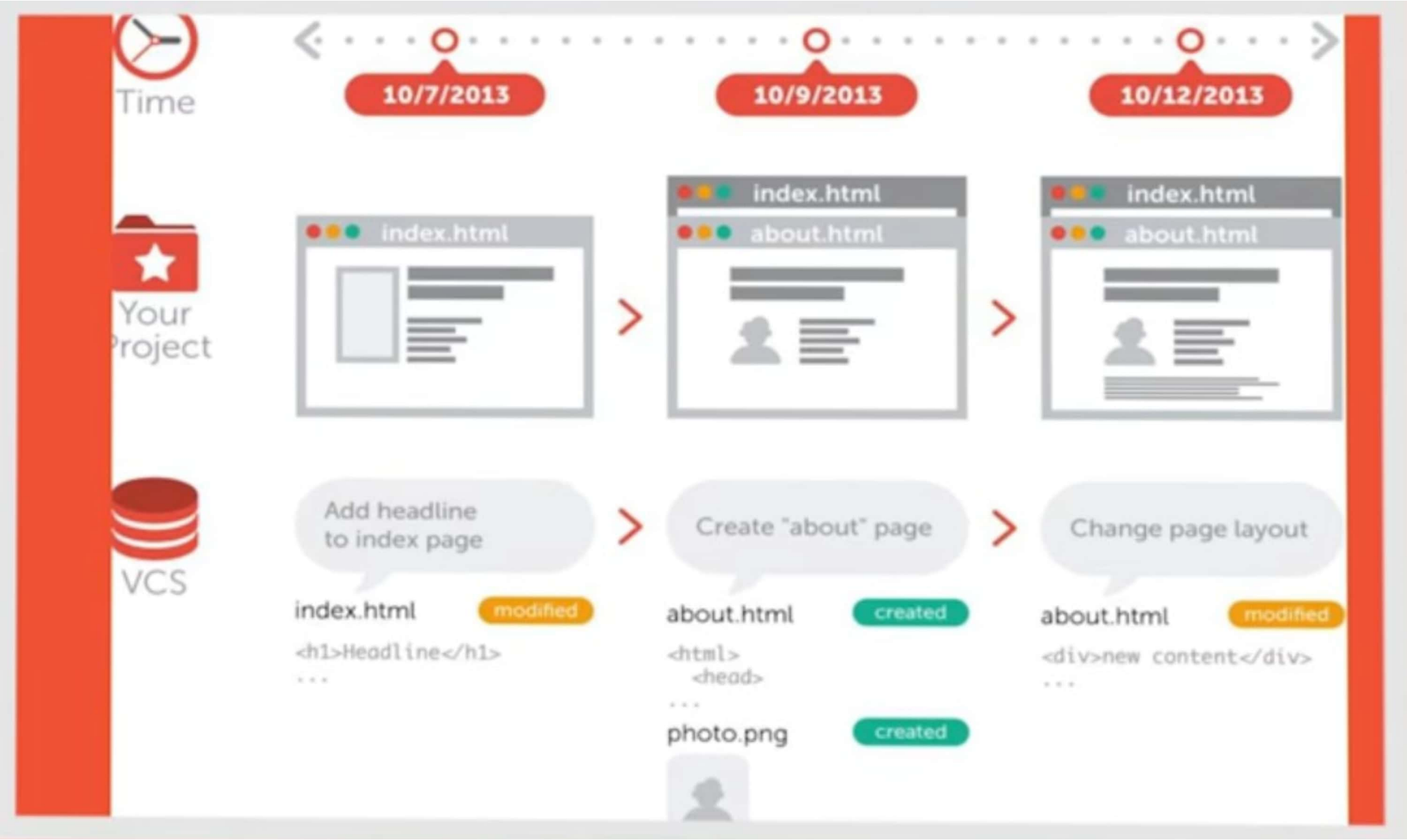
What is Version Control System?

- A software utility that tracks and manages changes to a filesystem.
- Also known as revision control or source control system
- It's a management of changes to documents, computer programs, large websites and other collection of information

What is Version Control System?

- VCS options include Git, Mercurial, SVN and preforce.
- You can think of a VCS as a kind of “database”.
- It lets you save a snapshot of your complete project at any time you want
- When you later take a look at an older snapshot your VCS shows you exactly how it differed from the previous one


Time
10/7/2013
10/9/2013
10/12/2013
Your Project
index.html
index.html
about.html
index.html
about.html
VCS
Add headline to index page
Create "about" page
Change page layout
index.html
modified
<h1>Headline</h1>
...
about.html
created
<html>
<head>
...
photo.png
created
about.html
modified
new content
...

What is Version Control System?

- Version control is independent of the kind of project / technology / framework you're working with
- It works just as well for an HTML website as it does for a design project or an iPhone app
- It lets you work with any tool you like; it doesn't care what kind of text editor, graphics program, file manager or other tool you use

Why Use a Version Control System?

- Collaboration
  - Without a VCS in place, you're probably working together in a shared folder on the same set of files.
  - And you have to coordinate with others so that they don't work on same file, it will be very difficult to manage
  - With a VCS, everybody on the team is able to work absolutely freel - on any file at any time.
  - The VCS will later allow you to merge all the changes into a common version.

Why Use a Version Control System?

- Restoring Previous Versions
  - Being able to restore older versions of a file or whole project
  - If the changes you've made
  - lately prove to be garbage, you can simply undo them in a few clicks

