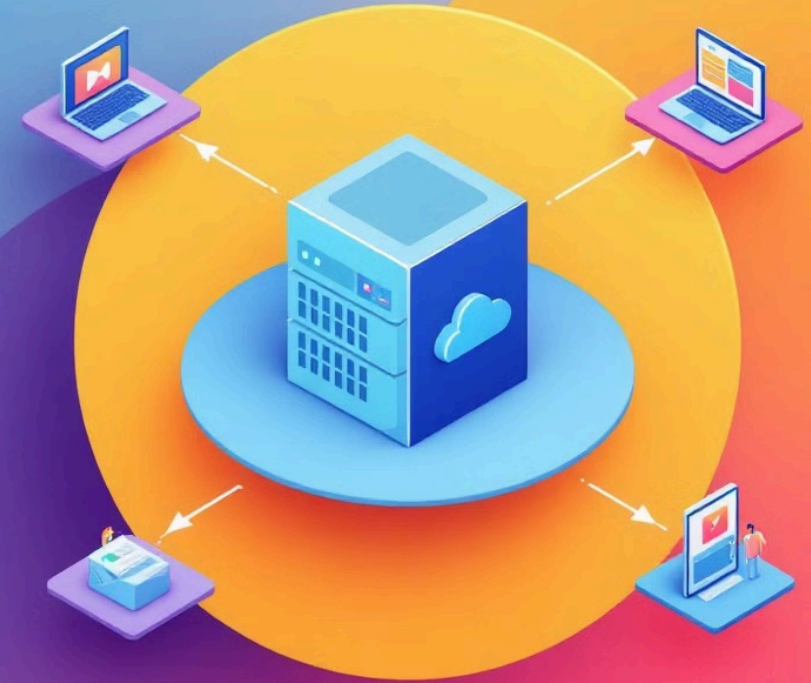


Configuring Virtual Hosts and Reverse Proxies

Welcome to Module 2 of our web server configuration series. In this presentation, we'll explore the concepts of virtual hosts in Apache and Nginx, and provide step-by-step guides for configuring them on different operating systems.

DK by Dan K





Understanding Virtual Hosts

What are Virtual Hosts?

Virtual Hosts allow a single web server to host multiple websites/domains, each with separate configurations.

Types of Virtual Hosts

Apache: Name-based and IP-based.
Nginx: Uses server blocks, equivalent to Apache virtual hosts.

When to Use Virtual Hosts

Hosting multiple websites, separating environments, enforcing different configurations per site.



Apache Configuration on AlmaLinux

1

Install Apache

Use `sudo dnf install httpd -y` and enable the service.

2

Create Directories

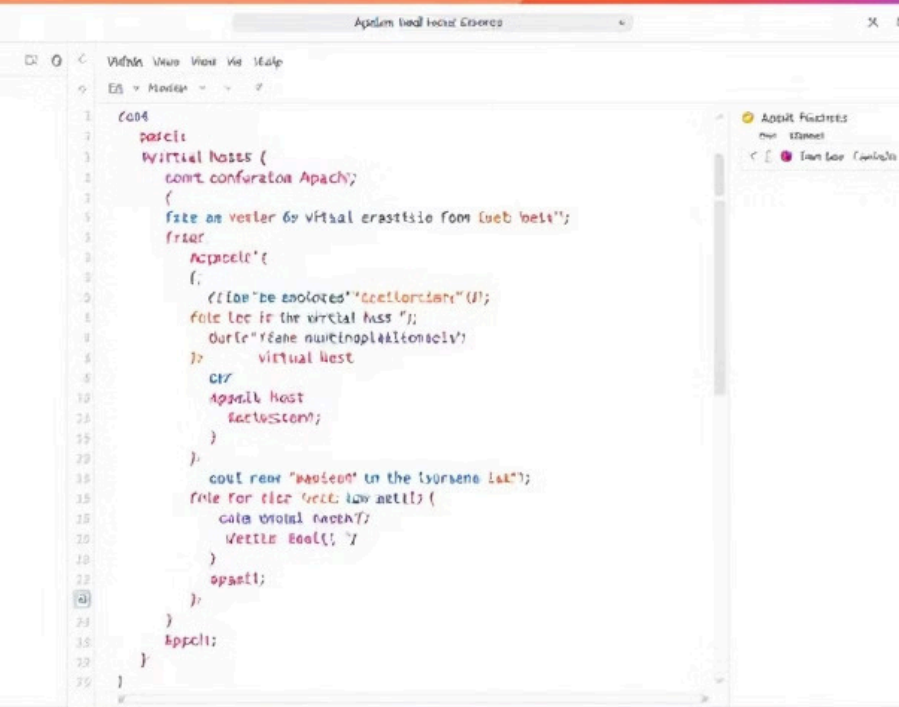
Make `sites-available` and `sites-enabled` directories.

