

# Templeton

## Table of contents

1 Templeton.....	2
------------------	---

# 1 Templeton

## 1.1 Templeton

### 1.1.1 Introduction

Templeton provides a REST-like web API for [HCatalog](#) and related Hadoop components. As shown in the figure below, developers make HTTP requests to access [Hadoop MapReduce](#), [Pig](#), [Hive](#), and [HCatalog DDL](#) from within applications. Data and code used by Templeton is maintained in [HDFS](#). HCatalog DDL commands are executed directly when requested. MapReduce, Pig, and Hive jobs are placed in queue by Templeton and can be monitored for progress or stopped as required. Developers specify a location in HDFS into which Templeton should place Pig, Hive, and MapReduce results.



### 1.1.2 URL format

Templeton resources are accessed using the following URL format:

`http://yourserver/templeton/v1/resource.format`

where "yourserver" is replaced with your server name, and "resource.format" is replaced by the Templeton resource name and output response format. Note that in Templeton version 0.1.0, JSON is the only supported response format.

For example, to check if the Templeton server is running you could access the following URL:

`http://www.myserver.com/templeton/v1/status.json`

### 1.1.3 Security

The current version of Templeton supports two types of security:

- Default security (without additional authentication)
- Authentication via [Kerberos](#)

#### 1.1.3.1 Standard Parameters

Every Templeton resource can accept the following parameters to aid in authentication:

- `user.name`: The user name as a string. Only valid when using default security.
- `SPNEGO credentials`: When running with Kerberos authentication.

### 1.1.4 WebHDFS and Code Push

Data and code that are used by Templeton resources must first be placed in Hadoop. The current version of Templeton does not attempt to integrate or replace existing web interfaces which can perform this task, like Web HDFS. (Integration of these functions in some way, perhaps forwarding, is planned for a future release.) This guide will sometimes refer to using Web HDFS to push code or data into Hadoop, but in all cases you can do this in whatever manner is most convenient for you.

### 1.1.5 Error Codes and Responses

The Templeton server returns the following HTTP status codes.

- **200 OK**: Success!
- **400 Bad Request**: The request was invalid.
- **401 Unauthorized**: Credentials were missing or incorrect.
- **404 Not Found**: The URI requested is invalid or the resource requested does not exist.
- **500 Internal Server Error**: We received an unexpected result.
- **503 Busy, please retry**: The server is busy.

Other data returned directly by Templeton is currently returned in JSON format. JSON responses are limited to 1MB in size. Responses over this limit must be stored into HDFS using provided options instead of being directly returned. If an HCatalog DDL command might return results greater than 1MB, it's suggested that a corresponding Hive request be executed instead.

### 1.1.6 Project Name

The Templeton project is named after the a character in the award-winning children's novel Charlotte's Web, by E. B. White. The novel's protagonist is a pig named Wilber. Templeton is a rat who helps Wilber by running errands and making deliveries.

## 1.2 Installation

### 1.2.1 Introduction

Templeton is deep in the middle of development and does not yet have a smooth install procedure. As such, this version of Templeton should only be installed expert developers.

### 1.2.2 Requirements

- [Ant](#), version 1.8 or higher
- [Tomcat](#), version 7.0 or higher
- [Hadoop](#), version 0.20.205.0
- [ZooKeeper](#). (Be sure to review and update the ZooKeeper related [Templeton configuration](#) as required.)
- [HCatalog](#),
  - To use the ddl resource, HCatalog must be installed in `/usr/local/hcat/bin/hcat`
  - To use the hive resource, see instructions [below](#) for placing HCatalog into the Hadoop distributed cache.
- [Pig](#), see instructions [below](#) for placing Pig into the Hadoop distributed cache.

### 1.2.3 Environment variable

TEMPLETON_HOME	Set to the Templeton directory
----------------	--------------------------------

### 1.2.4 Procedure

1. Ensure that the required related installations and environment variable are in place.
2. Review the [Templeton configuration](#) and update it as required. In particular ensure that site specific component installation locations are accurate.
3. Start Tomcat.
4. Build and install the Templeton war file using `ant install-war` from `$TEMPLETON_HOME`. The Tomcat webapps directory must be writable.
5. Check that your local install works. Assuming that Tomcat is running on port 8080, the following command would give output similar to that shown.

```
% curl -i http://localhost:8080/templeton/v1/status.json
HTTP/1.1 200 OK
```

```

Server: Apache-Coyote/1.1
Content-Type: application/json
Transfer-Encoding: chunked
Date: Tue, 18 Oct 2011 19:35:17 GMT

{"status": "ok", "version": "v1"}
%
```

### 1.2.5 Authentication

To run any of the authenticated API calls, sudo must be setup properly.

