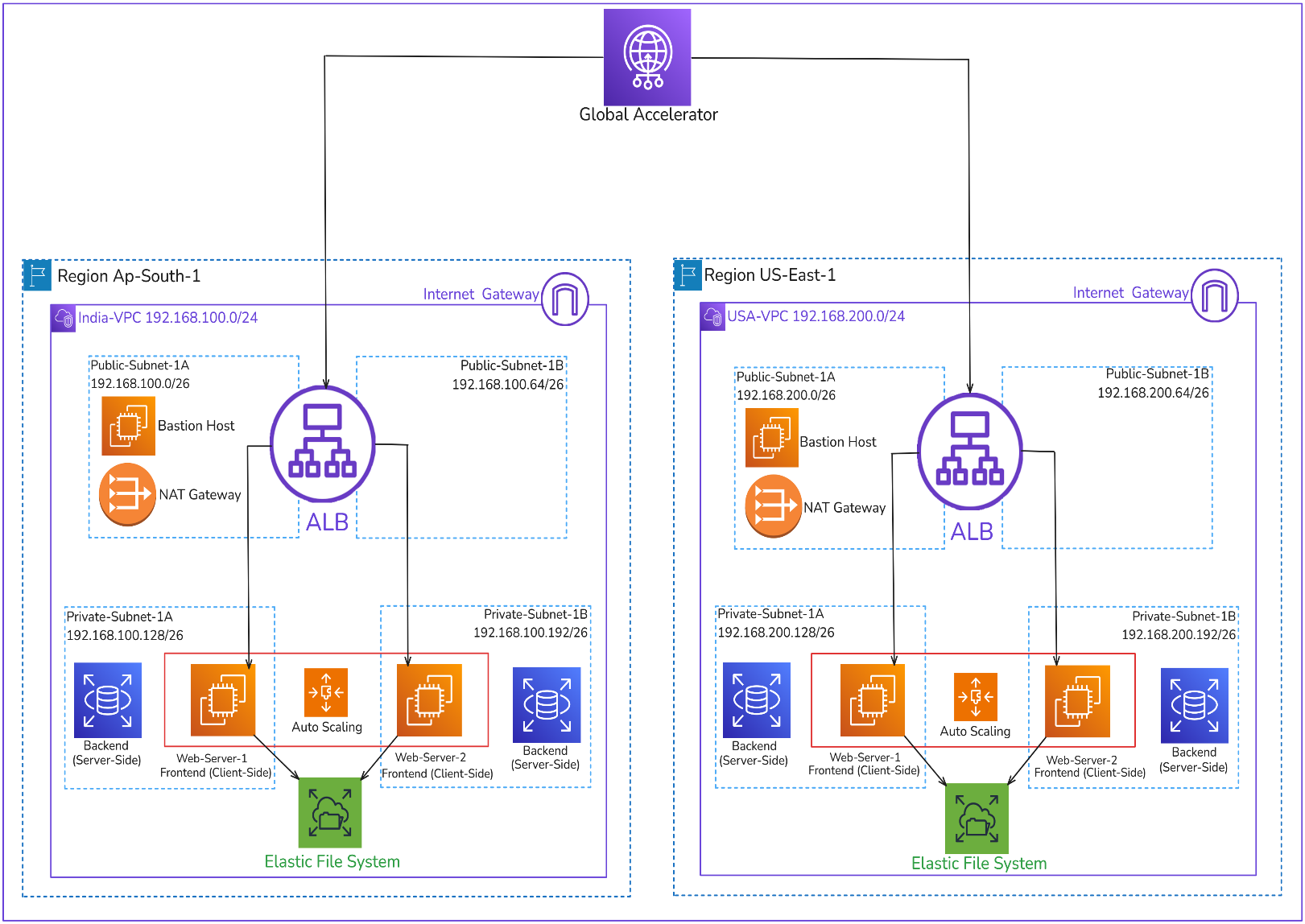
**Project Title: Multi-Region Web Acceleration with High Availability using AWS Global Accelerator**

**Objective:** This project demonstrates a **highly available**, **fault tolerance**, and **global low-latency access** to web applications by deploying resources across two AWS regions**: Asia Pacific (Mumbai) ap-south-1**, **US East (N. Virginia) us-east-1**. The **AWS Global Accelerator** ensures global users are routed to the **nearest healthy region**, optimizing performance and uptime.

