Apache Karaf in real life

ApacheCon NA 2014

APACHECON
DENVER
WESTIN DENVER DOWNTOWN
APRIL 7-9, 2014

APACHECON
DENVER
WESTIN DENVER DOWNTOWN
APRIL 7-9, 2014

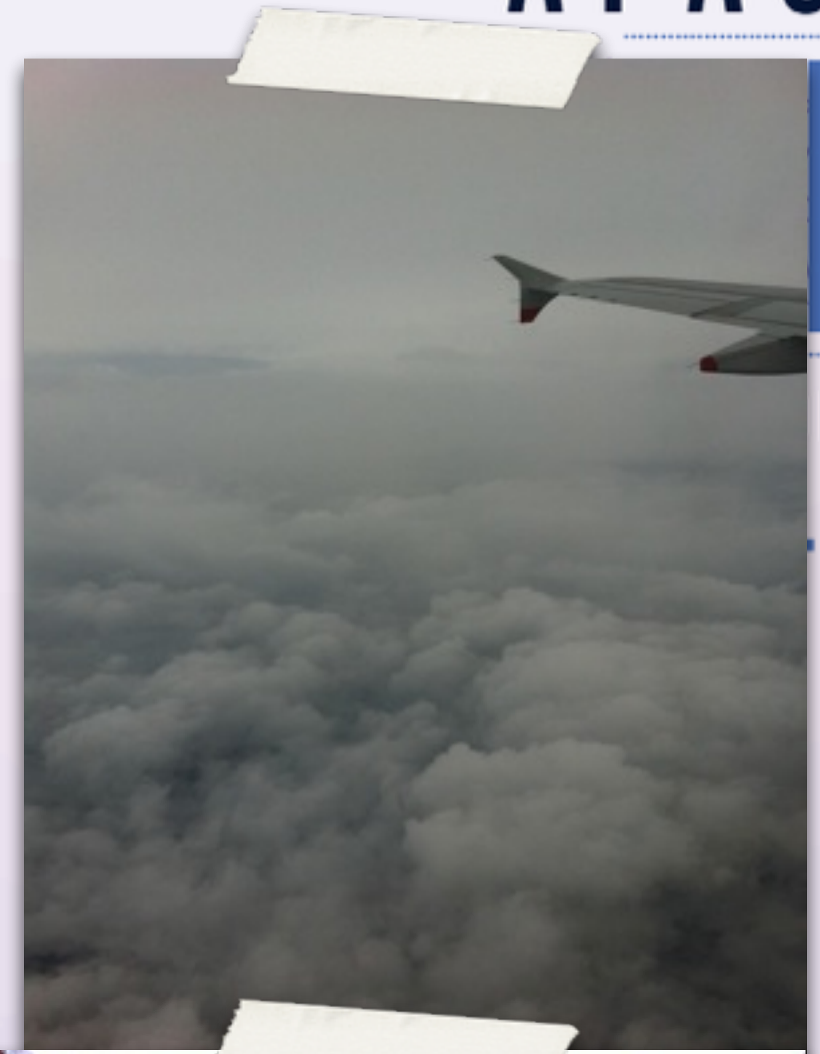
Agenda

APACHECON DENVER
WESTIN DENVER DOWNTOWN
APRIL 7-9, 2014

- Very short history of Karaf
- Karaf basis
- A bit deeper dive into OSGi
- Modularity vs Extensibility
- DIY - Karaf based solution
- What we have learned
- New and noteworthy in Karaf 3.0

About me …

- I'm Pole
- Husband
- Open source commiter & consultant
- Traveller
- .. and Construction worker :)



APACHECON DENVER
WESTIN DENVER DOWNTOWN
APRIL 7-9, 2014

Very short history of Apache Karaf

Karaf the beginning

- Born as ServiceMix 4 platform kernel in 2007
- Moved to Apache Felix community in 2009
- Top level project since June 2010

Karaf versions

- 1.x - legacy
- 2.1 - outdated
- 2.2.x - Based on OSGi 4.2
- 2.3.x - Based on OSGi 4.3
- 3.0 - Based on OSGi 5.0

