# Shiny Server Guide

**Executive Summary**

The intention of this guide is to walk the readers (Let’s be honest, the only reader is probably myself) through the detailed steps of deploying your shiny app on your own server as well as identifying the outstanding issues yet to be resolved.

If you are looking for a guide like this online, I am guessing you are looking for some serious s\*t, and if you are, you are in the right place. Here are what I assume that you would want to achieve in the end:

1. Host your own website and/or shiny apps using your own server (fancy word for PC, sometimes)
2. Access your app from internal network, if you are building an app for yourself or your family or your co-workers
3. If your app can be accessed within internal network, why not take one-step further and allow access from anywhere, right?
4. Wait a minute! If anybody can access your app from the web, perhaps you might want to spend some time on the security side of the business?

All right, all right, all right (To honor Matthew McConaughey), if you are only looking to do the first two things on the list, the process is quite simple. Assuming you have a fresh install of Ubuntu 16.04 LTS (Sorry windows users, you can stop reading now), you just need to install R, Shiny package and Shiny-server on your PC and allow traffic to port 3838 using Uncomplicated firewall (UFW). You are all set to go and don’t you worry, even though it is easy, I will still provide instructions on how to do that. It’s the last two items that require the most of your time and energy. That’s where the s\*t hits the fan. I will try to provide as detailed instructions as possible, but to be honest, a lot of the topics deserve a separate guide. In addition, I am actually learning on the go, so don’t think what I did was the golden principle. Meanwhile, if you don’t like what you read, write one yourself and stop bitching.

**Background**

My idea started with me building a trading portal for myself to use. It was so much pain logging in your broker account and fill out the trade form every time you want to make a $1,000 trade, so I decided to utilize the API my broker provided and wrote some codes to execute the trade. But, the problem is that you can only do it with your PC at home, say you are in the airport and just want to make a quick trade, what would you do? This gets me to think that I might need a server of my own to respond to my trade requests. Since I wrote my codes in R, shiny and shiny-server became a no-brainer.

You can find numerous great guides online. For instance, Dean Attali from DigitalOcean wrote an excellent guide on how to setup your shiny server (See reference). While I did follow his guide in the first place, I later realized that it’s not exactly I wanted. One example, he utilized a private server service ($5/month) from DigitalOcean to build the server, but I wanted to use my own domain name (free). On top of that, most guides don’t give you the rationale for why they did certain steps. This really becomes annoying when you want to customize certain things. Without knowing the why, you can’t do too much really. Plus, I don’t like blindly following someone else’s guide without really knowing if I should. These little details really drove me to write this guide.

While I was doing my researches, one couple of things I did find quite interesting were the NGINX web service and Uncomplicated Firewall (UFW). A lot of you guys probably heard about these terms before, but not to me. I did use Apache web service quite often with my previous job, but quite franking, I wasn’t sure what I was doing. I ended up spending a lot of time reading the documentations, so even though I am sure that I am still scratching the surface, (I think) I am in a much better position than I was in earlier. I would definitely recommend you to read the documentations if you want to fully unleash the power of NGINX. By the way, you can also use Apache to achieve the same result.

**Basic Instructions**

1. Setup your server. Clean up your existing Linux or install a fresh one. When I was young, I was installing a system pretty much every day. Now, I am more at a once-every-decade speed. That being said, I still installed a fresh Lubuntu 17.10. I would go with Ubuntu 16.04 LTS if I could, but my PC is really old, I thought a lightweight system might be better off.
2. Install R on your linux. Obviously, you need a working R for Shiny to run. RStudio is optional, but I don’t think I can live without RStudio at this point.
   1. I choose to install Microsoft R Open (https://mran.microsoft.com/download)
      1. Helper code: Wget “https://mran.blob.core.windows.net/install/mro/3.5.1/microsoft-r-open-3.5.1.tar.gz”
   2. Add R repository to Ubuntu sources.list using the command:
      1. Sudo echo "deb http://cran.rstudio.com/bin/linux/ubuntu xenial/" | sudo tee -a /etc/apt/sources.list
      2. This essentially connects you to CRAN, so the R you installed will usually be up-to-date.
   3. Add R to Ubuntu Keyring using the command:
      1. gpg --keyserver [keyserver.ubuntu.com](http://keyserver.ubuntu.com/) --recv-key E084DAB9
      2. gpg -a --export E084DAB9 | sudo apt-key add –

Keyring is essentially a database where login information are stored.

