Version Control with Android Studio (Kotlin)

Limashi Rashmika

NIBM POWERING GREAT MINDS

What is Version Control?

- Version Control is **a system that helps developers track changes in their code and manage different versions of a project**.
- It allows you to
  - Save versions of your code.
  - Go back to previous versions.
  - Collaborate with others without overwriting each other's work.

Why Do We Use Version Control?

- Keep track of every change
- Roll back to an earlier version if something breaks
- Work with a team without overwriting each other’s work
- Merge changes from different developers
- Keep code safely backed up online

Methods of Version Control

- There are three main types
  - **Local Version Control**
    - Track changes only on your computer
    - Example - File history on your PC
  - **Centralized Version Control**
    - One central server stores all versions
  - **Distributed Version Control**
    - Git (most popular)

What is Git?

- Git is a free, open-source distributed version control system.
- It helps you track changes, collaborate, and manage source code history.

What is GitHub?

- GitHub is a platform to host Git repositories online.
- It helps with sharing code, collaboration, and version tracking.

Other Git Hosting Services

- GitHub
- GitLab
- Bitbucket

Remote Repository

- A remote repository is the **online version of your project, usually hosted on platforms like GitHub**.
- It allows you to share your code and collaborate with others.
- Example https://github.com/NIBMLectures/SampleApp1

Commit

- A commit **saves a snapshot of your current code.**
- You add a message to explain what you changed.

Push

- A push **sends your local commits to the remote repository (GitHub).**
- This updates the online version of your project.
- You commit locally, then push to share.

Pull

- A pull **fetches the latest changes from the remote repository to your local machine**.
- Use this to get updates others made.
- **Always pull before starting new work!**

Pull Request

- A Pull Request (PR) **is a way to propose changes to a project in GitHub before those changes are merged into the main code**.

Branch

- A branch is **a copy of your code used to make changes or test features without affecting the main code (usually called main or master).**
- You can merge it back after testing.

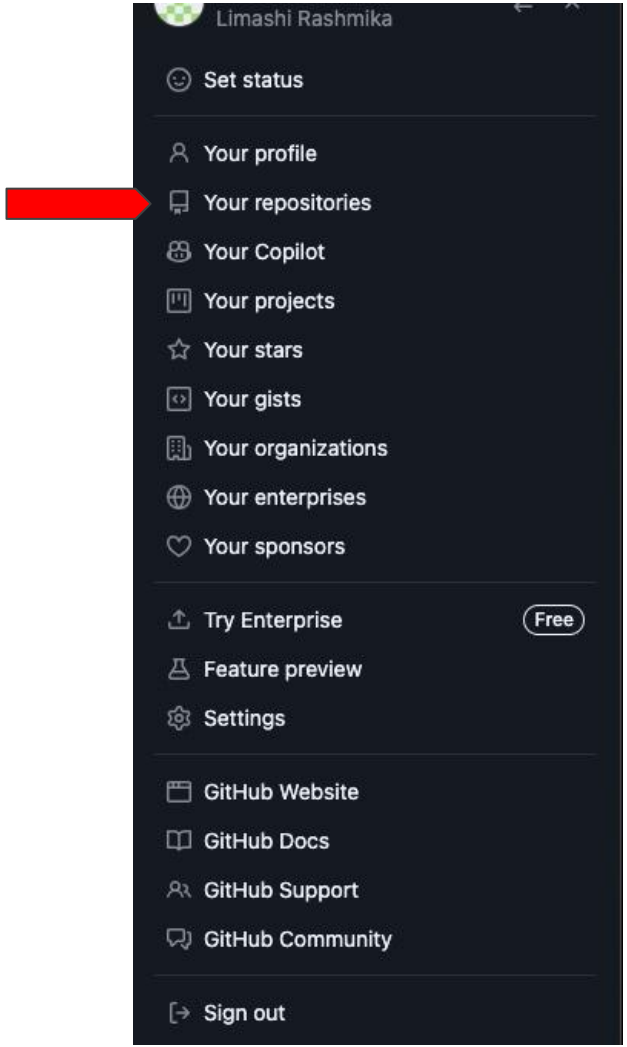
Clone

- To clone **means to download a remote repository onto your computer**.
- This gives you a local copy to work with.
- **First step when starting a new project from GitHub.**

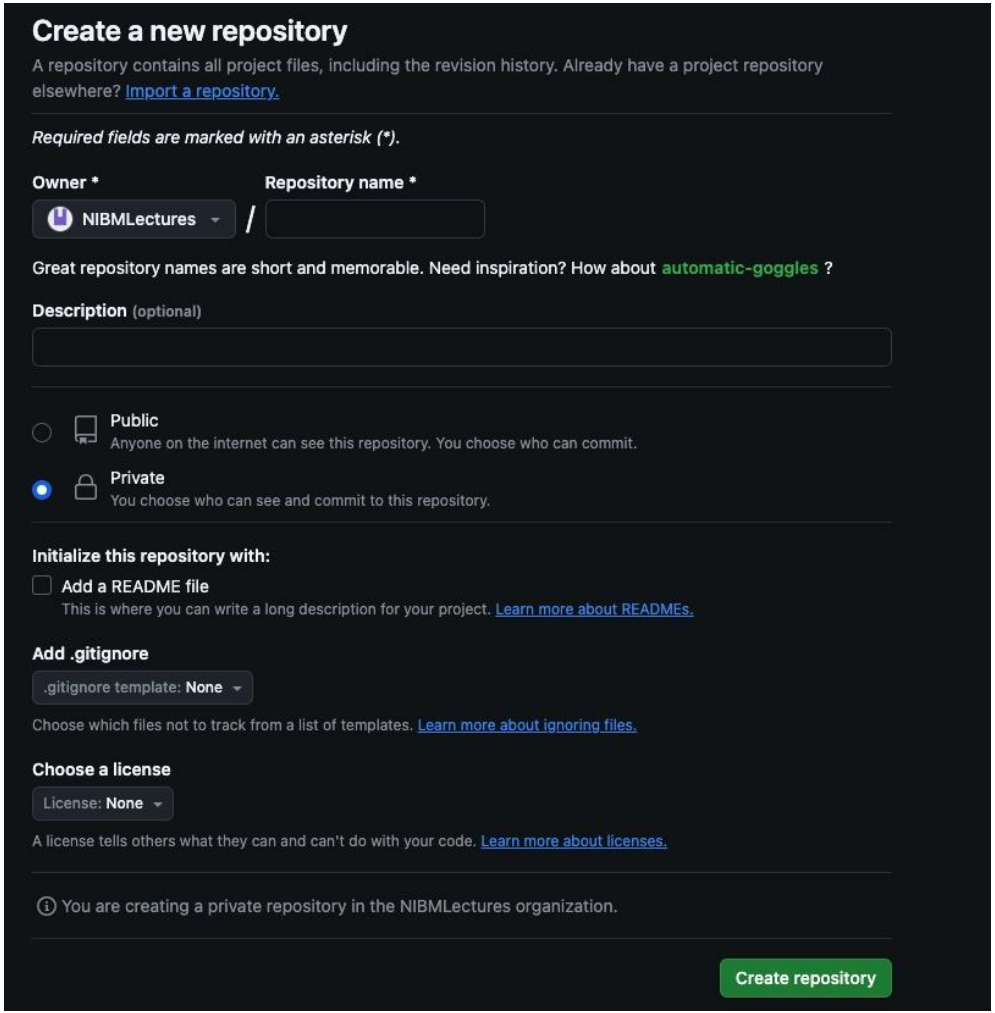
How to Create a Remote Repository on GitHub

