E-Commerce Microservices DevOps Pipeline

Automating Deployment Across Multiple Environments

Project Requirements

- **Multi-Environment Support**: QA, UAT, and Production
- **Automated Deployments**: CI/CD pipeline for all environments
- **Zero-Downtime Deployments**: Blue-green deployment strategy in Kubernetes...
- **Version Control**: Tracking and promoting versions between environments
- **Infrastructure as Code**: Define infrastructure in code
- **Containerization**: Package applications in containers
- **Automated Testing**: Smoke tests to verify deployments
- **Monitoring**: Track application health and performance

Initial Ambitious Plan

- **Terraform** for all infrastructure provisioning
- **Packer** for building custom AMIs
- **Docker** for containerization of all services
- **Ansible** for configuration management
- **EKS** for Kubernetes orchestration
- **RDS** with automated backups and failover
- **Email Notifications** for deployment events
- **Slack Integration** for alerts and approvals
- **Automated Security Scanning** for vulnerabilities

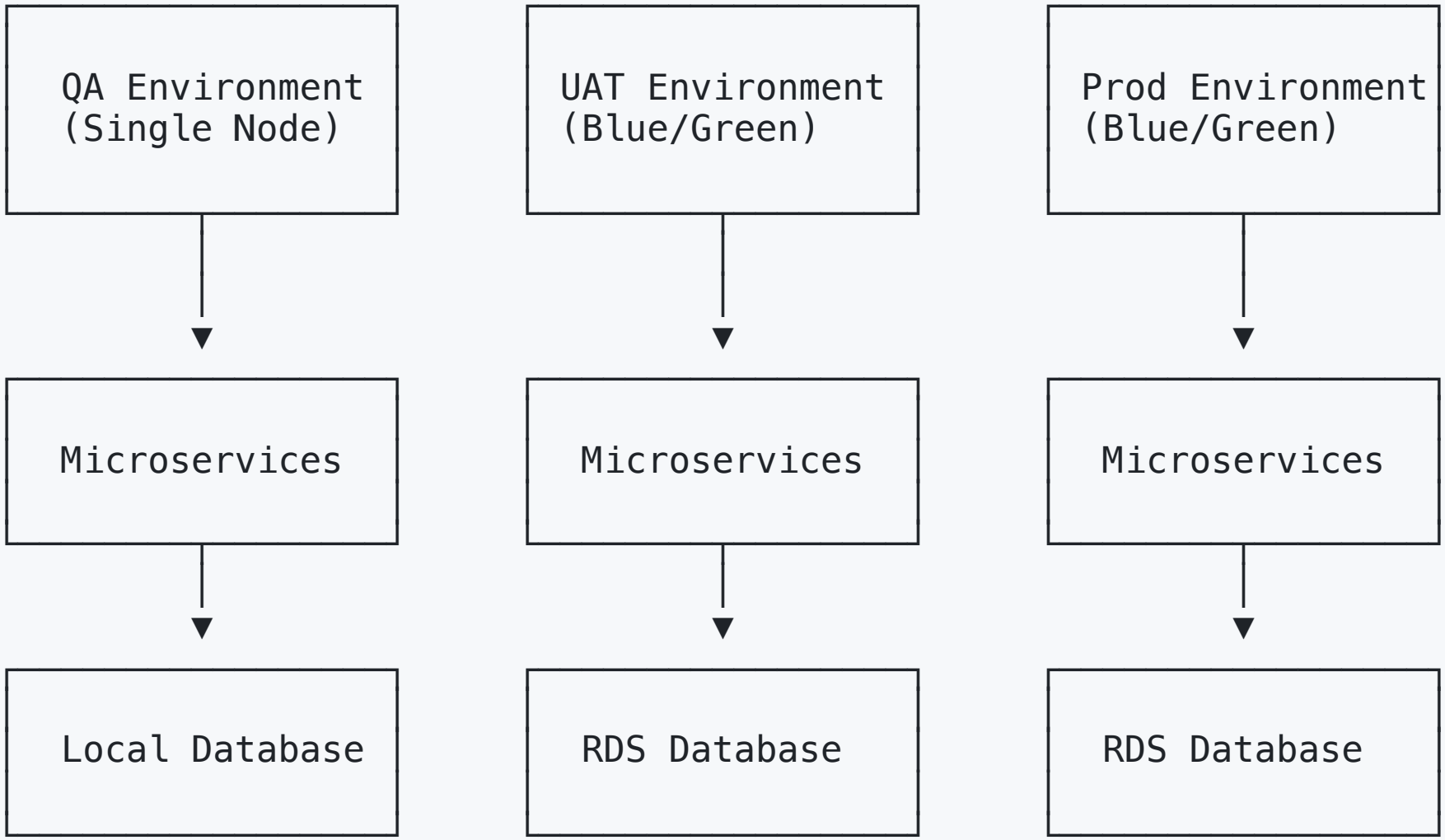
Initial Architecture Vision

- **Kubernetes (EKS)** for container orchestration
- **Terraform** for infrastructure provisioning
- **AWS Managed Services** for databases and networking
- **Prometheus & Grafana** for monitoring
- **GitHub Actions** for CI/CD pipelines

Pragmatic Implementation

- **Terraform** for infrastructure provisioning
- **RDS** for managed database services
- **EC2** for compute resources
- **Docker** for containerization
- **ECR** for image registry
- **GitHub Actions** for CI/CD automation
- **DNS-level Blue-Green Deployments** with Route53
- **Docker Swarm** for container orchestration
- **Automated Smoke Tests** for deployment verification

System Architecture



CI/CD Pipeline

1. **Code Changes** trigger GitHub Actions workflow
2. **Build & Test** containerized applications
3. **Push Images** to container registry
4. **Update Version File** with new version numbers
5. **Deploy to QA** for initial testing
6. **Promote to UAT** after QA approval
7. **Promote to Production** after UAT validation
8. **Rollback** at any time in UAT or Prod if necessary

Blue-Green Deployment Strategy

- **Two identical environments**: Blue and Green
