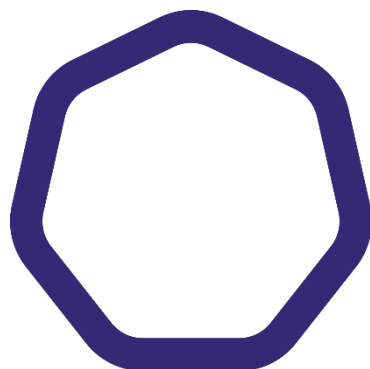


Министерство цифрового развития, связи и массовых коммуникаций
Российской Федерации
Ордена Трудового Красного Знамени федеральное государственное
бюджетное образовательное учреждение высшего образования
«Московский технический университет связи и информатики»
(МТУСИ)



Кафедра «Сетевые информационные технологии и сервисы»

Лабораторная работа №4
на тему «Согласованность приложений»

Выполнил:
Студенты 1 курса
Группы М092401(75)
Цыганков Р.О.
Проверил:
к.т.н., Шалагинов А. В.

Москва 2024 г

Цель работы: изучение принципов работы резервного копирования и восстановления собственных приложений Kubernetes с использованием Kasten K10 и Kanister, проекта Kasten с открытым исходным кодом, управление данными на уровне приложений.

Выполнение работы:

Для выполнения лабораторных работ необходимо:

1. Зайти на сайт: <https://learning.kasten.io/profile/>
2. Пройти процесс регистрации.
3. Перейти в свой профиль.
4. В разделе «Больше лабораторий для изучения» выбрать лабораторную работу «Создание резервной копии вашего приложения Kubernetes».
5. Нажать на кнопку «Start Lab».

Данный курс состоит из двух частей:

- Теоретический блок, на основании которого созданы вопросы из теста.
- Практический блок.

Теоретический блок:

Вопрос 1:

☒ MySQL

☒ MongoDB

☒ PostgreSQL

☒ Cassandra

2 ATTEMPTS

Incorrect solution 2/2

Not enough answers provided

Incorrect solution 1/2

Not enough answers provided

Kasten K10 Intro

Which of the following open source community database applications does Kasten K10 support? (Select one or more responses)

Correct!
Loading your next challenge...

Вопрос 2:

☒ BackupAction

☒ RestoreAction

☒ ExportAction

☐ DelayAction

Multiple answers can be checked as correct.

Kasten K10 Concepts - Actions

What are the action types available in K10? (Select one or more responses)

Correct!
Loading your next challenge...

Вопрос 3:

☐ Allow you to initiate data management operations

☒ Allow you to manage application protection and migration at scale

☐ Allow you to capture information about the state of the system

☐ Allow you to perform operations on profiles

Multiple answers can be checked as correct.

Kasten K10 Concepts - Policies

What do policies allow you to do?

Correct!
Loading your next challenge...

Вопрос 4:

☒ Google Cloud Storage

☐ Amazon S3

☒ Azure Storage

☒ NFS File Share

Multiple answers can be checked as correct.

Kasten K10 Concepts - Profiles

What are the storage providers supported by K10?
(Select one or more responses)

Correct!
Loading your next challenge...

Вопрос 5:

☒ Backup

☐ Delete

☒ Restore

☐ Run

Multiple answers can be checked as correct.

Introducing Kanister Enabled Applications in Kasten K10

What are the must-have actions to define a Blueprint?

Correct!
Loading your next challenge...

Вопрос 6:

☐ Custom Resource Definition

☒ Application Blueprint

☐ YAML Configuration

☐ Volume Snapshots

Multiple answers can be checked as correct.

Obtaining an Application-Consistent Backup with the Quiescing function

Where do you define the Quiescing function?

Correct!
Loading your next challenge...

