GIT DEMO (GitLab)

key concepts and basic commands

Some motivation to use Git

Name

- Super Cool Report v1.xlsx
- Super Cool Report v2.xlsx
- Super Cool Report v3.1.xlsx
- Super Cool Report v3.xlsx
- Super Cool Report v4.xlsx
- Super Cool Report v4a.xlsx
- Super Cool Report v4b.xlsx
- Super Cool Report v5.xlsx
- Super Cool Report vFinal.xlsx
- Super Cool Report vFinal_1.xlsx
- Super Cool Report vFinal_2.xlsx
- Super Cool Report vFinal_Final.xlsx
- Super Cool Report vFinal_Final-UPDATED.xlsx
- Super Cool Report vFinal_Final-UPDATED_NEW.xlsx

```
SuperImportantQuery.sql
SuperImportantQuery-Weekly.sql
SuperImportantQuery-April-Wk1.sql
SuperImportantQuery-April-New.sql
SuperImportantQuery-April-WK3.sql
SIQ2.sql
SIQ3.sql
```

Key Concepts

- git tracks changes, not files !
- allows to work many people on the same project
- the workflow is based on **branches**

[master] 6c6faa5 My first commit - John Doe

[develop] 3e89ec8 Develop a feature - part 1 - John Doe

[develop] e188fa9 Develop a feature - part 2 - John Doe

[master] 665003d Fast bugfix - John Fixer

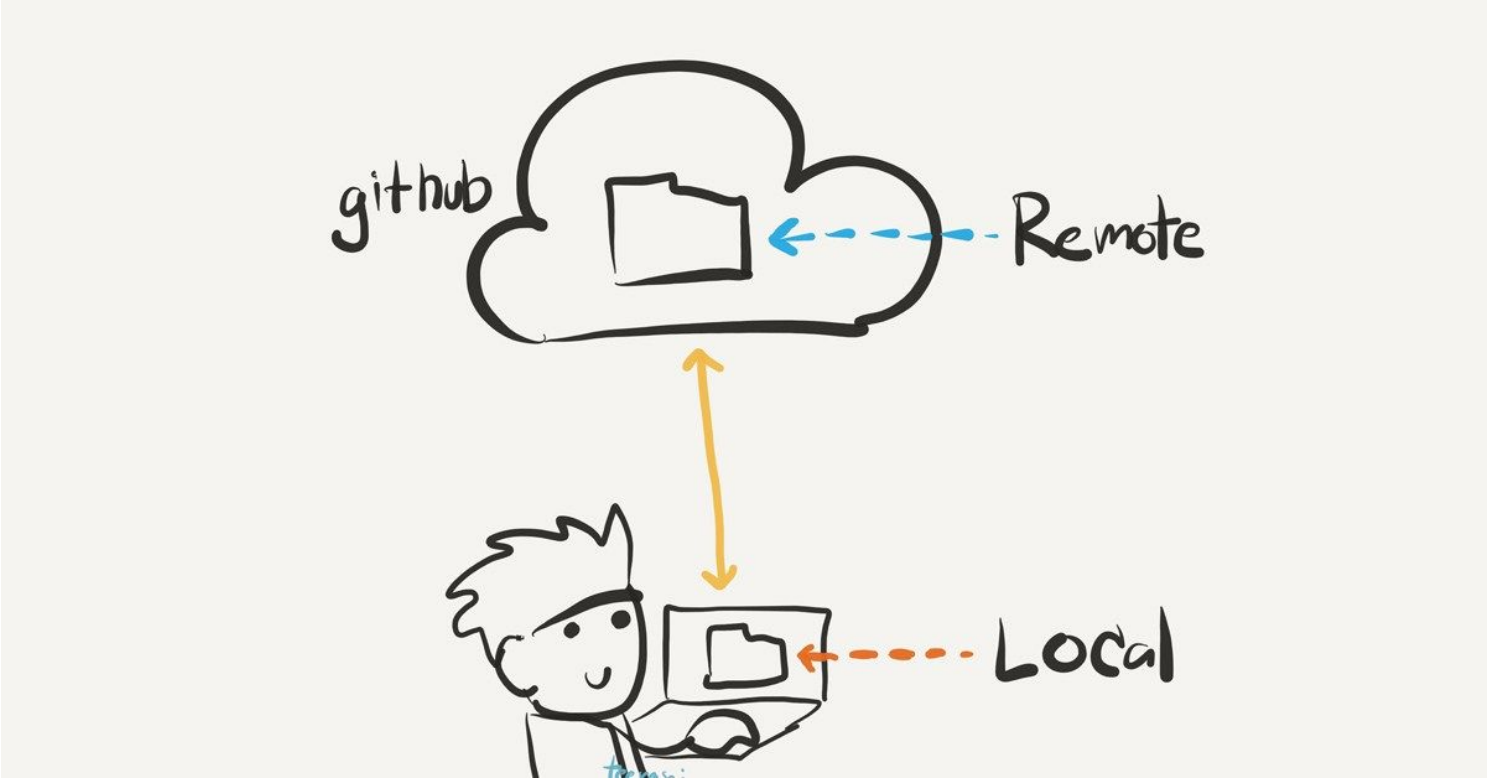
[myfeature] eaf618c New cool feature - John Feature

