National College of Ireland
MSc/PGDip in Data Analytics
Analytics Programming and Data Visualisation

Tutorial 1 - Creating a working environment for data analytics and visualisation

Prepared by Noel Cosgrave *National College of Ireland*

In this tutorial we will install software tools that we will be using throughout the module, namely Anaconda, GitHub Desktop and Docker Desktop. We will also explore some basic tasks that can be performed using these tools.

Keywords: Anaconda, Jupyter Notebook, GitHub, Docker

Prerequistes

Your computer must meet the minimum specification required for this programme, i.e. it should have a Core i5 processor or better, a minimum of 12GB RAM and 250GB free disk space. Ideally your computer should have SSD-based storage.

If you are running Windows, your system must have hardware virtualisation enabled. This is done using the hardware settings (formerly called the BIOS) of your computer. How this is accessed depends on the make and model of your computer.

Typically this will involve pressing the [F2], [F10], or [Del] key before Windows starts and then checking that Intel Hardware Virtualisation is enabled in the System Configuration settings. Consult your computer's user manual for details of the correct keys to press to access the hardware settings.

Finally, you should be running a recent version of Windows or MacOS. Older versions of these operating can prove unreliable with the software we will be installing.

Anaconda

Install Anaconda

Go to https://www.anaconda.com/download and download the Python 3.11 version of Anaconda for your operating system.

Mac users should first ensure that they download the correct version for the processor used in their system, choosing the 64-Bit Graphical Installer for Intel-based systems and 64-Bit (M1)

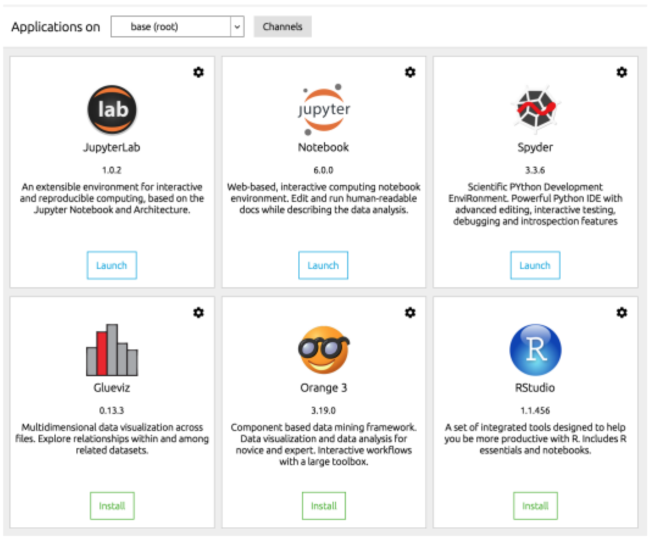
Graphical Installer for Apple Silicon based systems.

They should then open the package they downloaded and start the install, clicking *Continue* as necessary and accepting licenses when required.

Windows users should click *Next* as required, making sure to retain the default settings for installation.

Start Anaconda Navigator and install optional extras

Once Anaconda Navigator has been started, you should see something similar to screenshot in figure 1.



JupyterLab
1.0.2
An extensible environment for interactive and reproducible computing, based on the Jupyter Notebook and Architecture.

Notebook
6.0.0
Web-based, interactive computing notebook environment. Edit and run human-readable docs while describing the data analysis.

Spyder
3.3.6
Scientific PYthon Development EnviRonment. Powerful Python IDE with advanced editing, interactive testing, debugging and introspection features

Glueviz
0.13.3
Multidimensional data visualization across files. Explore relationships within and among related datasets.

Orange 3
3.19.0
Component based data mining framework. Data visualization and data analysis for novice and expert. Interactive workflows with a large toolbox.

RStudio
1.1.456
A set of integrated tools designed to help you be more productive with R. Includes R essentials and notebooks.

Figure 1: Anaconda Navigator

JupyterLab and Jupyter Notebook are installed by default. Other tools can be installed as required:

- **GlueViz** is a particularly useful tool for exploratory data analysis.
- **Orange** is a visual data mining tool that can be helpful for initial experimentation with models. However, it is not a substitute for well thought-out and properly written Python or R code.

For this module we will be using **Jupyter Notebook** exclusively.

GitHub

Set up a GitHub student account

For students, Github offer free access to features that are normally provided excluslivelly to paying customers. This includes private repositories. Please make sure you set up a student account rather than a general account by signing up through https://education.github.com/students.

