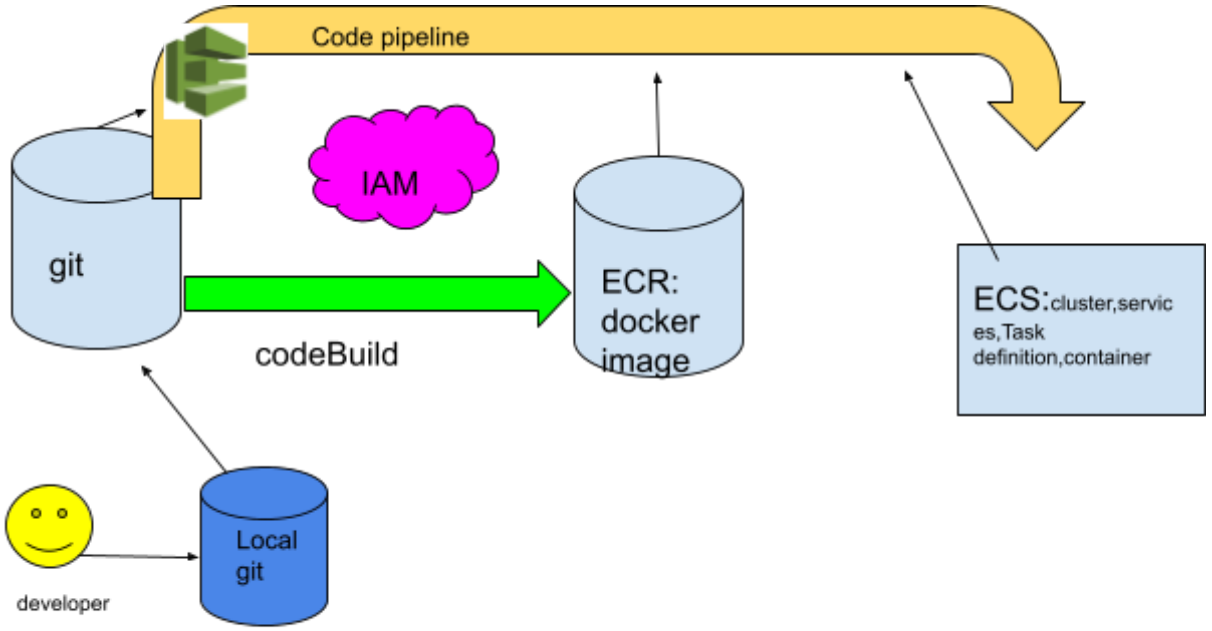
Containerize (Docker) din applikation och deploya på en lastbalanserad ECS och
Serva även en lolcat-bild som du laddat upp till S3 från din webbapp.

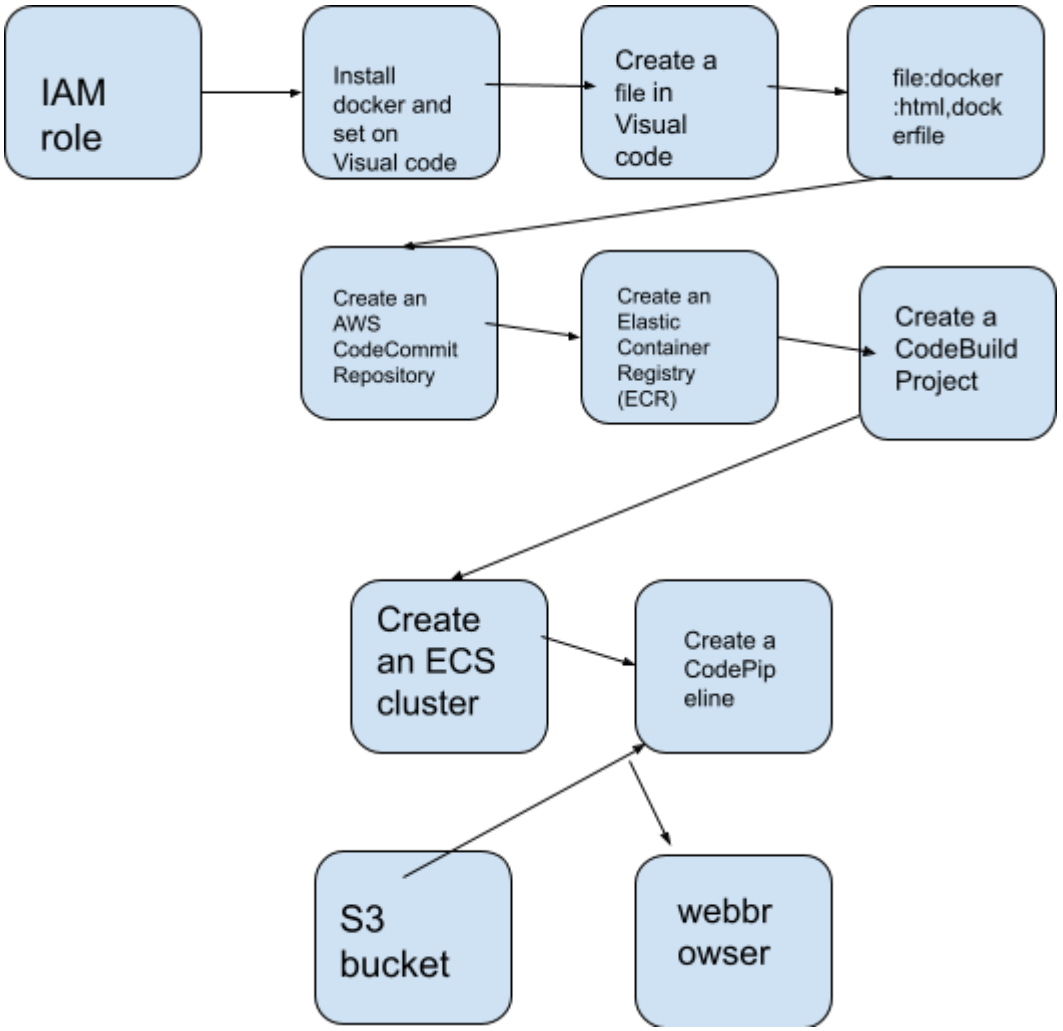
How it should work

:


Code pipeline
IAM
git
ECR:
docker
image
ECS:cluster,servic
es,Task
definition,container
codeBuild
Local
git
developer

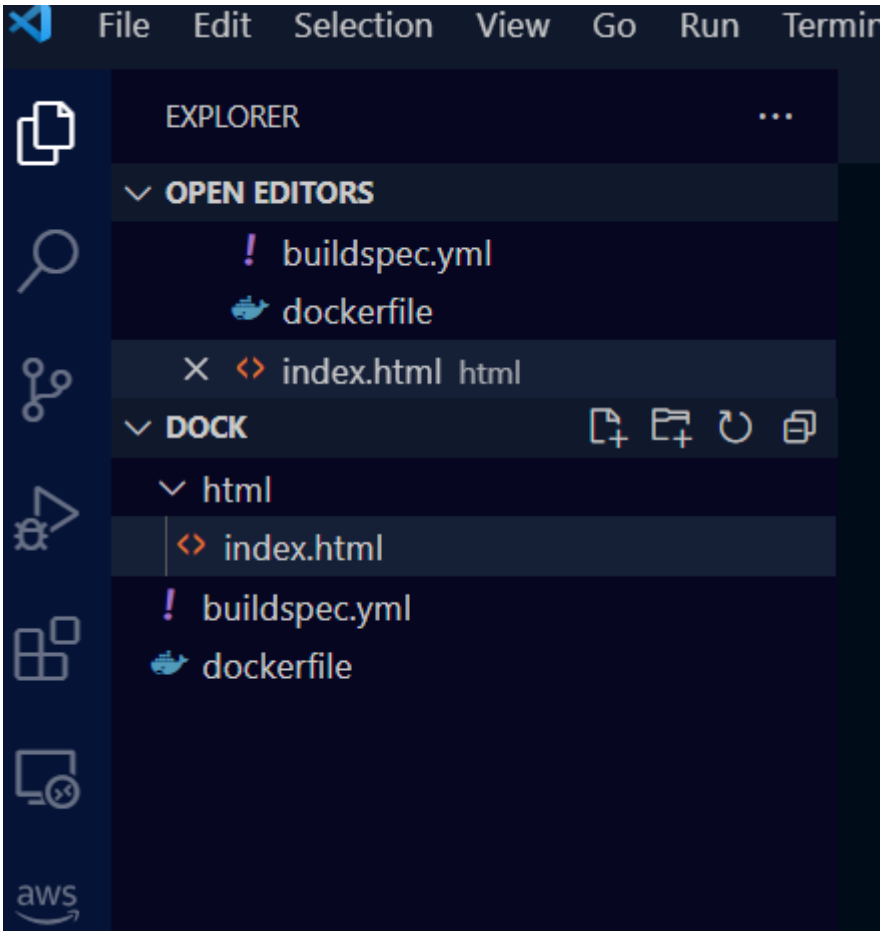
How it works!

Steps to achieve:



Step1#
Install Docker
Install docker app and run its feature on visual code/CMD/powershell.

After install create a folder:



Create a Index.html file:
Go to visual code and create a folder then inside that folder create a file name as html.Inside the html create a file name as index.html.

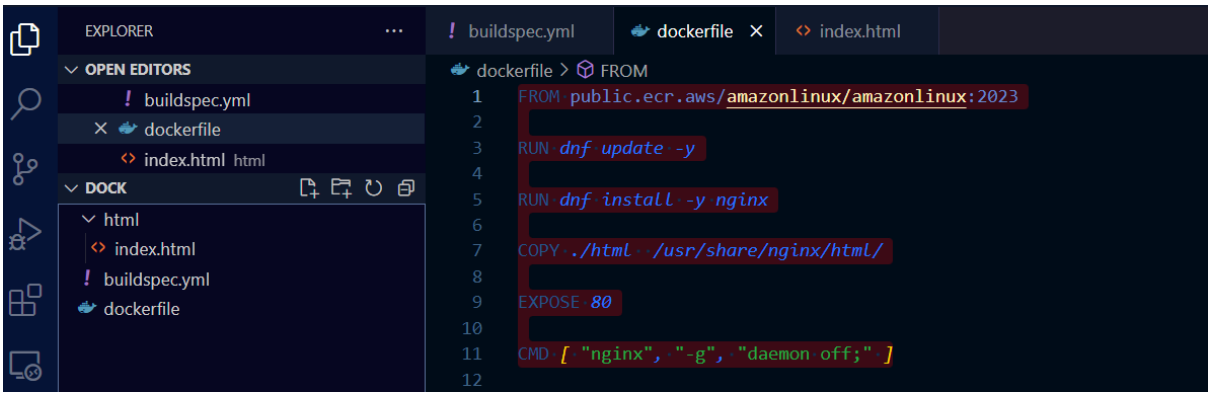
Their you need to create index.html using this data:

```
<!DOCTYPE html>
<html>
<head>
<title>My Simple Web Page</title>
</head>
<body style="background-color: lightblue;">
<h1>Welcome to ABC web page!</h1>
<p>This is a dockerized web site.</p>
</body>
</html>
```

Create a Dockerfile:
Create a docker file inside the dock folder of visual code where you have html:

Here you need to have this script in it and save it. This is the image which will be fetched to make the container.

```
FROM public.ecr.aws/amazonlinux/amazonlinux:2023

RUN dnf update -y

RUN dnf install -y nginx

COPY ./html  /usr/share/nginx/html/

EXPOSE 80

CMD [ "nginx", "-g", "daemon off;" ]
```



[1]

Import "buildspec.yml":
After creating folder and adding files as html and dockerfile now we will import "buildspec.yml" using this following command in visual code to import this file:

```
git clone https://github.com/larsappel/ECSDemo.git
```

Then use this to remove the file after copying it,

```
# Remove the local git repo
cd ECSDemo
rm -Rf .git
```

[1] Remember to put all folders separately - like:html,buildspec and dockerfile should be separated under the dock not inside another folder.

