Dockerize Shiny Apps

true

2022-07-09

Introduction

This is the first of a series of posts on how to leverage the technologies of: R, Shiny, Docker, Linux, Caddy, and AWS to provide open-source solutions to one of the central challenges in the practice of data science i.e. how to communicate analysis results to clients and collaborators.

This post provides a minimal example of applying SOTA technologies to provide an interactive tool for conducting a power analysis.

Starting with a simple, but still useful, shiny app and ending with a hosted web site with a custom domain name. Lets assume we're just finished developing a 'shiny' new shiny app, named `power0.R`, The source code for the app that resides in the directory `~/APPS/app3`. (where in my case I created the symbolic link `APPS` via the z-shell to `sh> ln -s ~/prj/c060/prj/launch ~/APPS`. We want to check that the app is working as expected locally. To do that we can run it with the shell command `APPS/app3> Rscript -e "runApp("myapp.R",launch.browser=T)"`.

Duck Duck Go. test](file:///Users/zenn/Dropbox/prj/blog/__posts/simpleshinypower0/simpleshinypower0.html)

Consider an app that is a balance of simple and functional – one that calculates the power for a 2-sample t-test as a function of the standardized effect size. Here is our shiny app `myapp.R`:

```r
ui <- fluidPage(
titlePanel("Power Calculator for Two Group Parallel Designs"),
sliderInput("N", "Total Sample Size:", min = 0, max = 300, value = 100),
plotOutput("plot"),
verbatimTextOutput("eff"))

server <- function(input, output, session) {
  delta = seq(0, 1.5,.05)
  pow = reactive(sapply(delta, function(x) power.t.test(input$N, d=x)$power ))
  eff =  renderText(power.t.test(input$N, power=.8)$d)
  output$plot <- renderPlot({
  plot(delta, pow(), cex=1.5, ylab="power")
  abline(h = .8,  col = "red", lwd =2.5, lty = 4)
  abline(v = eff(), col = "blue",lwd =2.5, lty = 4)})
  output$eff <- renderText(
    paste0("Std. effect detectable with power 80% = ", eff()) )
}
shinyApp(ui, server)
```

We are interested in hosting the app on a server to allow others to share in its wonderfulness There are many ways to accomplish this. First off we'll demonstrate a docker based approach.

Naturally, the first step is to dockerize the app. This provides a whole host of advantages over a more traditional approach. Lets discuss.

Photo by Ian Taylor on Unsplash

Here is our dockerfile:

```
FROM rocker/shiny:4.2.0
COPY app.R /srv/shiny-server/
CMD ["/usr/bin/shiny-server"]
```

Methods

Hosting

Once we have a prodeuction ready dockerized hsiny app available, ewe'll want to host it on the interjet; mTho do that we;ll need the following : 1) a virtual server, 2) a domain name 3) an SSL certificate (encrypted communication) 4) an authentication method (password protection) 5) a reverse proxy method 6) an index file (landing page)

This can all be done at no cost if you have your own (self-hosted) server and domain name, or at minimal cost using a cloud-hosting service (e.g. Amazon's EC2 or Digital Ocean) and a "leased" domain name from, e.g. GoDaddy.

On custom EC2 instance

- Open EC2 console.

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

or

```
https://us-west-1.console.aws.amazon.com/ec2/v2/home?region=us-west-1#
```

- log in
- side panel, click "ec2"
- side panel, click "Instances"
- from top bar, click "Launch Instances"

2. From "Quick Start" click Ubuntu button.

- Name the server say shiny-july22

- `Choose an AMI (instance template, operating system):`

Suggest choose "Ubuntu Server 22.04 LTS", but other linux distributions can be utilized, e.g. Red Hat, or SUSE.)

e.g. ubuntu/images/hvm-ssd/ubuntu-jammy-22.04-amd64-server-20220609

3. Next choose an instance **type**, e.g. "t2-micro". (different instances are mixtures of size, processors, memory, instance storage, network performance) click "Next: Configure Instance Details"

