Project 2: Scaling an E-Commerce App for High Traffic

# 1. Project Overview

|  |  |
| --- | --- |
| Client Name | ShopWave |
| Industry | E-Commerce |
| Problem Summary | Application crashes during seasonal or marketing campaigns due to load spikes |
| Current AWS Setup | EC2 + RDS |
| Business Goal | Ensure the app is highly scalable and available during traffic surges |
| Constraints | Budget-conscious: prefer Free Tier or Spot Instances. Small team: only 2 DevOps engineers, setup must remain easy to maintain. |

# 2. Discovery Phase

Key findings from the client discussion:

|  |  |
| --- | --- |
| Key Question | Answer |
| How does traffic behave? | ~10k users/day normally; spikes to 150k/day during sales. |
| Current App Hosting? | Monolithic Node.js app on a t3.medium EC2 (single AZ). |
| Database Setup? | RDS MySQL, single AZ. Read-heavy load slows it down. |
| Any caching layer? | None yet — all reads go to RDS directly. |
| CDN or S3? | S3 is used for images, but no CloudFront; EC2 serves all content. |
| Deployment Process? | Manual via SSH; PM2 restarts app with ~30s downtime. |
| Scaling Setup? | Basic CloudWatch alarm triggers 1 extra EC2 manually. |
| Monitoring? | Minimal CloudWatch, no alarms or log forwarding. |
| Pain Points? | Crashes under load, DB bottleneck, downtime during deploys. |
| Tooling Preferences? | GitHub and Terraform. No plans to adopt containers yet. |

# 3. Proposed High-Level Architecture

The architecture improves scalability and availability across four layers:

- Frontend & CDN: CloudFront distributes requests; S3 serves static assets.

- Application: ALB manages traffic across EC2 ASG. Cognito handles user auth. WAF and Shield protect traffic.

- Data Layer: Multi-AZ RDS with Read Replica and Redis for caching frequent reads.

- DevOps: GitHub Actions for CI/CD. CloudWatch, SNS for monitoring. Secrets Manager and VPC Flow Logs for security.

# 4. Service Selection & Justification

|  |  |  |
| --- | --- | --- |
| AWS Service | Purpose | Why This Service Was Chosen |
| EC2 + ASG | Host app and scale with traffic | Simpler than containers for a small team |
| ALB | Distribute traffic to EC2 | Handles spikes, supports autoscaling |
| RDS + Replica | Store relational data | Replica reduces read load, improves performance |
| S3 + CloudFront | Serve static content | Reduces EC2 load, speeds up delivery |
| IAM | Secure access control | Ensures least-privilege access |
| GitHub Actions | CI/CD | Simple, integrates well with GitHub-based workflow |
| CloudWatch + SNS | Monitoring and alerting | Tracks health and notifies DevOps |

# 5. DevOps Additions

|  |  |
| --- | --- |
| Tool | Use Case |
| Terraform | Provision infrastructure as code |
| CI/CD | Deploy app updates safely |
| CloudWatch | Set alerts and review logs |
| IAM Roles & Secrets Manager | Secure environment variables and access |

# 6. Phased Implementation Plan

|  |  |
| --- | --- |
| Phase | Description |
| Phase 1 | Migrate to ALB + ASG + CloudFront |
| Phase 2 | Add Read Replica and Redis cache |
| Phase 3 | Set up CI/CD, monitoring |
| Phase 4 | Optionally containerize the app |

# 7. Estimated Cost Plan

|  |  |  |
| --- | --- | --- |
| Resource | Tier/Type | Monthly Est. (USD) |
| EC2 | t3.medium (ASG) | ~$18–$54 |
| RDS | db.t3.micro (Multi-AZ) | ~$30–$60 |
| CloudFront | 1TB/mo | ~$0–$10 |
| ElastiCache | cache.t2.micro | ~$15 |
| CloudWatch Logs | Basic + 1 alarm | ~$2–$5 |
| Route 53 | Optional DNS | ~$1 |
| Secrets Manager | 3 secrets | ~$1.20 |
| S3 | ~5–10GB | ~$0.50–$1 |

Total Estimated Monthly Cost: ~$70–130 (depending on usage & Free Tier eligibility)

