SNIA DEVELOPER CONFERENCE
SDC 25
By Developers FOR Developers

Hyatt Regency Santa Clara, CA
September 15-17, 2025

War Stories from the Storage Trenches

Moving Data Across NFS, SMB, and S3

Carl D'Halluin (Datadobi) and Steve Leeper (Datadobi)

www.sniadeveloper.org

Speakers

Carl D'Halluin
CTO

Datadobi

Steve Leeper
VP Product Marketing

Datadobi

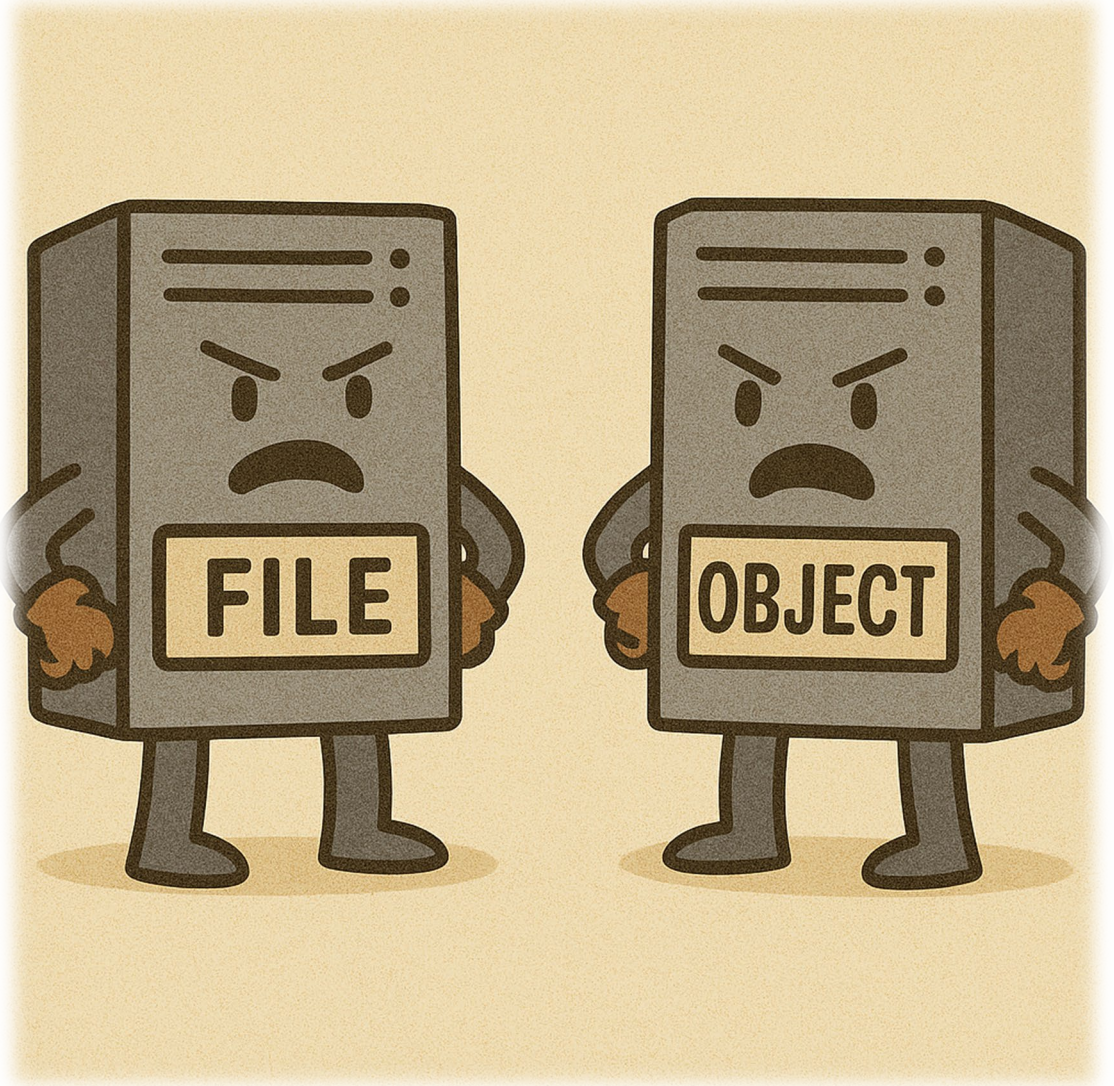
SDC 25

Overview

1. File and Object Storage
2. Why Copy Data?
3. How to Copy Data
4. Scenarios
5. War Stories
6. Lessons Learnt

Overview

1. File and Object Storage
2. Why Copy Data?
3. How to Copy Data
4. Scenarios
5. War Stories
6. Lessons Learnt



File and Object Storage

File Storage (~1980)

- File sharing, collaboration
  - High performance, often scale limitations
  - Scale getting better
- On-prem servers + a few cloud services
- NFS / SMB (Network-Attached Storage)
  - + FUSE or specialized FS drivers

Object Storage (~2000)

