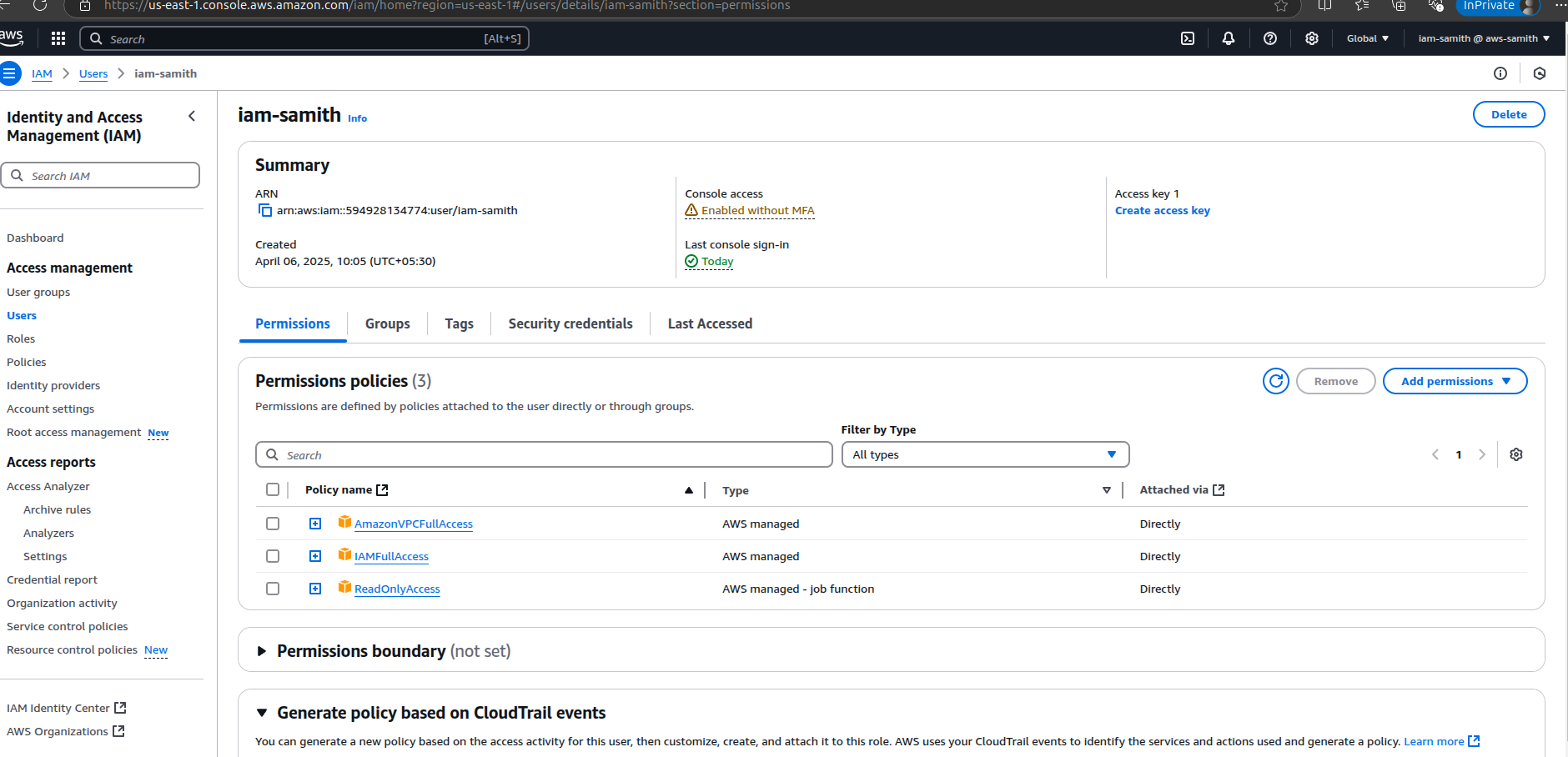
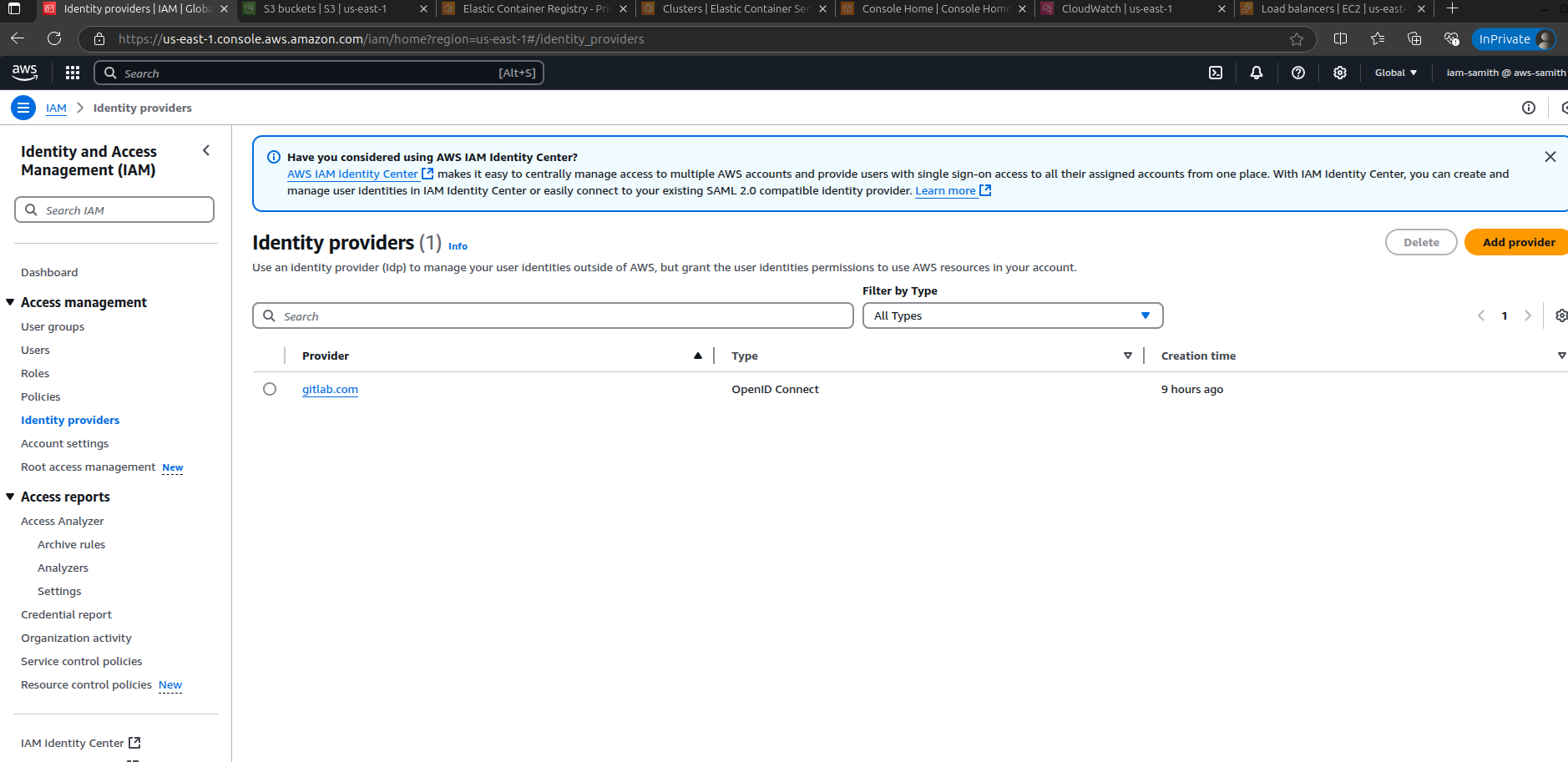
**Platform Engineer Assignment: Deploying a Containerized Applicaiton Using IaC**

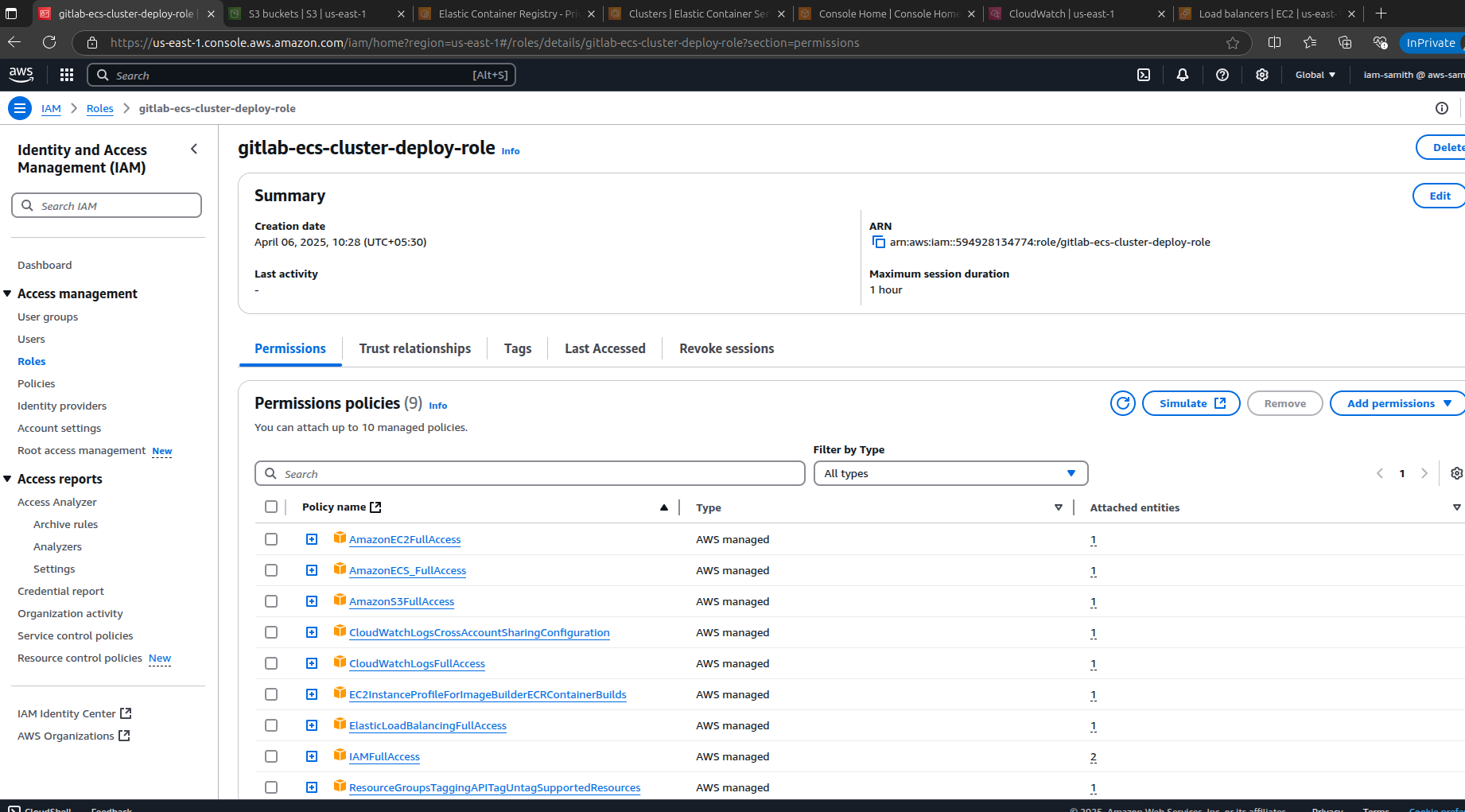
1. For the best practice, firstly I have created IAM account for the complete that assignment providing required permissions.



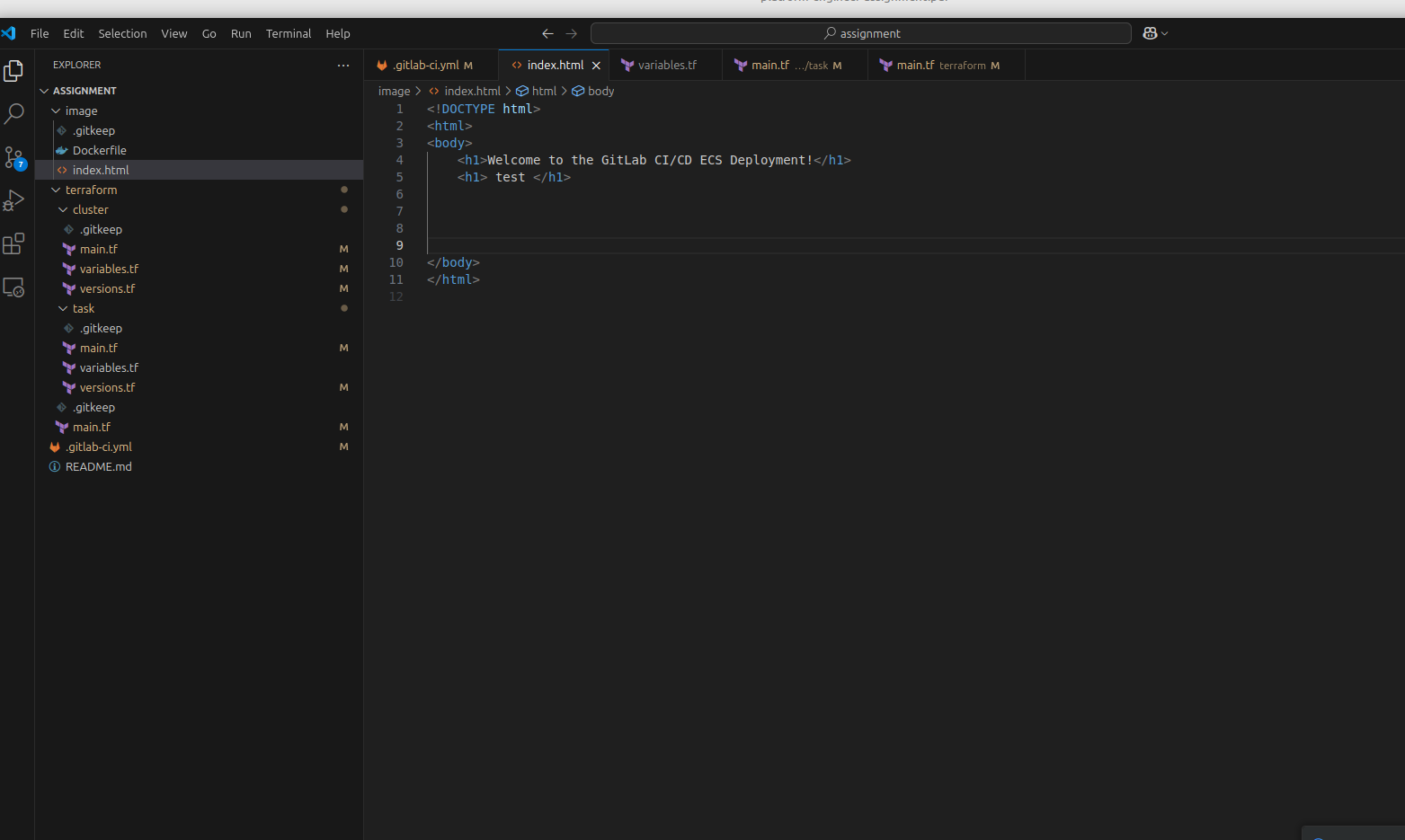
1. As per the assignment that mentioned for the pipeline creation use GitLab. Therefore, to authenticate, secure access to environment in AWS and GitLab we need to have an Identity provider.



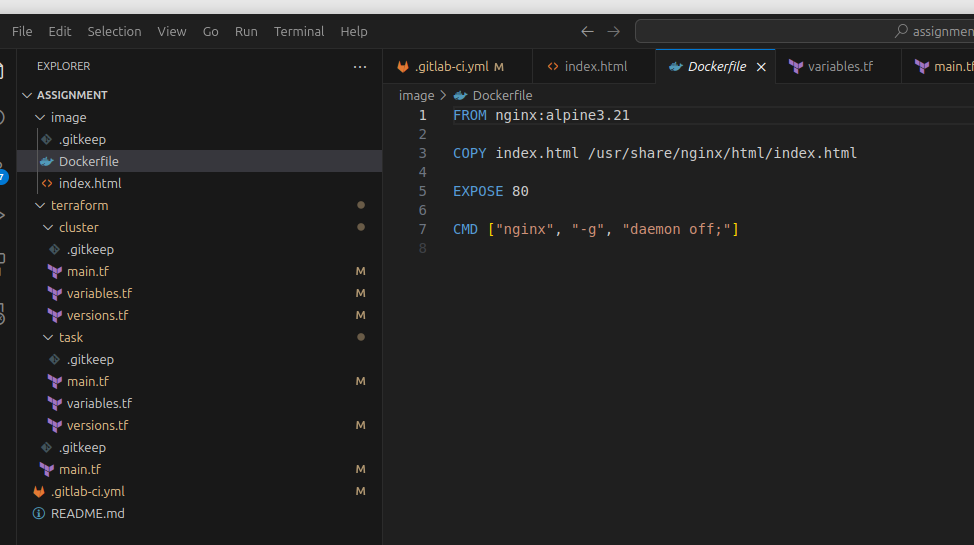
Then we need to have a role for the creation resource while running the pipeline in GitLab. Hence, we have assigned some required permission that that role and assigned that role to the above crated provider.

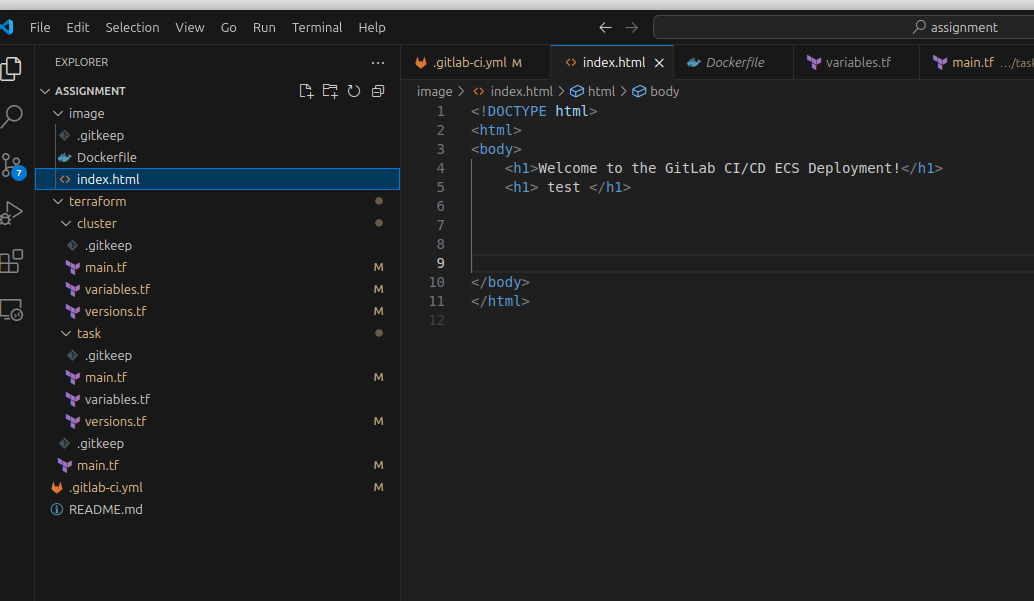


1. Then log into GitLab and create a new repo for the task. For the task file structure is Dockerfile and custom page under image folder. For the terraform configuration files are under the terraform folder. Finaly pipeline yaml file also there.