3

Edit Configuration

Add `IncludeOptional` directive to `httpd.conf` file.

Apache Virtual Host Configuration

A screenshot of a code editor window titled 'Apache Virtual Host Config'. The editor shows a configuration file with the following content:

```
1 #
2 # To be able to use the functionality of a module which was
3 # not loaded by the main httpd.conf file, you must have the
4 # right loadmodule lines at the beginning of this
5 # configuration file. The syntax for the loadmodule lines is:
6 #
7 # loadmodule module_name [module_path]
8 #
9 # To be able to use the functionality of a module that was
10 # built to work with a dynamically loaded HTTP module you
11 # may need to declare the module name in the
12 # VirtualHost section of the httpd.conf file, and not
13 # here. The syntax for that is shown below:
14 #
15 # #LoadModule module_name [module_path]
16 #
17 # To be able to use the functionality of a module that was
18 # built to work with a dynamically loaded HTTP module you
19 # may need to declare the module name in the
20 # VirtualHost section of the httpd.conf file, and not
21 # here. The syntax for that is shown below:
22 #
23 # #LoadModule module_name [module_path]
24 #
25 # To be able to use the functionality of a module that was
26 # built to work with a dynamically loaded HTTP module you
27 # may need to declare the module name in the
28 # VirtualHost section of the httpd.conf file, and not
29 # here. The syntax for that is shown below:
30 #
31 # #LoadModule module_name [module_path]
```

1

Create Virtual Host File

Create a new .conf file in sites-available directory.

2

Add Virtual Host Settings

Configure ServerName, DocumentRoot, and log file locations.

3

Create Website Directory

Make directory for website files and set permissions.

4

Enable the Site

Create symbolic link in sites-enabled directory.

Finalizing Apache Setup on AlmaLinux

Adjust SELinux Policies

If SELinux is enabled, set `httpd_unified` to 1. Check SELinux status with `cat /etc/selinux/config`.

Restart Apache

Use `sudo systemctl restart httpd` to apply changes. Your website should now be accessible.

Apache Configuration on Ubuntu/Debian

1

Install Apache

Use `sudo apt install apache2 -y` and enable the service.

2

Create Directory Structure

Make directories for your site and set permissions.

3

Create Default Page

Add an `index.html` file to your site's `public_html` directory.



