Implementation of a Secure WordPress Tech Blog on Microsoft Azure

Murdoch University

Bachelor of Information Technology

ICT171 Introduction to Server Environments and Architecture

Clark Carpentero

34180393

09 JUNE 2025

Table of Contents

Chapter 1: Project Overview

1.1 Introduction

This document provides an outline of the setup and configuration of a personal technology blog server.

- **Student Name:** Clark Carpentero
- **Student Number:** 34180393

1.2 Server Access Details

The live server provided can be accessed using the following details.

- **Public IP Address:** 20.11.57.75
- **Domain Name:** https://www.thecarpenteros.com/

1.3 Project Goal

The primary goal of this project is to establish a personal internet presence that showcases my technical capabilities and serves as a long-term platform for sharing technology updates and insights. The blog will focus on analysing and discussing the latest technology trends from industry leaders, including Apple (macOS, iOS), Microsoft (Windows), and Google (Android), as well as developments in the broader personal computing space.

This project is an opportunity to develop proficiency with the Linux command line, implement a server using Infrastructure as a Service (IaaS), and document the process thoroughly. The sole aim is to create a functional and reliable web platform that not only shares valuable tech insights but also demonstrates the skills required to build, configure, and manually manage a live web server. Ultimately, this server is intended as a long-term project that I can continue to develop and expand upon.

Chapter 2: Cloud Infrastructure Setup (IaaS)

This chapter details the selection of Microsoft Azure as the Infrastructure as a Service (IaaS) provider and the subsequent steps taken to provision and perform the initial configuration of the cloud server for the project. The primary goal is to utilise an IaaS platform to configure the server environment manually.

2.1 Provider Choice

For this given project, I have chosen Microsoft Azure.

This provider was selected to gain hands-on experience with a significant, enterprise-grade cloud platform widely used in the industry. The assignment also encourages the use of Azure as the platform; a key learning outcome is to implement servers using such virtualised infrastructure.

2.2 Server Creation

The server, an Azure Virtual Machine (VM), was provisioned with the following specifications. This documentation aims to be detailed enough to allow a colleague or fellow student to replicate the server build precisely and concisely.

- **Distribution:** Linux Ubuntu 24.04 LTS
- **Plan:** Standard_D2s_v3
- **Size:** 2 vcpus, 8 GiB memory
- **Region:** Zone 1, Australia East

2.3 The process for creating the virtual machine was as follows:

1. **Resource Group Creation:** A new or existing resource group was used/created in the Azure portal to organise all the resources for this project.

2. **VM Configuration:** I navigated to “Virtual machines” and clicked “Create”. The subscription, resource group, and virtual machine name, clarkTechBlog, were provided. The region and the specified Ubuntu Server image were selected from the Azure Marketplace.

3. **Authentication:** For secure access, the authentication type was set to SSH public key. A new key pair was generated, and the public key was stored in Azure, while the private key was saved locally for connecting with an SSH Client.

4. **Networking:** During setup, the inbound port rules for the Network Security Group (NSG) were configured to allow traffic for:

    i. HTTP (port 80)
    ii. HTTPS (port 443
    iii. SSH (port 22) – This is essential for accessing and managing web servers remotely via the command line.

5. **VM Deployment:** After reviewing the settings, the virtual machine was deployed. Azure then provisioned the server and assigned it a public address.

Chapter 3: Server Software Installation & Configuration

This chapter covers the manual installation and configuration if the web server software stack required to run a WordPress website. The assignment requires the server software to be configured and deployed manually, not from a pre-configured image. The chose stack is L.A.M.P. (Linux, Apache, MySQL, PHP) on the Ubuntu operating System.

3.1 Web Server Installation (Apache2)

Apache2 is a widely used, robust web server. It was installed using Ubuntu's package manager, apt (on the console).

