CNCF DevStats

Łukasz Gryglicki, Senior Software Developer, CNCF, @lukaszgryglicki

This presentation is available at:
https://github.com/cncf/presentations

CLOUD NATIVE
COMPUTING FOUNDATION

A tool to visualize developers statistics data for projects using GitHub repositories.

CLOUD NATIVE
COMPUTING FOUNDATION

DevStats

- Main repository on the GitHub: https://github.com/cncf/devstats
- Example deployment for a single project (Homebrew): https://github.com/cncf/devstats-example
- Written in Golang, using Postgres database for data and time-series storage.
- GitHub archives and git are used as the main datasource. GitHub API as a helper.
- Using Grafana for charts/dashboards, Let's encrypt for HTTPS, Apache proxy and wildcard domain names.
- Using Packet.net bare-metal servers (production and test) with Ubuntu 18.04 LTS.
- All components Open Source.

CLOUD NATIVE
COMPUTING FOUNDATION

DevStats

Defining projects:

- Projects are defined in projects.yaml file.
- The only requirement is that project must have a one or more repository/organization on GitHub.
- You need to specify org/repo configuration, start date, release names format, and optional CNCF join date, files skip pattern etc.
- You can create your own instance by forking the main repo (or devstats-example repo) and following this guide.
- projects.yaml is used as a DevStats config file for running cron sync job every hour.
- More detailed info about adding new projects is here.
- New projects are automatically deployed from CI/CD pipeline, see deploy all script.

CLOUD NATIVE
COMPUTING FOUNDATION

DevStats

Metrics

- You can use various kinds of metrics
- Charts (historical data and current values, moving averages), example commits in repository groups
- Tables/Histograms (data for last periods, for example last 10 days etc.): companies summary
- Heatmaps (histograms over time), example timezone stats
- Single stats for showing current metric value, for example issues age
- Bar charts showing aggregated stats per periods, and percent charts: for example countries stats
- Other (like the main dashboard showing all projects and basic hourly activity chart)

DevStats

Metrics

- You need to define a metric in projects metrics.yaml file.
- Metric requires SQL file to calculate its value, for example activity repository groups.
- Finally, you need a Grafana dashboard displaying results, for example activity repository groups.
- Metrics are calculated every hour from a cron job.
- Different periods are calculated at different times. Hourly metrics every hour, daily metrics at a specific hour, weekly metric in last hour of the week etc.
- There are also monthly, quarterly, yearly metrics, aggregated periods (like 24 hours MA), histogram ranges (like last week), past ranges like v1.9.0 - v1.10.0 (annotation ranges) - only once.
- For adding new metrics see this.

DevStats

Repository groups

- Each project can define its own so-called “repository groups”.
- Allow grouping multiple GitHub repositories/organizations into a special group.
- Also allows specifying single files/directories from a GitHub repository to belong to different repository groups (file level granularity - only used by Kubernetes)
- For a particular project ‘All CNCF’ each CNCF project is defined as a single repository group, see project statistics example.
- By default, every repository is a single repository group (used by most of the CNCF projects).
- Only kubernetes have a non-default configuration.

DevStats

Company affiliations

- We are importing user - company affiliations from cncf/gitdm
- Each user(actor) can have multiple company affiliations with start-end dates
- We have tried many automatic tools to research this (including commercial ones), but the data was rather poor
- We have a contractor working on this manually (researching GitHub profile, email, username on various sources: GitHub, LinkedIn, Twitter, Google and many more).
- Every time a new project is added to CNCF we're researching top unknown actors starting from those who made most contributions
- We are also rechecking top contributing ones regularly to check for possible affiliations changes.
- The final data file is here.

CLOUD NATIVE
COMPUTING FOUNDATION

DevStats

Company affiliations

- cncf/gitdm also allows users to update their affiliations by updating a user-readable text file and creating PR against it.
- Some users don't want their email/name listed everywhere, we are allowing defining a list of identifiers that will never be processed, see here, this is also compatible with GDPR.
- cncf/gitdm is periodically updated by importing manual work from those spreadsheets: first, second.
- Each user can have multiple GitHub accounts and multiple emails. Single GitHub account could have more than 1 email in the past, and 1 email could have been connected to multiple GitHub accounts in the past
- Users are changing their affiliations, so we are keeping time ranges. It's all rather complex.

DevStats

Architecture

- We are using GitHub archives as the main data source. It stores each hour of GitHub activity (for all GitHub projects) as a zipped array of JSONs (each JSON is a single GitHub event).
- Each hour contains about tens of thousands of JSONs. Unfortunately, GHA (GitHub archives) is not 100% reliable recently (starting from about Feb 2018).
- It sometimes misses some event due to capacity problems. GHA & GitHub are working on the long-term solution
- DevStats uses so-called short-term solution to deal with it. It uses GitHub API calls (very slow) each hour to make sure we're not missing issues/PRs state changes. It also registers all GitHub API results as separate event types in the Postgres database

CLOUD NATIVE
COMPUTING FOUNDATION

DevStats

Architecture - BigQuery as an alternative

- BigQuery - you can query which data you want, but the structure is flat and most of the useful data is inside the JSON payload field, which is huge and consumes a lot of transfer. (Full JSON is in a single column that must be queried, processed and paid for)
- This limits usage due to the need of parsing that JSON in DB queries
- BigQuery is commercial, paid and is quite expensive. It is not a standard SQL.
- GitHub archives already contain all the data BigQuery has, and it can be downloaded for free and parsed by DevStats discarding all the data that is not needed.

