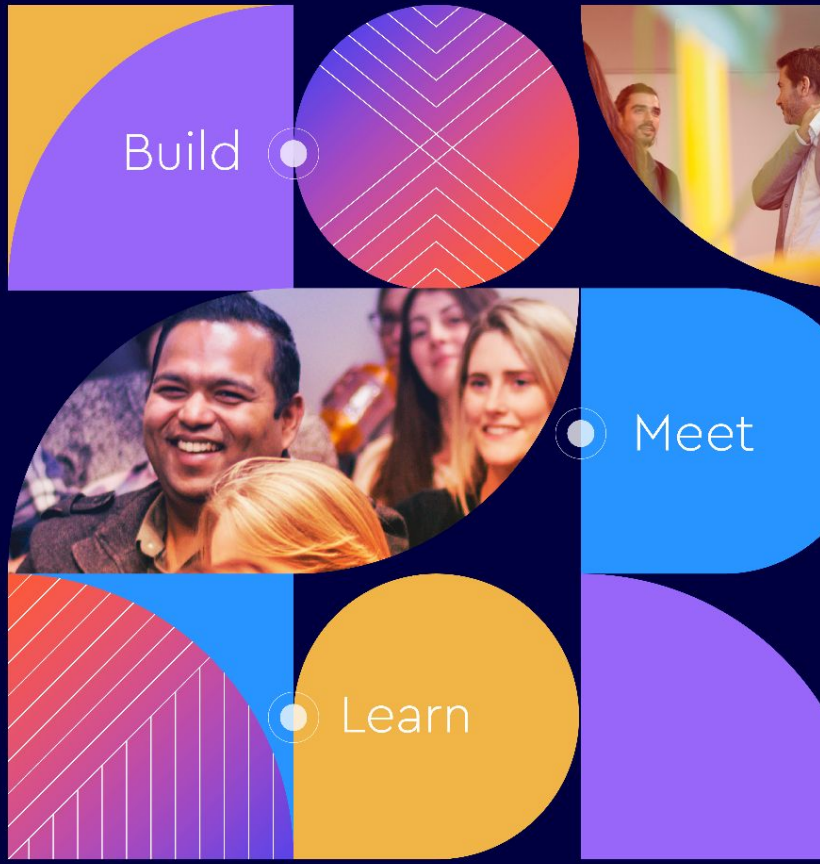


Backup and Restore

Learn Best Practices for a Distributed Database Backup Strategy

Purpose of Backups

Business/Application Continuity

- Traditional relational databases maintain a single copy of the data while online. Backups are conducted to maintain business continuity in the event of a disaster.
- Since YugabyteDB maintains at least 3 copies of the data while online, the focus shifts to application continuity and extreme disaster situations.
- YugabyteDB's backups are primarily for application corruption situations or if the majority of copies of a tablet are lost.

Backup/Restore in YugabyteDB

Features

- Can be run while the database is online
- Topologically independent backup format
- Nodes are backed up/restored in parallel
- Choice of file store targets:
  - AWS S3 (and compatibles like MinIO,IBM Cloud,Nutanix Buckets, etc)
  - Google GCS
  - Azure Blob Store
  - NFS (e.g. Filers, DataDomain, Linux FS)
- Table/Keyspace level backups for YCQL
- Database level backups for YSQL

