**FAMILIARISING VIM AND NANO**

Vim and Nano are completely different terminal text editors.

**Nano is simple, easy to use and master while Vim is powerful and tough to master.**

To differentiate, it will be better to list some features of them. I'm listing some of the most interesting features that help for the way we use them today.

**NANO:**

Nano is probably the easiest to work with from our list as its interface is comparable to GUI-based text editors. Most Linux distributions come with Nano pre-installed. Nowadays, Nano is pre-installed in LINUX systems.

* Easy to use and master.
* Nano has most of the shortcuts listed at the bottom of the window, making it extremely simple to use.
* Search function
* Search and replace
* "Goto line" command
* Automatic indentation

Some of the basic commands are:

* Ctrl+X – quit
* Ctrl+O –Save
* Alt+6 – Copy marked text
* Ctrl+u – Paste the text

Most of the basic commands will be given down the interface screen.

**VIM:**

Vim is powerful, but tough to learn. It is so called "modal editor", as opposed to usual editors, like nano. Vim allows you to reach all text with just a few key presses, so if you have touch-type ability, your movement without a mouse would be even faster than with one.

* Tough to get started with and master. The editing and command modes will confuse beginners.
* Session recovery
* Split screen
* Tab expansion
* Completion commands

Some of the basic commands for Vim are:

* **i** – Insert before cursor position
* **:w** – Save the file
* **:wq** or **:x** – Save & Quit
* **:q** – Quit
* **:q!** – Force quit, no save

**GREP COMMANDS**

**Grep**is an acronym that stands for **G**lobal **R**egular **E**xpression **P**rint. Grep is a Linux / UNIX command-line tool used to search for a string of characters in a specified file. The text search pattern is called a regular expression. When it finds a match, it prints the line with the result. The grep command is handy when searching through large log files.

The grep command consists of three parts in its most basic form. The first part starts with **grep**, followed by the pattern that you are searching for. After the string comes the file name that the grep searches through.

Syntax: *“grep <string> <filename>*

Some of other syntax is:

Searching in multiple files: *“grep <string> <filename1><filename2><filename3>”*

Searching all files in the directory: *“grep nix \*”*

Search strings in any cases: *“grep -i phoenix \*”*

To count the number of matches: *“grep -c phoenix \*”*

# LAMP APPLICATIONS AND INSTALLATION

LAMP installations (Linux + Apache + MySQL + PHP/Perl/Python) are a popular setup for Ubuntu servers. One advantage of LAMP is the substantial flexibility for different database, web server, and scripting languages. This stack is a group of open-source software that is typically installed together to enable a server to host dynamic websites and web apps.

Here are the steps how to install LAMP stack on any LINUX:

1. Installing Apache and Updating the Firewall:

* First, make sure your apt cache is updated with:

sudo apt update

* Once the cache has been updated, you can install Apache with:

sudo apt install apache2

Now your firewall needs to allow HTTP and HTTPS traffic. We can check that UFW has an application profile for Apache like so:

sudo ufw app list

We will get an output like:

Available applications:

Apache

Apache Full

Apache Secure

OpenSSH

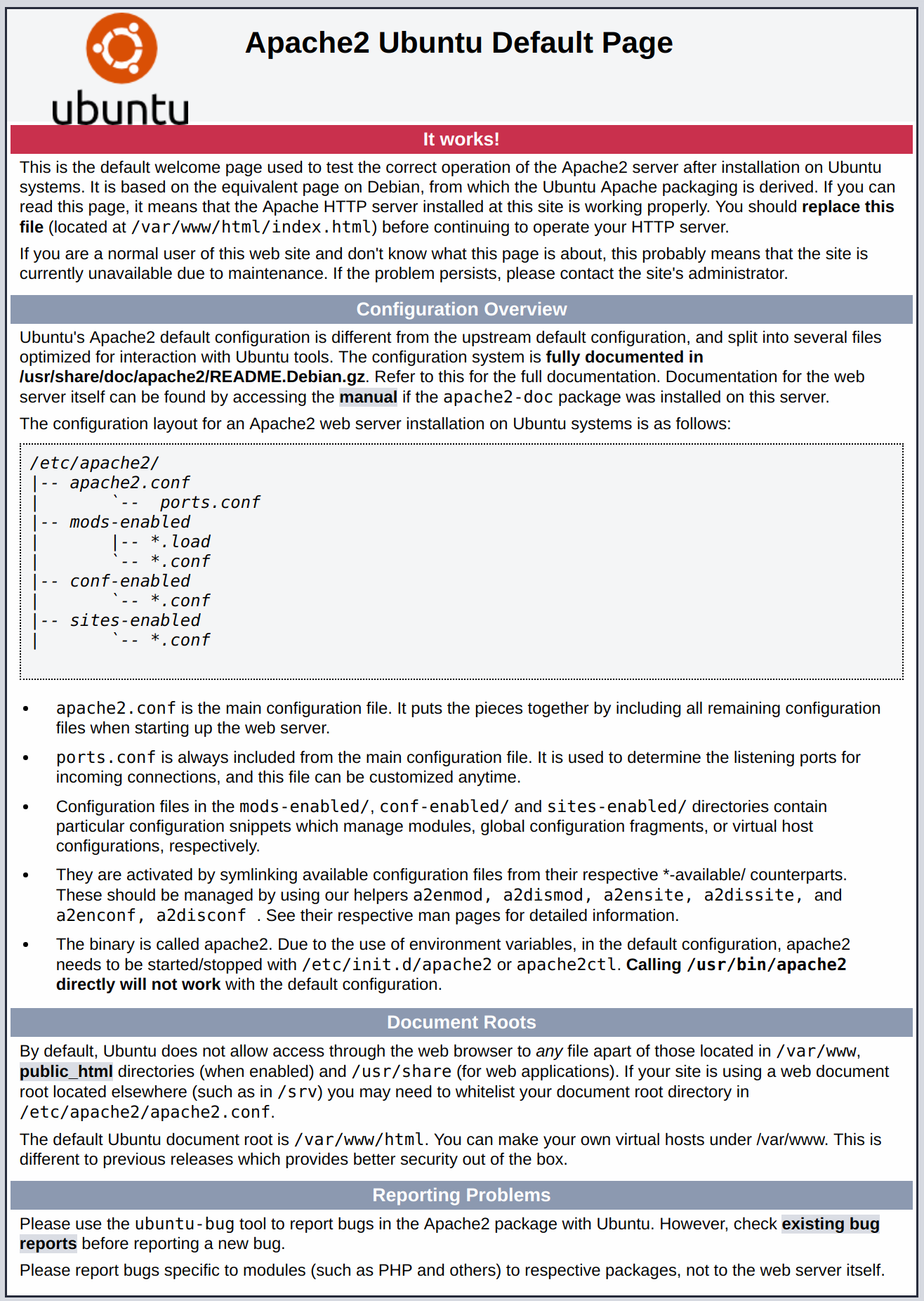
To allow incoming HTTP and HTTPS traffic for this server, run:

sudo ufw allow "Apache Full"

You can do a spot check right away to verify that everything went as planned by visiting your server’s public IP address in your web browser:

<http://your_server_ip>

You will get a preview similar to this:



* Inorder to bget your IP type:

curl ifconfig.me

curl icanhazip.com

1. Installing MySQL:

MySQL is a database management system. Basically, it will organize and provide access to databases where your site can store information.

* Use apt to acquire and install this software:

sudo apt install mysql-server

* When the installation is complete, run a simple security script that comes pre-installed with MySQL

sudo mysql\_secure\_installation

When you’re finished, test if you’re able to log in to the MySQL console by typing:

sudo mysql

## Installing PHP

PHP is the component of your setup that will process code to display dynamic content. It can run scripts, connect to your MySQL databases to get information, and hand the processed content over to your web server so that it can display the results to your visitors. Run the following command to install all three packages and their dependencies:

sudo apt install php libapache2-mod-php php-mysql

1. Setting up a Virtual Host

When using the Apache web server, you can use virtual hosts to encapsulate configuration details and host more than one domain from a single server. We will set up a domain called your\_domain, but you should replace this with your own domain name.

Let’s create a directory structure within /var/www for **your\_domain** site, leaving /var/www/html in place as the default directory to be served if a client request doesn’t match any other sites.

