

The background features several thin, light-colored lines that form abstract, angular shapes, resembling stylized mountains or architectural elements. These lines are scattered across the dark blue background, with some extending from the edges towards the center.

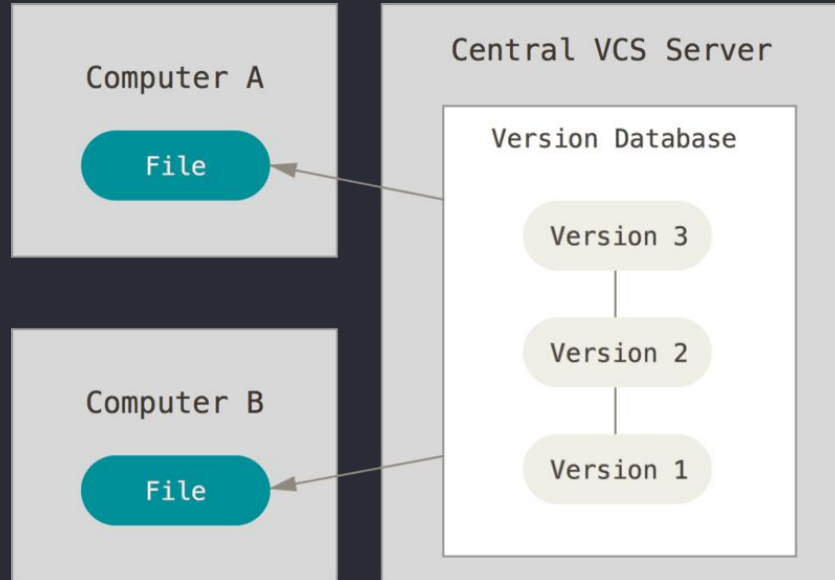
Version Control

WEEK 1

Advanced Native Mobile Programming
1604B062

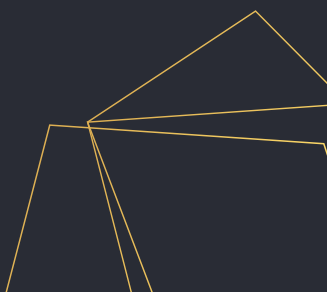
VERSION CONTROL

Version control, also known as revision control or source control, is the management of changes to documents, computer programs, large websites, and other collections of information.





VERSION CONTROL

- Track developments and changes in your files
 - Record the changes you made to your file in a way that you will be able to understand later
 - Experiment with different versions of a file while maintaining the original version
 - ‘Merge’ two versions of a file and manage conflicts between versions
 - Revert changes, moving ‘backward’ through your history to previous versions of your file
- 

TERMS

- **repository**

contain a collection of files of various different versions of a Project.

- **There are two kind of repositories:**

- Local repository
- Remote repository

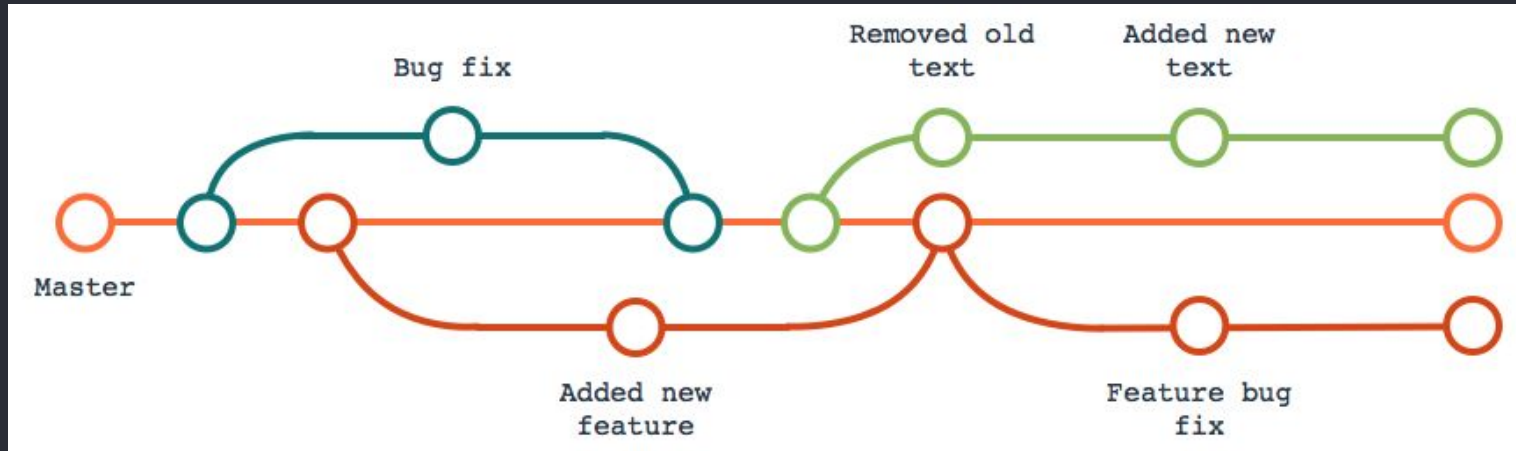
TERMS

- **clone**
Create a new repository containing the revisions from another repository.
- **commit**
To write or merge the changes made in the working copy back to the repository.
- **push**
Copy revisions from the current repository to a remote repository.
- **pull**
Copy revisions from a remote repository to the current repository.

TERMS

branch

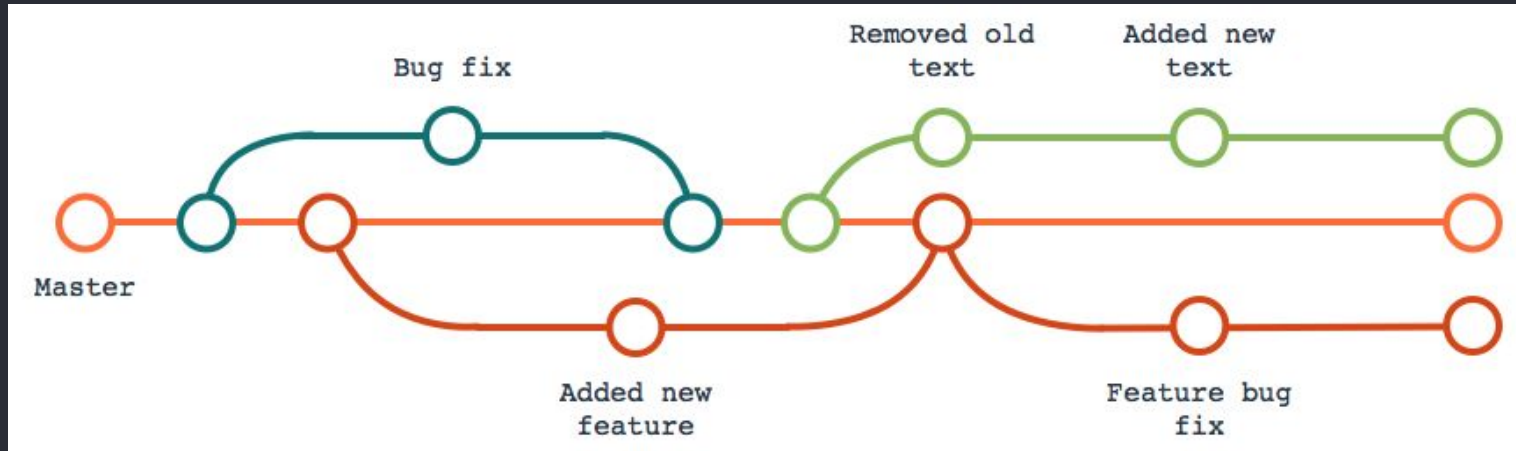
A separate working copy of files under version control which may be developed independently from the origin.



TERMS

merge

An operation in which two sets of changes are applied to a file or set of files.



