Important: Use the commands in the exact same order as shown in the course. At the end of that document you find a short explanation to each command so you know what it does and why we do it.

Command #1:
apt update && apt -y install ca-certificates curl gnupg

Command #2:
curl -fsSL https://get.docker.com | sh

Command #3:
systemctl enable --now docker

Command #4:
docker --version
docker compose version
systemctl status docker —no-pager

Command #5:
ufw allow 22

Command #6:
ufw allow 80

Command #7:
ufw allow 443

Command #8:
ufw --force enable

Command #9:
ufw status verbose

Command #10:
openssl rand -hex 32

Command #11:
mkdir -p /srv/n8n/data /srv/n8n/db

Command #12:
chown -R 1000:1000 /srv/n8n/data

chmod -R 700 /srv/n8n/data

Command #13:
chown -R 999:999 /srv/n8n/db

Command #14:
cd /srv/n8n

nano .env

Command #15: (Attention, continues next page!)

```
# --- DOMAIN / URL ---

N8N_HOST=example.yourdomain.com

N8N_PORT=5678

N8N_PROTOCOL=https

WEBHOOK_URL=https://example.yourdomain.com/


# --- SECURITY ---

N8N_ENCRYPTION_KEY=REPLACE_WITH_HEX_KEY

N8N_DISABLE_PRODUCTION_MAIN_PROCESS=false

N8N_BASIC_AUTH_ACTIVE=true

N8N_BASIC_AUTH_USER=admin

N8N_BASIC_AUTH_PASSWORD=change-this-strongly


# --- TIMEZONE ---

GENERIC_TIMEZONE=Europe/Sofia


# --- POSTGRES ---

POSTGRES_USER=n8n

POSTGRES_PASSWORD=change_me

POSTGRES_DB=n8n


# --- Optional Gmail SMTP (for password reset, invites, alerts) ---

N8N_EMAIL_MODE=smtp

N8N_SMTP_HOST=smtp.gmail.com

N8N_SMTP_PORT=587

N8N_SMTP_USER=YOUR_MAIL@gmail.com
```

N8N_SMTP_PASS=YOUR_GOOGLE_APP_PASSWORD

N8N_SMTP_SENDER=YOUR_MAIL@gmail.com

N8N_SMTP_SSL=false

Command #16:
cd /srv/n8n

nano docker-compose.yml

Command #17: (Attention, continues next page!)

```
services:

  db:

    image: postgres:16

    environment:

      POSTGRES_USER: ${POSTGRES_USER}

      POSTGRES_PASSWORD: ${POSTGRES_PASSWORD}

      POSTGRES_DB: ${POSTGRES_DB}

    volumes:

      - ./db:/var/lib/postgresql/data

    restart: unless-stopped


  n8n:

    image: n8nio/n8n:1.78.1

    depends_on:

      - db

    env_file:

      - .env

    environment:

      DB_TYPE: postgresdb

      DB_POSTGRESDB_HOST: db

      DB_POSTGRESDB_PORT: 5432

      DB_POSTGRESDB_DATABASE: ${POSTGRES_DB}

      DB_POSTGRESDB_USER: ${POSTGRES_USER}
```

```
    DB_POSTGRESDB_PASSWORD: ${POSTGRES_PASSWORD}
    N8N_HOST: ${N8N_HOST}
    N8N_PORT: ${N8N_PORT}
    N8N_PROTOCOL: ${N8N_PROTOCOL}
    WEBHOOK_URL: ${WEBHOOK_URL}
  ports:
    - "5678:5678"
  volumes:
    - ./data:/home/node/.n8n
  restart: unless-stopped
```

Command #18:
mkdir -p /etc/nginx/sites-available /etc/nginx/sites-enabled

Command #19:
nano /etc/nginx/sites-available/n8n

Command #20:
```
server {
  listen 80;
  server_name n8n.YOURDOMAIN.com;
  client_max_body_size 50m;

  location / {
    proxy_pass http://127.0.0.1:5678/;
    proxy_http_version 1.1;
    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_set_header Upgrade $http_upgrade;
    proxy_set_header Connection "upgrade";
  }
}
```