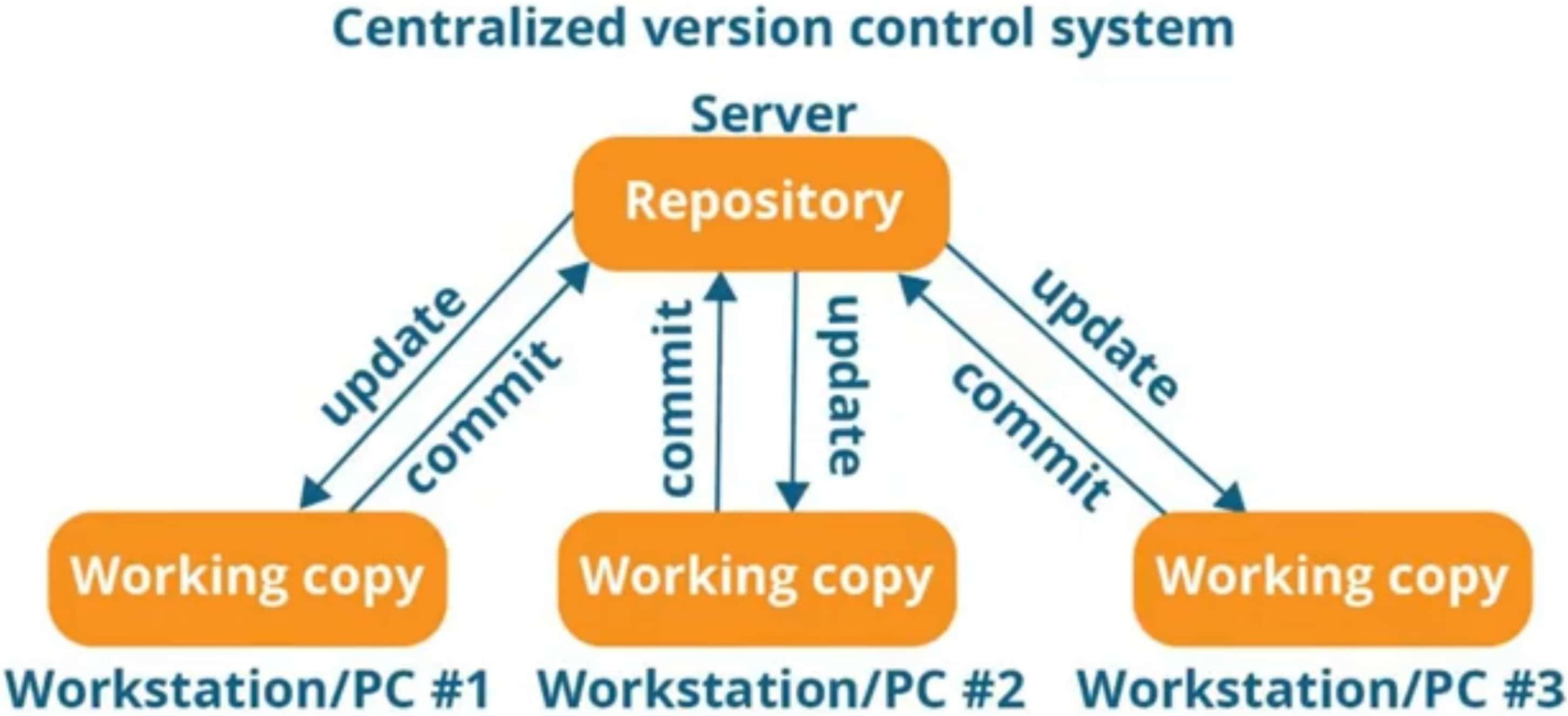
Why Use a Version Control System?

- Understanding What Happened
  - Every time you save a new version of your project, your VCS requires you to provide a short description of what was changed
  - Additionally (if it's a code / text file), you can see what exactly was changed in the file's content

Types of Version Control Systems

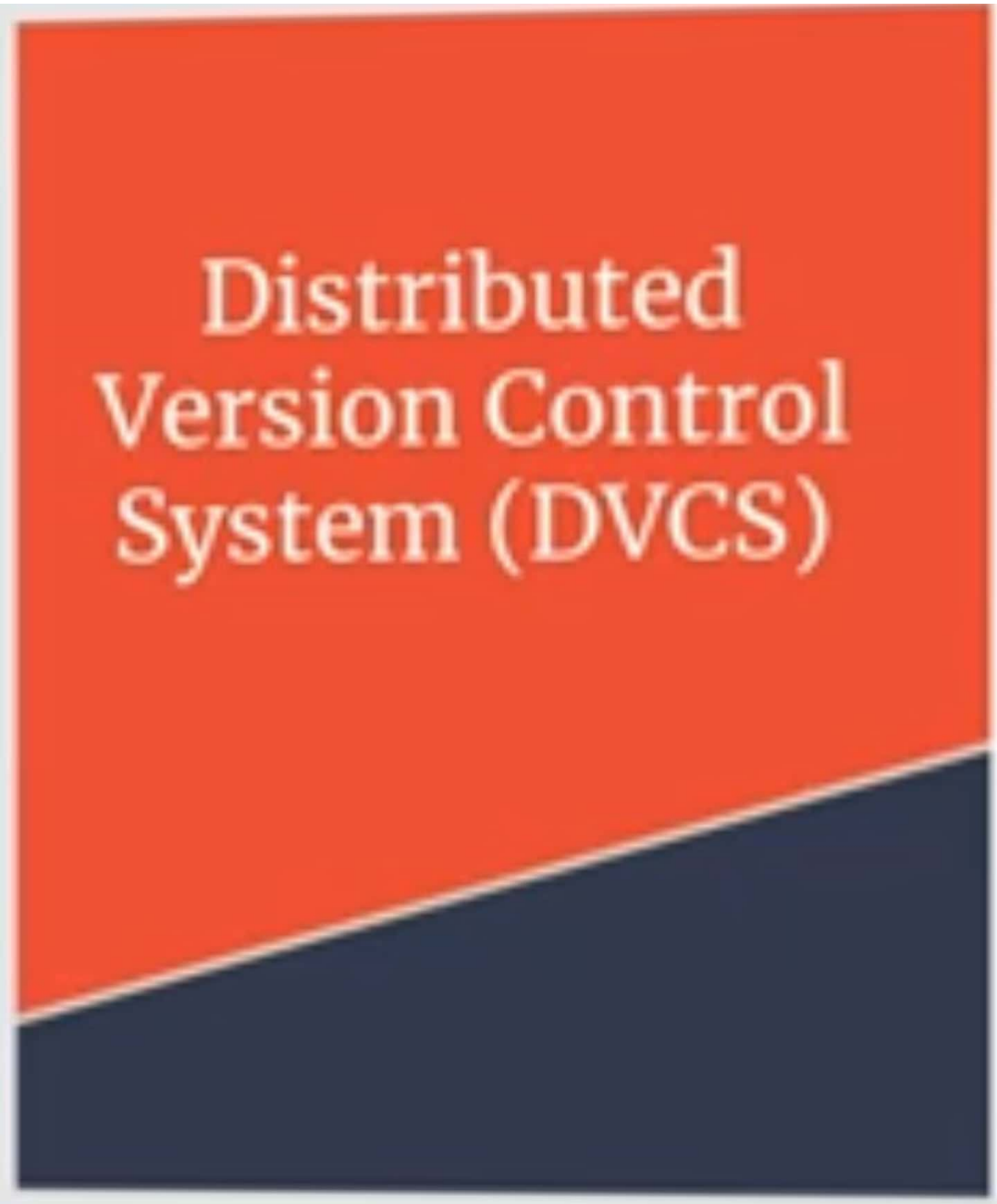
- Centralized Version Control System (CVCS)
- Distributed Version Control System (DVCS)


Centralized Version Control System (CVCS)
Every programmer can extract or update their workstations with the data present in the repository or can make changes to the data or commit in the repository. Every operation is performed directly on the repository.

