

# GIT basics

Jurgis Šidlauskas, Software Engineer

# Agenda

## 1. Git essentials

### 1.1 Remote

## 2. Extra content

### 2.1 Branching

### 2.2 Undoing

## 3. Bonus practice

The background of the slide is a close-up photograph of green oak leaves. The leaves are vibrant green with visible veins and serrated edges. They are arranged in a way that creates a dense, textured background. A large, white rectangular box is centered on the slide, serving as a backdrop for the title text.

# 1. Git essentials

# What is git?

- Distributed version control
- Users keep entire code and history on their location machines
  - Users can make any changes without internet access
  - (Except pushing and pulling changes from a remote server)

<https://git-scm.com/docs/>

# Configure Git

**git config --global user.name "[name]"**

- Sets the name you want attached to your commit transactions

**git config --global user.email "[email address]"**

- Sets the email you want attached to your commit transactions

# Create Repositories

## **git init [project-name]**

- Creates a new local repository with the specified name

## **git clone [url]**

- Downloads a project and its entire version history

# Check changes

## **git status**

- Lists all new or modified files to be committed

## **git log**

- Show commit logs

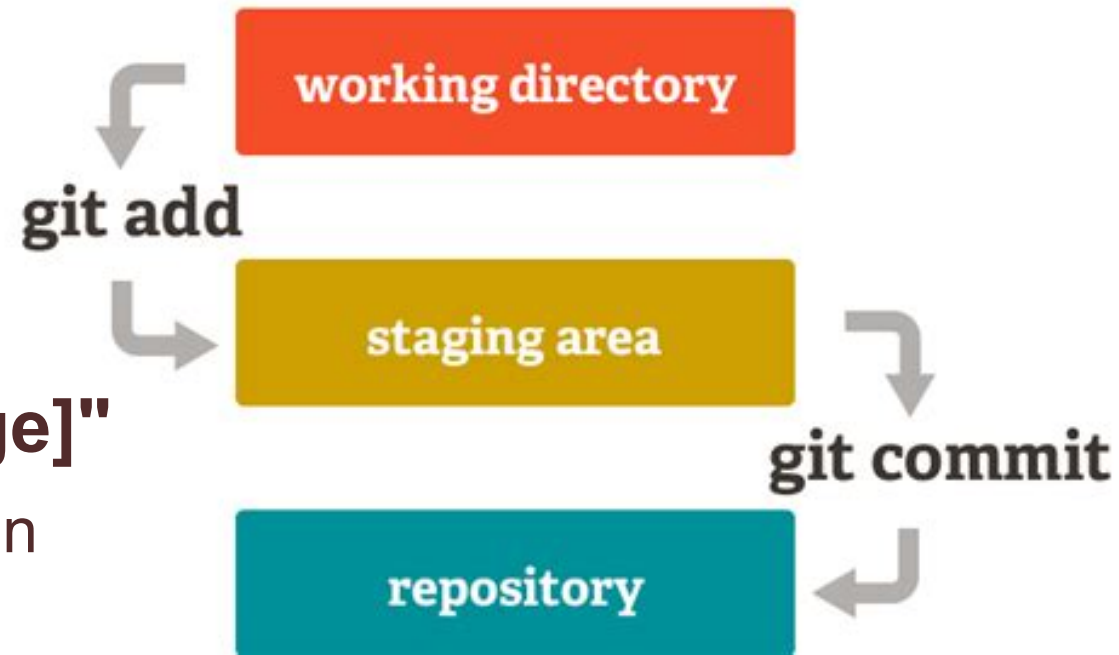
# Make changes

## **git add [file]**

- Snapshots the file in preparation for versioning

## **git commit -m "[descriptive message]"**

- Records the file snapshots permanently in version history





The background of the slide is a close-up photograph of green oak leaves. The leaves are vibrant green with prominent veins and characteristic lobed edges. They are arranged in a dense, overlapping pattern, filling the entire frame. A large, solid white rectangle is centered on the slide, serving as a backdrop for the title text.

# Practice

# Install git

- **Linux**

`sudo apt-get install git`

`sudo yum install git`

- **Mac**

<http://git-scm.com/download/mac>

- **Windows**

<http://git-scm.com/download/win>

# Task #1

- Initialize a new git project
- Create a new file “readme.txt”
- Commit changes to local repo

The background of the slide is a close-up photograph of green oak leaves. The leaves are vibrant green with prominent veins and characteristic lobed edges. They are arranged in a dense, overlapping pattern, filling the entire frame. A large, solid white rectangle is centered on the slide, serving as a backdrop for the title text.

