GIT

Git Advanced Commands

git commit -a	Stages files automatically
git log -p	Produces patch text
git show	Shows various objects
git diff	Is similar to the Linux `diff` command, and can show the differences in various commits

Git Advanced Commands

git diff --staged	An alias to --cached, this will show all staged files compared to the named commit
git add -p	Allows a user to interactively review patches to add to the current commit
git mv	Similar to the Linux `mv` command, this moves a file
git rm	Similar to the Linux `rm` command, this deletes, or removes a file

Git Revert Commands

- git checkout is effectively used to switch branches.
- git reset basically resets the repo, throwing away some changes. It's somewhat difficult to understand, so reading the examples in the documentation may be a bit more useful.
- git commit --amend is used to make changes to commits after-the-fact, which can be useful for making notes about a given commit.
- git revert makes a new commit which effectively rolls back a previous commit. It's a bit like an undo command.

git branch	Used to manage branches
git branch <name>	Creates the branch
git branch -d <name>	Deletes the branch
git branch -D <name>	Forcibly deletes the branch
git checkout <branch>	Switches to a branch.

git checkout -b <branch>	Creates a new branch and switches to it.
git merge <branch>	Merge joins branches together.
git merge --abort	If there are merge conflicts (meaning files are incompatible), --abort can be used to abort the merge action.
git log --graph --oneline	This shows a summarized view of the commit history for a repo.

Basic Interaction With Github

- There are various remote repository hosting sites:
- GitHub
- BitBucket
- Gitlab.

- Set up a free account, username, and password.

Github Workflow

The GitHub flow is a lightweight, branch-based workflow built around core Git commands used by teams around the globe—including ours. The GitHub flow has six steps, each with distinct benefits when implemented:

1.Create a branch: Topic branches created from the canonical deployment branch (usually main) allow teams to contribute to many parallel efforts. Short-lived topic branches, in particular, keep teams focused and results in quick ships.

2.Add commits: Snapshots of development efforts within a branch create safe, revertible points in the project's history.

3.Open a pull request: Pull requests publicize a project's ongoing efforts and set the tone for a transparent development process.

Github Workflow

4.Discuss and review code: Teams participate in code reviews by commenting, testing, and reviewing open pull requests. Code review is at the core of an open and participatory culture.

5.Merge: Upon clicking merge, GitHub automatically performs the equivalent of a local ‘git merge’ operation. GitHub also keeps the entire branch development history on the merged pull request.

6.Deploy: Teams can choose the best release cycles or incorporate continuous integration tools and operate with the assurance that code on the deployment branch has gone through a robust workflow.

Steps to Add a New Repo

1. In the upper-right corner of any page, use the drop-down menu, and select **New repository**.
2. Type a short, memorable name for your repository. For example, "hello-world".
3. Optionally, add a description of your repository. For example, "My first repository on GitHub.“
4. Choose a repository visibility.
5. Select **Initialize this repository with a README**.
6. Click **Create repository**.

Commit your first change

- A commit is like a snapshot of all the files in your project at a particular point in time.

1. In your repository's list of files, click ***README.md***.
2. Above the file's content, click on edit marker.
3. On the **Edit file** tab, type some information about yourself.
4. Above the new content, click **Preview changes**.
5. Review the changes you made to the file. You'll see the new content in green.

Commit your first change

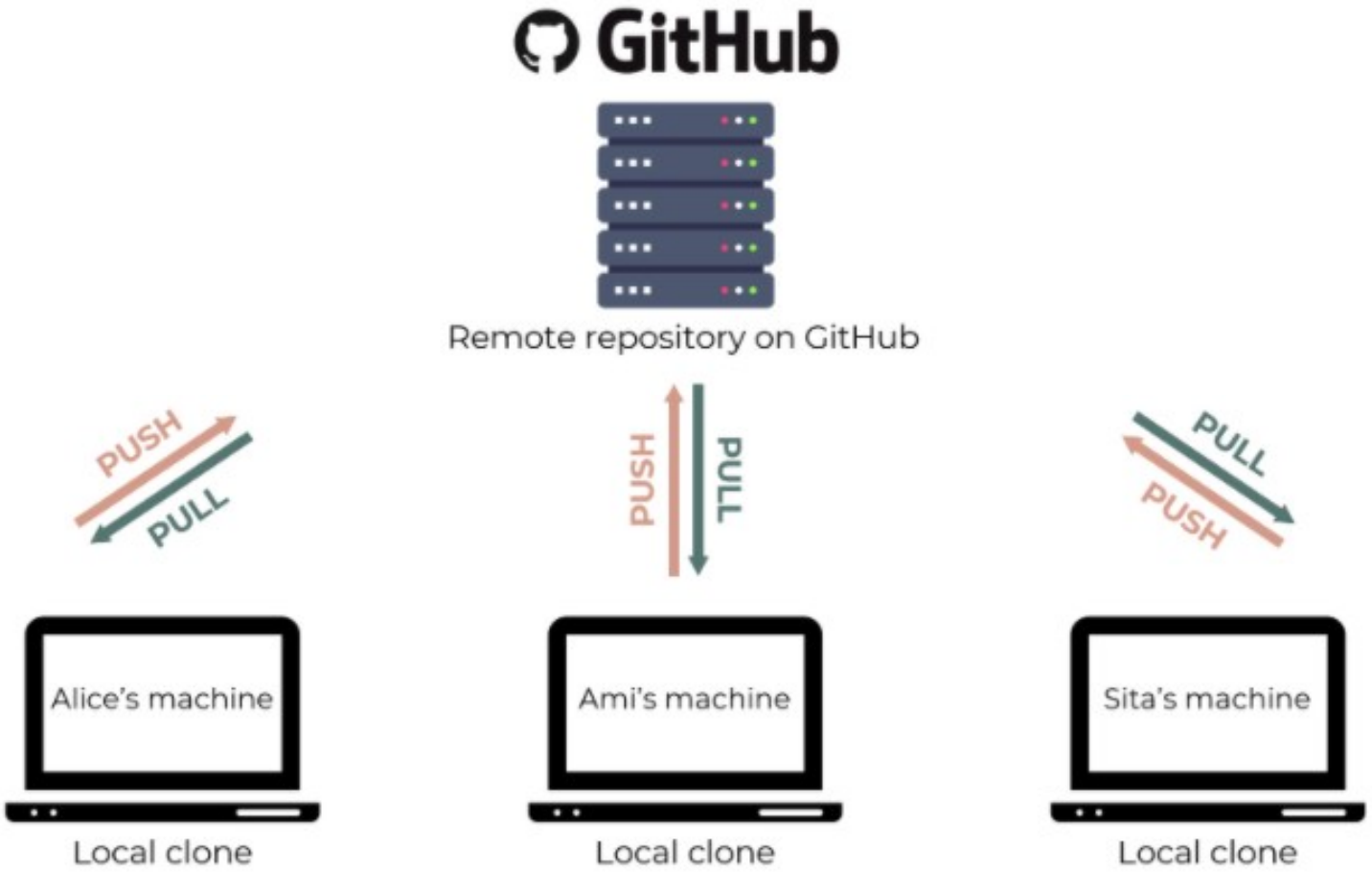
6. At the bottom of the page, type a short, meaningful commit message that describes the change you made to the file.
7. Below the commit message fields, decide whether to add your commit to the current branch or to a new branch. If your current branch is the default branch, you should choose to create a new branch for your commit and then create a pull request.
8. Click **Propose file change**.

