Building Microservices with **WildFly Swarm**

Bob McWhirter
QCon Rio

RED HAT® JBOSS® MIDDLEWARE | JBossDeveloper

Who is Bob?

- Red Hat Middleware (JBoss)
- Director of Research & Prototyping
- Co-lead of WildFly Swarm
- Founder of…
  - The Codehaus
  - Drools
  - TorqueBox

Microservices

- Decoupled Components.
- Independent release cycles (continuous deployment).
- HTTP, REST or otherwise networked.
- Cattle, not pets.

Microservices

- No limit on size, really.
- Micro functionality, not micro lines-of-code.
- Preferably self-contained.
- But not just Docker-izing everything.

WildFly

- Java-EE application-server
- ALL OF JAVA-EE
- So, it’s big.
- But it’s also fast, and awesome.
- Did I mention “big”?

WildFly Swarm

- WildFly, broken apart
- Maven-addressable components
- Fat-jarrable
- You can provide your own `main(…)`
- Programatic configuration (instead of `standalone.xml`)

WildFly Swarm

- Automatic configuration
- Convention over Configuration
- Beyond Java-EE
  - Netflix Ribbon
  - Logstash

Example Application
Booker

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

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

Clone Booker

- GitHub: https://github.com/wildfly-swarm/booker
- `git clone https://github.com/wildfly-swarm/booker.git`

Keycloak

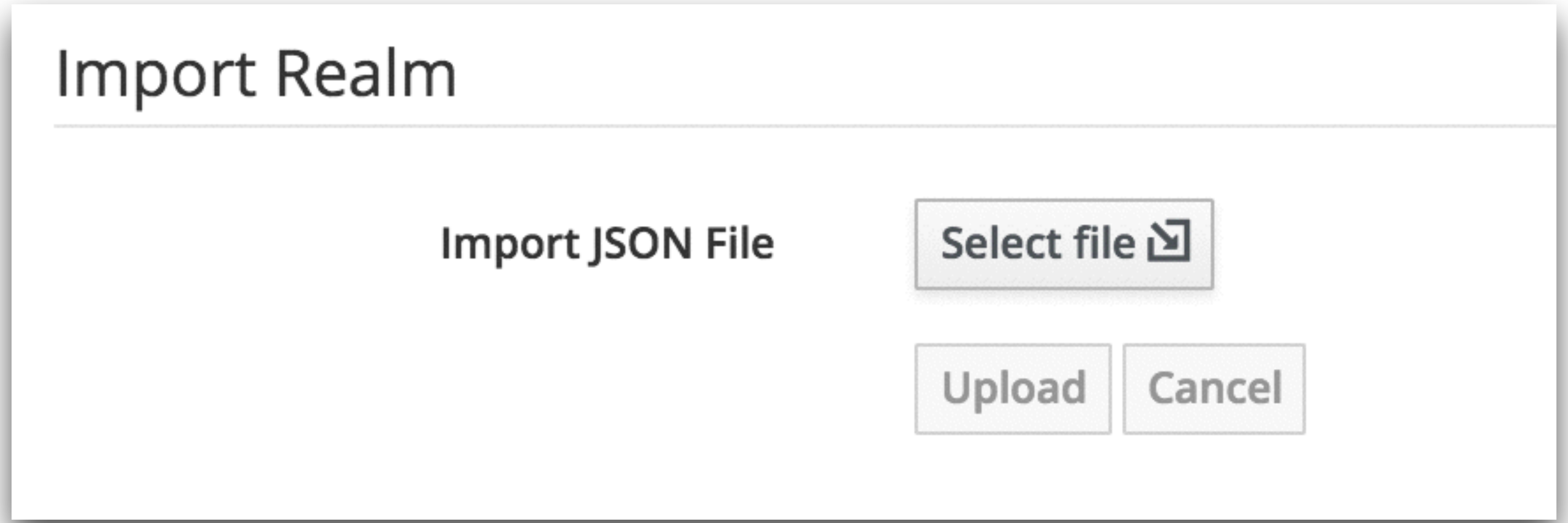
Keycloak

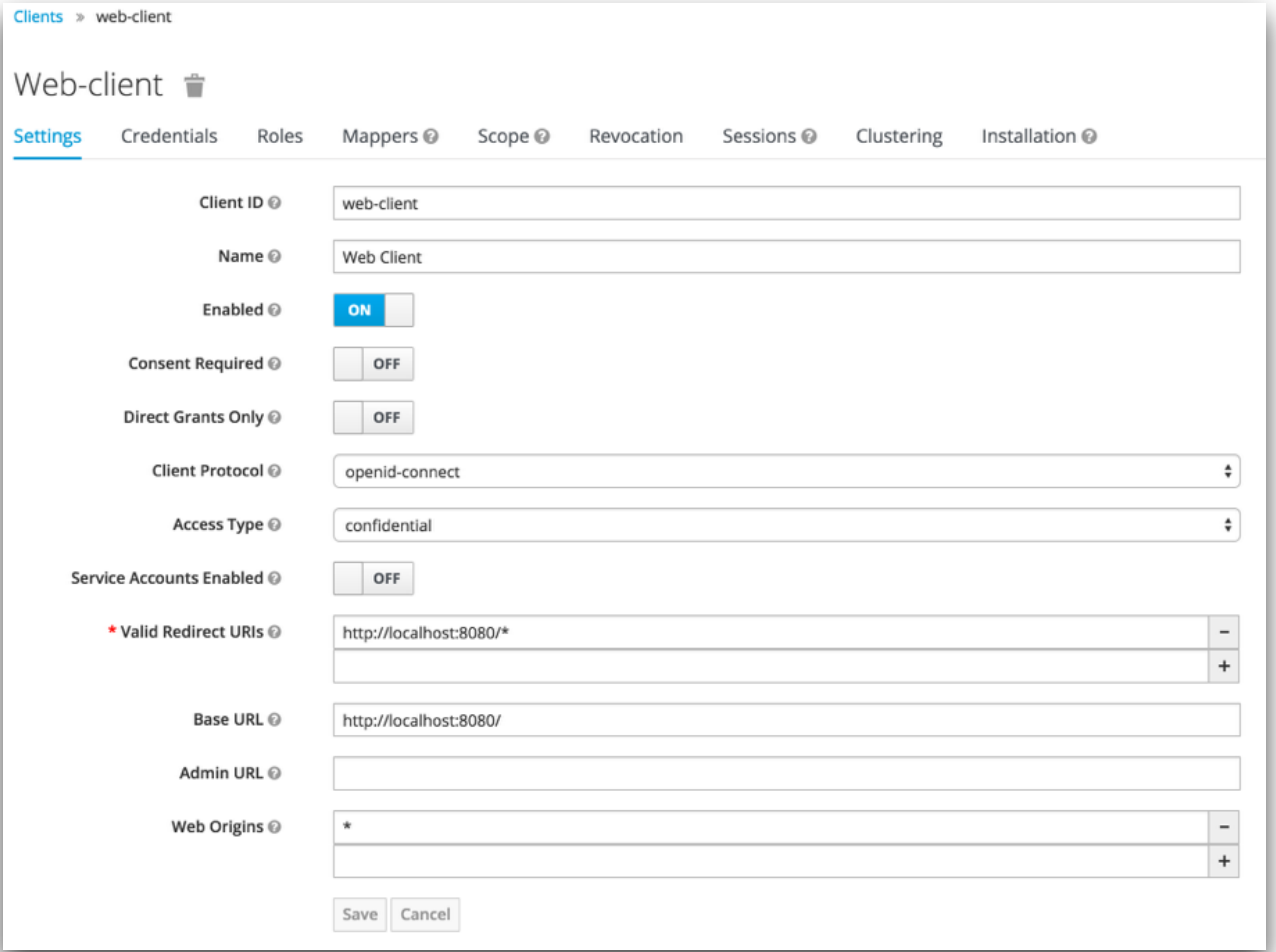
- Single-sign-on
- Supports social login
- Bearer tokens for automated access
- Token propagation for chained service invocations