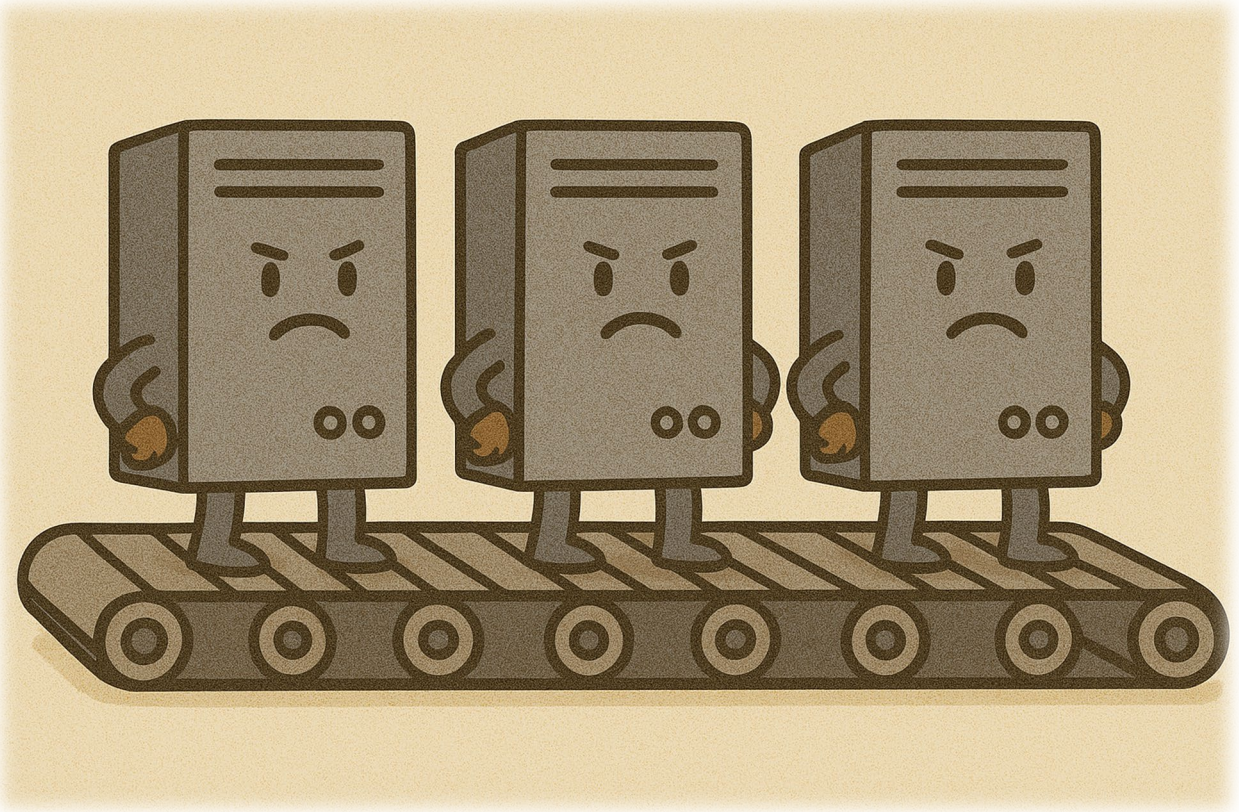
- Immutable objects + metadata
  - Low cost, low performance, large scale
  - But also "fast object"
- Cloud services + on-prem servers
- S3 emerged as a 'de facto standard'
  - + AzureBlob, GCS, …

There is more unstructured data in the enterprise:

- Specialized rich file services (OneDrive, Gdrive, Dropbox, Box, Sharepoint, ...)
- Application-captured unstructured data (e.g. Salesforce, Slack, Email, …)

Overview

1. File and Object Storage
2. Why Copy Data?
3. How to Copy Data
4. Scenarios
5. War Stories
6. Lessons Learnt



Why Copy Data?

- Data Move / Migrate
  - Vendor Swap
  - Technology Refresh
  - Datacenter Consolidation
  - Server/AD Consolidation
  - Cloud Adoption
  - Cloud Repatriation
  - Geopatriation
  - Free up primary storage
  - Data Relocation
  - Cleanup
- Data Replication
  - Often cross-vendor
  - Hybrid cloud
  - 3rd copy
- Data Archival
  - Often to tape-based object
- Data Copy
- AI Pipeline
- Data Backup
- Data Tiering

Overview

1. File and Object Storage
2. Why Copy Data?
3. How to Copy Data
4. Scenarios
5. War Stories
6. Lessons Learnt

Goal of Making a Data Copy

Copy files and objects between source and target storage.

Naive goal: “Target must be 100% identical to source”.

- ✓ File / object content
- ✓ Path, name
- ✓ Basic metadata (timestamps, ownership, permissions, S3 user-metadata)
- ✗ Other S3 metadata

  ETag, last modification time, version-id, history, storage class, retention, …
- ✗ Other file metadata

  change time, inode number, snapshots, SACL, stubs, tiering, size on disk, hard links, …

How to Copy/Migrate Data (1)

```
1. cp -r *.*
2. Point clients to target
```


How to Copy/Migrate Data (2)

```
1. cp -p -r *.*
2. Point clients to target
```


How to Copy/Migrate Data (3)

1. Make source read-only
2. cp -p -r *.*
3. Point clients to target

How to Copy/Migrate Data (4)

1. Do full copy (cp -r -p *.*)
2. Repeat [Do incremental copy]
3. Cutover:
   - Make source read-only
   - Final incremental copy
4. Point clients to target

How to Copy/Migrate Data (5)

1. Do full copy skipping recent changes
2. Repeat [Do incremental copy skipping recent changes]
3. Cutover:
   - Make source read-only
   - Final incremental copy (no skipping)
4. Point clients to target

How to Copy/Migrate Data (6)

1. Do full copy skipping recent changes
2. Repeat [Do incremental copy skipping recent changes]
3. Dry-run: Incremental copy (no skipping) + report
4. Plan Cutover
5. Execute Cutover:
   - Make source read-only
   - Final incremental copy (no skipping)
6. Point clients to target

How to Copy/Migrate Data (7)

