Creating a Virtual Machine on Azure and Hosting a Static Website : A Comprehensive Guide

Report By: Keshav Kumar
Date: 29th March 2025

ABSTRACT

This report presents a detailed, end-to-end methodology for establishing a robust static website hosting environment on Microsoft Azure, leveraging the power and flexibility of virtual machines (VMs). In an era dominated by cloud computing, understanding the practical application of Infrastructure as a Service (IaaS) solutions like Azure VMs is paramount for web developers and system administrators. This project aims to bridge the gap between theoretical knowledge and practical implementation, providing a clear, step-by-step guide to deploying a functional static website.

The report commences with the foundational stage of provisioning an Azure VM, meticulously detailing the selection of an appropriate operating system, configuring virtual hardware resources, and establishing secure network parameters. This involves navigating the Azure portal or utilizing command-line tools like Azure CLI or PowerShell, ensuring a scalable and resilient infrastructure foundation. Subsequently, the focus shifts to the critical task of installing and configuring a web server, specifically exploring the nuances of Nginx or Apache, both industry-standard solutions for serving web content. Detailed configuration examples are provided, demonstrating how to optimize server performance and security for static website delivery.

The deployment phase involves transferring the static website files to the VM's designated document root, utilizing secure file transfer protocols or cloud-based storage solutions. This process is accompanied by explanations of file permissions and directory structures, ensuring seamless integration with the chosen web server. Network security is then addressed through the configuration of network security groups (NSGs) and firewall rules, safeguarding the hosted website from unauthorized access and potential vulnerabilities. Furthermore, domain name system (DNS) configuration is explored, demonstrating how to map a custom domain to the VM's public IP address for user-friendly access.

Ultimately, this project serves as a practical demonstration of Azure's capabilities in the context of static website hosting, emphasizing the platform's scalability, flexibility, and control. It highlights how Azure VMs can be effectively employed to create a customizable and efficient hosting environment, adaptable to the evolving needs of web applications. This report's detailed methodology and analytical approach make it a valuable resource for anyone seeking to master the art of cloud-based web deployment.

Creating a VM on Azure and Hosting a Static Website :
A Comprehensive Guide

OBJECTIVE

This project aims to provide a comprehensive, hands-on understanding of deploying and managing a static website within the Microsoft Azure cloud environment. To achieve this, we will systematically address the following key objectives, each designed to build upon the previous and culminate in a fully functional, secure, and accessible web hosting solution.

1. Azure Virtual Machine Provisioning: Laying the Foundation for Cloud Infrastructure

- **Objective 1.1: Environment Setup and Resource Group Creation:**
    - Establish a functional Azure account, either through a free trial or a paid subscription, ensuring access to essential services.
    - Create a dedicated resource group to logically organize and manage all related resources, promoting efficiency and clarity.

- **Objective 1.2: Virtual Machine Deployment and Configuration:**
    - Select and deploy an appropriate virtual machine instance, considering factors like operating system compatibility (Linux distributions such as Ubuntu or CentOS), hardware specifications (CPU, memory, storage), and regional availability.
    - Configure the VM's virtual hardware resources to meet the performance requirements of the static website, balancing cost and efficiency.
    - Configure storage solutions for the VM, taking into account the type of disk required for optimal performance.

- **Objective 1.3: Network Infrastructure Setup:**
    - Establish a virtual network (VNet) and subnet to provide a secure and isolated network environment for the VM.
    - Assign a public IP address to the VM, enabling external access from the internet.
    - Configure network security groups (NSGs) to control both inbound and outbound traffic, laying a baseline for security.

2. Web Server Configuration and Static Website Deployment: Building the Web Serving Layer

- **Objective 2.1: Web Server Installation and Configuration:**

- Install and configure a chosen web server (Nginx or Apache) on the provisioned VM, demonstrating proficiency in server administration.
- Modify the web server's configuration files to define the document root, server name, and other essential parameters, ensuring correct website delivery.
- Optimize the chosen web server’s configuration for optimal performance when serving static content.

- **Objective 2.2: Static Website File Deployment:**
  - Transfer the static website files (HTML, CSS, JavaScript, images) to the designated document root directory on the VM, employing secure file transfer protocols (e.g., SCP, SFTP).
  - Verify file permissions and directory structures to ensure proper website rendering.

- **Objective 2.3: Network Security and Access Control:**
  - Configure firewall rules to allow HTTP (port 80) and HTTPS (port 443) traffic, enabling public access to the website.
  - Implement best practice security measures to protect the VM and web server from unauthorized access.

- **Objective 2.4: Domain Name System (DNS) Configuration:**
  - Configure DNS settings to map a custom domain name to the VM's public IP address, providing a user-friendly access point.
  - Understand the basic principles of DNS records, such as A records, and how they relate to web hosting.

- **Objective 2.5: Foundational Cloud Hosting Comprehension:**
  - Enhance understanding of the core concepts of cloud-based web hosting, including IaaS, scalability, and resource management.
  - Gain practical experience in managing a web server within the Azure cloud environment.
  - Develop a strong foundation for future cloud-based web application deployments.

Introduction

Introduction: Navigating the Landscape of Cloud-Based Static Website Hosting.

3.1. Cloud Computing and Azure: The Foundation of Modern IT Infrastructure :

The digital age has ushered in an era of unprecedented data generation and consumption, demanding flexible, scalable, and readily accessible IT infrastructure. Cloud computing has emerged as the cornerstone of this revolution, fundamentally transforming how organizations and individuals access and utilize computing resources. At its core, cloud computing provides on-demand access to a shared pool of configurable computing resources (e.g., servers, storage, applications, and services) over the internet.

This paradigm shift eliminates the need for substantial upfront investments in physical hardware and infrastructure, fostering agility, cost-efficiency, and rapid innovation.

Microsoft Azure stands as a leading cloud platform, offering a comprehensive suite of services that empower developers and businesses to build, deploy, and manage applications and services through Microsoft's globally distributed network of data centers.

Azure's extensive portfolio encompasses a wide array of offerings, including virtual machines (VMs), storage solutions, networking capabilities, databases, analytics, and artificial intelligence services. This platform's breadth and depth make it a powerful tool for a diverse range of applications, from hosting simple static websites to deploying complex enterprise-grade applications.

The adoption of Azure reflects the broader trend towards cloud-centric IT strategies, driven by the need for scalability, reliability, and global accessibility.

- **Hybrid Cloud Capabilities:** Azure supports hybrid cloud deployments, allowing organizations to integrate on-premises infrastructure with cloud resources.
- **Security and Compliance:** Azure provides robust security features and compliance certifications, ensuring data protection and meeting regulatory requirements.
- **Integration with Microsoft Ecosystem:** Azure seamlessly integrates with other Microsoft products and services, providing a unified platform for development and deployment.
- **Innovation and Development Tools:** Azure offers a wide range of developer tools and services, fostering innovation and accelerating application development.
- **Serverless Computing:** Azure provides serverless computing options like Azure Functions, allowing developers to run code without managing servers.
- **AI and Machine Learning:** Azure offers advanced AI and machine learning services, enabling developers to build intelligent applications.

3.2. Virtual Machines (VMs): The Building Blocks of Infrastructure as a Service

Within the Azure ecosystem, virtual machines (VMs) play a pivotal role in providing Infrastructure as a Service (IaaS). VMs are essentially software emulations of physical computers, encapsulating the operating system, applications, and data within a virtualized environment. This abstraction layer allows users to run multiple operating systems and applications concurrently on a single physical server, maximizing resource utilization and reducing hardware costs.