- $BOOKER_ROOT/extra/keycloak/README.md
- http://downloads.jboss.org/keycloak/1.4.0.Final/keycloak-1.4.0.Final.zip
- Unzip
- cd keycloak-1.4.0.Final/
- ./bin/standalone.sh -Djboss.http.port=9090

http://localhost:9090/

admin / admin


Import Realm
Import JSON File
Select file
Upload
Cancel


Clients » web-client

Web-client

Client ID	web-client
Name	Web Client
Enabled	ON
Consent Required	OFF
Direct Grants Only	OFF
Client Protocol	openid-connect
Access Type	confidential
Service Accounts Enabled	OFF
* Valid Redirect URIs	http://localhost:8080/*
Base URL	http://localhost:8080/
Admin URL	
Web Origins	*

Keycloak

- Managed separate from your application
- Organization-wide
- Can delegate to Kerberos, LDAP, etc
- Can do social login (Twitter, GitHub, etc)
- Deserves its own workshop, really…

Web Client

Web Client

- Single-Page Application
- React.js to draw the pages
- Reflux to manage state
- React-Router for routing requests without reloads

React.js

- Just the view portion
- Virtual DOM, straightforward to update browser
- We’re using .jsx compiler in the browser, for simplicity. DO NOT USE FOR PRODUCTION
- Deserves its own workshop, really…

Other Bits

- **Reflux**
  - Handles updating the view when state changes
- **React-Router**
  - Handles updating the view when the URL changes

Web Client

```
<packaging>war</packaging>
```

Web Client

No **`main(…)`**

Web Client

```
<dependency>
  <groupId>org.wildfly.swarm</groupId>
  <artifactId>wildfly-swarm-undertow</artifactId>
</dependency>
```

Web Client

```
mvn wildfly-swarm:run
```

```
12:10:30,510 INFO  [org.jboss.msc] (main) JBoss MSC version 1.2.6.Final
12:10:30,654 INFO  [org.jboss.as] (MSC service thread 1-6) WFLYSRV0049: WildFly Core 2.0.0.Beta1 "Kenny"
12:10:31,086 INFO  [org.jboss.as.controller.management-deprecated] (ServerService Thread Pool -- 6) WFLYC
2015-08-19 12:10:31,199 INFO  [org.jboss.as.clustering.jgroups] (ServerService Thread Pool -- 17) WFLYCLJ
2015-08-19 12:10:31,203 INFO  [org.wildfly.extension.io] (ServerService Thread Pool -- 12) WFLYIO001: Wor
2015-08-19 12:10:31,213 INFO  [org.jboss.as.naming] (ServerService Thread Pool -- 14) WFLYNAM0001: Activa
2015-08-19 12:10:31,223 INFO  [org.jboss.as.security] (ServerService Thread Pool -- 16) WFLYSEC0002: Acti
2015-08-19 12:10:31,226 INFO  [org.jboss.as.security] (MSC service thread 1-2) WFLYSEC0001: Current Picke
2015-08-19 12:10:31,231 INFO  [org.wildfly.extension.undertow] (MSC service thread 1-3) WFLYUT0003: Under
2015-08-19 12:10:31,231 INFO  [org.wildfly.extension.undertow] (ServerService Thread Pool -- 11) WFLYUT00
2015-08-19 12:10:31,253 INFO  [org.jboss.as.naming] (MSC service thread 1-5) WFLYNAM0003: Starting Naming
2015-08-19 12:10:31,275 INFO  [org.xnio] (MSC service thread 1-7) XNIO version 3.3.1.Final
2015-08-19 12:10:31,285 INFO  [org.xnio.nio] (MSC service thread 1-7) XNIO NIO Implementation Version 3.3
2015-08-19 12:10:31,329 INFO  [org.wildfly.extension.undertow] (MSC service thread 1-8) WFLYUT0012: Start
2015-08-19 12:10:31,382 INFO  [org.wildfly.extension.undertow] (MSC service thread 1-2) WFLYUT0006: Under
2015-08-19 12:10:31,512 WARNING [org.jgroups.protocols.UDP] (MSC service thread 1-4) JGRP000015: the rece
2015-08-19 12:10:31,513 WARNING [org.jgroups.protocols.UDP] (MSC service thread 1-4) JGRP000015: the rece
2015-08-19 12:10:31,516 INFO  [stdout] (MSC service thread 1-4)
2015-08-19 12:10:31,516 INFO  [stdout] (MSC service thread 1-4) ------------------------------------------
2015-08-19 12:10:31,516 INFO  [stdout] (MSC service thread 1-4) GMS: address=booker-web-client, cluster=s
2015-08-19 12:10:31,516 INFO  [stdout] (MSC service thread 1-4) ------------------------------------------
2015-08-19 12:10:31,522 WARNING [org.jgroups.protocols.UDP] (TransferQueueBundler,swarm-clustering,booker
2015-08-19 12:10:34,630 INFO  [org.jboss.as] (Controller Boot Thread) WFLYSRV0025: WildFly Core 2.0.0.Bet
2015-08-19 12:10:35,221 INFO  [org.jboss.as.server.deployment] (MSC service thread 1-5) WFLYSRV0027: Star
2015-08-19 12:10:36,300 WARN  [org.jboss.as.ee] (MSC service thread 1-8) WFLYEE0007: Not installing optio
2015-08-19 12:10:36,353 INFO  [org.wildfly.extension.undertow] (MSC service thread 1-7) WFLYUT0018: Host
2015-08-19 12:10:36,430 INFO  [org.wildfly.extension.undertow] (ServerService Thread Pool -- 10) WFLYUT00
2015-08-19 12:10:36,446 INFO  [org.jboss.as.server] (main) WFLYSRV0010: Deployed "booker-web-client.war"
```

