Azkaban Documentation

Liang

Jul 19, 2018

Contents:

CHAPTER 1

What is Azkaban

Azkaban is a distributed Workflow Manager, implemented at LinkedIn to solve the problem of Hadoop job dependencies. We had jobs that needed to run in order, from ETL jobs to data analytics products.

See the *Getting Started* for more details.

CHAPTER 2

Features

- Distributed Multiple Executor
- MySQL Retry
- Friendly UI
- Conditional Workflow
- Data Triggers
- High Security
- Support plug-in extensions, from Web UI to job Execution
- Full Authorship management system

2.1 Getting Started

After version 3.0, we provide two modes: the stand alone "solo-server" mode and distributed multiple-executor mode. The following describes the differences between the two modes.

In solo server mode, the DB is embedded H2 and both web server and executor server run in the same process. This should be useful if one just wants to try things out. It can also be used on small scale use cases.

The multiple executor mode is for most serious production environment. Its DB should be backed by MySQL instances with master-slave set up. The web server and executor servers should ideally run in different hosts so that upgrading and maintenance shouldn't affect users. This multiple host setup brings in robust and scalable aspect to Azkaban.

- Set up the database
- Configure database to use multiple executors
- Download and install the Executor Server for each executor configured in database
- Install Azkaban Plugins
- Install the Web Server

Below are instructions on how to set Azkaban up.

2.1.1 Building from Source

Azkaban builds use Gradle (downloads automatically when run using gradlew which is the Gradle wrapper) and requires Java 8 or higher.

The following commands run on *nix platforms like Linux, OS X.

```
# Build Azkaban
./gradlew build

# Clean the build
./gradlew clean

# Build and install distributions
./gradlew installDist

# Run tests
./gradlew test

# Build without running tests
./gradlew build -x test
```

These are all standard Gradle commands. Please look at Gradle documentation for more info.

Gradle creates .tar.gz files inside project directories. eg. ./azkaban-solo-server/build/distributions/azkaban-solo-server-0.1.0-SNAPSHOT.tar.gz. Untar using tar -xvzf path/to/azkaban-*.tar.gz.

2.1.2 Getting started with the Solo Server

The solo server is a standalone instance of Azkaban and the simplest to get started with. The solo server has the following advantages.

- **Easy to install** - No MySQL instance is needed. It packages H2 as its main persistence storage.
- **Easy to start up** - Both web server and executor server run in the same process.
- **Full featured** - It packages all Azkaban features. You can use it in normal ways and install plugins for it.

Installing the Solo Server

Follow these steps to get started:

1. Clone the repo:

```
git clone https://github.com/azkaban/azkaban.git
```

2. Build Azkaban and create an installation package:

```
cd azkaban; ./gradlew build installDist
```

3. Start the solo server:

```
cd azkaban-solo-server/build/install/azkaban-solo-server; bin/azkaban-solo-start.sh
```

Azkaban solo server should be all set, by listening to `8081` port at default to accept incoming network request. So, open a web browser and check out `http://localhost:8081/`

4. Stop server:

```
bin/azkaban-solo-shutdown.sh
```

The solo-server installation should contain the following directories.

Folder	Description
bin	The scripts to start/stop Azkaban solo server
conf	The configuration files for Azkaban solo server
lib	The jar dependencies for Azkaban
extlib	Additional jars that are added to extlib will be added to Azkaban's classpath
plugins	the directory where plugins can be installed
web	The web (css, javascript, image) files for Azkaban web server

Inside the `conf` directory, there should be three files:

- `azkaban.private.properties` - Used by Azkaban to store secrets like Mysql password
- `azkaban.properties` - Used by Azkaban for runtime parameters
- `global.properties` - Global static properties that are passed as shared properties to every workflow and job.
- `azkaban-users.xml` - Used to add users and roles for authentication. This file is not used if the XmLUserManager is not set up to use it.

The `The azkaban.properties` file is the main configuration file.

Configuring HTTPS server (*Optional*)

Azkaban solo server by default doesn't use SSL. But you could set it up the same way in a stand alone web server. Here is how:

Azkaban web server supports SSL socket connectors, which means a keystore will have to be available. You can follow the steps to generate a valid jetty keystore provided at here. Once a keystore file has been created, Azkaban must be given its location and password. Within `azkaban.properties` or `azkaban.private.properties` (recommended), the following properties should be overridden.

```
jetty.keystore=keystore
jetty.password=password
jetty.keypassword=password
jetty.truststore=keystore
jetty.trustpassword=password
```

And configure ssl port in *azkaban.properties*:

```
jetty.ssl.port=8443
```

2.1.3 Getting started with the Multi Executor Server

Databasea setup

We suggest users to opt for **Mysql** as Azkaban database, because we build up a few Mysql connection enhancements to facilitate AZ set up, and strengthen service reliability:

- Install Mysql

  Installation of MySQL DB won't be covered by these instructions, but you can access the instructions on MySQL Documentation Site.

