Shell and Utility Commands

Table of contents

1. Shell Commands

1.1. fs

Invokes any FsShell command from within a Pig script or the Grunt shell.

1.1.1. Syntax

fs subcommand subcommand_parameters

1.1.2. Terms

subcommand	The FsShell command.
subcommand_parameters	The FsShell command parameters.

1.1.3. Usage

Use the fs command to invoke any FsShell command from within a Pig script or Grunt shell. The fs command greatly extends the set of supported file system commands and the capabilities supported for existing commands such as ls that will now support globing. For a complete list of FsShell commands, see File System Shell Guide

1.1.4. Examples

In these examples a directory is created, a file is copied, a file is listed.

```
fs -mkdir /tmp
fs -copyFromLocal file-x file-y
fs -ls file-y
```

1.2. sh

Invokes any sh shell command from within a Pig script or the Grunt shell.

1.2.1. Syntax

sh subcommand subcommand_parameters

1.2.2. Terms

subcommand	The sh shell command.
subcommand_parameters	The sh shell command parameters.

1.2.3. Usage

Use the sh command to invoke any sh shell command from within a Pig script or Grunt shell.

Note that only real programs can be run form the sh command. Commands such as cd are not programs but part of the shell environment and as such cannot be executed unless the user invokes the shell explicitly, like "bash cd".

1.2.4. Example

In this example the ls command is invoked.

```
grunt> sh ls
bigdata.conf
nightly.conf
.....
grunt>
```

2. Utility Commands

2.1. clear

Clear the screen of Pig grunt shell and position the cursor at top of the screen.

2.1.1. Syntax

clear

2.1.2. Terms

key	Description.
none	no parameters

2.1.3. Example

In this example the clear command clean up Pig grunt shell.

```
grunt> clear
```

2.2. exec

Run a Pig script.

2.2.1. Syntax

exec [–param param_name = param_value] [–param_file file_name] [script]

2.2.2. Terms

–param param_name = param_value	See Parameter Substitution.
–param_file file_name	See Parameter Substitution.
script	The name of a Pig script.

2.2.3. Usage

Use the exec command to run a Pig script with no interaction between the script and the Grunt shell (batch mode). Aliases defined in the script are not available to the shell; however, the files produced as the output of the script and stored on the system are visible after the script is run. Aliases defined via the shell are not available to the script.

With the exec command, store statements will not trigger execution; rather, the entire script is parsed before execution starts. Unlike the run command, exec does not change the command history or remembers the handles used inside the script. Exec without any parameters can be used in scripts to force execution up to the point in the script where the exec occurs.

For comparison, see the run command. Both the exec and run commands are useful for debugging because you can modify a Pig script in an editor and then rerun the script in the Grunt shell without leaving the shell. Also, both commands promote Pig script modularity as they allow you to reuse existing components.

2.2.4. Examples

In this example the script is displayed and run.

```
grunt> cat myscript.pig
```

```
a = LOAD 'student' AS (name, age, gpa);
b = LIMIT a 3;
DUMP b;

grunt> exec myscript.pig
(alice,20,2.47)
(luke,18,4.00)
(holly,24,3.27)
```

In this example parameter substitution is used with the exec command.

```
grunt> cat myscript.pig
a = LOAD 'student' AS (name, age, gpa);
b = ORDER a BY name;

STORE b into '$out';

grunt> exec –param out=myoutput myscript.pig
```

In this example multiple parameters are specified.

```
grunt> exec –param p1=myparam1 –param p2=myparam2 myscript.pig
```

2.3. help

Prints a list of Pig commands or properties.

2.3.1. Syntax

-help [properties]

2.3.2. Terms

properties	List Pig properties.

2.3.3. Usage

The help command prints a list of Pig commands or properties.

2.3.4. Example

Use "-help" to get a list of commands.

```
$ pig -help

Apache Pig version 0.8.0-dev (r987348)
compiled Aug 19 2010, 16:38:44
```

```
USAGE: Pig [options] [-] : Run interactively in grunt shell.
       Pig [options] -e[xecute] cmd [cmd ...] : Run cmd(s).
       Pig [options] [-f[ile]] file : Run cmds found in file.
  options include:
    -4, -log4jconf - Log4j configuration file, overrides log conf
    -b, -brief - Brief logging (no timestamps)
    -c, -check - Syntax check
etc …
```

Use "-help properties" to get a list of properties.

```
$ pig -help properties

The following properties are supported:
    Logging:
        verbose=true|false; default is false. This property is the same as
-v switch
        brief=true|false; default is false. This property is the same as -b
switch
        debug=OFF|ERROR|WARN|INFO|DEBUG; default is INFO. This property is
the same as -d switch
        aggregate.warning=true|false; default is true. If true, prints
count of warnings
            of each type rather than logging each warning.
etc …
```

2.4. history

Display the list of statements used so far.

2.4.1. Syntax

history [-n]

2.4.2. Terms

key	Description.
-n	Omit line numbers in the list.

2.4.3. Usage

The history command shows the statements used so far.

