



I'd like to host <https://docs.posit.co/shiny-server/> in-house. Can you provide step by step instructions for someone who has little or no experience of either Linux or Shiny Applications?

You can host Shiny Server in-house by following a clear set of steps that guide you from basic Linux setup through to running your first Shiny application. Below are straightforward, beginner-friendly instructions to get you started with Shiny Server on a supported Linux system.

Overview

Shiny Server lets you run interactive R-powered web apps on your own infrastructure. You'll need a Linux machine, basic terminal access, and internet connectivity for downloading packages and files.^[1]

Step-by-Step Instructions

1. Choose and Prepare Your Linux System

- Use a supported Linux distribution such as **Ubuntu 16.04+**, **RedHat/CentOS 7+**, or **SUSE Linux Enterprise 12+**.^[1]
- If you don't have Linux installed, you can set up Ubuntu on a virtual machine (VM) or use a cloud server (like AWS EC2 or DigitalOcean).

2. Update System Packages

Open the terminal and run:

```
sudo apt update  
sudo apt upgrade
```

This ensures your package repository and system are up to date.^[1]

3. Install R

Shiny Server requires R, which you can install by adding the CRAN repository for the latest version.

- Run these commands:

```
sudo apt install --no-install-recommends software-properties-common dirmngr  
sudo apt-key adv --keyserver keyserver.ubuntu.com --recv-keys 51716619E084DAB9  
sudo add-apt-repository 'deb https://cloud.r-project.org/bin/linux/ubuntu focal-cran40/'  
sudo apt update  
sudo apt install r-base
```

- For Ubuntu, this will install R 3.0 or newer.^[1]

4. Install the Shiny R Package

- Enter R's prompt:

```
sudo su - -c "R"
```

- Then, at the R prompt, type:

```
install.packages("shiny", repos="https://cran.rstudio.com/")  
q()
```

This installs the Shiny package system-wide.^[1]

5. Download and Install Shiny Server

- Download the Shiny Server .deb package:

```
wget https://download3.rstudio.org/ubuntu-14.04/x86_64/shiny-server-1.5.20.1002-  
amd64.deb
```

(Adjust the URL for the latest Shiny Server release)

- Install it:

```
sudo apt install ./shiny-server-1.5.20.1002-amd64.deb
```

- Shiny Server installs to /opt/shiny-server/ and runs on port 3838 by default.^[1]

6. Check Server Status and Start/Stop Shiny Server

- Start Shiny Server:

```
sudo systemctl start shiny-server
```

- Check Status:

```
sudo systemctl status shiny-server
```

- Restart or Stop as needed:

```
sudo systemctl restart shiny-server  
sudo systemctl stop shiny-server
```

Shiny Server runs automatically at boot if enabled.^[1]

7. Add Your First Shiny Application

- Default app directory is /srv/shiny-server/.
- Create a sample app by entering:

```
sudo mkdir /srv/shiny-server/sample-app  
sudo nano /srv/shiny-server/sample-app/app.R
```

Paste in this simple demo code and save:

```
library(shiny)  
ui <- fluidPage(  
  titlePanel("Hello Shiny!"),  
  sidebarLayout(  
    sidebarPanel(  
      sliderInput("obs", "Number of observations:", min = 1, max = 1000, value = 500)  
    ),  
    mainPanel(  
      plotOutput("distPlot")  
    )  
  )  
)  
  
server <- function(input, output) {  
  output$distPlot <- renderPlot({  
    hist(rnorm(input$obs))
```

```
    })
}

shinyApp(ui = ui, server = server)
```

- Set permissions if needed:

```
sudo chown -R shiny:shiny /srv/shiny-server/
```

8. Access Your Shiny App

- Open a browser and go to:
<http://your-server-ip:3838/sample-app>
- You should see your running app!

9. Troubleshooting and Next Steps

- Use server logs in `/var/log/shiny-server/` for debugging.
- Check the [official admin guide](#) for advanced configuration (user authentication, SSL, custom ports).^[1]
- Explore how to deploy more apps by copying into `/srv/shiny-server/` or using custom configuration files.

These steps set up a basic, secure Shiny Server for in-house use. If you need more advanced features such as authentication or app scaling, review the "Professional" sections in the guide and consider Shiny Server Pro.^[1]

**

1. <https://docs.posit.co/shiny-server/>

That's a great help. I'd need to include authentication. Is this possible with the in-house version? Ideally this would be with Microsoft Entra as that is what we currently use.

