Persistent Volumes

Deploying WordPress and MySQL with Persistent Volumes

A `PersistentVolume` (PV) is a piece of storage in the cluster that has been manually provisioned by an administrator, or dynamically provisioned by Kubernetes using a StorageClass.

A `PersistentVolumeClaim` (PVC) is a request for storage by a user that can be fulfilled by a PV.

`PersistentVolumes` and `PersistentVolumeClaims` are independent from Pod lifecycles and preserve data through restarting, rescheduling, and even deleting Pods.

Using Persistent Disks with WordPress and MySQL

Install the Google Cloud SDK

```
# Create environment variable for correct distribution
export CLOUD_SDK_REPO="cloud-sdk-$(lsb_release -c -s)"

# Add the Cloud SDK distribution URI as a package source
echo "deb http://packages.cloud.google.com/apt $CLOUD_SDK_REPO main"
 | sudo tee -a /etc/apt/sources.list.d/google-cloud-sdk.list

# Import the Google Cloud Platform public key
curl https://packages.cloud.google.com/apt/doc/apt-key.gpg | sudo ap
t-key add -
```

```
# Update the package list and install the Cloud SDK
sudo apt-get update && sudo apt-get install google-cloud-sdk -y
```

```
gcloud init
```

Step 1: Create a Kubernetes Engine cluster

```
gcloud container clusters create admatic-cluster --num-nodes=3

WARNING: Accessing a Kubernetes Engine cluster requires the kubernet
es commandline
client [kubectl]. To install, run
  $ gcloud components install kubectl

WARNING: Currently node auto repairs are disabled by default. In the
 future this will change and they will be enabled by default. Use `-
-[no-]enable-autorepair` flag  to suppress this warning.
WARNING: Starting in Kubernetes v1.10, new clusters will no longer g
et compute-rw and storage-ro scopes added to what is specified in --
scopes (though the latter will remain included in the default --scop
es). To use these scopes, add them explicitly to --scopes. To use th
e new behavior, set container/new_scopes_behavior property (gcloud c
onfig set container/new_scopes_behavior true).
Creating cluster persistent-disk-tutorial...done.
Created [https://container.googleapis.com/v1/projects/espblufi-andro
id/zones/asia-south1-a/clusters/persistent-disk-tutorial].
To inspect the contents of your cluster, go to: https://console.clou
d.google.com/kubernetes/workload_/gcloud/asia-south1-a/persistent-di
sk-tutorial?project=espblufi-android
kubeconfig entry generated for persistent-disk-tutorial.

NAME                     LOCATION       MASTER_VERSION  MASTER_IP
    MACHINE_TYPE   NODE_VERSION  NUM_NODES  STATUS
persistent-disk-tutorial  asia-south1-a  1.8.10-gke.0    35.200.194.
80  n1-standard-1  1.8.10-gke.0  3          RUNNING
```

Step 2: Create your PersistentVolumes and PersistentVolumeClaims

```
cat << EOF > mysql-volumeclaim.yaml
```

```
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
  name: mysql-volumeclaim
spec:
  accessModes:
    - ReadWriteOnce
  resources:
    requests:
      storage: 200Gi
EOF

cat << EOF > wordpress-volumeclaim.yaml
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
  name: wordpress-volumeclaim
spec:
  accessModes:
    - ReadWriteOnce
  resources:
    requests:
      storage: 200Gi
EOF

kubectl apply -f mysql-volumeclaim.yaml
persistentvolumeclaim "mysql-volumeclaim" created

kubectl apply -f wordpress-volumeclaim.yaml
persistentvolumeclaim "wordpress-volumeclaim" created

kubectl get pvc
NAME                    STATUS    VOLUME
         CAPACITY   ACCESS MODES   STORAGECLASS   AGE
mysql-volumeclaim       Bound     pvc-f83cca9c-7089-11e8-a659-42010a
a000de   200Gi      RWO            standard       7s
wordpress-volumeclaim   Bound     pvc-d9cfbd43-7089-11e8-a659-42010a
a000de   200Gi      RWO            standard       58s
```

Step 3: Set up MySQL

```
kubectl create secret generic mysql --from-literal=password=Bigd4t4
secret "mysql" created
```

```
cat << EOF > mysql.yaml
apiVersion: extensions/v1beta1
kind: Deployment
metadata:
  name: mysql
  labels:
    app: mysql
spec:
  replicas: 1
  selector:
    matchLabels:
      app: mysql
  template:
    metadata:
      labels:
        app: mysql
    spec:
      containers:
        - image: mysql:5.6
          name: mysql
          env:
            - name: MYSQL_ROOT_PASSWORD
              valueFrom:
                secretKeyRef:
                  name: mysql
                  key: password
          ports:
            - containerPort: 3306
              name: mysql
          volumeMounts:
            - name: mysql-persistent-storage
              mountPath: /var/lib/mysql
      volumes:
        - name: mysql-persistent-storage
          persistentVolumeClaim:
            claimName: mysql-volumeclaim
EOF

kubectl create -f mysql.yaml
deployment.extensions "mysql" created

kubectl get pod -l app=mysql -o wide
NAME                     READY     STATUS    RESTARTS   AGE       IP
```

```
          NODE
mysql-6f554fcf64-r4nl2   1/1       Running   0          3m        10
.44.1.5   gke-persistent-disk-tuto-default-pool-44ec5f7c-wfjk
```

Create MySQL service

```
cat << EOF > mysql-service.yaml
apiVersion: v1
kind: Service
metadata:
  name: mysql
  labels:
    app: mysql
spec:
  type: ClusterIP
  ports:
    - port: 3306
  selector:
    app: mysql
EOF

kubectl create -f mysql-service.yaml
service "mysql" created

kubectl get service mysql
NAME      TYPE        CLUSTER-IP     EXTERNAL-IP   PORT(S)    AGE
mysql     ClusterIP   10.47.254.83   <none>        3306/TCP   1s
```

Step 4: Set up WordPress

Deploy WordPress

```
cat << EOF > wordpress.yaml
apiVersion: extensions/v1beta1
kind: Deployment
metadata:
  name: wordpress
  labels:
    app: wordpress
spec:
  replicas: 1
  selector:
```

```
    matchLabels:
      app: wordpress
  template:
    metadata:
      labels:
        app: wordpress
    spec:
      containers:
        - image: wordpress
          name: wordpress
          env:
          - name: WORDPRESS_DB_HOST
            value: mysql:3306
          - name: WORDPRESS_DB_PASSWORD
            valueFrom:
              secretKeyRef:
                name: mysql
                key: password
          ports:
            - containerPort: 80
              name: wordpress
          volumeMounts:
            - name: wordpress-persistent-storage
              mountPath: /var/www/html
      volumes:
        - name: wordpress-persistent-storage
          persistentVolumeClaim:
            claimName: wordpress-volumeclaim
EOF

kubectl create -f wordpress.yaml
deployment.extensions "wordpress" created

kubectl get pod -l app=wordpress -o wide
NAME                         READY     STATUS    RESTARTS   AGE
  IP          NODE
wordpress-5f574c476b-hjk5p   1/1       Running   0          2m
  10.44.1.6   gke-persistent-disk-tuto-default-pool-44ec5f7c-wfjk
```

Expose WordPress Service

```
cat << EOF > wordpress-service.yaml
apiVersion: v1
```

```
kind: Service
metadata:
  labels:
    app: wordpress
  name: wordpress
spec:
  type: LoadBalancer
  ports:
    - port: 80
      targetPort: 80
      protocol: TCP
  selector:
    app: wordpress
EOF

kubectl create -f wordpress-service.yaml
service "wordpress" created

kubectl get svc -l app=wordpress
NAME        TYPE           CLUSTER-IP      EXTERNAL-IP      PORT(S)
       AGE
wordpress   LoadBalancer   10.47.242.197   35.200.156.216   80:32668
/TCP   3m
```

Step 5: Visit your new WordPress blog

After finding out the IP address of your blog, point your browser to this IP address and you will see the WordPress installation screen

- EXTERNAL-IP: 35.200.156.216

Once you complete the WordPress setup, point your browser to the IP address of the WordPress app again to visit your blog. Everything is working as expected.

Step 6: (Optional) Test data persistence on failure

With PersistentVolumes, your data lives outside the application container. When your container becomes unavailable and gets rescheduled onto another compute instance by Kubernetes, Kubernetes Engine will make the PersistentVolume available on the instance that started running the Pod.

```
kubectl get pods -o=wide
NAME                           READY     STATUS    RESTARTS   AGE
  IP           NODE
mysql-6f554fcf64-r4nl2         1/1       Running   0          7m
  10.44.1.5    gke-persistent-disk-tuto-default-pool-44ec5f7c-wfjk
wordpress-5f574c476b-hjk5p     1/1       Running   0          6m
  10.44.1.6    gke-persistent-disk-tuto-default-pool-44ec5f7c-wfjk

kubectl delete pod -l app=mysql
pod "mysql-6f554fcf64-r4nl2" deleted

kubectl get pods -o=wide
NAME                           READY     STATUS    RESTARTS   AGE
  IP           NODE
mysql-6f554fcf64-q7rdl         1/1       Running   0          7m
  10.44.2.7    gke-persistent-disk-tuto-default-pool-44ec5f7c-dd0v
wordpress-5f574c476b-hjk5p     1/1       Running   0          13m
  10.44.1.6    gke-persistent-disk-tuto-default-pool-44ec5f7c-wfjk
```

Visit your blog again to see that the website is functioning properly and the data is persisted even though you deleted your Pod and the Pod is scheduled to another instance in your cluster.

```
kubectl exec mysql-6f554fcf64-q7rdl -i -t -- bash
root@mysql-6f554fcf64-q7rdl:/# mysql -u root -p
Enter password: Bigd4t4
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 29
Server version: 5.6.40 MySQL Community Server (GPL)

Copyright (c) 2000, 2018, Oracle and/or its affiliates. All rights r
eserved.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input
statement.

mysql>
```

```
SHOW DATABASES;
+--------------------+
| Database           |
+--------------------+
| information_schema |
| #mysql50#lost+found |
| mysql              |
| performance_schema |
| wordpress          |
+--------------------+
5 rows in set (0.00 sec)

USE wordpress;
Reading table information for completion of table and column names
You can turn off this feature to get a quicker startup with -A

Database changed

SHOW TABLES;
+----------------------+
| Tables_in_wordpress  |
+----------------------+
| wp_commentmeta       |
| wp_comments          |
| wp_links             |
| wp_options           |
| wp_postmeta          |
| wp_posts             |
| wp_term_relationships |
| wp_term_taxonomy     |
| wp_termmeta          |
| wp_terms             |
| wp_usermeta          |
| wp_users             |
+----------------------+
12 rows in set (0.00 sec)

SELECT ID,post_author,post_date,post_name from wp_posts;
+----+-------------+---------------------+----------------+
| ID | post_author | post_date           | post_name      |
+----+-------------+---------------------+----------------+
|  1 |           1 | 2018-06-15 10:57:02 | hello-world    |
|  2 |           1 | 2018-06-15 10:57:02 | sample-page    |
|  3 |           1 | 2018-06-15 10:57:02 | privacy-policy |
```

```
|  4 |           1 | 2018-06-15 10:57:14 |                |
+----+-------------+---------------------+----------------+
4 rows in set (0.00 sec)
```