Microservices (ECS) & AWS tools based, Complete DevOps CICD Pipeline, step by step Tutorial

AWS Elastic Container Service + Code Commit (Git) + Code Build + Code Deploy + Code Pipeline + SNS + ChatBot +Application Load Balancer (ALB) + Route 53

Hi Folks,

In this blog, I am going to explain the step by step tutorial for AWS ECS and CICD DevOps deployment pipeline with the help of native AWS tools. You can find the below video and GitHub code repo for reference.

So Lets Start …. !

Index:-

https://youtu.be/d7PTjQiahOQ

Step 1: Docker Basics

1. Create EC2 Instance

2. Install Docker

3. Pull centos:centos6

4. Create index.html

5. Create Dockerfile

6. Build a Docker image

```
1  yum update -y
    2  yum install docker -y
    3  docker info
    4  service docker start
    5  service docker status
    6  clear
    7  docker info
    8  docker pull centos:centos6
    9  docker imgaes
   10  docker images
   11  docker run -it -p 80:80 d0957ffdf8a2
   12  docker ps
   13  docker ps -a
```

Dockerfile

```
FROM centos:centos6MAINTAINER VarunMnaikRUN yum -y install httpdCOPY
index.html /var/www/html/CMD ["/usr/sbin/httpd", "-D",
"FOREGROUND"]EXPOSE 80
```

Index.html

```
<html>
<body><h1>CICD & Docker Tutorial By Varun Kumar Manik </h1><p
style="background-color:DodgerBlue;">Version One (V2.0) in Blue
Color.</p></body>
</html>
```

https://github.com/manikcloud/docker

Step 2: AWS ECR Creation

7. Create AWS ECR

8. Login to ECR

9. Tag existing image as AWS ECR repo

10. Push the image into the ECR

Step 3: Create ALB

11. Create Target group

12. Create ALB

Step 4: Create an AWS ECS Cluster

13. Create a Task with Fargate Computability

14. Create a FargateCluster

15. Create a Service with ALB enabled

16. Test ALB DNS, whether your site is running or not

Step 5: AWS Code commit

17. Create a Repo

18. Set ssh connectivity in your local machine

19. Push your code into the newly created repo

Step 6: AWS Code Build

20. Create a Code build Project

21. Select an Artifact for output

22. Build your Docker image & push to AWS ECR

buildspec.yaml

```
version: 0.2phases:
  pre_build:
    commands:
      - echo Logging in to Amazon ECR...
      - $(aws ecr get-login --no-include-email --region
$AWS_DEFAULT_REGION)
  build:
```

```
    commands:
      - echo Build started on `date`
      - echo Building the Docker image...
      - docker build -t  web:1 .
      - docker tag web:1 27673333333.dkr.ecr.us-east-
1.amazonaws.com/manikcloud
  post_build:
    commands:
      - echo Build completed on `date`
      - echo Pushing the Docker image...
      - docker push 27673333333.dkr.ecr.us-east-
1.amazonaws.com/manikcloud
```

Step 7: AWS Code Pipeline

23. Create a pipeline Project

24. Select you Code commit repo as an input repo from step 5

25. Select Code build project from Step 6

26. Select Code Deploy for ECS

27. Select your cluster and Service name from step 3

28. Crete the Code pipeline

Step 8: Error & Resolution

29. In the above pipeline, first 2 steps will run successfully

30. Step 3 deploy will give you an error

31. For this, you need to Create one “imagedefinitions.json” file and push it to the code commit

32. The pipeline will run again and you will again get an error on step 3

imagedefinitions.json

```
[
    {
        "name": "https",
        "imageUri": "27673333333.dkr.ecr.us-east-
1.amazonaws.com/manikcloud:latest"
```

```
    }
]
```

Step 9: Modify Input Artifacts Settings for Stage

33. Change the input

Step 10: Final Deployment Test and Validation

34. Push the new version of code in a code commit

35. It will automatically deploy the new task with the new version

36. At last, you can run the DNS ALB on your browser