dotHIV IT Overview

March 2015

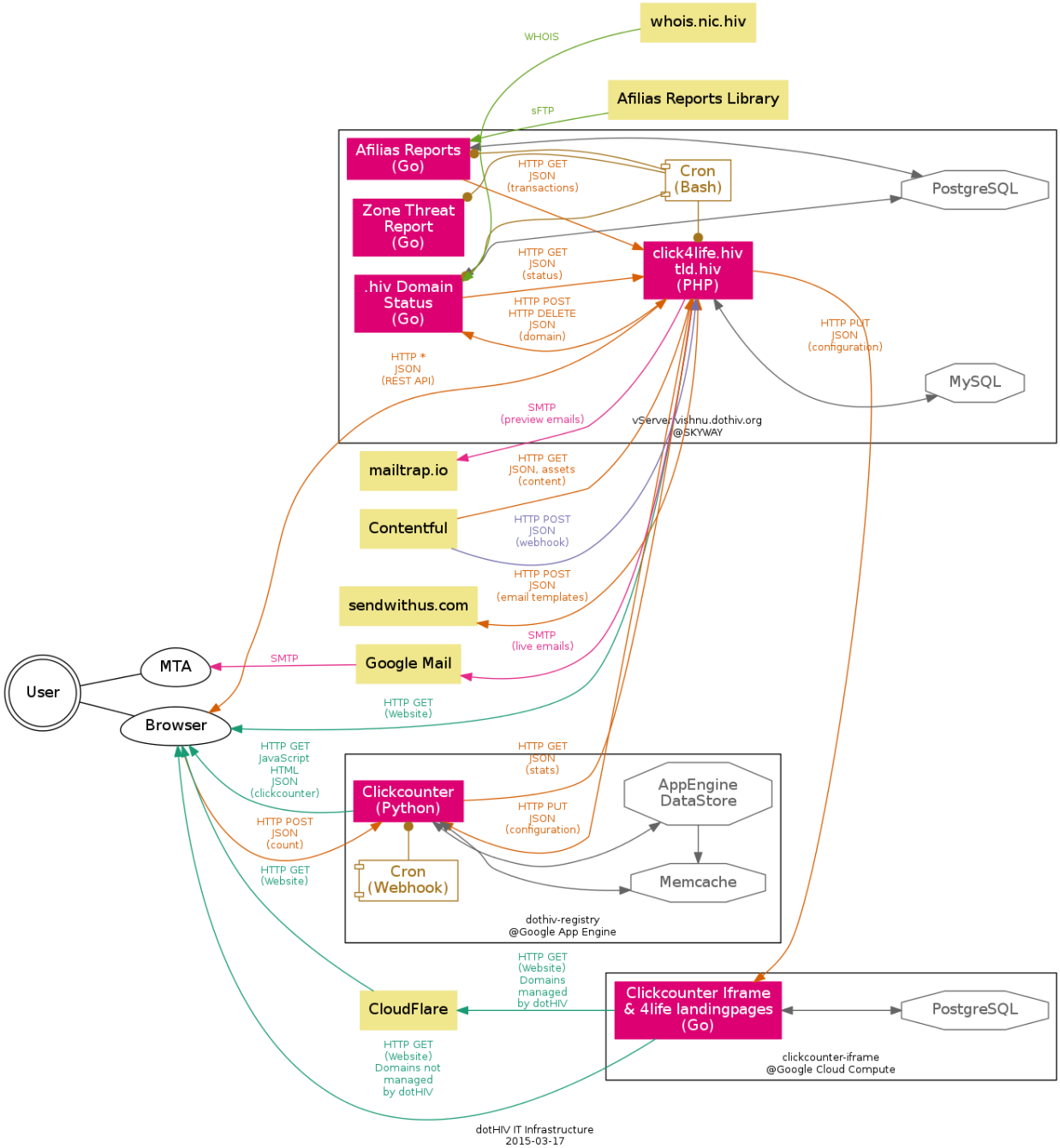
Introduction

dotHIV as a whole is built on a handful of individual building blocks. The concept was developed in 2012, implementation started in 2013 and has been in constant improvement since then. From a birdseye view dotHIV IT consists of these four systems:

1. a web application serving click4life.hiv and tld.hiv built with Symfony 2.3 (PHP5, MySQL 5.5) and AngularJS 1.2 hosted on a virtual root server (Ubuntu 14.04 LTS). Source code is in the dothiv repo.
2. a Google App Engine application serving the Clickcounter built in Google's limited subset of Python 2.7 hosted on the dothiv-registry project. Source code is in the clickcounter-backend repo.
3. a Google Cloud Compute instance 'clickcounter-iframe' (Ubuntu 14.04 LTS) that hosts the iframes (e.g. thjnk.hiv) and 4life landing pages (e.g. caro4life.hiv) via a microservice (Go 1.2, PostgreSQL 9). Source code is in the clickcounter-iframe repo. Those domains are set up via CloudFlare to reduce server load.
4. a Google Cloud Compute instance 'jenkins' (Ubuntu 14.04 LTS) that hosts the continuous integration server.

They are represented by black boxes in the overview chart below. The integration server is not pictured.

