Installation document for Programming Task

Contents

[Objective 2](#_Toc70583263)

[Prerequisite 2](#_Toc70583264)

[Current Design 2](#_Toc70583265)

[Assumptions: 3](#_Toc70583266)

[Build 3](#_Toc70583267)

[Run 12](#_Toc70583268)

[Clean up 18](#_Toc70583269)

[Things which can be Improved 21](#_Toc70583270)

[Troubleshooting 21](#_Toc70583271)

# Objective

Create a docker container/image that performs the following:

  Implements a webserver which accepts requests to manage a PostgreSQL DB

                1. Connects to a PostgreSQL DB

                2. Create a table with columns:  id, timestamp, JSONB

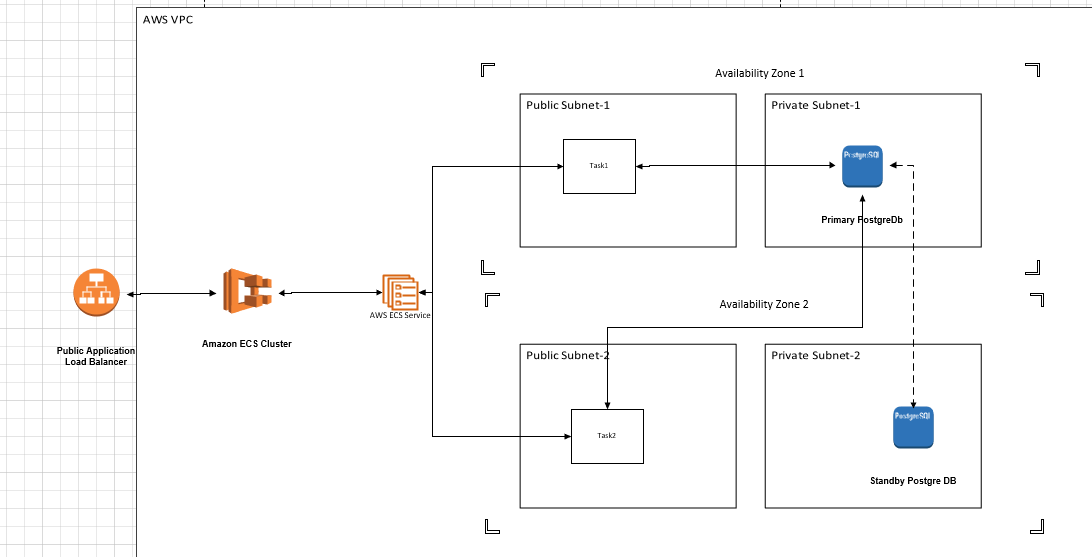
                3. Insert an entry into the table

                5  List the contents of the DB

# Prerequisite

1. Install **aws cli** on your laptop <https://docs.aws.amazon.com/cli/latest/userguide/install-cliv2-windows.html#cliv2-windows-install>
2. Install **python 3.6 or higher** on your laptop <https://www.python.org/downloads/>
3. Install **Docker Desktop** on your laptop <https://docs.docker.com/docker-for-windows/install/>
4. **Personal AWS Account log in Preferred**. We will be cleaning up resources with single click after work is completed.
5. Install Git client <https://git-scm.com/downloads>
6. Windows PowerShell
7. Postman <https://www.postman.com/downloads/>

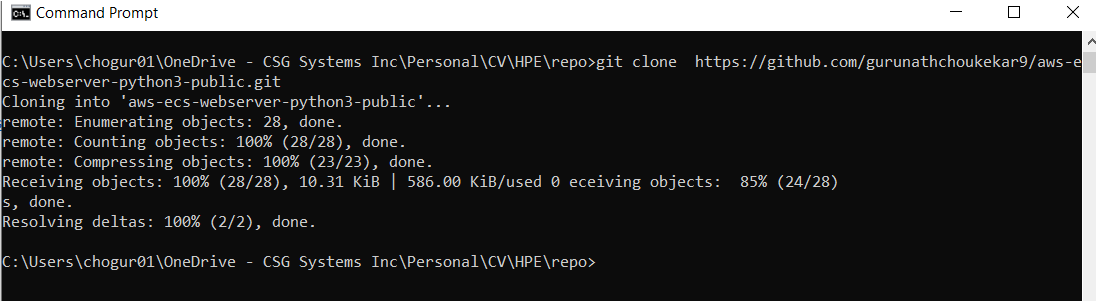
# Current Design



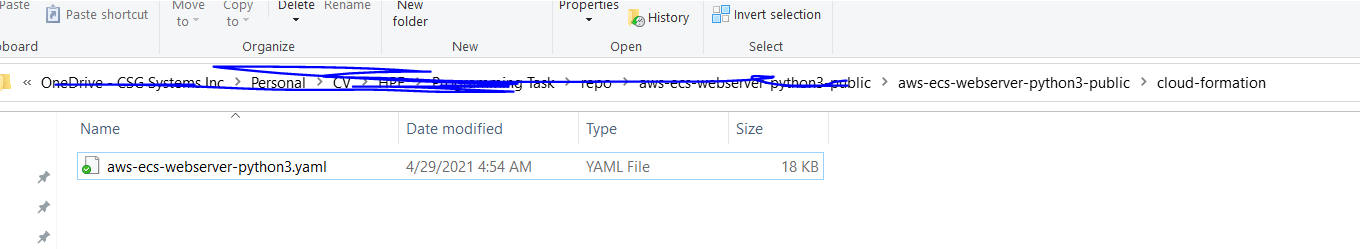
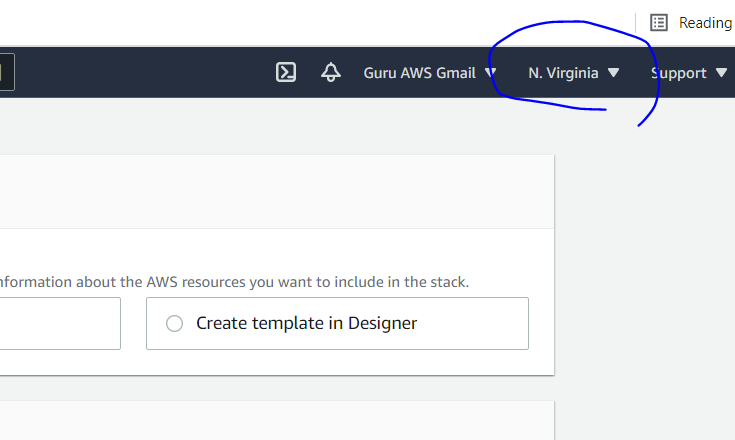
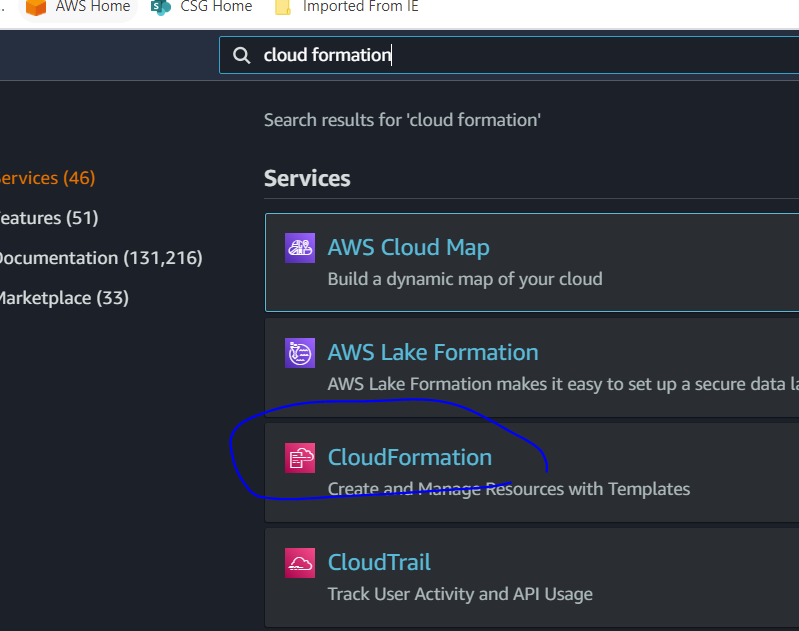
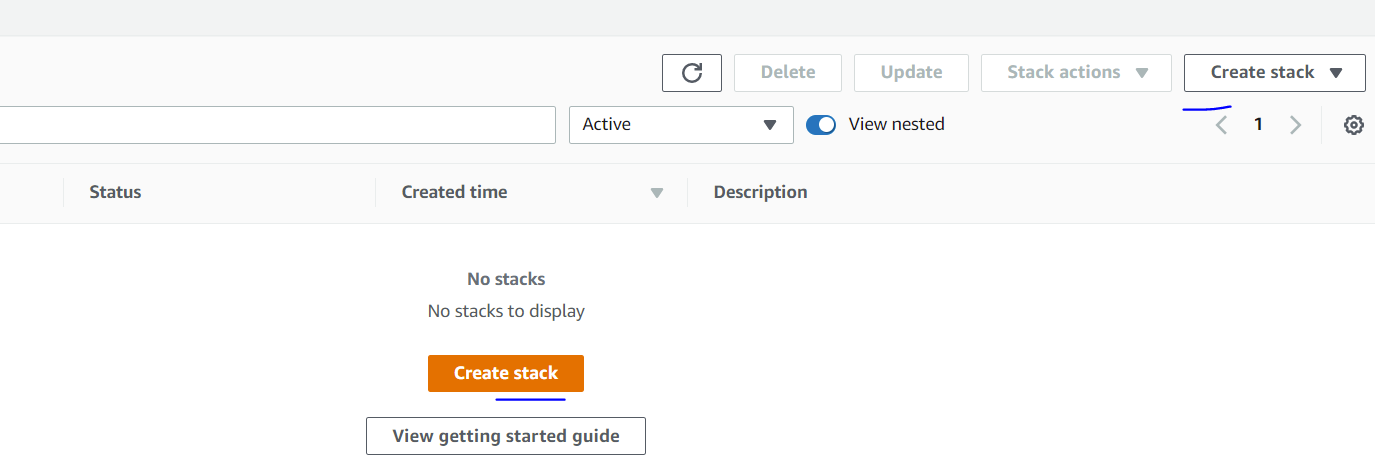
# Assumptions:

1. Prerequisite software installation is completed.
2. **Personal AWS Account log in Preferred**. We will be cleaning up resources with single click after work is completed.
3. **Use AWS Region us-east-1 (N Virginia) ONLY .** Cloud Formation template will not support other region this time

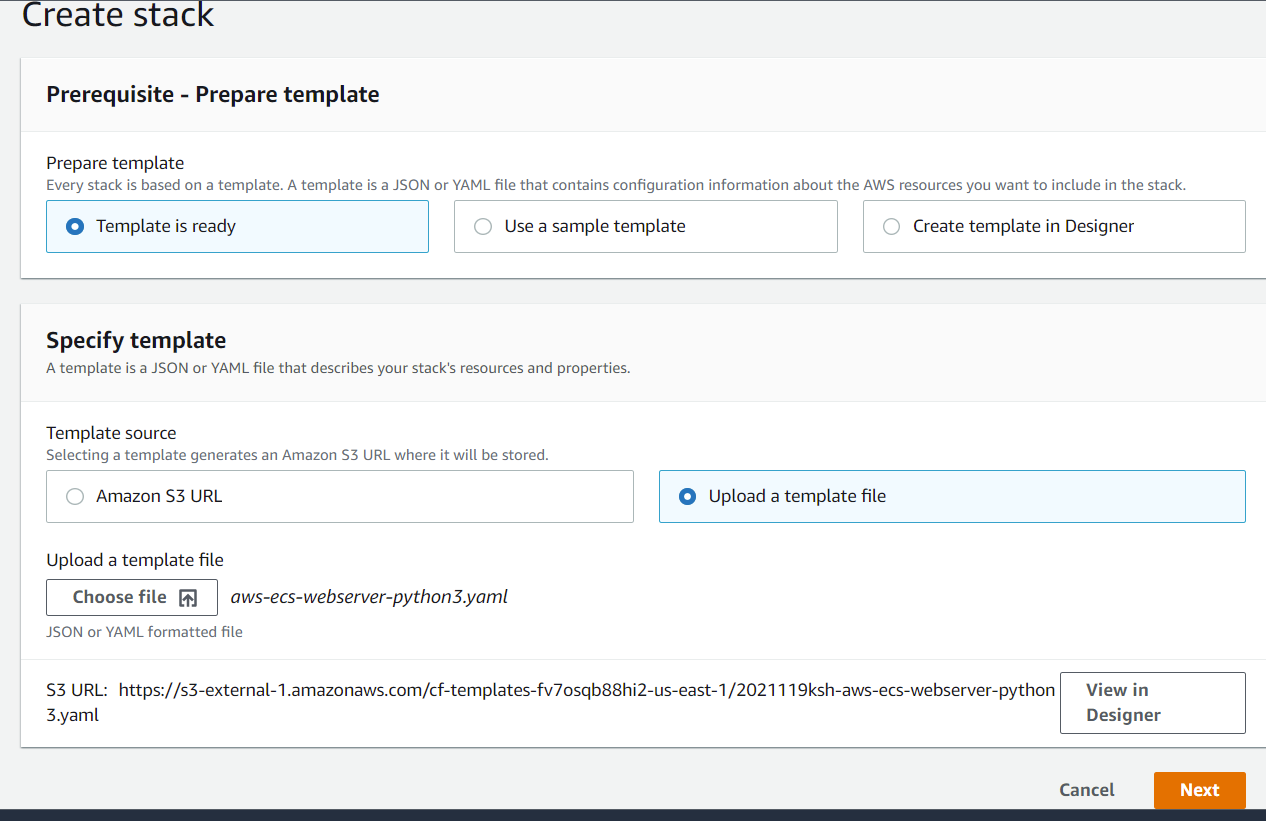
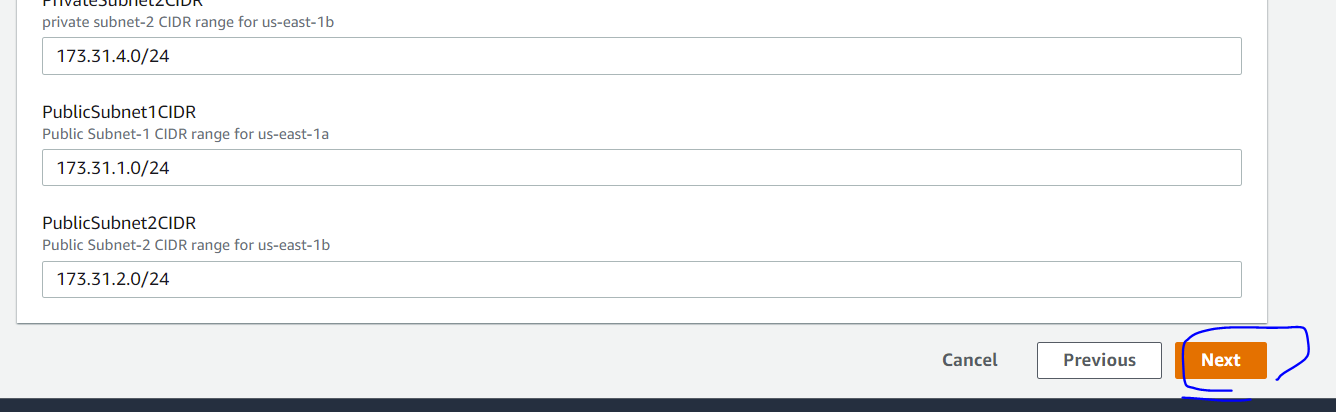
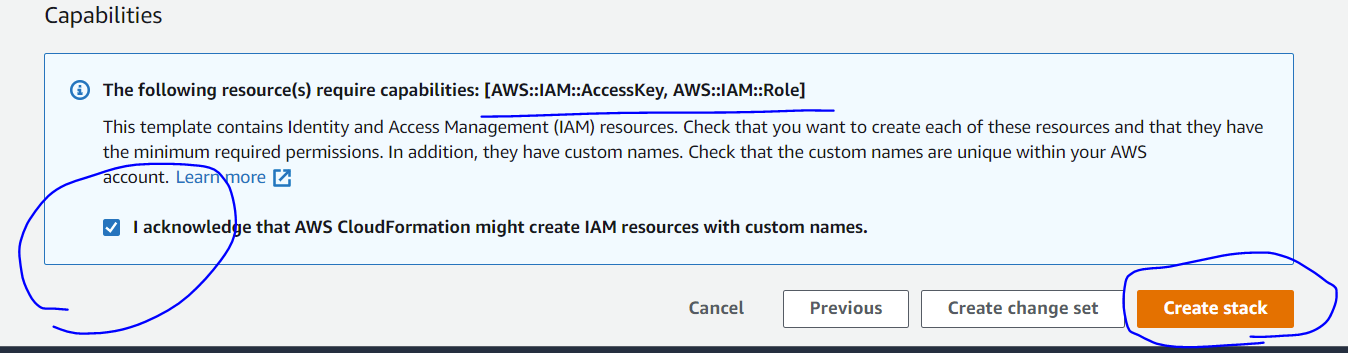
# Build

1. We need to run Cloud formation template from git repo here
2. download git code on your laptop <https://github.com/gurunathchoukekar9/aws-ecs-webserver-python3-public.git>
   1. git clone <https://github.com/gurunathchoukekar9/aws-ecs-webserver-python3-public.git>
   2. 
3. Go to cloud-formation folder

