🚀 Deployment Steps: Chat/Voice Bot (n8n) on Google Cloud🚀

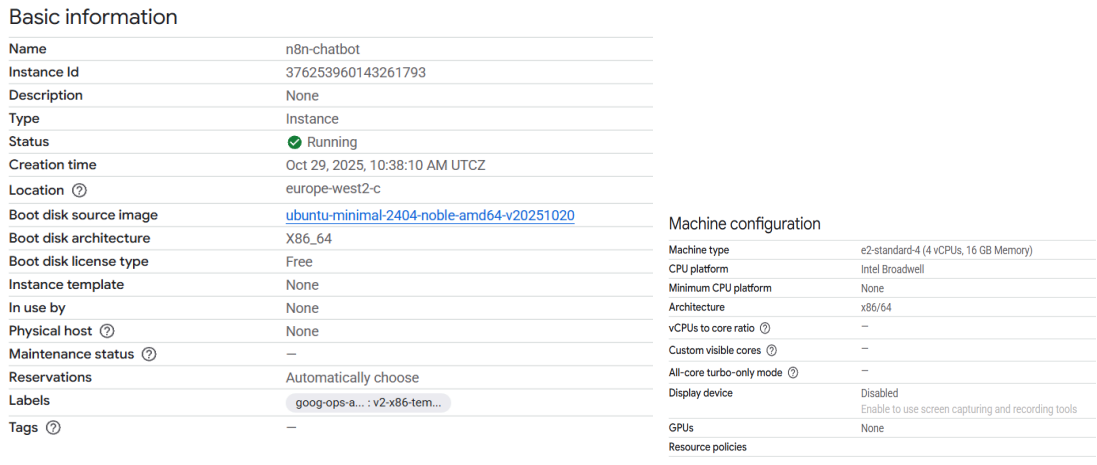
Step 1: Google Cloud Infrastructure Setup

1.1 Create Google Cloud Project and virtual machine

To begin the deployment, create a **Virtual Machine (VM) instance** in [Google Cloud Console]. This VM acts as the hosting environment for the n8n workflow automation tool, vector database connection, and the large language model (LLM) API integrations.

Steps Followed:

1. Logged into the Google Cloud Console.
2. Navigated to **Compute Engine → VM Instances → Create Instance**.
3. Selected the **free-tier eligible machine type** for the initial deployment to test and configure the environment.
    - **vCPU:** 4
    - **Memory:** 16 GB
    - **OS Image:** ubuntu-minimal-2404-noble-amd64-v20251020
    - **Storage:** 32 GB
4. Configured **firewall rules** to allow **HTTP and HTTPS traffic**.

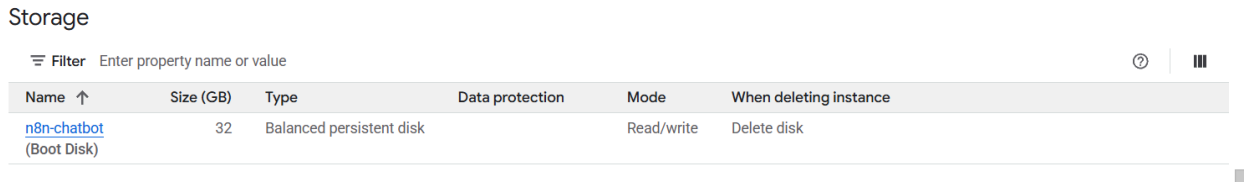


Basic information

Name	n8n-chatbot
Instance Id	376253960143261793
Description	None
Type	Instance
Status	Running
Creation time	Oct 29, 2025, 10:38:10 AM UTCZ
Location	europe-west2-c
Boot disk source image	ubuntu-minimal-2404-noble-amd64-v20251020
Boot disk architecture	X86_64
Boot disk license type	Free
Instance template	None
In use by	None
Physical host	None
Maintenance status	–
Reservations	Automatically choose
Labels	goog-ops-a... : v2-x86-tem...
Tags	–

Machine configuration

Machine type	e2-standard-4 (4 vCPUs, 16 GB Memory)
CPU platform	Intel Broadwell
Minimum CPU platform	None
Architecture	x86/64
vCPUs to core ratio	–
Custom visible cores	–
All-core turbo-only mode	–
Display device	Disabled Enable to use screen capturing and recording tools
GPUs	None
Resource policies	



