Unit 15: HA / DR

MarkLogic

Learning Objectives

- Describe backup and restore processes
- Describe journal archiving for point in time recovery
- Describe failover for High Availability
- Describe database replication for Disaster Recovery

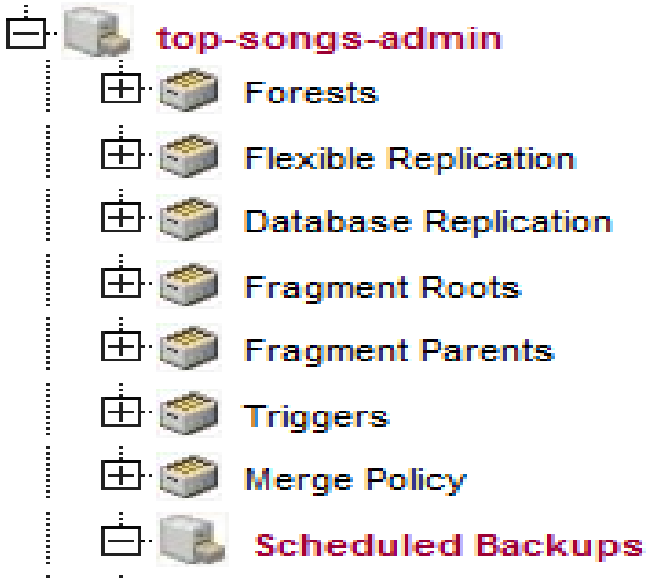
MarkLogic

Backup & Restore

- What is critical in order to restore?
  - Know your projects dependencies
- By default, when you back up a database you backup everything associated with it, including the following:
  - The configuration files
  - The Security database, including all of its forests
  - The Schemas database, including all of its forests
  - All of the forests of the database you are backing up