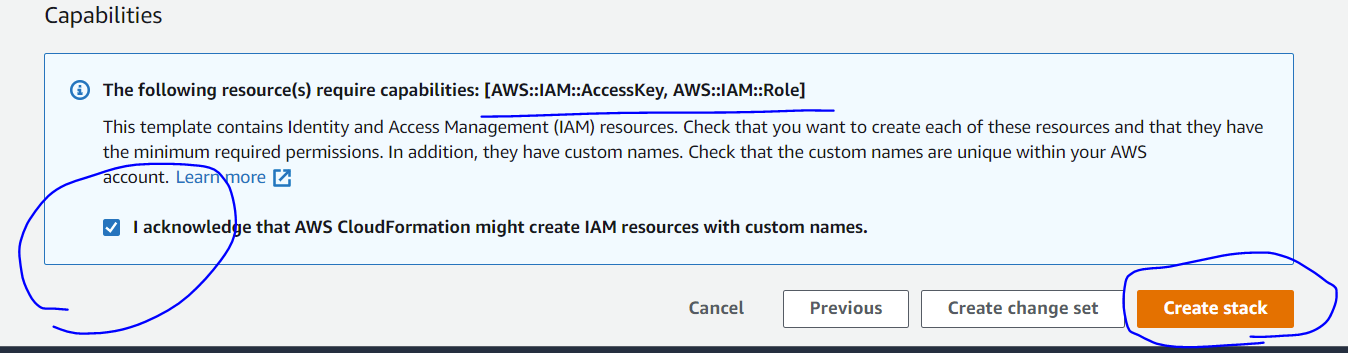
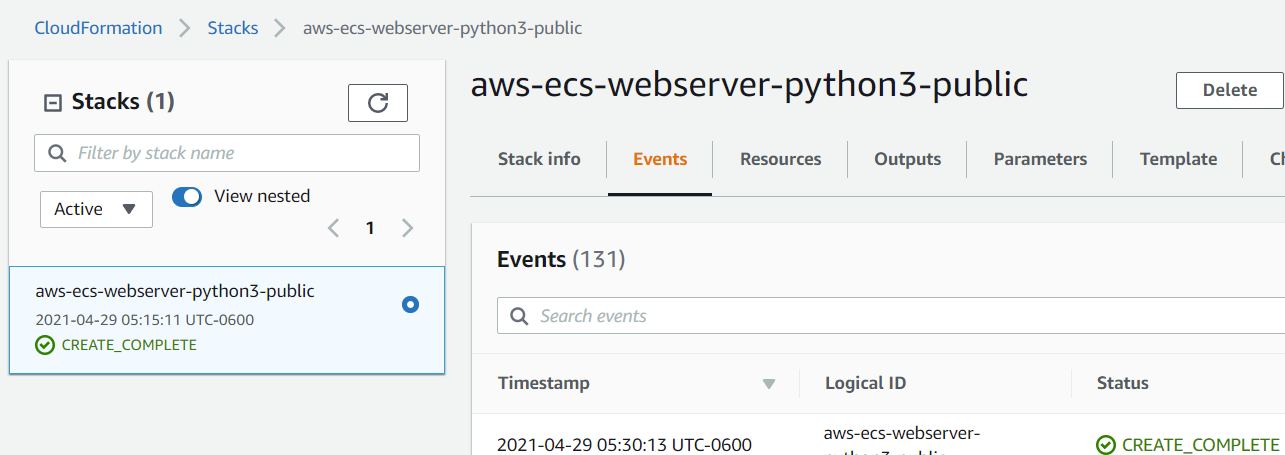
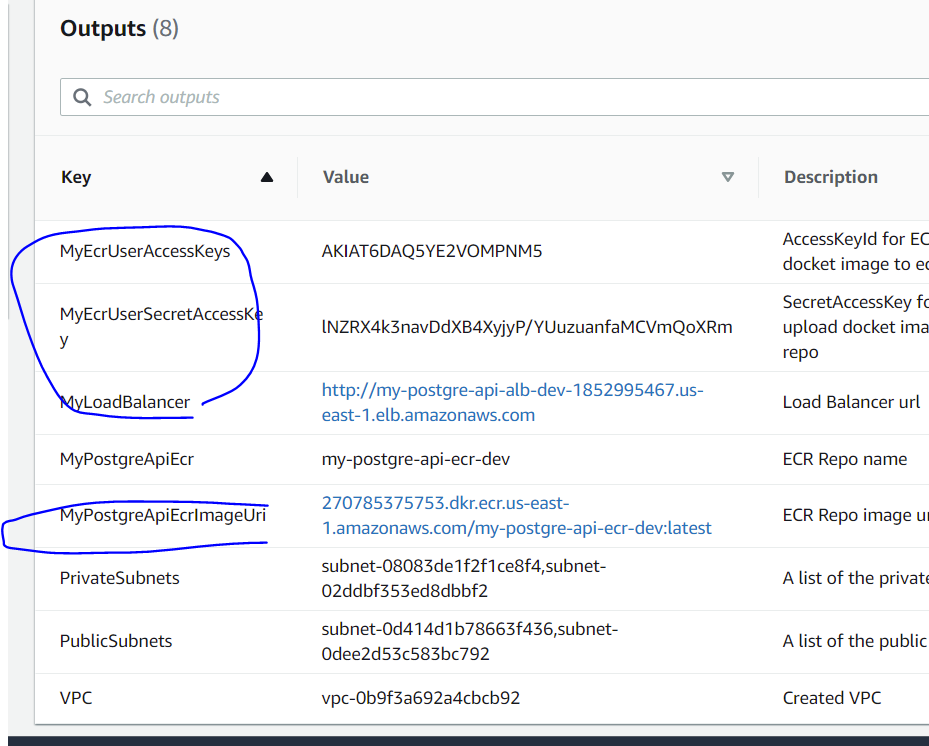
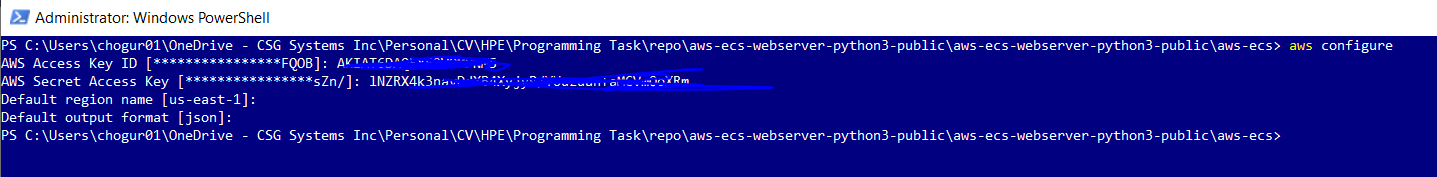
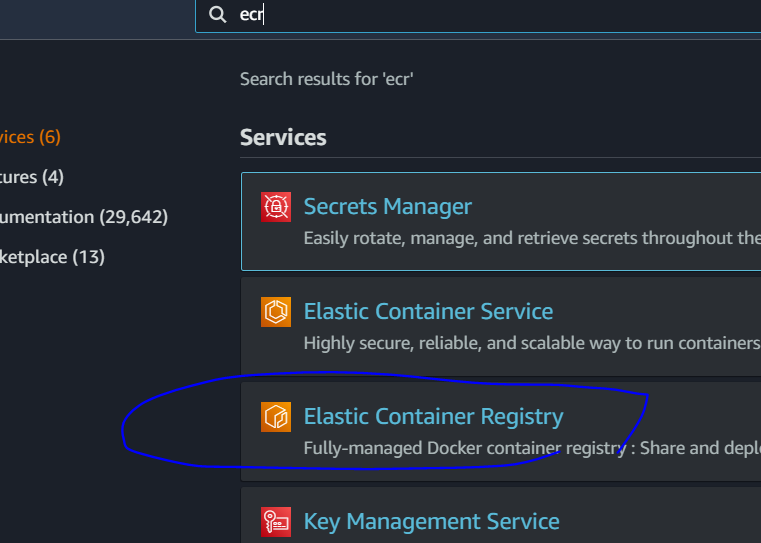
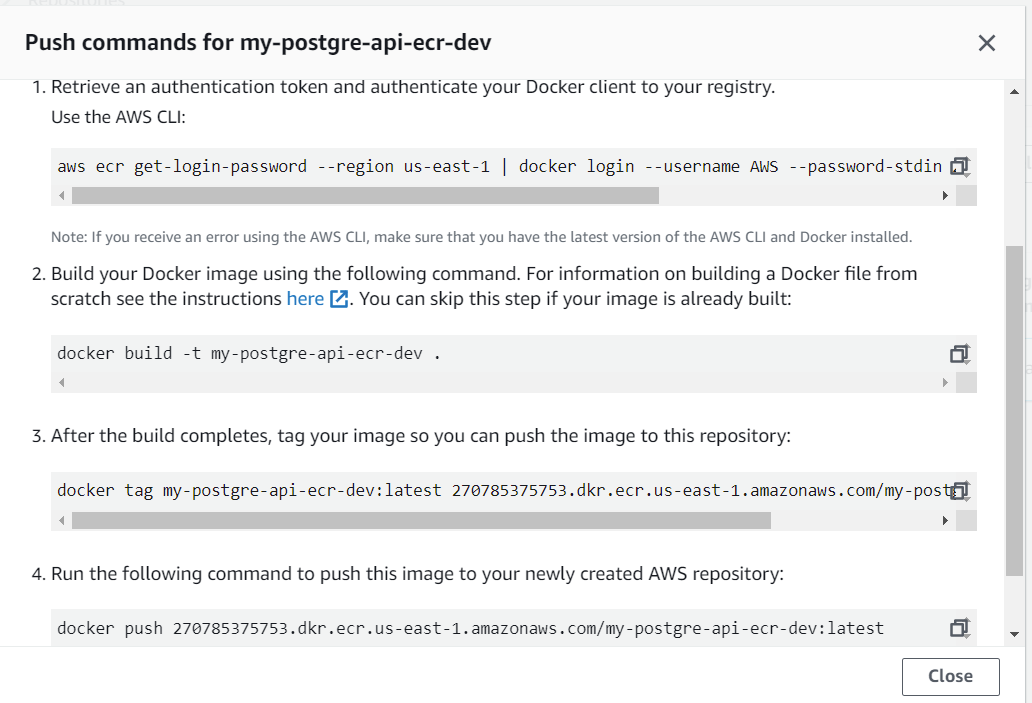
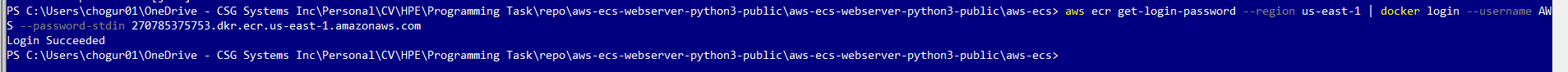
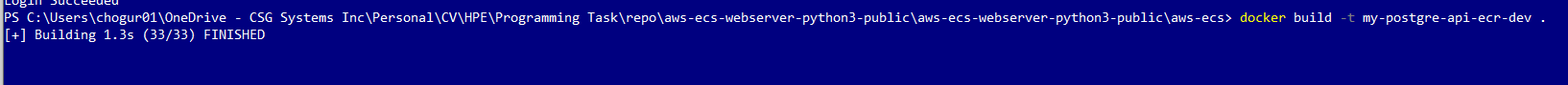