CLOUD NATIVE
COMPUTING FOUNDATION

DevStats

Architecture - GitHub API

- You can get the current state of the objects, but you cannot get repo, PR, issue state in the past
- It is limited by GitHub API usage per hour (5000), which makes local development harder.
- With this limit, it would take more than 3-4 months to get all Kubernetes GitHub events (estimate).
- It is much slower than processing GitHub archives or BigQuery.
- You can use GitHub hook callbacks, but they only fire for current events.
- We are only using it for current data to make sure we're not overlooking issues/PRs state changes (due to GHA capacity problems, missing events that started in Feb 2018 and aren't solved yet by GHA/GitHub team)

DevStats

Architecture GHA (GitHub Archives)

- It contains all GitHub activity for every hour
- DevStats fetches every hour and discards all events not related to projects defined in projects.yaml.
- Processing data for all 100+ Kubernetes repos from 1/1/14 takes just about 4-5 hours. It is done once, after that you only need to process single new hour - hourly via the cron job.
- You have all historical states of all events from the past (GitHub API cannot return past object state).
- Processing of GitHub archives is free, GHA is open source.
- GitHub events format changed in 1/1/15 - DevStats supports both old and new formats.
- DevStats can optionally save all JSONs processed (instead of processing them in memory).

CLOUD NATIVE
COMPUTING FOUNDATION

DevStats

Architecture - git

- The project also uses git to store a local copy of all projects repositories to allow file related analysis (like a list of files changed in a given commit, file sizes etc.).
- All projects repositories are cloned into the local directory
- All projects repositories are updated every hour using git (part of standard cron workflow).
- It allows file name analysis, assigning given files from repositories to repository groups (file level granularity) and file size analysis (for example file size growth in time).
- cncf/gitdm uses git to GitHub connection from git pushes to allow connecting committers with their GitHub accounts. It is used to determine committers company affiliations.

CLOUD NATIVE
COMPUTING FOUNDATION

DevStats

Architecture - tools

- DevStats is a set of go programs designed to do their jobs (planned to be used as a microservices in the future)
- structure - creates a DB structure, tables, views, indexes etc.
- gha2db - fetches data from GHA and saves to a Postgres database
- calc_metric - calculates given metric and saves results in a Postgres database (used as a time series database in this case, see this).
- gha2db_sync - higher level tool, calls the other tools to sync data: calls tags, annotations, gha2db, get_repos, columns and some other tools. Used to make full sync of a single project.

CLOUD NATIVE COMPUTING FOUNDATION

DevStats

Architecture - tools

- devstats - highest level tool, called directly from a cron job. Reads projects.yaml and calls gha2db_sync, get_repos and all other tools for all projects defined in projects.yaml.
- get_repos - this is a tool to sync “git” data source. Updates local repositories copies and does all git related analysis.
- ghapi2db - calls GitHub API to check current issues/PRs state (handles all GitHub API calls, requires “/etc/github/oauth” otherwise is dramatically limited by public GH API access and will probably fail). It is needed to make sure that gha2db leaves all issues/PRs in a correct state (GHA misses some events sometimes, starting Feb 2018).
- import_affs - imports company affiliations from cncf/gitdm.

CLOUD NATIVE
COMPUTING FOUNDATION

DevStats

Architecture - tools

- annotations - processes git tags defined on the projects main repo (each project has its own main repo defined in projects.yaml). If a given tag defined on the main repo matches regular expression for annotations defined for a given project - it is added as a version. For example v1.1, v1.2,..., v1.9. All those annotations are used in tabular dashboards to generate statistics for inter-annotation periods like v1.3 - v1.4 or v1.9 - now etc.
- tags - calculated tags used for most dashboards drop-down values, like the list of repository groups, usernames, reviewers, bot commands etc.
- columns - make sure that Postgres TSDB tables contain given columns on given tables - for some sparse table results, some columns can be missing causing PSQL errors in dashboards

CLOUD NATIVE
COMPUTING FOUNDATION

DevStats

Architecture - tools

- webhook - This is a CI signal listening tool (DevStats uses Travis CI for continuous integration). Once Travis CI run is complete, it sends webhook that is received by webhook tool. It decides about automatic deployment, install etc. depending on CI status: branch name, passed/failed etc.
- There are also more small tools for local development - for example for mass update Grafana dashboards, mass replace texts, variable defining/substitutions on a Postgres database, query execution, merge databases tools etc.