# 1.1. Remote

# Synchronize Changes

## **git remote add [remote-name] [url]**

- Add a new remote git repository as a shortname

## **git remote -v**

- Lists all remote git repositories

# Synchronize Changes

## **git push [alias] [branch]**

- Uploads all local branch commits to remote repo branch

## **git push -u [alias] [branch]**

- Sets upstream and uploads all local branch commits to remote repo branch (used only first time pushing a new branch)

## **git pull**

- Fetches and merges any commits from the tracking remote branch

# Synchronize Changes

## **git fetch**

- Download objects and refs from another repository

## **git pull**

- Fetches and merges any commits from the tracking remote branch



The background of the slide is a close-up photograph of green oak leaves. The leaves are vibrant green with visible veins and serrated edges. They are arranged in a way that creates a dense, textured background. A large, solid white rectangle is centered on the slide, serving as a backdrop for the text.

# Practice



# Create GitHub account

- [www.github.com](https://www.github.com)

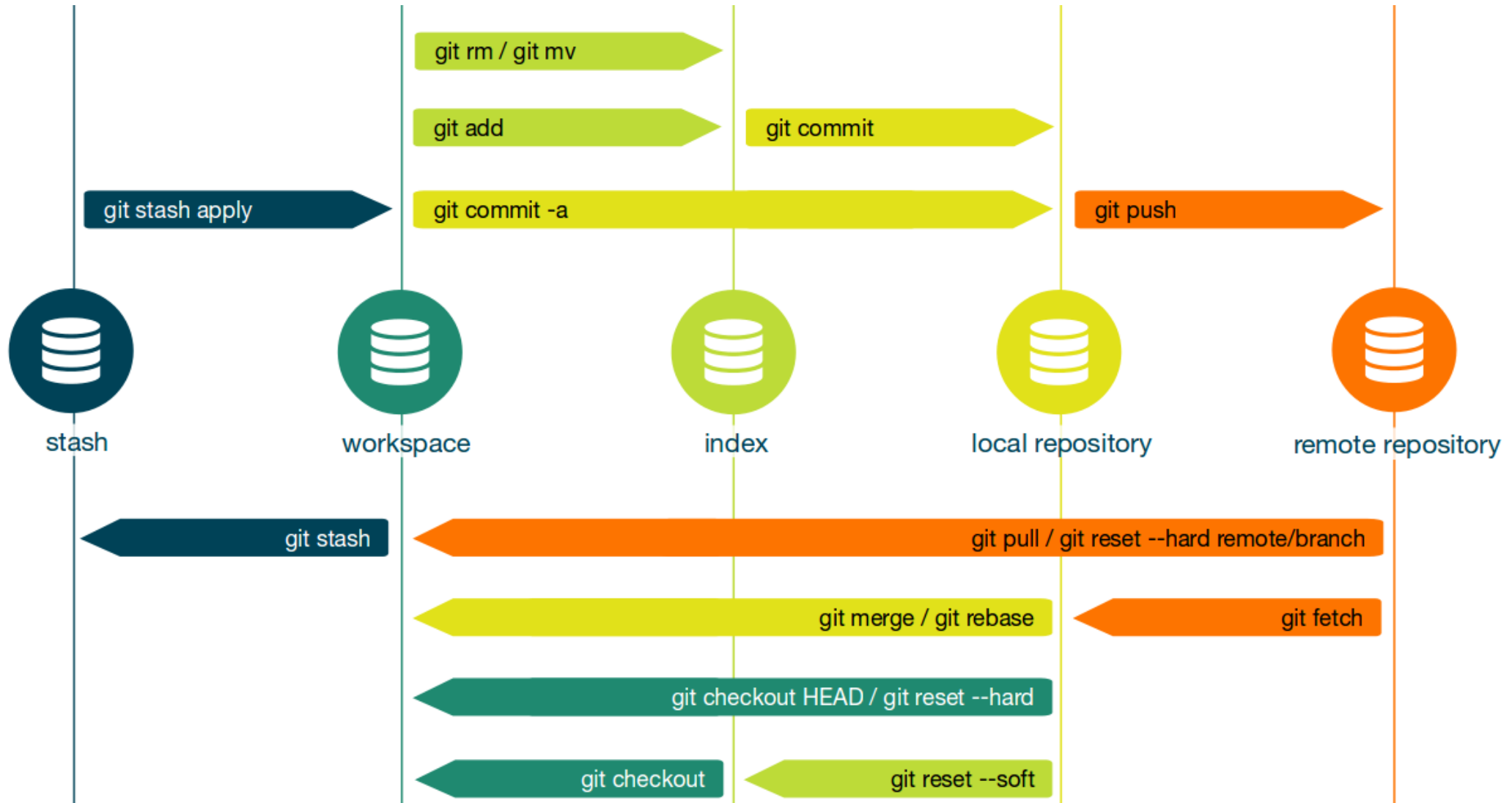
## Task #2

- Create a new repository in GitHub
- Setup remote in your local git project
- Push changes to GitHub

The background of the slide is a close-up photograph of green oak leaves. The leaves are vibrant green with visible veins and serrated edges. They are arranged in a way that creates a dense, textured background. A large, white rectangular box is centered on the slide, containing the text.

## 2. Extra content

# Git workflow



# .gitignore

- .gitignore – file, that tells git which files (or patterns) it should ignore. It's usually used to avoid committing transient files from your working directory that aren't useful to other collaborators, such as compilation products, temporary files IDEs create, etc.