[master] 8f1e0e7 Merge branch `develop` into `master` - John Doe

[master] 6a3dacc Merge branch `myfeature` into `master` - John Doe

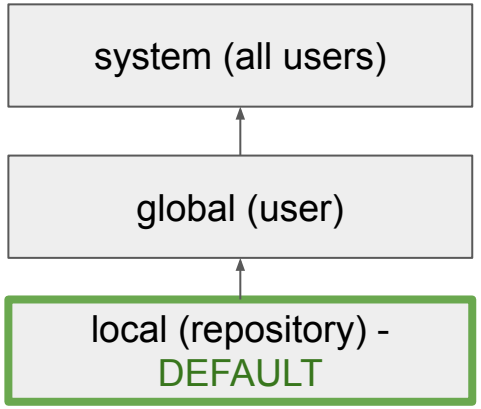
[master] abcdef0 Release of version 0.1 - John Releaser

Key Concepts



Basic Commands - Setting up

git init	initialized empty git repository
git config	(local/global/system - see below)
git config --global user.name <name> git config --global user.email <email>	set global user name set global user email




```
1. System : /etc/gitconfig
2. Global : ~/.gitconfig
3. Local  : .git/config
```

Key Concepts - stages

working directory
(files are untracked)

→

staging area (index) -
git add
(git starts to track the file)

→

git repository (.git) -
git commit
(git adds files to repo)

→

→

Basic Commands - add & status

git status	shows the current status of the project
git add .	add all unstaged files in the current folder to the staging area
git add <filename> (~ git add *.py)	add the changes of file to the staging area
git add -i	interactive adding
git add -p	patch command (hunk by hunk)

Basic Commands - commit

git commit	changes move from the staging area to git repository. Then you will need to enter a message
git commit -m “message” (~ git commit -m “variable a changed”)	commit by writing a message
git commit -a	stage any changes made to files that have been previously committed
git commit -am “message”	stage any changes made to files that have been previously committed + write a message
git commit --amend	change the latest commit (not pushed)

Basic Commands - commit

Each commit has a hash

17 Aug, 2021 2 commits

[dp_RANDD0022IP-8019] Resolve Linter code issues and configure SonarQube. sonar set up
u1093330 authored 2 weeks ago

51a985c2

[dp_RANDD0022IP-8019] Resolve Linter code issues and configure SonarQube....
u1093330 authored 2 weeks ago

