Paper SAS687-2017

Convergence of Big Data, the Cloud, and Analytics: A Docker Toolbox for the Data Scientist

Donna DeCapite, SAS Institute Inc.

ABSTRACT

Learn what a container is and how it can be used to run SAS® Analytics for Containers. This paper will describe the architecture of containers running in the public or private cloud. The topic of leveraging the distributed compute power of Hadoop will also be discussed. This paper discusses how to work with a SAS container running in a variety of Cloud platforms. Additional topics include provisioning web-browser based clients via Jupyter Notebooks and SAS® Studio to empower data scientists with the tool of their choice. A customer use case will be discussed on how SAS® Analytics for Containers enables their IT department to meet the ad hoc, compute intensive, and scaling demands of the organization. An exciting differentiator for the data scientist, is the ability to send a portion of the analytic workload to run inside their Hadoop cluster via the SAS Hadoop Accelerators. Lastly, we discuss extending the container by pushing the analytic workload to run inside the Hadoop cluster; thus, enabling the data scientists to dive inside the data lake and harness the power of all the data.

INTRODUCTION

This paper is intended to introduce the topic of working with containers, specifically the SAS® Analytics for Containers product. As the technology industry moves away from running software directly on a desktop PC, users of software will be running software in a container and won't even know it! It should be noted that once the initial setup of building and deploying the container to the container environment is complete, all it takes to run SAS is the URL to get to the SAS client. The portability of the SAS® Analytics for Containers container opens many deployment options for customers to take advantage of the capabilities the container offers.

SAS® Analytics for Containers includes the following:

- Base SAS® and other SAS® Foundation software
- SAS® Studio web-based browser interface
- SAS/STAT® software for statistical analysis
- SAS/ACCESS® Interface to Hadoop for out-of-the box connectivity between SAS and Hadoop
- SAS/GRAPH® software for presentation graphics

Optional software add-ons are:

- Any SAS/ACCESS engine of choice
- SAS/CONNECT®: Cross-platform computing connections and parallel processing
- SAS/ETS® software: Econometrics and time series analysis
- SAS/IML®: Interactive matrix programming and exploratory analysis with integration to R
- SAS/OR® software: Operations research and optimization modeling
- SAS/QC® software: Statistical process control
- SAS® In-Database Code Accelerator for Hadoop: Publish and run SAS DS2 code in parallel inside Hadoop.
- SAS® Scoring Accelerator for Hadoop: Automate scoring processes within Hadoop for faster model performance.

GETTING STARTED WITH CONTAINERS

WHAT IS A CONTAINER?

The best way to start describing a container is to compare it to a shipping container. Shipping containers are used all over the world to transport goods from one place to another. The contents of the container are loaded and the container is picked up and moved to its destination. Along the way, it can be moved around easily. All the containers look the same to the equipment moving the container, be it a forklift, ship, train, tractor trailer truck, and so on. The contents are completely unique to each container. A software container is like a shipping container. It has a standard format that is based on open-source standards, and each container is defined specifically to run the software. A software container holds the code executables, any libraries, system tools, or anything needed to run the software. The container also describes the operating system that the executables run on. Most containers are Linux. Centos is a popular flavor of Linux as it is an upstream open-source version of Red Hat Enterprise Linux (RHEL). If you've ever downloaded software and then needed to run additional updates or hunt down a software or library dependency, you can understand the utility a container provides. All the pieces are already in the container that the software requires and eliminates the need to have different configurations to cover conflicting dependencies across all the software that's installed on a system. The software container described in this paper is a Docker container; however, similar principles can be applied to other container providers. The container allows for being built once and then can be run anywhere that runs containers!

WHAT IS THE DIFFERENCE BETWEEN A VIRTUAL MACHINE AND A CONTAINER?

Many people are familiar with working with virtual machines and the container concept is very similar. A virtual machine also has all the necessary pieces to run software. However, the way that the virtual machine consumes resources is very different from that of the container. The biggest difference to end users is that the container shares the operating system and Linux kernel with other containers that are running on the same physical host. The operating system is already running for the containers, which allows for quicker boot up time. A virtual machine needs to boot up the operating system each time it starts up. Another advantage of using containers is that the container image is registered in a central repository. This allows the container to be referenced and easily deployed from that image. The container is pulled down from the repository and started up and the running software is then available to the end user.