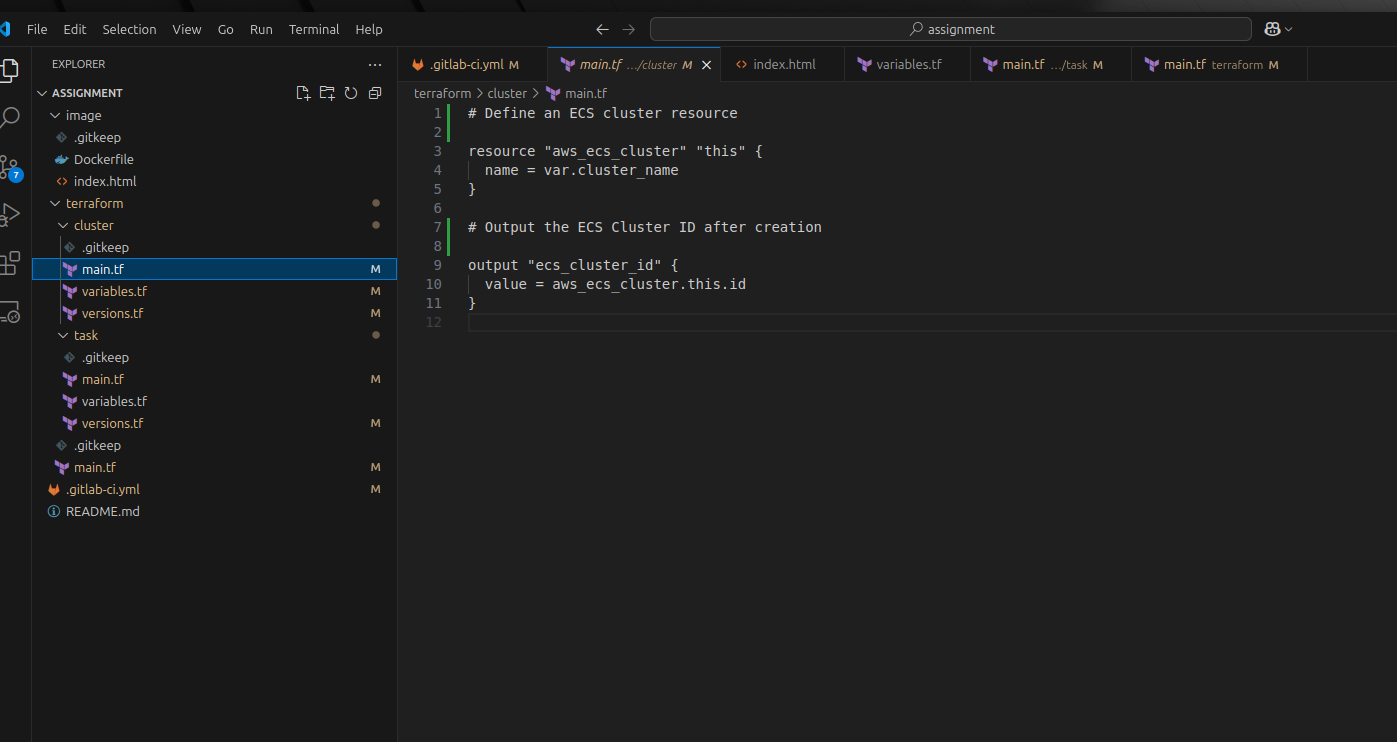


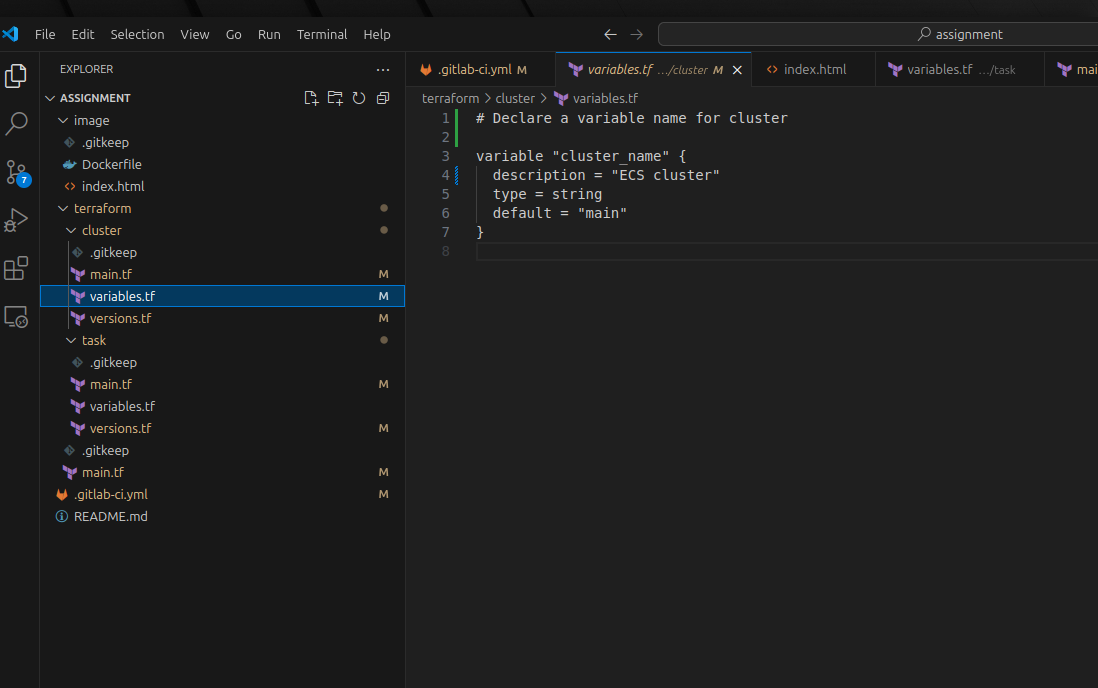
1. Firstly, Dockerfile and custom page is created. As per the image I have used alpine image because it's small and has less vulnerabilities. Then copy the custom html page into container and expose port 80.

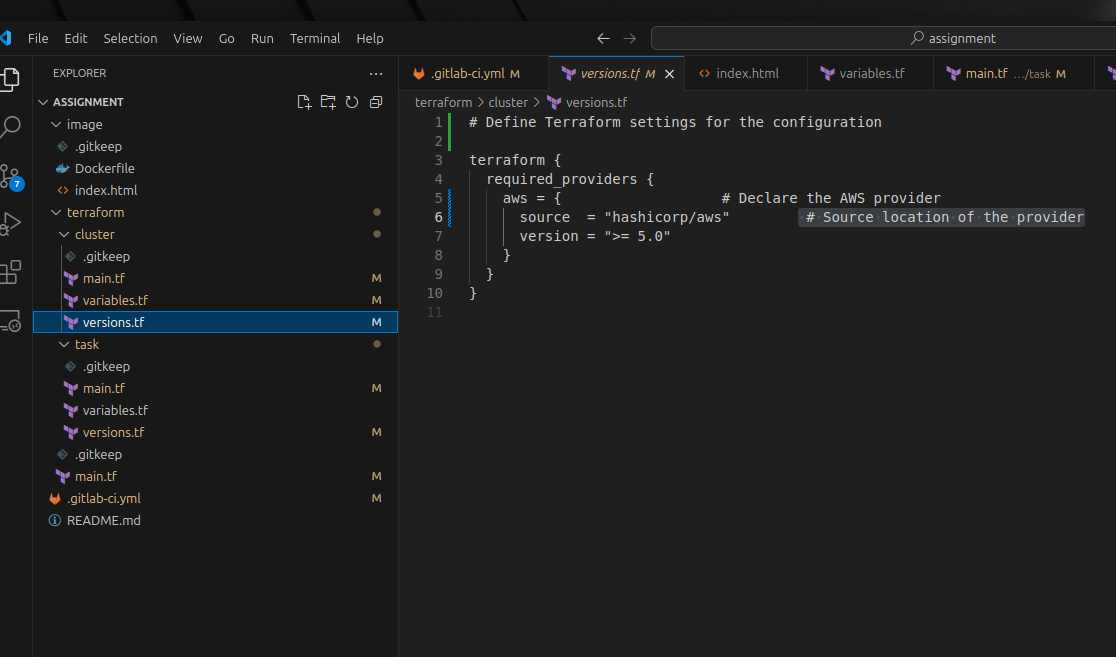




1. Then under the terraform/cluster folder define 3 terraform files to define variables and ECS cluster creation related codes. Specially under the versions file declare the AWS as provider for the deployments.