Yes, adding authentication to the in-house Shiny Server version is possible, but with some limitations. The basic open-source Shiny Server does provide support for simple authentication methods like flat-file password authentication or HTTP Basic Authentication via a reverse proxy (e.g., Nginx). However,

integration with enterprise identity providers such as Microsoft Entra (Azure AD) is not natively built into the open-source version.

For more advanced authentication methods—including Microsoft Entra—you would typically need Shiny Server Pro, which supports PAM (Pluggable Authentication Modules) allowing more flexible authentication integration including Active Directory and OAuth providers. Another approach is to set up an authentication proxy (such as using Nginx or a Node.js-based proxy) in front of the open-source Shiny Server that handles your Microsoft Entra OAuth/OpenID Connect login. This proxy will authenticate users using Microsoft Entra and forward authorized requests to Shiny Server internally.

A common practical solution is:

1. Use Nginx or another reverse proxy to handle HTTPS and authentication.
2. Configure Nginx to authenticate users via Microsoft Entra using OpenID Connect or OAuth2 modules.
3. After successful authentication, proxy requests to Shiny Server on localhost (port 3838).
4. Optionally enforce user access control in the proxy layer.
5. Keep Shiny Server itself running without authentication enabled; all access is guarded by the proxy layer.

There are also community examples and tools, such as "shiny-autho," which add OAuth2 authentication proxies protecting Shiny Server; a similar pattern can be done with Microsoft Entra.

Summary for your case:

- The open-source Shiny Server does not directly support Microsoft Entra authentication.
- You can implement Microsoft Entra authentication externally with a reverse proxy (like Nginx) configured to authenticate users before handing traffic off to Shiny Server.
- Shiny Server Pro does provide built-in integration options more appropriate for enterprise identity solutions including Microsoft Entra.
- Setting up this authentication proxy requires some Linux and web server configuration knowledge but is the recommended method for integrating Microsoft Entra with an in-house Shiny Server.

If beginner-friendly setup is a priority and you want Microsoft Entra integration, consider starting with a simple Nginx reverse proxy setup for authentication and basic network security around Shiny Server, or evaluate Shiny Server Pro for built-in enterprise authentication features.[\[1\]](#)[\[2\]](#)[\[3\]](#)[\[4\]](#)

Let me know if you would like detailed steps for setting up an Nginx reverse proxy with Microsoft Entra authentication for Shiny Server.

**

1. <https://autho.com/blog/adding-authentication-to-shiny-server/>
2. <https://www.r-bloggers.com/2020/06/shiny-server-with-http-authentication/>
3. <https://support.posit.co/hc/en-us/articles/219045987-Shiny-Server-Quick-Start-Require-user-authentication-on-an-application-Pro-only>
4. <https://support.posit.co/hc/en-us/articles/219494957-Shiny-Server-Pro-Active-Directory-Authentication-Examples>
5. <https://stackoverflow.com/questions/41168459/r-shiny-user-authentication-for-a-single-app-r>
6. <https://www.descope.com/blog/post/auth-sso-shiny>
7. <https://ploomber.io/blog/shiny-ldap/>
8. <https://datastorm-open.github.io/shinymanager/>
9. <https://www.datachamp.fr/en/posts/autho/>
10. <https://www.youtube.com/watch?v=Vq32tbqFdiM>

I would like detailed steps for setting up an Nginx reverse proxy with Microsoft Entra authentication for Shiny Server

To set up an Nginx reverse proxy with Microsoft Entra (Azure Active Directory) authentication for your in-house Shiny Server, follow these detailed steps:

Prerequisites

- Shiny Server installed and running on localhost port 3838.

- A Linux server with sudo/root access.
- A registered application in Microsoft Entra (Azure AD) with client ID and tenant information.
- Nginx installed on your Linux server.
- Basic knowledge of Linux terminal and editing config files.

Step 1: Install and configure Nginx

1. Install Nginx if not already installed:

```
sudo apt update
```

```
sudo apt install nginx
```

2. Allow traffic on HTTP and HTTPS ports (80 and 443):

```
sudo ufw allow 'Nginx Full'
```

Step 2: Configure Azure AD Application to allow authentication

1. Register an application in Microsoft Entra portal (Azure AD).
2. Configure redirect URIs to your Nginx server domain.
3. Save the **Client ID** and **Tenant ID** for use in authentication configuration.