#### For Development

1. Run Tomcat as yourself.
2. Make sure that you can sudo as yourself. For example, `sudo -u $USER date`

#### For Production

1. Create user runner with permission to run the required commands. In the sudoers file:

```
runner ALL=(%hadoop) NOPASSWD: command1[,command2, ...]
```

allows runner to run the commands if they are owned by a user in the group hadoop.

2. Run Tomcat as runner.

### 1.2.6 Hadoop Distributed Cache

To avoid the installation of Pig and Hive everywhere on the cluster, Templeton gathers a version of Pig or Hive from the [Hadoop distributed cache](#) whenever those resources are invoked. To install the required components in the locations specified by the default configuration, follow these steps:

1. Create `hive.tar`. If you installed HCatalog as noted above, this this is done as follows:

```
cd /usr/local/hcat
tar cf /tmp/hive.tar .
```

2. Place `hive.tar` into the cache.

```
hadoop fs -put /tmp/hive.tar /user/templeton/hive.tar
```

3. Install Pig locally, for example in `/usr/local/pig`.
4. Create `pig.tar`:

```
cd /usr/local/pig
tar cf /tmp/pig.tar .
```

5. Place `pig.tar` into the cache.

```
hadoop fs -put /tmp/pig.tar /user/templeton/pig.tar
```

The location of these archives (tar files) in the cache, and the location of the installations inside the archives can be specified using the following Templeton configuration variables. (See the [Configuration](#) documentation for more information on changing Templeton configuration parameters.)

Name	Default	Description
<b>templeton.pig.archive</b>	hdfs:///user/templeton/pig.tar	The path to the Pig archive.
<b>templeton.pig.path</b>	pig.tar/bin/pig	The path to the Pig executable.
<b>templeton.hive.archive</b>	hdfs:///user/templeton/hive.tar	The path to the Hive archive.
<b>templeton.hive.path</b>	hive.tar/bin/hive	The path to the Hive executable.

### 1.3 Configuration

The configuration for Templeton merges the normal Hadoop configuration with the Templeton specific variables.

The Templeton specific configuration is split into three layers:

1. **templeton-default.xml** - All the configuration variables that Templeton needs. This file sets the defaults that ship with Templeton and should only be changed by Templeton developers.
2. **templeton-dist.xml** - The (possibly empty) configuration file that can set variables for a particular distribution, such as an RPM file.
3. **templeton-site.xml** - The (possibly empty) configuration file in which the system administrator can set variables for their Hadoop cluster.

The configuration files are loaded in this order with later files overriding earlier ones.

To find the configuration files, Templeton first attempts to load a file from the CLASSPATH and then looks in the directory specified in the TEMPLETON\_HOME environment variable.

In addition the configuration files may access the special environment variable `env` for all environment variables. For example, the pig executable could be specified using:

```
${env.PIG_HOME}/bin/pig
```

