<?xml version="1.0" encoding="utf-8" ?>

TKLBAM

TurnKey GNU/Linux Backup and Migration

Author: Liraz Siri <liraz@turnkeylinux.org>

Date: 2013-11-11

Manual section: 8

Manual group: backup

SYNOPSIS

tklbam <command> [arguments]

ENVIRONMENT VARIABLES

TKLBAM_CONF: Path to TKLBAM configurations dir (default: /etc/tklbam)

TKLBAM_REGISTRY:

Path to TKLBAM registry (default: /var/lib/tklbam)

DESCRIPTION

TKLBAM (TurnKey GNU/Linux Backup and Migration), is a smart automated backup and restore facility for the TurnKey GNU/Linux Virtual Appliance Library.

Goals

TKLBAM is designed to provide an efficient system-level backup of changed files, users, databases and package management state. This system-level backup can be restored automatically on any installation of the same type of virtual appliance, regardless of the underlying hardware or location. The intended result is a functionally equivalent copy of the original system.

It is also designed to assist in migration of data and system configurations between different versions of the same type of virtual appliance though for some applications, additional manual steps, such as a database schema update, may be required to complete migration between versions.

Features

- Figures out which files, databases and packages need to be backed up automatically by detecting changes since installation
- Lets you simulate a backup beforehand to review if you need to include or exclude any file paths or databases
- Restoring automatically adds missing users and groups, fixes file ownership and permission issues
- Makes it easy to create lightweight backups that can be migrated across operating system versions
- Supports a wide variety of storage targets (e.g., Amazon S3, local filesystem, rsync, ftp, SSH, etc.)
- Leverages Duplicity to creates PGP encrypted, incremental backup archives
- Easy to embed in an existing backup solution

Key elements

TurnKey Hub: a web service which provides the front-end for backup management. The user links an appliance to a specific Hub account identified by an API KEY.

Backup profile: describes the installation state for a specific type and version of appliance. An appropriate profile is downloaded from the Hub the first time you backup, or as required if there is a profile update (e.g., bugfix).

Delta: a set of changes since installation to files, users, databases and package management state. This is calculated at backup time by comparing the current system state to the installation state described by the backup profile.

Encryption key: generated locally on your server and used to directly encrypt your backup volumes. By default key management is handled transparently by the Hub. For extra security, the encryption key may be passphrase protected cryptographically. An escrow key can be created to protect against data loss in case the password is forgotten.

Duplicity: back-end primitive that the backup and restore operations invoke to encode, transfer and decode encrypted backup volumes which contain the delta. It communicates directly with the storage target (e.g., Amazon S3). In normal usage the storage target is auto-configured by the Hub. Duplicity uses the rsync algorithm to support efficient incremental backups. It uses GnuPG for symmetric encryption (AES).

Amazon S3: a highly-durable cloud storage service where encrypted backup volumes are uploaded to by default. To improve network performance, backups are routed to the closest datacenter, based on a GeoIP lookup table.

Any storage target supported by Duplicity can be forced but this complicates usage as the Hub can only work with S3. This means backups, encryption keys and authentication credentials will need to be managed by hand.

Principle of operation

Every TKLBAM-supported TurnKey appliance has a corresponding backup profile that describes installation state and includes an appliance-specific list of files and directories to check for changes. This list does not include any files or directories maintained by the package management system.

A delta (I.e., changeset) is calculated by comparing the current system state to the installation state. Only this delta is backed up and only this delta is re-applied on restore.

An exception is made with regards to database contents. These are backed up and restored whole, unless

otherwise configured by the user.

In addition to direct filesystem changes to user writeable directories (e.g., /etc, /var/www, /home) the backup delta is calculated to include a list of any new packages not originally in the appliance's installation manifest. During restore, the package management system is leveraged to install these new packages from the configured software repositories.

Users and groups from the backed up system are merged on restore. If necessary, uids / gids of restored files and directories are remapped to maintain correct ownership.

Similarly, permissions for files and directories are adjusted as necessary to match permissions on the backed up system.

COMMANDS

init: Initialization (links TKLBAM to Hub account)

passphrase: Change passphrase of backup encryption key

escrow: Create a backup escrow key (Save this somewhere safe)

backup: Backup the current system

list: List backup records

restore: Restore a backup

restore-rollback:

Rollback last restore

EXAMPLE USAGE SCENARIO

Alon is developing a new web site. He starts by deploying TurnKey LAMP to a virtual machine running on his laptop. This will serve as his local development server. He names it DevBox.

He customizes DevBox by:

- creating user 'alon'.
- extracting an archive of his web application to /var/www
- tweaking Apache configuration directives in /etc/apache2/httpd.conf until his web application works.
- installing php5-xcache via the package manager
- enabling xcache by editing a section in /etc/php5/apache2/php.ini
- creating a new database user with reduced privileges for his web application.
- configuring and installing the web application, which creates a new MySQL database.

After a few days of hacking on the web application, Alon is ready to show off a prototype of his creation to some friends from out of town.

He logs into the TurnKey Hub and launches a new TurnKey LAMP server in the Amazon EC2 cloud. He names it CloudBox.

