# SAS® Viya® 3.5 Administration: Infrastructure Servers

## Infrastructure Servers: Overview

<table>
<thead>
<tr>
<th>Topic</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>SAS Viya Infrastructure Servers (Full Deployment)</td>
<td>3</td>
</tr>
<tr>
<td>SAS Viya Infrastructure Servers (Programming-only Deployment)</td>
<td>3</td>
</tr>
</tbody>
</table>

## SAS Configuration Server

<table>
<thead>
<tr>
<th>Topic</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Overview</td>
<td>4</td>
</tr>
<tr>
<td>Operate (Linux)</td>
<td>4</td>
</tr>
<tr>
<td>Operate (Windows)</td>
<td>5</td>
</tr>
<tr>
<td>Concepts</td>
<td>7</td>
</tr>
<tr>
<td>Log Files</td>
<td>7</td>
</tr>
</tbody>
</table>

## SAS Secrets Manager (Linux)

<table>
<thead>
<tr>
<th>Topic</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Overview</td>
<td>7</td>
</tr>
<tr>
<td>Operate</td>
<td>8</td>
</tr>
<tr>
<td>Concepts</td>
<td>9</td>
</tr>
<tr>
<td>Log Files</td>
<td>10</td>
</tr>
<tr>
<td>Auditing</td>
<td>10</td>
</tr>
</tbody>
</table>

## SAS Infrastructure Data Server

<table>
<thead>
<tr>
<th>Topic</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Overview</td>
<td>11</td>
</tr>
<tr>
<td>Concepts</td>
<td>11</td>
</tr>
<tr>
<td>How To (Cluster)</td>
<td>12</td>
</tr>
<tr>
<td>How To (General)</td>
<td>32</td>
</tr>
<tr>
<td>Upgrading PostgreSQL in SAS Viya</td>
<td>36</td>
</tr>
<tr>
<td>Routine Maintenance Tasks</td>
<td>42</td>
</tr>
<tr>
<td>Troubleshooting</td>
<td>46</td>
</tr>
<tr>
<td>Reference</td>
<td>47</td>
</tr>
</tbody>
</table>

## SAS Message Broker

<table>
<thead>
<tr>
<th>Topic</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Overview</td>
<td>48</td>
</tr>
<tr>
<td>Operate (Linux)</td>
<td>48</td>
</tr>
<tr>
<td>Operate (Windows)</td>
<td>49</td>
</tr>
<tr>
<td>Add a Node</td>
<td>51</td>
</tr>
<tr>
<td>Remove a Node</td>
<td>51</td>
</tr>
<tr>
<td>Concepts</td>
<td>52</td>
</tr>
<tr>
<td>SAS Message Broker Reference</td>
<td>53</td>
</tr>
</tbody>
</table>
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 Secrets Manager (Linux)”
- “SAS Infrastructure Data Server”
- “SAS Message Broker”
- “SAS Cache Locator and Cache Server”
- “Apache HTTP Server”

The following diagram identifies the infrastructure service components of a SAS Viya full deployment:
Note: SAS Secrets Manager does not run on Windows.

The following diagram shows that a programming-only deployment uses only the Apache HTTP Server from the SAS Viya server layer.

Figure 2  SAS Viya Infrastructure Servers (Programming-only Deployment)
SAS Configuration Server

Overview

SAS Configuration Server is based on HashiCorp Consul. SAS Configuration Server acts 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.

Operate (Linux)

SAS Viya provides a script in /etc/init.d that you use to stop, start, restart, and check the status of SAS Configuration Server. The script is named, sas-viya-consul-default.

Syntax

How you run sas-viya-consul-default depends on your operating system:

- Red Hat Enterprise Linux 7.x (or an equivalent distribution) and SUSE Linux Enterprise Server 12.x:
  
  ```
  sudo systemctl status | stop | start | restart sas-viya-consul-default
  ```

- Red Hat Enterprise Linux 6.x (or an equivalent distribution):
  
  ```
  sudo service sas-viya-consul-default status | stop | start | restart
  ```

Usage Notes and Tips

- You must be logged on to the machine where SAS Configuration Server resides. Also, you must have root-level privileges to run this script.


- If there are multiple instances of SAS Configuration Server, start them in this sequence:

  - First, start sas-viya-consul-default on all machines in the [consul] host group.
    
    The [consul] host group is located in the Ansible playbook inventory.ini file and it defines which machines host the SAS Configuration Server instances.

  - Next, start sas-viya-consul-default on all other machines in the deployment, which launches the agent processes for SAS Configuration Server.

- If there are multiple instances of SAS Configuration Server, stop them in this sequence:

  - First, stop sas-viya-consul-default on all machines not in the [consul] host group.
    
    The [consul] host group is located in the Ansible playbook inventory.ini file.

  - Next, stop sas-viya-consul-default on machines in the [consul] host group.
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.

On Linux systems that support systemd, use the systemctl command when running sas-viya-consul-default. The systemctl command maintains a record of service status that the service command and a direct call does not use.

**CAUTION**

On Red Hat Enterprise Linux 7.x (or an equivalent distribution) and SUSE Linux Enterprise Server 12.x, do not mix System V init and systemd commands. Mixing the System V init (service command) with the systemd (systemctl command) causes several issues. The systemctl command knows nothing about a SAS Viya service started with the service command. If you start sas-viya-consul-default on RHEL 7.x with the service command, and later attempt to shut down SAS Configuration Server using the systemctl command, the configuration server stops responding and does not shut down.

**Examples**

- To check status of SAS Configuration Server on Red Hat Enterprise Linux 7.x (or an equivalent distribution) and SUSE Linux Enterprise Server 12.x:

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

- To stop SAS Configuration Server on Red Hat Enterprise Linux 6.x (or an equivalent distribution):

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

- To start SAS Configuration Server on Red Hat Enterprise Linux 7.x (or an equivalent distribution) and SUSE Linux Enterprise Server 12.x:

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

- To restart SAS Configuration Server on Red Hat Enterprise Linux 6.x (or an equivalent distribution):

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

**Operate (Windows)**

Using the Services snap-in in the Microsoft Management Console, you can start, stop, and restart SAS Configuration Server (Consul).