# Configure Tooling

**git config --add --system core.editor notepad**

– sets notepad as a default editor (directed to windows users)

# Semantic Commit Messages

- feat: (new feature for the user, not a new feature for build script)
- fix: (bug fix for the user, not a fix to a build script)
- docs: (changes to the documentation)
- style: (formatting, missing semi colons, etc; no production code change)
- refactor: (refactoring production code, eg. renaming a variable)
- test: (adding missing tests, refactoring tests; no production code change)
- chore: (updating grunt tasks etc; no production code change)

The background of the slide is a close-up photograph of green oak leaves. The leaves are vibrant green with prominent veins and characteristic lobed edges. They are arranged in a dense, overlapping pattern, filling the entire frame. A large, solid white rectangle is centered on the slide, serving as a backdrop for the title text.

# Practice



## Task #3

- Create a new directory “unwanted”
- Create a new directory “important”
- Create new files in each directories “file.txt”
- Ignore “unwanted” dir
- Commit changes to local repo

The background of the slide is a close-up photograph of green oak leaves. The leaves are vibrant green with prominent veins and characteristic lobed edges. They are arranged in a dense, overlapping pattern, with some leaves in sharp focus and others blurred in the background, creating a sense of depth. A large, solid white rectangular area is centered on the slide, serving as a backdrop for the title text.