- Log in to your Github account (or sign up if you don’t have one)
- **Select Your Repositories > New > Create a new repository**

How to Create a Remote Repository on GitHub



How to Create a Remote Repository on GitHub



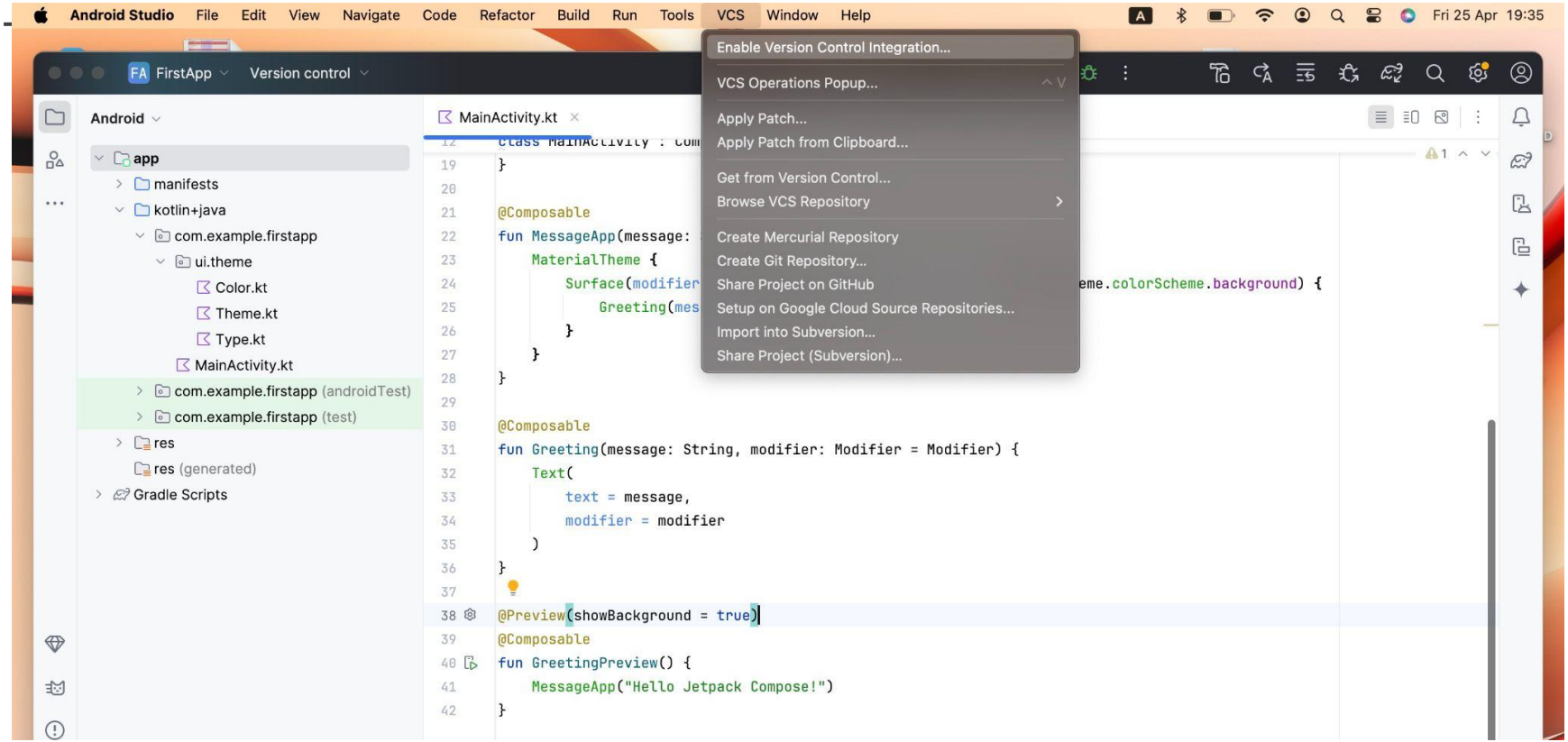
NOTE

- Repository Name- Eg: MyKotlinApp
- Description (optional)- Short project description
- Visibility
  - Public (anyone can see it)
  - Private (only you/your team can see it)
- Check “Initialize this repository with a README” (optional)
- Then click Create repository

How to Connect GitHub to Android Studio Kotlin Project

- Open your Kotlin project in Android Studio
- Go to **VCS > Enable Version Control Integration**
- Choose **Git** and click OK
- Now your project is initialized with Git locally.

How to Connect GitHub to Android Studio Kotlin Project



NOTE

- There are two ways to connect a GitHub account to Android Studio
  - **Login with GitHub Account**
  - **Login with Personal Access Token (PAT)**

Login with GitHub Account

- This method uses **OAuth authorization**.
- The user logs in through a browser popup **using their GitHub username and password**.
- Android Studio is then granted access to the GitHub account.

Login with Personal Access Token

- GitHub has disabled password authentication for Git operations, so tokens are now required.
- Users generate a PAT from GitHub (with appropriate permissions like repo) and paste it into Android Studio.
- This token is used as a secure alternative to passwords when pushing, pulling, or cloning repositories.

NOTE

- Both methods are valid, but using **Personal Access Tokens is more flexible and secure**, especially when working with private repositories or multiple GitHub accounts.

What is a Personal Access Token?

- A Personal Access Token is **like a password you generate on GitHub to securely connect tools like Android Studio, Git CLI, or Postman to your GitHub account**.

Why Do We Need PATs?

- GitHub no longer supports password-based authentication for Git operations (like push, pull) from the command line or tools like Android Studio.

