["https://docs.spring.io/spring-cloud-dataflow-server-kubernetes/docs/current/reference/htmlsingle/"](https://docs.spring.io/spring-cloud-dataflow-server-kubernetes/docs/current/reference/htmlsingle/)

---------------------------------------------------------

**Importing dependencies**

---------------------------------------------------------

# Install Kubernetes commandline utility [Local bash]

gcloud components install kubectl

---------------------------------------------------------

**Running commands**

---------------------------------------------------------

# Set the GCP Default Project [Local bash]

gcloud config set project stgcloudcoe

# Set the Default region for deploying your containers [Local bash]

gcloud config set compute/zone us-east1

* If you are working with zonal clusters, set your default [compute zone](https://cloud.google.com/compute/docs/zones):

gcloud config set compute/zone ***[COMPUTE\_ZONE]***

* If you are working with regional clusters, set your default [compute region](https://cloud.google.com/compute/docs/zones):

gcloud config set compute/region ***[COMPUTE\_REGION]***

Create a network

First, create a network for your cluster. The following command creates a network, my-net-1:

gcloud compute networks create my-net-1 \

--subnet-mode custom

# Set a VPC Network for cluster [Cloud Shell]

gcloud compute --project=synt-int-cloudcoe-rnd-prd networks create synt-vpc-scdf-network-1 --description="VPC Network for SCDF Kubernetes cluster" --subnet-mode=custom

Created [<https://www.googleapis.com/compute/v1/projects/synt-int-cloudcoe-rnd-prd/global/networks/synt-vpc-scdf-network-1>].

NAME                     SUBNET\_MODE  BGP\_ROUTING\_MODE  IPV4\_RANGE  GATEWAY\_IPV4

synt-vpc-scdf-network-1  CUSTOM       REGIONAL

Instances on this network will not be reachable until firewall rules

are created. As an example, you can allow all internal traffic between

instances as well as SSH, RDP, and ICMP by running:

$ gcloud compute firewall-rules create <FIREWALL\_NAME> --network synt-vpc-scdf-network-1 --allow tcp,udp,icmp --source-ranges <IP\_RANGE>

$ gcloud compute firewall-rules create <FIREWALL\_NAME> --network synt-vpc-scdf-network-1 --allow tcp:22,tcp:3389,icmp

Create a subnet and secondary ranges

Next, create a subnet, my-subnet-1, in the my-net-1 network, with secondary ranges my-pods-1 for Pods and my-services-1 for Services:

gcloud compute networks subnets create my-subnet-1 \

--network my-net-1\

--region us-central1 \

--range 192.168.0.0/20 \

--secondary-range my-pods-1=10.4.0.0/14,my-services-1=10.0.32.0/20 \

--enable-private-ip-google-access

# Set a VPC Subnet for cluster [Cloud Shell]

gcloud compute --project=synt-int-cloudcoe-rnd-prd networks subnets create synt-vpc-scdf-subnet-1 --network=synt-vpc-scdf-network-1 --region=us-east1 --range=10.160.0.0/20

Created [<https://www.googleapis.com/compute/v1/projects/synt-int-cloudcoe-rnd-prd/regions/asia-south1/subnetworks/synt-vpc-scdf-subnet-1>].

NAME                    REGION       NETWORK                  RANGE

synt-vpc-scdf-subnet-1  asia-south1  synt-vpc-scdf-network-1  10.160.0.0/20

# Set the name of container cluster [Cloud Shell] **- start from here**

gcloud container --project "synt-int-cloudcoe-rnd-prd" clusters create "synt-gke-scdf-cluster-1" --region "us-east1" --username "admin" --cluster-version "1.14.6-gke.1" --machine-type "n1-standard-1" --image-type "COS" --disk-type "pd-standard" --disk-size "100" --scopes "<https://www.googleapis.com/auth/devstorage.read_only>","<https://www.googleapis.com/auth/logging.write>","<https://www.googleapis.com/auth/monitoring>","<https://www.googleapis.com/auth/servicecontrol>","<https://www.googleapis.com/auth/service.management.readonly>","<https://www.googleapis.com/auth/trace.append>" --num-nodes "1" --enable-cloud-logging --enable-cloud-monitoring --enable-ip-alias --network "projects/synt-int-cloudcoe-rnd-prd/global/networks/synt-vpc-scdf-network-1" --subnetwork "projects/synt-int-cloudcoe-rnd-prd/regions/us-east1/subnetworks/synt-vpc-scdf-subnet-1" --default-max-pods-per-node "110" --addons HorizontalPodAutoscaling,HttpLoadBalancing --enable-autoupgrade --enable-autorepair --enable-autoscaling --min-nodes=1 --max-nodes=5

WARNING: Starting in 1.12, new clusters will not have a client certificate issued. You can manually enable (or disable) the issuance of the client certificate using the `--[no-]issue-client-certificate` flag.

