P22

A cloud infrastructure for R reports

Gergely Daróczi[1,2,3]; Aleksandar Blagotic[1,2]

Founder, Easystats Ltd, UK | Assistant lecturer, PPKE BTK, HUN | PhD candidate, Corvinus University, HUN | R/web-developer, Easystats Ltd, UK | MsC student, University of Niš, SRB

About rapporter.net

Rapporter is a **web application** dedicated to create comprehensive, reliable, literate and reproducible **statistical reports** on any mobile device or PC, using an intuitive user interface.

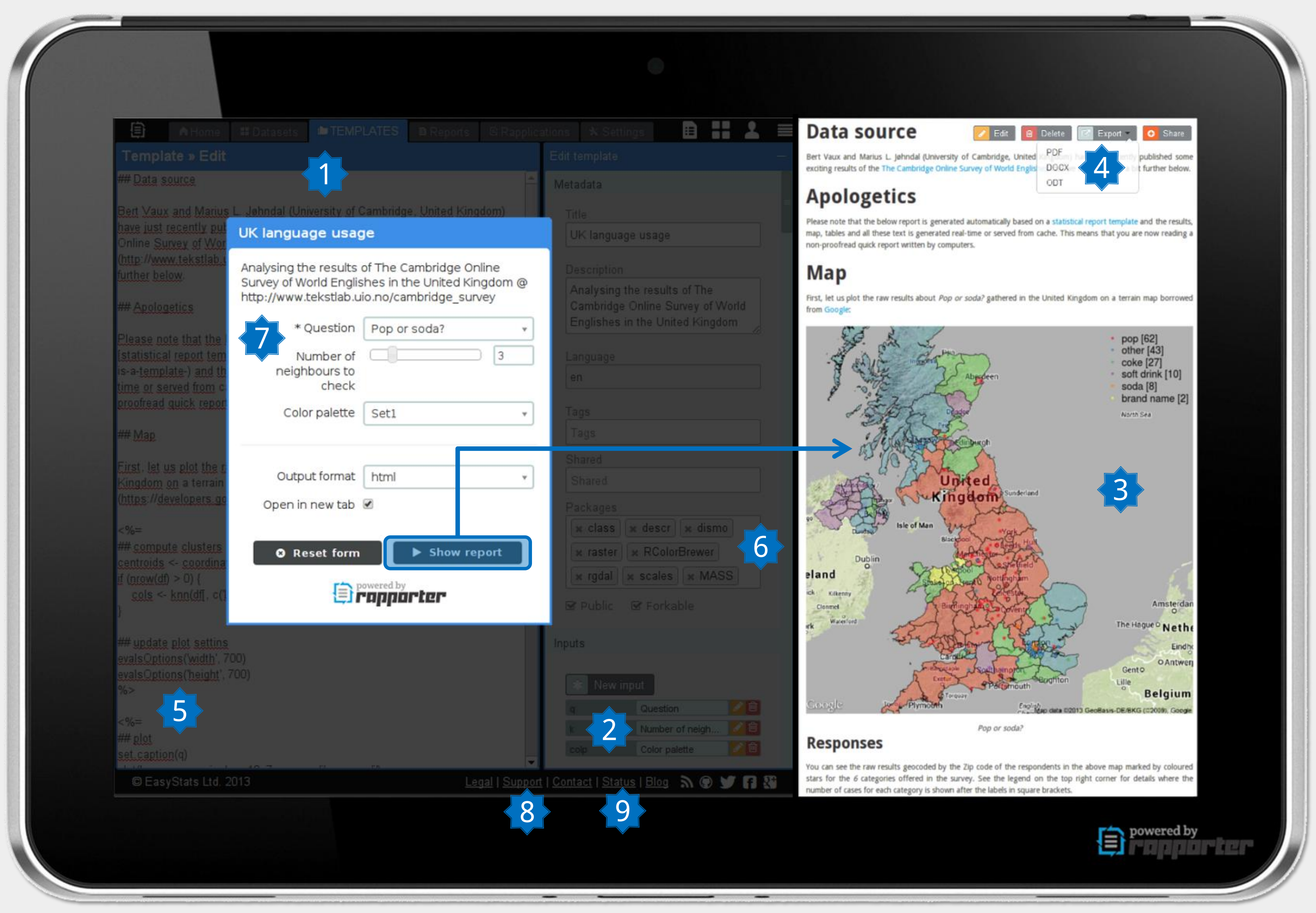
The application builds on the power of *R* beside other **open-source technologies** like *rApache* enabled *Apache* and *nginX* webservers on *Ubuntu LTS*, *Couch* and *Mongo* as our robust NoSQL database backends, *GlusterFS* for the distributed and replicated network filesystem, *Ruby on Rails* with a bunch of gems like *thin* webserver for the content management system and hundreds of R packages from CRAN and GitHub.