**Architecture Overview:**

****

**Key Components**

* **AWS Global Accelerator** – Directs user traffic to the optimal regional endpoint based on latency and health status, ensuring low-latency access and high availability.
* **Amazon VPC (India & USA)** – Two isolated Virtual Private Clouds, each configured with public and private subnets to host application components securely and efficiently.
* **Elastic Load Balancer (ALB)** – Distributes incoming traffic across EC2 instances in Auto Scaling groups located in private subnets, improving scalability and fault tolerance.
* **Amazon EC2 Instances (Web Servers)** – EC2 instances are launched in private subnets where web servers **(Apache)** are installed to host the frontend application. These servers are part of Auto Scaling groups for high availability.
* **Amazon EC2 Auto Scaling** – Automatically launches or terminates web server EC2 instances based on demand to maintain performance and cost-efficiency.
* **Amazon RDS (MySQL)** – A managed relational database deployed in private subnets, accessible only from web server instances, ensuring secure and reliable data storage.
* **Amazon EFS** – A shared file storage system mounted on web servers in each region, allowing consistent frontend content across all instances.
* **NAT Gateway & Internet Gateway** –
  + **Internet Gateway (IGW):** Provides internet access to public subnets.
  + **NAT Gateway:** Allows instances in private subnets to access the internet for updates without being exposed.
* **Bastion Host** – Deployed in public subnets to securely connect **(via SSH)** to private subnet resources, such as EC2 web and database servers.

**Project Overview:**

**Creating a Virtual Private Cloud (VPC):**

**1. VPC Setup (Region: ap-south-1 - India):**

* Create a custom VPC with a CIDR block **(192.168.100.0/24)** to provide an isolated network environment.
* Create public and private subnets across multiple Availability Zones to support high availability.
* Attach an Internet Gateway (IGW) to the VPC and update the route table for public subnet internet access.
* Create a NAT Gateway in the public subnet to allow outbound internet access from the private subnet **(for updates, install web server)**.
* Configure route tables for public and private subnets with appropriate routing rules.

**Create a route table for the public subnet and associate it with the subnet.**

* Add a route to the **Internet Gateway** **(IGW)** (0.0.0.0/0 → IGW) to allow internet access.

**Create a route table for the private subnet and associate it with the private subnet**.

* Add a route to the **NAT Gateway** (0.0.0.0/0 → NAT Gateway) to allow outbound internet access while keeping the instance private.

**VPC Configuration:**

* **CIDR Blocks:**
  + India: 192.168.100.0/24
* **Subnets:**
  + Public Subnet-A: 192.168.100.0/26
  + Public Subnet-B: 192.168.100.64/26
  + Private Subnet-A: 192.168.100.128/26
  + Private Subnet-B: 192.168.100.192/26
* **Components in Public Subnets:**
  + Bastion Host (SSH Access)
  + NAT Gateway
* **Components in Private Subnets:**
  + Auto Scaling EC2 Web Servers
  + EFS Mounts
  + RDS Instance (MySQL)

**2. VPC Setup (Region: us-east-1 - USA):**

* Create a custom VPC with a CIDR block **(192.168.200.0/24)** to provide an isolated network environment.
* Create public and private subnets across multiple Availability Zones to support high availability.
* Attach an Internet Gateway (IGW) to the VPC and update the route table for public subnet internet access.
* Create a NAT Gateway in the public subnet to allow outbound internet access from the private subnet **(for updates, install web server)**.
* Configure route tables for public and private subnets with appropriate routing rules.