On both DevBox and CloudBox Alon installs and initializes TKLBAM with the following commands:

```
apt-get update
apt-get install tklbam

# The API Key is needed to link tklbam to Alon's Hub account
tklbam-init QPINK3GD7HHT3A
```

On DevBox Alon runs a backup:

```
root@DevBox:~# tklbam-backup
```

Behind the scenes, TKLBAM downoads from the Hub a profile for the version of TurnKey LAMP Alon is using. The profile describes the state of DevBox right after installation, before Alon customized it. This allows TKLBAM to detect all the files and directories that Alon has added or edited since. Any new packages Alon installed are similarly detected.

As for his MySQL databases, it's all taken care of transparently but if Alon dug deeper he would discover that their full contents are being serialized and encoded into a special file structure optimized for efficiency on subsequent incremental backups. Between backups Alon usually only updates a handful of tables and rows, so the following incremental backups are very small, just a few KBs!

When TKLBAM is done calculating the delta and serializing database contents, it invokes Duplicity to encode backup contents into a chain of encrypted backup volumes which are uploaded to Amazon S3.

When Alon's first backup is complete, a new record shows up in the Backups section of his TurnKey Hub account.

Now to restore the DevBox backup on CloudBox:

```
root@CloudBox:~# tklbam-list
# ID  SKPP  Created     Updated     Size (GB)  Label
   1  No    2010-09-01  2010-09-01  0.02       TurnKey LAMP

root@CloudBox:~# tklbam-restore 1
```

When the restore is done Alon points his browser to CloudBox's IP address and is delighted to see his web application running there, exactly the same as it does on DevBox.

Alon, a tinkerer at heart, is curious to learn more about how the backup and restore process works. By default, the restore process reports what it's doing verbosely to the screen. But Alon had a hard time following the output in real time, because everything happened so fast! Thankfully, all the output is also saved to a log file at /var/log/tklbam-restore.

Alon consults the log file and can see that only the files he added or changed on DevBox were restored to CloudBox. Database state was unserialized. The xcache package was installed via the package manager. User alon was recreated. It's uid didn't conflict with any other existing user on CloudBox so the restore process didn't need to remap it to another uid and fix ownership of Alon's files. Not that it would matter to Alon either way. It's all automagic.

FILES

- /var/lib/tklbam: the registry

SEE ALSO

`tklbam-faq` (7)

<?xml version="1.0" encoding="utf-8" ?>

tklbam-init

Initialization: links TKLBAM to Hub account and downloads backup profile)

Author: Liraz Siri <liraz@turnkeylinux.org>

Date: 2013-11-05

Manual section: 8

Manual group: backup

SYNOPSIS

tklbam-init [--force] [*API-KEY*]

By default, this links TKLBAM to your Hub account and downloads an appropriate backup profile, which is used to calculate the list of system changes we need to backup. On a TurnKey system the profile describes the installation state of the appliance and contains a list of packages, filesystem paths to scan for changes and an index of the contents of those paths which records timestamps, ownership and permissions. On a non-TurnKey system the default backup profile will not describe installation state, only a list of directories to backup.

ARGUMENTS

API-KEY Cut and paste this from your Hub account's user profile.

If you do not provide the API-KEY as a command line argument you will be prompted for it interactively (unless you use the --solo option)

OPTIONS

`--force` Force re-initialization with new API-KEY

`--force-profile=`*`PROFILE_ID`*

Force a specific backup profile (e.g., "core", "turnkey-core-13.0-wheezy-amd64")

Without --force-profile the profile_id is automatically detected.

`--force-profile=`*`empty`*

"empty" is a special profile_id value that creates an empty backup profile. Backup configurations will only be taken from /etc/tklbam.

`--force-profile=`*`PATH`*

Path to a custom backup profile

Details: tklbam-internal create-profile --help

`--solo` Solo mode: disables link to Hub.

You'll need to --force-profile=empty or use a custom profile

tklbam-backup will only work with --address or --dump options

tklbam-restore will only work with --address or a backup extract

SECURITY WARNING

Providing your Hub account's APIKEY as a command line argument is potentially less secure than allowing tklbam-init to prompt you for it interactively:

- The shell may save the APIKEY to its history file (e.g., ~/.bash_history)
- The APIKEY may briefly show up in the process list

USAGE EXAMPLES

```
# initialize TKLBAM
tklbam-init

# initialize TKLBAM with the core profile
tklbam-init --force-profile=core

# initialize TKLBAM with a non-default registry path
TKLBAM_REGISTRY=/var/lib/tklbam2 tklbam-init

# initialize TKLBAM in solo mode with an empty profile
tklbam-init --solo --force-profile=empty
```

SEE ALSO

`tklbam` (8), `tklbam-faq` (7)

<?xml version="1.0" encoding="utf-8" ?>

tklbam-list

List backup records

Author: Liraz Siri <liraz@turnkeylinux.org>

Date: 2010-09-01

Manual section: 8

Manual group: backup

SYNOPSIS

tklbam-list [<format>]

ARGUMENTS

By default uses a built-in format, unless a user-specified format is specified.

