MIRACLE
SOFTWARE SYSTEMS

Jenkins Build and Deploy Process Flow

Request for Proposal | January 05th, 2017

Hanu Veluri
Director - Hybrid Integration
Miracle Software Systems, Inc.

January 04th, 2017

MIRACLE
SOFTWARE SYSTEMS

Table of Contents

Introduction

- Jenkins is a powerful application that allows “continuous integration” and “continuous delivery” of projects
- Jenkins is a Software that allows continuous integration and continuous delivery of projects, regardless of the platform
- We can integrate Jenkins with a number of testing and deployment technologies

Why Jenkins

- Jenkins is a software that allows continuous integration. Jenkins will be installed on a server where the central build will take place

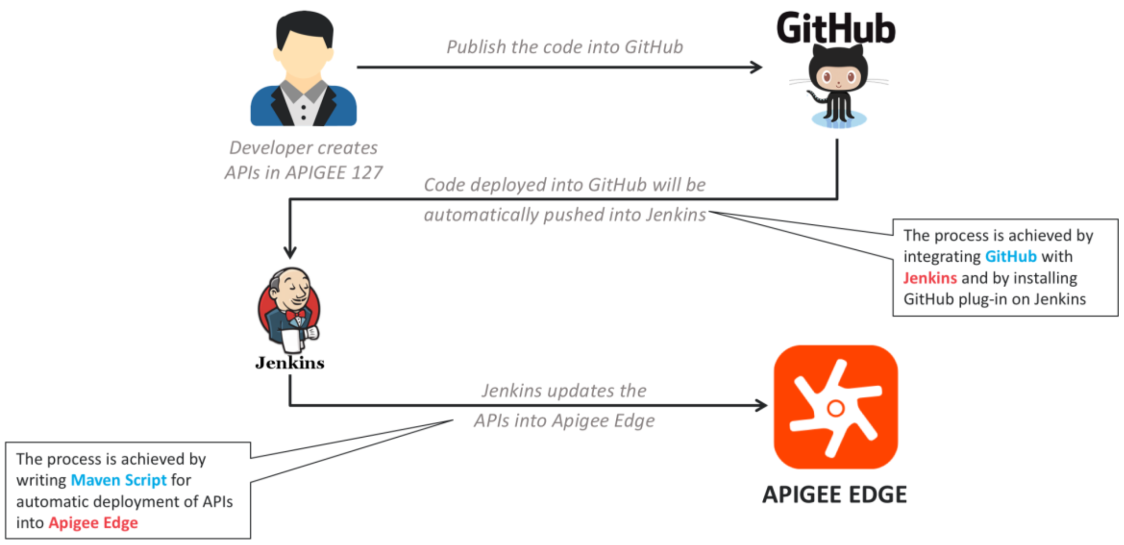
Continuous Integration

- Continuous Integration is a development practice that requires developers to integrate code into a shared repository at several times per day. Continuous Integration improves Software Quality and Reduces the Risk
- Committing code frequently
- Categorizing developer tests
- Using a dedicated integration build machine
- Using continuous feedback mechanisms
- Staging builds

Purpose

- It automates the process of building an API from Git Hub to Apigee Edge using Jenkins

Architectural Diagram


Publish the code into GitHub
GitHub
Developer creates APIs in APIGEE 127
Code deployed into GitHub will be automatically pushed into Jenkins
The process is achieved by integrating GitHub with Jenkins and by installing GitHub plug-in on Jenkins
Jenkins
Jenkins updates the APIs into Apigee Edge
The process is achieved by writing Maven Script for automatic deployment of APIs into Apigee Edge
APIGEE EDGE

Jenkins Architectural Diagram

Requirements

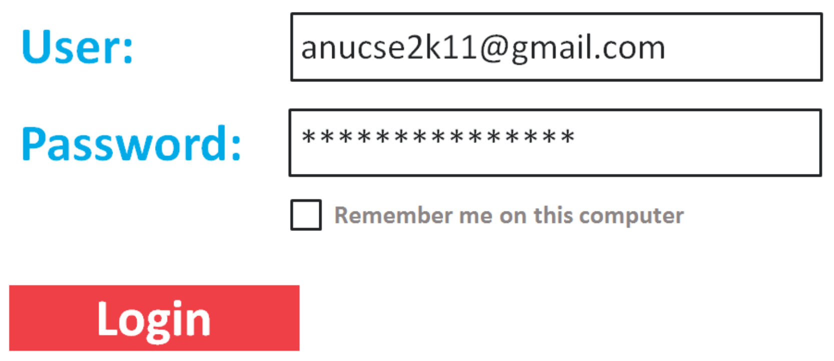
- Git Hub
- Jenkins

Jenkins Installation Steps

- For Installing Jenkins in Ubuntu we have to follow the below steps
- Open your terminal. In that go to root by clinking "sudo -s"
- and then follow the below commands
  - wget -q -O - https://pkg.jenkins.io/debian/jenkins-ci.org.key | sudo apt-key add -sudo sh -c 'echo deb http://pkg.jenkins.io/debian-stable binary/ > /etc/apt/sources.list.d/jenkins.list'
  - sudo apt-get update
  - sudo apt-get install Jenkins
- Now Jenkins is installed in your machine, to update it to a latest version run the following command
  - sudo apt-get update
  - sudo apt-get install Jenkins

Creating user in Jenkins

- So now Jenkins is installed, updated and it is in running state
- Go to browser and give “http://localhost:8080”
- Give your details as shown in the following screen shot.
- And click save and finish
- Now you can see the Jenkins screen as shown below.
- Click “start using Jenkins



User: anucse2k11@gmail.com

Password: ***************

Login to Jenkins

Adding required Plugins in Jenkins

- For Integrating Git Hub with Jenkins using SSH key install all the required plugins in Jenkins like
  – github plugin
  – git plugin
  – Credentials plugin

GitHub Plugin

- This plugins is mainly used for integrates Jenkins with Git Hub project.

