G4SE Technical Documentation

Release 1.0.x

Nicola Jordan

Nov 29, 2016

Contents

1 Overview **1**
- 1.1 Architecture . . . 1
- 1.2 Deployment . . . 1
- 1.3 Scaling Options . . . 1
- 1.4 Data Input . . . 3

1 Overview

1.1 Architecture

TODO: Add image for architecture (component diagram, with django, elasticsearch, postgres, redis, api etc and its dependencies)

TODO: Add image for data processing and return while searching (U-Diagram)

1.2 Deployment

The Service runs on the switch cloud infrastructure (SwitchEngines). It is a virtual server with:

- 2 cores
- 4 GB RAM
- 20 GB diskspace

It is powered by *Debian Jessie (Linux)*.

All services run on the same hardware, as shown below.

1.3 Scaling Options

There are multiple scaling options, some of which require more work than others.

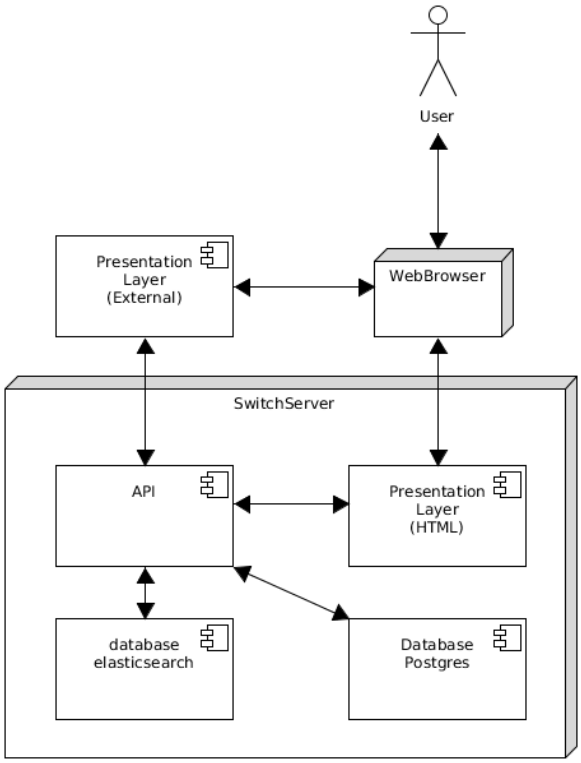

Fig. 1: Deployment Diagram as currently in use (Switch Cloud Infrastructure)

More CPUs, RAM

Assuming running with uwsgi, we need to harness the power, and because we have elasticsearch and postgres running on the same machine, we have to take not to degrade their performance.

A good rule of thumb, which has proven quite useful, is with N-Cores, where N is larger than 2:

N+1 processes, N/2 (if N is odd, add 0.5) threads.

Examples:

- N=2: uwsgi /uwsgi.ini –processes 3 –threads 1 (see code-snippet below)
- N=3: uwsgi /uwsgi.ini –processes 4 –threads 2
- N=4: uwsgi /uwsgi.ini –processes 5 –threads 2

Adding additional Servers

Putting every service on it's own server has the big advantage that scaling is possible much more easily.

This can be achieved using docker-cloud or a similar service, the configuration for this scenario is so divers, that it cannot be included in this documentation.

If much more power is required, the elasticsearch service can be run on a separate, dedicated machine or even be distributed on multiple machines. FOr Postgres the same can be done, using a master-slave configuration where for example writes go only to master, and reads only to slave.

The application/api should of course also be run separately for maximum benefit.

Switching to a more powerful server

This is the same as more CPU, RAM, just that I use have a real world example.

Using a Hetzner Server, specifically the https://www.hetzner.de/de/hosting/produkte_rootserver/ex51ssd with 2X500GB Harddisk, 64GB RAM and 4 Cores/8 Threads without much tweaking a load up to 600 to 700 Request per second was possible. This is more than 20 fold of what is possible with the server above - this method might be the most cost effective way.

1.4 Data Input

There are two ways for data to be entered into the system.

Zipped-XML-Import

Using the Admin-Interface of Django, a zip-file containing XML-Files can be imported. This deletes all the previously imported data (not the ones entered manually, though).

Manually added and maintained Metadata

Using the Admin-Interface one can add, delete and change the Metadata that has not been imported automatically.