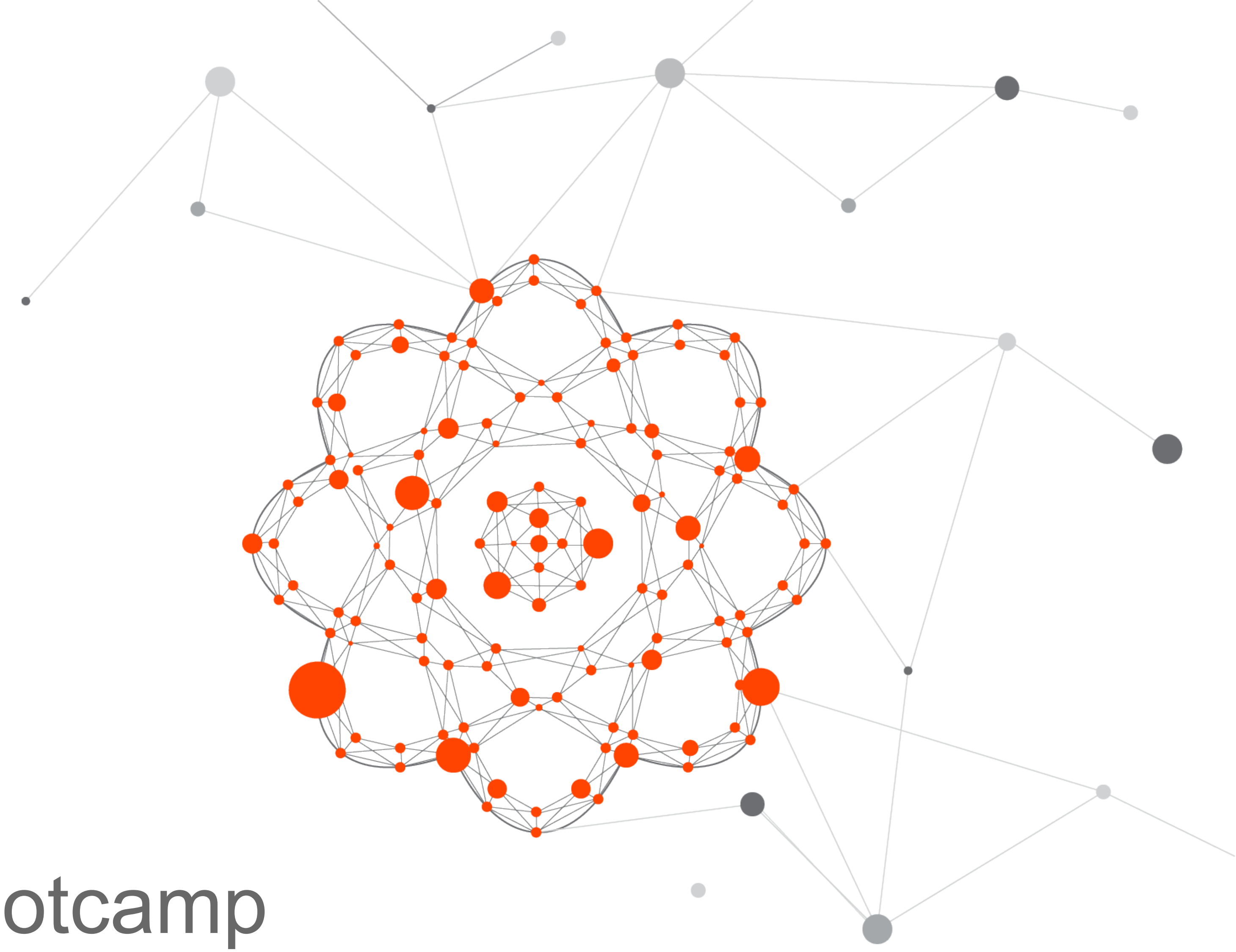
Aug, 2015

Developer Bootcamp

API Lifecycle Components



Description

Outcome:

In this session, you will learn about components that make up a subset of the API lifecycle. Advanced tools that enable offline development and deployment will be discussed, followed by API Documentation tools and practices.

You will learn:

- the importance of API build and Deployment tools
- how to use Maven for Apigee Edge offline build/deployments
- about documenting your APIs using Swagger and Apigee Smartdocs

Course Topics

Offline Development and Deploy

- apigeetool
- Apigee Deploy Maven Plugin

API Documentation

- Swagger
- SmartDocs

Offline Development and Deploy

Deploy Tools

A common tool used is:

apigeetool

- ✓ Easy to learn
- ✓ Short time to install
- ✓ Shell based tool
- ✓ Small foot print

- x No native Life Cycle
- x No dependency management
- x Lower reusability
- x Harder to transition to "Continuous Integration"
- x No integration with IDEs
- x Inconsistent. Every tool in unix has different syntax
- x Not 100% portable
- x No community

apigeetool

1. Clone repo https://github.com/apigee/api-platform-tools
2. Requires Python https://www.python.org/downloads/
3. Command:

apigeetool -n {apiName} -u {myname:mypass} -o {myorg} -e {environment} -b {basePath} -d {path to /apiproxy directory} –h {base URL}

apigeetool -n forecastweatherapi -u $ae_username:$ae_password -o testmyapi -e test -b /weather -d .

