GIT
PYTHON
CORE

softserve

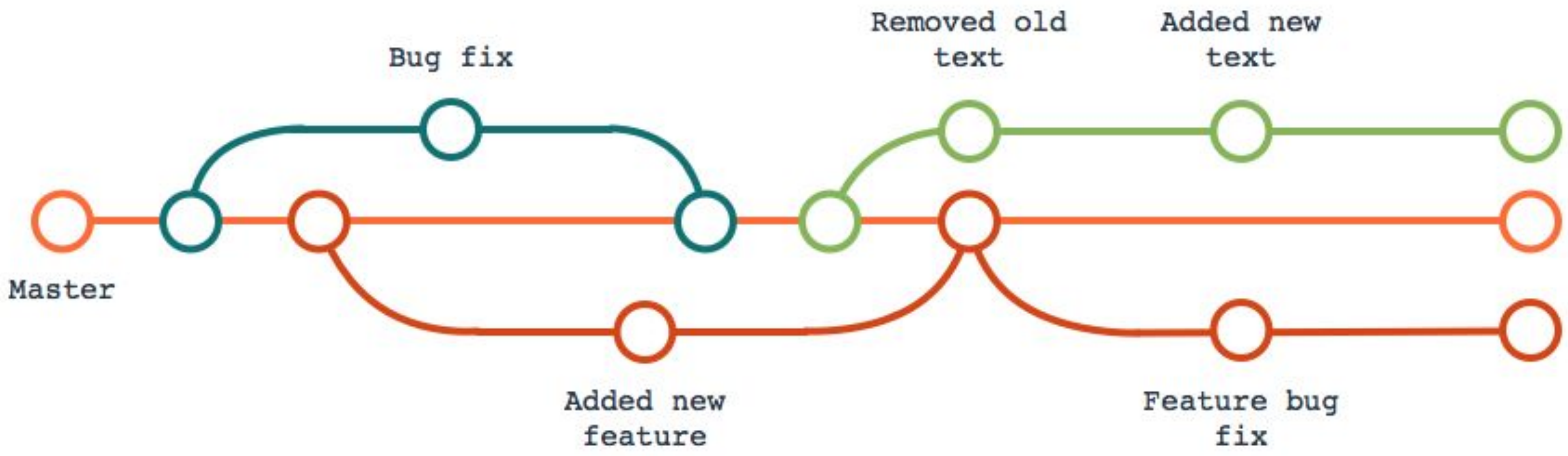
Agenda

- Source Control Management (SCM)
  - Fundamental Concepts
  - Terms
- Types of Version Control Systems
- Git
  - Before start
  - Configuration
  - Basics
  - Work cycle
  - Branches | Merging | Rebasing
- Practical tasks
- HomeWork

softserve

SCM

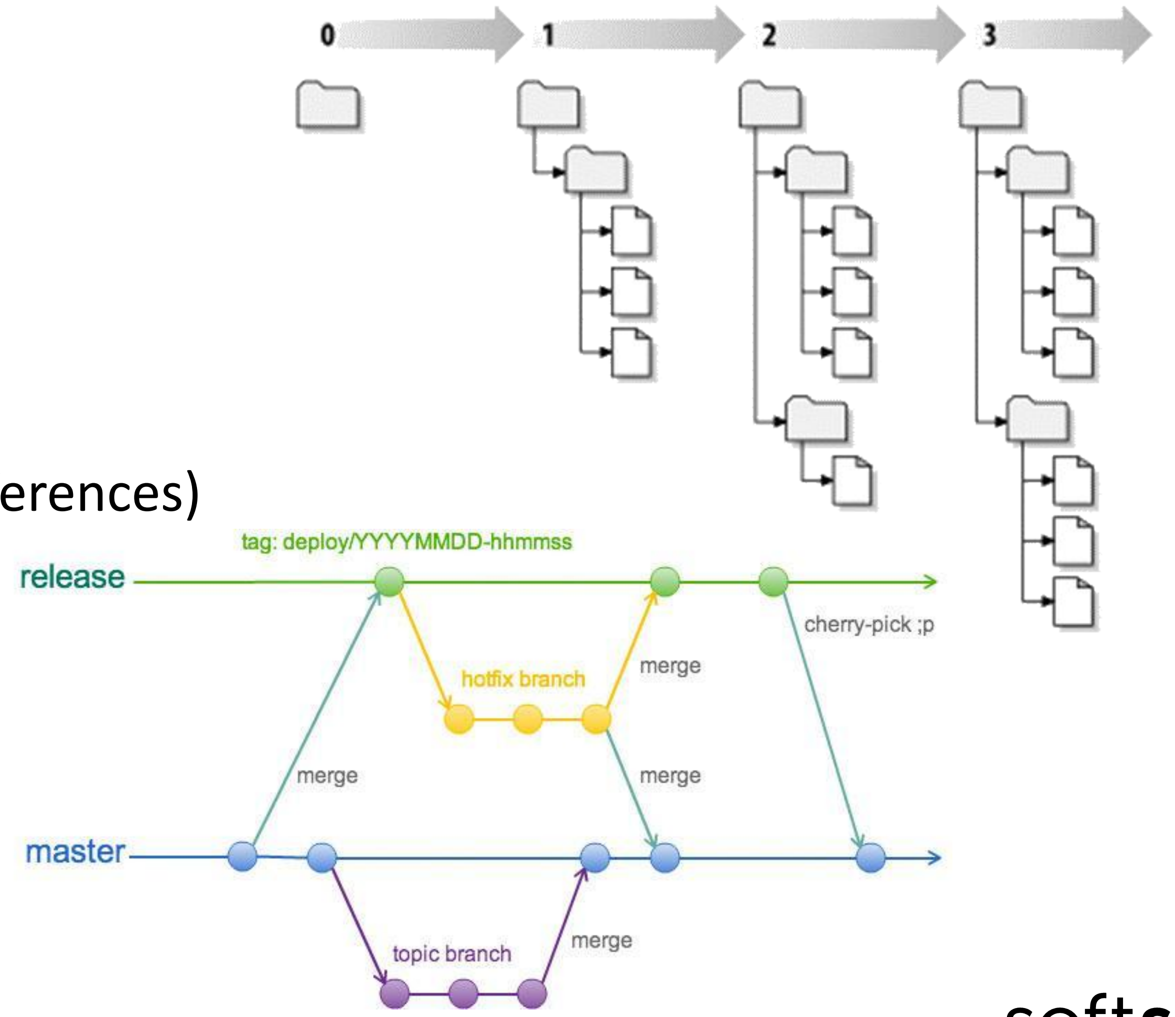
Revision control, also known as **version control** and **source control** (and an aspect of software configuration management), is the management of changes to documents, computer programs, large web sites, and other collections of information.



