CMPE 281 - LAB #6 - AWS NoSQL Riak Cluster

In this Lab, you will be deploying a Riak Cluster and then creating an API Gateway in front of the Load Balanced Cluster.

- https://github.com/paulnguyen/cmpe281/tree/master/labs/lab6

Part 1 - Setup Riak Cluster

- http://basho.com/posts/technical/riak-on-aws-deployment-options/
- http://docs.basho.com/riak/kv/2.2.3/developing/usage/
- http://docs.basho.com/riak/kv/2.2.3/setup/installing/amazon-web-services/
- http://docs.basho.com/riak/kv/2.2.3/using/running-a-cluster/#configure-the-first-node
- http://docs.basho.com/riak/kv/2.2.3/using/cluster-operations/adding-removing-nodes/
- http://docs.basho.com/riak/kv/2.2.3/developing/usage/conflict-resolution/
- https://aws.amazon.com/marketplace/pp/B00YFZ60X2?ref=cns_srchrow
- http://docs.basho.com/riak/kv/2.2.3/setup/installing/amazon-web-services/

Launch Riak Marketplace AMI (3 Nodes)

```
1. AMI:             Riak KV 2.2 Series
2. Instance Type:   t2.micro
3. VPC:             cmpe281
4. Network:         private subnet
5. Auto Public IP:  no
6. Security Group:  riak-cluster
7. SG Open Ports:   (see below)
8. Key Pair:        your key pair (i.e. cmpe281-us-west-2 or cmpe281-us-
east-1)

Riak Cluster Security Group (Open Ports):

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

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
```

http://docs.basho.com/riak/kv/2.2.3/setup/installing/amazon-web-services/
http://docs.basho.com/dataplatform/1.0.0/configuring/default-ports/

NOTE: Port Ranges above, as documented, will not work for Creating a Cluster.
There seems to be missing port(s) that needs to be open. As such, just add the following Rule instead of the 4369, 6000-7999 and 8099 rules.

Port range: 0-65535 (Source: Security Group ID)

Launch "Jump Box" AWS Linux AMI

1. AMI: Amazon Linux AMI 2018.03.0 (HVM)
2. Instance Type: t2.micro
3. VPC: cmpe281
4. Network: public subnet
5. Auto Public IP: yes
6. Security Group: cmpe281-dmz
7. SG Open Ports: 22, 80, 443
8. Key Pair: your key pair (i.e. cmpe281-us-west-2 or cmpe281-us-east-1)

SSH into Riak Instance (via Jump Box)

ssh -i <key>.pem ec2-user@<public ip> (access jump box)
ssh -i <key>.pem ec2-user@<private ip> (access riak node)

10.0.1.147 riak-node1
10.0.1.28 riak-node2
10.0.1.42 riak-node3

Setup Riak Cluster Nodes (3 Nodes)

- http://docs.basho.com/riak/kv/2.2.3/using/running-a-cluster/#configure-the-first-node

  You will need need to launch at least 3 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 all other nodes, use the internal IP address of the first node:

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

- After all of the nodes are joined, execute the following:

  sudo riak-admin cluster plan
  sudo riak-admin cluster status

If this looks good:

sudo riak-admin cluster commit

To check the status of clustering use:

sudo riak-admin member_status

You now have a Riak cluster running on AWS.

Riak Node 1

In /etc/riak, you should see the following settings in riak.conf after deploying the AMI.

Node 1 Config (riak.conf):

```
listener.http.internal = 10.0.1.147:8098
listener.protobuf.internal = 10.0.1.147:8087
nodename = riak@10.0.1.147
```

Start Node 1:

```
sudo riak start
sudo riak ping
sudo riak-admin status
```

Riak Node 2

Node 2 Config (riak.conf):

```
listener.http.internal = 10.0.1.28:8098
listener.protobuf.internal = 10.0.1.28:8087
riak.conf:nodename = riak@10.0.1.28
```

Start Node 2 / Join Cluster:

```
sudo riak start
sudo riak ping
sudo riak-admin status

sudo riak-admin cluster join riak@10.0.1.147  (node 1)
sudo riak-admin cluster plan
sudo riak-admin cluster status
```

Riak Node 3

Node 3 Config (riak.conf):

```
listener.http.internal = 10.0.1.42:8098
listener.protobuf.internal = 10.0.1.42:8087
riak.conf:nodename = riak@10.0.1.42
```

Start Node 3 / Join Cluster:

```
sudo riak start
sudo riak ping
sudo riak-admin status

sudo riak-admin cluster join riak@10.0.1.147  (node 1)
sudo riak-admin cluster plan
sudo riak-admin cluster status
```

Commit Plan for Riak Cluster

```
sudo riak-admin cluster plan
sudo riak-admin cluster status
```

If this looks good:

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

To check the status of clustering use:

```
    sudo riak-admin member_status
```

Sample Riak Usage Example

```
curl -i http://riak-node1:8098/ping

curl -i http://riak-node1:8098/buckets?buckets=true

curl -v -XPUT -d '{"foo":"bar"}' \
    http://riak-node2:8098/buckets/bucket/keys/key1?returnbody=true

curl -i http://riak-node3:8098/buckets/bucket/keys/key1
```

Setup Internal Load Balancers in front of Riak HTTP API (Port 8098)

Classic Load Balancer Settings

1. Name: aws-riak-elb-app
2. Internal: true
3. VPC: CMPE281
4. Ports: ELB Port: 80
   Instance Port: 8098
6. Subnet: Select Private Subnet
7. Security Group: riak-elb (create new)
   Open Ports: 80
8. Health Check: /ping
9. EC2 Instances: Select the three Riak Nodes

