



**OST**

Eastern Switzerland  
University of Applied Sciences

# **Distributed Systems (DSy)**

## **Monorepos vs. Polyrepos (Multirepo)**

Thomas Bocek

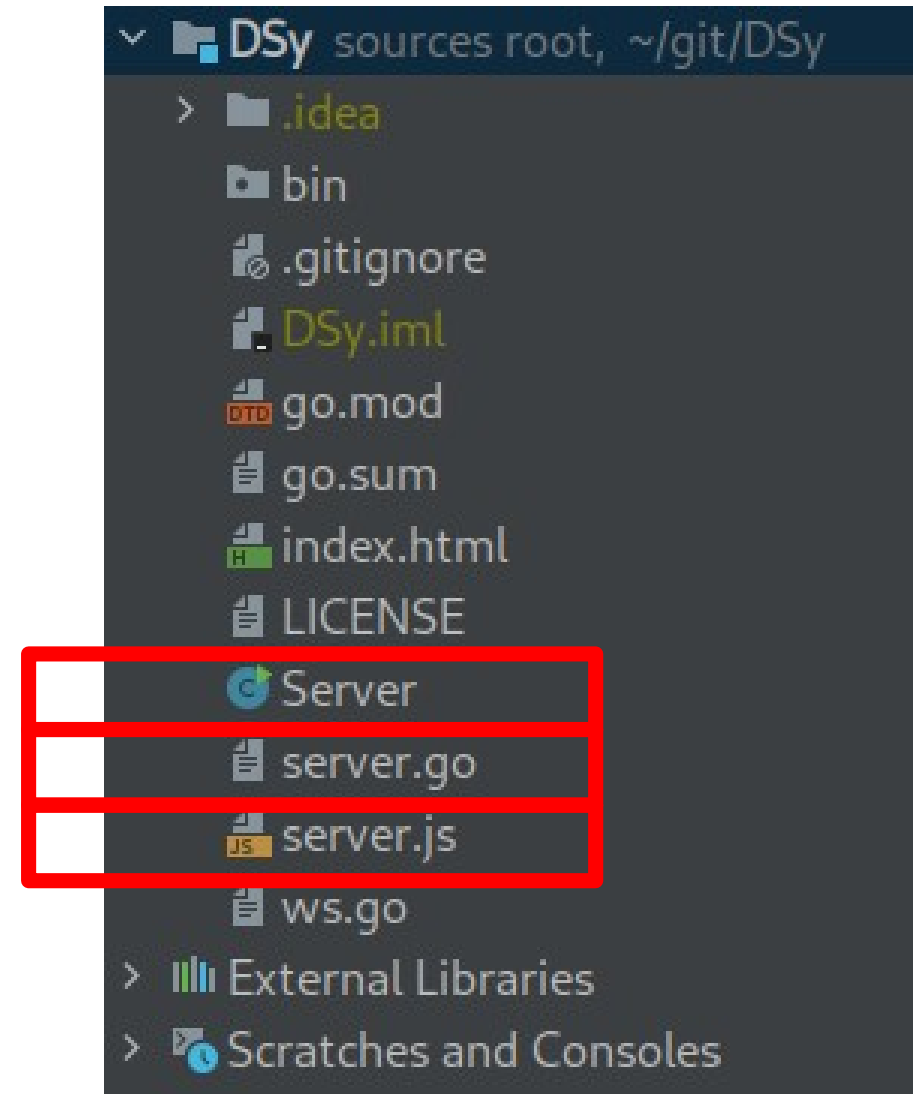
27.02.2023

# Learning Goals

- Lecture 3 (Repositories)
  - What is a monorepo, what is a polyrepo?
  - When to use which type?

