OS Fundamentals and Network Infrastructure Implementation

A Technical Consultancy Report

KV5051 Operating Systems

Abdulla AlBassam (23055814)

Word Count: 1632

January 16th, 2025

TABLE OF CONTENTS

WP1: Response to OS Related Questions

Question 1:

In modern operating systems, the process creation mechanism relies on system calls like fork() and exec(). Compare and contrast the fork() system call with the vfork() system call.

What are the advantages and disadvantages of each, particularly in terms of memory management and performance? In what situations would you prefer one over the other?

Both the fork() and vfork() system calls are used to create new processes, but differ in terms of memory management and performance. Fork() creates a child process by duplicating the parent's entire address space, resulting in separate memory spaces for both processes. On the other hand, vfork() ensures that both the parent and child processes use the **same** address. In terms of execution, with fork() both processes happen at the same time, which can be an advantage if the child process needs to perform large computations or operations before executing a new program, preventing interferences. Since it creates a separate memory, any changes made by the child does not affect the parent. Whereas with vfork(), the child process is executed while the parent is halted, until the child process either calls exec() or exits. This can be advantageous in cases where the child process calls exec() immediately in order to run another program, which minimises usage of space and improves efficiency, but may not be best suited for long child processes since if the child process modifies shared variables or memory before calling exec(), it can lead to the corruption of the parent process's state. (geeksforgeeks, 2024)

Question 2:

In the context of synchronization, mutexes and semaphores are widely used to manage access to shared resources. However, improper use can lead to deadlocks.

Explain the difference between mutexes and semaphores. Then, analyse the common causes of deadlocks in multi-threaded programs. Propose solutions or techniques to avoid deadlocks in critical sections.

A mutex is a locking mechanism that allows for only one thread to access a shared resource at a time, for example, imagine you have an online banking system where multiple users are transferring money at the same time. If two threads (two users) try to access and modify the same bank account balance at the same time, they could overwrite each other's changes, leading to incorrect balances. To prevent this, a mutex is used.

On the other hand, a semaphore allows multiple threads to access a shared resource up to a set limit. Take for instance a ticket reseller website where multiple users are trying to purchase tickets for a popular event. The backend database that stores ticket availability can only process 100 concurrent queries. A semaphore is used to ensure that only 100 threads access the database simultaneously. If all threads are in use, additional requests wait until one of the threads finishes and frees up the semaphore. In terms of deadlocks, the four conditions that result in them are:

1. Mutual Exclusion, which is when a resource can only be used by one thread at a time.
2. Hold and Wait, which is when a threat is holding onto one resource while waiting to get another resource.
3. No Pre-emption, which states that resources cannot be forcibly taken from a thread and must release them willingly after completing its task.
4. Circular Wait, which is when each thread is waiting for a resource held by the next thread (thread 1 waiting for 2, thread 2 waiting for 3, and so on)

Solutions to avoiding deadlocks involve targeting these conditions effectively, besides Mutual Exclusion, which is often unavoidable. If one of these conditions is not met, then it is impossible for a deadlock to occur, so focusing on one of the conditions is considered best practice. In terms of Hold and Wait, we can ensure threads acquire all resources at once, or release held resources if waiting. Secondly, with No Pre-emption, we can allow pre-emption and/or allow timeouts on locks, where if the thread cannot acquire a lock within a set time, it retries. Finally, regarding Circular Wait, employing the use of a deadlock detecting algo to identify Circular Waits and recover by terminating or going back to one of the threads in the cycle, releasing the locks and retrying is another option as well. However, in real-life situations, deadlock-free algorithms are implemented to avoid deadlocks entirely, such as the Banker's algorithm commonly used in OS'. (GeeksforGeeks, 2025) (Javapoint, 2024)

WP2: Executive Summary for Manager

Introduction:

This report puts forth a comprehensive design and implementation strategies/code for DNS servers and Apache web servers to ensure high availability, secure communication, and load-balanced services for your business. The setup ensures both cost-efficiency and full operational control, tailored to meet the company's ever-expanding needs.

Benefits of Proposed Design:

1. **Scalability**: The setup can handle increased traffic by adding more DNS or web servers on demand (HostThrive, 2024).
2. **Security**: Secure zone transfers between DNS servers using TSIG, HTTPS ensuring secure web communication (ManageEngine, n.d.).
3. **Reliability**: Load balancing minimizes downtime and distributes traffic evenly. A secondary DNS server provides redundancy (HostThrive, 2024).
4. **Cost Efficiency**: On-location infrastructure reduces dependency on third-party services and avoids reoccurring costs (AvenaCloud, 2024).
5. **Control and Customization**: Full control over DNS and web hosting allows custom configurations and security measures (Vera, 2020).

Servers:

- 1 Primary DNS Server (Tech Solutions Inc.).
- 1 Secondary DNS Server (Tech Solutions Inc. Secondary).
- 2 Apache Web Servers managed by a load balancer.

Overview of the Secure and Load-Balanced System:

This system is crucial for ensuring uninterrupted access to company resources and services. The load balancer evenly distributes incoming traffic to the two web servers, preventing overload on any single server and improving response times for visitors. The implementation of HTTPS secures communication, protecting sensitive information. By integrating DNS and web hosting this way, the system achieves both reliability and security.

Comparison with Other Solutions:

- **Cloud-Based Services**: While cloud-based DNS and web hosting services offer convenience, they come with recurring costs and limited control over configuration and security (Watson, 2024)
- **Single-Server Setups**: They are cost-effective for small networks but lack scalability, redundancy, and reliability, making them unsuitable for a growing company like yours (Larson, 2009).

This report outlines the step-by-step process to configure a DNS system on Linux and a load-balanced Apache web hosting solution. Detailed instructions, explanations, and configuration screenshots are included to help your IT team replicate the setup.

