SAS® Viya® 3.3 Administration: Infrastructure Servers

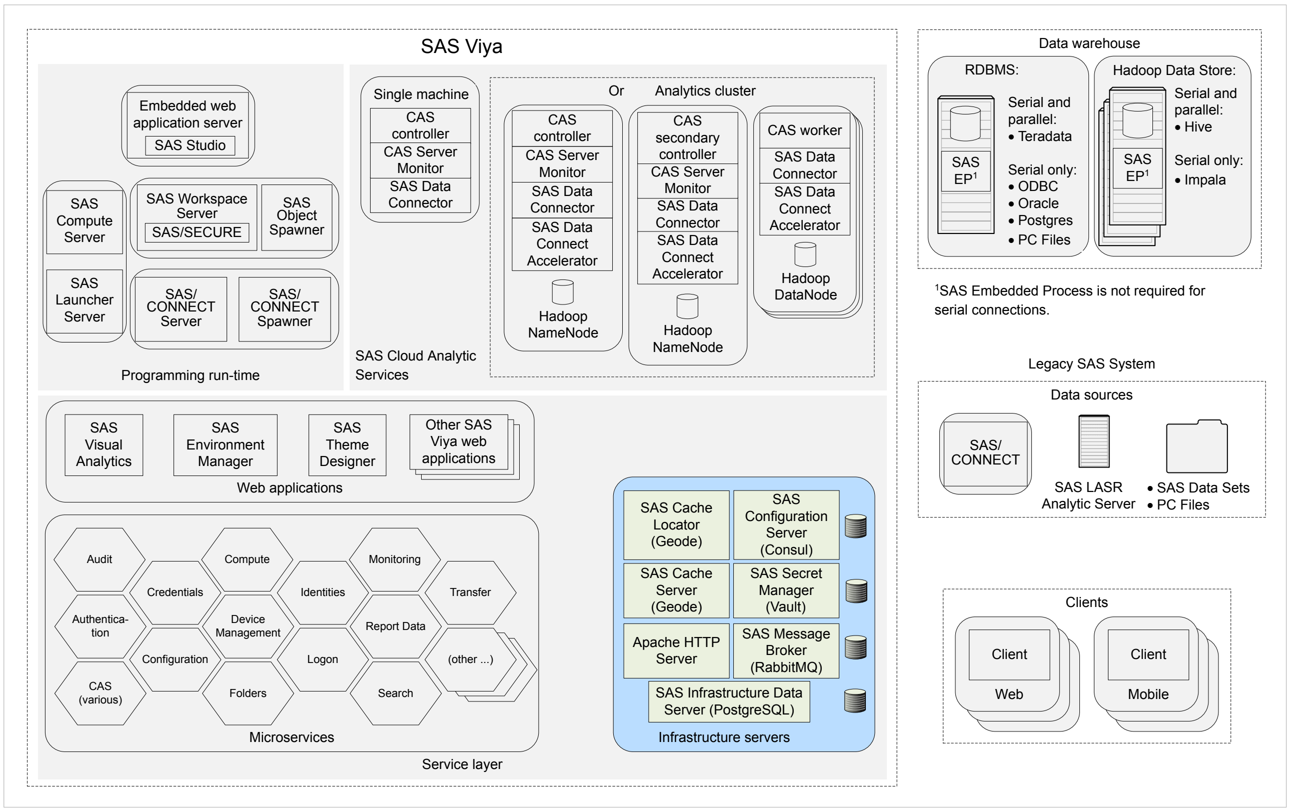
Infrastructure Servers: Overview

Note: A programming-only deployment uses only one of the infrastructure servers—Apache HTTP Server.

SAS Viya contains these infrastructure servers:

- "SAS Configuration Server"
- "SAS Secret Manager"
- "SAS Infrastructure Data Server"
- "SAS Message Broker"
- "SAS Cache Locator and Cache Server"
- "Apache HTTP Server"

Figure A.1 SAS Viya Infrastructure Servers



SAS Configuration Server

Overview

SAS Configuration Server is based on HashiCorp Consul 0.7.5. SAS Configuration Server uses Consul as a service configuration registry that serves as a central repository for configuration data, service discovery, and health status.

Note: A programming-only deployment does not use SAS Configuration Server.

How To

Operate

SAS Viya uses the operating system's default init system or systemd command to launch a script that can stop, start, restart, and check the status of SAS Configuration Server, which is based on Consul. This script, `sas-viya-consul-default`, resides in **`/etc/init.d`**.

Note: You must be signed in to the machine where the configuration server resides, and you must have sudo privileges to run this script.

To operate SAS Configuration Server, run the following command, as appropriate:

```
sas-viya-consul-default status | stop | start | restart
```

Note: For multi-machine deployments, run `sas-viya-consul-default` on every SAS Viya machine. Start or restart SAS Configuration Server *first*. Stop SAS Configuration Server *last*.

Note: You can use a script to manage and view the running state of all SAS Viya services. For more information, see “Start and Stop All Servers and Services” in SAS Viya Administration: General Servers and Services.

Here are a few examples of how to operate this script:

- To check status of SAS Configuration Server using a direct call:

  ```
  sudo /etc/init.d/sas-viya-consul-default status
  ```

- To stop SAS Configuration Server using the Red Hat Linux version 6 init system command:

  ```
  sudo service sas-viya-consul-default stop
  ```

- To start SAS Configuration Server using the Red Hat Linux version 7 systemd command:

  ```
  sudo systemctl start sas-viya-consul-default
  ```

- To restart SAS Configuration Server using a direct call:

  ```
  sudo /etc/init.d/sas-viya-consul-default restart
  ```

Locate Logs

SAS Configuration Server log files are located in **`/opt/sas/viya/config/var/log/consul/default`**.

Concepts

What Is SAS Configuration Server?

SAS Configuration Server is based on HashiCorp’s Consul. Consul is a distributed, highly available registry that contains service configuration data and availability and overall performance (health) information.

Configuration data resides in SAS Configuration Server as key-value pairs. This data is used by SAS Viya microservices at start-up to load default values and to discover any service dependencies.

During run time, whenever a service’s properties change, the service is notified, and it rereads its properties from SAS Configuration Server. (The exceptions are noted in “What Services Must Be Restarted?” in SAS Viya Administration: Configuration Properties.)

Each service registers its health checks when it starts. The Monitoring system periodically queries the status of the health checks.

How Does the SAS Configuration Service Work with SAS Configuration Server?

For information about how the SAS Configuration Service works with SAS Configuration Server, see “How SAS Viya Configuration Works” in SAS Viya Administration: Configuration Properties.

SAS Secret Manager

Overview

SAS Secret Manager is based on HashiCorp Vault 0.6.4. SAS Secret Manager uses Vault to store and generate secrets such as Transport Layer Security (TLS) certificates.

Note: A programming-only deployment does not use SAS Secret Manager. For more information, see "Deployment Types" in SAS Viya Administration: Orientation.

How To

Operate

SAS Viya uses the operating system's default init system or the systemd command to launch a script that can stop, start, restart, and check the status of SAS Secret Manager, which is based on Vault. This script, `sas-viya-vault-default`, resides in **`/etc/init.d`**.

Note: You must be signed in to the machine where configuration server resides, and you must have sudo privileges to run this script.

To operate SAS Secret Manager, run the following command, as appropriate:

`sas-viya-vault-default status | stop | start | restart`

Note: For multi-machine deployments, run `sas-viya-vault-default` on every SAS Viya machine that also contains SAS Configuration Server (Consul). SAS Secret Manager (Vault) is always deployed on the same machine as the Configuration server. (Machines that contain Configuration agents do not have SAS Secret Manager.) Start or restart SAS Secret Manager immediately *after* you run SAS Configuration Server. Stop SAS Secret Manager immediately *before* you stop SAS Configuration Server.

Note: There is a script with which you can manage and view the running state of all SAS Viya services. For more information, see "Start and Stop All Servers and Services" in SAS Viya Administration: General Servers and Services.

Here are a few examples of how to operate this script:

- To check status of SAS Secret Manager using a direct call:

  ```
  sudo /etc/init.d/sas-viya-vault-default status
  ```

- To stop SAS Secret Manager using the Red Hat Linux version 6 init system command:

  ```
  sudo service sas-viya-vault-default stop
  ```

- To start SAS Secret Manager using the Red Hat Linux version 7 systemd command:

  ```
  sudo systemctl start sas-viya-vault-default
  ```

- To restart SAS Secret Manager using a direct call:

  ```
  sudo /etc/init.d/sas-viya-vault-default restart
  ```

Locate Logs

SAS Secret Manager log files are located in **`/opt/sas/viya/config/var/log/vault/default`**.

Concepts

What Is SAS Secret Manager?

SAS Secret Manager is based on HashiCorp Vault. Vault is a distributed, highly available server used to manage secrets. A *secret* is information that you want to secure, such as keys, passwords, certificates, and so on. Vault provides a secure interface to secrets, in addition to access control, and audit logging.

Here are some features of secret manager and examples of how SAS Viya uses them:

- On-demand generation of secrets

Secret manager generates TLS certificates for SAS Viya servers at startup.

- Secure storage for secrets

  Microservices use secure storage so that multiple microservice instances running on the same machine do not request multiple TLS certificates.

- Encrypt and decrypt data without storing it

  SAS Compute Server uses this feature when it sends a password to child processes.

- Revocation of secrets

  SAS Viya services use this feature when rotating security artifacts. (For example, sevices use vault tokens to request TLS certificates).

For more information, see "Concepts" in Encryption in SAS Viya: Data in Motion.

Dependency on SAS Configuration Server (Consul)

SAS Secret Manager is installed on the same machines where SAS Configuration Server (Consul) resides. SAS Configuration Server contains a namespace where secret manager stores secrets in encrypted form, which enables all instances of secret manager access to consistent data. Also, secret manager relies on the configuration server for locking and leader election. Therefore, in order for SAS Secret Manager to be operational, the configuration server must be running.

