Apache Karaf in the enterprise

JB Onofré
jbonofre@apache.org
@jbonofre

APACHECON North America
Sept. 24-27, 2018
Who am I?

Jean-Baptiste Onofré <jbонофрэ@apache.org>

- Software Architect/Fellow at Talend
- Member of the Apache Software Foundation
- PMC member and committer for ~ 20 Apache projects (Karaf, Camel, ActiveMQ, Felix, Aries, Beam, Incubator, ...)

The Apache Software Foundation

Talend
Apache Karaf?

- Application runtime
- Lightweight & modular
- Very customisable
- Several packaging (standalone, custom distribution, docker)
- Static (immutable) or dynamic (mutable) bootstrapping
- Executable on premise, on cloud (docker, cellar)
- Bunch of features and extensions (including Karaf subprojects)
Apache Karaf in the enterprise

Different perspectives and needs:

- **Developers**
  - Features
  - Easy & fun
  - Tools (debugging, profiling, ...)

- **DevOps**
  - Packaging
  - Management (installing/updating containers, runtime, applications) & monitoring
  - Scaling
  - Integration in the ecosystem

- **End users**
  - Business ready tools
  - Insight in the business activity
Karaf for the devs

- Business applications
- Programming models
- Packaging
- Specifications & features
- Tools (debug, watcher, ...)
- Examples, documentation, karaf-boot
- Active user mailing list
Devs: business applications

- Backend, service/microservice approach
- Frontend
- IoT & Integration (with Apache Camel)
Devs: programming models

- **WebApp/WAR**
  karaf@root()> feature:install war
  karaf@root()> bundle:install -s webbundle:mvn:my/app/1.0/war?Web-ContextPath=my

- **Spring (any version from 3 to 5)**
  karaf@root()> feature:install spring
  karaf@root()> bundle:install -s wrap:mvn:my/app/1.0

- **CDI (OpenWebBeans & Weld)**
  karaf@root()> feature:install pax-cdi-weld
  karaf@root()> bundle:install -s wrap:mvn:my/app/1.0

- **OSGi (pure & native)**
  karaf@root()> bundle:install -s mvn:my/bundle/1.0

- **Blueprint (Aries & Gemini)**
  karaf@root()> feature:install aries-blueprint
  karaf@root()> bundle:install -s mvn:my/bundle/1.0

- **SCR**
  karaf@root()> feature:install scr
  karaf@root()> bundle:install -s mvn:my/bundle/1.0
Devs: artifacts packaging

- Regular jar (wrapping)
  
  ```bash
  karaf@root()> bundle:install -s wrap:mvn:my/app/1.0
  ```

- Regular war
  
  ```bash
  karaf@root()> bundle:install -s webbundle:mvn:my/app/1.0/war?Web-ContextPath=my
  ```

- OSGi bundles
  
  ```bash
  karaf@root()> bundle:install -s mvn:my/app/1.0
  ```

- Blueprint XML
  
  ```bash
  karaf@root()> bundle:install -s blueprint:mvn:my/app/1.0/xml
  ```

- Features
  
  ```bash
  karaf@root()> feature:repo-add mvn:my/app/1.0/xml/features
  karaf@root()> feature:install my-feature
  ```

- KAR
  
  ```bash
  karaf@root()> kar:install mvn:my/app/1.0/kar
  ```

- Custom artifacts (deployer and URL services)
Devs: specifications & features

- JNDI service (InitialContextFactory as service, names on services)
  
  karaf@root()> feature:install jndi

- JDBC service (DataSource as service, pooling DBCP, C3P0, transx, narayana, ...)
  
  karaf@root()> feature:install jdbc

- JMS service (ConnectionFactory as service, pooling DBCP, C3P0, transx, narayana, ...)