As part of this deal, you will be offered a "student pack" that includes credits on AWS Educate and Microsoft Azure as well as software from JetBrains, such as PyCharm, an excellent Python Integrated Development Environment.

Install the GitHub desktop client

First go to https://desktop.github.com/ and download the correct version for your operating system. Users of Apple M1, M2 or M3 based systems should note that there is a separate link on this page to download the software for their systems.

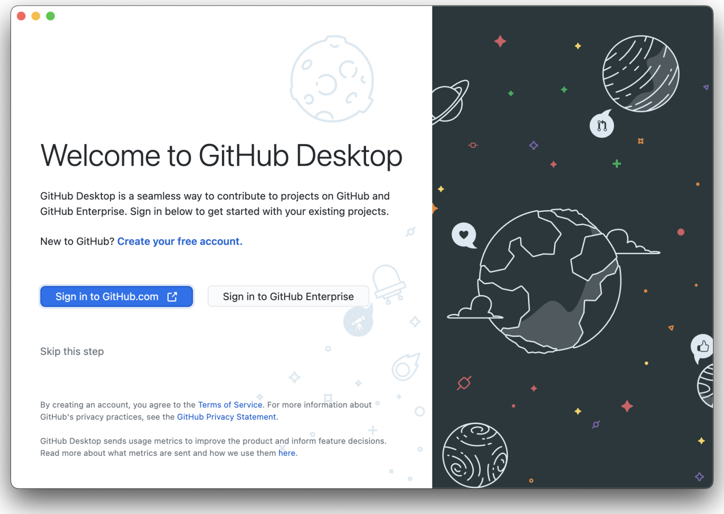
For Mac users, the download is a *zip* archive that must first be unpacked. After the unpacking is complete, you will have a file *GitHub Desktop.app*. This should be moved to your system's *Applications* directory.

Windows users will have downloaded an executable file. This must be run to install the GitHub Desktop client.

After installing the desktop client, you will need to sign in to Github. You will see an opening screen similar to that shown in figure 2.

Click on the *Sign In to GitHub.com* button. A pop up window similar to that shown in figure 3 will be displayed.

This will open a link in your browser, as shown in figure 4, where you should enter the username and password you created in the previous step.



Welcome to GitHub Desktop

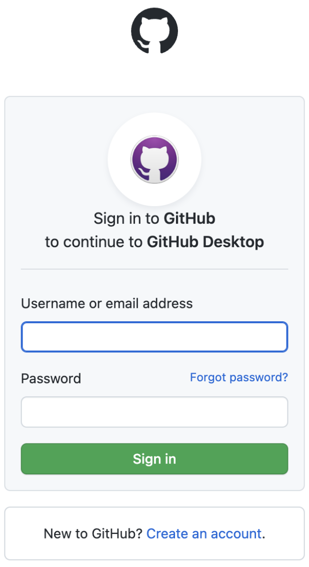
GitHub Desktop is a seamless way to contribute to projects on GitHub and GitHub Enterprise. Sign in below to get started with your existing projects.

New to GitHub? Create your free account.

By creating an account, you agree to the Terms of Service. For more information about GitHub's privacy practices, see the GitHub Privacy Statement.

GitHub Desktop sends usage metrics to improve the product and inform feature decisions. Read more about what metrics are sent and how we use them here.

Figure 2: GitHub Desktop



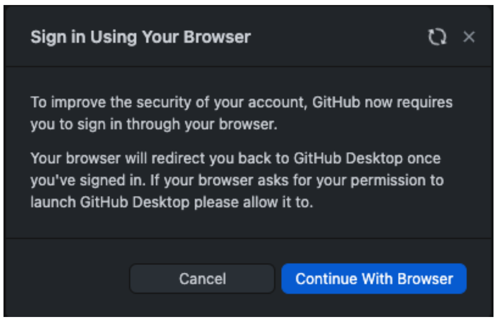
Sign in to **GitHub**
to continue to **GitHub Desktop**

Username or email address

Password

New to GitHub? Create an account.

Figure 4: GitHub Sign In


Sign in Using Your Browser

To improve the security of your account, GitHub now requires you to sign in through your browser.

Your browser will redirect you back to GitHub Desktop once you've signed in. If your browser asks for your permission to launch GitHub Desktop please allow it to.

