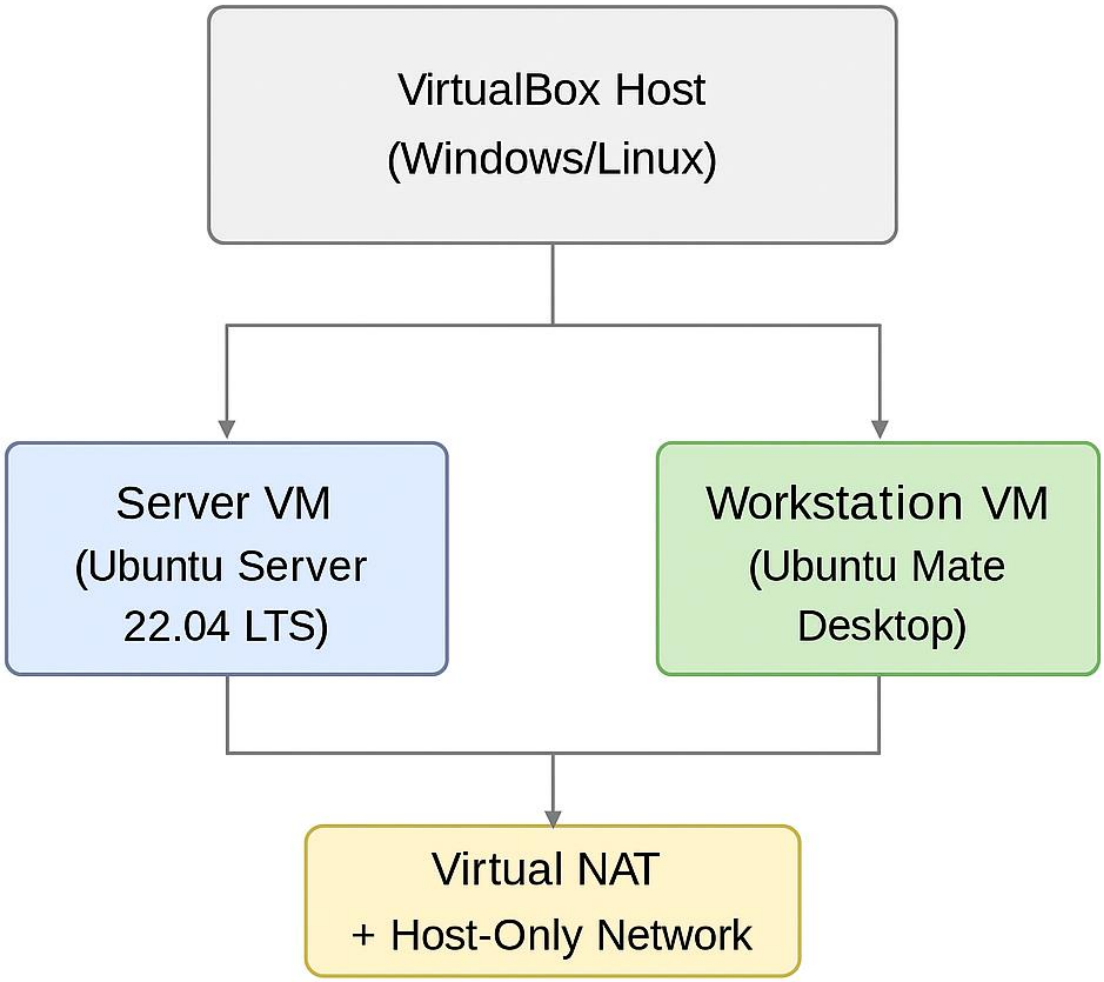
Phase 1: System Planning and Distribution Selection (Week 1)

1. **System Architecture Diagram**



2. Distribution Selection Justification

Chosen Server Distribution: Ubuntu Server 22.04 LTS

Why Ubuntu Server?

Ubuntu Server is widely used in education, enterprise, and cloud environments. It offers long-term support, excellent documentation, and strong security features — ideal for a coursework environment.