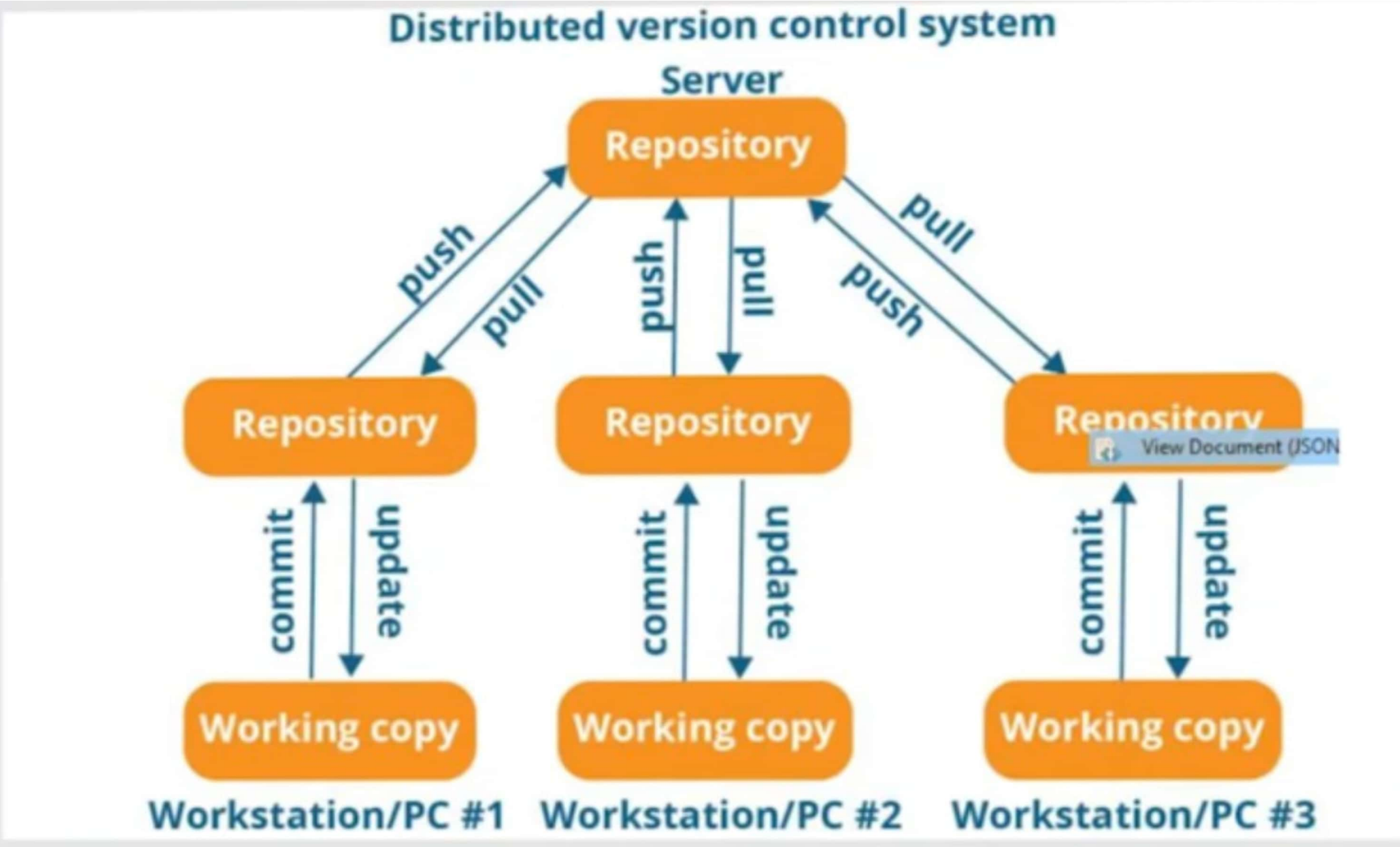

Centralized version control system
Server
Repository
update
commit
commit
update
update
commit
Working copy
Working copy
Working copy
Workstation/PC #1
Workstation/PC #2
Workstation/PC #3

CVCS Drawbacks

- It is not locally available; meaning you always need to be connected to a network to perform any action.
- Since everything is centralized, in any case of the central server getting crashed or corrupted will result in losing the entire data of the project.



These systems do not necessarily rely on a central server to store all the versions of a project file.


Distributed version control system
Server
Repository
push
pull
push
pull
pull
push
Repository
Repository
Repository
View Document (JSON
commit
update
commit
update
commit
update
Working copy
Working copy
Working copy
Workstation/PC #1
Workstation/PC #2
Workstation/PC #3

Distributed Version Control System (DVCS)

Programmer can update their local repositories with new data from the central server by an operation called "**pull**" and affect changes to the main repository by an operation called "**push**" from their local repository.

Distributed Version Control System (DVCS)

Advantages

All operations (except push & pull) are very fast because the tool only needs to access the hard drive, not a remote server. Hence, you do not always need an internet connection.

