restic: Backups done right

Brian DeLacey, BLU @ MIT
February 15, 2017

About Brian ...

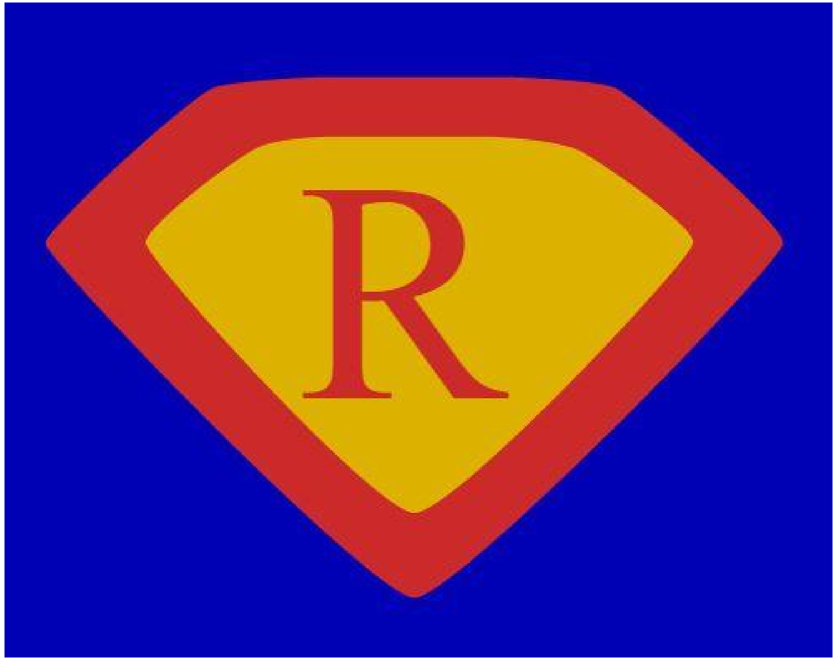
Brian worked for a number of years at Lotus Development and IBM in different roles within the development community. He's also worked in the research area at Harvard Business School.

Brian was an early adopter of the Mac, past Executive Director of the Macintosh User Group, and long-time member of the Boston Computer Society.

Brian has worked on mainframes, minis, micros, PCs, and IoT Devices in a variety of languages, across a number of operating systems.

Brian's recent activities involve Go and network-ware.

Welcome to restic



What do users want?

Users don't want to do backups.

Users want to do restores!

restic Design Principles

Easy

Fast

Verifiable

Secure

Efficient

Free

Security Threat Model

The design goals for restic include being able to securely store backups in a location that is not completely trusted, e.g. a shared system where others can potentially access the files or (in the case of the system administrator) even modify or delete them.

General assumptions: The host system a backup is created on is trusted. This is the most basic requirement, and essential for creating trustworthy backups.

Source: https://restic.readthedocs.io/en/stable/Design/

restic Highlights - "Backups done right!"

Restic is licensed under "BSD 2-Clause License".

Open source written in Go

Latest Version as of this talk is v0.4.0 (2017-02-02)

- Binaries available and also Debian Package

Smart - deduplication, backward compatibility of all repositories within one major version

Lead Author / Creator / Developer

- Alexander Neumann, https://github.com/fd0
- Aachen, Germany

Go Sidebar...

- Aka golang, is free and open source language created by Google
- Development began in 2007 Robert Griesemer, Rob Pike, Ken Thompson
- First appeared in the wild in November 10, 2009
- Runs mostly everywhere - aims to be productive, readable
- Compiled (fast), concurrent, imperative, structured, scalable
- Discipline is strong, typed, statically typed, inferred, structural
- Built-in garbage collection, with support for networking, multiprocessing
- Interface system (rather than virtual inheritance)
- Remote package management
- Language Tools (go build, go test, go fmt, go get, go run, godoc etc.)

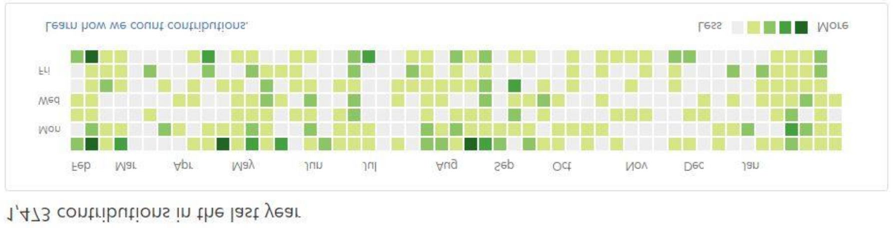
Go Versions... *sudo apt-get install golang-go*