Step 3: Set up OAuth2 Proxy for Microsoft Entra authentication

Nginx itself does not support OAuth2/OpenID Connect flows directly, so use an OAuth2 Proxy in front of Nginx that authenticates via Microsoft Entra.

1. Download and install [oauth2-proxy](#).
2. Configure oauth2-proxy with your app credentials:

Example oauth2-proxy.cfg:

```
provider = "azure"
client_id = "<your-client-id>"
client_secret = "<your-client-secret>"
redirect_url = "https://yourdomain.com/oauth2/callback"
oidc_issuer_url = "https://login.microsoftonline.com/<your-tenant-id>/v2.0"
cookie_secret = "<random-32-bytes-base64-secret>"
upstream = "http://localhost:3838"
http_address = "127.0.0.1:4180"
```

3. Start oauth2-proxy:

```
oauth2-proxy --config=/path/to/oauth2-proxy.cfg
```

Step 4: Configure Nginx as reverse proxy to oauth2-proxy and Shiny Server

Edit your Nginx site config, e.g. /etc/nginx/sites-available/shiny:

```
server {
    listen 80;
    server_name yourdomain.com;

    location /oauth2/ {
        proxy_pass http://127.0.0.1:4180/oauth2/;
        proxy_set_header Host $host;
        proxy_set_header X-Real-IP $remote_addr;
        proxy_set_header X-Scheme $scheme;
        proxy_connect_timeout 1;
        proxy_send_timeout 30;
        proxy_read_timeout 30;
    }

    location / {
        auth_request /oauth2/auth;
        error_page 401 = /oauth2/sign_in;
```

```

proxy_pass http://localhost:3838/;
proxy_http_version 1.1;
proxy_set_header Upgrade $http_upgrade;
proxy_set_header Connection $connection_upgrade;
proxy_set_header Host $host;
proxy_set_header X-Real-IP $remote_addr;
proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
proxy_set_header X-Forwarded-Proto $scheme;
proxy_read_timeout 20d;
proxy_buffering off;
}

location = /oauth2/auth {
    proxy_pass http://127.0.0.1:4180/oauth2/auth;
    proxy_set_header Host $host;
    proxy_set_header X-Real-IP $remote_addr;
    proxy_set_header X-Scheme $scheme;
    proxy_pass_request_body off;
    proxy_set_header Content-Length "";
    proxy_intercept_errors on;
    proxy_connect_timeout 1;
    proxy_send_timeout 30;
    proxy_read_timeout 30;
}
}

```

Enable the site and restart Nginx:

```

sudo ln -s /etc/nginx/sites-available/shiny /etc/nginx/sites-enabled/
sudo nginx -t
sudo systemctl restart nginx

```

Step 5: Secure your site with HTTPS

Use Let's Encrypt and Certbot to enable SSL so traffic is encrypted:

```
sudo apt install certbot python3-certbot-nginx  
sudo certbot --nginx -d yourdomain.com
```

How it works:

- Users access https://yourdomain.com.
- Nginx proxies requests to oauth2-proxy that handles Microsoft Entra login.
- After successful login, requests are forwarded to Shiny Server on localhost:3838.
- Only authenticated users can access Shiny applications.

Notes:

- This approach keeps Shiny Server unchanged and hidden behind Nginx + OAuth2 Proxy.
- The proxy layer manages Microsoft Entra OAuth2 authentication securely.
- You can also configure group-based access control in oauth2-proxy if needed.
- Logs from both Nginx and oauth2-proxy help in troubleshooting.

This configuration secures your in-house Shiny Server using Microsoft Entra authentication via an OAuth2 Proxy and Nginx reverse proxy, allowing users to log in with your existing corporate credentials.

If you want, further customization assistance for exact config files or troubleshooting can be provided.

