Department
for Education

DfE Statistics Development Team Workshops

Using git and GitHub (building a R-Shiny dashboard)

Contents

Introduction

This document provides a workthrough guide for statistics publication teams on how to work collaboratively using git, using the creation of a data dashboard as a relevant context.

Background Information

GitHub versus Dev Ops

GitHub and Dev Ops effectively provide the same service in terms of creating software via a git repository: they both act as the host for the remote repository, whilst offering important tools to manage bugs and issues, tasks, merging branches, deploying applications and so on.

Dev Ops is part of the Microsoft Azure platform and uses pricate DfE servers. This can allow you to connect or deploy your repository into wider Azure services. This includes SQL databases that you might already be storing data on as well as the DfE's implementation of rsconnect on DfE internal servers, which allows deployment of shiny apps for internal DfE use.

GitHub is hosted on external servers and therefore is more appropriate for making your code or application available for public access and use. For example, from a GitHub repository, you can deploy an R Shiny dashboard to shinyapps.io where members of the public may view and interact with your published data.

Pre-workshop requirements

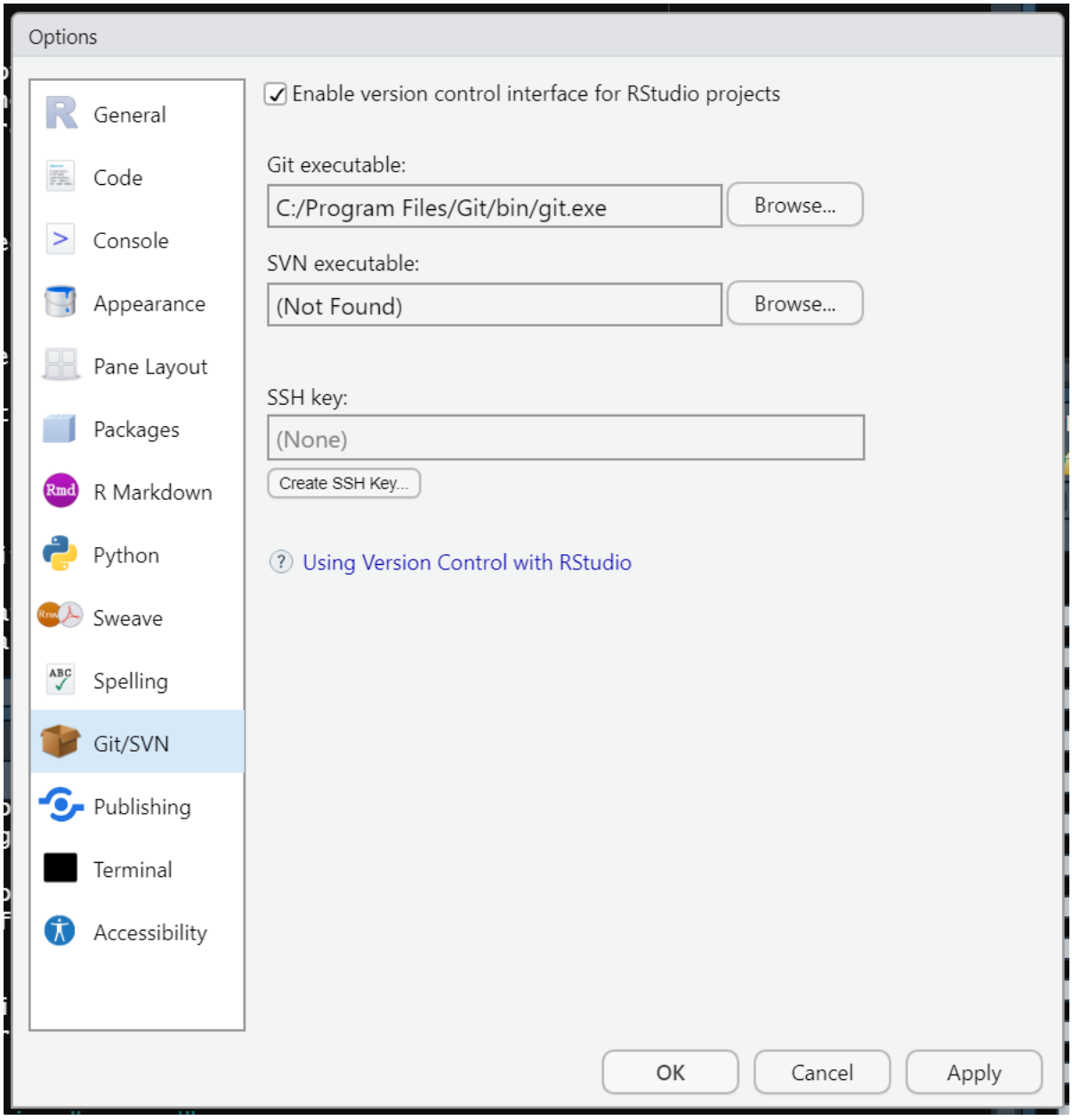
First of all, make sure to bring your laptop. This is going to be interactive and require you to do some coding.