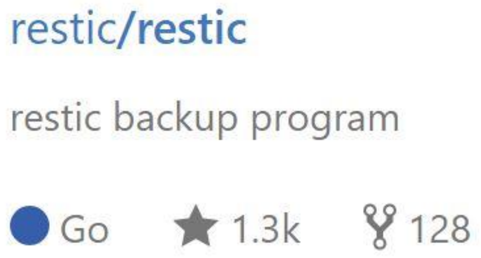
Go 1.8 (Expected early 2017)
Go 1.7 (August 2016)
Go 1.6 (February 2016)
Go 1.5 (August 2015)
Go 1.4 (December 2014)
Go 1.3 (June 2014)
Go 1.2 (December 2013)
Go 1.1 (May 2013)
Go 1 (March 2012)

https://golang.org/project/ and you can download here https://golang.org/dl/

restic Authorship

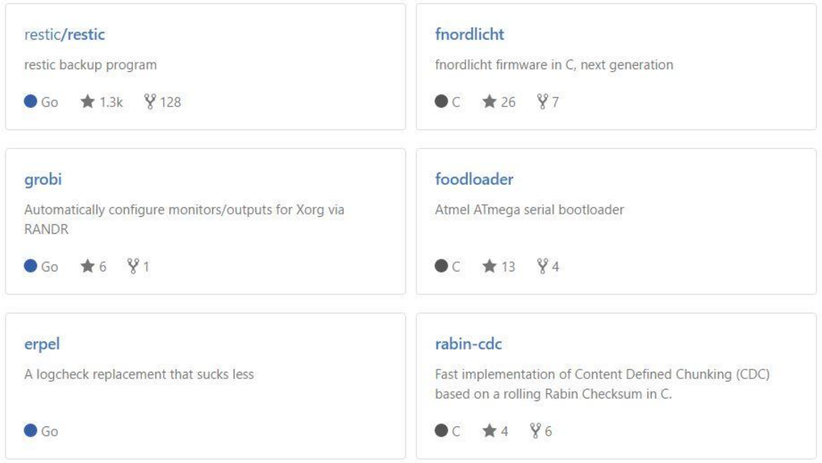
Alexander Neumann, https://github.com/fd0, Germany




restic/restic

restic backup program

● Go ★ 1.3k 128


restic/restic

restic backup program

● Go ★ 1.3k 128

fnordlicht

fnordlicht firmware in C, next generation

● C ★ 26 7

grobi

Automatically configure monitors/outputs for Xorg via RANDR

● Go ★ 6 1

foodloader

Atmel ATmega serial bootloader

● C ★ 13 4

erpel

A logcheck replacement that sucks less

● Go

rabin-cdc

Fast implementation of Content Defined Chunking (CDC) based on a rolling Rabin Checksum in C.

● C ★ 4 6

restic Lightning Talk

2015-02-01: Lightning Talk at FOSDEM 2015: A short introduction (with slightly outdated command line)

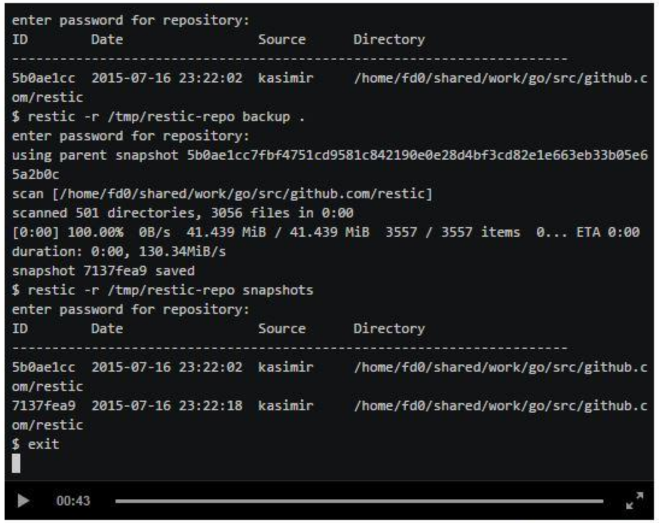
https://youtu.be/oM-MfeflUZ8?t=11m49s

Go Lightning Talks
The Go community on Go
Andrew Gerrand

restic
Backup Done Right
1 February 2015
Alexander Neumann (fd0)

