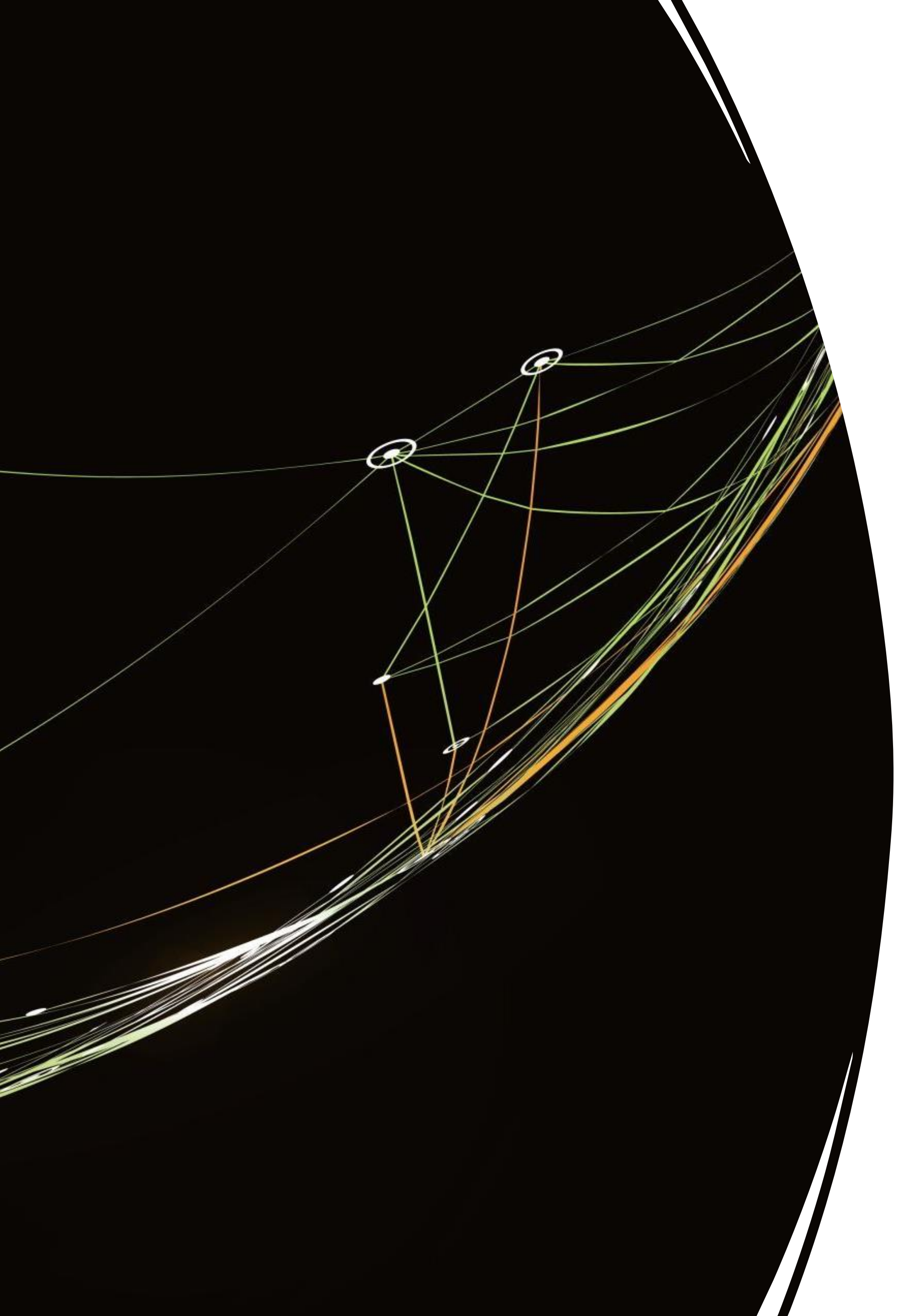

GIT ТА ЙОГО КОМАНДИ

КИЇВСЬКИЙ ФАХОВИЙ КОЛЕДЖ ЗВ'ЯЗКУ

ПРЕЗЕНТАЦІЯ З ДИСЦИПЛІНИ "ОПЕРАЦІЙНІ СИСТЕМИ"

НА ТЕМУ: "GIT ТА ЙОГО КОМАНДИ"
ВИКОНАЛИ СТУДЕНТИ ГРУПИ БІКС-03
КОЗИРЄВ СВЯТОСЛАВ СВЯТОСЛАВОВИЧ

ТА

ДЯЧЕНКО НИКИТА РОМАНОВИЧ

What is Git?

- Git is a distributed file versioning and collaboration system that allows you to track the history of software development and collaborate on complex projects from anywhere in the world. Because Git is distributed, it becomes possible to restore the code from a local copy if the server with the remote repository fails.

Advantages of Git over other systems

- The main advantage of Git is that it is very fast and transparent. It is convenient for non-linear development and is effective for both small projects and large systems with thousands of participants.
- Unlike Perforce, CVS, and others, Git stores snapshots of repositories rather than file change lists, and as a result, it runs much faster.

