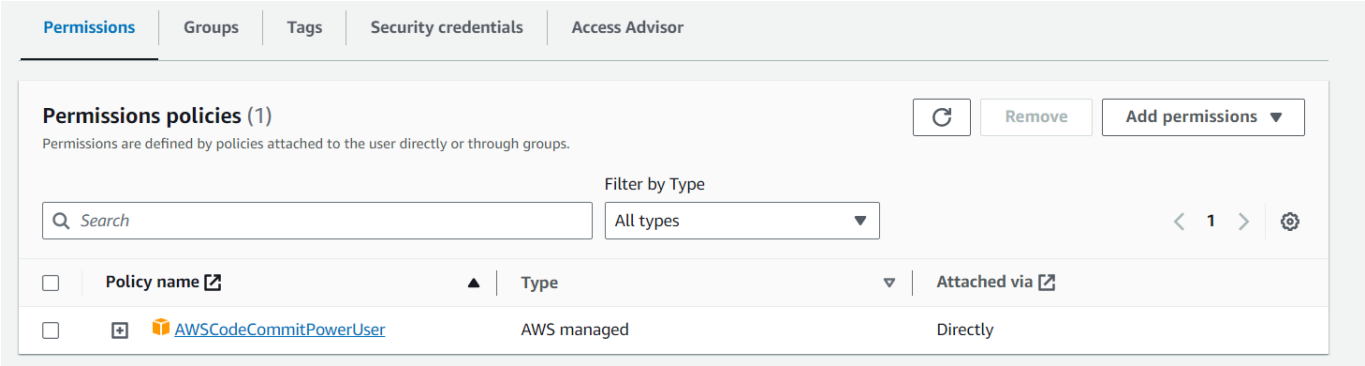
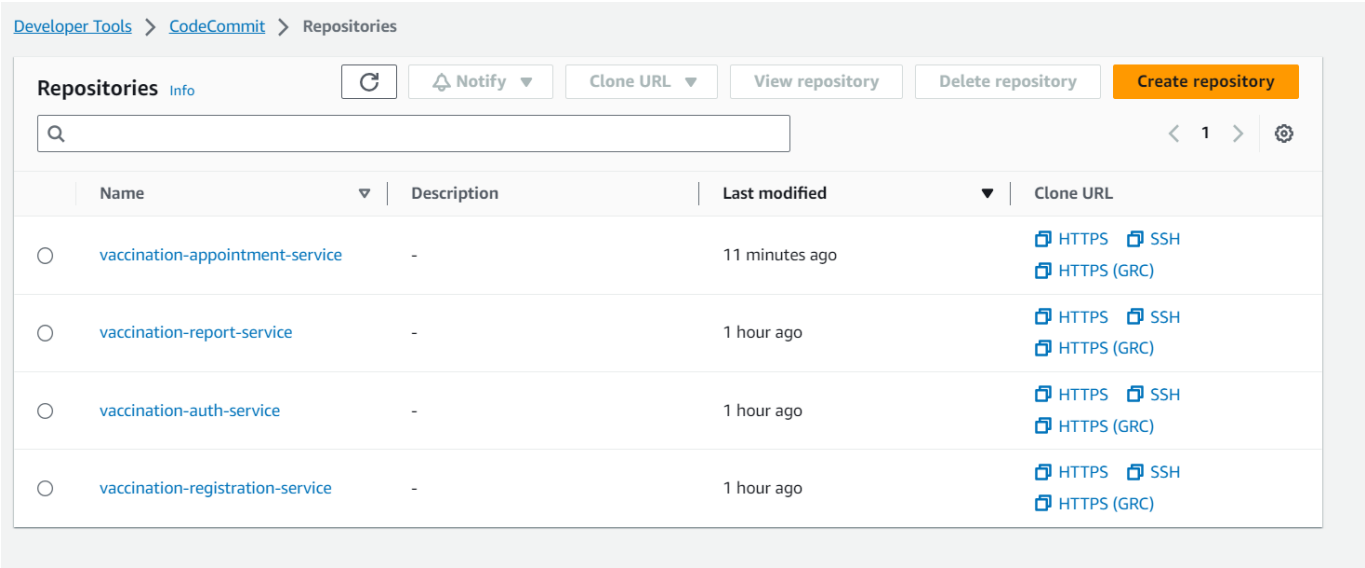
CodeCommit Configuration