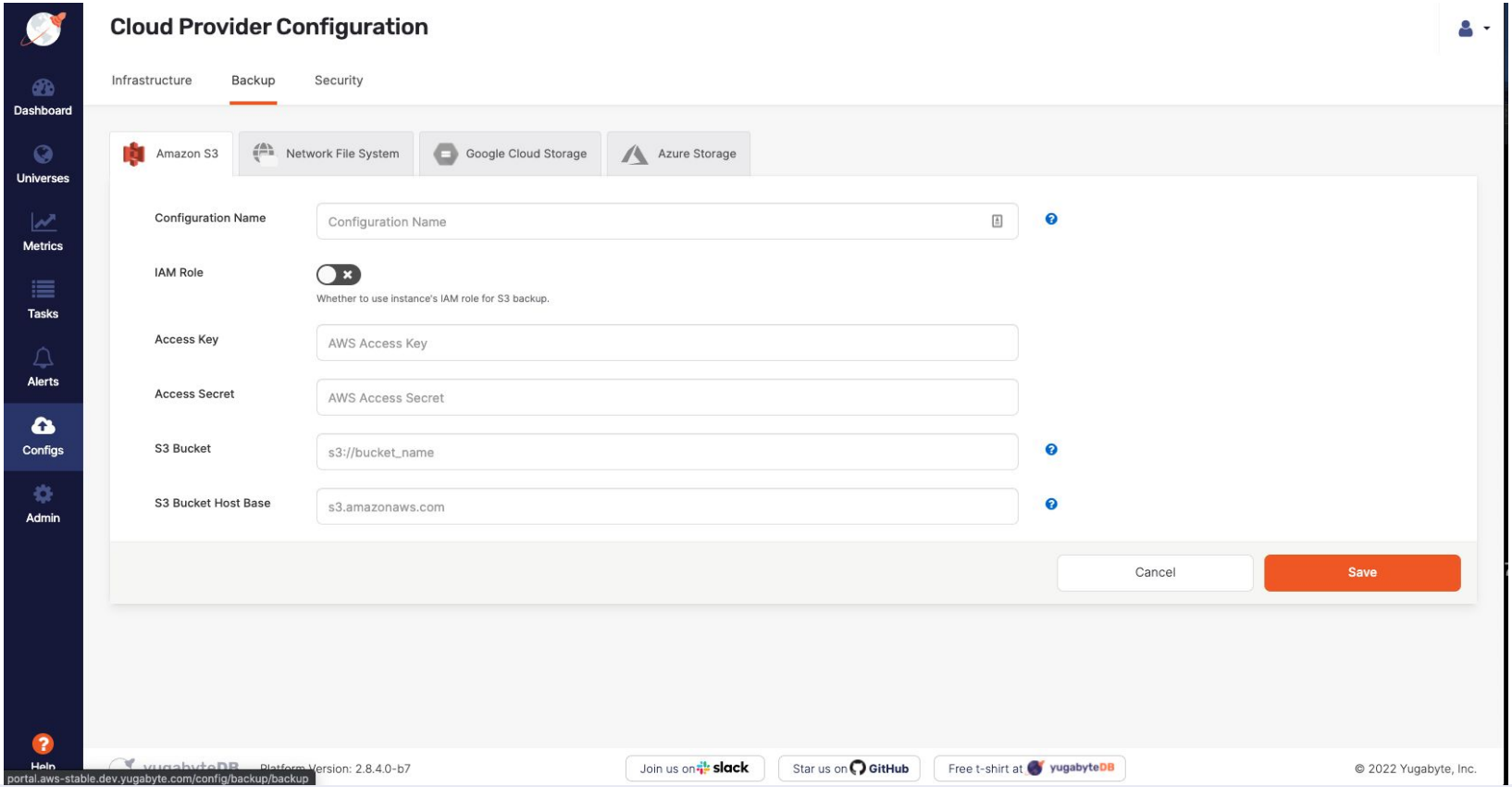
Backup/Restore in YugabyteDB

Core concept: Snapshots

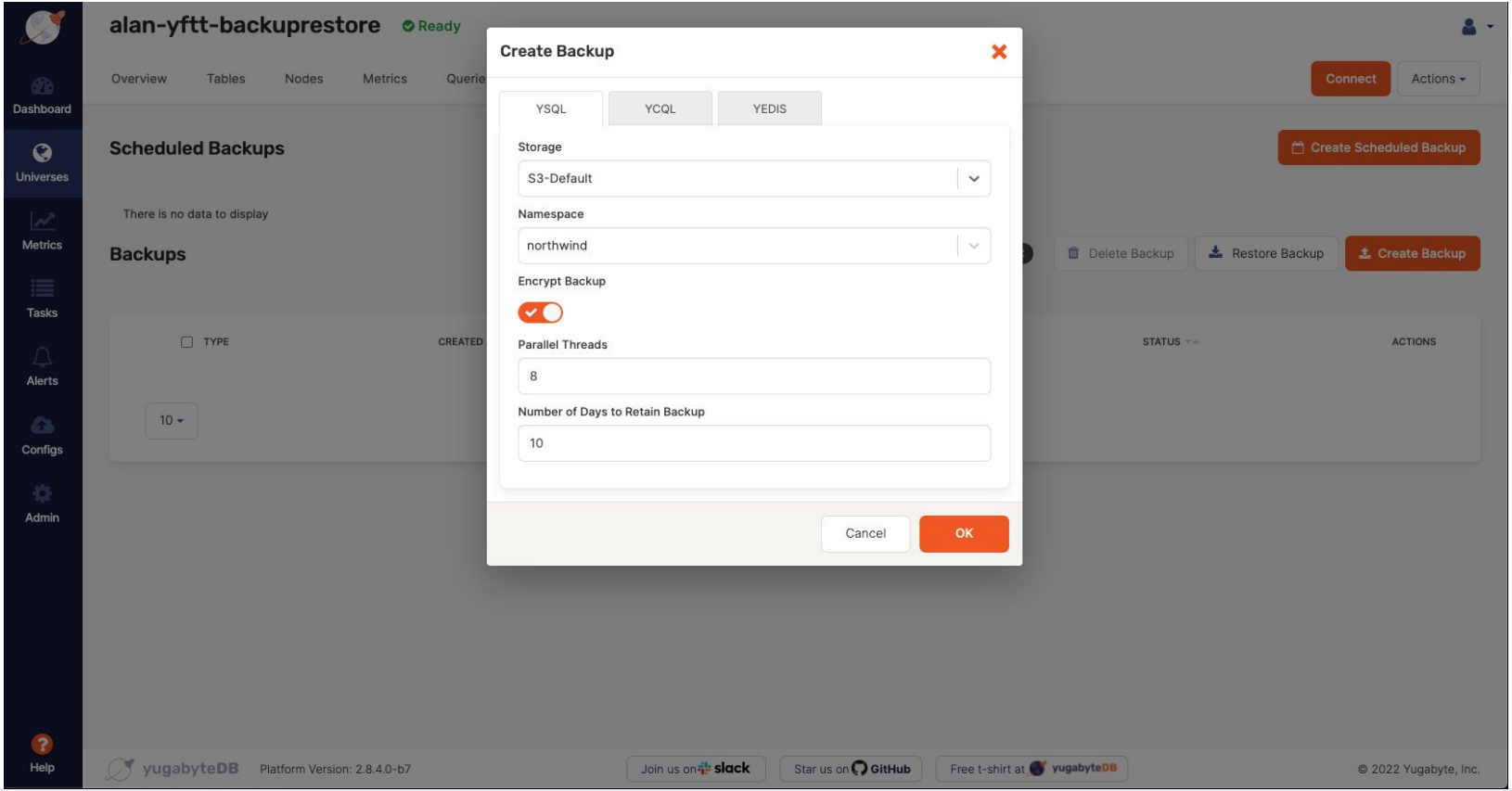
- A snapshot is a stable version of all of the flushed SST files at a specific point in time.
- For each table and each tablet, there is a <tablet_id>.snapshots directory
- Contained within each .snapshots directory, there is a <snapshot-id> directory that contains hard links to the flushed SST files
- These files will persist through compactions and will not change due to new data being inserted
- Deletion of a snapshot (typically at end of backup) removes these hard link files.

Backing up a database

First we need a place to put our backups



Then we can create a backup



What's going on behind the scenes?

Backup steps

1. For YSQL, check the catalog version
2. Create the snapshot
3. For YSQL, export the schema with ysql_dump and copy it to our S3 location.
4. For YSQL, check the catalog version again!
5. Export the snapshot and copy it to our S3 location
6. For each table, on the nodes with leader tablets, copy the snapshot directories to S3 (and checksum!)
7. Save the encryption key to S3 at the end. (If EAR is enabled)

Docs:
https://docs.yugabyte.com/latest/manage/backup-restore/snapshot-ysql/#create-a-snapshot-for-a-ysql-database

