OSGi R7 upcoming specifications

David Bosschaert & Carsten Ziegeler | Adobe

Bē
Craig Ward

Carsten Ziegeler

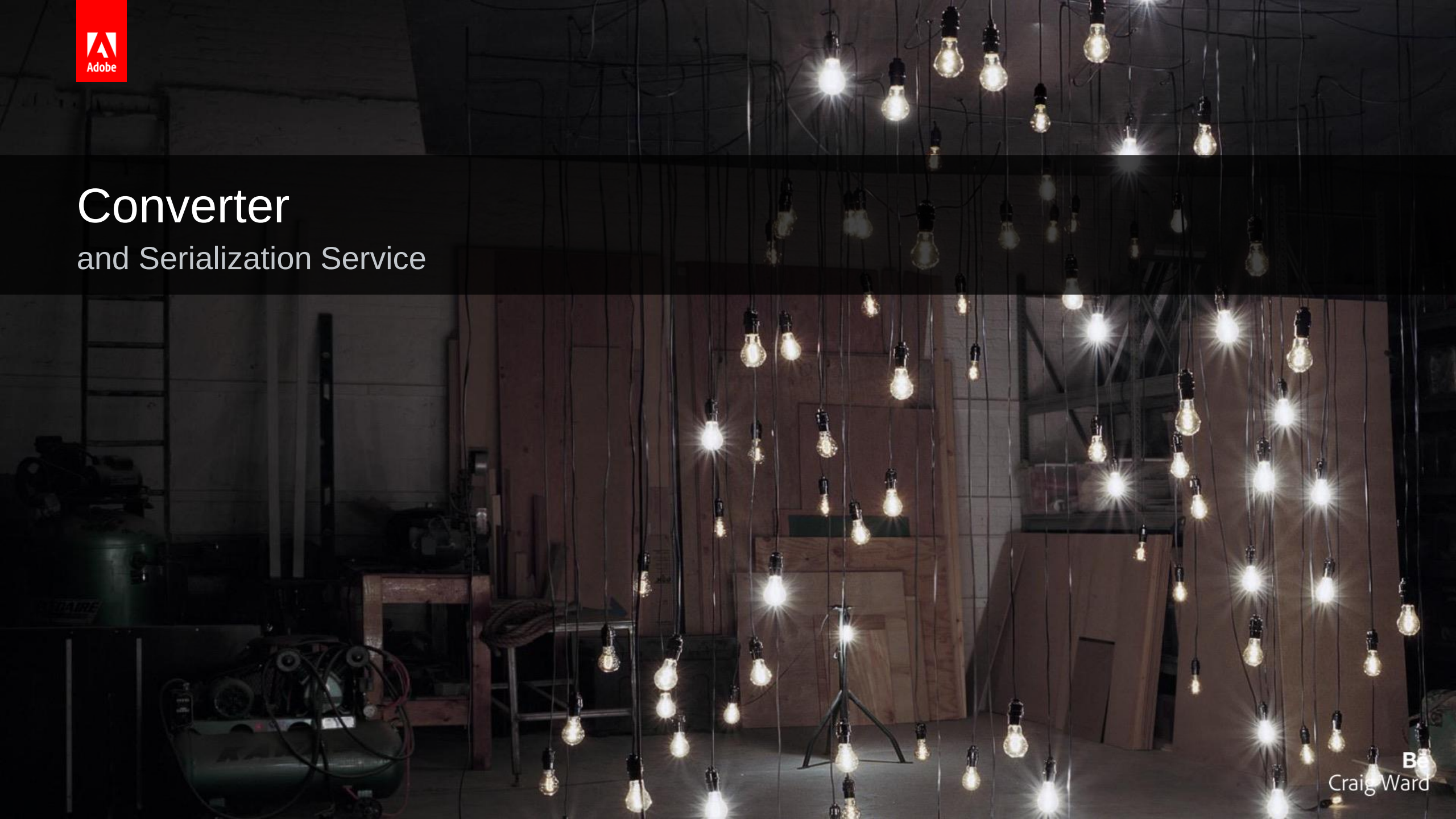
- RnD Adobe Research Switzerland
- Team Lead / Founder of Adobe Granite
- Member of the Apache Software Foundation
- VP of Apache Felix and Sling
- OSGi Expert Groups and Board member