WHAT IS THE CLOUD?

The term "cloud" is used quite often and can mean a variety of things. From the point of view of the person running software, cloud most often refers to taking advantage of a network of computers rather than rely on a single machine to run the software. In the case of the SAS® Analytics for Containers, the container can run anywhere in the cloud, and it still will be running on a single machine but those resources are provided by the cloud. Cloud environments can be built either public or private. Some examples of public cloud are: Amazon Web Services, Google Cloud Platform, and Microsoft Azure. An example of private cloud is OpenStack.

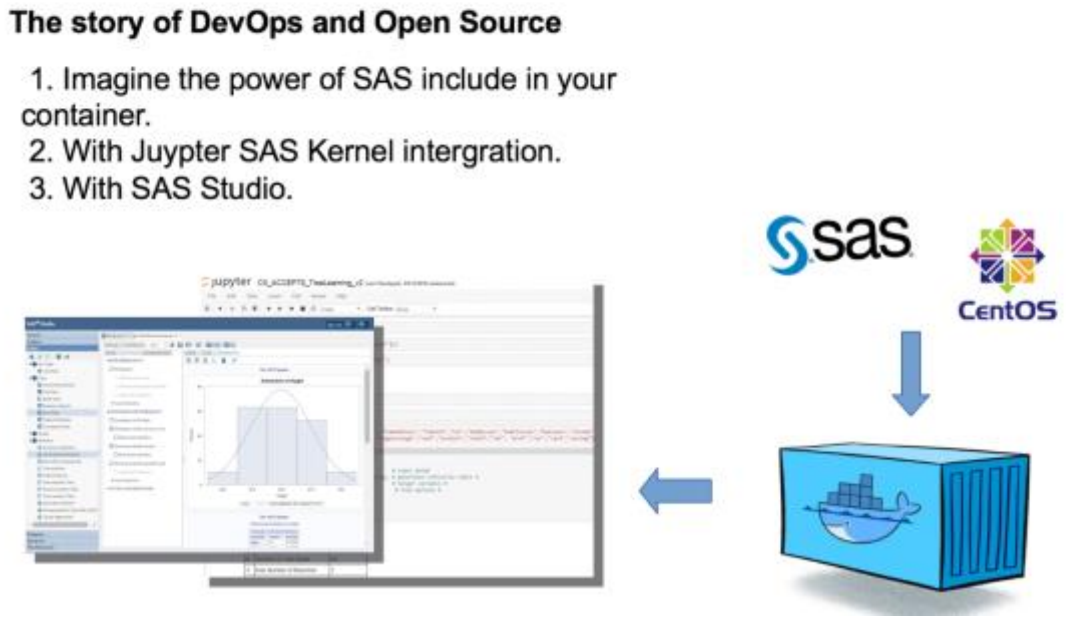
WHY WOULD I WANT TO RUN SAS IN A CONTAINER?

As more and more software is making its way to the cloud, it is important to know that you can also run SAS as a container in the cloud. Specifically, the SAS® Analytics for Containers offers various SAS components that can run in a container within a true cloud architecture.

That means that anywhere that you can run a container, you can run SAS® Analytics for Containers. Containers are cloud native, so it is easy to build and test a container and then port the container to any cloud environment without making any additional changes.

Some customers run containers in Oracle Cloud Platform, Amazon Web Services, Azure, Google Cloud Platform, and the list goes on.

There are many different deployment scenarios that support running containers. What the variety of container platforms have in common is they all have the capability to register the SAS container in the repository, start up the container and ultimately give connectivity necessary to work with SAS from a web browser.


The story of DevOps and Open Source

1. Imagine the power of SAS include in your container.
2. With Juypter SAS Kernel intergration.
3. With SAS Studio.

Figure 1. All the Tools the Data Scientist Needs Inside One Container

GETTING STARTED WITH CONTAINERS

From our experience with the SAS® Analytics for Containers product, a company's IT department will likely do all the necessary steps of pulling down the SAS Depot and building a container. Optionally, other software can be added to the container besides SAS. Ultimately, the container is the toolbox for the end user – statistician or data scientist. Once the container image is built, it is ready to be registered to a Docker repository. Once in the Docker repository, the container can then be started (from herein referred to as spun up) and the user can then connect to the container from a browser to interact with SAS in the container. Note: When deploying containers in production environments, see the section at the end of this document discussing Container Deployment possibilities.

