Paper SAS2075-2018

Auto Scaling SAS® Real-Time Decision Manager on Amazon Web Services

Chris West, SAS UK Professional Services

ABSTRACT

SAS® Real-Time Decision Manager combines SAS® Analytics with business logic and contact strategies to deliver enhanced real-time recommendations and decisions to interactive customer channels such as websites, call centers, point-of-sale (POS) locations, and automated teller machines (ATMs). SAS Real-Time Decision Manager helps you make smarter decisions by automating and applying analytics to the decision process during real-time customer interactions. Due to the interactive nature of these channels, the volume of requests can be subject to wide fluctuations, either hourly, daily, or seasonally.

To enable our customers to accommodate the variance caused by a Black Friday or Cyber Tuesday event, this session presents a live demonstration of how elasticity can be brought to SAS Real-Time Decision Manager using Amazon Web Services (AWS), enabling the platform to dynamically grow or contract in response to the observed workloads.

This paper demonstrates the use of AWS Auto Scaling groups, Launch configurations, and Scaling plans to simply and seamlessly match application capacity to demand–enabling SAS customers to best meet the demands of their customers, while minimizing their infrastructure costs.

INTRODUCTION

As competition grows, organizations will focus on improving the customer experience to create a unique, sustained competitive advantage and to rapidly respond to changing market dynamics. Innovative, interactive marketing tactics, such as real-time decision making, can prompt significant improvements in sales and retention, revenue and profitability, and customer experiences.

The purpose of this paper is to bring into focus the activities required to configure auto-scaling in Amazon Web Services for the SAS Real-Time Decision Manager run-time environment. We have chosen to implement this on AWS. However, the method described in this paper could easily be modified to apply to other cloud providers such as Google Cloud Platform (GCP) and Microsoft Azure. The paper details some of the technical aspects of the solution, in order to highlight the key design, planning and implementation concepts required in a live production environment.

This paper has been written for a technical audience of stakeholders and is aimed at Cloud Architects, Technical Architects, and consultants. In this paper, we provide an overview of the AWS technologies used to configure Auto Scaling, we look at the method that can be used to implement this, and we highlight some design considerations that should be taken into account when deploying the SAS Real-Time Decision Manager run-time environment in AWS. We conclude with some additional future implementation opportunities.

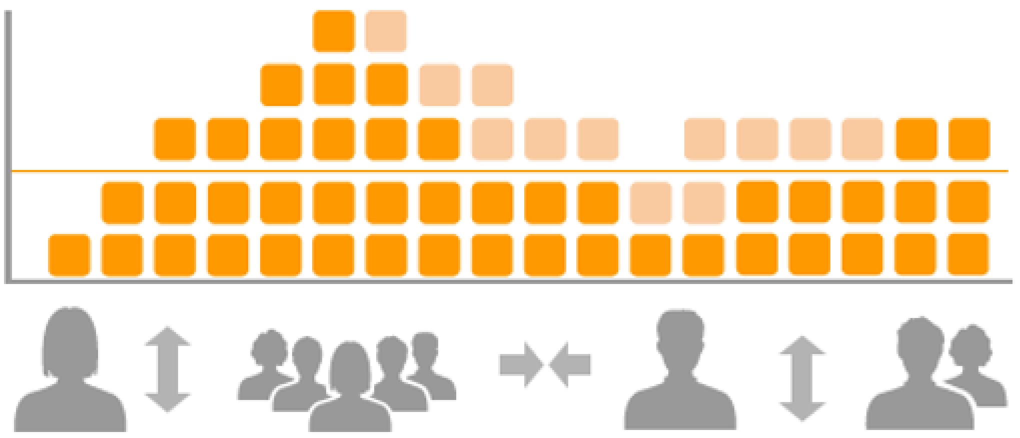


Figure 1. Scaling AWS EC2 Instances to Respond to Demand

The key takeaways from the paper are as follows:

It is possible to automatically scale SAS Real-Time Decision Manager dynamically in the cloud, and it is not hard to implement.

Scheduled scaling of SAS Real-Time Decision Manager for known times of massively increased customer interaction can be used effectively to ensure that your customers have a consistent experience regardless of the promotion or event.