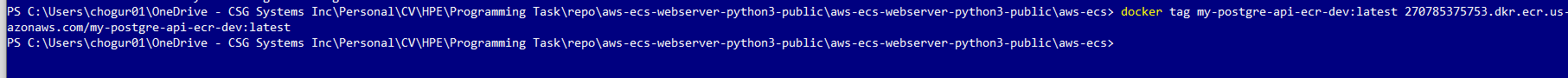
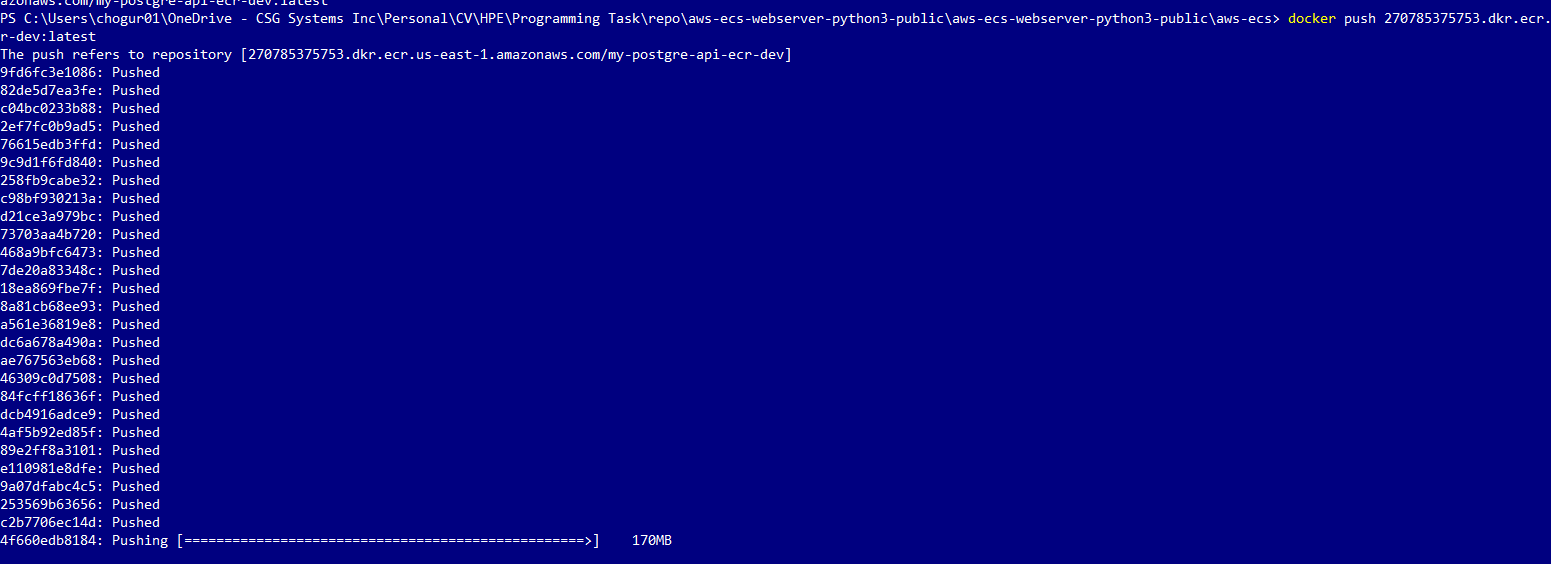
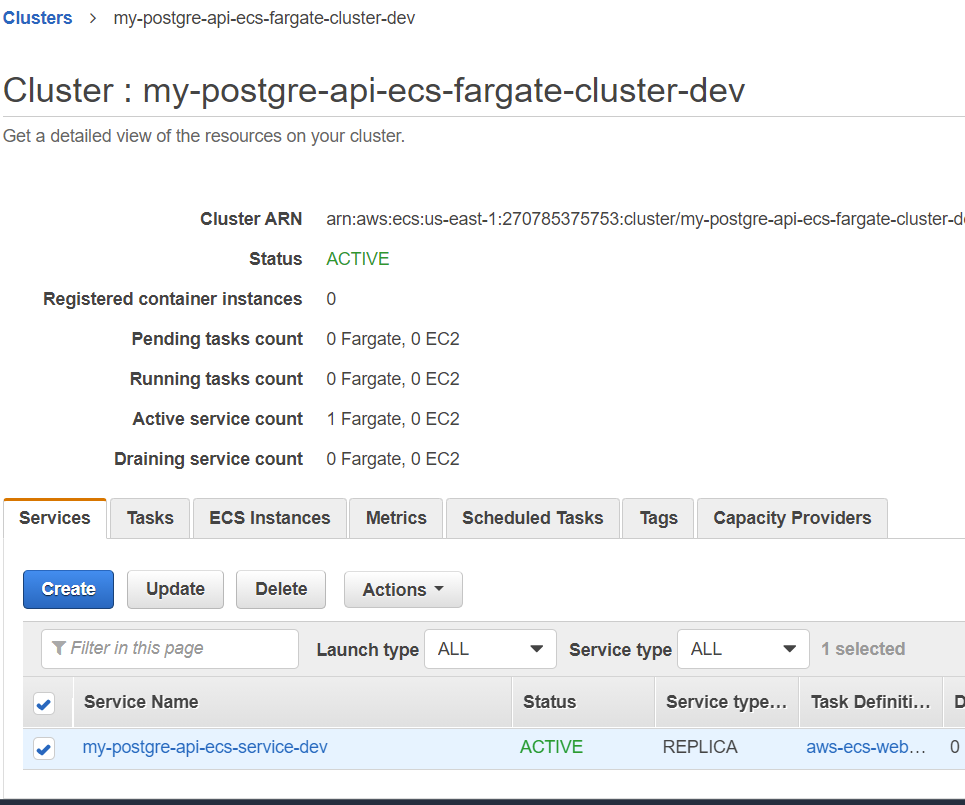
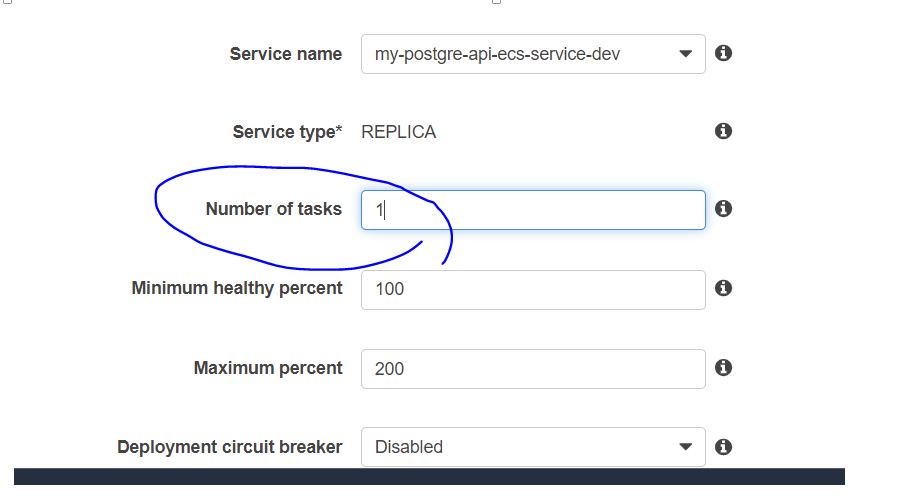
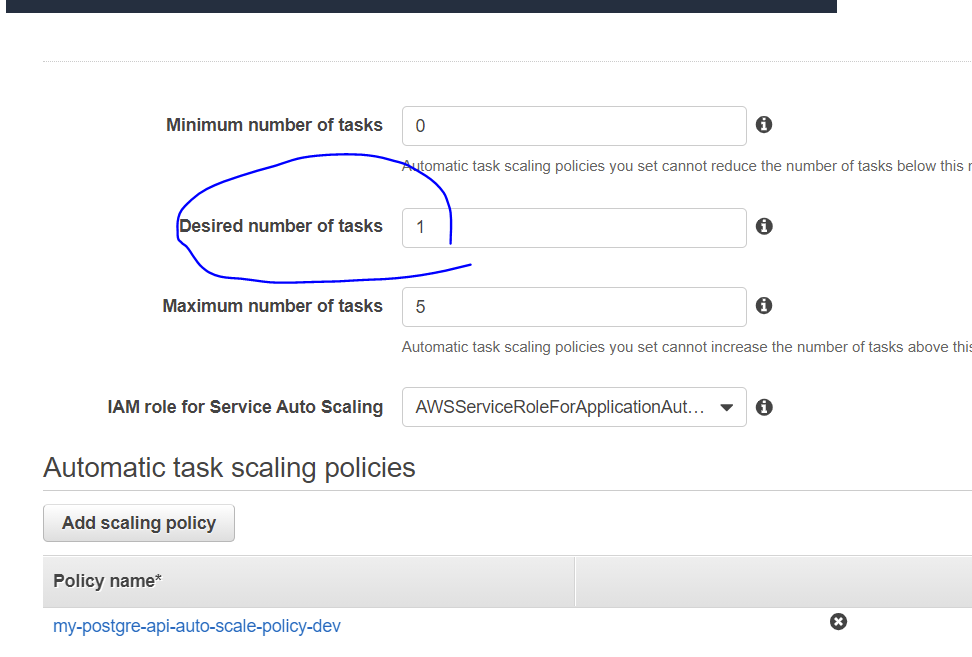
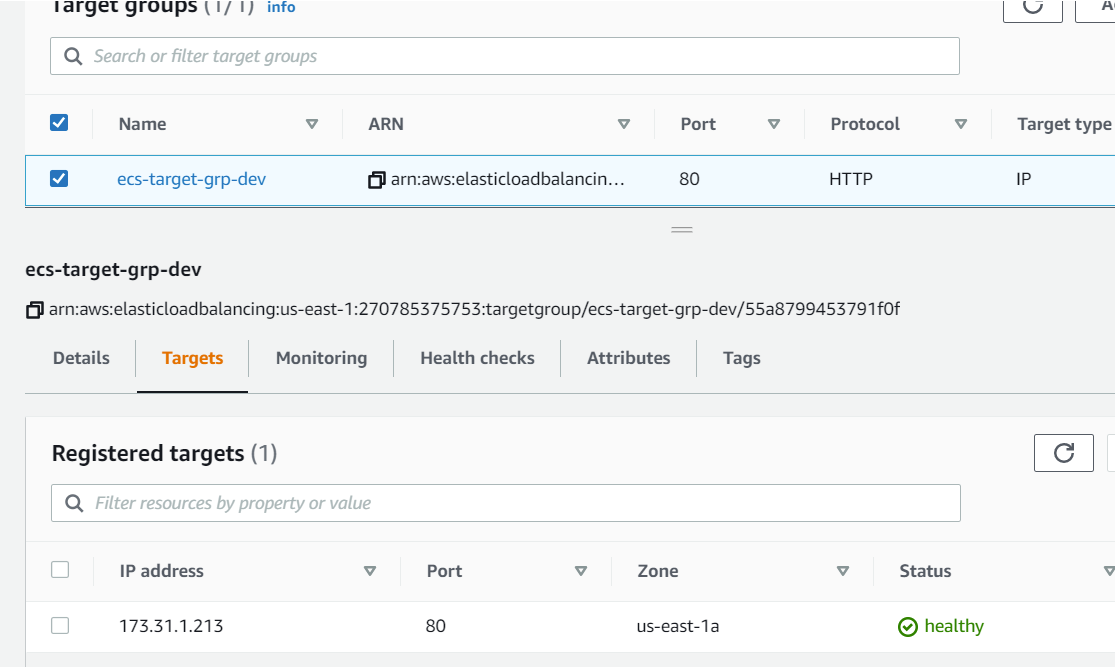
~\aws-ecs-webserver-python3-public\aws-ecs-webserver-python3-public\cloud-formation