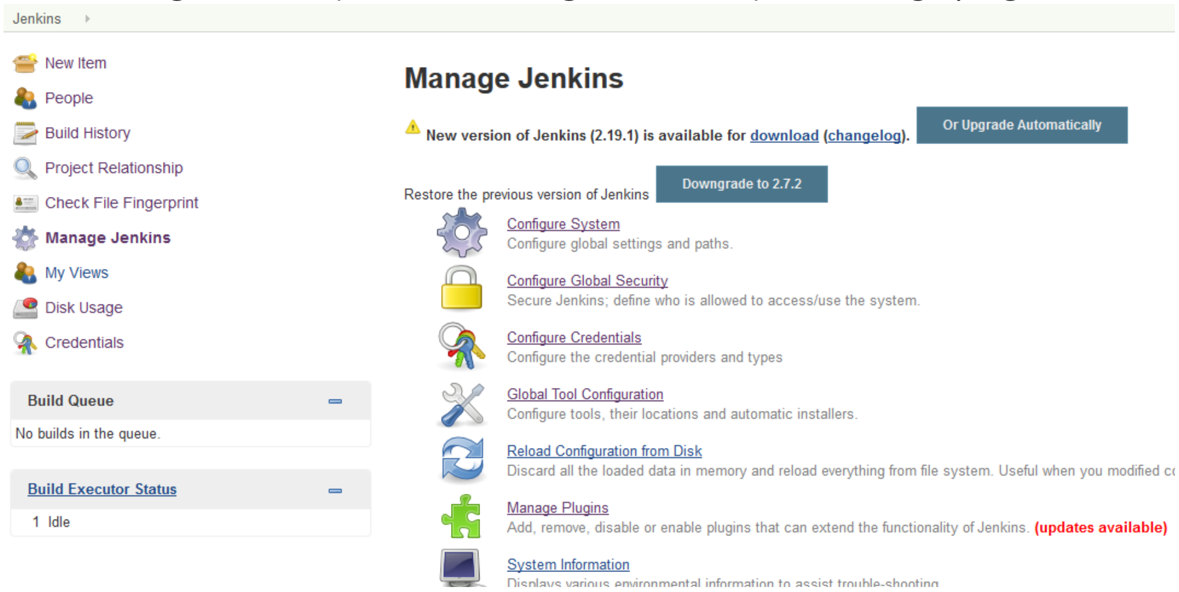
Maven Installation Plugin

- Apache Maven is a software project management and comprehension tool

- Based on the concept of a project object model (POM)
- When a Maven project is created, Maven creates default project structure

Credentials Plugin

- This plugin allows you to store credentials in Jenkins
- The credentials plugin provides a standardized API for other plugins to store and retrieve different types of credentials
- For installing the required plugins go to
- “Manage Jenkins (in the left navigation menu) --> Manage plugins”



Manage Jenkins

New version of Jenkins (2.19.1) is available for download (changelog).

Restore the previous version of Jenkins

No builds in the queue.

1 Idle

Configure System
Configure global settings and paths.

Configure Global Security
Secure Jenkins; define who is allowed to access/use the system.

Configure Credentials
Configure the credential providers and types

Global Tool Configuration
Configure tools, their locations and automatic installers.

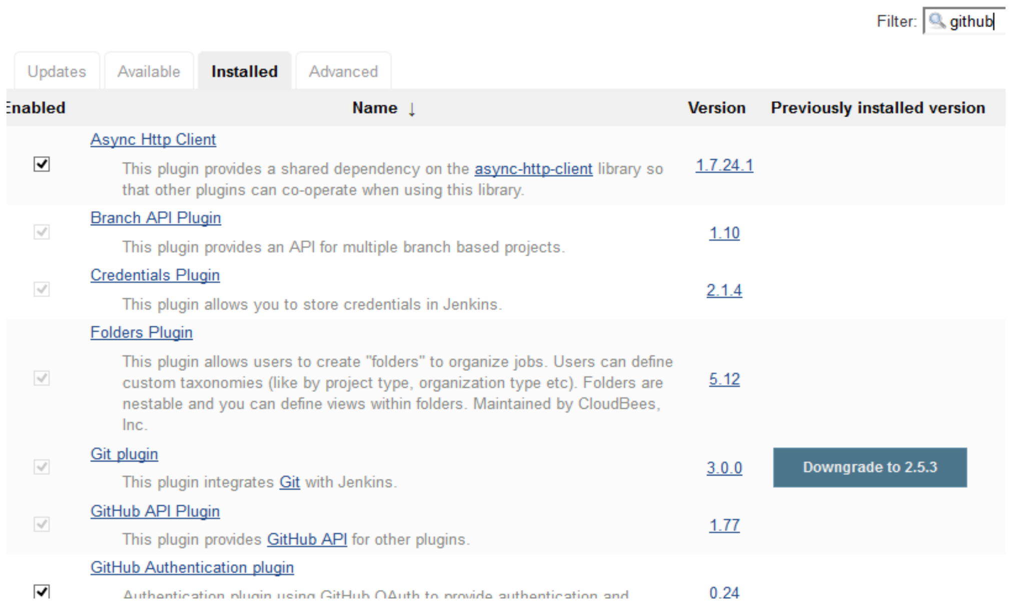
Reload Configuration from Disk
Discard all the loaded data in memory and reload everything from file system. Useful when you modified c

Manage Plugins
Add, remove, disable or enable plugins that can extend the functionality of Jenkins. (updates available)

System Information
Displays various environmental information to assist trouble-shooting

The above image is showing Adding Plugins

- Search for the required plugin to be installed in Available tab. If the plugin is already installed it was in Installed tab

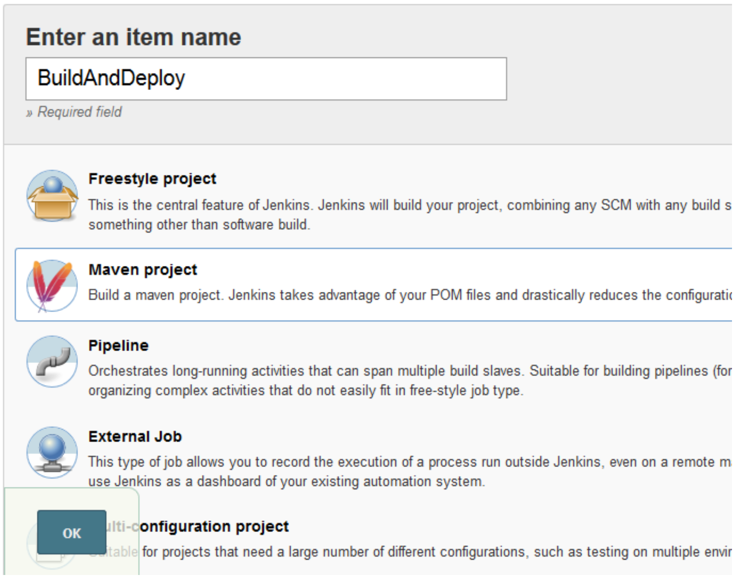

Filter: github

Enabled	Name ↓	Version	Previously installed version
☑	Async Http Client This plugin provides a shared dependency on the async-http-client library so that other plugins can co-operate when using this library.	1.7.24.1	
☑	Branch API Plugin This plugin provides an API for multiple branch based projects.	1.10	
☑	Credentials Plugin This plugin allows you to store credentials in Jenkins.	2.1.4	
☑	Folders Plugin This plugin allows users to create "folders" to organize jobs. Users can define custom taxonomies (like by project type, organization type etc). Folders are nestable and you can define views within folders. Maintained by CloudBees, Inc.	5.12	
☑	Git plugin This plugin integrates Git with Jenkins.	3.0.0	Downgrade to 2.5.3
☑	GitHub API Plugin This plugin provides GitHub API for other plugins.	1.77	
☑	GitHub Authentication plugin Authentication plugin using GitHub OAuth to provide authentication and	0.24	

The above image is showing the Installed Plugins

Creating Job

- Click on new item then select a Maven project


Enter an item name

BuildAndDeploy

» Required field

Freestyle project
This is the central feature of Jenkins. Jenkins will build your project, combining any SCM with any build s something other than software build.

