Yandex

Yandex Cloud

Unified Point-in-Time Recovery in the Cloud

Andrey Borodin, Team lead of opensource RDBMS development, Yandex.Cloud
Dmitry Smal, Team lead of Managed MySQL and SQL Server Development, Yandex.Cloud

- Managed Service for PostgreSQL
- Managed Service for MySQL
- Managed Service for MongoDB
- Many more DBs

What’s interesting about Yandex.Cloud managed databases?

Yandex

PostgreSQL at Yandex.Cloud

Yandex.Mail

- Hundreds of millions of users
- 10^{12} rows, 10^{6} queries per second, ~1 PB of data

Many others services use managed PostgreSQL

- Taxi, carsharing, food delivery, self driving cars, etc.
- Total of 3 million queries per second, 6 PB of space used

MySQL at Yandex.Cloud

Yandex.Direct

- Online advertising network
- Uses Percona build of MySQL at scale

Managed MySQL

- 400+ TB as of 2021

Less is more

- Cloud providers usually sell computing resources.
- We aim to utilize fewer resources for the same workload.

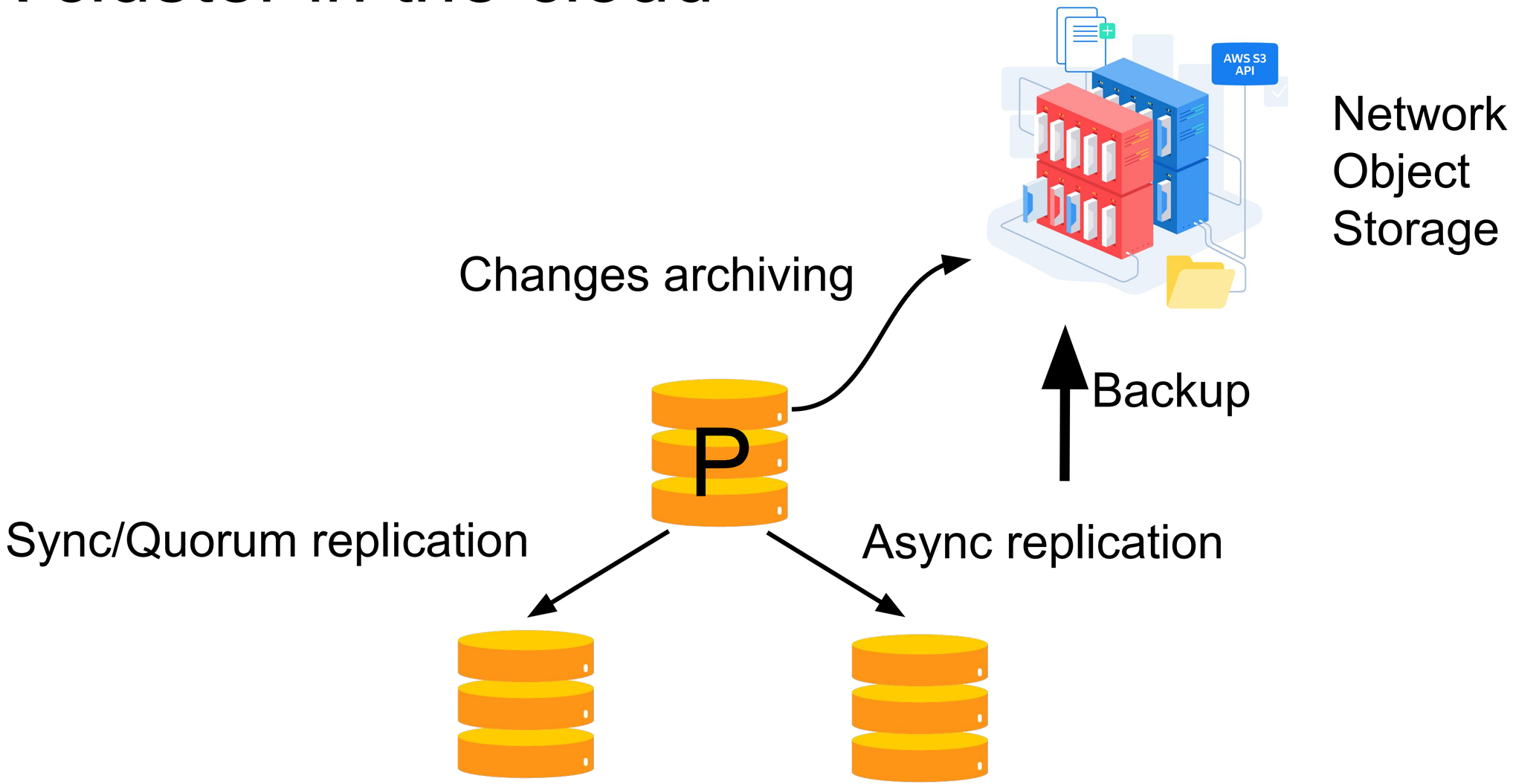
Point-in-time recovery

- Backup + changes
  - Scalable
  - Reliable
  - Efficient
  - Fast

Scalability

- Data: from 10 GB to 10 TB on a host
- RAM: from 2 GB
- Number of CPUs: from 0.05 to ~100
- Async and parallel whenever possible
- Don’t spill anything on a local disk

HA cluster in the cloud



Resources

- Storage space

Resources

- ~~Storage space~~
- CPU
- Net bandwidth
- Disk IOPS

Reliability

- Protection from human error via automation and safety checks
- Prevention of data corruption
- Consistency monitoring
- Integration with other systems (HA tool)
- Extensibility and unification of approaches
- Encrypted data in storage

Fast recovery

- OLAP

From start to consistency point

- OLTP standby

To starting streaming replication

- OLTP primary

Until recovery target and accept of write queries

Unacceptable

- Data locks

Business can’t wait

- Data loss

We call it a “database” after all

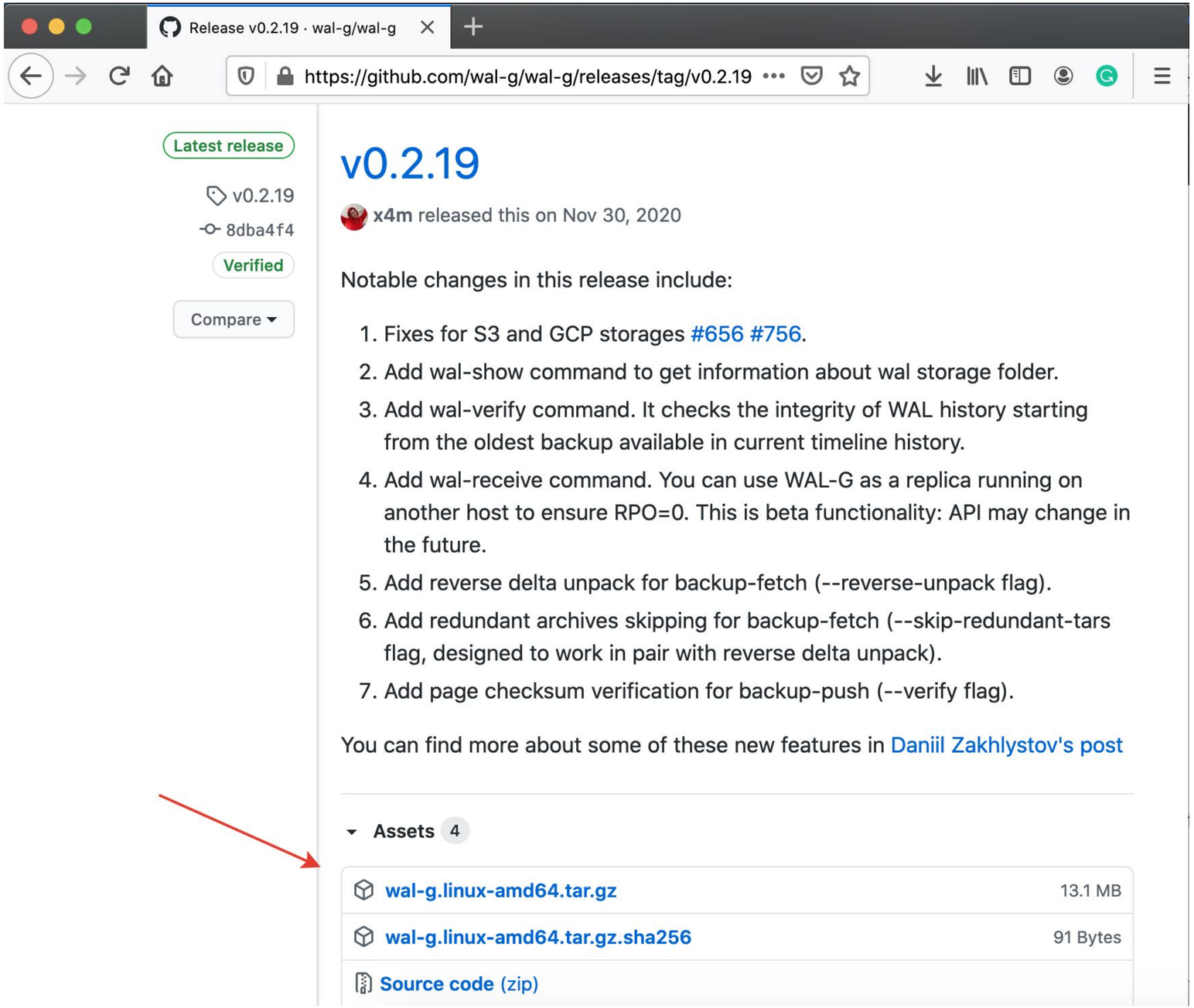
pgProBackup

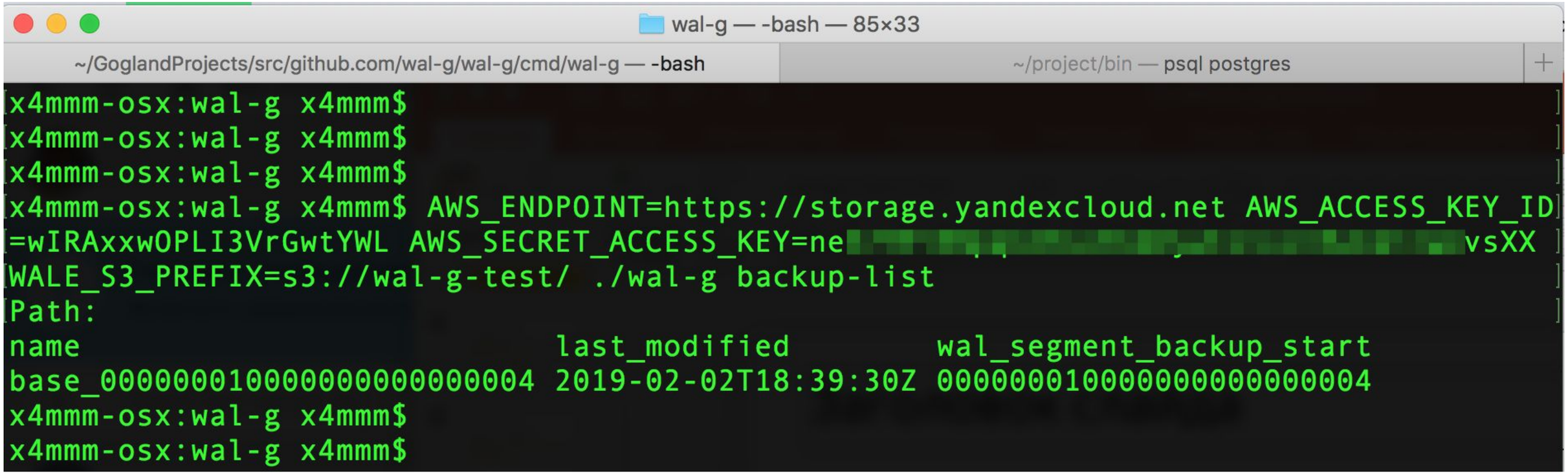
Barman

Backup and recovery manager for PostgreSQL

pgBackRest

WAL-G

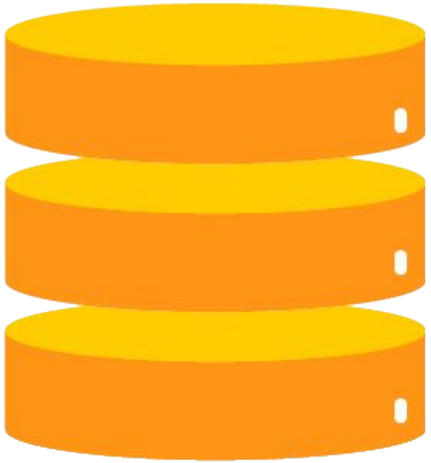

Release v0.2.19 · wal-g/wal-g
https://github.com/wal-g/wal-g/releases/tag/v0.2.19
Latest release
v0.2.19
8dba4f4
Verified
Compare
v0.2.19
x4m released this on Nov 30, 2020
Notable changes in this release include:
1. Fixes for S3 and GCP storages #656 #756.
2. Add wal-show command to get information about wal storage folder.
3. Add wal-verify command. It checks the integrity of WAL history starting from the oldest backup available in current timeline history.
4. Add wal-receive command. You can use WAL-G as a replica running on another host to ensure RPO=0. This is beta functionality: API may change in the future.
5. Add reverse delta unpack for backup-fetch (--reverse-unpack flag).
6. Add redundant archives skipping for backup-fetch (--skip-redundant-tars flag, designed to work in pair with reverse delta unpack).
7. Add page checksum verification for backup-push (--verify flag).
You can find more about some of these new features in Daniil Zakhlystov's post
Assets 4
wal-g.linux-amd64.tar.gz 13.1 MB
wal-g.linux-amd64.tar.gz.sha256 91 Bytes
Source code (zip)


wal-g — -bash — 85×33
~/GoglandProjects/src/github.com/wal-g/wal-g/cmd/wal-g — -bash
~/project/bin — psql postgres
x4mmm-osx:wal-g x4mmm$
x4mmm-osx:wal-g x4mmm$
x4mmm-osx:wal-g x4mmm$
x4mmm-osx:wal-g x4mmm$ AWS_ENDPOINT=https://storage.yandexcloud.net AWS_ACCESS_KEY_ID
=wIRAxxwOPLI3VrGwtYWL AWS_SECRET_ACCESS_KEY=ne
vsXX
WALE_S3_PREFIX=s3://wal-g-test/ ./wal-g backup-list
Path:
name last_modified wal_segment_backup_start
base_000000010000000000000004 2019-02-02T18:39:30Z 000000010000000000000004
x4mmm-osx:wal-g x4mmm$
x4mmm-osx:wal-g x4mmm$

