Paper SAS3481-2019

Proper Planning Prevents Possible Problems: SAS® Viya® High-Availability Considerations

Edoardo Riva, SAS Institute Inc., Cary, NC

ABSTRACT

SAS® Viya® is used for enterprise-class systems, and customers expect a reliable system. Highly available deployments are a key goal for SAS Viya. This paper addresses SAS Viya high-availability considerations through different phases of the SAS® software life cycle. After an introduction to SAS Viya, design principles, and intra-service communication mechanisms, we present how to plan and design your SAS Viya environment for high availability. We also describe how to install and administer a highly available environment. Finally, we examine what happens when services fail and how to recover.

INTRODUCTION

This paper assumes a basic understanding of SAS® Viya® architecture, so it does not describe the role and functionality of its different components.

Jerry Read's paper describes the process of creating highly available SAS Viya 3.3 environments (Read 2018). This paper follows a more theoretical line, highlighting key considerations that are required with the design and administration of such an environment in SAS Viya 3.4, the currently available release.

SAS VIYA DESIGN PRINCIPLES

SAS Viya has been designed with services redundancy in mind, to increase services availability and improve performance through load sharing. Any single failure in the system should have the following effects:

- Require no immediate response from an administrator.
- Have minimal impact to current users of the system.
- Have no impact on future users of the system.
- Result in immediate notification to administrators of the failure.

Expectations about what can be considered *minimal impact to active users* can vary, depending on customer requirements, the application involved, and the failure, but a typical acceptable impact can be any of these:

- An in-progress action fails with a server error.
- Users might need to refresh the browser to recover.
- The system might continue to exhibit failures or delays for a short period--on the order of few minutes.

An administrator should be able to return a failed system component to health without taking the system offline or affecting active users.

Well, thanks to stateless microservices, all these objectives become much simpler to obtain. Individual groups of services can be clustered independently of others. After service instances have been started successfully, they register themselves within the SAS® Configuration Server (based on HashiCorp Consul) and are available to service requests.

The SAS Configuration Server continuously checks the health status of registered services, and connections are routed only to healthy instances.

For a good description of microservices architecture, see Eric Bourn's paper (Bourn 2018).

Stateful services are less dynamic than microservices, but a similar concept applies to most of them. Most of these components have some support for high availability, and you can deploy multiple instances of the server as a cluster across different machines, whether it's SAS® Message Broker or SAS® Infrastructure Data Server.

HOW SERVICE DISCOVERY AND ROUTING WORKS

Service discovery and routing is built on the idea that clients should not need to know the physical location of services. This concept originated within cloud environments, where services can be started on demand or moved to a different host at any time. But it makes perfect sense also when dealing with high availability clusters of redundant services: clients should be insulated from the details of how many service instances are running or where they are located.

Within SAS Viya, this is possible because Apache HTTP Server is the front door to all web applications and microservices; its mod_proxy module routes requests to services at the designated port and balances the traffic among multiple service instances. Starting with SAS Viya 3.3 Apache also proxies any access to programming components such as SAS® Cloud Analytics (CAS) Server Monitor and SAS® Studio 4. Apache proxies both external connections (coming from a client such as a browser) and internal ones (service-to-service). Figure 1 shows an external connection (red arrows) from a browser going through the proxy to reach SAS® Studio 5. SAS Studio itself then opens an internal connection (blue arrows) (for example, to talk to the SASLogon microservice), and the connection is proxied as well.

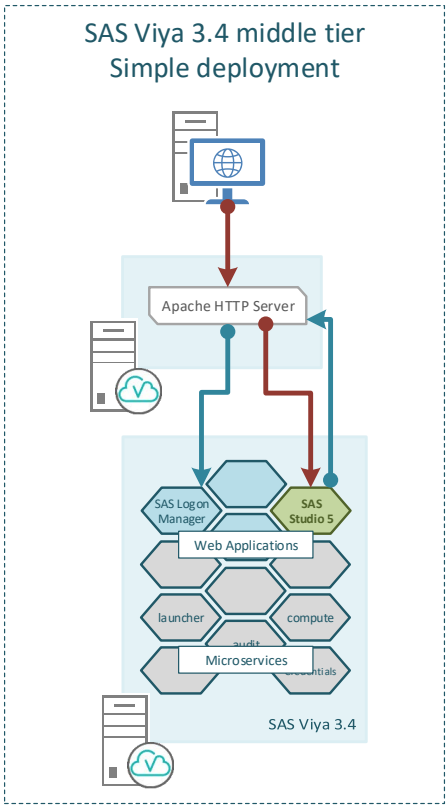

Figure 1. Apache HTTP Server Proxying a Simple SAS Viya Deployment

Here is a key point to remember: there are no direct internal connections, ever. Apache HTTP Server proxies every connection to microservices. This is one of the principles followed by SAS in designing SAS Viya and is key for SAS Viya to be ready for cloud environments.

DO YOU WANT TO SEE IT?

It's easy to verify how Apache can forward all connections to the right endpoints. All proxy directives are stored in a specific file, /etc/httpd/conf.d/proxy.conf. Since this is a custom configuration file, upgrading Apache will not overwrite, modify, or delete it. Here is an extract from an environment with a two-machine, middle-tier cluster:

```
... more lines ...


# Proxy to SASBackupManager service
<Proxy balancer://SASBackupManager-cluster>
  BalancerMember https://viya01.example.com:46560 route=backupmanager-192-168-0-2
  BalancerMember https://viya02.example.com:46073 route=backupmanager-192-168-0-18
  ProxySet scolonpathdelim=on stickysession=JSESSIONID
</Proxy>
Redirect /SASBackupManager /SASBackupManager/
ProxyPass /SASBackupManager/ balancer://SASBackupManager-cluster/SASBackupManager/
ProxyPassReverse  /SASBackupManager/ balancer://SASBackupManager-
cluster/SASBackupManager/

# Proxy to SASDataExplorer service
<Proxy balancer://SASDataExplorer-cluster>
  BalancerMember https://viya01.example.com:42954 route=dataexplorer-192-168-0-2
  BalancerMember https://viya02.example.com:37072 route=dataexplorer-192-168-0-18
  ProxySet scolonpathdelim=on stickysession=JSESSIONID
</Proxy>
Redirect /SASDataExplorer /SASDataExplorer/
ProxyPass /SASDataExplorer/ balancer://SASDataExplorer-cluster/SASDataExplorer/
ProxyPassReverse  /SASDataExplorer/ balancer://SASDataExplorer-
cluster/SASDataExplorer/

... more lines ...
```

Here are some points that you can understand from this fragment:

- Each service in this environment has been clustered and is currently running on two hosts; each instance is listening on an ephemeral port.
- There is a separate "Balancer" per service; this way, all services are independent and could be deployed, scaled up/down, started, and stopped independently from other ones.
- Once a client session is established, the parameter *stickysession=JSESSIONID* keeps it connected to the same instance of the clustered service.

The last point might raise some concerns: *sticky sessions* could be an issue for high availability! If a session is always connected (for example, to host #1), what happens if that machine dies? Won't you lose all your work? Another key SAS Viya architecture design feature comes to the rescue: all microservices are stateless--that is, they do not save anything internally. The status of the current session, for example, is saved in the SAS

Cache Server. Were host #1 to die, Apache would route any requests for services to the surviving instances--for example, on host #2. That, in turn, would extract the session ID from the incoming request and retrieve its status from the external cache. Everything is preserved, and end users do not notice any issue.

BACK TO THE THEORY

Up to now, you have seen *service routing*—that is, how to route a connection from a client to a running service. Apache HTTP Server can do it for you. If you think about it for a moment, you might realize this has not solved the issue. It's simply been moved down one level, from the client to the proxy. How can Apache actually know where services are running? That's the focus of *service discovery*. To do it, SAS Viya relies on two additional components, and also relies on the way services relate with them. These components are the *SAS Configuration Server* and the *httpproxy service*.

The SAS Configuration Server, despite its name, is not only a central repository for configuration data, but also the core component for service discovery and service health status.

Every time a SAS Viya service is started, it connects to the SAS Configuration Server and registers its name, ID, hostname, and port, plus additional information. It also registers a check that the SAS Configuration Server performs every few seconds to verify that the service is actually up and responsive. In a similar way, every time a SAS Viya service is gracefully stopped, it connects to the SAS Configuration Server and deletes its registration and any associated health check.

