Full Deployment, Server Management, and Docker Guide

1. Localhost vs Server

- On localhost you run multiple apps using unique ports.

- On a real server you do EXACTLY the same, but use Nginx to map domain/subdomain → port.

- Server acts like remote localhost accessible via SSH.

2. Accessing Server

- Use SSH (VS Code Remote SSH, PowerShell, PuTTY).

- Once connected, server terminal behaves like your local machine.

- You can create folders, pull GitHub repo, install dependencies, run builds, etc.

3. Multi-Environment Setup (Dev, QA, Prod)

- Create folders:

/var/www/dev

/var/www/qa

/var/www/prod

- Each has own backend, frontend, .env, DB.

- Each runs on unique PORT.

- Nginx routes:

dev.xyz.com → localhost:4000

qa.xyz.com → localhost:5000

xyz.com → localhost:6000

4. Database (MySQL, MongoDB, PostgreSQL)

- You create multiple DBs:

db_dev, db_qa, db_prod

- Each environment uses its own DB.

- Imported using MySQL Workbench or CLI.

- Same concept for any database type.

5. Deployment Flow (Without Docker)

- SSH into server

- git pull

- npm install

- npm run build

- pm2 restart

- Nginx reverse-proxy handles domain routing

6. Nginx Explained

- Nginx checks server_name and forwards to the correct backend port.

- Example:

server_name dev.xyz.com;

proxy_pass http://localhost:4000;

7. Adding New Domains/Subdomains

- Create new folders

- Create unique port + DB

- Add new Nginx config

- DNS → A record → server IP

8. Docker Concept

- Each project runs inside a container.

- Containers have their own OS layer, dependencies, ports, and runtimes.

- Example:

P1 uses Node 18 → Container C1

P2 uses Node 24 → Container C2

9. Docker + Nginx Flow

User → DNS → Server → Nginx → Correct Docker container

10. Docker Benefits

- No version conflicts
- Clean server
- Easy migration
- Same environment in dev/qa/prod
- Containers are isolated
- Can run Node 18, Node 20, Python 3.12, PHP 8 all on the same server

11. docker-compose Architecture

Defines:

- Backend container
- Frontend container
- Database container
- Nginx container (optional)
- Networks
- Volumes

12. CI/CD Automation

Almost everything can be automated using GitHub Actions:

- Pull code
- Build frontend/backend
- Deploy to server
- Restart PM2 or Docker containers
- Run tests

- Send notifications

13. What More You Should Know

- Load balancing

- Horizontal scaling

- Docker networks & volumes

- Health checks

- Monitoring (Grafana, Prometheus)

- Backup strategies

- Firewall & SSH security

- Secrets management

- Infrastructure as Code (Ansible, Terraform)

- Zero downtime deployment

- Logging systems (ELK, Loki)

You now understand complete full-stack deployment and DevOps foundation:

Localhost → Server → Nginx → Docker → CI/CD → Scaling → Security.