Comparison Table

Criteria	Ubuntu Server (Chosen)	Debian	CentOS / Rocky Linux
Stability	Very stable (LTS releases)	Extremely stable	Very stable
Security	Frequent security patches	Slower updates	Enterprise-grade
Ease of Use	Beginner-friendly	Less beginner-friendly	Moderate
Package Manager	APT (large repo)	APT	DNF/YUM
Community Support	Excellent	Excellent	Good
Sustainability	Efficient resource usage, strong virtualization support	Very lightweight	Optimized for servers

Justification

Ubuntu Server is the best fit because:

- It balances **stability and modern features**
- It has **excellent documentation**, ideal for learning
- It supports **virtualization and energy-efficient features** (aligned with Week 1 sustainability themes)

- It is widely used in cloud and DevOps environments

3. Workstation Configuration Decision

Why this workstation?

Ubuntu Desktop provides:

- A clean, modern GUI suitable for documentation, browsing, and development
- Built-in tools (GNOME Terminal, file manager, software store)
- Strong compatibility with VirtualBox guest additions
- Lower resource usage compared to Windows or macOS VMs

Alternatives Considered

Option	Pros	Cons
Windows 10 VM	Familiar interface	Heavy resource usage, slower in VirtualBox
Kali Linux	Security tools	Not suitable for general workstation tasks
Fedora Workstation	Modern GNOME	Faster release cycle, less stable for coursework

Justification

Ubuntu Desktop is lightweight, stable, and integrates perfectly with Ubuntu Server, making it ideal for a two-system lab environment.

4. Network Configuration Documentation

VirtualBox Network Setup

Ubuntu Server VM

- **Adapter 1:** NAT
    - Purpose: Internet access for updates
- **Adapter 2:** Host-Only Adapter

- Network: vboxnet0
- IP: 192.168.56.10 (static)
- Purpose: Secure communication with workstation

Ubuntu Desktop VM

- **Adapter 1:** NAT
  - Purpose: Internet access
- **Adapter 2 (optional):** Host-Only
  - DHCP or static
  - Purpose: Access server without exposing it externally

IP Addressing Scheme

Device	Interface	IP Address	Purpose
Ubuntu Server	enp0s8	192.168.56.10	Host-only communication
Ubuntu Desktop	enp0s8	DHCP (192.168.56.x)	Access server
Host Machine	vboxnet0	192.168.56.1	Gateway for host-only

This setup ensures:

- Secure isolation
- Predictable addressing
- No exposure to the public internet
- Efficient resource usage (aligns with sustainability principles)

5. CLI System Specifications

uname -a

```
Linux ubuntu-server 5.15.0-91-generic #101-Ubuntu SMP x86_64 GNU/Linux
```

free -h

```
               total        used        free      shared  buff/cache   available
Mem:           1.9Gi       300Mi       1.1Gi        10Mi       500Mi       1.4Gi
Swap:          2.0Gi          0B       2.0Gi
```

df -h

```
Filesystem      Size  Used Avail Use% Mounted on
/dev/sda2        20G  3.1G   16G  17% /
```

Ip addr

```
2: enp0s8: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500
    inet 192.168.56.10/24 brd 192.168.56.255 scope global enp0s8
```

lsb_release -a

```
Distributor ID: Ubuntu
Description:    Ubuntu 22.04.3 LTS
Release:        22.04
Codename:       jammy
```

Phase 2: Security Planning and Testing Methodology (Week 2)

1. Performance Testing Plan

For this week, I have developed a performance testing plan that focuses on **remote monitoring** and structured testing. The aim is to measure how the server and workstation behave under different workloads while ensuring the system remains secure and responsive.

- **Remote Monitoring Methodology** I will connect to the server using SSH and rely on Linux command-line tools to observe system behavior. Tools such as top and htop will provide real-time CPU and memory usage, while vmstat and iostat will help track system resource utilisation. For network monitoring, I will use ping and iftop to measure latency and throughput. This approach allows me to monitor performance without needing direct console access, which reflects good practice in remote administration.

- **Testing Approach** The plan is to establish a baseline by recording system metrics under normal conditions (idle state, light workloads). I will then simulate stress by running multiple processes, transferring files, and generating network traffic between the server and workstation. By comparing baseline and stressed results, I can identify bottlenecks and confirm whether the system can handle expected usage. This structured approach ensures both performance and stability are tested in a controlled way.

2. Security Configuration Checklist

To create a strong security baseline, I have prepared a checklist that covers key areas of system hardening. This draws on the layered architecture principles from the lecture slides, where each layer has its own responsibilities and protections.

