GeoSolutions CITE automation proposal

GeoSolutions SAS
04/12/2019
Version 01.00
Proposal #111/2019

Table of Contents

Introduction

The GeoServer community is currently lacking an automated way to run CITE conformance tests against nightly builds, on a day to day basis. GSIP 176 is standing a Request For Proposal open to third parties for a technical solution to the problem, with the following requirements:

- Run CITE tests against nightly builds in an automated fashion, and report results (no requirement to pass the tests). The tests must be run on build.geoserver.org
- Support the test suites that the old automation covered, that is, WCS 1.0 and 1.1, WMS 1.1 and 1.3, WFS 1.0 and 1.1
- Recommends, but does not require, supporting WCS 2.0, WFS 2.0, WMTS 1.0, while other applicable test suites are considered optional
- The delivery should include all scripts, source code and documentation in a way that matches the GeoServer contribution guidelines
- There is no desire to set up a fork of the Team Engine or the tests themselves.
- The tests must be runnable on a Ubuntu 18 image (as bash scripts).

Proposed solution

Overview

We propose to use a Docker based solution using a compose based on three images:

- An optional stock PostgreSQL/PostGIS image, that the docker compose will use for WFS tests, loading the test data on startup
- An image running GeoServer, which will include the test data directory, and will download the latest nightly during the startup, and set it up to run in Tomcat with a data directory indicated by the compose
- An image running the tests, including the teamengine and all relevant test suites, which will be provided with an indication of what test to run, and a form file with answers to the test setup questions (e.g., location of the capabilities, which profiles to perform).

The overall setup will be contained in its own GitHub repository, placed inside the GeoServer organization, with the following expected structure[1]:

- `cite-console` (Maven project with docker file and dependencies to build the test runner)
- `<service-version>` (module with all necessary bits to run a specific test)
    - docker-compose.yml (recipe to run the specific test)
    - data-dir (data directory for the specific test, moved from the current)
    - form.xml (Teamengine's `test.sh` setup parameters)
    - setup.sql (eventual setup file for the database)
- `<service-version>` (repeat for all other services)
- `run.sh` (script to startup the right docker compose and run a particular test)

Test running image

OpenGeoSpatial already provides a project building docker image running the TeamEngine as a web application, with a rich set of test suites. The setup is clean, in particular:

- The main pom file depends on the various test suite packages, each version of which is a property.
- The Dockerfile composes them in a suitable and ready to use installation, including the test engine and the necessary bits to run it.

There are two main downsides to this approach:

- Several of the test suite packages are not actually deployed on Maven repositories (only a subset of the most recent ones are)

[1] The delivery structure might be different, or split the same folders in more than one repository.

- For the sake of running the test on the build server the web application setup is not suitable, as the REST API automation only supports a small subset of the possible test suites, leaving out several of those that are in the requirements.

Our plan is to copy the approach, creating a new module in the GeoServer CITE repository, with a few variants:

- Only depend on the test suites we'll actually use
- Setup both a Tomcat and a command line runner for the tests
- Provide a startup script for the image that will, depending on the parameters:
  - Run a web application setup, useful for local development, which will provide ease of debugging for those fixing the compliance issues
  - Run the specific test suite (by name and version) and form file, with an exit status that can be used to determine if the test passed.

The image in question will be built by a dedicated build on build.geoserver.org, and deployed on a GeoServer DockerHub repository, that we'll stand up as part of the automation activities.

In parallel, we'll create a build job on build.geoserver.org for each test suite, allowing to build and deploy the test suite zip package for a given tag or branch. Each build will work off the test suite public repositories, (e.g.. https://github.com/opengeospatial/ets-wms11) and will accept the tag to be built and deployed.

With this setup we'll be able to automate the full suite execution on a daily basis, while providing a suitable companion for local development and testing, when working on GeoServer own compliance towards passing a given test suite.

The one downside is probably that the various test suite versions numbers will have to be updated manually in the cite-console project pom file, however, that also ensures we're going to have a reliable test suite, faced with a small manual overhead of upgrading the version numbers as OGC makes new test suites available.

Deliverables

The activity will provide the following deliverables:

- A new repository to host the GeoServer CITE test machinery, including the necessary files to build the docker images described above
- Documentation on how to use the builds, both on the server and on the developer machine, as part of the GeoServer developer guide

- One new build for each protocol and version, on build.geoserver.org, to build the test suite packages, should OGC not make them available. The builds will be run manually as we decide to update the test suite.
- One new build for each protocol and version, to run the test suite via the associated docker-compose file found in the above repository.

Compensation and deadlines

The price proposed for this is 10.000,00€ with invoicing as follows:

- 30% at contract signature
- 70% on delivery

Payments terms are 30 days net.

Deliveries are planned as follows:

- By December 20th 2019 we'll deliver the jobs to build each test suite zip file and deploy it in a maven repository
- The rest of the work will be delivered by the end of February 2020