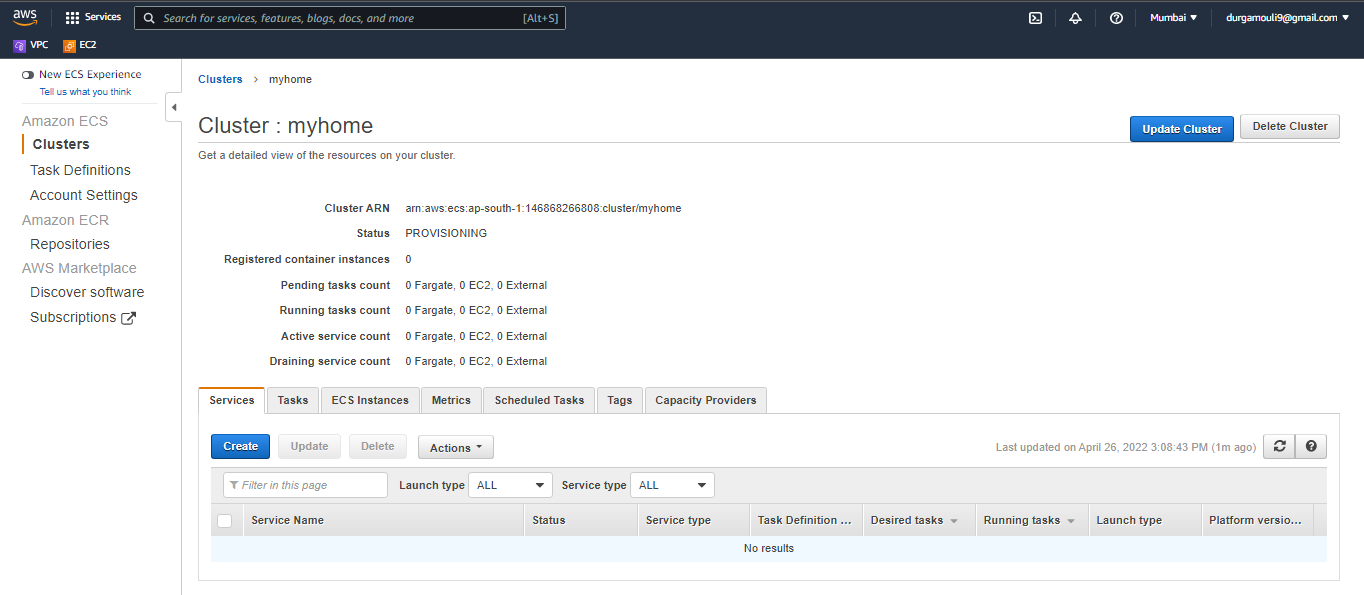
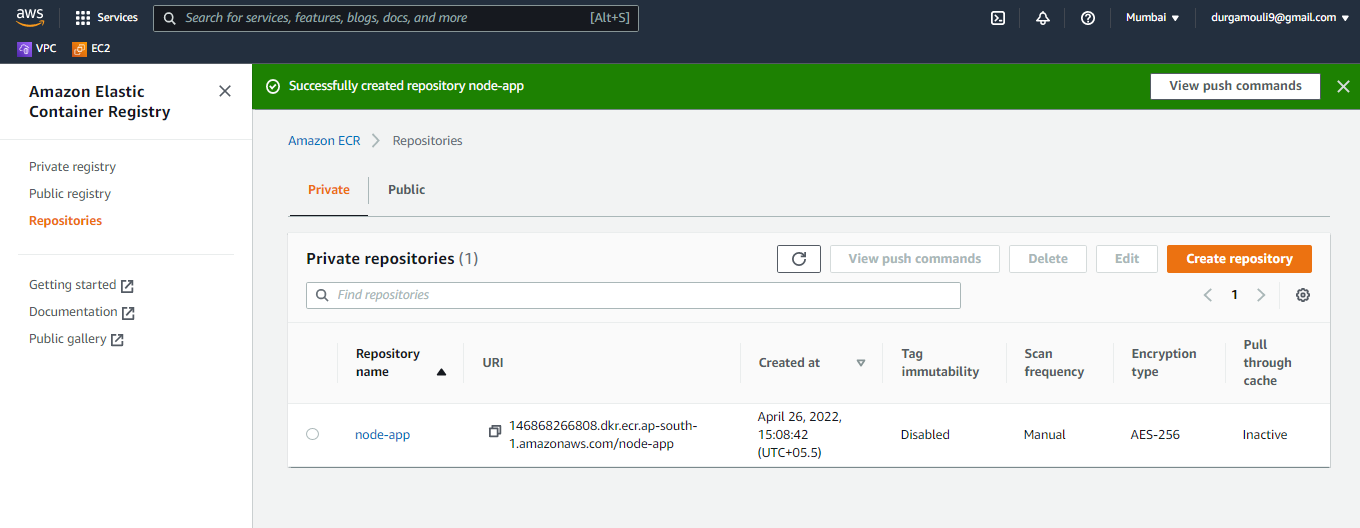
**CICD Pipeline using AWS Code Pipeline:-**