1. 
2. Cloud formation template will take care of **resource creation like VPC,PostgreSQL RDS , Task, ECS Cluster, ECR empty repository , ECS Service with ASG , Application Load Balancer**
3. Log on to your AWS Account with us-east-1 (N Virginia) region
4. 
5. Go to Cloud Formation
6. 
7. Create Stack
8. 

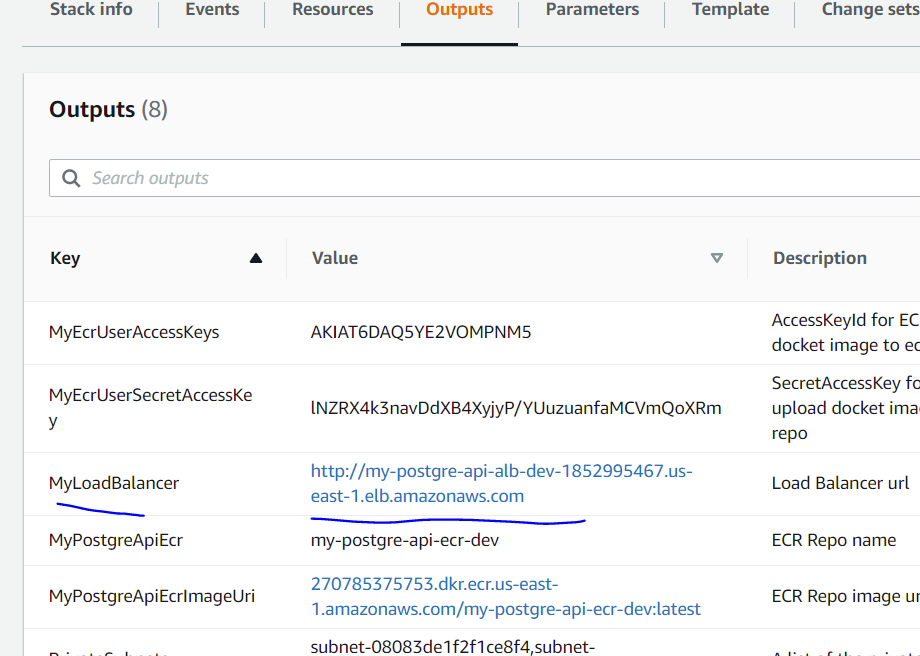