Apigee's Maven Plugin

- ✓ More time to focus on what really matters by automating repetitive tasks
- ✓ Innovation ready. Extensible plugin-based platform
- ✓ Promotes productivity. Promotes usage of CLI (Command-Line Interface). No need for IDEs
- ✓ Easy to adopt. No need of CLI. Eclipse IDE Support through M2E and IntellijIDEA, WebStorm
- ✓ Easy to configure and to track changes. All of its artifacts can live in version control as text files
- ✓ Multilanguage support. One JVM to rule them all (Ruby, Jython, JavaScript, Groovy, Scala) or even Shell scripts
- ✓ Tens Thousands plugins ready in Maven Central
- ✓ Backed up by Apigee and the open source community

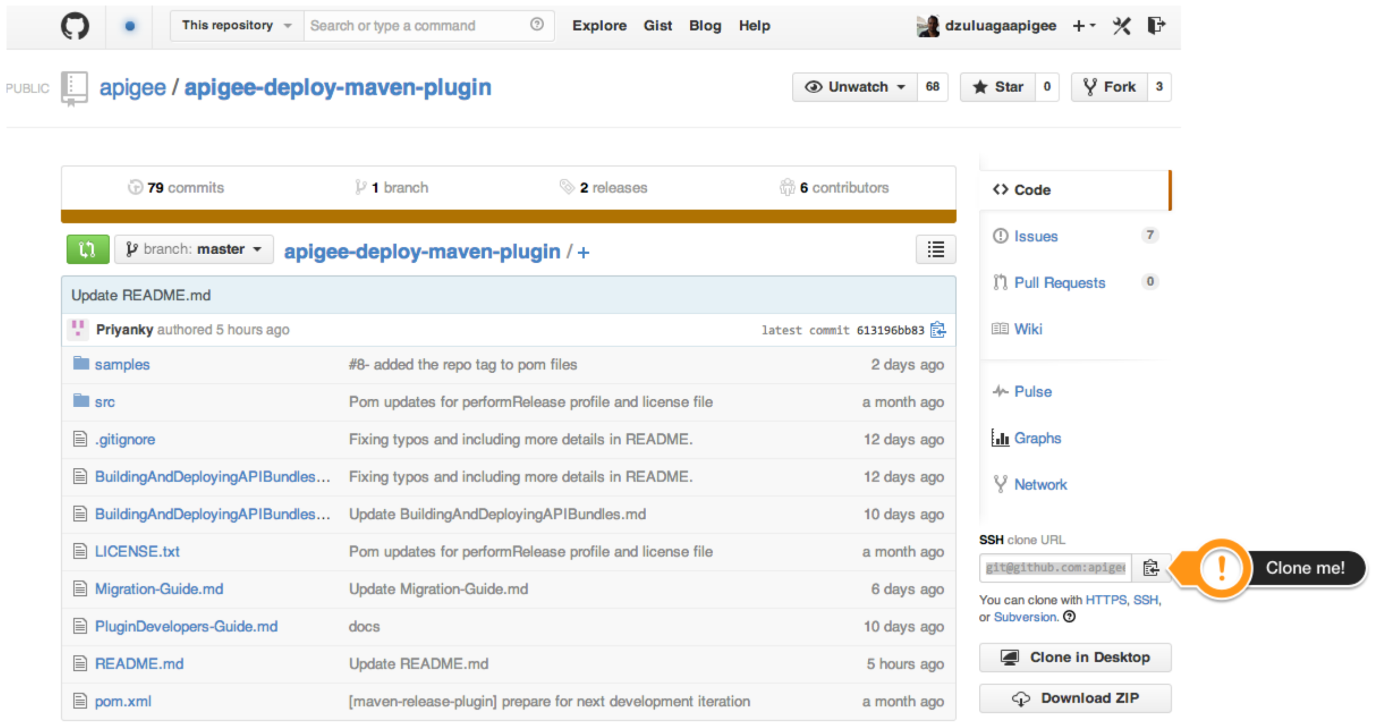
Apigee Maven Plugin Prerequisites

What you will need:

- Java(TM) SE Runtime Environment 1.6 or later
- Apache Maven 3.0.+ http://maven.apache.org/download.cgi#Maven_3.0.5
- Access/Perms to deploy to Apigee Edge over HTTPS

Download Maven Plugin Artifacts

- Get samples first by cloning this repo
  https://github.com/apigee/apigee-deploy-maven-plugin



This repository | Search or type a command | Explore | Gist | Blog | Help | dzuluagaapigee

