NoSQL Database in the Cloud: Riak on AWS

June 2013

Brian Holcomb

(Please consult **http://aws.amazon.com/whitepapers/** for the latest version of this paper)

Table of Contents

Abstract

Amazon Web Services (AWS) is a flexible, cost-effective, easy-to-use cloud computing platform. Running your own NoSQL data store on Amazon EC2 may be ideal if your application or service requires the unique properties offered by NoSQL databases. NoSQL systems are some of the most widely deployed software packages within the Amazon cloud.

This white paper will help you understand one of the more popular NoSQL options available with the AWS cloud computing platform—the open source database Riak. Riak is developed by Basho, a distributed systems company. You'll find an overview of general best practices and details of important Riak implementation characteristics like performance, durability, and security. You'll also learn some key specifics about the scalability, high availability, and fault tolerance of Riak databases.

Note: In this guide, items that begin with `$` are run at a standard shell prompt, which may require syntax adjustment on other systems.

Overview

Riak is an open source, distributed NoSQL database. Riak is architected for multiple advantages:

- **Availability**–Riak replicates and retrieves data intelligently so it is available for read and write operations, even in failure conditions.
- **Fault tolerance**–You can lose access to many nodes due to network partition or hardware failure without losing data.
- **Operational simplicity**–You can add new machines to your Riak cluster easily without incurring a larger operational burden; the same ops tasks apply to small clusters as large clusters.
- **Scalability**–Riak automatically distributes data around the cluster and yields a near-linear performance increase as you add capacity.

Riak uses a simple key-value model for object storage. Objects in Riak consist of a unique key and a value, stored in a flat namespace called a bucket. You can store virtually any type of content you want in Riak: text, images, JSON, XML, and HTML documents; user and session data; backups; log files; and more.

Riak provides a straightforward, RESTful API as well as a protocol buffers interface. There are many client libraries for Riak, including Java, Python, Perl, Erlang, Ruby, PHP, .NET, and more. For more information, see http://docs.basho.com/riak/latest/references/Client-Libraries/.

Basic Installation

With AWS, you can easily to create and launch one or more Amazon EC2 Instances running Riak.

