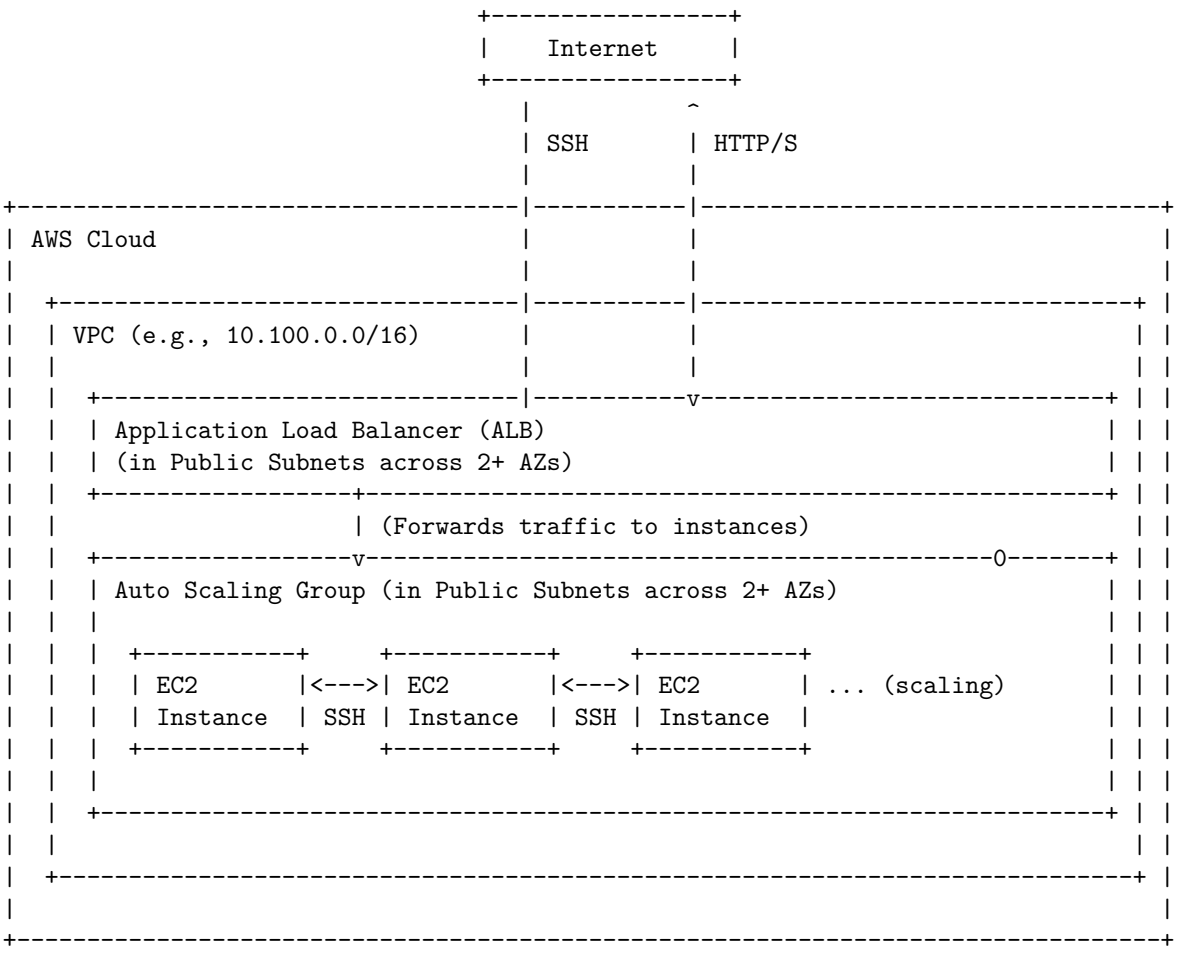
Terraform AWS Web Application Stack

This Terraform project deploys a scalable, highly available, and secure web application infrastructure on AWS. It is designed to be modular, reusable, and easily configurable for different environments.

The architecture includes a custom VPC, an internet-facing Application Load Balancer, and an Auto Scaling Group of EC2 instances running in **public** subnets to allow for direct SSH access. It follows AWS best practices for security, resilience, and operational excellence.

Architecture Diagram



Prerequisites

- Terraform >= 1.0
- An AWS account with the necessary permissions.
- AWS CLI configured with credentials (or an EC2 instance profile). The configuration uses the profile `epcvip-asg`.
- The provided shell scripts must be executable: `chmod +x ../*.sh`

Workflow using Helper Scripts

This project includes a set of bash scripts to streamline the Terraform workflow for different environments (e.g., `dev`, `uat`, `prod`). Using these scripts is the recommended approach.

1. **Configure Backend:** For production or team use, uncomment the `s3` backend configuration in `terraform/backend.tf`. You will also need to create environment-specific backend configuration files (e.g., `terraform/backend-dev.config`) for the `terraform_init.sh` script to use.

2. **Initialize Terraform:** Run the init script for your target environment. This will configure the backend and download the necessary providers.

   ```
   ../terraform_init.sh dev
   ```

3. **Plan the Deployment:** Run the plan script to see what changes will be made. This is a safe way to preview changes without applying them.

   ```
   ../terraform_plan.sh dev
   ```

4. **Apply the Configuration:** The apply script will first generate a plan file and then apply it, ensuring that only the planned changes are executed.

   ```
   ../terraform_apply.sh dev
   ```

5. **Destroy the Infrastructure:** When you no longer need the resources, you can destroy them using the destroy script. **Note:** This script uses `-auto-approve` and will not ask for confirmation.

   ```
   ../terraform_destroy.sh dev
   ```

Modules

This project is composed of the following modules:

- `./modules/basic_vpc`: Creates a VPC with public and private subnets, an Internet Gateway, and a NAT Gateway.
- `./modules/application_load_balancer`: Deploys an Application Load Balancer with listeners and security groups.
- `./modules/ec2_launch_template`: Defines an EC2 launch template, including AMI selection and **EC2 Key Pair for SSH access**.
- `./modules/ec2_asg_target`: Creates an Auto Scaling Group, a Target Group, and associated scaling policies and security groups.

Inputs

Name	Description	Type	Default	Required
`target_environment`	The target environment (e.g., dev, uat, prd) to append to resource names.	`string`	n/a	yes

Outputs

Name	Description
`alb_dns_name`	The DNS name of the created Application Load Balancer.
`alb_security_group_id`	The ID of the security group attached to the ALB.
`ec2_asg_target_group_arn`	The ARN of the web application's ALB Target Group.
`ec2_asg_target_asg_name`	The name of the web application's Auto Scaling Group.
`vpc_id`	The ID of the created VPC.
`public_subnet_ids`	A list of IDs of the created public subnets.
`private_subnet_ids`	A list of IDs of the created private subnets.
`ssh_private_key_path`	The local path where the generated SSH private key (.pem) is stored. Use this for SSH access.
`generated_key_pair_name`	The name of the EC2 key pair created and managed by Terraform.