Preferably before coming along, you'll need to go through the following list of things you'll need to make sure are set up on your DfE laptop:

- Create a GitHub account: https://github.com/join;
- Install git on your laptop: https://git-scm.com/downloads;
- Install R-Studio on your machine: Download **R for Windows (x64)** and **RStudio** from the Software Centre on your DfE laptop.

You'll also need to make sure that git is set up in the git/SVN pane of global options in R-Studio (found in the Tools drop down menu). Make sure the path to your git executable is entered in the git path box and git should automatically be integrated with R-Studio.

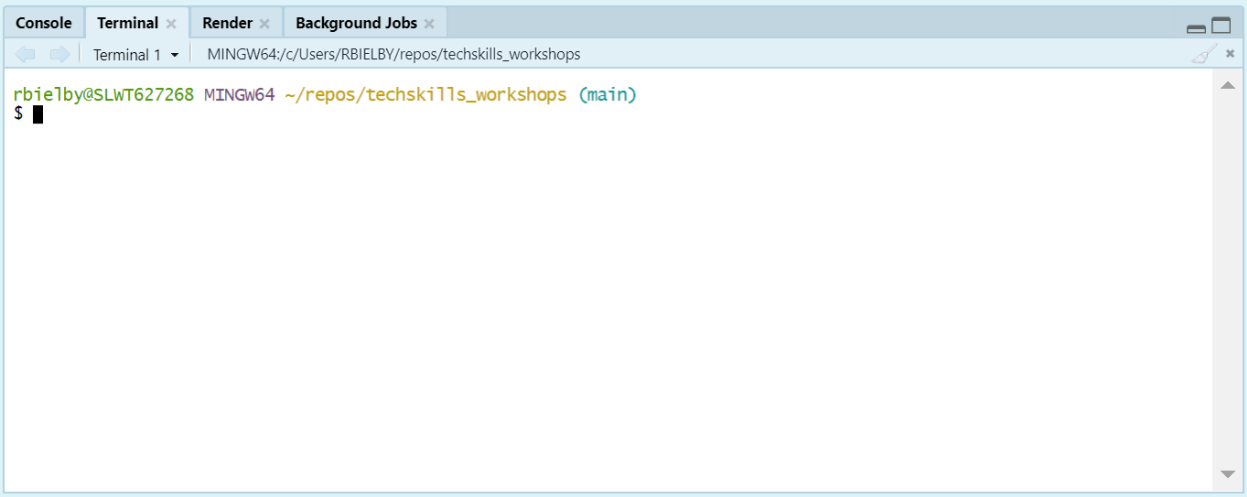
Figure 1: Enter the path to your git executable in the git path option box



Once you open a repository, you'll get an extra panel, named 'git', in the top right pane of R-Studio and you'll also be able to use git in the 'Terminal' tab at the bottom left (in the same place as the R console). A useful thing here if you want to use git commands in the terminal is to switch the terminal from the default Windows Command Prompt to `git BASH`.

You can do this in the Terminal tab of R-Studio's global options - just select `git BASH` from the 'New terminal opens with' pull down menu. Click apply and then select the Terminal tab (next to the Console tab), click 'Terminal 1' and then select 'New terminal' from the drop down menu. You should see something similar to the figure below.

