How to Set Up Shiny Server on Ubuntu 16.04

Posted October 25, 2017 29.7k APPLICATIONS R UBUNTU 16.04

By: finid

An older version of this article was written by Dean Attali.

Introduction

While many people primarily turn to the open-source programming language R for statistical and graphics applications, Shiny is an R package that allows you to convert your R code into interactive webpages. And when combined with Shiny Server — available in both a free, open-source and a paid, professional format — you can also host and manage Shiny applications and interactive R markdown documents.

In this tutorial, you'll install and configure Shiny and the open-source version of Shiny Server on a server running Ubuntu 16.04, secure the connection to the Shiny server using a Let's Encrypt SSL certificate, and then install an additional package to run interactive R Markdown documents.

Prerequisites

To complete this tutorial, you'll need the following:

- One Ubuntu 16.04 server with **a minimum of 1GB of RAM** set up by following this Ubuntu 16.04 initial server setup tutorial, including a sudo non-root user and a firewall.

> **Warning:** Anything less than 1GB of RAM on your server may cause the installation of Shiny Server or its related R packages to fail.

- The latest version of R installed by following Step 1 in this installing R on Ubuntu 16.04 tutorial.
- Nginx installed by following this How To Install Nginx on Ubuntu 16.04 tutorial, including allowing access to ports `80` and `443` in Step 2 with the command `sudo ufw allow 'Nginx Full'`.
- A fully registered domain name. This tutorial will use `example.com` throughout. You can purchase a domain name on Namecheap, get one for free on Freenom, or use the domain registrar of your choice.
- Both of the following DNS records set up for your server. You can follow this hostname tutorial for details on how to add them.
  - An A record with `example.com` pointing to your server's public IP address.
  - An A record with `www.example.com` pointing to your server's public IP address.
- A Let's Encrypt SSL certificate for the domain installed by following this Let's Encrypt Certbot tutorial.

Once all of the prerequisites are in place, we'll start by installing Shiny on the server.

Step 1 — Installing Shiny

Before installing Shiny Server, you'll need to install the Shiny R package which provides the framework that Shiny web applications run on.

If you're familiar with R, you might be tempted to install packages directly from R instead of from the command line. But, using the following command is the safest way to ensure that the package gets installed for all users and not just for the user currently running R.

`c` option specifies the command that will be run.

The `install.packages` is the R command used to install R packages. So, in this command specifically, the `shiny` package is installed from the specified repository.

```
$ sudo su - -c "R -e \"install.packages('shiny', repos='http://cran.rstudio.com/')\""
```

When complete, R will tell you that the installation is `DONE` and where it put the downloaded source packages:

```
Output
$ ...
$ * DONE (shiny)
$
$ The downloaded source packages are in
$     ‘/tmp/Rtmp2GcWv4/downloaded_packages’
```

With Shiny in place, you're now ready to install Shiny Server and bring up its default welcome screen in your browser.

Step 2 — Installing Shiny Server

In this step, you'll install Shiny server and tweak the firewall to allow traffic through the port that Shiny Server listens on.

Per Shiny Server's official installation instructions, we'll use `wget` to download a pre-built binary for 64-bit architecture. Because Shiny Server is in active development, you should consult the official Shiny Server download page to get the URL for the latest 64bit, pre-built binary matching your operating system. Once you have the address, change the URL in the following command accordingly.

```
$ wget https://download3.rstudio.org/ubuntu-12.04/x86_64/shiny-server-1.5.5.872-amd64.deb
```

Once the file is downloaded, verify its integrity by comparing the output of the following command with the MD5 checksum listed on the RStudio Shiny Server download page at the top of 64bit, pre-built binary download instructions.

```
$ md5sum shiny-server-1.5.5.872-amd64.deb
```

If the checksums don't match, re-download the file and try to verify its integrity again before moving on.