Launching Riak VMs via the AWS Marketplace

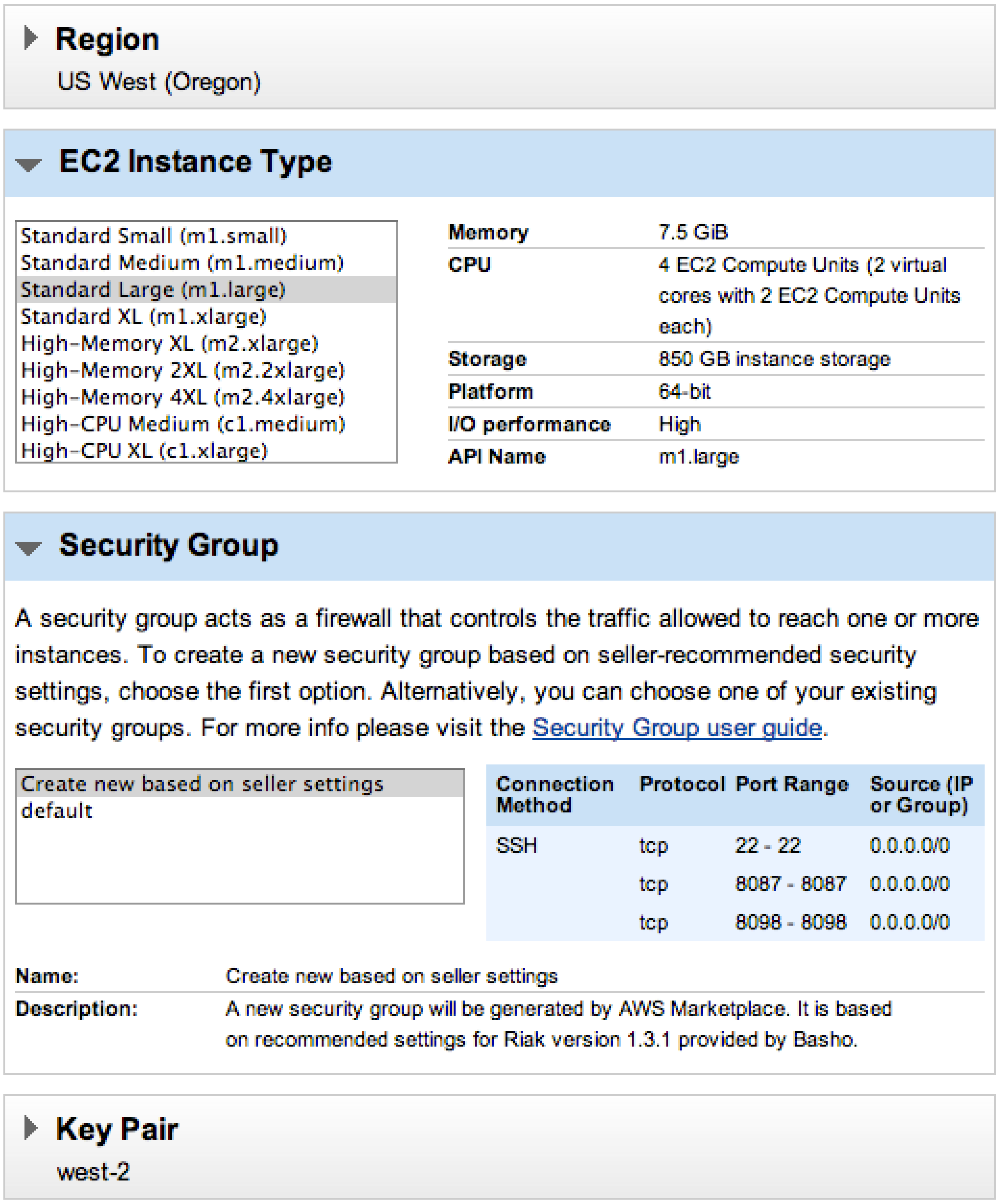
In order to launch a Riak virtual machine via the AWS Marketplace, you will first need to sign up for an AWS account at http://aws.amazon.com/ (if you do not already have one).

1. Navigate to https://aws.amazon.com/marketplace/ and sign in with your Amazon Web Services account.

2. Locate Riak in the **Databases & Caching** category or search for Riak from any page. Click **Riak** to open its page.

3. On the Riak page, click **Continue**.

4. Set your desired AWS region, Amazon EC2 instance type, security group settings, and key pair. It is recommended that you use the latest version of Riak available.



Region
US West (Oregon)

EC2 Instance Type

Standard Small (m1.small)
Standard Medium (m1.medium)
Standard Large (m1.large)
Standard XL (m1.xlarge)
High-Memory XL (m2.xlarge)
High-Memory 2XL (m2.2xlarge)
High-Memory 4XL (m2.4xlarge)
High-CPU Medium (c1.medium)
High-CPU XL (c1.xlarge)

Memory	7.5 GiB
CPU	4 EC2 Compute Units (2 virtual cores with 2 EC2 Compute Units each)
Storage	850 GB instance storage
Platform	64-bit
I/O performance	High
API Name	m1.large

Security Group

A security group acts as a firewall that controls the traffic allowed to reach one or more instances. To create a new security group based on seller-recommended security settings, choose the first option. Alternatively, you can choose one of your existing security groups. For more info please visit the Security Group user guide.

Create new based on seller settings
default

Connection Method	Protocol	Port Range	Source (IP or Group)
SSH	tcp	22 - 22	0.0.0.0/0
	tcp	8087 - 8087	0.0.0.0/0
	tcp	8098 - 8098	0.0.0.0/0

Name:	Create new based on seller settings
Description:	A new security group will be generated by AWS Marketplace. It is based on recommended settings for Riak version 1.3.1 provided by Basho.

Key Pair
west-2

5. Click **Launch with 1-Click**.

Security Group Settings

Once the virtual machine is created, you should verify your selected Amazon EC2 security group is configured properly for Riak. More information about the logic and requirements for this configuration is detailed in the Security section below.

1. In the Amazon EC2 Management Console (https://console.aws.amazon.com/ec2/), click **Security Groups** on the left. Then click the name of the security group for your Riak VM (by default, **Riak-1-3-1-AutogenByAWSMP-**).

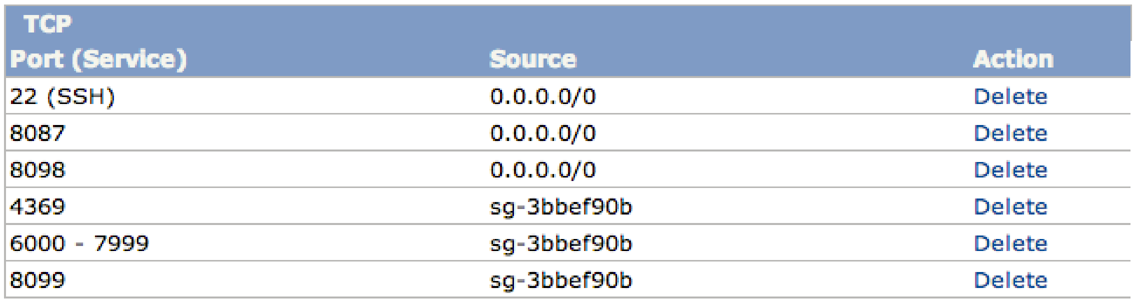
2. Click the **Inbound** tab in the lower pane. Your security group should include the following open ports:

    - 22 (SSH)
    - 8087 (Riak Protocol Buffers Interface)
    - 8098 (Riak HTTP Interface)

3. You will need to add additional rules within this security group to allow your Riak instances to communicate. For each port range below, create a new **Custom TCP rule** with the source set to the current security **Group ID** (found on the **Details** tab).

    - Port range: 4369
    - Port range: 6000–7999
    - Port range: 8099

When complete, your security group should contain all of the rules listed below. (Note that **sg-3bbef90b** is an example; your security group will have a different identifier.) If you are missing any rules, add them in the lower panel and then click **Apply Rule Changes**.



TCP Port (Service)	Source	Action
22 (SSH)	0.0.0.0/0	Delete
8087	0.0.0.0/0	Delete
8098	0.0.0.0/0	Delete
4369	sg-3bbef90b	Delete
6000 - 7999	sg-3bbef90b	Delete
8099	sg-3bbef90b	Delete

For SSH, it's usually best to restrict access to only hosts that will need shell access. For ports 8087 and 8098, restrict access to only those hosts needing to access the Riak cluster using the EC2 section of the AWS console.

Clustering Riak on AWS

You will need to launch at least three instances to form a Riak cluster. When the instances have been provisioned and the security group is configured, you can connect to them using SSH or PuTTY as the ec2-user. For resiliency launch these instances in different Availability Zones.

For more information about connecting to an instance, see the Amazon EC2 instance guide at http://docs.amazonwebservices.com/AWSEC2/latest/UserGuide/AccessingInstances.html.

Once you have connected to your instances, you can do the following to create a cluster:

1. On the first node obtain the internal IP address:

```
$ curl http://169.254.169.254/latest/meta-data/local-ipv4
```

2. For all other nodes, use the internal IP address of the first node:

```
$ sudo riak-admin cluster join riak@<ip.of.first.node>
```

3. After all of the nodes are joined, execute the following:

```
$ sudo riak-admin cluster plan
```

4. If this looks good, then run this command:

```
$ sudo riak-admin cluster commit
```

5. To check the status of clustering, use the following:

```
$ sudo riak-admin member_status
```

Congratulations! You have a highly available three-node Riak cluster running on AWS!

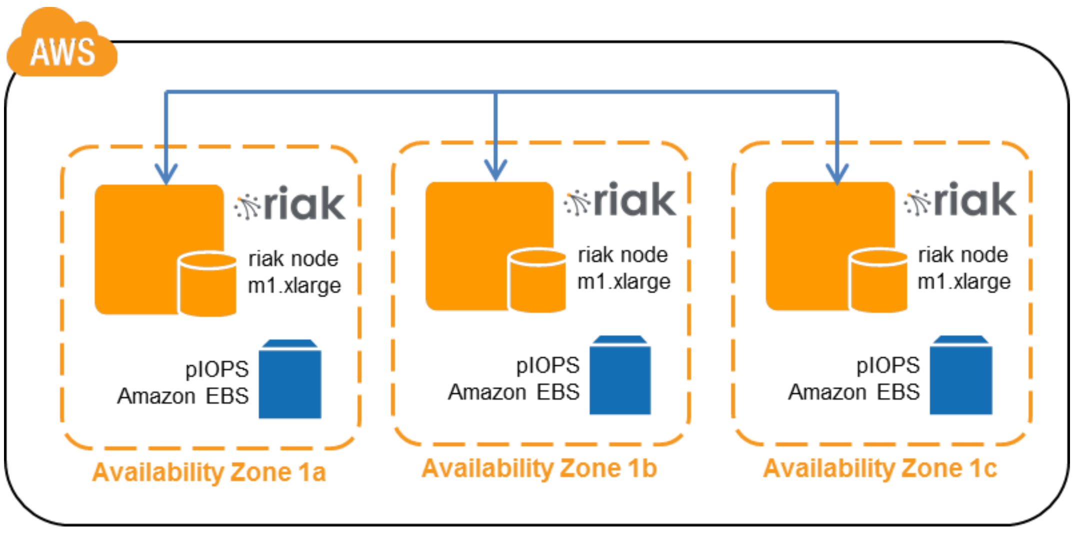


Figure 1: Three-node Riak Cluster on AWS

Architecture and Scale

The design of your Riak installation on EC2 is largely dependent on the scale at which you're trying to operate. For example, are you experimenting with the framework on your own for a private project? If so, it's likely that you will configure a minimal number of Riak nodes leveraging relatively small instance hardware.

Architecture

To understand the system requirements, it is important to consider the core architecture of Riak.

What Is a Riak Node?

Each node in a Riak cluster is the same, containing a complete, independent copy of the Riak package. There is no "master." This uniformity provides the basis for Riak's fault tolerance and scalability. Riak is written in Erlang, a language designed for massively scalable systems.

Data Distribution

Data is distributed across nodes using consistent hashing. Consistent hashing allows data to be evenly distributed around the cluster and new nodes can be added automatically with minimal reshuffling.

Replication

Riak automatically replicates data in the cluster (default three replicas per object). You can lose access to many nodes in the cluster due to failure conditions and still maintain read and write availability.

When Nodes Fail

If a node fails or is partitioned from the rest of the cluster, a neighboring node will take over its storage operations. When the failed node returns, the updates received by the neighboring node are handed back to it. This allows availability for writes or updates and happens automatically.

Scaling

Basho recommends deployments of five nodes or greater (Figure 2 below) to provide a foundation for high performance and growth as the cluster expands. As Riak scales linearly with the addition of more nodes, users find improved performance, reliability, and throughput with larger clusters. For more information read the blog post at http://basho.com/why-your-riak-cluster-should-have-at-least-five-nodes/.

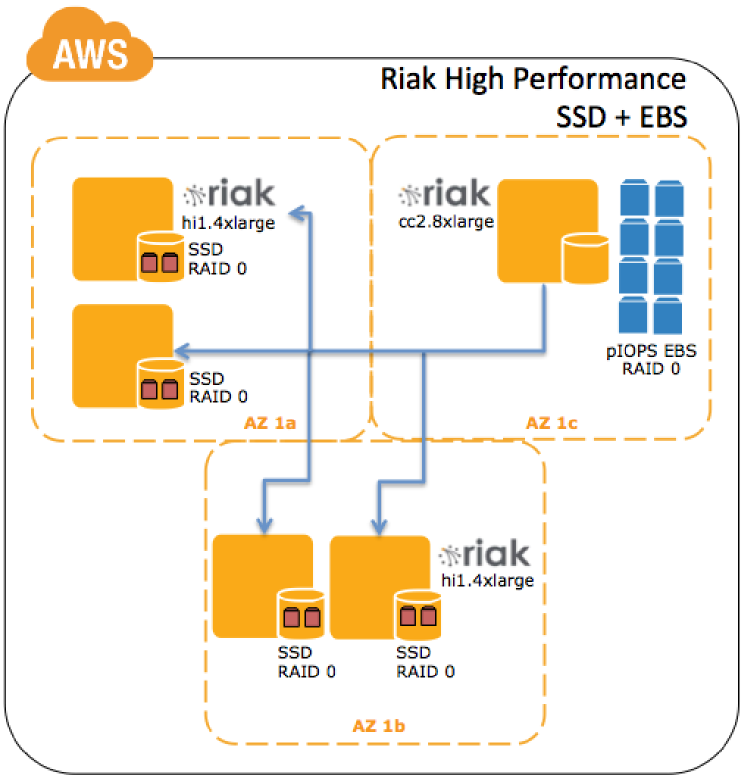


Figure 2: Five-node Riak Architecture on AWS

It is tempting to think of scale as a vertical activity. However, vertical scaling doesn't offer all of the same benefits of horizontal scaling. Higher performance instances can provide many of the performance benefits of a more complex replicated topology, but remember that they come with none of the significant fault-tolerance benefits. In concert, running Riak on AWS can provide for simple horizontal scale based upon expected and experienced load. When you add nodes to a Riak cluster, the data is rebalanced automatically with no downtime. As any node can accept or route requests, there is no need to deal with the underlying complexity of data location as is necessary when sharding.

Scaling step by step when you know you need a big system isn't efficient. It is important to test your clustered instances with expected data patterns in advance of production launch.

Operational Considerations

Running Riak on AWS provides operational simplicity at scale, but there are some considerations you should take when planning a deployment.

EC2 Instance Sizing

Amazon EC2 instances are available as predefined types that encapsulate a fixed amount of computing resources, the most important of which to Riak are Disk I/O, RAM, and Network I/O, followed by CPU cores. With this in mind, Riak users have reported success with Large, Extra Large, and Cluster Compute Instance types for use as cluster nodes in the EC2 environment.

The most commonly used instance types for Riak cluster nodes are the **m1.large** or **m1.xlarge**. In cases where 10-gigabit Ethernet networking is desired, the Cluster Compute class of EC2 instances, such as **cc1.4xlarge** or **cc2.8xlarge** can be used. Amazon also offers a High I/O Quadruple Extra Large instance (**hi1.4xlarge**) that is backed by solid state drives (SSD) and features very high I/O performance with very low latency. For more information about instance types, see http://aws.amazon.com/ec2/instance-types/.

Storage Configuration

Riak's primary bottleneck will be disk and network I/O. Riak has multiple storage back ends, and each has different benefits and performance characteristics. The default is Bitcask which features sequential writes and random reads, but LevelDB could be used if secondary indexes are required. See http://docs.basho.com/riak/latest/tutorials/choosing-a-backend/ for choosing the back end that is right for your application.

EBS-Optimized instances deliver dedicated throughput between Amazon EC2 and Amazon EBS, with options between 500 megabits per second and 1,000 megabits per second depending on the instance type used. EBS Provisioned IOPS volumes can deliver up to 4000 IOPS per volume with a 16K IO size. For more information, see http://aws.amazon.com/about-aws/whats-new/2012/07/31/announcing-provisioned-iops-for-amazon-ebs/.

Ephemeral storage can be lost when a node is terminated so it is recommended to use EBS in a RAID 10 configuration to achieve durability and performance.

In any case, you will need proper benchmarking and tuning to achieve the desired performance.

Finally, try these helpful storage tips:

- Use XFS or ext4 for the data volume.
- Turn off `atime` and `diratime` when you mount the data volume. Doing so reduces I/O overhead by disabling features that aren't useful to Riak.
- If using ext4, set `barrier=0,data=writeback` when you mount the data volume. These features are not necessary at the file system level due to the storage design of Riak.
- If using XFS, set `nobarrier,logbufs=8,logbsize=128k,allocsize=2M` when you mount the data volume.
- Use the deadline scheduler for EBS volumes. To set the scheduler in use for block device `xvdf`, for example, use the following command:

  ```
  $ echo deadline > /sys/block/xvdf/queue/scheduler
  ```

Network Configuration

Amazon EC2 instances that are not provisioned inside a VPC change the following parameters after a restart.

- Private IP address
- Public IP address
- Private DNS
- Public DNS

Since Riak binds to an IP addresses and communicates with other nodes based on this address, executing certain admin commands are necessary to bring the node back up. The following steps must be performed.

1. Stop the node to rename.

   ```
   $ sudo riak stop
   ```

2. Mark it `down` from another node in the cluster.

   ```
   $ sudo riak-admin down 'old nodename'
   ```

3. Rename the node in vm.args.
4. Delete the ring directory.
5. Start the node.

   ```
   $ sudo riak start
   ```

   It will come up as a single instance, which you should verify.

   ```
   $ sudo riak-admin member-status
   ```

6. Join the node to the cluster.

   ```
   $ sudo riak-admin cluster join 'cluster nodename'
   ```

7. Set it to replace the old instance of itself.

   ```
   $ sudo riak-admin cluster replace
   ```

8. Plan the changes.

   ```
   $ sudo riak-admin cluster plan
   ```

9. Commit the changes.

   ```
   $ sudo riak-admin cluster commit
   ```

To avoid this inconvenience, you can deploy Riak to a VPC. Instances inside the VPC do not change their private IP address on restart. In addition you get the following benefits:

- Access control lists can be defined at various levels (load balancers, individual servers, VPC groups).
- Should the private nodes need to contact the Internet, they can do so through a NAT instance.
- VPC is free.

To achieve better network performance when using a cluster compute instance with 10-gigabit Ethernet, change the following `sysctl` settings:

```
net.core.rmem_default = 8388608
net.core.rmem_max = 8388608
net.core.wmem_default = 8388608
net.core.wmem_max = 8388608
net.core.netdev_max_backlog = 10000
```

For maximum resiliency, it is recommended that you have nodes in your Riak cluster in multiple Availability Zones.

Benchmarking

Using a tool such as Basho Bench (https://github.com/basho/basho_bench), you can generate loads that simulate application operations by constructing and communicating approximately compatible data payloads with the Riak cluster directly. Benchmarking is critical to determining the appropriate EC2 instance types and is strongly recommended. More information is available on benchmarking Riak clusters with Basho Bench in the Basho Bench documentation at http://docs.basho.com/riak/latest/cookbooks/Benchmarking/.

In addition to running Basho Bench, it is also advisable to load test Riak with your own tests to ensure that load imparted by M/R queries, linking, link-walking, full-text queries, and index queries are within the expected range.

Simulating Upgrades, Scaling, and Failure States

In addition to simply measuring performance, it is also important to measure how performance degrades when the cluster is not in steady state. While simulating a live load, you can model the following states:

1. Stop one or more nodes normally and restart them after a few moments to simulate a rolling upgrade.
2. Join two or more nodes to the cluster.
3. Leave nodes from the cluster (after the previous step).
4. Hard-kill the Riak `beam.smp` process (i.e., `kill -9`) and then restart it.
5. Hard-reboot a node's instance using the AWS console and then restart it.
6. Hard-stop and destroy a node's instance and build a new one from backup.
7. Via networking (for example, by revoking access in a security group), partition one or more nodes from the rest of the cluster and then restore the original configuration.

Monitoring

AWS provides robust monitoring of EC2 instances, EBS volumes, and other services via the Amazon CloudWatch service (http://aws.amazon.com/cloudwatch/). CloudWatch can alert you, via SMS or email, when individual AWS services reach user-defined thresholds. A great example would be an alarm on excessive storage throughput. For more information, see http://docs.aws.amazon.com/AmazonCloudWatch/latest/DeveloperGuide/US_AlarmAtThresholdEBS.html.