*. choose Key pair (use in place aws18.pem) or set up new pair

Add security group, e.g. 'shiny' (sg-0f37c94ac1e1b6250) allowing ports 80 (http), 22 (ssh), 443 (https), and 3838 (shiny). and 8787

choose 30 GB of EBS General Purpose (SSD) or Magnetic storage

click Launch Instance

Use elastic IP (to allow server to be run "on-demand") * click on elastic IP in left panel * select associate Elastic IP 13.57.139.31 choose an instance (shiny-july22) to associate with.

Log into new instance with ssh from local

```
ssh -i ~/.ssh/aws18.pem ubuntu@13.57.139.31
```

or

```
ssh ec2
```

set up the environment on server

```
export HOST="134.122.40.112"
ssh root@$HOST

apt-get -y update
apt-get -y upgrade

apt-get install -y \
    docker \
    docker-compose

systemctl enable docker
# or use homebrew for linux
```

```
docker pull registry.gitlab.com/analythium/shinyproxy-hello/hello:latest
docker pull analythium/covidapp-shiny:minimal
```

if you've set up a `config` file in `~/.ssh` (see Tips at the end of the blog)

Go to godaddy or Amazon route 53 to associate a domain name with the Elastic IP in EC2.

To associate domain name rgtlab.org. 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 13.57.139.31

3) Install Docker on server. sudo apt update sudo apt install apt-transport-https ca-certificates curl software-properties-common curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo gpg –dearmor -o /usr/share/keyrings/docker-archive-keyring.gpg echo "deb [arch=$(dpkg –print-architecture) signed-by=/usr/share/keyrings/docker-archive-keyring.gpg] https://download.docker.com/linux/ubuntu $(lsb_release -cs) stable" | sudo tee /etc/apt/sources.list.d/docker.list > /dev/null sudo apt update apt-cache policy docker-ce sudo apt install docker-ce sudo systemctl status docker sudo docker run -d -p 80:80 –name test4 -p 443:443 -v ~/mywebsite:/usr/share/caddy/ -v ~/Caddyfile:/etc/caddy/Caddyfile -v caddy_data:/data caddy:2

to debug

sudo docker logs test4

cat Caddyfile (bob/utter credentials)

```
rgtlab.org {
    basicauth * {
        bob JDJhJDE0JEw4Zk5MQi5nYndEaHRjVTFVeGl2a2VBZTVPcFI0THhEZUd0OEJkOTR2d0VjRUxsMDBsQUg2
    }
    respond "It works!"
    root * /var/www/html
}
```

4)
5)

```
sudo apt install -y debian-keyring debian-archive-keyring apt-transport-https -y

curl -1sLf 'https://dl.cloudsmith.io/public/caddy/stable/gpg.key' | sudo gpg --dearmor -o /usr/share/

curl -1sLf 'https://dl.cloudsmith.io/public/caddy/stable/debian.deb.txt' | sudo tee /etc/apt/sources.
sudo apt-get update

sudo apt-get install caddy -y
sudo systemctl enable --now caddy
```

6. run container of image with parameters:

a. background: -d
b. name=appX (X between 1 and 9)
c. restart=always
d. p 400X:3838 (X between 1 and 9) -p 4001:3838 ec2> sudo docker run -d –name=app2 –restart=always -p 4002:3838 rgt47/app2:latest

7. edit /etc/caddy/Caddyfile to provide reverse proxy from port container ec2> cd etc/caddy ec2> sudo vim Caddyfile running on port 4002

handle_path /app2/* { reverse_proxy 0.0.0.0:4002 }

8. edit /var/www/html/index.html to provide link to the container name (app2) Call it "demo app" ec2> cd /var/www/html ec2> sudo vim index.html <!DOCTYPE html>

   rgthomas lab apps

   demo app

9. restart web server ec2> sudo systemctl restart caddy

10. Run the shiny app from the apps subdomain of rgthomaslab.com https://apps.rgthomaslab.com \end{verbatim}

11. Add zenn as a user and include in sudo list. all members of admin group have sudo privileges. copy private key from ubuntu user to zenn home directory and set correct permissions.