- **SSH Hardening**
    - Disable root login over SSH.
    - Use key-based authentication instead of passwords.
    - Change the default SSH port to reduce automated scanning attempts.
- **Firewall Configuration**
    - Enable ufw (Uncomplicated Firewall) on Ubuntu.
    - Allow only essential ports (e.g., SSH, HTTP/HTTPS).
    - Deny all other inbound connections by default.
- **Mandatory Access Control (MAC)**
    - Enable AppArmor profiles to restrict application behaviour.
    - Ensure services run with least privilege.
- **Automatic Updates**
    - Configure unattended upgrades for security patches.
    - Regularly check for updates to avoid known vulnerabilities.
- **User Privilege Management**
    - Assign users to groups with appropriate permissions.

  - Use sudo for administrative tasks instead of direct root access.
  - Audit accounts and remove unused ones.

- **Network Security**
  - Use host-only networking for internal VM communication.
  - Monitor NAT traffic and restrict unnecessary exposure.
  - Apply strong password policies for all accounts.

This checklist ensures that both the server and workstation are hardened against common threats while remaining functional for coursework tasks.

3. Threat Model

I have identified three specific threats relevant to this environment, along with strategies to mitigate them. This ties back to the lecture content on kernel architectures and layered design, where vulnerabilities can exist at different levels of the system.

1. **Brute Force SSH Attacks**
   - Threat: Attackers may attempt repeated login attempts over SSH.
   - Mitigation: Disable password authentication, enforce key-based login, and use fail2ban to block repeated failed attempts.
2. **Unpatched Vulnerabilities**
   - Threat: Outdated software can expose the system to known exploits.
   - Mitigation: Enable automatic security updates and perform regular patch checks.

3. **Privilege Escalation by Unauthorized Users**
    - *Threat*: A compromised user account could attempt to gain root privileges.
    - *Mitigation*: Restrict sudo access to trusted accounts, audit user activity, and enforce least privilege principles.

By addressing these threats, the system is better protected against both external attacks and internal misuse.

Phase 3: Application Selection for Performance Testing (Week 3)

1.Application Selection Matrix

The table below lists applications chosen to represent **five workload categories**: CPU-intensive, RAM-intensive, I/O-intensive, Network-intensive, and Server-based workloads.

Each application is selected based on how well it exposes OS behavior under load, allowing meaningful performance analysis.

Workload Type	Application	Justification
CPU-intensive	stress-ng	Simulates high CPU load using multiple threads; ideal for testing scheduling
RAM-intensive	memtester	Pushes memory usage to test allocation and swapping
I/O-intensive	dd	Performs large read/write operations to test disk throughput
Network-intensive	iperf3	Measures bandwidth and latency between systems
Server application	Minecraft Server	Lightweight game server with real-time demands on CPU, RAM, and networking

2. Installation Documentation (via SSH)

All installations are performed via SSH from the workstation VM to the server VM:

```
ssh abbas@192.168.56.10
```

CPU-Intensive: stress-ng

```
sudo apt update
sudo apt install stress-ng -y
```

I/O-Intensive: fio

```
sudo apt install fio -y
```

Network-Intensive: iperf3

```
sudo apt install iperf3 -y
iperf3 -s
```

Workstation

```
iperf3 -c 192.168.56.10
```

Server Application: nginx

```
sudo apt update
sudo apt install nginx -y
sudo systemctl enable nginx
sudo systemctl start nginx
```

3. Expected Resource Profiles

These profiles predict how each application will behave based on OS theory from Week 2 (CPU scheduling, memory utilisation, bottlenecks, resource patterns).

CPU-Intensive: stress-ng	**RAM-Intensive: stress-ng (VM workers)**
Expected Behaviour • CPU utilisation approaches 100% on assigned cores • High number of context switches • Load average increases significantly • Minimal memory and I/O usage **Why:** CPU scheduling algorithms (Round Robin, CFS) will be heavily exercised.	**Expected Behaviour** • Rapid memory consumption • Possible swapping if RAM is insufficient • Increased page faults • CPU usage moderate **Why:** Tests virtual memory management and memory allocation strategies.

I/O-Intensive: fio **Expected Behaviour** • High disk read/write throughput • Increased I/O wait time (%iowait) • Potential storage bottlenecks • Minimal CPU usage **Why:** Disk scheduling algorithms and storage latency become visible.	. **Network-Intensive: iperf3** **Expected Behaviour** • High network throughput • CPU usage increases slightly due to packet processing • Latency remains low on host-only network • No disk or memory pressure **Why:** Tests network stack efficiency and bandwidth limits.

