

IMPORTANT: This GitHub Enterprise instance has been upgraded to v2.12.7. [Release notes](#).

 [AMSPreSales-Demos](#) / [Trainings](#)

DevOps Filling In Octane Analysis (Ex 1)

Edit

New Page

Dave Flynn edited this page 30 minutes ago · 18 revisions

Goal

These exercises are meant to bring your attention to how areas of the Octane Pipeline Analysis screen relates to activities people would be doing through out the process of creating and executing tests along with code changes.

By the end of the exercises, you may not feel like an expert but you should be able to walk away with some confidence on the steps you can do to populate the Octane Pipeline Analysis screen for a custom demo.

It is expected that you have completed the pre-work prior to the start. If you have not, it will be wisest if you cover [DevOps Pre-Work](#) now.

For these exercises, you will need the following containers:

- AutoPass
- DevOps
- IntelliJ
- Octane

Create a Business Rule to Automatically Assign Tests to the

► Pages 16

General

- [Lynda.com](#)
- [Nimbus](#)

AMS Boot Camp 2018

- [Pre-work](#)
 - [Nimbus & Containers](#)
 - [DevOps](#)
 - [PPM & Enterprise Agile](#)
 - [StormRunner Functional](#)
- [Class Work](#)
 - Nimbus and Containerization
 - [PPM & Enterprise Agile](#)
 - Borland and Serena Portfolios
 - [DevOps](#)

Quality Module

STOP - If you didn't do the pre-work, it will be quite obvious as you need to add the following script to your devops container. 😞

```
docker exec devops bash -c "git clone --mirror https://github.com/panama69/oscillating.git /gitrepo/oscillating;chmod -R 777 /gitrepo/oscillating"
```

NOTE If you are using a devops image prior to 1.1.3.x, you will need to use '/GitRepo/oscillating' as the target path.

Steps you will need to perform to accomplish this task

1. Create a job called "Corndog" in Jenkins
2. Restrict your job to run on NimbusServer
3. Using "<http://nimbusserver:8091/git/oscillating>" tests
4. Build using Maven with the following goals "clean test -Dtest=LoginTests,StableTests"
5. Add Post-build to publish Junit Tests using "**/TEST-*"
 - i. Take a look in the jobs workspace target/surefire-reports folder to see why "TEST-*" was used (Hint: take a look in the "target/surefire-reports" folder to figure it out. Formatting questions can be answered by pressing the ? in the "Post-build Actions" of Jenkins")
6. Go to the Quality Module and filter based on "01 map to application model" tag
7. Right on the tests and create a business rule to assign the StableTests to Search and the LoginTests to Security
8. Add a new test to StableTests class and re-run the pipeline. What happens? Where are the tests?

1. Create a job called "Corndog" in Jenkins

- Roadmaps
- ROI Tools
- Demo Narrative discussion
- Hands-on

Previous Boot Camps

- [AMS Bootcamp 2017](#)
 - StormRunner Functional
 - Predictive Overview
 - APM Overview
 - Octane in DevOps Context
 - [Pre-work](#)
 - [Hands-on](#)

Clone this wiki locally


https://github.houston.s...

Clone in Desktop

Enter an item name

Corndog

» Required field

 **Freestyle project**
This is the central feature of

2. Restrict to nimbusserver

☒ Restrict where this project can be run

Label Expression

nimbusserver

[Label nimbusserver](#) is
this job from running

3. Using "<http://nimbusserver:8091/git/oscillating>" tests and "<http://nimbusserver:8091/gitweb/?p=oscillating>" for "gitweb" for Repository Browser

Git

Repositories

Repository URL

http://nimbusserver:8091/git/oscillating

Credentials

- none -

Branches to build

Branch Specifier (blank for 'any')

*/master

Repository browser

gitweb

URL

http://nimbusserver:8091/gitweb/?p=oscillating

4. Build using Maven with the following goals "clean test -Dtest=LoginTests,StableTests" and select the other items as shown below under the "Advanced" button

The screenshot shows a configuration window titled "Invoke top-level Maven targets". It contains several input fields and checkboxes. The "Goals" field is highlighted in yellow and contains the text "clean test -Dtest=LoginTests,StableTests". The "Settings file" field is highlighted in yellow and contains the text "provided settings.xml". Below this, there is a "Provided Settings" section with a dropdown menu showing "leanft-devops-settings". The "Global Settings file" field is highlighted in yellow and contains the text "Use default maven global settings".