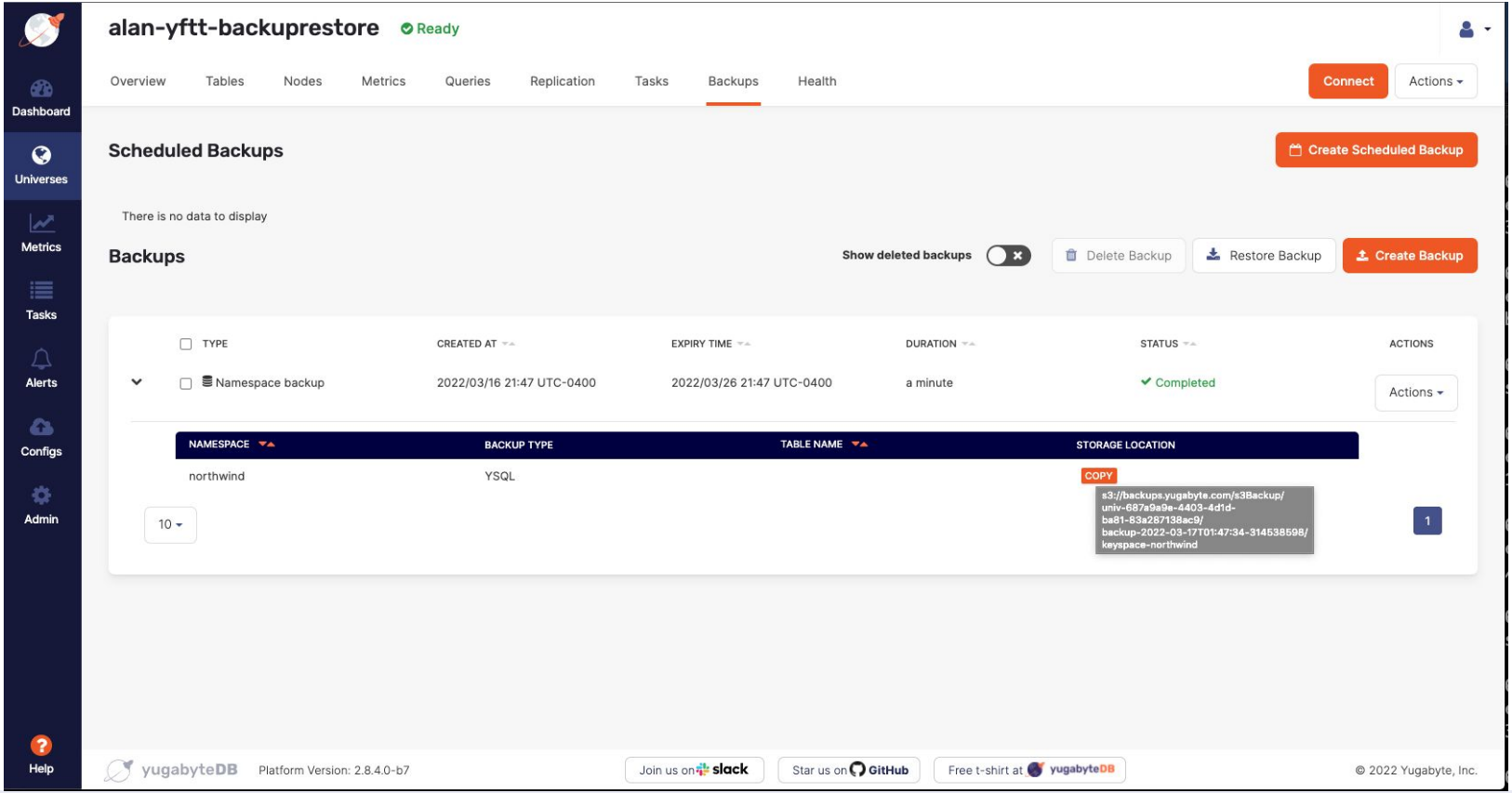
Why all the fuss with rechecking the catalog version?