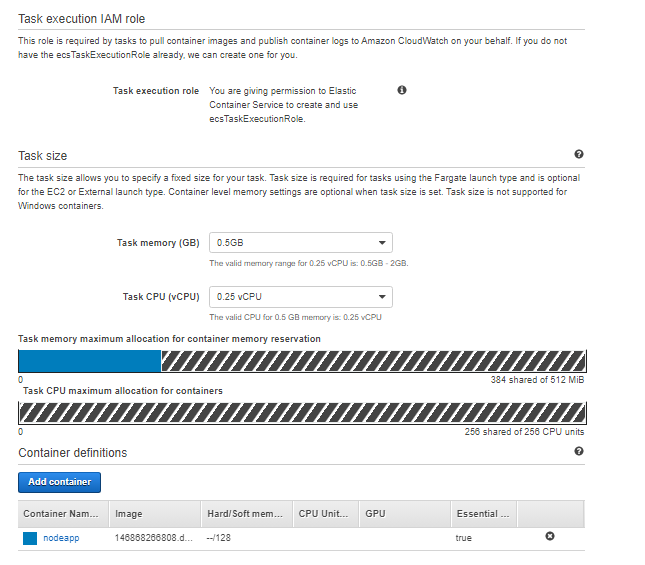
We are going to use components:

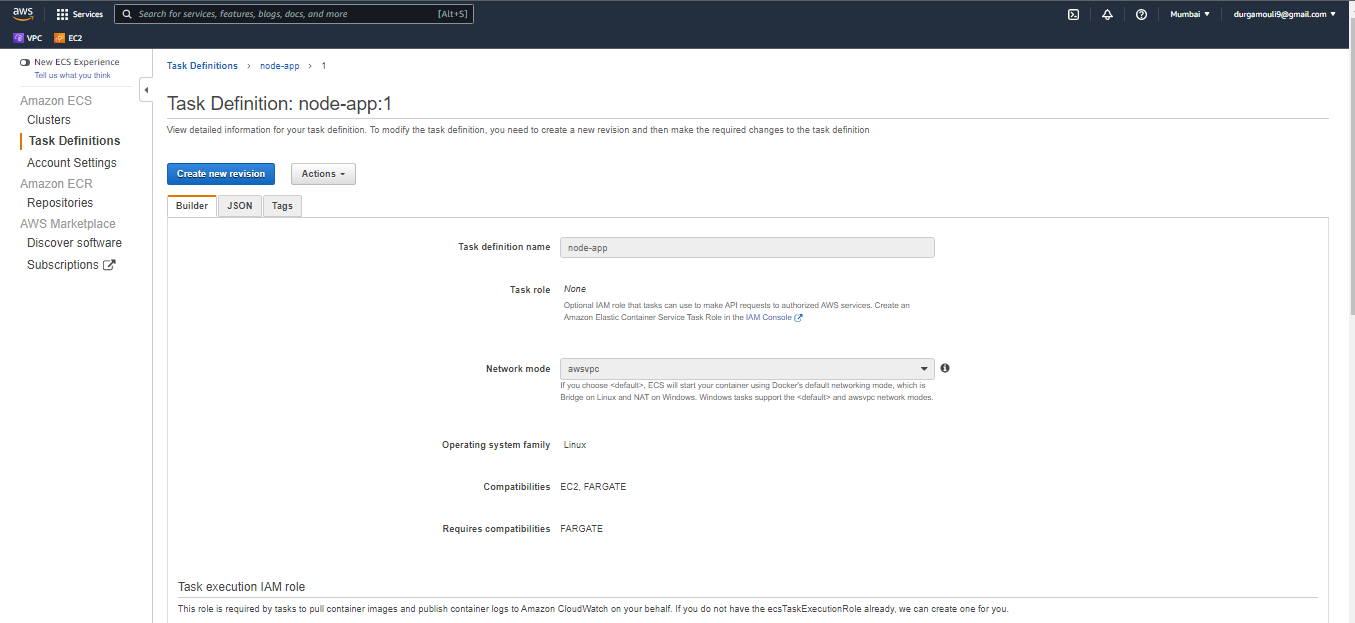
GitHub: Source code Management & for web hook part   
code build : to build Docker images   
ECS fargate cluster : For deploying applications to environments  
  
ECS fargate is AWS web services for orchestrating production grade Docker work loads .

