

Hosting Week-12

1. NGINX

Overview: NGINX is a popular open-source web server known for its high performance, scalability, and low resource consumption. It excels at serving static content (e.g., images, stylesheets, JavaScript) and proxying requests to backend servers.

Key Features:

- **Web Server**: Serves web pages to users.
- **Reverse Proxy**: Forwards client requests to backend servers, hiding their details.
- **Load Balancer**: Distributes traffic across multiple servers for stability and performance.
- **HTTP Cache**: Caches content to speed up delivery of frequently requested files.

Advantages:

1. High performance
2. Scalability
3. Load balancing

4. Reverse proxy and caching
5. SSL/TLS support

Disadvantages:

1. Complex configuration
2. Lack of Windows support
3. Logging complexity

2. EC2

****Overview****: Amazon EC2 (Elastic Compute Cloud) provides virtual servers in the cloud, enabling you to host websites, run applications, or store data without managing physical hardware.

3. Choosing Ubuntu as Amazon Machine Image (AMI)

****Why Ubuntu?****: Ubuntu is a popular AMI choice for EC2 due to its user-friendly, secure, and stable environment. It offers:

- Wide community support
- Frequent updates

- Compatibility with many tools
- Cost-effective (free and open-source)

4. Key-Pair for Remote Access

Definition: A key pair is a security mechanism for secure remote access to EC2 servers, consisting of:

1. **Private Key:** Stored on your computer (secret).
2. **Public Key:** Stored on the server.

Why Use Key Pairs?

- **Secure Authentication:** Replaces passwords (harder to hack).
- **No Password Needed:** Simplifies login.
- **Encryption:** Protects data during transmission.

5. Purpose of Setting Inbound Rules

Purpose:

- Control access to your server.

- Enhance security by blocking unauthorized access.
- Expose only necessary services.
- Apply the principle of least privilege (limit access to essentials).

6. IP (Internet Protocol)

Definition: An IP address is a unique identifier for devices on a network, enabling communication and data exchange.

Examples:

- Browsing:** "example.com" → DNS resolves to IP (e.g., 93.184.216.34) → Website loads.
- Device Communication:** Phone (192.168.1.2) sends data to Laptop (192.168.1.3) via IP.

IPv4 vs IPv6:

Feature	IPv4	IPv6
Address Length	32 bits (~4.3B)	128 bits (near infinite)
Format	192.168.1.1	2001:0db8::8a2e:0370:7334
Types	Broadcast	Multicast
Header	Complex	Simplified

Config	Manual/DHCP	Auto-config	
Security	Add-on (IPsec)	Built-in IPsec	
NAT	Common	Not needed	
Adoption	Widely used	Growing	

****Importance****: IPv6 addresses the IPv4 shortage and improves efficiency.

7. TCP

****Definition****: Transmission Control Protocol (TCP) ensures reliable data delivery over networks by:

- Delivering packets error-free.
- Resending lost/damaged data.
- Establishing connections first.
- Maintaining data order.

8. DNS

****Definition**:** Domain Name System (DNS) translates domain names (e.g., www.google.com) into IP addresses (e.g., 142.250.64.110).

How It Works:

1. Type "www.google.com".
2. Check local cache → If not found, query DNS resolver.
3. Resolver asks: Root → TLD (.com) → Google's server.
4. Returns IP → Site loads.

9. SSL/TLS

****Definition**:** SSL (Secure Socket Layer) and TLS (Transport Layer Security) secure browser-server communication by:

- Encrypting sensitive data (e.g., passwords).
- Verifying website authenticity.
- Ensuring data integrity.

10. Load Balancer

****Definition**:** A load balancer distributes traffic across multiple servers to:

- Prevent overload.
- Improve performance, reliability, and availability.

11. Proxy Server

****Definition**:** A proxy server acts as an intermediary between clients and the internet, forwarding requests and responses for:

- Enhanced speed
- Security
- Access control

12. Proxy Cache

****Definition**:** Proxy cache stores frequently accessed web resources locally to:

- Speed up access.
- Reduce bandwidth usage.
- Serve cached content directly.

