howest
university of applied sciences

Lecture V
A DevOps Journey with Laravel

From "It Works on My Machine"
to Production Deployment
Guy Van Eeckhout
2025-2026

(inspired by Alex Desmedt)

Meet TaskMaster PRO

- A Laravel Based task Management app

- Team collaboration features
- Real-time updates
- User authentication
- Database-driven architecture
- RESTful API endpoints

YAY ! WE'RE DONE !

After several weeks of hard coding (sic)

It works ! WOOHOO !

So can we put it in Production ?

Hello ? Anyone ?

THE DEPLOYMENT CHALLENGE

- Why does it work perfectly on your laptop (AND ONLY THERE) ?
- What happens when users need to access it?
- How do we bridge the gap between development and production?
- What infrastructure do we need?
- What are the different hosting options?

Bu it works on my machine

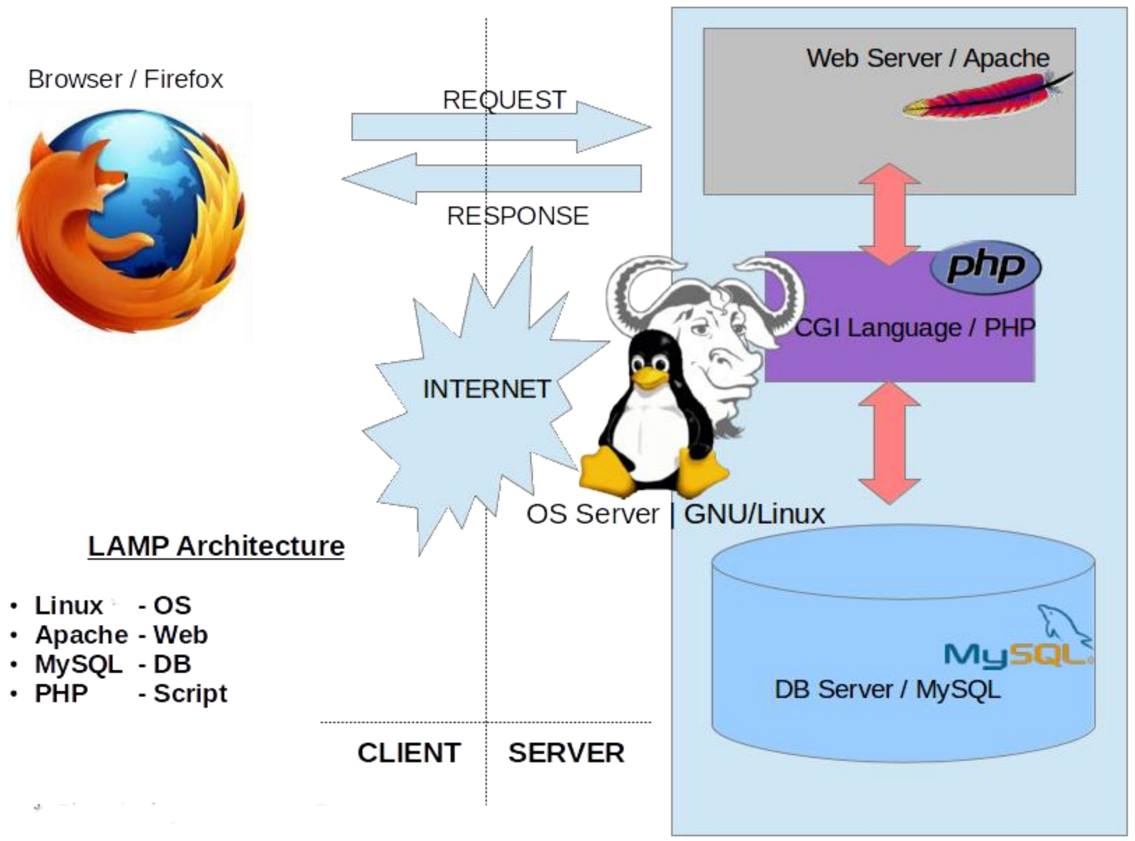
The classic developer phrase everyone has heard (or said!)

- Development environment perfectly configured for YOU
- All dependencies installed locally
- Database seeded with test data
- No network constraints
- Debug tools readily available
- Single user (you!)