Next, here are details about the httpproxy service: its role is to query the SAS Configuration Server for service events and to update Apache.

- When a new service instance starts responding to health checks, httpproxy reads its name, host, and port and adds its route to the proxy.conf file described above. Apache is then forced to reload its configuration and starts routing client connections to this new service instance.
- When a service instance is stopped or does not respond to health checks, httpproxy removes the corresponding entry from the proxy.conf file. Apache is then forced to reload its configuration and stops routing client connections to the dead service instance.

YOU CAN CHECK THIS, TOO

SAS Viya provides the sas-bootstrap-config command-line utility to interact with the SAS Configuration Server. We can use it to perform service discovery manually, as in the following examples.

1. Assume that you previously started one instance of the audit service. You can check its registration:

```
# define env variables if not already defined
$ [[ -z "$CONSUL_HTTP_ADDR" ]] && . /opt/sas/viya/config/consul.conf
$ [[ -z "$CONSUL_TOKEN" ]] && export CONSUL_TOKEN=$(sudo cat
/opt/sas/viya/config/etc/SASSecurityCertificateFramework/tokens/consul/defaul
t/client.token);
# discover the audit service
$ /opt/sas/viya/home/bin/sas-bootstrap-config catalog service audit
{
    "items": [
        {
```

```
            "address": "192.168.0.2",
            "node": "viya01.example.com",
            "serviceAddress": "viya01.example.com",
            "serviceEnableTagOverride": false,
            "serviceID": "audit-192-168-0-2",
            "serviceName": "audit",
            "servicePort": 40888,
            "serviceTags": [
                "proxy",
                "rest-commons",
                "https",
                "contextPath=/audit"
            ],
            "taggedAddresses": {
                "lan": "192.168.0.2",
                "wan": "192.168.2.2"
            }
        }
    ]
}

# verify Apache configuration
$ grep audit /etc/httpd/conf.d/proxy.conf
# Proxy to audit service
Redirect /audit /audit/
ProxyPass /audit/ https://viya01.example.com:40888/audit/
ProxyPassReverse  /audit/ https://viya01.example.com:40888/audit/
```

2. Now start an additional instance of the audit service, and check again:

```
# discover the audit service
$ /opt/sas/viya/home/bin/sas-bootstrap-config catalog service audit
{
    "items": [
        {
            "address": "192.168.0.2",
            "node": "viya01.example.com",
            "serviceAddress": "viya01.example.com",
            "serviceEnableTagOverride": false,
            "serviceID": "audit-192-168-0-2",
            "serviceName": "audit",
            "servicePort": 40888,
            "serviceTags": [
                "proxy",
                "rest-commons",
                "https",
                "contextPath=/audit"
            ],
            "taggedAddresses": {
                "lan": "192.168.0.2",
                "wan": "192.168.0.2"
            }
        },
        {
```

```
            "address": "192.168.0.18",
            "node": "viya02.example.com",
            "serviceAddress": "viya02.example.com",
            "serviceEnableTagOverride": false,
            "serviceID": "audit-192-168-0-18",
            "serviceName": "audit",
            "servicePort": 3044,
            "serviceTags": [
                "proxy",
                "rest-commons",
                "https",
                "contextPath=/audit"
            ],
            "taggedAddresses": {
                "lan": "192.168.0.18",
                "wan": "192.168.0.18"
            }
        }
    ]
}

# verify Apache configuration
$ grep audit /etc/httpd/conf.d/proxy.conf
# Proxy to audit service
  BalancerMember https://viya01.example.com:40888 route=audit-192-168-0-2
  BalancerMember https://viya02.example.com:3044 route=audit-192-168-0-18
Redirect /audit /audit/
ProxyPass /audit/ balancer://audit-cluster/audit/
ProxyPassReverse  /audit/ balancer://audit-cluster/audit/
```

3. The output of the commands above gives you the URL of each service instance. With these, you can manually check the health of each of the cluster members.

```
$ curl -k https://viya01.example.com:40888/audit/commons/health
{"description":"Composite Discovery Client","status":"UP"}
$ curl -k https://viya02.example.com:3044/audit/commons/health
{"description":"Composite Discovery Client","status":"UP"}
```

ADDRESS VIRTUALIZATION AND LOAD BALANCERS

The inter-service routing works very well when there is only a single instance of Apache running. When there are multiple instances of Apache, services should not use any that are down. A useful feature with SAS Viya is that it keeps track of Apache health, using the httpproxy service with health checks to track when an Apache HTTP Server goes down. Microservices include a services resolver that provides them with up-to-date information about where to find each active Apache HTTP Server instance. This is represented in Figure 2 by the multiple blue and red lines connecting microservices and Apache HTTP Server instances.

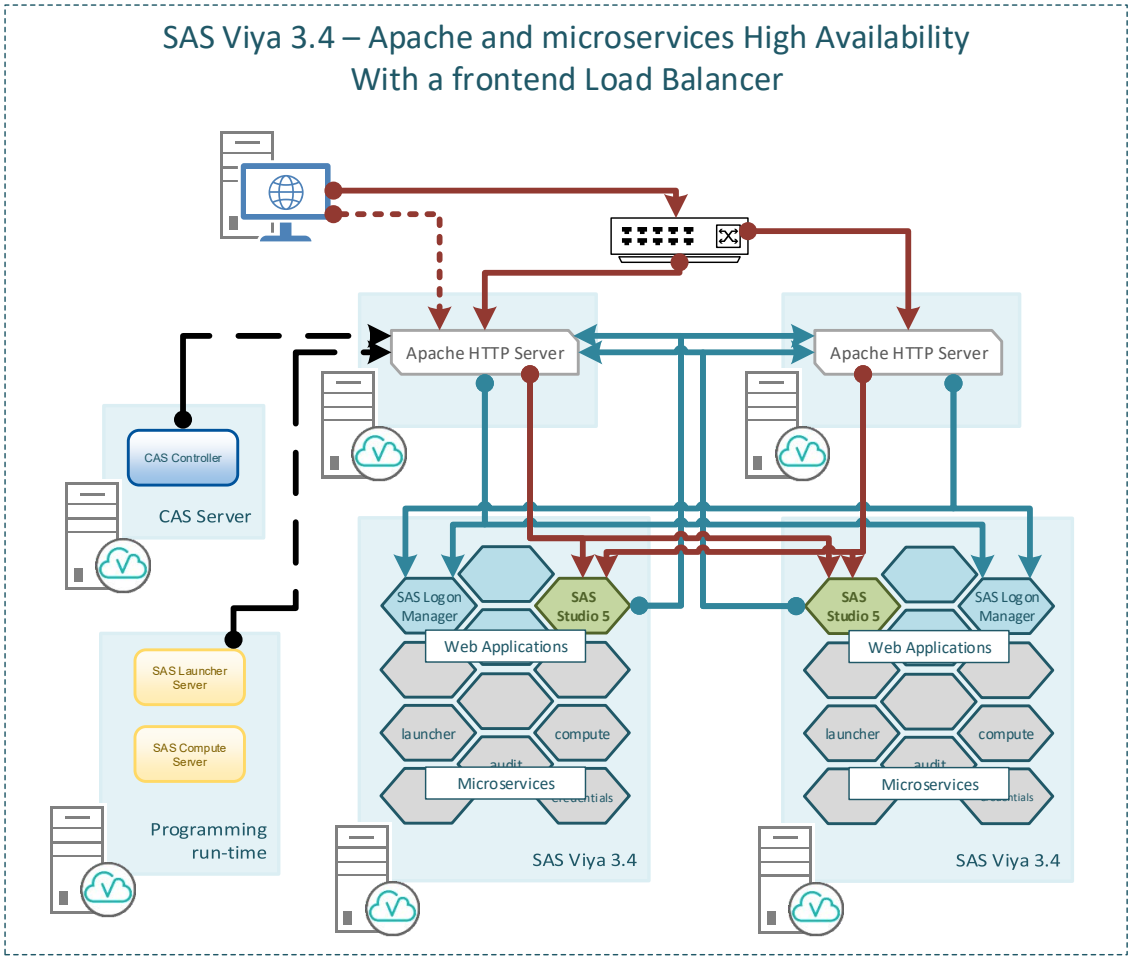

Figure 2 – A SAS Viya Middle-Tier Cluster

While this works extremely well as of SAS Viya 3.4, there are just a few components that do not automatically take advantage of the service resolver. The SAS Launcher Server, CAS, and applications that generate links to SAS Viya reports for sharing outside SAS Viya, all remain hard wired to the Apache HHTP Server instance running on the first machine listed in the inventory file under the [httpproxy] host group. This is illustrated in Figure 2 by the dotted black and red lines.