Create the directory for **your\_domain** as follows:

sudo mkdir /var/www/your\_domain

Next, assign ownership of the directory with the $USER environment variable, which references the current logged user:

sudo chown -R $USER:$USER /var/www/your\_domain

The permissions of your web root directory should be correct if you haven’t modified its umask value, but you can make sure by typing:

sudo chmod -R 755 /var/www/your\_domain

Next, create a sample index.html page using nano or your favorite editor:

nano /var/www/your\_domain/index.html

Inside, add the following sample HTML:

<html>

<head>

<title>Welcome to Your\_domain!</title>

</head>

<body>

<h1>Success! The your\_domain server block is working!</h1>

</body>

</html>

In order for Apache to serve this content, it’s necessary to create a virtual host file with the correct directives. Instead of modifying the default configuration file directly, let’s make a new one:

sudo nano /etc/apache2/sites-available/your\_domain.conf

<VirtualHost \*:80>

ServerAdmin webmaster@localhost

ServerName your\_domain

ServerAlias www.your\_domain

DocumentRoot /var/www/your\_domain

ErrorLog ${APACHE\_LOG\_DIR}/error.log

CustomLog ${APACHE\_LOG\_DIR}/access.log combined

</VirtualHost>

Save and close the file when you are finished.

Let’s enable the file with the a2ensite tool:

sudo a2ensite your\_domain.conf

Disable the default site defined in 000-default.conf:

sudo a2dissite 000-default.conf

Next, let’s test for configuration errors:

sudo apache2ctl configtest

Restart Apache to implement your changes:

sudo systemctl restart apache2

Apache should now be serving your domain name. You can test this by navigating to <http://your_domain>

1. Testing PHP Processing on your Web Server

In order to test that your system is properly configured for PHP, create a PHP script called info.php. In order for Apache to find this file and serve it correctly, it must be saved to your web root directory.

sudo nano /var/www/your\_domain/info.php

This will open a blank file. Add the following text, which is valid PHP code, inside the file:

<?php

phpinfo();

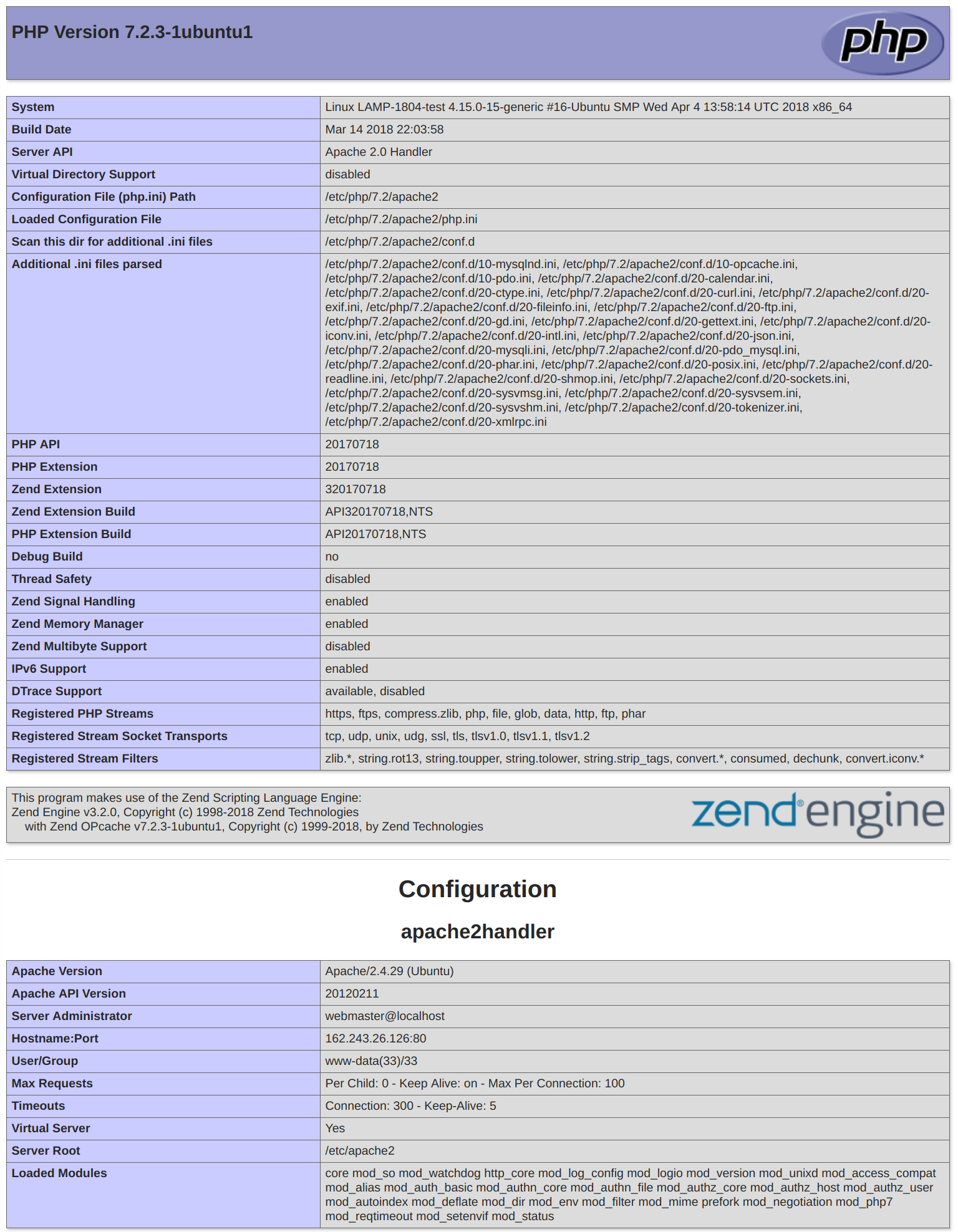
When you are finished, save and close the file.