Consistency

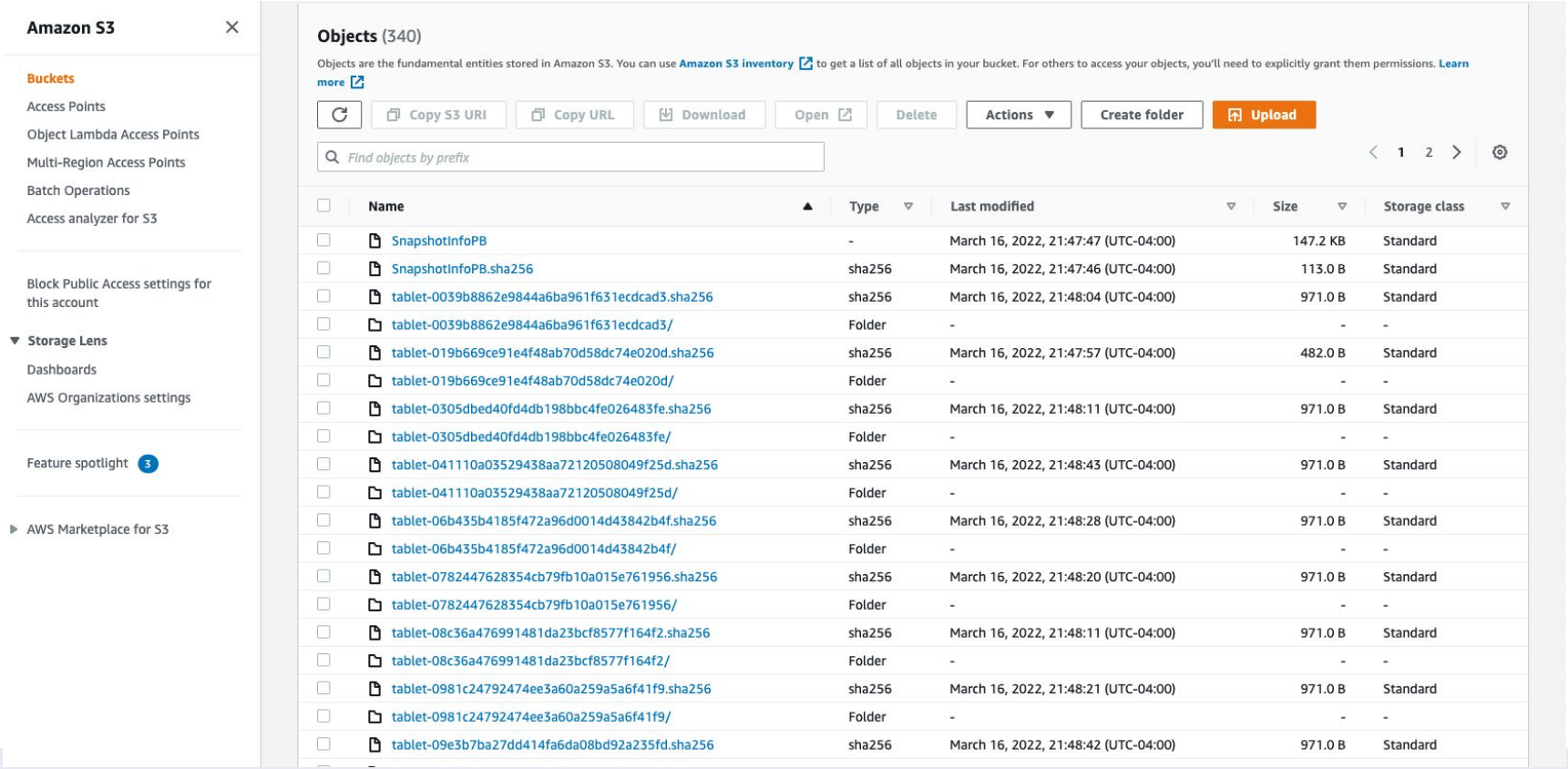
- The snapshot file contains DocDB's notion of what the table schema is as well as the tablet information for each table.
- The schema dump conducted by the *ysql_dump* utility must match in the number of objects, attributes in each table, and the **ORDER** of objects

If there is a mismatch between the two, then the restore will fail.

