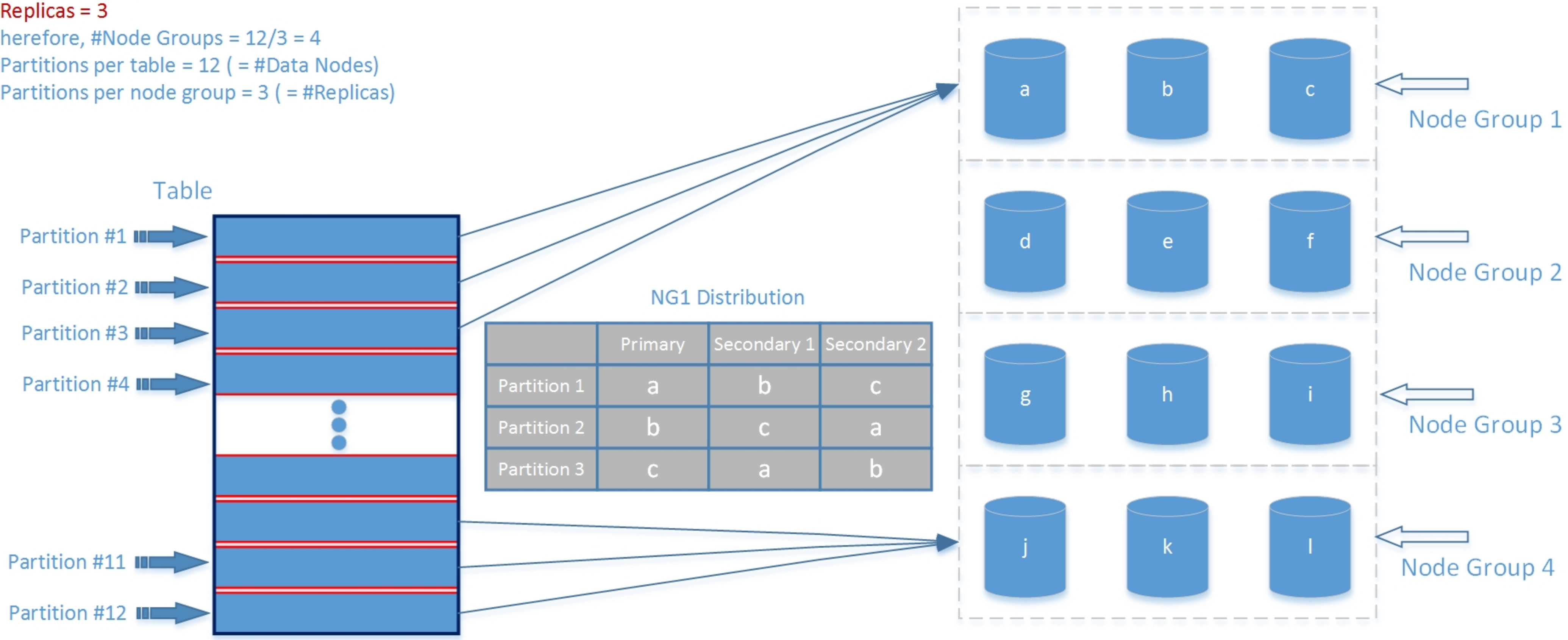
# Fault Tolerant MySQL

- Source (before my edits) https://blogs.msdn.microsoft.com/opensourcemsft/2015/11/04/mysql-cluster-iaas-best-practices-for-azure-high-availability-fault-tolerance-scalability/

# Fault Tolerant MySQL

#Data Nodes = 12
#Replicas = 3
Therefore, #Node Groups = 12/3 = 4
#Partitions per table = 12 ( = #Data Nodes)
#Partitions per node group = 3 ( = #Replicas)



