ClickHouse® Disaster Recovery
Tips and Tricks to Avoid Trouble in Paradise

Alexander Zaitsev - Altinity CTO
Robert Hodges - Altinity CEO

Let's make some introductions

Robert Hodges

Database geek for 40 years. Open source since 2006. ClickHouse since 2019.

Alexander Zaitsev

Expert in high scale analytics systems design and implementation. Altinity CTO

Altinity

Authors of Altinity.Cloud®, Altinity Kubernetes Operator for ClickHouse®, Altinity Grafana plugin for ClickHouse®, clickhouse-backup project, etc.

What is Disaster Recovery?

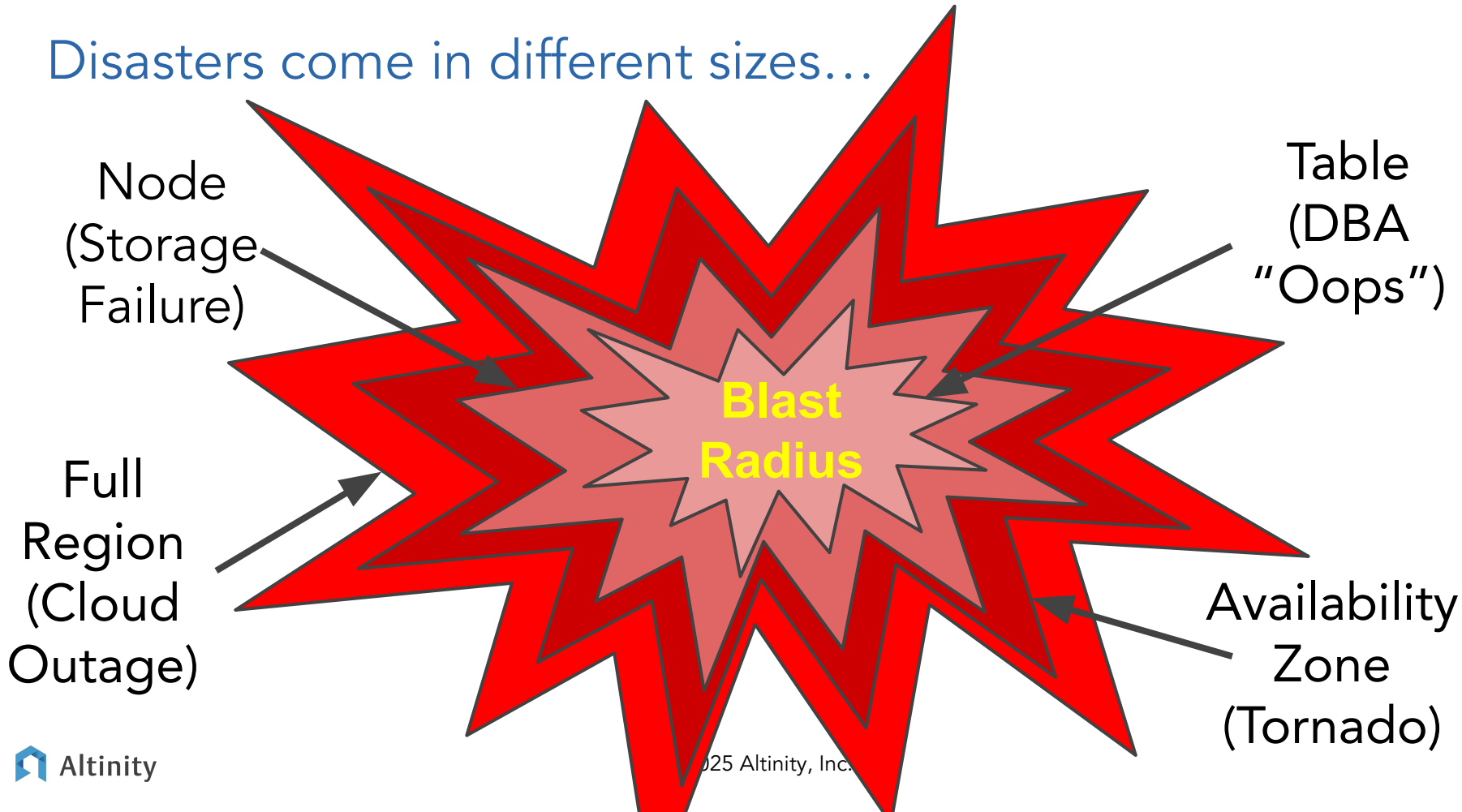
And what options are there for ClickHouse installations?

What is Disaster Recovery?

Disaster recovery (DR) is an organization's ability to restore access and functionality to IT infrastructure after a disaster event, whether natural or caused by human action (or error).

Google Cloud Documentation

Disasters come in different sizes…



And there are a lot of questions to think about

How much data can I lose? (aka Recovery Point Objective or RPO)

How long will it take to restart the service? (aka Recovery Time Objective or RTO)

How much will it cost?

How much does it add to management complexity?

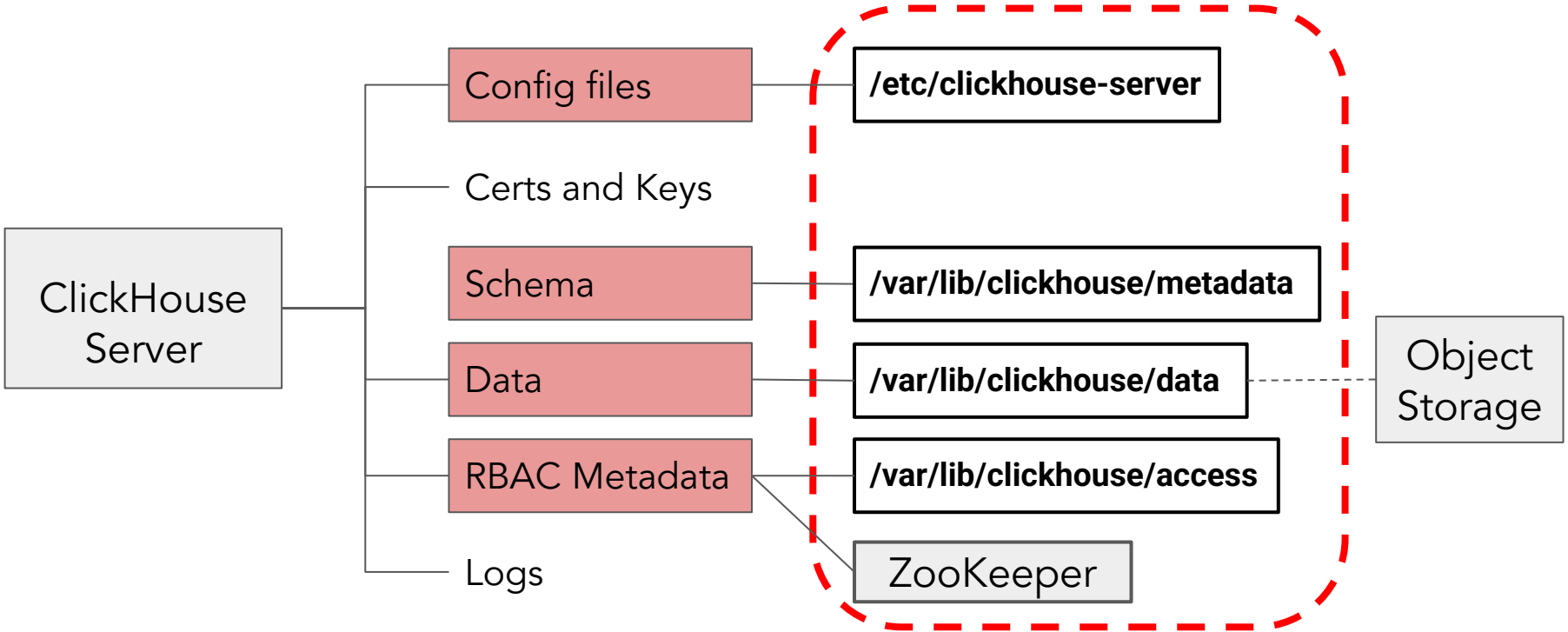
ClickHouse users have multiple choices for DR