PUBLIC apigee / **apigee-deploy-maven-plugin**

Unwatch 68 | Star 0 | Fork 3

79 commits | **1** branch | **2** releases | **6** contributors

branch: **master** | apigee-deploy-maven-plugin / +

Update README.md

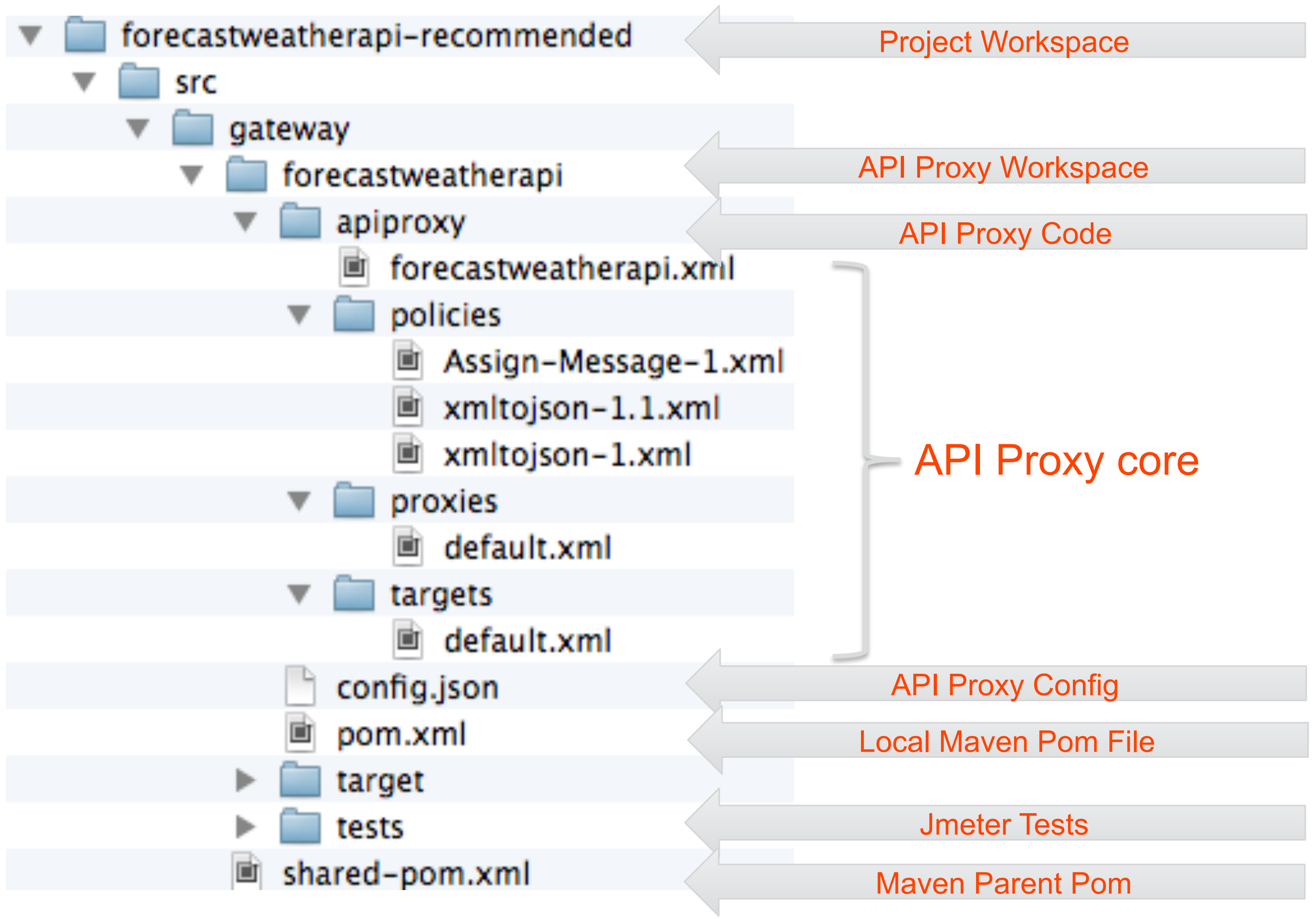
Priyanky authored 5 hours ago — latest commit 613196bb83

samples	#8- added the repo tag to pom files	2 days ago
src	Pom updates for performRelease profile and license file	a month ago
.gitignore	Fixing typos and including more details in README.	12 days ago
BuildingAndDeployingAPIBundles…	Fixing typos and including more details in README.	12 days ago
BuildingAndDeployingAPIBundles…	Update BuildingAndDeployingAPIBundles.md	10 days ago
LICENSE.txt	Pom updates for performRelease profile and license file	a month ago
Migration-Guide.md	Update Migration-Guide.md	6 days ago
PluginDevelopers-Guide.md	docs	10 days ago
README.md	Update README.md	5 hours ago
pom.xml	[maven-release-plugin] prepare for next development iteration	a month ago

SSH clone URL: git@github.com:apigee

You can clone with HTTPS, SSH, or Subversion.