VARIOUS VERSION CONTROL SYSTEM



GIT

used for source code management in software development.
Created by Linus Torvalds in 2005



MERCURIAL

written in python and intended for software developers



SUBVERSION

abbreviated as SVN.
Founded in 2000 by CollabNet, distributed as open source under the Apache License

POPULAR GIT REPOSITORY

- **github**
- **gitlab**
- **bitbucket**
- **sourceforge**

...or create a new repository on the command line

```
echo "# CoolApp" >> README.md
git init
git add README.md
git commit -m "first commit"
git remote add origin https://github.com/chieger/CoolApp.git
git push -u origin master
```



...or push an existing repository from the command line

```
git remote add origin https://github.com/chieger/CoolApp.git
git push -u origin master
```



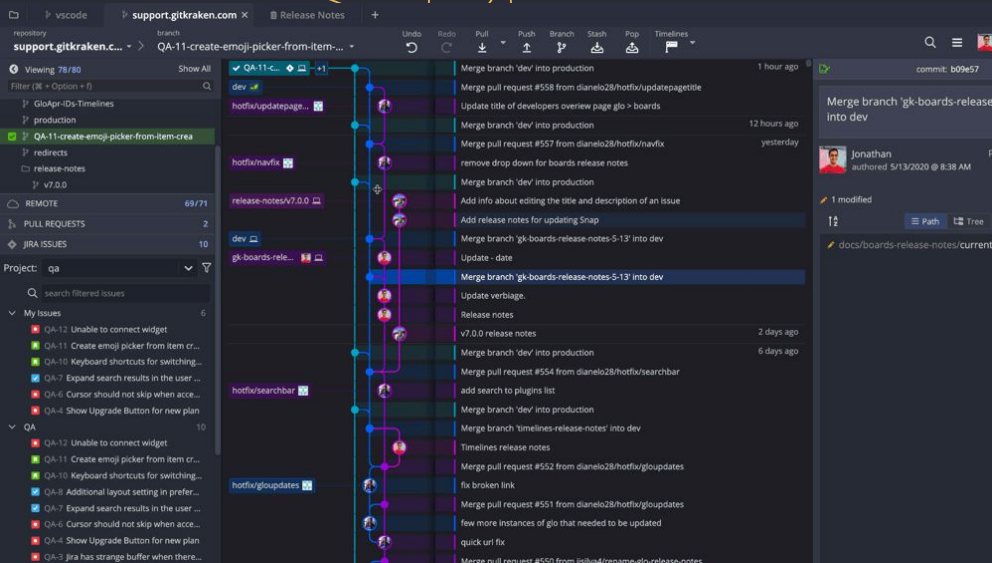
...or import code from another repository

You can initialize this repository with code from a Subversion, Mercurial, or TFS project.

Import code

OPERATING GIT

Using terminal or command
prompt



OPERATING GIT

Using GUI

ANDROID STUDIO

Android Studio has an integration with Git for source control management.

It means you can do the different Git operations (commit, push, pull, branch, etc.) from inside Android Studio



GROUP ASSESSMENT


- Make a group of maximum three students
- Follow all steps on the next slides
- **Each student must create their own repository**
- Practice git operation (pull, push, clone, merge, branch, checkout, and so on) have been executed on one of repositories
- Group leader must submit / writes github repository URL of all member at ULS
- Deadline: next week

CREATE GITHUB ACCOUNT

Go to <https://github.com/login>, click on “Create an account” link to create a new github account. Follow all steps.

Don't forget to try sign in into github after completing account creation.

(if you already have github account, you may skip this step)



Sign in to GitHub

Username or email address

Password [Forgot password?](#)

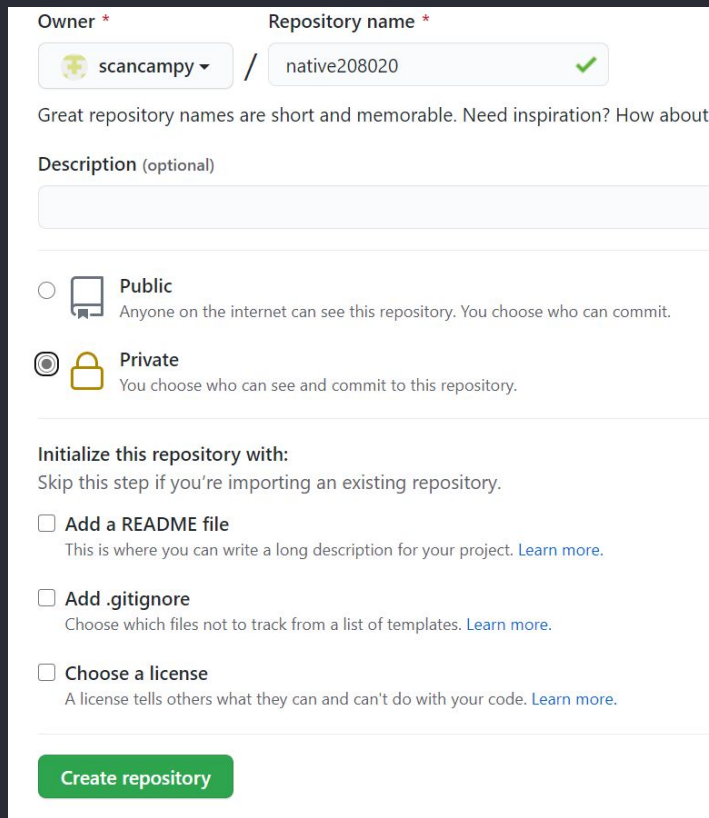
Sign in

New to GitHub? [Create an account.](#)

CREATE NEW REPOSITORY

Once sign in into github, click new button to create new repository

1. Write repo name (without space)
native[NPK]
2. Choose private repo
3. Click create repository



The screenshot shows the GitHub 'Create new repository' form. At the top, the 'Owner' is set to 'scancampy' and the 'Repository name' is 'native208020', which is marked as valid with a green checkmark. Below this, a message states: 'Great repository names are short and memorable. Need inspiration? How about...'. The 'Description (optional)' field is empty. Under the 'Visibility' section, the 'Private' option is selected with a radio button, accompanied by a lock icon and the text 'You choose who can see and commit to this repository.' The 'Public' option is unselected. The 'Initialize this repository with:' section includes three checkboxes: 'Add a README file' (unchecked), 'Add .gitignore' (unchecked), and 'Choose a license' (unchecked). Each checkbox has a brief description and a 'Learn more' link. At the bottom, there is a green 'Create repository' button.

Owner * Repository name *

scancampy / native208020 ✓

Great repository names are short and memorable. Need inspiration? How about

Description (optional)

☐ Public
Anyone on the internet can see this repository. You choose who can commit.

☒ Private
You choose who can see and commit to this repository.

Initialize this repository with:

Skip this step if you're importing an existing repository.

☐ Add a README file
This is where you can write a long description for your project. [Learn more.](#)

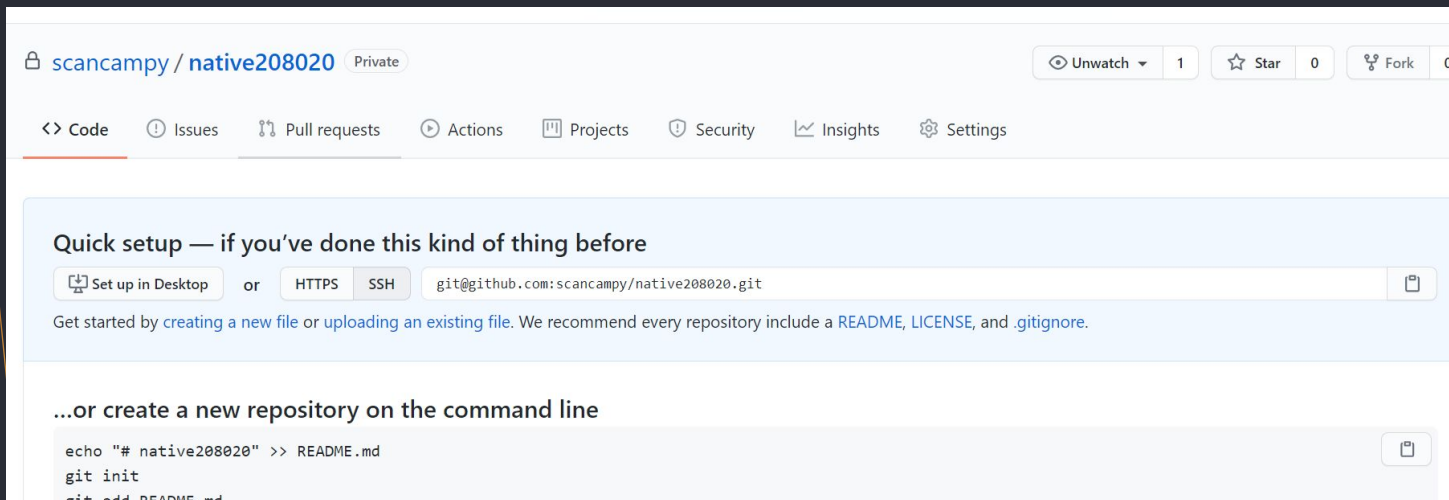
☐ Add .gitignore
Choose which files not to track from a list of templates. [Learn more.](#)