Since a cluster of Apache HTTP Servers requires an external front-end proxy or a load balancer to route external incoming connections, you can set additional properties to identify this external single entry-point for these components that do not use the service resolver.

First, since the external proxy/load balancer should be set up to use transport layer security (TLS), you must install its certificate authorities (CA) certificate chain into the SAS Viya truststore, as described in "Add Certificates to Truststores Using Ansible (Linux Full Deployment)" in *Encryption in SAS Viya 3.4: Data in Motion*.

SAS Cloud Analytics Services (CAS) Server

The CAS server does not use the SAS Configuration Server to find an active instance of Apache HTTP Server. It stores in its configuration a URI to locate SAS Viya microservices in order to validate OAuth tokens being presented for authentication from other SAS Viya web applications or to obtain OAuth tokens for clients authenticating with user/password

credentials. By default, this URI is fixed to the Apache HTTP Server that is deployed on the first machine listed for the [httpproxy] inventory host group. You can override this default by setting the value for its properties *cas.SERVICESBASEURL, env.CAS_VIRTUAL_HOST, env.CAS_VIRTUAL_PORT,* and *env.CAS_VIRTUAL_PROTOCOL* to point to the front-end external proxy/load balancer that routes traffic across multiple Apache HTTP Servers.

Example: cas.SERVICESBASEURL = 'https://loadbalancer.viya.customer.com:443/'
env.CAS_VIRTUAL_HOST ='loadbalancer.viya.customer.com'
env.CAS_VIRTUAL_PORT = '443'
env.CAS_VIRTUAL_PROTOCOL = 'https'

If you know the properties before the deployment, you can specify these properties in the vars.yml file as described in *SAS Viya 3.4 for Linux: Deployment Guide.* Otherwise, you can change them post-deployment, by updating the configuration file /opt/sas/viya/config/etc/cas/default/casconfig_usermods.lua, and then restarting the CAS controller.

Of course, CAS must have a network route to the load balancer.

SAS Launcher Server

When Compute Servers are launched, they ultimately need to talk back to the Compute service and should do so through the load balancer. By default, however, the Launcher Server passes only the address of the first Apache HTTP Server instance listed in the inventory file to each launched Compute Server. The SAS Launcher Server requires a property retrieved from the SAS Configuration Server (*config/launcher-server/global/sas-services-url*) to specify the URL for the single entry point to SAS Viya. If you know this URL before the deployment, you can specify the required property in the sitedefault.yml, following the instructions in *SAS Viya 3.4 for Linux: Deployment Guide*. Otherwise, you can add the property post-deployment using the following commands:

```
. /opt/sas/viya/config/consul.conf
export CONSUL_TOKEN=$(sudo cat /opt/sas/viya/config/etc/SASSecurityCertificat
eFramework/tokens/consul/default/client.token)
/opt/sas/viya/home/bin/sas-bootstrap-config kv write config/launcher-
server/global/sas-services-url "https://<loadbalancer-address>:443/"
```

In this latter case, restart the *sas-viya-runlauncher-default* service.

Again, a network path must exist from the Compute Server hosts to the load balancer and not be preempted by a firewall rule.

URL Generators

Applications that generate links to SAS Viya objects, such as SAS® Visual Analytics reports, also require a configuration change, specific to the SAS Viya service, that will override an otherwise hard-wired affinity to the primary Apache HTTP Server instance. You can use the properties `config/viya/sas.httpproxy.external.hostname` and `config/viya/sas.httpproxy.external.port`. Again, if you know these properties before the deployment, you can specify them in the sitedefault.yml, following the instructions in *SAS Viya 3.4 for Linux: Deployment Guide*. Otherwise, you can add them post-deployment using the following commands:

```
. /opt/sas/viya/config/consul.conf
```

```
export CONSUL_TOKEN=$(sudo cat /opt/sas/viya/config/etc/SASSecurityCertificat
eFramework/tokens/consul/default/client.token)
/opt/sas/viya/home/bin/sas-bootstrap-config kv write
config/viya/sas.httpproxy.external.hostname <loadbalancer_hostname>
/opt/sas/viya/home/bin/sas-bootstrap-config kv write
config/viya/sas.httpproxy.external.port <loadbalancer_port>
```

After this, restart the *sas-viya-reportdistribution-default* and *sas-viya-reportalerts-default* services.

Finally, it should be noted that a virtual address for the microservices (for example, using a front-end proxy or a hardware load balancer) is also important for the user experience. It not only provides the convenience of a single bookmarked entry point for SAS Viya, but it makes single signon and signoff among SAS Viya applications possible. SAS Logon tracks session information mapped to the hostname being used to connect to it. This allows SAS Logon to share a single session for a user among many instances of itself. When you log off from SAS Logon, through any running instance, you′re logged off from SAS Viya completely: all web applications in all browser sessions (for a given browser). Conversely, if you allow users to reach SAS Logon at multiple addresses, then multiple sessions will be created, and they won't benefit from the single signon and signoff.

Figure 3 shows the final result of all of the previous configuration changes. All single-address connections are routed through the front-end load balancer.