Buildspec will be import from git clone https://github.com/larsappel/ECSDemo.git

Step2#:

Create an AWS CodeCommit Repository:

1.Create a local code repository using this script in visual studio:

```
git init
git add .
git commit -m 'Add simple web site'
```

2. Create a CodeCommit code repository

```
aws codecommit create-repository --repository-name DockerDemo --repository-description
"Docker Demo Repository
```

How it will look when you run this command:

```
PS C:\Users\88amnsoh\dock> aws codecommit create-repository --repository-name DockerDemo --repository-description "Docker Demo Repository
"
{
    "repositoryMetadata": {
        "accountId": "605728483337",
        "repositoryId": "9d9ece62-5d74-4fe7-a9ac-f08e5969d4e4",
        "repositoryName": "DockerDemo",
        "repositoryDescription": "Docker Demo Repository",
        "lastModifiedDate": "2023-11-14T23:36:38.110000+01:00",
        "creationDate": "2023-11-14T23:36:38.110000+01:00",
        "cloneUrlHttp": "https://git-codecommit.eu-west-1.amazonaws.com/v1/repos/DockerDemo",
        "cloneUrlSsh": "ssh://git-codecommit.eu-west-1.amazonaws.com/v1/repos/DockerDemo",
        "Arn": "arn:aws:codecommit:eu-west-1:605728483337:DockerDemo"
    }
}
```

3. Set CodeCommit repo as origin. Get info from the output above

```
git remote add origin
https://git-codecommit.eu-west-1.amazonaws.com/v1/repos/DockerDemo
```

4. Push to CodeCommit

```
git push -u origin master
```

How it look like:

```
PS C:\Users\88amnsoh\dock> git push -u origin master
Enumerating objects: 15, done.
Counting objects: 100% (15/15), done.
Delta compression using up to 12 threads
Compressing objects: 100% (15/15), done.
Writing objects: 100% (15/15), 2.02 KiB | 82.00 KiB/s, done.
Total 15 (delta 2), reused 0 (delta 0), pack-reused 0
remote: Validating objects: 100%
To https://git-codecommit.eu-west-1.amazonaws.com/v1/repos/DockerDemo
 * [new branch]      master -> master
branch 'master' set up to track 'origin/master'.
```

Now run these commands:

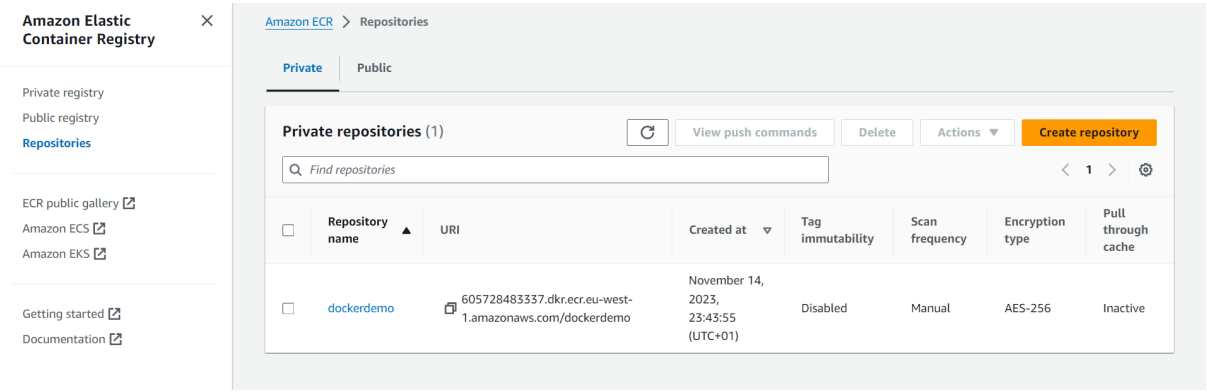
```
git add .
git status
```

To get the status and update the addition in anything new added.

Step3#: Create an Elastic Container Registry (ECR)

1. Create the ECR repository
Run this script to make ECR:

```
aws ecr create-repository --repository-name dockerdemo
```



2. Add a *buildspec.yml* file to the CodeCommit repo

Now edit this script it's highlighted lines:

```
version: 0.2

phases:
  pre_build:
    commands:
      # Fill in ECR information
```

```
    - REGISTRY_URI=605728483337.dkr.ecr.eu-west-1.amazonaws.com
    - IMAGE_NAME=dockerdemo
    - REGION=eu-west-1
    # Fill in ECS information
    - CONTAINER_NAME=DockerDemoContainer # TaskDefinition: container definition
name (Wrapper for imageUri)
    # -----------------------
    - IMAGE=$REGISTRY_URI/$IMAGE_NAME
    - COMMIT=$(echo $CODEBUILD_RESOLVED_SOURCE_VERSION | cut -c 1-8)
    - aws ecr get-login-password --region $REGION | docker login --username AWS
--password-stdin $REGISTRY_URI
  build:
    commands:
      - docker build --tag $IMAGE .
      - docker tag $IMAGE $IMAGE:$COMMIT
  post_build:
    commands:
      - docker push $IMAGE
      - docker push $IMAGE:$COMMIT
      # Create imagedefinitions.json. This is used by ECS to know which docker image to
use.
      - printf '[{"name":"%s","imageUri":"%s"}]' $CONTAINER_NAME $IMAGE:$COMMIT >
imagedefinitions.json
artifacts:
  files:
    # Put imagedefinitions.json in the artifact zip file
    - imagedefinitions.json
```

Now save the file and run these script:
Now run these commands:

```
git add .
git commit -m "My Simple Web Page"
—---
```

Output:

```
PS C:\Users\88amnsoh\dock> git add .
>> git commit -m 'Add simple web site'
[master 9d4e1aa] Add simple web site
 Committer: Amna Sohail <88amnsoh@mk.se>
Your name and email address were configured automatically based
on your username and hostname. Please check that they are accurate.
You can suppress this message by setting them explicitly:

    git config --global user.name "Your Name"
    git config --global user.email you@example.com

After doing this, you may fix the identity used for this commit with:

    git commit --amend --reset-author

 1 file changed, 1 insertion(+), 1 deletion(-)
```

```
git push -u origin master
```

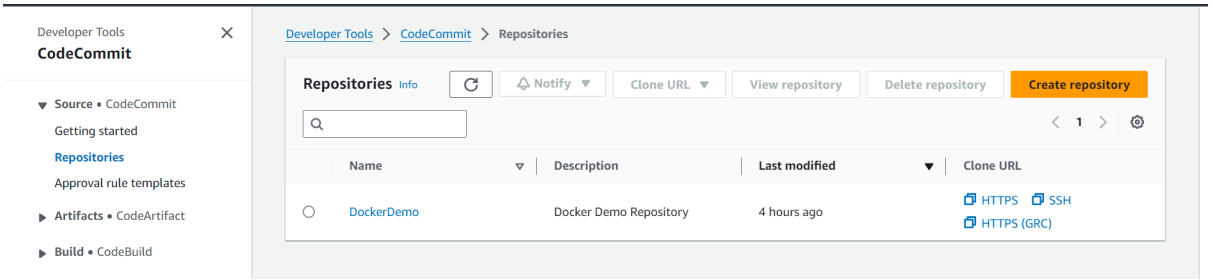
```
PS C:\Users\88amnsoh\dock> git push -u origin master
Enumerating objects: 15, done.
Counting objects: 100% (15/15), done.
Delta compression using up to 12 threads
Compressing objects: 100% (15/15), done.
Writing objects: 100% (15/15), 2.02 KiB | 82.00 KiB/s, done.
Total 15 (delta 2), reused 0 (delta 0), pack-reused 0
remote: Validating objects: 100%
To https://git-codecommit.eu-west-1.amazonaws.com/v1/repos/DockerDemo
 * [new branch]      master -> master
branch 'master' set up to track 'origin/master'.
```