For information about SAS Secret Manager topology, see "Fault Tolerance in SAS Viya" in SAS Viya Administration: General Servers and Services.

SAS Infrastructure Data Server

Overview

SAS Infrastructure Data Server is based on PostgreSQL version 9 and is configured specifically to support SAS software. SAS Infrastructure Data Server stores user content, such as reports, custom groups, comments, authorization rules, selected source definitions, attachments, audit records, and user preferences.

Note: A programming-only deployment does not use SAS Infrastructure Data Server.

How To (Cluster)

Operate a Cluster

SAS Viya uses the operating system's default init system or systemd command to launch a script that can stop, start, restart, and check the status of the SAS Infrastructure Data Server cluster. (A data server cluster consists of all the PostgreSQL data nodes and Pgpool-II.) The script, `sas-viya-sasdatasvrc-postgres`, resides in **`/etc/init.d`**.

Note: You must be signed in to the machine where the pgpool server resides, and you must have sudo privileges to run the script.

To operate a data server cluster, run the following command, as appropriate:

```
sas-viya-sasdatasvrc-postgres status | stop | start | restart
```

Note: You can use a script to manage and view the running state of all SAS Viya servers and services. For more information, see "Start and Stop All Servers and Services" in SAS Viya Administration: General Servers and Services.

Here are a few examples of how to operate this script:

- To check status of the data server cluster using a direct call:

  ```
  sudo /etc/init.d/sas-viya-sasdatasvrc-postgres status
  ```

- To stop the data server cluster using the Red Hat Linux version 6 init system command:

  ```
  sudo service sas-viya-sasdatasvrc-postgres stop
  ```

- To start the data server cluster using the Red Hat Linux version 7 systemd command:

  ```
  sudo systemctl start sas-viya-sasdatasvrc-postgres
  ```

- To restart the data server cluster using a direct call:

  ```
  sudo /etc/init.d/sas-viya-sasdatasvrc-postgres restart
  ```

Recover a Failed Cluster

The SAS Infrastructure Data Server cluster is considered to be failed under these conditions: when it fails to start, and when all its data nodes are marked as `unhealthy` in the cluster definition file **`/opt/sas/viya/config/etc/sasdatasvrc/postgres/pgpool0/pool.cdf`**.

Common causes for cluster failure include a power failure, network connectivity issues, or a machine reboot. Another cause of cluster failure can be from lack of system resources. Examples are disk space, memory, number of processes, number of open files, ports, semaphores, and shared memory. In such cases, the data server logs contain failure information.

The cluster will not start until the problem that caused the cluster failure has been fixed, and the server nodes are marked as `healthy` in pool.cdf.

On the pgpool server machine, you can manually update pool.cdf, or you can run the repair_postgres_nodes.sh script. After the script updates pool.cdf, it attempts to start the cluster.

1. Before attempting any cluster repair procedures, do the following:
   - Examine the integrity of the data on the data server.

     One or more failovers might have occurred. Therefore, examine the server with the most current data.
   - Back up the data server data directories.
2. Fix the problem that caused the cluster to fail.
3. Make sure that the following servers are running and are accessible:
   - SAS Configuration Server (Consul)
   - SAS Secret Manager (Vault)
4. As the SAS install user (sas) or with sudo privileges, sign in to the pgpool server machine.
5. Choose one of the following methods to update pool.cdf:

   Note: Each method attempts to restart the cluster after pool.cdf has been modified.
   - Run the repair_postgres_nodes.sh script to mark all cluster nodes as healthy in pool.cdf:

     ```
     sudo /opt/sas/viya/home/libexec/sasdatasvrc/script/maintenance/
     repair_postgres_nodes.sh
     ```
   - Using a text editor, open the cluster definition file and mark all of the data server nodes as healthy:

     ```
     vi /opt/sas/viya/config/etc/sasdatasvrc/postgres/pgpool0/pool.cdf
     ```

     Change `node0=unhealthy` to `node0=healthy`.

6 Check the status of the data server cluster.

   A **status** of `up` in the cluster status list indicates that the node is connected and is an active part of the cluster. There should be only one primary server, with no standby servers or one or more standby servers, as appropriate.

Failback a Cluster

Failback refers to the restoration of the high availability (HA) SAS Infrastructure Data Server cluster to its original configuration before the failover.

A PostgreSQL *original configuration* is indicated when the cluster status list displays the following:

- node0 has the **role** of `primary`
- all other nodes have a **role** of `standby`
- all nodes have a **status** of `up`

To failback a SAS Infrastructure Data Server cluster to its original configuration before the failover, follow these steps:

1 Make sure that the following servers are running and are accessible:

   - SAS Configuration Server (Consul)
   - Pgpool server
   - SAS Infrastructure Data Server (PostgreSQL) cluster

2 As the SAS install user (sas) or with sudo privileges, sign in to the pgpool server machine.

3 To ensure that the cluster is in a failover condition, run the following command:

   `sudo service sas-viya-sasdatasvrc-`*`service-name`*` status`

   Verify that node0 has a **status** of `down` and a **role** of `standby`.

   Here is an example where the *service-name* is named **`postgres2`**:

   `sudo service sas-viya-sasdatasvrc-postgres2 status`

   Here is typical output:

```
Checking status of sas-viya-sasdatasvrc-postgres2...

PGPool is running with PID=11445
Checking Postgresql nodes status...
 node_id | hostname | port | status | lb_weight |  role   | select_cnt | load_balance_node | replication_delay
---------+----------+------+--------+-----------+---------+------------+-------------------+-------------------
 0       | machine1 | 5452 | down   | 0.250000  | standby | 1          | false             | 0
 1       | machine2 | 5452 | up     | 0.250000  | primary | 0          | true              | 0
 2       | machine3 | 5452 | up     | 0.250000  | standby | 0          | false             | 0
 3       | machine4 | 5452 | up     | 0.250000  | standby | 0          | false             | 0
(4 rows)
```

4 In the cluster status list, if node0 has a **status** of `up` and a **role** of `standby`, you can go directly to Step 7.

5 To recover (start) node0, run the following command:

   `sudo service sas-viya-sasdatasvrc-`*`service-name`*`-node0 start`

   Here is an example:

   `sudo service sas-viya-sasdatasvrc-postgres2-node0 start`

Here is typical output:

```
Starting sas-viya-sasdatasvrc-postgres2-node0 service...
                                                             [  OK  ]
```

6 Run the following command:

sudo service sas-viya-sasdatasvrc-*service-name* status

Verify that node0 has a **status** of `up`.

Here is an example:

sudo service sas-viya-sasdatasvrc-postgres2 status

Here is typical output:

```
Checking status of sas-viya-sasdatasvrc-postgres2...

PGPool is running with PID=11445
Checking Postgresql nodes status...
 node_id | hostname | port | status | lb_weight |  role   | select_cnt | load_balance_node | replication_delay
---------+----------+------+--------+-----------+---------+------------+-------------------+-------------------
 0       | machine1 | 5452 | up     | 0.250000  | standby | 1          | false             | 0
 1       | machine2 | 5452 | up     | 0.250000  | primary | 0          | true              | 0
 2       | machine3 | 5452 | up     | 0.250000  | standby | 0          | false             | 0
 3       | machine4 | 5452 | up     | 0.250000  | standby | 0          | false             | 0
(4 rows)
```

7 To stop the primary node of the cluster, run the following command:

sudo service sas-viya-sasdatasvrc-*service-name-node-name* stop

Here is an example where *node-name* is named **node1**:

sudo service sas-viya-sasdatasvrc-postgres2-node1 stop

Here is typical output:

```
Stopping sas-viya-sasdatasvrc-postgres2-node1 service...
                                                               [  OK  ]
```

8 Run the following command:

sudo service sas-viya-sasdatasvrc-*service-name* status

Verify that **node0** has a **status** of `up` and a **role** of `primary`.

Here is an example where *service-name* is named **postgres2**:

sudo service sas-viya-sasdatasvrc-postgres2 status

Here is typical output:

```
Checking status of sas-viya-sasdatasvrc-postgres2...

PGPool is running with PID=11445
Checking Postgresql nodes status...
 node_id | hostname | port | status | lb_weight |  role   | select_cnt | load_balance_node | replication_delay
---------+----------+------+--------+-----------+---------+------------+-------------------+-------------------
 0       | machine1 | 5452 | up     | 0.250000  | primary | 1          | false             | 0
 1       | machine2 | 5452 | down   | 0.250000  | standby | 0          | true              | 0
 2       | machine3 | 5452 | up     | 0.250000  | standby | 0          | false             | 0
 3       | machine4 | 5452 | up     | 0.250000  | standby | 0          | false             | 0
(4 rows)
```

Note: The remaining running nodes of the cluster initially show a **status** of `down` while replication is established for the new primary node. Continue to monitor the cluster status until all running nodes have a **status** of `up`.

