## Title:

Migration of Apigee proxies and other configs using Azure DevOps CI/CD

## Problem Statement

Apigee is a powerful API management platform that allows organizations to design, secure, and scale APIs. Migrating Apigee proxies and configurations between environments can be a challenging task but leveraging Azure DevOps CI/CD (Continuous Integration and Continuous Deployment) can greatly streamline the process.

## Tech Approach

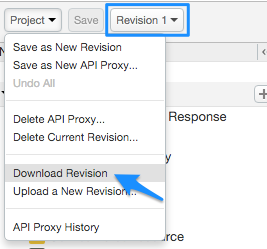
For smooth migration of API proxies using CICD, we use the maven deploy plugin to deploy the proxies and their configuration. For migration, we need to get the proxy bundle from the APIGEE UI. Then after the Maven plugin deploys the proxy in the given environment.

## Prerequisites

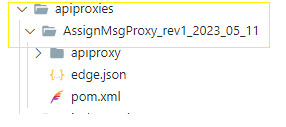
1. To trigger the pipeline, we need to push the code in develop, initial-branch and main branch. Other branches will be ignored.
2. The following variables are predefined. in the Azure DevOps pipeline, if needed you can change as per your requirement.
3. Env
4. Org
5. GCP\_SERVICE\_ACCOUNT

## Procedure

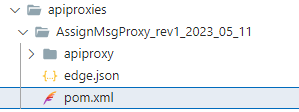
1. Download the proxy bundle from Apigee UI
2. Sign in to [**apigee.com/edge**](https://apigee.com/edge)
3. Select **Develop > API Proxies** in the left navigation bar.
4. Click the API proxy in the list that you want to migrate.
5. Select the proxy revision that you want to download.
6. Select **Project > Download** Revision.



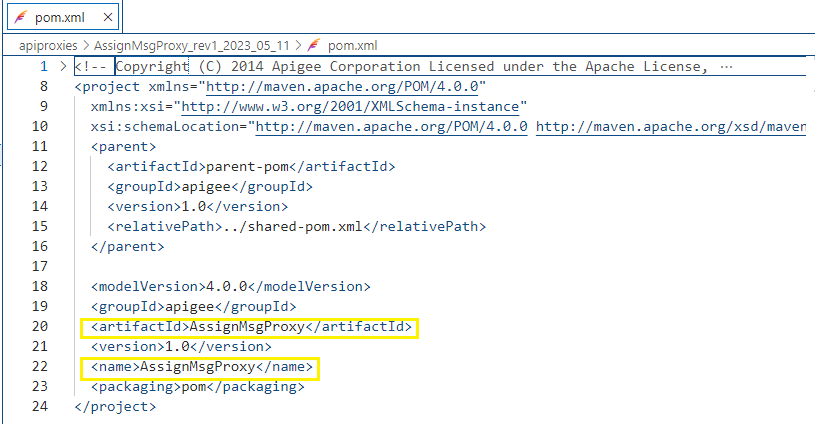
1. Unzip the bundle and place it into the repo’s apiproxies folder.



1. Create a pom file inside the proxy folder with the given sample file.



1. Change the parameters such as name and artifact-id in the pom file.

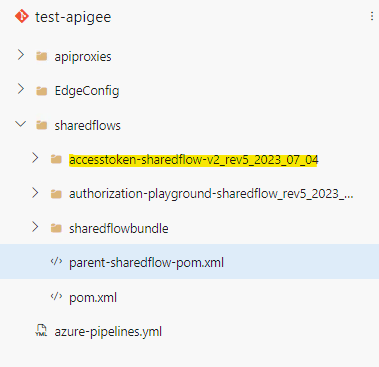


1. Add config like *KVM* and *Targetserver* in the config in the [*edge*.*json*](https://dev.azure.com/digitalrealtyprojects/Global%20API/_git/test-apigee?path=/apiproxies/AssignMsgProxy_rev1_2023_05_11/edge.json) if needed –

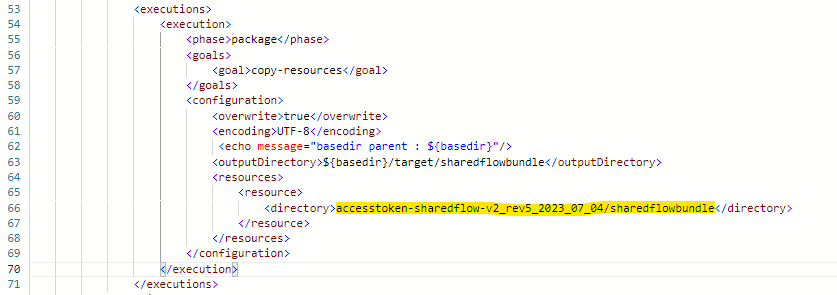
File path **- /apiproxies/${proxy\_name} /edge.json**



