Troubleshooting Guide

Comprehensive troubleshooting guide for common issues in the Course Platform deployment.

General Troubleshooting Approach

1. **Check Service Status**
2. **Review Logs**
3. **Verify Configuration**
4. **Test Connectivity**
5. **Check Resources**

Docker Issues

Container Won't Start

Symptoms:
- Container exits immediately
- "Exited (1)" status
- Application not accessible

Diagnosis:

```
# Check container status
docker-compose -f docker-compose.dev.yml ps

# View container logs
docker-compose -f docker-compose.dev.yml logs app-dev

# Check container details
docker inspect course-platform-app-dev
```

Common Solutions:

1. **Environment Variables Missing**
   ```bash
   # Check if .env file exists
   ls -la .env*

# Verify environment variables
docker-compose -f docker-compose.dev.yml config
```

1. **Port Conflicts**
   ```bash
   # Check what's using the port
   sudo netstat -tulpn | grep :3000
   sudo lsof -i :3000
   ```

~~~~
# Kill conflicting process
sudo kill -9
```
~~~~

1. **Permission Issues**

   ```
   bash
     # Fix file permissions
     sudo chown -R $USER:$USER .
     chmod +x scripts/*.sh
   ```

2. **Build Issues**

   ```
   bash
     # Rebuild containers
     docker-compose -f docker-compose.dev.yml build --no-cache
     docker-compose -f docker-compose.dev.yml up -d
   ```

Database Connection Issues

Symptoms:
- “Connection refused” errors
- “Database does not exist” errors
- Prisma connection timeouts

Diagnosis:

```bash
# Check database container
docker-compose -f docker-compose.dev.yml logs postgres-dev

# Test database connection
docker-compose -f docker-compose.dev.yml exec postgres-dev psql -U dev_user -d course_shop_platform_dev -c "SELECT 1;"
```

Solutions:

1. **Database Not Ready**

   ```
   bash
     # Wait for database to be ready
     docker-compose -f docker-compose.dev.yml up -d postgres-dev
     sleep 30
     docker-compose -f docker-compose.dev.yml up -d app-dev
   ```

2. **Wrong Database URL**

   ```
   bash
     # Check DATABASE_URL format
     echo $DATABASE_URL
     # Should be: postgresql://user:password@host:port/database
   ```

3. **Database Doesn’t Exist**

   ~~~~
   ```bash
   # Create database manually
   docker-compose -f docker-compose.dev.yml exec postgres-dev createdb -U dev_user course_shop_platform_dev
   ~~~~

# Run migrations
./scripts/db_migrate.sh dev
```

## Docker Compose Issues

**Symptoms:**
- “Service not found” errors
- “Network not found” errors
- Services can’t communicate

**Diagnosis:**

```
# Validate compose file
docker-compose -f docker-compose.dev.yml config

# Check networks
docker network ls
docker network inspect course-platform-dev

Solutions:

1. **Invalid Compose File**
   ```bash
   # Validate YAML syntax
   python3 -c “import yaml; yaml.safe_load(open(‘docker-compose.dev.yml’))”

# Check compose file version
docker-compose version
```

1. **Network Issues**

```
bash
    # Remove and recreate network
    docker-compose -f docker-compose.dev.yml down
    docker network prune -f
    docker-compose -f docker-compose.dev.yml up -d
```

Nginx Issues

Nginx Won’t Start

Symptoms:
- “nginx: [emerg]” errors
- 502 Bad Gateway
- Connection refused

Diagnosis:

```
# Test nginx configuration
sudo nginx -t

# Check nginx status
sudo systemctl status nginx

# View nginx logs
sudo tail -f /var/log/nginx/error.log
```

Solutions:

1. **Configuration Syntax Error**
   ```bash
   # Test configuration
   sudo nginx -t

# Fix syntax errors in config files
sudo nano /etc/nginx/sites-available/default
```

1. **Port Already in Use**
   ```bash
   # Check what's using port 80/443
   sudo netstat -tulpn | grep :80
   sudo netstat -tulpn | grep :443

# Stop conflicting service
sudo systemctl stop apache2 # if Apache is running
```

1. **SSL Certificate Issues**
   ```bash
   # Check certificate files
   sudo ls -la /etc/nginx/ssl/

# Verify certificate
sudo openssl x509 -in /etc/nginx/ssl/example.com.crt -text -noout
```

502 Bad Gateway

Symptoms:
- Nginx returns 502 error
- Backend service unreachable

Diagnosis:

```
# Check if backend is running
curl -I http://localhost:3000/health

# Check nginx upstream configuration
sudo nginx -T | grep upstream

# Check nginx error logs
sudo tail -f /var/log/nginx/error.log
```