1. **Install Apache2:** The following command was used to install Apache

   ```
   # Install the apache2 package
   sudo apt install apache2 -y
   ```

2. **Verify Installation:** To confirm Apache was installed and running correctly, accessed my server's public IP address in a web browser. The default Ubuntu Apache welcome page was displayed, also checked the service status to ensure it was active.

   ```
   # Check if the apache2 service is active
   sudo systemctl status apache2
   ```

3.2 Database Installation (MySQL Server)

WordPress requires a database to store all site content, user information, and settings. MySQL is the standard database server for this purpose.

1. **Install MySQL Server:**

   ```
   # Install the mysql-server package
   sudo apt install mysql-server -y
   ```

2. **Secure the Installation:** A security script was run immediately after installation to set a root password, remove anonymous users, and disable remote root login. This is a critical step for securing the database.

```
# Run the interactive security script

sudo mysql_secure_installation
```

3.3 PHP Installation

PHP is the server-side scripting language that WordPress is built on. It processes code to generate the dynamic content for the website.

1. **Install PHP and necessary extensions:** The core PHP package, the Apache module for PHP, and common extensions needed by WordPress were isntalled with a single command.

```
# Install PHP, the Apache module, and required extensions

sudo apt install php libapache2-mod-php php-mysql php-curl php-gd php-mbstring php-xml php-xmlrpc php-soap php-intl php-zip -y
```

2. **Verify PHP Installation:** To check the PHP version, I used the following command:

```
# Display the installed PHP version
php -v
```

3.4 WordPress Installation and Configuration

With the server stack in place, the final step was to install WordPress itself.

1. **Create a Database for WordPress:** I logged into MySQL and created a dedicated database and user for the WordPress installation.

```
-- Log into MySQL as the root user

sudo mysql -u root -p
```

```
-- Create a new database

CREATE DATABASE wordpress_db;


-- Create a new user and set a secure password

CREATE USER 'wp_user'@'localhost' IDENTIFIED BY 'ICT171_Password';


-- Grant all privileges on the database to the new user

GRANT ALL PRIVILEGES ON wordpress_db.* TO 'wp_user'@'localhost';


-- Flush privileges to apply the changes

FLUSH PRIVILEGES;


-- Exit MySQL

EXIT;
```

2. **Download WordPress:** I navigate to the /tmp directory and downloaded the latest version of WordPress.

```
# Move to the temporary directory

cd /tmp


# Download the WordPress tarball

wget https://wordpress.org/latest.tar.gz
```

3. **Extract and Prepare Files:** The downloaded archive was extracted, and the sample configuration file was copied to create the active configuration file.

```
# Extract the archive
```

```
tar -xzvf latest.tar.gz


# Move into the new WordPress directory
cd /tmp/wordpress


# Copy the sample config file to create the actual config file
cp wp-config-sample.php wp-config.php
```

4. **Configure wp-config.php:** I edited the wp-config.php file using a text editor (like nano) to add the database details (database name, username, and password) created in the first step.

```
# Edit the config file with a text editor
nano wp-config.php
```

5. **Move Files to Web Root:** The WordPress files were moved to Apache's default web root directory, /var/www/html/. It's essential to set the correct ownership and permissions so the web server can manage the files.

```
# Use rsync to copy the WordPress files to the web root
sudo rsync -av /tmp/wordpress/ /var/www/html/


# Set correct ownership and permissions for the web server
sudo chown -R www-data:www-data /var/www/html/
sudo chmod -R 755 /var/www/html/
```

6. **Finalize Installation:** The installation was completed through the web interface by navigating to my server's domain name or IP address in a browser. This final step involved setting the site title, creating an admin user, and logging into the WordPress dashboard for the first time.

Chapter 4: DNS and Domain Name Configuration

This chapter documents the process of linking the server's static public IP address, assigned by Microsoft Azure, to a human-readable domain name. This step is essential for making the website professionally accessible and is a core requirement of the project outlined.

4.1 Domain Name Registration

To establish a permanent address for the web server, a custom domain name was registered for this project.