Now you can test whether your web server is able to correctly display content generated by this PHP script. To try this out, visit this page in your web browser. You’ll need your server’s public IP address or domain name again.

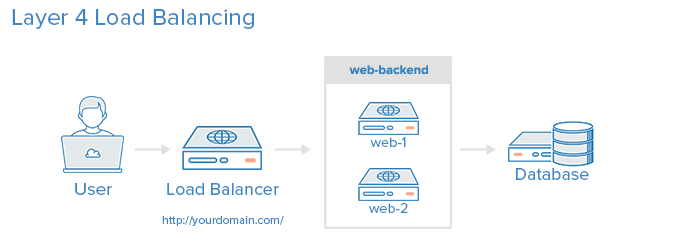
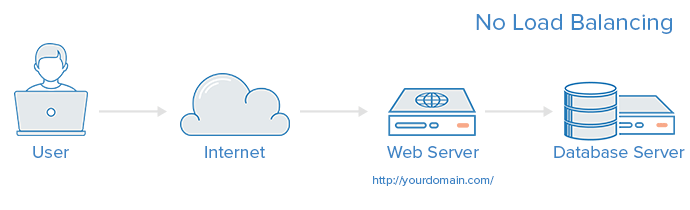
The address you will want to visit is:

http://your\_domain/info.php

The page that you come to should look something like this:



**LOAD BALANCING AND HAPROXY**

HAProxy, which stands for High Availability Proxy, is a popular open source software TCP/HTTP Load Balancer and proxying solution which can be run on Linux, Solaris, and FreeBSD. Its most common use is to improve the performance and reliability of a server environment by distributing the workload across multiple servers (e.g. web, application, database). It is used in many high-profile environments, including: GitHub, Imgur, Instagram, and Twitter. 

**Load Balancing Algorithms**

The load balancing algorithm that is used determines which server, in a backend, will be selected when load balancing. HAProxy offers several options for algorithms. In addition to the load balancing algorithm, servers can be assigned a weight parameter to manipulate how frequently the server is selected, compared to other server.

A few of the commonly used algorithms are as follows:

* **ROUNDROBIN**: Round Robin selects servers in turns. This is the default algorithm.
* **LEASTCONN**: Selects the server with the least number of connections–it is recommended for longer sessions. Servers in the same backend are also rotated in a round-robin fashion.
* **SOURCE**: This selects which server to use based on a hash of the source IP i.e. your user’s IP address. This is one method to ensure that a user will connect to the same server.

## Sticky Sessions: Some applications require that a user continues to connect to the same backend server. This persistence is achieved through sticky sessions, using the app session parameter in the backend that requires it.