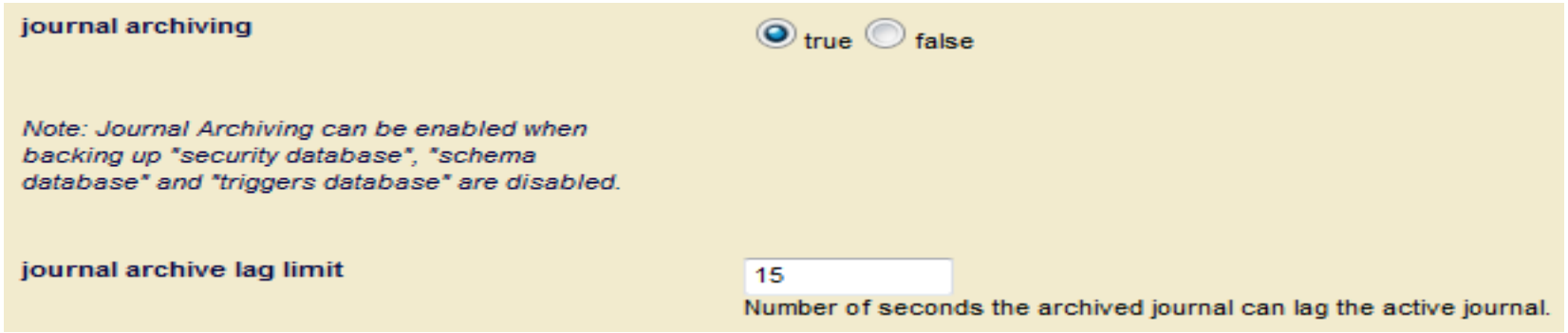
Backup & Restore

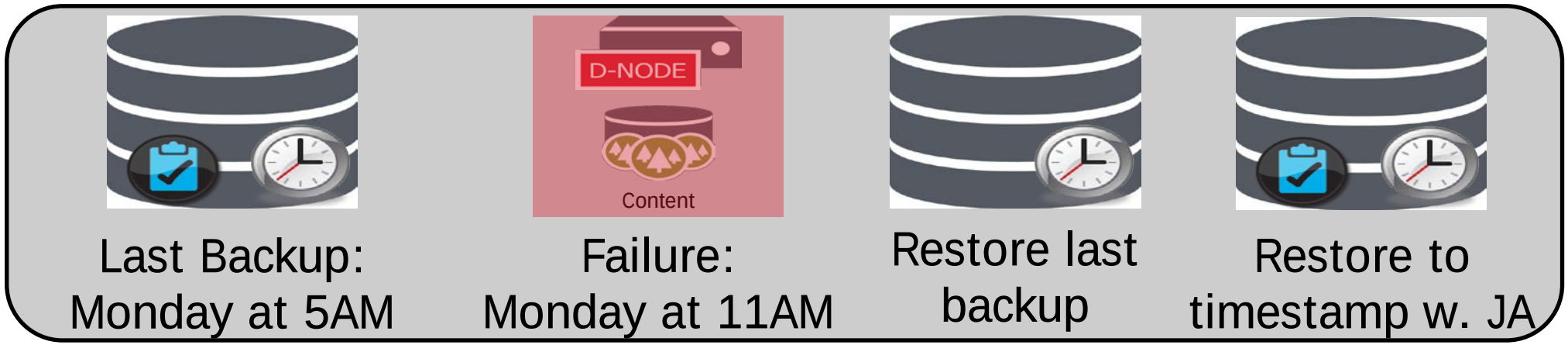
- Scheduled Backups



Point in Time Recovery

- Journal Archiving
  - Allows you to restore data to a point in time since the last backup

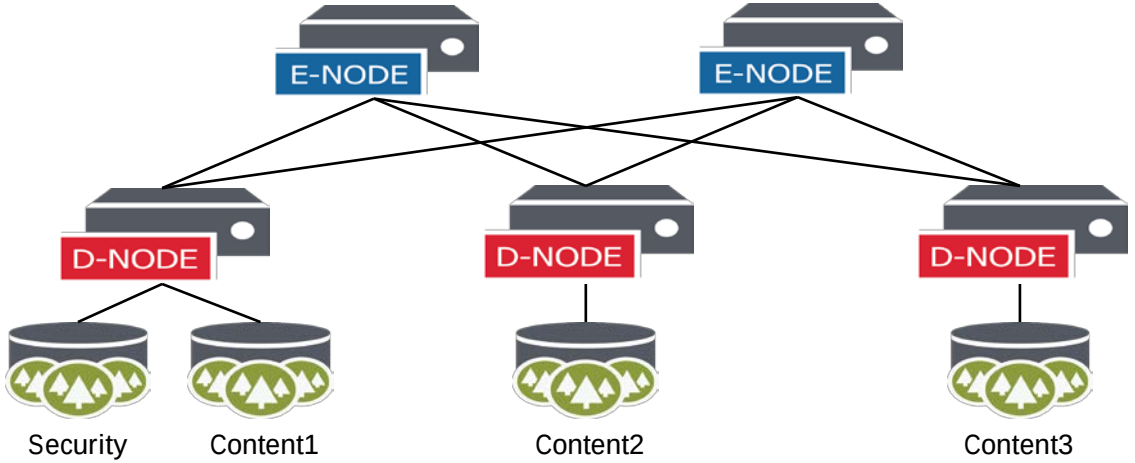




MarkLogic

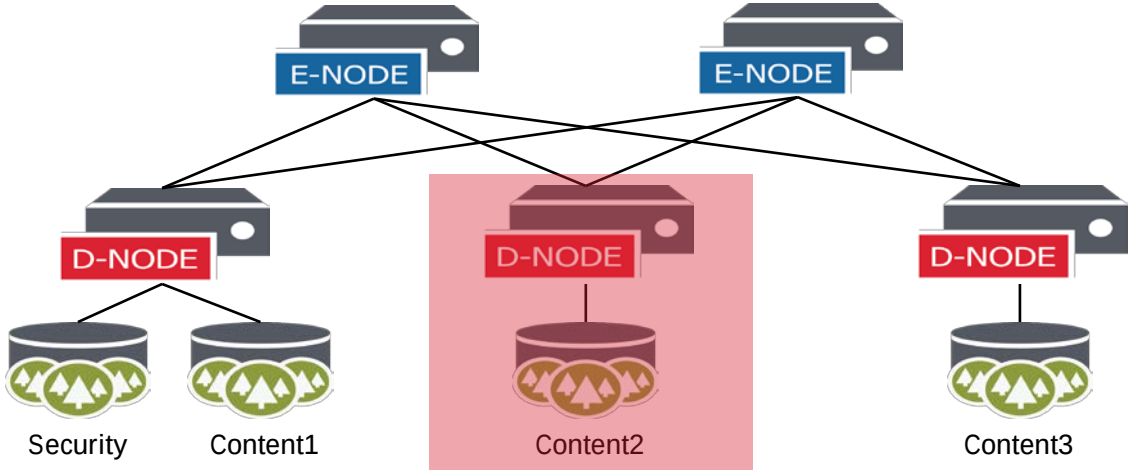
Failover

- Objectives:
  - Enable High Availability of a **Single Site** (Cluster)
  - If a Data Node goes down, ensure that its data is **automatically** made available elsewhere in the cluster
- Note:
  - Failover is managed at the forest level, not the database level
  - Failover can be configured for the same forest on multiple hosts to add additional redundancy
  - Forest replicas are subject to the same sizing guidelines as primaries
    - 2 CPUs per forest
    - 3x disk space
    - RAM