### 1.3.1 Variables

Name	Default	Description
<b>templeton.hadoop.config.dir</b>	<code>\$(env.HADOOP_CONFIG_DIR)</code>	The path to the Hadoop configuration.
<b>templeton.jar</b>	<code>\${env.TEMPLETON_HOME}/build/templeton/templeton-0.1.0-dev.jar</code>	The path to the Templeton jar file.
<b>templeton.libjars</b>	<code>\${env.TEMPLETON_HOME}/build/ivy/lib/templeton/zookeeper-3.3.4.jar</code>	Jars to add to the classpath.
<b>templeton.streaming.jar</b>	<code>hdfs:///user/templeton/hadoop-streaming.jar</code>	The hdfs path to the Hadoop streaming jar file.
<b>templeton.hadoop</b>	<code>\${env.HADOOP_PREFIX}/bin/hadoop</code>	The path to the Hadoop executable.
<b>templeton.pig.archive</b>	<code>hdfs:///user/templeton/pig.tar</code>	The path to the Pig archive.
<b>templeton.pig.path</b>	<code>pig.tar/bin/pig</code>	The path to the Pig executable.
<b>templeton.hcat</b>	<code>\${env.HCAT_PREFIX}/bin/hcat</code>	The path to the Hcatalog executable.
<b>templeton.hive.archive</b>	<code>hdfs:///user/templeton/hive.tar</code>	The path to the Hive archive.
<b>templeton.hive.path</b>	<code>hive.tar/bin/hive</code>	The path to the Hive executable.
<b>templeton.hive.properties</b>	<code>hive.metastore.local=false hive.metastore.uris=thrift://localhost:9933, hive.metastore.sasl.enabled=false</code>	Properties to set when running hive.
<b>templeton.sudo</b>	<code>/usr/bin/sudo</code>	The path to the sudo program.
<b>templeton.exec.encoding</b>	<code>UTF-8</code>	The encoding of the stdout and stderr data.
<b>templeton.exec.timeout</b>	<code>10000</code>	How long in milliseconds a program is allowed to run on the Templeton box.
<b>templeton.exec.max-procs</b>	<code>16</code>	The maximum number of processes allowed to run at once.

Name	Default	Description
<b>templeton.exec.max-output-bytes</b>	1048576	The maximum number of bytes from stdout or stderr stored in ram.
<b>templeton.exec.envs</b>	HADOOP_PREFIX, HADOOP_HOME	The environment variables passed through to sudo.
<b>templeton.zookeeper.hosts</b>	127.0.0.1:2181	ZooKeeper servers, as comma separated host:port pairs
<b>templeton.zookeeper.session-timeout</b>	30000	ZooKeeper session timeout in milliseconds
<b>templeton.callback.retry.interval</b>	10000	How long to wait between callback retry attempts in milliseconds
<b>templeton.callback.retry.attempts</b>	5	How many times to retry the callback
<b>templeton.zookeeper.cleanup.interval</b>	43200000	The maximum delay between a thread's cleanup checks
<b>templeton.zookeeper.cleanup.maxage</b>	604800000	The maximum age of a templeton job

## 1.4 Reference

### 1.4.1 Templeton Resources

Resource	Description
<a href="#">status</a>	Returns the Templeton server status.
<a href="#">ddl</a>	Performs an HCatalog DDL command.
<a href="#">mapreduce/streaming</a>	Creates and queues Hadoop streaming MapReduce jobs.
<a href="#">mapreduce/jar</a>	Creates and queues standard Hadoop MapReduce jobs.
<a href="#">pig</a>	Creates and queues Pig jobs.
<a href="#">hive</a>	Runs Hive queries and commands.
<a href="#">queue</a>	Returns a list of all jobids registered for the user.
<a href="#">GET queue/:jobid</a>	Returns the status of a job given its ID.



Resource	Description
<a href="#">DELETE queue/:jobid</a>	Kill a job given its ID.

## 1.4.2 GET status

### 1.4.2.1 Description

Returns the current status of the Templeton server. Useful for heartbeat monitoring.

### 1.4.2.2 URL

`http://www.myserver.com/templeton/v1/status.json`

### 1.4.2.3 Parameters

Only the [standard parameters](#) are accepted.

### 1.4.2.4 Results

Name	Description
<b>status</b>	"ok" if the Templeton server was contacted.
<b>version</b>	String containing the version number similar to "v1".

### 1.4.2.5 Example

#### Curl Command

```
% curl 'http://www.myserver.com/templeton/v1/status.json?user.name=charlotte'
```

#### JSON Output

```
{
  "status": "ok",
  "version": "v1"
}
```

## 1.4.3 POST ddl

### 1.4.3.1 Description

Performs an [HCatalog DDL](#) command. The command is executed immediately upon request. Responses are limited to 1MB. For requests which may return longer results consider using the [Hive resource](#) as an alternative.

### 1.4.3.2 URL

`http://www.myserver.com/templeton/v1/ddl.json`

### 1.4.3.3 Parameters

Name	Description	Required?	Default
<b>exec</b>	The HCatalog ddl string to execute	Required	None
<b>group</b>	The user group to use when creating a table	Optional	None
<b>permissions</b>	The permissions string to use when creating a table. The format is "rwxrw-r-x".	Optional	None

### 1.4.3.4 Results

Name	Description
<b>stdout</b>	A string containing the result HCatalog sent to standard out (possibly empty).
<b>stderr</b>	A string containing the result HCatalog sent to standard error (possibly empty).
<b>exitcode</b>	The exitcode HCatalog returned.

