# 

Apigee

Maven Deployment Set-up

**Table of Contents**

Objective 3

Pre-requisites 3

How to 3

Structure of the config.json 7

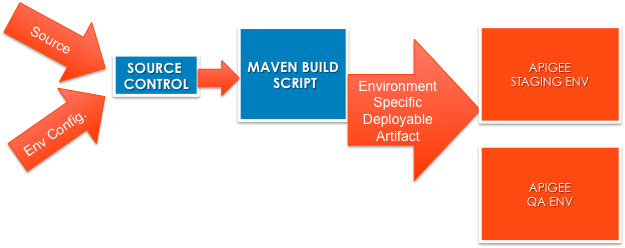
Commands 8

References and Code Samples 9

Appendix A: Maven Installation for Windows 10

# Objective

1. To automate deployment of Apigee application configuration and sources to target Gateway
2. To be able to automatically replace environment specific configurations based on the target environment. E.g. rate limiting in a production environment may be set differently from development environment. The deployment profiles should take care of environment specific configurations
3. Ability to test an application end-point as part of automated deployment



# Pre-requisites

1. Java(TM) SE Runtime Environment 1.6 or later
2. Apache Maven 3.0.3 or later
3. Read access to Apigee artifactory maven repository (maven settings.xml will need the credentials)
4. Access to Apigee Enterprise or On-Premise (4G) environment over HTTP

# How to

The system, from which the build will be run, must satisfy pre-requisites #1 to #3. The maven settings.xml file should have the following entries

<?xml version="1.0" encoding="UTF-8"?>

<settings xmlns="http://maven.apache.org/SETTINGS/1.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:schemaLocation="http://maven.apache.org/SETTINGS/1.0.0 http://maven.apache.org/xsd/settings-1.0.0.xsd">

<profiles>

<profile>

<repositories>

<repository>

<snapshots>

<enabled>false</enabled>

</snapshots>

<id>apigee-cs-libs-release</id>

<name>libs-release</name>

<url>http://repo.apigee.com:8081/artifactory/repo</url>

</repository>

</repositories>

<pluginRepositories>

<pluginRepository>

<snapshots>

<enabled>false</enabled>

</snapshots>

<id>apigee-cs-plugins-release</id>

<name>plugins-release</name>

<url>http://repo.apigee.com:8081/artifactory/repo</url>

</pluginRepository>

</pluginRepositories>

<id>apigee-artifactory</id>

</profile>

</profiles>

<servers>

<server>

<id>apigee-cs-libs-release</id>

<username>client</username>

<password>\*\*\*\*\*\*\*\*</password> <!-- obtain your access from apigee ops -->

</server>

<server>

<id>apigee-cs-plugins-release</id>

<username>client</username>

<password>\*\*\*\*\*\*\*\*</password> <!-- obtain your access from apigee ops -->

</server>

</servers>

<activeProfiles>

<activeProfile>apigee-artifactory</activeProfile>

</activeProfiles>

<pluginGroups>

<pluginGroup>com.apigee.cs</pluginGroup>

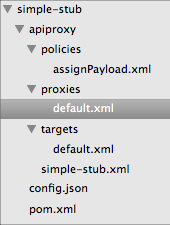
<pluginGroup>com.apigee.build-tools.enterprise4g</pluginGroup>

</pluginGroups>

</settings>

Download the sample project from the reference [1] below. It has a structure like below.

Here the simple-stub folder is the root of the project folder.



Within the project folder the config.json contains all the environment specific configuration information.

There is also a parent level pom that has the details of apigee build profile. At the time of the build the profile name should be provided so that the script can generate the respective build artifact.