Failover

- Objectives:
  - Enable High Availability of a **Single Site** (Cluster)
  - If a Data Node goes down, ensure that its data is **automatically** made available elsewhere in the cluster
- Potential Causes of D Node Failure:
  - Operating system crashes
  - Hardware failures
  - Power failures
  - MarkLogic Server restarts
    - Note: Server restarts are usually very fast and don't exceed the timeout setting that triggers failover.



- Result without Failover:
  - Quorum of nodes?
    - Yes
  - Security Database Available?
    - Yes
  - All forests for the DB available?
    - No – database is not available

MarkLogic

Local-Disk Failover

- Primary forest is attached to the database.
- Create one or more replica forests for each primary forest.
- Replicas contain the exact same data as the primary and are kept up to date transactionally as updates to the forest occur.
- Replica forests should be on a different host than the primary.
- Both primary and replica forests have their own local-disk space allocated

MarkLogic

Local-Disk Failover

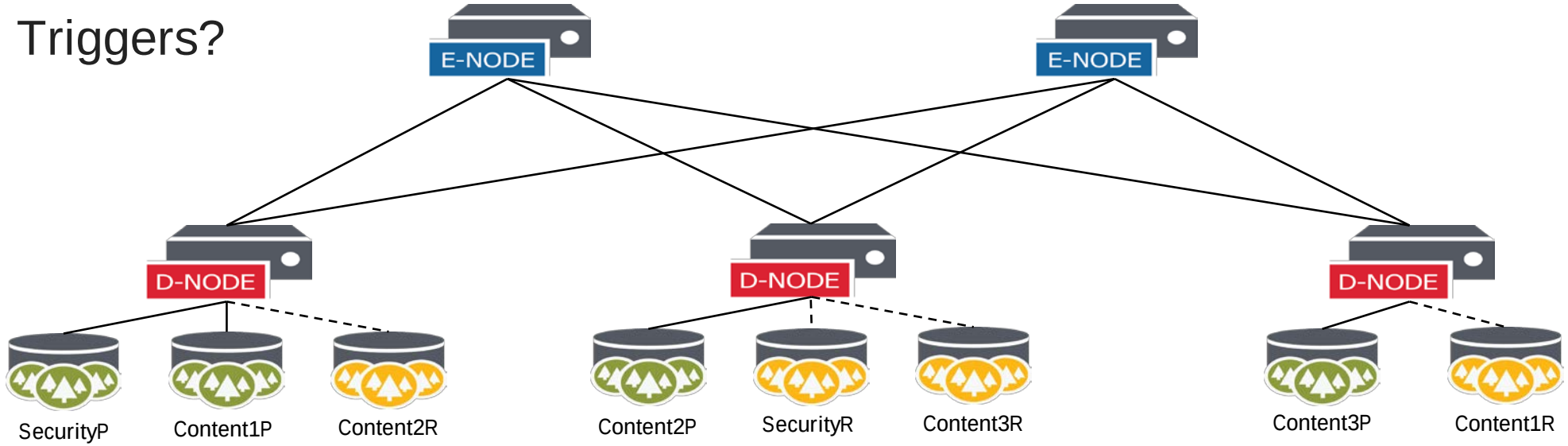
- Replication is at the journal level
  - Bulk synchronization for zero-day or after extended offline period
  - Ongoing journal replay once synchronized
  - Equivalent copy, not “bit for bit”

