

ELK Stack Integration Guide

Overview

This guide explains how to set up and use the ELK (Elasticsearch, Logstash, Kibana) stack to view and analyze application logs.

Prerequisites

- Docker Desktop installed and running
- Project dependencies updated (run `./gradlew build`)

Files Created

1. `build.gradle` (Updated)

- Added Logstash encoder dependency: `net.logstash.logback:logstash-logback-encoder:7.4`

2. `logback-spring.xml`

- Configures logging output format (JSON for ELK)
- Sets up dual output: console and Logstash
- Sends logs to Logstash on port 5000

3. `docker-compose.yml`

- Defines ELK stack services
- **Elasticsearch**: Stores logs (ports 9200, 9300)
- **Logstash**: Processes logs (port 5000)
- **Kibana**: Web UI for viewing logs (port 5601)

4. `logstash/pipeline/logstash.conf`

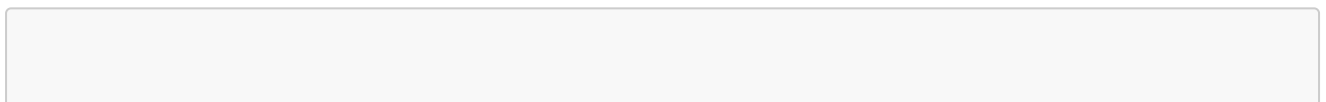
- Defines log processing pipeline
- Input: TCP on port 5000
- Output: Elasticsearch index pattern `spring-boot-logs-YYYY.MM.dd`

5. `logstash/config/logstash.yml`

- Basic Logstash configuration

Setup Steps

Step 1: Start ELK Stack



```
cd /Users/aerloki/Downloads/Personal-Project
docker-compose up -d
```

Wait 1-2 minutes for services to fully start.

Step 2: Verify Services

Check that all containers are running:

```
docker-compose ps
```

You should see three containers:

- elasticsearch (healthy)
- logstash (healthy)
- kibana (healthy)

Step 3: Restart Spring Boot Application

```
pkill -f "gradle.*bootRun"
./gradlew bootRun
```

Accessing Kibana

1. Open Kibana

Open your browser and navigate to:

```
http://localhost:5601
```

2. Configure Index Pattern (First Time Only)

1. Go to **Management** → **Stack Management** → **Index Patterns**
2. Click **Create index pattern**
3. Enter pattern: **spring-boot-logs-***
4. Click **Next step**
5. Select **@timestamp** as Time field
6. Click **Create index pattern**

3. View Logs

1. Click on **Discover** in the left sidebar

2. Select `spring-boot-logs-*` index pattern
3. You'll see all application logs in real-time!

Log Structure

Our logs include:

- **timestamp**: When the log was created
- **level**: Log level (INFO, DEBUG, ERROR, etc.)
- **logger**: Class that generated the log
- **message**: Log message with structured data
- **application**: "personal-project"
- **traceId/spanId**: For distributed tracing

Example: Order Logs

When an order is placed, you'll see logs like:

```
{
  "@timestamp": "2025-12-13T02:00:00.000Z",
  "level": "INFO",
  "logger": "CheckoutController",
  "message": "Order placement initiated - Name: John Doe, Address: 123
Main St, Seattle, WA 98101",
  "application": "personal-project"
}
```

Useful Kibana Queries

View only order-related logs:

```
message: "Order"
```

View ERROR level logs:

```
level: "ERROR"
```

View logs from CheckoutController:

```
logger: "CheckoutController"
```

Troubleshooting

Logs not appearing in Kibana?

1. Check Logstash is running: `docker logs logstash`
2. Verify connection: `curl http://localhost:9200/_cat/indices`
3. Restart ELK stack: `docker-compose restart`

Can't access Kibana?

- Wait 2-3 minutes after `docker-compose up`
- Check status: `docker-compose ps`
- View logs: `docker logs kibana`

Stopping ELK Stack

```
docker-compose down
```

To remove volumes (deletes all logs):

```
docker-compose down -v
```

Important Notes

1. **Form Field Issue:** Remember to REFRESH your browser (F5) on the checkout page to see the pre-filled default values for Name and Address fields
2. **Structured Logging:** Our SLF4J logs are automatically formatted as JSON for Kibana
3. **Real-time:** Logs appear in Kibana immediately (1-2 second delay)
4. **Retention:** Logs are stored indefinitely unless you delete the volume

Next Steps

1. Start ELK stack: `docker-compose up -d`
2. Wait 2 minutes
3. Refresh Spring Boot application
4. Access Kibana at `http://localhost:5601`
5. Create index pattern (first time only)
6. View logs in Discover tab!

Happy logging! 🎉