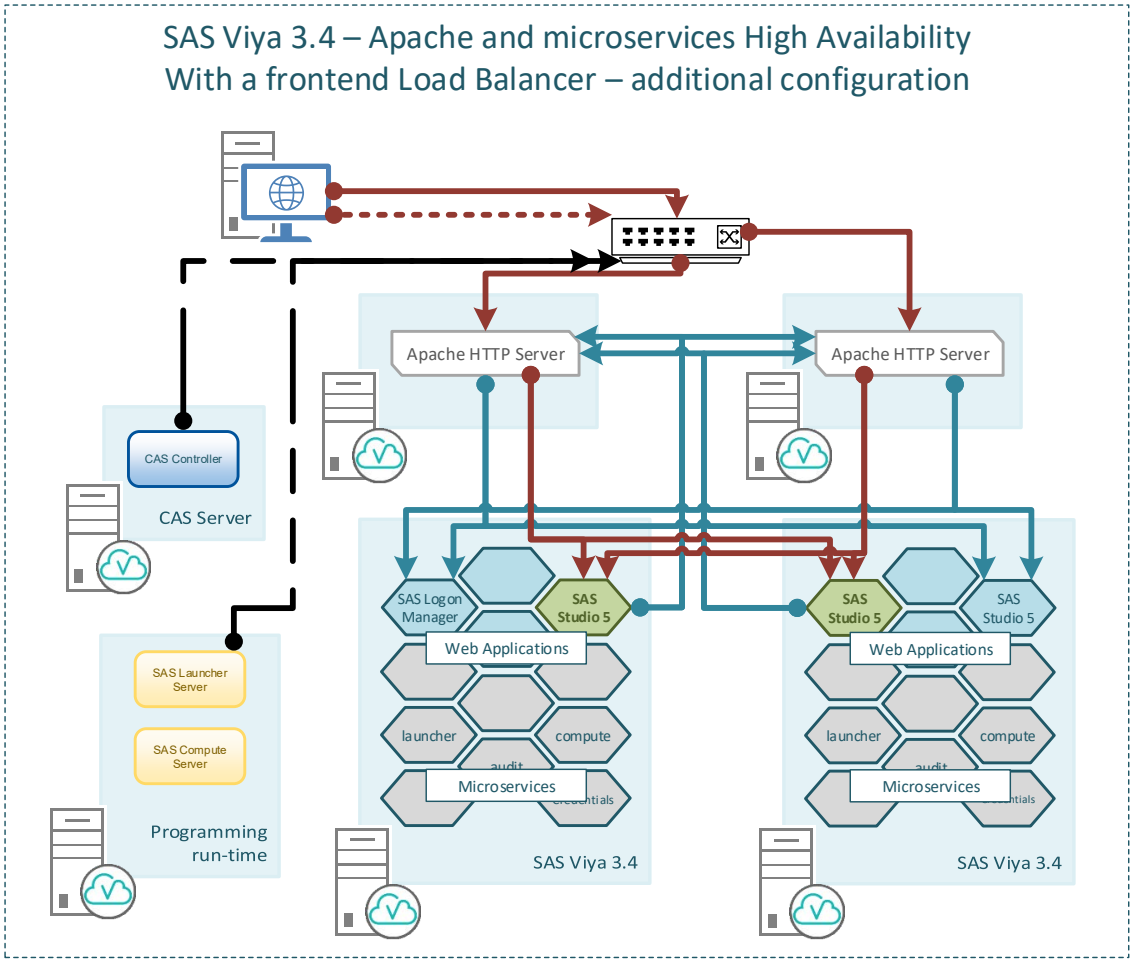

Figure 3 - SAS Viya Middle-Tier Full High Availability

PLAN AND DESIGN YOUR **SAS** VIYA ENVIRONMENT FOR HIGH AVAILABILITY.

SAS Viya servers and services can be clustered to increase their availability. With clustering, if a member of the cluster goes down, all of the other ones keep servicing client requests.

In order to guard against unexpected issues, such as hardware, operating system, or network failures, it is recommended that cluster instances be distributed across multiple machines. Deploying redundant instances of each service results in a highly available and more robust system that requires less attention when a failure occurs.

For some components, clustering should be planned and configured before starting the deployment. Other can be clustered at any time.

The following table lists the main SAS Viya components with their currently supported clustering status.

Component	Clusterable at Deployment?	Clusterable Post-deployment?
SAS Cloud Analytics Services (CAS)	Y	Y

SAS Studio V (5.*x*)	Y	Y
Apache HTTP Server	Y	Y
SAS Infrastructure Data Server (PostgreSQL)	Y	Y
Microservices	Y	Y
SAS Configuration Server (Consul, includes Vault)	Y	Y[1]
SAS Message Broker (RabbitMQ)	Y	N[2]
SAS Compute Server	Y	N
SAS Studio (4.*x*)	Y	N
Pgpool II	N	N[3]
Operations	N[4]	N[4]

[1]. The only tested--and thus supported--case is when you add a new Consul server, after the initial deployment, on hosts that do not already have a Consul agent on it. This means that the cluster can be expanded only on machines that do not already host any other SAS Viya software.

[2]. Although clustering RabbitMQ post-deployment should be possible, it has not been officially tested. Thus, it is not supported.

[3]. SAS R&D has recently tested a supported way to add Pgpool II nodes post-deployment. For more information, contact SAS Technical Support or a member of SAS Professional Services.

[4]. The Operations host group contains services that accumulate metric, log, and notification events from RabbitMQ, and then process those into CAS tables that are consumed by the SAS® Environment Manager application. Only one instance can be deployed per environment. In case of failure, end users will be unaffected: only administrators will be affected. They will not be able to use SAS Environment Manager to consume the information provided by the Operations microservice, but they should still be able to get the same information from other sources.

SAS CLOUD ANALYTICS SERVICES

SAS Cloud Analytics Services (CAS) can be deployed in a distributed analytic cluster (MPP). In this configuration, CAS Server is more resilient to failures.

Even if a CAS worker node fails, the service as a whole is still available. Likewise, data, that by default is replicated, is not lost.

Starting with SAS Viya 3.3, CAS can also have one (and only one) backup (or secondary) controller. A CAS backup controller provides fault tolerance in case the primary CAS controller fails. It can be used only in a distributed server architecture, and its deployment is optional.

The primary and backup controller hosts should be identical (for example, in sizing, operating system version and settings, prerequisites, and so on).

The primary and backup controller should share the following directory:

/opt/sas/viya/config/data/cas

Since only one of the controllers can service client requests (warm standby), if you have a core-based SAS license, the cores of the backup controller do not count toward the total number of cores.

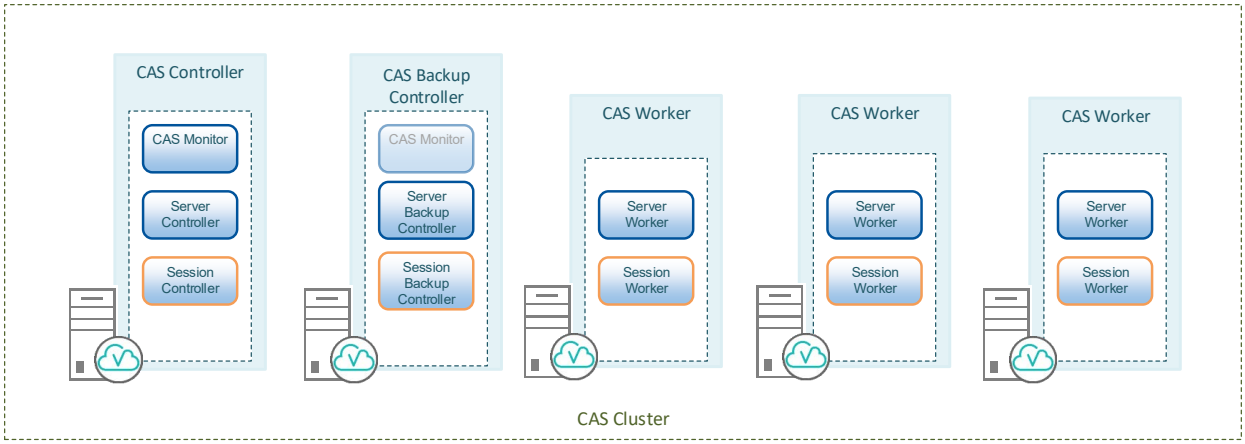

Figure 4 – CAS Cluster with a Backup Controller

PROGRAMMING RUN-TIME SERVERS

SAS Studio 4 with its supporting components (SAS Object Spawner, SAS Workspace Server) can be clustered. Similarly, the SAS Compute Server and SAS Launcher Server-- that support SAS Viya applications such as SAS Studio 5--can be clustered.

The two clusters can share hosts, as shown in Figure 5, or, starting with SAS Viya 3.4, reside on different hosts, as shown in Figure 6.

Each instance is independent, and there is no session failover. In the event of a failure, a new session can be established on a different host. This happens after the user performs a new logon, with SAS Studio 4, or automatically, with SAS Studio 5.

All instances of the cluster must be able to access the same saved configuration data. To enable this, you should do the following:

- Set up a shared file system and configure SAS Studio 4 to use a shared drive in that file system for all the user data you want to save. For more information, see the description of "webdms.studioDataParentDirectory" in *SAS Viya 3.4 Administration: Configuration Properties*.
- Enable file sharing for home directories on all hosts where programming interfaces are installed.