☐ Choose a license
A license tells others what they can and can't do with your code. [Learn more.](#)

Create repository

NEW REPOSITORY CREATED

New empty (remote) repository created



The screenshot shows the GitHub interface for a newly created repository. At the top, the repository name 'scancampy / native208020' is displayed with a 'Private' label. To the right are buttons for 'Unwatch' (with a dropdown arrow), '1' watch, '0' stars, and '0' forks. Below this is a navigation bar with links for 'Code', 'Issues', 'Pull requests', 'Actions', 'Projects', 'Security', 'Insights', and 'Settings'. The main content area has a light blue header with the text 'Quick setup — if you've done this kind of thing before'. Below this header are three buttons: 'Set up in Desktop', 'or', and 'HTTPS' (selected), followed by an 'SSH' button. To the right of these buttons is a text input field containing the repository URL 'git@github.com:scancampy/native208020.git' and a copy icon. Below the buttons, a paragraph states: 'Get started by [creating a new file](#) or [uploading an existing file](#). We recommend every repository include a [README](#), [LICENSE](#), and [.gitignore](#).' The bottom section is titled '...or create a new repository on the command line' and contains a code block with the following commands: 'echo "# native208020" >> README.md', 'git init', and 'git add README.md'. A copy icon is also present to the right of the code block.

scancampy / native208020 Private

Unwatch 1 Star 0 Fork 0

<> Code ! Issues 🔗 Pull requests 🔄 Actions 📁 Projects 🛡 Security 📈 Insights ⚙ Settings

Quick setup — if you've done this kind of thing before

Set up in Desktop or HTTPS SSH git@github.com:scancampy/native208020.git

Get started by [creating a new file](#) or [uploading an existing file](#). We recommend every repository include a [README](#), [LICENSE](#), and [.gitignore](#).

...or create a new repository on the command line

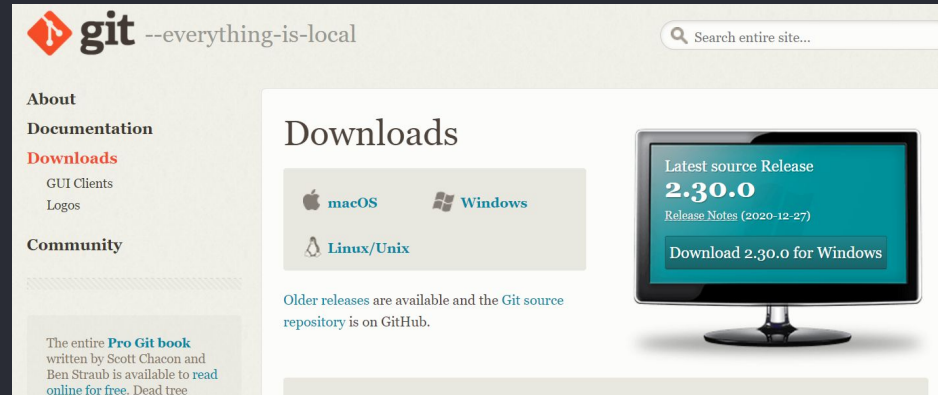
```
echo "# native208020" >> README.md
git init
git add README.md
```


DOWNLOAD AND INSTALL GIT

Go to
<https://git-scm.com/downloads>
to download Git system to your
computer

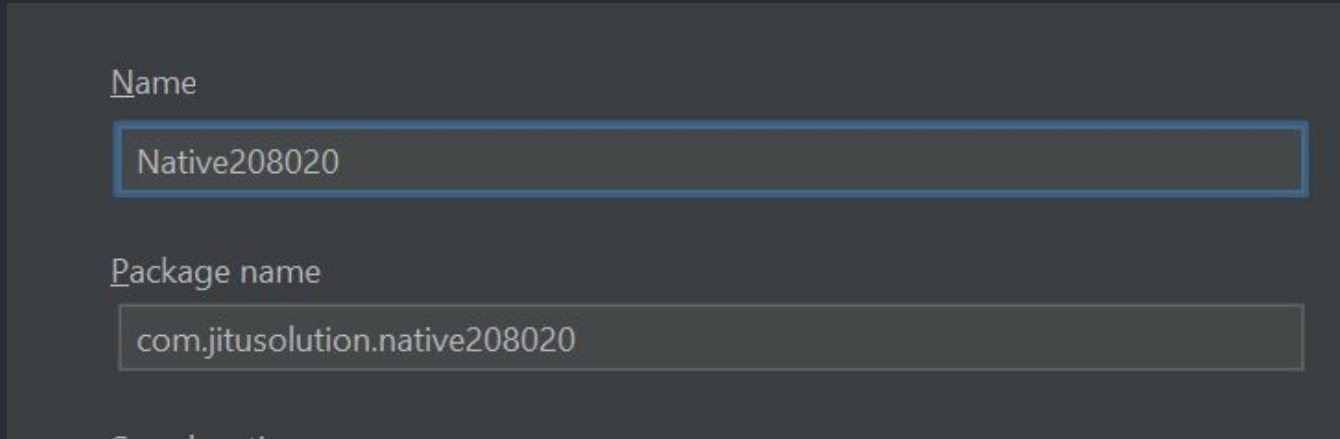
Install and follow the default
settings

You may follow installation
tutorial here:
<https://phoenixnap.com/kb/how-to-install-git-windows>



CREATE NEW ANDROID STUDIO PROJECT

Create new project, use empty activity, and name project as **Native[NPK]**



Name

Native208020

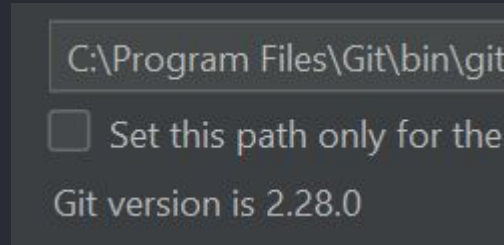
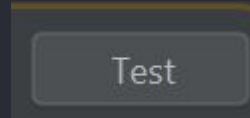
Package name

com.jitusolution.native208020

CHECK GIT INTEGRATION

Next, we need to test whether Git is configured in Android Studio

1. Click file > settings
2. Type git in search bar
3. Click “test” button
4. If you can see git version, means that android studio is configured properly with Git



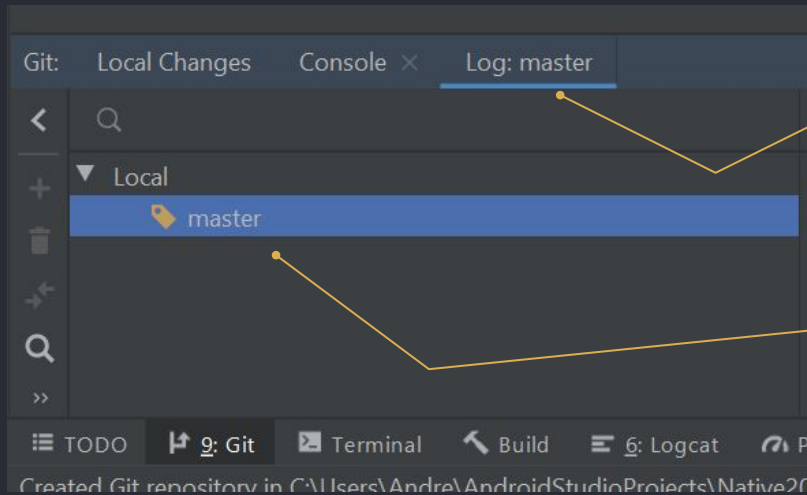
INTEGRATED GIT INTO PROJECT

Close setting dialog. Next, we need to enable VCS integration into our current project

1. Click VCS > Enable Version Control Integration
2. Popup dialog shown, choose git and press OK
3. Now a new repository with “master” branch have been created on your local drive

