SAS® GLOBAL FORUM 2020 #SASGF

Paper 4765-2020

SAS® Grid Execution with Amazon Web Services (AWS)

Piyush Singh, Gyanendra Shukla
TATA Consultancy Services Ltd., Indianapolis, IN.

David Glemaker, SAS Institute., Cary, NC.

ABSTRACT

Because of cost and maintenance, industries are moving towards the cloud adaption. Day by day, companies are pushing their data and processing in the cloud. Having data and processing closer makes data processing efficient. Currently, AWS is one of the key cloud providers in the market. Increase in the use of the AWS environment in the industries makes it important to establish the connection between AWS and SAS® as well.

This paper explains the techniques and configurations that SAS® users should consider to access and process the AWS files in the SAS® Grid environment. Though there are SAS® procedures like S3 which can be used in SAS® code to read/write files, we need to make other configuration changes, proxy settings, and access checks before we process AWS files in the SAS® Grid platform. This paper explains the techniques which can help the SAS® user to process cloud data easily and efficiently in the SAS® Grid environment.

INTRODUCTION

Almost every technology is being integrated with the cloud environment, and we need to make it clearly understood what we are trying to achieve with cloud migration. We need to make sure that we are not over-promising what can be expected.

This paper can be helpful as the starting point for the users/businesses who are interested in migrating to cloud SAS Grid environments from on prime deployment. This paper will explain basic expectations from SAS Grid migration to AWS and include a few trouble-shooting mechanisms while running SAS code in on-premise SAS Grid to access AWS data.

WHY MOVE TO AWS?

This is one of the most important and frequently asked questions—why do you need to move to AWS? What is the expectation? Is it cost, performance, or IT strategy to move? These questions need to be answered first before the start of the actual move, and this will help to have clear expectations from this migration.

COST SAVING

One of the key general perceptions about the AWS move is to save the significant cost associated with building and supporting platforms. This may not be as straight forward with respect to SAS Grid deployment in AWS. SAS Grid deployment cost comes with two sides, infrastructure and SAS licensing which are given below:

AWS Infrastructure

Different customers have different terms and relationships with AWS such as long term vs short term. Infrastructure cost depends on the customer strategy with the AWS contract and what kind of SAS job execution SLA they can live with. Generally, SAS Grid in AWS performs better with larger AWS ("i" instances) as it comes with the RAM, CPU, Network Speed, and large fast Ephemeral Disk, which is what we need for performance. Underneath storage for SAS Grid should meet 100MB/sec/CPU or 50MB/sec/vCPU (vCPU is a thread of a

CPU core and default multithreading is 2). This kind of EC2 instance, with the required RAM, is expensive. But if we try to save AWS cost by compromising the required infrastructure as per available SAS core, you won't be able to take full advantage of what you are paying for in the SAS Grid license. Paying more for a SAS license doesn't mean you get the expected throughput if the required infrastructure is not available. (Because of) the kind of requirement that SAS Grid needs with the AWS platform, it may not be cheaper to run on the cloud.

SAS Licensing Cost

There is no difference between adding on-premise servers and AWS instances. X cores of EC2 instances need to deploy with X/2 core of AWS SAS license, and this license is irrespective of whether it's on-premise or AWS (1 real core = 2 AWS vCPUs). Considering the core licensing mechanism, again, SAS Grid in AWS may not be cheaper than on-premise SAS Grid.

BUSINESS S/ IT STRATEGY

If SAS is the only analytical platform that is on-premise and consumes the business data which is in AWS, or if your organization has all other platforms in AWS, in any of these cases, it's advisable to keep the execution closer to data to keep the data transfer/access within the AWS network. Likewise, today’s businesses are moving more towards the cloud, and if your entire data is already in the cloud, it makes sense to have SAS compute there in the cloud with the data. This may be one of the key reasons for a cloud move for SAS computing.

