LIVE IMPLEMENTATION COURSE

cubensquare
THINK . CODE . DELIVER

AWS & Docker Grocery Project

2-Week Live Implementation — Real-world Cloud Deployment

2-Week Live | Docker + AWS | Grocery App



HANDS-ON • PROJECT-BASED • LIVE

cubensquare
THINK . CODE . DELIVER

What is This Course?

A 2-week hands-on program where learners deploy a real **Grocery App (v2.1)** on AWS using Docker.

You'll go from container creation to a production go-live with CI/CD, scaling, SSL, monitoring, and a managed database.

Build confidence with real infra and proven workflows.

Course Roadmap

1. Overview & Setup
2. Tools & Tech
3. Week 1 Containerization
4. Week 2 Cloud Deployment
5. Final Architecture

Tools & Technologies Used

cubensquare
THINK . CODE . DELIVER

Docker
containerization engine

AWS EC2
virtual servers for app runtime

Amazon ECR
private image registry

IAM
secure access control

Elastic Load Balancer
traffic distribution

CloudWatch
monitoring & alerts

Linux Ubuntu 22.04
base OS

GitHub Actions
CI/CD pipeline

AWS CLI
command-line access



AWS Managed Services for Production

cubensquare
THINK . CODE . DELIVER

Route 53
DNS routing & health checks for robust domain management and failover

AWS Certificate Manager (ACM)
Automated SSL/TLS provisioning and auto-renewal for secure in-transit encryption

Amazon RDS
MySQL 8.0 managed database with Multi-AZ high availability and automated backups

Amazon S3
Scalable object storage for static assets, media files, and system backup storage



Week 1 — Containerization

Days 1–7 · Docker Setup, App Packaging & Local Validation

- Dockerfile creation
- Multi-stage builds
- Image scanning with Trivy
- Container local testing
- EC2 instance setup
- ECR image push

PROGRESS: WEEK 1 OF 2



cubensquare
THINK . CODE . DELIVER

Week 1 Deep Dive: Docker

cubensquare
THINK . CODE . DELIVER

Docker Setup & Containerization

1. **Write Dockerfile**
   Define the base image, dependencies, and app runtime environment.

2. **Build Image**
   Package the application using `docker build`.

3. **Run Locally**
   Test the containerized app at `localhost:8080`.

4. **Multi-stage Optimization**
   Achieve a **68% image size reduction** for faster deployments.

```
# 1. Build the production image
$ docker build -t grocery-app:2.1 .

# 2. Run container with local env vars
$ docker run -d -p 8080:8080 \
  --name grocery-app \
  --env-file .env.local \
  grocery-app:2.1

# 3. Verify image size reduction
$ docker image ls grocery-app
$ docker history grocery-app:2.1
```



Week 1 Deep Dive: EC2 & ECR

cubensquare
THINK . CODE . DELIVER

EC2 Instance Setup

- Launch **t3.medium** in us-east-1, running Ubuntu 22.04 LTS.
- Configure Security Group with strict inbound rules (Ports 80/443/22).
- Attach IAM instance role with least-privilege policies for ECR and CloudWatch access.
- Provide **User Data script** to automatically install Docker and start the service on boot.

Amazon ECR Push

- Create an Amazon ECR private repository to securely host Docker images.
- Implement a lifecycle policy to automatically retain only the **last 10 images**.
- Authenticate local Docker client using the AWS CLI credentials.
- Tag the grocery app image and push securely to ECR, verifying the **SHA256 digest**.

Image successfully pushed to ECR



Week 2 — Cloud Deployment

Days 8–14 · ECR Integration, EC2 Deploy & Production Observability

- GitHub Actions CI/CD
- Elastic Load Balancer
- Auto Scaling Group
- CloudWatch Alarms
- Route 53 + ACM SSL
- RDS & S3 Integration

