Apache Karaf Cellar and Fuse Fabric

Ioannis Canellos

Camel One – May 2012

FuseSource
integration everywhere

Your Speaker

- Ioannis Canellos
  - iocanel@fusesource.com
  - Blog: http://iocanel.blogspot.com
- Software Architect @ FuseSource
- Open Source Contributor
  - Apache Karaf
  - Apache ServiceMix
  - Apache Camel
  - Apache Whirr
  - Jclouds
  - Founder of Apache Karaf Cellar

Agenda

- Managing distributed OSGi Runtimes
- Apache Karaf Cellar
- Fuse Fabric
- Questions & Answers

Agenda

- **Managing distributed OSGi Runtimes**
- Apache Karaf Cellar
- Fuse Fabric
- Questions & Answers

Agenda

- **Managing distributed OSGi Runtimes**
  - What is OSGi?
  - What makes OSGi really cool?
  - A new “challenge”
  - When OSGi cross the boundaries of a single runtime
- Apache Karaf Cellar
- Fuse Fabric
- Questions & Answers

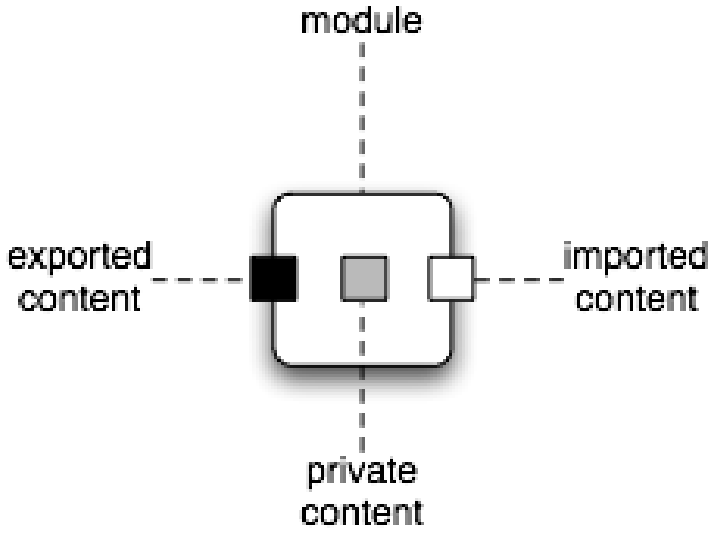
Managing distributed OSGi Runtimes:
What is OSGi?

- **A set of standards**
  - The missing modularity layer for the Java virtual machine
  - Additional layers for dynamic applications
    - Lifecycle layer
    - Service layer

- **Some core concepts**
  - Bundle
    - A jar with well defined capabilities, requirements & content visibility
    - A jar with lifecycle
  - Service
    - An object usually implementing an interface + Properties
    - Registered, Looked Up & “Listened”

Managing distributed OSGi Runtimes:
What is OSGi? (Bundles)

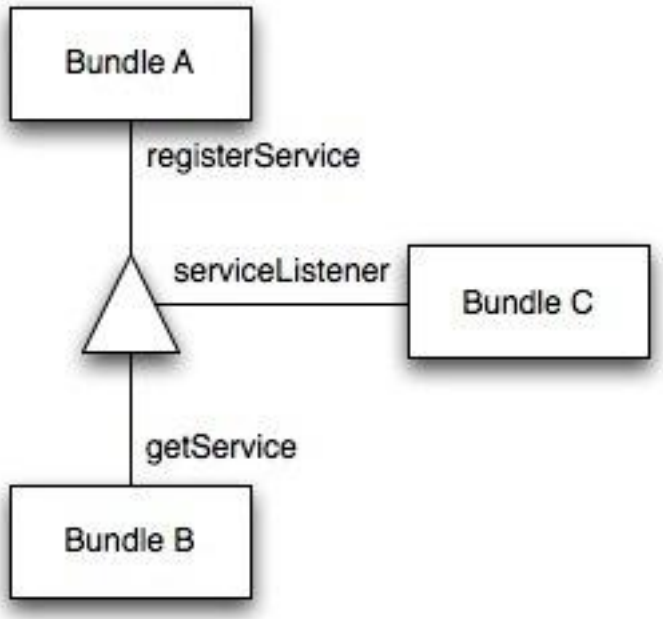
- **Bundle**
  - Group of classes & resources
  - Content visibility metadata
- **Versioning of Classes**
  - Multiple versions of Class
  - Importing with version ranges
- **Bundle Lifecycle**
  - Installed
  - Resolved
  - Started
  - Stopping
  - Uninstalled



FuseSource
integration everywhere

Managing distributed OSGi Runtimes:
What is OSGi? (Services)

- **What is a service?**
  - An Object
  - A set of properties
- **The Service Registry**
  - A global registry for all bundles
  - Registering a service
  - Getting a service
  - Listening for services



FuseSource
integration everywhere

Managing distributed OSGi Runtimes:
What makes OSGi really cool?