softserve

Fundamental Concepts of SCM

- Tracking changes
- Committing
- Revisions and Change sets
- Getting updates
- Conflicts
- Diffing (or, viewing the differences)
- Branching and merging



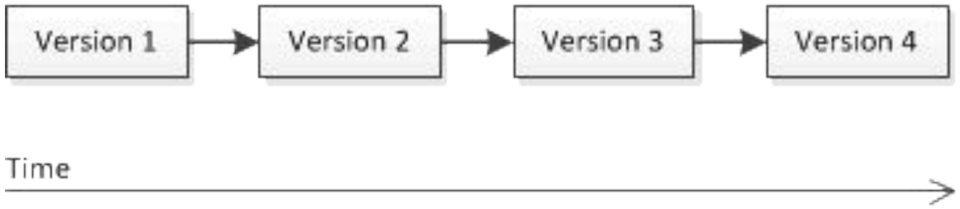
Main terms

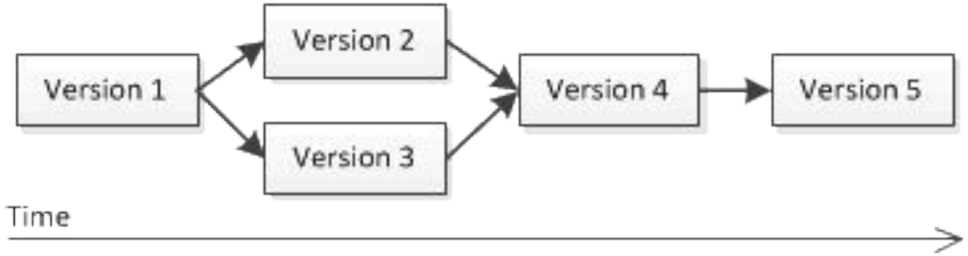
Repository

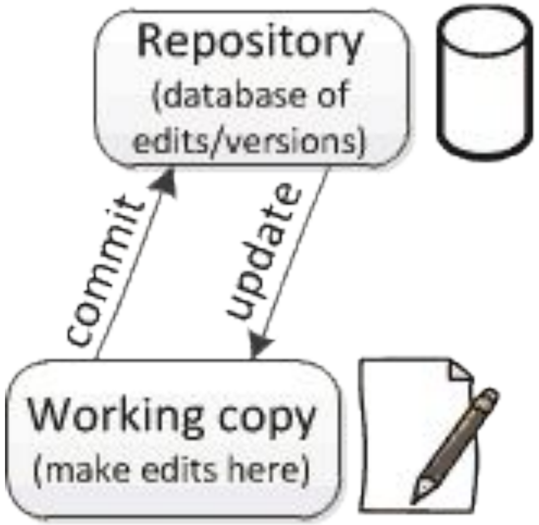
Working Copy

Merging

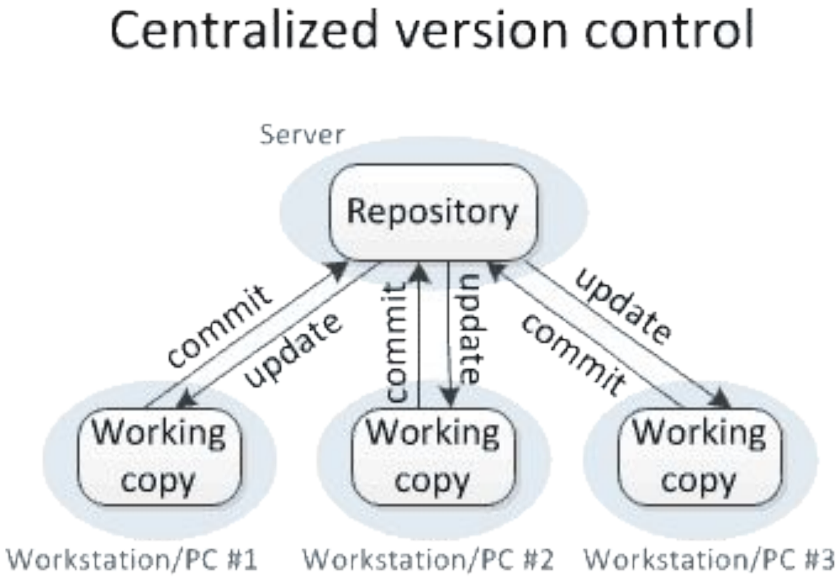
Revision

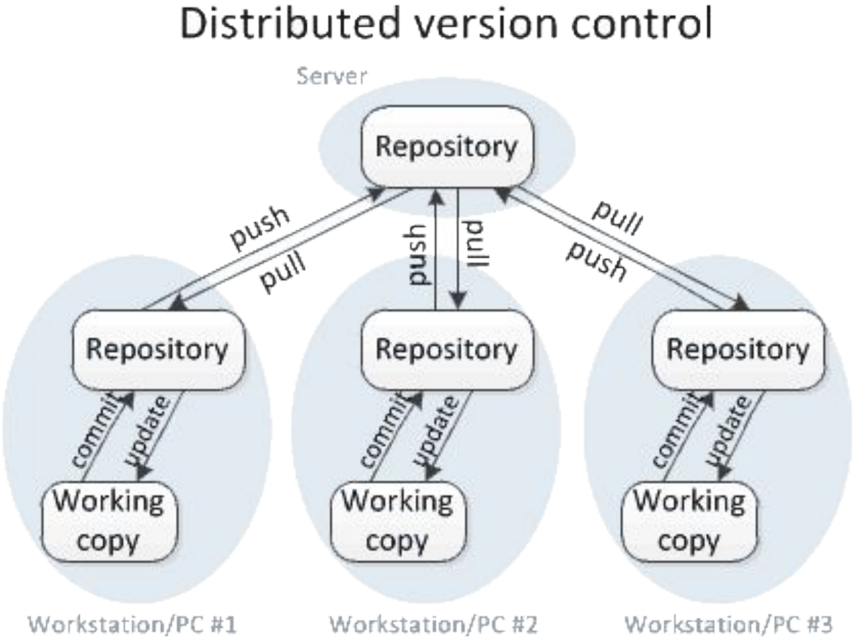






System version control





Centralized: CVS, Perforce, **SVN,**
Team Foundation Server (**TFS**)

Distributed: **Git**, Mercurial

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.

Before start

Firstly we need to check if we have a git client software.

Download and install **git**

Linux OS

Debian Family (Debian, Ubuntu, Mint)

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

Red Hat Family (RHEL, CentOS, Fedora)

```
#yum install git
```

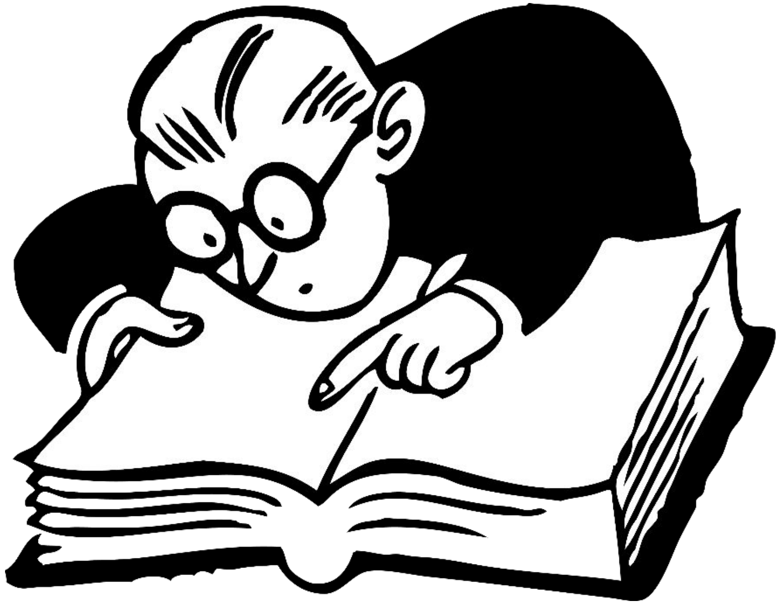
MS Windows OS

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

If we need to know sth ☺️

Help yourself

```
$git help <command>
$git <command> --help
$man git-<command>
```



Let’s configure git ☺️

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

Identity

```
$ git config --global user.name “Liubov Koliasa“
$ git config --global user.email lkoliasa@mail.com
```