9 To recover (start) the previous primary node, run the following command:

`sudo service sas-viya-sasdatasvrc-`*`service-name-node-name`*` start`

Here is an example where the previous primary node is named **`node1`**:

`sudo service sas-viya-sasdatasvrc-postgres2-node1 start`

Here is typical output:

```
Starting sas-viya-sasdatasvrc-postgres2-node1 service...

                                                [  OK  ]
```

10 Run the following command:

`sudo service sas-viya-sasdatasvrc-`*`service-name`*` status`

Verify that the cluster has returned to its initial configuration:

- **`node0`** has the **role** of `primary`
- all other nodes have a **role** of `standby`
- all nodes have a **status** of `up`

Here is an example where the data server service is named **`postgres2`**:

`sudo service sas-viya-sasdatasvrc-postgres2 status`

Here is typical output:

```
Checking status of sas-viya-sasdatasvrc-postgres2...

PGPool is running with PID=11445
Checking Postgresql nodes status...
 node_id | hostname | port | status | lb_weight |  role   | select_cnt | load_balance_node | replication_delay
---------+----------+------+--------+-----------+---------+------------+-------------------+-------------------
 0       | machine1 | 5452 | up     | 0.250000  | primary | 1          | false             | 0
 1       | machine2 | 5452 | up     | 0.250000  | standby | 0          | true              | 0
 2       | machine3 | 5452 | up     | 0.250000  | standby | 0          | false             | 0
 3       | machine4 | 5452 | up     | 0.250000  | standby | 0          | false             | 0
(4 rows)
```

Add a Cluster (Ansible)

1 Sign on your Ansible controller with administrator privileges, and locate the file, **`/playbook/vars.yml`**.

2 Using a text editor, open vars.yml and locate the `INVOCATION_VARIABLES` section.

```
INVOCATION_VARIABLES:
  Machine1:
    pgpoolc:
    - PCP_PORT: '5430'
      PGPOOL_PORT: '5431'
      SANMOUNT: '{{ SAS_CONFIG_ROOT }}/data/sasdatasvrc'
      SERVICE_NAME: postgres
    sasdatasvrc:
    - NODE_NUMBER: '0'
      NODE_TYPE: P
      PG_PORT: '5432'
      SANMOUNT: '{{ SAS_CONFIG_ROOT }}/data/sasdatasvrc'
      SERVICE_NAME: postgres
```

3 Copy and paste an existing cluster definition.

In this example, the new cluster is being added to Machine2:

```
INVOCATION_VARIABLES:
   Machine1:
    pgpoolc:
    - PCP_PORT: '5430'
      PGPOOL_PORT: '5431'
      SANMOUNT: '{{ SAS_CONFIG_ROOT }}/data/sasdatasvrc'
      SERVICE_NAME: postgres
    sasdatasvrc:
    - NODE_NUMBER: '0'
      NODE_TYPE: P
      PG_PORT: '5432'
      SANMOUNT: '{{ SAS_CONFIG_ROOT }}/data/sasdatasvrc'
      SERVICE_NAME: postgres

   Machine2:
    pgpoolc:
    - PCP_PORT: '5430'
      PGPOOL_PORT: '5431'
      SANMOUNT: '{{ SAS_CONFIG_ROOT }}/data/sasdatasvrc'
      SERVICE_NAME: postgres
    sasdatasvrc:
    - NODE_NUMBER: '0'
      NODE_TYPE: P
      PG_PORT: '5432'
      SANMOUNT: '{{ SAS_CONFIG_ROOT }}/data/sasdatasvrc'
      SERVICE_NAME: postgres
```

4 Configure the new cluster definition for the pgpool server (pgpoolc) and the data server nodes (sasdatasvrc):

- Pgpool server definition parameters:
  - PCP_PORT

    Specifies the pcp port for the pgpool instance.
  - PGPOOL_PORT

    Specifies the pgpool port. This is the primary port through which all databases connect.
  - SANMOUNT

    Specifies the location of the data files. This path is typically the same value as the other data nodes.

- SERVICE_NAME

  Specifies the unique service name for the data server cluster. `SERVICE_NAME` should be the same for the pgpool server and all nodes in the cluster.

- Data server node definition parameters:
  - NODE_NUMBER

    Specifies the sequential node identifier. The primary node is 0. Standby nodes start at 1 and are incremented sequentially.
  - NODE_TYPE

    Specifies the type of node that you are adding. The primary node should have a value of **`P`**. Standby nodes should have a value of **`S`**.
  - PG_PORT

    Specifies the Postgres database port. The pgpool server communicates with the database on this port. Clients use the `PGPOOL_PORT`. The port must be available for use on the deploy target.
  - SANMOUNT

    Specifies the location of the data files. This path is typically the same value as the other data nodes.
  - SERVICE_NAME

    Specifies the unique service name for the data server cluster. `SERVICE_NAME` should be the same for the pgpool server and all the nodes in the cluster.

Here is an example:

```
INVOCATION_VARIABLES:
  Machine2:
    pgpoolc:
    - PCP_PORT: '5430'
      PGPOOL_PORT: '5431'
      SANMOUNT: '{{ SAS_CONFIG_ROOT }}/data/sasdatasvrc'
      SERVICE_NAME: postgres2
    sasdatasvrc:
    - NODE_NUMBER: '0'
      NODE_TYPE: P
      PG_PORT: '5432'
      SANMOUNT: '{{ SAS_CONFIG_ROOT }}/data/sasdatasvrc'
      SERVICE_NAME: postgres2
    - NODE_NUMBER: '1'
      NODE_TYPE: S
      PG_PORT: '5432'
      SANMOUNT: '{{ SAS_CONFIG_ROOT }}/data/sasdatasvrc'
      SERVICE_NAME: postgres2
```

5 Run your Ansible playbook using the sitedefault.yml file.

Here is an example:

`ansible-playbook site.yml`

For a complete list of playbook commands, see “Commands” in SAS Viya for Linux: Deployment Guide.

Delete a Node or a Cluster

CAUTION! Do not delete a primary node unless you plan to delete the entire cluster. Doing so would increase chances of introducing data corruption. To delete the primary node, failover the node to a standby node, and wait for all remaining nodes to indicate that they are available. When the nodes are available, it is safe to delete the former primary node. Do not delete a pgpool node without first moving the pgpool node of the cluster. Failure to do so will make the cluster unusable. If you choose to delete the node data using the `-d` option, its data files are deleted. Use caution when deciding to use the `-d` option.

1 As root or with an account that has sudo privileges, sign in to the machine where the node that you want to remove resides.

2 Change the directory to **`/opt/sas/viya/home/libexec/sasdatasvrc/script`**.

3 Run the sds_delete_node.sh script with the following options:

   Note: When the sds_delete_node.sh script runs, it stops the cluster.

   - -s *service-name*
   - -n *cluster-name* | *node-name*
   - -d y | n

   CAUTION! A yes (y) value specifies that the script deletes the node or the cluster data files.

   - -c *absolute-path*/sds_env_var.sh

   Here is an example:

```
sudo ./sds_delete_node.sh -s postgres -n node1 -d y
-c /opt/sas/viya/config/etc/sasdatasvrc/postgres/node1/sds_env_var.sh
```

   Every time the script runs, it generates a new log file in **`/tmp/sds_uninstall_log`**.

4 After the script runs, be sure to delete the node or the cluster definition in the `INVOCATION_VARIABLES` section of vars.yml. For more information, see “Add a Node (Ansible)” on page 15.

How To (Nodes)

Check the Status of a Node

SAS Viya uses the operating system’s default init system or systemd command to launch a script that can check the status of a SAS Infrastructure Data Server node. This script, `sas-viya-sasdatasvrc-postgres-node`*`n`*, resides in **`/etc/init.d`**.

Note: As the SAS install user (sas) or with sudo privileges, you must be signed in to the machine where the node resides.

Note: Each node script is numbered, starting at zero (0).

Run the script:

- Red Hat Linux version 6:

  `sudo service sas-viya-sasdatasvrc-postgres-node`*`n`*` status`

- Red Hat Linux version 7:

  `sudo systemctl status sas-viya-sasdatasvrc-postgres-node`*`n`*

Note: On Red Hat Linux version 7, the first time you run the `sas-viya-sasdatasvrc-postgres-node`*`n`* command, you must run the command twice. Red Hat Linux version 7 has backward compatibility for the system V service, and it does not have initial systemd unit files. The first time the `sas-viya-sasdatasvrc-postgres-node`*`n`* command runs, it builds the service unit file from the system V init file. Therefore, until the unit file is built, the `sas-viya-sasdatasvrc-postgres-node`*`n`* commands do not function properly.

Stop a Node

SAS Viya uses the operating system's default init system or systemd command to launch a script that can stop a SAS Infrastructure Data Server node. This script, `sas-viya-sasdatasvrc-postgres-noden`, resides in **`/etc/init.d`**.

CAUTION! The act of stopping individual nodes changes the cluster state. Stopping the primary node causes a failover to occur in a cluster of two or more nodes. In addition, stopping a standby node removes the node from the active cluster. Stopped nodes must be recovered in order for them to added back to the cluster. The recovery of stopped nodes occurs automatically during node start-up. During failover of the primary node (0), other healthy standby nodes (2 and 3) go through a process of "following" the new primary node (1). During failover, nodes 2 and 3 briefly detach from the cluster and display a status of 3 (unhealthy). Wait several minutes and then recheck the cluster status. Eventually, nodes 2 and 3 should re-attach to the cluster and display a **status** of `up` (healthy).

Note: As the SAS install user (sas) or with sudo privileges, you must be signed in to the machine where the node resides.

Note: Each node script is numbered, starting at zero (0).

Note: On Red Hat Linux version 7, the first time you run the `sas-viya-sasdatasvrc-postgres-noden` command, you must run the command twice. Red Hat Linux version 7 has backward compatibility for the system V service, and it does not have initial systemd unit files. The first time the `sas-viya-sasdatasvrc-postgres-noden` command runs, it builds the service unit file from the system V init file. Therefore, until the unit file is built, the `sas-viya-sasdatasvrc-postgres-noden` commands do not function properly.

Run the script:

- Red Hat Linux version 6:

  ```
  sudo service sas-viya-sasdatasvrc-postgres-noden stop
  ```

- Red Hat Linux version 7:

  ```
  sudo systemctl stop sas-viya-sasdatasvrc-postgres-noden
  ```

Start a Node (Recover a Node)

A SAS Infrastructure Data Server data node is considered to be "unhealthy" when it has a **status** of `down` in the cluster status list. If a PostgreSQL data node has been stopped or has been taken offline, the pgpool server removes this node from the cluster.