DR Solution	Benefits	Drawbacks
Backups	• Cheapest • Easy to implement • Covers DBA errors	• Slow restore on large tables • Potentially large data loss
Single-region replication	• Easiest to implement • Available instantly	• Region failure not covered
Cross-region replication	• Broadest coverage • Available instantly for reads	• Complex setup (esp. networking) • Requires failover procedure • Costly to operate
Independent clusters	• Broadest coverage • Available instantly for read/write	• Data drift between clusters • Potentially most costly solution

Backups

Including how to shorten the data loss window

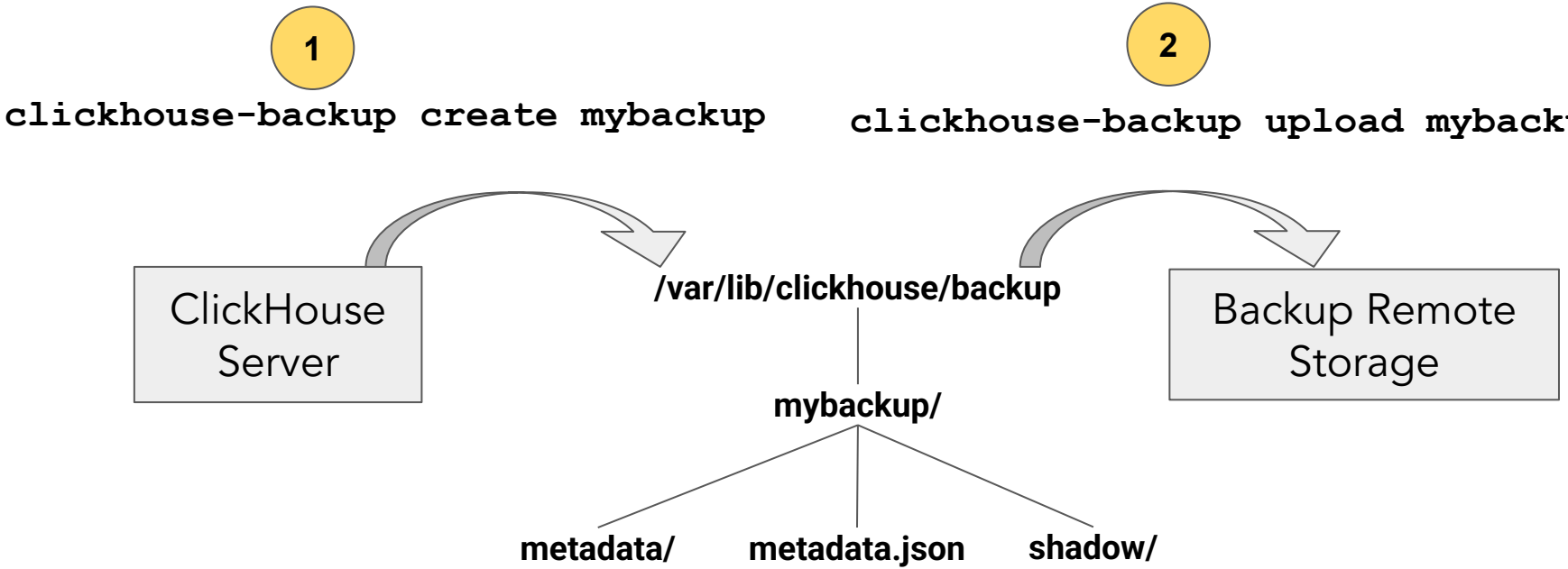
What do we need to protect in ClickHouse?



Common backup/restore options for ClickHouse

Tool	Description	Configs	Schema	Data	RBAC
ClickHouse Copier	Does full or partial copy of data to another ClickHouse	☐	✅	✅	☐
Altinity Backup for ClickHouse® (aka clickhouse-backup)	Standalone backup utility for all ClickHouse versions	✅	✅	✅	✅
ClickHouse BACKUP & RESTORE	Built-in SQL operations in ClickHouse	☐	✅	✅	☐

How to create a backup with clickhouse-backup



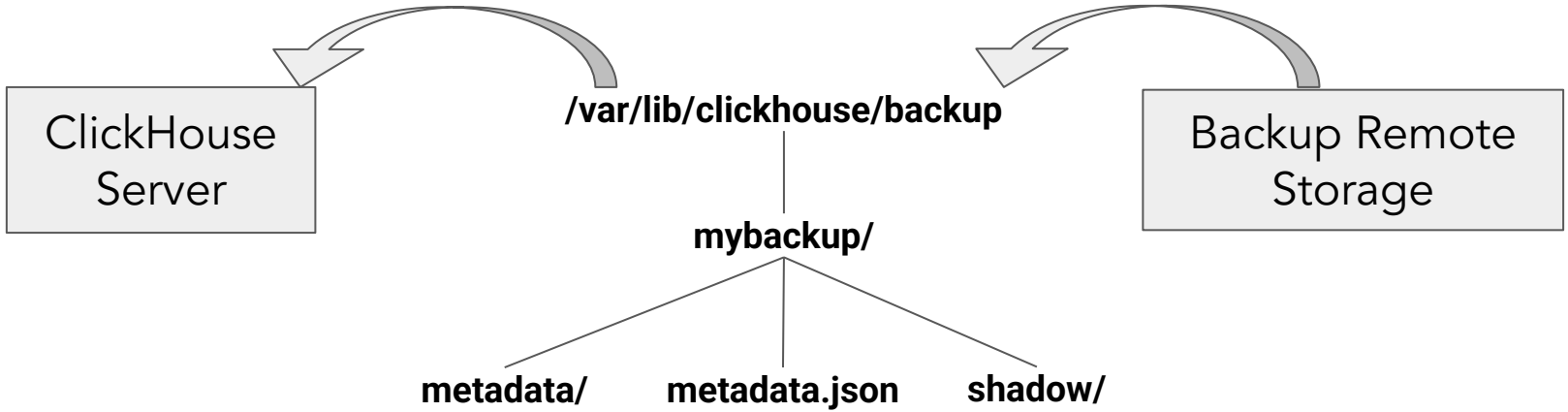
And how to restore it

4

```
clickhouse-backup restore mybackup
```

3

```
clickhouse-backup download mybackup
```



Different restore options

```
clickhouse-backup restore mybackup --schema

clickhouse-backup restore mybackup --configs

clickhouse-backup restore mybackup --table=my_table

clickhouse-backup restore mybackup --table=my_table
--partitions=202503

clickhouse-backup restore mybackup
--restore-database-mapping=my_db:my_restored_db
```