Note:

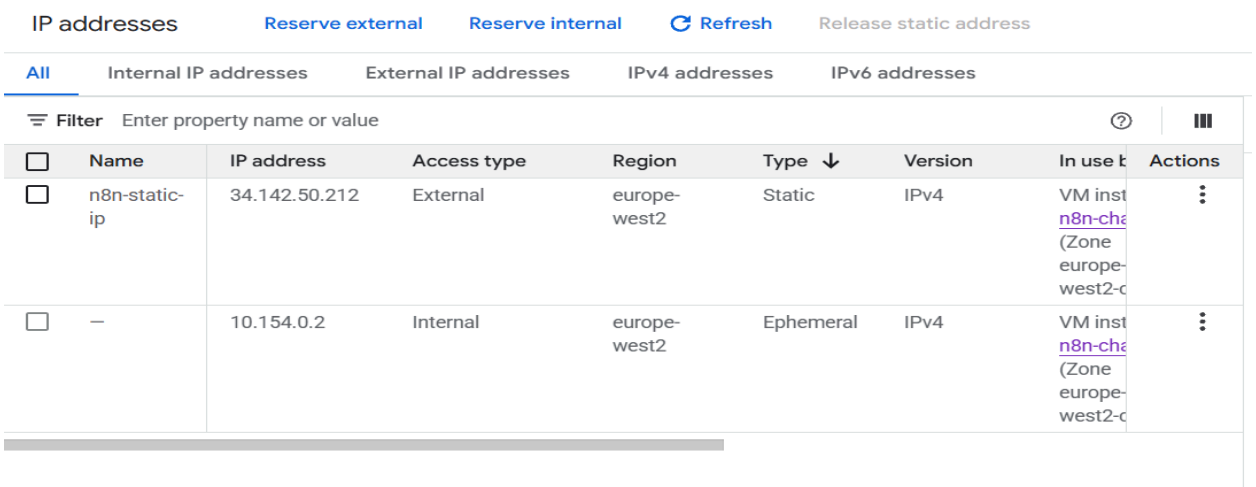
For now, I used the **maximum specification allowed under the Google Cloud Free Tier**. However, since the workflow involves **crawling, text to speech and vice-versa**, this VM configuration should be **upgraded** in the future to ensure better performance and stability during concurrent user interactions.

1.2 Reserve Static IP Address

Static IP addresses solve a critical problem: virtual machines can get different IP addresses when they restart, which would break your domain configuration and make your n8n instance unreachable. A static IP ensures your domain always points to the same location.

In VPC network > IP addresses:

1. Click “**Reserve External**”
2. Name: n8n-static-ip
3. Type: Regional
4. Region: europe-west2
5. Attach to: created VM instances
6. Click “Reserve”



