

Principles of Software Design

GIT and some other stuff

Robert Lukočka

lukotka@dcs.fmph.uniba.sk

www.dcs.fmph.uniba.sk/~lukotka

M-255

Software configuration management

- Software configuration management is the task of tracking and controlling changes in the software. It includes tracking changes in source code, documentation, and other artefacts.
- Primarily done using Version control systems (VCS).
- Some other systems can be useful in this context (e.g. Issue tracking systems)

What a larger project needs?

- All artefacts are have a defined place.
- More versions of the same artefacts.
- Multiple people working on the same artefacts concurrently.
- Storing historical versions of the artefacts.
- ...

What to track?

We should track exactly what is necessary to build, run, and work on the project.

- Manually written source files.

What to track?

We should track exactly what is necessary to build, run, and work on the project.

- Manually written source files. Yes

What to track?

We should track exactly what is necessary to build, run, and work on the project.

- Manually written source files. Yes
- Generated source files.

What to track?

We should track exactly what is necessary to build, run, and work on the project.

- Manually written source files. Yes
- Generated source files. No - but we need to save the artefacts needed to generate the source.

What to track?

We should track exactly what is necessary to build, run, and work on the project.

- Manually written source files. Yes
- Generated source files. No - but we need to save the artefacts needed to generate the source.
- Final binary.

What to track?

We should track exactly what is necessary to build, run, and work on the project.

- Manually written source files. Yes
- Generated source files. No - but we need to save the artefacts needed to generate the source.
- Final binary. No.

What to track?

We should track exactly what is necessary to build, run, and work on the project.

- Manually written source files. Yes
- Generated source files. No - but we need to save the artefacts needed to generate the source.
- Final binary. No.
- Images.

What to track?

We should track exactly what is necessary to build, run, and work on the project.

- Manually written source files. Yes
- Generated source files. No - but we need to save the artefacts needed to generate the source.
- Final binary. No.
- Images. Yes.

What to track?

We should track exactly what is necessary to build, run, and work on the project.

- Manually written source files. Yes
- Generated source files. No - but we need to save the artefacts needed to generate the source.
- Final binary. No.
- Images. Yes.
- Requirements.

What to track?

We should track exactly what is necessary to build, run, and work on the project.

- Manually written source files. Yes
- Generated source files. No - but we need to save the artefacts needed to generate the source.
- Final binary. No.
- Images. Yes.
- Requirements. Yes.

What to track?

We should track exactly what is necessary to build, run, and work on the project.

- Manually written source files. Yes
- Generated source files. No - but we need to save the artefacts needed to generate the source.
- Final binary. No.
- Images. Yes.
- Requirements. Yes.
- Deployment scripts.

What to track?

We should track exactly what is necessary to build, run, and work on the project.

- Manually written source files. Yes
- Generated source files. No - but we need to save the artefacts needed to generate the source.
- Final binary. No.
- Images. Yes.
- Requirements. Yes.
- Deployment scripts. Definitely.

What to track?

We should track exactly what is necessary to build, run, and work on the project.

- Manually written source files. Yes
- Generated source files. No - but we need to save the artefacts needed to generate the source.
- Final binary. No.
- Images. Yes.
- Requirements. Yes.
- Deployment scripts. Definitely.
- Compiler.

What to track?

We should track exactly what is necessary to build, run, and work on the project.

- Manually written source files. Yes
- Generated source files. No - but we need to save the artefacts needed to generate the source.
- Final binary. No.
- Images. Yes.
- Requirements. Yes.
- Deployment scripts. Definitely.
- Compiler. Well, maybe ...

Why you need to store different versions of your software

- You need to fix errors in older “versions” of your product still in use.
- Different deployment targets (OS)/
- Each historical “version” is its own state.
 - Useful e.g. if you need to track a newly introduced bug.
- Development “versions” of the software.
- ...

What is a version?

Commit:

- Creates a new version of the system
- Unit of change in VCS
- Each commit should make sense on its own.
- A single commit should not be easy to divide to more commits..
- After a commit the project should remain in a sound state (what sound means varies, e.g. development branch vs mainline branch).

What is a version?

Branch:

- Separate copies of the system.
- A commit affects only one branch.
- Branches can be created and merged with other branches.
- There are various reasons to have slightly different copies of the system (development, major releases, experimental).