6dd7630a

Basic Knowledge - .gitignore

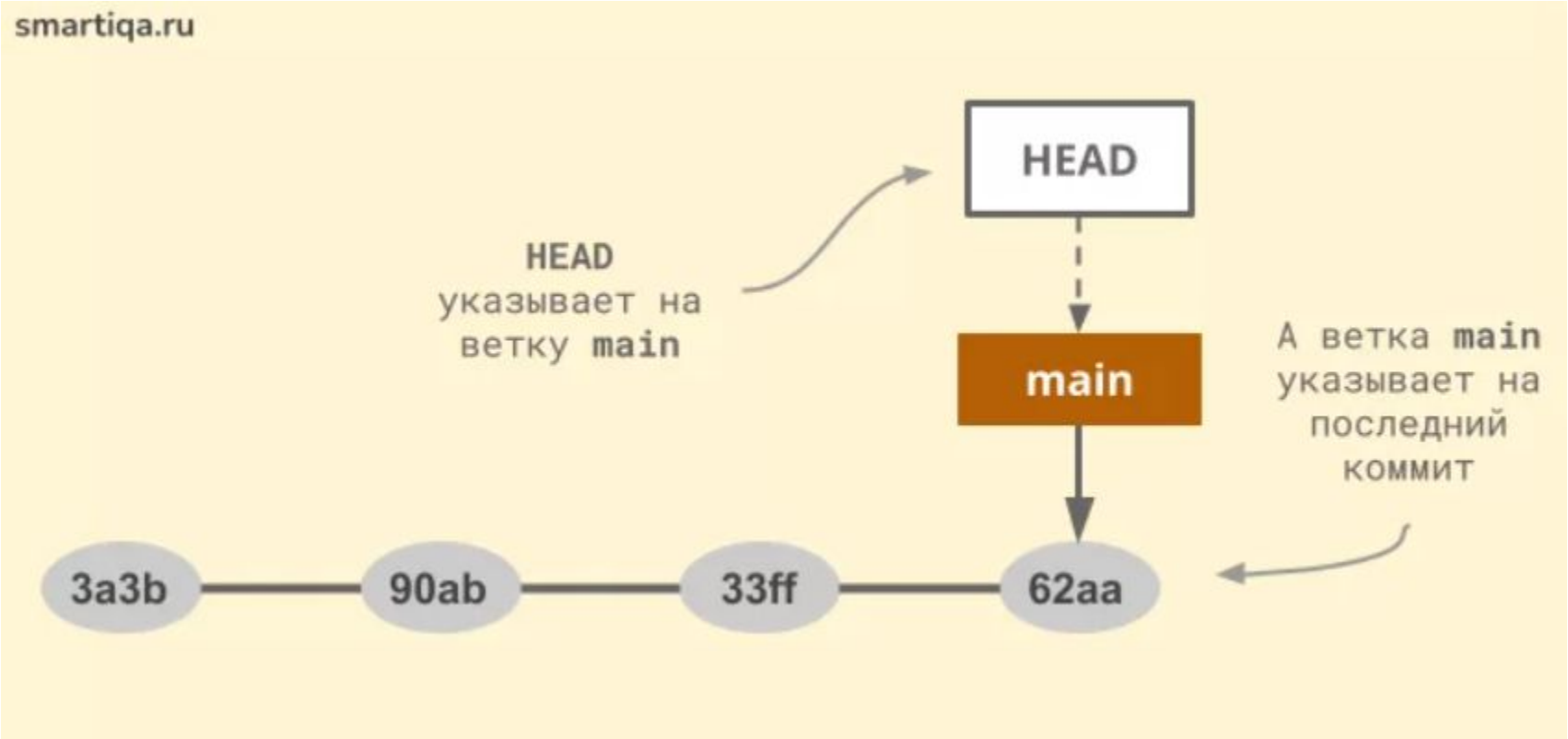
.gitignore

```
# Default ignored files
/.idea/shelf/
/.idea/workspace.xml

# Datasource local storage ignored files
/.idea/dataSources/
dataSources.local.xml

# Editor-based HTTP Client requests
/.idea/httpRequests/
rest-client.private.env.json
http-client.private.env.json
```

Key Concepts - HEAD



Basic Commands - diff

git diff	comparing working directory with staged files
git diff <filename>	comparing working directory with staged file
git diff --staged git diff --cached	difference between the staging area and the latest commit
git diff HEAD	difference between working directory and the latest commit
git diff <hash> <filename>	what was changed in the file since the hashed commit
git diff <hash> <hash>	difference between any 2 hashes