Maven project
Build a maven project. Jenkins takes advantage of your POM files and drastically reduces the configuratio

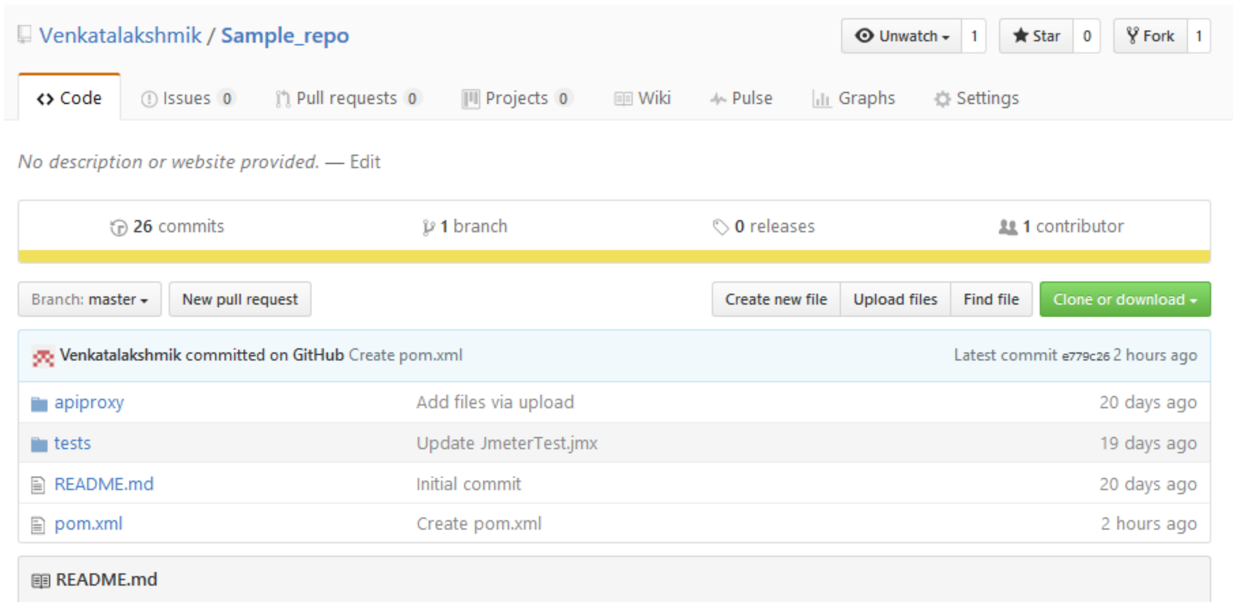
Pipeline
Orchestrates long-running activities that can span multiple build slaves. Suitable for building pipelines (for organizing complex activities that do not easily fit in free-style job type.

External Job
This type of job allows you to record the execution of a process run outside Jenkins, even on a remote m use Jenkins as a dashboard of your existing automation system.

Multi-configuration project
Suitable for projects that need a large number of different configurations, such as testing on multiple envir

The above image is showing the Creation of Maven Project

- Now configure the job with the following details
- In the "Source Code Management " section click on "Git " radio button and give the following github url.https://github.com/papajohns-ds/api-proxy.git



Venkatalakshmik / Sample_repo

No description or website provided. — Edit

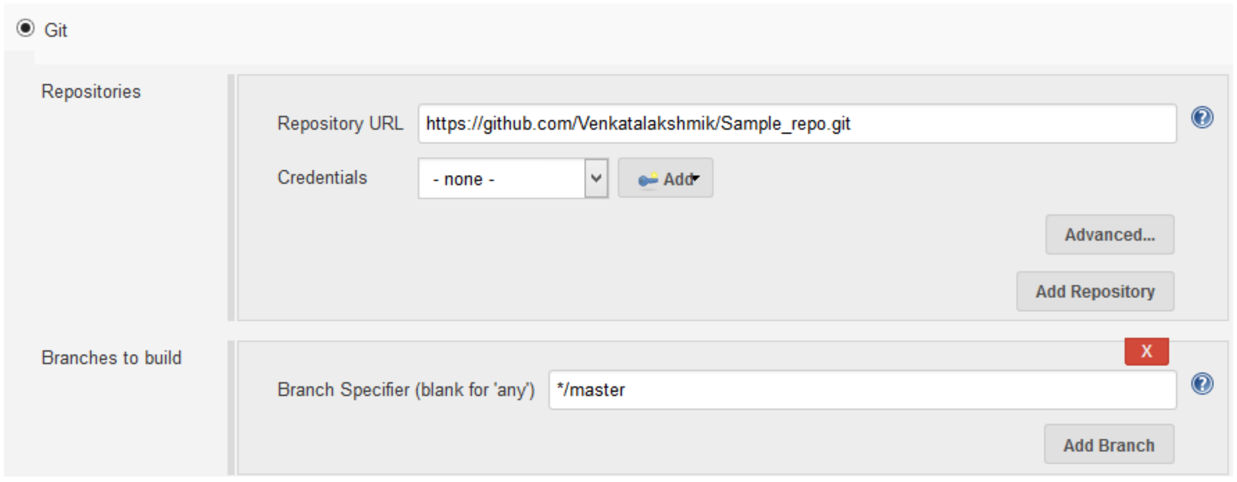
26 commits · 1 branch · 0 releases · 1 contributor

Venkatalakshmik committed on GitHub Create pom.xml — Latest commit e779c26 2 hours ago

apiproxy	Add files via upload	20 days ago
tests	Update JmeterTest.jmx	19 days ago
README.md	Initial commit	20 days ago
pom.xml	Create pom.xml	2 hours ago