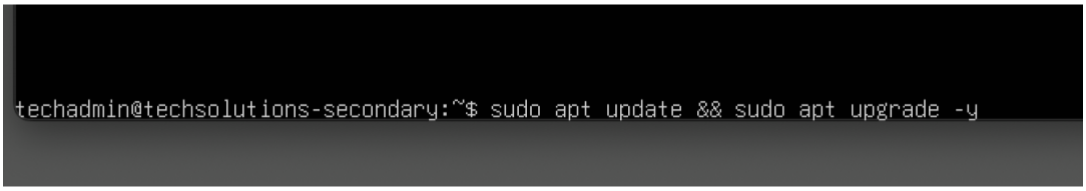
WP3: DNS Infrastructure Setup

Server Setup

For the implementation of the DNS infrastructure, the chosen Linux distribution is **Ubuntu**. Ubuntu was selected for its stability, active community support, and extensive documentation. The DNS servers were installed using BIND9, a widely used and reliable DNS software.

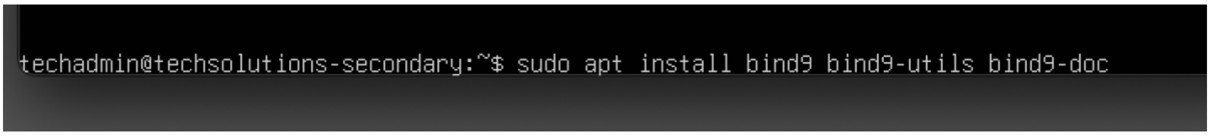
Installation Steps (On primary and secondary servers)

1. **Begin updating system for afficiency**:



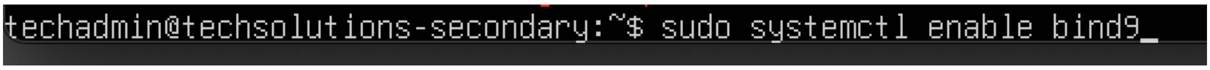
```
techadmin@techsolutions-secondary:~$ sudo apt update && sudo apt upgrade -y
```

2. **Install BIND9 and all necessary utilities**:

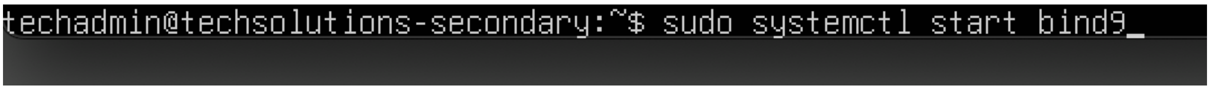


```
techadmin@techsolutions-secondary:~$ sudo apt install bind9 bind9-utils bind9-doc
```

3. **Enable and start BIND9**:



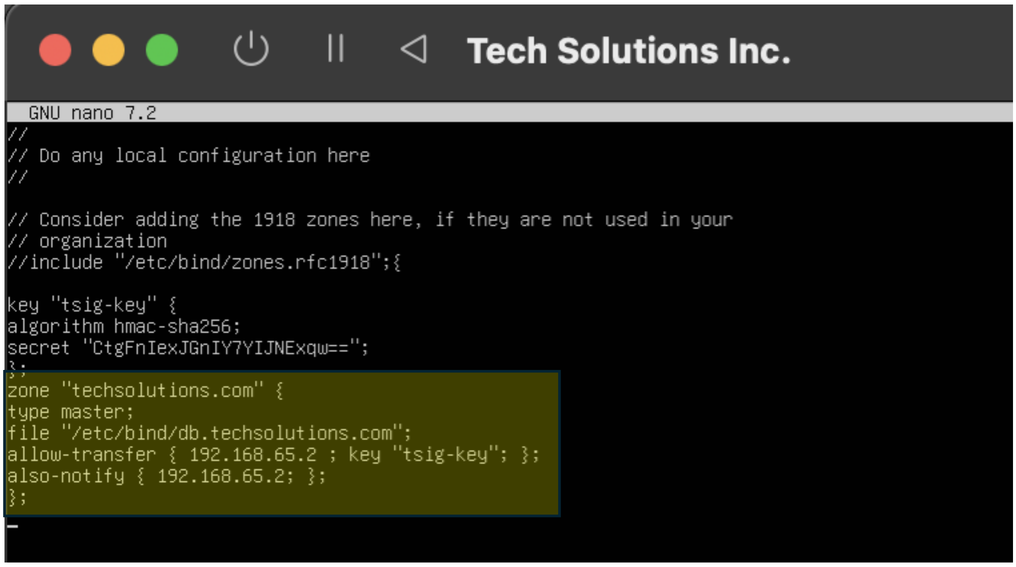
```
techadmin@techsolutions-secondary:~$ sudo systemctl enable bind9_
```



```
techadmin@techsolutions-secondary:~$ sudo systemctl start bind9_
```

Primary Server Zone File Configuration

The primary server is configured to manage the DNS zone techsolutions.com. The zone file is located at sudo nano (nano used on MacOS) /etc/bind/db.techsolutions.com. Zone declaration in sudo nano /etc/bind/named.conf.local. Configurations used:

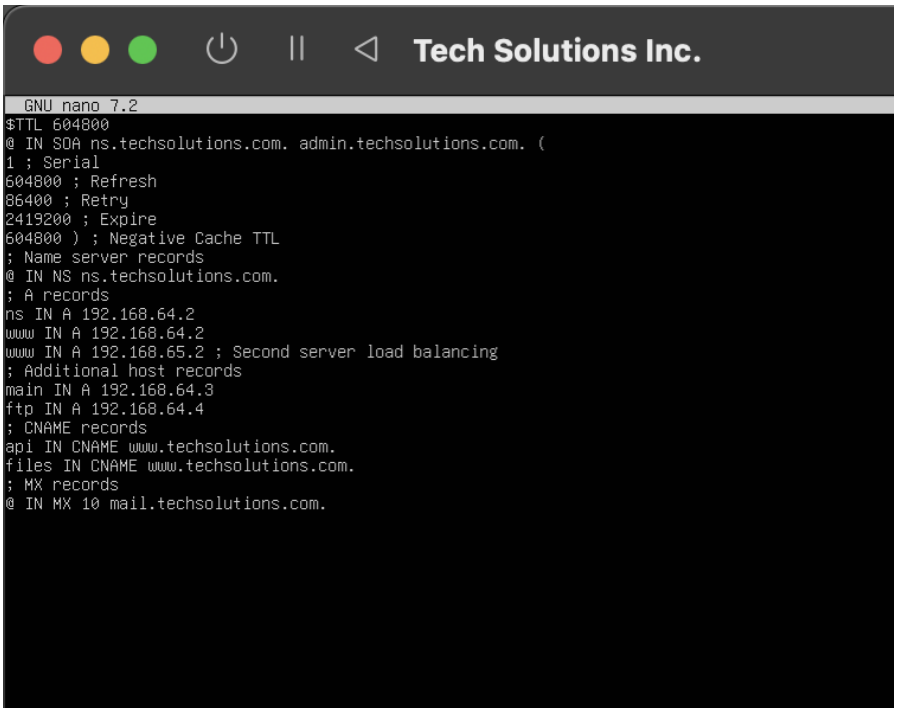


```
  GNU nano 7.2
//
// Do any local configuration here
//

// Consider adding the 1918 zones here, if they are not used in your
// organization
//include "/etc/bind/zones.rfc1918";{

key "tsig-key" {
algorithm hmac-sha256;
secret "CtgFnIexJGnIY7YIJNExqw==";
};
zone "techsolutions.com" {
type master;
file "/etc/bind/db.techsolutions.com";
allow-transfer { 192.168.65.2 ; key "tsig-key"; };
also-notify { 192.168.65.2; };
};
_
```

Zone File at sudo nano /etc/bind/db.techsolutions.com:

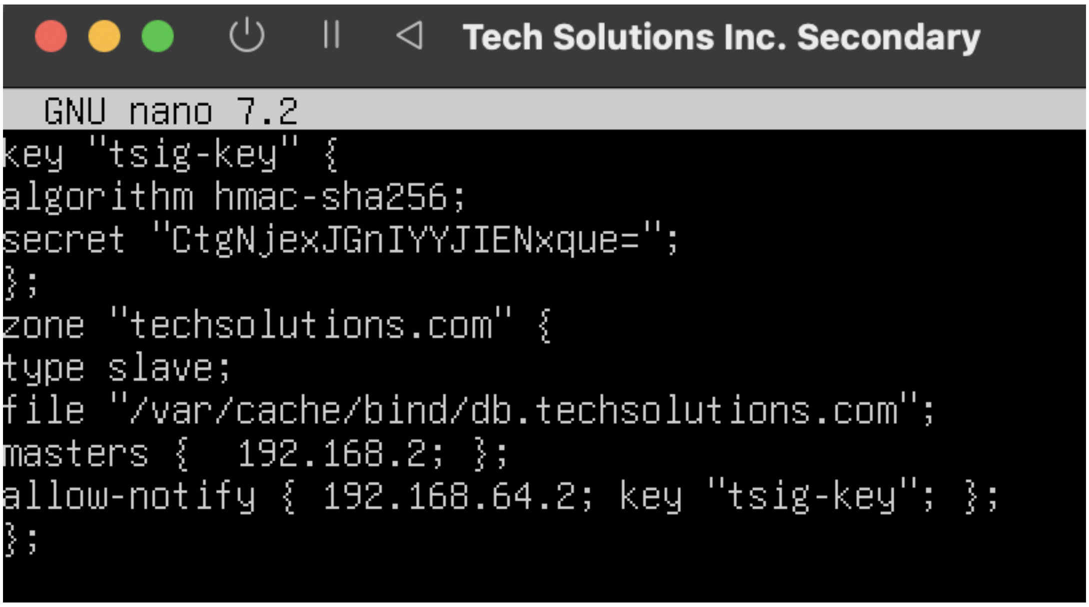


```
  GNU nano 7.2
$TTL 604800
@ IN SOA ns.techsolutions.com. admin.techsolutions.com. (
1 ; Serial
604800 ; Refresh
86400 ; Retry
2419200 ; Expire
604800 ) ; Negative Cache TTL
; Name server records
@ IN NS ns.techsolutions.com.
; A records
ns IN A 192.168.64.2
www IN A 192.168.64.2
www IN A 192.168.65.2 ; Second server load balancing
; Additional host records
main IN A 192.168.64.3
ftp IN A 192.168.64.4
; CNAME records
api IN CNAME www.techsolutions.com.
files IN CNAME www.techsolutions.com.
; MX records
@ IN MX 10 mail.techsolutions.com.
```

Replication Process (Secondary Server)

Below is the configuration for the secondary server configured to receive updates from the primary server:

Zone Declaration in /etc/bind/named.conf.local:



```
  GNU nano 7.2
key "tsig-key" {
algorithm hmac-sha256;
secret "CtgNjexJGnIYYJIENxque=";
};
zone "techsolutions.com" {
type slave;
file "/var/cache/bind/db.techsolutions.com";
masters {  192.168.2; };
allow-notify { 192.168.64.2; key "tsig-key"; };
};
```

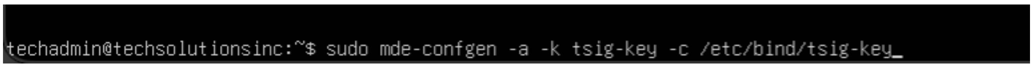
DNS Configuration

Zone Transfers

To ensure the secondary server receives updates from the primary server, configure zone transfers allow-transfer and also-notify in the primary server's zone declaration. Zone transfers are authenticated using a TSIG key to ensure secure communication as well.