Invoke top-level Maven targets	
Maven Version	(Default)
Goals	clean test -Dtest=LoginTests,StableTests
POM	
Properties	
JVM Options	
Inject build variables	<input type="checkbox"/>
Use private Maven repository	<input type="checkbox"/>
Settings file	provided settings.xml
Provided Settings	leanft-devops-settings
Global Settings file	Use default maven global settings

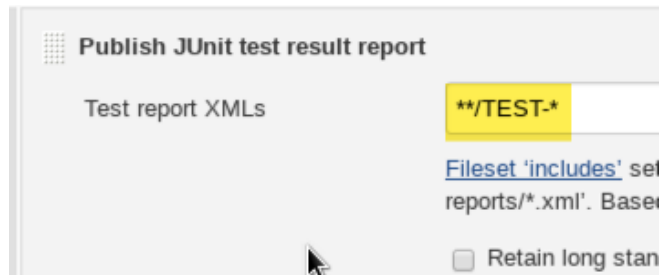
Useful Tidbits

- If the above does not make sense, review your pre-work on [Maven Surefire Plugin - Running a Single Test](#)
- You may try using the above with a newly created LeanFT project and it will fail.
 - Reason is the effective pom uses an older version of the Surefire Plugin and you need

2.19+.

- You will notice in the pom file of the oscillating project, the maven-surefire-plugin is set to use version 2.20.1.
- How can you know what the version in effect is? Run [mvn effective-pom](#) from your project to see the version of the maven-surefire-plugin.
- You may also be wondering why all the test classes (LoginTests, PayPalTest, StableTests, UnstableTest) have a form of "Test" in the class name. Why didn't you just call it **Corndog** or **Prontopuff**? 😊
 - By default, [Maven Inclusions and Exclusions of Tests](#) uses some form of "Test", which is case sensitive, to identify the classes it is to run as a test. You can modify this if you wish by following the instructions with link provided in this line. This is also why the LeanFT test has a class name of "LeanFtTest" to start.

4. Add Post-build to publish Junit Tests using "***/TEST-*



Create and run Run your pipeline in Octane

Add Pipeline

Select a job for the root of the pipeline and view a graphic representation of the pipeline flow.
View job status, duration, time of last run, as well as connections between jobs that run in sequence or run each other.

CI Server

Name:

test

Job:

Corndog

Pipeline name:

Corndog

Release:

AOS (Default)

Type:

Test

☐ Notify committers upon run failure

☐ Notify test owners if their tests fail

Save

Cancel

✓	Corndog		
Run	#1		
Duration	28 s		
Last Run	02/28/2018 11:20		
5	0	1	6

Why is there a skipped test?

- Hint: Take a look at the StableTests class in IntelliJ

Why was *TEST-** used when creating the job in Jenkins?

- Take a look in the jobs workspace target/surefire-reports folder to see why *TEST-** was used (Hint: take a look in the "target/surefire-reports" folder to figure it out.
- Formatting questions can be answered by pressing the '?' in the "Post-build Actions" of Jenkins)

5. Go to the Quality Module and filter Tests based on “Corndog” tag

<

Application Modules

Overview

Tests

Features

Defects

+ Manual Test

▼

↺

✕

🔍

📄

🔗

✉

▶

📅

🔗

🔗

📄

Corndog

+ Add Filter

AT

2001 alwaysPasses - StableTests (net.demo)

☐

AT

2002 forgotUsernameLogin - LoginTests (net.demo.login)

AT

2003 simpleUsernameLogin - LoginTests (net.demo.login)

AT

2004 skippedTest - StableTests (net.demo)

