Monorepo vs multirepo

Summary

Issue

Our project involves developing three major categories of software:

- Front-end GUIs
- Middleware services
- Back-end servers

When we develop, our source code management (SCM) version control system (VCS) is git.

We need to choose how we use git to organize our code.

The top-level choice is to organize as a "monorepo" or "polyrepo" or "hybrid":

- Monorepo means we put all pieces into one big repo
- Polyrepo means we put each piece in its own repo
- Hybrid means some mix of monorepo and polyrepo

For more please see https://github.com/joelparkerhenderson/monorepo-vs-polyrepo

Decision

Monorepo when an organization/team/project is relatively small, and rapid iteration is higher priority than sustaining stability.

Polyrepo when an organization/team/project is relatively large, and sustaining stability is higher priority than rapid iteration.

Status

Decided. Open to revisiting if/when new tooling becomes available for managing monorepos and/or polyrepos.

Details

Assumptions

All the code that we're developing is for one organization's offerings, and not for the general public. I.e. the Broker-Dealer isn't aiming to have anything like general public volunteer developers.

Constraints

Constraints are well-documented at https://github.com/joelparkerhenderson/monorepo-vs-polyrepo

Positions

We considered monorepos in the style of Google, Facebook, etc. We think any monorepo scaling issues are so far in the future that we will be able to leverage the same practices as Google and Facebook, by the time we need them.

We considered polyrepos in the style of typical Git open source projects, such as Google Android, Facebook React, etc. We think these are the best choice for general public participation (e.g. anyone in the world can work on the code) and individual availability (e.g. the project is used on its own, without any other pieces).

Argument

When an organization/team/project is relatively small, we choose monorepo, because rapid iteration is significantly higher in priority than sustaining stability

When an organization/team/project is relatively large, we choose polyrepo, because sustaining stability is significantly higher in priority than rapid iteration.

Implications

If there's an existing pipeline for CI+CD, then we may need to adjust it for testing multiple projects within one repo.

CI+CD could take more time for a full build for a monorepo, because CI+CD could build all the projects in the monorepo.

If an organization/team/project grows, then a monorepo will have scaling issues.

Monorepo scaling issues may make it increasing valuable to transition to a polyrepo.

Transition from monorepo to polyrepo is a signficant devops task, and will need to be planned, managed, and programmed.

Related

Related decisions

We will create decisions for related tooling to manage monorepos (e.g. Google Bazel) and polyrepos (e.g. Lyft Refactorator).

Related requirements

We need to develop the CI+CD pipeline to work well with git.

Related artifacts

We expect the repo organization to have related artifacts for provisioning, configuration management, testing, and similar devops areas.

Related principles

Easily reversable. If the monorepo doesn't work in practice, or isn't wanted by leadership, it's simple to change to polyrepo.

Customer Obsession. We value getting the project in the hands of customers, and we believe that a monoreop can get us there faster than a polyrepo, and also help us iterate faster.

Think big. Google and Facebook are very strong advocates for monorepos over polyrepos, because all the core offerings can be developed/tested/deployed in concert.

Notes

Add any notes here.