Solutions:

1. **Backend Service Down**
   ```bash
   # Start backend service
   docker-compose -f docker-compose.dev.yml up -d app-dev

# Check service health
docker-compose -f docker-compose.dev.yml exec app-dev curl http://localhost:3000/health
```

1. **Wrong Upstream Configuration**
   ```bash
   # Check nginx configuration
   grep -r “proxy_pass” /etc/nginx/

# Verify backend is accessible
curl http://app-dev:3000/health # from nginx container
```

SSL Certificate Issues

Symptoms:
- “SSL certificate problem” errors
- Browser security warnings
- Certificate expired errors

Diagnosis:

```
# Check certificate status
sudo certbot certificates

# Test SSL connection
openssl s_client -connect example.com:443 -servername example.com

# Check certificate expiry
echo | openssl s_client -servername example.com -connect example.com:443 2>/dev/null |
openssl x509 -noout -dates
```

Solutions:

1. **Certificate Expired**
   ```bash
   # Renew certificate
   sudo certbot renew

# Force renewal
sudo certbot renew –force-renewal
```

1. **Certificate Not Found**
   `bash`
   `   # Generate new certificate`
   `   sudo certbot --nginx -d example.com -d www.example.com`

2. **Wrong Certificate Path**
   ```bash
   # Check nginx SSL configuration
   sudo grep -r “ssl_certificate” /etc/nginx/

# Update paths in nginx config
sudo nano /etc/nginx/sites-available/default
```

Database Issues

PostgreSQL Connection Problems

Symptoms:
- “Connection refused” errors
- “Authentication failed” errors
- “Database does not exist” errors

Diagnosis:

```
# Check PostgreSQL status
docker-compose -f docker-compose.dev.yml logs postgres-dev

# Test connection
docker-compose -f docker-compose.dev.yml exec postgres-dev pg_isready -U dev_user

# Check database exists
docker-compose -f docker-compose.dev.yml exec postgres-dev psql -U dev_user -l
```

Solutions:

1. **Wrong Credentials**
   ```bash
   # Check environment variables
   grep DATABASE_URL .env.dev

# Reset password
docker-compose -f docker-compose.dev.yml exec postgres-dev psql -U postgres -c “ALTER USER dev_user PASSWORD ‘new_password’;”
```

1. **Database Doesn’t Exist**
   ```bash
   # Create database
   docker-compose -f docker-compose.dev.yml exec postgres-dev createdb -U dev_user course_shop_platform_dev

# Run migrations
./scripts/db_migrate.sh dev
```

1. **Connection Limit Reached**
   ```bash
   # Check active connections
   ```

docker-compose -f docker-compose.dev.yml exec postgres-dev psql -U dev_user -c “SELECT count(*) FROM pg_stat_activity;”

# Kill idle connections
docker-compose -f docker-compose.dev.yml exec postgres-dev psql -U dev_user -c “SELECT pg_terminate_backend(pid) FROM pg_stat_activity WHERE state = ‘idle’;”
```

## Prisma Migration Issues

**Symptoms:**
- “Migration failed” errors
- “Schema drift” warnings
- “Database out of sync” errors

**Diagnosis:**

```
# Check migration status
./scripts/db_migrate.sh dev status

# View migration history
docker-compose -f docker-compose.dev.yml exec postgres-dev psql -U dev_user -d course_s
hop_platform_dev -c "SELECT * FROM _prisma_migrations;"

Solutions:

1. **Failed Migration**
   ```bash
   # Reset database (development only)
   ./scripts/db_migrate.sh dev reset

# Or manually fix and retry
./scripts/db_migrate.sh dev deploy
```

1. **Schema Drift**
   ```bash
   # Generate new migration
   cd app
   npx prisma migrate dev –name fix_schema_drift

# Deploy migration
./scripts/db_migrate.sh dev deploy
```

Application Issues

Next.js Build Failures

Symptoms:
- Build process fails
- “Module not found” errors
- TypeScript compilation errors

Diagnosis:

```
# Check build logs
docker-compose -f docker-compose.dev.yml logs app-dev

# Try building locally
cd app
npm install
npm run build
```

Solutions:

1. **Missing Dependencies**
   ```bash
   # Install dependencies
   cd app
   npm install

# Clear cache and reinstall
rm -rf node_modules package-lock.json
npm install
```

1. **TypeScript Errors**
   ```bash
   # Check TypeScript errors
   cd app
   npx tsc –noEmit

# Fix type errors or add type ignores
```

