Deploying Modular Applications with Apache ACE

Marcel Offermans

Marcel Offermans

Fellow at Luminis Technologies
Member at the Apache Software Foundation

@m4rr5

- Top Level Project since december 2011
- Software distribution framework based on OSGi
- active community, 11 committers
- http://ace.apache.org/

Agenda

- The case for modularity
- Taming deployment complexity
- Targeting different containers
- Into the cloud
- Future directions

The case for modularity

Maintainability

Adaptability

Quest for Reuse

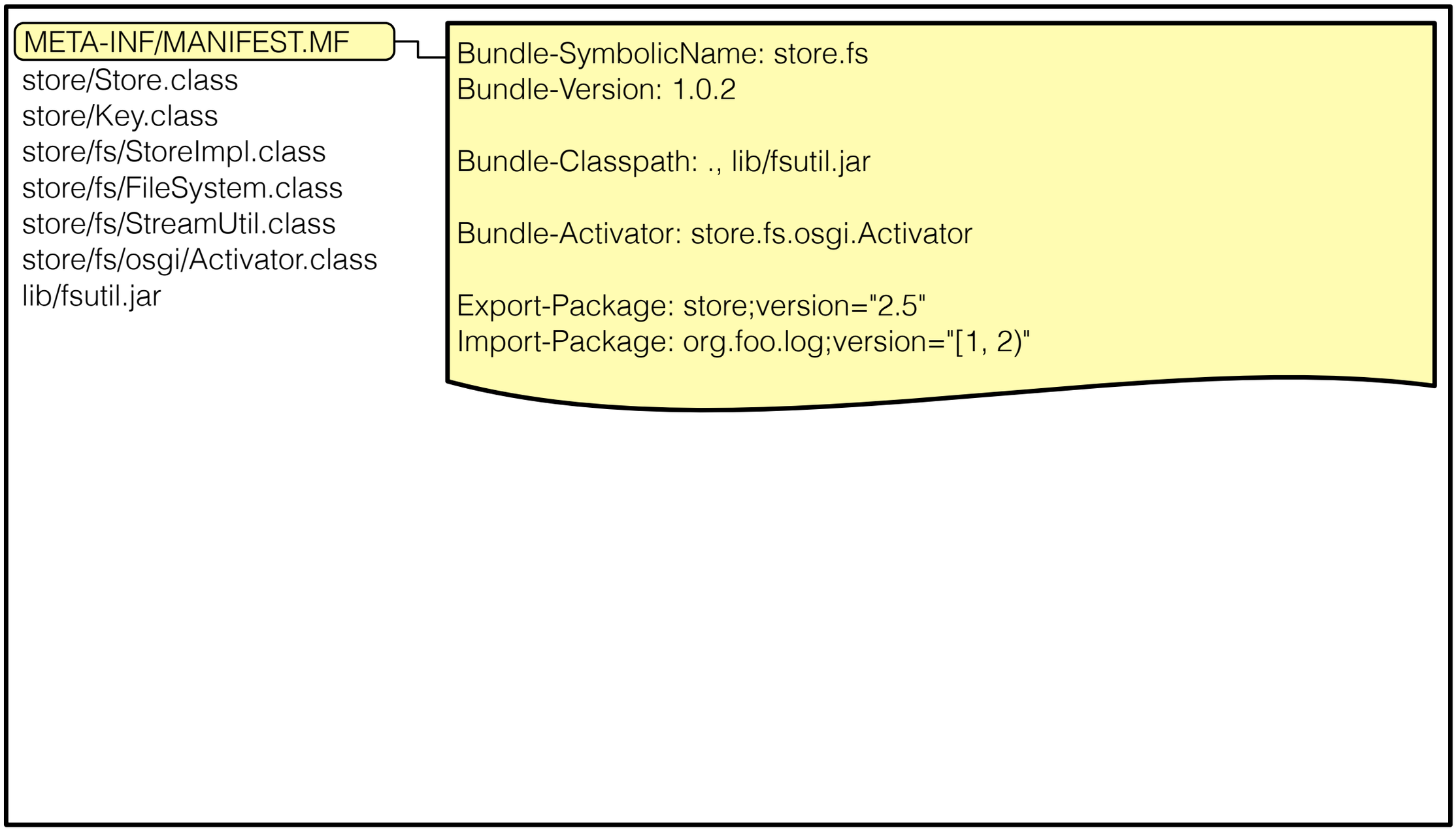
Copy / Paste

Quest for Reuse Object Oriented

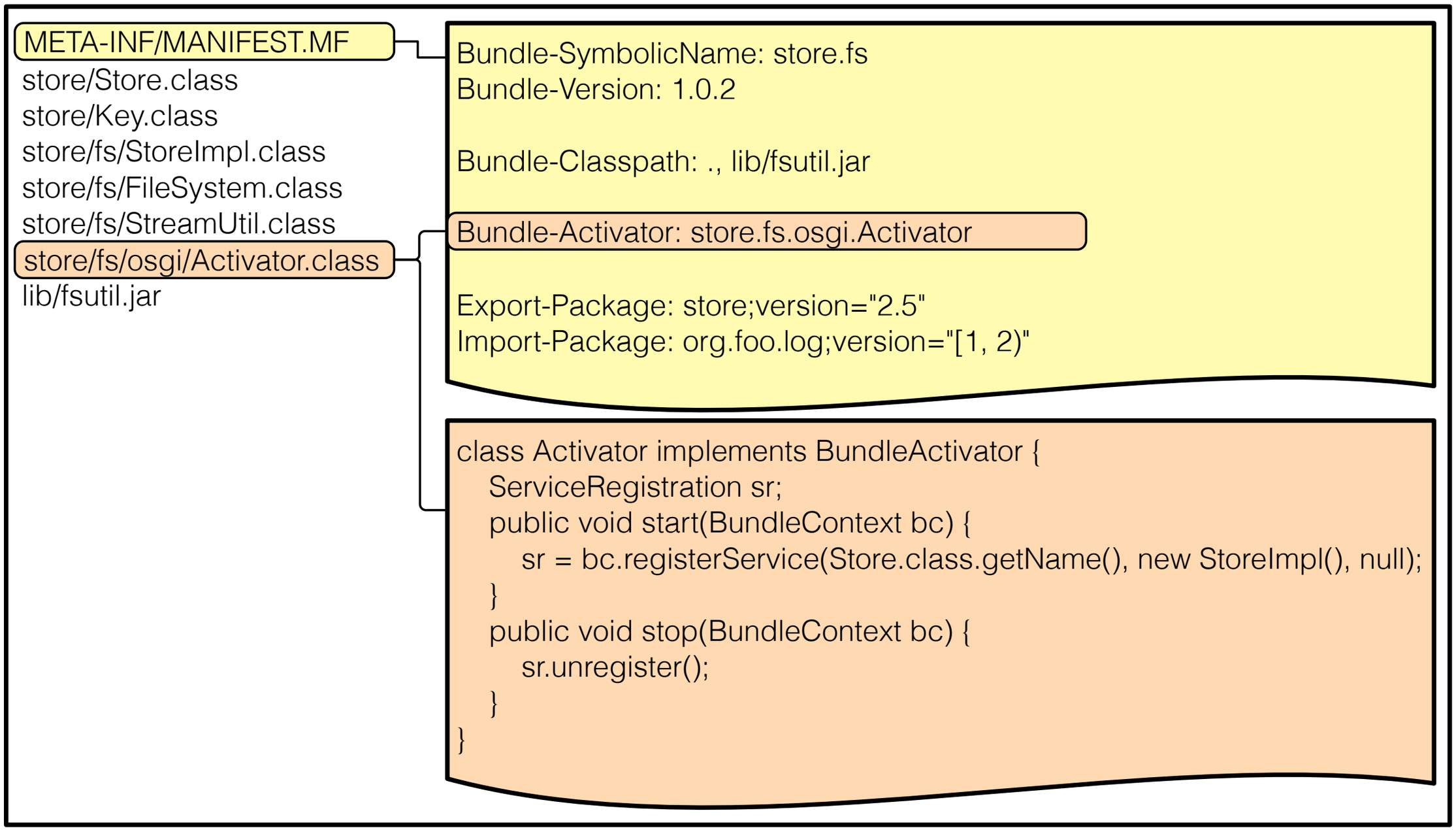
Quest for Reuse

Component Based

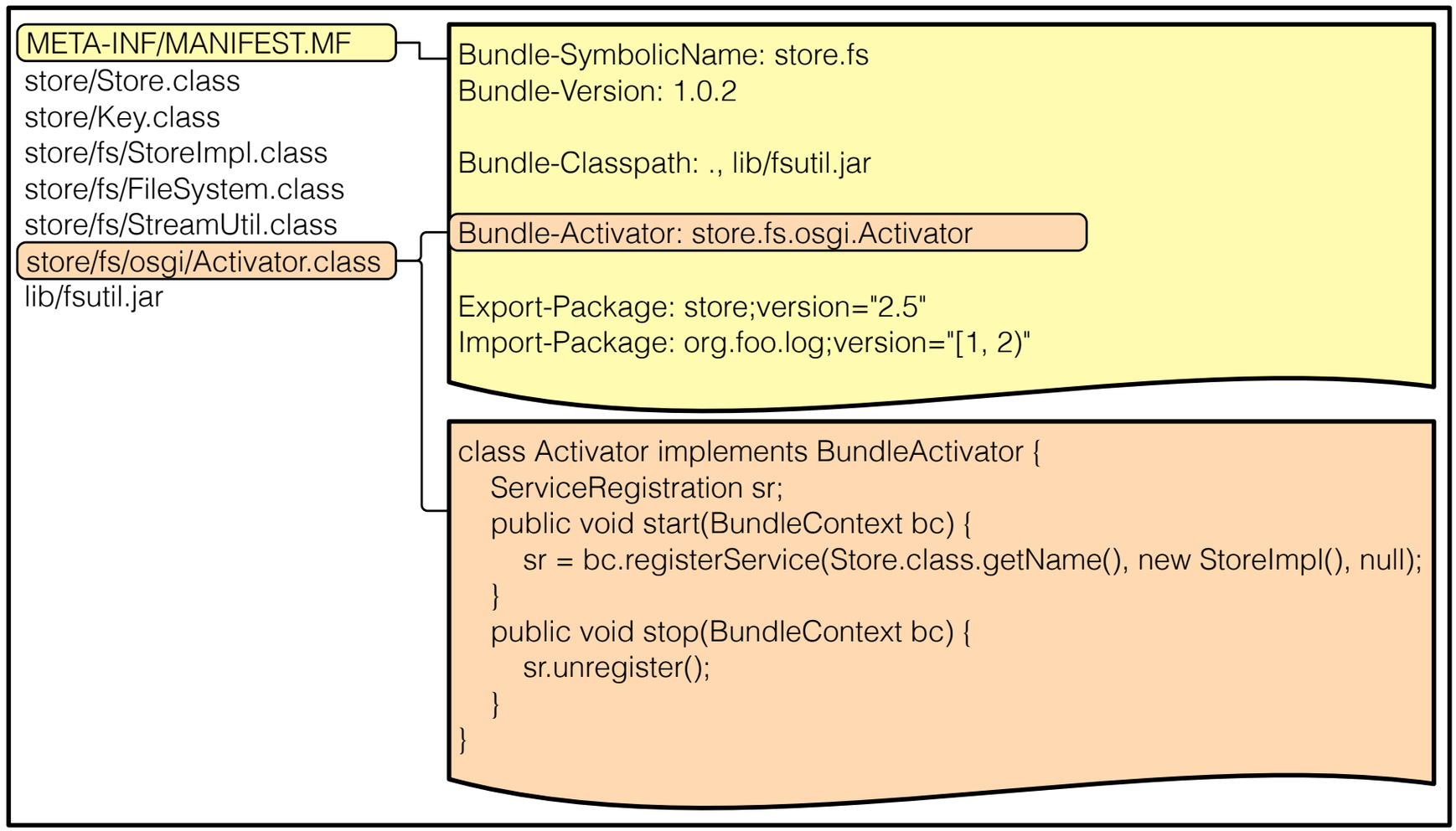
OSGi


META-INF/MANIFEST.MF
store/Store.class
store/Key.class
store/fs/StoreImpl.class
store/fs/FileSystem.class
store/fs/StreamUtil.class
store/fs/osgi/Activator.class
lib/fsutil.jar
Bundle-SymbolicName: store.fs
Bundle-Version: 1.0.2
Bundle-Classpath: ., lib/fsutil.jar
Bundle-Activator: store.fs.osgi.Activator
Export-Package: store;version="2.5"
Import-Package: org.foo.log;version="[1, 2)"