Editor

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

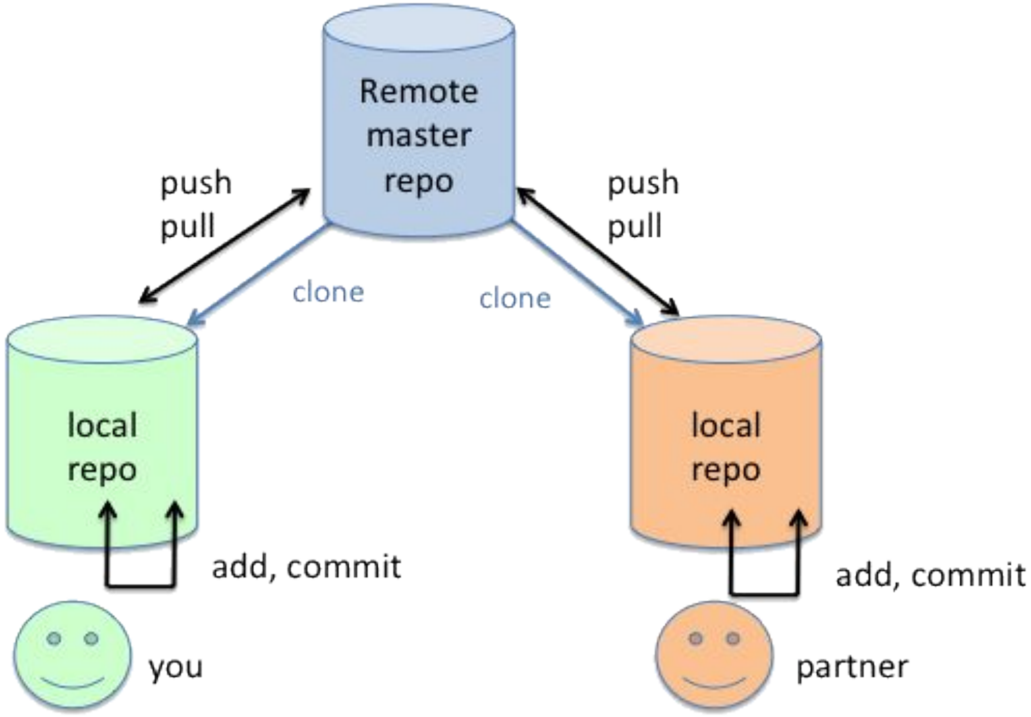
Check settings

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

Create repository

`git init` – create an empty local repo

