WildFly Swarm 服务构建实战
Building MicroServices with WildFly Swarm

Jim Ma
Senior software engineer
2015-11-13

Who is Jim

- Program with java since 2000
- Wildfly/JBoss WebService core developer
- Apache CXF PMC/Commitor
- Contributed code to various open source project:

  jBPM, JAXB, WSS4j, Xerces...
- weibo.com/jimmacn

MicroServices 微服务

- Decoupled Components.

  系统解耦，拆分成多个服务
- Independent release cycles (continuous deployment).

  服务开发拥有独立的周期
- HTTP, REST or otherwise networked.

  服务之间通常采用 Http, Rest 通讯方式
- No limit on size, really.

  服务不局限于开发语言和大小
-

MicroService 微服务

- Micro functionality, not micro lines-of-code.

  服务具有通常具有简单小巧的功能，但不是说一个服务只有几行代码

- Preferably self-contained.

  服务通常不需要容器运行

- But not just Docker-izing everything.

  不只是 Docker-izing 现有的东西

WildFly

- Java-EE application-server

  Java-EE 应用服务器
- ALL OF JAVA-EE

  是包含 JavaEE 规范所规定的所有组件
- So, it’s big but it's faster

  160.7 M ～ 2s 启动

WildFly Swarm

- WildFly, broken apart

  将 wildfly 拆分，自由组合
- Maven-addressable components

  提供 maven plugin 来组织服务
- Fat-jarrable fat-jar 方式启动服务
- You can provide your own main(...)

  支持用户创建的 main 函数
- Programatic configuration (instead of standalone.xml)