Because Shiny Server depends on GDebi — a tool that installs local deb packages while simultaneously resolving and installing additional dependencies — for its installation, you'll need to update your package list and then install the `gdebi-core` package next.

```
$ sudo apt-get update
$ sudo apt-get install gdebi-core
```

You're now ready to install Shiny Server.

```
$ sudo gdebi shiny-server-1.5.5.872-amd64.deb
```

Type `y` when GDebi asks you to confirm that you want to install the package.

```
[Secondary_label Output]
Shiny Server
 Shiny Server is a server program from RStudio, Inc. that makes Shiny applications available over the web. Shiny is a web app
Do you want to install the software package? [y/N]:y
```

At this point, the output should indicate that a service named `ShinyServer` is both installed and an `active` Systemd service. If the output indicates that there's a problem, re-trace your previous steps before continuing.

```
[Secondary_label Output]
...
● shiny-server.service - ShinyServer
                                                    reset: enabled)
```

```
   Active: active (running) since Fri 2017-10-13 14:24:28 UTC; 2 days ago
...
```

Next, verify that Shiny Server is indeed listening on port `3838`.

```
$ sudo netstat -plunt | grep -i shiny
```

If successful, the output will include the following line:

```
Output
tcp        0      0 0.0.0.0:3838            0.0.0.0:*               LISTEN      18749/shiny-server
```

If your output doesn't look like this, double-check your terminal for additional warnings and error messages.

Now, modify the firewall to allow traffic through to Shiny Server.

```
$ sudo ufw allow 3838
```

Finally, point your browser to `http://www.example.com:3838` to bring up the default Shiny Server homepage, welcoming you to Shiny Server and congratulating you on your installation.

> **NOTE:** You may see a small box on the right-hand side of the screen with a message saying, **An error has occurred**. Once you install `rmarkdown` in Step 4, the error message will be replaced with an interactive Shiny Doc.

You now have both Shiny and Shiny Server installed and tested, so let's secure the setup by configuring Nginx to serve as a reverse proxy and route all traffic over HTTPS.

Step 3 — Securing Shiny Server with a Reverse Proxy and SSL Certificate

In this step, you'll configure Nginx to forward incoming requests to Shiny Server by way of WebSocket, a protocol for messaging between web servers and clients.

Because we want to create configuration variables that any Nginx server block can use, open the main Nginx configuration file, `nginx.conf`, for editing.

```
$ sudo nano /etc/nginx/nginx.conf
```

Using Nginx's map module, create variables for the values that WebSocket needs by copying the following directive into the `http` block:

/etc/nginx/nginx.conf

```
http {
    ...
    # Map proxy settings for RStudio
    map $http_upgrade $connection_upgrade {
        default upgrade;
        '' close;
    }
}
```

The `map` directive compares `$http_upgrade` — the value of the client's **Upgrade** header — to the conditions in the curly braces. If the value is `''`, `map` creates the `$connection_upgrade` variable and sets it to `close`. Otherwise, `map` creates the `$connection_upgrade` variable and sets it to the default value, `upgrade`.

Save your work and close the file to continue.

Next, create a completely new Nginx server block, so that you still have the default configuration file to revert back to if you run into a problem later.

Create a new set of directives for Shiny Server, by copying and pasting the following into the new file:

example.com'>/etc/nginx/sites-available/example.com

```
server {
   listen 80 default_server;
   listen [::]:80 default_server ipv6only=on;
   server_name example.com www.example.com;
   return 301 https://$server_name$request_uri;

}
server {
   listen 443 ssl;
   server_name example.com www.example.com;
   ssl_certificate /etc/letsencrypt/live/example.com/fullchain.pem;
   ssl_certificate_key /etc/letsencrypt/live/example.com/privkey.pem;
   ssl_protocols TLSv1 TLSv1.1 TLSv1.2;
   ssl_prefer_server_ciphers on;
   ssl_ciphers AES256+EECDH:AES256+EDH:!aNULL;

   location / {
       proxy_pass http://your_server_ip:3838;
       proxy_redirect http://your_server_ip:3838/ https://$host/;
       proxy_http_version 1.1;
       proxy_set_header Upgrade $http_upgrade;
       proxy_set_header Connection $connection_upgrade;
       proxy_read_timeout 20d;
   }
}
```

The net effect of this configuration is that all incoming requests to the server on ports `80` and `3838` are redirected to use HTTPS on port `443`.

An overview of some of the more complex aspects of this configuration includes:

- **return**: Creates a permanent redirect for requests coming in as plain HTTP to HTTPS.
- **proxy_pass**: Tells Nginx to forward requests coming in at the root of the web server application to the IP address of the server listening on port `3838`.
- **proxy_redirect**: Rewrites the incoming string, `http://your_server_ip:3838/`, to its HTTPS equivalent on the server processing the request. The `$host` variable evaluates to the hostname of the server Nginx is running on.
- **proxy_set_header**: Redefines or appends fields to the request header passed to the proxied server.
- **proxy_read_timeout**: Sets a timeout for reading a response from the proxied server between two successive read operations.