See full list at https://github.com/Altinity/clickhouse-backup

EMBEDDED (SQL) BACKUP

```
BACKUP DATABASE test_database TO Disk('backups', 'test_database_backup');

RESTORE DATABASE test_database FROM Disk('backups',
'test_database_backup');

RESTORE TABLE test_database.test_table_1 AS restore_database.test_table_1
FROM Disk('backups', 'test_database_backup');
```

https://clickhouse.com/docs/operations/backup

Backup Database Engine (25.2+)

```
CREATE DATABASE backup_database
    ENGINE = Backup('test_database_backup', 'backup_destination')

SHOW TABLES FROM backup_database

┌─name─────────┐
│ test_table_1 │
│ test_table_2 │
│ test_table_3 │
└──────────────┘
```

https://clickhouse.com/docs/engines/database-engines/backup

Which Backup Tool to Use

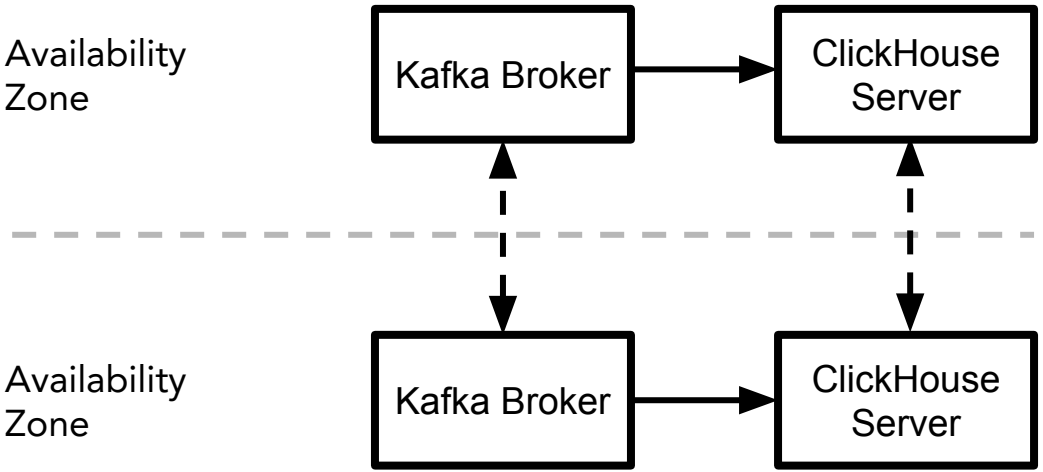
	clickhouse-backup	SQL BACKUP/RESTORE
Destination options	Local, S3, GCS, Azure, Tencent, FTP, SFTP, rsync, restic and others	Local, S3, GCS, Azure
Recovery options	Very flexible	Limited, varies on ClickHouse version
Incremental backup	Supported	Supported
Backup settings, RBAC	Supported	–
Automation	Supported (server mode, retention)	–
Version dependency	Works with any ClickHouse version	Functionality depends on ClickHouse version
Ease of extension	Very easy, regular updates	Complicated PR review to ClickHouse upstream

Backing up (Zoo)Keeper?

Backup shared objects: RBAC, UDFs, etc.
(Use --rbac on clickhouse-backup)

SYSTEM RESTORE REPLICA recreates table metadata if you lose everything

Tip: Kafka can erase the data loss window in backups



Backup Script:

1. Stop ingest.
2. Store topic offsets in ClickHhouse table.
3. Wait for replica sync
4. Backup ClickHouse
5. Re-enable ingest

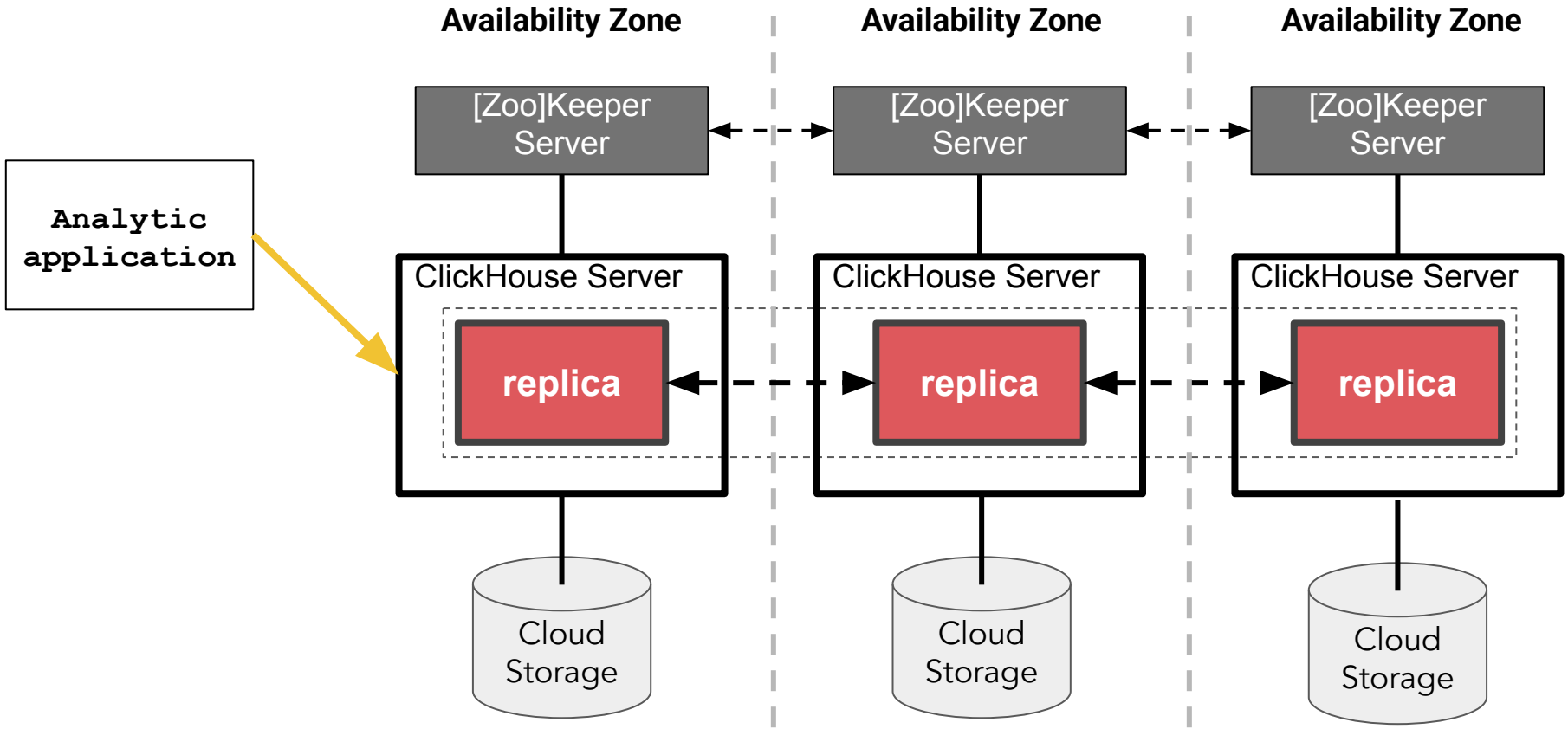
Restore Script:

1. Restore tables.
2. Reset Kafka topic offsets from ClickHouse.
3. Re-enable ingest

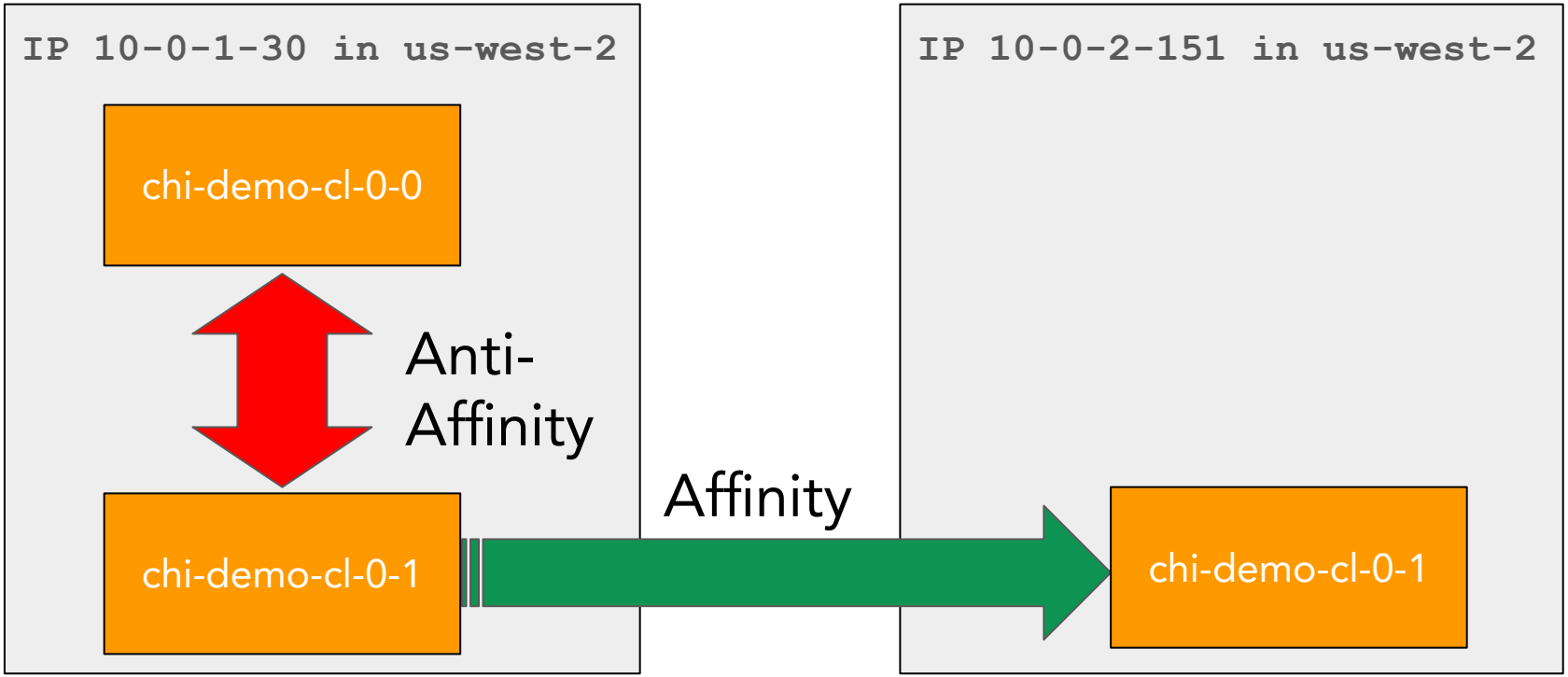
Cross-AZ Replication

Handling failure within single data centers

Setting up cross-AZ operation in public cloud



Affinity vs. anti-affinity



Tip: Use Altinity Kubernetes operator to run ClickHouse

ClickHouse Installation manifest

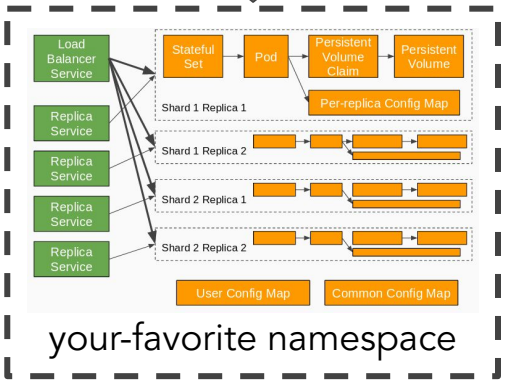
kubectl -f apply demo.yaml

kube-system namespace

Altinity ClickHouse Operator

Apache 2.0 source, distributed as Docker image

"Adjust reality"



Best practice deployment

Assign pod to a VM type

```
apiVersion: "clickhouse.altinity.com/v1"
kind: "ClickHouseInstallation"
metadata:
  name: "demo"
spec:
  configuration:
    clusters:
      - name: "us-west-2"
        layout:
          shards:
            - replicas:
                - templates:
                    podTemplate: replica-in-zone-us-west-2a
                - templates:
                    podTemplate: replica-in-zone-us-west-2b
```

Two replicas for 1 shard

Separate pod template for each AZ

Add affinity rules for each pod

```
templates:
  podTemplates:
    - name: replica-in-zone-us-west-2a
      zone:
        values:
          - "us-west-2a"
      podDistribution:
        - type: ClickHouseAntiAffinity
          scope: ClickHouseInstallation
      spec:
        containers:
        - name: clickhouse
          image: altinity/clickhouse-server:23.8.8.21.altinitystable
```

Pod must be scheduled in us-west-2a

Keep pods on different hosts

Where are my pods running?

```
kubectl get pod
-o=custom-columns=NAME:.metadata.name,STATUS:.status.phase,NODE:.spec.nodeName
NAME                            STATUS    NODE
chi-demo-us-west-2-0-0-0        Running   ip-10-0-1-30.us-west-2.compute.internal
chi-demo-us-west-2-0-1-0        Running   ip-10-0-2-151.us-west-2.compute.internal
```

