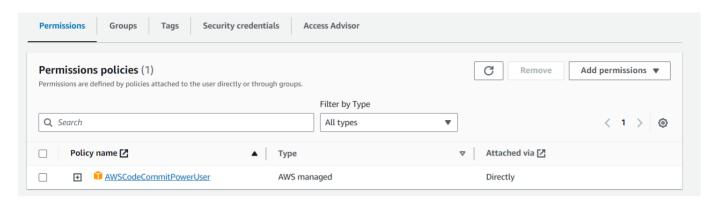
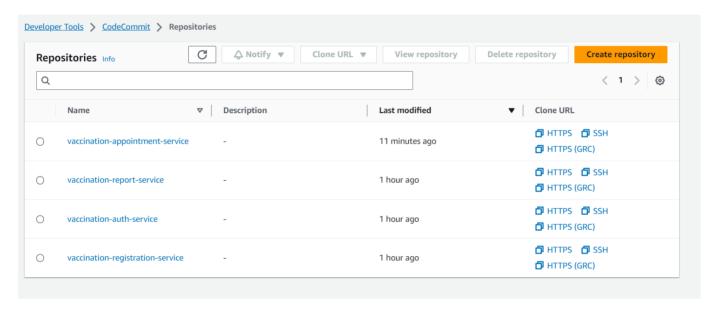
# **CodeCommit Configuration**

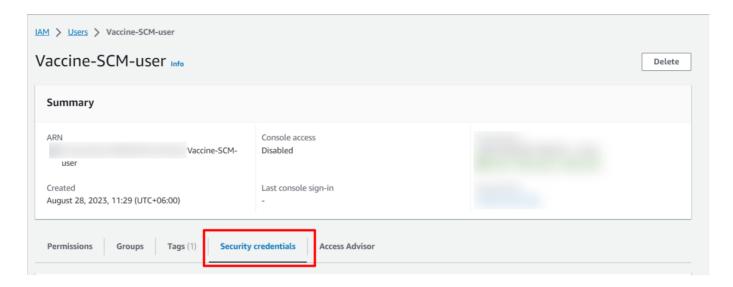
Step 1: Create an IAM User with AWSCodeCommitPowerUser policy.

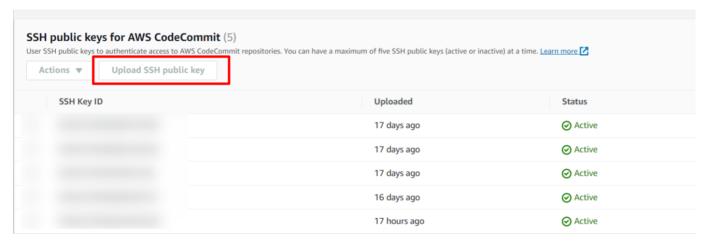


Step 2: Create Repositories

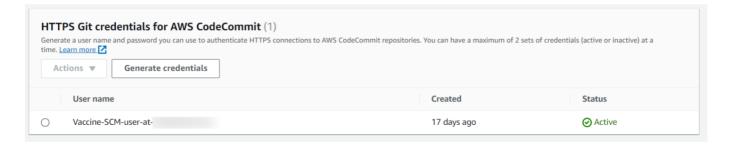


Step 3: Add your SSH keys to the newly created user in Step 1 security credentials. Up to 5 SSH can be added per IAM user.





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

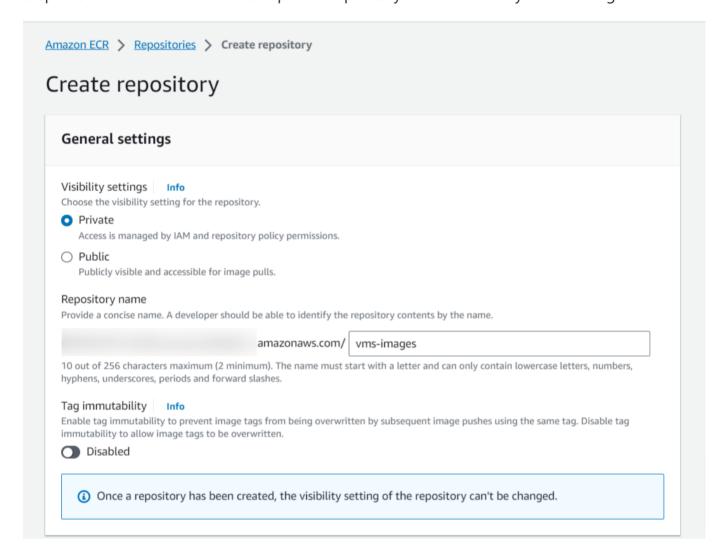


Step 5: Copy the username and password that IAM generated for you, either by showing, copying, and then pasting this information into a secure file on your local computer, or by choosing Download credentials to download this information as a .CSV file. You need this information to connect to CodeCommit.