The generated key is used in both primary and secondary server configurations. Shorter TSIG key generated for simplicity, otherwise use: "dnssec-keygen -a HMAC-SHA256 -b 256 -n HOST tsig-key"

To Generate a shorter version of TSIG key:

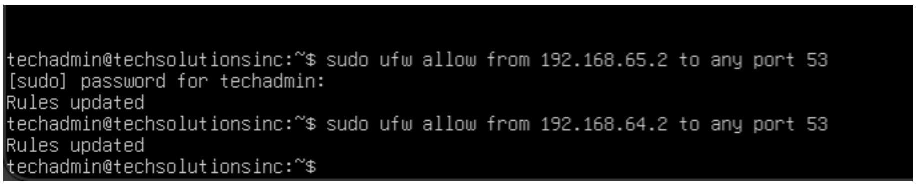


```
techadmin@techsolutionsinc:~$ sudo mde-confgen -a -k tsig-key -c /etc/bind/tsig-key_
```

Security Measures

1. The use of TSIG keys ensures that only authorized servers can request and receive zone transfers.
2. The allow-transfer directive limits zone transfers to the secondary server's IP address.
3. The allow-notify directive restricts notifications to the secondary server only.

Firewall Configuration:



```
techadmin@techsolutionsinc:~$ sudo ufw allow from 192.168.65.2 to any port 53
[sudo] password for techadmin:
Rules updated
techadmin@techsolutionsinc:~$ sudo ufw allow from 192.168.64.2 to any port 53
Rules updated
techadmin@techsolutionsinc:~$
```

Registering Apache Web Servers in DNS

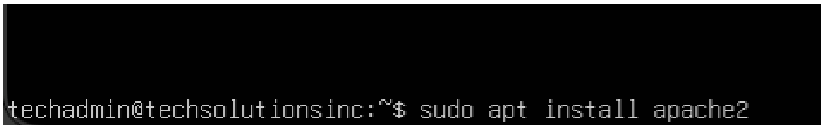
Both Apache web servers are registered in the DNS database with the A records pointing to their IP addresses. The load balancing setup includes both servers under www.techsolutions.com. CNAME and MX records are also configured as shown in the zone file screenshot presented earlier.

WP4: Web Server Infrastructure Setup

Apache Installation

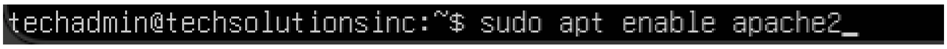
Apache was chosen for its reliability, and performance. Below are the steps to install Apache on Linux servers (Ubuntu 20.04 was used for my setup).

1. **Installation**: Apache was installed using the following command line:



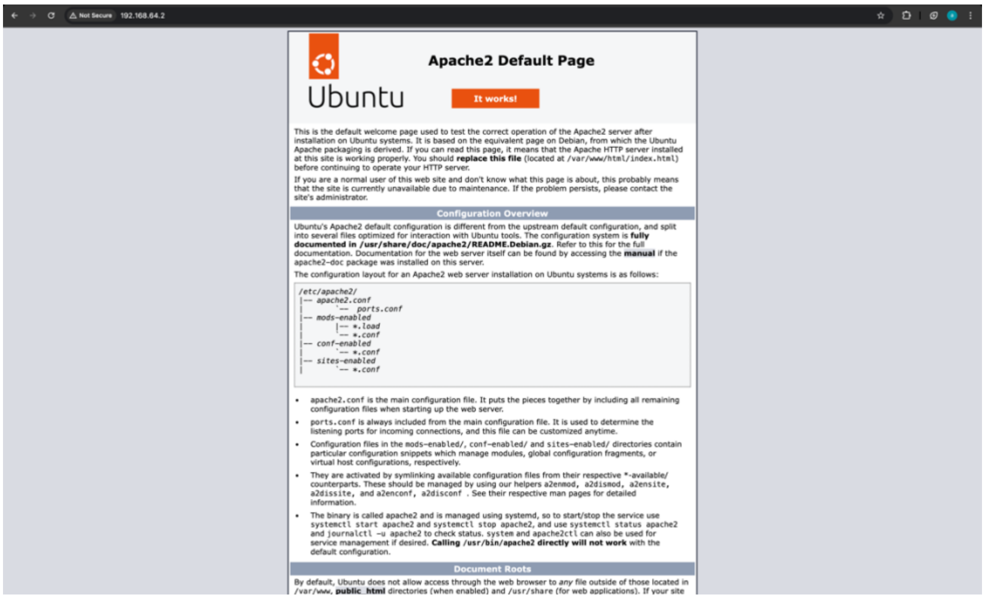
```
techadmin@techsolutionsinc:~$ sudo apt install apache2
```

2. **Service Management**: After installation, the service was started and enabled to run when you boot up:



```
techadmin@techsolutionsinc:~$ sudo apt enable apache2_
```

3. **Looking up the primary server IP address on your local machine should result in this message:**



Apache2 Default Page

It works!

This is the default welcome page used to test the correct operation of the Apache2 server after installation on Ubuntu systems. It is based on the equivalent page on Debian, from which the Ubuntu Apache packaging is derived. If you can read this page, it means that the Apache HTTP server installed at this site is working properly. You should **replace this file** (located at /var/www/html/index.html) before continuing to operate your HTTP server.

If you are a normal user of this web site and don't know what this page is about, this probably means that the site is currently unavailable due to maintenance. If the problem persists, please contact the site's administrator.

