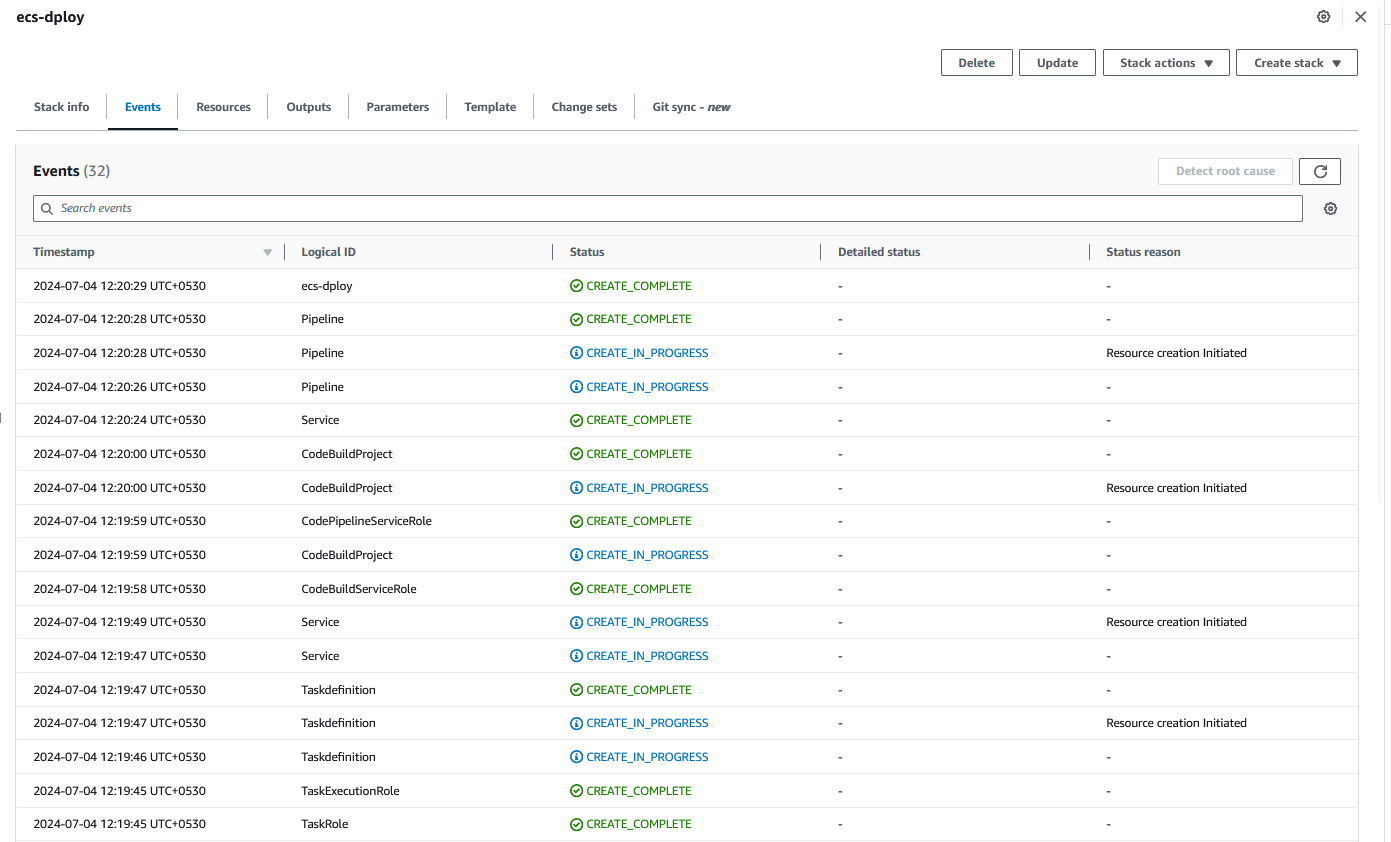
3) Select the check box and click on submit **  
  
