GIT

Viktoriya Ryazhska

softserve

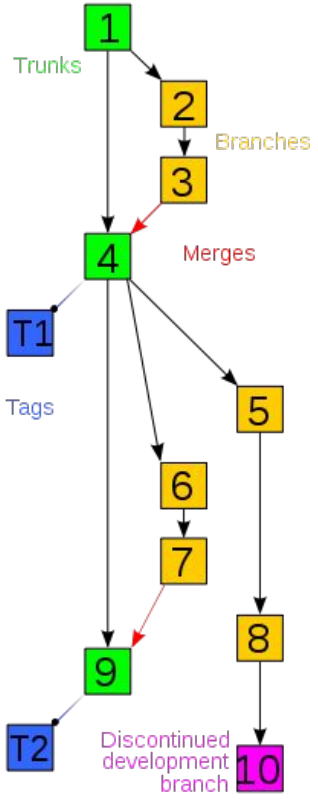
Agenda

- Source Control Management (SCM)
- Types of Version Control Systems
- Git
  - Configuration
  - Basics
  - Work cycle
  - Branches | Merging | Rebasing
- Practical tasks

softserve

SCM

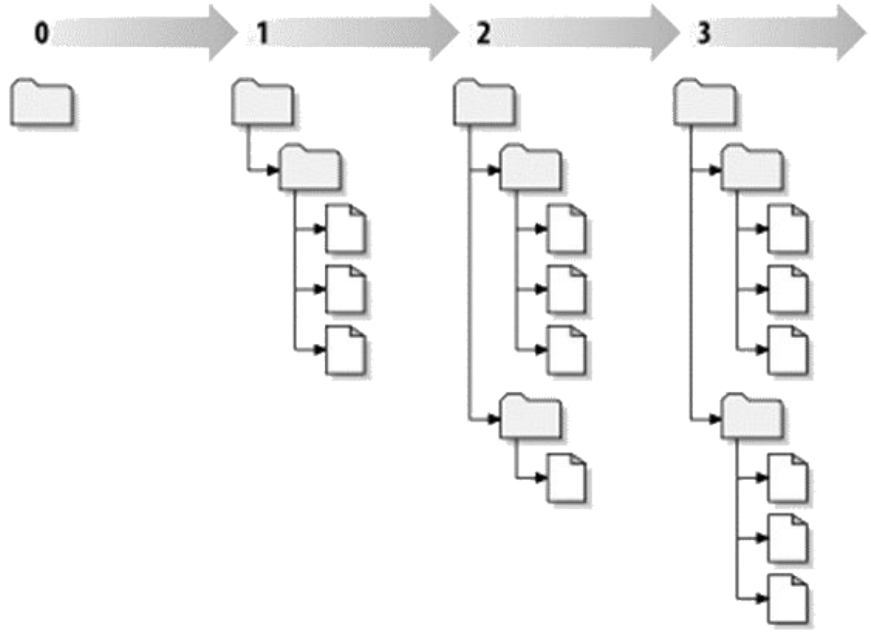
- **Revision control**, also known as **version control** and **source control** (and an aspect of software configuration management), is the management of changes to documents, computer programs, large web sites, and other collections of information.



softserve

Fundamental Concepts of SCM

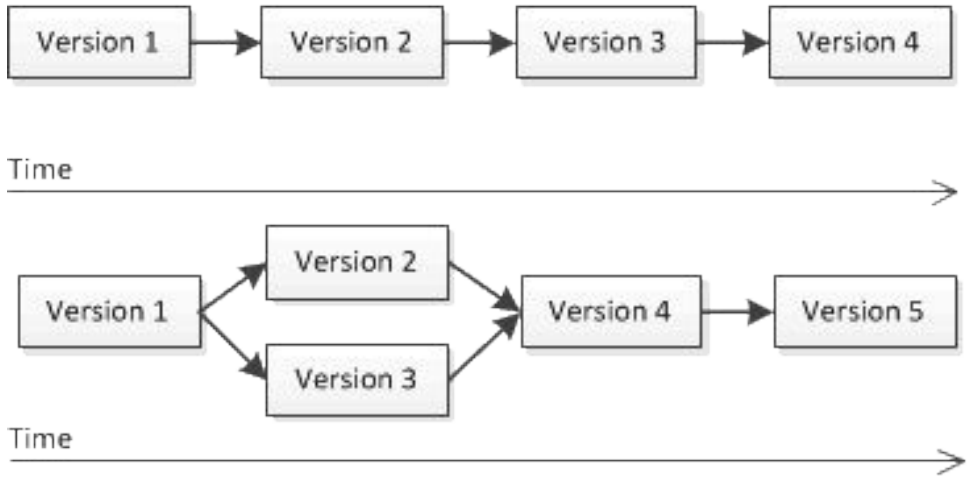
- Tracking changes
- Making updates
- Getting updates
- Conflicts
- Diffing (viewing the differences)
- Branching and merging