### 1.4.3.5 Example

#### Curl Command

```
% curl -d user.name=ctdean \
  -d 'exec=show tables;' \
  'http://www.myserver.com/templeton/v1/ddl.json'
```

#### JSON Output

```
{
  "stdout": "important_table
            my_other_table
            my_table
            my_table_2
            pokes
            ",
  "stderr": "WARNING: org.apache.hadoop.metrics.jvm.EventCounter is deprecated...
            Hive history file=/tmp/ctdean/hive_job_log_ctdean_201111111258_2117356679.txt"
```

```

    OK
    Time taken: 1.202 seconds
    ",
    "exitcode": 0
  }

```

## 1.4.4 POST mapreduce/streaming

### 1.4.4.1 Description

Create and queue an [Hadoop streaming MapReduce](#) job.

### 1.4.4.2 URL

`http://www.myserver.com/templeton/v1/mapreduce/streaming.json`

### 1.4.4.3 Parameters

Name	Description	Required?	Default
<b>input</b>	Location of the input data in Hadoop.	Required	None
<b>output</b>	Location in which to store the output data. If not specified, Templeton will store the output in a location that can be discovered using the <a href="#">queue</a> resource.	Optional	See description
<b>mapper</b>	Location of the mapper program in Hadoop.	Required	None
<b>reducer</b>	Location of the reducer program in Hadoop.	Required	None
<b>file</b>	Add an HDFS file to the distributed cache.	Optional	None
<b>define</b>	Set an Hadoop configuration variable using the syntax <code>define=NAME=VALUE</code>	Optional	None
<b>cmdenv</b>	Set an environment variable using the syntax <code>cmdenv=NAME=VALUE</code>	Optional	None
<b>arg</b>	Set a program argument.	Optional	None

Name	Description	Required?	Default
<b>callback</b>	Define a URL to be called upon job completion. You may embed a specific job ID into this URL using <code>\$jobId</code> . This tag will be replaced in the callback URL with this job's job ID.	Optional	None

#### 1.4.4.4 Results

Name	Description
<b>id</b>	A string containing the job ID similar to "job_201110132141_0001".
<b>info</b>	A JSON object containing the information returned when the job was queued. See the Hadoop documentation ( <a href="#">Class TaskController</a> ) for more information.

#### 1.4.4.5 Example

### Code and Data Setup

```
% cat mydata/file01 mydata/file02
Hello World Bye World
Hello Hadoop Goodbye Hadoop

% hadoop fs -put mydata/ .

% hadoop fs -ls mydata
Found 2 items
-rw-r--r--  1 ctdean supergroup      23 2011-11-11 13:29 /user/ctdean/mydata/file01
-rw-r--r--  1 ctdean supergroup      28 2011-11-11 13:29 /user/ctdean/mydata/file02
```

### Curl Command

```
% curl -d user.name=ctdean \
-d input=mydata \
-d output=mycounts \
-d mapper=/bin/cat \
-d reducer="/usr/bin/wc -w" \
'http://www.myserver.com/templeton/v1/mapreduce/streaming.json'
```

### JSON Output

```
{
  "id": "job_201111111311_0008",
  "info": {
    "stdout": "packageJobJar: [ ] [/Users/ctdean/var/hadoop/hadoop-0.20.205.0/share/
hadoop/contrib/streaming/hadoop-streaming-0.20.205.0.jar...
               templeton-job-id:job_201111111311_0008
               ",
    "stderr": "11/11/11 13:26:43 WARN mapred.JobClient: Use GenericOptionsParser for
parsing the arguments
               11/11/11 13:26:43 INFO mapred.FileInputFormat: Total input paths to
process : 2
               ",
    "exitcode": 0
  }
}
```

## Results

```
% hadoop fs -ls mycounts
Found 3 items
-rw-r--r--  1 ctdean supergroup          0 2011-11-11 13:27 /user/ctdean/mycounts/_SUCCESS
drwxr-xr-x  - ctdean supergroup          0 2011-11-11 13:26 /user/ctdean/mycounts/_logs
-rw-r--r--  1 ctdean supergroup        10 2011-11-11 13:27 /user/ctdean/mycounts/
part-00000

% hadoop fs -cat mycounts/part-00000
8
```

### 1.4.5 POST mapreduce/jar

#### 1.4.5.1 Description

Creates and queues a standard [Hadoop MapReduce](#) job.

