Paper SAS 3198-2019

Full Stream Ahead: Design and Engineering Patterns for SAS® Event Stream Processing in the Cloud.

Alessio Tomasino and Louis Katsouris - SAS UK and Ireland Professional Services

ABSTRACT

In recent years, there has been a fundamental shift in the requirements of IT organizations to seamlessly manage and deploy their software estates in the public and private clouds. SAS® Event Stream Processing is at this forefront of this revolution, providing the benefits of a cloud-ready solution coupled with powerful analytics at scale when used alongside AI, machine learning, IoT, and real-time data streaming. This paper discusses advanced design, deployment, and engineering patterns for SAS Event Stream Processing in the cloud and its integration with key third-party technologies such as Apache Kafka.

INTRODUCTION

Looking at the IT industry in the past few years, the increase of cloud-based technologies and DevOps best practices has resulted in IT organizations reducing lead times for provisioning infrastructure and services. As a result, the time-to-value ratio for software investments has been greatly optimized.

The SAS Viya generation of software has been engineered from the ground up to integrate and leverage cloud-based technologies and DevOps frameworks.

This paper intends to demonstrate how SAS Viya leverages these technologies to deploy advanced end-to-end architectures, which include immediate provisioning of a clustered deployment of the SAS ESP server, failover capabilities, and zero downtime updates, all of which leverage the powerful orchestration capabilities of Kubernetes.

The result is faster time-to-market and a reduced total cost of ownership for mission critical systems, whilst providing enterprise-grade resilience.

Although these technologies provide exciting capabilities, they are still embryonic, and industry best practices are yet to emerge. This paper aims to explore the boundaries of what is technically feasible.

This paper is organized as follows:

- We outline the fundamental technological advances of DevOps, cloud, and containers, and the impact these advances have on business results.
- We discuss the architecture of SAS ESP and how it naturally embraces modern technical developments.
- We articulate how SAS ESP on containers can provide significant benefits to organizations and can open the door to adoption of autonomous computing through use of a custom Kubernetes Operator.

THE DEVOPS REVOLUTION

DevOps (a term derived from development and operations) is a set of practices that facilitates the collaboration of IT professionals involved in the creation, testing, deployment,

monitoring, and automation of software and infrastructure implementations and subsequent maintenance.

DevOps applies a layer of best practice to remove the risk involved in the deployment of software and infrastructure. This layer includes technologies for bundling hardware components, software, and services, and orchestrating them into containerized units. This containerization of applications-as-a-service is facilitated by cloud providers through the availability of the supporting technologies for enabling containerization and automation within a change audited environment.

CLOUD PROVIDERS

SAS is supported on cloud providers because the cloud provider enables functional equivalence to bare metal infrastructure. SAS is therefore independent from the cloud provider's hypervisors, which is demonstrated by multiple customers already using SAS software on various cloud providers such as AWS, Azure, and GCP, as shown In Figure 1.

For the purposes of this paper we will demonstrate our implementation using Microsoft Azure.

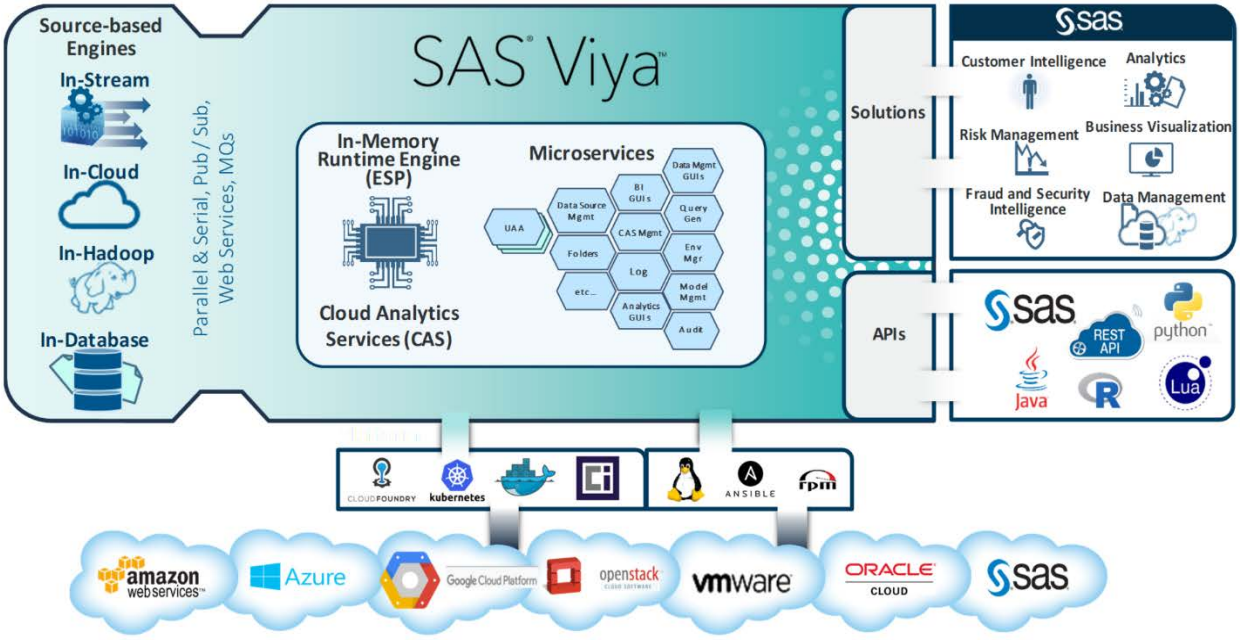

Figure 1. SAS on the Cloud

ANSIBLE, DOCKER, AND KUBERNETES

For the purposes of this paper we will be using an industry standard definition of a container, as shown in Figure 2. Docker software provides an operating-system-level form of virtualization that bundles together inter-related software components (such as applications and supporting libraries) into a single package known as a container. These containers can then be executed against a single operating system kernel, which makes them more streamlined than traditional virtualization technologies. Docker and Kubernetes play the role of container builder and container orchestrator respectively. Ansible is used for the deployment of containers and the automated build of infrastructure services within the cloud as an IaC (infrastructure as code) deployment tool.

