**How to push image to ECR and deploy docker container using AWS codePipeline:**

**Step 1:**

First we create ECR repo to push image.

Login to aws account and goto ECR(Elastic Container Registry). Click on “Get Started”

A blue background with white text

Description automatically generated

Give name of repo.

A screenshot of a computer

Description automatically generated

Enable “Scan on Push” and click on “create Repository”.

A screenshot of a computer

Description automatically generated

Now repository created.

A screenshot of a computer

Description automatically generated

**Step 2:**

Now we create our repo in codecommit and add our deployment files to it.

Goto “codecommit” and click on “create repository”.

A screenshot of a computer

Description automatically generated

Gove repository name and click on “create”.

A screenshot of a computer

Description automatically generated

Now we add our required files to our repo.

A close-up of a message

Description automatically generated

Files available in github:

https://github.com/ksnithya/AWS-code-pipeline.git

Dockerfile:

A screenshot of a computer program

Description automatically generated

demo.py:

A screenshot of a computer program

Description automatically generated

requirements.txt

A screenshot of a computer

Description automatically generated

Buildspec.yml

A screenshot of a computer

Description automatically generated

**Step 3:**

Now we create a build project. “buildspec.yml” file should exist in repository. This is default file name for codebuild. If you are using different file name we need to give that file name while creating codebuild. This file contains the instruction to build the project without this file we cant build it.

Goto “codebuild” and click on “create project”.

A screenshot of a computer

Description automatically generated

Give project name.

A screenshot of a computer

Description automatically generated

Now select the source. Click the “source Provider”. From the list select “Aws CodeCommit”. We are going to use the repository we created in previous step.

A screenshot of a computer

Description automatically generated

Now we need to select the compute resource where we are going to build. Here I am selecting “Amazon Linux”. We can select other OS also from dropdown list.

A screenshot of a computer

Description automatically generated

We need to create a service role. Name will be automatically taken.

A screenshot of a computer

Description automatically generated

Now we are going to give buildspec details. We can directly type the command or we can select “Use a buildspec file” option. Here I am using “Use buildspec file”. I already have file in my repo.

A screenshot of a computer

Description automatically generated

If we are using different file name need to give the file name. By default file name is “buildspec.yml”.

A screenshot of a computer

Description automatically generated

We can leave the remaining filed as it is and click on “create build project”

A screenshot of a computer program

Description automatically generated

Now we add the below Environment variables.

$AWS\_REGION = ap-south-1

$ECR\_ID = 822626997628

$REPO\_IMAGE =

$TAG =

To add environment variable go inside the project and click “edit”.

A screenshot of a computer

Description automatically generated

Under “Environment” -> Additional configuration -> Enable “Privilaged”, because we are building docker image.

A screenshot of a computer

Description automatically generated

Then click on “Add environment variable”.

A white background with black and white clouds

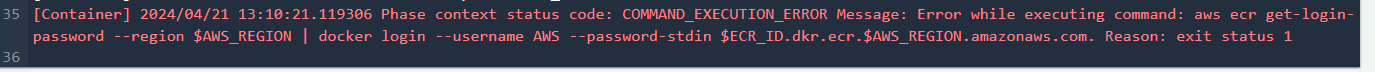
Description automatically generated

We can add all our environment variables as below and click on “update project”

A screenshot of a computer

Description automatically generated

Now we can test our build project. Click on “Start build” and test it. Our build will fail because our codebuild project has don’t have access to ECR registry. To fix this we need to add a ECR policy to our build role.



Now we add ECR policy to our role.

arn:aws:iam::822626997628:role/service-role/**codebuild-MyBuildProject-service-role**

A screenshot of a computer

Description automatically generated

Now goto “IAM” and edit the role.

A screenshot of a computer

Description automatically generated

Now click on “Add Permission” -> Attach policy.

A screenshot of a computer

Description automatically generated

Add the “[AmazonEC2ContainerRegistryFullAccess](https://us-east-1.console.aws.amazon.com/iam/home?region=ap-south-1#/policies/details/arn%3Aaws%3Aiam%3A%3Aaws%3Apolicy%2FAmazonEC2ContainerRegistryFullAccess)” policy. -> Add permission.

A screenshot of a computer

Description automatically generated

Now we try build again and check it. Now build is success. We can also see image pushed to ECR

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

**Step 4:**

Now we are going to create docker container using the image we pushed to ECR in above step.

We are going to use “codeDeploy” to build docker container. To create codedeploy we need to create the resource where we are going to deploy. We can create anyone of the below resources. Here we are going to create EC2 to deploy our docker container.

1. EC2
2. S3
3. ECS

First we need to create Instancerole. We are going to add below policies to it.

1. **AmazonEC2RoleforAWSCodeDeploy**
2. **AmazonSSMManagedInstanceCore**
3. **AmazonEC2ContainerRegistryFullAccess (we are going to pull image from ECR so we need to add this policy so that our EC2 can connect to ECR. Without this policy our deployment will fail).**

**To create an instance role**

1. Open the IAM console at <https://console.aws.amazon.com/iam/>).
2. From the console dashboard, choose Roles.
3. Choose Create role.
4. Under Select type of trusted entity, select AWS service. Under Choose a use case, select EC2. Under Select your use case, choose EC2. Choose Next: Permissions.
5. Search for and select the policy named AmazonEC2RoleforAWSCodeDeploy.
6. Search for and select the policy named AmazonSSMManagedInstanceCore.
7. Search for and select the policy named AmazonEC2ContainerRegistryFullAccess
8. Choose Next: Tags.
9. Choose Next: Review. Enter a name for the role (for example, EC2InstanceRole).