Практический блок

Установка оснастки Kasten K10:

```
root@kubernetes:~# helm install k10 kasten/k10 --namespace=kasten-io --create-namespace
NAME: k10
LAST DEPLOYED: Mon Apr 18 16:52:41 2022
NAMESPACE: kasten-io
STATUS: deployed
REVISION: 1
TEST SUITE: None
NOTES:
Thank you for installing Kasten's K10 Data Management Platform!
Documentation can be found at https://docs.kasten.io/.
How to access the K10 Dashboard:

The K10 dashboard is not exposed externally. To establish a connection to it use the following 'kubectl' command:

'kubectl --namespace kasten-io port-forward service/gateway 8080:8080'

The Kasten dashboard will be available at: 'http://127.0.0.1:8080/k10/#/'
root@kubernetes:~# watch -n 2 "kubectl -n kasten-io get pods"
root@kubernetes:~# kubectl annotate volumesnapshotclass csi-hostpath-snapclass k10.kasten.io/is-snapshot-class=true
volumesnapshotclass.snapshot.storage.k8s.io/csi-hostpath-snapclass annotated
root@kubernetes:~# cat > k10-nodeport-svc.yaml << EOF
> apiVersion: v1
> kind: Service
> metadata:
>   name: gateway-nodeport
>   namespace: kasten-io
> spec:
>   selector:
>     service: gateway
>   ports:
>     - name: http
>       port: 8080
>       nodePort: 32000
>       type: NodePort
> EOF
root@kubernetes:~# kubectl apply -f k10-nodeport-svc.yaml
```

```
metadata:
  name: gateway-nodeport
  namespace: kasten-io
spec:
  selector:
    service: gateway
  ports:
    - name: http
      port: 8080
      nodePort: 32000
      type: NodePort
EOF
```

Now, let's create the actual NodePort Service

```
kubectl apply -f k10-nodeport-svc.yaml
```

View the K10 Dashboard

Once completed, you should be able to view the K10 dashboard in the other tab on the left. We recommend taking the K10 dashboard tour at this point.

Correct!
Loading your next challenge...

Настройка расположения резервной копии хранилища объектов (MinIO):

Settings

Locations

Manage cloud location settings

Infrastructure

Settings for infrastructure platforms

K10 Disaster Recovery

Enable backup/recovery of K10

Licenses

View K10 product licenses

User Roles

Manage User Access Privileges

Support

Cluster and contact info

Dashboard

Location Profiles

Create profiles that **define cloud credentials and bucket locations** needed to move data in and out of the cluster. You'll select from these profiles when creating policies or exporting a restore point.

[New Profile](#)

LOCATION PROFILE	revalidate	yaml	edit	delete
k10				
STATUS	CLOUD PROVIDER	BUCKET NAME		
Valid	Generic S3	k10		
ENDPOINT	http://minio.minio.svc.cluster.local:9000			

The command will install our object storage.

Configure a Location Profile

Next, we are going to create a Location Profile using the object storage system we configured in the previous step.

- Click on **K10 Dashboard** tab in the upper left corner
- Click **Settings**
- From the left click **Locations**
- Click **New Profile**
- Enter the profile name **k10**
- Select **S3 Compatible**
- Fill out the fields with the following key information:
 - S3 Access Key: **AKIAIOSFODNN7EXAMPLE**
 - S3 Secret Key: **wJalrXUtnFEMI/K7MDENG/bPxrFiCYEXAMPLE**
 - Endpoint: **http://minio.minio.svc.cluster.local:9000**
 - Bucket Name: **k10**

Click **Check** when the profile is created successfully!

Correct!
Loading your next challenge...

