Business Continuity and DR in YugabyteDB

Val Kulichenko, Product Manager @ Yugabyte
Feb 10, 2023

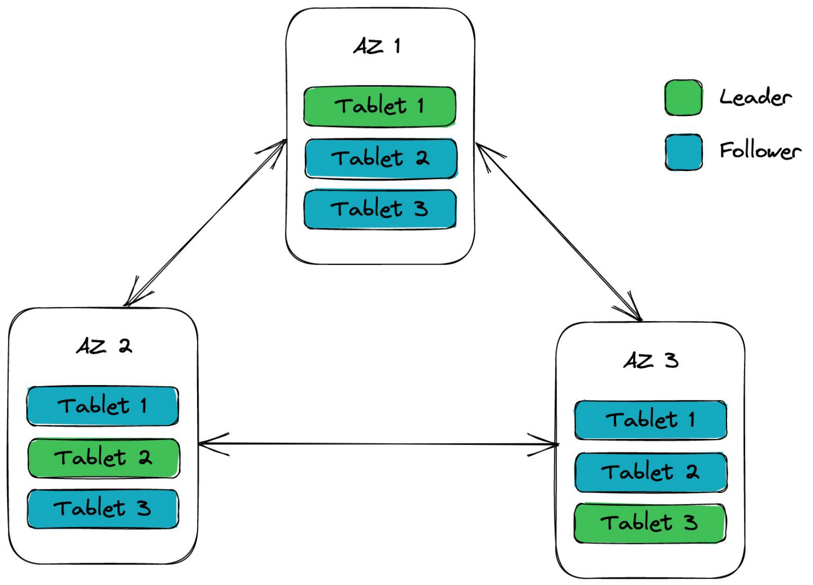
YugabyteDB Friday Tech Talks

YFTT

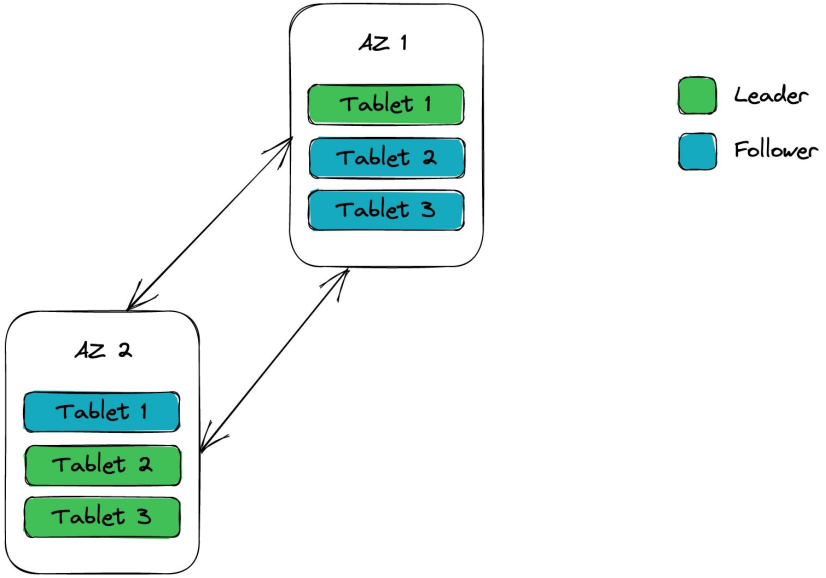
Talks by Engineers for Engineers

yugabyteDB

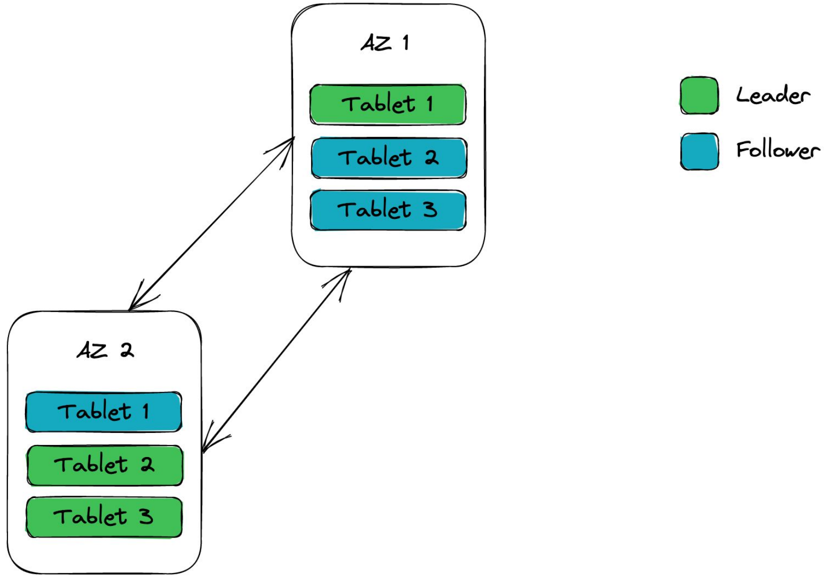
Why DR?