DevStats

Architecture - database

- DevStats uses Postgres for data sources storage (GHA, GitHub API, git)
- DevStats uses Postgres as a TSDB (time-series database). Metrics results are saved in TSDB and used by Grafana to display dashboards
- Most of the GitHub events tables contain “event_id” column and the corresponding object stats from the time of “event_id” was generated.
- So, for example, a single GitHub issue is saved in the gha_issues table with id (issue ID) and event_id. This single issue can have even 1000s of state with a different event_id (historical states).
- Some tables are not changing in time and have no event_id, for example actors, orgs, repos etc.

CLOUD NATIVE
COMPUTING FOUNDATION

DevStats

Architecture - database

- Most tables use columns starting with “dup_”, they’re duplicates of the original column from another table and allows faster queries by saving additional joins.
- Those “dup_” columns are filled automatically by DevStats tools.
- Most tables have indexes on columns that are used in at least one metric; some have indexes on functions like for example lower(login) etc.

CLOUD NATIVE
COMPUTING FOUNDATION

DevStats

- DevStats uses multiple Grafana instances running on the same bare metal server provided by packet.net. Both test & prod servers are 48 CPU machines. DevStats tools are multithreaded.
- Devstats uses wildcard domains *.cncftest.io and *.devstats.cncf.io. The first point to a test server, the second point to a production server.
- Apache proxy is used to point to a proper Grafana instance depending on the current hostname (any from the wildcard domains)
- Let's encrypt is uses as an HTTPS/SSL provider
- Google analytics is running on production sites *.devstats.cncf.io.
- Grafana dashboards are defined as JSONs, for example here for Prometheus. Each CNCF project has its own set of Grafana dashboards. Anonymous login with read-only access is enabled.
- See dashboards documentation for more info.

CLOUD NATIVE
COMPUTING FOUNDATION

DevStats

Some history:

- This toolset was first implemented in Ruby with Postgres database.
- Then MySQL support was added.
- MySQL proved to be slower and harder to use than Postgres.
- Entire toolset was rewritten in Go.
- Go version only support Postgres, it proved to be a faster than the Ruby version.
- Developers affiliations import from cncf/gitdm was added.
- Finally, the Ruby version was dropped.
- Git data source was added to support file-level granularity repository groups definition for Kubernetes.
- InfluxDB was used as a time series database. Then it was dropped and replaced with Postgres.
- Postgres is faster as a time series database than a dedicated time series database InfluxDB.
- Some additional events not included in GitHub events (like (un)labeled, (de)milestoned, referenced, (un)subscribed etc.) are fetched using GitHub API. This requires GitHub OAuth2 token saved in /etc/github/oauth.
- GitHub API is used to make sure we're not missing events due to a bug in GHA.

CLOUD NATIVE
COMPUTING FOUNDATION

DevStats

DevStats usage:

- It uses standard tools for compilation: make, make install, see compilation for details.
- It uses standard tools for testing: make test, make dbtest etc, see testing.
- It uses Travis CI for continuous integration.
- Continuous deploy: runs on successful Travis CI build from a webhook, see: continuous deployment.
- For local development see this.
- There are many configuration options (via the environment variables).
- Some historical benchmarks are here.

CLOUD NATIVE
COMPUTING FOUNDATION

DevStats

DevStats - GDPR

- Some people want their data to be hidden
- It is possible in devstats by following this file. It will replace given login/name/email with anon-# strings.
- cncf/gitdm (used for actors company affiliation) also supports GDPR, see this file.

DevStats - bots

- Bots activity is skipped in DevStats metrics. See this file for more information.

DevStats

- More general information about annotations, bots, periods, repository groups, aliases, tags, variables can be found here.
- Databases tables are described in details here. The database structure is described here.
- Each Grafana dashboard has its own documentation (displayed on the dashboard). See here for details - kubernetes and shared dashboards. Some special variables are used in the shared documentation, like for example '[[full_name]]' is replaced with the current project name.
- There are multiple periods definitions for many dashboards. There can be single periods like an hour, week etc. Averaged periods like 24 hours MA (moving average), 7 days MA. Last x periods (like last week, month, 10 days). Inter annotation periods like version 1 - version 2 or version 3 - now etc. See periods.

CLOUD NATIVE COMPUTING FOUNDATION

DevStats

DevStats - localization-related dashboards

- Some dashboards (timezone stats, countries stats) use actors locations.
- This uses an actor's provided GitHub "location" value which can be anything (this is a developer input)
- We're using geonames.org database to have about 16M+ location names and then use a smart algorithm to guess users real location. All we need is a country_id (2 letters like us, pl, de, ru etc.) and timezone name (like 'America/Los_Angeles or Europe/Warsaw). Timezone name also gives us timezone UTC offset.
- With this data, we can create timezone offset related dashboard to see the projects actor's diversity and countries dashboard to see (aggregated and/or cumulated) stats per country.