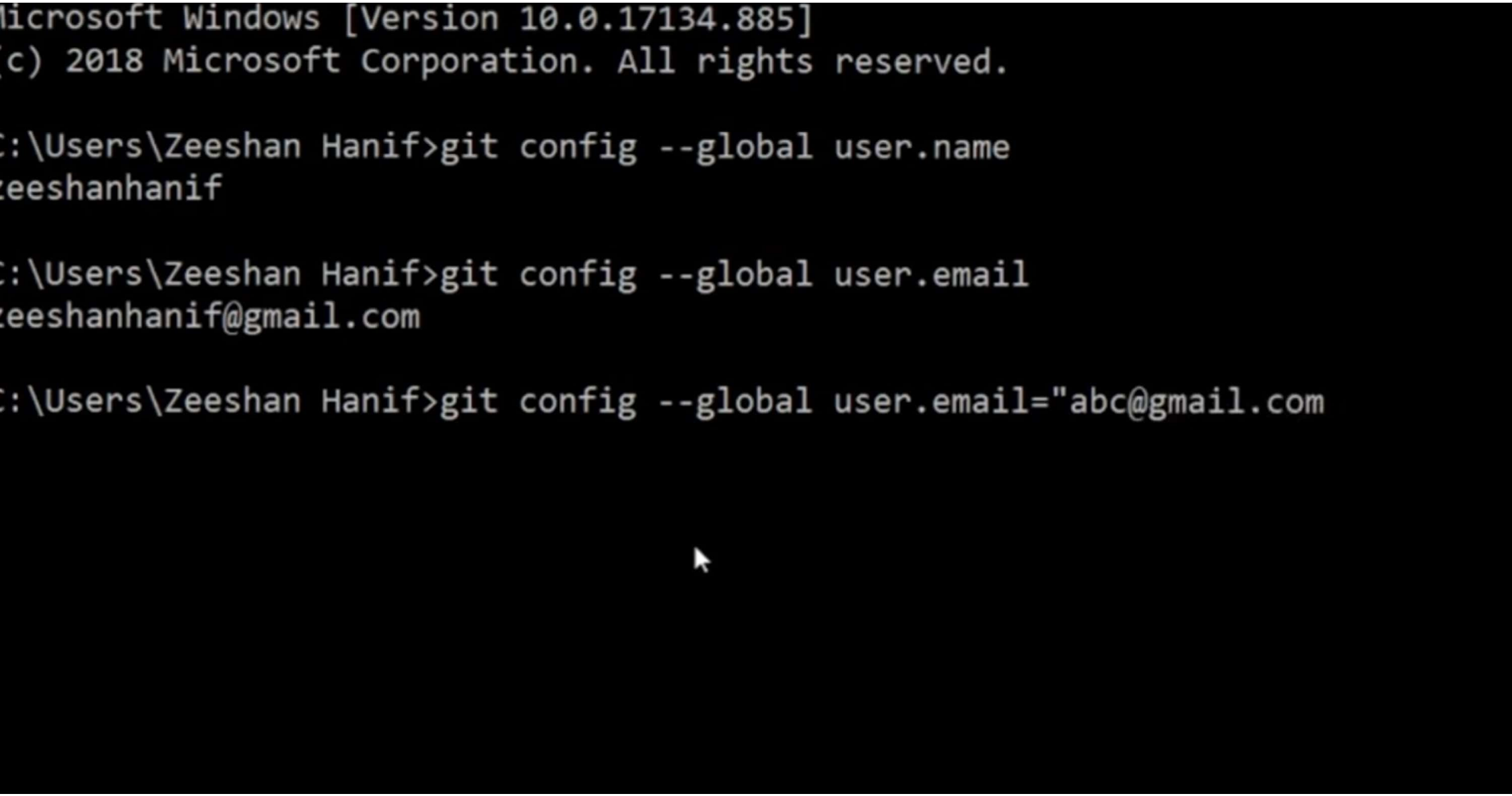

Distributed
Version Control
System (DVCS)
GIT is Distributed version
Control System

Installing and Setting Up SmartGit GUI Tool

- Go to https://www.syntevo.com/smartgit/download
- Download SmartGit
- Install SmartGit with all default settings

Installing and Setting Up Git

- Open Terminal and run following commands to setup your name and email id
  - git config --global user.name “Your Name”
  - git config --global user.email “Your Email”


Microsoft Windows [Version 10.0.17134.885]
(c) 2018 Microsoft Corporation. All rights reserved.
C:\Users\Zeeshan Hanif>git config --global user.name
Zeeshanhanif
C:\Users\Zeeshan Hanif>git config --global user.email
Zeeshanhanif@gmail.com
C:\Users\Zeeshan Hanif>git config --global user.email="abc@gmail.com