When you restart an unhealthy node, pgpool server automatically initiates the node recovery process. To recover an unhealthy data node, follow these steps:

1. Make sure that the following servers are running and accessible:
   - SAS Configuration Server (Consul)
   - pgpool server
   - SAS Infrastructure Data Server (PostgreSQL) cluster

2. As the SAS install user (sas) or with sudo privileges, sign in to the pgpool server machine, and run the following command:

   Note: On Red Hat Linux version 7, the first time you run the `sas-viya-sasdatasvrc-postgres-noden` command, you must run the command twice. Red Hat Linux version 7 has backward compatibility for the system V service, and it does not have initial systemd unit files. The first time the `sas-viya-sasdatasvrc-postgres-noden` command runs, it builds the service unit file from the system V init file. Therefore, until the unit file is built, the `sas-viya-sasdatasvrc-postgres-noden` commands do not function properly.

sudo service sas-viya-sasdatasvrc-*service-name* status

Verify that the unhealthy data node has a **status** of `down` and a **role** of `standby`.

Here is an example where the data server service is named **`postgres2`**:

sudo service sas-viya-sasdatasvrc-postgres2 status

Here is typical output:

Note: In this example, the unhealthy node is **`node0`**.

```
Checking status of sas-viya-sasdatasvrc-postgres2...

PGPool is running with PID=11445
Checking Postgresql nodes status...
 node_id | hostname | port | status | lb_weight |  role   | select_cnt | load_balance_node | replication_delay
---------+----------+------+--------+-----------+---------+------------+-------------------+-------------------
 0       | machine1 | 5452 | down   | 0.250000  | standby | 1          | false             | 0
 1       | machine2 | 5452 | up     | 0.250000  | primary | 0          | true              | 0
 2       | machine3 | 5452 | up     | 0.250000  | standby | 0          | false             | 0
 3       | machine4 | 5452 | up     | 0.250000  | standby | 0          | false             | 0
(4 rows)
```

3 As the SAS install user (sas) or with sudo privileges, sign in to the machine that contains the unhealthy data node.

4 Make sure that the unhealthy node is stopped by running the following command:

sudo service sas-viya-sasdatasvrc-*service-name-node-name* stop

Here is an example where the unhealthy node is named **`node0`**.

sudo service sas-viya-sasdatasvrc-postgres2-node0 stop

Here is typical output:

```
Service sas-viya-sasdatasvrc-postgres2-node0 is not running.
                                                   [  OK  ]
```

5 Recover the node as a standby server by running the following command:

sudo service sas-viya-sasdatasvrc-*service-name-node-name* start

The pgpool server automatically starts the unhealthy node.

A node **status** of `up` indicates that the node is connected and is an active part of the cluster. There should be only one server with a **role** of `primary`, with zero or more servers with a **role** of `standby`.

Here is an example:

sudo service sas-viya-sasdatasvrc-postgres2-node0 start

Here is typical output:

```
Starting sas-viya-sasdatasvrc-postgres2-node0 service...

                                                 [  OK  ]
```

6 Make sure that the node has been successfully added to the cluster by running the following command:
sudo service sas-viya-sasdatasvrc-*service-name* status

Here is an example:

sudo service sas-viya-sasdatasvrc-postgres2 status

Here is typical output:

Note: In this example, the previously unhealthy node (`node0`) has a **status** of `up`.

```
Checking status of sas-viya-sasdatasvrc-postgres2...

PGPool is running with PID=11445
Checking Postgresql nodes status...
 node_id | hostname | port | status | lb_weight |  role   | select_cnt | load_balance_node | replication_delay
---------+----------+------+--------+-----------+---------+------------+-------------------+-------------------
 0       | machine1 | 5452 | up     | 0.250000  | standby | 1          | false             | 0
 1       | machine2 | 5452 | up     | 0.250000  | primary | 0          | true              | 0
 2       | machine3 | 5452 | up     | 0.250000  | standby | 0          | false             | 0
 3       | machine4 | 5452 | up     | 0.250000  | standby | 0          | false             | 0
(4 rows)
```

Note: If starting (recovering) a node fails, refer to the Troubleshooting section on page 28.

Add a Node (Ansible)

Adding a data node to your SAS Infrastructure Data Server cluster consists of modifying the vars.yml file and running your Ansible playbook.

1 With administrator privileges, sign in to your Ansible controller , and locate the file, **`/playbook/vars.yml`**.

2 Using a text editor, open vars.yml and locate the `INVOCATION_VARIABLES` section.

```
INVOCATION_VARIABLES:
  Machine1:
    pgpoolc:
    - PCP_PORT: '5430'
      PGPOOL_PORT: '5431'
      SANMOUNT: '{{ SAS_CONFIG_ROOT }}/data/sasdatasvrc'
      SERVICE_NAME: postgres
    sasdatasvrc:
    - NODE_NUMBER: '0'
      NODE_TYPE: P
      PG_PORT: '5432'
      SANMOUNT: '{{ SAS_CONFIG_ROOT }}/data/sasdatasvrc'
      SERVICE_NAME: postgres
```

3 Copy an existing node definition and place it under the deploy target on which the node will be configured.

Here is an example:

```
INVOCATION_VARIABLES:
  Machine1:
    pgpoolc:
    - PCP_PORT: '5430'
      PGPOOL_PORT: '5431'
      SANMOUNT: '{{ SAS_CONFIG_ROOT }}/data/sasdatasvrc'
      SERVICE_NAME: postgres
    sasdatasvrc:
    - NODE_NUMBER: '0'
      NODE_TYPE: P
      PG_PORT: '5432'
      SANMOUNT: '{{ SAS_CONFIG_ROOT }}/data/sasdatasvrc'
      SERVICE_NAME: postgres
    - NODE_NUMBER: '0'
      NODE_TYPE: P
      PG_PORT: '5432'
      SANMOUNT: '{{ SAS_CONFIG_ROOT }}/data/sasdatasvrc'
      SERVICE_NAME: postgres
```

4 Configure the node definition in order to meet the requirements of the cluster:

- NODE_NUMBER

  Specifies the sequential node identifier. Standby nodes start at 1 and are incremented sequentially.

  For example, if you have only a primary node, the node that you are adding should have a `NODE_NUMBER` of 1. If the last standby node in your cluster has the value of 1, the node that you are adding should have a `NODE_NUMBER` of 2.

- NODE_TYPE

  Specifies the type of node that you are adding. The only acceptable value is `S` (standby). After initial deployment, you cannot add a primary node.

- PG_PORT

  Specifies the Postgres database port. The pgpool server communicates with the database on this port. Clients use the `PGPOOL_PORT`. The port must be available for use on the deploy target.

- SANMOUNT

  Specifies the location of the data files. This path is typically the same value as the other data nodes.

- SERVICE_NAME

  Specifies the service name for the data server cluster. It must be an exact match of the name of the cluster to which you are adding a data node.

Here is an example:

```
INVOCATION_VARIABLES:
  Machine1:
    pgpoolc:
    - PCP_PORT: '5430'
      PGPOOL_PORT: '5431'
      SANMOUNT: '{{ SAS_CONFIG_ROOT }}/data/sasdatasvrc'
      SERVICE_NAME: postgres
    sasdatasvrc:
    - NODE_NUMBER: '0'
      NODE_TYPE: P
      PG_PORT: '5432'
      SANMOUNT: '{{ SAS_CONFIG_ROOT }}/data/sasdatasvrc'
      SERVICE_NAME: postgres
    - NODE_NUMBER: '1'
      NODE_TYPE: S
      PG_PORT: '5432'
      SANMOUNT: '{{ SAS_CONFIG_ROOT }}/data/sasdatasvrc'
      SERVICE_NAME: postgres
```

5 Run your Ansible playbook using the sitedefault.yml file.

Here is an example:

`ansible-playbook site.yml`

For a complete list of playbook commands, see “Commands” in SAS Viya for Linux: Deployment Guide.

Move a Node (Ansible)

Moving a data node to your SAS Infrastructure Data Server cluster consists of modifying the vars.yml file and running your Ansible playbook.

1 With administrator privileges, sign in to your Ansible controller , and locate the file **`/playbook/vars.yml`**.

2 Using a text editor, open vars.yml and locate the `INVOCATION_VARIABLES` section.

```
INVOCATION_VARIABLES:
  Machine1:
    pgpoolc:
    - PCP_PORT: '5430'
      PGPOOL_PORT: '5431'
      SANMOUNT: '{{ SAS_CONFIG_ROOT }}/data/sasdatasvrc'
      SERVICE_NAME: postgres
    sasdatasvrc:
    - NODE_NUMBER: '0'
      NODE_TYPE: P
      PG_PORT: '5432'
      SANMOUNT: '{{ SAS_CONFIG_ROOT }}/data/sasdatasvrc'
      SERVICE_NAME: postgres
    - NODE_NUMBER: '1'
      NODE_TYPE: S
      PG_PORT: '5432'
      SANMOUNT: '{{ SAS_CONFIG_ROOT }}/data/sasdatasvrc'
      SERVICE_NAME: postgres
```

3 Copy the existing node definition and place it under the deploy target to which you want to move the node.

In this example, the deploy target is Machine2:

```
INVOCATION_VARIABLES:
  Machine2:
    pgpoolc:
    - PCP_PORT: '5430'
      PGPOOL_PORT: '5431'
      SANMOUNT: '{{ SAS_CONFIG_ROOT }}/data/sasdatasvrc'
      SERVICE_NAME: postgres2
    sasdatasvrc:
    - NODE_NUMBER: '0'
      NODE_TYPE: P
      PG_PORT: '5432'
      SANMOUNT: '{{ SAS_CONFIG_ROOT }}/data/sasdatasvrc'
      SERVICE_NAME: postgres2
    - NODE_NUMBER: '1'
      NODE_TYPE: S
      PG_PORT: '5432'
      SANMOUNT: '{{ SAS_CONFIG_ROOT }}/data/sasdatasvrc'
      SERVICE_NAME: postgres2
```

4 Configure the node definition to meet the requirements of the cluster:

- NODE_NUMBER

  Specifies the sequential node identifier. This number should change to fit with the target cluster. For example, if the last standby node in the cluster is 1, the node that you are moving should have a `NODE_NUMBER` of 2. If there is only a primary node in the target cluster, the node that you are moving should have a `NODE_NUMBER` of 1.