Our main idea was to create a webapp for statistical analysis that can be used in **any modern browser** even on tablets/iPads and doing the heavy computations on the **scalable server side** in a triply-**secured environment** (user roles, *sandboxR* and *RAppArmor* packages along with *AppArmor*).

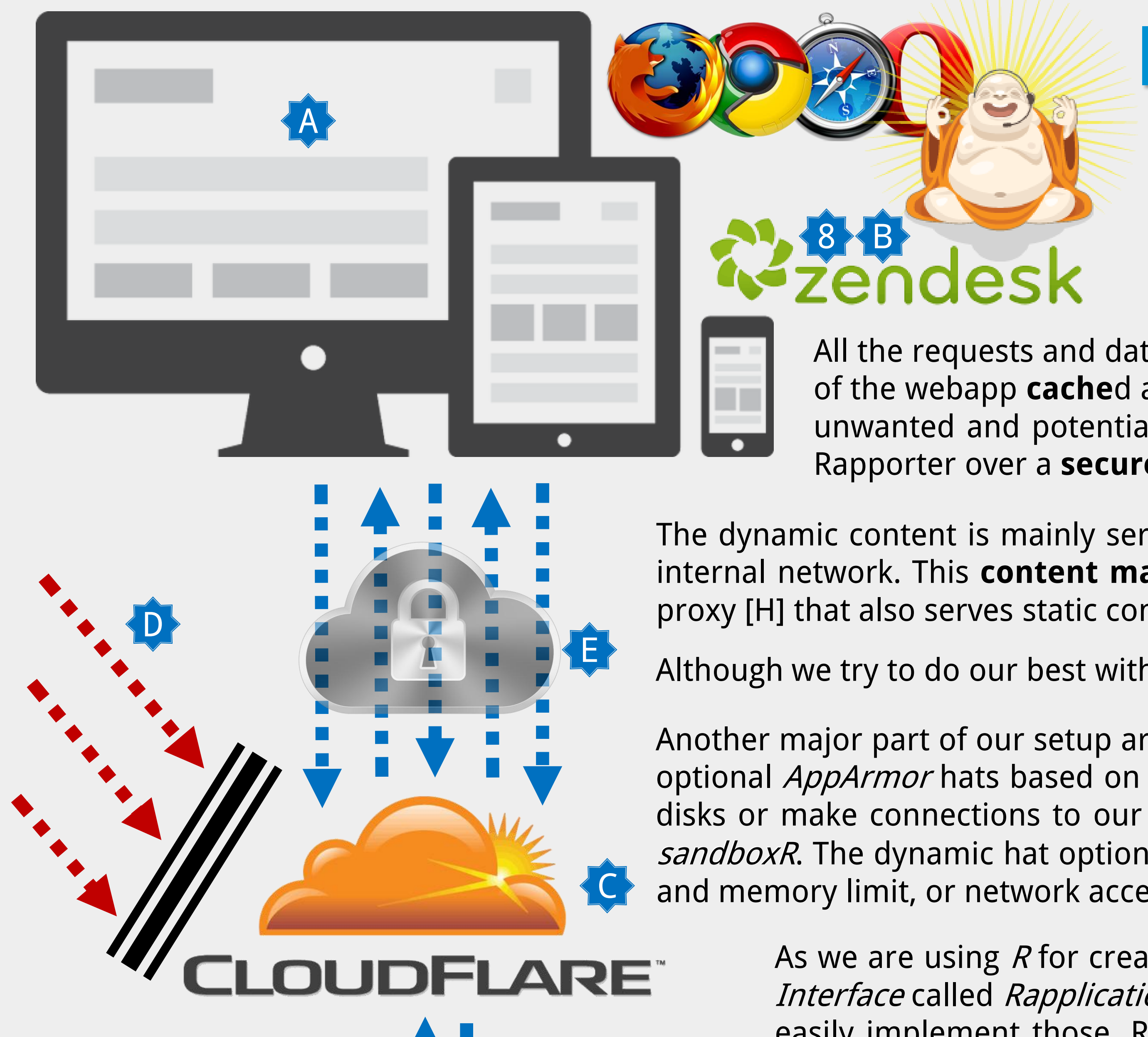
So the user simply uploads some data in various formats and can apply a wide variety of **statistical templates** [1] on those with the help of diversified input methods [2], then export the reports [3] to *pdf*, *odt*, *docx* or *HTML* formats [4]. As the templates are written in a *brew*-like syntax [5], it is extremely easy to fork or modify those with even minimal programming skills or write new methods from scratch using any *R* packages [6] and functions.