Structure Your Offline Code

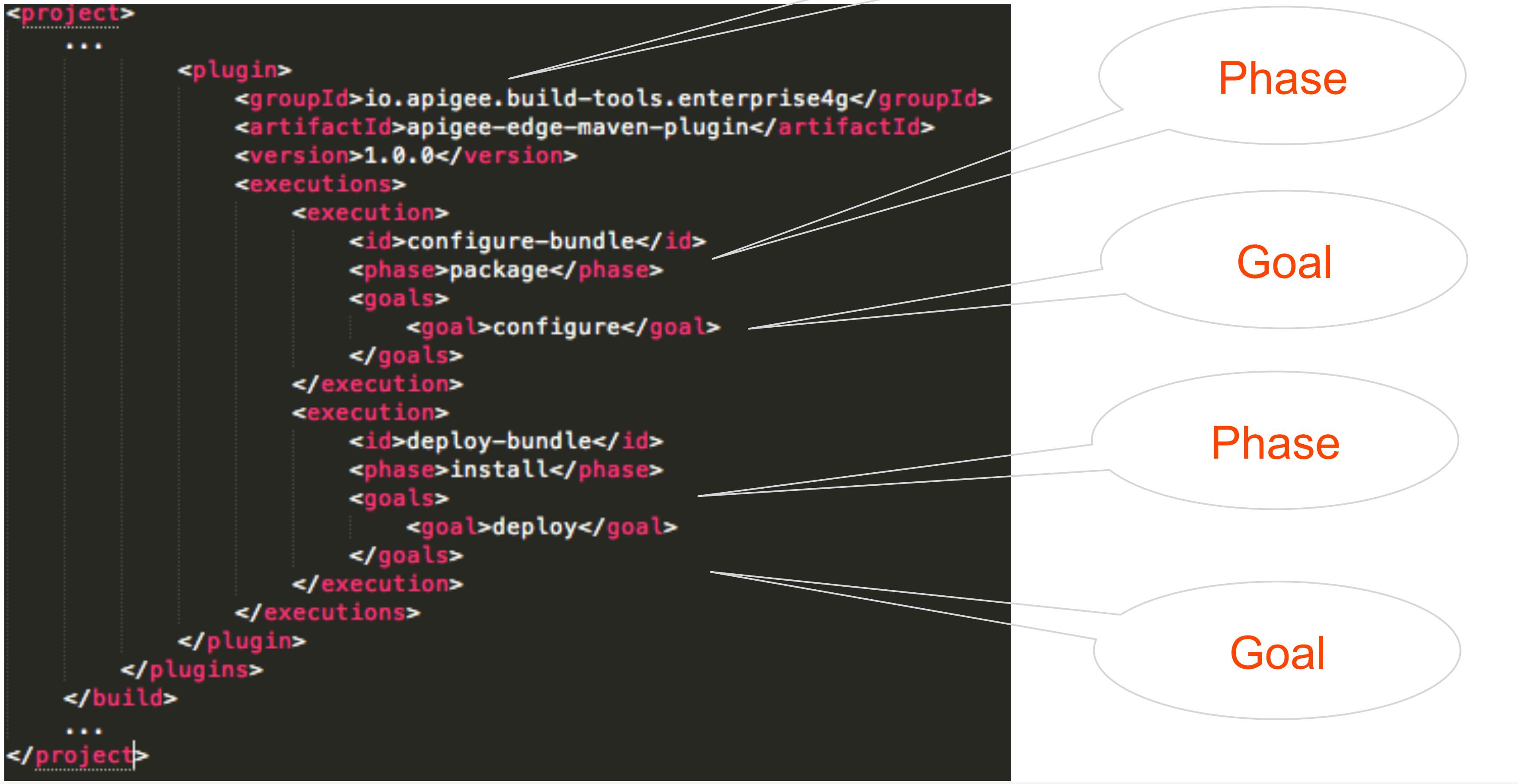


```
▼ forecastweatherapi-recommended
    ▼ src
        ▼ gateway
            ▼ forecastweatherapi
                ▼ apiproxy
                      forecastweatherapi.xml
                    ▼ policies
                          Assign-Message-1.xml
                          xmltojson-1.1.xml
                          xmltojson-1.xml
                    ▼ proxies
                          default.xml
                    ▼ targets
                          default.xml
                  config.json
                  pom.xml
                ▶ target
                ▶ tests
              shared-pom.xml
```

Maven Plugin Components – Parent POM

The maven parent pom file (shared-pom.xml) contains the maven configuration that can be used across all Apigee Edge API proxies.