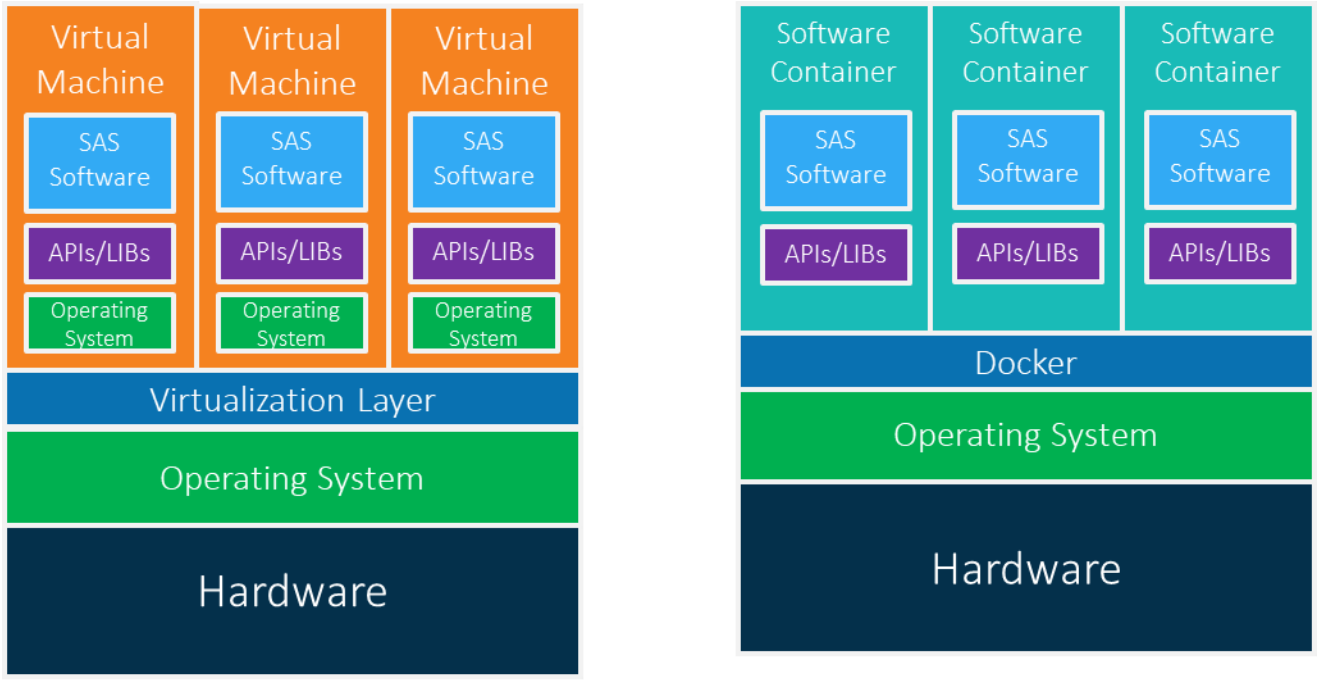

Figure 2. Containers Versus Virtualization

Docker provides the framework to build container images that are stored in the container registry, which is a service for storing and sharing container images. Container images become containers at run time.

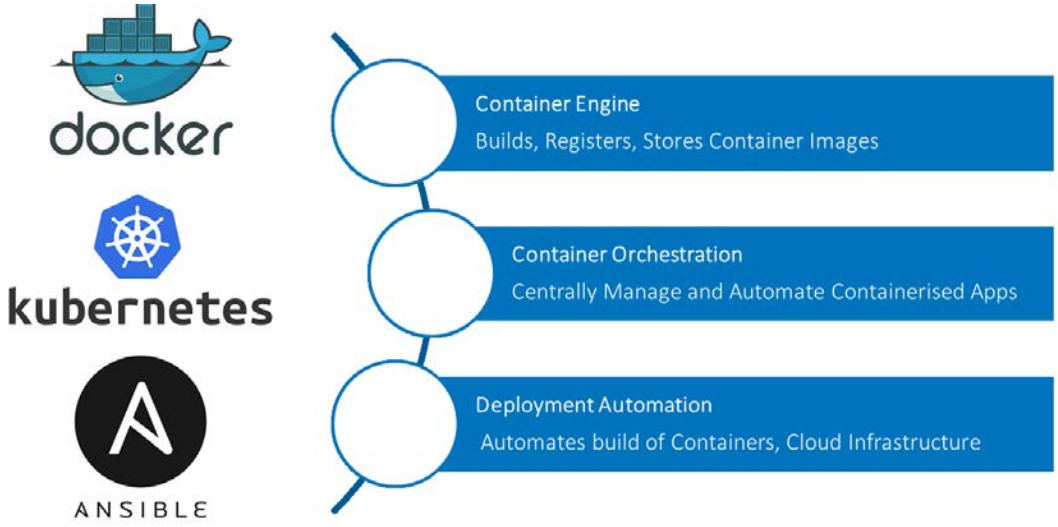

Figure 3 - Relationship Between Docker, Kubernetes, and Ansible

The containers themselves contain the application software. In this case, we have built containers for various SAS applications by using the SAS Viya Container Recipes GitHub project.

