COMPLETE SERVER WORKFLOW GUIDE

Windows → Linux VPS → Deploy / Update Projects

PART 1 — Connect From Windows

1. Open Command Prompt (Windows)

Press:

```
Windows + R
```

Type:

```
cmd
```

Press Enter.

You will see:

```
C:\Users\YourName>
```

This is your local machine terminal.

---

2. Connect to Server Using SSH

Command:

```
ssh root@SERVER_IP
```

Example:

```
ssh root@172.252.13.97
```

What this does

- Opens a secure remote terminal.
- You now control the Linux server.

You will be asked for password.

After login:

```
root@linux:~#
```

You are now inside the server.

PART 2 — Understand Server Folder Structure

Linux main directories:

```
/
├── home        ← user folders (where projects should live)
├── etc         ← system configs
├── var         ← logs
└── root        ← root user's private folder
```

Projects should NOT be stored in `/root`.

Go to user home directory

```
cd /home
```

Check users:

```
ls
```

Example output:

```
tania  arif  salman
```

Enter your user folder

```
cd tania
```

Now you are here:

```
/home/tania
```

PART 3 — Create Project Workspace

Create a dedicated projects directory.

```
mkdir projects
```

Go inside:

```
cd projects
```

Check location:

```
pwd
```

Output:

```
/home/tania/projects
```

Recommended structure

```
/home/tania/projects
    ├── project_one
    ├── project_two
    └── project_three
```

Each project must have its own folder.

PART 4 — Deploy a NEW Repository

1. Clone repository from GitHub

Command:

```
git clone REPOSITORY_URL
```

Example:

```
git clone
https://github.com/taniajasmin/Twilio-Call-Bridge-FastAPI.git
```

Now check:

```
ls
```

You will see:

```
Twilio-Call-Bridge-FastAPI
```

---

2. Enter project folder

```
cd Twilio-Call-Bridge-FastAPI
```

Structure example:

```
Twilio-Call-Bridge-FastAPI
├── main.py
├── requirements.txt
├── .env
└── README.md
```

PART 5 — Prepare Python Environment

Install virtual environment tool (one-time)

```
sudo apt install python3-venv -y
```

Create virtual environment

```
python3 -m venv venv
```

New structure:

```
Twilio-Call-Bridge-FastAPI
├── venv/
├── main.py
└── requirements.txt
```

Activate environment

```
source venv/bin/activate
```

Prompt changes:

```
(venv) root@linux:...
```

Install dependencies

```
pip install -r requirements.txt
```

What this does:

- Installs all libraries required by the project.

PART 6 — Test Application Manually

Run server:

```
uvicorn main:app --host 0.0.0.0 --port 8002
```

Open browser:

```
http://SERVER_IP:8002/docs
```

If Swagger UI appears → application works.

Stop server:

```
CTRL + C
```

PART 7 — Run App as Background Service

We use **systemd** so app runs permanently.

Create service file

```
sudo nano /etc/systemd/system/project-name.service
```

Example:

```
[Unit]
Description=FastAPI Application
After=network.target

[Service]
User=tania
WorkingDirectory=/home/tania/projects/Twilio-Call-Bridge-FastAPI
ExecStart=/home/tania/projects/Twilio-Call-Bridge-FastAPI/venv/bin/u
vicorn main:app --host 0.0.0.0 --port 8002
Restart=always
```

```
[Install]
WantedBy=multi-user.target
```

Save:

```
CTRL + O
ENTER
CTRL + X
```

Start service

```
sudo systemctl daemon-reload
sudo systemctl start project-name
sudo systemctl enable project-name
```

Check status

```
sudo systemctl status project-name
```

Expected:

```
Active: active (running)
```

Press `q` to exit.

PART 8 — Updating an EXISTING Project

Whenever GitHub code changes:

Go to project

```
cd /home/tania/projects/Twilio-Call-Bridge-FastAPI
```

Fix git ownership (only once)

```
git config --global --add safe.directory
/home/tania/projects/Twilio-Call-Bridge-FastAPI
```

Pull latest code

```
git pull origin main
```

Activate environment

```
source venv/bin/activate
```

Install new packages

```
pip install -r requirements.txt
```

Restart service

```
sudo systemctl restart project-name
```

PART 9 — View Logs

Live logs:

```
journalctl -u project-name -f
```

Exit:

```
CTRL + C
```

PART 10 — Deploy Multiple Projects

Each project must have:

- separate folder
- separate virtual environment
- separate port
- separate service

Example:

```
projects/
├── ai_agent        → port 8000
├── rag_system      → port 8001
└── call_bridge     → port 8002
```

PART 11 — Useful Commands

Check running services:

```
systemctl list-units --type=service
```

Restart service:

```
sudo systemctl restart SERVICE_NAME
```

Check open ports:

```
ss -tulnp
```

FINAL FLOW SUMMARY

```
Windows CMD
    ↓
```

```
SSH login
   ↓
/home/user/projects
   ↓
git clone / git pull
   ↓
venv + pip install
   ↓
systemd service
   ↓
FastAPI running continuously
```