Configuration Overview

Ubuntu's Apache2 default configuration is different from the upstream default configuration, and split into several files optimized for interaction with Ubuntu tools. The configuration system is **fully documented in /usr/share/doc/apache2/README.Debian.gz**. Refer to this for the full documentation. Documentation for the web server itself can be found by accessing the **manual** if the apache2-doc package was installed on this server.

The configuration layout for an Apache2 web server installation on Ubuntu systems is as follows:

```
/etc/apache2/
|-- apache2.conf
|       `--  ports.conf
|-- mods-enabled
|       |-- *.load
|       `-- *.conf
|-- conf-enabled
|       `-- *.conf
|-- sites-enabled
|       `-- *.conf
```

- apache2.conf is the main configuration file. It puts the pieces together by including all remaining configuration files when starting up the web server.
- ports.conf is always included from the main configuration file. It is used to determine the listening ports for incoming connections, and this file can be customized anytime.
- Configuration files in the mods-enabled/, conf-enabled/ and sites-enabled/ directories contain particular configuration snippets which manage modules, global configuration fragments, or virtual host configurations, respectively.
- They are activated by symlinking available configuration files from their respective *-available/ counterparts. These should be managed by using our helpers a2enmod, a2dismod, a2ensite, a2dissite, and a2enconf, a2disconf . See their respective man pages for detailed information.
- The binary is called apache2 and is managed using systemd, so to start/stop the service use systemctl start apache2 and systemctl stop apache2, and use systemctl status apache2 and journalctl -u apache2 to check status. system and apache2ctl can also be used for service management if desired. **Calling /usr/bin/apache2 directly will not work** with the default configuration.

Document Roots

By default, Ubuntu does not allow access through the web browser to *any* file outside of those located in /var/www, **public_html** directories (when enabled) and /usr/share (for web applications). If your site

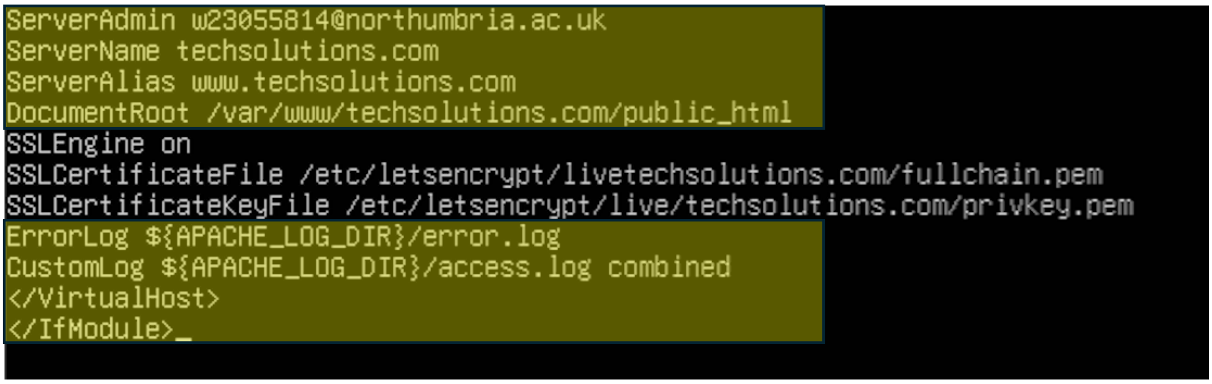
Virtual Host Configuration

Virtual hosts were configured to display web content for the domain techsolutions.com and its subdomain www.techsolutions.com. This configuration guarantees proper hosting and request handling.

1. **Virtual Host File**:

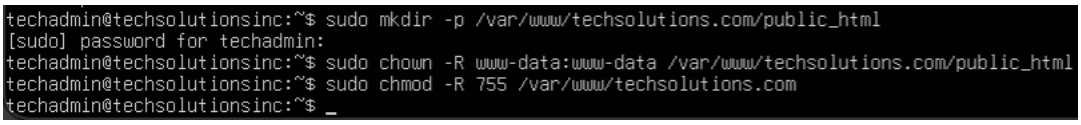
/etc/apache2/sites-available/techsolutions.com.conf

The configuration file includes the following:



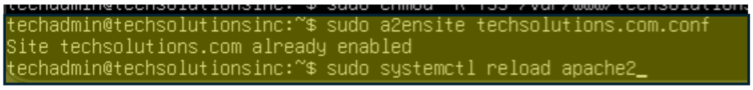
```
ServerAdmin w23055814@northumbria.ac.uk
ServerName techsolutions.com
ServerAlias www.techsolutions.com
DocumentRoot /var/www/techsolutions.com/public_html
SSLEngine on
SSLCertificateFile /etc/letsencrypt/livetechsolutions.com/fullchain.pem
SSLCertificateKeyFile /etc/letsencrypt/live/techsolutions.com/privkey.pem
ErrorLog ${APACHE_LOG_DIR}/error.log
CustomLog ${APACHE_LOG_DIR}/access.log combined
</VirtualHost>
</IfModule>_
```

2. **Directory Setup**: The document root /var/www/techsolutions.com/public_html was created, and permissions were changed to meet reqs:



```
techadmin@techsolutionsinc:~$ sudo mkdir -p /var/www/techsolutions.com/public_html
[sudo] password for techadmin:
techadmin@techsolutionsinc:~$ sudo chown -R www-data:www-data /var/www/techsolutions.com/public_html
techadmin@techsolutionsinc:~$ sudo chmod -R 755 /var/www/techsolutions.com
techadmin@techsolutionsinc:~$ _
```

3. **Enable the Virtual Host**: The virtual host was enabled using the line below, reloading to ensure changes occur:



```
techadmin@techsolutionsinc:~$ sudo a2ensite techsolutions.com.conf
Site techsolutions.com already enabled
techadmin@techsolutionsinc:~$ sudo systemctl reload apache2_
```