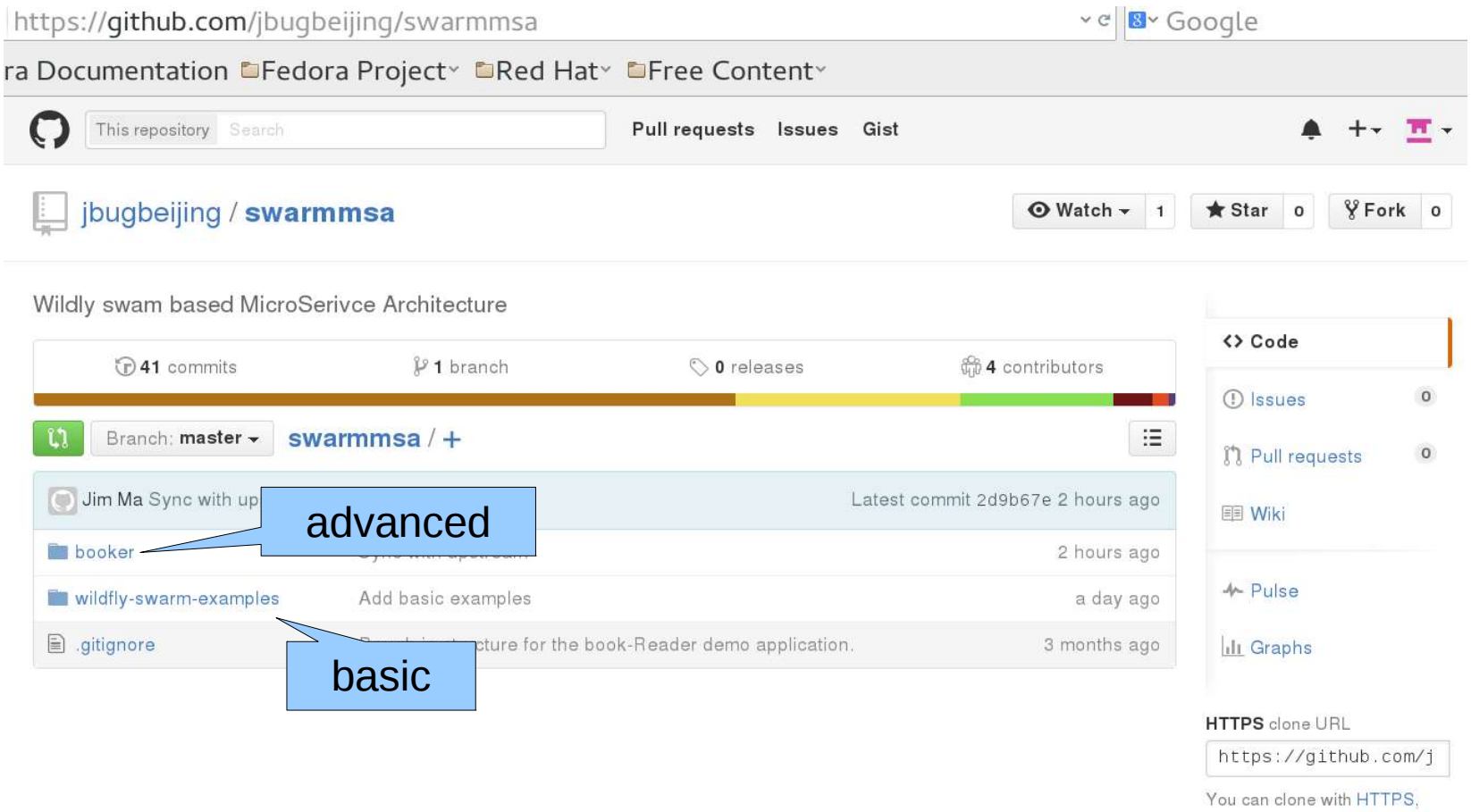
  通过程序来配置运行时组件

WildFly Swarm

- Automatic configuration 自动进行配置
- Convention over Configuration 约定优于配置
- Beyond Java-EE 不只是 Java EE
  - Netflix Ribbon
  - Logstash

Let's get started

- git clone https://github.com/jbugbeijing/swarmmsa.git



Login to laptop:

- Fedora22
- username/password: ***jboss/jboss***
- java/maven/ant are ready
- Eclipse4 : ***/home/jboss/java/eclipse4***
- All demo code already cloned under: ***/home/jboss/code***

Example1: Create Service with Servlet

requirement:

- **java8**
- **maven3**
- **git clone https://github.com/jbugbeijing/swarmmsa.git**

Code is here

jbugbeijing / swarmmsa

Watch 1 Star 0 Fo

Branch: master swarmmsa / wildfly-swarm-examples / servlet / +

Jim Ma Add basic examples Latest commit 14ff1db a day ago

..

src/main/java/org/wildfly/swarm/examples/servlet	Add basic examples	a day ago
README.md	Add basic examples	a day ago
pom.xml	Add basic examples	a day ago

README.md

Servlet .war Example

This example takes a normal servlet build, and wraps it into a `-swarm` runnable jar.

> Please raise any issues found with this example in this repo: https://github.com/wildfly-swarm/wildfly-swarm-examples
>
> Issues related to WildFly Swarm core should be raised in the main repo: https://github.com/wildfly-swarm/wildfly-swarm/issues

Simple plugin and simple dependency

- Now

```
<build>
  <plugins>
    <plugin>
      <groupId>org.apache.maven.plugins</groupId>
      <artifactId>maven-war-plugin</artifactId>
      <configuration>
        <failOnMissingWebXml>false</failOnMissingWebXml>
      </configuration>
    </plugin>
    <plugin>
      <groupId>org.wildfly.swarm</groupId>
      <artifactId>wildfly-swarm-plugin</artifactId>
    </plugin>
  </plugins>
</build>

<dependencies>
  <dependency>
    <groupId>org.wildfly.swarm</groupId>
    <artifactId>wildfly-swarm-undertow</artifactId>
    <version>${version.wildfly-swarm}</version>
  </dependency>
  <dependency>
    <groupId>joda-time</groupId>
    <artifactId>joda-time</artifactId>
    <version>2.7</version>
  </dependency>
</dependencies>
```