- **Hardware Abstraction:**
    - VMs abstract the underlying physical hardware, providing a consistent and predictable computing environment regardless of the physical server.
    - This abstraction facilitates portability and simplifies resource management.
- **Operating System Choice:**
    - Azure offers a wide range of operating system images, including various Linux distributions (Ubuntu, CentOS, Debian) and Windows Server versions.
    - Users can select the OS that best suits their application requirements and technical expertise.
- **Customization and Control:**
    - Users have root/administrator access to their VMs, enabling them to install and configure any software or service.
    - This level of control is essential for deploying custom applications and optimizing performance.
- **Scalability and Flexibility:**
    - Azure provides a variety of VM sizes and configurations, allowing users to scale their resources up or down as needed.
    - This scalability ensures that applications can handle fluctuating workloads without performance degradation.
- **Networking Capabilities:**
    - Azure VMs can be integrated into virtual networks (VNets), enabling secure and isolated communication between VMs and other Azure resources.
    - Users can configure network security groups (NSGs) to control inbound and outbound traffic, enhancing security.
- **Storage Options:**
    - Azure offers various storage options for VMs, including managed disks, which provide high-performance and reliable storage.
    - Users can choose the storage type and size that best suits their application's performance and capacity requirements.
- **High Availability and Redundancy:**

  - Azure provides features like availability sets and availability zones, which enable users to deploy VMs across multiple fault domains and data centers.
  - These features enhance application availability and resilience.
- **Cost Optimization:**
  - Azure's pay-as-you-go pricing model allows users to pay only for the resources they consume.
  - Users can optimize costs by selecting the appropriate VM size and configuration and by utilizing features like reserved instances.
- **Automation and Management:**
  - Azure provides tools like Azure CLI and PowerShell, which enable users to automate VM deployment and management.
  - This automation simplifies infrastructure management and reduces manual effort.
- **DevOps Integration:**
  - Azure VMs integrate well into DevOps workflows, allowing for easy deployment using tools like Azure DevOps, and other CI/CD pipelines.
- **Security features:**
  - Integration with Azure security center, and other security tools to provide a secure enviroment.

3.3. Benefits of Hosting on Azure VM:

Hosting a static website on an Azure VM offers several distinct advantages, making it a compelling choice for developers and businesses seeking a balance of control, performance, and cost-effectiveness.

- **Scalability:** Azure VMs can be easily scaled up or down to accommodate fluctuating traffic demands. This elasticity ensures that websites remain responsive and accessible, even during peak usage periods.
- **Flexibility:** Users retain complete control over the operating system and software installed on the VM, allowing them to customize their hosting environment to meet specific requirements. This level of flexibility is crucial for fine-tuning performance and security settings.
- **Reliability:** Azure's redundant infrastructure and high availability services ensure that websites remain accessible and operational, minimizing downtime and maximizing uptime.
- **Customization:** Web server configurations can be tailored to specific needs, enabling developers to optimize performance, implement security measures, and integrate with other services.
- **Cost-effectiveness:** Azure's pay-as-you-go pricing model allows users to pay only for the resources they consume, eliminating the need for substantial upfront investments. This cost-effective approach makes Azure VMs an attractive option for both small and large-scale deployments.

By leveraging the power of Azure VMs, developers can create robust, scalable, and highly customizable hosting environments for their static websites. This approach provides a strategic advantage, enabling them to deliver high-performance websites while maintaining control over their infrastructure.

METHODOLOGY

This section details the systematic methodology employed to establish a secure SSH (Secure Shell) connection to a provisioned Azure Virtual Machine (VM). This connection is fundamental for remote administration, configuration, and deployment of web hosting environments within the Azure cloud platform. The process involves leveraging cryptographic keys for authentication and ensuring secure communication between the local machine and the remote VM.

2. Pre-requisites and Initial Configuration:

- ### 2.1 Azure VM Provisioning:
    - Prior to initiating the SSH connection, an Azure VM must be successfully provisioned. This involves:
        - Creation of a dedicated Resource Group within the Azure portal or via Azure CLI to organize related resources.
        - Deployment of a Linux-based operating system (OS) on the VM (e.g., Ubuntu 22.04.5 LTS)
        - Configuration of a Virtual Network (VNet) and associated subnet to facilitate network connectivity.
        - Assignment of a public IP address to the VM, enabling external access from the internet.
        - Implementation of Network Security Groups (NSGs) with rules permitting SSH traffic (port 22) for secure remote access.

- ### 2.2 Private Key Acquisition:
    - During the VM creation process, a private key file AllProjectKeyPair.pem in the provided screenshot) is generated or provided. This key is paramount for secure authentication.
    - The private key file must be securely stored on the local machine and protected from unauthorized access.

- ### 2.3 Local Machine Environment:
    - The local machine must have an SSH client installed. Windows users can utilize the built-in command prompt or PowerShell, as demonstrated in the screenshot, or utilize a third party application like Putty.
    - Proper understanding of basic command-line operations is necessary to execute the ssh command.

3. SSH Connection Establishment :

- ## 3.1 SSH Command Execution:
  - The ssh command is executed from the local machine's command-line interface. The syntax is as follows:
    - ssh -i AllProjectKeyPair.pem projectuser@135.13.8.254
    - -i AllProjectKeyPair.pem: Specifies the path to the private key file used for authentication.
    - projectuser: Represents the username associated with the VM's login credentials.
    - 135.13.8.254: Represents the public IP address of the Azure VM.

- ## 3.2 Host Key Verification:
  - Upon the initial connection attempt, the SSH client will present a host key fingerprint.
  - This fingerprint serves as a security measure to verify the authenticity of the remote VM.
  - The user is prompted to confirm the connection by typing "yes," which adds the VM's host key to the known_hosts file, preventing future warnings.

- ## 3.3 Successful Connection and VM Access:
  - Upon successful authentication, the user gains command-line access to the Azure VM.
  - The screenshot confirms a successful connection, displaying the welcome message and system information for the Ubuntu 22.04.5 LTS VM.
  - This access allows for remote administration, software installation, and website deployment.

- ## 3.4 Security Considerations:
  - The use of Private keys removes the need for password authentication, increasing security.
  - It is of upmost importance to keep the private key secure.
  - NSG rules must be configured properly to only allow needed traffic.

- ## 4.1 User Environment and Location:
  - **4.1.1 User Identification:** The user currently logged into the system is "projectuser".
  - **4.1.2 Server Hostname:** The server's hostname is "ProjectMachine1".
  - **4.1.3 Home Directory:** The present working directory is the user's home directory.
  - **4.1.4 Path Confirmation:** The pwd command is used to explicitly print the current directory path, confirming it is "/home/projectuser".
- ## 4.2 System Update Phase
  - **4.2.1 Update Command:** The command sudo apt-get update -y is executed to refresh the package index.
  - **4.2.2 Administrative Privileges:** sudo is used to execute the command with superuser privileges, necessary for system-level changes.
  - **4.2.3 Package List Synchronization:** The apt-get update tool retrieves the latest package information from the configured software repositories.
  - **4.2.4 Automated Confirmation:** The -y flag automatically confirms any prompts during the update process, allowing for non-interactive execution.
  - **4.2.5 Repository Sources:** The output displays the sources from which package information is obtained, including "http://azure.archive.ubuntu.com/ubuntu jammy InRelease" for the main Ubuntu repository and similar sources for updates, backports, and security updates.
  - **4.2.6 Data Volume:** The amount of data fetched during the update is provided (e.g., "Fetched 2884 kB in 1s").
  - **4.2.7 Package List Status:** The final message "Reading package lists... Done" indicates successful completion of the update process.
- ## 4.3 System Upgrade Phase:
  - **4.3.1 Upgrade Command:** The command sudo apt-get upgrade -y is executed, aiming to upgrade installed packages.
  - **4.3.2 Dependency Resolution:** The system analyzes package dependencies to determine which packages can be safely upgraded.
  - **4.3.3 System State Evaluation:** The system reads the current state of installed packages to calculate the upgrade plan.
  - **4.3.4 Upgrade Calculation:** The system calculates the necessary changes to bring the installed packages to their latest versions.