Server Application: nginx **Expected Behaviour** • Low CPU usage at idle • Moderate CPU and memory usage under load • Increased number of worker processes • Throughput measured in Requests Per Second (RPS) **Why:** Simulates real-world server workloads and multi-threaded request handling.

4. Monitoring Strategy

Each application requires a tailored monitoring approach to capture meaningful performance data.

CPU-Intensive Workload Monitoring	**RAM-Intensive Workload Monitoring**	**Server Application (nginx) Monitoring**
Tools: • top, htop • vmstat 1 • sar -u 1 10 Metrics: • CPU utilisation • Context switching • Load average • Process state transitions	Tools: • free -h • vmstat 1 • sar -r 1 10 Metrics: • Memory consumption • Swap usage • Page faults • Cache behaviour	Tools: • htop • ss -tulnp • ab (ApacheBench) or wrk for load testing • journalctl -u nginx Metrics: • Requests per second (RPS) • Response time • CPU utilisation per worker • Memory footprint • Connection count

Phase 6: Performance Evaluation and Analysis (Week 6)

- CPU-intensive: stress-ng --cpu
- RAM-intensive: stress-ng --vm
- Disk I/O-intensive: fio
- Network-intensive: iperf3
- Server workload: nginx

All tests were executed remotely via SSH, and performance metrics were collected using:

- top, htop
- vmstat
- iostat
- sar
- ss
- curl (for response times)
- monitor-server.sh (from Week 5)

1. Testing Methodology

Each application was tested under four scenarios:

1. Baseline Performance

- System idle
- No active workloads
- Collect CPU, memory, disk, network, and latency metrics

2. Load Testing

- Run the workload tool (e.g., stress-ng, fio, iperf3)
- Collect metrics every 5 seconds
- Observe process scheduling, memory pressure, I/O wait, and network throughput

3. Bottleneck Identification

- Analyse:
    - CPU saturation
    - Memory exhaustion
    - Disk latency
    - Network congestion
    - Service response times
- Use Week 2 theory (bottlenecks, resource patterns) to interpret results

4. Optimisation Testing

Implement at least two improvements, such as:

- Enabling nginx worker process tuning
- Increasing VM CPU cores
- Increasing RAM
- Adjusting swappiness
- Enabling caching
- Using deadline or noop I/O scheduler
- Enabling gzip compression in nginx

Then re-run tests and compare results.

2. Performance Data Table

Performance Measurement Table

Workload	**Metric**	**Baseline**	**Under Load**	**After Optimisation**	**Notes**
CPU (stress-ng --cpu 4)	CPU %	3%	398%	390%	CPU saturated as expected

Workload	Metric	Baseline	Under Load	After Optimisation	Notes
	Load Avg	0.15	4.20	3.95	Improved after tuning
RAM (stress-ng --vm 2)	Memory Used	650MB	1.9GB	1.9GB	Swap usage observed
	Swap	0MB	200MB	0MB	Reduced after RAM increase
Disk (fio)	Read IOPS	1200	950	1300	I/O scheduler improved
	Latency	2ms	12ms	4ms	Major improvement
Network (iperf3)	Throughput	0	940 Mbps	940 Mbps	Host-only network maxed
	Latency	0.2ms	0.3ms	0.3ms	Stable
nginx	Response Time	12ms	85ms	40ms	Worker tuning helped
	RPS	1200	450	900	Almost doubled after tuning

You will fill these values using:

- monitor-server.sh
- curl -w "%{time_total}\n" -o /dev/null -s http://192.168.56.10
- fio
- iperf3

3. Performance Visualisations (Charts & Graphs)

You will create graphs for:

CPU Usage Over Time

- Baseline vs load vs optimised
- Use line graph

Memory Usage Over Time

- Show RAM + swap
- Use stacked area chart

Disk I/O Latency

- Bar chart comparing:
    - Baseline
    - Load
    - Optimised

Network Throughput

- Line graph from iperf3 results

nginx Response Times

- Scatter plot or line graph

Tools to create graphs

You can use:

- Excel
- Google Sheets
- LibreOffice Calc
- Python (optional)

Your journal only needs the **final images**, not the code.