Checking Kubernetes worker locations

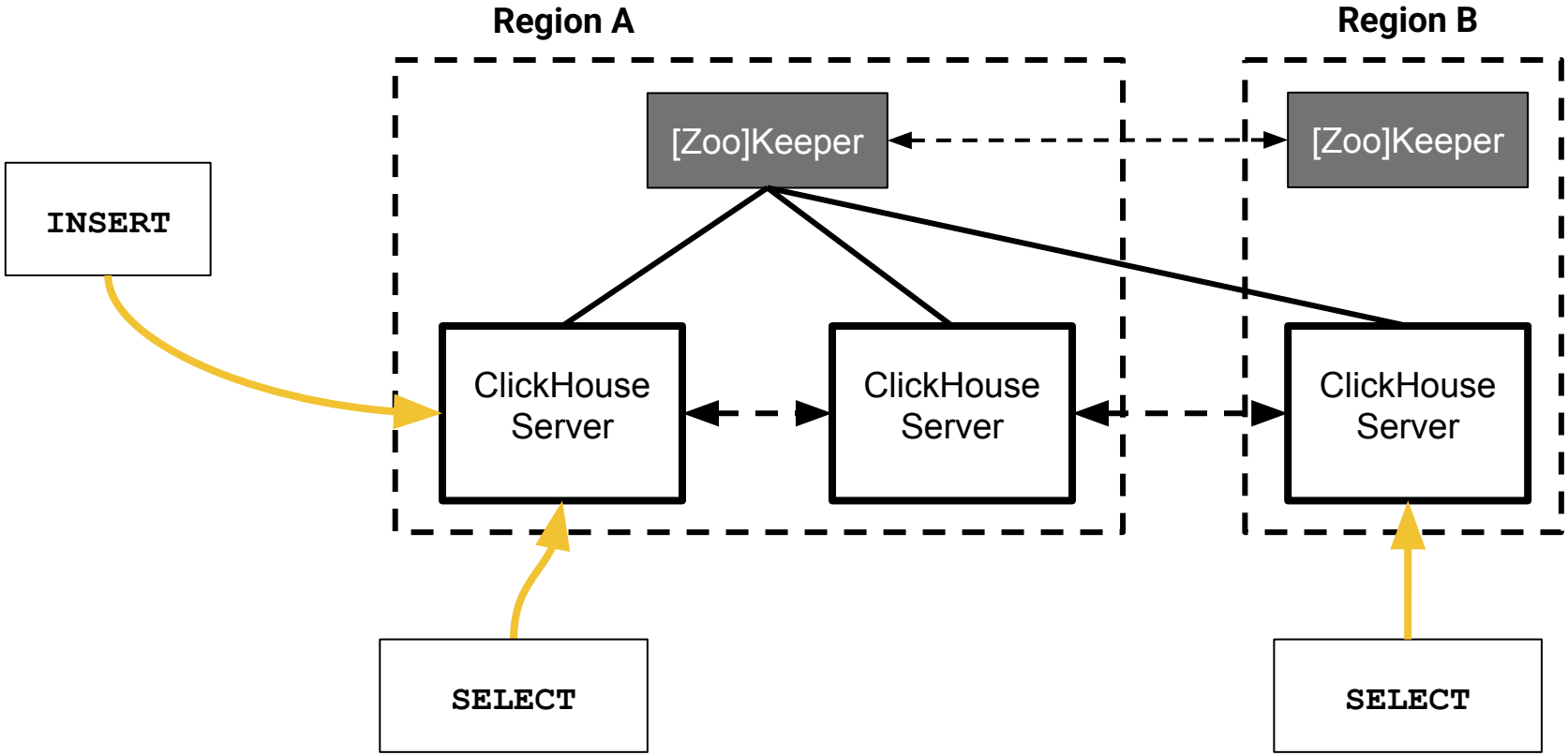
```
kubectl get node -o=custom-columns=NODE:.metadata.name,ZONE:
.metadata.labels.'topology\.kubernetes\.io/zone'

NODE                                           ZONE
ip-10-0-1-30.us-west-2.compute.internal        us-west-2a
ip-10-0-2-151.us-west-2.compute.internal       us-west-2b
ip-10-0-3-126.us-west-2.compute.internal       us-west-2c
```