During the build of a Docker image, the execution of the commands in the Dockerfile adds layers on top of each other to build the container image. Although manually creating

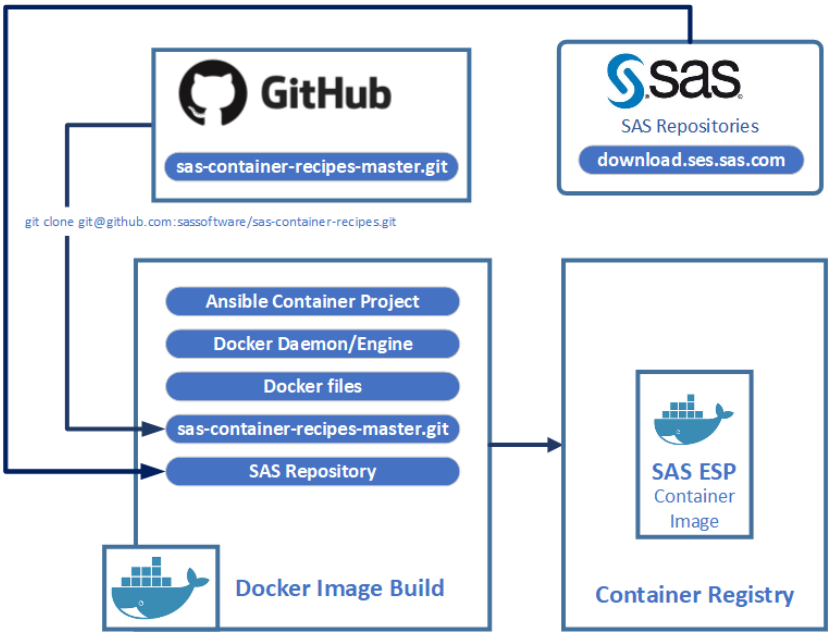

Figure 4 The Docker Image Build Process

a Dockerfile can involve many steps, we used the Ansible Container project to create and customize the container images. In particular, we used the project to add the relevant Kafka libraries that are required for integration with the Kafka message bus.

The output of this process is the Docker image that is stored in the container registry service hosted on Microsoft Azure. After the Docker images are in the registry, Kubernetes takes over. Kubernetes is essentially a portable, extensible, open-source container orchestration tool that manages the containers as part of the deployment.

Kubernetes allows us to deliver portable deployments across multiple cloud providers, preventing us from being locked in to a specific cloud vendor. The configuration capabilities of Kubernetes enable us to deploy SAS services in a repeatable manner, which enables multiple environments (Test, Dev, Prod) to be seamlessly created from templates (called manifests). The manifests execute one or more containers in the smallest execution unit of Kubernetes, called a pod.

We also explored and implemented advanced orchestration concepts for SAS ESP server by using a Kubernetes Operator. A Kubernetes Operator is a method of packaging, deploying, and managing a Kubernetes application in an autonomous manner by using Kubernetes APIs. Using a Kubernetes Operator enables you to engineer smarter applications in which backup, recovery, self-healing, and zero downtime updates can be mostly automated, which in turn increases reliability and enables repeatable error-free execution. The Kubernetes Operator runs in a separate pod in the same Kubernetes namespace. For this paper, the SAS Professional Services division developed a custom Kubernetes Operator called SAS ESP Operator that understands the logic of the main operations performed on a SAS ESP server.

In this design we leveraged the Azure Kubernetes Service (AKS) cluster, but similar Kubernetes services are offered by other cloud providers. The provisioning of the Kubernetes cluster is automated by using an Ansible playbook, which leverages the Azure-specific module for use as infrastructure as code.

Kubernetes exposes a set of APIs to describe an application deployment, but it uses the Docker run-time environment to run the containers on the Kubernetes nodes.

The architecture shown in Figure 5, contains the AKS cluster, which consists of several nodes.

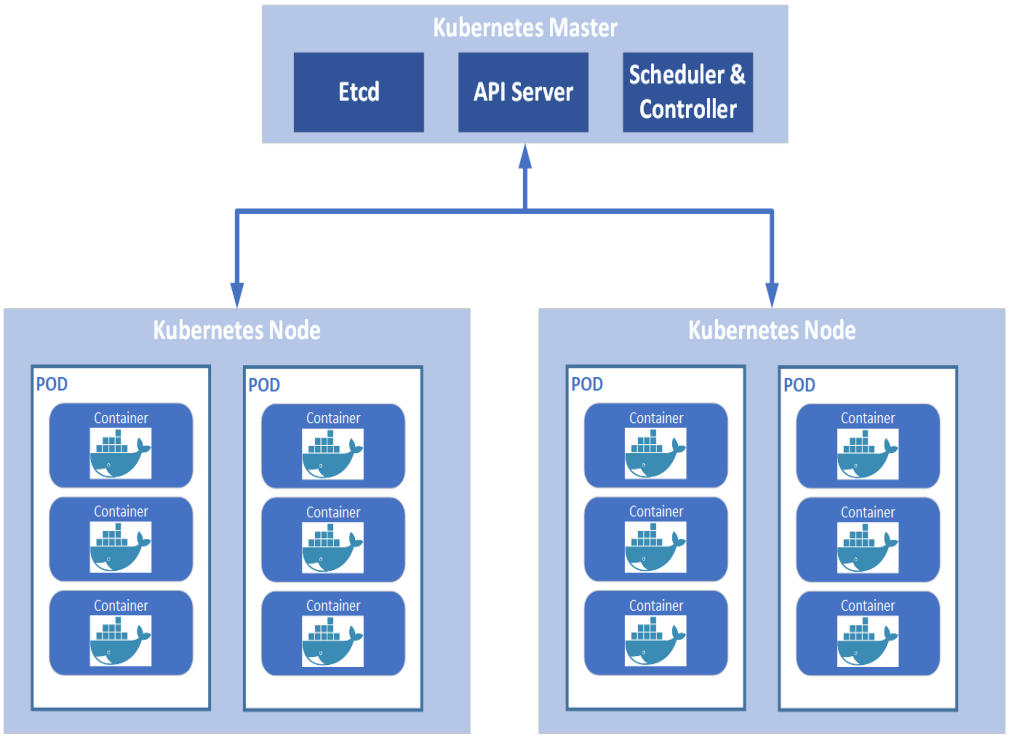

Figure 5 Kubernetes Architecture

TIME TO MARKET

Hardware provisioning and software deployment, maintenance, and configuration have sometimes been perceived as complexity factors, with significant cost associated with them in terms of technical skills and effort required.

The impact of DevOps, cloud computing, and containerization on these factors has been revolutionary. Businesses can now decrease the time to market of IT services that are supported by an effective adoption of these technologies and methodologies, providing them with a competitive advantage over rival organizations.