**Note:** There is one service, SAS Services Manager, that you can use to start and stop all SAS Viya servers and services. SAS Services Manager recognizes the dependencies between services and starts and stops services in the correct sequence. For more information, see “Start and Stop All Servers and Services” in SAS Viya Administration: General Servers and Services.
Because there is a particular sequence in which the servers and services must be started and stopped, the individual services are not configured to run automatically when the SAS Viya machine is booted.

**IMPORTANT** SAS Configuration Service (Consul), SAS Infrastructure Data Server (PostgreSQL), SAS HTTP Proxy Server (Apache HTTP Server), and SAS Message Broker (RabbitMQ) are dependencies for the other SAS Viya services. If you are operating one or more services individually, always start each of these four services first and stop them last.

**Note:** There is one service, SAS Services Manager, that you can use to start and stop all SAS Viya servers and services. SAS Services Manager recognizes the dependencies between services and starts and stops services in the correct sequence. For more information, see “Start and Stop All Servers and Services” in SAS Viya Administration: General Servers and Services.
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.

Log Files

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

SAS Secrets Manager (Linux)

Overview

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

**IMPORTANT** SAS Secrets Manager is not deployed on Windows.

Note: A programming-only deployment does not use SAS Secrets Manager. For more information, see “Types of Deployment” in SAS Viya Administration: Getting Started.
Operate

SAS Viya provides a script in `/etc/init.d` that you use to stop, start, restart, and check the status of SAS Secrets Manager. The script is named, `sas-viya-vault-default`.

Syntax
How you run `sas-viya-vault-default` depends on your operating system:

- Red Hat Enterprise Linux 7.x (or an equivalent distribution) and SUSE Linux Enterprise Server 12.x:
  ```bash
  sudo systemctl status | stop | start | restart sas-viya-vault-default
  ```
- Red Hat Enterprise Linux 6.x (or an equivalent distribution):
  ```bash
  sudo service sas-viya-vault-default status | stop | start | restart
  ```

Usage Notes and Tips
- You must be logged on to the machine where SAS Secrets Manager resides. Also, you must have root-level privileges to run this script.
- For multi-machine deployments, run `sas-viya-vault-default` on every SAS Viya machine that also contains SAS Configuration Server (Consul). SAS Secrets Manager (Vault) is always deployed on the same machine as the Configuration server. (Machines that contain Configuration agents do not have SAS Secrets Manager.)
  
  Start or restart SAS Secrets Manager immediately after you run SAS Configuration Server.
  
  Stop SAS Secrets Manager immediately before you stop SAS Configuration Server.
- 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.
- On Linux systems that support systemd, use the `systemctl` command when running `sas-viya-vault-default`. The `systemctl` command maintains a record of service status that the `service` command and a direct call does not use.

**CAUTION**
On Red Hat Enterprise Linux 7.x (or an equivalent distribution) and SUSE Linux Enterprise Server 12.x, do not mix System V init and systemd commands. Mixing the System V init (service command) with the systemd (systemctl command) causes several issues. The systemctl command knows nothing about a SAS Viya service started with the service command. If you start `sas-viya-vault-default` on RHEL 7.x with the service command, and later attempt to shut down SAS Secrets Manager using the systemctl command, secrets manager stops responding and does not shut down.

Examples
- To check status of SAS Secrets Manager on Red Hat Enterprise Linux 7.x (or an equivalent distribution) and SUSE Linux Enterprise Server 12.x:
  ```bash
  sudo systemctl status sas-viya-vault-default
  ```
- To stop SAS Secrets Manager on Red Hat Enterprise Linux 6.x (or an equivalent distribution):
  ```bash
  sudo service sas-viya-vault-default stop
  ```
- To start SAS Secrets Manager on Red Hat Enterprise Linux 7.x (or an equivalent distribution) and SUSE Linux Enterprise Server 12.x:
To restart SAS Secrets Manager on Red Hat Enterprise Linux 6.x (or an equivalent distribution):

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

## Concepts

### What Is SAS Secrets Manager?

SAS Secrets 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 secrets manager and examples of how SAS Viya uses them:

- **On-demand generation of secrets**
  - Secrets manager generates TLS certificates for SAS Viya servers at start-up.

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

In multiple-machine, fault tolerant deployments, SAS Secrets Manager has a primary (leader) and one or more standbys (hot standbys). For information about SAS Secrets Manager topology, see “Fault Tolerance in SAS Viya (Linux)” in *SAS Viya Administration: General Servers and Services*.

### TTL Precedence Rules

The SAS Secrets Manager (Vault) time to live (TTL) properties have certain rules that you must follow. Failure to follow these rules can cause secrets manager not to start.
Log Files

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

Auditing

SAS Secrets Manager Auditing

Given the sensitive nature of the data being stored by the SAS Secrets Manager server, security-sensitive customers require security-related auditing.

Enable Auditing

The SAS Secrets Manager audit logging is enabled at all times and cannot be disabled automatically.

Security Considerations

The SAS Secrets Manager audit log entry contains the value of the secret in a hashed format, but can be deciphered with the appropriate access to the SAS Secrets Manager server. File permissions should be set for authorized users.

Audit Method

SAS Secrets Manager uses the file audit method to centralize logging. The file name is `sas-vault-audit_date-time.log`. This log is located at `/opt/sas/viya/config/var/log/vault/default/`.

Figure 4  Time to Live Properties Precedence Rules
Verify Auditing

To verify that you are receiving SAS Secrets Manager audit logs, perform an action to trigger the creation of an audit entry. The following actions generate audit logs:

- initial deployment
- sas-viya-all-services start-up

For additional details about HashiCorp Vault auditing, see Audit Devices.

SAS Infrastructure Data Server

Overview

SAS Infrastructure Data Server is based on PostgreSQL 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.

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.

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.
Pgpool-II

SAS provides Pgpool-II open-source software to enable you to manage PostgreSQL clusters for high availability (failover management). 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.

How To (Cluster)

Operate a Cluster (Linux)

SAS Viya provides three ways to start and stop your SAS Infrastructure Data Server cluster (cluster).

**IMPORTANT**  Consul and the `consul-template` services on the HA PostgreSQL cluster hosts must be running so that the start or stop script runs successfully.

**Note:** You must have root-level privileges to run the scripts.

**Script Invocation**

Used only when you need to start or stop the cluster.
The scripts can be run from any data or Pgpool node. The scripts are located in the `/opt/sas/viya/config/data/sasdatasvrc/clusterName/nodeName` directory.