Note

Make a note of your role name for the next step. You choose this role when you are creating your instance.

Choose Create role.

**To launch an instance**

1. Open the Amazon EC2 console at <https://console.aws.amazon.com/ec2/>.
2. From the side navigation, choose Instances, and select Launch instances from the top of the page.
3. In Name, enter Docker. This assigns the instance a tag Key of Name and a tag Value of Docker. Later, you create a CodeDeploy application that deploys the sample application to this instance. CodeDeploy selects instances to deploy based on the tags.
4. Under Application and OS Images (Amazon Machine Image), locate the Amazon Linux AMI option with the AWS logo, and make sure it is selected. (This AMI is described as the Amazon Linux 2 AMI (HVM) and is labeled "Free tier eligible".)
5. Under Instance type, choose the free tier eligible t2.micro type as the hardware configuration for your instance.
6. Under Key pair (login), choose a key pair or create one.

You can also choose Proceed without a key pair.

Note

For the purposes of this tutorial, you can proceed without a key pair. To use SSH to connect to your instances, create or use a key pair.

1. Under Network settings, do the following.

In Auto-assign Public IP, make sure the status is Enable.

* + Next to Assign a security group, choose Create a new security group.
  + In the row for SSH, under Source type, choose My IP.
  + Choose Add security group, choose HTTP, and then under Source type, choose My IP.

1. Expand Advanced details. In IAM instance profile, choose the IAM role you created in the previous procedure (for example, EC2InstanceRole).
2. Under Summary, under Number of instances, enter 1..
3. Choose Launch instance.
4. You can view the status of the launch on the Instances page. When you launch an instance, its initial state is pending. After the instance starts, its state changes to running, and it receives a public DNS name. (If the Public DNS column is not displayed, choose the Show/Hide icon, and then select Public DNS.)

A screenshot of a computer

Description automatically generated

**Step 5:**

Now we create “codedeploy” to deploy our docker container on EC2 instance created on above steps.

The appspec.yml file stands for *application specification file*, app spec file for short, and is a file unique to CodeDeploy. It is designed to manage our deployments by a series of hooks, or events, that are defined in the Hooks section of the file. This is a file in YAML or JSON format. This file tells CodeDeploy what to install on our instances, and what to run based on hooks in response to our updates to the code. We have added this file in our repo already.

Note:

We need to move all build artifact files to deploy then only it will take appspec.yml and it dependent files. To do that we need to add below lines in builspec.yml file.

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

Goto “codedeploy” > application -> create application

A screenshot of a computer

Description automatically generated

Give application name and select “Compute Platform” from dropdown list. We are using EC2/On-Permises and click on “create appliocation”.

A screenshot of a computer

Description automatically generated

We need to create “deployment Group”. Click on “Create deployment group”.

A screenshot of a computer

Description automatically generated

Give deploymenet group name and select the service role we created in previous step.

A screenshot of a computer

Description automatically generated

Under “Environment Configuration” select Ec2 Instance, in Key select “Name” and value select the name of EC2 we created previously. Ex. Docker

A screenshot of a computer

Description automatically generated

Under “**Agent configuration with AWS Systems Manager”** select “Now and schedule update”.

A screenshot of a computer

Description automatically generated

Under “Deployment Settings” select “CodeDeployDefault.OneAtATime”. Uncheck “LoadBalancer”

A screenshot of a computer

Description automatically generated

Then click on “Create deployment group”.

A screenshot of a computer

Description automatically generated

**Step 6:**

Now we create codepipeline to deploy the application on docker container.

Goto “codepipeline” -> create pipeline.

A screenshot of a computer

Description automatically generated

Give name to pipeline.

A screenshot of a computer

Description automatically generated

New service will be automatically created. Leave as it is. Then click on “Next”.

A screenshot of a computer

Description automatically generated

We need to select the source we created in “codecommit” stage. Then click on “Next.

A screenshot of a computer

Description automatically generated

Now we need to select the build provider. We need to select the codebuild project we created in previous step.

A screenshot of a computer

Description automatically generated

A screenshot of a project

Description automatically generated

Then click on next.

Now we need to select the “deploy provider” we have created earlier.

A screenshot of a computer

Description automatically generated

Then click on “Next”.

A screenshot of a computer screen

Description automatically generated

Now we can review our pipeline and click on

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

Then click on “Create Pipeline”.

We can see the execution status.

A screenshot of a computer

Description automatically generated

First “codecommit” stage will execute.

A white background with blue text

Description automatically generated

Then “codebuild” stage will execute.

A white rectangular object with blue text

Description automatically generated

Then “codedeploy” stage will execute.

A screenshot of a computer program

Description automatically generated

A black background with many small colored letters

Description automatically generated with medium confidence

Now application is running.