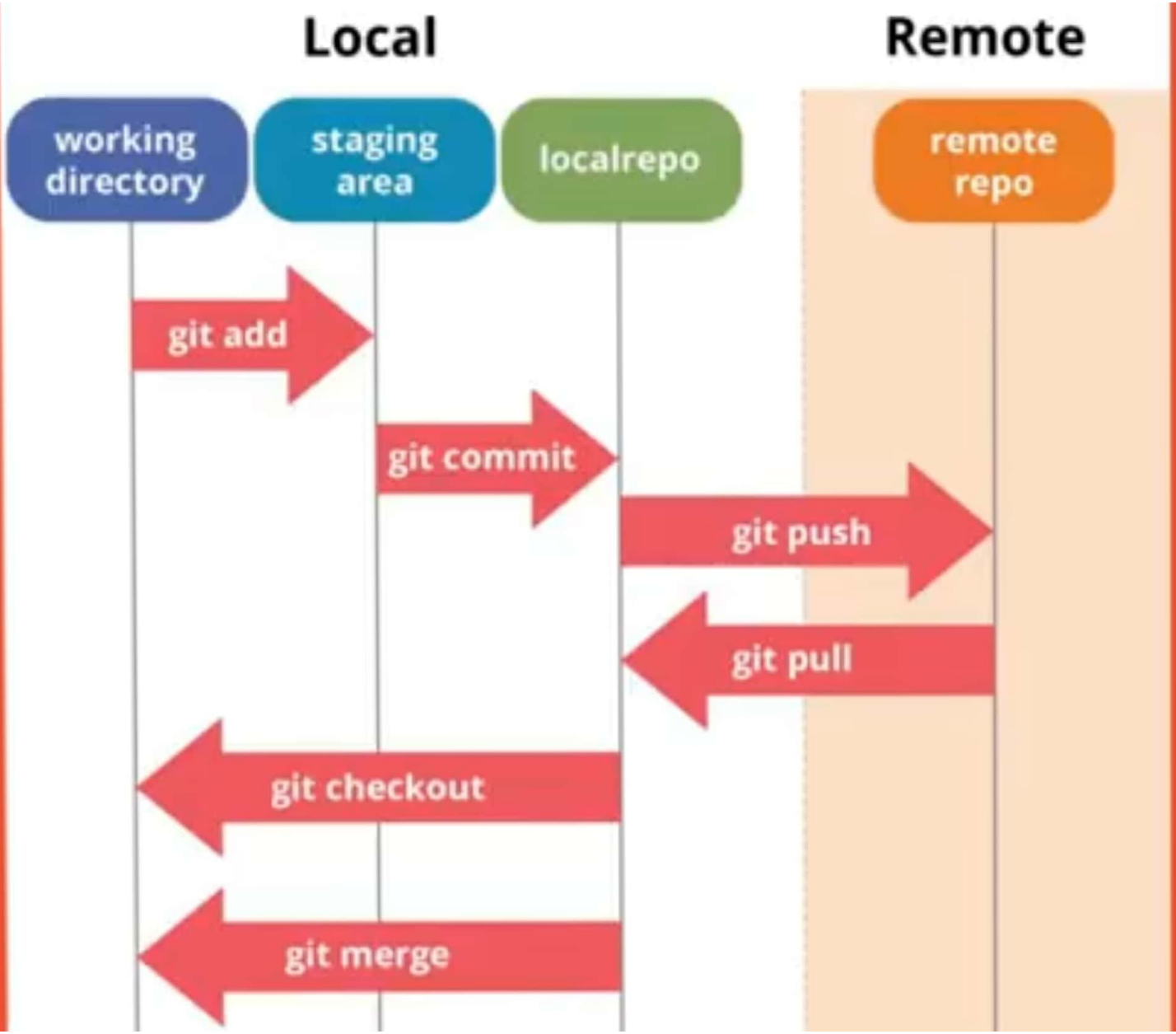
Basic operations in Git are

- Initialize
- Add
- Commit
- Pull
- Push

Advanced Git operations

- Branching
- Merging
- Rebasing

Git Operations



Important Terms

1. Repository

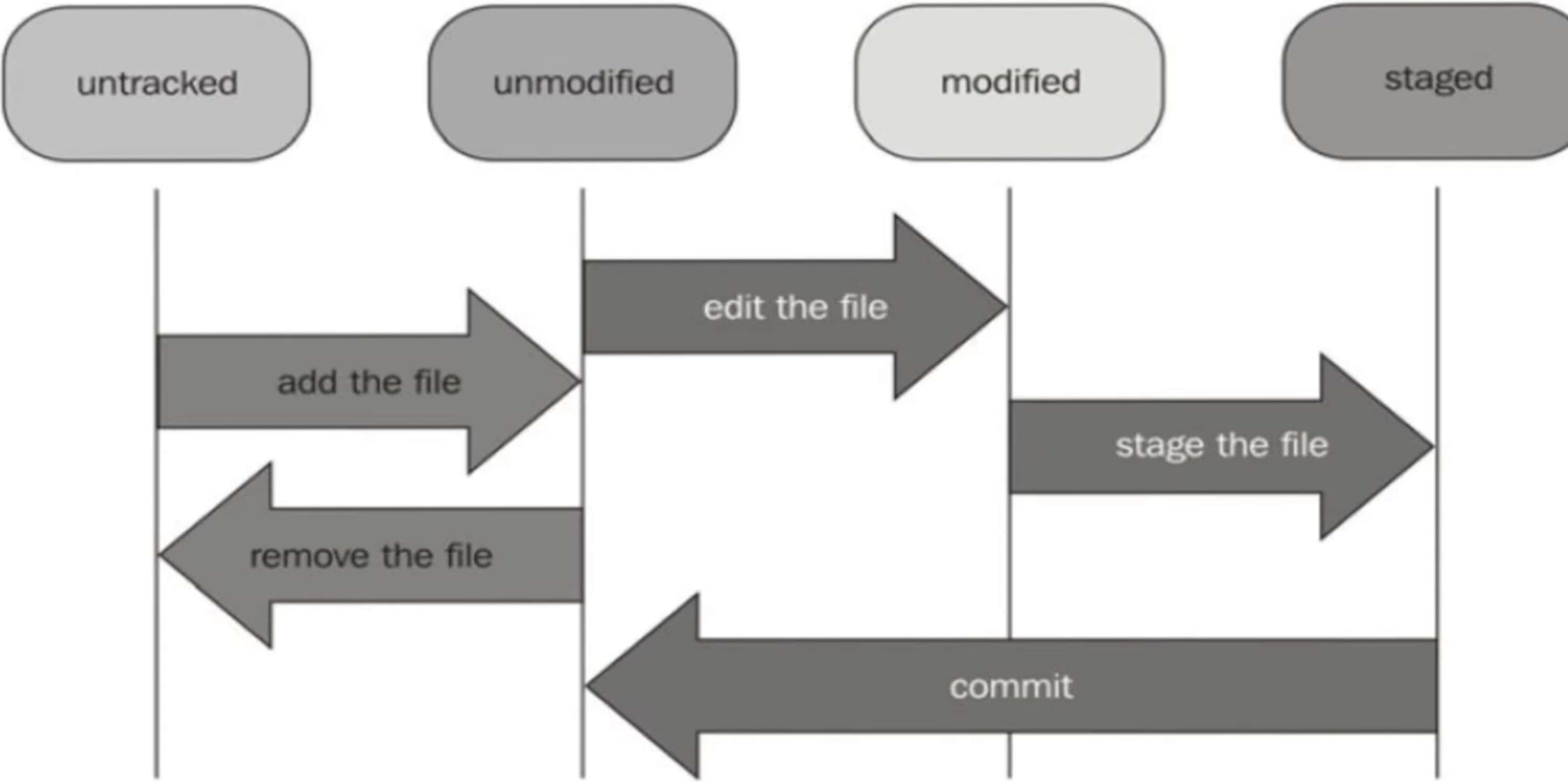
- Think of a repository as a kind of database where your VCS stores all the versions and metadata that accumulate in the course of your project.
- In Git, the repository is just a simple hidden folder named ".git" in the root directory of your project

Important Terms

2. Working Directory

The root folder of your project is often called the "working copy" (or "working directory"). It's the directory on your local computer that contains your project's files.