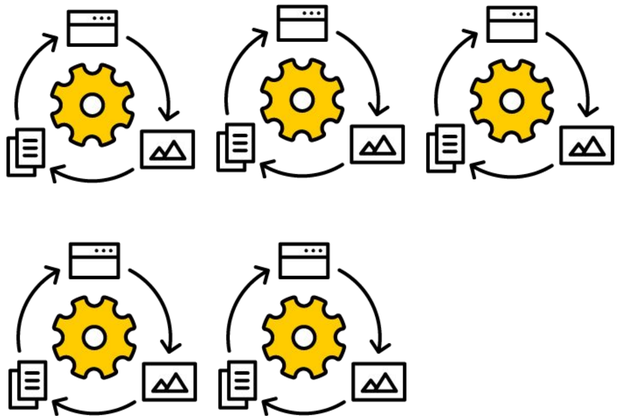
```
x4mmm-osx:wal-g x4mmm$
x4mmm-osx:wal-g x4mmm$ AWS_ENDPOINT=https://storage.yandexcloud.net AWS_ACCESS_KEY_ID
=wIRAxxwOPLI3VrGwtYWL AWS_SECRET_ACCESS_KEY=neh7EEYANpqGS5GJEbmOywhznxcIBukG3IamvsXX
WALE_S3_PREFIX=s3://wal-g-test/ ./wal-g backup-push ~/DemoDb
Path:
INFO: 2019/02/02 21:56:42.509465 Doing full backup.
WARNING: 2019/02/02 21:56:42.526434 It seems your archive_mode is not enabled. This w
ill cause inconsistent backup. Please consider configuring WAL archiving.
INFO: 2019/02/02 21:56:42.740377 Walking ...
INFO: 2019/02/02 21:56:42.742571 Starting part 1 ...
INFO: 2019/02/02 21:56:43.112485 Finished writing part 1.
INFO: 2019/02/02 21:56:48.744337 Starting part 2 ...
INFO: 2019/02/02 21:56:48.761395 /global/pg_control
INFO: 2019/02/02 21:56:48.764006 Finished writing part 2.
INFO: 2019/02/02 21:56:48.878931 Starting part 3 ...
INFO: 2019/02/02 21:56:48.894990 backup_label
INFO: 2019/02/02 21:56:48.895030 tablespace_map
INFO: 2019/02/02 21:56:48.895056 Finished writing part 3.
INFO: 2019/02/02 21:56:49.523658 Uploaded 3 compressed tar Files.
x4mmm-osx:wal-g x4mmm$
```

```
# - Archiving -

archive_mode = on

archive_command = '/usr/bin/envdir /etc/wal-g/envdir
/usr/bin/timeout 600 /usr/bin/wal-g wal-push %p'
```

Normal backup



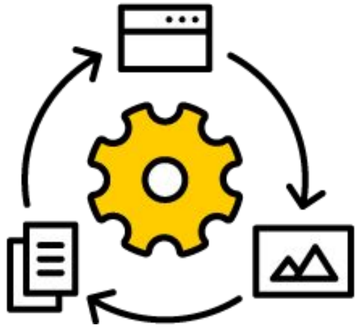
DB copy



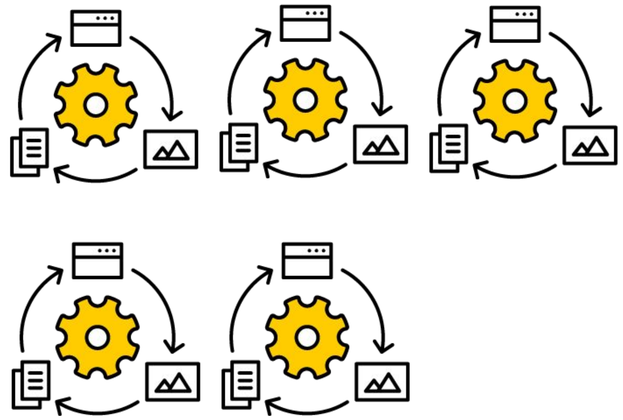
Changes (WAL)

Delta backups

DB copy

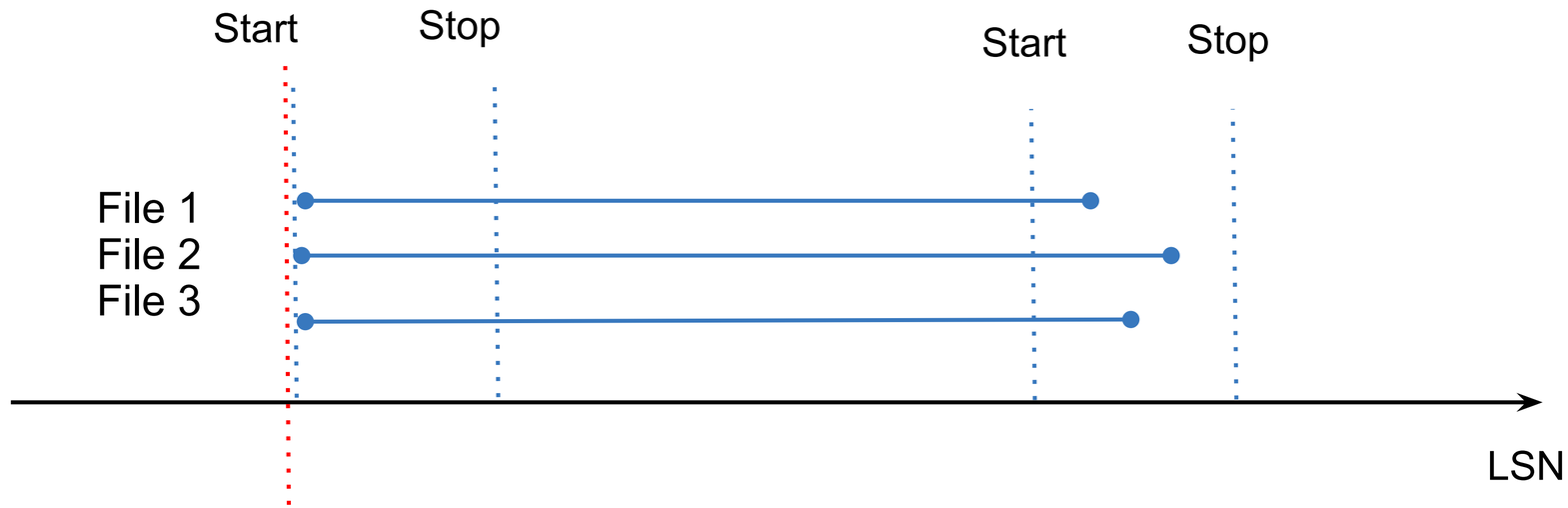


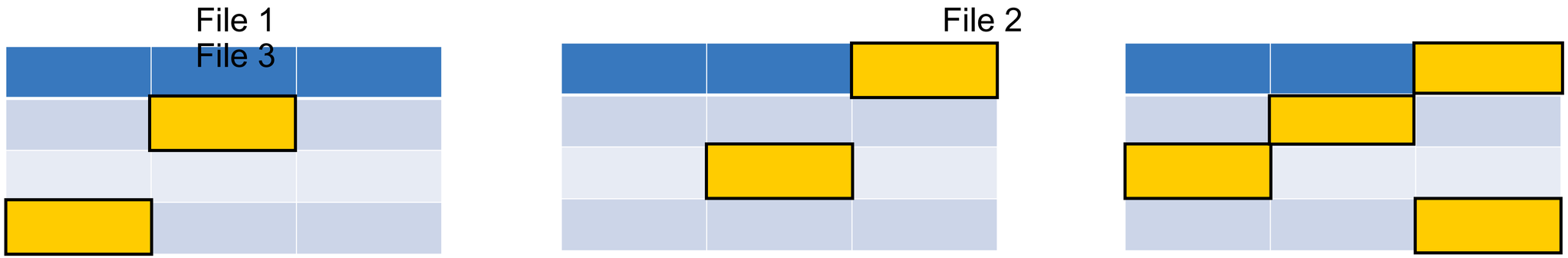
Delta copy



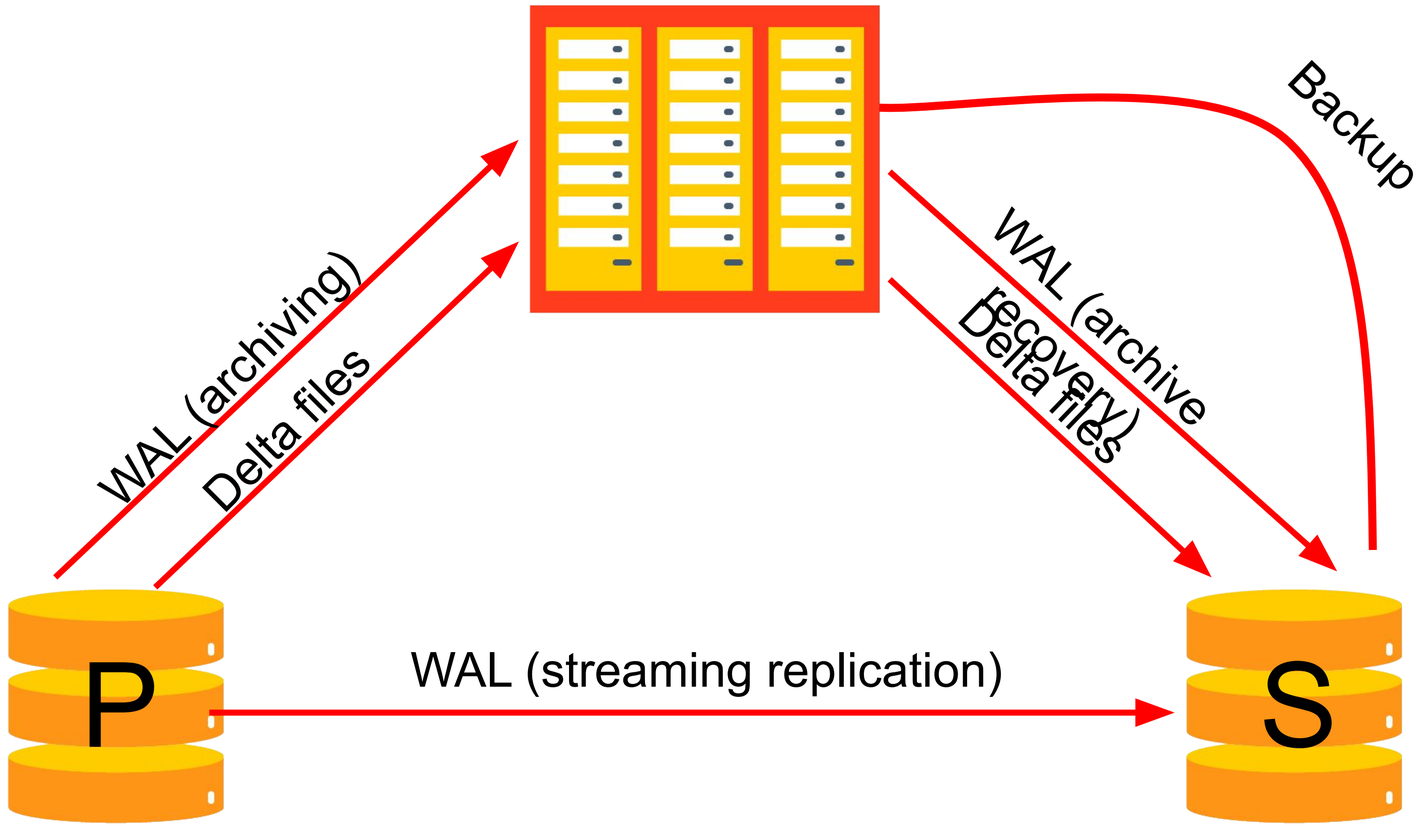
Changes (WAL)