AT

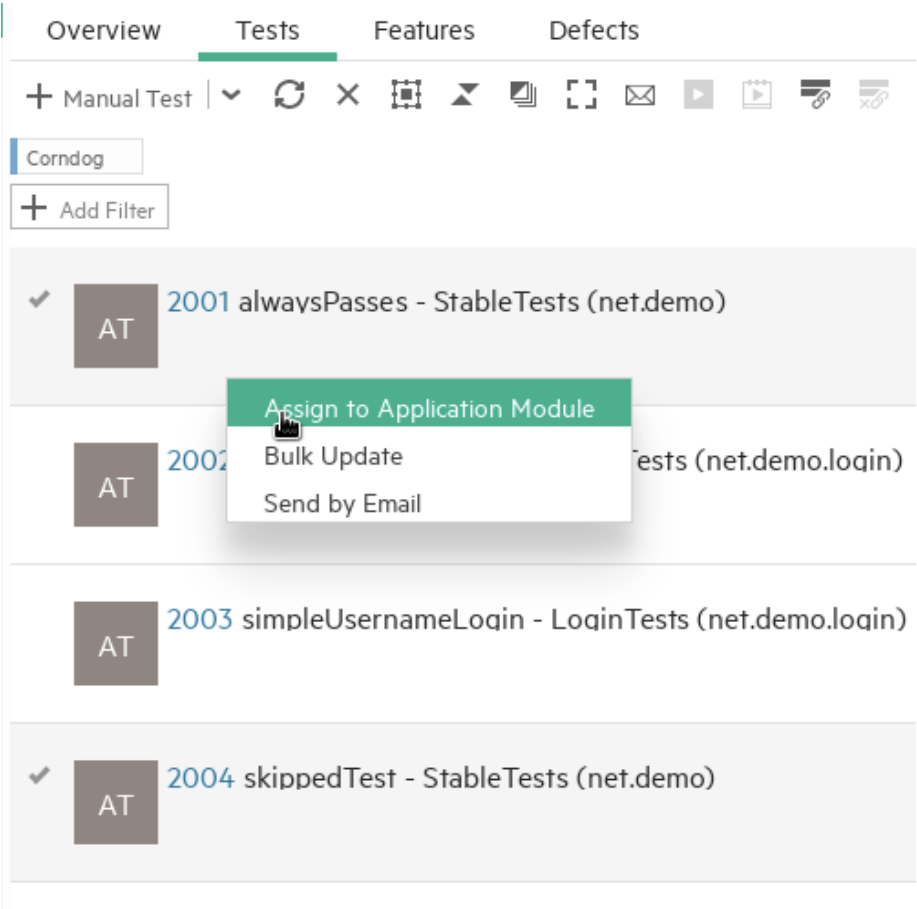
2005 simpleEmailLogin - LoginTests (net.demo.login)

AT

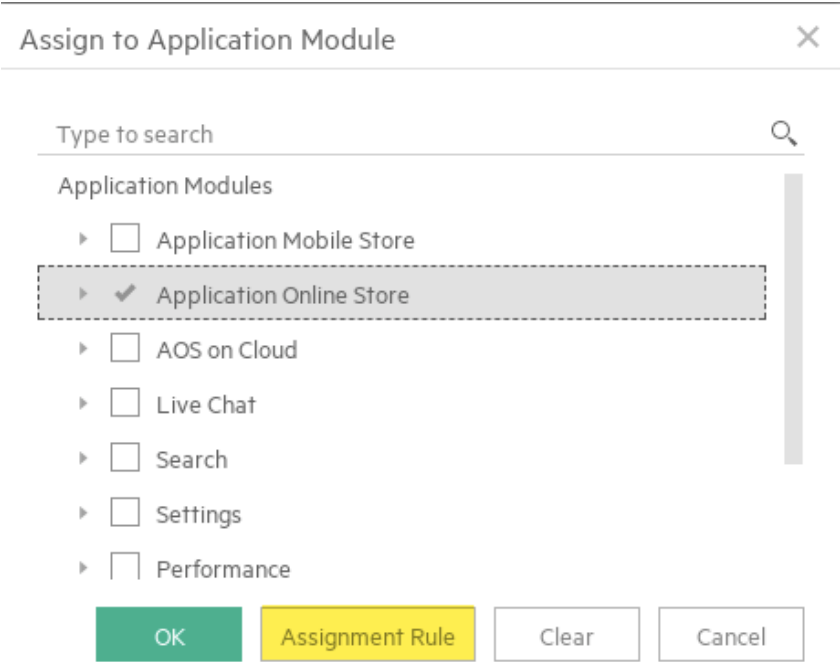
2006 forgotPasswordLogin - LoginTests (net.demo.login)