Command #20:
apt update && apt -y install nginx

Command #21:
ln -s /etc/nginx/sites-available/n8n /etc/nginx/sites-enabled/

nginx -t && systemctl reload nginx

Command #23:
apt -y install certbot python3-certbot-nginx

certbot --nginx -d course.germanaicreator.com -m germanaicreator@gmail.com --agree-tos --redirect -n

Command #24:
systemctl start docker

Command #25:
cd /srv/n8n

Command #26:
docker compose up -d

Command #27:
docker compose ps

Command #28: (Open n8n with YOUR domain)
https://example.yourdomain.com

Explanation:

Command #1 – Updates the package list so your server knows about the latest available software versions, then installs basic tools needed to handle secure connections, download files, and verify signatures. This ensures the rest of the setup runs smoothly.

Command #2 – Downloads and installs Docker using the official install script from Docker. Docker is the container engine that will run both n8n and its database.

Command #3 – Configures Docker to automatically start when the server boots and starts it right now. Without this, you'd have to manually start Docker every time the server restarts.

Command #4 – Checks that Docker and Docker Compose are installed and working, and verifies the Docker service is running properly in the background.

Command #5 – Opens port 22 on the firewall so you can connect to the server via SSH. Without this, you'd lose remote access.

Command #6 – Opens port 80, which is needed for standard web (HTTP) traffic before we set up HTTPS.

Command #7 – Opens port 443, which is the secure web traffic (HTTPS) port that browsers use after SSL is enabled.

Command #8 – Turns the firewall on and applies your rules without asking for confirmation, protecting your server from unwanted traffic.

Command #9 – Shows the firewall's current status and which ports are open so you can confirm everything is correct.

Command #10 – Generates a 32-character random encryption key. This key is used by n8n to encrypt sensitive data like credentials inside workflows.

Command #11 – Creates the folders where n8n will store workflow data and where PostgreSQL will store the database. Separating them helps with organization and backups.

Command #12 – Changes ownership and sets secure permissions for the data folder so only the n8n process can read and write to it.

Command #13 – Adjusts ownership of the db folder so the PostgreSQL database container has the correct permissions to operate.

Command #14 – Navigates to the /srv/n8n folder and opens the .env file in a text editor. This file stores all environment variables that configure n8n.

Command #15 – Fills out the .env file with your domain, port settings, encryption key, security credentials, timezone, database connection info, and optional email settings for features like password resets.

Command #16 – Stays in /srv/n8n and opens the docker-compose.yml file, which defines how Docker should run n8n and the database.

Command #17 – Defines two services: PostgreSQL (the database) and n8n (the automation app). This section sets ports, links services, and ensures data is stored persistently in the folders you created.

Command #18 – Creates the necessary NGINX folders if they aren't already present. NGINX will act as the reverse proxy in front of n8n.

Command #19 – Opens a new NGINX config file where you'll define how NGINX routes web requests to your n8n instance.

Command #20 – Adds reverse proxy rules to send all requests from your domain to the n8n service running on port 5678. This also forwards important headers so n8n works correctly.

Command #21 – Installs NGINX on the server so it can serve as your reverse proxy and handle SSL termination.

Command #22 – Links your NGINX site configuration into the "enabled" folder, checks the config for syntax errors, and reloads NGINX so the new settings take effect.

Command #23 – Installs Certbot and requests a free SSL/TLS certificate from Let's Encrypt for your domain. It also configures NGINX to automatically redirect HTTP to HTTPS.

Command #24 – Starts Docker in case it's not already running, making sure your containers can launch.

Command #25 – Moves back into the /srv/n8n folder where all your config files are stored.

Command #26 – Starts the n8n and PostgreSQL containers in detached mode, meaning they run in the background and keep working after you log out.

Command #27 – Shows which containers are running, their status, and their port mappings so you can confirm everything is online.

Command #28 – Opens your browser and goes to the domain you configured, now secured with HTTPS, to access your n8n instance.