1. Do full copy skipping recent changes
2. Repeat [Do incremental copy skipping recent changes]
3. (optional) Small test at target. Goto 2
4. Dry-run: Incremental copy (no skipping) + report
5. Plan Cutover
6. Execute Cutover:
   - Make source read-only
   - Final incremental copy (no skipping)
7. Point clients to target
8. Panic! Problems on target during production usage!
   - Copy-back changes from target to source
   - Point clients back to source
   - Goto 2

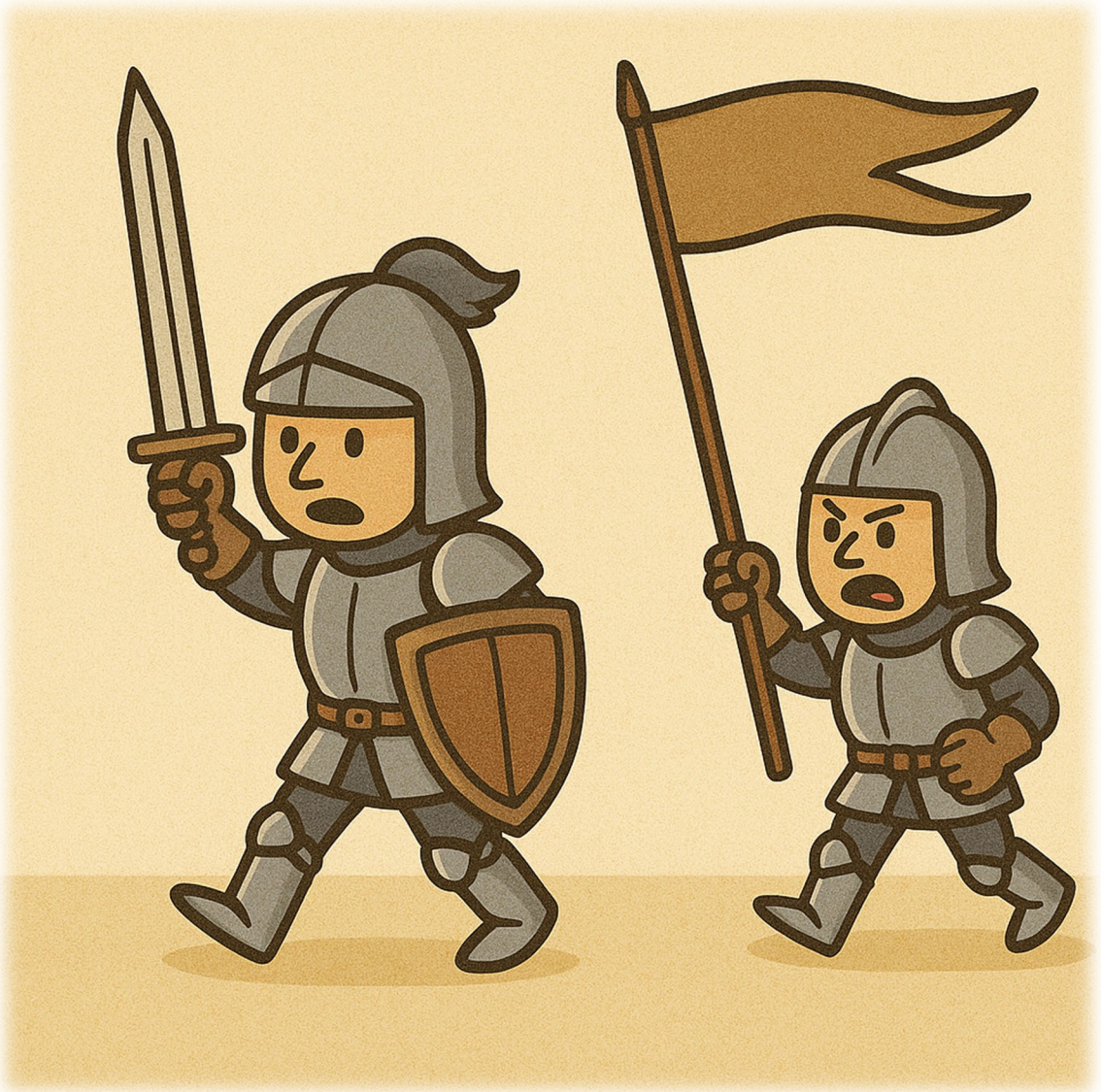
Data Copy/Migration - Summary

- And much more:
  - Pre-migration assessment & sizing
  - Filtering out unwanted files & objects
  - Selectively retrying specific failures
  - Parallelization & throttling
  - Copy data retention dates & legal holds
  - Mapping identities
  - “Chain of Custody” report
  - Resuming half-failed migrations
  - …

➔ Migration is a non-trivial non-linear non-fully-plannable process!