http://localhost:8080/

But nothing *actually* works.

Store

Store

- JAX-RS **`StoreResource`**
- CDI-injected **`Store`**
- We provide a **`main(…)`**
- Uses Ribbon

Store

```
<dependency>
  <groupId>org.wildfly.swarm</groupId>
  <artifactId>wildfly-swarm-jaxrs-weld</artifactId>
</dependency>
```

Store

```java
Container container = new Container();

JAXRSArchive deployment = ShrinkWrap.create(JAXRSArchive.class);

deployment.addPackage(Main.class.getPackage());

deployment.as(RibbonArchive.class).setApplicationName("store");

deployment.as(Secured.class);

deployment.addAllDependencies();

container.start();

container.deploy(deployment);
```

Store

```
@Path("/")
public class StoreResource {

    @Inject
    private Store store;

    @Inject
    private PricingService pricingService;

    @GET
    @Path("/search")
    @Produces("application/json")
    public Store.SearchResult search(@QueryParam("q") String q,
                                     @QueryParam("page") Integer page) {
        . . .
    }

    @GET
    @Path("/book")
    @Produces("application/json")
    public void get(@Suspended final AsyncResponse asyncResponse,
                    @QueryParam("id") String id) {
        . . .
    }
}
```

StoreResource

```
@Inject
private Store store;
```

Store

```
@ApplicationScoped
public class Store {

    public Book get(String id) {
        ...
    }

    public SearchResult search(String q, int page) {
        ...
    }

}
```

Store

- Our `main(…)` builds the deployment
- Since we have `wildfly-swarm-jaxrs-weld` we
    - …don’t have to provide an `Application`
    - …can use CDI

Stores have prices, though…

NetflixOSS Ribbon

Ribbon

- Client-side load-balancing to other services
- Reference service by name, not host
- Allows Java interface for invoking service

Ribbon

```java
@ResourceGroup(name="pricing")
public interface PricingService {

    @TemplateName("get")
    @Http(
            method = Http.HttpMethod.GET,
            uri = "/book/{id}"
    )
    RibbonRequest<ByteBuf> get(@Var("id") String id);
}
```

Ribbon

```
pricingService.get( “42” )
```

$\downarrow$

```
GET http://${HOST}/book/42
```

Store

```
@ApplicationScoped
public class ServicesFactory {

    @Produces
    @ApplicationScoped
    public static PricingService getInstance() {
        return SecuredRibbon.from(PricingService.class);
    }
}
```

`SecuredRibbon`

- Provided by WildFly Swarm
- Emulates original Ribbon class
- Has exactly 1 method: `.from(someInterface)`
- Wires up the interface with WildFly Clustering to know where services live
- Propagates Keycloak authentication tokens to the invocation