SAS is a global leader across multiple markets and industries. With open-source technologies becoming more accessible, SAS is aligned with our customers and their adoption of DevOps tools and cloud platforms by integrating with the best of breed in this space, including AWS, Azure, GCP, Kubernetes, Ansible, Terraform, Packer, and Docker. This integration then allows for swifter and more manageable deployments of SAS in the cloud.

Various initiatives and projects have spawned from this integration, such as SAS Viya Container Recipes, which is a GitHub project consisting of a collection of recipes and other resources for building containers that include SAS Viya software.

SAS EVENT STREAM PROCESSING

SAS ESP is a real-time analytical engine. It can process streaming data in-memory, analyze and make sense of millions of events per second and detect patterns of interest as they occur.

From a technical perspective, SAS ESP is a lightweight application that can run with a minimal infrastructure foot print. This makes it an ideal candidate for containerization.

The SAS ESP use cases vary considerably, such as these examples:

Internet of Things (IOT) space – analysis of event data from continuously streaming sensors in SMART devices and IOT-connected vehicles

Manufacturing - connection to industrial sensor devices for measuring threshold alerts for mission critical infrastructure in the oil industry

Retail – analysis of text analytics from Twitter feeds and harvesting of clickstream data from corporate websites

For the purposes of this paper, the architecture demonstrates SAS ESP integration with the Kafka message bus. This is a pattern we used for production designs used by many organizations that use SAS ESP.

The SAS ESP technology itself consists of six key components shown in Figure 6:

1. ESP Engine – A server process that acts as a container for executing ESP projects.
2. The ESP Project. A contained set of logic that executes one or more continuous queries operating on event stream data.
3. Continuous Query – A query over a stream of data that executes continuously over streaming data, enabling execution of analytical operations against the data in real time.
4. Input adapters and connectors - Connectors and adapters use the publish/subscribe API to interface with a variety of communication interfaces, clients, and drivers.
5. ESP windows. A continuous query contains one or more source windows and one or more derived windows. Windows are connected by edges, which have an associated direction depending on whether data is streaming in or out of the ESP service.

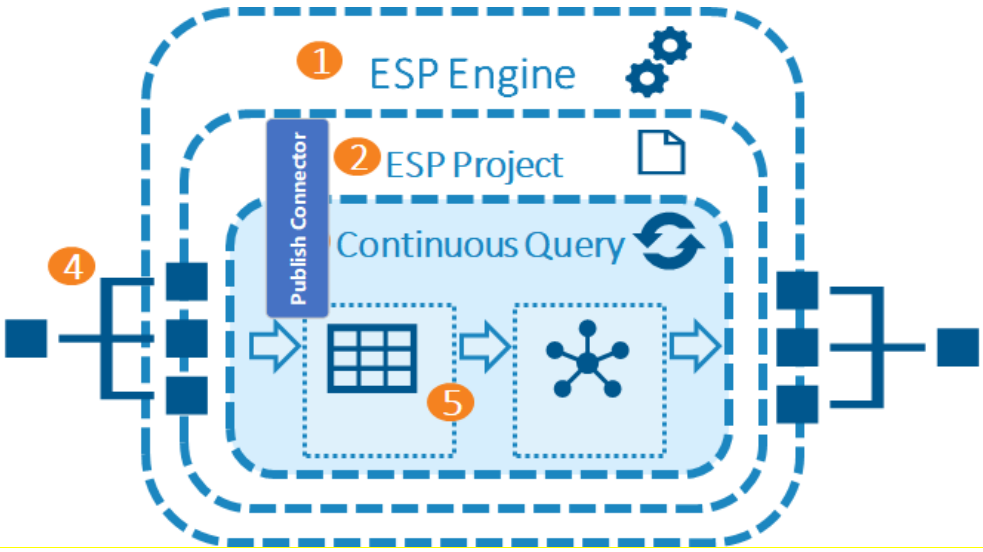

Figure 6. SAS ESP Functional Overview

Now that we have discussed the fundamentals of SAS ESP, we can examine advanced topics: 1+N failover and integration with the Kafka message bus.

SAS ESP AND KAFKA INTEGRATION

As part of the architecture shown in Figure 7, SAS ESP integrates with a Kafka message bus through the publish and subscribe APIs that are provided by specialized SAS ESP Kafka connectors.

The Kafka connectors for SAS ESP are underpinned by key Kafka run-time libraries that you must incorporate into your SAS ESP container image, as discussed later in this paper. In addition, the ZooKeeper libraries must also be included.

The Kafka libraries provide the integration layer between SAS ESP connectors and Kafka. The ZooKeeper libraries enable the SAS ESP servers to monitor the presence of each other in the failover group by using a ZooKeeper Watcher object.