- NODE_TYPE

  Specifies the type of node that you are moving. The only acceptable value is `S` (standby). After initial deployment, you cannot move a primary node.

- PG_PORT

  Specifies the Postgres database port. The pgpool server communicates with the database on this port. Clients use the `PGPOOL_PORT`. The port must be available for use on the deploy target.

- SANMOUNT

  Specifies the location of the data files. This path is typically the same value as other data nodes.

- SERVICE_NAME

  Specifies the service name for the data server cluster.

  Note: Do not change this value.

5 Run your Ansible playbook using the sitedefault.yml file.

Here is an example:

`ansible-playbook site.yml`

For a complete list of playbook commands, see "Commands" in SAS Viya for Linux: Deployment Guide.

Change the Port Number or the Data Directory for a Node (Ansible)

CAUTION! To avoid data corruption, do not change the port number or the data directory on a primary node.

1 As the SAS install user (sas) or with sudo privileges, sign in to the pgpool machine.

2 To stop the node whose port or directory you want to change, run the following command:

`sudo service sas-viya-sasdatasvrc-`*`service-name`*` stop`

Here is an example:

```
sudo service sas-viya-sasdatasvrc-postgres stop
```

3 To check the status of the node, run the following command:

`sudo service sas-viya-sasdatasvrc-`*`service-name`*` status`

Here is an example:

```
sudo service sas-viya-sasdatasvrc-postgres status
```

Verify that failover has successfully occurred. In the cluster status list, the **status** of the node should be `down`.

4 With administrator privileges, sign in to your Ansible controller, and locate the file**`/playbook/vars.yml`**.

5 Using a text editor, open vars.yml and locate the `INVOCATION_VARIABLES` section.

```
INVOCATION_VARIABLES:
  Machine1:
    pgpoolc:
    - PCP_PORT: '5430'
      PGPOOL_PORT: '5431'
      SANMOUNT: '{{ SAS_CONFIG_ROOT }}/data/sasdatasvrc'
      SERVICE_NAME: postgres
    sasdatasvrc:
    - NODE_NUMBER: '0'
      NODE_TYPE: P
```

```
        PG_PORT: '5432'
        SANMOUNT: '{{ SAS_CONFIG_ROOT }}/data/sasdatasvrc'
        SERVICE_NAME: postgres
      - NODE_NUMBER: '1'
        NODE_TYPE: S
        PG_PORT: '5432'
        SANMOUNT: '{{ SAS_CONFIG_ROOT }}/data/sasdatasvrc'
        SERVICE_NAME: postgres
```

6 Make the necessary changes to the port number or the data directory in the definition for the node:

- NODE_NUMBER

  Specifies the sequential node identifier.

  Note: Do not change this value.

- NODE_TYPE

  Specifies the type of node: `P` (primary) or `S` (standby).

  Note: Do not change this value.

- PG_PORT

  Specifies the Postgres database port. The pgpool server communicates with the database on this port. Clients use the `PGPOOL_PORT`. The port must be available for use on the deploy target.

- SANMOUNT

  Specifies the location of the data files. This path is typically the same value as other data nodes.

- SERVICE_NAME

  Specifies the service name for the data server cluster.

  Note: Do not change this value.

7 Run your Ansible playbook using the sitedefault.yml file.

Here is an example:

```
ansible-playbook site.yml
```

For a complete list of playbook commands, see “Commands” in SAS Viya for Linux: Deployment Guide.

8 Restart all SAS Viya services.

Note: You do not need to restart the SAS Viya servers.

For more information, see “General Servers and Services: Operate” in SAS Viya Administration: General Servers and Services.

9 Check the status of the node. In the cluster status list, the **status** of the node should be `up`.

How To (General)

Get Current Passwords

1 As the SAS install user (sas) or with sudo privileges, sign in to any SAS Infrastructure Data Server machine.

2 Obtain the security token from the configuration server, and set it as an environment variable, using the appropriate command:

- Install user or root accounts:

```
export CONSUL_TOKEN=$(cat /opt/sas/viya/config/etc/SASSecurityCertificateFramework/tokens/consul/\
default/client.token)
```

- With sudo privileges (but not as the install user), install accounts:

  ```
  export CONSUL_TOKEN=$(sudo cat /opt/sas/viya/config/etc/SASSecurityCertificateFramework/tokens/consul/\
  default/client.token)
  ```

3 Run the sas-bootstrap-config script for the data server user ID whose password you want to obtain:

- sas

  ```
  /opt/sas/viya/home/bin/sas-bootstrap-config kv read config/postgres/sas.dataserver.pooluser/common/\
  sr_check_password
  ```

- dbmsowner

  ```
  /opt/sas/viya/home/bin/sas-bootstrap-config kv read config/application/sas/database/postgres/password
  ```

Change User Passwords

The script, sds_change_user_pw.sh, changes SAS Infrastructure Data Server passwords and synchronizes them with SAS Configuration Server (Consul) and configuration files.

CAUTION! To avoid data loss, change the sas user account password only during a scheduled maintenance when users are not accessing SAS Viya. The data server must be running when you change the sas user's password Changing the password for the database user, sas, causes all nodes on the database cluster to restart.

Note: To change the password, you must know the current password. For more information, see "Get Current Passwords".

1 As the SAS install user (sas), sign in to the SAS Infrastructure Data Server Pgpool machine.

  Note: The change user password script requires sudo execution privileges.

2 You can determine the status of your cluster by running the cluster init status command as a service.

3 Run the following command:

  `sudo service sas-viya-sasdatasvrc-`*`service-name`*` status`

  Before you run the change password script, verify that the cluster is in its initial configuration state (running and healthy):

  - node0 has the **role** of `primary`
  - all other nodes have a **role** of `standby`
  - all nodes have a **status** of `up`

  Here is an example where the data server service is named **`postgres2`**:

  `sudo service sas-viya-sasdatasvrc-postgres2 status`

  Here is typical output:

```
Checking status of sas-viya-sasdatasvrc-postgres2...

PGPool is running with PID=11445
Checking Postgresql nodes status...
 node_id | hostname | port | status | lb_weight |  role   | select_cnt | load_balance_node | replication_delay
---------+----------+------+--------+-----------+---------+------------+-------------------+-------------------
 0       | machine1 | 5452 | up     | 0.250000  | primary | 1          | false             | 0
 1       | machine2 | 5452 | up     | 0.250000  | standby | 0          | true              | 0
 2       | machine3 | 5452 | up     | 0.250000  | standby | 0          | false             | 0
 3       | machine4 | 5452 | up     | 0.250000  | standby | 0          | false             | 0
(4 rows)
```

4 Locate the data server environment variables file, sds_env_var.sh, and record its location.

   By default, sds_env_var.sh resides in **`/opt/sas/viya/config/etc/sasdatasvrc/postgres/pgpool0`**.

5 The script prompts for the following information. Have this information ready when you run the script in a later step:

   - database user name
   - current database password
   - new database password

     Note: Your password must conform to the data server password policy on page 30.

6 Using the location of sds_env_var.sh noted in Step 4, run the script using the following command:

```
sudo -Hu sas /opt/sas/viya/home/libexec/sasdatasvrc/script/sds_change_user_pw.sh
-config_path
/opt/sas/viya/config/etc/sasdatasvrc/postgres/pgpool0/sds_env_var.sh
```

   TIP If you run the script from the directory where it resides, you might see several `cannot open [No such file or directory]` messages. This is a known issue, and you can safely ignore these messages.

7 Enter the information that you collected in Step 5 as the script prompts you for it.

   After you provide the values in response to the prompts, the script connects to SAS Configuration Server and updates all instances of the database user password that it finds. Changes made in the configuration server are synchronized with the proper SAS Infrastructure Data Server configuration files. Finally, the script issues the necessary SQL commands in the data server to update the permissions for the database user.

8 To validate that your password has successfully changed, connect to the data server first database, **postgres**, using the PostgreSQL interactive terminal, psql:

   `/opt/sas/viya/home/bin/psql -h` ***`data-server-machine-name`*** **`-U`** ***`user-IDservice-name`***

9 When prompted, enter the new password for dbmsowner.

10 Restart all SAS Viya services.

   For more information, see "Start and Stop All Servers and Services" in SAS Viya Administration: General Servers and Services.

Clean Up after a Hardware Failure

If the machine on which the high availability (HA) SAS Infrastructure Data Server cluster runs was stopped unexpectedly, you might need to perform some cleanup steps after you restart the machine.

These steps involve removing any socket-lock files and any PID files that might have become orphaned after the PostgreSQL and pgpool servers were improperly shut down.

1 As the SAS install user (sas) or with sudo privileges, sign in to the pgpool machine.

2 Stop the HA data server cluster by running the appropriate command:

   - Red Hat Linux version 6:

     ```
     sudo service sas-viya-sasdatasvrc-postgres stop
     ```

   - Red Hat Linux version 7:

     ```
     sudo systemctl stop sas-viya-sasdatasvrc-postgres
     ```

3 Delete any socket-lock file (in the form .s.PGSQL.`xxxx`) or any PID file (in the form *server*.pid) that corresponds to your HA data server cluster ports.