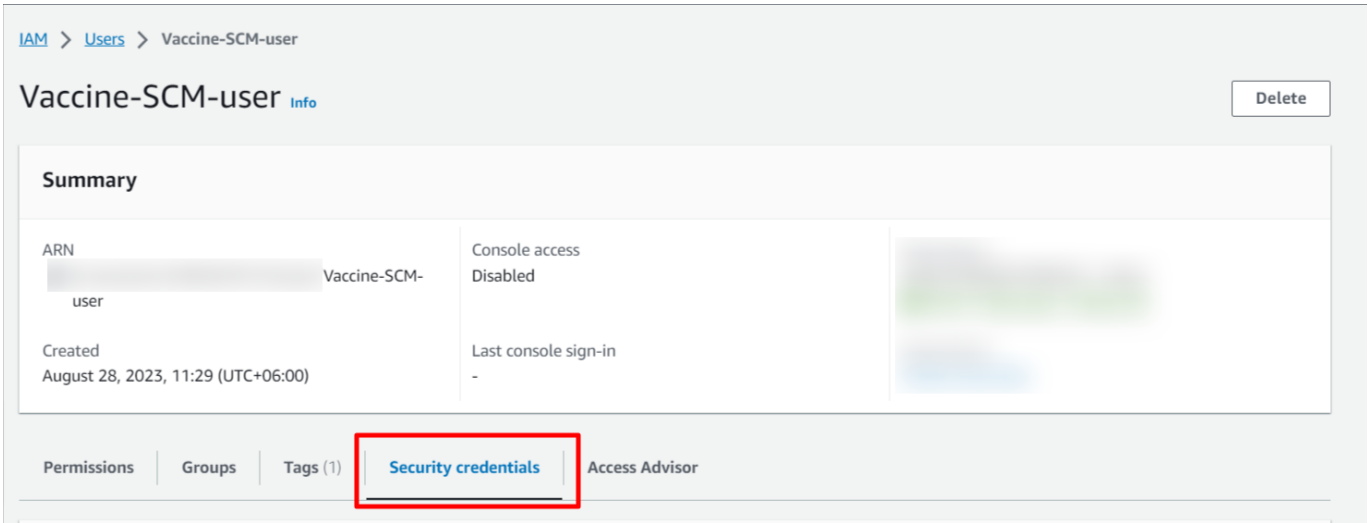
Step 1: Create an IAM User with AWSCodeCommitPowerUser policy.