**Create a route table for the public subnet and associate it with the subnet.**

* Add a route to the **Internet Gateway** **(IGW)** (0.0.0.0/0 → IGW) to allow internet access.

**Create a route table for the private subnet and associate it with the private subnet**.

* Add a route to the **NAT Gateway** (0.0.0.0/0 → NAT Gateway) to allow outbound internet access while keeping the instance private.

**VPC Configuration**

* **CIDR Blocks:**
  + USA: 192.168.200.0/24
* **Subnets:**
  + Public Subnet-A: 192.168.200.0/26
  + Public Subnet-B: 192.168.200.64/26
  + Private Subnet-A: 192.168.200.128/26
  + Private Subnet-B: 192.168.200.192/26
* **Components in Public Subnets:**
  + Bastion Host (SSH Access)
  + NAT Gateway
* **Components in Private Subnets:**
  + Auto Scaling EC2 Web Servers
  + EFS Mounts
  + RDS Instance (MySQL)

**Security Group Configuration:**

* **NACLs**: Optional additional layer of network security.
* **IAM Roles**: EC2 instances use roles for EFS and CloudWatch logging.

**Web Server Security Group**

**Web-Server-SG-IN:** Security group attached to web servers in India

**Web-Server-SG-US:** Security group attached to web servers in USA

* **Inbound Rules**:
  + Allow **HTTP (port 80)** and **HTTPS (port 443)** from the **ALB SG**
  + Allow **MySQL (port 3306)** from the **RDS SG** (for app-db interaction)
* **Outbound Rules**:
  + Allow all traffic to enable updates and communication (can be restricted further)
  + Especially ensure port 2049 (NFS) is open for **EFS**.

**EFS Security Group (EFS-SG):**

**Inbound Rules (Allow Mount Access):**

**Type Protocol Port Range Source Description**

NFS TCP 2049 Web-Server-SG-IN Allow web servers from India region

NFS TCP 2049 Web-Server-SG-IN Allow web servers from India region

**Outbound rule Rules:**

**Type Protocol Port Range Source Description**

All Traffic All All 0.0.0.0/0 (Default) Allow all outbound traffic

**RDS Security Group**

* **Inbound Rules**:
  + Allow **MySQL/Aurora (port 3306)** only from the **Web Server SG**
* **Outbound Rules**:
  + Allow all (default setting unless using VPC Peering/Private-Link)

**Bastion Host Security Group**

* **Inbound Rules**:
  + Allow **SSH (port 22)** only from **your specific IP** (use your home/office static IP)
* **Outbound Rules**:
  + Allow all (for admin tasks like updates or repo access)

**ALB Security Group**

* **Inbound Rules**:
  + Allow **HTTP (80)** and **HTTPS (443)** from **anywhere (0.0.0.0/0)** (or restrict by region)
* **Outbound Rules**:
  + Allow traffic to **Web Server Security Group** on ports **80/443**

**Deployment Configuration:**

**Web Server Configuration (Frontend Deployment):**

* **Tech Stack**: HTML, CSS, PHP (can be any framework)
* **Deployment**:
  + Frontend files are hosted on EC2 instances (web servers) in private subnets.
  + EFS is used to keep frontend files synced across multiple web servers.
  + Apache is configured to serve the frontend.

**RDS Database Setup:**

* **Engine**: MySQL
* **Deployed in**: Private Subnet-A and Subnet-B
* **Security**:
  + Only accessible from web servers' security group.
  + Port **3306** open only to internal traffic.
* **High Availability**: Enabled with Multi Availability Zone deployment.

**EC2, AMI, and Auto Scaling Setup Steps for Web Server with EFS + RDS Integration:**

**Step 1: Launch EC2 Instance (Free Tier)**

Launch a t2-micro EC2 instance in the public subnet (for initial configuration).

