Redis Cluster Setup Using Helm

1. Login to the Bastion Host

Use the public address of the bastion host obtained in the previous Terraform setup.

The PEM file is located in the files directory of your Terraform setup.
Example command:

```
ssh -i files/bastion_key ubuntu@ec2-xx-xxx-xx-xx.compute-1.amazonaws.com
```

2. Set the Kubernetes Context

Once logged in to the bastion, set the Kubernetes context to your EKS cluster:

```
aws eks update-kubeconfig --region us-east-1 --name concentric-ai-cluster
```

3. Install Helm

Install Helm on the bastion host using the following commands:

```
curl -fsSL -o get_helm.sh https://raw.githubusercontent.com/helm/helm/main/scripts/get-helm-3
chmod 700 get_helm.sh
./get_helm.sh
```

4. Prepare Redis Helm Chart

Pull your Redis zip file to the bastion host, unzip it, and navigate into the Redis directory:

```
unzip redis-helm-chart.zip
cd redis
```

5. Install Redis Helm Chart

Install the Redis Helm chart using the following command:

```
helm install redis .
```

The output will confirm the successful deployment of the Redis cluster:

```
Release "redis" does not exist. Installing it now.
NAME: redis
LAST DEPLOYED: Fri Aug 9 19:48:29 2024
NAMESPACE: default
STATUS: deployed
REVISION: 1
```

6. Verify Redis Pods

Check if the Redis master and replica pods are up and running:

```
kubectl get po
```

The expected output should show 3 master pods and 6 replica pods:

```
NAME             READY   STATUS    RESTARTS   AGE
redis-master-0    1/1     Running   0         112s
redis-master-1    1/1     Running   0         84s
redis-master-2    1/1     Running   0         82s
redis-replica-0   1/1     Running   0         112s
redis-replica-1   1/1     Running   0         82s
redis-replica-2   1/1     Running   0         81s
redis-replica-3   1/1     Running   0         68s
redis-replica-4   1/1     Running   0         66s
redis-replica-5   1/1     Running   0         65s
```

7. Setup Redis Cluster

Trigger the Redis cluster setup by running the setup_redis.sh script:

```
bash setup_redis.sh
```

The output will guide you through the setup process, and you should see confirmation that the Redis cluster is operational and data replication is successful:

```
ubuntu@ip-10-1-7-62:~/redis$ bash setup_redis.sh >>>
Performing hash slots allocation on 9 nodes...
Master[0] -> Slots 0 - 4095
Master[1] -> Slots 4096 - 8191
Master[2] -> Slots 8192 - 12287
Master[3] -> Slots 12288 - 16383
Adding replica 10.1.4.206:6379 to 10.1.4.84:6379
Adding replica 10.1.5.151:6379 to 10.1.5.15:6379
Adding replica 10.1.5.61:6379 to 10.1.5.239:6379
Adding replica 10.1.4.60:6379 to 10.1.4.178:6379
Adding extra replicas...
Adding replica 10.1.5.254:6379 to 10.1.4.84:6379
M: 13c6cdda755318701a517eb75897e6012126fda0 10.1.4.84:6379 slots:[0-4095] (4096 slots) master
M: 3001e467965c9cecab45115ec4df6a8a211d2296 10.1.5.15:6379 slots:[4096-8191] (4096 slots) master
M: ea68ed55f90ba1681098583636df291ed16314c9 10.1.5.239:6379 slots:[8192-12287] (4096 slots)
master
M: ccc1291a5dde817831116505e85fa0b3cab4adfb 10.1.4.178:6379 slots:[12288-16383] (4096 slots)
master
```

```
S: 46d001858e5d180011e933e07c22f192af99c998 10.1.5.254:6379 replicates
13c6cdda755318701a517eb75897e6012126fda0
S: e5bdc57415b7c26f3ddda8214100e53fb9266f95 10.1.4.206:6379 replicates
13c6cdda755318701a517eb75897e6012126fda0
S: b2af8b7e89ab1450dab1960bc6f6074f084890fa 10.1.5.151:6379 replicates
3001e467965c9cecab45115ec4df6a8a211d2296
S: 1dee8cf205d02b64c9641682db67b173f573fab3 10.1.5.61:6379 replicates
ea68ed55f90ba1681098583636df291ed16314c9
S: 94b7cb801ac9fa02e1bb12c94f8b22db8fd9aa4a 10.1.4.60:6379 replicates
ccc1291a5dde817831116505e85fa0b3cab4adfb
Can I set the above configuration? (type 'yes' to accept): >>> Nodes configuration updated >>> Assign a
different config epoch to each node >>>
Sending CLUSTER MEET messages to join the cluster
Waiting for the cluster to join ... >>>
Performing Cluster Check (using node 10.1.4.84:6379)
M: 13c6cdda755318701a517eb75897e6012126fda0 10.1.4.84:6379 slots:[0-4095] (4096 slots) master 2
additional replica(s)
S: e5bdc57415b7c26f3ddda8214100e53fb9266f95 10.1.4.206:6379 slots: (0 slots) slave replicates
13c6cdda755318701a517eb75897e6012126fda0
M: ea68ed55f90ba1681098583636df291ed16314c9 10.1.5.239:6379 slots:[8192-12287] (4096 slots)
master 1 additional replica(s)
M: 3001e467965c9cecab45115ec4df6a8a211d2296 10.1.5.15:6379 slots:[4096-8191] (4096 slots) master
1 additional replica(s)
S: 94b7cb801ac9fa02e1bb12c94f8b22db8fd9aa4a 10.1.4.60:6379 slots: (0 slots) slave replicates
ccc1291a5dde817831116505e85fa0b3cab4adfb
S: 1dee8cf205d02b64c9641682db67b173f573fab3 10.1.5.61:6379 slots: (0 slots) slave replicates
ea68ed55f90ba1681098583636df291ed16314c9
S: b2af8b7e89ab1450dab1960bc6f6074f084890fa 10.1.5.151:6379 slots: (0 slots) slave replicates
3001e467965c9cecab45115ec4df6a8a211d2296
S: 46d001858e5d180011e933e07c22f192af99c998 10.1.5.254:6379 slots: (0 slots) slave replicates
13c6cdda755318701a517eb75897e6012126fda0
M: ccc1291a5dde817831116505e85fa0b3cab4adfb 10.1.4.178:6379 slots:[12288-16383] (4096 slots)
master 1 additional replica(s)
[OK] All nodes agree about slots configuration.
>>> Check for open slots...
>>> Check slots coverage...
[OK] All 16384 slots covered.
Cluster is operational on node 10.1.4.84.
Pushing data to master node 10.1.4.84...
OK Data successfully pushed to master node 10.1.4.84.
Data replication successful. ubuntu@ip-10-1-7-62:~/redis$
```