   For the default HA data server instance with one data node, remove the following files:

   - `/tmp/.s.PGSQL.5430`
   - `/tmp/.s.PGSQL.5431`
   - `/tmp/.s.PGSQL.5432`
   - `/tmp/.s.PGSQL.5432.lck`
   - `/opt/sas/viya/config/data/sasdatasvrc/postgres/node0/postmaster.pid`
   - `/opt/sas/viya/config/data/sasdatasvrc/postgres/pgpool0/run/pgpool.pid`

4 Restart the HA data server cluster by running the appropriate command:

   - Red Hat Linux version 6:

     ```
     sudo service sas-viya-sasdatasvrc-postgres start
     ```

   - Red Hat Linux version 7:

     ```
     sudo systemctl start sas-viya-sasdatasvrc-postgres
     ```

Remove a Persistent Lock on a Database Table

Persistent locks on a SAS Infrastructure Data Server database table are caused by an uncommitted transaction or a long running query. To fix this problem, you must identify the process IDs of the client connections that are locking the table and terminate these connections.

1 As the SAS install user (sas) or with sudo privileges, sign in to the pgpool server machine.

2 If you know the PostgreSQL dbmsowner (superuser) password, go to Step 3. Otherwise, follow the steps in “Get Current Passwords”.

3 If you have not already done so, install pgAdmin on any machine (including Microsoft Windows) that has access to the machine that is running the pgpool server.

4 In pgAdmin, perform the following steps:

   a Create a **New Server Registration** object and specify the following information:

      - **Host**: ***`machine-name`***

        The name of the machine on which the pgpool server resides.

      - **Port**: ***`pgpool-client-connection-port`***

        The default port is `5431`.

- **Maintenance DB**: `SharedServices`

  Do not use the default, `postgres`.
- **Username**: *superuser*

  The database superuser. The default is `dbmsowner`.
- **Password**: string

  The superuser password.

b Connect to the pgpool server.

c Highlight the server name, and choose **Tools** ⇨ **Server Status**.

The status panel shows all the client connections in the top panel. The second panel will show the persistent locks.

d Choose **Actions** ⇨ **Refresh** multiple times in order to determine whether the listed locks are transient or persistent. A transient lock disappears, and a persistent lock remains throughout refreshes.

e (Optional) You can cross-reference the process identifiers (PIDs) for the locked tables with the connection listing in order to identify the client (the application name) that has locked the table.

Note: If you choose this option, open a SAS Technical Support track about this issue. Include the **PID**, **Application Name**, **Connection State**, and **Query** (if applicable) from the top connections section. Also include the PID and the persistent locked table names from the **Lock** section.

f To clear the locks, run the **pg_terminate_backend()** command on each PID that has a persistent lock.

To do this, go back to the main pgAdmin panel. Highlight the **SharedServices** database name in the server **Object Browser** and choose **Tools** ⇨ **Query Tool** to open an SQL query execution window.

g Execute the **pg_terminate_backend(__PID__)** command to close each connection that is associated with a table that has a persistent lock.

Here is an example:

```
SELECT pg_terminate_backend(14826);
SELECT pg_terminate_backend(16697);
SELECT pg_terminate_backend(22246);
```

h Select **Tools** ⇨ **Server Status** and refresh the panel (**Actions** ⇨ **Refresh** from the menu).

If all the persistent locks have been removed from the second **Locks** section, the persistent locks are successfully removed.

i Exit pgAdmin.

Routine Maintenance Tasks

Overview

Routine maintenance for a SAS Infrastructure Data Server consists of the following tasks:

- Adhering to a rigid schedule of performing database backups.
- Performing a re-index, vacuuming, and analyzing each table in the database during a maintenance cycle.
- Inspecting the data server logs periodically to make sure that there is no data corruption.
- Removing large orphaned data objects from the database periodically to free disk space.

- Track PostgreSQL software patches, and apply the patches that contain critical fixes, such as the CVE-2017-7547 security patch.

Re-Index, Vacuum, and Analyze Database Tables

Follow these steps to re-index, vacuum, and analyze each table in the SAS Infrastructure Data Serverdatabases. SAS recommends that you perform these steps during a maintenance cycle in order to reduce the chance of a PostgreSQL database command hanging because of a long-term lock on a table. If you encounter a hang condition, to remove the lock, you might need to restart the SAS Infrastructure Data Server.

1 As the SAS install user (sas) or with sudo privileges, sign in to the pgpool server machine.

2 If you know the PostgreSQL dbmsowner (superuser) password, go to Step 3. Otherwise, follow the steps in "Get Current Passwords".

3 Run the following commands to set up the PostgreSQL command-line environment:

```
export PATH=/opt/sas/viya/home/bin:$PATH
```

```
export LD_LIBRARY_PATH=/opt/sas/viya/home/lib:/opt/sas/viya/home/
lib64:$LD_LIBRARY_PATH
```

4 (Optional) Stop all SAS Viya services, and run only SAS Infrastructure Data Server.

5 Run the following commands:

Note: For illustration, `5431` is used as the client connection port, **`/opt/sas/viya/home`** is used as the installation directory, and `dbmsowner` is used as the database superuser. Substitute the values that are appropriate for your site.

> **TIP** To prevent having to enter the superuser password multiple times, you can create a **`~/.pgpass`** file.

- Re-index all databases:

```
./reindexdb -a -p 5431 -h localhost -U dbmsowner
```

- Perform a full vacuum and analyze all databases:

```
./vacuumdb -p 5431 -h localhost -U dbmsowner -f -v -z -a
```

Note: If you encounter a hang condition, you might need to restart the SAS Infrastructure Data Server to remove the lock.

If there were no errors in the previous step, then you are done.

If you stopped the all of the SAS Viya services, then you can now stop the SAS Infrastructure Data Server and restart all the SAS Viya services.

Remove Large Orphaned Data Objects

Large objects in the SAS Infrastructure Data Server database are stored separately from the tables that reference them. When a particular row is updated or deleted, these objects can become orphaned (unattached) from a table. Periodically, these orphaned large objects must be manually removed to free disk space.

1 Create an SQL command file named lo-cleanup.sql with the following content:

```
DROP FUNCTION IF EXISTS sas_lob_cleanup();
CREATE FUNCTION sas_lob_cleanup() RETURNS VOID AS $function$
DECLARE
   possible_lob_col record;
   possible_oid_row record;
   possible_oid_val bigint;
BEGIN
   DROP TABLE IF EXISTS sas_possible_lobs;

   CREATE TABLE sas_possible_lobs (table_schema TEXT NOT NULL, table_name
   TEXT NOT NULL, column_name TEXT NOT NULL, column_value BIGINT);

   FOR possible_lob_col IN SELECT * FROM information_schema.columns WHERE
   udt_name IN ('int4', 'int8', 'numeric', 'oid', 'text', 'varchar',
   'char', 'lo') AND NOT (table_schema = 'pg_catalog' AND table_name =
   'pg_shdepend') AND NOT (table_schema = 'pg_catalog' AND table_name =
   'pg_largeobject') AND table_name != 'sas_possible_lobs'

   LOOP

      BEGIN

          FOR possible_oid_val IN EXECUTE 'SELECT CAST(' ||
      possible_lob_col.column_name || ' AS BIGINT) FROM ' ||
      possible_lob_col.table_schema || '.' || possible_lob_col.table_name
      || ' WHERE ' || possible_lob_col.column_name || ' IS NOT NULL AND
      CAST(' || possible_lob_col.column_name || ' AS BIGINT) < ((2 ^ 32) -
      1) AND CAST(' || possible_lob_col.column_name || ' AS BIGINT) > 0'

          LOOP

          --raise notice 'successfully cast % % %',
          possible_lob_col.table_schema,
          possible_lob_col.table_name,
          possible_lob_col.column_name;

              INSERT INTO sas_possible_lobs (table_schema, table_name,
          column_name, column_value) VALUES (possible_lob_col.table_schema,
          possible_lob_col.table_name, possible_lob_col.column_name,
          possible_oid_val);

           END LOOP;

           EXCEPTION

               WHEN cannot_coerce THEN

                  --RAISE NOTICE 'error coercing % % %',
          possible_lob_col.table_schema, possible_lob_col.table_name,
          possible_lob_col.column_name;

               WHEN invalid_text_representation THEN

                  --RAISE NOTICE 'error casting % % %',
          possible_lob_col.table_schema, possible_lob_col.table_name,
          possible_lob_col.column_name;

               WHEN others THEN

                  RAISE NOTICE 'unexpected failure';
```

```
            END;

        END LOOP;

        SELECT LO_UNLINK(lo.loid) FROM pg_catalog.pg_largeobject lo GROUP BY loid
        HAVING (NOT EXISTS (SELECT 1 FROM public.sas_possible_lobs pl WHERE lo.loid
        = pl.column_value));

        DROP TABLE IF EXISTS sas_possible_lobs;

    END;

    $function$ LANGUAGE PLPGSQL;

    SELECT sas_lob_cleanup();
```

2 As the SAS install user (sas) or with sudo privileges, sign in to the pgpool server machine

3 Run the following command for each database:

   Note: For illustration, `5431` is used as the client connection port, **`/opt/sas/viya/home`** is used as the installation directory, and `dbmsowner` is used as the database superuser. Substitute the values that are appropriate for your site.

   ```
   psql -p 5431 -h localhost -U dbmsowner -d postgres -a -f lo-cleanup.sql
   ```

   ```
   psql -p 5431 -h localhost -U dbmsowner -d SharedServices -a -f lo-cleanup.sql
   ```

Apply the CVE-2017-7547 Security Patch

A new security patch, CVE-2017-7547, fixes a password security issue in PostgreSQL databases.

Sites that deploy SAS Viya 3.3 have a newer version of PostgreSQL (version 9.4.13) that contains the fix for this security issue.

However, sites running SAS Viya 3.2 (and earlier) that upgrade to SAS Viya 3.3, must manually apply patch SAS Infrastructure Data Server with patch CVE-2017-7547.