Figure 3: GitHub Sign In Popup

GitHub Basics

Creating repositories

Repositories are a method of organising projects and can contain files of various types, including program code, data files and any other type of file that you need for your project.

In the upper-left of the GitHub Desktop window, click on *Add* and then select *Create New Repository…* from the pop-up menu, as shown in figure 5.

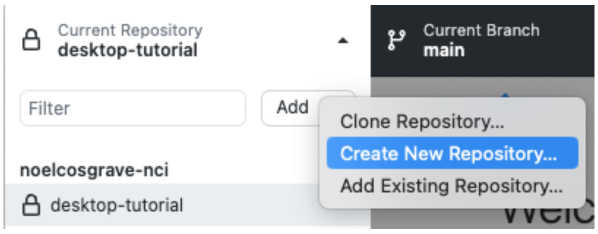


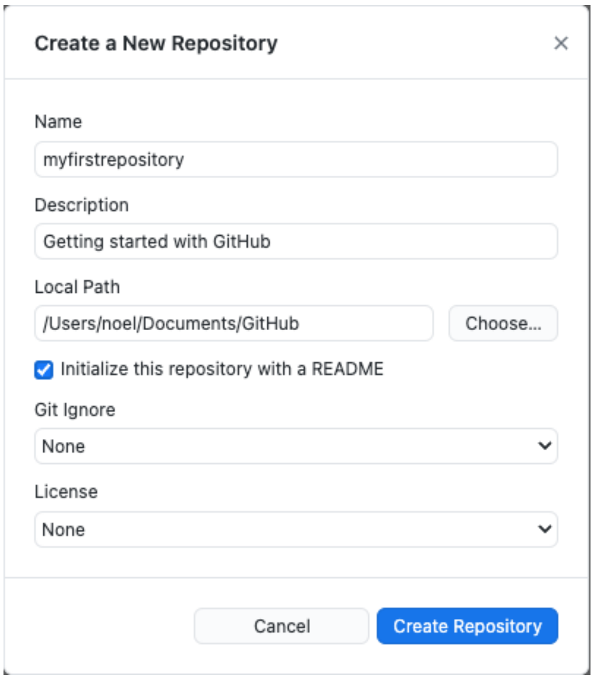
Figure 5: Creating a GitHub repository

In the pop-up dialogue box (see figure 6), enter in the details of your new repository. At a very minimum, you must give the repository a name. It is also highly recommended that you also give the repository a description. Make sure to note the local directory on your system where the repository will be created.

Note: Windows users should avoid creating a repository in any directory shared using OneDrive.

In most cases, a repository should include a README file. This file will eventually contain important information about your project and is written using the Markdown language. For further information on Markdown, see the Markdown Cheat Sheet.

When all details have been correctly entered, click on the *Create Repository* button.


Create a New Repository

Name: myfirstrepository

Description: Getting started with GitHub

Local Path: /Users/noel/Documents/GitHub

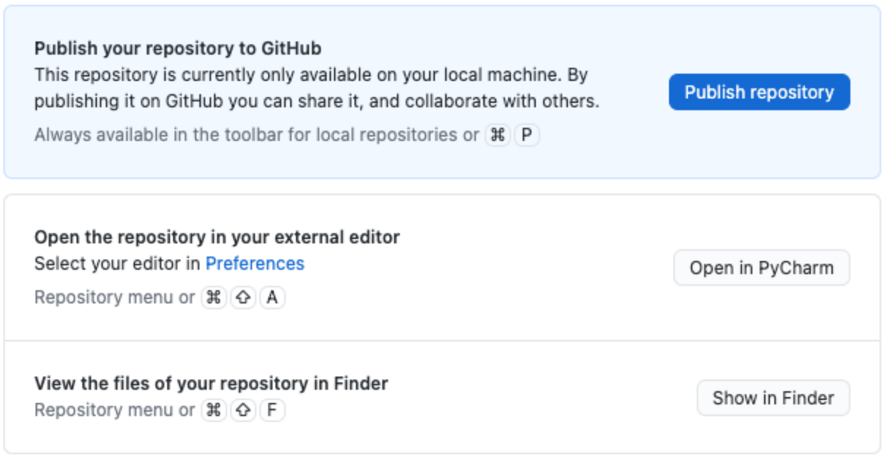
✓ Initialize this repository with a README

Git Ignore: None

License: None

Figure 6: Creating a GitHub repository

Then publish your repository to GitHub by clicking on the *Publish repository* button as seen in figure 7.