Format variables:

```
%id                     Backup id
%label                  Descriptive label
%turnkey_version        Appliance version code
%server_id              Associated server id (- if empty)
%created                Date the backup record was created
%updated                Date the backup record was last updated
%size                   Aggregate size of backup, in bytes
%address                Backup target address
%key                    Base64 encoded encrypted keypacket
%skpp                   Secret Key Passphrase Protection (Yes/No)
```

USAGE EXAMPLES

list

list "backup_id=%backup_id label=%label size=%{size}"

SEE ALSO

`tklbam` (8), `tklbam-faq` (7)

<?xml version="1.0" encoding="utf-8" ?>

tklbam-passphrase

Change passphrase of backup encryption key

Author: Liraz Siri <liraz@turnkeylinux.org>

Date: 2010-09-01

Manual section: 8

Manual group: backup

SYNOPSIS

tklbam-passphrase [-options]

OPTIONS

`--random` Choose a secure random password (and print it)

SEE ALSO

`tklbam` (8), `tklbam-faq` (7)

<?xml version="1.0" encoding="utf-8" ?>

tklbam-escrow

Create a backup escrow key

Author: Liraz Siri <liraz@turnkeylinux.org>

Date: 2010-09-01

Manual section: 8

Manual group: backup

SYNOPSIS

tklbam-escrow [-options] *KEYFILE*

DESCRIPTION

Creates an escrow key you can pass directly to the restore command. Save this somewhere safe.

ARGUMENTS

KEYFILE File path to save the escrow key (- for stdout)

OPTIONS

`--no-passphrase, -P`

Don't encrypt escrow key with a passphrase

`--random-passphrase, -R`

Choose a secure random passphrase (and print it)

SEE ALSO

`tklbam` (8), `tklbam-faq` (7)

<?xml version="1.0" encoding="utf-8" ?>

/etc/tklbam/hooks.d

TKLBAM Hooks

Author: Liraz Siri <liraz@turnkeylinux.org>

Date: 2013-08-26

Manual section: 5

Manual group: backup

DESCRIPTION

TKLBAM has a nifty, general purpose hooks mechanism you can use to trigger useful actions on backup and restore.

Things you might want to use hooks for:

- Cleaning up temporary files
- Stopping/starting services to increase data consistency
- Encoding/decoding data from non-supported databases
- Using LVM to create/restore a snapshot of a fast changing volume

Hooks are located at /etc/tklbam/hooks.d

Only executable hooks are executed. To enable a hook:

```
chmod +x /etc/tklbam/hooks.d/example
```

Outline of hook invocation

Tip: try enabling the example hook and examine the backup or restore console output to see where the hook is executed.

Outline of hook invocation in backup process:

```
"pre" hook
tklbam creates "extras" (backup metadata)
"inspect" hook: current working directory is /TKLBAM
tklbam runs duplicity to create/update backup archives
"post" hook
```

Outline of hook invocation in restore process:

```
"pre" hook
tklbam runs duplicity to get extras (backup metadata) + overlay
"inspect" hook: current working directory is /tmp/<random-archive-path>/TKLBAM
tklbam applies restore to system
"post" hook
```

Example hook

```
#!/bin/bash -e
# This is a disabled hook example.
# To enable, make it executable: chmod +x /etc/tklbam/hooks.d/example

# hooks are always called with two arguments
op=$1
state=$2

if [ "$op" = "restore" ]; then
    echo -n "A restore operation called this hook "
elif [ "$op" = "backup" ]; then
    echo -n "A backup operation called this hook "
fi

if [ "$state" = "pre" ]; then
    echo "BEFORE $op started"
elif [ "$state" = "inspect"]; then

    if [ "$op" = "restore" ]; then
        echo -n "Inspect hook runs after duplicity downloaded backup archive. extras
path = $(pwd)"
    elif [ "$op" = "backup" ]; then
        echo -n "Inspect hook runs before duplicity uploads backup archive. extras
path = $(pwd)"
```

```
    fi

elif [ "$state" = "post" ]; then
    echo "AFTER $op finished"
fi

# `false` returns a non-zero exitcode
# Uncomment the next line to raise an error

#false
```

Default hooks

- /etc/tklbam/hooks.d/fixclock: activated by default, this hooks uses ntpdate to sync the clock, to prevent duplicity from ignoring backup archives "from the future".
- /etc/tklbam/hooks.d/example: disabled by default. When enabled this hook prints out "debugging" messages showing where the hook is being executed. This is designed to illustrate how tklbam hooks work.

<?xml version="1.0" encoding="utf-8" ?>

tklbam-backup

Backup the current system

Author: Liraz Siri <liraz@turnkeylinux.org>

Date: 2013-09-05

Manual section: 8

Manual group: backup

SYNOPSIS

tklbam-backup [-options] [override ...]

ARGUMENTS

<*override*> := `<filesystem-override>` | `<database-override>`

Overrides are usually configured in /etc/tklbam/overrides.

Filesystem overrides

`<filesystem-override>` := -?/path/to/include/or/exclude

This includes or excludes additional files and directories from being backed up if they've changed since

installation.