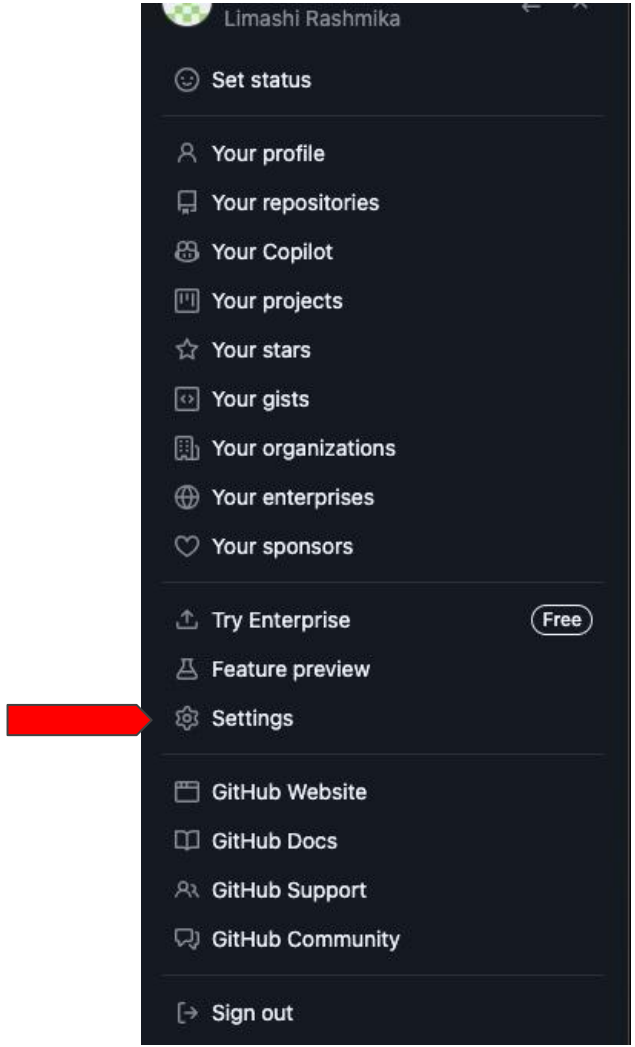
When Do You Use a PAT?

- Pushing code from Android Studio to GitHub
- Cloning private repos using HTTPS
- Using GitHub with third-party apps like Postman, CI/CD tools, etc.

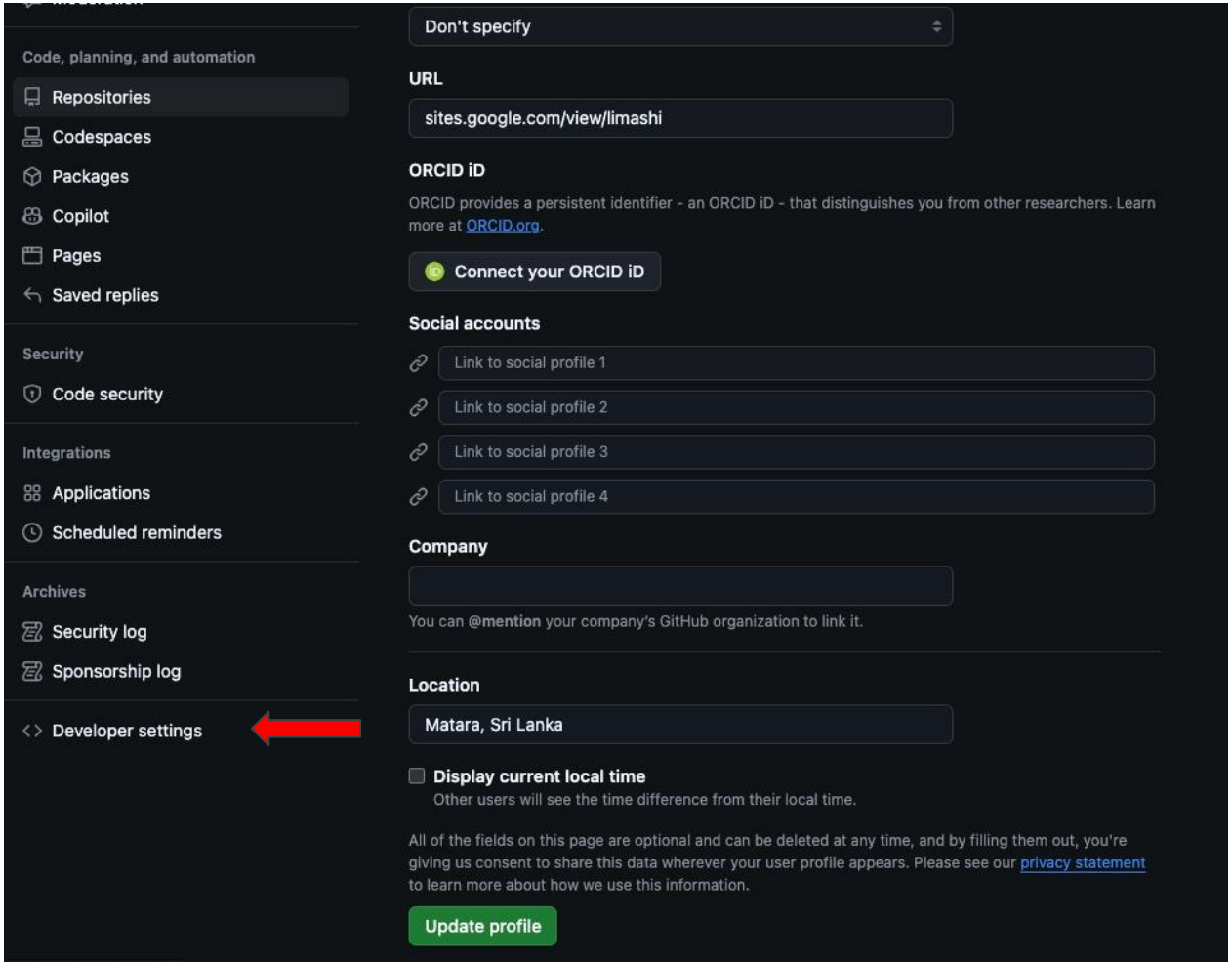
How to Create a Personal Access Token on GitHub

- Login to your GitHub account
- Click your profile icon → Go to Settings
- Click Developer Settings
- Select Personal Access Tokens → Click “**Tokens (classic)**”

