Deploy Spring Boot Application to Lightsail Server

AWS Lightsail Deployment Course

2026-04-08

Table of contents

1 Step 4: Deploy Spring Boot Application

Deploy your Spring Boot application to AWS Lightsail server

- Install Java Runtime Environment
- Transfer application files
- Configure production environment
- Set up system service
- Monitor and troubleshoot

This step covers the complete deployment process from installing Java to monitoring the running application. We'll create a production-ready deployment with proper security and monitoring.

---

2 Understanding the Deployment Process

2.1 Key Components

- Java Runtime Environment (JRE)

- Compiled JAR file transfer
- Production environment variables
- System service configuration

2.2 Best Practice

Always test your application locally with production profiles before deploying

The deployment process involves several interconnected components that work together to create a robust production environment. Testing locally with production profiles helps catch configuration issues early.

---

3 Installing Java on Lightsail

3.1 Connect and Install OpenJDK 17

```
# SSH to your instance
ssh -i your-key.pem ubuntu@your-lightsail-ip

# Update and install Java
sudo apt update
sudo apt install openjdk-17-jdk -y

# Verify installation
java -version
```

OpenJDK 17 is the current LTS version and fully compatible with Spring Boot 3.x. The installation process is straightforward on Ubuntu using the apt package manager.

---

4 Preparing Application Structure

4.1 Create Directory Structure

```
# Create application directory
sudo mkdir -p /opt/myapp

# Create logs directory
sudo mkdir -p /var/log/myapp

# Create application user
sudo useradd -r -s /bin/false myapp

# Set ownership
sudo chown -R myapp:myapp /opt/myapp /var/log/myapp
```

Creating a dedicated user for the application improves security by following the principle of least privilege. The application runs with minimal permissions needed for operation.

---

5 Building and Transferring Application

5.1 Build JAR File Locally

```
# Using Maven
./mvnw clean package -DskipTests

# Using Gradle
./gradlew build -x test
```

5.2 Transfer to Server

```
# Transfer JAR file
scp -i your-key.pem target/myapp-0.0.1-SNAPSHOT.jar \
  ubuntu@your-lightsail-ip:/home/ubuntu/

# Move to application directory
sudo mv /home/ubuntu/myapp-0.0.1-SNAPSHOT.jar /opt/myapp/myapp.jar
```

Building locally ensures you have the exact same environment for building. SCP provides a secure way to transfer files. Alternative methods include rsync, SFTP, or Git-based deployments.

---

6 Configuring Production Environment

6.1 Create Environment File

```
# Create environment configuration
sudo nano /opt/myapp/application.env
```

6.2 Environment Variables

- SPRING_PROFILES_ACTIVE=prod
- Database connection settings
- Server configuration
- JVM memory options
- Logging levels

Environment variables allow you to configure the application for different environments without changing code. This includes database settings, memory allocation, and feature flags.

---

7 Environment Configuration Example

```
# /opt/myapp/application.env
SPRING_PROFILES_ACTIVE=prod

# Database Configuration
DB_HOST=your-lightsail-db-endpoint
DB_PORT=3306
DB_NAME=myapp_prod
DB_USERNAME=myappuser
```

```
DB_PASSWORD=your-secure-password

# JVM Options
JAVA_OPTS=-Xms512m -Xmx1024m -Dspring.profiles.active=prod
```

7.1 Set Secure Permissions

```
sudo chown myapp:myapp /opt/myapp/application.env
sudo chmod 600 /opt/myapp/application.env
```

Setting restrictive permissions (600) ensures only the application user can read sensitive configuration data like database passwords.

---

8 Creating System Service

8.1 Systemd Service File

```
sudo nano /etc/systemd/system/myapp.service
```

```
[Unit]
Description=My Spring Boot Application
After=network.target

[Service]
Type=exec
User=myapp
Group=myapp
ExecStart=/usr/bin/java -jar /opt/myapp/myapp.jar
EnvironmentFile=/opt/myapp/application.env
WorkingDirectory=/opt/myapp
Restart=always
```

Systemd services provide automatic startup, restart on failure, and integration with the system's logging infrastructure. The service runs as the dedicated application user for security.

---

9 Enable and Start Service

9.1 Service Management Commands

```
# Reload systemd configuration
sudo systemctl daemon-reload

# Enable service to start on boot
sudo systemctl enable myapp.service

# Start the service
sudo systemctl start myapp.service

# Check service status
sudo systemctl status myapp.service
```

Enabling the service ensures it starts automatically when the server boots. The status command shows you if the service is running and any recent log entries.

---

10 Production Application Properties

10.1 Database Configuration

```
# application-prod.properties
spring.datasource.url=jdbc:mysql://${DB_HOST}:${DB_PORT}/${DB_NAME}
spring.datasource.username=${DB_USERNAME}
spring.datasource.password=${DB_PASSWORD}

# Production optimizations
spring.jpa.hibernate.ddl-auto=validate
spring.jpa.show-sql=false

# Logging
logging.file.name=/var/log/myapp/application.log
```

Production properties use environment variables for configuration and disable development features like SQL logging. The ddl-auto setting is set to validate to prevent accidental schema changes.

---

11 Testing Your Deployment

11.1 Verification Steps

- Check service status: `sudo systemctl status myapp.service`
- View logs: `sudo tail -f /var/log/myapp/application.log`
- Test health endpoint: `curl http://localhost:8080/actuator/health`
- Verify port listening: `sudo netstat -tlnp | grep :8080`

11.2 Configure Lightsail Firewall

Open port 8080 in Lightsail console → Networking tab

These verification steps help ensure the application is running correctly and accessible. The firewall configuration is temporary - we'll use Nginx as a reverse proxy in the next step.

---

12 Monitoring and Log Management

12.1 Log Rotation Configuration

```
# /etc/logrotate.d/myapp
/var/log/myapp/*.log {
    daily
    rotate 30
    compress
    delaycompress
    missingok
    notifempty
    create 0644 myapp myapp
}
```

12.2 Essential Monitoring Commands

- `sudo systemctl status myapp.service`
- `sudo journalctl -u myapp.service -f`
- `free -h` (memory usage)
- `df -h` (disk usage)

Log rotation prevents disk space issues by automatically compressing and removing old log files. These monitoring commands help you keep track of application health and system resources.

---

13 Troubleshooting Common Issues

13.1 Application Won't Start

- Check Java installation
- Verify JAR permissions
- Review systemd logs
- Check environment file syntax

13.2 Database Connection Issues

- Test database connectivity
- Verify connection strings
- Check firewall rules
- Validate credentials

13.3 Memory Issues

- Monitor usage with `free -h`
- Adjust JVM heap settings
- Consider instance upgrade

Most deployment issues fall into these categories. Systematic troubleshooting using these checkpoints helps identify and resolve problems quickly.

---

14 Summary: What You've Accomplished

14.1 Successfully Deployed

- Installed Java 17 on Lightsail server
- Created secure directory structure
- Transferred application JAR file
- Configured production environment variables
- Set up systemd service for automatic startup
- Implemented logging with rotation
- Configured basic monitoring tools

14.2 Next Step Preview

Configure Nginx reverse proxy for HTTP/HTTPS traffic on standard ports

Your Spring Boot application is now running in production on AWS Lightsail with proper security, monitoring, and automatic startup. The next step will add Nginx for better performance and standard web ports.