Shiny + Docker

Svetlana Vinogradova

https://github.com/kintany

HARVARD MEDICAL SCHOOL

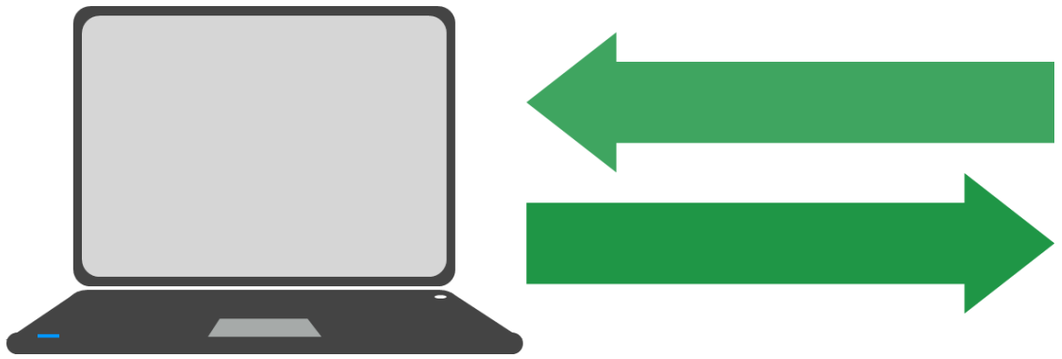
BROAD INSTITUTE

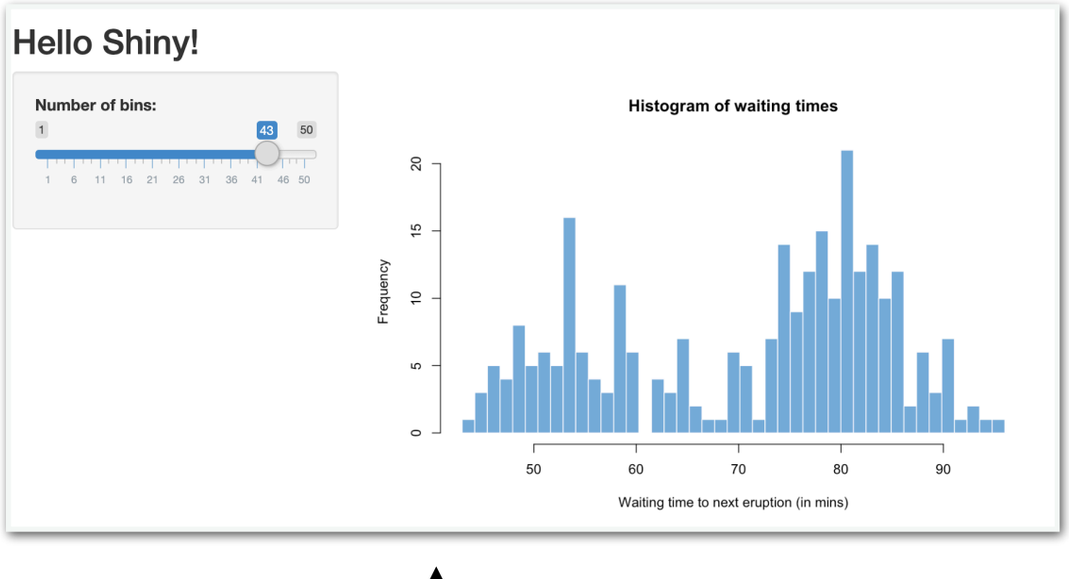
DANA-FARBER CANCER INSTITUTE

What is Shiny?

- R package from RStudio
- Web application framework for R
- R code → interactive web page
- No HTML/CSS/JavaScript knowledge required
- Great for sharing R analysis with someone scared of R

What is a Shiny App?





