Set up a virtual server on AWS (in anticipation of hosting Shiny apps)

Ronald (Ryy) Glenn Thomas

3/26/23

Table of contents

Photo by Nathan Waters on Unsplash

Introduction

Once we have a working shiny app, we can move on to the task of hosting the app on a (virtual) server to share with our collaborators. There are many ways to accomplish this. In this post, we'll describe how to 'spin' up a server on Amazon Web Service EC2, and in the next post show how, in just a few steps, through the application of Docker, R, Shiny, and Caddy (webserver) functionality we can have a fully functional and secure web app to share with colleagues.

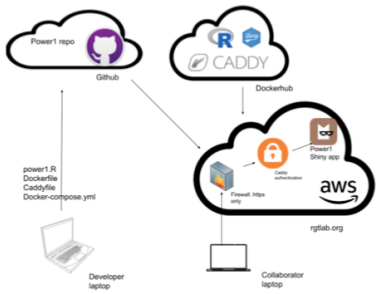


Figure 1: *Data flow*

Hosting

Figure 3 illustrates the tools we'll use and the flow of program and configuration files. In order to host a shiny app, say `power1_shiny`, online we'll need to complete the following tasks: 1. create a virtual server (connected via ssh) with a firewall 2. obtain a static IP address (to identify the server online) 3. obtain a domain name (name for IP address) 4. install and configure a webserver (tool to interact with https protocol requests and respond) 5. obtain and install an SSL certificate (to allow encrypted communication) 6. setup an authentication method (password protection) 7. configure a reverse proxy method (translate https (port 443) requests to Shiny (port 3838)

In this post we'll address the first three of these. Tasks four through seven will be accomplished using docker-compose and described in the next post.

At first glance these 7 requirements can appear daunting, but on closer inspection all can be met with relative ease and minimal cost (using a cloud-hosting service, e.g. Amazon's EC2 or Digital Ocean, and a "leased" domain name from, e.g. GoDaddy, or Amazon's Route 53) or at no cost(if you have your own server with IP address, and domain name)

Select a hosting service

ok! got my shiny app running. Works great! Now how do I get it up on the web and shared with my client?

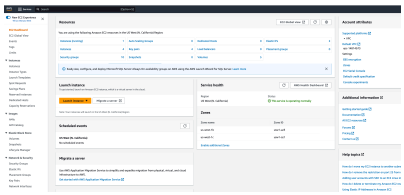
There are a number of cloud based server options to choose from: Microsoft Azure, Oracle, Google Cloud, Amazon AWS EC2, Digital Ocean to name a few. Each has their own approach to setting up a custom virtual server. Several have free or low-cost service tiers available.

An overview of the process with EC2 follows.

AWS is a reasonable choice for setting up a small custom server.

To start open the EC2 console.

```
https://aws.amazon.com/console
```



Choose regional service. For me its "N. California".

Create an account or sign in and navigate to the EC2 dashboard.

Along with selecting a server you'll need to set up a working environment. This environment consists of four main components:

1. ssh key-pair to allow you to remotely and securely login to the virtual server.
2. A firewall to restrict access to only secure connections.
3. A static IP address. This is required for maintaining the link between the domain name and the server when rebooting. (The default is for the instance/server to be assigned a new IP each time its rebooted.)

and

4. A domain name, say rgtlab.org. A domain name is not required but will facilitate collaborator access by not needing to use the IP address directly.

The host host setup process is a simpler if you set up the working environment first.

Ssh key pair

The first time you create an AWS account you need to exchange an ssh (secure shell) key pair with AWS. This will allow you to login remotely to any server you launch. You can generate a ssh key pair locally on you workstation and upload the public key to EC2. To do this create a directory to hold the keys. e.g. `~/.ssh`. From inside `~/.ssh` directory you can generate the keys with the command

```
ssh-keygen -m PEM
```

In the dialog that ensues name the key prefix something like `power1_appssh-rsa`.

Back in the browser on EC2 select `security/keys`, A dialog starts and asks for the location of the public key. Browse to the `~/.ssh` directory on your workstation and import the public key `power1_appssh-rsa.pub`.

Alternatively, you can select `Create Key Pair` in EC2. Give the pair a name, say `power1_appssh`, and a pair of keys will be created and the private key `power1_appssh.pem` will be downloaded to you local machine. In my case to the default `~/Downloads` directory. Move the file to the `~/.ssh` directory. Change the access permissions: `sudo chmod 600 power1ssh.pem`. - update software on server: `sudo apt-get update`

firewall

6. Add security group, e.g. 'power1.firewall' allowing 22 (ssh), and 443 (https).

- Create a firewall using `Network settings` pane Choose `Create security group` and select `Allow SSH traffic` and `Allow HTTPS traffic`.