Overview

1. File and Object Storage
2. Why Copy Data?
3. How to Copy Data
4. Scenarios
5. War Stories
6. Lessons Learnt



Basic Scenarios

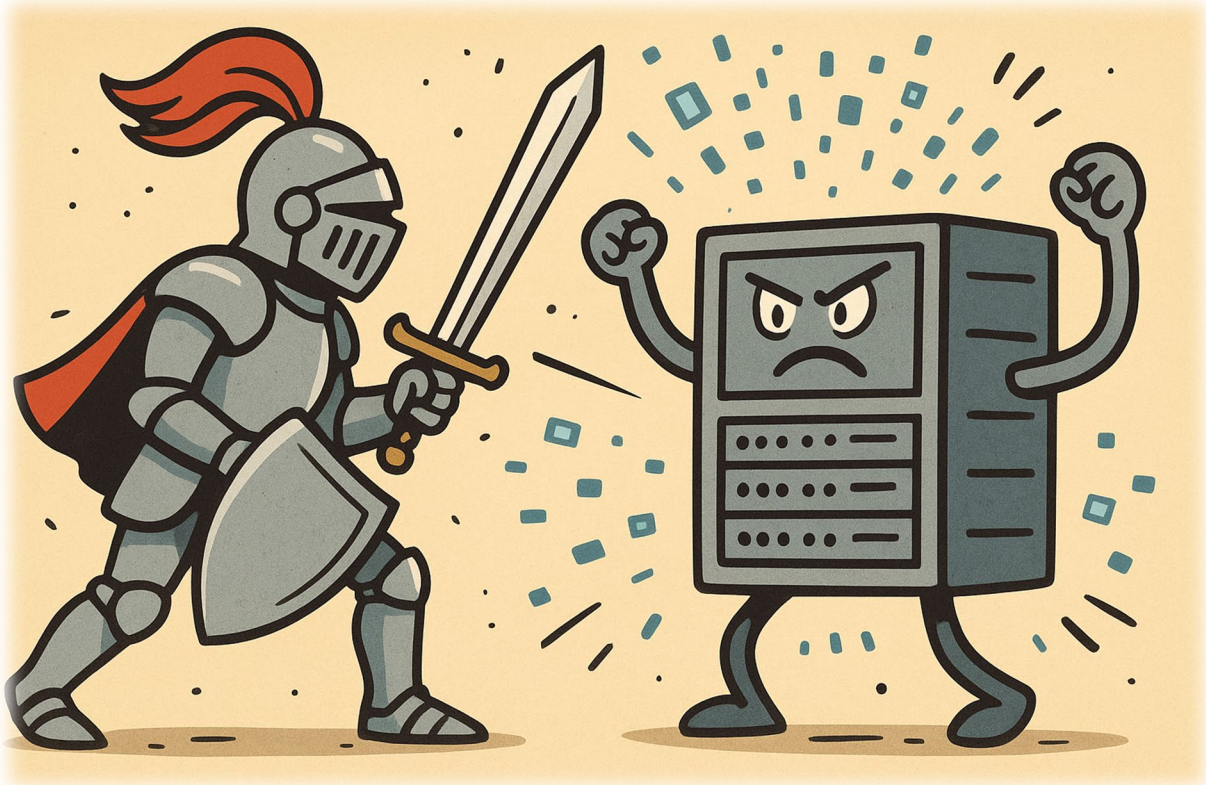
- Full server copy
- Cross-vendor
- Same-protocol
  - SMB → SMB
  - NFS → NFS
  - S3 → S3

Advanced Scenarios

- SMB+NFS → SMB+NFS
  - *"multi-protocol NAS"*
- File gateways
- Locked data: retention & legal hold
- Granular data reorganization
- Protocol change
  - File → Object → File
  - (S3 → AzureBlob → S3)

Overview

1. File and object storage
2. Why copy data?
3. How to copy data
4. Scenarios
5. War stories
6. Lessons learnt



War Stories – Protecting Customers Against Themselves

- Sanity checks before you start
  - Non-empty target
  - "Automatic" retention (lock) on target
- Why do you need a formal cutover?
  - A "quick" write test against target: can screw up synchronization
  - Applications or users pointing at target before formal cutover is finished
  - Data loss…
  - Split brain…

War Stories - Configuration

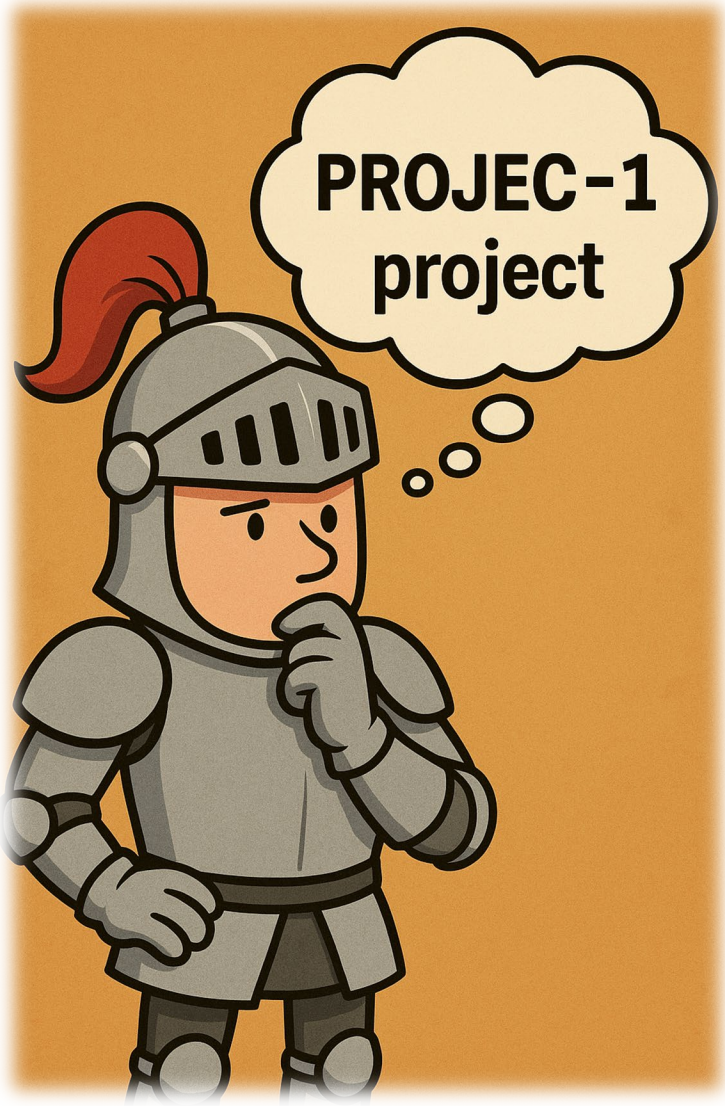
- Accessing the wrong server…
  - Source and target
  - Misconfigured proxies
  - DNS fun
    `https://s3.company.net/bucket/`
    `https://bucket.s3.company.net/`
- Insufficient Access rights
  - E.g. cannot access subdir
- Credentials
  - Mistyped passwords and access keys
  - App team not willing to share credentials