Computer running R

Web page

ui.R

ui.R × server.R ×

Run App

```
1  #
2  # This is the user-interface definition of a Shiny web appli
3  # run the application by clicking 'Run App' above.
4  #
5  # Find out more about building applications with Shiny here:
6  #
7  #    http://shiny.rstudio.com/
8  #
9
10 library(shiny)
11
12 # Define UI for application that draws a histogram
13 shinyUI(fluidPage(
14
15   # Application title
16   titlePanel("Old Faithful Geyser Data"),
17
18   # Sidebar with a slider input for number of bins
19   sidebarLayout(
20     sidebarPanel(
21        sliderInput("bins",
22                    "Number of bins:",
23                    min = 1,
24                    max = 50,
25                    value = 30)
26     ),
27
28     # Show a plot of the generated distribution
29     mainPanel(
30        plotOutput("distPlot")
31     )
32   )
33 ))
```

Title of your app

Side panel
(commonly used for parameters)

Main panel with results
and visualization

server.R

ui.R | server.R

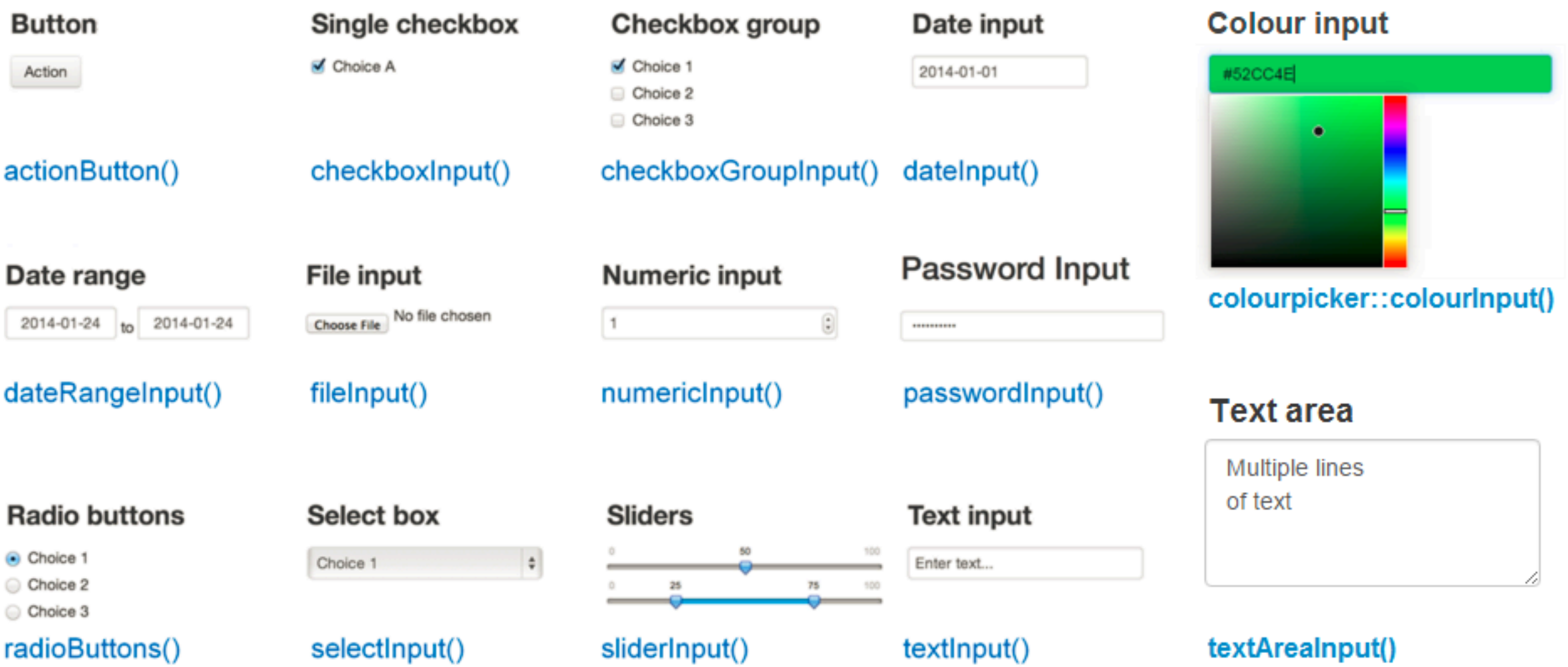
Run App

```
#
# This is the server logic of a Shiny web application. You can run the
# application by clicking 'Run App' above.
#
# Find out more about building applications with Shiny here:
#
#    http://shiny.rstudio.com/
#

library(shiny)

# Define server logic required to draw a histogram
shinyServer(function(input, output) {

  output$distPlot <- renderPlot({

    # generate bins based on input$bins from ui.R
    x    <- faithful[, 2]
    bins <- seq(min(x), max(x), length.out = input$bins + 1)

    # draw the histogram with the specified number of bins
    hist(x, breaks = bins, col = 'darkgray', border = 'white')

  })

})
```

