Octopus

Copyright 2013 The Netherlands eScience Center

Author: Jason Maassen (J.Maassen@esciencecenter.nl)

Version: Userguide v0.1, Octopus v0.9

Last modified: 27 June 2013

Copyrights & Disclaimers

What is it?

Octopus is a middleware abstraction library. It provides a simple programming interface to various pieces of software that can be used to access distributed compute and storage resources.

TODO: mention Java!

Why Octopus?

Octopus is developed by the Netherlands eScience Center as a support library for our projects. Several projects develop end-user applications that require access to distributed compute and storage resources. Octopus provides a simple API to access those resources, allowing those applications to be developed more rapidly. The experience gained during the development of these end-user applications is used to improve the Octopus API and implementation.

Installation

TODO: explain intallation procedure

Design

Octopus is designed with extensibility in mind. It uses a modular and layer design as shown in Figure 1.

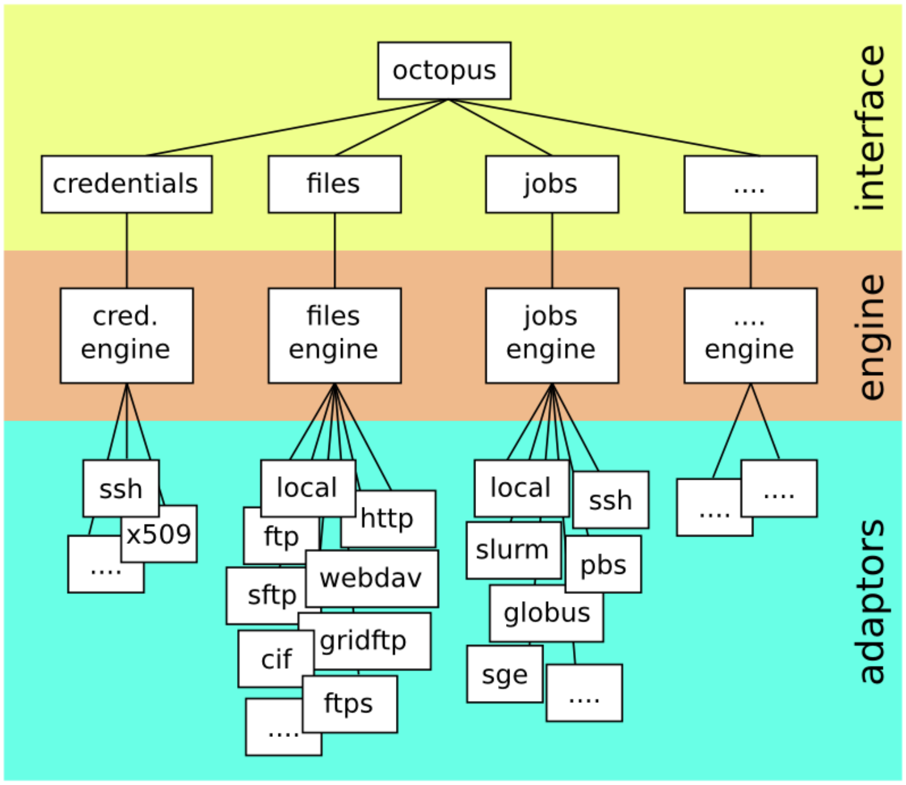

Figure 1: Octopus design

Octopus consists of three layers, an *interface layer*, an *engine layer* and an *adaptor layer*.

The *interface layer* is used by the application using octopus. It contains several specialized interfaces:

- Octopus: this is the main entry point used to retrieve the other interfaces.

- Files: contains functionality related to files, e.g., creation, deletion, copying, reading, writing, obtaining directory listings, etc.
- Jobs: contains functionality related to job submission, e.g., submitting, polling status, cancelling, etc.
- Credentials: contains functionality related to credentials. Credentials (such as a username password combination) are often needed to gain access to files or to submit jobs.

The modular design of octopus allows us to add additional interfaces in later versions, e.g., a Clouds interface to manage virtual machines, or a Networks interface to manage bandwidth-on-demand networks.

The *adaptor layer* contains the adaptors for the each of the middlewares that octopus supports. An *adaptor* offers a middleware specific implementation for the functionality offered by one of the interfaces in octopus.

For example, an adaptor may provide an *sftp* specific implementation of the functions in the octopus *file interface* (such as *copy* or *delete*) by translating each of these functions to *sftp* specific code and commands.

For each interface in octopus there may be multiple adaptors translating its functionality to different middlewares. To distinguises between these adaptors, octopus uses the *scheme* they support, such as "sftp", "http" or "ssh". There can be only one adaptor for each scheme.

The *engine layer* of octopus contains the "glue" that connects each interface to the adaptors that implement its functionality. When a function of the interface layer is invoked, the call will be forwarded to the engine layer. It is then the responsibility of the engine layer to forward this call to the right adaptor.

To perform this selection, the engine layer matches the *scheme* of the object on which the operation needs to be performed, to the *schemes* supported by each of the adaptors. When the schemes match, the adaptor is selected. This will be explained in more detail below (*TODO*).

Interfaces and datatypes

This section will explain each of the interfaces and related datatypes.

Package Structure

TODO explain package structure of public API:

- `nl.esciencecenter.octopus`

- `nl.esciencecenter.octopus.credentials`
- `nl.esciencecenter.octopus.files`
- `nl.esciencecenter.octopus.jobs`
- `nl.esciencecenter.octopus.exeptions`
- `nl.esciencecenter.octopus.util`

Octopus factory and interface

TODO explain functionality in package: `nl.esciencecenter.octopus`

```
public class OctopusFactory {
   public static Octopus newOctopus(Properties properties) throws OctopusException;
   public static void endOctopus(Octopus octopus) throws OctopusException;
   public static void endAll()
}
```

Credentials interface

TODO explain functionality in package: `nl.esciencecenter.octopus.credentials`

Files interface

TODO explain functionality in package: `nl.esciencecenter.octopus.files`

Jobs interface

TODO explain functionality in package: `nl.esciencecenter.octopus.jobs`

Exceptions

TODO explain functionality in package: `nl.esciencecenter.octopus.exceptions`

Utilities classes

TODO explain functionality in package: `nl.esciencecenter.octopus.util`

Examples

This section will show examples of how to use each of the interfaces in octopus..