War Stories - Timestamps

- Make target identical to source?
  - Target timestamp changes post-write (antivirus?)
  - Lifecycle acting on target object
  - Listing items vs getting item metadata (stat, HEAD)
  - Timestamp granularity
    - `12:12:01.332` vs `12:12:02`
  - Cannot set S3 modification time, or file `ctime`
- Heisenberg Observer Effect: source `atime`
- Can we trust `mtime`
  - Open an MS Word doc but do not make any changes.
  - File content is modified but `mtime` is reset to its original value

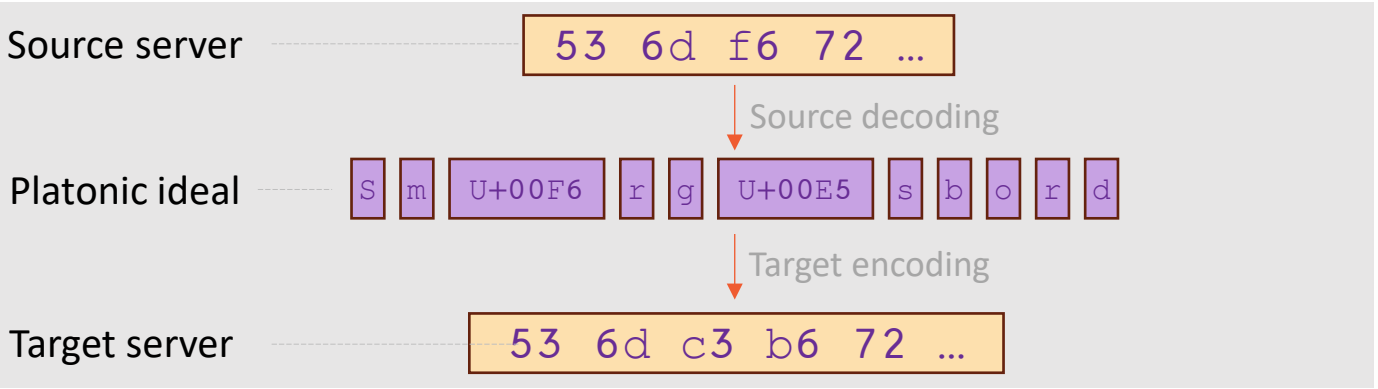
War Stories – Names

- Inconsistent length limits
  - Files: NAME_MAX, PATH_MAX, MAX_PATH
  - S3: 1024 UTF-8 BYTES
  - Azure Blob: 254 (or 63) "path segments"
- Illegal characters
  - \ / : * ? " < > | 0x00
- Illegal names
  - SMB
    - CON, PRN, NUL, NUL.txt, AUX, CLOCK$, COM1…9, LTP1…9
    - "file.", "file ", ".", ".."
  - S3
    - too/many/../../../dotdot/segments
    - object_key_ending_in_slash/
    - soap
- SMB Case sensitivity & "8.3 names"
    - project, PROJECT, Project/
    - PROJEC~1, PROJEC~2, … depends on creation order



War Stories – Name Encoding

- Seen this before?
  - Smörgåsbord - SmörgÍsbord
  - Résultat_élève.xls - Rщsultat щlшve.xls
  - Müller - M?ller - MÃ¼ller - M�ller



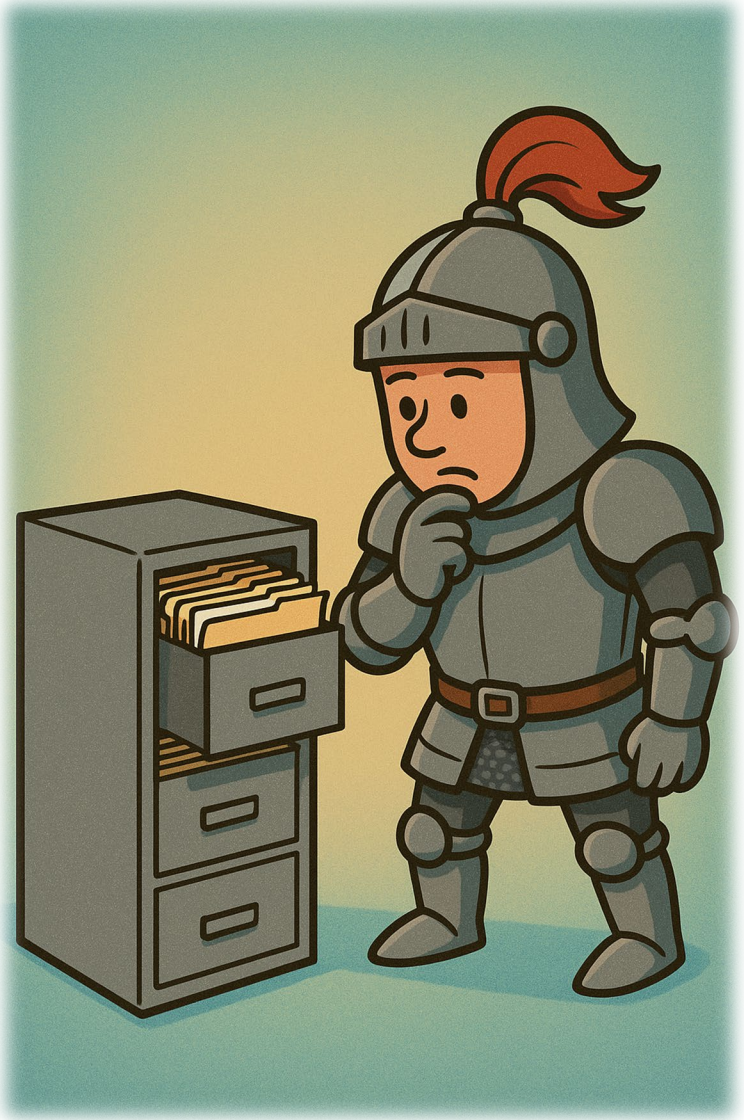
- Encoding ISO-8859-1, UTF-16, UTF-8, …

  "UTF-16 is a dumb useless hack. UTF-8 is a brilliant useful hack."