Components



Note: Source code and a PDF version of this chart can be found at https://trello.com/c/jKWtYMt3

Web Application

The web application (labeled “click4life.hiv…”) is built with Symfony 2.3 LTS and AngularJS 1.2 and is the central element in dotHIV’s IT world. This is the place where customers configure their clickcounters, it publishes the configuration to the clickcounter instance, sends reminders to customers and it’s where reports are created via the admin dashboard. Its source code is organized

into more than a dozen bundles to reduce cohesion. In general we followed a test-driven development approach and leverage the frameworks dependency-injection feature to build a software that follows the SOLID principle. Tests are run continuously on a jenkins instance (not pictured) which is hosted on a separate Google Cloud Compute VM.
We rely on an internal reverse proxy to cache all responses generated by the web application.
Data is persisted in a local MySQL database.
The initial version was based on Twitter Bootstrap 2, AngularJS 1.1 and not mobile friendly. Step by step we refactored existing pages and built new pages with Bootstrap 3, Angular JS and a mobile friendly approach.
Source code: https://github.com/dothiv/dothiv

All content on the website is managed via Contentful, an API-CMS. The content is fetched via their API and stored locally in the database. Images and attachments are downloaded. All content is served then from the local webserver. Support for webhooks has been implemented that allows nearly real-time propagation of changes made in Contentful to the website.
Source code for this functionality is bundled in: ContentfulBundle.

The web application is also in charge of sending email reminders and transactional emails. On the live system they are sent out via the Google Mail Accounts `domains@tld.hiv` or `non-profit-support@click4life.hiv`. On the preview system they are sent to a mailtrap.io mailbox. The content of the transactional emails is managed in Contentful (content model: eMail). The templates are generated with the help of Jekyll. The project transactional-emails is the authoritative source for the email templates.

As this approach is quite cumbersome, the reminders that have been implemented in February 2014 are using the template rendering api of sendwithus to generate the content of these emails.
Source code for this functionality is bundled in: UserReminderBundle.

Part of the web application is the admin dashboard which is used to provide insights into registered domains and non-profit applications for dotHIV staff. Please refer to this separate document for a detailed description of the module.

Afilias Reports

dotHIV's concept requires a domain owner to configure the clickcounter for their domain and setup their domain in a way that the clickcounter is displayed. After a domain has been registered we send the registrant contact an email asking them to join the micro-donation program and configure the clickcounter. Afilias provides hourly and daily reports about domain transactions in text files which are accessible via sFTP. Those files are fetched hourly and processed by the afilias-registry-operator-reports service which is implemented in Go.
This service provides a REST API which is called by a CLI job of the web application every hour. The transactions are pulled by the API on demand and applied on the web application. Like with all other services we decided to not use a message queue but provide an idempotent way to transfer data. If the

web application needs to be refactored it can fetch all transactions again from this service and apply them again.
Source code: https://github.com/dothiv/afilias-registry-operator-reports

.hiv Domain Status

The web application registers and unregisters domains at this service and it crawls every domain once a day checking if the clickcounter is installed correctly. The status of the domain is persisted in the service's PostgreSQL database and fetched by the web application. It is implemented in Go.
Source code: https://github.com/dothiv/hiv-domain-status

Zone Threat Report

This service generates a report file (example) for every registered domain using the zone file and other checks to determine if .hiv are used illegally. The report is generated every monday and sent out via email.
Source code: https://github.com/dothiv/zonethreatreport

Clickcounter (Iframe window)

The clickcounter is in charge of delivering the JavaScript, HTML and CSS for displaying the clickcounters / individual iframe windows on the .hiv domains and counting the visitors per domain. The source code is separated in two projects. For the visible part (frontend) `clickcounter` and the backend `clickcounter-backend`. The frontend uses vanilla JavaScript. The backend uses Google AppEngines Subset of the Python 2.7 API. Data is persisted in AppEngine's DataStore, an object store implementation. For performance reasons Memcache is used heavily to offload writes to an asynchronous job.
Source code (frontend): https://github.com/dothiv/clickcounter
Source code (backend): https://github.com/dothiv/clickcounter-backend

Clickcounter iFrame & 4life.hiv landing pages

To simplify the deployment of a new .hiv domain we implemented a web service that serves iframes and landing pages for .hiv domains. It is implemented in Go and stores its data in a local PostgreSQL database. It is run on a VM on Google Cloud Compute. The individual configuration for the domains is pushed to it from the web application.
Customers are asked to use `iframe.clickcounter.hiv` as the CNAME for their domain's www record and redirect the apex domain to www.
To reduce load on the machine we set up domains that are controlled by dotHIV to use the free CloudFlare plan.
Source code: https://github.com/dothiv/clickcounter-iframe

Deployment

The **dothiv** project is automatically deployed to the preview and live host (which both reside on the vServer) when code is pushed to the master respectively the live branch. As the deployment scripts contain sensitive information (e.g. encrypted SSL certificates) it is hosted in the github repo

deployment (`git@git.tld.hiv:deployment.git`) on the vserver. The cronjobs triggering the deployment are kept there, too.
The **clickcounter-backend** project is automatically deployed to live when code is pushed to the master branch.
The **afilias-registry-operator-reports, hiv-domain-status, clickcounter-iframe, zonethreatreport** project is manually deployed after a release has been created.

Keywords, References

- vServer: https://trello.com/c/8UZo9AJt
- Google App Engine Project: https://appengine.google.com/dashboard?&app_id=s~dothiv-registry
- Google Cloud Compute Instance: https://console.developers.google.com/project/dothiv-registry/compute/instancesDetail/zones/europe-west1-b/instances/clickcounter-iframe
- GitHub: https://github.com/dothiv
- SSL certificate: https://trello.com/c/SN9k9Ns9
- Domains: https://trello.com/b/cVRN9dud/inventory
- CloudFlare: https://trello.com/c/13ve4kF9
- Admin Dashboard: https://tld.hiv/admin
- SOLID principle: http://en.wikipedia.org/wiki/SOLID_%28object-oriented_design%29
- Contentful: https://app.contentful.com/
- mailtrap.io: https://mailtrap.io/