Paper 3174 -2019

Lessons Learned Architecting a Modern Data Analytics Platform in the Cloud Using MapR and SAS® Viya®

Shane Gibson, PitchBlack

ABSTRACT

During 2018, I was the Technical Lead on an Analytics 2.0 project for a large New Zealand Government organization that was deploying a MapR converged data platform in Microsoft Azure and a SAS® Viya® and SAS® 9.4 Platform via SAS® Global Hosting in Amazon Web Services (AWS). This presentation covers the technical architecture that was defined for the integrated platforms and the lessons learned during the architecture of the platforms.

INTRODUCTION

This paper outlines the lessons learnt architecting a MapR converged data platform in Microsoft Azure and a SAS® Viya® and SAS® 9.4 Platform via SAS® Global Hosting in Amazon Web Services (AWS) and provides recommendations on how to mitigate some of issues that were encountered.

The papers covers the following key areas:

1. Implementing multi-cloud platforms;
2. SAS Viya multi-tenancy architecture design;
3. Implementing SAS Viya and SAS 9.4—why you need to consider the tradeoffs;
4. Defining authentication and authorization in a managed platform environment;
5. Data integration patterns between SAS Viya and a MapR data lake or data vault; and
6. The automation required to achieve a DataOps vision.

The presentation itself provides a number of architecture diagrams and more detailed discussion on the issues encountered.

BACKGROUND

The Government organisation had a legacy SAS data platform which was developed over 20 years ago and currently operating based on a SAS 9.4 technology stack. They also had a newer data platform that was implemented based on an Oracle technology stack and was planned to replace the legacy SAS platform. However, after a number of years of implementation, the Oracle project was completed without fully replacing and decommissioning the legacy SAS platform, resulting in both platforms being operational and requiring ongoing maintenance. In addition, the Oracle data platform was coupled to data in the legacy SAS platform during its implementation.

As part of a digital transformation program tasked was driving large scale organizational and technology changes, the organisation decided to replace both the legacy SAS and Oracle platforms with a new Analytics 2.0 platform based on MapR, SAS Viya and SAS 9.4.

The initial phase of the Analytics 2.0 project implemented a data lake and operational reporting for a new source system using the MapR and SAS capabilities, however a number of challenges with this implementation resulted in the inability to scale the platforms to a larger number of source systems or users in subsequent phases. This included the coupling the new MapR/SAS platform to both the legacy SAS and Oracle platforms.

I joined the organisations project team to assist in architecting a scalable solution utilising MapR and SAS Viya as part of the second implementation phase.

Learning	Recommendation
Couple to your legacy environment at your risk	If you have no choice lightly couple, aka via micro service patterns to enable future decoupling
Decommissioning the last 20% of a legacy solution is hard	It needs to be decommissioned otherwise you will bleed ongoing resources maintaining both
When you incur technical debt, it has future consequences	Keep architecture decisions and technical debt registers so they are visible and can be remediated in the future
Evolving architecture is great, but there is risk of tactical solutions	You need an initial blueprint (hypothesis) and regular validation to ensure it evolves into a scalable architecture
A data lake isn't one solution to rule them all, especially if your replacing operational reporting to thousands of information consumers	You still need to repeatably transform data to provide trusted information to information consumers, so ensure you architect this as part of your solution
If you're not experienced at systems integration, don't pretend you are	Engage somebody for their systems integration expertise, ideally one of the vendors who own the solution

Table 1. Lessons learned from previous projects

MULTI CLOUD INTEGRATION

The Government organisation had a cloud first strategy and a multi-cloud strategy which resulted in both Microsoft Azure and Amazon Web Services (AWS) being approved cloud infrastructure platforms. The second phase of the project moved from a model where the platforms were managed by the Government organisation to a model where the software vendors provided managed platforms. MapR selected Microsoft Azure as their preferred cloud platform and SAS Global Hosting selected AWS.

Learning	Recommendation
Multi-cloud strategy constantly challenged by organizational behavior	Make sure the strategy is documented and agreed by those that could challenge it, "urban legends" won't cut it
Contract negotiations may take longer than provisioning the platform	Define success/acceptance criteria early before starting negotiations

Learning	Recommendation
Worry about where the highest volume of data moves	It is probably between your systems of record and your data lake, and between your data lake and your data scientists tools
Worry about where the highest volume of adhoc users and the highest volume of their data are co-located	It's probably between your self-service analytics tool for citizen data scientists and your transformed data
Obtaining access to production-esk data volumes is difficult	If necessary, fall back to using open data that has representative data volumes
Watch out for network trombone's	Validate the network at a detailed level to ensure it is configured the way expected

Table 2. Lessons learned Multi-Cloud Integration

MULTI-TENANCY

SAS Viya can be deployed as a managed platform via SAS Global Hosting utilising a multi-tenancy architecture. In a multi-tenancy architecture, a provider manages one or more tenants within a single deployment. The essential characteristics of multi-tenancy are separation and sharing.

Many SAS Viya components are shared across tenants, for example, applications are shared across tenants. Some components have a dedicated instance for each tenant, for example, each tenant has its own dedicated CAS controller.

Multiple tenancies can be used to efficiently manage environment separation within the SAS Viya platform, while retaining a single installation of the core SAS software components.

