GemsTracker As A Service Hosting

Introduction

At the moment the GemsTracker Development Team is searching for a suitable hosting company where we can host GemsTracker As A Service (GAAS).

This document is an addition to the GemsTracker Hosting document (V3) from September 2013 that contains a high level description of GemsTracker itself and all the technical requirements concerning PHP versions, libraries and initialization, library installation and other technical hosting details.

This document is concerned only with what is needed to be able to host GAAS for our clients, using our experience during two years of hosting GemsTracker on several Unix installations – instead on just one server operating at the Erasmus MC.

About GemsTracker

GemsTracker is a PHP web application for “Patient Reported Outcome Measurement” (Proms) and healthcare research. The core software is Open Source, but all deployment projects have their own custom code containing both functional extensions and settings for the project.

GemsTracker projects store patient diagnostic and treatment data, therefore the security requirements are high.

Shared hosting but dedicated to GemsTracker

To bring down the per project cost of a GemsTracker installation we want to offer GemsTracker as a service, with multiple projects – each with their own code directories, databases and URL – sharing the same server, but otherwise not aware of each other. In other words: a dedicated (virtual) server for GAAS serving multiple projects with security isolation between projects.

Each project has its own code and data, but the differences are small enough that all projects can be managed the same way. Multiple projects on a server allows cost reduction, not only by having a single server and management procedure, but also sharing the costs for e.g. penetration testing. As the server hosts only managed GemsTracker projects a single test procedure can suffice for all projects.

The hosting of GAAS would be modeled on the current hosting of Pulse for Equipe zorgbedrijven: the production server there is tightly managed by the hosting company, while the developers have more access to the acceptance / demo / test server (development is done on localhost at the developers machines). At least two virtual servers are needed: a regulated server for production and a less regulated server for acceptance, demo and testing.

However, as the next table show, it would probably be better to use three virtual servers: the managed production server, the acceptance server (containing copies of the production data) and the less secure demo and test environments. All server should preferably share the same versions of OS, webserver and PHP version, but differ in the access the developers have. While the production and acceptance server would be used only by paying customers, the demo / test server would also contain test setup and demo installations for potential new customers.

	Production	**Acceptance**	**Demo**	**Testing**	**Development**
Role	Production	Testing before new release	Training and demonstration	Testing and trying out new features	Development
Security requirements	High! Contains patient treatment data, daily backup	High! Monthly copy of production data with patient treatment data (no backup of the database required)	Low. Identical setup data, but made up patients, no backup by hosting provider. Snapshot possible by developer	Low. Identical setup data, but made up patients, no backup by hosting provider. Snapshot possible by developer	Low. Contains garbage from development, no backup of the database
Availability	For most customers 5 x 8, for 24 x 7 for those who pay extra	5 x 8	5 x 8	5 x 8	None, hosted by developers
Usage	Paying customers, each with their own URL and certificate	Paying customers, shared URL and certificate	Paying and starting customers, shared URL and certificate	Paying and starting customers, shared URL and certificate	On localhost
Developer access	New versions and projects managed by the hosting company, read only access to the database	New projects managed by the hosting company, new versions by developers with read and write access to the database	Project and version management by developers including database creation	Project and version management by developers including database creation	Project and version management by developers including database creation

The projects share the hosting and management on the server: only one server needs to be maintained and managed and all projects on the server are managed using the same rules and procedures. When the number of customers on the production server exceeds the capacity a scenario is to start employing multiple productions servers. All still sharing the same setup and management.

Production server management

On the production server the code is preferably managed by the hosting company. They download new tagged versions using (currently) SVN or (in the near future) GitHub. Each installation has its own certificate and URL. E.g. www.caprimonitor.nl, depar.ciceroreumatologie.nl. Some projects even use multiple URL's. On the other hand the databases of the projects can be accessible using a single phpMyAdmin installation, where the developers have read-only access to the databases from restricted IP-ranges or similar security measures.

To summarize, the production virtual server should be tightly managed:

- All code is downloaded from tagged versions in SVN or Git.
- Updates are performed by the hosting company.
- Developers have only read-only access to the databases.
- New projects should provide an url and either provide an SSL certificate or pay for one.

The hosting company is responsible for both server and code management on this server.

Acceptance

The acceptance virtual server is similar to the production server, but with different rules and procedures. The main difference is that every installation uses the same URL, differentiated by sub-directory: capri-acc, cicero-acc. Developers manage the code themselves using SSL access and SVN/GitHub and have read and write access when using phpMyAdmin (from restricted IP-addresses). However, new projects should be setup by the hosting company as it is tightly linked to the production server.

Production / Acceptance integration

The Equipe server now copies the production database to the acceptance database every month. We would want to set this up for every project as it is a great tool for testing and removes the need for acceptance database backups.

This might require to separate the acceptance server from the demo and test environments. Developers would be able to create new projects only on the demo and test server, while the acceptance server is only changed in conjunction with the production server.

Demo/Test server

The demo/test virtual server should allow the developers and consultants to quickly create a test server installation for potential customers, by providing default demo/testing environments.

This server contains subdirectories named capri-test, capri-demo, cicero-test, cicero-demo, etc.… Test projects for potential new customers, say friesland-test, are created by the developers. They will be created with their own databases and software using an installation script.

When the “friesland” project is aborted, then the only cost should be the hours spent by the developer. Does the “friesland” project proceed to production, then they start to pay for hosting, server management and their own URL, etc…

To enable all this – without giving developers the freedom to everything – we will write a script that can be used to quickly create an installation by specifying the SVN/Git repository, the name of the project and whether it is a demo or test installation. The script then downloads the code, creates the necessary databases and makes the project htdocs directory accessible from the server webroot in a standard “projectformat”-demo|test format.

This script will be written by the developers. The hosting company is only responsible for server management and at the most weekly whole system backups.

As this server is used to attract new customers, the demo/test server is the first server that should be delivered.

The code is within the repository and projects can be setup in a very short time eliminating the need for daily backups by the hosting provider.

Future complexity

Multiple servers

As projects and the number of projects grows GAAS may take up too much server capacity. In that case we will have to start using multiple virtual servers.

Using the image of the previous server and the document management tools developed for the hosting this should be relatively easy to do.

Different projects for different developers

The initial plan is to give all developers access to all databases, be it read-only on the production server. As the number of projects grows this will no longer be possible as not all developers will have agreements with all customers.

Still: we can grant developers access to multiple database re-using individual accounts instead of creating new accounts for every project.

The same concern is valid for access to the code: initially all developers can see all the code for all the projects, but in the near future we should start to manage the access.

Responsibility and invoicing

We hope that within a year we will have a foundation or company that will be responsible for giving work orders to the hosting company and for invoicing the customers. As we cannot wait for this, at the start MagnaFacta B.V. will be the responsible party paying the invoices from the hosting company while sending invoices to the GemsTracker customers.

Timeline

The plan is to start using the test/demo server in November 2015.

The use of the production and acceptance servers is less fixed, but we currently expect to need it in January / February 2016.

To choose between hosting companies we will need prices for:

- Monthly rates for hosting the demo/test server.
- Expected setup costs for the demo/test server.
- Monthly rates for hosting the production and acceptance servers.
- Expected initial setup costs for the production and acceptance servers.
- Expected cost of adding a new customer (with one SSL certificate and without any setup for external communication).
- Expected cost for version management on the production server for both standard updates from code as well as applying hotfixes. (Not for major updates.)

We realize that specifying the expected costs is difficult when the actions are not clear. Specifying an hourly rate for the activities is therefore sufficient.