Cloud Server Deployment using Hostinger VPS

Shashank Shelar[1] Mrs. Vrushali M. Shinde[2]
[1]Student [2]Guide
[1,2]Master of Computer Applications
[1,2]P. E. S. Modern College of Engineering, Pune, India

Abstract — This paper presents a detailed implementation of cloud infrastructure deployment using Virtual Private Servers (VPS) and Amazon Web Services (AWS). Traditional hosting systems face challenges such as high cost, limited scalability, and manual configuration complexity. This study demonstrates a practical approach using Linux based environments, SSH access, and NGINX web server configuration. The results highlight improvements in deployment speed, scalability, and operational efficiency. The project bridges the gap between theoretical knowledge and real-world cloud engineering practices.

Keywords: Cloud Computing, VPS, AWS, NGINX, SSH, Linux, Deployment, Web Hosting

I. Introduction

Cloud computing has become a fundamental component of modern IT infrastructure. It enables organizations to deploy applications without investing heavily in physical hardware. Traditional hosting systems rely on on-premise servers, which are costly and difficult to scale. In contrast, cloud platforms provide OnDemand resources, flexibility, and remote accessibility. This paper focuses on implementing a real-world cloud deployment pipeline using VPS and AWS technologies.

II. Problem Statement

Traditional application hosting methods rely heavily on physical infrastructure and on-premise server environments, which present several limitations in modern computing scenarios. These systems require significant initial investment for hardware setup and involve ongoing maintenance costs, making them less cost effective for scalable applications.

Additionally, traditional hosting environments lack dynamic scalability, as resources such as CPU, memory, and storage cannot be adjusted easily based on demand. This results in either underutilization or resource shortages during peak loads. The configuration and management of such systems are largely manual, increasing the risk of human error and reducing deployment efficiency.

Another key limitation is the lack of remote accessibility and flexibility, which makes it difficult to manage and deploy applications efficiently in distributed environments. Furthermore, implementing security mechanisms such as encrypted communication and access control adds additional complexity.

These challenges highlight the need for a cloud-based solution that provides OnDemand resource allocation, simplified deployment processes, remote accessibility, and built-in security features. This project addresses these issues by implementing a cloud infrastructure using VPS and AWS, enabling efficient, scalable, and secure application deployment.

III. Objectives

The main objectives of this project are as follows:

1) To deploy web applications using cloud infrastructure through VPS and AWS platforms.
2) To configure and manage a Linux based server environment for hosting applications.
3) To establish secure remote access using the Secure Shell (SSH) protocol.
4) To install and configure the NGINX web server for handling HTTP/HTTPS requests.
5) To implement domain mapping using DNS configuration for public accessibility.
6) To enable secure communication using SSL/TLS certificates (HTTPS).
7) To analyze the advantages of cloud-based deployment over traditional hosting systems.
8) To gain practical, hands-on experience in real world cloud infrastructure deployment.

IV. Literature Review

Cloud computing has been widely studied as a modern solution for scalable and cost-effective application deployment. According to Amazon Web Services (AWS) documentation, cloud platforms provide OnDemand resources, enabling flexible infrastructure management and reducing the need for physical hardware [1]. Similarly, studies such as Armbrust et al. highlight that cloud computing significantly improves scalability, availability, and cost efficiency compared to traditional hosting systems [5].

Virtual Private Servers (VPS) are also commonly used for application hosting due to their affordability and dedicated resource allocation. Hosting providers such as Hostinger offer VPS solutions that allow users to deploy and manage applications with greater control over the server environment [2].

Web server technologies like NGINX play a critical role in handling HTTP requests efficiently. NGINX is widely recognized for its high performance and ability to manage multiple concurrent connections, making it suitable for production level deployment [3].

Linux based server environments are preferred in cloud deployments due to their stability, security, and flexibility. Ubuntu server documentation emphasizes the importance of package management and system configuration for maintaining a reliable server environment [4].

Despite the extensive theoretical knowledge available, many academic programs provide limited hands-on exposure to real-world deployment processes. Most studies focus on conceptual understanding rather than practical implementation involving server provisioning, SSH access, and live deployment.

This project addresses this gap by implementing a complete cloud deployment pipeline, including VPS setup, server configuration, web server deployment, domain mapping, and SSL security, thereby bridging the gap between theory and practical application.

V. METHODOLOGY

The methodology for this project follows a structured pipeline for deploying applications on cloud infrastructure. The approach is designed to ensure systematic setup, configuration, and validation of the system while maintaining security and efficiency.

The process begins with the provisioning of a Virtual Private Server (VPS) using a cloud hosting provider. The server is configured with necessary computational resources and assigned a public IP address, enabling remote connectivity.