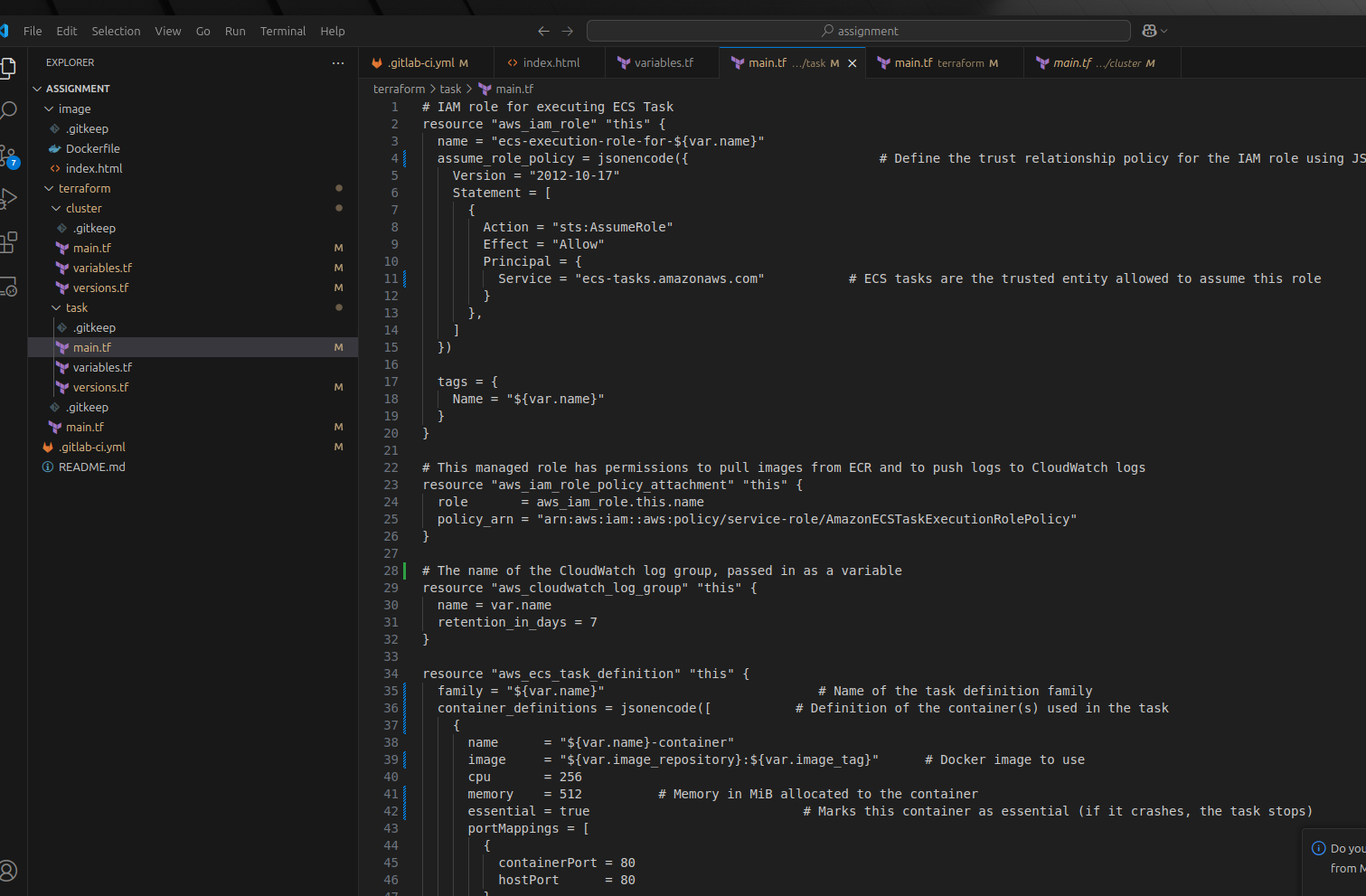


1. Then under the terraform/task folder define 3 terraform files to define variables and other AWS resources creation related codes.

In main.tf file I have created assume role for while ECS task creating resources it has required some permissions. Hence i have created IAM role.

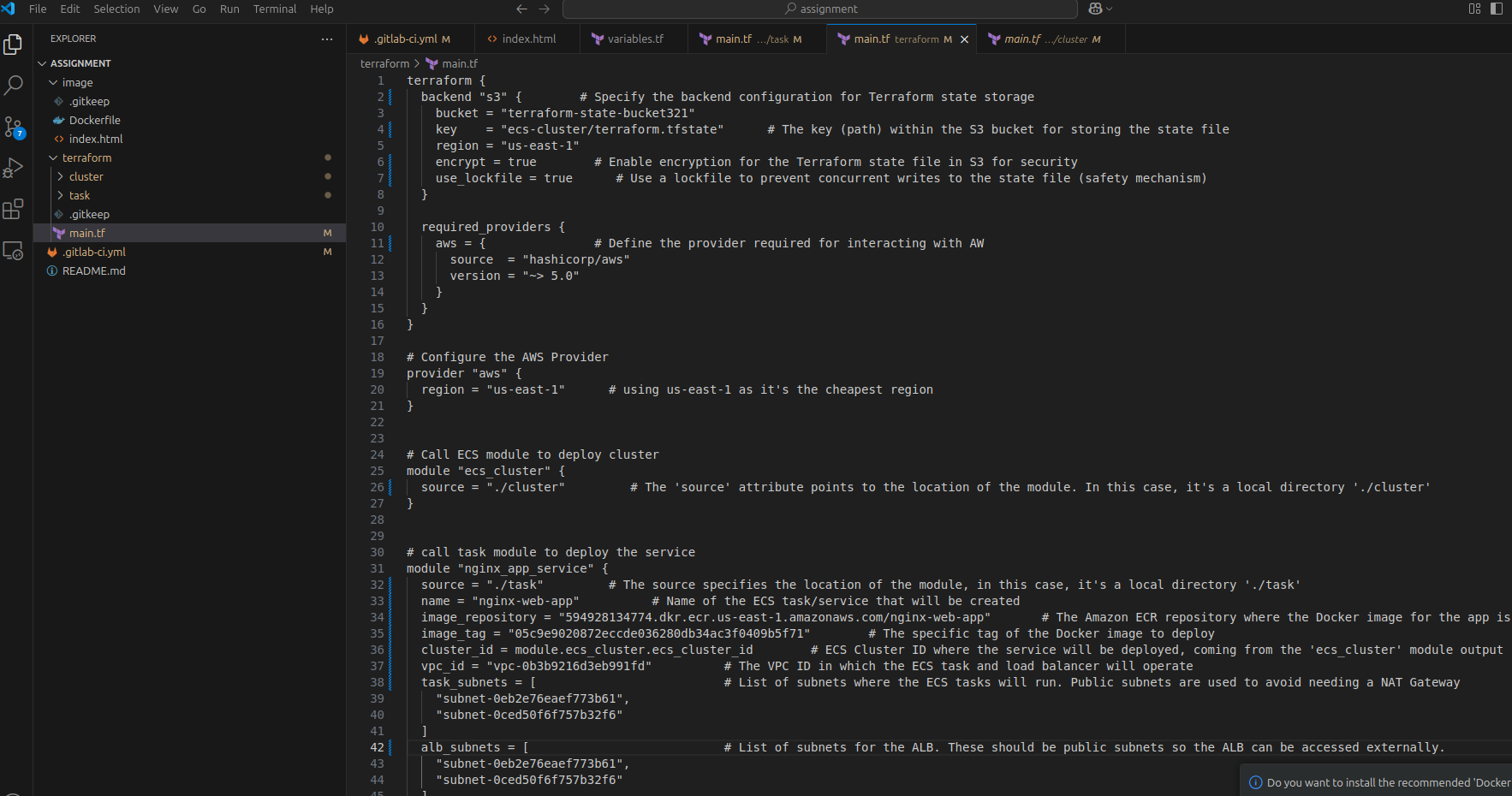
That managed role also has pull images from ECR and push logs to cloud watch. And i have configured those logs only retain 7 days.

Also in that file we need to define ALB related security group details, ingress, egress, listener, ECS service Secuity group etc



1. Finally, in main.tf file define S3 bucket for the storage of state file. It also enables encryption for file security. Then that will call ECS module to deploy the cluster above mentioned. Then task module will call to deploy the service.

**Special note: I had already created a VPC. Hence, I used that. It was created manually. In that VPC has 2 public subnets and private subnets.**

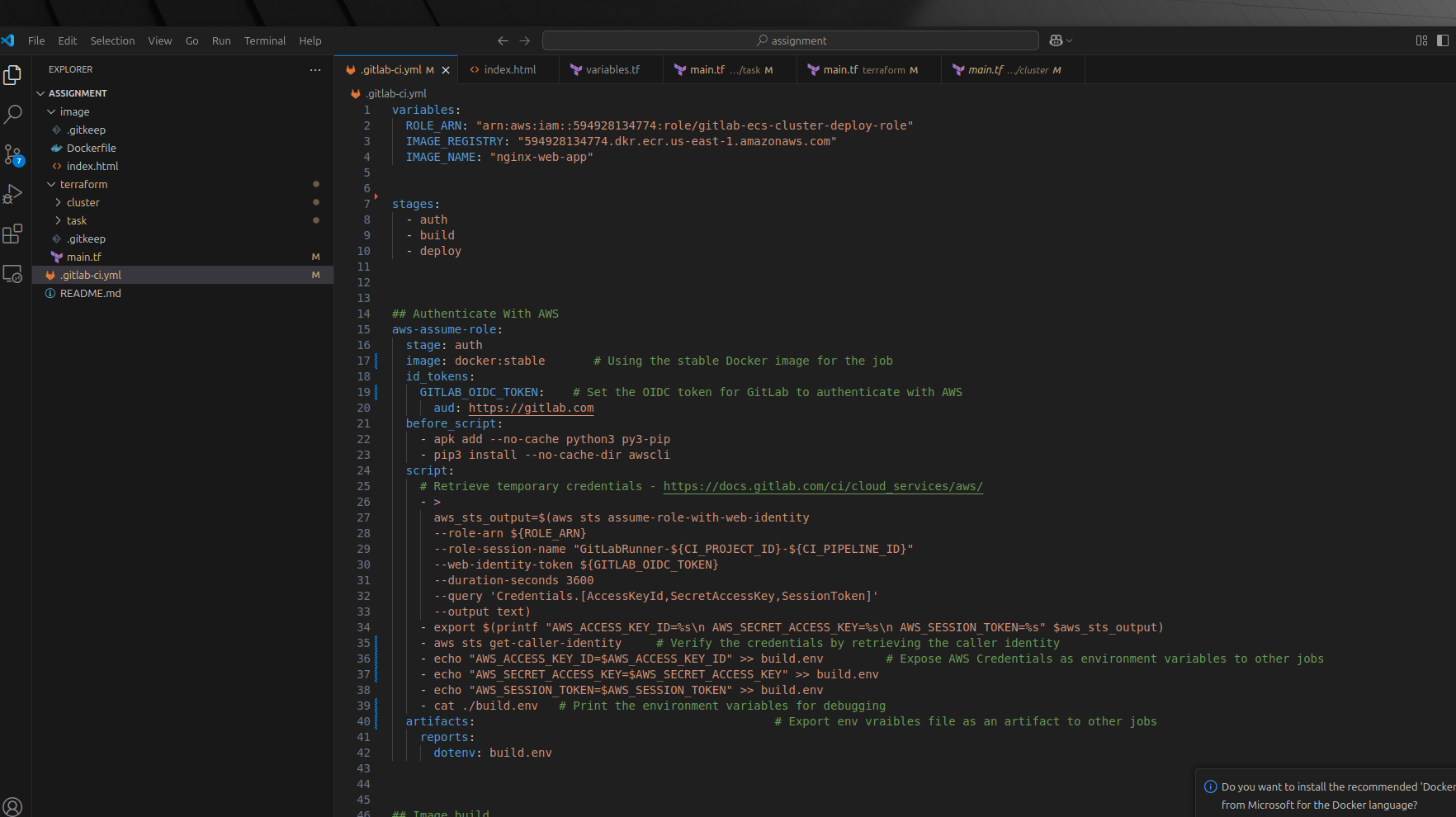


1. As per the last step we need to create a pipeline. Their i define 3 stages for the pipeline.

In the first stage that will authenticate with AWS using the OIDC method. Also, in that stage create artifact as AWS credentials as environment variables and export that as a file, because we need that to run other stages as well.

