ONLINE, AROUND-THE-WORLD OPEN SOURCE DATABASE CONFERENCE
PERCONA
LIVEONLINE
MAY 12 - 13th
2021

Database Backup and Import with Kanister

PERCONA LIVEONLINE

Hello!

Aaron Alpar

Member Technical Staff
Kasten by Veeam
aaron.alpar@veeam.com

Introduction

- Who is Kasten?
- Products
- Kanister
- Backup and import using Kanister

kasten by Veeam

Kasten by Veeam

Kasten by Veeam

- Kasten
- K10
  - Kubernetes data management
  - Backup and restore
  - Disaster recovery
  - Application Mobility

kasten by Veeam

Kasten Open Source

- Kubestr

  Volume characterization for Kubernetes

- Kanister

  Data-oriented backups for Kubernetes

kasten by Veeam

What you need

What you'll need

- Kubernetes cluster
- Experience creating and retrieving resources with kubectl
- Kanister Operator

```
$ helm repo add kanister https://charts.kanister.io
$ helm install --name myrelease --namespace kanister kanister/kanister-operator \
    --set image.tag=0.50.0
```

Kanister

Kanister

- Kanister Operator
  - Responsible for custom resources and state management
- Blueprints
  - How to Backup, Restore, Delete
- Profiles
  - Describe backup destination or restore sources
- ActionSets
  - Run an action to Backup, Restore, Delete

Backup a simple application

A simple example: Application

```
$ cat <<EOF | kubectl create -f -
  apiVersion: apps/v1
  kind: Deployment
  metadata:
    name: time-logger
    namespace: timelogger
    spec:
      selector:
        matchLabels:
          app: time-logger
      replicas: 1
      template:
        metadata:
          labels:
            app: time-logger
        spec:
          containers:
          - name: test-container
            image: containerlabs/aws-sdk
            command: ["sh", "-c"]
            args: ["while true; do for x in $(seq 1200); do date >> /var/log/time.log; sleep 1; done; truncate
/var/log/time.log; done"]
EOF
```

A simple example: Application

```
$ cat <<EOF | kubectl create -f -
  apiVersion: apps/v1
  kind: Deployment
  metadata:
    name: time-logger
    namespace: timelogger
    spec:
      selector:
        matchLabels:
          app: time-logger
      replicas: 1
      template:
        metadata:
          labels:
            app: time-logger
        spec:
          containers:
          - name: test-container
            image: containerlabs/aws-sdk
            command: ["sh", "-c"]
            args: ["while true; do for x in $(seq 1200); do date >> /var/log/time.log; sleep 1; done; truncate
/var/log/time.log; done"]
EOF
```

The Blueprint

- Tells Kanister how to backup the application
- Action
- Phases
- Commands (aka Args)

A simple example: Blueprint

```
$ cat <<EOF | kubectl create -f -
  apiVersion: cr.kanister.io/v1alpha1
  kind: Blueprint
  metadata:
    name: time-log-bp
    namespace: kanister
  actions:
    backup:
      type: Deployment
      phases:
      - func: KubeExec
        name: backupToLog
        args:
          namespace: "{{ .Deployment.Namespace }}”
          pod: "{{ index .Deployment.Pods 0 }}”
          container: test-container
          command:
          - sh
          - -c
          - cat /var/log/time.log
EOF
```

A simple example: Blueprint

```
$ cat <<EOF | kubectl create -f -
  apiVersion: cr.kanister.io/v1alpha1
  kind: Blueprint
  metadata:
    name: time-log-bp
    namespace: kanister
  actions:
    backup:
      type: Deployment
      phases:
      - func: KubeExec
        name: backupToLog
        args:
          namespace: "{{ .Deployment.Namespace }}”
          pod: "{{ index .Deployment.Pods 0 }}”
          container: test-container
          command:
          - sh
          - -c
          - cat /var/log/time.log
EOF
```

A simple example: Blueprint

```
$ cat <<EOF | kubectl create -f -
  apiVersion: cr.kanister.io/v1alpha1
  kind: Blueprint
  metadata:
    name: time-log-bp
    namespace: kanister
  actions:
    backup:
      type: Deployment
      phases:
      - func: KubeExec
        name: backupToLog
        args:
          namespace: "{{ .Deployment.Namespace }}”
          pod: "{{ index .Deployment.Pods 0 }}”
          container: test-container
          command:
          - sh
          - -c
          - cat /var/log/time.log
EOF
```

A simple example: Blueprint

```
$ cat <<EOF | kubectl create -f -
  apiVersion: cr.kanister.io/v1alpha1
  kind: Blueprint
  metadata:
    name: time-log-bp
    namespace: kanister
  actions:
    backup:
      type: Deployment
      phases:
      - func: KubeExec
        name: backupToLog
        args:
          namespace: "{{ .Deployment.Namespace }}”
          pod: "{{ index .Deployment.Pods 0 }}”
          container: test-container
          command:
          - sh
          - -c
          - cat /var/log/time.log
EOF
```

A simple example: Blueprint

```
$ cat <<EOF | kubectl create -f -
  apiVersion: cr.kanister.io/v1alpha1
  kind: Blueprint
  metadata:
    name: time-log-bp
    namespace: kanister
  actions:
    backup:
      type: Deployment
      phases:
      - func: KubeExec
        name: backupToLog
        args:
          namespace: "{{ .Deployment.Namespace }}”
          pod: "{{ index .Deployment.Pods 0 }}”
          container: test-container
          command:
          - sh
          - -c
          - cat /var/log/time.log
EOF
```

A simple example: Blueprint

```
$ cat <<EOF | kubectl create -f -
  apiVersion: cr.kanister.io/v1alpha1
  kind: Blueprint
  metadata:
    name: time-log-bp
    namespace: kanister
  actions:
    backup:
      type: Deployment
      phases:
      - func: KubeExec
        name: backupToLog
        args:
          namespace: "{{ .Deployment.Namespace }}”
          pod: "{{ index .Deployment.Pods 0 }}”
          container: test-container
          command:
          - sh
          - -c
          - cat /var/log/time.log
EOF
```

The ActionSet

- Tells a Blueprint to run an Action
- Tells a Blueprint which workload to backup
  - A workload is Deployment or StatefulSet
- Tells a Blueprint which Profile to use to send backups to

A simple example: ActionSet

```
$ cat <<EOF | kubectl create -f -
  apiVersion: cr.kanister.io/v1alpha1
  kind: ActionSet
  metadata:
    generateName: s3backup-
    namespace: kanister
  spec:
    actions:
    - name: backup
      blueprint: time-log-bp
      object:
        kind: Deployment
        name: time-logger
        namespace: timelogger
EOF
```

A simple example: ActionSet

```
$ cat <<EOF | kubectl create -f -
  apiVersion: cr.kanister.io/v1alpha1
  kind: ActionSet
  metadata:
    generateName: s3backup-
    namespace: kanister
  spec:
    actions:
    - name: backup
      blueprint: time-log-bp
      object:
        kind: Deployment
        name: time-logger
        namespace: timelogger
EOF
```

A simple example

```
$ kubectl get actionsets -n kanister

NAME             AGE
s3backup-cv7z8   3m24s

$ kubectl get actionsets -n kanister s3backup-cv7z8 -o yaml
apiVersion: cr.kanister.io/v1alpha1
kind: ActionSet
metadata:
  name: s3backup-cv7z8
  namespace: kanister
  …
spec:
  actions:
  - blueprint: time-log-bp
    name: backup
    …
status:
  actions:
  - artifacts: null
    blueprint: time-log-bp
    name: backup
    …
    phases:
    - name: backupToS3
      state: complete
      …
  state: complete
  …
```

Kanister tools

Kanister tools

- Kanister has tools to manage components
- Profiles
  - Componentized backup source and destinations
  - Separate details of storage from how the backup is performed
- Kanctl
  - Used at the command line to create profiles and actionsets
- Kando
  - Used within containers to export backups and integrate with K10

Creating an actionset: kubectl

```
$ cat <<EOF | kubectl create -f -
  apiVersion: cr.kanister.io/v1alpha1
  kind: ActionSet
  metadata:
    generateName: s3backup-
    namespace: kanister
  spec:
    actions:
    - name: backup
      blueprint: time-log-bp
      object:
        kind: Deployment
        name: time-logger
        namespace: timelogger
EOF
```

Creating an actionset using kanctl

```
$ kanctl create actionset \
  --action backup \
  --namespace kanister \
  --blueprint time-log-s3-bp \
  --deployment timelogger/time-logger
```

Creating a profile using kanctl

```
$ kanctl create profile s3compliant \
    --bucket my-bucket \
    --access-key ${AWS_ACCESS_KEY_ID} \
    --secret-key ${AWS_SECRET_ACCESS_KEY} \
    --region us-west-1 \
    --namespace kanister
```

Kando: exporting using a Profile

```
$ cat <<EOF | kubectl create -f -
apiVersion: cr.kanister.io/v1alpha1
kind: Blueprint
metadata:
  name: time-log-s3-bp
  namespace: kanister
actions:
  backup:
    type: Deployment
    phases:
    - func: KubeExec
      name: backupToS3
      args:
        namespace: "{{ .Deployment.Namespace }}”
        pod: "{{ index .Deployment.Pods 0 }}”
        container: test-container
        command:
        - sh
        - -c
        - |
          cat /var/log/time.log | kando location push --profile '{{ toJson .Profile }}' --path "timelogger" -
          kando output backupLocation "timelogger”
EOF
```

Kando: Previous example Blueprint

```
$ cat <<EOF | kubectl create -f -
apiVersion: cr.kanister.io/v1alpha1
kind: Blueprint
metadata:
  name: time-log-bp
  namespace: kanister
actions:
  backup:
    type: Deployment
    phases:
    - func: KubeExec
      name: backupToLog
      args:
        namespace: "{{ .Deployment.Namespace }}”
        pod: "{{ index .Deployment.Pods 0 }}”
        container: test-container
        command:
        - sh
        - -c
        - cat /var/log/time.log
EOF
```

Backup PostgreSQL

A real example: postgresql

- Blueprint support export and import of postgresql database
- Blueprint can be modified to suit backup needs
- https://github.com/kanisterio/kanister

A PostgreSQL example

```
$ cat <<EOF | kubectl create -f -
apiVersion: cr.kanister.io/v1alpha1
kind: Blueprint
metadata:
  name: postgres-bp
  namespace: kanister
actions:
  backup:
    kind: StatefulSet
    outputArtifacts:
      cloudObject:
        keyValue:
          backupLocation: "{{ .Phases.pgDump.Output.backupLocation }}"
    phases:
    - func: KubeTask
      name: pgDump
      objects:
        pgSecret:
          kind: Secret
          name: '{{ index .Object.metadata.labels "app.kubernetes.io/instance" }}-postgresql'
          namespace: '{{ .StatefulSet.Namespace }}'
      args:
        image: ghcr.io/kanisterio/postgres-kanister-tools:0.50.0
        namespace: '{{ .StatefulSet.Namespace }}'
        command:
        - bash
        - -c
        - |
          set -o pipefail
          set -o errexit
          export PGHOST='{{ index .Object.metadata.labels "app.kubernetes.io/instance" }}-postgresql.{{ .StatefulSet.Namespace }}.svc.cluster.local'
          export PGUSER='postgres'
          export PGPASSWORD='{{ index .Phases.pgDump.Secrets.pgSecret.Data "postgresql-password" | toString }}'
          BACKUP_LOCATION=pg_backups/{{ .StatefulSet.Namespace }}/{{ .StatefulSet.Name }}/{{ toDate "2006-01-02T15:04:05.999999999Z07:00" .Time | date "2006-01-02T15:04:05Z07:00" }}/backup.sql.gz
          pg_dumpall --clean -U \$PGUSER | gzip -c | kando location push --profile '{{ toJson .Profile }}' --path "\${BACKUP_LOCATION}" -
          kando output backupLocation "\${BACKUP_LOCATION}"
  restore:
    kind: StatefulSet
    inputArtifactNames:
    - cloudObject
    phases:
    - func: KubeTask
      name: pgRestore
      objects:
        pgSecret:
          kind: Secret
          name: '{{ index .Object.metadata.labels "app.kubernetes.io/instance" }}-postgresql'
          namespace: '{{ .StatefulSet.Namespace }}'
      args:
        image: ghcr.io/kanisterio/postgres-kanister-tools:0.50.0
        namespace: '{{ .StatefulSet.Namespace }}'
        command:
        - bash
        - -c
        - |
          set -o pipefail
          set -o errexit
          export PGHOST='{{ index .Object.metadata.labels "app.kubernetes.io/instance" }}-postgresql.{{ .StatefulSet.Namespace }}.svc.cluster.local'
          export PGUSER='postgres'
          export PGPASSWORD='{{ index .Phases.pgRestore.Secrets.pgSecret.Data "postgresql-password" | toString }}'
          BACKUP_LOCATION={{ .ArtifactsIn.cloudObject.KeyValue.backupLocation }}
          kando location pull --profile '{{ toJson .Profile }}' --path "\${BACKUP_LOCATION}" - | gunzip -c -f | psql -q -U "\${PGUSER}"
EOF
```

