AI-03978 NGINX Web Server

Owner	Ⓡ Rigel Arcayan
Tags	Project
Created time	June 25, 2026 6:36 PM

Generate Nginx Autoindex PDF server Compose but consider what I have running in my dockerlab:

Bridge=dockerlab-network
Domain=blindcaveman.duckdns.org
I have a running NPM server
Location of compose= /home/rarcayan/dockerlab/apps/webserver
Location of files to share= /home/rarcayan/inetpub/

Create a procedure on how to configure the proxy configuration as well.

Here is the complete deployment configuration and reverse proxy procedure tailored precisely to your dockerlab environment.

1. Directory Setup & Configuration Files

Navigate to your webserver app directory:

```
mkdir -p /home/rarcayan/dockerlab/apps/webserver
cd /home/rarcayan/dockerlab/apps/webserver
```

Create the following two files inside /home/rarcayan/dockerlab/apps/webserver:

nginx.conf

This file configures Nginx to enable the directory listing (autoindex), show file sizes in a human-readable format, use local time, and ensure PDFs open directly in the browser instead of forcing a download.

```
server {
    listen 80;
```

```
    server_name localhost;

    location / {
        root /usr/share/nginx/html;

        # Enable autoindex listing
        autoindex on;
        autoindex_exact_size off; # Displays file size in KB, MB, or GB
        autoindex_localtime on;   # Uses the container's local timezone for file dates

        # Ensure correct MIME types so PDFs open directly in-browser
        include /etc/nginx/mime.types;
        default_type application/octet-stream;

        # Optional: Force inline header for PDFs just in case
        location ~* \.pdf$ {
            add_header Content-Disposition "inline";
        }
    }
}
```

docker-compose.yml

This compose file mounts your shared documents from /home/rarcayan/inetpub/ as read-only (:ro) to protect your data, attaches to your existing dockerlab-network, and exposes no public ports directly, keeping it securely routed through your Nginx Proxy Manager (NPM).

```yaml
version: '3.8'

services:
  pdf-server:
    image: nginx:alpine
    container_name: pdf-autoindex-server
    restart: unless-stopped
    volumes:
      - /home/rarcayan/inetpub:/usr/share/nginx/html:ro
      - ./nginx.conf:/etc/nginx/conf.d/default.conf:ro
    networks:
      - dockerlab-network
    environment:
      - TZ=Asia/Manila
```

```
networks:
  dockerlab-network:
    external: true
```

Deploy the Container

Run the following command to start the service:

```
docker compose up -d
```

2. Nginx Proxy Manager (NPM) Configuration Procedure

Since your new pdf-autoindex-server container is attached to the same **dockerlab-network** as your NPM container, NPM can route traffic directly using the container name as the hostname.

Step 1: Access NPM Admin Dashboard

1. Open your web browser and log into your NPM administration console.

Step 2: Create a New Proxy Host

1. Navigate to **Hosts** > **Proxy Hosts**.
2. Click the **Add Proxy Host** button in the top right.

Step 3: Populate the "Details" Tab

- **Domain Names:** Input your desired subdomain (e.g., pdf.blindcaveman.duckdns.org or share.blindcaveman.duckdns.org). Press Enter after typing.
- **Scheme:** http
- **Forward Host / IP:** pdf-autoindex-server *(This matches the container_name defined in your compose file)*
- **Forward Port:** 80
- **Cache Assets:** Optional (Leave disabled if files change frequently).

- **Block Common Exploits: Enabled** (Recommended security baseline).
- **Websockets Support:** Disabled (Not required for basic autoindex serving).

Step 4: Populate the "SSL" Tab

1. Click on the **SSL** tab at the top of the window.
2. Under **SSL Certificate**, select **Request a new SSL Certificate** from the dropdown menu.
3. Toggle **Force SSL** to **Enabled** (Ensures all traffic to your PDF server is encrypted).
4. Toggle **HTTP/2 Support** to **Enabled**.
5. Enter your email address associated with your Let's Encrypt registration.
6. Check **I Agree to the Let's Encrypt Terms of Service**.