Overrides defaults in /var/lib/tklbam/profile/dirindex.conf

Gotchas:

- If you add a directory handled by package management this may break package management on the system you restore to.
- Only changes (e.g., new files, edited files, deleted files) from the base installation are included in a backup.

Examples:

```
# exclude log files in /var/www
-/var/www/*/logs

# ignores changes to webmin configuration
-/etc/webmin

# include the contents of an external hard disk...
/mnt/images
```

Database overrides

`<database-override>` := -?mysql:database[/table]

`<database-override>` := -?pgsql:database[/table]

By default ALL databases and database tables are backed up.

Adding a positive override (e.g., mysql:mydatabase) changes the default behavior so that only the database or table specified in the override is included in the backup.

Negative overrides exclude a database (e.g., -mysql:mydatabase) or table (e.g., -mysql:mydatabase/mytable) from being included in the backup.

Excluding a table only excludes its data. The schema of an excluded table is still backed up, as it takes up a trivial amount of space and other tables or views may depend on it.

You can mix positive overrides with negative overrides.

Examples:

```
# exclude Drupal6 sessions table
-mysql:drupal6/sessions

# only include drupal6 database
mysql:drupal6

# only include mahara postgres database
pgsql:mahara
```

OPTIONS

`--dump=`*`DIR`* Dump a raw backup extract to path. Tip: tklbam-restore path/to/raw/extract/

`--raw-upload=`*`PATH`*

Use Duplicity to upload raw path contents to --address

`--address=`*`TARGET_URL`*

custom backup target URL. Default: S3 storage bucket automatically configured via Hub

Supported storage backends and their URL formats:

```
file:///some_dir
rsync://user[:password]@other.host[:port]//absolute_path
rsync://user[:password]@other.host[:port]/relative_path
rsync://user[:password]@other.host[:port]::/module/some_dir
s3://other.host/bucket_name[/prefix]
s3+http://bucket_name[/prefix]
ftp://user[:password]@other.host[:port]/some_dir
ftps://user[:password]@other.host[:port]/some_dir
hsi://user[:password]@other.host[:port]/some_dir
imap://user[:password]@other.host[:port]/some_dir
scp://user[:password]@other.host[:port]/some_dir
ssh://user[:password]@other.host[:port]/some_dir
tahoe://alias/directory
webdav://user[:password]@other.host/some_dir
webdavs://user[:password]@other.host/some_dir
gdocs://user[:password]@other.host/some_dir
```

Note: Alternate targets may require additional dependancies E.g. to use SSH you will need to

install 'python-paramiko' (*apt-get update && apt-get install python-paramiko*). Currently the requirements of each target are not documented; however if you keep in mind that TKLBAM uses Duplicity as a back end, google should provide guidance.

`--resume` Resume aborted backup session

`--disable-resume`

Disable implicit --resume when rerunning an aborted backup

`--simulate` Simulate operation. Don't actually backup. Useful for inspecting /TKLBAM by hand.

`--quiet, -q` Be less verbose

`--logfile=`*`PATH`* Path of file to log output to. Default: /var/log/tklbam-backup

`--debug` Run $SHELL before Duplicity

Configurable options

`--volsize `*`MB`* Size of backup volume in MBs. Default: 50

`--s3-parallel-uploads=`*`N`*

Number of parallel volume chunk uploads Default: 1

`--full-backup` *`FREQUENCY`*

Time frequency of full backup. Default: 1M

<frequency> := now | <int>[DWM]

e.g.,:

```
now - always do a full backup

60m - 60 minutes
12h - 12 hours
3D - three days
2W - two weeks
1M - one month
```

`--skip-files` Don't backup filesystem

`--skip-database`

Don't backup databases

`--skip-packages`

Don't backup new packages

`--force-profile=`*`PROFILE_ID`*

Force backup profile (e.g., "core")

Resolution order for configurable options:

1. comand line (highest precedence)
2. configuration file (/etc/tklbam/conf):

   ```
   # comment
   <option-name> <value>
   ```

3. built-in default (lowest precedence)

USAGE EXAMPLES

```
# Full system-level backup
tklbam-backup

# Same result as above but in two steps: first dump to a directory, then upload it
tklbam-backup --dump=/tmp/mybackup
tklbam-backup --raw-upload=/tmp/mybackup

# Backup Duplicity archives to a custom address on the local filesystem
```

```
tklbam-backup --address=file:///mnt/backups/mybackup
tklbam-escrow this-keyfile-needed-to-restore-mybackup.escrow

# Simulate a backup that excludes the mysql customers DB and the 'emails' table in
the webapp DB
# Tip: usually you'd want to configure excludes in /etc/tklbam/overrides
tklbam-backup --simulate -- -/srv -mysql:customers -mysql:webapp/emails

# Create separate backups with unique backup ids containing the previously excluded
items
# Tip: use tklbam-status after tklbam-backup to determine the Hub backup ID
export TKLBAM_REGISTRY=/var/lib/tklbam.customers-and-webapp-emails
tklbam-backup --skip-files --skip-packages -- mysql:customers mysql:webapp/emails

export TKLBAM_REGISTRY=/var/lib/tklbam.raw-srv
tklbam-backup --raw-upload=/srv
```

