Securely Deploying Your Shiny App Online: A Step-by-Step Guide

Ronald (Ryy) Glenn Thomas

2025-02-15

Table of contents

Photo by Nathan Waters on Unsplash

1 Overview

This guide demonstrates how to deploy a Shiny application from your local workstation to a secure web environment. We'll use a stack of open-source technologies including Linux, R, Shiny, Docker, and Caddy, deployed on AWS EC2. While we focus on AWS here, the principles apply to other cloud providers like Hetzner, which we'll cover in future posts.

2 Prerequisites

Before we begin, you'll need: - A working Shiny application - Basic familiarity with the command line - An AWS account (or another cloud provider)

3 The Example Application

Let's start with a simple but practical example: a power calculator for two-sample t-tests. While straightforward, this application demonstrates all the key deployment concepts.

```
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)
```

4 Deployment Checklist

To host our Shiny app securely online, we need to:

1. Obtain a static IP address
2. Register a domain name
3. Configure a firewall
4. Set up the virtual server
5. Install and configure a web server
6. Implement SSL encryption
7. Set up user authentication
8. Configure reverse proxy routing

While this might seem complex, we'll break it down into manageable steps.

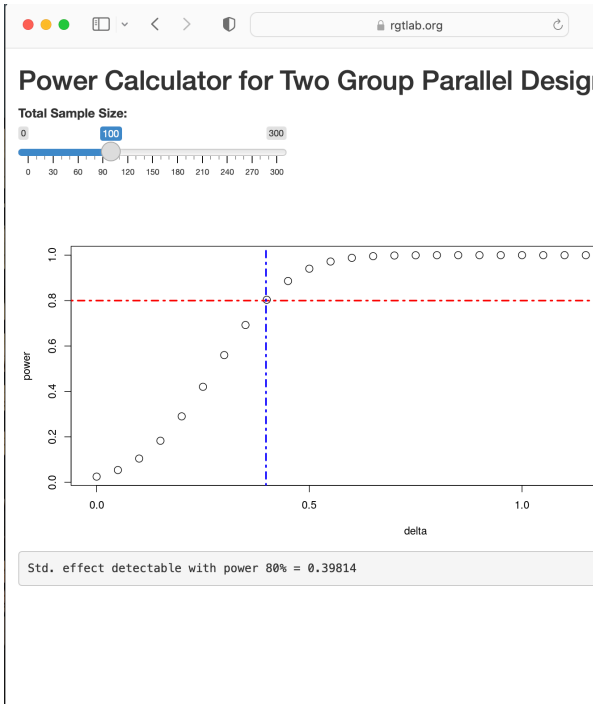

Figure 1: Shiny app interface

5 Step 1: Server Setup

First, we'll prepare our AWS EC2 environment:

1. Create or access your AWS account
2. Generate SSH key-pair
3. Configure firewall settings
4. Obtain static IP
5. Register domain name
6. Launch Ubuntu instance (t2-micro is sufficient)
7. Connect via SSH:

```
ssh -i "~/.ssh/power1_app.pem" ubuntu@rgtlab.org
```

6 Step 2: Installing Required Software

On your server, install Docker and Caddy:

```
sudo apt update
sudo apt install docker.io -y
sudo apt install -y curl debian-keyring debian-archive-keyring apt-transport-https
curl -1sLf 'https://dl.cloudsmith.io/public/caddy/stable/gpg.key' | \
sudo gpg --dearmor -o /usr/share/keyrings/caddy-stable-archive-keyring.gpg
curl -1sLf 'https://dl.cloudsmith.io/public/caddy/stable/debian.deb.txt' | \
sudo tee /etc/apt/sources.list.d/caddy-stable.list
sudo apt update
sudo apt install caddy -y
```

7 Step 3: Containerizing the Application

Create a `Dockerfile` in your app directory:

```
FROM rocker/shiny:4.2.0
RUN rm -rf /srv/shiny-server
COPY /power1_shiny/* /srv/shiny-server/
USER shiny
CMD ["/usr/bin/shiny-server"]
```

8 Step 4: Configuring the Web Server

Create a `Caddyfile`:

```
rgtlab.org {
    basicauth * /power1_shiny/* {
        bob $2a$14$pYWd5O7JqNeGLS4m4CKkzemM2pq5ezn9bcTDowofZTl5wRVl8NTJm
    }
    root * /var/www/html
    handle_path /power1_shiny/* {
            reverse_proxy 0.0.0.0:3838
    }
    file_server
}
```

Create an `index.html`:

```
<!DOCTYPE html>
<html>
  <body>
    <h1>Power1 app</h1>
    <ul>
      <li><a href="./power1_shiny/">Power1 app</a></li>
    </ul>
  </body>
</html>
```

9 Step 5: Deployment

1. Copy files to server:

```
scp -r ~/prj/power1_app/ ubuntu@rgtlab.org:~
```

2. Build and run Docker container:

```
docker build -t power1_image .
docker run -d --name=power1_shiny -p 3838:3838 --restart=always power1_image
```

3. Configure Caddy:

```
sudo cp ./Caddyfile /etc/caddy/
cp ./index.html /var/www/html/
sudo systemctl reload caddy
```

Your app should now be available at `https://rgtlab.org`!

10 Advanced Tips

For easier SSH access, create a `~/.ssh/config` file:

```
Host rgtlab.org
HostName 13.57.139.31
StrictHostKeyChecking no
User ubuntu
Port 22
IdentityFile ~/.ssh/power1_app.pem
```

This enables simple SSH access:

```
ssh rgtlab.org
```

11 Conclusion

You now have a secure, containerized Shiny application running on the web with SSL encryption and basic authentication. This setup provides a robust foundation for hosting data science applications, whether for personal use, collaboration, or client delivery.