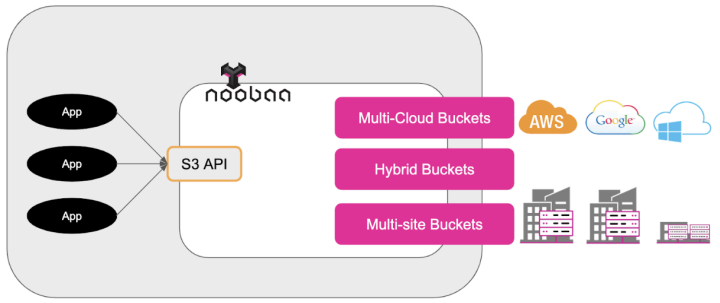
NooBaa Cheat Sheet

NooBaa is a free and open source data management platform for Kubernetes and Red Hat OpenShift platforms. NooBaa is designed to provide agnostic data management on-prem and in the cloud. It enables easy data portability and tiering seamlessly to the application.

THE BASICS

Cloud First

NooBaa's API is an AWS S3 compatible API and its backing stores can be a mixture of any on-prem persistent data volumes and cloud-native storage, and as such helps with cloud-first development approach. Developers can develop their application using NooBaa as their object storage regardless of the future hosting environment.



Efficiency

NooBaa's data management applies deduplication, compression, and encryption on the managed data. NooBaa is easy to deploy and has a minimal footprint on the platform.

Important Links

GitHub project	https://github.com/noobaa
Open issue	https://github.com/noobaa/noobaa-core/issues/new/choose
Contribute	https://github.com/noobaa/noobaa-operator#developing
NooBaa website	https://www.noobaa.io

INSTALLATION

The NooBaa CLI is the best tool to deploy and manage NooBaa. Install the NooBaa CLI wherever you need to run.

For Mac

```
brew install noobaa/noobaa/noobaa
```

Alternative for Mac

```
curl https://github.com/noobaa/noobaa-operator/releases/download/v2.0.6/noobaa-mac-v2.0.6 -L -o noobaa; chmod +x noobaa;mv noobaa /usr/local/bin
```

For Linux

```
curl https://github.com/noobaa/noobaa-operator/releases/download/v2.0.6/noobaa-linux-v2.0.6 -L -o noobaa; chmod +x noobaa;mv noobaa /usr/local/bin
```

Basic help

```
noobaa --help
```

NOOBAA DEPLOYMENT

Log in to your Kubernetes or OpenShift.

Run deployment with the following command. By default, it will deploy NooBaa's operator and pod under the `noobaa` namespace. Use `-n` for another namespace

```
noobaa install -n noobaa
```

The deployment will deploy everything and print to the screen all the information that is needed to start using NooBaa. You can skip to S3 information, connect your application, and start working.

continued on next page →

NOOBAA DEPLOYMENT CONTINUED

You can access the NooBaa management console with the management information:

```
#------------------#
#- Mgmt Addresses -#
#------------------#

ExternalDNS : []
ExternalIP  : []
NodePorts   : [https://10.0.2.15:32184]
InternalDNS : [https://noobaa-mgmt.noobaa:443]
InternalIP  : [https://10.102.65.220:443]
PodPorts    : [https://172.17.0.5:8443]

#--------------------#
#- Mgmt Credentials -#
#--------------------#

email: admin@noobaa.io
password: hLKP1mvNerAYTKmcpfrPMA==
system: noobaa

#----------------#
#- S3 Addresses -#
#----------------#

ExternalDNS : []
ExternalIP  : []
NodePorts   : [https://10.0.2.15:32247]
InternalDNS : [https://s3.noobaa:443]
InternalIP  : [https://10.104.50.70:443]
PodPorts    : [https://172.17.0.5:6443]

#------------------#
#- S3 Credentials  -#
#------------------#

AWS_ACCESS_KEY_ID: KtLbeGH84iEuVplRGHgK
AWS_SECRET_ACCESS_KEY:
d15xdTzuIu8IZ+dYWTE4HpSlozubTAMbmjLEUy1U
```

MONITORING AND MAINTENANCE

Deployment status and credentials

You can get the deployment information and credentials at any time with:

```
noobaa status -n noobaa
```

Another option to get more little hacks is using the following command:

```
kubectl describe noobaa -n noobaa
```

Skip to the welcome section:

```
Welcome to NooBaa!
  -----------------

  Lets get started:

  1. Connect to Management console:

    Read your mgmt console login information (email
& password) from secret: "noobaa-admin".
     kubectl get secret noobaa-admin -n noobaa -o json
| jq '.data|map_values(@base64d)'

    Open the management console service - take
External IP/DNS or Node Port or use port
forwarding:

      kubectl port-forward -n noobaa service/
noobaa-mgmt 11443:8443 &
      open https://localhost:11443

  2. Test S3 client:

    kubectl port-forward -n noobaa service/s3
10443:443 &
    NOOBAA_ACCESS_KEY=$(kubectl get secret noobaa-
admin -n noobaa -o json | jq -r
'.data.AWS_ACCESS_KEY_ID|@base64d')
    NOOBAA_SECRET_KEY=$(kubectl get secret noobaa-
admin -n noobaa -o json | jq -r
'.data.AWS_SECRET_ACCESS_KEY|@base64d')
    alias s3='AWS_ACCESS_KEY_ID=$NOOBAA_ACCESS_KEY
AWS_SECRET_ACCESS_KEY=$NOOBAA_SECRET_KEY aws --
endpoint https://localhost:10443 --no-verify-ssl
s3'
    s3 ls
```

** Note: you will need jq (json cli processor) and aws-cli if you don't have it.*

Add cloud backing store

Use this command to add new backing stores, used by NooBaa to keep the data. By default, it uses internal temporary datastore, used for POC only.

Make sure that you are using an existing bucket on AWS for the `--target-bucket` flag:

```
noobaa backingstore create aws-s3 noobaa-default-
backing-store
 --access-key=<AWS ACCESS KEY> --secret-key=<AWS
SECRET KEY> --target-bucket=bucket-for-noobaa-on-
aws
```

Provision object service to an application

Similar to Persistent Volumes, developers can add a few more lines to their deployment YAML and get the endpoint, access key, and secret key available as environment variables.

Add the bold lines to the application YAML. That's all

continued on next page →

MONITORING AND MAINTENANCE CONTINUED

An example of a job can be found below. The first part is the Object Bucket Claim itself. The second part is the mapping between the bucket claim result, which is a config map with data and a secret with the credentials.

```
apiVersion: objectbucket.io/v1alpha1
kind: ObjectBucketClaim
metadata:
  name: "obc-test"
spec:
  generateBucketName: "obc-test-noobaa"
  storageClassName: noobaa
---
apiVersion: batch/v1
kind: Job
metadata:
  name: testjob
spec:
  template:
    spec:
      restartPolicy: OnFailure
      containers:
        - image: devrepo/testcontainer:1.0
          name: test
          env:
            - name: BUCKET_NAME
              valueFrom:
                configMapKeyRef:
                  name: obc-test
                  key: BUCKET_NAME
            - name: BUCKET_HOST
              valueFrom:
                configMapKeyRef:
                  name: obc-test
                  key: BUCKET_HOST
            - name: BUCKET_PORT
              valueFrom:
                configMapKeyRef:
                  name: obc-test
                  key: BUCKET_PORT
            - name: AWS_ACCESS_KEY_ID
              valueFrom:
                secretKeyRef:
                  name: obc-test
                  key: AWS_ACCESS_KEY_ID
            - name: AWS_SECRET_ACCESS_KEY
              valueFrom:
                secretKeyRef:
                  name: obc-test
                  key: AWS_SECRET_ACCESS_KEY
      volumes:
      - name: persistent-storage
        emptyDir: {}
```

Once you applied the YAML, you can expect the following environment variables:

1. BUCKET_HOST – The endpoint to use in the application.
2. BUCKET_PORT – The port available for this application.
3. BUCKET_NAME – Requested or generated bucket name.
4. AWS_ACCESS_KEY_ID – Access key which is part of the credentials.
5. AWS_SECRET_ACCESS_KEY – Secret access key which is part of the credentials.

Providing object service via CLI

Using the CLI to generate an email with the details of a new bucket and credentials:

```
noobaa obc create test-bucket
```

The output would be:

```
INFO[0000] ✅ Created: ObjectBucketClaim "test-bucket"
```

The bucket name can be found with the following command:

```
kubectl describe cm test-bucket -n noobaa
```

The output would be:

```
….
Data
====
BUCKET_HOST:
----
10.0.2.15
BUCKET_NAME:
----
test-bucket-06e1b150-9be1-401d-95e9-ef39b55ef03a
BUCKET_PORT:
----
32247
BUCKET_REGION:
----

BUCKET_SUBREGION:
----

Events:  <none>
```

Bucket credentials can be found with the following command:

```
kubectl get secret test-bucket -n noobaa -o json |
jq -r '.data.AWS_ACCESS_KEY_ID|@base64d'; kubectl
get secret test-bucket -n noobaa -o json | jq -r
'.data.AWS_SECRET_ACCESS_KEY|@base64d'
```

The output would be:

```
WzA04ppdOlhIr7AbFhVC
jFvspZ1qgj36enwLe0TyXTpd3cBZi5YKAbG4LQte
```

UNINSTALL

To uninstall NooBaa deployment, run the following command:

```
noobaa uninstall -n <namespace>
```

continued on next page →

QUICK TROUBLESHOOTING

Check the pods status

Run the following command and make sure all pods are in status Running and Ready.

```
kubectl get pod -n noobaa
```

For a pod that is not running, run the following command. You should be able to figure out the reason.

```
kubectl describe pod <pod name> -n noobaa
```

Common reasons include:

1. Not enough storage capacity. To solve it, you can uninstall noobaa and install it with the flag `--db-volume-size-gb=<size in GB>`
2. Not enough CPU/Memory. To solve it, you will need to add resources to your environment, as the default setting is minimal.

ABOUT THE AUTHOR

Eran Tamir

Eran Tamir is a Senior Principal Product Manager within Red Hat Storage covering NooBaa focused on enterprise data services for distributed hybrid cloud.

https://github.com/tamireran