In some of the cases, customers are not willing to follow the same traditional mechanism to maintain the on-premise servers irrespective of their utilization. Apart from that, the most attractive reason from the IT perspective is the ease of new resources procurement if needed. If you need to extend the SAS Grid nodes, the cloud certainly can help to reduce the time from weeks or months to minutes or hours. If you have the SAS license available, you can immediately launch the AWS EC2 instance/AMI and configure it with SAS Grid.

Note - Because of insignificant benefits, the migration is the point when customers are looking into the SAS Viya platform as an alternative to building the SAS Grid platform in the cloud. The SAS Viya is built for cloud technologies, and it may be a better alternative of moving SAS Grid to the cloud, depending on the type of SAS job. The AWS infrastructure for SAS Viya is not as expensive as the SAS Grid, and the overall deployment process is much easier. A generally shared file system plays as a single point of failure for SAS Grid, but SAS Viya doesn't require a shared file system which reduces the associated cost as well.

However, if a shared drive is mounted on the CAS Controller and CAS Workers in the same directory path, then

- CAS can execute parallel loads from the shared file system into the memory of the following data types: sas7bdat, csv, and sashdat
- Users will experience a more consistent and ease of access of data. The SAS Studio, SAS Grid, CAS (Services, Controller, and Workers) can all access the shared file system as needed, consistently, when the shared file system is mounted identically to each machine.

DATA ACCESS PERFORMANCE

Sometimes, as per organizational IT strategy, it becomes important to move the compute engine closer to data to avoid any additional latency. If your data is already in AWS, it makes sense to move SAS Grid into the AWS environment and this will make execution more efficient and better performing.

SAS ADMINISTRATION

There is no difference between on-premise and AWS SAS Grid from an administration perspective. Most of the SAS and admin related scripts would work in the same way irrespective of being on-premise or in the cloud. The SAS Management Console is still going to be the primary application for user access management in the cloud deployment. The key challenge with administration automation is the hostname with different SAS and LSF configuration files with the SAS deployment.

Note - As of now, SAS doesn't provide any cloud formation template for launching new EC2 instances as a Grid node, but there is a quick-start template for SAS Grid on the AWS website: https://aws.amazon.com/quickstart/architecture/sas-grid/. Customers are still in a learning mode and trying to automate the process as per their own existing AWS environment and expertise.

HOW TO ACCESS AWS DATA IN SAS?

There are different ways to access the AWS data in SAS:

PROC S3

You can use the PROC S3 procedure to access the data to and from AWS. This is one of the simplest ways to access the AWS data, available with SAS9.4 M4 and the above release of SAS. It doesn't require any additional SAS component to use PROC S3. A simple form of PROC S3 code is as below:

```
filename _s3file temp;
%let _localfile=%sysfunc(pathname(_s3file));
PROC S3
keyid="AWS_key_id"
secret="AWS_secret_key"
region=sasregion_correspoinding_toAWS /* Note */
get "AWS_bucket/file" "&_localfile"
run;
```

You can use "`_s3file`" as input file reference in the below SAS code to read the data fetched from AWS in above SAS code:

```
proc import datafile=_s3file replace
dbms=csv
out=awsdata;
getnames=yes;
run;
```

But the problem with this kind of example is to have clear text credentials in SAS code. This may not be a good approach from the security point of view.

Note- PROC S3 doesn't take the AWS specified regions as it is. There are PROC S3 regions corresponding to AWS regions and you need to provide the SAS value instead of the AWS name for the region.

PROC S3 FOR LIST, COPY, DELETE, PUT, GETDIR AND PUTDIR

In addition to the get command, in the above proc s3, the list, copy, delete, put, getdir, and putdir commands are available.