`git clone <URL>` – create local repo from remote repo

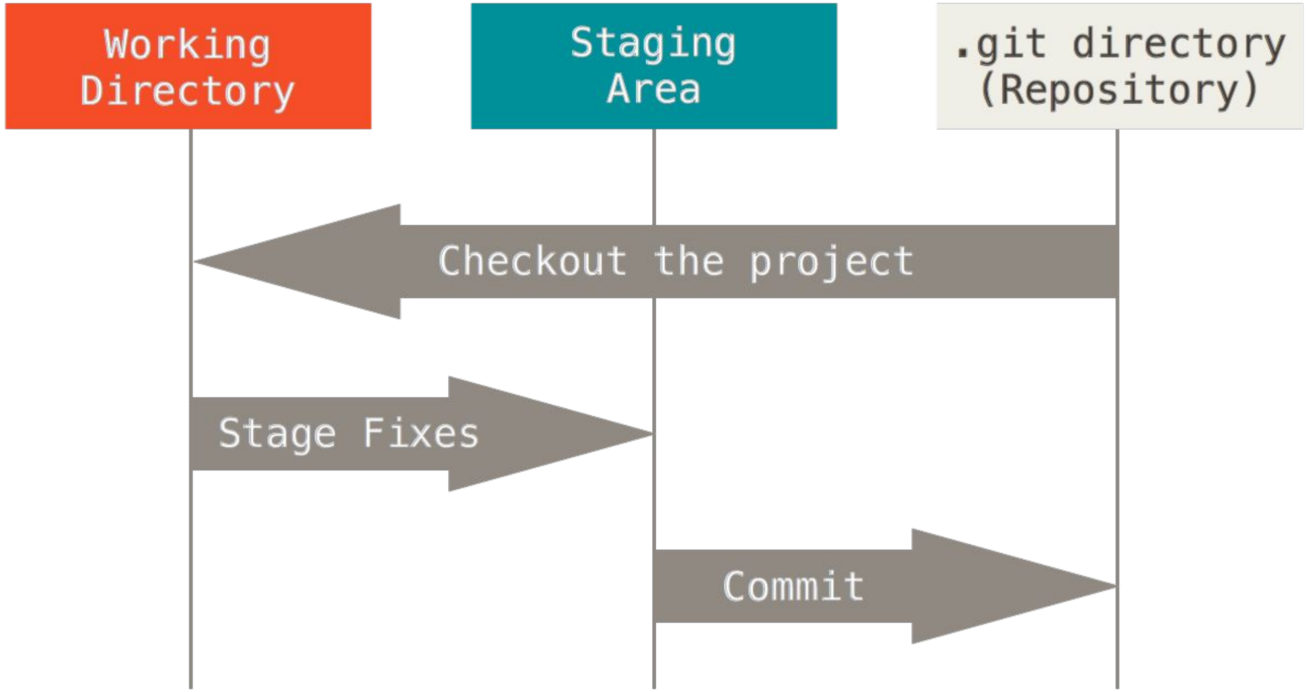


softserve

GIT basics

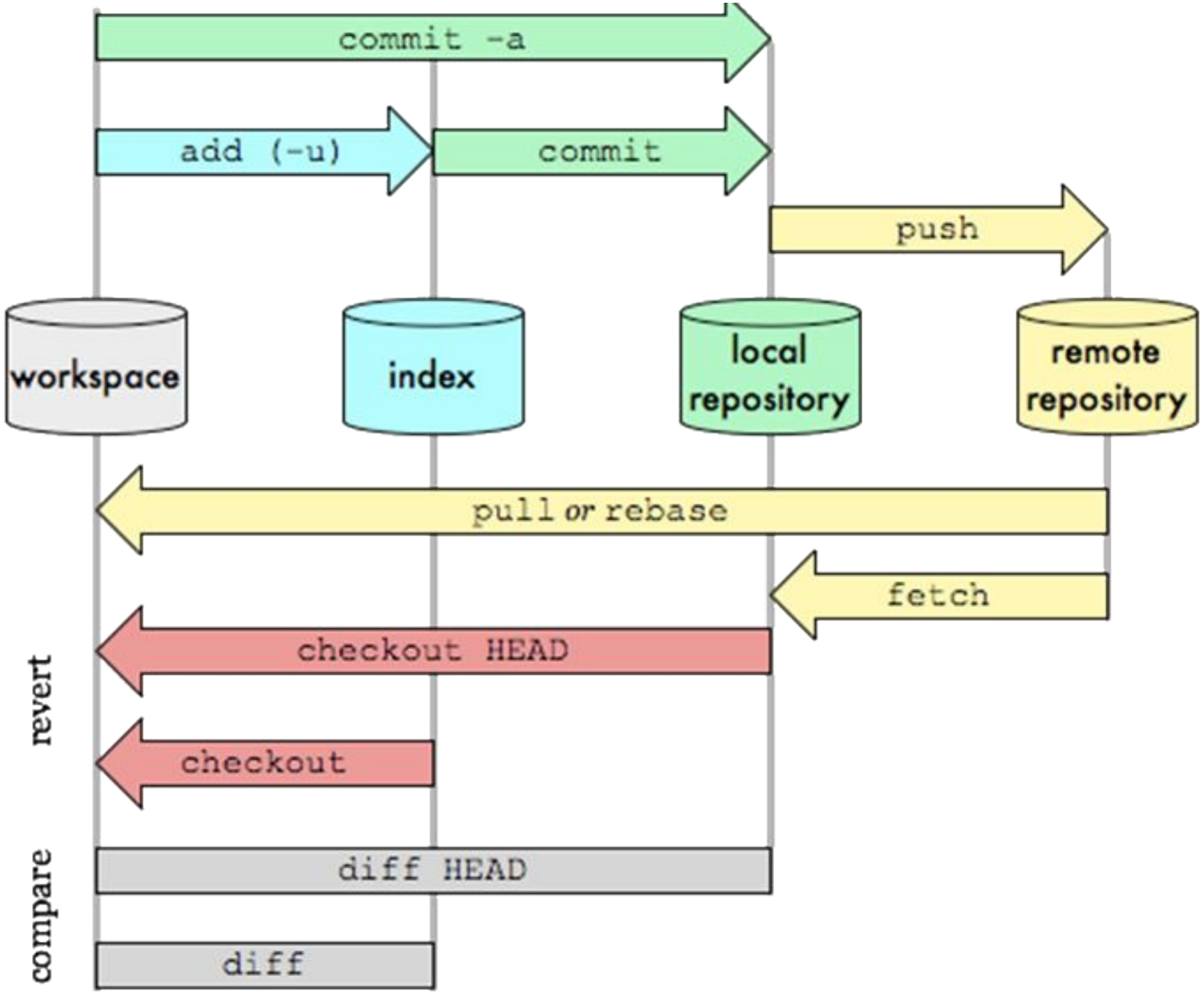
Git store snapshots of file system not differences!!!