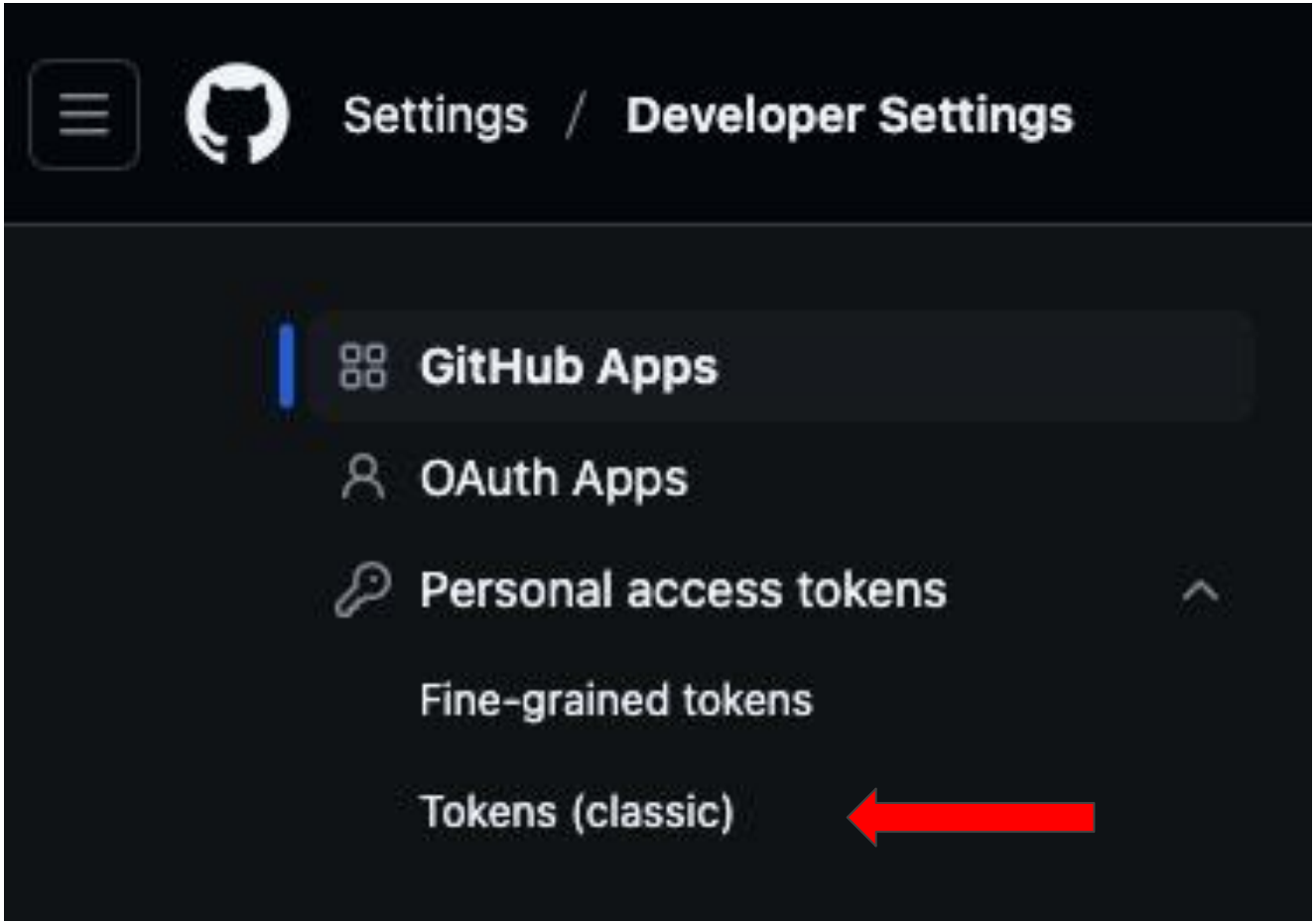
How to Create a Personal Access Token on GitHub



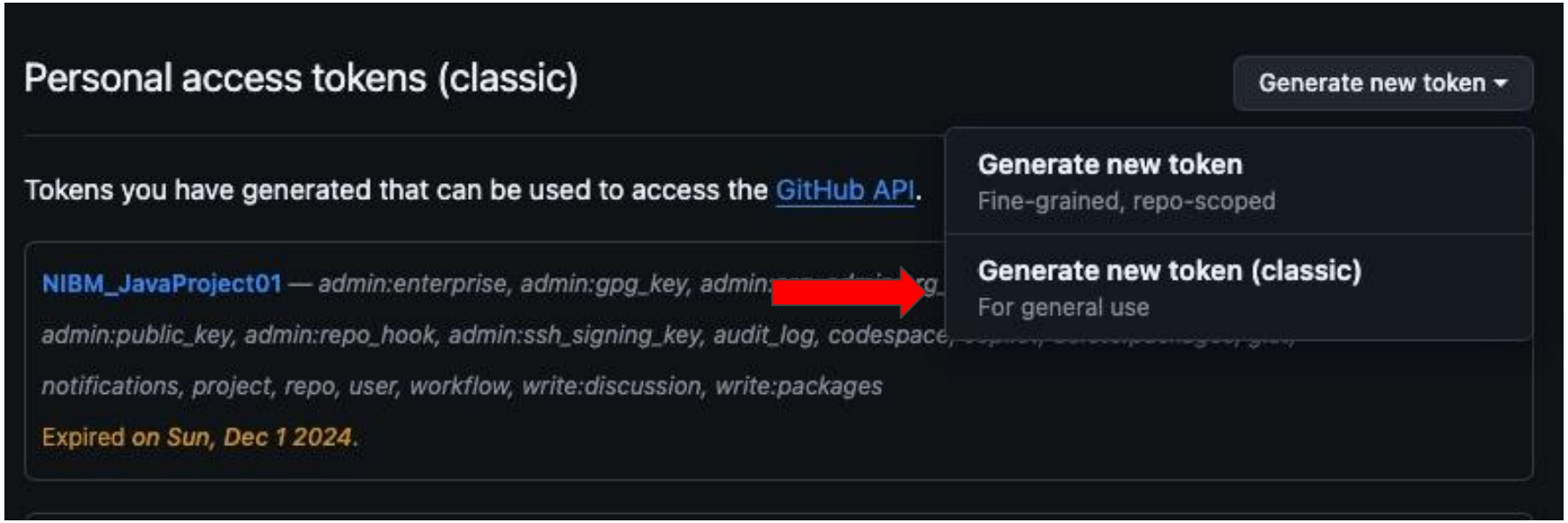
How to Create a Personal Access Token on GitHub



How to Create a Personal Access Token on GitHub



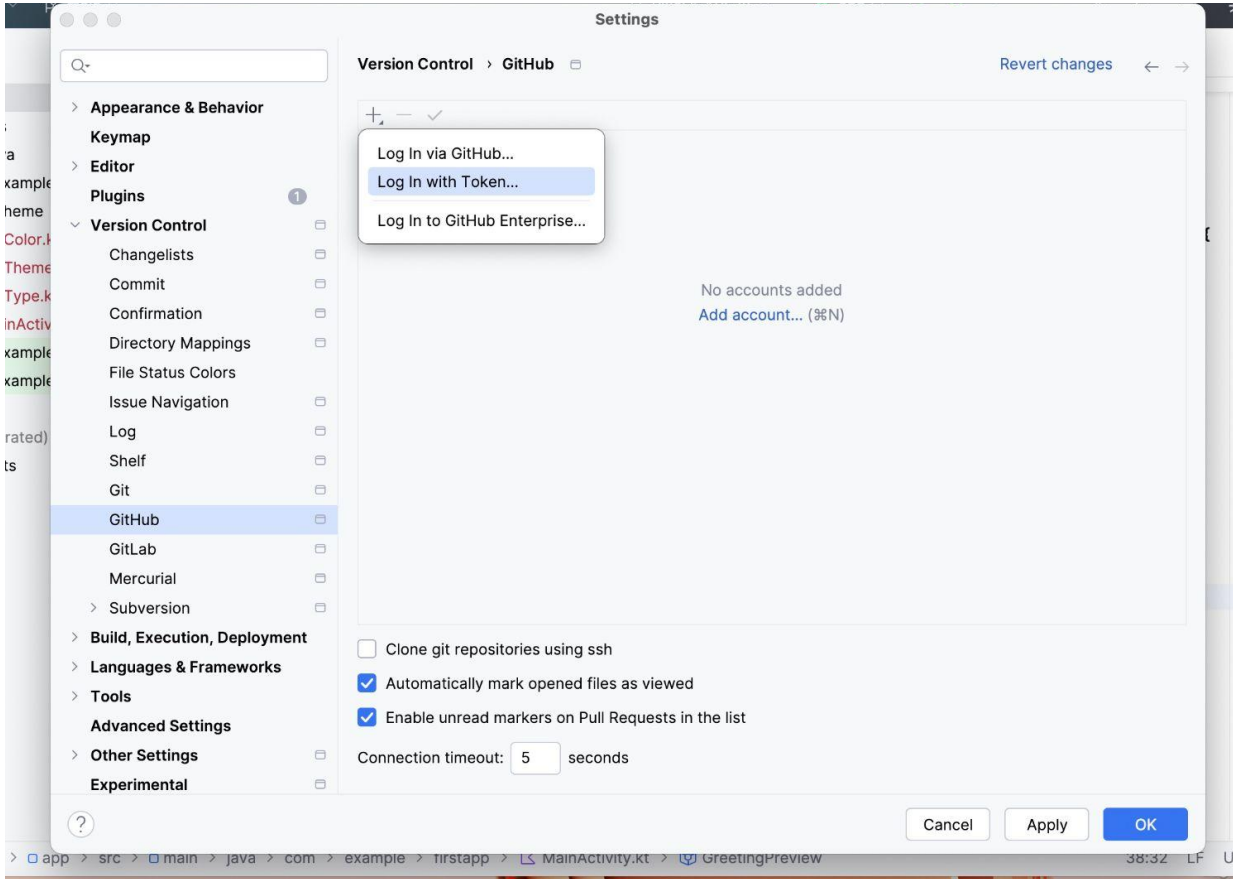
How to Create a Personal Access Token on GitHub



