Pukcab User's Guide

Lyonel Vincent (lyonel@ezix.org)

March 2015

Contents

Chapter 1

Features

- lightweight (just 1 binary to be installed on both the client and the server)
- easy to install (only 1 username with SSH connectivity is required to set up a server)
- flexible configuration
- sensible defaults
- automatic retention schedules
- incremental/full backups
- data de-duplication
- data compression
- (optional) web interface

Chapter 2

Requirements

`pukcab` was written with UNIX-like operating systems in mind, it is therefore currently unsupported on Microsoft Windows[1].

`pukcab` has been tested on the following operating systems and platforms[2]:

- Linux 3.*x* on 64-bit Intel or AMD processors
- Linux 3.*x* on 32-bit Intel or AMD processors[3]
- Linux 3.*x* on 32-bit ARM processors
- Mac OS X on 64-bit Intel processors

Backup server

To run a `pukcab` backup server, you will need:

- SSH server
- dedicated user (recommended)
- disk space
- scalable filesystem
- enough memory or swap space (running without swap space is *not* recommended)

The "scalable filesystem" requirement might seem surprising but, to store backups, you will need a modern filesystem that can gracefully handle thousand and thousand of files per directory, sometimes big, sometimes small.

On Linux, XFS, ext4 and Btrfs are known to work.

[1] This might change in the future, though.
[2] The main development platform is Fedora Linux x86-64.
[3] Some old Pentium III machines may misbehave.

Clients

The requirements for a client are very limited. In short, nearly any Linux/OS X box will do.

- SSH client
- functional `tar` command (tested with GNU tar, should work with BSD (libarchive) tar and Jörg Schilling's star)
- `root` access (if you want to backup files other than yours)

Chapter 3

Installation

Just copy the `pukcab` binary[1] into your path (`/usr/bin/pukcab` will be just fine) on the backup server and each client.

OS	Platform		Packages
Linux	x86-64	64-bit	ZIP
Linux	i686	32-bit	ZIP
Linux	ARM	32-bit	ZIP
Mac OS X	x86-64	64-bit	ZIP
any	*any*	*any*	Source[2]

On the backup server

1. create a dedicated user – this user does not need specific privileges (i.e. do *NOT* use `root`)
2. allow key-based SSH login for that user (**mandatory**)
3. *optional*: allow password-based SSH login and set a password for the dedicated user (if you want to be able to register new clients using that password)

[1] Linux users should prefer RPM packages or check if their distribution already includes `pukcab`.

[2] To rebuild `pukcab`, you will need a Go development environment (and some courage).

On the clients (manual)

1. create SSH keys for the user which will launch the backup (most probably `root`)
2. add the user's public key to the dedicated user's `authorized_keys` on the backup server

On the clients (password registration)

1. create SSH keys for the user which will launch the backup (most probably `root`)
2. register to the backup server

Chapter 4

Configuration

`pukcab` is configured with a simple INI-like text file:

```
; comment
name1 = number
name2 = "text value"
name3 = [ "list", "of", "text", "values" ]
...
```

The default is to read `/etc/pukcab.conf` then `~/.pukcabrc` (which means that this user-defined file can override values set in the global configuration file).

Both client and server use the same configuration file format and location, only the values determine the client or server role (a client will have a `server` parameter set).

Notes

- text values must be enclosed in `"`
- lists of values are enclosed in `[` and `]` with comma-separated items

Server

The `pukcab` server contains the data files for all clients (in the `vault`) and a database of all backup sets (the `catalog`).

parameter	type	default	description
`user`	text	*none*	user name `pukcab` will run under (*mandatory*)
`vault`	text	`"vault"`	folder where all archive files will be created
`catalog`	text	`"catalog.db"`	name of the catalog database
`maxtries`	number	10	number of retries in case of concurrent client accesses

Table 4.1: server configuration

Notes

- `vault` and `catalog` paths can be absolute (starting with `/`) or relative to `user`'s home directory.
- the `vault` folder must be able to store many gigabytes of data, spread over thousands of files
- the `catalog` database may become big and must be located in a folder where `user` has write access
- the `vault` folder **must not be used to store anything**[1] else than `pukcab`'s data files; in particular, do **NOT** store the `catalog` there

Example

```
; all backups will be received and stored by the 'backup' user
user="backup"
vault="/var/local/backup/vault"
catalog="/var/local/backup/catalog.db"
```

Client

parameter	type	default	description
`user`	text	*none*	user name to use to connect (*mandatory*)
`server`	text	*none*	backup server (*mandatory*)
`port`	number	22	TCP port to use on the backup server
`include`	list	OS-dependent	filesystems / folders / files to include in the backup
`exclude`	list	OS-dependent	filesystems / folders / files to exclude from the backup

[1] `pukcab` will *silently* delete anything you may store there

parameter	type	default	description

Table 4.2: client configuration

Example

```
user="backup"
server="backupserver.localdomain.net"
```

OS-dependent defaults

`pukcab` tries to apply "sane" defaults, especially when taking a backup. In particular, it will only attempt to backup "real" filesystems and skip temporary files or pseudo-filesystems.

