**DEVOPS REFERENCE**

**Jenkins-CODECOMMIT Pipeline in Ubuntu**

* Use Jenkins-dev branch as source Repository which is on AWS Codecommit.
* Create an instance named Jenkins-test and upload the docker-compose file to the instance the file link is [here.](https://github.com/sahith-palika-99/Project-codes/tree/main/Jenkins_cicd) Unzip it in the instance to create a docker in docker container.  
  We are using it to create a separate docker container for the build environment.

|  |
| --- |
| In the compose.yml file edit with the following |
| jenkins-docker:  image: docker:dind  privileged: true  volumes:  - jenkins-data:/var/jenkins\_home  - jenkins-docker-certs:/certs/client  networks:  jenkins:  aliases:  - docker  environment:  - DOCKER\_STORAGE\_DRIVER=overlay2  - DOCKER\_TLS\_CERTDIR=/certs  ports:  - "80:3000"  - "2376:2376"  restart: always |

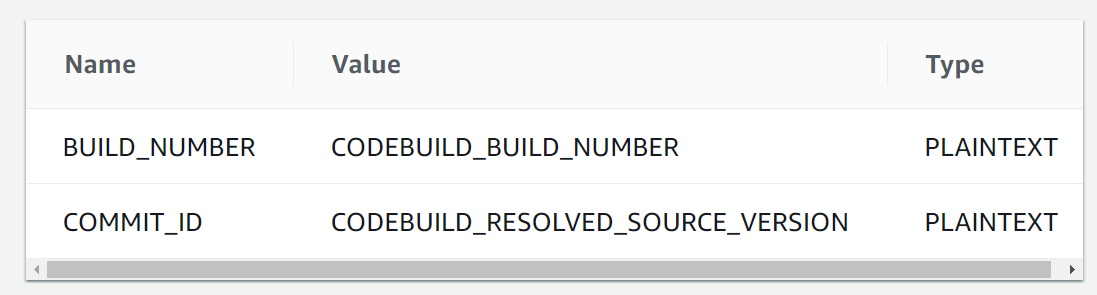
* Install docker using this link: [Install Docker Engine on Ubuntu | Docker Documentation](https://docs.docker.com/engine/install/ubuntu/)
* Then navigate to the directory of compose file and run <docker compose up --build --detach>
* Access the jenkins console in browser with the instance\_ip\_address:8080
* Login to the jenkins server container with <docker exec –it –u root “container-name” bin/bash>
* Nano /var/Jenkins home/secrets/initialAdminPassword, u can see the password copy this passwd and paste it in the Jenkins console
* Create new admin user. Start using the console
* Create an IAM user with appropriate permissions then go to security credentials tab. Navigate to **https git credentials for aws codecommit** and generate credentials
* Create a new item named aws-jenkins and select pipeline. Navigate to Pipeline option and under Definition choose pipeline script from SCM.  
  Under SCM choose Git and paste the repository URL. Under the credentials tab, click Add and choose Global credentials under Domain. Under Kind choose Username with password. Under scope select global and paste the username and password from the git credentials for AWS codecommit and give a unique identification ID. The Script path is Jenkinsfile. under branch tab Add the branch without \*/ in the branch name.
* Download the Cloudbees for AWS plugin and install it.
* Now navigate to manage jenkins > configure credentials and click on global and add credentials.  
  Choose Kind as AWS credentials, give unique ID (here I gave jenkins-build-1) and paste the access and secret key and click on create.
* If jenkins still throwing the aws not found error then manually install the awscli in jenkins-blueocean container(jenkins-server) as follows:  
  >> **apt-get install awscli**

|  |
| --- |
| NOTE – if your ecr get-login is failing from jenkins then add the following script  stages{  stage('Logging into AWS ECR') {  steps {  withCredentials([[$class: 'AmazonWebServicesCredentialsBinding', credentialsId: 'jenkins-build-1']]) {  echo "New IMAGE\_TAG: ${env.IMAGE\_TAG}"  sh "aws ecr get-login-password --region ${AWS\_DEFAULT\_REGION} | docker login --username AWS --password-stdin ${AWS\_ACCOUNT\_ID}.dkr.ecr.${AWS\_DEFAULT\_REGION}.amazonaws.com"  }}} |

