AWS CICD

Configuration and Setup

- Create an IAM user and configure aws cli on git bash using aws configure command.
- Launch an EC2 instance with Amazon Linux AMI Attach a Role to this EC2 with S3 Access,CodeDeploy Access, and CodeDeploy Agent should have access to list/get Objects from S3
    - Add below shell commands to install CodeDeploy Agent on the EC2 while userdata.

```
#!/bin/bash
sudo yum update -y
sudo yum install -y ruby wget
wget https://aws-codedeploy-eu-west-1.s3.eu-west-1.amazonaws.com/latest/install
chmod +x ./install
sudo ./install auto
sudo service codedeploy-agent status
sudo service codedeploy-agent enable
```

- Tag the EC2 instance with Name and Environment Values as
    - Name : Webserver
    - Environment : Dev

> Alternatively, a CF template to launch an EC2 instance with above commands as Userdata.

Creating a CodeDeploy Application

- Go to `CodeDeploy Service` > `Create Application` > Select Compute Platform as `"EC2/On-Premises"`
- Create a Deployment Group with name `DevEC2DeploymentGroup`
- The `Deployment type` selected by default is in-place deployment:

- Create a IAM Role for CodeDeploy Service that will have access to EC2, SNS etc service.
- Provide related Tag Groups as per Environment.
- Under `Environment configuration`, select `Amazon EC2 Instances`
  - Specify the right Tag Group `Key:Value` that is set on the EC2 instance.
- For Deployment Configuration, select the `CodeDeployDefault.OneAtATime`
- Deployment configuration
  - `CodeDeployDefault.AllAtOnce` :
    - Status is **Succeeded** if the application revision is deployed to one or more of the instances.
    - Status is **Failed** if the application revision is not deployed to any of the instances.
  - `CodeDeployDefault.HalfAtATime` :
    - Status is **Succeeded** if the application revision is deployed to at least half of the instances.
  - `CodeDeployDefault.OneAtATime` :
    - Status is **Succeeded** if the application revision is deployed to all of the instances. The exception to this rule is that if deployment to the last instance fails, the overall deployment still succeeds.
    - Status is **Failed** if the application revision fails to be deployed to any but the last instance.

CodeDeploy with S3 having Source Code.

- A deployment needs to be created once source code is present on S3 bucket.
- Create a new or use existing S3 bucket and enable versioning, replace below `<S3BUCKETNAME>` with your bucket name.
- To upload the source code from local machine to S3 bucket, run the `deploy push` command within the directory where your `appspec.yml` is present

```
aws deploy push --application-name CICD-Test --s3-location
s3://<S3BUCKETNAME>/CICD-Test/app.zip --ignore-hidden-files
```

- Once Deployment Group is created, we can click **Create deployment**, and specify the S3 path of the package code and the deployment of the artifact on your EC2 instances will begin.

- The deployment can be started using the **AWS CodeDeploy Service UI** or by using below **AWS CLI command**, replace appropriate content in the command below as per your account.

- To deploy with this revision, run:

  Start Deployment using aws cli

  ```
  aws deploy create-deployment --application-name CICD-Test --s3-location
  bucket=<S3BUCKETNAME>,key=codedeploy-
  demo/app.zip,bundleType=zip,eTag=3e960dd6842eea91c61843157946259d,version=1o
  Hw_ZABTFVWAvcXb7PrFft08pB4JW1T --deployment-group-name <deployment-group-
  name> --deployment-config-name <deployment-config-name> --description
  <description>
  ```

Start Deployment using AWS Console

- Navigate to **CodeDeploy Application** > Click on specific **Deployment Group** > Click on **Create Deployment**.
    - Under **Revision location**, specify the S3 `app.zip` revision file from S3 Bucket Path that was previously uploaded.

- Once the Deployment will start, there will be a Deployment ID created like `d-GC3B88ROA`
- Click on Deployment Id to see the event details executed on specific EC2 instance.
- On View Events on a specific EC2 instance ARN is displayed : `arn:aws:ec2:ap-south-1:<ACCOUNT_ID>:instance/i-0789556ecdf884653`
    - Details under this Page shows all the Event Hooks that are executed by the `CodeDeploy Agent` on the EC2.

Viewing the CodeDeploy Agent Logs and Deployment logs on the EC2 instance.