❓ But what happens when 1,000 users need to access it simultaneously?

TASKMASTER PRO'S ECOSYSTEM

- PHP 8.x installed and configured
- Composer for dependency management
- Local MySQL/MariaDB instance running
- Built-in PHP development server (php artisan serve)
- Configured .env file with local settings
- Storage directories with proper write permissions
- Debugging tools (Xdebug, Laravel Telescope)

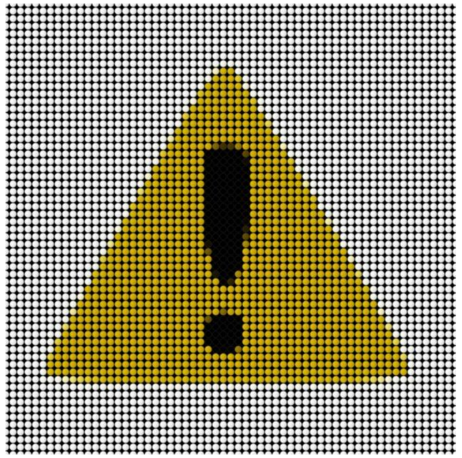


NEWS FLASH : LOCAL ≠ PRODUCTION

- Operating system variations (Windows dev → Linux production)
- Different PHP versions or missing extensions
- Database configuration differences
- File permission models (Windows vs Linux)
- Network security constraints and firewalls
- Resource availability (CPU, RAM, bandwidth)
- Concurrent user handling capabilities
- Error visibility and logging approaches

Reality Check : what could go wrong if you “just” deploy ?

- Missing PHP extensions
- File permission errors (storage/ not writable)
- Database connection failures
- Configuration mismatches (.env not configured correctly)
- Performance issues under load
- Security vulnerabilities exposed (debug mode on!)
- Dependency version conflicts



WHAT DOES 'PRODUCTION' MEAN?

- Accessible to real users 24/7/365
- Must handle concurrent requests efficiently
- Requires security hardening
- Needs monitoring and logging systems
- Demands reliability and high uptime (99.9%+)
- Subject to backup and recovery procedures
- Requires professional infrastructure
- Must scale with user growth

DEPLOYMENT PREPARATION CHECKLIST

e.g. Before deploying TaskMaster Pro:

☐ Document all dependencies (check composer.json)

☐ Review application requirements (PHP version, extensions)