FILES

Configuration files:

/etc/tklbam/overrides, /etc/tklbam/conf,
/etc/tklbam/hooks.d

Local cache of profile:

$TKLBAM_REGISTRY/profile

By default TKLBAM_REGISTRY=/var/lib/tklbam

SEE ALSO

`tklbam` (8), `tklbam-faq` (7), `tklbam-hooks` (5)

<?xml version="1.0" encoding="utf-8" ?>

tklbam-restore

Restore a backup

Author: Liraz Siri <liraz@turnkeylinux.org>

Date: 2013-09-05

Manual section: 8

Manual group: backup

SYNOPSIS

tklbam-restore [-options] [<hub-backup>]

ARGUMENTS

<*hub-backup*> := backup-id || unique label pattern || path/to/backup/extract

OPTIONS

General options

--raw-download=path/to/backup/ Download backup to directory without doing a system restore

Duplicity related options

`--time=`*`TIME`* Time to restore from

TIME := YYYY-MM-DD | YYYY-MM-DDThh:mm:ss | <int>[mhDWMY]

e.g.,:

```
2010-08-06 - 2010, August 6th, 00:00

2010-08-07T14:00 - 2010, August 7th 14:00 UTC

6m - 6 minutes
5h - 5 hours
4D - 4 days ago
3W - 3 weeks ago
2M - 2 months ago
1Y - 1 year ago
```

`--keyfile=`*`KEYFILE`*

Path to tklbam-escrow created keyfile

Default: automatically retrieved from the Hub

`--address=`*`TARGET_URL`*

Custom backup target URL (needs --keyfile)

Default: S3 storage bucket automatically provided by Hub

Supported storage backends and their URL formats:

```
file:///some_dir
rsync://user[:password]@other.host[:port]//absolute_path
rsync://user[:password]@other.host[:port]/relative_path
rsync://user[:password]@other.host[:port]::/module/some_dir
s3://other.host/bucket_name[/prefix]
```

```
s3+http://bucket_name[/prefix]
ftp://user[:password]@other.host[:port]/some_dir
ftps://user[:password]@other.host[:port]/some_dir
hsi://user[:password]@other.host[:port]/some_dir
imap://user[:password]@other.host[:port]/some_dir
scp://user[:password]@other.host[:port]/some_dir
ssh://user[:password]@other.host[:port]/some_dir
tahoe://alias/directory
webdav://user[:password]@other.host/some_dir
webdavs://user[:password]@other.host/some_dir
gdocs://user[:password]@other.host/some_dir
```

System restore options

`--simulate` Do a dry run simulation of the system restore

`--limits=`*`LIMITS`*

Restore filesystem or database limitations. You can use this

to control what parts of the backup will be restored.

Preceding a limit with a minus sign turns it into an exclusion.

LIMITS := "`LIMIT`-1 .. `LIMIT`-N"

`LIMIT` := -?(/path/to/include/or/exclude | mysql:database[/table] | pgsql:database[/table])

`--skip-files` Don't restore filesystem

`--skip-database`

Don't restore databases

`--skip-packages`

Don't restore new packages

`--logfile=`*`PATH`* Path to log file. Default: /var/log/tklbam-restore

`--no-rollback` Disable rollback

`--silent` Disable feedback

`--force` Disable sanity checking

`--debug` Run $SHELL after Duplicity

Configurable options

`--restore-cache-size=`*`SIZE`*

The maximum size of the download cache default: 50%

`--restore-cache-dir=`*`PATH`*

The path to the download cache directory default:
/var/cache/tklbam/restore

Resolution order for configurable options:

1. comand line (highest precedence)
2. configuration file (/etc/tklbam/conf):

   ```
   # comment
   <option-name> <value>
   ```

3. built-in default (lowest precedence)

USAGE EXAMPLES

```
# Restore Hub backup id 1
tklbam-restore 1

# Same result as above but in two steps: first download the extract, then apply it
tklbam-restore 1 --raw-download=/tmp/mybackup
tklbam-restore /tmp/mybackup

# Restore backup created with tklbam-backup --raw-upload=/srv
tklbam-restore 2 --raw-download=/srv

# Restore from Duplicity archives at a custom backup address on the local filesystem
tklbam-restore --address=file:///mnt/backups/mybackup --keyfile=mybackup.escrow

# Simulate restoring Hub backup id 1 while excluding changes to the /root path,
# mysql 'customers' DB, and the 'emails' table in the 'webapps' DB
tklbam-restore 1 --simulate --limits="-/root -mysql:customers -mysql:webapp/emails"

# Simulate restoring only the /root files in Hub backup id 1
tklbam-restore 1 --simulate --skip-database --skip-packages --limits="/root"
```

FILES

Configuration files:

/etc/tklbam/conf, /etc/tklbam/hooks.d

Restore cache: /var/cache/tklbam/restore (by default, see --restore-cache-dir=PATH option)

SEE ALSO

`tklbam` (8), `tklbam-faq` (7), `tklbam-hooks` (5)

<?xml version="1.0" encoding="utf-8" ?>

TKLBAM-FAQ

Frequently Asked Questions

Author: Liraz Siri <liraz@turnkeylinux.org>

Date: 2013-11-07

Manual section: 7

Manual group: backup

GENERAL QUESTIONS

Is TKLBAM free software?