#### 1.4.5.2 URL

<http://www.myserver.com/templeton/v1/mapreduce/jar.json>

#### 1.4.5.3 Parameters

Name	Description	Required?	Default
<b>jar</b>	Name of the jar file for Map Reduce to use.	Required	None
<b>class</b>	Name of the class for Map Reduce to use.	Required	None
<b>libjars</b>	Comma separated jar files to include in the classpath.	Optional	None

Name	Description	Required?	Default
<b>files</b>	Comma separated files to be copied to the map reduce cluster	Optional	None
<b>arg</b>	Set a program argument.	Optional	None
<b>define</b>	Set an Hadoop configuration variable using the syntax <code>define=NAME=VALUE</code>	Optional	None
<b>statusdir</b>	A directory where Templeton will write the status of the Map Reduce job. If provided, it is the caller's responsibility to remove this directory when done.	Optional	None
<b>callback</b>	Define a URL to be called upon job completion. You may embed a specific job ID into this URL using <code>\$jobId</code> . This tag will be replaced in the callback URL with this job's job ID.	Optional	None

#### 1.4.5.4 Results

Name	Description
<b>id</b>	A string containing the job ID similar to "job_201110132141_0001".
<b>info</b>	A JSON object containing the information returned when the job was queued. See the Hadoop documentation ( <a href="#">Class TaskController</a> ) for more information.

#### 1.4.5.5 Example

### Code and Data Setup

```
% hadoop fs -put wordcount.jar .
```

```
% hadoop fs -put transform.jar .

% hadoop fs -ls .
Found 2 items
-rw-r--r--    1 ctdean supergroup      23 2011-11-11 13:29 /user/ctdean/wordcount.jar
-rw-r--r--    1 ctdean supergroup      28 2011-11-11 13:29 /user/ctdean/transform.jar
```

## Curl Command

```
% curl -d user.name=ctdean \
-d jar=wordcount.jar \
-d class=org.myorg.WordCount \
-d libjars=transform.jar \
-d arg=wordcount/input \
-d arg=wordcount/output \
'http://www.myserver.com/templeton/v1/mapreduce/jar.json'
```

## JSON Output

```
{
  "id": "job_201111121211_0001",
  "info": {
    "stdout": "templeton-job-id: job_201111121211_0001",
    "stderr": "",
    "exitcode": 0
  }
}
```

### 1.4.6 POST pig

#### 1.4.6.1 Description

Create and queue a [Pig](#) job.

#### 1.4.6.2 URL

<http://www.myserver.com/templeton/v1/pig.json>

#### 1.4.6.3 Parameters

Name	Description	Required?	Default
<b>execute</b>	String containing an entire, short pig program to run.	One of either "execute" or "file" is required	None
<b>file</b>	HDFS file name of a pig program to run.	One of either "exec" or "file" is required	None
<b>arg</b>	Set a program argument.	Optional	None

Name	Description	Required?	Default
<b>files</b>	Comma separated files to be copied to the map reduce cluster	Optional	None
<b>statusdir</b>	A directory where Templeton will write the status of the Pig job. If provided, it is the caller's responsibility to remove this directory when done.	Optional	None
<b>callback</b>	Define a URL to be called upon job completion. You may embed a specific job ID into this URL using \$jobId. This tag will be replaced in the callback URL with this job's job ID.	Optional	None

#### 1.4.6.4 Results

Name	Description
<b>id</b>	A string containing the job ID similar to "job_201110132141_0001".
<b>info</b>	A JSON object containing the information returned when the job was queued. See the Hadoop documentation ( <a href="#">Class TaskController</a> ) for more information.

