Terraform-Based Portfolio Website with Azure Blob Image Hosting

Project Overview:

This project demonstrates an end-to-end Azure Infrastructure as Code (IaC) implementation using Terraform. A single Azure Virtual Machine is provisioned to host a NGINX-based portfolio website. Additionally, an Azure Storage Account and Blob Container are created using Terraform backend configuration. A local image file is uploaded to the Blob Container entirely through Terraform code and consumed directly in the portfolio HTML page running on the VM. No manual uploads or portal-based configurations are performed.

Aim:

The objective of this project is to automate Azure infrastructure provisioning using Terraform, store static assets in Azure Blob Storage, and integrate those assets into a VM-hosted portfolio website without any manual intervention.

Architecture:

- Azure Resource Group
- Virtual Network and Subnet
- Azure Virtual Machine (Ubuntu Linux)
- Network Security Group allowing HTTP (Port 80)
- Azure Storage Account
- Azure Blob Container for static assets

Tools & Technologies Used:

- Microsoft Azure
- Terraform
- Ubuntu Linux
- NGINX Web Server
- Azure Blob Storage
- HTML / CSS

Terraform Backend & Storage Configuration:

Terraform backend configuration is used to manage state securely. Using the same Terraform codebase, an Azure Storage Account and Blob Container are provisioned automatically. A local image file is uploaded to the Blob Container using Terraform resources, ensuring full automation without manual uploads via Azure Portal.

Virtual Machine & NGINX Setup:

After provisioning the VM with Terraform, NGINX is installed using Linux package manager commands. The default web root directory
/var/www/html is overridden with custom portfolio HTML code.

Image Integration into Portfolio:

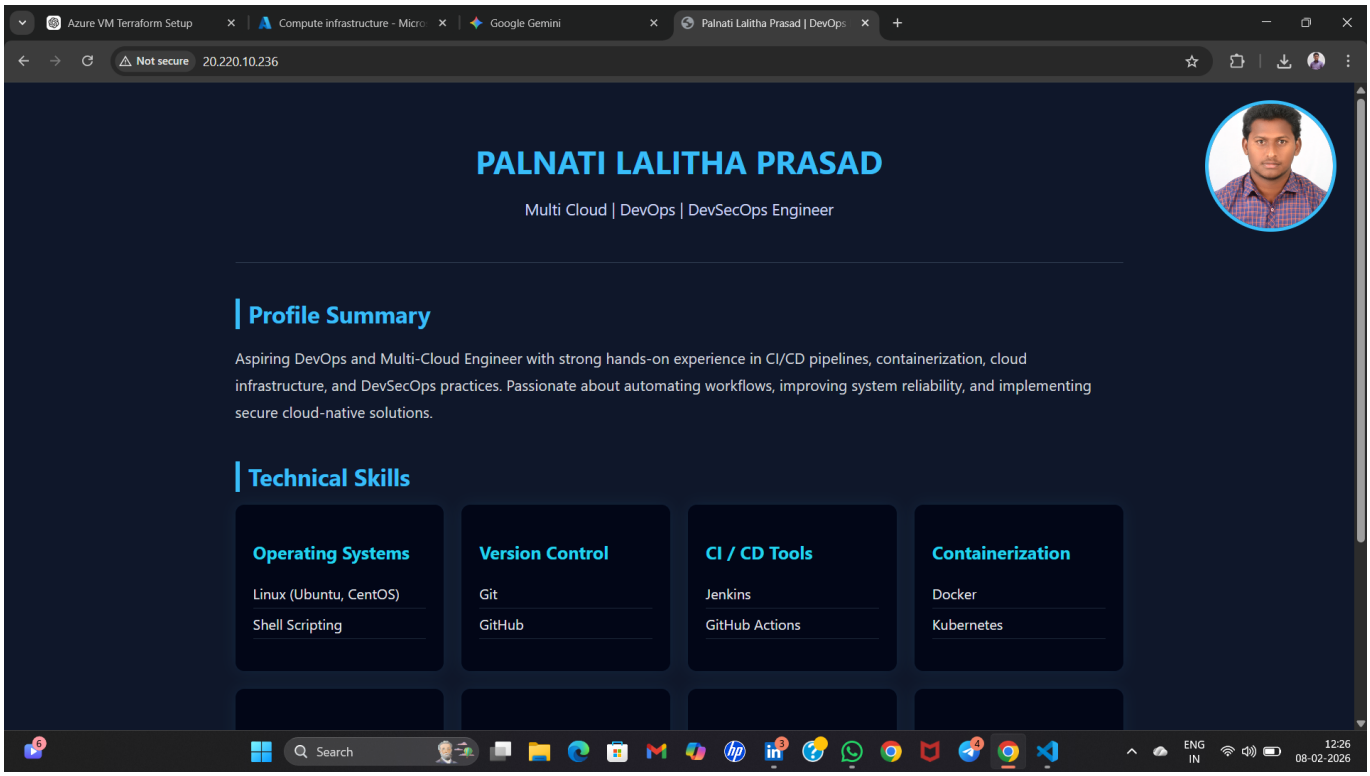
The image uploaded to Azure Blob Storage via Terraform is referenced directly in the portfolio HTML code by calling the Blob. The VM-hosted website dynamically loads the image from Azure Storage, demonstrating separation of compute and static asset storage.

Execution Steps:

- terraform init
- terraform validate
- terraform plan
- terraform apply

Output & Verification:

The portfolio website is accessed using the public IP address of the Azure VM. The image stored in Azure Blob Storage is successfully rendered on the website, confirming correct Terraform-based provisioning and integration.



Conclusion:

This project demonstrates a real-world DevOps workflow using Terraform to manage both compute and storage resources in Azure. It highlights automation, backend state management, and best practices by eliminating manual configuration and uploads entirely.