Under Linux, there are many exclusions caused by the extensive use of pseudo-filesystems.

parameter	default value
`include`	`[ "ext2", "ext3", "ext4", "btrfs", "xfs", "jfs", "vfat" ]`
`exclude`	`[ "/proc", "/sys", "/selinux", "tmpfs", "./.nobackup" ]`

Table 4.3: Linux defaults

For Mac OS X, the list of included filesystems is much shorter (it currently includes just the default filesystem, HFS).

parameter	default value
`include`	`[ "hfs" ]`
`exclude`	`[ "devfs", "autofs", "afpfs", "./.nobackup" ]`

Table 4.4: Mac OS X defaults

Chapter 5

Usage

Synopsis

`pukcab` *command* [*options* ...] [*files* ...]

backup, save	take a new backup
continue, resume	continue a partial backup
restore	restore files
verify, check	verify files in a backup
delete, purge	delete a backup
expire	apply retention schedule to old backups
vacuum	vault and catalog clean-up
config, cfg	display `pukcab`'s configuration
history, versions	list history for files
info, list	list backups and files
ping, test	check server connectivity
register	register to backup server
web	starts the built-in web interface

Table 5.1: available commands

backup

The `backup` command launches a new backup:

- creates a new backup set (and the corresponding date/id) on the backup server
- builds the list of files to be backed-up, based on the `include`/`exclude` configuration directives
- sends that list to the backup server
- computes the list of changes since the last backup (if `--full` isn't specified)
- sends the files to be includes in the backup
- closes the backup

Syntax `pukcab backup` [–full] [–name=*name*] [–schedule=*schedule*]

Notes

- the name and schedule options are chosen automatically if not specified
- interrupted backups can be resumed with the continue command

continue

The `continue` command continues a previously interrupted backup.

Syntax `pukcab continue` [–name=*name*] [–date=*date*]

Notes

- the name option is chosen automatically if not specified
- the date option automatically selects the last unfinished backup
- only unfinished backups may be resumed

restore

The `restore` command restores files as they were at a given date.

Syntax `pukcab restore` [–name=*name*] [–date=*date*] [*FILES* ...]

Notes

- the name option is chosen automatically if not specified
- the date option automatically selects the last backup
- this operation currently requires a working `tar` system command (usually GNU tar)

`verify`

The verify command reports files which have changed since a given date.

Syntax `pukcab verify` [–name=*name*] [–date=*date*] [*FILES* ...]

Notes

- the name option is chosen automatically if not specified
- the date option automatically selects the last backup if not specified

`delete`

The `delete` command discards the backup taken at a given date.

Syntax `pukcab delete` [–name=*name*] –date=*date*

Notes

- the name option is chosen automatically if not specified
- the date must be specified

`expire`

The `expire` command discards backups following a given schedule which are older than a given age (or date).

Standard retention schedules have pre-defined retention periods:

schedule	retention period
`daily`	2 weeks

schedule	retention period
`weekly`	6 weeks
`monthly`	365 days
`yearly`	10 years

Table 5.2: default retention schedules

Syntax `pukcab expire` [–name=*name*] [–schedule=*schedule*] [–age=*age*] [–keep=*keep*]

Notes

- on the backup server, the name option defaults to all backups if not specified
- on a backup client, the name option is chosen automatically if not specified
- the schedule and [expiration] are chosen automatically if not specified

`vacuum`

The `vacuum` command initiates clean-up of the catalog and vault to save disk space.

Syntax `pukcab vacuum`

Notes

- can only be run on the server
- the clean-up may take a while and delay new backups

`config`

The `config` command displays the current configuration.

Syntax `pukcab config`

history

The `history` command shows the different versions stored in backups for given files. Backup sets can be filtered by name and/or date and files.

Syntax `pukcab history` [–name=*name*] [–date=*date*] [*FILES* ...]

Notes

- if date is specified, the command lists only history after that date
- on server, if name is not specified, the command lists all backups, regardless of their name

info

The `info` command lists the backup sets stored on the server. Backup sets can be filtered by name and/or date and files.

Syntax `pukcab info` [–short] [–name=*name*] [–date=*date*] [*FILES* ...]

Notes

- if date is specified, the command lists only details about the corresponding backup
- on server, if name is not specified, the command lists all backups, regardless of their name
- verbose mode lists the individual files

ping

The `ping` command allows to check connectivity to the server.