To start the entire cluster, you must run the `startall` script as the sas user. Here is an example that shows that the cluster is started from `node0`.

```
sudo su - sas -c "'/opt/sas/viya/config/data/sasdatasvrc/postgres/node0/startall"
```

To stop the entire cluster, you must run the `shutdownall` script as the sas user.

```
sudo su - sas -c "'/opt/sas/viya/config/data/sasdatasvrc/postgres/node0/shutdownall"
```

### Init Services for sas-viya-all-services

**Note:** If your deployment spans multiple nodes, then start or stop your services by running the SAS Viya Multi-Machine Services Utilities playbooks.

When you need to start or stop all the services on a single host or shut down the host completely, use `sas-viya-all-services` script.

To start or stop or to check the status of all the SAS Viya services, run the following command:

```
sudo /etc/init.d/sas-viya-all-services start | stop | status
```

### Init Services for each PGPool or Data Node

Init services are for the individual nodes only. The SAS Infrastructure Data Server deployment creates several init scripts. They are categorized as follows:

- **Pgpool Node Level**
- **Data Node Level**
- **Consul Template on page 14**

The init scripts are deployed to the `/etc/init.d` directory on each server that is a deploy target for the SAS Infrastructure Data Server nodes. The init scripts support the following commands: Start, Stop, Restart, and Status.

You must have `sudo` or root privileges to run all the commands. However, the Status command can be invoked by any user.

The scripts are generally named as: `sas-viya-product-serviceName-nodeName`.

Here are few examples:

```
IMPORTANT The systemctl command does not provide a status of a service that it did not start. If the systemctl command has to respond to a status command of a service that was started by another process, then you must first run the start command of that service to bind it to the process and then run the status command.
```

**Note:** To see the full cluster status, use the script command.

```
- Script
  cd /etc/init.d
  sudo ./sas-viya-sasdatasvrc-postgres-node0 start | stop | restart | status

- On Red Hat Enterprise Linux 6.x (or an equivalent distribution)
  sudo service sas-viya-sasdatasvrc-postgres-node0 start | stop | restart | status

- On Red Hat Enterprise Linux 7.x (or an equivalent distribution) and SUSE Linux Enterprise Server 12.x
```
PGPool

Each Pgpool node has its own init script. Here is an example of a standard cluster service script:

`sas-viya-sasdatasvrc-postgres-pgpool0`

The PGPool init script supports the following operations:

- Start: starts the PGPool node
- Stop: stops the PGPool node
- Restart: stops and then starts the PGPool node
- Status: returns the status of the cluster by node if PGPool is running. Otherwise, it returns a message that PGPool is not running.

Here is example output:

```
PGPool is running with PID=11445
Checking Postgresql nodes status...

node_id | hostname | port | status | lb_weight | role   | select_cnt | load_balance_node
---------+----------+------+--------+-----------+---------+------------+-------------------
 0       | machine1 | 5452 | up     | 0.250000  | primary | 1          | true
 1       | machine2 | 5452 | up     | 0.250000  | standby | 0          | false
 2       | machine3 | 5452 | up     | 0.250000  | standby | 0          | false
 3       | machine4 | 5452 | up     | 0.250000  | standby | 0          | false
```

(4 rows)

For more information about the status of a cluster, see “Understand Status Output” on page 16.

Data Node

Every SAS Viya Infrastructure Server data node has its own node level init script. Here is an example of a standard service script for a SAS Viya Infrastructure Server cluster: `sas-viya-sasdatasvrc-postgres-node0`.

The data node init script supports the following operations:

- Start: starts the data node
- Stop: stops the data node
- Restart: stops the data node, but will not restart until the data node is rebooted
- Status: returns the status of the process that is associated with the data node

```
Note: There is no particular order to stop and start a node across the cluster nodes. However, to avoid unintended failover, it is recommended that you stop the primary node after you have stopped all other nodes in the cluster.
```

Consul Template

The Consul template init scripts support the following operations:

- Start: starts the Consul template daemon process
- Stop: stops the Consul template daemon process
- Restart: stops and then starts the Consul template daemon process
- Status: returns the status of the process that is associated with the Consul template daemon process

There are two Consul template init services that are created per cluster node:

- Operation

  It is used to manage cross-node communication within a cluster.
HBA

It is used to keep the hba.conf file synchronized with Consul.

The scripts are generally named as follows:

- `sas-viya-product-serviceName-nodeName-consul-template-operation_node`
- `sas-viya-product-serviceName-nodeName-consul-template-pg_hba`

Here are the default Consul template init scripts that are created for the SAS Viya Infrastructure Data Server cluster.

For each PGPool:

- `sas-viya-sasdatasvrc-postgres-pgpool0-consul-template-operation_node`
- `sas-viya-sasdatasvrc-postgres-pgpool0-consul-template-pool_hba`

For each data node:

- `sas-viya-sasdatasvrc-postgres-node0-consul-template-operation_node`
- `sas-viya-sasdatasvrc-postgres-node0-consul-template-pool_hba`

Note: The number at the end of the node name (such as node0 and pgpool0) specifies the node number. A second node would be node1 and pgpool1.

Usage Notes and Tips

- There is a script that 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.

**CAUTION**

On Red Hat Enterprise Linux 7.x (or an equivalent distribution) and SUSE Linux Enterprise Server 12.x, do not mix System V init and systemd commands. Mixing the System V init (service command) with the systemd (systemctl command) causes several problems. The systemctl command knows nothing about a SAS Viya service that was started with the service command. If you start `sas-viya-sasdatasvrc-postgres-pgpool0` on RHEL 7.x with the service command, and later attempt to shut down the data server cluster using the systemctl command, the data server stops responding and does not shut down.

Operate a Cluster (Windows)

Using the Services snap-in in the Microsoft Management Console, you can start, stop, and restart SAS Infrastructure Data Server.
Because there is a particular sequence in which the servers and services must be started and stopped, the individual services are not configured to run automatically when the SAS Viya machine is booted.

**IMPORTANT** SAS Configuration Service (Consul), SAS Infrastructure Data Server (PostgreSQL), SAS HTTP Proxy Server (Apache HTTP Server), and SAS Message Broker (RabbitMQ) are dependencies for the other SAS Viya services. If you are operating one or more services individually, always start each of these four services first and stop them last.