Установим MySQL и создадим демонстрационную базу данных:

```

Services:
echo Primary: mysql.mysql.svc.cluster.local:3306

Execute the following to get the administrator credentials:
echo Username: root
MYSQL_ROOT_PASSWORD=$(kubectl get secret --namespace mysql mysql -o jsonpath="{.data.mysql-root-password}" | base64 --decode)

To connect to your database:
1. Run a pod that you can use as a client:
    kubectl run mysql-client --rm --tty -i --restart='Never' --image docker.io/bitnami/mysql:8.0.28-debian-10-r63 --namespace mysql --command -- bash
2. To connect to primary service (read/write):
    mysql -h mysql.mysql.svc.cluster.local -uroot -p"$MYSQL_ROOT_PASSWORD"

To upgrade this helm chart:
1. Obtain the password as described on the 'Administrator credentials' section and set the 'root-password' parameter as shown below:
    ROOT_PASSWORD=$(kubectl get secret --namespace mysql mysql -o jsonpath="{.data.mysql-root-password}" | base64 --decode)
    helm upgrade --namespace mysql mysql bitnami/mysql --set auth.rootPassword=$ROOT_PASSWORD
root@kubernetes:~# watch -n 2 'kubectl -n mysql get pods'
root@kubernetes:~# MYSQL_ROOT_PASSWORD=$(kubectl get secret --namespace mysql mysql -o jsonpath="{.data.mysql-root-password}" | base64 --decode; echo)
root@kubernetes:~# kubectl exec -it --namespace=mysql $(kubectl --namespace=mysql get pods -o jsonpath="{.items[0].metadata.name}") \
root@kubernetes:~# kubectl exec -it --namespace=mysql $(kubectl --namespace=mysql get pods -o jsonpath="{.items[0].metadata.name}") \
error: you must specify at least one command for the container
root@kubernetes:~# MYSQL_ROOT_PASSWORD=$(kubectl get secret --namespace mysql mysql -o jsonpath="{.data.mysql-root-password}" | base64 --decode; echo)
root@kubernetes:~# kubectl exec -it --namespace=mysql $(kubectl --namespace=mysql get pods -o jsonpath="{.items[0].metadata.name}") \
> -- mysql -u root --password=$MYSQL_ROOT_PASSWORD -e "CREATE DATABASE k10demo"
mysql: [warning] Using a password on the command line interface can be insecure.
root@kubernetes:~#

```

Install MySQL and Create a Demo Database

To experiment with backup and recovery of a cloud-native application, we will install MySQL and create a database in this step.

First, install MySQL using the following commands:

```
kubectl create namespace mysql
helm install mysql bitnami/mysql --namespace mysql
```

To ensure that MySQL is running, check the pod status to make sure they are all in the Running and Ready 1/1 state:

```
watch -n 2 'kubectl -n mysql get pods'
```

Once all pods have a Running and Ready 1/1 status, hit CTRL + C to exit watch and then run the following commands to create a local database.

Correct!
Loading your next challenge...

Выполнение логического резервного копирования:

```

root@kubernetes:~# kubectl --namespace kasten-io apply -f \
> https://raw.githubusercontent.com/kanisterio/kanister/0.69.0/examples/stable/mysql/blueprint-v2/mysql-blueprint.yaml
blueprint-cr.kanister.io/mysql-blueprint created
root@kubernetes:~# kubectl --namespace mysql annotate statefulset/mysql \
> kanister.kasten.io/blueprint=mysql-blueprint
statefulset.apps/mysql annotated
root@kubernetes:~#

```

Perform Logical Backup

Logical MySQL Backup

The following commands will install the MySQL Blueprint in the K10 namespace and add an annotation on the MySQL Deployment to instruct K10 to use the Blueprint when performing operations on this MySQL instance.

```
kubectl --namespace kasten-io apply -f \
https://raw.githubusercontent.com/kanisterio/kanister/0.69.0/examples/stable/mysql/blueprint-v2/mysql-blueprint.yaml
```

```
kubectl --namespace mysql annotate statefulset/mysql \
kanister.kasten.io/blueprint=mysql-blueprint
```

Finally, use K10 to backup and restore the application.

Correct!
Loading your next challenge...

Политики резервного копирования Kasten K10:

The screenshot shows the Kasten K10 dashboard. At the top, there's a navigation bar with 'Docs', 'Settings', and a user profile 'k10-admin'. The main section is titled 'Policy Run' and shows a successful run of 'backup-mysql' with a duration of 1 min, 25 secs. Below this, there's a table of 'Actions (1)' with columns for Backup, Applications, Protected Object, Originating Policy, Artifacts, and Start. The table shows one action: 'Backup' (scheduled-dups7) for the 'mysql' application, with artifacts 'kanister' and 'spec', and a start time of 'Today, 8:08pm'.

Kasten K10 Backup Policies

We will now create a policy to backup MySQL to the previously configured object storage location.

From the main K10 dashboard, click on the Policies card. There should be no policies visible at this point.