- **A slide that could be a presentation on its own**
  - Modularity
  - Sensible programming model
- **Let's focus**
  - Dynamic nature
    - Updatable bundles
    - Installing / Uninstalling bundles at runtime *(no restarts needed)*
    - Dynamically adding functionality via OSGi services
- **Why is that so important?**
  - Things are bound to change

Managing distributed OSGi Runtimes:
What is Karaf's added coolness?

- **A set of standard services**
  - Logging Service
  - Configuration Admin
- **Features Concept**
  - Easy to use provisioning mechanism
    - Grouping of bundles & configuration into features
    - Composing an application from multiple features
    - *Interaction with the OBR*
- **Deployers**
  - Bundle
  - Features
  - War
  - Spring / Blueprint

Managing distributed OSGi Runtimes:
A new “challenge”

- **Embracing dynamism is great**
  - Being able to update parts of the application
  - Being able to modify the runtime behavior of an existing application
  - Being able to dynamically reconfigure the application
- **What happens in distributed environments?**
  - How do I reconfigure multiple runtimes?
  - How do I install a new bundle in multiple runtimes?
  - How do I install a feature in multiple runtimes?
  - Can one runtime consume an OSGI service provided by an other?
  - How can I discover which logical services provided by which runtime?

Managing distributed OSGi Runtimes:
A new “challenge”

- Projects that were designed to help you meet this challenge
  - **Apache Karaf Cellar**
  - **Fuse Fabric**

Agenda

- *Managing distributed OSGi Runtimes*
- **Apache Karaf Cellar**
- Fuse Fabric
- Questions & Answers

Agenda

- *Managing distributed OSGi Runtimes*
- **Apache Karaf Cellar**
  - Architecture
  - Groups
  - Configuration admin integration
  - Bundle replication
  - Features integration
  - Distributed services
- Fuse Fabric
- Questions & Answers

Apache Karaf Cellar
Overview

- **Basic principals**
  - Keep it as simple as possible *(K.I.S.S. principal)*
  - Replicate changes changes across multiple runtimes
  - Mimic the steps that would otherwise be manual
- **Core features**
  - Uses and manages Hazelcast
  - Pluggable discovery *(multicast, unicast, cloud)*
  - Groups of runtimes
  - Keeping containers in sync
    - Configuration replication
    - Bundle / Feature replication
  - Distributed OSGi services