Virtual Host Configuration on Ubuntu/Debian

Create Virtual Host File

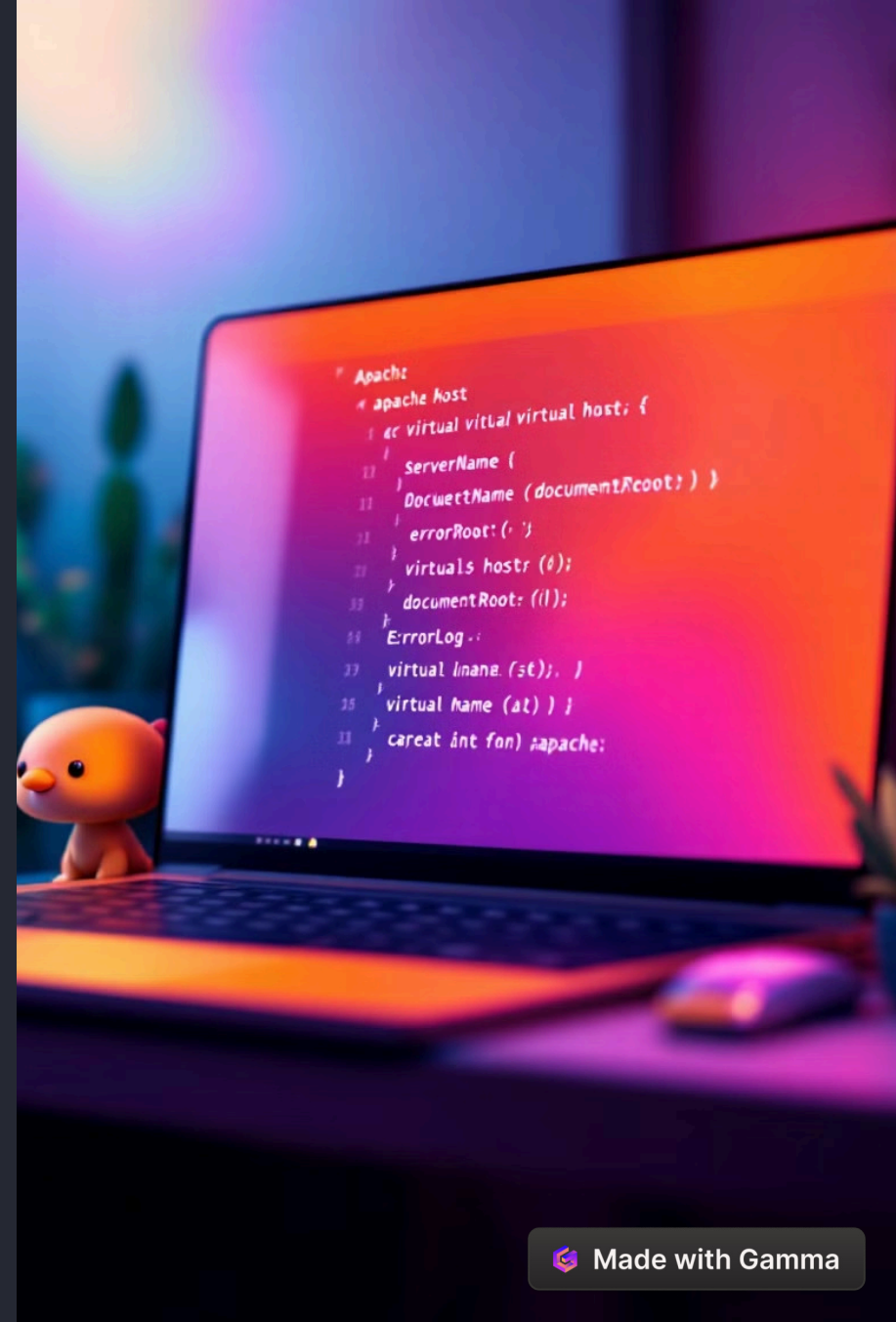
Use `sudo nano` to create a `.conf` file in `/etc/apache2/sites-available/`

Add Configuration

Set `ServerName`, `DocumentRoot`, and log file locations in the `VirtualHost` block.

Enable Site

Use `a2ensite` command to enable the new virtual host.



Nginx Virtual Host Configuration



Server Blocks

Nginx uses server blocks, equivalent to Apache's virtual hosts.



Enable Site

Create a symbolic link in `/etc/nginx/sites-enabled/`



Configuration

Create a new server block in `/etc/nginx/sites-available/`

Best Practices for Virtual Hosts

1 Use Descriptive Names

Choose clear, meaningful names for your virtual host configuration files.

3 Regular Backups

Regularly backup your virtual host configurations and website files.

2 Separate Logs

Maintain separate log files for each virtual host for easier troubleshooting.

4 Security Measures

Implement proper security measures for each virtual host, including SSL certificates.



Next Steps

1

Test Configurations

Verify each virtual host is working correctly.

2

Implement SSL

Set up SSL certificates for secure connections.

3

Optimize Performance

Fine-tune server settings for better performance.

