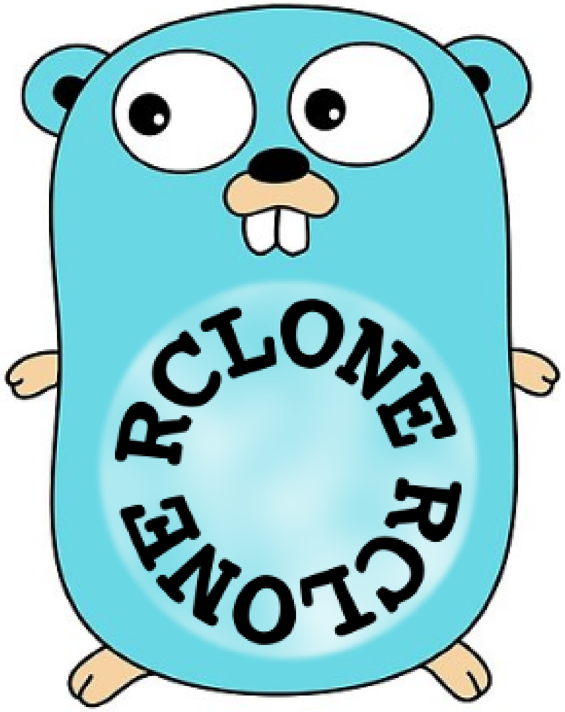
Go London User Group - 21st November 2018

- **Rclone “rsync for cloud storage”**
  - https://rclone.org
  - https://github.com/ncw/rclone
- **Talk by**
  - Nick Craig-Wood
  - Twitter: @njcw
  - Email: nick@craig-wood.com



About me

- **Nick Craig-Wood**
  - CTO of Memset Ltd by day
  - Open Source coder by night
  - Keen interest in storage, data integrity
  - Reformed data hoarder (ha!)

Contents

Rclone - “rsync for cloud storage”

- **Rclone is a command line program to sync files and directories to and from cloud providers**
- **MD5/SHA1 hashes checked at all times for file integrity**
- **Timestamps preserved on files**
- **Copy mode to just copy new/changed files**
- **Sync (one way) mode to make a directory identical**
- **Check mode to check for file hash equality**
- **Can sync to and from network, eg two different cloud accounts**
- **Encryption backend**
- **Cache backend**
- **Optional FUSE mount (rclone mount)**

Rclone vs Rsync

From Wikipedia

- rsync is a utility for efficiently transferring and synchronizing files across computer systems, by checking the timestamp and size of files. ✓
- It is commonly found on Unix-like systems and functions as both a file synchronization and file transfer program. ✓
- The rsync algorithm is a type of delta encoding, and is used for minimizing network usage. ✗

Cloud providers supported by rclone

- Amazon Drive
- Amazon S3
- Backblaze B2
- Box
- Ceph
- DigitalOcean Spaces
- Dreamhost
- Dropbox
- FTP
- Google Cloud Storage
- Google Drive
- HTTP
- Hubic
- Jottacloud
- IBM COS S3
- Memset Memstore
- Mega

- Microsoft Azure Blob Storage
- Microsoft OneDrive
- Minio
- Nextcloud
- OVH
- OpenDrive
- Openstack Swift
- Oracle Cloud Storage
- ownCloud
- pCloud
- put.io
- QingStor
- Rackspace Cloud Files
- SFTP
- Wasabi
- WebDAV
- Yandex Disk
- The local filesystem

Rclone platforms

Plan 9 from Bell Labs

Microsoft | Windows

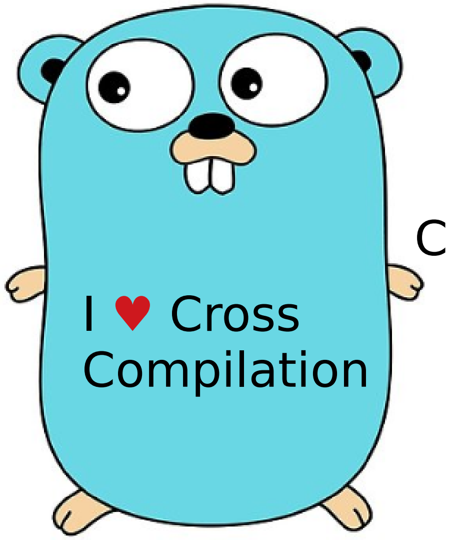
FreeBSD

NetBSD®

OpenBSD

solaris™

OS



CPU

AMD 64™

ARM®

MIPS TECHNOLOGIES

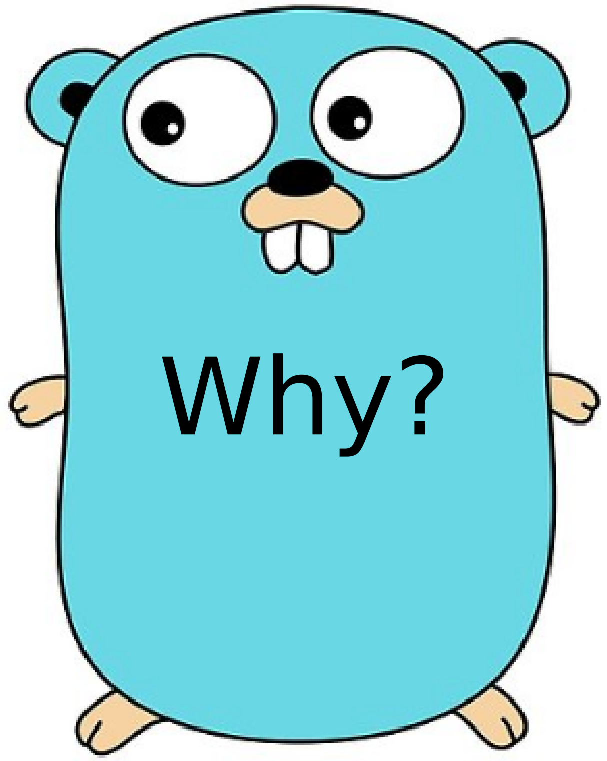
AARCH 64

How rclone came to be

- **Started as a tool to exercise**
  - github.com/ncw/swift
  - originally was “swiftsync”
- **First version in 2012**
  - Go 1.0
  - 3 backends
- **Somewhat outgrew its original design!**

Why Go?

- **Single binary deploy**
- **Excellent concurrency**
- **Great cross platform**
- **Fast!**
- **Standard library**
- **New challenge for me**
- **Easy for contributors to pick up**

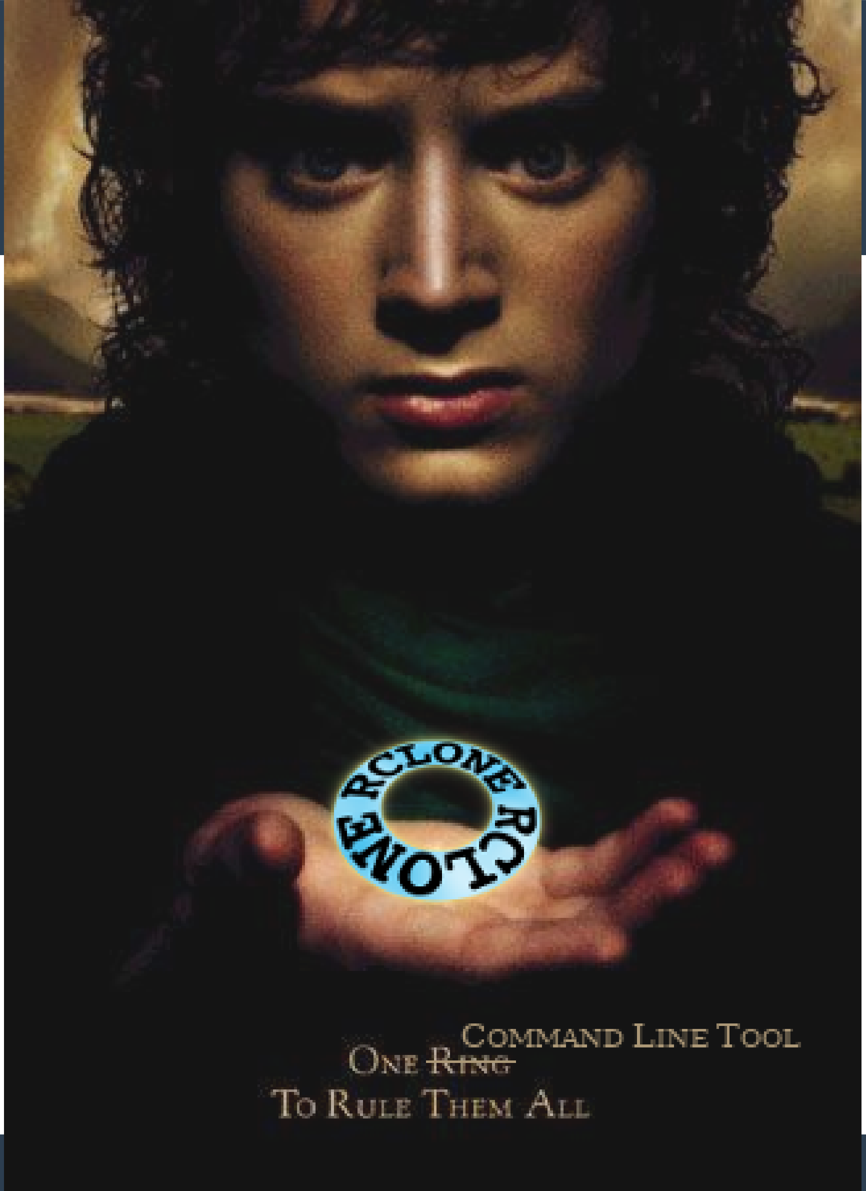


One tool to rule them all

What started as a tiny exercise

- 11,000 stars on Github
- 200 contributors
- 500 pull requests
- 1,500 issues
- 250,000 downloads a month
- Packaged in Ubuntu, Arch, Debian, Homebrew, Chocolatey and more

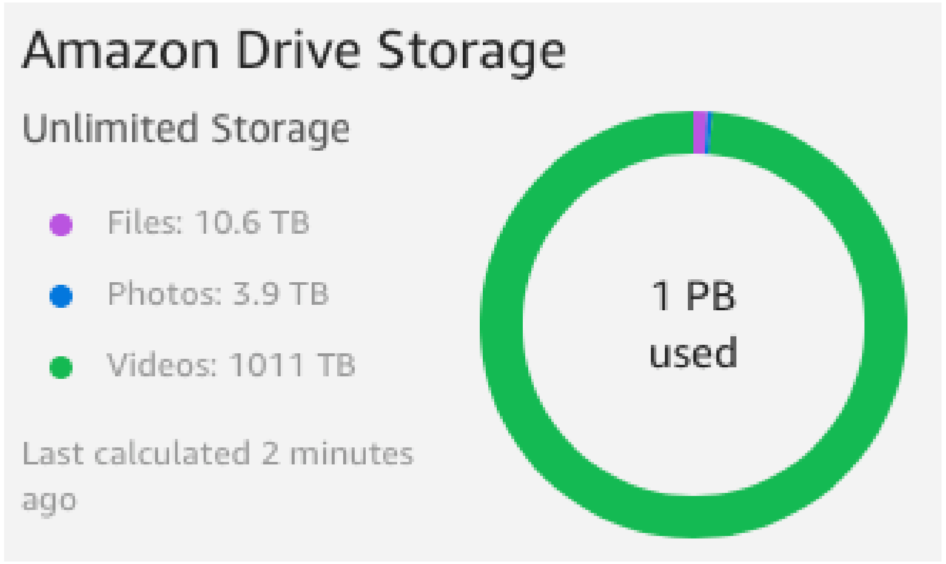
...is now an enormous project.