- Synchronous and transactional
  - Commit to master is commit to replica

MarkLogic

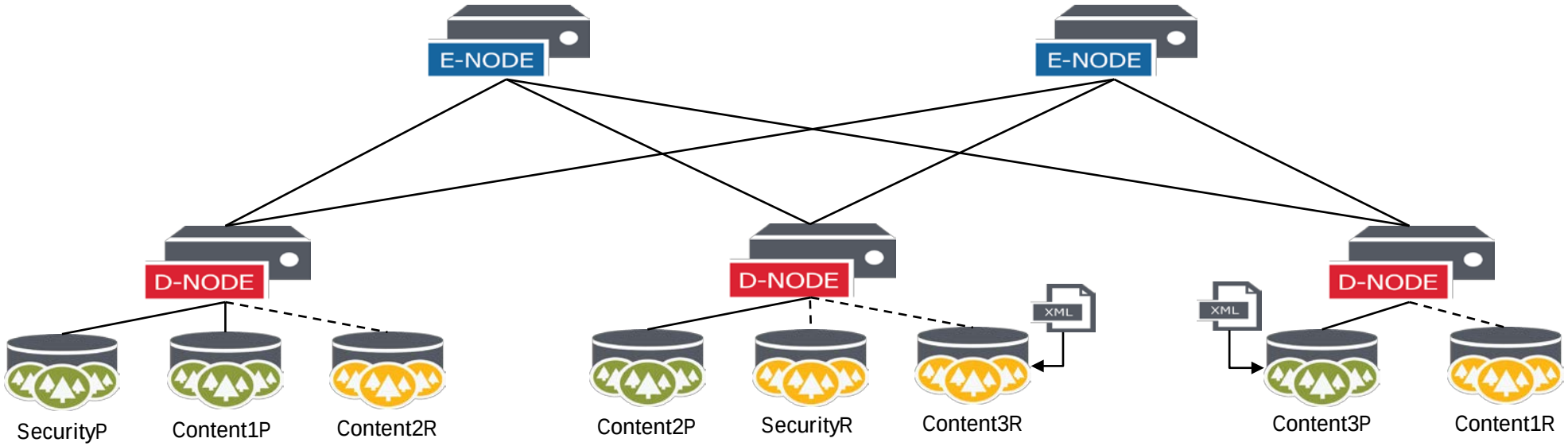
Local-Disk Failover Example

- All critical forests are configured with a replica on a separate host
  - Content & Modules
  - Security
  - Schemas?
  - Triggers?



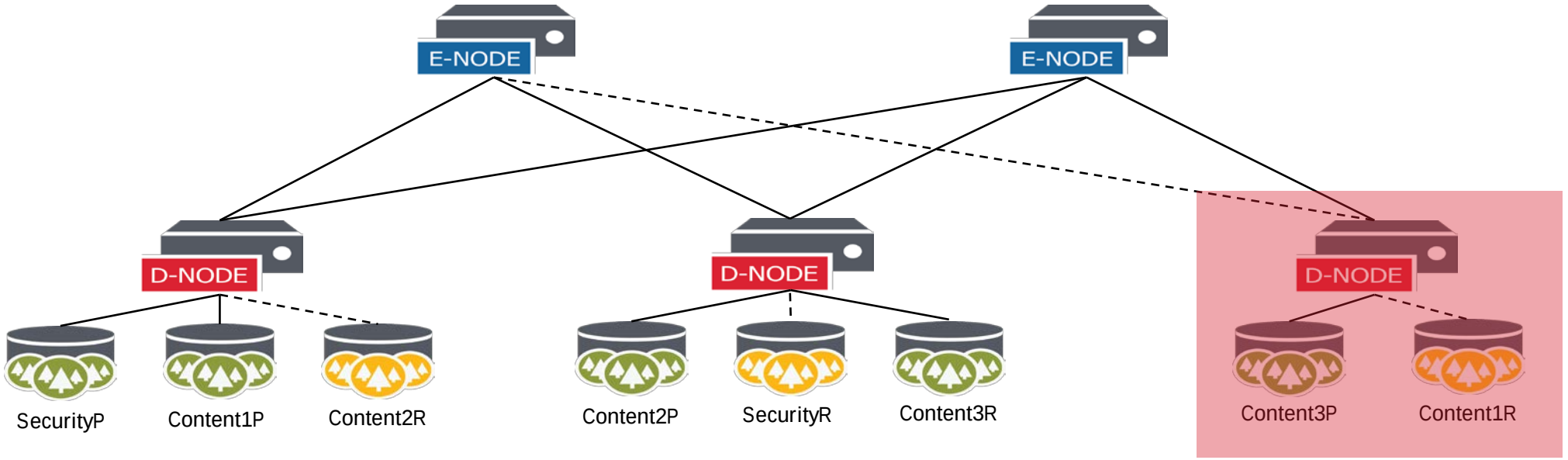
Local-Disk Failover Example

- Transactions occur synchronously between primary and replicas
  - /doc/myURI.xml gets inserted and lands in forest Content3P
  - The insert transaction is synchronously committed to Content3R