GIT TAB

New git tab shown on the bottom part of android studio



Click log tab

Notice that new branch "master" created

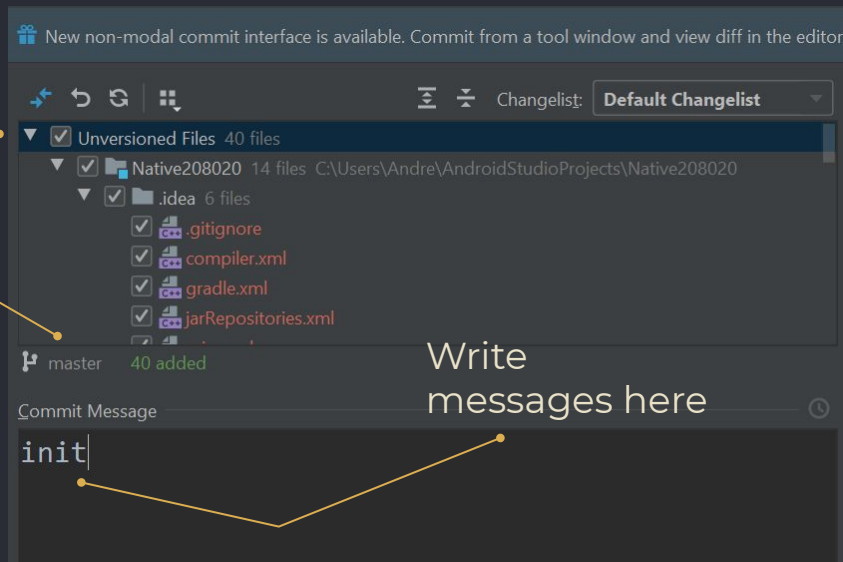
STAGE & COMMIT

The project is ready to use with Git version control. To stage and commit your changes, go to **VCS > Commit**.

Tick this
checkbox

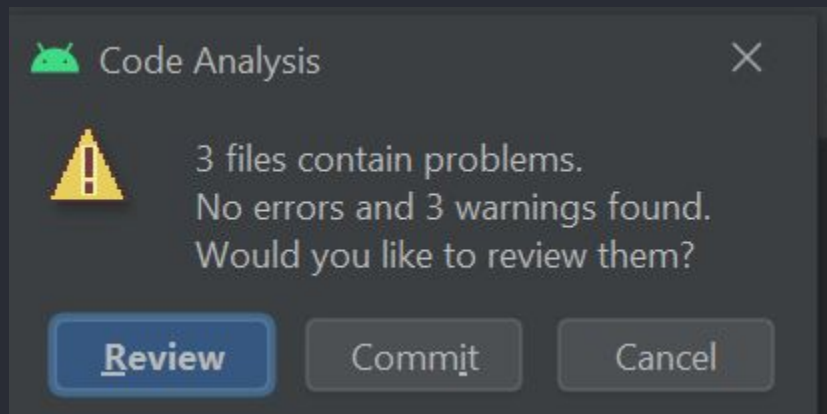
Current
working
branch

Write
messages here



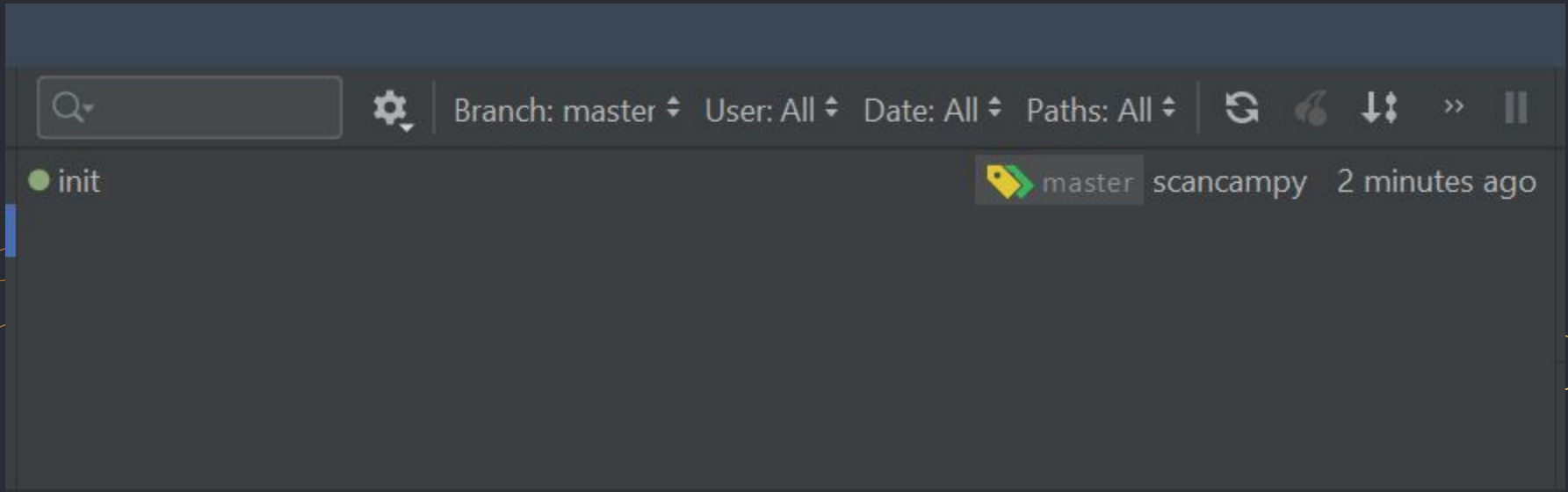
STAGE & COMMIT

Press commit button. Code analysis will show warning dialog like this. Warnings usually related with unfinished TODO, warning on UI or others. For now just press **Commit**



STAGE & COMMIT

Notice in git tab log, a new commit shown

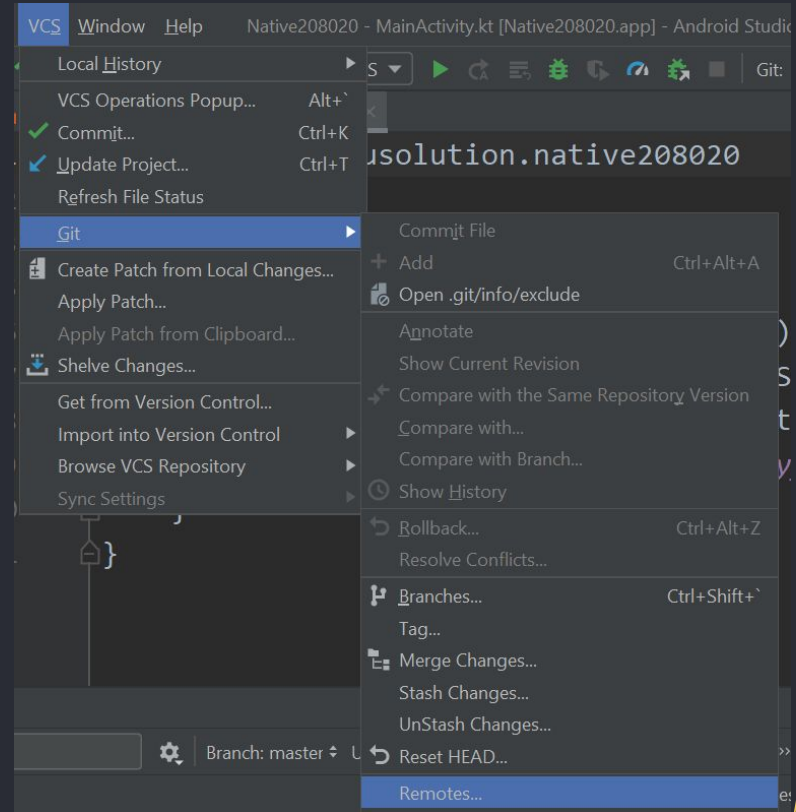


SETUP REMOTE CONNECTION

Previously we only staged and committed files into our own local repository

We need to push our commit to remote repository

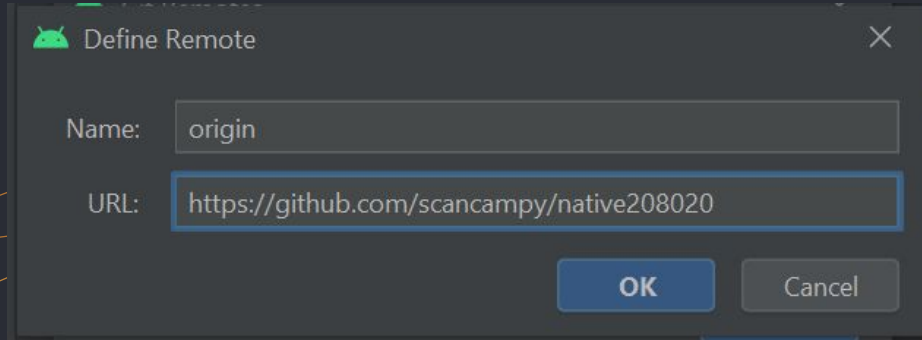
To configure remote repository click **VCS > Git > remote**



