Adding and Removing Nodes

Contents

This page describes the process of adding and removing nodes to and from a Riak cluster. We'll look at how you join nodes together into a cluster, and what happens when you add or remove nodes.

Preconditions

For most operations you need to access configuration files, whose location depends on your mode of installation and the operating system. Check the Configuration Files page for details on where to find them.

Creating the First Node

After installing Riak on a system using either the binary packages or from source,

there's some initial configuration steps you need to take that depend on your networking infrastructure and security measures.

Your node should not be running. If it is, stop it using the _riak stop_ command or `/etc/init.d/riak stop`). Before you can start up the node again, a couple of changes need to made. If your new node was already running before making the configuration changes outlined below, it's best to delete your ring directory before starting it up again. Just delete the directory ring/ in your Riak data directory. In general, the steps outlined below should be taken before you bring up a new node.

Change the Node Name

The node name is an important setting for the Erlang VM, especially when you want to build a cluster of nodes, as the node name identifies both the Erlang application and the host name on the network. All nodes in the Riak cluster need these node names to communicate and coordinate with each other.

The node name is configured in `vm.args`. Change the following line, which defaults to 127.0.0.1 (a.k.a. localhost):

Erlang

```
-name riak@127.0.0.1
```

Change it to something that corresponds to either the IP address or a resolvable host name for this particular node, like so:

Erlang

```
-name riak@192.168.1.10
```

Change the HTTP and Protocol Buffers binding address

By default, Riak's HTTP and Protocol Buffers services are bound to the local interface, i.e. 127.0.0.1, and are therefore unable to serve requests from the outside network. The relevant setting is configured in _app.config_, under the riak_core section that reads:

Erlang

```
{http, [ {"127.0.0.1", 8098 } ]},
```

Either change it to use an IP address that corresponds to one of the server's network interfaces, or 0.0.0.0 to allow access from all interfaces and networks, e.g.:

Erlang

```
{http, [ {"0.0.0.0", 8098 } ]},
```

The same configuration should be changed for the Protocol Buffers interface if you intend on using it. Do the same as above for the line in the `riak_kv` section that reads:

Erlang

```
{pb_ip,   "127.0.0.1" },
```

Start the Node

Just like the initial configuration steps, this step has to be repeated for every node in your cluster. Before a node can join an existing cluster it needs to be started. Depending on your mode of installation, use either the init scripts installed by the Riak binary packages or simply the script `riak`:

Shell

```
/etc/init.d/riak start
```

or

Shell

```
bin/riak start
```

When the node starts, it will look for a cluster description, known as the "ring file", in its data directory. If a ring file does not exist, it will create a new ring file based on the initially configured `ring_creation_size`, claiming all partitions for itself. Once this process completes, the node will be ready to serve requests.

Add a Node to an Existing Cluster

When the node is running, it can be added to an existing cluster. (Note that this step isn't necessary for the first node but only the ones you want to add after.) Pick a random node in your existing cluster and use the `riak-admin cluster join` command to stage a join request from the new node. The example shown below uses the IP 192.168.2.2 as the so-called "seed node", the node that seeds the existing cluster data to the new node.

Shell

```
riak-admin cluster join riak@192.168.2.2
```

This should result in a message similar to the following:

```
Success: staged join request for 'riak@192.168.2.5' to 'riak@192
```

Repeat this process on each new node that will joined to form the cluster.

Joining Nodes to Form a Cluster

The process of joining a cluster involves several steps, including staging the proposed cluster nodes, reviewing the cluster plan, and committing the changes.

After staging each of the cluster nodes with `riak-admin cluster join` commands, the next step in forming a cluster is to review the proposed plan of changes. This can be done with the `riak-admin cluster plan` command, which is shown in the example below.

```
=============================== Staged Changes ================
Action         Nodes(s)
---------------------------------------------------------------
join           'riak@192.168.2.2'
join           'riak@192.168.2.2'
join           'riak@192.168.2.2'
join           'riak@192.168.2.2'
---------------------------------------------------------------


NOTE: Applying these changes will result in 1 cluster transition

###############################################################
                         After cluster transition 1/1
###############################################################

================================= Membership ==================
Status     Ring    Pending    Node
---------------------------------------------------------------
valid     100.0%     20.3%    'riak@192.168.2.2'
valid       0.0%     20.3%    'riak@192.168.2.3'
valid       0.0%     20.3%    'riak@192.168.2.4'
valid       0.0%     20.3%    'riak@192.168.2.5'
valid       0.0%     18.8%    'riak@192.168.2.6'
---------------------------------------------------------------
```

```
Valid:5 / Leaving:0 / Exiting:0 / Joining:0 / Down:0

Transfers resulting from cluster changes: 51
  12 transfers from 'riak@192.168.2.2' to 'riak@192.168.2.3'
  13 transfers from 'riak@192.168.2.2' to 'riak@192.168.2.4'
  13 transfers from 'riak@192.168.2.2' to 'riak@192.168.2.5'
  13 transfers from 'riak@192.168.2.2' to 'riak@192.168.2.6'
```

If the plan is to your liking, submit the changes by typing `riak-admin cluster commit`.

Note: The algorithm that distributes partitions across the cluster during membership changes is non-deterministic. As a result, there is no optimal ring. In the event a plan results in a slightly uneven distribution of partitions, the plan can be cleared. Clearing a cluster plan with `riak-admin cluster clear` and running `riak-admin cluster plan` again will produce a slightly different ring.

Removing a Node From a Cluster

A node can be removed from the cluster in two ways. One assumes that a node is decommissioned, for example because its added capacity is not needed anymore or because it's explicitly replaced with a new one. The second is relevant for failure scenarios, where a node has crashed and is irrecoverable, so it must be removed from the cluster from another node.

The command to remove a running node is `riak-admin cluster leave`. This command must be executed on the node that's supposed to be removed from the cluster.

Similarly to joining a node, after executing `riak-admin cluster leave`, the cluster plan must be reviewed with `riak-admin cluster plan`, and the changes committed with `riak-admin cluster commit`.

Learn more about the new cluster command introduced in Riak version 1.2.

The other command is `riak-admin cluster leave <node>`, where `<node>` is an Erlang node name as specified in the node's vm.args file, e.g.

Shell

```
riak-admin cluster leave riak@192.168.2.1
```

This command can be run from any other node in the cluster.

Under the hood, both commands basically do the same thing. Running `riak-admin cluster leave <node>` just selects the current node for you automatically.

As with `riak-admin cluster leave`, the plan to have a node leave the cluster must be first reviewed with `riak-admin cluster plan`, and committed with `riak-admin cluster commit` before any changes will actually take place.

How Cluster Membership Changes Work

Note: Since Riak 1.0, all partition ownership decisions in a cluster are made by a single node (the claimant.)

When a node joins or leaves the cluster, the cluster's claimant creates a new ring, attempting to distribute partitions evenly across the cluster. This ring is not immediately used, but serves as a template for the final state of the cluster's ring once all partition ownership transfers have completed.

Once created, the claimant uses this new ring to generate a list of pending changes to the cluster. These changes need to occur before the transition to the new ring can be completed. This list consists of partitions whose ownership needs to be transfered between nodes, as well as the state of the transfers (complete or awaiting.) This list is distributed to the cluster members via the gossip protocol.

Once the pending changes list is gossiped to the other members of the cluster, nodes will begin handing off partitions. As transfers of partitions between nodes complete, the pending changes list is updated. The updated pending changes list is distributed to members of the cluster as updates are made to it.

Throughout the handoff process, the claimant uses this updated list to make incremental changes to the ring. Each time an incremental change is made, the new ring is distributed to all cluster members to reflect the new owner of the recently transfered partition. Once all transfers are complete, the ring distributed by the claimant will be the one created when the join command was executed, and the ownership handoff process will be complete.

In the case of leaving a node, the leaving node will shutdown once all if it's partitions have been transferred successfully.

These May Also Interest You

- Downloads
- Log Messages FAQs
- Operating Riak FAQs
- Configuration Files
- Load Balancing and Proxy Configuration
- Configuring Secondary Indexes