Before

```
    <dependency>
        <groupId>org.jboss.spec.javax.annotation</groupId>
        <artifactId>jboss-annotations-api_1.2_spec</artifactId>
        <scope>provided</scope>
    </dependency>

    <!-- Import the Servlet API, we use provided scope as the API is included in WildFly -->
    <dependency>
        <groupId>org.jboss.spec.javax.servlet</groupId>
        <artifactId>jboss-servlet-api_3.1_spec</artifactId>
        <scope>provided</scope>
    </dependency>

</dependencies>

<build>
    <!-- Set the name of the WAR, used as the context root when the app
        is deployed -->
    <finalName>${project.artifactId}</finalName>
    <plugins>
        <!-- WildFly plug-in to deploy the WAR -->
        <plugin>
            <groupId>org.wildfly.plugins</groupId>
            <artifactId>wildfly-maven-plugin</artifactId>
            <version>${version.wildfly.maven.plugin}</version>
            <!--
```

Simple to get this start:

- Now

```
package org.wildfly.swarm.examples.servlet;

import java.io.IOException;

import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

import org.joda.time.DateTime;

/**
 * @author Bob McWhirter
 */
@WebServlet("/*")
public class MyServlet extends HttpServlet {

    @Override
    protected void doGet(HttpServletRequest req, HttpServletResponse resp) throws ServletException, IOException {
        resp.getWriter().write("Howdy at " + new DateTime());
    }
}
```

- java -jar wildfly-swarm-example-servlet-swarm.jar

- Before

```
package org.wildfly.swarm.examples.servlet;

import java.io.IOException;

import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

import org.joda.time.DateTime;

/**
 * @author Bob McWhirter
 */
@WebServlet("/*")
public class MyServlet extends HttpServlet {

    @Override
    protected void doGet(HttpServletRequest req, HttpServletResponse resp) throws ServletException, IOException {
        resp.getWriter().write("Howdy at " + new DateTime());
    }
}
```

- ./bin/standalon.sh
- Deploy war

Example2: Create service with jax-rs

requirement:

- java8
- maven3
- git clone https://github.com/jbugbeijing/swarmmsa.git

Code

jbugbeijing / swarmmsa

Watch 1 | Star 0 | Fork 0

Branch: master swarmmsa / wildfly-swarm-examples / jaxrs / +

Jim Ma Add basic examples — Latest commit 14ff1db a day ago

..		
src/main/java/org/wildfly/swarm/examples/jaxrs	Add basic examples	a day ago
README.md	Add basic examples	a day ago
pom.xml	Add basic examples	a day ago
verify.sh	Add basic examples	a day ago

2 plugins and 2 dependencies

```
<build>
  <plugins>
    <plugin>
      <groupId>org.apache.maven.plugins</groupId>
      <artifactId>maven-war-plugin</artifactId>
      <configuration>
        <failOnMissingWebXml>false</failOnMissingWebXml>
        <packagingExcludes>WEB-INF/lib/wildfly-swarm-*.jar</packagingExcludes>
      </configuration>
    </plugin>
    <plugin>
      <groupId>org.wildfly.swarm</groupId>
      <artifactId>wildfly-swarm-plugin</artifactId>
    </plugin>
  </plugins>
</build>

<dependencies>
  <dependency>
    <groupId>org.wildfly.swarm</groupId>
    <artifactId>wildfly-swarm-jaxrs</artifactId>
    <version>${version.wildfly-swarm}</version>
  </dependency>
  <dependency>
    <groupId>joda-time</groupId>
    <artifactId>joda-time</artifactId>
    <version>2.7</version>
  </dependency>
</dependencies>
```