LSN-based deltas





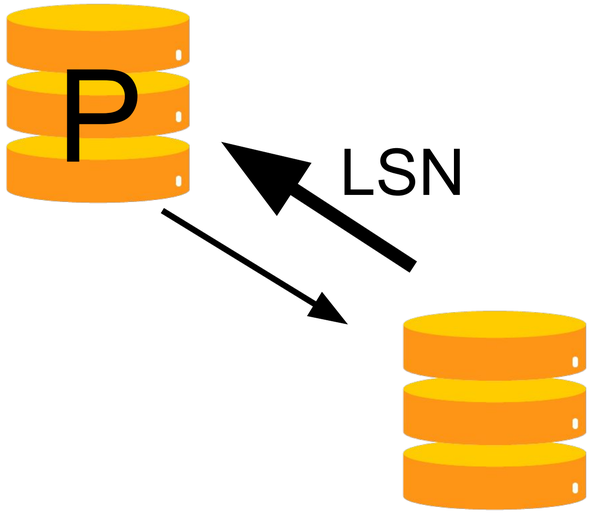
Data flows in the system



PG features

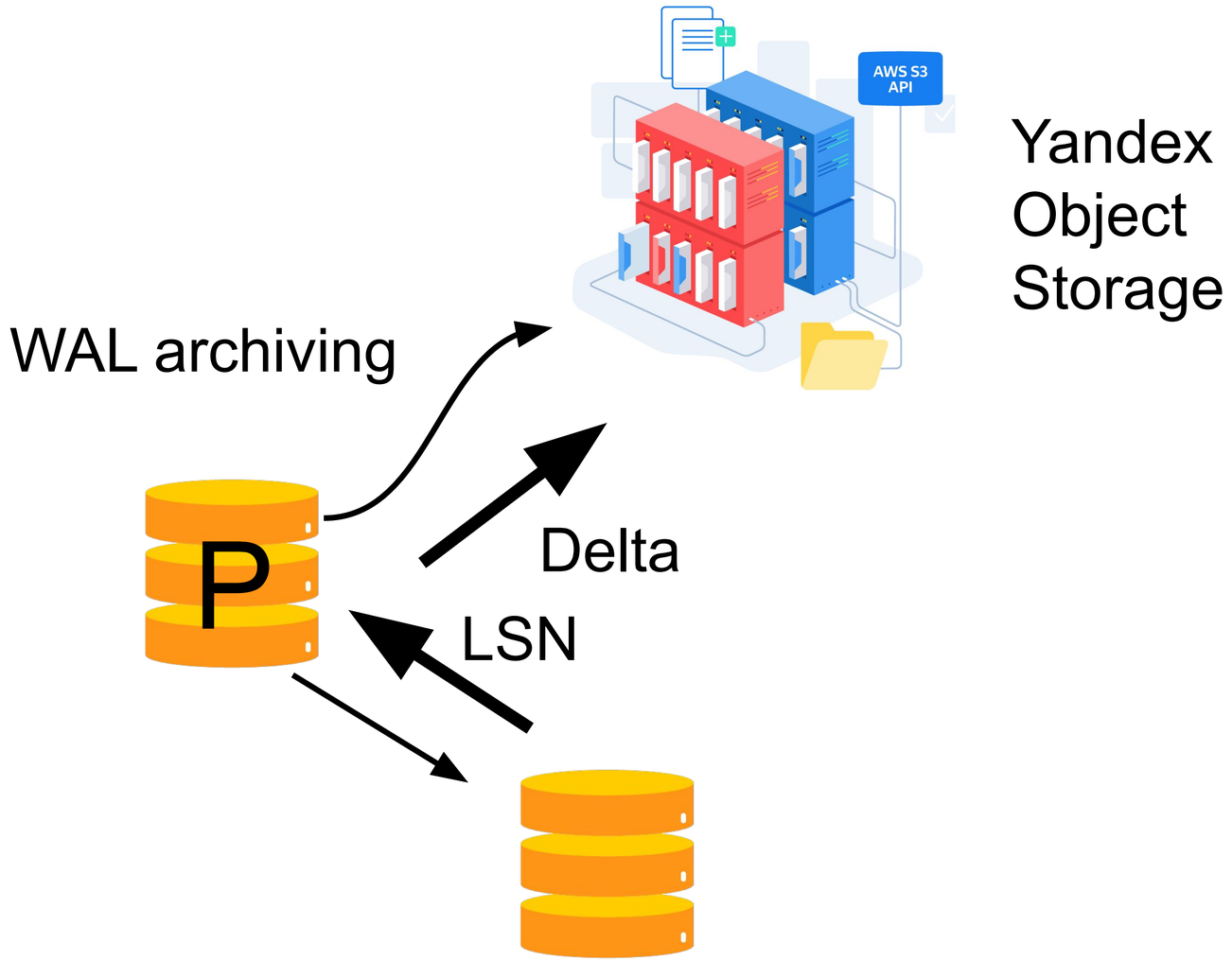
Catchup

Yandex
Object
Storage



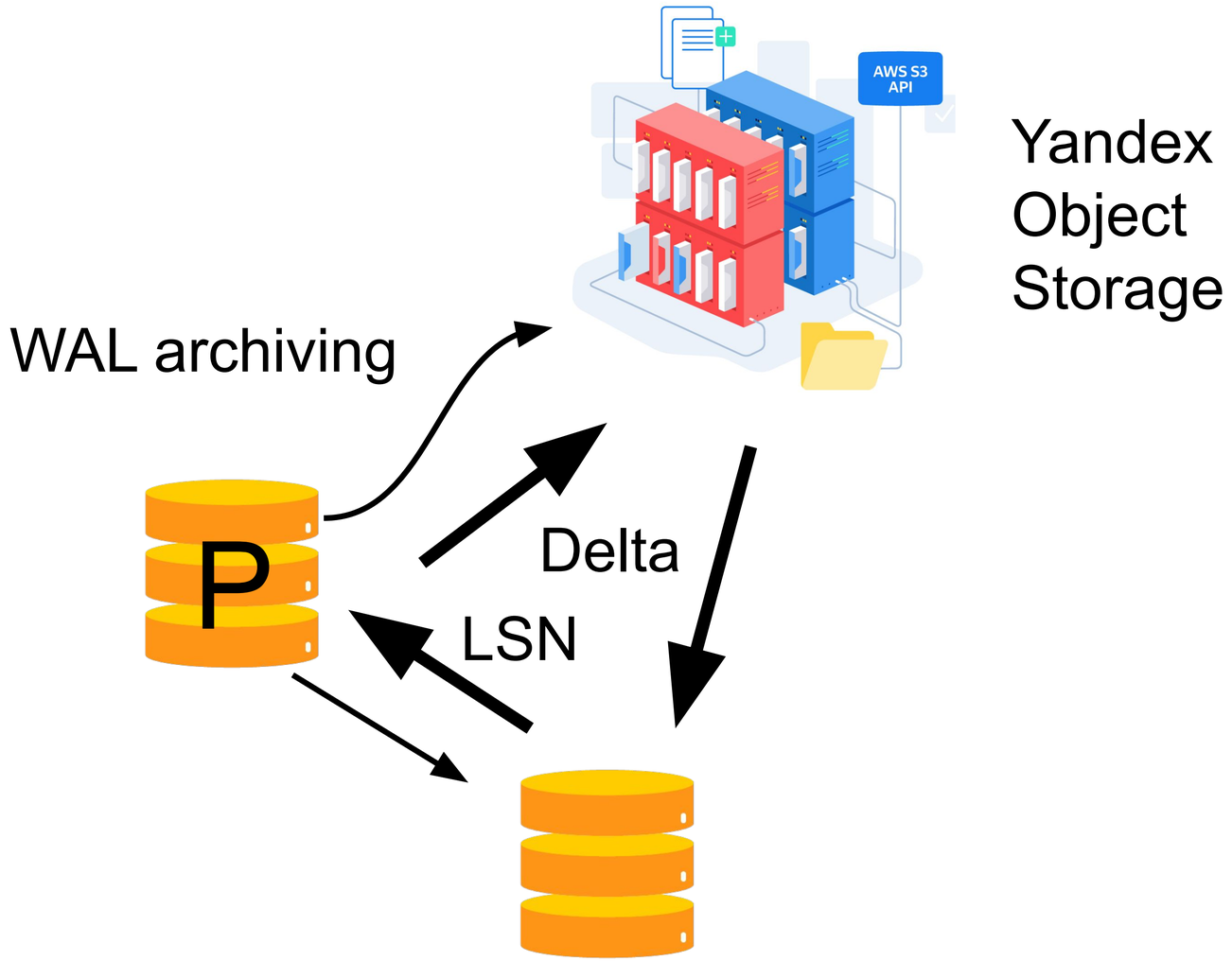
github.com/wal-g/wal-g/issues/363

Catchup



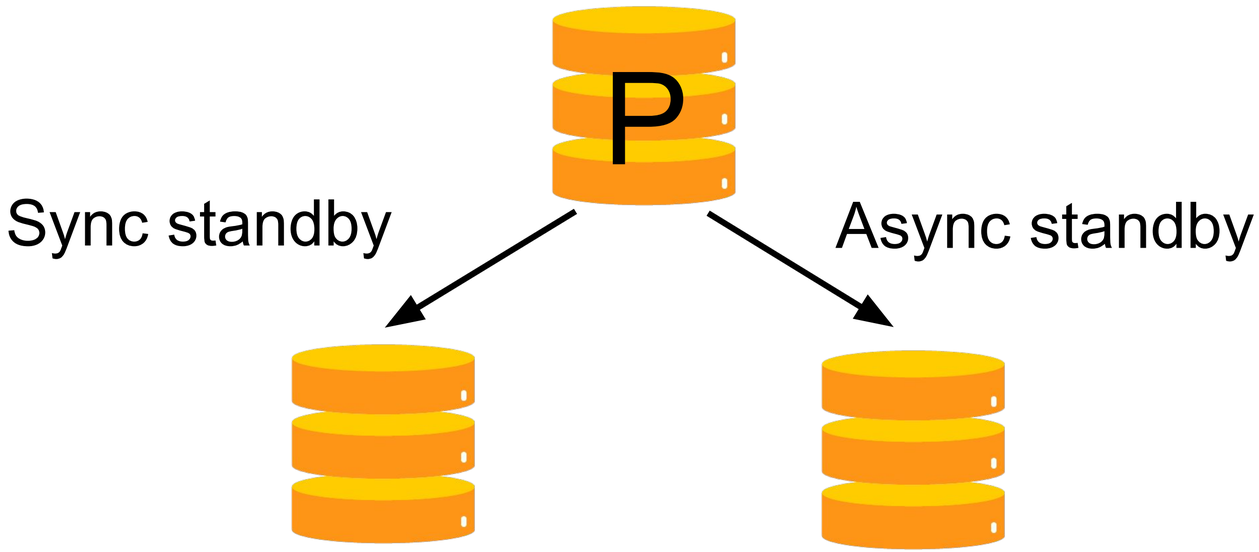
github.com/wal-g/wal-g/issues/363

Catchup



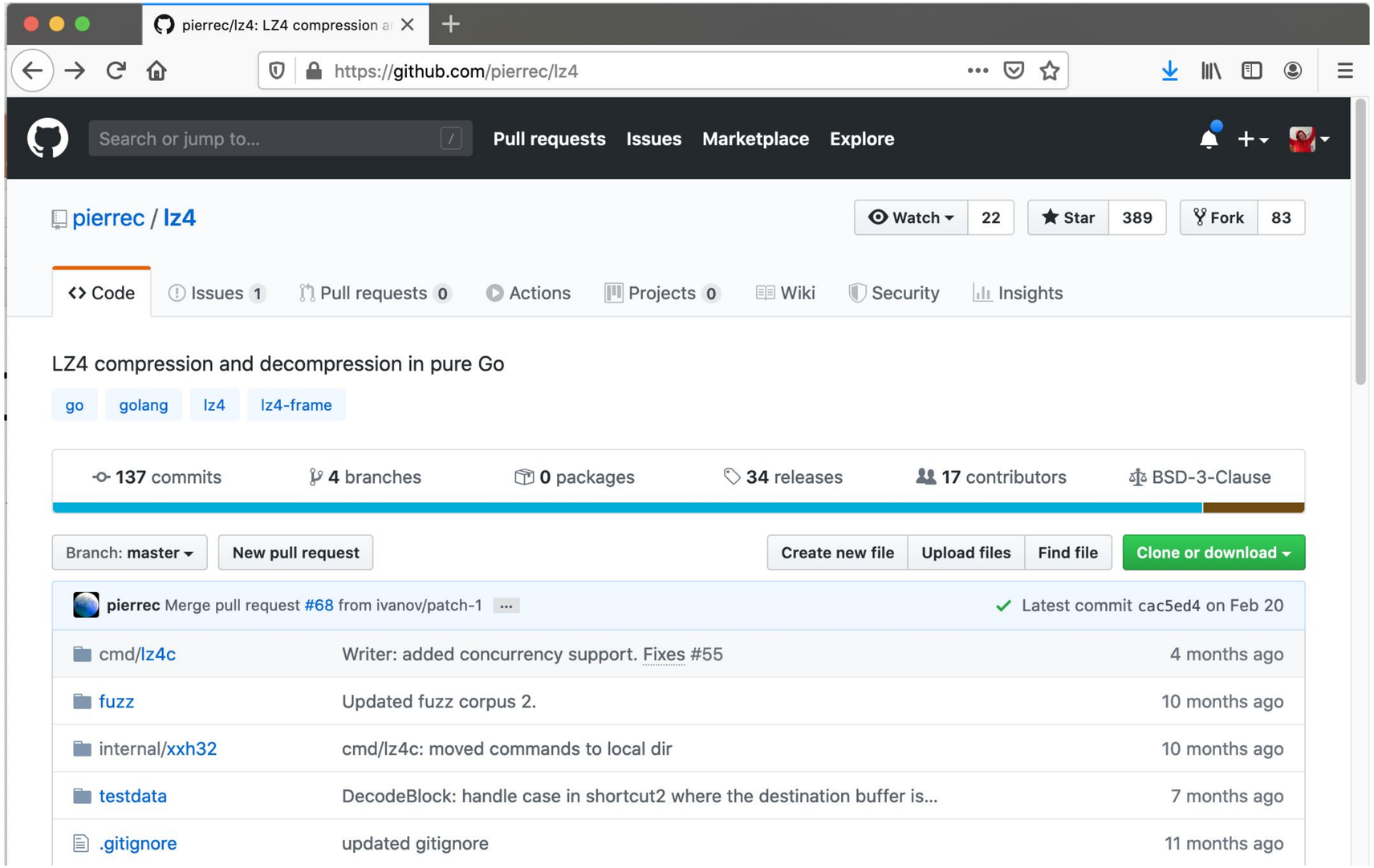
github.com/wal-g/wal-g/issues/363

Backup-push quorum

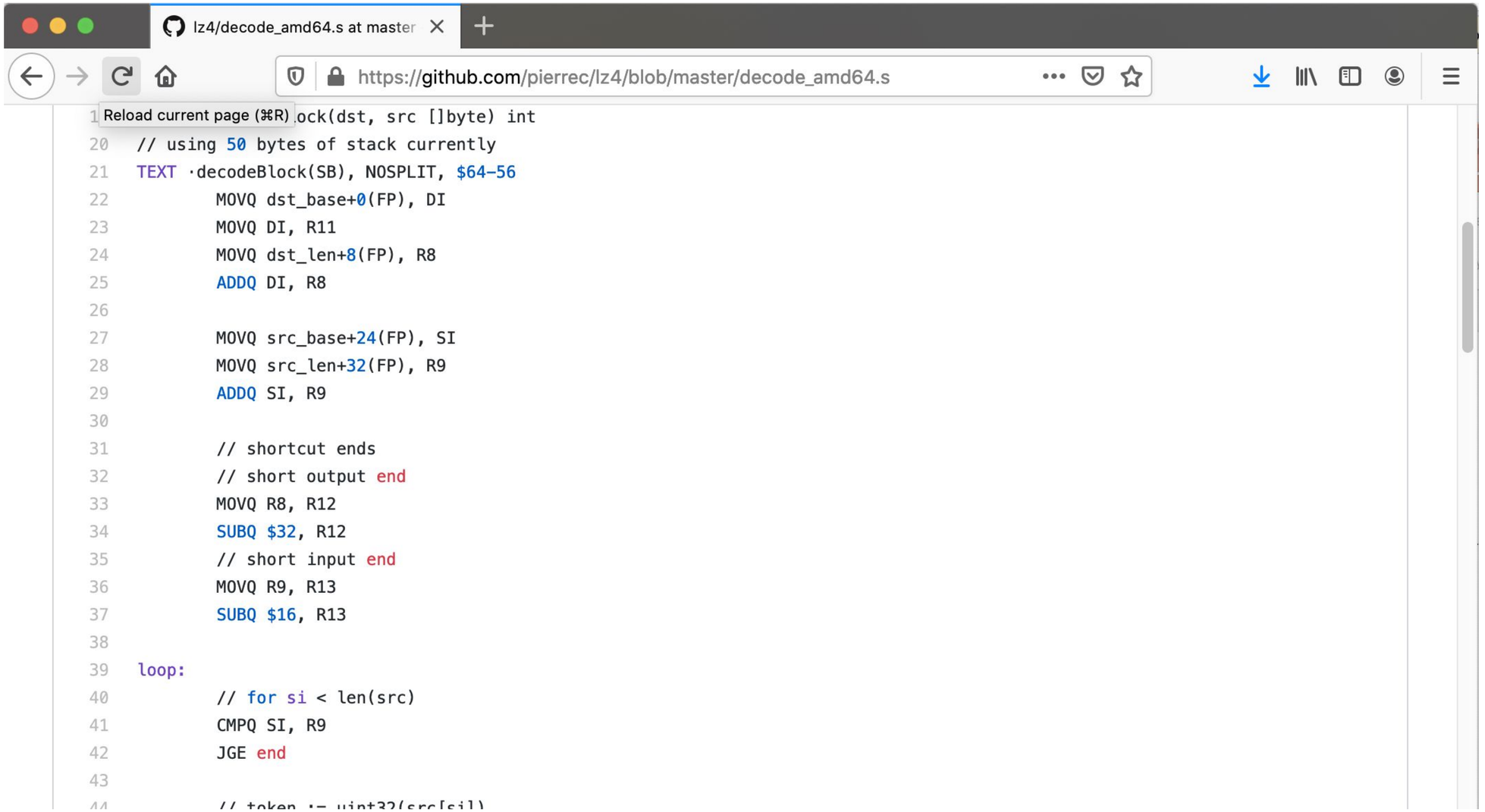


Compress some bytes

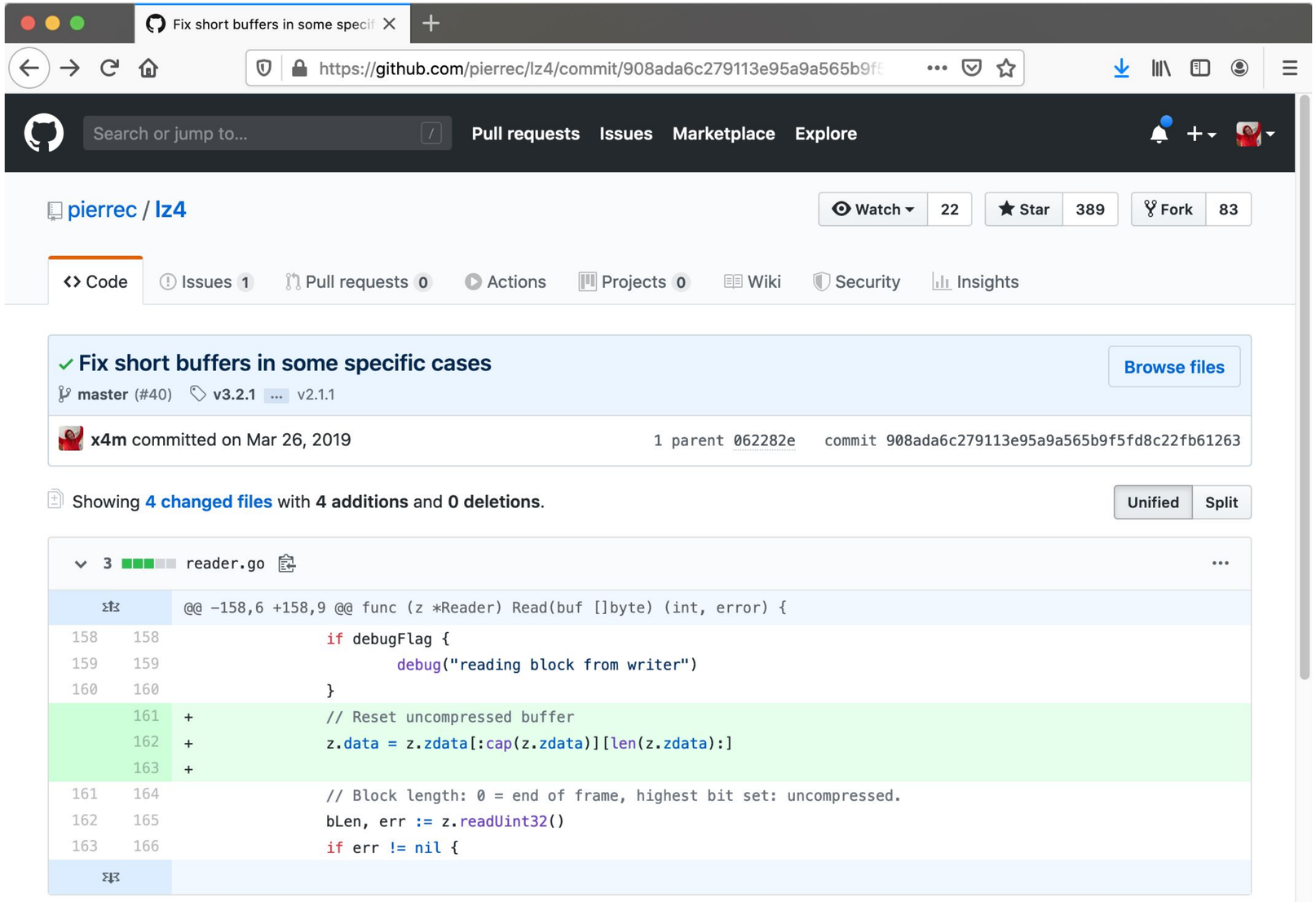
lz4


pierrec/lz4: LZ4 compression a
https://github.com/pierrec/lz4
Search or jump to...
Pull requests
Issues
Marketplace
Explore
pierrec / lz4
Watch 22
Star 389
Fork 83
Code
Issues 1
Pull requests 0
Actions
Projects 0
Wiki
Security
Insights
LZ4 compression and decompression in pure Go
go
golang
lz4
lz4-frame
137 commits
4 branches
0 packages
34 releases
17 contributors
BSD-3-Clause
Branch: master
New pull request
Create new file
Upload files
Find file
Clone or download