* 1. Install r-base and r-base-dev
     1. sudo apt-get update
     2. sudo apt-get install r-base r-base-dev
  2. Install rstudio.
     1. sudo apt-get install gdebi-core
     2. wget https://download1.rstudio.org/rstudio-xenial-1.1.456-amd64.deb
     3. sudo gdebi -n rstudio-xenial-1.1.456-amd64.deb
     4. rm rstudio-xenial-1.1.456-amd64.deb

1. Install Shiny package and Shiny server. I think you know why we do this step, otherwise, you wouldn’t be here, would you?
   1. Install R Shiny package:
      1. sudo su - -c "R -e \"install.packages('shiny', repos='http://cran.rstudio.com/')\""
   2. Install Shiny-Server.
      1. wget <https://download3.rstudio.org/ubuntu-14.04/x86_64/shiny-server-1.5.8.921-amd64.deb>
      2. sudo gdebi shiny-server-1.5.8.921-amd64.deb
      3. rm shiny-server-1.5.8.921-amd64.deb
2. This is where you can check if Shiny-Server is installed correctly. On your web browser, type localhost:3838 and see if you can see a page with the title “Welcome to Shiny Server” and a sample web app with the title “It’s Alive!” on the right hand side of the webpage. If you do, Shiny-server is good to go on your machine.
3. Now, you are pretty much done if you only want to setup Shiny within internal network, such as your office or your home. To test if this is true, you need to do the following steps:
   1. Find out the internal IP of your server. In terminal, enter ifconfig and remember the IP address, say the IP is 192.168.1.1.
   2. Punch your co-worker in the face and then take his laptop
   3. On his laptop, type 192.168.1.1:3838 on your web browser, see if you can see the same page you saw on your server.
   4. I bet you can’t. If you can, I will be damned. Guess what is missing?
4. If anyone within your internal network can access your PC without any configuration, you would be seriously concerned, right? You need to somehow allow others to connect your PC. This is why the server is called a server, not a PC. It should allow some traffic through. To do this, you need UFW. (A great guide for UFW is in reference.)
   1. Install UFW using the command:
      1. sudo apt-get update
      2. sudo apt-get install ufw
   2. The default settings of UFW is this:
      1. Deny all requests to your server
      2. Allow all requests from your server
   3. For someone in your group to access your Shiny app, you are going to allow some requests, right? It turns out for your server to respond to these requests, you need to allow port 3838 to accept requests from outside. Does 3838 sound familiar? Remember when you tested if Shiny-Server is installed correctly, you went to localhost:3838 to check? This is NOT a coincidence. By default, Shiny-server is setup so that it only listens to requests sent through port 3838. This is why you need to open port 3838 on your server. Use this command
      1. sudo ufw allow 3838 (Open port)
      2. sudo ufw status (Check if 3838 is actually opened)
   4. Now, let’s pause for a second to think about what a port is. I tend to think of a port as a door of your house. If you only have one door, you can just refer it as “the door”. But if you have a mansion, you probably have ten doors, now how would you refer to each door? Naturally, you would start to assign numeric numbers to these doors and when people come to visit, you just tell them the numeric number and they know when to go in and/or out, right? However, if people keep asking you which door to use, you might get annoyed. What can you do? You start not to tell people about all the doors. Instead, just tell them you have one door and that door is called “Door1”. Sooner or later, people will forget that you have so many doors and start to use “Door1”. Eventually, they will even forget what “Door1” is and start to refer to it as “Door”. This is really what ports are. Just as you know, “Door1” is port 80. This is the default door/port for pretty much everything. Now, a small test, go to your web browser and enter “google.com:80”, this will take you back to “google.com”. But if you type “google.com:3838”, you will get an error. Why? This is because door/port 3838 is not opened on Google’s server.
5. You are really done here. If you type “192.168.1.1:3838” on your co-worker’s laptop, you should see the same page you saw earlier, da-da.
6. If you have a working Shiny app already, you can move it to the following folder:
   1. /srv/shiny-server (This is the default location to host your apps)
   2. If you create a folder under this folder and name it “MyyyShinyyy”, you and anyone within your internal network will be able to access this ShinyApp through 192.168.1.1:3838/MyyyShinyyy/
   3. This can host a lot of ShinyApps. I don’t know about the limit, sorry, guys.