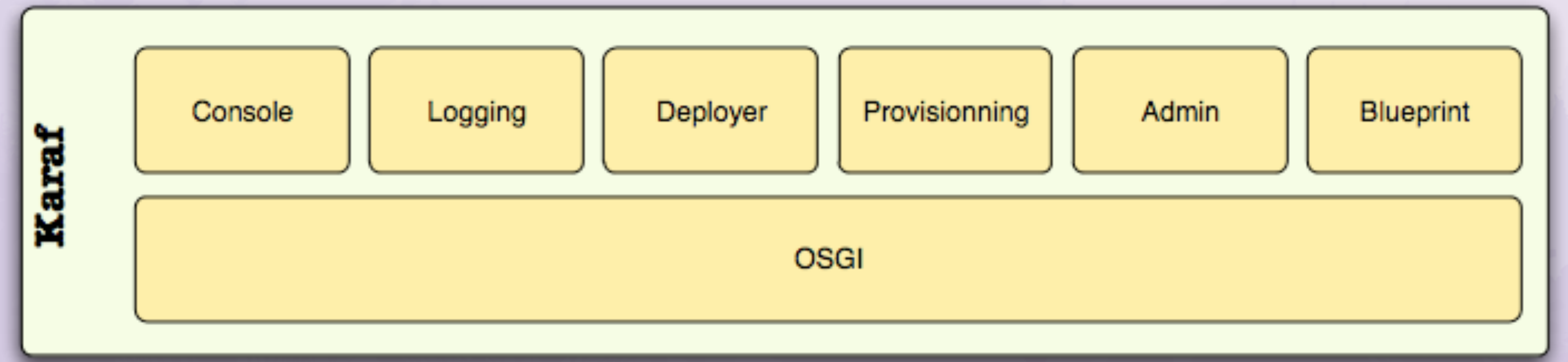
Karaf versions cont.

- 2.4.x - might support OSGi 5.0 frameworks and features
- 3.1 - future release for things delaying 3.0 release