The GET, GETDIR, PUT, and PUTDIR statements take advantage of this faster data transfer method when transfer acceleration is enabled, https://docs.aws.amazon.com/AmazonS3/latest/dev/transfer-acceleration.html

Note: the code below uses the .tks3.conf file detailed in the next section.

```
/* Listing */
proc s3 ;
   list "/mybucketname/mydir";
run;

/* Copy bucket to bucket */
proc s3 ;
   copy "/mybucketname/mydir/myfile" "/mybucketname/mydir/myfile";
run;

/* Copy local file to bucket file */
proc s3 ;
   put "/home/myid/myfile" "/mybucketname/mydir/myfile1" ;
run;

/* Copy bucket file to local file */
proc s3 ;
   get "/mybucketname/mydir/myfile1" "/home/myid/myfile1";
run;

/* Copy bucket dir to local dir */
proc s3 ;
   getdir "/mybucketname/mydir" "/home/myid/mydir";
run;

/* Copy local dir to bucket dir */
proc s3 ;
   putdir "/home/myid/mydir" "/mybucketname/mydir";
run;

/* Deletes a file from bucket */
proc s3 ;
   delete "/mybucketname/mydir/myfile";
run;
```

PROC S3 WITH .TKS3.CONF

.tks3.conf can be used to avoid the clear text credentials in the PROC S3 SAS code. Keyid and secret keys can be stored in the .tks3.conf file in the user's home directory. Though this is hidden file, it can be protected by giving 700 or 600 permission so that no one else can read this file.

```
$ cat .tks3.conf
keyID=XXXXXXXXXXXXXXXXXXX
secret=YYYYYYYYYYYYYYYYYYYYYYYYYYY
region=sasregion_correspoinding_toAWS
$
```

After creation of this file you can execute PROC S3 and it will use the credentials directly from one's home directory without mentioning clear text in the SAS code. Now PROC S3 can be executed without keyid or secret key as shown below:

```
filename _s3file temp;
%let _localfile=%sysfunc(pathname(_s3file));
PROC S3
get "AWS_bucket/file" "&_localfile"
run;
```

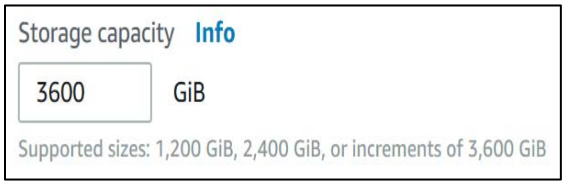
PROC S3 WITHOUT KEYID AND SECRET KEY

If you are running SAS9.4M6 with the latest hotfix, then there is no need to provide AWS credentials like keyid and secret key in PROC S3 SAS code or .tks3.conf configuration file. PROC S3 can read the credentials from the AWS service for the IAM role used by the EC2 instance. SAS can access the AWS credentials automatically.

AWS FSX SHARED FILE SYSTEM

SAS Grid performance is always dependent on the performance of the shared file system that is chosen for the shared data storage. Some customers are considering AWS FSx as the shared file system because it is fully managed, works natively with S3, is scalable, and handles high-performance workloads. Although this is quick and easy to use, be sure you are aware of how FSx works and the nuances that come with it before deploying to a highly critical production environment. Also see the FSx for Lustre user guide on page 8 to install the Lustre Client before mounting or running any Lustre CLI commands.

1. Supported sizes as of the time of this paper are 1.2TiB, 2.4TiB, or increments of 3.6TiB (not editable after creation; be sure to use the right size for storage and Throughput capacity).

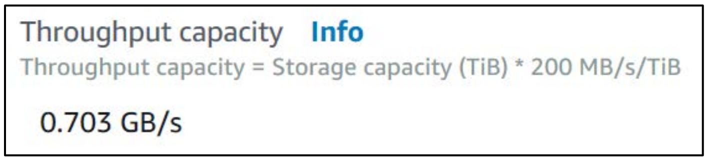

Storage capacity Info

3600 GiB

Supported sizes: 1,200 GiB, 2,400 GiB, or increments of 3,600 GiB

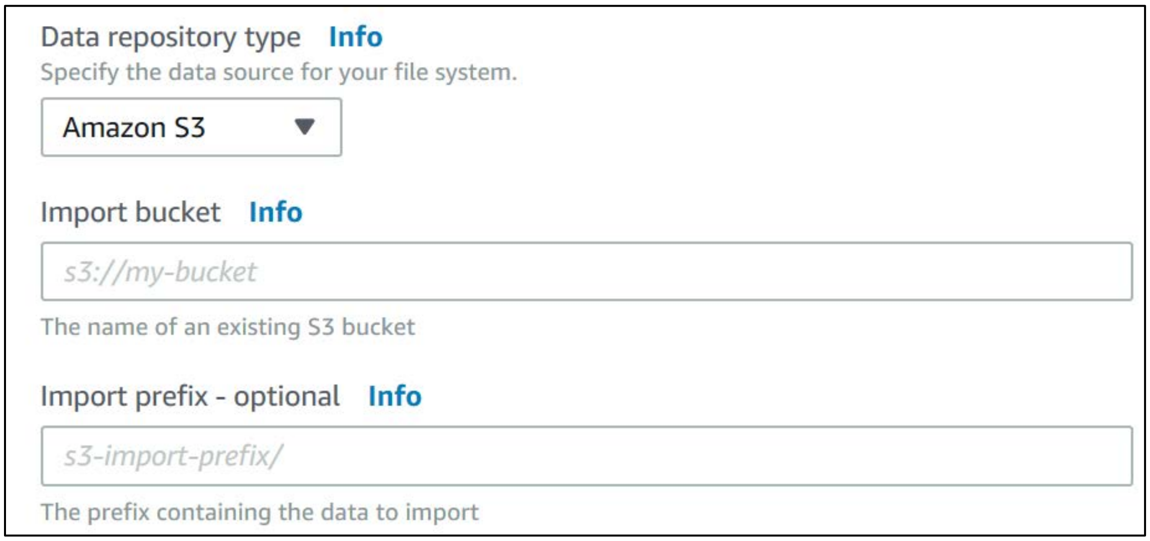
Display 1. Enter the Desired Storage Capacity needed in AWS Management Console for FSx

2. Throughput capacity = Storage capacity (TiB) * 200 MB/s/TiB. Note below when 3600 GiB is selected, then the Throughput is 0.703 GB/s calculated by AWS.


Throughput capacity Info
Throughput capacity = Storage capacity (TiB) * 200 MB/s/TiB
0.703 GB/s

Display 2. You will see the calculated Throughput Capacity in the AWS Management Console for FSx

3. FSx does an initial read of the designated import S3 bucket location to create metadata of all the existing files and directories at launch time. It will then populate these files in FSx as they are referenced on the file system. When creating an FSx filesystem enter the name of your bucket. You can also enter a directory if needed.

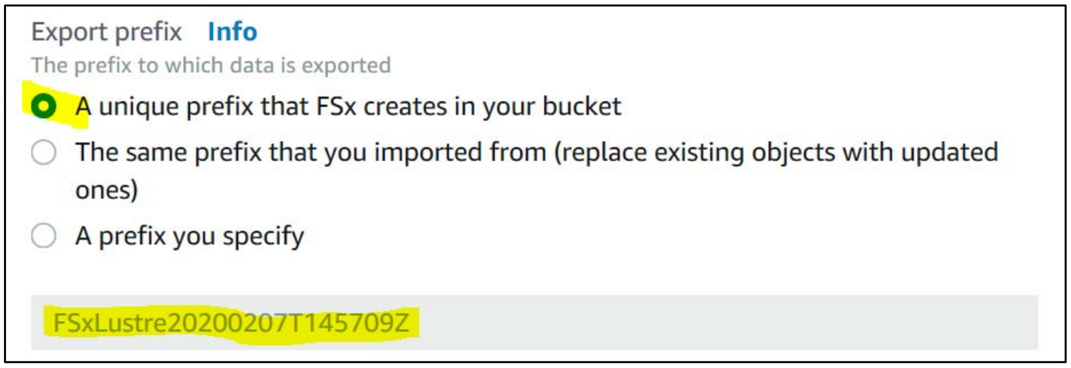

Data repository type Info
Specify the data source for your file system.
Amazon S3
Import bucket Info
s3://my-bucket
The name of an existing S3 bucket
Import prefix - optional Info
s3-import-prefix/
The prefix containing the data to import

Display 3. Select S3 Data Repository and provide Bucket information in the AWS Management Console for FSx

If you know the majority of the files are going to be hit by users, you can force an initial load of all data to the FSx filesystem using the commandline tool. This will help to avoid the user experiencing a performance hit of the initial load of data at first reference time. Force the load of all data with the following command.

```
nohup find local/directory -type f -print0 | xargs -0 -n 1 sudo lfs
hsm_restore &
```

4. If you add files to the import S3 location after the initial launch, they are not realized in the FSx environment unless you force a reload of the metadata.
5. If you create new files in the FSx environment, they do not automatically populate in the original import S3 location.
6. You can force newly created files in the FSx environment to populate the S3 location with the commandline tool. However, it does not populate the original import S3 location by default. By Default, it uses the export S3 location given at launch time with timestamped directory name in the format of FSxLustreYYYYMMDDT999999Z.


Export prefix Info

The prefix to which data is exported

[✓ A unique prefix that FSx creates in your bucket]

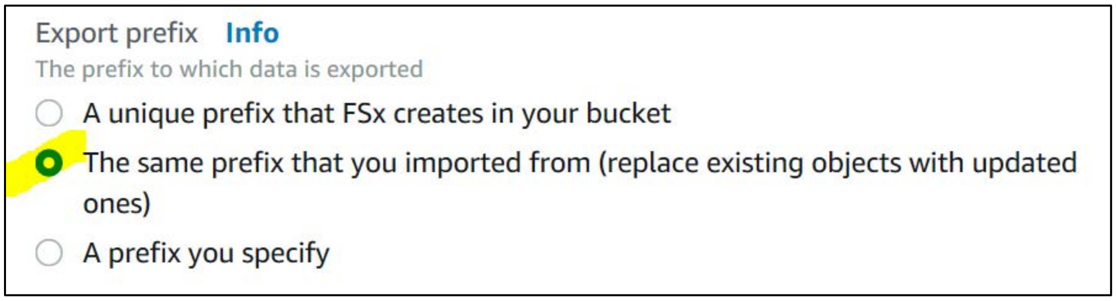
The same prefix that you imported from (replace existing objects with updated ones)

A prefix you specify

FSxLustre20200207T145709Z

Display 4. Default Export Location in the AWS Management Console for FSx

You can set this at launch time to the same location by checking "The same prefix" radio button.


Export prefix Info

The prefix to which data is exported

A unique prefix that FSx creates in your bucket

[✓ The same prefix that you imported from (replace existing objects with updated ones)]

A prefix you specify

Display 5. Select "Same Prefix" in the AWS Management Console for FSx if that is your desired Export Location

You can export individual files to the export S3 location with the following command.

```
sudo lfs hsm_archive path/to/export/file
```

You can export the entire filesystem to the export S3 location with the following command.

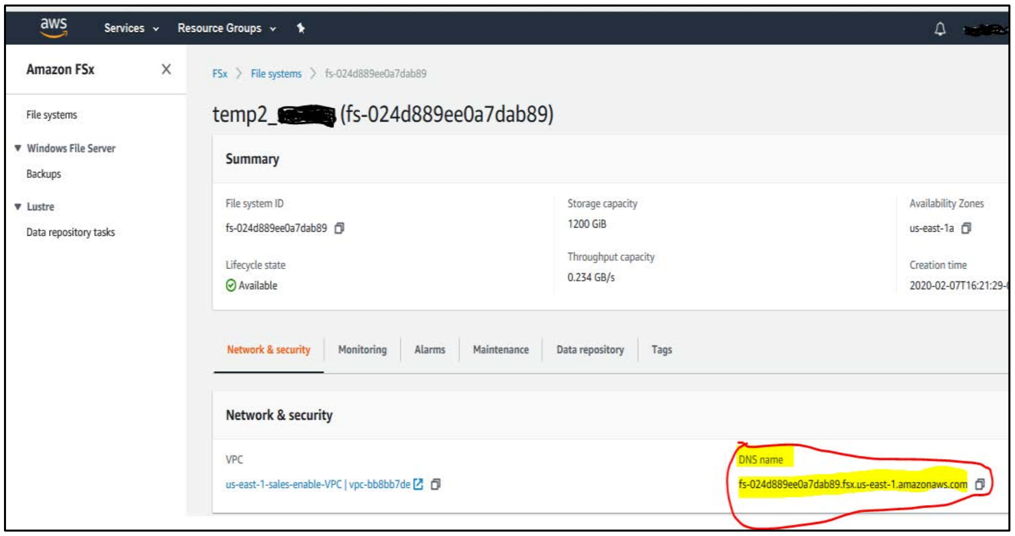
```
nohup find local/directory -type f -print0 | xargs -0 -n 1 sudo lfs
hsm_archive &
```

If you are using this as a backup for your data, then have a strategy to schedule this command to run at appropriate time intervals to meet your requirements. Also keep this in mind when you select the Export Location as to which prefix location best meets your need for the desired Export Location.

7. You can mount your FSx filesystem to your EC2 instance /fsx directory with this command and substituting in your ***FSx dnsname***.

   Note: for optimal performance you need to add the "-o noatime, flock" options.

```
sudo mount -t lustre -o noatime,flock file_system_dns_name@tcp:/fsx /fsx
```



temp2_[redacted] (fs-024d889ee0a7dab89)

Summary

File system ID: fs-024d889ee0a7dab89
Lifecycle state: Available
Storage capacity: 1200 GiB
Throughput capacity: 0.234 GB/s
Availability Zones: us-east-1a
Creation time: 2020-02-07T16:21:29-

Network & security

VPC: us-east-1-sales-enable-VPC | vpc-bb8bb7de
DNS name: fs-024d889ee0a7dab89.fsx.us-east-1.amazonaws.com

Display 6. Find your FSx DSN name for the mount command in the AWS Management Console under FSx services "file systems" and click on your FSx "File system name" or "File system ID" and then under "Nework & Security" tab

8. Refer to the FSx Lustre user's guide in the reference section for detailed information on the above and other related topics.

OTHER AWS DATA SERVICES

S3 is not the only AWS data that SAS can access. The SAS Access engines like SAS/Access to Hadoop for the AWS EMR data, SAS Access to JDBC and ODBC for many other AWS data services, and Database vendor Access engines as they are already or are starting to provide their data services in the AWS cloud.

AWS CLI

Although AWS provides a very nice GUI Management Console, you commandline gurus will want to also install AWS CLI. This will allow you, from the commandline, to explore information about your AWS environment as well as control it, for example starting and stopping EC2 instances. You can find the install instructions in the AWS CLI User Guide. You may want to enable some of the commands with SAS xcmd and SAS Macros. In order for a user to run AWS CLI commands, they will need to run the AWS configure command below. The user guides tell how to get the values for the prompts.

```
aws configure
```

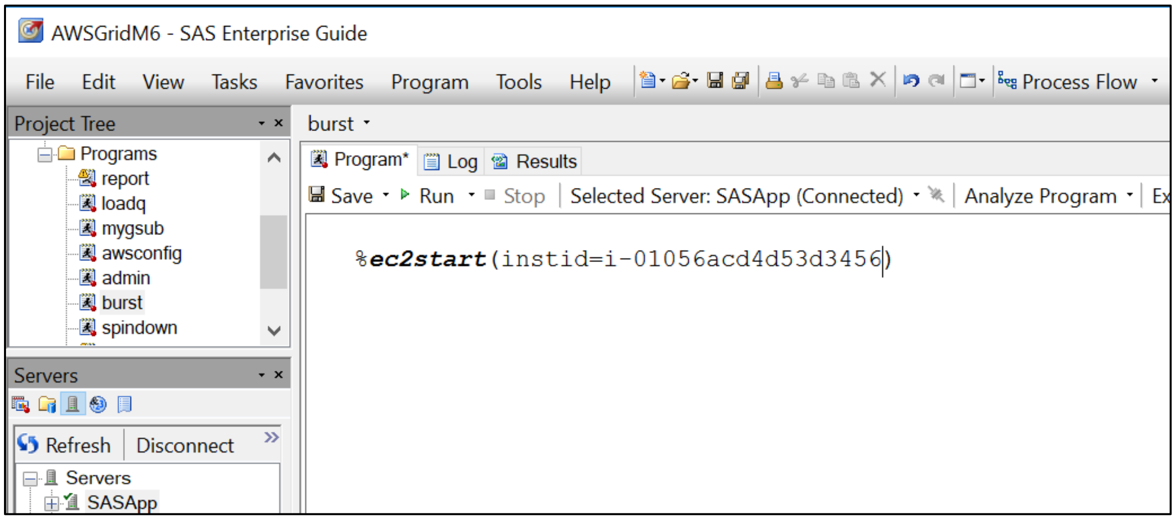
One example of enabling xcmd and SAS macros is the EC2 start command.

```
aws ec2 start-instances --instance-ids myinstanceid
```

Take that code and wrap a SAS Macro around it.

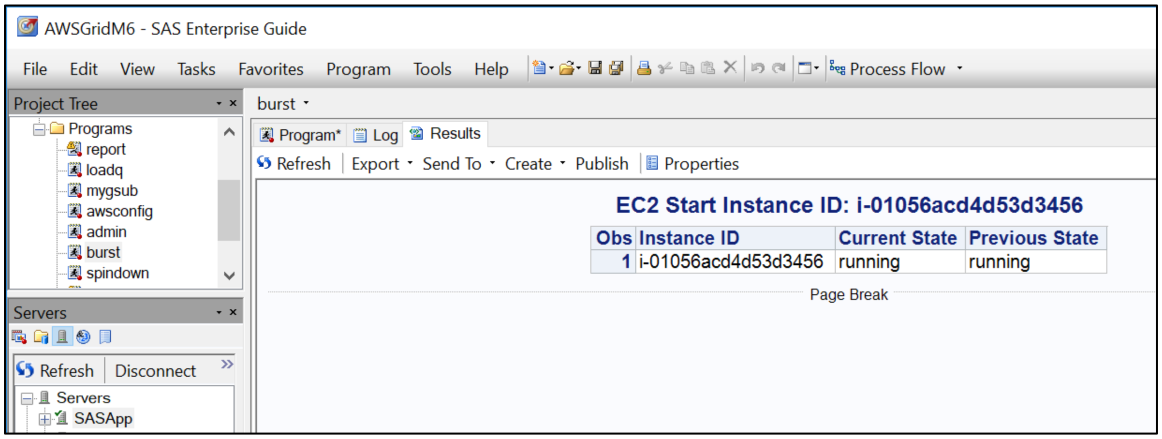
```
%macro ec2start(instid=yourdefaultinstid);
filename ec2strt pipe "aws ec2 start-instances --instance-ids &instid";

data ec2strt;
   infile ec2strt missover firstobs=7;
   retain instid currst prevst;
      input;
          if index(_infile_,'Name') and _n_=1 then
      do;
         length currst $20.;
         currst =
compress(substr(_infile_,index(_infile_,'Name')+5),':,"');
         instid = "&instid";
      end;
          if index(_infile_,'Name') and _n_=5 then
      do;
         length prevst $20.;
         prevst =
compress(substr(_infile_,index(_infile_,'Name')+5),':,"');
         output;
      end;
   lable instid="Instance ID"
         currst="Current State"
         prevst="Previous State";
run;

title "EC2 Start Instance ID: &instid";
proc print label;
run;
title;
%mend ec2start;
```

Call your macro from your user interface of choice, providing the instance id.