pierrec Merge pull request #68 from ivanov/patch-1
Latest commit cac5ed4 on Feb 20
cmd/lz4c
Writer: added concurrency support. Fixes #55
4 months ago
fuzz
Updated fuzz corpus 2.
10 months ago
internal/xxh32
cmd/lz4c: moved commands to local dir
10 months ago
testdata
DecodeBlock: handle case in shortcut2 where the destination buffer is...
7 months ago
.gitignore
updated gitignore
11 months ago

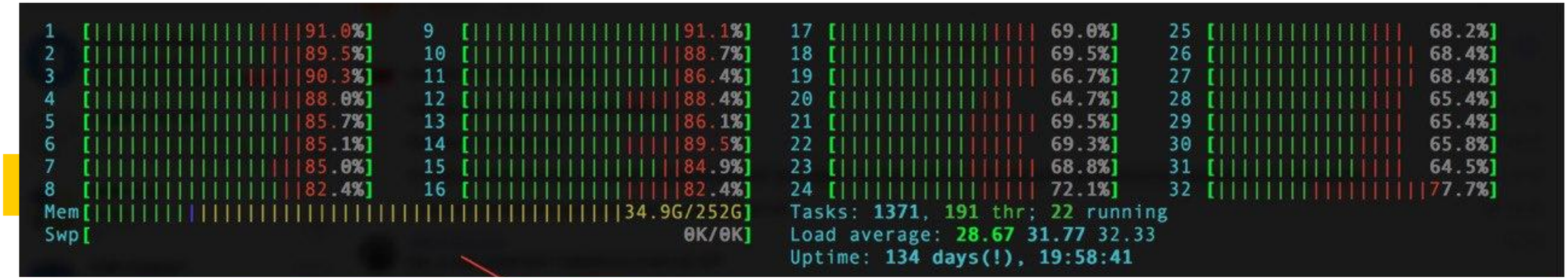
lz4


lz4/decode_amd64.s at master
https://github.com/pierrec/lz4/blob/master/decode_amd64.s
Reload current page (⌘R)
1 ock(dst, src []byte) int
20 // using 50 bytes of stack currently
21 TEXT ·decodeBlock(SB), NOSPLIT, $64-56
22 MOVQ dst_base+0(FP), DI
23 MOVQ DI, R11
24 MOVQ dst_len+8(FP), R8
25 ADDQ DI, R8
26
27 MOVQ src_base+24(FP), SI
28 MOVQ src_len+32(FP), R9
29 ADDQ SI, R9
30
31 // shortcut ends
32 // short output end
33 MOVQ R8, R12
34 SUBQ $32, R12
35 // short input end
36 MOVQ R9, R13
37 SUBQ $16, R13
38
39 loop:
40 // for si < len(src)
41 CMPQ SI, R9
42 JGE end
43
44 // token := uint32(src[si])