Load Balanced & Secured Configuration

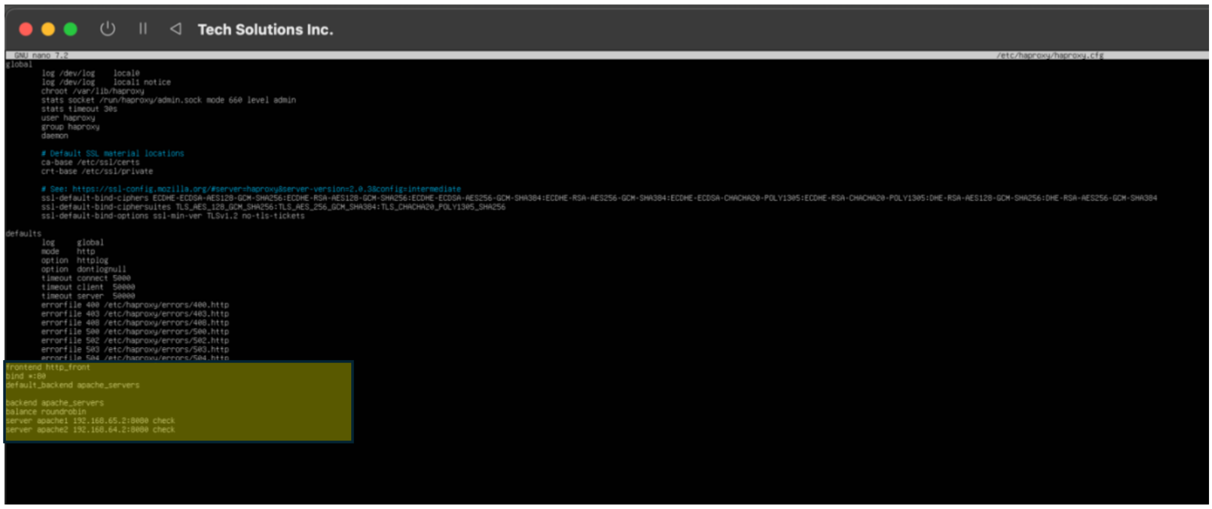
Load Balancing

To ensure high availability and distribute traffic evenly across the servers, HAProxy was configured as a load balancer. It is a free, open-sourced load balancer that can ensure high performance 24/7, as well as possible scalability options and advanced security features for your growing business (LogicMonitor, 2024).

1. **HAProxy Configuration**: The configuration file is located at:

/etc/haproxy/haproxy.cfg

The configuration includes:



```
  GNU nano 7.2                                                                                    /etc/haproxy/haproxy.cfg
global
        log /dev/log    local0
        log /dev/log    local1 notice
        chroot /var/lib/haproxy
        stats socket /run/haproxy/admin.sock mode 660 level admin
        stats timeout 30s
        user haproxy
        group haproxy
        daemon

        # Default SSL material locations
        ca-base /etc/ssl/certs
        crt-base /etc/ssl/private

        # See: https://ssl-config.mozilla.org/#server=haproxy&server-version=2.0.3&config=intermediate
        ssl-default-bind-ciphers ECDHE-ECDSA-AES128-GCM-SHA256:ECDHE-RSA-AES128-GCM-SHA256:ECDHE-ECDSA-AES256-GCM-SHA384:ECDHE-RSA-AES256-GCM-SHA384:ECDHE-ECDSA-CHACHA20-POLY1305:ECDHE-RSA-CHACHA20-POLY1305:DHE-RSA-AES128-GCM-SHA256:DHE-RSA-AES256-GCM-SHA384
        ssl-default-bind-ciphersuites TLS_AES_128_GCM_SHA256:TLS_AES_256_GCM_SHA384:TLS_CHACHA20_POLY1305_SHA256
        ssl-default-bind-options ssl-min-ver TLSv1.2 no-tls-tickets

defaults
        log     global
        mode    http
        option  httplog
        option  dontlognull
        timeout connect 5000
        timeout client  50000
        timeout server  50000
        errorfile 400 /etc/haproxy/errors/400.http
        errorfile 403 /etc/haproxy/errors/403.http
        errorfile 408 /etc/haproxy/errors/408.http
        errorfile 500 /etc/haproxy/errors/500.http
        errorfile 502 /etc/haproxy/errors/502.http
        errorfile 503 /etc/haproxy/errors/503.http
        errorfile 504 /etc/haproxy/errors/504.http
frontend http_front
bind *:80
default_backend apache_servers

backend apache_servers
balance roundrobin
server apache1 192.168.65.2:8080 check
server apache2 192.168.64.2:8080 check
```

2. **Traffic Distribution**: The balance directive “roundrobin” in the HAProxy configuration ensures traffic is distributed evenly between the Apache servers (apache1 and apache2).

3. **HAProxy Restart**: Changes were applied using the command “sudo systemctl restart haproxy”.