APACHECON DENVER
WESTIN DENVER DOWNTOWN
APRIL 7-9, 2014

Karaf basis

What you already saw in most of presentations



Source: http://felix.apache.org/

What was told to you

Source: http://denise-theonepersonproject.blogspot.com/

That's what you can do

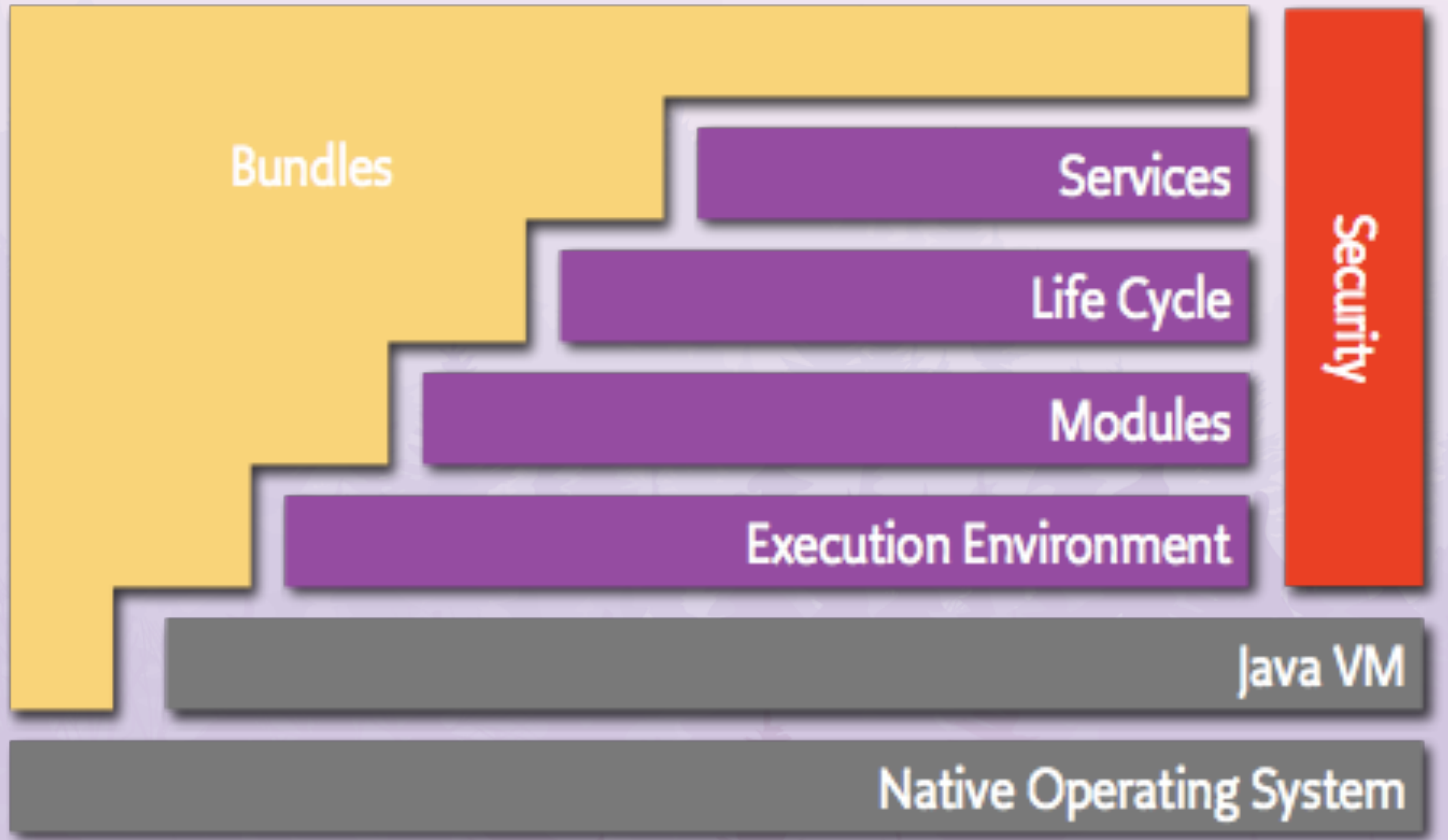
Source: http://freefoto.com/

Karaf grown to be a platform foundation

- Might run multiple applications at same time
- Each app may use different technologies
- Keeps things separate
- Has support for full product life cycle
- Failover & load balancing features
- Powerful infrastructure for extensions