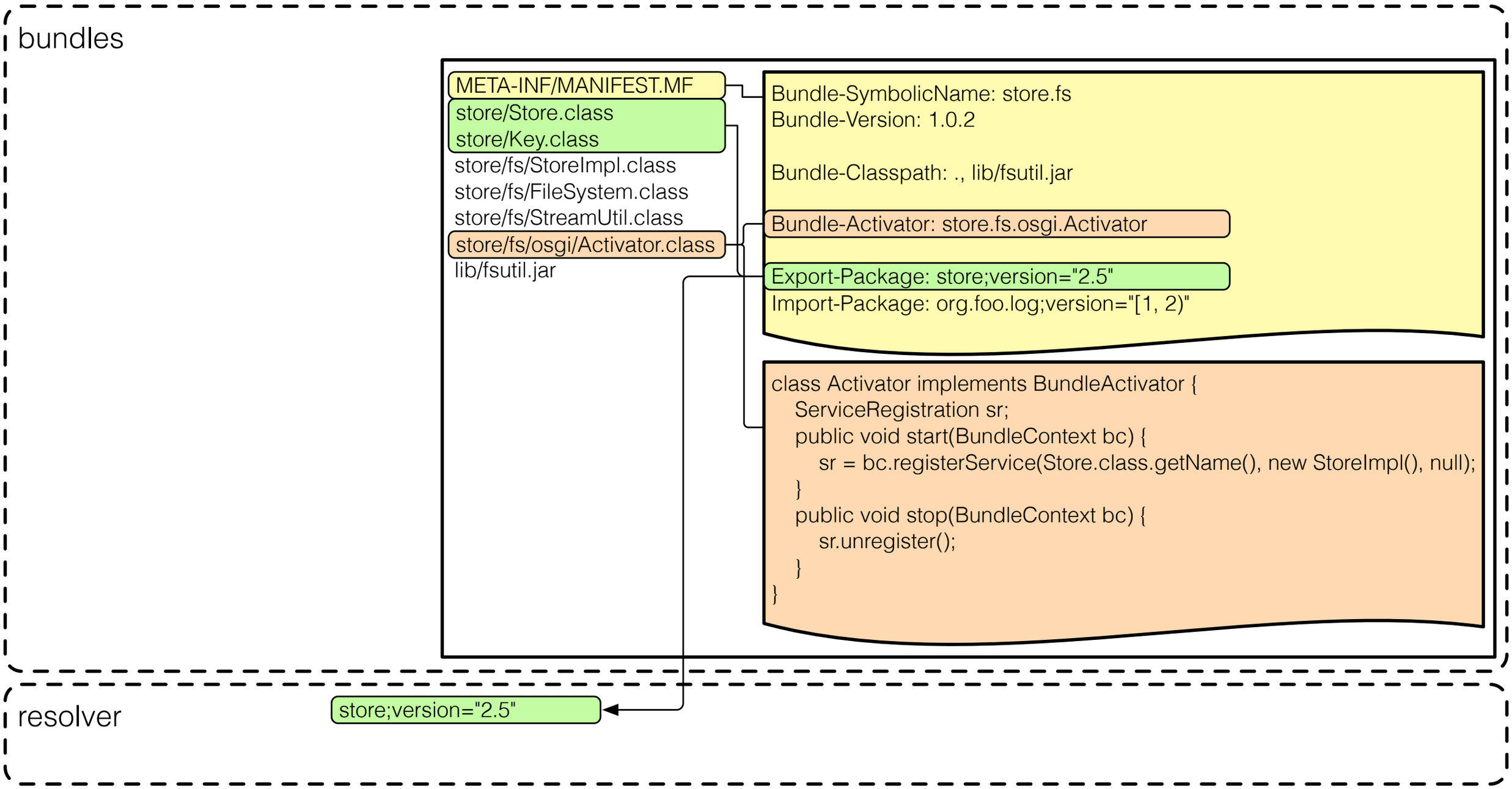
OSGi


META-INF/MANIFEST.MF
store/Store.class
store/Key.class
store/fs/StoreImpl.class
store/fs/FileSystem.class
store/fs/StreamUtil.class
store/fs/osgi/Activator.class
lib/fsutil.jar
Bundle-SymbolicName: store.fs
Bundle-Version: 1.0.2
Bundle-Classpath: ., lib/fsutil.jar
Bundle-Activator: store.fs.osgi.Activator
Export-Package: store;version="2.5"
Import-Package: org.foo.log;version="[1, 2)"
class Activator implements BundleActivator {
ServiceRegistration sr;
public void start(BundleContext bc) {
sr = bc.registerService(Store.class.getName(), new StoreImpl(), null);
}
public void stop(BundleContext bc) {
sr.unregister();
}
}

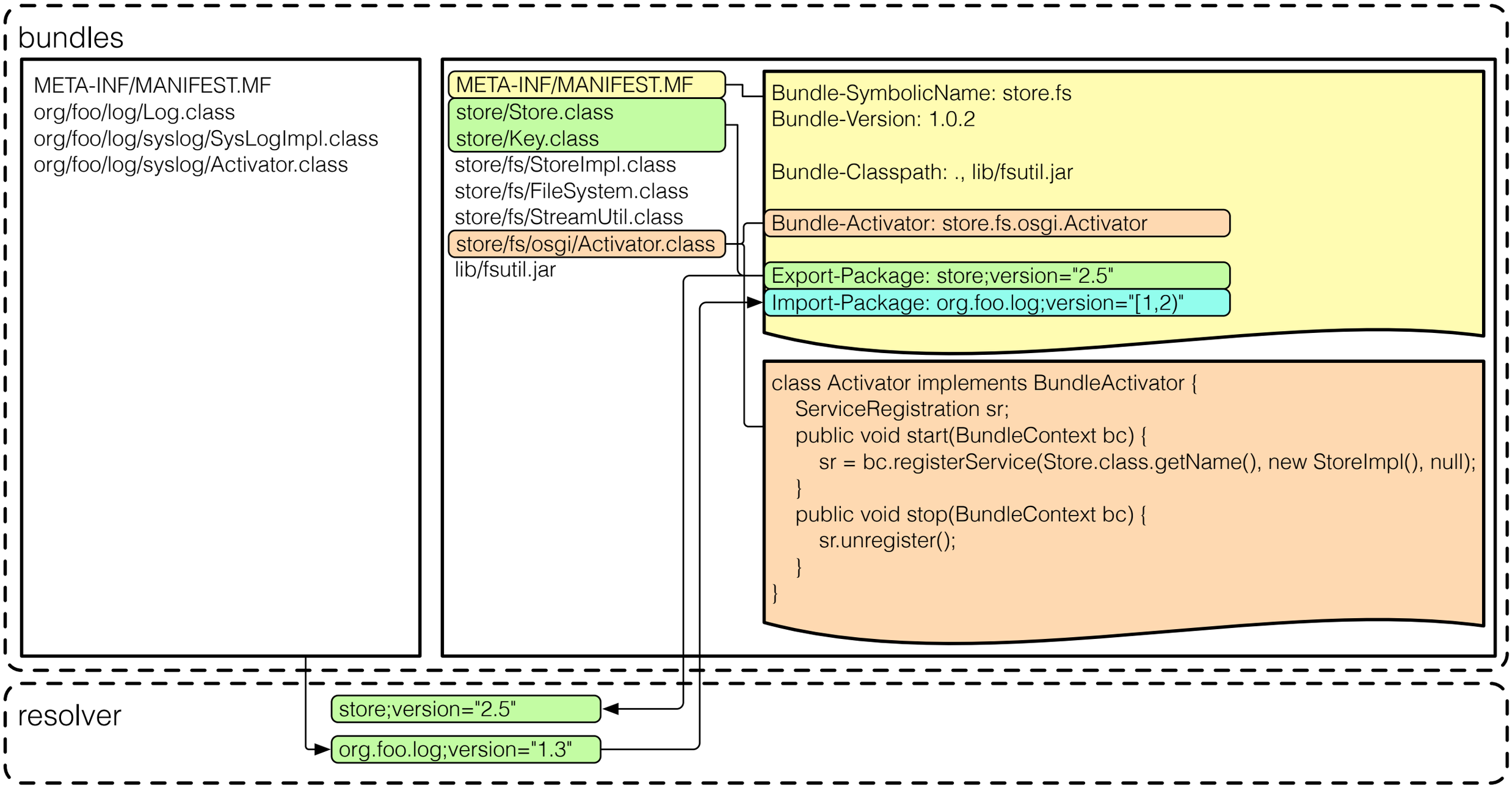
OSGi


META-INF/MANIFEST.MF
store/Store.class
store/Key.class
store/fs/StoreImpl.class
store/fs/FileSystem.class
store/fs/StreamUtil.class
store/fs/osgi/Activator.class
lib/fsutil.jar
Bundle-SymbolicName: store.fs
Bundle-Version: 1.0.2
Bundle-Classpath: ., lib/fsutil.jar
Bundle-Activator: store.fs.osgi.Activator
Export-Package: store;version="2.5"
Import-Package: org.foo.log;version="[1, 2)"
class Activator implements BundleActivator {
ServiceRegistration sr;
public void start(BundleContext bc) {
sr = bc.registerService(Store.class.getName(), new StoreImpl(), null);
}
public void stop(BundleContext bc) {
sr.unregister();
}
}

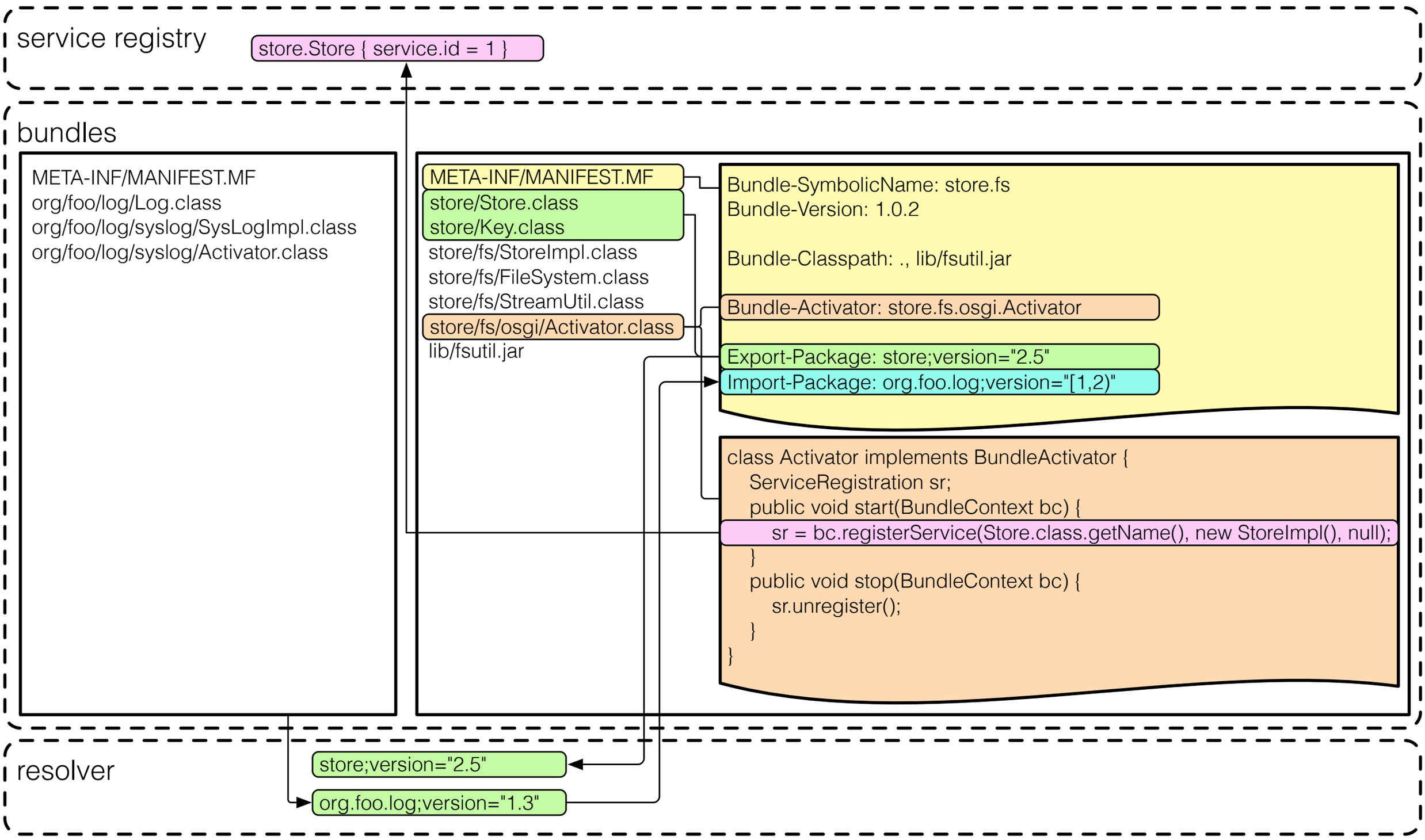
OSGi


bundles
META-INF/MANIFEST.MF
store/Store.class
store/Key.class
store/fs/StoreImpl.class
store/fs/FileSystem.class
store/fs/StreamUtil.class
store/fs/osgi/Activator.class
lib/fsutil.jar
Bundle-SymbolicName: store.fs
Bundle-Version: 1.0.2
Bundle-Classpath: ., lib/fsutil.jar
Bundle-Activator: store.fs.osgi.Activator
Export-Package: store;version="2.5"
Import-Package: org.foo.log;version="[1, 2)"
class Activator implements BundleActivator {
ServiceRegistration sr;
public void start(BundleContext bc) {
sr = bc.registerService(Store.class.getName(), new StoreImpl(), null);
}
public void stop(BundleContext bc) {
sr.unregister();
}
}
resolver
store;version="2.5"