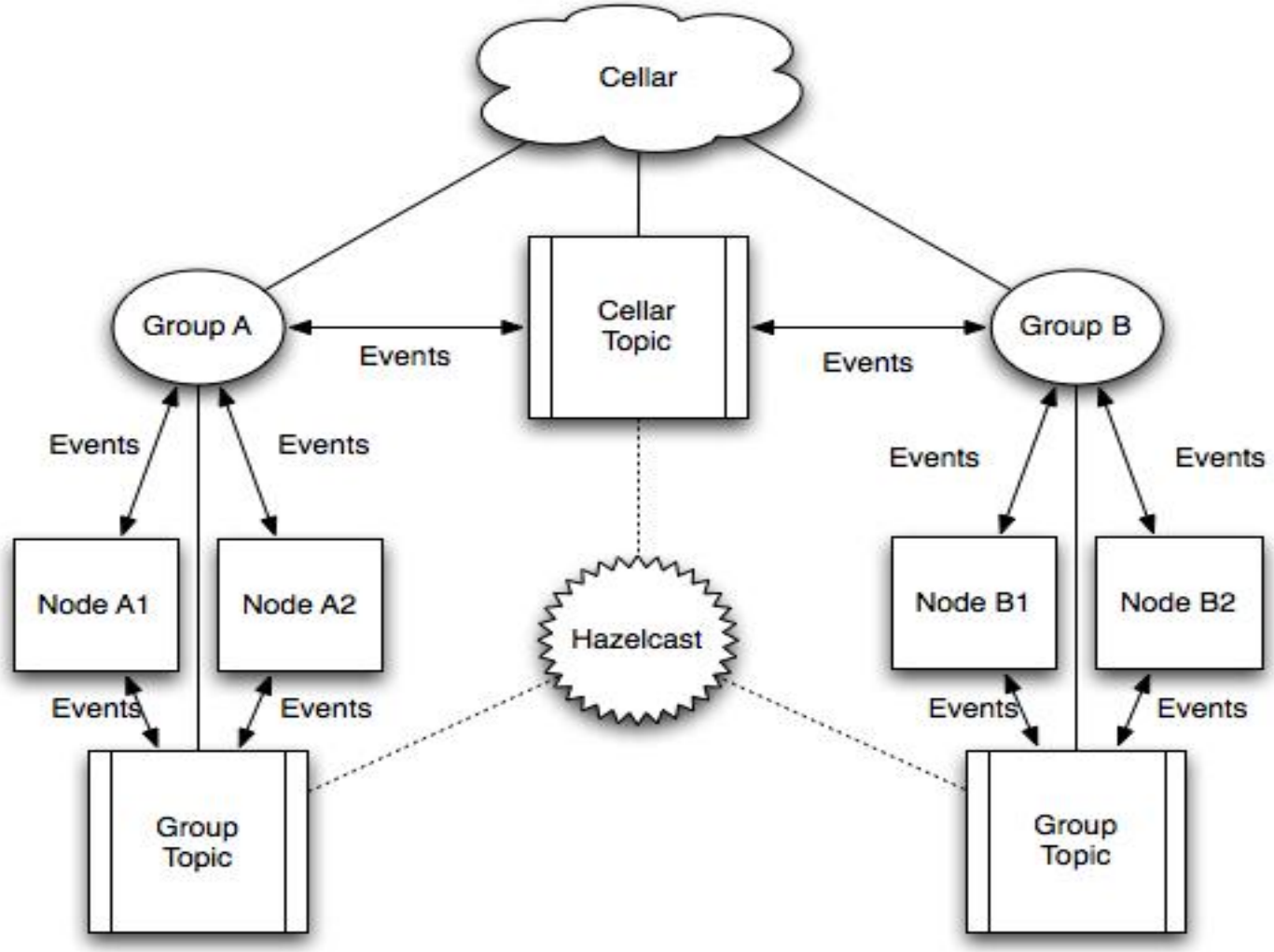
Apache Karaf Cellar
Overview: “Installing Cellar”

```
  Fuse ESB (7.0.0.fuse-060)
  http://fusesource.com/products/fuse-esb-enterprise/

Hit '<tab>' for a list of available commands
and '[cmd] --help' for help on a specific command.
Hit '<ctrl-d>' or 'osgi:shutdown' to shutdown Fuse ESB.

FuseESB:karaf@root>
```

Apache Karaf Cellar
Architecture



FuseSource
integration everywhere

Apache Karaf Cellar
Groups

- Why do I need groups?
  - You are not always running a single application.
  - Not all containers need to host exactly the same layers
    - Fronted, Backend, integration etc.

- How does cellar treat groups?
  - Nodes can be grouped together
  - Each group has a dedicated communication transport
  - Nodes can “sync” state with group buddies

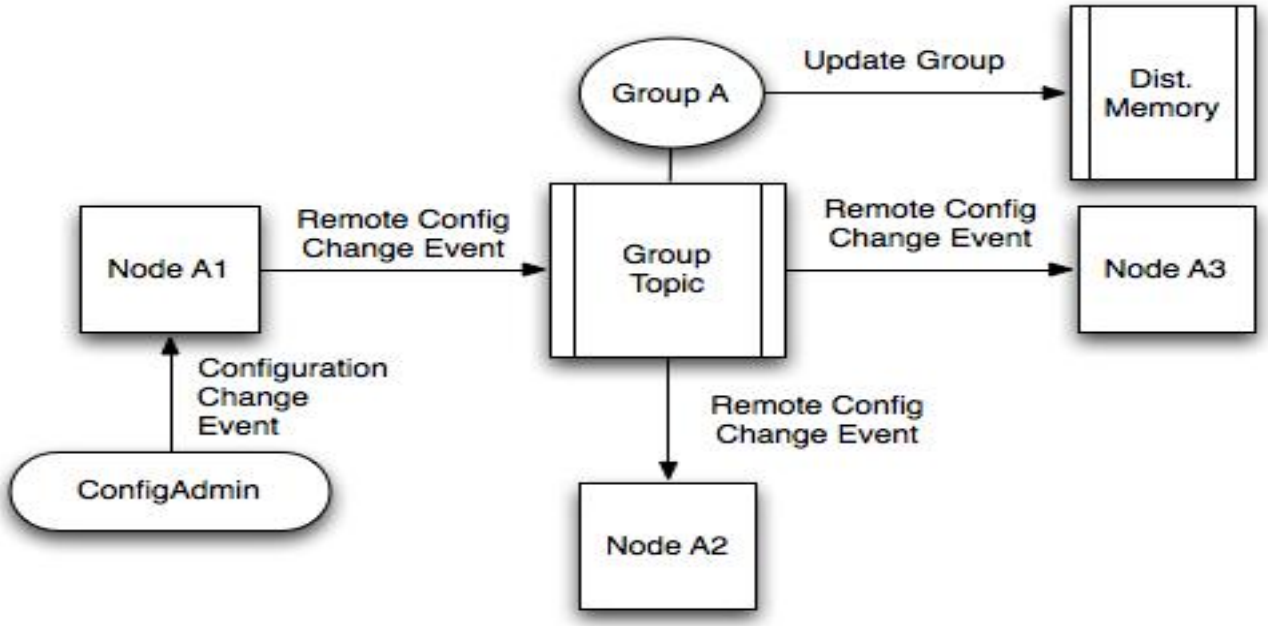
Apache Karaf Cellar
Groups

```
FuseESB:karaf@root>
```


