Holland Documentation

Release

Holland Core Team

February 02, 2017

Contents

CHAPTER 1

Table of Contents

1.1 Introduction to Holland

Holland is an Open Source backup framework originally developed by Rackspace and written in Python. The original intent was to offer more reliability and flexilibity when backing up MySQL databases, though the current version is now able to backup MySQL and PostgreSQL databases. Because Holland is plugin-based framework, it can conceivably backup most anything you want by whatever means you want.

1.1.1 Dependencies

The core Holland framework has the following dependencies (available on any remotely modern Linux distribution):

- Python >= 2.3
- pkg_resources
- python-setuptools

MySQL based plugins additional require the MySQLdb python connector:

- MySQLdb

For Red-Hat Enterprise Linux 5, all dependencies are available directly from the base channels. For Red-Hat Enterprise Linux 4, EPEL is required for python-setuptools.

Note that other plugins may have additional dependency requirements.

1.1.2 Installation

Holland has ready-made packages available for Red-Hat, CentOS, and Ubuntu which are available via the OpenSUSE build system. Other distributions may download the generic tarball or pull directly from github.

1.2 Holland Command-Line Reference

Here are the commands available from the 'holland' command-line tool:

1.2.1 help (h)

Usage: `holland help <command>`

Provides basic information about the provided command. If no command is provided, it displays global help instead.

1.2.2 backup (bk)

Usage: `holland backup [backup-set1, backup-set2, ..., backup-setN]`

Runs the backup operation. If no backup-sets are specified, all active backup-sets (those defined in the 'backupsets' variable in holland.conf) are backed up.

One or more backup-sets can be specified directly, in which case only those backup-sets are backed up.

Additional Command Line Arguments:

`--dry-run` (`-n`): Can be used here to simulate, but not actually run, a backup. This should be used when troubleshooting a particular error before trying to run a real backup.

`--no-lock` (`-f`): Normally, only one instance of Holland can run at any given time using lock-files. Using this flag causes the lock-files to be ignored. This has some very clear use-cases but otherwise be mindful of using this setting as it can cause backups to fail in some cases.

`--abort-immediately`: abort on the first backup-set that fails (assuming multiple backupsets were specified)

Examples:

`# holland bk --dry-run weekly`: Attempts a dry-run of the weekly backup-set.

`# holland bk --no-lock --abort-immediately`: Attempts a backup of all the default backup-sets ignoring locks and aborting immediately if one of the backup-sets fails.

1.2.3 list-backups (lb)

Usage: `holland list-backups`

Provides extended information about available backups.

1.2.4 list-plugins (lp)

Usage: `holland list-plugins`

Lists all the available (installed) plugins available to Holland.

1.2.5 mk-config (mc)

Usage: `holland mk-config <provider>`

Generates a template backup-set for a particular provider (such as mysqldump). By default, the output is sent to standard out but can be copied to a file, either by using the `--file`, `--edit`, or `-name` options (see below).

Additional Command Line Arguments:

`--edit`: Load the file into the system text-editor for further modifications.

`--file=FILE` (`-f`): Write the output directly to provided file.

`--name=NAME`: Creates a backup-set usable in Holland, which basically means that a file is created of the provided name under the backup-set directory.

`--provider`: Indicates that the default provider configuration should be outputted instead. This is really only used when creating a provider config specifically - it should not be used for backup-sets.

Examples:

`# holland mk-config mysql-lvm > mysql-lvm.conf`: Output the default configuration for MySQL-LVM backups and write the contents out to mysql-lvm.conf in the current working directory.

`# holland mc mysqldump --name=Bob --edit`: Create a backup-set using the mysqldump provider named Bob and allow interactive editing of the backup-set before saving the file.

1.2.6 purge (pg)

Usage: `holland purge <backup-set>/<backup-id>`

Purges old backups by specifying the backup-set name and set-id.

For example: `# holland purge mybackups/20090502_155438`: Purge one of the backups taken on May 2nd, 2009 from the mybackups backup-set.

1.3 Usage and Implementation Overview

Because Holland is very pluggable, it may first seem a bit confusing when it comes to configuring Holland to do something useful. Out of the box, Holland is designed to backup MySQL databases using the mysqldump provider. This is the simplest setup, and may be sufficient for most people. However, others may wish to have more fine-grained control over their backups and/or use another method other than mysqldump.

For instance, one can configure a backup set to backup certain databases using mysqldump, others using the mysql-lvm plugins etc. All this is done by a mix of plugins (sometimes called providers) and backup-sets.