<?xml version="1.0"?>

<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.xsd">

<modelVersion>4.0.0</modelVersion>

<groupId>apigee</groupId>

<artifactId>parent-pom</artifactId>

<packaging>pom</packaging>

<version>1.0</version>

<pluginRepositories>

<pluginRepository>

<id>central</id>

<name>Maven Plugin Repository</name>

<url>http://repo1.maven.org/maven2</url>

<layout>default</layout>

<snapshots>

<enabled>false</enabled>

</snapshots>

<releases>

<updatePolicy>never</updatePolicy>

</releases>

</pluginRepository>

<pluginRepository>

<id>apigee-cs-repo</id>

<url>http://repo.apigee.com:8081/artifactory/repo</url>

<snapshots>

<enabled>true</enabled>

</snapshots>

<releases>

<updatePolicy>never</updatePolicy>

</releases>

</pluginRepository>

</pluginRepositories>

<modules>

<module>../sample-stub</module>

</modules>

<build>

<pluginManagement>

<plugins>

<plugin>

<groupId>com.apigee.build-tools.enterprise4g</groupId>

<artifactId>4G-gateway-maven-build-pack</artifactId>

<version>0.0.9</version>

</plugin>

</plugins>

</pluginManagement>

<plugins>

<plugin>

<artifactId>maven-resources-plugin</artifactId>

<version>2.3</version>

<executions>

<execution>

<id>copy-resources-step</id>

<phase>package</phase>

<goals>

<goal>copy-resources</goal>

</goals>

<configuration>

<!-- this is important -->

<overwrite>true</overwrite>

<!-- target -->

<outputDirectory>${basedir}/target/apiproxy</outputDirectory>

<resources>

<resource>

<!-- source -->

<directory>apiproxy</directory>

</resource>

</resources>

</configuration>

</execution>

</executions>

</plugin>

<plugin>

<groupId>com.apigee.build-tools.enterprise4g</groupId>

<artifactId>4G-gateway-maven-build-pack</artifactId>

<configuration>

<skip>true</skip>

</configuration>

</plugin>

<plugin>

<groupId>org.apache.maven.plugins</groupId>

<artifactId>maven-compiler-plugin</artifactId>

<version>2.3.2</version>

<configuration>

<source>1.6</source>

<target>1.6</target>

</configuration>

</plugin>

<plugin>

<groupId>org.apache.maven.plugins</groupId>

<artifactId>maven-surefire-plugin</artifactId>

<version>2.9</version>

<configuration>

<testFailureIgnore>false</testFailureIgnore>

</configuration>

</plugin>

</plugins>

</build>

<!-- This is where you add the environment specific properties under various profile names -->

<profiles>

<profile>

<id>prod</id> <!-- this is used in the maven command -->

<properties>

<apigee.profile>prod</apigee.profile> <!-- this is profile name corresponds to the config.jsob -->

<apigee.hosturl>https://api.enterprise.apigee.com</apigee.hosturl>

<apigee.apiversion>v1</apigee.apiversion>

<apigee.org>apigee-cs</apigee.org>

<apigee.env>prod</apigee.env> <!-- this is same as environment name in the 4G org -->

<apigee.username>${username}</apigee.username>

<apigee.password>${password} </apigee.password>

<apigee.override.delay>10</apigee.override.delay>

<apigee.delay>1000</apigee.delay>

<apigee.options>clean,validate</apigee.options>

</properties>

</profile>

</profiles>

<dependencies>

<dependency>

<groupId>junit</groupId>

<artifactId>junit</artifactId>

<version>4.8.2</version>

<scope>test</scope>

</dependency>

</dependencies>

<repositories>

<repository>

<id>apigee-cs-repo</id>

<url>http://repo.apigee.com:8081/artifactory/repo</url>

</repository>

</repositories>

</project>

## Structure of the config.json

The config.json contains configuration for each apigee.profile as mentioned in the parent pom.



Each configuration contains an xpath and corresponding value for that target xpath. During the build maven plugin replaces this values based on the profile.