File Status Lifecycle



Important Terms

4. Staging Area

Staging area is a virtual place that collects all the files you want to include in the next commit

In Git, simply making some changes doesn‘t mean they‘re automatically committed.

Working Copy

Your Project's Files

Git watches tracked files for new local modifications...

Staging Area

Changes included in the Next Commit

Local Repository

The ".git" Folder

Tracked (and modified)

M — If a file was modified since it was last committed, you can stage & commit these changes

stage

M — Changes that were added to the Staging Area will be included in the next commit

commit

All changes contained in a commit are saved in the local repository as a new revision

M — Changes that are **not staged** will not be committed & remain as local changes until you stage & commit or discard them

Untracked

? — Changes in untracked files aren't watched. If you want them included in version control, you have to tell Git to start tracking them. If not, you should consider ignoring them.

Starting with an Unversioned, Local Project

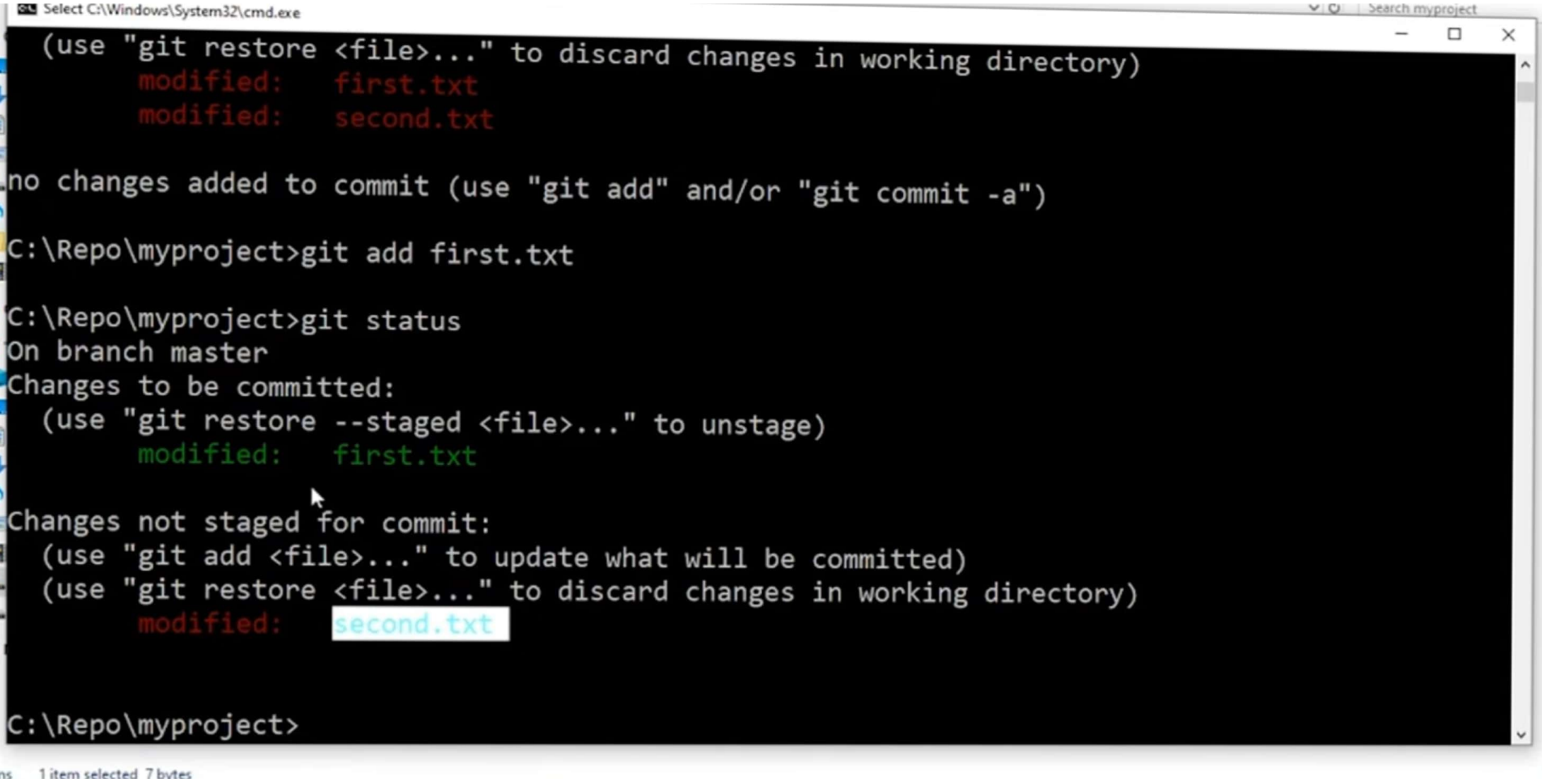
1. Open terminal and create directory on your machine
   a. C:\Repo\myproject
2. Go into directory in terminal
3. Initialize repository in this directory
   a. git init
   b. This will create .git hidden folder in your directory which will make your current folder, a git repository

Starting with an Unversioned, Local Project

4. Create first.txt and second.txt
5. Check status, it will show you two untracked files
    a. git status
6. Add these files to staging area
    a. Two ways to add files in staging
        i. git add first.txt second.txt OR
        ii. git add .

Starting with an Unversioned, Local Project

4. Commit your files to VCS with commit message
   a. git commit -m "implemented new feature"
5. Commit message is very important, you should provide proper commit message so that it can be refer back to identify what was added in that commit
6. Check logs to see commit history
   a. git log


Select C:\Windows\System32\cmd.exe
(use "git restore <file>..." to discard changes in working directory)
modified: first.txt
modified: second.txt
no changes added to commit (use "git add" and/or "git commit -a")
C:\Repo\myproject>git add first.txt
C:\Repo\myproject>git status
On branch master
Changes to be committed:
(use "git restore --staged <file>..." to unstage)
modified: first.txt
Changes not staged for commit:
(use "git add <file>..." to update what will be committed)
(use "git restore <file>..." to discard changes in working directory)
modified: second.txt
C:\Repo\myproject>

Ignoring Files

- Typically, in every project and on every platform, there are a couple of files that you don't want to be version controlled
- E.g
  - .DS_Store
  - Node_modules
  - Build
  - logs