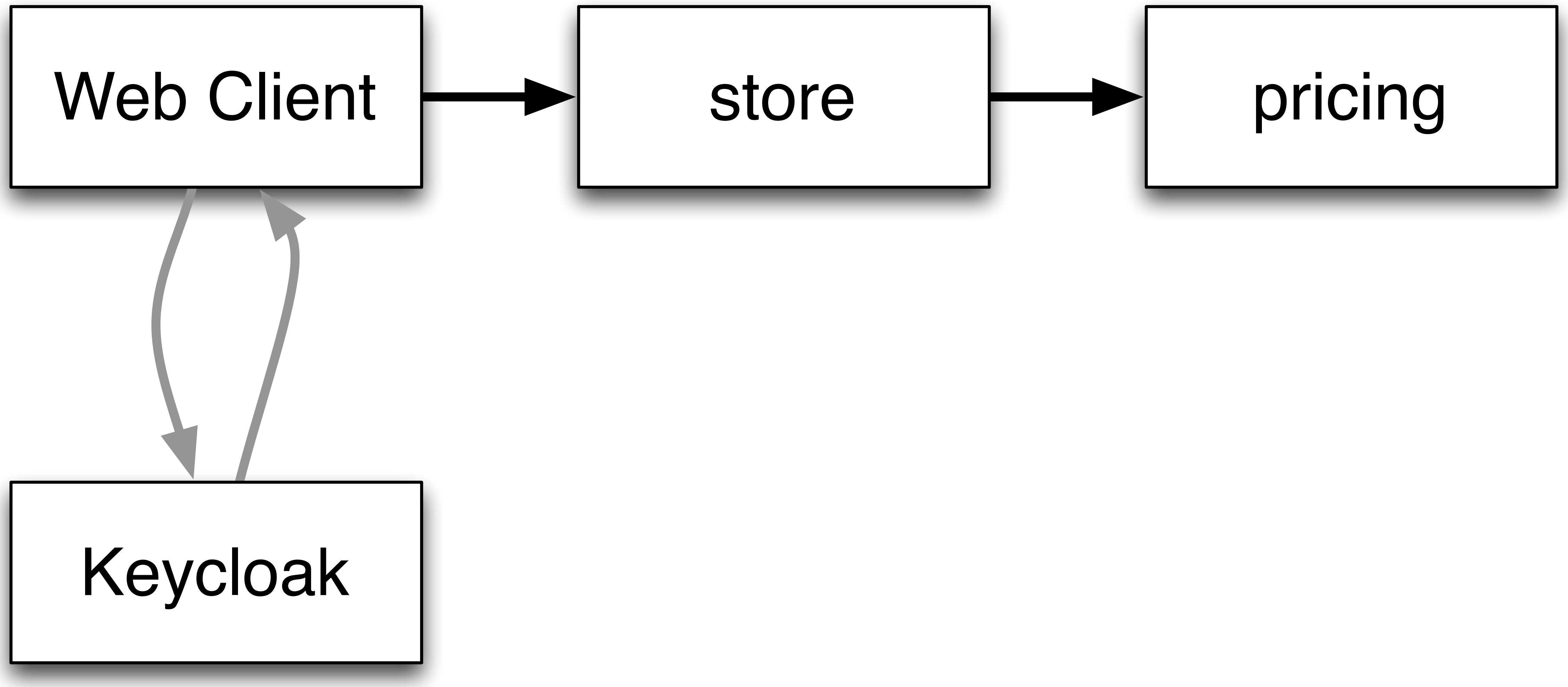
Pricing

Pricing

- Service just to demonstrate Ribbon + Keycloak
- If a request is un-authenticated, returns $10
- If a request is authenticated, returns $9

Pricing

```java
@Path("/")
public class PricingResource {

    @GET
    @Path("/book/{id}")
    @Produces("application/json")
    public Integer search(@PathParam("id") String id, @Context SecurityContext context) {
        KeycloakPrincipal principal = (KeycloakPrincipal) context.getUserPrincipal();
        if ( principal != null && principal.getKeycloakSecurityContext() != null ) {
            return 9;
        }
        return 10;
    }
}
```


Web Client
store
pricing
Keycloak

Ribbon Requests

```
Observable<ByteBuf> obs = service.method(params).observe();

obs.subscribe(
        (result) -> {
            // result is a ByteBuf
            . . .
        },
        (err) -> {
            // err is an Exception
            . . .
        }
);
```

Ribbon Requests

```
@GET
@Path("/book")
@Produces("application/json")
public void get(@Suspended final AsyncResponse asyncResponse,
                @QueryParam("id") String id) {

    Book book = this.store.get(id);
    Observable<ByteBuf> obs = pricingService.get(id).observe();
    obs.subscribe(
            (result) -> {
                int price = Integer.parseInt(result.toString(UTF8));
                book.setPrice(price);
                asyncResponse.resume(book);
            },
            (err) -> {
                asyncResponse.resume(err);
            }
    );
}
```

Back to Keycloak…

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'));
  });
})
```

Keycloak

- We `check-sso` when we initialize Keycloak
- Bounces user to Keycloak to see if they are auth’d

Secure Requests from `.js`

```
if ( keycloak.token ) {
  headers.Authorization = 'Bearer ' + keycloak.token;
}
```

Ribbon from `.js`

- Ribbon in Java uses WildFly Clustering to wire together services.
- What about Javascript?

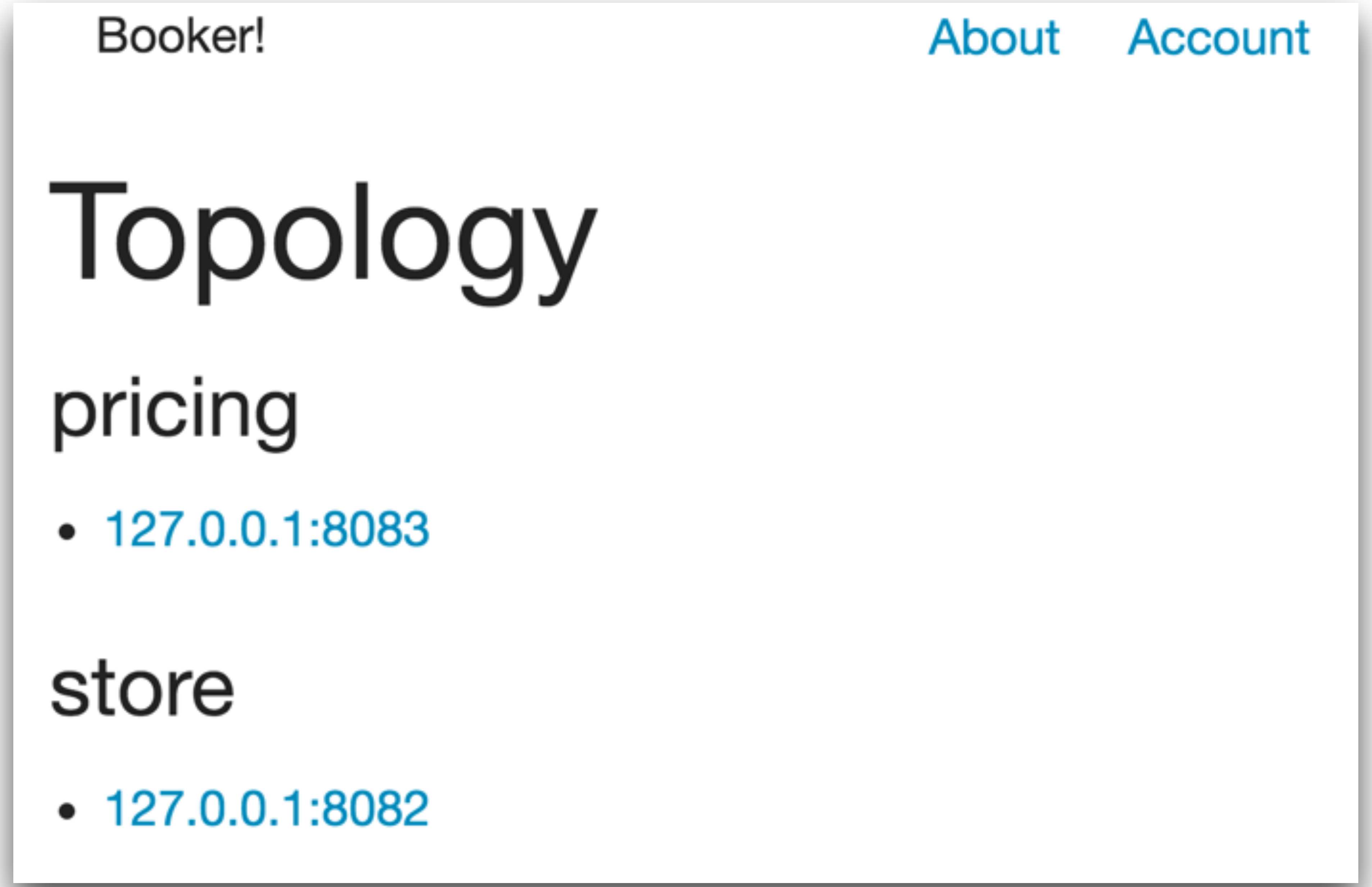
RibbonToTheCurb

- An async servlet in the `web-client` WAR
- Serves Ribbon topology information to the single-page-app via Server-Sent-Events (SSE)

RibbonToTheCurb