OSGi

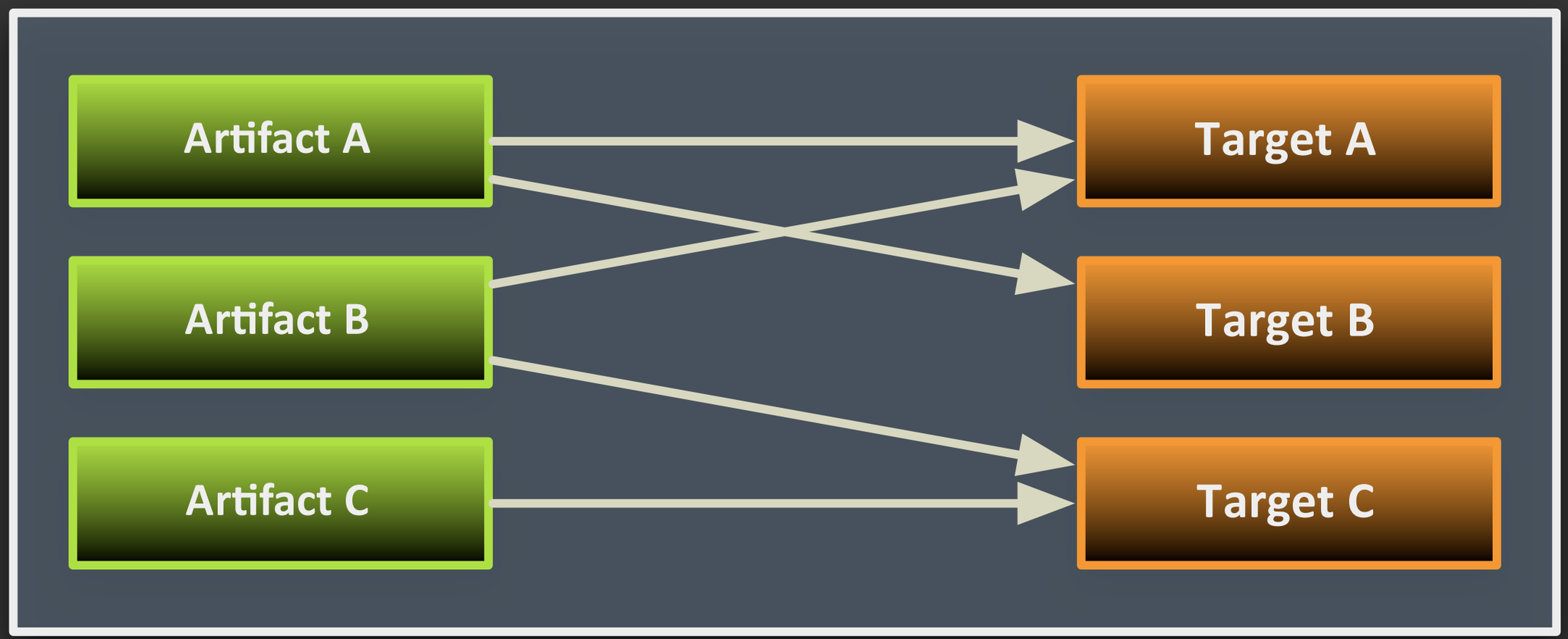

bundles
META-INF/MANIFEST.MF
org/foo/log/Log.class
org/foo/log/syslog/SysLogImpl.class
org/foo/log/syslog/Activator.class
META-INF/MANIFEST.MF
store/Store.class
store/Key.class
store/fs/StoreImpl.class
store/fs/FileSystem.class
store/fs/StreamUtil.class
store/fs/osgi/Activator.class
lib/fsutil.jar
Bundle-SymbolicName: store.fs
Bundle-Version: 1.0.2
Bundle-Classpath: ., lib/fsutil.jar
Bundle-Activator: store.fs.osgi.Activator
Export-Package: store;version="2.5"
Import-Package: org.foo.log;version="[1,2)"
class Activator implements BundleActivator {
ServiceRegistration sr;
public void start(BundleContext bc) {
sr = bc.registerService(Store.class.getName(), new StoreImpl(), null);
}
public void stop(BundleContext bc) {
sr.unregister();
}
}
resolver
store;version="2.5"
org.foo.log;version="1.3"

OSGi

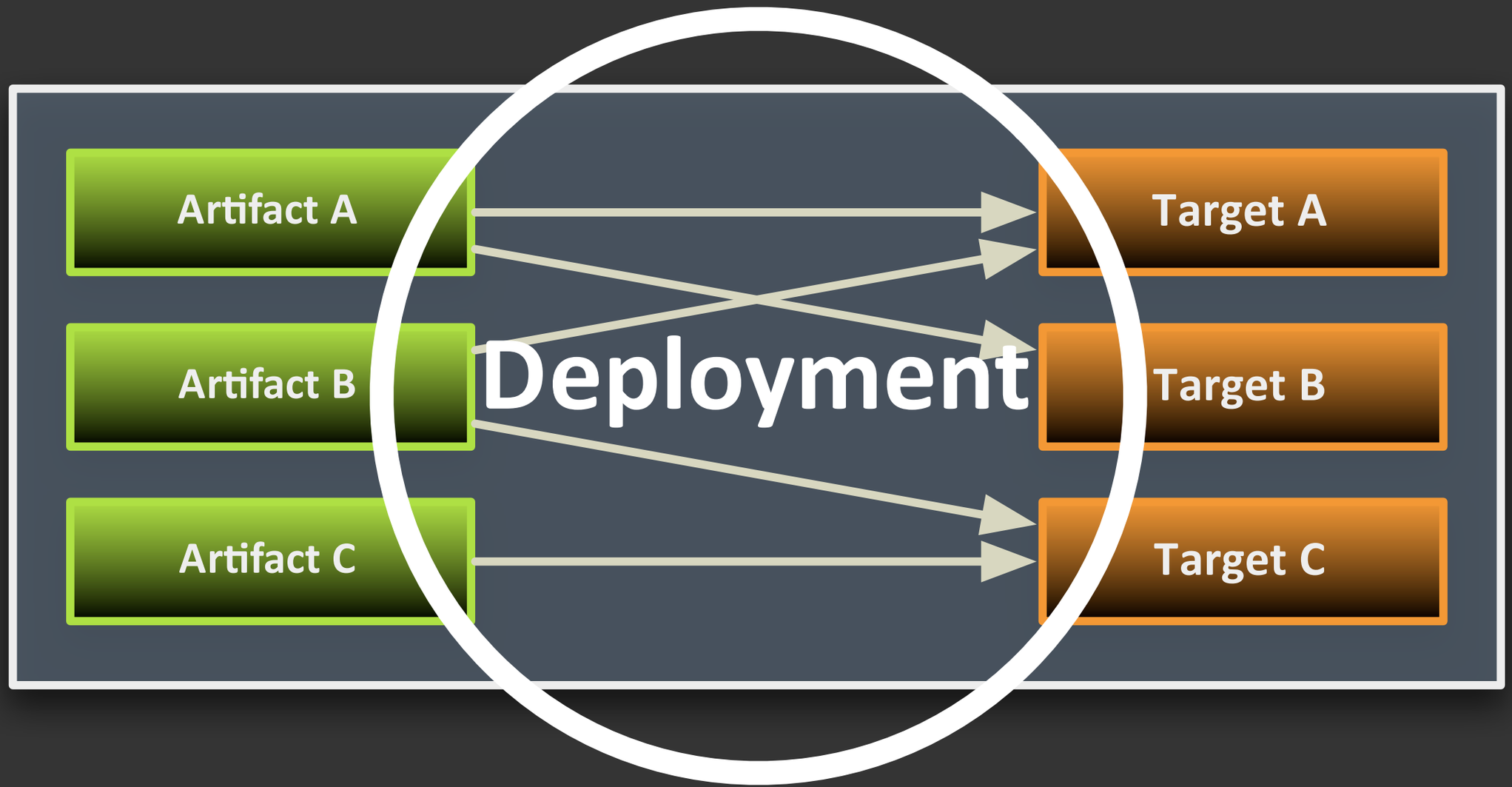

service registry
store.Store { service.id = 1 }
bundles
META-INF/MANIFEST.MF
org/foo/log/Log.class
org/foo/log/syslog/SysLogImpl.class
org/foo/log/syslog/Activator.class
META-INF/MANIFEST.MF
store/Store.class
store/Key.class
store/fs/StoreImpl.class
store/fs/FileSystem.class
store/fs/StreamUtil.class
store/fs/osgi/Activator.class
lib/fsutil.jar
Bundle-SymbolicName: store.fs
Bundle-Version: 1.0.2
Bundle-Classpath: ., lib/fsutil.jar
Bundle-Activator: store.fs.osgi.Activator
Export-Package: store;version="2.5"
Import-Package: org.foo.log;version="[1,2)"
class Activator implements BundleActivator {
ServiceRegistration sr;
public void start(BundleContext bc) {
sr = bc.registerService(Store.class.getName(), new StoreImpl(), null);
}
public void stop(BundleContext bc) {
sr.unregister();
}
}
resolver
store;version="2.5"
org.foo.log;version="1.3"