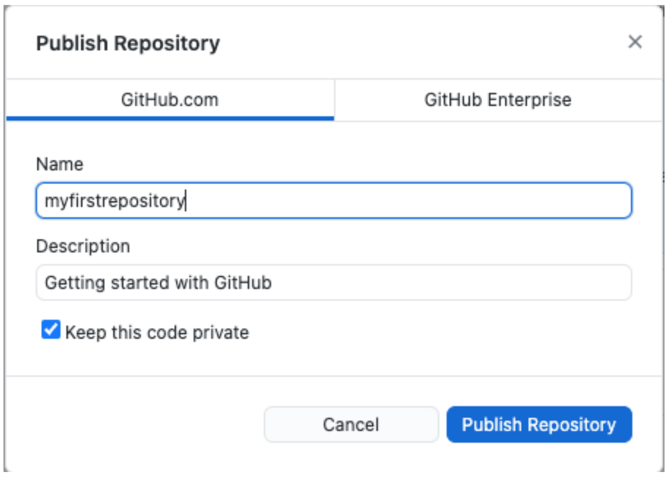

Publish your repository to GitHub
This repository is currently only available on your local machine. By publishing it on GitHub you can share it, and collaborate with others.
Always available in the toolbar for local repositories or ⌘ P

Open the repository in your external editor
Select your editor in Preferences
Repository menu or ⌘ ⇧ A

View the files of your repository in Finder
Repository menu or ⌘ ⇧ F

Figure 7: Publishing the repository

Confirm the details of your repository, as shown in figure 8 and then click the *Publish Repository* button.



Name: myfirstrepository

Description: Getting started with GitHub

☑ Keep this code private

Figure 8: Publishing the repository

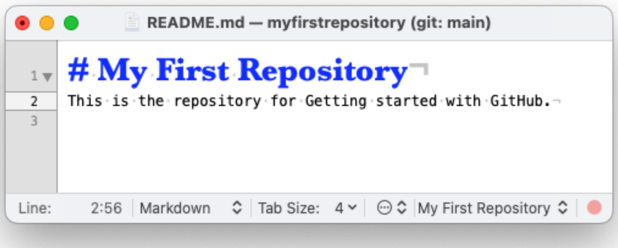
Committing changes

Let's make a change to the README file created when the repository was initialised. Locate the README file in the repository's directory and open it with any plain-text editor.

If you have difficulty locating the directory, in the GitHub desktop client, click on the *Show In Finder* button (MacOS) or the *Show in Explorer* button (Windows) as seen in figure 7 on the previous page.

For Windows users, Notepad++ is highly recommended. For MacOS users, TextMate provides similar functionality.

Once you have located and opened the README file, edit it as shown in figure 9 and save the file.



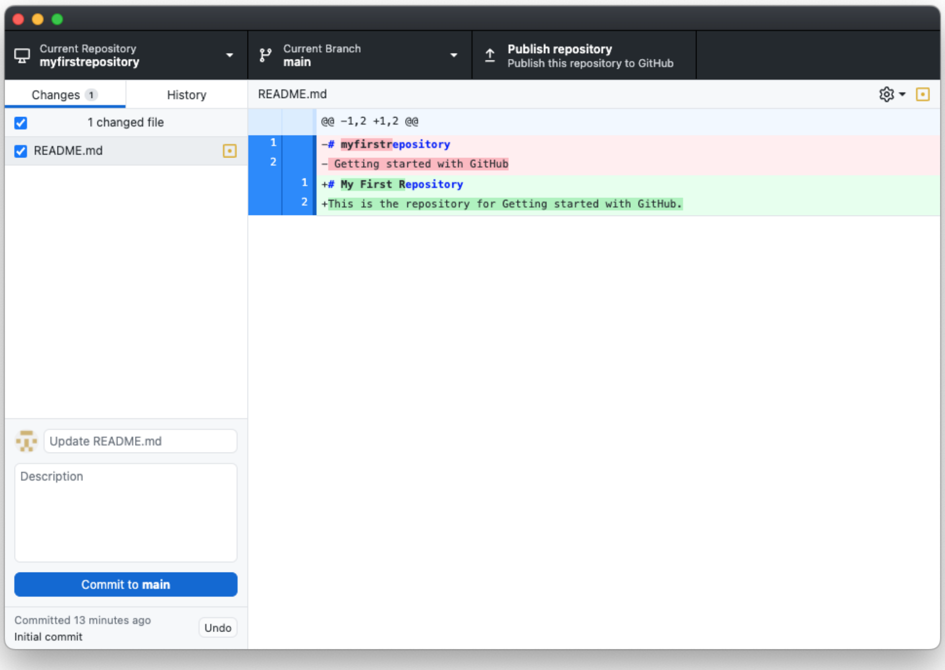
```
# My First Repository
This is the repository for Getting started with GitHub.
```

Figure 9: Editing the README file

Back in the GitHub Desktop client, the change should be detected, as can be seen in figure 10 on the following page.

You will note that text that has been deleted is shown with a red background and text that has been added is shown with a green background.

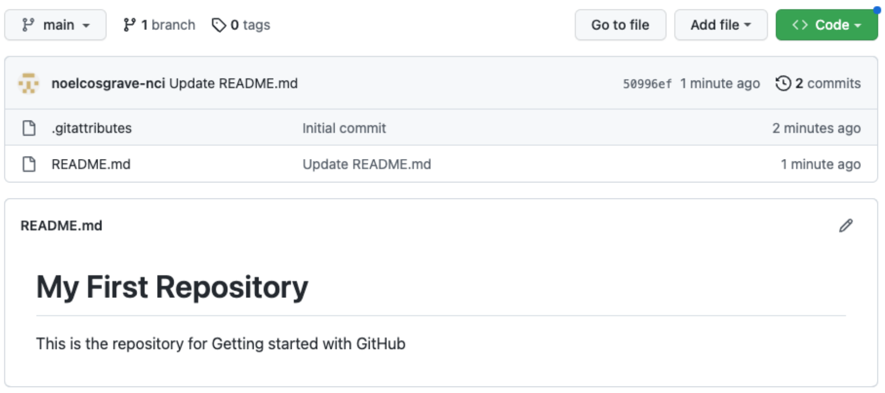
Ensure that the update has a title (shown in the figure as update README.md) and click on the *Commit to main* button. When the *Push Origin* button appears, click it to copy the changed files from your local system to Github.com.



```
@@ -1,2 +1,2 @@
-# myfirstrepository
- Getting started with GitHub
+# My First Repository
+This is the repository for Getting started with GitHub.
```

Figure 10: Repository content change

You can verify that the changes have been committed by opening github.com in your browser, the navigating to the repository and checking that the content of README.md has changed, as seen in figure 11.



noelcosgrave-nci Update README.md 50996ef 1 minute ago 2 commits

.gitattributes	Initial commit	2 minutes ago
README.md	Update README.md	1 minute ago

README.md

My First Repository

This is the repository for Getting started with GitHub

Figure 11: Verifying changes to the README file

Pulling changes

When others make changes to the repository, you must pull those changes from Github.com to your local system. Click on the *Fetch origin* button in the top bar of the GitHub Desktop client, as shown in figure 12.

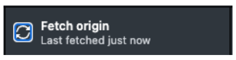


Figure 12: The *Fetch origin* button

Then click on the *Pull origin* button to download the changes to your system.

Creating branches

Using branching, you can maintain multiple separate versions of a repository.

When initially created, a repository will have one automatically-created branch named *main*. You can create new branches from the *main* branch and use these branches to maintain multiple different versions of a project.

To create a branch, first click on the *Current Branch* button in the toolbar at the top of the GitHub Desktop window. A popup window will be displayed, as shown in figure 13.

