This document follows a discussion that happened on the prometheus-dev mailing list regarding the current CI infrastructure.

Current situation

The Prometheus organization uses currently 2 different systems for CI: Circle CI and Travis CI. Some repositories use none (docs for instance), others use only one and the rest uses both.

The main drawback of having 2 systems in place is the increased maintenance burden and the cognitive load. In the last months we had to deal with:

- Circle CI configuration migration from 1.0 to 2.0.
- GitHub Services being deprecated and replaced by GitHub Apps which Travis CI doesn't fully support yet.

Historically Circle CI had been introduced because Travis CI wasn't able to push images to Docker. This isn't the case anymore which is why it may be the right time to revisit the situation.

Travis CI

Used for:

- Running unit tests. It supports testing with multiple Go versions (used by client_golang) in parallel.

Not used by:

- node_exporter
- busybox
- golang-builder
- memcached_exporter

Current plan:

- Free
- 2 vCPUs / 4GB RAM (see https://docs.travis-ci.com/user/reference/overview/#virtualisation-environment-vs-operating-system)

We get up to 2 vCPUs and 8GB or RAM at no additional cost.

Known limitations:

- ~~No support for Windows.~~ Windows support is available for early access since the end of September 2018, see https://travis-ci.community/t/windows-early-release/195.
- No support for storing build artifacts on forked PRs unlike CircleCI. One can upload build artifacts to S3 for instance but for security reasons, S3 credentials (or any other type of credentials) aren't accessible to forked PRs.

Circle CI

Used for:

- Running unit tests.
- Build tarball releases for multiple architectures.
- Publish releases.
- Publish container images on docker.io and quay.io (both for master and tagged releases).

Not used by:

- client_golang
- client_java
- client_ruby
- client_python
- cloudwatch_exporter
- common
- tsdb
- prometheus_api_client_ruby

Current plan:

- free
- 15 containers
- 2 vCPUs / 4GB RAM (see https://circleci.com/docs/2.0/configuration-reference/#resource_class)

With a paid plan, we could get up to 8 vCPUs and 16GB of RAM.

Known limitations:

- It can't run the tests on Prometheus because it exhausts the resources (OOM).
- No support for Windows.

- Cannot trigger a build for an open PR with custom parameters (use case: push a container image before the PR is merged for testing). Note that it wouldn't be recommended as it could potentially leak encrypted/hidden data (such as credentials) to external contributors (see Travis documentation but the same applies to CircleCI).

BuildKite

Used for:

- node_exporter
- Non-x86 and Non-Linux native testing.

Current plan:

- Free Packet.net bare-metal server.
- Running on ansible-built VMs.

Known limitations:

What other projects use?

Most of the CNCF projects use Travis CI minus Kubernetes (self hosted) and Envoy (https://github.com/envoyproxy/envoy/pull/2224, using CircleCI with xlarge resources with paid plan from CNCF).

Notable users of CircleCI are Envoy, Grafana (they ditched Travis too) and Istio.

Alternatives

Statu-quo

Pros:

- Not tied to a single platform. In case one of them has an outage, goes out of business or changes its plan, there's a backup solution.

Cons:

- Maintenance burden.

Use Travis CI only

Pros:

- No showstopper identified. It seems to support what we have currently for releasing binaries and images.

Cons:

- Not clear whether the hardware specs can be increased, even with a paid plan.
- It would require to update and test all the projects with the release jobs.

Use Circle CI only

Pros:

- Less work than to replace it with Travis CI.

Cons:

- Resource limitations would probably require a paid plan (which can probably be sponsored by the CNCF).

Select one CI on per-project basis

Projects that are only using Travis CI will continue to do so. Projects using both systems will switch to CircleCI only.

Pros:

- Minimal efforts.

Cons:

- Both systems still have to be supported but on a smaller surface.

CNCF CI

There is a group in the CNCF org working on CI. Currently it builds the latest stable and master release of Prometheus (but not AlertManager, node_exporter and other which are taken from Docker Hub) then deploys everything to various cloud providers on top of Kubernetes. The whole process runs on a daily basis.

See also https://cncf.ci/

OpenLab

OpenLab is an initiative originating from the OpenStack community which provides CI services for various projects more or less related to the OpenStack ecosystem (for instance gophercloud, the Go client library for OpenStack and cloud-provider-openstack). It is available for any open

source project that wants to run unit and/or integration tests. Compute resources are offered by cloud providers and shared across projects.

Technically the solution is based on Zuul + Ansible playbooks.

It could be interesting to use it for testing the OpenStack SD mechanism: OpenLab provides ready-to-use playbooks that can deploy an OpenStack cloud in a box.

Roll our own CI system

There have been discussions about having self-hosted CI in the context of the Prometheus benchmarks.

Pros:

- We control everything.

Cons:

- Need to build and maintain, this isn't sustainable for the team.

Questions

- Support for more hardware platforms?
    - Travis CI and CircleCI support Linux and OSX. Travis supports Windows too (as of Oct 2018).
        - https://circleci.com/docs/2.0/testing-ios/
        - https://docs.travis-ci.com/user/multi-os/
- Take into account the future prometheus-community effort?
- Crossbuilds on CircleCI are slow but this is more a limitation of promu than CircleCI itself.
- Multi-arch container images has been asked too (https://github.com/prometheus/promu/issues/89). Is any solution more suited at this?