- The folder where related `revisions`, `deployment history`, and `deployment scripts` on the instance are stored.

```
tree /opt/codedeploy-agent/deployment-root
```

    - Output of above command is below directory structure.

```
    /opt/codedeploy-agent/deployment-root
├── deployment-instructions
│   ├── fb00c799-7826-4e8a-834b-98e83c278506-cleanup
│   ├── fb00c799-7826-4e8a-834b-98e83c278506-install.json
│   ├── fb00c799-7826-4e8a-834b-98e83c278506_last_successful_install
│   └── fb00c799-7826-4e8a-834b-98e83c278506_most_recent_install
├── deployment-logs
│   └── codedeploy-agent-deployments.log
├── fb00c799-7826-4e8a-834b-98e83c278506
│   └── d-0TY70HROA
│       ├── bundle.tar
│       ├── deployment-archive
│       │   ├── appspec.yml
│       │   ├── buildspec.yml
│       │   ├── index.html
│       │   └── scripts
│       │       ├── after_install.sh
│       │       ├── install_dependencies.sh
│       │       ├── start_server.sh
│       │       ├── stop_server.sh
│       │       └── validate_service.sh
│       └── logs
```

```
    |           └── scripts.log
    └── ongoing-deployment
```

- `Deployment Group ID` : This is the deployment group directory's name having an ID like `fb00c799-7826-4e8a-834b-98e83c278506`.
    - Each deployment group directory contains one subdirectory for each attempted deployment in that deployment group.
- `Deployment ID` directory represent each deployment in a deployment group with directory's name is its ID `d-0TY70HROA`. Each directory contains:
    - `bundle.tar` : a compressed file with the contents of the deployment's revision. Use a zip decompression utility if you want to view the revision.
    - `deployment-archive` : contains the contents of the deployment's revision.
    - `logs` :contains a `scripts.log` file that lists the output of all scripts specified in the deployment's AppSpec file.

- The folder on the instance where log files related to CodeDeploy agent operations are stored.

```
cat /var/log/aws/codedeploy-agent/codedeploy-agent.log
```

- To determine the location of the last successfully deployed application revision

```
cat /opt/codedeploy-agent/deployment-root/deployment-instructions
```

Lifecycle Event hooks in appspec.yml file

- **ApplicationStop** – This deployment lifecycle event occurs even before the application revision is downloaded.
- **DownloadBundle**- During this deployment lifecycle event, the CodeDeploy agent copies the application revision files to a temporary location.
- **BeforeInstall**– You can use this deployment lifecycle event for preinstall tasks, such as decrypting files and creating a backup of the current version.
- **Install** – During this deployment lifecycle event, the CodeDeploy agent copies the revision files from the temporary location to the final destination folder.
- **AfterInstall** – You can use this deployment lifecycle event for tasks such as configuring your application or changing file permissions.
- **ApplicationStart** – You typically use this deployment lifecycle event to restart services that were stopped during ApplicationStop.
- **ValidateService** – This is the last deployment lifecycle event. It is used to verify that the deployment was completed successfully.

- Structure of 'hooks' Section

```
hooks:
   deployment-lifecycle-event-name:
     - location: script-location
       timeout: timeout-in-seconds
       runas: user-name
```

Environment variables in CodeDeploy

```
APPLICATION_NAME
DEPLOYMENT_ID
DEPLOYMENT_GROUP_NAME
DEPLOYMENT_GROUP_ID
LIFECYCLE_EVENT
```

Orchestrate pipeline using CodeCommit,Codebuild,CodeDeploy and CodePipeline

Launching Dev and Prod EC2 instances

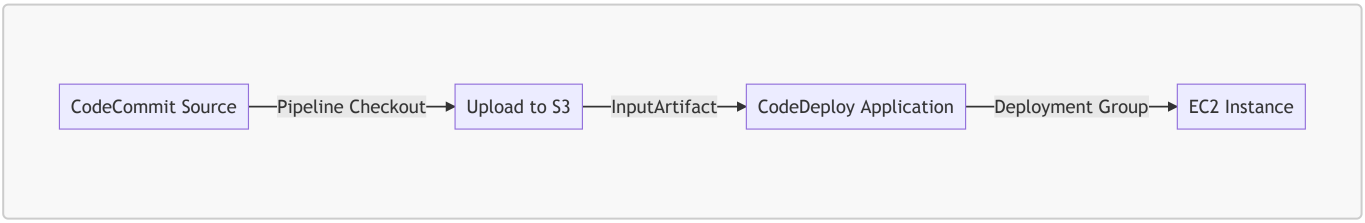
> Use CF template to create AWS Infra as per SDLC Environment.