* Install Codecommit trigger plugin from jenkins managed plugins SQS BUILD trigger SQS trigger
* Jenkins Auto-trigger selects Poll SCM and add ‘\* \* \* \* \*’ expression to auto-trigger.
* Notification to Teams channel - [How to Send AWS Notifications (AWS SNS) to Microsoft Teams - DEV Community](https://dev.to/aws-builders/how-to-send-aws-notifications-aws-sns-to-microsoft-teams-1d1l)

|  |
| --- |
| Python Lambda code to send commit details to Teams |
| import json  import boto3  import requests    def lambda\_handler(event, context):  print(type(event))    #getting the commit ID  commit\_id=event['Records'][0]["codecommit"]["references"][0]["commit"]    #getting the repo name  arn=event['Records'][0]["eventSourceARN"]  arn=arn.split(':')  repo\_name=arn[-1]    #get commit details  codecommit\_client = boto3.client('codecommit')  commit\_response = codecommit\_client.get\_commit(  repositoryName=repo\_name,  commitId=commit\_id  )    #extract commiter details  commit\_message = commit\_response['commit']['message']  committer\_name = commit\_response['commit']['committer']['name']  committer\_email = commit\_response['commit']['committer']['email']  #print(commit\_message,committer\_email,committer\_name)    #send the data to teams channel  message = f"{committer\_name} committed with {committer\_email} email to '{repo\_name}' with commit\_ID:'{commit\_id}' and message:'{commit\_message}'"  payload = {  "text": message  }  headers = {"Content-Type":"application/json"}  webhook\_url='https://cloudangles.webhook.office.com/webhookb2/4724a4ef-7c95-4e25-b101-a61e3feb3de9@4d2ad8cd-d755-4346-b482-67dd4b3c7074/IncomingWebhook/9df99f70d30941f29999a6e39c526538/f80844fb-2894-413f-a864-490a994fb316'  response = requests.post(webhook\_url, json=payload, headers=headers)  print(response) |
| **Lambda Layer** pip install requests –t location directory – mylayer.zip>>layer>>python>>----library |
| Add Lambda trigger from lambda console itseld as from codecommit it is acting weird |

**AWS Pipeline with Ubuntu**

* Create a code Pipeline that has source as your codecommit repo.
* In Add source stage give your repo details and choose AWS Cloudwatch events in the change detection option and Codepipeline default in Output Artifact format.
* Under Add build stage, choose build provider as CodeBuild and select create Project
* Enter the project name and in Environment section choose Managed image. For OS choose Ubuntu, runtime-standard and Image-aws/codebuild/standard:6.0.  
  Always use the latest image for this runtime version.  
  Select privileged checkbox to build Docker images.  
  Enter buildspec file path or if it is in root directory leave it blank.
* Add Environment Variables as below to tag the image with the Build number and commit ID  
  
* We will add the deployment stage later below the document. Skip the deployment and create
* If your build fails with at ecr get-login step with not authorized to ecr:GetAuthorizationTocken then add necessary permission to the IAM role of pipeline.
* Go to Codedeploy and create an application. Choose compute platform as EC2/On-premises and create application.
* Next create a deployment group and give it a name. Give the necessary IAM role. Choose Deployment type as In-place. Under Environment configuration choose EC2 instances and choose the instance. To see your instance match it with the tags. Under agent configuration Install AWS Codedeploy agent – Now and Schedule updates.
* Install code-deploy-agent in ec2 server by using following commands. Make sure you also have system manager agent.

