Version Control & Git

Brandon Krakowsky

Penn Engineering

Version Control Systems

Version Control Systems

- Version control (or source control) is all about managing multiple versions of documents, programs, web sites, etc.
  - Almost all “real” projects use some kind of version control
  - This is essential for team projects, but also very useful for individual projects
- Some well-known version control systems are CVS, Subversion, Mercurial, and Git
  - CVS and Subversion use a “central” repository where users “check out” files, work on them, and then “check them in”
  - Mercurial and Git treat all repositories as equal
- *Distributed* systems like Mercurial and Git are newer and have basically replaced centralized systems like CVS and Subversion

Why Version Control?

- For working by yourself:
  - Gives you a “time machine” for going back to earlier versions of your work
  - Gives you great support for different versions of the same basic project (e.g. web app vs. desktop app)
- For working with others:
  - Greatly simplifies concurrent work and merging changes
- For getting an internship or job:
  - Just about all companies use some kind of version control

Why Git?

- Git is a *distributed* version-control system designed for coordinating work among programmers, but it can be used to track changes in any set of files
- Git has many advantages over earlier systems such as CVS and Subversion
  - More efficient
  - Better workflow
  - Lots of online support
  - Feature-rich IDE plug-ins
  - Etc.
- Git is increasingly the “expected” version control system for development
  - It’s the standard for all different kinds of projects
  - It’s used at all different kinds of companies
- Just Google search “Version Control” and Git is the first result you’ll find

Learning Git

- Git is often used from the command line, but it's also supported by many GUIs
- We'll learn some of the basic git commands using the Eclipse IDE
- There are other GUIs supporting git
  - SmartGit (https://www.syntevo.com/smartgit/)
  - The GitHub hosting platform has its own client (https://desktop.github.com)
  - TortoiseGit (https://code.google.com/p/tortoisegit/)
- We'll also learn how to use GitHub, a very popular code hosting platform for collaboration

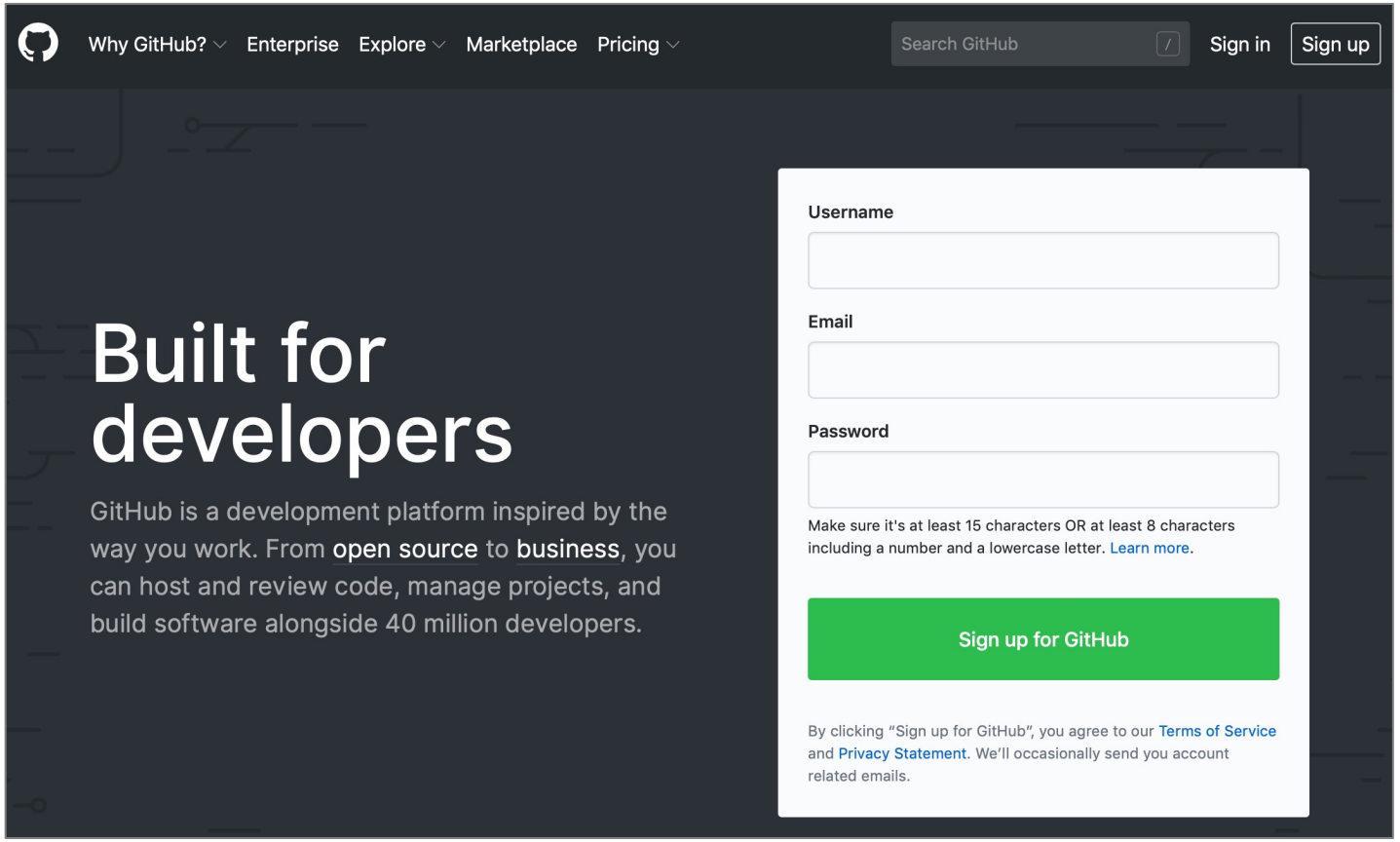
Setting Up & Configuring GitHub

Setting Up a GitHub Account

- GitHub is a code hosting platform for version control and collaboration using Git
  - We'll use it to host *remote* git repositories
- It can be used to host just about any type of project repository, including Python and Java projects
- There are other code hosting platforms using git
  - GitLab (https://about.gitlab.com)
  - Bitbucket (https://bitbucket.org)

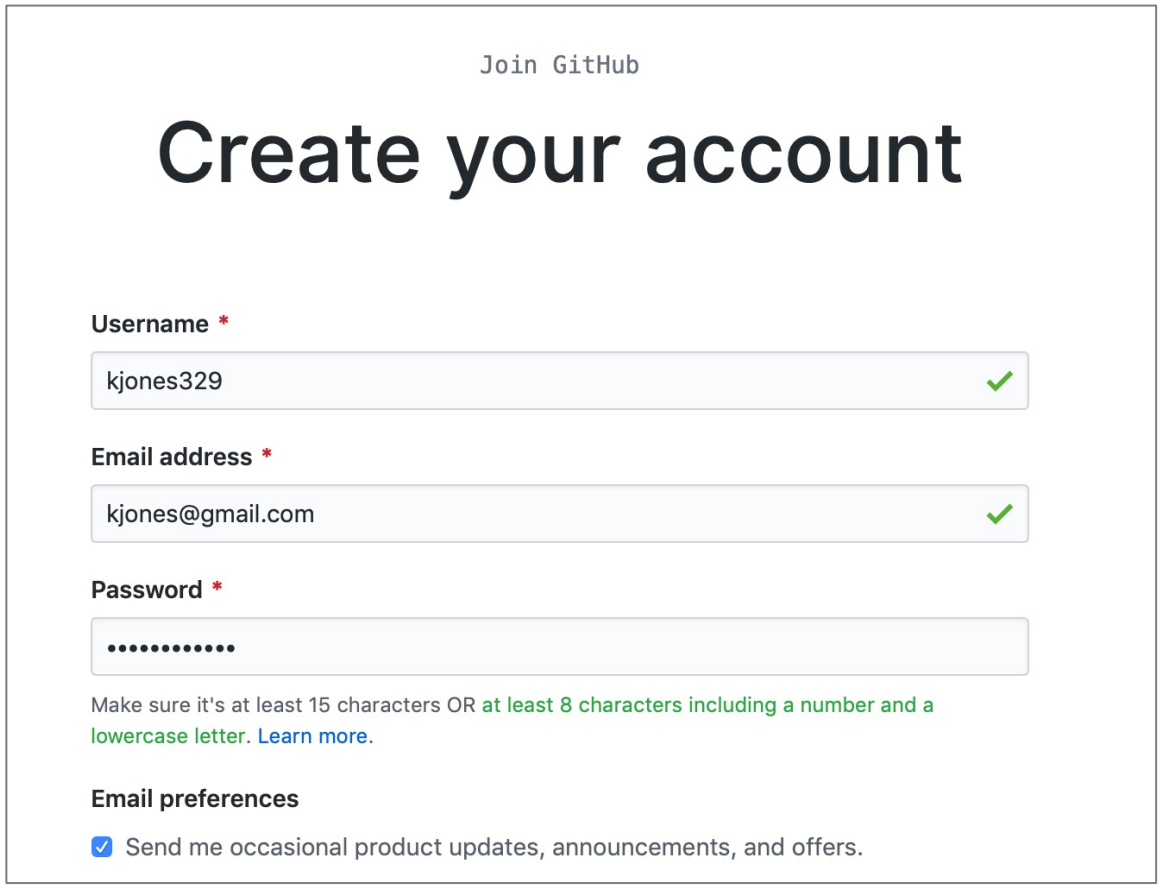
Setting Up a GitHub Account

- To create an account or log in to GitHub, go here: https://github.com



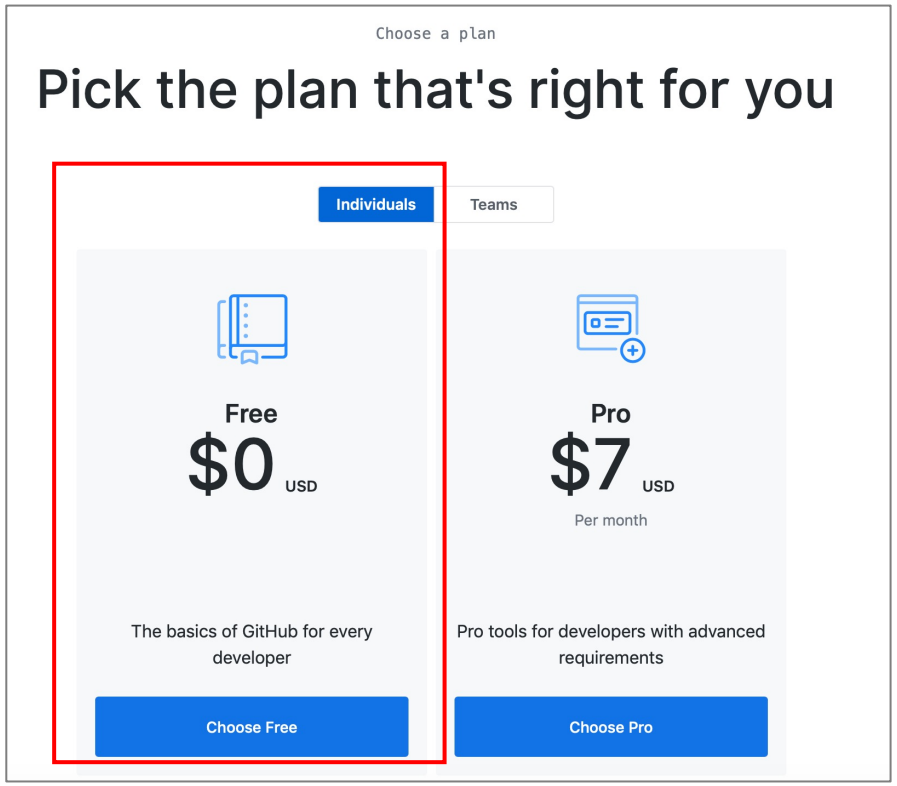
Setting Up a GitHub Account

- If you're creating a new account, enter a username, email address, and password



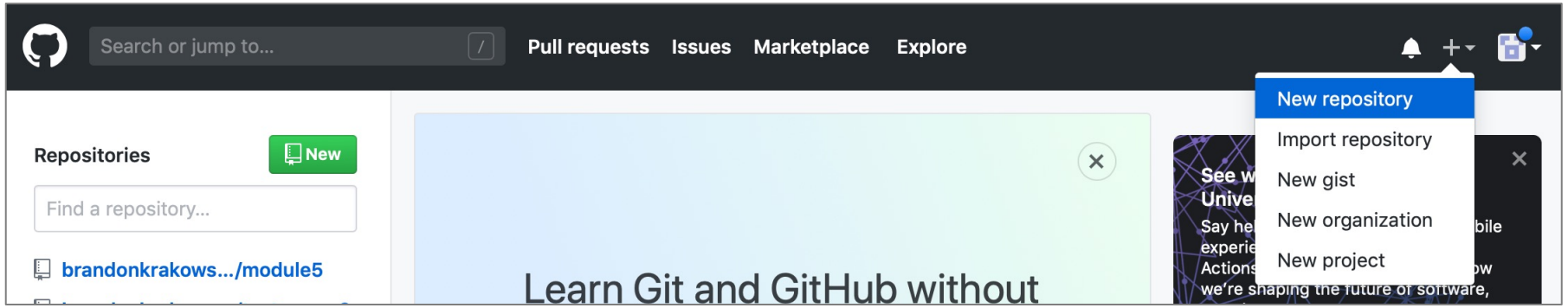
Setting Up a GitHub Account

- As an individual, you can more than likely pick the Free plan
- Go here (https://github.com/pricing) for more information about the Free vs. Pro plan, and the pricing



Creating a Repository

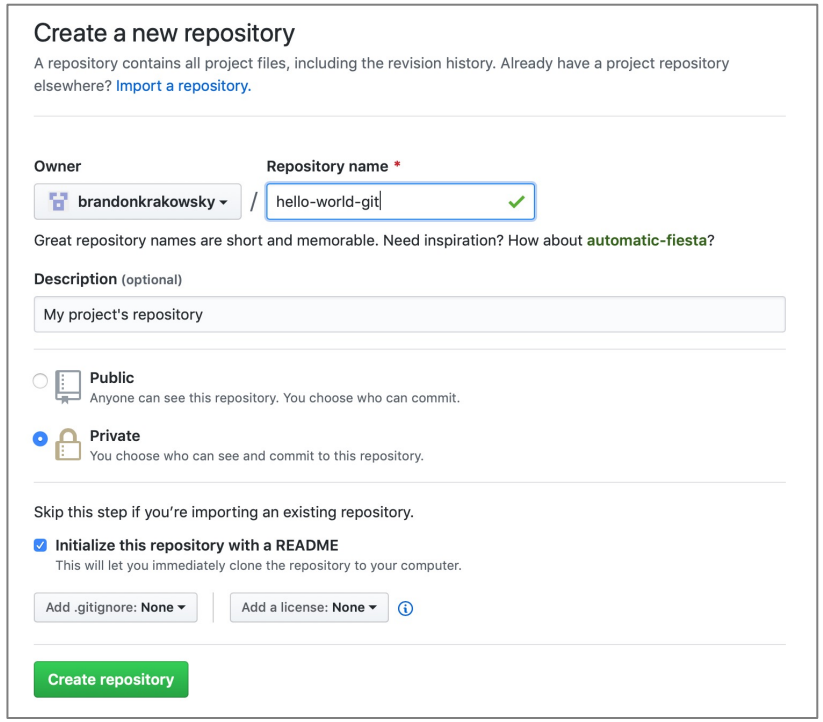
- As a first step, create a repository
  - A repository is usually used to organize a single project
- Repositories can contain folders and files, images, videos, spreadsheets, and data sets – essentially anything your project needs
- In the upper right corner, next to your avatar or identicon, click and then select “New repository”



Ref: https://guides.github.com/activities/hello-world/

Creating a Repository

- Name your repository and write a short description



- Select ”Private” so you can choose who can access your repository
- Select “Initialize this repository with a README”
  - A README is a file with information about your project

- Click “Create repository”

Ref: https://guides.github.com/activities/hello-world/

Setting Up & Configuring Eclipse

Git Support for Eclipse

- The Eclipse IDE provides support for Git
  - It allows you to easily perform Git commands from within the IDE
- If you want to learn about the usage of the Git command line, there are plenty of resources
  - You can start here (at the main Git website): https://git-scm.com/book/en/v2/Getting-Started-The-Command-Line

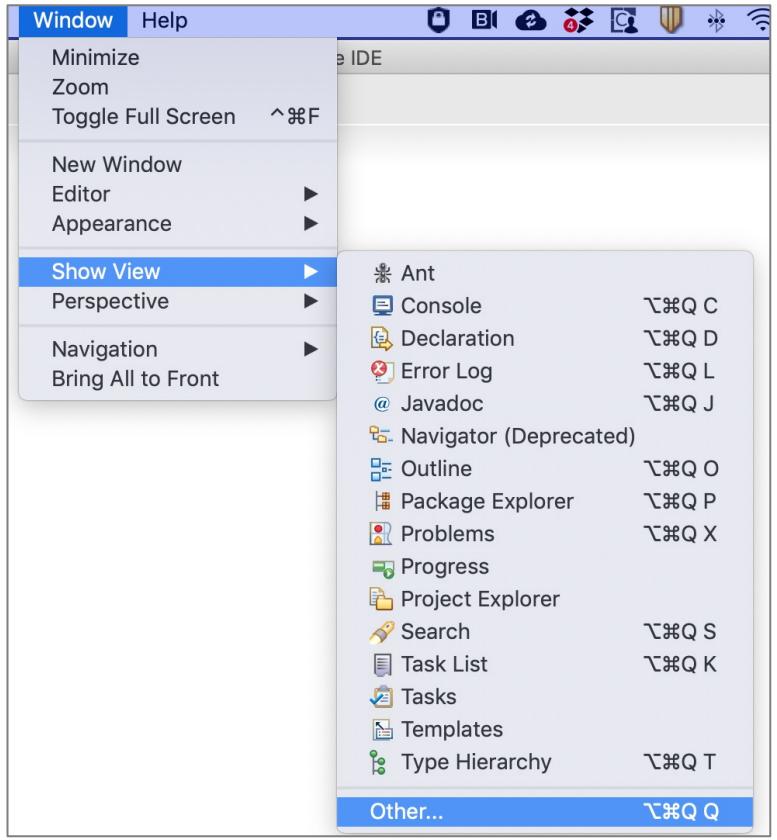
Installation of Git Support in Eclipse

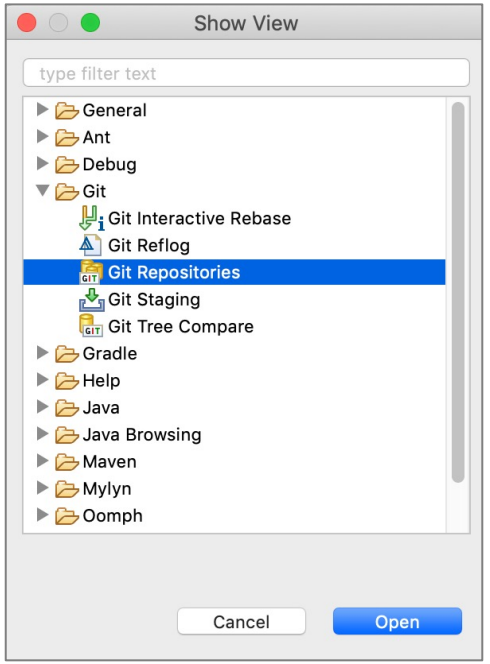
- Most Eclipse IDE distributions from Eclipse.org already contain support for Git
  - In this case no additional installation is required
- If the Git tooling is not available, you can install it via the Eclipse installation manager
  - Go to Help → Install New Software…

Ref: https://www.vogella.com/tutorials/EclipseGit/article.html

Show the Git View in Eclipse

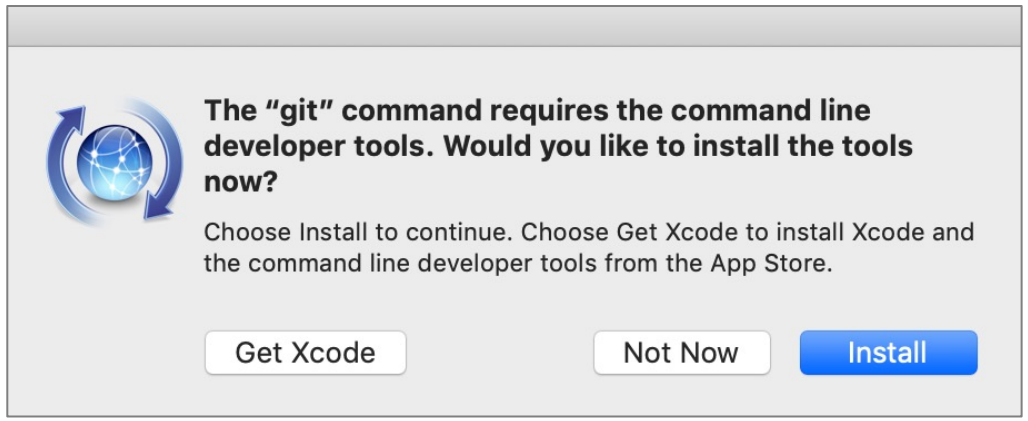
- Open the “Git Repositories” view in Eclipse by going to Window → Show View → Other… → Git → Git Repositories