git status
ls

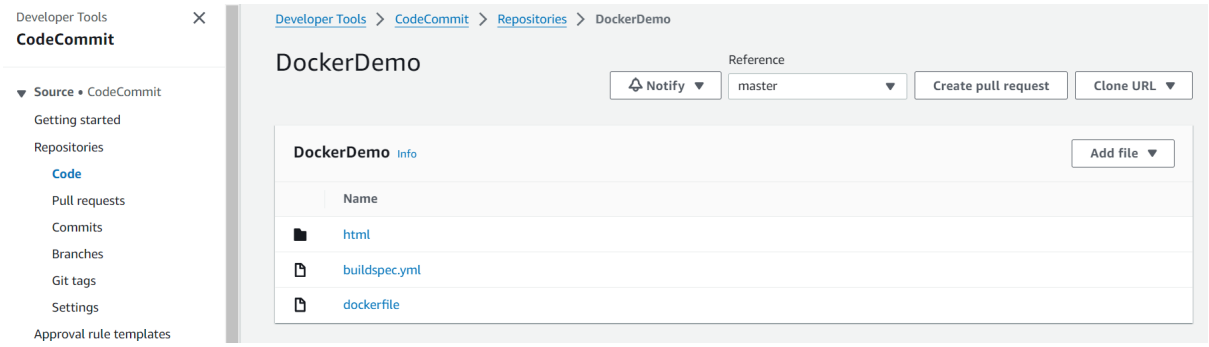
```
PS C:\Users\88amnsoh\dock> ls

    Directory: C:\Users\88amnsoh\dock

Mode                LastWriteTime         Length Name
----                -------------         ------ ----
d----         15/11/2023     10:08                html
-a---         14/11/2023     23:36           1117 buildspec.yml
-a---         09/11/2023     13:06            163 dockerfile
```

[2]

How it looks the repository in the aws console at code commit:



and when you click on the “dockerdemo” you can see these file over their



[2] When you do “ls” you can see all files are separated from each other. This is a very important step because only if they are separated can you build a project and create an image and proceed further.

Step4# Create a CodeBuild Project:

steps	Fill in material	Further detail
1**. Create build project**	1. Project name: **BuildDockerDemo**	2. **Source provide**r: *AWS CodeCommit* 3. **Repository:** *DockerDemo* 4. **Branch**: *master*
	Environment	1. Managed image 2. EC2 3. **Operating system:** *Ubuntu* 4. **Runtime**: Standard 5. **Image**: aws/codebuild/standard:7.0 6. **Privileged**: Yes (check) **7. New service role** 8. **Role name:** *codebuild-BuildDockerDemo-service-role* ***9. Use a buildspec file*** **Create build project**
2. Set permissions	Find the newly created role: **codebuild-BuildDockerDemo-service-role** [3]	Add the managed policy: **AmazonEC2ContainerRegistryPowerUser**

[3] When you're trying to push to an AWS repository, the Git username is typically the one associated with your AWS CodeCommit credentials. Here's how you can find it:

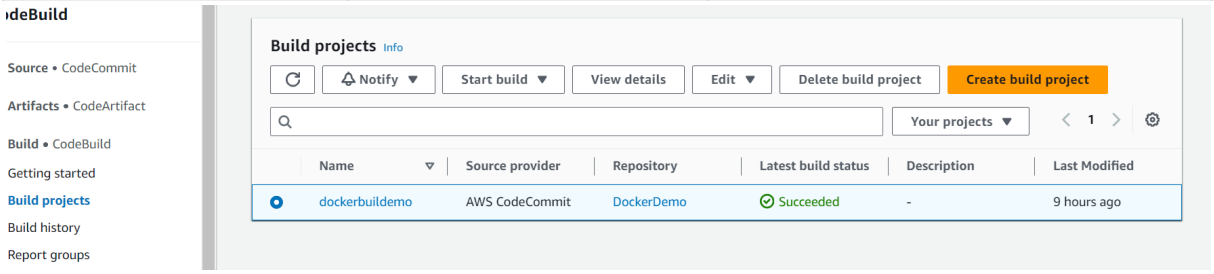
1. Go to **IAM** in your AWS Management Console.
2. Click on **Users** and select your user.
3. Go to the **Security credentials** tab.
4. Here, click on **Generate credentials** under **HTTPS Git credentials for AWS CodeCommit**[2].

You'll get a username and its secret key. This username is what Git uses when you're pushing to an AWS repository[2].

Please note that these are the credentials that AWS uses to authenticate your Git operations with AWS CodeCommit repositories. They are not the same as the `user.name` and `user.email` settings in your local Git configuration[4]. If you want to check your local Git username, you can use the command `git config user.name` in your terminal[4]. If you want to check your global Git username, you can use the command `git config --global user.name`[4].

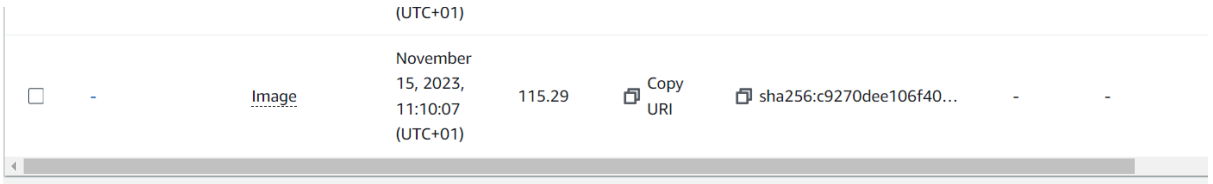
Remember to keep your AWS CodeCommit credentials secure and do not share them with anyone[2].