AZ 1
Tablet 1
Tablet 2
Tablet 3
AZ 2
Tablet 1
Tablet 2
Tablet 3
AZ 3
Tablet 1
Tablet 2
Tablet 3
Leader
Follower

Why DR?

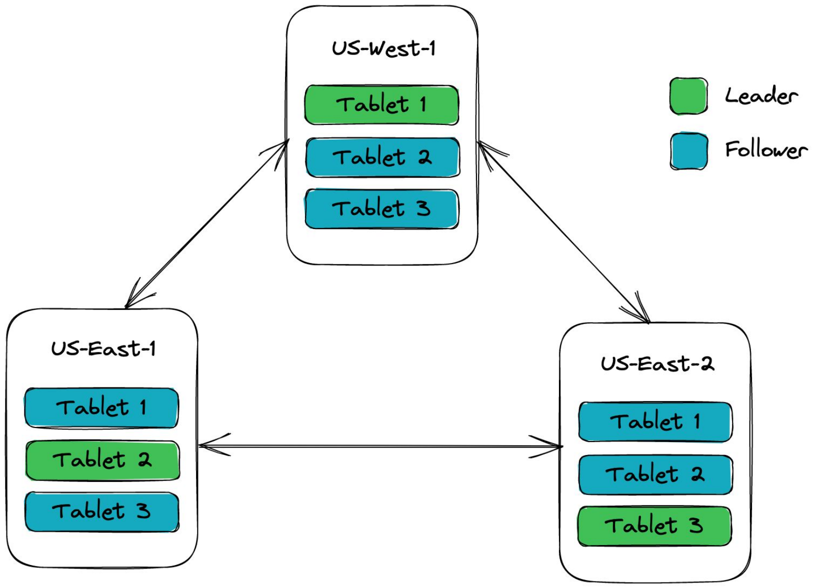

AZ 1
Tablet 1
Tablet 2
Tablet 3
AZ 2
Tablet 1
Tablet 2
Tablet 3
Leader
Follower

Why DR?



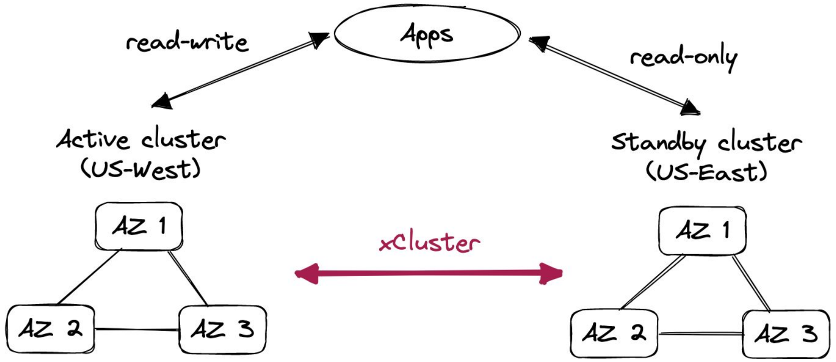
How to recover from a region outage?

DR with stretched cluster (sync replication)



- Optimized for **data protection**
- Pros:
  - Zero RPO at all times
  - Low operational complexity
- Cons:
  - High write latency

DR with xCluster (async replication)



- Optimized for **performance**
- Pros:
  - Lower write latency
- Cons:
  - Non-zero RPO in case of failover
  - More operations required to recover from a disaster

DR with xCluster: New in 2.17

- Transactional reads on standby cluster
  - Atomicity
  - Global ordering
- New APIs
  - Switch cluster role between active and standby
  - Wait for replication drain
  - Get standby safe time
- Failover workflows
  - Planned
  - Unplanned

Transactional standby reads: Atomicity

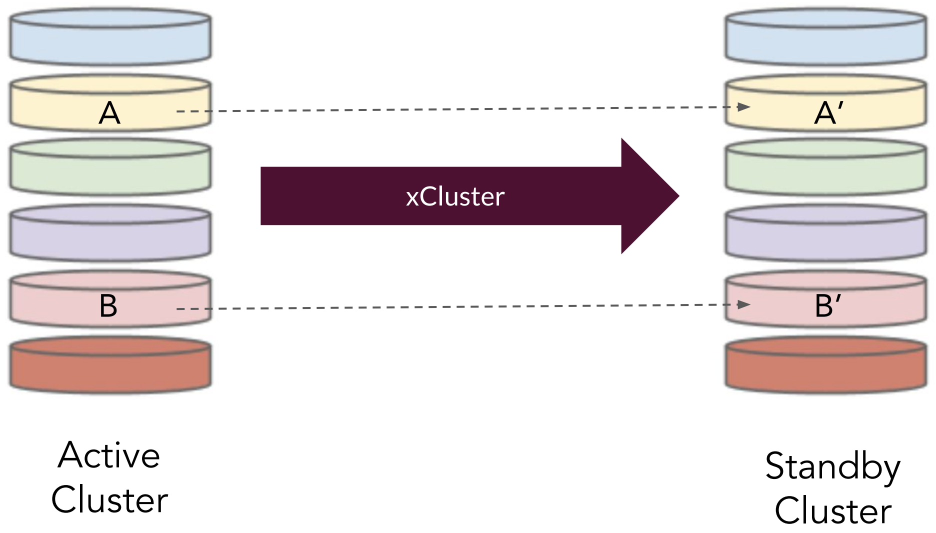
```
begin transaction
  update A
  update B
commit transaction
```