Visualising Rclone’s History

2012-11-18
Initial commit - some small parts working
Visualising Rclone's Code

Rclone becomes popular and breaks Amazon Cloud Drive


Amazon Drive Storage

Unlimited Storage

● Files: 10.6 TB

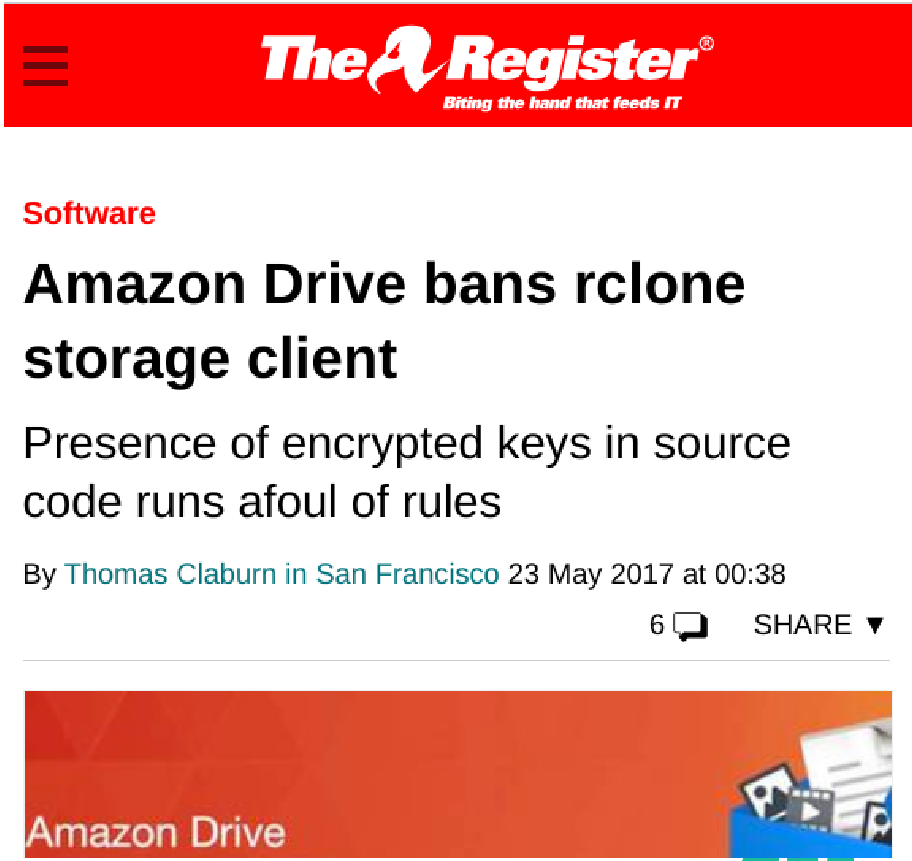
● Photos: 3.9 TB

● Videos: 1011 TB

Last calculated 2 minutes ago

⇒

?