☐ Plan database migration strategy

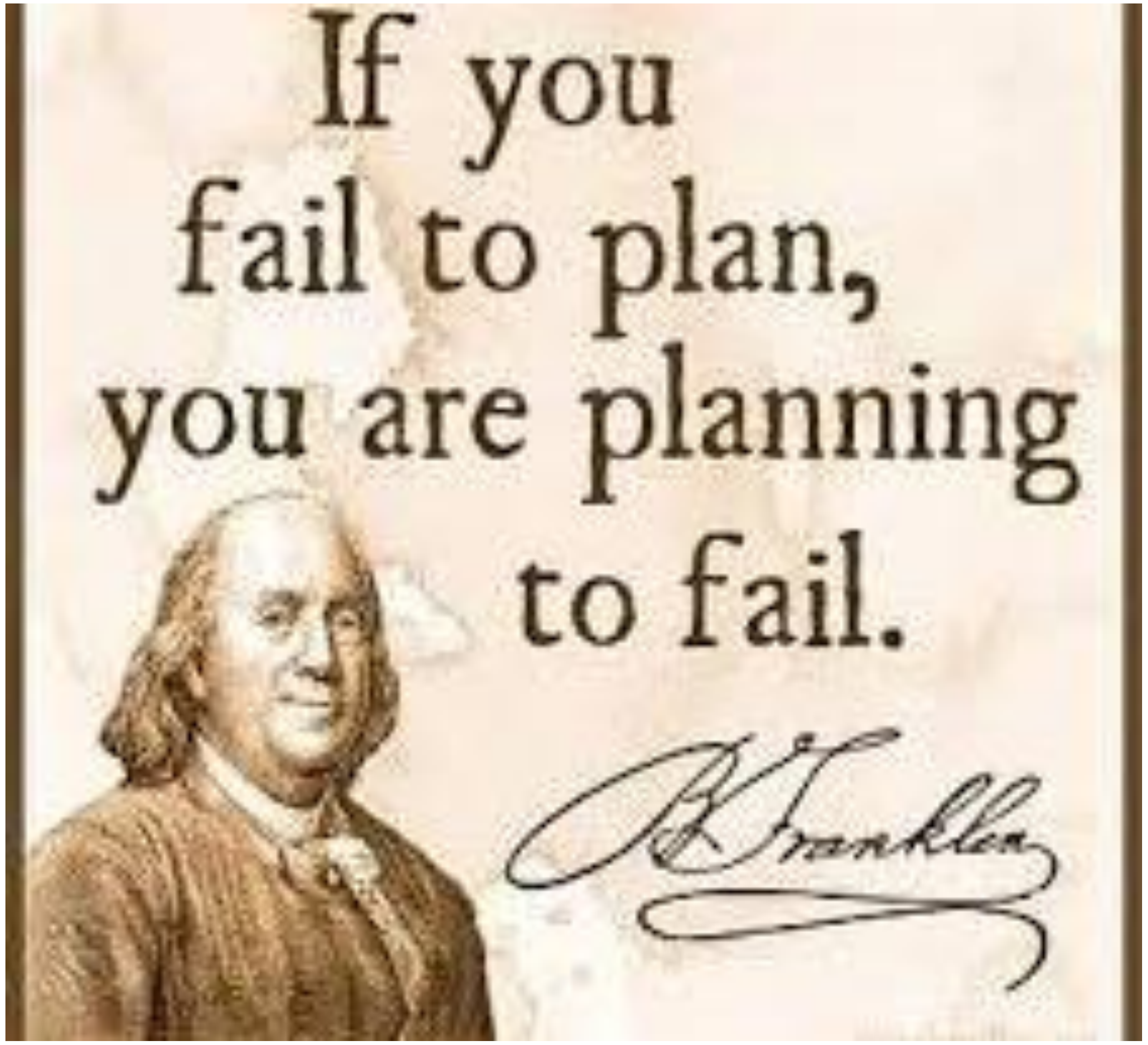
☐ Configure environment variables (.env for production)

☐ Set up error logging (logs, not browser!)

☐ Prepare rollback procedure

☐ Test in staging environment

☐ Security audit (disable debug mode!)



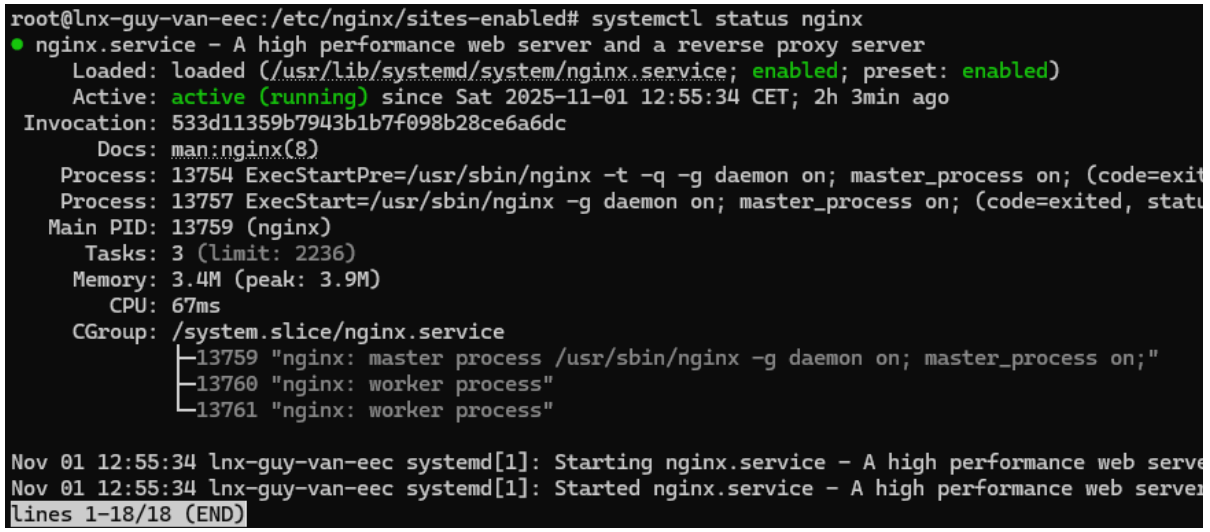
From Code to Users: The Deployment Pipeline