Active sees either:

- Changes to A and B
- Changes to neither A nor B

Standby can see (old behavior):

- Changes to A and B
- Changes to neither A nor B
- Changes to A but not to B
- Changes to B but not to A



Transactional standby reads: Atomicity

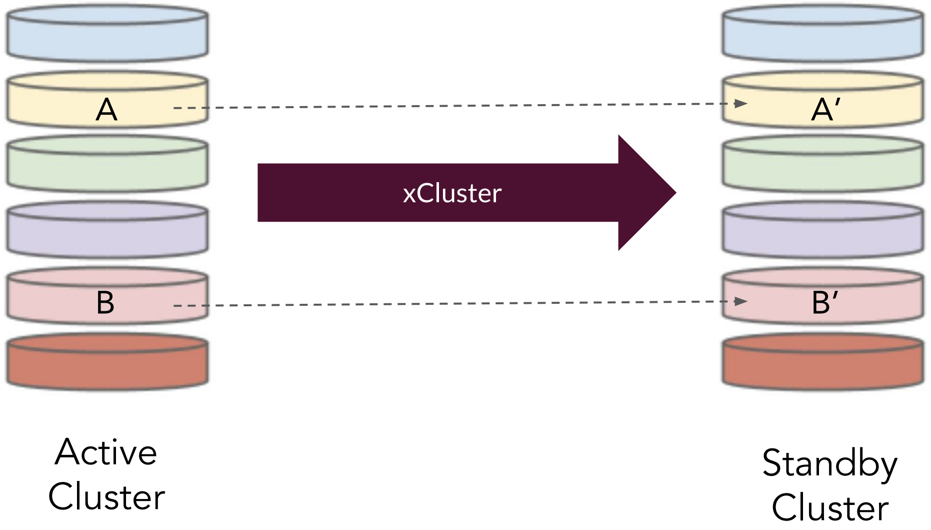
```
begin transaction
  update A
  update B
commit transaction
```

Active sees either:

- Changes to A and B
- Changes to neither A nor B

Standby can see (new behavior):

- Changes to A and B
- Changes to neither A nor B
- ~~Changes to A but not to B~~
- ~~Changes to B but not to A~~



Transactional standby reads: Global ordering

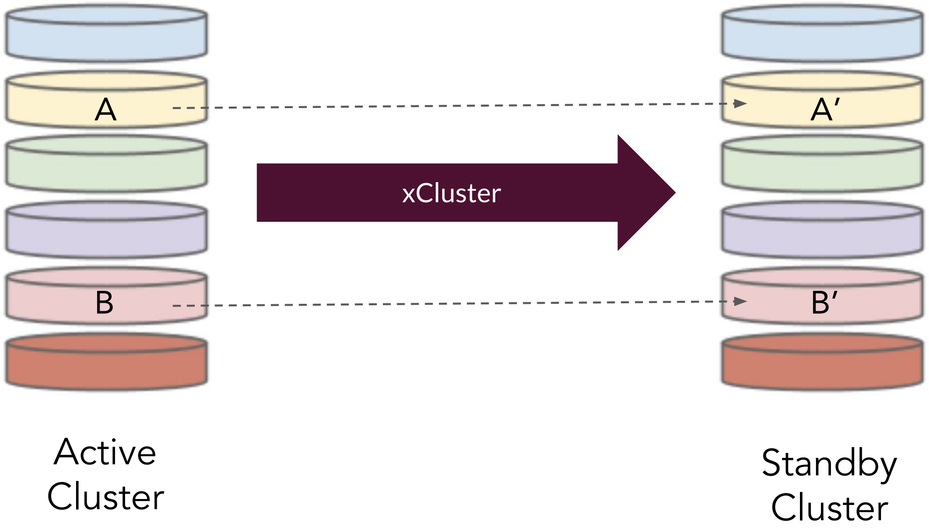
```
begin transaction
  update A
commit transaction
begin transaction
  update B
commit transaction
```

Active can see:

- Changes to neither A nor B
- Changes to A but not B
- Changes to A and B

Standby can see (old behavior):

- Changes to neither A nor B
- Changes to A but not to B
- Changes to A and B
- Changes to B but not to A



Transactional standby reads: Global ordering

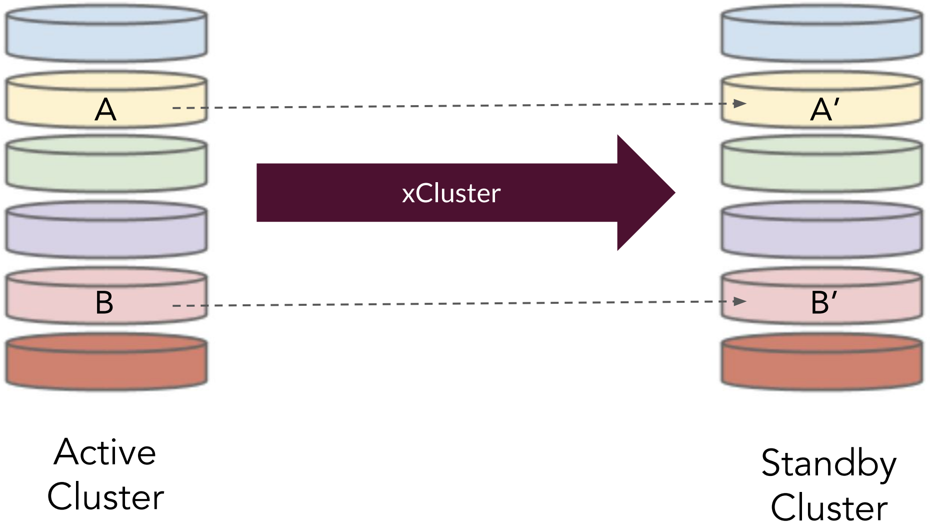
```
begin transaction
  update A
commit transaction
begin transaction
  update B
commit transaction
```

Active can see:

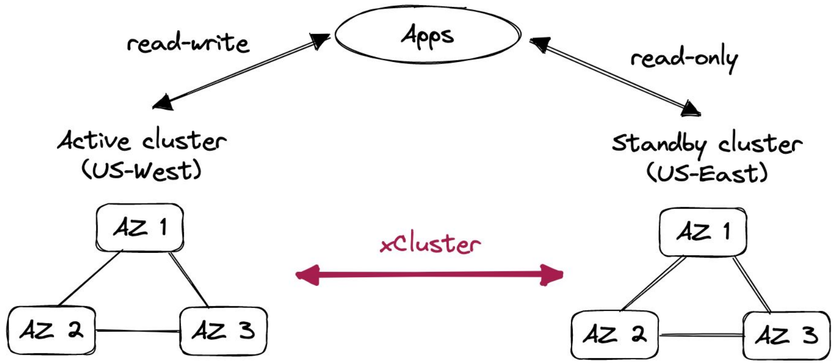
- Changes to neither A nor B
- Changes to A but not B
- Changes to A and B

Standby can see (new behavior):

- Changes to neither A nor B
- Changes to A but not to B
- Changes to A and B
- ~~Changes to B but not to A~~



Initial DR setup



1. Turn on the `enable_replicate_transaction_status_table` GFlag on both clusters
   - Global GFlag will not be required in GA
2. Enable PITR for all participating databases on both clusters
3. Create **bi-directional** xCluster replication for the tables you want to include
4. Identify the US-East cluster as standby

```
yb-admin -master_addresses <us.east.cluster> change_xcluster_role STANDBY
```

Note on the standby role

- The standby role:
  - Forces the read-only mode (*future work*)
  - Forces reads to happen based on the **safe time**, rather than the latest time