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.

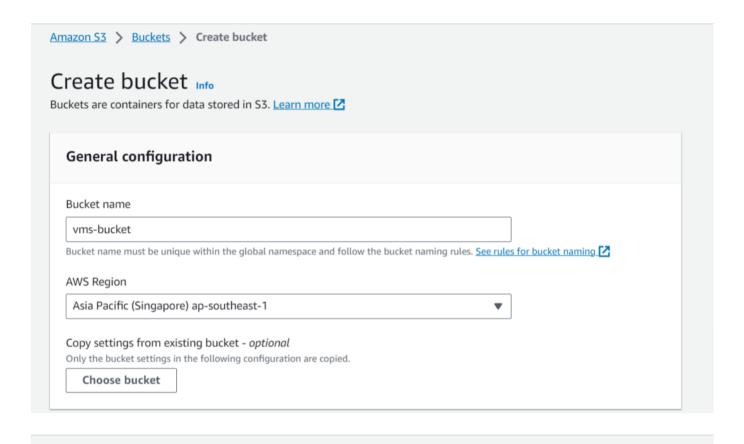


Step 2: Next, go to Permissions>Edit JSON Policy and delete the default and set the following permissions for the repository

```
"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.



#### 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.

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. 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

#### **Block Public Access settings for this bucket**

Public access is granted to buckets and objects through access control lists (ACLs), bucket policies, access point policies, or all. In order to ensure that public access to this bucket and its objects is blocked, turn on Block all public access. These settings apply only to this bucket and its access points. AWS recommends that you turn on Block all public access, but before applying any of these settings, ensure that your applications will work correctly without public access. If you require some level of public access to this bucket or objects within, you can customize the individual settings below to suit your specific storage use cases. Learn more

#### Block all public access

Turning this setting on is the same as turning on all four settings below. Each of the following settings are independent of one another.

- Block public access to buckets and objects granted through new access control lists (ACLs)
   S3 will block public access permissions applied to newly added buckets or objects, and prevent the creation of new public access ACLs for existing buckets and objects. This setting doesn't change any existing permissions that allow public access to S3 resources using ACLs.
- Block public access to buckets and objects granted through any access control lists (ACLs)
   S3 will ignore all ACLs that grant public access to buckets and objects.
- Block public access to buckets and objects granted through new public bucket or access point policies
   S3 will block new bucket and access point policies that grant public access to buckets and objects. This setting doesn't change any existing policies that allow public access to S3 resources.
- Block public and cross-account access to buckets and objects through any public bucket or access point policies

S3 will ignore public and cross-account access for buckets or access points with policies that grant public access to buckets and objects.

#### **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

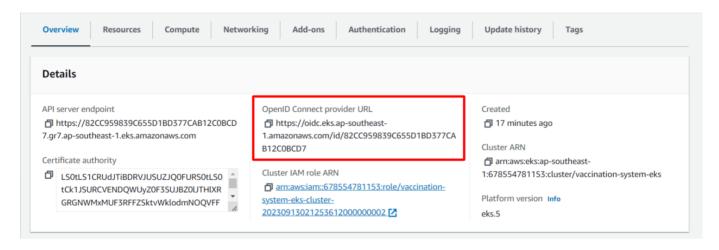
for /f "tokens=\*" %%a in ('aws eks describe-cluster --region %region\_code% --name %cluster\_name% --query "cluster.certificateAuthority.data" --output text') do set certificate data=%%a

aws eks update-kubeconfig --region %region\_code% --name %cluster\_name% aws eks -region ap-southeast-1 update-kubeconfig -name vaccination-system-eks

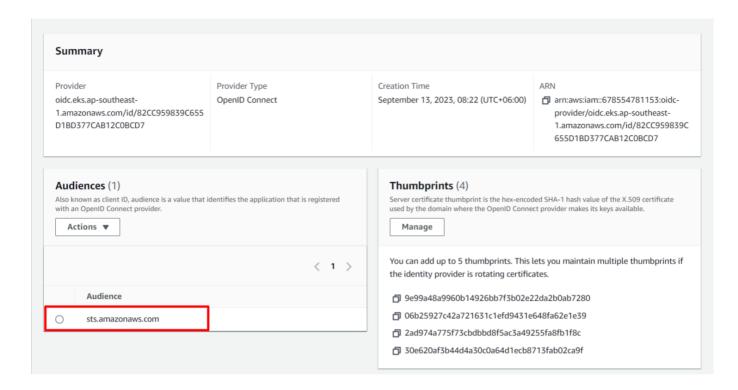
# AWS Load balancer Controller Configuration

### **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