Basic Commands - log

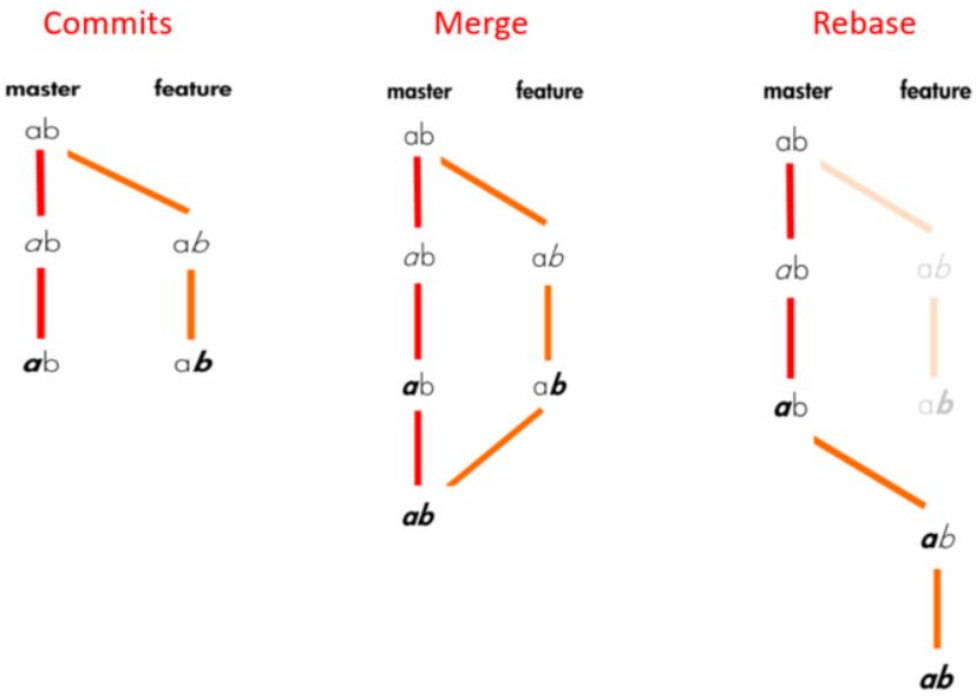
git log	history of commits
git log --stat	history of commits with statistics
git log --oneline	history of commits briefly
git log --graph	graph of commits
git log --oneline --graph git log --oneline --graph --all --decorate	one line graph of commits commits from all the branches
git log --pretty=”%h, %cn”	formatting of logs
gitk gitk --all	opens a separate window with project history

Basic Commands - branch & checkout

git branch	list of branches that we have
git branch <name>	creates local branch name
git checkout <name>	switch to the branch name
git checkout -b <name>	create a new branch and checkout to it
git branch -d <name>	delete branch
git checkout --orphan <name>	create orphan branch
git branch -a	list all branches (including remotes)

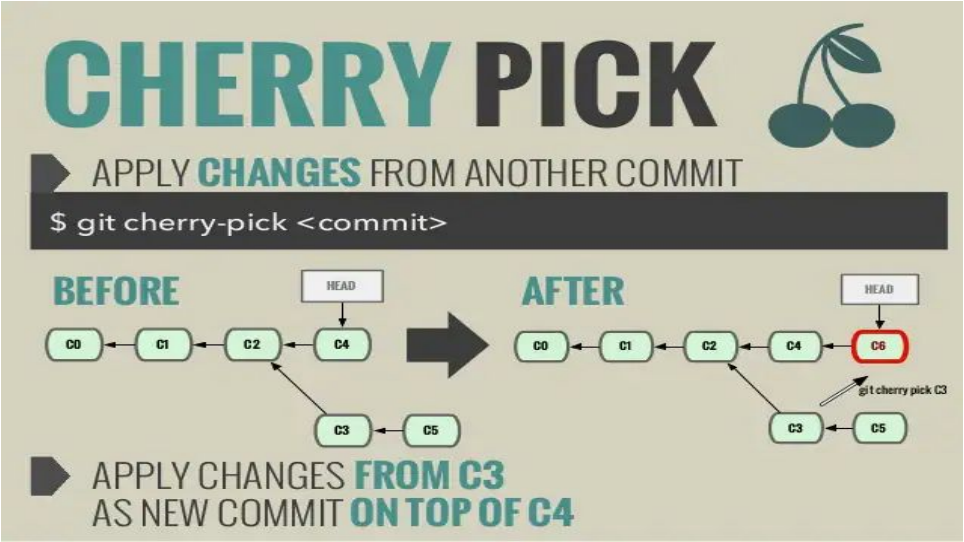
Basic Commands - merge & rebase

git merge <branch-name>	merge branch-name to the current branch
git rebase <branch-name>	rebase project structure, keep it linear and clear