Quickstart

A short recorded demo of restic



```
enter password for repository:
ID        Date                 Source      Directory
----------------------------------------------------------------------
5b0ae1cc  2015-07-16 23:22:02  kasimir     /home/fd0/shared/work/go/src/github.c
om/restic
$ restic -r /tmp/restic-repo backup .
enter password for repository:
using parent snapshot 5b0ae1cc7fbf4751cd9581c842190e0e28d4bf3cd82e1e663eb33b05e6
5a2b0c
scan [/home/fd0/shared/work/go/src/github.com/restic]
scanned 501 directories, 3056 files in 0:00
[0:00] 100.00%  0B/s  41.439 MiB / 41.439 MiB  3557 / 3557 items  0... ETA 0:00
duration: 0:00, 130.34MiB/s
snapshot 7137fea9 saved
$ restic -r /tmp/restic-repo snapshots
enter password for repository:
ID        Date                 Source      Directory
----------------------------------------------------------------------
5b0ae1cc  2015-07-16 23:22:02  kasimir     /home/fd0/shared/work/go/src/github.c
om/restic
7137fea9  2015-07-16 23:22:18  kasimir     /home/fd0/shared/work/go/src/github.c
om/restic
$ exit
```

Powered by asciinema

```
$restic -r /tmp/restic-repo init
enter password for new backend:
enter password again:
Created restic backend NNNNNNNN at /tmp/restic-repo
```

Please note that knowledge of your password is required …
Losing your password means that your data is irrecoverably lost.

```
$ls -al
$restic -r /tmp/restic-repo backup .
Enter password for repostiory:
```

Source: https://restic.github.io/

Flow

1. Init repository
2. Backup files / directories / lists (with optional tags)
3. Snapshot history
4. Restore
5. Forget (policy qualifications) then periodically Prune
6. Check (for integrity)

Common Commands - For Backup and Restore

```
$ ./restic --help

$ ./restic init --repo /tmp/backup (Initialize a repository)

$ ./restic -r /tmp/backup backup ~/work (create  snapshot at point in time - ie. backup data)

$ ./restic -r /tmp/backup backup ~/work.txt (backup individual file)

$./restic -r /tmp/backup snapshots (list all snapshots)

$ ./restic -r /tmp/backup snapshots --path="/srv" (filter by path name)

$ restic -r /tmp/backup restore 79766175 --target ~/tmp/restore-work

$ restic -r /tmp/backup check (check integrity and consistency)
```

Common Commands - Forgetting and Pruning

```
$ restic -r /tmp/backup snapshots
enter password for repository:
ID        Date                 Host      Tags  Directory
----------------------------------------------------------------------
40dc1520  2015-05-08 21:38:30  kasimir         /home/user/work
79766175  2015-05-08 21:40:19  kasimir         /home/user/work
bdbd3439  2015-05-08 21:45:17  luigi           /home/art
```

```
$ restic -r /tmp/backup forget bdbd3439 (removes references to data)
```

```
$ restic -r /tmp/backup prune (required to cleanup unreferenced data)
```

Fancy Command - Exclude

Exclude folders / files by specifying exclude patterns

```
$ cat exclude
# exclude go-files
*.go
# exclude foo/x/y/z/bar foo/x/bar foo/bar
foo/**/bar
$ restic -r /tmp/backup backup ~/work --exclude=*.c --exclude-file=exclude
```

Source: https://restic.readthedocs.io/en/latest/Manual/

Fancy Command - Filter and choose from File

backup files that have a certain filename in them

```
$ find /tmp/somefiles | grep 'PATTERN' > /tmp/files_to_backup
```

use restic to backup the filtered files

```
$ restic -r /tmp/backup backup --files-from /tmp/files_to_backup
```

combine --files-from with the normal files args

```
$restic -r /tmp/backup backup --files-from /tmp/files_to_backup /tmp/some_additional_file
```

Fancy Command - Tags

Snapshots can have one or more tags, short strings which add identifying information. Just specify the tags for a snapshot with --tag

```
$ restic -r /tmp/backup backup --tag projectX ~/shared/work/web
[...]
```

Advanced Commands (we won't cover them all...)

Mount a repository and browse backup as a regular file system (via FUSE)

Create an SFTP repository (set up SSH server, use the URL scheme with init)

- $ restic -r sftp:user@host:/tmp/backup init

Create a REST server repository (backup via HTTP or HTTPS)

Create an Amazon S3 repository

Create a Minio Server repository

$ restic -r /tmp/backup list snapshots (blobs, packs, index, snapshots, keys or locks)

Removing snapshots according to a policy (This we do cover … in the next slide)

How to run backups on a schedule?

You can use a cron job but the question becomes how to pass the password?