A PostgreSQL example

```
  ...
  backup:
    kind: StatefulSet
    outputArtifacts:
      cloudObject:
        keyValue:
          backupLocation: "{{ .Phases.pgDump.Output.backupLocation }}"
    phases:
    - func: KubeTask
      name: pgDump
      objects:
        pgSecret: ...
          name: '{{ index .Object.metadata.labels "app.kubernetes.io/instance" }}-postgresql'
          namespace: '{{ .StatefulSet.Namespace }}'
      args:
        image: ghcr.io/kanisterio/postgres-kanister-tools:0.50.0
        namespace: '{{ .StatefulSet.Namespace }}'
        command:
        - bash ...
          export PGHOST='{{ index .Object.metadata.labels "app.kubernetes.io/instance" }}-postgresql.{{ .StatefulSet.Namespace }}.svc.cluster.local'
          export PGUSER='postgres'
          export PGPASSWORD='{{ index .Phases.pgDump.Secrets.pgSecret.Data "postgresql-password" | toString }}'
          BACKUP_LOCATION=pg_backups/{{ .StatefulSet.Namespace }}/{{ .StatefulSet.Name }}/{{ toDate "2006-01-02T15:04:05.999999999Z07:00" .Time | date
"2006-01-02T15:04:05Z07:00" }}/backup.sql.gz
          pg_dumpall --clean -U \$PGUSER | gzip -c | kando location push --profile '{{ toJson .Profile }}' --path "\${BACKUP_LOCATION}" -
          kando output backupLocation "\${BACKUP_LOCATION}"
```

A PostgreSQL example

```
...
backup:
  kind: StatefulSet
  outputArtifacts:
    cloudObject:
      keyValue:
        backupLocation: "{{ .Phases.pgDump.Output.backupLocation }}"
  phases:
  - func: KubeTask
    name: pgDump
    objects:
      pgSecret: ...
        name: '{{ index .Object.metadata.labels "app.kubernetes.io/instance" }}-postgresql'
        namespace: '{{ .StatefulSet.Namespace }}'
    args:
      image: ghcr.io/kanisterio/postgres-kanister-tools:0.50.0
      namespace: '{{ .StatefulSet.Namespace }}'
      command:
      - bash ...
        export PGHOST='{{ index .Object.metadata.labels "app.kubernetes.io/instance" }}-postgresql.{{ .StatefulSet.Namespace }}.svc.cluster.local'
        export PGUSER='postgres'
        export PGPASSWORD='{{ index .Phases.pgDump.Secrets.pgSecret.Data "postgresql-password" | toString }}'
        BACKUP_LOCATION=pg_backups/{{ .StatefulSet.Namespace }}/{{ .StatefulSet.Name }}/{{ toDate "2006-01-02T15:04:05.999999999Z07:00" .Time | date
"2006-01-02T15:04:05Z07:00" }}/backup.sql.gz
        pg_dumpall --clean -U \$PGUSER | gzip -c | kando location push --profile '{{ toJson .Profile }}' --path "\${BACKUP_LOCATION}" -
        kando output backupLocation "\${BACKUP_LOCATION}"
```

A PostgreSQL example

```
  ...
  backup:
    kind: StatefulSet
    outputArtifacts:
      cloudObject:
        keyValue:
          backupLocation: "{{ .Phases.pgDump.Output.backupLocation }}"
    phases:
    - func: KubeTask
      name: pgDump
      objects:
        pgSecret: ...
          name: '{{ index .Object.metadata.labels "app.kubernetes.io/instance" }}-postgresql'
          namespace: '{{ .StatefulSet.Namespace }}'
      args:
        image: ghcr.io/kanisterio/postgres-kanister-tools:0.50.0
        namespace: '{{ .StatefulSet.Namespace }}'
        command:
        - bash ...
          export PGHOST='{{ index .Object.metadata.labels "app.kubernetes.io/instance" }}-postgresql.{{ .StatefulSet.Namespace }}.svc.cluster.local'
          export PGUSER='postgres'
          export PGPASSWORD='{{ index .Phases.pgDump.Secrets.pgSecret.Data "postgresql-password" | toString }}'
          BACKUP_LOCATION=pg_backups/{{ .StatefulSet.Namespace }}/{{ .StatefulSet.Name }}/{{ toDate "2006-01-02T15:04:05.999999999Z07:00" .Time | date
"2006-01-02T15:04:05Z07:00" }}/backup.sql.gz
          pg_dumpall --clean -U \$PGUSER | gzip -c | kando location push --profile '{{ toJson .Profile }}' --path "\${BACKUP_LOCATION}" -
          kando output backupLocation "\${BACKUP_LOCATION}"
```

A PostgreSQL example

```
  ...
  backup:
    kind: StatefulSet
    outputArtifacts:
      cloudObject:
        keyValue:
          backupLocation: "{{ .Phases.pgDump.Output.backupLocation }}"
    phases:
    - func: KubeTask
      name: pgDump
      objects:
        pgSecret: ...
          name: '{{ index .Object.metadata.labels "app.kubernetes.io/instance" }}-postgresql'
          namespace: '{{ .StatefulSet.Namespace }}'
      args:
        image: ghcr.io/kanisterio/postgres-kanister-tools:0.50.0
        namespace: '{{ .StatefulSet.Namespace }}'
        command:
        - bash ...
          export PGHOST='{{ index .Object.metadata.labels "app.kubernetes.io/instance" }}-postgresql.{{ .StatefulSet.Namespace }}.svc.cluster.local'
          export PGUSER='postgres'
          export PGPASSWORD='{{ index .Phases.pgDump.Secrets.pgSecret.Data "postgresql-password" | toString }}'
          BACKUP_LOCATION=pg_backups/{{ .StatefulSet.Namespace }}/{{ .StatefulSet.Name }}/{{ toDate "2006-01-02T15:04:05.999999999Z07:00" .Time | date
"2006-01-02T15:04:05Z07:00" }}/backup.sql.gz
          pg_dumpall --clean -U \$PGUSER | gzip -c | kando location push --profile '{{ toJson .Profile }}' --path "\${BACKUP_LOCATION}" -
          kando output backupLocation "\${BACKUP_LOCATION}"
```

A PostgreSQL example

```
  ...
  backup:
    kind: StatefulSet
    outputArtifacts:
      cloudObject:
        keyValue:
          backupLocation: "{{ .Phases.pgDump.Output.backupLocation }}"
    phases:
    - func: KubeTask
      name: pgDump
      objects:
        pgSecret: ...
          name: '{{ index .Object.metadata.labels "app.kubernetes.io/instance" }}-postgresql'
          namespace: '{{ .StatefulSet.Namespace }}'
      args:
        image: ghcr.io/kanisterio/postgres-kanister-tools:0.50.0
        namespace: '{{ .StatefulSet.Namespace }}'
        command:
        - bash ...
          export PGHOST='{{ index .Object.metadata.labels "app.kubernetes.io/instance" }}-postgresql.{{ .StatefulSet.Namespace }}.svc.cluster.local'
          export PGUSER='postgres'
          export PGPASSWORD='{{ index .Phases.pgDump.Secrets.pgSecret.Data "postgresql-password" | toString }}'
          BACKUP_LOCATION=pg_backups/{{ .StatefulSet.Namespace }}/{{ .StatefulSet.Name }}/{{ toDate "2006-01-02T15:04:05.999999999Z07:00" .Time | date
"2006-01-02T15:04:05Z07:00" }}/backup.sql.gz
          pg_dumpall --clean -U \$PGUSER | gzip -c | kando location push --profile '{{ toJson .Profile }}' --path "\${BACKUP_LOCATION}" -
          kando output backupLocation "\${BACKUP_LOCATION}"
```

A PostgreSQL example

```
  ...
  backup:
    kind: StatefulSet
    outputArtifacts:
      cloudObject:
        keyValue:
          backupLocation: "{{ .Phases.pgDump.Output.backupLocation }}"
    phases:
    - func: KubeTask
      name: pgDump
      objects:
        pgSecret: ...
          name: '{{ index .Object.metadata.labels "app.kubernetes.io/instance" }}-postgresql'
          namespace: '{{ .StatefulSet.Namespace }}'
      args:
        image: ghcr.io/kanisterio/postgres-kanister-tools:0.50.0
        namespace: '{{ .StatefulSet.Namespace }}'
        command:
        - bash ...
          export PGHOST='{{ index .Object.metadata.labels "app.kubernetes.io/instance" }}-postgresql.{{ .StatefulSet.Namespace }}.svc.cluster.local'
          export PGUSER='postgres'
          export PGPASSWORD='{{ index .Phases.pgDump.Secrets.pgSecret.Data "postgresql-password" | toString }}'
          BACKUP_LOCATION=pg_backups/{{ .StatefulSet.Namespace }}/{{ .StatefulSet.Name }}/{{ toDate "2006-01-02T15:04:05.999999999Z07:00" .Time | date
"2006-01-02T15:04:05Z07:00" }}/backup.sql.gz
          pg_dumpall --clean -U \$PGUSER | gzip -c | kando location push --profile '{{ toJson .Profile }}' --path "\${BACKUP_LOCATION}" -
          kando output backupLocation "\${BACKUP_LOCATION}"
```

References

- https://kanister.io
- https://kubestr.io
- https://www.kasten.io
- https://github.com/kanisterio/kanister
- https://github.com/kanisterio/kanister/tree/master/examples/stable/postgresql

Thanks!

Any questions?

You can find me at:

- aaron.alpar@veeam.com

THANK YOU !

PERCONA
LIVEONLINE
MAY 12 - 13th
2021