

## KumoScale Deployment for Bare-Metal Using Ansible

This section explains how a KumoScale solution is implemented for bare-metal under Ansible. It is useful to review this section before using the modules and playbooks available in the KumoScale Ansible package.

### Resiliency using Cross-Domain Data Replication

KumoScale™ software supports resiliency using Cross-Domain Data Replication (CDDR). This capability is based on data replication and enables users to flexibly specify the number of replicas on a per-storage-class, or per application, basis. In CDDR, write operations are processed synchronously in parallel (i.e., forwarded to each replica), and then acknowledged back to the application only once it has been committed to non-volatile media at each replica.

### Provisioning and Monitoring

Provisioning and monitoring the resiliency configuration are supported by KumoScale software installed separate from the KumoScale cluster:

**Provisioning** typically runs as a service on a server designated by the data center DevOps team. The main functions of the KumoScale Provisioner service are to:

- Maintain the KumoScale software deployed inventory information.
- Maintain information regarding volume mapping and other statistics across all KumoScale storage nodes.
- Receive storage provisioning requests from the provisioning framework, specifying the required capacity and the storage class.
- Determine the optimal provisioning scenario across the KumoScale software inventory.
- Provision the requested storage resources.

**Monitoring** is conducted by the KumoScale agent software running on the initiator. It is based on a telemetric feed that is sent to a time-series database (TSDB). A solution is continuously monitored, and alerts are generated upon certain events. Telemetric data and certain events, such as degradation in the state of a resilient volume are sent to and stored in a Syslog server. This topic is discussed further under *Failure Recovery* in [Maintenance and Monitoring with KumoScale Ansible](#).

### Example KumoScale Deployment Using Ansible

An example of a KumoScale deployment illustrating CDDR is shown in Figure 1. In this configuration:

- Triple volume replicas are provisioned in separate failure domains.
- ToR switches connect KumoScale storage nodes to application servers and connect different racks via a spine router.
- An Ansible script is executed to provision this solution.
- The KumoScale Provisioner service interfaces with Ansible playbooks and provisions storage across the appropriate KumoScale storage nodes.
- The volume replicas are maintained in sync and are re-synced upon recovery from failure in one of the domains.
- A monitoring and alerting application based on a TSDB maintains the state of the solution and generates alerts as required.

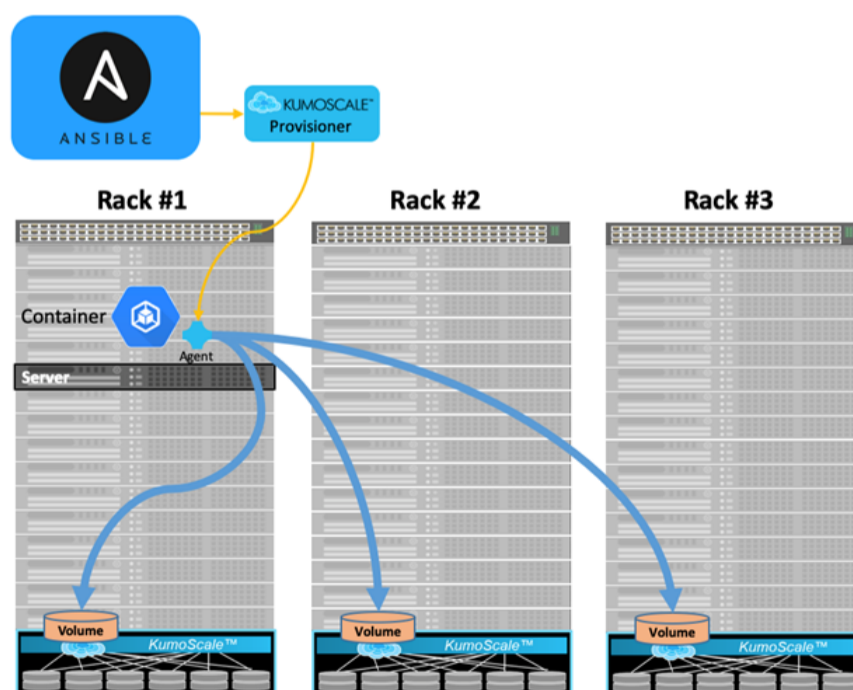


Figure 1. Example Deployment of a Cross-Domain Resilient Volume

### Failure Recovery

The KumoScale solution is designed to recover from several failure scenarios:

1. Short-term disconnect between application server and storage volume.
2. Long-term disconnect, such application server boot.
3. Long-term and permanent disconnect due to a network issue, such as ToR maintenance, KumoScale software upgrade, Rack powerdown.

Data resiliency is maintained via a Linux RAID module in the application server. This layer is managed and monitored as part of the resiliency solution. It can be configured to maintain two or three volume replicas, and to re-sync the replicas upon recovery from a failure.

**Note:** The KumoScale solution under Ansible makes use of a standard Linux™ RAID module (i.e. RAID 0 or RAID 1) located on the initiator servers. The Linux RAID service is configured and monitored as part of the solution. Due to the fact that volume resiliency is implemented at a data center level across multiple failure domains, KumoScale software does not implement resiliency on the drive (i.e. using RAID 5 or RAID 6).

Next: [Ansible Module Installation and Configuration](#)

---

---