Ignoring Files

- Create an empty file in your favorite editor and save it as ".gitignore" in your project's root folder.
- You can define rules in ".gitignore" file to ignore files
- Add file or directory path or extension or name

Ignoring Files -- Examples

- Ignore one specific file
  - path/to/file.ext
- Ignore all files with a certain name (anywhere in the project)
  - filename.ext
- Ignore all files of a certain type (anywhere in the project)
  - *.ext
- Ignore all files in a certain folder:
  - path/to/folder/*


*.gitignore - Notepad
File Edit Format View Help
*.txt
*.doc
*.js
hello.js
Ln 7, Col 9
100%
Windows (CRLF)
UTF-8

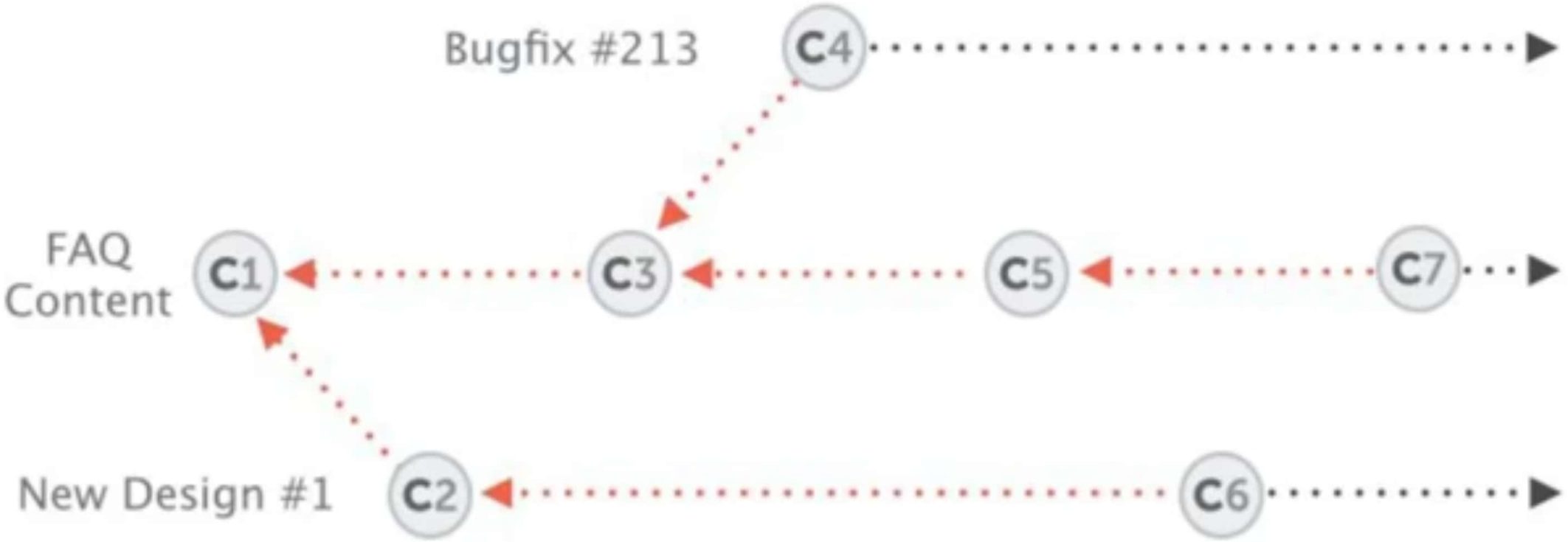
Why branches are important

- In real-world projects, work always happens in multiple of these contexts in parallel:
  - While you're preparing two new variations of your website's design (context 1 & 2)
  - you're also trying to fix an annoying bug (context 3).
  - On the side, you also update some content on your FAQ pages (context 4)
  - one of your teammates is working on a new feature for your shopping cart (context 5)
  - and another colleague is experimenting with a whole new login functionality (context 6).

Branches to the Rescue

- Branches are what we need to solve context problems.
- Because a branch represents exactly such a context in a project and helps you keep it separate from all other contexts.

Branches to the Rescue



Branches to the Rescue

- All the changes you make at any time will only apply to the currently active branch; all other branches are left untouched
- This gives you the freedom to both work on different things in parallel and, above all, to experiment - because you can‘t mess up!
- In case things go wrong you can always go back / undo / start fresh / switch contexts

Working with Branches

- Without knowing, we were already working on a branch.
- This is because branches aren't optional in Git: you are always working on a certain branch (the currently active, or "checked out", or "HEAD" branch).
- Check 'git status' and it will show you current branch
- The "master" branch was created by Git automatically for us when we started the project.

Branch Commands

- git branch
  - Show list of branches
- git branch -v
  - Show list of branches with some details
- git branch new-dev
  - Creates new branch with name 'new-dev'
- Git checkout new-dev
  - Switch to 'new-dev' branch

Branch Commands

- git merge new-dev
  - Merge ‘new-dev’ branch into current active branch
- git log new-dev..master
  - Show commit difference in two branches

Remote Repositories

- About 90% of version control related work happens in the local repository: staging, committing, viewing the status or the log/history, etc.
- If you're the only person working on your project, chances are you'll never need to set up a remote repository.
- Only when it comes to sharing data with your teammates, a remote repo comes into play.

- Github is online service for git to create remote repositories on github's server and collaborate with teammates/colleague

Work on remote repository

- After clone, make changes to your project on your local machine
- Add file or change a file
- Add file to staging area by git add .
- Commit file git commit -m "updated feature"
- Run command "git push" to push your change to remote repository on server

Commands to interact remote repository

- git push
  - Push changes to remote repository
- git fetch
  - Fetch changes from remote repository
- git merge
  - Merge changes that was fetch by 'git fetch' command

Commands to interact remote repository

- git pull
  - ➢ Fetch and merge changes from remote repository
- git remote -v
  - ➢ Show remote urls
- git remote show origin
  - ➢ Show details of origin

Commands to interact remote repository

- "git remote add myremote https://github.com/zeeshanhanif/MyProject.git"

This command will add remote repo in local repository

Publish a local repository to GitHub

- If you already have local repository on your machine and want to connect it with remote repository then:
  - Create repository on github
  - Open terminal in your local repository
  - Run following command
  - git remote add origin https://github.com/zeeshanhanif/MyProject.git
  - git push -u origin master