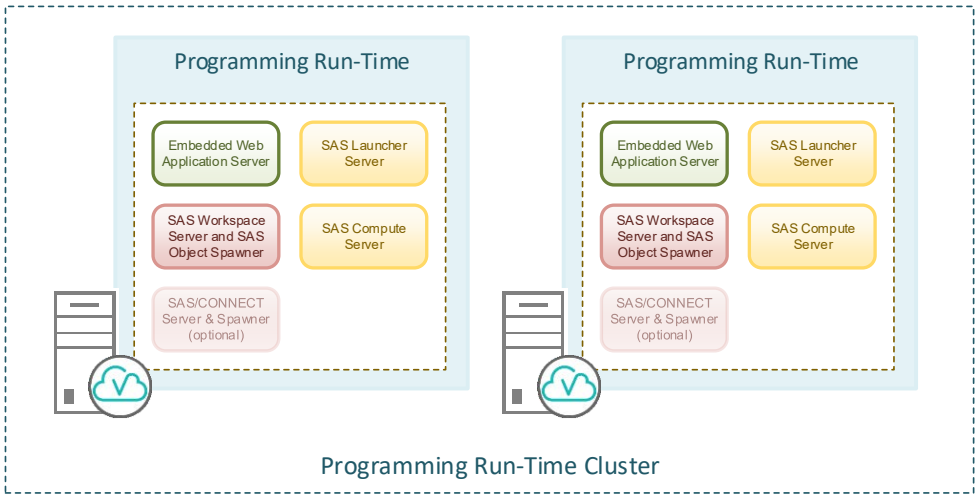

Figure 5 – Example of Programming Run-Time Cluster

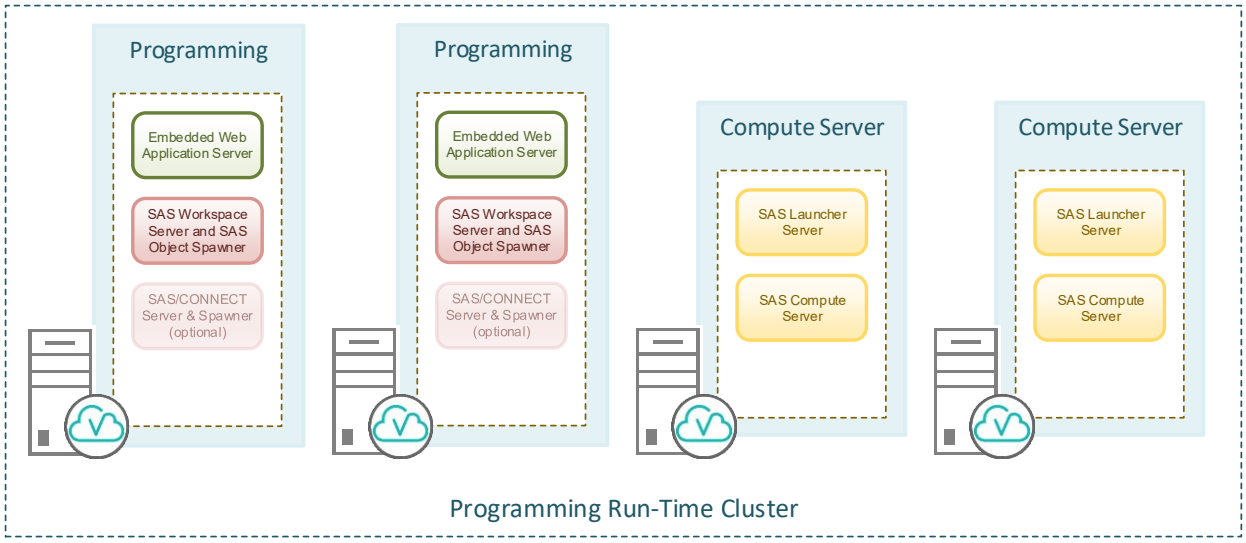

Figure 6 – Programming Components Split from SAS Compute Server and SAS Launcher Server

INFRASTRUCTURE SERVERS

Apache HTTP Server can be clustered. All active instances can proxy incoming requests to microservices or web applications, as described in the "How Service Discovery and Routing works" section. An external proxy server or hardware load balancer is required in front of the cluster, so that every external web request transits through it. This is shown in Figure 7.

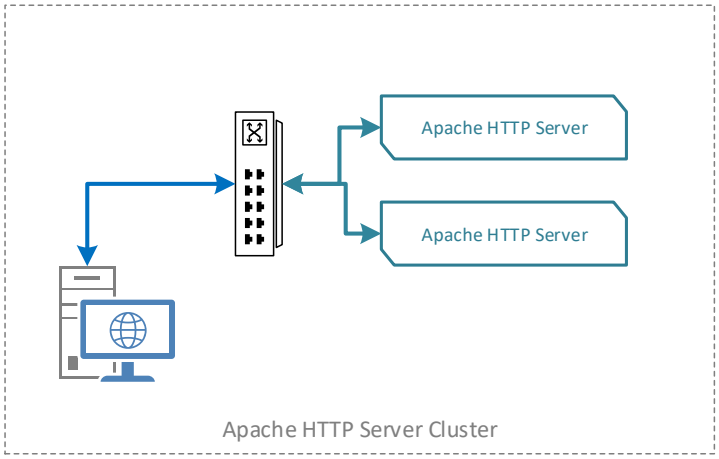

Figure 7 - Apache HTTP Server Cluster

The external proxy should use the https protocol; it must forward requests through the SAS Viya Apache HTTP Servers without changes in the URL path. The external proxy or load balancer is responsible for routing requests only to active Apache HTTP Server instances; round robin or load balanced routing is recommended.

SAS Configuration Server (Consul) and SAS Secret Manager (Vault) can be clustered. The cluster elects a leader: while all nodes can answer client requests, only the leader is responsible for Writes and updates to the cluster, as shown in Figure 8. A Consul cluster should always have an odd number of members. Three or five are recommended in most situations; an excessive number of servers will affect performance.

Vault uses Consul as its back-end storage and is always deployed on each Consul server.

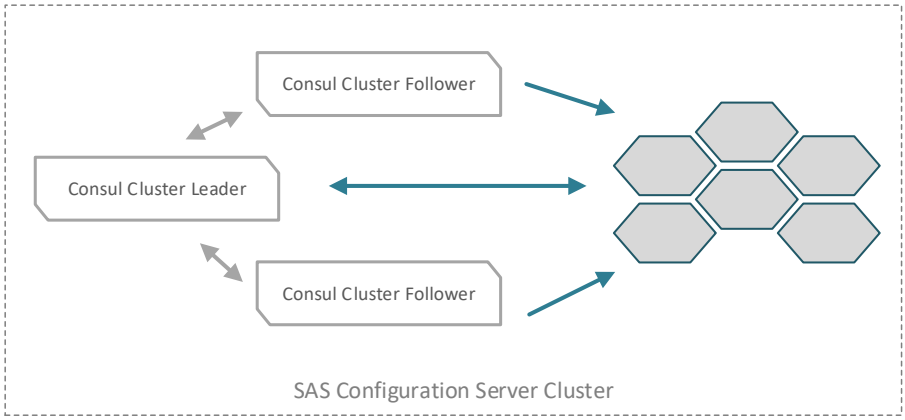

Figure 8 - SAS Configuration Server Cluster

SAS® Infrastructure Data Server (PostgreSQL) can be clustered. One instance assumes the primary role and services incoming requests. All other instances have the standby role and become active only in case the primary fails. All data transactions are replicated from the primary to the standby nodes, thus ensuring data safety in case of loss of the primary host. Figure 9 shows a horizontal cluster with two nodes, each on a separate host. While SAS Infrastructure Data Server supports numerous other highly available topologies, discussing all of them is outside the scope of this paper. To learn more, see "Creating High Availability PostgreSQL Clusters" in the *SAS Viya 3.4 for Linux: Deployment Guide*.

SAS provides Pgpool-II open-source software to manage PostgreSQL clusters. Pgpool software resides and operates between SAS Infrastructure Data Servers and clients, as shown in Figure 9. All data connections and database requests are routed through the Pgpool service. Currently, SQL queries are relayed only to the primary node. The arrow in Figure 9 depicting queries being also sent to a standby node, to load balance the workload, represents an improvement being researched for a future SAS Viya release. Pgpool is also responsible for monitoring the PostgreSQL cluster and, in case of failure of the primary node, it promotes one of the standby nodes to become the new primary and reconfigures any additional standby node to follow the new primary.

For the current release, Pgpool itself is a single point of failure. This was a conscious decision because the software release that was initially deployed with SAS Viya did not support any reliable way to establish quorum in case of network issues (split-brain problem). Were this split-brain problem to happen, it could lead to possible corruption of the managed databases. SAS has thus chosen to enforce data integrity over availability.

Recent releases of Pgpool have finally solved this issue. As of publication time, work is underway to provide an out-of-the-box deployment of a cluster of Pgpool instances for SAS Viya. In the meantime, contact SAS Technical Support or SAS Professional Services to manually implement this new configuration in your existing SAS Viya environment.