- **4.3.5 Upgrade Outcome (First Execution):** The output "0 upgraded, 0 newly installed, 0 to remove and 0 not upgraded" indicates that no packages were upgraded during the first execution of the command.
- **4.3.6 Upgrade Outcome (Second Execution):** The sudo apt-get upgrade -y command is executed a second time, producing the same output as the first execution, confirming that no package changes were available or necessary.
- **4.3.7 System Consistency:** The repeated execution of the upgrade command reinforces the system's consistency and confirms that it is already in an up-to-date state.

5 Nginx Service Management:

5.1 Service Status Check:

- sudo systemctl status nginx is executed to check the status of the Nginx service.
- The output confirms that Nginx is "active (running)".
- It also displays information about the service's process ID, memory usage, and uptime.

5.2 Service Restart (Potentially Implicit):

- The output from a previous command (not directly visible in this page but implied) indicates that services like packagekit.service and ssh.service were restarted. This suggests system maintenance or configuration changes occurred prior to the Nginx status check.

6- PHP-FPM Installation:

6.1 Package Installation:

- sudo apt-get install php8.1-fpm -y is executed to install PHP 8.1-FPM and related packages.
- The -y flag automatically confirms all prompts.

6.2 Package Details:

- The output lists the additional packages that will be installed, including bzip2, mailcap, mime-support, php-common, php8.1-cli, php8.1-common, php8.1-opcache, and php8.1-readline.

6.3 Installation Summary:

- It's noted that 9 new packages will be installed.
- The process involves fetching 5253 kB of archives and will use 21.7 MB of additional disk space.

6.4 Package Retrieval:

- The output shows the retrieval of individual packages from the Ubuntu repositories.

7. PHP-FPM Configuration and Setup

- **7.1 Package Unpacking and Configuration:**
- The output details the unpacking and configuration of the installed packages, including php8.1-opcache, mime-support, php8.1-cli, and php8.1-fpm.
- This involves creating configuration files (e.g., /etc/php/8.1/mods-available/opcache.ini, /etc/php/8.1/cli/php.ini, /etc/php/8.1/fpm/php.ini) and setting up symbolic links.
- **7.2 Post-Installation Actions:**
- The system processes triggers for man-db, php8.1-cli, and php8.1-fpm.
- It then performs system checks, including scanning processes and Linux images, and confirms that the kernel is up-to-date.
- **7.3 System State Verification:**
- The output confirms that no services or containers need to be restarted, no user sessions are running outdated binaries, and no VM guests are running outdated hypervisor binaries.
- **7.4 Redundant PHP-FPM Installation:**
- sudo apt-get install php-fpm is executed again, seemingly redundantly.
- The output indicates that php-fpm is already installed.

8. Nginx Configuration :

- **8.1 Directory Navigation:**
- The user navigates to the /etc directory and then to the /etc/nginx/ directory using the cd command.
- ls command is used to list the contents of the /etc/nginx/ directory, showing configuration files and directories like sites-available and sites-enabled.
- **8.2 Sites-Enabled Directory:**
- The user navigates to the /etc/nginx/sites-enabled/ directory and lists its contents, revealing the default configuration file.
- **8.3 Nginx Configuration Editing:**
- sudo nano default and sudo nano /etc/nginx/sites-enabled/default are used to edit the default Nginx configuration file.
- **8.4 Configuration Testing:**
- sudo nginx -t is used to test the Nginx configuration for syntax errors.

- An error is encountered: nginx: [emerg] open() "/etc/nginx/sites-enabled/mysite" failed (2: No such file or directory) in /etc/nginx/nginx.conf:60. This indicates an issue with a mysite configuration that Nginx is trying to load.
- **8.5 Symbolic Links and File Management:**
- ls -l /etc/nginx/sites-enabled/ is used to list symbolic links in the sites-enabled directory. It shows default, mysite, and your-config-file are symbolic links.
- The user attempts to create a symbolic link (sudo ln -s /etc/nginx/sites-available/mysite /etc/nginx/sites-enabled/), but it fails because the file already exists.
- ls -l /etc/nginx/sites-available/ is used to list files in the sites-available directory.
- The user removes the problematic symbolic links (sudo rm /etc/nginx/sites-enabled/mysite, sudo rm /etc/nginx/sites-enabled/your-config-file).
- The user edits the mysite configuration file (sudo nano /etc/nginx/sites-available/mysite) and then recreates the symbolic link.
- **8.6 Configuration Re-testing:**
- sudo nginx -t is executed again, and this time the configuration test is successful.
- **8.7 Nginx Service Restart:**
- sudo systemctl restart nginx is used to restart the Nginx service to apply the configuration changes.

9. File Permissions and PHP Info File :

- **9.1 Directory Permissions:**
- sudo chmod -R 777 /var/www/html is used to set the permissions of the /var/www/html directory to 777 (read, write, and execute for all users).
- **Caution:** While this simplifies file access, it's generally **not recommended** for production environments due to security implications.
- **9.2 PHP Info File Creation:**
- echo "<?php phpinfo(); ?> >> /var/www/html/info.php" is used to create a info.php file in the /var/www/html directory. This file will display PHP configuration information when accessed through a web browser.

10. Adding PHP Repository and Installing PHP-FPM :

- **10.1 Adding PHP Repository:**
- The user attempts to add the Ondrej PHP repository using sudo add-apt-repository ppa:ondrej/php.

- A warning message is displayed about potential issues with non-UTF-8 locales, along with a workaround.
- The repository is added, updating the system's package sources.
- **10.2 Updating Package Lists:**
- sudo apt update is executed to refresh the package lists after adding the new repository.
- **10.3 Checking Upgradable Packages:**
- apt list --upgradable is mentioned in the output, indicating a check for packages that can be upgraded.
- **10.4 Installing PHP 8.1-FPM (Potentially Redundant):**
- sudo apt -y install php8.1-fpm is executed again, but the output indicates that PHP 8.1-FPM is already the newest version.
- **10.5 Starting and Enabling PHP 8.1-FPM:**
- sudo systemctl start php8.1-fpm starts the PHP 8.1-FPM service.
- sudo systemctl enable php8.1-fpm enables the PHP 8.1-FPM service to start automatically at boot.
- **10.6 Enabling Nginx:**
- sudo systemctl enable nginx enables the Nginx service to start automatically at boot.
- **11. Nginx Configuration Verification (Page 10)**
- **11.1 Navigating to Nginx Directory:**
- The user navigates back to the /etc/nginx/ directory using cd /etc/nginx/ and lists its contents using ls.
- **11.2 Navigating to Sites-Available:**
- The user attempts to navigate to /sites-available/ (which fails due to an extra /) and then correctly navigates to sites-available/.
- ls is used to list the contents, showing default and mysite.
- **11.3 Editing Default Configuration and Testing:**
- sudo nano default is used to edit the default Nginx configuration again.
- sudo nginx -t is used to test the configuration, which passes.
- sudo systemctl restart nginx restarts the Nginx service.