David Bosschaert

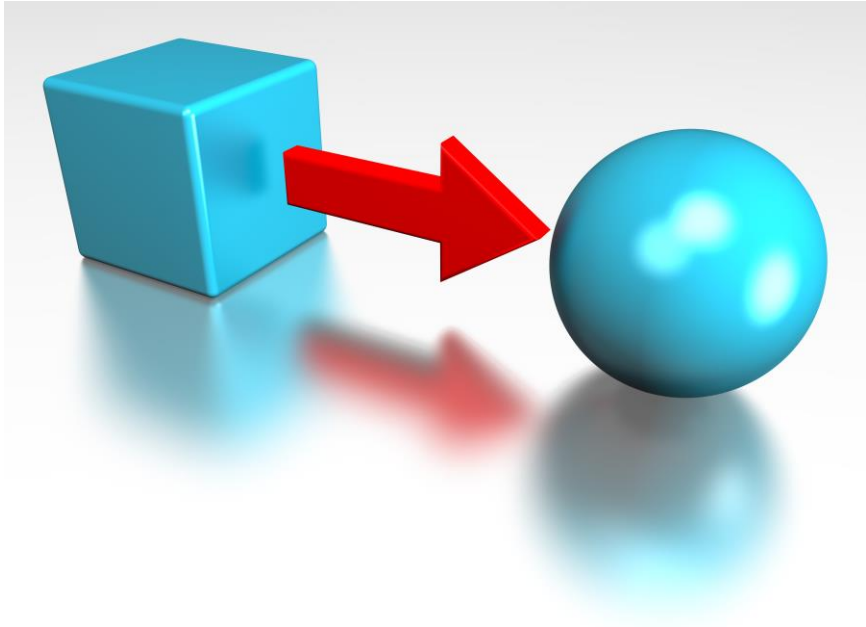
- Adobe R&D Ireland
- OSGi EEG co-chair
- Apache member and committer
- ISO JTC1 SC38 (Cloud Computing) Ireland committee member

OSGi Specification Process

- Requirements, Use Cases -> RFP
- Technical Specification -> RFC
  - https://github.com/osgi/design
- Reference Implementation
- Compatibility Testsuite
- Spec Chapter


Adobe
Converter
and Serialization Service
Craig Ward

Object Conversion



- Wouldn’t you like to convert anything to everything?
- Peter Kriens’ work in Bnd started it

- Convert
  - Scalars, Collections, Arrays
  - Interfaces, maps, DTOs, JavaBeans, Annotations

- Need predictable behaviour – as little implementation freedom as possible
  - Can be customized

RFC 215 – Implementation in Apache Felix (/converter)

Sample conversions

- Examples:

```
Converter c = new StandardConverter();

// Convert scalars
int i = c.convert("123").to(int.class);
UUID id = c.convert("067e6162-3b6f-4ae2-a171-2470b63dff00").to(UUID.class);

List<String> ls = Arrays.asList("978", "142", "-99");
short[] la = c.convert(ls).to(short[].class);
```

Convert configuration data into typed information with defaults

```
Map<String, String> myMap = new HashMap<>();
myMap.put("refresh", "750");
myMap.put("other", "hello");


// Convert map structures
@interface MyAnnotation {
  int refresh() default 500;
  String temp() default "/tmp";
}
MyAnnotation myAnn = converter.convert(someMap).to(MyAnnotation.class)


int refresh = myAnn.refresh();  // 750
Strine temp = myAnn.temp();     // "/tmp"
```