1.3.1 Backup-Sets

Each backup-set implements a backup plugin (provider) and often some helper plugins for things such as compression. Plugins come with a set of defaults such that only values that need to be overridden need to be specified, although it is perfectly acceptable to specify options that are already default - one would merely be stating the obvious. Doing so would also make sure future changes to the defaults in Holland do not impact existing backup-sets.

1.3.2 Provider Plugins

Provider plugins provide a backup service for use in a backup set. They are the interface between Holland and the method of backing up data. As of Holland 1.0.8, there are 5 providers included with Holland:

- mysqldump

  Uses the `mysqldump` utility to backup MySQL databases.

- MySQL + LVM

  Backup MySQL databases using LVM snapshots which allows for near lockless or fully lockless (when transactional engines are used) backups. MySQL must be running on an LVM volume with sufficient free extents to store a working snapshot. It is also extremely ill-advised to store the backup on the same volume as MySQL.

- Support for Percona XtraBackup

  New in version 1.0.8.

  Backup MySQL databases using Percona XtraBackup. This provides a near lockless backup when using the InnoDB storage engine while also providing a mysqlhotcopy style backup for MyISAM tables.

- pgdump

  Backup PostgreSQL databases using the `pgdump` utility.

- Example

  This is used solely as a template for designing providers. It otherwise does nothing.

As Holland is a framework, it can actually backup most anything as long as there is a provider plugin for it. This includes things that have nothing to do with databases. The idea is to present an easy to use and clear method of backing up and restoring backups no matter the source.

1.4 Configuring Holland

By default, Holland's configuration files reside in /etc/holland. The main configuration file is holland.conf, however there are a number of other configuration files for configuring default settings for providers and for configuring backup sets.

Each configuration file has one ore more sections, defined by square brackets Underneath each section, one or more configuration option can be specified. These options are in a standard "option = value" format. Comments are prefixed by the # sign.

Note that many settings have default values and, as a result, can either be commented out or omitted entirely.

1.4.1 Global Config

The main configuration file (usually `/etc/holland/holland.conf`) defines both global settings as well as the active backup sets. It is divided into two sections *[holland]* and *[logging]*.

[holland]

plugin_dirs = [directory1],[directory2],...,[directoryN]
Defines where the plugins can be found. This can be a comma-separated list but usually does not need to be modified. For most installations, this will usually be `/usr/share/holland/plugins`.

Deprecated since version 1.0.8: This option is no longer required and can be omitted.

backup_directory = [directory]
Top-level directory where backups are held. This is usually `/var/spool/holland`.

backupsets = [backupset1],[backupset2],...,[backupsetN]
A comma-separated list of all the backup sets Holland should backup. Each backup set is defined in `/etc/holland/backupsets/<name>.conf` by default.

umask = [0000-7777]
Sets the umask of the resulting backup files.

path = <directory1>:<directory2>:...:<directoryN>
Defines a path for holland and its spawned processes.

[logging]

filename = [path]/[filename]
The log file itself.

level = [debug|info|warning|error|critical]
Sets the verbosity of Holland's logging process. Available options are `debug`, `info`, `warning`, `error`, and `critical`

Example

```
## Root holland config file
[holland]

## Paths where holland plugins may be found.
## Can be comma separated
plugin_dirs = /usr/share/holland/plugins

## Top level directory where backups are held
backup_directory = /var/spool/holland

## List of enabled backup sets. Can be comma separated.
## Read from <config_dir>/backupsets/<name>.conf
# backupsets = example, traditional, parallel_backups, non_transactional
backupsets = mydbbackup, pgdump-full, mysql-lvm-reportingdb

# Define a umask for file generated by holland
umask = 0007

# Define a path for holland and its spawned processes
path = /usr/local/bin:/usr/local/sbin:/bin:/sbin:/usr/bin:/usr/sbin

[logging]
## where to write the log
filename = /var/log/holland.log

## debug, info, warning, error, critical (case insensitive)
level = info
```

1.4.2 Backup-Set Configs

Backup-Set configuration files are housed in `/etc/holland/backupsets` with the name of the backup-set being the name of the file before the .conf suffix (e.g. 'myfavoritebackup.conf' is the configuration file for the 'myfavorite-backup' backup-set).

The backups themselves are placed under the directory defined in the *backup_directory* section of the main configuration file. Each backup resides under a directory corresponding to the backup-set name followed by one or more date-encoded directories.