Useful commands used with Github

git clone URL	Git clone is used to clone a remote repository into a local workspace
git push	Git push is used to push commits from your local repo to a remote repo
git pull	Git pull is used to fetch the newest updates from a remote repository

Remote Repository

- When we clone the newly created GitHub repository, we had our local Git Repo to interact with a remote repository.
- A remote in Git is a common repository that all team members use to exchange their changes. In most cases, such a remote repository is stored on a code hosting service like GitHub or on an internal server.


GitHub
Remote repository on GitHub
PUSH
PULL
PUSH
PULL
PULL
PUSH
Alice's machine
Ami's machine
Sita's machine
Local clone
Local clone
Local clone

GitHub Remote Commands

git remote	Lists remote repos
git remote -v	List remote repos verbosely
git remote show <name>	Describes a single remote repo
git remote update	Fetches the most up-to-date objects
git fetch	Downloads specific objects
git branch -r	Lists remote branches; can be combined with other branch arguments to manage remote branches

Contribute to an existing repository

```
# download a repository on GitHub.com to our machine
• git clone https://github.com/me/repo.git

# change into the `repo` directory
• cd repo

# create a new branch to store any new changes
• git branch my-branch

# switch to that branch (line of development)
• git checkout my-branch
```

Contribute to an existing repository

```
# make changes, for example, edit `file1.md` and `file2.md` using
the text editor
```

```
# stage the changed files
```

- `git add file1.md file2.md`

```
# take a snapshot of the staging area (anything that's been added)
```

- `git commit -m "my snapshot"`

```
# push changes to github
```

- `git push --set-upstream origin my-branch`

Start a new repository and publish it to GitHub

```
# create a new directory, and initialize it with git-specific functions
git init my-repo

# change into the `my-repo` directory
cd my-repo

# create the first file in the project
touch README.md

# git isn't aware of the file, stage it
git add README.md
```

Start a new repository and publish it to GitHub

```
# take a snapshot of the staging area
git commit -m "add README to initial commit"

# provide the path for the repository you created on github
git remote add origin https://github.com/YOUR-USERNAME/YOUR-REPOSITORY.git

# push changes to github
git push --set-upstream origin main
```

Contribute to an existing branch on GitHub

```
# assumption: a project called `repo` already exists on the machine, and a new branch has been pushed to GitHub.com since the last time changes were made locally


# change into the `repo` directory

cd repo


# update all remote tracking branches, and the currently checked out branch

git pull


# change into the existing branch called `feature-a`

git checkout feature-a
```

Contribute to an existing branch on GitHub

```
# make changes, for example, edit `file1.md` using the text editor

# stage the changed file
git add file1.md

# take a snapshot of the staging area
git commit -m "edit file1"

# push changes to github
git push
```

Models for collaborative development

- There are two primary ways people collaborate on GitHub:
  - Shared repository - With a *shared repository*, individuals and teams are explicitly designated as contributors with read, write, or administrator access.
  - Fork and pull - For an open source project, or for projects to which anyone can contribute, managing individual permissions can be challenging, but a *fork and pull* model allows anyone who can view the project to contribute

Some Resources

- The GitHub team has created a library of educational videos and guides to help users continue to develop their skills and build better software.

- Beginner projects to explore
- GitHub video guides
- GitHub on-demand training
- GitHub training guides
- GitHub training resources