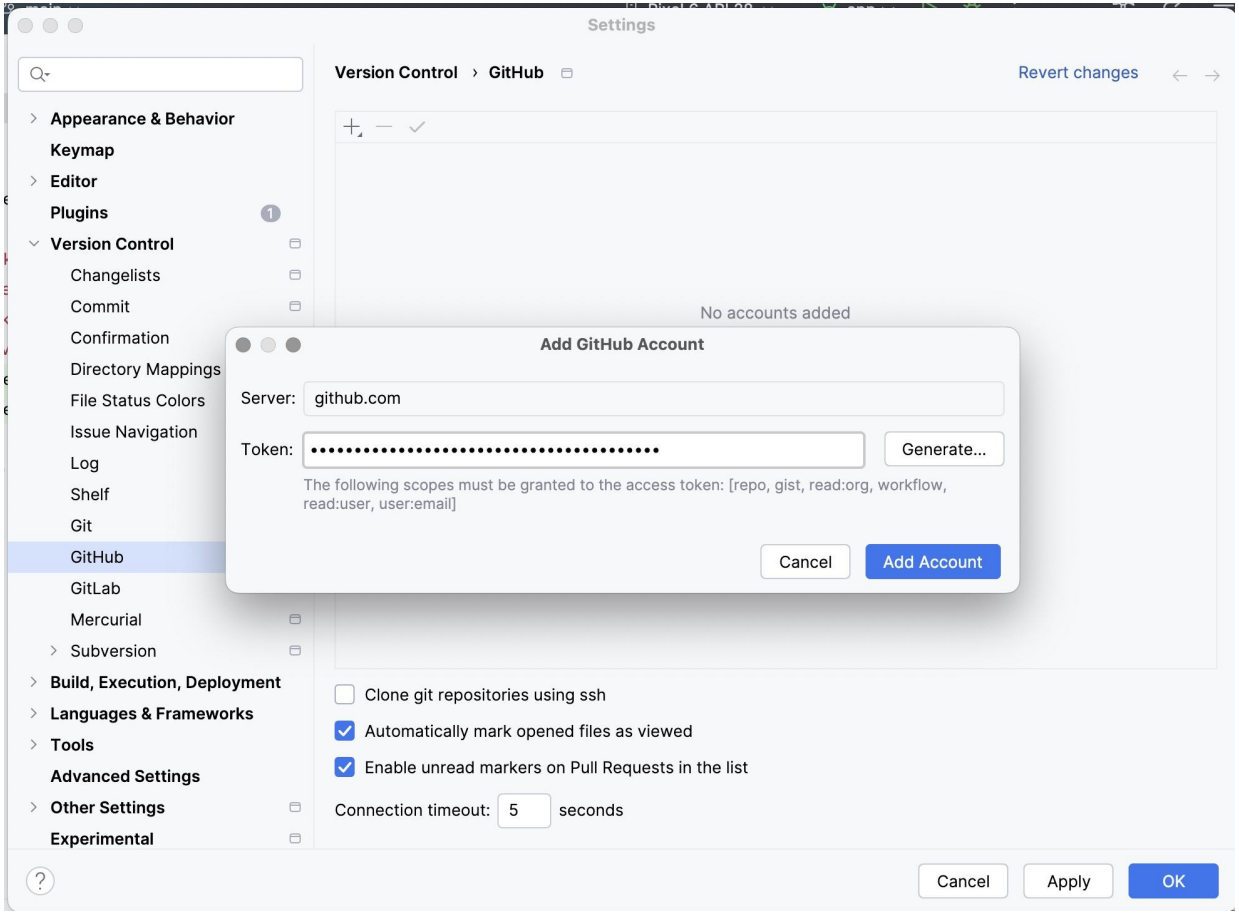
Sign In to GitHub

- Go to **File > Settings**
- Navigate to **Version Control > GitHub**
- Click **“+”** to add an account
- Choose “**Log in with GitHub**” or use a **Personal Access Token (PAT)**
- Authorize and complete the login