Scaling can be used to reduce infrastructure costs, ensuring that additional capacity is only initialized, and therefore paid for, during peak hours.

SAS REAL-TIME DECISION MANAGER ARCHITECTURE

SAS Real-Time Decision Manager consists of two distinct phases of processing: the design phase and the operational environment or "run-time" phase. The logical architecture of SAS Real-Time Decision Manager reflects these phases and their different requirements.

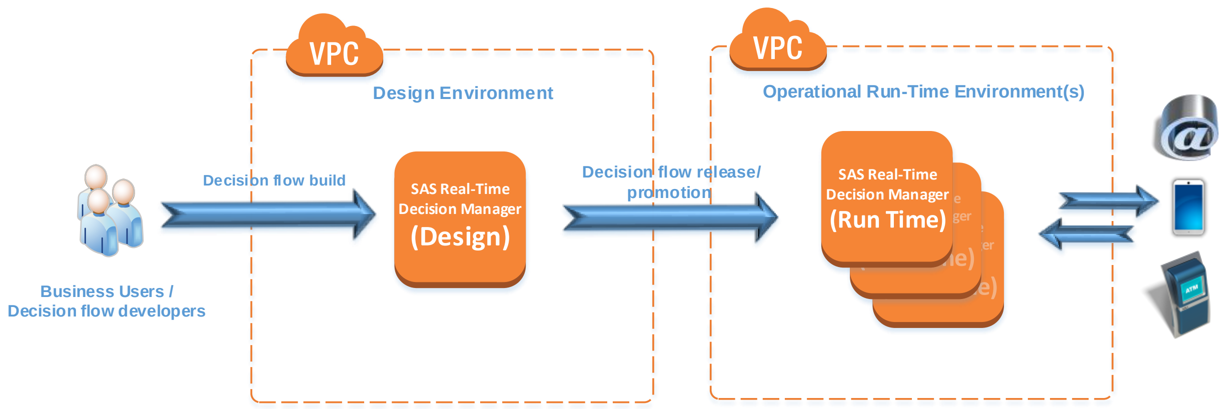

VPC
Design Environment
VPC
Operational Run-Time Environment(s)
Decision flow build
SAS Real-Time Decision Manager (Design)
Decision flow release/ promotion
SAS Real-Time Decision Manager (Run Time)
Business Users / Decision flow developers

Figure 2. SAS Real-Time Decision Manager Logical Architecture

DESIGN ENVIRONMENT

The Architecture for the design phase resembles a typical SAS solution environment. It contains a traditional set of servers (SAS Metadata, SAS Middle-tier, and SAS Compute). These servers are configured to manage the design process only. This is where a campaign is created with a decision flow to make a recommendation based on the customer's eligibility to receive a particular offer. After the design phase is complete, the campaign is promoted to the operational environment.

Because of the similarity of the design environment to a typical SAS solution deployment, we shall not spend any more time describing it here.

RUN-TIME ENVIRONMENT

In contrast, the operational run-time environment is separated, or distinct, from the design environment. This does not have to be a physical or network separation, but, in reality, it should be. It is generally best practice to have a separation of concerns between design and run-time environments, due to sensitivity in terms of availability, outage, and the channel-facing nature of the run-time environment.

The SAS Real-Time Decision Manager Decision Services Engine is a stateless application that is designed to access a common database. It delivers relevant interactive decisions in high-throughput scenarios, based on automated analytical techniques. These features make it an ideal candidate for auto-scaling.

Regardless of the hosting method, workload for the operational run-time environment should be load balanced across two or more servers to accommodate high volumes of requests and provide acceptable response times and higher availability. A multi-tiered architecture is used to distribute functionality across computer resources. This paper demonstrates how you can scale the operational run-time nodes, specifically those running SAS Real-Time Decision Manager Engine and SAS Federation server.

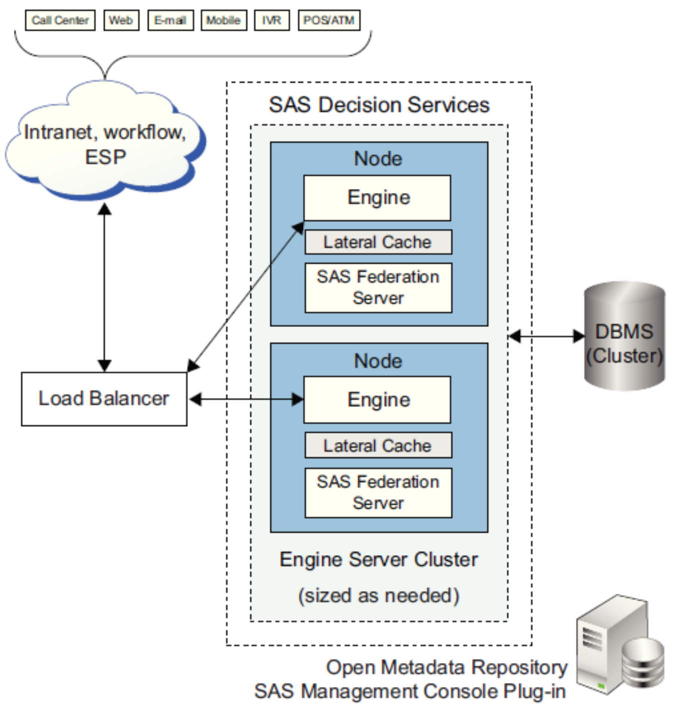


Figure 3. Typical SAS Real-Time Decision Manager Operational Run-Rime Environment

OVERVIEW OF AWS COMPONENTS

For the purpose of this paper, we have assumed a degree of awareness of Amazon Web Services. Therefore, although we leverage and make reference to, we do not plan to cover foundational AWS topics such as VPCs, EC2, RDS, and so on.

AWS ELASTIC LOAD BALANCING

Elastic Load Balancing automatically distributes incoming application traffic across multiple targets, such as Amazon EC2 instances, containers, and IP addresses. It can handle the varying load of your application traffic in a single Availability Zone or across multiple Availability Zones. Elastic Load Balancing offers three types of load balancers that all feature the high availability, automatic scaling, and robust security necessary to make your applications fault tolerant. In this paper, we leverage the Application load balancer.

Elastic Load Balancing provides confidence that applications will scale as needed. With the ability to trigger Auto Scaling for our Amazon EC2 instance fleet based on our own rules, we can ensure that SAS Real-Time Decision Manager will always be ready to serve the next request and minimize infrastructure costs.

The AWS Application Load Balancer operates at the request level (layer 7), routing traffic to targets: EC2 instances, containers, and IP addresses based on the content of the request. It is ideal for advanced load balancing of HTTP and HTTPS traffic. Hence, the decision to leverage its capabilities in this paper. The Application Load Balancer (ALB) provides advanced request routing targeted at delivery of modern application architectures, including microservices and container-based applications.

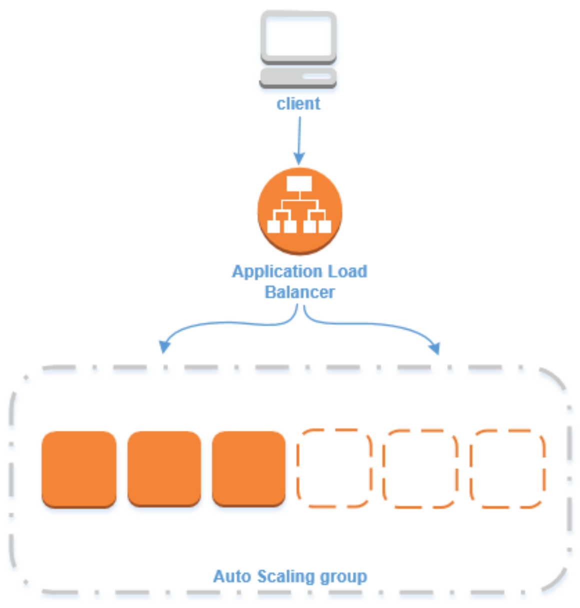


Figure 4. Application Load Balancer (ALB) in AWS

AWS LAUNCH CONFIGURATIONS

When you create an Auto Scaling group, you need to specify a launch template, launch configuration, or an Amazon Elastic Compute Cloud (EC2) instance. Launch templates enable you to reduce the number of steps required to create an instance by capturing all the launch parameters in one resource. A Launch Configuration is a template that an Auto Scaling Group uses to launch EC2 instances. When you create a launch configuration, you specify information for the instances such as the ID of the Amazon Machine Image (AMI), the instance type, a key pair, one or more security groups, and a block device mapping.