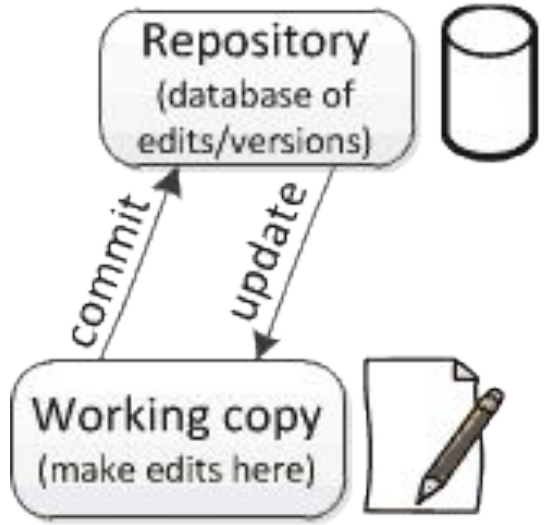


softserve

Terms

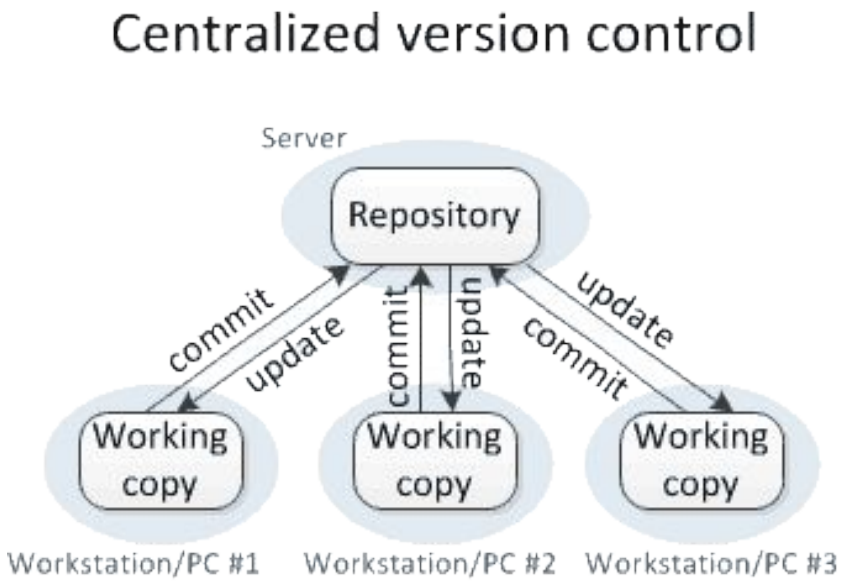
- Repository
- Working Copy
- Merging
- Revision