One block of output

Add different types of inputs

Inputs are what gives users a way to interact with a Shiny app. Shiny provides many input functions to support many kinds of interactions that the user could have with an app. For example, `textInput()` is used to let the user enter text, `numericInput()` lets the user select a number, `dateInput()` is for selecting a date, `selectInput()` is for creating a select box (aka a dropdown menu).



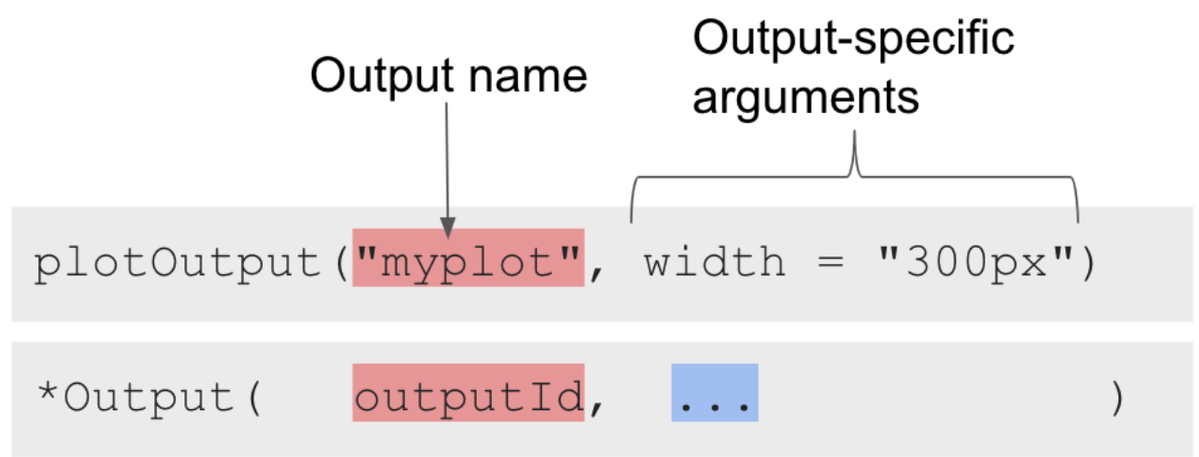
All input functions have the same first two arguments: `inputId` and `label`. The `inputId` will be the name that Shiny will use to refer to this input when you want to retrieve its current value so it should be unique.

Outputs

- Plots, tables, text - anything that R creates and users see

- Initialize as empty placeholder space until object is created

Function	Outputs
plotOutput()	plot
tableOutput()	table
uiOutput()	Shiny UI element
textOutput()	text