Yes, TKLBAM is licensed under the GPL3. You don't have to care about free software ideology to appreciate the advantages. Any code running on your server doing something as critical as encrypted backups should be available for peer review and modification.

Where can I install TKLBAM?

If you're using any TurnKey derived system, you don't need to install it as TKLBAM is a bundled into the TurnKey Core.

If you're using a generic Debian or Ubuntu derived system you can install it with the following shell command:

```
wget -O - -q https://raw.github.com/turnkeylinux/tklbam/master/contrib/ez-apt-install.sh | PACKAGE=tklbam /bin/bash
```

This adds the TurnKey package repository to your APT sources and uses APT to install the tklbam package and its dependencies.

Using TKLBAM on TurnKey GNU/Linux provides the best experience but it will also work well with any Debian or Ubuntu derived system and even with other Linux distributions if you install from source and use the --skip-packages option to disable integration with APT, the Debian package manager.

When you use TKLBAM with TurnKey GNU/Linux it takes advantage of the known fixed installation state to make the smallest possible backup. For example, it will only backup /etc configuration files that you have changed. This makes migration easier by increasing visibility into what actually changed. By comparison on a generic Debian or Ubuntu system it will backup all /etc configuration files.

What can I use for backup storage?

Pretty much anything, though storing backups to Amazon S3 is easiest because authentication and key management are automatic. You just need to run:

```
tklbam-backup
```

But you can also backup to any storage target supported by TKLBAM's back-end Duplicity including the local filesystem, NFS, Rsync, SSH, FTP, WebDAV, Rackspace CloudFiles and even IMAP.

The local filesystem is one of the easier storage targets to use because you don't need to mess around with authentication credentials.

So assuming you want to store your backup at /mnt/otherdisk:

```
tklbam-backup --address file:///mnt/otherdisk/tklbam/backup
tklbam-escrow /mnt/otherdisk/tklbam/key
```

And restore like this:

```
tklbam-restore --address file:///mnt/otherdisk/tklbam/backup \
               --keyfile=/mnt/otherdisk/tklbam/key
```

Not as easy as the Hub-enabled "automatic" mode, but still vastly easier than your conventional backup process. The disadvantage is that you won't be able to restore/test your backup in the cloud, or from a VM running in another office branch (for example). Also keep in mind that a physical hard disk, even a RAID array, provides much lower data reliability compared with Amazon S3.

For this reason we recommend users use local backups to supplement cloud backups (e.g., providing fast local access).

Which databases are supported?

Currently, only MySQL and PostgreSQL have built-in support but TKLBAM can work with other databases so long as you configure custom serialization/unserialization procedures in a hook script.

USAGE QUESTIONS

How do I tune and optimize a TKLBAM backup?

One of my favorite ways to do this:

```
# step 1: generate a backup dump
tklbam-backup --dump=/tmp/mybackup

# step 2: interactively review the dump's file contents & disk usage
cd /tmp/mybackup
apt-get install ncdu
ncdu

# step 3: add includes or excludes, go back to step 1, rinse, repeat
vim /etc/tklbam/overrides

# Everything perfect?
tklbam-backup --upload-raw=/tmp/mybackup
```

By default, TKLBAM will automatically determine what paths and databases need to be backed up on a given TurnKey system according to the backup profile it gets from the Hub. The default profile tracks changes to the user-servicable, customizable parts of the filesystem (e.g., /etc /root /home /var /usr/local /var /opt /srv) while ignoring changes in areas maintained by the package management system.

You can "override" the default backup profile configuration by specifying overrides, either on the command line, or preferably by editing the /etc/tklbam/overrides configuration file.

How does TKLBAM know what to backup on my system?

Every TurnKey appliance that TKLBAM supports has a corresponding backup profile, which is downloaded from the Hub when you initialize TKLBAM. The profile is used to calculate the list of system changes we need to backup. It usually describes the installation state of a TurnKey appliance and contains a list of packages, filesystem paths to scan for changes and an index of the contents of those paths which records timestamps, ownership and permissions.

You can also generate your own custom profiles with the following command:

```
tklbam-internal create-profile
```

The backup profile is stored in /var/lib/tklbam/profile and contains the following text files:

1. dirindex.conf: a list of directories to check for changes by default. This list does not include any files or directories maintained by the package management system.
2. dirindex: appliance installation state - filesystem index
3. packages: appliance installation state - list of packages

Users can override which files and directories are checked for changes by configuring overrides (See below).

Why is an override called an override?

Because it "overrides" the default backup configuration in the appliance profile.

How do I remove a file or directory from being included in my backup?