1. Source code of TaskMaster Pro in version control (Git)
2. Transfer to production server (git clone)
3. Install dependencies (composer install)
4. Configure environment (.env setup)
5. Run database migrations (php artisan migrate)
6. Set proper permissions (www-data ownership)
7. Configure web server (Nginx/Apache)
8. Test and monitor (verify everything works)

INITIAL BASIC CONFIGURATION

System setup fundamentals :

- User privileges and sudo access (security best practice)
- SSH for remote administration (no more console!)
- Basic service management (systemd/systemctl)
- Security hardening (firewall, fail2ban)
- Firewall configuration (only needed ports open)



```
root@lnx-guy-van-eec:/etc/nginx/sites-enabled# systemctl status nginx
● nginx.service - A high performance web server and a reverse proxy server
     Loaded: loaded (/usr/lib/systemd/system/nginx.service; enabled; preset: enabled)
     Active: active (running) since Sat 2025-11-01 12:55:34 CET; 2h 3min ago
 Invocation: 533d11359b7943b1b7f098b28ce6a6dc
       Docs: man:nginx(8)
    Process: 13754 ExecStartPre=/usr/sbin/nginx -t -q -g daemon on; master_process on; (code=exit
    Process: 13757 ExecStart=/usr/sbin/nginx -g daemon on; master_process on; (code=exited, statu
   Main PID: 13759 (nginx)
      Tasks: 3 (limit: 2236)
     Memory: 3.4M (peak: 3.9M)
        CPU: 67ms
     CGroup: /system.slice/nginx.service
             ├─13759 "nginx: master process /usr/sbin/nginx -g daemon on; master_process on;"
             ├─13760 "nginx: worker process"
             └─13761 "nginx: worker process"

Nov 01 12:55:34 lnx-guy-van-eec systemd[1]: Starting nginx.service - A high performance web serve
Nov 01 12:55:34 lnx-guy-van-eec systemd[1]: Started nginx.service - A high performance web server
lines 1-18/18 (END)
```

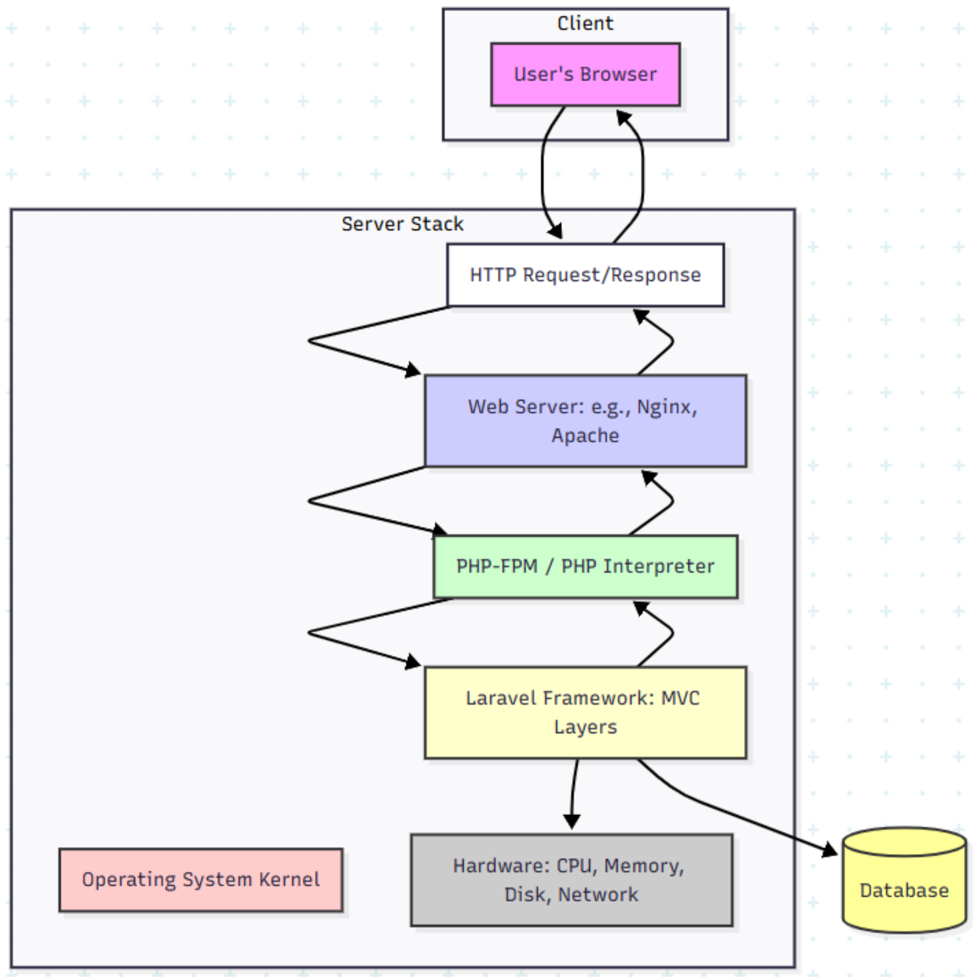
💭 Why is SSH better than working in the VMware console?