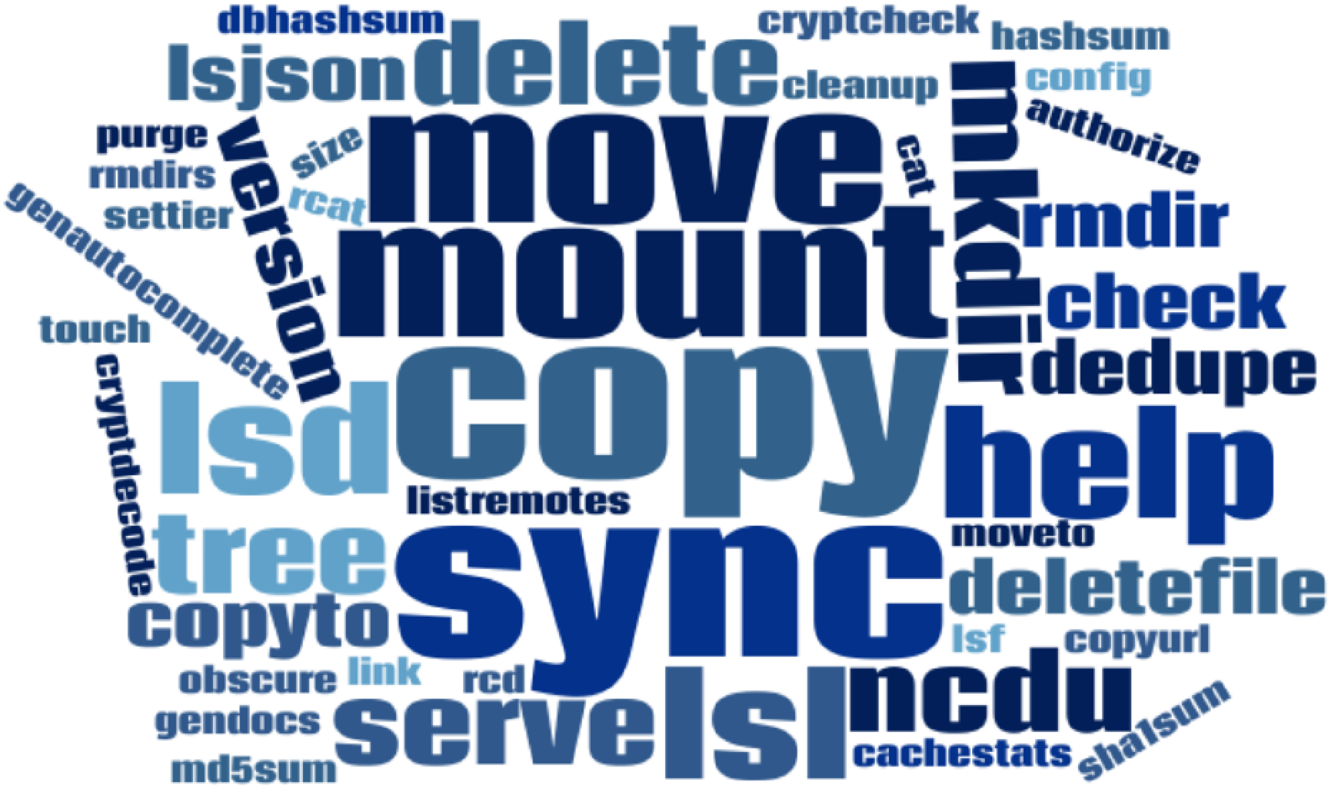
Software

Amazon Drive bans rclone storage client

Presence of encrypted keys in source code runs afoul of rules

By Thomas Claburn in San Francisco 23 May 2017 at 00:38

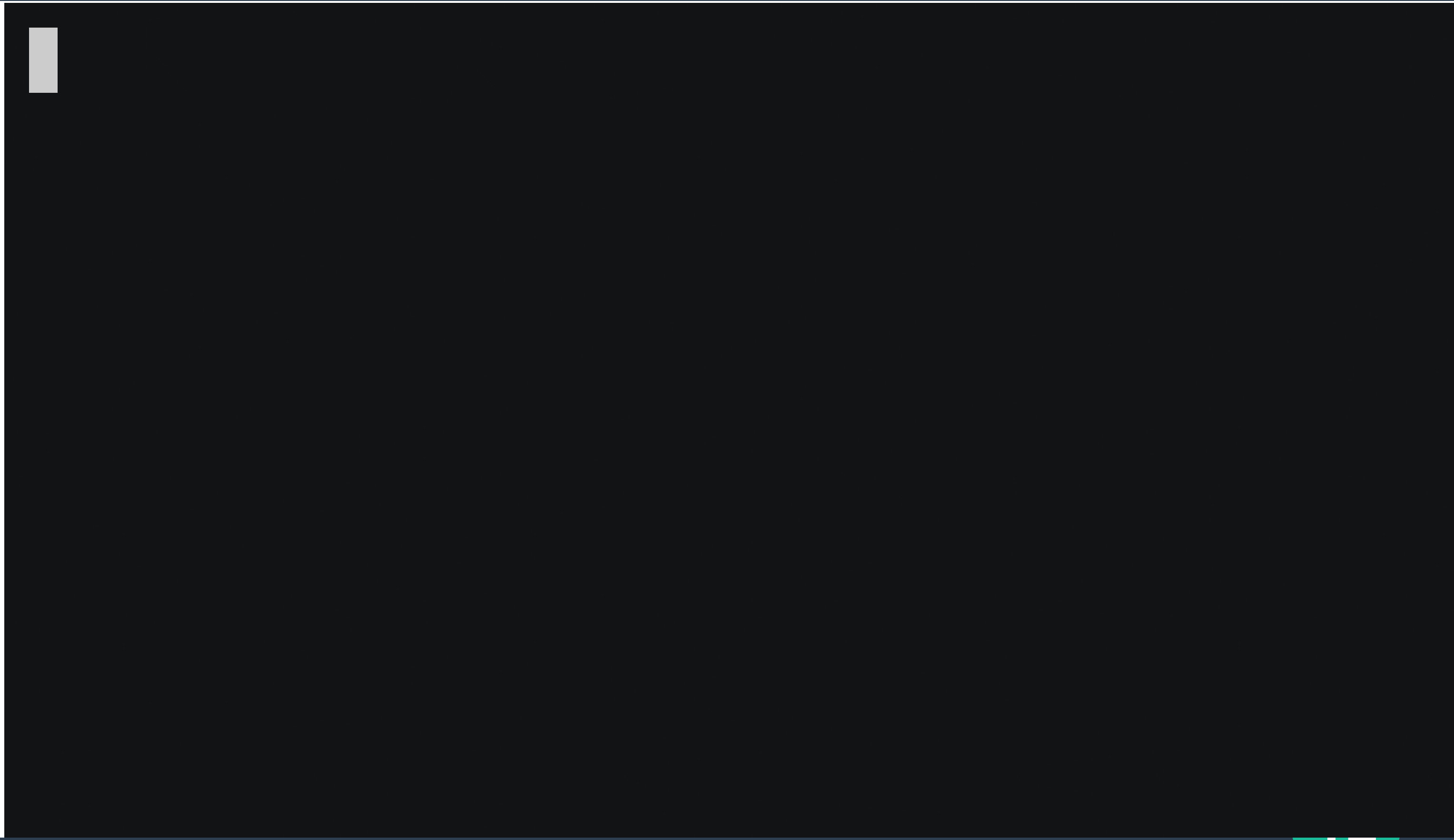
Rclone verbs – bigger = more popular



rclone config - Config Wizard

Old School Config Wizard

- Text based
- Easy to use
- Not pretty
- Calls your browser to do oauth



rclone copy - demo

rclone copy

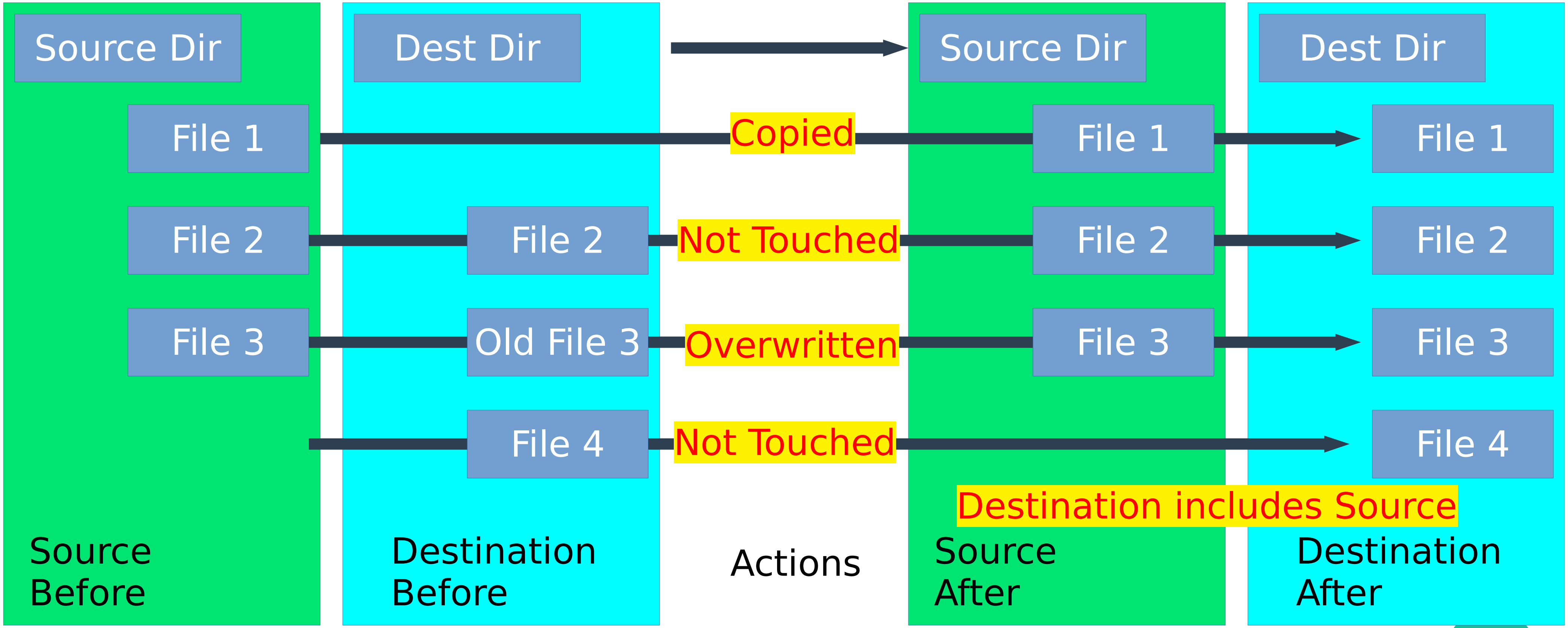
- Copy new files to destination
- Don’t delete files from destination
- Your go to rclone command!

rclone sync - demo

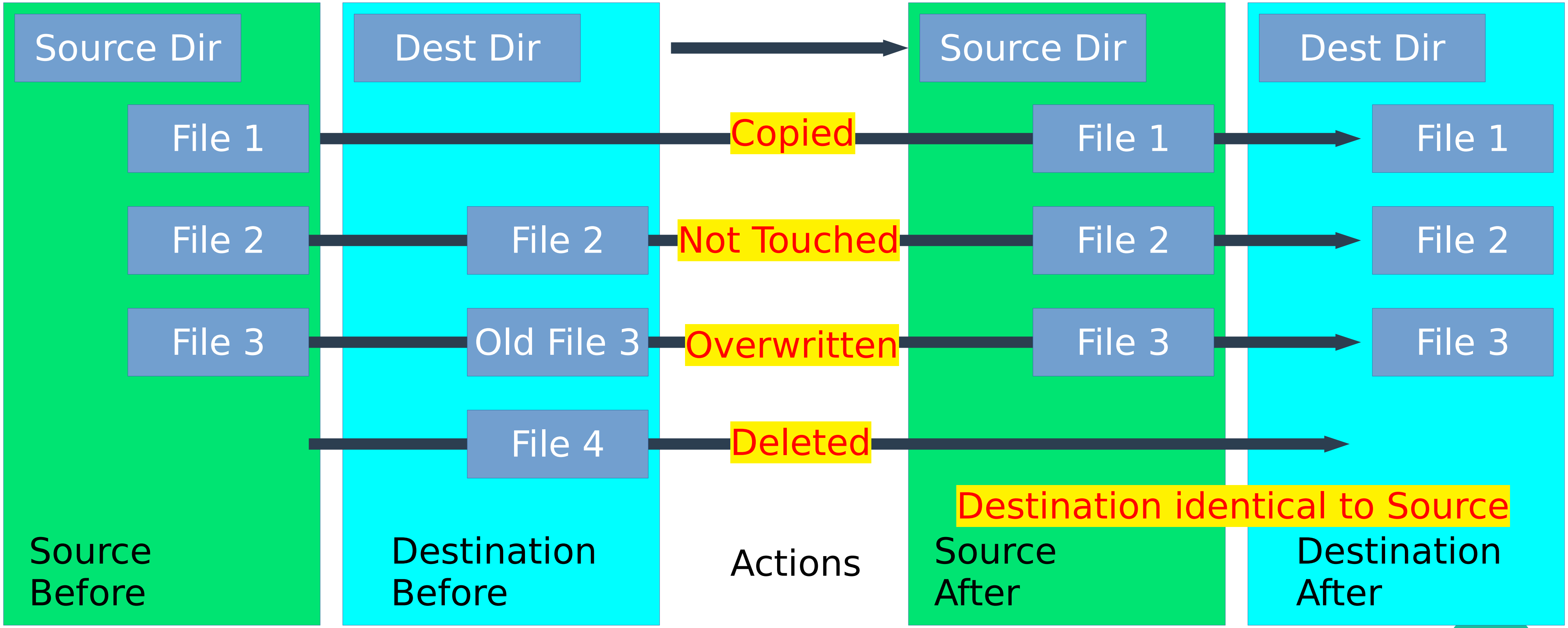
rclone sync

- Copy new files to destination
- Delete destination files not in source
- Use with –dry-run first recommended

rclone copy “Source Dir” “Dest Dir”

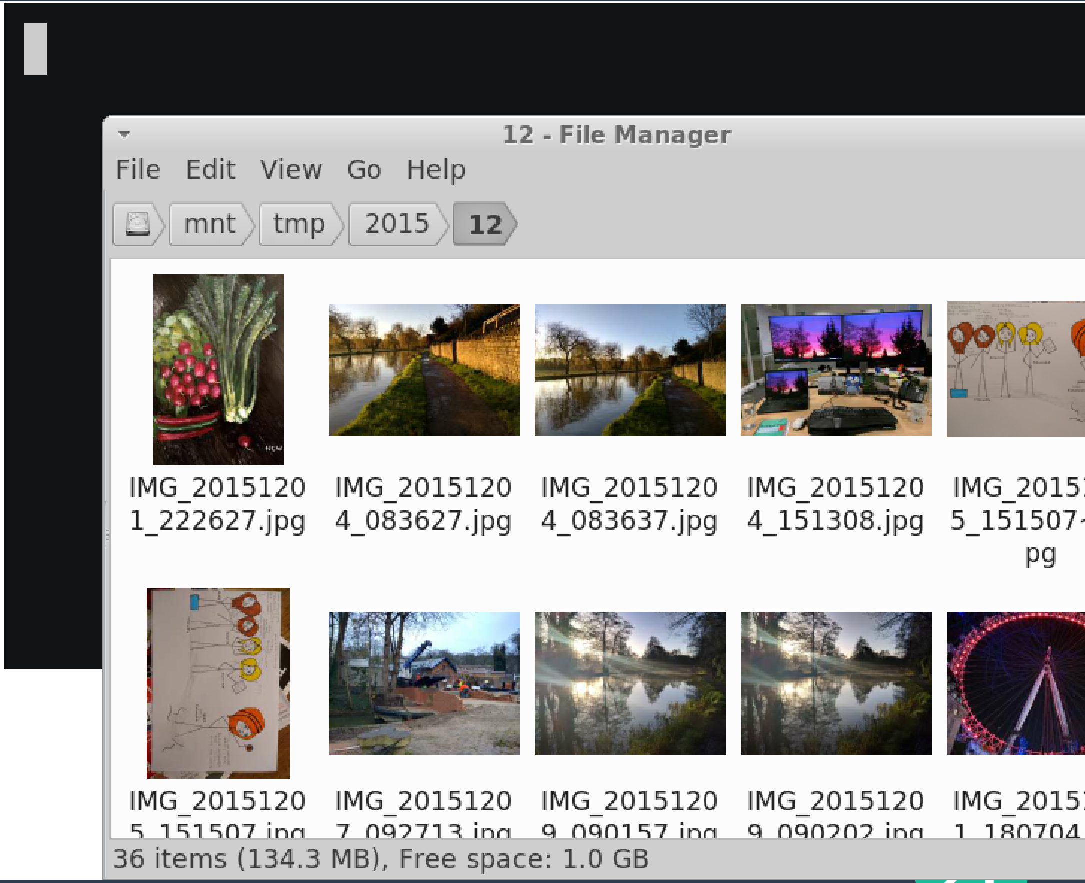


rclone sync “Source Dir” “Dest Dir”



