Integration with Concourse

Concourse is a CI/CD tool available on-premises, highly focused on pipelines.

Xray does not provide a specific plugin for Concourse. However, similarly to CircleCI, Gitlab and other tools, it is easy to configure a pipeline and integrate it with Xray.

For this, you may simply take advantage of Xray's REST API for submitting results for example.

- JUnit example
- References

JUnit example

In this scenario, we want to get visibility of the automated test results from some tests implemented in Java, using the JUnit framework.

This recipe could also be applied for other frameworks such as NUnit, TestNG or Robot.

The tests are implemented in a JUnit class (stored in a git repository) as follows.

CalcTest.java

```
package com.xpand.java;

import org.junit.After;
import org.junit.Before;
import org.junit.Test;

import static org.hamcrest.CoreMatchers.is;
import static org.junit.Assert.assertThat;

public class CalcTest {

    @Before
    public void setUp() throws Exception {

    }

    @After
    public void tearDown() throws Exception {

    }

        @Test
    public void CanAddNumbers()
    {
        assertThat(Calculator.Add(1, 1), is(2));
        assertThat(Calculator.Add(-1, 1), is(0));
    }


    @Test
    public void CanSubtract()
    {
        assertThat(Calculator.Subtract(1, 1), is(0));
        assertThat(Calculator.Subtract(-1, -1), is(0));
        assertThat(Calculator.Subtract(100, 5), is(95));
    }


    @Test
    public void CanMultiply()
    {
        assertThat(Calculator.Multiply(1, 1), is(1));
        assertThat(Calculator.Multiply(-1, -1), is(1));
        assertThat(Calculator.Multiply(100, 5), is(500));
    }


    public void CanDivide()
    {
        assertThat(Calculator.Divide(1, 1), is(1));
        assertThat(Calculator.Divide(-1, -1), is(1));
        assertThat(Calculator.Divide(100, 5), is(20));
    }


    @Test
    public void CanDoStuff()
    {
        assertThat(true, is(true));
    }


}
```

What concerns Concourse itself, we start by setting up a pipeline based on a YAML configuration file.

pipeline.yml

```
---
resources:
- name: repo
  type: git
  source:
    uri: https://example.com/java-junit-calc.git
    username: john
    password: xxxxxxx

jobs:
- name: tests
  plan:
  - get: repo
    trigger: true
  - task: mvn-test
    config:
      platform: linux
      image_resource:
        type: docker-image
        source:
          repository: maven
      inputs:
      - name: repo
      run:
        path: bash
        args:
        - -c
        - |
          set -e
          cd repo/java-junit-calc
          mvn test
          curl -H "Content-Type: multipart/form-data" -u admin:admin -F "file=@target/surefire-reports/TEST-com.
xpand.java.CalcTest.xml" "http://jiraserver.example.com/rest/raven/1.0/import/execution/junit?projectKey=CALC"
```

Note that this file is just an example; you should avoid hardcoded usernames, passwords in the .yml file.

In order to submit test results, we'll just need to invoke the REST API (as detailed in Import Execution Results - REST).

We're using "curl" utility that comes in Unix based OS'es but you can easily use another tool to make the HTTP request; however, "curl" is provided in most Docker images.