  - But UTF-8 is not perfect
    - BOM, overlong encoding, 0x00
    - "Problematic code points" RFC-9839
    - *WTF-8 to the rescue ?*

War Stories – File Storage

- Annoying limitations
  - Number of files per dir
  - Number of inodes
- Mutable & Locked files
- NAS gateways / tiered solutions
  - Rehydration doesn't work at scale
  - ➔ Use gateway for metadata, bypass it for data



War Stories – Object Storage

- HTTP 200 is not always a success

War Stories – Object Storage

```
GET / HTTP/1.1
Host: bucket.s3.company.net
```

```
HTTP/1.1 200 OK

<?xml version="1.0" encoding="UTF-8"?>
<ListBucketResult>
<Contents>
    <Key>sample1.txt</Key>
    <LastModified>2024-11-04T19:39:04.000Z</LastModified>
    <Size>22</Size>
</Contents>
<Contents>
    <Key>sample2.txt</Key>
    <LastModified>2024-11-04T19:39:04.000Z</LastModified>
    <Size>22</Size>
</Contents>
…
</ListBucketResult>
```


War Stories – Object Storage

```
GET / HTTP/1.1
Host: bucket.s3.company.net
```

```
HTTP/1.1 200 OK

<?xml version="1.0" encoding="UTF-8"?>
<ListAllMyBucketsResult>
<Buckets>
  <Bucket>
    <Name>engineering</Name>
    <CreationDate>2025-0526T19:37:25.000Z</CreationDate>
  </Bucket>
  <Bucket>
    <Name>marketing</Name>
    <CreationDate>2024-11-09T10:20:55.000Z</CreationDate>
  </Bucket>
  ...
</ListAllMyBucketsResult>
```


War Stories – Object Storage

```
GET / HTTP/1.1
Host: bucket.s3.company.net
```

```
HTTP/1.1 200 OK

<html>
  <body>
    <h1>Welcome to Intranet</h1>
    <p>Welcome all!</p>
  </body>
</html>
```


War Stories – Object Storage

```
POST /?delete HTTP/1.1
Host: bucket.s3.company.net

<Delete>
  <Object><Key>sample1.txt</Key></Object>
  <Object><Key>sample2.txt</Key></Object>
</Delete>
```

```
HTTP/1.1 200 OK

<?xml version="1.0" encoding="UTF-8"?>
<DeleteResult xmlns="http://s3.amazonaws.com/doc/2006-03-01/">
  <Deleted>
    <Key>sample1.txt</Key>
  </Deleted>
  <Error>
    <Key>sample2.txt</Key><Code>AccessDenied</Code>
  </Error>
</DeleteResult>
```


War Stories – Object Storage

- HTTP 200 is not always a success
- Better safe than sorry

War Stories – Object Storage

Suppose AWS S3 introduces a new object “subresource” such as tagging.
Some S3 application uses that feature against an ‘old’ S3-compatible server.

```
PUT /important_doc?newfeature HTTP/1.1
Host: bucket.s3.company.net

<?xml version=”1.0” encoding=”UTF-8”?>
<NewFeature>
    <SomeConfig />
</NewFeature>
```

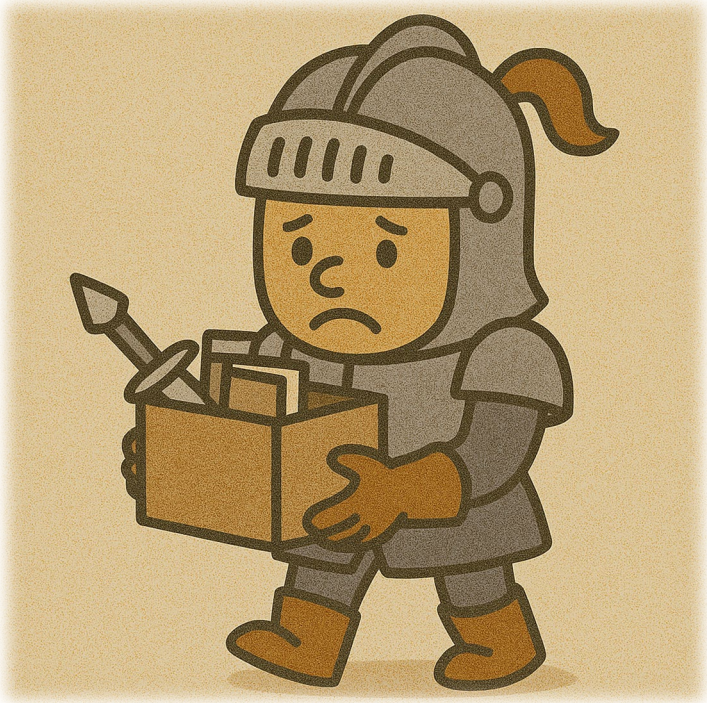
```
HTTP/1.1 200 OK
```

```
GET /important_doc HTTP/1.1
Host: bucket.s3.company.net
```

```
HTTP/1.1 200 OK

<?xml version=”1.0” encoding=”UTF-8”?>
<NewFeature>
    <SomeConfig />
</NewFeature>
```

