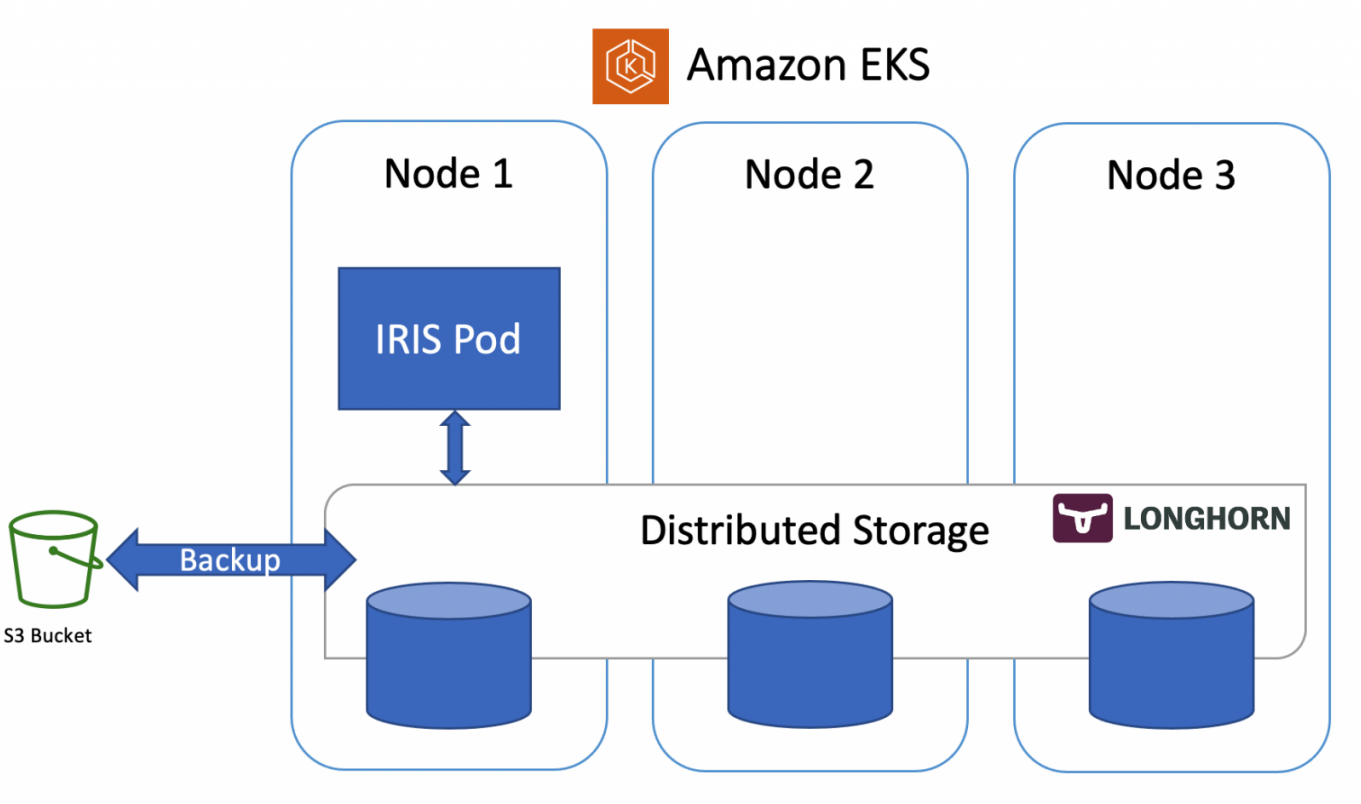
記事
Toshihiko Minamoto · 2021年9月1日 8m read

Open Exchange

Amazon EKSとIRIS。 高可用性とバックアップ

記事で使用されているすべてのソースコード: https://github.com/antonum/ha-iris-k8s



前の記事
では、従来型のミラーリングではなく分散ストレージに基づいて、高可用性のあるk8sでIRISをセットアップする方法について説明しました。 その記事では例としてAzure AKSクラスタを使用しました。
この記事では引き続き、k8sで可用性の高い構成を詳しく見ていきますが、 今回は、Amazon EKS（AWSが管理するKubernetesサービス）に基づき、Kubernetes Snapshotに基づいてデータベースのバックアップと復元を行うためのオプションが含まれます。

インストール

早速作業に取り掛かりましょう。 まず、AWSアカウントが必要です。AWS CLI、kubectl、およびeksctl ツールがインストールされている必要があります。
新しいクラスタを作成するために、次のコマンドを実行します。

```
eksctl create cluster \
--name my-cluster \
--node-type m5.2xlarge \
--nodes 3 \
--node-volume-size 500 \
--region us-east-1
```

このコマンドは約15分掛けてEKSクラスタをデプロイし、それをkubectlツールのデフォルトのフォルダに設定します。 デプロイを確認するには、次のコードを実行します。

```
kubectl get nodes
NAME                                           STATUS   ROLES    AGE   VERSION
ip-192-168-19-7.ca-
central-1.compute.internal    Ready    &amp;lt;none&gt;   18d   v1.18.9-eks-d1db3c
ip-192-168-37-96.ca-
central-1.compute.internal   Ready    &amp;lt;none&gt;   18d   v1.18.9-eks-d1db3c
ip-192-168-76-18.ca-
central-1.compute.internal   Ready    &amp;lt;none&gt;   18d   v1.18.9-eks-d1db3c
```

次に、Longhorn分散ストレージエンジンをインストールします。

```
kubectl create namespace longhorn-system
kubectl apply -f https://raw.githubusercontent.com/longhorn/longhorn/v1.1.0/deploy/is
csi/longhorn-iscsi-installation.yaml --namespace longhorn-system
kubectl apply -f https://raw.githubusercontent.com/longhorn/longhorn/master/deploy/lo
nghorn.yaml --namespace longhorn-system
```

そして最後に、IRIS自体をインストールします。

```
kubectl apply -f https://github.com/antonum/ha-iris-k8s/raw/main/tldr.yaml
```

この時点で、Longhorn分散ストレージとIRISデプロイがインストールされたEKSクラスタが完全に機能できる状態になります。 前の記事に戻って、クラスタとIRISデプロイにあらゆるダメージを与えて、システムがどのように修復するのかを確認するとよいでしょう。 「障害をシミュレートする」セクションをご覧ください。

特典1 ARM上のIRIS

IRIS EKSとLonghornはARMアーキテクチャをサポートしているため、ARMアーキテクチャに基づき、AWS Gravition 2インスタンスを使用して同じ構成をデプロイできます。

EKSノードのインスタンスタイプを 'm6g'
ファミリーに変更し、IRISイメージをARMベースに変更するだけです。

eksctl create cluster \
--name my-cluster-arm \
--node-type **m6g.2xlarge** \
--nodes 3 \
--node-volume-size 500 \
--region us-east-1

tldr.yaml

```
containers:
#- image: store/intersystems/iris-community:2020.4.0.524.0
- image: store/intersystems/irishealth-community-arm64:2020.4.0.524.0
name: iris
```

または、単に以下を使用します。

```
kubectl apply -f https://github.com/antonum/ha-iris-k8s/raw/main/tldr-iris4h-arm.yaml
```

以上です！ ARMプラットフォームで実行するIRIS Kubernetesクラスタが出来上がりました。

特典2 - バックアップと復元

本番環境グレードのアーキテクチャでよく見過ごされがちな部分に、データベースのバックアップを作成して必要なときに素早く復元するか複製する機能があります。

Kubernetesでは、一般的にPersistent Volume
Snapshots（永続ボリュームスナップショット）を使用してこれを行います。

まず、必要なすべてのk8sコンポーネントをインストールする必要があります。

```
#Install CSI Snapshotter and CRDs

kubectl apply -n kube-system -f https://raw.githubusercontent.com/kubernetes-csi/exte
rnal-snapshotter/master/client/config/crd/snapshot.storage.k8s.io_volumesnapshotconte
nts.yaml
kubectl apply -n kube-system -f https://github.com/kubernetes-csi/external-snapshotte
r/raw/master/client/config/crd/snapshot.storage.k8s.io_volumesnapshotclasses.yaml
kubectl apply -n kube-system -f https://raw.githubusercontent.com/kubernetes-csi/exte
rnal-
snapshotter/master/client/config/crd/snapshot.storage.k8s.io_volumesnapshots.yaml
kubectl apply -n kube-system -f https://raw.githubusercontent.com/kubernetes-csi/exte
rnal-snapshotter/master/deploy/kubernetes/snapshot-controller/setup-snapshot-
controller.yaml
kubectl apply -n kube-system -f https://raw.githubusercontent.com/kubernetes-csi/exte
rnal-snapshotter/master/deploy/kubernetes/snapshot-controller/rbac-snapshot-
controller.yaml
```

次に、LonghornのS3バケット資格情報を構成します（詳細な手順を参照）。

```
#Longhorn backup target s3 bucket and credentials longhorn would use to access that b
ucket
#See https://longhorn.io/docs/1.1.0/snapshots-and-backups/backup-and-restore/set-
backup-target/ for manual setup instructions
longhorn_s3_bucket=longhorn-backup-123xx #bucket name should be globally unique, unle
ss you want to reuse existing backups and credentials
longhorn_s3_region=us-east-1
longhorn_aws_key=AKIAVHCUNTEXAMPLE
longhorn_aws_secret=g2q2+5DVXk5p3AHIB5m/Tk6U6dXrEXAMPLE
```

