# Monorepos

## Working with a single, big, scary version control repository

Benjamin Eberlei

October 13, 2015

# Outline

- Definition and History
- Reasons to Use Monorepos
- Reasons **not** to Use Monorepos
- Tools for Monorepos

# What is a Monolithic Repository (monorepo)?

A **single** version control repository containing multiple

- projects
- applications
- libraries,
- using a common build system.

# Monolithic Applications?

Code in monolithic repositories **must not** be monolithic itself

# Goal of this Talk

Challenge Your Believes and Practices

# History of Version Control

Monorepos where widespread when we used Subversion

# And Then came Git...

- We unlearned monorepos
- NPM and Heroku force deployable-unit=one-repository pattern
- Many package managers followed suite (Composer..)
- Lead to the ”Microlibrary” movement

# Example

Apache Project has code from **all** projects in one SVN repository.

# Example

Linux Kernel contains support for tons of different drivers in one central repository.

# Who is doing this? Google

Google use monorepo for their whole codebase (custom VCS)

# Who is doing this? Facebook

Facebook merge their three monorepos into one bigger monorepos

# Who is doing this? Twitter

Twitter migrated from repository/library to monorepo

# Who is doing this? We are!

https://tideways.io

# But...

**Monorepos** (qafoo.com)

35 points by luu 45 days ago | past | web | 42 comments

bsimpson 45 days ago

Everyone I've ever met who's worked at Twitter *hates* the monorepo.

They have a single git repo with *way* more files than that tool was ever designed to maintain. As such, normally instantaneous commands like status, log, ls, etc. take an inordinately long amount of time to execute. Apparently their onboarding process is "have someone copy his working copy to a flash drive, then copy it from there to your laptop" because git pull would take too long.

# But...

neilk 44 days ago

This has to be a sick joke. The monorepo at Google was the bane of my existence, and everyone else that I knew.

# Reasons to use Monorepos

# High Discoverability For Developers

- Developers can read and explore the whole codebase
- `grep`, IDEs and other tools can search the whole codebase
- IDEs can offer auto-completion for the whole codebase
- Code Browsers can links between all artifacts in the codebase

# Code-Reuse is cheap

Near zero cost in introducing a new library

- Extract library code into a new directory/component
- Use library in other components
- Profit!

# Refactorings in one commit

Allow large scale refactorings with one single, atomic, history-preserving commit

- Extract Library/Component
- Rename Functions/Methods/Components
- Housekeeping (phpcs-fixer, Namespacing, ...)

# Pull Requests in one commit

- Simplify changes that affects multiple layers and components
- Allows to revert them

# No Dependency/Version Mess

- No need for versioning in components
- No need to manage dependencies and versions
- Avoids multi-pull-request merging in required order
- Avoids forgetting to update composer/git submodule
- Forces dependees to update to newest version

# Efficient Testing and Continuous Integration/Delivery

- Single repository makes dependency resolving easy
- Only run the tests for components that changed
- But also run the tests for all dependencies, recursively
- Avoid manual test micromanagement

# Increased Developer Productivity/Onboarding

- New developers can get started more easily
- All developers can move between different projects
- Avoids information silos

# Reasons not to use Monorepos

# Require Much Stronger Collective Responsibility

# Require Trunk-Based Development

Force you to have only one version of everything

Mess up projects with different development cycles

# Scalability Requirements for the Repository

# Tools for Monorepos

# We need to discuss

- Project Structure
- Dependencies
- Version Control
- Builds
- CI Systems

# Example Project Structure

```
components/chrome-extension
components/landingpage
components/php-extension
components/php-library
components/profiler
components/profiler-ui
components/replicator
golang/src/tideways/cli
golang/src/tideways/collector
golang/src/tideways/daemon
golang/src/tideways/sql
golang/src/tideways/xhprof
infrastructure/automation
playground/*
tools/db-deploy
```

Qafoo
passion for software quality

# Components

- A component is a library, application, ...
- Organize components in useful directory hierachy
- A single component **can** be monolithic
- ..or a ”micro-service/library”
- Components have dependencies to each other
  - by directly sharing code
  - by calling external APIs (REST, RPC)

# Multiple composer.json and monorepos

- Cause vendor folder duplications (slow)
- Autoloading must be tricked into loading other monorepo dependencies.
- Different dependencies for each component cause crashes

# Relative Autoloading

```
# components/Foo/composer.json
{
    "autoload": {
        "psr-0": {"Foo": "src/"}
    }
}
```

```
# components/Bar/composer.json
{
    "autoload": {
        "psr-0": {
            "Foo": "../Foo/src/"
            "Bar": "src/"
        }
    }
}
```

# Dependency Clash

```
# components/Foo/composer.json
{
    "require": "symfony/http-foundation": "~3.0"
}
```

```
# components/Bar/composer.json
{
    "require": "symfony/http-foundation": "2.4.*"
}
```

# Dependency Clash: Use both Foo and Bar?

```
# components/Baz/composer.json
{
    "autoload": {
        "psr-0": {
            "Foo": "../Foo/src",
            "Bar": "../Bar/src"
        }
    },
    "require": {
        "symfony/http-foundation": "~3.0"
    }
}
```

# One composer.json and monorepo

- Use the same `vendor/autoload.php` in all components
- Or use Fiddler to build custom autoloaders for each component

# Fiddler

- monorepo support on top of Composer
- `https://github.com/beberlei/fiddler`

# Fiddler Concepts

- Fiddler Package Names are directory names
- One global composer.json
- One version of every dependency onl
- Each component with a fiddler.json

# Fiddler: Global composer.json

```
1  # ./composer.json
2  {
3      "require": {
4          "symfony/http-foundation": "~3.0",
5          "doctrine/dbal": "~2.5"
6      }
7  }
```

# Fiddler: Component with Vendor Dependency

```
# components/Foo/fiddler.json
{
    "autoload": {
        "psr-0": {"Foo": "src"}
    },
    "require": [
        "vendor/symfony/http-foundation"
    ]
}
```

# Fiddler: Component

```
# components/Bar/fiddler.json
{
    "autoload": {
        "psr-0": {"Bar": "src"}
    },
    "require": [
        "components/Foo",
        "vendor/doctrine/dbal"
    ]
}
```

# A Common Build System

High level build-system that standardizes continouos integration stages

- Bazel/Blaze by Google
- Buck by Facebook
- Pants by Twitter

We use Ant


# THANK YOU

Rent a quality expert
qafoo.com