RStudio Server

Jim Harner

10/2/2020

1.2 RStudio Server

RStudio is a powerful integrated development environment (IDE) primarily used for developing code for R projects, including R packages. RStudio supports the integration of R, Python, bash, and SQL code chunks,among other languages, within Rmarkdown documents (and notebooks).

As a frontend to R, it can be run on your local machine (Mac or Windows) natively, but this is difficult for doing data science, which requires a vast number of add-on components. As a result, we use RStudio Server, which runs on Linux and is accessed with a web-based client.

1.2.1 Running RStudio Server in a Docker container

We run RStudio Server in a Docker container within a virtual computing environment on your local machine or in the cloud. The development repository (repo) for this environment is called `rspark` and can be found on GitHub here: github.com/jharner/rspark. This virtual environment will allow you to run R, Spark, Hadoop, etc.

`rspark` can be installed in one of several ways:

- running Docker directly on your computer (https://github.com/jharner/rspark);
- downloading images from Docker Hub to your computer (https://github.com/jharner/rspark-docker);
- installing Vagrant/ VirtualBox on your own computer (https://github.com/jharner/rspark-vagrant);
- running Docker on an AWS instance.

The first option for `rspark` is for development purposes. It can be used, but it changes often. Instead, you should use `rspark-docker` (see Subsection 1.3.3) or `rspark-vagrant` (see Subsection 1.3.4). The AWS solution involves an Amazon account, which is inexpensive, but not free. It is more complex to set-up, but easy to launch (see the ansible scripts in `rspark`).

The Docker containers run Ubuntu Linux as a base. At this point `rspark` has the following containers: rstudio, postgres, hadoop, hive, spark, and three workers. I plan to add other containers for other components of the Hadoop and Spark ecosystems. Ultimately, I would like to build a complete system for both static and streaming data.

Running RStudio Server on your local machine using Docker or Vagrant requires a terminal and some knowledge of Linux. The steps for doing this are described in Section 1.3 after Linux is introduced.

1.2.2 Running RStudio Server on AWS

Amazon Web Services (AWS) allows users to create virtual machines in the AWS cloud among other services. The cost of using AWS services is based on the computing power required, the amount of storage, and the run-time of the service. However, this cost is minimal for the services needed in this course. Thus, to use

`rspark` with AWS, you need to be prepared to pay for the cloud services you utilize unless you have received an academic free allocation.

Prerequisites:

- Amazon Web Services Account
- Modern Browser (Safari, Chrome, or Firefox)

Detailed instructions for running `rspark` on AWS are available here: AWS rspark

The pre-built image on AWS is called `rsparkbox` and it contains this tutorial. Once you follow the steps in the directions, connect to your `rspark` server through a web browser.

Note If you are given an IP address for your instance, enter it into your browser's URL bar as 'http://0.0.0.0:8787' replacing '0.0.0.0' with the IP address of your instance. In this case you do not need an AWS account. Log into RStudio with the credentials:

```
username: rstudio
password: rstudiojh
```

IMPORTANT: When you have finished using `rspark`, you need to stop or terminate your EC2 instance. If you neglect to do this, you will be charged by Amazon for the duration your instance is left running.

Shutting Down You have two primary options when shutting down an AWS instance of `rspark`. The difference between the two options is determined by your need to preserve your workspace.

Preserving the `rspark` Workspace In order to preserve your workspace and leave your files on the instance itself:

1. Return to your AWS Instance Dashboard;
2. Ensure your `rspark` instance is selected;
3. Click `Actions`;
4. Click `Instance State`;
5. Select `Stop`.

This will suspend your instance, rather than terminate it. You will be charged a small amount per hour for storage when your instance remains suspended (typically fractions of a cent per day).

When you are ready to work again:

- Repeat steps 1. through 4. from above, but
- Select `Start` in step 5.

Note: the IP address of your instance will have changed after being restarted. This new IP address is needed for connecting to RStudio so be sure to copy it down.

Disposing of the `rspark` Workspace *IMPORTANT: Be sure to download all important files to your local computer using RStudio's `Export...` command from the `More` submenu of the Files menu before terminating your AWS instance.*

If you do not need to preserve the state and workspace of your `rspark` instance, be sure to terminate the instance. Follow these steps to ensure you are not charged for AWS services while it is not actively being used.

