**🚀 Project Workflow**

This section outlines the structured approach I followed to design, deploy, and showcase a scalable static website infrastructure on AWS using Terraform. My objective was to demonstrate my expertise in cloud engineering, automation, and scalability while solving real-world challenges in cloud-based architecture.

**1. Project Planning**

I began by clearly defining the scope of this project: to deploy a fault-tolerant, highly available infrastructure to host a static website. The key objectives were automation, scalability, and adherence to cloud engineering best practices.

* **Key Tools and Services**:
  + **Terraform**: Infrastructure as Code (IaC) to automate resource provisioning.
  + **AWS Services**: VPC, Subnets, Security Groups, Application Load Balancer (ALB), EC2, and Auto Scaling.
* **Architectural Design**:
  + Public-facing ALB routing traffic to auto-scaled EC2 instances.
  + Resources distributed across multiple Availability Zones for high availability.

**2. Environment Setup**

I ensured the development environment was configured for seamless deployment:

1. Installed and configured **Terraform CLI** and **AWS CLI** with appropriate IAM credentials.
2. Verified network access to AWS resources.
3. Created key project files:
   * **main.tf**: Core Terraform configuration.
   * **variables.tf**: Centralized input parameters for reusability.
   * **outputs.tf**: Defined outputs like the ALB DNS name for easy access.

**3. Infrastructure Development**

I implemented the infrastructure using Terraform, prioritizing modularity and clarity:

1. **VPC & Subnets**: Built a custom VPC with two public subnets for redundancy.
2. **Security**: Designed Security Groups with strict access policies for SSH and HTTP.
3. **Load Balancer**: Configured an ALB to distribute incoming traffic across EC2 instances.
4. **Auto Scaling**: Implemented an Auto Scaling Group with CloudWatch metrics to dynamically adjust capacity based on demand.
5. **Static Website Hosting**: Configured EC2 instances to host a static HTML page, ensuring high availability and scalability.

I committed all configuration files to Git for version control and traceability.

**4. Deployment and Validation**

With the infrastructure code ready, I followed these deployment steps:

1. **Terraform Initialization**:
   * Ran terraform init to download necessary provider plugins and initialize the working directory.
2. **Plan and Apply**:
   * Used terraform plan to preview the resources and ensure correctness.
   * Deployed the infrastructure with terraform apply.
3. **Validation**:
   * Confirmed that all resources were successfully created via the AWS Management Console.
   * Verified accessibility of the static website using the ALB’s DNS.

**5. Testing and Troubleshooting**

I rigorously tested the infrastructure to ensure it met the requirements:

* **Load Balancer Validation**: Verified EC2 instances passed ALB health checks.
* **Scaling Simulation**: Tested the Auto Scaling Group by simulating load conditions, ensuring it scaled up and down as expected.
* **Connectivity**: Confirmed the static website was accessible through the ALB’s DNS.

**6. Documentation and Deliverables**

To make this project easy to understand and reproducible, I documented every step:

* **Architecture Diagram**: Created a clear visual representation of the infrastructure.
* **Screenshots**: Included screenshots of Terraform output, AWS resources, and the final deployed website.
* **README File**: Wrote a detailed README explaining the project’s purpose, architecture, setup steps, and key results.

**7. Resource Cleanup**

After testing, I deprovisioned the infrastructure to minimize costs by running terraform destroy. This step highlighted my attention to resource efficiency and cost management.

**8. Showcasing the Project**

I prepared the project for publication to maximize its impact:

1. **GitHub Repository**:
   * Structured the repository to highlight my approach, including configuration files, architecture diagrams, and screenshots.
2. **Portfolio Website**:
   * Featured this project as a key highlight, emphasizing my ability to build scalable cloud infrastructure.
3. **LinkedIn Post**:
   * Shared my learnings and achievements, engaging my professional network and showcasing my technical expertise.

**9. Key Takeaways**

Through this project, I demonstrated:

* Proficiency in Terraform for automating cloud resource provisioning.
* Hands-on experience with AWS services to design and deploy scalable, fault-tolerant infrastructure.
* Strong problem-solving skills, addressing challenges like instance connectivity and ALB configurations.
* Commitment to delivering high-quality, well-documented, and reproducible solutions.

This workflow captures not only my technical skills but also my systematic approach to problem-solving and my ability to deliver robust cloud solutions. This project is a testament to my readiness for a Cloud Support Engineer role and my dedication to continuous learning.