**

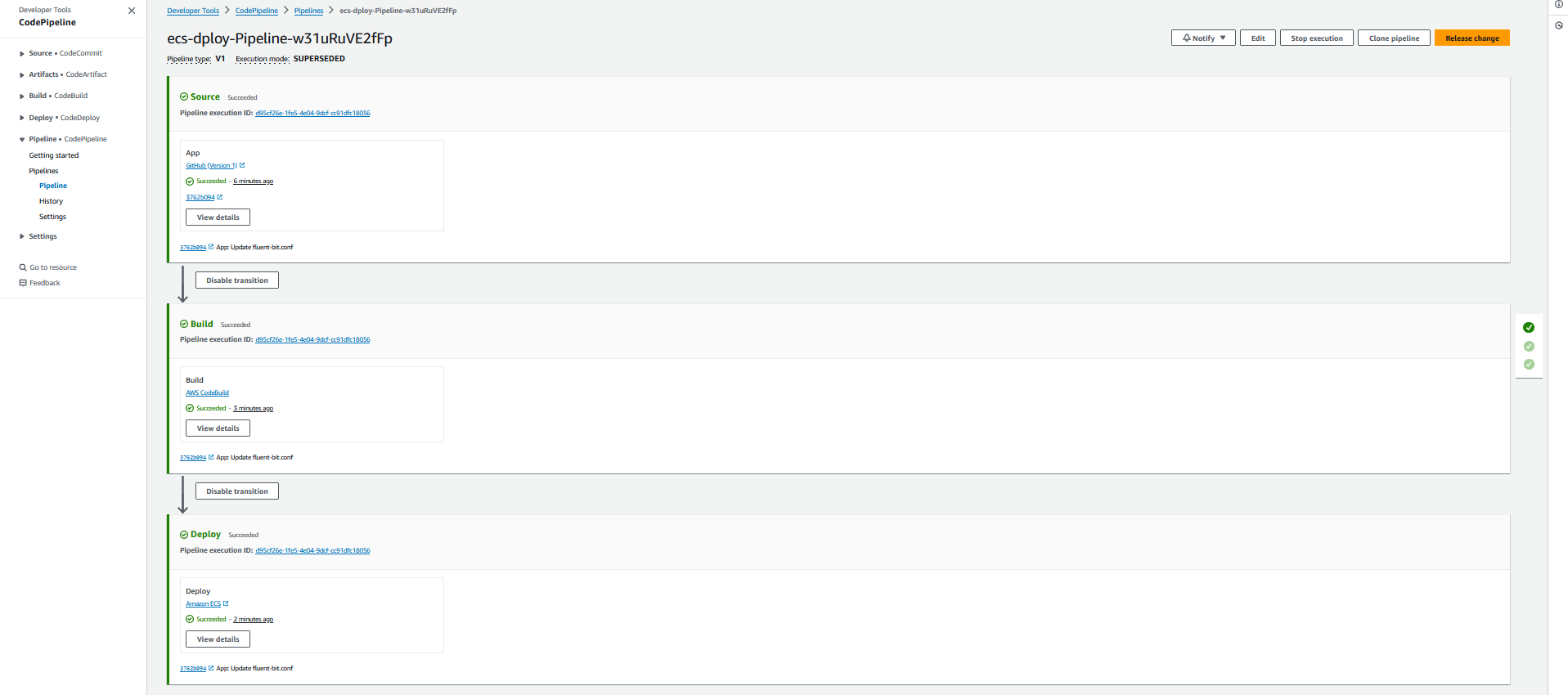
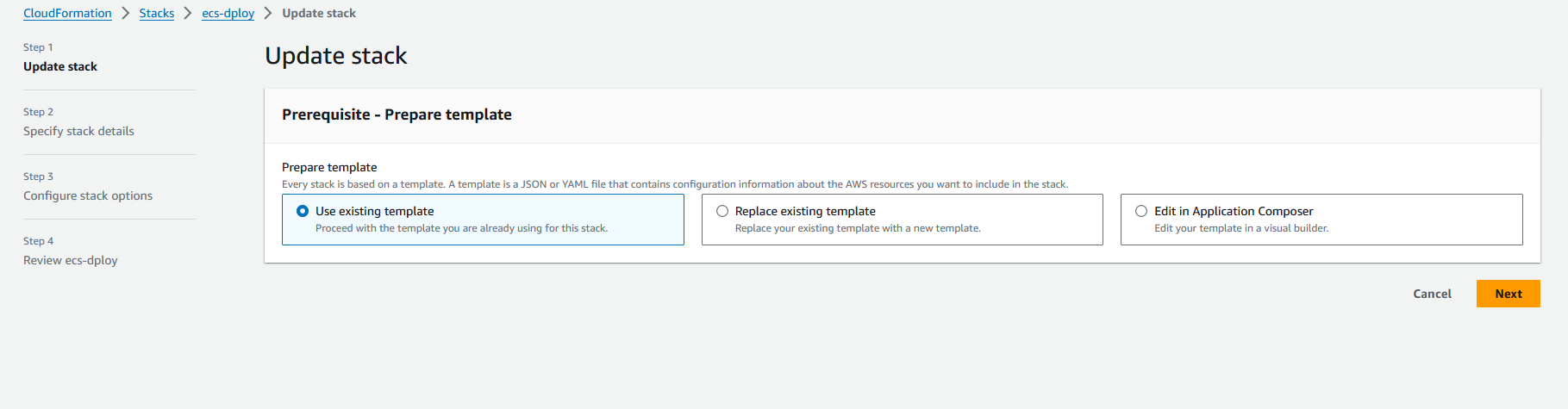
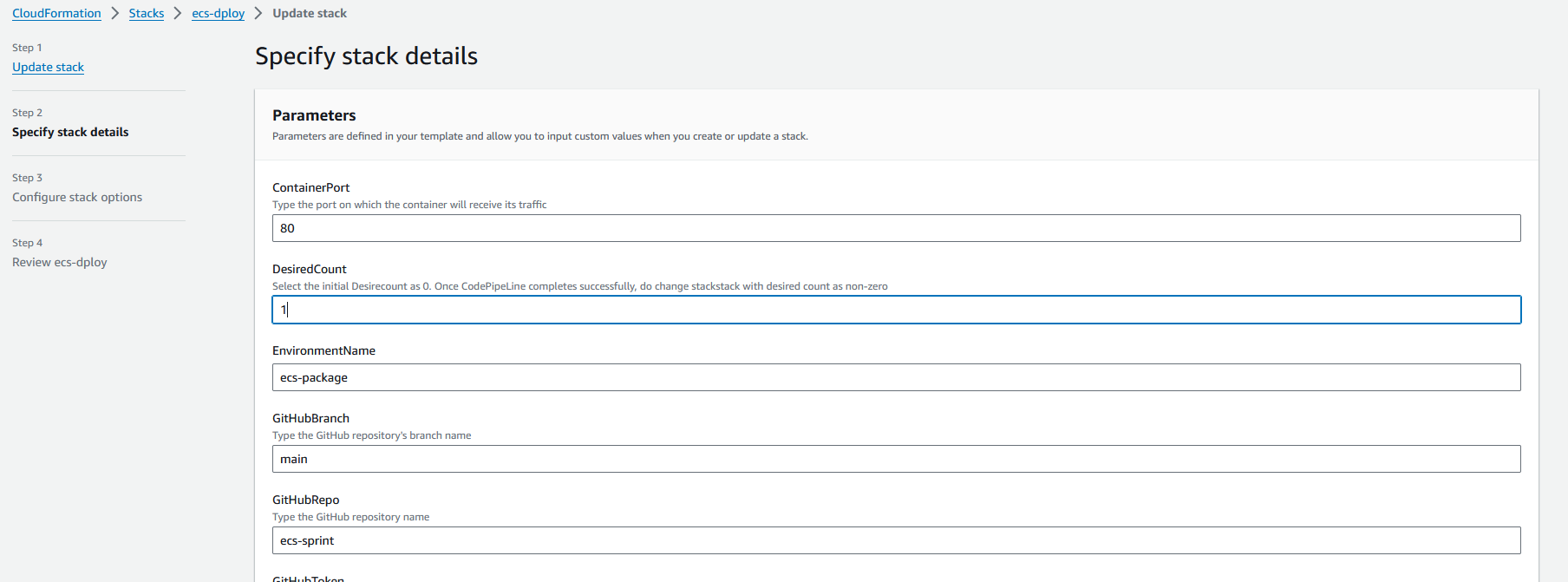
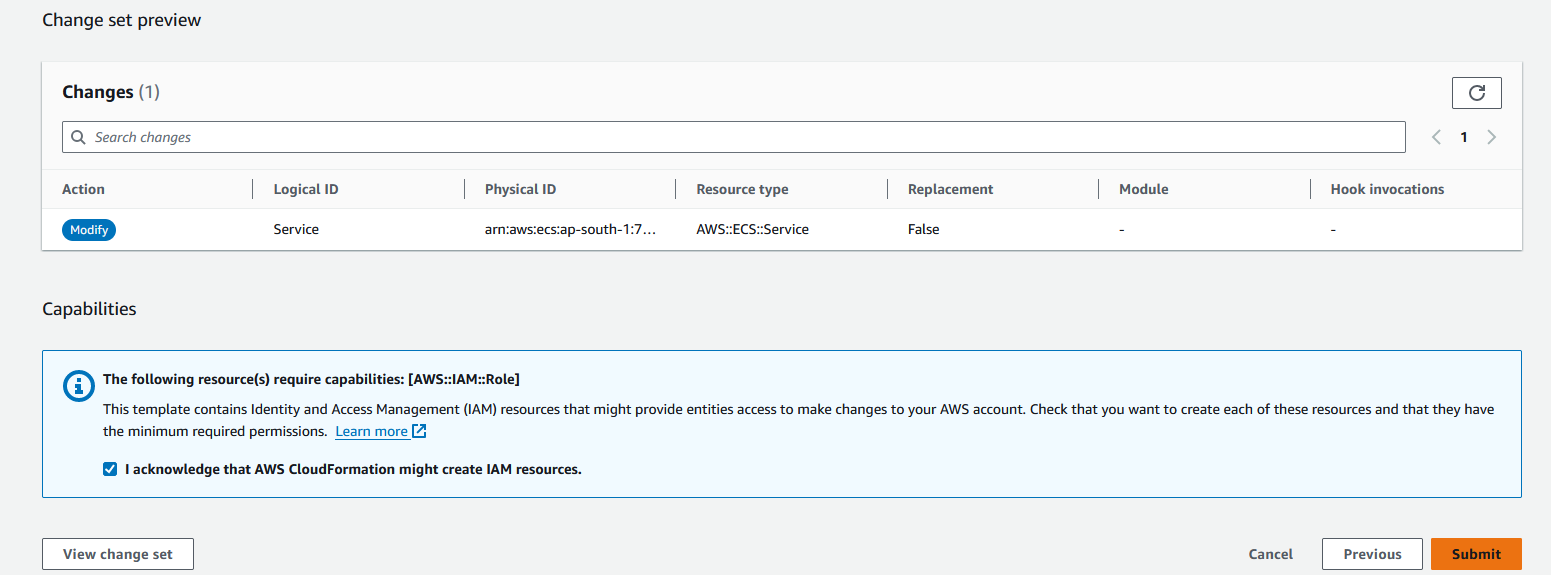
4) Now upload the deployment\_pipeline.yaml file ****