Local-Disk Failover Example

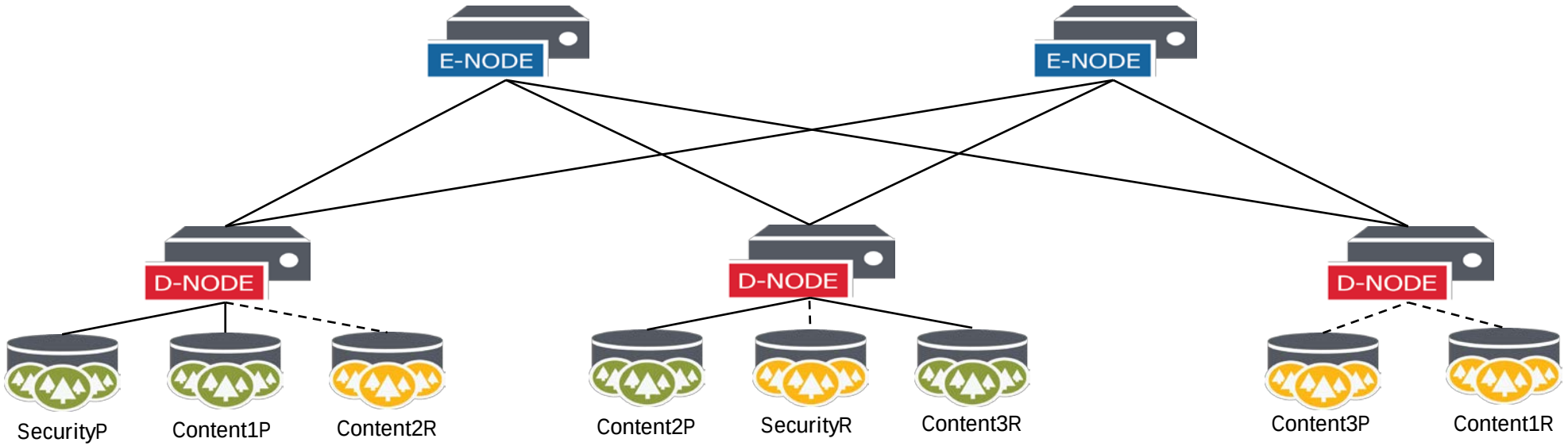
- What happens if a host goes down?
  - Replica forests will automatically mount
  - Life goes on – transactions, queries, etc.



MarkLogic

Local-Disk Failover Example

- What happens when a host comes back up?
  - The forest on the host that comes back up (what used to be the primary) will continue on as a replica.



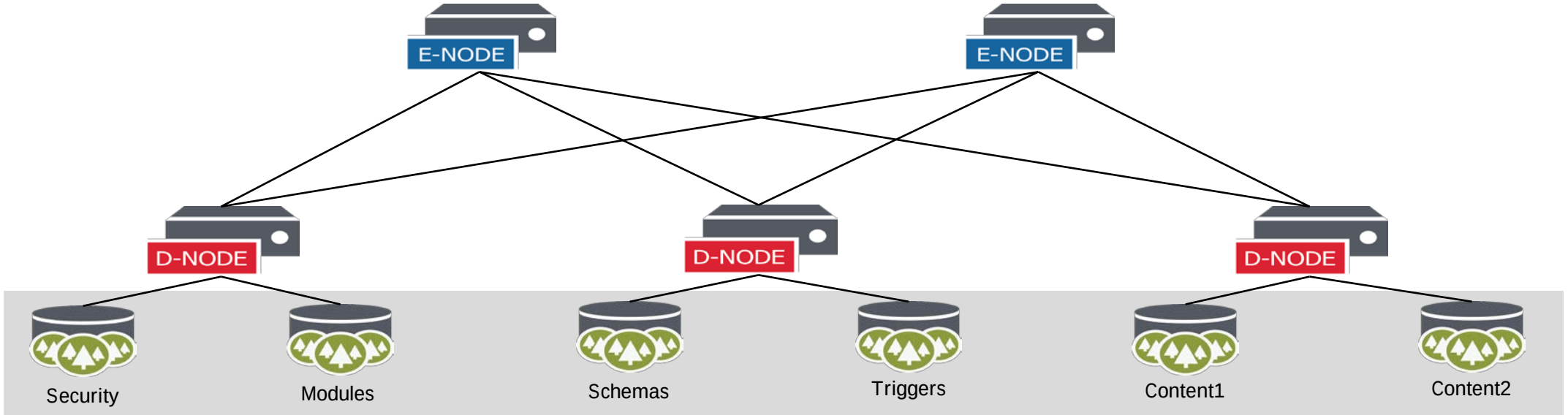
MarkLogic

Shared-Disk Failover

- Shared-disk failover uses a clustered filesystem
- The clustered filesystem must be available with the same path on each host that is configured as a failover host.
- If a host configured for shared disk failover goes down, another host can take over the assignment of that forest.