12. Domain Acquisition and DNS Configuration :

- ## 12.1 Domain Name Registration:
  - ### 12.1.1 Domain Provider Selection:
    - Choose a domain name provider (e.g., GoDaddy, Namecheap, Google Domains, AWS Route 53).
    - Consider factors like pricing, customer support, and additional services offered (e.g., email hosting, SSL certificates).
  - ### 12.1.2 Domain Name Search and Availability Check:
    - Use the registrar's search tool to check the availability of the desired domain name.
    - Select an appropriate domain extension (e.g., .com, .org, .net, .co.uk) based on the website's purpose and target audience.
  - ### 12.1.3 Domain Name Purchase:
    - Complete the domain registration process by providing the necessary contact information and payment details.
    - Configure domain privacy settings (if desired) to protect personal information in the WHOIS database.
- ## 12.2 DNS Record Configuration:
  - ### 12.2.1 Accessing DNS Management:
    - Log in to the account at the domain registrar where the domain was purchased.
    - Navigate to the DNS management settings for the registered domain.
  - ### 12.2.2 A Record Creation/Modification:
    - Create or modify an "A" record to point the domain name (or a subdomain, like "www") to the public IP address of the Azure VM.
    - The "A" record maps a hostname to an IPv4 address.
  - ### 12.2.3 TTL (Time to Live) Setting:
    - Configure the TTL for DNS records. TTL determines how long DNS servers cache the record before querying for updates.

- Lower TTL values allow for faster propagation of changes but can increase DNS traffic.

12.2.4 DNS Propagation:

- Understand that DNS changes take time to propagate across the internet. This propagation period can vary from a few minutes to several hours (or, in rare cases, up to 48 hours).

12.3 DNS Calls and Performance Considerations:

12.3.1 DNS Query Process:

- When a user enters a domain name in their browser, a DNS query is initiated to resolve the domain name to the corresponding IP address.
- This query typically travels through a series of DNS servers:
  - The user's local DNS resolver (often provided by their ISP).
  - Root DNS servers.
  - Top-Level Domain (TLD) servers (e.g., for .com, .org).
  - The authoritative DNS server for the domain (managed at the registrar or a third-party DNS provider).

12.3.2 DNS Caching:

- DNS servers cache DNS records to reduce the number of queries and improve response times.
- TTL values control how long these records are cached.

12.3.3 DNS Performance Optimization:

- **12.3.3.1 Choosing a Fast DNS Provider:** Using a reliable and fast DNS provider can significantly improve website loading times.
- **12.3.3.2 DNS Prefetching:** Implementing DNS prefetching on the website can hint to the browser to resolve domain names of linked resources in advance.
- **12.3.3.3 CDNs (Content Delivery Networks):** CDNs can further improve performance by caching website content closer to users and also often offer DNS services.

12.3.4 DNS Security:

- **12.3.4.1 DNSSEC (Domain Name System Security Extensions):** Implementing DNSSEC adds a layer of security to DNS by verifying the authenticity of DNS responses and preventing DNS spoofing.