```
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();
    }
};
```

http://localhost:8080/topology



Topology

pricing

- 127.0.0.1:8083

store

- 127.0.0.1:8082

RibbonToTheCurb

- Service end-points don’t have to be encoded into the Javascript
- If end-points move, go up, go down, that’s okay
- Topology changes are *immediately* known by the web clients

ribbon.js

```
Ribbon.ajax = function(serviceName, url, settings) {
  var allServers = Booker.State.Topology.servers( serviceName );

  if ( ! settings ) {
    settings = {};
  }

  var headers = settings.headers || {};

  if ( keycloak.token ) {
    headers.Authorization = 'Bearer ' + keycloak.token
  }

  settings.url = '//' + allServers[0] + url;
  settings.headers = headers;

  if ( allServers.length > 0 ) {
    return $.ajax( settings );
  }

  var deferred = $.Deferred();
  return deferred.reject();
}
```

ribbon.js

- Uses jQuery to perform AJAX requests
- Mixes in Keycloak bearer token (if present)
- Selects server from list provided by Ribbon
- Works with the promises API (from jQuery)

Library

Library

- JAX-RS Service
- Requires authentication
- JPA to store user/book associations for purchases
- Invokes the `store` service for book titles

Library

```
Container container = new Container();

container.fraction(new JPAFraction()
        .inhibitDefaultDatasource()
        .defaultDatasourceName(“LibraryDS"));

container.fraction(new DatasourcesFraction()
        .driver(new Driver("h2")
                .datasourceClassName("org.h2.Driver")
                .xaDatasourceClassName("org.h2.jdbcx.JdbcDataSource")
                .module("com.h2database.h2"))
        .datasource(new Datasource("LibraryDS")
                .driver("h2")
                .connectionURL("jdbc:h2:./library;DB_CLOSE_ON_EXIT=TRUE")
                .authentication("sa", "sa")));
```

Library

```
JAXRSArchive deployment = ShrinkWrap.create(JAXRSArchive.class);
deployment.addPackage(Main.class.getPackage());
deployment.as(RibbonArchive.class).setApplicationName("library");
deployment.as(Secured.class)
        .protect("/items")
        .withMethod("GET")
        .withRole("*");
```

Library

```
deployment.add(new ClassLoaderAsset("META-INF/persistence.xml", Main.class.getClassLoader()),
               “WEB-INF/classes/META-INF/persistence.xml");
```

```
<?xml version="1.0" encoding="UTF-8"?>
<persistence
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    version="2.1"
    xmlns="http://xmlns.jcp.org/xml/ns/persistence"
    xsi:schemaLocation="http://xmlns.jcp.org/xml/ns/persistence
http://xmlns.jcp.org/xml/ns/persistence/persistence_2_1.xsd">
  <persistence-unit name="LibraryPU" transaction-type="JTA">
    <properties>
   </properties>
  </persistence-unit>
</persistence>
```

Library

```
@Entity
@Table(name="LibraryItem")
public class LibraryItem {

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

    @Column private String userId;
    @Column private String bookId;

    @Transient private String title;
    @Transient private String author;

    public LibraryItem() {

    }

    public LibraryItem(String userId, String bookId) {
        this.userId = userId;
        this.bookId = bookId;
    }

    . . .
}
```

Library

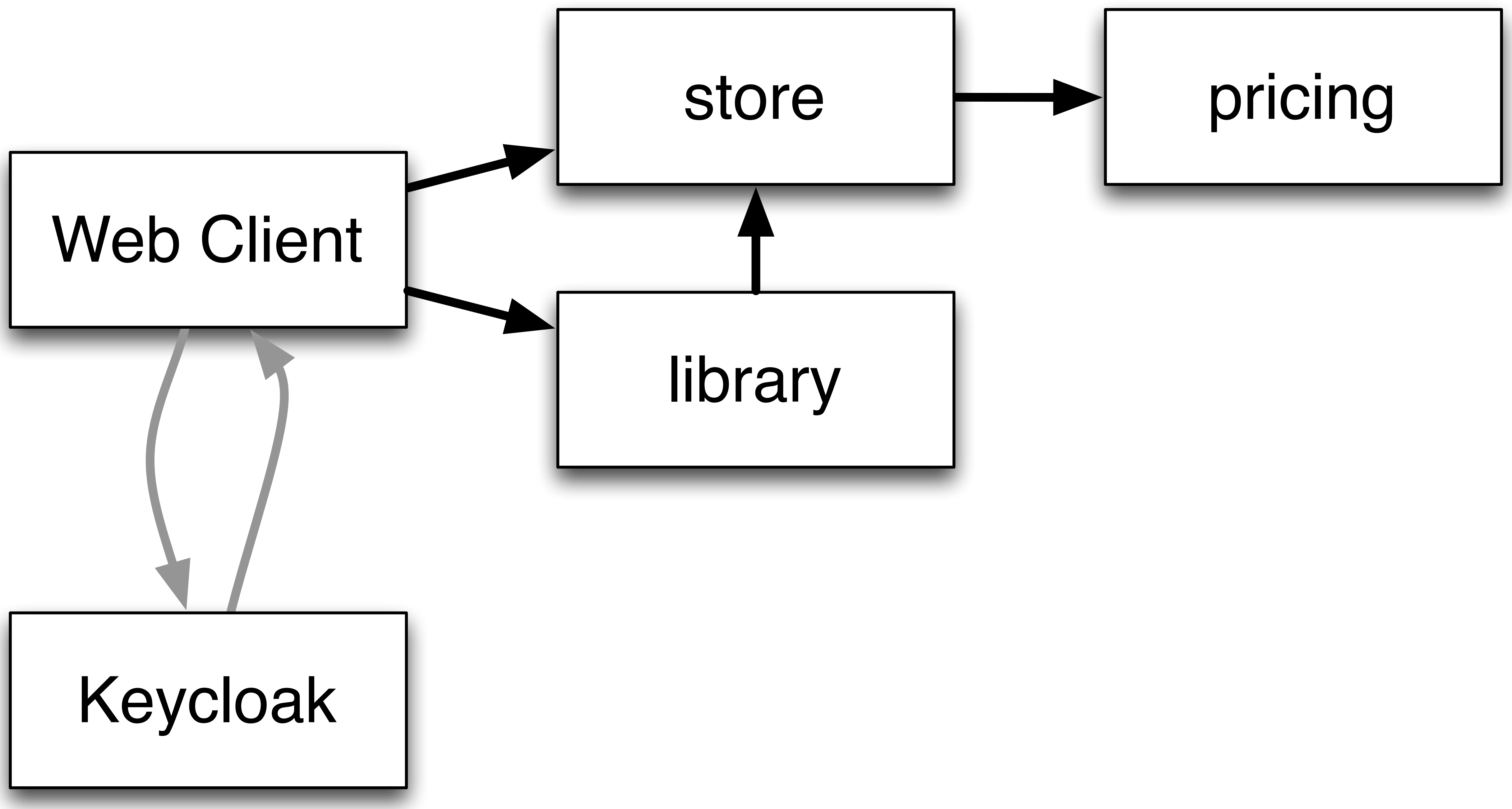
```
@Path("/")
@Stateless
public class LibraryResource {