  Possible to install ActiveMQ broker directly in Karaf

  karaf@root()> feature:repo-add activemq

  karaf@root()> feature:install activemq-broker

- JPA service (EntityManager as service, abstracting OpenJPA, Hibernate, EclipseLink)
  
  karaf@root()> feature:install jpa

  karaf@root()> feature:install openjpa

- JTA (TransactionManager as service, abstracting Narayana, ...)

  karaf@root()> feature:install transaction

- JMX (MBean whiteboard pattern or MBean services)

  karaf@root()> feature:install management

- CDI (possible to use OSGi services as CDI injection)

- HTTP (HttpService service, servlet whiteboard pattern)

- JAXRS (CXF or Aries JAXRS whiteboard)

- JAXWS (CXF)

- Integration (Camel)
Devs: additional features & tooling

- Scheduler (executable runnable service or command as a cron)
  `karaf@root()> feature:install scheduler`
- Instances (create/clone Karaf instances)
  `karaf@root()> instance:create myinstance`
- Logging (abstract logging frameworks, centralized and dynamic configuration)
- Configuration (dynamic and centralized configuration)
- Hot deployment & deployer (extensible)
- Shell console (extensible)
- Maven plugin
  - Build and verify distribution
  - Run a distribution
  - Client and deploy on a running instance
  - Create docker image (WIP)
Devs: runtime tooling

- Remote debugging
  
  `bin/karaf debug`

- Developer commands
  
  `karaf@root()> bundle:diag`
  `karaf@root()> bundle:load-test`
  `karaf@root()> bundle:tree-show`
  `karaf@root()> system:framework -debug`

- Artifacts watcher (automatically update SNAPSHOT)
  
  `karaf@root()> bundle:watch *`

- Shell scripting

- Complete dump on demand (heapdump, threaddump, log, env, ...)
  
  `karaf@root()> dev:dump-create`
Devs: easy start and support

- Turnkey examples directly in the distribution
  https://github.com/apache/karaf/tree/master/examples
- Very active community
- Commercial support available
- Towards karaf-boot (WIP) providing annotations
Karaf for DevOps

- Packaging, provisioning & custom distribution
- Docker (image & feature)
- Security (JAAS, Syncope)
- Cellar cluster and distributed configuration
- Decanter for monitoring & alerting
- Cave for artifacts repository
- Cave for farm deployer
- Administration over SSH
- Toolkit for administration like auto diagnostic and dump creation
DevOps: Packaging, provisioning, custom distributions

- Mutable runtime provisioning options:
  - Hot deployment (deployer services)
  - Installing single artifact & bundle
  - Installing features
  - Installing KAR

- Immutable runtime provisioning options:
  - Static profile
  - Custom distribution (boot features, configuration; ...)

- Hybrid runtime provisioning options:
  - Custom distribution
  - Update the custom distribution on the fly
DevOps: docker

- Create docker image with provided tool
  
  vanilla: `assembly/docker/build.sh --from-release --karaf-version 4.2.1 --image-name karaf`
  
  custom: `assembly/docker/build.sh --from-local-dist --archive /path/mykaraf.tar.gz --image-name my-karaf`

- **docker** feature to manipulate Docker daemon from Karaf
  - `docker:search`, `docker:ps`, `docker:run`, `docker:pull`, `docker:push`, `docker:tag`

- **docker** feature can create a Docker image using your running Karaf instance
  `karaf@root()> docker:provision mykaraf`

- Karaf HTTP proxy service to proxy Docker container port in Karaf
  `karaf@root()> http:proxy-add /elasticsearch http://localhost:9200`

- Official Apache Karaf Docker image (WIP)
DevOps: security

- Dynamic keystore loading
- Complete RBAC for commands, MBeans, services
- Auditing of all actions performed in Karaf
- JAAS Realms with dedicated commands
- Provided LoginModules
- Support Apache Syncope
DevOps: Karaf Cave

- Artifacts repository
  - Bundles Repository
  - Maven Repository
  - Docker hub (WIP)
- Easy to install
- Karaf Features Gateway
- Deployer to manage Karaf instances farm
- REST API
DevOps: Karaf Cave architecture

Repositories

Features Gateway

Deployer

Karaf

mvn/OBR resolution
resolve/download
manage/provision
deploy/upload
DevOps: Karaf Cellar

- Clustering deployment solution
- Distributed configuration
- Distributed administration (bundles, features, ...)
- Replication policies (no SPOF)
- Cluster HTTP Load balancing
- Cluster log service
- Distributed OSGi & Cluster RPC
DevOps: Karaf Cellar architecture
DevOps: management

- SSH
- Remote management (MBean server with RBAC)
- Remote debugging
- WebConsole
DevOps: Karaf Decanter (monitoring & alerting)

