Deploying Heterogeneous Artifacts to the Cloud with OSGi

Neil Bartlett - Paremus
neil.bartlett@paremus.com
Motivation

- Good OSGi Bundles are Self-Describing
- Dependency Management, Resolving, Provisioning...
- Consistent Interface: Bundle Install/Uninstall/Update/Start/Stop, Config Admin, Metatype, Log Service...
- If I Have a Bundle, I Always Know How to Deploy It!
Unfortunately...

- Not All Software is Built in OSGi (yet). Lots of useful things are not even Java!
- How do we ensure this software is installed where it needs to go?
- Current state-of-the-art: assume it’s already there!
Proposal

- Wrap Native Artifacts in OSGi Bundles
- **Tie the Lifecycle.** Bundle Start => Launch, etc.
- Link to Standard OSGi Services: Configuration Admin, Metatype, Log Service...
Separation of Concerns

1. What and How Should we Run?
2. When, How Many, and How Configured?
3. Actually Run and Monitor It.
Separation of Concerns

1. What and How Should we Run? => Package Type
3. Actually Run and Monitor It => Watchdog
The PackageType

- Usually **Contains** the Native Parts, or **Knows** Where to Get Them
- **Installs** the Native Program
- Returns **Scripts** to the Packager for: start, stop, ping...
- Is **Platform Specific**
The Package Type

- One Bundle for Each Type/Platform Combo
- Use Generic Caps/Reqs, e.g.:

**Provide-Capability**: packager.type; 
  packager.type=mongodb; 
  version:Version=2.2.0

**Require-Capability**: osgi.native; 
  filter:=”(& 
  (osgi.native.osname=Linux) \ 
  (osgi.native.processor=x86-64) \ 
  )”
Example PackageType

@Component(properties = "package.type=mongodb")
public class MongoPackagerUNIX implements PackageType {
    @Override
    public PackageDescriptor create(Map<String, Object> properties, File data) throws Exception {
        MongoProperties config = Converter.cnv(MongoProperties.class, properties);
        PackageDescriptor pd = new PackageDescriptor();

        // Expand mongod executable
        File mongod = new File(data, "mongod");
        if (!mongod.isFile()) {
            IO.copy(getClass().getResource("/data/mongod"), mongod);
            run("chmod a+x " + mongod.getAbsolutePath());
        }

        // Construct the start command
        StringBuilder sb = new StringBuilder().append(mongod.getAbsolutePath());
        if (config.port() != 0)
            sb.append(" --port ").append(config.port());
        pd.startScript = sb.toString();
        // ...
        return pd;
    }
}
The ProcessGuard

- **Gathers** Process Config Properties
- Signals **When** to Start/Stop
- Receives State Change **Events** (started, crashed...)
- **Advertises** the Running Package (Optional)
MongoDB

Packager Manager

ProcessGuard

type=mongodb

Endpoint
uri=mongodb://10.0.0.1:27017/

service.exported.interfaces=*
The ProcessGuard

1. **Published** Service Means “Start the Process”
2. Packager Calls `getProperties()` to get Config
3. Passes Config to PackageType, gets **Script**
4. **Unpublished** Service Means “Stop the Process”
ProcessGuard Dependencies

- Use Declarative Services to Easily Create Dependencies for the Package
- E.g. Required Configuration
- Can Also Use Service References...
Example ProcessGuard

```java
@Reference(target = "(uri=http://*/rest)", type='*")
void setRestService(Endpoint ep, Map<String, String> props) throws Exception {
    this.restSvcUri = new URI(props.get("uri"));
}
```
Publishing an Endpoint

- Guard can Optionally Publish an Endpoint Service
- Just a Marker Interface with a URI Property, e.g.: `mongodb://10.0.0.1:27017/`
- Use OSGi Remote Services (RSA) Discovery to Publish Across the Network
MongoDB Client

- `mongodb://10.0.0.1/`
- `mongodb://10.0.0.2/`
- `mongodb://10.0.0.3/`

OSGi Remote Services (RSA) Discovery

Mongo Client
Consuming Endpoints

- Guards can be Endpoint Consumers as Well
- Example, Nginx
A request comes from a web browser, such as Firefox, to an Nginx Load Balancer. The Load Balancer then forwards the request to one of the web servers. This is a diagram illustrating the process of load balancing in a web server setup.
Nginx

- Guard Tracks Endpoints with a DS Multiple/Dynamic Reference
- PackageType Rewrites Nginx Config and Sends a UNIX Signal
Watchdog

- Java Program that Sits **Between** OSGi and Native Package
- **Execs** Native as a Child Process
- Death of Watchdog $\Rightarrow$ **Death** of Native Package
- **Heartbeat** UDP Between OSGi and Watchdog
Native Process

OSGi Application

Watchdog

Native Process

heartbeatt

exec
Watchdog

- Must Be as **Simple** As Possible!
- **Mustn’t Crash**
- Might Need Elevated **Privileges** (e.g. setuid)
DEMO...
Tooling

- Bndtools Highly Recommended
- Project Templates Coming Soon...
- It’s Just Plain Declarative Services!
Conclusion
Availability

- API is Open Source (EPL): github.com/bndtools/bndtools-rt/
- Paremus implementation releasing later this week, stay tuned: http://blogs.paremus.com
- Register interest: info@paremus.com
- Examples: MongoDB, Mosquitto, Nginx, Derby, Custom Play App
- In progress: RabbitMQ, Riak