Backup is complete

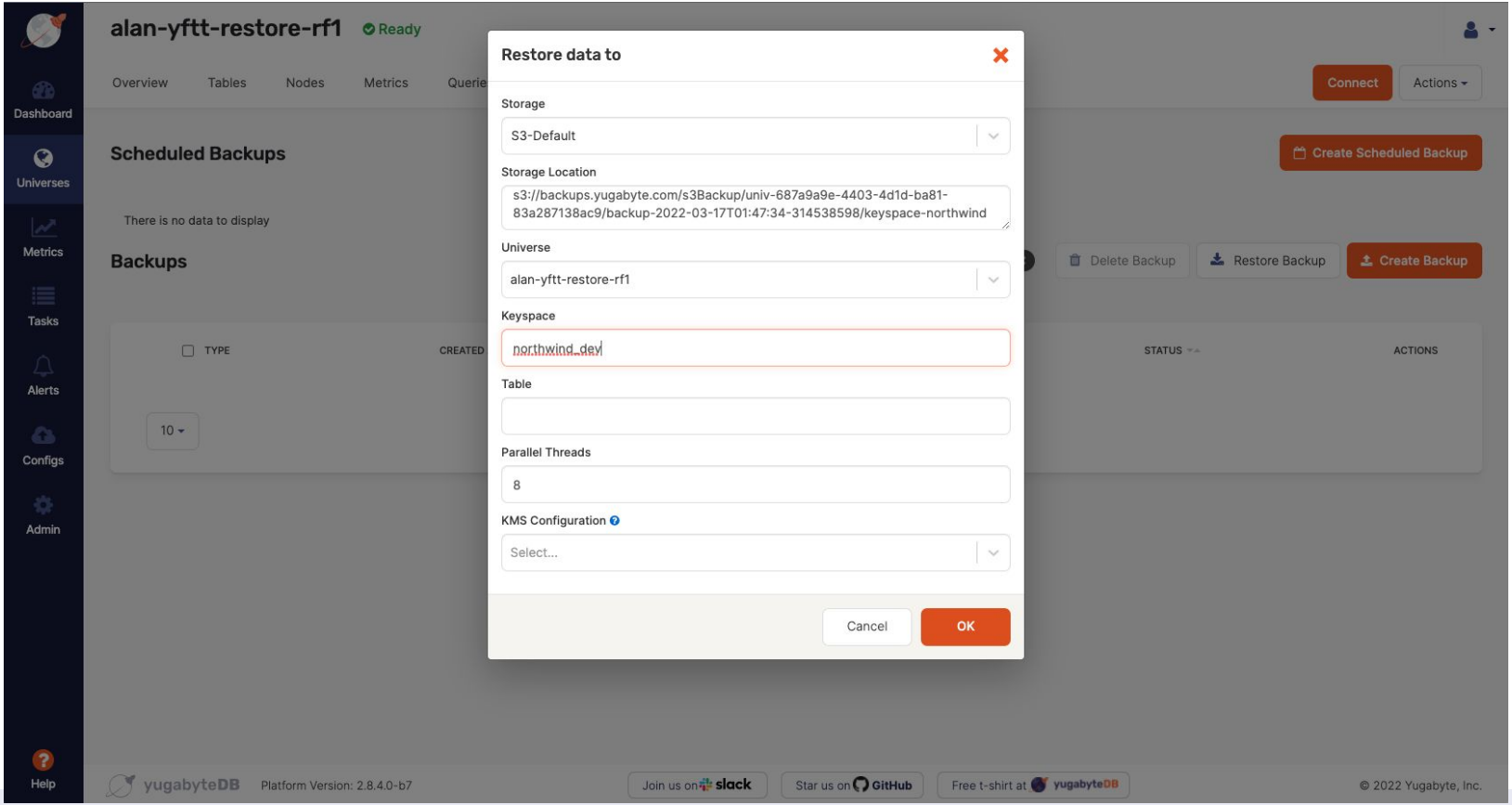


So how does it look in S3?



Restoring a database

Now lets restore to a single node cluster



What's going on behind the scenes when restoring?

Restore steps

1. Copy the YSQLDump from S3 and execute the YSQLDump script to create the tables & tablets necessary to receive the data.
2. Copy the SnapshotInfoPB down from S3
3. Import the SnapshotInfoPB file. The output of this step contains all of the new mappings for tablet-uuids generated on the new cluster.
4. Copy each tablets snapshot data from S3 to ALL tablet peers (and verify checksums!) to a temporary location.
5. Restore the snapshot to make it visible to the cluster.

Docs: https://docs.yugabyte.com/latest/manage/backup-restore/snapshot-ysql/#restore-a-snapshot

Point-in-time recovery

Point-in-time Recovery

What is PITR?

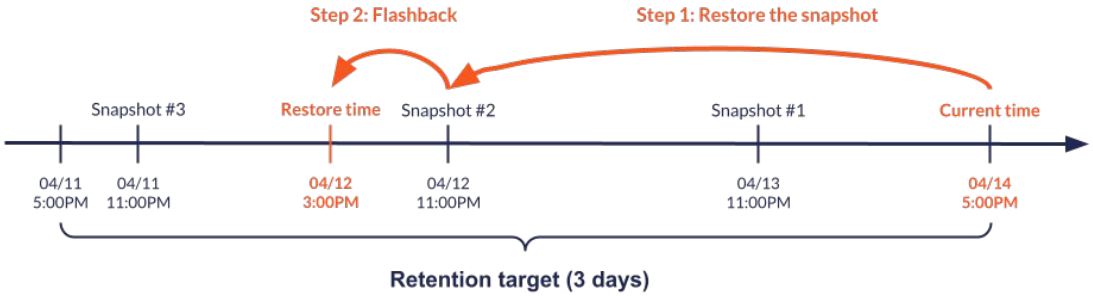
- Ability to restore a database or a keyspace to an arbitrary point in time representing the latest known working state
- Typically used to recover from user or software errors
  - Incorrect data modifications
  - Accidental schema changes
- GA since 2.14

Why PITR?