4

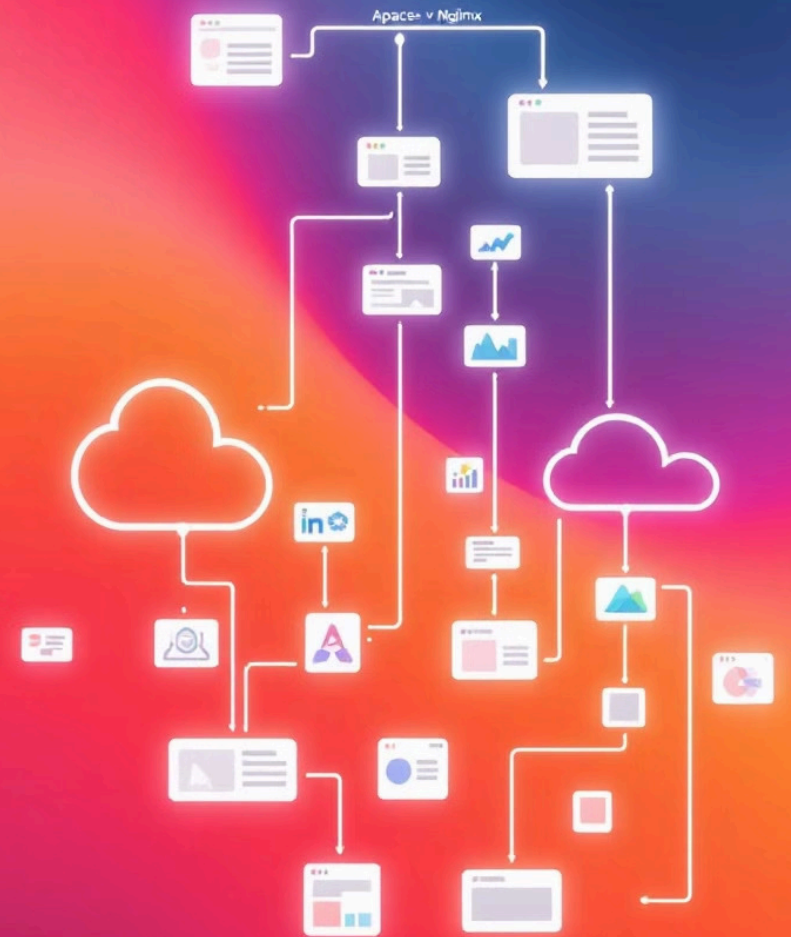
Monitor and Maintain

Regularly check logs and update configurations as needed.

Web Server Configuration: Apache and Nginx

This presentation covers the configuration of virtual hosts and server blocks for Apache and Nginx web servers on different Linux distributions. We'll explore the setup process, key configuration files, and best practices for hosting multiple websites on a single server.

DK by Dan K



Apache Virtual Host Configuration



1

Disable Default Configuration

Use **sudo a2dissite 000-default.conf** to disable the default Apache configuration.

2

Enable Custom Site

Enable your custom site with **sudo a2ensite example.com.conf**.

3

Reload and Test

Reload Apache configuration, test it, and restart the service.

Nginx Configuration on AlmaLinux

1

Install Nginx

Use **sudo dnf install nginx -y** to install Nginx on AlmaLinux.

2

Create Directory Structure

Set up the necessary directories and permissions for your website.

3

Configure Server Blocks

Create and edit the server block configuration file for your domain.

Nginx Server Block Configuration

```
f server web settings
/s < of ljswrl, configured, and, and l
tince sirver <intl blocks onfs (166)>
<block elctigns/1( {
  (actious inat Fouln the vervife
  block = loca; fast on whert serv))
//= Ser vers block block (block,
  lettion aertink settings, (166)
  the "server (ilnts haew settions)
  <lact setting block>
  /:= calb:
  ?
};
```

Listen Directive

Specify the port Nginx should listen on, typically 80 for HTTP.

Server Name

Define the domain names that should match this server block.

Root Directory

Set the root directory for serving files for this domain.

Location Blocks

Configure how Nginx should handle requests for different URL patterns.

Nginx Configuration on Ubuntu/Debian

Installation

Install Nginx using **sudo apt install nginx -y** and enable the service.

Directory Setup

Create the necessary directory structure and set proper permissions for your website files.

Server Block Creation

Create and configure the Nginx server block file for your domain in the sites-available directory.

Enabling Nginx Server Blocks

1

Create Symlink

Link the server block file from sites-available to sites-enabled.