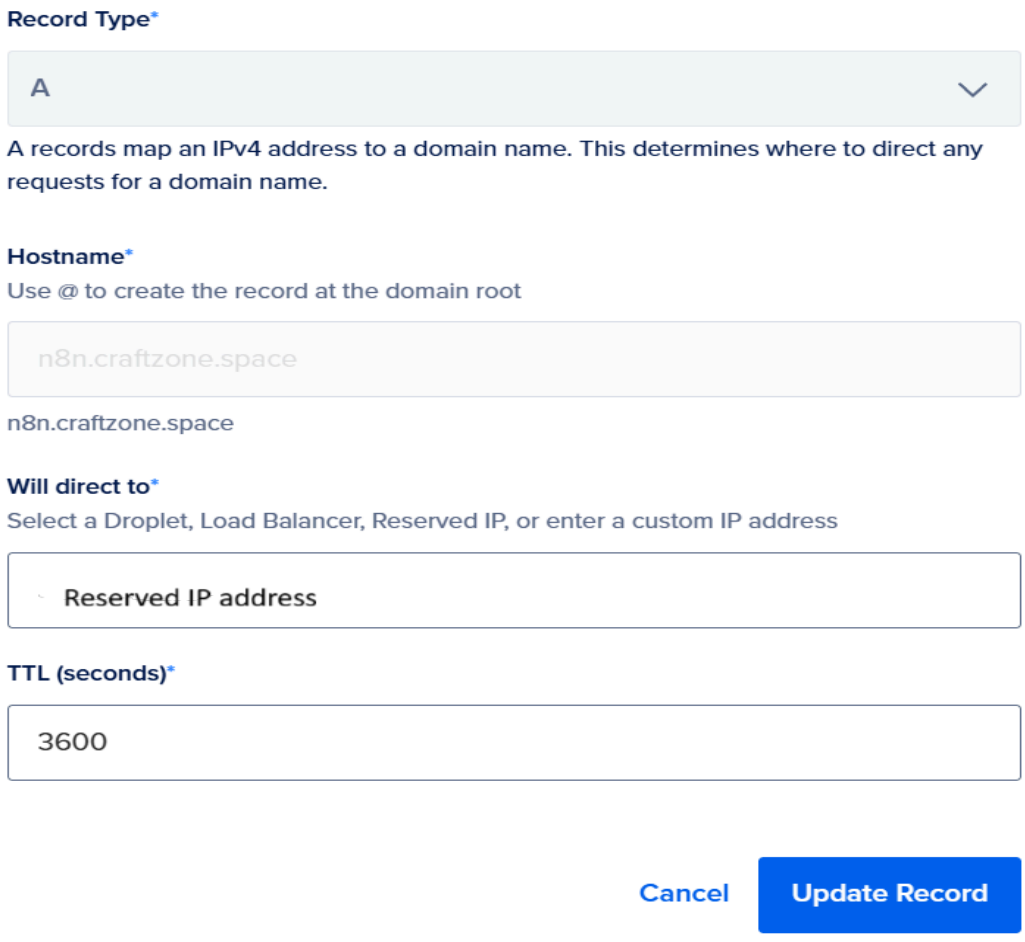
Note:

Document this IP address as you’ll need it for DNS configuration in the next step.

Step 2: Domain DNS Configuration

2.1 Add DNS Record
In your domain provider's DNS management interface, create an A record that points your chosen subdomain to your Google Cloud server:

Record Type: A
Name/Host: n8n (for subdomain) or @ (for root domain)
Value: [Your Static IP from Step 1.2]
TTL: provider default
For a subdomain like n8n.example.com, use n8n as the name. The TTL (Time To Live) value determines how long other servers cache this DNS information before checking for updates.



Step 3: Server Preparation

3.1 Connect to Server

Google Cloud provides browser-based SSH access, eliminating the need for additional software or key management. This secure terminal connection allows you to execute commands directly on your server. In the Compute Engine console, find your VM and click the “SSH” button to open a secure terminal session.

3.2 Update System Packages

Before installing any new software, we need to ensure your server has the latest security patches and package versions. This foundational step prevents compatibility issues and security vulnerabilities:

```
# Update package lists to get information about newer versions
sudo apt update

# Upgrade all installed packages to their latest versions
sudo apt upgrade -y
```

Think of this as getting your server up to current standards before we start building our n8n environment on top of it.

Step 4: Docker Packages Installation (Source)

Docker containerization simplifies n8n deployment by packaging the application with all its dependencies into a portable container. This approach ensures consistency across different environments and simplifies updates:

4.1 Install using the apt repository

Before you install Docker Engine for the first time on a new host machine, you need to set up the Docker apt repository. Afterward, you can install and update Docker from the repository.

```
# Add Docker's official GPG key:
sudo apt-get update
sudo apt-get install ca-certificates curl
sudo install -m 0755 -d /etc/apt/keyrings
sudo curl -fsSL https://download.docker.com/linux/ubuntu/gpg -o
```