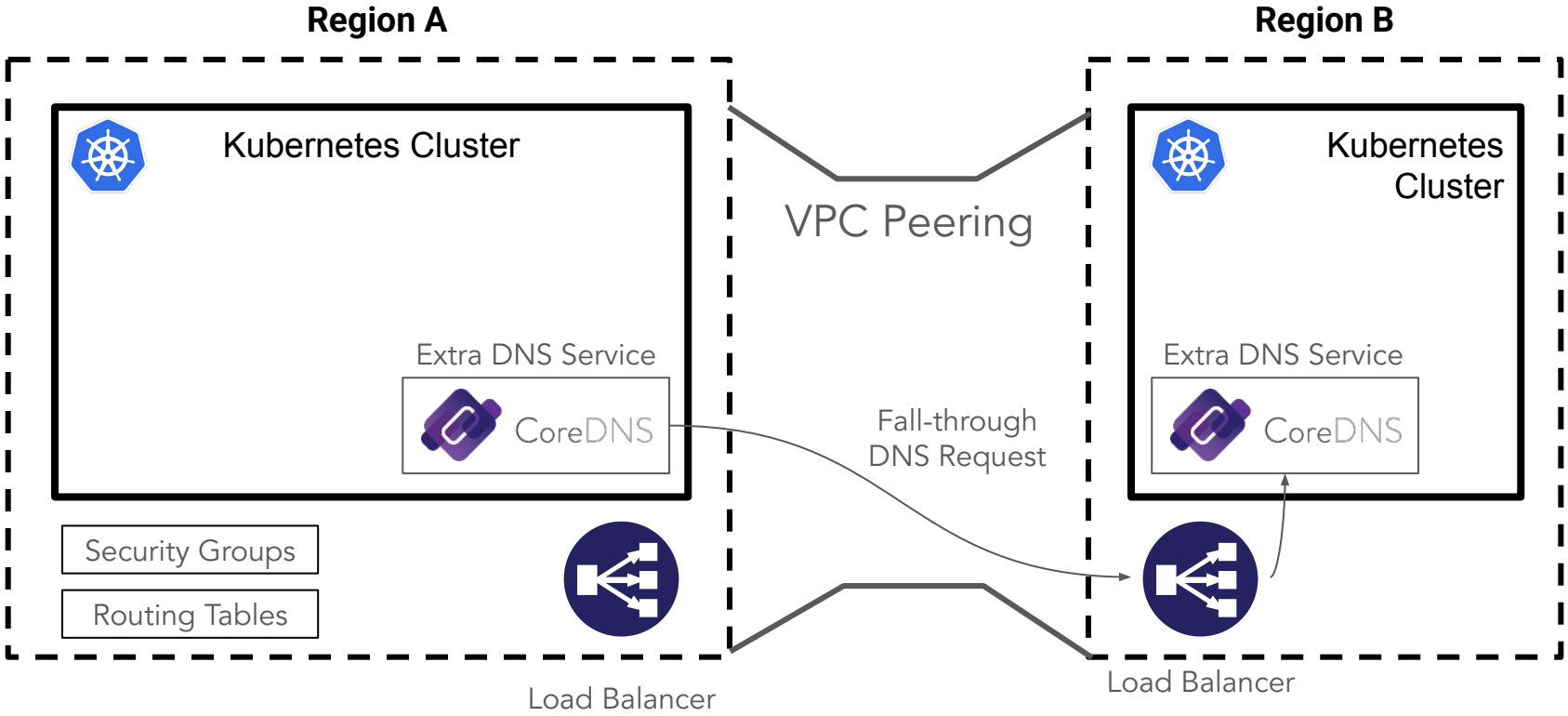
Cross-Region Replication

Protecting against failures in entire regions

Cross-region replication topology



Connecting regions across a single secure network

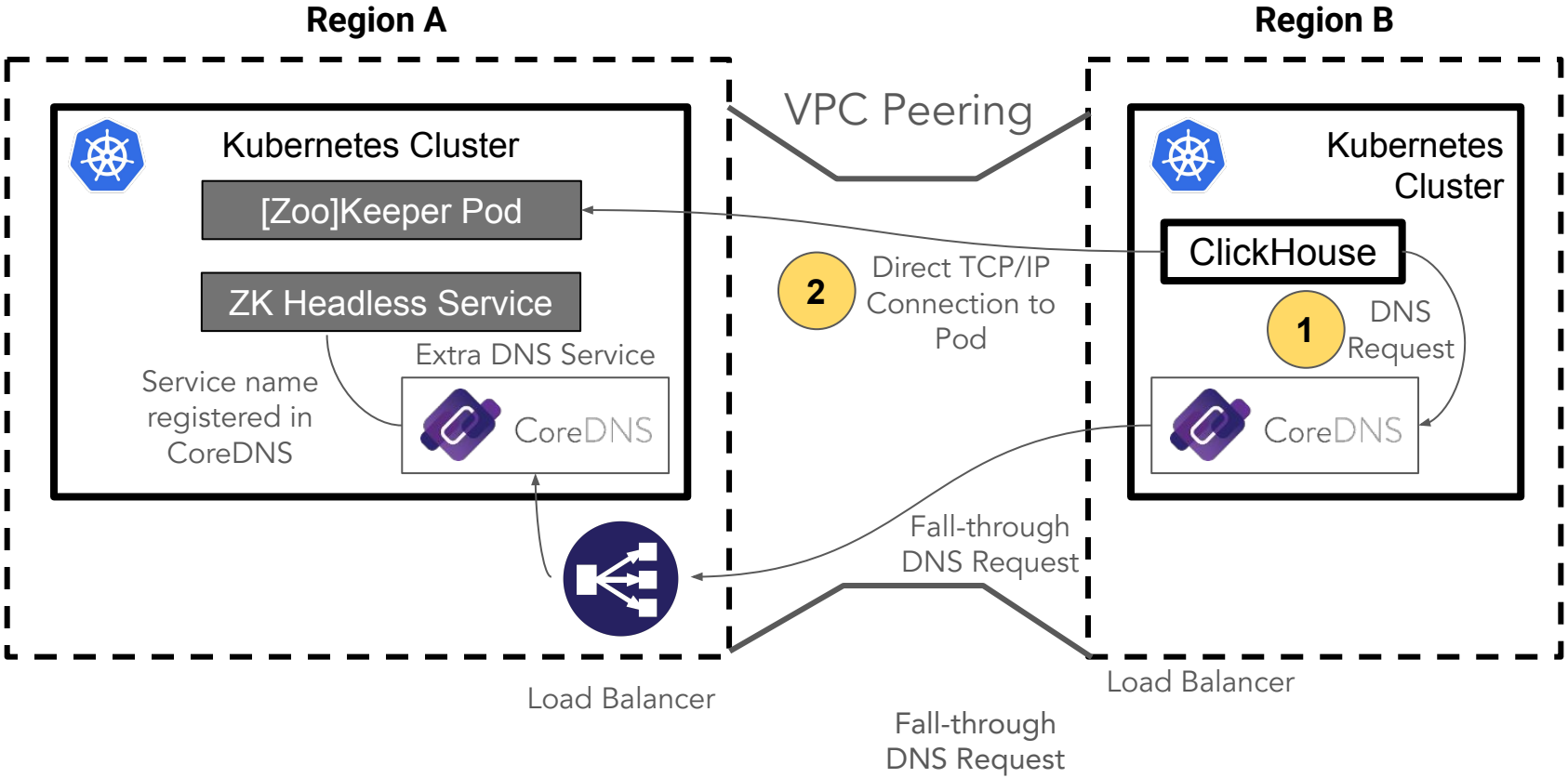


Updating CoreDNS to forward to peer DNS server(s)

Configmap settings for CoreDNS server:

```
kubernetes cluster.local in-addr.arpa ip6.arpa {
    pods insecure
    fallthrough cluster.local in-addr.arpa ip6.arpa
}
prometheus :9153
forward cluster.local <peering-dns-ip-1> <peering-dns-ip-2>
forward . /etc/resolv.conf
```

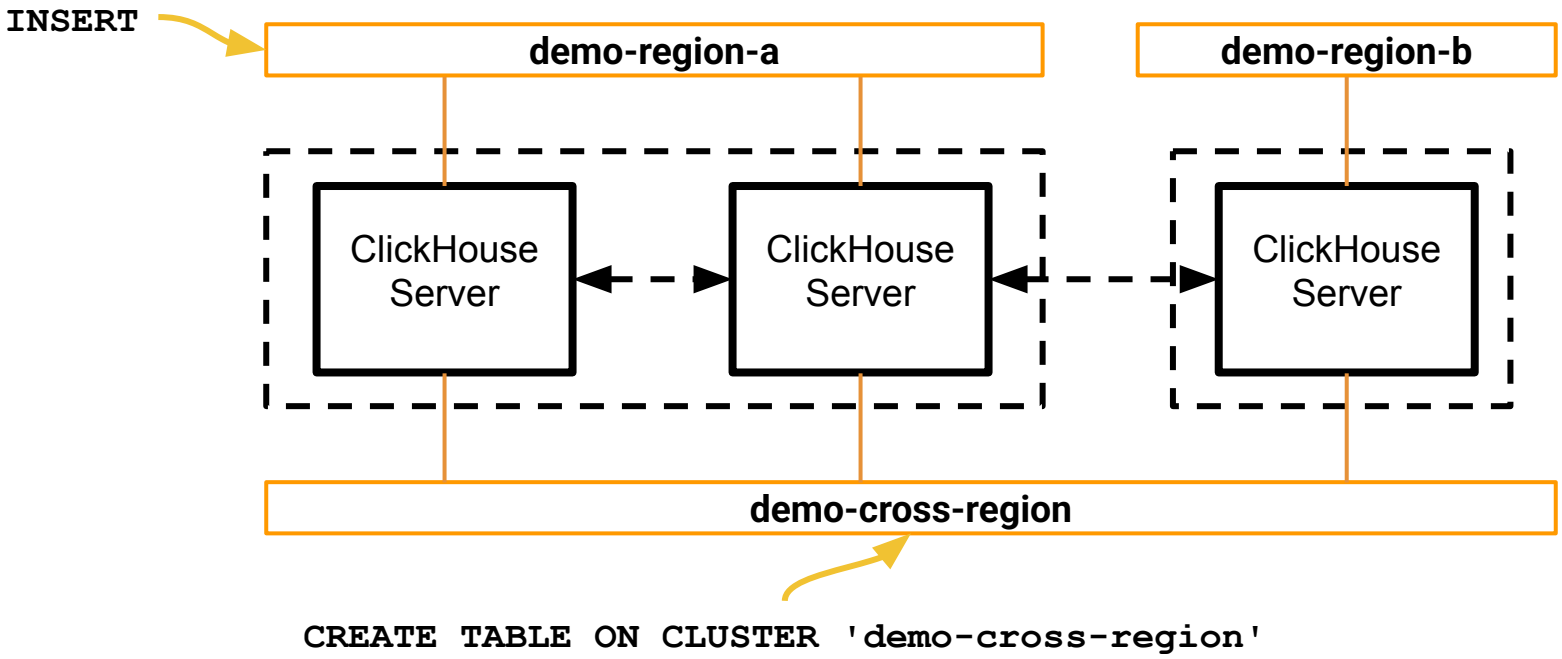
DNS lookup and TCP/IP connection between regions