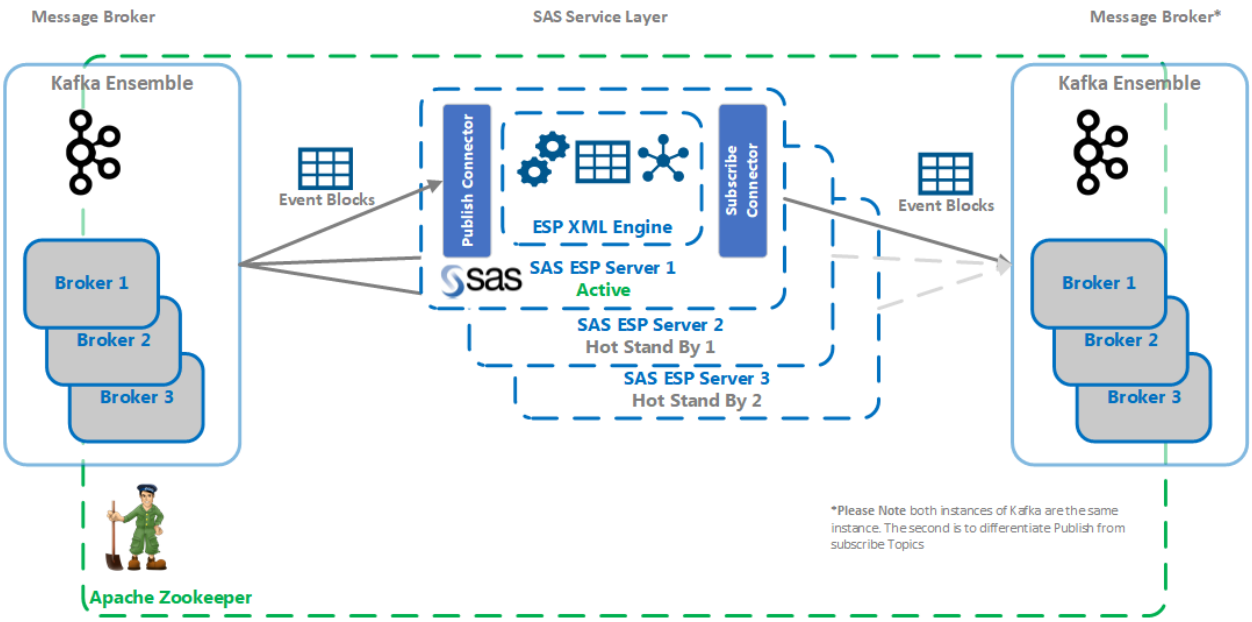

Figure 7. SAS ESP Integration with Kafka

ESP 1+N FAILOVER

The 1+N failover model is a proven industry architecture pattern that provides stable resilience. In this case we are using three instances of SAS ESP to form a cluster for the 1+N pattern. The benefit of this configuration over a two-instance cluster is that three instances prevent a single point of failure when one instance goes down.

As previously mentioned, ZooKeeper libraries are required to enable the coordination of interactions between Kafka and SAS ESP. This is essential for the 1+N failover model.

The 1+N failover process uses in-memory synchronization across all the instances of SAS ESP across the cluster. The loss of the active node implies that one of the standby nodes will acquire the role of active instance and will seamlessly continue the processing of streaming data, as illustrated in Figures 8 and 9.