**Advanced Instructions**

1. If you are like me, you wouldn’t stop at the basic instructions. I had so many questions: Why do we have to use this weird port 3838, not 1000, something that’s easy to remember? What if I want to put my ShinyApps somewhere not /srv/shiny-server? What if I already have a personal website and want to integrate Shiny with my existing website? And most importantly, what if I want to access my ShinyApp from all around the world? I shall provide some advanced instructions here.
2. The first two questions require less time to answer. They are essentially parameters, and parameters, by its definition, are meant to be modified.
   1. The entire Shiny app is controlled by /etc/shiny-server/shiny-server.conf. This .conf file looks like this:

|  |
| --- |
| # Instruct Shiny Server to run applications as the user "shiny"  run\_as shiny; |
|  |
| # Define a server that listens on port 3838 |
| server { |
| listen 3838; |
|  |
| # Define a location at the base URL |
| location / { |
|  |
| # Host the directory of Shiny Apps stored in this directory |
| site\_dir /srv/shiny-server; |
|  |
| # Log all Shiny output to files in this directory |
| log\_dir /var/log/shiny-server; |
|  |
| # When a user visits the base URL rather than a particular application, |
| # an index of the applications available in this directory will be shown. |
| directory\_index on; |
| } |
| } |

* 1. I think you probably figured this out already. Shiny is a server who listens on port 3838, if you change the port to something, it will still work most of the time.
  2. You might also notice the line: site\_dir /srv/shiny-server. Right, if you change this location to the directory you want, you can run ShinyApps there.
  3. Not too bad, huh! A few more details are discussed below.
  4. Note the line listen 3838 doesn’t have a host name? It defaults to localhost, in other words, listen 3838; is equivalent to listen localhost:3838;.
  5. Let’s look at the line location / { … }:
     + This is how Shiny handles request. When Shiny hears a signal like localhost:3838/, it starts to act. It appends “/” to site\_dir, which is “/srv/shiny-server” and sends whatever in this folder to the signal sender.
     + Now, say I change this block of code to location /MyShiny/ { … }, when you type localhost:3838/ again, it won’t work anymore since our new location block only listens to localhost:3838/MyShiny/.
  6. Hopefully by now you should have a basic understanding of why ShinyApp works. This is actually not only true for Shiny, the same logic applies for Apache and Nginx, which we will talk about later
  7. Enough of theory. Here are how to do it in practice:
     + First, backup shiny-server.conf with the command:

sudo cp /etc/shiny-server/shiny-server.conf /etc/shiny-server/shiny-server.conf.backup

* + - Then, use gedit or nano to edit this file:

sudo gedit /etc/shiny-server/shiny-server.conf (make changes as you feel appropriate)

* + - Restart Shiny server. Whenever you change something in shiny-server.conf, you need to restart Shiny server. It can be done via:

sudo systemctl restart shiny-server

* + - If the changes you made are not valid, you will get an error when you restarted the server. This is the time you go back to check if your changes make sense.
  1. I highly recommend you to read the Administrator’s Guide, if you want to fully control Shiny.