- **Domain Name:** thecarpenteros.com
- **Domain Registrar:** GoDaddy

The domain was registered for a period of three years. This registrar was chosen because of its competitive pricing for a span of one year's subscription price, ease of use as easy to familiar with their system, positive reviews from well-known companies around the world, and the productivity they offer and add-ons you can acquire towards your current subscription whenever needed.

4.2 DNS Configuration

After registering the domain, the Domain Name System (DNS) records were configured to point the domain to the Azure server's public IP address. This was achieved by creating an 'A' record. An 'A' record maps a domain name directly to an IPv4 address.

The Process for configuring the DNS record was as follows:

1. **Log in to Registrar:** I logged in to my **GoDaddy** account.
2. **Navigate to DNS Management:** I navigated to the domain management panel for **thecarpenteros.com** and selected the "DNS Settings" or Advanced DNS" option.
3. **Create 'A' Record:** I created a new DNS record with the following details:
    a. **Type:** A
    b. **Host/Name:** @ (This symbol is standard for pointing to the root domain itself, e.g., thecarpenteros.com).
    c. **Value/Points to:** 20.11.57.75
    d. **TTL (Time To Live):** Set to Automatic or 30 minutes

4. **Create CNAME Record for ‘www’ (Optional but Recommended):** To ensure that www.thecarpenteros.com also works, a CNAME record was created to point the www subdomain to the root domain.
    a. **Type:** CNAME
    b. **Host/Name:** www
    c. **Value/Points to:** thecarpenteros.com
    d. **TTL (Time To Live):** Set to Automatic or 30 minutes

After saving these changes, I waited for DNS propagation, which can take anywhere from a few minutes to a few hours. Once propagated, typing http://www.thecarpenteros.com into a web browser successfully loaded the Apache2 default page hosted on the Azure virtual machine. This confirms the DNS was configured correctly.

Chapter 5: Security (SSL/TLS Configuration)

This chapter outlines the process of securing the web server with an SSL/TLS certificate, enabling HTTPS. Implementing HTTPS is crucial for website security, as it encrypts the data exchanged between the client's browser and the server, thereby protecting user privacy and establishing trust. The manual setup of SSL/TLS is undertaken here to meet the assignment's criteria. The free and trusted certificate authority Let's Encrypt will be used for this project.

5.1 Installing Certbot

Certbot is a client that automates the process of obtaining and renewing Let's Encrypt SSL certificates. It was installed along with its Apache plugin to streamline the configuration.

1. **Install Certbot and Apache Plugin:** The following command was used to install Certbot and the specific Apache plugin, which enables Certbot to configure the server's SSL settings automatically.

```
# Install certbot and the python3-certbot-apache package

sudo apt install certbot python3-certbot-apache
```

5.2 Obtaining and Installing the SSL Certificate

With Certbot installed, a single command was used to obtain the certificate for the domain and have Certbot automatically edit the Apache configuration to use it.

1. **Run Certbot for Apache:** the command below initiates the process. Certbot automatically detects the domain name from the Apache Configuration.

```
# Run certbot for the Apache plugin

sudo certbot --apache
```

2. **Follow Interactive Prompts:** Certbot presented a series of prompts to configure the certificate.

   a. **Email Address:** An email address was provided for urgent renewal and security notices.
   b. **Terms of Service:** I agreed to the Let's Encrypt terms of service.
   c. **Domain Selection:** Certbot listed the domains configured in Apache. I selected the domain to be secure.

3. **Automatic Configuration:** Once the domain was verified, Certbot installed the certificate and prompted to redirect all HTTP traffic to HTTPS automatically. I selected the redirect option for enhanced security.

5.3 Verifying SSL/TLS Configuration

After Certbot completed the installation, the configuration was verified in two ways:

1. **Browser Test:** I navigated to https://www.thecarpenteros.com/ and confirmed that the browser showed a padlock icon in the address bar, indicating a secure connection and a valid certificate.
2. **Online SSL Test:** I used an online service, such as Qualys SSL Labs, to test the server's SSL configuration, which confirmed an 'A' grade rating.