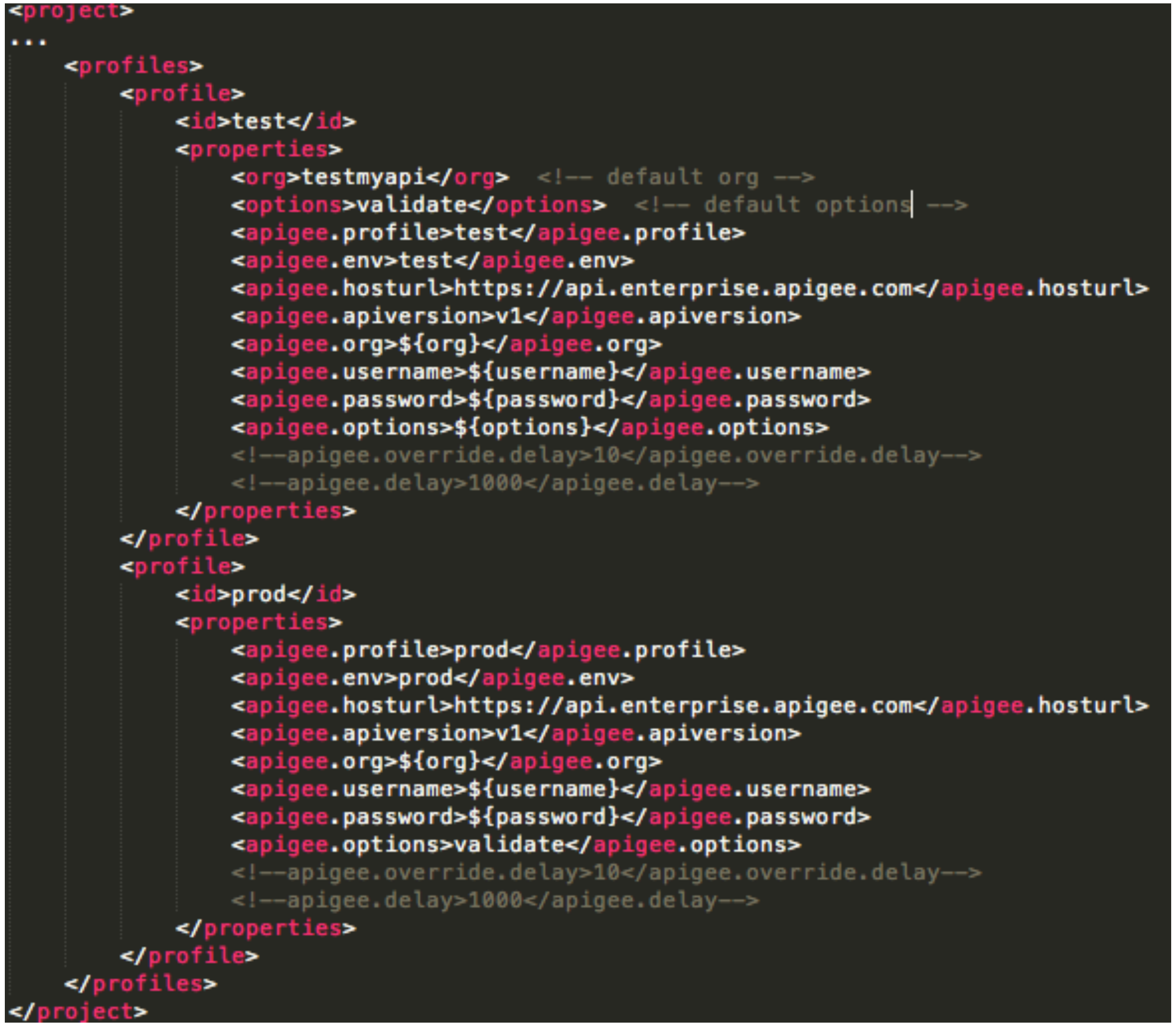
- **Location:** src/gateway
- Contains common dependencies across all bundles
- Enables common behavior through inheritance



```
<project>
    ...
            <plugin>
                <groupId>io.apigee.build-tools.enterprise4g</groupId>
                <artifactId>apigee-edge-maven-plugin</artifactId>
                <version>1.0.0</version>
                <executions>
                    <execution>
                        <id>configure-bundle</id>
                        <phase>package</phase>
                        <goals>
                            <goal>configure</goal>
                        </goals>
                    </execution>
                    <execution>
                        <id>deploy-bundle</id>
                        <phase>install</phase>
                        <goals>
                            <goal>deploy</goal>
                        </goals>
                    </execution>
                </executions>
            </plugin>
        </plugins>
    </build>
    ...
</project>
```