What is GIT?

- Distributed version control
- Created for the development of Linux kernel
L. Torvalds: I'm an egotistical bastard, and I name all my projects after myself. First 'Linux', now 'git'.
- GitHub - web based version control repository and Internet hosting service – **do not confuse it with git**. Alternatives include GitLab, BitBucket, SourceForge, . . . ,
- GIT is just one particular VCS, there are alternatives too, e.g. CVS, SVN, . . . Some of the above services support other VCS than git.
- Version control services have many other features to manage projects unrelated to git.

Distributed version control [1]

- Clients instead of just taking the versions they need to work on have full mirror of the central repository.
- There may be more equivalent repositories (a central one needs not to exist, but typically it does).

GIT configuration [2]

There are three main levels of configuration:

- computer level (`--system`)
- user level (`--global`)
- project level

You need to set

- Name
- E-mail address

You want to set

- Your favorite text editor

GIT configuration [2]

- `git config --global user.name "Robert Lukočka"`
- `git config --global user.email lukotka@dcs.fmph.uniba.sk`
- `git config --global core.editor vim`

Creating a local repository [3]

- git init
- git clone

File states [4]

- Untracked
- Unmodified
- Modified
- Staged

Working in local repository [4]

- git status
- git add
- git reset
- git commit (git commit -a)
- git rm/mv
- git diff
- git checkout (-) "file"
- git commit --amend

Viewing commit history [5]

- git log - Zillions of options [6]
- git blame
- git revert [commit] - This does not change the history, just adds a new commit.
- gitk

Branches [7]

- HEAD - points at the current version of the current branch.
- git branch (newname)
- git checkout "branchname"
- git branch -d
- git tag

Branches - merging [8]

- git merge - merges some other branch into current branch, the merged branch still exists.
 - git tries to merge stuff automatically
 - if he does not know what to do, it lets you resolve the conflicts
- Alternative to merging: rebasing, adds several commits as if the parallel work was done sequentially [9]

Remote repository [10]

- 1 git clone
- 2 git pull
- 3 git fetch
- 4 git push
- 5 git push origin -delete "branchname"
- 6 git remote

Very basic work flow

- git pull
- repeat
 - do stuff
 - git add
 - git commit
- git push

Stash

How to push while you have uncommitted changes and you do not want to lose them?

- git stash
- git pull
- git stash pop - may create a conflict that needs to be resolved

If you have committed changes branches will be created and merged.

.gitignore

- Used to determine which files should be untracked.
- It is good idea to track this file.

GIT Hooks [11]

A way to fire off custom scripts when certain important actions occur.

- can be used to block the action
- Client side / sever side
- On commit / on merge / on push / ...
- e.g. push runs automated tests, if they do not succeed, push fails.

Pull request

The contributor requests that the project maintainer pull the source code change, hence the name "pull request". The maintainer has to merge the pull request if the contribution should become part of the source. base. [13]

Workflows [12]

- master
- development
- feature branches

Make

Allows to run various commands

- Compared to shell scripts, it checks prerequisites
- You create a file named “Makefile”. Basic syntax:
goal: dependencies (files or other goals)
 <tab> command
 <tab> command
 <tab> ...
- Further examples

Markdown

A lightweight approach to add formatting to text files.

What else should you know

- SSH, SCP, SFTP, rsync.
- To make a deployment script - shell script.











How to initiate a small project

- Initiate version controlling (e.g. git)
- Set up how the project is compiled and build (e.g. Makefile)
- Deployment script
- Basic documentation template (e.g. Markdown)
- Set coding standards, workflows, how quality will be enforced, how automatic testing integrates the workflow ... (git, makefile, ...)
- Set up reasonable project structure to attain these goals.

Resources I

- Distributed version control
- Getting Started - First-Time Git Setup
- Creating a repository
- Working with local repository
- Viewing commit history
- Branches
- Merging
- Git tutorial
- Hooks
- Example workflows
- GIT hooks
- Makefile tutorial
- An Introduction to Makefiles
- Mastering Markdown

References |

-  Distributed version control
-  Getting Started - First-Time Git Setup
-  Creating a repository
-  Working with local repository
-  Viewing commit history
-  Git log
-  Branches
-  Merging
-  Rebase
-  Remotes

References II



Hooks



Example workflows



Pull request - Wikipedia