1. **Environment Variables**

   ```
   bash
      # Check required environment variables
      cd app
      node -e "console.log(process.env.DATABASE_URL)"
   ```

API Route Issues

Symptoms:
- 500 Internal Server Error
- API endpoints not responding
- CORS errors

Diagnosis:

```
# Check application logs
docker-compose -f docker-compose.dev.yml logs app-dev

# Test API endpoint directly
curl -X GET http://localhost:3000/api/health

# Check network connectivity
docker-compose -f docker-compose.dev.yml exec app-dev curl http://postgres-dev:5432
```

Solutions:

1. **Database Connection in API**
   ```bash
   # Check database connection
   docker-compose -f docker-compose.dev.yml exec app-dev npx prisma db pull

# Test Prisma client
docker-compose -f docker-compose.dev.yml exec app-dev node -e “const { PrismaClient } = require(‘@prisma/client’); const prisma = new PrismaClient(); prisma.\$connect().then(() => console.log(‘Connected’)).catch(console.error);”
```

1. **CORS Issues**
   `bash`
   `  # Check CORS headers`
   `  curl -H "Origin: http://localhost:3000" -H "Access-Control-Request-Method: POST" -X OPTIONS http://localhost:3000/api/courses`

Deployment Issues

Deployment Script Failures

Symptoms:
- Deployment script exits with error
- rsync failures
- SSH connection issues

Diagnosis:

```
# Run deployment with verbose output
./scripts/deploy.sh dev --dry-run

# Test SSH connection
ssh ubuntu@dev.example.com "echo 'SSH working'"

# Test rsync
rsync -avz --dry-run . ubuntu@dev.example.com:/tmp/test/
```

Solutions:

1. **SSH Key Issues**
   ```bash
   # Check SSH key
   ssh-add -l

# Add SSH key
ssh-add ~/.ssh/id_rsa
   ```

~~~
# Test SSH connection
ssh -v ubuntu@dev.example.com
```
~~~

1. **Permission Issues**

   ~~~
   ```bash
   # Fix script permissions
   chmod +x scripts/*.sh
   ~~~

~~~
# Fix remote directory permissions
ssh ubuntu@dev.example.com “sudo chown -R ubuntu:ubuntu /var/www/course-platform”
```
~~~

1. **Network Issues**

   ~~~
   ```bash
   # Test connectivity
   ping dev.example.com
   ~~~

~~~
# Check DNS resolution
nslookup dev.example.com
```
~~~

Docker Registry Issues

Symptoms:
- “Image not found” errors
- Push/pull failures
- Authentication errors

Diagnosis:

```
# Check Docker login
docker info

# Test registry connectivity
docker pull hello-world

# Check image exists
docker images | grep course-platform
```

Solutions:

1. **Authentication Issues**

   ~~~
   ```bash
   # Login to registry
   docker login ghcr.io
   ~~~

~~~
# Check credentials
cat ~/.docker/config.json
```
~~~

1. **Image Not Found**

   ```
   bash
      # Build and push image
   ```

```
docker build -t ghcr.io/username/course-platform:latest .
docker push ghcr.io/username/course-platform:latest
```

Performance Issues

Slow Application Response

Symptoms:
- High response times
- Timeouts
- Poor user experience

Diagnosis:

```
# Check resource usage
docker stats

# Monitor application logs
docker-compose -f docker-compose.dev.yml logs -f app-dev

# Test response times
curl -w "@curl-format.txt" -o /dev/null -s http://localhost:3000/
```

Solutions:

1. **Resource Constraints**
   ```bash
   # Check available resources
   free -h
   df -h

# Increase container resources
# Edit docker-compose.yml to add resource limits
```

1. **Database Performance**
   ```bash
   # Check slow queries
   docker-compose -f docker-compose.dev.yml exec postgres-dev psql -U dev_user -c “SELECT query, mean_time, calls FROM pg_stat_statements ORDER BY mean_time DESC LIMIT 10;”

# Analyze query performance
docker-compose -f docker-compose.dev.yml exec postgres-dev psql -U dev_user -c “EXPLAIN ANALYZE SELECT * FROM courses LIMIT 10;”
```

High Memory Usage

Symptoms:
- Out of memory errors
- Container restarts
- System slowdown

Diagnosis:

```bash
# Check memory usage
free -h
docker stats --no-stream

# Check for memory leaks
docker-compose -f docker-compose.dev.yml exec app-dev node --expose-gc -e
"global.gc(); console.log(process.memoryUsage());"
```

Solutions:

1. **Increase Memory Limits**