**INSTALLATION**

Install haproxy with this command: sudo apt install -y haproxy

Once installed HAProxy should already have a template for configuring the load balancer. Open the configuration file, for example, using nano with the command underneath:

sudo nano /etc/haproxy/haproxy.cfg

Add the following sections to the end of the file. Replace the <server name> with whatever you want to call your servers on the statistics page and the <private IP> with the private IPs for the servers you wish to direct the web traffic to.

frontend http\_front

bind \*:80

stats uri /haproxy?stats

default\_backend http\_back

backend http\_back

balance roundrobin

server <server1 name> <private IP 1>:80 check

server <server2 name> <private IP 2>:80 check

After making the configurations, save the file and restart HAProxy with the next command:

sudo systemctl restart haproxy

With the HAProxy configured and running, open your load balancer server’s public IP in a web browser and check that you get connected to your backend correctly.

http://<load balancer public IP>/haproxy?stats

**PASSWORDLESS AUTHENTICATION**

Secure Shell (SSH) is a cryptographic network protocol used for secure connection between a client and a server and supports various authentication mechanisms. The two most popular mechanisms are passwords based authentication and public key based authentication.

**Setup SSH Passwordless Login**

To set up a passwordless SSH login in Linux all you need to do is to generate a public authentication key and append it to the remote hosts ~/.ssh/authorized\_keys file.

* 1. Generate a new SSH key pair:

The following command will generate a new 4096 bits SSH key pair with your email address as a comment:

ssh-keygen -t rsa

Next, it will ask for filename and passphrase. Press enter to save it by default and with an empty passphrase.

* 1. Copy the public key

Now that you have generated an SSH key pair, in order to be able to login to your server without a password you need to copy the public key to the server you want to manage.

The easiest way to copy your public key to your server is to use a command called ssh-copy-id. On your local machine terminal type:

ssh-copy-id remote\_username@server\_ip\_address

Once the user is authenticated, the public key will be appended to the remote user authorized\_keys file and connection will be closed.

* 1. Login to your server using SSH keys

After completing the steps above you should be able log in to the remote server without being prompted for a password.

**DISABLING ROOT LOGIN**

Add the user. In the following example, we will use the user name **admin**. The command adduser will automatically create the user, initial group, and home directory.

**adduser admin**

Set the password for the admin user. When prompted, type and then retype the password.

**passwd admin**

For sudo permissions for your new admin user, use the following command.

**usermod –aG sudo admin**

To disable root SSH login, edit /etc/ssh/sshd\_config with your text editor

**vi /etc/ssh/sshd\_config**

Change this line:

**#PermitRootLogin yes**

Edit to this:

**PermitRootLogin no**

**VIEWING LOGS**

Linux logs provide a timeline of events for the Linux operating system, applications and system and are a valuable troubleshooting tool when you encounter issues. When issues arise, analyzing log files is the first thing an administrator needs to do.

For desktop app-specific issues, log files are written to different locations. Where a desktop application writes logs depends on the developer and whether or not the app allows for custom log configuration. Chrome, for example, writes crash reports to ‘~/.chrome/Crash Reports’.

Linux log files are stored in plain-text and can be found in the **/var/log** directory and subdirectory. There are Linux logs for everything: system, kernel, package managers, boot processes, Xorg, Apache, MySQL, etc. In this article, we will focus specifically on Linux system logs.

### We can group most directories into one of four categories:

* Application Logs
* Event Logs
* Service Logs
* System Logs

Monitoring every log is a monumental task and one reason we included centralized [log management](https://stackify.com/log-management/) when we created [Retrace](https://stackify.com/retrace/). Log monitoring and management is essential for all developers, but the logs that you monitor will depend on your goals or other variables. There is some consensus about the most critical, [must-monitor logs](https://www.eurovps.com/blog/important-linux-log-files-you-must-be-monitoring).

**ABOUT CLOUD IaaS**

Infrastructure-as-a-Service, commonly referred to as simply “IaaS,” is a form of [cloud computing](https://www.ibm.com/cloud/learn/cloud-computing) that delivers fundamental compute, network, and storage resources to consumers on-demand, over the internet, and on a pay-as-you-go basis. IaaS enables end users to scale and shrink resources on an as-needed basis, reducing the need for high, up-front capital expenditures or unnecessary “owned” infrastructure, especially in the case of “spiky” workloads. In contrast to PaaS and SaaS (even newer computing models like containers and serverless), IaaS provides the lowest-level control of resources in the cloud.

**Advantages**

**Pay-as-you-Go:** Unlike traditional IT, IaaS does not require any upfront, capital expenditures, and end users are only billed for what they use.

**Speed:** With IaaS, users can provision small or vast amounts of resources in a matter of minutes, testing new ideas quickly or scaling proven ones even quicker.

**Availability:** Through things like multizone regions, the availability and resiliency of cloud applications can exceed traditional approaches.