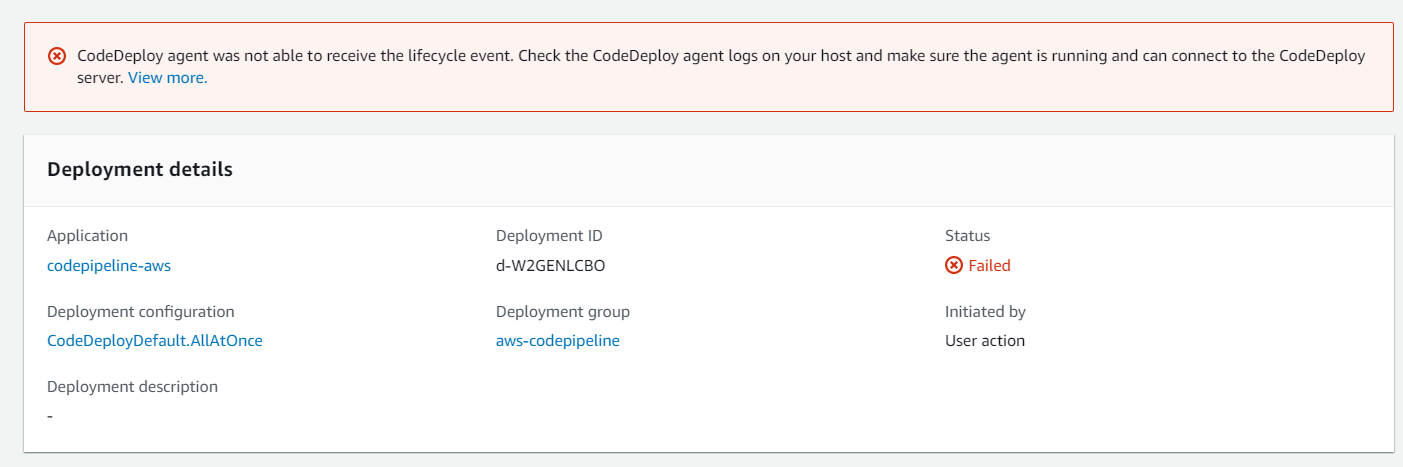
|  |
| --- |
| apt-get install -y codedeploy-agent  >> apt update  >> apt install ruby-full  >> apt install wget  >> wget <https://aws-codedeploy-us-east-1.s3.us-east-1.amazonaws.com/latest/install> (Amazon S3 bucket that contains the CodeDeploy Resource Kit files for your region. *region-identifier* is the identifier for your region. For example, for the US East (Ohio) Region, replace *bucket-name* with aws-codedeploy-us-east- and replace *region-identifier* with us-east-2. For a list of bucket names and region identifiers, see [Resource kit bucket names by Region](https://docs.aws.amazon.com/codedeploy/latest/userguide/resource-kit.html#resource-kit-bucket-names).)  >> chmod +x ./install  >> sudo ./install auto  >> systemctl status codedeploy-agent |

* To use codedeploy install ssm-agent and code-deploy agent.

|  |
| --- |
| To install ssm-agent >> snap install amazon-ssm-agent –classic  >> snap start amazon-ssm-agent >> snap services amazon-ssm-agent |

* Next to add the deployment stage go to pipeline and click edit.
* Add a stage named deploy and next Action group and give a name to action group. Under action provider choose Codedeploy. Under Input artifacts choose BuildArtifact.
* Give the Application name that you have created before and choose the deployment group.
* Click Done and Save the pipeline.

**Note -**

* If your deployment is failing at ApplicationStop event, then most likely the issue is your EC2 instance does not have the necessary permissions to get the artifacts from S3 bucket.
  + Your EC2 instance must have an IAM role attached which gives it enough permissions to download the artifacts from S3 bucket
  + Your EC2 must be **started with** an IAM role. So you may have to reboot your instance after attaching the role to it.
* If the docker pull is not working for pulling images or limit exceeded, then create a docker hub account and then login into it using username and password (username is different from email)
* If deployment failed at Download Bundle state, then add necessary permissions to EC2 IAM role.
* If your code pipeline deploy stage is facing the follow error:   
    
  

|  |
| --- |
| Checked the code deploy logs from cli: tail –f /var/log/aws/codedeploy-agent/codedeploy-agent.log and found this error |
| ERROR [codedeploy-agent(685)]: InstanceAgent::Plugins::CodeDeployPlugin::CommandPoller: Cannot reach InstanceService: Aws::CodeDeployCommand::Errors::AccessDeniedException - Aws::CodeDeployCommand::Errors::AccessDeniedException |
| * Then change the IAM role, give permissions stop the ssm agent and codedeploy agent and finally restart the instance. * Delete the .aws file from /root/ and restart codedeploy agent |

* Remember to install ecr-credential helper plugin in download [here](https://github.com/awslabs/amazon-ecr-credential-helper).
* To resolve the “No basic auth credentials” error   
  Replace the **~/.docker/config.json** with the below:

|  |
| --- |
| {  "credsStore": "ecr-login" } |

* Reference: [awslabs/amazon-ecr-credential-helper: Automatically gets credentials for Amazon ECR on docker push/docker pull (github.com)](https://github.com/awslabs/amazon-ecr-credential-helper#Configuration)