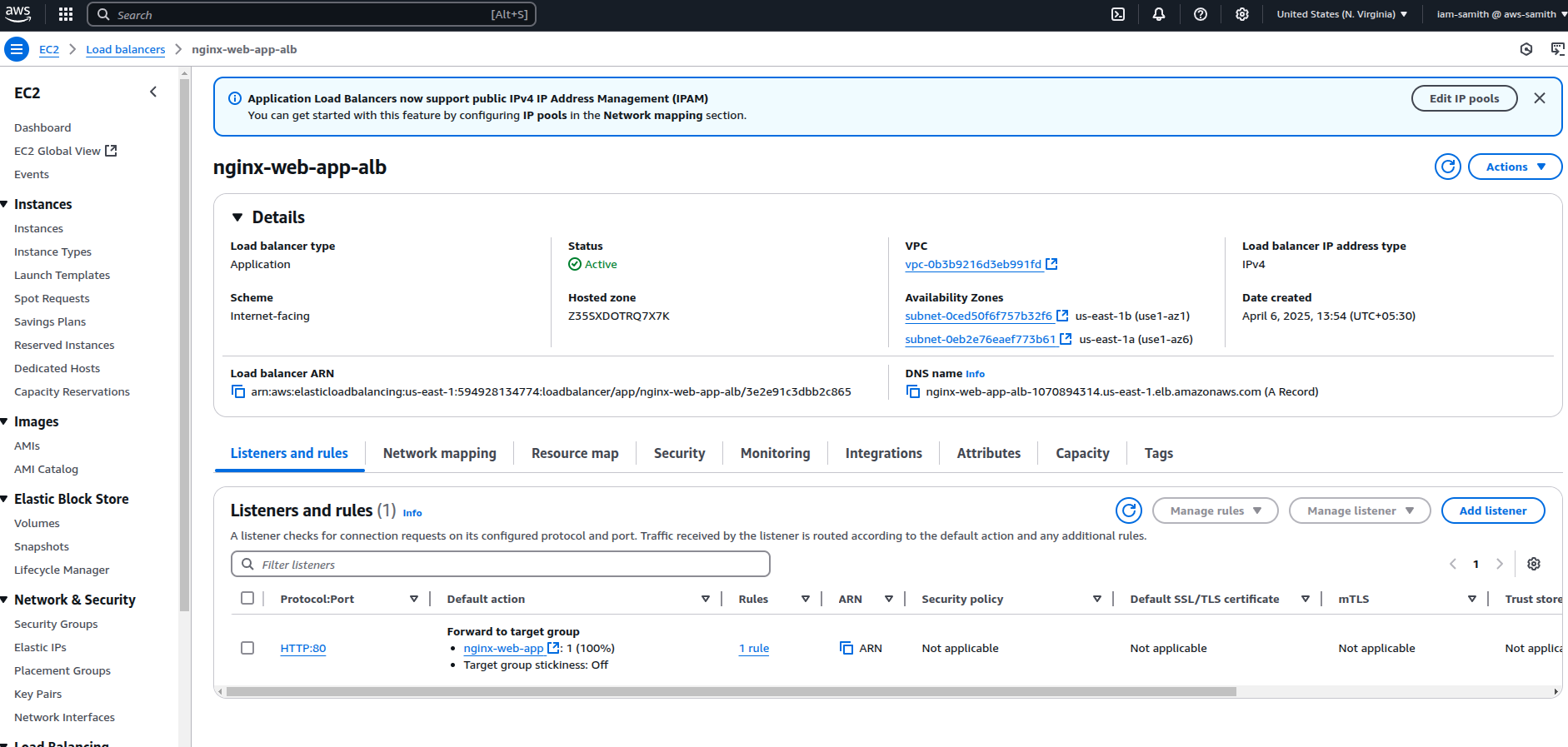
In the second stage it will build the image. Also, here i have specified a rule that only builds images if files inside image/ directory changes.

In the third stage and final stage configured to deploy that in ECS. Also provided some dependencies as well because stages are run in sequence from top to bottom, but jobs in each stage are run in parallel.