AWS AUTOSCALING GROUPS

AWS Auto Scaling monitors applications and automatically adjusts capacity to maintain steady, predictable performance at the lowest possible cost. Using AWS Auto Scaling in this paper, we have set up application scaling for the SAS RDTM engine nodes, as depicted in Figure 1.

AWS Auto Scaling enables you to configure automatic scaling for the scalable AWS resources for your application in a matter of minutes. AWS Auto Scaling uses the Auto Scaling and Application Auto Scaling services to configure scaling policies for your scalable AWS resources.

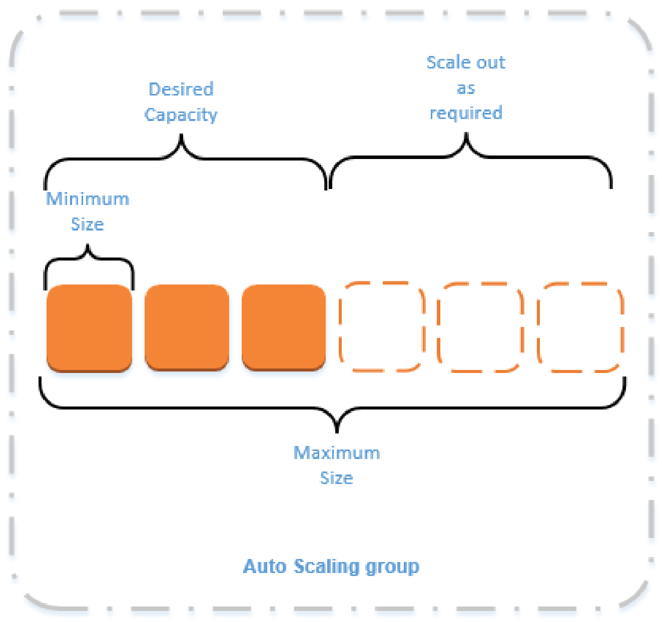


Figure 5. Auto Scaling in AWS

With AWS Auto Scaling, we have created a scaling plan with a set of instructions used to configure dynamic scaling for the SAS Engine nodes. Scheduled scaling can be used to expand services when increased demand is predicted, for example on a Black Friday promotional sale. For simplicity, this is the type of scheduling we will be using in this paper. AWS Auto Scaling creates target tracking scaling policies for the SAS Real-Time Decision Manager Engines in our scaling plan. Target tracking scaling policies adjust the capacity of your scalable resource as required to maintain resource use at the target value that you specified.