NG1 Distribution

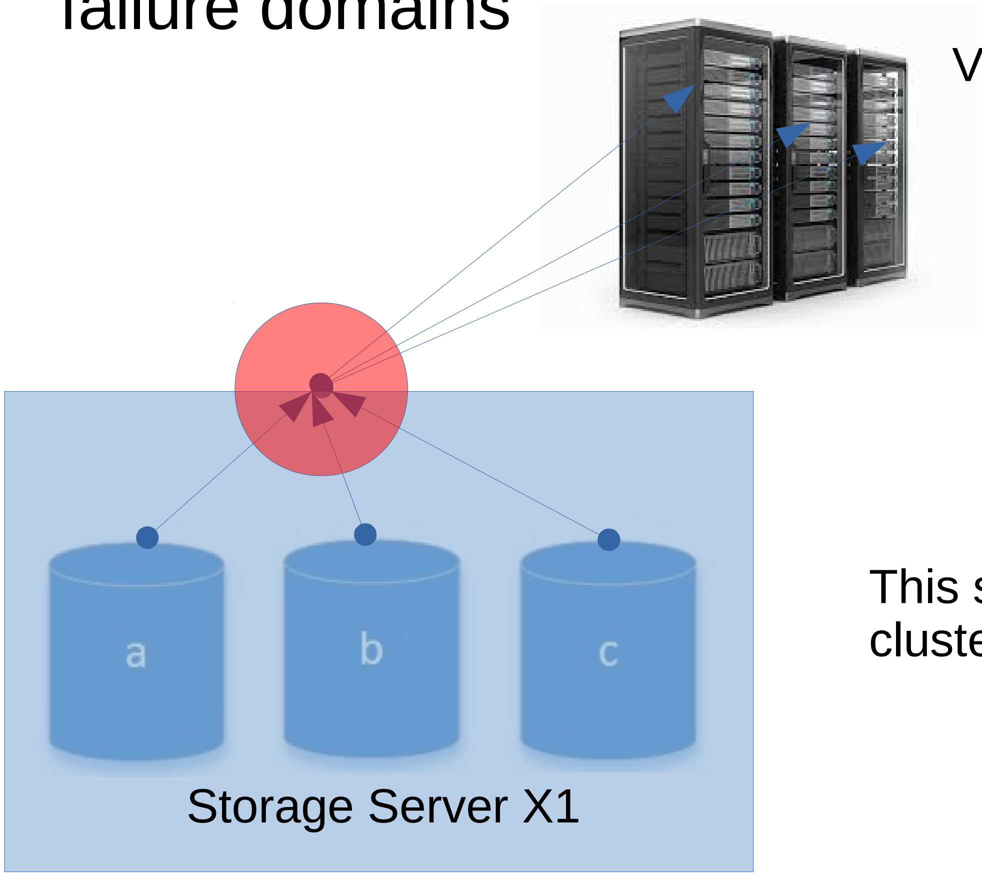
| | Primary | Secondary 1 | Secondary 2 |
|---|---|---|---|
| Partition 1 | a | b | c |
| Partition 2 | b | c | a |
| Partition 3 | c | a | b |

# Fault Tolerant MySQL (Node-Internal Volumes)

- In previous picture consider Node Group 1 with Volumes a-c
- If these Volumes are based on internal drives then fault tolerance is achieved by ensuring that the associated Publishing Nodes are in different failure domains as described in https://github.com/container-storage-interface/spec/issues/7
  - The failure domain of a Volume should match that of the Node that is able to Publish it. Basically they fail together

# Fault Tolerant MySQL (Node-External Volumes)

- Again consider Node Group 1 with Volumes a-c but place them in a single external storage system, like NetApp/Dell NFS/SAN deployments
- The Publishing nodes are each in different racks with different failure domains