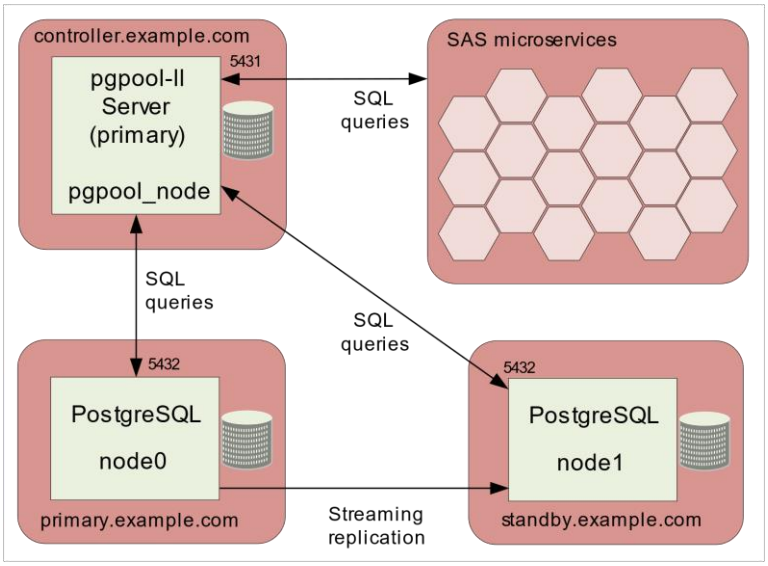

Figure 9 – SAS Infrastructure Data Server Cluster

SAS Message Broker (RabbitMQ) can be clustered. At deployment time, the first machine defined in the cluster is the primary node and initializes the cluster. The RabbitMQ cluster model is quite complex. It can be considered Active/Active for failover purposes because all members of the cluster are active and take requests directly, but there are nuances. RabbitMQ queues are assigned to different members of the cluster. For each queue, that RabbitMQ instance is "in charge" of that queue, such that activity in that queue is managed by that particular RabbitMQ instance. Other cluster members effectively forward requests for that queue to that manager instance, and that manager maintains and mirrors state information and content back to the other cluster members. If that queue's manager fails, a different RabbitMQ instance takes over management of that queue (and retains it even if the original comes back online).

SAS has recently discovered that the default configuration is susceptible to loss of messages, in the case of network problems or partitions. To overcome this issue, it is required to deploy an odd number of RabbitMQ instances, as shown in Figure 10, and verify that the configuration property *cluster_partition_handling = pause_minority* is present in the file */opt/sas/viya/config/etc/rabbitmq-server/rabbitmq.config.ssl* on each node. Three or five nodes are recommended in most situations, just as with the SAS Configuration Server. For additional details, see SAS Note 63804.

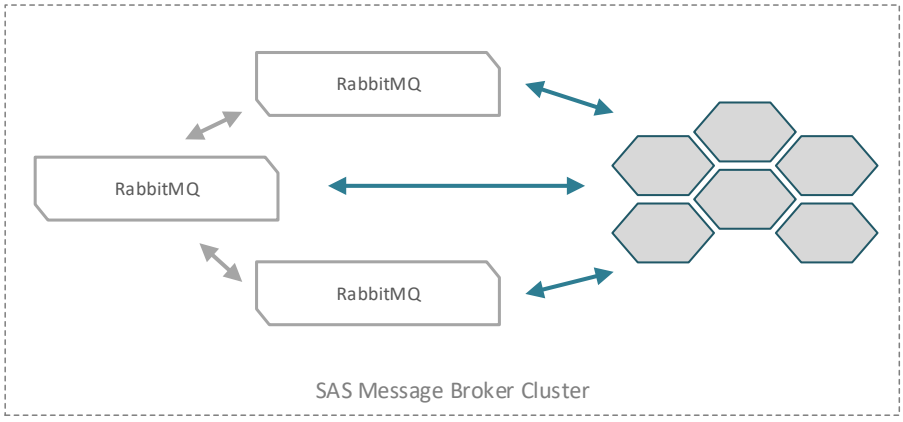

Figure 10 - SAS Message Broker Cluster

MICROSERVICES

Microservices and web applications are stateless, and an arbitrary number of instances can be started to form a cluster. At least two instances of each service are required for high availability.

Each instance registers with the SAS Configuration Server when it starts, and it is continuously monitored. If it fails, it is removed from the SAS Configuration Server service catalog, and client requests are routed to other instances.

Instances should be distributed across multiple machines to guard against host hardware, VM, OS, or network failures.

ADMINISTERING A HIGHLY AVAILABLE ENVIRONMENT.

MONITORING

Administrators can monitor the status of clustered environments using different tools.

The primary interface is SAS Environment Manager. The Dashboard page, shown in Figure 11, provides a summary view of each instance of every registered service across all machines. Figure 12 shows the Machines page, which gives a detailed view by host. To gather information for both pages, SAS Environment Manager uses the monitoring microservice, which, in turn, queries the SAS Configuration Server to get services and hosts statuses.