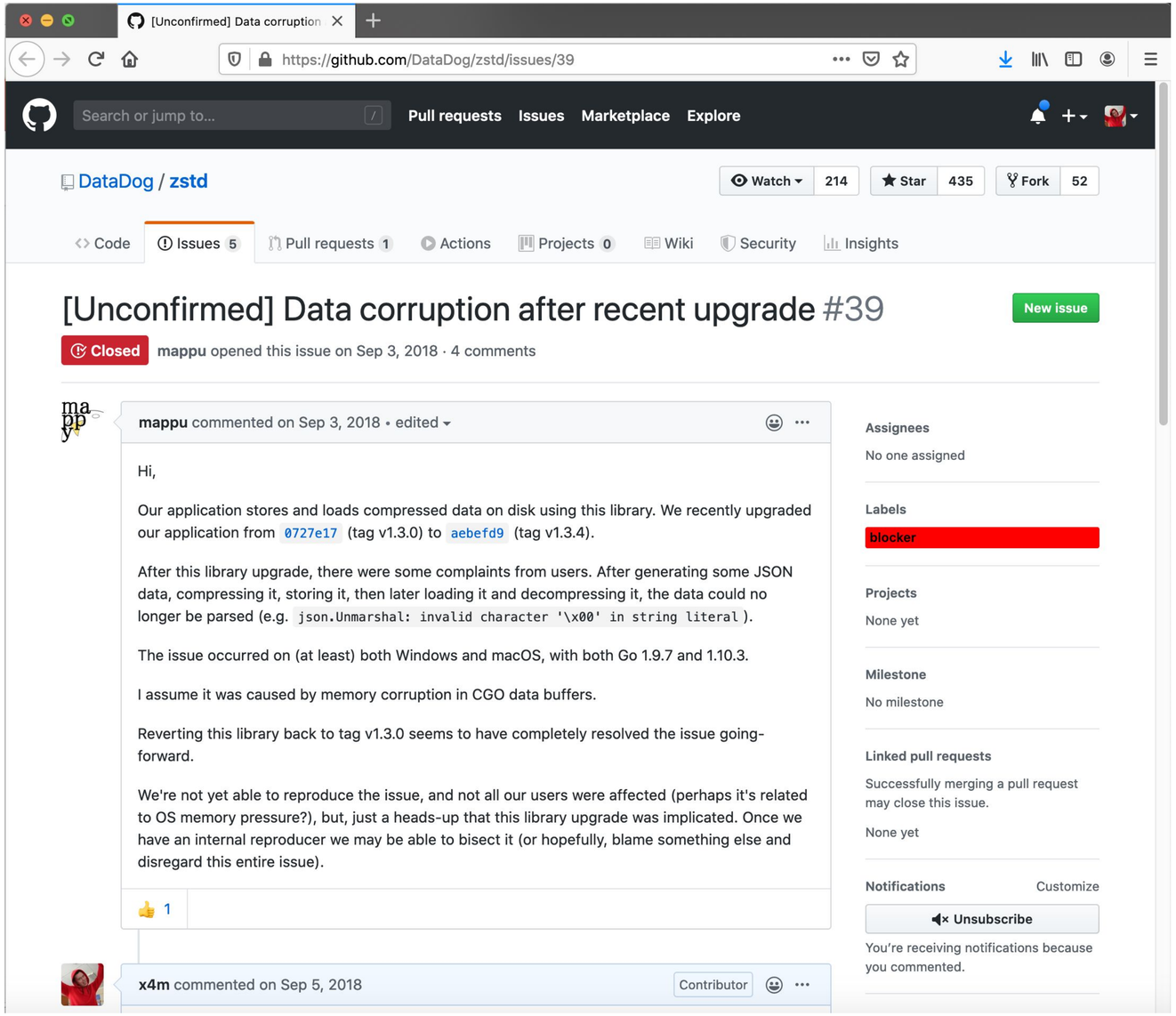
lz4


Fix short buffers in some specif
https://github.com/pierrec/lz4/commit/908ada6c279113e95a9a565b9f
Search or jump to...
Pull requests
Issues
Marketplace
Explore
pierrec / lz4
Watch 22
Star 389
Fork 83
Code
Issues 1
Pull requests 0
Actions
Projects 0
Wiki
Security
Insights
Fix short buffers in some specific cases
Browse files
master (#40)
v3.2.1 ... v2.1.1
x4m committed on Mar 26, 2019
1 parent 062282e
commit 908ada6c279113e95a9a565b9f5fd8c22fb61263
Showing 4 changed files with 4 additions and 0 deletions.
Unified
Split
3 reader.go
@@ -158,6 +158,9 @@ func (z *Reader) Read(buf []byte) (int, error) {
158 158 if debugFlag {
159 159 debug("reading block from writer")
160 160 }
161 + // Reset uncompressed buffer
162 + z.data = z.zdata[:cap(z.zdata)][len(z.zdata):]
163 +
161 164 // Block length: 0 = end of frame, highest bit set: uncompressed.
162 165 bLen, err := z.readUint32()
163 166 if err != nil {

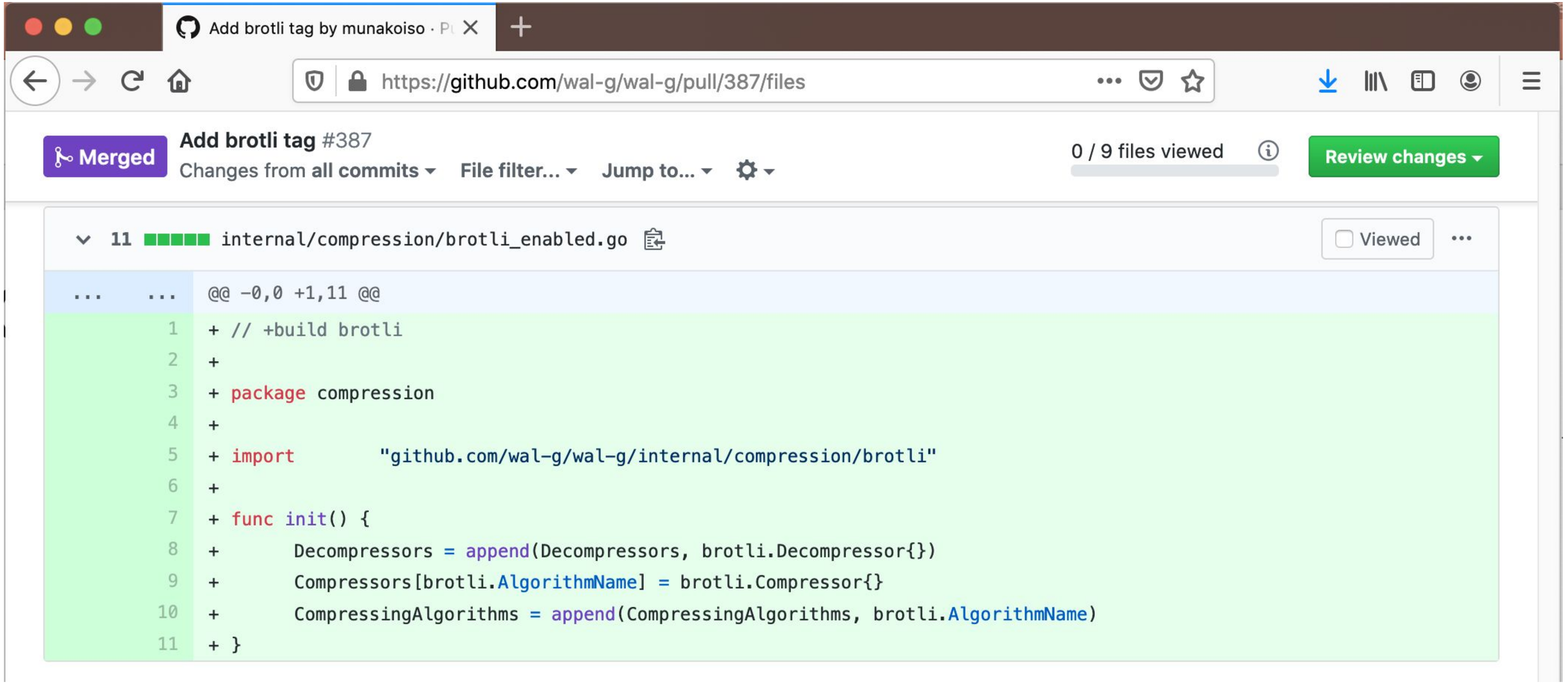
lzma


1 [||||||||||||||||||91.0%]
2 [||||||||||||||||||89.5%]
3 [||||||||||||||||||90.3%]
4 [||||||||||||||||||88.0%]
5 [||||||||||||||||||85.7%]
6 [||||||||||||||||||85.1%]
7 [||||||||||||||||||85.0%]
8 [||||||||||||||||||82.4%]
9 [||||||||||||||||||91.1%]
10 [||||||||||||||||||88.7%]
11 [||||||||||||||||||86.4%]
12 [||||||||||||||||||88.4%]
13 [||||||||||||||||||86.1%]
14 [||||||||||||||||||89.5%]
15 [||||||||||||||||||84.9%]
16 [||||||||||||||||||82.4%]
17 [||||||||||||||||| 69.0%]
18 [||||||||||||||||| 69.5%]
19 [||||||||||||||||| 66.7%]
20 [||||||||||||||||| 64.7%]
21 [||||||||||||||||| 69.5%]
22 [||||||||||||||||| 69.3%]
23 [||||||||||||||||| 68.8%]
24 [||||||||||||||||| 72.1%]
25 [||||||||||||||||| 68.2%]
26 [||||||||||||||||| 68.4%]
27 [||||||||||||||||| 68.4%]
28 [||||||||||||||||| 65.4%]
29 [||||||||||||||||| 65.4%]
30 [||||||||||||||||| 65.8%]
31 [||||||||||||||||| 64.5%]
32 [||||||||||||||||77.7%]
Mem[||||||||||||||||||||||||||||||||||34.9G/252G]
Swp[0K/0K]
Tasks: 1371, 191 thr; 22 running
Load average: 28.67 31.77 32.33
Uptime: 134 days(!), 19:58:41

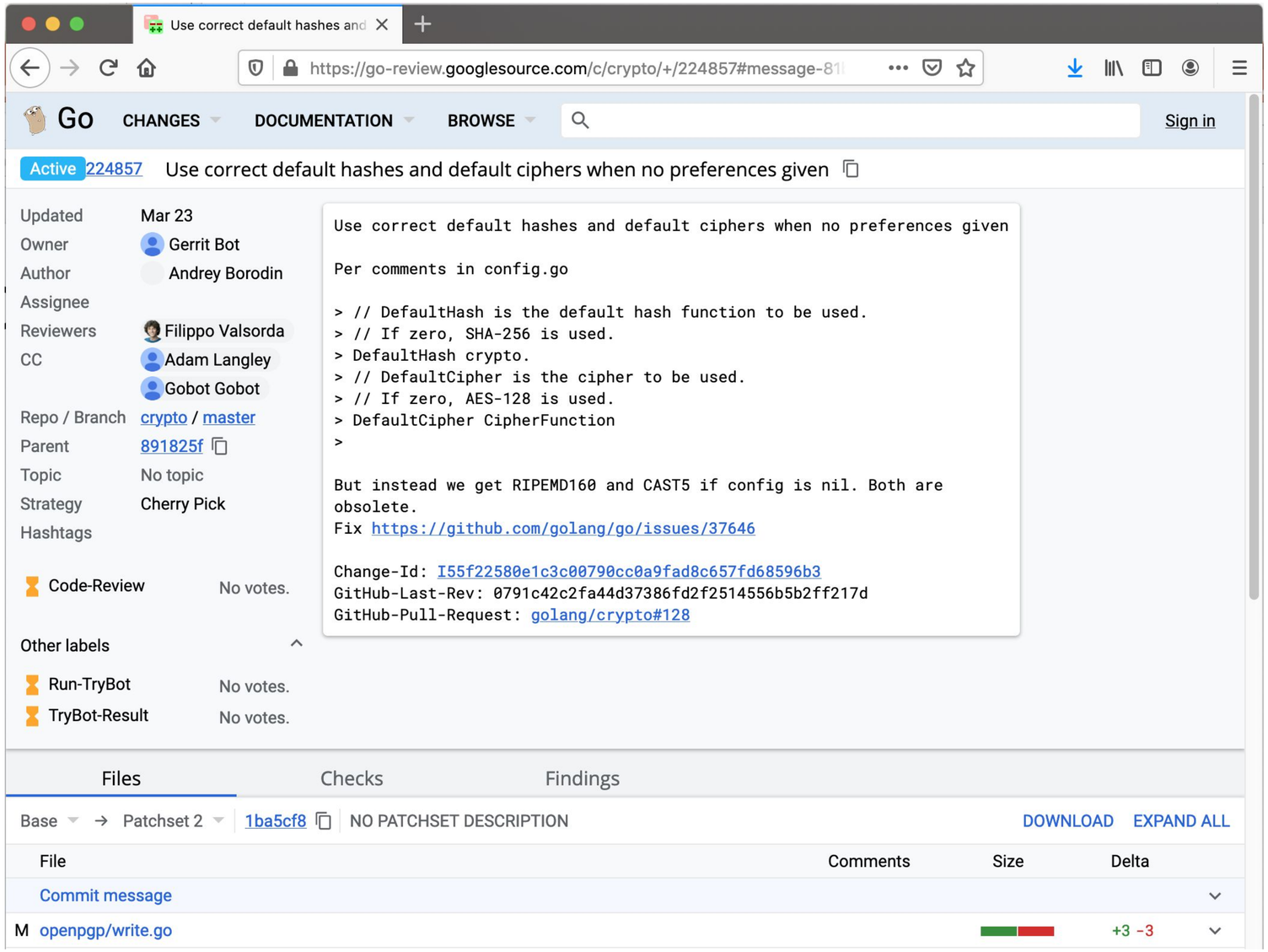
ZStd


[Unconfirmed] Data corruption
https://github.com/DataDog/zstd/issues/39
Search or jump to...
Pull requests
Issues
Marketplace
Explore
DataDog / zstd
Watch 214
Star 435
Fork 52
Code
Issues 5
Pull requests 1
Actions
Projects 0
Wiki
Security
Insights
[Unconfirmed] Data corruption after recent upgrade #39
New issue
Closed
mappu opened this issue on Sep 3, 2018 · 4 comments
mappu commented on Sep 3, 2018 • edited
Hi,
Our application stores and loads compressed data on disk using this library. We recently upgraded our application from 0727e17 (tag v1.3.0) to aebefd9 (tag v1.3.4).
After this library upgrade, there were some complaints from users. After generating some JSON data, compressing it, storing it, then later loading it and decompressing it, the data could no longer be parsed (e.g. json.Unmarshal: invalid character '\x00' in string literal).
The issue occurred on (at least) both Windows and macOS, with both Go 1.9.7 and 1.10.3.
I assume it was caused by memory corruption in CGO data buffers.
Reverting this library back to tag v1.3.0 seems to have completely resolved the issue going-forward.
We're not yet able to reproduce the issue, and not all our users were affected (perhaps it's related to OS memory pressure?), but, just a heads-up that this library upgrade was implicated. Once we have an internal reproducer we may be able to bisect it (or hopefully, blame something else and disregard this entire issue).
1
x4m commented on Sep 5, 2018
Contributor
Assignees
No one assigned
Labels
blocker
Projects
None yet
Milestone
No milestone
Linked pull requests
Successfully merging a pull request may close this issue.
None yet
Notifications
Customize
Unsubscribe
You're receiving notifications because you commented.