Backup-Set configuration files inherit the configuration options of the specified plugins (though these settings can be overridden). To define a provider plugin for the backup set, you must put the following at the top of the backup set configuration file.

```
[holland:backup]
plugin = <plugin>
```

```
backups-to-keep = #
estimated-size-factor = #
```

[holland:backup] Configuration Options

This section is for configuring Holland specific options and is usually at the top of a particular backup set configuration file. These options control the behavior of Holland itself, rather than of the backup provider and associated helper plugins (which are defined within their own sections - see below).

plugin = [provider plugin]
: This is the name of the provider that will be used for the backup-set. This is required in order for the backup-set to function.

backups-to-keep = #
: Specifies the number of backups to keep for a backup-set. Defaults to retaining 1 backup.

estimated-size-factor = #
: Specifies the scale factor when Holland decides if there is enough free space to perform a backup. The default is 1.0 and this number is multiplied against what each individual plugin reports its estimated backup size when Holland is verifying sufficient free space for the backupset.

auto-purge-failures = [yes|no]
: Specifies whether to keep a failed backup or to automatically remove the backup directory. By default this is on with the intention that whatever process is calling holland will retry when a backup fails. This behavior can be disabled by setting auto-purge-failures = no when partial backups might be useful or when troubleshooting a backup failure.

purge-on-demand = [yes|no]
: If enabled, this option will cause holland to attempt to purge old backups within the same backupset to free enough space to allow a new backup to start rather than failing when it appears that there is insufficient space to run a new backup. If the space consumed by all purgable backups is less than the estimated space for a new backup, no backups are purged as the new backup will fail regardless.

purge-policy = [manual|before-backup|after-backup]
: Specifies when to run the purge routine on a backupset. By default this is run after a new successful backup completes. Up to `backups-to-keep` backups will be retained including the most recent.

  `purge-policy = before-backup` will run the purge routine just before a new backup starts. This will retain up to `backups-to-keep` backups before the new backup is even started allowing purging all previous backups if `backups-to-keep` is set to 0. This behavior is useful if some other process is retaining backups off-server and disk space is at a premium.

  `purge-policy = after-backup` will run the purge routine after the completion of the backup. This means more space is required during the backup run. This is safer than `before-backup` due to failed backups not causing good backups to be purged.

  `purge-policy = manual` will never run the purge routine automatically. Either `holland purge` must be run externally or an explicit removal of desired backup directories can be done at some later time.

Hooks

before-backup-command = string
: Run a shell command before a backup starts, such as setting up an iptables rule (taking a mysql slave out of a load balancer) or aborting the backup based on some external condition.

  The backup will fail if this command exits with a non-zero status.

New in version 1.0.7.

after-backup-command = string
Run a shell command after a **successful** backup, such as sending out an e-mail or performing external post backup cleanup tasks (such putting a server back in a load balancer for example).

Note if the backup fails, `failed-backup-command` will be run instead. Thus, if doing things like pulling the server out of a load-balancer, such commands should added to `after-backup-command` and `failed-backup-command` to avoid an inconsistent state.

The backup will fail if this command exits with a non-zero status.

New in version 1.0.7.

failed-backup-command = string
Run a shell command if a backup failed, such as firing off an e-mail to notify relevant folks about the failure.

New in version 1.0.7.

For all hook commands, Holland will perform simple text substitution on the three parameters:

- **hook**: The name of the hook being called (one of: `before-backup-command`, `after-backup-command`, `failed-backup-command`)
- **backupdir**: The path to the current backup directory (e.g. `/var/spool/holland/mysqldump/YYYYmmdd_HHMMSS`)
- **backupset**: The name of the backupset being run (e.g. `mysql-lvm`)

For Example

```
[holland:backup]
plugin = mysqldump
before-backup-command = /usr/local/bin/my-custom-script --hook ${hook} --backupset ${backupset} --bac
after-backup-command = echo ${backupset} completed successfully.  Files are in ${backupdir}
failed-backup-command = echo "${backupset} failed!" | mail -s "${backupset} backup failed" sysadmins(
```

Provider Plugin Configs

The following are the provider plugins that can be used in a backup-set. These are used within their own braced section in the backup-set configuration file. For specific information on how to configure a desired provider, see the list below.

For advanced users, the defaults for each provider plugin can be changed by editing the default configuration file for said provider. These files are located in `/etc/holland/providers` by default.

MySQL Plugins

mysqldump Provider Configuration [mysqldump] Backs up one or more MySQL databases using the mysqldump tool.