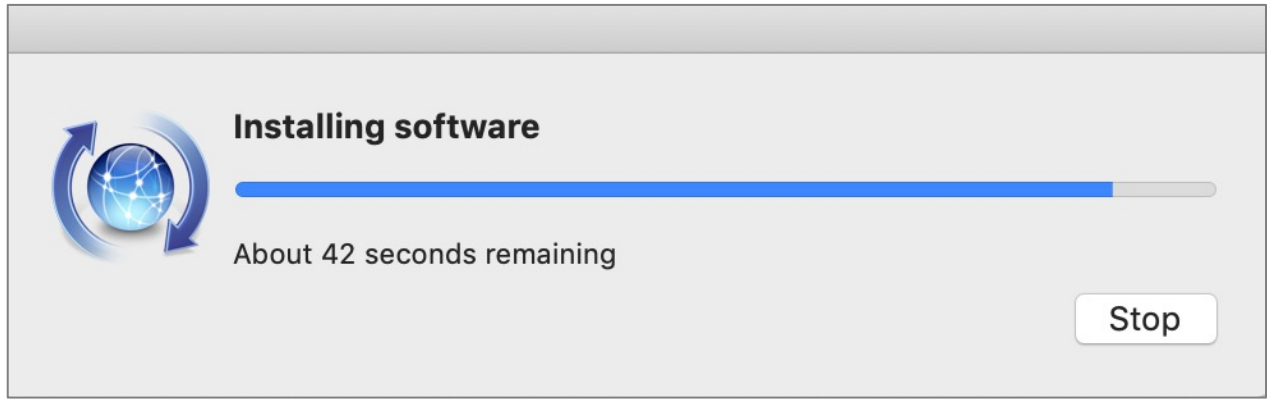




Show the Git View in Eclipse

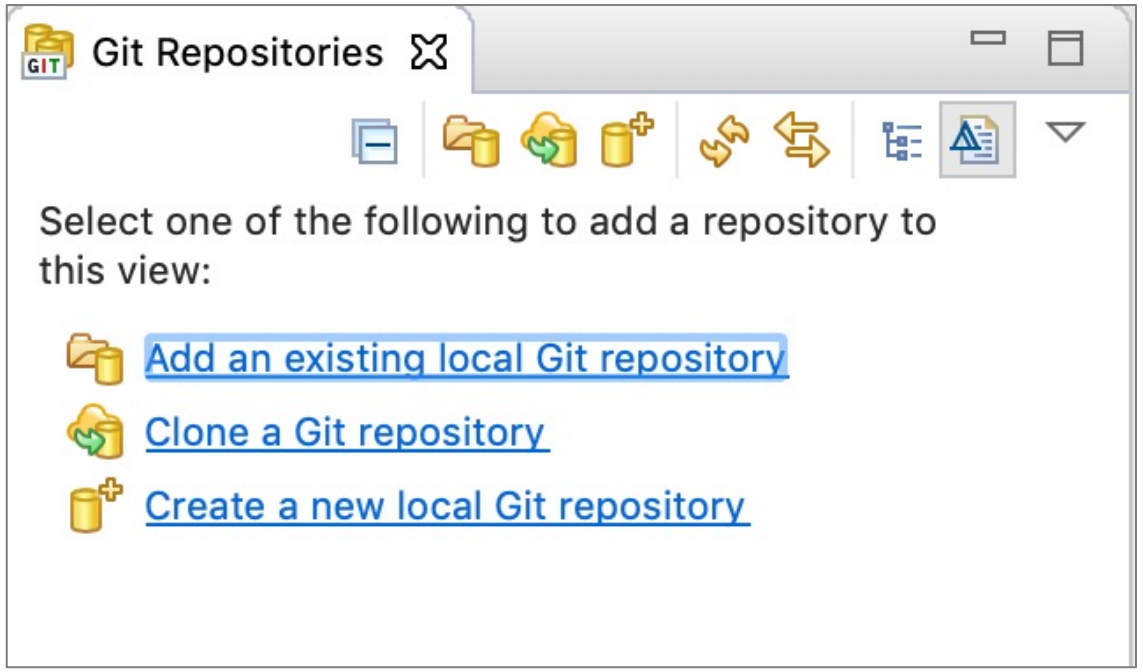
- You might be prompted to install Xcode and the Git command line tools
  - Click “Install”





Show the Git View in Eclipse

- You should see the “Git Repositories” view in Eclipse



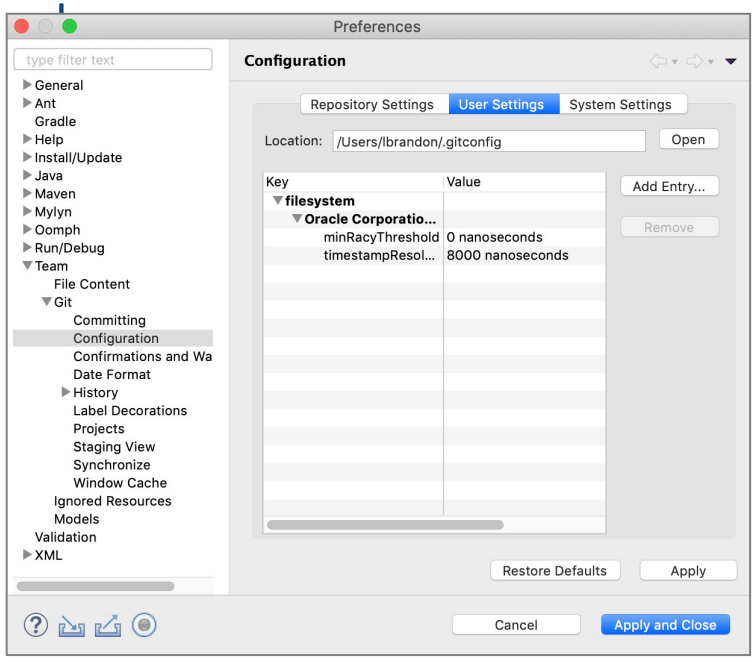
Git User Configuration

- The Eclipse IDE uses the same configuration files as the Git command line tools
  - This makes it easier to use the Eclipse IDE and the command line for Git, interchangeably

Ref: https://www.vogella.com/tutorials/EclipseGit/article.html

Configure Email & User for Git

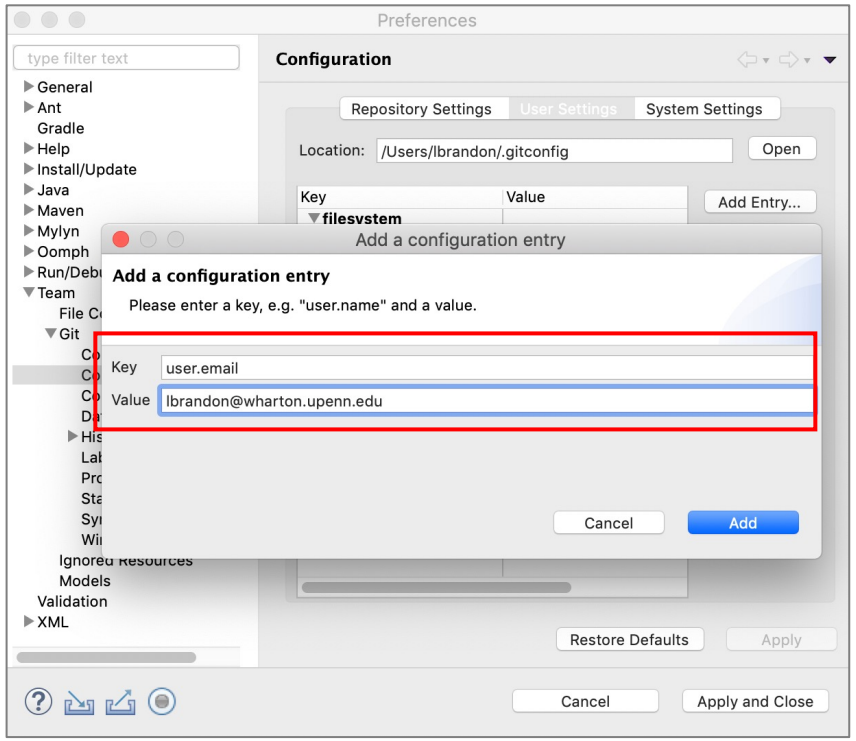
- Select Eclipse → Preferences → Team → Git → Configuration
- Ensure that your full name and email is available in the user settings
  - Again, the Eclipse IDE uses the same settings as the Git command line, this might already be



Ref: https://www.vogella.com/tutorials/EclipseGit/article.html

Configure Email & User for Git

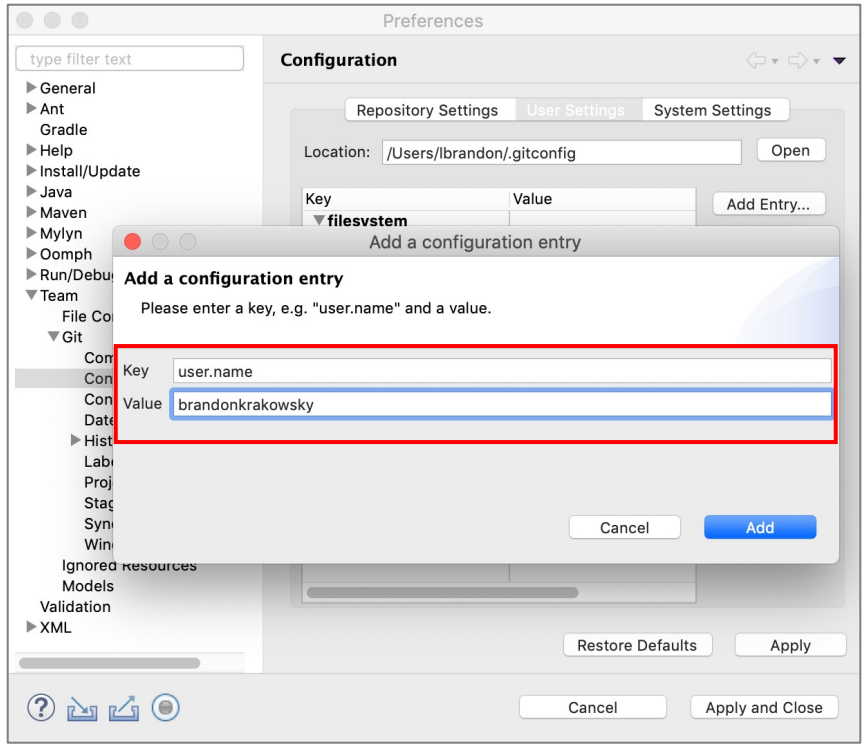
- Press “Add Entry…” to add a user.email key
  - This email will be used when making and committing changes to git repositories



Ref: https://www.vogella.com/tutorials/EclipseGit/article.html

Configure Email & User for Git

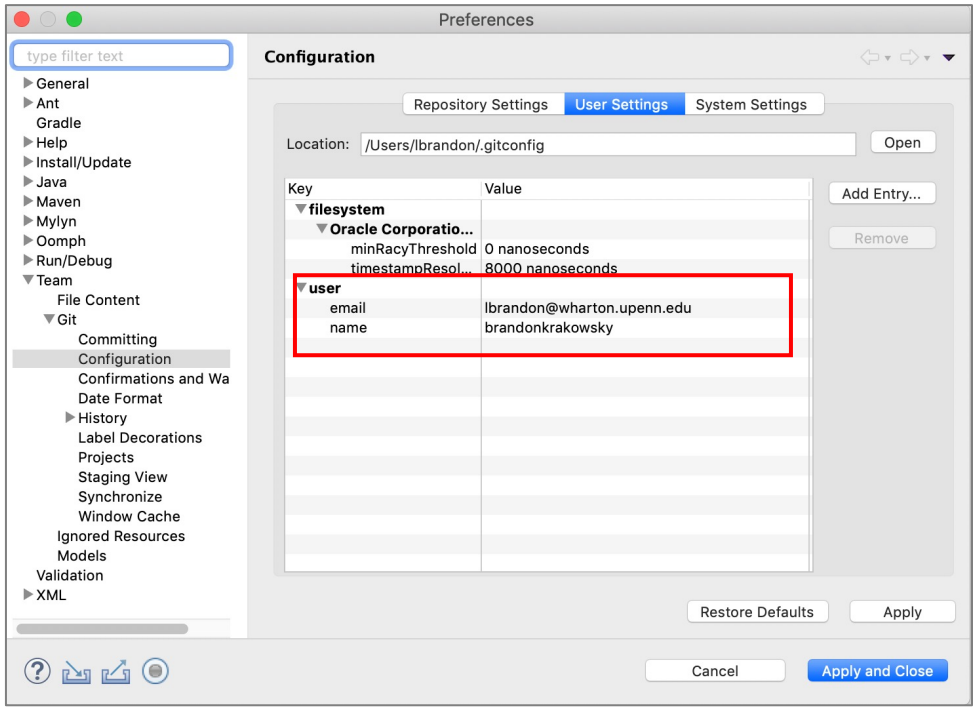
- Press “Add Entry…” again to add a user.name key
  - Again, this username will be used when making and committing changes to git repositories



Ref: https://www.vogella.com/tutorials/EclipseGit/article.html

Configure Email & User for Git – .gitconfig

- You'll see the keys added to your .gitconfig
  - The .gitconfig is a hidden file in your Git repository
  - It's commonly used for storing project settings, etc.



Ref: https://www.vogella.com/tutorials/EclipseGit/article.html

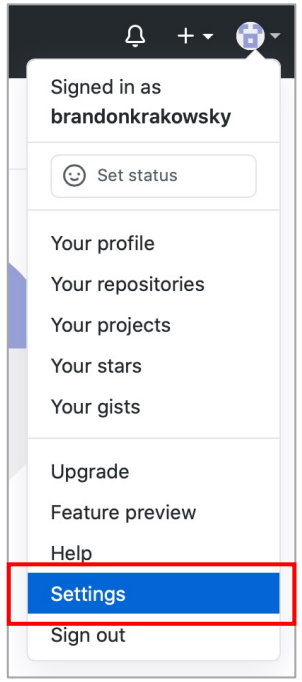
Create Personal Access Token to Authenticate to GitHub

- To work with a remote repository, you need to authenticate to GitHub
  - To do this, you'll need to create a Personal Access Token (PAT)
- First, make sure the email address associated with your account in GitHub is verified

Ref: https://docs.github.com/en/free-pro-team@latest/github/authenticating-to-github/creating-a-personal-access-token

Create Personal Access Token to Authenticate to GitHub

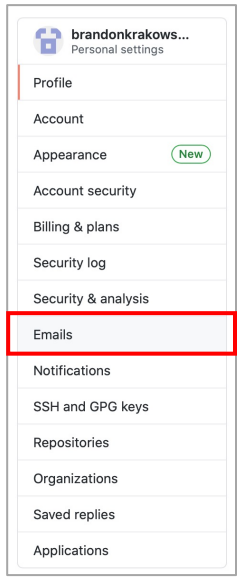
- To work with a remote repository, you need to authenticate to GitHub
  - To do this, you'll need to create a Personal Access Token (PAT)
- First, make sure the email address associated with your account in GitHub is verified
- In the upper-right corner of any page, click your profile photo, then click "Settings"



Ref: https://docs.github.com/en/free-pro-team@latest/github/authenticating-to-github/creating-a-personal-access-token

Create Personal Access Token to Authenticate to GitHub

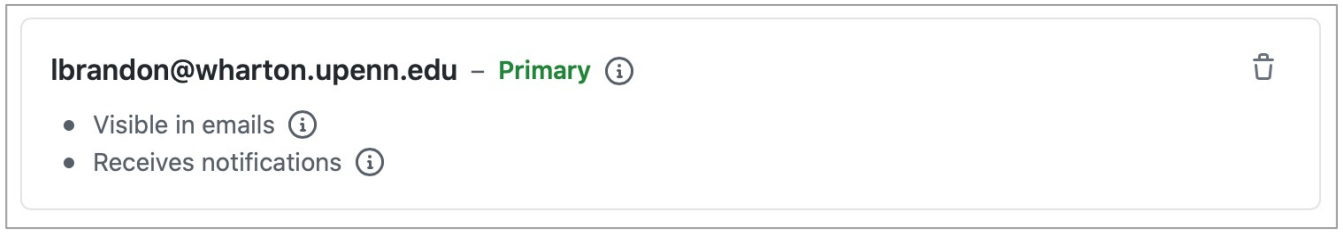
- To work with a remote repository, you need to authenticate to GitHub
  - To do this, you'll need to create a Personal Access Token (PAT)
- First, make sure the email address associated with your account in GitHub is verified
- In the upper-right corner of any page, click your profile photo, then click "Settings"
- In the left sidebar, click "Emails"



Ref: https://docs.github.com/en/free-pro-team@latest/github/authenticating-to-github/creating-a-personal-access-token

Create Personal Access Token to Authenticate to GitHub

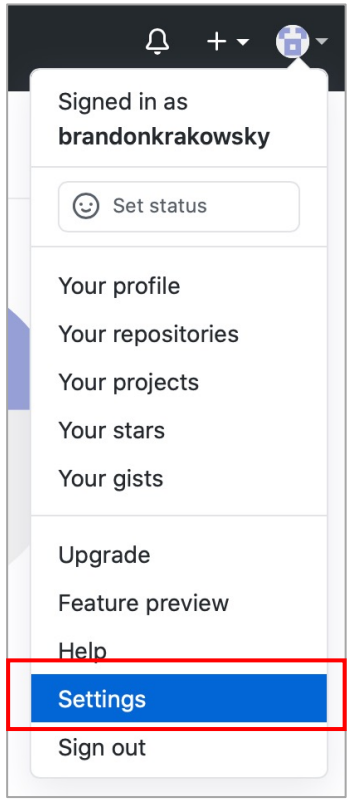
- To work with a remote repository, you need to authenticate to GitHub
  - To do this, you'll need to create a Personal Access Token (PAT)
- First, make sure the email address associated with your account in GitHub is verified
- In the upper-right corner of any page, click your profile photo, then click "Settings"
- In the left sidebar, click "Emails"
- Confirm that your email is verified. If not, resend the GitHub verification email and click the link to confirm.