Figure 2: The `git BASH` terminal in R-Studio

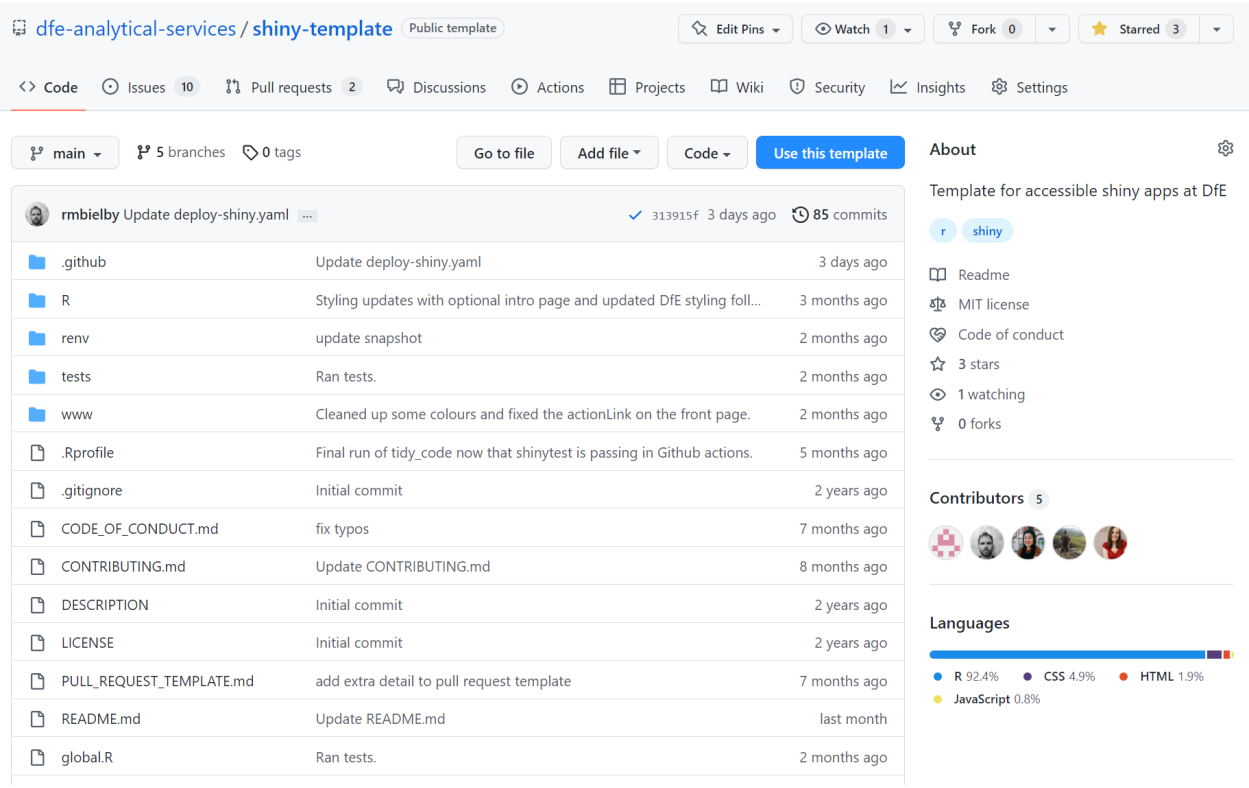


Setting up the repository

Creating a new repository on GitHub

At this point, we're ready to create a new repository. The conext of this exercise is to create a dashboard, so let's get a head start on that by using the DfE R-Shiny template.

Figure 3: Click the use this template button to create the new repository



You can access the template here:

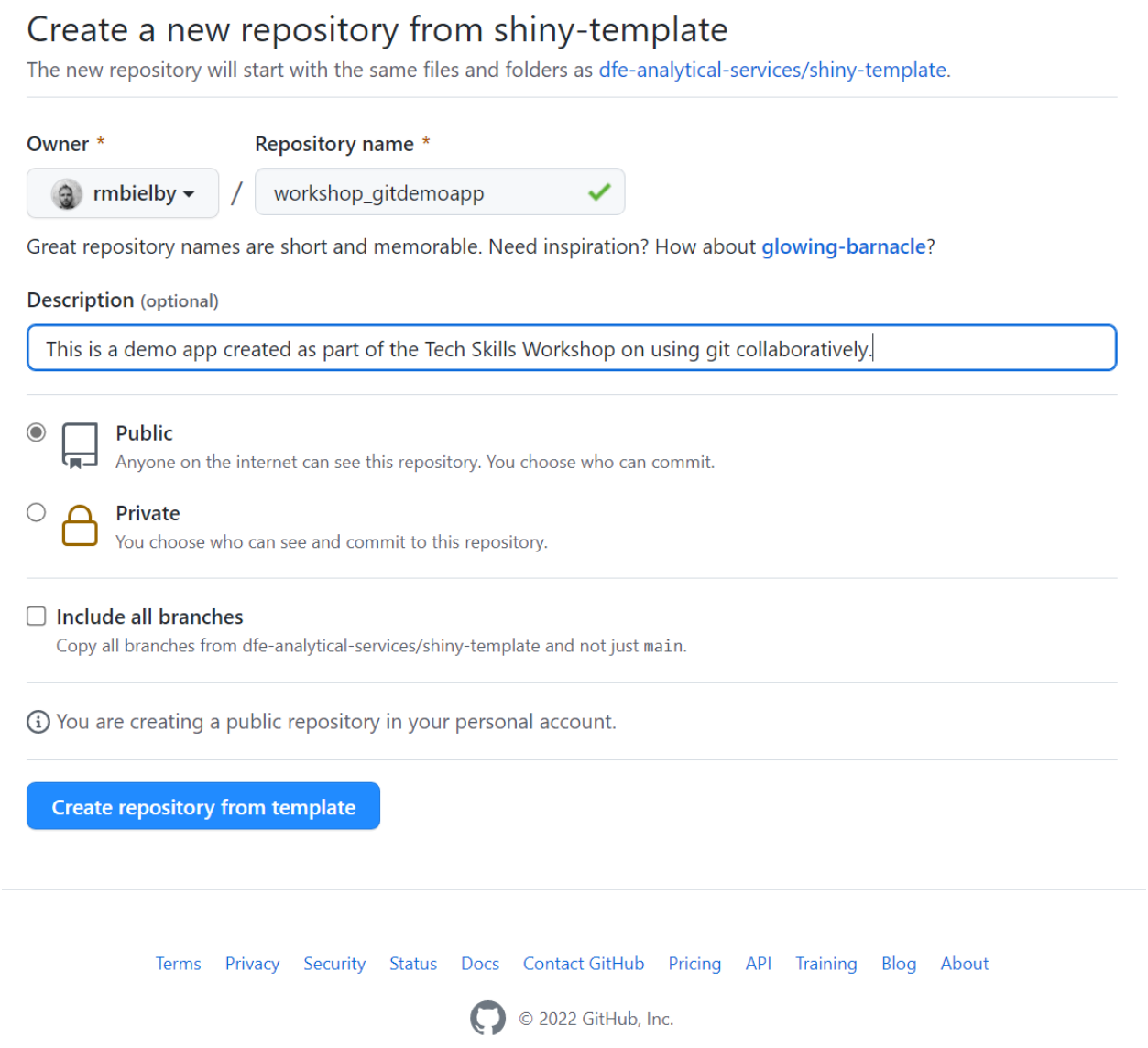
https://github.com/dfe-analytical-services/shiny-template

On that page, you'll see the main repository page. This contains a menu bar to navigate the range of GitHub features (e.g. Code, Issues, Pull Requests, Discussions, Actions and others); shortcuts to access different branches within the repository; some top-level summary information on the repository; a listing of the files and folders in the repository's root directory; and a markdown render of the repository's Readme file if one exists.

In the case of our template, you'll also see a button saying use this repo as a template. At this point, one (and only one!) of your group should click that button, which will take you to the create repository page. Here you have the option to create a copy of the template in

your own GitHub area as shown below. Give the new repository a name and a description and then click “Create repository from template”.

