Operations Manual

Naeem Dosh (Fiverr)

February 3, 2026

Contents

Operations Documentation

Daily Operations

Health Checks

1. **Application Status**

   ```
   curl -s -o /dev/null -w "%{http_code}" https://taxplanner.app/
   # Expected: 200
   ```

2. **ALB Health**

   - Check AWS Console | EC2 | Target Groups | hipaa-poc-tg
   - All targets should be "healthy"

3. **CloudWatch Logs**

   ```
   aws logs tail /hipaa-poc/application --since 1h --region us-east-2
   ```

Connecting to EC2

Via SSM Session Manager

```
# From AWS CLI
aws ssm start-session --target i-0500bfb3b4ad44e24 --region us-east-2

# Once connected
sudo su -
cd /app
docker compose ps
```

Via AWS Console

1. Go to EC2 | Instances
2. Select the instance
3. Click "Connect" | "Session Manager" | "Connect"

Application Management

View Container Status

```
cd /app
docker compose ps
```

View Application Logs

```
docker compose logs -f php
docker compose logs -f nginx
```

Restart Application

```
cd /app
docker compose restart
```

Update Application

```
cd /app
git pull origin main
docker compose down
docker compose up -d --build
```

View Database

```
sqlite3 /data/db/app.sqlite
.tables
SELECT * FROM users;
SELECT * FROM messages ORDER BY created_at DESC LIMIT 10;
SELECT * FROM audit_log ORDER BY created_at DESC LIMIT 20;
.quit
```

Backup Operations

Manual Backup

```
/usr/local/bin/backup-db.sh
```

List Backups

```
aws s3 ls s3://hipaa-poc-backups-730543776652/backups/ --region us-east-2
```

Download Backup

```
aws s3 cp s3://hipaa-poc-backups-730543776652/backups/db_20260131_020000.sqlite.gz /tmp/ --reg
gunzip /tmp/db_20260131_020000.sqlite.gz
```

Restore from Backup

```
# Stop application
cd /app
```

```
docker compose down

# Backup current database
cp /data/db/app.sqlite /data/db/app.sqlite.old

# Restore from backup
aws s3 cp s3://hipaa-poc-backups-730543776652/backups/db_YYYYMMDD_HHMMSS.sqlite.gz /tmp/
gunzip /tmp/db_YYYYMMDD_HHMMSS.sqlite.gz
cp /tmp/db_YYYYMMDD_HHMMSS.sqlite /data/db/app.sqlite
chown 1000:1000 /data/db/app.sqlite

# Start application
docker compose up -d
```

Secrets Management

View Current Secrets

```
aws secretsmanager get-secret-value \
  --secret-id hipaa-poc/app-secrets \
  --region us-east-2 \
  --query SecretString \
  --output text | jq .
```

Update Secrets

1. Update in AWS Secrets Manager:

   ```
   aws secretsmanager update-secret \
     --secret-id hipaa-poc/app-secrets \
     --secret-string '{"GOOGLE_CLIENT_ID":"new-id","GOOGLE_CLIENT_SECRET":"new-secret",...}'
     --region us-east-2
   ```

2. Restart EC2 or re-deploy to load new secrets:

   ```
   # On EC2
   cd /app
   # Re-fetch secrets and update .env
   SECRET_JSON=$(aws secretsmanager get-secret-value --secret-id hipaa-poc/app-secrets --regi
   # Update .env file accordingly
   docker compose restart
   ```

Terraform Operations

Initialize (New Machine)

```
cd terraform
terraform init
```

Plan Changes

```
terraform plan
```

Apply Changes

```
terraform apply
```

View Current State

```
terraform show
terraform output
```

Import Existing Resource

```
terraform import aws_instance.app i-0500bfb3b4ad44e24
```

Monitoring

CloudWatch Metrics

Key metrics to monitor: - EC2: CPUUtilization, StatusCheckFailed - ALB: HealthyHostCount, RequestCount, TargetResponseTime - NAT Gateway: BytesOutToDestination

CloudWatch Alarms (Recommended)

```
# Example: ALB unhealthy hosts alarm
aws cloudwatch put-metric-alarm \
  --alarm-name hipaa-poc-unhealthy-hosts \
  --metric-name UnHealthyHostCount \
  --namespace AWS/ApplicationELB \
  --statistic Average \
  --period 60 \
  --threshold 1 \
  --comparison-operator GreaterThanOrEqualToThreshold \
  --evaluation-periods 2 \
  --dimensions Name=TargetGroup,Value=targetgroup/hipaa-poc-tg/xxx \
  --alarm-actions arn:aws:sns:us-east-2:730543776652:alerts \
  --region us-east-2
```

Troubleshooting

Application Not Loading

1. Check EC2 status:

   ```
   aws ec2 describe-instance-status --instance-ids i-0500bfb3b4ad44e24 --region us-east-2
   ```

2. Check target group health:

   ```
   aws elbv2 describe-target-health --target-group-arn <tg-arn> --region us-east-2
   ```

3. Connect via SSM and check containers:

   ```
   docker compose ps
   docker compose logs
   ```

OAuth Login Failed

1. Check redirect URI matches in Google Console
2. Verify secrets in Secrets Manager
3. Check `.env` file on EC2:

   ```
   cat /app/.env
   ```

Database Issues

1. Check database file exists:

   ```
   ls -la /data/db/
   ```

2. Check permissions:

   ```
   ls -la /data/db/app.sqlite
   # Should be owned by 1000:1000
   ```

3. Test database:

   ```
   sqlite3 /data/db/app.sqlite "SELECT COUNT(*) FROM users;"
   ```

SSL Certificate Issues

1. Check certificate status:

   ```
   aws acm describe-certificate --certificate-arn <cert-arn> --region us-east-2
   ```

2. Verify DNS validation record exists
3. Check certificate is attached to ALB listener

Scaling (Future)

Horizontal Scaling

To scale horizontally: 1. Create AMI from current EC2 2. Create Auto Scaling Group 3. Update ALB target group to use ASG 4. Migrate database to RDS or MongoDB Atlas

Vertical Scaling

To increase instance size: 1. Update `instance_type` in `terraform.tfvars` 2. Run `terraform apply` 3. Instance will be replaced (brief downtime)

Disaster Recovery

RTO/RPO

Metric	Target
RTO (Recovery Time Objective)	1 hour
RPO (Recovery Point Objective)	24 hours (daily backups)

Recovery Procedure

1. **If EC2 fails:**
   - Terraform will recreate from scratch
   - Restore database from S3 backup
2. **If region fails:**
   - Deploy to new region using Terraform
   - Update DNS to new ALB
   - Restore database from S3 (cross-region replication recommended for production)

Full Recovery Steps

```
# 1. Deploy infrastructure
cd terraform
terraform init
terraform apply

# 2. Wait for EC2 to initialize (check user-data logs)
aws ssm start-session --target <new-instance-id> --region us-east-2
tail -f /var/log/user-data.log

# 3. Restore database from backup
/usr/local/bin/restore-db.sh <backup-file>

# 4. Verify application
curl https://taxplanner.app/
```

Maintenance Windows

Recommended Schedule

Task	Frequency	Window
OS Updates	Monthly	Sunday 2-4 AM
Application Updates	As needed	Off-peak hours
Backup Verification	Weekly	N/A
Security Review	Monthly	N/A

Applying OS Updates

```
# Connect to EC2
aws ssm start-session --target i-0500bfb3b4ad44e24 --region us-east-2

# Update packages
sudo dnf update -y

# Reboot if kernel updated
sudo reboot
```

Contacts

Role	Contact
DevOps Engineer	Naeem
Project Owner	appcropolis