The default name for the firewall will be something like `launch-wizard-6`.

Set up static IP address

Use "elastic IP" to get a static IP that can be assigned to the server. Navigate to Network and Security again and select Allocate Elastic IP. An IP will be assigned from the EC2 pool of IPv4 IP addresses. (there is a fee for use of the static IP).

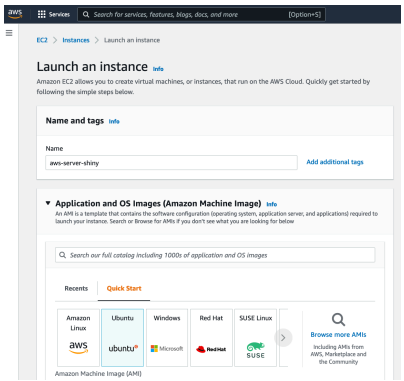
Domain Name

To obtain a dedicated domain name go to godaddy.com or Amazon route 53 to select a domain name and associate it with your Elastic IP.

Once a domain name is aquired, eg rgtlab.org, you want to associate it with your static IP address. To associate domain name rgtlab.org with elastic IP do as follows.

in Route 53:

- click on 'hosted zones' in side panel
- click on rgtlab.org in center panel
- click on checkbox for rgtlab.org type=A line
- then click on edit record in right panel
- change ip address to the Elastic IP (e.g. 13.57.139.31).



Select and launch instance

2. From "Quick Start" in the EC2 dashboard click `Ubuntu` button.

- Name the server, say power1

3. Next choose an instance **type**, e.g. "t2-micro". Different instance types are mixtures of size, processors, memory, instance storage capacity, network performance.
4. click "Next: Configure Instance Details"
5. choose a Key pair (use power1.rsa from your environment) .
6. Add security group, e.g. 'power1.firewall' from your environment.
7. choose 30 GB of EBS General Purpose (SSD) or Magnetic storage
8. click Launch Instance

- Configure 30 GiB of gp2 (general purpose SSD) storage. Define the size and type of disc storage.

Now Launch the Instance.

We still need a static IP address and a domain name.

- Navigate back to the EC2 page.
- Select `Elastic IPs` from the `Network & Security` section.
- check box for `power1`, and choose `Associate IP address` from the `Actions` drop down menu. Choose the `power1server` server from the `Instance` drop down list.

Select Domain Name

Access instance

On your laptop log into server with

```
ssh -i "~/.ssh/power1ssh.pem" ubuntu@power1app.org
```

Appendix: Tips and Tricks

Tip 1.

For convenience, construct a `config` file in `~/.ssh` as:

```
Host ec2
HostName 13.57.139.31 # static IP
User ubuntu # default user on ubuntu server
Port 22  # the default port ssh uses
IdentityFile ~/Downloads/power1.rsa
```

then you can ssh into the new server with

```
sh> ssh ec2
```

Open the Amazon EC2 console at https://console.aws.amazon.com/ec2/

From the top navigation bar, select a Region to create an instance in. For me its "N. California".

Create an account or sign in and navigate to the EC2 dashboard.

In the left side panel select **Key Pairs** (under Network and Security).

At the top right select the **Create key pair** button. A **Key Pair** form will open.

Give the key pair a name. Something like `power1_appssh`. Select a key pair type, suggest `RSA`. Select a **Private key file format**, suggest `.pem`

Below the form select the **Create key pair** button. A pair of keys will be created and the private key `power1_appssh.pem` will be downloaded to you local machine. In my case to the default `~/Downloads` directory.

Move the file to the `~/.ssh` directory: `mv ~/Downloads/power1_appssh.pem ~/.ssh`

Change the access permissions: `sudo chmod 600 power1ssh.pem` to be more restrictive.

** REWRITE TO HERE…. 2023-03-27

In the navigation pane, choose **Instances**.

Select your instance and, in bottom half of the screen, choose the Security tab. Security groups lists the security groups that are associated with the instance. Inbound rules displays a list of the inbound rules that are in effect for the instance.

For the security group to which you'll add the new rule, choose the security group ID link to open the security group.

On the Inbound rules tab, choose Edit inbound rules.

On the Edit inbound rules page, do the following: