Post Installation

Contents

1. Starting a Riak Node
2. Does it work?
3. Riaknostic
4. Now what?

After you've installed Riak, you might next wish to check the liveness of each node and ensure that requests are being properly served.

The following are common ways to verify that your Riak nodes are operating correctly. After you've determined that your nodes are functioning and you're ready to put Riak to work, be sure to check out the resources in the **Now What?** section below.

Starting a Riak Node

Note about source installations

To start a Riak node that was installed by compiling the source code, you can add the Riak binary directory from the installation directory you've chosen to your PATH.

For example, if you compiled Riak from source in the `/home/riak` directory, then you can add the binary directory (`/home/riak/rel/riak/bin`) to your PATH so that Riak commands can be used in the same manner as with a packaged installation.

To start a Riak node, use the `riak start` command:

Shell

```
riak start
```

A successful start will return no output. If there is a problem starting the node, an error message is printed to standard error.

To run Riak with an attached interactive Erlang console:

Shell

```
riak console
```

A Riak node is typically started in console mode as part of debugging or troubleshooting to gather more detailed information from the Riak startup sequence. Note that if you start a Riak node in this manner, it is running as a foreground process which will be exited when the console is closed.

You can close the console by issuing this command at the Erlang prompt:

Erlang

```
q().
```

Once your node has started, you can initially check that it is running with the `riak ping` command:

Shell

```
riak ping
pong
```

The command will respond with **pong** if the node is running, or**pang** if it is unreachable for any reason.

Open Files Limit

As you may have noticed, if you haven't adjusted your open files limit (`ulimit -n`), Riak will warn you at startup about the limit. You're advised to increase the operating system default open files limit when running Riak. You can read more about why in the Open Files

Limit documentation.

Does it work?

One convenient means to test the readiness of an individual Riak node and its ability to read and write data is with the `riak-admin test` command:

Shell

```
riak-admin test
```

Successful output from `riak-admin test` looks like this:

Text

```
Attempting to restart script through sudo -H -u riak
Successfully completed 1 read/write cycle to 'riak@127.0.0.1
```

You can also test whether Riak is working by using the `curl` command-line tool. Now that you have Riak running on a node, try this command to retrieve the `test` bucket and its properties:

Shell

```
curl -v http://127.0.0.1:8098/riak/test
```

Replace `127.0.0.1` in the example above with your Riak node's IP address or fully qualified domain name; you should get a response from the command that looks like this:

Text

```
* About to connect() to 127.0.0.1 port 8098 (#0)
*   Trying 127.0.0.1... connected
* Connected to 127.0.0.1 (127.0.0.1) port 8098 (#0)
> GET /riak/test HTTP/1.1
> User-Agent: curl/7.21.6 (x86_64-pc-linux-gnu)
> Host: 127.0.0.1:8098
> Accept: */*
```

```
>
< HTTP/1.1 200 OK
< Vary: Accept-Encoding
< Server: MochiWeb/1.1 WebMachine/1.9.0 (someone had painted
< Date: Wed, 26 Dec 2012 15:50:20 GMT
< Content-Type: application/json
< Content-Length: 422
<
* Connection #0 to host 127.0.0.1 left intact
* Closing connection #0
{"props":{"name":"test","allow_mult":false,"basic_quorum":fa
 "big_vclock":50,"chash_keyfun":{"mod":"riak_core_util",
 "fun":"chash_std_keyfun"},"dw":"quorum","last_write_wins":f
 "linkfun":{"mod":"riak_kv_wm_link_walker","fun":"mapreduce_
 "n_val":3,"notfound_ok":true,"old_vclock":86400,"postcommit
 "precommit":[],"pw":0,"r":"quorum","rw":"quorum","small_vcl
 "w":"quorum","young_vclock":20}}
```

The above output shows a successful response (HTTP 200 OK) and additional details from the verbose option. The response also contains the bucket properties for Riak's test bucket.

Riaknostic

It is a good idea to also verify some basic configuration and general health of the Riak node after installation by using Riak's built-in diagnostic utility *Riaknostic*.

Ensure that Riak is running on the node, and issue the following command:

```
riak-admin diag
```

Make the recommended changes from the command output to ensure optimal node operation.

Now what?

You have a working Riak node!

From here you might want to check out the following resources.

- Check out the Client Libraries to use Riak with your favorite programming language
- Learn about the high level concepts of Riak

Tutorial Nav: Installing and Upgrading

Installing Riak From Source

These May Also Interest You

- Installing on AWS Marketplace
- Installing on Windows Azure
- Installing on Debian and Ubuntu
- Installing Erlang
- Installing on FreeBSD
- Installing Riak from Source