Git and GitLab Basics

Be smart. Think open source.

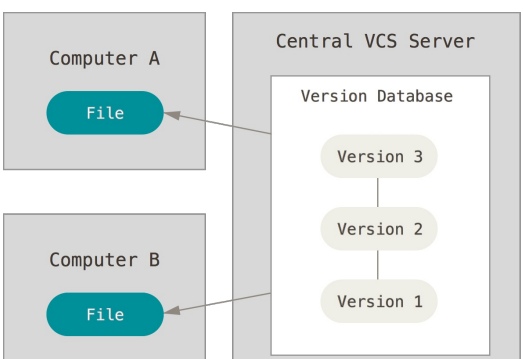
Version Control Systems

- See who changed when why what

Benefits

- Restore
- Archive
- Collaboration

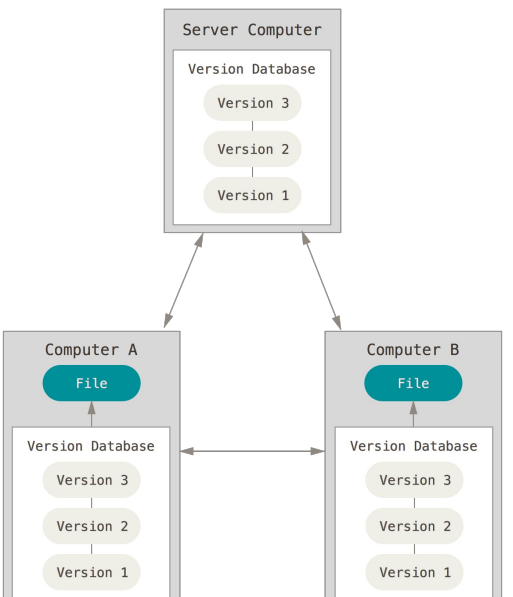
Server - Client



Server - Client

- server-client
- CVS, SVN, ...

Distributed



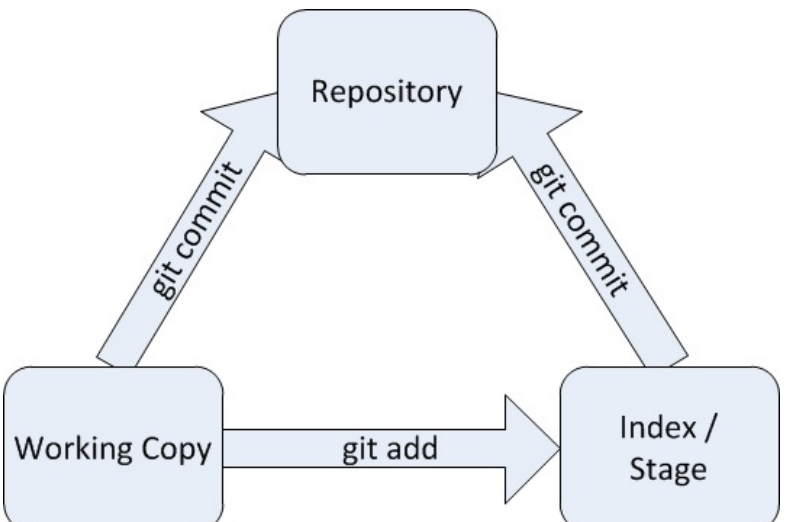
Distributed

- Distributed
- Git, Mercurial, ...

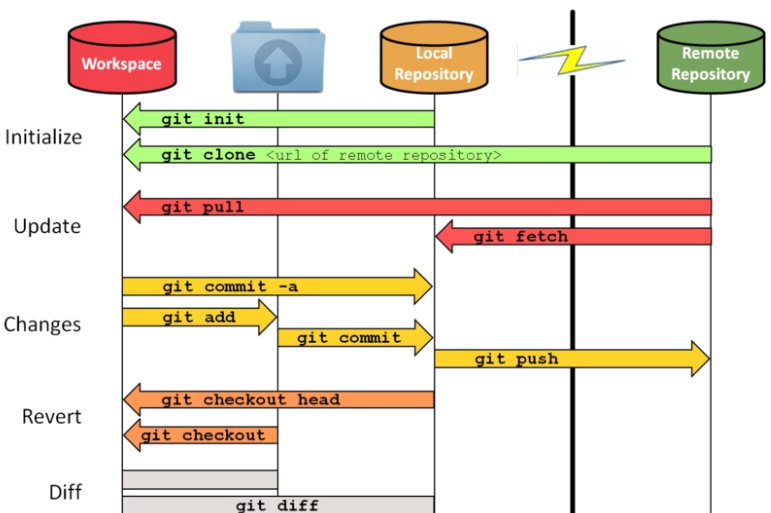
Git

Distributed offline VCS

Working Copy, Stash, Commit



Remote Handling



Log

```
git log --graph --decorate --abbrev-commit --all --pretty=oneline
```

Git Getting started

Configure

```
git config user.name "John Doe"
git config user.email "john.doe@example.org"
```

Create first project

```
git init project1
cd project1
```

Create files and add them

```
touch file1
git add file1  # (not persistent, only signaling)
git commit -m "My first commit"  # (persistent)
```