****

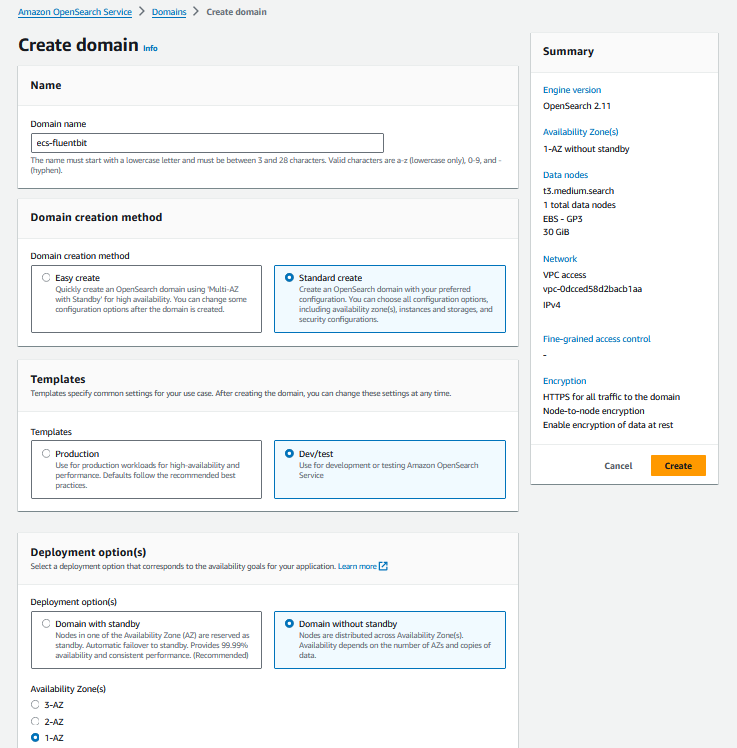
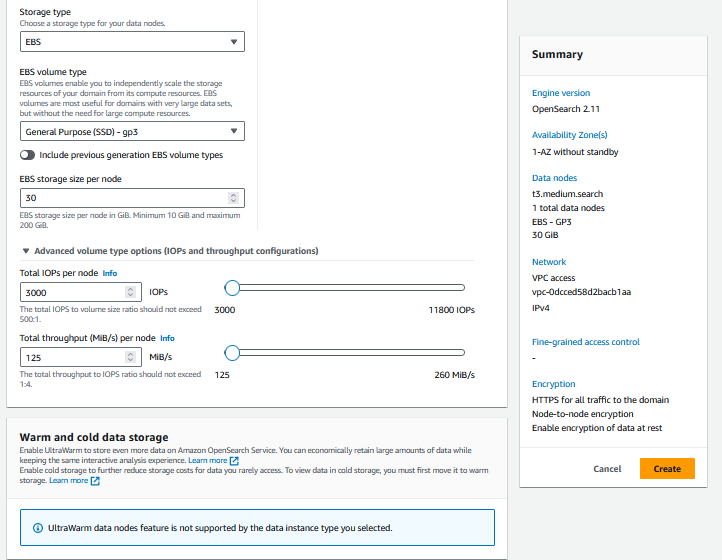
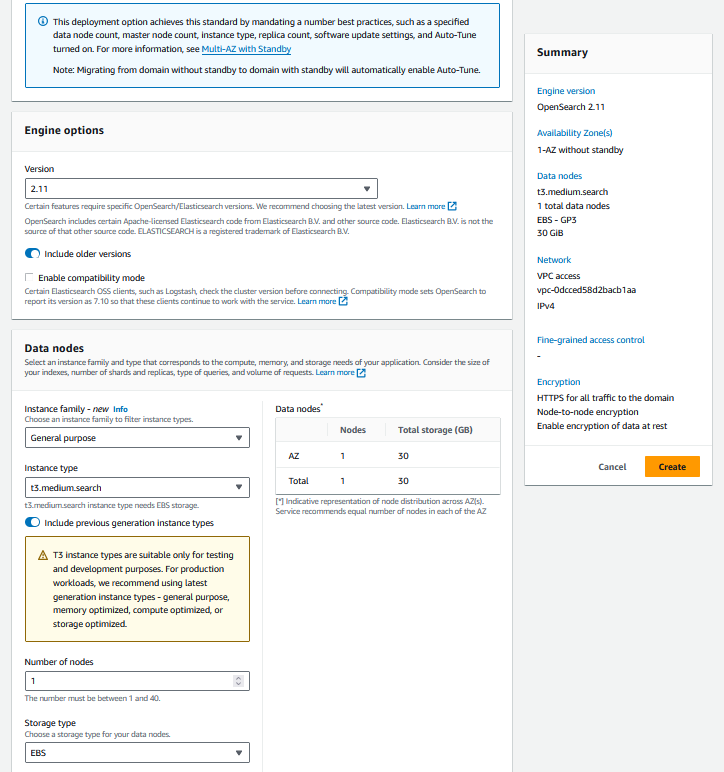
5) Fill up details accordingly & click on submit