[mysqldump] mysql-binpath = /path/to/mysql/bin

Defines the location of the MySQL binary utilities. If not provided, Holland will use whatever is in the path.

lock-method = flush-lock | lock-tables | single-transaction | auto-detect | none

Defines which lock method to use. By default, auto-detect will be used.

- flush-lock

flush-lock will place a global lock on all tables involved in the backup regardless of whether or not they are in the backup-set. If file-per-database is enabled, then flush-lock will lock all tables for every database being backed up. In other words, this option may not make much sense when using file-per-database.

- lock-tables

  lock-tables will lock all tables involved in the backup. If file-per-database is enabled, then lock-tables will only lock all the tables associated with that database.

- single-transaction

  Forces the use of `--single-transaction` which enabled semi-transparent backups of transactional tables. Forcing this can cause inconsistencies with non-transactional tables, however. While non-transactional tables will still lock, they will only lock when they are actually being backed up. **Use this setting with extreme caution when backing non-transactional tables.**

- auto-detect

  Let Holland decide which option to use by checking to see if a database or backup-set only contains transactional tables. If so, `--single-transaction` will be used. Otherwise, `--lock-tables` will be used.

- none

  Does absolutely no explicit locking when backing up the databases or backup-set. This should only be used when backing up a slave and only after the slave has been turned off (ie, this can be used with the **stop-slave** option).

exclude-invalid-views = yes | no (default: no)

Whether to automate exclusion of invalid views that would otherwise cause mysqldump to fail. This adds additional overhead so this option is not enabled by default.

When enabled, thos option will scan the INFORMATION_SCHEMA.VIEWS table and execute SHOW FIELDS against each view. If a view is detects as invalid, an ignore-table option will be added to exclude the table. Additionally, the plugin will attempt to save the view definion to 'invalid_views.sql' in the backupset's backup directory.

New in version 1.0.8.

dump-routines = yes | no (default: yes)

Whether or not to backup routines in the backup set directly. Routines are stored in the 'mysql' database, but it can sometimes be convenient to include them in a backup-set directly.

Changed in version 1.0.8: This option now enabled by default.

dump-events = yes | no

Whether or not to dump events explicitly. Like routines, events are stored in the 'mysql' database. Nonetheless, it can sometimes be convenient to include them in the backup-set directly.

Note: This feature requires MySQL 5.1 or later. The mysqldump plugin will automatically disable events if the version of mysqldump is too old.

Changed in version 1.0.8: This option is now enabled by default

stop-slave = yes | no

Stops the SQL_THREAD during the backup. This means that writes from the master will continue to spool but will not be replayed. This helps avoid lock wait timeouts among things while still allowing data to be spooled from the master.

Note that previous versions of Holland prior to 1.0.6 simply ran a STOP SLAVE instead, which suspends both replication threads.

Holland will log some fairly useful information to the 'backup.conf' file in regards to log files and positions:

```
[mysql:replication]
slave_master_log_pos = 147655593
slave_master_log_file = db2-bin.007120
master_log_file = mysqld-bin.000001
master_log_pos = 313
```

The `slave_master_*` values refer to the file and position that the slave has replicated to from its master server (often useful when cloning slaves, for instance).

The `master_log_*` values refer to the binary log and position **of the slave** itself and is only logged is binary logging on the slave is enabled. This information can be used to setup chained replication as well as point in time recovery of that slave (which would be useful if one is only doing backups on the slave and not the master).

bin-log-position = yes | no

Record the binary log name and position at the time of the backup as a SQL comment in backup SQL file itself. This directly correlates to the `--master-data=2` 'mysqldump' command-line option.

flush-logs = yes | no

Whether or not to run FLUSH LOGS in MySQL with the backup. When FLUSH LOGS is actually executed depends on which if database filtering is being used and whether or not file-per-database is enabled. Generally speaking, it does not make sense to use flush-logs with file-per-database since the binary logs will not be consistent with the backup.

file-per-database = yes | no

Whether or not to split up each database into its own file. Note that it can be more consistent an efficient to backup all databases into one file, however this means that restore a single database can be difficult if multiple databases are defined in the backup set.