Some might consider Rapporter as a customisable and user-friendly **graphical user interface to R** running on the cloud, but the recently introduced API highly extends this use-case. The so-called *Rapplications* [7] can be included in any homepage without any programming skills, providing an easy to use front-end to any statistical templates with publication-ready outputs.

Feature requests, bug reports or other questions are very welcomed on our support/help page [8, B].



Infrastructure

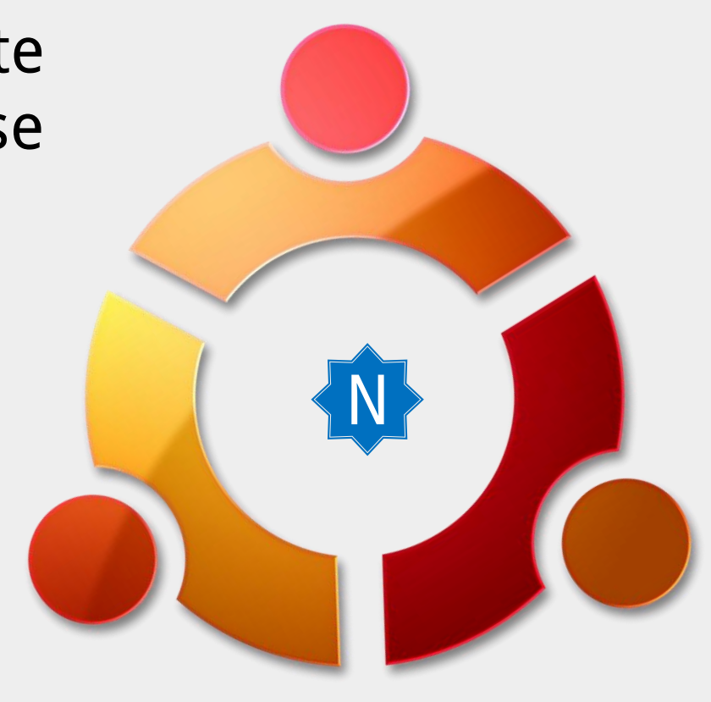


The goal of Rapporter was to provide a front-end to *R* in **all modern browsers** running on **various platforms** [A] – let it be a desktop, a notebook, tablet or a mobile phone. Users can access their data, reports and statistical tools stored in the Rapporter cloud from any place over the Internet and even do it collaboratively with other fellows and contributors.

A minor but useful part of the infrastructure is hosted at *Zendesk* that provides an extensible **knowledge base** and **support forum** [B].

All the requests and data packets sent by the clients to Rapporter servers hit our *Content Delivery Network* provider [C] first that would return all static content of the webapp **cache**d at several locations around the world for improved response times. The CDN also operates as a front-line **firewall** and filters out some unwanted and potentially dangerous packets and queries [D] beside minimizing the risk of (Distributed) Denial-of service attacks. Users can optionally use Rapporter over a **secure channel** by HTTPS protocol [E], as the data transmitted to and from CloudFlare is encrypted on demand for improved security.

The dynamic content is mainly served by our *Ruby on Rails* [F] workers in the means of a cluster of *thin* servers [G] running inside of our private internal network. This **content management system** is made of several separate threads replying to user request via a **load balancing** reverse proxy [H] that also serves static content, plus JavaScript, CSS and image assets.

