Step 6: Set Up Custom Domain and SSL Certificate

Deploy Spring Boot Apps to AWS Lightsail

2026-04-20

Table of contents

1 Welcome to Step 6 2

2 Why SSL and Custom Domains Matter 2

3 Prerequisites Check 3

4 Setting Up Lightsail DNS Zone 3

5 Configuring DNS Records 3

6 Update Domain Registrar 4

7 Installing SSL Tools 4

8 Configure Nginx for Domain 5

9 Test and Reload Nginx 5

10 Obtain SSL Certificate 6

11 Automatic Certificate Renewal 6

12 Update Spring Boot for HTTPS 7

13 Security Configuration (Optional) 7

14 Rebuild and Deploy 8

15 Verify Your HTTPS Setup 8

16 Troubleshooting Common Issues 8

17 Certificate Management 9

18 Summary: What We Accomplished 10

19 Next Steps and Best Practices 10

20 Congratulations! 10

1 Welcome to Step 6

Configure a custom domain and SSL certificate for your Spring Boot application

What we'll accomplish: - Set up professional domain name - Enable HTTPS encryption - Configure automatic certificate renewal - Ensure production-ready security

This final step transforms your application from a development prototype into a production-ready service that users can trust and easily remember.

---

2 Why SSL and Custom Domains Matter

SSL Benefits: - Encrypts data transmission - Browser trust indicators - SEO ranking improvements - Protection from attacks

Custom Domain Benefits: - Professional appearance - Easy to remember URLs - Brand recognition - User confidence

Modern browsers actively warn users about non-HTTPS sites, making SSL essential. Custom domains provide credibility and make your application memorable for users.

---

3 Prerequisites Check

Before we begin, ensure you have:

- Registered domain name
- Access to DNS management
- Lightsail instance running
- Nginx configured (from Step 5)
- Spring Boot app accessible via IP

If any of these prerequisites are missing, go back and complete the previous steps before proceeding with domain and SSL configuration.

---

4 Setting Up Lightsail DNS Zone

Step 1: Create DNS Zone 1. Navigate to **Networking** → **DNS zones** 2. Click **Create DNS zone** 3. Enter your domain name 4. Click **Create DNS zone**

> Tip
>
> The DNS zone centralizes all DNS record management for your domain

Lightsail's DNS zones provide a simple interface for managing DNS records without needing to work directly with Route 53 or external DNS providers.

---

5 Configuring DNS Records

Required DNS Records:

```
# A Record: Points domain to your instance
example.com → Your Lightsail Instance IP

# CNAME Record: Points www to root domain
www.example.com → example.com
```

Configuration Steps: - **Subdomain**: Leave blank for root, enter `www` for www - **Resolves to**: Select your Lightsail instance

The A record creates the primary connection between your domain and server, while the CNAME record ensures both www and non-www versions work correctly.

---

6 Update Domain Registrar

Steps: 1. Copy Lightsail name servers 2. Access registrar control panel 3. Replace existing name servers 4. Save changes

Example Name Servers:

```
ns-123.awsdns-12.com
ns-456.awsdns-45.net
ns-789.awsdns-78.org
ns-012.awsdns-01.co.uk
```

DNS propagation takes up to 48 hours

This step tells the internet where to find your domain. The name servers act like a phone book, directing traffic to your Lightsail DNS zone.

---

7 Installing SSL Tools

SSH into your instance and install Certbot:

```
# Update package list
sudo apt update

# Install Certbot and Nginx plugin
sudo apt install certbot python3-certbot-nginx -y

# Verify installation
certbot --version
```

Certbot is the standard tool for obtaining and managing Let's Encrypt SSL certificates. The Nginx plugin automates the configuration process.

---

8 Configure Nginx for Domain

Update your Nginx configuration:

```
sudo nano /etc/nginx/sites-available/default
```

```
server {
    listen 80;
    server_name example.com www.example.com;

    location / {
        proxy_pass http://localhost:8080;
        proxy_set_header Host $host;
        proxy_set_header X-Real-IP $remote_addr;
        proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
        proxy_set_header X-Forwarded-Proto $scheme;
    }
}
```

The server_name directive tells Nginx which domains this configuration should handle. The proxy headers ensure your Spring Boot app receives correct client information.

---

9 Test and Reload Nginx

```
# Test configuration syntax
sudo nginx -t

# Reload Nginx with new configuration
sudo systemctl reload nginx
```

Expected output:

```
nginx: configuration file /etc/nginx/nginx.conf test is successful
```

Always test Nginx configuration before reloading. A syntax error could cause Nginx to fail, making your site inaccessible.

---

10 Obtain SSL Certificate

Use Certbot to automatically get and install SSL:

```
sudo certbot --nginx -d example.com -d www.example.com
```

Certbot will prompt for: {.incremental} - Email address for notifications - Agreement to terms of service - Optional email sharing with EFF - Automatic HTTPS redirect (recommended: Yes)

Certbot handles the entire SSL certificate process: domain validation, certificate generation, Nginx configuration updates, and HTTPS redirect setup.

