Darwin Harbour sediment monitoring program analysis application manual

Murray Logan

12/07/2024

Table of contents

About

This document comprises the manual for the Darwin Harbour sediment monitoring program analysis application. It provides information on:

- a broad overview of the structure of the application
- the application dependencies and how to install them
- starting the application
- progressing through the analysis pipeline
- visualising, interpreting and extracting outputs

Structural overview

R Graphical and Statistical Environment offers an ideal platform for developing and running complex statistical analyses as well as presenting the outcomes via professional graphical/tabular representations. As a completely scripted language it also offers the potential for both full transparency and reproducibility. Nevertheless, as the language, and more specifically the extension packages are community developed and maintained, the environment evolves over time. Similarly, the underlying operating systems and programs on which R and its extension packages depend (hereafter referred to as the *operating environment*) also change over time. Consequently, the stability and reproducibility of R codes have a tendency to change over time.

Docker containers

One way to attempt to future proof a codebase that must be run upon a potentially unpredictable operating environment is to **containerise** the operating environment, such that it is preserved to remain unchanged over time. Containers (specifically docker containers) are lightweight abstraction units that encapsulate applications and their dependencies within standardized, self-contained execution environments. Leveraging containerization technology, they package application code, runtime, libraries, and system tools into isolated units (*containers*) that abstract away underlying infrastructure differences, enabling consistent and predictable execution across diverse computing platforms.

Containers offer several advantages, such as efficient resource utilization, rapid deployment, and scalability. They enable developers to build, test, and deploy applications with greater speed and flexibility. Docker containers have become a fundamental building block in modern software development, enabling the development and deployment of applications in a consistent and predictable manner across various environments.

Shiny applications

Shiny is a web application framework for R that enables the creation of interactive and data-driven web applications directly from R scripts. Developed by Rstudio, Shiny simplifies the process of turning analyses into interactive web-based tools without the need for extensive web development expertise.

What makes Shiny particularly valuable is its seamless integration with R, allowing statisticians and data scientists to build and deploy bespoke statistical applications, thereby making data visualization, exploration, and analysis accessible to a broader audience. With its interactive and user-friendly nature, Shiny serves as a powerful tool for sharing insights and engaging stakeholders in a more intuitive and visual manner.

Git and github

Git, a distributed version control system, and GitHub, a web-based platform for hosting and collaborating on Git repositories, play pivotal roles in enhancing reproducibility and transparency in software development. By tracking changes in source code and providing a centralized platform for collaborative work, Git and GitHub enable developers to maintain a detailed history of code alterations. This history serves as a valuable asset for ensuring the reproducibility of software projects, allowing users to trace and replicate specific versions of the codebase.

GitHub Actions (an integrated workflow automation feature of GitHub), automates tasks such as building, testing, and deploying applications and artifacts. Notably, through workflow actions, GitHub Actions can build docker containers and act as a container registry. This integration enhances the overall transparency of software development workflows, making it easier to share, understand, and reproduce projects collaboratively.

Figure **??** provides a schematic overview of the relationship between the code produced by the developer, the Github cloud repositiory and container registry and the shiny docker container run by user.

Installation

Installing docker desktop

To retrieve and run docker containers requires the installation of Docker Desktop on Windows and MacOSx

Windows

The steps for installing Docker Desktop are:

- **Download the Installer:** head to https://docs.docker.com/desktop/install/windows-install/ and follow the instructions for downloading the appropriate installer for your Windows version (Home or Pro).
- **Run the Installer:** double-click the downloaded file and follow the on-screen instructions from the installation wizard. Accept the license agreement and choose your preferred installation location.
- **Configure Resources (Optional):** Docker Desktop might suggest allocating some system resources like CPU and memory. These settings can be adjusted later, so feel free to use the defaults for now.
- **Start the Docker Engine:** once installed, click the "Start Docker Desktop" button. You may see a notification in the taskbar - click it to confirm and allow Docker to run in the background.
- **Verification:** open a terminal (or Powershell) and run `docker --version`. If all went well, you should see information about the installed Docker Engine version.

Additional Tips:

- Ensure Hyper-V (virtualization) is enabled in your BIOS settings for optimal performance.