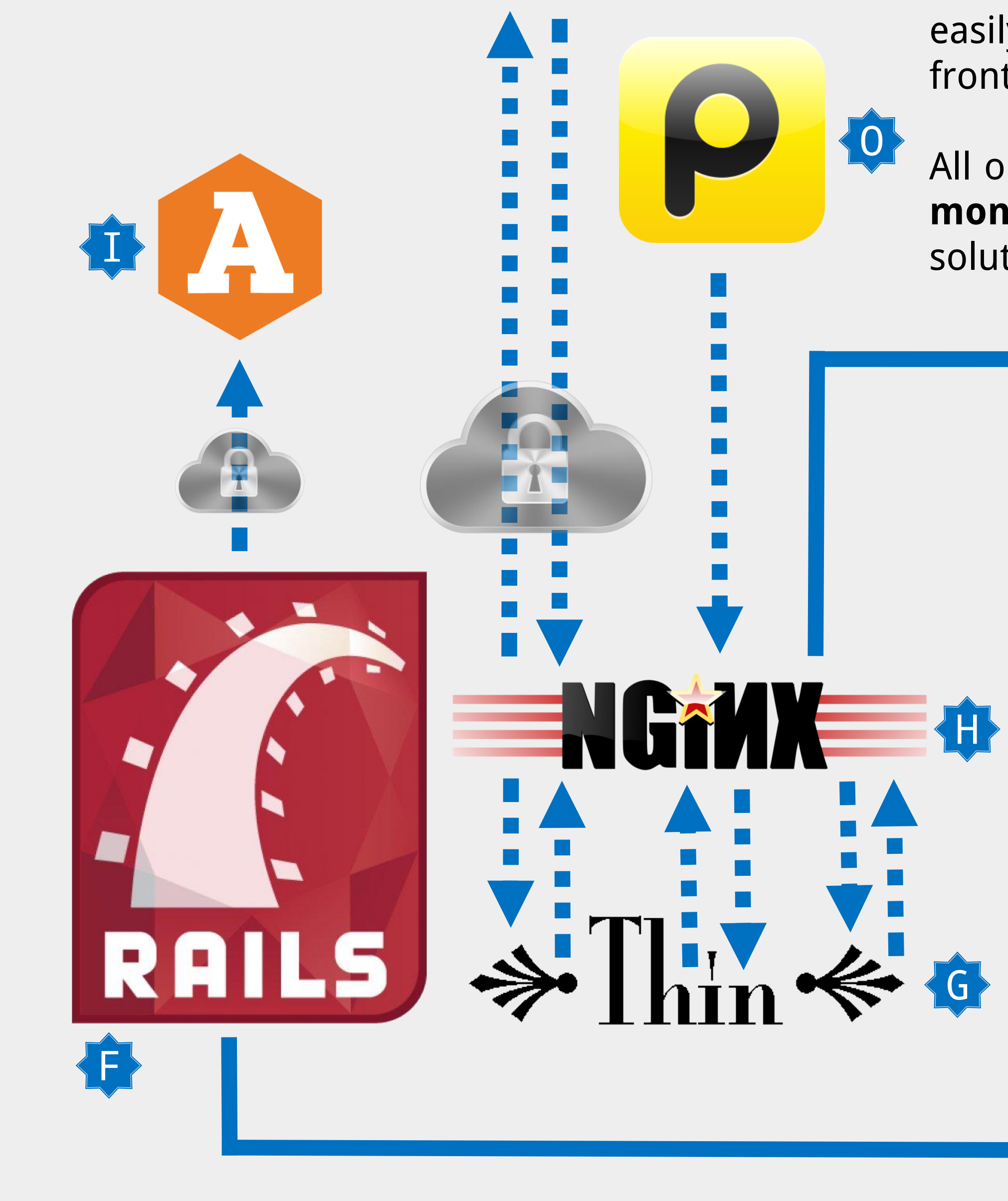


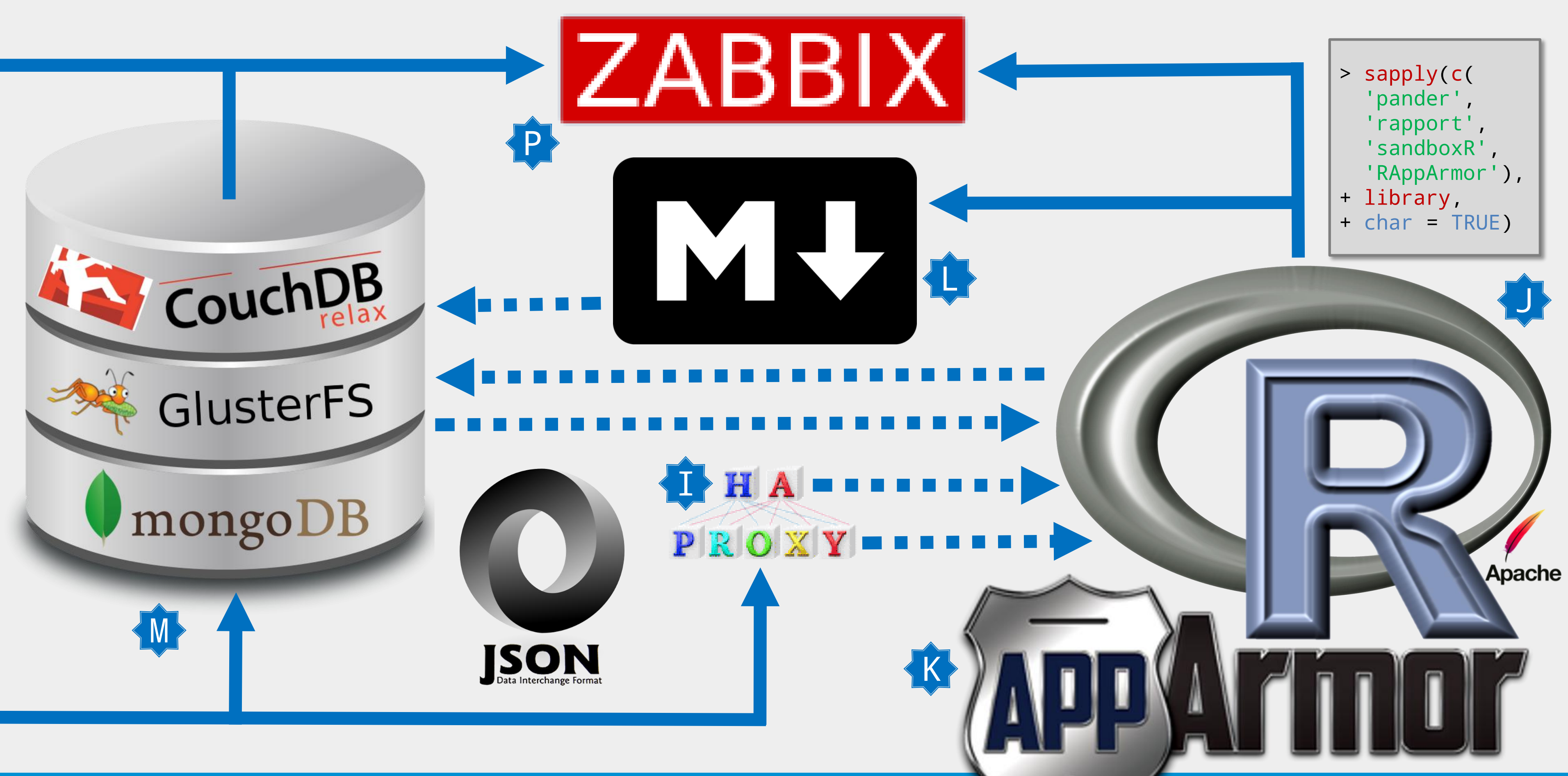
Although we try to do our best with deploying working code on production servers, we also collect possible Rails **error messages** with *Errbit* [I].

Another major part of our setup are the *HAProxy* cluster [I] of *rApache* based *R* workers [J] running within an enforced *AppArmor* [K] profile and optional *AppArmor* hats based on user privileges. This latter Linux kernel **security module** ensures that the users could not directly touch the disks or make connections to our databases – even if some malicious code would somehow escape our in-house developed **sandbox** called *sandboxR*. The dynamic hat option allows fine-grained control over the **hardware resources** on a per-user basis in the means of e.g. CPU power and memory limit, or network access. Please see some further **security considerations** below.