rclone **mount** remote:path /mount/point

- **FUSE Filesystem**
  - Linux, macOS, FreeBSD
  - Windows va WinFSP
- **Optional caching layer**
  - Needed as can't write to middle of object
  - Or read and write together
- **Can run as daemon**



rclone ncdu

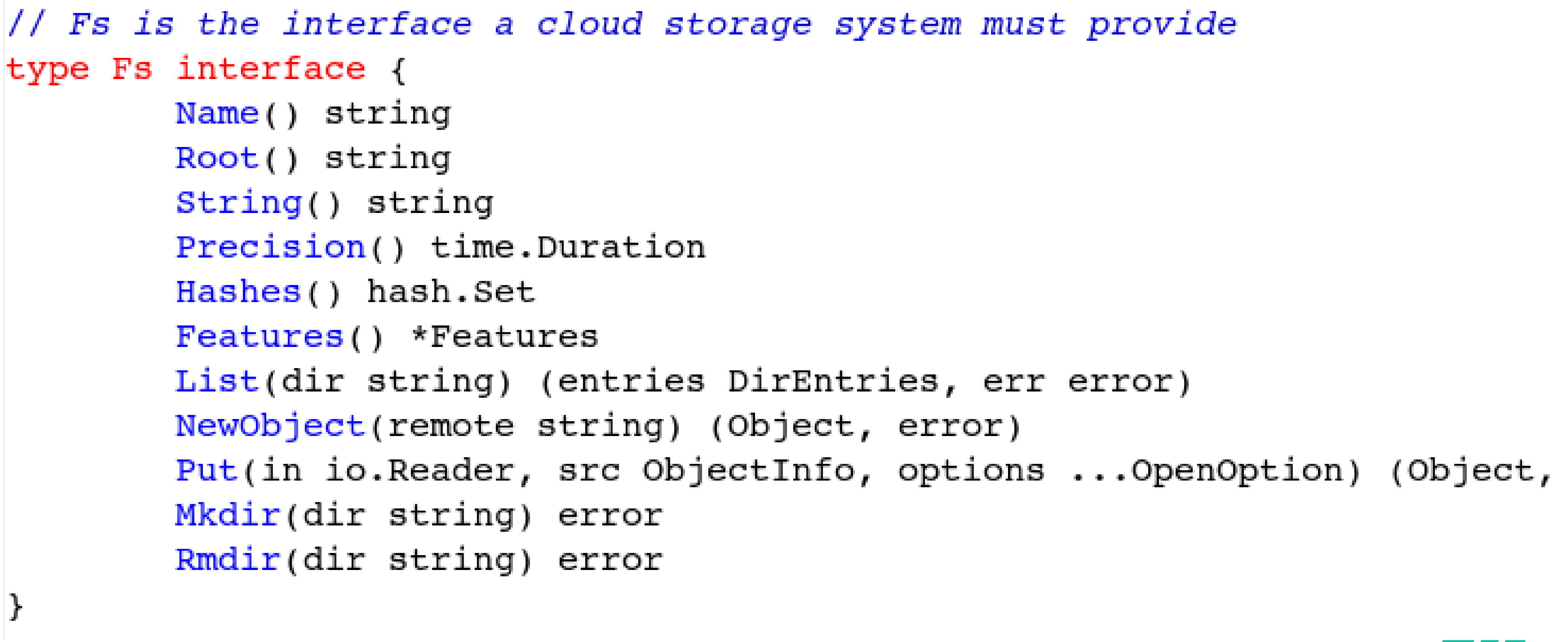
```
ncw@dogger:~$ █
```

This displays a text based user interface allowing the navigation of a Remote.

It is most useful for answering the question:

What is using all my disk space?

Backend interface

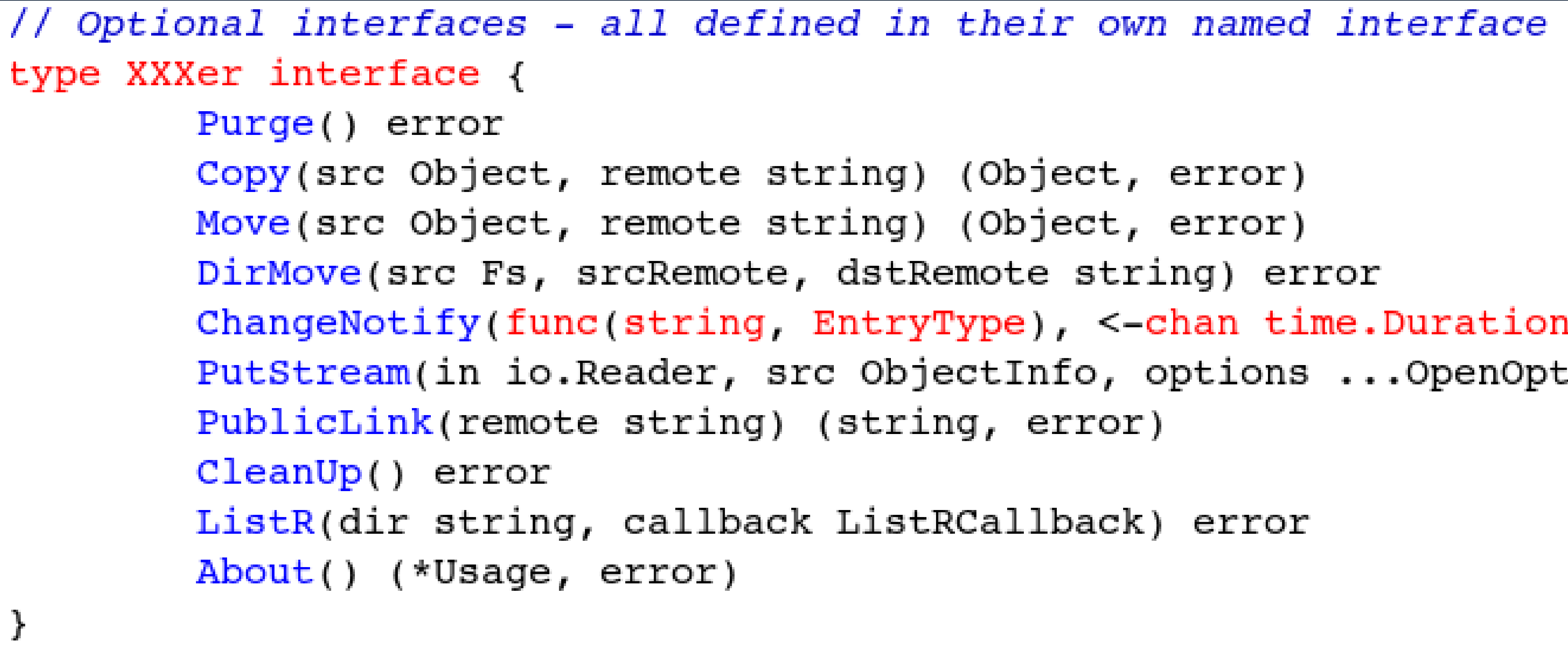


```
// Fs is the interface a cloud storage system must provide
type Fs interface {
        Name() string
        Root() string
        String() string
        Precision() time.Duration
        Hashes() hash.Set
        Features() *Features
        List(dir string) (entries DirEntries, err error)
        NewObject(remote string) (Object, error)
        Put(in io.Reader, src ObjectInfo, options ...OpenOption) (Object,
        Mkdir(dir string) error
        Rmdir(dir string) error
}
```

Object interface

```
// Object is a filesystem like object provided by an Fs
type Object interface {
	String() string
	Remote() string
	ModTime() time.Time
	Size() int64
	Fs() Info
	Hash(hash.Type) (string, error)
	Storable() bool
	SetModTime(time.Time) error
	Open(options ...OpenOption) (io.ReadCloser, error)
	Update(in io.Reader, src ObjectInfo, options ...OpenOption) error
	Remove() error
}
```

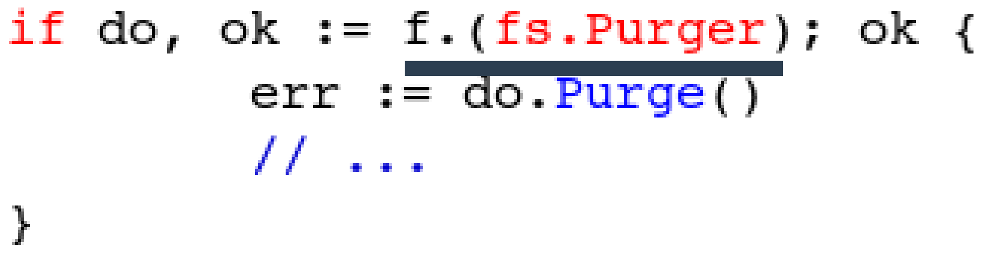
Optional interfaces for Fs



```go
// Optional interfaces - all defined in their own named interface
type XXXer interface {
        Purge() error
        Copy(src Object, remote string) (Object, error)
        Move(src Object, remote string) (Object, error)
        DirMove(src Fs, srcRemote, dstRemote string) error
        ChangeNotify(func(string, EntryType), <-chan time.Duration
        PutStream(in io.Reader, src ObjectInfo, options ...OpenOpt
        PublicLink(remote string) (string, error)
        CleanUp() error
        ListR(dir string, callback ListRCallback) error
        About() (*Usage, error)
}
```

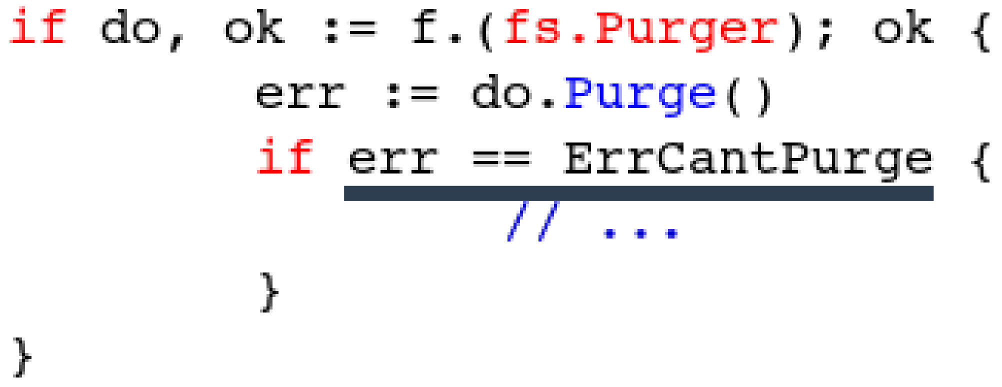
Using an optional interface

- Do a type assertion for the interface to see if it exists.