- Multipurpose
  - Activity data collection (metrics, log, ...)
  - Auditing
  - Alerting
  - BAM (business users)
- Collect data sent to a dispatcher
- Dispatch and check data (alerting)
- Append data to a backend
- Easy to install
- Dynamic
- Extensible
DevOps: Karaf Decanter architecture

Collector

harvest data

Dispatcher / Checker

send alerts to backend

Appender

send data to backend

Alerter
## DevOps: Karaf Decanter

### collectors/appenders/alerters

<table>
<thead>
<tr>
<th>Collectors</th>
<th>Appenders</th>
<th>Alerters</th>
</tr>
</thead>
<tbody>
<tr>
<td>Camel</td>
<td>Camel</td>
<td>Camel</td>
</tr>
<tr>
<td>Dropwizard</td>
<td>Cassandra</td>
<td>Email</td>
</tr>
<tr>
<td>EventAdmin</td>
<td>Dropwizard</td>
<td>Log</td>
</tr>
<tr>
<td>File, Log, Log4j Socket</td>
<td>Elasticsearch</td>
<td>… any appender</td>
</tr>
<tr>
<td>JDBC</td>
<td>File, Log</td>
<td></td>
</tr>
<tr>
<td>JMS, MQTT, Kafka</td>
<td>JDBC</td>
<td></td>
</tr>
<tr>
<td>JMX</td>
<td>JMS, MQTT, Kafka, Redis</td>
<td></td>
</tr>
<tr>
<td>System</td>
<td>MongoDB, OrientDB</td>
<td></td>
</tr>
<tr>
<td>REST, Socket</td>
<td>REST, Socket</td>
<td></td>
</tr>
</tbody>
</table>
DevOps: Karaf Decanter
DevOps: cloud ready

- Docker support
- Apache jClouds features supporting several providers (blobstore service, ...)
- Karaf Cellar supports jClouds and Kubernetes discovery
- Karaf Cellar is able to distribute applications and configuration on instances located on premise or on cloud
- Karaf Cave Deployer with jClouds can provision instance on cloud providers (WIP)
- Karaf Cave Deployer with jClouds can provision applications and configurations on a running Karaf instance on cloud (WIP)
DevOps: cloud management

- SSH/JMX to any cloud instance
- Interact with Docker directly from Karaf
- Use a local Karaf Cellar instance to manage remote cloud instances
- Karaf Decanter can monitor (harvesting JMX, gathering log, ...) on premise or cloud instances
Karaf provides solution customizable for business/end users

- Karaf Decanter as BAM solution (optionally with big data analytics)
- Karaf Vineyard as API Management solution
Business users: Karaf Vineyard (API Management)

- API Management
- Gateway, dual API: REST & GraphQL
- Registry (resources, schema, policies)
- Policies (security, QoS, ...)
- Discovery (OpenAPI, Swagger, GraphQL, ...)

Business users: Karaf Vineyard architecture

- **Karaf Decanter**
- **Registry (API, schema, policies)**
  - bootstrap API resources
  - target endpoints, policies
- **Gateway (REST & GraphQL)**
  - populate
- **Discovery Plugin**

- reporting metric
Business users: Karaf Decanter as BAM

- Karaf Decanter can use custom data or log to follow business activity
- Easy way do business activity reporting and analytic
- Support alerting on business activity (fraud detection, ...)
Business users: Decanter with big data analytics

- Karaf Decanter can collect any kind of data on the fly
- Appender can be used to send data to big data backend (Kafka, HDFS (WIP))
- Marshaller can be used to transform internal data Map as CSV
- Use to distributed execution engines on the collected data (Apache Beam, Apache Spark, Apache Flink, ...)
Karaf Community

- WELCOME to Karaf!
- We love contributions and ideas!
- Updated website
- Periodical release cycle (~ every 3 months)
Apache

http://karaf.apache.org

GitHub mirrors:
  https://github.com/apache/karaf
  https://github.com/apache/karaf-cellar
  https://github.com/apache/karaf-cave
  https://github.com/apache/karaf-decanter
  https://github.com/jbonofre/karaf-boot
  https://github.com/jbonofre/karaf-vineyard

Mailing Lists:
  users@karaf.apache.org
  dev@karaf.apache.org