PROGRESS: WEEK 2 OF 2

cubensquare
THINK . CODE . DELIVER

CI/CD Pipeline with GitHub Actions

cubensquare
THINK . CODE . DELIVER

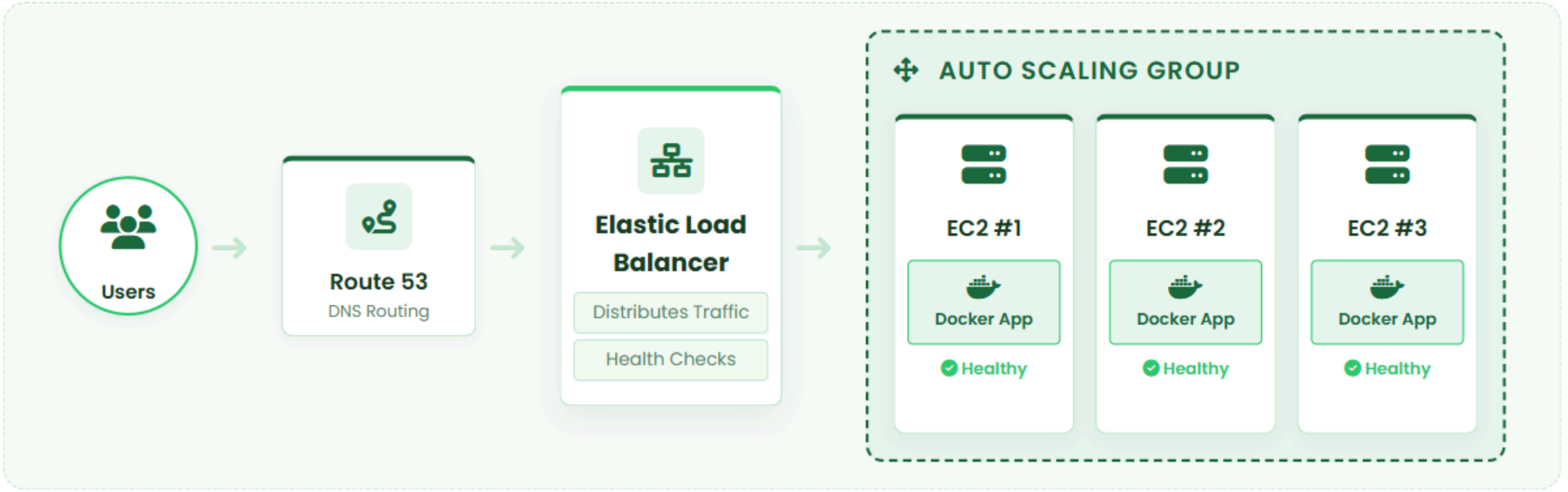
Code Commit → CI Trigger → Docker Build + Trivy Scan → Push to Amazon ECR → Deploy to AWS EC2 → ELB Health Check → Live at grocery-app-prod