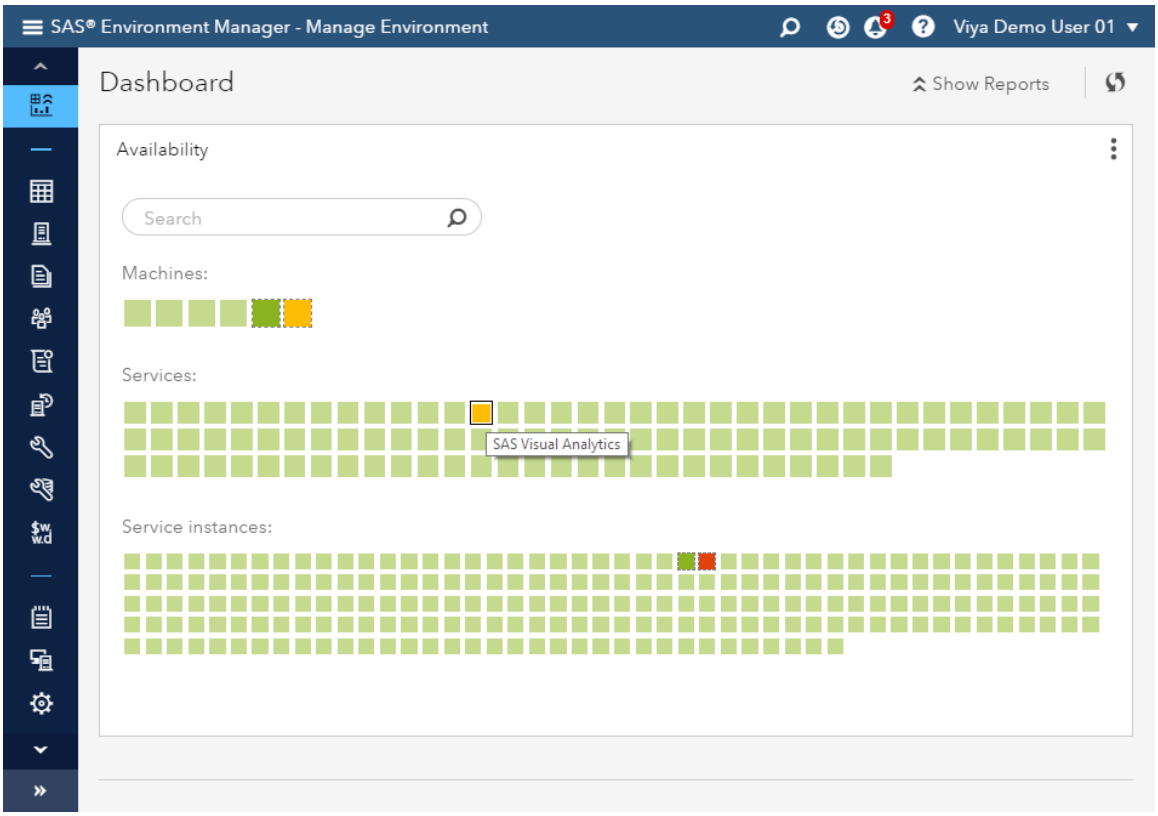

Figure 11 - SAS Environment Manager Dashboard Page

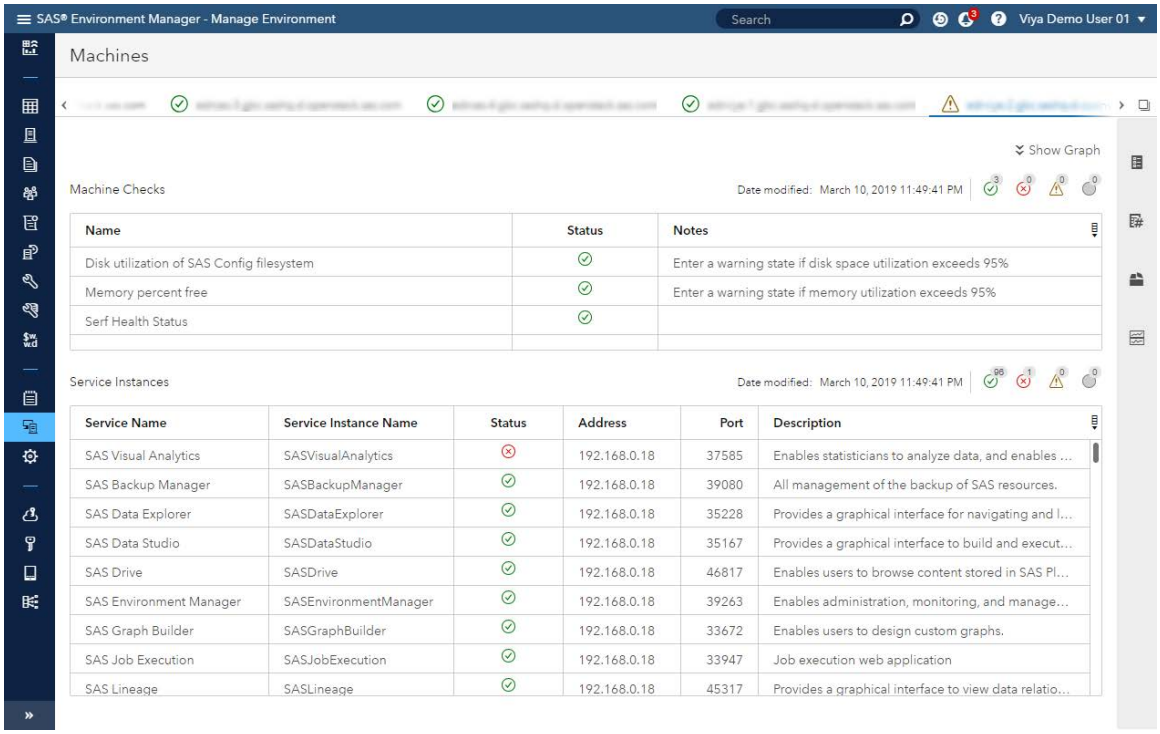

Figure 12 - SAS Environment Manager Machines Page

Services deregister themselves from the SAS Configuration Server when they are intentionally shut down so that it is possible to make a distinction between "intentionally shut down" and "crashed."

Therefore, it is important to understand that SAS Environment Manager reports a service instance as "down" only when it becomes unavailable because of a failure and stops answering to SAS Configuration Server health checks. This is not the case with service instances that are properly stopped: after they deregister themselves from the SAS Configuration Server, they simply disappear from the dashboard and from the machines page. When all services deployed on a machine are properly stopped, the whole machine disappears from the dashboard.

To get a comprehensive status of all services, including the ones that have been properly stopped by an administrator, you can use the sas-viya-all-services command-line utility, which gives a listing similar to the one shown in Output 1.

```
$ sudo /etc/init.d/sas-viya-all-services status
Getting service info from consul...
  Service                          Status  Host           Port     PID
  sas-viya-consul-default          up      N/A             N/A    1129
  sas-viya-vault-default           up      192.168.0.18   8200    1413
  sas-viya-httpproxy-default       up      N/A             N/A    5754
  sas-viya-rabbitmq-server-default up      192.168.0.18   5671    None
  sas-viya-sasstudio-default       up      N/A             N/A    9584
  sas-viya-spawner-default         up      N/A             N/A    9808
  sas-viya-runlauncher-default     up      192.168.0.18   5284   10275
  sas-viya-alert-track-default     up      192.168.0.18      0   10809
  sas-viya-authorization-default   up      192.168.0.18   0928   11140
  sas-viya-cachelocator-default    down    None           None   11165
```

```
  sas-viya-cacheserver-default       up      192.168.0.18  33132   11190
  sas-viya-configuration-default     up      192.168.0.18  41235   11252
  sas-viya-identities-default        up      192.168.0.18  36207   11263
...
```

Output 1 – sas-viya-all-services Listing Services Status

OPERATING

Starting or stopping SAS Viya services requires following the correct sequence to avoid operations issues. When SAS Viya is deployed on a single machine, it is possible to use the sas-viya-all-services script deployed in the /etc/init.d directory. Currently, when you have a multi-machine deployment, including services replicated on a multi-node cluster, this script cannot be used because it is not capable of orchestrating services across machine boundaries.

In this case it is really important to be familiar with the correct sequence, as detailed in *SAS Viya 3.4 Administration: General Servers and Services.* There is, however, a tool to help with this effort: the Multi-Machine Services Utilities (MMSU), part of the SAS Viya Administration Resource Kit (Viya-ARK). Viya-ARK is a collection of tools and utilities aimed at making SAS Viya deployment and administration easier, safer, and faster. Viya-ARK is hosted as Git repository on the SAS Software GitHub page. It is accessible to anyone. MMSU gives you a set of playbooks to start, stop, and check the status of all SAS Viya services across all the machines identified in the inventory.ini file that was used to deploy the environment. For example, to start all services gracefully, execute the following:

```
ansible-playbook viya-ark/playbooks/viya-mmsu/viya-services-start.yml
```

The start-up/shutdown script for a whole CAS cluster is available only on the primary controller. At start-up, it always attempts to start the primary node as the active instance and the backup node as a standby. Then it connects to and starts all worker nodes.

As an administrator, you can manage CAS Server nodes, including stopping them individually, using the Servers page of SAS Environment Manager, as shown in Figure 13.

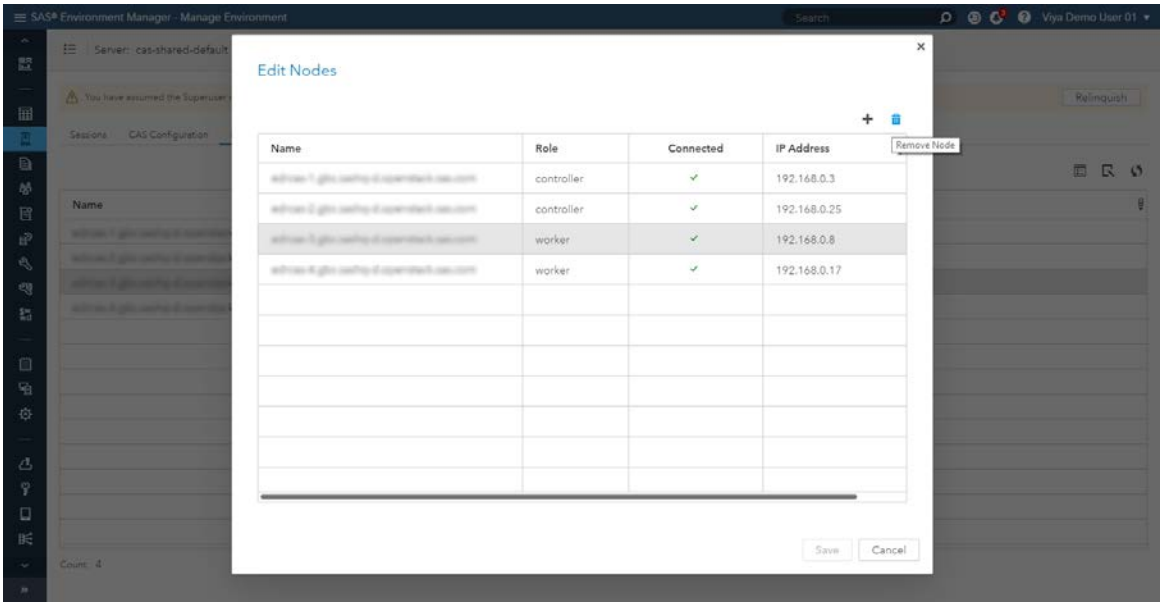

Figure 13 - Operating CAS Nodes from SAS Environment Manager

SERVICES FAILURE AND RECOVERY OPTIONS

To design an architecture that minimizes the impact on end users, it is important to understand what happens when a component fails, and how users can recover from failure.