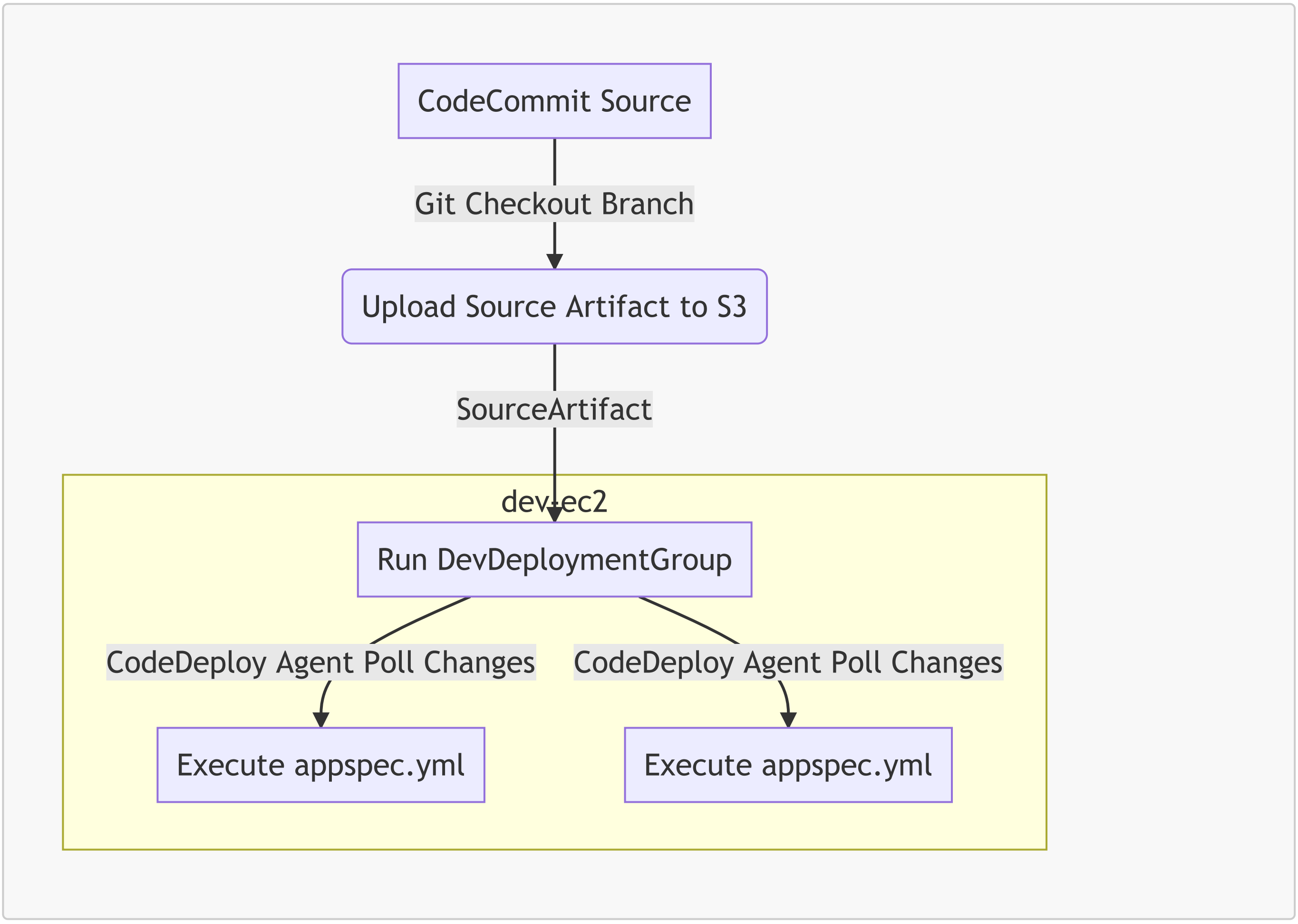
- Launch more two EC2 instances with CodeDeploy Agent installation Script in the UserData during Launch.
- Tag the new instances as `SDLC_ENVIRONMENT` : `prod`
- Go to the same CodeDeploy Application -> Create a new Deployment Group with name `ProdEC2DeploymentGroup` for the newly launched EC2 instances Tag Key:Value pair as `SDLC_ENVIRONMENT` : `prod` Tag.