Data loss… Caused by ignoring unknown query string params, or request headers

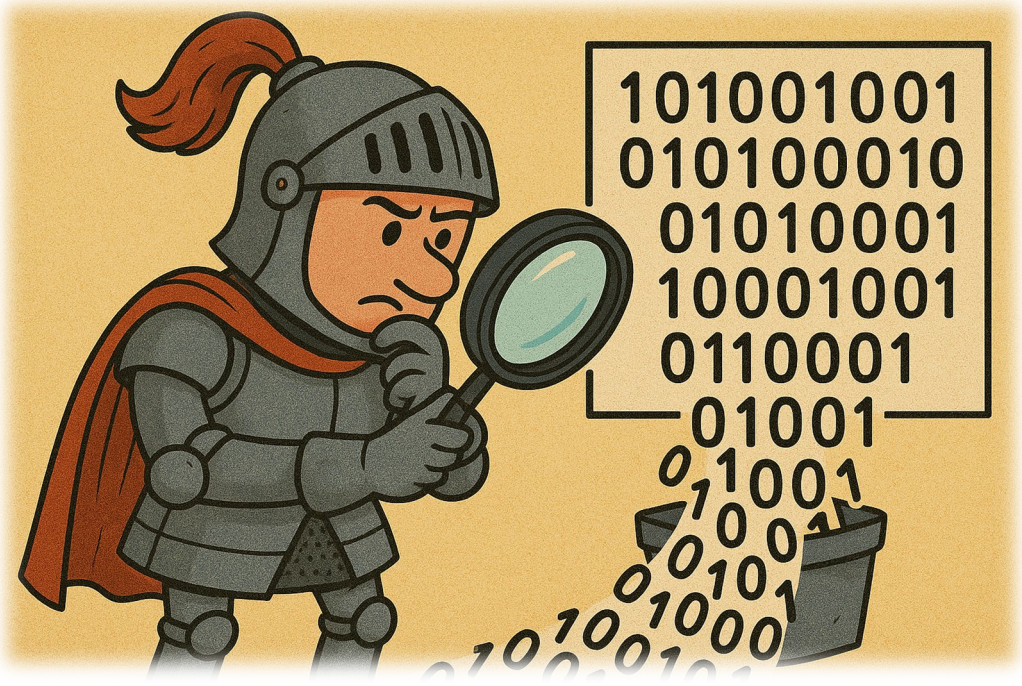


War Stories – Object Storage

- HTTP 200 is not always a success
- Better safe than sorry
- HTTP HEAD requests
  - No (error) response body
- HTTP response status code
  - Create → 200 or 201 ?
  - Delete → 200 or 204 or 404 ?
- Beware
  - ETag is not (always) the content MD5
  - S3 Version-id and timestamp not settable

War Stories – Content

- Content size limits
- Metadata size limits
- (SMB) Alternate Data Streams
- Eventual consistency
  - Not present in listing but can access
  - Present in listing but FileNotFound
  - Outdated content
- Trust but verify
  - Read back what you just wrote to target