    @Inject
    StoreService store;

    @Inject
    EntityManager em;

    @GET
    @Produces("application/json")
    @Path("/items")
    public void get(@Suspended final AsyncResponse asyncResponse,
                    @Context SecurityContext context) {
        . . .
    }

    @POST
    @Produces("application/json")
    @Path("/items")
    public LibraryItem addItem(@Context SecurityContext context,
                               @FormParam("id") String bookId) throws URISyntaxException {
        . . .
    }
}
```

Library

```
@POST
@Produces("application/json")
@Path("/items")
public LibraryItem addItem(@Context SecurityContext context,
                           @FormParam("id") String bookId) throws URISyntaxException {
    KeycloakPrincipal principal = (KeycloakPrincipal) context.getUserPrincipal();
    String userId = principal.getName();
    LibraryItem item = new LibraryItem(userId, bookId);
    em.persist(item);
    return item;
}
```

But that’s only

`userId+bookId`


store
pricing
Web Client
library
Keycloak

Library

```
@ResourceGroup(name = "store")
public interface StoreService {

    @TemplateName("get")
    @Http(
            method = Http.HttpMethod.GET,
            uri = "/book?id={bookId}"
    )
    RibbonRequest<ByteBuf> get(@Var("bookId")String bookId);

}
```

Library

```
@ApplicationScoped
public class ServicesFactory {

    @Produces
    @ApplicationScoped
    public static StoreService getInstance() {
        return SecuredRibbon.from(StoreService.class);
    }
}
```

Library

```
@GET
@Produces("application/json")
@Path("/items")
public void get(@Suspended final AsyncResponse asyncResponse,
                @Context SecurityContext context) {
    KeycloakPrincipal principal = (KeycloakPrincipal)
context.getUserPrincipal();
    String userId = principal.getName();
    TypedQuery<LibraryItem> q = this.em.createQuery(
        "SELECT li FROM LibraryItem li WHERE li.userId = :userId”,
        LibraryItem.class);
    List<LibraryItem> items = q.setParameter("userId", userId)
        .getResultList();
```

How do we populate titles in that list?

Library

```
for (LibraryItem each : items) {
    Observable<ByteBuf> obs = store.get(each.getBookId()).observe();
    root = root.zipWith(obs, new Func2<List<LibraryItem>, ByteBuf, List<LibraryItem>>() {
        @Override
        public List<LibraryItem> call(List<LibraryItem> libraryItems, ByteBuf byteBuf) {
            ObjectMapper mapper = new ObjectMapper();
            ObjectReader reader = mapper.reader();
            JsonFactory factory = new JsonFactory();
            try {
                JsonParser parser = factory.createParser(new ByteBufInputStream(byteBuf));
                Map map = reader.readValue(parser, Map.class);
                each.setTitle((String) map.get("title"));
                each.setAuthor((String) map.get("author"));
            } catch (IOException e) {
            }
            libraryItems.add(each);
            return libraryItems;
        }
    });
}
```

Fire off Requests

```
for (LibraryItem each : items) {
    Observable<ByteBuf> obs = store.get(each.getBookId()).observe();
    . . .
}
```

Zip them together

```java
Observable<List<LibraryItem>> root = Observable.just(new ArrayList<>());
. . .

