Host n8n on a Google Cloud VM instance (server) completely **FREE**, using your own domain

YouTube tutorial: https://youtu.be/5xmSb7q9wuk?si=Wg_O-dVaKOPRU_GM

Step 1: Connect to your VM instance via SSH:

From the Google Cloud Console, go to your VM instance list.

Click on SSH to connect directly through the browser.

Step 2: Update your VM and install Docker & Docker Compose:

Run these commands sequentially to install Docker and Docker Compose easily:

```
sudo apt update -y

sudo apt install docker.io -y

sudo systemctl start docker

sudo systemctl enable docker

sudo usermod -aG docker $USER
```

Restart SSH session after adding your user to Docker group for changes to take effect:

```
exit

ssh username@your-vm-instance-ip
```

Step 3: Create and run your n8n Docker container:

Create a dedicated folder:

```
mkdir ~/n8n

cd ~/n8n
```

Step 3.5 (if it is the case): Check Ownership

First, verify who owns the directory:

```
ls -ld ~/.n8n
```

If the directory is owned by `root` or another user, you need to change its ownership to your user (in your case, will be the email's username. E.g. email@example.com, the user is email). You can do this with the following command:

```
sudo chown -R $(whoami):$(whoami) ~/.n8n
```

After you've changed the ownership, set the proper permissions:

```
chmod 755 ~/.n8n
```

Step 4: Run your Docker container with your domain configuration

Replace `example.com` with your actual domain name:

```
sudo docker run -d --restart unless-stopped -it \

--name n8n \

-p 127.0.0.1:5678:5678 \

-e N8N_HOST="n8n.example.com" \

-e WEBHOOK_TUNNEL_URL="https://n8n.example.com/" \

-e WEBHOOK_URL="https://n8n.example.com/" \

-v ~/.n8n:/root/.n8n \

n8nio/n8n
```

Step 5: Allow access through the VM firewall, opening the Firewall Port (5678) in Google Cloud:

In the Google Cloud Console:

- Navigate to **VPC Network** → **Firewall**.

- Click **Create Firewall Rule**:
    - **Name**: `allow-n8n`
    - **Targets**: `All instances`
      IP ranges: `0.0.0.0/0`
    - **Protocols and ports**: Check "TCP" and type `5678`.
- Click **Create**.

Step 6: Access n8n interface:

On your domain registrar (Cloudflare, Namecheap, GoDaddy, etc.):

- Create an **A Record** pointing `n8n.example.com` to your VM's external IP:
    - **Type**: A
    - **Name:** n8n
    - **Content:** your-vm-instance-external-ip
    - **TTL:** 1 hr

Step 7: Secure your n8n installation (HTTPS)

To secure your n8n installation, consider adding a reverse proxy like **Caddy** (easiest) with Let's Encrypt certificates.

```
sudo apt update # optional if we already have installed Caddy

sudo apt install -y debian-keyring debian-archive-keyring
apt-transport-https curl

curl -1sLf 'https://dl.cloudsmith.io/public/caddy/stable/gpg.key'
| sudo gpg --dearmor -o
/usr/share/keyrings/caddy-stable-archive-keyring.gpg

curl -1sLf
'https://dl.cloudsmith.io/public/caddy/stable/debian.deb.txt' |
sudo tee /etc/apt/sources.list.d/caddy-stable.list

sudo apt update

sudo apt install caddy -y
```

Create a new Caddyfile:

```
sudo nano /etc/caddy/Caddyfile
```

Paste the following configuration:

```
n8n.example.com {

    reverse_proxy 127.0.0.1:5678

}
```

Save (`Ctrl + O`, Enter) and exit (`Ctrl + X`).

Restart Caddy to apply changes:

```
sudo systemctl restart caddy
```

Step 8 (Optional): Secure Your n8n Workspace

To prevent unauthorized access, it's highly recommended to enable basic authentication. You can do this by setting the following environment variables when you run your Docker container:

```
-e N8N_BASIC_AUTH_ACTIVE=true \
-e N8N_BASIC_AUTH_USER="your_username" \
-e N8N_BASIC_AUTH_PASSWORD="your_secure_password" \
```

First, remove the existing container with:

```
sudo docker rm -f n8n
```

Update the Docker command with the basic authentication enabled, wich might look like:

```
sudo docker run -d --restart unless-stopped -it \

--name n8n \

-p 127.0.0.1:5678:5678 \

-e N8N_HOST="n8n.example.com" \

-e WEBHOOK_TUNNEL_URL="https:/n8n.example.com" \

-e WEBHOOK_URL="https://n8n.example.com" \

-e N8N_BASIC_AUTH_ACTIVE=true \

-e N8N_BASIC_AUTH_USER="your_username" \

-e N8N_BASIC_AUTH_PASSWORD="your_secure_password" \

-v ~/.n8n:/root/.n8n \

n8nio/n8n
```

You can add an extra layer of authentication at the reverse proxy level so that visitors must enter a password or code before they even reach the n8n sign-in page. We need to create a hashed password, run the following command and copy the given hashed password string (it will look something like `$2a$12$...`):

```
caddy hash-password --plaintext "your_password"
```

Open your Caddyfile:

```
sudo nano /etc/caddy/Caddyfile
```

Add the basic auth Directive replacing the contents with the following configuration, making sure to substitute `your_hashed_password_string` with the hash pass you generated:

```
n8n.example.com {

    basicauth {

        "your_username_to_unlock" your_hashed_password_string

    }

    reverse_proxy 127.0.0.1:5678

}
```

Save (`Ctrl + O`, Enter) and exit (`Ctrl + X`).

Restart Caddy to apply changes:

```
sudo systemctl restart caddy
```

See you in your next automation project! :)

NovoAIGen team.

Considerations:

- When you are creating the new A record in your domain registrar (GoDaddy, Namecheap, Hostinger, etc.), remove or disable any placeholder/parking pages from them if they're still enabled.
- In GoDaddy DNS, look for any A record or CNAME with a "Name" or "Host" like "WebsiteBuilder Site" or pointing to a GoDaddy IP, to avoid conflicts when open the domain.
- You can check your domain with a DNS checker (e.g., dnschecker.org)
- Google Cloud Console: https://console.cloud.google.com/
- Google Cloud Free Program usage limits: https://cloud.google.com/free/docs/free-cloud-features