Creating CICD Resources:

- Create below resources:
  - Configure CodeCommit IAM https credentials.
  - Create/Use exising CodeCommit Repo, push the source code to Codecommit Repo into master branch.
  - First for above setup, using CodeDeploy, we will first have our Code in CodeCommit.
  - Create a new AWS CodePipeline:
    - Navigate to AWS CodePipeline > Pipeline and click `Create pipeline` > `dev-ec2-deployment-pipeline`
    - Input the name for the service for CodePipeline and the S3 bucket that holds the artifacts for this pipeline.
    - Add a source stage such as `CodeCommit repository` and `branch` on which code version you want to deploy.
  - Skip the CodeBuild as of now.

- Add a deploy stage using `AWS CodeDeploy`. Select the CodeDeploy Application and Deployment group.
- Select the CodeDeploy Application name that we have already created, and select the `DevEC2DeploymentGroup`
- Review the Stages and Create the Pipeline.
- A **Cloudwatch Event Rule** is created in the background that will trigger the CodePipeline automatically when there is any change in specified branch of CodeCommit Repo. - Navigate and view the CloudWatch Event.
- If you make changes in the branch that is configured in cloudwatch event, the codepipeline will automatically.





Testing the Pipeline Executions

- Change some content in the source code file in the branch configured above, and push the changes in the CodeCommit Repo.
- This will trigger the CodePipeline with (Codecommit and CodeDeploy stage)
- Above CodePipeline Project will directly deploy anything that is pushed to Repo.

- Create a branch `develop` and `master` branch.

- Create another Pipeline with similar above stages and Configure the Cloudwatch Event to Run the ProdCodePipeline from master branch.

- Configure `develop` branch for dev-pipeline, and `master` branch for prod-pipeline.

- **Standard DevOps Approach to be followed:**

  - Make sure `DevCodePipeline` is triggered with `DevDeploymentGroup` only when there are code changes in `develop` branch, Also `ProdCodePipeline` is triggered with `ProdDeploymentGroup` only when there are code changes in `master` branch.
  - Changes will be made in child/feature branches, and once changes are merged into `develop` branch, the DevCodePipeline should trigger automatically to deploy changes done in develop branch.
  - Once the application is tested using develop and changes are deployment into Dev Environment, raise a Pull Request to merge from `develop` branch into `master`.
  - In a Continuous Delivery scenario.once code changes reviewed by requested Team Member, and code is merged into master branch, the `ProdCodePipeline` is automatically triggered.

- Add Build and Testing, edit one of the CodePipeline.

  - Lets add the CodeBuild Stage in the same CodePipeline.
  - Before that create a CodeBuild Project, and specify default configuration, and enter the buildspec.yml file path into CodeBuild Project.
  - Add a stage in CodePipeline by selecting the CodeBuild Project. `(Codecommit > Codebuild > CodeDeploy)`
  - Modify the InputArtifacts and OutputArtifacts as necessary for CodeBuild and CodeDeploy

- Release a change or make some changes in the source files.

- Adding necessary stages in the CodePipeline as below flow `(Codecommit > Codebuild > CodeDeploy(Dev) > Manual Approval > CodeDeploy(Prod))`

- Add a Stage CodeDeploy stage and action group "ProdDeployStage"

- Select the `ProdDeploymentGroup`

- Create a Manual Approval Action group Before CodeDeploy to Production in the Same Stage

- Provide URL of the Dev Instance to test the webpage.`this will be sequential`

- Provide appropriate comments, so that this will be a message sent to user who will approve.

- Create a Notification Rule for CodeBuild,CodeDeploy and CodePipeline. Specify the SNS Topic to get Notified on Execution Status.

Pipeline Execution and verify the webpage on each EC2 instances.

- Once Pipeline is successfully executed, verify the status of Deployments and test the Webpage the is available.

Troubleshooting Scenarios

- InstanceAgent::Plugins::CodeDeployPlugin::CommandPoller: Missing credentials - please check if this instance was started with an IAM instance profile
- Make Sure instance is started with IAM role attached,if you attach IAM Role to EC2 after the instance is already running, the Instance needs to be restarted. If codedeploy agent is installed and later IAM Role is attached, make sure Codedeploy agent is restarted or EC2 is restarted.