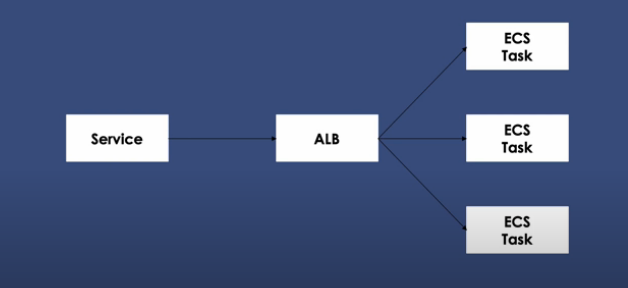
1. Creating of ECS Cluster:  
      
   create with default VPC – name of cluster myhome.  
     
   
2. Create an Repository:



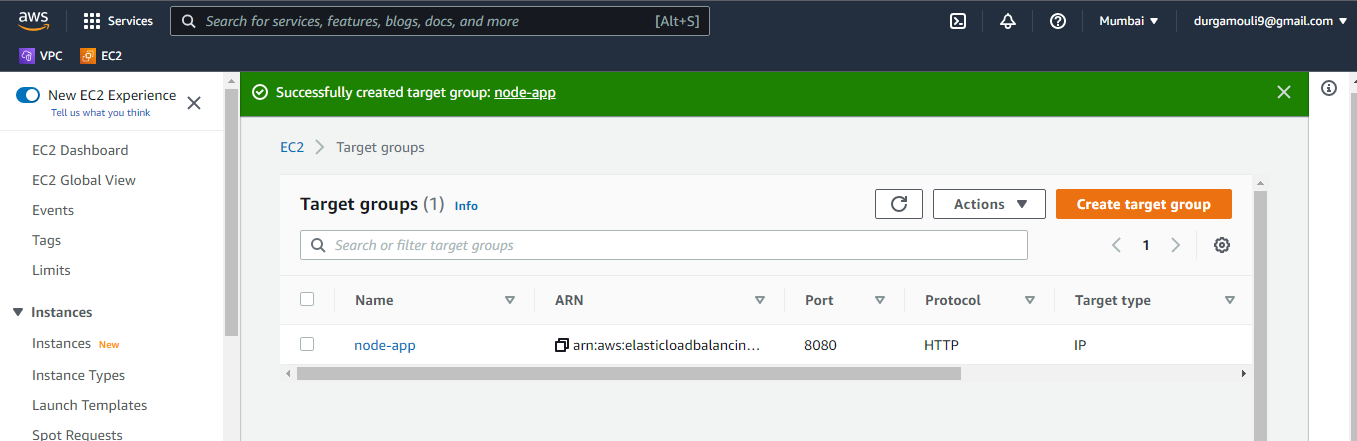
3) Create Task definition:

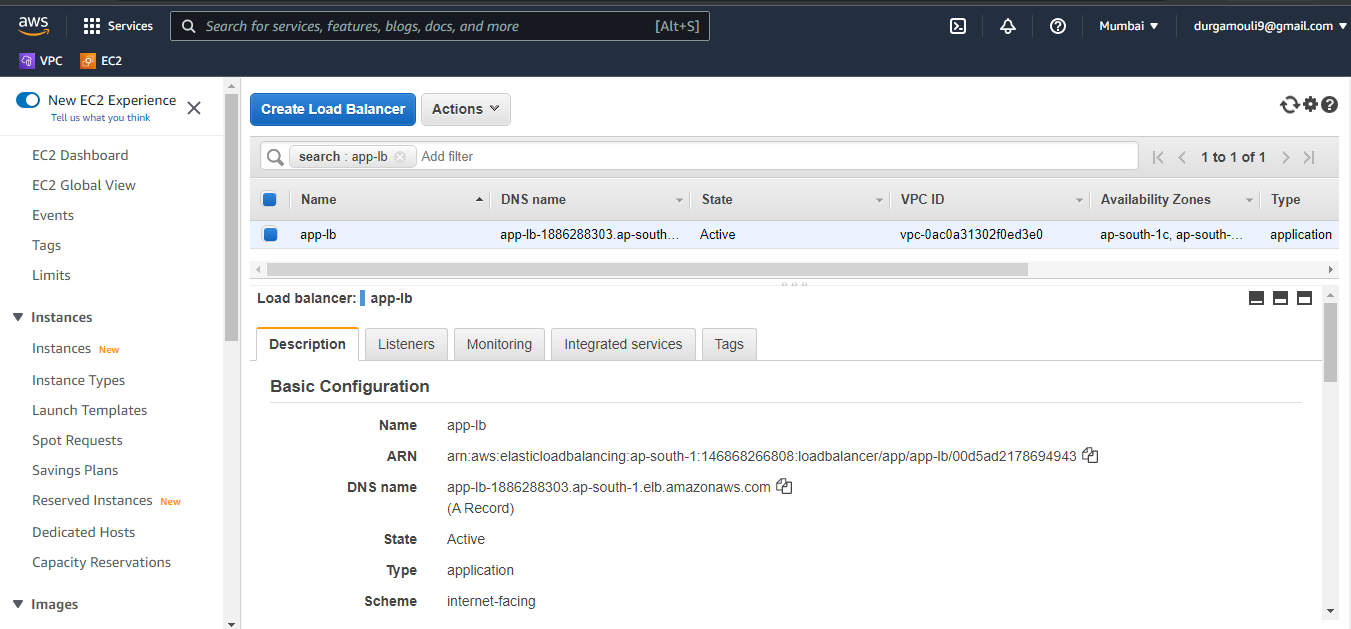




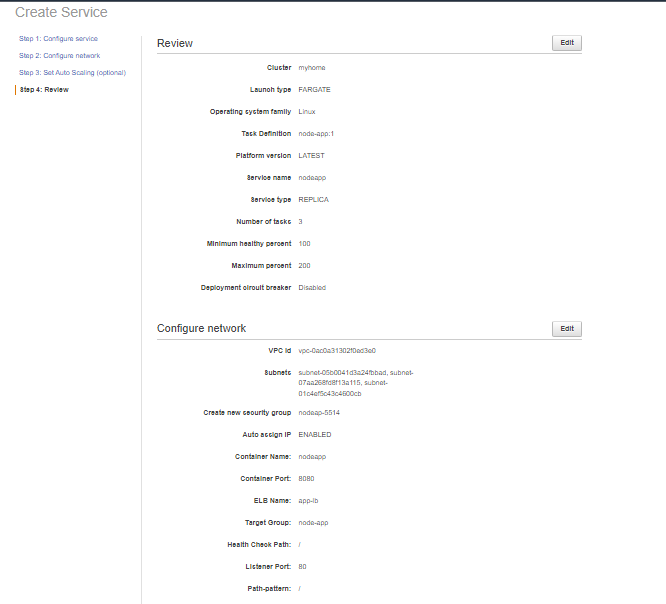
\*\*Using service we will be going to create multiple tasks nothing but replicas\*\*  
  
  


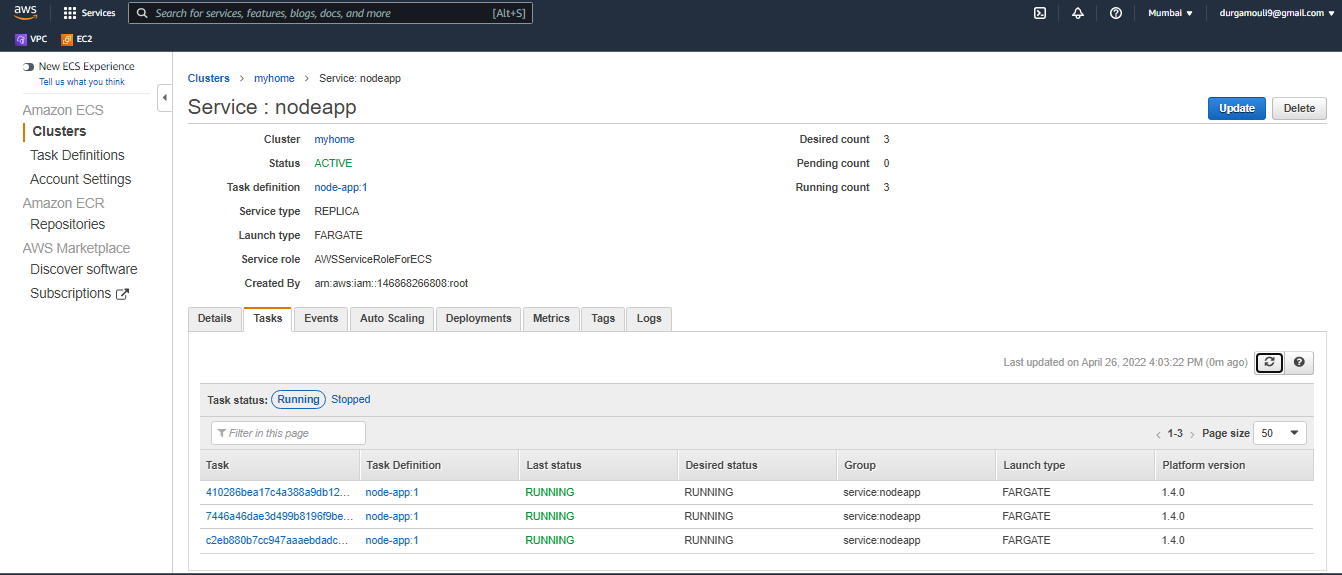
Customer’s hits to application load balancer and alb distributes the traffic across multiple replicas  
  