FuseSource
integration everywhere

Apache Karaf Cellar
Configuration admin integration

- Listens for configuration change events
- Broadcasts events to nodes of the same group *(optional)*
- Supports event blacklist / white list
- Supports group pre configuration *(optional)*

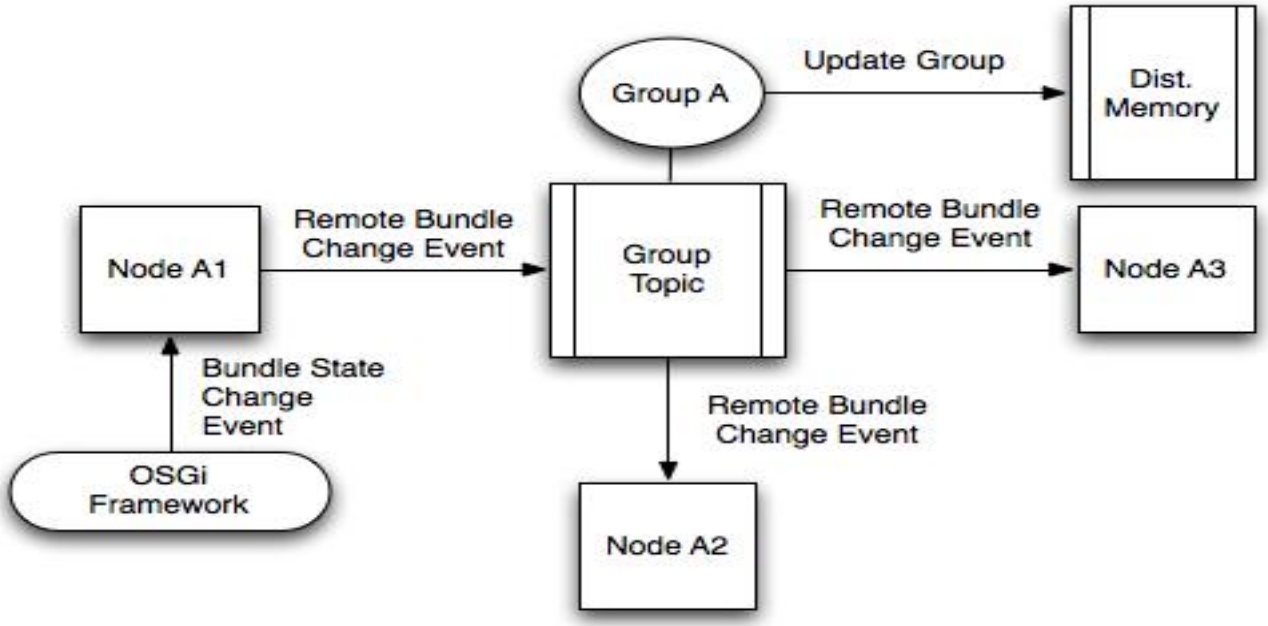


Apache Karaf Cellar
Syncing configuration between members

```
FuseESB:karaf@root>
```