Creating a VM on Azure and Hosting a Static Website :
A Comprehensive Guide

SCREENSHOTS

```
Microsoft Windows [Version 10.0.26100.3476]
(c) Microsoft Corporation. All rights reserved.

C:\Users\chaos>ssh
usage: ssh [-46AaCfGgKkMNnqsTtVvXxYy] [-B bind_interface] [-b bind_address]
           [-c cipher_spec] [-D [bind_address:]port] [-E log_file]
           [-e escape_char] [-F configfile] [-I pkcs11] [-i identity_file]
           [-J destination] [-L address] [-l login_name] [-m mac_spec]
           [-O ctl_cmd] [-o option] [-P tag] [-p port] [-Q query_option]
           [-R address] [-S ctl_path] [-W host:port] [-w local_tun[:remote_tun]]
           destination [command [argument ...]]

C:\Users\chaos>ssh -i AllProjectKeyPair.pem projectuser@135.13.8.254
The authenticity of host '135.13.8.254 (135.13.8.254)' can't be established.
ED25519 key fingerprint is SHA256:QgzMvnDlNihJZJyKM6mpCCK3rjOncCeWDbZ2E7prTaU.
This key is not known by any other names.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '135.13.8.254' (ED25519) to the list of known hosts.
Welcome to Ubuntu 22.04.5 LTS (GNU/Linux 6.8.0-1021-azure x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/pro

 System information as of Wed Mar 26 14:29:42 UTC 2025

  System load:  0.66              Processes:             107
  Usage of /:   5.2% of 28.89GB   Users logged in:       0
  Memory usage: 28%               IPv4 address for eth0: 10.0.0.4
  Swap usage:   0%

Expanded Security Maintenance for Applications is not enabled.
```

```
-rw-r--r-- 1 projectuser projectuser 3771 Jan  6  2022 .bashrc
drwx------ 2 projectuser projectuser 4096 Mar 23 03:13 .cache
-rw-r--r-- 1 projectuser projectuser  807 Jan  6  2022 .profile
drwx------ 2 projectuser projectuser 4096 Mar 22 12:11 .ssh
projectuser@ProjectMachine1:~$ pwd
/home/projectuser
projectuser@ProjectMachine1:~$ sudo apt-get update -y
Hit:1 http://azure.archive.ubuntu.com/ubuntu jammy InRelease
Get:2 http://azure.archive.ubuntu.com/ubuntu jammy-updates InRelease [128 kB]
Get:3 http://azure.archive.ubuntu.com/ubuntu jammy-backports InRelease [127 kB]
Get:4 http://azure.archive.ubuntu.com/ubuntu jammy-security InRelease [129 kB]
Get:5 http://azure.archive.ubuntu.com/ubuntu jammy-updates/main amd64 Packages [2390 kB]
Get:6 http://azure.archive.ubuntu.com/ubuntu jammy-backports/main amd64 Packages [68.4 kB]
Get:7 http://azure.archive.ubuntu.com/ubuntu jammy-backports/main Translation-en [11.1 kB]
Get:8 http://azure.archive.ubuntu.com/ubuntu jammy-backports/universe amd64 Packages [30.1 kB]
Fetched 2884 kB in 1s (2148 kB/s)
Reading package lists... Done
projectuser@ProjectMachine1:~$ sudo apt-get upgrade -y
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
Calculating upgrade... Done
0 upgraded, 0 newly installed, 0 to remove and 0 not upgraded.
projectuser@ProjectMachine1:~$ sudo apt-get upgrade -y
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
Calculating upgrade... Done
0 upgraded, 0 newly installed, 0 to remove and 0 not upgraded.
projectuser@ProjectMachine1:~$
```

```
Scanning linux images...

Running kernel seems to be up-to-date.

Restarting services...
 systemctl restart packagekit.service ssh.service

No containers need to be restarted.

No user sessions are running outdated binaries.

No VM guests are running outdated hypervisor (qemu) binaries on this host.
projectuser@ProjectMachine1:~$ sudo systemctl status nginx
● nginx.service - A high performance web server and a reverse proxy server
     Loaded: loaded (/lib/systemd/system/nginx.service; enabled; vendor preset: enabled)
     Active: active (running) since Sun 2025-03-23 03:21:10 UTC; 54s ago
       Docs: man:nginx(8)
    Process: 13387 ExecStartPre=/usr/sbin/nginx -t -q -g daemon on; master_process on; (code=exited, status=0/SUCCESS)
    Process: 13388 ExecStart=/usr/sbin/nginx -g daemon on; master_process on; (code=exited, status=0/SUCCESS)
   Main PID: 13481 (nginx)
      Tasks: 2 (limit: 1064)
     Memory: 4.1M
        CPU: 30ms
     CGroup: /system.slice/nginx.service
             ├─13481 "nginx: master process /usr/sbin/nginx -g daemon on; master_process on;"
             └─13484 "nginx: worker process" "" "" "" "" "" "" "" "" "" "" "" "" "" "" "" "" "" "" "" "" "" "" "" "" "">

Mar 23 03:21:10 ProjectMachine1 systemd[1]: Starting A high performance web server and a reverse proxy server...
```

```
projectuser@ProjectMachine1:~$ sudo apt-get install php8.1-fpm -y
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following additional packages will be installed:
  bzip2 mailcap mime-support php-common php8.1-cli php8.1-common php8.1-opcache php8.1-readline
Suggested packages:
  bzip2-doc php-pear
The following NEW packages will be installed:
  bzip2 mailcap mime-support php-common php8.1-cli php8.1-common php8.1-fpm php8.1-opcache php8.1-readline
0 upgraded, 9 newly installed, 0 to remove and 0 not upgraded.
Need to get 5253 kB of archives.
After this operation, 21.7 MB of additional disk space will be used.
Get:1 http://azure.archive.ubuntu.com/ubuntu jammy/main amd64 bzip2 amd64 1.0.8-5build1 [34.8 kB]
Get:2 http://azure.archive.ubuntu.com/ubuntu jammy/main amd64 mailcap all 3.70+nmu1ubuntu1 [23.8 kB]
Get:3 http://azure.archive.ubuntu.com/ubuntu jammy/main amd64 mime-support all 3.66 [3696 B]
Get:4 http://azure.archive.ubuntu.com/ubuntu jammy/main amd64 php-common all 2:92ubuntu1 [12.4 kB]
Get:5 http://azure.archive.ubuntu.com/ubuntu jammy-updates/main amd64 php8.1-common amd64 8.1.2-1ubuntu2.20 [1127 kB]
Get:6 http://azure.archive.ubuntu.com/ubuntu jammy-updates/main amd64 php8.1-opcache amd64 8.1.2-1ubuntu2.20 [365 kB]
Get:7 http://azure.archive.ubuntu.com/ubuntu jammy-updates/main amd64 php8.1-readline amd64 8.1.2-1ubuntu2.20 [13.5 kB]
Get:8 http://azure.archive.ubuntu.com/ubuntu jammy-updates/main amd64 php8.1-cli amd64 8.1.2-1ubuntu2.20 [1833 kB]
Get:9 http://azure.archive.ubuntu.com/ubuntu jammy-updates/universe amd64 php8.1-fpm amd64 8.1.2-1ubuntu2.20 [1840 kB]
Fetched 5253 kB in 3s (1656 kB/s)
Selecting previously unselected package bzip2.
(Reading database ... 62924 files and directories currently installed.)
Preparing to unpack .../0-bzip2_1.0.8-5build1_amd64.deb ...
Unpacking bzip2 (1.0.8-5build1) ...
Selecting previously unselected package mailcap.
Preparing to unpack .../1-mailcap_3.70+nmu1ubuntu1_all.deb ...
Unpacking mailcap (3.70+nmu1ubuntu1) ...
```

```
Setting up php8.1-opcache (8.1.2-1ubuntu2.20) ...

Creating config file /etc/php/8.1/mods-available/opcache.ini with new version
Setting up mime-support (3.66) ...
Setting up php8.1-cli (8.1.2-1ubuntu2.20) ...
update-alternatives: using /usr/bin/php8.1 to provide /usr/bin/php (php) in auto mode
update-alternatives: using /usr/bin/phar8.1 to provide /usr/bin/phar (phar) in auto mode