  
4) Create Target Groups:



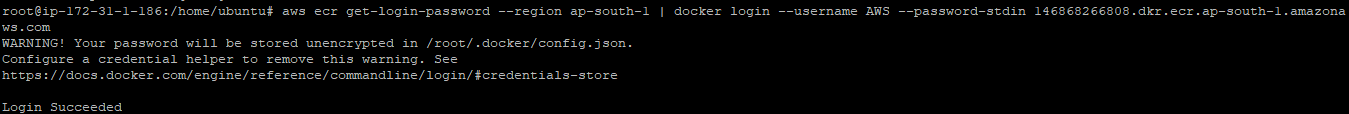
5) Create an auto scaling load balancer:   
  
  


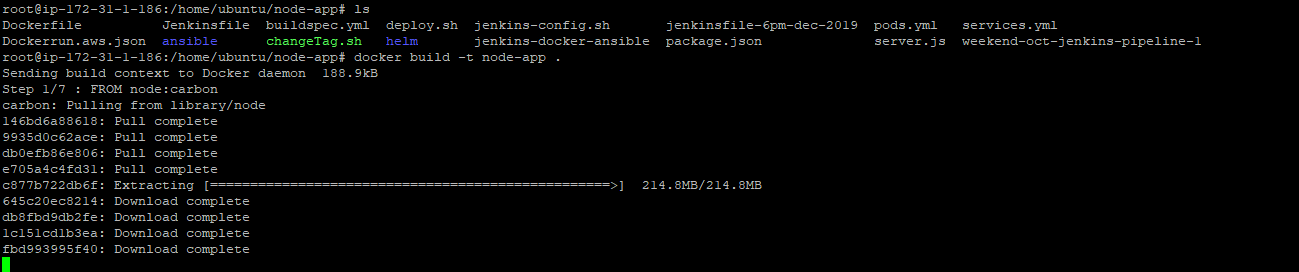
6) Create a service:   
  