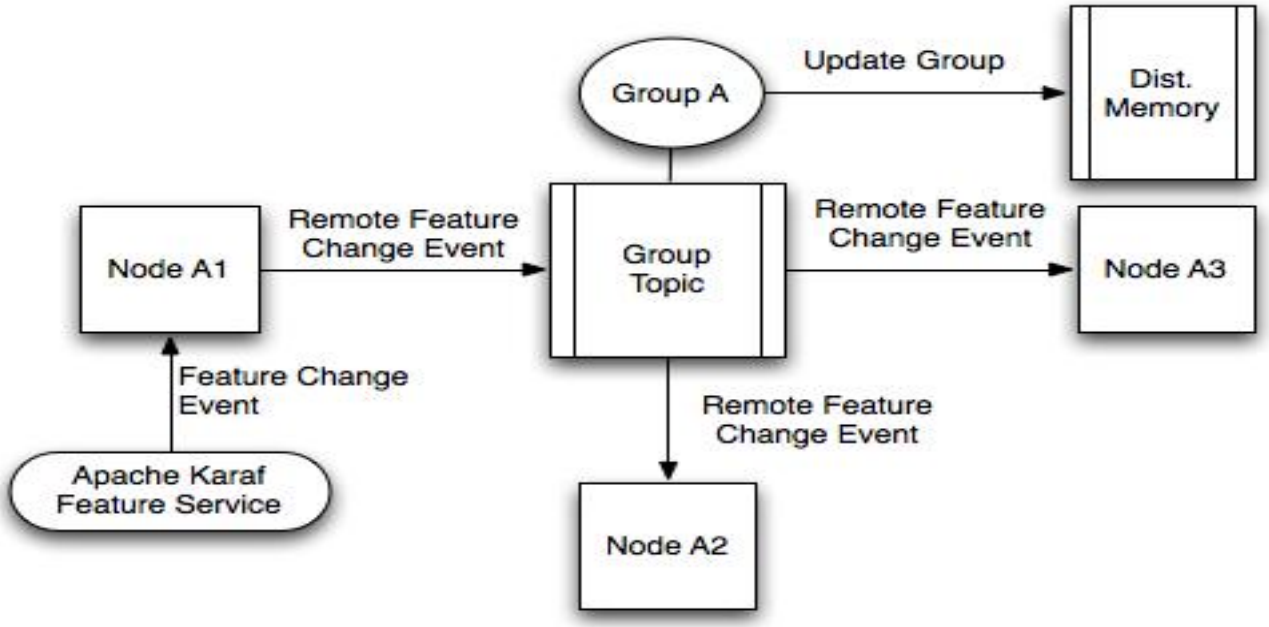
Apache Karaf Cellar
Bundle state replication

- Listens for bundle events
- Broadcasts events to nodes of the same group *(optional)*
- Supports event blacklist / white list
- Supports group pre configuration *(optional)*



Apache Karaf Cellar
Feature Service Integration

- Listens for bundle events
- Broadcasts events to nodes of the same group *(optional)*
- Supports event blacklist / white list
- Supports group pre configuration *(optional)*