After Launch and login run the following to add Shiny and new user zenn

testing with Shiny3 might not need the following if ubuntu_startup.sh works

```
sudo R -e "install.packages('shiny', repos='http://cran.rstudio.com/')"
sudo apt-get install gdebi-core
wget https://download3.rstudio.org/ubuntu-14.04/x86_64/shiny-server-1.5.16.958-amd64.deb
sudo gdebi shiny-server-1.5.16.958-amd64.deb
# not sure about this example:
# sudo /opt/shiny-server/bin/deploy-example default
sudo adduser --home /home/zenn --shell /bin/bash\
    --ingroup admin --disabled-password zenn
# passwd zenn
sudo cp -R /home/ubuntu/.ssh /home/zenn/.ssh
sudo chown -R zenn:admin /home/zenn/.ssh
sudo chmod 600 /home/zenn/.ssh/*
sudo chmod 700 /home/zenn/.ssh
```

```
view app at
http://ec2-13.57.139.31.us-west-1.compute.amazonaws.com:3838/zenn/hello/
```

Security Set-up

1. Get domain name, e.g. from Godaddy.com. Mine is inferencex.com
2. get an elastic IP, reserved for your personal use. Needed for security certificate use, e.g. my elastic IP is 13.57.139.31

Add encryption to server

get lets encrypt certificate

```
sudo apt install apache2
sudo ufw enable
sudo ufw allow 'Apache Full'
sudo snap install core; sudo snap refresh core
sudo snap install --classic certbot
sudo ln -s /snap/bin/certbot /usr/bin/certbot
sudo certbot --apache

IMPORTANT NOTES:
 - Congratulations! Your certificate and chain have been saved at:
   /etc/letsencrypt/live/rgthomaslab.com/fullchain.pem
   Your key file has been saved at:
   /etc/letsencrypt/live/rgthomaslab.com/privkey.pem
   Your certificate will expire on 2021-04-15. To obtain a new or
   tweaked version of this certificate in the future, simply run
   certbot again with the "certonly" option. To non-interactively
   renew *all* of your certificates, run "certbot renew"
```

Modify apache configuration file to find add certificate

On digital ocean

on EC2 instance pull image from docker hub local> ssh ec2 ec2> sudo docker pull rgt47/app2:latest

Up next …

and move it to an inplace ec2 instance (https:apps.rgthomaslab.com) existing web server: caddy (ssl certificates) security model (ssh, https, shiny, rstudio) elastic IP docker installed

References

Tips

Build and run instance (call it four) open in shell for exploration

```
docker build -t four .
docker run -it --rm four /bin/bash
```

Construct a config file in ~/.ssh as:

```
Host ec2
    HostName 13.57.139.31
    User ubuntu
    Port 22
    IdentityFile ~/.ssh/aws18.pem
```

then you ssh into new server with

```
sh> ssh ec2
```

The Caddyfile and docker-compose.yml file are

Caddyfile

```
rgtlab.org {

    root * /srv
    handle_path /power0/* {
        reverse_proxy power0:3838
    }
    file_server
}
```

docker-compose.yml

```
version: "3.7"

services:
  power0:
    image: rgt47/power0:latest
  caddy:
    image: caddy:2.3.0-alpine
    ports:
      - "443:443"
    volumes:
      - $PWD/Caddyfile:/etc/caddy/Caddyfile
      - $PWD/site:/srv
      - caddy_data:/data
      - caddy_config:/config
volumes:
  caddy_data:
  caddy_config:
```

The index.html file

```
<!DOCTYPE html>
<html>
  <body>
      <a href="./power0/">Power0</a>
  </body>
</html>
```