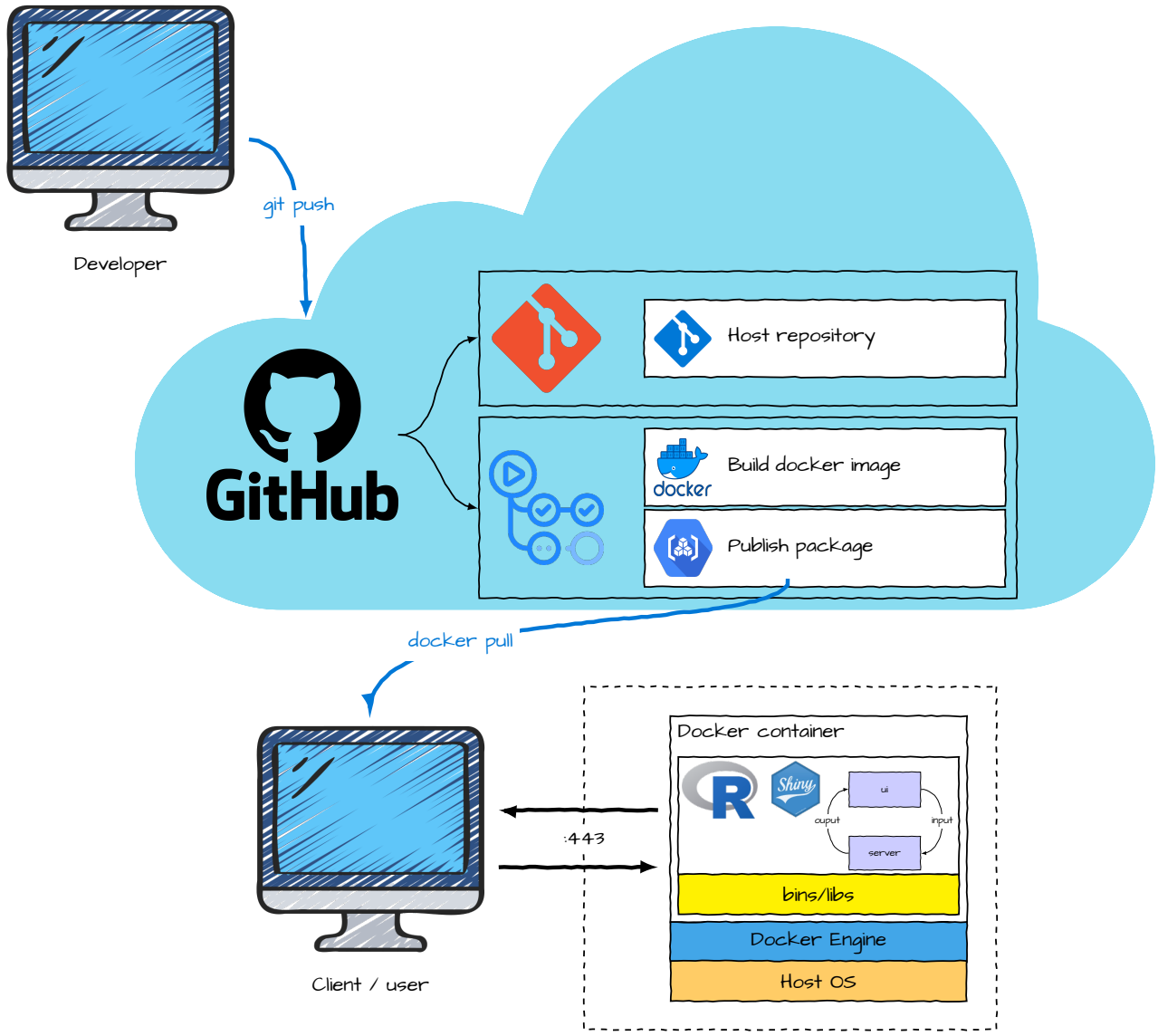

Figure 1: Diagram illustrating the relationship between the code produced by the developer and the shiny docker container utilised by user with a Github cloud conduit. The developed codebase includes a Shiny R application with R backend, `Dockerfile` (instructions used to assemble a full operating environment) and github workflow file (instructions for building and packaging the docker image on github via `actions`).