THE METHOD

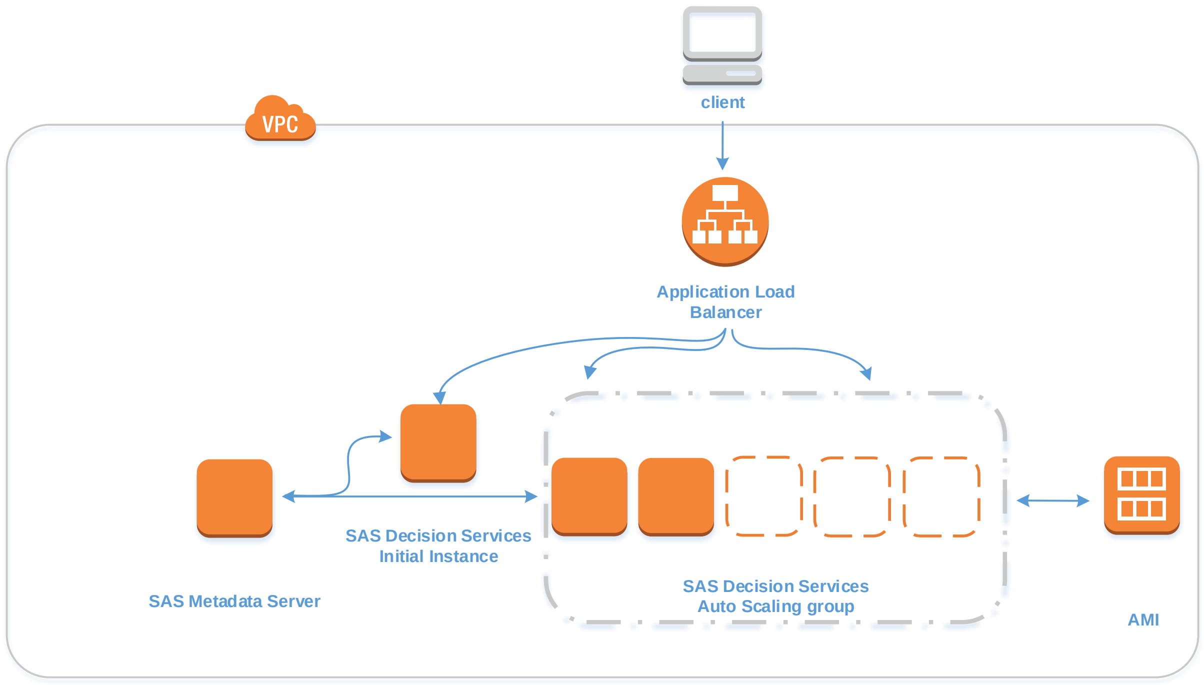

client
VPC
Application Load Balancer
SAS Decision Services Initial Instance
SAS Metadata Server
SAS Decision Services Auto Scaling group
AMI

Figure 6. Auto Scaling Architecture in AWS

This part of the paper provides insight into the configuration in SAS and AWS that is required to provision scheduled auto scaling for SAS Real-Time Decision Manager. Applying these techniques will result in a deployment architecture layout similar to Figure 6.

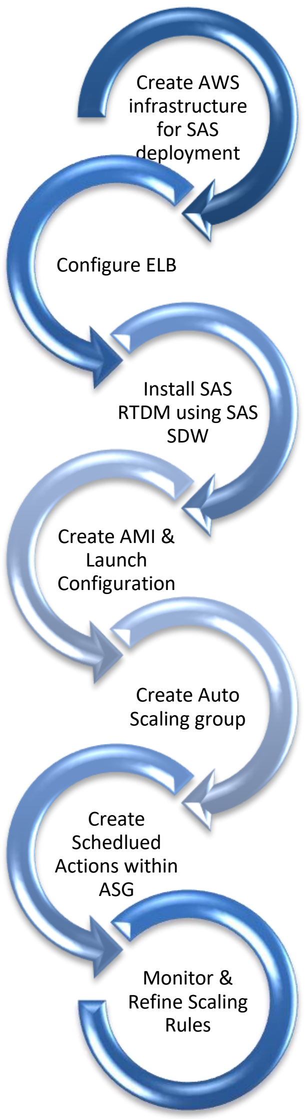

Create AWS infrastructure for SAS deployment
Configure ELB
Install SAS RTDM using SAS SDW
Create AMI & Launch Configuration
Create Auto Scaling group
Create Schedlued Actions within ASG
Monitor & Refine Scaling Rules

Figure 7. Method

CREATE AWS INFRASTRUCTURE FOR SAS DEPLOYMENT

Your first step should be to design the platform so that you can then create the EC2 instances and other infrastructure that are required for the deployment. See the later section on Considerations for SAS Real-Time Decision Manager Deployment in AWS for more details. Create instances for SAS® Metadata and the SAS® Decision Services Engine Server. These will form the core of the environment.

Create an additional instance that will be used to create the AMI snapshot that will form the basis of the auto scaling group. This AMI will be used to drive the auto scaling group that supplement the core SAS Real-Time Decision Manager Run-time environment that will be available regardless of the scaling rules.

An Elastic Load Balancer (ELB) is required, in the form of an Application Load Balancer (ALB). As part of the ALB configuration, you will define a Target Group. You will configure the Decision Execution Services (SAS Decision Services Engine and SAS Federation Server) to use this ALB. This Target Group will allow the number of instances running to change dynamically and this will be transparent to the end users.

DEPLOY SAS REAL-TIME DECISION MANAGER

You aim is to create a launch configuration by capturing an AMI of a freshly built SAS Real-Time Decision Manager Run-Time Engine Node. This engine node will be created via the standard SAS® Deployment Wizard method on the EC2 instance that you have defined. The SAS Real-Time Decision Manager Design Environment is a separate deployment and not covered in this process.

Ensure that the SAS Federation Server and the Decision Execution Services are configured to use the Elastic Load Balancer address. The load balancer will route incoming traffic to the Decision Execution Services between the active nodes. The number of active nodes will vary depending on the Auto Scaling Group schedule and rules.

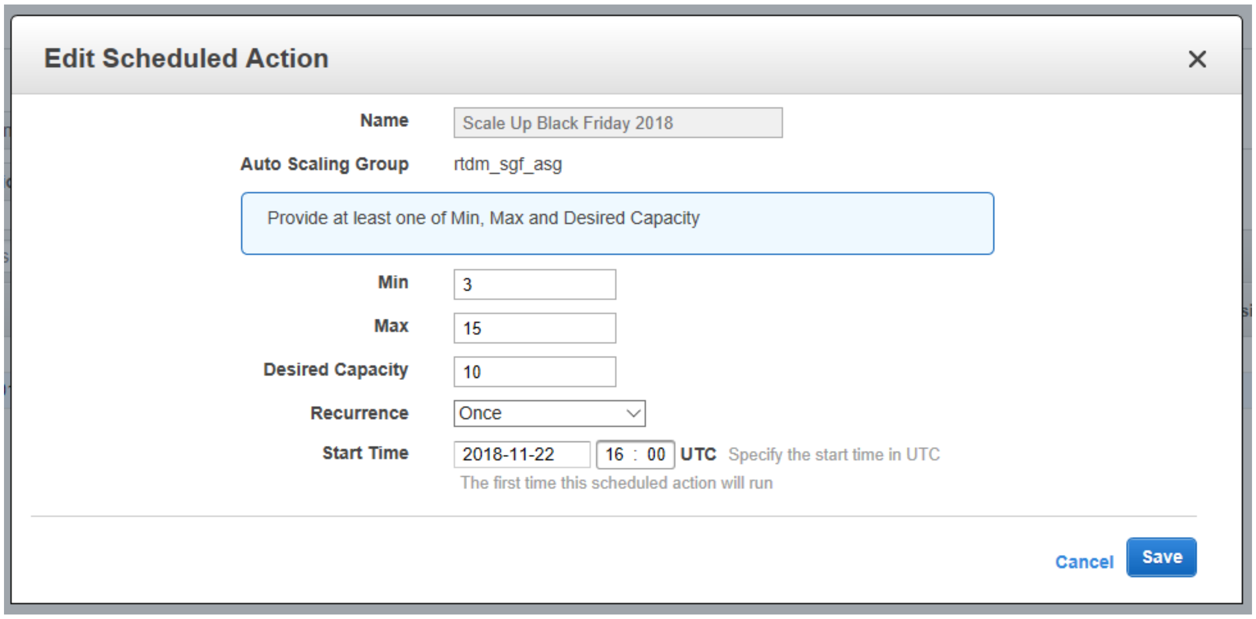
CREATE AMI AND SET UP LAUNCH CONFIGURATION

Create a snapshot of the additional instance. This will create an Amazon Machine Image (AMI), which you will define as the AMI ID in the Launch Configuration. We are using a Launch Configuration for this step, but in theory, you could use a Launch Template instead. Then create a Launch Configuration. Here you choose your AMI and specify the instance type (see the section for Considerations for SAS Real-Time Decision Manager deployment in AWS) and your Launch Configuration name.

CONFIGURE AUTO SCALING GROUP

To configure an Auto Scaling Group (ASG), choose the previously defined Launch Configuration and define the number of initial servers that you want to run. Because we are going to configure schedule based scaling for known increases in traffic, do not configure any scaling policies at this point. Once you have defined the ASG, you need to modify it to associate it with the previously defined Target Group that is associated with the ALB.

Lastly, you need to define the Scheduled Actions within the ASG. Here you can specify the desired number of instances that you want to have running at any time. Take into consideration that instances and applications will require certain start-up times so ensure that you schedule the start before the increased capacity is required. These Scheduled Actions should be defined and monitored as part of a continuous improvement process.


Edit Scheduled Action

Name	Scale Up Black Friday 2018
Auto Scaling Group	rtdm_sgf_asg

Provide at least one of Min, Max and Desired Capacity

Min	3
Max	15
Desired Capacity	10
Recurrence	Once
Start Time	2018-11-22 16 : 00 UTC Specify the start time in UTC

The first time this scheduled action will run

Figure 8. Example Scheduled Action in AWS

CONSIDERATIONS FOR SAS REAL-TIME DECISION MANAGER DEPLOYMENT IN AWS

You should consider the following when designing the SAS Real-Time Decision Manager Run-Time environment for AWS:

The Auto Scaling AMI should contain at a minimum the Java Virtual Machine (JVM) containing the SAS® Decision Services Engine Server and SAS® Federation Server components. You might choose to deploy the SAS® Web Server and configure this to route to local host, or you could route the ALB directly to the JVM.

The instance types that you define in the launch configuration should have enhanced networking integrated, as this provides higher bandwidth, higher packet per second (PPS) performance, and consistently lower inter-instance latencies.

Whether any shared storage is required (for example, on an EFS mount) to support specific federation server usage.

Whether to run the SAS Shared Services and other supporting databases in AWS Relational Database Service (RDS).

FUTURE OPPORTUNITIES

There are several potential opportunities where you can enhance this service in the future. Here are some examples:

Configuring the Auto Scaling Group to use performance metrics in order to drive the number of nodes that are active. These could be based on CPU usage or traffic requests for example.

Automating the build and deployment of the SAS Real-Time Decision Manager environments through Cloud Formation scripts and other automation languages.

Defining a Launch Template instead of a Launch Configuration because it allows multiple versions of a template to be stored at any time. This will provide extra flexibility when updating SAS Real-Time Decision Manager software.

Similar approaches could be explored for solutions that rely on massively parallel processing (MPP) architecture topologies such as CAS in SAS Viya.

CONCLUSION

This paper provides insight into the technical configuration of a schedule-based, auto scaling process of a SAS Real-Time Decision Manager run-time Environment in Amazon Web Services. AWS contains the cloud enablement technologies to automatically scale SAS Real-Time Decision Manager, a key part of your Interactive customer channels. This enables you to scale up or down your SAS Real-Time Decision Manager run-time environment to meet your needs and minimize your infrastructure costs while ensuring that your customers receive a consistent service.

REFERENCES

Amazon Web Services – Auto Scaling Overview:
https://docs.aws.amazon.com/autoscaling/ec2/userguide/what-is-amazon-ec2-auto-scaling.html

Amazon Web Services – Auto Scaling Getting Started Tutorial:
https://docs.aws.amazon.com/autoscaling/ec2/userguide/GettingStartedTutorial.html

Amazon Web Services – Enhanced Networking:
https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/enhanced-networking.html

SAS Online Documentation: http://support.sas.com/documentation

To learn more about SAS Real-Time Decision Manager, download white papers, view screenshots and see other related material, please visit http://www.sas.com/realtimedecision/

ACKNOWLEDGMENTS

I would like to thank Ekaitz Goienola for his support whilst writing this paper and Andrew MacFarlane for sharing his passion about technical architecture and deployment.

CONTACT INFORMATION

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

Chris West
Wittington House
Henley Road Marlow
Buckinghamshire
SL7 2EB,
United Kingdom
chris.west@sas.com