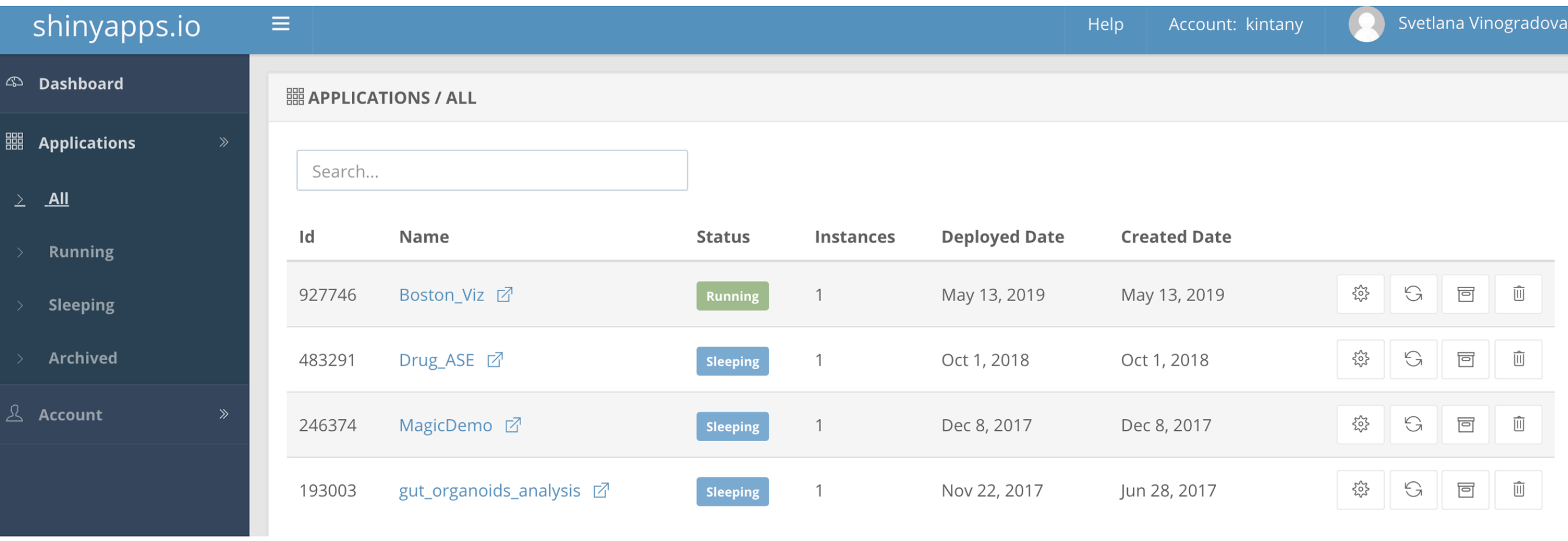
Publishing your Shiny App

Log in at https://www.shinyapps.io (you need to create an account first)

Go to Account > Tokens > Add Token > Show Secret > Copy

In R studio, button “Publish”, copy to the window > Publish

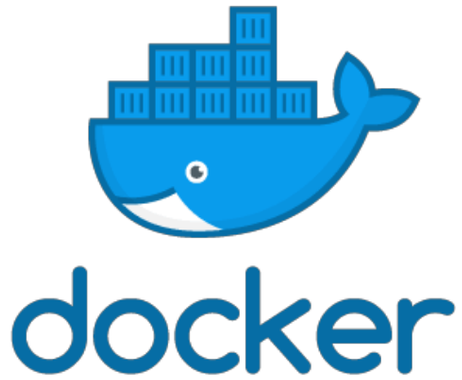
https://kintany.shinyapps.io/Boston_Viz/



Resources

- Shiny official tutorial http://shiny.rstudio.com/tutorial
- Shiny cheatsheet http://shiny.rstudio.com/images/shiny-cheatsheet.pdf
- Lots of short useful topics http://shiny.rstudio.com/articles
- Shiny in Rmarkdown http://rmarkdown.rstudio.com/authoring_shiny.html
- Get help from https://groups.google.com/forum/#!forum/shiny-discuss
  or http://stackoverflow.com/questions/tagged/shiny

Docker



Using Docker allows you to

- increase the performance of your app
- scale it,
- reduce its costs,
- make it reproducible,
- run it on wherever you want

Step 1: Install Docker

If you don't have Docker installed, visit docker.com and install the Community Edition. If you are not sure if you have Docker installed, try running:

```
docker --version
```

More help can be found here: https://docs.docker.com/get-started/ For example, try running a test image:

```
docker run hello-world
```

Step 2: Create a directory and copy your ShinyApp

Create a directory where you store all necessary files. In this folder, we will add our source code and further configuration files.