Ref: https://docs.github.com/en/free-pro-team@latest/github/authenticating-to-github/creating-a-personal-access-token

Create Personal Access Token to Authenticate to GitHub

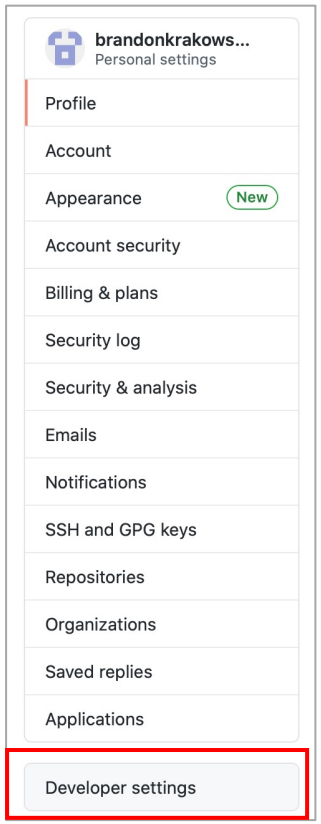
- To create a Personal Access Token (PAT), click “Settings” under your profile photo



Ref: https://docs.github.com/en/free-pro-team@latest/github/authenticating-to-github/creating-a-personal-access-token

Authenticate Using a Personal Access Token (PAT)

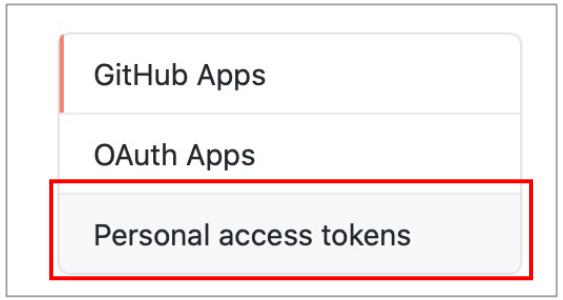
- In the left sidebar, click “Developer settings”



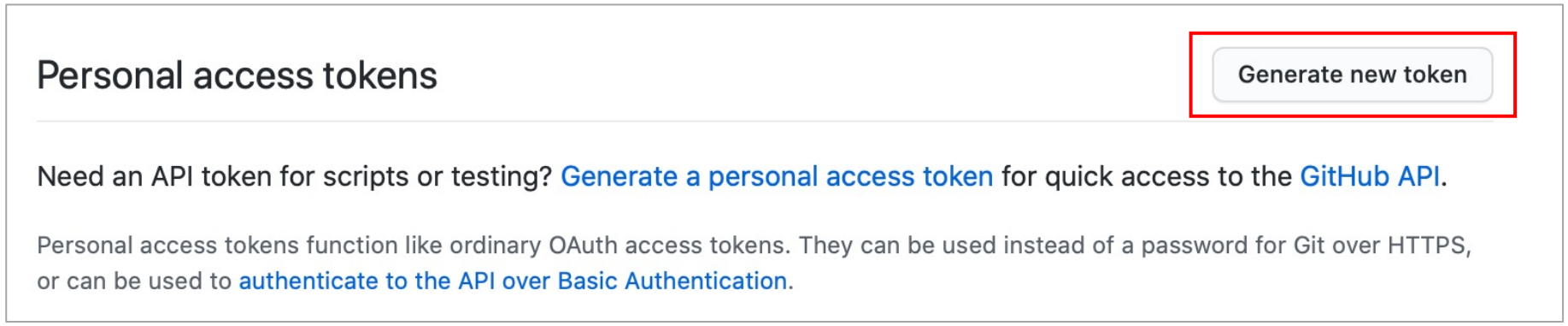
Ref: https://docs.github.com/en/free-pro-team@latest/github/authenticating-to-github/creating-a-personal-access-token

Authenticate Using a Personal Access Token (PAT)

- In the left sidebar, click “Personal access tokens”



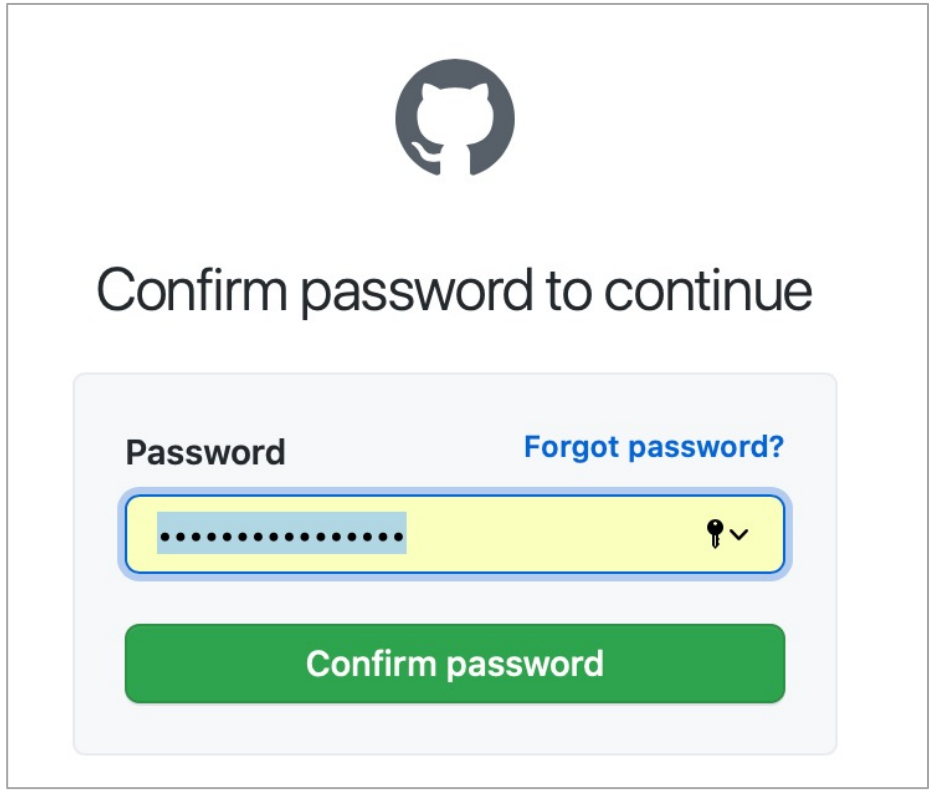
- Click “Generate new token”



Ref: https://docs.github.com/en/free-pro-team@latest/github/authenticating-to-github/creating-a-personal-access-token

Authenticate Using a Personal Access Token (PAT)

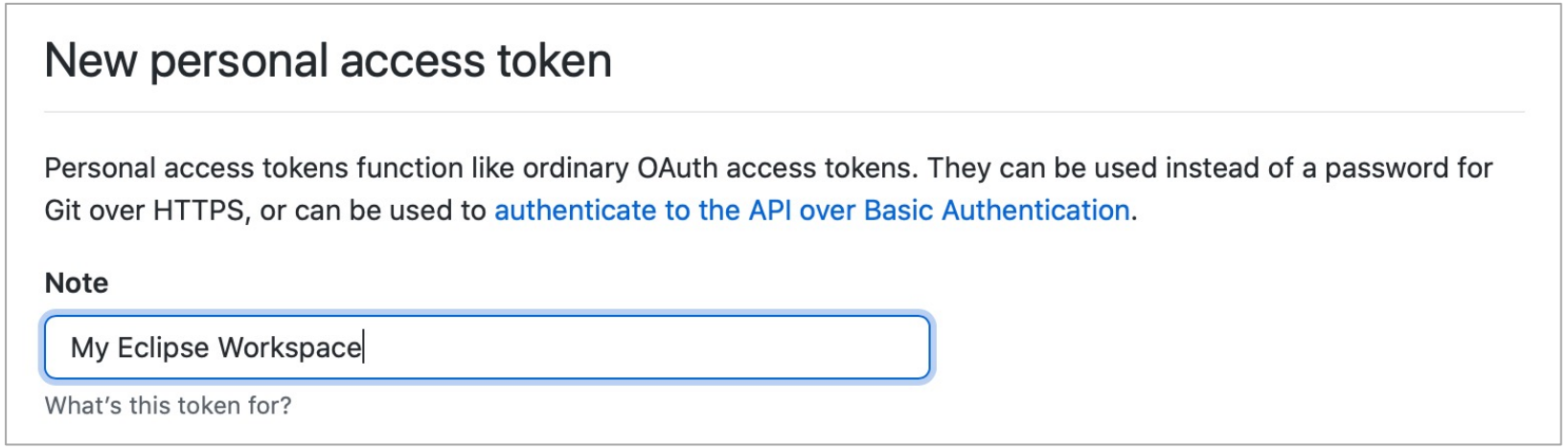
- If prompted, confirm your password to continue



Ref: https://docs.github.com/en/free-pro-team@latest/github/authenticating-to-github/creating-a-personal-access-token

Authenticate Using a Personal Access Token (PAT)

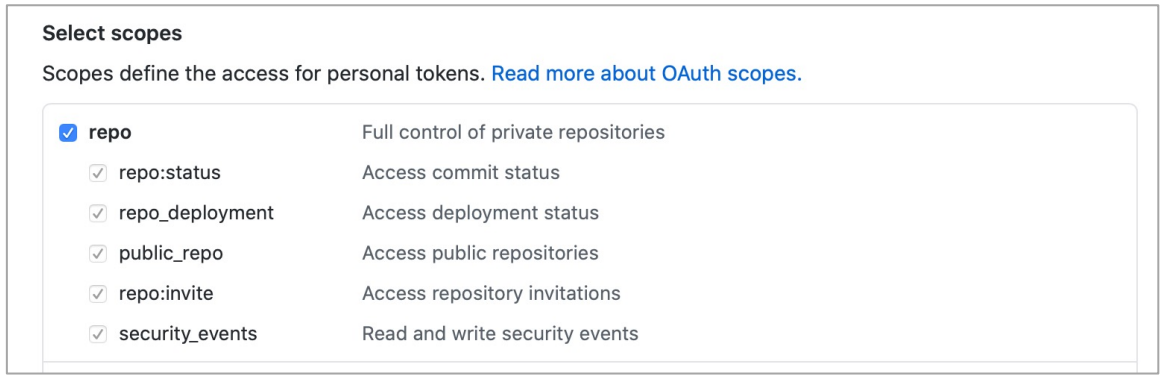
- Give your token a descriptive name



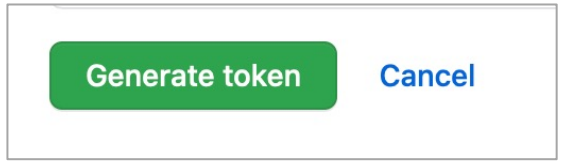
Ref: https://docs.github.com/en/free-pro-team@latest/github/authenticating-to-github/creating-a-personal-access-token

Authenticate Using a Personal Access Token (PAT)

- Select the scopes, or permissions, you'd like to grant this token. To use your token to access repositories, select **repo**.



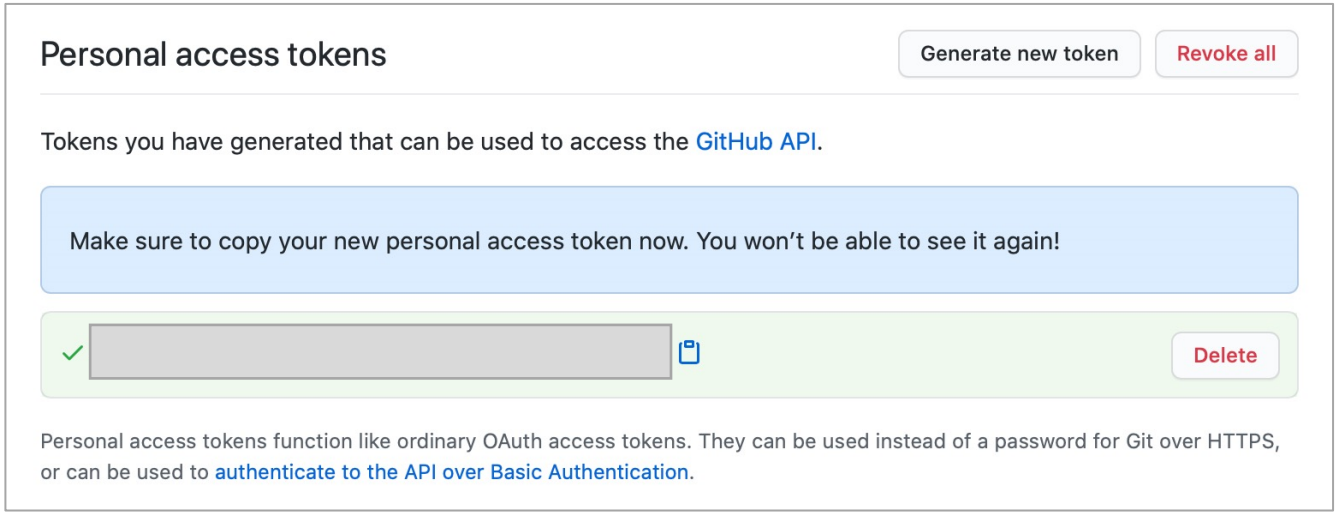
- Click “Generate token”



Ref: https://docs.github.com/en/free-pro-team@latest/github/authenticating-to-github/creating-a-personal-access-token

Authenticate Using a Personal Access Token (PAT)

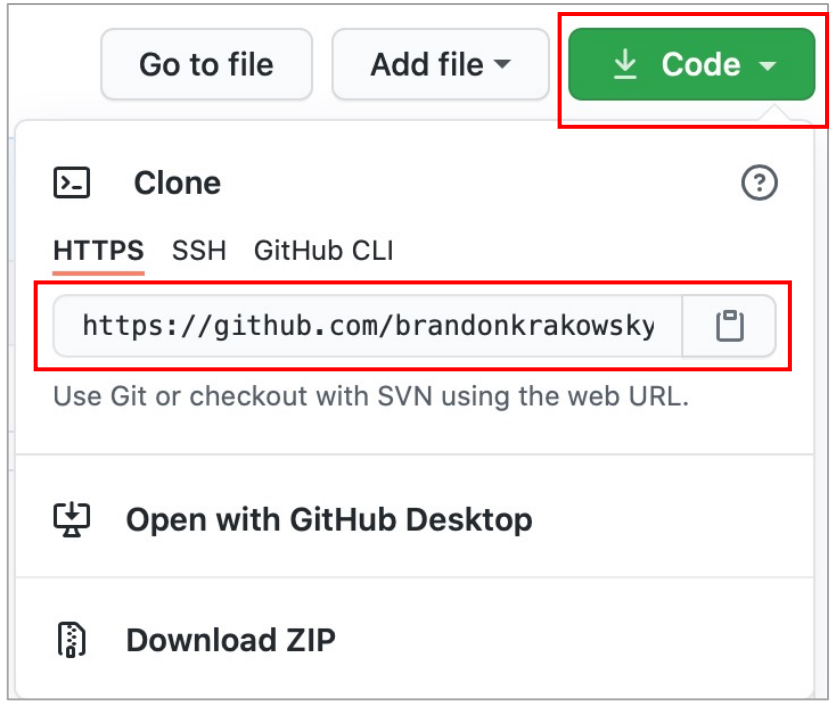
- Copy your token



Ref: https://docs.github.com/en/free-pro-team@latest/github/authenticating-to-github/creating-a-personal-access-token

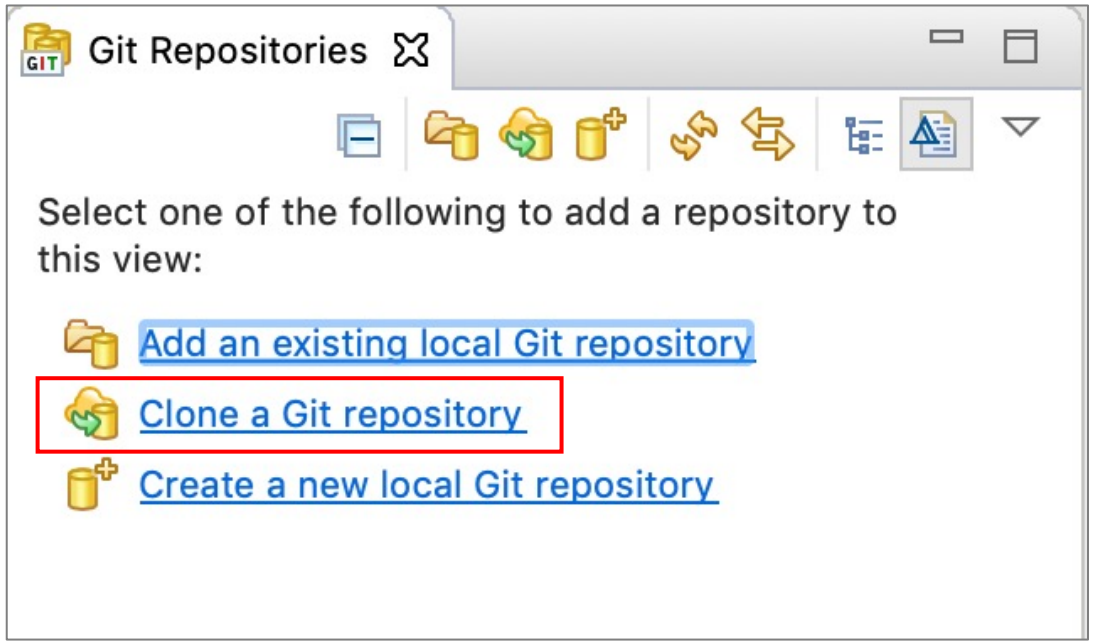
Cloning a Repository from GitHub

- To work with a remote repository, clone it to your local machine
- In GitHub, go to your project's repository and click the "Code" button
- To clone with HTTPS, copy the given web URL



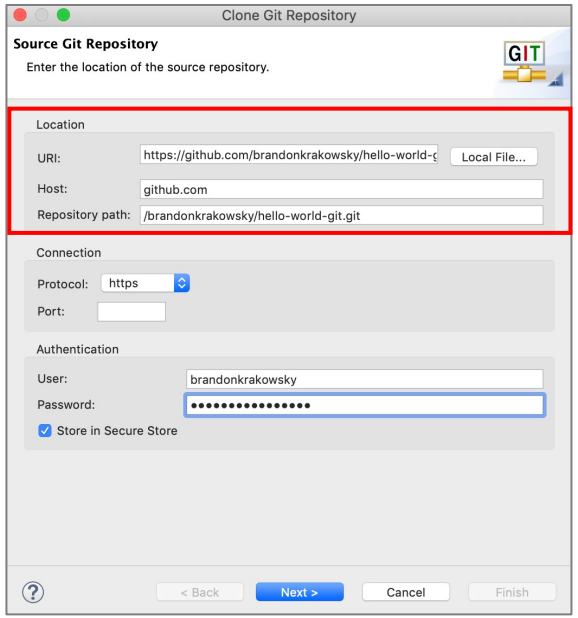
Cloning a Repository in Eclipse

- To clone a remote repository and add it to the “Git Repositories” view, click the “Clone a Git repository” link
- You can also click the “Clone a Git Repository” button



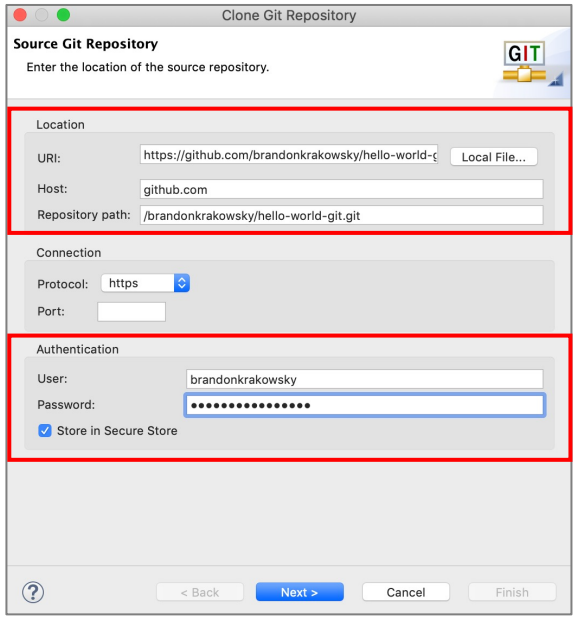
Cloning a Repository in Eclipse

- Paste the web URL for the remote git repository into the Location URI field
  - This should automatically populate other Location fields



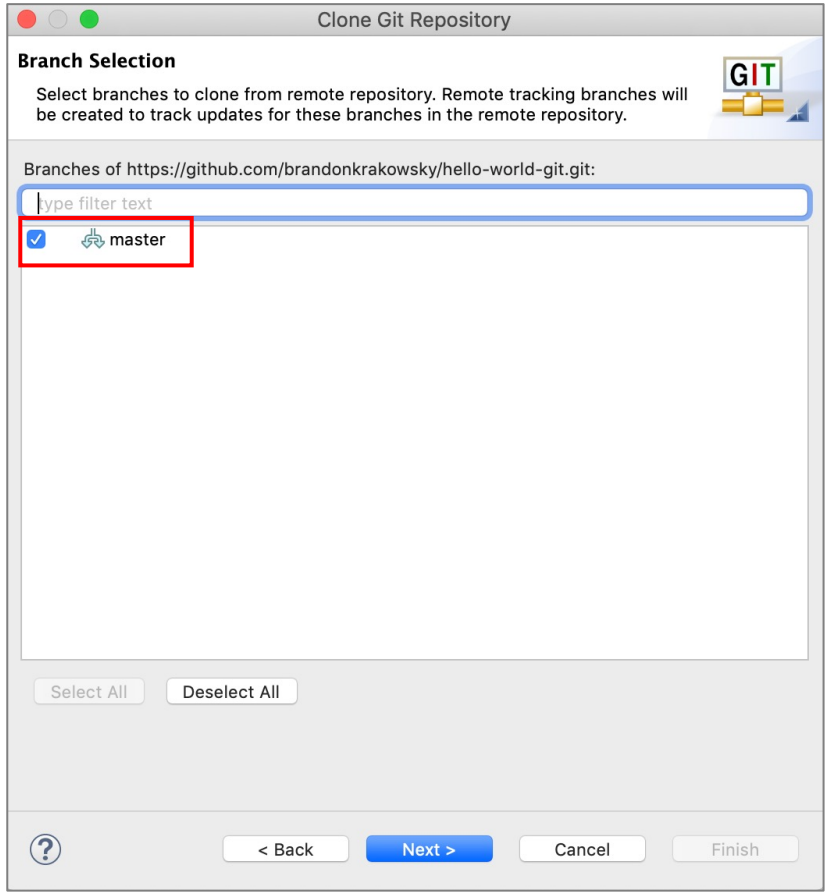
Cloning a Repository in Eclipse

- Paste the web URL for the remote git repository into the Location URI field
  - This should automatically populate other Location fields
- Enter the Authentication User and Personal Access Token (PAT) associated with your GitHub account
  - Optionally select the “Store in Secure Store” option



Cloning a Repository in Eclipse

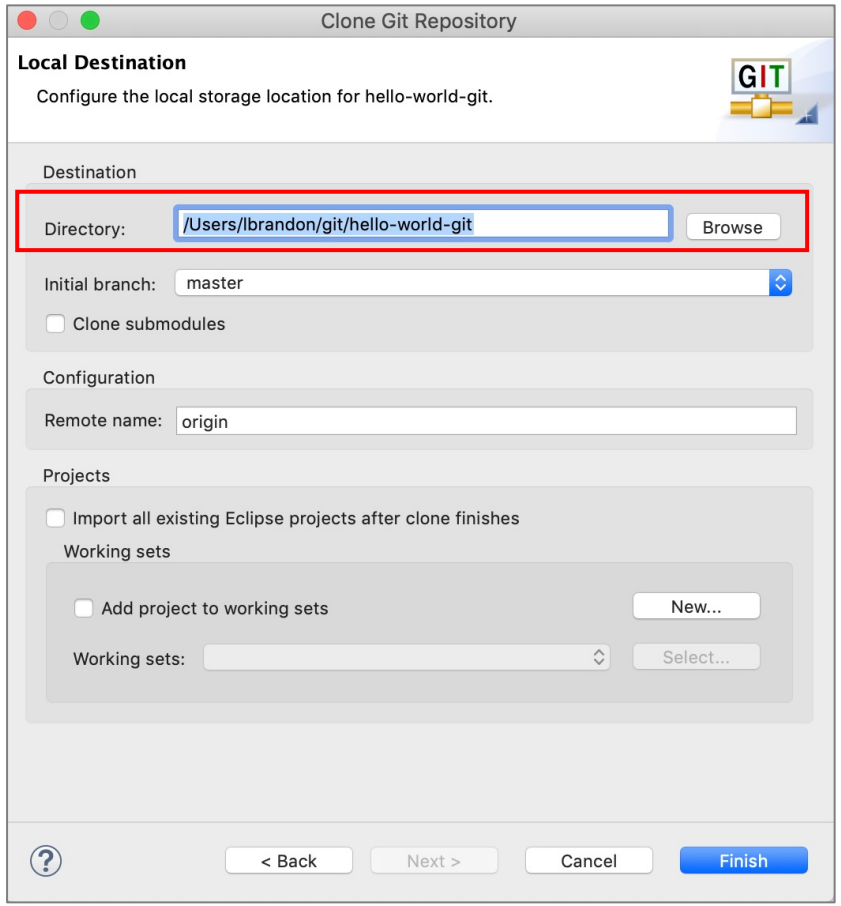
- Select the default master branch



- Branching is the way to work on different versions of a repository at one time
- By default your repository has one branch named master which is considered to be the definitive branch