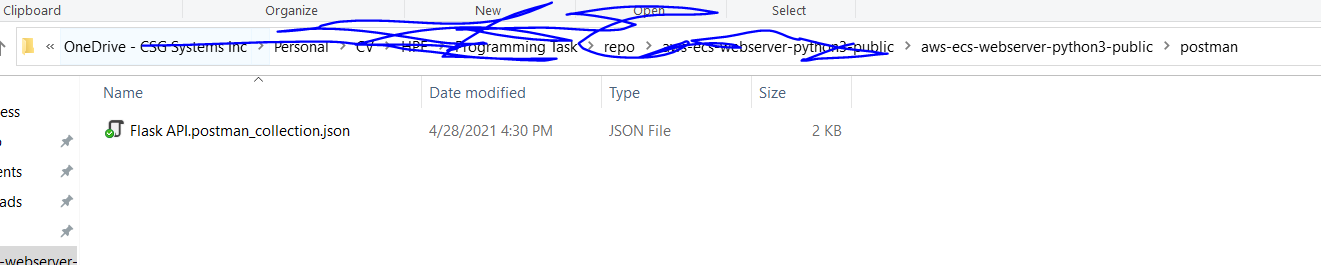
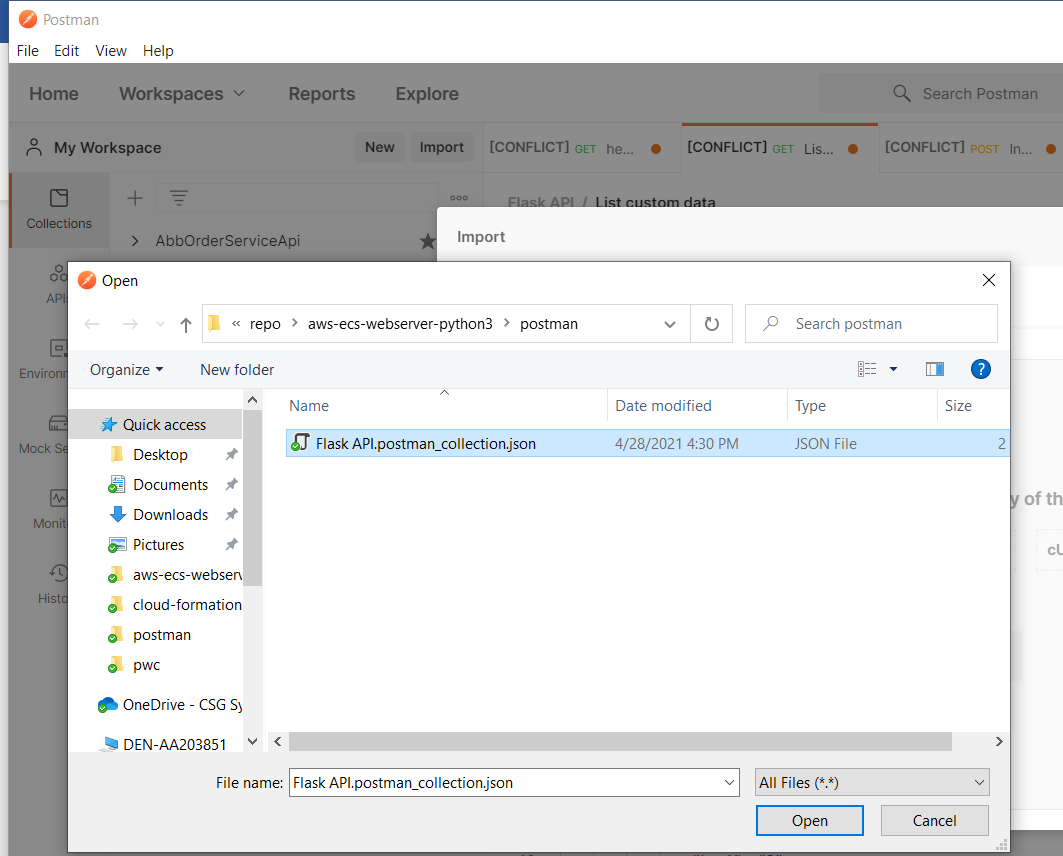
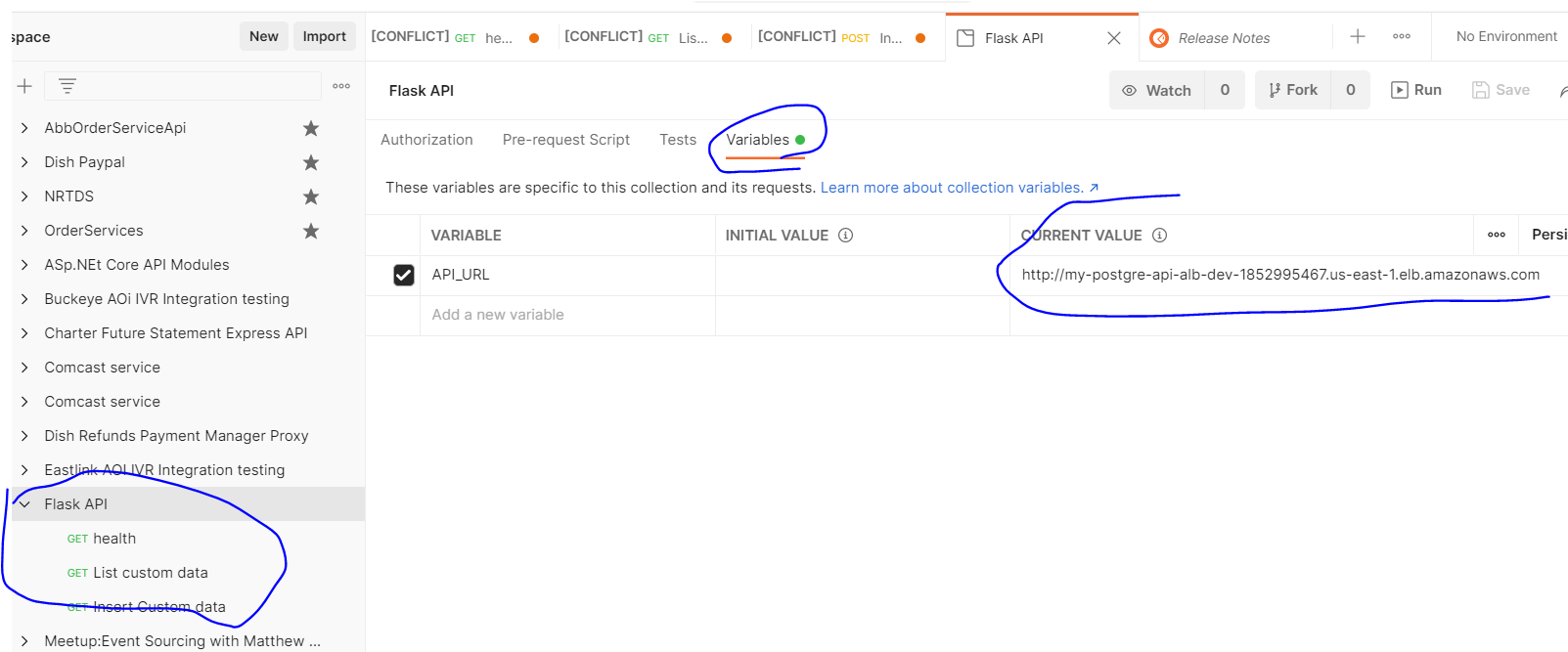
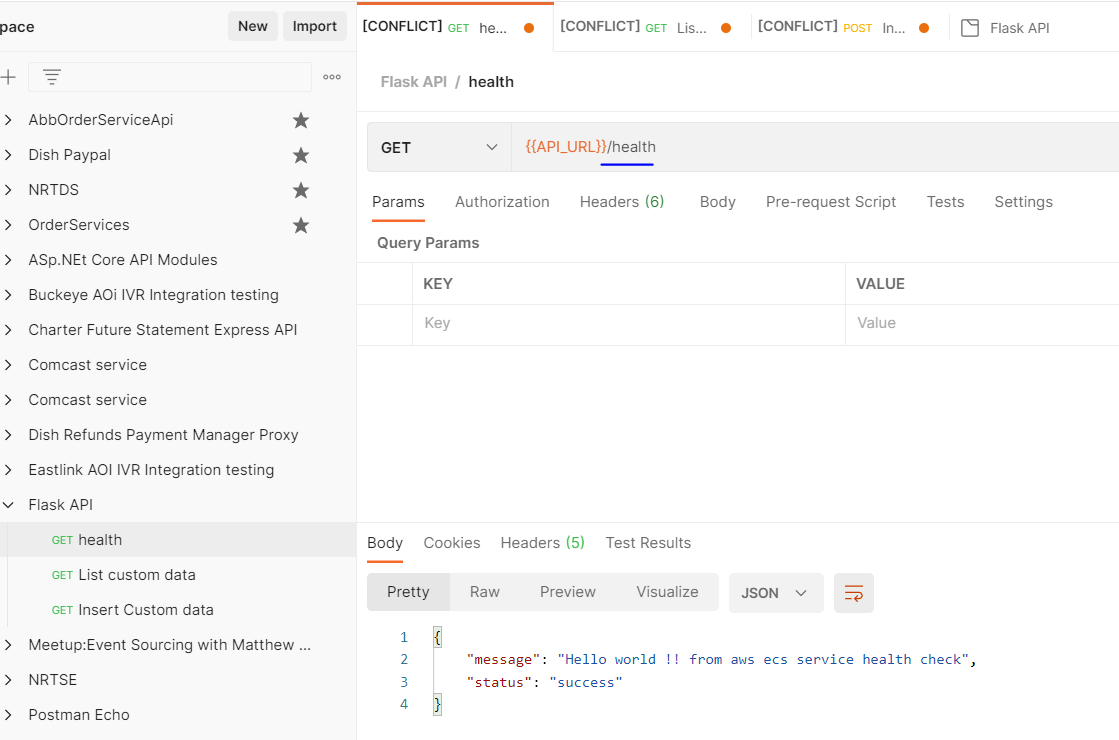
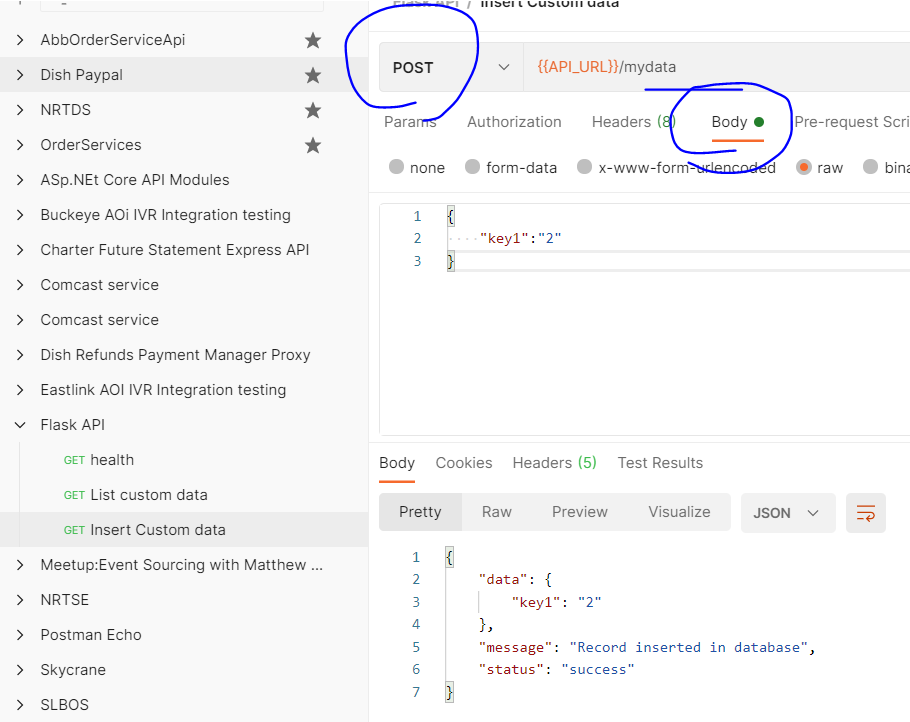
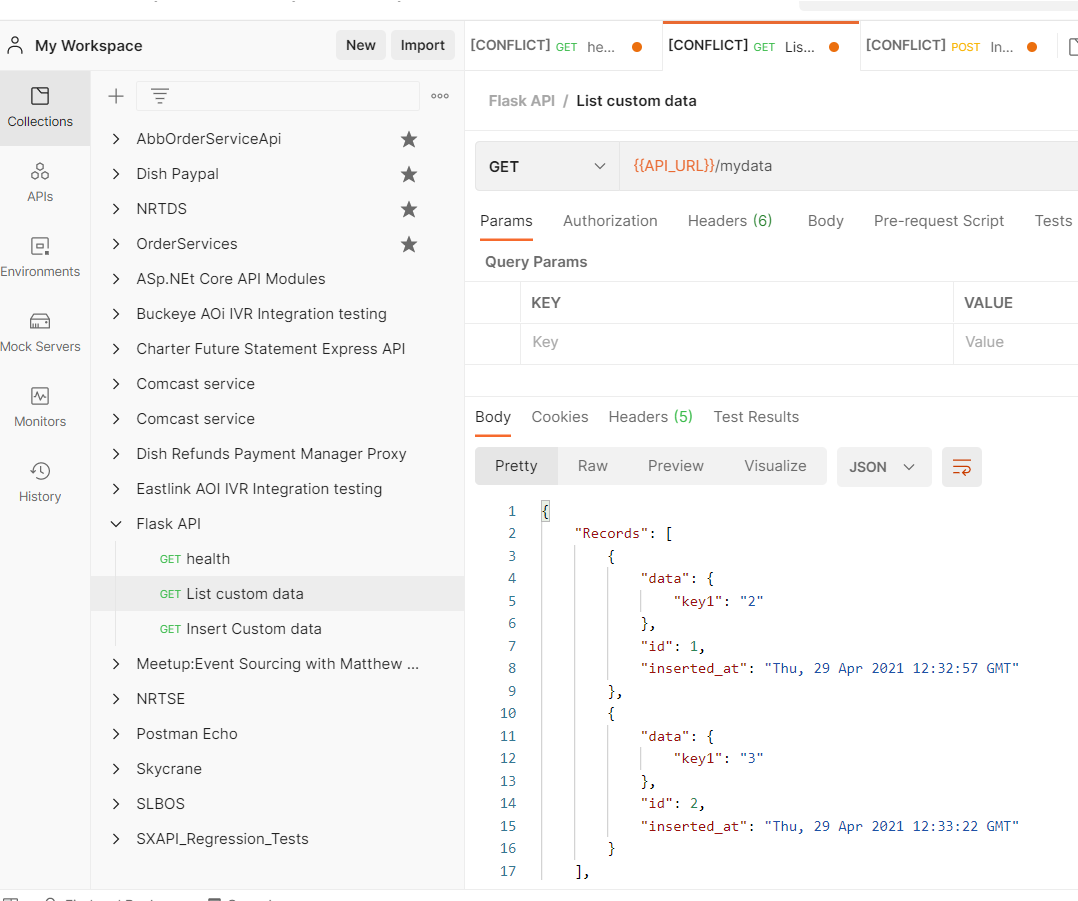
Upload Cloud formation template from downloaded git repo ~\aws-ecs-webserver-python3-public\aws-ecs-webserver-python3-public\cloud-formation