When backing up all databases within a single file, the backup file will be named `all_databases.sql`. If compression is used, the compression extension will be appended to the filename (e.g. `all_databases.sql.gz`).

additional-options = <mysqldump argument>[, <mysqldump argument>]

Can optionally specify additional options directly to `mysqldump` if there is no native Holland option available. This option accepts a comma delimited list of arguments to pass on the commandline.

extra-defaults = yes | no (default: no)

This option controls whether mysqldump will only read options as set by holland or if additional options from global config files are read. By default, the plugin only uses optons as set in the backupset config and includes authentication credentials only from the [client] section in ~/.my.cnf.

estimate-method = plugin | const:<size> (default: plugin)

This option will skip some of the heavyweight queries necessary to calculate the size of tables to be backed up. If a constant size is specified, then only table names are evaluated and only if table filtering is being used. Additionally, engines will be looked up via SHOW CREATE TABLE if lock-method = auto-detect, in order for the plugin to determine if tables are using a transactional storage engine. With 'plugin', the default behavior of reading both size information and table names from the information schema is used, which may be slow particularly for a large number of tables.

Database and Table filtering

Database and Table filtering **databases** = <glob>

exclude-databases = <glob>

tables = <glob>

exclude-tables = <glob>

The above options accepts GLOBs in comma-separated lists. Multiple filtering options can be specified. When filtering on tables, be sure to include both the database and table name.

Be careful with quotes. Normally these are not needed, but when quotes are necessary, be sure to only quote each filtering statement, as opposed to putting quotes around all statements.

Below are a few examples of how these can be applied:

Default (backup everything):

```
databases = *
tables = *
```

Using database inclusion and exclusions:

```
databases = drupal*, smf_forum,
exclude-databases = drupal5
```

Including Tables:

```
tables = phpBB.sucks, drupal6.node*, smf_forum.*
```

Excluding Tables:

```
exclude-tables = mydb.uselesstable1, x_cart.*, *.sessions
```

[compression] Specify various compression settings, such as compression utility, compression level, etc.

method = gzip | pigz | bzip | lzop | lzma | gpg

> Define which compression method to use. Note that some methods may not be available by default on every system and may need to be compiled or installed and may not work with all the compression options.

inline = yes | no

> Whether or not to pipe the output of mysqldump into the compression utility. Enabling this is recommended since it usually only marginally impacts performance, particularly when using a lower compression level.

level = 0-9

> Specify the compression ratio. The lower the number, the lower the compression ratio, but the faster the backup will take. Generally, setting the lever to 1 or 2 results in favorable compression of textual data and is noticeably faster than the higher levels. Setting the level to 0 effectively disables compression.

bin-path = <full path to utility>

> This only needs to be defined if the compression utility is not in the usual places or not in the system path.

options = <string of otpions>

> Pass additional options not included in the above directly to the compression utility (e.g. `--compress-algo=bzip2` if using 'gpg').

MySQL connection info [mysql:client] These are optional and, if left undefined, Holland will try to login using the standard .my.cnf conventions.

user = <user>

The user to connect to MySQL as.

password = <password>

The password for the MySQL user

socket = <socket>

The socket file to connect to MySQL with.

host = <host>

This would be used for connecting to MySQL remotely.

port = <port>

Used if MySQL is running on a port other than 3306.

MySQL LVM Provider Configuration [mysql-lvm] Creates an LVM snapshot of a running MySQL instance and performs a binary-based backup with minimal locking. MySQL must be running on an LVM volume with reserved space for snapshots. It is highly recommended that this volume be separate from the one storing the resulting backups.

[mysql-lvm] snapshot-size = <size-in-MB>

The size of the snapshot itself. By default it is 20% of the size of the MySQL LVM mount or the remaining free-space in the Volume Group (if there is less than 20% available) up to 15GB. If snapshot-size is defined, the number represents the size of the snapshot in megabytes.

snapshot-name = <name>

The name of the snapshot, the default being the name of the MySQL LVM volume + "_snapshot" (ie Storage-MySQL_snapshot)

snapshot-mountpoint = <path>

Where to mount the snapshot. By default a randomly generated directory under /tmp is used.

innodb-recovery = yes | no (default: no)

Whether or not to run an InnoDB recovery operation. This avoids needing to do so during a restore, though will make the backup process itself take longer.

force-innodb-backup = yes | no (default: no)

Whether to attempt a backup even if the mysql-lvm plugin thinks it cannot obtain a good backup. This can occur when innodb data files are outside of the mysql datadir or exist on entirely separate logical volumes.

lock-tables = yes | no (default: yes)

Whether or not to run a FLUSH TABLES WITH READ LOCK to grab various bits of information (such as the binary log name and position). Disabling this requires that binary logging is disabled and InnoDB is being used exclusively. Otherwise, it is possible that the backup could contain crashed tables.

extra-flush-tables = yes | no (default: yes)

Whether or not to run a FLUSH TABLES before running the full FLUSH TABLES WITH READ LOCK. Should make the FLUSH TABLES WITH READ LOCK operation a bit faster.