- **DNS-level switching** using Route53 weighted routing
- **Zero-downtime deployments**:
  - Deploy to inactive environment
  - Run smoke tests to verify deployment
  - Switch traffic gradually to new environment
  - Keep old environment as fallback

Version Management

```
"api-gateway": {
  "qa": "1.0.11",
  "uat": "1.0.10",
  "prod": "1.0.9",
  "latest": "1.0.11",
  "timestamp": "2025-05-10T08:21:04Z",
},
"other microservices..."
```

- **Single source of truth** for all environment versions
- **Automated promotion** between environments
- **Git-based version control** for audit trail
- **Code review** required to promote

GitHub Actions Workflows

- **Version Watcher**: Detects changes to version.json and handles deployment accordingly
- **Nightly Build**: Runs daily, detecting changes to a microservice and rebuilding accordingly
- **UAT/Prod Swap**: Handles blue-green rollback

```yaml
on:
  push:
    branches:
      - main
    paths:
      - "version.json"
```

Database Management

- **QA**: Local MySQL container with initialization scripts
- **UAT/Prod**: Amazon RDS with automated backups
- **Migration Strategy**:
  - Schema changes tested in QA
  - Applied to UAT/Prod during deployment
- **Connection Management**: Credentials stored securely

Smoke Testing

```
# Test the API Gateway
echo "Testing API Gateway..."
curl -s "http://${IP_ADDRESS}:8080/api-gateway/health" | grep "ok" || exit 1

# Test the Frontend
echo "Testing Frontend..."
curl -s "http://${IP_ADDRESS}:8081/health" | grep "ok" || exit 1

# Test the Backend
echo "Testing Backend..."
curl -s "http://${IP_ADDRESS}:8082/health" | grep "ok" || exit 1

echo "All tests passed!"
exit 0
```

Lessons Learned

- **Start Simple, Scale Later**: Begin with minimal viable infrastructure
- **Focus on Business Value**: Prioritize working automation over perfect architecture
- **Automate Everything**: Even simple scripts save time and reduce errors
- **Test Thoroughly**: Automated testing is critical for reliable deployments
- **Document As You Go**: Documentation is essential for knowledge transfer

Future Enhancements

- **Kubernetes Migration**: Move to EKS for better scaling
- **Terraform Expansion**: More infrastructure as code
- **Enhanced Monitoring**: Add Prometheus and Grafana

Thank You!