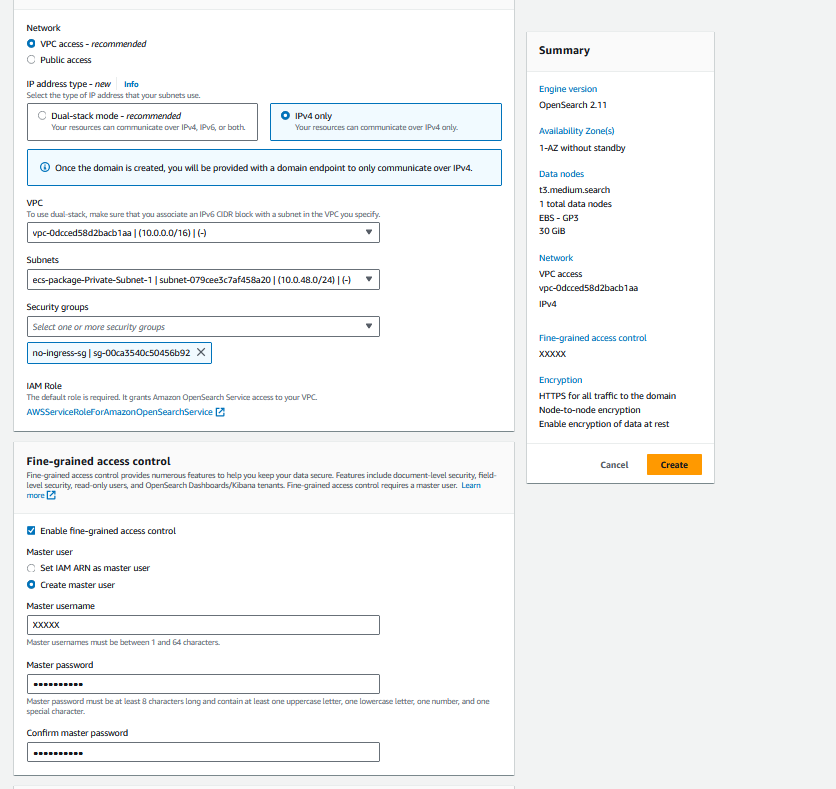
****

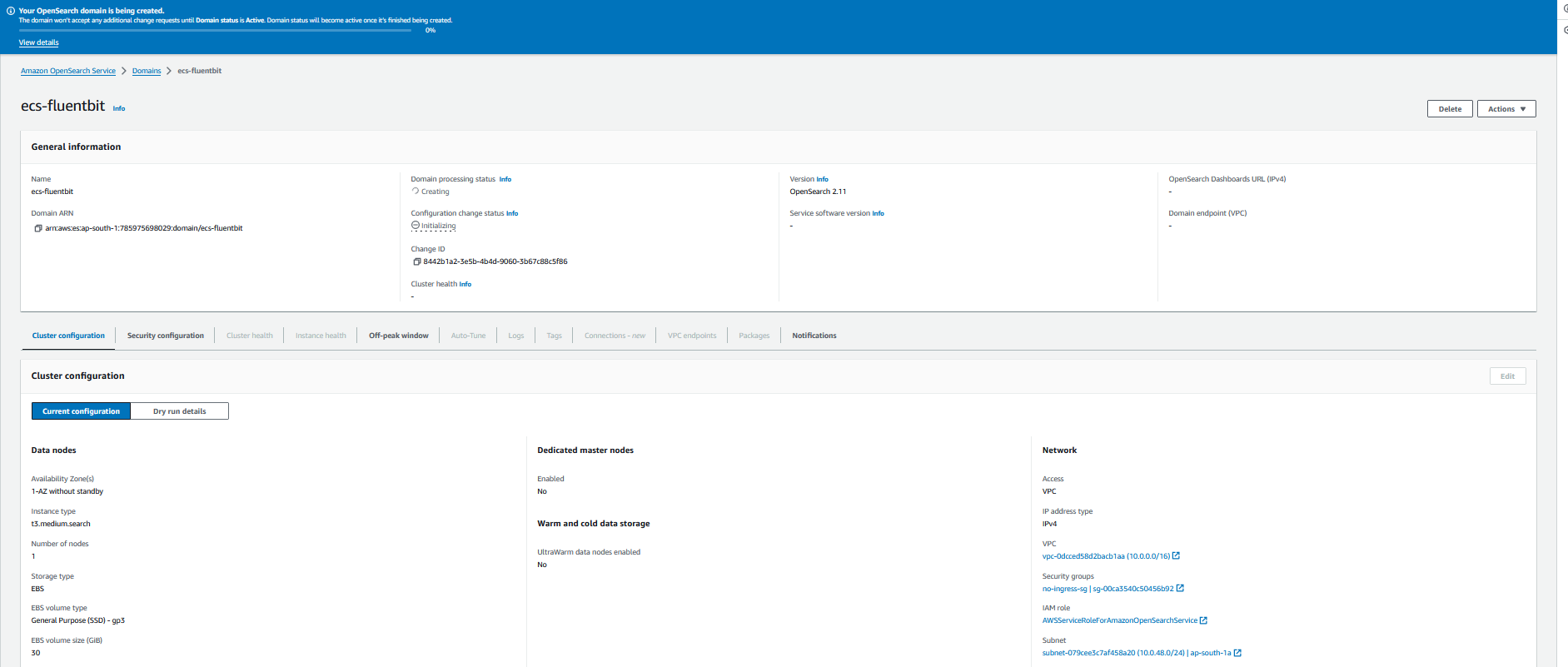
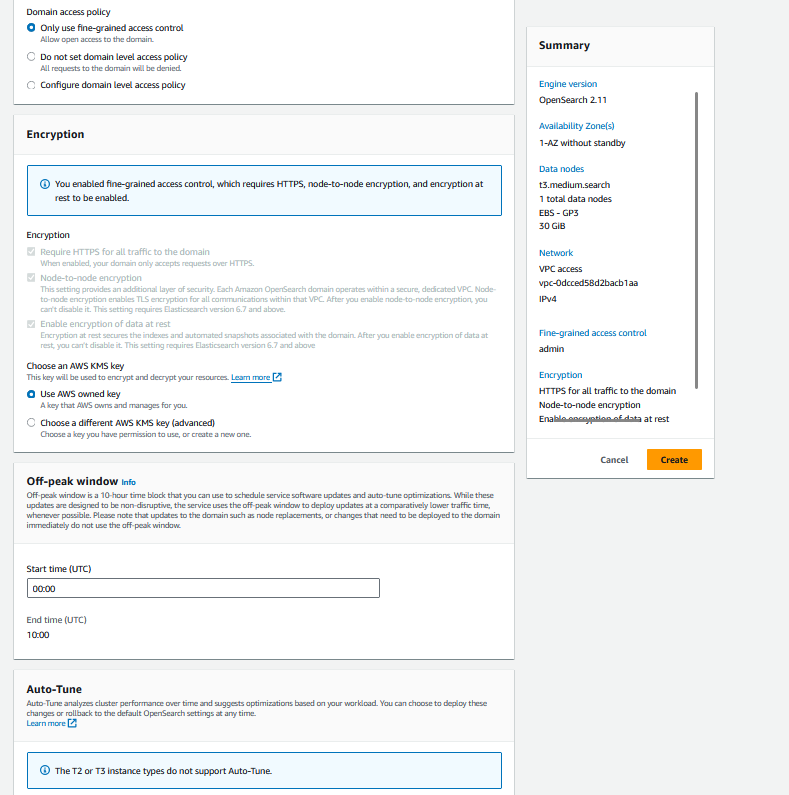
****