Almost every operation is local



softserve

Git data transport commands



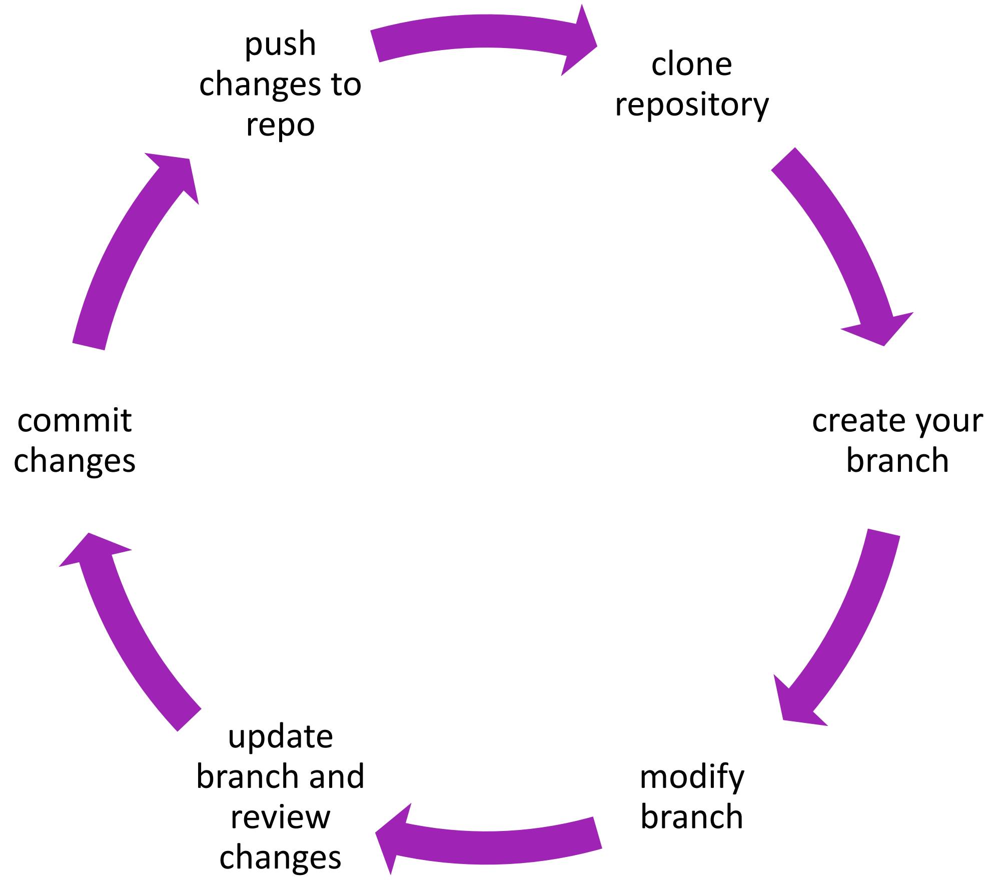
Must know commands!

`git status` - Show the working tree status

`git log` – Show commit logs

`git rm` – Remove files from the working tree and from the index

GIT Work Cycle



Clone repository
- git clone
- git init

Create your branch
- git branch

Modify working copy
- git add
- git reset
- git mv
- git rm

Update branch and review changes
- git status
- git log
- git diff
- git fetch

Commit changes
- git commit

Push changes to repo
- git push

softserve

Branch

A **branch** represents an independent line of development. Branches serve as an abstraction for the **edit/stage/commit** process

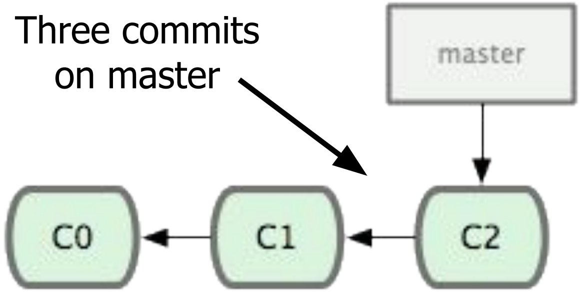
Commands

- `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”