#### 1.4.6.5 Example

### Code and Data Setup

```
% cat id.pig
A = load 'passwd' using PigStorage(':');
B = foreach A generate $0 as id;
dump B;

% cat fake-passwd
ctdean:Chris Dean:secret
pauls:Paul Stolorz:good
carmas:Carlos Armas:evil
dra:Deirdre McClure:marvelous
```



```
% hadoop fs -put id.pig .
% hadoop fs -put fake-passwd passwd
```

## Curl Command

```
% curl -d user.name=ctdean \
-d file=id.pig \
-d arg=-v \
'http://www.myserver.com/templeton/v1/pig.json'
```

## JSON Output

```
{
  "id": "job_201111101627_0018",
  "info": {
    "stdout": "templeton-job-id: job_201111101627_0018",
    "stderr": "",
    "exitcode": 0
  }
}
```

### 1.4.7 POST hive

#### 1.4.7.1 Description

Runs a [Hive](#) query or set of commands.

#### 1.4.7.2 URL

<http://www.myserver.com/templeton/v1/hive.json>

#### 1.4.7.3 Parameters

Name	Description	Required?	Default
<b>execute</b>	String containing an entire, short hive program to run.	One of either "execute" or "file" is required	None
<b>file</b>	HDFS file name of a hive program to run.	One of either "exec" or "file" is required	None
<b>define</b>	Set a Hive configuration variable using the syntax <code>define=NAME=VALUE</code> .	Optional	None
<b>statusdir</b>	A directory where Templeton will write the	Optional	None

Name	Description	Required?	Default
	status of the Hive job. If provided, it is the caller's responsibility to remove this directory when done.		
<b>callback</b>	Define a URL to be called upon job completion. You may embed a specific job ID into this URL using <code>\$jobId</code> . This tag will be replaced in the callback URL with this job's job ID.	Optional	None

#### 1.4.7.4 Results

Name	Description
<b>id</b>	A string containing the job ID similar to "job_201110132141_0001".
<b>info</b>	A JSON object containing the information returned when the job was queued. See the Hadoop documentation ( <a href="#">Class TaskController</a> ) for more information.

#### 1.4.7.5 Example

##### Curl Command

```
% curl -d user.name=ctdean \
-d execute="select*+from+pokes;" \
-d statusdir="pokes.output" \
'http://www.myserver.com/templeton/v1/hive.json'
```

##### JSON Output

```
{
  "id": "job_2011111111311_0005",
  "info": {
    "stdout": "templeton-job-id: job_2011111111311_0005",
    "stderr": "",
    "exitcode": 0
  }
}
```

## Results

```
% hadoop fs -ls pokes.output
Found 2 items
-rw-r--r--  1 ctdean supergroup      610 2011-11-11 13:22 /user/ctdean/pokes.output/
stderr
-rw-r--r--  1 ctdean supergroup      15 2011-11-11 13:22 /user/ctdean/pokes.output/
stdout

% hadoop fs -cat pokes.output/stdout
1      a
2      bb
3      ccc
```

### 1.4.8 GET queue

#### 1.4.8.1 Description

Return a list of all job IDs registered to the user.

#### 1.4.8.2 URL

`http://www.myserver.com/templeton/v1/queue.json`

#### 1.4.8.3 Parameters

Only the [standard parameters](#) are accepted.

#### 1.4.8.4 Results

Name	Description
<b>ids</b>	A list of all job IDs registered to the user.

#### 1.4.8.5 Example

### Curl Command

```
% curl 'http://www.myserver.com/templeton/v1/queue.json?user.name=ctdean'
```

### JSON Output

```
{
  "job_2011111111311_0008",
  "job_2011111111311_0012"
}
```

### 1.4.9 GET queue/:jobid

#### 1.4.9.1 Description

Check the status of a job and get related job information given its job ID. Substitute ":jobid" with the job ID received when the job was created.

#### 1.4.9.2 URL

`http://www.myserver.com/templeton/v1/queue/:jobid.json`

#### 1.4.9.3 Parameters

Name	Description	Required?	Default
<b>:jobid</b>	The job ID to check. This is the ID received when the job was created.	Required	None

#### 1.4.9.4 Results

Name	Description
<b>status</b>	A JSON object containing the job status information. See the Hadoop documentation ( <a href="#">Class JobStatus</a> ) for more information.
<b>profile</b>	A JSON object containing the job profile information. See the Hadoop documentation ( <a href="#">Class JobProfile</a> ) for more information.
<b>id</b>	The job ID.
<b>parentId</b>	The parent job ID.
<b>percentComplete</b>	The job completion percentage, for example "75% complete".
<b>exitValue</b>	The job's exit value.
<b>user</b>	User name of the job creator.
<b>callback</b>	The callback URL, if any.
<b>completed</b>	A string representing completed status, for example "done".