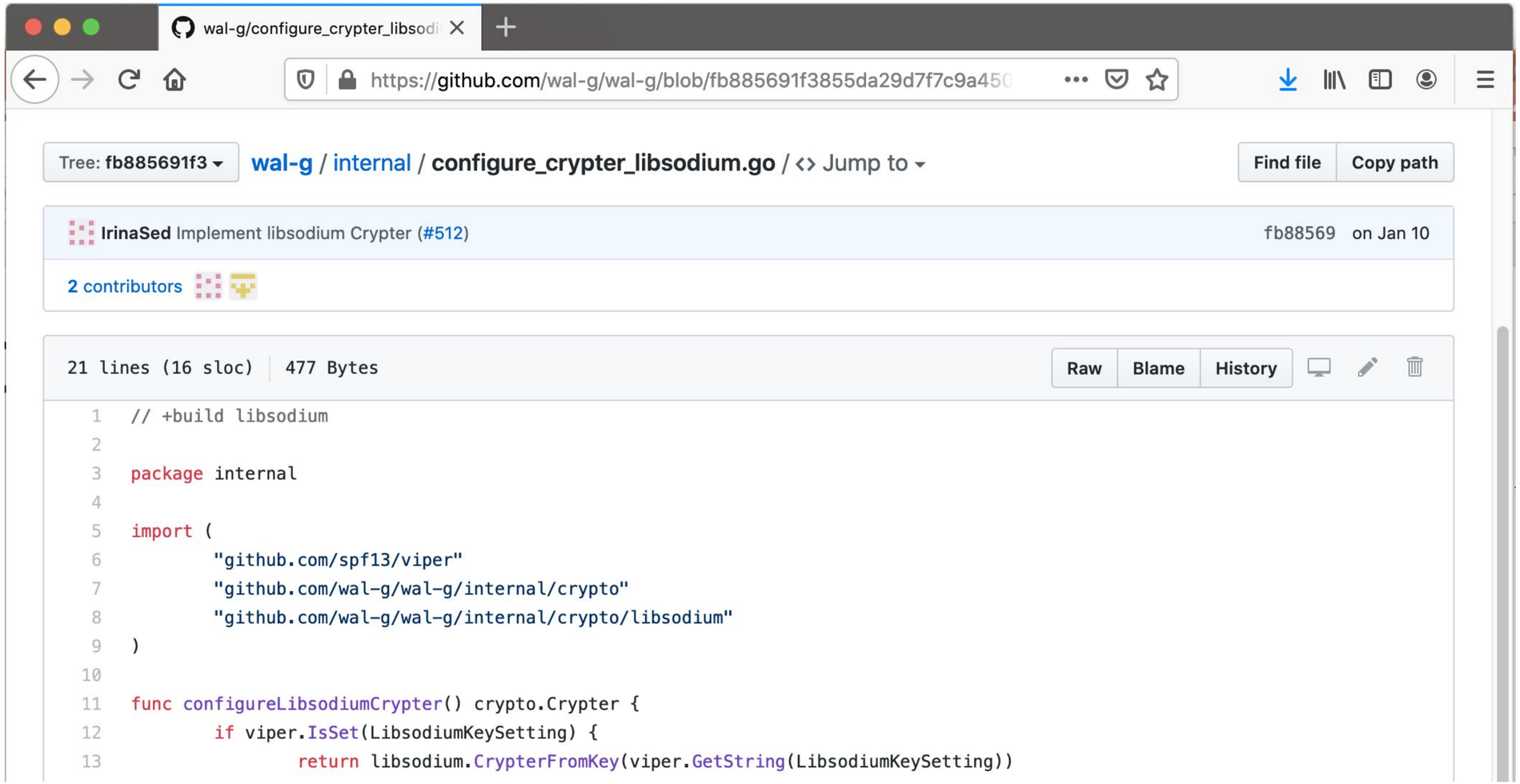
brotli


Add brotli tag by munakoiso · P
https://github.com/wal-g/wal-g/pull/387/files
Merged
Add brotli tag #387
Changes from all commits
File filter...
Jump to...
0 / 9 files viewed
Review changes
11 internal/compression/brotli_enabled.go
Viewed
@@ -0,0 +1,11 @@
+ // +build brotli
+
+ package compression
+
+ import "github.com/wal-g/wal-g/internal/compression/brotli"
+
+ func init() {
+ Decompressors = append(Decompressors, brotli.Decompressor{})
+ Compressors[brotli.AlgorithmName] = brotli.Compressor{}
+ CompressingAlgorithms = append(CompressingAlgorithms, brotli.AlgorithmName)
+ }

OpenPGP


Use correct default hashes and
https://go-review.googlesource.com/c/crypto/+/224857#message-81
Go
CHANGES
DOCUMENTATION
BROWSE
Sign in
Active 224857 Use correct default hashes and default ciphers when no preferences given
Updated Mar 23
Owner Gerrit Bot
Author Andrey Borodin
Assignee
Reviewers Filippo Valsorda
CC Adam Langley
Gobot Gobot
Repo / Branch crypto / master
Parent 891825f
Topic No topic
Strategy Cherry Pick
Hashtags
Code-Review No votes.
Other labels
Run-TryBot No votes.
TryBot-Result No votes.
Use correct default hashes and default ciphers when no preferences given
Per comments in config.go
> // DefaultHash is the default hash function to be used.
> // If zero, SHA-256 is used.
> DefaultHash crypto.
> // DefaultCipher is the cipher to be used.
> // If zero, AES-128 is used.
> DefaultCipher CipherFunction
>
But instead we get RIPEMD160 and CAST5 if config is nil. Both are obsolete.
Fix https://github.com/golang/go/issues/37646
Change-Id: I55f22580e1c3c00790cc0a9fad8c657fd68596b3
GitHub-Last-Rev: 0791c42c2fa44d37386fd2f2514556b5b2ff217d
GitHub-Pull-Request: golang/crypto#128
Files
Checks
Findings
Base → Patchset 2 1ba5cf8 NO PATCHSET DESCRIPTION
DOWNLOAD
EXPAND ALL
File
Comments
Size
Delta
Commit message
M openpgp/write.go
+3 -3

libsodium


wal-g/configure_crypter_libsodi
https://github.com/wal-g/wal-g/blob/fb885691f3855da29d7f7c9a450
Tree: fb885691f3
wal-g / internal / configure_crypter_libsodium.go / Jump to
Find file
Copy path
IrinaSed Implement libsodium Crypter (#512)
fb88569 on Jan 10
2 contributors
21 lines (16 sloc) 477 Bytes
Raw
Blame
History
1 // +build libsodium
2
3 package internal
4
5 import (
6 "github.com/spf13/viper"
7 "github.com/wal-g/wal-g/internal/crypto"
8 "github.com/wal-g/wal-g/internal/crypto/libsodium"
9)
10
11 func configureLibsodiumCrypter() crypto.Crypter {
12 if viper.IsSet(LibsodiumKeySetting) {
13 return libsodium.CrypterFromKey(viper.GetString(LibsodiumKeySetting))

Push-based vs Pull-based executer

io.Reader vs io.Writer

```
type Reader interface {
    Read(p []byte) (n int, err error)
}
```

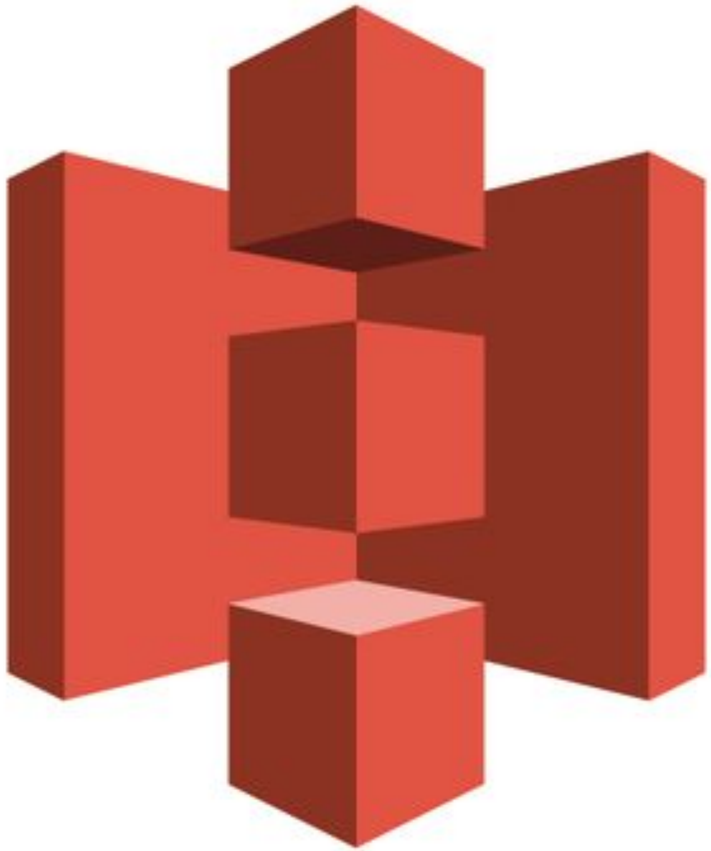
```
type Writer interface {
    Write(p []byte) (n int, err error)
}
```

io.ReadFull

```
func ReadAtLeast(r Reader, buf []byte, min int) (n int, err error) {
    if len(buf) < min : 0, ErrShortBuffer ↗
    for n < min && err == nil {
        var nn int
        nn, err = r.Read(buf[n:])
        n += nn
    }
    if n >= min {
        err = nil
    } else if n > 0 && err == EOF {
        err = ErrUnexpectedEOF
    }
    return
}
```