Two classes

```java
package org.wildfly.swarm.examples.jaxrs;

import javax.ws.rs.ApplicationPath;
import javax.ws.rs.core.Application;

/**
 * @author Bob McWhirter
 */
@ApplicationPath("/")
public class MyApplication extends Application {

    public MyApplication() {
    }
}
```

```java
package org.wildfly.swarm.examples.jaxrs;

import javax.ws.rs.GET;
import javax.ws.rs.Path;
import javax.ws.rs.Produces;

import org.joda.time.DateTime;

/**
 * @author Bob McWhirter
 */
@Path("/")
public class MyResource {

    @GET
    @Produces("text/plain")
    public String get() {
        // Prove we can use an external dependency and wei
        return "Howdy at " + new DateTime() + ".  Have a JI
    }
}
```

Example3:Create service with jaxrs,cdi and jpa

requirement:

- java8
- maven3
- git clone https://github.com/jbugbeijing/swarmmsa.git

Simple plugins dependencies

- Focus on code, not package and deploy

```
<build>
  <plugins>
    <plugin>
      <groupId>org.apache.maven.plugins</groupId>
      <artifactId>maven-war-plugin</artifactId>
      <version>2.6</version>
      <configuration>
        <failOnMissingWebXml>false</failOnMissingWebXml>
      </configuration>
    </plugin>
    <plugin>
      <groupId>org.wildfly.swarm</groupId>
      <artifactId>wildfly-swarm-plugin</artifactId>
      <executions>
        <execution>
          <goals>
            <goal>package</goal>
          </goals>
        </execution>
      </executions>
    </plugin>
  </plugins>
</build>

<dependencies>
  <dependency>
    <groupId>org.wildfly.swarm</groupId>
    <artifactId>wildfly-swarm-jaxrs-weld</artifactId>
    <version>${version.wildfly-swarm}</version>
  </dependency>
  <dependency>
    <groupId>org.wildfly.swarm</groupId>
    <artifactId>wildfly-swarm-jpa</artifactId>
    <version>${version.wildfly-swarm}</version>
  </dependency>
</dependencies>
```

JPA,JAX-RS And CDI

```
package org.wildfly.swarm.examples.jpa;

import javax.inject.Inject;
import javax.ws.rs.GET;
import javax.ws.rs.Path;
import javax.ws.rs.Produces;

/**
 * @author Ken Finnigan
 */
@Path("/")
public class EmployeeResource {

    @Inject
    PersistenceHelper helper;

    @GET
    @Produces("application/json")
    public Employee[] get() {
        return helper.getEntityManager()
          .createNamedQuery("Employee.findAll", Employee.class)
          |.getResultList().toArray(new Employee[0]);
    }
}
```

JPA,JAX-RS And CDI

```
@Entity
@Table(name = "REST_DB_ACCESS")
@NamedQueries({
        @NamedQuery(name = "Employee.findAll", query = "SELECT e FROM Employee e")
})
@XmlRootElement
public class Employee implements Serializable {
    private static final long serialVersionUID = 1L;

    @Id
    @GeneratedValue(strategy = GenerationType.AUTO)
    private int id;

    @Column(length = 40)
    private String name;

    public Employee() {
    }

    public Employee(String name) {
        this.name = name;
    }
```

Your turn now...

- **Run these examples on the laptops:**

  - servlet

  - jax-rs

  - jax-rs, cdi and jpa

- **Implement a service that uses jax-rs and jpa apis and verify it works**

  - complete code in EmployeeResource.java

  - service to add an employee

  - use curl to test the rest api works

Advanced Example: Booker

What is Booker ?