- Restore to any time => minimum RPO
- Based on **in-cluster** snapshots => minimum RTO

Point-in-time Recovery: under the hood

- PITR = periodic snapshots + MVCC
  - MVCC - history of past changes within a single snapshot (a.k.a. flashback)
- Example:
  - Retention target: 3 days
  - Snapshot interval: 1 day
  - Every snapshot holds 1 day worth of data changes



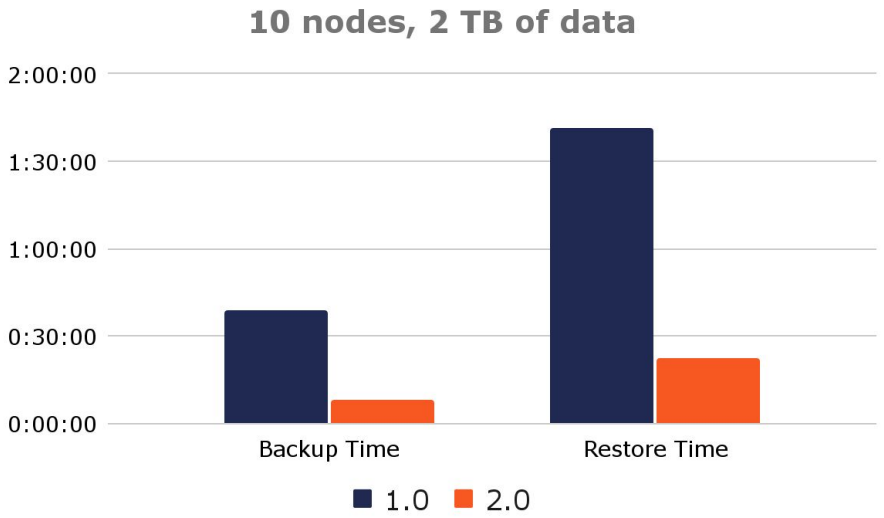
Coming attractions

Coming Soon!

- Point-in-time recovery management
- Backups 2.0
  - Performance improvements
  - Cost improvements

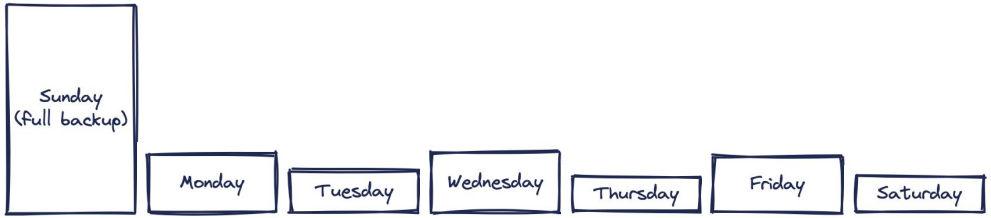
Backups 2.0

- Updated architecture and implementation for backup management
- Improved performance, lower resource consumption, better metrics



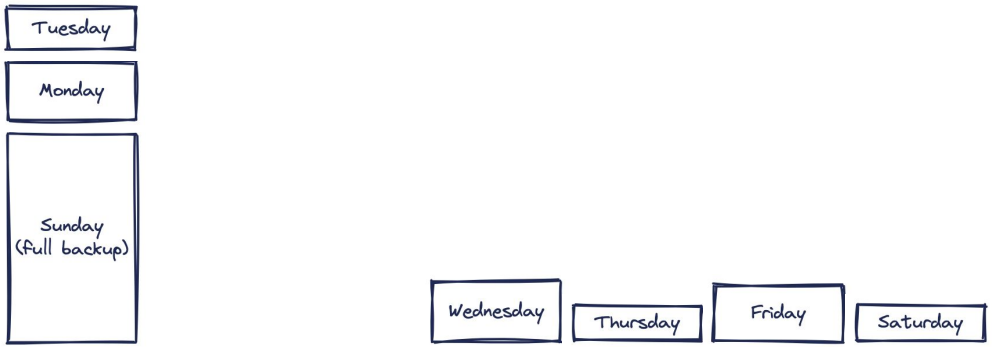
Backups 2.0

- Only copy added or updated since the latest backup
- Back up large datasets faster
- Reduce storage and data transmission cost