**Scale:** With seemingly limitless capacity and the ability to scale resources either automatically or with some supervision, it’s simple to go from one instance of an application or workload to many.

**Latency and performance:** Given the broad geographic footprint of most IaaS providers, it’s easy to put apps and services closers to your users, reducing latency and improving performance.

**ABOUT CLOUD SaaS**

Software-as-a-Service, or SaaS for short, is a cloud-based method of providing software to users. SaaS users subscribe to an application rather than purchasing it once and installing it. Users can log into and use a SaaS application from any compatible device over the Internet. The actual application runs in [cloud](https://www.cloudflare.com/learning/cloud/what-is-the-cloud/) servers that may be far removed from a user's location.

A SaaS application may be accessed through a browser or through an app. Online email applications that users access through a browser, such as Gmail and Office 365, are common examples of SaaS applications.

The difference between SaaS and a software installation on a user's computer is somewhat like the difference between streaming a TV show online and buying all the seasons of the TV show on DVD.

**Access from anywhere, on any device:** Typically, users can log into SaaS applications from any device and any location. This offers a great deal of flexibility – businesses can allow employees to operate all over the world, and users can access their files no matter they are.

**No need for updates or installations:** The SaaS provider updates and patches the application on an ongoing basis.

**Scalability:** The SaaS provider handles scaling up the application, such as adding more database space or more compute power as usage increases.

**Cost savings:** SaaS cuts down on internal IT costs and overhead. The SaaS provider maintains the servers and infrastructure that supports the application, and the only cost to a business is the subscription cost of the application.

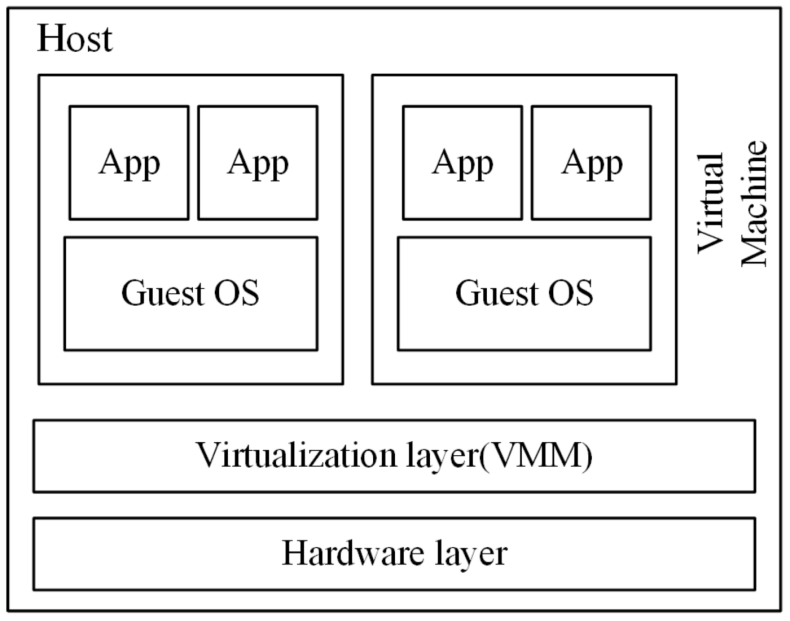
**VIRTUALIZATION AND HYPERVISORS**

Virtualization is the formation of virtual computer inside the real computer. It is an abstraction layer above the hardware layer. It is hardware reducing ,memory saving and cost energy saving technology that is rapidly transforming the IT landscape. In computing, virtualization refers to the creation of a virtual (rather than a real) version of something, including virtual computer hardware platforms, storage devices, and computer resources**.** In other words, virtualization can also be perceived as a technique which allows us to share a physical instance of a resource or an application among multiple customers and organizations.