13. One Entry Point

****Definition****: One entry point routes all requests through a single public server (gateway) for:

- Traffic filtering
- Improved security and management

14. Compression and Segmentation

- ****Compression****: Reduces data size (e.g., HTML, CSS) for faster loading.
- ****Segmentation****: Breaks large data into smaller chunks for efficient transfer.

15. NGINX Configuration

- ****HTTP to HTTPS Redirect****: Forces secure connections.

- **SSL/TLS Setup**: Encrypts traffic with certificates.
- **Load Balancing**: Distributes requests across servers.
- **Caching**: Stores frequent data for speed.

16. HTTP Ports

- **80**: HTTP (unencrypted)
- **443**: HTTPS (encrypted)
- **8080**: Alternative HTTP (testing)
- **8443**: Alternative HTTPS
- **3000, 5000, 8000**: Development server ports (e.g., Node.js)

17. Subdomain

Definition: A subdomain (e.g., blog.example.com) organizes content under a main domain.

Advantages:

- Separates concerns (e.g., blog vs store).

- Enhances security isolation.

18. Site-Enabled vs Site-Available

- **Site-Available**: Config exists but inactive.
- **Site-Enabled**: Config active and serving.
- **Purpose**: Manage multiple sites efficiently.

19. Alternatives to NGINX

1. **Apache**: Flexible, excels with dynamic content (.htaccess).
2. **LiteSpeed**: High-performance, caching pro (great for WordPress).
3. **Caddy**: Simple, auto-HTTPS setup.

20. NGINX vs Apache

Feature	NGINX	Apache	
---------	-------	--------	--

-----	-----	-----
Architecture	Event-driven	Process/thread-based
Performance	Static & high traffic	Dynamic content
Security	Minimal attack surface	Extensive modules
.htaccess	No support	Full support
Load Balancing	Built-in	Via modules

21. Sudo

Definition: SuperUser Do grants admin powers in Linux for:

- **Permission:** Access restricted tasks.
- **Security:** Temporary admin access.

22. PM2

Definition: Process Manager for Node.js:

- Keeps apps running.
- Auto-restarts on crashes.
- Manages multiple apps & logs.

GIT

1. Git Reset

- **Reset**: Unstages changes, keeps commits.
- **Reset --soft**: Deletes last commit, keeps changes staged.
- **Reset --hard**: Deletes last commit & changes.

2. Git Stash

- **Stash**: Saves uncommitted changes.
- **Pop**: Applies & removes stash.
- **Apply**: Applies stash, keeps it.
- **Drop**: Deletes a stash.

3. Git Clone/Fetch

- **Clone**: Copies entire remote repo.
- **Fetch**: Updates local repo without merging.

4. Git Ignore

- **.gitignore**: Excludes files from tracking (e.g., secrets).

5. Git Fork

- Copies repo to your account → Pull request to original.

6. Git Repository

- Stores project files & history.

7. Git Head

- ****HEAD****: Points to current commit/branch.
- ****Detached HEAD****: Points to a specific commit.

8. Git Master

- Default branch after cloning.

9. Git Origin

- Default name for remote repo.

10. Git Branch

- Separate project versions:
 - Create: ``git branch <name>``
 - Delete: ``git branch -d/-D <name>``
 - Switch: ``git checkout <name>``

11. Git Rebase

- Linearly adds branch commits to main.

12. Git Squash

- Combines multiple commits into one via ``git rebase -i``.

13. Git Cherry-Pick

- Applies specific commit to another branch.

14. Git Log

- Shows commit history.

15. Git Diff

- Compares changes (working dir, commits, branches).

16. Git Revert

- Undoes commit with a new commit.

17. Git Rm

- Removes files from repo & working dir.

18. Git Pull vs Fetch

Feature	**Git Pull**	**Git Fetch**	
---------	---------------------	----------------------	--

Purpose	Fetches & merges	Fetches only	
Effect	Updates local branch	No merge	
Command	`git pull origin main` `git fetch origin main`		

arunji