Paper 3733-2019

SAS® on Kubernetes: Container Orchestration of Analytic Work Loads

Sumanth Yamala, Core Compete

ABSTRACT

With vast amounts of data, the infrastructure demands for advanced analytics is ever increasing. There are a class of analytic problems like optimization, forecasting, machine learning in domains of retail, finance, and other verticals, where the problem could be broken down into smaller sub-problems. Example, in retail; assortment by store, or price optimization by category, in finance; security portfolio optimization by industry, the entire problem is broken down into subset of smaller jobs. In this paper we present orchestrating these jobs on Kubernetes, **GKE** powered by Google Cloud Platform (GCP). We marry the best in analytics; SAS®, with the best in container orchestration, GKE, to build a robust solution to manage analytic workloads. The solution will have the following properties: scalable, performant, elastic, cost efficient and simple to deploy.

INTRODUCTION

Containers are becoming the currency of deployment. Enterprises are increasingly running container technologies in production. GKE is a container orchestration system that automates the deployment, management, scaling, networking and availability. In this paper we introduce **analytic**, **storage**, **compute** and **orchestration** subsystems. At the end, an integrated view of subsystems is presented, that enables orchestration of SAS® analytic workloads on GKE. This paper assumes that the reader has knowledge of Kubernetes. To make this easier to follow, a problem is defined and then a solution is designed using Kubernetes.

PROBLEM STATEMENT

A retailer is planning to make decisions on SKU's to be included in the product portfolio, based on current inventory levels. The objective is to identify substitutes for products. A retailer will have a hierarchy of products and locations. In this problem the decisions are at city and sub-class level. Data pipelines, which have run on a daily basis, have already pre-partitioned inventory and sales data at sub-class level and city level and persisted the data on cloud storage (GCS). One of the analytic components, is to run a decision tree model across these hierarchies.

ANALYTIC SUBSYSTEM

From the problem statement, a SAS Viya ML decision tree model is evaluated at the retailer hierarchies. There are two containers that are part of the analytic system which run in a single pod. The containers are the following:

1) The SMP CAS container
2) A python container which uses the swat library to connect to CAS and run the decision tree model.

The following is the deployment file

```
---
apiVersion: apps/v1beta1
kind: Deployment
metadata:
  name: gcs
  labels:
    app: gcs
spec:
  replicas: 1
  strategy:
    rollingUpdate:
      maxSurge: 1
      maxUnavailable: 1
    type: RollingUpdate
  template:
    metadata:
      labels:
        app: gcs
    spec:
      containers:
      - name: gcs-container
        image: gcr.io/sgf-sas-on-k8/cas-viya-job
        imagePullPolicy: IfNotPresent
        securityContext:
          privileged: true
          capabilities:
            add:
              - SYS_ADMIN
        lifecycle:
          postStart:
            exec:
              command: ["gcsfuse", "sask8queue", "/etc/gcs"]
          preStop:
            exec:
              command: ["fusermount", "-u", "/etc/gcs"]
```

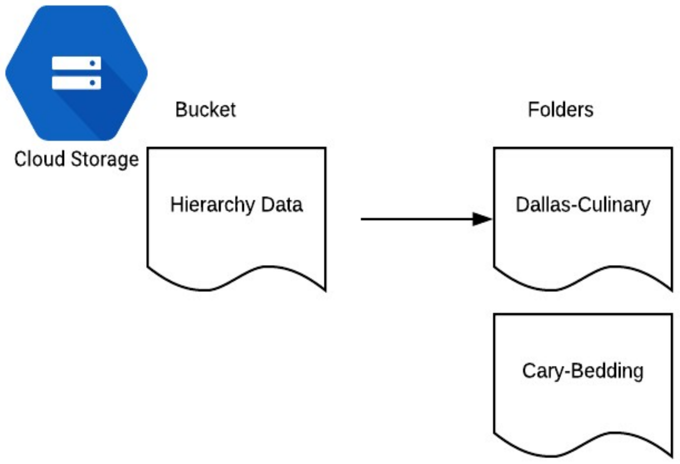
```
- name: python-swat-container
  image: gcr.io/sgf-sas-on-k8/python-swat-container
  imagePullPolicy: Always
```