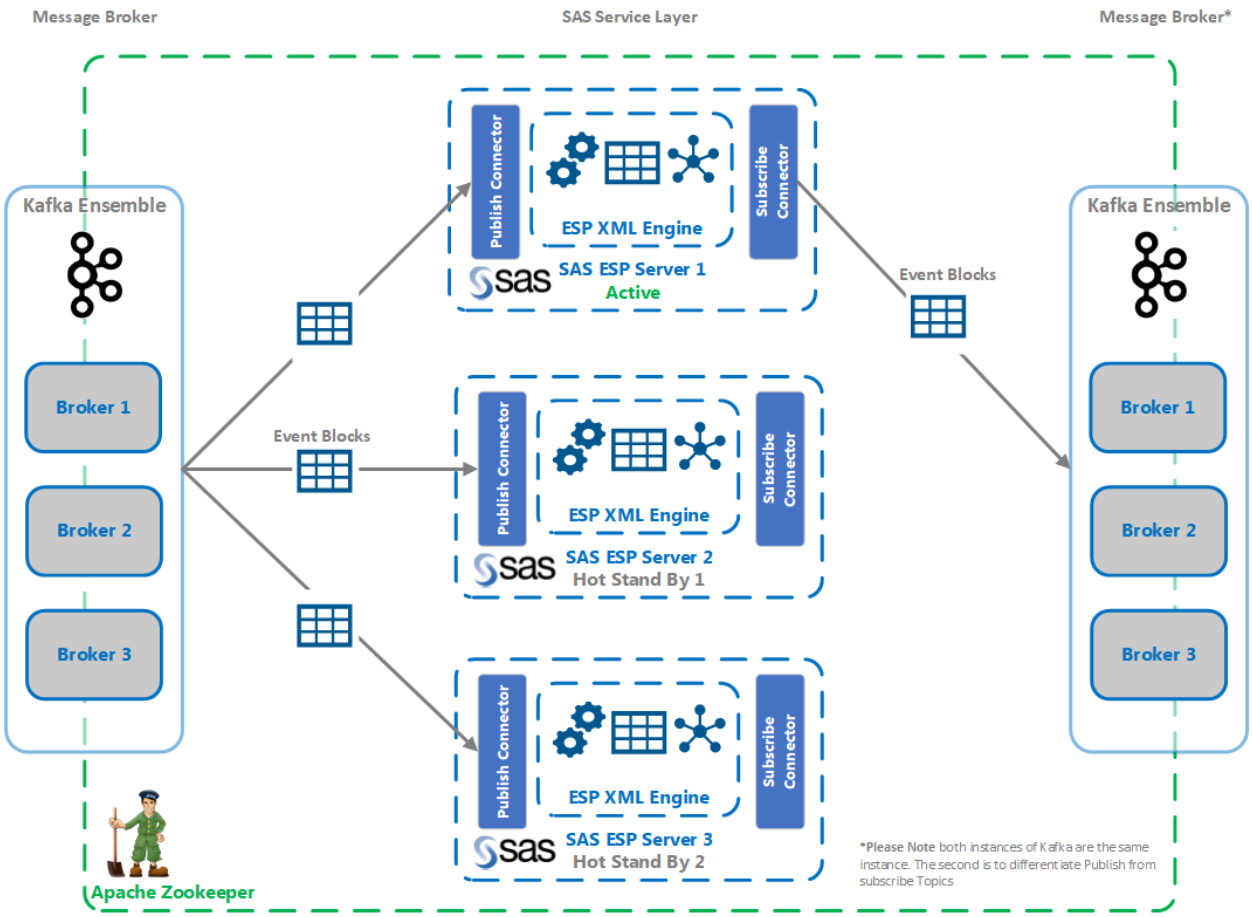

Figure 8. Kafka 1+N Failover Technology

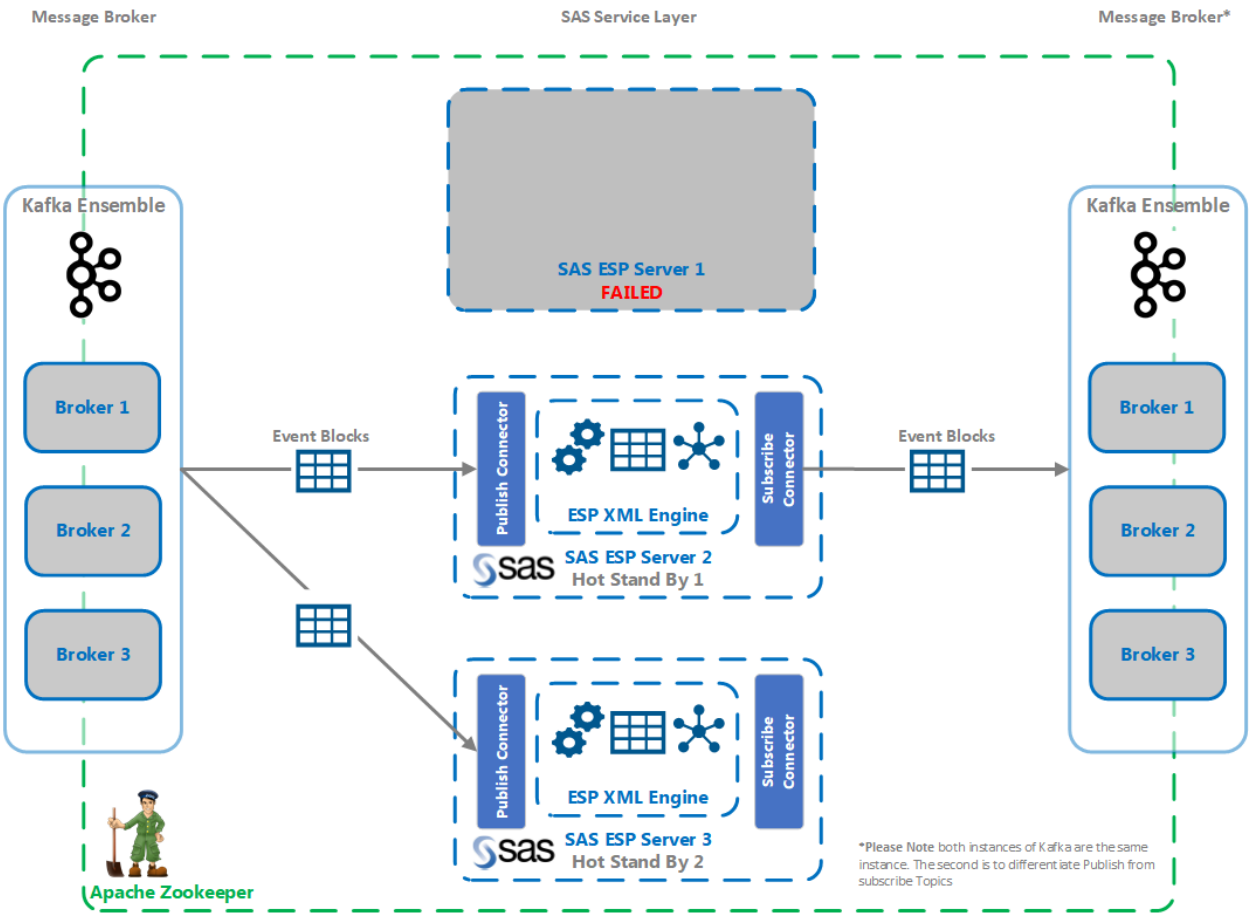

Figure 9. Kafka 1+N Post-Failure

SAS ESP ON CONTAINERS

The rest of the paper is dedicated to discussing in more detail the technical aspects and operational advantages of a deployment of SAS ESP using the technologies described in the first section of this paper.

In particular, we will discuss the following topics:

- Automated build of SAS ESP containers
- Zero downtime update of SAS ESP
- Failover and recovery of SAS ESP containers

Figure 10 shows a SAS ESP deployment in Microsoft Azure.