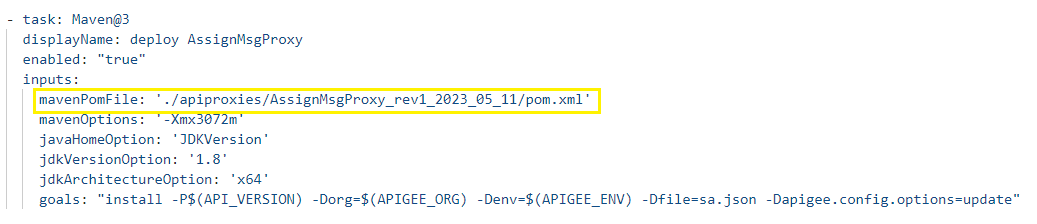
1. Deploy *sharedflow* if needed to the pipeline file -
2. Download the sharedflow bundle and unzip into the sharedflow folder as follows  
   [(./sharedflows/accesstoken-sharedflow-v2\_rev5\_2023\_07\_04](https://dev.azure.com/digitalrealtyprojects/Global%20API/_git/test-apigee?path=/sharedflows/accesstoken-sharedflow-v2_rev5_2023_07_04&version=GBinitial-branch&_a=contents)) –



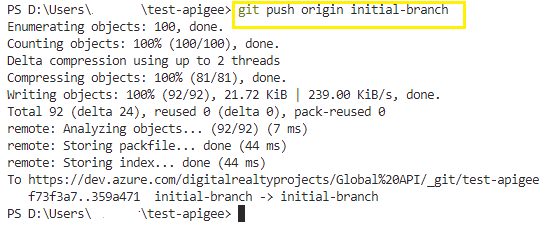
1. Give the sharedflow bundle path in the [parent-sharedflow-pom.xml](https://dev.azure.com/digitalrealtyprojects/Global%20API/_git/test-apigee?path=/sharedflows/parent-sharedflow-pom.xml&version=GBinitial-branch&line=66&lineEnd=66&lineStartColumn=37&lineEndColumn=118&lineStyle=plain&_a=contents) file



1. Create a task inside the [*azure-pipeline.yml*](https://dev.azure.com/digitalrealtyprojects/Global%20API/_git/test-apigee?path=/azure-pipelines.yml&version=GBinitial-branch&line=52&lineEnd=62&lineStartColumn=1&lineEndColumn=1&lineStyle=plain&_a=contents) file for deploying the proxy. Provide the pom file path in the parameter.



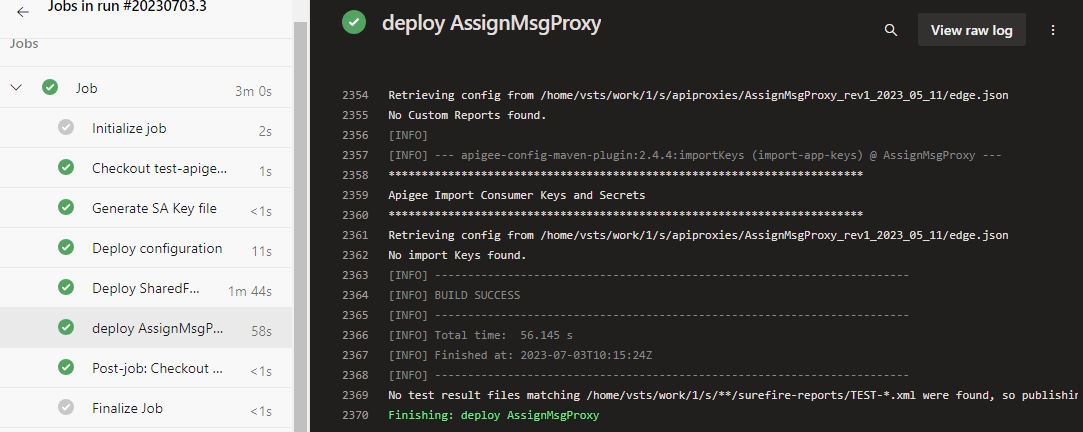
1. Push the code into the repo and pipeline will trigger automatically.



## Solution Outcome

By leveraging Azure DevOps CI/CD for Apigee proxy’s migration, you can achieve a streamlined, repeatable, and reliable deployment process, reducing manual errors and enhancing the overall efficiency of your API management workflows.

After successful deployment, Apigee proxy should be migrated. And you will get the following result.



## Reference Documents

1. [apigee-config-maven-plugin](https://github.com/apigee/apigee-config-maven-plugin)
2. [DLR-test-apigee](https://dev.azure.com/digitalrealtyprojects/Global%20API/_git/test-apigee) repo