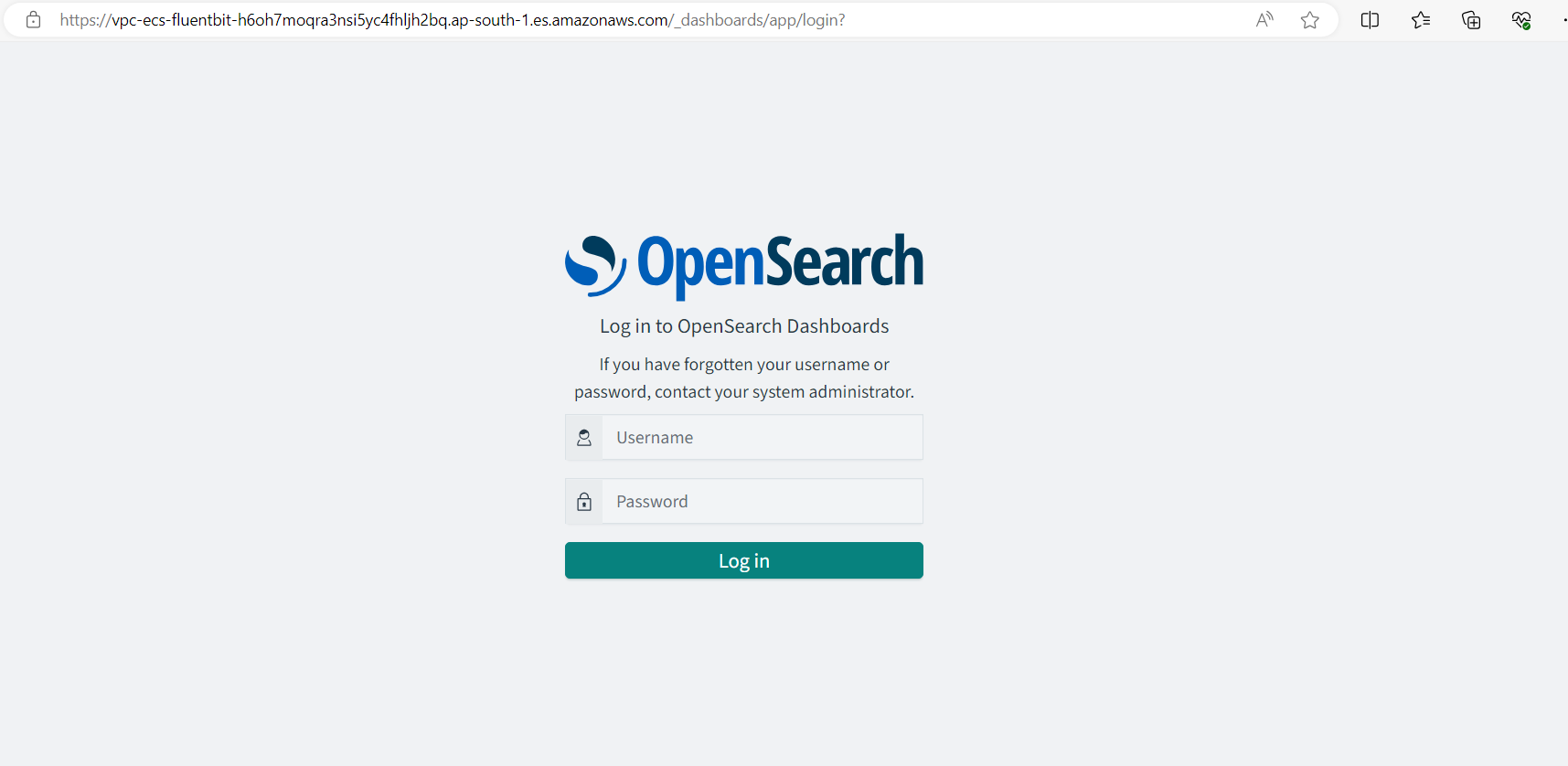
6) Check the codepipeline  **  
  
**7) Now update the Desire count as no of container in deployment yaml file & submit **  
**

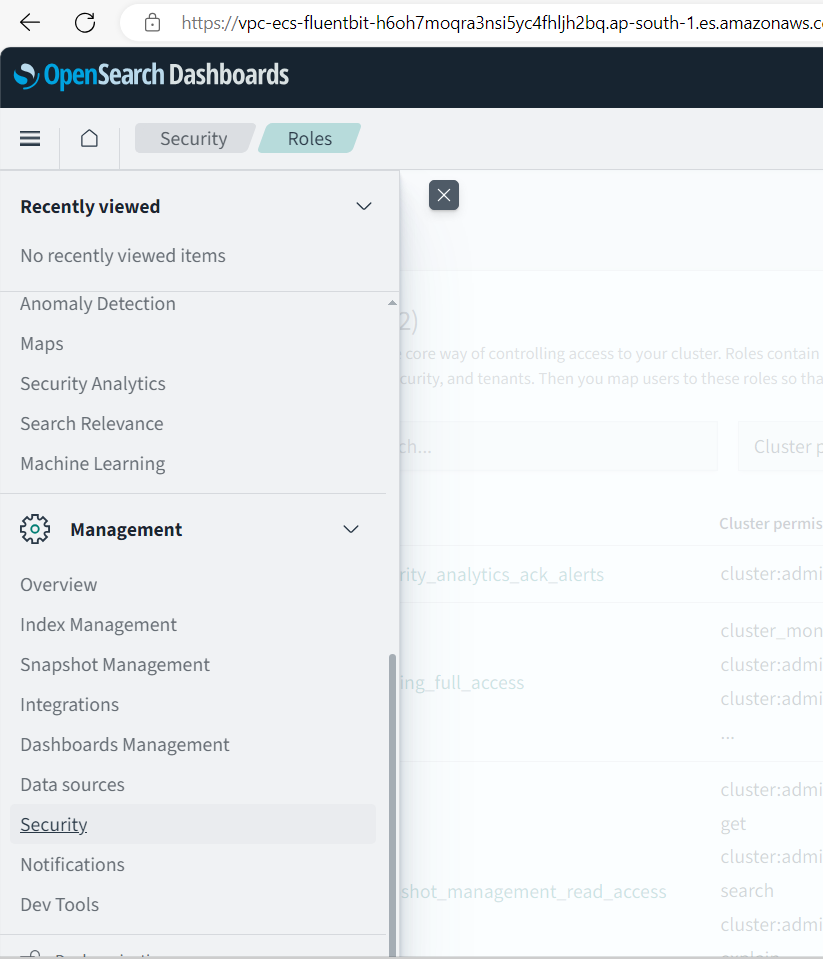
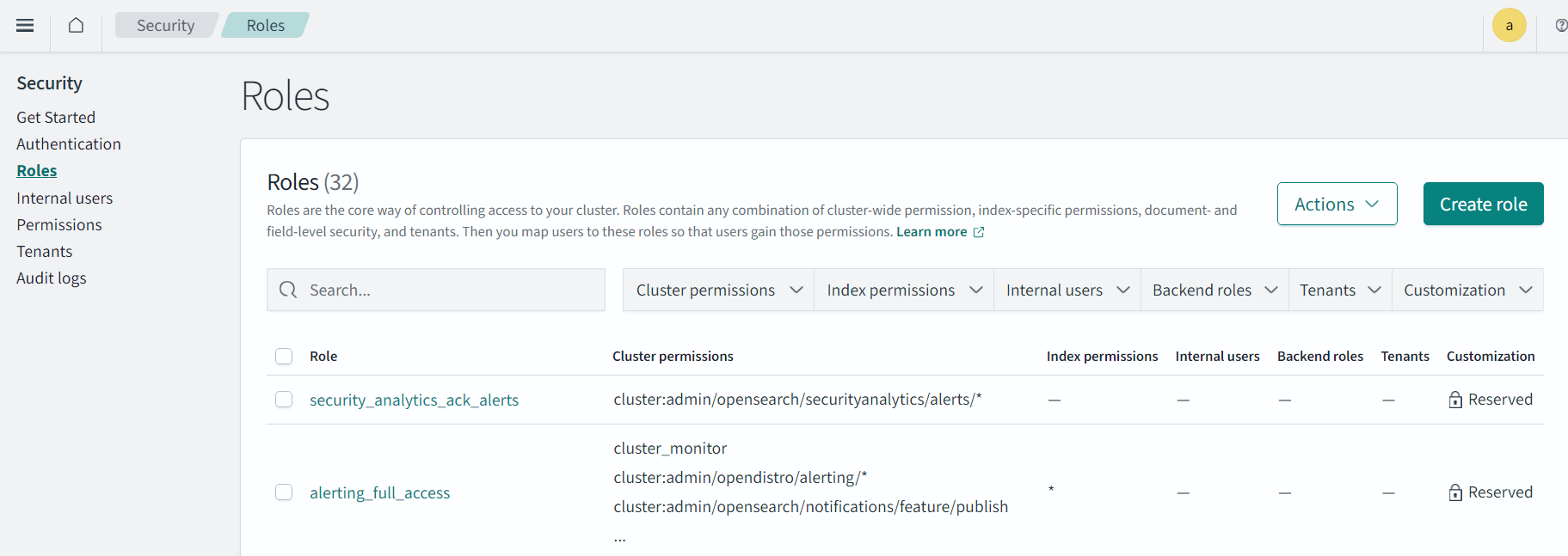
**Opensearch Creation:** 1) Fill up the details accordingly as per the reuirement

