

KNOWLEDGE BASE · PUBLIC DOCUMENT

Running a Go program on a VPS as a service that restarts by itself

Node.js, Python & Docker

CODE

RLX-AO-KB-1856

VERSION

v2026.10.15

COUNTRY · LANGUAGE

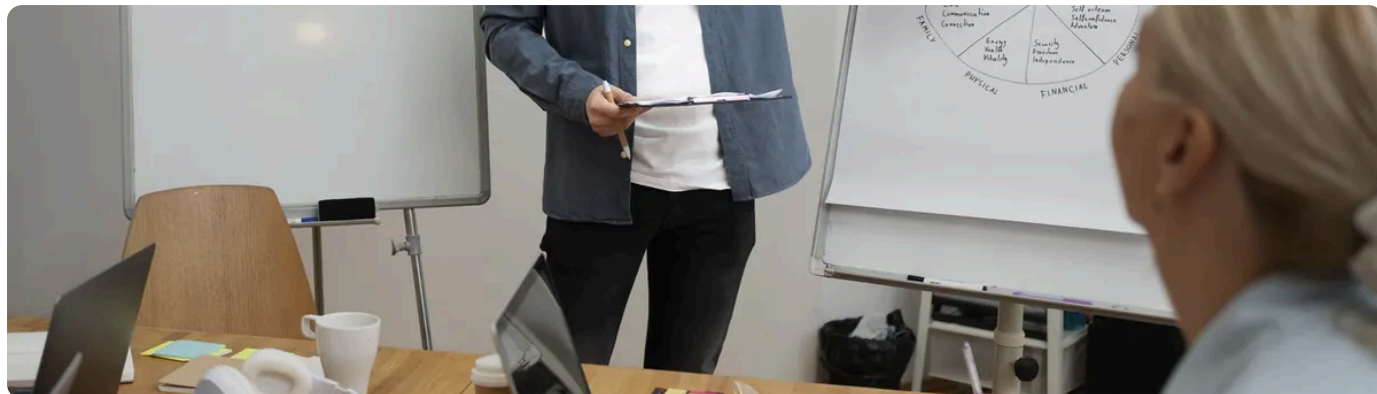
Angola · English

Interweb Angola

RLX Angola, Lda. · Group RLX Holdings

Tax ID (NIF) 500 260 5724 · Commercial registration no. 26464-25/250724

<https://www.meu.interweb.ao/knowledgebase/1856>



The short answer: a **Go** program compiles to a single file, with nothing to install on the server. On a VPS, the right path is: compile, copy the file, and run it as a **system service (systemd)**, which starts it with the server and restarts it if it fails. Then a web server in front handles the address and https.

1. Compile for Linux

On your computer (Windows, Mac or Linux), in the project folder:

```
GOOS=linux GOARCH=amd64 go build -o myapp .
```

This creates the file `myapp` for a 64-bit Linux server. On Windows, set the two variables before running the command, according to your terminal.

2. Copy and prepare

1

Upload the file to the VPS: **SSH shortcuts and scp**. Put it in a folder like `/opt/myapp/`.

2

Create a powerless user to run it (`sudo useradd --system --no-create-home myapp`) and give it the file and the folder.

3

Make the program listen only on `127.0.0.1` on a port (for example, 8080): whoever exposes it to the internet is the web server, not it.

4

Give it execute permission: `sudo chmod +x /opt/myapp/myapp .`

3. The service (systemd)

Create the file `/etc/systemd/system/myapp.service` with:

[Unit]

Description=My Go program

After=network.target

[Service]

User=myapp

WorkingDirectory=/opt/myapp

ExecStart=/opt/myapp/myapp

Restart=on-failure

Environment=PORT=8080

[Install]

WantedBy=multi-user.target

To	Command
Read the new file	<code>sudo systemctl daemon-reload</code>
Start now and at every server restart	<code>sudo systemctl enable --now myapp</code>
See whether it is running	<code>sudo systemctl status myapp</code>
See what it writes (the logs)	<code>sudo journalctl -u myapp -f</code>
After uploading a new version	<code>sudo systemctl restart myapp</code>

If the service does not start, `status` says why: **A service that will not start.**

4. Showing it on the internet

Put nginx in front, passing your domain's requests to `127.0.0.1:8080` : **Nginx as a reverse proxy**, and the certificate: **HTTPS for a container**. Open only ports 80 and 443 in the firewall: **The ufw firewall**.

Do not run the program as administrator (root) or leave it listening directly to the internet. And do not put passwords in the code: use environment variables in the service file, with closed permissions (`EnvironmentFile=`). See **Keeping passwords out of your code**.

The pattern is the same for a program in Rust, .NET or anything else that produces an executable. For one that runs in a terminal window and ends when you close it, see **tmux and screen**, which is the answer for one-off tasks, not for a service.

Want help putting a program of yours on a VPS? Open a ticket saying the language, the port and the domain.

[Open a support ticket](#)

SEE ALSO

[A service will not start on my VPS: reading systemctl status](#)

[Nginx as a reverse proxy in front of a container](#)

[FastAPI on a VPS: uvicorn workers, proxy headers and a service that restarts](#)

[SSH shortcuts: the config file, and copying files with scp](#)



Online version of this document

Interweb is a leading internet services company, with a mission to help businesses to be born and prosper through technology.

2020

FOUNDED

37,000+

CLIENTS

130+

SIGN-UPS A DAY

Interweb Angola

RLX Angola, Lda. • Group RLX Holdings

TAX ID (NIF)

500 260 5724

COMMERCIAL
REGISTRATION NO.

26464-25/250724

REGISTERED OFFICE

Rua Direita do Patriota, Bairro Dangereux
Belas, Luanda, Angola

CONTACT

info@interweb.ao • +244 942 011 191 • www.interweb.ao

OPENING HOURS

Monday to Friday - 08 AM to 04 PM
Saturday & Sunday - Closed