Figure 4: Put in a name for the new repo and a quick description and then click on Create repository from template



One of you at least should now have their own repository produced from this starting template project. We need everyone in your team to be able to work on this project however, so now you’ll need to give access to your other team members. To do so, navigate to settings (at the far right of the menu bar on your repository page) and =

Now you’ll need to add your other team members as collaborators in order to allow them to contribute to the code development in this repository.

Cloning the repository to your local machine

Cloning the repostory refers to creating a copy of the remote repository (i.e. the copy on GitHub or Dev Ops) on the disk on your local machine (i.e. your DfE laptop). For an R project, there are two basic options to choose from for doing this:

- using the R-Studio new project wizard, or
- using `git BASH`.

We'd recommend trying the different options across your working group.

Cloning in `git BASH`

You can open up a `git BASH` terminal, by typing `git BASH` in the Windows search bar and select `git BASH` when it comes up. With a terminal, you can interact with it just by typing, similar to working in the R console in RStudio. First let's make a directory in which to store our repositories:

```
mkdir repos
```

We can then move into the directory we just created using:

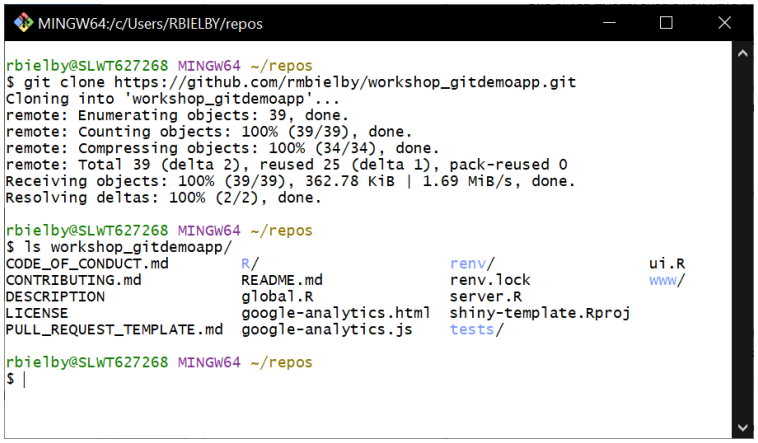
```
cd repos
```

Now grab the repo url and replace `<repo_url>` in the next command with the actual url:

```
git clone <repo_url>
```

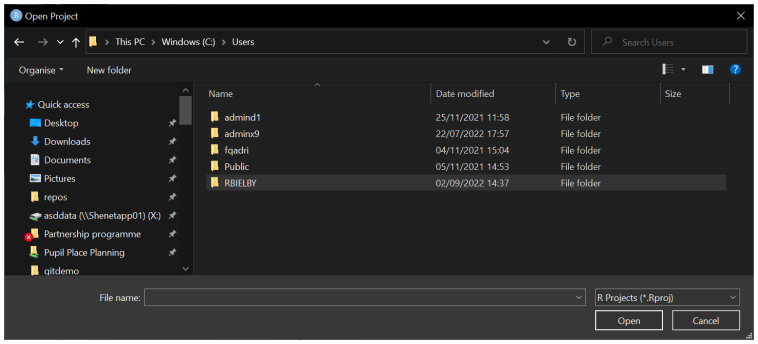
You should get some messages letting you know git is connecting to the server and cloning the repository and it should look something like the figure below.

Figure 5: Cloning a repository in git BASH



If all went well, you'll now have a complete copy of the repository on your laptop. To open the repository in RStudio, start up RStudio and select Open project. In the file explorer window that opens up, type `C:\Users\` and hit enter (see the screenshot below) and then open up your home folder.

Figure 6: Open a cloned project in RStudio

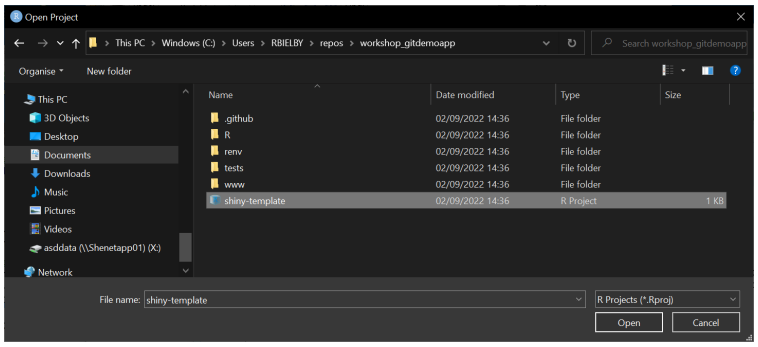


Then navigate into `repos` and the repository folder. The full path should be something along the lines of:

`This PC > Windows (C:) > Users > <USERNAME> > repos > <REPONAME>`

Select the R project file and select open.