Full automation from code push to live production



Elastic Load Balancer + Auto Scaling

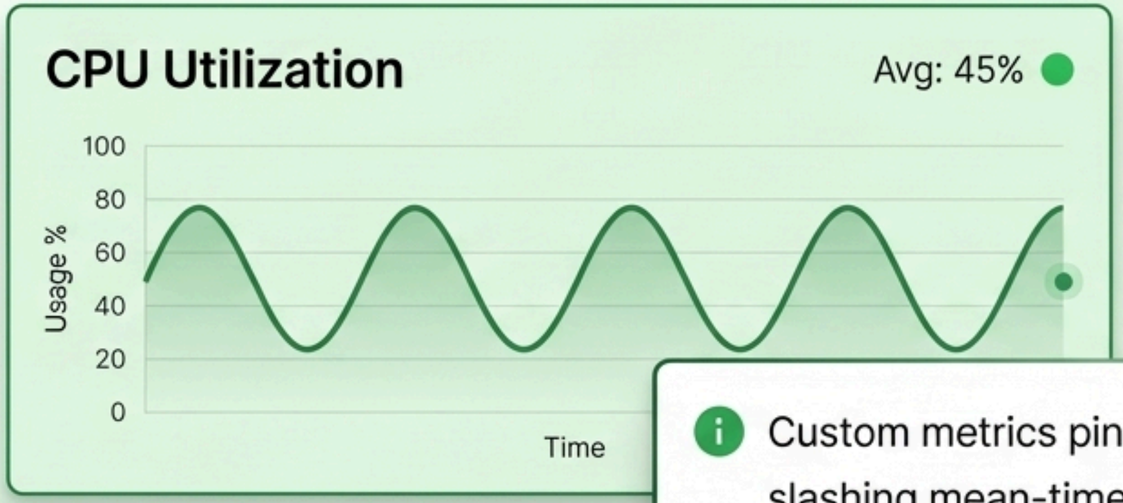
cubensquare
THINK . CODE . DELIVER

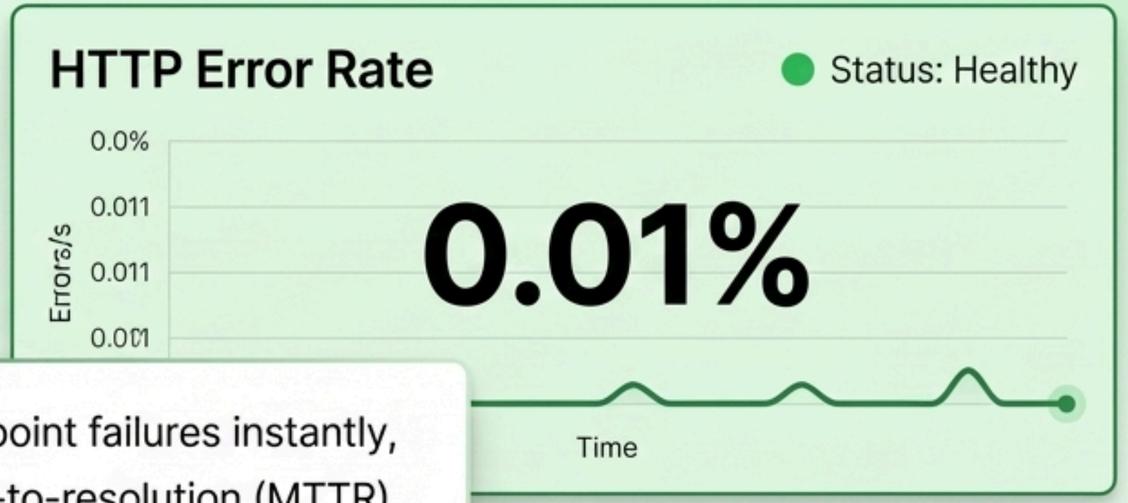


100% traffic routing to healthy containers only via ELB health checks

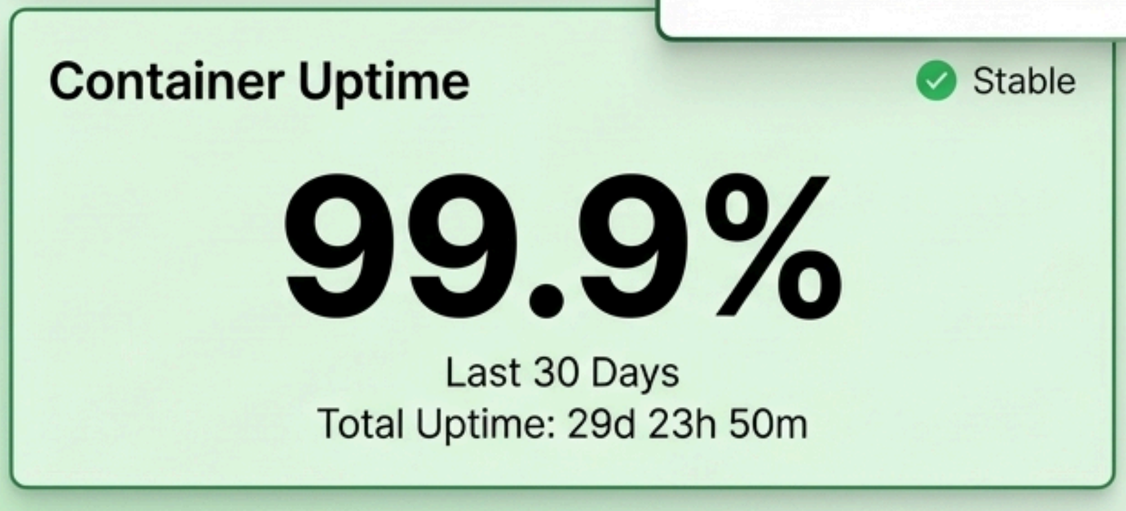
Auto Scaling rules dynamically adjust capacity based on CPU & request load

Real-time Telemetry: Achieving root-cause analysis in under 3 minutes.