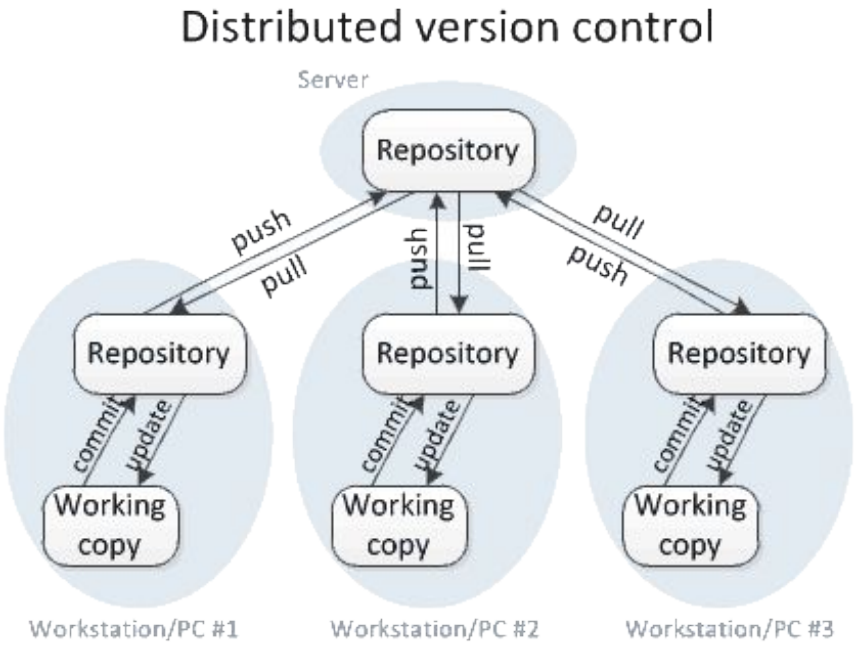




softserve

System version control





- Centralized: CVS, Perforce, **SVN,** Team Foundation Server (**TFS**)
- Distributed: **Git**, Mercurial

softserve

GIT Intro

- **Git** – is a distributed revision control system with an emphasis on speed, data integrity, and support for distributed, non-linear workflows.
- **Git** was initially designed and developed by *Linus Torvalds* for Linux kernel development in 2005, and has since become the most widely adopted version control system for software development.
- Every Git working directory is a **full-fledged repository** with **complete history** and **full revision tracking capabilities**, not dependent on network access or a central server.

softserve

Install git

https://git-scm.com/download/win

Mac OS

Step 1 - Install Homebrew

```
#ruby -e "$(curl -fsSL
https://raw.githubusercontent.com/
Homebrew/install/master/install)"
brew doctor
```

Step 2 - Install git

```
#brew install git
```

Linux OS

Debian Family (Debian, Ubuntu, Mint)

```
#apt-get install git
```

Red Hat Family (RHEL, CentOS, Fedora)

```
#yum install git
```

softserve

Let’s configure git 😊

Git comes with tool called **`git config`**

Identity

```
$ git config --global user.name “Vasia Pupkin“
$ git config --global user.email vpupkin@mail.com
```

Editor

```
$ git config --global core.editor notepad.exe
```

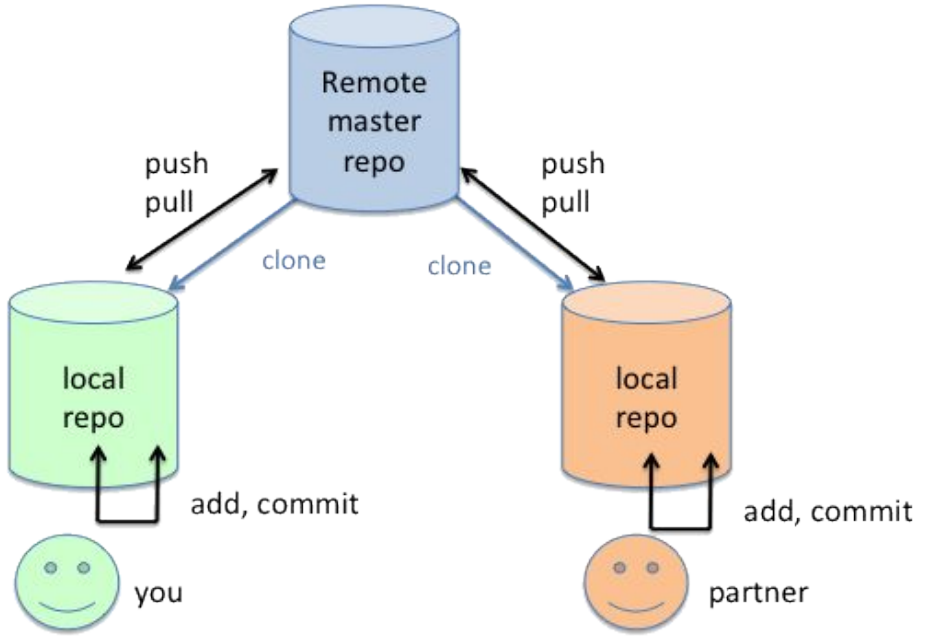
Check settings

```
$ git config --list
```