History of Git

- The project was created by Linus Torvalds to manage linux kernel development, and today is supported by Junio Hamano. Git is one of the most efficient, reliable, and high-performance version control systems, providing flexible, nonlinear development tools based on branching and merging. To ensure the integrity of history and resistance to change, cryptographic methods are retroactively used, and it is also possible to bind the digital signatures of developers to tags and commits.

How Git works

- The system is designed as a set of programs specially designed taking into account their use in scripts. This allows you to conveniently create specialized Git-based version control systems or user interfaces. Git is like a kind of file system with tools that work on top of it. For each file you're tracking, Git preserves the size, creation time, and last modified. This data is stored in an index file located in the .git folder The entire Git database is stored in a folder called .git In Git, files can be in one of 3 states: fixed, modified, and provisioned.

Local operations

- Most of the actions can be performed on the local file system without using an Internet connection. The entire history of changes is saved locally and, if necessary, uploaded to a remote repository. Unlike Subversion, where without an Internet connection you can only edit files, but it is impossible to save changes to your database (since it is disconnected from the repository). Any commit is first made locally and then uploaded to the remote repository.

Data integrity

- In its base, Git stores everything by hashes of files that are hashed by the SHA-1 function. Before you save the files, Git calculates the file's SHA-1 hash and the resulting hash becomes the file index in Git. Using a hash, Git easily tracks changes to files.

Branching (branches)

- Branching is a distinction from the main line of development. Git lets you create multiple branches and switch between them. This is useful because it allows several developers to work on their functionality without interfering with others or spoiling the main branch. By default, Git creates a branch called master. A branch in Git is simply a pointer to one of the fixes. Each time you commit again, a branch in Git moves automatically. A branch is a simple file that contains 40 characters of the SHA-1 commit checksum. Creating a new branch is very fast because it is the same as writing to a 41-byte file.

Merge and rebase data

- Git supports two ways to integrate changes from branch to branch: merge and rebase. The main difference is that rebase remembers fixes in the form of patches, rewinds the branch and applies patches in the form of fixations, unlike merge, which merges two branches into one.

Projects where Git is used

- Examples of projects using Git include the Linux kernel, Android, LibreOffice, Cairo, GNU Core Utilities, Mesa 3D, Wine, many projects from X.org, XMMS2[ru], GStreamer, Debian DragonFly BSD, Perl, Eclipse, GNOME, KDE, Qt, Ruby on Rails, PostgreSQL, VideoLAN, PHP, One Laptop Per Child (OLPC), Koha ABIS, GNU LilyPond, and ELinks, and some GNU/Linux distributions

Git Core Commands

- git add
- git status
- git diff
- git difftool
- git commit

- git reset
- git rm
- git mv
- git clean

git add

- The git add command adds the contents of the staging area for the next commite. By default, git commit uses only this index, so you can use git add to compose a cast of the next commit.
- This is one of the key Git teams. Below are the most interesting ways to use this command.
- git add Use to add only certain parts of a modified file to the index.

git status

- The git status command shows the status of files in the working directory and index: which files have been modified but not added to the index; who are waiting for a commit in the index. In addition to this, hints are displayed on how to change the status of files.

git diff

- The git diff command is used to calculate the difference between any two Git trees. This could be the difference between your working copy and the index (git diff proper), the difference between the index and the last commit (git diff --staged), or between any two commits (git diff master branchB).

- The --check argument can be used to check for problems with spaces.

- You can compare branches using git diff A... B

- Using the -w option to hide differences in space characters. It is possible to compare conflicting changes with options by using --theirs, --ours, and --base.

- Using this command with the --submodule option is possible to compare changes in submodules.

git difftool

- The git difftool command simply runs an external comparison utility to show the differences in the two trees, in case you want to use something different from the built-in git diff viewer.

git commit

- The git commit command takes all the data added to the index using git add and stores its cast in the internal database, and then shifts the index of the current branch to that cast.
- There's also the use of -a options to add all changes to the index without using git add, which can be useful in everyday use, and -m to pass a commit message without running a full-fledged editor.
- The --asend option is used to change the last commit committed.
- You can commit commits using the -S option

git reset

- The git reset command, as you might guess from the name, is mainly used to undo changes. It changes the HEAD index, and optionally, the state of the index. Also, this command can change the files in the working directory when using the --hard parameter, which can lead to loss of developments if used improperly, so make sure that your intentions are serious before using it.
- Git reset is also used to remove a file from the index added there using git add.
- git reset –hard is used to undo the merge. Using the git merge –abort command also uses a git for this purpose, which works as a wrapper on git reset.

git rm

- The git rm command is used by Git to remove files from the index and a working copy. It's similar to git add with the only exception that it deletes rather than adds files for the next commit.
- Delete files from the working directory and index and only from the index, possibly using the --cached flag.
- Another use case for git rm becomes possible if you use the --ignore-unmatch option when running git filter-branch, which suppresses deleting errors in deleting non-existent files. This can be useful for scripts that are automatically executed.

git mv

- The git mv command is just a convenient way to move the file, and then run git add for the new file and git rm for the old one.

git clean

- The git clean command is used to remove garbage from the working directory. These can be the results of a project or mail merge conflict files.

Thank you for your attention!