Active

```
update A (time t1)
update B (time t2)
update C (time t3)
```

$\Longrightarrow$

Standby

```
update A (t1)
update C (t3)
```

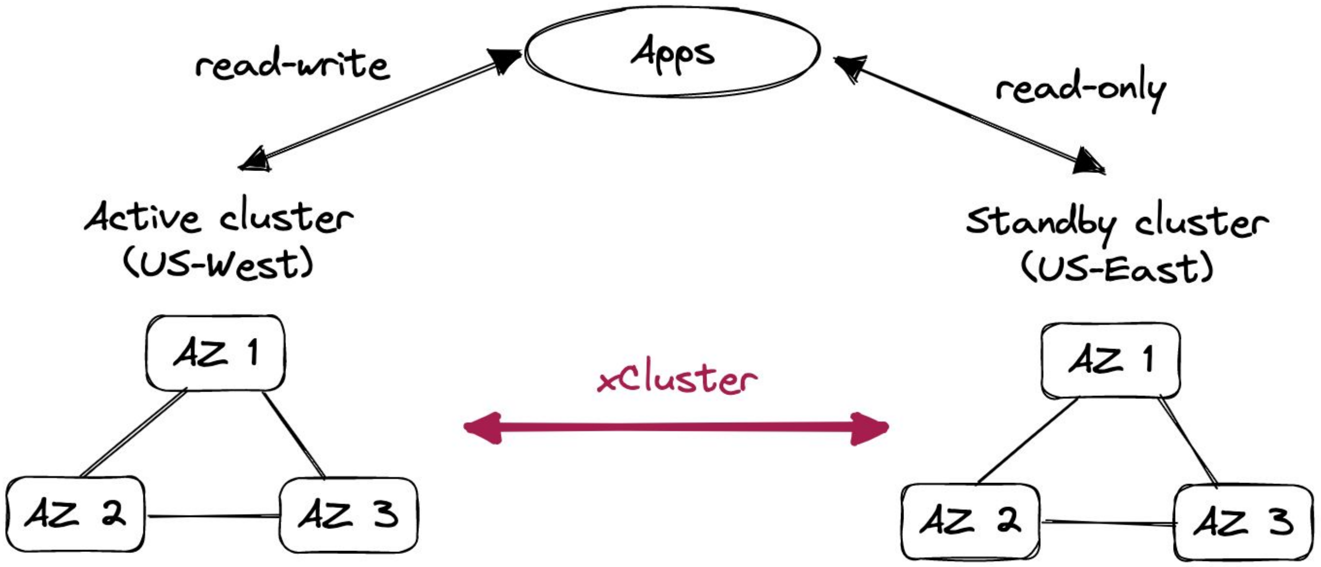
Standby can't serve C until B arrives!

=> `safe_time = t1`

```
yb-admin -master_addresses <standby.cluster> get_xcluster_safe_time
```

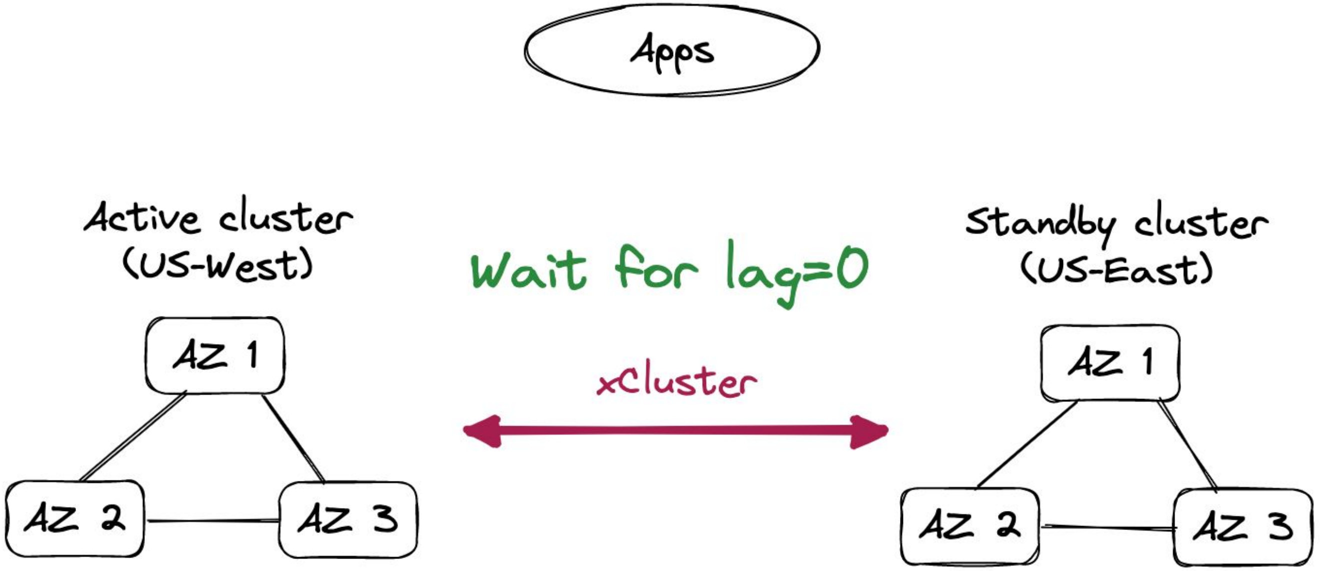
Planned failover

- Usually performed during maintenance, e.g. as a “fire drill” for DR
- Requirement: no data loss!