#### 1.4.9.5 Example

#### Curl Command

```
% curl 'http://www.myserver.com/templeton/v1/queue/job_201112212038_0003.json?
user.name=ctdean'
```

## JSON Output

```
{
  "status": {
    "startTime": 1324529476131,
    "username": "ctdean",
    "jobID": {
      "jtIdentifier": "201112212038",
      "id": 4
    },
    "jobACLs": {
    },
    "schedulingInfo": "NA",
    "failureInfo": "NA",
    "jobId": "job_201112212038_0004",
    "jobPriority": "NORMAL",
    "runState": 2,
    "jobComplete": true
  },
  "profile": {
    "url": "http://localhost:50030/jobdetails.jsp?jobid=job_201112212038_0004",
    "jobID": {
      "jtIdentifier": "201112212038",
      "id": 4
    },
    "user": "ctdean",
    "queueName": "default",
    "jobFile": "hdfs://localhost:9000/tmp/hadoop-ctdean/mapred/staging/
ctdean/.staging/job_201112212038_0004/job.xml",
    "jobName": "PigLatin:DefaultJobName",
    "jobId": "job_201112212038_0004"
  },
  "id": "job_201112212038_0004",
  "parentId": "job_201112212038_0003",
  "percentComplete": "100% complete",
  "exitValue": 0,
  "user": "ctdean",
  "callback": null,
  "completed": "done"
}
```

### 1.4.10 DELETE queue/:jobid

#### 1.4.10.1 Description

Kill a job given its job ID. Substitute ":jobid" with the job ID received when the job was created.

#### 1.4.10.2 URL

`http://www.myserver.com/templeton/v1/queue/:jobid.json`

#### 1.4.10.3 Parameters

Name	Description	Required?	Default
<b>:jobid</b>	The job ID to delete. This is the ID received when the job was created.	Required	None

#### 1.4.10.4 Results

Name	Description
<b>status</b>	A JSON object containing the job status information. See the Hadoop documentation ( <a href="#">Class JobStatus</a> ) for more information.
<b>profile</b>	A JSON object containing the job profile information. See the Hadoop documentation ( <a href="#">Class JobProfile</a> ) for more information.
<b>id</b>	The job ID.
<b>parentId</b>	The parent job ID.
<b>percentComplete</b>	The job completion percentage, for example "75% complete".
<b>exitValue</b>	The job's exit value.
<b>user</b>	User name of the job creator.
<b>callback</b>	The callback URL, if any.
<b>completed</b>	A string representing completed status, for example "done".

#### 1.4.10.5 Example

##### Curl Command

```
% curl -X DELETE 'http://www.myserver.com/templeton/v1/queue/job_201111111311_0009.json?user.name=ctdean'
```

##### JSON Output

```

{
  "status": {
    "startTime": 1321047216471,
    "username": "ctdean",
    "jobID": {
      "jtIdentifier": "201111111311",
      "id": 9
    },
    "jobACLs": {
    },
    "schedulingInfo": "NA",
    "failureInfo": "NA",
    "jobId": "job_201111111311_0009",
    "jobPriority": "NORMAL",
    "runState": 1,
    "jobComplete": false
  },
  "profile": {
    "url": "http://localhost:50030/jobdetails.jsp?jobid=job_201111111311_0009",
    "user": "ctdean",
    "jobID": {
      "jtIdentifier": "201111111311",
      "id": 9
    },
    "queueName": "default",
    "jobFile": "hdfs://localhost:9000/tmp/hadoop-ctdean/mapred/staging/
ctdean/.staging/job_201111111311_0009/job.xml",
    "jobName": "streamjob3322518350676530377.jar",
    "jobId": "job_201111111311_0009"
  }
  "id": "job_201111111311_0009",
  "parentId": "job_201111111311_0008",
  "percentComplete": "10% complete",
  "exitValue": 0,
  "user": "ctdean",
  "callback": null,
  "completed": "false"
}

```

**Note:** The job is not immediately deleted, therefore the information returned may not reflect deletion, as in our example. Use [GET queue/:jobid](#) to monitor the job and confirm that it is eventually deleted.

## 1.5 PDF