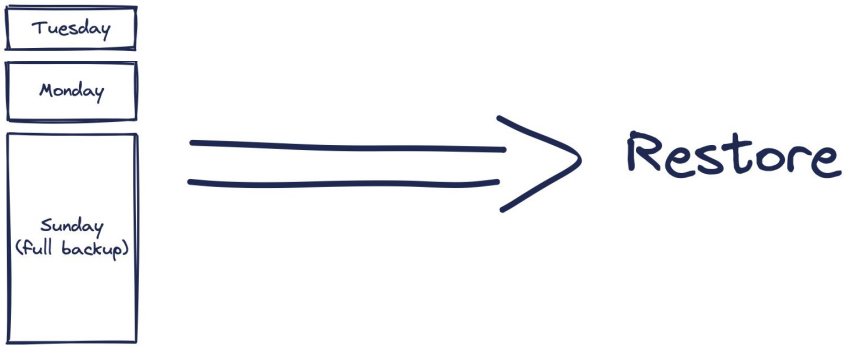
Backups 2.0

- Only copy added or updated since the latest backup
- Back up large datasets faster
- Reduce storage and data transmission cost



Backups 2.0

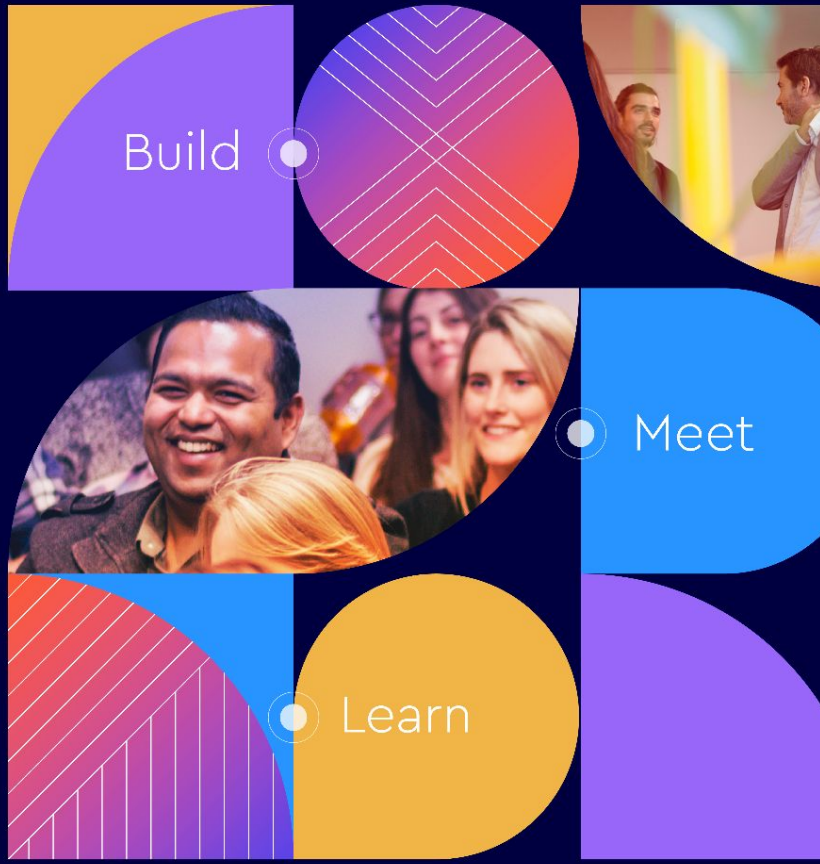
- Only copy added or updated since the latest backup
- Back up large datasets faster
- Reduce storage and data transmission cost



Summary

- Distributed snapshots are at the core of YugabyteDB backup/restore functionality
- YB Anywhere and YB Managed provide scheduled and on-demand backups stored in cloud storage or NFS
- PITR allows to quickly recover from user or software errors with minimal RPO and RTO
- Backups 2.0 will further reduce backup creation time and minimize the cost

https://docs.yugabyte.com/preview/manage/backup-restore/



Thank You

Join us on Slack: yugabyte.com/slack

Star us on Github: github.com/yugabyte/yugabyte-db