Maven Profiles Configuration – Parent POM

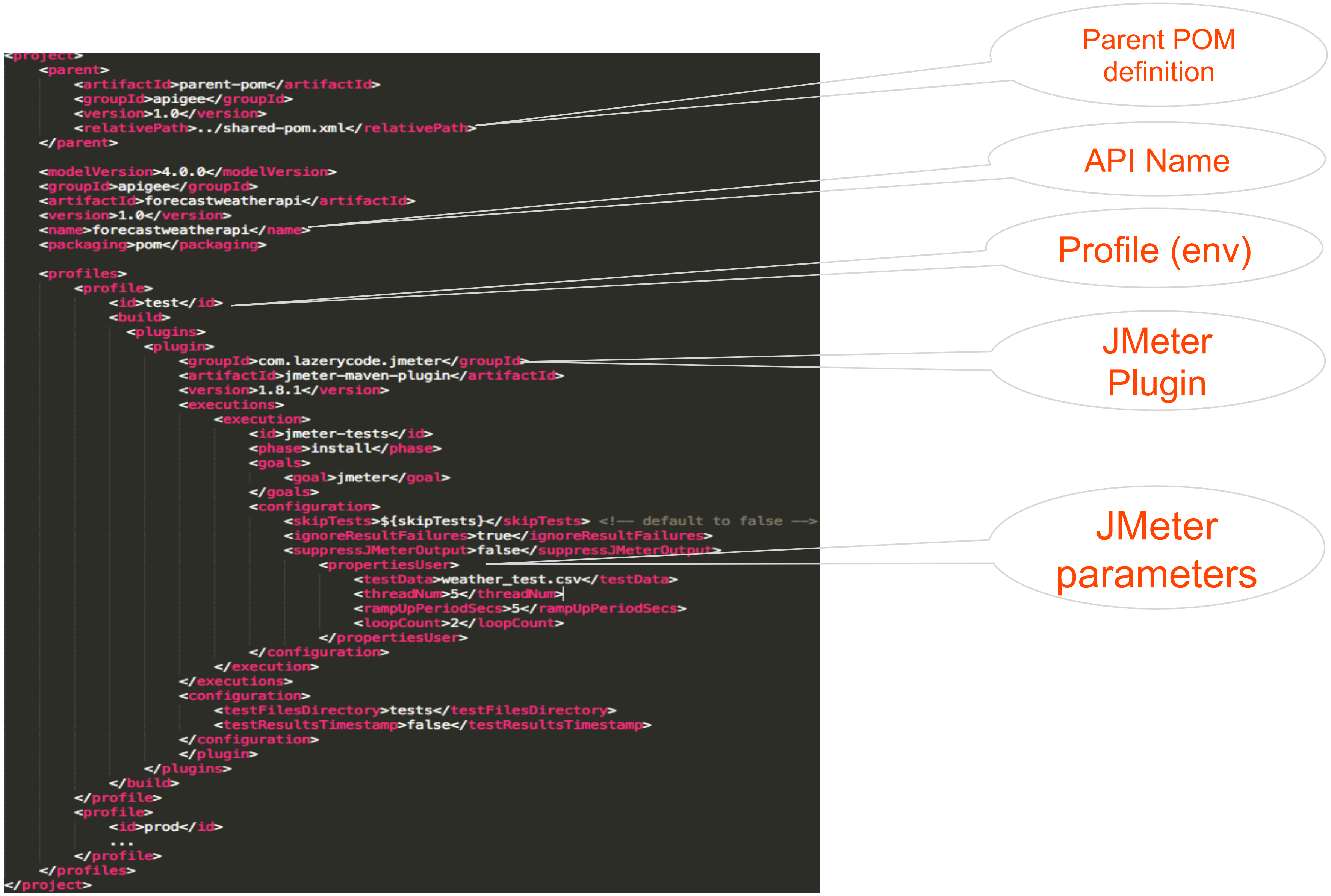
Also in the maven parent pom file (shared-pom.xml), you should include Apigee Edge organization and environment configurations as these are common to all APIs. These are defined using maven ***profiles***.



```xml
<project>
...
    <profiles>
        <profile>
            <id>test</id>
            <properties>
                <org>testmyapi</org>  <!-- default org -->
                <options>validate</options>  <!-- default options -->
                <apigee.profile>test</apigee.profile>
                <apigee.env>test</apigee.env>
                <apigee.hosturl>https://api.enterprise.apigee.com</apigee.hosturl>
                <apigee.apiversion>v1</apigee.apiversion>
                <apigee.org>${org}</apigee.org>
                <apigee.username>${username}</apigee.username>
                <apigee.password>${password}</apigee.password>
                <apigee.options>${options}</apigee.options>
                <!--apigee.override.delay>10</apigee.override.delay-->
                <!--apigee.delay>1000</apigee.delay-->
            </properties>
        </profile>
        <profile>
            <id>prod</id>
            <properties>
                <apigee.profile>prod</apigee.profile>
                <apigee.env>prod</apigee.env>
                <apigee.hosturl>https://api.enterprise.apigee.com</apigee.hosturl>
                <apigee.apiversion>v1</apigee.apiversion>
                <apigee.org>${org}</apigee.org>
                <apigee.username>${username}</apigee.username>
                <apigee.password>${password}</apigee.password>
                <apigee.options>validate</apigee.options>
                <!--apigee.override.delay>10</apigee.override.delay-->
                <!--apigee.delay>1000</apigee.delay-->
            </properties>
        </profile>
    </profiles>
</project>
```