Taming deployment
complexity

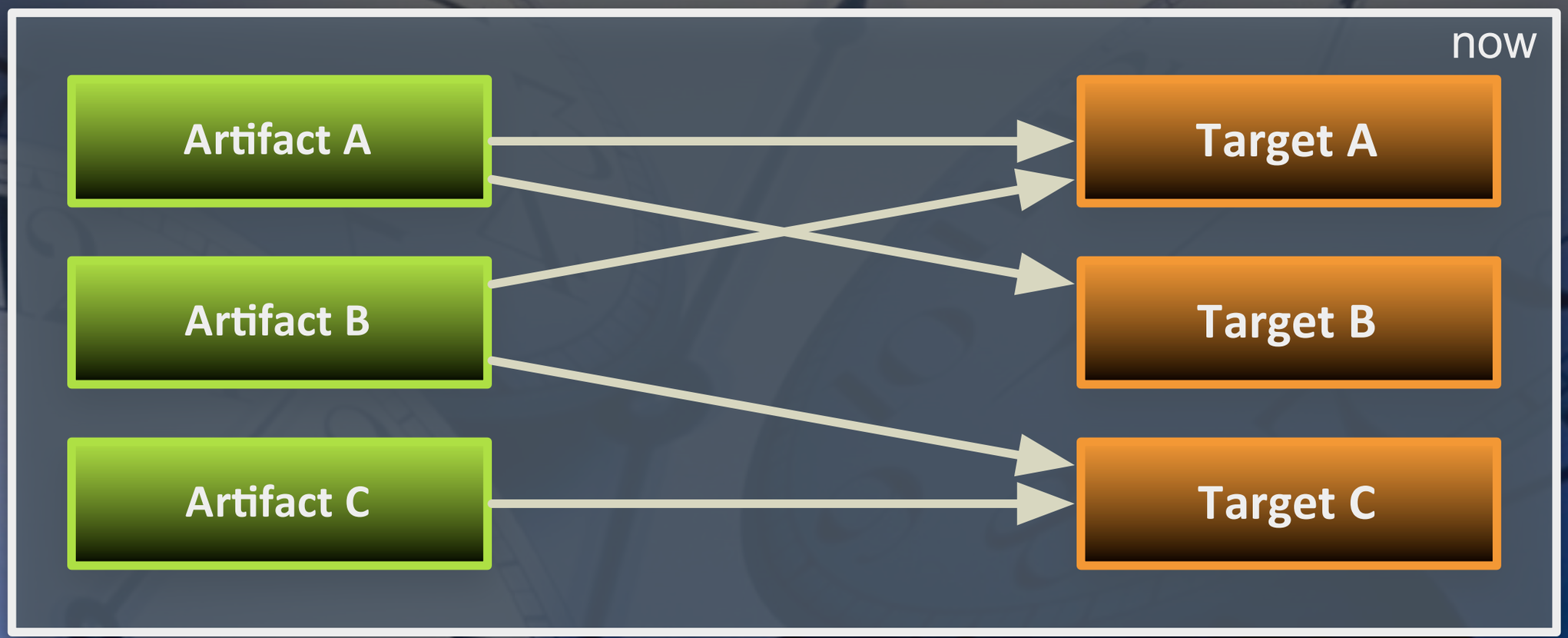
Deployment



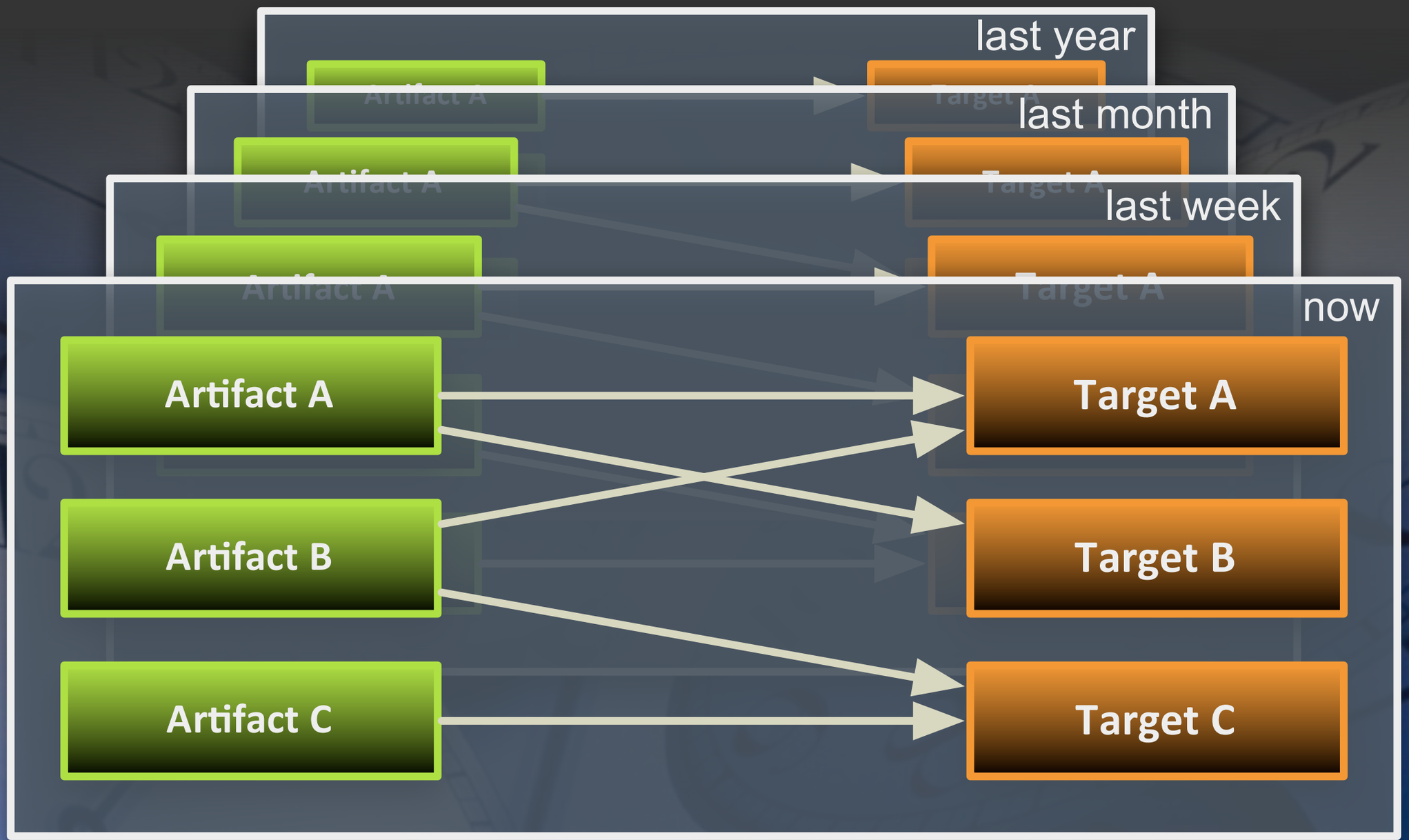
Deployment


Artifact A
Artifact B
Artifact C
Deployment
Target A
Target B
Target C

Keeping the history



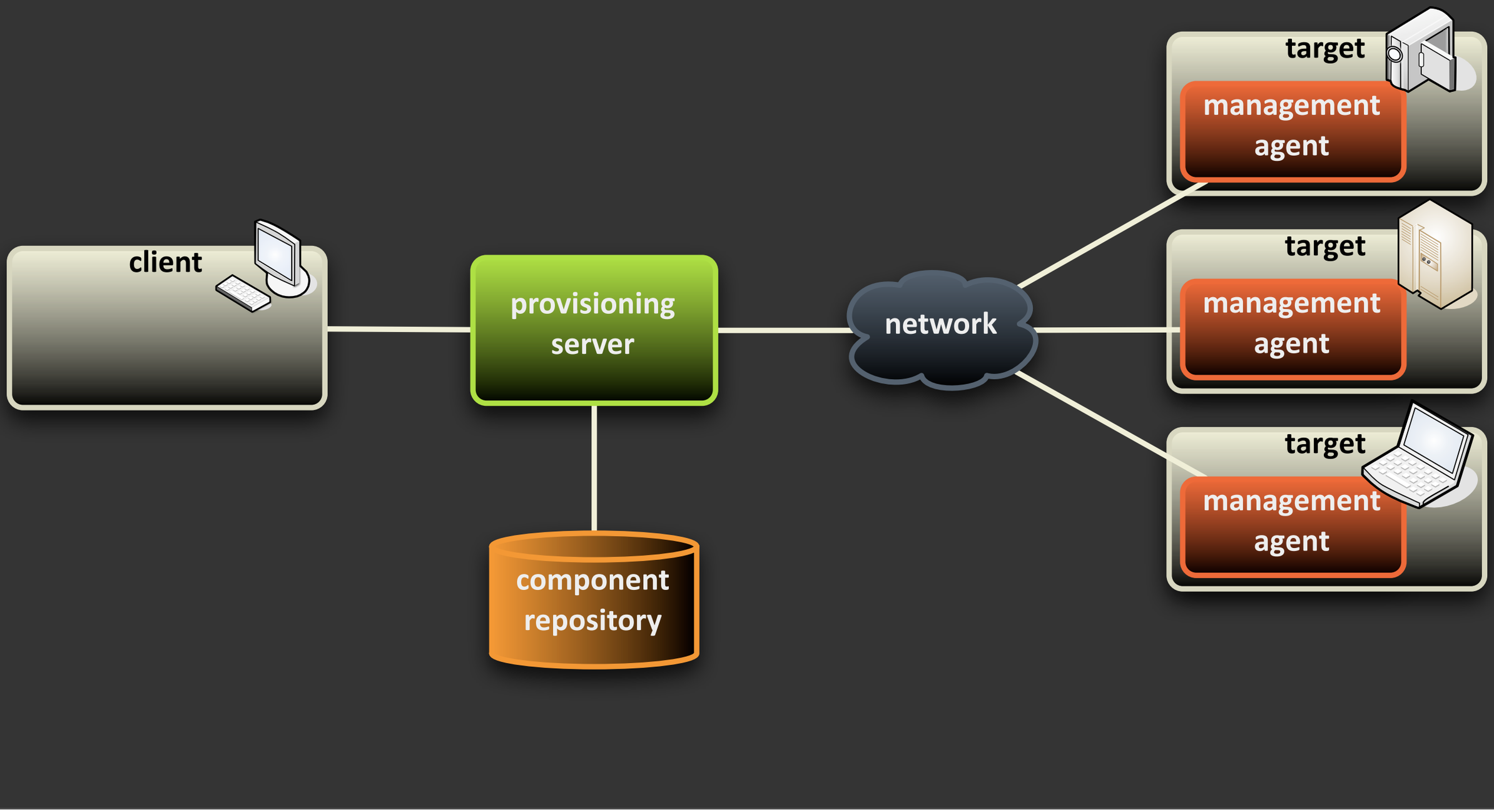
Keeping the history



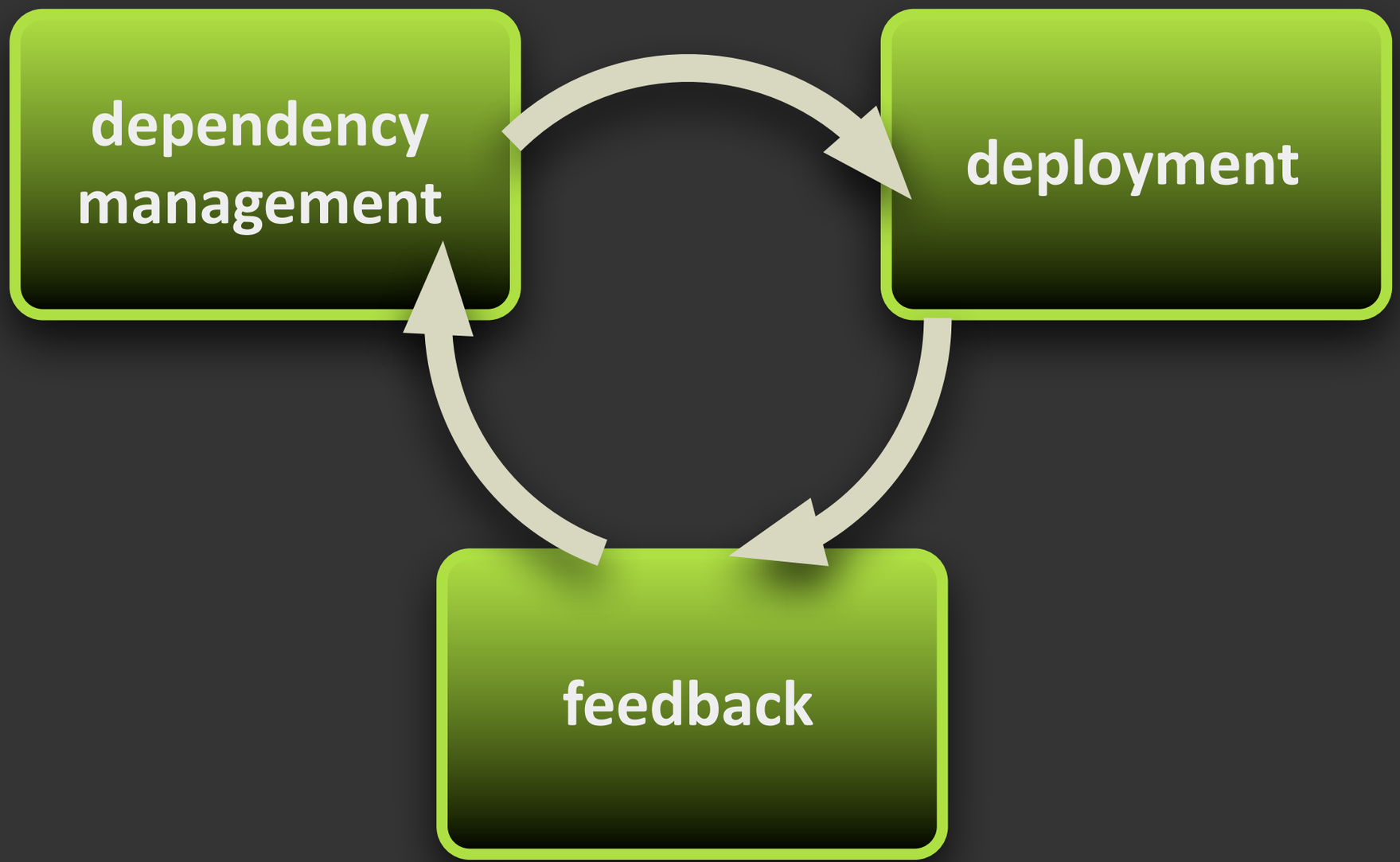
Why?