README.md

The above image is showing the Git Hub Repository Details



● Git

Repositories

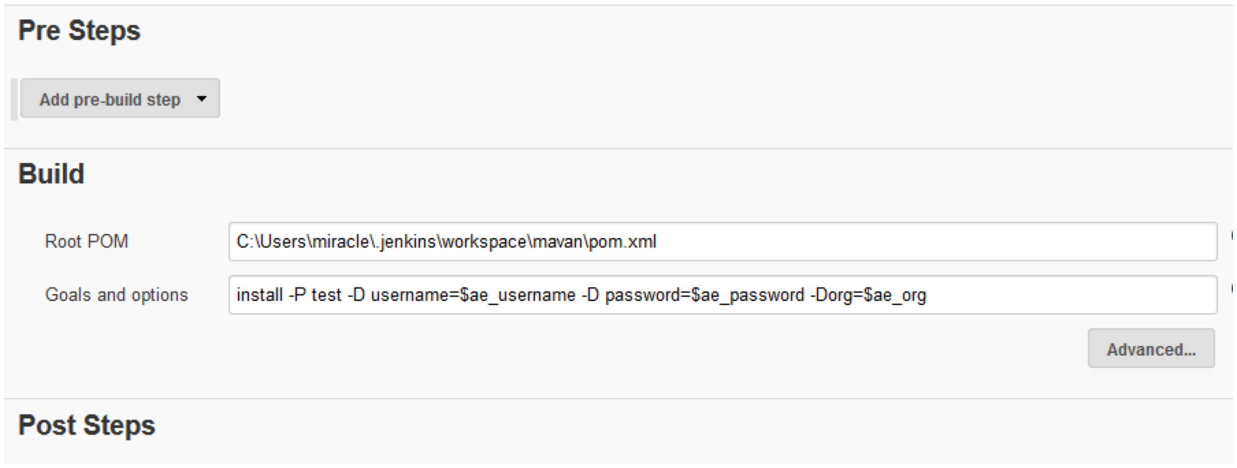
Repository URL: https://github.com/Venkatalakshmik/Sample_repo.git

Credentials: - none -

Branches to build

Branch Specifier (blank for 'any'): */master

The above image is showing the Configuring Jenkins with Git Hub Details



Root POM: C:\Users\miracle\.jenkins\workspace\mavan\pom.xml

Goals and options: install -P test -D username=$ae_username -D password=$ae_password -Dorg=$ae_org

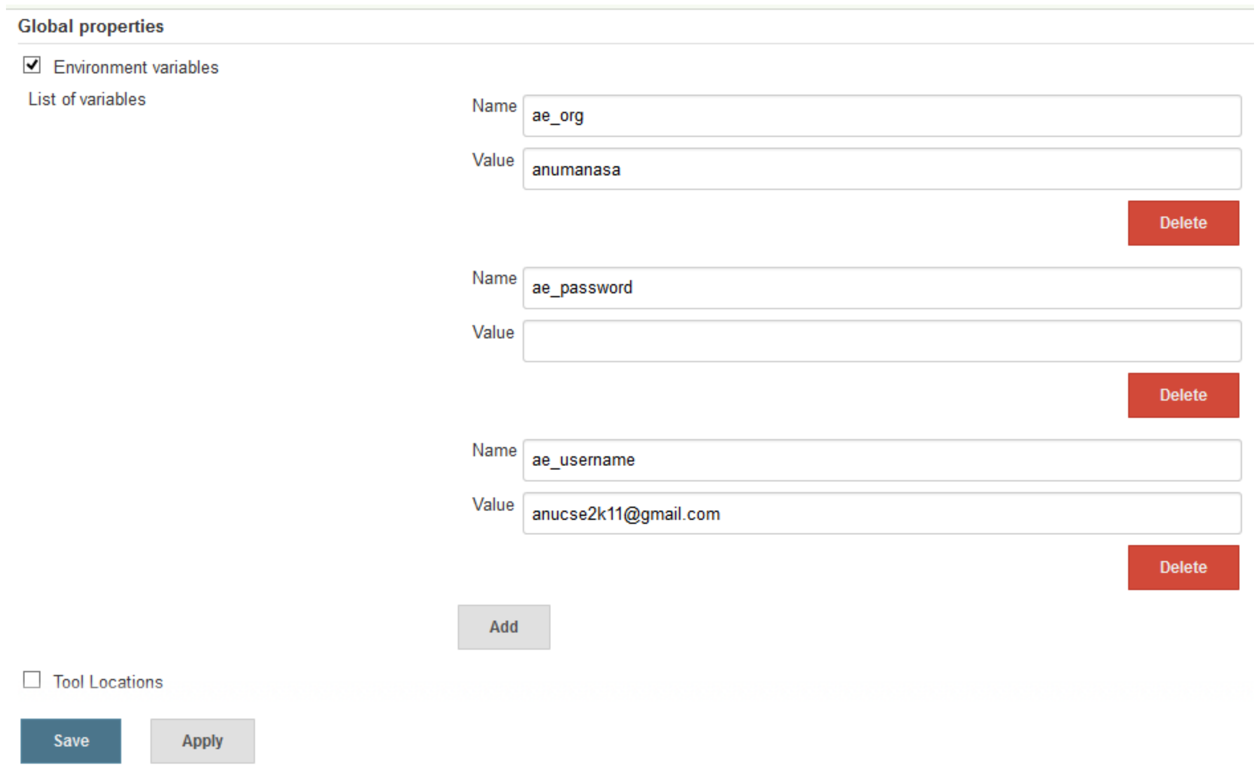
The above image is showing the Configuring pom.xml

- In Build tab we have to mention the path of pom.xml. Here pom.xml is a build file. It defines the goals we have to achieve
- In Goals & Actions we have to give the maven command for integrating with Apigee Edge

The following is the command

install -P test -D username=$ae_username -D password=$ae_password -Dorg=$ae_org

- Next we need to specify the parameters used in Maven command
- For that click on Manage Jenkins and go to the Configure system



Global properties

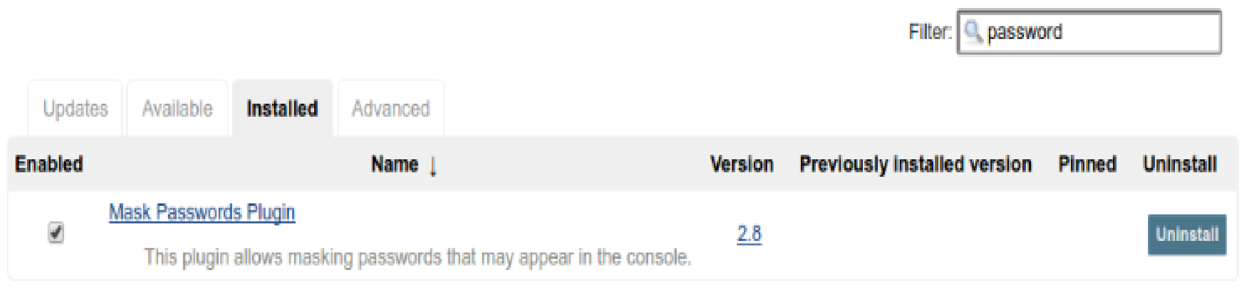
☑ Environment variables

List of variables

Name	ae_org
Value	anumanasa
Name	ae_password
Value	
Name	ae_username
Value	anucse2k11@gmail.com