Customize your converter

The converter service itself always behaves the same

- Need customizations? Create a new converter: ConverterBuilder

- Change default behaviour of array to scalar conversion
- Convert char[] to String via adapter

```
// Default behaviour
String s = converter.convert(new char[] {‘h’, ‘i’}).to(String.class); // s = “h”

Converter c = converter.newConverterBuilder()
  .rule(char[].class, String.class, MyCnv::caToString, MyCnv::stringToCA)
  .build();
String s2 = c.convert(new char[] {‘h’, ‘i’}).to(String.class); // ss=“hi”
```

Serialization service

- Take an object, turn it into a file/stream, and back
- with the help of the converter
- YAML, JSON, or your own format
- Serializations can be customized with the converter

Example JSON Serializer

```
@Reference(target="(mimetype=application/json)")
Serializer jsonSerializer;

public void someMethod() {
  Map m = … some map …;
  String s = jsonSerializer.serialize(m).toString();

  MyBean someBean  = … some Java Bean …;
  String s2 = jsonSerializer.serializer(someBean).toString();

  // and deserialize back
  MyBean recreated = jsonSerializer.deserialize(MyBean.class).from(s2);
```

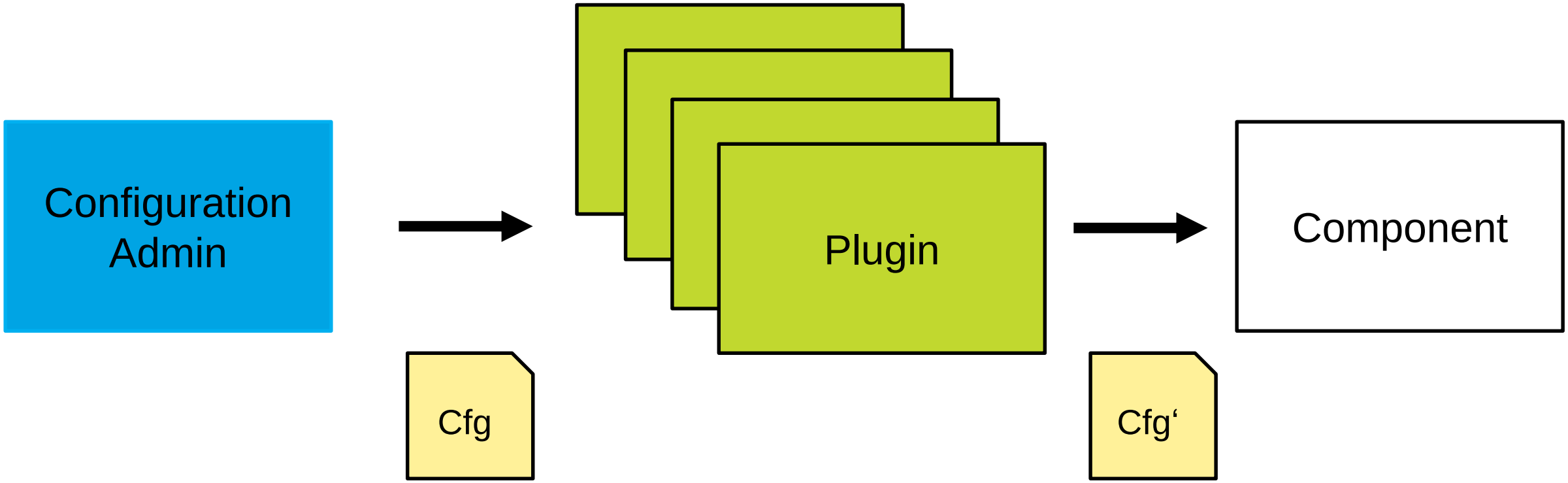
```
{"prop1": "val1",
 "prop2": 42,
 "nested":
  {"really": true}
}
```

Config Admin and Configurator

OSGi Configurations

- Configuration Admin Improvements
  - Alias for factory configurations
  - Optimized change handling
  - Revived ConfigurationPlugin