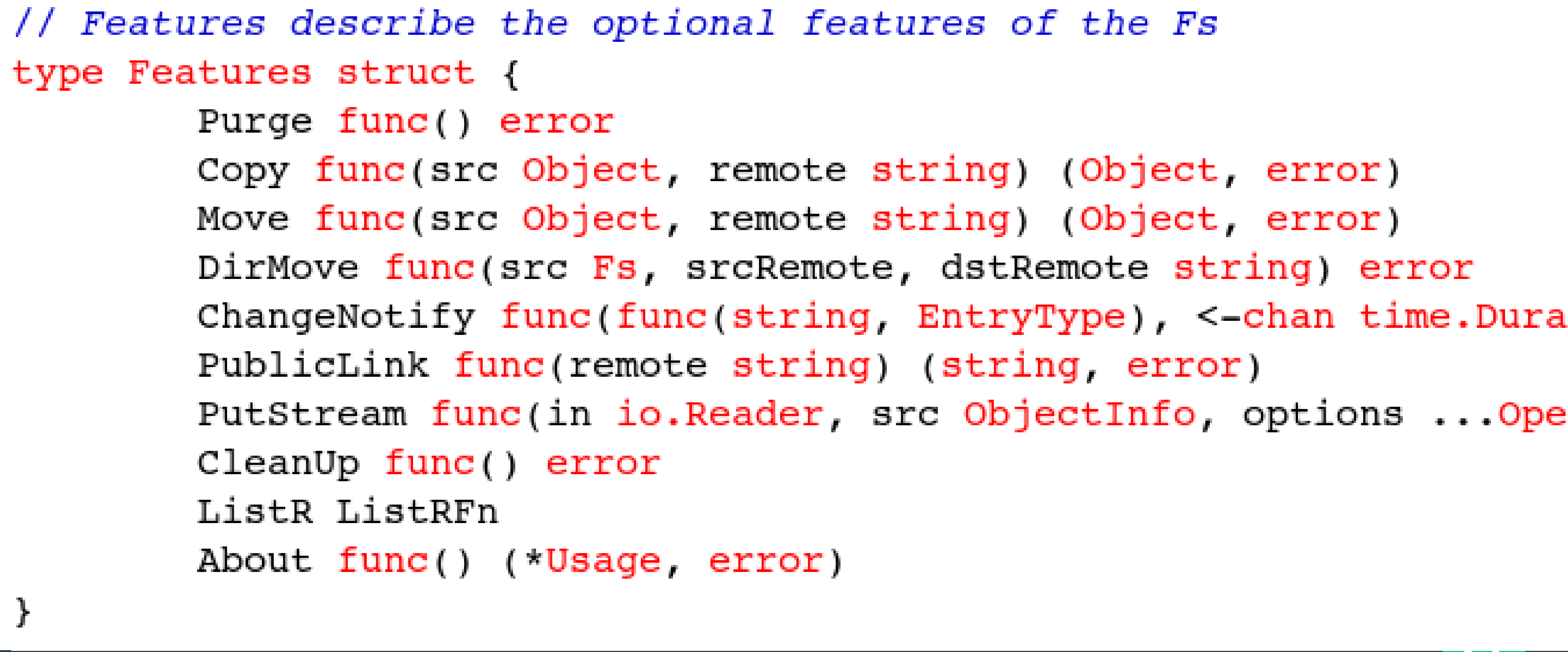
```
if do, ok := f.(fs.Purger); ok {
        err := do.Purge()
        // ...
}
```

- But what if this is a wrapper backend wrapping a backend that doesn’t support Purge?
- And if we need to know in advance?...



```
if do, ok := f.(fs.Purger); ok {
        err := do.Purge()
        if err == ErrCantPurge {
                // ...
        }
}
```

The solution



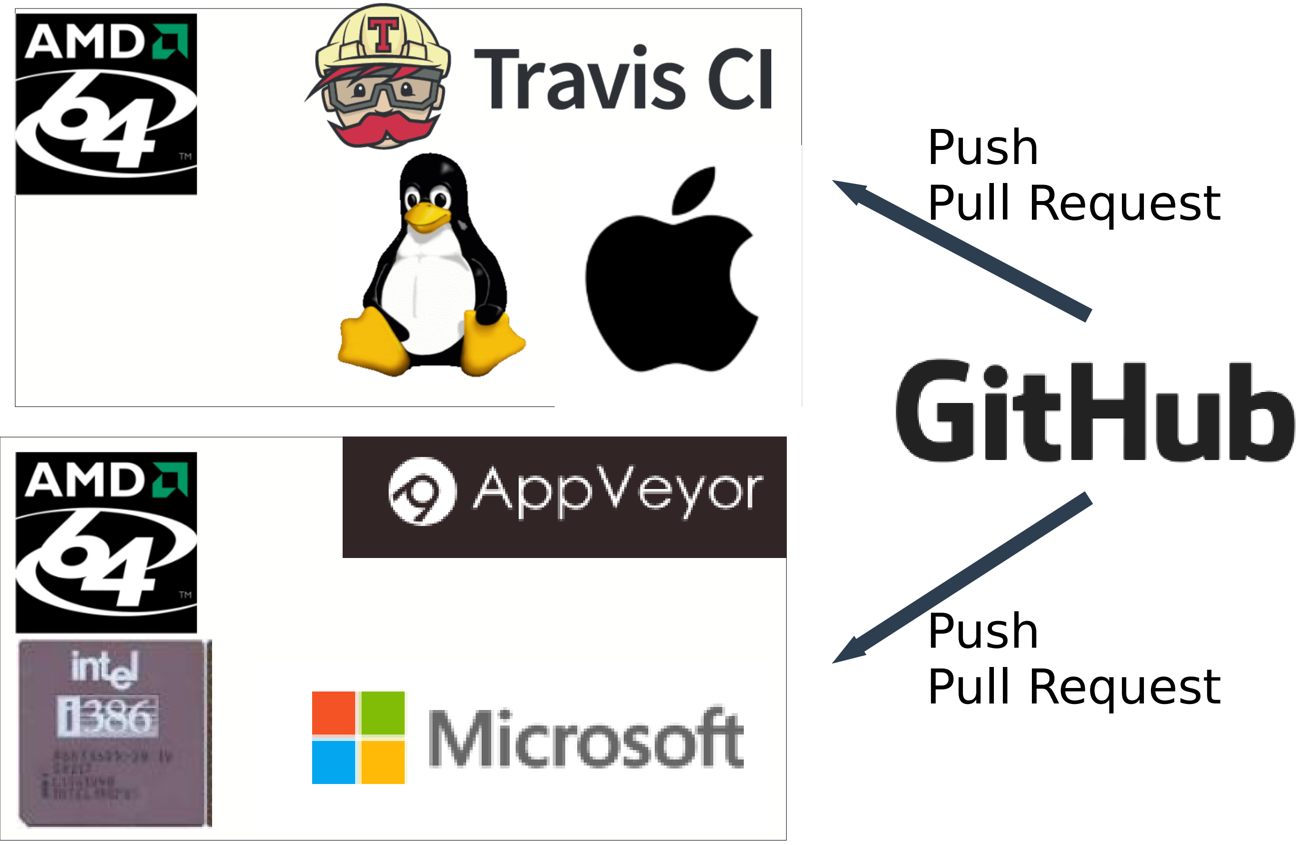
```
// Features describe the optional features of the Fs
type Features struct {
        Purge func() error
        Copy func(src Object, remote string) (Object, error)
        Move func(src Object, remote string) (Object, error)
        DirMove func(src Fs, srcRemote, dstRemote string) error
        ChangeNotify func(func(string, EntryType), <-chan time.Dura
        PublicLink func(remote string) (string, error)
        PutStream func(in io.Reader, src ObjectInfo, options ...Ope
        CleanUp func() error
        ListR ListRFn
        About func() (*Usage, error)
}
```

Testing

- **How to test**
  - 27 backends
  - x 50 commands
  - x 8 OSes
  - x 6 CPU Architectures
  - x 4 Go versions?
- **69k lines of code**
- **26k lines of test code**