softserve

Create repository

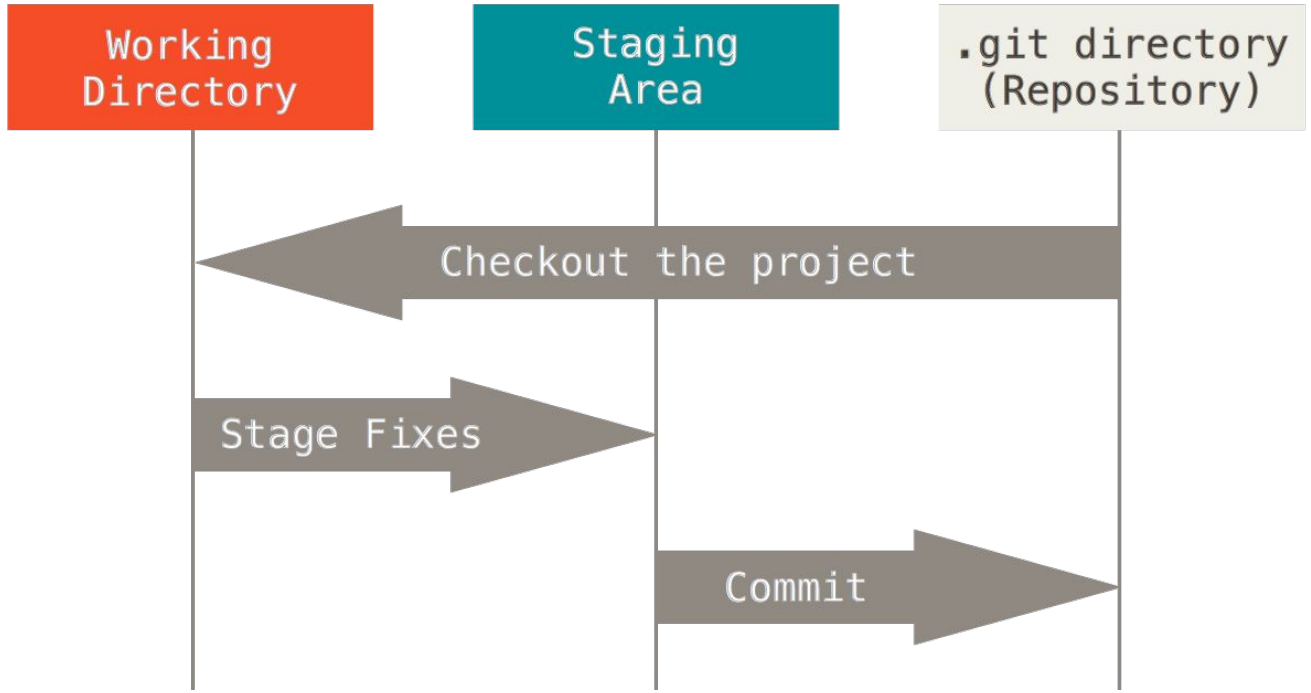
- **`git init`** – create an empty local repo
- **`git clone <URL>`** – create local repo from remote repo
- **`git remote add origin <URL>`** – add a remote repo to a local repo



softserve

Basic terms

- Local repository stored in hidden folder .git
- Working directory - folder with code
- Commit - snapshot of working directory
- Staging area or Index



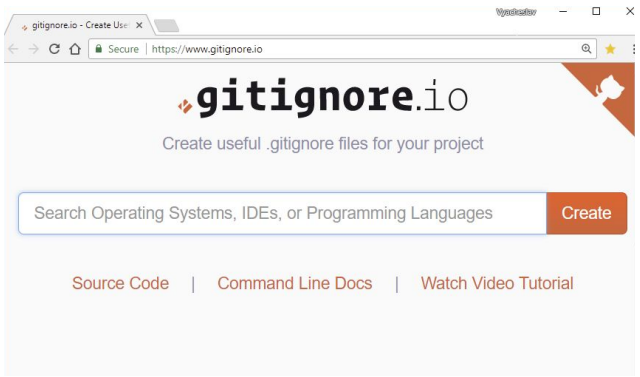
softserve

.gitignore

`.gitignore` - contains list of files and folders that are ignored by git in working folder