- Automate deployment
- Insight into who uses what
- History of each system
- Consistent development, testing, production
- Basis for several possible extensions

Topology


target
management agent
client
provisioning server
network
target
management agent
target
management agent
component repository

High level overview



High level overview

dependency management

Dependency Management

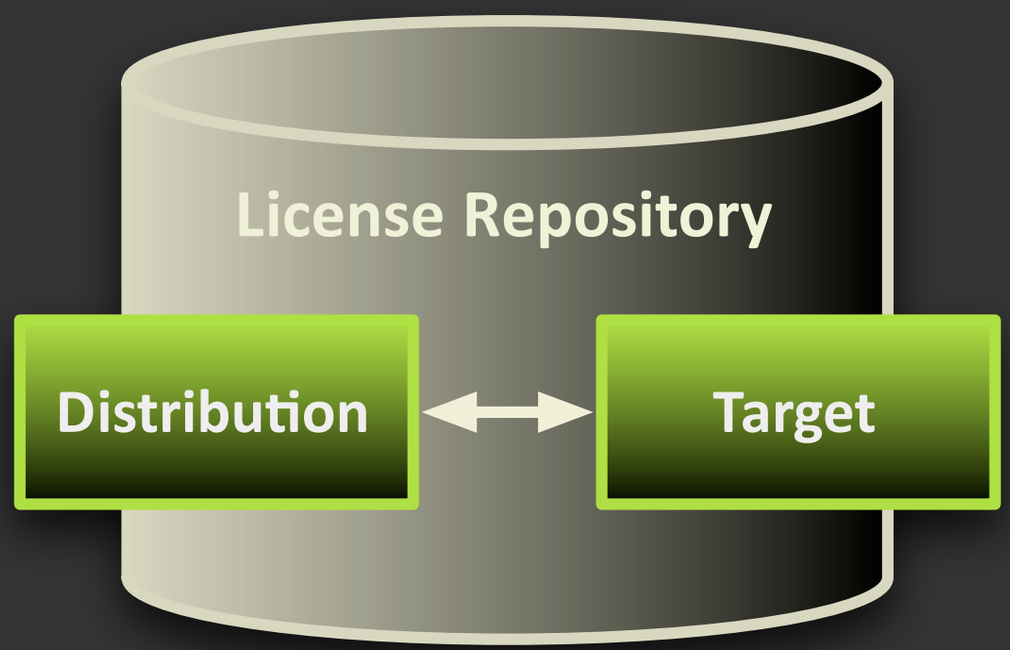
- Organizing artifacts
- Mapping them to targets

Organizing artifacts

- group artifacts: makes them manageable
- two levels: feature and distribution
- Analogy: IKEA catalog
- data is kept in “store repository”

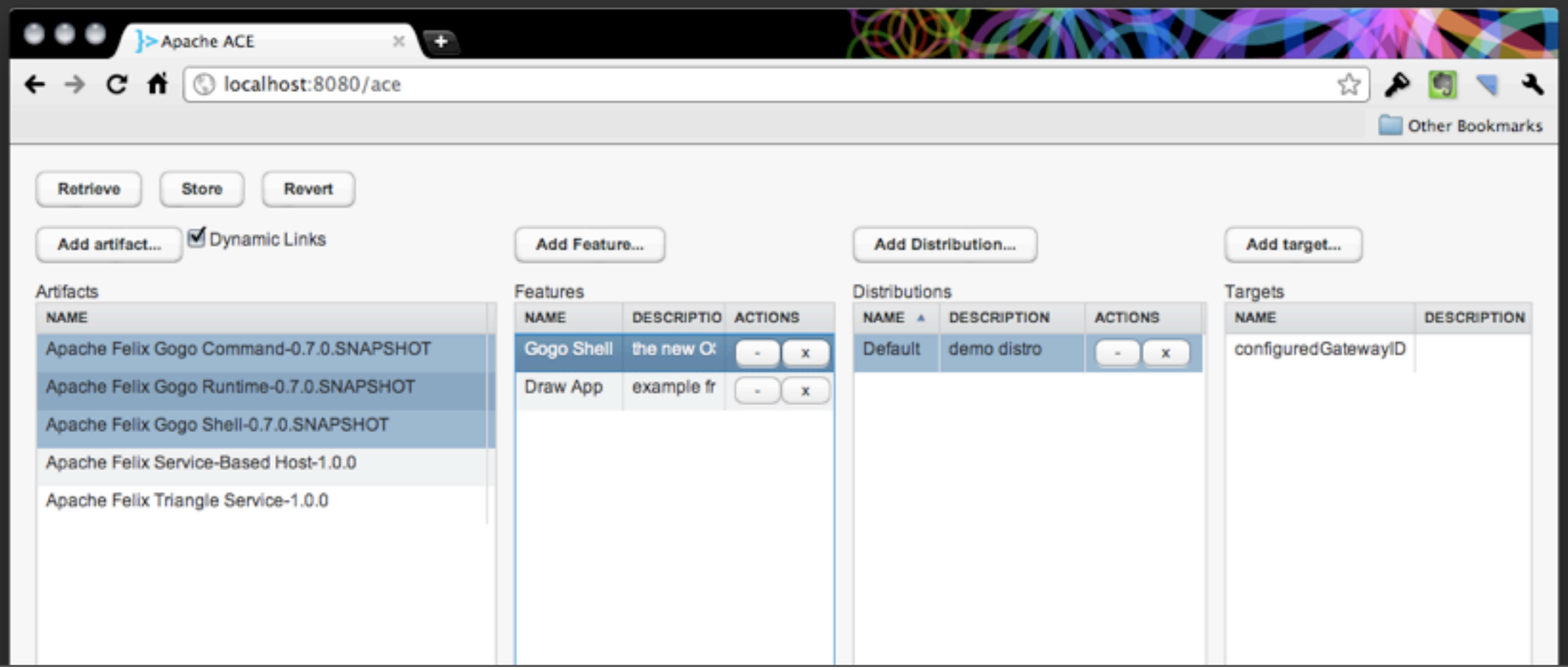
Mapping them onto targets

- mapping distributions to targets
- sometimes done by an external system
- data kept in “license repository”

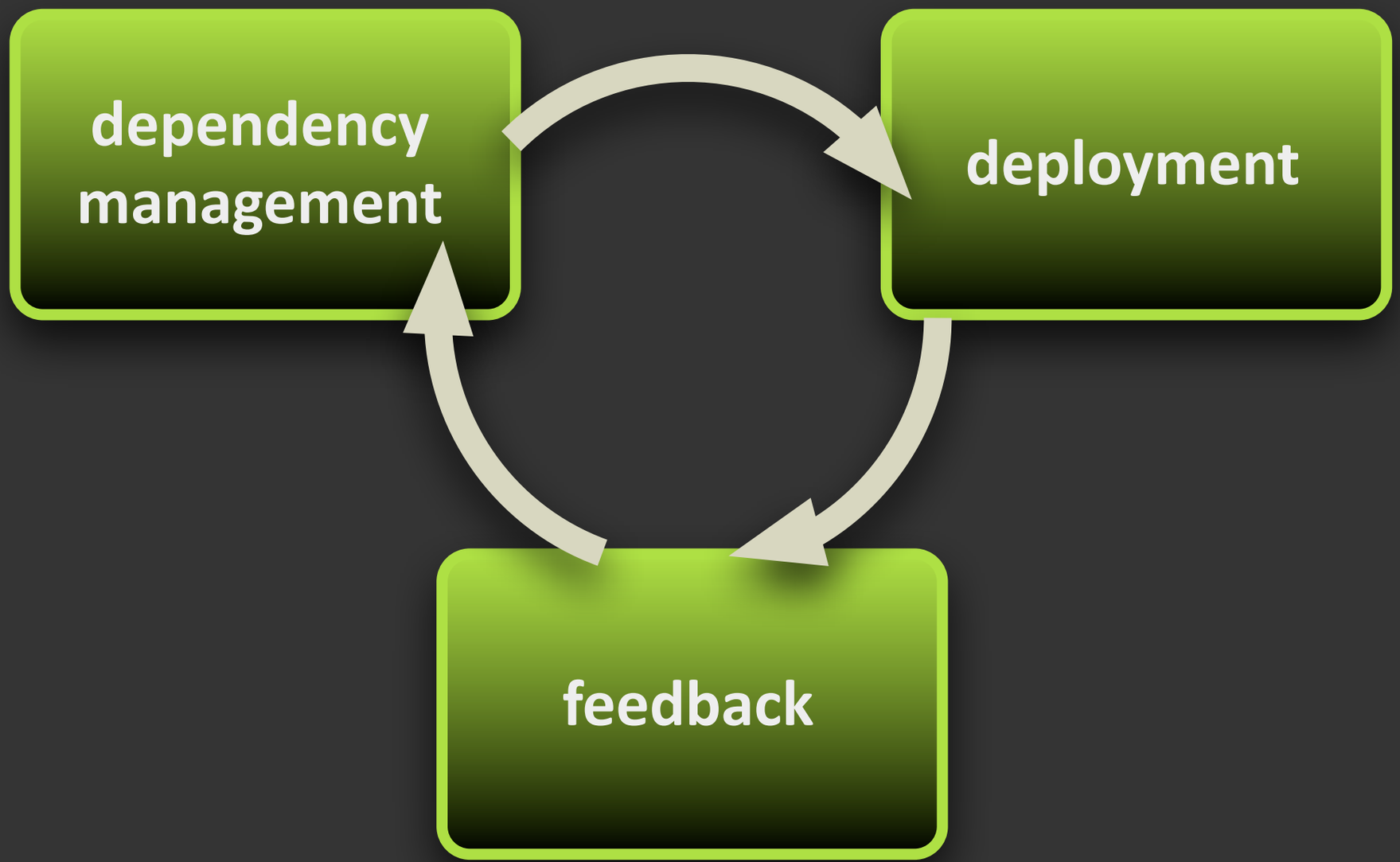


User Interface

- retrieve, modify and store
- interact with OBR
- web UI, REST UI and Java API available