Specific Maven Config API Proxy Local POM

- Local POM - pom.xml



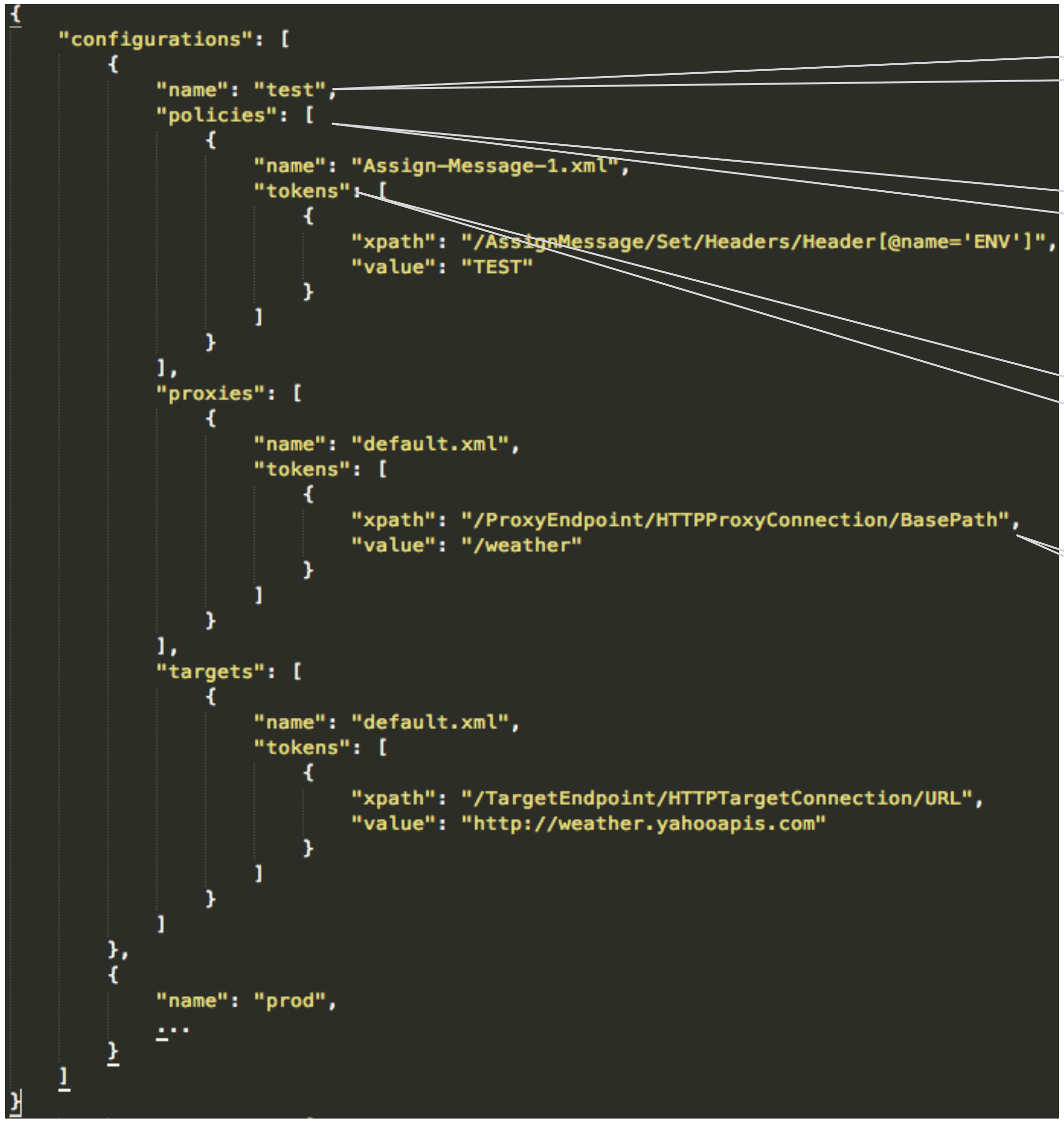
```xml
<project>
    <parent>
        <artifactId>parent-pom</artifactId>
        <groupId>apigee</groupId>
        <version>1.0</version>
        <relativePath>../shared-pom.xml</relativePath>
    </parent>

    <modelVersion>4.0.0</modelVersion>
    <groupId>apigee</groupId>
    <artifactId>forecastweatherapi</artifactId>
    <version>1.0</version>
    <name>forecastweatherapi</name>
    <packaging>pom</packaging>

    <profiles>
        <profile>
            <id>test</id>
            <build>
              <plugins>
                <plugin>
                    <groupId>com.lazerycode.jmeter</groupId>
                    <artifactId>jmeter-maven-plugin</artifactId>
                    <version>1.8.1</version>
                    <executions>
                        <execution>
                            <id>jmeter-tests</id>
                            <phase>install</phase>
                            <goals>
                                <goal>jmeter</goal>
                            </goals>
                            <configuration>
                                <skipTests>${skipTests}</skipTests> <!-- default to false -->
                                <ignoreResultFailures>true</ignoreResultFailures>
                                <suppressJMeterOutput>false</suppressJMeterOutput>
                                    <propertiesUser>
                                        <testData>weather_test.csv</testData>
                                        <threadNum>5</threadNum>
                                        <rampUpPeriodSecs>5</rampUpPeriodSecs>
                                        <loopCount>2</loopCount>
                                    </propertiesUser>
                            </configuration>
                        </execution>
                    </executions>
                    <configuration>
                        <testFilesDirectory>tests</testFilesDirectory>
                        <testResultsTimestamp>false</testResultsTimestamp>
                    </configuration>
                    </plugin>
                </plugins>
            </build>
        </profile>
        <profile>
            <id>prod</id>
            ...
        </profile>
    </profiles>
</project>
```

Applying API Proxy Configs w/ config.json

- JSON based



```
{
    "configurations": [
        {
            "name": "test",
            "policies": [
                {
                    "name": "Assign-Message-1.xml",
                    "tokens": [
                        {
                            "xpath": "/AssignMessage/Set/Headers/Header[@name='ENV']",
                            "value": "TEST"
                        }
                    ]
                }
            ],
            "proxies": [
                {
                    "name": "default.xml",
                    "tokens": [
                        {
                            "xpath": "/ProxyEndpoint/HTTPProxyConnection/BasePath",
                            "value": "/weather"
                        }
                    ]
                }
            ],
            "targets": [
                {
                    "name": "default.xml",
                    "tokens": [
                        {
                            "xpath": "/TargetEndpoint/HTTPTargetConnection/URL",
                            "value": "http://weather.yahooapis.com"
                        }
                    ]
                }
            ]
        },
        {
            "name": "prod",
            ...
        }
    ]
}
```

Top level array is ***apigee.profile*** *from parent POM*

Policies mapped in an array – each is an object

Tokens hold elements to apply configuration

Xpath used for search and replace

Executing Maven Deploy

Using the default apigee.option configuration “validate” creates new revision in Apigee Edge when executing the maven command to build/deploy the API proxy.

```
mvn deploy -Ptest -Dusername=$ae_username -Dpassword=
$ae_password
```

Getting Started – Apigee Maven Plugin

Other useful maven command options:

- Skips tests `-DskipTests=true`
- Overrides current revision `-Doptions=validate,update`
- Deletes currently deployed bundle `-Doptions=clean`
- Imports without deploying `-Doptions=inactive`
- Packages bundle

```
mvn package -Ptest
```