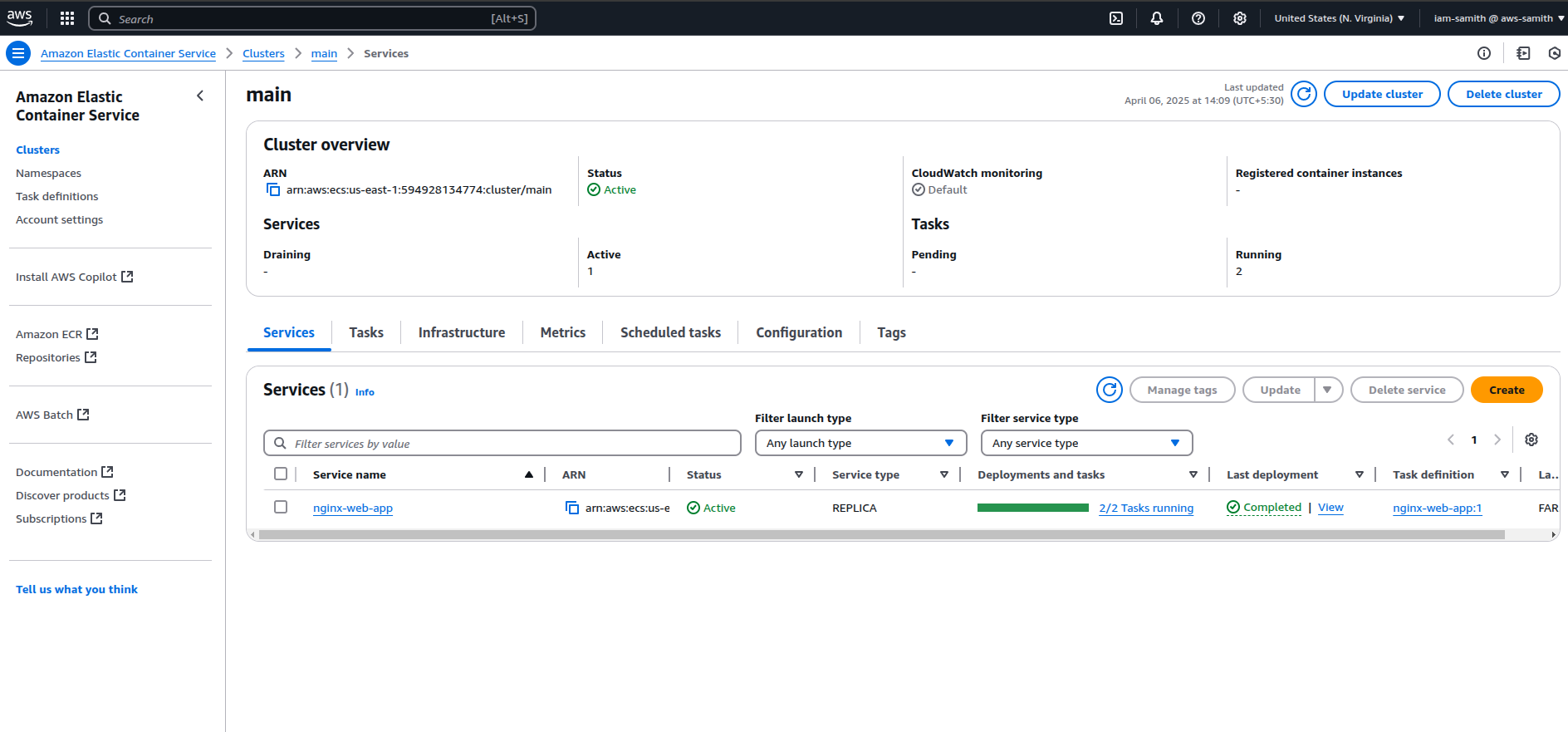


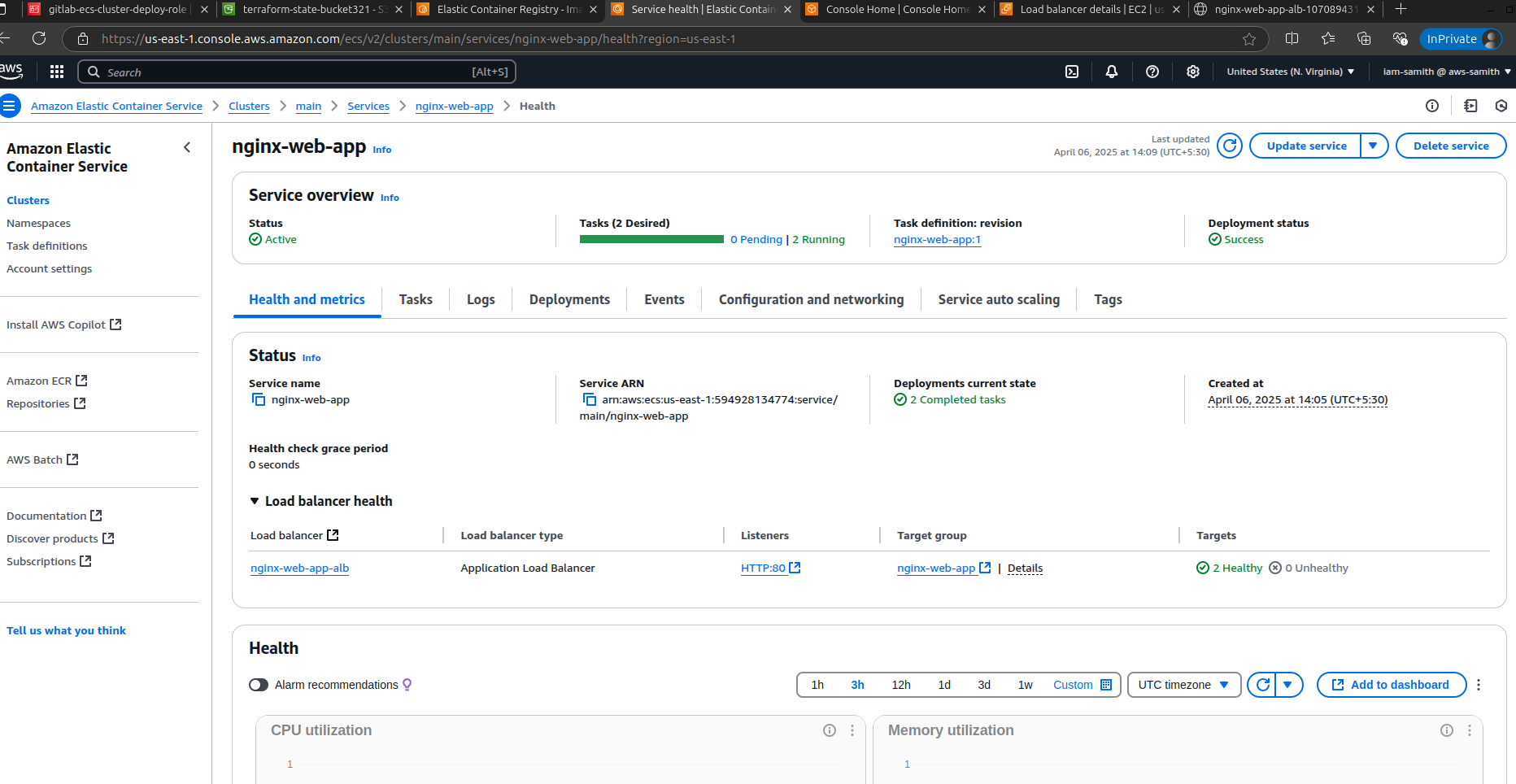
**Evidence**

Load balancer details

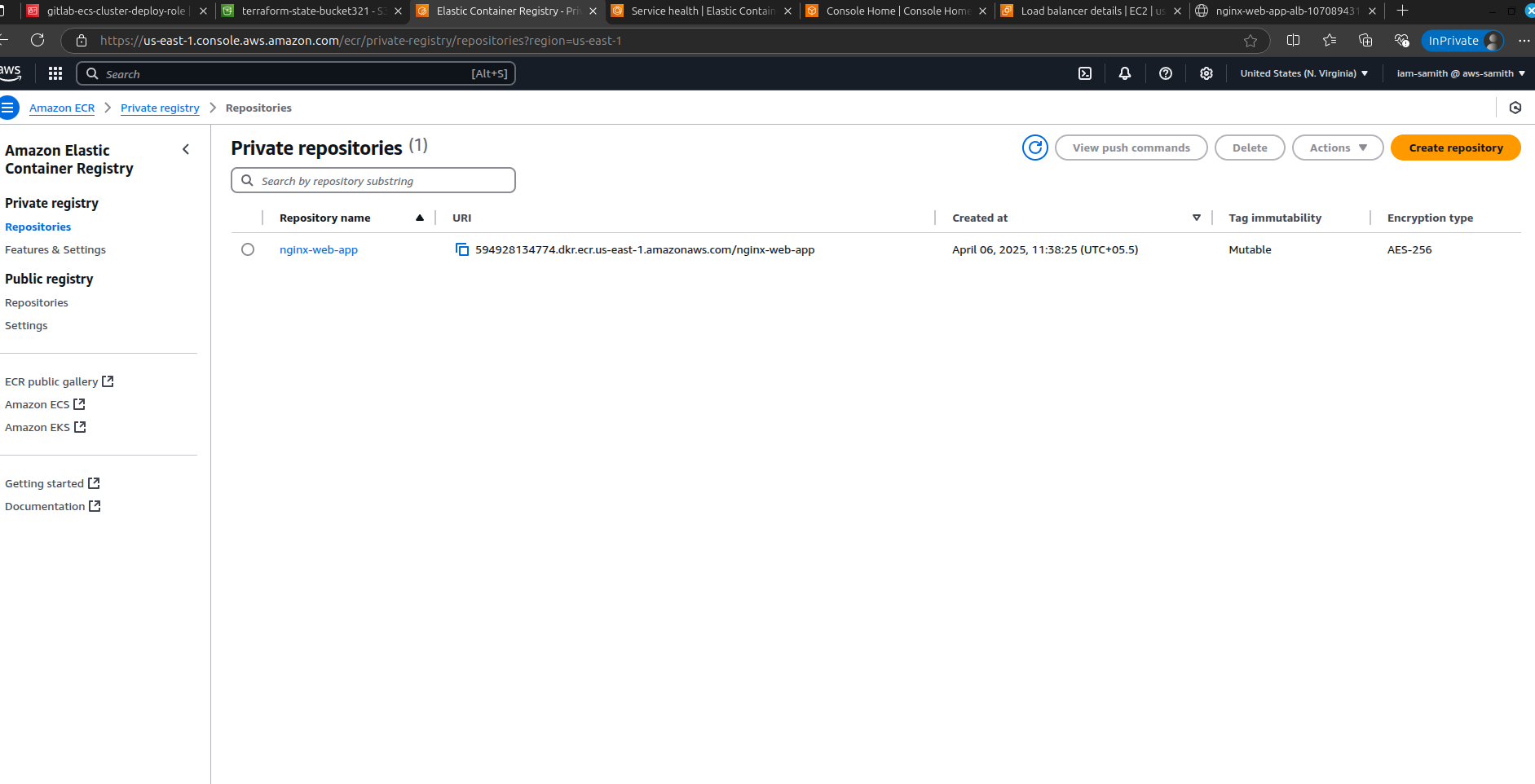