Once the server is provisioned, secure access is established using the Secure Shell (SSH) protocol. This allows remote command line control of the server for performing administrative operations and configurations.

The next step involves setting up the Linux based environment by updating system packages and installing required dependencies. This ensures system stability and prepares the server for application deployment.

Following this, a web server (NGINX) is installed and configured to handle incoming HTTP and HTTPS requests. The server configuration is customized to define routing behavior and resource handling.

After configuring the web server, application files are deployed to the server using file transfer mechanisms. The files are placed in the appropriate directory for hosting and accessibility.

To enable user-friendly access, domain name configuration is performed using DNS records. The domain is mapped to the server's public IP address, allowing users to access the application via a readable URL.

Finally, security is implemented by installing an SSL certificate using Certbot, enabling HTTPS communication. The entire system is then validated to ensure proper functionality, security, and accessibility.

This structured methodology ensures a reliable and efficient deployment pipeline, covering all essential stages from infrastructure setup to secure application hosting.

VI. SYSTEM ARCHITECTURE

When a user accesses the application through a web browser, a request is initiated using a domain name. This request is first resolved through the Domain Name System (DNS), which maps the domain name to the public IP address of the cloud server.

Once resolved, the request is transmitted over the internet using HTTP or HTTPS protocols and reaches the Virtual Private Server (VPS) hosted on the cloud. The server runs a Linux based operating system that manages system level operations and resource allocation.

The incoming request is handled by the NGINX web server, which acts as a reverse proxy and request handler. NGINX processes the request and retrieves the appropriate application files from the server directory (such as /var/www/html). If required, it also manages routing and request forwarding based on server configuration.

For secure communication, SSL/TLS encryption is implemented, ensuring that data exchanged between the client and server is protected. This is achieved through HTTPS enabled via SSL certificates.

After processing the request, the server generates an appropriate response, which is sent back to the user's browser through the same network path. The browser then renders the response as a web page.

This architecture ensures efficient request handling, improved performance, secure communication, and the ability to scale resources as per demand in a cloud environment.

VII. IMPLEMENTATION DETAILS

The implementation of the cloud infrastructure was carried out in multiple stages, involving server provisioning, configuration, deployment, and validation.

A. Server Provisioning

A Virtual Private Server (VPS) was provisioned using Hottinger. The server was allocated with required CPU, RAM, and storage resources. A public IP address and root credentials were generated, enabling remote access.

B. Secure Access using SSH

The server was accessed remotely using Secure Shell (SSH). A secure encrypted connection was established using the following command:

```
ssh root@your_server_ip
```

This allowed command line control over the remote Linux server for configuration and management.

C. Linux Environment Setup

The server environment was prepared by updating system packages and installing necessary dependencies:

```
sudo apt update sudo
apt upgrade
```

This ensured that the system was UpToDate and stable for further configuration.

D. Web Server Installation (NGINX)

The NGINX web server was installed to handle incoming HTTP requests:

```
sudo apt install nginx sudo
systemctl start nginx
sudo systemctl enable nginx
```

The server status was verified using: sudo systemctl status nginx

This confirmed that the web server was active and running.

E. Configuration of NGINX

The NGINX configuration file was modified to define server behavior, including domain routing and root directory:

```
sudo nano /etc/nginx/sitesavailable/default
```

The configuration includes setting the root directory for website files, defining the server_name for domain mapping, and configuring request handling.

1) The configuration was tested using:

```
sudo nginx t
```

After successful validation, the server was restarted to apply the changes:
sudo systemctl restart nginx

F. File Deployment

Application files were uploaded to the server directory (/var/www/html) using an FTP client such as FileZilla or WinSCP. This step made the application available for hosting.

G. Domain and DNS Configuration

A domain name was mapped to the server using DNS A records. The domain was configured to point to the VPS public IP address, allowing users to access the application via a readable URL.

H. SSL Certificate Installation

To secure communication, an SSL certificate was installed using Certbot:
sudo apt install certbot python3certbotnginx sudo certbot nginx

This enabled HTTPS and ensured encrypted communication between client and server.

I. Validation and Testing

The deployment was validated by accessing the website through a web browser, checking HTTP and HTTPS responses, and verifying server status and logs. All configurations were tested to ensure the system was secure, stable, and fully functional.

VIII. Results and Analysis

The implemented system successfully demonstrated the deployment of applications on cloud infrastructure using VPS and AWS. The deployment process was completed efficiently, and the application was made accessible over the internet through a domain name with HTTPS security.