The above image is showing the Adding Credentials of Apigee Edge User

- Now we have to give the Global properties for particular Apigee Edge user

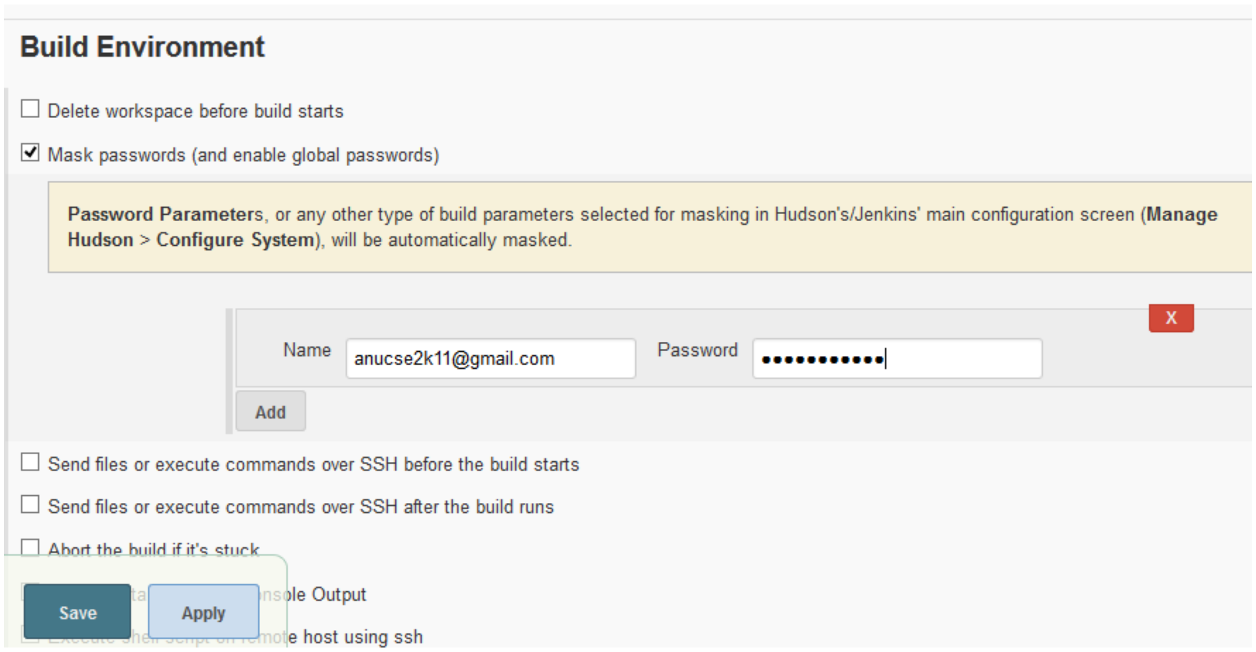


Filter: password

Enabled	Name ↓	Version	Previously installed version	Pinned	Uninstall
✓	Mask Passwords Plugin This plugin allows masking passwords that may appear in the console.	2.8			Uninstall

The above image is showing the Adding Mask Password Plugin

- For masking the password of Apigge Edge account in the console output we have to add the Mask Password Plugin


Build Environment

☐ Delete workspace before build starts

☑ Mask passwords (and enable global passwords)

Password Parameters, or any other type of build parameters selected for masking in Hudson's/Jenkins' main configuration screen (**Manage Hudson** > **Configure System**), will be automatically masked.

Name: anucse2k11@gmail.com Password: ●●●●●●●●●●●●

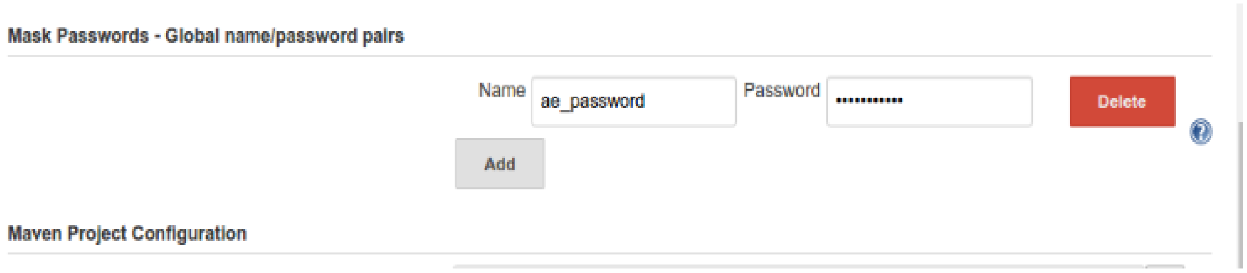
☐ Send files or execute commands over SSH before the build starts

☐ Send files or execute commands over SSH after the build runs

☐ Abort the build if it's stuck

The above image is showing the Configuring Password for masking in Jenkins job

- Select the Mask passwords (and enable global passwords) in the Build Environment for the particular job
- Mention the Name & password


Mask Passwords - Global name/password pairs

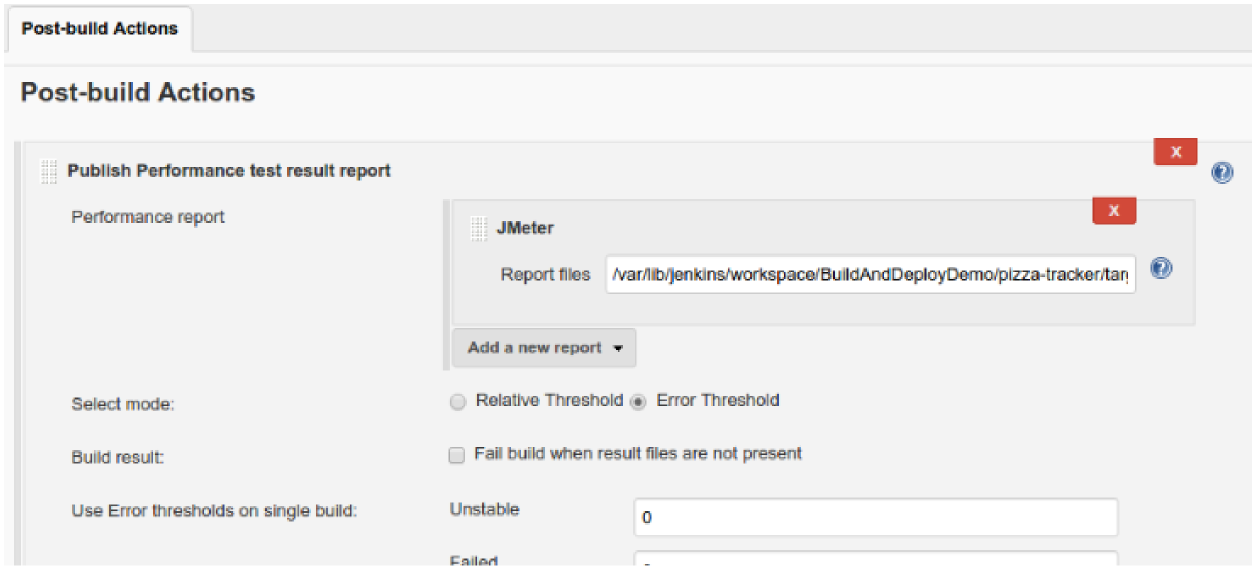
Name: ae_password Password: ••••••••••••