Save and close the file to continue.

Next, enable the new server block by creating a symbolic link for it in the `/etc/nginx/sites-enabled` directory.

```
$ sudo ln -s /etc/nginx/sites-available/example.com /etc/nginx/sites-enabled/example.com
```

And, because our new server block now handles all requests on port `80`, you can disable the default block by deleting the symbolic link to it in the `sites-enabled` directory.

```
$ sudo rm -f /etc/nginx/sites-enabled/default
```

Now, test your new configuration before activating the changes.

```
$ sudo nginx -t
```

If you run into any problems, follow the instructions in the output to resolve them.

anges by reloading Nginx.

```
$ sudo systemctl restart nginx
```

After Nginx has restarted, verify that your Shiny Server is serving requests via HTTPS by pointing your browser to `https://example.com`. You should see the same default Shiny Server homepage as you saw at the end of Step 2.

Then, verify that incoming HTTP requests are redirected to HTTPS by typing `http://example.com` into your browser's address bar. If it's working correctly, you should automatically be redirected to `https://example.com`.

Shiny Server is now secured with a reverse proxy and SSL certificate, so you're ready to configure your setup for interactive R Markdown documents.

Step 4 — Hosting Interactive R Documents

Shiny Server is useful not only for hosting Shiny applications but also for hosting interactive R Markdown documents.

At this point, you have a working Shiny Server that can host Shiny applications, but it can't yet host interactive R Markdown documents because the `rmarkdown` R package isn't installed.

So, using a command that works like the one from Step 1 for installing the Shiny package, install `rmarkdown`.

```
$ sudo su - -c "R -e \"install.packages('rmarkdown', repos='http://cran.rstudio.com/')\""
```

Then, verify the installation by going to `https://example.com/sample-apps/rmd/`. You should see an interactive R Markdown document in your browser. Additionally, if you return to `https://example.com`, the error message you received earlier should now be replaced with dynamic content.

If you receive an error message, follow the on-screen instructions and review your terminal output for more information.

Your Shiny Server setup is complete, secured, and ready to serve both Shiny applications as well as Interactive R Markdown documents.

Conclusion

In this tutorial, you set up and secured a fully-functioning Shiny Server that can host Shiny applications and interactive R Markdown documents.

To build on your current setup, you can:

- Learn how to manage and customize the server for your exact needs, with the Shiny Server Administrator's Guide.
- Learn more about writing Shiny applications, with the tutorials on rstudio.com.
- Learn more about writing interactive R markdown documents, by checking out the R Markdown page on rstudio.com.

By: finid

Editor:
Dave Rankin

Related Tutorials

How To Send Web Push Notifications from Django Applications

How To Build a Modern Web Application to Manage Customer Information with Django and React o...

How to Install, Run, and Connect to Jupyter Notebook on a Remote Server

How To Install R on Debian 9

How To Install Webmin on Debian 9

3 Comments

Leave a comment...

Log In to Comment

mdogan *February 20, 2018*

0 it is amazingly nice tutorial but there are some minor issues with the following items;

- Nginx installed by following this How To Install Nginx on Ubuntu 16.04 tutorial, including allowing access to ports 80 and 443 in Step 2 with the command sudo ufw allow 'Nginx Full'
- An A record with example.com pointing to your server's public IP address.
- An A record with www.example.com pointing to your server's public IP address.
- A Let's Encrypt SSL certificate for the domain installed by following this Let's Encrypt Certbot tutorial.

It would be nice to put everything into one tutorial and explain all the points piece by piece

vitalibcf5bd04d9ee52be638e *May 9, 2018*

0 Dean, thanks for this, very useful, I keep struggling with memory allocation though. I cannot open 5GB data.table on 8 GB shiny server while Digital Ocean VM logs only show 33% memory use. If I upgrade to 16GB server all works well, but I don't need 16GB. Any thoughts on how to deal with this memory allocation problem?

```
sudo su - -c "R -e \"install.packages('shiny', repos='http://cran.rstudio.com/')\""
```

To work, changed to:

```
sudo su - -c "R -e \\"install.packages('shiny', repos='http://cran.rstudio.com/')\\""
```