Hazelcast Documentation

version 0.2-SNAPSHOT

Aug 26, 2016

Jet - Hazelcast | Documentation: version 0.2-SNAPSHOT

Publication date Aug 26, 2016

Contents

Chapter 1

Preface

Welcome to the Jet Reference Manual. This manual includes concepts, instructions, and samples to guide you on how to use Jet and build Jet applications.

As the reader of this manual, you must be familiar with the Java programming language and you should have installed your preferred Integrated Development Environment (IDE).

Chapter 2

Code Deployment

Code deployment feature enables you to distribute various resources to be used in your processors. Those resources can be a class file, JAR file or any type of file. By deploying your classes/JAR files to the cluster, you don't need to worry about restarting the cluster to update the business logic anymore. The only thing you need to do is just configure required class/JAR files for your Job and Hazelcast Jet will handle the distribution of your classes to the all nodes and loading of them when required. Hazelcast JET also provides isolation of the resources in the Job scope.

Note: The deployed classes will be available only to Jet Processors' classloader. If you'd like to use Hazelcast data structures as sources or sinks, the types you want to store in Hazelcast data structures must be present on the Hazelcast classpath.

2.1 Configuration

You can use any of the following methods to specify the deployment on your `JobConfig` object. The specified with resource will be uploaded to the all worker nodes when the Job starts it's execution.

“‘java /** * Add class to the job classLoader @param classes classes, which will be used during calculation */ public void addClass(Class... classes)

/** * Add JAR to the job classLoader @param url location of the JAR file */ public void addJar(URL url)

/** * Add JAR to the job classLoader @param url location of the JAR file * @param id identifier for the JAR file */ public void addJar(URL url, String id)

/** * Add JAR to the job classLoader @param file the JAR file */ public void addJar(File file)

/** * Add JAR to the job classLoader @param file the JAR file * @param id identifier for the JAR file */ public void addJar(File file, String id)

/** * Add JAR to the job classLoader @param path path the JAR file */ public void addJar(String path)

/** * Add JAR to the job classLoader @param path path the JAR file * @param id identifier for the JAR file */ public void addJar(String path, String id)

/** * Add resource to the job classLoader @param url source url with classes */ public void addResource(URL url)

/** * Add resource to the job classLoader @param url source url with classes * @param id identifier for the resource */ public void addResource(URL url, String id)

/** * Add resource to the job classLoader @param file resource file */ public void addResource(File file)

/** * Add resource to the job classLoader @param file resource file * @param id identifier for the resource */ public void addResource(File file, String id)

/** * Add resource to the job classLoader @param path path of the resource */ public void addResource(String path)

/** * Add resource to the job classLoader @param path path of the resource * @param id identifier for the resource */ public void addResource(String path, String id) “‘

2.2 Example Configuration

```
JobConfig config = new JobConfig();
String jarPath = "/path/to/jar/file.jar";
config.addJar(path);

Job job = JetEngine.getJob(hazelcastInstance, name, dag, config);
job.execute(); // The JAR file will be uploaded to the all nodes
// All the classes it contains will be loaded by the worker class loader
```

2.3 Retrieving deployed resources on the Processor

All the deployments will be available as a resource to the `Processor` class loader. To access those resources inside a processor, you need to get a reference to the class loader, then access to the resource either using `getResource(String name)` or `getResourceAsStream(String name)` methods. Resources can be accessed via their file names or with the identifiers (if assigned any while configuring the deployment).

2.3.1 Example

```
@Override
public boolean process(ProducerInputStream inputStream,
                       ConsumerOutputStream outputStream,
                       String sourceName, ProcessorContext processorContext) throws Exception {

    ClassLoader contextClassLoader = Thread.currentThread().getContextClassLoader();
    URL url = contextClassLoader.getResource("lookupTable");
    // URL points to the deployed `lookupTable` resource.
    ....
    return true;
}
```