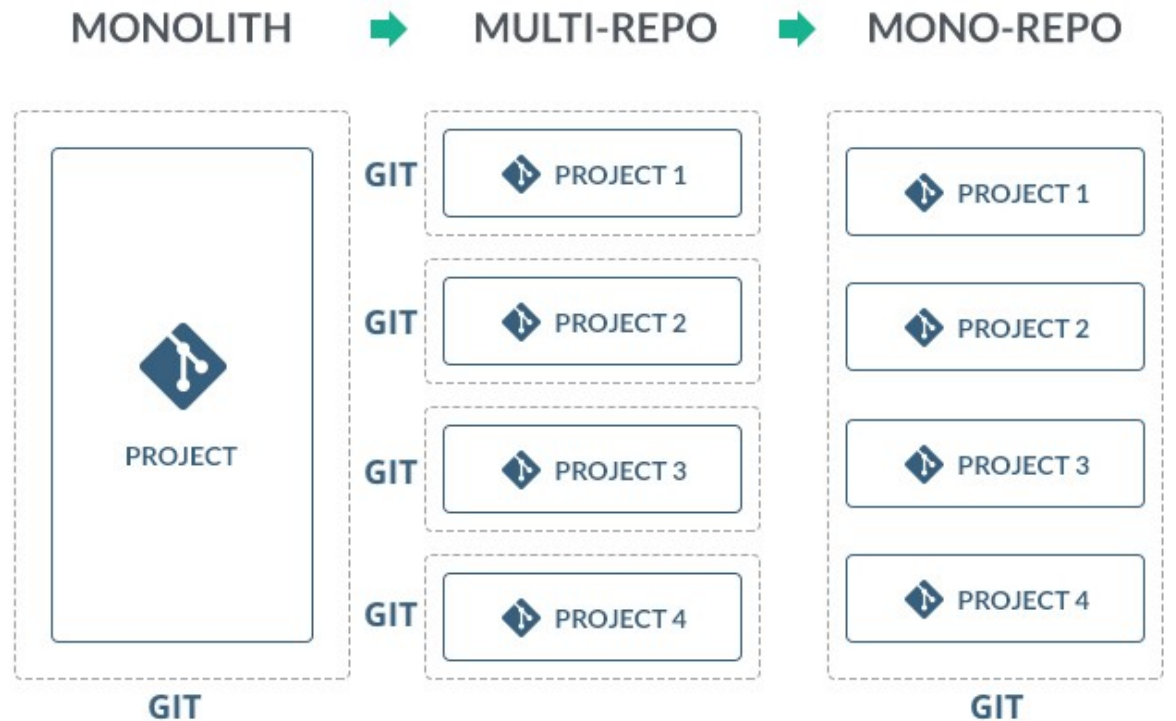
# Project Setup

- Project setup the wrong way:
  - <https://github.com/tbocek/DSy>
  - Everything (Java, Golang, Javascript) flat in one directory – do not do this
- Split up
  - Backend, frontend, service1, etc. in separate repositories
  - ~1 technology per split
    - Most likely you won't have a frontend mix of frontend technologies e.g., Angular with Vue
    - Sometimes you do :)
    - Sometimes you have a script directory, with different languages (bash, javascript)
    - Sometimes you don't :)