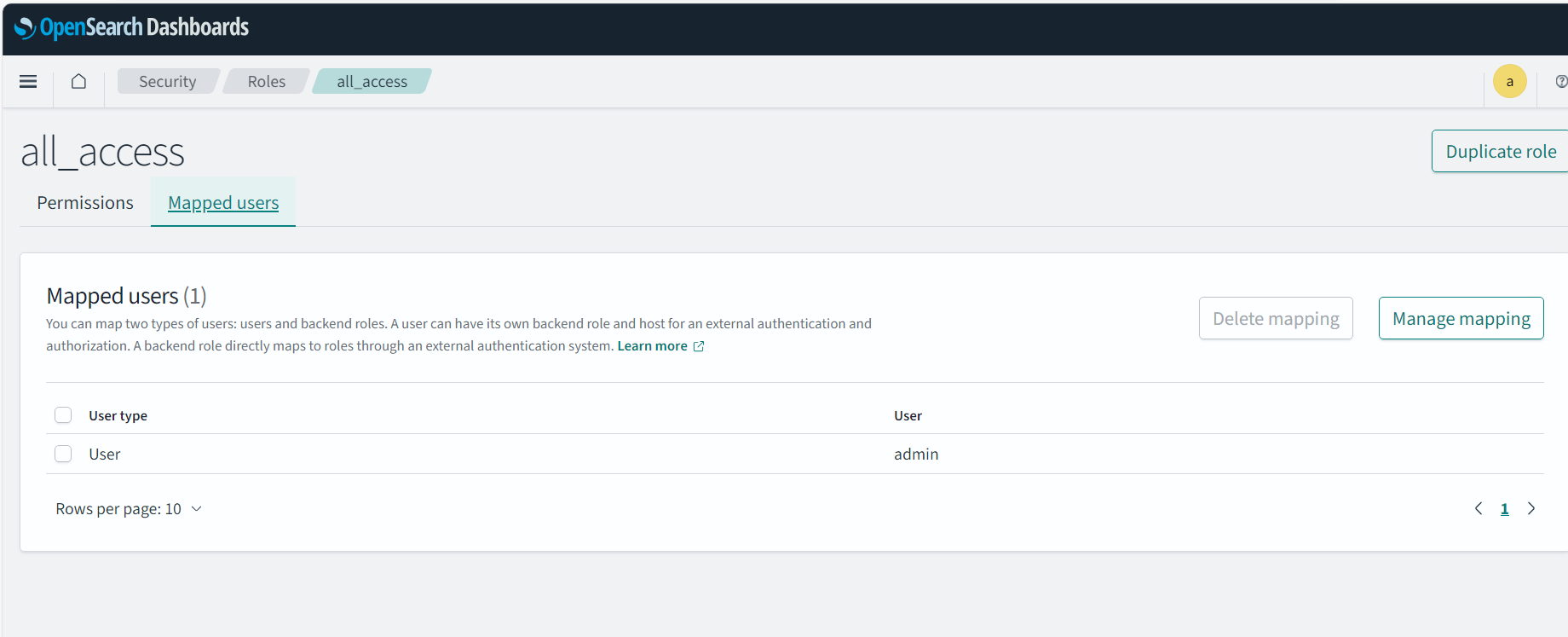
****2) fill the details like version , instance type , ebs storage , select the vpc & create****