3.Start build		



4. Verify that the image is in the ECR repository:

Image will be deploy when code child is started:



Step 5# Create an ECS cluster

There are few steps:

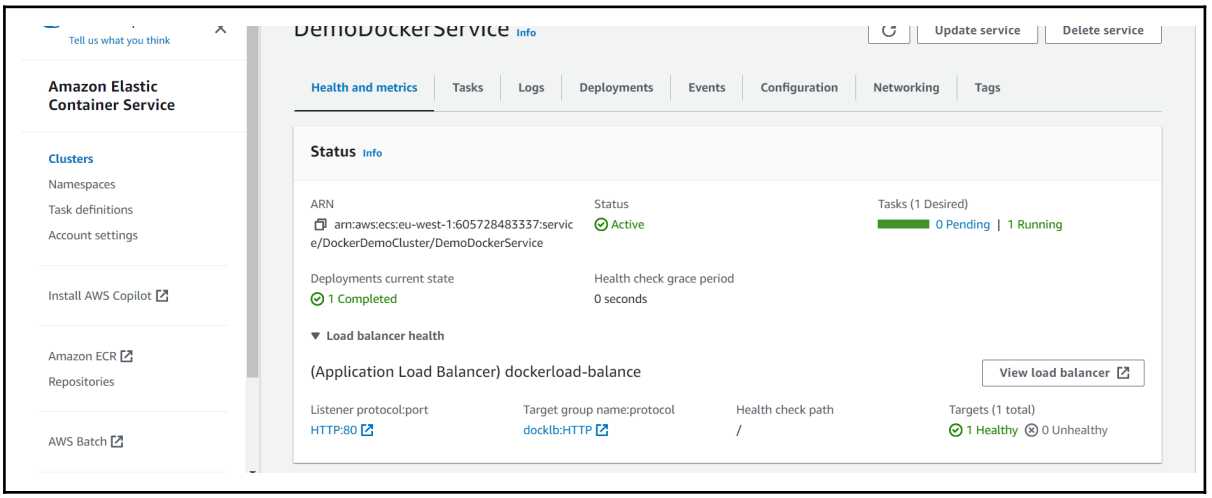
1. TASK DEFINITION
2. CLUSTER
3. SERVICES
4. LOAD BALANCE

TASK DEFINITION	• **Task definition family:** DockerDemoTask • **AWS Fargate** • **Container definition name:** DockerDemoContainer • **Image URI**: Copy the URI of the image • **Create**
CLUSTER	**Cluster name**: DockerDemoCluster **Create**
Create Service	**Select the cluster** **Go to Services tab** -> Create **Family**: DockerDemoTask **Service name**: DockerDemoService **Replica**: 1 **Expand Networking**[4]: Choose or create a security group open

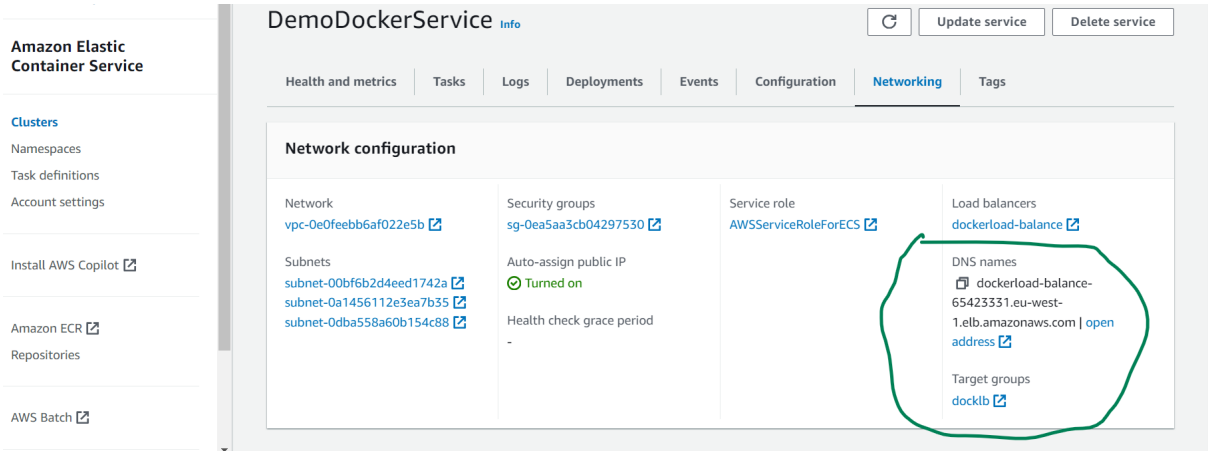
[4] IT IS BEST PRACTICE TO CREATE A SECURITY GROUP IN EARLY STAGE WITH PORT 80.THEN ADD THAT GROUP

	for HTTP Create
LOAD BALANCE	HERE YOU CAN CREATE A LOAD BALANCE BY GIVING A NEW NAME AND CREATING A TARGET GROUP. NOTE THAT IT SHOULD BE APPLICATION LOAD BALANCE.
Verify that the service started a task/container.	Find the public IP and open in a browser

OUTCOME OF CODE BUILD



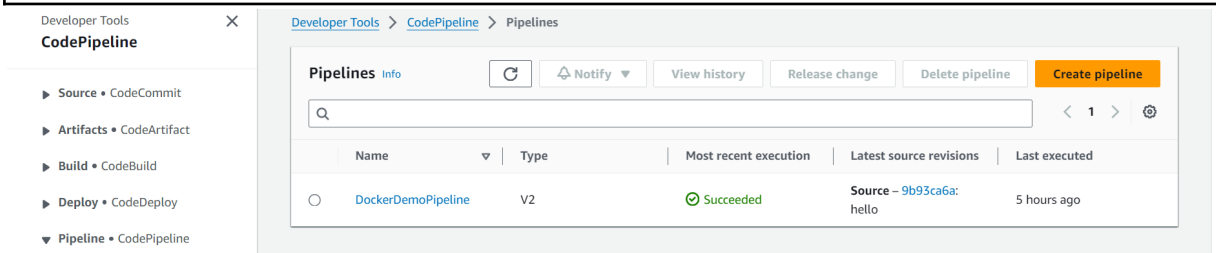
When you go to network on the same page you will find DNS to visit the page:



Step 6# Create a CodePipeline
1.Create a PIPELINE:

1. **Pipeline name**: DockerDemoPipeline
2. **New service role** (AWSCodePipelineServiceRole-eu-west-1-DockerDemoPipeline)

3. **Next**
4. **Source provider**: AWS CodeCommit
5. **Repository name:** DockerDemo
6. **Branch name:** master
7. **Amazon CloudWatch Events (recommended)**
8. **CodePipeline default**
9. **Next**
10. **Build provider:** AWS CodeBuild
11. **Project name:** BuildDockerDemo
12. **Next**
13. **Deploy provider**: Amazon ECS
14. **Cluster name**: DockerDemoCluster
15. **Service name:** DockerDemoService
16. **Next**
17. **Create pipeline**