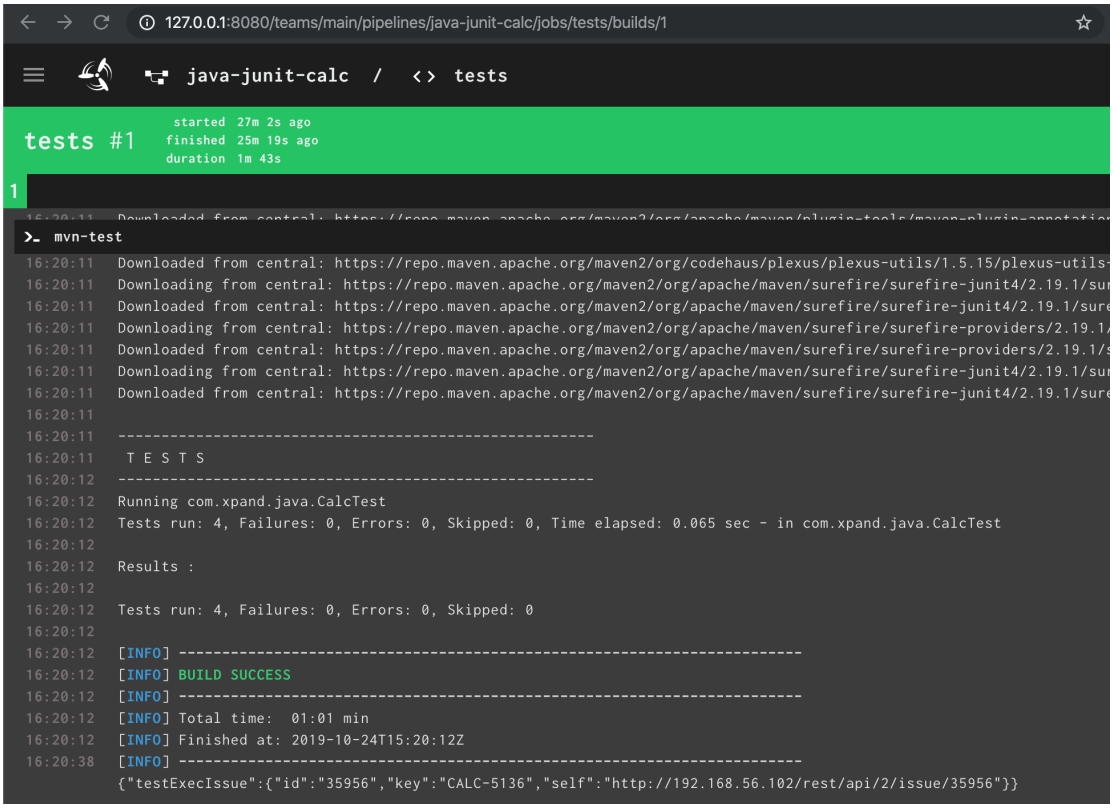
Please note

The Jira user (i.e. username and password) mentioned in the configuration below must exist in the Jira instance and have permission to Create Test and Test Execution Issues.

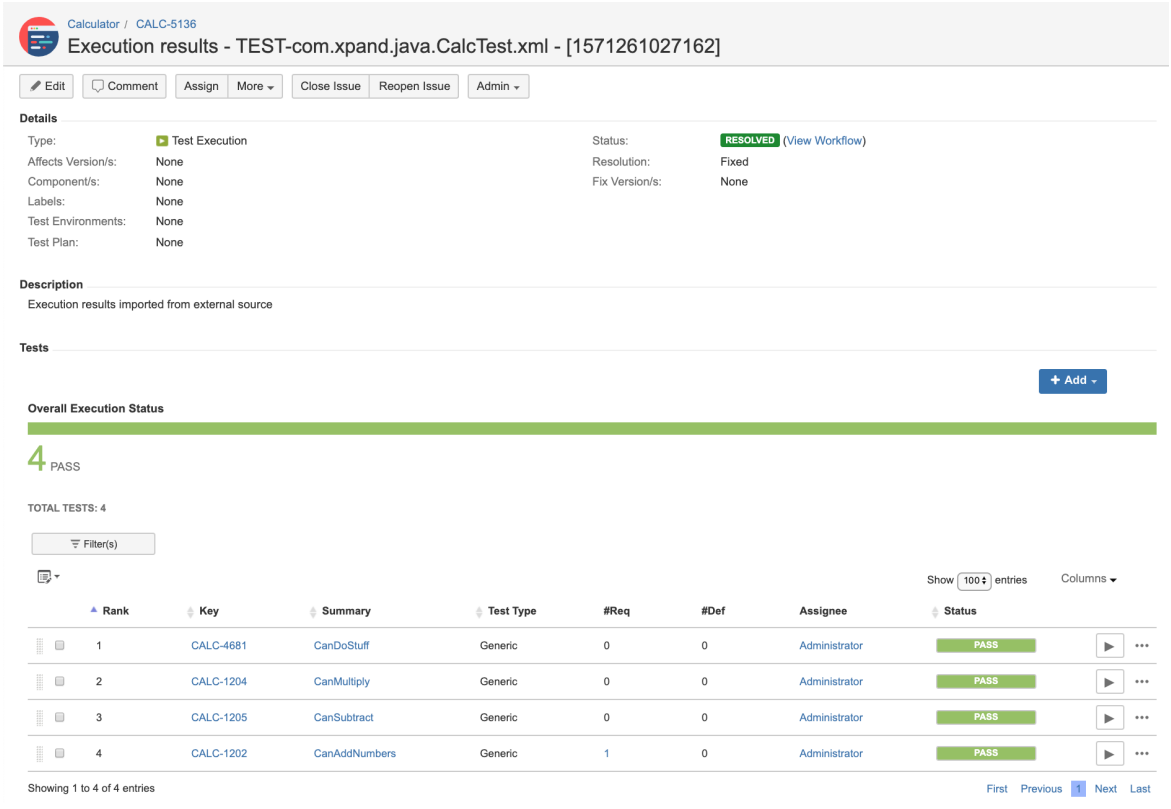
After preparing the pipeline configuration file, it needs to be submitted to Concourse.

```
fly -t tutorial set-pipeline -p java-junit-calc -c pipeline.yml
```

You can open your browser (e.g. *http://127.0.0.1:8080/teams/main/pipelines/java-junit-calc/jobs/tests/*) and execute/check the execution progress there.



And in Xray, in this case, we'll have a new Test Execution containing the results of the automated tests.



References

- https://concourse-ci.org/
- https://concoursetutorial.com/basics/basic-pipeline/