- Set up Mysql

  1. create database for Azkaban.:

  ```
  # Example database creation command, although the db name doesn't need to be
  ↪'azkaban'
  mysql> CREATE DATABASE azkaban;
  ```

  2. create a mysql user for Azkaban. For example,:

  ```
  # Example database creation command. The user name doesn't need to be 'azkaban
  ↪'
  mysql> CREATE USER 'username'@'%' IDENTIFIED BY 'password';
  # give the user INSERT, SELECT, UPDATE, DELETE permission on all tables in␣
  ↪the Azkaban db.
  mysql> GRANT SELECT,INSERT,UPDATE,DELETE ON azkaban.* to '<username>'@'%'␣
  ↪WITH GRANT OPTION;
  ```

  3. Mysql Packet Size may need to be re-configured. MySQL may have, by default, a ridiculously low allowable packet size. To increase it, you'll need to have the property max_allowed_packet set to a higher number, say 1024M. To configure this in linux, open /etc/my.cnf. Somewhere after mysqld, add the following:

  ```
  [mysqld]
  ...
  max_allowed_packet=1024M
  ```

  To restart MySQL, you can run:

  ```
  $ sudo /sbin/service mysqld restart
  ```

- Create the Azkaban Tables

  Run individual table creation scripts from latest table statements on the MySQL instance to create your tables.

  Alternatively, run create-all-sql-<version>.sql generated by build process. The location is the file is at `/Users/latang/LNKDRepos/azkaban/azkaban-db/build/distributions/azkaban-db-<version>`, after you build *azkaban-db* module by

```
cd azkaban-db; ../gradlew build installDist
```

Installing Azkaban Executor Server

Azkaban Executor Server handles the actual execution of the workflow and jobs. You can build the latest version from the master branch. See here for instructions on *Building from Source*.

Extract the package (executor distribution tar.gz from build folder) into a directory after gradle build. There should be the following directories.

Folder	Description
bin	The scripts to start/stop Azkaban solo server
conf	The configuration files for Azkaban solo server
lib	The jar dependencies for Azkaban
extlib	Additional jars that are added to extlib will be added to Azkaban's classpath
plugins	the directory where plugins can be installed

For quick start, we may directly use the Installation directory *azkaban/azkaban-exec-server/build/install/azkaban-exec-server* generated by gradle. we only need to change mysql username and password inside `azkaban.properties`:

```
# Mysql Configs
mysql.user=<username>
mysql.password=<password>
```

Then run:

```
cd azkaban-solo-server/build/install/azkaban-exec-server
./bin/start-exec.sh
```

After that, remember to activate the executor by calling:

```
cd azkaban-exec-server/build/install/azkaban-exec-server
curl -G "localhost:$(<./executor.port)/executor?action=activate" && echo
```

Then, one executor is ready for use. Users can set up multiple executors by distributing and deploying multiple executor installation distributions.

Installing Azkaban Web Server

Azkaban Web Server handles project management, authentication, scheduling and trigger of executions. You can build the latest version from the master branch. See here for instructions on *Building from Source*.

Extract the package (executor distribution tar.gz from build folder) into a directory after gradle build. There should be the following directories.

Folder	Description
bin	The scripts to start/stop Azkaban solo server
conf	The configuration files for Azkaban solo server
lib	The jar dependencies for Azkaban
web	The web (css, javascript, image) files for Azkaban web server

For quick start, we may directly use the Installation directory *azkaban/azkaban-web-server/build/install/azkaban-web-server* generated by gradle. we only need to change mysql username and password inside `azkaban.properties`:

```
# Mysql Configs
mysql.user=<username>
mysql.password=<password>
```

Then run

```
cd azkaban-web-server/build/install/azkaban-web-server
./bin/start-web.sh
```

Then, a multi-executor Azkaban instance is ready for use. Open a web browser and check out `http://localhost:8081/` You are all set to login to Azkaban UI.

2.2 Configurations

BlaBlaBla

2.3 User Guide

BlaBlaBla

CHAPTER 3

Community

Users and development team are in the Gitter.

Developers interested in getting more involved with Azkaban may join the mailing lists mailing lists, report bugs, and make contributions.