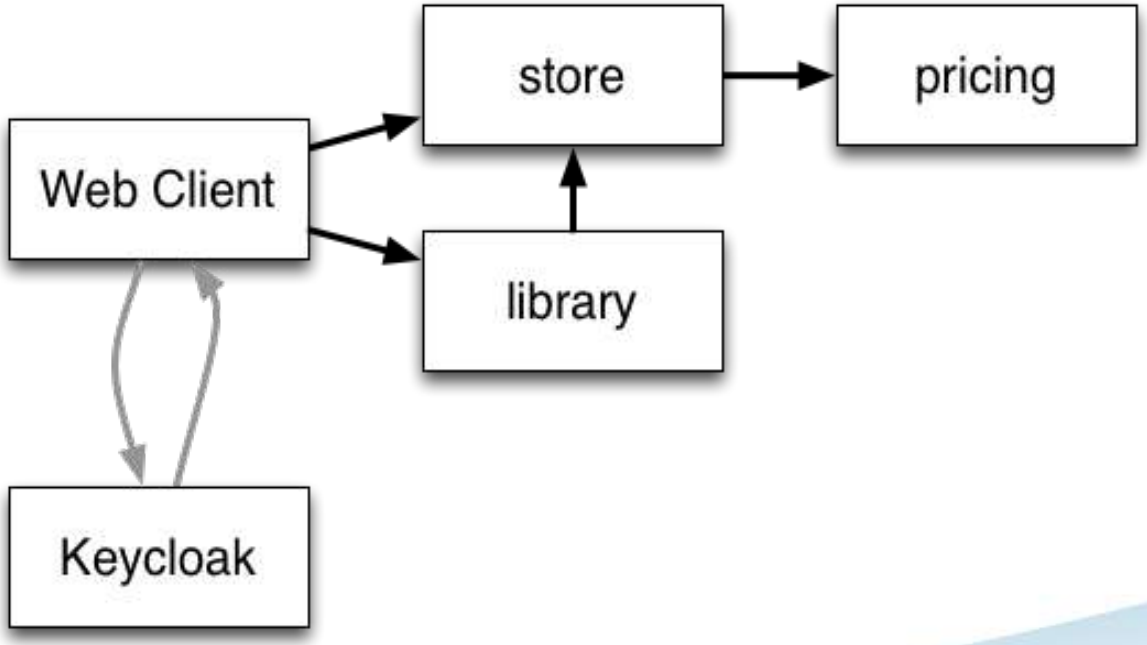
- Akin to the Amazon Kindle Store
- Search for books
- Get price for books
- Buy books
- Keep your library of purchased books

What is Booker ?

- Authentication using Keycloak
- Logging using Logstash
- React.js single-page-app Web UI
- Inter-service communication using Ribbon

What is Booker ?

- Authentication using Keycloak
- Logging using Logstash
- React.js single-page-app Web UI
- Inter-service communication using Ribbon



Keycloak

- Single-sign-on with JSON Web Token(JWT)
- Supports social login
- Bearer tokens for automated access
- Token propagation for chained service invocations
- $BOOKER_HOME/extra/keycloak/README.md
- http://downloads.jboss.org/keycloak/1.5.0.Final/keycloak-1.5.0.Final.zip
- Unzip
- cd keycloak-1.5.0.Final/
- ./bin/standalone.sh -Djboss.http.port=9090

Securing from .js

```
keycloak.init({ onLoad: 'check-sso' }).success( function() {
  if ( keycloak.authenticated ) {
    keycloak.loadUserInfo().success( function(info) {
      Booker.Actions.UserLoggedIn( info );
    });
  }
  Router.run(routes, Router.HistoryLocation, function (Handler) {
    React.render(<Handler/>, document.getElementById('app'));
  });
})
```

Ribbon

- Ribbon is a client side IPC library that is battle-tested in cloud. It provides the following features

  Load balancing

  Fault tolerance

  Multiple protocol (HTTP, TCP, UDP) support in an asynchronous and reactive model

  Caching and batching

- Ribbon in Java uses WildFly Clustering to wire together services

Ribbon

- Ribbon is a client side IPC library that is battle-tested in cloud. It provides the following features