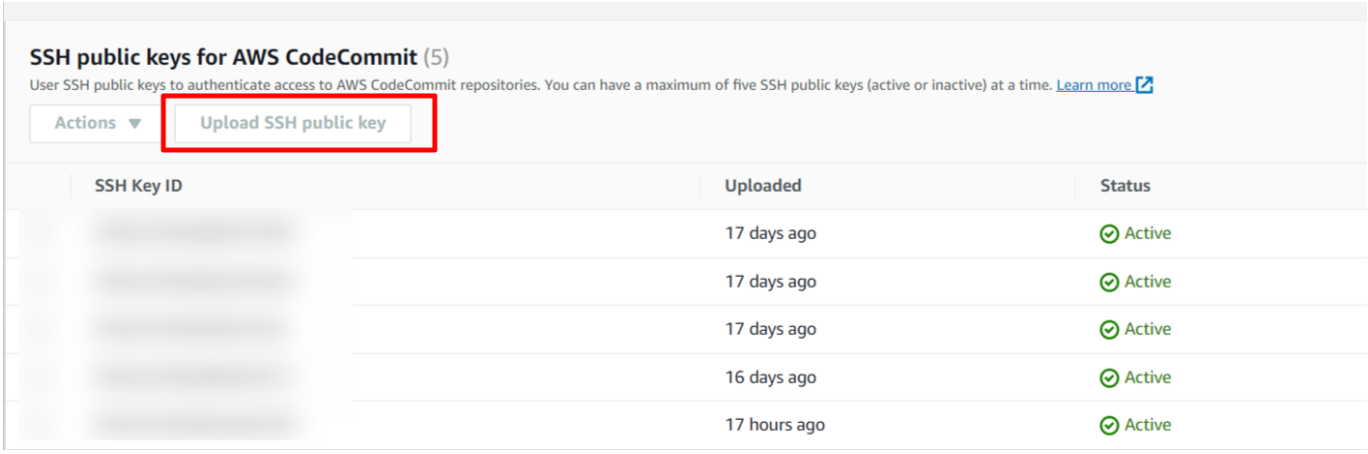


Step 2: Create Repositories

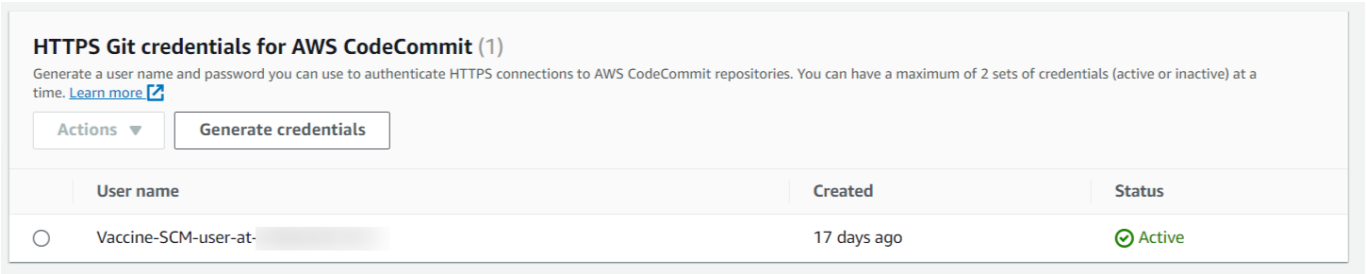


Step 3: Add your SSH keys to the newly created user in Step 1 security credentials. Up to 5 SSH can be added per IAM user.





Step 4: Again under Security Credentials for HTTPS access to your repositories you need to generate git credentials for your account.