BUILDING THE SOFTWARE STACK

What does TaskMaster Pro need to run in production?

- Operating System (Linux - Debian/Ubuntu)
- Web Server (Nginx or Apache)
- PHP Runtime with required extensions
- Database Server (MySQL/MariaDB)
- Process manager for PHP (PHP-FPM)
- Supporting tools (Git, Composer, etc.)

Each component plays a specific role in the architecture.



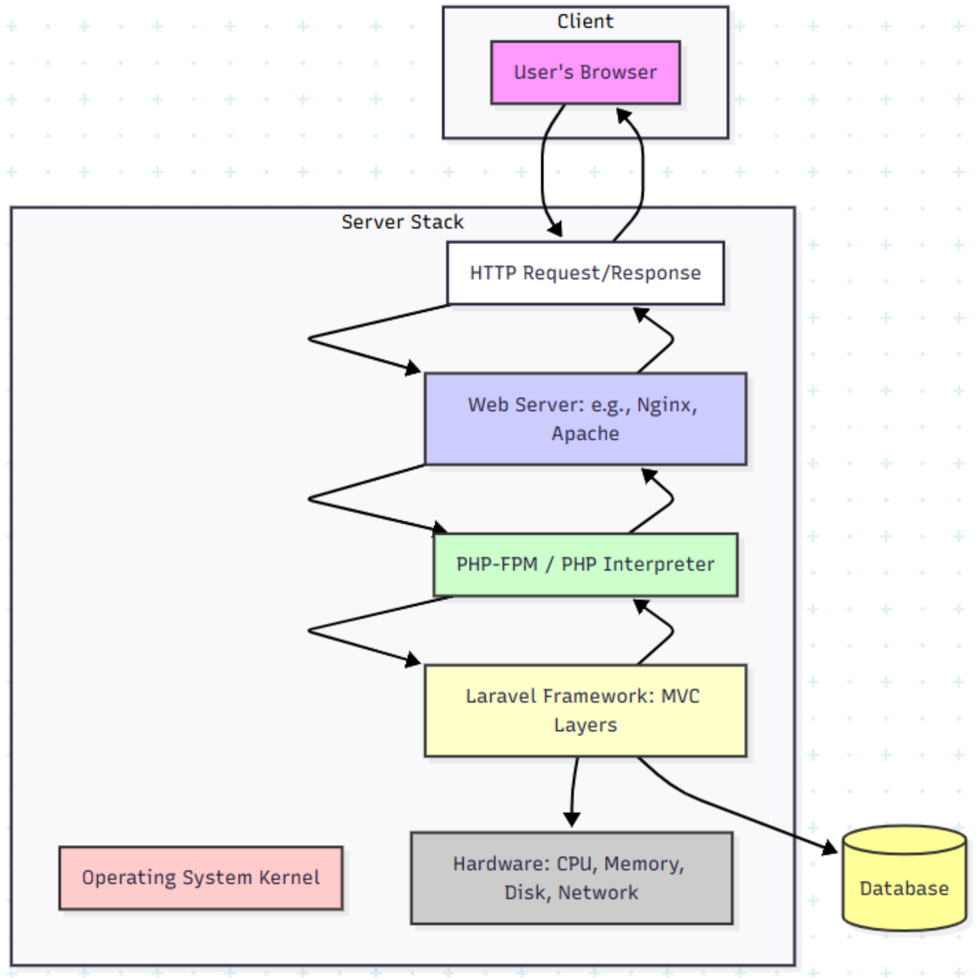
Client
User's Browser
Server Stack
HTTP Request/Response
Web Server: e.g., Nginx, Apache
PHP-FPM / PHP Interpreter
Laravel Framework: MVC Layers
Operating System Kernel
Hardware: CPU, Memory, Disk, Network
Database