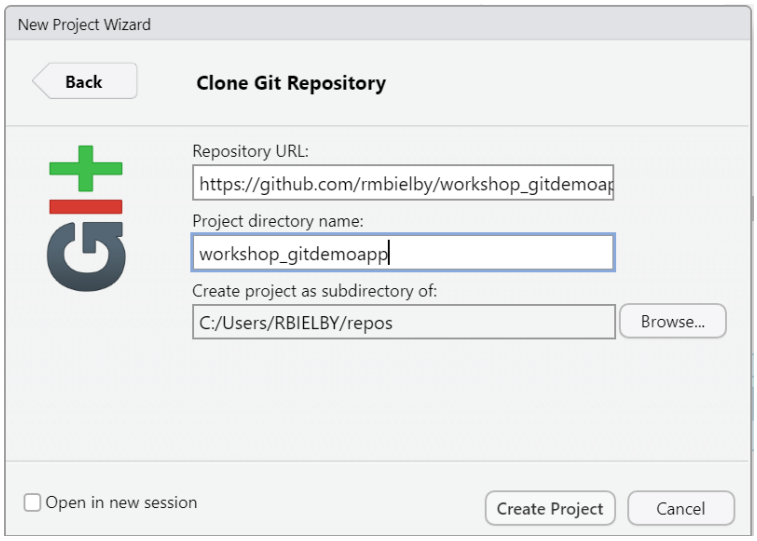
Figure 7: Open a cloned project in RStudio



Cloning using the RStudio wizard

If that looks like a bit too much text based effort, RStudio offers a way to clone a repository with it's New project wizard. To do this navigate the menu bar to **File > New Project...**, select **Version Control** and then Git. This opens up a dialogue box to enter the repository url and select where to save it. As with the git BASH version, copy and paste your remote repo URL here and set a directory where you want it saved on your laptop.

Figure 8: Clone a project using the RStudio git wizard



A note on local repository clones and OneDrive

Note that saving a repository within your OneDrive folder structure can cause some awkward issues. If you use git to perform version control on a repositorry saved within a OneDrive folder, you may start receiving warning messages that large numbers of files have been removed from OneDrive. In addtion, it can put a heavy burden on your internet connection as OneDrive tries to keep up with changes to the files managed by git. Best practice therefore is to store your repositories somewhere outside your OneDrive file structure. We recommend creating a `repos` directory within your base User directory (i.e. `C:\Users\<USERNAME>\repos\`. Windows sometimes tries to make it awkward for you to navigate to places on your laptop outside of the OneDrive folders, so a useful tip is to add your `repos` folder to your Quick access list in File Explorer.

Controlling packages with renv

Running the dashboard template locally

Summary

Resources and further reading

Basics of git

Adding, commiting and pushing

Pulling from the remote repository

Summary of git basics

Resources and further reading

Working collaboratively with git

Git only really makes proper sense once multiple people start working on a project collaboratively. Solo working, git is useful for version control and syncing your work to a remote repository site like Dev Ops and GitHub, but doesn't necessarily offer all that much more beyond that. Once we start working collaboratively however, the benefits of using git (alongside GitHub or Dev Ops) become more apparent. We'll now look further into this with some worked examples.

Branches and splitting tasks

Task management

One useful management tool that we can use from GutHub is the Issues tab. Here we can create individual tasks, assign them to team member and then create new "branches" from those tasks. You can think of branches as self contained copies of the repository that complementary or conflicting differences with all other branches in the repository.

Task 1a

Task 1b

Task 1c

Department for Education

About this publication:

enquiries www.education.gov.uk/contactus
download www.gov.uk/government/publications

Follow us on Twitter: @educationgovuk

Like us on Facebook: facebook.com/educationgovuk