## Build Commands

**Standard build command**

# cd <root of the project folder >

# mvn apigee-enterprise:deploy -P{profile-id as in parent pom} –Dusername=< your apigee edge login > -Dpassword=<your apigee edge password>

For the sample bundle the sample command would be :

mvn apigee-enterprise:deploy -Pprod [–Dusername=rajeshmishra@apigee.com](mailto:–Dusername=rajeshmishra@apigee.com) -Dpassword=crackme

**More Options for standard build**

1. A parameter called **apigee.options** can be used with the standard build command. Following are the available values for the parameters.
2. clean - This will first un-deploy and then delete the last deployed revision in an environment.

mvn apigee-enterprise:deploy -P{profile-id as in parent pom} -Dapigee.options=clean –Dusername=<your apigee edge login> -Dpassword=<your apigee edge password>

1. override - This is used for seamless (zero down time) deployment . This must be supplied with apigee.override.delay parameter. The apigee.override.delay expects delay to be given in seconds.

mvn apigee-enterprise:deploy -P{profile-id as in parent pom} -Dapigee.options=override –Dusername=<your apigee edge login> -Dpassword=<your apigee edge password>

1. update - It will update the deployed revision .  This is similar to import with validation but no new revision is created. If there any errors in the bundle, error is thrown and the existing bundle is left intact. In case the revision they are trying to update is deployed, it will internally trigger undeployment and deployment. It is completely in the background and not visible in the response. ***It is advised not to update the deployed revision***. Because in case of errors during the update process the existing revision may transition to undeployed state.

mvn apigee-enterprise:deploy -P{profile-id as in parent pom} -Dapigee.options=update –Dusername=<your apigee edge login> -Dpassword=<your apigee edge password>

1. validate - This will validate a bundle before importing . Thus if you want strict validation then its required.
2. inactive - This will just import the bundle without activating the bundle.
3. You can use validate option in combination with other options by passing multiple options as comma separated values.

mvn apigee-enterprise:deploy -P{profile-id as in parent pom} -Dapigee.options=override,validate –Dusername=<your apigee edge login> -Dpassword=<your apigee edge password>

Please note that you can not use **clean, override, inactive & update** options with each other. This means that if you specify apigee.options as **override,clean,validate**  only **one of the options** will be triggered.

1. You can specify the  apigee.delay (in milliseconds) . This will ensure to add a delay between the operations like delete, import, activate, deactivate etc . The *apigee.override.delay apigee.delay, apigee.options* are optional parameters.

**Deactivate (Undeploy) the last active build**

mvn apigee-enterprise:deploy -Ptest1 -Dbuild.option=undeploy –Dusername=< your apigee edge login > -Dpassword=<your apigee edge password>

Note: This is Deprecated, use *-Dapigee.options=clean* instead

**Deploy a new build without activation ( Previous build remains avtivated)**

mvn apigee-enterprise:deploy -Ptest1 -Dbuild.option=deploy-inactive –Dusername=< your apigee edge login > -Dpassword=<your apigee edge password>

Note: This is Deprecated, use *-Dapigee.options=inactive* instead

# References and Code Samples

Github apigeecs folder.

## https://github.com/apigeecs/devops-tools/tree/master/4G-gateway-maven-build-pack/samples Appendix A: Maven Installation for Windows

1. Download Maven http://apache.ziply.com//maven/binaries/apache-maven-3.0.3-bin.zip  
2. Extract to say - c:\apigee\tools\maven   
3. set maven home to your environment variables (user or system level)  
    MAVEN\_HOME = c:\apigee\tools\maven\  
4. set java home if not already set  
   e.g., JAVA\_HOME = c:\jdk\  
5. Done, to verify it, in command prompt, type “mvn –version” to verify the installation detail.  
    If you see similar message as below, means your Apache Maven is installed successful on Windows.  
    Apache Maven 2.2.1 (r801777; 2009-08-07 00:46:01+0530)  
    Java version: 1.6.0\_26  
    Java home: C:\Program Files\Java\jdk1.6.0\_26\jre  
    Default locale: en\_IN, platform encoding: Cp1252  
    OS name: "windows 7" version: "6.1" arch: "amd64" Family: "windows"

**Note**: MacOS already contains a maven installation