# 8. Final Deliverables

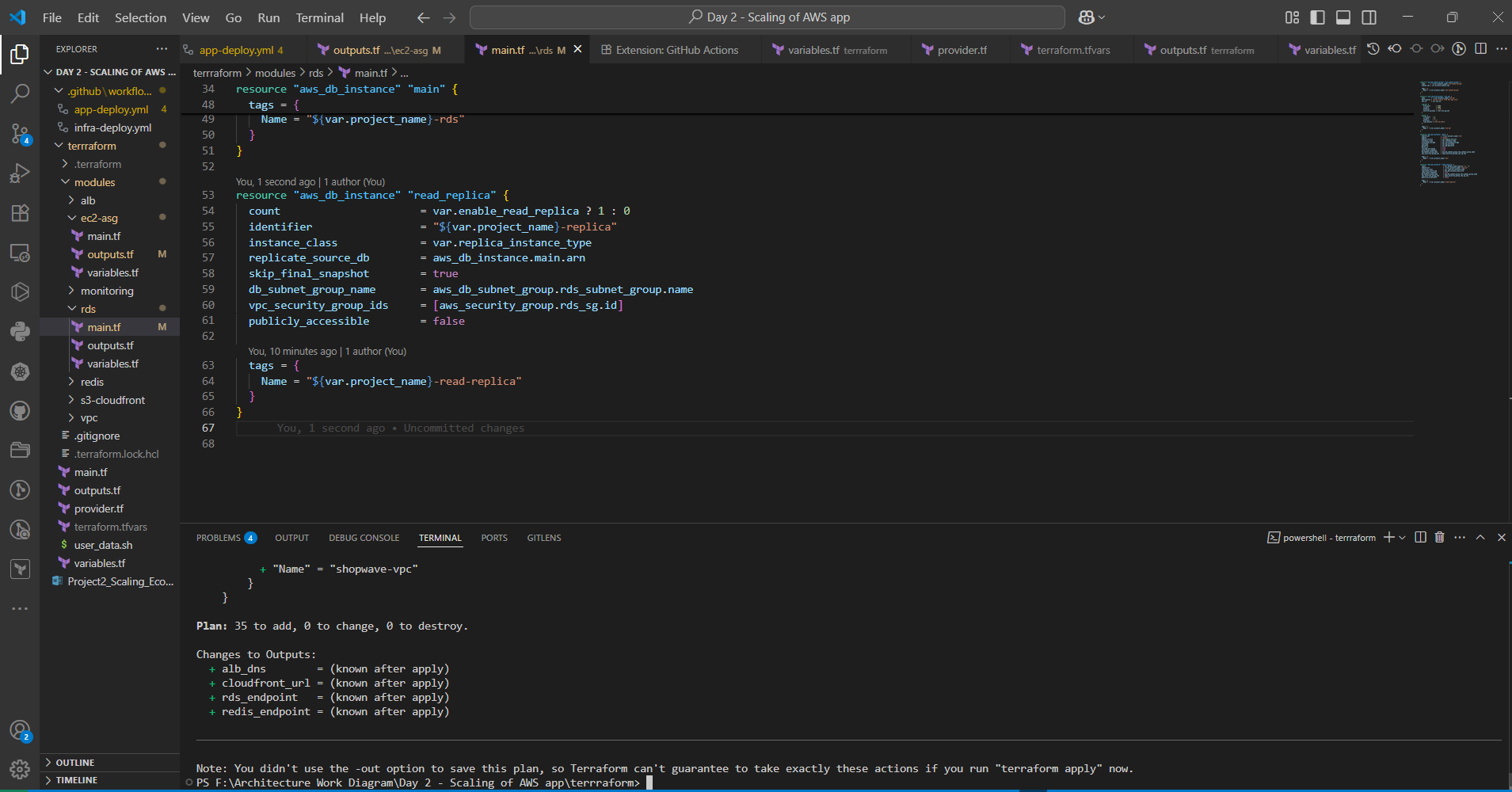
- Architecture Diagram

- Terraform Codebase

- CI/CD Script (GitHub Actions)

- Screen shots

# 9. Terraform Codebase and Screen shots



A screenshot of a computer

AI-generated content may be incorrect.