for (LibraryItem each : items) {
    Observable<ByteBuf> obs = store.get(each.getBookId()).observe();
    root = root.zipWith(obs, ((libraryItems, byteBuf) -> {
        ObjectMapper mapper = new ObjectMapper();
        ObjectReader reader = mapper.reader();
        JsonFactory factory = new JsonFactory();
        try {
            JsonParser parser = factory.createParser(new ByteBufInputStream(byteBuf));
            Map map = reader.readValue(parser, Map.class);
            each.setTitle((String) map.get("title"));
            each.setAuthor((String) map.get("author"));
        } catch (IOException e) {
        }
        libraryItems.add(each);
        return libraryItems;
    }));
}
```

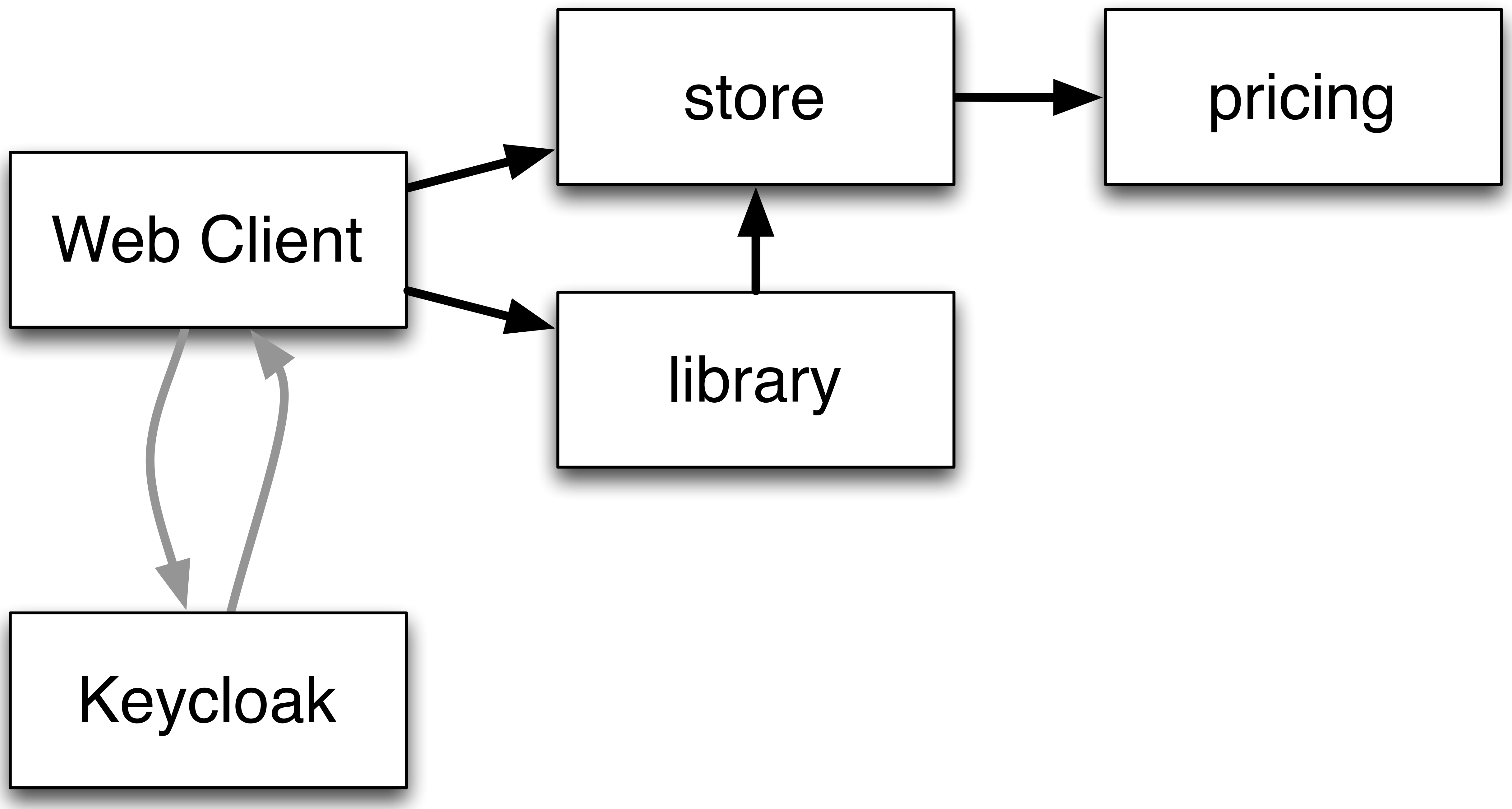
Process each response

```
Observable<List<LibraryItem>> root = Observable.just(new ArrayList<>());
. . .

for (LibraryItem each : items) {
    Observable<ByteBuf> obs = store.get(each.getBookId()).observe();
    root = root.zipWith(obs, ((libraryItems, byteBuf) -> {
        ObjectMapper mapper = new ObjectMapper();
        ObjectReader reader = mapper.reader();
        JsonFactory factory = new JsonFactory();
        try {
            JsonParser parser = factory.createParser(new ByteBufInputStream(byteBuf));
            Map map = reader.readValue(parser, Map.class);
            each.setTitle((String) map.get("title"));
            each.setAuthor((String) map.get("author"));
        } catch (IOException e) {
        }
        libraryItems.add(each);
        return libraryItems;
    }));
}
```

When all are complete

```
root.subscribe(
        (result) -> asyncResponse.resume(result),
        (err) -> asyncResponse.resume(err)
);
```

Let’s Review…


store
pricing
Web Client
library
Keycloak

Review

- `web-client`
    - simple .war
    - static assets
    - `RibbonToTheCurbSSEServlet`
- `store`
    - simple main(…)
    - JAX-RS
    - no security
    - searchable list of books/authors
    - components injected with CDI
    - `SecureRibbon` to pricing

Review

- **`pricing`**
  - simple `main(…)`
  - JAX-RS
  - optional security
- **`library`**
  - complex `main(…)`
  - JAX-RS
  - required security
  - JPA
  - CDI
  - **`SecureRibbon`** to store

Logstash

- 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

Logstash

- https://download.elastic.co/logstash/logstash/logstash-1.5.3.zip
- https://www.elastic.co/downloads/kibana

logstash-wildfly.conf

```
input {
  tcp {
    port => 8000
  }
}

filter {
  json {
    source => "message"
  }
}

output {
  elasticsearch {
    # Use the embedded elasticsearch for convienence
    embedded => true
    protocol => "http"
  }
}
```