- Runs JMeter Tests

```
mvn jmeter:jmeter -Ptest -Dusername=$ae_username -Dpassword=
$ae_password -Dorg=testmyapi -DtestData=weather_test.csv -
DthreadNum=5 -DrampUpPeriodSecs=5 -DloopCount=2
```

Maven Setup

Demo/Discussion

Maven Setup

API Documentation

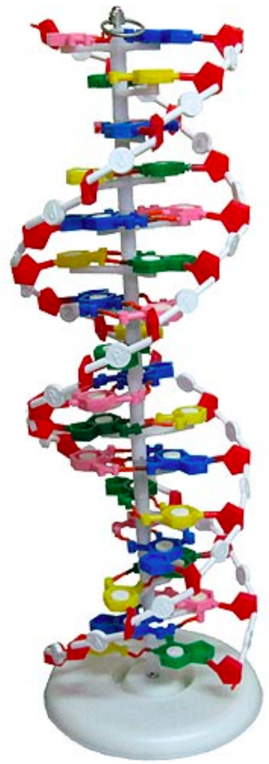
Things to Think About…

- Interactive documentation is becoming the standard for documenting your APIs (e.g. swagger).
- Always treat documentation as code and keep it in version control. Functional changes to code likely change how consumers use the API.
- Deploy documentation when you deploy the API code.

Apigee SmartDocs Overview

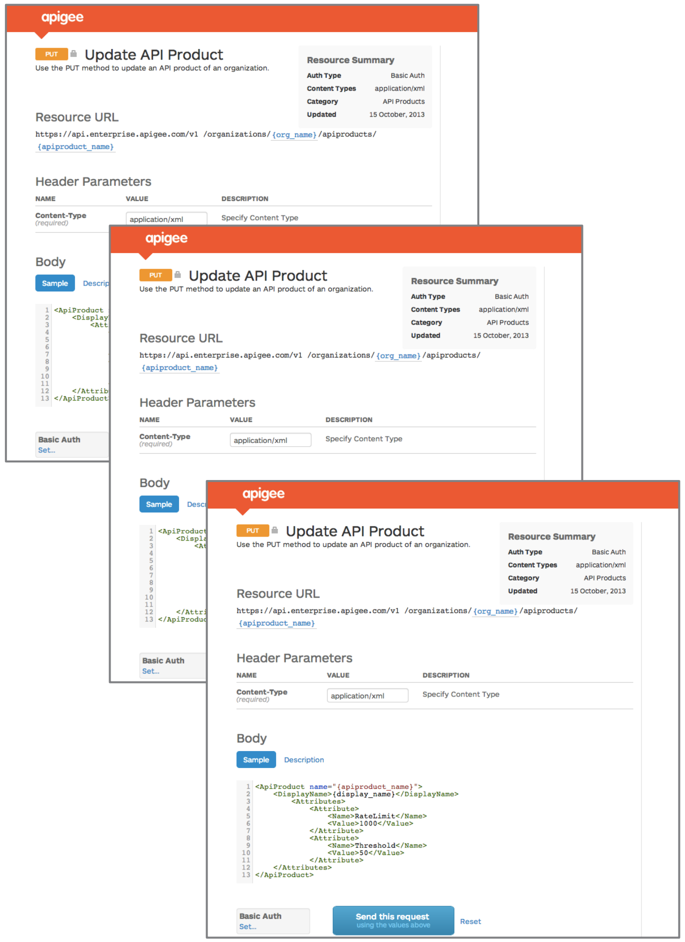
API Modeling

Describe an API structure



SmartDocs

Generate interactive documentation



PUT Update API Product

Use the PUT method to update an API product of an organization.

Resource Summary

Auth Type	Basic Auth
Content Types	application/xml
Category	API Products
Updated	15 October, 2013

Resource URL

```
https://api.enterprise.apigee.com/v1 /organizations/{org_name}/apiproducts/{apiproduct_name}
```

Header Parameters

NAME	VALUE	DESCRIPTION
Content-Type *(required)*	application/xml	Specify Content Type

Body

```
<ApiProduct name="{apiproduct_name}">
    <DisplayName>{display_name}</DisplayName>
        <Attributes>
            <Attribute>
                <Name>RateLimit</Name>
                <Value>1000</Value>
            </Attribute>
            <Attribute>
                <Name>Threshold</Name>
                <Value>50</Value>
            </Attribute>
    </Attributes>
</ApiProduct>
```

API-based

Integrate with any portal / CMS

Apigee Edge Developer Services

Other CMS

gh-pages

Apigee SmartDocs Features

- Method-level documentation
  - Rich detail
  - Internal and external benefits
- Interactive
  - Make requests without leaving the page
- A tool that learns
  - Remembers developers’ preferred values and credentials
- Completely customizable
  - Handlebars-driven templates
  - Complete control over layout, interactions, and look and feel
- Supports Swagger and WADL import

Which Format? WADL or Swagger

Here is some more details to help make the decision to use WADL or Swagger for documenting your APIs.

- WADL is XML-based
- Swagger is JSON and YAML
- Swagger Spec - https://github.com/wordnik/swagger-spec/

Thank you

apigee

Fall 2014