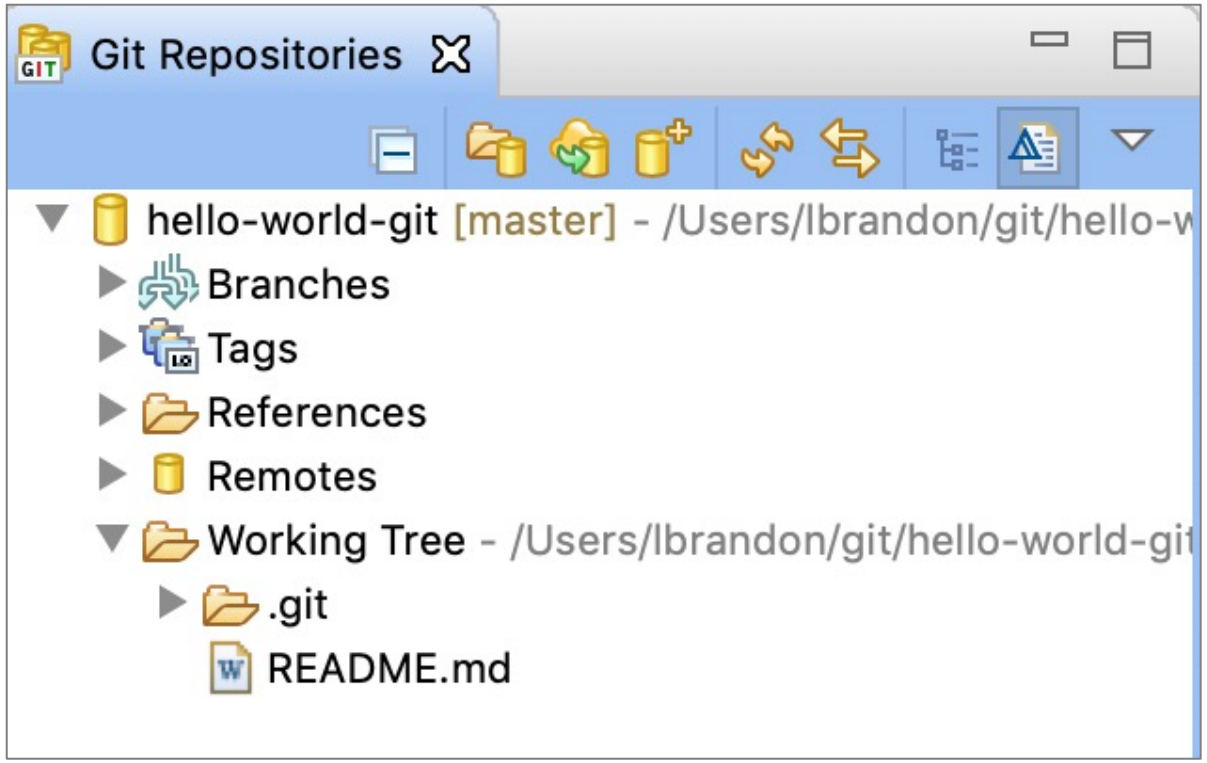
Cloning a Repository in Eclipse

- Configure the local storage location for the remote git repository



Cloning a Repository in Eclipse

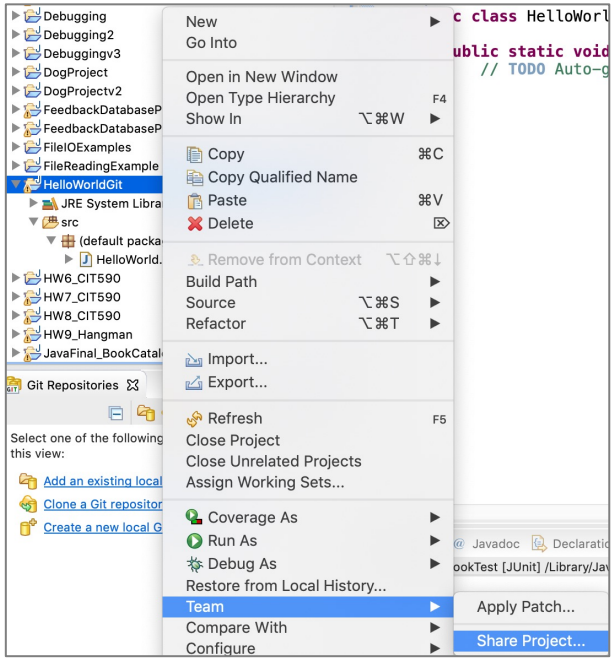
- You'll see the newly cloned local repository in the "Git Repositories" view

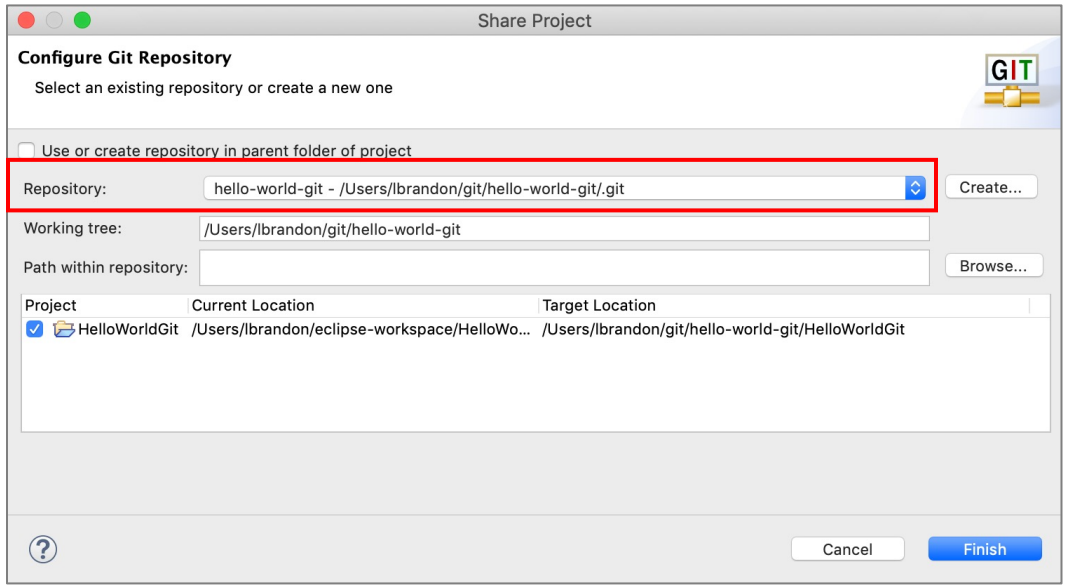


Adding an Eclipse Project to Git

Adding an Eclipse Project to the Local Repository

- Create a new Java Project, add a file, and put the entire project under version control
- Right click on the project and go to Team → Share Project

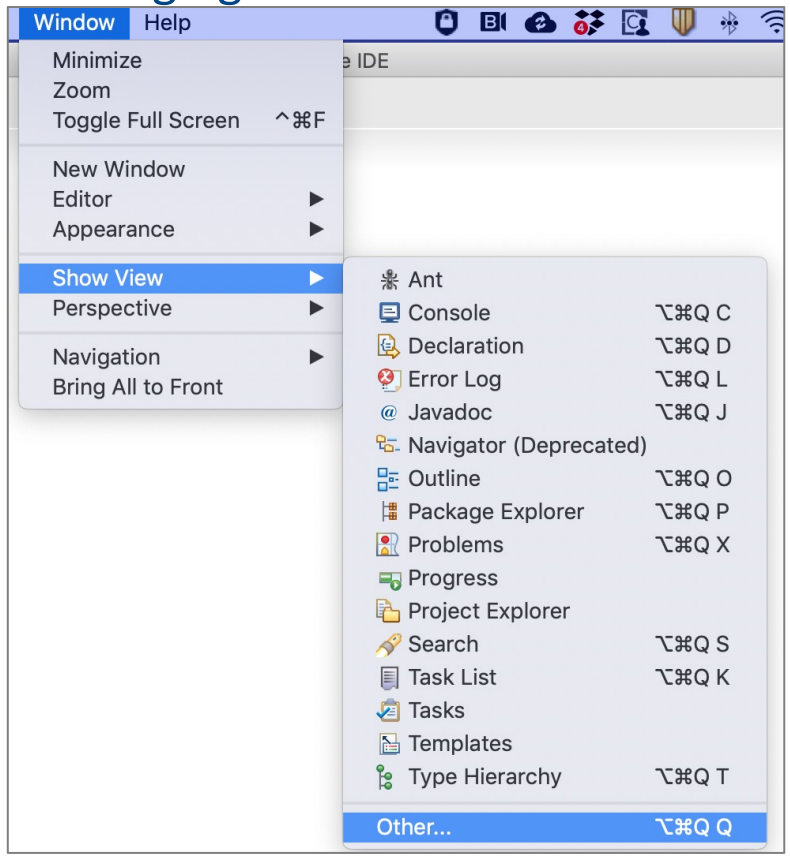


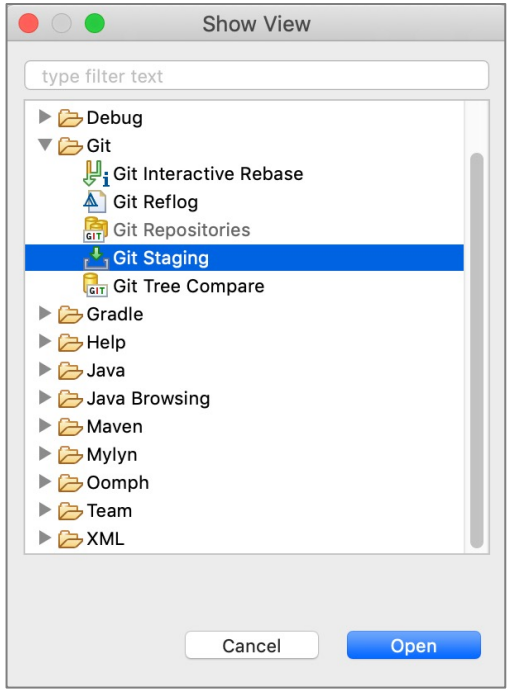


- Select your existing local git repository from the drop-down and click “Finish”

Making Initial Git Commit via the Git Staging View

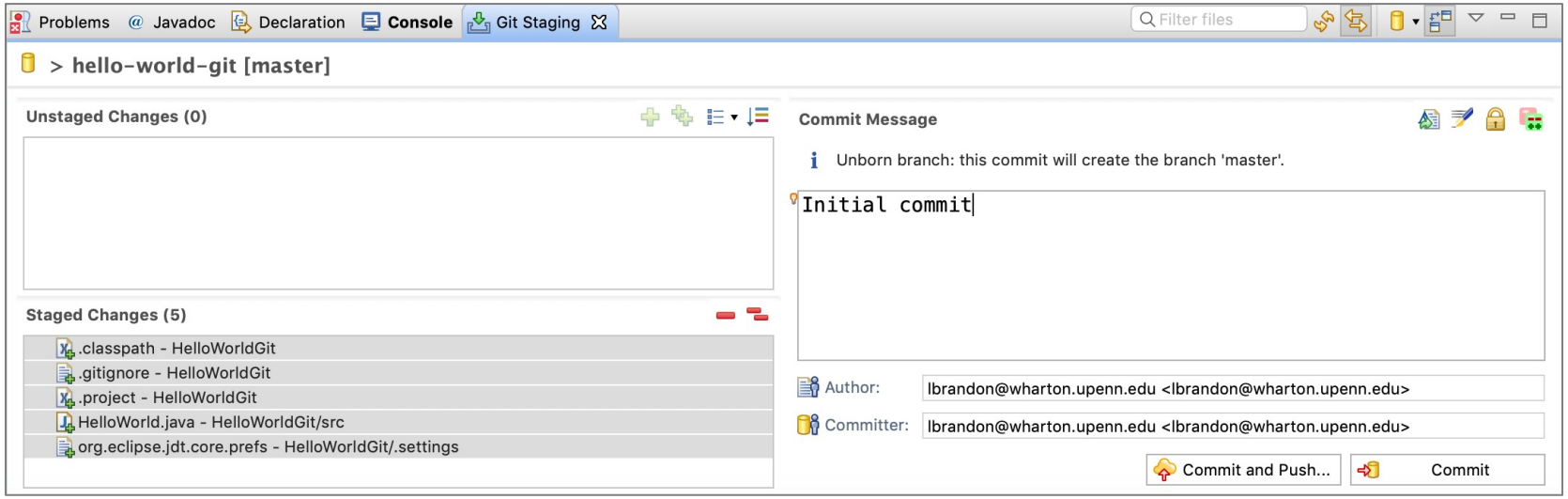
- Open the “Git Staging” view in Eclipse by going to Window → Show View → Other… → Git → Git Staging





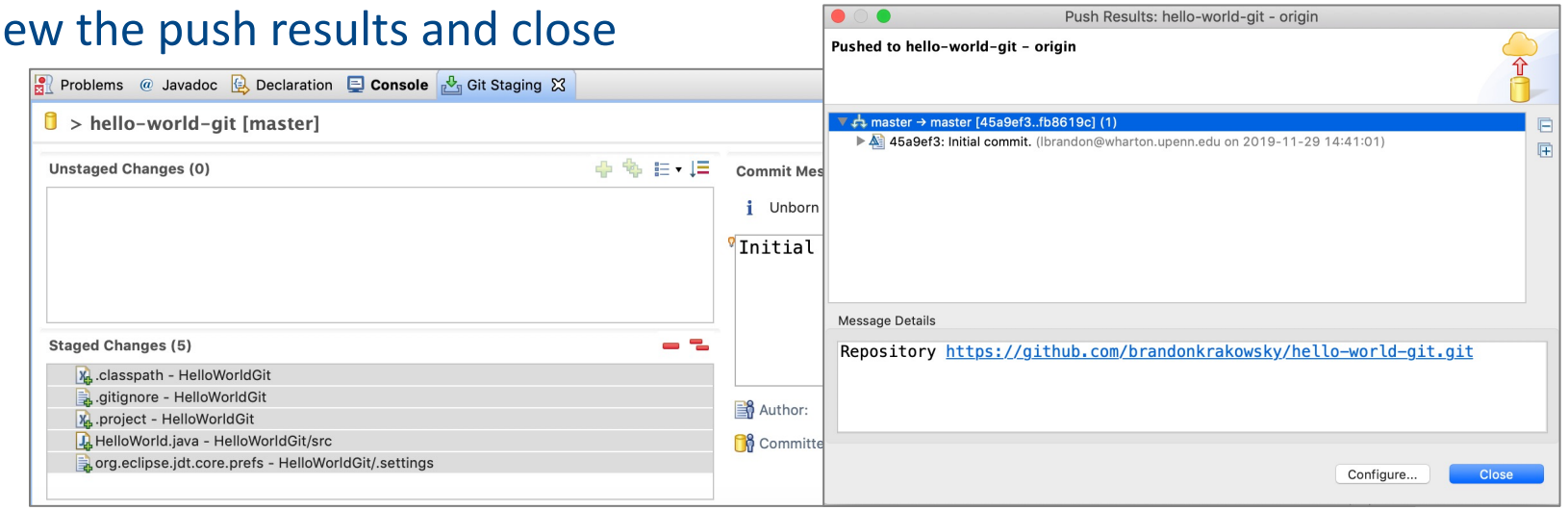
Making Initial Git Commit

- Drag all of the files into the “Staged Changes” area, type a meaningful commit message, and press “Commit and Push”
  - A commit is when you tell git that a change (or addition) you have made is ready to be included in the project
  - Push is when you upload the changes to the remote repository
  - The commit message should state specifically what you have done



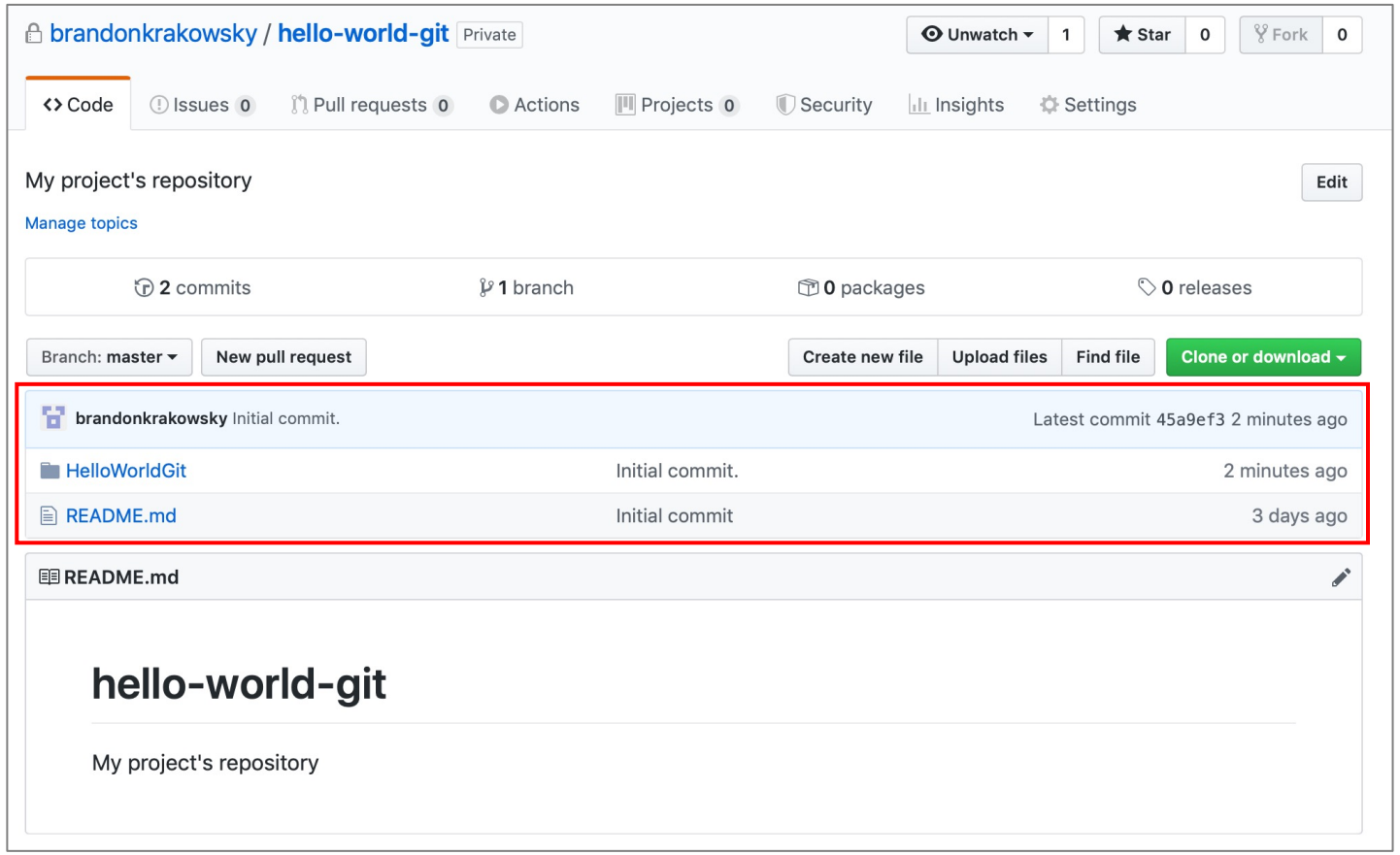
Making Initial Git Commit

- Drag all of the files into the “Staged Changes” area, type a meaningful commit message, and press “Commit and Push”
  - A commit is when you tell git that a change (or addition) you have made is ready to be included in the project
  - Push is when you upload the changes to the remote repository
  - The commit message should state specifically what you have done
- Review the push results and close