2. Verify that the pipeline works

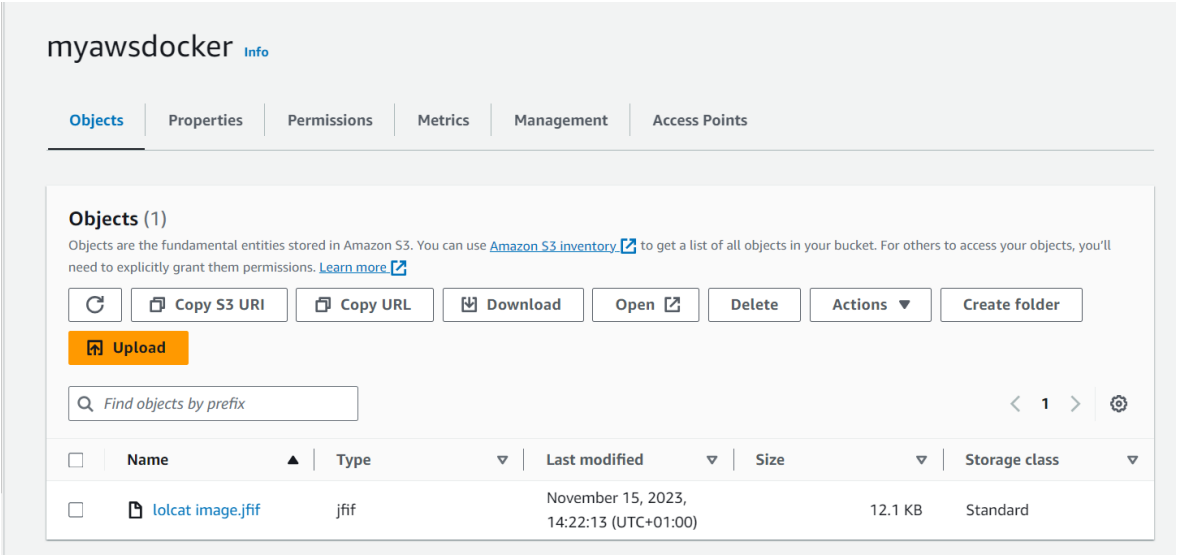
Browse to the service:

Now changing the code and commit it:
For this I try to do some extra thing:

- Go to S3

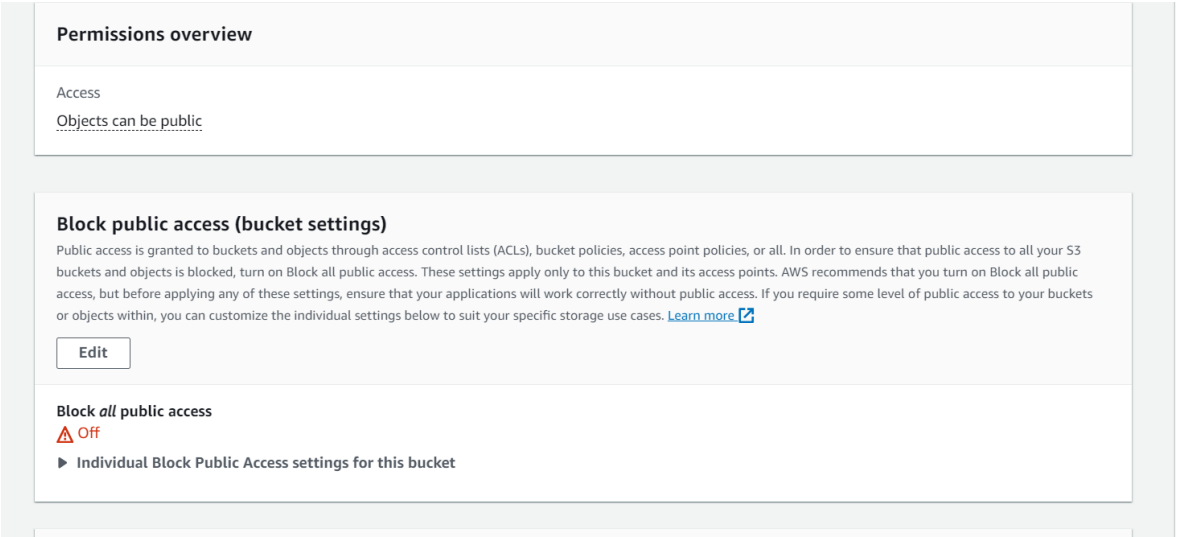
- Create a bucket
- Unblock all public access and create

Now ,choose an image some LOL cat and download it on your local system then upload it on your S3 bucket.



Now go to your bucket and make these certain changes:

Step 1#



step2#

Change in “bucket policy”

Bucket policy

Edit Delete

The bucket policy, written in JSON, provides access to the objects stored in the bucket. Bucket policies don't apply to objects owned by other accounts. Learn more