[tar] **exclude** = pattern[, pattern...]

Patterns to exclude from archive. These should be relative paths and are almost always relative to the mysql data directory. For instance to exclude binary logs in the data directory from the backup you might specify: exclude = ./bin-log.*, mysql.sock

pre-args = <string>

Additional arguments to append to the tar commandline before the backup path is specified. This should be the full string as you might specify on the commandline. Shell globbing is not supported.

For instance you might add the /etc/my.cnf to the tar archive via: pre-args = -C /etc ./my.cnf

post-args = <string>

Additional arguments to append to the tar commandline after the backup path is specified. This should be a string exactly as you might specify on the commandline. Shell globbing is not evaluated.

[compression] Specify various compression settings, such as compression utility, compression level, etc.

method = gzip | pigz | bzip | lzop | lzma | gpg

> Define which compression method to use. Note that some methods may not be available by default on every system and may need to be compiled or installed and may not work with all the compression options.

inline = yes | no

> Whether or not to pipe the output of mysqldump into the compression utility. Enabling this is recommended since it usually only marginally impacts performance, particularly when using a lower compression level.

level = 0-9

> Specify the compression ratio. The lower the number, the lower the compression ratio, but the faster the backup will take. Generally, setting the lever to 1 or 2 results in favorable compression of textual data and is noticeably faster than the higher levels. Setting the level to 0 effectively disables compression.

bin-path = <full path to utility>

> This only needs to be defined if the compression utility is not in the usual places or not in the system path.

options = <string of otpions>

> Pass additional options not included in the above directly to the compression utility (e.g. `--compress-algo=bzip2` if using 'gpg').

MySQL connection info [mysql:client] These are optional and, if left undefined, Holland will try to login using the standard .my.cnf conventions.

user = <user>

The user to connect to MySQL as.

password = <password>

The password for the MySQL user

socket = <socket>

The socket file to connect to MySQL with.

host = <host>

This would be used for connecting to MySQL remotely.

port = <port>

Used if MySQL is running on a port other than 3306.

mysqldump LVM Provider Configuration [mysqldump-lvm] Backs up one or more MySQL databases by creating an LVM snapshot and then starting a instance of MySQL on top of it to then perform a mysqldump. This effectively produces a non-blocking logical backup.

[mysql-lvm] snapshot-size = <size-in-MB>

The size of the snapshot itself. By default it is 20% of the size of the MySQL LVM mount or the remaining free-space in the Volume Group (if there is less than 20% available) up to 15GB. If snapshot-size is defined, the number represents the size of the snapshot in megabytes.

snapshot-name = <name>

The name of the snapshot, the default being the name of the MySQL LVM volume + "_snapshot" (ie Storage-MySQL_snapshot)

snapshot-mountpoint = <path>

Where to mount the snapshot. By default a randomly generated directory under /tmp is used.

innodb-recovery = yes | no (default: no)

Whether or not to run an InnoDB recovery operation. This avoids needing to do so during a restore, though will make the backup process itself take longer.

lock-tables = yes | no (default: yes)

Whether or not to run a FLUSH TABLES WITH READ LOCK to grab various bits of information (such as the binary log name and position). Disabling this requires that binary logging is disabled and InnoDB is being used exclusively. Otherwise, it is possible that the backup could contain crashed tables.

extra-flush-tables = yes | no (default: yes)

Whether or not to run a FLUSH TABLES before running the full FLUSH TABLES WITH READ LOCK. Should make the FLUSH TABLES WITH READ LOCK operation a bit faster.

[mysqld] mysqld-exe = <path>[, <path>...] (default: mysqld in PATH, /usr/libexec/mysqld)

This provides a list of locations where the mysqld process to use might be found. This is searched in order of entries in this list.

user = <name>

The –user parameter to use with mysqld.

innodb-buffer-pool-size = <size> (default: 128M)

How large to size the innodb-buffer-pool-size.

tmpdir = <path> (default: system tempdir)

Path to the –tmpdir that mysqld should use.

[mysqldump] mysqldump-lvm supports almost all of the options from the mysqldump plugin. –master-data is not supported, as the mysqld process will not read binary logs, so this plugin will automatically disable bin-log-position, if set.

Binary log information from SHOW MASTER STATUS and SHOW SLAVE STATUS is recorded in the ${backup_directory}/backup.conf file under the [mysql:replication] section.