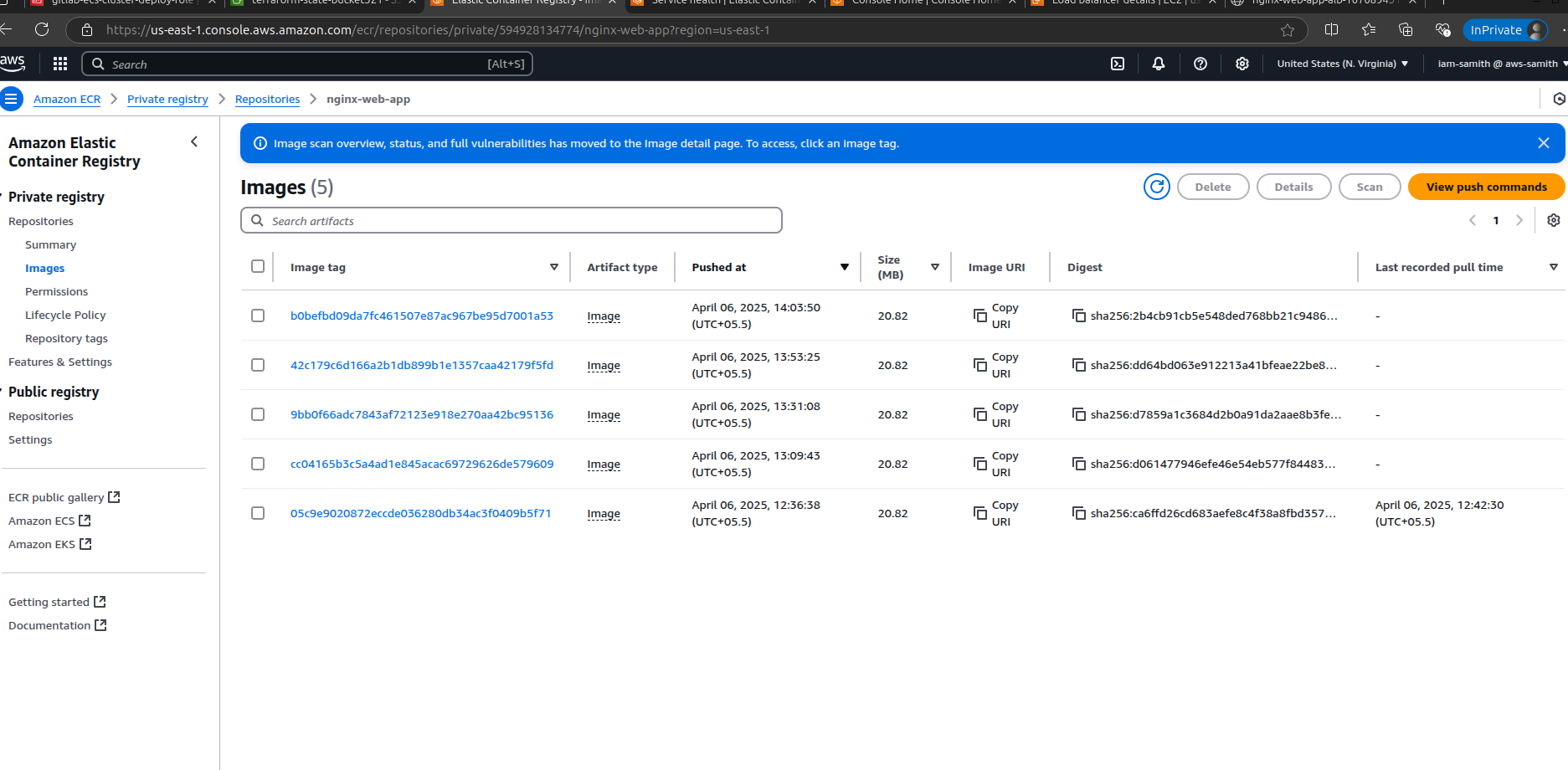
ECS details



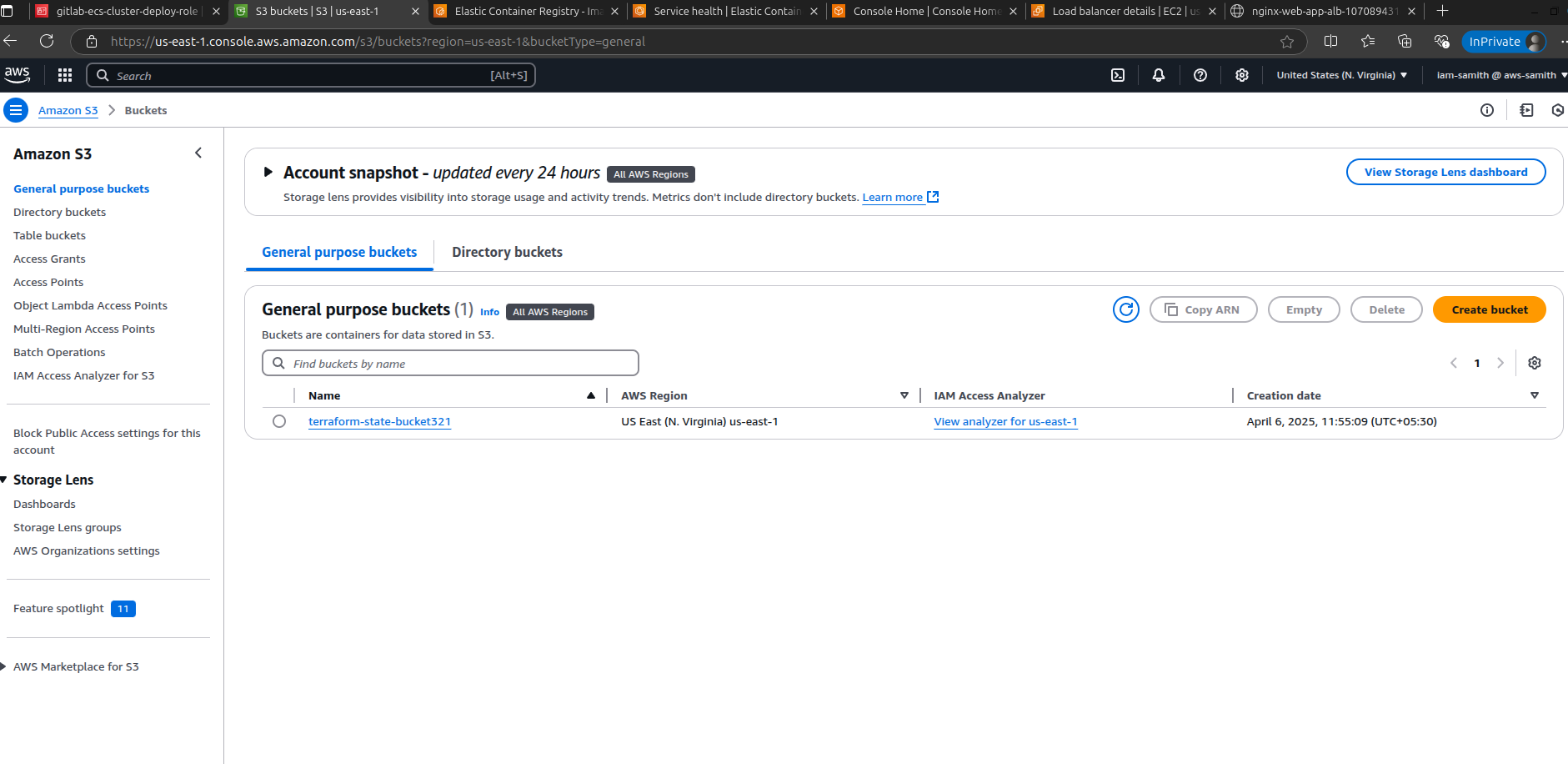


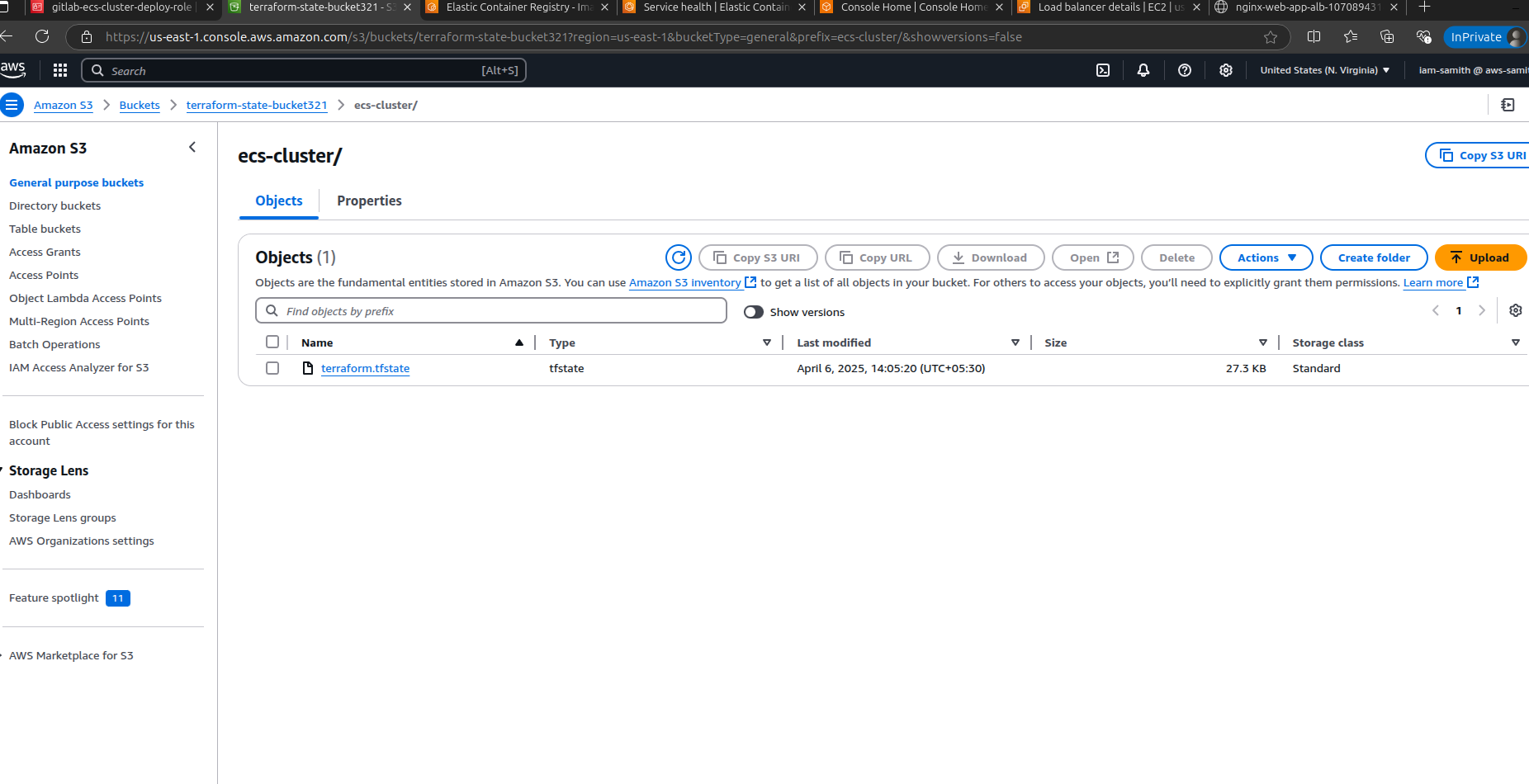
ECR Details



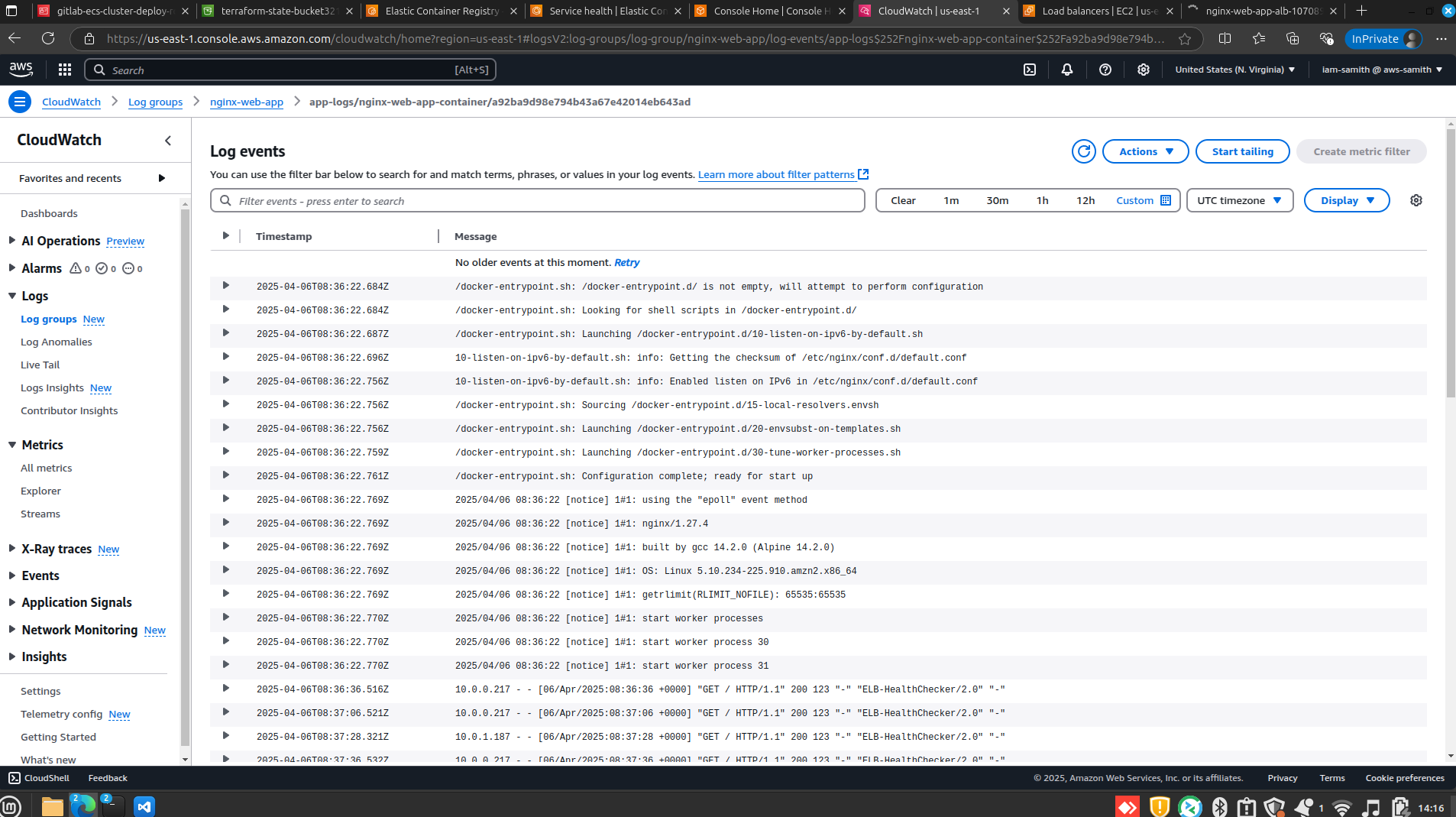