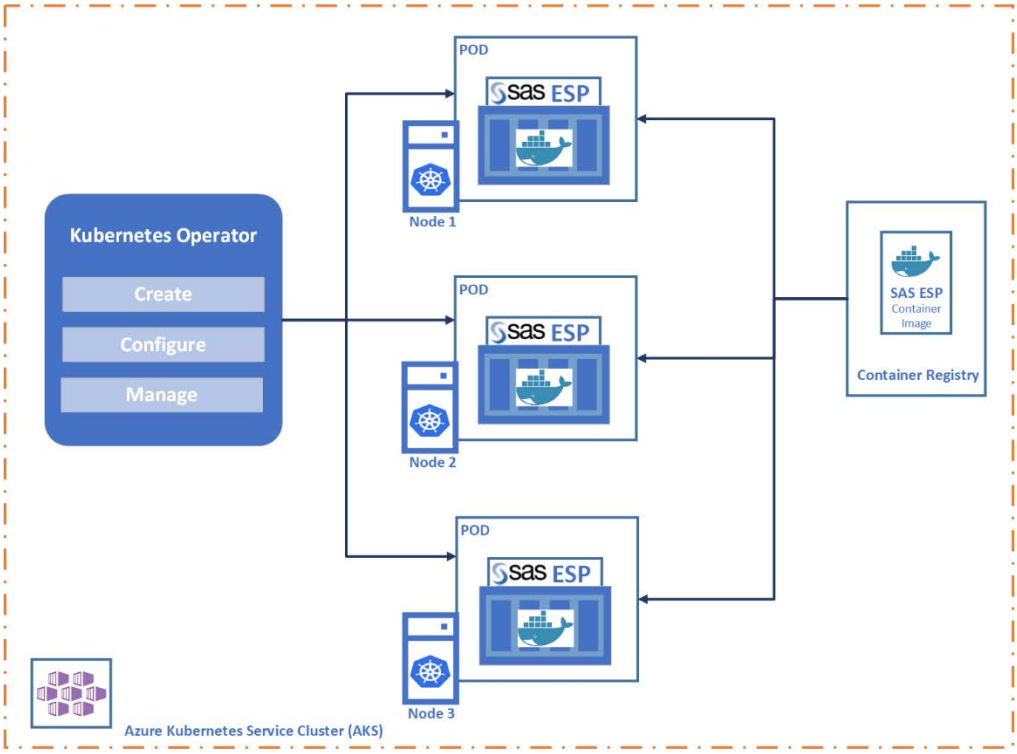

Figure 10 - SAS ESP on Containers

SAS ESP BUILT AS A CONTAINER IMAGE

The pipeline in Figure 4 describes the stages involved in building a SAS ESP Docker image. We used the SAS Viya Container Recipes GitHub project to create a SAS ESP Docker image containing the libraries needed to use SAS ESP connectors and adapters to connect to Apache Kafka. We also configured the SAS ESP containers to map a shared volume on Azure Files, which enables us to share a snapshot of the SAS ESP memory. This facilitates faster start-up during recovery of the ESP engine and as well as provides alignment across multiple engines running on different Kubernetes pods. To achieve this goal, we used the Ansible Container project, which enables you to add Ansible roles to configure the Docker images. This ensures that the container runs a copy of SAS ESP Server that contains the correct version and location of the required libraries every single time, without performing any additional configuration.

Multiple pods are immediately scheduled by Kubernetes according to the Kubernetes deployment options, which uses the Docker images available in the container registry. In our case, we set the value of replicas to three in order to launch three pods running SAS ESP Server containers.

At run time, the SAS ESP Server container performs the following logic at start-up:

1. If a recovery point is available, load it from shared storage.
2. If a recovery point is not available, load the project without recovery.
3. Run the SAS ESP project.

Each of the three running SAS ESP instances then acquire a role of either active or standby. Only the active instance writes to the outbound destination specified in the SAS ESP model subscriber. The standby instances join the active instance in receiving the incoming stream of data, therefore keeping their in-memory view aligned.

SAS ESP ROLLING UPDATES WITH ZERO DOWNTIME

Our approach takes advantage of the advanced orchestration capabilities offered by the SAS ESP Kubernetes Operator object discussed earlier in this paper. The SAS ESP Operator object uses custom watchers to identify significant events happening on the Kubernetes objects that it manages. An example of such an event is a request to update or patch the container image. If this event occurs, the SAS ESP Operator detects the discrepancy between the desired version and current version, and then performs a set of steps to bring the environment to the desired state.

The custom logic performed by the SAS ESP Operator is as follows:

1. Label the Kubernetes pods with a label "role=standby" or "role=active", depending on the role of SAS ESP server.
2. Take a recovery point of the first SAS ESP server that is labeled with the role *"standby"* and persist the recovery point to the shared storage.
3. Update the first standby SAS ESP server instance.
4. Update the successive SAS ESP standby instances sequentially.
5. Update the active SAS ESP server instance.

Figure 11 shows stages 1 to 4 of this process.

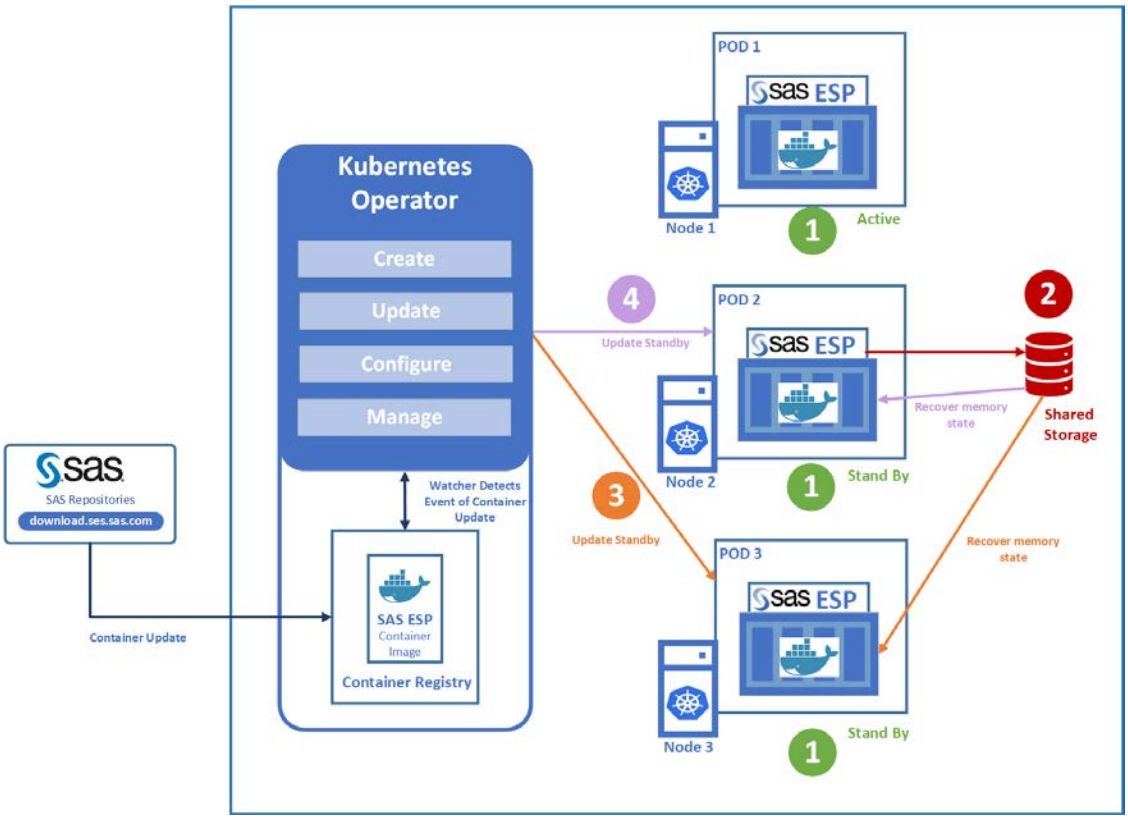

Figure 11. Rolling Updates to SAS ESP Container, Stages 1-4

Figure 12 shows stage 5 when pod 1 takes a standby role while the container is updated.