Typically ignored files:

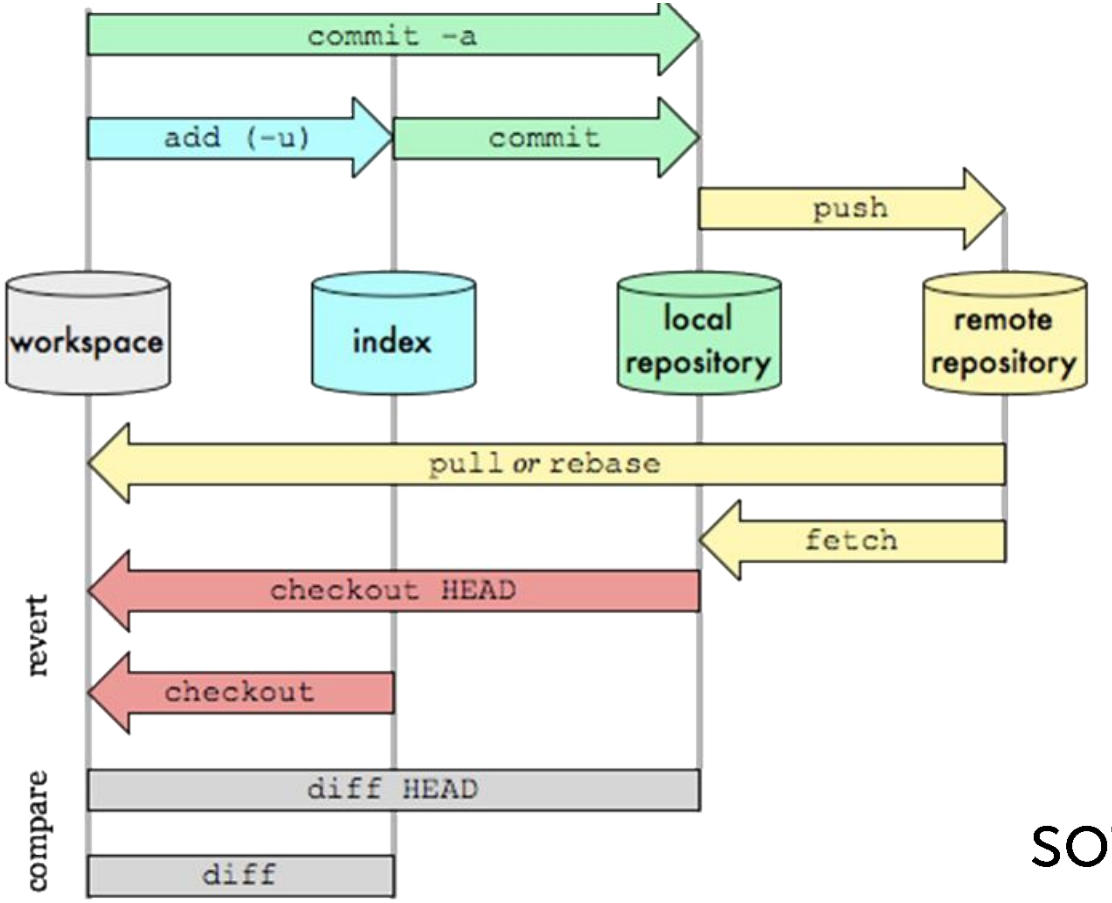
- Operating system files (Thumbs.db, .DS_Store)
- Application/IDE configuration files (.vscode)
- Generated files (*.exe, *.min.js)
- Language/framework files (.sass_cache, npm-debug.log)
- Files downloaded with package managers (node_modules)
- Credentials/tokens (wp-config.php)



softserve

Git data transport commands

- `git add`
- `git commit`
- `git push`
- `git fetch`
- `git checkout`
- `git merge`



softserve

Additional important commands

Get help:

- **`git help <command>`**
- **`git <command> --help`**

Show status and log:

- **`git status`** – Show the working tree status
- **`git log`** – Show commit logs
- **`git ls-files -s`** - Show files in the index

Remove and revert:

- **`git rm`** – Remove files from the working tree and from the index
- **`git reset`** - Resets changes

softserve

Additional important commands

Shortcuts:

- **`git commit -am`** - combines `add` and `commit`
- **`git pull`** - Combines `fetch` and `merge`

`Remote:`

- **`git remote -v`** - `List remote repos`
- **`git remote add`** - `Add remote repo`
- **`git remote rm`** - `Remove remote repo`

softserve

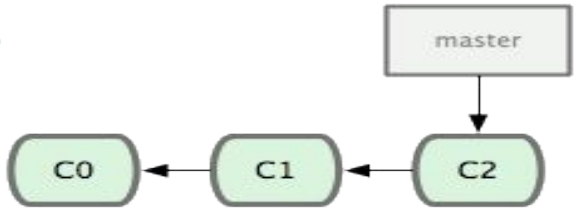
Branch

- A **branch** represents an independent line of development. Branches serve as an abstraction for the **edit/stage/commit** process
- Commands
  - **`git branch`** – list of branches in local repo
  - **`git branch <name>`** – create new local branch named “name”
  - **`git branch –d <name>`** – delete the branch named “name”
  - **`git branch –m <name>`** – rename the current branch to “name”

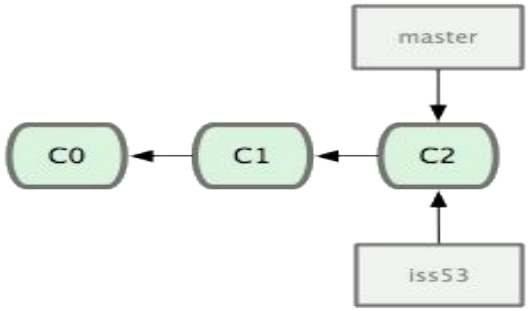
softserve

Let's imagine

1

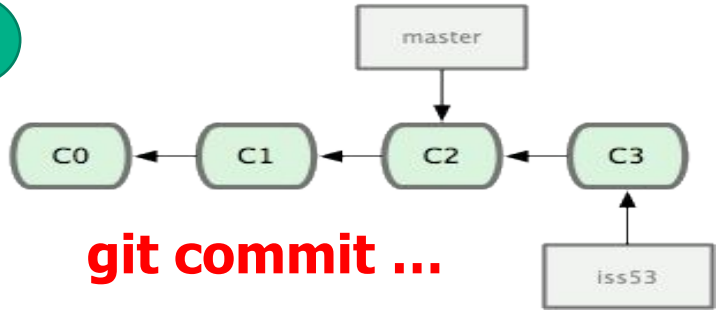


2



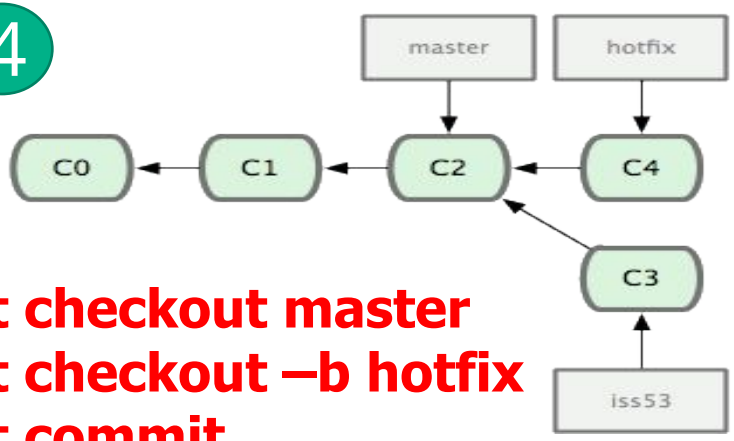
git checkout –b iss53

3



git commit …

4