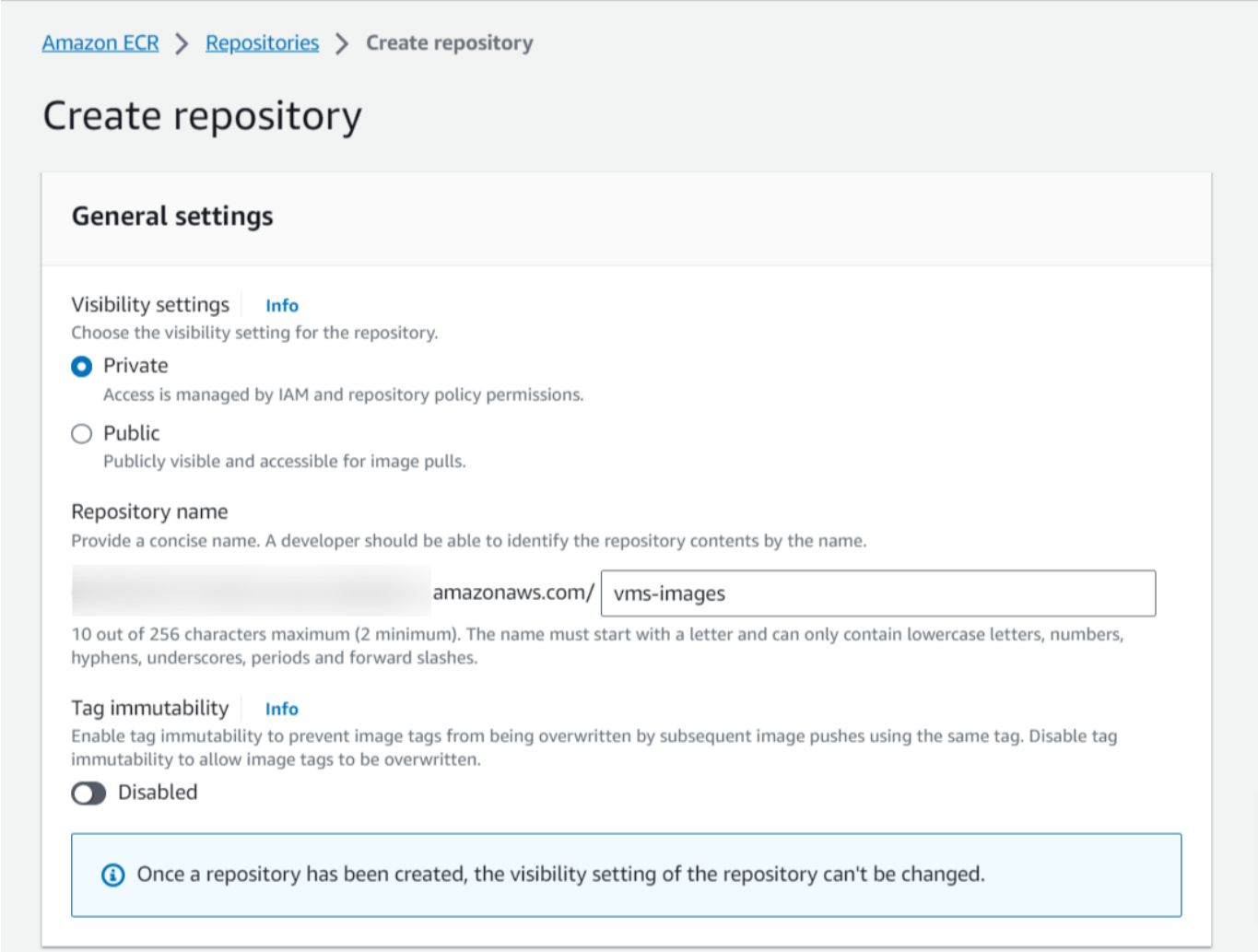


Step 5: Copy the username and password that IAM generated for you, either by showing, copying, and then pasting this information into a secure file on your local computer, or by choosing Download credentials to download this information as a .CSV file. You need this information to connect to CodeCommit.

Step 6: Check your connection by cloning one of the repositories.

ECR(Elastic Container Registry) Setup

Step 1: Go over to ECR and create a private repository with a name of your choosing.



Step 2: Next, go to Permissions>Edit JSON Policy and delete the default and set the following permissions for the repository

