Deploy Kafka, Zookeeper, mids and Spark containers in GCP using Kubernetes

Initial Setup:

Visit kubernetes engine page and select the project (This will take several mins to start kubernetes engine)
https://console.cloud.google.com/projectselector/kubernetes?_ga=2.201928657.-1707404544.1516051830

Once kubernetes engine is ready:

```
gcloud components install kubectl
```

Prepare and push Images:

Check docker images that are locally stored and tag them to be pushed to google cloud:. We are interested in mids/spark/kafka/zookeeper

```
[kurapati-new:~] kurapati% docker images
REPOSITORY                        TAG          IMAGE ID        CREATED          SIZE
midsw205/base                     0.1.8        ee33b8fcc42c     4 weeks ago      2.68GB
midsw205/spark-python                0.0.5         0756ef14312d     5 weeks ago       3.17GB
confluentinc/cp-kafka              latest        8fa6da41c4ae      6 weeks ago      535MB
confluentinc/cp-zookeeper            latest        2acdb712eee3     6 weeks ago       535MB
```

NOTE : I trimmed the above output. You may have more entries, but we are interested in four images for now. Let's now tag them and push

```
[kurapati-new:~] kurapati% docker tag ee33b8fcc42c gcr.io/w205-1/mids
[kurapati-new:~] kurapati% docker tag 0756ef14312d gcr.io/w205-1/spark-python
[kurapati-new:~] kurapati% docker tag 8fa6da41c4ae gcr.io/w205-1/kafka
[kurapati-new:~] kurapati% docker tag 2acdb712eee3 gcr.io/w205-1/zookeeper
[kurapati-new:~] kurapati%
[kurapati-new:~] kurapati% gcloud docker -- push gcr.io/w205-1/mids
[kurapati-new:~] kurapati% gcloud docker -- push gcr.io/w205-1/spark-python
[kurapati-new:~] kurapati% gcloud docker -- push gcr.io/w205-1/kafka
[kurapati-new:~] kurapati% gcloud docker -- push gcr.io/w205-1/zookeeper
```

Create a cluster in Google Cloud:

I am creating a cluster with 5 nodes below (5 nodes is the maximum in n1-standard-1 flavor). You can change the zone based on where you live.

[kurapati-new:~] kurapati% gcloud container clusters create kafka --num-nodes=5 --zone northamerica-northeast1-a

Check if the cluster is created and the computes are operational:

```
[kurapati-new:~] kurapati% gcloud compute instances list
NAME                                  ZONE                    MACHINE_TYPE   PREEMPTIBLE  INTERNAL_IP  EXTERNAL_IP    STATUS
gke-kafka-default-pool-bad8e9fd-2pnb  northamerica-northeast1-a  n1-standard-1              10.162.0.2   35.203.13.30   RUNNING
gke-kafka-default-pool-bad8e9fd-9jc6  northamerica-northeast1-a  n1-standard-1              10.162.0.6   35.203.7.100   RUNNING
gke-kafka-default-pool-bad8e9fd-mrk1  northamerica-northeast1-a  n1-standard-1              10.162.0.3   35.203.10.111  RUNNING
gke-kafka-default-pool-bad8e9fd-mtrm  northamerica-northeast1-a  n1-standard-1              10.162.0.4   35.203.3.234   RUNNING
gke-kafka-default-pool-bad8e9fd-w4t1  northamerica-northeast1-a  n1-standard-1              10.162.0.5   35.203.10.53   RUNNING
[kurapati-new:~/W205/flask-with-kafka-and-spark] kurapati%
```

Deployment Templates:

Now, let us get to creating deployment templates to spin up our containers. Since we have docker-compose already we need to convert them into kubernetes deployment template (Yes, they both are different!). Easiest way is to use a fantastic tool called compose. Refer to http://kompose.io/ for more details of the project.

You need to grab the Kompose binary from the above link.

Snippet for MAC :

```
curl -L https://github.com/kubernetes/kompose/releases/download/v1.11.0/kompose-darwin-amd64 -o kompose

chmod +x kompose
sudo mv ./kompose /usr/local/bin/kompose
```

Once downloaded, run the tool from the directory where docker-compose.yaml is present :

```
kompose convert
```

This should generate a bunch of service and deployment templates.

Here are the files on my computer:
*[kurapati-new:~/W205/flask-with-kafka-and-spark] kurapati% ls *deployment**

```
kafka-deployment.yaml       mids-deployment.yaml       spark-
deployment.yaml       zookeeper-deployment.yaml

[kurapati-new:~/W205/flask-with-kafka-and-spark] kurapati% ls *service*
kafka-service.yaml    mids-service.yaml    spark-service.yaml    zookeeper-service.yaml

[kurapati-new:~/W205/flask-with-kafka-and-spark] kurapati% ls *volume*
mids-claim0-persistentvolumeclaim.yaml    spark-claim0-persistentvolumeclaim.yaml
[kurapati-new:~/W205/flask-with-kafka-and-spark] kurapati%
```

For MIDS and spark containers, I had local volume mounts. In GCP, this can be ignored for the moment but you can create the volumes as well if you desire to mount files from host to container.

Sample deployment and service templates for zookeeper is below:

```
[kurapati-new:~/W205/flask-with-kafka-and-spark] kurapati% more zookeeper-
deployment.yaml
```

```
apiVersion: extensions/v1beta1
kind: Deployment
metadata:
  annotations:
    kompose.cmd: /usr/local/bin/kompose convert
    kompose.version: 1.11.0 (39ad614)
  creationTimestamp: null
  labels:
    io.kompose.service: zookeeper
  name: zookeeper
spec:
  replicas: 1
  strategy: {}
  template:
    metadata:
      creationTimestamp: null
      labels:
        io.kompose.service: zookeeper
    spec:
      containers:
      - env:
        - name: ZOOKEEPER_CLIENT_PORT
          value: "32181"
        - name: ZOOKEEPER_TICK_TIME
          value: "2000"
        image: confluentinc/cp-zookeeper:latest
        name: zookeeper
        resources: {}
      restartPolicy: Always
status: {}
```

[kurapati-new:~/W205/flask-with-kafka-and-spark] kurapati% more zookeeper-service.yaml

```
apiVersion: v1
kind: Service
metadata:
  annotations:
    kompose.cmd: /usr/local/bin/kompose convert
    kompose.version: 1.11.0 (39ad614)
  creationTimestamp: null
  labels:
    io.kompose.service: zookeeper
  name: zookeeper
spec:
  clusterIP: None
  ports:
  - name: headless
    port: 55555
    targetPort: 0
  selector:
    io.kompose.service: zookeeper
status:
  loadBalancer: {}
```

Deploying service:

Now let us bring up the service and deployments in using kubectl using below commands

```
kubectl create --filename zookeeper-deployment.yaml
kubectl create --filename zookeeper-service.yaml
kubectl create --filename kafka-deployment.yaml
kubectl create --filename kafka-service.yaml
kubectl create --filename spark-service.yaml
kubectl create --filename spark-deployment.yaml
kubectl create --filename spark-claim0-persistentvolumeclaim.yaml [optional]
kubectl create --filename mids-service.yaml
kubectl create --filename mids-deployment.yaml
kubectl create --filename mids-claim0-persistentvolumeclaim.yaml [optional]
```

Let's now check for all the deployments and services:

```
[kurapati-new:~] kurapati% kubectl get pods -o wide
NAME                       READY   STATUS    RESTARTS   AGE    IP          NODE
kafka-b7ccfbf79-69vjj        1/1     Running   0          54m    10.20.4.6   gke-kafka-default-pool-bad8e9fd-2pnb
mids-988bf7b8d-25trl         1/1     Running   0          50m    10.20.3.5   gke-kafka-default-pool-bad8e9fd-9jc6
spark-654964b959-r8sqp       1/1     Running   0          51m    10.20.4.7   gke-kafka-default-pool-bad8e9fd-2pnb
zookeeper-5bbc74bd56-zrlhk   1/1     Running   0          54m    10.20.5.6   gke-kafka-default-pool-bad8e9fd-mrk1
[kurapati-new:~] kurapati%
```

Testing our deployment:

Login to Kafka container and create a topic:

```
[kurapati-new:~/W205/flask-with-kafka-and-spark] kurapati% kubectl exec -it kafka-
b7ccfbf79-69vjj bash

root@kafka-b7ccfbf79-69vjj:/# kafka-topics --create --topic events --partitions 1 --
replication-factor 1 --if-not-exists --zookeeper zookeeper:32181
Created topic "events".
```

Produce a random sequence and publish to the topic

```
root@kafka-b7ccfbf79-69vjj:/# seq 42 | kafka-console-producer --request-required-acks
1 --broker-list localhost:29092 --topic events
>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
root@kafka-b7ccfbf79-69vjj:/#
```

Login to mids container and read

```
[kurapati-new:~/W205/flask-with-kafka-and-spark] kurapati% kubectl exec -it mids-
988bf7b8d-25trl bash

root@mids-988bf7b8d-25trl:~# kafkacat -C -b kafka:29092 -t events
1
2
3
4
..
42

^C
Processed a total of 42 messages
```

Login to Spark container and fetch messages through pyspark.

```
[kurapati-new:~/W205/flask-with-kafka-and-spark] kurapati% kubectl exec -it spark-
654964b959-r8sqp bash
root@spark-654964b959-r8sqp:/spark-2.2.0-bin-hadoop2.6#

root@spark-654964b959-r8sqp:/spark-2.2.0-bin-hadoop2.6# pyspark
Python 3.6.1 |Anaconda 4.4.0 (64-bit)| (default, May 11 2017, 13:09:58)
[GCC 4.4.7 20120313 (Red Hat 4.4.7-1)] on linux

>>> numbers = spark \
...   .read \
...   .format("kafka") \
...   .option("kafka.bootstrap.servers", "kafka:29092") \
...   .option("subscribe","events") \
...   .option("startingOffsets", "earliest") \
...   .option("endingOffsets", "latest") \
...   .load()
>>> numbers_as_strings=numbers.selectExpr("CAST(key AS STRING)", "CAST(value
AS STRING)")
>>> numbers_as_strings.show()
+----+-----+
| key|value|
+----+-----+
|null|    1|
|null|    2|
|null|    3|

..
|null|   19|
|null|   20|
+----+-----+
only showing top 20 rows

>>>
```

Enjoy the cloud!

References:
https://cloud.google.com/kubernetes-engine/docs/tutorials/hello-app
https://kubernetes.io/docs/admin/cluster-large/