Maven Project Configuration

The above image is showing the Configuring the name and password for masking

- For hiding the password go to the configure system, Click on Add button in Mask Passwords-Global name/password pairs
- Here mention Name and password
- In Post-build Actions add publish performance test result report for showing performance of the Jmeter test cases

- Mention the path for storing the Jmeter test cases result file in the Report files text field



Post-build Actions

Post-build Actions

Publish Performance test result report

Performance report: JMeter — Report files: /var/lib/jenkins/workspace/BuildAndDeployDemo/pizza-tracker/tar

Add a new report

Select mode: Relative Threshold [✓ Error Threshold]

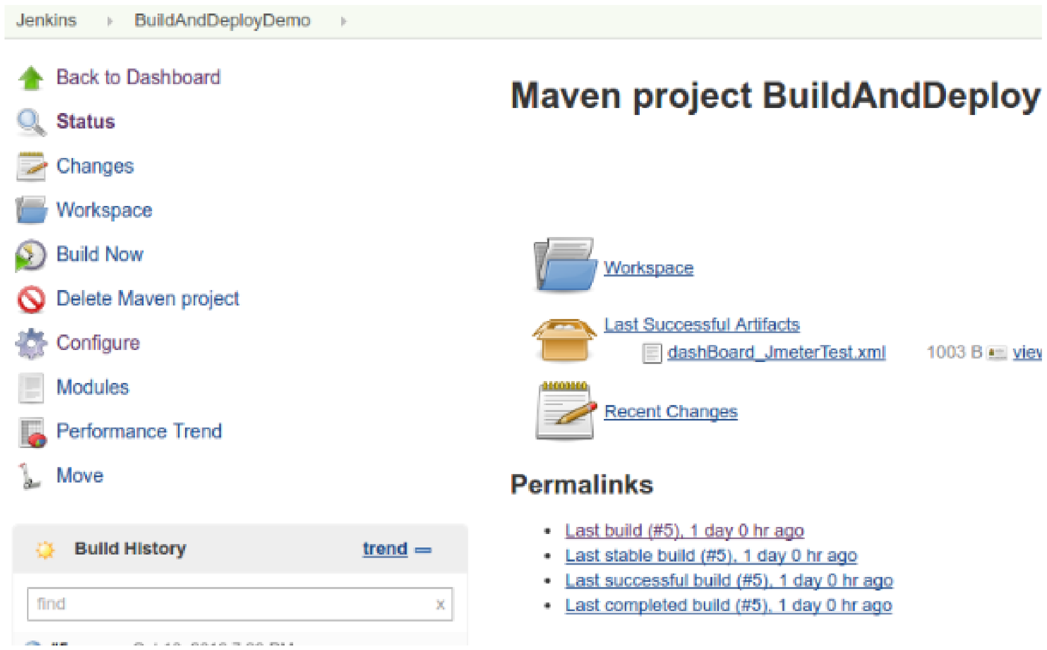
Build result: Fail build when result files are not present

Use Error thresholds on single build: Unstable 0

Failed

The above image is showing the Specify the path for Result Report

- Next Build the project and see the console output



Jenkins ▸ BuildAndDeployDemo ▸

Maven project BuildAndDeploy

- Back to Dashboard
- Status
- Changes
- Workspace
- Build Now
- Delete Maven project
- Configure
- Modules
- Performance Trend
- Move

Build History trend

find

Workspace

Last Successful Artifacts

dashBoard_JmeterTest.xml 1003 B view

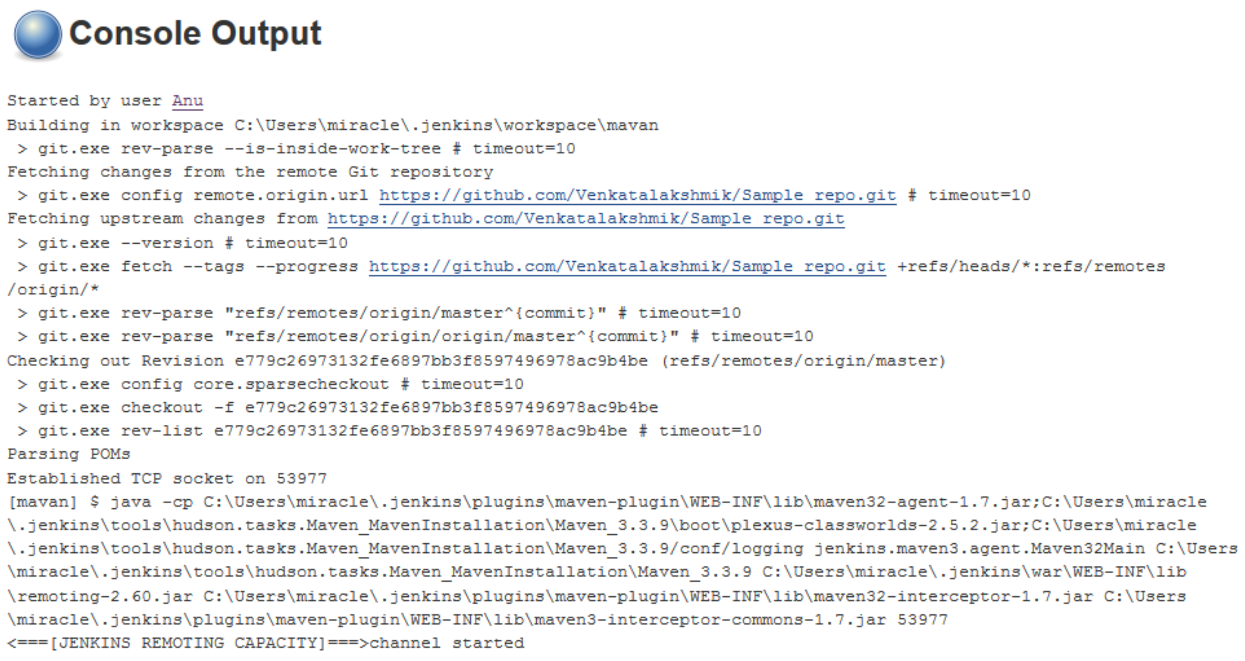
Recent Changes

Permalinks

- Last build (#5), 1 day 0 hr ago
- Last stable build (#5), 1 day 0 hr ago
- Last successful build (#5), 1 day 0 hr ago
- Last completed build (#5), 1 day 0 hr ago