2

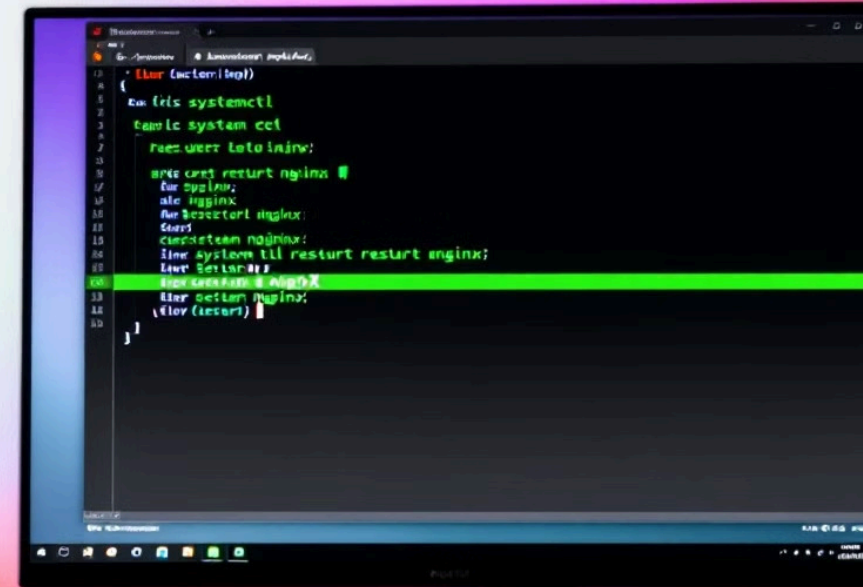
Test Configuration

Use **sudo nginx -t** to check for any configuration errors.

3

Restart Nginx

Apply changes by restarting the Nginx service.



```
12 # systemctl enable nginx.service
13 # systemctl restart nginx
14 # systemctl status nginx
15 # systemctl restart nginx
16 # systemctl status nginx
17 # systemctl restart nginx
18 # systemctl status nginx
19 # systemctl restart nginx
20 # systemctl status nginx
21 # systemctl restart nginx
22 # systemctl status nginx
23 # systemctl restart nginx
24 # systemctl status nginx
25 # systemctl restart nginx
26 # systemctl status nginx
27 # systemctl restart nginx
28 # systemctl status nginx
29 # systemctl restart nginx
30 # systemctl status nginx
31 # systemctl restart nginx
32 # systemctl status nginx
33 # systemctl restart nginx
34 # systemctl status nginx
35 # systemctl restart nginx
36 # systemctl status nginx
37 # systemctl restart nginx
38 # systemctl status nginx
39 # systemctl restart nginx
40 # systemctl status nginx
41 # systemctl restart nginx
42 # systemctl status nginx
43 # systemctl restart nginx
44 # systemctl status nginx
45 # systemctl restart nginx
46 # systemctl status nginx
47 # systemctl restart nginx
48 # systemctl status nginx
49 # systemctl restart nginx
50 # systemctl status nginx
```



Reverse Proxy Basics



Security

Protects backend servers from direct exposure to the internet.



Load Balancing

Distributes incoming requests across multiple backend servers.



Performance

Improves speed through caching and efficient request handling.



Caching
(caul by caching)



SSL meaking.
The elbcess for maldoye in

Common Reverse Proxy Use Cases

Load Balancing

Distribute incoming traffic across multiple backend servers to improve performance and reliability.

Caching

Store and serve static content to reduce load on backend servers and improve response times.

SSL Termination

Handle SSL/TLS encryption and decryption to offload this task from backend servers.

Virtual Hosts vs Server Blocks

Apache Virtual Hosts

Apache's method for hosting multiple websites on a single server. Configured in separate .conf files within the sites-available directory.

Nginx Server Blocks

Nginx's equivalent to Apache's virtual hosts. Allows hosting multiple domains on one server, configured in the nginx.conf file or separate configuration files.



Key Takeaways

1

Configuration Flexibility

Both Apache and Nginx offer flexible ways to host multiple websites on a single server.

2

Security Considerations

Proper configuration of virtual hosts and server blocks is crucial for maintaining security and isolation between websites.

3

Performance Optimization

Reverse proxies can significantly enhance web server performance and security when properly implemented.