service uses task definition to create containers

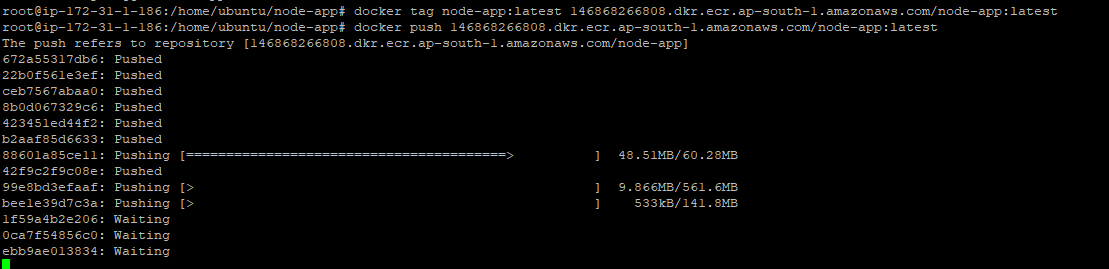


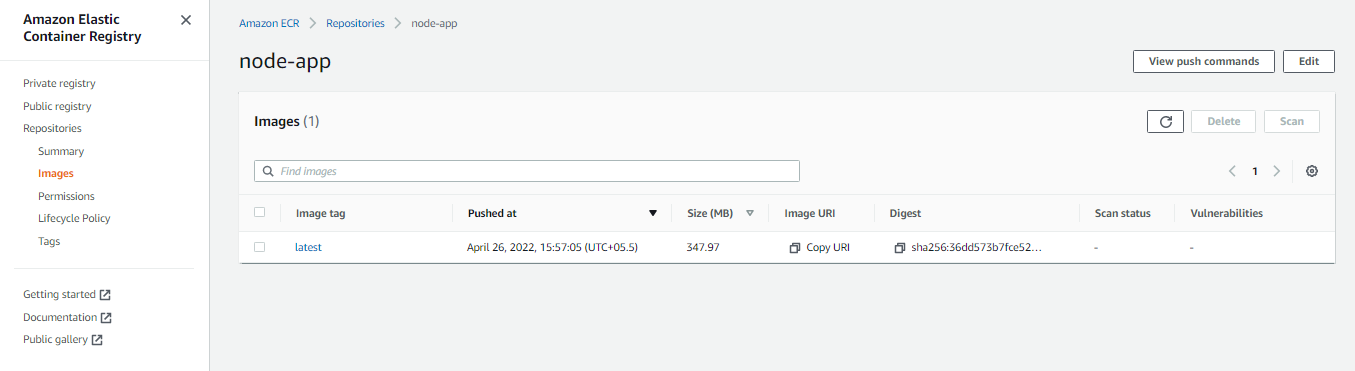


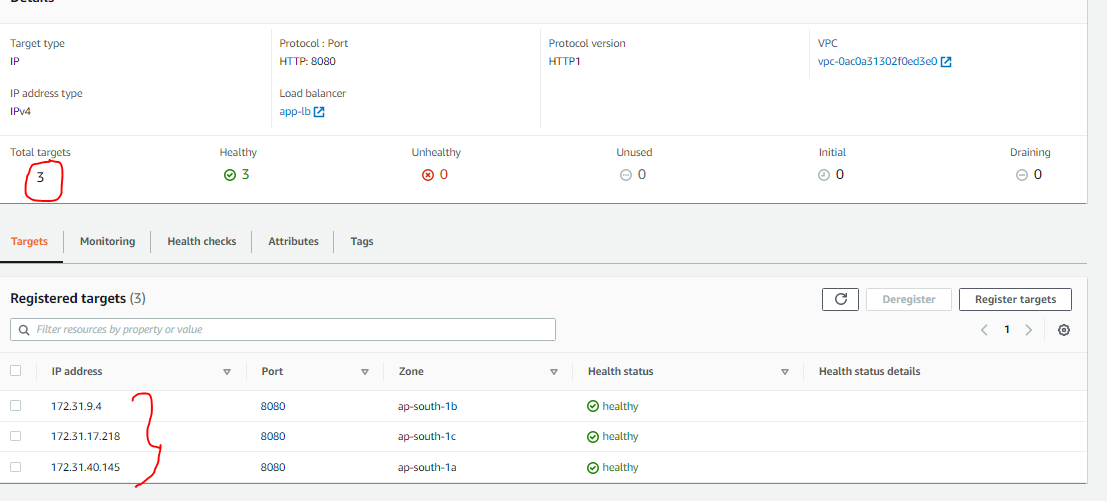
7) Push sample code and create image from EC2 Instance:   
  
Push commands:  
  
\* Login

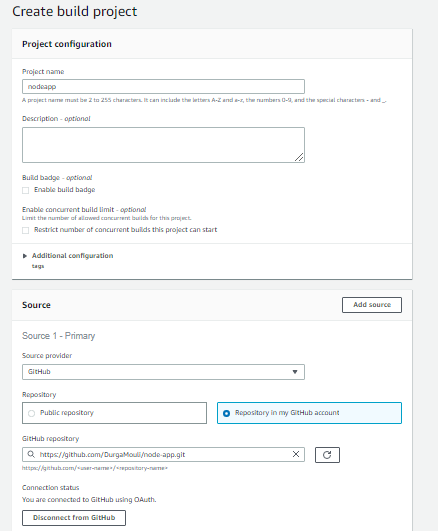


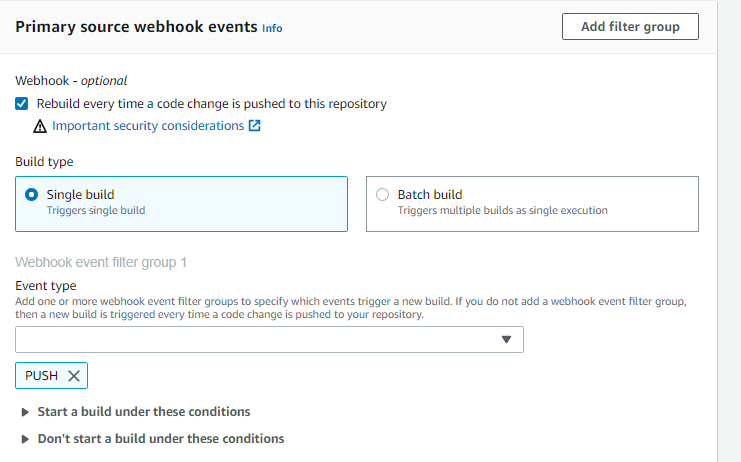
* Building an Docker image  
    
  