Steps to Build a Container with SAS® Analytics for Containers

Here are the basic components needed to build a SAS® Analytics for Containers container:

- A Linux SASHome with SAS Studio tarball
- Dockerfile that Docker uses to build the container

Tarball Creation:

Start with a Linux 64-bit SAS install depot for SAS® Analytics for Containers.

Typically, all new users to the SAS® Analytics for Containers will start by performing a traditional Linux install from the SAS Install depot. Best practice recommendation is to start with a Linux kernel and distribution that closely matches your container environment. This will look like any other SAS Linux install EXCEPT that we recommend changing the default location for the SAS Studio install to be under the following directory: /usr/local/SASHOME.

1. Build a tarball of SAS

After the install is complete, the next step is to create a tarball of the SASHome directory by submitting the following command (note the use of -> to depict the command line):

```
➔ tar -cvf SASHomeTar.tar /usr/local/SASHome
```

This is the tarball that will be used to build a Docker Image

2. Create the Docker image.

Create a directory where you have decided to run Docker. For example:

```
➔ mkdir SASDocker
➔ cd SASDocker
```

In this directory place the SASHomeTar.tar file

You will also need to create a Dockerfile to instruct Docker how to create an image.

Any startup script would also be added to this directory.

Below is an example Dockerfile that is intended as a sample. The filename is Dockerfile and includes the following:

```
FROM centos
MAINTAINER Donna DeCapite Donna.DeCapite@sas.com
## install necessary libraries
RUN yum -y install numactl-libs.x86_64
RUN yum -y install libXp
RUN yum -y install passwd
RUN yum -y install libpng12
RUN yum -y install libXmu.x86_64

## example adding staff group
RUN useradd -m sas
RUN groupadd -g 1001 sasstaff

#### example adding sas user
RUN usermod -a -G sasstaff sas

## set default password for this example only
## another technique - point to /etc/passwd
RUN echo -e "foobar" | /usr/bin/passwd --stdin sas

## make SASHome directory
RUN mkdir -p /usr/local/SASHome

ADD SASHomeTar.tar /

RUN chown -R sas:sasstaff /usr/local/SASHome

EXPOSE 38080
## An optional startup script can be added (more details below)
ADD startup.sh /
ENTRYPOINT [“/startup.sh”]
```

The optional startup.sh convenience script contains the following:

```
#! /bin/bash
/usr/local/SASHome/SASFoundation/9.4/utilities/bin/setuid.sh
/usr/local/SASHome/sas/studioconfig/sasstudio.sh start
tail -f /dev/null
```

Build the Container

With both the Dockerfile, SASHomeTar.tar and the optional startup script in the directory, the following command will build the Docker image:

- ➔ `docker build -t sa4c:v1 .`

The following command displays the image that was created:

- ➔ `docker images`

The following command will run the image:

- ➔ `docker run -d -p 38080:38080 sa4c:v1`
- ➔

After the successful build, execute Docker images to see the ID that was created in the Docker repository and now you can run the container as mentioned above or refer to the name of the container image. Note: SAS Studio needs to be running in the container.

Tips and Tricks on Docker

There are a variety of alternatives to running containers with Docker. The following section outlines some commands that can help when just getting started with Docker. Depending on how the container is started, you might find it doesn't stay running if SAS Studio has not been started properly.

The following docker command shows the running containers:

- ➔ `docker ps –a`

The above command provides the container ID associated with the image.

A method to force the container to stay running is to add the –f /dev/null to the end of the command:

- ➔ `docker run -d -p 38080:38080 containeID tail -f /dev/null`

In the sample Dockerfile an entry point "ENTRYPOINT" is added to run a script at start-up. The ENTRYPOINT has been inconsistent and this command in the Dockerfile does not work with our testing of the Mac. However, if a build is done on a Linux Docker image, the Mac Docker can pull and run the image without any problem. If the ENTRYPOINT needs to be omitted from the Dockerfile, start the container with the following command:

```
docker run -d -p 38080:38080 sa4c:v1 tail -f /dev/null
```

If the startup.sh was added to the container, it could also be executed with the following command (where 6ef9b0d11e52 is the containerID):

```
docker exec 6ef9b0d11e52 ./startup.sh
```

Alternatively, the orchestration layer could submit the start-up command upon initializing the container. Docker provides many different ways to interact with the container. For example, the following commands connect to the running containers and submit commands:

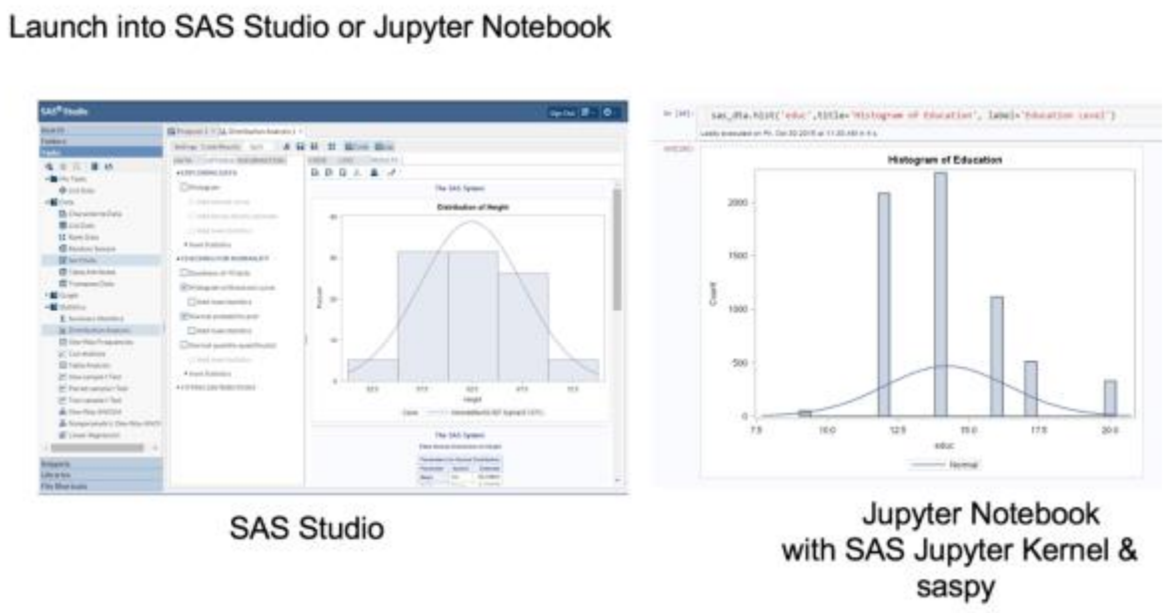
```
docker exec containerID
/usr/local/SASHome/SASFoundation/9.4/utilities/bin/setuid.sh
docker exec containerID /usr/local/SASHome/sas/studioconfig/sasstudio.sh start
```

The above details about interacting with the container are helpful when getting started with Docker. The orchestrators for containers make this interaction easier and provide hooks to place startup commands.

Ultimately to run the container, an IP address and port is needed that points to where the container is running. Depending on how the container is spun up, different cloud providers have different ways of gathering this information. Some providers give a single address for the container and have a dynamic method of producing a different port for each container spun up. Integration with the orchestration layer would be necessary to get this information. In the most simple of examples, Docker can provide an IP address with a fixed port. For example, Docker on Mac provides an IP address of 192.168.99.100. After the container and SAS Studio is started, a browser can connect via: http://192.168.99.100:38080

If the sample Dockerfile is used, the user would log in as sas with a password of foobar.

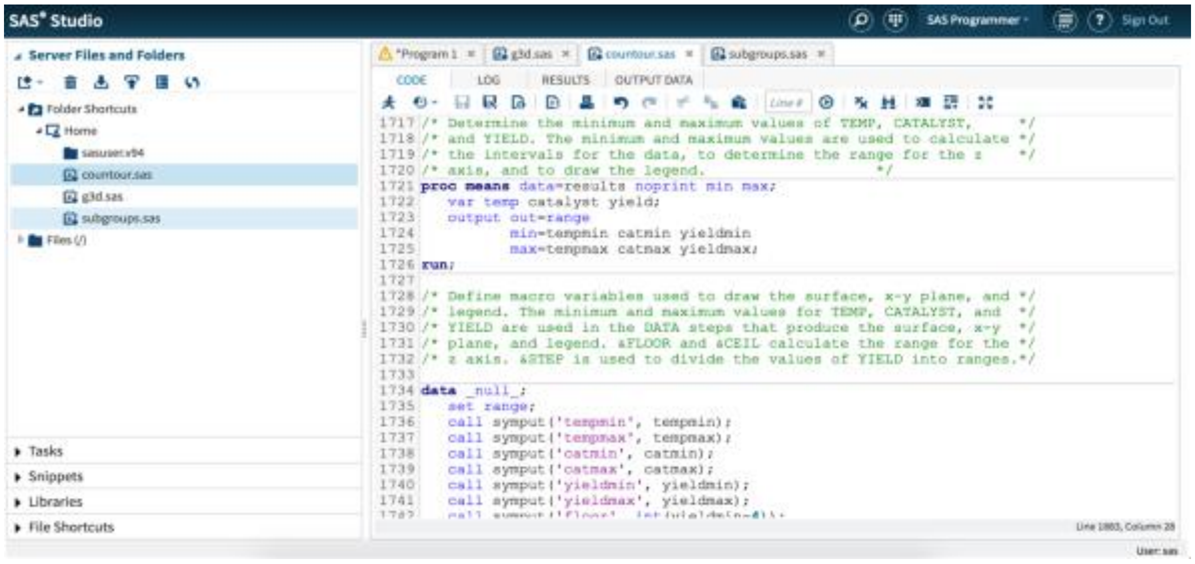
SAS Studio uses system OS authentication to connect via the web browser. That is why in the sample Dockerfile above, we created the user ‘sas’ with a password of foobar.   Therefore, once presented with the SAS Studio login page, following the above Dockerfile as an example, we could log on with the sas user ID. Jupyter Notebook can also be used as the interface to SAS® Analytics for Containers. See the following link to learn more about how to integrate with Jupyter Notebook: https://github.com/sassoftware/sas_kernel.


Launch into SAS Studio or Jupyter Notebook
SAS Studio
Histogram of Education
Distribution of Height
Jupyter Notebook with SAS Jupyter Kernel & saspy

```
In [46]:  sas_dta.hist('educ',title='Histogram of Education', label='Education Level')
```

Figure 2. SAS Studio or Jupyter Notebook

Much of this paper talks about where and how software is deployed. The next two figures show how the SAS functionality offered in the container can be leveraged from SAS Studio. For experienced SAS users, it’s still the same SAS you are used to with a couple of minor details about where SAS is running (in the container) and where the SAS files are written to (shared storage). Since the file system for SAS is technically in the container, it is recommended that a shared drive be set up. This would provide for a shared file directory where a user’s files can be written. The shared file directory would be available independent of the container so that if a container were to stop running any saved work would still be available.



```
/* Determine the minimum and maximum values of TEMP, CATALYST,      */
/* and YIELD. The minimum and maximum values are used to calculate */
/* the intervals for the data, to determine the range for the z    */
/* axis, and to draw the legend.                               */
proc means data=results noprint min max;
   var temp catalyst yield;
   output out=range
          min=tempmin catmin yieldmin
          max=tempmax catmax yieldmax;
run;

/* Define macro variables used to draw the surface, x-y plane, and */
/* legend. The minimum and maximum values for TEMP, CATALYST, and  */
/* YIELD are used in the DATA steps that produce the surface, x-y  */
/* plane, and legend. &FLOOR and &CEIL calculate the range for the */
/* z axis. &STEP is used to divide the values of YIELD into ranges.*/

data _null_;
   set range;
   call symput('tempmin', tempmin);
   call symput('tempmax', tempmax);
   call symput('catmin', catmin);
   call symput('catmax', catmax);
   call symput('yieldmin', yieldmin);
   call symput('yieldmax', yieldmax);
```

Figure 3. Code from SAS Studio

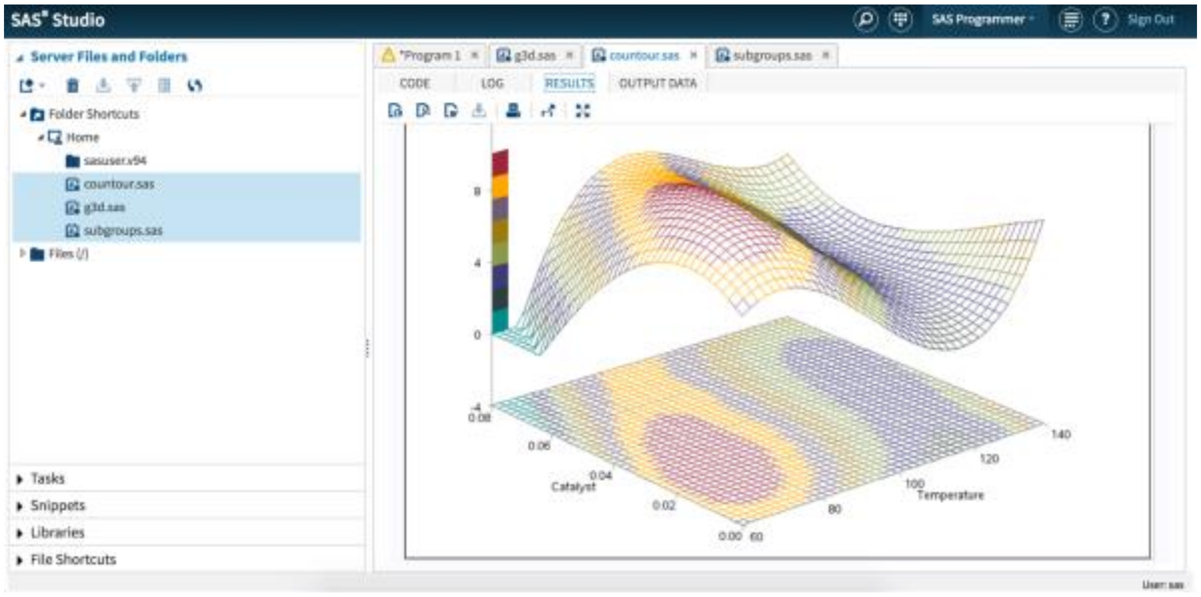


Figure 4. Output from SAS Studio

The SAS® Analytics for Containers offers some different bundle options. The basic bundle offers SAS/ACCESS Interface to Hadoop. This offers Hive interaction with Hadoop where data is pulled from Hadoop, as well as the ability to push six procedures to run inside of Hadoop (with no data movement). It also offers explicit and implicit pass-through, which enables the data scientist to run HiveQL without leaving the SAS environment. To gain the most value out of the distributed compute power of Hadoop, the SAS Accelerators for Hadoop can also be included with SAS® Analytics for Containers with Hadoop. This allows for publishing of models to the Hadoop Cluster and submitting work to Hadoop via the SAS DS2 language.

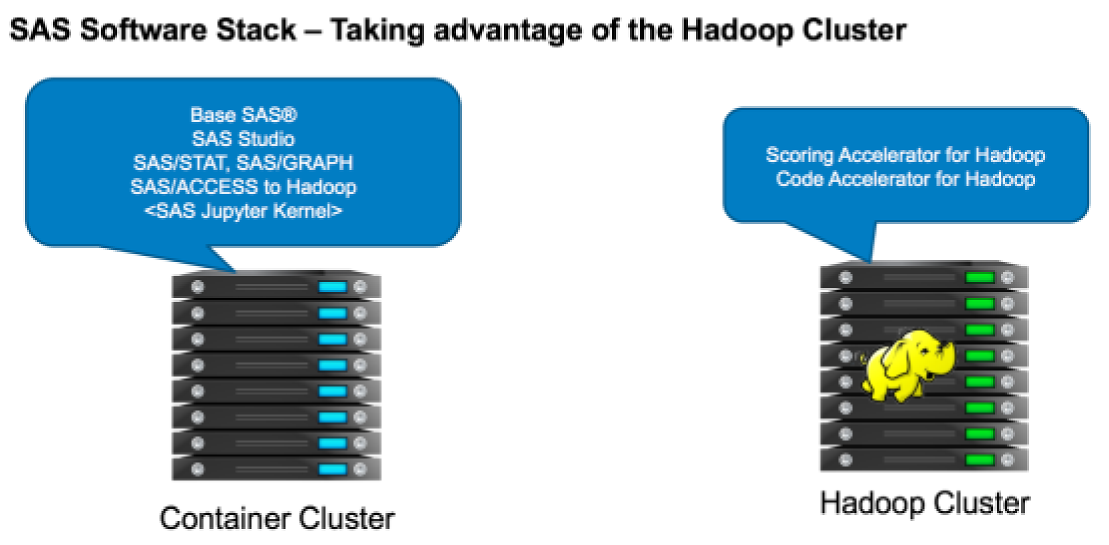

SAS Software Stack – Taking advantage of the Hadoop Cluster
Base SAS®
SAS Studio
SAS/STAT, SAS/GRAPH
SAS/ACCESS to Hadoop
<SAS Jupyter Kernel>
Container Cluster
Scoring Accelerator for Hadoop
Code Accelerator for Hadoop
Hadoop Cluster

Figure 5. Leverage Hadoop Cluster

As mentioned above, an exciting differentiator for the data scientist, is the ability to send a portion of the analytic workload to run inside their Hadoop cluster via the SAS Hadoop Accelerators. Keep in mind that although you have moved to a larger more server-based container, most container hosts are not as powerful as a multiple-node Hadoop cluster. Think of going from one server to multiple servers. Couple that with the fact that you do not even have to move the data that you are working with across the wire to another location. Most performance-related drags are tied to data movement. Why would IT spend time, resources, and money building a data lake, to then only have people suck data out of it through a straw?

Taking advantage of the processing power of Hadoop is accomplished by SAS® In-Database Code Accelerator for Hadoop and SAS® Scoring Accelerator for Hadoop. Both solutions enable the data scientist to send his or her SAS models and DATA step code to run inside the Hadoop data cluster. This workload is managed by YARN (Yet Another Resource Negotiator) by the Hadoop ecosystem. The win for the SAS developer is that he or she does not have to learn all the nuances of working with Hadoop, creating MapReduce jobs, and scheduling. Instead, the SAS data scientist only programs in SAS, just like he or she always has. For example, here is a model to score home equity loans:

PUBLISH A MODEL INSIDE HADOOP

With SAS Scoring Accelerator for Hadoop, publish the model to the Hadoop Cluster with the following code:

```
%let indconn=
    hdfs_server=&server
    hdfs_port=8020
    user=&hiveuser;

/*Publish the model*/
%indhd_publish_model(
    dir=D:\HPA_Demo_Code\ScoringAccel            /* the local directory for 
the model code */
    ,datastep=score.sas                          /* the score code as 
exported from EM */
    ,xml=score_62.xml                  /* the XML file as exported from EM 
*/
```

```
    ,modeldir=&modeldir                     /* the destination directory
for the model */
    ,modelname=DemoModel                    /* the model name which becomes
a directory under modeldir */
    ,action=replace                             /* overwrite if it exists
*/
    ,trace=yes
    );
```

EXECUTE A MODEL INSIDE HADOOP

To execute the model to run inside Hadoop, submit the following code:

```
/* First, connect to the MapReduce job tracker.  */
 %let indconn=
          hdfs_server=&server
          hdfs_port=8020
          MAPRED_SERVER=&server
          MAPRED_PORT=8021
          user=&hiveuser
          ;


/*  Run the model.  */
%indhd_run_model(
          inmetaname=&metadir./hmeq_score_input.sashdmd               /*
the metadata created by PROC HDMD */
          ,outdatadir=&tabledir./hmeq_score_output                    /*
the HDFS path to the output table */
          ,outmetadir=&metadir./hmeq_score_output.sashdmd             /*
the HDFS path to the metadata for the output table */
          ,scorepgm=&modeldir./DemoModel/DemoModel.ds2          /* the DS2
code of the model created by the publish function */
          ,forceoverwrite=true                                        /*
overwrite the table if it exists */
          ,INPUTFILE=&tabledir./hmeq_score_input                      /*
the input data set */
          ,trace=yes
          );
```

CUSTOMER USE CASE OF LEVERAGING SAS® ANALYTICS FOR CONTAINERS

Mesos and Marathon

At a large financial institutional, Mesos and Marathon are used to orchestrate their containers so that each time, someone can decide what tools and applications are needed to accomplish the task at hand. If a data scientist is given a new project and he or she quickly wants to do analysis, he or she might need a combination of software and infrastructure to make it all happen. This institution has provided a Dev/Ops oriented process to make sure that everything that the data scientist needs can all come together quickly, and containers are the driving force to make it all happen. The data scientist simply goes to a web page that provisions the Docker containers and selects the tools that he or she would like to use to complete the task by checking radio buttons on a web page. The container is then created and instantiated with a resulting web page containing the links (URLs) to the thin clients of the tools that the data scientist would like to use, like SAS Studio or Jupyter Notebook.

As this container is being spun up, a process mounts a file shared home directory, allowing the data scientist to save his or her work and share it with others. This is mounted during the start-up phase and is transparent to the end user. The user can save SAS files, Jupyter notebooks, and so on, to the shared drive in a similar manner as saving to a directory on a desktop PC.

This customer also used their Hadoop distributed computing environment that they installed on-premise. The container cloud was also located on-premise. The customer used the DS2 language to submit work to the Hadoop cluster all from the SAS container. The processing of the Hadoop work all took place inside the Hadoop cluster.

CONTAINER DEPLOYMENT POSSIBILITIES

The portability of the container allows for many different deployment configurations. This paper touches on some of the various methods, but due to the portability of the container only some methods are highlighted.

Container Services

There are a variety of offerings to help manage containers. Oracle Cloud, AWS, Google, Microsoft Azure all have a service offering to integrate with the Docker Repository to spin up containers.

- Oracle Cloud Container Service (https://cloud.oracle.com/container).
- Amazon EC2 Container Service (https://aws.amazon.com/ecs/)
- Microsoft Azure Container Service https://azure.microsoft.com/en-us/services/container-service
- Google Cloud Platform Container Clusters https://cloud.google.com/container-engine/docs/clusters/

Running Containers in OpenStack

In addition to using a public cloud to run containers, a private cloud can also host containers. There are a variety of ways to leverage OpenStack (IAAS – Infrastructure as a Service). One example is using the PAAS – Platform as a Service – OpenShift. OpenShift provides a framework for running Docker containers and leverages Kubernetes for production grade container orchestration.services to run the containers.

CONCLUSION

In conclusion, we are excited about offering SAS® Analytics for Containers! As Docker adoption continues to grow year-over-year, containers will become the norm for deploying software. This paper contains information gathered from working with building containers, deploying them to various cloud infrastructure and working with submitting work to the Hadoop cluster. Given the fast-moving technology of containers, this paper might be updated. For notifications of these updates along with other container tuning guidelines, please subscribe to the SAS Administration and Deployment Community.

ACKNOWLEDGMENTS

Many thanks to Douglas Liming, Jonah Justice and Jonathan Walker for helping with this paper.

RECOMMENDED READING

- For more information about working with the Amazon EC2 Container Service for Amazon Web Services see https://aws.amazon.com/ecs/.
- For more information about working with containers in OpenStack, see https://www.openstack.org/containers/.
- For more information about working with containers in Microsoft Azure, see https://azure.microsoft.com/en-us/services/container-service/.
- For more information about working with containers in Oracle Cloud, see

https://cloud.oracle.com/iaas/ebooks/Oracle_Container_Cloud_Service.pdf.

- For more information about integration with Jupyter Notebook, see https://github.com/sassoftware/sas_kernel.
- For more information about SAS Analytics for Containers, see https://www.sas.com/content/dam/SAS/en_us/doc/factsheet/sas-analytics-for-containers-108454.pdf
- For more information about the System Requirements to run SAS 9.4 Linux, see http://support.sas.com/documentation/installcenter/en/ikfdtnlaxsr/66396/PDF/default/sreq.pdf
-

CONTACT INFORMATION

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

Donna De Capite
100 SAS Campus Drive
Cary, NC 27513
SAS Institute Inc. Work Phone: (919) 677-8000
Fax: (919) 677-4444
Donna.DeCapite@sas.com
Web: support.sas.com
@geekyDonna

Disclaimer of Warranty

Limitation of Liability