5.4 Automatic Certificate Renewal

The Certbot packaged automatically configures a scheduled task (a system timer or cronjob) to renew the certificates before they expire. The renewal process was tested to ensure it would run without issues.

```
# Perform a "dry run" to test the renewal process without making changes sudo
certbot renew --dry-run
```

The successful output of this command confirms that the certificate will be renewed automatically, ensuring the site remains secure without manual intervention.

Chapter 6: Scripting

This chapter details the custom script developed for this project. The assignment requires the development of a script in a language of choosing to demonstrate programming proficiency. To add useful and relevant function to the server, I have created a script that automates the backup process for the WordPress website.

6.1 Script Purpose

The purpose of this script is to automatically create a complte backup of the WordPress site. This includes two critical components:

1. **The WordPress Database:** The script performs a full dump of the MySQL database, which contains all posts, pages, comments, and settings.
2. **The WordPress Files:** The script creates a compressed archive of the entire directory, which includes the WordPress core files, themes, plugins, and uploaded media.

The script then combines these two components into a single, timestamped file for easy storage and restoration. This provides a simple and effective disaster recovery solution, which is a vital task for managing a live website. The script is designed to be run automatically by a cron job, ensuring regular backups are taken without manual intervention needed.

6.2 Script Code

Below is the commented code for the backup_wordpress.sh script. The script is designed to be easy to read and modify.

```
# This script creates a compressed archive of the WordPress files

# and a backup of the MySQL database.


# --- Configuration ---


# Directory to store the backups

BACKUP_DIR="/var/backups/wordpress"
```

```
# WordPress installation directory
WP_DIR="/var/www/html"

# MySQL database name
DB_NAME="wordpress_db"

# MySQL database user
DB_USER="wp_user"

# MySQL database password
DB_PASS="ICT171_Password"

# Timestamp for the backup file (e.g., 2025-06-09)
TIMESTAMP=$(date +"%F")

# --- Script Execution ---

# Create the backup directory if it doesn't exist
mkdir -p "$BACKUP_DIR"

echo "Starting WordPress backup..."

# 1. Backup the MySQL database
echo "Backing up the database: $DB_NAME..."
mysqldump -u "$DB_USER" -p"$DB_PASS" "$DB_NAME" >
"$BACKUP_DIR/$DB_NAME-$TIMESTAMP.sql"
```

```
# 2. Archive the WordPress files
echo "Archiving the WordPress directory: $WP_DIR..."
tar -czf "$BACKUP_DIR/wordpress-files-$TIMESTAMP.tar.gz" -C /var/www/ html .

echo "Backup complete!"
echo "Files located in: $BACKUP_DIR"
```

6.3 Output Verification

The output of this script can be easily independently verified.

1. **Verify output files:** After running the script, verify the backup files by listing the contents of the WordPress backup directory.
   a. ls -l /var/backups/wordpress/

   The expected output is a list of files showing the SQL database backup and the compressed archive of the website files, both with the current date and time as a timestamp.

This provides a clear, verifiable evidence that the script has executed successfully and create the necessary backup artifacts.

Chapter 7: Project Documentation and Version Control

This chapter describes the methodology used for documenting this project and managing its versions. A key learning outcome for this is to develop proficiency in using GitHub for version control and to report a project effectively.

7.1 Version Control with GitHub

In line with modern development and IT administration practices, this entire project, including all documentation, is managed on a GitHub repository. Using GitHub provides version history, a single source of truth for the project's status, and fulfils a core requirement of the submission process.

- **Repository Link:** The project repository can be found at:
    - https://github.com/TheKing277/ICT171_Assignment_2.git

7.2 Documentation Approach

The documentation for this server has been written with a specific audience in mind: an IT colleague or fellow student who might need to replicate the server build from scratch. The guiding principle was to create a document so complete that the server could be reimplemented in under an hour.