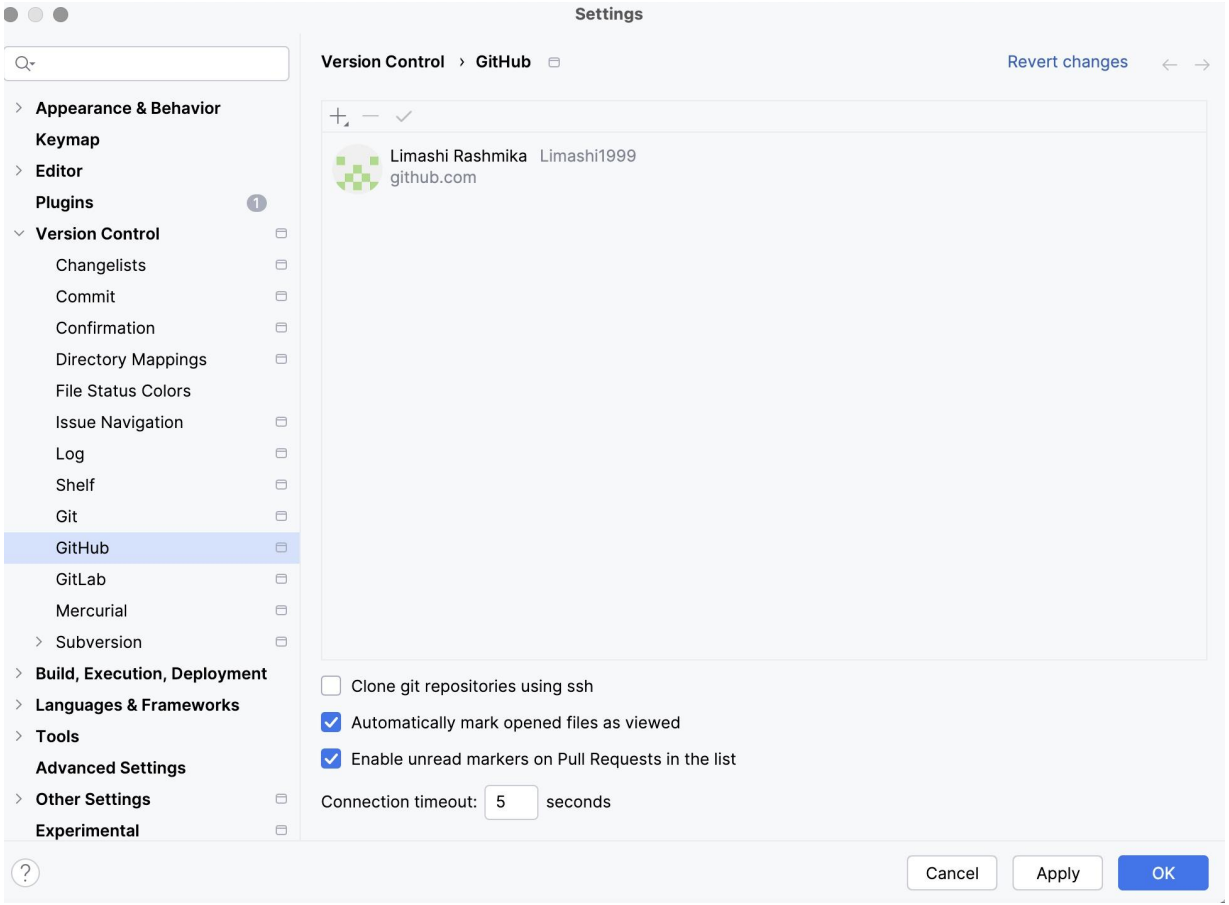
Sign In to GitHub



Sign In to GitHub



Sign In to GitHub

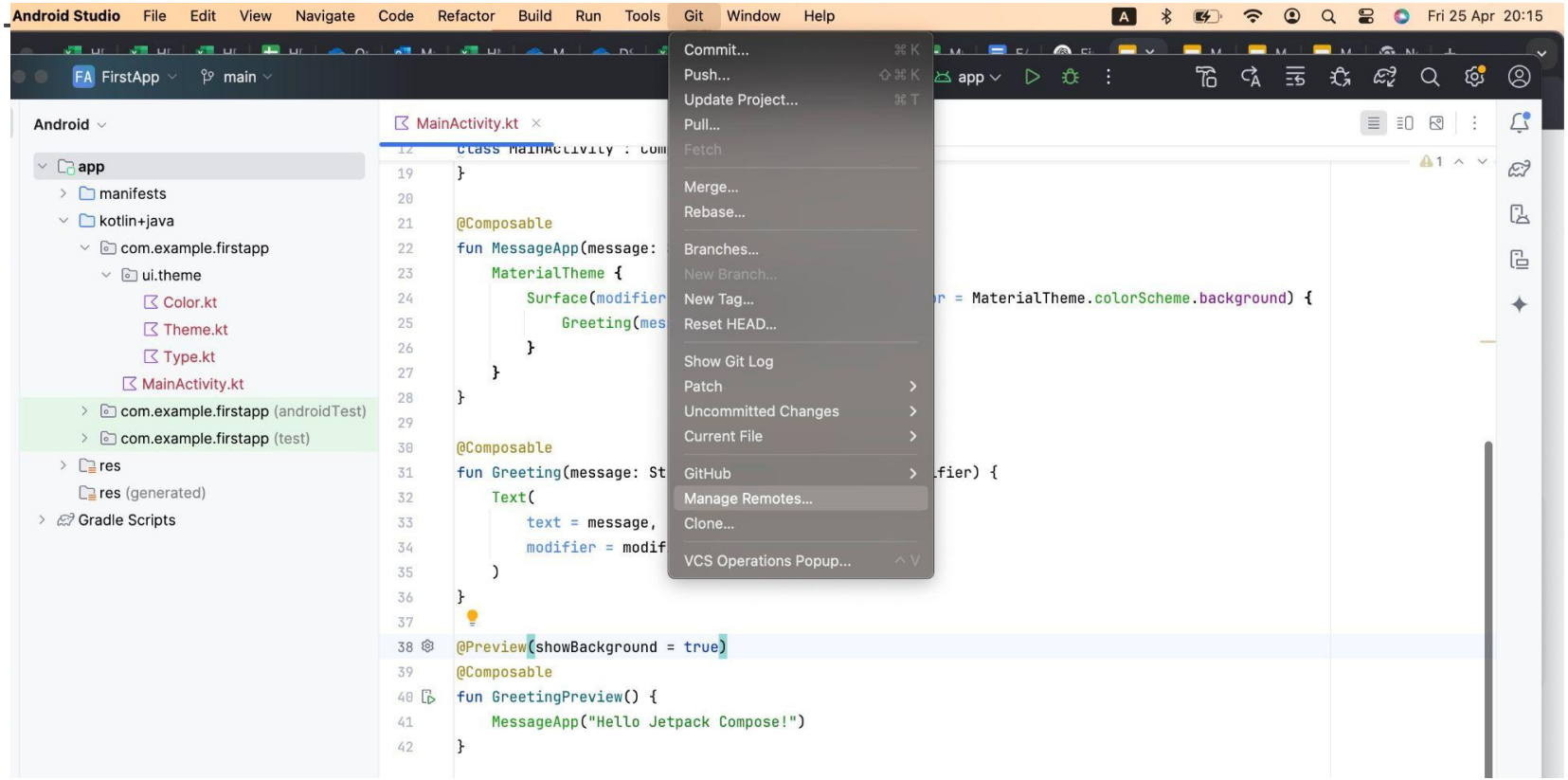


Connect Android Studio Project to Remote GitHub Repository

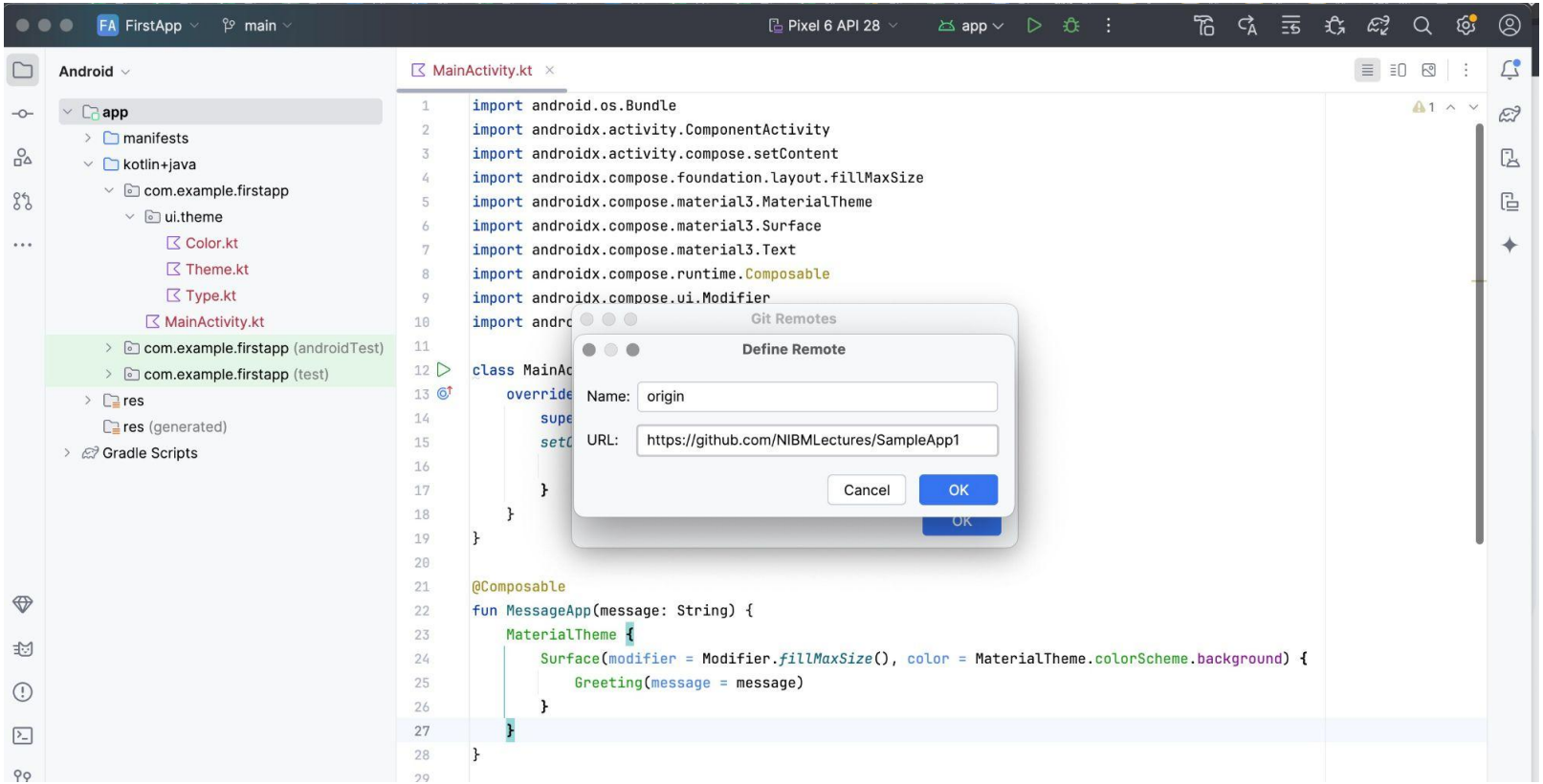
– – –

- Open Your Project in Android Studio
- Go to **VCS > Git > Remotes…**
- Click the “**+**” (Add) button.
- Enter the remote name (e.g., origin) and **repository URL** (**https://github.com/NIBMLectures/SampleApp1**)
- Click OK

Connect Android Studio Project to Remote GitHub Repository



Connect Android Studio Project to Remote GitHub Repository



What is a Branching Strategy?