1 Make sure that you have upgraded to SAS Viya 3.3.

2 As the SAS install user (sas) or with sudo privileges, sign in to the pgpool server machine.

3 If you know the PostgreSQL dbmsowner (superuser) password, go to Step 4. Otherwise, follow the steps in "Get Current Passwords".

4 Stop all SAS Viya services, and run only the SAS Infrastructure Data Server.

5 Run the CVE maintenance script:

   ```
   sudo /opt/sas/viya/home/libexec/sasdatasvrc/script/maintenance/CVE-2017-7547.sh
   ```

6 Stop the SAS Infrastructure Data Server and restart all the SAS Viya services.

Concepts

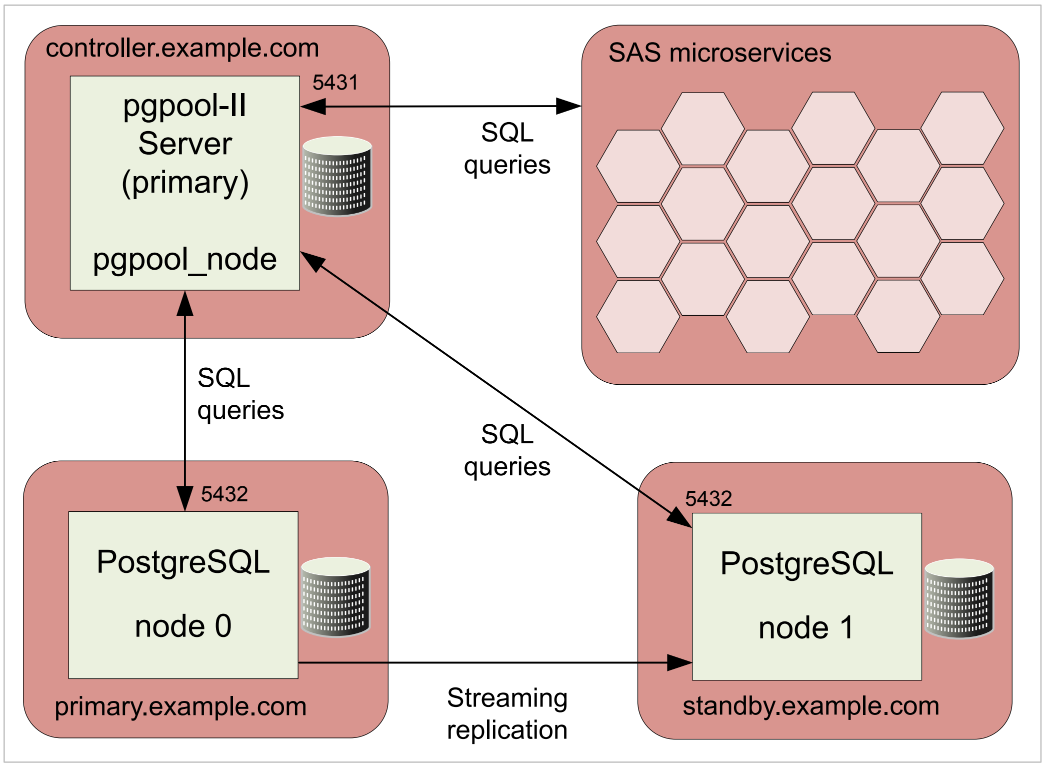
What is the SAS Infrastructure Data Server?

SAS Infrastructure Data Server is used for transactional storage by SAS middle-tier software. It is also used by some SAS solutions software for user content such as reports, custom groups, comments, authorization rules, selected source definitions, attachments, and user preferences. The server is configured specifically to support SAS software, and is based on PostgreSQL version 9.

By default, the SAS installer account is used to start the server.

The databases that are managed by the server are backed up and restored with the Backup and Recovery Deployment Tool. For more information, see *SAS Viya Administration: Backup and Restore*.

Figure A.2 *SAS Infrastructure Data Server Architecture*



Pgpool-II

SAS provides Pgpool-II (version 3) open-source software to enable you to manage PostgreSQL clusters. The Pgpool-II software resides and operates between SAS Infrastructure Data servers and clients. All data connections and database requests are routed through the pgpool service.

- High availability (failover management)
- Load balancing (read SELECT query scaling, but not writes)
- Connection pooling

Troubleshooting

psql: server closed the connection unexpectedly. This probably means the server terminated abnormally before or while processing the request.

Explanation:

The SAS Viya environment was shut down abnormally.

Resolution:

Restart the SAS Viya environment using the sas-viya-all-services start command. For more information, see "Start and Stop All Servers and Services" in SAS Viya Administration: General Servers and Services.

/opt/sas/viya/config/etc/sasdatasvrc/../node.cdf was already marked with 'recoveryInProgress=y'. Exiting from auto-recovery.

Explanation:

PostgreSQL was in the process of recovering the SAS Infrastructure Data Server node when it encountered an error, and stopped the recovery process. Whenever it restarts a data server node, PostgreSQL always inserts the line, `recoveryInProgress=y`, in the node.cdf file to avoid a simultaneous recovery.

Resolution:

1. Review the recovery log to determine what the problem is.

   (The recovery log is located here: **`/opt/sas/viya/config/var/log/sasdatasvrc/cluster/nodex/sds_auto_recovery_node.log`**.)

2. Fix the problem.

3. Remove the following line from the node's node.cdf file:

   ```
   recoveryInProgress=y
   ```

4. Restart (recover) the node.

EDTERROR: missing chunk number 0 for toast value 9558737 in pg_toast_2619
EDTCONTEXT: automatic analyze of table "SharedServices.public.sas_audit"
EDTERROR: could not read block 3062 in file "base/18797/19703": read only 0 of 8192 bytes
***yyyy-mm-dd* EDTERROR: unexpected data beyond EOF in block 0 of relation base/16715/107679**

Explanation:

There is a high probability that your SAS Infrastructure Data Server database is corrupted.

Resolution:

After you correct the cause of the data corruption, and recover the database using a restored backup.

ERROR: Cluster stop failed. Please review the log file. /opt/sas/viya/config/var/log/sasdatasvrc/postgres/pgpool0/sas-viya-sasdatasvrc-postgres-service_YYYYMMDD_####.log [FAILED]
Unexpected response code: 500 ERROR: Unable to read a key Unexpected response code: 500 (rpc error: failed to get conn: dial tcp someotherhost.com:8300: getsockopt: connection refused) ERROR: Unable to list the nodes that provide the service 'postgres'

Explanation:

SAS Infrastructure Data Server fails to stop because it cannot connect to SAS Configuration Server (Consul) even though Consul is running.

This problem occurs in a multi-machine deployment where the primary data server node is not on the same system as the primary Consul server (the server designated in the inventory.ini file). If the primary Consul server has already been shut down, the data server fails to stop, even if a local Consul service is running.

Resolution:

Always stop SAS Infrastructure Data Server before the primary Consul server, regardless of which machine it is located on. For more information, see “Read This First: Start and Stop Servers and Services” in SAS Viya Administration: General Servers and Services.

Reference

Recommended Connection, ulimit, and Semaphore Settings

Connection Settings

To change the number of connections available to clients, modify the following configuration properties:

- sas.dataserver.conf.common.max_connections

  For more information, see https://www.postgresql.org/docs/9.1/static/runtime-config-connection.html#RUNTIME-CONFIG-CONNECTION-SETTINGS.

- sas.dataserver.pool.common.num_init_children

  For more information, see http://www.pgpool.net/docs/pgpool-II-3.5.4/doc/pgpool-en.html#NUM_INIT_CHILDREN.

Note: The `max_connections` value should be slightly higher than the `num_init_children` value to allow for direct connections outside pgpool for administrative use, such as backup and recovery.

ulimit Settings

Recommended settings for the `sas` user in **`/etc/security/limits.conf`**:

```
sas soft nofile 150000
sas hard nofile 150000
sas soft nproc 100000
sas hard nproc 100000
sas soft stack 10240
sas hard stack 10240
```

Semaphore Settings

Recommended semaphore settings in **`/etc/sysctl.conf`**:

```
kernel.sem=512 32000 100 1024
 net.core.somaxconn=2048

for SEMMSL, SEMMNS, SEMOPM, and SEMMNI
```

For more information, including formulas and minimum values, see https://www.postgresql.org/docs/9.5/static/kernel-resources.html.

Note: Changing Linux semaphore settings requires a machine reboot.

Note: You might need to adjust additional Linux operating system settings in order to support these recommended ulimit and semaphore settings. To optimize your PostgreSQL resources, you should also scale the server's working memory settings in accordance with Tuning the PostgreSQL Data Server in *SAS Web Applications: Tuning for Performance and Scalability*.

Database

TIP All PostgreSQL data servers have a *first database* named **postgres**. For more information, see Creating a Database in PostgreSQL documentation.

In a SAS Viya deployment, SAS Infrastructure Data Server is configured to manage the SharedServices database. SAS Viya microservices create database schemas within SharedServices.

If your deployment includes SAS solutions software that supports SAS Infrastructure Data Server, more databases might be configured on the server.

Default Users

dbmsowner
: The PostgreSQL database owner and the SAS database administrator user.

sas
: The SAS Viya install user and the account used for SAS Infrastructure Data Server cluster management.

Network Access

SAS Infrastructure Data Server is configured to accept connections on all network interfaces, and it requires password authentication. By default, SAS configures the server to use network port number 5431.

PostgreSQL instances are configured with JDBC data sources that reference the SharedServices database.

Password Policy

The user name and password for the SAS Infrastructure Data Server administrator are specified during deployment. The password can be updated. Passwords for SAS Infrastructure Data Server are subject to the following guidelines:

- The password must not contain any non-alphanumeric characters.

  Examples are underscores (_), hyphens (-), and periods (.).
- The password must be at least six characters long.
- The password can contain alphanumeric characters.
- There are no restrictions for including numbers or mixed-case characters.