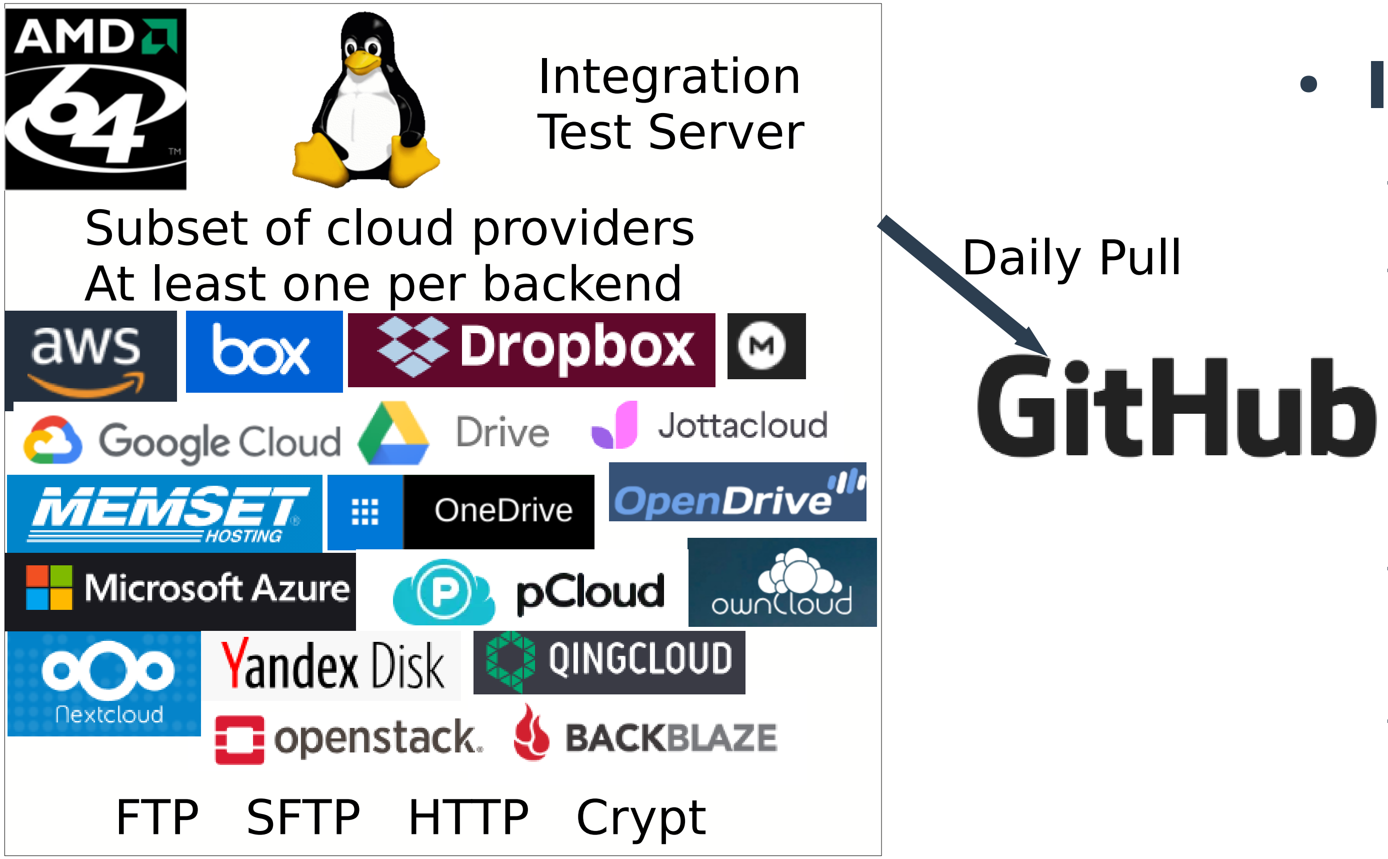
- **Unit test what we can**
  - Some things are easy
  - Who wants to write mocks for 27 different cloud providers?
- **Integration test**
  - Integration tests use go test framework
  - Run daily

CI – Unit testing and build



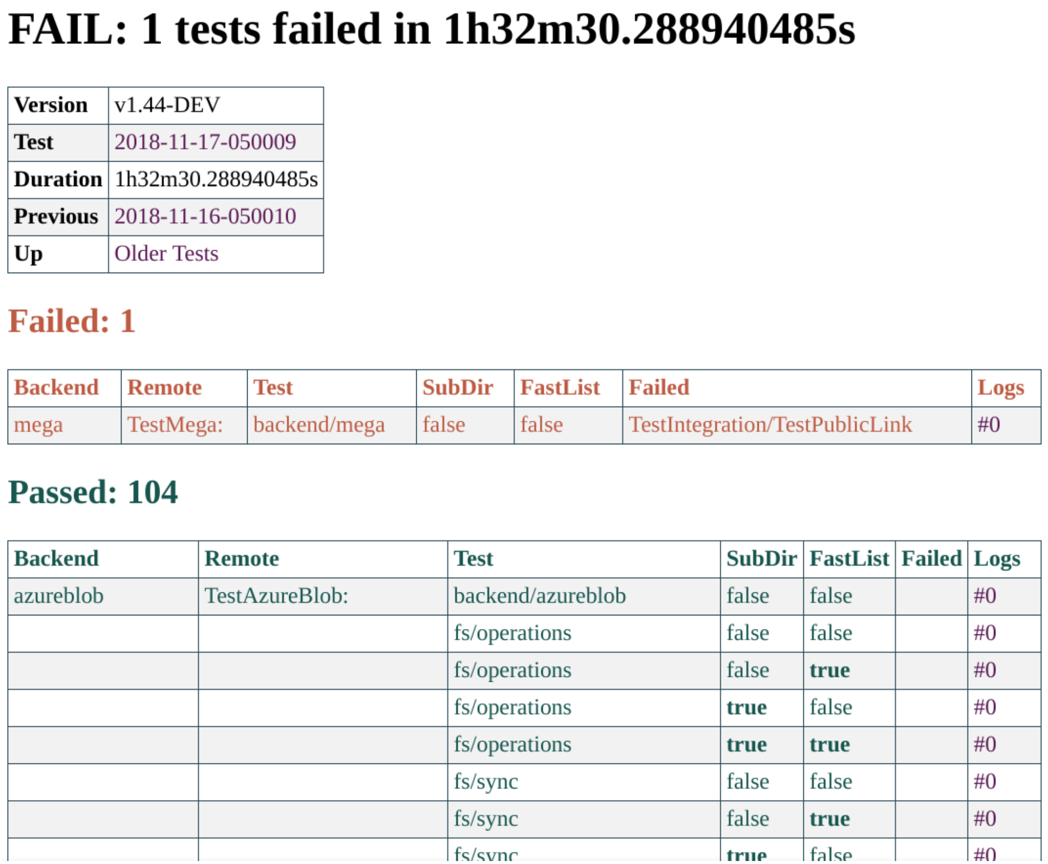
- **CI Pipeline**
  - Runs all non integration tests
  - Tests mount
  - Builds for all
  - Makes binaries
  - Uploads to beta release

Integration testing



- **Integration test**
  - Run daily
  - Too expensive to run on every push
    - Cost ~ 30p
    - Time ~ 1 Hour
  - Creates fancy report
  - Not integrated with Github (yet)

Integration tests



FAIL: 1 tests failed in 1h32m30.288940485s

Version	v1.44-DEV
Test	2018-11-17-050009
Duration	1h32m30.288940485s
Previous	2018-11-16-050010
Up	Older Tests

Failed: 1

Backend	Remote	Test	SubDir	FastList	Failed	Logs
mega	TestMega:	backend/mega	false	false	TestIntegration/TestPublicLink	#0

Passed: 104

Backend	Remote	Test	SubDir	FastList	Failed	Logs
azureblob	TestAzureBlob:	backend/azureblob	false	false		#0
		fs/operations	false	false		#0
		fs/operations	false	**true**		#0
		fs/operations	**true**	false		#0
		fs/operations	**true**	**true**		#0
		fs/sync	false	false		#0
		fs/sync	false	**true**		#0
		fs/sync	**true**	false		#0

Retrying integration tests

- **test_all framework**
  - Runs standard go tests
  - Runs lots of tests in parallel
  - Provides flags as specified in a config file
  - Parses the output of the tests
  - Retries the just the failing tests
  - Should probably become an opensource package in its own right!

```
Attempt 1/5
./operations.test
-test.v
-test.timeout 30m0s
-remote TestAzureBlob:
```

```
Attempt 2/5
./operations.test
-test.v
-test.timeout 30m0s
-remote TestAzureBlob:
-test.run '^(TestPurge|
TestRmdirsNoLeaveRoot)$'
```

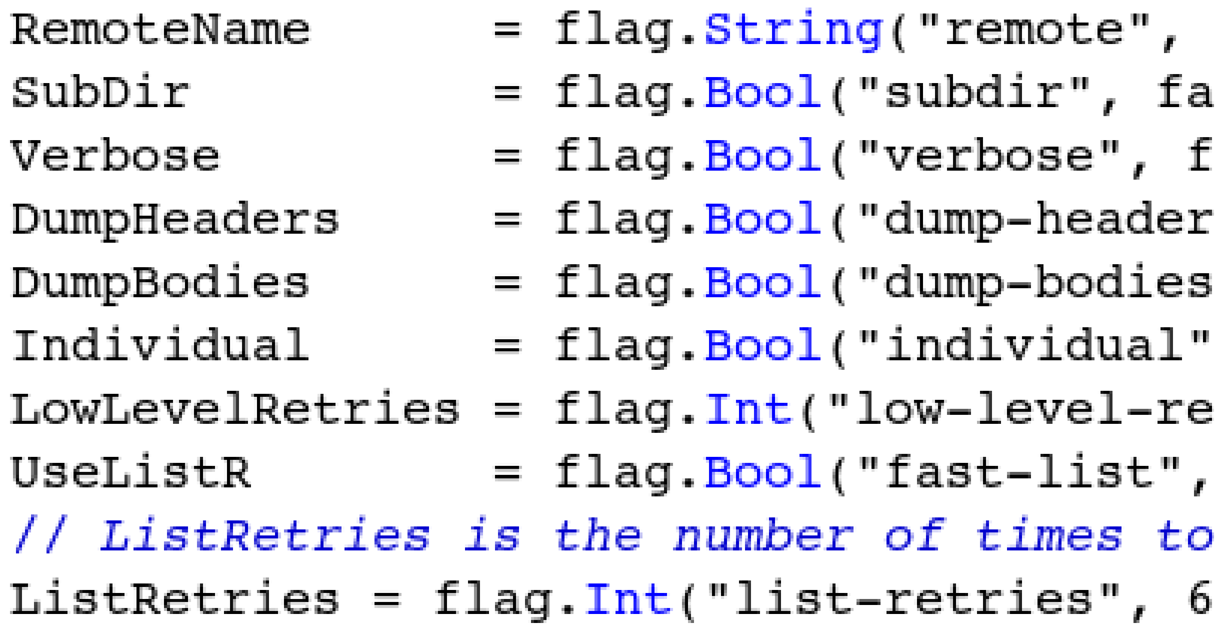
Integration tests for backends

```go
// Test B2 filesystem interface
package b2

import (
	"testing"

	"github.com/ncw/rclone/fs"
	"github.com/ncw/rclone/fstest/fstests"
)

// TestIntegration runs integration tests against the remote
func TestIntegration(t *testing.T) {
	fstests.Run(t, &fstests.Opt{
		RemoteName: "TestB2:",
		NilObject:  (*Object)(nil),
		ChunkedUpload: fstests.ChunkedUploadConfig{
			MinChunkSize:       minChunkSize,
			NeedMultipleChunks: true,
		},
	})
}
```