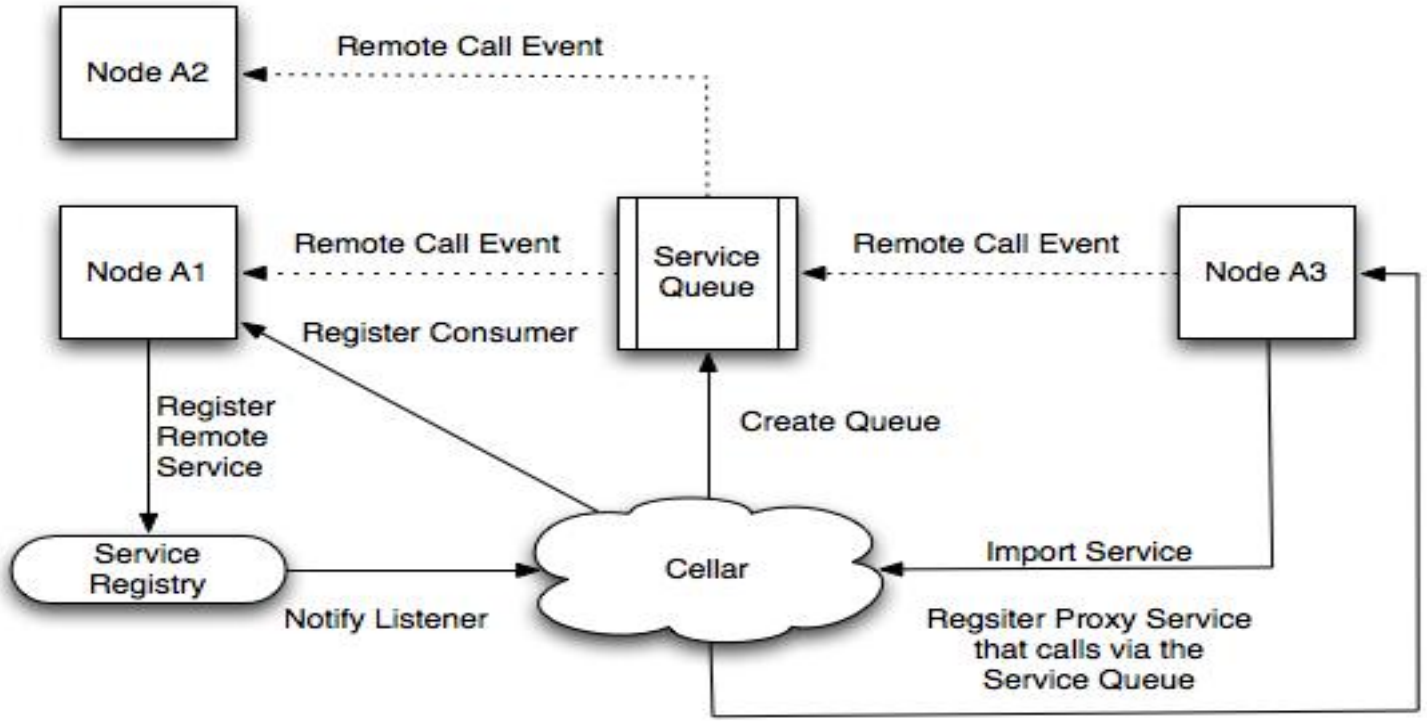
FuseSource
integration everywhere

Apache Karaf Cellar
Syncing / assigning features to groups

```
FuseESB:karaf@root>
```

Apache Karaf Cellar
Distributed Service Execution

- Implementation of OSGi remote service spec. *(partial)*
- Evenly distributed load across nodes
- Dynamically scales



FuseSource
integration everywhere

Apache Karaf Cellar
Summarizing

- **A really simple solution**
  - Configure one container per group and sync the rest
  - Pluggable discovery mechanism
  - Helps you scale up

Agenda

- *Managing distributed OSGi Runtimes*
- *Apache Karaf Cellar*
- **Fuse Fabric**
- Questions & Answers

Agenda

- *Managing distributed OSGi Runtimes*
- *Apache Karaf Cellar*
- **Fuse Fabric**
  - Overview
  - Architecture
  - Registry
  - Fabric Agent
  - Profiles
  - Creating remote containers
  - Middleware integration
  - Distributed OSGi
- Questions & Answers

Fuse Fabric:
Overview

- **Open Source System for:**
  - Distributed configuration
  - Distributed provisioning
  - Distributed management
- **Supports Karaf based containers:**
  - Fuse ESB
  - Fuse MQ
  - Karaf
  - Service Mix
- **Well integrated with:**
  - Camel
  - ActiveMQ
  - CXF