Note: There is one service, SAS Services Manager, that you can use to start and stop all SAS Viya servers and services. SAS Services Manager recognizes the dependencies between services and starts and stops services in the correct sequence. For more information, see “Start and Stop All Servers and Services” in SAS Viya Administration: General Servers and Services.

Understand Status Output
Note: The following examples are shown with init.d system commands on RHEL 6.x versions. Note that you should use the commands that are appropriate for your environment. See “Init Services for each PGPool or Data Node” on page 13 for more information.

Status All

This script provides the status of the entire environment including all PGPool nodes within the cluster. The script is called statusall and is located in the pgpool data directory of your cluster. Here is an example path: /opt/sas/viya/config/data/sasdatasvrc/postgres/pgpool0/statusall.

You must have sudo privileges to run the script.

```
sudo /opt/sas/viya/config/data/sasdatasvrc/sds-ci/pgpool0/statusall
```

The output of the script is written to the console.

Here is example output of the script with three PGPool nodes and four data nodes:

```
Consul Service Health Check

service:postgres-datanode0, passing
service:postgres-datanode1, passing
service:postgres-datanode2, passing
service:postgres-datanode3, passing
service:postgres-pgpool0, passing
service:postgres-pgpool1, passing
service:postgres-pgpool2, passing

/opt/sas/viya/home/bin/psql -c "show pool_nodes" with PGHOST=<IP address> PGPORT=5451
PGUSER=dbmsowner PGPASSWORD=vxxxxxxxxxxxxxxxxxxxxxx PGDATABASE=postgres

<table>
<thead>
<tr>
<th>node_id</th>
<th>hostname</th>
<th>port</th>
<th>status</th>
<th>lb_weight</th>
<th>role</th>
<th>select_cnt</th>
<th>load_balance_node</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>machine1.com</td>
<td>5452</td>
<td>up</td>
<td>0.250000</td>
<td>primary</td>
<td>1</td>
<td>true</td>
</tr>
<tr>
<td>1</td>
<td>machine2.com</td>
<td>5452</td>
<td>up</td>
<td>0.250000</td>
<td>standby</td>
<td>0</td>
<td>false</td>
</tr>
<tr>
<td>2</td>
<td>machine3.com</td>
<td>5452</td>
<td>up</td>
<td>0.250000</td>
<td>standby</td>
<td>0</td>
<td>false</td>
</tr>
<tr>
<td>3</td>
<td>machine4.com</td>
<td>5452</td>
<td>up</td>
<td>0.250000</td>
<td>standby</td>
<td>0</td>
<td>false</td>
</tr>
</tbody>
</table>

(4 rows)

/opt/sas/viya/home/bin/pcp_watchdog_info -h <IP Address> -p 5450 -U sas --no-password 2>&1

2 YES machine3.com:5451 Linux machine3.com machine3.com
machine4.com:5451 Linux machine4.com machine4.com 5451 9000 4 MASTER
machine2.com:5451 Linux machine2.com machine2.com 5451 9000 4 STANDBY

Testing the same 'show pool_nodes' command on machine4.com...
...test successful
Testing the same 'pcp_watchdog_info' command on machine4.com...
...test successful
Testing the same 'show pool_nodes' command on machine3.com...
...test successful
Testing the same 'pcp_watchdog_info' command on machine3.com...
...test successful
```

Init Status

When the cluster is stopped, the status is shown as follows:
The status of the running default single node PostgreSQL cluster is as follows:

```
sudo service sas-viya-sasdatasetsvrc-postgres-pgpool0 status
PGPool is not running
```

The status of a multi-node cluster contains information about each node in the cluster. It identifies the primary node. A status of up means that the node is running and available. A status of down means that the node is unhealthy and needs to be recovered. Here is an example:

```
sudo service sas-viya-sasdatasetsvrc-postgres2-pgpool0 status
PGPool is running with PID=11445
Checking Postgresql nodes status...
+ node_id | hostname | port | status | lb_weight | role    | select_cnt | load_balance_node |
+---------+----------+-------+--------+-----------+---------+------------+-------------------+--
| 0       | machine1 | 5462  | up     | 1.000000  | primary | 1          | true              |
(1 row)
```

In the following example, the cluster contains four data nodes. Node0 is the primary node. Node2 is unhealthy because it has a status of down. Node2 must be recovered.

```
sudo service sas-viya-sasdatasetsvrc-postgres2-pgpool0 status
PGPool is running with PID=11445
Checking Postgresql nodes status...
+ node_id | hostname | port | status | lb_weight | role    | select_cnt | load_balance_node |
+---------+----------+-------+--------+-----------+---------+------------+-------------------+--
| 0       | machine1 | 5462  | up     | 0.250000  | primary | 1          | true              |
| 1       | machine2 | 5452  | up     | 0.250000  | standby | 0          | false             |
| 2       | machine3 | 5452  | down   | 0.250000  | standby | 0          | false             |
| 3       | machine4 | 5452  | up     | 0.250000  | standby | 0          | false             |
(4 rows)
```

When the primary data node stops, failover occurs. A stopped state can be determined by checking the status output. Notice that node0 has a status of down and has the role of standby. Node1 is now the primary data node.

```
sudo service sas-viya-sasdatasetsvrc-postgres2-pgpool0 status
```
Checking status of sas-viya-sasdatasvrc-postgres2-pgpool0...

PGPool is running with PID=11445

Checking PostgreSQL nodes status...

<table>
<thead>
<tr>
<th>node_id</th>
<th>hostname</th>
<th>port</th>
<th>status</th>
<th>lb_weight</th>
<th>role</th>
<th>select_cnt</th>
<th>load_balance_node</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>machine1</td>
<td>5452</td>
<td>down</td>
<td>0.250000</td>
<td>standby</td>
<td>1</td>
<td>false</td>
</tr>
<tr>
<td>1</td>
<td>machine2</td>
<td>5452</td>
<td>up</td>
<td>0.250000</td>
<td>primary</td>
<td>0</td>
<td>true</td>
</tr>
<tr>
<td>2</td>
<td>machine3</td>
<td>5452</td>
<td>up</td>
<td>0.250000</td>
<td>standby</td>
<td>0</td>
<td>false</td>
</tr>
<tr>
<td>3</td>
<td>machine4</td>
<td>5452</td>
<td>up</td>
<td>0.250000</td>
<td>standby</td>
<td>0</td>
<td>false</td>
</tr>
</tbody>
</table>

(4 rows)

Add a Cluster

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

You can have zero or more standby nodes and one primary node.