Syntax `pukcab ping`

Notes

- verbose mode displays detailed information during the check

`register`

The `register` command registers a client's SSH public key to the server.

Syntax `pukcab register`

Notes

- to register to the backup server, `pukcab` will ask for the dedicated user's password (set on the server)
- verbose mode displays detailed information during the registration

`web`

The `web` command starts the built-in web interface.

Syntax `pukcab web` [–listen=[*host*]:*port*]

Notes

- by default, `pukcab` listens on `localhost` on port 8080
- available features depend on the local system's role (client or server)

Chapter 6

Options

`pukcab` is quite flexible with the way options are provided:

- options can be provided in any order
- options have both a long and a short (1-letter) name (for example, `--name` is `-n`)
- options can be prefixed with 1 or 2 minus signs (`--option` and `-option` are equivalent)
- `--option=value` and `--option value` are equivalent (caution: `=` is mandatory for boolean options)

This means that the following lines are all equivalent:

```
pukcab info -n test
pukcab info -n=test
pukcab info --n test
pukcab info --n=test
pukcab info -name test
pukcab info -name=test
pukcab info --name test
pukcab info --name=test
```

General options

The following options apply to all commands:

option	description
`-c`, `--config`[`=`]*file*	specify a configuration file to use
`-v`, `--verbose`[`=true`]	display more detailed information
`-h`, `--help`	display online help

`date`

Dates are an important concept for `pukcab`.

All backup sets are identified by a unique numeric id and correspond to a set of files at a given point in time (the backup id is actually a UNIX timestamp). The numeric id can be used to unambiguously specify a given backup set but other, more user-friendly formats are available:

- a duration in days (default when no unit is specified), hours, minutes is interpreted as relative (in the past, regardless of the actual sign of the duration you specify) to the current time
- a human-readable date specification in YYYY-MM-DD format is interpreted as absolute (00:00:00 on that date)
- `now` or `latest` are interpreted as the current time
- `today` is interpreted as the beginning of the day (local time)

Syntax `--date`[=]*date*

`-d` *date*

Examples

- `--date 1422577319` means *on the 30th January 2015 at 01:21:59 CET*
- `--date 0`, `--date now` and `--date latest` mean *now*
- `--date today` means *today at 00:00*
- `--date 1` means *yesterday same time*
- `--date 7` means *last week*
- `--date -48h` and `--date 48h` both mean *2 days ago*
- `--date 2h30m` means *2 hours and 30 minutes ago*
- `--date 2015-01-07` means *on the 7th January 2015 at midnight*

name

In `pukcab`, a name is associated with each backup when it's created. It is a free-form text string.

Syntax `--name`[=]*name*

`-n` *name*

Default value current host name (output of the `hostname` command)

schedule

In `pukcab`, a retention schedule is associated with each backup when it's created and is used when expiring old backups. It is a free-form text string but common values include `daily`, `weekly`, `monthly`, etc.

Syntax `--schedule`[=]*schedule*

`-r` *schedule*

Default value (the default value depends on the current day)

`daily` from Monday to Saturday

`weekly` on Sunday

`monthly` on the 1st of the month

`yearly` on 1st January

full

Forces a full backup: `pukcab` will send all files to the server, without checking for changes.

Syntax `--full`[`=true`]

`--full=false`

`-f`

Default value `false`

keep

When expiring data, keep at least a certain number of backups (even if they are expired).

Syntax `--keep`[=]*number*

`-k` *number*

Default value `3`

short

Display a more concise output.

Syntax `--short`[`=true`]

`--short=false`

`-s`

Default value `false`

listen

Force the built-in web server to listen for connections on a different address/port.

Syntax `--listen`[=][*host*]:*port*

`-l` [*host*]:*port*

Default value `localhost:8080`

Files

File names can be specified using the usual shell-like wildcards `*` (matches any number of characters) and `?` (matches exactly one character). The following conventions apply:

- file names starting with a slash ('`/`') are absolute
- file names not starting with a slash ('`/`') are relative
- specifying a directory name also selects all the files underneath

Examples

- `/home` includes `/home/john`, `/home/dave`, etc. and all the files they contain (i.e. all users' home directories)
- `*.jpg` includes all `.jpg` files (JPEG images) in all directories
- `/etc/yum.repos.d/*.repo` includes all repositories configured in Yum
- `/lib` includes `/lib` and all the files underneath but not `/usr/lib`, `/var/lib`, etc.

Chapter 7

Examples

Launch a new backup - default options

```
[root@myserver ~]# pukcab backup --verbose
Starting backup: name="myserver" schedule="daily"
Sending file list... done.
New backup: date=1422549975 files=309733
Previous backup: date=1422505656
Determining files to backup... done.
Incremental backup: date=1422549975 files=35
Sending files... done
[root@myserver ~]#
```