A cloud infrastructure for R reports

Gergely Daroczi¹,²,³; Aleksandar Blagotic¹,²

1 Founder, EasyStats Ltd, UK | Assistant lecturer, PPKE, BTK, HUN | PhD candidate, Corvinus University, HUN | K/web-developer, EasyStats Ltd, UK | MSc student, University of Niš, SRB
2 PhD candidate, Corvinus University, HUN
3 Software engineer, EasyStats Ltd, UK

About rapporter.net

Rapporter is a web application dedicated to creating comprehensive, reliable, 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 Apache enabled Apache and nginx webservers on Ubuntu LTS, Couch and Mongo as our robust NoSQL database backends, GlusterFS for the distributed filesystem, and Gluster for the clustered filesystem, with a bunch of gems like thin webservice for the content management system and hundreds of R packages from CRAN and Github.

Our main idea was to create a webservice for statistical analysis that can be used in any modern browser even on tablets/phones and doing the heavy computations on the scalable server side in a triply-secured environment (user roles, sandbox 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, DOCX, PDF or HTML formats [4]. As the templates are written in a drawlilex 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 Rapporter RAP is integrated in any homegroup or personal programming environment, 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 (B, B).

Infrastructure

The goal of Rapporter was to provide a front-end to R through 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 collaborative 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 if the requested network at several locations around the world for improved response times. The CDN also opens, as a front-line firewall and filters out any 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 our private intranet network. This content management system is made of several separate threads replying to user requests 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 ErbIf [I].

Another major part of our setup are the HAProxy cluster [J] of Apache-based load balancers [K] running within an enforced AppArmor [L] profile and optional AppArmor hats based on user privileges. This latter Linux kernel security module ensures that the user 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 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 Interfaces for Rapportations, our home-made internal IF functions do not deal with any statistical issues but rather provide an environment for the user 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 [O] Pingdom availability monitor [Q] and a more detailed and technical, enterprise-class monitoring solution with thousands of metrics, called Zabbix [P] – beside Google Analytics of course.

References (in order of mention)

- rapporter.net
- project.org
- rpackages.net
- rpkg.org
- www.ubuntu.com
- couchdb.apache.org
- mongodb.org
- www.gluster.org
- rubyonrails.org
- code.macourmoyer.com/thin
- github.com/rappor/sandboxR
- hackme.rapporter.net
- github.com/jeroenooms/RAppArmor
- github.com/jeroenooms/sandbox
- www.cloudflare.com
- github.com/erbit/erbit
- www.zabbix.com
- www.zendesk.com
- www.pingdom.com
- sitemaps.package.info
- github@rapporter.io/pander

Data storage [M]

Although administering and maintaining several 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.

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 8 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 RAppArmor 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?

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 formats, and rapport provides a way to create reproducible, dynamic and literate statistical templates with dynamic inputs for easy and iterative reporting.

Contact: daroczi@rapporter.net | alex@rapporter.net