Laboratory I: Git

NAZARBAYEV UNIVERSITY

TACTILE LAB
NAZARBAYEV UNIVERSITY

Background

- Why Git?
- Git is the most popular Distributed Version Control System (DVCS)
- Every file, branch, and iteration of a project is fully accessible through DVCS, and every user has access to a complete and self-contained history of all changes.
- Team members can maintain alignment while working separately by viewing a visible history of modifications, who made them, and how they contributed to the progress of a project.

NAZARBAYEV UNIVERSITY

TACTILE LAB NAZARBAYEV UNIVERSITY

Git basics

- **Repository** - includes all of the files and folders connected to a project, as well as each file's revision history.
- **Branch** - deviation from the primary development path. It is a technique for updating the code without affecting the main version.
- **Merge** - a source branch's contents are combined with a target branch's. Only the target branch is altered throughout this operation.
- **Fork** - the repository's copy. You can freely experiment with modifications by forking a repository without impacting the original project.

NAZARBAYEV UNIVERSITY

TACTILE LAB NAZARBAYEV UNIVERSITY

Git spreadsheet

- **git init -** starts tracking an existing directory and creates a new Git repository. It alters the existing directory by adding a hidden subfolder that contains the internal data structure needed for version control
- **git clone -** enables the creation of a local clone of a remote project. The project's files, history, and branches are all included in the clone.

Git spreadsheet

- **git add -** stages a transition. Git tracks changes made to a developer's codebase, but in order for the changes to be recorded in the project's history, they must first be staged and captured in a snapshot. The first step of a two-step procedure, staging, is carried out by this command. Any staged modifications will be recorded in the project's history and the following snapshot. Developers can fully manage the history of their project by staging and committing separately without affecting the way they code.

NAZARBAYEV UNIVERSITY

TACTILE LAB
NAZARBAYEV UNIVERSITY

Git spreadsheet

- **git commit -** completes the change-tracking process by saving the snapshot to the project history. In essence, committing works like taking a picture. With git commit, anything that was staged with git add will be included in the snapshot
- **git status -** shows the status of changes as untracked, modified, or staged.
- **git branch** - demonstrates the local branches in development.

Git spreadsheet

- **git merge -** merges different development paths. Usually, this command is used to integrate changes from two different branches. When a developer wants to push modifications from a feature branch into the main branch, for instance, they would merge
- **git pull -** updates from the remote line of development are applied to the local line of development. When a teammate commits to a branch remotely and the developer wants their local environment to reflect the changes, they use this command.

NAZARBAYEV UNIVERSITY

TACTILE LAB NAZARBAYEV UNIVERSITY

Git spreadsheet

- **git push -** updates any local branch commits to the remote repository
- **git fetch -** collects and puts in your local repository any commits from the target branch that don't already exist in your current branch. It doesn't, however, merge them into your current branch
- **git checkout -** 'checks out' of one branch of code to access another branch. Working on a new branch, an existing branch, or a remote branch requires checkout.

Git spreadsheet

- **git diff -** runs a diff functions between different data sources
- there are a lot more commands, however, the listed ones are the most popular. In order to get familiar with other commands, refer to https://git-scm.com/docs

Installing Git

- Ubuntu
  - $ sudo apt install git
- macOS
  - $ git –version
  - If git is not installed it will prompt you to install it

Examples

- First of all, to work with any remote repository we need a Git hosting service: GitHub, BitBucket, GitLab
- GitHub is the most used one, so if you don’t have an account there, sign up first
- After that, create a new repository

Example

- Now, let's create a local project that we want to connect with our new repository
  - **$ mkdir project/**
  - **$ cd project**
  - **$ echo "My name is $YOUR_NAME$. This is my repository" >> README.md**
  - **$ git init**
  - **$ git add . (dot stands for all content within the current directory)**

Example

- So, with **git add** we stored a snapshot in a temporary staging area which is called “index”. We can make it permanently store the content of index in the repository by committing it
  - **$ git commit –m “Adding a ReadMe” (message is obligatory for understanding what this commit is about)**
- However, new versions require setting account default identity
  - **git config --global user.email “your@email”**
  - **git config --global user.name “yourname”**
- Congratulations, the first version of your project is now saved in Git!

NAZARBAYEV UNIVERSITY

TACTILE LAB NAZARBAYEV UNIVERSITY

Example

- Let’s upload our local repository to the one we created in GitHub!
- Let’s make the current branch as Master
  - **git branch –M main**
- Finally, connect your local repository with the remote one
  - **git remote add origin <repository_url>**
- Update the remote repository with our created content
  - **git push –u origin main**

NAZARBAYEV UNIVERSITY

TACTILE LAB
NAZARBAYEV UNIVERSITY

Example

- From recent updates, now you need to generate a token instead of typing your password
  - https://github.com/settings/tokens
- Okay, after all of these steps, your remote repository is updated with your local content!
- Every collaborator can then download it via
  - **git clone <repository_url>** or just
  - **git pull**

Example

- What if we want to make changes/contribute to an existing repository?
  - **git clone <repository_url>** download it
  - **cd <dir_name>** move to the local folder
  - **git branch new-branch** create new branch
  - **git checkout new-branch** switch to the new branch
  - **echo “File #1” >> file1.txt** creating some new files
  - **echo “File #2” >> file2.txt** creating some new files

Example

- Continue
  - **git add <file_name> <file_name2>** stage your changes
  - **git commit -m “Adding new files”** take a snapshot
  - **git push --set-upstream origin new-branch** push changes
  - When you push to a remote and you use the –set-upstream flag git sets the branch you are pushing to as the remote tracking branch of the branch you are pushing.

Example

- We can also merge our branch with the main one
  - **git checkout main**
  - **git merge new-branch**
  - **git push**
- When you work with several collaborators, a good thing would be to set some workflow and not mess up everybody works by random merges. Read this guideline as a homework
  - https://www.atlassian.com/git/tutorials/comparing-workflows

Thank you for your attention!

NAZARBAYEV UNIVERSITY

TACTILE LAB NAZARBAYEV UNIVERSITY