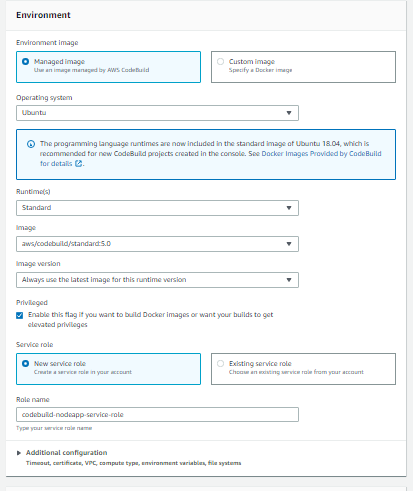
\*Tag & push and image to ECR  
  


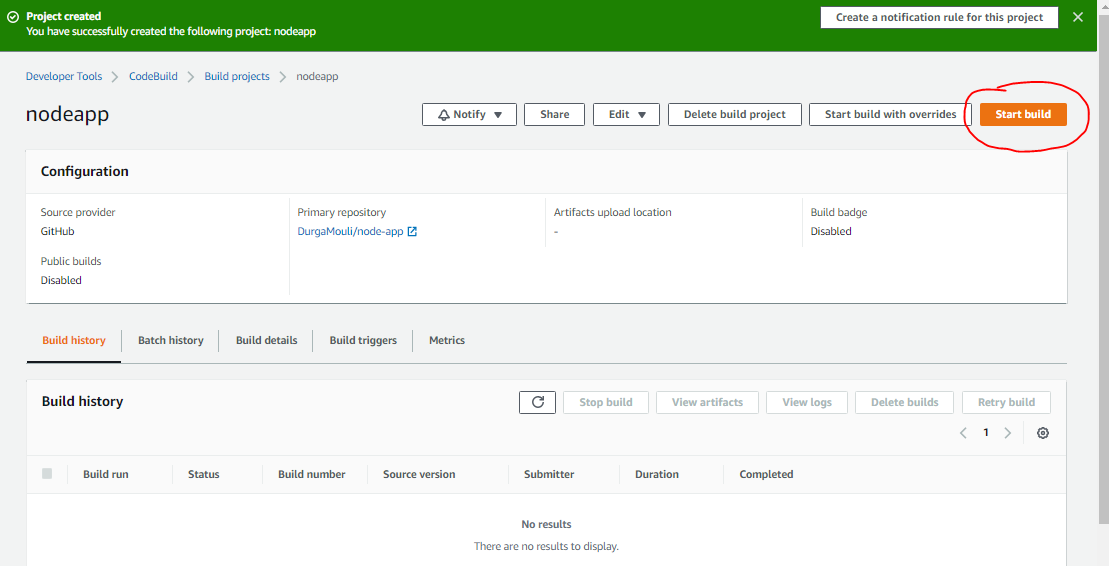


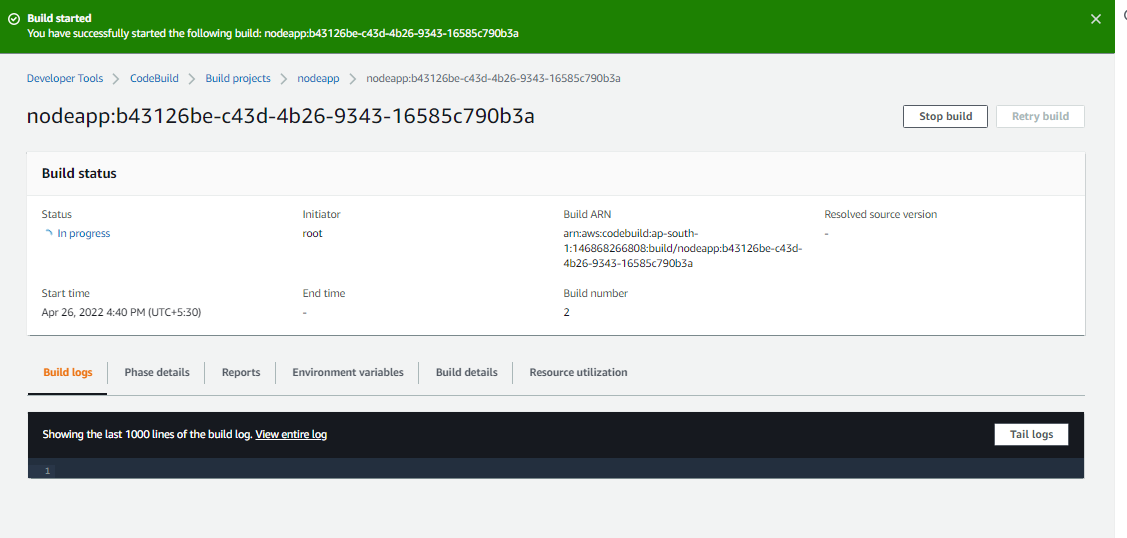
\*\*To cross-check up to now either we have done correct or not \*\*  
  