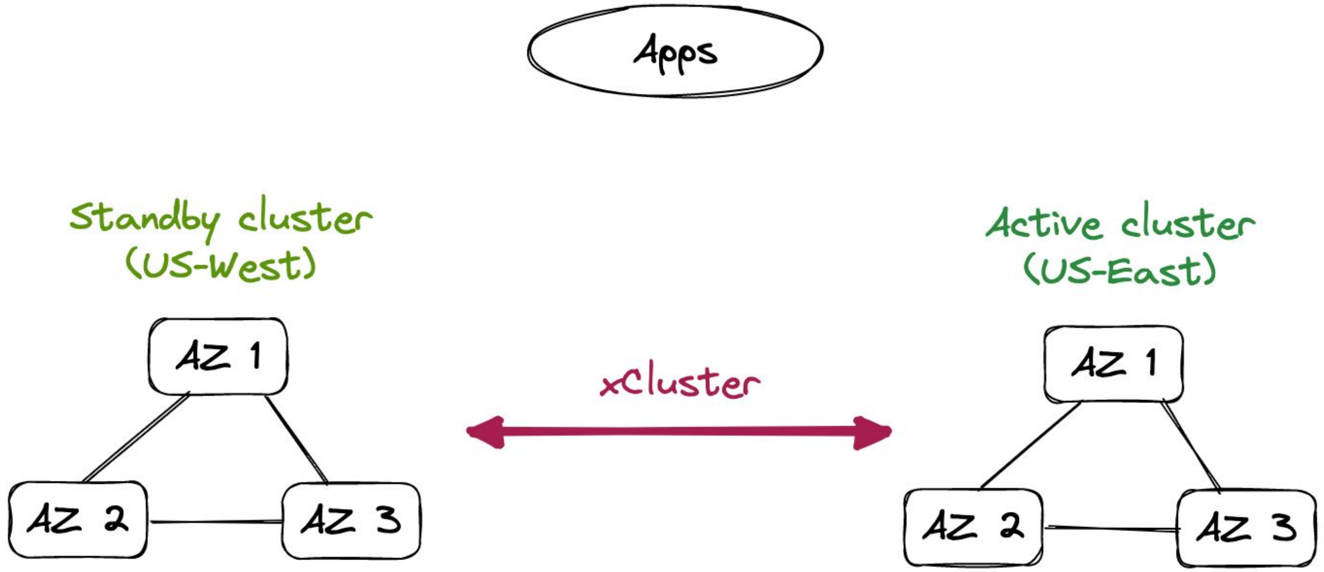
Planned failover

- Step 1: Stop the applications
- Step 2: Wait for the replication drain



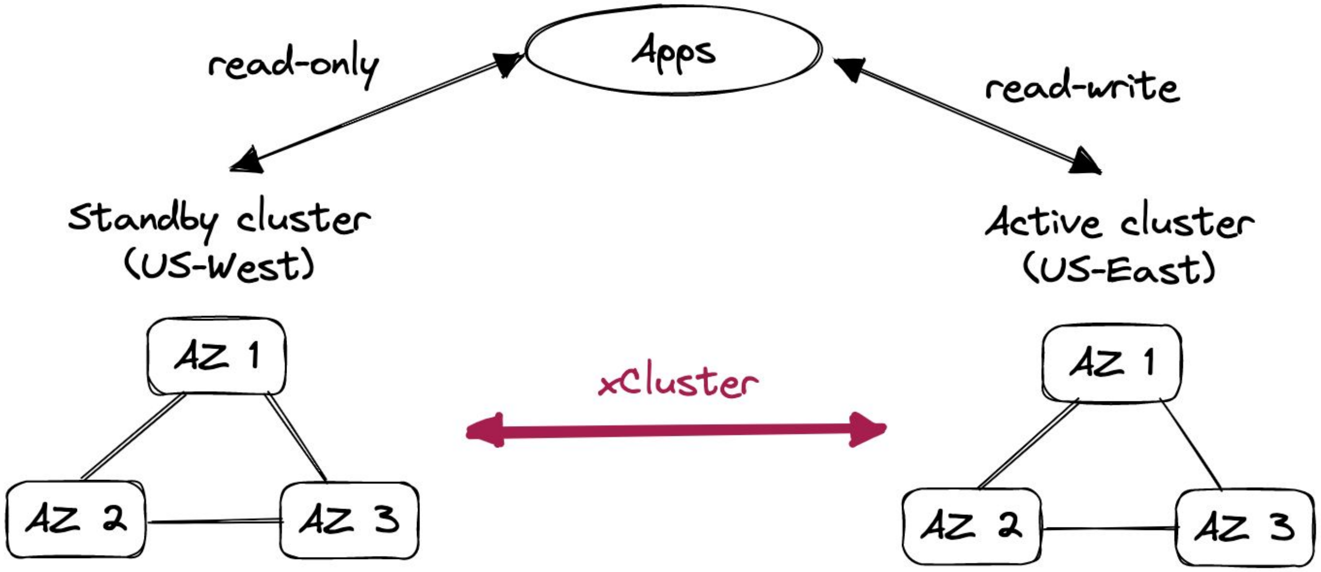
Planned failover

- Step 3: Demote US-West to standby role
- Step 4: Promote US-East to active role