Custom metrics pinpoint failures instantly, slashing mean-time-to-resolution (MTTR).





cubensquare
THINK . CODE . DELIVER

Production-Grade Enhancements

cubensquare
THINK . CODE . DELIVER

Route 53

Domain grocery-app-prod.us with latency-based routing for optimal global performance

ACM SSL

Auto certificate provisioning and renewal for secure HTTPS connections

RDS MySQL 8.0

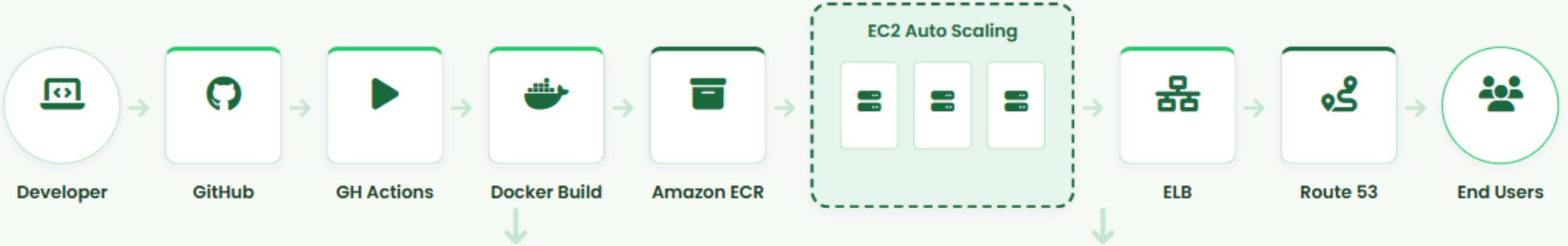
Multi-AZ deployment, encrypted at rest, with automated snapshot backups

Amazon S3 + CloudFront

Static assets delivery with OAI-protected CDN for high speed and security

End-to-End Production Architecture

cubensquare
THINK . CODE . DELIVER



Amazon RDS

MySQL 8.0
Multi-AZ Deployment

Amazon S3

Static Assets Storage
CloudFront Integration

Lessons from a Real 2-Week AWS Deployment

cubensquare
THINK . CODE . DELIVER

1. **Docker multi-stage builds cut image size by 68%**
   Optimized packaging accelerated Amazon ECR pushes and EC2 deployment times significantly.

2. **EC2 Launch Type gave full control for container debugging**
   Direct SSH access and environment inspection allowed rapid troubleshooting during the initial setup phase.

3. **ELB health checks prevented all traffic to unhealthy containers**
   Ensured 100% availability by automatically routing users only to healthy and responsive EC2 instances.

4. **CloudWatch custom metrics enabled root-cause in under 3 minutes**
   Real-time dashboards provided immediate, actionable visibility into CPU spikes and application latency.

5. **Separation of concerns (S3 + RDS + EC2) enabled independent scaling**
   Decoupling static assets, the managed database, and compute instances ensured a robust production architecture.

What You Will Achieve

cubensquare
THINK . CODE . DELIVER

Deploy a production Dockerized app on AWS EC2

Stand up resilient, scalable compute behind an Elastic Load Balancer with health checks and Auto Scaling.

Build an end-to-end CI/CD pipeline with GitHub Actions

Automate code build, vulnerability scan with Trivy, push to Amazon ECR, and safe deployment to EC2.

Monitor and debug with CloudWatch metrics

Track SLOs, set alarms for CPU and latency, and establish real-time observability to troubleshoot fast.

Grocery App v2.1 — Successfully Live on AWS.

Day 14 — Full Production Cutover Complete



cubensquare
THINK . CODE . DELIVER

Docker | AWS EC2 | GitHub Actions | CloudWatch | Ubuntu

Enroll Now in the 2-Week Live Batch

Secure your spot. Build it yourself.