Configuration Admin Plugins



Reusable Configuration Format I

```
configurations:
  - my.special.component:
      some_prop: 42
      and_another: "some string"

  - and.a.factory.cmp#foo
    ...
  - and.a.factory.cmp#bar
    ...
```

Reusable Configuration Format II

```
configurations:
  - my.special.component:
      some_prop: 42
      and_another: "some string"
      :configurator:environments:
        - dev
        - test
```

OSGi Configurator

- Configurations as bundle resources
- Environment support
- Binaries
- State handling and ranking

Push Streams

Push Streams

Like Java 8 streams, but data is pushed

For event-based data, which may or may not be infinite

- Async processing and buffering
- Supports back pressure
- Mapping, filtering, flat-mapping
- Coalescing and windowing
- Merging and splitting

Example:

- Humidity reader/processor stream
  - sends an alarm when over 90%

RFC 216 – Implementation will be in Apache Aries

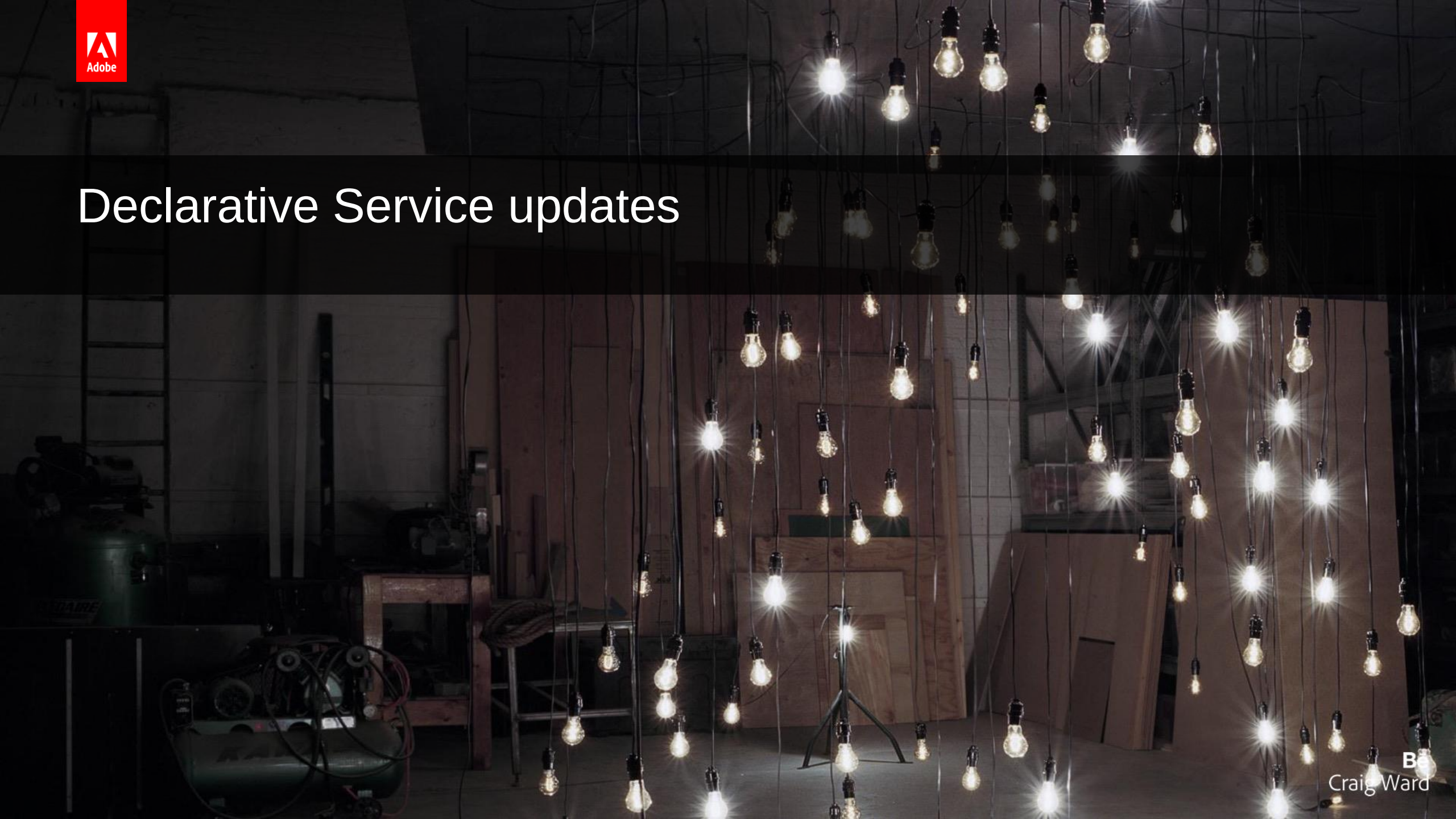
Asynchronous Push Streams example

```
PushStreamProvider psp = new PushStreamProvider();

try (SimplePushEventSource<Long> ses = psp.createSimpleEventSource(Long.class)) {
  ses.connectPromise().then(p -> {
    long counter = 0;
    while (counter < Long.MAX_VALUE && ses.isConnected()) {
      ses.publish(++counter);
      Thread.sleep(100);
      System.out.println("Published: " + counter);
    }
    return null;
  });

  psp.createStream(ses).
    filter(l -> l % 2L == 0).
    forEach(f -> System.out.println("Consumed even: " + f));
```