Shared-Disk Failover

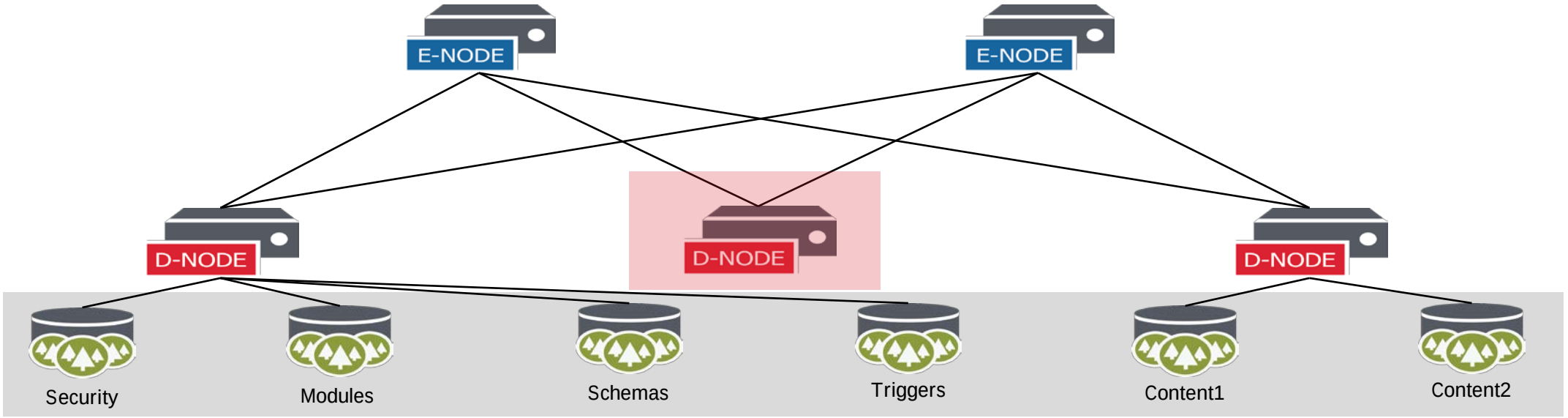
- Shared-disk failover uses a clustered filesystem
- No replica forests
- The clustered filesystem must be available with the same path on each host that is configured as a failover host.
- If a host configured for shared disk failover goes down, another host can take over the assignment of that forest.



MarkLogic

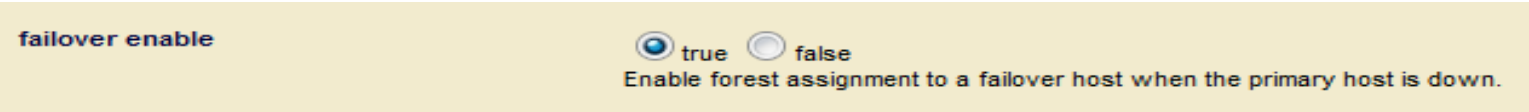
Shared-Disk Failover

- If a host goes down, and it contains a forest configured for failover, another host will attach to that forest
- When a host comes back up, it does not get the forest back.

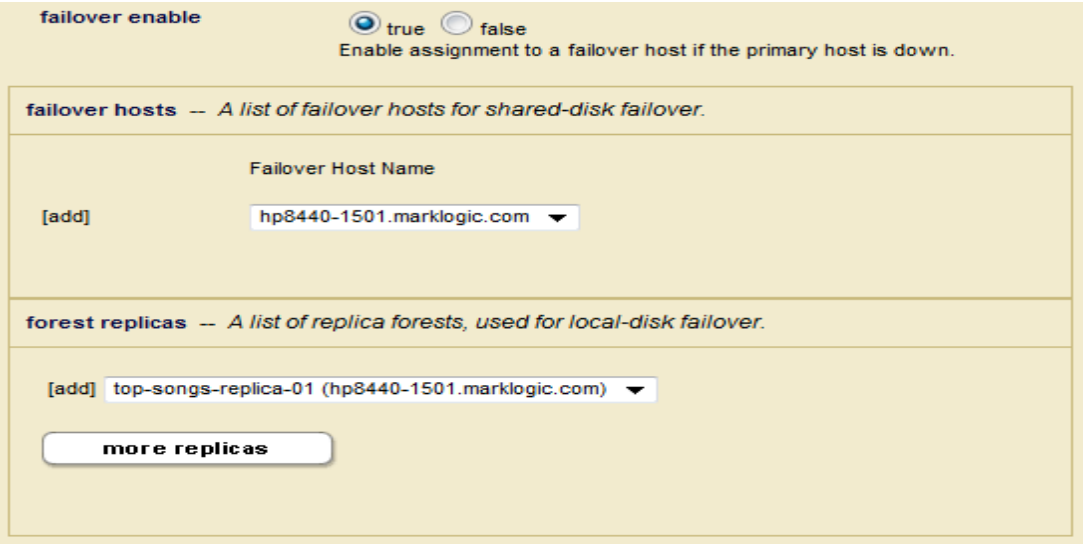


Failover: Configuration

- Enable Failover at the Group Level



- Create the primary forest and attach to database
- Create the replica forest (leave unattached)
- Configure the primary forest for failover:



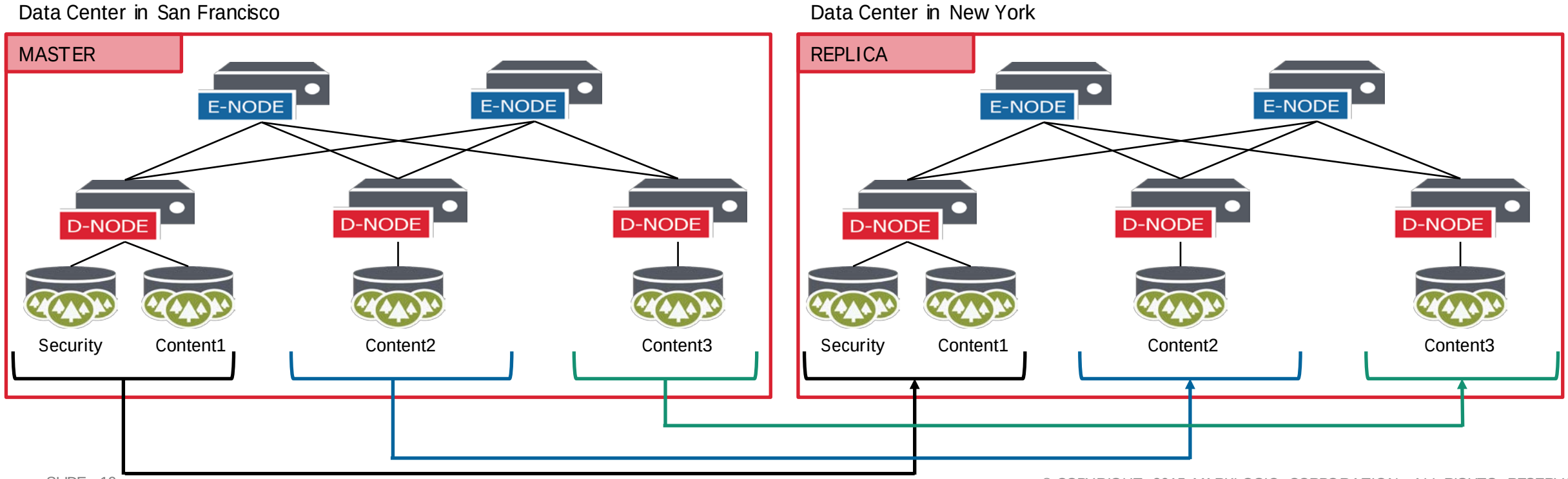
MarkLogic

Failover: Forest Mount States

- SYNC Replicating**
  - Forest is a standby replica that is synchronously updated with the primary forest (the primary being the forest in an OPEN state)
- ASYNC Replicating**
  - The replica is catching up to the primary. Once caught up it will be in the SYNC state.
  - Example: When you add a new replica it will be in this state initially.
- WAIT Replicating**
  - The forest is waiting to get into one of the other replicating states
  - Example: When a host starts up, it will be in this state initially.
- OPEN
  - Forest is available and acting as the primary.
- ERROR
  - Forest is not available due to error.
  - Example: Insufficient disk space
- ** only applies to Local-Disk failover