S3 bucket Details

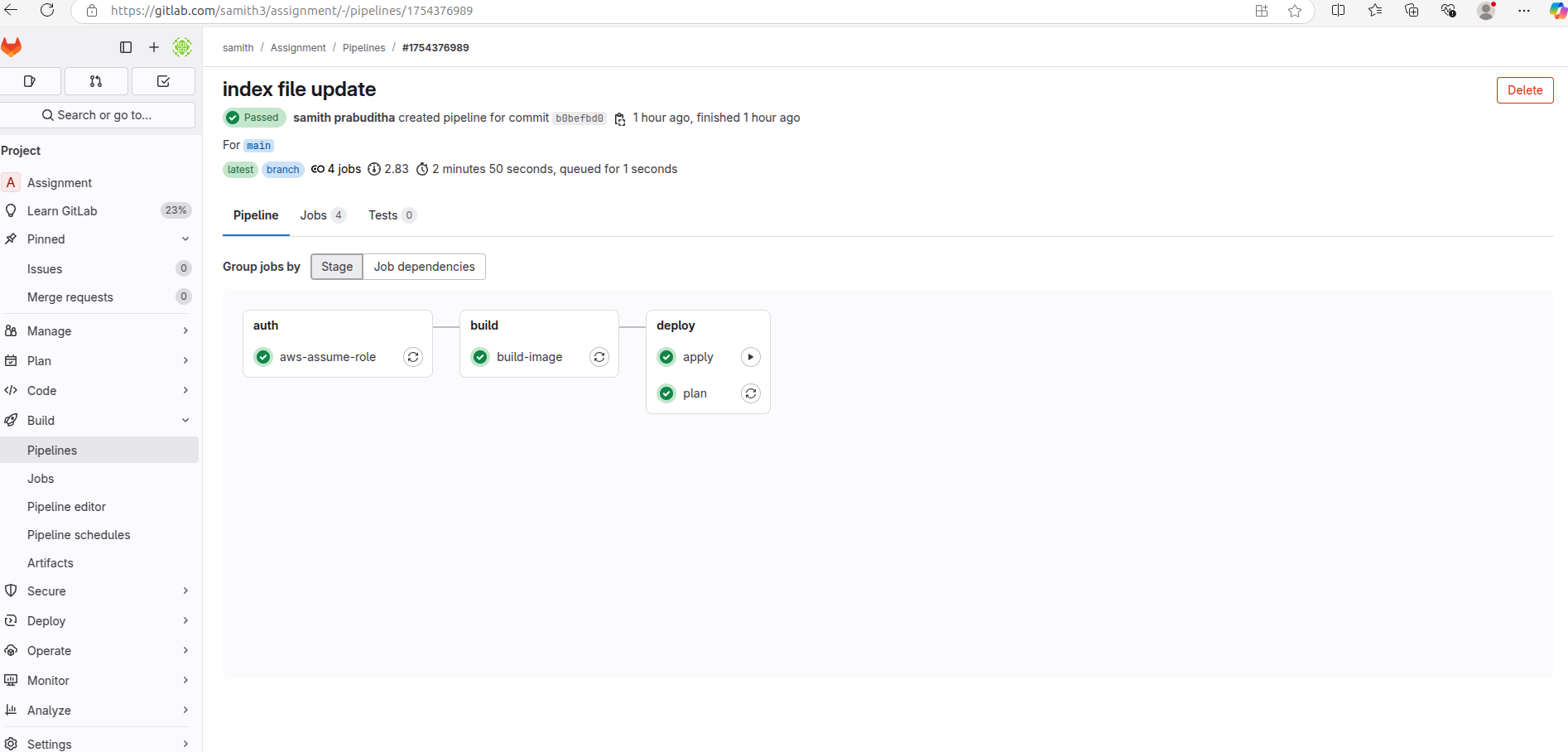


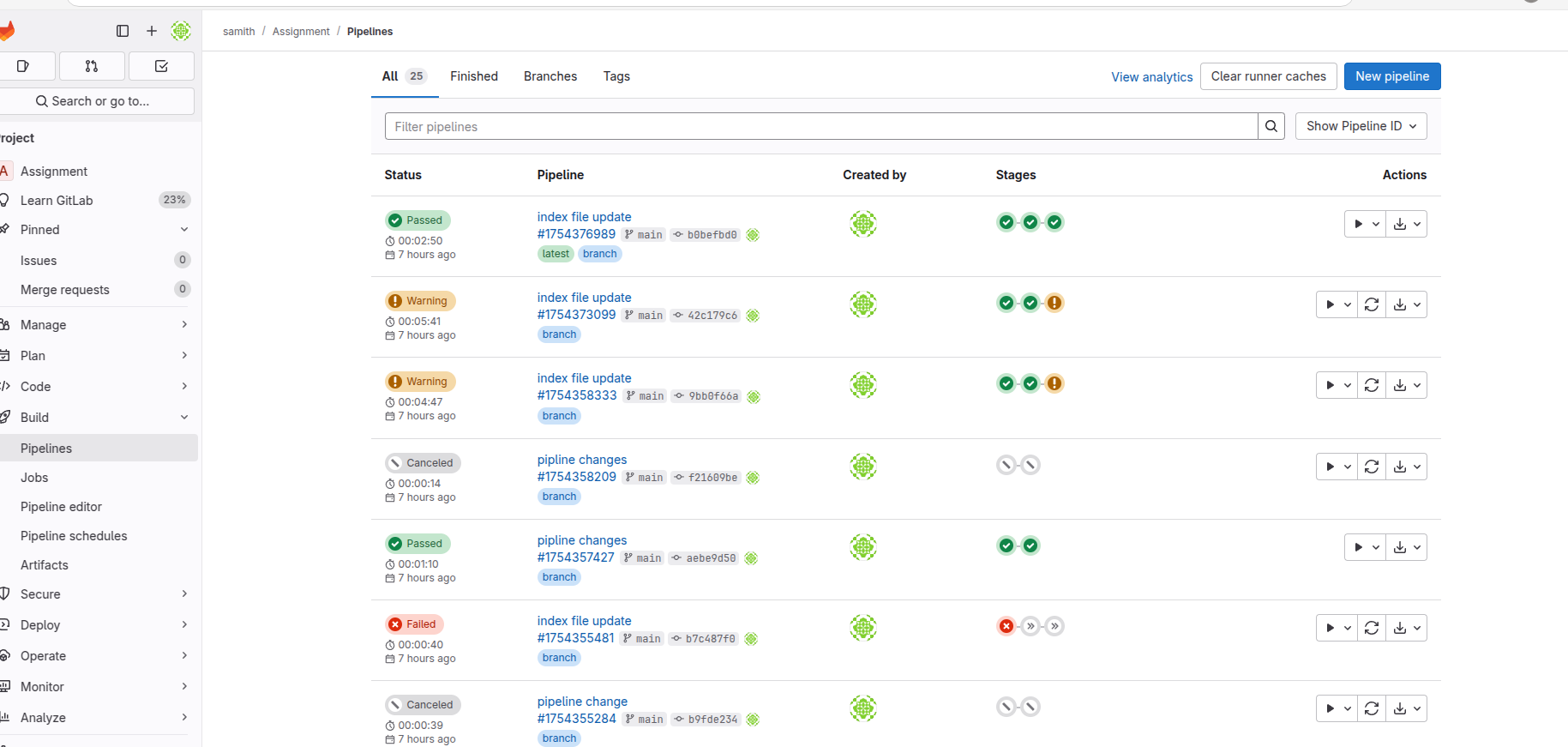


Cloud Watch Log Details



Finally, Deployment





Web Page