Git Branches

Create a new branch

```
git branch feature-1                # create a branch
git checkout -b feature-2           # create and switch to branch

touch file2
git add file2                       # stage file for next commit
git commit -m "Adding feature file2"  # commit the changes
```

Merge branch into master

- Fast-forward (default)

```
git checkout master
git merge feature-2
```

- No-fast-forward

```
git checkout master
git merge --no-ff feature-2
```

Hands-on

https://try.github.io/

Preparing

```
# clone a remote repository and switch into it
git clone https://github.com/keachi/octocat.git && cd octocat
git remote show
git remote show origin
```

Checkout

```
git checkout origin/dev -b dev
git checkout master
(create/edit some content)
git status
git checkout -- .  # undo changes
git checkout 5e620bb -B master
```

Create branch

```
# create branch feature-1
git checkout origin/master -B master
git branch feature-1
(or)
git checkout origin/master -B feature-1
```

Show branches

```
git branch           # list local branches
git branch --remotes  # list only remote branches
git branch --all      # list all branches
```

Delete a branch

```
git branch -d feature-1
```

Create a merge

```
# create branch feature-1
git checkout origin/master -B feature-1
(add an exclamation mark to the end of the line in octocat.txt)
git add .
git commit -m 'some comment'
git merge origin/dev  # a merge confict happens
git status
```

Fix merge conflicts

```
git diff
(fix the confict with your favorite editor)
git diff
git add octocat.txt
git merge --continue
```

History

```
git log
git log --oneline --decorate --graph --all
```

Remotes

```
(open a terminal in another empty directory)
$ git init --bare
$ pwd
/home/user/bare
$ (go to your first repository)
$ pwd
/home/user/octocat
$ git remote add bare /home/user/bare
$ git remote show
```

Push and pull

```
git checkout master   # switch from feature-1 to master branch
git branch --remotes  # nothing in repository bare
git push bare master  # push local master to remote bare
git branch --set-upstream-to=bare/master  # track remote branch
git branch --remotes  # branch master is in repository bare too
git fetch bare        # fetch changes from bare
git push bare :master # remove branch master remote
```

Cherry-Picking

```
# Use one single commit and apply it to the local branch
git cherry-pick d17cfc8  # <-- points to a commit on
                         #     https://github.com/keachi/octocat
```

Reset

```
# reset all tracked files and checkout branch master
git reset --hard master
```

Tags

```
# create a tag on the current commit
git tag v1.0.0 -m 'Version 1.0.0'
# create a tag on a specific commit
git tag v0.5.0 -m 'Version 0.5.0' 5e620bb
git push --tags
# list all tags
git tag
```

Cleanup Git repository

```
# cleanup unnecessary files and optimize the local repository
git gc
```

GitLab

Features

- Pages
- Wiki
- Issues
- Merge requests
- Todos
- Milestones

Advanced Features

- CI/CD pipelines
- Issue and MR templates
- Container registry

Users

- Admin permissions
  - Standard user
  - Group owner
  - GitLab admin
- Permissions

Groups

- Project namespace
- User permissions

Project settings

- Project permissions
- Issue template
- Merge Request method
- Approvals

Members

- Grant access by groups
- Add special access to users or groups
- Permissions

Integrations

- Webhooks
- Trigger by different actions
- Services, e.g.
  - Jira
  - Bugzilla
  - Mattermost

Deploy keys

- Special SSH Keys for deploying
- Full repository access
- Used to deploying and in CI/CD
- Access restriction

Protected Branches

- Allow only some users to push to certain branches
- Allow only to merge in certain branches

Runners

- Worker for CI/CD pipelines
- Enforce additional rules for merging
- Run after push on HEAD

Variables

- Key-Value-pair
- Used in CI/CD as environment variables
- e.g.
  - Credentials
  - Run parameters

Triggers

- Trigger a URL for special API calls
- e.g.
  - build documentation
  - deploy website

CI/CD pipelines

- Run on special GitLab worker hosts
- e.g.
  - Compile software
  - Run linting software
- Create artifacts
- Pipelines and jobs
- .gitlab-ci.yml

Project features

- Issues
- Merge requests
- Milestones
- Wiki

Issues

- Issue tracker
- Manage project issues
- Labels
- Milestones
- Assign to someone
- Time tracking
- Due date
- Weight

Merge requests

- Corresponding branch (same or forked repository)
- Reviewing code before merge into master
- Comments
- CI/CD pipeline
- Work in progress (WIP)
- Fix/resolv issue

Administration Interface

- Global Administration
  - Project
  - User
  - Group
- Monitoring

GitLab Update Cycle

- Released frequently
- Update badge in admin area
  - asap: Security issues
  - soon: New features
  - up-to-date: Newest version

Documentation

- Documentation
- Workflow
- GitLab Flow

Thank you!

Be smart. Think open source.

Feel Free to Contact Us

www.adfinis-sygroup.ch

Tech Blog

GitHub

info@adfinis-sygroup.ch

Twitter

Attribution / License

- Git Handling, by ferqwerty
  MIT License,
  https://github.com/fer/dotfiles/wiki/git