Trick: Use templates for headless services by default

```
apiVersion: "clickhouse.altinity.com/v1"
kind: "ClickHouseInstallationTemplate"
metadata:
  name: headless-replica-service-template
spec:
  templating:
    policy: auto
  templates:
    serviceTemplates:
      - name: replica-service-template-1
        spec:
          type: ClusterIP
          clusterIP: None
          ports:
            - name: http
              port: 8123
              targetPort: 8123
            - name: client
              port: 9000
              targetPort: 9000
            - name: replica
              port: 9009
              targetPort: 9009
```

Trick: Define a cross-region cluster for DDL



Setting up a cross-region cluster definition in Kubernetes

```
spec:
  configuration:
    files:
      config.d/demo-cross-region.xml: |
        <clickhouse>
          <remote_servers>
            <demo-cross-region>
              <secret>demo-cross-region</secret>
              <shard>
                <internal_replication>true</internal_replication>
                <replica>
                  <host>chi-demo-us-west-1-0-0</host><port>9000</port>
                </replica>
                <replica>
                  <host>chi-demo-us-west-1-1-0</host><port>9000</port>
                </replica>
                <replica>
                  <host>chi-demo-us-west-2-0-0</host><port>9000</port>
                </replica>
              </shard> ...
```

Failing over (it's complicated)

Option 1: Recreate ZooKeeper/Keeper Metadata

1. Stop ingest processes (if running)
2. Start new [Zoo]Keeper ensemble.
3. Reconfigure ClickHouse to point to new ensemble.
4. Use SYSTEM RESTORE REPLICA to reload [Zoo]Keeper metadata from each server.
5. Restart ingest and resume full operations.

Downsides:

- tables will be R/O for a while
- need scripting to iterate through all RMTs

Option 2: Promote ZooKeeper observers to full members of ensemble.

1. Prerequisite:
   a. Set up ZooKeeper cluster with all nodes configured as observers.
2. Failover:
   a. Reconfigure ZooKeeper nodes to followers and restart.
   b. Reconfigure ClickHouse to point to new ensemble.
   c. Test health.

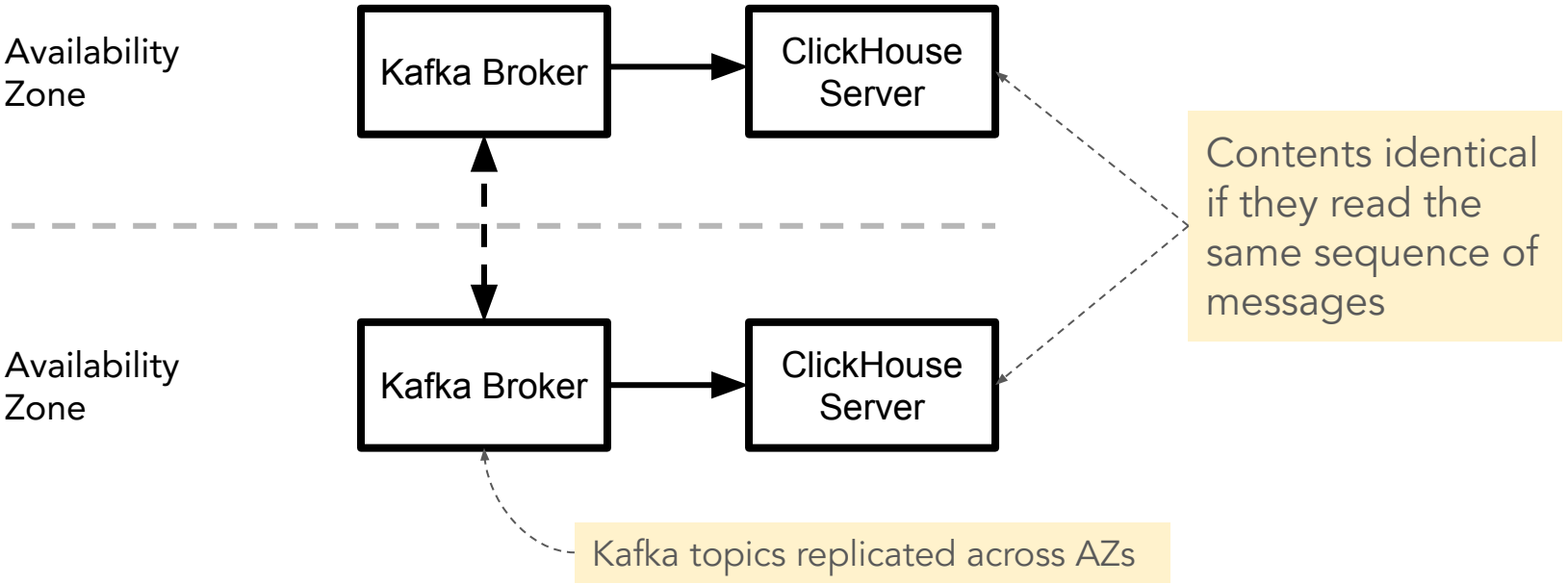
Downsides:

- tables are still in R/O while switching
- difficult to do with Keeper at this time.