Learning	Recommendation
Multi-tenancy is a great way of managing environment separation	Use separate tenancies for each environment, but architect how you will share the data across tenancies where practical
Multi-tenancy support is enabled during provisioning and cannot be changed	Enable multi-tenancy by default, you cannot turn it on later
Multi-tenancy increases complexity of the environment	As SAS Global Hosting administer the managed platform, the level of complexity is irrelevant to you as the customer
SAS Solutions can be isolated in a separate tenancy	If you are utilising a SAS Viya based solution, for example the SAS Fraud solution, then deploy it in its own dedicated tenancy

Learning	Recommendation
SAS 9.4 does not have the concept of tenancies	You will need to define an architecture that provides a similar model for sharing and separating the things you care about
You cannot stand up and burn down tenancies on demand	You need to put thought into the number of tenancies required and the applicable use cases

Table 3. Lessons learned Multi-Tenancy

SAS VIYA VS SAS 9.4 TRADEOFFS

SAS are undertaking a gradual transition from the legacy SAS 9.4 architecture to the new SAS Viya architecture.

SAS are progressively re-architecting many of the SAS 9.4 components and solution to utilise the SAS Viya architecture, for example SAS Data Management. A number of SAS 9.4 solutions are being replaced with new solutions built upon the SAS Viya architecture, for example the SAS Fraud solution and SAS Enterprise Miner. While a number of the other SAS 9.4 components are being updated to integrate with SAS Viya components while retaining their SAS 9.4 architecture, for example SAS Enterprise Guide. Some other components, for example SAS Studio, have both a SAS 9.4 version and a SAS Viya version (in fact SAS Studio had two SAS Viya versions available in 2018).

Some of the new SAS Viya components and solutions do not yet have feature parity with the legacy SAS 9.4 versions. This forces a number of decisions on what version of each component and solution should be architected into the platform.

Learning	Recommendation
Migrating from the SAS 9.4 to SAS Viya version is not trivial	Implement the SAS Viya version where possible
SAS Viya versions often have limited features	Understand the core use cases and map to the features required, to see if the SAS Viya version is viable
Enterprise guide is still the equivalent of the swiss army knife and users still love it	SAS Studio and the planned notebook capability will eventually replace EG, but until then you will probably need to include it
SAS Desktop tools provide possible ongoing compatibility issues with a SAS Managed Platform	Provide the tools on virtual desktops, and if possible, include in the SAS Global Hosting managed services

Table 4. Lessons learned in the SAS Viya vs SAS 9.4 Tradeoff

AUTHENTICATION AND AUTHORISATION

SAS Viya, SAS 9.x and MapR all have a number of components and a number of the component have unique authentication and/or authorization models.

SAML was identified as the preferred authentication model, to ensure internal user names and passwords were not exposed or stored in the managed platforms. However, SAS 9.x and MapR were unable to support this model for the majority of their components.

The creation of unique user names and passwords on each managed platform was deemed to be a difficult model to scale to thousands of information consumers. A hybrid authentication model was adopted.

Learning	Recommendation
You may need multiple authentication approaches depending on the solution capabilities and user roles	Create and validate persona's early so you can describe the authentication approach for each
You need to manage security of non-production data even if nobody else in the organisation does	Plan to secure and/or obfuscation data in your non production environments
Everybody has an opinion on what data should be secured	Decide if you core authorization principal is visible by default or hidden by default

Table 5. Lessons learned designing Authentication and Authorization

DATA INTEGRATION PATTERNS

Another challenge encountered was how to do you identify the best way to integrate the data between the many components being used in MapR and the many components in SAS Viya and SAS 9.x.

To resolve this challenge, we used a combination of use cases to identify the potential data flows needed and data integration patterns to identify and select the best integration pattern for each use case.

Learning	Recommendation
When there are lots of moving parts and lots of options you may not know which patterns to focus on	Create use cases to identify the main flows of data and then design for those
Your vendor may not understand using a pattern approach	Co-design the patterns together, co-design provides a number of additional benefits
You won't know that your patterns will work until you prove them	Do a Proof of Concept (PoC) for each integration pattern
When undertaking performance testing with volumes your pattern may not survive	Ensure the PoC is proven with representative volumes of data, use large open data sets if necessary

Learning	Recommendation
Your first pattern may not be production viable	Ensure you always have an alternative pattern defined for each use case

Table 6. Lessons learned using data patterns for integration

DATAOPS

DevOps is an approach to software development that accelerates the build lifecycle (formerly known as release engineering) using automation**.** DataOps uses automation to reduce the end-to-end cycle time of data analytics, from the origin of ideas to the literal creation of data, visualisations and models that create value. So while DataOps leverages a lot of the DevOps principals and patterns of automation, it is about automating everything you can, not just the infrastructure and code.

One of the challenges is that a lot of software solutions that have been around for a while are not architected to enable a DataOps approach. For example the ability to check-out and check-in components of a dashboard as you incrementally build it.

Learning	Recommendation
DataOps is new, vendors who say they have automated solutions still often use manual techniques to make changes as they haven't created full automation yet	Validate early what is actually automated with the vendor
Manual "changes" to data and configuration is dangerous and unsustainable	Changes should be deployed as automated code. If not available yet, at least the manual processes should follow a version control check-in / check-out approach
Most Visualisation tools don't support a DataOps approach yet, i.e. you can't use git to create and maintain visualisation code objects	Use git during the release / promotion process
Contracts and or technology doesn't support the burn down and stand up of environments on the fly	Identify this early with your vendor and then design your architecture with these constraints in mind
Contracts often don't support auto-scaling of resources, or the customers financial models cannot support variable cost every day	Determine the financial model the customer needs to operate in and architect to match that model, but also enable change to the model in the future

Table 7. Lessons learned implementing a DataOps approach

CONCLUSION

Architecting a multi-cloud solution utilising both SAS Viya and SAS 9.4 as well as MapR and Attunity is challenging. Engage and utilise experts from each vendor on the best way to architect, provision and manage the respective platforms.

Where possible move to a managed cloud platform, where the vendors are accountable for provisioning and maintaining their own software solutions.

CONTACT INFORMATION

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

Shane Gibson
PitchBlack
Shane.gibson@pitchblack.nz
https://pitchblack.nz