WEB SERVER - THE FRONT DOOR

- Listen for HTTP/HTTPS requests on ports 80/443
- Serve static files (CSS, JS, images) directly
- Forward dynamic requests to PHP-FPM
- Handle SSL/TLS certificates for HTTPS
- Manage virtual hosts (multiple sites on one server)
- Log access and errors for monitoring

The web server is the first point of contact for all requests.

💭 What process manages nginx?

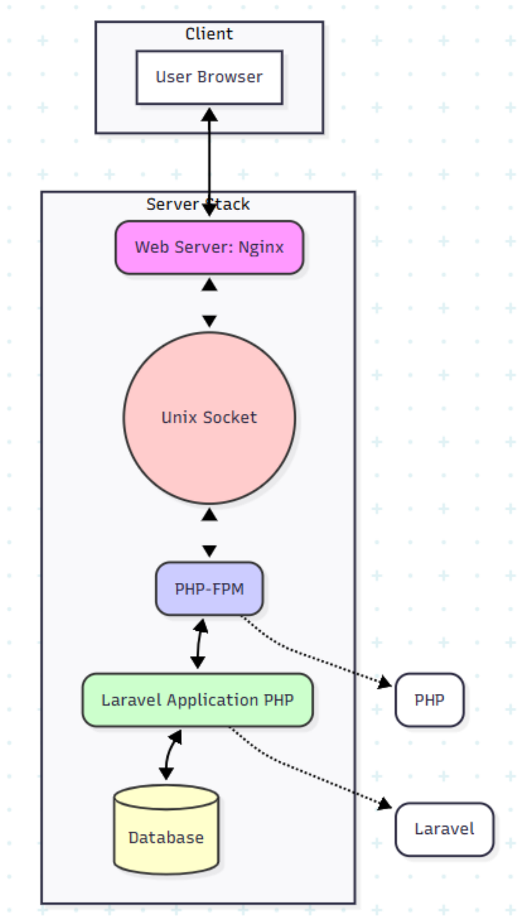

Client
User's Browser
Server Stack
HTTP Request/Response
Web Server: e.g., Nginx, Apache
PHP-FPM / PHP Interpreter
Laravel Framework: MVC Layers
Operating System Kernel
Hardware: CPU, Memory, Disk, Network
Database

PHP-FPM - DYNAMIC CONTENT PROCESSING

FastCGI Process Manager (PHP-FPM):

- Separates PHP processing from web server
- Pool-based resource management
- Better performance under load
- Process lifecycle control

💭 What connects Nginx to PHP-FPM?

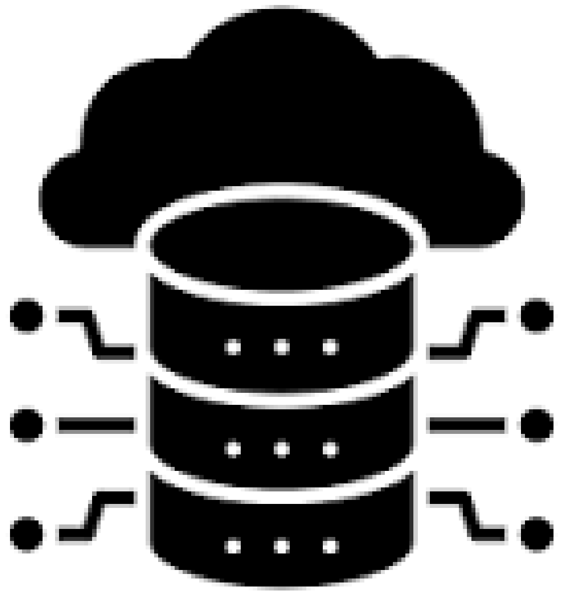


DATABASE SERVER - THE DATA FOUNDATION

Database layer considerations:

- MySQL/MariaDB installation
- User authentication and privileges setup
- Database creation with proper permissions
- Connection configuration (host, port, credentials)
- Performance tuning and optimization

🔍 How does Laravel know how to connect?