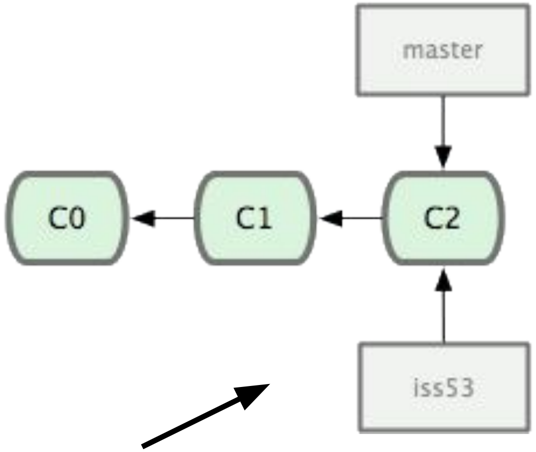
Let’s imagine

1

Three commits on master

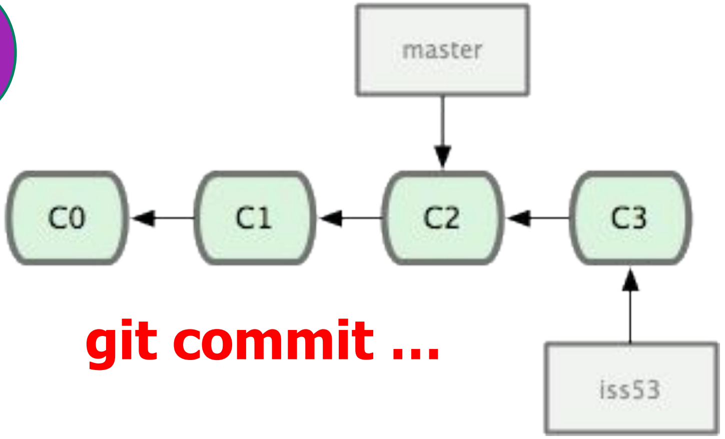


2



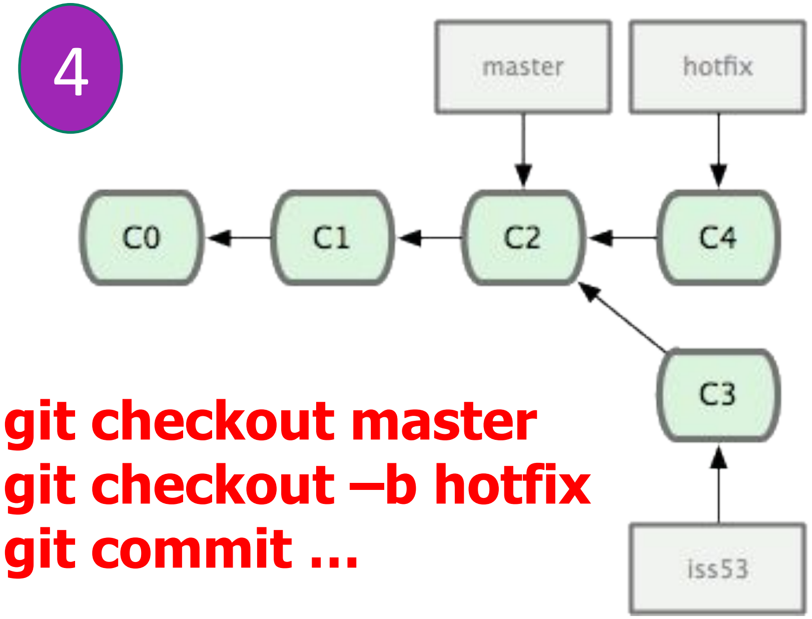
git checkout –b iss53

3



git commit …

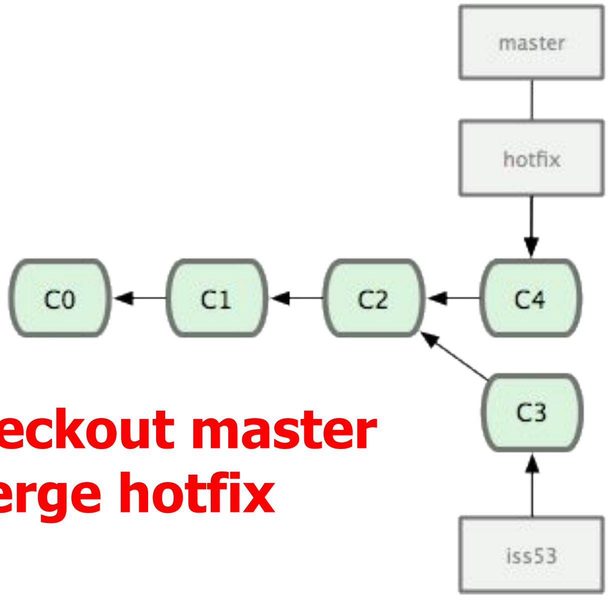
4



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

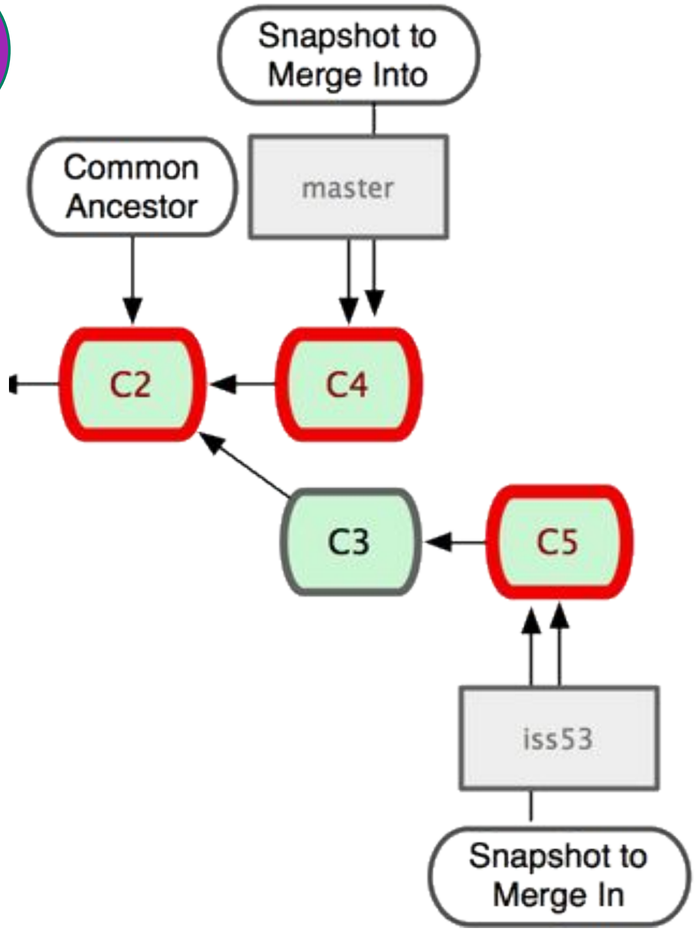
Merging

5

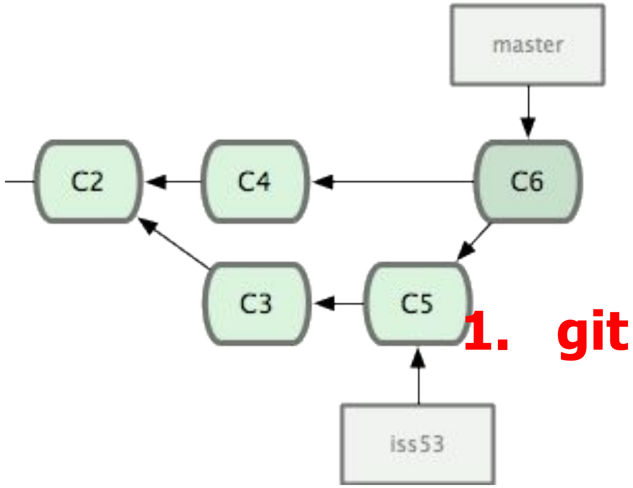


1. **git checkout master**
2. **git merge hotfix**

6



7

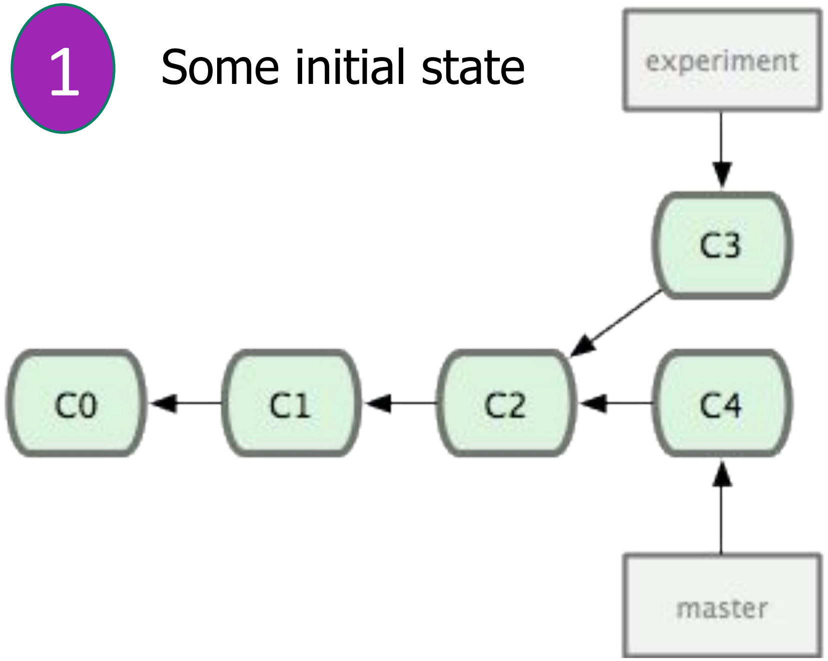


1. **git merge iss53**

softserve

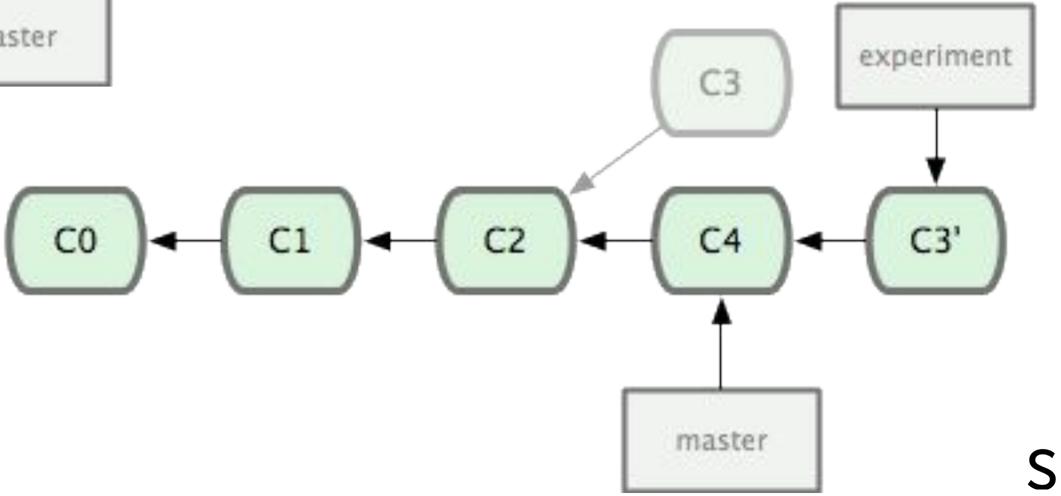
Rebasing

1 Some initial state



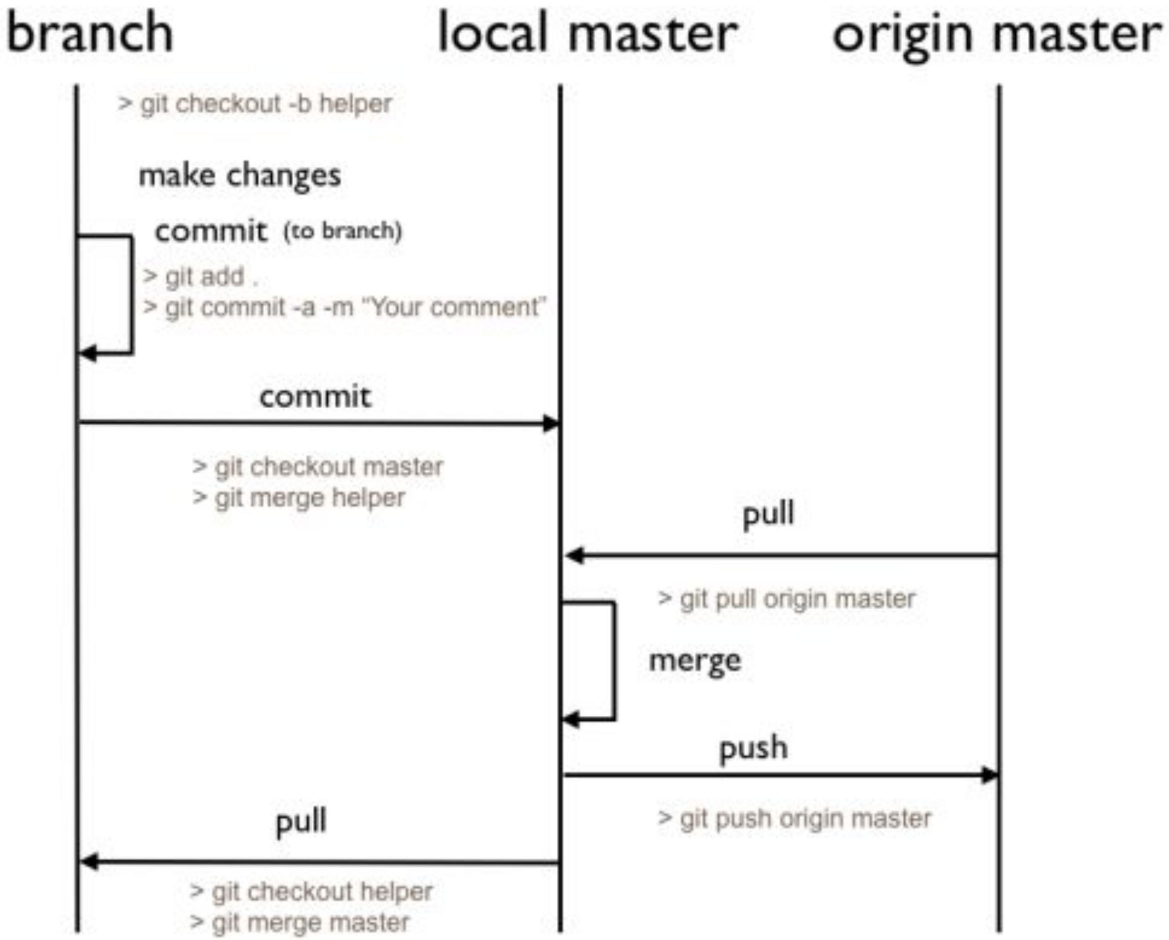
1. **git checkout experiment**
2. **git rebase master**

2



softserve

Team player / issue / bug fix philosophy



```
> git checkout -b helper
```