2.4.4. Example

In this example the history command shows all the statements with line numbers and without them.

```
grunt> a = LOAD 'student' AS (name, age, gpa);
grunt> b = order a by name;
grunt> history
1 a = LOAD 'student' AS (name, age, gpa);
2 b = order a by name;

grunt> c = order a by name;
grunt> history -n
a = LOAD 'student' AS (name, age, gpa);
b = order a by name;
c = order a by name;
```

2.5. kill

Kills a job.

2.5.1. Syntax

kill jobid

2.5.2. Terms

jobid	The job id.

2.5.3. Usage

Use the kill command to kill a Pig job based on the job id.

The kill command will attempt to kill any MapReduce jobs associated with the Pig job. Under certain conditions, however, this may fail; for example, when a Pig job is killed and does not have a chance to call its shutdown procedures.

2.5.4. Example

In this example the job with id job_0001 is killed.

```
grunt> kill job_0001
```

2.6. quit

Quits from the Pig grunt shell.

2.6.1. Syntax

exit

2.6.2. Terms

none	no parameters

2.6.3. Usage

The quit command enables you to quit or exit the Pig grunt shell.

2.6.4. Example

In this example the quit command exits the Pig grunt shall.

```
grunt> quit
```

2.7. run

Run a Pig script.

2.7.1. Syntax

run [–param param_name = param_value] [–param_file file_name] script

2.7.2. Terms

–param param_name = param_value	See Parameter Substitution.
–param_file file_name	See Parameter Substitution.
script	The name of a Pig script.

2.7.3. Usage

Use the run command to run a Pig script that can interact with the Grunt shell (interactive mode). The script has access to aliases defined externally via the Grunt shell. The Grunt shell has access to aliases defined within the script. All commands from the script are visible in the

command history.

With the run command, every store triggers execution. The statements from the script are put into the command history and all the aliases defined in the script can be referenced in subsequent statements after the run command has completed. Issuing a run command on the grunt command line has basically the same effect as typing the statements manually.

For comparison, see the exec command. Both the run and exec commands are useful for debugging because you can modify a Pig script in an editor and then rerun the script in the Grunt shell without leaving the shell. Also, both commands promote Pig script modularity as they allow you to reuse existing components.

2.7.4. Example

In this example the script interacts with the results of commands issued via the Grunt shell.

```
grunt> cat myscript.pig
b = ORDER a BY name;
c = LIMIT b 10;

grunt> a = LOAD 'student' AS (name, age, gpa);

grunt> run myscript.pig

grunt> d = LIMIT c 3;

grunt> DUMP d;
(alice,20,2.47)
(alice,27,1.95)
(alice,36,2.27)
```

In this example parameter substitution is used with the run command.

```
grunt> a = LOAD 'student' AS (name, age, gpa);

grunt> cat myscript.pig
b = ORDER a BY name;
STORE b into '$out';

grunt> run –param out=myoutput myscript.pig
```

2.8. set

Shows/Assigns values to keys used in Pig.

2.8.1. Syntax

```
set [key 'value']
```

2.8.2. Terms

key	Key (see table). Case sensitive.
value	Value for key (see table). Case sensitive.

2.8.3. Usage

Use the set command to assign values to keys, as shown in the table. All keys and their corresponding values (for Pig and Hadoop) are case sensitive. If set command is used without key/value pair argument, Pig prints all the configurations and system properties.

Key	Value	Description
default_parallel	a whole number	Sets the number of reducers for all MapReduce jobs generated by Pig (see Use the Parallel Features).
debug	on/off	Turns debug-level logging on or off.
job.name	Single-quoted string that contains the job name.	Sets user-specified name for the job
job.priority	Acceptable values (case insensitive): very_low, low, normal, high, very_high	Sets the priority of a Pig job.
stream.skippath	String that contains the path.	For streaming, sets the path from which not to ship data (see DEFINE (UDFs, streaming) and About Auto-Ship).

All Pig and Hadoop properties can be set, either in the Pig script or via the Grunt command line.

2.8.4. Examples

In this example key value pairs are set at the command line.

```
grunt> SET debug 'on'
grunt> SET job.name 'my job'
grunt> SET default_parallel 100
```

In this example default_parallel is set in the Pig script; all MapReduce jobs that get launched will use 20 reducers.

```
SET default_parallel 20;
A = LOAD 'myfile.txt' USING PigStorage() AS (t, u, v);
B = GROUP A BY t;
C = FOREACH B GENERATE group, COUNT(A.t) as mycount;
D = ORDER C BY mycount;
STORE D INTO 'mysortedcount' USING PigStorage();
```

In this example multiple key value pairs are set in the Pig script. These key value pairs are put in job-conf by Pig (making the pairs available to Pig and Hadoop). This is a script-wide setting; if a key value is defined multiple times in the script the last value will take effect and will be set for all jobs generated by the script.

```
...
SET mapred.map.tasks.speculative.execution false;
SET pig.logfile mylogfile.log;
SET my.arbitrary.key my.arbitary.value;
...
```