Change all image links to relative links
Matuzalem Muller authored 23 hours ago

cc9342dc

Name	Last commit	Last update
..		
rook	Public release	23 hours ago
wordpress	Public release	23 hours ago
README.md	Change all image links to relative links	23 hours ago

README.md

This documentation presents step by step instructions on how to run a WordPress application with MySQL database in a remote Kubernetes cluster using Rook as the storage orchestrator.

- Below are the versions of each software used in this project:
  - Rook chart v0.8.2 (beta)
  - WordPress chart v2.1.10 (App v4.9.8 - stable)
  - MySQL chart v0.10.1 (App 5.7.14 - sable)
  - Docker v17.03.3
  - kubectl v1.11.0
  - Kubernetes v.v1.11.1

Table of contents

Set up remote infrastructure

See this documentation for instructions on how to set up the remote infrastructure and local environment.

Install Rook

Install Rook Operator chart using Helm

Add the rook beta channel to Helm:

```
helm repo add rook-beta https://charts.rook.io/beta
```

Install the rook chart:

```
helm install rook-beta/rook-ceph --namespace rook-ceph-system --name rook-chart --set agent.flexVolumeDirPath=/v
```

Installing the operator will create 7 pods:

- 3 rook agents, which will be running in each node
- 3 rook discovers, which will be running in each node
- 1 rook operator, which will be running in the master node

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

```
matuzalem-macbook:rook matuzalem$ kubectl get pod -n rook-ceph-system
NAME                                  READY     STATUS    RESTARTS   AGE
rook-ceph-agent-4vbxb                 1/1       Running   0          27m
rook-ceph-agent-nzgtj                 1/1       Running   0          27m
rook-ceph-agent-shxvj                 1/1       Running   0          27m
rook-ceph-operator-67bf669467-cwzh6   1/1       Running   0          27m
rook-discover-2dhls                   1/1       Running   0          27m
rook-discover-8g2rm                   1/1       Running   0          27m
rook-discover-h2n8h                   1/1       Running   0          27m
matuzalem-macbook:rook matuzalem$
```

For more information about the configuration "agent.flexVolumeDirPath=/var/lib/kubelet/volumeplugins", visit this link

---

Create Rook Cluster

This will deploy a rook cluster with monitors (MON), OSDs and a manager (MGR). All the necessary requirements such as namespaces and roles will also be created. However, it will still be necessary to setup for what rook will be used (i.e. object store, filesystem, etc).

```
kubectl create -f k8s-deployment/rook/cluster.yaml
```

This command will create 10 pods:

- 3 monitors, which will be running in each node
- 3 osd prepare, which will run and complete in each node
- 3 osds, which will be running in each node
- 1 rook manager, which will be running in the master node

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

```
matuzalem-macbook:rook matuzalem$ kubectl get pod -n rook-ceph
NAME                                  READY     STATUS      RESTARTS   AGE
rook-ceph-mgr-a-5b49f47b87-fzsjm      1/1       Running     0          24m
rook-ceph-mon0-66sjw                  1/1       Running     0          25m
rook-ceph-mon1-k9mj9                  1/1       Running     0          25m
rook-ceph-mon2-2xjwl                  1/1       Running     0          25m
rook-ceph-osd-id-0-78674d5795-nvdxf   1/1       Running     0          24m
rook-ceph-osd-id-1-7bc6dcc877-8w2gs   1/1       Running     0          24m
rook-ceph-osd-id-2-5cc58d89b4-427gt   1/1       Running     0          24m
rook-ceph-osd-prepare-master-bb2l8    0/1       Completed   0          24m
rook-ceph-osd-prepare-worker1-48c8p   0/1       Completed   0          24m
rook-ceph-osd-prepare-worker2-dqfvr   0/1       Completed   0          24m
```

Install WordPress chart and create Rook volume & bucket to store files

Install NGINX Controller

Install controller so http requests are forwarded to WordPress pod and Rook Object Store:

```
helm install stable/nginx-ingress --name nginx --set rbac.create=true
```

Create Rook Storage Class

Deploy Rook Storage Class. Volumes will now be created using rook:

```
kubectl create -f k8s-deployment/rook/storage-class.yaml
```

Generate certificate for remote VMs

- Point your domain to both worker VM IPs
- Generate a certificate using Let's Encrypt: https://certbot.eff.org/lets-encrypt/debianstretch-other
- Combine both `cert.pem` and `privkey.pem` in one file and encode output to base64. Insert the encoded output to the `cert` parameter of the `secrets.yaml` file

```
cat file.txt | base64
```

- Encode `privkey.pem` and `cert.pem` to base64 and add both to `k8s-deployment/rook/secrets.yaml` file in the `tls.key` and `tls.crt` parameters, respectively

---

Create secrets for Rook Object Store

Create TLS secrets for Rook Object Store and Object Store Ingress resource. This will allow secure connections to be established with the Object Store:

```
kubectl create -f k8s-deployment/rook/secrets.yaml
```

Create Rook Object Store

Create the Object Store, which will expose a S3 API to store and manage data:

```
kubectl create -f k8s-deployment/rook/object-store.yaml
```

- For more information about Rook Object Store, see https://rook.io/docs/rook/master/object.html
- A new pod will be created in namespace `rook-ceph`. Wait for its status to change to Running before proceeding to the next step

---

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

```
matuzalem-macbook:tcc-engtelecom matuzalem$ kubectl get pod -n rook-ceph
NAME                                   READY     STATUS      RESTARTS   AGE
rook-ceph-mgr-a-5b49f47b87-fzsjm       1/1       Running     0          1d
rook-ceph-mon0-66sjw                   1/1       Running     0          1d
rook-ceph-mon1-k9mj9                   1/1       Running     0          1d
rook-ceph-mon2-2xjwl                   1/1       Running     0          1d
rook-ceph-osd-id-0-78674d5795-nvdxf    1/1       Running     0          1d
rook-ceph-osd-id-1-7bc6dcc877-8w2gs    1/1       Running     0          1d
rook-ceph-osd-id-2-5cc58d89b4-427gt    1/1       Running     0          1d
rook-ceph-osd-prepare-master-bb2l8     0/1       Completed   0          1d
rook-ceph-osd-prepare-worker1-48c8p    0/1       Completed   0          1d
rook-ceph-osd-prepare-worker2-dqfvr    0/1       Completed   0          1d
rook-ceph-rgw-my-store-7878f8b699-9krgl   1/1    Running     0          1d
```

Run Rook Toolbox

Rook toolbox allows to connect to the cluster via CLI and analyze the underlying Ceph system running cluster, which helps troubleshooting issues. It will also allow to launch a S3 client to create buckets and manage data in the Rook Object Store.

```
kubectl create -f k8s-deployment/rook/toolbox.yaml
```

Create S3 bucket using radosgw

Access the rook toolbox pod and install the s3cmd client to manage data in the Rook Object Store (you can also simply deploy a `s3cmd` container such as this one):

```
kubectl -n rook-ceph exec -it rook-tools-XXX bash
yum --assumeyes install s3cmd
```

Create rgw user to be able to manage data in the Object Store (still in the toolbox pod):

```
radosgw-admin user create --uid rook-user --display-name "A rook rgw User" --rgw-realm=my-store --rgw-zonegroup=
```

- Save the following output from the `radosgw-admin` command:

```
{
    "user": "rook-user",
    "access_key": "XXXXXXXXXXXXXXXXXXX",
    "secret_key": "XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX"
}
```

In the toolbox shell, export the following variables to use them when managing data with s3cmd (the values of `AWS_ACCESS_KEY_ID` and `AWS_SECRET_ACCESS_KEY` are variables from the previous step):

```
export AWS_HOST=rook-ceph-rgw-my-store.rook-ceph
export AWS_ENDPOINT=rook-ceph-rgw-my-store.rook-ceph.svc.cluster.local
export AWS_ACCESS_KEY_ID=XXXXXXXXXXXXXXXXXXX
export AWS_SECRET_ACCESS_KEY=XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
```

Create a S3 bucket using s3cmd:

```
s3cmd mb --no-ssl --host=${AWS_HOST} --host-bucket= s3://rookbucket
```

Save some data to later add to the bucket. For example, a picture:

```
curl -o image.jpg https://cdn.pixabay.com/photo/2017/02/19/16/01/mountaineer-2080138_960_720.jpg
```

- This image has no copyright and has been retrieved from this page

"Put" the data in the bucket created and change its permissions to public access:

```
s3cmd put image.jpg --no-ssl --host=${AWS_HOST} --host-bucket=  s3://rookbucket
s3cmd setacl s3://rookbucket/image.jpg --acl-public --no-ssl --host=${AWS_HOST} --host-bucket=s3://rookbucket
```

The following command can be used to list the objects stored in the bucket:

```
s3cmd ls s3://rookbucket --no-ssl --host=${AWS_HOST} --host-bucket=s3://rookbucket
```

- More s3cmd commands are available at https://s3tools.org/usage
- The flag `--no-ssl` is being used as this communication is internal to the cluster

---

Create Ingress record for S3 bucket

Add the domain that was previously used to create the certificate and is pointing to the remote worker nodes (VMs) to the `host` parameter of the `k8s-deployment/rook/object-ingress.yaml` file:

```
(line 13) host: __________
```

Create Ingress record for S3 bucket:

```
kubectl create -f k8s-deployment/rook/object-ingress.yaml
```

- You will now be able to access your image from outside the cluster over HTTPS by accessing the URL www.domain.com/rook/rookbucket/image.jpg (where www.domain.com is the domain that was previously used to create the certificate and is pointing to the remote worker nodes - VMs).

Install MySQL chart

Run MySQL database which will be used by WordPress:

```
helm install stable/mysql --name mysql --version v0.10.1 -f k8s-deployment/wordpress/mysql-values.yaml
```

- This will install WordPress and create volumes based in the storage class `rook-ceph-block`
- More configurable parameters can be checked at https://github.com/helm/charts/tree/master/stable/mysql

Create TLS secrets for WordPress

Encode the `privkey.pem` and `cert.pem` files generated from the certificate to base64 and add both to `k8s-deployment/wordpress/wordpress-secrets.yaml` file in the `tls.key` and `tls.crt` parameters, respectively.

Create TLS secrets so it's possible to connect to WordPress securely:

```
kubectl create -f k8s-deployment/wordpress/wordpress-secrets.yaml
```

Install WordPress chart

Change the `host` parameter in the `k8s-deployment/wordpress/wordpress-values.yaml` file to include the domain that is pointing to the remote nodes:

```
(line 100) - name: _________________
```

Install WordPress chart using Helm:

```
helm install stable/wordpress --name wordpress --version v2.1.10 -f k8s-deployment/wordpress/wordpress-values.ya
```

- This will install WordPress and create volumes based in the storage class `rook-ceph-block`
- More configurable parameters can be checked at https://github.com/helm/charts/tree/master/stable/wordpress

---

Common issues

- Can't install chart because there's already a chart with that name installed even though it was removed: delete chart again using `--purge` flag

  ```
  helm delete __chart__ --purge
  ```

- `rook-ceph` namespace stuck in terminating status: https://github.com/rook/rook/issues/1488#issuecomment-397241621

  ```
  kubectl -n rook-ceph patch clusters.ceph.rook.io rook-ceph -p '{"metadata":{"finalizers": []}}' --type=merge
  ```

- `rook-ceph-system` namespace is still available after deleting chart: this is a known issue described here. It is necessary to manually remove the resources created by the chart even after deleting the chart.
- Monitors failing to start: https://github.com/rook/rook.github.io/blob/master/docs/rook/v0.7/common-problems.md#failing-mon-pod
- OSDs failing to start: https://github.com/rook/rook.github.io/blob/master/docs/rook/v0.7/common-problems.md#osd-pods-are-failing-to-start
- Volume creation doesn't work: https://github.com/rook/rook.github.io/blob/master/docs/rook/v0.7/common-problems.md#volume-creation