WARNING: Starting in 1.12, default node pools in new clusters will have their legacy Compute Engine instance metadata endpoints disabled by default. To create a cluster with legacy instance metadata endpoints disabled in the default node pool, run `clusters create` with the flag `--metadata disable-legacy-endpoints=true`.

This will enable the autorepair feature for nodes. Please see <https://cloud.google.com/kubernetes-engine/docs/node-auto-repair> for more information on node autorepairs.

This will enable the autoupgrade feature for nodes. Please see <https://cloud.google.com/kubernetes-engine/docs/node-management> for more information on node autoupgrades.

Creating cluster standard-cluster-1 in asia-south1... Cluster is being health-checked (master is healthy)...done.

Created [<https://container.googleapis.com/v1beta1/projects/synt-int-cloudcoe-rnd-prd/zones/asia-south1/clusters/standard-cluster-1>].

To inspect the contents of your cluster, go to: <https://console.cloud.google.com/kubernetes/workload_/gcloud/asia-south1/standard-cluster-1?project=synt-int-cloudcoe-rnd-prd>

kubeconfig entry generated for standard-cluster-1.

NAME                LOCATION     MASTER\_VERSION  MASTER\_IP       MACHINE\_TYPE   NODE\_VERSION  NUM\_NODES  STATUS

standard-cluster-1  asia-south1  1.11.7-gke.4    35.200.182.184  n1-standard-1  1.11.7-gke.4  6          RUNNING

-> to delete cluster

gcloud container clusters delete [CLUSTER\_NAME]

# Get Credentials for gke cluster [Cloud Shell] Get credentials, so that you can use kubectl to access the cluster:

This command configures kubectl to use the cluster you created.

gcloud container clusters get-credentials synt-gke-scdf-cluster-1 --project "synt-int-cloudcoe-rnd-prd" --region us-east1

Fetching cluster endpoint and auth data.

kubeconfig entry generated for standard-cluster-1.

**run this also**

rm -rf  spring-cloud-dataflow-server-kubernetes

# Git Clone Deployment files [Cloud Shell]

$ git clone <https://github.com/spring-cloud/spring-cloud-dataflow-server-kubernetes>

# Git Clone Deployment files [Cloud Shell]

$cd spring-cloud-dataflow-server-kubernetes

/\*Optional

# Start RabbitMQ service [Cloud Shell]

kubectl create -f src/kubernetes/rabbitmq/

deployment.extensions/rabbitmq created

service/rabbitmq created

# Verify RabbitMQ deployment [Cloud Shell]

kubectl get all -l app=rabbitmq

NAME                            READY     STATUS    RESTARTS   AGE

pod/rabbitmq-76df66dc8c-p8zvk   1/1       Running   0          3m

NAME               TYPE        CLUSTER-IP      EXTERNAL-IP   PORT(S)    AGE

service/rabbitmq   ClusterIP   10.228.11.219   <none>        5672/TCP   3m

NAME                       DESIRED   CURRENT   UP-TO-DATE   AVAILABLE   AGE

deployment.apps/rabbitmq   1         1         1            1           3m

NAME                                  DESIRED   CURRENT   READY     AGE

replicaset.apps/rabbitmq-76df66dc8c   1         1         1         3m

 \*/

# Start Kafka service [Cloud Shell]

kubectl create -f src/kubernetes/kafka/

deployment.extensions/kafka-broker created

service/kafka created

deployment.extensions/kafka-zk created

service/kafka-zk created

# Verify Kafka deployment [Cloud Shell]

kubectl get all -l app=kafka

NAME                                READY     STATUS    RESTARTS   AGE

pod/kafka-broker-696786c8f7-ksls2   1/1       Running   0          10m

NAME               TYPE        CLUSTER-IP      EXTERNAL-IP   PORT(S)                      AGE

service/kafka      ClusterIP   10.228.11.201   <none>        9092/TCP                     10m

service/kafka-zk   ClusterIP   10.228.6.212    <none>        2181/TCP,2888/TCP,3888/TCP   10m

NAME                           DESIRED   CURRENT   UP-TO-DATE   AVAILABLE   AGE

deployment.apps/kafka-broker   1         1         1            1           10m

deployment.apps/kafka-zk       1         1         1            1           10m

NAME                                      DESIRED   CURRENT   READY     AGE

replicaset.apps/kafka-broker-696786c8f7   1         1         1         10m

You can use kubectl get all -l app=kafka to verify that the deployment, pod, and service resources are running. You can use kubectl delete all -l app=kafka to clean up afterwards.

# Start MySQL service [Cloud Shell]

kubectl create -f src/kubernetes/mysql/

kubectl create -f src/kubernetes/postgres/

**kubectl apply -f mysql-service.yaml**

deployment.extensions/mysql created

persistentvolumeclaim/mysql created

secret/mysql created

service/mysql created

 \

kubectl exec -it mysql-f5986679b-wslfx -- mysql -uroot -pyourpassword

kubectl exec -it postgres-7d57bf9597-hl2rd -- postgres -upostgres -proot

# Verify MySQL deployment [Cloud Shell]

kubectl get all -l app=mysql

kubectl get all -l app=postgres

NAME                        READY     STATUS    RESTARTS   AGE

pod/mysql-f878678df-zw598   1/1       Running   0          51s

NAME            TYPE        CLUSTER-IP     EXTERNAL-IP   PORT(S)    AGE

service/mysql   ClusterIP   10.228.6.225   <none>        3306/TCP   50s

NAME                    DESIRED   CURRENT   UP-TO-DATE   AVAILABLE   AGE

deployment.apps/mysql   1         1         1            1           51s

NAME                              DESIRED   CURRENT   READY     AGE

replicaset.apps/mysql-f878678df   1         1         1         51s

# Start Redis service [Cloud Shell]

kubectl create -f src/kubernetes/redis/

deployment.extensions/redis created

service/redis created

--pod name is mysqlxxxxxxx

kubectl exec -it mysql-f5986679b-9h2vd -- mysql -uroot -pyourpassword

mysql db

root name

yourpassword pwd

# Verify Redis deployment [Cloud Shell]

kubectl get all -l app=redis

NAME                         READY     STATUS    RESTARTS   AGE

pod/redis-748db48b4f-7zgrl   1/1       Running   0          44s

NAME            TYPE        CLUSTER-IP     EXTERNAL-IP   PORT(S)    AGE

service/redis   ClusterIP   10.228.12.36   <none>        6379/TCP   44s

NAME                    DESIRED   CURRENT   UP-TO-DATE   AVAILABLE   AGE

deployment.apps/redis   1         1         1            1           46s

NAME                               DESIRED   CURRENT   READY     AGE

replicaset.apps/redis-748db48b4f   1         1         1         46s

Deploy the Metrics Collector.

The Metrics Collector provides message rates for all deployed stream applications. These message rates are visible in the Dashboard UI. Run one of the following commands (depending on your message broker rabbitMQ or KAFKA) to start the Metrics Collector:

/\* Optional

# Deploy RabbitMQ Metrics Collector [Cloud Shell]

kubectl create -f src/kubernetes/metrics/metrics-deployment-rabbit.yaml

deployment.apps/metrics created

 \*/

# Deploy Kafka Metrics Collector [Cloud Shell]

kubectl create -f src/kubernetes/metrics/metrics-deployment-kafka.yaml

deployment.apps/metrics created

# Deploy Metrics Collector Service [Cloud Shell] Create the metrics service:

kubectl create -f src/kubernetes/metrics/metrics-svc.yaml

service/metrics created

# Verify Metrics Collector deployment [Cloud Shell]

kubectl get all -l app=metrics

NAME                           READY     STATUS    RESTARTS   AGE

pod/metrics-6d87c65f79-5hbnx   1/1       Running   0          2m

NAME              TYPE        CLUSTER-IP     EXTERNAL-IP   PORT(S)   AGE

service/metrics   ClusterIP   10.228.13.74   <none>        80/TCP    1m

NAME                      DESIRED   CURRENT   UP-TO-DATE   AVAILABLE   AGE

deployment.apps/metrics   1         1         1            1           2m

NAME                                 DESIRED   CURRENT   READY     AGE

replicaset.apps/metrics-6d87c65f79   1         1         1         2m

You can use kubectl get all -l app=metrics to verify that the deployment, pod, and service resources are running. You can use kubectl delete all -l app=metrics to clean up afterwards.

Optionally, you can deploy [Skipper](http://cloud.spring.io/spring-cloud-skipper/) to leverage the features of upgrading and rolling back Streams, since Data Flow delegates to Skipper for those features

# Deploy Skipper service [Cloud Shell] (optional, recommended)

kubectl create -f src/kubernetes/skipper/skipper-deployment.yaml

deployment.apps/skipper created

# Start Skipper service [Cloud Shell]

kubectl create -f src/kubernetes/skipper/skipper-svc.yaml

service/skipper created

# Verify Skipper deployment [Cloud Shell]

kubectl get all -l app=skipper

NAME              TYPE           CLUSTER-IP    EXTERNAL-IP   PORT(S)        AGE

service/skipper   LoadBalancer   10.228.2.56   <pending>     80:31487/TCP   45s

NAME                      DESIRED   CURRENT   UP-TO-DATE   AVAILABLE   AGE

deployment.apps/skipper   1         0         0            0           1m

NAME                                 DESIRED   CURRENT   READY     AGE

replicaset.apps/skipper-6ddddd5497   1         0         0         1m

# Verify Data Flow Server [Cloud Shell]

NAME                             READY     STATUS    RESTARTS   AGE

pod/scdf-server-d56b88c6-vwgn2   1/1       Running   0          1h

NAME                          DESIRED   CURRENT   UP-TO-DATE   AVAILABLE   AGE

deployment.apps/scdf-server   1         1         1            1           1h

NAME                                   DESIRED   CURRENT   READY     AGE

replicaset.apps/scdf-server-d56b88c6   1         1         1         1h

# Create Cluster-Admin binding to IAM user [Cloud Shell]

kubectl create clusterrolebinding kamana\_kanjani-cluster-admin-binding --clusterrole=cluster-admin [--user=kamana.kanjani@atos.net](mailto:--user=kshitij.lipare@atos.net)

clusterrolebinding.rbac.authorization.k8s.io/kshitij\_lipare-cluster-admin-binding created

# Create Server Roles [Cloud Shell]

kubectl create -f src/kubernetes/server/server-roles.yaml

role.rbac.authorization.k8s.io/scdf-role created

# Create Server Role Bindings [Cloud Shell]

kubectl create -f src/kubernetes/server/server-rolebinding.yaml

rolebinding.rbac.authorization.k8s.io/scdf-rb created

# Create Server Service Account [Cloud Shell]

kubectl create -f src/kubernetes/server/service-account.yaml

serviceaccount/scdf-sa created

/\* Optional

# Create RabbitMQ ConfigMap [Cloud Shell]

kubectl create -f src/kubernetes/server/server-config-rabbit.yaml

configmap/scdf-server created

 \*/

# Create Kafka ConfigMap [Cloud Shell]

kubectl create -f src/kubernetes/server/server-config-kafka.yaml

Error from server (AlreadyExists): error when creating "src/kubernetes/server/server-config-kafka.yaml": configmaps "scdf-server" already exists

# Create Server Service Deployment [Cloud Shell]

kubectl create -f src/kubernetes/server/server-svc.yaml

service/scdf-server created

# Create Server Deployment [Cloud Shell]

kubectl create -f src/kubernetes/server/server-deployment.yaml

Error from server (AlreadyExists): error when creating "src/kubernetes/server/server-deployment.yaml": deployments.apps "scdf-server" already exists

# Verify Server Deployment [Cloud Shell]

kubectl get all -l app=scdf-server

NAME                             READY     STATUS    RESTARTS   AGE

pod/scdf-server-d56b88c6-qj6bd   1/1       Running   0          1m

NAME                  TYPE           CLUSTER-IP      EXTERNAL-IP     PORT(S)        AGE

service/scdf-server   LoadBalancer   10.228.12.113   35.244.16.162   80:30571/TCP   1m

NAME                          DESIRED   CURRENT   UP-TO-DATE   AVAILABLE   AGE

deployment.apps/scdf-server   1         1         1            1           1m

NAME                                   DESIRED   CURRENT   READY     AGE

replicaset.apps/scdf-server-d56b88c6   1         1         1         1m

# Locate External IP Address [Cloud Shell]

kubectl get svc scdf-server

NAME          TYPE           CLUSTER-IP      EXTERNAL-IP     PORT(S)        AGE

scdf-server   LoadBalancer   10.228.12.113   35.244.16.162   80:30571/TCP   2m

---------------------------------------------------------

Deploying Streams with

---------------------------------------------------------

# Deploying Streams [Cloud Shell]

wget <http://repo.spring.io/release/org/springframework/cloud/spring-cloud-dataflow-shell/1.7.4.RELEASE/spring-cloud-dataflow-shell-1.7.4.RELEASE.jar>

2019-03-25 03:56:17 (7.16 MB/s) - ‘spring-cloud-dataflow-shell-1.7.4.RELEASE.jar’ saved [23833349/23833349]

---------------------------------------------------------

Starting SCDF Server with Skipper

---------------------------------------------------------

# Initialize the Data Flow server with skipper [Cloud Shell]

java -jar spring-cloud-dataflow-shell-1.7.4.RELEASE.jar

# Login into Data Flow server

dataflow config server --username user --password password --uri <http://35.229.27.55/>

app register --type source --name awssources3 --uri docker://kamanakanjani/sourcedemo1:latest

app register --type sink --name awssinks3 --uri docker://kamanakanjani/sinkdemo:latest

stream create --name awss3done --definition 'awssources3 | awssinks3'

app register --type source --name source --uri docker://kamanakanjani/sourcefinal:latest

app register --type sink --name sink --uri docker://kamanakanjani/sinkfinal:latest

app register --type processor --name processor --uri docker://kamanakanjani/processorfinal:latest

stream create --name database --definition 'source | processor | sink '

app register --type source --name finaljdbcsource --uri docker://kamanakanjani/finaljdbcsource:latest

app register --type sink --name finaljdbcsink --uri docker://kamanakanjani/finaljdbcsink:latest

stream create --name awss3done --definition 'finaljdbcsource | finaljdbcsink'

stream create --name awss3done --definition 'messagesource | messagesink'

stream deploy --name awss3done

app register --type source --name sourcejdbc --uri docker://kamanakanjani/sourceawsjdbc:latest

app register --type processor --name processosjdbc --uri docker://kamanakanjani/processorawsjdbc:latest

app register --type sink --name sinkjdbc --uri docker://kamanakanjani/sinkawsjdbc:latest

app register --type source --name messagesource --uri docker://kamanakanjani/messagesource:latest

app register --type sink --name messagesink --uri docker://kamanakanjani/messagesink:latest

stream create --name messagedone --definition 'messagesource | messagesink'

stream deploy --name messagedone

awssources3

stream create --name jdbcdone --definition 'sourcejdbc | processosjdbc | sinkjdbc '

stream deploy --name jdbcdone

jdbcdone

kamanakanjani/sinkjdbc

kamanakanjani/sourcejdbc

app register --type source --name messagesources3 --uri docker://kamanakanjani/messagesources3:latest

app register --type processor --name messageprocessors3 --uri docker://kamanakanjani/messageprocessors3:latest

app register --type sink --name messagesinks3--uri docker://kamanakanjani/messagesinks3:latest

stream create --name messages3done --definition 'messagesources3 | messageprocessors3 | messagesinks3'

stream deploy --name messages3done

app register --type source --name awssources32 --uri docker://kamanakanjani/sources3change:latest

app register --type sink11 --name awssinks32 --uri docker://kamanakanjani/sinks3change11:latest

app register --type source --name awssources32 --uri docker://kamanakanjani/awssources3finaldone:latest

app register --type sink --name awssinks32 --uri docker://kamanakanjani/awssink3finaldone:latest

! kubectl get pods -l role=spring-app

! kubectl logs .....podsname

kubectl describe pods/ticktock-log-qnk72

and edit yaml of deployment

cpu: 500m

and than create hpa

kubectl get deployments

kubectl autoscale deployment database-processor-v7 --cpu-percent=8 --min=1 --max=2

kubectl autoscale deployment database-sink-v7 --cpu-percent=5 --min=1 --max=3

kubectl autoscale deployment awss3done-awssinks3-v1 --cpu-percent=8 --min=1 --max=10

kubectl get pods -l role=spring-app --field-selector=status.phase=Running

kubectl get hpa

kubectl delete hpa hpaname

\*\*\*\*\*\*\*\*\*\*\*\*\*doneeeee