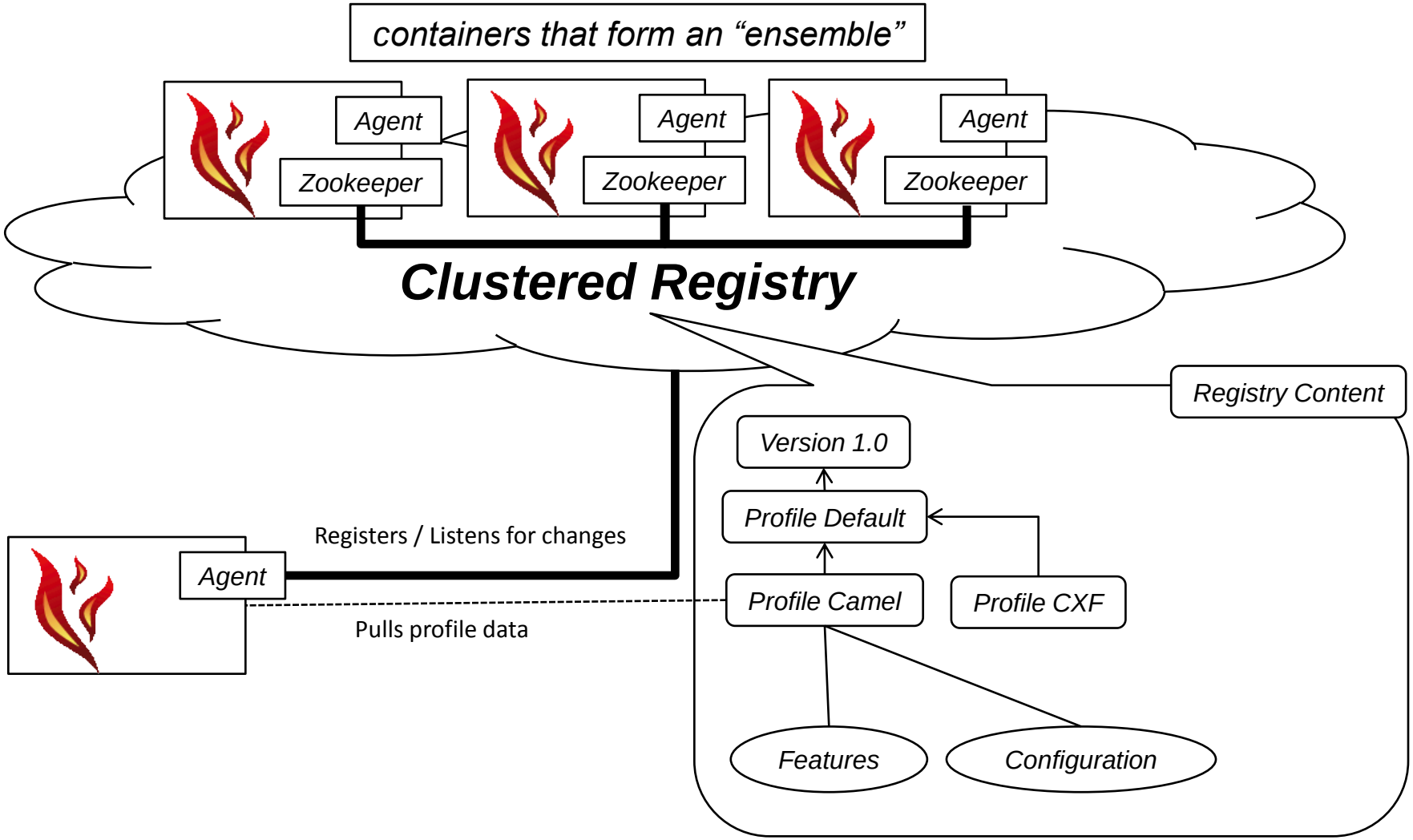
Fuse Fabric:
Overview *(cont.)*

- **Core concepts**
  - Fabric registry
    - Formed by an even number of containers
    - Holds all configuration data
    - Registry for distributed services
  - Profile
    - Describes the container setup
      - Features, Bundles, FABs, Configuration PIDs etc
    - Hierarchical structure
    - Versioning
      - Easy means to upgrade / rollback containers
  - Fabric Agent
    - Runs on each container
    - Makes sure that the container “provisions” its assigned profiles.

Fuse Fabric:
Overview *(cont.)*

- **Core features**
  - Provisioning
    - Deploy apps to containers using profiles
    - Incremental upgrades / rollbacks for containers
      - From configuration to the OSGi framework itself
  - Discovery
    - ActiveMQ brokers
    - Camel endpoints
    - CXF endpoints
  - Installation of remote containers
    - Install the runtime itself on remote hosts.
    - Creation of cloud instances & installation of runtime.
  - Management
    - Fuse Management Console *(aka FMC)*
    - *Fuse IDE*

Fuse Fabric:
Architecture



FuseSource
integration everywhere

Fuse Fabric:
Registry

- **Based on Apache Zookeeper**
  - A highly available service that provides
    - Configuration information services
    - Distrusted synchronization etc
- **Registry Model**
    - Hierarchy of “znode” similar to a file system
    - Each “znode” can hold data and/or have children.
- **Setup Options**
    - Create a zookeeper ensemble from fabric (fabric managed)
      - *Create, add or remove containers from the ensemble*
    - Use an existing zookeeper ensemble to host the registry
- **Management**
    - Shell commands to interact with the registry at zookeeper level
    - *Tools to import & export registry content to files*

Fuse Fabric:
Registry: “Create a new registry”

```
  Fuse ESB (7.0.0.fuse-060)
  http://fusesource.com/products/fuse-esb-enterprise/

Hit '<tab>' for a list of available commands
and '[cmd] --help' for help on a specific command.
Hit '<ctrl-d>' or 'osgi:shutdown' to shutdown Fuse ESB.

FuseESB:karaf@root>
```


FuseSource
integration everywhere

Fuse Fabric:
Registry: “Join an existing registry”

```
  Fuse ESB (7.0.0.fuse-060)
  http://fusesource.com/products/fuse-esb-enterprise/

Hit '<tab>' for a list of available commands
and '[cmd] --help' for help on a specific command.
Hit '<ctrl-d>' or 'osgi:shutdown' to shutdown Fuse ESB.

FuseESB:karaf@root>
```


FuseSource
integration everywhere

Fuse Fabric:
Registry: “Managing the ensemble”

```
[id]                            [version] [alive] [profiles]                      [provision status]
root*                           1.0       true    fabric, fabric-ensemble-0000-1
ssh1                            1.0       true    default                         success
ssh2                            1.0       true    default                         success
```