```
> git add .
> git commit -a -m "Your comment"
```

```
> git checkout master
> git merge helper
```

```
> git pull origin master
```

```
> git push origin master
```

```
> git checkout helper
> git merge master
```

Command syntax	Description
repo init	initializes a new client
repo sync	syncs client to repositories
repo start	starts a new branch
git add	stages files (adds to index)
repo status	shows status of current branch
git commit	commits staged files
git branch	shows current branches
git branch [branch]	creates new topic branch
git checkout [branch]	switches HEAD to specified branch
git merge [branch]	merges [branch] with current branch
git diff	shows diff of unstaged changes
git log	shows history on current branch
repo upload	Uploads changes to review server

Tasks

Clone repository https://github.com/kolyasalubov/Lv-367.PythonCore.git

Add to file «ZenPython.txt» few lines and commit it to local repository.

Push it to remote repository.

Make branch and checkout to it

Add few lines in the file.

Push changes to remote repo.

Merge the branch with master

Resolve conflicts, if needed

View master log.

HomeWork (online course)

Play on site https://try.github.io

Please register on Learn Git Branching: http://learngitbranching.js.org/

and play game

Clone repo

https://github.com/kolyasalubov/Lv-416.PythonCore.git

Create branch <your name>

Push into this branch your project from HW 1

References and Sources

Simplified views:

- Everyday commands
- Visual guide to GIT
- Easy version control with GIT

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

THANK YOU FOR ATTENTION

softserve