Bit deeper dive into OSGi

OSGi



Source: http://osgi.org/

APACHECON DENVER
WESTIN DENVER DOWNTOWN
APRIL 7-9, 2014

In many cases

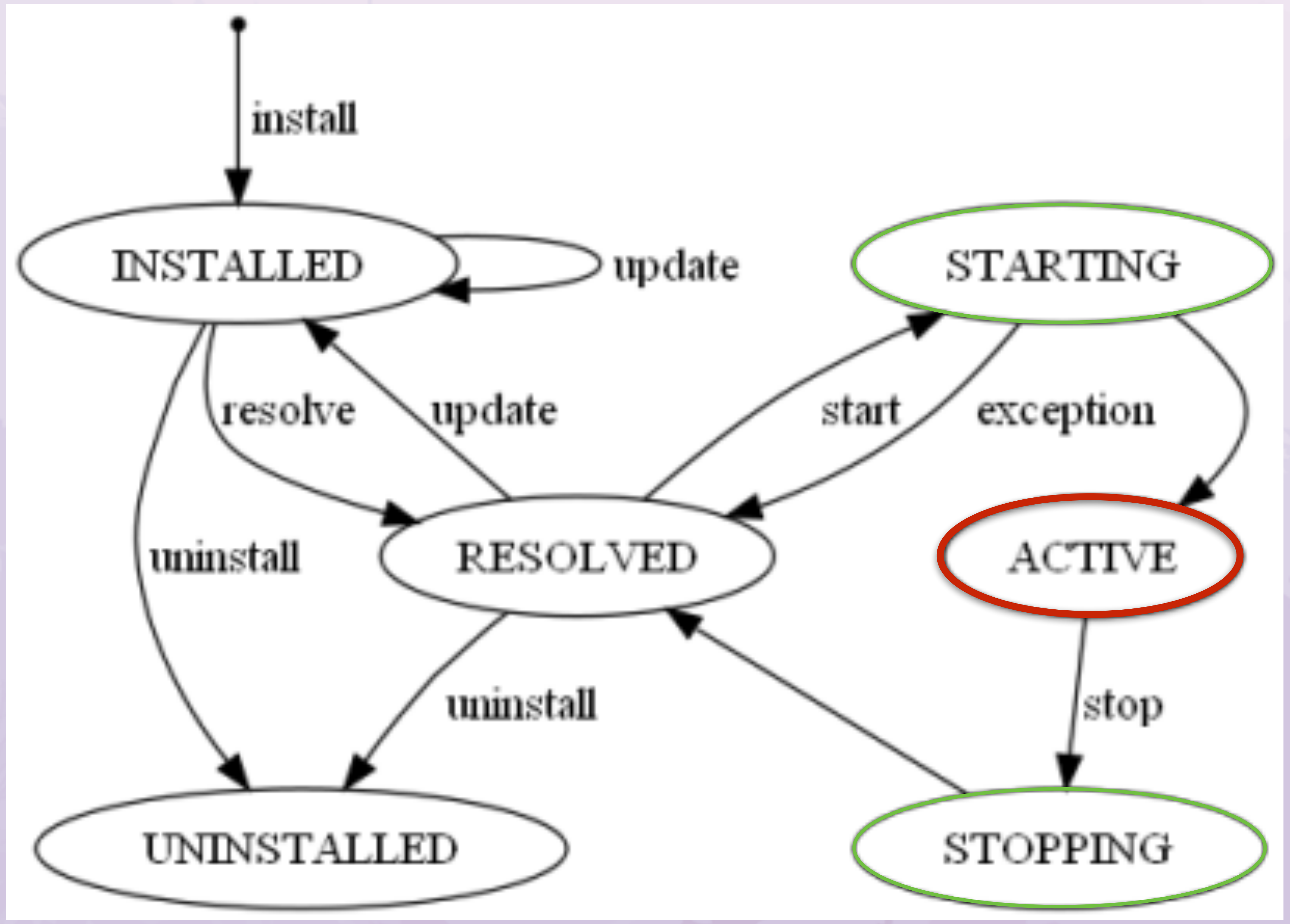
- OSGi is shown as worthless technology forcing you to have extra entries in manifest
- OSGi brings more problems than solutions
- Many libraries deny to support it because small user base

OSGi popularity by Maven Central

- Maven central is 426k JARs, assigned to 46k projects
- Just 10% of projects provides valid OSGi bundles
- OSGi core jar is at 36th position in download statistics
- 24k projects has transient OSGi dependencies

Source: http://blog.osgi.org/ analysis done by Peter Kriens

OSGi lifecycle



Bundle Activator call

Source: http://docs.spring.io/

OSGi services
are main source
of dynamism in system

OSGi related design patterns

APACHECON DENVER
WESTIN DENVER DOWNTOWN
APRIL 7-9, 2014

- Whiteboard
- Extender

Modularity vs Extensibility

Modular software

- Provides clear API and SPI contract
- Small set of dependencies
- Fine grained dependencies
- Lack of implementation specific depdenencies

Extensible software

- Provides plug in capabilities
- Does not lock extension to have its own extensions
- Extension registration should be as less verbose as possible

You can't build extensible solutions without making them modular first

Your solution might be modular but not extensible at all

Your software can be modular and extensible and still not run under OSGi

Karaf based solution

Building solution on top of Karaf

- Make it OSGi friendly
- Make use of services
- Decide to use helpers like blueprint/spring/declarative services
- Extend Karaf to support your domain
- Brand it, assemble as new product
- Ship it to customer

Making things OSGi friendly

- For maven users - use maven-bundle-plugin
- Take care about OSGification of your dependencies
  - Push changes back to origin project if you can
  - If fail - then wrap it
  - If you do - then share results with community
- Keep naming convention strict & clear

Maven users

```
<build>
  <plugins>
    <plugin>
      <groupId>org.apache.felix</groupId>
      <artifactId>maven-bundle-plugin</artifactId>
      <extensions>true</extensions>
    </plugin>
  </plugins>
</build>
```

```
<parent>
    <groupId>org.example</groupId>
    <artifactId>parent</artifactId>
    <version>2.0.0-SNAPSHOT</version>
</parent>
<artifactId>api</artifactId>
<packaging>bundle</packaging>
```

Dependency OSGification

- Quite simple if dependency uses Maven
- Best thing you can do is giving OSGi support back to project
- Exporting wrapped packages is bad practice
- If you wrap - then make separate bundle
- Submit patch for ServiceMix bundles gain karma