- Repeat steps 1. through 4. from above, and then

- Select `Terminate` in step 5.

NOTE: If your instance is terminated, all start-up steps must be redone to start a new instance.

1.2.3 Principal RStudio Components

As an IDE, RStudio has many panels and menus for software coding. Although RStudio is best learned by using it, brief summaries follow.

The Source Panel Once you log into RStudio Server you will see four panels, which can be customized by choosing `Global Options ...` from the `Tools` menu. You can then rearrange what appears in the panels to suit your needs. I would recommend that you leave `Source` in the upper left panel and the `Console` (and `Terminal`) to the lower left panel. It is useful to have "History, Plots, Packages, Help, and Viewer" in the upper right panel so that output is beside the source code.

You can use R interactively from the `Console`, but most of your work will consist of developing source files in the Source panel. For the most part you will be creating R Markdown files and/or R Notebooks, but in some cases you will develop R scripts. New files can be created by selecting `New File` from the `File` menu and then selecting the type of file desired, typically either `R Markdown` or `R Notebook`. We usually specify `html_document` and `html_notebook` as outputs.

The notebook feature is very powerful (the most powerful I have seen) for interactively debugging code. It creates output for each chunk in a very informative way. LaTeX equations are displayed as typed.

Other file types of interest in this class are: `R Script`, e.g., for Hadoop files, `Shell Script` for `bash` scripts, and `Text File` for data in a text format. The Shiny Web App is for producing reactive client/server web apps, typically with interactive plots.

Files support built-in LaTeX for mathematics and code chunks containing not only R, but also Bash, Python, C++, SQL, etc. You can mix these code chunks within a single document.

R supports Projects which allow related files to be organized together. Projects can then be tied to version control systems such as `git`. We will be using `rspark-tutorial` in this tutorial, optionally as an RStudio Project.

The Console Panel The `Console` is for interactively executing R statements. Certain types of R statements generated from other sources also appear here. The `R Markdown` and `Jobs` tabs are transitory and appears when knitting takes place. R Markdown errors are also displayed in the `R Markdown` tab.

The `Terminal` tab provides access to a command line interface (CLI) to Linux running in the RStudio container using the `bash` shell. By default, you open a session in `Terminal 1`. Your working directory is displayed to the right of the terminal name, which can be changed. The prompt is given by `rstudio@CONTAINER_ID`. Additional terminals can be launched by selecting `Terminal` and then `New Terminal` in the `Tools` menu. Each terminal can be set to a specific directory for executing commands.

The RStudio Terminal does not allow us to `ssh` into the another running container directly. However, an external termainal app can be used assuming it is running in an environment with Docker installed. To `ssh` into another container using your favorite `term` app, use `docker exec`, e.g.,

```
docker exec -it postgres bash
```

to access the `postgres` container. Once inside of `postgres`, you can run `psql` to access databases, etc. Note: Section 5 of module 2 shows how to access databases using `psql` from within the `rstudio` container.

Environments and Files R has a workspace in which all assigned data structures or functions are kept. These structures or functions show up in the `Environment` tab. Files and directories are available in the `Files` tab. Spark data structures appear in the `Connections` tab.

Viewers The `html` output is displayed in the `Viewer` and is also saved in the working directory. `Word` and `pdf` outputs are saved as files in the working directory and `pdf` files are also displayed in a `Viewer`. Plots are saved in the `Plots` viewer, although they are usually embedded in output documents.

Packages R has a very robust package system currently consisting of over 18,000 official packages and many more being developed on GitHub ($> 64{,}000$) that do almost everything of interest concerning nearly any type of data. For a complete listing see: rdrr.io. The `Package` tab gives you access to the currently installed packages. Loaded packages have a check beside them. Clicking on the package name brings up its documentation in the Help tab. Each function or data frame can be clicked to get detailed information about their usage.

Clicking `Install` under `Packages` brings up a dialog for installing packages. It is also possible to `Update` currently installed packages.

A comprehensive guide to R packages on CRAN, R-Forge, and GitHub can be found here.