The environment variable RESTIC_PASSWORD allows you to set the password for the automated backup process. This is documented in the manual here http://restic.readthedocs.io/en/latest/Manual/#initialize-a-repository

Since environment variables may be shown in the ps outputs, which may sometime sbe written to log files for diagnostic reasons, having the password in a env variable isn't always the best. So they added a feature to store the password in a file https://github.com/restic/restic/issues/278 and also here https://github.com/restic/restic/pull/613

Policy

command-line parameter --dry-run (print which snapshots would be removed)

The forget command accepts the following parameters:

--keep-last n (never delete the n last - most recent - snapshots)
--keep-hourly n (for last n hours a snapshot was made, keep only the last snapshot for each hour)
--keep-daily n (for last n days which have one or more snapshots, keep last one for that day)
--keep-weekly n (for last n weeks, keep the last one for that week)
--keep-monthly n (for last n months which have one or more snapshots, keep the last one for month)
--keep-yearly n (for last n years which have one or more snapshots, keep the last one for year)
--keep-tag (keep snapshots with all tags specified by this option-can be specified multiple times)

Version History

v0.4.0 (2017-02-02) Begins shipping pre-compiled binaries

- Restic Release Page: https://github.com/restic/restic/releases/tag/v0.4.0

v0.3.3 (2017-01-08) Maintenance

v0.3.2 (2016-12-18)

v0.3.1 (2016-12-13)

v0.3.0 (2016-10-02)

v0.2.0 (2016-07-30)

v0.1.0 (2015-08-21) First public release of restic (1122 commits to master since this first public release)

Debian Package: https://packages.debian.org/sid/main/restic

Building restic

“restic is written in the Go programming language and you need at least Go version 1.6. Building restic may also work with older versions of Go, but that's not supported. See the Getting started guide of the Go project for instructions how to install Go.”

In order to build restic from source ...

Build restic yourself

```
$ git clone https://github.com/restic/restic
[...]

$ cd restic

$ go run build.go
```

Design Principle - Easy

Easy: Doing backups should be a frictionless process, otherwise you might be tempted to skip it. Restic should be easy to configure and use, so that, in the event of a data loss, you can just restore it. Likewise, restoring data should not be complicated.

https://github.com/restic/restic/blob/master/doc/index.md

Design Principle - Fast

Fast: Backing up your data with restic should only be limited by your network or hard disk bandwidth so that you can backup your files every day. Nobody does backups if it takes too much time. Restoring backups should only transfer data that is needed for the files that are to be restored, so that this process is also fast.

https://github.com/restic/restic/blob/master/doc/index.md

Design Principle - Verifiable

Verifiable: Much more important than backup is restore, so restic enables you to easily verify that all data can be restored.

https://github.com/restic/restic/blob/master/doc/index.md

Design Principle - Secure

Secure: restic uses cryptography to guarantee confidentiality and integrity of your data. The location the backup data is stored is assumed not to be a trusted environment (e.g. a shared space where others like system administrators are able to access your backups). Restic is built to secure your data against such attackers.

https://github.com/restic/restic/blob/master/doc/index.md

Design Principle - Efficient

Efficient: With the growth of data, additional snapshots should only take the storage of the actual increment. Even more, duplicate data should be de-duplicated before it is actually written to the storage back end to save precious backup space.

https://github.com/restic/restic/blob/master/doc/index.md

Design Principle - Free

Free: restic is free software and licensed under the BSD 2-Clause License and actively developed on GitHub.

https://restic.github.io/

Compatibility?

Backward compatibility for backups is important so that our users are always able to restore saved data. Therefore restic follows Semantic Versioning to clearly define which versions are compatible. The repository and data structures contained therein are considered the "Public API" in the sense of Semantic Versioning. This goes for all released versions of restic, this may not be the case for the master branch.

We guarantee backward compatibility of all repositories within one major version; as long as we do not increment the major version, data can be read and restored. We strive to be fully backward compatible to all prior versions.

https://github.com/restic/restic/blob/master/doc/index.md

restic Trivia

Joe Restic was the longest-serving football coach at Harvard

His idea? Restic's 2011 obituary in the NY Times

"create doubt in the best athletes," as Restic once put it.

References

1. restic Introduction - https://restic.github.io/
2. Restic design document - https://restic.readthedocs.io/en/stable/Design/
3. GitHub Source
    - https://github.com/restic/restic
4. Manual - overview of the basic functionality provided by restic
    - https://restic.readthedocs.io/en/latest/Manual/
5. Documentation for restic
    - https://restic.readthedocs.io/en/latest/
6. GoDocs - https://godoc.org/github.com/howeyc/restic
7. Lightning Talk - Why another backup program?

Q & A

Thank you!

Questions?