Declarative Service updates



Declarative Service Updates

- Field Injection of activation objects
- Constructor injection

Activation Objects

```
@Component
public class MyComponent {

        public @interface Config {
                int easy_max() default 10;

                int medium_max() default 50;

                int hard_max() default 100;
        }

        @Activate
        private BundleContext bundleContext;

        @Activate
        private Config config;
```

Constructor Injection

```
@Component
public class MyComponent {

    @Activate
        public MyComponent(
            BundleContext bundleContext,

            @Reference EventAdmin eventAdmin,

            Config config,

            @Reference List<Receivers> receivers) {
```

Cluster Information

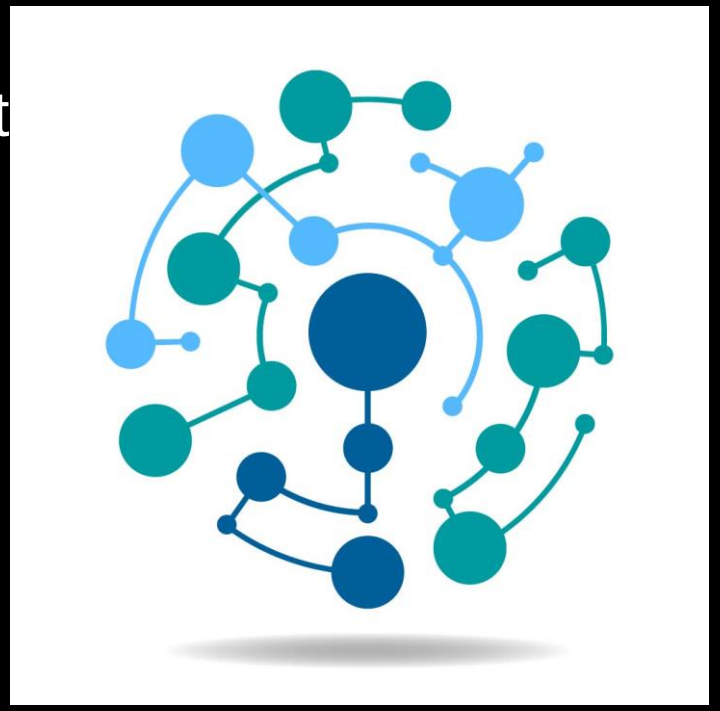
previously known as Cloud Ecosystems

Craig Ward

Cluster Information

OSGi frameworks running in a distributed environment

- Cloud
- IoT
- <buzzword xyz in the future>



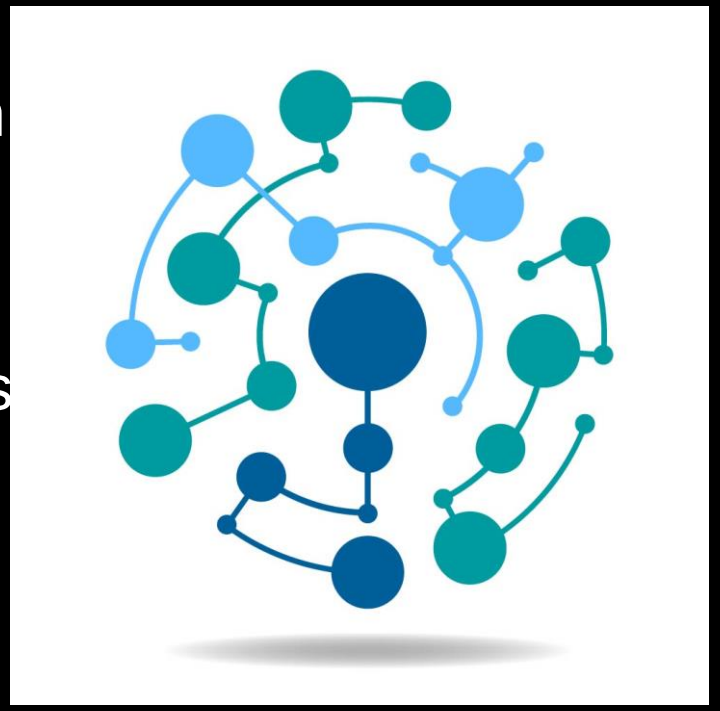
Many frameworks

- New ones appearing
- … and disappearing
- Other entities too, like databases

Cluster Information

Discovery of OSGi frameworks in a distributed environ

- Find out what is available
  - get notifications of changes
- Provision your application within the given resources
- Also: non-OSGi frameworks (e.g. Database)
- Monitor your application
  - Make changes if needed



RFC 183 – Implementation in Eclipse Concierge

Cluster Information

`FrameworkNodeStatus` service provides metadata

- Java version
- OSGi version
- OS version