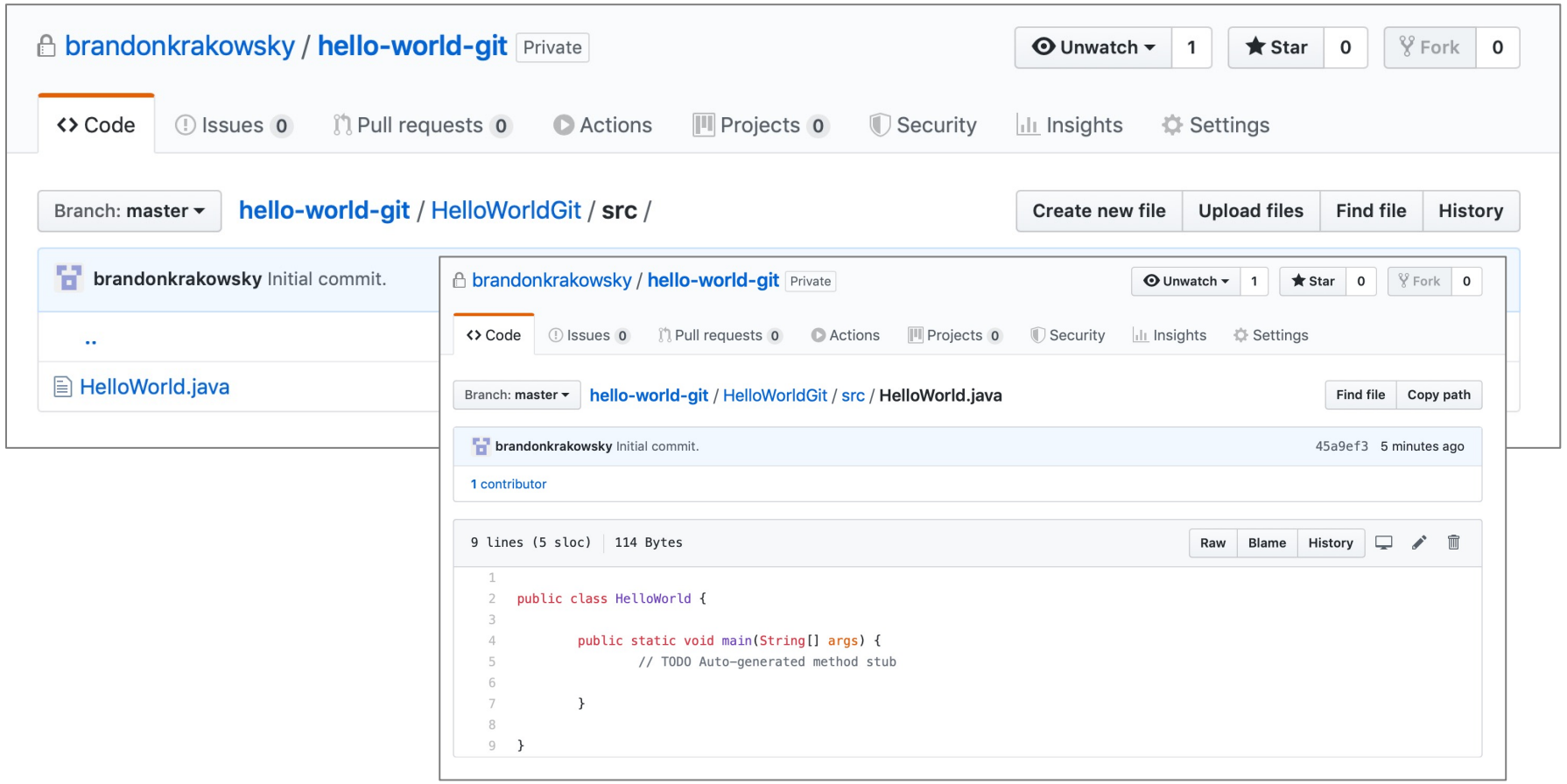
Viewing Changes in GitHub

- In GitHub, refresh your project's repository and note the initial commit and project files



Viewing Changes in GitHub

- Navigate through the folder structure and view the files



Making a Change & Committing It

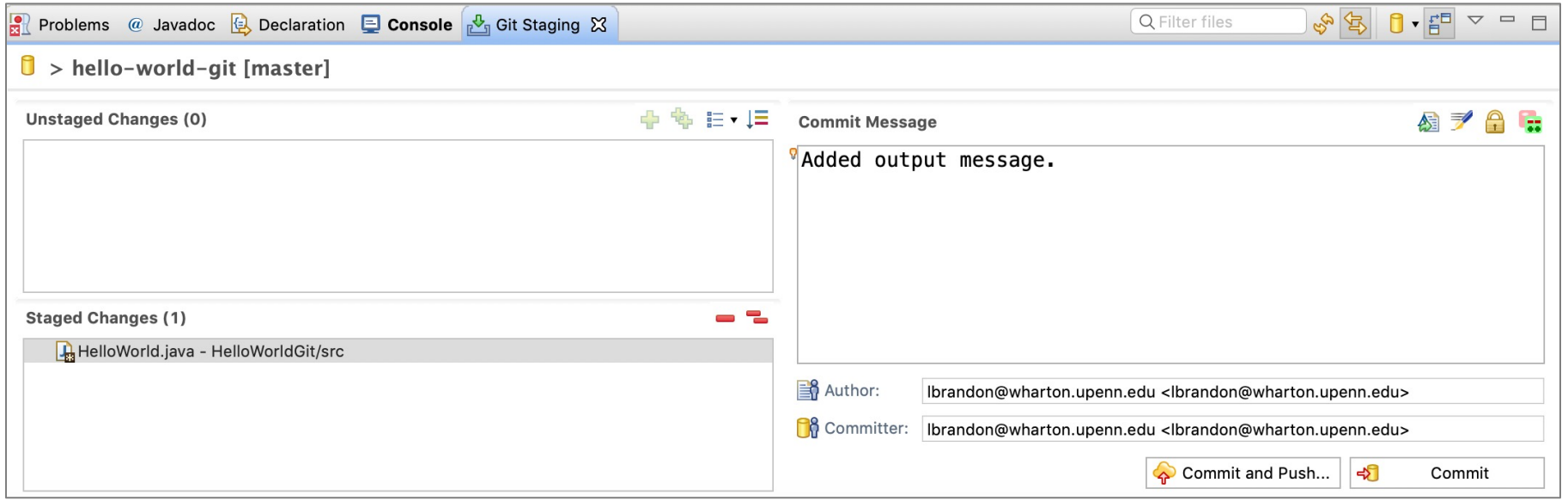
- Make a change to your project
  - Any change will suffice, add a System.out.println statement

HelloWorld.java

```
1
2  /**
3   * Hello World project!  Under version control via Git.
4   * @author lbrandon
5   *
6   */
7  public class HelloWorld {
8
9      public static void main(String[] args) {
10
11         System.out.println("Hello World project under version control!");
12     }
13 }
14
```

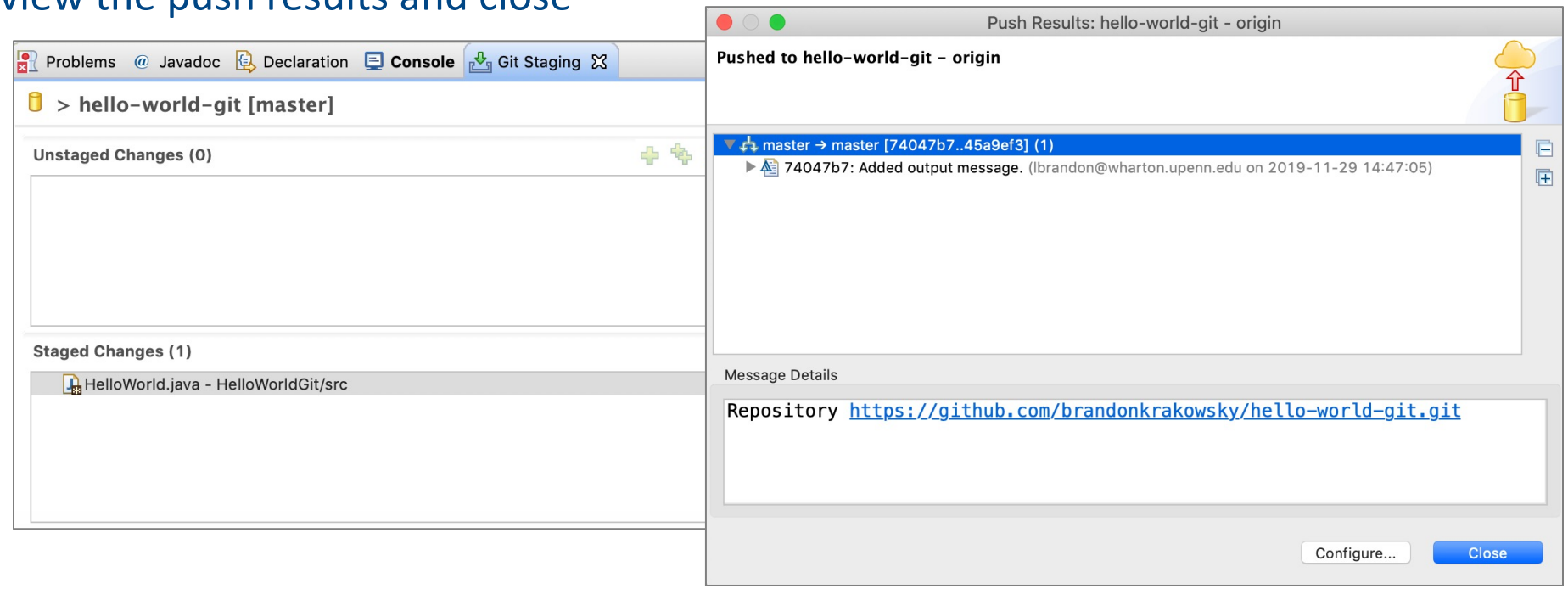
Making a Change & Committing It

- Drag the updated file into the “Staged Changes” area, type a meaningful commit message, and press “Commit and Push”
  - Again, this will upload your file and changes to the remote git repository



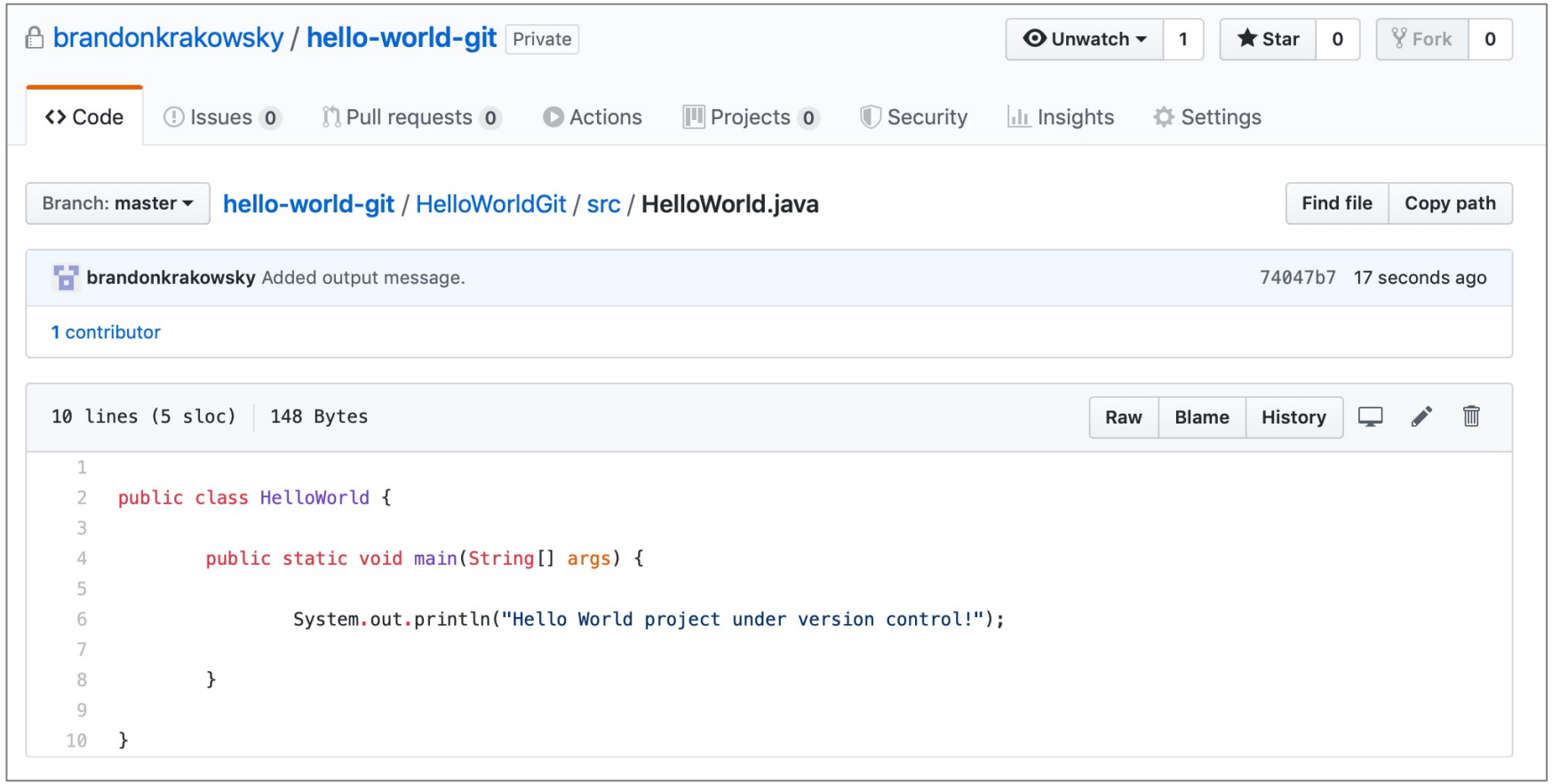
Making a Change & Committing It

- Drag the updated file into the “Staged Changes” area, type a meaningful commit message, and press “Commit and Push”
  - Again, this will upload your file and changes to the remote git repository
- Review the push results and close



Viewing Changes in GitHub

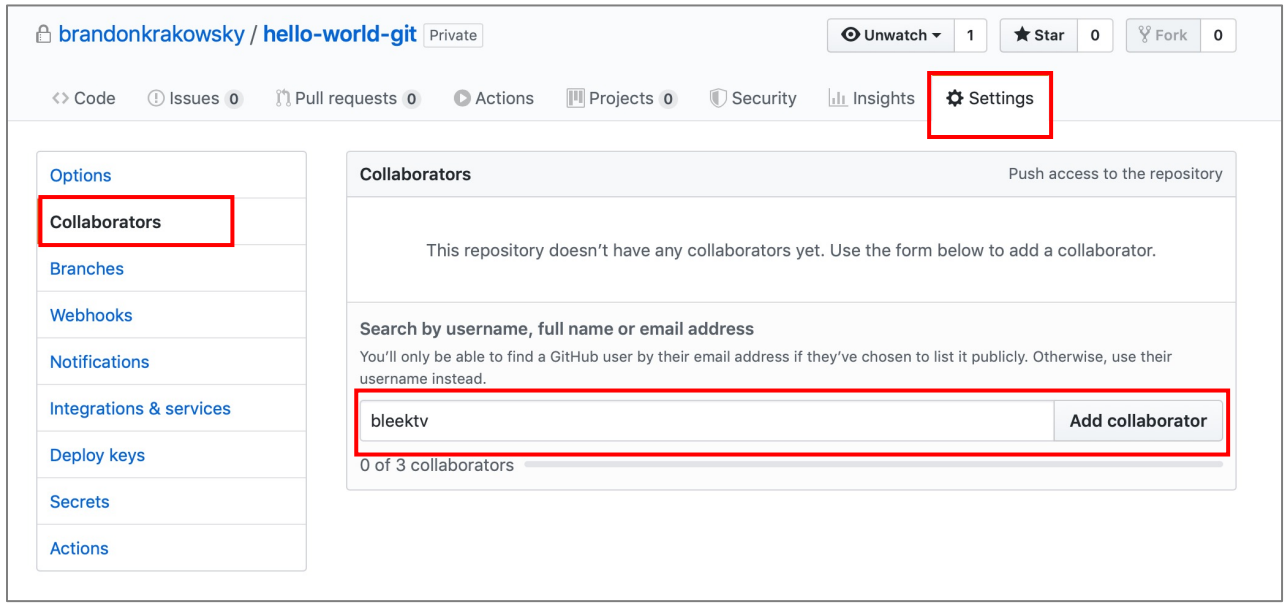
- Refresh the file in GitHub to see the changes



Setting Up & Configuring a Collaborator

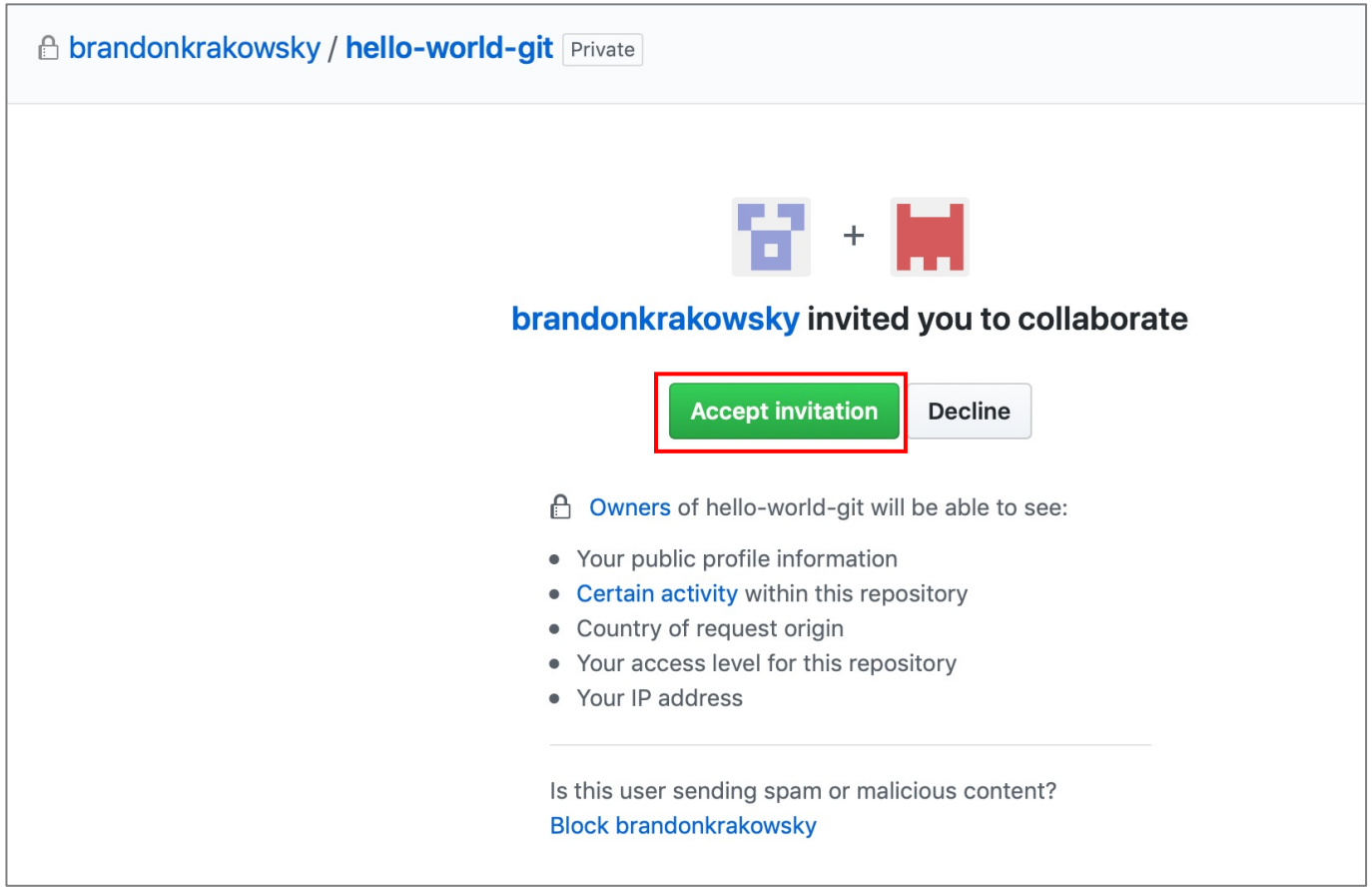
Adding a Collaborator to a Repository

- As a repository owner, you can add a collaborator to a project repository
  - This gives them push access to the repository
- In GitHub, go to your project's repository, and go to Settings → Collaborators, then add a collaborator
  - The GitHub user will receive an invitation via email



Accepting an Invitation to a Repository

- The invited collaborator should accept the invitation to gain push access

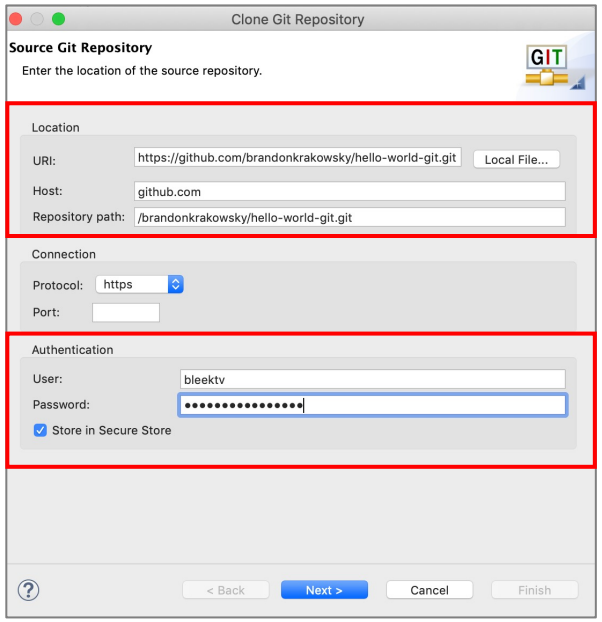


Configure Git in Eclipse (Collaborator)