  
In your targets groups we need to see 3 targets:  
  
  
  
  
\*\*Pick up DNS name from ELB & paste it into browser:

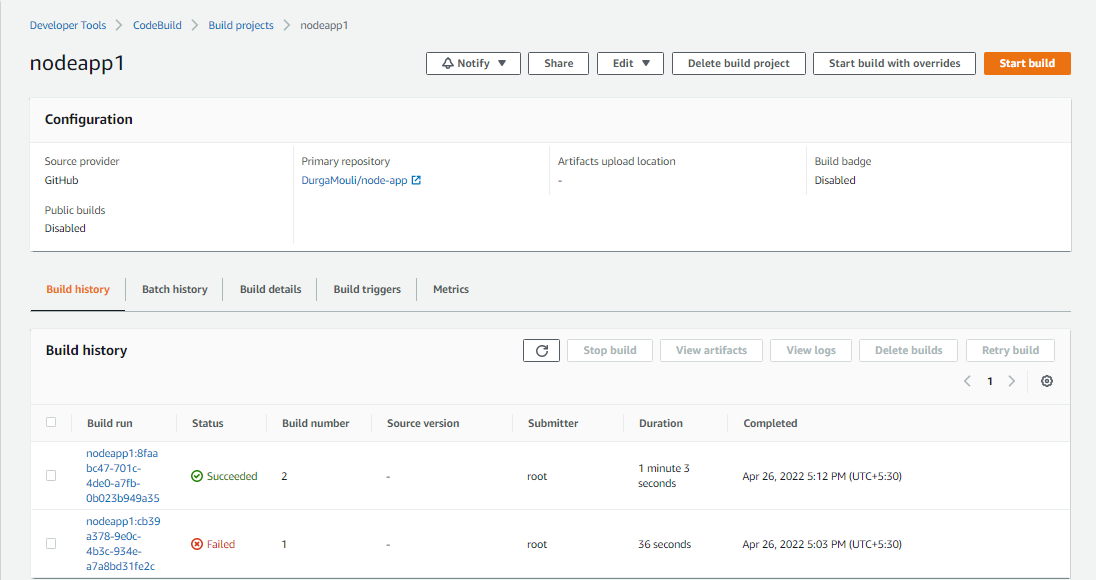
8)Setting up CICD pipeline:  
  
\* Code Build:  
  


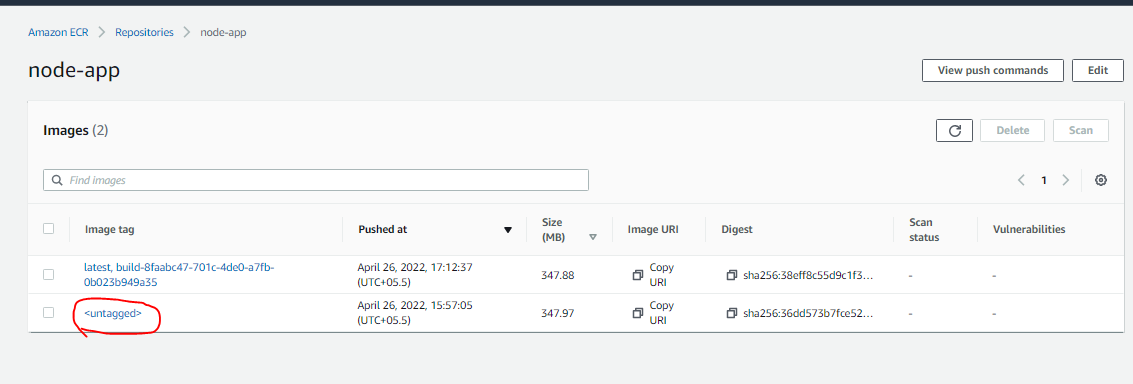
If you need to push automatically once any commit happens in SCM, you need to enable webhook,  
  










New Image has been pushed to ECR Repo:  
  


Next step is to setup code pipeline:

9) Code Pipeline:

