Basic Configuration

Contents

This document captures the parameters that are commonly tweaked when setting up a new cluster, but it is highly advisable to review the detailed Configuration Files document before moving a cluster into production.

All configuration values discussed here are managed via the `app.config` file on each node, and a node must be restarted for any changes to take effect.

It is advisable to make as many of the changes below as practical **before** joining the nodes together as a cluster. Once `app.config` has been configured on each node, refer to Basic Cluster Setup to complete the clustering process.

Use riak-admin member-status to determine whether any given node is a member of a cluster.

Ring size

The ring size in Riak parlance is the number of data partitions which comprise the cluster. This quantity impacts the scalability and performance of a cluster, and importantly, **it must be established before the cluster starts receiving data**.

If the ring size is too large for the number of servers, disk I/O will be negatively impacted by the excessive number of concurrent databases running on each server.

If the ring size is too small, the servers' other resources (primarily CPU and RAM) will go underutilized.

See Planning for a Riak System and Scaling and Operating Riak Best Practices for more details on choosing a ring size.

The steps involved in changing the ring size depend on whether the servers (nodes) in the cluster have already been joined together.

Cluster already in active use

If you have a cluster in use and need to preserve its data while resizing the ring, you may either migrate your data to a new cluster or contact Basho to discuss the possibility of performing a dynamic ring resizing operation.

Cluster joined, but no data needs to be preserved

1. Uncomment the `ring_creation_size` parameter (by removing the `%` that precedes it) in the `riak_core` section in `app.config` on each node and set the appropriate quantity
2. Stop all nodes
3. Remove the ring data file on each node (see Backing up Riak for the location of this file)
4. Start all nodes
5. Re-add each node to the cluster (see Adding and Removing Nodes) or finish reviewing this document and proceed to Basic Cluster Setup

New servers, have not yet joined a cluster

1. Uncomment the `ring_creation_size` parameter (by removing the `%` that precedes it) in the `riak_core` section in `app.config` on each node and set the appropriate quantity

2. Stop all nodes
3. Remove the ring data file on each node (see Backing up Riak for the location of this file)
4. Finish reviewing this document and proceed to Basic Cluster Setup

Verifying ring size

The `riak-admin` command can verify the ring size:

```
$ sudo /usr/sbin/riak-admin status | egrep ring
ring_members : ['riak@10.160.13.252']
ring_num_partitions : 8
ring_ownership : <<"[{'riak@10.160.13.252',8}]">>
ring_creation_size : 8
```

If `ring_num_partitions` and `ring_creation_size` do not agree, that means that the `ring_creation_size` value was changed too late and the proper steps were not taken to start over with a new ring.

Note that Riak will not allow two nodes with different ring sizes to be joined into a cluster.

Backend

The most critical decision to be made is the backend to use. The choice of backend strongly influences the performance characteristics and feature set for a Riak environment.

See Choosing a Backend for a list of supported backends; each referenced document includes the necessary configuration bits.

As with ring size, changing the backend will result in all data being effectively lost, so spend the necessary time up front to evaluate and benchmark backends.

If still in doubt, consider using the Multi backend for future flexibility.

If you do change backends from the default (typically Bitcask) make certain you change it across all nodes. It is possible but generally unwise to use different backends on different nodes; this would limit the effectiveness of backend-specific features.

Default bucket properties

Bucket properties are also very important to performance and behavioral characteristics.

The properties for any individual bucket can be configured dynamically, but default values for those properties can be defined in `app.config`, in the `riak_core` section.

Below is an example of setting default bucket properties (admittedly, in this example to the values which Riak would define to them in the absence of such a configuration).

```
{default_bucket_props, [
    {n_val,3},
    {r,quorum},
    {w,quorum},
    {allow_mult,false},
    {last_write_wins,false},
    ]}
```

In short: replicate data 3 times, require that 2 of those 3 replicas respond before any read or write is considered successful, and do not present conflicting values to the application for resolution.

The `r` and `w` values can be overridden with each request, and few users need to change `n_val`, but choosing an appropriate value for `allow_mult` is vital for a robust application.

For more on the implications of these settings, please see Eventual Consistency, Replication, and the Basho blog series, "Understanding Riak's Configurable Behaviors": Part 1, Part 2, Part 3, Part 4, and the Epilogue.

If the default bucket properties are modified in `app.config` and the node restarted, any existing buckets will **not** be directly impacted, although the mechanism described in HTTP Reset Bucket Propertiescan be used to force them to pick up the new defaults.

System tuning

Please review the following documents before conducting anybenchmarking and/or rolling out a live production cluster.

- Open Files Limit
- File System Tuning
- Linux Performance Tuning
- AWS Performance Tuning
- Configuration Files

Joining the nodes together

Please see Basic Cluster Setup for the cluster creation process.

These May Also Interest You

- Installing Custom Code
- Production Checklist
- Configuring an S3 Client
- Configuring DragonDisk
- Configuring Riak CS
- Configuring Riak for CS