A comparative analysis was performed between traditional hosting and cloud-based deployment across key parameters such as deployment time, scalability, cost efficiency, and system performance. The results indicate that cloud-based deployment significantly reduces setup time due to automated provisioning and preconfigured environments.

In terms of scalability, cloud infrastructure provides dynamic resource allocation, allowing the system to handle varying workloads efficiently.

Unlike traditional hosting, where resources are fixed, the cloud environment enables OnDemand scaling, improving system responsiveness and availability.

Performance evaluation, as illustrated in the comparative graph, shows that cloud hosting achieves higher efficiency in terms of response handling and system throughput. The values represented in the graph indicate relative performance improvements, where cloud infrastructure outperforms traditional systems due to optimized resource utilization and reduced latency.

Cost analysis reveals that cloud-based deployment minimizes upfront investment by adopting a pay asyougo model, eliminating the need for physical hardware and reducing maintenance overhead.

Additionally, automation in cloud environments reduces manual configuration efforts, thereby decreasing the likelihood of human errors and improving overall deployment reliability.

Overall, the results validate that cloud infrastructure provides a more efficient, scalable, and cost-effective solution for modern application deployment compared to traditional hosting systems.

IX. Discussion

The implementation of the cloud deployment pipeline demonstrates that even a relatively simple setup can effectively represent real-world cloud engineering practices. By integrating key components such as VPS provisioning, SSH-based remote management, Linux server configuration, NGINX web server deployment, domain mapping, and SSL security, the system reflects the fundamental architecture used in production environments.

One of the key observations from this implementation is the importance of automation and centralized control in cloud environments. Tasks that are traditionally manual in on-premise systems, such as server setup and configuration, can be streamlined in cloud-based platforms, improving efficiency and reducing the likelihood of errors.

Additionally, the use of cloud infrastructure enables remote accessibility and flexibility, allowing system management from any location. The integration of security mechanisms such as SSH authentication and HTTPS encryption further highlights the importance of secure deployment practices in modern applications.

Although the implementation is basic in scale, it provides a strong foundation for understanding advanced concepts such as load balancing, auto scaling, and distributed system design. The project successfully demonstrates how core cloud technologies can be combined to create a functional and secure deployment environment.

X. Conclusion

This paper presents a practical implementation of cloud infrastructure deployment using VPS and AWS, focusing on real-world application hosting and system configuration. The project successfully demonstrates the complete deployment pipeline, including server provisioning, secure access using SSH, web server setup using NGINX, application deployment, domain configuration, and SSL-based security implementation.

The results highlight the advantages of cloud computing over traditional hosting systems, particularly in terms of scalability, cost efficiency, deployment speed, and ease of management. By leveraging cloud technologies, it is possible to overcome the limitations of physical infrastructure and achieve a more flexible and efficient deployment environment.

Furthermore, the project provides valuable hands-on experience in cloud engineering practices, bridging the gap between theoretical knowledge and practical implementation. The successful deployment and validation of the system confirm that cloud-based solutions are well-suited for modern application hosting requirements.

Overall, the study concludes that cloud infrastructure offers a reliable, scalable, and secure platform for application deployment, making it an essential component of modern software systems.

XI. Future Scope

Future improvements include implementing load balancing, autoscaling, CI/CD pipelines, containerization using Docker, and advanced monitoring systems. These enhancements will further improve system performance and automation.

References

[1] Amazon Web Services, "AWS Documentation – Cloud Computing Services," Available: https://docs.aws.amazon.com/
[2] Hostinger, "VPS Hosting Documentation and Setup Guide," Available: https://www.hostinger.com/tutorials/vps
[3] NGINX Inc., "NGINX Official Documentation," Available: https://nginx.org/en/docs/
[4] Canonical Ltd., "Ubuntu Server Guide," Available: https://ubuntu.com/server/docs
[5] M. Armbrust et al., "A View of Cloud Computing," Communications of the ACM, vol. 53, no. 4, pp. 50–58, 2010.
[6] T. Erl, R. Puttini, and Z. Mahmood, Cloud Computing: Concepts, Technology & Architecture, Prentice Hall, 2013.
[7] Google Cloud, "Introduction to Cloud Computing," Available: https://cloud.google.com/learn/whatiscloudcomputing
[8] Certbot, "Certbot Documentation – HTTPS SSL Setup," Available: https://certbot.eff.org/docs/
[9] Red Hat, "What is a Web Server?," Available: https://www.redhat.com/en/topics/webservers/whatisawebserver
[10] DigitalOcean, "How to Install NGINX on Ubuntu," Available: https://www.digitalocean.com/community/tutorials/howtoinstallnginxonubuntu