update-alternatives: using /usr/bin/phar.phar8.1 to provide /usr/bin/phar.phar (phar.phar) in auto mode

Creating config file /etc/php/8.1/cli/php.ini with new version
Setting up php8.1-fpm (8.1.2-1ubuntu2.20) ...

Creating config file /etc/php/8.1/fpm/php.ini with new version
Created symlink /etc/systemd/system/multi-user.target.wants/php8.1-fpm.service → /lib/systemd/system/php8.1-fpm.service.
Processing triggers for man-db (2.10.2-1) ...
Processing triggers for php8.1-cli (8.1.2-1ubuntu2.20) ...
Processing triggers for php8.1-fpm (8.1.2-1ubuntu2.20) ...
Scanning processes...
Scanning linux images...

Running kernel seems to be up-to-date.

No services need to be restarted.

No containers need to be restarted.

No user sessions are running outdated binaries.

No VM guests are running outdated hypervisor (qemu) binaries on this host.
projectuser@ProjectMachine1:~$
```

```
projectuser@ProjectMachine1:~$ sudo apt-get install php-fpm
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following NEW packages will be installed:
  php-fpm
0 upgraded, 1 newly installed, 0 to remove and 0 not upgraded.
Need to get 2838 B of archives.
After this operation, 14.3 kB of additional disk space will be used.
Get:1 http://azure.archive.ubuntu.com/ubuntu jammy/universe amd64 php-fpm all 2:8.1+92ubuntu1 [2838 B]
Fetched 2838 B in 0s (12.3 kB/s)
Selecting previously unselected package php-fpm.
(Reading database ... 63112 files and directories currently installed.)
Preparing to unpack .../php-fpm_2%3a8.1+92ubuntu1_all.deb ...
Unpacking php-fpm (2:8.1+92ubuntu1) ...
Setting up php-fpm (2:8.1+92ubuntu1) ...
Scanning processes...
Scanning linux images...

Running kernel seems to be up-to-date.

No services need to be restarted.

No containers need to be restarted.

No user sessions are running outdated binaries.

No VM guests are running outdated hypervisor (qemu) binaries on this host.
projectuser@ProjectMachine1:~$
```

```
ca-certificates.conf.dpkg-old  hosts            mime.types          rc0.d                 systemd
chrony                         hosts.allow       mke2fs.conf         rc1.d                 terminfo
cifs-utils                     hosts.deny        modprobe.d          rc2.d                 timezone
cloud                          init              modules             rc3.d                 tmpfiles.d
console-setup                  init.d            modules-load.d      rc4.d                 ubuntu-advantage
cron.d                         initramfs-tools   mtab                rc5.d                 ucf.conf
cron.daily                     inputrc           multipath           rc6.d                 udev
cron.hourly                    iproute2          multipath.conf      rcS.d                 ufw
cron.monthly                   iscsi             nanorc              request-key.conf      update-manager
cron.weekly                    issue             needrestart         request-key.d         update-motd.d
crontab                        issue.net         netconfig           resolv.conf           update-notifier
cryptsetup-initramfs           kernel            netplan             rmt                   usb_modeswitch.conf
crypttab                       kernel-img.conf   network             rpc                   usb_modeswitch.d
dbus-1                         keyutils          networkd-dispatcher rsyslog.conf          vim
debconf.conf                   landscape         networks            rsyslog.d             vmware-tools
debian_version                 ld.so.cache       newt                screenrc              vtrgb
default                        ld.so.conf        nftables.conf       security              waagent.conf
deluser.conf                   ld.so.conf.d      nginx               selinux               wgetrc
depmod.d                       ldap              nsswitch.conf       sensors.d             xattr.conf
dhcp                           legal             nvme                sensors3.conf         xdg
dpkg                           libaudit.conf     opt                 services              zsh_command_not_found
e2scrub.conf                   libblockdev       os-release          shadow
projectuser@ProjectMachine1:/etc$ cd nginx/
projectuser@ProjectMachine1:/etc/nginx$ ls
conf.d        fastcgi_params  koi-win     modules-available  nginx.conf    scgi_params      sites-enabled  uwsgi_params
fastcgi.conf  koi-utf         mime.types  modules-enabled    proxy_params  sites-available  snippets       win-utf
projectuser@ProjectMachine1:/etc/nginx$ cd sites-enabled/
projectuser@ProjectMachine1:/etc/nginx/sites-enabled$ ls
default
projectuser@ProjectMachine1:/etc/nginx/sites-enabled$ sudo nano default
```

```
projectuser@ProjectMachine1:/etc/nginx/sites-enabled$ sudo nano /etc/nginx/sites-enabled/default
projectuser@ProjectMachine1:/etc/nginx/sites-enabled$ sudo nginx -t
nginx: [emerg] open() "/etc/nginx/sites-enabled/mysite" failed (2: No such file or directory) in /etc/nginx/nginx.conf:6
0
nginx: configuration file /etc/nginx/nginx.conf test failed
projectuser@ProjectMachine1:/etc/nginx/sites-enabled$ ls -l /etc/nginx/sites-enabled/
total 0
lrwxrwxrwx 1 root root 34 Mar 23 03:21 default -> /etc/nginx/sites-available/default
lrwxrwxrwx 1 root root 33 Mar 23 05:05 mysite -> /etc/nginx/sites-available/mysite
lrwxrwxrwx 1 root root 43 Mar 23 04:57 your-config-file -> /etc/nginx/sites-available/your-config-file
projectuser@ProjectMachine1:/etc/nginx/sites-enabled$ sudo ln -s /etc/nginx/sites-available/mysite /etc/nginx/sites-enab
led/
ln: failed to create symbolic link '/etc/nginx/sites-enabled/mysite': File exists
projectuser@ProjectMachine1:/etc/nginx/sites-enabled$ ls -l /etc/nginx/sites-available/
total 4
-rw-r--r-- 1 root root 410 Mar 23 05:20 default
projectuser@ProjectMachine1:/etc/nginx/sites-enabled$ sudo rm /etc/nginx/sites-enabled/mysite
sudo rm /etc/nginx/sites-enabled/your-config-file
projectuser@ProjectMachine1:/etc/nginx/sites-enabled$ sudo nano /etc/nginx/sites-available/mysite
projectuser@ProjectMachine1:/etc/nginx/sites-enabled$ sudo ln -s /etc/nginx/sites-available/mysite /etc/nginx/sites-enab
led/
projectuser@ProjectMachine1:/etc/nginx/sites-enabled$ sudo nginx -t
nginx: the configuration file /etc/nginx/nginx.conf syntax is ok
nginx: configuration file /etc/nginx/nginx.conf test is successful
projectuser@ProjectMachine1:/etc/nginx/sites-enabled$ sudo nano default
projectuser@ProjectMachine1:/etc/nginx/sites-enabled$ sudo systemctl restart nginx
projectuser@ProjectMachine1:/etc/nginx/sites-enabled$ cd ~
projectuser@ProjectMachine1:~$ sudo chmod -R 777 /var/www/html
projectuser@ProjectMachine1:~$ echo "<?php phpinfo(); ?>" >> /var/www/html/info.php
projectuser@ProjectMachine1:~$
```