As we are using *R* for creating **complex** or one-time and temporary **reports** via a *Graphical User Interface* or the recently introduced *Application Programming Interface* called *Rapplications*, our home-made internal *R* functions do not deal with any statistical problems, but rather provides an environment for the users to easily implement those. Rapporter is basically made of our open-source ***rapport* and *pander* packages** (please see below) beside the above described Rails front-end and hardened security tools, and the data, methods and results all bundled in various JSON driven databases [M].

All our servers are running *Ubuntu LTS* [N] on 64 bit with a decent amount of memory and CPU cores optionally dedicated to VIP customers, and continuously **monitored** 24 hours a day, 7 days a week via the public [9] *Pingdom* availability monitor [O] and a more detailed and technical. enterprise-class monitoring solution with thousands of metrics, called *Zabbix* [P] – beside Google Analytics of course.





```
> sapply(c(
  'pander',
  'rapport',
  'sandboxR',
  'RAppArmor'),
+ library,
+ char = TRUE)
```

Data storage [M]

Although administering and maintaining several similar database engines might not make much sense in most setup, we use two NoSQL databases for **improved performance**. *CouchDB* is awesome for its disk-based B-tree views, simple attachment concept and eventual consistency schema, while *MongoDB* makes the Rails models a lot more convenient to work with.

Gluster is a network filesystem that stores R generated images on a **replicated** and optionally distributed storage attached to the highly available Rails servers.

References (in order of mention)

❖ rapporter.net
❖ r-project.irg
❖ rapache.net
❖ nginx.org
❖ www.ubuntu.com
❖ couchdb.apache.org
❖ www.mongodb.org
❖ www.gluster.org
❖ rubyonrails.org
❖ code.macournoyer.com/thin
❖ github.com/rapporter/sandboxR
❖ hackme.rapporter.net
❖ github.com/jeroenooms/RAppArmor
❖ johnmacfarlane.net/pandoc
❖ www.cloudflare.com
❖ github.com/errbit/errbit
❖ www.zabbix.com
❖ www.zendesk.com
❖ www.pingdom.com
❖ rapport-package.info
❖ rapporter.github.io/pander

Security considerations for evaluating untrusted R commands in the cloud

The major drawback or rather just difficulty of running *R* in a shared, hosted session and environment is that *R* was written to be used on the localhost, and more importantly: by a single user. So the core has bunch of internal functions granting **access to the storages**, **network interfaces** and **other peripherals**, even letting users to easily launch **system commands** or bring up a **live console**. The possible workarounds were discussed by Jeroen Ooms (2013) in "Security Policies in R on Linux" at JSS. Although we highly appreciate Jeroen's *RAppArmor* and also using that on all our *R* workers, we still see some security holes to be addressed by other tools – like *sandboxR*. Please see some examples on the right demonstrating the features of the two package.

So we ended up using a **sandbox in an enforced hard-limit AppArmor profile** with some fine-tuned and customizable hats, so that the internal functions could access those resources locked from users in the means of direct access. But hey, if there is a convenient way to load datasets (with the help of internal functions), why would anyone try to `read.table` from the disk after all?

```
> system("rm -fr /")
Forbidden function called: system.

> foo <- "system('/usr/bin/bash')"
> eval(parse(text = foo))
Forbidden function called: parse.

> get('system')
Forbidden function used as symbol: system.

> get(paste("", "y", "tem", sep = "s"))("whoami")
Forbidden function used as symbol: system.

> options("sandboxR.disabled.options")
Not available option(s) queried.

> forkbomb <- function(){       #  (c) Jeroen Ooms
+  repeat                       # RAppArmor package
+    parallel::mcparallel(forkbomb())
+ }
> forkbomb()
Error in mcfork() :
unable to fork, possible reason:
Resource temporarily unavailable
```

Reproducible statistical templates

Our *pander* package acts as a wrapper around *Pandoc* [L], the universal document converter, automatically **mapping *R* objects to markdown** and then transforming the resulting markdown files to **other document formarts**, and *rapport* provides a way to **create reproducible, dynamic and literate statistical templates** with dynamic inputs for easy and iterative reporting.