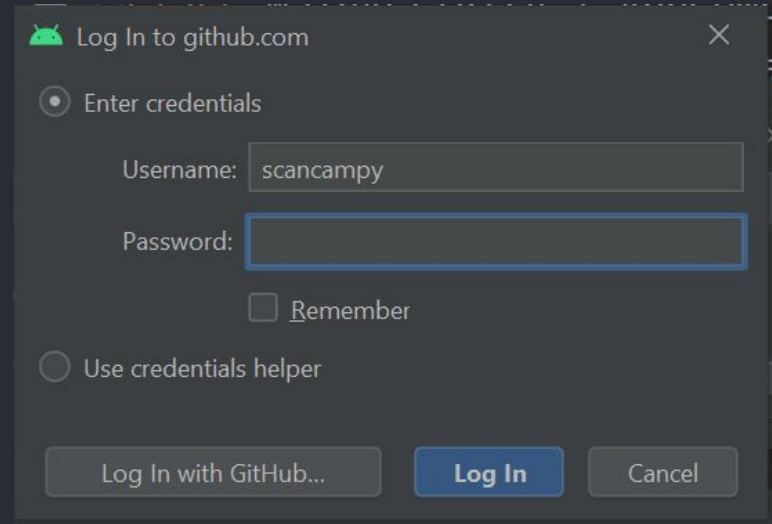
SETUP REMOTE CONNECTION

Click on + button, enter URL of your github repository that you created previously. Press OK

If authentication dialog shows up, you may need to provide github username and password.



A dialog box titled "Define Remote" with a close button (X) in the top right corner. It contains two input fields: "Name:" with the text "origin" and "URL:" with the text "https://github.com/scancampy/native208020". The URL field is highlighted with a blue border. At the bottom, there are two buttons: "OK" and "Cancel".



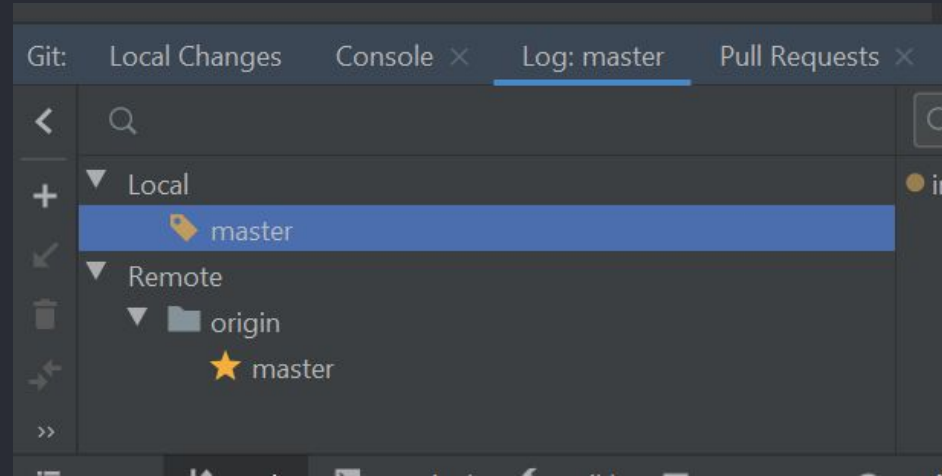
A dialog box titled "Log In to github.com" with a close button (X) in the top right corner. It has two main sections. The first section, "Enter credentials", is selected with a radio button. It contains "Username:" with the text "scancampy" and "Password:" with an empty field highlighted by a blue border. Below these is a checkbox labeled "Remember". The second section, "Use credentials helper", is unselected. At the bottom, there are three buttons: "Log In with GitHub...", "Log In" (highlighted with a blue border), and "Cancel".

PUSH THE CHANGES TO REMOTE

To push your local changes to the remote repository, go to **VCS > Git > Push**

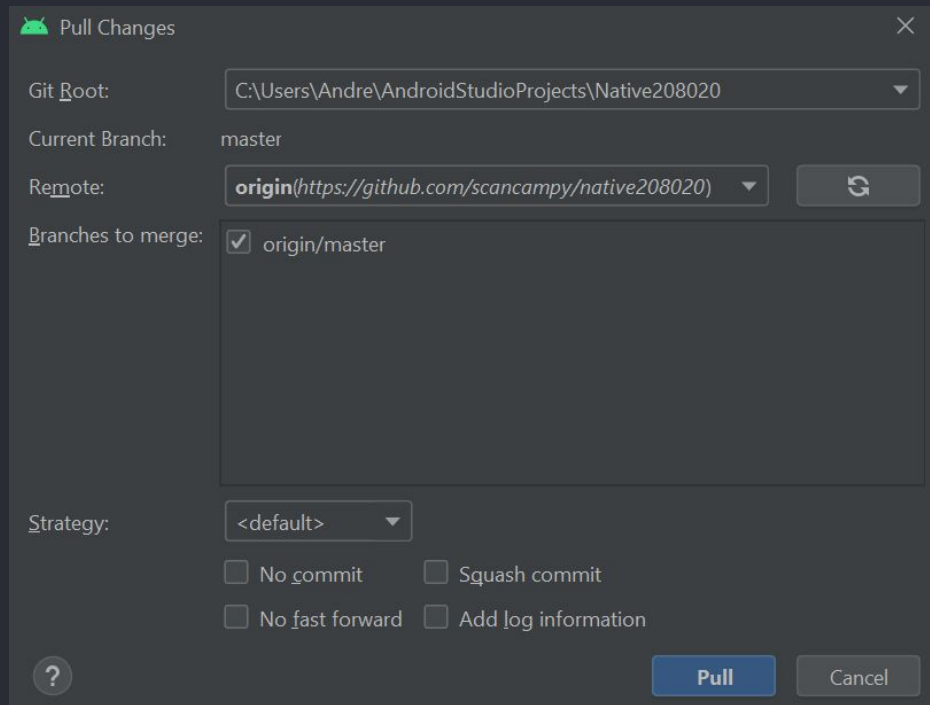
And then press push button again to begin push operation