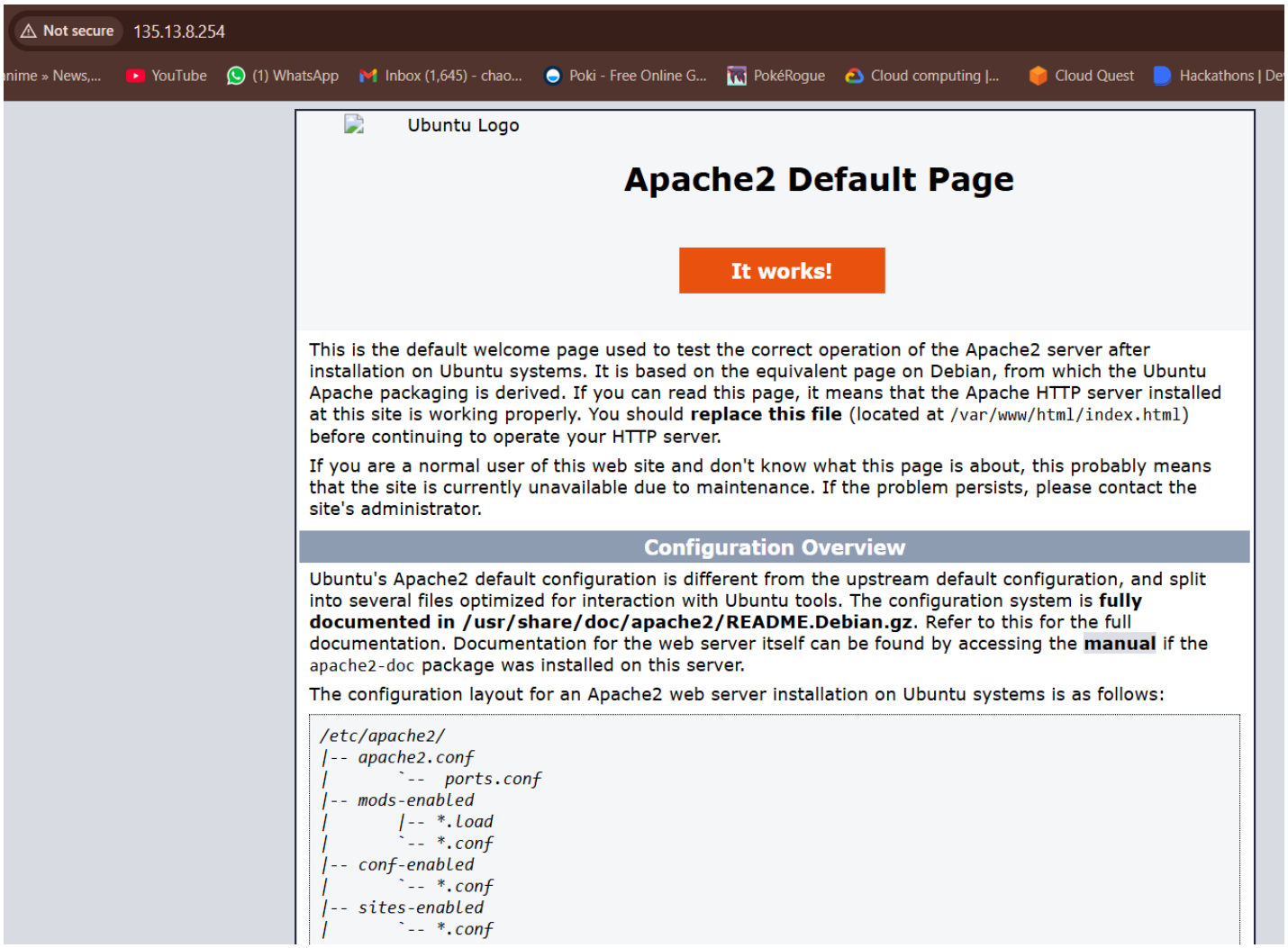
```
WARNING: add-apt-repository is broken with non-UTF-8 locales, see
https://github.com/oerdnj/deb.sury.org/issues/56 for workaround:

# LC_ALL=C.UTF-8 add-apt-repository ppa:ondrej/php
More info: https://launchpad.net/~ondrej/+archive/ubuntu/php
Adding repository.
Press [ENTER] to continue or Ctrl-c to cancel.
Adding deb entry to /etc/apt/sources.list.d/ondrej-ubuntu-php-jammy.list
Adding disabled deb-src entry to /etc/apt/sources.list.d/ondrej-ubuntu-php-jammy.list
Adding key to /etc/apt/trusted.gpg.d/ondrej-ubuntu-php.gpg with fingerprint B8DC7E53946656EFBCE4C1DD71DAEAAB4AD4CAB6
Hit:1 http://azure.archive.ubuntu.com/ubuntu jammy InRelease
Hit:2 http://azure.archive.ubuntu.com/ubuntu jammy-updates InRelease
Hit:3 http://azure.archive.ubuntu.com/ubuntu jammy-backports InRelease
Hit:4 http://azure.archive.ubuntu.com/ubuntu jammy-security InRelease
Get:5 https://ppa.launchpadcontent.net/ondrej/php/ubuntu jammy InRelease [24.6 kB]
Get:6 https://ppa.launchpadcontent.net/ondrej/php/ubuntu jammy/main amd64 Packages [137 kB]
Get:7 https://ppa.launchpadcontent.net/ondrej/php/ubuntu jammy/main Translation-en [42.8 kB]
Fetched 204 kB in 3s (76.6 kB/s)
Reading package lists... Done
projectuser@ProjectMachine1:/var/www/html$ sudo apt update
Hit:1 http://azure.archive.ubuntu.com/ubuntu jammy InRelease
Hit:2 http://azure.archive.ubuntu.com/ubuntu jammy-updates InRelease
Hit:3 http://azure.archive.ubuntu.com/ubuntu jammy-backports InRelease
Hit:4 http://azure.archive.ubuntu.com/ubuntu jammy-security InRelease
Hit:5 https://ppa.launchpadcontent.net/ondrej/php/ubuntu jammy InRelease
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
12 packages can be upgraded. Run 'apt list --upgradable' to see them.
projectuser@ProjectMachine1:/var/www/html$
```

```
Zend Engine v4.3.19, Copyright (c) Zend Technologies
    with Zend OPcache v8.3.19, Copyright (c), by Zend Technologies
projectuser@ProjectMachine1:/var/www/html$ sudo apt -y install php8.1-fpm
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
php8.1-fpm is already the newest version (8.1.32-1+ubuntu22.04.1+deb.sury.org+1).
0 upgraded, 0 newly installed, 0 to remove and 0 not upgraded.
projectuser@ProjectMachine1:/var/www/html$ sudo systemctl start php8.1-fpm
projectuser@ProjectMachine1:/var/www/html$ sudo systemctl enable php8.1-fpm
Synchronizing state of php8.1-fpm.service with SysV service script with /lib/systemd/systemd-sysv-install.
Executing: /lib/systemd/systemd-sysv-install enable php8.1-fpm
projectuser@ProjectMachine1:/var/www/html$ sudo systemctl enable nginx
Synchronizing state of nginx.service with SysV service script with /lib/systemd/systemd-sysv-install.
Executing: /lib/systemd/systemd-sysv-install enable nginx
projectuser@ProjectMachine1:/var/www/html$ cd /etc/nginx/
projectuser@ProjectMachine1:/etc/nginx$ ls
conf.d        fastcgi_params  koi-win     modules-available  nginx.conf    scgi_params      sites-enabled  uwsgi_params
fastcgi.conf  koi-utf         mime.types  modules-enabled    proxy_params  sites-available  snippets       win-utf
projectuser@ProjectMachine1:/etc/nginx$ cd/sites-available/
-bash: cd/sites-available/: No such file or directory
projectuser@ProjectMachine1:/etc/nginx$ cd sites-available/
projectuser@ProjectMachine1:/etc/nginx/sites-available$ ls
default  mysite
projectuser@ProjectMachine1:/etc/nginx/sites-available$ sudo nano default
projectuser@ProjectMachine1:/etc/nginx/sites-available$ sudo nginx -t
nginx: the configuration file /etc/nginx/nginx.conf syntax is ok
nginx: configuration file /etc/nginx/nginx.conf test is successful
projectuser@ProjectMachine1:/etc/nginx/sites-available$ sudo systemctl restart nginx
projectuser@ProjectMachine1:/etc/nginx/sites-available$
```

```
default  mysite  project1.fstackdev.xyz
projectuser@ProjectMachine1:/etc/nginx/sites-available$ sudo nginx -t
nginx: the configuration file /etc/nginx/nginx.conf syntax is ok
nginx: configuration file /etc/nginx/nginx.conf test is successful
projectuser@ProjectMachine1:/etc/nginx/sites-available$ sudo chown -R $USER:$USER /var/www/project1.fstack.xyz/
projectuser@ProjectMachine1:/etc/nginx/sites-available$ sudo chown -R $USER:$USER /var/www/project1.fstackdev.xyz/
chown: cannot access '/var/www/project1.fstackdev.xyz/': No such file or directory
projectuser@ProjectMachine1:/etc/nginx/sites-available$ sudo chown -R $USER:$USER /var/www/project1.fstackdev.xyz/
chown: cannot access '/var/www/project1.fstackdev.xyz/': No such file or directory
projectuser@ProjectMachine1:/etc/nginx/sites-available$ ls -ld /var/www/project1.fstackdev.xyz
ls: cannot access '/var/www/project1.fstackdev.xyz': No such file or directory
projectuser@ProjectMachine1:/etc/nginx/sites-available$ sudo mkdir -p /var/www/project1.fstackdev.xyz
projectuser@ProjectMachine1:/etc/nginx/sites-available$ sudo chown -R $USER:$USER /var/www/project1.fstackdev.xyz/
projectuser@ProjectMachine1:/etc/nginx/sites-available$ sudo chown -R www-data:www-data /var/www/project1.fstackdev.xyz/
projectuser@ProjectMachine1:/etc/nginx/sites-available$ sudo chmod -R 755 /var/www/project1.fstackdev.xyz/
projectuser@ProjectMachine1:/etc/nginx/sites-available$ ls -ld /var/www/project1.fstackdev.xyz
drwxr-xr-x 2 www-data www-data 4096 Mar 23 10:13 /var/www/project1.fstackdev.xyz
projectuser@ProjectMachine1:/etc/nginx/sites-available$ sudo chmod -R g+w /var/www/project1.fstackdev.xyz/
projectuser@ProjectMachine1:/etc/nginx/sites-available$ sudo ln -s /etc/nginx/sites-available/project1.fstackdev.xyz /et
c/nginx/sites-enabled/
projectuser@ProjectMachine1:/etc/nginx/sites-available$ sudo nginx -t
nginx: the configuration file /etc/nginx/nginx.conf syntax is ok
nginx: configuration file /etc/nginx/nginx.conf test is successful
projectuser@ProjectMachine1:/etc/nginx/sites-available$ sudo systemctl reload nginx
projectuser@ProjectMachine1:/etc/nginx/sites-available$ sudo systemctl reload nginx^C
Invalid unit name "nginx^C" escaped as "nginx\x5eC" (maybe you should use systemd-escape?).
Failed to reload nginx\x5eC.service: Unit nginx\x5eC.service not found.
projectuser@ProjectMachine1:/etc/nginx/sites-available$ sudo nano /etc/nginx/sites-available/project1.fstackdev.xyz
projectuser@ProjectMachine1:/etc/nginx/sites-available$ sudo nano /etc/nginx/sites-enabled/project1.fstackdev.xyz
projectuser@ProjectMachine1:/etc/nginx/sites-available$
```