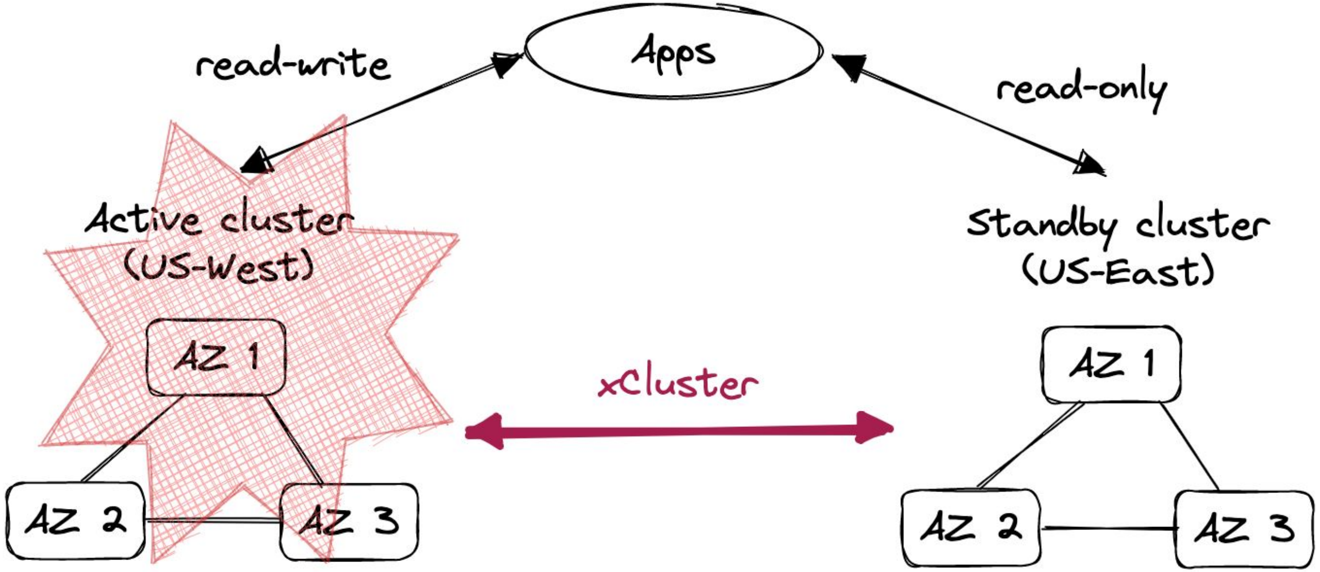
Planned failover

- Step 5: Resume the applications



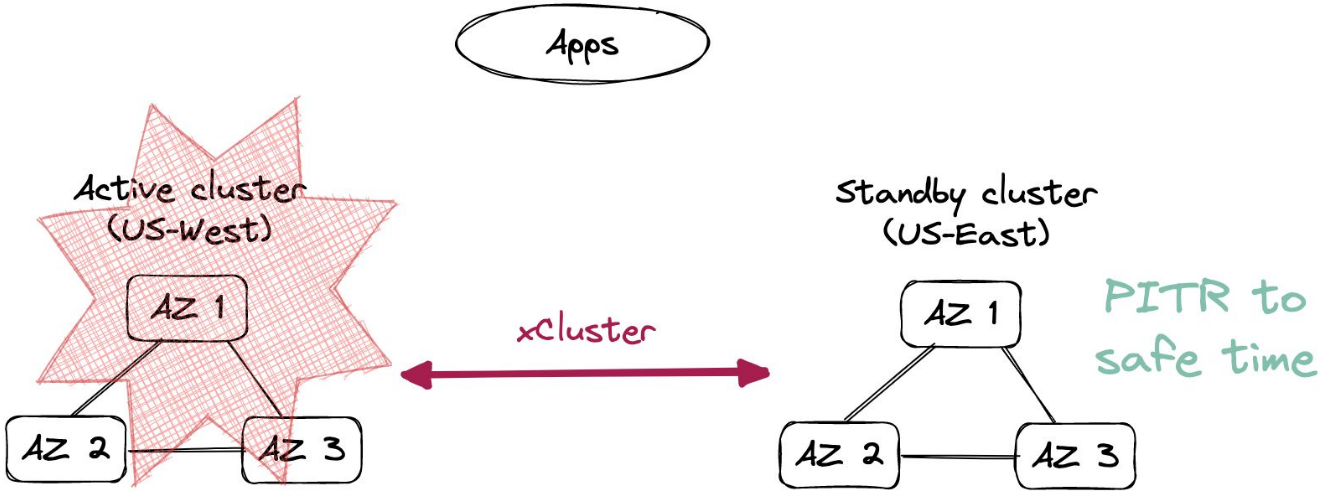
Unplanned failover

- Used to recover from an active cluster outage (disaster scenario)
- Some data loss is acceptable



