# Devops-CC Code Flow Documentation

**CODE FLOW DOCUMENTATION**

**INTRODUCTION:**

DevOps Platform provisions an Integrated Tool Chain and configures it with industry best practices. We use AWS as our cloud platform and use the serverless framework to automate the configurations, policies, etc on AWS. We use several AWS resources to build our platform which includes S3, Lambda, DynamoDB, IAM, Cognito, API Gateway, etc… Currently, we segregated our code in three repositories devops-platform, devops-platform-ui, tmp-devops-ecs.

**Repositories:**

**#devops-platform**

DevOps Platform provisions an Integrated Tool Chain and configures it with industry best practices.

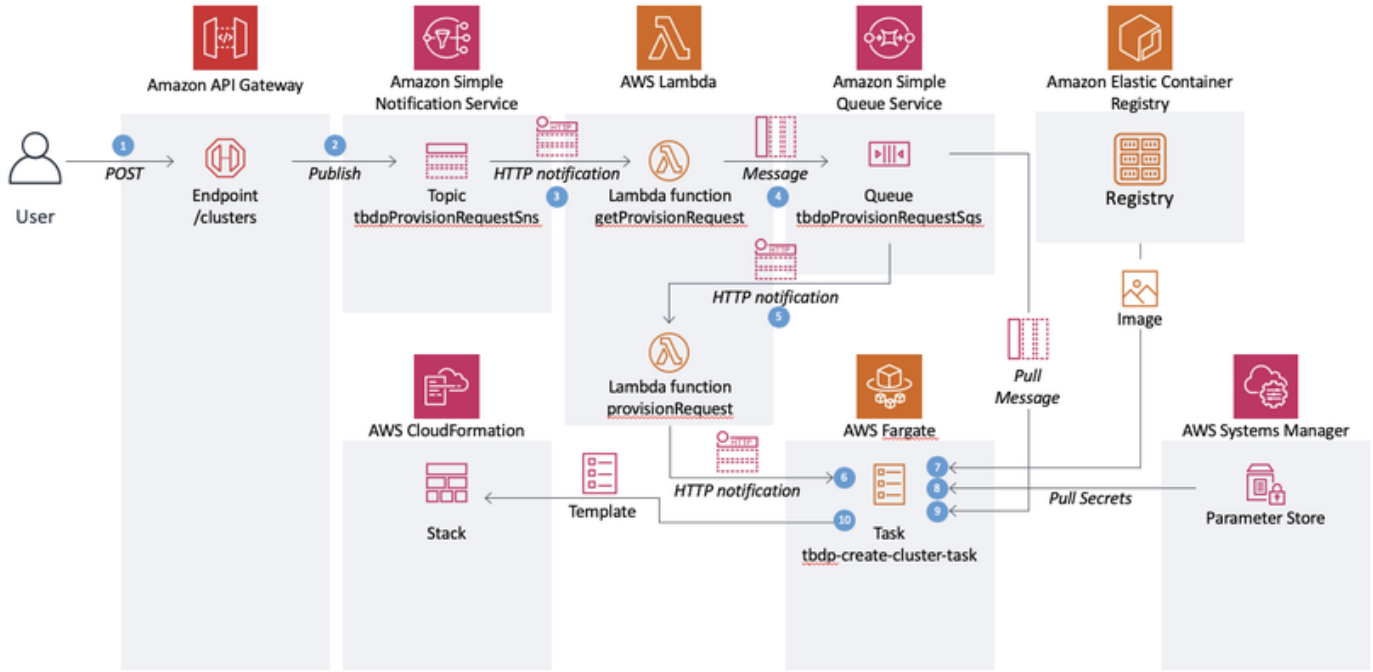
The backend stuff resides here.

**#devops-platform-ui**

This is UI for devops-platform.

**# tmp-devops-ecs**

This is a temporary repository to house the DevOps Platform Fargate ECS code. This will be refactored into the devops-platform repository in the near future.




**High Level Flow Chart**

**GET REQUEST**

**Summary:** Get all the clusters

**Description:** When the endpoint is hit by user or client an event is generated which triggers Lambda function and data gets fetched from Dynamo dB and the data is nothing but cluster payload and severed to the client.

**Overview:**

**Endpoint:** https://dev-api.dp.altimetrik.io/v1/clusters

**Serverless Function:** getClusters

Path: devops-platform\services\clusters\serverless.yml\functions

**End lambda function:** getAllClusters

Path: devops-platform\services\clusters\get.py\getAllClusters

Once the lambda function is triggered it will return a payload response of clusters.

**POST REQUEST**

**Summary:** Create a cluster with a set of tools.

**Description:** Create a post request of the cluster which contains a set of tools chosen by the client.

**Overview:**

**Endpoint:** https://dev-api.dp.altimetrik.io/v1/clusters

**Notification Service and Message Queue:** Generally, for dev purposes, we can directly do a post request without any message queue but in real-time we cannot predict how many requests are going to hit(maybe 100's or 100000's) so if we don't use a message queue this will result into a breakdown.Here comes Message queue to handle the incoming events.

**Cluster post event handling with SQS and SNS**

**Lambda Function**: devops-platform/post.py/getRequest

We create our SNS resource and SQS message queue:

**SNS resource:** dpProvisionRequestSns.

Description: Whenever an event is triggered, SNS publish is invoked.

**SQS Queue:** dpProvisionRequestSqs.

Description: Whenever SNS publish is done event is pushed to SQS queue.

**Dead Letter Queue:** dpProvisionRequestDLQ

Description: In any case if any event fails in SQS queue then the event is pushed into dead letter queue to redrive.

Create a redrive policy for failed messages and pass them to dead letter queue.


**Setting SQS Queue Policies**: ProvisionRequestSnsToSqsPolicy.

Description: Here we define our SQS policy.

**Creating SNS and SQS Subscription to publish messages from SNS to SQS**: dpProvisionRequestSqsSubscription.

Description: This subscribes SNS with SQS to publish messages from SNS to SQS.

**Licenses and selenium Layer:**

When a cluster is deployed with some tools, licensing of the tools is automated using Selenium with Python with chrome driver.

**Setup:**

This sets the **CloudFormation** template linked with SNS which is imported in devops-platform\services\clusters\cloudformation\provision.yml\dpProvisionEventSubscriptionToLambda.

**TMP-DEVOPS-ECS REPO**

**Overview:** This is a temporary repository to house the DevOps Platform Fargate ECS code. This will be refactored into the devops-platform repository in the near future. This particular repo we can even run it locally with some of the files added to this repo. The files that need to be added are docker.env, build-docker.sh, run-docker.sh, template-based-devops-platform.pem.

**Data_files:** The files here are processed when the request is 'None'. This contains two files cluster.yml and service_to_port.yml. Cluster basically consists of cluster information and service to ports contains cluster tools information.

**Path:** tmp-devops-ecs\data_files

**Roles:** This particularly holds the tools and their default configuration we provide when we provision them and also contain platform configuration tools like swarm manager which creates a docker image for our clusters and Grafana, Kibana which are used for monitoring purposes.

**Path:** tmp-devops-ecs\roles

**Templates:** This generally contains the tool configurations which need to be configured in AWS.

**Path:** tmp-devops-ecs\templates

**DevOps-setup.py:** Here stays the lambda part where everything is hooked that is a lambda, ansible, and other stuff.

**Path:** tmp-devops-ecs\devops-setup.py