- Branching Strategy is a plan for how your team will use Git branches to develop, test, and release code.

Why Use Branching?

- Develop new features without breaking the main code
- Test or fix bugs separately
- Collaborate safely

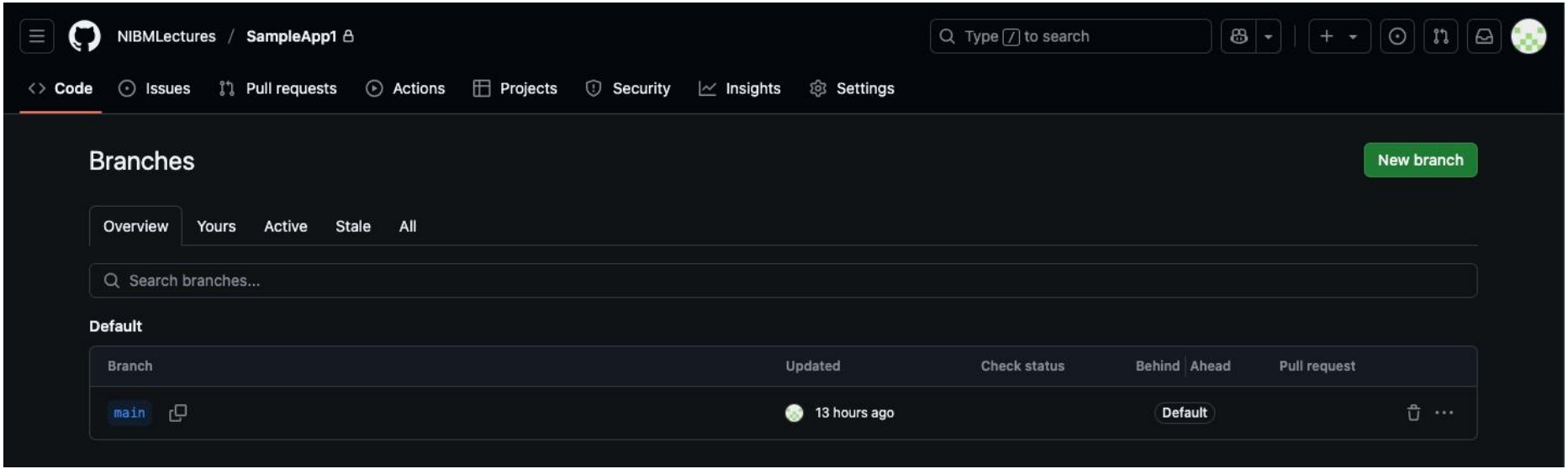
Common Branching Strategies

- **Main / Master** - Stable, production-ready version
- **Feature Branch** - For new features. Merged when complete
- **Development Branch** - Where integration happens before production
- **Hotfix Branch** - For urgent bug fixes in production
- **Release Branch** - Prepare for a new release (testing, polishing)