Publish a local repository to GitHub

- git push -u origin master
  - The "-u" flag establishes a tracking connection between remote and our local

Commands to interact remote repository

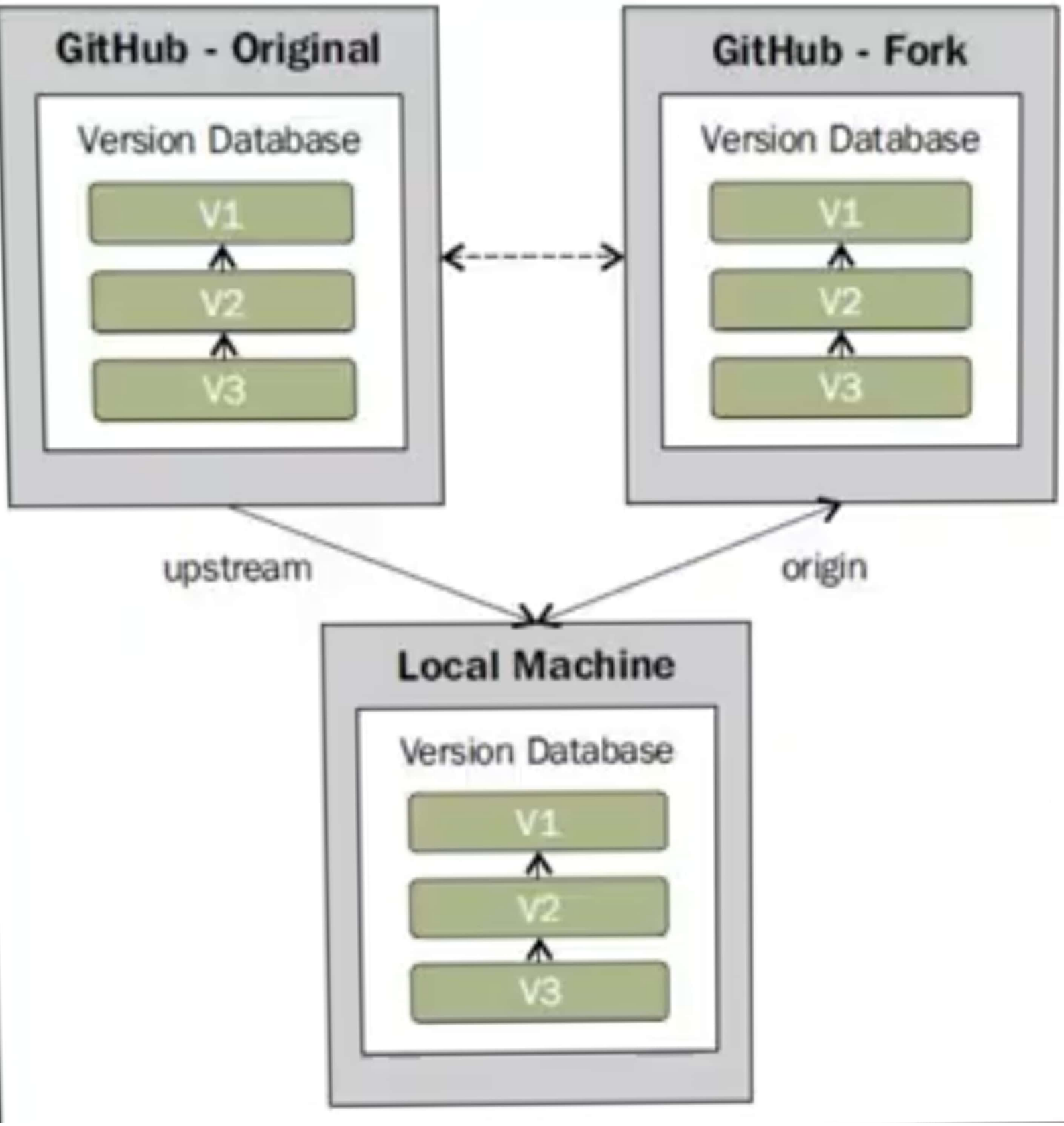
- git push 'remote' 'branch'
  - git push origin master
- Git push command require which remote repository you want to push and in which branch of remote repository

Commands to interact remote repository

- git log
  - Show logs for current branch
- git log remote/branch
  - git log origin/master
- In git log command you specify remote repository and branch so it will show logs from that branch of remote.

Social Coding – Forking a repository

- First, remember that fork is not a Git feature, but a GitHub invention
- When you fork on GitHub, you get a server-side clone of the repository on your GitHub account


GitHub - Original
Version Database
V1
V2
V3
GitHub - Fork
Version Database
V1
V2
V3
upstream
origin
Local Machine
Version Database
V1
V2
V3

Deleting Branches

- When you are done with a branch and it is no longer needed then you can delete the branch
  - git branch -d contact-form
- Deleting remote branch, add "r" flag
  - git branch -dr origin/contact-form

```
C:\Repo\thriddemo>git branch -dr origin/test
Deleted remote-tracking branch origin/test (was 5dbcc9f).

C:\Repo\thriddemo>git status
On branch master
Your branch is up to date with 'origin/master'.

nothing to commit, working tree clean

C:\Repo\thriddemo>git pull
From https://github.com/zeeshanhanif/thriddemo
 * [new branch]      test       -> origin/test
Already up to date.

C:\Repo\thriddemo>git push origin --delete test
```

```
    added first.txt

C:\Repo\seconddemo>git remote -v

C:\Repo\seconddemo>git remote add hello https://github.com/zeeshanhanif/secondproject.git

C:\Repo\seconddemo>git status
On branch master
nothing to commit, working tree clean

C:\Repo\seconddemo>
```

```
Date:   Sat Sep 14 14:25:12 2019 +0500

    updated first.txt and added second.txt

commit 88642f84f1c411a62b5af8c73ad741b5e1cc95d3
Author: zeeshanhanif <zeeshanhanif@gmail.com>
Date:   Sat Sep 14 14:24:20 2019 +0500

    added first.txt

C:\Repo\seconddemo>
C:\Repo\seconddemo>git remote -v
hello   https://github.com/zeeshanhanif/secondproject.git (fetch)
hello   https://github.com/zeeshanhanif/secondproject.git (push)

C:\Repo\seconddemo>git push -u hello dev
```