

apache karaf cookbook

Published: September 19, 2026 | Index ID: #-

Introduction

In the ever evolving world of Java application servers **Apache Karaf** stands out as a lightweight, modular runtime that brings OSGi flexibility to the enterprise. Whether you're a seasoned developer or just getting acquainted with the OSGi ecosystem, having a practical reference guide can make all the difference. That's why this article serves as a **cookbook**—a collection of tried and true recipes for deploying, managing, and optimizing applications on Apache Karaf. By following these step by step instructions, you'll gain confidence in harnessing Karaf's full potential, from basic setup to advanced configuration.

What Is Apache Karaf?

Apache Karaf is an OSGi-based runtime container that provides a lightweight, modular platform for building and deploying Java applications. It wraps the OSGi framework, adds a convenient command line interface (CLI), web console, and a set of built in features such as:

- Dynamic bundle installation and update
- Feature management with dependency resolution
- Integrated scripting with JavaScript, Groovy, and more
- Secure communication via SSL/TLS
- Support for multiple protocols (HTTP, HTTPS, SSH, JMX, etc.)

Because Karaf is built on top of Equinox (the OSGi implementation used by Eclipse), it inherits robust lifecycle management and modularity while keeping the footprint small.

Why a Cookbook Is Essential

While the official documentation is thorough, it can sometimes feel like a reference manual rather than a practical guide. A cookbook offers:

- Concise, ready to copy recipes for common tasks
- Contextual explanations of why each step matters
- Best practice tips that help avoid common pitfalls
- Examples of real world scenarios, from simple feature installation to complex multi tenant deployments

By treating Karaf as a kitchen where you mix and match components, you'll find it easier to experiment and iterate.

Getting Started: Installation

System Requirements

Before you begin, ensure your environment meets the following:

- Java 8 or newer (Java 11+ recommended for long term support)
- Operating system: Linux, macOS, Windows
- At least 2 /GB of RAM for a minimal installation

Download and Extraction

1. Visit the official download page.
2. Choose the latest stable release (e.g., apache-karaf-4.2.10.zip).
3. Unzip the archive to a directory of your choice.

Setting Environment Variables

On Unix like systems, add the following to your `~/.bashrc` or `~/.zshrc`:

```
export KARAF_HOME=/path/to/apache-karaf-4.2.10
export PATH=$PATH:$KARAF_HOME/bin
```

On Windows, set `KARAF_HOME` in System Properties and append `%KARAF_HOME%\bin` to the Path variable.

Starting Karaf

Run:

```
karaf
```

You'll see the `karaf>` prompt. The first boot will download necessary bundles and configure default settings.

Basic CLI Commands

The Karaf shell is powerful. Below are the essential commands you'll use frequently:

- `feature:list` – List installed features.
- `feature:install <feature>` – Install a feature.
- `bundle:list` – View all OSGi bundles.
- `bundle:start <id>` / `bundle:stop <id>` – Control bundle lifecycle.
- `config:list` – Show current configuration properties.
- `config:edit <pid>` – Edit a configuration.
- `log:tail` – Monitor logs in real time.

Installing Features

Features are packages of bundles and configurations that provide specific functionality. The Karaf feature repository includes many common use cases, from HTTP servers to database connectors.

Discovering Available Features

Run:

```
feature:list -l
```

The `-l` flag gives a long listing, showing descriptions and dependencies.

Installing a Feature

Example: Install the camel-http feature for building REST services.

```
feature:install camel-http
```

Karaf will resolve dependencies, download bundles, and start them automatically.

Uninstalling a Feature

To remove the feature:

```
feature:uninstall camel-http
```

All associated bundles will be stopped and uninstalled.

Deploying Your Own Bundles

When you create a new OSGi bundle, you can deploy it to Karaf in several ways.

Using the Deploy Folder

Drop your .jar file into \$KARAF_HOME/deploy. Karaf automatically installs and starts it. This is ideal for quick testing.

Using the Bundle:Install Command

For more control, use:

```
bundle:install -s mvn:com.example/mybundle/1.0.0
```

The -s flag starts the bundle immediately. Replace the coordinates with your Maven coordinates.

Using Features for Production

In production, it's best to package your bundle within a custom feature. Create a feature.xml file, then deploy it:

1. Place feature.xml in \$KARAF_HOME/etc/features.d.
2. Reload the feature repository:

```
feature:repo-add mvn:org.apache.karaf.features/standard/4.2.10/xml/features
```

Install the feature:

```
feature:install my-custom-feature
```

Configuration Management

Karaf uses the **Configuration Admin** service to manage runtime properties. You can edit configurations via the CLI, the web console, or directly in \$KARAF_HOME/etc.