AWSGridM6 - SAS Enterprise Guide
File Edit View Tasks Favorites Program Tools Help Process Flow
Project Tree
Programs
report
loadq
mygsub
awsconfig
admin
burst
spindown
Servers
Refresh Disconnect
Servers
SASApp
burst
Program* Log Results
Save Run Stop Selected Server: SASApp (Connected) Analyze Program Ex

```
%ec2start(instid=i-01056acd4d53d3456)
```

Display 7. SAS Enterprise Guide 7.1 running %ec2start macro



EC2 Start Instance ID: i-01056acd4d53d3456

Obs	Instance ID	Current State	Previous State
1	i-01056acd4d53d3456	running	running

Display 8. SAS Enterprise Guide 7.1 Results from running %ec2start macro

TROUBLESHOOT AWS ACCESS ISSUES IN SAS SESSION

Though the issue and fix with AWS access can vary from customer to customer and their AWS environment, there are a few common errors that you can see during AWS access in a SAS session like:

```
"ERROR: Could not get object.", "503 Service Unavailable"
```

and proxy related issues. Other key error codes can be found in https://docs.aws.amazon.com/AmazonS3/latest/API/ErrorResponses.html document.

LOGGER STATEMENT

As stated earlier, there may be different issues as per individual customer AWS environments, but the below macros can be helpful to pinpoint the issues, and then you can take appropriate action to resolve them. If you are facing any issue with your SAS code accessing AWS, you can add the below mentioned macros at the top of your SAS code and execute:

```
%log4sas();
%log4sas_logger("App.tk.s3","level=trace");
%log4sas_logger("App.tk.htclient","level=trace");
```

These macros are known as logger statements. These macros will provide detailed information on your terminal about the connections and communication at the different layers with which SAS is interacting.

You can use these loggers in Base SAS sessions. If you use these loggers in SAS Enterprise Guide, it won't produce the desired output as it can't change the app logger location because the SAS Workspace Server session has already been established before running any SAS code. So you can launch the Base SAS session and run the logger appended SAS code here and detailed messages will be printed.

PROXY SETTING

This is one of the commonly seen issues when someone starts the work with AWS using SAS. Mostly customers set their own proxy settings to work with their cloud environment. One needs to know the organization proxy setting and use the proxy information at the top of their SAS code and then execute, as given below:

```
options set=HTTP_proxy="customer_proxy_setting";
options set=HTTPS_proxy="customer_proxy_setting";
```

CONCLUSION

This paper is written to help SAS users who are new to the AWS environment to understand the basic concept, basic SAS procedures to use with AWS buckets, and a few frequently seen issues in SAS sessions. The purpose of this paper is to increase the awareness about the SAS platform in the AWS cloud and as a starting point to move into the cloud.

REFERENCES

Amazon FSx for Lustre User Guide.
https://docs.aws.amazon.com/fsx/latest/LustreGuide/what-is.html
https://docs.aws.amazon.com/fsx/latest/LustreGuide/LustreGuide.pdf

AWS CLI User Guide.
https://aws.amazon.com/quickstart/architecture/sas-grid/
https://docs.aws.amazon.com/cli/latest/userguide/cli-chap-welcome.html

Sangwan, Prasoon, Tanuj Gupta, and Piyush Singh. "Key Requirements for SAS® Grid Users." In *SAS Global Forum*. 2016.
https://support.sas.com/resources/papers/proceedings16/7140-2016.pdf

Singh, Piyush, Sangwan, Prasoon, and Randolph, Steven. "Read SAS® Metadata in SAS® Enterprise Guide" In *SAS Global Forum*. 2017.
http://support.sas.com/resources/papers/proceedings17/1275-2017.pdf

Singh, Piyush Kumar, and Prasoon Sangwan. "Load balancing and job scheduling manager within a high-performance computing environment." U.S. Patent Application 16/007,698, filed December 13, 2018.

CONTACT INFORMATION

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

Piyush Singh	Gyanendra Shukla	David Glemaker
piyushkumar.singh@tcs.com	gyanendra.shukla@tcs.com	david.glemaker@sas.com
www.tcs.com	www.tcs.com	www.sas.com