MANAGING SOFTWARE STACKS

Stack management approaches:

- Manual installation (apt packages)
- Configuration management (Ansible, Puppet)
- Containerization (Docker – Here it is!)
- Platform-as-a-Service solutions (Heroku, Laravel Forge)
- Managed hosting services

🔍 install with apt, manage with systemctl commands(systemctl status, start, stop, restart, enable)

BARE METAL SERVER (recap)

Pros:

✓ Direct hardware access - maximum performance

✓ Single-tenant security - no noisy neighbors

✓ Full customization of hardware and software

✓ Predictable resource availability

Cons:

✗ Limited flexibility - can't easily resize

✗ Difficult to scale quickly

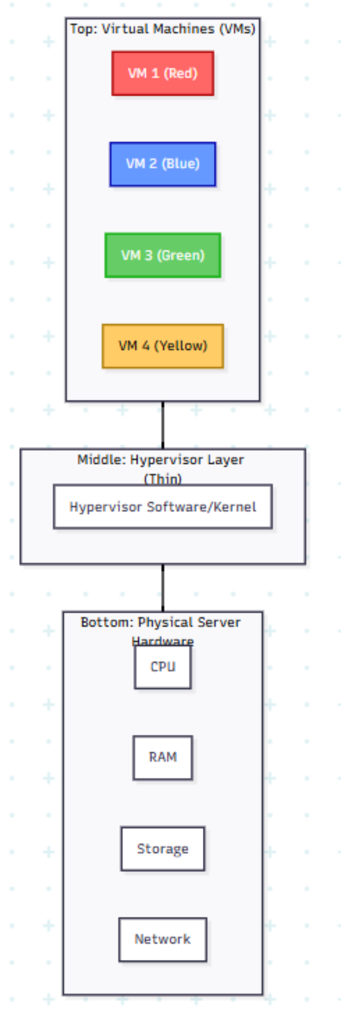
✗ Higher upfront cost

✗ Manual hardware maintenance required

Best for: High-performance computing, compliance requirements, dedicated workloads

Virtualization (recap)

- Hypervisor layer abstracts physical hardware
- Multiple VMs run on one physical server
- Each VM has its own complete OS and resources
- Software-defined infrastructure
- Snapshot and cloning capabilities
- Flexible resource allocation


Top: Virtual Machines (VMs)
VM 1 (Red)
VM 2 (Blue)
VM 3 (Green)
VM 4 (Yellow)
Middle: Hypervisor Layer (Thin)
Hypervisor Software/Kernel
Bottom: Physical Server Hardware
CPU
RAM
Storage
Network

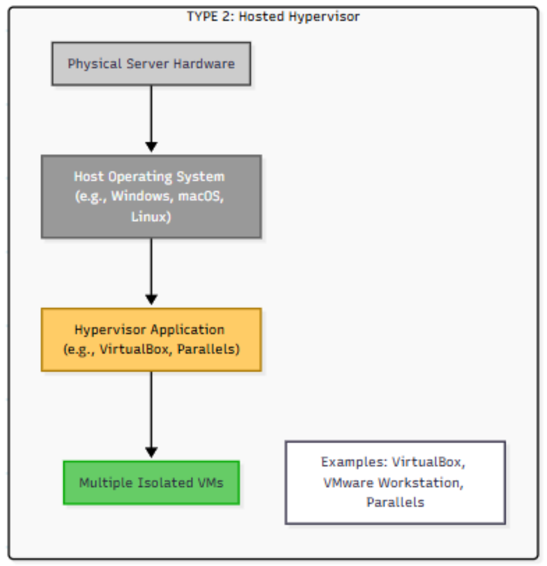
TYPE 1 VS TYPE 2 HYPERVISORS (recap)