Click **Create New Policy** and:

- Give the policy the name: backup-mysql
- Select Snapshot for the Action
- Select Hourly for the Action Frequency
- Leave the Snapshot Retention selection as-is
- Select By Name for Select Applications and then, from the dropdown, select mysql
- Leave all other settings as-is and select **Create Policy**

Running the Policy

The above policies will only run at the scheduled time (by default, at the top of the hour). To run the policies manually for the first time, click on **Run Once** on the Policies page. Confirm by clicking **Run**

Correct!
Loading your next challenge...

Восстановим приложение:

```
root@kubernetes:~# MYSQL_ROOT_PASSWORD=$(kubectl get secret --namespace mysql mysql -o jsonpath="{.data.mysql-root-password}" | base64 --decode; echo)
root@kubernetes:~# kubectl exec -it --namespace=mysql $(kubectl --namespace=mysql get pods -o jsonpath="{.items[0].metadata.name}") -- mysql -u root --password=$MYSQL_ROOT_PASSWORD -e "DROP DATABASE k10demo"
mysql: [Warning] Using a password on the command line interface can be insecure.
root@kubernetes:~# kubectl exec -it --namespace=mysql $(kubectl --namespace=mysql get pods -o jsonpath="{.items[0].metadata.name}") -- mysql -u root --password=$MYSQL_ROOT_PASSWORD -e "SHOW DATABASES LIKE 'k10demo'"
mysql: [Warning] Using a password on the command line interface can be insecure.
root@kubernetes:~# MYSQL_ROOT_PASSWORD=$(kubectl get secret --namespace mysql mysql -o jsonpath="{.data.mysql-root-password}" | base64 --decode; echo)
root@kubernetes:~# kubectl exec -it --namespace=mysql $(kubectl --namespace=mysql get pods -o jsonpath="{.items[0].metadata.name}") -- mysql -u root --password=$MYSQL_ROOT_PASSWORD -e "SHOW DATABASES LIKE 'k10demo'"
mysql: [Warning] Using a password on the command line interface can be insecure.
root@kubernetes:~# kubectl exec -it --namespace=mysql $(kubectl --namespace=mysql get pods -o jsonpath="{.items[0].metadata.name}") -- mysql -u root --password=$MYSQL_ROOT_PASSWORD -e "SHOW DATABASES LIKE 'k10demo'"
mysql: [Warning] Using a password on the command line interface can be insecure.
+-----+
| Database (k10demo) |
+-----+
| k10demo             |
+-----+
root@kubernetes:~#
```

Restore the Application

Now that we have a MySQL backup, let's go simulate accidental data loss and then recover the system from that loss.

Causing Data Loss

For the purposes of this test drive, we will simply drop the k10demo database we had created earlier.

```
MYSQL_ROOT_PASSWORD=$(kubectl get secret --namespace=mysql $(kubectl exec -it --namespace=mysql $(kubectl get pods -o jsonpath="{.items[0].metadata.name}") -- mysql -u root --password=$MYSQL_ROOT_PASSWORD -e "DROP DATABASE k10demo"; echo)
```

Verify that the database has been deleted by running:

```
kubectl exec -it --namespace=mysql $(kubectl get pods -o jsonpath="{.items[0].metadata.name}") -- mysql -u root --password=$MYSQL_ROOT_PASSWORD -e "SHOW DATABASES LIKE 'k10demo'"
```

Recovering Data

Correct!
Loading your next challenge...

4-я работа выполнена:

```
root@kubernetes:~#
```

Congratulations

You have completed the lab successfully!

Additional Learning

To extend the learning experience, Kasten offers a variety of resources whitepapers, case studies, data sheets and E-Books.

Follow this link
<https://www.kasten.io/kubernetes/resources> to explore these learning materials.

Note that there are a variety of other Kubernetes learning resources you can access after you finish this lab. Please go to the "Resources" tab of the learning.kasten.io site to access this material.

Your final challenge is to exit the lab:

- First, press "Next"
- Then, provide feedback
- Finally, click on "Finish" in the bottom right hand corner

Correct!
You have successfully completed the track!

Заключение

В процессе лабораторной работы были изучены принципы работы резервного копирования и восстановления собственных приложений Kubernetes с использованием Kasten K10 и Kanister.