[compression] Specify various compression settings, such as compression utility, compression level, etc.

method = gzip | pigz | bzip | lzop | lzma | gpg

Define which compression method to use. Note that some methods may not be available by default on every system and may need to be compiled or installed and may not work with all the compression options.

inline = yes | no

Whether or not to pipe the output of mysqldump into the compression utility. Enabling this is recommended since it usually only marginally impacts performance, particularly when using a lower compression level.

level = 0-9

Specify the compression ratio. The lower the number, the lower the compression ratio, but the faster the backup will take. Generally, setting the lever to 1 or 2 results in favorable compression of textual data and is noticeably faster than the higher levels. Setting the level to 0 effectively disables compression.

bin-path = <full path to utility>

This only needs to be defined if the compression utility is not in the usual places or not in the system path.

options = <string of otpions>

Pass additional options not included in the above directly to the compression utility (e.g. `--compress-algo=bzip2` if using 'gpg').

MySQL connection info [mysql:client] These are optional and, if left undefined, Holland will try to login using the standard .my.cnf conventions.

user = <user>

The user to connect to MySQL as.

password = <password>

The password for the MySQL user

socket = <socket>

The socket file to connect to MySQL with.

host = <host>

This would be used for connecting to MySQL remotely.

port = <port>

Used if MySQL is running on a port other than 3306.

Holland Plugin for Percona XtraBackup Configuration [xtrabackup] Backup a MySQL instance using Percona XtraBackup.

Note: Percona XtraBackup is a trademark of Percona LLC The Holland Project does not intend the use or display of Percona's trademark to imply a relationship with, or endorsement or sponsorship of the Holland Project by Percona.

[xtrabackup] global-defaults = <path> (default: /etc/my.cnf)

The MySQL configuration file for xtrabackup to parse. This is !include'd into the my.cnf the xtrabackup plugin generates

innobackupex = <name> (default: innobackupex)

The path to the innobackupex script to run. If this is a relative path this will be found in holland's environment PATH as configured in /etc/holland/holland.conf.

ibbackup = <name>

The path to the ibbackup command to use. By default, no –ibbackup option is pass to the innobackupex script. Usually innobackupex will detect this by itself and this should not need to be set.

stream = tar|xbstream|yes|no (default: tar)

Whether to generate a streaming backup.

Changed in version 1.0.8: 'tar' and 'xbstream' are now valid options. The old stream = yes is now equivalent to stream = tar and stream = no disables streaming entirely and will result in a normal directory copy with xtrabackup

apply-logs = yes | no (default: yes)

Whether to run `innobackupex --apply-logs` at the end of the backup. This is only supported when performing a non-streaming, non-compressed backup. In this case, even if apply-logs = yes (the default), the prepare stage will be skipped. Even with an uncompressed, non-streaming backup you may want to disable apply-logs if you wish to use incremental backups.

New in version 1.0.8.

slave-info = yes | no (default: yes)

Whether to enable the –slave-info innobackupex option

safe-slave-backup = yes | no (default: yes)

Whether to enable the –safe-slave-backup innobackupex option.

no-lock = yes | no (default: no)

Whether to enable the –no-lock innobackupex option

tmpdir = <path> (default: `${backup_directory}`)

The path for the innobackupex –tmpdir option. By default this will use the current holland backup directory to workaround the following bug: https://bugs.launchpad.net/percona-xtrabackup/+bug/1007446

New in version 1.0.8.

additional-options = <option>[, <option>...]

A list of additional options to pass to innobackupex. This is a comma separated list of options.

pre-command = <command-string>

A command to run prior to running this xtrabackup run. This can be used, for instance, to generate a mysqldump schema dump prior to running xtrabackup. instances of ${backup_directory} will be replaced with the current holland backup directory where the xtrabackup data will be stored.

[compression] Specify various compression settings, such as compression utility, compression level, etc.

method = gzip | pigz | bzip | lzop | lzma | gpg

Define which compression method to use. Note that some methods may not be available by default on every system and may need to be compiled or installed and may not work with all the compression options.

inline = yes | no

Whether or not to pipe the output of mysqldump into the compression utility. Enabling this is recommended since it usually only marginally impacts performance, particularly when using a lower compression level.

level = 0-9

Specify the compression ratio. The lower the number, the lower the compression ratio, but the faster the backup will take. Generally, setting the lever to 1 or 2 results in favorable compression of textual data and is noticeably faster than the higher levels. Setting the level to 0 effectively disables compression.

bin-path = <full path to utility>

This only needs to be defined if the compression utility is not in the usual places or not in the system path.

options = <string of otpions>

Pass additional options not included in the above directly to the compression utility (e.g. `--compress-algo=bzip2` if using 'gpg').