- Assuming an Eclipse IDE is being used, the collaborator will need to:
  - Set up and configure Git in Eclipse
  - Open the “Git Repositories” and “Git Staging” views
  - Configure the git user.email and user.name

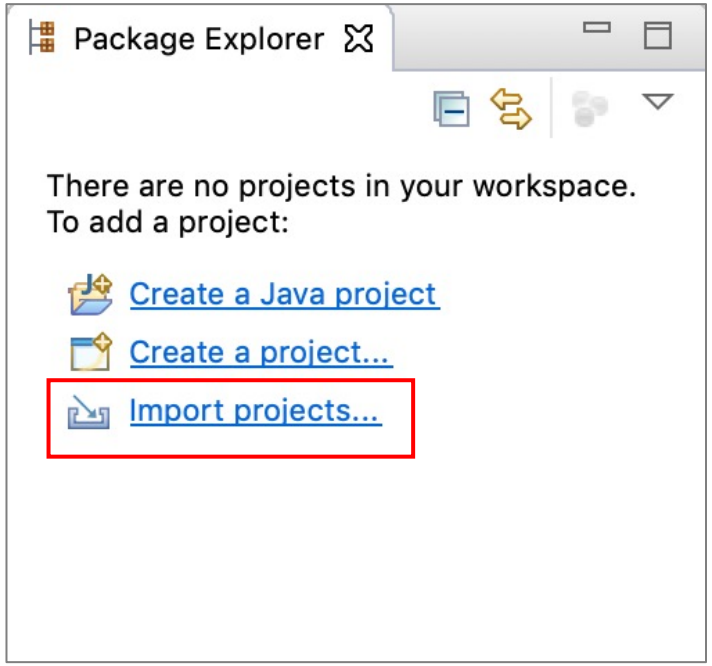
Cloning the Repository from GitHub (Collaborator)

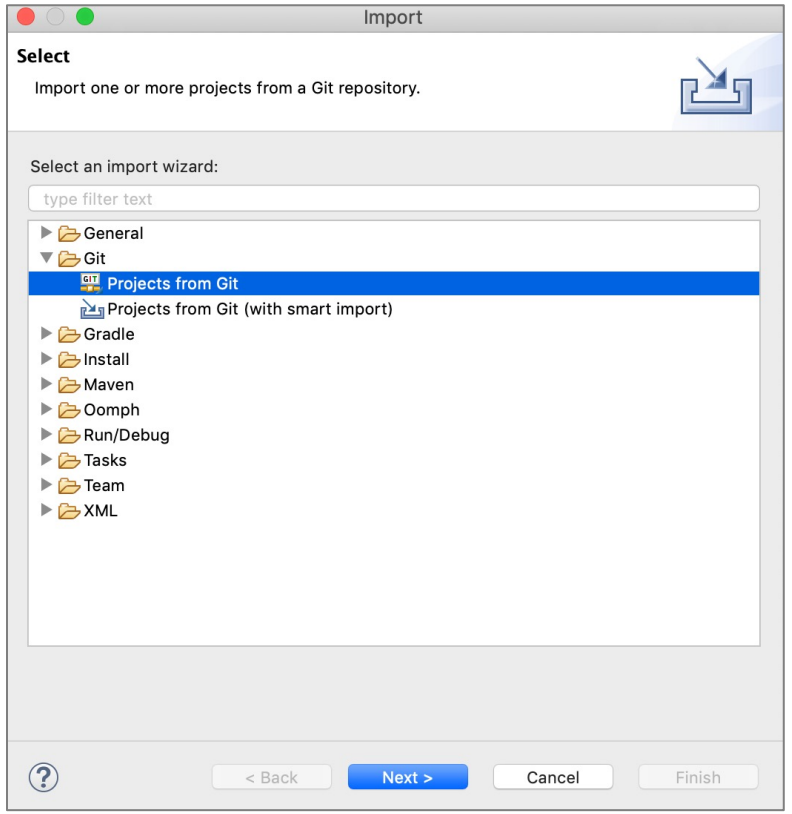
- Then clone the same remote repository from GitHub
- First, make sure you've verified your email and set up a Personal Access Token (PAT) as a collaborator
- Then use the correct GitHub account username and personal access token



Importing the Eclipse Project from Git (Collaborator)

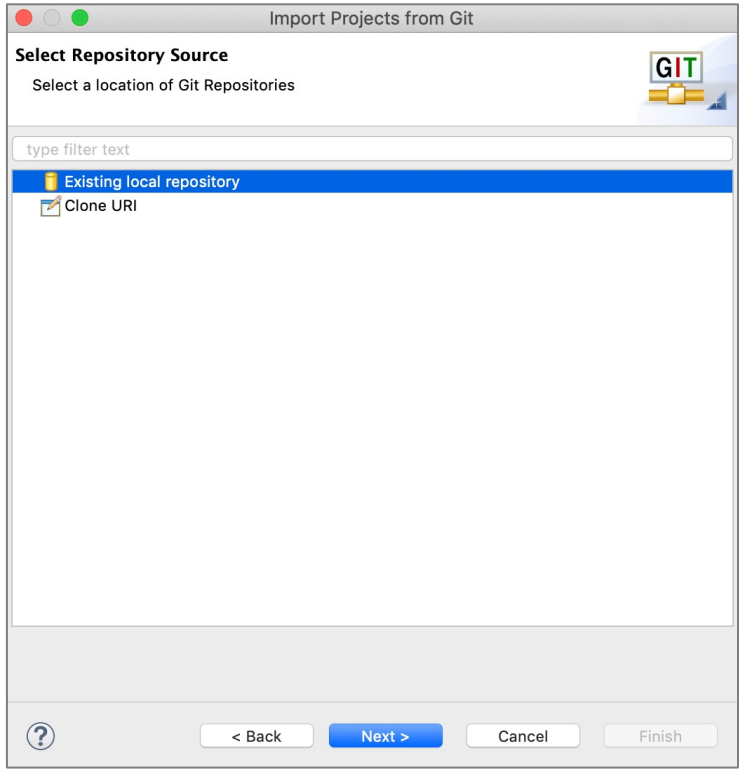
- The collaborator will then need to import the project into Eclipse
- Go to Import projects … → Git → Projects from Git

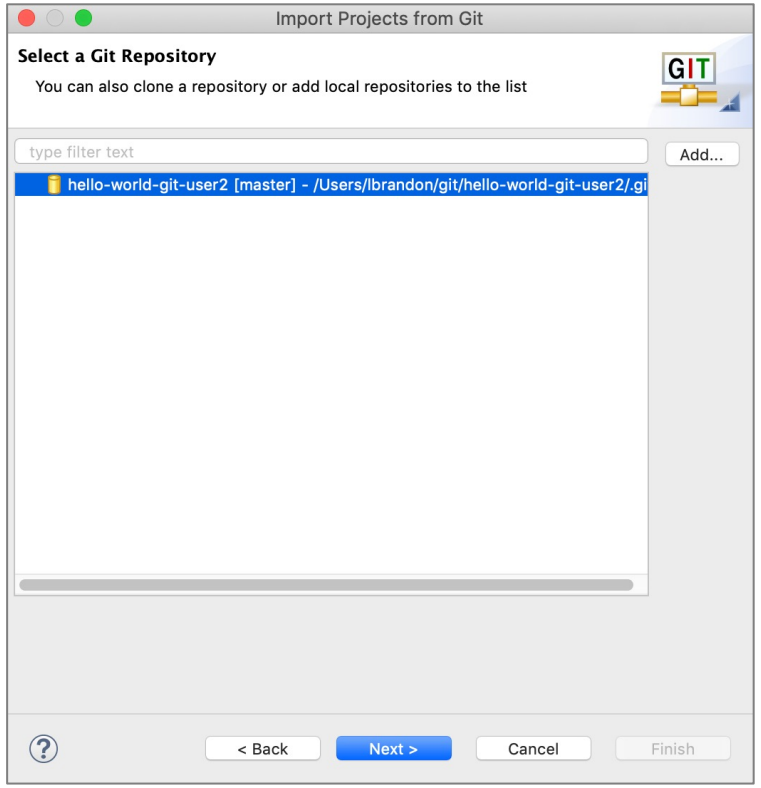




Importing the Eclipse Project from Git (Collaborator)

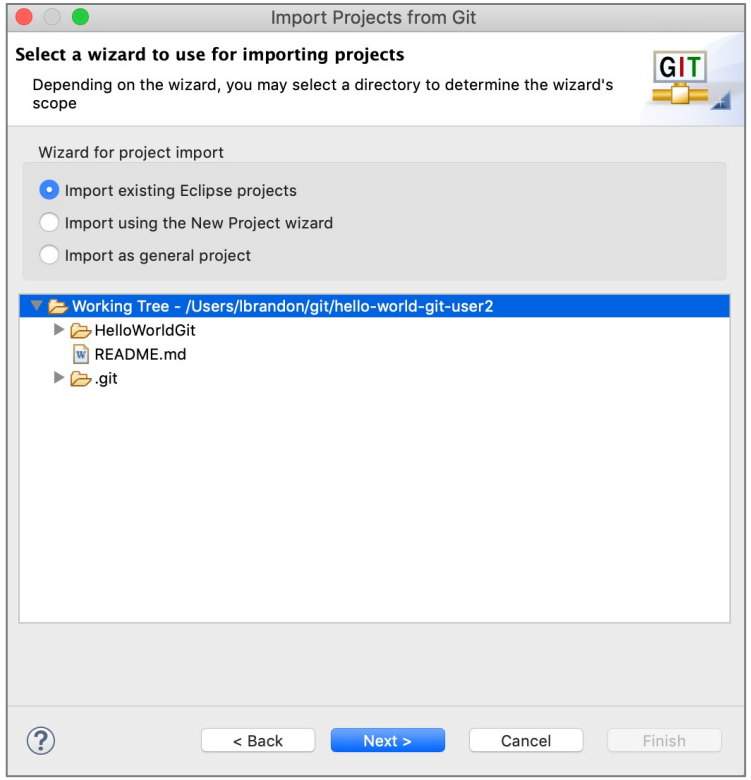
- Select “Existing local repository” and click “Next”
- Then select the correct local git repository and click “Next”

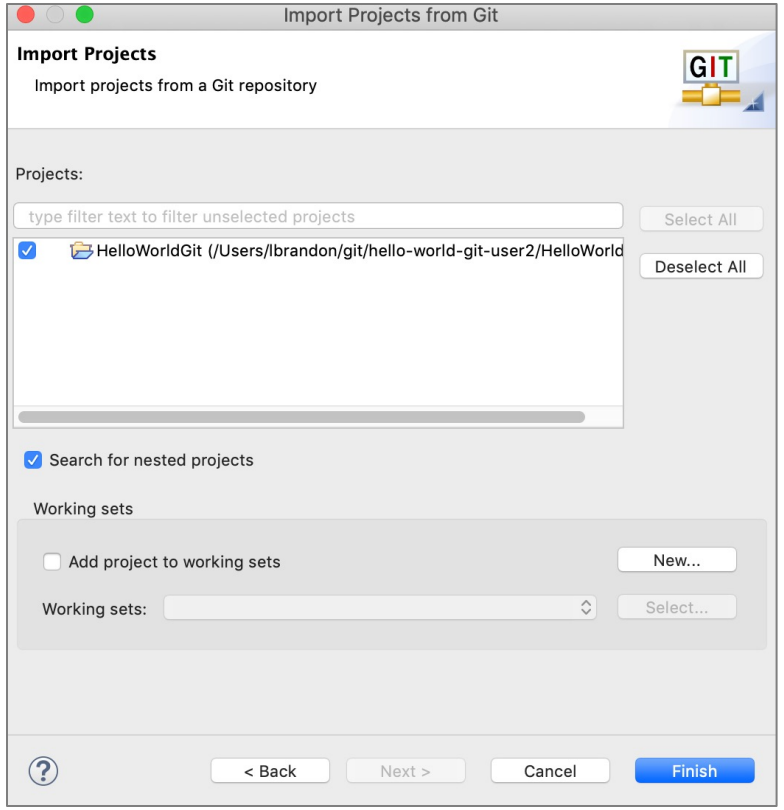




Importing the Eclipse Project from Git (Collaborator)

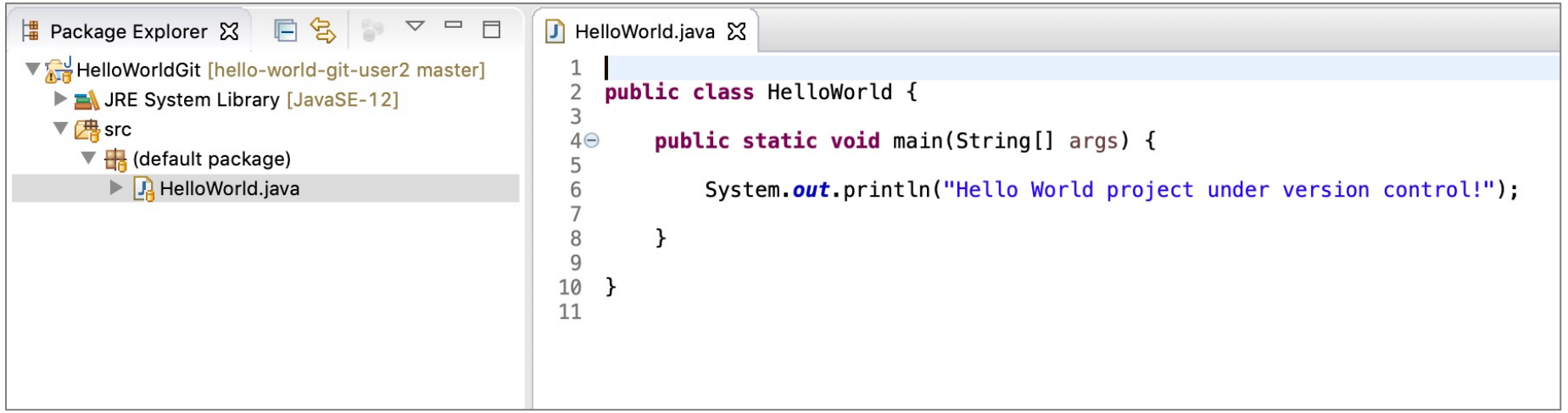
- Select “Import existing Eclipse projects” and click “Next”
- Then select the correct Eclipse project and click “Finish”





Importing the Eclipse Project from Git (Collaborator)

- You'll see the Eclipse project in the Package Explorer



Making a Change & Committing It (Collaborator)

- Make a change to a file in the project

HelloWorld.java

```
1
2  public class HelloWorld {
3
4      public static void main(String[] args) {
5
6          System.out.println("Welcome to the Hello World project under version control!");
7          System.out.println("Tell me your name: ");
8
9      }
10
11 }
12
```

Making a Change & Committing It (Collaborator)

- Add a file to the project

HelloWorld.java | MyScanner.java

```
1  import java.util.Scanner;
2
3  /**
4   * Creates a Scanner and gets user input.
5   * @author lbrandon
6   *
7   */
8  public class MyScanner {
9
10     /**
11      * Scanner for user input.
12      */
13     private Scanner scanner;
14
15     /**
16      * Gets user input.
17      * @return User input
18      */
19     public String getInput() {
20
21         this.scanner = new Scanner(System.in);
22         String input = this.scanner.nextLine();
23         this.scanner.close();
24
25         return input;
26     }
27 }
28
```

Making a Change & Committing It (Collaborator)

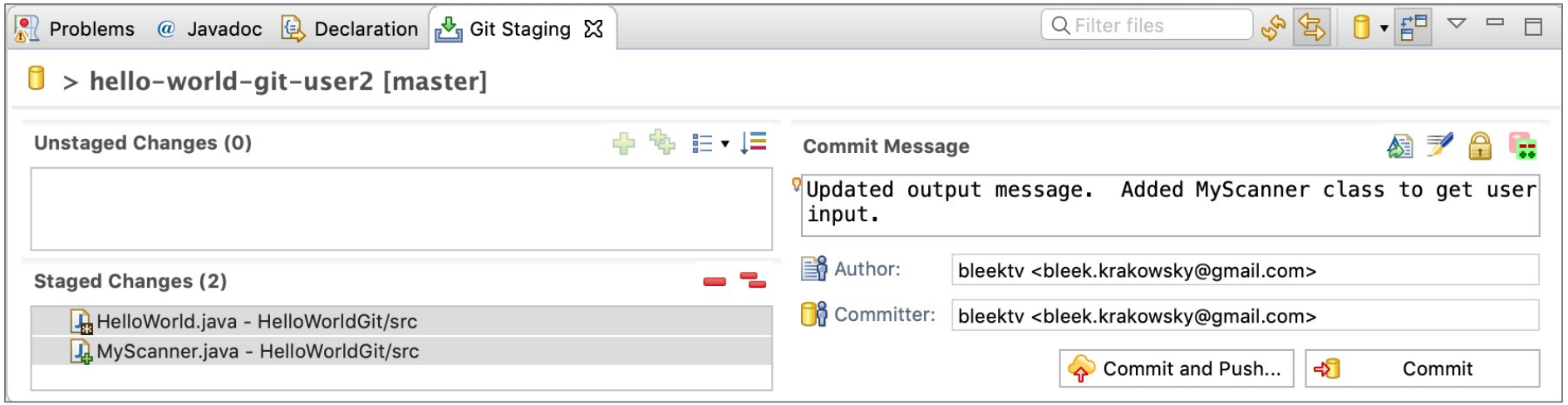
- Make another change to a file in the project

HelloWorld.java | MyScanner.java

```
1
2  public class HelloWorld {
3
4      public static void main(String[] args) {
5
6          System.out.println("Welcome to the Hello World project under version control!");
7          System.out.println("Tell me your name: ");
8
9          MyScanner myScanner = new MyScanner();
10         String input = myScanner.getInput();
11         System.out.println("Hello " + input + "!");
12
13     }
14
15 }
16
```

Making a Change & Committing It (Collaborator)

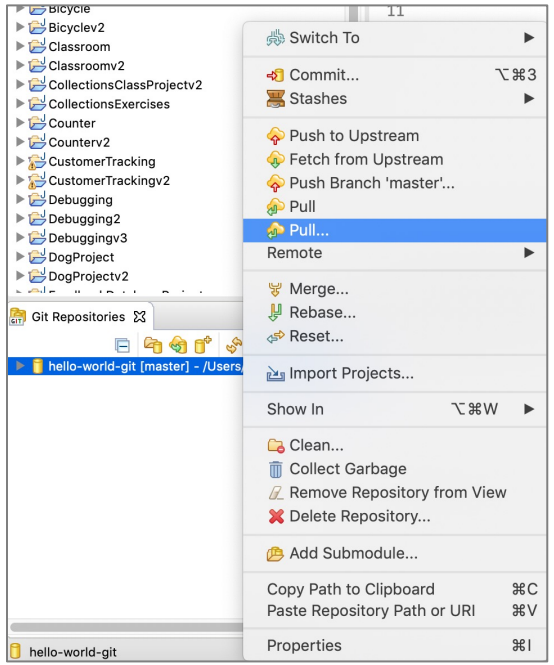
- Drag the updated file and new file into the “Staged Changes” area, type a meaningful commit message, and press “Commit and Push”

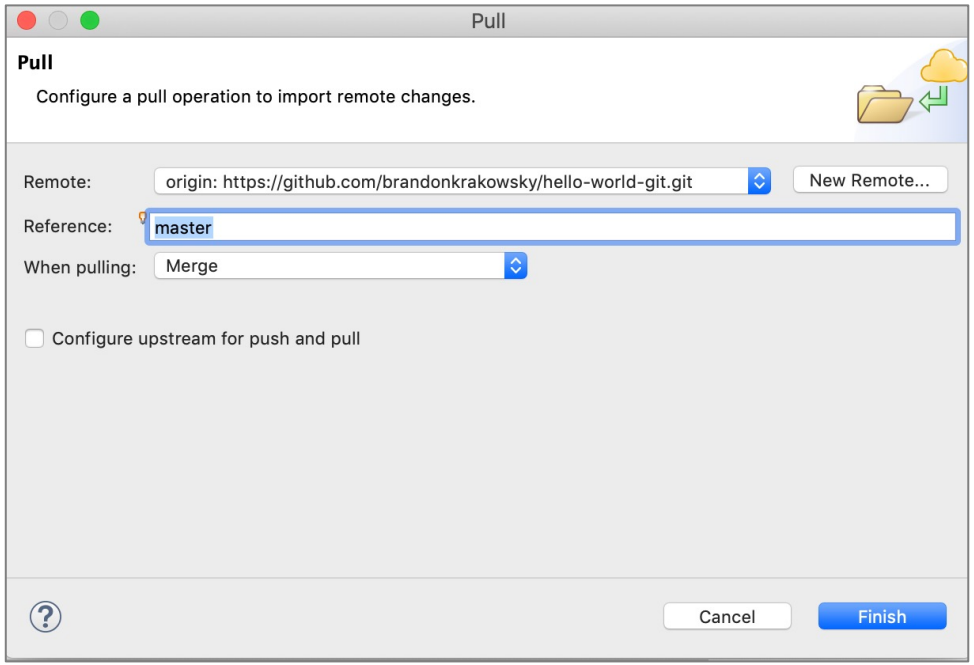


Pulling & Merging Changes

Pulling & Merging Changes (Repository Owner)