Storage Server X1

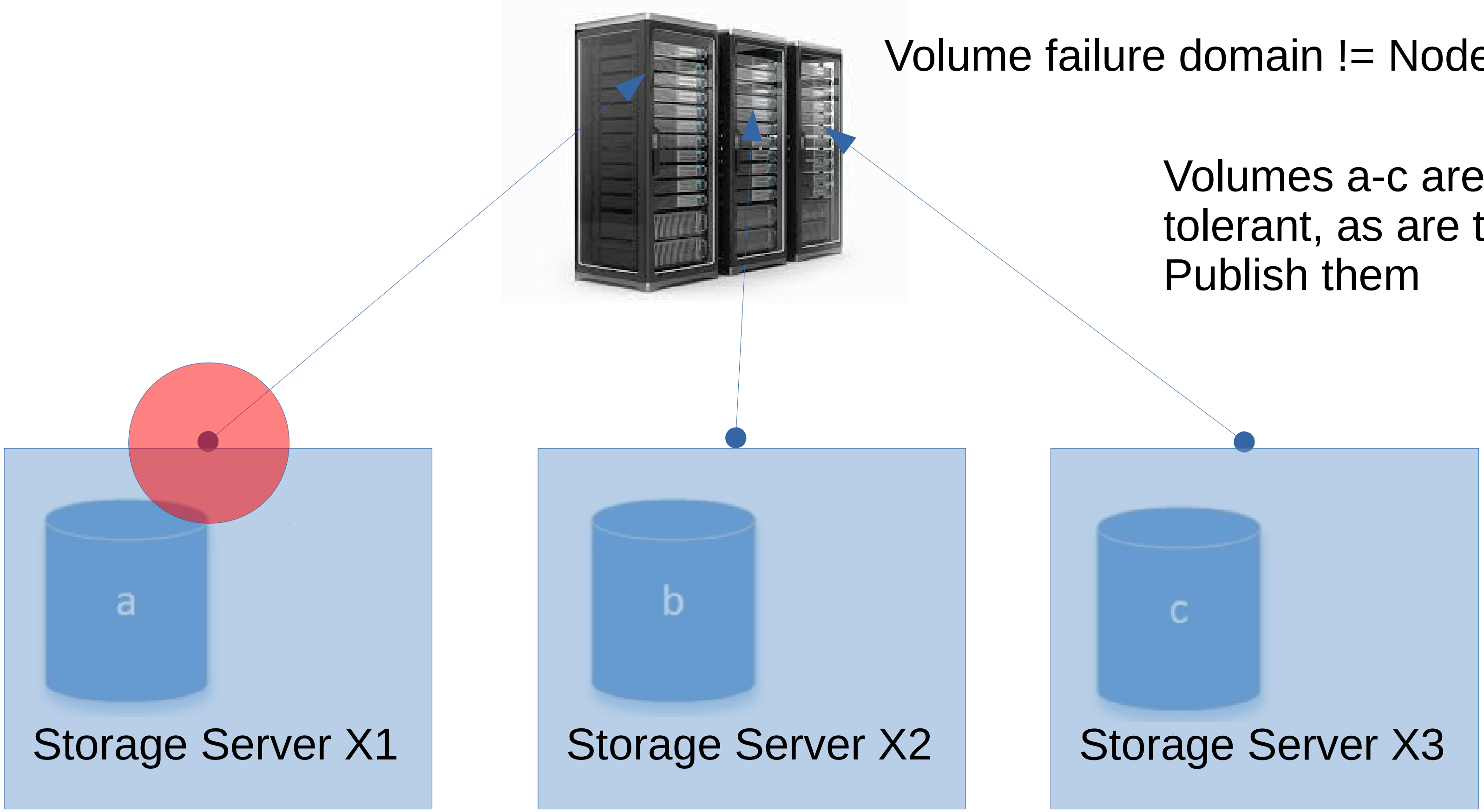
Volume failure domain != Node failure domain

Volumes a-c are NOT fault tolerant even though Publishing nodes are in different domains

This storage server is not in the same racks as cluster nodes above

# Fault Tolerant MySQL (Node-External Volumes)

- The Volumes are each in different external storage systems with different failure domains



a

b

c

Storage Server X1

Storage Server X2

Storage Server X3

Volume failure domain != Node failure domain

Volumes a-c are now fault tolerant, as are the Nodes that Publish them

These storage servers are not in the same racks as cluster nodes above