Securing Communication

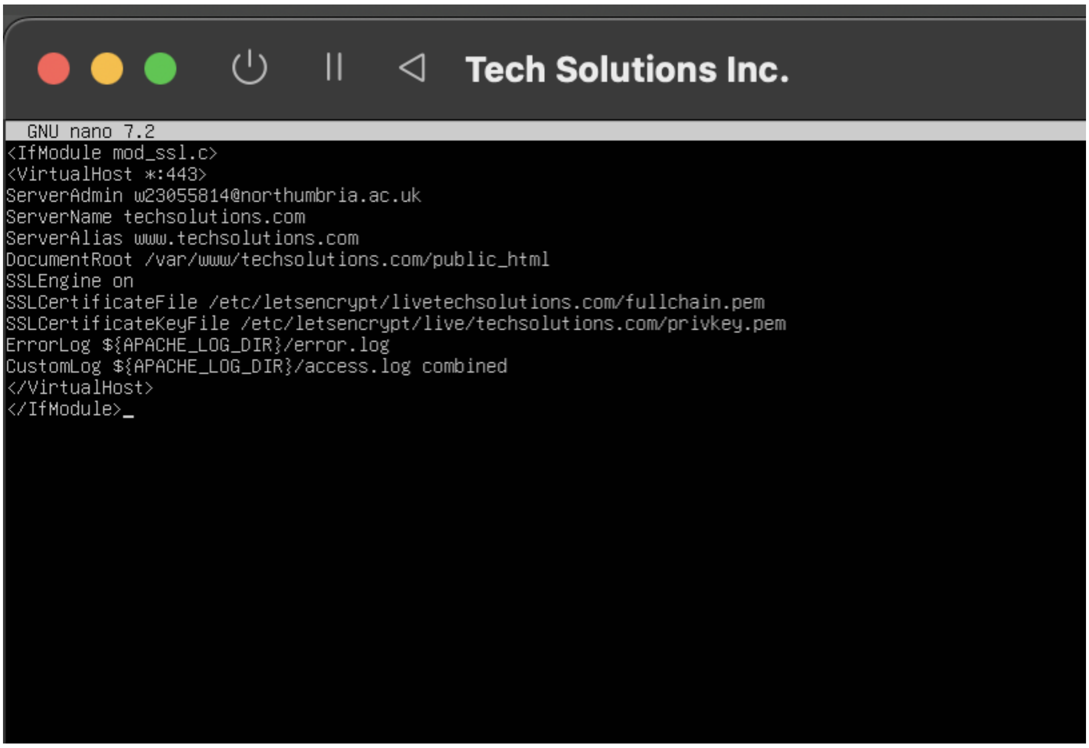
To secure web traffic, SSL was configured using Certbot and Let’s Encrypt.

Let's Encrypt is commonly used since it provides SSL certificates (making secure HTTPS accessible) at extra no cost. It also offers robust encryption, ensuring secure communication between users between all servers (Kothari, 2024).

1. **SSL Certificate**: Certbot was used to generate SSL certificates for techsolutions.com with the command: sudo certbot --apache

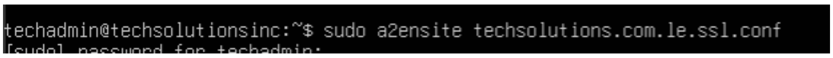
2. **SSL Virtual Host**: The SSL configuration file is located at: sudo nano /etc/apache2/sites-available/techsolutions.com-le-ssl.conf

Configuration:

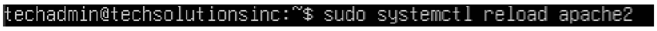


```
  GNU nano 7.2
<IfModule mod_ssl.c>
<VirtualHost *:443>
ServerAdmin w23055814@northumbria.ac.uk
ServerName techsolutions.com
ServerAlias www.techsolutions.com
DocumentRoot /var/www/techsolutions.com/public_html
SSLEngine on
SSLCertificateFile /etc/letsencrypt/livetechsolutions.com/fullchain.pem
SSLCertificateKeyFile /etc/letsencrypt/live/techsolutions.com/privkey.pem
ErrorLog ${APACHE_LOG_DIR}/error.log
CustomLog ${APACHE_LOG_DIR}/access.log combined
</VirtualHost>
</IfModule>_
```

3. **Enabling SSL Site**: The SSL configuration was enabled using:



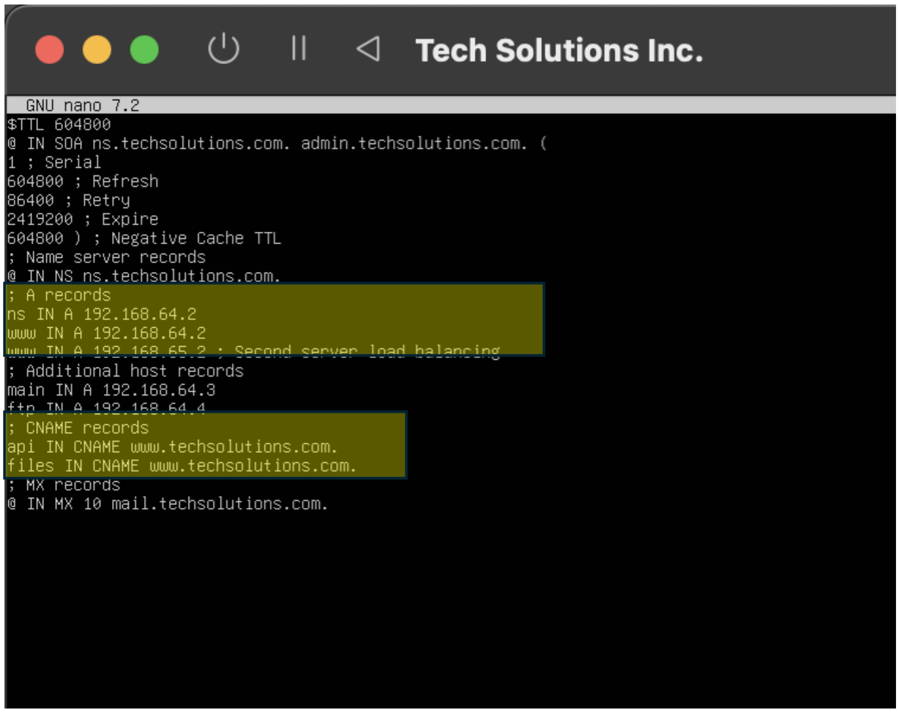
```
techadmin@techsolutionsinc:~$ sudo a2ensite techsolutions.com.le.ssl.conf
[sudo] password for techadmin:
```



```
techadmin@techsolutionsinc:~$ sudo systemctl reload apache2
```