Editing Configurations via CLI

To change a property:

```
config:edit org.apache.camel.component.http  
config:property set port 8080  
config:update
```

Replace the PID and property name as needed.

Using the Web Console

Navigate to <http://localhost:8181/system/console>. Under the “Configurations” tab, you can search, edit, and create

configurations through a user friendly interface.

Security Best Practices

Securing Karaf involves multiple layers: SSH authentication, HTTPS for the web console, and fine grained permissions via the OSGi `org.apache.karaf.jaas` module.

Enabling SSH Access

Karaf ships with an SSH daemon. Configure it in `$KARAF_HOME/etc/org.apache.karaf.shell.cfg`:

```
ssh.host=0.0.0.0
ssh.port=8101
ssh.hostKey=ssh/ssh_host_key
```

Restart Karaf to apply changes.

Setting Up HTTPS for the Console

Create a keystore:

```
keytool -genkeypair -alias karaf -keyalg RSA -keystore keystore.jks -storepass changeit
```

Then edit `org.apache.karaf.web.cfg`:

```
web.ssl.enabled=true
web.ssl.keystore=keystore.jks
web.ssl.keystore.password=changeit
```

After restarting, the console will be available at `https://localhost:8181/system/console`.

Fine Grained Permissions

Karaf uses JAAS for role based access. Edit `org.apache.karaf.jaas.cfg` to define roles and map them to users. Example:

```
org.apache.karaf.jaas.loginModule=org.apache.karaf.jaas.modules.SimpleLoginModule
org.apache.karaf.jaas.loginModule.options.users=admin,alice
org.apache.karaf.jaas.loginModule.options.passwords=admin,alice
org.apache.karaf.jaas.loginModule.options.roles=admin,dev
```

Integrating with External Systems

Karaf's modularity makes it easy to connect to databases, message brokers, and cloud services.

Database Connectivity

Use the `org.apache.karaf.jdbc` feature to set up JDBC drivers and `DataSource` configurations. Example for PostgreSQL:

```
feature:install jdbc-postgresql
config:edit org.apache.karaf.jdbc.ds
config:property set driverClassName org.postgresql.Driver
```

```
config:property set url jdbc:postgresql://localhost:5432/mydb
config:property set username dbuser
config:property set password dbpass
config:update
```

Messaging with ActiveMQ

Install the ActiveMQ feature:

```
feature:install activemq
```

Configure connection factories via the web console or CLI.

Advanced Topics

Hot Swapping and Live Updates

Karaf's `bundle:update` command allows you to replace a running bundle without downtime. For critical services, consider using the `bundle:refresh` command to reload packages safely.

Multiple Instances and Clustering

Karaf supports clustering through the Pax Red5 or Pax Web frameworks. You can deploy a cluster of Karaf instances behind a load balancer, each sharing the same feature set.

Containerization with Docker

Running Karaf in Docker simplifies deployment. A minimal Dockerfile looks like this:

```
FROM openjdk:11-jre-slim
ENV KARAF_VERSION=4.2.10
RUN wget https://archive.apache.org/dist/karaf/${KARAF_VERSION}/apache-karaf-${KARAF_VERSION}.tar.gz && \
    tar xzf apache-karaf-${KARAF_VERSION}.tar.gz -C /opt && \
    rm apache-karaf-${KARAF_VERSION}.tar.gz
WORKDIR /opt/apache-karaf-${KARAF_VERSION}
CMD ["bin/karaf", "start", "-m", "karaf"]
```

Build and run the container to expose the console on port 8181.

Troubleshooting Common Issues

- Feature installation fails – Check the dependency tree with `feature:repo-list` and ensure all required features are available.

Baca Juga (Artikel Terkait)

- [an online airline reservation information system case](#)

URL: <http://adulvibex.com/an-online-airline-reservation-information-system-case.pdf>

- [an offset algorithm for polyline curves timeguy](#)

URL: <http://adulvibex.com/an-offset-algorithm-for-polyline-curves-timeguy.pdf>

- [an officer and a gentlewoman](#)

URL: <http://adulvibex.com/an-officer-and-a-gentlewoman.pdf>

- [an ocean of air a natural history the atmosphere gabrielle walker](#)

URL: <http://adulvibex.com/an-ocean-of-air-a-natural-history-the-atmosphere-gabrielle-walker.pdf>

Tags: [#apache](#) [#karaf](#) [#cookbook](#)