Copy

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": "*",
      "Action": "s3:GetObject",
      "Resource": "arn:aws:s3:::myawsdocker/*"
    }
  ]
}
```

step3#
ACL look:

Access control list (ACL)

Edit

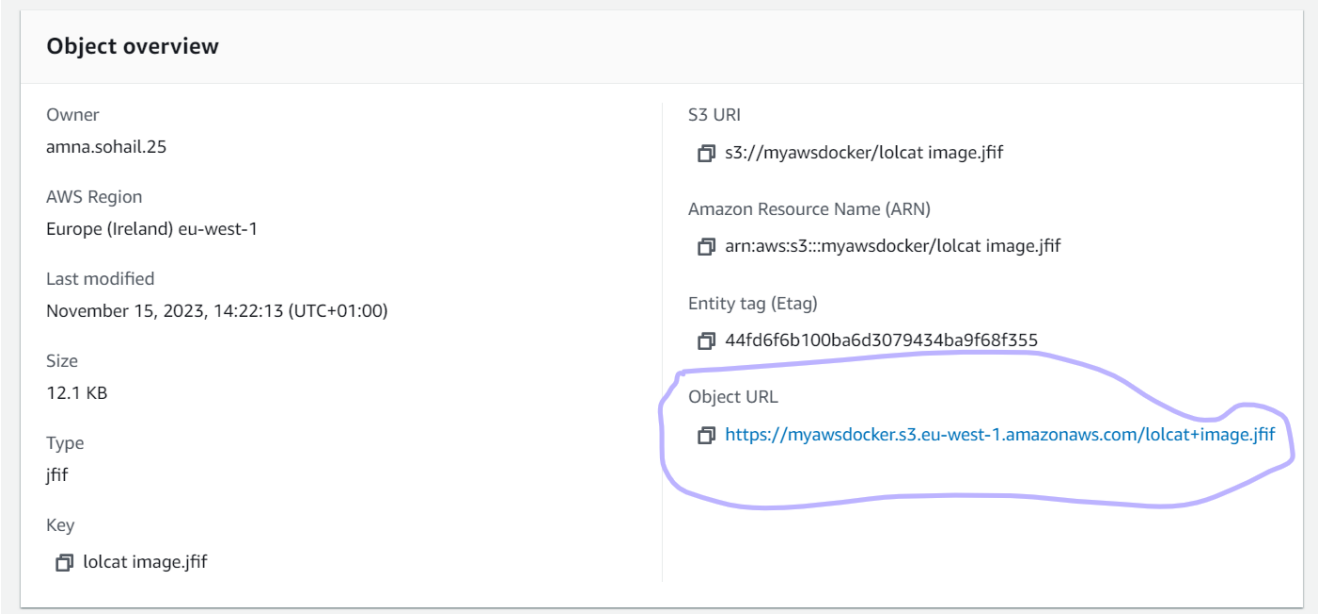
Grant basic read/write permissions to other AWS accounts. Learn more

The console displays combined access grants for duplicate grantees
To see the full list of ACLs, use the Amazon S3 REST API, AWS CLI, or AWS SDKs.

Grantee	Objects	Bucket ACL
Bucket owner (your AWS account) Canonical ID: b6733d4b5f5bf247bf86e18ccfbd508f1ca8e5919efa0f30ac2be9c7927e3116	List, Write	Read, Write
Everyone (public access) Group: http://acs.amazonaws.com/groups/global/AllUsers	-	-
Authenticated users group (anyone with an AWS account) Group: http://acs.amazonaws.com/groups/global/AuthenticatedUsers	-	-
S3 log delivery group		

Make this all changes in your bucket 👍

Then go to S3 bucket where you have uploaded the image and take the URL



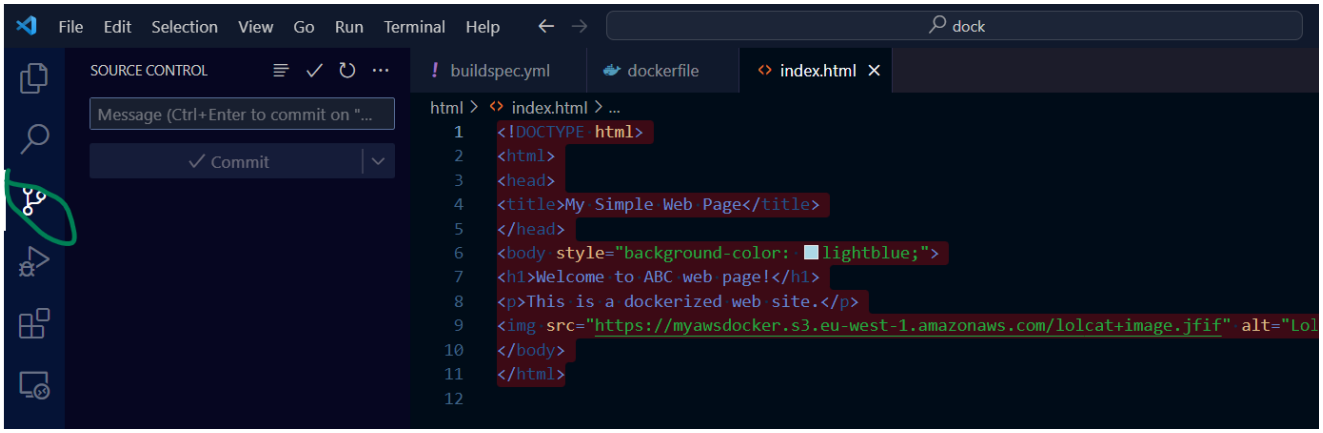
and copy it: then paste the url in you visualcode index.html file like this:

```
<!DOCTYPE html>
<html>
<head>
<title>My Simple Web Page</title>
</head>
<body style="background-color: lightblue;">
<h1>Welcome to ABC web page!</h1>
<p>This is a dockerized web site.</p>
<img src="https://myawsdocker.s3.eu-west-1.amazonaws.com/lolcat+image.jfif"
alt="Lolcat Image">
</body>
</html>
```

- Then do
- Git status
- Git add..
- Git commit -m “xyz”
- Git push -u origin master

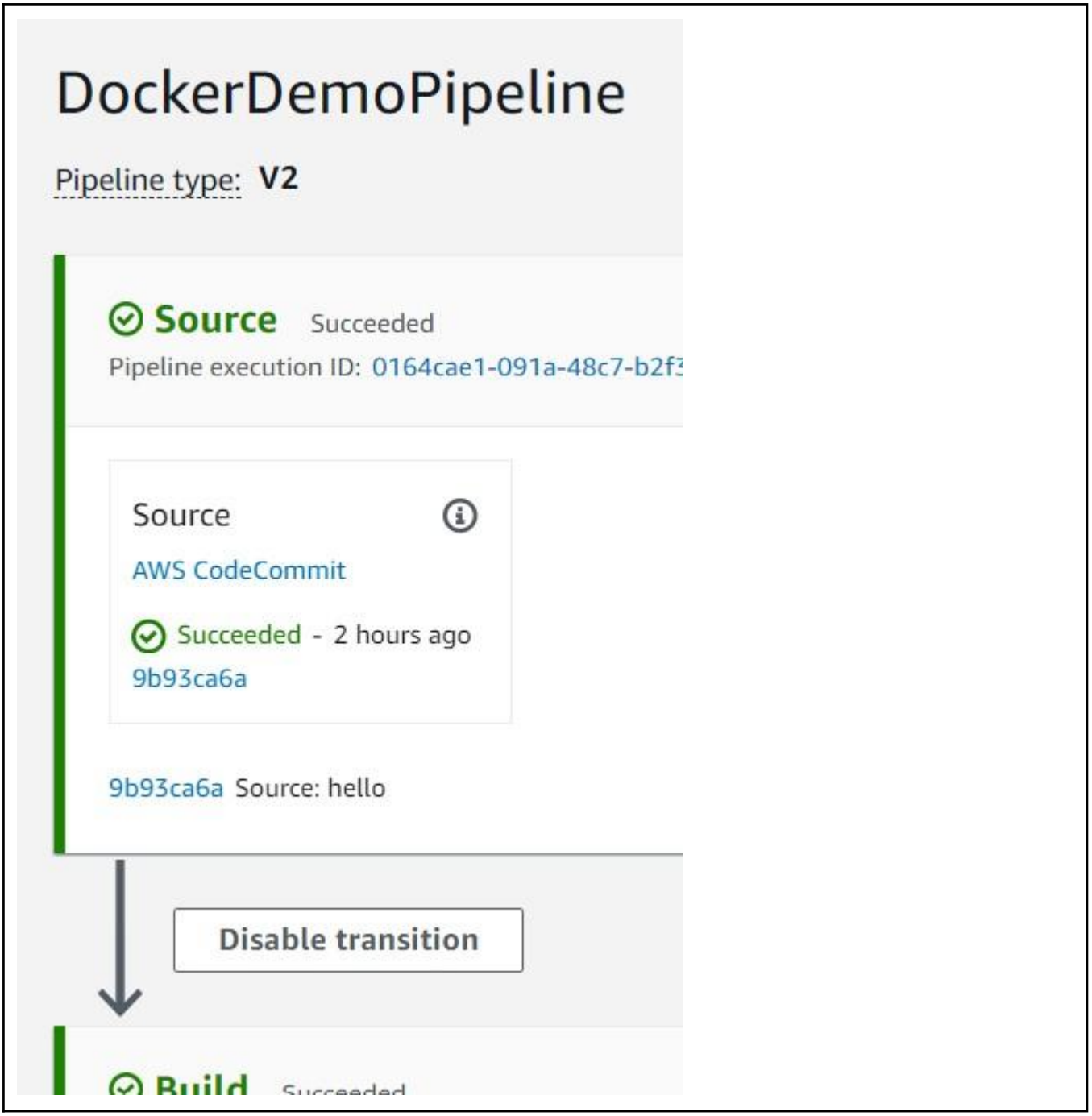
alt2#

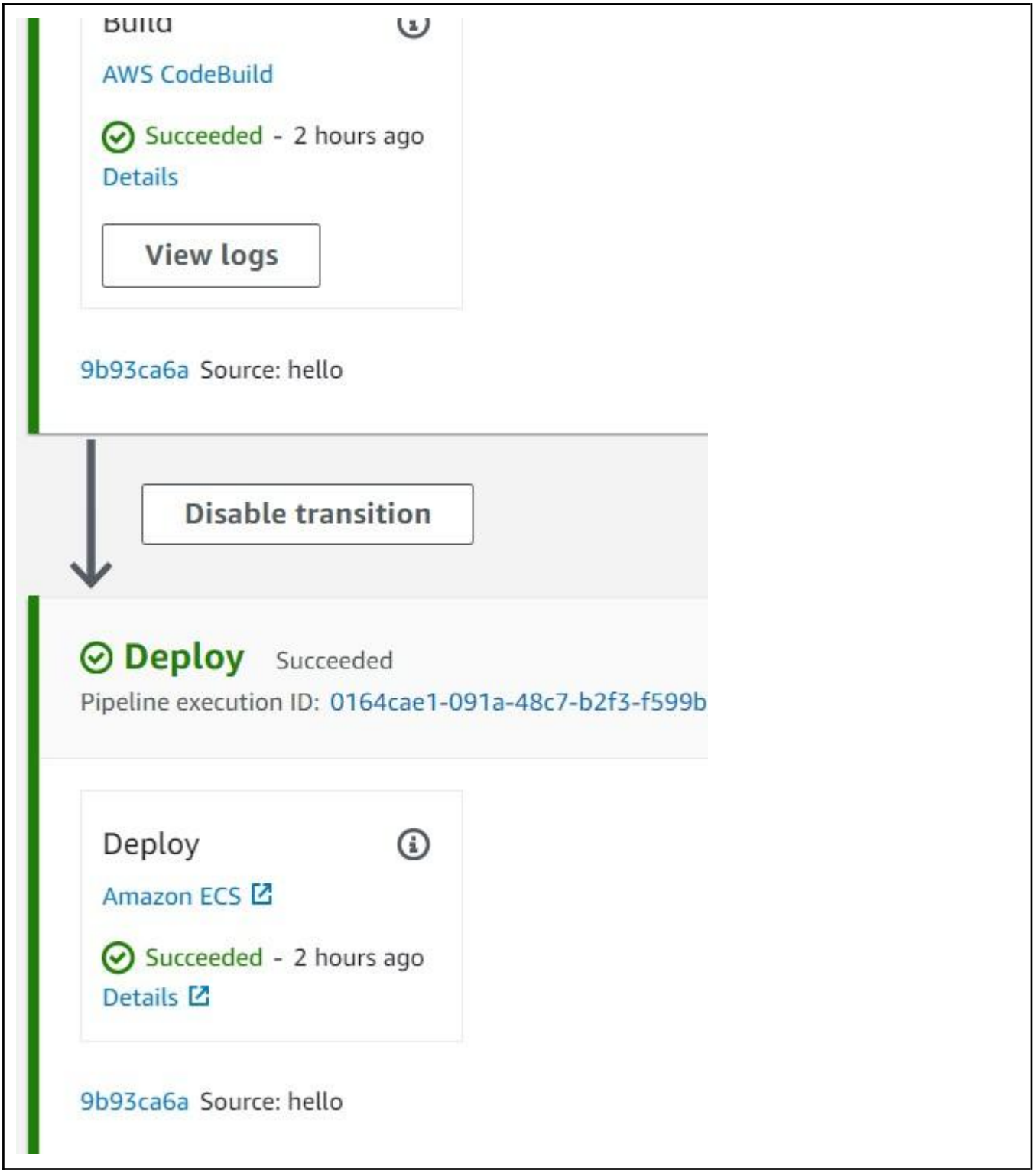
Amna Sohail Mov22:17



Click here on visual code where the green circle you can see then write anything here and then click commit,the sync and then see on AWS console how it goes for you to deploy code and deploy goes on application:

This is how process goes:


DockerDemoPipeline
Pipeline type: V2
Source Succeeded
Pipeline execution ID: 0164cae1-091a-48c7-b2f3
Source
AWS CodeCommit
Succeeded - 2 hours ago
9b93ca6a
9b93ca6a Source: hello
Disable transition


Build
AWS CodeBuild
Succeeded - 2 hours ago
Details
View logs
9b93ca6a Source: hello
Disable transition
Deploy Succeeded
Pipeline execution ID: 0164cae1-091a-48c7-b2f3-f599b
Deploy
Amazon ECS
Succeeded - 2 hours ago
Details
9b93ca6a Source: hello

Result:

Welcome to ABC web page!

This is a dockerized web site.