## 2.1. Branching

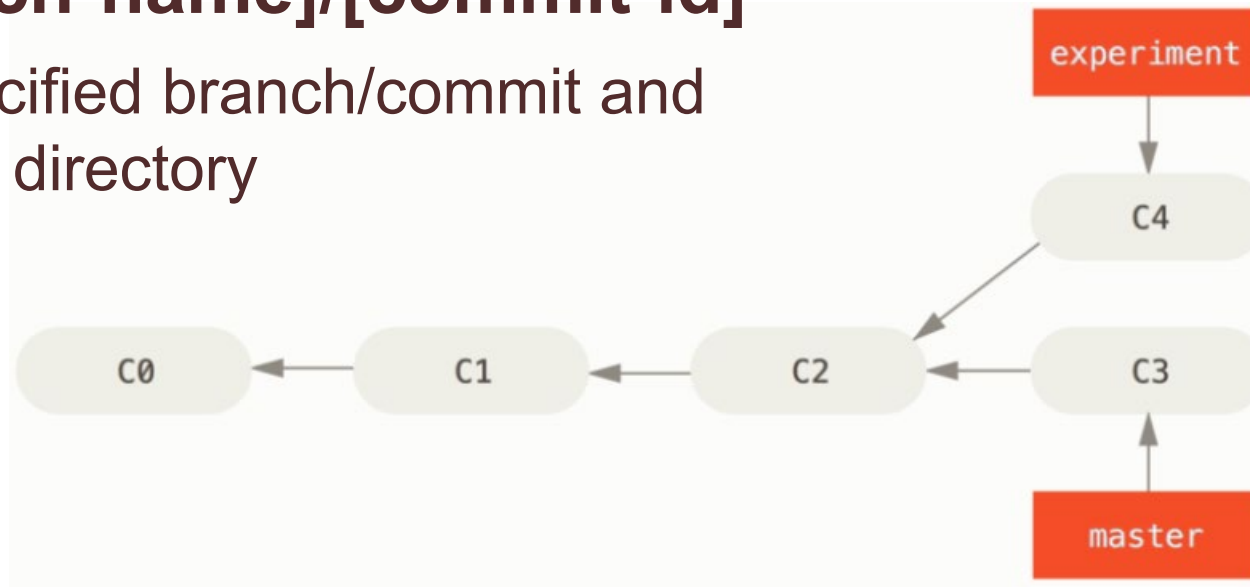
# Group Changes

## **git branch [branch-name]**

- Creates a new branch

## **git checkout [branch-name]/[commit-id]**

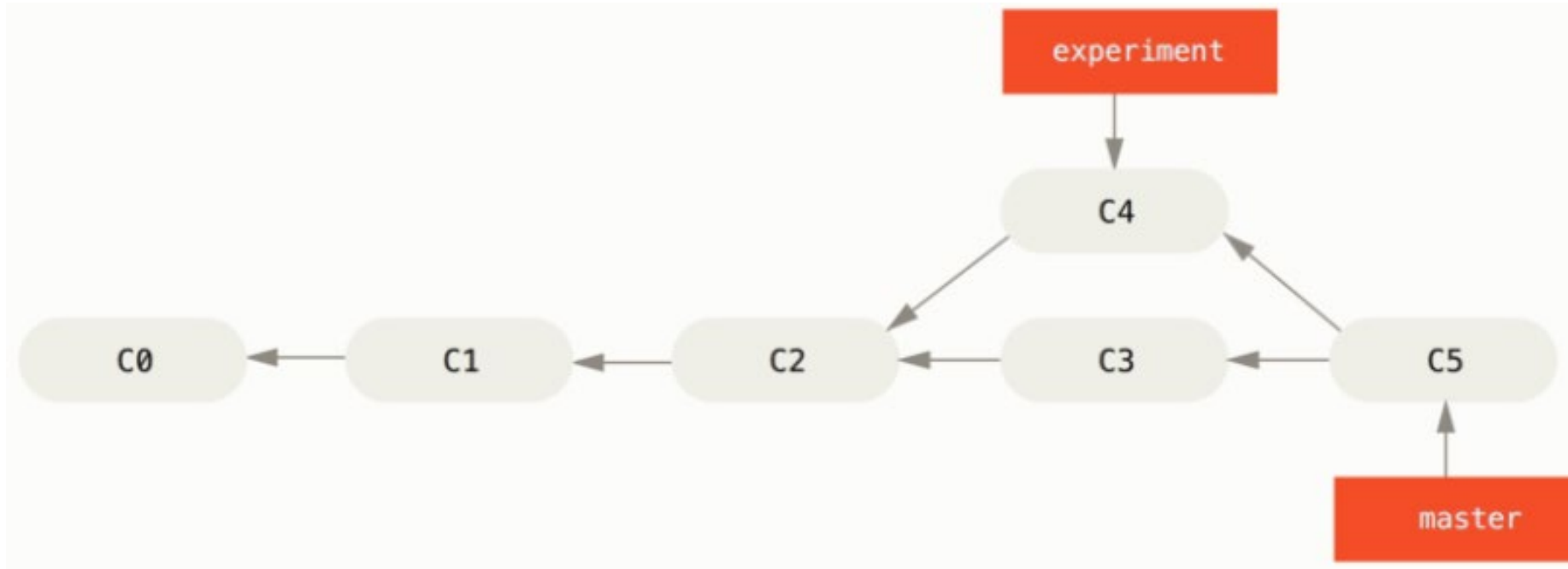
- Switches to the specified branch/commit and updates the working directory