# Monorepo

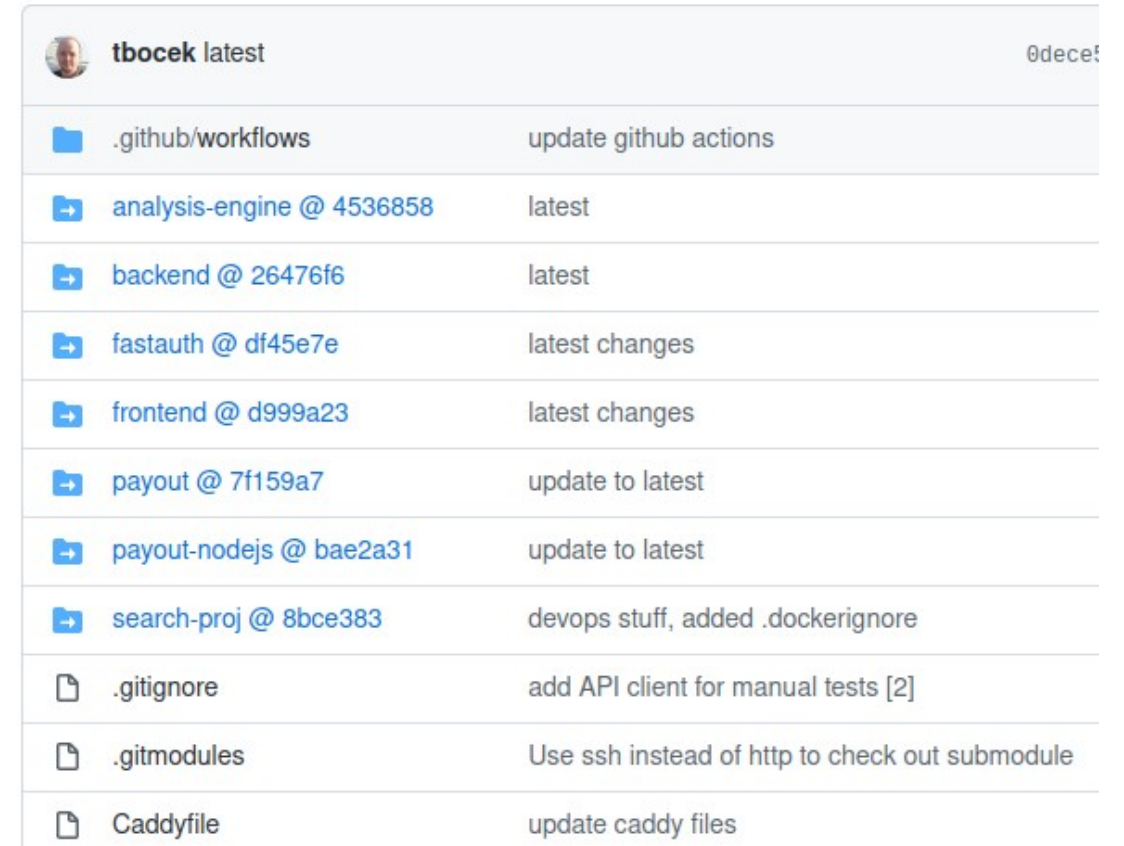
- One repository for all projects
  - 1 sub-directory with frontend, 1 sub-directory with backend, etc.
  - Tools e.g., **lerna** - update dependencies, **hoisting**
- Other names: onerepo or unirepo
- Examples
  - **Simform**
    - Started with monorepo, switched to mulirepo, now with hybrid approach “you can’t blindly follow any approach”
  - Google, Facebook, Twitter
    - Use monorepos (others **do not**)
  - **Flatfeestack**
    - Uses hybrid approach, as not much cross-repository functionality needed (reduce complexity)



<https://codefresh.io/continuous-integration/using-codefresh-with-mono-repos/>

# Polyrepo

- Multiple repositories for a project
  - Frontend in a different repository than the backend
  - Example: <https://github.com/flatfeestack>
    - Wip, not ready to make it public...
    - Frontend: Svelte, npm
    - Backend: Golang
  - Other names: manyrepo or multirepo
- Sync via git **submodules** or via bash script
  - **Submodules**: can also be used as dependency management



File	Commit Message
.github/workflows	update github actions
analysis-engine @ 4536858	latest
backend @ 26476f6	latest
fastauth @ df45e7e	latest changes
frontend @ d999a23	latest changes
payout @ 7f159a7	update to latest
payout-nodejs @ bae2a31	update to latest
search-proj @ 8bce383	devops stuff, added .dockerignore
.gitignore	add API client for manual tests [2]
.gitmodules	Use ssh instead of http to check out submodule
Caddyfile	update caddy files



# Pro/Cons - Opinion

- **Monorepo**

- Tight coupling of projects
  - E.g., generating openapi.yml from backend, generate types for frontend → simply copy
- Everyone sees all code / commits
- Encourages code sharing within organization
- Scaling: large repos, specialized tooling

- Opinion: **Accenture** - “From my experience, for a smaller team, starting with mono-repo is always safe and easy to start. Large and distributed teams would benefit more from poly-repo”
- My opinion: for small teams and “independent” project, use polyrepo. (I worked with small teams with mono and polyrepo, I have worked in big projects with polyrepos, but never in a big project with monorepos). If you have a tight coupling between projects (OpenAPI), use monorepos.
- Other opinion (sales pitch): <https://monorepo.tools>

- **Polyrepo**

- Loose coupling of projects
  - If you want to generate openapi.yml, you need access from the backend repository to the frontend (e.g., curl+token)
- Fine grained access control
- Encourages code sharing across organizations
- Scaling: many projects, special coordination