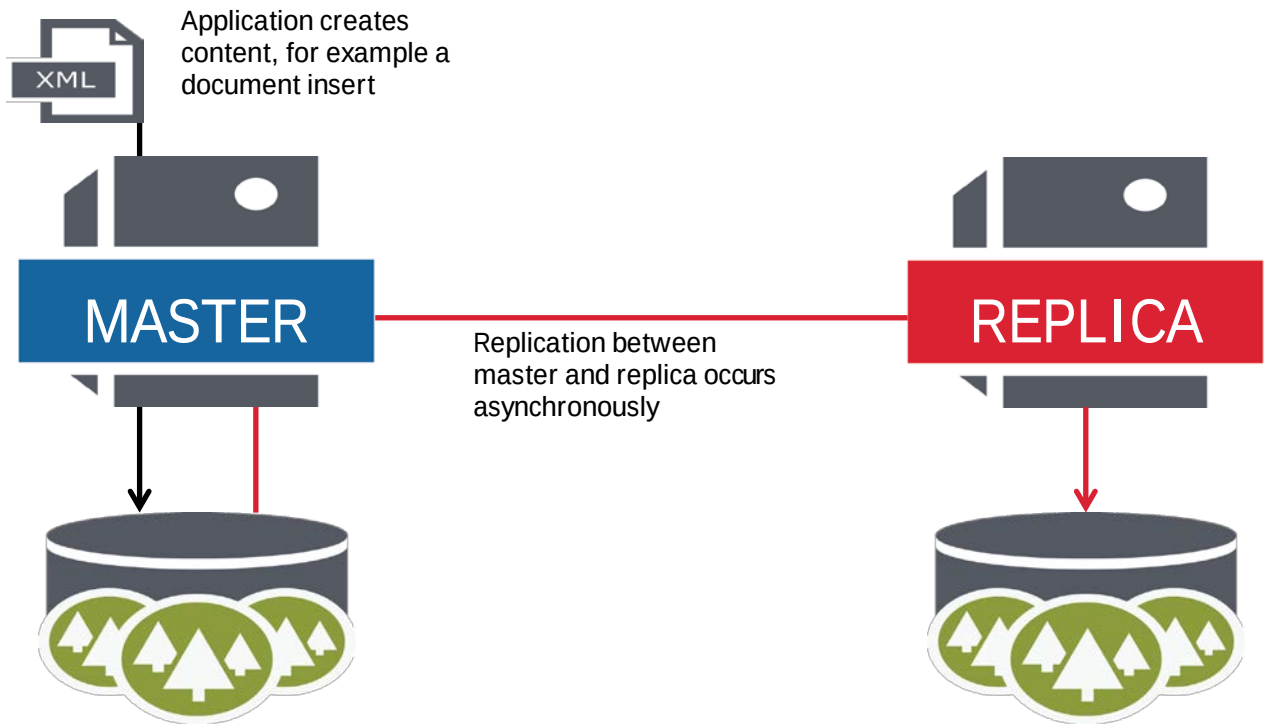
Database Replication

- Disaster Recovery. Protect against the loss of an entire cluster.
- Operates at the forest level by copying journal frames in the master and replaying them in the replica. Host : Host connections.
- Replica databases can be queried but not updated



Flexible Replication

- Original content is created on the Master MarkLogic Server.
- Replication copies the content to one or more Replica MarkLogic Servers.
- The Master and Replica servers are typically in different clusters, which may be in the same location or in different locations.



MarkLogic

Let's think about it…

- How would you protect against a single (or potentially multiple) node failure in a cluster?

- How would you ensure as little data loss as possible after some failure (such as disk loss, human error)?

- How would you safeguard against a disaster that took down an entire cluster or data center?

MarkLogic

Let's think about it…

- How would you protect against a single (or potentially multiple) node failure in a cluster?
  - Failover (local or shared disk)
- How would you ensure as little data loss as possible after some failure (such as disk loss, human error)?
  - Scheduled backups
  - Journal archiving
- How would you safeguard against a disaster that took down an entire cluster or data center?
  - Database replication

Labs: Unit 15

Exercise 1: Explore Inside MarkLogic Server

Unit Review Question 1:

What is the minimum number of nodes for a highly available cluster?

1. 2 nodes
2. 3 nodes
3. 5 nodes
4. None of the above

Unit Review Question 1:

What is the minimum number of nodes for a highly available cluster?

1. 2 nodes
2. **3 nodes**
3. 5 nodes
4. None of the above

Unit Review Question 2:

Point in time recovery is enabled through:

1. Database backups and failover
2. Failover and database replication
3. Database backups and journal archiving
4. Journal archiving

Unit Review Question 2:

Point in time recovery is enabled through:

1. Database backups and failover
2. Failover and database replication
3. **Database backups and journal archiving**
4. Journal archiving

Unit Review Question 3:

With shared disk failover, replica forests are maintained:

1. True
2. False

Unit Review Question 3:

With shared disk failover, replica forests are maintained:

1. True
2. **False**

Unit Review Question 4:

With local disk failover, replica forests are maintained. To increase redundancy, multiple replicas for each forest may be distributed throughout the cluster:

1. True
2. False

Unit Review Question 4:

With local disk failover, replica forests are maintained. To increase redundancy, multiple replicas for each forest may be distributed throughout the cluster:

1. **True**
2. False