6. Right on the tests and create a business rule to assign the StableTests to Search

1. Select the "StableTests", then right click and "Assign to Application Module"



3. Select the "Application Online Store" module and then select **Assignment Rule**



4. Give it a rule name. (notice the screen tells you how many tests your rule will affect based on your filter)

Repeat for LoginTests and assign them to "Application Mobile Store"

7. Add a new test to StableTests class, commit, push and re-run the pipeline.

1. Open the the oscillating test in IntelliJ and add a new test.

```
💡@Test
public void myNewTest() {
}
}
```

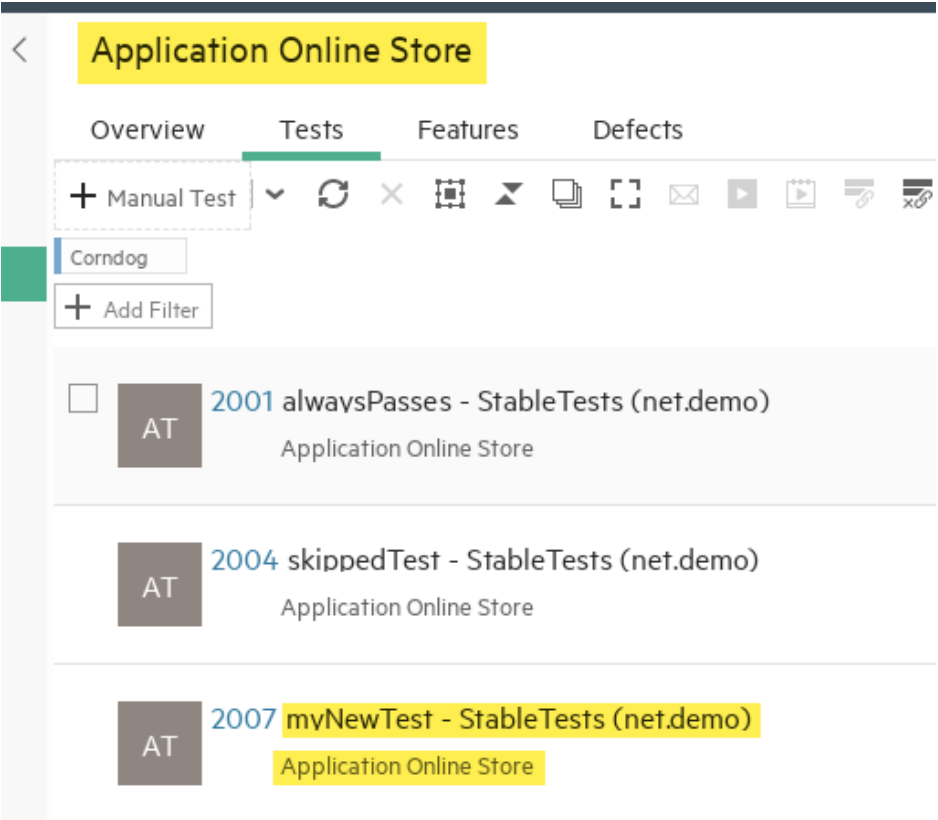
You can copy/paste following if you wish:

```
@Test
public void myNewTest() {
}
```

2. Commit and push the changes. Then run the pipeline again.

Where did the new test appear in Octane?

Why does it appear here?



Continue on to [Exercise 2](#) 🖱️

+ Add a custom footer