Store some bytes

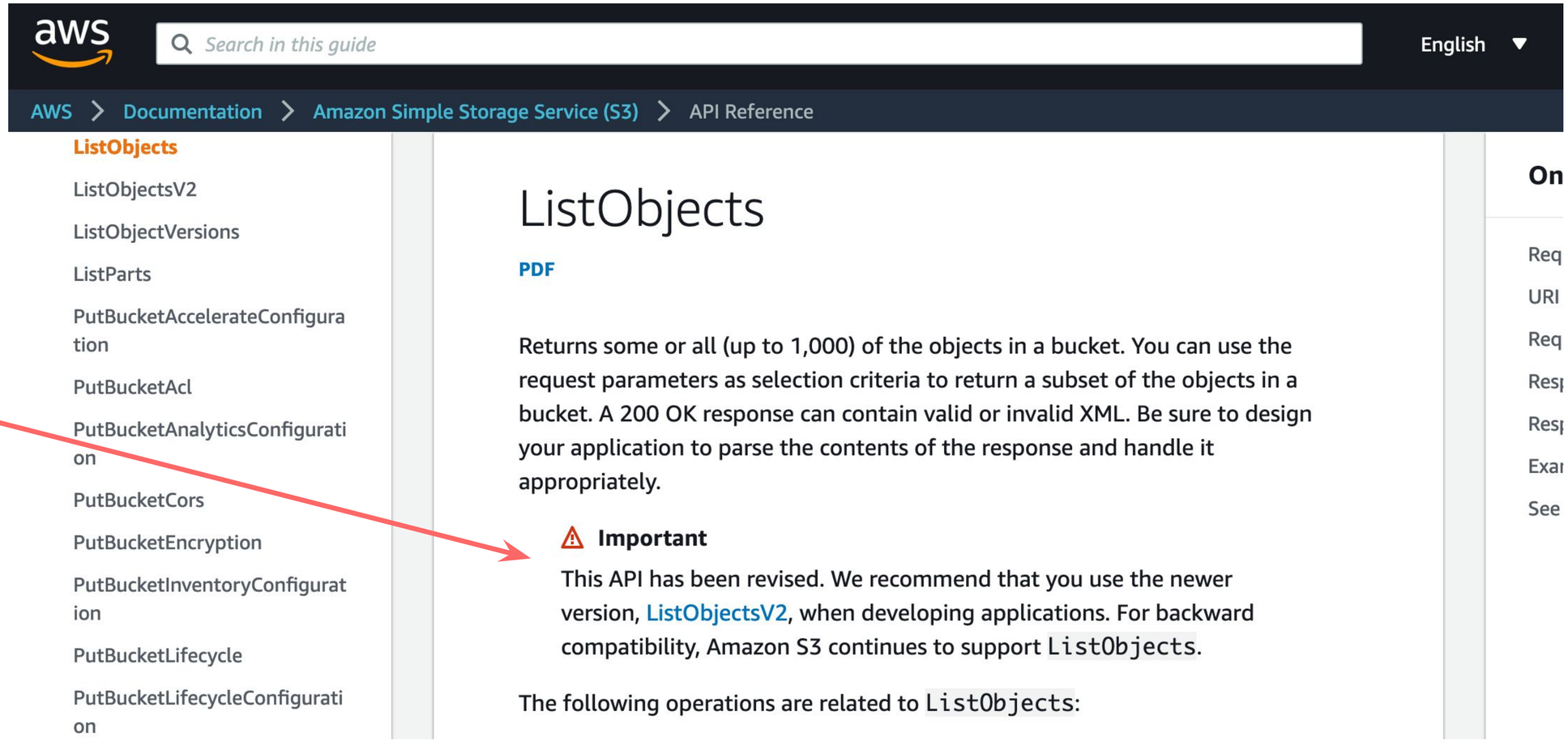
S3-based WAL-G



S3-based WAL-G

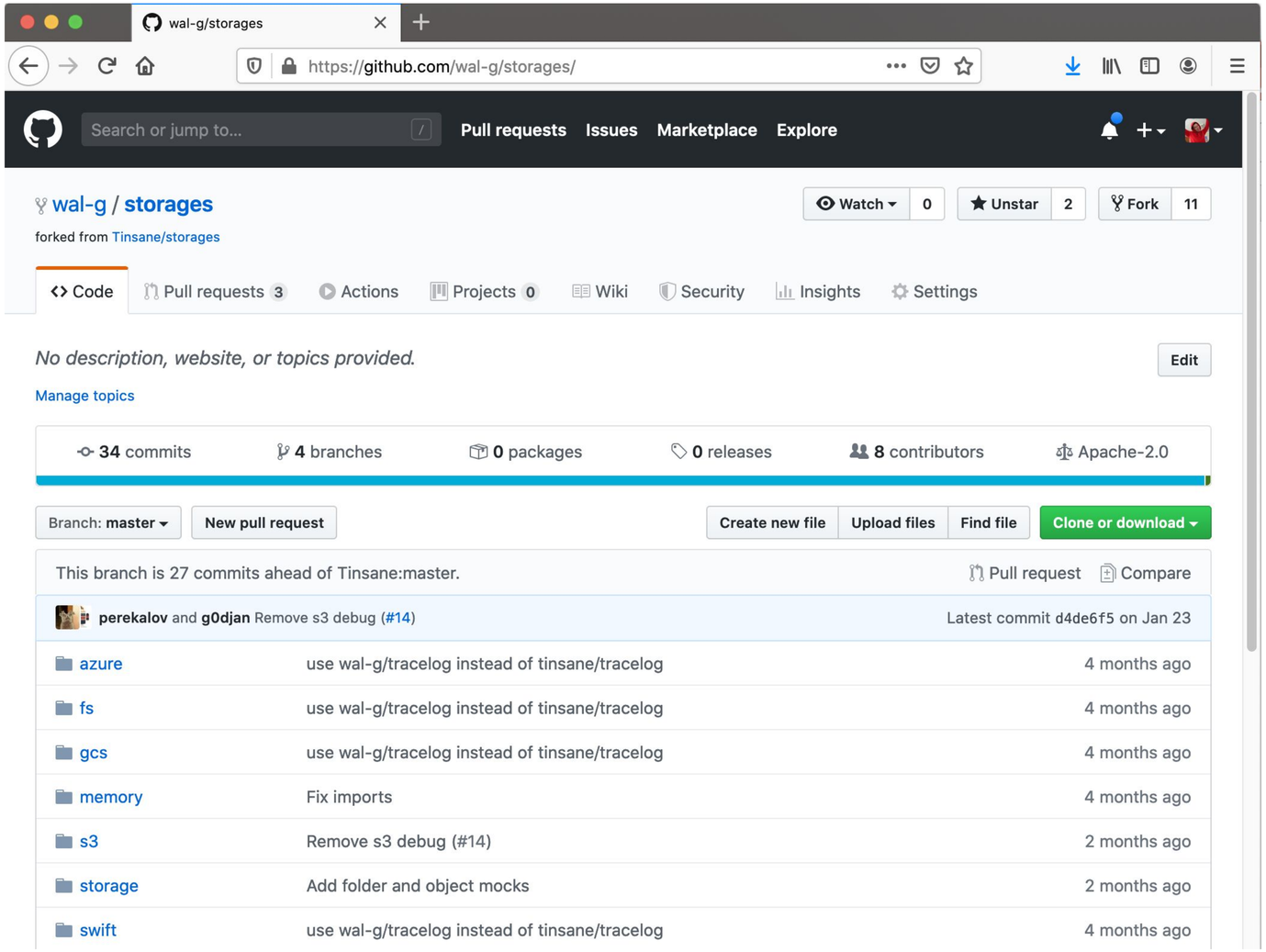
- AWS has it’s own cryptography

S3 ListObjectsV2()



S3 infiltrated the whole codebase

Refactor storages

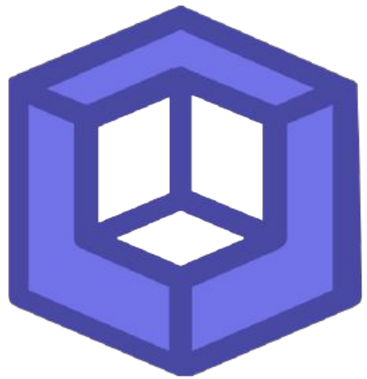


GCP

- No multipart uploads
- A lot of problems with retries

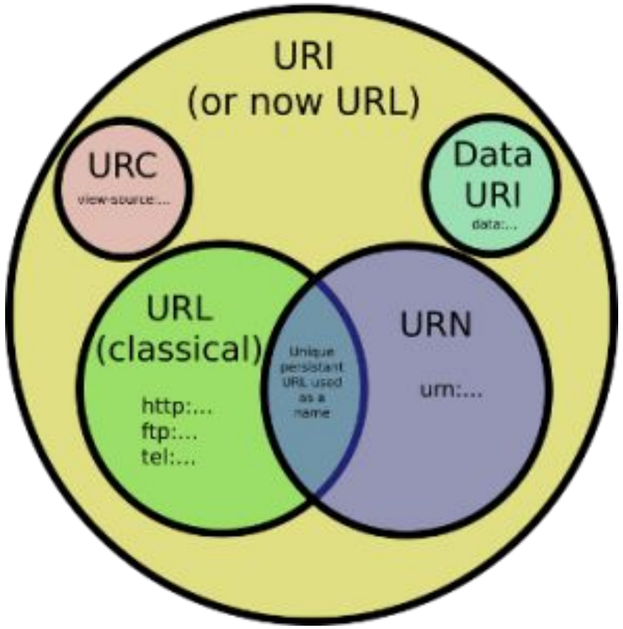
Azure

- Contribution from KubeDB



FS

- Problems with Windows build



SWIFT

No problems, does anyone use WAL-G + SWIFT?

SSH/SCP

- Because we need to back up S3 too

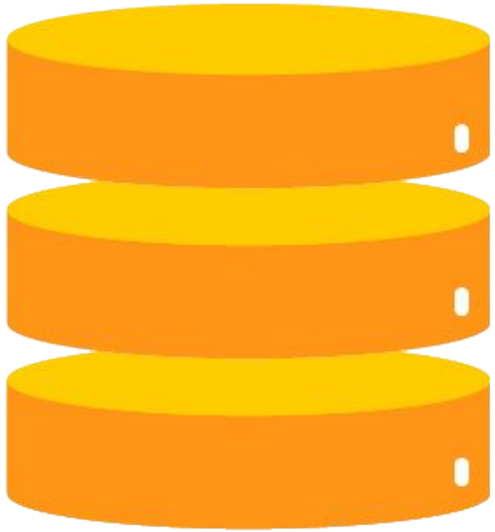
WAL-G for other databases

WAL-G for MySQL®

What to archive?

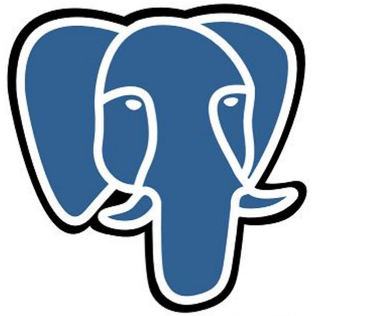
Data

Transactions log



Differences from PostgreSQL

- Storage engines
- Redo log is circular
- No archive command
- PITR is based on binlog, not redo



Data archiving

xtrabackup

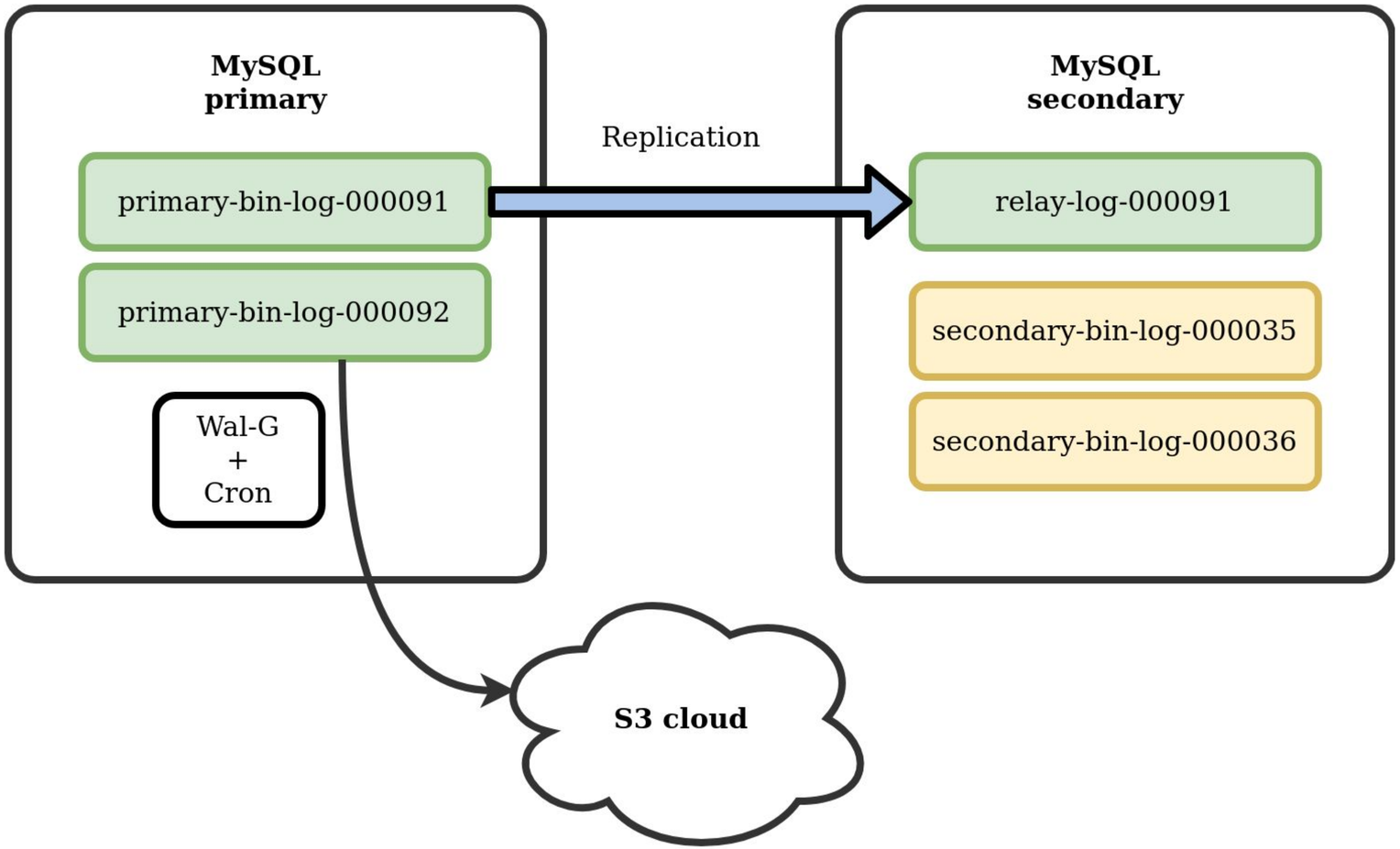
mysqldump

What is binlog?