# Group Changes

## `git merge [branch]`

- Combines the specified branch's history into the current branch



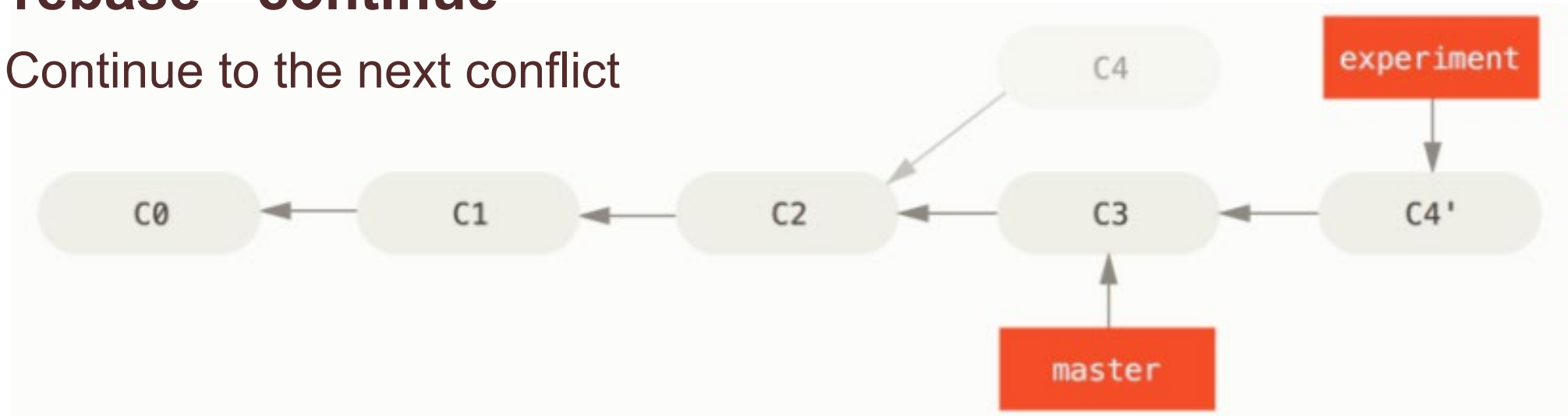
# Group Changes

## **git rebase [branch]**

- Rebases specified branch's history into the current branch

## **git rebase --continue**

- Continue to the next conflict



# Synchronize Changes

## **git merge [alias]/[branch]**

—merges a remote branch into your current branch to bring it up to date

The background of the slide is a close-up photograph of green oak leaves. The leaves are vibrant green with prominent veins and characteristic lobed edges. They are arranged in a dense, overlapping pattern, filling the entire frame. A large, solid white rectangle is centered on the slide, serving as a backdrop for the title text.

# Practice

## Task #4

- Create a new branch “feature”
- Make some changes in that branch
- Merge master with branch



## Task #5

- Create a new branch “conflicts” from master
- Create conflicting commits in master and new branch
- Merge master with branch (resolve conflict)

## Task #5 bonus

- Create another conflicting commit in master and “conflicts” branch
- Rebase master with branch

The background of the slide is a close-up photograph of green oak leaves. The leaves are vibrant green with prominent veins and characteristic lobed edges. They are arranged in a dense, overlapping pattern, filling the entire frame. A large, solid white rectangle is centered on the slide, serving as a backdrop for the title text.

## 2.2. Undoing

# Redo commits

## **git reset [commit]**

- Undoes all commits after [commit], preserving changes locally

## **git reset --hard [commit]**

- Clears staging area, rewrites working tree from specified [commit]

The background of the slide is a close-up photograph of green oak leaves. The leaves are vibrant green with visible veins and serrated edges. They are arranged in a way that creates a dense, textured background. A large, white rectangular box is centered on the slide, containing the text.

# 3. Bonus practice

## Task #6 (using IntelliJ IDE)

- Create a new “Hello world!” project in IntelliJ
- Run it!
- Init git repo
- Ignore compiled files including the directories they’re in
- Commit changes

## Task #7 (using IntelliJ IDE)

- Create a new branch “feature”
- Make some changes in that branch
- Merge master with branch

## Task #8 (using IntelliJ IDE)

- Create a new branch “conflicts” from master
- Create conflicting commits in master and new branch
- Rebase master with branch (resolve conflict)



## Task #9 (using IntelliJ IDE)

- Share project on GitHub
- Create some changes on a new branch
- Create a pull request
- Merge PR (in the browser)

## Task #10

- Clone your colleagues GitHub repo
- Make your colleague a collaborator in your project
- Create a new branch from “master” in git bash – “collaborate”
- Change something
- Push it to remote “collaborate” branch
- Create a pull request
- Review and merge the pull request in your GitHub project