Environment Parameters

Export the following path in order to execute PostgreSQL and the pgpool commands:

```
export LD_LIBRARY_PATH=/opt/sas/viya/home/lib:/opt/sas/viya/home/lib64
```

Configuration Files

- `/opt/sas/viya/config/etc/sasdatasvrc/postgres/node0/node.cdf`
- `/opt/sas/viya/config/etc/sasdatasvrc/postgres/pgpool0/pool.cdf`
- `/opt/sas/viya/config/data/sasdatasvrc/postgres/pgpool0/pgpool.conf`
- `/opt/sas/viya/config/data/sasdatasvrc/postgres/pgpool0/pcp.conf`
- `/opt/sas/viya/config/data/sasdatasvrc/postgres/pgpool0/pool_hba.conf`
- `/opt/sas/viya/config/data/sasdatasvrc/postgres/pgpool0/pool_passwd`

- **/opt/sas/viya/config/data/sasdatasvrc/postgres/node0/postgresql.conf**
- **/opt/sas/viya/config/data/sasdatasvrc/postgres/node0/pg_hba.conf**

Log Files

SAS Infrastructure Data Server log files are located in **/opt/sas/viya/config/var/log/sasdatasvrc**.

SAS Message Broker

Overview

SAS uses a set of event APIs that are dependent on Spring Integration and Spring AMQP to interact with the message broker. The AMQP-compliant message broker that SAS uses is Pivotal’s RabbitMQ, version 3. RabbitMQ includes the Erlang platform, version 19.

Note: A programming-only deployment does not use SAS Message Broker.

How To

Operate

SAS Viya uses the operating system’s default init system or systemd command to launch a script that can stop, start, restart, and check the status of SAS Message Broker. This script, `sas-viya-rabbitmq-server-default`, resides in **/etc/init.d**.

Note: You must be signed in to the machine where message broker resides, and you must have sudo privileges to run this script.

To operate the message broker, run the following command, as appropriate:

sas-viya-rabbitmq-server-default status | stop | start | restart

Note: You can also run a script to manage and view the running state of all SAS Viya servers and services. For more information, see “Start and Stop All Servers and Services” in SAS Viya Administration: General Servers and Services.

Here are a few examples of how to operate this script:

- To check status of the message broker using a direct call:

  ```
  sudo /etc/init.d/sas-viya-rabbitmq-server-default status
  ```

- To stop the message broker using the Red Hat Linux version 6 init system command:

  ```
  sudo service sas-viya-rabbitmq-server-default stop
  ```

- To start the message broker using the Red Hat Linux version 7 systemd command:

  ```
  sudo systemctl start sas-viya-rabbitmq-server-default
  ```

- To restart the message broker using a direct call:

  ```
  sudo /etc/init.d/sas-viya-rabbitmq-server-default restart
  ```

Concepts

What is SAS Message Broker?

SAS Message Broker is an integral part of the event-driven architecture in which SAS Viya services participate. SAS uses a set of event APIs that are dependent on Spring Integration and Spring AMQP for interacting with the message broker. The AMQP-compliant message broker that SAS uses is Pivotal's RabbitMQ. The SAS event APIs provide a layer of abstraction between the message broker and its clients. The SAS event APIs also prevent code from breaking, which could result if SAS changed its third-party message broker from RabbitMQ to another third-party message broker in the future.

How Does Message Broker Work?

SAS Message Broker accepts messages in a standard format and routes them through exchanges and queues, which provide transaction acknowledgment, message persistence, and redundancy. Message broker exchanges accept messages from publishers and route them to queues, as appropriate. The exchange type controls whether messages are sent to a specific queue, to all associated queues, or only to queues that accept a particular message routing key or that match a key pattern.

SAS Message Broker Reference

Exchanges

SAS Message Broker uses the following exchanges:

- sas.application
- sas.application.backup
- sas.backup.topic
- sas.ledger
- sas.log
- sas.metric
- sas.notification
- sas.search.schema.topic

Configuration Files

Note: Change these configuration files only when instructed to do so by SAS Technical Support.

- `/opt/sas/viya/config/etc/rabbitmq-server/rabbitmq.config`
- `/opt/sas/viya/config/etc/rabbitmq-server/rabbitmq-env.conf`

Log Files

SAS Message Broker log files are located in `/opt/sas/viya/config/var/log/rabbitmq-server/default`.

SAS Cache Locator and Cache Server

Overview

SAS Cache Locator and SAS Cache Server provide a distributed cache technology to microservices in SAS Viya.

Operate

SAS Viya uses the operating system's default init system or systemd command to launch scripts that can stop, start, restart, and check the status of SAS Cache Locator and SAS Cache Server. These scripts, `sas-viya-cachelocator-default` and `sas-viya-cacheserver-default`, reside in **`/etc/init.d`**.

Note: You must be signed in to the machine where SAS Cache Locator and SAS Cache Server reside, and you must have sudo privileges to run this script.

To operate SAS Cache Locator or SAS Cache Server, run the following command, as appropriate:

`sas-viya-cachelocator-default status | stop | start | restart`

`sas-viya-cacheserver-default status | stop | start | restart`

Note: You can use a script to manage and view the running state of all SAS Viya servers and services. For more information, see "Start and Stop All Servers and Services" in SAS Viya Administration: General Servers and Services.

Here are a few examples of how to operate this script:

- To check status of the cache locator or the cache server using a direct call:

  ```
  sudo /etc/init.d/sas-viya-cachelocator-default status
  ```

  ```
  sudo /etc/init.d/sas-viya-cacheserver-default status
  ```

- To stop the cache locator or the cache server using the Red Hat Linux version 6 init system command:

  ```
  sudo service sas-viya-cachelocator-default stop
  ```

  ```
  sudo service sas-viya-cacheserver-default stop
  ```

- To start the cache locator or the cache server using the Red Hat Linux version 7 systemd command:

  ```
  sudo systemctl start sas-viya-cachelocator-default
  ```

  ```
  sudo systemctl start sas-viya-cacheserver-default
  ```

- To restart the cache locator or the cache server using a direct call:

  ```
  sudo /etc/init.d/sas-viya-cachelocator-default restart
  ```

  ```
  sudo /etc/init.d/sas-viya-cacheserver-default restart
  ```

Concepts

SAS Cache Locator

SAS Cache Locator is a server that provides discovery information to SAS Viya microservices for the purpose of forming a distributed data cache. SAS Cache Locator is based on the open source Apache Geode project.

SAS Cache Server

SAS Cache Server hosts long-lived data regions (a cache) and serves the contents to SAS Viya microservices. Like SAS Cache Locator, SAS Cache Server is based on the open source Apache Geode project.

Configuration

SAS Cache Locator and SAS Cache Server embed the Apache Geode API within their respective SAS Viya microservices, cachelocator and cacheserver.

The cachelocator and cacheserver microservices enable the cache locator and the cache server to gain access to SAS Configuration Server (Consul) in order to dynamically register and to retrieve properties with the SAS Viya Configuration service. For more information, see “Non-Spring-Based Servers” in SAS Viya Administration: Configuration Properties.

When configuration changes are made to cachelocator and cacheserver, you must restart SAS Cache Locator and SAS Cache Server in order for their changes to take effect. For information about how to modify the configuration for cachelocator and cacheserver, see “Edit Configuration Instances” in SAS Viya Administration: Configuration Properties.

Log Files

Log files for SAS Cache Locator and SAS Cache Server are located in **`/opt/sas/viya/config/var/log/cachelocator/default`** and **`/opt/sas/viya/config/var/log/cacheserver/default`**.

Apache HTTP Server

Overview

SAS Viya uses Apache HTTP Server to serve static HTML content and to proxy client connections. A high-availability proxy environment is not installed by default, but is a supported configuration.

Red Hat Linux version 6 uses Apache HTTP Server upstream v2.2 and Red Hat Linux version 7 uses Apache HTTP Server upstream v2.4. For more information, see “Apache httpd” in SAS Viya for Linux: Deployment Guide.

How To

Operate

Note: You must be signed in to the machine where Apache HTTP Server resides, and you must have sudo privileges to run this script.

Note: For complete information about httpd arguments, see https://httpd.apache.org/docs/2.0/programs/httpd.html

- To the operate HTTP Server on Red Hat Linux version 6, use the init system command:

  `sudo service httpd status | stop | start | restart`

  In this example, the following command checks the status of Apache HTTP Server on the current machine:

  ```
  sudo service httpd status
  ```

- To the operate HTTP Server on Red Hat Linux version 7, use the systemd command:

`sudo systemctl status | stop | start | restart httpd`

In this example, the following command stops Apache HTTP Server on the current machine:

```
sudo systemctl stop httpd
```

Change Time-Out Interval

When SAS web applications return HTTP 502 proxy errors, you might have to change the time-out interval for your Apache HTTP Server.

1 Sign in as the SAS install user (sas), or sign on with sudo privileges, to the Apache HTTP Server machine.

2 Using a text editor, open **`/etc/httpd/conf/httpd.conf`**.

3 Modify the `Timeout` and `Keepalive`values as follows:

```
Timeout 2400
Keepalive On
```

4 Add the following two parameters, and save the file:

```
ProxyTimeout 2400
ProxyBadHeader Ignore
```

5 Restart Apache HTTP Server.

Locate Log Files

Note: You must be signed in with sudo privileges to the machine where the service resides in order to view log files.

By default, Apache HTTP Server log files are located in **`/var/log/httpd`**.

Concepts

SAS Viya uses Apache HTTP Server as a web server. HTTP Server serves static HTML content and proxies client communication.

A third-party load balancer is required in order to provide high availability for HTTP Server. You can also install your own web server on a separate machine in order to proxy connections from the internet to HTTP Server. For more information about making HTTP Server highly available, see “Apache httpd” in SAS Viya for Linux: Deployment Guide.