This approach is widely used to enhance Shiny Server security for enterprises without needing Shiny Server Pro and supports seamless integration with Microsoft Entra (Azure AD).[\[1\]](#)[\[2\]](#)[\[3\]](#)[\[4\]](#)

**

1. <https://www.digitalocean.com/community/tutorials/how-to-set-up-shiny-server-on-ubuntu-20-04>
2. <https://heds.nz/posts/ubuntu-shiny-server/>
3. <https://support.posit.co/hc/en-us/articles/213733868-Running-Shiny-Server-with-a-Proxy>
4. <https://support.posit.co/hc/en-us/articles/219494957-Shiny-Server-Pro-Active-Directory-Authentication-Examples>
5. <https://globalparametrics.github.io/SecuringShiny/>
6. https://cran.r-project.org/web/packages/ReviewR/vignettes/deploy_server.html
7. <https://www.r-bloggers.com/2021/08/custom-domain-and-security-for-shinyproxy-with-nginx/>
8. <https://stackoverflow.com/questions/57793976/how-to-fix-reverse-proxy-nginx-with-shiny-server>
9. https://weihanglo.tw/debian-R-setup/doc/other_nginx.html
10. <https://groups.google.com/g/shiny-discuss/c/uLiguxpnM5k>
11. <https://lukesingham.com/shiny-containers-with-shinyproxy/>

Rather than use nginx, could I use a YARP (on Windows) to do the Entra auth and proxy to the linux machine?

Yes, you can use YARP (Yet Another Reverse Proxy) on Windows to perform Microsoft Entra (Azure AD) authentication and proxy requests to your Shiny Server running on Linux.

YARP is a flexible reverse proxy toolkit built on [ASP.NET](#) Core that allows integration of custom middleware, including authentication handlers. This means you can configure your Windows-based YARP reverse proxy to handle Microsoft Entra OpenID Connect or OAuth2 authentication using standard [ASP.NET](#) Core authentication libraries and then forward authenticated traffic to your Linux Shiny Server.

How the setup would work:

- YARP runs on a Windows server or VM.

- It is configured to authenticate users with Microsoft Entra using OpenID Connect middleware.
- After authentication, YARP proxies requests to the Shiny Server URL on the Linux machine (e.g., <http://linux-server:3838>).
- Shiny Server itself remains unchanged and does not perform authentication.
- All user authentication and authorization is handled at the YARP proxy level.

Benefits:

- Allows you to leverage your existing Microsoft Entra authentication provider centrally.
- Keeps your Shiny Server simpler by offloading security to the proxy.
- Works well in heterogeneous environments where you want to keep your Linux Shiny Server and Windows infrastructure aligned.

High-level configuration steps:

1. Create a new [ASP.NET](#) Core web app project for your reverse proxy.
2. Add the YARP NuGet package (`Microsoft.ReverseProxy`).
3. Configure YARP routes and clusters to proxy requests to your Shiny Server backend.
4. Add Microsoft Entra (Azure AD) authentication middleware in `Startup.cs/Program.cs` using `AddMicrosoftIdentityWebApp` or standard OpenID Connect setup.
5. Configure authentication to protect your YARP routes.
6. Deploy and run the YARP proxy on Windows.
7. Direct users to the YARP proxy URL for authentication and access to Shiny apps.

This approach is fully supported and common in enterprise environments where authentication is integrated at the edge proxy layer running on Windows, while backend services run elsewhere.

If needed, detailed YARP sample projects and Microsoft identity integration guides are available from Microsoft documentation to help you implement this quickly.

In summary, using YARP on Windows to do Microsoft Entra authentication and proxy traffic to your Linux Shiny Server is a practical and recommended approach for your scenario. [\[1\]](#)[\[2\]](#)[\[3\]](#)

**

1. <https://learn.microsoft.com/en-us/aspnet/core/fundamentals/servers/yarp/config-files?view=aspnetcore-9.0>
2. <https://systemshogun.com/p/whats-a-proxy-and-a-reverse-proxy>
3. <https://learn.microsoft.com/en-us/shows/on-dotnet/yarp-the-net-reverse-proxy>
4. <https://www.youtube.com/watch?v=gJqJ1yg5FoA>
5. <https://www.youtube.com/watch?v=-SiYAYp5AOI>
6. <https://github.com/alexandrejulien/yarp-reverse-proxy>
7. <https://learn.microsoft.com/en-us/aspnet/core/fundamentals/servers/yarp/getting-started?view=aspnetcore-9.0>
8. <https://microsoft.github.io/reverse-proxy/>
9. https://www.reddit.com/r/dotnet/comments/1oy2cud/how_to_host_yarp_on_windows_server/