1. User control. Notice the first line in shiny-server.conf, it reads as run\_as shiny;. What if I don’t want to run as Shiny, and I want to run it as myself. You can do this with a special syntax:

|  |
| --- |
| run\_as :HOME\_USER: |
| … |
| Location /users { |
| User\_dirs;  member\_of shinyusers; |
| } |

1. Assuming you have a user called APPLE and you have two projects called APP1, what this code essentially does is that when you type 192.168.1.1:3838/APPLE/APP1, the server will respond with whatever in /home/APPLE/ShinyApps/APP1 folder. The benefits of this is that when you have multiple users on your PC, they can each have their own ShinyApps. You can also assign different levels of security setting to them.
2. The Line member\_of shinyusers; allows only the user in group shinyusers to have their shiny app assessable.
3. Access your ShinyApp from anywhere from the world. You will need some basic network knowledge to be able to pull this off. I have just enough knowledge to get it to work, but I am not a expect either. A few concepts you need to know:
   1. Public IP and private IP. Public IP is the virtual address that is assigned to you by you ISP carrier (AT&T, Bell Aliant). It is just like your home address, but different. You ISP could potentially assign different public IP to your network, but usually, you wouldn’t know and you wouldn’t care as long as you can watch free porn, right? But, when you want to access your home network from outside, you need to know exactly what the address is. If the public IP changes, then you connect request will be rejected if you send it to the old IP. To fix this, you need something that is always pointing to the most recent public IP. This is where dynamic DNS comes into play. I use the combination of Bell Aliant Fibre Op and no-ip.com and it works great for me. Basically, no-ip.com assigns you a domain name and you assign that domain name to the dynamic DNS part of your router. Then, you don’t really need to track your public IP anymore. As long as you use the domain name given by no-ip.com, you are always up-to-date.
   2. But, this is not it. Even though your public IP is fixed, your private IP is not. Now let’s talk about private IP. Think of private IP as rooms in your home. When mails are sent to your house, your parents pick up your mail and wants to send it to you, they would need to delivery it to your room. You room is then the private IP. It is private in a sense that only your parents know it, but the mailman doesn’t. Just like you can switch rooms, private IP can change from time to time. Well, if you keep the server running and never shuts it down, then it sure will not change. But to make sure every time when your server is running, it has the same private IP, you need something else, this is called Static (private) IP. Static means it’s static, not changing, make sense?
   3. When you think you are done, no no no, you are not. Just because both public IP and private IP are in place now, it doesn’t mean they know how to talk to each other. For them to start talking, you need another weapon, and that, my friend, is called port forwarding. Yes, port comes into play again. In your router, you can setup so that when requests are sent to Public IP:port, it will directed to a destination you specified.
   4. As an example, if your public IP is 101.102.103.104 and your server (private) IP is again 192.168.1.1. Then you need to setup so that the requests sent to 101.102.103.104:3838 will be internally sent to 192.168.1.1. This way, when your request from outside comes in, your public IP will be smart enough to pass the request to your sever at 192.168.1.1:3838.
   5. Now, we are done.
4. If you have already came this far, you might as well just keep reading.
5. How do you integrate your existing website with Shiny? And since you can access your Shiny app from outside, what if someone else accidentally accessed your Shiny app, this is not good, right? It turns out both questions can be answered by using a web service. I used Nginx (See reference) and I think it is great.
   1. Installing nginx is rather easy:
      * sudo apt-get update
      * sudo apt-get install nginx
   2. Then, you will have allow traffic for nginx on your server:
      * sudo ufw allow 'Nginx HTTP'
      * sudo ufw status (Check if nginx is allowed by UFW)
   3. Some common commands to manage nginx
      * sudo systemctl stop nginx
      * sudo systemctl start nginx
      * sudo systemctl restart nginx
      * sudo nginx –s reload (Every time you change the main .conf file, you would need to run this command)
   4. Now, let’s talk about the structure of Nginx:
   5. Just like Shiny, there is main configuration file at /etc/nginx/nginx.conf. However, you are not recommended to modify this file. Rather, you should create a customized .conf file under the directory /etc/nginx/conf.d. Say, you create this file named apple.conf and it should look like this:

|  |
| --- |
| server { |
| listen 1000 default\_server; |
| location /shiny/ { |
| rewrite ^/shiny/(.\*)$ /$1 break; |
| proxy\_pass http://localhost:3838; |
| proxy\_redirect <http://localhost:3838/> $scheme://$host/shiny/; |
| proxy\_http\_version 1.1; |
| proxy\_set\_header Upgrade $http\_upgrade; |
| proxy\_set\_header Connection “upgrade”; |
| proxy\_read\_timeout 20d; |
| proxy\_buffering off; |
| } |

* 1. Note that port 1000 and location /shiny/ is really randomly picked. Feel free to change it to whatever you want.
  2. Note the line proxy\_pass <http://localhost:3838>;. This is the line that redirects all income messages from port 1000 to Shiny-Server. This way, there is a layer between outside crazy world and you Shiny-Server.
  3. You can customize this .conf in various ways, for instance,
     + You can add username and password for better security
     + You can add another location block to handle your personal website traffic.
  4. As a note, there are a lot of documentations for Nginx, I didn’t read all of them. I will keep chewing away though.

1. Basic security. Username and password authorization can be added using nginx.
   1. Install a utility software using command:

sudo apt-get install apache2-utils

* 1. Create a password file and add a user

sudo htpasswd –c /etc/nginx/.htpasswd /\*user\_name\*/ and enter a password

* 1. Additional user can be added as

sudo htpasswd /etc/nginx/.htpasswd /\*user\_name\*/

* 1. Use this password file for your website or shiny app

auth\_basic “Restricted Content”;

auth\_basic\_user\_file /etc/nginx/.htpasswd;

1. Now, I am pretty much done here. A few outstanding notes:
   1. You can enhance the security of your ShinyApp using Amazon AWS or Microsoft Azure. I am not particularly interested in this at this point, but I might be in future.
2. Check out appendixes for some additional information.
3. All right. That’s it for the day.

**Conclusion**

All right amigos. I am out!

Author: Ke Min, FSA, ACIA, CERA

Date: 2018-03-09

**Appendix A**

Given that my IP address is 100.100.10.1, my username is KMIN and my app name is KminShinyApp. My final setup of my shiny app is as follows:

Main page: <http://100.100.10.1:1000> (/home/kmin/mainserver/index.html)

Corporate shiny app: <http://100.100.10.1:1000/shiny/corp/KminShinyApp/> (/srv/shiny-server/KminShinyApp)

User shiny app: <http://100.100.10.1:1000/shiny/users/kmin/KminShinyApp/> (/home/kmin/ShinyApps/KminShinyApp) (Only users in group shinyusers can have shiny apps)

**Appendix B**

Common Linux commands related to shiny app user control:

* Switch to root mode: sudo –i
* Exit root mode: exit
* Add user: useradd /\*user\_name\*/
* Add group: groupadd /\*group\_name\*/
* Add user to a grup: usermod –a –G /\*group\_name\*/ /\*user\_name\*/
* Check user information: id /\*user\_name\*/
* Switch user: su /\*user\_name\*/

**References**

(Dean Attali) <https://deanattali.com/2015/05/09/setup-rstudio-shiny-server-digital-ocean/>

(Lubuntu) <https://lubuntu.net/>

(UFW) <https://www.digitalocean.com/community/tutorials/how-to-set-up-a-firewall-with-ufw-on-ubuntu-16-04>

(Shiny Administrator’s Guide) <http://docs.rstudio.com/shiny-server/#systemd-redhat-7-ubuntu-15.04-sles-12>

(Nginx) <https://www.nginx.com/>