- Always make sure you are up-to-date before starting to work!
- To pull in the latest changes from the remote repository, right-click on the local git repository and select Pull …
- Pull from the master branch and changes will be merged automatically





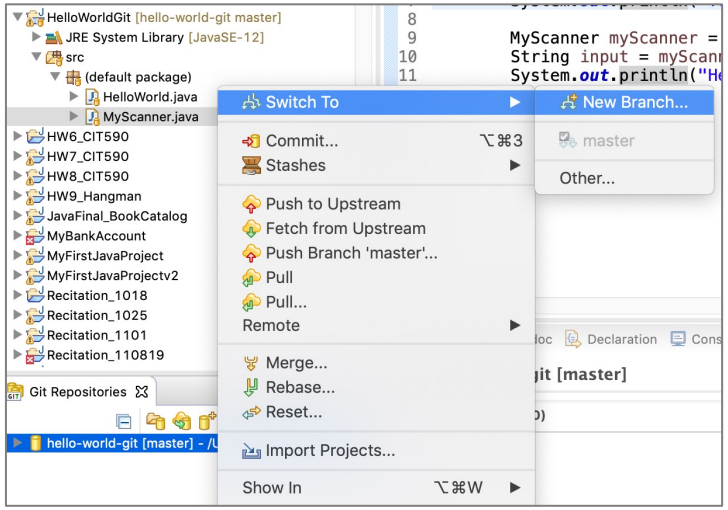
Branching

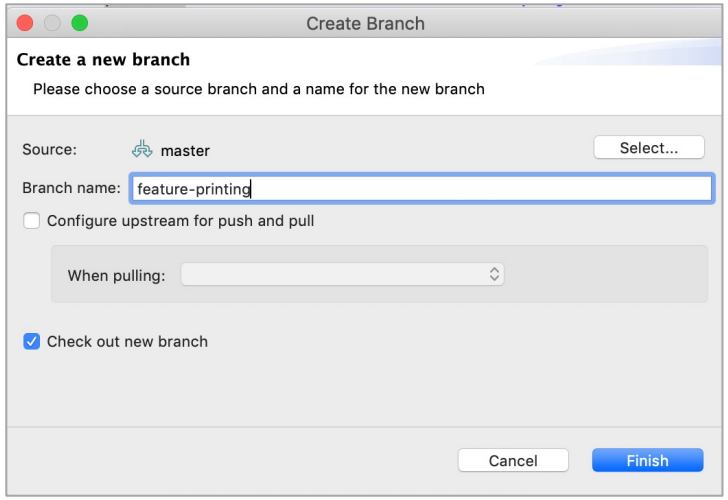
- Branching is the way to work on different versions of a repository at one time
- By default your repository has one branch named master which is considered to be the definitive branch
- We use branches to develop features, fix bugs, or safely experiment and make edits before committing them to master
- When you create a branch off the master branch, you're making a copy, or snapshot, of master as it was at that point in time

Ref: https://guides.github.com/activities/hello-world/

Creating a Branch (Repository Owner)

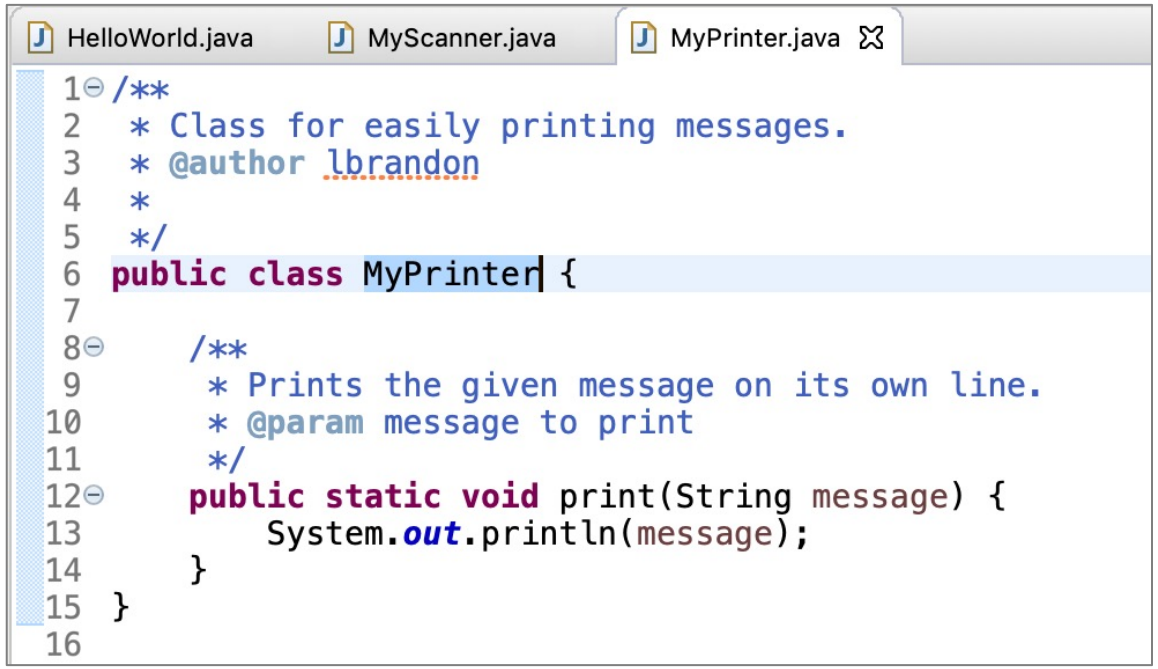
- Branching allows each developer to branch out from the original code base and isolate their work from others
  - It also helps Git to easily merge versions later on
- For example, you can create a branch to work on a particular feature …
- Right-click on the local git repository and go to Switch To → New Branch…
- Give your branch a name and click “Finish”





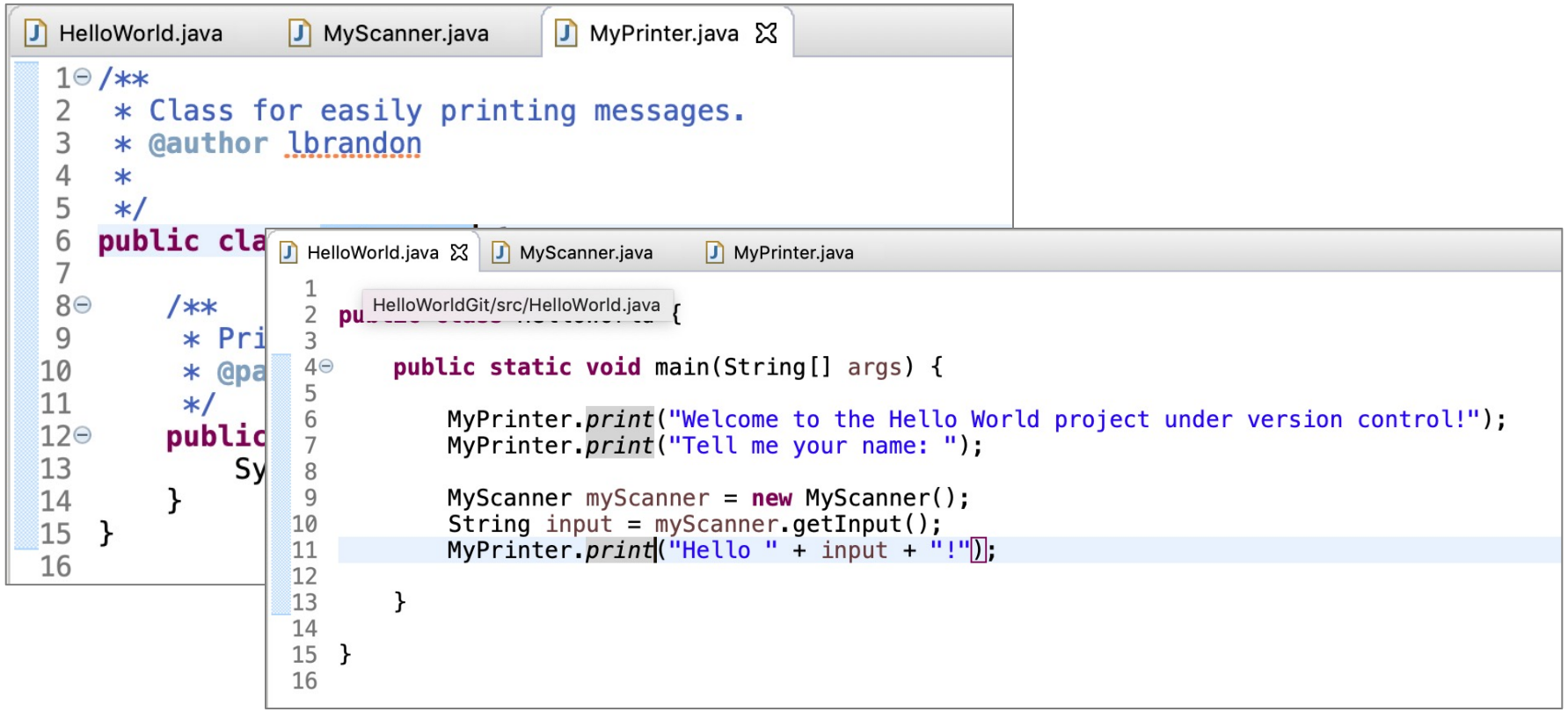
Work Inside of a Branch (Repository Owner)

- Add a new file



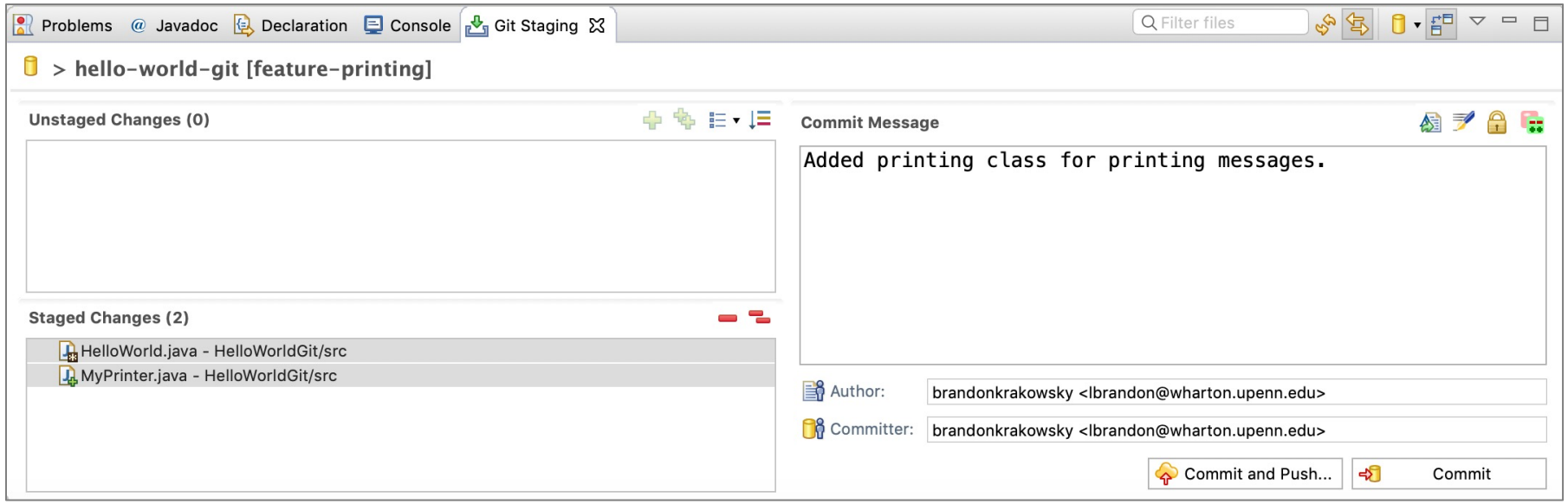
Work Inside of a Branch (Repository Owner)

- Add a new file
- Update code in an existing file



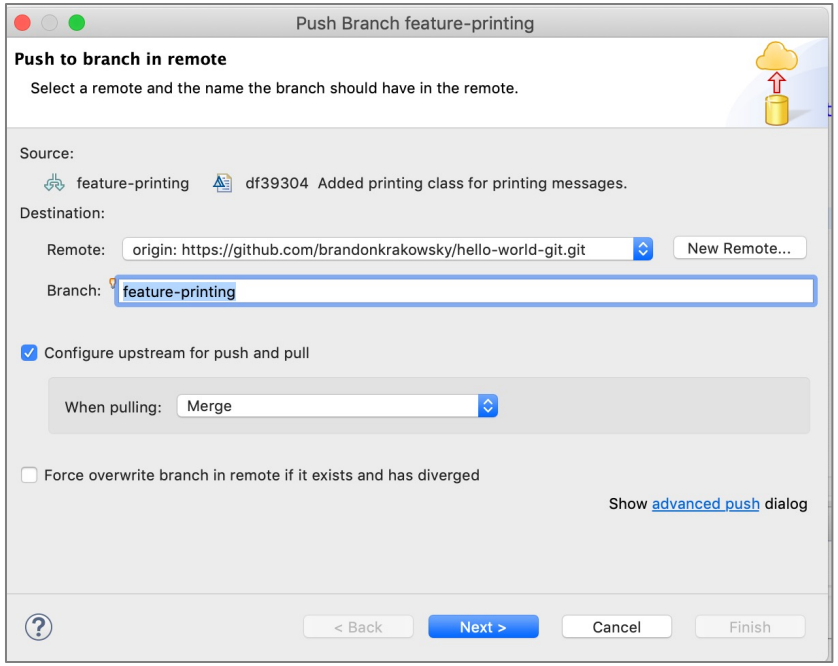
Commit & Push the Branch (Repository Owner)

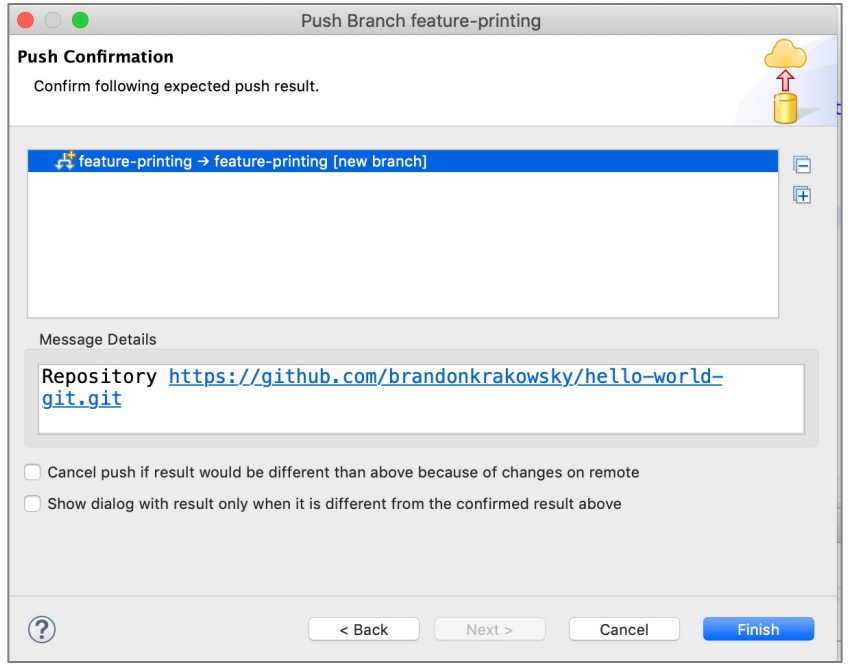
- Drag the updated file and new file into the “Staged Changes” area, type a meaningful commit message, and press “Commit and Push”
  - This pushes the changes in the current branch to the remote repository



Commit & Push the Branch (Repository Owner)

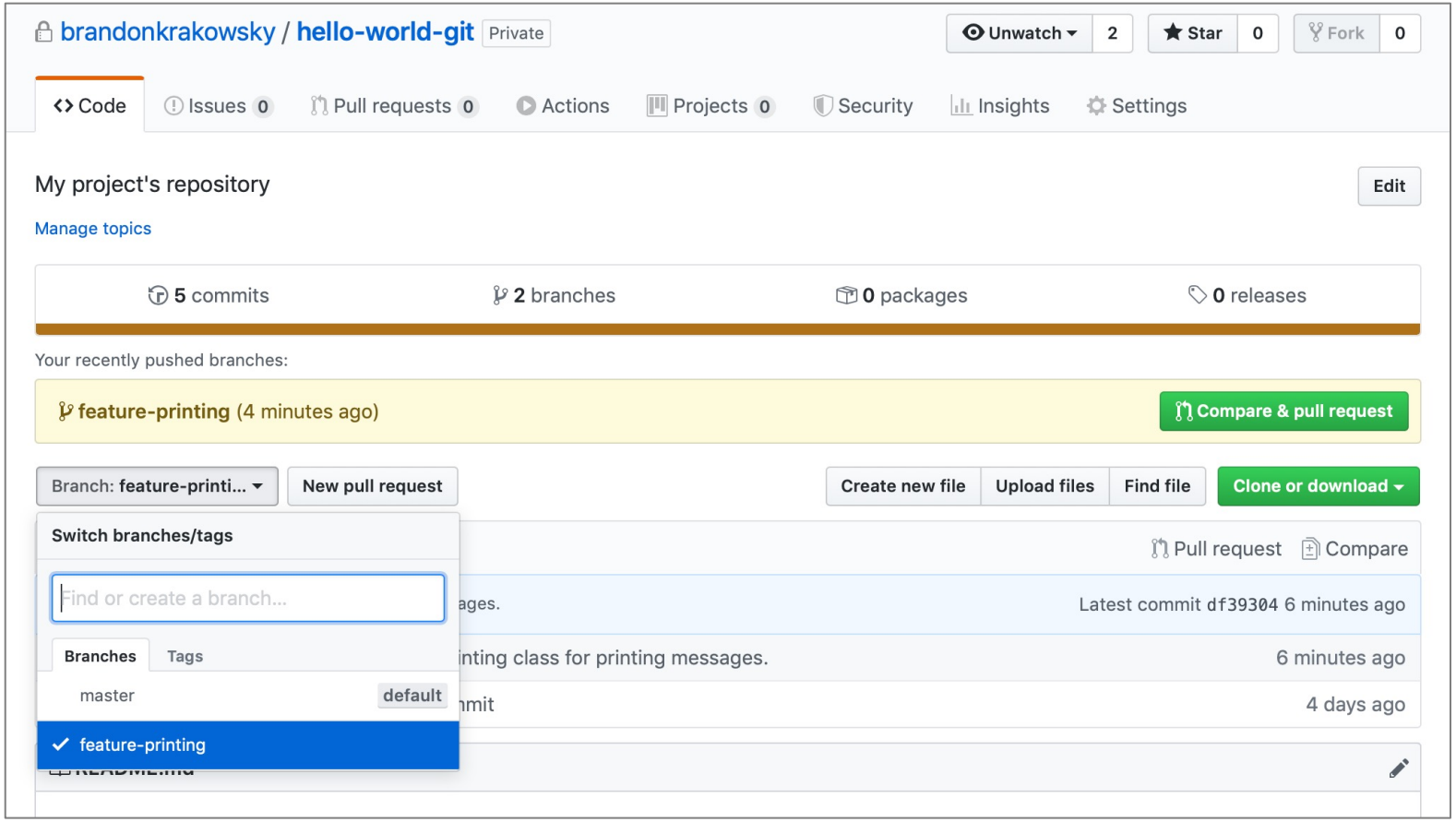
- Confirm the branch details and click “Next”
- Review the push confirmation and click “Finish”





Commit & Push the Branch (Repository Owner)

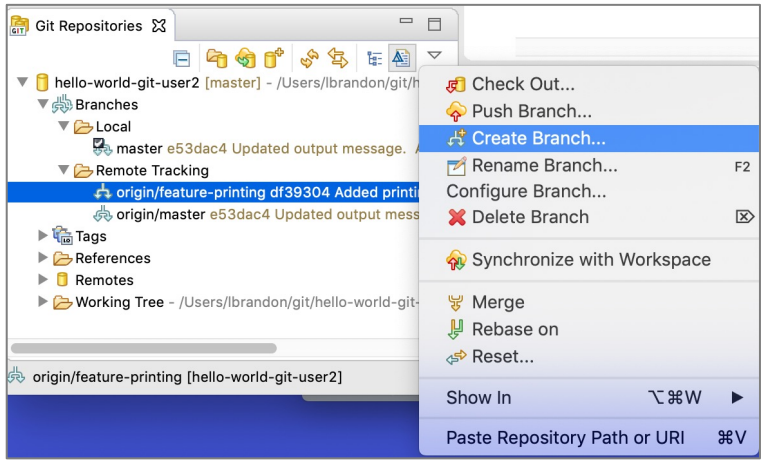
- On GitHub, you can switch to specific branches by selecting the branch from the Branch drop-down