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

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

Note: The service name can contain letters, numbers, and underscores only.

3 Copy and paste an existing cluster definition.

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

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

19
Configure the new cluster definition for the pgpool server (pgpoolc) and the data server nodes (sasdatasvrc). See “Cluster Definition for PGPool Server and Data Server Nodes” on page 20 for more information.

If the cluster machines that you are adding are already a part of your SAS Viya deployment, and they are already in [pgpoolc] and [sasdatasvrc] host groups, then go to Step 6.

Otherwise, add the machines to your Ansible inventory.ini file at the top of the file in the target list, and in the [pgpoolc] and [sasdatasvrc] host groups, respectively.

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 “Deploy the Software” in SAS Viya for Linux: Deployment Guide.

Cluster Definition for PGPool Server and Data Server Nodes

Note: For detailed information about the variable values, see “Values in vars.yml” in SAS Viya for Linux: Deployment Guide.

Here are the PGPool server definition parameters:

- **HA_PGPOOL_VIRTUAL_IP**
  
  If it is a clustered PGPool, then a virtual IP is required. The virtual IP address is assigned to all PGPools within a cluster. All PGPool instances within a cluster must have the same virtual IP.
  
  The virtual IP for a PGPool must be accessible to all cluster members and all PostgreSQL clients. It is recommended that the virtual IP be assigned to the PGPool instance for the duration of the deployment.

- **HA_PGPOOL_WATCHDOG_PORT**
  
  The port that the PGPool watchdog process listens on. All PGPool instances within a cluster must have the same watchdog port.

- **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.

- **POOL_NUMBER**
The sequential PGPool node identifier starting at 0 per service name. This value cannot be changed.

- **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.

Here are the data server node definition parameters:

- **NODE_NUMBER**
  Specifies the sequential node identifier. The primary node is 0. Standby nodes start at 1 and are increased sequentially.

- **PG_PORT**
  Specifies the PostgreSQL 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.

### Delete a Node or a Cluster

**CAUTION**

**Do not delete a primary node unless you plan to delete the entire cluster.** Deleting a primary node would increase chances of introducing data corruption. To delete the primary node, fail over 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 makes 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. A cluster must have at least half of the configured PGPool nodes running. Otherwise, it is not possible to have a PGPool failover. This condition makes the cluster unusable.

A delete node script is provided to facilitate the removal of any cluster node from a cluster. The script can also be used to remove an entire cluster or a node.

1. As root or with an account that has root-level privileges, sign in to the machine where the node that you want to remove resides. Failure to run as a root or sudoer user results in the following error:

   ```
   ERROR: The current user is 'sas' and the script must be executed under 'root' or a 'sudo privilege' user.
   ```

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 serviceName`
- `n nodeName`
- `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`

Examples:

**Delete PGPool node**

```
/opt/sas/viya/home/libexec/sasdatasvrc/script/sds_delete_node.sh -s sds-petrichor -n pgpool0 -d y
- c /opt/sas/viya/config/etc/sasdatasvrc/sds-petrichor/pgpool0/sds_env_var.sh
```

**Delete PostgreSQL primary node and keep the data directory**

```
/opt/sas/viya/home/libexec/sasdatasvrc/script/sds_delete_node.sh -s sds-ci -n node0 -d n
- c /opt/sas/viya/config/etc/sasdatasvrc/sds-ci/node0/sds_env_var.sh
```

**Delete PostgreSQL standby node and data directory**

```
/opt/sas/viya/home/libexec/sasdatasvrc/script/sds_delete_node.sh -s sds-ci -n node1 -d y
- c /opt/sas/viya/config/etc/sasdatasvrc/sds-ci/node1/sds_env_var.sh
```

4 Each time the script runs, it generates a new log file in /tmp/sds_uninstall_log. Here is an example: `sds_delete_sds-va.node1_20171019174625.log`

5 After the script runs, be sure to delete the node or the cluster definition in the INVOCATION_VARIABLES section of the vars.yml file. This removes the node from the PGPool cluster definition.

**Failover**

Failover happens when a PGPool server loses connection to the primary data node. PGPool automatically iterates through its node list, always starting at node0. It checks the next available node and attempts to promote the node to primary. The following steps show how the failover takes place:

1 As the SAS install user (sas) or the user who has root-level privileges, sign in to the PGPool server machine.

2 Obtain the status of the cluster from the PGPool host and identify the primary node. Note the name of the host computer that the primary node runs on. Run the following command that is appropriate for your operating system:

   - Red Hat Enterprise Linux 7.x (or an equivalent distribution) and SUSE Linux Enterprise Server 12.x:
     ```
     cd /etc/init.d
     sudo ./sas-viya-sasdatasvrc-serviceName-pgpool0 status
     ```
   
   - Red Hat Enterprise Linux 6.x (or an equivalent distribution):
     ```
     sudo service sas-viya-sasdatasvrc-serviceName-pgpool0 status
     ```

Verify that all the nodes have a status of up.
Here is an example of running the command on Red Hat Enterprise Linux 6.x (or an equivalent distribution):

```
sudo service sas-viya-sasdatasvrc-postgres2-pgpool0 status
```

Here is typical output:

```
Checking status of sas-viya-sasdatasvrc-postgres2-pgpool0...
PGPool is running with PID=11445
Checking Postgresql nodes status...

+---------+----------+------+--------+-----------+---------+------------+-------------------+
<table>
<thead>
<tr>
<th>node_id</th>
<th>hostname</th>
<th>port</th>
<th>status</th>
<th>lb_weight</th>
<th>role</th>
<th>select_cnt</th>
<th>load_balance_node</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>machine1</td>
<td>5452</td>
<td>up</td>
<td>0.250000</td>
<td>primary</td>
<td>1</td>
<td>true</td>
</tr>
<tr>
<td>1</td>
<td>machine2</td>
<td>5452</td>
<td>up</td>
<td>0.250000</td>
<td>standby</td>
<td>0</td>
<td>false</td>
</tr>
<tr>
<td>2</td>
<td>machine3</td>
<td>5452</td>
<td>up</td>
<td>0.250000</td>
<td>standby</td>
<td>0</td>
<td>false</td>
</tr>
<tr>
<td>3</td>
<td>machine4</td>
<td>5452</td>
<td>up</td>
<td>0.250000</td>
<td>standby</td>
<td>0</td>
<td>false</td>
</tr>
</tbody>
</table>
+---------+----------+------+--------+-----------+---------+------------+-------------------+
(4 rows)
```

