記事
Tomohiro Iwamoto · 2021年5月18日 18m read

microk8sでKubernetesをお手軽に試す

目的

Japan Virtual Summit 2021で、Kubernetesに関するセッションを実施させていただいたのですが、AzureのアカウントやIRIS評価用ライセンスキーをお持ちの方が対象になっていました。もう少し手軽に試してみたいとお考えの開発者の方もおられると思いますので、本記事では仮想環境でも利用可能なk8sの軽量実装であるmirok8sで、IRIS Community Editionを稼働させる手順をご紹介いたします。

2022/1/7 若干の加筆・修正しました

マルチノード化する手順はこちらに記載しています。

参考までに私の環境は以下の通りです。

用途	O/S	ホストタイプ	IP
クライアントPC	Windows10 Pro	物理ホスト	172.X.X.30/24, (vmware NAT)192.168.11.1/24
mirok8s環境	ubuntu 20.04.1 LTS	上記Windows10上の仮想ホスト(vmware)	192.168.11.49/24

ubuntuは、ubuntu-20.04.1-live-server-amd64.isoを使用して、最低限のサーバ機能のみをインストールしました。

概要

IRIS Community EditionをKubernetesのStatefulSetとしてデプロイする手順を記します。
IRISのシステムファイルやユーザデータベースを外部保存するための永続化ストレージには、microk8shostpath もしくはLonghornを使用します。
使用するコードはこちらにあります。

インストレーション

microk8sをインストール・起動します。

```
$ sudo snap install microk8s --classic --channel=1.20
$ sudo usermod -a -G microk8s $USER
$ microk8s start
$ microk8s enable dns registry storage metallb
  ?
  ?
Enabling MetalLB
Enter each IP address range delimited by comma (e.g. '10.64.140.43-10.64.140.49,192.168.0.105-192.168.0.111'):192.168.11.110-192.168.11.130
```

ロードバランサに割り当てるIPのレンジを聞かれますので、適切な範囲を設定します。私の環境はk8sが稼働しているホストのCIDRは192.168.11.49/24ですので適当な空いているIPのレンジとして、[192.168.11.110-192.168.11.130]と指定しました。

この時点で、シングルノードのk8s環境が準備されます。

```
$ microk8s kubectl get pods -A
NAMESPACE            NAME                                         READY   STATUS    REST
ARTS   AGE
metallb-system       speaker-
gnljw                             1/1     Running   0          45s
metallb-system       controller-559b68bfd8-bkrdz                  1/1     Running   0
       45s
kube-system          hostpath-
provisioner-5c65fbdb4f-2z9j8     1/1     Running   0          48s
kube-system          calico-node-
bwp2z                         1/1     Running   0          65s
kube-system          coredns-86f78bb79c-
gnd2n                   1/1     Running   0          57s
kube-system          calico-kube-controllers-847c8c99d-
pzvnb   1/1     Running   0          65s
container-registry   registry-9b57d9df8-bt9tf                     1/1     Running   0
       48s
$ microk8s kubectl get node
NAME     STATUS   ROLES    AGE   VERSION
ubuntu   Ready    <none>   10d   v1.20.7-34+df7df22a741dbc
```

kubectl実行時に毎回microk8sをつけるのは手間なので、下記コマンドでエリアスを設定しました。以降の例ではmicrok8sを省略しています。

> 注意
> すでに"普通の"kubectlがインストールされていると、そちらが優先されてしまいますので、alias名をkubectl2にするなど衝突しないようにしてください。

```
$ sudo snap alias microk8s.kubectl kubectl
$ kubectl get node
NAME     STATUS   ROLES    AGE   VERSION
ubuntu   Ready    <none>   10d   v1.20.7-34+df7df22a741dbc
```

> 元の状態に戻すには sudo snap unalias kubectl

環境が正しく動作することを確認するためにIRISを起動してみます。下記コマンドの実行で、USERプロンプトが表示されるはずです。

```
$ kubectl run iris --image=intersystemsdc/iris-community:2022.1.0.209.0-zpm
$ watch kubectl get pod
$ kubectl exec -ti iris -- iris session iris
USER>
```

今後の作業に備えて、作成したPODを削除しておきます。

```
$ kubectl delete pod iris
```

起動

```
$ kubectl apply -f mk8s-iris.yml
```

> IRIS Community版なので、ライセンスキーもコンテナレジストリにログインするためのimagePullSecretsも指定していません。

しばらくするとポッドが2個作成されます。これでIRISが起動しました。

```
$ kubectl get pod
NAME     READY   STATUS    RESTARTS   AGE
data-0   1/1     Running   0          107s
data-1   1/1     Running   0          86s
$ kubectl get statefulset
NAME   READY   AGE
data   2/2     3m32s
$ kubectl get service
NAME         TYPE           CLUSTER-IP       EXTERNAL-IP      PORT(S)           AGE
kubernetes   ClusterIP      10.152.183.1     <none>           443/TCP           30m
iris         ClusterIP      None             <none>           52773/TCP         8m55s
iris-ext     LoadBalancer   10.152.183.137   192.168.11.110   52773:31707/TCP   8m55s
```

この時点で、下記コマンドでirisのREST/APIで提供されているメトリックスを取得できるはずです。

```
$ curl http://192.168.11.110:52773/api/monitor/metrics
```

ポッドのSTATUSがrunningにならない場合、下記コマンドでイベントを確認できます。イメージ名を間違って指定していてPullが失敗したり、なんらかのリソースが不足していることが考えられます。

```
$ kubectl describe pod data-0

Events:
  Type     Reason            Age                 From               Message
  ----     ------            ----                ----               -------
  Warning  FailedScheduling  4m (x3 over 4m3s)   default-scheduler  0/1 nodes are avai
lable: 1 pod has unbound immediate PersistentVolumeClaims.
  Normal   Scheduled         3m56s               default-
scheduler  Successfully assigned default/data-0 to ubuntu
  Normal   Pulling           3m56s               kubelet            Pulling image "con
tainers.intersystems.com/intersystems/iris-community:2021.1.0.215.3"
  Normal   Pulled            69s                 kubelet            Successfully pulle
d image "containers.intersystems.com/intersystems/iris-
community:2021.1.0.215.3" in 2m46.607639152s
  Normal   Created           69s                 kubelet            Created container
iris
  Normal   Started           68s                 kubelet            Started container
iris
```

下記コマンドでirisにO/S認証でログインできます。

```
$ kubectl exec -it data-0 -- iris session iris
Node: data-0, Instance: IRIS
USER>
```

下記で各IRISインスタンスが使用するPVCが確保されていることが確認できます。

```
$ kubectl get pvc
NAME              STATUS   VOLUME                                     CAPACITY   ACCE
SS MODES   STORAGECLASS        AGE
dbv-mgr-data-0    Bound    pvc-fbfdd797-f90d-4eac-83a8-f81bc608d4bc   5Gi        RWO
           microk8s-hostpath   12m
dbv-data-data-0   Bound    pvc-b906a687-c24c-44fc-
acd9-7443a2e6fec3   5Gi        RWO            microk8s-hostpath   12m
dbv-mgr-data-1    Bound    pvc-137b0ccf-406b-40ac-
b8c5-6eed8534a6fb   5Gi        RWO            microk8s-hostpath   9m3s
dbv-data-data-1   Bound    pvc-4f2be4f1-3691-4f7e-
ba14-1f0461d59c76   5Gi        RWO            microk8s-hostpath   9m3s
```

dfを実行すると、データベースファイルを配置するための/vol-dataがマウント対象に表示されていなくて、一瞬、？となりますが、--all指定すると表示されます。

```
irisowner@data-0:~$ df --all
Filesystem     1K-blocks     Used Available Use% Mounted on
  ?
  ?
/dev/sda2      205310952 26925908 167883056  14% /iris-mgr
/dev/sda2      205310952 26925908 167883056  14% /vol-data
/dev/sda2      205310952 26925908 167883056  14% /irissys/cpf
/dev/sda2      205310952 26925908 167883056  14% /etc/hosts
/dev/sda2      205310952 26925908 167883056  14% /dev/termination-log
/dev/sda2      205310952 26925908 167883056  14% /etc/hostname
/dev/sda2      205310952 26925908 167883056  14% /etc/resolv.conf
  ?
  ?
```

/dev/sda2はコンテナ内のデバイスではなく、ホスト上のデバイスなので、microk8s-hostpathの仕組み上、そのような表示になるのでしょう。

個別のポッド上のIRISの管理ポータルにアクセスする

下記コマンドで各ポッドの内部IPアドレスを確認します。

```
$ kubectl get pod -o wide
NAME     READY   STATUS    RESTARTS   AGE   IP             NODE     NOMINATED NODE
READINESS GATES
data-0   1/1     Running   0          46m   10.1.243.202   ubuntu   <none>
<none>
data-1   1/1     Running   0          45m   10.1.243.203   ubuntu   <none>
<none>
$
```

私の仮想環境のLinuxはGUIがありませんので、下記のコマンドを実行することで、Windowsのブラウザから管理ポータルにアクセスできるようにしました。

```
$ kubectl port-forward data-0 --address 0.0.0.0 9092:52773
$ kubectl port-forward data-1 --address 0.0.0.0 9093:52773
```

対象	URL	ユーザ	パスワード
ポッドdata-0上のIRIS	http://192.168.11.49:9092/csp/sys/%25CSP.Portal.Home.zen	SuperUser	SYS
ポッドdata-1上のIRIS	http://192.168.11.49:9093/csp/sys/%25CSP.Portal.Home.zen	SuperUser	SYS

パスワードはCPFのPasswordHashで指定しています

データベースの構成を確認してください。下記のデータベースがPV上に作成されていることを確認できます。

データベース名	path
IRISSYS	/iris-mgr/IRISconf.d/mgr/
TEST-DATA	/vol-data/TEST-DATA/

停止

作成したリソースを削除します。

```
$ kubectl delete -f mk8s-iris.yml --wait
```

これで、IRISのポッドも削除されますが、PVCは保存されたままになっていることに留意ください。これにより、次回に同じ名前のポッドが起動した際には、以前と同じボリュームが提供されます。つまり、ポッドのライフサイクルと、データベースのライフサイクルの分離が可能となります。次のコマンドでPVCも削除出来ます(データベースの内容も永久に失われます)。

```
$ kubectl delete pvc -l app=iris
```

O/Sをシャットダウンする際には下記を実行すると、k8s環境を綺麗に停止します。

```
$ microk8s stop
```

O/S再起動後には下記コマンドでk8s環境を起動できます。

```
$ microk8s start
```

microk8s環境を完全に消去したい場合は、microk8s
stopを「実行する前」に下記を実行します。(やたらと時間がかかりました。日頃は実行しなくて良いと思います)

```
$ microk8s reset --destroy-storage
```

観察

ストレージの場所

興味本位の観察ではありますが、/iris-mgr/はどこに存在するのでしょう？microk8sはスタンドアロンで起動するk8s環境ですので、storageClassNameがmicrok8s-hostpathの場合、ファイルの実体は同ホスト上にあります。まずはkubectl get pvで、作成されたPVを確認します。

```
$ kubectl apply -f mk8s-iris.yml
$ kubectl get pv
NAME                                       CAPACITY   ACCESS MODES   RECLAIM POLICY
 STATUS   CLAIM                                 STORAGECLASS        REASON   AGE
pvc-ee660281-1de4-4115-a874-9e9c4cf68083   20Gi       RWX            Delete
 Bound    container-registry/registry-claim   microk8s-hostpath            37m
pvc-772484b1-9199-4e23-9152-d74d6addd5ff   5Gi        RWO            Delete
 Bound    default/dbv-data-data-0             microk8s-hostpath            10m
pvc-112aa77e-2f2f-4632-9eca-4801c4b3c6bb   5Gi        RWO            Delete
 Bound    default/dbv-mgr-data-0              microk8s-hostpath            10m
pvc-e360ef36-627c-49a4-a975-26b7e83c6012   5Gi        RWO            Delete
 Bound    default/dbv-mgr-data-1              microk8s-hostpath            9m55s
pvc-48ea60e8-338e-4e28-9580-b03c9988aad8   5Gi        RWO            Delete
 Bound    default/dbv-data-data-1             microk8s-hostpath            9m55s
```

ここで、data-0ポッドのISC_DATA_DIRECTORYに使用されている、default/dbv-mgr-data-0 をdescribeします。

```
$ kubectl describe pv pvc-112aa77e-2f2f-4632-9eca-4801c4b3c6bb
  ?
  ?
Source:
    Type:          HostPath (bare host directory volume)
    Path:          /var/snap/microk8s/common/default-storage/default-dbv-mgr-
data-0-pvc-112aa77e-2f2f-4632-9eca-4801c4b3c6bb
```

このpathが実体ファイルのありかです。

```
$ ls /var/snap/microk8s/common/default-storage/default-dbv-mgr-
data-0-pvc-112aa77e-2f2f-4632-9eca-4801c4b3c6bb/IRIS_conf.d/
ContainerCheck  csp  dist  httpd  iris.cpf  iris.cpf_20210517  _LastGood_.cpf  mgr
$
```

> storageClassNameにhostpathは使用しないでください。microk8shostpathとは異なり、同じフォルダに複数IRISが同居するような状態(破壊された状態)になってしまいます。

ホスト名の解決

StatefulSetでは、各ポットにはmetadata.nameの値に従い、data-0, data-1などのユニークなホスト名が割り当てられます。
ポッド間の通信に、このホスト名を利用するために、Headless Serviceを使用しています。

```
kind: StatefulSet
```

```
metadata:
  name: data

kind: Service
spec:
  clusterIP: None # Headless Service
```

> この特徴は、ノード間で通信をするShardingのような機能を使用する際に有益です。本例では直接の便益はありません。

nslookupを使いたいのですが、kubectlやk8sで使用されているコンテナランタイム(ctr)にはdockerのようにrootでログインする機能がありません。また、IRISのコンテナイメージはセキュリティ上の理由でsudoをインストールしていませんので、イメージのビルド時以外のタイミングで追加でソフトウェアをapt
install出来ません。ここではbusyboxを追加で起動して、そこでnslookupを使ってホスト名を確認します。

```
$ kubectl run -i --tty --image busybox:1.28 dns-test --restart=Never --rm
/ # nslookup data-0.iris
Server:    10.152.183.10
Address 1: 10.152.183.10 kube-dns.kube-system.svc.cluster.local

Name:      data-0.iris
Address 1: 10.1.243.202 data-0.iris.default.svc.cluster.local
/ #
```

10.152.183.10はk8sが用意したDNSサーバです。data-0.irisには10.1.243.202というIPアドレスが割り当てられていることがわかります。FQDNはdata-0.iris.default.svc.cluster.localです。同様にdata-1.irisもDNSに登録されています。

独自イメージを使用する場合

現在のk8sはDockerを使用していません。ですので、イメージのビルドを行うためには別途Dockerのセットアップが必要です。

> k8sはあくまで運用環境のためのものです

サンプルイメージを使用する場合

イメージはどんな内容でも構いませんが、ここでは例
としてsimple
を使用します。このイメージはMYAPPネームスペース上で、ごく簡単なRESTサービスを提供します。データの保存場所をコンテナ内のデータベース(MYAPP-DATA)から外部データベース(MYAPP-DATA-
EXT)に切り替えるために、cpfのactionにModifyNamespaceを使用しています。
mk8s-simple.ymlとしてご用意しました(mk8s-iris.ymlとほとんど同じです)。これを使用して起動します。

自分でイメージをビルドする場合

ご自身でビルドを行いたい場合は、下記の手順でmicrok8sが用意した組み込みのコンテナレジストリに、イメージをpushします。

> 内容のわからない非公式コンテナイメージって...ちょっと気持ち悪いかも、ですよね。

(Docker及びdocker-composeのセットアップが済んでいること)

```
$ git clone https://github.com/IRISMeister/simple.git
$ cd simple
$ ./build.sh
$ docker tag dpmeister/simple:latest localhost:32000/simple:latest
$ docker push localhost:32000/simple:latest
```

このイメージを使用するように、ymlを書き換えます。

```
mk8s-simple.yml???
?)        image: dpmeister/simple:latest
?)        image: localhost:32000/simple
```

起動方法

既にポッドを起動しているのであれば、削除します。

```
$ kubectl delete -f mk8s-iris.yml
$ kubectl delete pvc -l app=iris
```

```
$ kubectl apply -f mk8s-simple.yml
$ kubectl get svc
NAME         TYPE           CLUSTER-IP       EXTERNAL-IP      PORT(S)           AGE
kubernetes   ClusterIP      10.152.183.1     <none>           443/TCP           3h36m
iris         ClusterIP      None             <none>           52773/TCP         20m
iris-ext     LoadBalancer   10.152.183.224   192.168.11.110   52773:30308/TCP   20m
$
$ curl -s -H "Content-Type: application/json; charset=UTF-8" -H "Accept:application/j
son" "http://192.168.11.110:52773/csp/myapp/get" --user "appuser:SYS" | python3 -mjso
n.tool
{
    "HostName": "data-1",
    "UserName": "appuser",
    "Status": "OK",
    "TimeStamp": "05/17/2021 19:34:00",
    "ImageBuilt": "05/17/2021 10:06:27"
}
```

curlの実行を繰り返すと、HostName(RESTサービスが動作したホスト名)がdata-0だったりdata-1だったりしますが、これは(期待通りに)ロードバランスされているためです。

> まれにログインに失敗したり、待たされることがあります。Community EditionはMAX 5セッションまでですが、以前の何らかの操作によりその上限を超えてしまっている可能性があります。その場合、ライセンス上限を超えた旨のメッセージがログされます。

```
$ kubectl logs data-0
```

```
  ?
  ?
05/17/21-19:21:17:417 (2334) 2 [Generic.Event] License limit exceeded 1 times since i
nstance start.
```

Longhornを使用する場合

> 分散KubernetesストレージLonghornについては、こちらを参照ください。

IRISのようなデータベース製品にとってのメリットは、クラウド環境でアベーラビリティゾーンをまたいだデータベースの冗長構成を組めることにあります。アベーラビリティゾーン全体の停止への備えは、ミラー構成を組むことで実現しますが、Kubernetes環境であれば、分散Kubernetesストレージの採用という選択子が増えます。

> ミラー構成とは異なり、データベース以外のファイルも保全できるというメリットもあります。ただしパフォーマンスへの負のインパクトには要注意です。

起動方法

longhornを起動し、すべてのポッドがREADYになるまで待ちます。

```
$ kubectl apply -f https://raw.githubusercontent.com/longhorn/longhorn/v1.2.3/deploy/
longhorn.yaml
$ kubectl -n longhorn-system get pods
NAME                                        READY   STATUS    RESTARTS   AGE
longhorn-ui-5b864949c4-72qkz                1/1     Running   0          4m3s
longhorn-manager-wfpnl                      1/1     Running   0          4m3s
longhorn-driver-deployer-ccb9974d5-w5mnz    1/1     Running   0          4m3s
instance-manager-e-5f14d35b                 1/1     Running   0          3m28s
instance-manager-r-a8323182                 1/1     Running   0          3m28s
engine-image-ei-611d1496-qscbp              1/1     Running   0          3m28s
csi-attacher-5df5c79d4b-gfncr               1/1     Running   0          3m21s
csi-attacher-5df5c79d4b-ndwjn               1/1     Running   0          3m21s
csi-provisioner-547dfff5dd-pj46m            1/1     Running   0          3m20s
csi-resizer-5d6f844cd8-22dpp                1/1     Running   0          3m20s
csi-provisioner-547dfff5dd-86w9h            1/1     Running   0          3m20s
csi-resizer-5d6f844cd8-zn97g                1/1     Running   0          3m20s
csi-resizer-5d6f844cd8-8nmfw                1/1     Running   0          3m20s
csi-provisioner-547dfff5dd-pmwsk            1/1     Running   0          3m20s
longhorn-csi-plugin-xsnj9                   2/2     Running   0          3m19s
csi-snapshotter-76c6f569f9-wt8sh            1/1     Running   0          3m19s
csi-snapshotter-76c6f569f9-w65xp            1/1     Running   0          3m19s
csi-attacher-5df5c79d4b-gcf4l               1/1     Running   0          3m21s
csi-snapshotter-76c6f569f9-fjx2h            1/1     Running   0          3m19s
```

mk8s-iris.ymlの全て(2箇所あります)のstorageClassNameをlonghornに変更してください。
もし、microk8shostpathで既に起動しているのであれば、ポッド、PVCともに全て削除したうえで、上述の手順を実行してください。つまり...

```
$ kubectl delete -f mk8s-iris.yml --wait
$ kubectl delete pvc -l app=iris
```

mk8s-iris.yml??

```
?)storageClassName: microk8s-hostpath
?)storageClassName: longhorn

$ kubectl apply -f mk8s-iris.yml
```

> マウントしたLonghorn由来のボリュームのオーナがrootになっていたのでsecurityContext:fsGroupを指定しています。これ無しでは、データベース作成時にプロテクションエラーが発生します。
> fsGroup指定なしの場合
>
> ```
> $ kubectl exec -it data-0 -- ls / -l
> drwxr-xr-x   3 root      root         4096 May 18 15:40 vol-data
> ```
>
> fsGroup指定ありの場合
>
> ```
> $ kubectl exec -it data-0 -- ls / -l
> drwxrwsr-x   4 root      irisowner     4096 Jan  5 17:09 vol-data
> ```
>
> 2021.1まではfsGroup:52773を指定すると動きましたが、2022.1以後はfsGroup:51773を指定すると動きました。

下記を実行すれば、Windowsのブラウザから、Longhorn UIを参照できます。

```
$ microk8s enable ingress
$ kubectl apply -f longhorn-ui-ingress.yml
```

ポート80を他の用途に使ってる場合、下記のようにport-forwardを使う方法もあります。この場合ポートは8080なので、URLはこちらになります。

```
$ kubectl -n longhorn-system port-forward svc/longhorn-
frontend 8080:80 --address 0.0.0.0
```

> UIで、VolumeのStateが"Degraded"になっていますが、これはReplicaの数がnumberOfReplicasの既定値3を満たしていないためです。

以降の操作は、同様です。不要になれば削除します。

```
$ kubectl delete -f mk8s-iris.yml
$ kubectl delete pvc --all
```

削除方法

Longhorn環境が不要になった場合は、下記のコマンドで削除しておくと良いようです(いきなりdeleteしてはダメ)。

```
$ kubectl create -f https://raw.githubusercontent.com/longhorn/longhorn/v1.2.3/uninst
```

```
all/uninstall.yaml
$ kubectl get job/longhorn-uninstall -n default -w
NAME                 COMPLETIONS   DURATION   AGE
longhorn-uninstall   1/1           79s        97s
^C
$ kubectl delete -f https://raw.githubusercontent.com/longhorn/longhorn/v1.2.3/deploy
/longhorn.yaml
$ kubectl delete -f https://raw.githubusercontent.com/longhorn/longhorn/v1.2.3/uninst
all/uninstall.yaml
```

apply時のエラー

Longhornの前回の使用時に綺麗に削除されなかった場合に、apply時に下記のようなエラーが出ることがあります。

```
$ kubectl apply -f https://raw.githubusercontent.com/longhorn/longhorn/master/deploy/
longhorn.yaml
  ?
  ?
Error from server (Forbidden): error when creating "https://raw.githubusercontent.com
/longhorn/longhorn/master/deploy/longhorn.yaml": serviceaccounts "longhorn-service-ac
count" is forbidden: unable to create new content in namespace longhorn-
system because it is being terminated
Error from server (Forbid
```

上記のuninstall.yamlを使った削除手順をもう一度実行したら回復しました。

その他気づいた事

storageClassにmicrok8shostpathを指定した場合、 マルチノード環境
ではsecurityContext:fsGroupが正しく機能しないようです。その結果、下記のようなエラーが発生して、データベースの作成に失敗します(Error=-13はPermission denieです)。longhornは問題なく動作しました。

```
01/07/22-23:11:32:729 (1205) 1 [Utility.Event] ERROR #503: Error executing [Actions]
section in file /iris-mgr/IRIS_conf.d/merge_actions.cpf
01/07/22-23:11:32:729 (1205) 1 [Utility.Event] ERROR #507: Action 'CreateDatabase' fa
iled at line 2, Method Config.CPF:CreateOneDatabase(), Error=ERROR #5032: Cannot crea
te directory '/vol-data/db/TEST-DATA/, Error=-13'
```

InterSystems Kubernetes Operator

IKOもmicrok8sで動作しますが、Community向けの機能ではないので、今回のご紹介は見送りました。

#Kubernetes #InterSystems IRIS #InterSystems IRIS for Health

ソースURL:
https://jp.community.intersystems.com/post/microk8s%E3%81%A7kubernetes%E3%82%92%E3%81%8A%E6%89%8B%E8%BB%BD%E3%81%AB%E8%A9%A6%E3%81%99