If success, you can see two repositories (local and remote) shown on git log tab



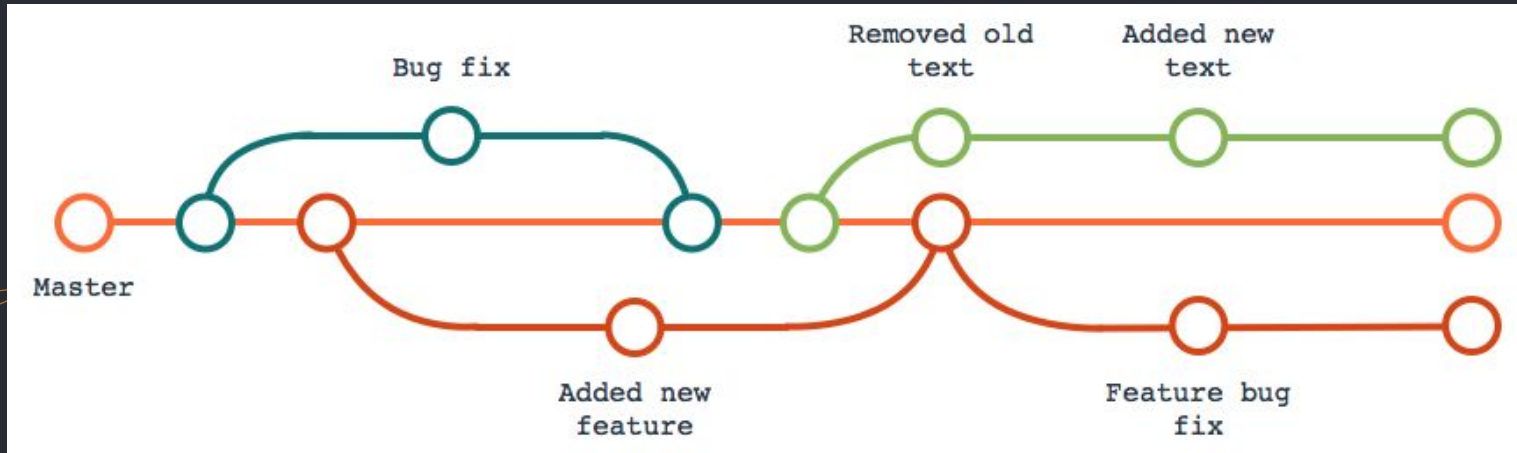
FETCH THE CHANGES FROM REMOTE

To download the latest changes from remote, go to **VSC > Git > Pull**



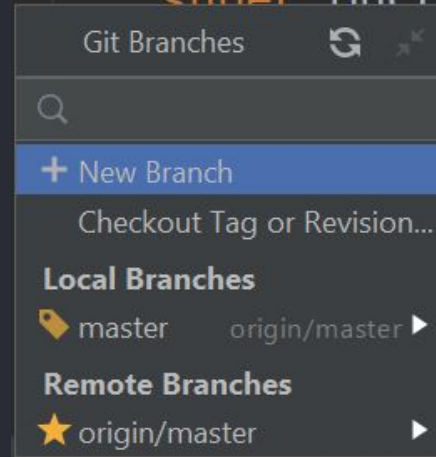
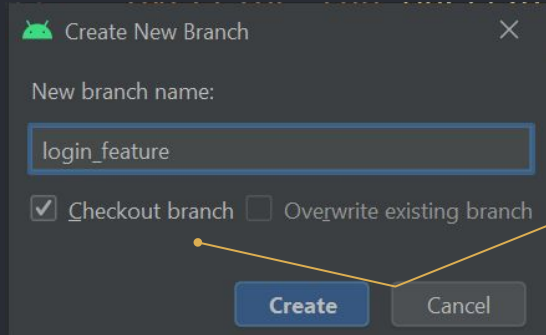
CREATE BRANCHES

- Branching is used to maintain stability while isolated changes.
- Usually branch used for working on bug fixes, develop new features, integrate new versions after they have been tested in isolation



CREATE NEW BRANCH

- Go to VCS > Git > Branches.
- Choose + New Branch
- Name new branch
- Press create

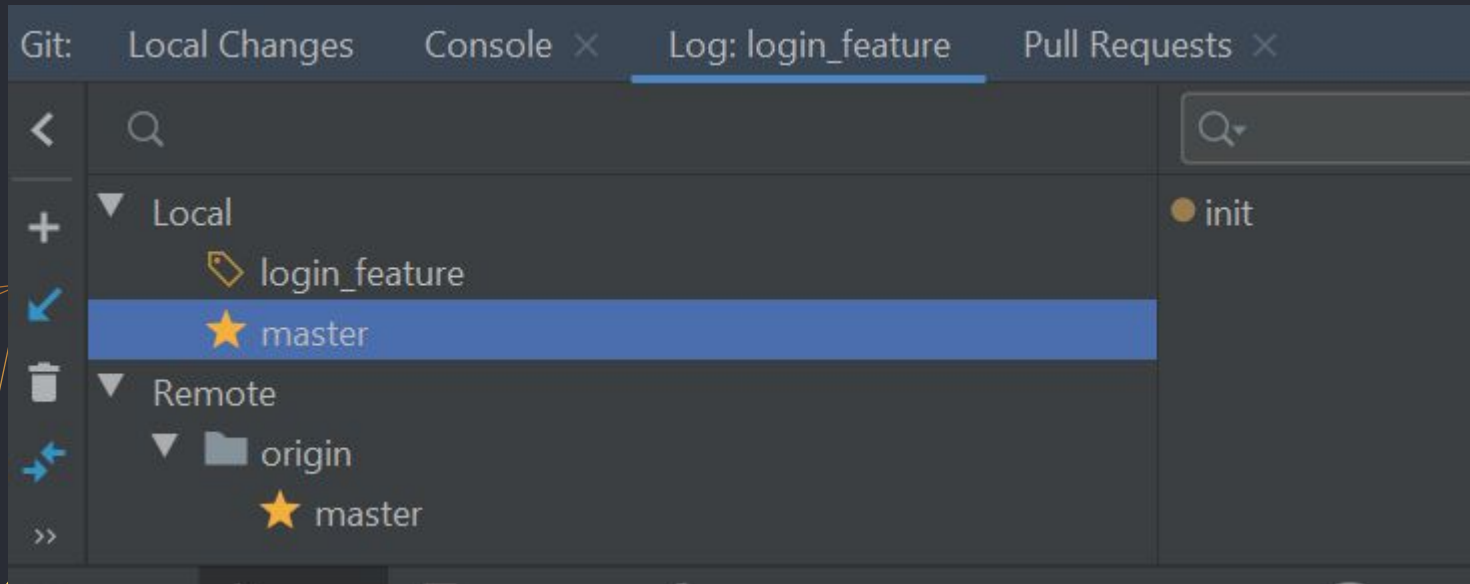


Checking out a branch updates the files in the working directory to match the version stored in that branch, and it tells Git to record all new commits on that branch

NEW BRANCH CREATED

Notice that now in local repository, there are a new branch “login feature”

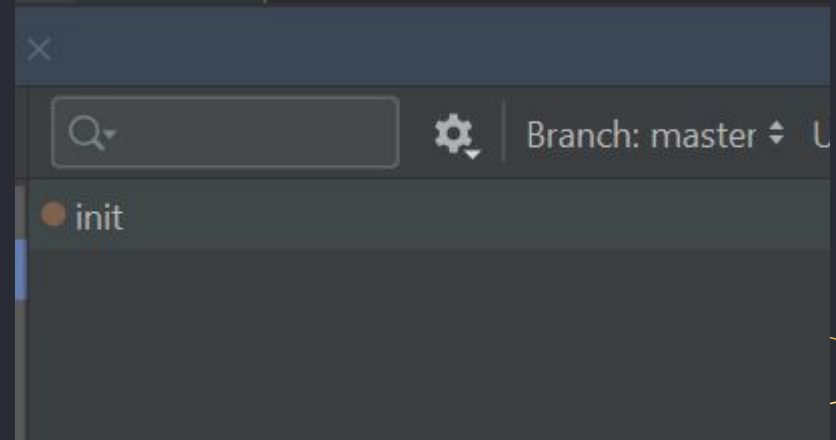
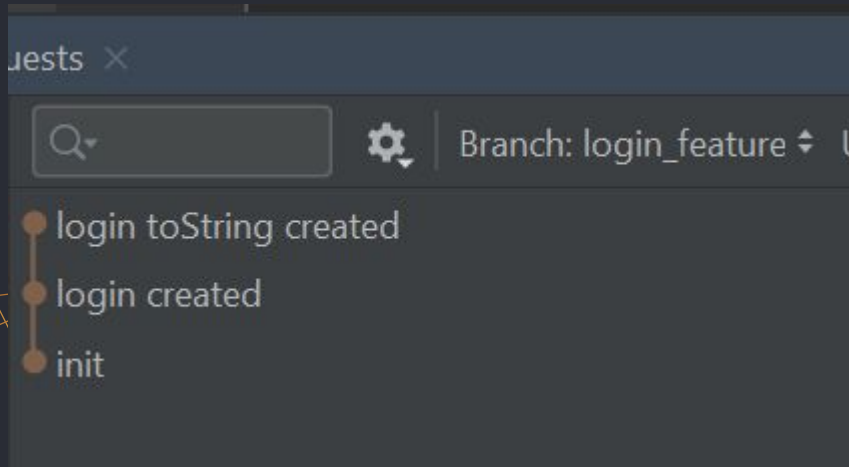
Any commit and push will affect this new branch



BRANCH

Below images are example of new “login_feature” branch already had two commit.

