Development

This is where local development takes place on task/feature branches. All local development should run local testing against the application. Once the feature request has been completed and all tests have passed, the changes should be committed and pushed to GitHub. A Pull Request (PR) to master will then be submitted and approved by at least one other team member. On a successful merge to master, a Jenkins Cron Job will poll the master branch for any new commits and if any are found then the Commit Pipeline will begin.

Commit Pipeline

This pipeline will poll Github for any new commits to the master branch. If a new commit is found then it will start the pipeline and checkout the master branch. The pipeline will then provision a test database for testing at a later stage. It will then proceed to build the application, creating new Docker images at that point in time. Once this has completed, both the unit test suite and regression test suite against the new images. If successful, the images will then be tagged in the format [YYYYmmddHHMMSS-GitCommitHash] as a method of versioning. The tagged images will then be pushed to the relevant Elastic Container Registry (ECR) repository. As a final step, the pipeline will then tag the checked out commit with the same tag as the one used for the Docker images so they can be matched easily. This Git Tag once pushed via the pipeline will trigger the Deployment pipeline. On completion the pipeline will clean up the workspace. On failure the pipeline will send a slack notification to a specific slack channel to notify team members.

Temporary Manual Step

For a deployment to progress to the UAT or PROD environment, then a manual Git Tag will need to be pushed in the format of [YYYYmmddHHMMSS-PROFILE] where profile matches the desired environment for deployment.

Deployment Pipeline

This pipeline will poll Github for any new Tag commits to the master branch. If any new tags are found then it will start the pipeline and checkout the tagged commit from the master branch. This Git Tag will correspond to a matching Docker tag on a Docker image in ECR, and will then pull this image. It will then determine the environment to deploy to based on the tag suffix, if it is a profile then it will go to the desired profile but if it is a commit hash it will deploy to NonProd. The first step will take a snapshot of the current DB in case of any failures as a backup. It will then deploy any infrastructure via Terraform if the infrastructure does not already exist. After all infrastructure is successfully deployed, the application will be deployed to Kubernetes, any data updates to the database will be carried out. Finally a smoke test will run to ensure everything is working as expected. If the profile is Production then a slack notification will be sent on every success. In the case of any failures, a rollback will be implemented which will check out the previous version for that profile and download the corresponding Docker image. Re-deploy and infrastructure in case of changes and re-deploy the previous version of the application to Kubernetes. Finally it will revert any data changes and finish off with a smoke test to ensure the rollback was successful. The pipeline will send a slack notification on every failure, regardless of the profile.

CI/CD diagram for DoS Capacity Status API

2020/08/19 14:03:21