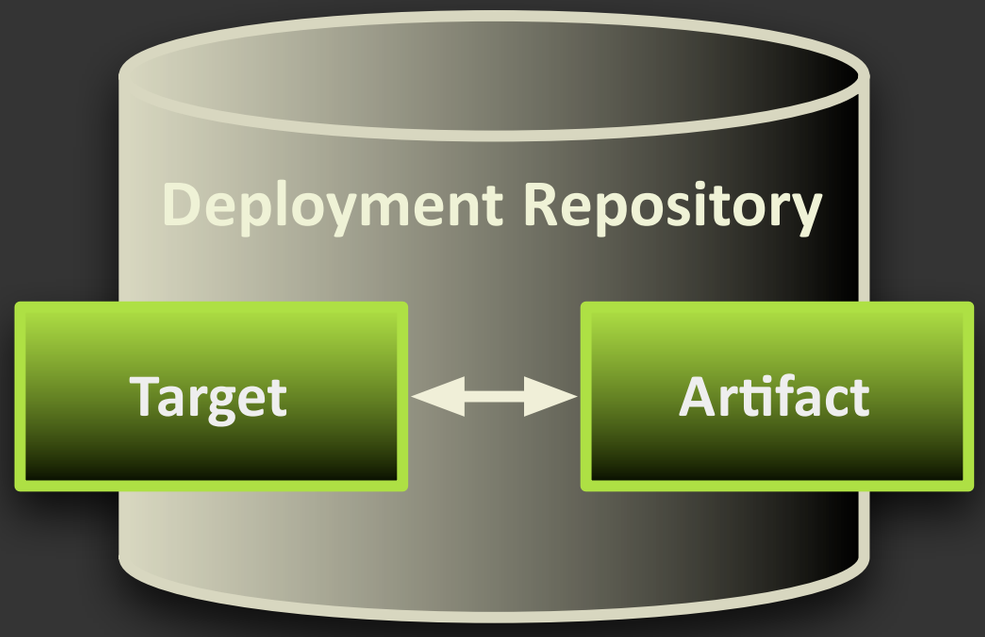
High level overview



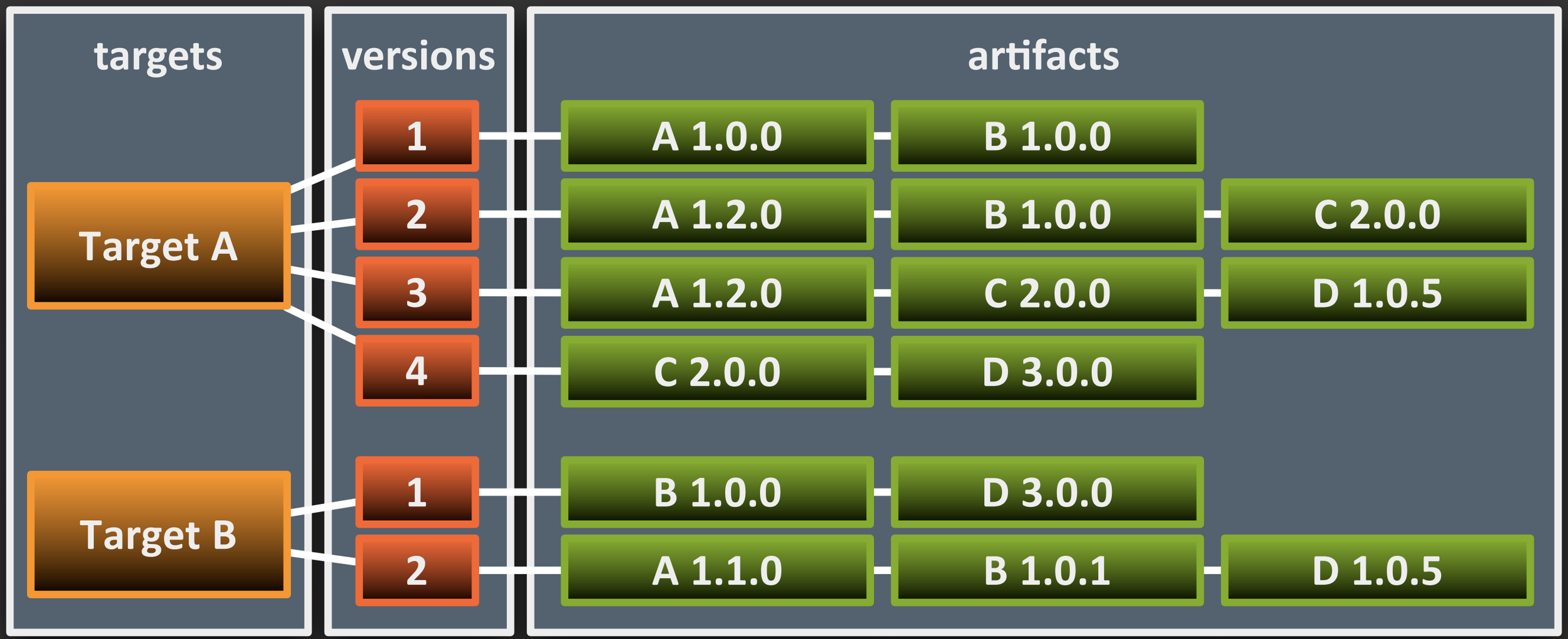
High level overview

Deployment

- deployment repository
- management agent



Deployment Repository

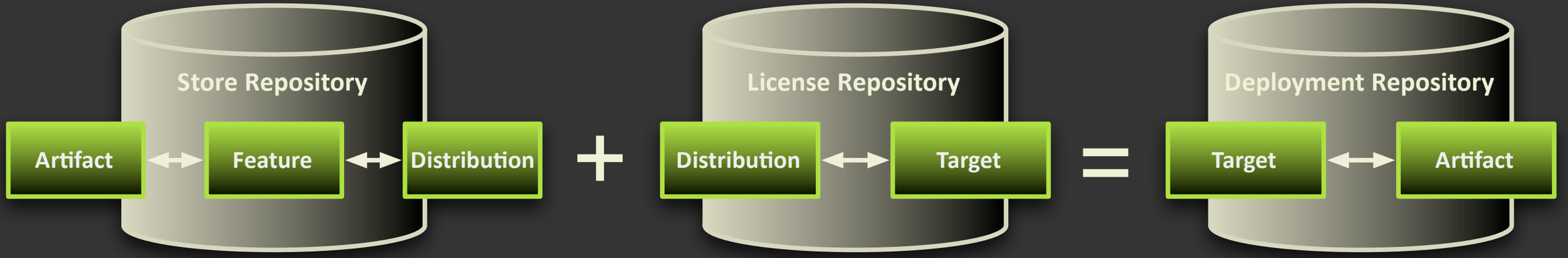


Management Agent

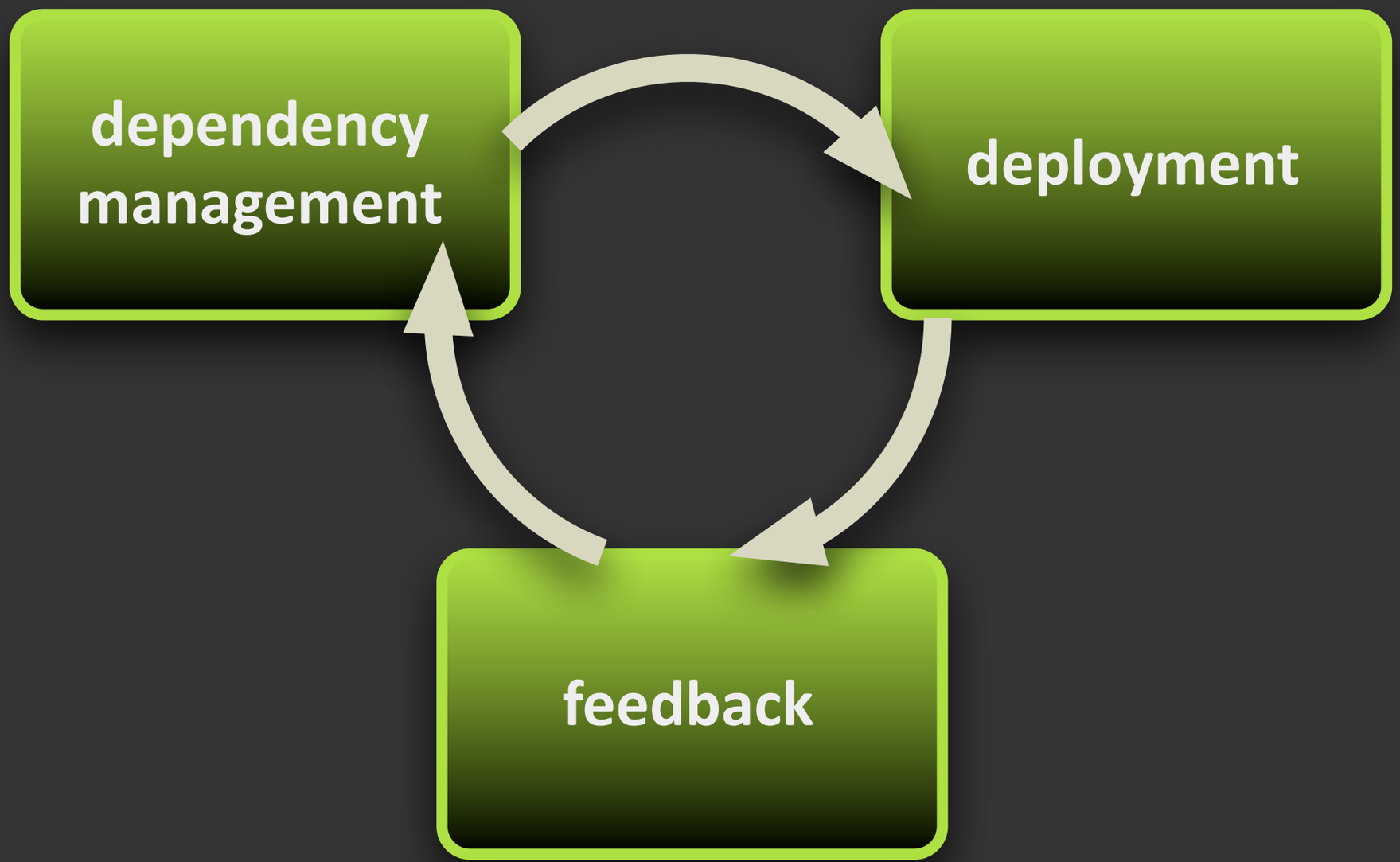
Deployment Admin

- deployment packages
- versioned set of artifacts
- transactional install/update
- fix packages provide deltas
- signing makes them secure
- extensible through resource processors
- AutoConfig defines configuration admin data

From dependency to deployment

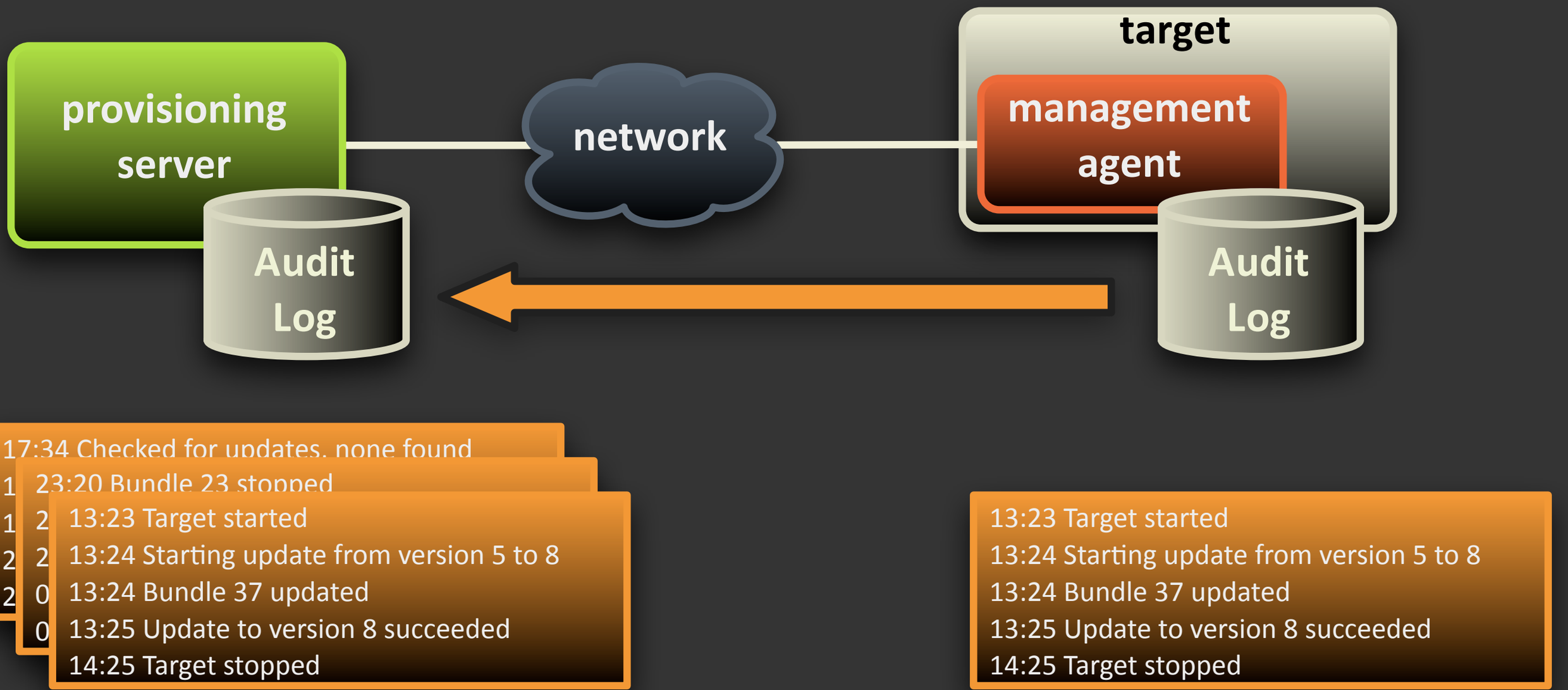


High level overview



High level overview

Feedback


provisioning server
network
target
management agent
Audit Log
Audit Log
17:34 Checked for updates, none found
23:20 Bundle 23 stopped
13:23 Target started
13:24 Starting update from version 5 to 8
13:24 Bundle 37 updated
13:25 Update to version 8 succeeded
14:25 Target stopped
13:23 Target started
13:24 Starting update from version 5 to 8
13:24 Bundle 37 updated
13:25 Update to version 8 succeeded
14:25 Target stopped