By adding a negative override to /etc/tklbam/overrides:

```
echo -/var/www/*/logs >> /etc/tklbam/overrides
```

You can also specify overrides on the command line at backup time:

```
tklbam-backup -- -/var/www/*/logs
```

How do I add a directory to my backup?

By adding an override to /etc/tklbam/overrides:

```
echo /mnt/images >> /etc/tklbam/overrides
```

Or on the command line:

```
tklbam-backup /var/www/*/logs
```

Make sure you understand the implications of doing this. For example, if you add a directory handled by package management this may break package management on the system you restore to.

How do I exclude a database or table from my backup?

By adding a negative database override to /etc/tklbam/overrides:

```
# exclude drupal5 database
echo -mysql:drupal5 >> /etc/tklbam/overrides

# exclude sessions table in drupal6 database
echo -mysql:drupal6/sessions >> /etc/tklbam/overrides
```

Or on the command line:

```
tklbam-backup -- -mysql:drupal6/page_cache
```

By default ALL databases are backed up so adding a negative database override excludes only that database or table from the backup.

Excluding a table only excludes its data. The schema is still backed up as long as the database is included.

Specifying a positive database override changes the default behavior so that only the database or table specified in the override is included in the backup.

You can mix positive overrides with negative overrides.

Can I have multiple TKLBAM backups on a single system?

Yes. For example, let's say your default TKLBAM backup is several gigabytes in size and you'd like to create a lighter 100 MB backup that will be updated more frequently and take less time to update/restore:

```
cp -a /etc/tklbam /etc/tklbam.light

echo -/var/www/\*/logs >> /etc/tklbam.light/overrides
echo -/home/liraz/bigfiles >> /etc/tklbam.light/overrides
echo -mysql:mydatabase/bigtable >> /etc/tklbam.light/overrides

export TKLBAM_CONF=/etc/tklbam.light

mkdir /var/lib/tklbam.light
export TKLBAM_REGISTRY=/var/lib/tklbam.light

tklbam-init
tklbam-backup
```

For convenience you may want to create a script that sets the TKLBAM_REGISTRY and TKLBAM_CONF environment variables:

```
cat > /usr/local/bin/tklbam-backup-light << EOF
#!/bin/bash
export TKLBAM_CONF=/etc/tklbam.light
export TKLBAM_REGISTRY=/var/lib/tklbam.light

tklbam-backup
EOF

chmod +x /usr/local/bin/tklbam-backup-light
```

Can I use TKLBAM to only backup a single directory?

Yes. Here are a couple of recommended ways to do this:

1. Create a separate backup with an empty backup profile:

   ```
   export TKLBAM_REGISTRY=/var/lib/tklbam.srv
   export TKLBAM_CONF=/etc/tklbam.srv

   tklbam-init --force-profile=empty
   tklbam-backup --skip-packages --skip-database -- /srv
   ```

2. Use the --raw-upload option

   This lobotomizes TKLBAM so instead of creating a system level backup it just backs up the directory you specify. In other words, --raw-upload turns TKLBAM into a directory-level backup tool rather than a system-level backup tool.

   For example, let's say you have a collection of big files at /srv that you don't want to include in your system backup (e.g., because you don't want to bloat your backup).

   So you configure an overrides to exclude the /srv directory from your default backup and create another TKLBAM backup just for the big files:

   ```
   echo -/srv >> /etc/tklbam/overrides
   export TKLBAM_REGISTRY=/var/lib/tklbam.srv-raw
   tklbam-backup --raw-upload=/srv
   ```

   Later, you'll need to use the --raw-download option to restore:

   ```
   tklbam-restore --raw-download=/srv <your-backup-id>
   ```

   If you don't use the raw-download option, TKLBAM will assume you are trying to restore a system-level backup and you'll get an error.

What's the difference between a full backup and an incremental backup?

A full backup is a backup that can be restored independently of any other backup. An incremental backup links with the last backup before it and only includes changes made since.

Backup chains are links of backup sessions which start with a full backup, and then a series of incremental backups each recording only the changes made since the backup before it. Incremental backups are useful because they are fast and efficient.

Restoring an incremental backup requires retrieving the volumes of all backup sessions made before it, up to and including the full backup that started the chain. The longer the backup chain, the more time it will take to restore.

How often does a does a full backup happen, how can I configure this?

By default, a full backup will happen if the last full backup is older than 30 days. Between full backups, all backup sessions are incremental.

We recommend enabling the daily backup cron job so that daily incremental backups happen automatically:

```
chmod +x /etc/cron.daily/tklbam-backup
```

You can override the default by setting the full-backup parameter in the tklbam configuration:

```
# create a full backup every 14 days
echo full-backup 14D >> /etc/tklbam/conf
```

I forgot my passphrase, and I "lost" my escrow key. Can you help me?

Sorry, if your server is gone (e.g., terminated EC2 instance) nobody can help you. Next time either save an escrow key somewhere[s] safe or don't set a passphrase.

The encryption key for your backup was generated locally on your server not ours. Passphrase protection uses special cryptographic countermeasures to make typical cracking techniques (e.g., dictionary attacks) very difficult even with access to massive computer resources.