- **Backend integration tests**
  - Easy to add thanks to go1.6 nested tests
  - Give a recipe to follow when making a new backend
  - Just make the integration tests pass
  - Originally done with code gen pre go1.6

Integration tests elsewhere

- **You can add flags to tests**
  - Rclone uses this with a “-remote” flag to signal that the test should be done remotely
  - There are other flags for debugging and more in depth tests



```
RemoteName      = flag.String("remote",
SubDir          = flag.Bool("subdir", fa
Verbose         = flag.Bool("verbose", f
DumpHeaders     = flag.Bool("dump-header
DumpBodies      = flag.Bool("dump-bodies
Individual      = flag.Bool("individual"
LowLevelRetries = flag.Int("low-level-re
UseListR        = flag.Bool("fast-list",
// ListRetries is the number of times to
ListRetries = flag.Int("list-retries", 6
```

Standing on the shoulders of giants

- **Rclone**
  - 95,000 lines of code
  - 450 source files
  - Not including “vendor”

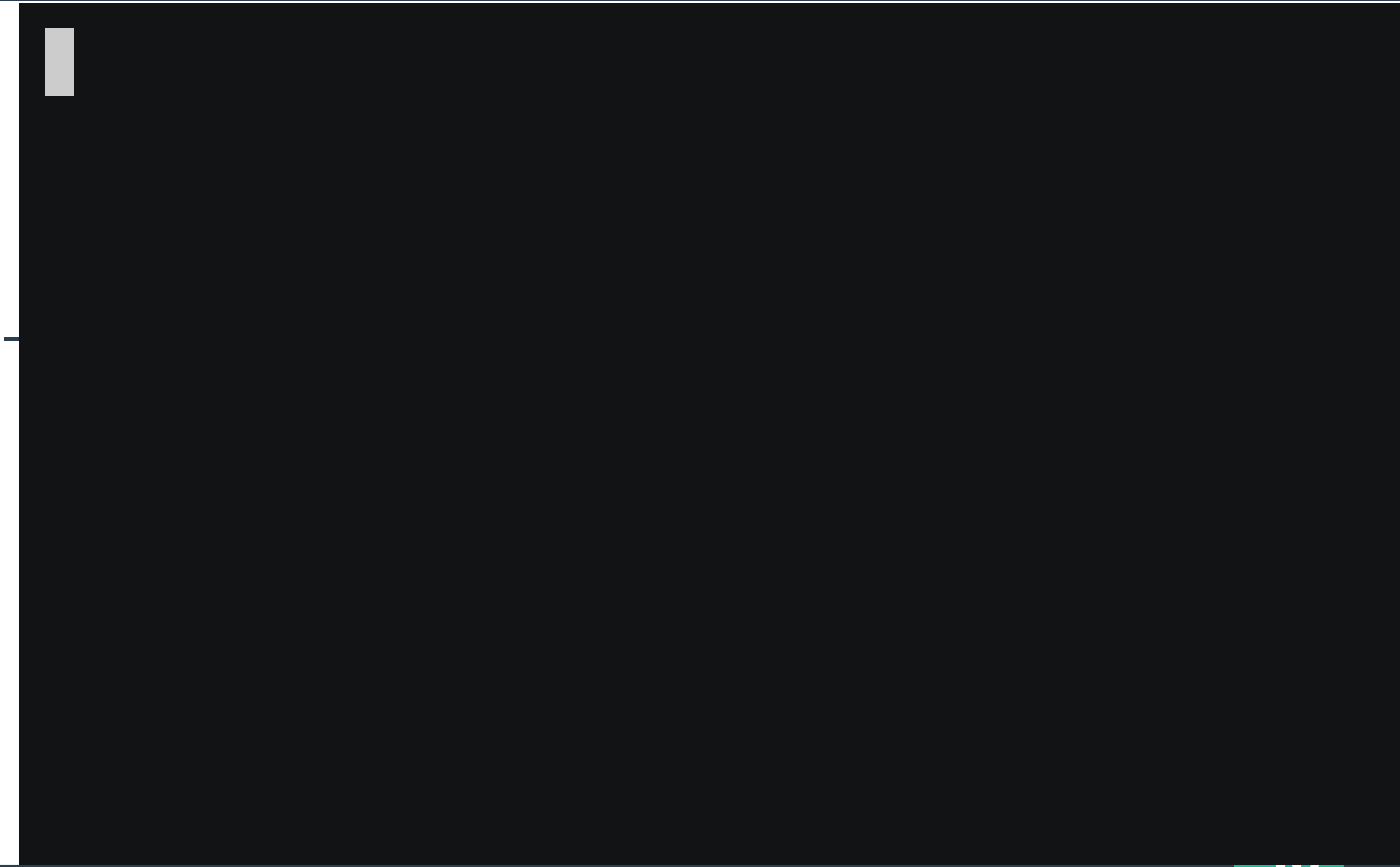
- **Rclone’s libraries**
  - 520,000 lines of code
  - 1,100 files
  - All stored in “vendor”

All build on top of the excellent standard library

Favourite libraries and tools: golang.org/x/tools/cmd/goimports

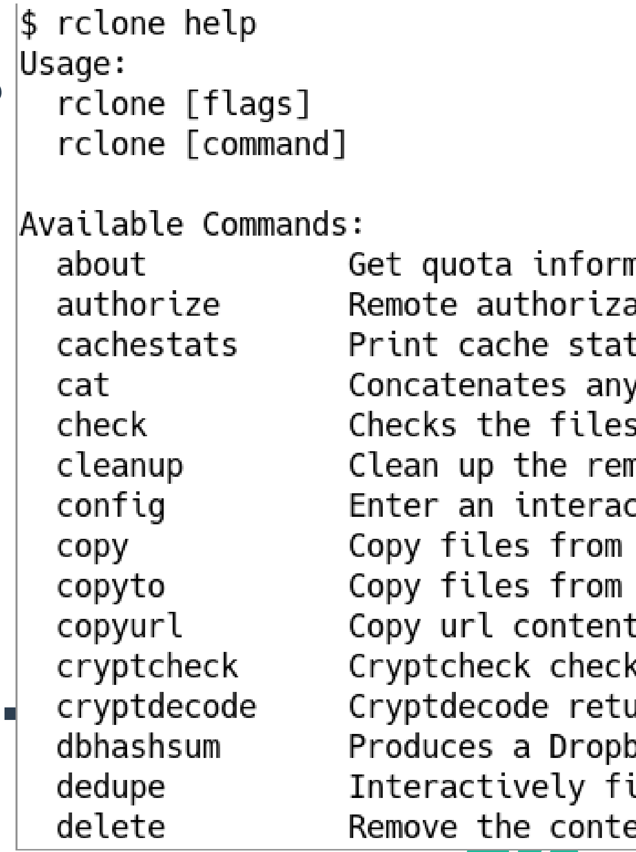
- Get it in your editor – never type an import statement again

- Run it as a save hook – it will `go fmt` your code too



github.com/spf13/cobra

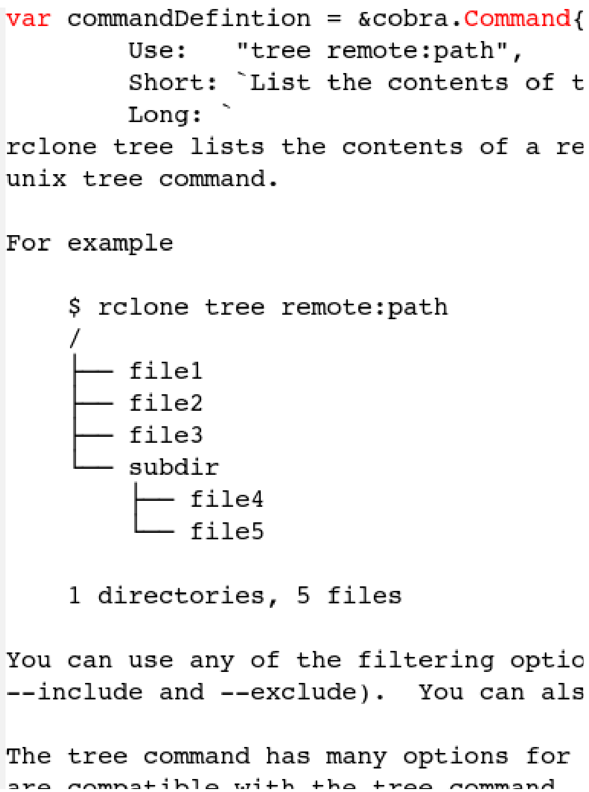
- **Make commands with subcommands**
- **Very flexible / extensible**
- **Used by Kubernetes / Hugo / Docker**
- **POSIX flags `--flag` with spf13/pflag**
- **Creates bash completion scripts**
- **Creates docs**
- **Makes coffee and cleans the kitchen.**



```
$ rclone help
Usage:
  rclone [flags]
  rclone [command]

Available Commands:
  about           Get quota inform
  authorize       Remote authoriza
  cachestats      Print cache stat
  cat             Concatenates any
  check           Checks the files
  cleanup         Clean up the rem
  config          Enter an interac
  copy            Copy files from
  copyto          Copy files from
  copyurl         Copy url content
  cryptcheck      Cryptcheck check
  cryptdecode     Cryptdecode retu
  dbhashsum       Produces a Dropb
  dedupe          Interactively fi
  delete          Remove the conte
```