Another approach would be to write a custom metric to CloudWatch, such as current free memory on your instances, and to trigger automatic responses off of those measures. For more information, see http://docs.amazonwebservices.com/AmazonCloudWatch/latest/DeveloperGuide/publishingMetrics.html.

Riak exposes numerous forms of vital statistic information that can be aggregated, monitored, analyzed, graphed, and reported on in a variety of ways using numerous open-source and commercial solutions. A lengthy discussion of statistics, counters, metrics, and more can be found on the Basho documentation portal at http://docs.basho.com/riak/latest/cookbooks/Statistics-and-Monitoring/.

Security

The security group settings required for Riak to function are enumerated in the installation instructions above. These settings enable intracluster communication. The following discussion expands on their configuration.

General Settings and Configuration

Riak has two classes of access control: other Riak nodes participating in the cluster and clients making use of the Riak cluster.

The settings for both access groups are located in `app.config`. The configuration directives for client access all end in `ip` and `port`: `web_ip`, `web_port`, `pb_ip`, and `pb_port`.Make note of those and configure your firewall to allow incoming TCP access to those ports or IP address and port combinations. Exceptions to this are the `handoff_ip` and `handoff_port` directives. Those are for communication between Riak nodes only.

Riak uses the Erlang distribution mechanism for most internode communication. Riak identifies other machines in the ring using Erlang identifiers with the form *<hostname or IP>*—for example, `riak@10.9.8.7`. Erlang resolves these node identifiers to a TCP port on a given machine via the Erlang Port Mapper daemon (`epmd`) running on each cluster node.

By default, `epmd` binds to TCP port 4369 and listens on the wildcard interface. For internode communication, Erlang uses an unpredictable port by default; it binds to port 0, which means the first available port.

Default Communication Ports

Riak nodes in a cluster need to be able to communicate freely with one another on the following ports:

- epmd listener: TCP:4369
- handoff_port listener: TCP:8099
- range of ports specified in `app.config`

Riak clients must be able to contact at least one machine in a Riak cluster on the following ports:

- web_port: TCP:8098
- pb_port: TCP:8087

Replication

Riak Enterprise is Riak with multidatacenter replication, monitoring, and 24×7 support. You can leverage this commercial extension to Riak to replicate data across Availability Zones. For more information on running Riak Enterprise on AWS, please contact Basho at http://basho.com/contact/.

Customers can use multidatacenter replication to serve global traffic, maintain active backups, run secondary analytics clusters, or meet disaster recovery and regulatory requirements. Multidatacenter replication can be used in two or more sites.

Riak Enterprise features two options for multidatacenter replication: full sync and real-time sync. With full sync, replication of data occurs at scheduled intervals between two clusters. The default interval is six hours. When full sync is initiated, clusters generate and compare hashes for all of their objects. During the comparison process, the primary cluster detects missing or out-of-date objects in the secondary cluster(s). It then streams any new objects or updates so the clusters have the same data.

With real-time sync, replication to the secondary data center(s) is triggered by updates to the primary data center. After writing an object to the primary cluster, writes are sent to the secondary cluster(s) via postcommit hook. All multidatacenter replication occurs over TCP connection and SSL is supported. By default, the connection is unidirectional; however, bidirectional replication can be achieved by establishing two unidirectional connections between clusters.

Conclusion

The AWS cloud provides a unique platform for any NoSQL application, including Riak. With capacities that can meet dynamic needs, cost that is based on use, and easy integration with other AWS products like Amazon CloudWatch, the AWS cloud enables you to run a variety of NoSQL applications without having to manage the hardware yourself.

In addition, management features inherent in NoSQL systems can be leveraged in the cloud. The Riak hardware configurations, architecture designs, HA and DR approaches, and security details provided here, can help you in setting up a secure, high capacity, fault tolerant and scalable Riak system.

Further Reading

1. For more information about Riak, see Basho's documentation at http://docs.basho.com.
2. For deployment options and AWS CloudFormation templates, see http://basho.com/riak-on-aws-deployment-options/.