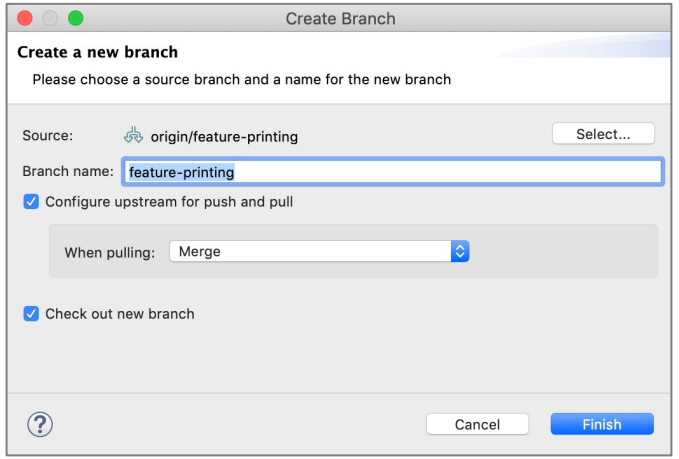


Pulling & Merging Branches

Pulling a Branch (Collaborator)

- To pull in the latest changes from the remote repository, right-click on the local git repository and select “Pull …”
  - Pull from the master branch and changes will be merged automatically
- To pull in the latest changes for another branch, select the branch name under “Remote Tracking”, right-click and select “Create Branch…”
  - This will create a local branch based on the remote one
- Confirm the details of the new branch, then click ”Finish”



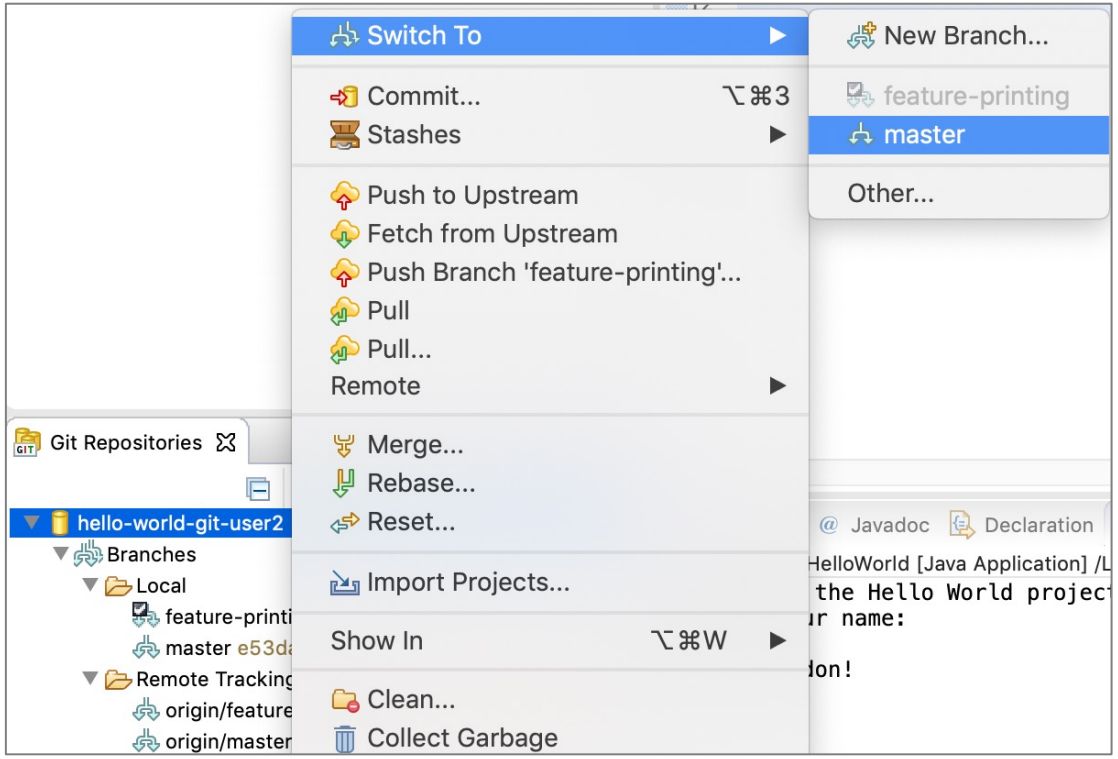


Pulling a Branch (Collaborator)

- This will switch you to the new branch and you'll see the updated files in your Eclipse project

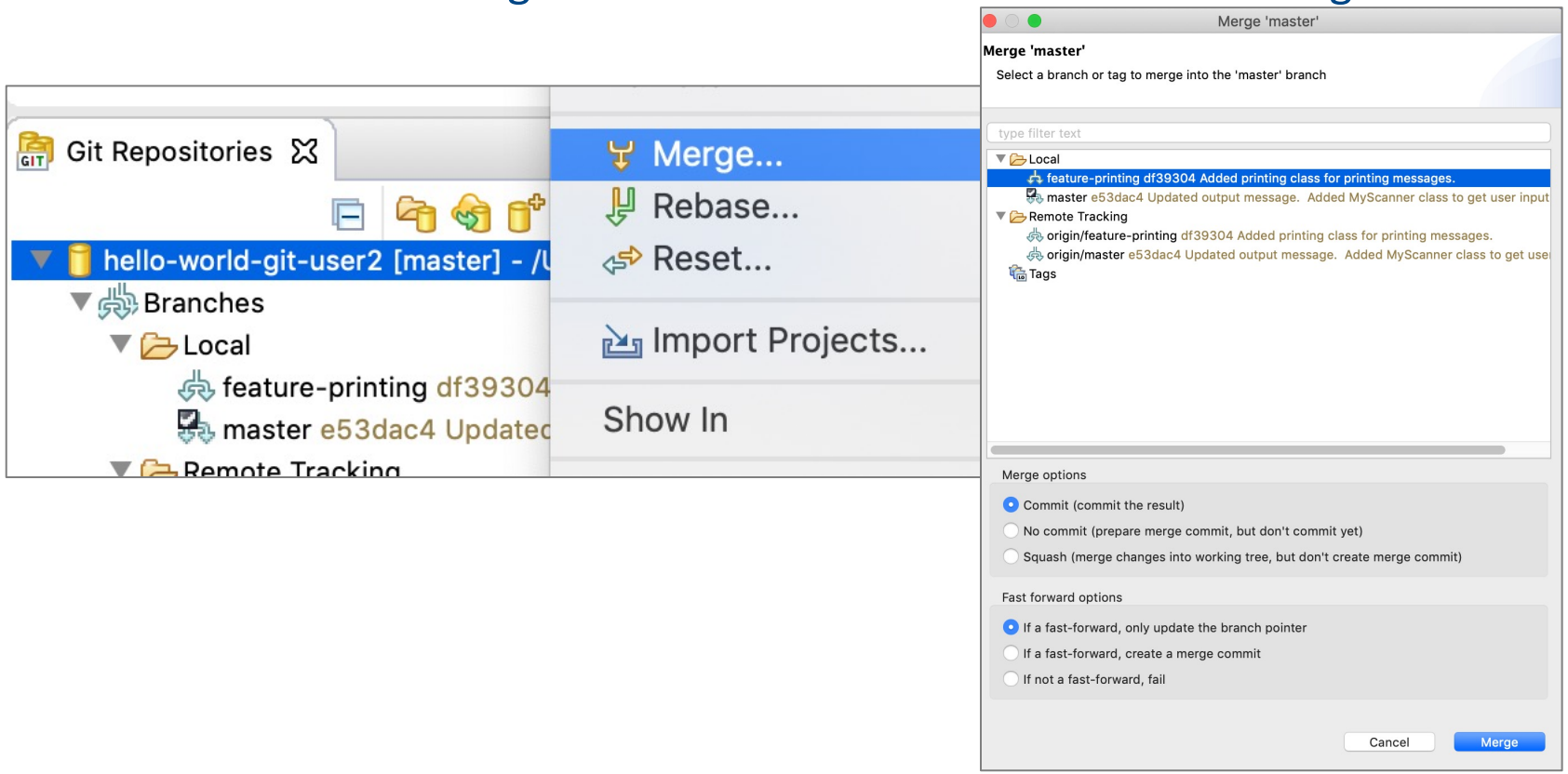
Switching to a Branch (Collaborator)

- To merge the new branch with the master branch, switch back to the master branch
- Right-click on the git repository and select “Switch To” → “master”



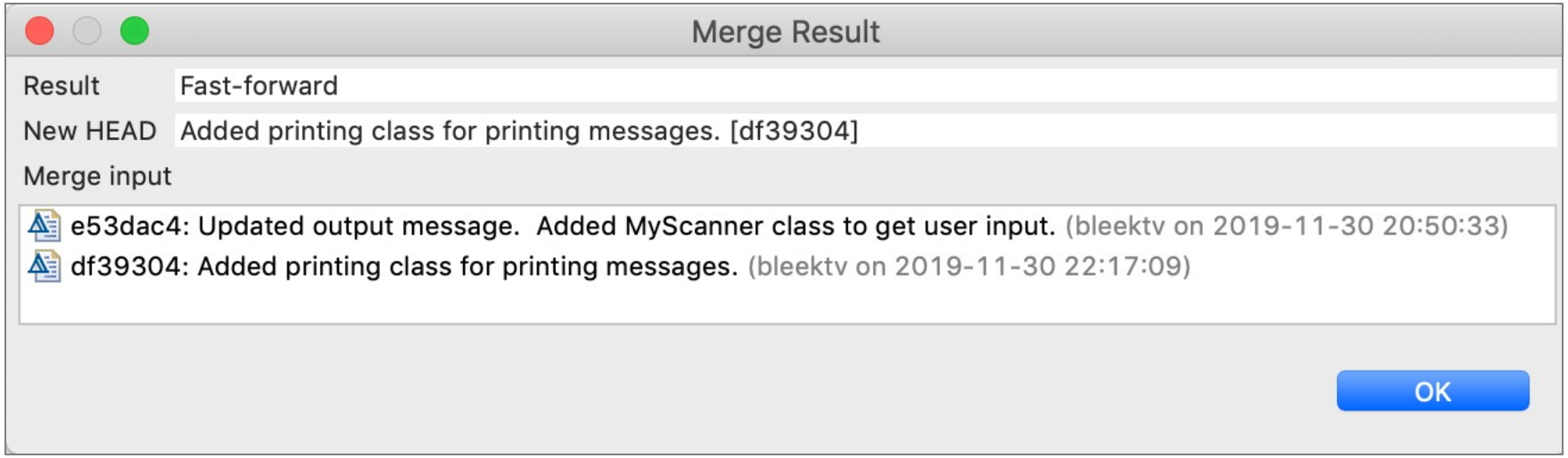
Merging a Branch (Collaborator)

- Right-click on the git repository and select “Merge…”
- Select the local branch to merge into the master branch and click “Merge”



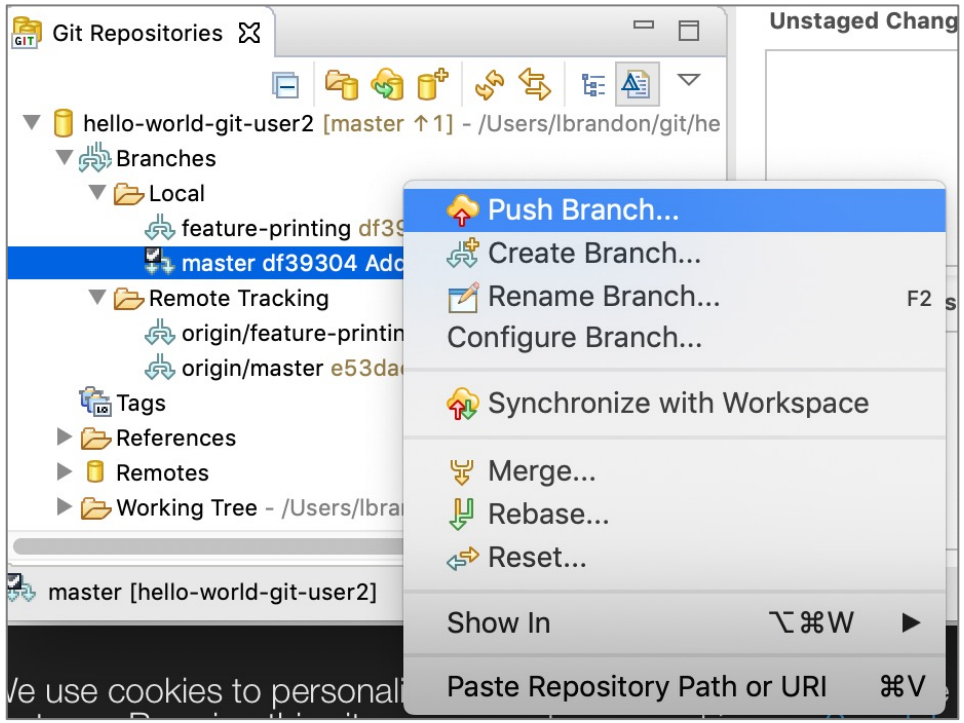
Merging a Branch (Collaborator)

- Confirm the merge result and click “OK”



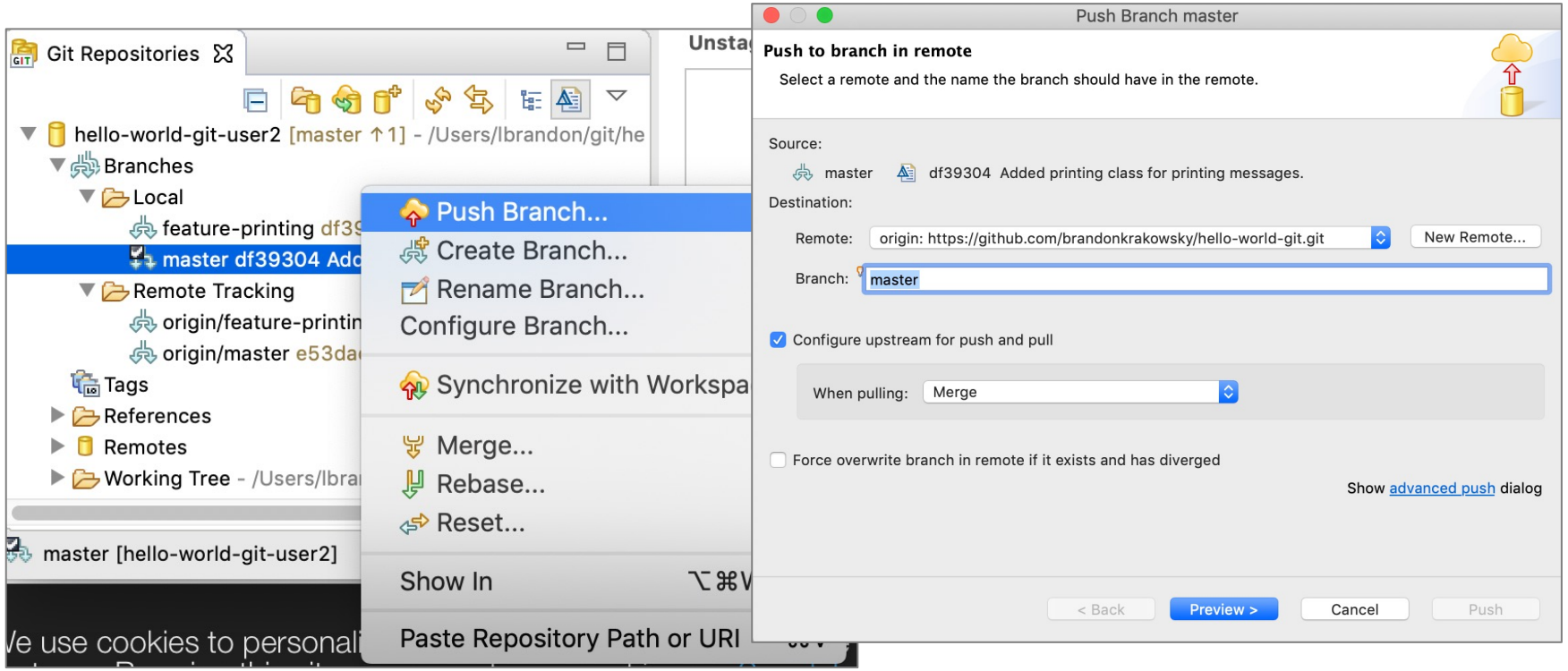
Pushing a Branch (Collaborator)

- Push the updated master branch to the remote repository
- Right-click on the local master branch under Local and select “Push Branch”



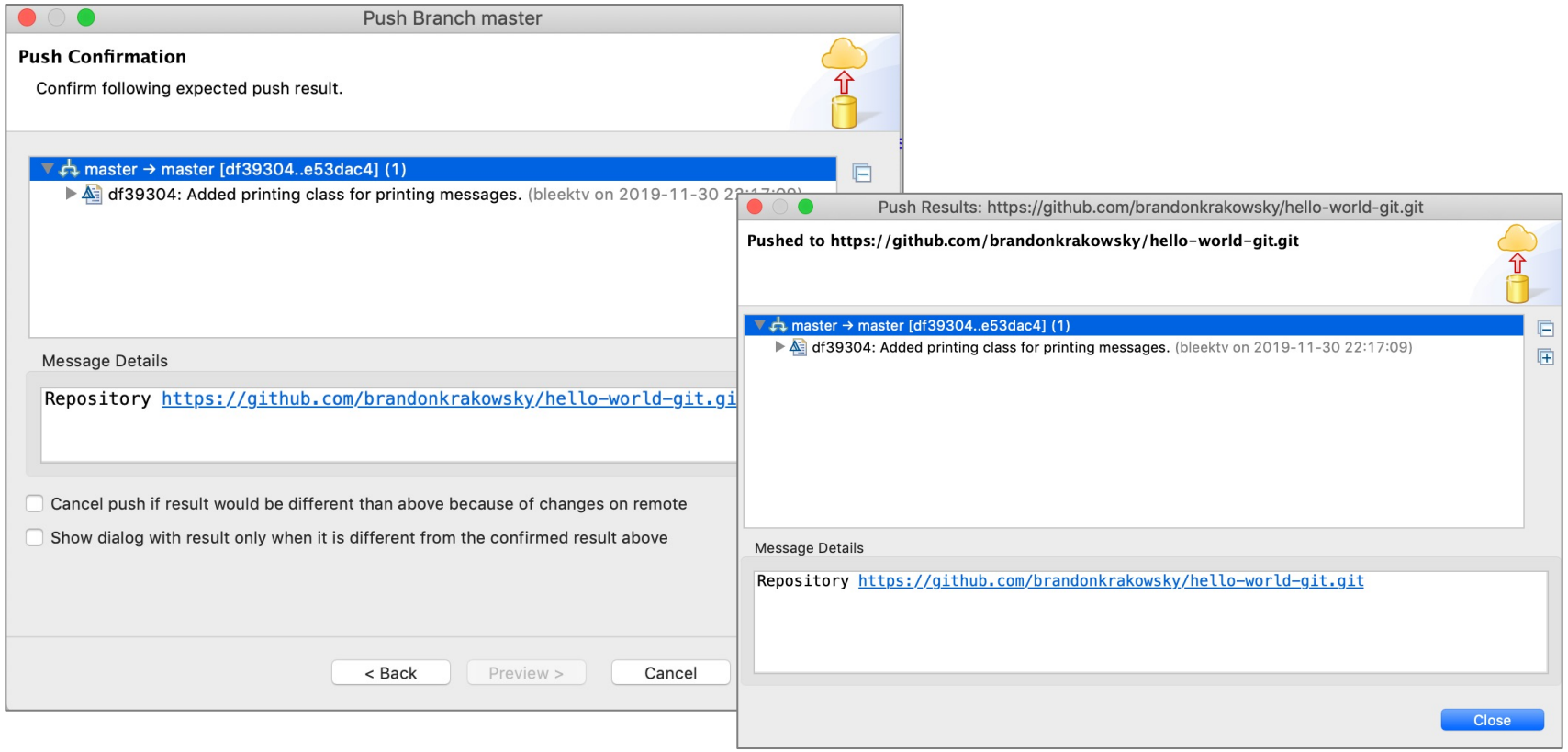
Pushing a Branch (Collaborator)

- Push the updated master branch to the remote repository
- Right-click on the local master branch under Local and select “Push Branch”
- Confirm the details of the master branch and click “Preview”



Pushing a Branch (Collaborator)

- Confirm the push confirmation and click “Push”
- Review the push results and click “Close”



Where to Go From Here

- Create a GitHub (or other web-based version control system) account
- Upload your coding projects to individual (private) repositories
- Start using Git for your day to day coding
- Learn more about Git!
  - Forking: Making a copy of a remote repository to another (GitHub) account
  - Pull requests: Notifying collaborators that you have completed a feature branch and it's ready to be merged
  - Resolving merge conflicts: Merging branches that have competing commits. You need to decide which changes to incorporate in the final merge.
  - Checking out commits: Going to a specific version of your project
  - Adding .gitconfig to each project: Having separate user configuration/credentials for each project in your local git

For Reference: Git for Python

- Using Git for Python
  - PyCharm has a Git Integration plugin
    - Make sure it's enabled in Settings/Preferences
    - More info: https://www.jetbrains.com/help/pycharm/using-git-integration.html
  - Of course, you can always use Git from the command line!
- If you want to learn more about the Git command line
  - https://git-scm.com/book/en/v2/Getting-Started-The-Command-Line