```
yaml
  # In docker-compose.yml
  services:
    app-dev:
      deploy:
        resources:
          limits:
            memory: 2G
```

2. **Optimize Application**

```
bash
  # Check for memory leaks in Node.js
  cd app
  npm install --save-dev clinic
  npx clinic doctor -- node server.js
```

System Issues

Disk Space Issues

Symptoms:
- "No space left on device" errors
- Application crashes
- Backup failures

Diagnosis:

```bash
# Check disk usage
df -h
du -sh /var/www/course-platform/*

# Check Docker disk usage
docker system df
```

Solutions:

1. **Clean Up Docker**
   ```bash
   # Remove unused containers, networks, images
   docker system prune -a -f
   ```

```
# Remove old images
docker image prune -a -f

# Remove unused volumes
docker volume prune -f
```

1. **Clean Up Logs**
```bash
# Rotate logs
sudo logrotate -f /etc/logrotate.conf

# Clean old logs
find /var/www/course-platform/logs -name "*.log" -mtime +30 -delete
```

1. **Clean Up Backups**
`bash`
```
   # Remove old backups
   find /var/www/course-platform/backups -name "*.tar.gz" -mtime +30 -delete
```

Network Connectivity Issues

Symptoms:
- Services can’t communicate
- External API calls fail
- DNS resolution issues

Diagnosis:

```
# Test network connectivity
ping google.com
nslookup example.com

# Check Docker networks
docker network ls
docker network inspect course-platform-dev

# Test service connectivity
docker-compose -f docker-compose.dev.yml exec app-dev ping postgres-dev
```

Solutions:

1. **DNS Issues**
```bash
# Check DNS configuration
cat /etc/resolv.conf

# Test DNS resolution
nslookup example.com 8.8.8.8
```

1. **Firewall Issues**
```bash
```

    # Check firewall rules
    sudo ufw status verbose

# Allow specific ports
sudo ufw allow 3000/tcp
```

1. **Docker Network Issues**

   ```
   bash
      # Recreate Docker networks
      docker-compose -f docker-compose.dev.yml down
      docker network prune -f
      docker-compose -f docker-compose.dev.yml up -d
   ```

Troubleshooting Checklist

Quick Diagnosis

- [ ] Check service status: `docker-compose ps`
- [ ] Review recent logs: `docker-compose logs --tail=50`
- [ ] Verify configuration: `docker-compose config`
- [ ] Test connectivity: `curl http://localhost:3000/health`
- [ ] Check resources: `docker stats`

Common Commands

```
# Service management
docker-compose -f docker-compose.dev.yml ps
docker-compose -f docker-compose.dev.yml logs app-dev
docker-compose -f docker-compose.dev.yml restart app-dev
docker-compose -f docker-compose.dev.yml down && docker-compose -f docker-compose.dev.yml up -d

# Database troubleshooting
./scripts/db_migrate.sh dev status
docker-compose -f docker-compose.dev.yml exec postgres-dev psql -U dev_user -d course_shop_platform_dev

# System troubleshooting
sudo systemctl status nginx
sudo nginx -t
sudo tail -f /var/log/nginx/error.log
sudo ufw status verbose

# Resource monitoring
free -h
df -h
docker stats
htop
~~~

## Emergency Recovery

1. **Complete System Reset**
   ```bash
~~~

```
    # Stop all services
    docker-compose -f docker-compose.dev.yml down

# Remove all containers and volumes
docker system prune -a -f –volumes

# Restart from scratch
./scripts/deploy_dev.sh
```

1. **Database Recovery**

```bash
    # Restore from backup
    ./scripts/rollback.sh dev –database-only

# Or reset and migrate
./scripts/db_migrate.sh dev reset
```

1. **Application Recovery**

```bash
    # Rollback to previous version
    ./scripts/rollback.sh dev

# Or redeploy
./scripts/deploy_dev.sh
```

Getting Help

Log Locations

- Application logs: `/var/www/course-platform/logs/`
- Nginx logs: `/var/log/nginx/`
- System logs: `/var/log/syslog`
- Docker logs: `docker-compose logs`

Useful Resources

- Docker Documentation (https://docs.docker.com/)
- Next.js Documentation (https://nextjs.org/docs)
- Prisma Documentation (https://www.prisma.io/docs)
- Nginx Documentation (https://nginx.org/en/docs/)

Support Channels

- Check GitHub Issues
- Review application documentation
- Consult team knowledge base
- Contact system administrator

Remember: Always backup before making significant changes, and test solutions in development before applying to production.