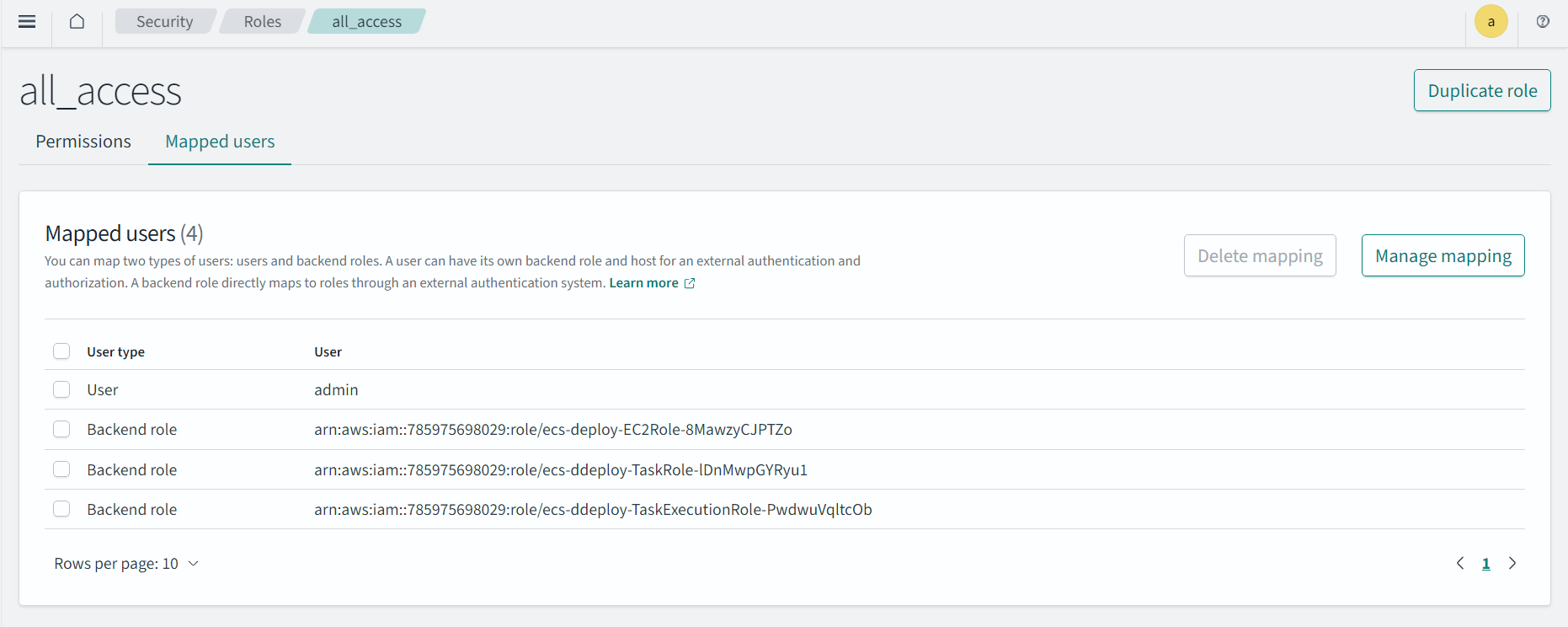
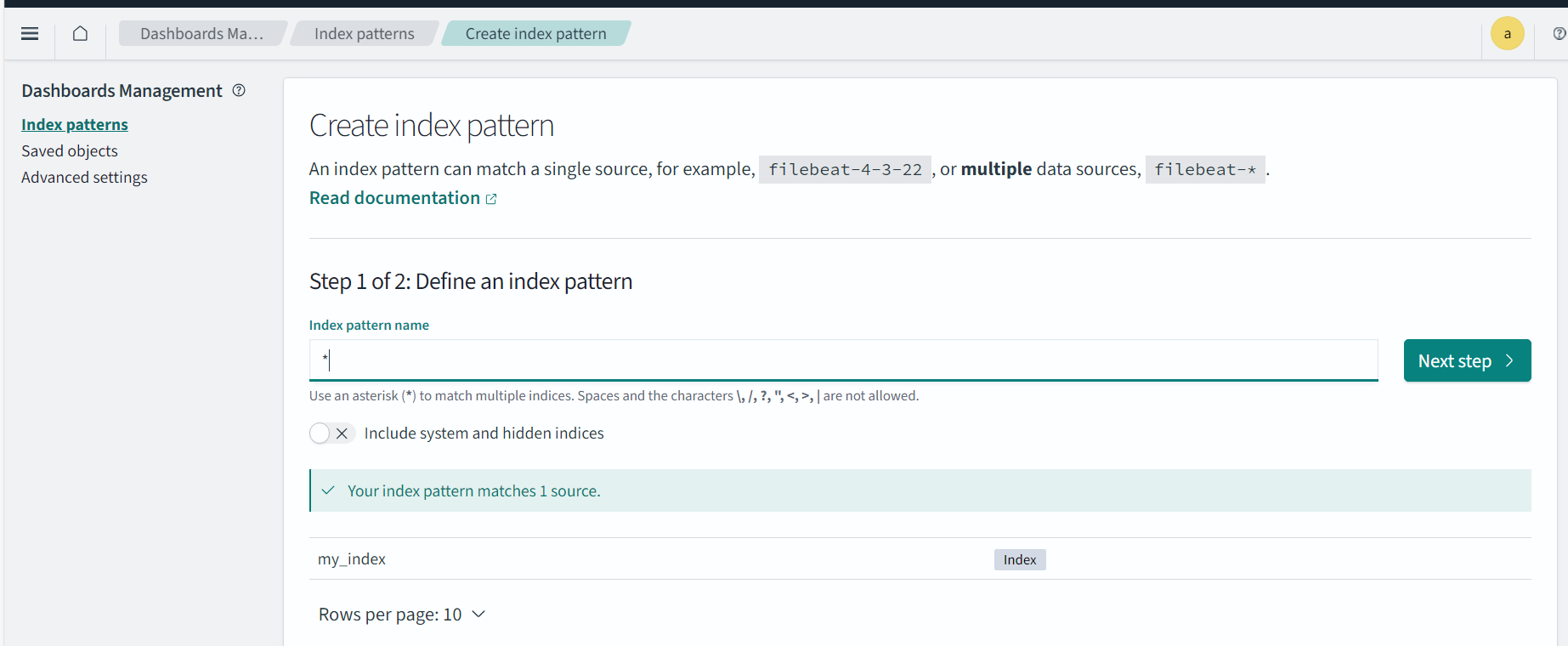
****