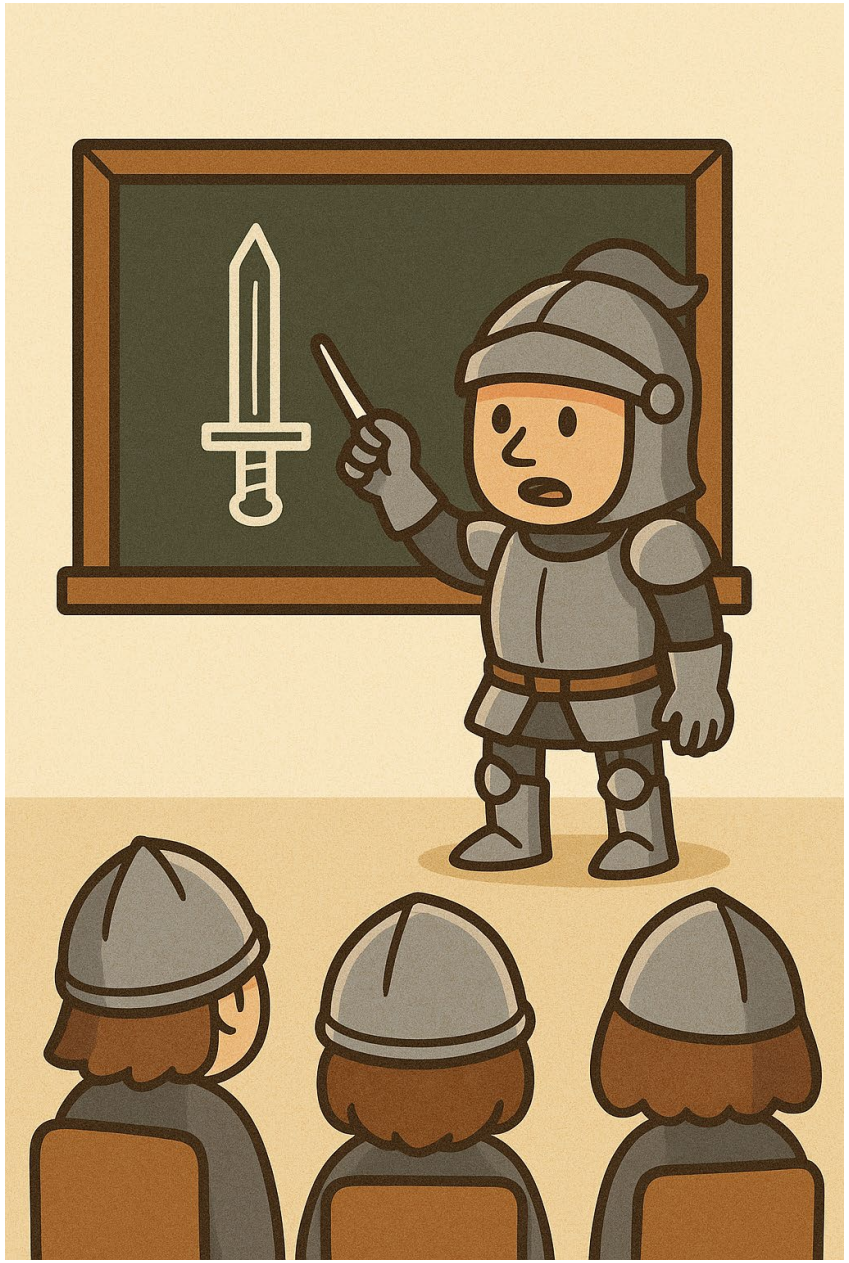


War Stories – Multi-Protocol

- NFS/SMB multi-protocol – Case sensitivity
  - NFS: `file.txt and FILE.TXT` - SMB: read `FILE.TXT`
  - Could get worse: `DIR/ and dir/ and DIR~1/`
- Single item becomes multiple files/objects
  - `my/cool/document.txt`
- Cannot copy object back to file
  - `my/cool/project/`
  - `my/cool/project` vs `my/cool/project/file.txt`

Overview

1. File and object storage
2. Why copy data?
3. How to copy data
4. Scenarios
5. War stories
6. Lessons learnt



Lessons Learnt – Developers of Data Mobility Software

- Defensive coding. Trust no one. Double check everything:
  - Server bugs
  - Ill-defined protocols
  - Process mistakes, people making errors
- Support old storage systems.
  - Don’t expect vendor bug fixes.
- Support server-specific quirks
- Read specs & docs
  - Understand how things work *in theory*
- Read again specs & docs
  - Be creative, get in the mind of the server developer, prevent mistakes

Lessons Learnt – Storage System Developers

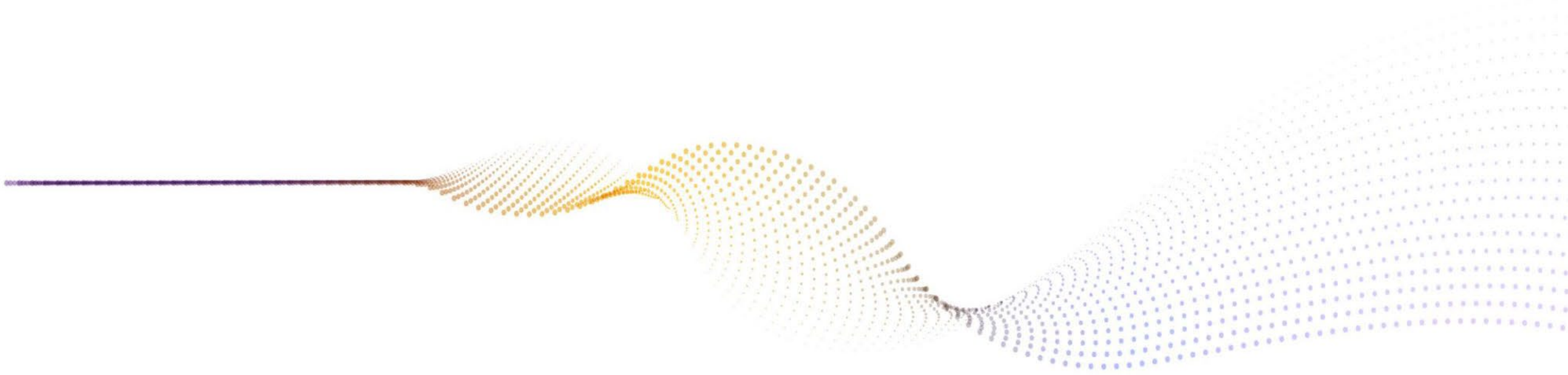
- Rich logging, rich error responses
- Sanitize your inputs
  - Refuse everything you don't support (request headers, query string params, XML request body elements, …)
  - Loosen up using whitelists
- Think of data mobility!
  - Settable S3 credentials
  - Settable S3 version-id
  - One day cutover could be as simple as a DNS change
- Fast way to get full + incremental inventory of all files/objects
  - Offer a way to detect that something DIDN'T change (changetime, version-id, LastModified)
  - Don't forget metadata changes (e.g. for S3 retention changes or object tags)
- Semantics of S3 dialects (S3-RDMA ?)
- Join SNIA collaboration efforts – SMB and S3 plugfests; compatibility test suite

Lessons Learnt - Application Developers

- Test against various storage systems
- Use safe characters, short names, compact metadata, bounded-size content
- Do not assume the server understands your requests
  - Process both error and success responses diligently
- S3
  - Do not use custom HTTP headers, query string params
  - When using shiny new features (e.g. conditional PUT), provide a fallback code path
  - Do not (only) rely on versionid or lastmodified timestamp. Make sure you can recover / rebuild.

Lessons Learnt – Storage Administrators

- Test your application with your storage system (or ask storage vendor)
- Understand your scaling and performance requirements
- Think of data mobility from day one
  - Especially if you consider gateways or tiering solutions
- Migration, replication, data copying
  - Requires careful planning
  - Requires dedicated software and methodology. It is not just “running a script”.



Thank you for attending!

Please remember to rate this session. You get access the presentations at
http://sniadeveloper.org/conference

