AWS Web Application Infrastructure Architecture Summary

Project: Scalable Web Application Infrastructure
Technology: Terraform Infrastructure as Code
Cloud Provider: Amazon Web Services (AWS)
Document Version: 1.0
Date: November 2025
Repository: https://github.com/sthakur1985/aws-web-app-deploy

---

Executive Summary

This document provides a comprehensive overview of a production-ready, scalable web application infrastructure deployed on AWS using Terraform. The architecture implements a 3-tier design with security best practices, high availability, and multi-environment support.

Key Benefits

- **Scalability**: Auto Scaling Groups with configurable capacity
- **Security**: Multi-layer security with private subnets and centralized IAM
- **High Availability**: Multi-AZ deployment across 2 availability zones
- **Cost Optimization**: Environment-specific resource sizing
- **Compliance**: Security frameworks and audit trail capabilities

---

Page 1: Architecture Overview & Network Design

3-Tier Architecture Design

The infrastructure follows a modern 3-tier architecture pattern:

Presentation Tier (Public Layer)

- **Application Load Balancer (ALB)**: Traffic distribution and SSL termination
- **Public Subnets**: 2 subnets across 2 Availability Zones
- **Security**: HTTPS enforcement, security headers, CIDR-based access control
- **Features**: HTTP/2 support, connection draining, health checks

Application Tier (Private Layer)

- **Auto Scaling Group**: 1-10 EC2 instances based on environment
- **EC2 Private Subnets**: 2 subnets across 2 AZs with NAT Gateway access
- **Security**: No public IPs, security groups, centralized IAM roles
- **Features**: Automatic scaling, health monitoring, user data automation

Data Tier (Isolated Layer)

- **RDS MySQL/PostgreSQL**: Managed database service
- **RDS Private Subnets**: 2 subnets across 2 AZs, no internet access
- **Security**: Encryption at rest/transit, AWS Secrets Manager, network isolation
- **Features**: Multi-AZ deployment, automated backups, performance monitoring

Network Architecture

```
Internet Gateway
       |
   Public Subnets (2 AZs)
   [ALB] [ALB]
       |
   NAT Gateways (2)
       |
   EC2 Private Subnets (2 AZs)
   [EC2] [EC2] [EC2]
       |
   RDS Private Subnets (2 AZs)
   [RDS Primary] [RDS Standby]
```

Subnet Design Strategy

- **CIDR Allocation**: /16 VPC with /24 subnets
- **Public Subnets**: 10.0.1.0/24, 10.0.2.0/24 (ALB placement)
- **EC2 Private**: 10.0.3.0/24, 10.0.4.0/24 (Application servers)
- **RDS Private**: 10.0.5.0/24, 10.0.6.0/24 (Database isolation)

Security Groups & Network ACLs

- **ALB Security Group**: HTTP/HTTPS from internet (0.0.0.0/0)
- **EC2 Security Group**: HTTP from ALB security group only
- **RDS Security Group**: MySQL/PostgreSQL port from EC2 security group only
- **Network Isolation**: RDS subnets have no internet routes

Security Architecture

Defense in Depth Strategy

1. **Network Layer**: VPC, subnets, security groups, NACLs
2. **Application Layer**: ALB security headers, HTTPS enforcement
3. **Data Layer**: Encryption, Secrets Manager, network isolation
4. **Identity Layer**: Centralized IAM, least privilege access
5. **Monitoring Layer**: VPC Flow Logs, CloudWatch alarms

Compliance & Governance

- **Encryption**: AES-256 for S3, RDS encryption at rest
- **Access Control**: Role-based access with IAM policies
- **Audit Trail**: CloudTrail logging, VPC Flow Logs
- **Secret Management**: AWS Secrets Manager for database credentials
- **Tagging Strategy**: Comprehensive resource tagging for cost allocation

---

Page 2: Service Components & Integration

☁ Content Delivery & DNS

CloudFront CDN

- **Global Distribution**: 400+ edge locations worldwide
- **Performance**: Intelligent caching, compression, HTTP/2
- **Security**: Origin Access Control (OAC), HTTPS enforcement
- **Cost Optimization**: Configurable price classes per environment

Route53 DNS Management

- **Hosted Zones**: Optional creation or existing zone integration
- **Alias Records**: ALB and CloudFront integration
- **Health Checks**: Automatic failover capabilities
- **Subdomain Strategy**: Configurable organization (app.domain.com, cdn.domain.com)

S3 Static Content

- **Security**: Public access blocked, OAC integration
- **Features**: Versioning enabled, server-side encryption
- **Integration**: CloudFront origin with bucket policies
- **Cost**: Lifecycle policies for storage optimization

Identity & Access Management

Centralized IAM Strategy

- **EC2 Role**: Systems Manager access, CloudWatch agent permissions
- **RDS Monitoring Role**: Enhanced monitoring capabilities (optional)
- **Instance Profiles**: Secure EC2 service access
- **Principle of Least Privilege**: Minimal required permissions

Security Best Practices

- **No Hardcoded Credentials**: AWS Secrets Manager integration
- **Service Roles**: Dedicated roles per AWS service
- **Managed Policies**: AWS-maintained security updates
- **Resource Tagging**: Complete metadata for governance