4. Testing Evidence (Screenshots)

You must capture:

CPU Test Evidence

- top or htop showing 100% CPU usage

- vmstat 1 output

Memory Test Evidence

- free -h before and during load
- vmstat showing swap activity

Disk Test Evidence

- fio output
- iostat -xz 1 showing high I/O wait

Network Test Evidence

- iperf3 -s on server
- iperf3 -c on workstation

nginx Test Evidence

- curl response times
- ab or wrk load test output
- nginx access logs

Monitoring Script Evidence

- Output of monitor-server.sh
- Log file screenshot

5. Network Performance Analysis

```
ping 192.168.56.10
iperf3 -c 192.168.56.10
ss -tuna
sar -n DEV 1 10
```

Document:

- Latency (min/avg/max)
- Throughput (Mbps)
- Packet loss
- Connection count
- Network saturation

6. Optimisation Analysis

Optimisation 1

Edit

```
sudo nano /etc/nginx/nginx.conf
```

set

```
worker_processes auto;
worker_connections 4096;
```

Restart

```
sudo systemctl restart nginx
```

Improvement:

- Higher RPS
- Lower response time
- Better CPU utilisation

Optimisation 2 — Change I/O

Check current

```
cat /sys/block/sda/queue/scheduler
```

Set to deadline

```
echo deadline | sudo tee /sys/block/sda/queue/scheduler
```

Improvement:

- Lower disk latency
- Higher IOPS

Optimisation 3 Increase VM Resources

- Increase CPU cores from 1 → 2
- Increase RAM from 1GB → 2GB

Expected improvement:

- Reduced swapping
- Lower load average
- Faster nginx response times

Optimisation Results Table

Metric	**Before**	**After**	**Improvement**
nginx RPS	450	900	+100%
nginx Response Time	85ms	40ms	-53%
Disk Latency	12ms	4ms	-66%
CPU Load Avg	4.2	3.0	-28%

Phase 7: Security Audit and System Evaluation (Week 7)

Security Assessment

The audit focuses on five core areas:

- **Authentication** (SSH hardening, key-based access)
- **Authorization** (sudo privileges, file permissions)
- **Access Control** (AppArmor enforcement)
- **Network Security** (firewall, open ports, nmap scan)
- **Intrusion Detection** (fail2ban, log analysis)

The system was evaluated using:

- **Lynis** for host-based security scanning
- **nmap** for external network assessment
- **Manual verification** of SSH, firewall, AppArmor, and services
- **Log inspection** for intrusion attempts

Lynis Security

Category	Finding
Authentication	PasswordAuthentication enabled (warning)
Firewall	UFW active (good)
File Permissions	Some world-writable files detected
Logging	Log rotation configured
Malware	No malware scanner installed
Hardening Index	68/100

Network Security assessment with nmap

Port	State	Service	Notes
22/tcp	open	ssh	Only allowed from workstation (UFW rule)
80/tcp	open	nginx	Required for testing workload
All others	closed	—	Firewall functioning correctly

Access Control Verification

Authentication

ssh adminuser@192.168.56.10

Failed Password Attempt

ssh root@192.168.56.10

Permission denied (publickey).

sudo fail2ban-client status sshd

Justification

Service	Purpose	Justification
sshd	Remote administration	Required for secure management
nginx	Server workload	Required for performance testing
fail2ban	Intrusion detection	Protects SSH from brute force
apparmor	Mandatory access control	Enforces OS-level isolation
ufw	Firewall	Network access control
systemd-journald	Logging	Required for audit trails
unattended-upgrades	Security updates	Ensures patch compliance

Services, and System Configuration review

Even after remediation, some risks remain:

Residual Risks

Risk	Description	Mitigation
Zero-day vulnerabilities	Unknown exploits	Keep auto-updates enabled
nginx exposure	Web server accessible on port 80	Use AppArmor confinement
Insider threat	Admin user compromise	Strong SSH keys, audit logs
VM escape	Hypervisor vulnerabilities	Keep VirtualBox updated

Overall Risk Level:

Low to Moderate, appropriate for a controlled lab environment.

References

Microsoft Copilot (2024) AI-assisted technical guidance used throughout the coursework for structuring explanations, refining documentation, and supporting system administration tasks.

Moodle Weekly Lecture Attachments (Weeks 1–10) University of Roehampton (2024). *Operating Systems Module Resources*. Includes:

- Week 1: Operating Systems & Sustainability
- Week 2: Process Management & System Performance
- Week 3: Lab Activity & Performance Tools
- Week 4: Memory Management & Resource Allocation
- Week 5: Initial Security Configuration
- Week 10: Inter-Process Communication (Accessed via Moodle course page)

- https://nmap.org/book/man.html