The file structure will look like the following:

```
Docker_ShinyApp
└── Boston_Viz
    ├── server.R
    ├── ui.R
    └── data
```

Step 3: Create a Dockerfile

The Dockerfile is the blueprint of any Docker image. It tells Docker what to install and in which order.

As a basis for the Dockerfile, I use a modified copy from the official Shiny-Docker image:

```
FROM r-base:latest

MAINTAINER Svetlana Vinogradova "kintany@gmail.com"

# Install dependencies and Download and install shiny server
RUN apt-get update && apt-get install -y -t unstable \
    sudo \
    gdebi-core \
    make \
    git \
    gcc \
    zlib1g-dev \
    libcurl4-gnutls-dev \
    libcairo2-dev/unstable \
    libxt-dev && \
    R -e "install.packages(c('shiny', 'rmarkdown', 'DT'), repos='https://cran.rstudio.com/')" && \
        R -e "install.packages(c('dplyr','magnittr','ggplot2','shinythemes', 'remotes','lubridate'))"  && \
      R -e "remotes::install_github('thomasp85/shinyFiles')" && \
    rm -rf /var/lib/apt/lists/*

EXPOSE 3838

COPY shiny-server.sh /usr/bin/shiny-server.sh
COPY Boston_Viz/ /srv/shiny-server/Boston_Viz/

CMD ["/usr/bin/shiny-server.sh"]
RUN ["chmod", "+x", "/usr/bin/shiny-server.sh"]
```

Step 4: Add configuration files for your ShinyApp

Download or create shiny-server.sh (*Adapted from* rocker-org)

```
Rscript -e ".libPaths('/usr/lib/R/library/'); shiny::runApp('/srv/shiny-server/Boston_Viz/', launch.browser =
F, port = 3838, host = '0.0.0.0')"
```

In the end, this directory should have the following structure:

```
Docker_ShinyApp
├── Dockerfile
├── shiny-server.sh
└── Boston_Viz
    ├── server.R
    ├── ui.R
    └── data
```

Step 5: Build the docker image

Building the Docker Image is straightforward. Make sure you are in the `Docker_ShinyApp` directory with the Terminal and type in:

```
docker build -t My_ShinyApp .
```

You can name the Docker Image whatever you like.

This process might take a couple of minutes as Docker will download the dedicated R and ShinyApp version, as well as all dependencies and packages.

Step 6: Publish your Docker

You can follow the instructions here: https://docs.docker.com/docker-hub/repos/

> To push a repository to the Docker Hub, you must name your local image using your Docker Hub username, and the repository name that you created through Docker Hub on the web.
>
> You can add multiple images to a repository, by adding a specific `:<tag>` to it (for example `docs/base:testing`). If it's not specified, the tag defaults to `latest`.
>
> You can name your local images either when you build it, using
>
> `docker build -t <hub-user>/<repo-name>[:<tag>]`, by re-tagging an existing local image
>
> `docker tag <existing-image> <hub-user>/<repo-name>[:<tag>]`, or by using
>
> `docker commit <existing-container> <hub-user>/<repo-name>[:<tag>]` to commit changes.
>
> Now you can push this repository to the registry designated by its name or tag.
>
> `$ docker push <hub-user>/<repo-name>:<tag>`
>
> The image is then uploaded and available for use by your teammates and/or the community.

Step 6: Run the docker image with your ShinyApp

When this process is done, you can run your ShinyApp in Docker 🎉

```
docker run --rm -p 3838:3838 My_ShinyApp
```

Now, you can open the app with any browser by visiting `http://localhost`