Git and Github Guide

Kaiyi Ye

Table of contents

1 Set up RStudio Project and Create a `.qmd` File

Establishing good practices for organizing and accessing project files is essential for effectively sharing your work with others.

1.1 Step 1: Create a New Project

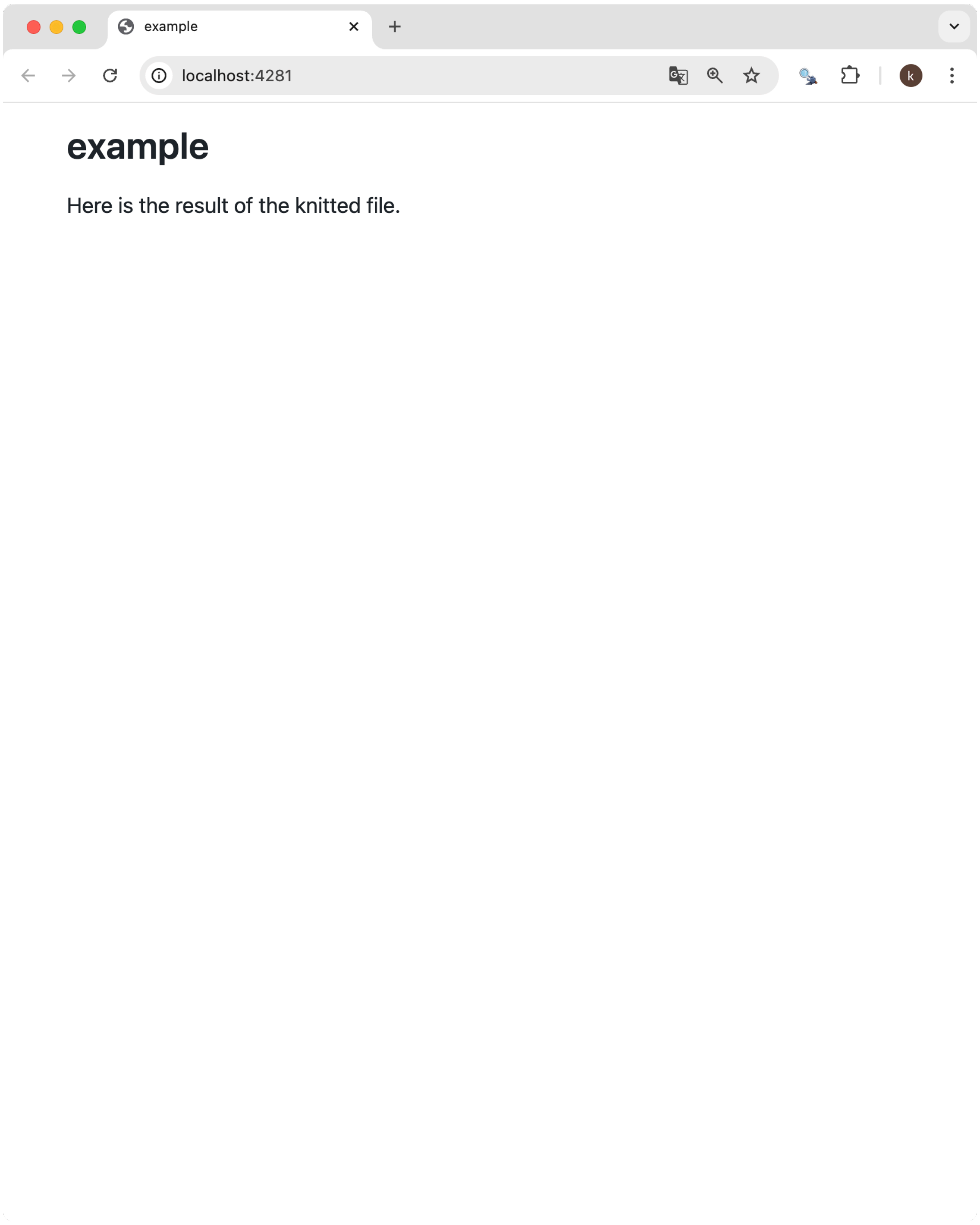
1. Open RStudio.
2. Go to the top menu and select `File` > `New Project....`
3. In the "New Project" wizard, select `New Directory`.
4. Choose `New Project`.
5. Enter a name for your project and select a location on your computer where you want to save it.
6. Click `Create Project`.