  Load balancing

  Fault tolerance

  Multiple protocol (HTTP, TCP, UDP) support in an asynchronous and reactive model

  Caching and batching

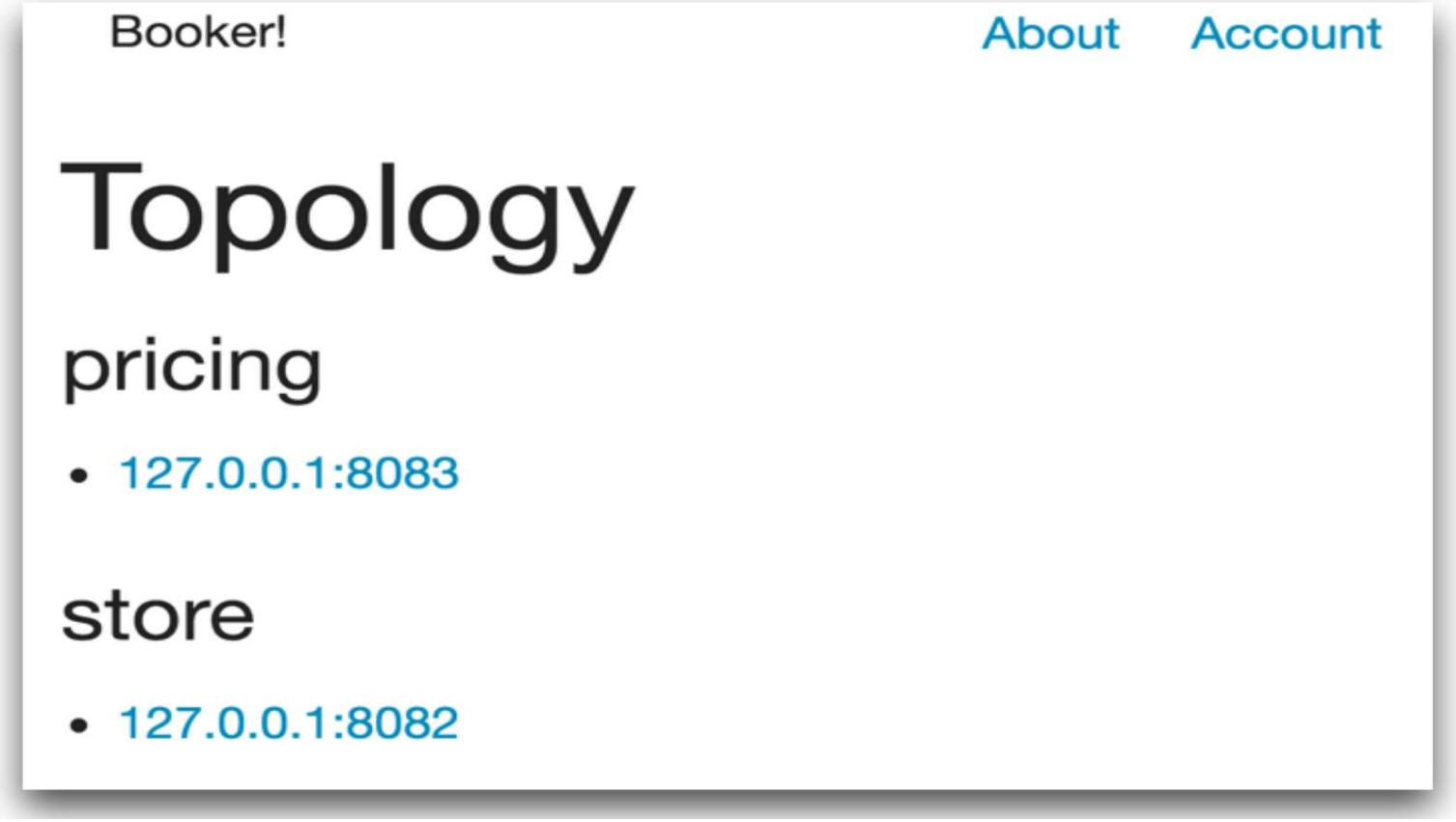
- Ribbon in Java uses WildFly Clustering to wire together services