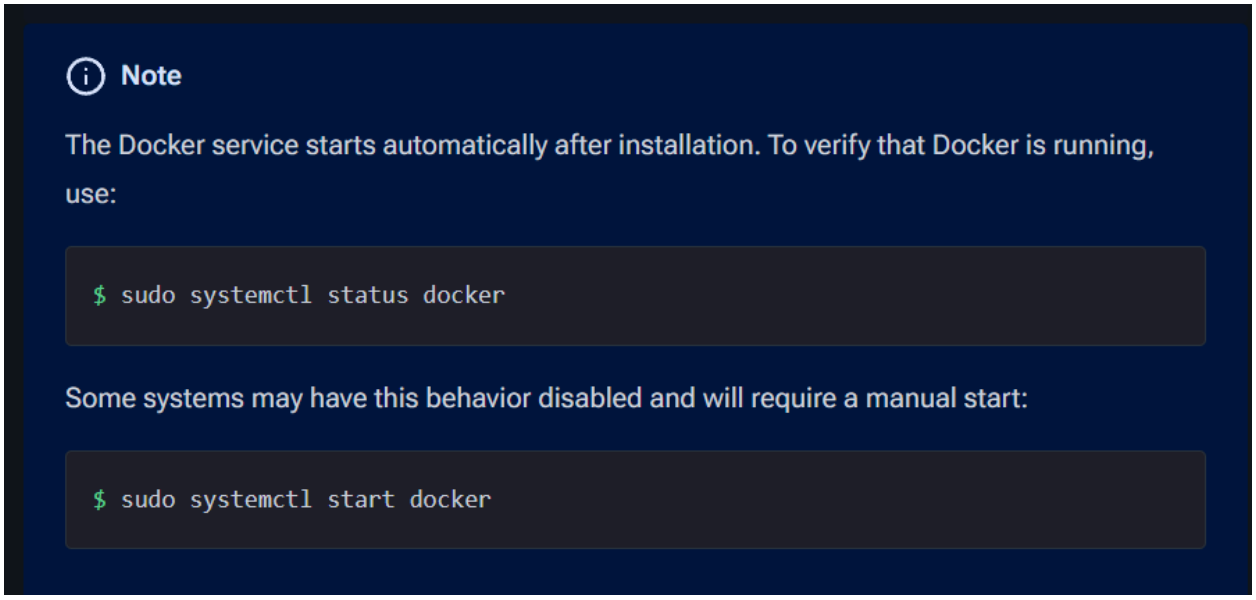
```
/etc/apt/keyrings/docker.asc
sudo chmod a+r /etc/apt/keyrings/docker.asc

# Add the repository to Apt sources:
echo \
  "deb [arch=$(dpkg --print-architecture)
signed-by=/etc/apt/keyrings/docker.asc]
https://download.docker.com/linux/ubuntu \
  $(. /etc/os-release && echo "${UBUNTU_CODENAME:-$VERSION_CODENAME}")
stable" | \
  sudo tee /etc/apt/sources.list.d/docker.list > /dev/null
sudo apt-get update
```

4.2 Install the Docker packages

```
sudo apt-get install docker-ce docker-ce-cli containerd.io
docker-buildx-plugin docker-compose-plugin
```

Note:



Step 5: Create Docker Compose Configuration

5.1 Create Project Directory

```
# Create and navigate to n8n project directory
mkdir ~/n8n-docker && cd ~/n8n-chatbot
```

This directory will contain your Docker Compose configuration and any other related files, keeping everything organized and version-controllable.

5.2 Create .env

```
nano .env
```

```
#n8n config
SUBDOMAIN=<your-subdomain>
DOMAIN= <your-domain>
DATA_FOLDER=/home/<ssh-user>/n8n-chatbot
GENERIC_TIMEZONE=Europe/London

#Postgres
POSTGRES_PASSWORD=postgres
POSTGRES_USER=postgres
```