**Types of virtualization**

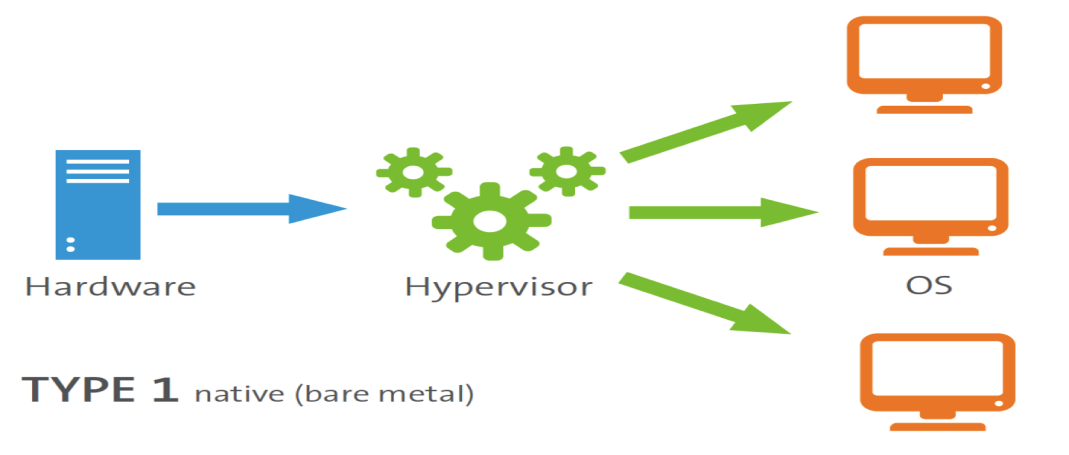
* Server virtualization
* Desktop virtualization
* Application virtualization
* Network virtualization
* Storage virtualization

A Hypervisor or Virtual Machine Monitor(VMM) is computer software, firmware or hardware that creates and runs virtual machines. A computer on which a hypervisor runs one or more virtual machines is called a host machine, and each virtual machine is called guest machine. The hypervisor presents the guest operating systems with a virtual operating platform and manages the execution of the guest operating systems. Multiple instances of a variety of operating systems may share the virtualized hardware resources: for example, Linux, Windows and mac OS instances can all run on a single physical x86 machines.



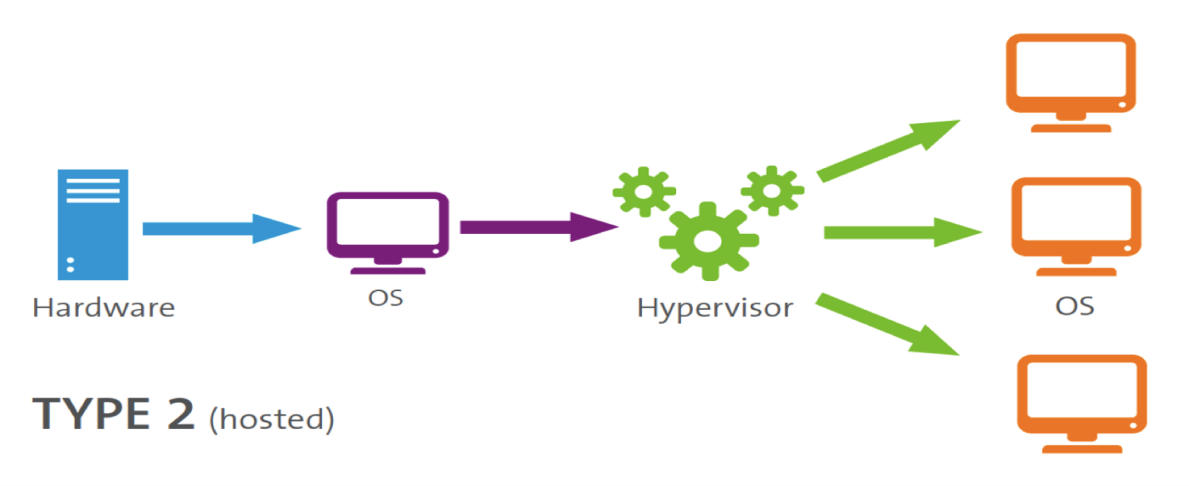
**Types of Hypervisors**

* **Bare-Metal**or**Native**or**Type-I Hypervisor:**



These hypervisors run on the top of physical hardware of the system. Guest OS and applications run on the hypervisor. A major advantage is that any problem in one of the virtual machine or guest operating system does not affect the other guest operating systems running on the hardware. Example: Microsoft Hyper-V hypervisor, VMware, Oracle VM server

* Embedded or Hosted or Type-II Hypervisor:



These hypervisors run within a host OS. That means Type-II Hypervisor run as an application on the host OS. It is completely dependent on Host Operating System for its operations. While having a base operating system allows better specification of policies, *any problem in the base operating system affects the entire system even if the hypervisor running above the base OS is secure.* Example: VMware Workstation, Microsoft Virtual PC, Oracle Virtual Box