---

11 Automatic Certificate Renewal

SSL certificates expire every 90 days. Set up automatic renewal:

```
# Test renewal process (dry run)
sudo certbot renew --dry-run

# Check renewal timer status
sudo systemctl status certbot.timer

# Enable timer if needed
sudo systemctl enable certbot.timer
sudo systemctl start certbot.timer
```

Let's Encrypt certificates are short-lived for security. The automatic renewal system ensures your certificates stay valid without manual intervention.

---

12 Update Spring Boot for HTTPS

Production Properties:

```
# application-prod.properties
server.forward-headers-strategy=native
server.tomcat.use-relative-redirects=true
server.servlet.session.cookie.secure=true
server.servlet.session.cookie.http-only=true
```

These settings ensure your app works correctly behind the Nginx HTTPS proxy.

The forward-headers-strategy tells Spring Boot to trust headers from Nginx, while cookie settings enhance security for HTTPS connections.

---

13 Security Configuration (Optional)

If using Spring Security, add HTTPS enforcement:

```
@Configuration
@EnableWebSecurity
public class SecurityConfig {
    @Bean
    public SecurityFilterChain filterChain(HttpSecurity http) throws Exception {
        http
            .requiresChannel(channel ->
                channel.requestMatchers(r -> r.getHeader("X-Forwarded-Proto") != null)
                       .requiresSecure())
            .headers(headers ->
                headers.httpStrictTransportSecurity(hstsConfig ->
                    hstsConfig.maxAgeInSeconds(31536000)
                              .includeSubdomains(true)));
        return http.build();
    }
}
```

This configuration adds HTTP Strict Transport Security headers and ensures all requests use HTTPS, providing additional security layers.

---

14 Rebuild and Deploy

Deploy your updated application:

```
# Build locally
./mvnw clean package -DskipTests

# Transfer to server
scp target/myapp.jar ubuntu@your-domain.com:~/

# Deploy on server
sudo systemctl stop myapp
sudo cp /home/ubuntu/myapp.jar /opt/myapp/
sudo systemctl start myapp
```

This deployment cycle should become familiar. Always stop the service before replacing the JAR file to ensure a clean restart with new configurations.

---

15 Verify Your HTTPS Setup

Testing checklist: {.incremental} 1. Navigate to `https://yourdomain.com` 2. Verify SSL certificate (lock icon) 3. Confirm HTTP → HTTPS redirect 4. Test with SSL Labs: `ssllabs.com/ssltest`

```
# Command line verification
curl -I https://yourdomain.com
```

Thorough testing ensures your users will have a seamless, secure experience. The SSL Labs test provides a comprehensive security analysis.

---

16 Troubleshooting Common Issues

Domain not resolving:

```
dig yourdomain.com
nslookup yourdomain.com
```

SSL certificate problems:

```
curl -I http://yourdomain.com
sudo systemctl status nginx
sudo tail -f /var/log/nginx/error.log
```

Mixed content warnings: - Update hardcoded HTTP URLs - Use protocol-relative URLs (`//example.com/resource`)

Most issues stem from DNS propagation delays or firewall configurations. These diagnostic commands help identify the root cause quickly.

---

17 Certificate Management

Monitor certificate status:

```
# View all certificates
sudo certbot certificates

# Check certificate details
openssl x509 -in /etc/letsencrypt/live/yourdomain.com/cert.pem -text -noout

# Manual renewal (if needed)
sudo certbot renew
```

Regular monitoring ensures you catch any renewal issues before certificates expire. The automatic system is reliable, but manual oversight provides peace of mind.

---

18 Summary: What We Accomplished

Production-Ready Features: - Custom domain with professional appearance - SSL certificate with HTTPS encryption - Automatic certificate renewal - Security headers and browser trust - SEO-friendly HTTPS configuration

Security Enhancements: - Encrypted data transmission - Protection from man-in-the-middle attacks - Browser security indicators - Enhanced user confidence

Your application now meets enterprise standards for security and professionalism. Users can trust and easily access your Spring Boot application.

---

19 Next Steps and Best Practices

Recommended next actions: - Set up monitoring for domain/certificate health - Implement additional security headers - Plan for scaling beyond Lightsail if needed - Regular security audits and updates

Maintenance schedule: - Monthly SSL certificate checks - Quarterly security reviews - Monitor application performance and logs

With SSL and custom domain configured, your focus shifts to monitoring, maintenance, and potential scaling as your application grows in usage.

---

20 Congratulations!

Your Spring Boot application is now: - **Accessible** via custom domain - **Secure** with HTTPS encryption
- **Professional** with browser trust indicators - **Production-ready** for real users

You've successfully deployed a complete Spring Boot application to AWS Lightsail!

This represents a significant achievement - you've taken an application from development through to production deployment with enterprise-grade security and accessibility.