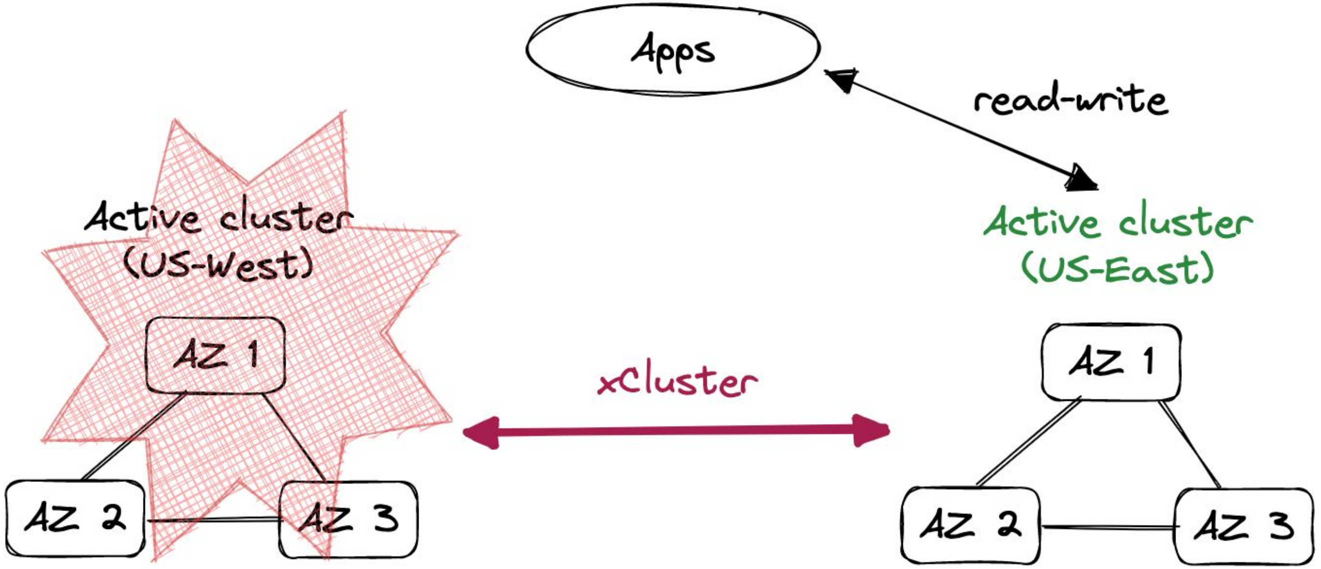
Unplanned failover

- Step 1: Stop the applications
- Step 2: Use PITR to restore US-East to safe time



Unplanned failover

- Step 3: Promote US-East to active role
- Step 2: Resume the applications on US-East



Note on estimated data loss estimation

- Active-side metric: replication lag
  - Not available after failure!
- Standby-side metric: safe time skew
  - `consumer_safe_time_skew = max_safe_time - current_safe_time`

Combine two metrics for a best effort estimate of the data loss in case of unplanned failover.

Adding table or index

1. Create on the standby cluster
2. Create on the active cluster
3. Set up the replication for the new table or index

Caution! It’s highly recommended that a new table is updated only after the replication is set up. Otherwise, full bootstrap might be required.

Altering a table

Recommended way:

1. Alter on the standby cluster
2. Alter on the active cluster

Possible way:

1. Alter on the active cluster
2. Alter on the standby cluster

In the latter case, the replication will be automatically paused and resume in between the steps.

```
Active: 127.0.0.1
Standby: 127.0.0.2

./bin/ysqlsh -h 127.0.0.1 -c "create table car(a int)"
./bin/ysqlsh -h 127.0.0.2 -c "create table car(a int)"

./bin/ysqlsh -h 127.0.0.1 -c "alter table car add column b int"  ====> Automatically pauses active to standby replication
./bin/ysqlsh -h 127.0.0.2 -c "alter table car add column b int"  ====> Automatically resumes replication

./bin/ysqlsh -h 127.0.0.2 -c "alter table car add column c int"  ====> Replication does not pause
./bin/ysqlsh -h 127.0.0.1 -c "alter table car add column c int"  ====> Replication does not pause
```

Future work

- Read-only mode for standby cluster
- Ability to set the role on replication level
- DB-level replication (automatically replicate new tables and indexes)
- Automatic DDL replication
- Quick failback based on PITR
- One-to-many replication
- YCQL support

Demo

Thank You

Join us on Slack: yugabyte.com/slack (#yftt channel)

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

YugabyteDB Friday Tech Talks

YFTT

Talks by Engineers for Engineers

yugabyteDB