1. 
2. Enter stack name as “**aws-ecs-webserver-python3-public**” or anything you like
3. 
4. Hit Next button
5. Hit Next button again.
6. 
7. Click on Require capabilities [AWS::IAM::AccessKey, AWS::IAM::Role]

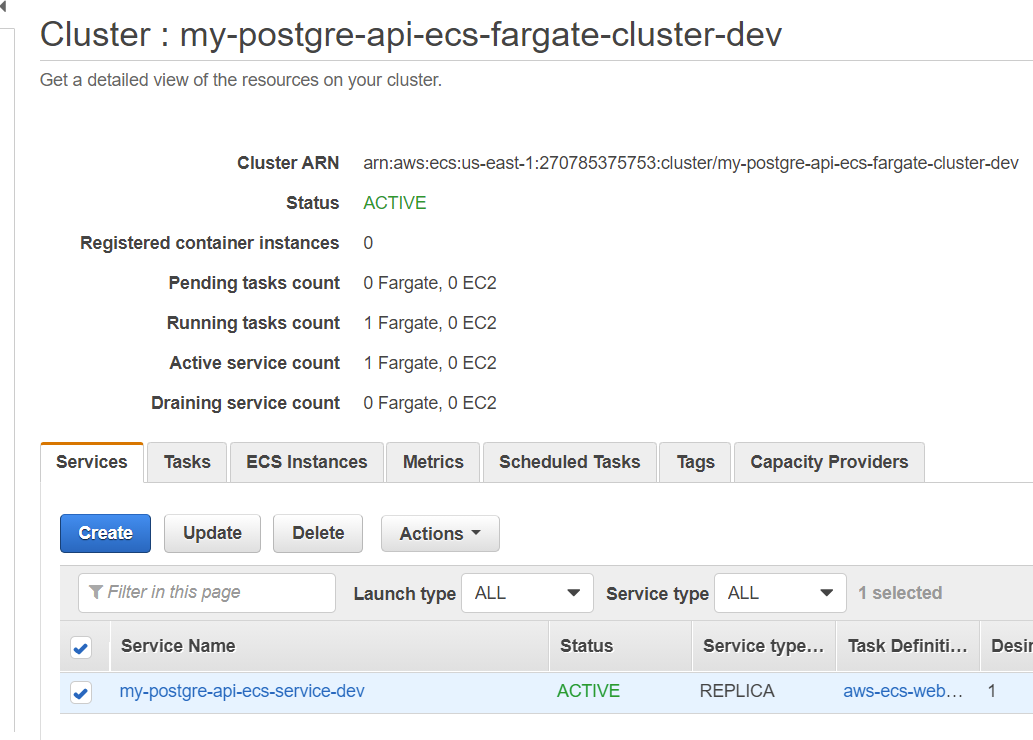
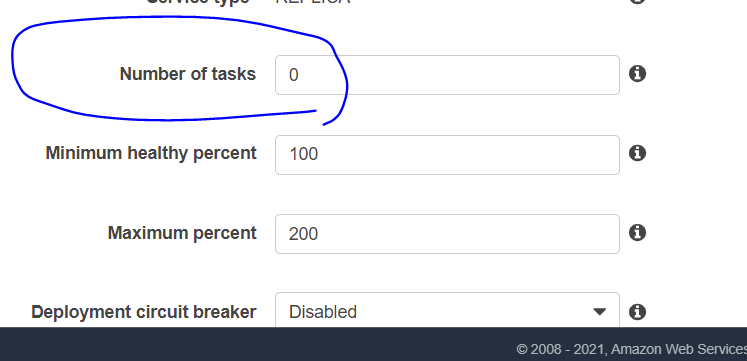
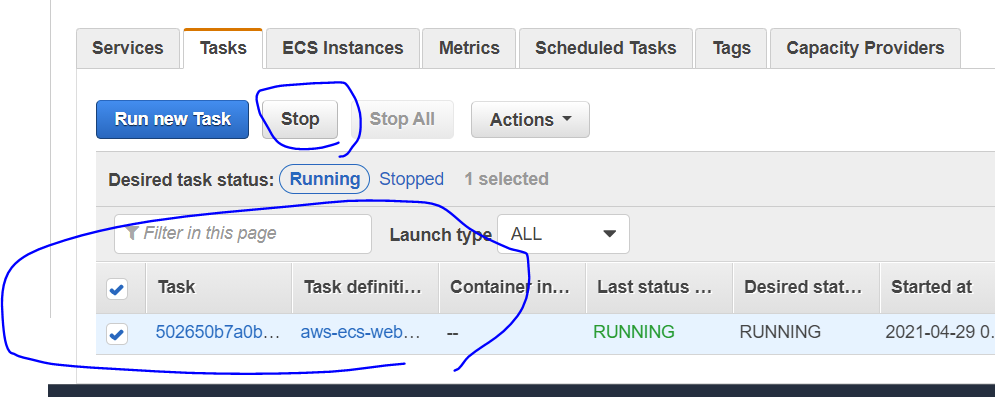
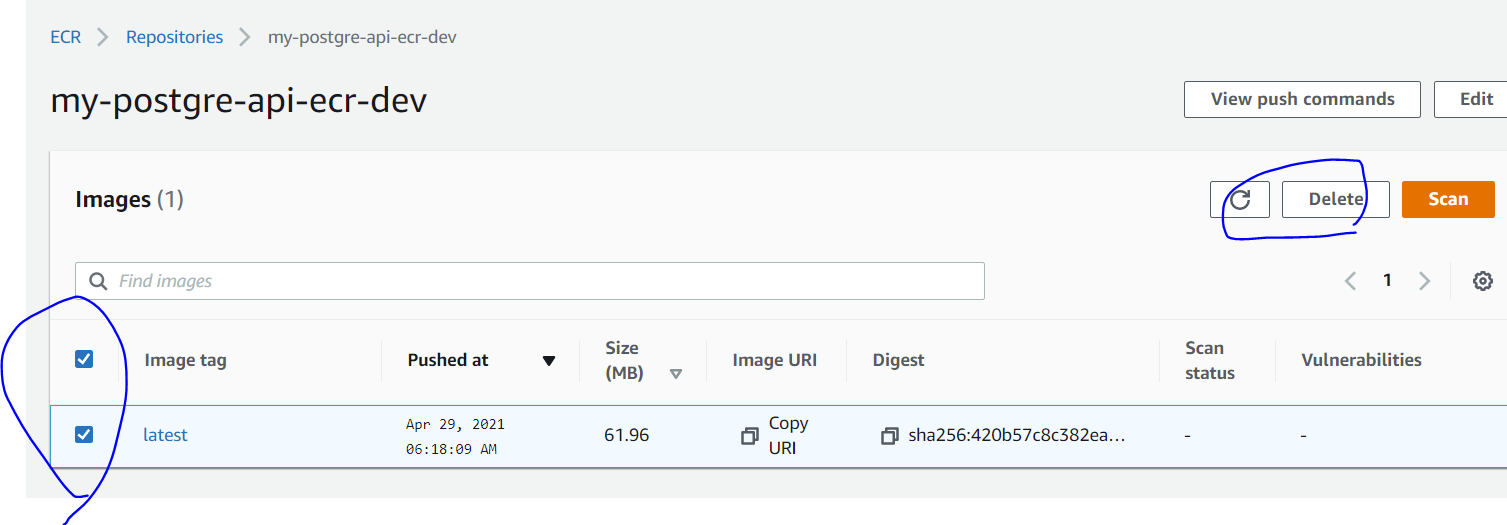
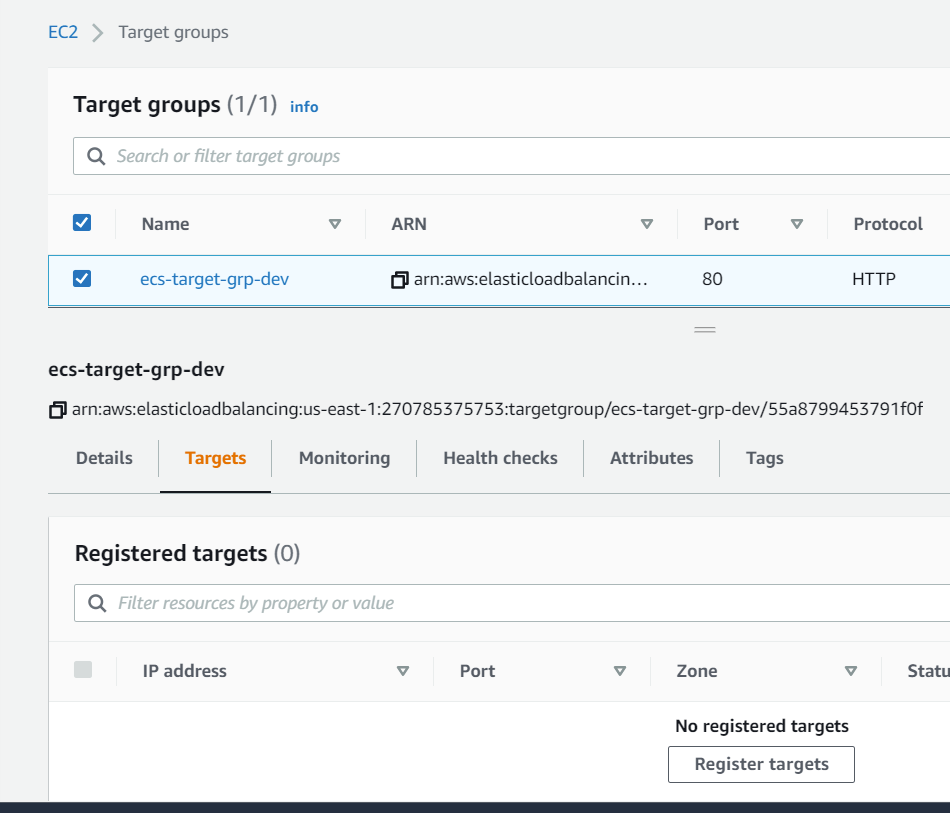
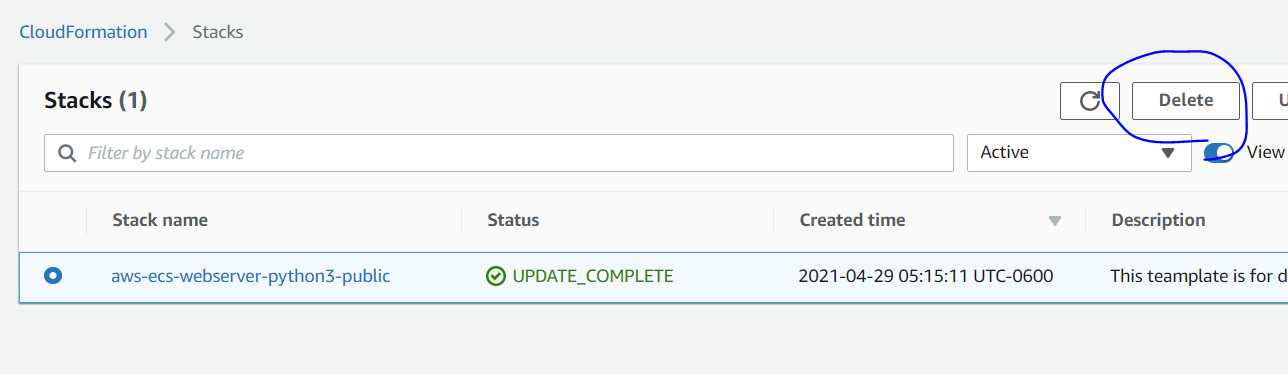
Internally Cloud formation is creating IAM user for ECR image upload. With no other access