Network Load Balancer Settings

1. Name: aws-riak-elb-net
2. Schema: internal
3. Port: 80
4. VPC: CMPE281
5. Subnet: Select Private Subnet
6. Routing:
   - Target Group: RiakNodes
   - Target Port: 8098
   - Target Type: instance
   - Protocol: TCP
   - Health Check: TCP (Port: 8098)
7. Instances: Select the three Riak Nodes

Test from Jump Box

```
## Test Classic Load Balancer (Sample DNS Name -- Replace with your own)

curl -i http://internal-aws-riak-elb-app-1260081257.us-
west-2.elb.amazonaws.com/ping
curl -i http://internal-aws-riak-elb-app-1260081257.us-
west-2.elb.amazonaws.com/buckets/bucket/keys/key1

## Test Network Load Balancer (Sample DNS Name -- Replace with your own)

curl -i http://aws-riak-elb-net-6f2f46db7180948d.elb.us-west-2.amazonaws.com/
ping
curl -i http://aws-riak-elb-net-6f2f46db7180948d.elb.us-west-2.amazonaws.com/
buckets/bucket/keys/key1
```

Part 2 - Setup API Gateway In Front of Network Load Balancer

- https://docs.aws.amazon.com/apigateway/latest/developerguide/set-up-private-integration.html
- https://docs.aws.amazon.com/apigateway/latest/developerguide/set-up-private-integration.html
- https://docs.aws.amazon.com/apigateway/latest/developerguide/getting-started-with-private-integration.html
- https://aws.amazon.com/premiumsupport/knowledge-center/iam-authentication-api-gateway/
- https://docs.aws.amazon.com/apigateway/latest/developerguide/how-to-use-postman-to-call-api.html
- https://docs.aws.amazon.com/apigateway/latest/developerguide/api-gateway-control-access-using-iam-policies-to-invoke-api.html
- https://docs.aws.amazon.com/apigateway/latest/developerguide/api-gateway-iam-policy-examples-for-api-execution.html
- https://docs.aws.amazon.com/apigateway/latest/developerguide/api-gateway-create-api-from-example-console.html

Create Initial Empty API

1. Name: RiakAPI
2. End Point Type: Regional

Create API Gateway VPC Link

1. Name: RiakLink
2. Target NLB: aws-riak-elb-net

Update API Settings

1. Create a Proxy Resource:
   Name: proxy
   Path: /{proxy}+

2. Create an ANY Method:
   Integration: VPC Link
   Authorization: AWS_IAM
   Use Proxy Integration: True
   VPC Link: Select RiakLink
   Endpoint URL: http://endpoint/{proxy}
   Use Network Load Balancer End Point

For Example:
http://aws-riak-elb-net-6f2f46db7180948d.elb.us-west-2.amazonaws.com/{proxy}

3. Set API Resource Policy

Note the API ARN from "ANY METHOD". For Example:
ARN: arn:aws:execute-api:us-west-2:633868400030:5kce6w7kwj/*/*/*

Format is as follows:

```
"arn:aws:execute-api:{region}:{account-id}:{api-id}/{stage}/{method}/{path}"
```

Edit the API Resource Policy as follows:

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": "*",
            "Action": "execute-api:Invoke",
            "Resource": "arn:aws:execute-api:us-west-2:633868400030:5kce6w7kwj/*/*/*"
        }
    ]
}
```

This Policy allows any IAM Authenticated User from your Account to Invoke the Gateway API.

4. Deploy API to "prod" Stage

Note your Stage Invoke Endpoint Generated. For Example:

https://bnn2cma4c4.execute-api.us-west-2.amazonaws.com/prod

5. Create IAM User and Note Access and Security Keys for API Authentication

Use Postman to Test API: https://docs.aws.amazon.com/apigateway/latest/developerguide/how-to-use-postman-to-call-api.html

Test API Gateway From Postman (I.E. -- AWS Auth Header Excluded)

```
curl -X GET \
    https://5kce6w7kwj.execute-api.us-west-2.amazonaws.com/prod/ping

curl -X GET \
    https://5kce6w7kwj.execute-api.us-west-2.amazonaws.com/prod/buckets?buckets=true

curl -X PUT \
    https://5kce6w7kwj.execute-api.us-west-2.amazonaws.com/prod/buckets/bucket/keys/
key1?returnbody=true \
  -d '{
    "foo": "bar"
    }'

curl -X GET \
  https://5kce6w7kwj.execute-api.us-west-2.amazonaws.com/prod/buckets/bucket/keys/key1
```

Part 3 (Optional) - Partition Tolerance Testing:

In this part, you will be testing the **Partition Tolerance** using the procedures described in the following article: https://www.infoq.com/articles/jepsen. Note: you don't have to follow the directions in the article "verbatim". Adjust the steps as you see necessary -- for example, in how you create a partition in AWS.

Notes:

- Use the steps in the article as guidance only.
- Feel free to diverge from those steps as needed to accomplish your testing goals.
- Using alternative testing programs, programming languages, tools and/or approaches to creating network partitions are allowed.
- Please document any steps you take that diverge from the steps in the article.
- ***Partition Testing Experiment:***
    - Set up your cluster as AWS EC2 Instances. (# of Nodes and Topology is open per your design)
        - Make sure to note your approach to creating a "network partition" for experiments.
    - Set up the Experiments (i.e. Test Cases) to answer the following questions:
        - AP (Riak DB Cluster):
            - How does the system function during normal mode (i.e. no partition)
            - What happens to the nodes during a partition?
            - Can stale data be read from a node during a partition?
            - What happens to the system during partition recovery?
    - Run the Experiments and Record results.