3. As the SAS install user (sas) or the user who has root-level privileges, sign in to the primary node machine.

Stop the primary node of the cluster. Run the following command:

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

Here is the output:

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

4. Sign in to the PGPool server. Check the status of the cluster and verify that the previous primary node has a status of down and that another node has been selected as a primary node.

```
sudo service sas-viya-sasdatasvrc-postgres2-pgpool0 status
```

Here is the output:

```
Checking status of sas-viya-sasdatasvrc-postgres2-pgpool0...
PGPool is running with PID=11445
Checking Postgresql nodes status...

+---------+----------+------+--------+-----------+---------+------------+-------------------+
<table>
<thead>
<tr>
<th>node_id</th>
<th>hostname</th>
<th>port</th>
<th>status</th>
<th>lb_weight</th>
<th>role</th>
<th>select_cnt</th>
<th>load_balance_node</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>machine1</td>
<td>5452</td>
<td>down</td>
<td>0.250000</td>
<td>standby</td>
<td>1</td>
<td>false</td>
</tr>
<tr>
<td>1</td>
<td>machine2</td>
<td>5452</td>
<td>up</td>
<td>0.250000</td>
<td>primary</td>
<td>0</td>
<td>true</td>
</tr>
<tr>
<td>2</td>
<td>machine3</td>
<td>5452</td>
<td>up</td>
<td>0.250000</td>
<td>standby</td>
<td>0</td>
<td>false</td>
</tr>
<tr>
<td>3</td>
<td>machine4</td>
<td>5452</td>
<td>up</td>
<td>0.250000</td>
<td>standby</td>
<td>0</td>
<td>false</td>
</tr>
</tbody>
</table>
+---------+----------+------+--------+-----------+---------+------------+-------------------+
(4 rows)
```

Note: The remaining running nodes of the cluster initially show a status of down while replication is re-established to the new primary node. Keep reviewing the cluster status until all running nodes have a status of up. The former primary node is marked as a standby node and has a status of down because it is currently unavailable.
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 data node has been stopped or has been taken offline, the PGPool server marks this node as down.

To manually add this node back to an active cluster, perform the following steps:

1. Make sure that the following servers are running and accessible:
   - SAS Configuration Server (Consul)
   - PGPool server

2. As the SAS install user (sas) or the user who has root-level privileges, sign in to the host machine of the node and run the following command that is appropriate for your operating system:
   - Red Hat Enterprise Linux 7.x (or an equivalent distribution) and SUSE Linux Enterprise Server 12.x:
     ```
     cd /etc/init.d
     sudo ./sas-viya-sasdatasvrc-serviceName-nodeName status
     ```
   - Red Hat Enterprise Linux 6.x (or an equivalent distribution):
     ```
     sudo service sas-viya-sasdatasvrc-serviceName-nodeName status
     ```

   On the PGPool node, verify that the unhealthy data node has a status of down and a role of standby.

   Here is an example on Red Hat Enterprise Linux 6.x (or an equivalent distribution), where the data server service is named postgres2 and the name of the node is node0:
   ```
   sudo service sas-viya-sasdatasvrc-postgres2-pgpool0 status
   ```

   Here is typical output:

   Check status of sas-viya-sasdatasvrc-postgres2-pgpool0...
   
   PGPool is running with PID=11445
   Checking PostgreSQL nodes status...
   node_id hostname port status lb_weight role select_cnt load_balance_node
   +-------------------+----------+------+--------+-----------+---------+------------+-------------------
   0 machine1 5452 down 0.250000 standby 1 false
   1 machine2 5452 up 0.250000 primary 0 true
   2 machine3 5452 up 0.250000 standby 0 false
   3 machine4 5452 up 0.250000 standby 0 false
   (4 rows)

3. Recover the node as a standby server by running the following command that is appropriate for your operating system:
   - Red Hat Enterprise Linux 7.x (or an equivalent distribution) and SUSE Linux Enterprise Server 12.x:
     ```
     sudo systemctl start sas-viya-sasdatasvrc-serviceName-nodeName
     ```
   - Red Hat Enterprise Linux 6.x (or an equivalent distribution):
     ```
     sudo service sas-viya-sasdatasvrc-serviceName-nodeName 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 on Red Hat Enterprise Linux 6.x (or an equivalent distribution):

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

Here is typical output:

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

4. Make sure that the node has been successfully added to the cluster by running the following command on the PGPool node that is appropriate for your operating system:

- Red Hat Enterprise Linux 7.x (or an equivalent distribution) and SUSE Linux Enterprise Server 12.x:
  ```bash
cd /etc/init.d
sudo ./sas-viya-sasdatasvrc-serviceName-pgpool0 status
```

- Red Hat Enterprise Linux 6.x (or an equivalent distribution):
  ```bash
  sudo service sas-viya-sasdatasvrc-serviceName-pgpool0 status
  ```

Here is an example on Red Hat Enterprise Linux 6.x (or an equivalent distribution):

```bash
sudo service sas-viya-sasdatasvrc-postgres2-pgpool0 status
```

Here is typical output:

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

<table>
<thead>
<tr>
<th>Checking status of sas-viya-sasdatasvrc-postgres2...</th>
</tr>
</thead>
<tbody>
<tr>
<td>PGPool is running with PID=11445</td>
</tr>
<tr>
<td>Checking Postgresql nodes status...</td>
</tr>
<tr>
<td>node_id</td>
</tr>
<tr>
<td>---------</td>
</tr>
<tr>
<td>0</td>
</tr>
<tr>
<td>1</td>
</tr>
<tr>
<td>2</td>
</tr>
<tr>
<td>3</td>
</tr>
<tr>
<td>(4 rows)</td>
</tr>
</tbody>
</table>

**Note:** If starting (recovering) a node fails, refer to the **Troubleshooting section on page 46**.

---

**Add a Data Node or PGPool Node**

You can add a data node or a PGPool node to an existing deployment.