```
./bin/logstash agent -f logstash-wildfly.conf

./bin/kibana
```

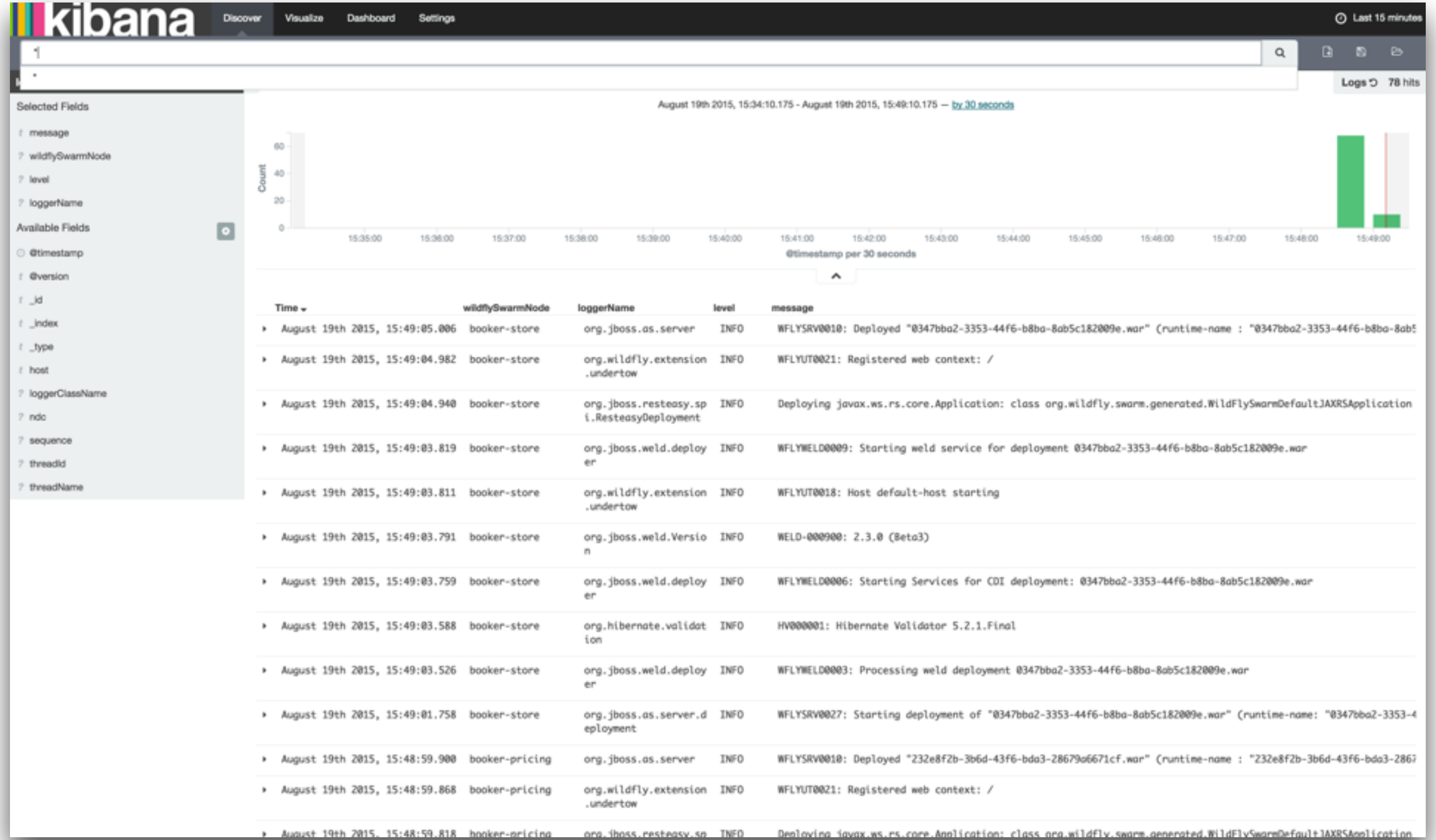
http://localhost:5601

Logstash has been configured for all services this entire time!

```
<!–

<swarm.logstash.hostname>localhost</swarm.logstash.hostname>
<swarm.logstash.port>8000</swarm.logstash.port>

-->
```

```
mvn wildfly-swarm:run
```

http://localhost:5601/



Time	wildflySwarmNode	loggerName	level	message
August 19th 2015, 15:49:05.006	booker-store	org.jboss.as.server	INFO	WFLYSRV0010: Deployed "0347bba2-3353-44f6-b8ba-8ab5c182009e.war" (runtime-name : "0347bba2-3353-44f6-b8ba-8ab5
August 19th 2015, 15:49:04.982	booker-store	org.wildfly.extension.undertow	INFO	WFLYUT0021: Registered web context: /
August 19th 2015, 15:49:04.940	booker-store	org.jboss.resteasy.spi.ResteasyDeployment	INFO	Deploying javax.ws.rs.core.Application: class org.wildfly.swarm.generated.WildFlySwarmDefaultJAXRSApplication
August 19th 2015, 15:49:03.819	booker-store	org.jboss.weld.deployer	INFO	WFLYWELD0009: Starting weld service for deployment 0347bba2-3353-44f6-b8ba-8ab5c182009e.war
August 19th 2015, 15:49:03.811	booker-store	org.wildfly.extension.undertow	INFO	WFLYUT0018: Host default-host starting
August 19th 2015, 15:49:03.791	booker-store	org.jboss.weld.Version	INFO	WELD-000900: 2.3.0 (Beta3)
August 19th 2015, 15:49:03.759	booker-store	org.jboss.weld.deployer	INFO	WFLYWELD0006: Starting Services for CDI deployment: 0347bba2-3353-44f6-b8ba-8ab5c182009e.war
August 19th 2015, 15:49:03.588	booker-store	org.hibernate.validation	INFO	HV000001: Hibernate Validator 5.2.1.Final
August 19th 2015, 15:49:03.526	booker-store	org.jboss.weld.deployer	INFO	WFLYWELD0003: Processing weld deployment 0347bba2-3353-44f6-b8ba-8ab5c182009e.war
August 19th 2015, 15:49:01.758	booker-store	org.jboss.as.server.deployment	INFO	WFLYSRV0027: Starting deployment of "0347bba2-3353-44f6-b8ba-8ab5c182009e.war" (runtime-name: "0347bba2-3353-4
August 19th 2015, 15:48:59.900	booker-pricing	org.jboss.as.server	INFO	WFLYSRV0010: Deployed "232e8f2b-3b6d-43f6-bda3-28679a6671cf.war" (runtime-name : "232e8f2b-3b6d-43f6-bda3-2867
August 19th 2015, 15:48:59.868	booker-pricing	org.wildfly.extension.undertow	INFO	WFLYUT0021: Registered web context: /
August 19th 2015, 15:48:59.818	booker-pricing	org.jboss.resteasy.sp	INFO	Deploying javax.ws.rs.core.Application: class org.wildfly.swarm.generated.WildFlySwarmDefaultJAXRSApplication

More stuff!

wildfly-swarm-maven-plugin

- **`mvn wildfly-swarm:run`**
  - Just runs your project
- **`mvn wildfly-swarm:package`**
  - creates the fatjar

```
java -jar booker-store-swarm.jar
```

Arquillian

```
@RunWith(Arquillian.class)
public class MyTest {

    @Deployment
    public static Archive createDeployment() {
       . . . create archive
    }

    @Test @RunAsClient
    public void testOutside() {
        . . . random tests against the deployment
    }

    @Test
    public void testInside() {
        . . . can @Inject
    }

}
```

What more would you like to discuss?