5.3 Docker Compose File

```
# Create the Docker Compose configuration file
nano docker-compose.yml
```

```
version: '3.8'

services:
  n8n:
    image: docker.n8n.io/n8nio/n8n
    container_name: n8n-service
    restart: always
    ports:
      - 5678:5678
```

```
    environment:
      - N8N_HOST=${SUBDOMAIN}.${DOMAIN}
      - N8N_PORT=5678
      - N8N_PROTOCOL=https
      - NODE_ENV=production
      - WEBHOOK_URL=https://${SUBDOMAIN}.${DOMAIN}/
      - GENERIC_TIMEZONE=${GENERIC_TIMEZONE}
    volumes:
      - n8n_data:/home/node/.n8n
      - ${DATA_FOLDER}/local_files:/files
    networks:
      - backend
    depends_on:
      - postgres

  postgres:
    image: postgres:18
    container_name: postgres-service
    ports:
      - '5432:5432'
    restart: unless-stopped
    environment:
      POSTGRES_USER: ${POSTGRES_USER}
      POSTGRES_PASSWORD: ${POSTGRES_PASSWORD}
      PGDATA: /var/lib/postgresql/data
    volumes:
      - n8n_db:/var/lib/postgresql/data
    networks:
      - backend

  crawl4ai:
    image: unclecode/crawl4ai:latest
```

```
    container_name: crawl4ai
    ports:
      - '11235:11235'
    volumes:
      - /dev/shm:/dev/shm
    networks:
      - backend
    restart: always

  kokoro-fastapi-cpu:
    image: ghcr.io/remsky/kokoro-fastapi-cpu
    container_name: kokoro-service
    ports:
      - 8880:8880
    restart: always
    networks:
      - backend

  whisper-asr-webservice:
    image: onerahmet/openai-whisper-asr-webservice:latest
    container_name: whisper-service
    environment:
      - ASR_MODEL=base
      - ASR_ENGINE=openai_whisper
    ports:
      - 9000:9000
    volumes:
      - cache-whisper:/root/.cache
    networks:
      - backend
    restart: always
```

```
volumes:
  n8n_data:
    external: true
  n8n_db:
    external: true
  cache-whisper:
    external: true

networks:
  backend:
    driver: bridge
```

5.4 Create Volumes

```
docker volume create n8n_data
docker volume create n8n_db
docker volume create cache-whisper
```

5.5 Deploy n8n with Docker Compose

Make sure you're in the directory where you created the Docker Compose file.

```
docker compose pull
docker compose up -d
```

Step 6: Nginx Installation and Configuration

6.1 Install Nginx

Nginx serves as a reverse proxy, creating a secure front door for your n8n instance. This architecture provides several benefits: SSL termination, better security than exposing n8n directly, improved performance through caching, and the ability to serve multiple applications from the same server:

```
# Install Nginx web server
sudo apt install nginx -y
```

Nginx will handle all incoming web traffic and route it securely to your n8n Docker container running on port 5678.

6.2 Create Nginx Configuration
Configure Nginx to properly proxy requests to your containerized n8n instance:

```
# Create n8n-specific Nginx configuration
sudo nano /etc/nginx/sites-available/n8n.conf
```

Add this configuration, replacing n8n.example.com with your domain:

```
server {
listen 80;
server_name n8n.example.com;

# Redirect all HTTP traffic to HTTPS (will be added by Certbot)
location / {
proxy_pass http://localhost:5678;
proxy_http_version 1.1;

# Disable buffering for real-time functionality
chunked_transfer_encoding off;
proxy_buffering off;
proxy_cache off;

# WebSocket support for n8n's real-time features
proxy_set_header Upgrade $http_upgrade;
proxy_set_header Connection "upgrade";

# Essential proxy headers for proper functionality
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;

# Extended timeout for long-running workflows
proxy_read_timeout 86400;
proxy_connect_timeout 60;
proxy_send_timeout 60;
}
}
```

6.3 Enable Nginx Configuration

```
# Enable the n8n site configuration
sudo ln -s /etc/nginx/sites-available/n8n.conf /etc/nginx/sites-enabled/

# Test configuration syntax to catch any errors
sudo nginx -t

# Restart Nginx to apply changes
sudo systemctl restart nginx

# Verify Nginx is running properly
sudo systemctl status nginx
```

The symbolic link approach is a standard Nginx practice that allows you to easily enable and disable site configurations. Testing the configuration before restarting prevents Nginx from failing due to syntax errors.

Step 7: SSL Certificate Setup

7.1 Install Certbot

Certbot provides free SSL certificates from Let's Encrypt, a certificate authority that has revolutionized web security by making SSL certificates freely available. The Nginx plugin automatically configures your web server:

```
# Install Certbot and Nginx integration plugin
sudo apt install certbot python3-certbot-nginx -y
```

Let's Encrypt certificates are trusted by all major browsers and provide the same level of security as paid certificates.

7.2 Obtain SSL Certificate

Generate and install an SSL certificate for your domain:

```
# Request SSL certificate (replace with your domain)
sudo certbot --nginx -d n8n.example.com
```

Navigate to https://n8n.example.com in your web browser