The above image is displays about Building the Job in project

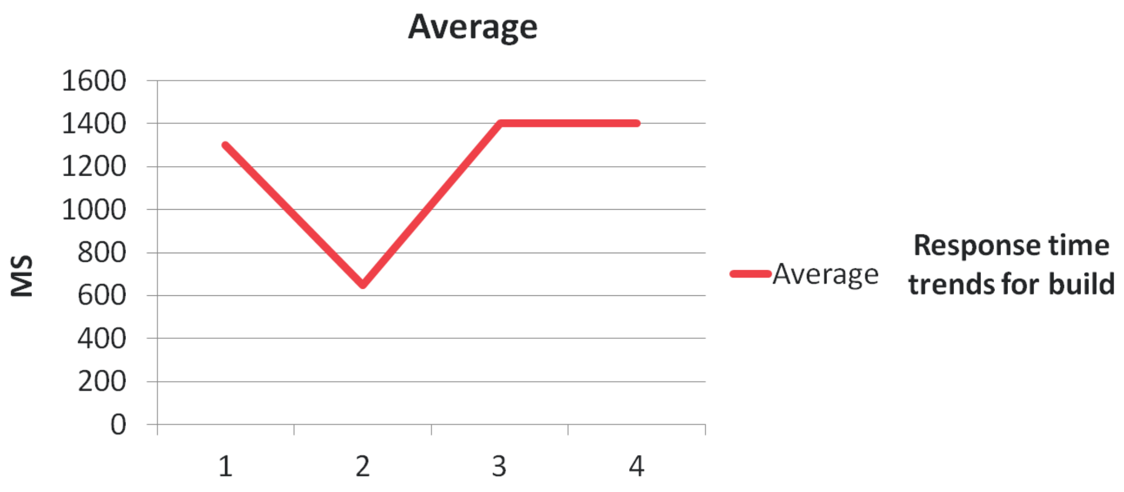
- In the below figure we observe that Jmeter test cases running successfully, and stored into JmeterTest.jtl file

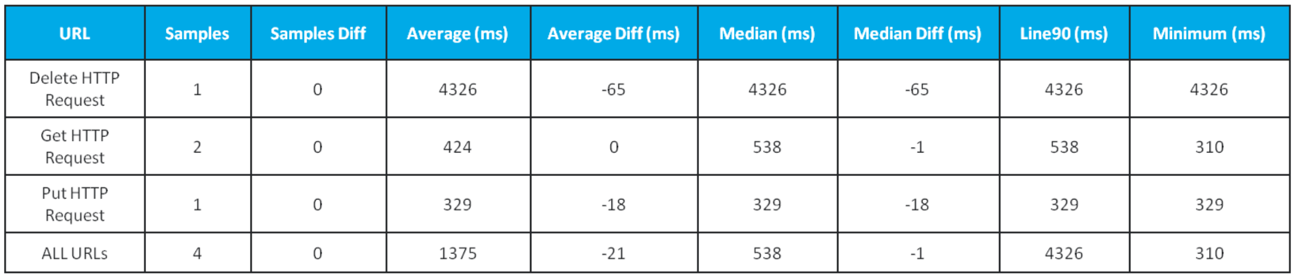


Console Output

```
Started by user Anu
Building in workspace C:\Users\miracle\.jenkins\workspace\mavan
 > git.exe rev-parse --is-inside-work-tree # timeout=10
Fetching changes from the remote Git repository
 > git.exe config remote.origin.url https://github.com/Venkatalakshmik/Sample_repo.git # timeout=10
Fetching upstream changes from https://github.com/Venkatalakshmik/Sample_repo.git
 > git.exe --version # timeout=10
 > git.exe fetch --tags --progress https://github.com/Venkatalakshmik/Sample_repo.git +refs/heads/*:refs/remotes
/origin/*
 > git.exe rev-parse "refs/remotes/origin/master^{commit}" # timeout=10
 > git.exe rev-parse "refs/remotes/origin/origin/master^{commit}" # timeout=10
Checking out Revision e779c26973132fe6897bb3f8597496978ac9b4be (refs/remotes/origin/master)
 > git.exe config core.sparsecheckout # timeout=10
 > git.exe checkout -f e779c26973132fe6897bb3f8597496978ac9b4be
 > git.exe rev-list e779c26973132fe6897bb3f8597496978ac9b4be # timeout=10
Parsing POMs
Established TCP socket on 53977
[mavan] $ java -cp C:\Users\miracle\.jenkins\plugins\maven-plugin\WEB-INF\lib\maven32-agent-1.7.jar;C:\Users\miracle
\.jenkins\tools\hudson.tasks.Maven_MavenInstallation\Maven_3.3.9\boot\plexus-classworlds-2.5.2.jar;C:\Users\miracle
\.jenkins\tools\hudson.tasks.Maven_MavenInstallation\Maven_3.3.9/conf/logging jenkins.maven3.agent.Maven32Main C:\Users
\miracle\.jenkins\tools\hudson.tasks.Maven_MavenInstallation\Maven_3.3.9 C:\Users\miracle\.jenkins\war\WEB-INF\lib
\remoting-2.60.jar C:\Users\miracle\.jenkins\plugins\maven-plugin\WEB-INF\lib\maven32-interceptor-1.7.jar C:\Users
\miracle\.jenkins\plugins\maven-plugin\WEB-INF\lib\maven3-interceptor-commons-1.7.jar 53977
<===[JENKINS REMOTING CAPACITY]===>channel started
```