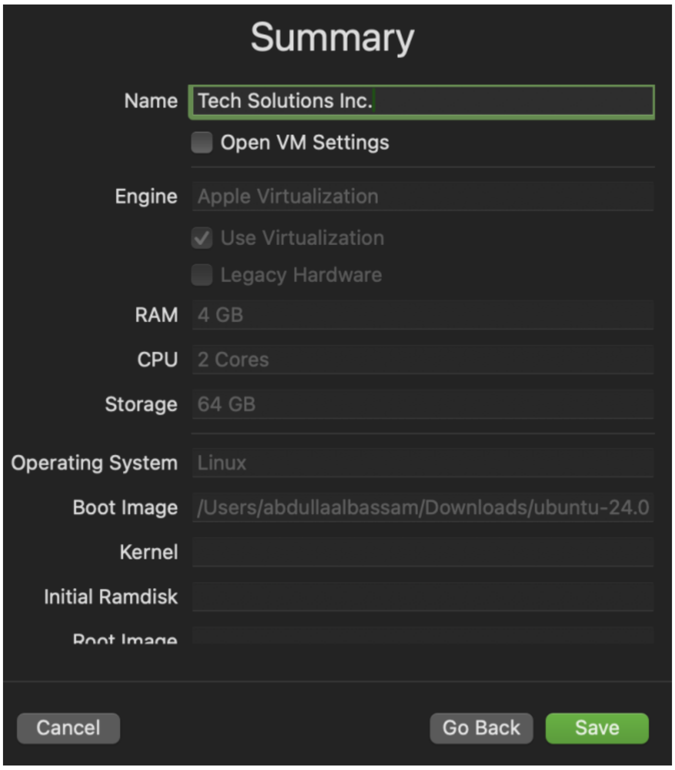
DNS Records for Load Balancer

The DNS records were configured to point the domain techsolutions.com to the IP address of the load balancer. The configuration in the DNS zone file:



```
  GNU nano 7.2
$TTL 604800
@ IN SOA ns.techsolutions.com. admin.techsolutions.com. (
1 ; Serial
604800 ; Refresh
86400 ; Retry
2419200 ; Expire
604800 ) ; Negative Cache TTL
; Name server records
@ IN NS ns.techsolutions.com.
; A records
ns IN A 192.168.64.2
www IN A 192.168.64.2
www IN A 192.168.65.2 ; Second server load balancing
; Additional host records
main IN A 192.168.64.3
ftp IN A 192.168.64.4
; CNAME records
api IN CNAME www.techsolutions.com.
files IN CNAME www.techsolutions.com.
; MX records
@ IN MX 10 mail.techsolutions.com.
```

Appendix: Additional Screenshot(s):



Summary

Name: Tech Solutions Inc.

□ Open VM Settings

Engine: Apple Virtualization

☑ Use Virtualization

□ Legacy Hardware

RAM: 4 GB

CPU: 2 Cores

Storage: 64 GB

Operating System: Linux

Boot Image: /Users/abdullaalbassam/Downloads/ubuntu-24.0

Kernel:

Initial Ramdisk:

Root Image:

References:

1. **Ace Cloud Hosting**, n.d., *In-house server vs cloud hosting*. Available at: https://www.acecloudhosting.com/blog/in-house-server-vs-cloud-hosting/ [Accessed 9 January 2025].

2. **AvenaCloud.com**, 2020, *How to scale your dedicated server for traffic surges*. Available at: https://avenacloud.com/blog/how-to-scale-your-dedicated-server-for-traffic-surges-2?utm_source=chatgpt.com [Accessed 12 January 2025].

3. **GeeksforGeeks.org**, 2025, *Deadlock Prevention*. Available at: https://www.geeksforgeeks.org/deadlock-prevention/ [Accessed 5 January 2025].

4. **GeeksforGeeks.org**, 2025, *Difference between fork() and vfork()*. Available at: https://www.geeksforgeeks.org/difference-between-fork-and-vfork/ [Accessed 9 January 2025].

5. **HostingChacha.com**, n.d., *The ultimate guide to Let's Encrypt SSL: pros and cons*. Available at: https://hostingchacha.com/tutorials/the-ultimate-guide-to-lets-encrypt-ssl-pros-cons/?utm_source=chatgpt.com [Accessed 11 January 2025].

6. **HostThrive.com**, 2024, *How to implement server load balancing for high-traffic websites*. Available at: https://hostthrive.com/server-management/how-to-implement-server-load-balancing-for-high-traffic-websites/?utm_source=chatgpt.com [Accessed 5 January 2025].

7. **JavaTPoint.com**, 2024, *OS Deadlock Prevention*. Available at: https://www.javatpoint.com/os-deadlock-prevention [Accessed 9 January 2025].

8. **Larson**, 2009, *Overview of Single vs Multi-Server Architecture*. Available at: https://lethain.com/overview-of-single-vs-multi-server-architecture/ [Accessed 9 January 2025].

9. **LinuxUnbound.com**, 2024, *Securing DNS transactions with transactional signatures (TSIG)*. Available at: https://www.linuxunbound.com/2020/10/05/securing-dns-transactions-with-transactional-signatures-tsig/?utm_source=chatgpt.com [Accessed 9 January 2025].

10. LogicMonitor, 2023. What is HAProxy and what is it used for? [online] Available at: https://www.logicmonitor.com/blog/what-is-haproxy-and-what-is-it-used-for [Accessed 14 January 2025].

11. **ManageEngine**, n.d., *Configuring TSIG Keys*. Available at: https://pitstop.manageengine.com/portal/en/kb/articles/configuring-tsig-keys?utm_source=chatgpt.com#TSIG_Transaction_Signature [Accessed 5 January 2025].