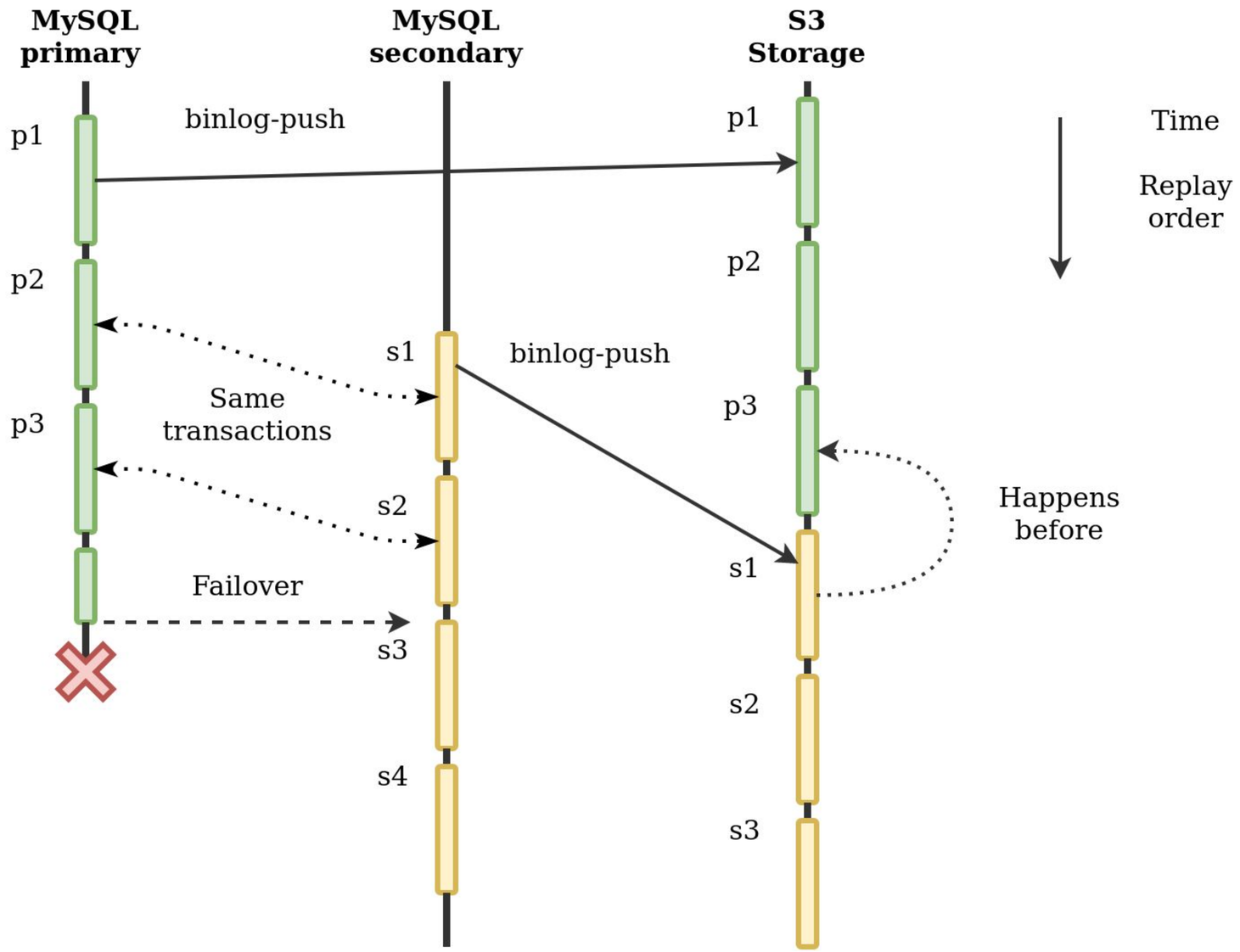
- Transactions (rows or queries)
- Timestamps
- GTID - unique ID

```
$ mysqlbinlog -v /var/lib/mysql/binlog.001242
....
SET @@SESSION.GTID_NEXT=
'63bebac7-c424-11e9-bb82-15ed63279c98:31214803';

TIMESTAMP=1619172002;

BEGIN
BINLOG '
opqCYBMCAAAAOgAAALoxHgAAAG0AAAAAAAEABW15c3
FsAAxtZGJfcmVwbF9tb24AAgMRAQMC+ajWYw==
opqCYB8CAAAAOgAAAPQxHgAAAG0AAAAAAAEAAgAC///8
AQAAAGCCmqEaNvwBAAAAYIKaohpA3Bx6WA==
 '
....
```

Binlog archivation



Binlog backup and replay order



Speed up replay

- Run MySQL on high port (3308)
- Disable all possible logging
- Don’t forget to put everything

  back into production!

```
double_write_buffer = OFF

innodb_flush_log_at_trx_commit = 0

sync_binlog = 1000

event-scheduler = OFF
```

Complete scenario

Back up

- Daily:

  wal-g backup-push

- Each minute:

  wal-g binlog-push

Restore

- wal-g backup-fectch LATEST
- Start mysql on high port
- wal-g binlog-replay

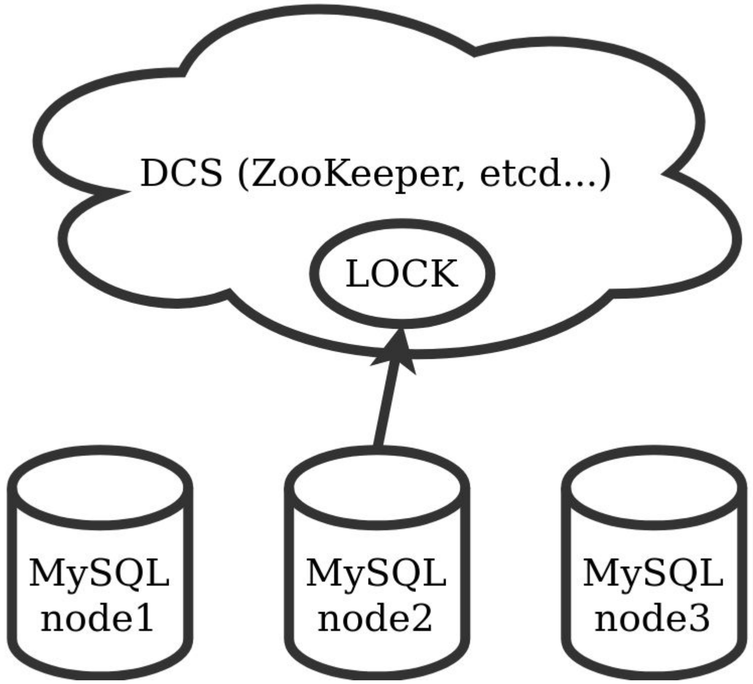
  --since LATEST

  --until 2021-05-12T12:00:00Z

- Restart MySQL with production config

Out of scope

- How to choose which node to back up?
- How to not upload binlog from a crashed master?
- How to handle binlog name collisions?



Does it work at all?

> 1k
clusters

> 1.5k
nodes

*In Yandex

WAL-G for Microsoft SQLServer™

SQLServer backup features

- Database backups
- Log backups
- Incremental backups
- Point-in-time recovery

Backup methods

- DISK
- TAPE
- VIRTUAL DEVICE
- URL (Azure)

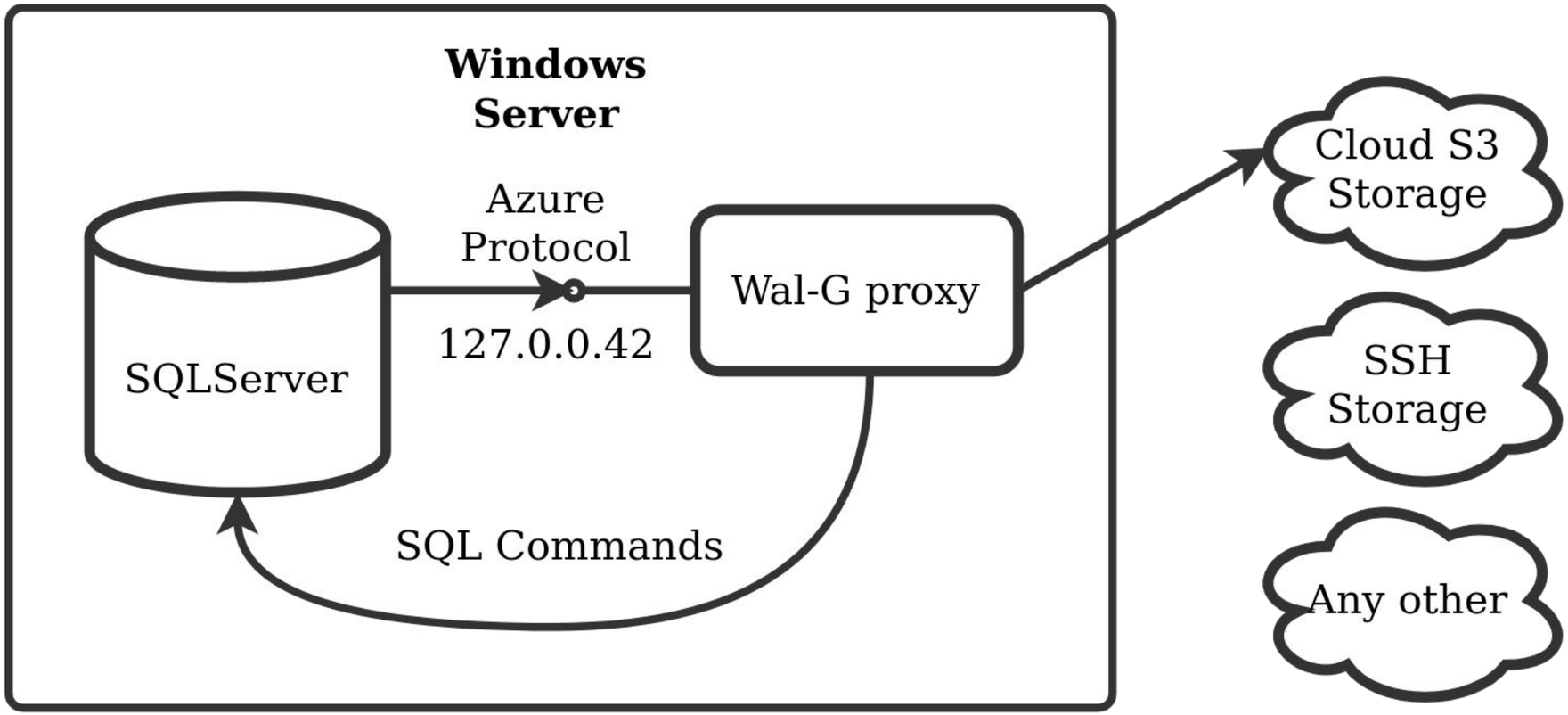
Backup to DISK

- Additional disk space required
- Additional IOPs required
- Still need to upload to storage

OR

- Additional network disk

WAL-G approach: emulate Azure



Features and hacks

1. Need domain name
2. Need HTTPS
3. Only default HTTPS port - 443
4. Azure credentials required

```
$ Add-Content -Path
    -Value '127.0.0.42 backup.local'
    -Path 'C:\Windows\System32\Drivers\etc\hosts'

$ Import-Certificate
    -CertStoreLocation cert:\LocalMachine\Root \
    -FilePath 'C:\backup.local.cert.pem'

$ Invoke-Sqlcmd "
  CREATE CREDENTIAL
 [https://backup.local/basebackups_005]
  WITH DENTITY='SHARED ACCESS SIGNATURE',
  SECRET = 'does_not_matter'
"
```

Case1: flexibility

1. Run WAL-G in background

```
wal-g --config c:\walg.yaml proxy
```

2. Now you have your own private Azure

```
BACKUP DATABASE [db1]
TO URL = 'https://walg.local/somepath/weekly/20210501'
```

Case 2: PG-like CLI

wal-g backup-push

wal-g backup-list

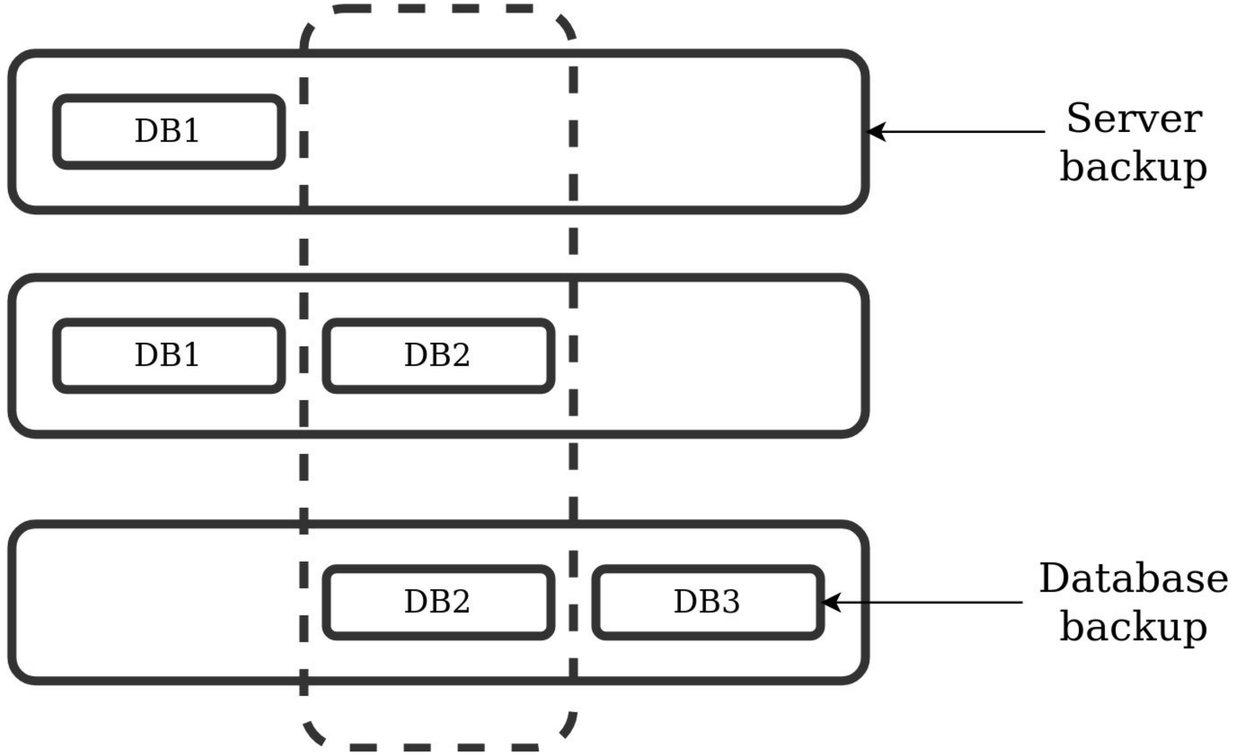
wal-g log-push

wal-g backup-restore LATEST

wal-g log-restore --since LATEST --until TS

Multiple databases

- backup-push
- backup-push -d db2
- log-restore
- log-restore -d db1,db2



Questions

Andrey Borodin

 x4mmm@yandex-team.ru

 x4mmm

Dmitry Smal

 mialinx@yandex-team.ru

 mialinx