Ribbon

```
this.topology = (RibbonTopology) context.lookup("jboss/ribbon/cluster");

. . .

resp.setContentType("text/event-stream");
resp.setCharacterEncoding("UTF-8");

AsyncContext asyncContext = req.startAsync();
PrintWriter writer = resp.getWriter();

RibbonTopologyListener topologyListener = new RibbonTopologyListener() {
    @Override
    public void onChange(RibbonTopology topology) {
        String json = topologyToJson();
        writer.write( "data: " + json );
        writer.flush();
    }
};
```

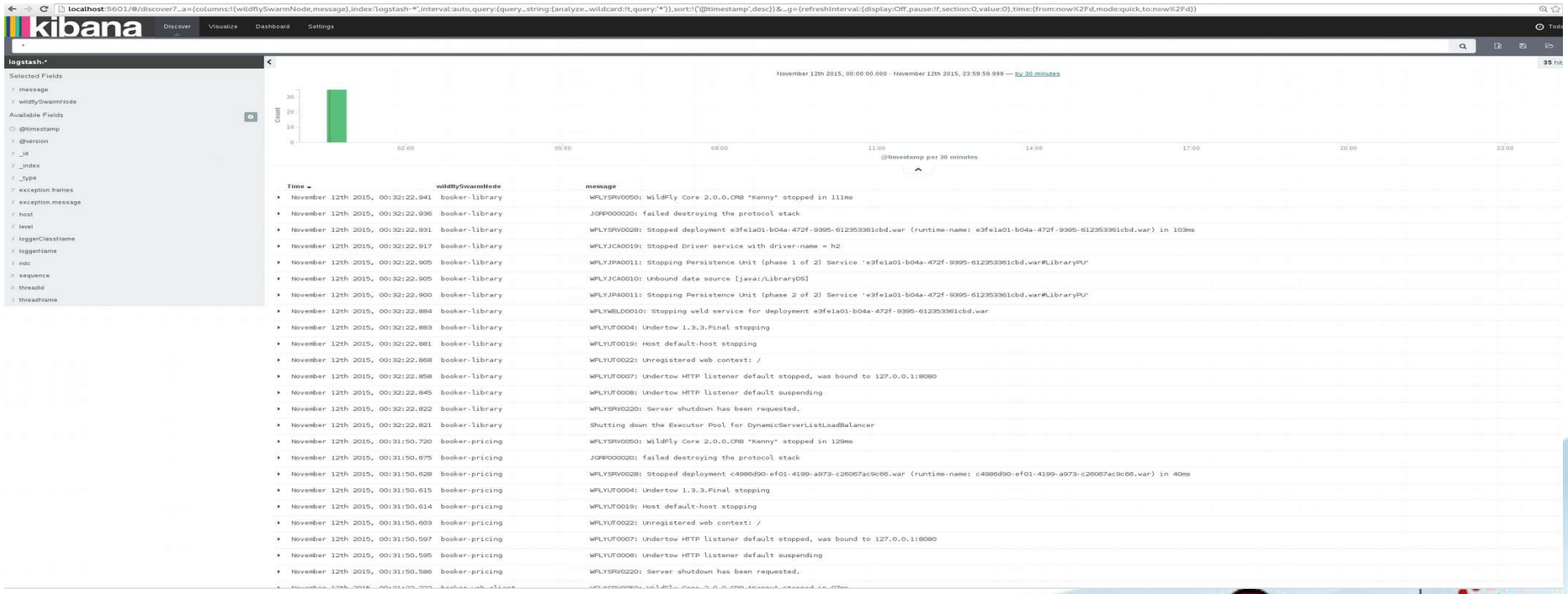
Ribbon

http://localhost:8080/topology



Logstash + Kibana

- If you deploy a lot of services, that’s a lot of logs to keep up with
- Logstash + Kibana lets you log to a central
- location, and search them in aggregate



Start all services

- Export BOOKER_HOME
- BOOKER_HOME/bin/run_all.sh
- http://localhost:8080

Q&A