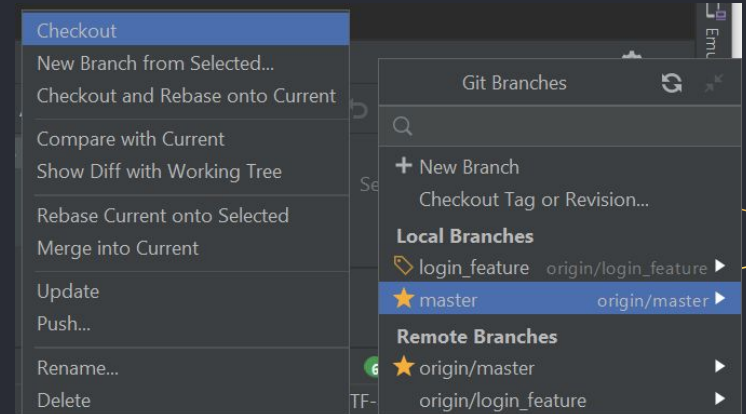
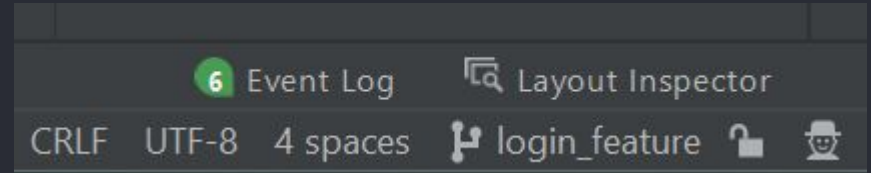
Master branch still only have 1 commit “init”



SWITCH/CHECKOUT BETWEEN BRANCH

Its very easy to switch between branch.

1. Look at the bottom right corner and you'll see the current working branch
2. Click on it to reveal all branches.
3. Click on branch that you want to switch/checkout
4. Choose checkout



MERGE BRANCH

Merging is Git's way of putting a forked history back together again

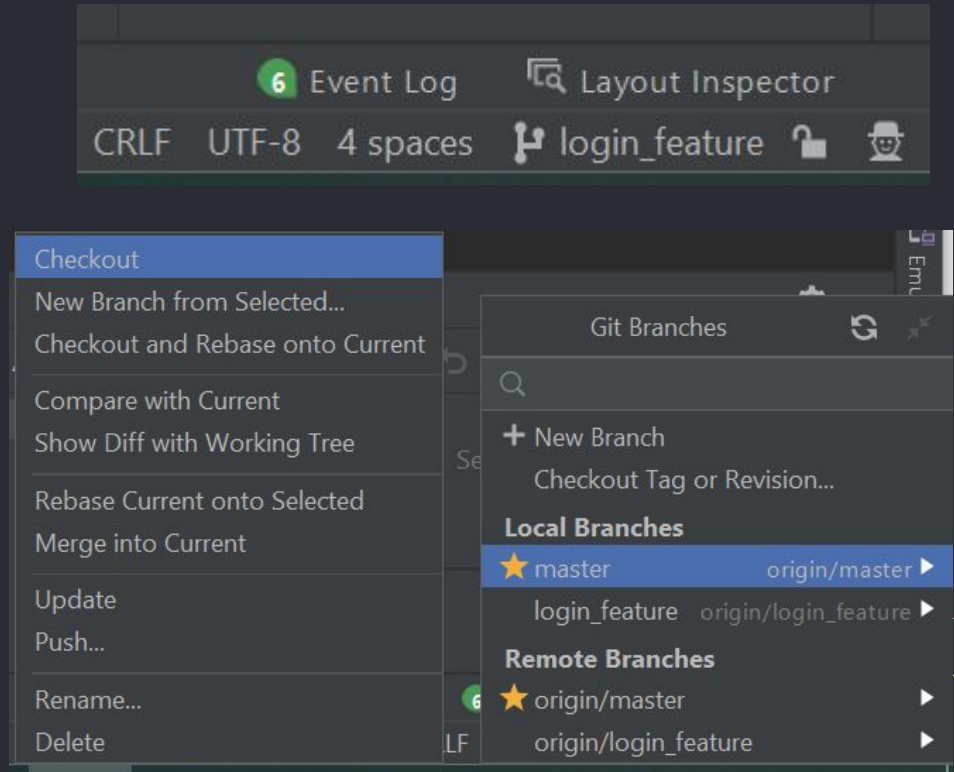


MERGE BRANCH

Let's say we want to merge **login_feature** branch into master branch

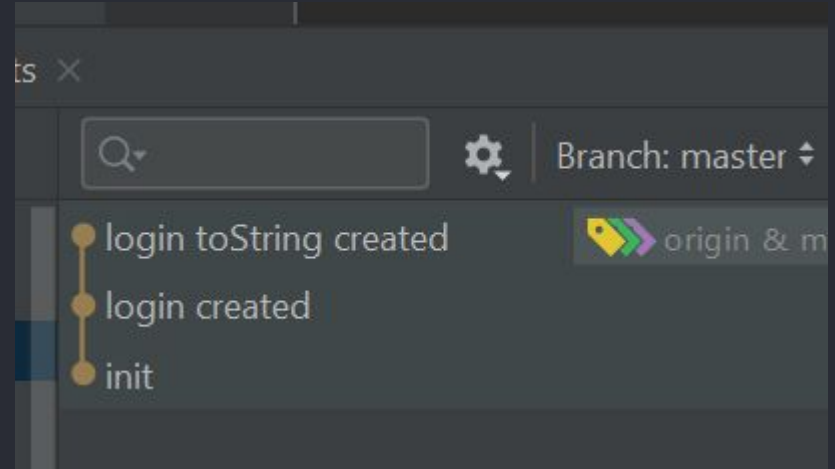
To do that look at the bottom right corner, click on the branch name

Choose master branch and click on “Checkout and Rebase onto Current”



BRANCH HAVE BEEN MERGED

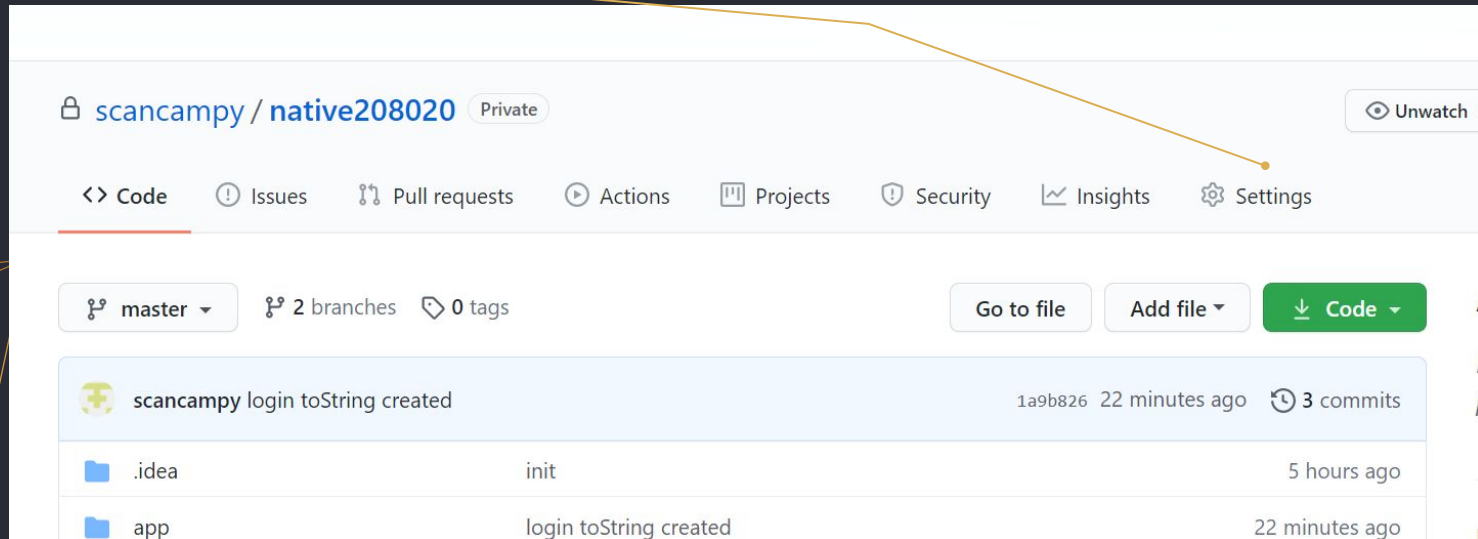
If you look on git log tab, you will see that “master” branch have been merged with “login_feature”



ALLOW ACCESS FOR OTHER TEAM MEMBER

Since your repository is private, you must manually add other github account to give access privilege

To manage user access, open your repository from browser and click on "Settings"



ALLOW ACCESS FOR OTHER TEAM MEMBER

Click menu
Manage access

The screenshot displays the GitHub repository settings interface. On the left is a sidebar menu with the following items: Options, Manage access (highlighted with a red vertical bar and a yellow callout line), Security & analysis, Branches, Webhooks, Notifications, Integrations, Deploy keys, Actions, and Secrets. The main content area is titled 'Who has access' and contains two panels: 'PRIVATE REPOSITORY' (with a lock icon) stating 'Only those with access to this repository can view it.' and a blue 'Manage' link; and 'DIRECT ACCESS' (with a person icon) stating '0 collaborators have access to this repository. Only you can contribute to this repository.' Below these panels is a section titled 'Manage access' which contains a large box with a collaborator icon (a person with a lock) and the text 'You haven't invited any collaborators yet'. A green button labeled 'Invite a collaborator' is at the bottom of this box. A yellow callout line points from the text 'Click invite a collaborator' to the 'Invite a collaborator' button.

