LAMP Stack Application Architecture - AWS Deployment

Project Overview

This document outlines the architecture for deploying a simple LAMP stack application on AWS, following Well-Architected Framework principles with emphasis on scalability and availability.

Architecture Components

Core LAMP Stack Components

- **Linux**: Operating system foundation (Amazon Linux 2 or Ubuntu)
- **Apache**: Web server for serving HTTP requests
- **MySQL**: Relational database for data storage
- **PHP**: Server-side scripting language for application logic

AWS Infrastructure Components

- **EC2 Instances**: Virtual servers hosting the web application
- **RDS (MySQL)**: Managed database service for data persistence
- **Application Load Balancer**: Traffic distribution and high availability
- **Auto Scaling Group**: Automatic scaling based on demand
- **VPC**: Isolated network environment with public/private subnets
- **Security Groups**: Network-level security controls

High-Level Architecture

- Internet Gateway
- ↓
- Application Load Balancer (Public Subnet)
- ↓
- Web Servers - EC2 Instances (Private Subnet)
- ↓
- RDS MySQL Database (Private Subnet)

Requirements Analysis

Scalability Requirements

- **Horizontal Scaling**: Auto Scaling Group to handle traffic spikes
- **Database Scaling**: RDS with read replicas if needed
- **Load Distribution**: Application Load Balancer for traffic management

Availability Requirements

- **Multi-AZ Deployment**: Resources across multiple Availability Zones
- **Database Backup**: Automated RDS backups and snapshots
- **Health Checks**: Load balancer health monitoring

Security Requirements

- **Network Security**: VPC with public/private subnet separation
- **Access Control**: Security groups with least privilege principles
- **Database Security**: RDS in private subnet, encrypted at rest
- **Web Security**: HTTPS termination at load balancer

Performance Requirements

- **Expected Traffic**: Moderate web traffic (specify based on use case)
- **Response Time**: Target < 2 seconds for page loads
- **Database Performance**: Optimized queries and appropriate instance sizing

Network Design

VPC Configuration

- **CIDR Block**: 10.0.0.0/16
- **Public Subnets**: 10.0.1.0/24, 10.0.2.0/24 (for load balancer)
- **Private Subnets**: 10.0.3.0/24, 10.0.4.0/24 (for web servers and database)

Security Groups

- **Web Server SG**: Allow HTTP/HTTPS from load balancer, SSH from bastion
- **Database SG**: Allow MySQL (3306) from web server security group only
- **Load Balancer SG**: Allow HTTP/HTTPS from internet

Deployment Strategy

- **Infrastructure as Code**: Use CloudFormation or Terraform
- **Application Deployment**: Automated deployment pipeline
- **Configuration Management**: Consistent server configuration

- **Monitoring**: CloudWatch for metrics and logging

Success Criteria

- Application accessible via load balancer URL
- Database connectivity established and functional
- Auto scaling responds to load changes
- Multi-AZ deployment provides high availability
- Security groups properly restrict access