MySQL connection info [mysql:client] These are optional and, if left undefined, Holland will try to login using the standard .my.cnf conventions.

user = <user>

The user to connect to MySQL as.

password = <password>

The password for the MySQL user

socket = <socket>

The socket file to connect to MySQL with.

host = <host>

This would be used for connecting to MySQL remotely.

port = <port>

Used if MySQL is running on a port other than 3306.

MySQL connection info [mysql:client] These are optional and, if left undefined, Holland will try to login using the standard .my.cnf conventions.

user = <user>

The user to connect to MySQL as.

password = <password>

The password for the MySQL user

socket = <socket>

The socket file to connect to MySQL with.

host = <host>

This would be used for connecting to MySQL remotely.

port = <port>

Used if MySQL is running on a port other than 3306.

Other Plugins

pgdump Provider Configuration [pgdump] Backs up a PostgreSQL instance using the pgdump utility.

[pgdump] **format** = custom | tar | plain (default: custom)

Defines the –format option for pg_dump. This defaults to –format=custom. The custom format is required for pg_restore to do partial restore as well as enabling parallel restores.

additional-options = <command-string>

Pass additional options to the pg_dump command

[compression] Specify various compression settings, such as compression utility, compression level, etc.

method = gzip | pigz | bzip | lzop | lzma | gpg

Define which compression method to use. Note that some methods may not be available by default on every system and may need to be compiled or installed and may not work with all the compression options.

inline = yes | no

Whether or not to pipe the output of mysqldump into the compression utility. Enabling this is recommended since it usually only marginally impacts performance, particularly when using a lower compression level.

level = 0-9

Specify the compression ratio. The lower the number, the lower the compression ratio, but the faster the backup will take. Generally, setting the lever to 1 or 2 results in favorable compression of textual data and is noticeably faster than the higher levels. Setting the level to 0 effectively disables compression.

bin-path = <full path to utility>

This only needs to be defined if the compression utility is not in the usual places or not in the system path.

options = <string of otpions>

Pass additional options not included in the above directly to the compression utility (e.g. `--compress-algo=bzip2` if using 'gpg').

[pgauth] **username** = <name>

Username for pg_dump to authenticate with

password = <string>

Password for pg_dump to authenticate with

hostname = <string>

Hostname for pg_dump to connect with

port = <integer>

TCP port for pg_dump to connect on

Example Provider Configuration [example] There are currently no configuration options for the example provider. It is provided as a skeleton for anyone wishing to write their own provider plugin.

Helper Plugins

[compression] Specify various compression settings, such as compression utility, compression level, etc.

method = gzip | pigz | bzip | lzop | lzma | gpg

Define which compression method to use. Note that some methods may not be available by default on every system and may need to be compiled or installed and may not work with all the compression options.

inline = yes | no

Whether or not to pipe the output of mysqldump into the compression utility. Enabling this is recommended since it usually only marginally impacts performance, particularly when using a lower compression level.

level = 0-9

Specify the compression ratio. The lower the number, the lower the compression ratio, but the faster the backup will take. Generally, setting the lever to 1 or 2 results in favorable compression of textual data and is noticeably faster than the higher levels. Setting the level to 0 effectively disables compression.

bin-path = <full path to utility>

This only needs to be defined if the compression utility is not in the usual places or not in the system path.

options = <string of otpions>

Pass additional options not included in the above directly to the compression utility (e.g. `--compress-algo=bzip2` if using 'gpg').

Backup Set Config Example

Here is an example backup set which uses mysqldump to backup all but a few databases, in a one-file-per-database fashion. For more specific examples, consult the documentation for the specific provider plugin you wish to use (see above).

```
[holland:backup]
plugin = mysqldump
backups-to-keep = 1
auto-purge-failures = yes
purge-policy = after-backup
estimated-size-factor = 0.25

[mysqldump]
extra-defaults = no
lock-method = auto-detect
databases = *
exclude-databases = "mydb", "myotherdb"
```

```
exclude-invalid-views = no
flush-logs = no
flush-privileges = yes
dump-routines = no
dump-events = no
stop-slave = no
max-allowed-packet = 128M
bin-log-position = no
file-per-database = yes
estimate-method = plugin

[compression]
method = gzip
inline = yes
level = 1

[mysql:client]
defaults-extra-file = ~/.my.cnf
```

CHAPTER 2

Indices and tables

- genindex
- modindex
- search