OUTPUTS :


Not secure
135.13.8.254
YouTube
(1) WhatsApp
Inbox (1,645) - chao...
Poki - Free Online G...
PokéRogue
Cloud computing |...
Cloud Quest
Ubuntu Logo
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

PHP Version 8.1.32	
System	Linux ProjectMachine1 6.8.0-1021-azure #25~22.04.1-Ubuntu SMP Thu Jan 16 21:37:09 UTC 2025 x86_64
Build Date	Mar 13 2025 18:27:13
Build System	Linux
Server API	FPM/FastCGI
Virtual Directory Support	disabled
Configuration File (php.ini) Path	/etc/php/8.1/fpm
Loaded Configuration File	/etc/php/8.1/fpm/php.ini
Scan this dir for additional .ini files	/etc/php/8.1/fpm/conf.d
Additional .ini files parsed	/etc/php/8.1/fpm/conf.d/10-opcache.ini, /etc/php/8.1/fpm/conf.d/10-pdo.ini, /etc/php/8.1/fpm/conf.d/20-calendar.ini, /etc/php/8.1/fpm/conf.d/20-ctype.ini, /etc/php/8.1/fpm/conf.d/20-exif.ini, /etc/php/8.1/fpm/conf.d/20-ffi.ini, /etc/php/8.1/fpm/conf.d/20-fileinfo.ini, /etc/php/8.1/fpm/conf.d/20-ftp.ini, /etc/php/8.1/fpm/conf.d/20-gettext.ini, /etc/php/8.1/fpm/conf.d/20-iconv.ini, /etc/php/8.1/fpm/conf.d/20-phar.ini, /etc/php/8.1/fpm/conf.d/20-posix.ini, /etc/php/8.1/fpm/conf.d/20-readline.ini, /etc/php/8.1/fpm/conf.d/20-shmop.ini, /etc/php/8.1/fpm/conf.d/20-sockets.ini, /etc/php/8.1/fpm/conf.d/20-sysvmsg.ini, /etc/php/8.1/fpm/conf.d/20-sysvsem.ini, /etc/php/8.1/fpm/conf.d/20-sysvshm.ini, /etc/php/8.1/fpm/conf.d/20-tokenizer.ini
PHP API	20210902
PHP Extension	20210902
Zend Extension	420210902
Zend Extension Build	API420210902,NTS
PHP Extension Build	API20210902,NTS
Debug Build	no
Thread Safety	disabled
Zend Signal Handling	enabled
Zend Memory Manager	enabled
Zend Multibyte Support	disabled
Zend Max Execution Timers	disabled
IPv6 Support	enabled
DTrace Support	available, disabled
Registered PHP Streams	https, ftps, compress.zlib, php, file, glob, data, http, ftp, phar
Registered Stream Socket Transports	tcp, udp, unix, udg, ssl, tls, tlsv1.0, tlsv1.1, tlsv1.2, tlsv1.3
Registered Stream Filters	zlib.*, string.rot13, string.toupper, string.tolower, convert.*, consumed, dechunk, convert.iconv.*

This program makes use of the Zend Scripting Language Engine:
Zend Engine v4.1.32, Copyright (c) Zend Technologies
with Zend OPcache v8.1.32, Copyright (c), by Zend Technologies

Configuration

calendar

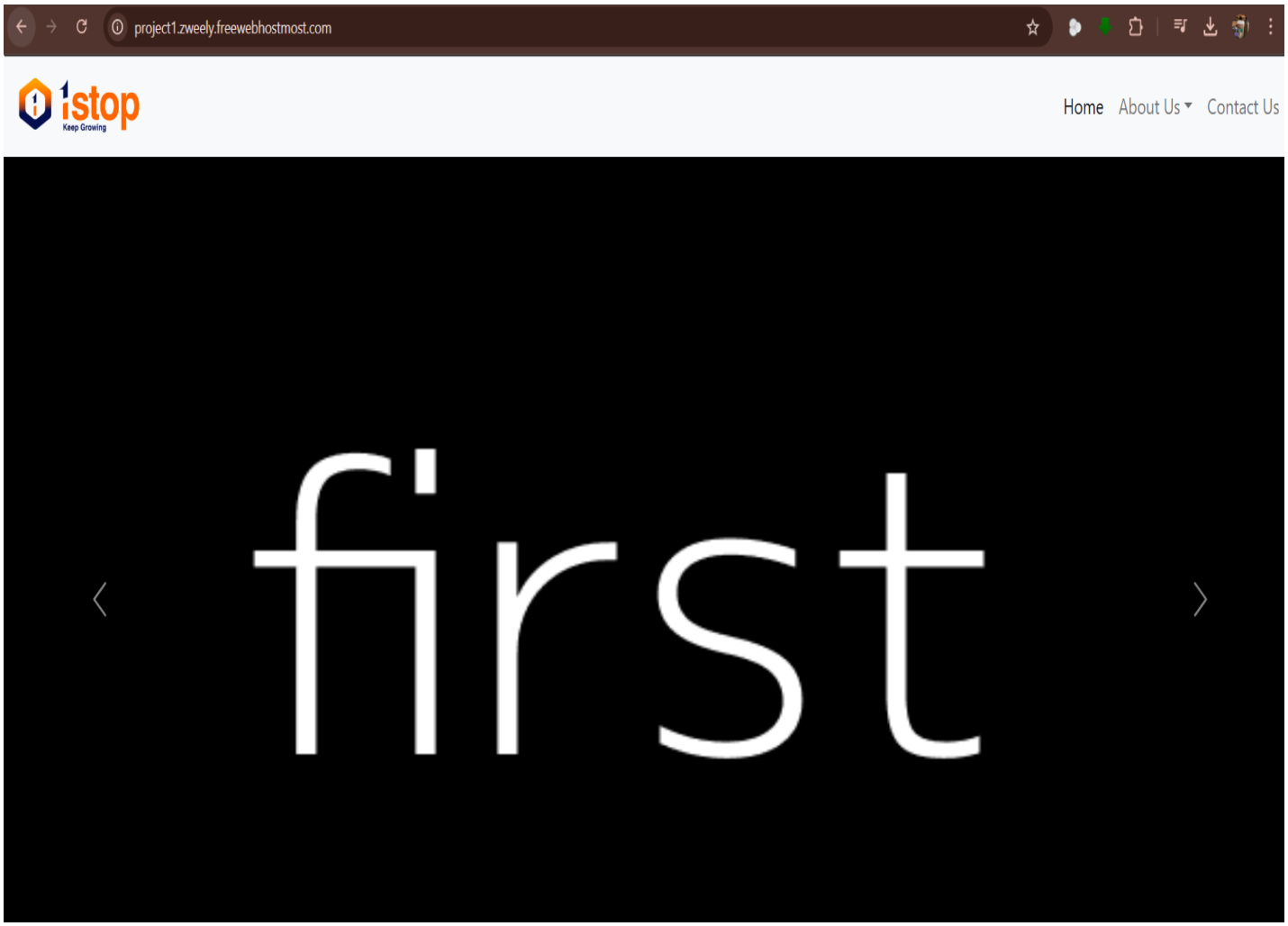
Calendar support	enabled

cgi-fcgi

php-fpm	active

Directive	Local Value	Master Value
cgi.discard_path	Off	Off
cgi.fix_pathinfo	On	On
cgi.force_redirect	On	On
cgi.nph	Off	Off
cgi.redirect_status_env	*no value*	*no value*
cgi.rfc2616_headers	Off	Off
fastcgi.error_header	*no value*	*no value*
fastcgi.logging	On	On
fpm.config	*no value*	*no value*

Core


project1.zweely.freewebhostmost.com
1stop
Keep Growing
Home
About Us
Contact Us
first

CONCLUSION :

- This report has provided a comprehensive guide to deploying and managing a static website on Microsoft Azure.
- The process involves provisioning an Azure Virtual Machine (VM), configuring a web server (Nginx), and setting up DNS records for domain accessibility.
- Azure VMs offer a scalable and flexible environment for hosting websites, providing users with control over the server and its configurations.
- Proper security measures, including Network Security Groups and secure key management, are crucial for protecting the VM and the hosted website.
- The methodology outlined in this report enables users to efficiently deploy and manage static websites on the Azure cloud platform.