Fuse Fabric:
Profiles

- **A description of how a container should be provisioned**
  - Framework
  - System properties
  - OSGi configuration
  - Features, bundles, FABs
- **Structure & usage**
  - Hierarchical *(supports multiple parents)*
  - One or more profiles can be assigned to a container
  - Can be used to define logical groups of containers
- **Defaults Profiles**
  - default, karaf
  - camel, cxf, mq, esb

Fuse Fabric:
Profiles

```
FuseESB:karaf@root>
```


FuseSource
integration everywhere

Fuse Fabric:
Agent

- **Runs on each container that is part of Fabric**
  - Connects to the fabric registry
  - Reads profiles assigned to the container
  - Applies configuration, installs bundles, features etc
  - Listens for changes
- **Upgrades and rollbacks**
  - Can incrementally update / rollback containers in the cluster
  - Can update even itself and go as low as the OSGi framework
- **Where does it get the "artifacts" from?**
  - From a predefined set of public maven repositories
  - Fabric containers can as maven proxies themselves
    - Support uploading artifacts in an mvn:deploy manner
    - Support downloading artifacts

Fuse Fabric:
Agent: “Changing the profile”

```
[id]                            [version] [alive] [profiles]                    [provision status]
root*                           1.0       true    fabric, fabric-ensemble-0000-1
  child1                        1.0       true    default                       success
```


FuseSource
integration everywhere

Fuse Fabric:
Middleware integration

- **MQ integration**
  - Containers using the “mq” profiles will automatically start a broker
  - The broker will register itself in fabric
  - Fabric containers can discover brokers via fabric
  - Master / slave support
- **Camel integration**
  - Any provider endpoint can be registered in fabric
    - Example: **from(“fabric:myendpoint:http://0.0.0.0:8383”)**
  - Discovery & load balancing
    - Example: **from(“direct:start”).to(“fabric:myendpoint”)**
- **CXF integration**
  - Similar to camel

Fuse Fabric:
Distributed OSGi services

- **Distributed OSGi services implementation**
  - As simple as adding a property to a service
    - service.exported.interfaces
  - Uses insanely fast hawtdispatch
  - Can be used consumed from non-OSGi clients

  - Examples:
    - Fabric ships a dosgi profile you can use “out of the box”
    - Fabric examples contain a camel & dosgi example

Fuse Fabric:
Creating remote containers

"Fabric can weave itself"

- **Can create "new' containers with any profile from scratch**
  - Locally in a separate jvm
  - On remote ssh enabled hosts
  - In the cloud
    - Public Cloud *(EC2, Rackspace etc)*
    - Private Cloud
    - Hybrid Cloud

Fuse Fabric:
Creating remote containers: “In the local network”

- Can make use of your existing servers
- Installs fabric via ssh
  - Support public key authentication
  - Supports passphrase on key
- Starts the runtime
- Automatically joins the cluster
- Can be assigned any profile

- Example use cases & benefits
  - Add a new message broker in your network in no time
  - Scale your application by dynamically adding runtimes
  - Reduce the maintenance overhead

Fuse Fabric:
Creating remote containers: “In the cloud”

- Works with most public cloud providers
- Supports private clouds

- How fabric makes cloud easy for you
  - Creates cloud instances
  - Performs the minimal required firewall management *(for hybrids)*
  - Installs all required software *(java, curl etc)*
  - Install & starts fabric
  - Automatically joins the cluster

Fuse Fabric:
Creating remote containers: In the cloud

```
FuseESB:karaf@root> fabric:create
FuseESB:karaf@root> features:install fabric-jclouds jclouds-aws-ec2
FuseESB:karaf@root> fabric:cloud-provider-add aws-ec2 AKIAJVJF2ZXAOQYGULDQ XJWKmeEpBShuuPZSghjTpshCJSJ2UYsKqNWPXA8L

Waiting for aws-ec2 service to initialize.
FuseESB:karaf@root>
```

Fuse Fabric:
Summarizing

- **Central management of how the cluster should be provisioned**
  - Define your profiles & let provisioning to fabric
  - Not just configuration and deployment
    - Broader scope that go as low as the runtime itself
  - Clean way to manage your upgrades & rollbacks
- **Scaling**
  - Distributed OSGi services
  - Native support for your “favorite” middleware
- **Cloud support**
  - Can make use of your “own nodes”
  - Makes installing container & app to cloud “a piece of cake”
    - Public clouds
    - Private clouds
    - Hybrid couds

Agenda

- *Managing distributed OSGi Runtimes*
- *Apache Karaf Cellar*
- *Fuse Fabric*
- **Questions & Answers**

Thank You 

FuseSource
integration everywhere