**Note:** You cannot add new data nodes or PGPool nodes to your deployment until after you upgrade PostgreSQL.
In the following example, one data node and two PGPool nodes are added. Adding PGPool nodes makes the cluster an HA PGPool cluster.

- If you are adding a node to a new host computer, make sure that the host is listed as a deploy target at the top of the inventory.ini file.

  ```
  deployTarget1 ansible_ssh_host=machine_address1 ansible_ssh_user=userid
  ansible_ssh_private_key_file=keyfile
  deployTarget2 ansible_ssh_host=machine_address2 ansible_ssh_user=userid
  ansible_ssh_private_key_file=keyfile
  deployTarget3 ansible_ssh_host=machine_address3 ansible_ssh_user=userid
  ansible_ssh_private_key_file=keyfile
  ```

- Add the new node host as an entry in the sasdatasvrc host group for data nodes or the pgpoolc host group for PGPool nodes if it is not already listed.

  ```
  [sasdatasvrc]
  deployTarget1
  deployTarget2

  [pgpoolc]
  deployTarget1
  deployTarget2
  deployTarget3
  ```

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.

   ```yaml
   INVOCATION_VARIABLES:
   deployTarget1:
   pgpoolc:
     - HA_PGPOOL_VIRTUAL_IP: '10.104.21.10'
     - HA_PGPOOL_WATCHDOG_PORT: '9000'
     - PCP_PORT: '5430'
     - PGPOOL_PORT: '5431'
     - POOL_NUMBER: '0'
     - SANMOUNT: '{{ SAS_CONFIG_ROOT }}/data/sasdatasvrc'
     - SERVICE_NAME: postgres
   sasdatasvrc:
     - NODE_NUMBER: '0'
     - PG_PORT: '5432'
     - SANMOUNT: '{{ SAS_CONFIG_ROOT }}/data/sasdatasvrc'
     - SERVICE_NAME: postgres
   ```

3. Copy an existing data node definition and PGPool node definition. Place it under the deploy target where the node is configured.

   Here is an example:

   ```yaml
   INVOCATION_VARIABLES:
   deployTarget1:
   pgpoolc:
     - HA_PGPOOL_VIRTUAL_IP: '10.104.21.10'
     - HA_PGPOOL_WATCHDOG_PORT: '9000'
     - PCP_PORT: '5430'
     - PGPOOL_PORT: '5431'
     - POOL_NUMBER: '0'
   ```
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.

- **PG_PORT**
  Specifies the PostgreSQL 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.

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

Here is an example:
Determine the Primary Data Node

There are three ways to determine the data node that is currently the primary node.

- **Using** Run the service command for the PGPool
  
  `sudo service sas-viya-sasdatasvrc-postgres-pgpool0 status`

  Here is an example of the output:

    Checking status of sas-viya-sasdatasvrc-postgres2-pgpool0...
    PGPoll is running with PID=11445
    Checking Postgresql nodes status...
    node_id | hostname | port | status | lb_weight | role   | select_cnt | load_balance_node |
    --------+----------+------|--------+-----------+---------+------------+-------------------+--
    0       | machine1 | 5452 | up     | 0.250000  | standby | 0          | false             |
    1       | machine2 | 5452 | up     | 0.250000  | primary | 2348449    | true              |
    2       | machine3 | 5452 | up     | 0.250000  | standby | 0          | false             |
    (3 rows)

- **Run the statusall script on any node.**
  
  `sudo /opt/sas/viya/config/data/sasdatasvrc/postgres/pgpool0/statusall`

  For an example of the output, see “Status All” on page 17.

- **Query the primary node from Consul.**

  Log on as a sas install user (an example is sas). Set up the environment variables for Consul.

  ```bash
  source /opt/sas/viya/config/consul.conf
  export CONSUL_HTTP_TOKEN=$(cat /opt/sas/viya/config/etc/SASSecurityCertificateFramework/
  tokens/consul/default/client.token)
  /opt/sas/viya/home/bin/sas-bootstrap-config kv read --recurse config | grep primary_datanode
  ```

  Here is an example of the output:

    config/postgres/admin/primary_datanode_host=machine2
    config/postgres/admin/primary_datanode_number=1
    config/postgres/admin/primary_datanode_port=5432

Determine the Master PGPool Node

To determine the master PGPool node, execute the statusall script within the pool data directory.

`sudo /opt/sas/viya/config/data/sasdatasvrc/postgres/pgpool0/statusall`

Here is an excerpt from example output:

```plaintext
....
machine1:5431 Linux machine1 machine1 5431 9000 7 STANDBY
machine2:5431 Linux machine2 machine2 5431 9000 4 MASTER
....
```
Determine the Database Size

There are two ways to find out the size of the database.

Connect to psql. To list the databases, run the following command:

```
\l+
```

This command lists all the details of the existing databases. Here is example output:

| Name           | Owner     | Encoding | Collate | Ctype | Access privileges | Size    | Tablespace |
|----------------|-----------+----------+---------+-------+-------------------+---------+------------+----------------+---------------------------------------------------|
| SharedServices | dbmsowner | UTF8     | C       | C     |                   | 7577 kB | pg_default |
| postgres       | sas       | UTF8     | C       | C     |                   | 7701 kB | pg_default |
| template0      | sas       | UTF8     | C       | C     | =c/sas            +| 7561 kB | pg_default |
| template1      | sas       | UTF8     | C       | C     | =c/sas            +|  7717 kB| pg_default |

You can also run an SQL statement to list the size of the database:

```sql
select t1.datname AS db_name, pg_size_pretty(pg_database_size(t1.datname))
as db_size from pg_database t1 order by pg_database_size(t1.datname) desc;
```

This command lists only the name of the database and its size. Here is example output:

<table>
<thead>
<tr>
<th>db_name</th>
<th>db_size</th>
</tr>
</thead>
<tbody>
<tr>
<td>template1</td>
<td>7717 kB</td>
</tr>
<tr>
<td>postgres</td>
<td>7701 kB</td>
</tr>
<tr>
<td>SharedServices</td>
<td>7577 kB</td>
</tr>
<tr>
<td>template0</td>
<td>7561 kB</td>
</tr>
</tbody>
</table>

Manage Connections to PGPool and PostgreSQL

This information is useful to account for an increasing number of microservices and their individual connection pools. Additional Linux operating system-level settings might be required in order to support the increased number of connections. To optimize your PostgreSQL resources, you should also scale the server’s working memory settings. For more information, see Tuning the PostgreSQL Data Server in SAS Web Applications: Tuning for Performance and Scalability.