Extending Karaf

- Adopt default configuration to use own authentication source (or write new)
- Group your bundles into Karaf features
- Provide new set of commands supporting your product

Authentication sources

- Following authentication mechanisms are supported (JAAS login modules):
  - JDBC
  - LDAP
  - OSGi configuration admin
  - Property files
  - Public key

Karaf features

- Simple descriptor with information which bundles from where should be installed
- Supports multiple protocols
  - File
  - HTTP / FTP etc
  - mvn (pax-url-mvn)
- Supports transient dependencies, version ranges and custom resolvers

Karaf features cont.

APACHECON DENVER
WESTIN DENVER DOWNTOWN
APRIL 7-9, 2014

```
<features name="repository-name">
 <repository>mvn:groupId/artifactId/version/xml/features</repository>

 <feature name="my-solution" version="2.0">
    <feature version="(1.1,1.1]">dependency-feature-name</feature>
    <bundle>mvn:groupId/artifactId/2.0</bundle>
    <bundle>mvn:groupId/another-artifactId/2.0</bundle>
  </feature>
</features>
```

Karaf features cont.

- Install it:
  - features:addurl mvn:groupId/artifactId/2.0/xml/features
  - features:install my-solution
- Add to default configuration
  - etc/org.apache.karaf.features.cfg
    - featuresRepositories += url
    - featuresBoot += my-solution

Implementing commands

```
package org.apachecon.na;

import org.apache.felix.gogo.commands.Command;
import org.apache.karaf.shell.console.OsgiCommandSupport;

@Command(scope="example", name="do")
public class SimpleCommand extends OsgiCommandSupport {

    @Override
    protected Object doExecute() throws Exception {
        System.out.println("Something ...");
        return null;
    }
}
```

Implementing commands

- Don't forget about few additional things:
  - blueprint descriptor
  - transient dependencies
    - org.apache.felix.service.command
    - org.apache.felix.gogo.commands
    - org.apache.karaf.shell.console

Brand it

- create file branding.properties in org.apache.karaf.branding package
- Package as OSGi bundle…
- … and copy to lib/ directory

.. and assemble

- Use features-maven-plugin
  - add-features-to-repo goal
- Most important configuration
  - descriptors
  - features
  - repository => output directory

.. and assemble cont.

- At the end use assembly maven plugin to create ZIP/TAR archive

- karaf-maven-plugin for Karaf 3.0 will create assembly with your features without any additional steps

What we have learned*

* These things might be obvious ..

Getting runtime metrics from couple Apache projects is hard

Common problems

- Developers not familiar with OSGi..
  - .NET developers using Java in first project..
- Dependency upgrades
- Feature version ranges are hard to use
- By default shutdown timeout for Java Service Wrapper is just 30 seconds
- VM queues are danger when processing is very slow (OOME)

Common problems cont.

- Pre-defined assemlies are big
- Integration testing is slow
- Visual tools produces nice but useless artifacts
  - Generated code
  - Problematic testing
- Handling of KAR files
- Karaf feature URLs management

Standard OSGi problems

- Usage of legacy libraries not actively maintained
- Blueprint service caching in threads - they are blind for bundle/service upgrades
  - Very hard to write perfect update scenario
  - Impossible to determine change impact

Good things goes last

- Camel scales up very well
- ActiveMQ is good for having load balancing
- VM endpoints are damn fast
- Cassandra is damn fast too!
- Increase load by 300% just by enabling aggregation & batch processing

Few patterns which works pretty well

- ActiveMQ queue as central entry point
  - Content based routing to glue components
  - Very easy to track incoming load and spikes
  - Messages are published via service call
- Separate destinations for OSGi services
  - Uses Service Tracker to dynamically create camel routes
  - VM endpoints everywhere, JMS where bottlenecks can occur

APACHECON DENVER
WESTIN DENVER DOWNTOWN
APRIL 7-9, 2014

New and noteworthly in Karaf 3.0

Q&A

APACHECON DENVER
WESTIN DENVER DOWNTOWN
APRIL 7-9, 2014

Stay in touch

Twitter & skype: ldywicki

luke@code-house.org

https://github.com/splatch/apachecon