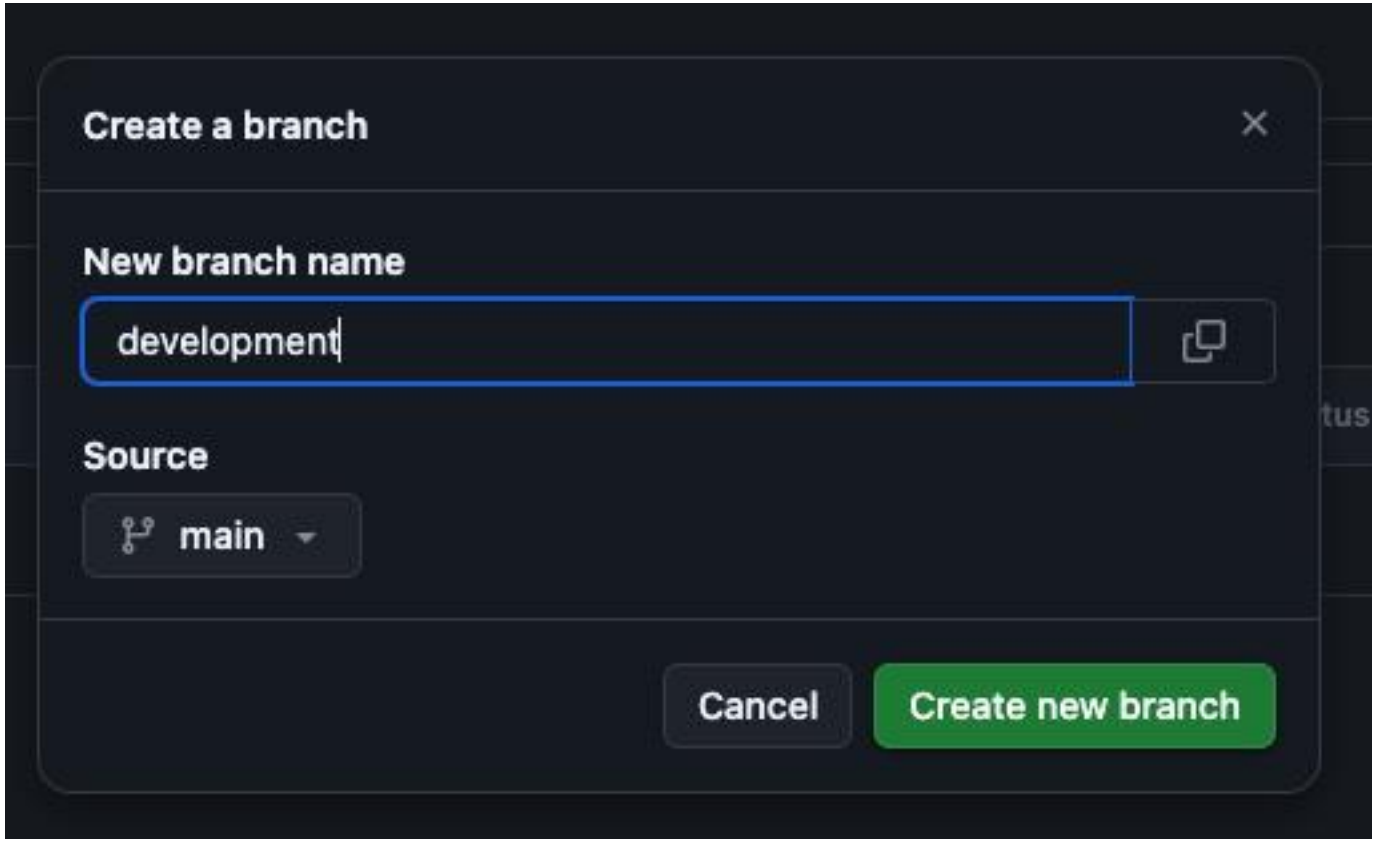
Create Branches in a GitHub Repository

- Go to your repository on GitHub
- Click the branch dropdown (next to the default branch, e.g., main)
- Type a new branch name.
- Click Create branch: [your-branch-name] from main

Create Branches in a GitHub Repository



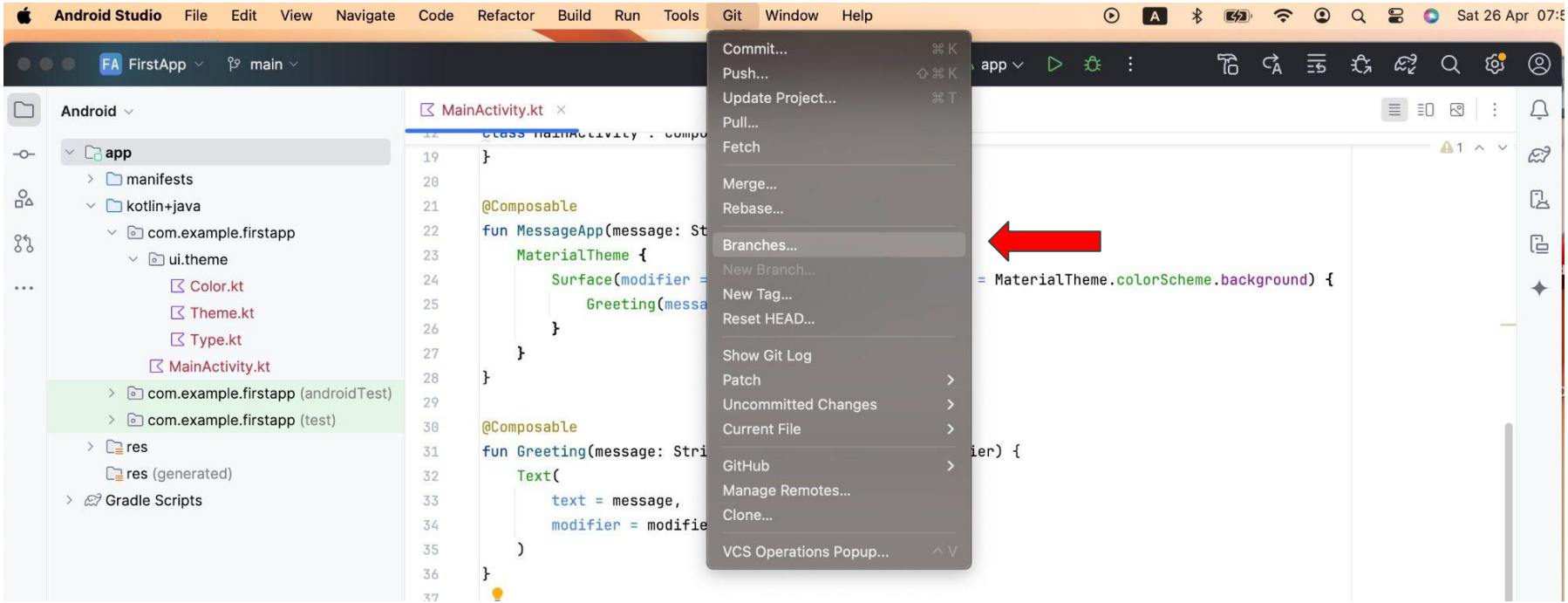
Create Branches in a GitHub Repository



Create a Branch in Android Studio (and Push to GitHub)

- In Android Studio, go to Git > Branches
- Click New Branch
- Enter a branch name (e.g., feature-login) and click Create
- After doing some changes, commit your changes
- Then go to Git > Push to push the branch to GitHub

Create a Branch in Android Studio (and Push to GitHub)



End

NIBM POWERING GREAT MINDS