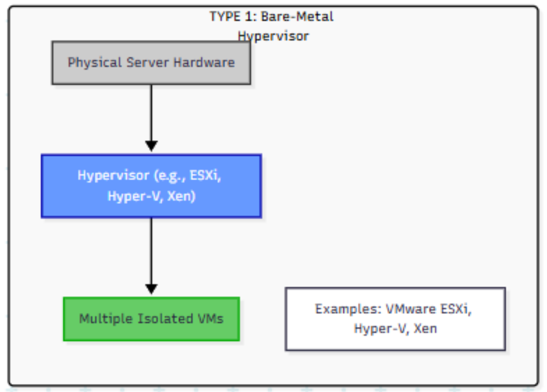
TYPE 1 (Bare Metal Hypervisors):

- Runs directly on hardware
- Examples: VMware ESXi, Microsoft Hyper-V, KVM
- Better performance (less overhead)
- Used in production environments

TYPE 2 (Hosted Hypervisors):

- Runs on top of host OS
- Examples: VMware Workstation, VirtualBox
- Easier for development and testing
- More overhead (OS + Hypervisor + Guest OS)


TYPE 2: Hosted Hypervisor
Physical Server Hardware
Host Operating System (e.g., Windows, macOS, Linux)
Hypervisor Application (e.g., VirtualBox, Parallels)
Multiple Isolated VMs
Examples: VirtualBox, VMware Workstation, Parallels


TYPE 1: Bare-Metal Hypervisor
Physical Server Hardware
Hypervisor (e.g., ESXi, Hyper-V, Xen)
Multiple Isolated VMs
Examples: VMware ESXi, Hyper-V, Xen

VIRTUALIZATION BENEFITS (yes, more recap)

Why virtualize?

- Server consolidation (better resource utilization: 70-80% vs 15-20%)
- Rapid provisioning (minutes vs days/weeks)
- Easy backup and recovery (snapshots!)
- Testing and development isolation
- Cost efficiency (multiple VMs per physical server)
- Disaster recovery capabilities
- Flexibility to scale up/down
- Hardware independence (move VMs between servers)

LOCAL DEVELOPMENT WITH VMWARE WORKSTATION

- Type 2 hypervisor running on your laptop
- Better than local development :

  - Development and testing of TaskMaster Pro on a VM !

  - Isolated
- Local network simulation
- Snapshot capabilities for experimentation
- Resource constraints of laptop hardware
- Perfect for learning and testing in a near-production env !

VMWARE ESXI - PRODUCTION-GRADE HYPERVISOR (final recap, I swear !)

- Type 1 bare-metal hypervisor
- Runs directly on server hardware
- Minimal footprint (~130MB installation)
- Enterprise management via vCenter
- High availability and fault tolerance features
- Production-ready infrastructure
- Supports up to 768 CPUs and 24TB RAM per host
- Used by enterprises worldwide

Moving TaskMaster Pro to the cloud

1. Export VM from Workstation (OVF/OVA format)
2. Convert format if needed
3. Upload to cloud provider (ESXi-based infrastructure)
4. Configure networking and storage
5. Assign public IP address
6. Configure DNS records
7. Deploy TaskMaster Pro!
8. You’re in the cloud !

THE CONTAINER REVOLUTION (recap, but also look ahead)

VIRTUAL MACHINES:	CONTAINERS:
• Full OS per instance (~GB in size)	• Shared OS kernel (~MB in size)
• Larger resource footprint	• Minimal resource usage
• Minutes to start	• Seconds to start
• Hardware-level isolation	• Process-level isolation
• Can run different OS on same host	• Must use same OS as host

Today’s LAB

- Setting up a production-ready Debian VM (no cloud today ☺)
- Configuring each layer of the software stack
- Understanding service dependencies (systemd)
- Installing and configuring Nginx
- Setting up PHP-FPM for dynamic content
- Database configuration (MariaDB)
- Deploying an existing Laravel application
- Troubleshooting deployment issues
- Best practices for security and permissions

Remember: Every production deployment started with someone asking "How do I get this to work for real users?"