1. ****
2. Cloud formation template will take **around 15 min to Complete.**
3. 
4. After Cloud Formation is completed go to Outputs tab
5. And Note down **MyEcrUserAccessKeys , MyEcrUserSecretAccessKey , MyLoadBalancer ,** **MyPostgreApiEcr , MyPostgreApiEcrImageUri** output values
6. 
7. Open Windows powershell in run as amin mode and go to git repo folde ~\ aws-ecs-webserver-python3-public\aws-ecs
8. Perform **aws configure** with **MyEcrUserAccessKeys , MyEcrUserSecretAccessKey** received from Cloud formation output variables
9. 
10. Make sure Docker Desktop is running on your laptop . we need to build and upload docker image to ECR
11. Go to AWS ECR
12. 
13. Get the push command from ECR for repo **MyPostgreApiEcr = my-postgre-api-ecr-dev**
14. 
15. Open Windows powershell in run as amin mode and go to git repo folder ~\ aws-ecs-webserver-python3-public\aws-ecs
16. Log on to aws ecr
    1. aws ecr get-login-password --region us-east-1 | docker login --username AWS --password-stdin {your\_aws\_account\_id}.dkr.ecr.us-east-1.amazonaws.com
17. 
18. Build container image.
    1. docker build -t my-postgre-api-ecr-dev .
    2. 
19. Tag container image
    1. docker tag my-postgre-api-ecr-dev:latest {your\_aws\_account\_id}.dkr.ecr.us-east-1.amazonaws.com/my-postgre-api-ecr-dev:latest
    2. 
20. Push container image to aws ecr
    1. docker push your\_aws\_account\_id}.dkr.ecr.us-east-1.amazonaws.com/my-postgre-api-ecr-dev:latest
    2. 
    3. This will take around 5 min to upload 170MB docker image to aws ecr
21. Go to AWS ECS 🡪 Update Service 🡪 **we need to update Number of Tasks from 0 to** 1
    1. 
    2. 
    3. 
22. Go to AWS EC2 🡪 Target Groups 🡪 ecs-target-grp-dev
    1. 
    2. You will see Service Tasks getting recognized soon with “healthy” green color
23. Next steps are in “Run” section of this document

# Run

1. Go to Cloud formation Outputs and get **MyLoadBalancer** uri which we can use it in postman project provided in git repo
   1. Ex 🡪 http://my-postgre-api-alb-dev-{your\_aws\_account\_id}.us-east-1.elb.amazonaws.com
2. 
3. Go to downloaded git repo ~\postman
4. 
5. Open Postman software 🡪 File 🡪 Import🡪 provide above postman collection json file
6. 
7. Update APP\_URL in postman project variable with **MyLoadBalancer** url from cloud formation outputs
8. 
9. Run **/health** GET check to make sure Application load balancer url is returning properly
10. 
11. Run **/mydata POST** to insert some json input into PostgreSQL database
12. 
13. Run **/mydata GET** to get list of records from PostgreSQL database
14. 
15. Behind the scene service will create PostgreSQL database table if not exists for the first time

# Clean up

1. To Clean up we first have to **update AWS ECS Service Task count from 1 to 0**
2. 
3. 
4. Stop running AWS Tasks manually by going into Tasks tab **STOP button**
5. 
6. Go to AWS ECR 🡪 open repo “my-postgre-api-ecr-dev”
7. **DELETE All container images inside AWS ECR** repo
8. 
9. Go to Aws EC2🡪 Target Groups 🡪 **And make sure Registered Targets are empty . IF it is Draining you need to wait**
10. 
11. Go to AWS Cloud formation and **Delete Stack**
12. 
13. Delete stack will delete all resources which we created for this programming task
14. It may take around 10 min to complete the run

# Things which can be Improved

1. Swagger Documentation with Auto update rather than manual addition of request and response
2. Architecture from current Application Load balancer exposed to world to use VPC endpoint for private Load balancer and then use Aws API Gateway
3. In this way we can add authentication on AWS API gateway API Keys/ Lambda Authorizer etc
4. Use AWS Code pipeline so that when developer check in new code . AWS Code pipeline can be run to build and deploy new image to AWS ECR
5. Unit tests and Integration Tests
6. Add password in SSM Parameter store for database.
7. Add SNS topic notification in case of Task failure.
8. Add PostgreDB read replica so that GET APIs can refer that Read replica

# Troubleshooting

1. Delete Cloud formation stack failed for some resources.
2. Go back and try Delete Stack one more time. This may happen if Cleanup steps above are skipped.