1.2 Step 2: Create a `.qmd` File

1. Go to the top menu and select `File` > `New file` > `Quarto Document`.
2. Save the file as `example.qmd`.
3. Modify the YAML to set the document to render as HTML:

```yaml
title: "example"
author: write your name
format: html
```

4. Save the file, then render to preview the document output.



example

Here is the result of the knitted file.

2 Create a Git Repository from Existing work

If you have an existing project that you want to version control with Git, follow these steps:

2.1 Step 1: Initialize the Git Repository

1. Open your terminal or command prompt
2. Navigate to the root directory of your existing project folder using the `cd` command. For example: `cd path/to/your/project`
3. Run `git init` to initialize this directory as a Git repository. This command creates a new subdirectory named `.git` that contains all of your necessary repository files.

2.2 Step 2: Add Your Files to the Repository

1. Add all of your project files to the staging area: `git add .`
2. Commit the files to the repository with a descriptive message: `git commit -m "Initial commit"`

2.3 Step 3: Set Up the Remote Repository

1. Go to GitHub and create a new repository. Do not initialize it with a README, .gitignore, or license.
2. Copy the URL (SSH) of the new GitHub repository.
3. In your terminal, add the remote repository URL to your local repository:

```
git remote add origin git@github.com:your-username/your-repository.git
```

4. Push your local repository to GitHub: `git push -u origin main`

Now you should be able to see your repository on GitHub.

Tip

We use the `-u` flag to tell Git to remember the connection between our local main branch and the remote origin/main branch.

3 Create a New Branch and Make Changes

3.1 Step 1: Create a New Branch

1. In the terminal, make sure you are inside your project folder: `cd my-new-project`
2. Create and switch to a new branch called `testbranch`:

```
git branch tsetbranch
git switch testbranch
```

Or you can also do this in one command:

```
git switch -c testbranch
```

Tip

The `-c` flag is short for `--create`. It tells git to create a new branch and immediately switch to it.

3.2 Step 2: Make Changes to a File

1. Open the `example.qmd` file in RStudio.
2. Make sure you are working on branch `testbranch`: `git branch`
3. Add a new line to the bottom: `This is a change I made on the testbranch.`
4. Save the file.

3.3 Step 3: Stage and Commit the Changes

1. Check which files were changed: `git status`
2. Stage the file: `git add example.qmd`
3. Commit the change: `git commit -m "Added a line to example on testbranch"`

Your changes made to the file are now saved locally in your local branch.

4 Amend the Last Commit

4.1 Step 1: Create a Folder inside the Project

1. Inside the RStudio project, create a folder called `data`.
2. Place the data into this folder.

4.2 Step 2: Amend the Last Commit to Include the Folder

1. Stage and Commit the Changes:

```
git add .
git commit --amend
```

2. Edit the commit message at the top of the text editor. For example: "Added a line to example on testbranch and created a folder".
3. Save the commit and close the editor.
4. Push this amended commit to the remote: `git push -u origin testbranch`

You should now see the `testbranch` and the commit messages appear on GitHub.

Tip

We need the `-u` flag on the `push` command to set the remote branch as the default tracking branch. After running this, Git will remember that our local local is connected to the remote (origin). From now on, we can simply run `git push` without needing to specify the remote or branch name again.

5 Create a Conflict

1. Switch back to branch `main`: `git switch main`
2. Double check which branch you are currently working on: `git branch`
3. Open the `example.qmd` file and edit the exact same line in `main` that you previously changed in `testbranch` to trigger a conflict. For example: "This line was edited on main."
4. Save the file, then stage and commit the change:

```
git add .
git commit -m"Added the same line on main to make a conflict"
git push origin main
```

6 Fix the Merge Conflict

6.1 Step 1: Merge the `testbranch` into `main`

Try to merge the changes in `testbranch` onto `main`: `git merge testbranch` You'll see a merge conflict message that Git can't automatically merge the file.

```
<<<<<<< HEAD
This line was edited on main.
=======
This is a change I made on the testbranch.
>>>>>>> testbranch
```

This means: - Everything between `<<<<<<< HEAD` and `=======` is from `main` - Everything after `=======` is from `testbranch`