Targeting different
containers

OSGi

- Apache Felix

- Eclipse Equinox

- Knopflerfish

Glassfish

- Modularized the application server itself, based on Apache Felix

- Allows developers to deploy modular applications

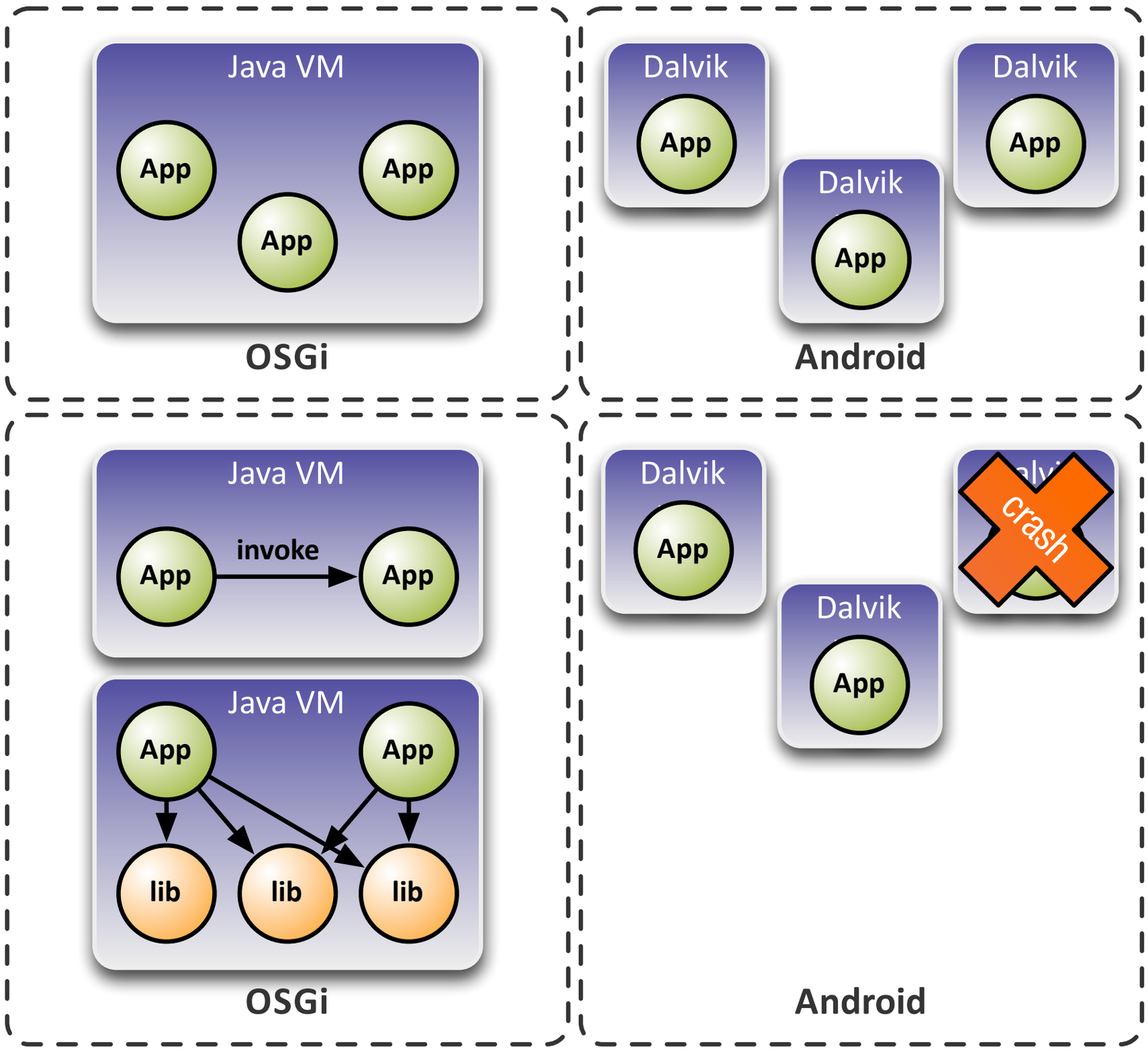
- Modular runtime for C

- Inspired by OSGi, with some obvious differences

- Works with deployment packages

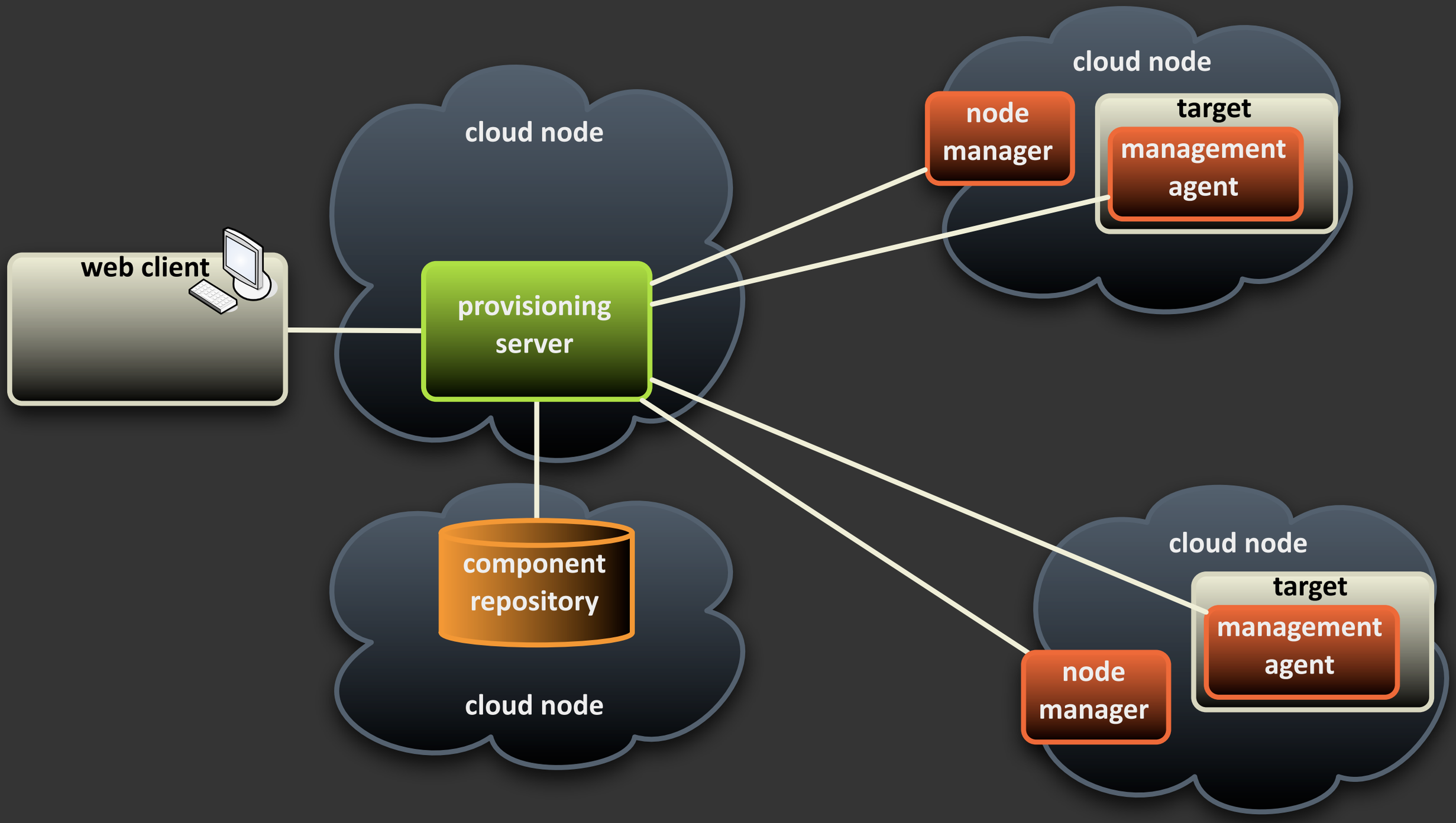
*) Apache Celix is currently incubating at the ASF

Android

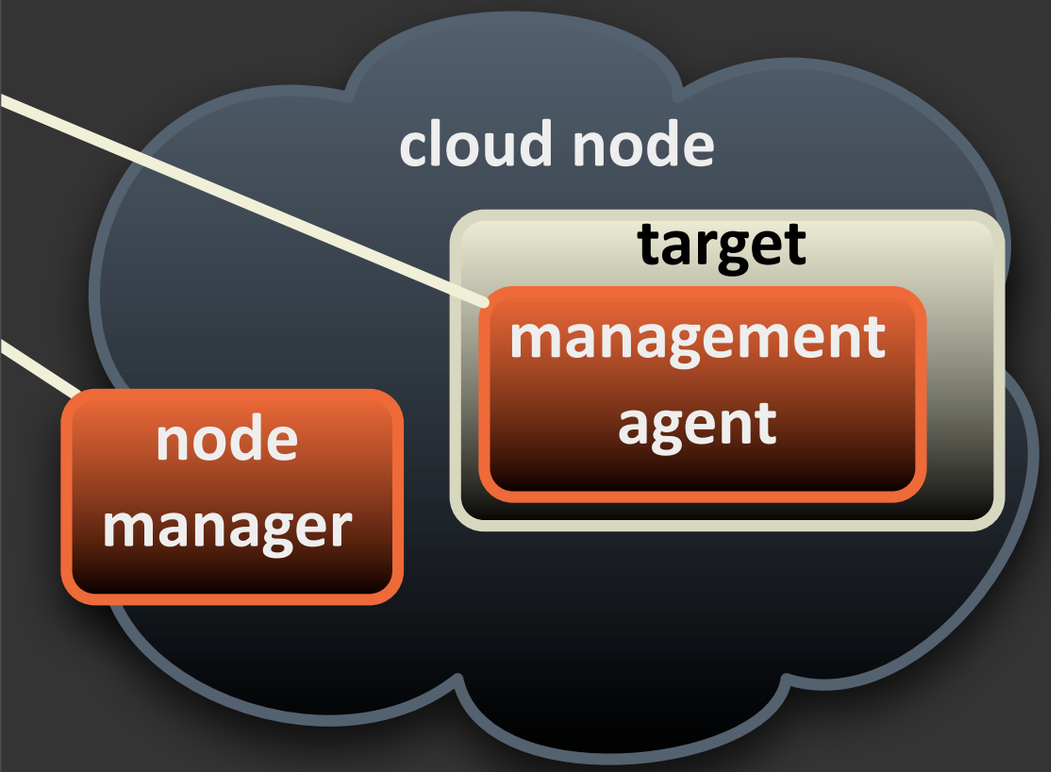

Java VM
App
App
App
OSGi
Dalvik
App
Dalvik
App
Dalvik
App
Android
Java VM
App
invoke
App
Java VM
App
App
lib
lib
lib
OSGi
Dalvik
App
Dalvik
App
crash
Android