- IP address
- Physical location (country, ISO 3166)
- Metrics
- tags / other / custom

- Provisioning API (install/start/stop/uninstall bundles etc…)

JAX-RS support

JAX-RS Services

- JAX-RS inOSGi
- Service / Whiteboard Model
- (Http Service)

Simple Resource

```
@Component(service = TestService.class,
        property = {
                "osgi.jaxrs.resource.base=/myapp"
        })
@Path("service")
public class TestService {

        @GET
        public String getHelloWorld() {
                return "Hello World";
        }
}
```

OSGi Enabled

```
@Component(service = TestService.class,
        property = {
                "osgi.jaxrs.resource.base=/myapp"
        })
@Path("service")
public class TestService {

    @Reference
        private GameController game;

        @Activate
        protected void activate(final Config config) {}

        @GET
        public String getHelloWorld() {
                return "Hello World";
        }
}
```

Parameters

```
@Component(service = TestService.class,
        property = {
                "osgi.jaxrs.resource.base=/myapp"
        })
@Path("service/{name}")
public class TestService {

        @GET
        public String getHelloWorld(@PathParam("name") String name) {
                return "Hello " + name;
        }
}
```

JAX-RS Support

- Get, Post, Delete with Parameters
- Application support
- Filters and Interceptors
- Request Scope vs Singleton

Additional OSGi R7 work

And much more...

- Service Decorations
- Http Whiteboard Update
- Transaction Control
- JPA Updates
- Remote Service Intents


Adobe