Monitoring & Observability

CloudWatch Integration

- **EC2 Monitoring**: CPU utilization, instance health
- **RDS Monitoring**: Database performance, storage capacity
- **ALB Monitoring**: Response times, error rates, unhealthy hosts
- **Custom Thresholds**: Environment-specific alert levels

Alerting Strategy

- **SNS Integration**: Email, SMS, webhook notifications
- **Escalation Policies**: Different thresholds per environment
- **Cost Control**: Optional monitoring features
- **Dashboard**: Centralized operational visibility

Deployment & Automation

Infrastructure as Code

- **Terraform Modules**: Reusable, parameterized components
- **Environment Separation**: Dev, staging, production configurations
- **State Management**: S3 backend with DynamoDB locking
- **Version Control**: Git-based infrastructure versioning

CI/CD Integration

- **GitHub Actions**: Automated deployment workflows
- **Environment Promotion**: Consistent deployment across environments
- **Secret Management**: Pipeline-based credential handling
- **Testing**: Validation and dry-run capabilities

Page 3: Environment Strategy & Operational Considerations

Multi-Environment Architecture

Environment Differentiation

Component	Development	Staging	Production
EC2 Instances	t3.micro (1-2)	t3.small (1-3)	t3.medium (2-10)
RDS Configuration	Single-AZ, t3.micro	Multi-AZ, t3.small	Multi-AZ, t3.medium
Monitoring	Basic CloudWatch	Enhanced monitoring	Full monitoring + PI
Backup Retention	7 days	14 days	30 days
SSL Certificates	Optional	Required	Required
Flow Logs	Disabled	Enabled	Enabled
Cost Optimization	Aggressive	Balanced	Performance-focused

Environment Isolation Strategy

- **Separate AWS Accounts**: Complete isolation (recommended)
- **Separate VPCs**: Network-level isolation within account
- **Separate State Files**: Independent Terraform state management
- **Environment-Specific Variables**: Tailored configurations

Cost Optimization Strategy

Resource Sizing

- **Development**: Minimal resources, single AZ deployment
- **Staging**: Production-like but smaller scale
- **Production**: Performance and availability optimized

Cost Components (Monthly Estimates)

- **EC2 Instances**: $8-50 per instance (t3.micro to t3.medium)
- **RDS**: $15-100 (db.t3.micro to db.t3.medium, Multi-AZ)
- **ALB**: $16 base + $0.008 per LCU
- **NAT Gateways**: $45 each (2 for HA)
- **CloudFront**: $0.085 per GB + requests
- **S3**: $0.023 per GB + requests

Cost Control Measures

- **Auto Scaling**: Automatic capacity adjustment
- **Reserved Instances**: Long-term cost reduction

- **Spot Instances**: Development environment optimization
- **Lifecycle Policies**: S3 storage class transitions

Operational Excellence

Deployment Process

1. **Code Review**: Terraform configuration validation
2. **Planning**: Dry-run execution with terraform plan
3. **Approval**: Manual approval for production changes
4. **Deployment**: Automated terraform apply
5. **Validation**: Post-deployment health checks

Disaster Recovery

- **Multi-AZ**: Automatic failover capabilities
- **Backup Strategy**: Automated RDS backups, S3 versioning
- **Infrastructure Recovery**: Terraform-based infrastructure recreation
- **Data Recovery**: Point-in-time recovery for RDS

Security Operations

- **Patch Management**: Systems Manager patch automation
- **Vulnerability Scanning**: Regular security assessments
- **Access Reviews**: Periodic IAM permission audits
- **Incident Response**: CloudWatch alarms and notification workflows

Performance Optimization

- **Auto Scaling Policies**: CPU and custom metric-based scaling
- **Database Optimization**: Performance Insights, parameter tuning
- **CDN Optimization**: CloudFront caching strategies
- **Application Monitoring**: Custom application metrics

Scalability & Future Enhancements

Horizontal Scaling

- **Auto Scaling Groups**: Automatic instance management
- **Load Balancer**: Traffic distribution across instances
- **Database**: Read replicas for read-heavy workloads
- **CDN**: Global edge location distribution

Future Enhancements

- **Container Integration**: ECS/EKS for containerized workloads
- **Serverless Components**: Lambda functions for specific tasks
- **Advanced Monitoring**: X-Ray tracing, custom dashboards
- **Security Enhancements**: WAF integration, GuardDuty monitoring

---

Conclusion

This AWS web application infrastructure provides a robust, scalable, and secure foundation for modern web applications. The architecture implements industry best practices for security, availability, and operational excellence while maintaining cost

efficiency through environment-specific optimizations.

Key Success Factors: - Modular Terraform design for maintainability - Multi-environment strategy for safe deployments - Comprehensive security implementation - Automated monitoring and alerting - Cost-optimized resource allocation

Contact Information:
Technical Lead: Soumya Thakur (soumyathakur85@gmail.com)
Project Repository: AWS Web Application Infrastructure
Documentation: Complete module-level README files available