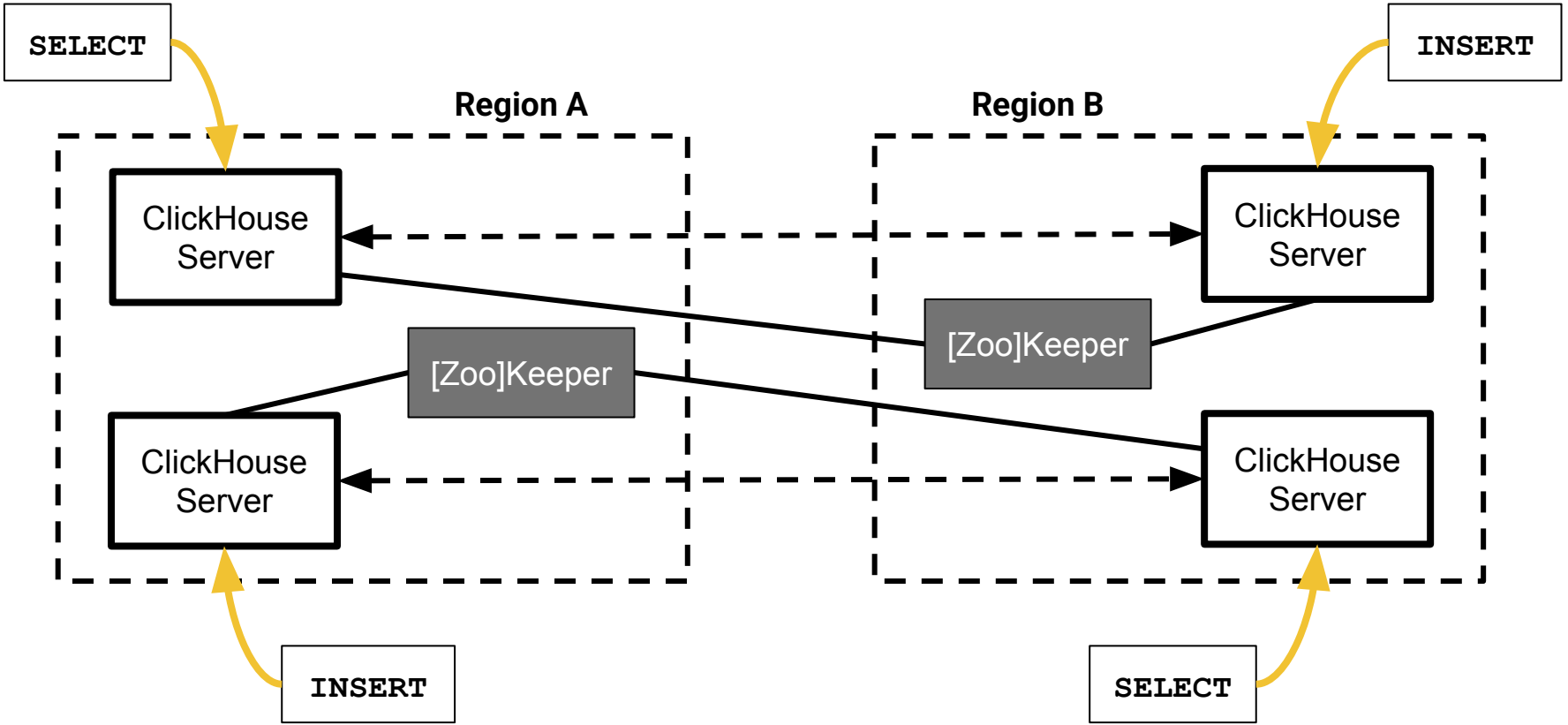
Independent Clusters and Other Tricks

Creative ways to spread replicas across zones and regions

You don't *need* ClickHouse replication to have replicas



An elegant topology with independent clusters



Wrap-up and Questions

Implementing DR for ClickHouse clusters

- Backup is cheapest and handles dropped tables; recovery is slow
- Cross-AZ DR works out-of-box with Altinity Kubernetes Operator
- Cross-region DR requires custom networking setup but also works
- Creative alternatives can reduce RPO / RTO and simplify management

- Altinity.Cloud is adding full support for cross-region DR by Q3!

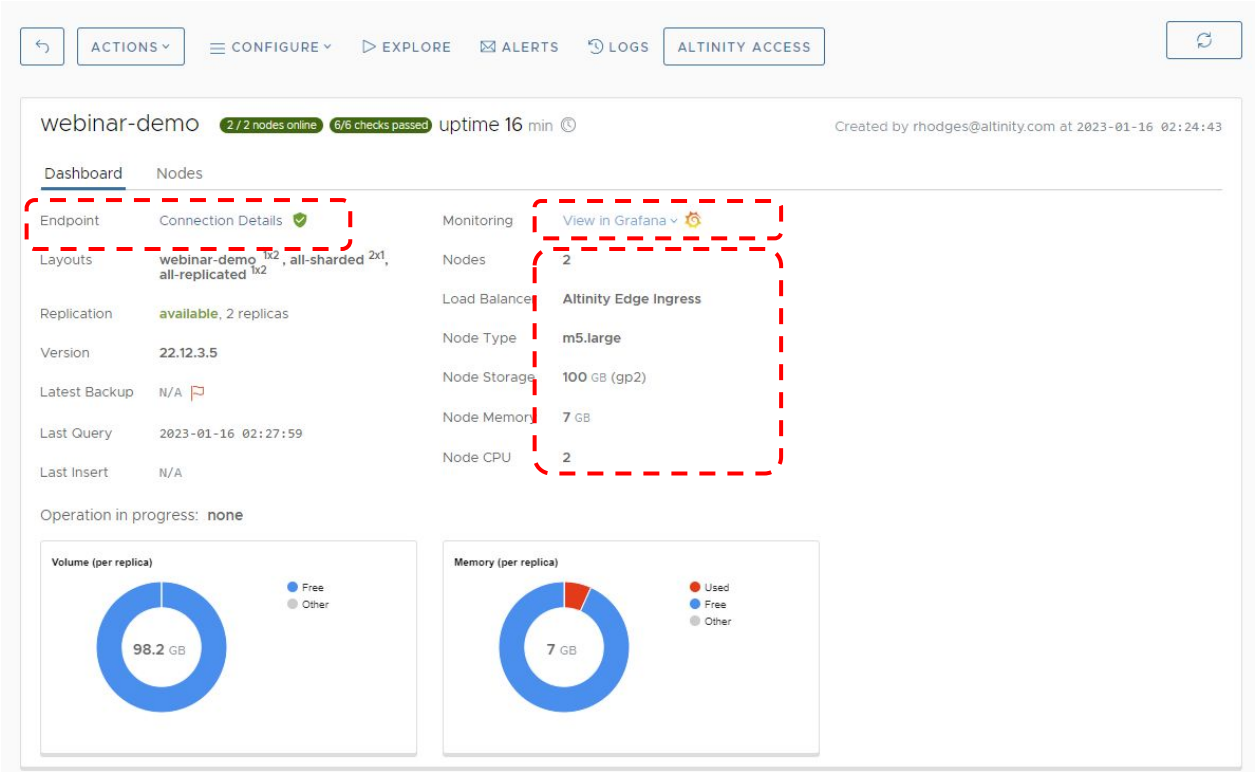
More information about ClickHouse DR

- Altinity Blog
- Altinity YouTube Channel
- ClickHouse Docs
- ClickHouse Source code

Altinity Blog: Setting up Cross-Region ClickHouse® Replication in Kubernetes

Altinity YouTube: Safety First! Using Altinity Backup for ClickHouse® for ClickHouse Backup and Restore

Looking for an easier way? Check out Altinity.Cloud.



Thank you!

Questions?

Website: https://altinity.com
Slack: https://altinity.com/slack