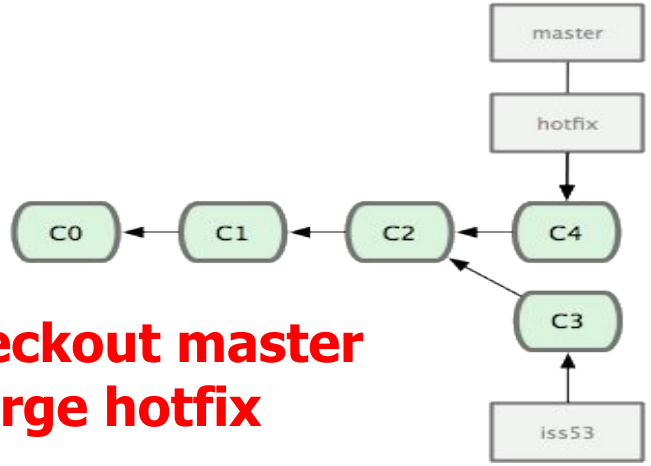


1. **git checkout master**
2. **git checkout –b hotfix**
3. **git commit …**

softserve

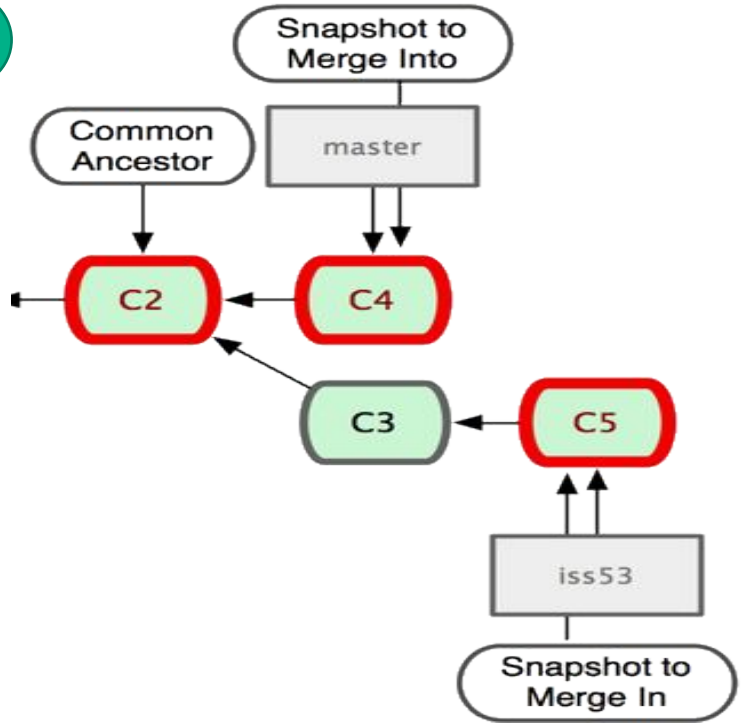
Merging

5

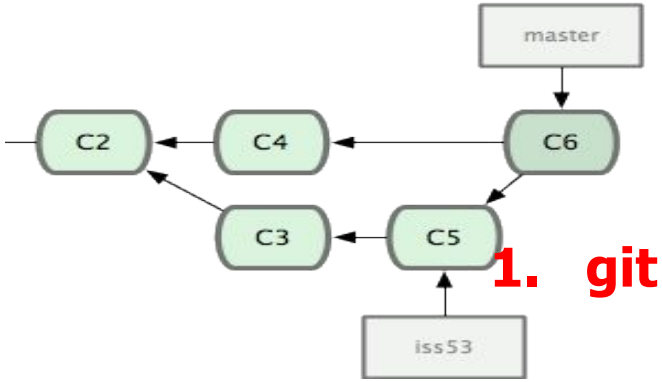


1. git checkout master
2. git merge hotfix

6



7

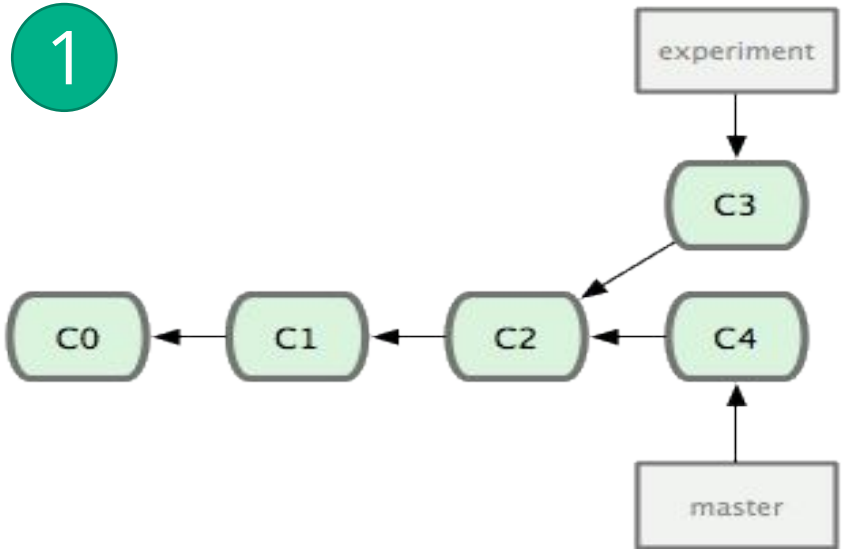


1. git merge iss53

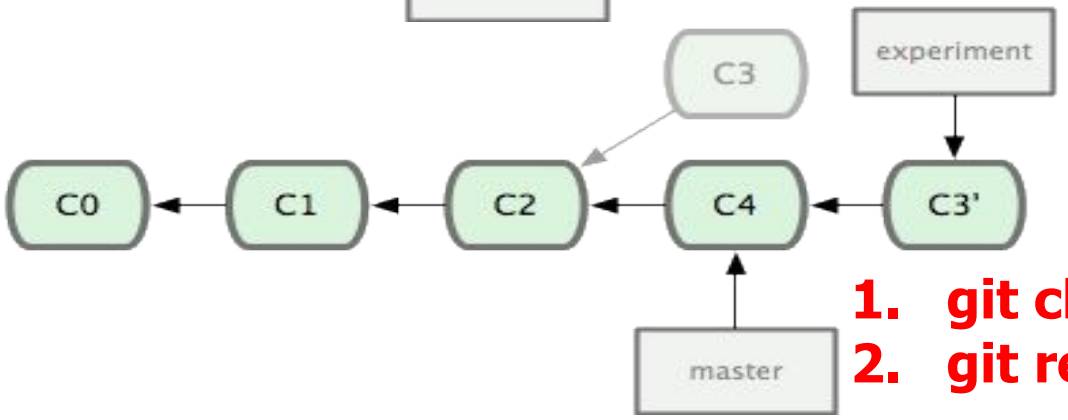
softserve

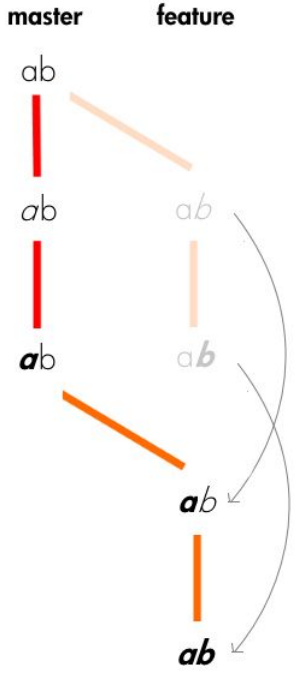
Rebasing

1



2





1. **git checkout experiment**
2. **git rebase master**

softserve

stash

- **git stash**
- **git stash list**
- **git stash apply**
- **git stash apply <stash number>**
- **git stash drop <stash number>**
- **git stash pop**

Practical tasks

1. Clone repository
2. Add to file «Zapovit.txt» few lines and commit it to local repository.
3. Push it to remote repository. Resolve conflict if needed
4. Make branch and checkout to it
5. Add few lines in the file.
6. Push changes to remote repo.
7. Merge the branch with master
8. Resolve conflicts, if needed
9. View master log.

softserve

References and Sources

Simplified views:

- Everyday commands
- Visual guide to GIT
- Easy version control with GIT
- https://ndpsoftware.com/git-cheatsheet.html#loc=local_repo

Some videos

- What is GIT
- Overview of Branching, Cloning, Pulling, and Merging. Demo of it on Git Bash
- Merge Conflicts. Git Tagging
- GIT for small teams
- Workflow for small teams

Advanced philosophy:

- Advanced programmer guide to GIT
- Version control SVN and GIT

softserve

References and Sources

https://git-scm.com/book/en/v2 - original documentation from Git team

https://www.atlassian.com/git/tutorials - Atlassian git tutorial

https://try.github.io - git course from codeschool

https://learngitbranching.js.org/ - practical course on git branching

softserve


FOR
THE
FUTURE
softserve