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": "*",
      "Action": [
        "ecr:BatchGetImage",
        "ecr:DescribeImages",
        "ecr:GetDownloadUrlForLayer",
        "ecr:PullImage"
      ]
    }
  ]
}
```

S3 Bucket Configuration

Step 1: Go over to S3 and create a private bucket for the project. Check if the settings matches the following screenshots and keep the defaults for rest of the configurations.

Amazon S3 > Buckets > Create bucket

Create bucket Info

Buckets are containers for data stored in S3. Learn more

General configuration

Bucket name

vms-bucket

Bucket name must be unique within the global namespace and follow the bucket naming rules. See rules for bucket naming

AWS Region

Asia Pacific (Singapore) ap-southeast-1

Copy settings from existing bucket - *optional*

Only the bucket settings in the following configuration are copied.

Choose bucket

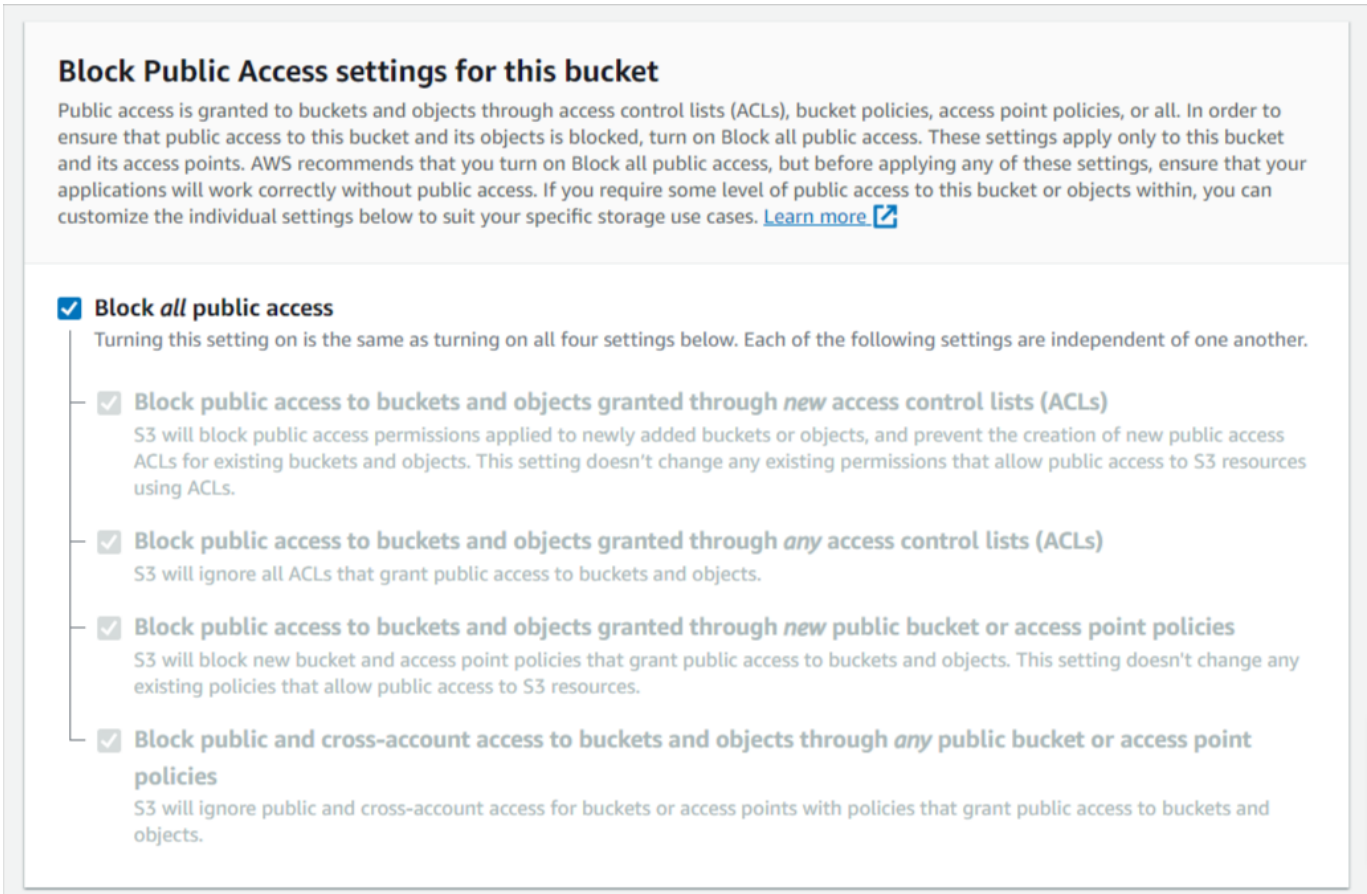
Object Ownership Info

Control ownership of objects written to this bucket from other AWS accounts and the use of access control lists (ACLs). Object ownership determines who can specify access to objects.

- (selected) **ACLs disabled (recommended)**
  All objects in this bucket are owned by this account. Access to this bucket and its objects is specified using only policies.
- (not selected) **ACLs enabled**
  Objects in this bucket can be owned by other AWS accounts. Access to this bucket and its objects can be specified using ACLs.

Object Ownership

Bucket owner enforced



AWS EKS Setup