Step 5: Save and Test

1. Click **Save**. NPM will communicate with Let's Encrypt to issue your certificate (this may take 10–30 seconds).
2. Once saved, open a browser window and navigate to https://your-configured-subdomain.blindcaveman.duckdns.org.
3. You should instantly see a clean, indexed list of all files and folders located within your /home/rarcayan/inetpub/ directory. Clicking any PDF will load it natively inside the browser frame.

How to upload my local files to the pdf-auto index-server which is hosted inside my GCP VM instance? Create a simple procedure or give me options on what software to use to sync. Can I use a Google Drive folder?

You have a few excellent options for getting your files from your local machine into that /home/rarcayan/inetpub/ directory on your GCP VM.
Since you asked about **Google Drive**, let’s look at how to set that up first, followed by two fast, secure alternatives if you want to bypass the cloud entirely.

Option 1: Bi-Directional Sync Using Google Drive (Recommended for Automation)

You can absolutely use Google Drive. To do this on a headless Linux server, you use an open-source tool called **Rclone**. It acts as a bridge, syncing a specific Google Drive folder directly to your /home/rarcayan/inetpub/ directory.

Step 1: Install Rclone on your GCP VM

Connect to your VM via SSH and run:

```
sudo curl <https://rclone.org/install.sh> | sudo bash
```

Step 2: Configure the Link to your Google Drive

1. Run rclone config.
2. Type n for a new remote, and name it gdrive.
3. Look through the list and find **Google Drive** (usually number 18 or similar) and enter its number.
4. Leave client_id and client_secret blank (just press Enter).
5. Choose scope 1 (Full access to all files).
6. When it asks to **Use auto-config?**, type n (because you are on a remote cloud server without a web browser).
7. Rclone will output a command that looks like rclone authorize "drive" "...".
8. **On your local machine**, install Rclone briefly, paste that command into your local terminal, log into your Google account in the browser window that pops up, and copy the success code it gives you. Paste that code back into your GCP terminal.
9. Quit the config tool (q).

Step 3: Run the Sync Command

Create a folder in your Google Drive named PDF-Share. To pull those files down to your Nginx server, run:

```
rclone sync gdrive:PDF-Share /home/rarcayan/inetpub/ --progre
```

```
ss
```

(To automate this, you can drop this single line into a cron job to run every 15 minutes).

Option 2: Drag-and-Drop via SFTP (Best for Manual Uploads)

If you just want to manually drag files from your local computer's desktop straight to the server without relying on Google Drive, use an **SFTP Client** like **FileZilla** or **Cyberduck**.
Because it's a GCP VM, you'll connect using the exact same SSH key pair you use to log into the terminal.

1. Get your SSH Key
   Local Setup
   Locate the private key file (.ssh/id_rsa or .pem) that you use to SSH into your GCP instance.
2. Configure your SFTP Client
   FileZilla / Cyberduck
   Open the app, go to Settings/Preferences → SFTP, and click Add key file. Point it to your private key.
3. Connect to the VM
   Protocol: SFTP
   Host: Your VM's external IP address (or blindcaveman.duckdns.org if it points to the WAN IP).
   Protocol: SFTP (Port 22)
   Logon Type: Key File / Ask for Password
   User: rarcayan
4. Transfer Files
   Permissions Check
   Navigate the right-hand panel to /home/rarcayan/inetpub/ and drag your local files into it.

Option 3: Terminal Sync via rsync (Fastest for Power Users)

If you use a Linux or macOS terminal locally (or WSL on Windows), rsync is the most efficient protocol available. It compares your local folder with the GCP folder

and only uploads files that are new or changed.
Run this command from your **local terminal**:

```
rsync -avz -e "ssh -i /path/to/your/gcp-private-key" /path/to/local/pdf/folder/ rarcayan@blindcaveman.duckdns.org:/home/rarcayan/inetpub/
```

⚠️ A Quick Note on Permissions

Because Nginx runs inside a container as a non-root user (nginx), ensure that whatever method you use leaves the files readable. If Nginx throws a 403 Forbidden error after you upload files, quickly jump onto the VM and fix the permissions with:

```
sudo chmod -R 755 /home/rarcayan/inetpub/
```

Which of these workflows fits best into your current routine? If you go the Rclone route and want a quick cron snippet to handle the background automation, just let me know.