SAS CLOUD ANALYTICS SERVICES

SAS Cloud Analytics Services can survive failures of worker nodes in MPP deployments thanks to data replication. When you load data into CAS, you can specify the number of redundant blocks, or inactive copies, to create. In the event of node failure, a surviving node accesses the data from the redundant block. Among all copies of the same data segment, only one block on only one node is active at any given time.

Secondly, the CAS controller node can be deployed with a standby instance. When a CAS cluster is started, the primary node is always made the active instance. In case of failure, the CAS controller fails over automatically to the backup controller, with no loss of data, provided a shared file system is in place for /opt/sas/viya/config/data/cas. Once failover occurs, CAS remains operating without fault tolerance for the controller. It would not be possible to automatically fail back to the primary even if it were restarted. Bringing the primary machine back online requires following a particular sequence of steps, which includes a planned outage. For complete instructions, see "Recover a Failed Controller" in *SAS Viya Administration: SAS Cloud Analytic Services*.

With the initial version of SAS Viya 3.4, some termination scenarios for the CAS secondary controller might cause the primary controller to terminate also. For example, when you terminate the secondary controller with the SIGTERM or SIGQUIT signals, the primary controller also stops. Unfortunately, this can also happen during a regular shutdown or reboot of the machine hosting the CAS secondary controller. SAS Problem Note 63522 provides instructions about accessing and applying a software update to correct this issue.

SAS STUDIO 5 IMPROVED RECOVERY

Previous releases of SAS Studio can be clustered for scalability and high availability, providing multiple instances of the web application. However, before SAS Studio 5.*x*, end-user sessions were bound to a specific instance known as a "sticky session."

To understand the disadvantages of a sticky session, consider the following example. Suppose that you have two SAS Studio 4.*x* instances running on server1 and server2. When you sign in to SAS Studio 4.*x*, the front-end Apache HTTP Server directs you to server2. Until you sign out, your session remains on server2. If server2 abnormally shuts down—even though server1 is unaffected—your session is lost, and you must sign out, possibly losing your work. When you sign in again, Apache HTTP Server redirects you to the available server, in this case, server1.

There is also another, more subtle issue. Each instance of SAS Studio 4.*x* can access only an object spawner and workspace servers running locally on the same machine. Suppose that all SAS Studio 4.*x* instances are running fine, but the object spawner on server2 goes down. Even if your SAS Studio 4.*x* session is unaffected, you are unable to perform any work. If you decide to sign out and back in, there is a chance that you will again be routed to server2, because Apache HTTP Server continues to see SAS Studio 4.*x* active on server2.

SAS Studio 5.*x* solves all these issues. Because it is a stateless microservice, Studio 5.x end-user sessions are not bound to any specific instance. If one instance stops abnormally, SAS Studio 5.*x* continues working using another microservice instance. Also, to run SAS code, SAS Studio 5.*x* does not use an object spawner and workspace server. Instead, Studio 5.*x* submits code to a Compute Server started by a launcher server. The compute and launcher server infrastructure ensures that there are no dependencies to a specific

server machine. Therefore, SAS Studio 5.*x* can use a launcher and Compute Server located on any machine.

INFRASTRUCTURE SERVERS

Infrastructure servers are stateful services and are at the core of SAS Viya; yet, the SAS® platform is quite resilient in surviving service failure. Only in a limited number of cases has the failure of a single service caused the whole system to become unavailable. The most critical time is during start-up; if any of the stateful services fails to start, the whole platform might be unable to become fully operational. After a successful start-up, however, if any of the infrastructure servers fails, the rest of the platform can be resilient enough to stay operative and keep servicing end users, even if in a diminished capacity. As described in the previous section, a way to guard against server failures is to cluster them all. Then, if an instance fails, after the cause of a service failure has been solved, you can simply restart it. With some applications, users might have to refresh their pages, but no work should ever be lost. Usually, failures do not cascade to other instances or to different servers, and no other services have to be restarted.

An incorrect termination of PostgreSQL can leave it in a status where it refuses to restart. In this case, see recovery instructions in "Recover a Failed Cluster" and "Failback a Cluster" in *SAS Viya 3.4 Administration: Infrastructure Servers.*

MICROSERVICES

Microservices are stateless, independent, and are designed to remain operational even when required resources become unavailable. SAS Viya applications can still work when all instances of a microservice fail: a specific functionality might be unavailable until the failure is resolved, but everything else keeps working. For example, suppose that no instance of the folders microservice is available. Users cannot navigate folders, open reports, or save new reports. Yet, they can still consult reports that were already open, and keep building new reports. As soon as one instance of the folders microservice comes back online, any report can be successfully saved, and no work is lost.

Obviously, clustering provides a great safeguard even from this eventuality.

There is one service whose failure can be critical to others: SAS Cache Locator. Most microservices are stateless in part because they share a distributed cache across all their cluster instances. They use the SAS Cache Locator to discover existing members of the cluster. Microservices, which are locator clients, fetch the list of locators at client start. If a single locator fails or is restarted, the client has a list with other locators and continues to run. If all locators fail or are restarted within a short time, the client no longer can contact a locator. Clients must then be restarted to reload the list of locators from the SAS Configuration Server. Otherwise, service clients (if not restarted) can fall back to isolated mode and not coordinate the cache with other instances of the same service, giving unpredictable results.

CONCLUSION

SAS Viya resiliency is getting better with each new version. For most components, if they fail, you can simply restart them. Most of the services are independent for almost every operation; in case of a single failure, service recovery does not require a restart of any other service--key exception, at start-up. Almost all components can be easily and automatically clustered, so that, if a member of a cluster fails, the system as a whole is unaffected. CAS Server can replicate data so that, if a worker node fails, data tables are available from other nodes.

REFERENCES

Bourn, Eric. 2018. "SAS® Viya® Service Layer Architecture Overview." *Proceedings of the SAS Global Forum 2018,* Cary, NC: SAS Institute Inc. Available at https://www.sas.com/content/dam/SAS/support/en/sas-global-forum-proceedings/2018/2272-2018.pdf

Read, Jerry. 2018. "SAS® Viya®: Architect for High Availability Now and Users Will Thank You Later." *Proceedings of the SAS Global Forum 2018,* Cary, NC: SAS Institute Inc. Available at https://www.sas.com/content/dam/SAS/support/en/sas-global-forum-proceedings/2018/1835-2018.pdf

SAS Institute Inc. 2018. SAS® Viya® 3.4 Administration: SAS® Cloud Analytic Services. Cary, NC: SAS Institute Inc. Available https://go.documentation.sas.com/?docsetId=calserverscas&docsetTarget=titlepage.htm&docsetVersion=3.4

SAS Institute Inc. 2019. SAS Note 63804. "A SAS® Message Broker cluster may not recover from a node failure." Available http://support.sas.com/kb/63/804.html

SAS Institute Inc. 2019. SAS Note 63522. "The Primary SAS® Cloud Analytic Services Controller Stops When the Secondary SAS® Cloud Analytic Services Controller Stops." Available http://support.sas.com/kb/63/522.html

ACKNOWLEDGMENTS

I'd like to thank Phil Hopkins https://www.linkedin.com/in/philip-hopkins-b176a12/ for sharing some of his experience and research material.

My thanks to many colleagues in SAS R&D who have shared valuable information, including key settings for use within this paper. Thank you all.

RECOMMENDED READING

- SAS Institute Inc. 2019. *SAS Viya 3.4 for Linux: Deployment Guide.* Cary, NC: SAS Institute Inc. Available *https://support.sas.com/en/documentation/install-center/viya/deployment-guides.html*
- SAS Institute Inc. 2018. *SAS Viya 3.4 Administration: Orientation.* Cary, NC: SAS Institute Inc. Available *http://documentation.sas.com/?cdcId=calcdc&cdcVersion=3.4*

CONTACT INFORMATION

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

Edoardo Riva
100 SAS Campus Drive
Cary, NC 27513
SAS Institute Inc.
+1 919 531 7293
edoardo.riva@sas.com
http://www.sas.com