Shiny Server is a free and open-source option for self hosting Shiny apps and it is one of the 3 options listed on the Shiny website. In a previous post you saw how to secure Shiny Server with a custom domain. Here you will lear how to add and update Shiny apps to your server.

Follow the instructions from the previous post or spin up a brand new virtual machine on DigitalOcean using the RStudio 1-click app in minutes.

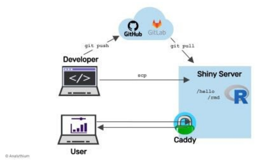
The `index.html` file for the Shiny Server's landing page and the `hello` and `rmd` apps are in the `/srv/shiny-server/` folder:

```
$ ls /srv/shiny-server/
index.html  sample-apps

$ ls /srv/shiny-server/sample-apps/hello
server.R  ui.R

$ ls /srv/shiny-server/sample-apps/rmd
index.Rmd
```

These directories map to the server path as:

- `http://$HOST` is the landing page (`index.html`),
- `http://$HOST/sample-apps/hello/` is the `hello` app,
- `http://$HOST/sample-apps/rmd/` is the `rmd` app.

`$HOST` is the custom domain (e.g. yourdomain.com) your Shiny Server server is using. You can see that the folder structure inside `/srv/shiny-server/` directly translates to the server paths. How do you add more apps to the server? Just copy the Shiny apps directly into folders within the `/srv/shiny-server/` directory. Here are three options for doing it.



File based deployment of Shiny apps to Shiny Server

Edit text files on the server

Let's add an app called `histogram` to the `http://$HOST/histogram/` path:

```
mkdir /srv/shiny-server/histogram
cd /srv/shiny-server/histogram
touch app.R
nano app.R
```

Copy-paste the Shiny app from below into `app.R` that you just opened with `nano` (Ctrl+O to save, Ctrl+X to exit `nano`):

```
library(shiny)
ui = fluidPage(
    mainPanel(
        sliderInput("obs",
            "Number of observations",
            min = 1,
            max = 5000,
            value = 100),
        plotOutput("distPlot")
    )
)
server = function(input, output) {
    output$distPlot = renderPlot({
        dist = rnorm(input$obs)
        hist(dist,
            col="purple",
            xlab="Random values")
    })
}
shinyApp(ui = ui, server = server)
```

Now if you visit `http://$HOST/histogram/` you'll see a range slider controlling sample size and a purple coloured histogram of a Normal distribution that must be familiar from a previous post.

Secure copy the files

You can use `scp` to copy local files to the server. Let's create a small Shiny app on your local machine. Use the same code as for the purple histogram, but change some of the settings:

```
mkdir pink
touch pink/app.R
nano pink/app.R
```

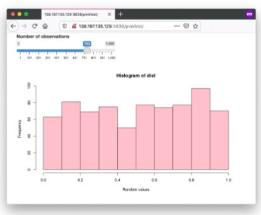
Copy this code into the file `pink/app.R`:

```
library(shiny)
ui = fluidPage(
    mainPanel(
        sliderInput("obs",
            "Number of observations",
            min = 1,
            max = 1000,
            value = 100),
        plotOutput("distPlot")
    )
)
server = function(input, output) {
    output$distPlot = renderPlot({
        dist = runif(input$obs)
        hist(dist,
            col="pink",
            xlab="Random values")
    })
}
shinyApp(ui = ui, server = server)
```

The following script copies the local `pink` directory over to the `pinkhist` directory of the Shiny Server (this use of `scp` assumes you access the server using your `ssh` key pair, otherwise you'll be prompted to provide your username/password):

```
export APPDIR="pink"
export SHINYDIR="pinkhist"

scp -r $APPDIR root@$HOST:/srv/shiny-server/$SHINYDIR
```

Visiting the `http://$HOST/pinkhist/` address should reveal the new app:



Note that the `scp` protocol uses port 22 similarly to `ssh` and `sftp`. If you are using the non-secure FTP protocol to copy the files, make sure port 21 of your server is open for incoming traffic.

Git based deployment

Set up `git` so you can work with your public and private repositories (`git` is already installed, this is a one-time setup for each user):

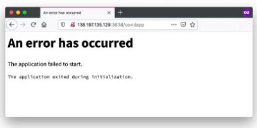
```
git config --global user.email "you@example.com"
git config --global user.name "Your Name"
```

Change to your home directory with `cd ~` and run the following script. This will clone the GitHub repository specified in the environment variables, or pull changes if the directory already exists; then makes a directory for Shiny Server if that directory is not already there, then copies the app files (`$APPDIR` can be empty):

```
#!/bin/bash

export GITHOST="https://github.com"
export GITUSER="analythium"
export GITREPO="covidapp-shiny"
export APPDIR="02-shiny-app/app"
export SHINYDIR="covidapp"

# clone or pull repo
if [[ ! -e $GITREPO ]]; then
    git clone $GITHOST/$GITUSER/$GITREPO.git
else
    cd $GITREPO
    git pull
    cd ..
fi

# make dir if not already there
mkdir -p /srv/shiny-server/$SHINYDIR

# copy repo contents to Shiny Server
cp -rf $GITREPO/$APPDIR/* /srv/shiny-server/$SHINYDIR
```

You can modify this script for other repositories and save in a file, e.g. `update_covid.sh` . Then you can run `bash update_covid.sh` every time you need the app to be updated. It is also possible to set up a cron job to update the app daily or tie the script to a webhook event triggered by a successful GitHub action.

But before you do any of that, check if the app is running fine at `http://$HOST/covidapp/`. If all went wrong, you should see this message:



An error has occurred

The application failed to start.

The application exited during initialization.

This happened because the Shiny Server setup only included the most basic R packages and we missed the forecast package required by the COVID-19 app. Let's remedy this:

```
R -q -e "install.packages('forecast')"
```

If you refresh the page now the app should work fine.

This immediately highlights one of the shortcomings of Shiny Server when it comes to continuous integration and delivery (CICD). You need to be extra careful when managing packages, R and package versions, etc. when you have multiple apps with possibly conflicting dependencies. Dockerizing Shiny apps is one solution to isolate your applications.

> Dockerized Shiny Apps with Dependencies
> The wealth of contributed R packages can supercharge Shiny app development. This also means that you have to manage these dependencies. Learn about dependency management when working with R and Docker.
>
> Hosting Data AppsPeter Solymos

Coronavirus COVID-19 Global Cases by the Center for Systems
Total Confirmed
1,002,159
Confirmed Cases by Country/Region/Sovereignty
236,339 US
Italy

License

R and the shiny R package are both licensed under the GNU General Public License that permits making a modified version and letting the public access it on a server without ever releasing its source code to the public. However, it is important to note that the open source version of Shiny Server is licensed under the GNU Affero General Public License (AGPLv3).

AGPL closes the application service provider (ASP) loophole and has a controversial reputation in tech circles, see for example Google's policy on banning the license. We are not providing legal advice on this matter here, merely stating what the license says:

> The GNU General Public License permits making a modified version and letting the public access it on a server without ever releasing its source code to the public. The GNU Affero General Public License is designed specifically to ensure that, in such cases, the modified source code becomes available to the community. It requires the operator of a network server to provide the source code of the modified version running there to the users of that server. Therefore, public use of a modified version, on a publicly accessible server, gives the public access to the source code of the modified version. – AGPLv3 Preamble

The interpretation of modified source code and public accessibility on a server leaves a lot of room for interpretation (see some discussions here) which might or might not bother you depending on your intentions and corporate environment. RStudio offers paid exception to the AGPL license in the form of RStudio Connect – this model called dual-licensing (read about it here).