The following standards were used:

- **Format:** The documentation was written in Markdown directly within the GitHub repository, as strongly advised in the unit materials. This approach is more flexible and version-control-friendly than a static PDF or Word document.
- **Clarity:** Care was taken to separate narrative explanations from technical commands, using Markdown's code block formatting. This ensures that commands can be easily copied and pasted.
- **Completeness:** The documentation aims to be a standalone guide, meaning another ICT171 student could replicate the server without needing to perform independent research. Any external guides or resources used are credited in the Reference chapter.
- **Entry Point:** The main README.md file in the GitHub repository serves as the primary entry point, providing an overview of the project and a link to the live server domain.

Chapter 8: Video Explainer

As per the assignment submission checklist, a video explainer has been created to demonstrate the project and its development.

The video provides a tour of the live website, discusses the server architecture, and shows the evidence of the iterative improvements made to the project over several weeks. The quality of the video and the clarity of the student's identity have been considered in accordance with the marking rubric.

- **Link to Video Explainer: Anyone with the link can view**
    - **https://drive.google.com/file/d/1VL-PUQgDS6soXDLCGnFq7LHYg9K7AF9Q/view?usp=share_link**

Chapter 9: References

This chapter lists all the external guides, tutorials, and documentation that were consulted during the setup and configuration of this server. The project documentation I designed to be a standalone guide, and these references are provided for attribution and to acknowledge the resources that informed the process.

DigitalOcean. (2022, June 1). *How To Install WordPress on Ubuntu 22.04 with a LAMP Stack*. DigitalOcean Community. Retrieved June 9, 2025, from https://www.digitalocean.com/community/tutorials/how-to-install-wordpress-on-ubuntu-22-04-with-a-lamp-stack

Let's Encrypt. (n.d.). *Certbot Instructions*. Electronic Frontier Foundation. Retrieved June 9, 2025, from https://certbot.eff.org/instructions

Microsoft. (2023, October 12). *Quickstart: Create a Linux virtual machine in the Azure portal*. Microsoft Learn. Retrieved June 9, 2025, from https://learn.microsoft.com/en-us/azure/virtual-machines/linux/quick-create-portal

WordPress. (n.d.). *Installing WordPress*. WordPress.org Codex. Retrieved June 9, 2025, from https://wordpress.org/documentation/article/installing-wordpress/

Oracle Corporation. (n.d.). *4.5.4 mysqldump — A Database Backup Program*. MySQL 8.0 Reference Manual. Retrieved June 9, 2025, from https://dev.mysql.com/doc/refman/8.0/en/mysqldump.html

Free Software Foundation. (n.d.). *GNU tar: an archiver tool*. GNU Operating System. Retrieved June 9, 2025, from https://www.gnu.org/software/tar/

Microsoft. (n.d.). *Linux virtual machines in Azure*. Microsoft Learn. Retrieved June 9, 2025, from https://learn.microsoft.com/en-us/azure/virtual-machines/linux/

Canonical. (n.d.). *Ubuntu Server documentation*. Ubuntu. Retrieved June 9, 2025, from https://ubuntu.com/server/docs

The Apache Software Foundation. (n.d.). *Apache HTTP Server version 2.4 documentation*. Apache HTTP Server Project. Retrieved June 9, 2025, from https://httpd.apache.org/docs/2.4/

WordPress.org. (n.d.). *WordPress documentation*. Retrieved June 9, 2025, from https://wordpress.org/documentation/

Electronic Frontier Foundation. (n.d.). *Certbot documentation*. Certbot. Retrieved June 9, 2025, from https://certbot.eff.org/docs

Free Software Foundation. (n.d.). *GNU Bash reference manual*. GNU.org. Retrieved June 9, 2025, from https://www.gnu.org/software/bash/manual/