Options

Manage access

Security & analysis

Branches

Webhooks

Notifications

Integrations

Deploy keys

Actions

Secrets

Who has access

PRIVATE REPOSITORY

Only those with access to this repository can view it.

[Manage](#)

DIRECT ACCESS

0 collaborators have access to this repository. Only you can contribute to this repository.

Manage access

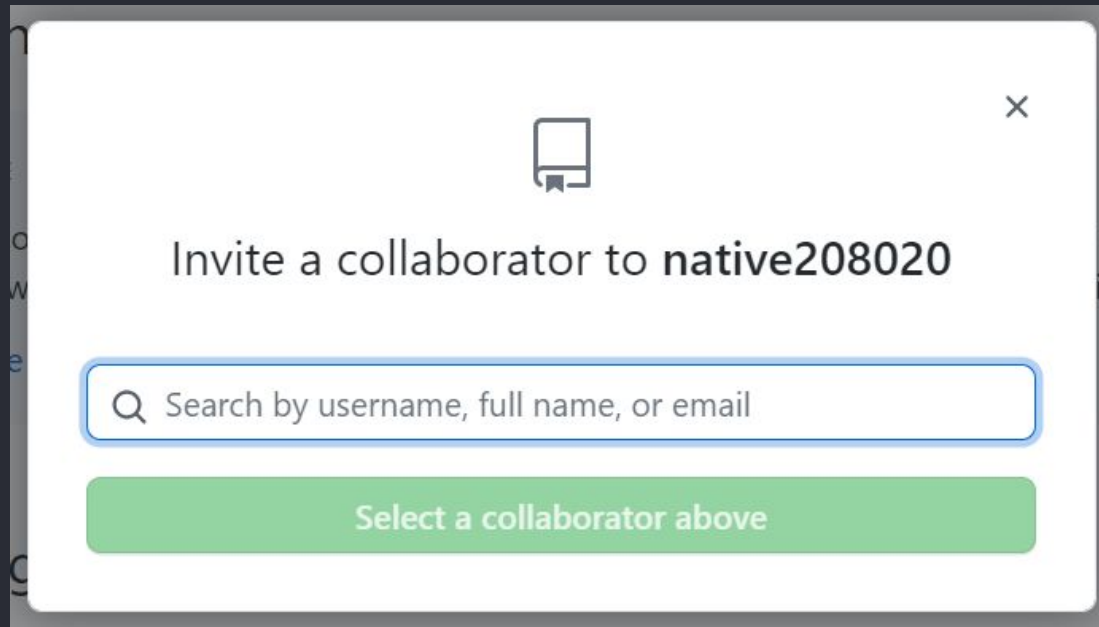
Click invite a collaborator

You haven't invited any collaborators yet


[Invite a collaborator](#)

ALLOW ACCESS FOR OTHER TEAM MEMBER

Write your friend email or github username. If success, your friend will have access to run git operation on this repository.



The image shows a screenshot of a GitHub web interface dialog box titled "Invite a collaborator to native208020". The dialog box is white with a gray border and a close button (X) in the top right corner. At the top center is a GitHub logo. Below the title is a search input field with a magnifying glass icon and the placeholder text "Search by username, full name, or email". Below the search field is a green button with the text "Select a collaborator above".



×

Invite a collaborator to **native208020**

Select a collaborator above

CLONE EXISTING PROJECT

There are tons of available github project on the internet.

Using android studio you can easily clone github project and start work on it

In this case lets clone your friend repository

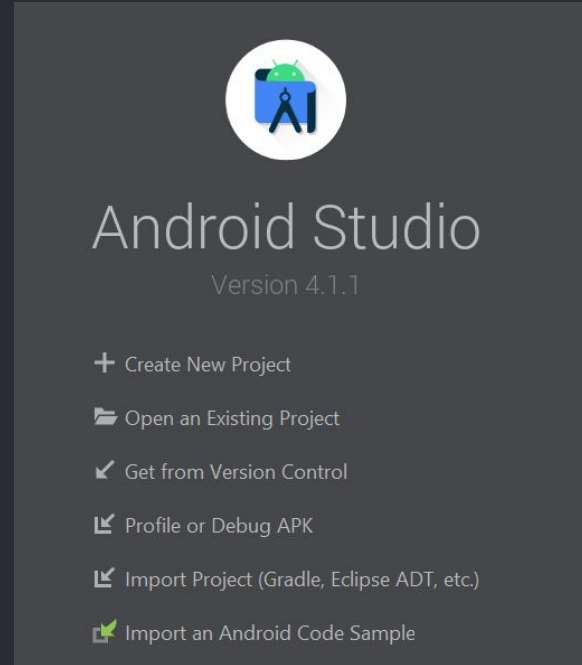


Images of Clone Trooper
(courtesy of Starwars Universe)

CLONE EXISTING PROJECT

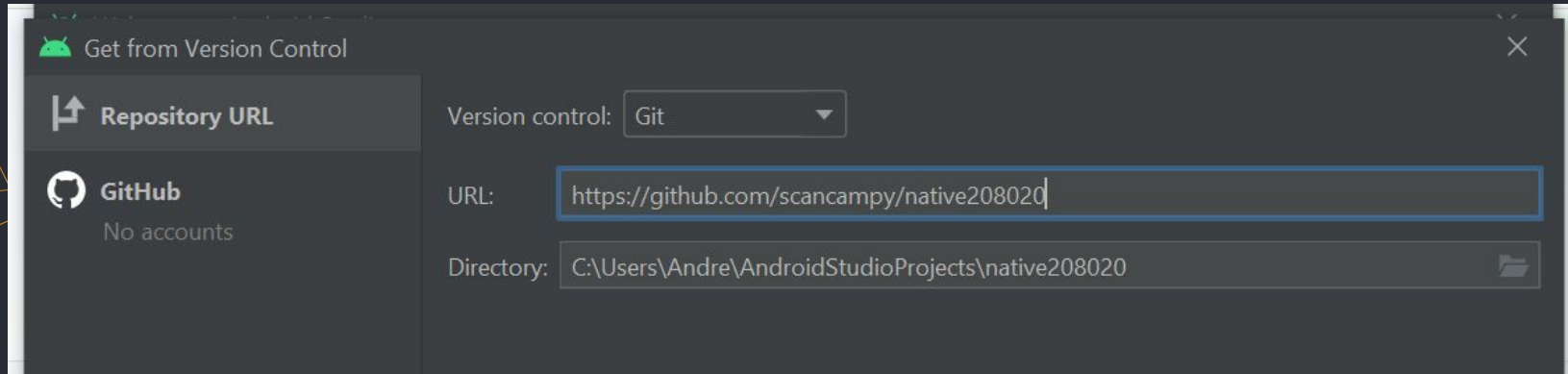
First close all project in android studio

On the Android Studio Welcome screen, click on “Get from Version Control” menu



CLONE EXISTING PROJECT

Copy paste your friend repository URL into URL textbox. Make sure your account already have access.



THANKS!

DO YOU HAVE ANY QUESTION?

andre@staff.ubaya.ac.id



CREDITS: This presentation template was created by [Slidesgo](#), including icons by [Flaticon](#), infographics & images by [Freepik](#).

Please keep this slide for attribution.