Basic Commands - cherry-pick

git cherry-pick <commit hash>	take the commit and create an exact copy of it in the current branch
git cherry-pick <hash of the first commit> … <hash of the last commit>	take all commits in the sequence and create exact copies of them in the current branch

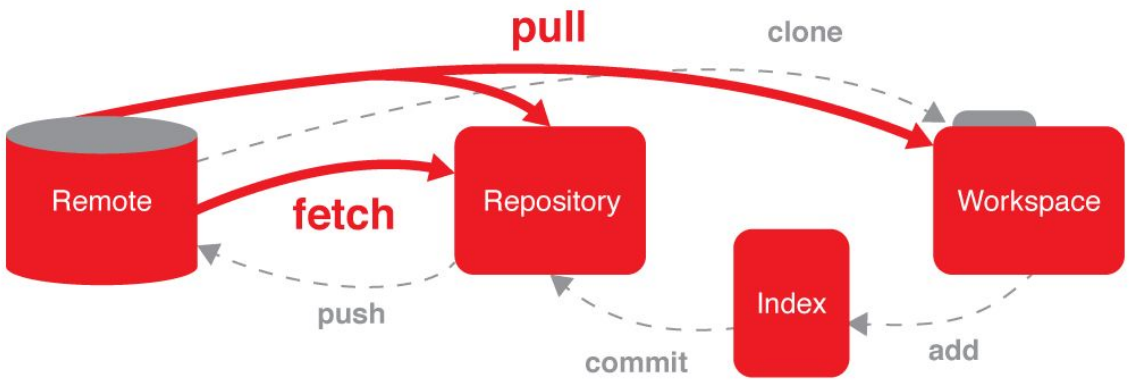


Basic Commands - push, pull & clone

git remote add <name> <link> (~ git remote add origin <link>)	add remote repo (link)
git push origin <branch-name>	push changes from local to remote
git push -u origin <branch-name>	push new branch
git pull	pull changes from remote to local
git clone <url>	download repository

Basic Commands - fetch

git fetch git merge origin/<name> git pull = git fetch + git merge	Download objects and refs from another repository to local repository. However, it does not merge them into the current branch. To merge commits into the main branch, you need to use merge.



Basic Commands - revert & reset

git revert <commit address> (~ git revert HEAD~1)	create a new commit which undoes the changes made in the commit (sequence of commits)
git reset <file>	unstage a file without overwriting changes
git reset --hard <commit address>	move the current branch tip backward to <commit> and reset the staging area and the working directory to match
git reset --soft <commit address>	took the commit tree back in time

Basic Commands - rm & restore

git rm <filename>	remove file from working directory and index
git rm --cached <filename>	unstage the file to untracked
git restore <filename>	restore file **from index**, if there is no such file in index, it will be deleted
git restore --staged <filename>	restore the content in the index **from HEAD**
git restore --staged --worktree <filename>	restore both the working tree and the index **from HEAD** *use --source to restore from a different commit

Basic Commands - stash

git stash	put changes aside
git stash save “message”	put changes aside with message
git stash list	list of all we have stashed
git stash apply <name>	by default applies the latest stash, but it remains in the stash stack after the applying
git stash pop <name>	apply stash name and then drop it
git stash drop <name>	drop stash from stash stack

Basic Commands - alias

git config --global alias.<alias> <command> (git config --global alias.s status)	create alias for git command create alias for status (git s = git status)
alias ga=’git add’	create alias in bash_profile

Sources

- Atlassian git tutorial
- The (written) unwritten guide to pull requests
- Hexlet course
- https://git-scm.com/docs/
- https://smartiqa.ru/courses/git

Safety first.