以下のコマンドは、前の手順から環境変数を拾い、それを使ってLonghorn Backupを構成します。

```
#configure Longhorn backup target and credentials

cat &lt;&lt;EOF | kubectl apply -f -
apiVersion: longhorn.io/v1beta1
kind: Setting
metadata:
 name: backup-target
 namespace: longhorn-system
value: "s3://$longhorn_s3_bucket@$longhorn_s3_region/" # backup target here
---
apiVersion: v1
kind: Secret
```

```
metadata:
 name: "aws-secret"
 namespace: "longhorn-system"
 labels:
data:
 # echo -n '&lt;secret>' | base64
 AWS_ACCESS_KEY_ID: $(echo -n $longhorn_aws_key | base64)
 AWS_SECRET_ACCESS_KEY: $(echo -n $longhorn_aws_secret | base64)
---
apiVersion: longhorn.io/v1beta1
 kind: Setting
metadata:
 name: backup-target-credential-secret
 namespace: longhorn-system
value: "aws-secret" # backup secret name here
EOF
```

たくさんの作業に見えるかもしれませんが、基本的にLonghornに対し、指定された資格情報で特定のS3バケットを使用し、バックアップのコンテンツを保存するように指示しています。

以上です！ Longhorn UIに移動すると、バックアップを作成して復元などを行えるようになっています。

Longhorn UIに接続する方法を簡単におさらいしましょう。

```
kubectl get pods -n longhorn-system
# note the full pod name for 'longhorn-ui-...' pod
kubectl port-forward longhorn-ui-df95bdf85-469sz 9000:8000 -n longhorn-system
```

これによって、Longhorn UIへのトラフィックはhttp://localhost:9000に転送されるようになります。

プログラムによるバックアップ/復元

Longhorn
UIを介して行うバックアップと復元は、最初のステップとしては十分かもしれませんが、もう一歩先に進み、k8s Snapshot APIを使用して、プログラムでバックアップと復元を実行してみましょう。

まず、スナップショットそのものが必要です。 iris-volume-snapshot.yaml

```
apiVersion: snapshot.storage.k8s.io/v1beta1
kind: VolumeSnapshot
metadata:
  name: iris-longhorn-snapshot
spec:
  volumeSnapshotClassName: longhorn
  source:
    persistentVolumeClaimName: iris-pvc
```

このボリュームスナップショットは、IRISデプロイに使用するソースボリュームである 'iris-pvc'
を参照しています。 そのため、これを適用するだけですぐにバックアッププロセスが開始します。

IRIS書き込みデーモンの凍結と解凍をスナップショットの前後に実行することをお勧めします。

```
#Freeze Write Daemon
echo "Freezing IRIS Write Daemon"
kubectl exec -it -n $namespace $pod_name -- iris session iris -U%SYS "##Class(Backup.
```

```
General).ExternalFreeze()"
status=$?
if [[ $status -eq 5 ]]; then
 echo "IRIS WD IS FROZEN, Performing backup"
 kubectl apply -f backup/iris-volume-snapshot.yaml -n $namespace
elif [[ $status -eq 3 ]]; then
 echo "IRIS WD FREEZE FAILED"
fi
#Thaw Write Daemon
kubectl exec -it -n $namespace $pod_name -- iris session iris -U%SYS "##Class(Backup.
General).ExternalThaw()"
```

復元プロセスは非常に簡単です。
基本的には、新しいPVCを作成して、スナップショットをソースとして指定しています。

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: iris-pvc-restored
spec:
  storageClassName: longhorn
  dataSource:
    name: iris-longhorn-snapshot
    kind: VolumeSnapshot
    apiGroup: snapshot.storage.k8s.io
  accessModes:
    - ReadWriteOnce
  resources:
    requests:
      storage: 10Gi
```

次に、このPVCに基づいて新しいデプロイを作成するだけです。
これを順に行うこちらのGitHubリポジトリにあるテストスクリプトをご覧ください。

- まったく新しいIRISデプロイを作成します。
- IRISにデータを追加します。
- 書き込みデーモンを凍結し、スナップショットを取得して、書き込みデーモンを解凍します。
- そのスナップショットをベースに、IRISデプロイのクローンを作成します。
- すべてのデータが含まれていることを確認します。

この時点で、同一のIRISデプロイが2つ存在することになります。1つはもう片方のデプロイのclone-via-backupです。

どうぞお楽しみください！

#AWS #DevOps #クラウド #コンテナ化 #デプロイ #バックアップ #高可用性 #InterSystems IRIS #InterSystems IRIS for Health #Open Exchange
InterSystems Open Exchangeで関連アプリケーションを確認してください

ソースURL:https://jp.community.intersystems.com/post/amazon-eks%E3%81%A8iris%E3%80%82-%E9%AB%98%E5%8F%AF%E7%94%A8%E6%80%A7%E3%81%A8%E3%83%90%E3%83%83%E3%82%AF%E3%82%A2%E3%83%83%E3%83%97