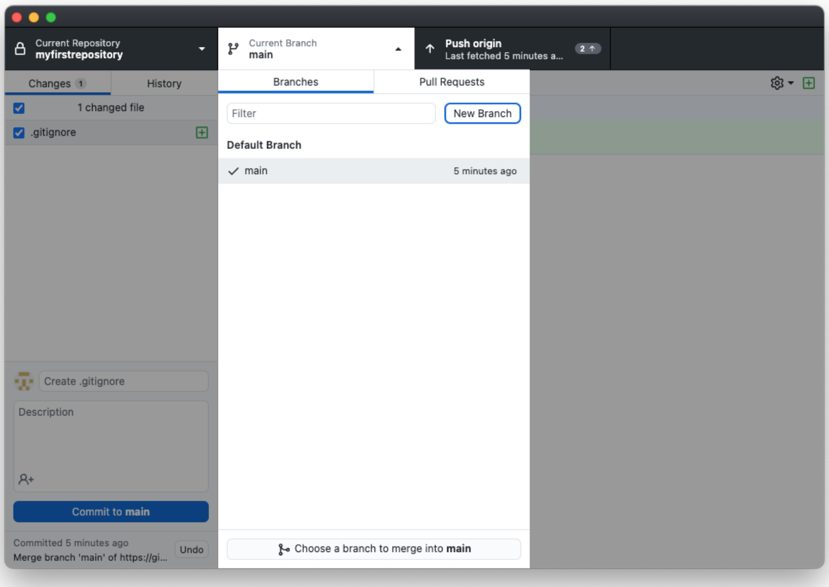
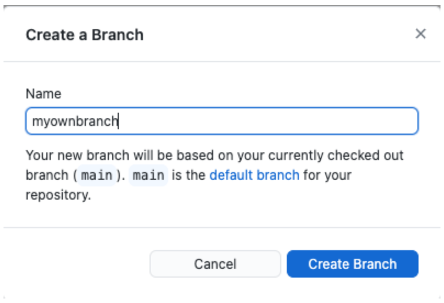


Figure 13: Creating a branch

Click on the *New Branch* button. A popup window will be displayed, as can be seen in figure 14.


Create a Branch

Name

myownbranch

Your new branch will be based on your currently checked out branch (`main`). `main` is the default branch for your repository.

Figure 14: Specifying a name for the new branch

Click on the *Create Branch* button to create the new branch. The contents of the new branch should be exactly the same as those in the *main* branch at the time the new branch was created.

This feature allows you to make experimental changes to the source code for a project without altering the code in the *main* branch. Any changes you make to your own branch will only be available in the *main* branch when you merge those changes. We will come back to this later in the module.

Note: When you create a branch from the *main* branch, you are making copy or taking a "snapshot" of the *main* branch at that particular point in time. If another contributor to your repository makes changes to the *main* branch while you are working on your own branch, you will need to pull in those updates to keep your branch up-to-date.

We will return to GitHub in the coming weeks, and in particular we will explore how repositories can be managed using the command line (Windows) or terminal (MacOS).

Docker

Docker is a framework that enables you isolate your software applications from specific infrastructure requirements in order that you can speed up software development.

Docker allows you to develop and run an application in lightweight isolated environments called **containers**. Containers contain everything that is required for you to run the application, freeing you from worrying about what software and libraries are available on the host system. You can create and share container specifications, secure in the knowledge that anyone you share those specifications with will have a container that works in exactly the same manner.

If you are already familiar with virtual machines, you can think of containers as being a much less resource-hungry alternative to VMs.

Docker Desktop

Docker Desktop is application available for both Windows and MacOS that includes the core of Docker - the Docker Engine, as well as the Docker command line client, a tool for creating containers called Docker Compose, as well as other useful utilities.

Install Docker Desktop

In your browser, navigate to https://www.docker.com/products/docker-desktop/ and download the correct version for your operating system. Once again, users of Apple Silicon (M1/M2/M3) systems should note that there is a separate download link for their version of MacOS.

Windows users may need to update Windows Subsystem for Linux to version 2 before the Docker engine will successfully start. To do this, open a Command Prompt **with Administrator privileges** and run the following command. Restart your computer once the update completes.

```
wsl --update
```

Introduction to Docker

We will be using Docker extensively over the remainder of this semester. Today our objective is to look at a basic example of running a Docker container.

If you have not already done so, open the Docker Desktop application and ensure that Docker is running. Once the application is open, move your mouse to the lower right hand corner of the window. If the engine is running, there should be a green bar. If you mouse over this bar, the pop-up message should display *Engine running*, as shown in figure 15.

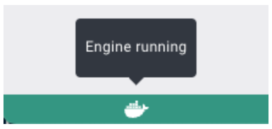


Figure 15: Checking that the Docker engine is running

Next open a terminal session (MacOS) or Command Prompt (Windows) and type the following command:

```
docker run -d -p 80:80 docker/getting-started
```

Then in your web browser, navigate to http://localhost/.

This should open the Docker tutorial on the *Getting Started* page. Read through this page to gain an insight into the basics of Docker.