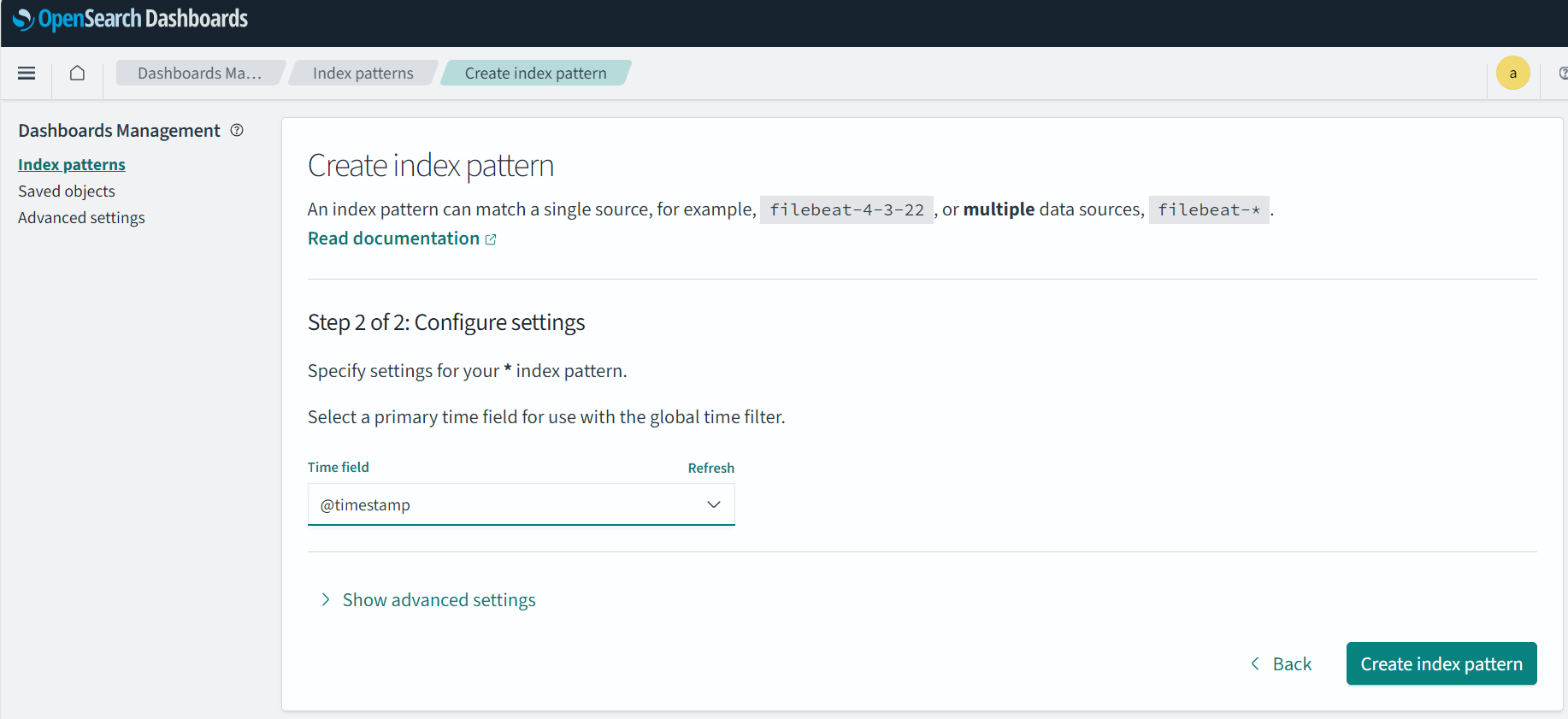
****

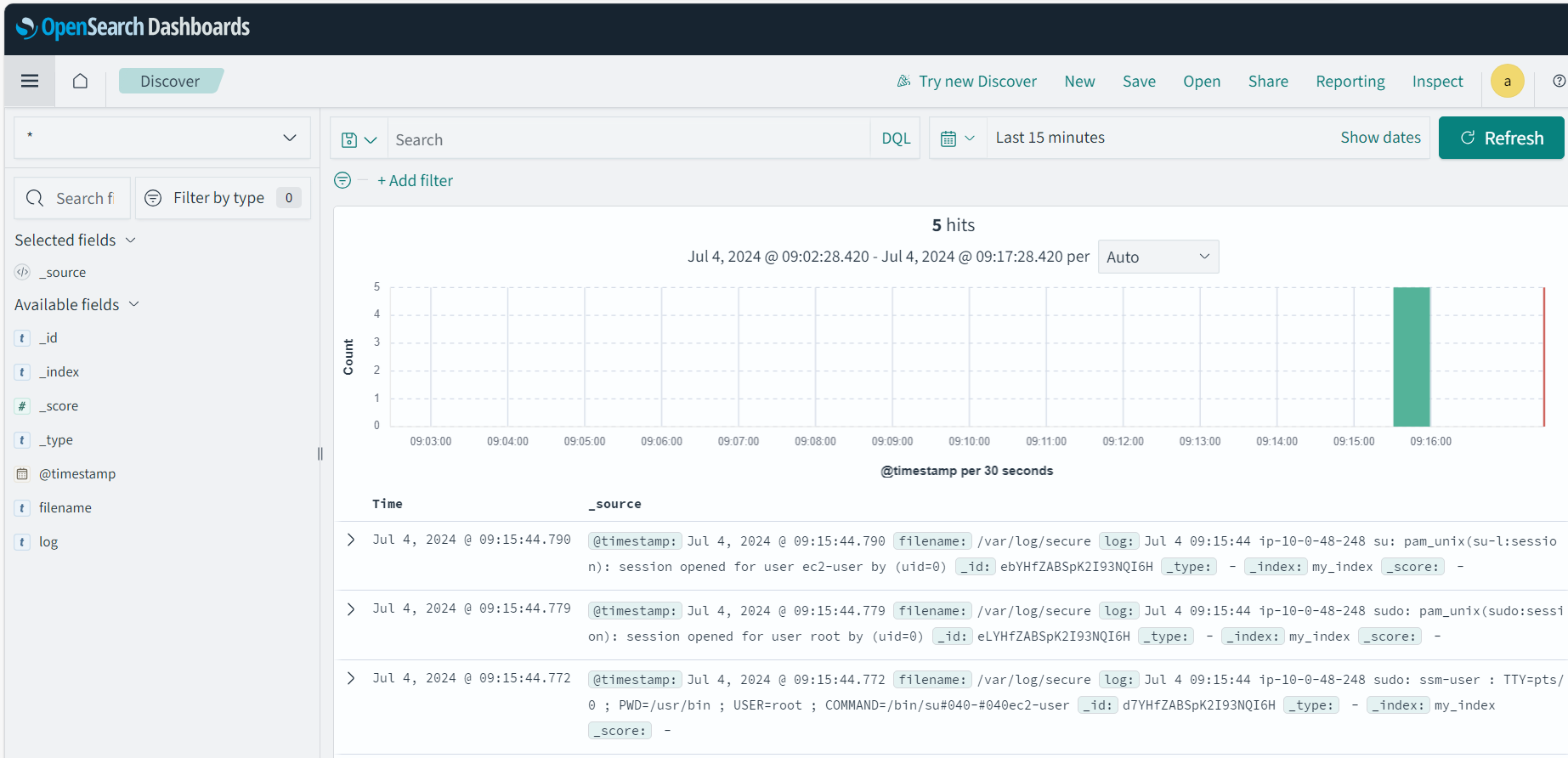
3) wait till the status turns into Active ****

4) Use the domain and enter the same username & password , click on left top 3 lines & select **security   
  
  
  
**5) click on roles and mapped roles - task & task execution

****

**  
  
**6) Enter \* and click on next step , select timestamp in dropdown

****

****- Click on Discover will direct to this page of logs

## 

## **Benefits**

* **Automated Deployment:** Simplifies the deployment process with CloudFormation and CodePipeline**.**
* **Centralized Logging:** Provides a robust logging solution with Fluent Bit, enabling real-time log analysis and long-term storage**.**
* **Scalability:** Easily scalable infrastructure with ECS and VPC configuration**.**
* **Security:** Ensures secure communication between services and log destinations using private subnets and security groups**.**

## **Conclusion**

By implementing this ECS CloudFormation solution with Fluent Bit for logging, you achieve a highly automated and scalable deployment pipeline. This setup ensures efficient log management and enhances the monitoring and debugging capabilities of your containerized applications**.**