STORAGE SUBSYSTEM

All analytic systems process large volumes of data. Most analytic data are stored in object storage like GCS, or persistent disks. They are inexpensive, durable and highly available. Kubernetes provides a neat abstraction between storage and compute. This delineation helps in scaling on-demand compute independent of storage. Kubernetes refers to storage as persistent volumes. https://kubernetes.io/docs/concepts/storage/persistent-volumes explains them in detail. Dynamic binding of persistent disks based on store number, user profile or any other criterion can be achieved using persistent volume claims (PVC). Local SSD's will serve as CAS cache for high throughput. The storage subsystem designed here will have the following types of volumes:

1. **GCS Fuse Mount** – This will mount a GCS bucket to all containers in the pod. This is a durable storage. It will serve as a repository for analytic data mart.

When using object storage like GCS, one would need to logically partition the data that maps to business use cases. From the problem statement, data has been pre-partitioned at the city-subclass level. There are other storage options to leverage with GKE.



MESSAGE PIPELINE SUBSYSTEM

The objective of an analytic job is to crunch data and produce results, to enable decision making. Clients invoke these analytical jobs. Clients could be users, user interfaces or machines. We abstract the client concept, by introducing a queue. In this queue clients publish messages that carries instructions which identifies:

1) Sub-class of the product hierarchy
2) City location of the location hierarchy
3) Dynamic variables and model parameters

Google Pub Sub (https://cloud.google.com/pubsub/docs/overview), provides that infrastructure, to deliver messages in a reliable and idempotent way. Messages can be in any format that the analytic system can understand. JSON is a format that makes the message easily readable and can be designed to conform to a schema. The following JSON schema has an instruction set to run decision tree model for the city – Dallas and for the sub-class – Culinary whose ID is 124901. It also describes model parameters which need to be used for tuning the model.

```
{
        "city": "Dallas",
        "sub-class": "Culinary",
        "sub-class-id": "124901",
        "model_params": {
                "maxEvals": 50,
                "maxIters": 5,
                "maxTime": 3600,
                "popSize": 10,
                "randomSeed": 445
        }
}
```

COMPUTE SUBSYSTEM

In Kubernetes one can create a cluster that meets the needs of the underlying analytical process. Examples:

1) If it is an optimization process – the cluster needs to be built on multi-core CPU's.
2) Machine learning on SAS Viya – powerful multi-core CPU and a large amount of memory should characterize the underlying clusters.

A Kubernetes cluster is a collection of VMs. This process of creating a cluster in GKE is explained in detail here. https://cloud.google.com/kubernetes-engine/docs/how-to/creating-a-cluster

SECURITY

GKE has security at node level, network level and at the control plane level. At the control plane level, the cluster can be hosted on private clusters and master authorized networks. Ingress and egress communication can be controlled by managing network policies in a namespace hosting the pods.

KUBERNETES Compute Elasticity

In the approach presented here we scale the analytic pod horizontally, based on number of messages in the message pipeline. The replication controller manages the number of analytic pods based on number of messages (analytic job instructions) in the queue.

https://cloud.google.com/kubernetes-engine/docs/tutorials/external-metrics-autoscaling describes the horizontal pod scaling set up.

ORCHESTRATION SUBSYSTEM

The orchestration system unifies the storage, compute and message pipeline tiers. The instructions carried in the message pipeline are executed by SAS Viya containers. The number of SAS Viya containers changes based on the load on the system.

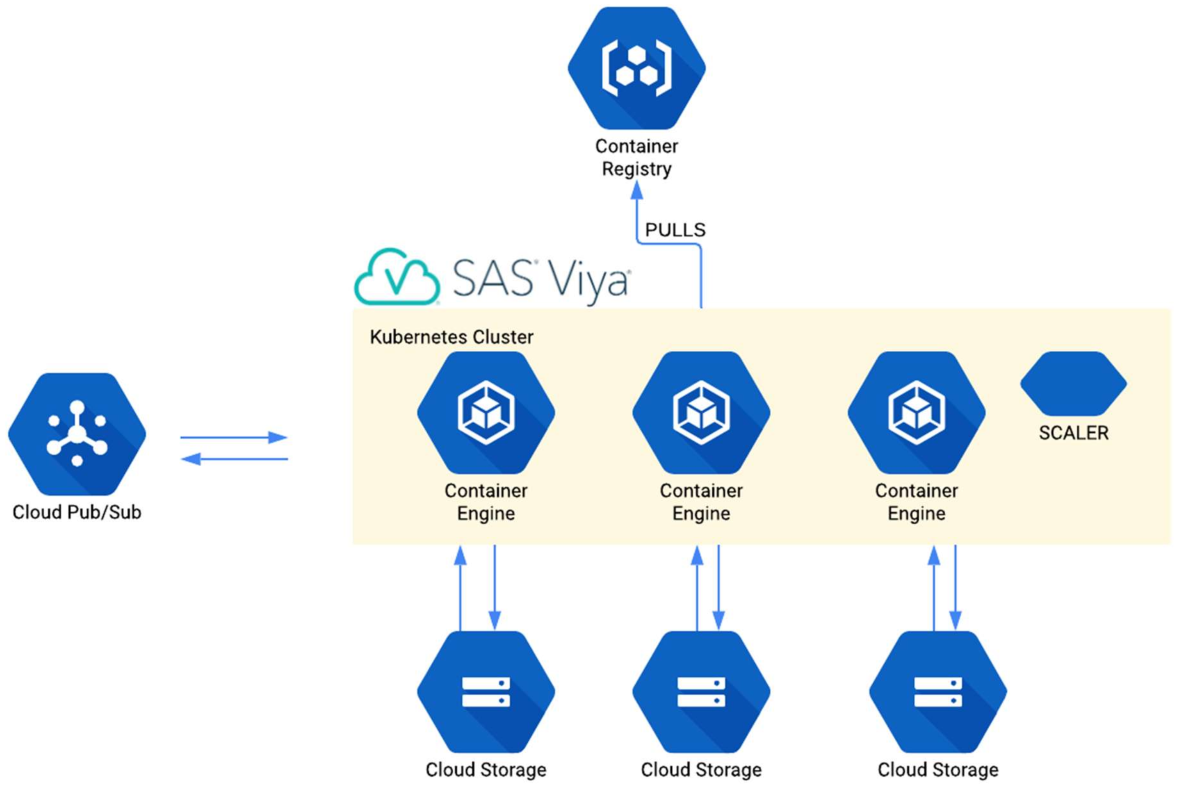


Figure 1: Flow diagram of SAS Viya on Kubernetes – processing analytical workloads

CONCLUSION

SAS® VIYA REST API’s and its support for Python makes it an open platform. Its support for containers makes it compatible with container platforms like Docker Swarm, Kubernetes and AWS Batch. The presentation here highlights how enterprises can execute SAS analytical workloads in a cloud native way. Using containers as a deployment currency along with dynamic storage management and elastic compute creates a solid foundation to manage analytic workloads and build idempotent pipelines. In addition, the solution presented here is cost efficient and simple to deploy.

REFERENCES

https://kubernetes.io/docs/home/

https://github.com/sassoftware/sas-viya-programming

ACKNOWLEDGMENTS

I would like to thank Srikanth Yadav Dodlakadi, for his contribution towards building the infrastructure.

RECOMMENDED READING

- *Pod Scaling* - https://kubernetes.io/docs/tasks/run-application/horizontal-pod-autoscale/
- *Container Recipes* - https://github.com/sassoftware/sas-container-recipes

CONTACT INFORMATION

Your comments and questions are valued and encouraged. Contact the author at:

Sumanth Yamala
Core Compete
sumanth.yamala@corecompete.com