Into the cloud

Cloud topology



Node Manager



- bootstraps the node
- launches targets
- measures performance data

Configuration

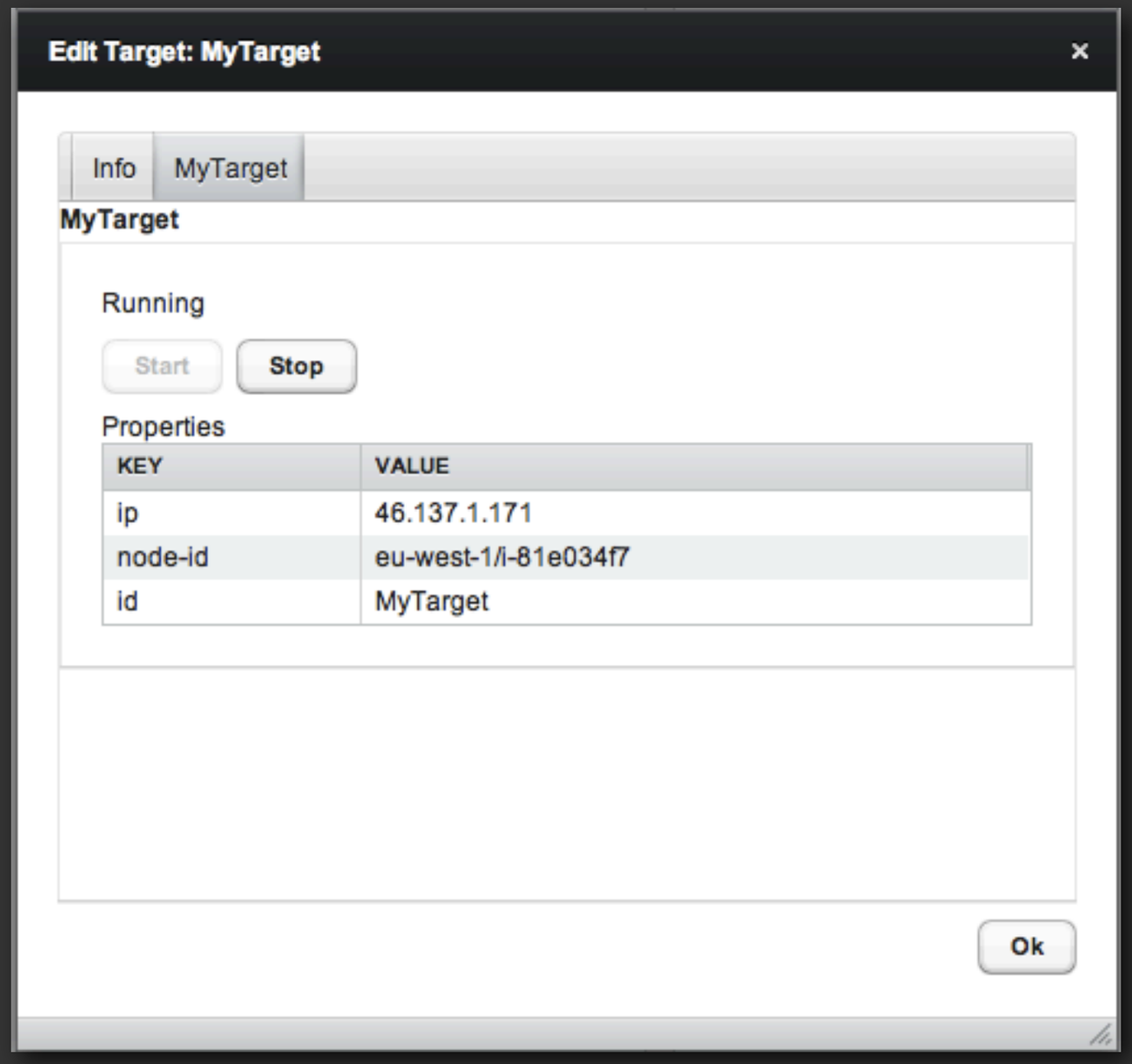
```
# Server URL, should be reachable for a cloud node
server=http://localhost:8080
# Note that AMIs are specific to an Amazon availability zone
amiId=ami-6a31041e
location=eu-west-1
# Your access key ID and secret access key (AWS console, top right menu "Security
Credentials")
accessKeyid=Your_access_key_here!
secretAccessKey=Your_secret_key_here!
# Tag prefix for instance names
tagPrefix=ace
# Use this bootstrap to use an Oracle VM instead of the OpenJDK one provided by Amazon
nodeBootstrap=cd ~; wget -Ojava.bin http://javadl.sun.com/webapps/download/AutoDL?
BundleId=43871 ;chmod +x java.bin;./java.bin /y; export PATH=`pwd`/jre1.6.0_23/bin:$PATH
# Open up any extra ports (comma separated list)
extraPorts=9090,9443
# Should we run the process as root? Only works if you're in the sudoers file.
runAsRoot=true
# List of extra artifacts that will be downloaded from the OBR
additionalObrDownloads=
# List of extra artifacts that will be downloaded from the specified URLs
externalDownloadUrls=
```

Launching a node

- Uses jclouds.org
- Implementation for Amazon EC-2
- Support for other clouds on roadmap

Launching a node

- Uses jclouds.org
- Implementation for Amazon EC-2
- Support for other clouds on roadmap

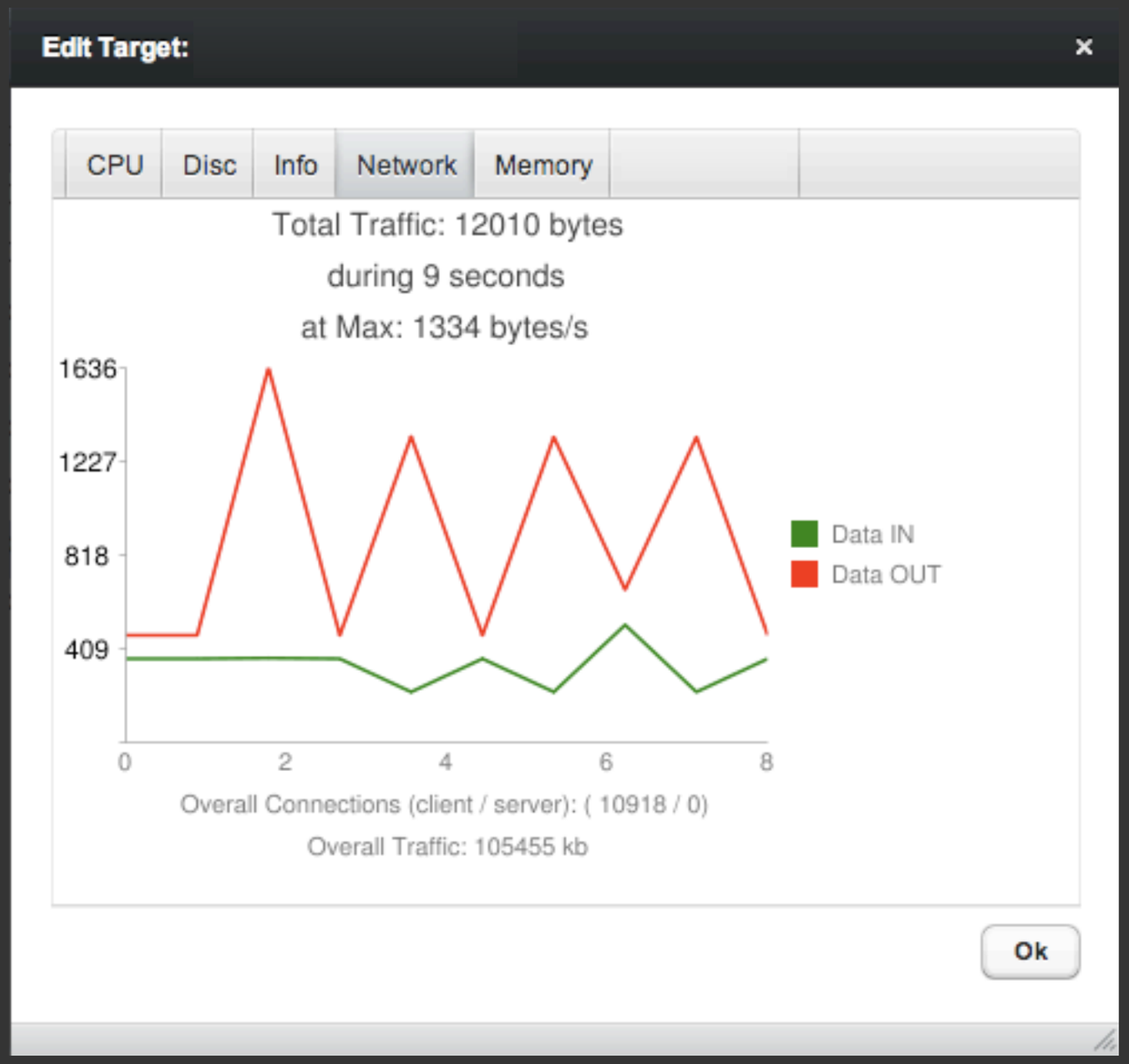


Monitoring

- Extensible with custom probes
- Aggregates data on server for off-line analysis
- Base for accounting and clustering

Monitoring

- Extensible with custom probes
- Aggregates data on server for off-line analysis
- Base for accounting and clustering

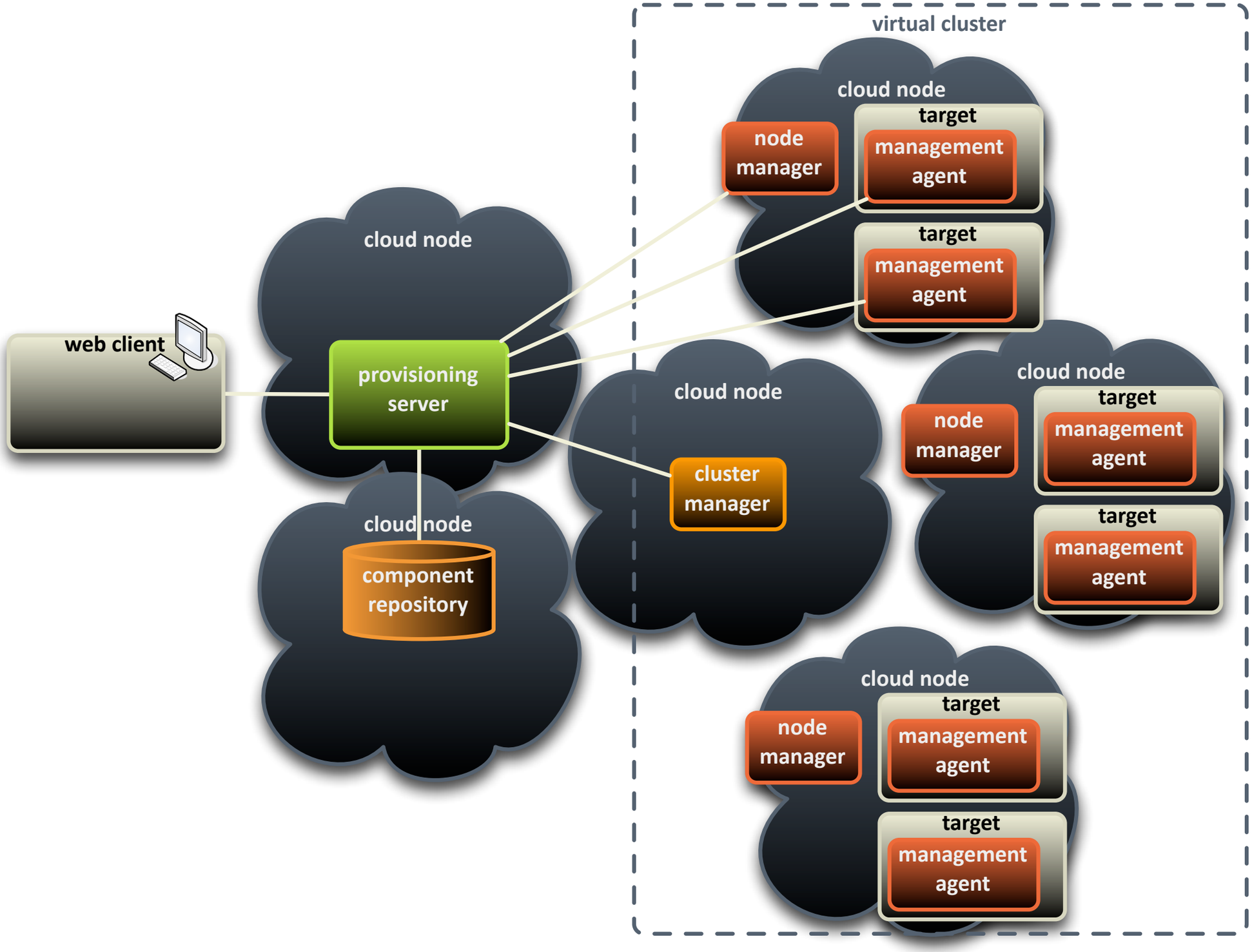


Node Launcher API

```
public interface NodeLauncher {
    /** Starts a new node with the given ID. */
    void start(String id) throws Exception;

    /** Starts a new node with the given ID. */
    void start(String id, NodeLauncherConfig config) throws Exception;

    /** Destroys the node with the given ID. */
    void stop(String id) throws Exception;

    /** If available, return a default configuration object. */
    NodeLauncherConfig getDefaultConfig();

    /** @return the currently set configuration object */
    NodeLauncherConfig getCurrentConfig();

    /** Retrieves properties from the node. Including 'ip', its IP address */
    Properties getProperties(String id) throws Exception;
}
```

Future directions

Dynamic Clustering



Integrate with

- Amdatu
  http://amdatu.org/

- Bndtools
  http://bndtools.org/

<shameless-plug>
Tomorrow, 3 - 4 pm - Parc 55 - Market Street
CON5055 - OSGi in the Cloud: A Case Study
</shameless-plug>

Marcel Offermans

Fellow at Luminis Technologies

@m4rr5

Paul Bakker

Architect at Luminis Technologies

@pbakker

Questions?