Choose Amazon Linux 2 or Ubuntu as the base AMI.

Use a key pair for SSH access.

Assign a security group allowing SSH (22) and HTTP (80) if needed.

Attach IAM role with permission to access EFS and SSM (optional).

**Step 2: Install and Configure Web Server**

* SSH into the EC2 instance.
* Install Apache-

sudo yum update -y

sudo yum install -y httpd

sudo systemctl start httpd

sudo systemctl enable httpd

**Step 3: Mount EFS**

sudo yum install -y amazon-efs-utils

**Mount EFS to a directory (e.g., /var/www/html):**

sudo mkdir -p /var/www/html

sudo mount -t efs fs-xxxxxxx:/ /var/www/html

Add EFS to /etc/fstab for auto-mount on boot.

**Step 4: Add Web Files and Configurations**

* Deploy website content in /var/www/html (EFS mount).
* Web server now serves content stored in EFS — **shared across all future EC2s**.

**Step 5: Install MySQL Client & Store Credentials**

**Install MySQL client:**

sudo yum install -y mysql

**Create a credentials file in EFS (/var/www/html/db-config.php)**

<?php

$dbhost = "your-rds-endpoint";

$dbuser = "admin";

$dbpass = "your-password";

$dbname = "your-db-name";

?>

**Step 6: Test RDS Connection**

mysql -h your-rds-endpoint -u admin -p

**Step 7: Create AMI from Configured EC2**

* Stop the instance (optional).
* Go to EC2 console → Actions > Create Image (AMI).
* Name it like webserver-efs-rds-base-ami.

**Step 8: Create Launch Template**

* Go to EC2 → Launch Templates → Create Template.
* Select your AMI ID.
* Choose:
* Instance type: t2-micro (for free tier / testing).
* Key pair
* IAM role
* Security group (Web Server SG)
* User data (if needed for remounting EFS or service restart).
* Save the template.

**Auto Scaling Group:**

* Use the launch template just created.
* Automatically adjusts the number of EC2 instances in response to traffic patterns.
* Maintains performance and reduces costs by adding/removing instances.
* Integrated with **ALB health checks** for replacing unhealthy instances.

**Elastic File System (EFS):**

* Mounted on all **Web Servers** across Availability Zones within each region.
* Provides a shared file system for storing:
  + Static assets (images, HTML, CSS)
  + User uploads
  + Web content that needs to be consistent across instances
* Ensures **data consistency** and **centralized storage**.

**Application Load Balancer (ALB):**

* Placed in each region to distribute traffic evenly across web servers.
* Integrated with **Auto Scaling Groups**.
* Registered targets are EC2 instances in the **private subnets**.
* Accepts requests from **Global Accelerator** and routes them to the appropriate backend.

**Global Accelerator Configuration:**

**Purpose:** Provides a single static IP address for your global application and routes traffic to the optimal AWS region based on latency and health checks**.**

**Integration:**

* Routes traffic to Application Load Balancers (ALBs) in both India and USA regions.
* Ensures users around the world reach the nearest, most responsive infrastructure.
* **Listener**: TCP on ports 80 and 443
* **Endpoints**: ALBs in ap-south-1 and us-east-1
* **Health checks**: Route traffic to healthy endpoints only
* **Traffic Distribution**: Automatically routes users to the closest healthy region

**Benefits:**

* Reduced latency for users across the globe.
* High availability using multi-region failover.
* Scalable frontend using Auto Scaling.
* Centralized and persistent storage with EFS.
* Managed database layer with RDS.
* Simple SSH management via Bastion Host.

**Testing & Validation**

* Test access from different global locations using a **VPN** or **GeoPeeker** (See how a site appears to the rest of the world).
* Simulate instance failure and validate Global Accelerator’s failover.
* Verify EFS syncing across web servers.
* Confirm database connectivity from web layer.