Configure the following in the machine you are going to access the cluster:

- AWS CLI
- AWS IAM Authenticator
- Kubectl

Run the following command to get kubeconfig file for the new cluster:

Linux:

aws eks –region $(terraformoutput - rawregion)update - kubeconfig - -name$
(terraform output -raw cluster_name)

Windows:

set region_code=region-code

set cluster_name=my-cluster

set account_id=111122223333

for /f "tokens=*" %%a in ('aws eks describe-cluster --region %region_code% --name %cluster_name% --query "cluster.endpoint" --output text') do set cluster_endpoint=%%a

for /f "tokens=*" %%a in ('aws eks describe-cluster --region %region_code% --name %cluster_name% --query "cluster.certificateAuthority.data" --output text') do set certificate_data=%%a

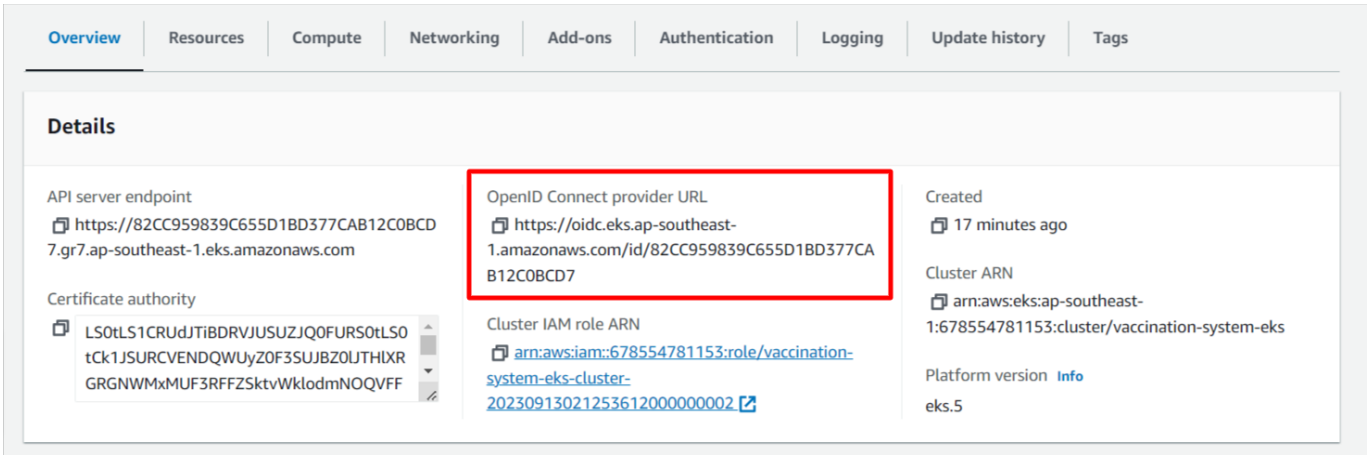
aws eks update-kubeconfig --region %region_code% --name %cluster_name%

aws eks –region ap-southeast-1 update-kubeconfig –name vaccination-system-eks

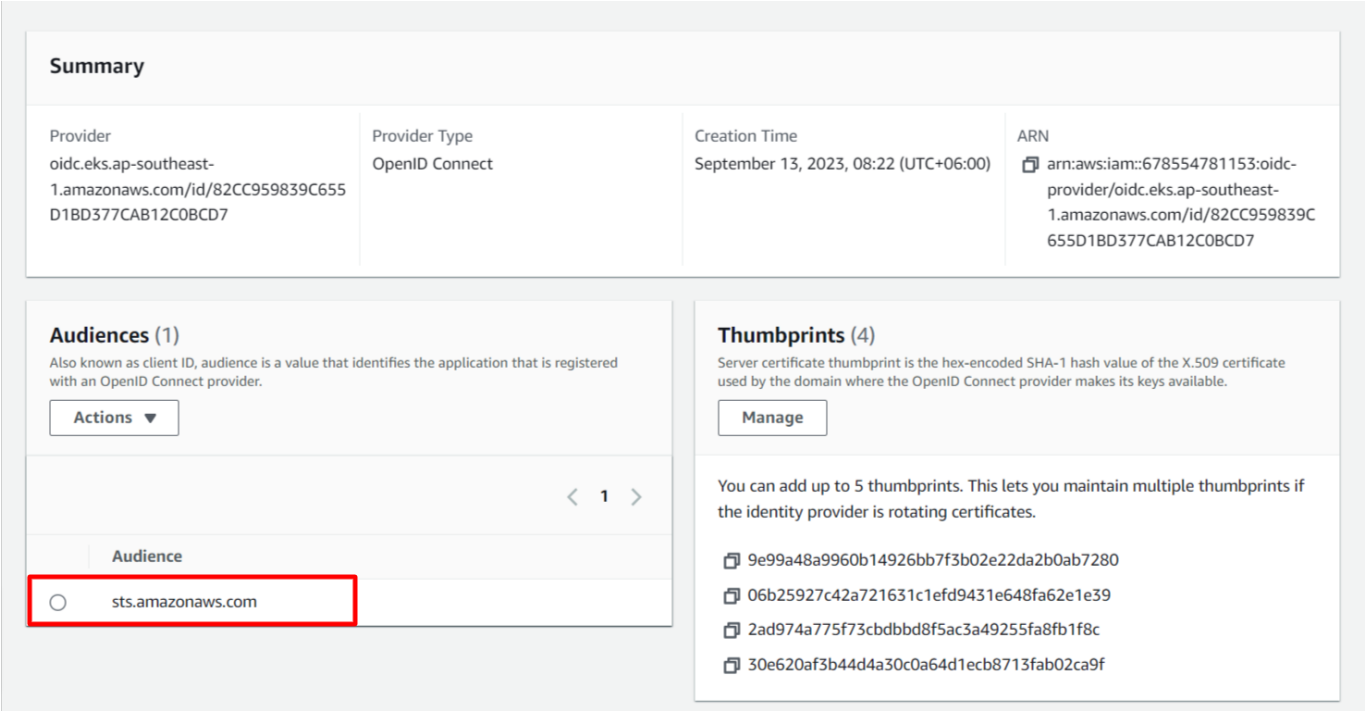
AWS Load balancer Controller Configuration

Create Identity provider

Step 1: Copy OpenIDConnect URL from EKS overview



Step 2: Go to IAM console>Identity Provider and create a OpenID Connect provider using the connector provider URL copied in the earlier step. Use sts.amazonaws.com as the audience.



Step 3: Now create an IAM policy from AWS load balancer controller documentation for the version you are using. I am using v2.6.1 in this project.
https://github.com/kubernetes-sigs/aws-load-balancer-controller/blob/v2.6.1/docs/install/iam_policy.json