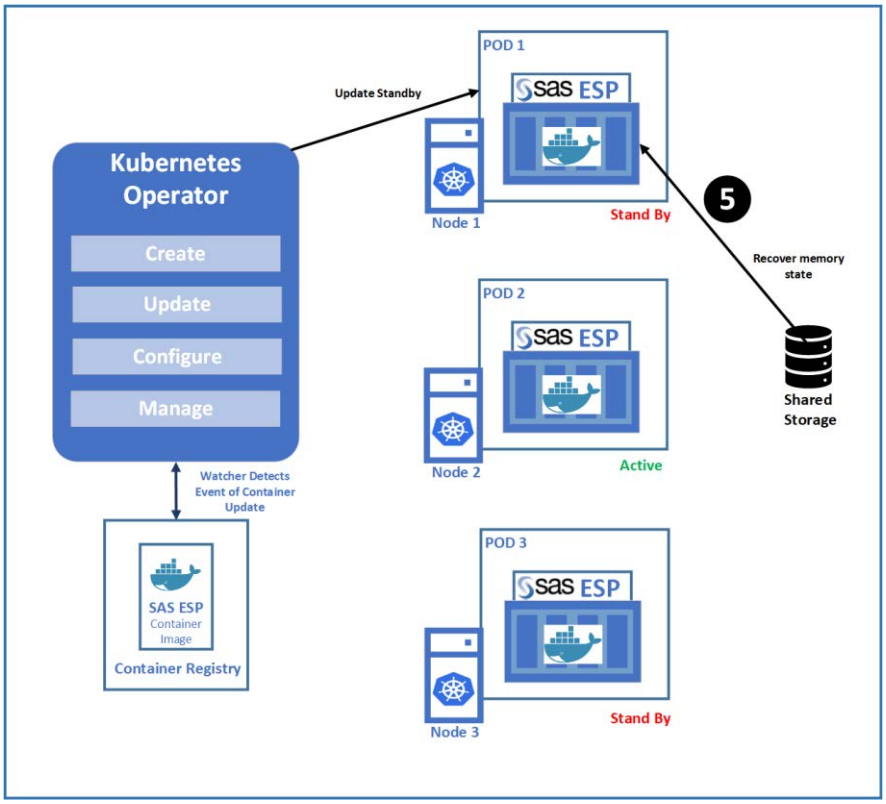

Figure 12. Stage 5, Remaining Active Server is Set to Standby

In step 2, the SAS ESP Operator identifies a suitable pod where SAS ESP runs in standby mode, and issues the appropriate HTTP calls to stop the project and persist the data to a shared location.

The update strategy implemented in our configuration terminates and re-creates each container individually in an orderly fashion and ensures that the current container is ready before moving on to the next one. In order to avoid unnecessary failovers, the process starts with the SAS ESP server standby nodes. From step 3 to 5, every SAS ESP server uses the recovery point taken in step 2 to synchronize its memory state. Because the recovery point was taken just before the update, the delta of data required to re-align the sequence of messages is minimal.

This orderly update process ensures that a designated active SAS ESP server will always be present in the cluster, which ensures continuous processing.

The configuration of the container has been carefully considered by implementing a readiness probe, which allows enough time for the restore of the recovery point data to take place before updating the next container.

By leveraging the SAS ESP Operator, we can seamlessly deploy new versions of the SAS ESP server image with zero downtime and a high degree of automation, which removes the element of risk in the overall operation and enforces consistency.

ESP RECOVERY FROM FAILOVER

If a pod fails for any reason, Kubernetes detects that the current state of the deployment does not match its desired state. For example, consider the case where the number of running pods is less than the required number of replicas, as defined in the configuration. If this occurs, Kubernetes attempts to eliminate this gap by launching a new pod running the SAS ESP server container. This process is completely seamless and requires no human intervention. After the container is running, the state of the project can be restored from the recovery point stored on shared storage. This brings the newly started instance in sync with the existing active and standby instances and enables it to acquire the appropriate role in the SAS ESP cluster.

CONCLUSION

We demonstrate that a high degree of automation can be achieved by adoption of container technology for the deployment of SAS ESP. Our approach leverages the advanced in-built failover capabilities of SAS ESP coupled with the benefits of containerization on a cloud provider. This approach quickly delivers a highly available deployment of SAS ESP that is integrated with Kafka. The software components of the deployment can be seamlessly updated in an automated fashion with zero downtime through the advanced customization capabilities of SAS ESP Operator. Finally, such an approach is fully repeatable across multiple environments and cloud providers because of the cloud-neutral nature of Kubernetes.

Although we have explored the capabilities of these technologies, the journey to achieving a mature level of best practices will evolve together with the ability of IT organizations to adopt and integrate them into their ways of working.

Advanced engineering and design patterns have been democratized through the adoption of the technologies and methodologies described in this paper. As a result, the time to market and the total cost of ownership for SAS ESP at 'full stream' have been significantly reduced.

ACKNOWLEDGMENTS

SAS Viya Container Recipes contributors (see https://github.com/sassoftware/sas-container-recipes/graphs/contributors)

SAS R&D

SAS Global Enablement Team

SAS ESP Product Team

SAS Colleagues from Professional Services Division

SAS ESP Customers in UK

REFERENCES

SAS Institute Inc. 2019. SAS Software /SAS Viya Container Recipes. Accessed March 21, 2019. Available https://github.com/sassoftware/sas-container-recipes

RECOMMENDED READING

SAS Institute Inc. 2019. "SAS for Containers." Available https://support.sas.com/rnd/containers/index.html. Accessed March 21, 2019.

SAS Institute Inc. 2019. "The Basics." Available https://github.com/sassoftware/sassoftware/sas-container-recipes/wiki/The-Basics. Accessed March 21, 2019.

Furbee, Joe. "Getting Started with SAS Containers." Available https://blogs.sas.com/content/sgf/2019/03/06/getting-started-sas-containers/. Last Modified March 6, 2019. Accessed March 21, 2019.

CONTACT INFORMATION

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

Alessio Tomasino
Principal Technical Consultant
SAS UK and Ireland Professional Services
Alessio.Tomasino@sas.com

Louis Katsouris
Senior Solutions Architect
SAS UK and Ireland Professional Services
Louis.Katsouris@sas.com