The below image is showing the Console Output

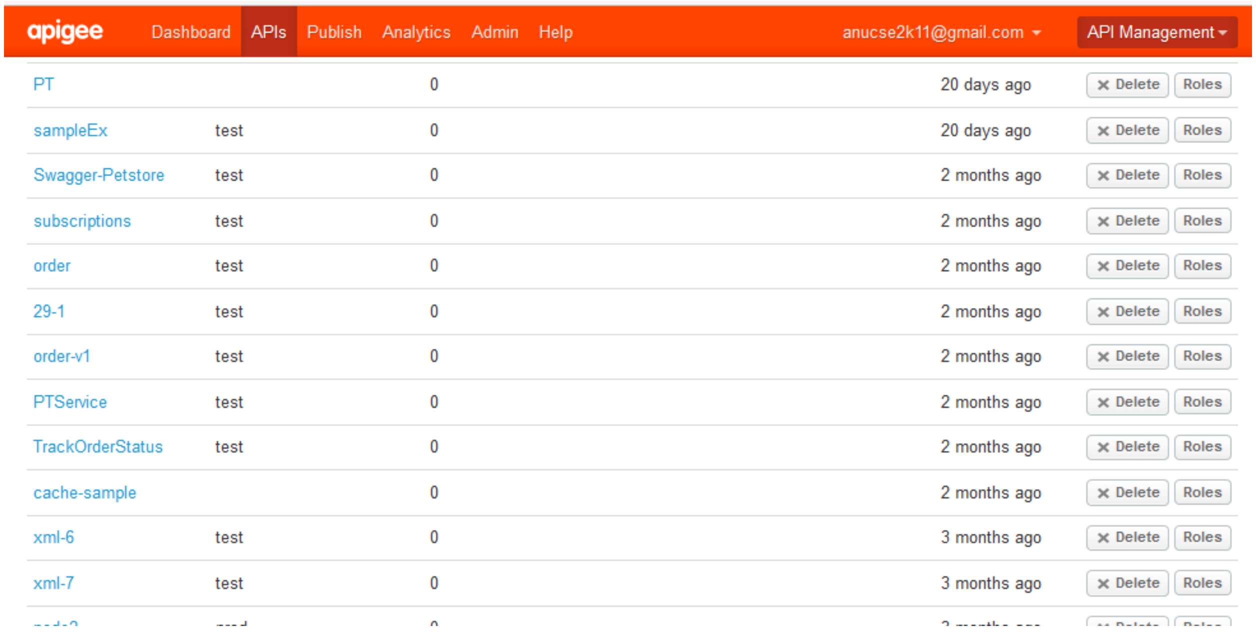




URL	Samples	Samples Diff	Average (ms)	Average Diff (ms)	Median (ms)	Median Diff (ms)	Line90 (ms)	Minimum (ms)
Delete HTTP Request	1	0	4326	-65	4326	-65	4326	4326
Get HTTP Request	2	0	424	0	538	-1	538	310
Put HTTP Request	1	0	329	-18	329	-18	329	329
ALL URLs	4	0	1375	-21	538	-1	4326	310

The above image is showing the Performance Report

- If the project is success then the API proxy will be deployed into Apigee Edge as shown below



PT		0	20 days ago	Delete	Roles
sampleEx	test	0	20 days ago	Delete	Roles
Swagger-Petstore	test	0	2 months ago	Delete	Roles
subscriptions	test	0	2 months ago	Delete	Roles
order	test	0	2 months ago	Delete	Roles
29-1	test	0	2 months ago	Delete	Roles
order-v1	test	0	2 months ago	Delete	Roles
PTService	test	0	2 months ago	Delete	Roles
TrackOrderStatus	test	0	2 months ago	Delete	Roles
cache-sample		0	2 months ago	Delete	Roles
xml-6	test	0	3 months ago	Delete	Roles
xml-7	test	0	3 months ago	Delete	Roles

The above image is showing the Deployed API Proxy in Apigee Edge

Ubuntu Directory Locations

Following are the files and Directories used for the Jenkins build and deploy process.

/var/lib/jenkins/workspace/BuildandDeploy/pizza-tracker
/var/lib/jenkins/workspace/BuildandDeploy/pizza-tracker/apiproxy
/var/lib/jenkins/workspace/BuildandDeploy/pizza-tracker/apiproxy/policies
/var/lib/jenkins/workspace/BuildandDeploy/pizza-tracker/apiproxy/proxies

/var/lib/jenkins/workspace/BuildandDeploy/pizza-tracker/apiproxy/resources
/var/lib/jenkins/workspace/BuildandDeploy/pizza-tracker/apiproxy/target
/var/lib/jenkins/workspace/BuildandDeploy/pizza-tracker/apiproxy/tests/JmeterTest.jmx
/var/lib/jenkins/workspace/BuildandDeploy/pizza-tracker/apiproxy/tests/order_test.csv
/var/lib/jenkins/workspace/BuildandDeploy/pizza-tracker/apiproxy/pom.xml
/var/lib/jenkins/workspace/BuildandDeploy/pizza-tracker/target

Jenkins-Log

```
2016-10-10 19:39:54 Started by user Venkata Lakshmi
2016-10-10 19:39:54 Building in workspace /var/lib/jenkins/workspace/BuildAndDeployDemo
2016-10-10 19:39:54  > git rev-parse --is-inside-work-tree # timeout=10
2016-10-10 19:39:54 Fetching changes from the remote Git repository
2016-10-10 19:39:54  > git config remote.origin.url ssh://git@github.com/papajohns-ds/api-proxy.git #
timeout=10
2016-10-10 19:39:54 Fetching upstream changes from ssh://git@github.com/papajohns-ds/api-proxy.git
2016-10-10 19:39:54  > git --version # timeout=10
2016-10-10 19:39:54 using GIT_SSH to set credentials
2016-10-10 19:39:54  > git fetch --tags --progress ssh://git@github.com/papajohns-ds/api-proxy.git
+refs/heads/*:refs/remotes/origin/*
2016-10-10 19:39:58  > git rev-parse refs/remotes/origin/Macharya-Enhancements-10042016^{commit}
# timeout=10
2016-10-10 19:39:58  > git rev-parse refs/remotes/origin/origin/Macharya-Enhancements-
10042016^{commit} # timeout=10
2016-10-10 19:39:58 Checking out Revision 6b726e41069747ebe031e065d47953c853e70833
(refs/remotes/origin/Macharya-Enhancements-10042016)
2016-10-10 19:39:58  > git config core.sparsecheckout # timeout=10
2016-10-10 19:39:58  > git checkout -f 6b726e41069747ebe031e065d47953c853e70833
2016-10-10 19:39:58  > git rev-list 6b726e41069747ebe031e065d47953c853e70833 # timeout=10
2016-10-10 19:39:58 Parsing POMs
2016-10-10 19:39:58 Established TCP socket on 48498
2016-10-10 19:39:58 [pizza-tracker] $ /usr/lib/jvm/java-7-openjdk-amd64/bin/java -cp
/var/lib/jenkins/plugins/maven-plugin/WEB-INF/lib/maven3-agent-
1.7.jar:/usr/share/maven/boot/plexus-classworlds-2.x.jar org.jvnet.hudson.maven3.agent.Maven3Main
/usr/share/maven /var/cache/jenkins/war/WEB-INF/lib/remoting-2.60.jar
/var/lib/jenkins/plugins/maven-plugin/WEB-INF/lib/maven3-interceptor-1.7.jar
/var/lib/jenkins/plugins/maven-plugin/WEB-INF/lib/maven3-interceptor-commons-1.7.jar 48498
2016-10-10 19:39:59 <===[JENKINS REMOTING CAPACITY]===>channel started
2016-10-10 19:40:00 Executing Maven:  -B -f /var/lib/jenkins/workspace/BuildAndDeployDemo/pizza-
tracker/pom.xml install -P test -D username=anucse2k11@gmail.com -D password=******** -
Dorg=anumanasa
2016-10-10 19:40:01 [INFO] Scanning for projects...
2016-10-10 19:40:01 [INFO]
2016-10-10 19:40:01 [INFO] ------------------------------------------------------------------------
2016-10-10 19:40:01 [INFO] Building PT 1.0
2016-10-10 19:40:01 [INFO] ------------------------------------------------------------------------
2016-10-10 19:40:01 [INFO]
2016-10-10 19:40:01 [INFO] --- maven-resources-plugin:2.6:copy-resources (default) @ PT ---
2016-10-10 19:40:02 [INFO] Using 'UTF-8' encoding to copy filtered resources.
2016-10-10 19:40:02 [INFO] Copying 49 resources
2016-10-10 19:40:02 [INFO]
```

Pom.xml

```
<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>PT</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>

</pluginRepositories>

<properties>

<main.basedir>${project.basedir}</main.basedir>

</properties>

<build>
```

MIRACLE
SOFTWARE SYSTEMS