Note, if the system you backed up is still available, just log into it as root and change the passphrase (you don't need to know the old passphrase):

```
tklbam-passphrase
```

AMAZON S3 QUESTIONS

How do I monitor how much traffic is being uploaded or downloaded?

We recommend the following package:

```
apt-get install iftop
iftop
```

How can I throttle how much bandwidth TKLBAM uses?

Here's one way to do it:

```
apt-get install trickle
trickle -u 100  -d 100 tklbam-backup
```

Do I have to store my backups in the Amazon S3 storage cloud?

No! TKLBAM stores backups in the cloud for convenience, but it also supports local / custom backup storage targets.

There are two main alternatives to letting TKLBAM store a backup in the cloud:

1. Low-level tklbam-backup --dump option: lets you dump the raw TKLBAM backup extract to a directory, which you can then store anyway you like.

For example here's how we'd a system backup into a simple unencrypted tarball:

```
cd /tmp
mkdir mybackup
tklbam-backup --dump=mybackup/
tar jcvf mybackup.tar.bz2 mybackup/
```

And later restore it like this:

```
cd /tmp
tar jxvf mybackup.tar.bz2
tklbam-restore mybackup/
```

The --dump option bypasses Duplicity, which usually create a series of encrypted archive files that can be incrementally updated. These archive files are stored by default in the Amazon S3 storage cloud but you can override this with the --address option and specify any storage back-end supported by Duplicity (e.g., local directory, rsync over ssh, ftp, sftp, etc).

2. High-level tklbam-backup --address option: lets you specify a custom backup target URL that is passed on to Duplicity.

   It is highly recommended to rehearse a trial restore. Testing your backups is always a good idea, and even more so with a custom --address as this may complicate usage.

   The Hub normally helps you manage your backup's metadata when it auto-configures the storage address. If you specify a manual address you need to manage storage locations, encryption keys and authentication credentials by hand.

Why can't I access TKLBAM storage buckets with other Amazon S3 tools?

TKLBAM doesn't store its data in generic S3 buckets, but in an isolated TKLBAM-specific area on S3. This means generic S3 tools such as the AWS management console, or S3Fox will not be able to access the storage buckets in which TKLBAM backup volumes reside.

What are the advantages of isolating TKLBAM Amazon S3 storage?

1. Easier sign up process. Users don't need to know anything about S3 API keys or understand the implications of giving them to us.
2. Security: you don't need to give us access to your generic S3 account. If someone compromises your regular AWS API Key they still can't get to your encrypted backup volumes and say... delete them.
3. Cost transparency: TKLBAM related storage charges show up separately from your generic S3 storage.

What happens if my payment method to Amazon is invalidated?

Amazon supports payment by credit card and bank account. We recommend heavy users add a bank account as their payment method, as it's usually more permanent than a credit card.

In any case, if your payment method is invalidated (e.g., cancelled or expired credit card), billing will fail and Amazon will attempt to contact you (e.g., by e-mail) to provide a new, valid payment method.

How much does cloud backup storage cost?

Amazon S3 cloud storage fees are around $0.15/GB per month.

You can use simulation mode to calculate how much uncompressed data TKLBAM is going to store in a full backup:

```
$ tklbam-backup --simulate
CREATING /TKLBAM
FULL UNCOMPRESSED FOOTPRINT: 148.30 MB in 6186 files
```

In practice, the actual footprint of a full backup will usually be smaller due to compression, but this depends on the type of data being compressed (e.g., text compresses very well, video very poorly).

By default, a full backup is performed if one month has passed since the last full backup. In between, incremental backups will be performed which only record changes since the last backup. The full backup frequency can be customized. See the manual page for details.

The Hub says my backup costs $0.00, what am I really paying?

If you notice $0.00 in the backups console, there's no need to open a support request. It's not a bug. At 15 cents per gigabyte, if you have just a few megabytes of data Amazon doesn't charge you anything.

Backups start from around 100KB for a freshly installed TurnKey appliance. Remember, TKLBAM only saves changes you've made since the appliance was installed.

FAULT TOLERANCE FOR THE PARANOID IT GUY

Is the Hub a central point of failure for TKLBAM?

No, for a couple of reasons:

1. After the initial setup, TKLBAM communicates directly with Amazon S3. Even if the Hub does down, backups will not be interrupted.
2. You can use TKLBAM without linking it to the Hub at all. See the tklbam-init --solo option.

If the Hub goes down, will my backup cron jobs still work?

Yes. Backups which have already been configured will continue to work normally. If TKLBAM can't reach the Hub it just uses the locally cached profile and S3 address.

If my connection to the Hub goes down, can I still restore?

Yes - manually. It just won't be as easy. You'll need to do a couple of steps by hand:

1. transfer the escrow key to the restore target.

   This means you'll need to have stored the escrow key somewhere safe or be able to create it on the backed up machine.

2. specify the S3 address and the key manually when you restore.

   For more details see the tklbam-restore documentation.

If the Hub fails can I still create a new backup?

Yes - but you'll have to manage it manually. The Hub won't know anything about these backups so you'll have manage keys and authentication credentials by hand.

SEE ALSO

`tklbam` (8)