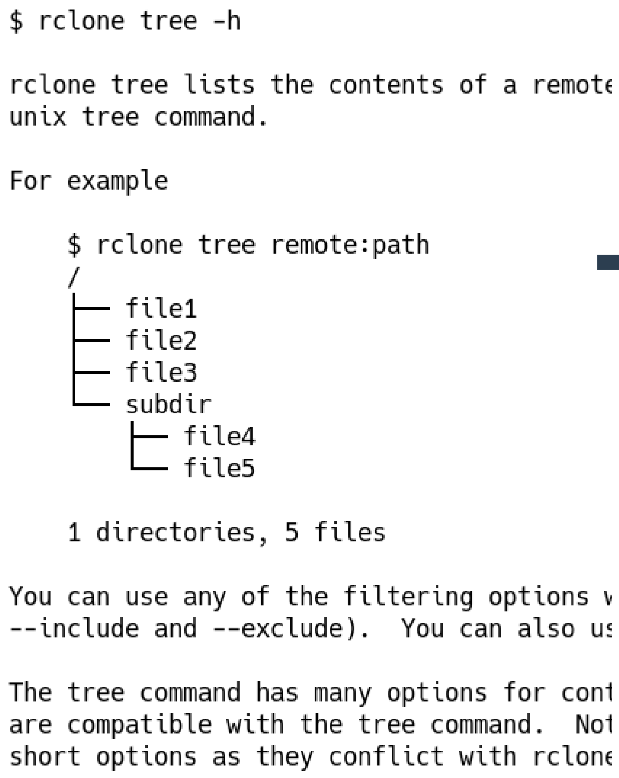
Documentation with github.com/spf13/cobra

Go code defines help…



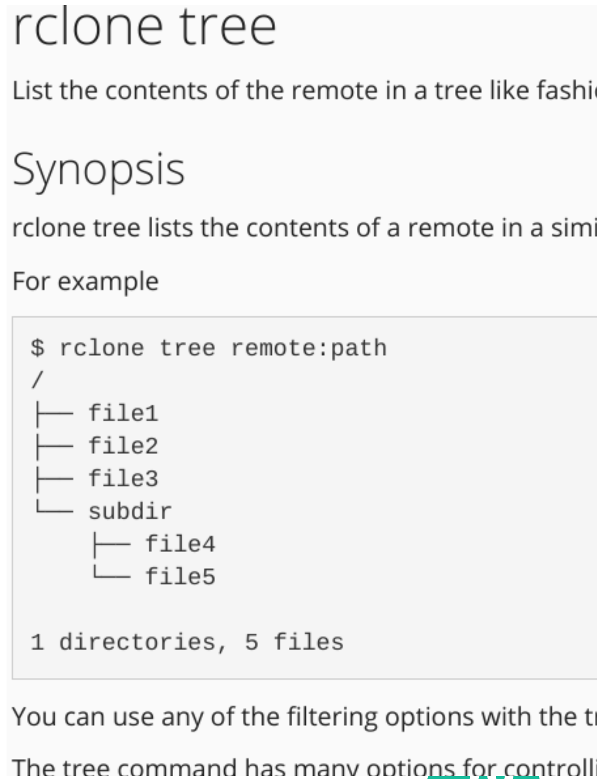
```
var commandDefintion = &cobra.Command{
        Use:   "tree remote:path",
        Short: `List the contents of t
        Long: `
rclone tree lists the contents of a re
unix tree command.

For example

    $ rclone tree remote:path
    /
    ├── file1
    ├── file2
    ├── file3
    └── subdir
        ├── file4
        └── file5

    1 directories, 5 files

You can use any of the filtering optio
--include and --exclude).  You can als

The tree command has many options for
are compatible with the tree command
```

…becomes -h output…



```
$ rclone tree -h

rclone tree lists the contents of a remote
unix tree command.

For example

    $ rclone tree remote:path
    /
    ├── file1
    ├── file2
    ├── file3
    └── subdir
        ├── file4
        └── file5

    1 directories, 5 files

You can use any of the filtering options w
--include and --exclude).  You can also us

The tree command has many options for cont
are compatible with the tree command.  Not
short options as they conflict with rclone
```

…and markdown for web.



rclone tree

List the contents of the remote in a tree like fashi

Synopsis

rclone tree lists the contents of a remote in a simi

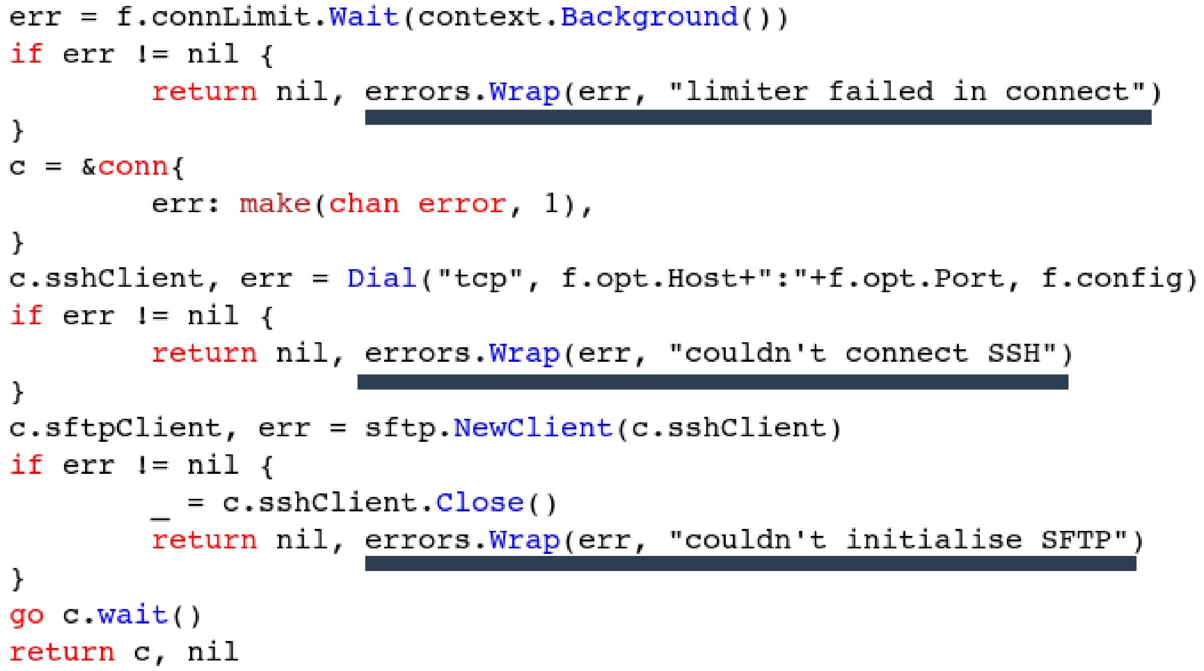
For example

```
$ rclone tree remote:path
/
├── file1
├── file2
├── file3
└── subdir
    ├── file4
    └── file5

1 directories, 5 files
```

You can use any of the filtering options with the tr

The tree command has many options for controll

github.com/pkg/errors

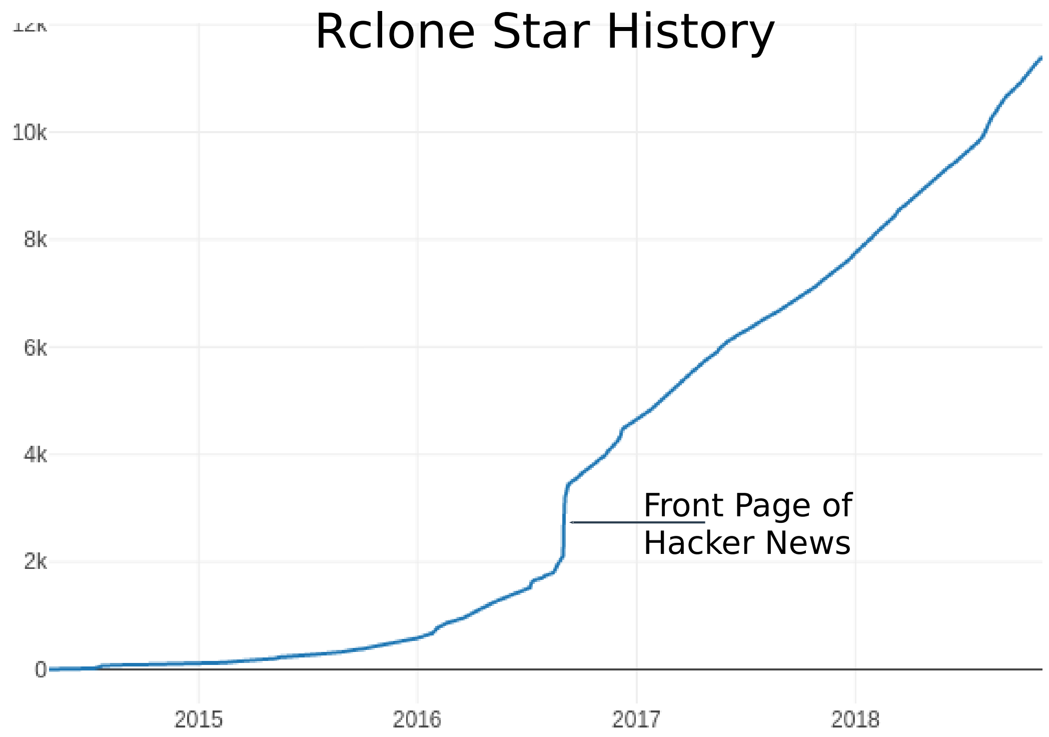


```go
err = f.connLimit.Wait(context.Background())
if err != nil {
        return nil, errors.Wrap(err, "limiter failed in connect")
}
c = &conn{
        err: make(chan error, 1),
}
c.sshClient, err = Dial("tcp", f.opt.Host+":"+f.opt.Port, f.config)
if err != nil {
        return nil, errors.Wrap(err, "couldn't connect SSH")
}
c.sftpClient, err = sftp.NewClient(c.sshClient)
if err != nil {
        _ = c.sshClient.Close()
        return nil, errors.Wrap(err, "couldn't initialise SFTP")
}
go c.wait()
return c, nil
```

- **Turns an error like this**
  - “unexpected EOF”
- **Into**
  - “NewFs creating backend: couldn’t connect SSH: unexpected EOF”

What to do if your open source project takes off...

- **Don’t Panic!**
- **Open a forum (Discourse is good)**
- **Ask everyone who makes an issue for help**
- **Recruit pull requesters as contributors**
- **Make good contributing docs**
- **Get octobox.io**



Thank you for listening

- **Rclone "rsync for cloud storage"**
  - https://rclone.org
  - https://github.com/ncw/rclone
- **Talk by**
  - Nick Craig-Wood
  - Twitter: @njcw
  - Email: nick@craig-wood.com
- **Special effects by**
  - Gource – source code history visualisation
  - Asciinema and asciicast2gif – terminal GIFs