PGPool and PostgreSQL have a total of 256 reserved connections:

<table>
<thead>
<tr>
<th>Number of Connections</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>128</td>
<td>Minimum number of replication connections for pgpool back ends.</td>
</tr>
<tr>
<td>3</td>
<td>Minimum number of replication connections for pgpool pcp, health, and worker.</td>
</tr>
</tbody>
</table>

Table 1  Reserved Connections
<table>
<thead>
<tr>
<th>Number of Connections</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>125</td>
<td>Minimum number of replication connections for tenants, super user, etl, pgadmin, and miscellaneous.</td>
</tr>
</tbody>
</table>

The default connection values for PGPool and PostgreSQL in SAS Viyva 3.5 are as follows:

- **PGPool**
  
  \[
  \text{num\_init\_children} = 1024
  \]

- **PostgreSQL**
  
  \[
  \text{max\_connection} = 1280
  \]

  \[
  \text{max\_prepared\_transactions} = 1280
  \]

To increase the number of connections that are available to clients, modify the values for num\_init\_children, max\_connections, and max\_prepared\_transactions in SAS Environment Manager:

1. Log on to SAS Environment Manager as SAS administrator.
2. In the navigation bar, click Configuration. See “Navigation” in SAS Viyva Administration: Configuration Properties for more information.
4. Find the sas.dataserver.pool:common configuration instance and edit the num\_init\_children property field.
5. Find the sas.dataserver.conf:common configuration instance and edit the max\_connections and max\_prepared\_transactions property fields.

If you increase the value of num\_init\_children, you must also adjust the values of max\_connections and max\_prepared\_transactions.

Here is an example:

1. Adjust the max\_connections and max\_prepared\_transactions to the value of num\_init\_children plus the total reserved connections on page 29.

<table>
<thead>
<tr>
<th>Table 2</th>
<th>Adjusted Values</th>
</tr>
</thead>
<tbody>
<tr>
<td>Variable</td>
<td>Value</td>
</tr>
<tr>
<td>num_init_children</td>
<td>2048</td>
</tr>
<tr>
<td>max_connections</td>
<td>2304 (2048 num_init_children + 256 reserved connections)</td>
</tr>
<tr>
<td>max_prepared_transactions</td>
<td>2304 (2048 num_init_children + 256 reserved connections)</td>
</tr>
</tbody>
</table>

If you attempt to complete the deployment after increasing these connection settings without adjusting the ulimit and semaphore system setting values, you might experience a deployment error with the following log entries:
Deployment log
recovery node x ...BackendError
Configuration of pool failed

PostgreSQL log
Different types of PostgreSQL errors:
- Shared memory error
  create shared memory segment: Cannot allocate memory
- Memory error for semaphores
  UTCFATAL: could not create semaphores: No space left on device

To resolve these errors, adjust the ulimit and semaphore operating system settings.

2 Update the sas user limits to the recommended settings in the `/etc/security/limits.conf` file as follows:

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

A brief explanation of the arguments to these settings follows:

`nofile` refers to the number of open files

`stack` specifies the stack size in KB

`nproc` specifies the maximum number of user processes

For details about ulimit, see the documentation for your operating system.

3 Update the semaphore operating-system settings.

Note: A change to semaphore settings in Linux requires a reboot of the server.

In the file `/etc/sysctl.conf`, locate the `kernel.sem` line.

```bash
kernel.sem=SEMMSL,SEMMNS,SEMOPM,SEMMNI
net.core.somaxconn=2048
```

The default values for semaphores are as follows:

- SEMMSL : 512
- SEMMNS : 32000
- SEMOPM : 100
- SEMMNI : 1024

Here are typical values for semaphores. For more information, see PostgreSQL documentation.
Table 3 Semaphore Values

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
<th>Typical Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>SEMMSL</td>
<td>Maximum number of semaphores per set</td>
<td>17 (minimum)</td>
</tr>
<tr>
<td>SEMMNS</td>
<td>Maximum number of semaphores system-wide</td>
<td></td>
</tr>
<tr>
<td></td>
<td>((max_connections + (\text{autovacuum_max_workers} + \text{max_worker_processes} + 5)) / 16) * 17 plus room for other applications</td>
<td></td>
</tr>
<tr>
<td>SEMOPM</td>
<td>Maximum number of semaphore operations that can be performed per semaphore call</td>
<td>100 (minimum)</td>
</tr>
<tr>
<td>SEMMNI</td>
<td>Maximum number of semaphore identifiers (sets)</td>
<td>((\text{minimum}) ((\text{max_connections} + (\text{autovacuum_max_workers} + \text{max_worker_processes} + 5)) / 16))</td>
</tr>
</tbody>
</table>

1By default, `autovacuum_max_workers` is set to 3.

To verify the default value, run the command:

```
postgres=# show autovacuum_max_workers;
autovacuum_max_workers
```

Here is the output:

```
------------------------
3
(1 row)
```

How To (General)

Get Current Passwords (Linux)

1. As the SAS install user (sas) or the user who has root-level 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:
     ```
source /opt/sas/viya/config/consul.conf

export CONSUL_HTTP_TOKEN=$(cat /opt/sas/viya/config/etc/SASSecurityCertificateFramework/tokens/consul/default/client.token)
```
   - With root-level privileges (but not as the install user), install accounts:
     ```
source /opt/sas/viya/config/consul.conf

export CONSUL_HTTP_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

Note: By default SASHOME is /opt/sas/viya/home. If your deployment is using a different home directory, the paths should be modified to match your deployment.

Change User Passwords (Linux)

The script, sds_change_user_pw.sh, changes SAS Infrastructure Data Server passwords and synchronizes them with SAS Configuration Server (Consul) and configuration files. The script must be run on the host machine that has Pgpool server.

**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 (Linux)”.

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 following command:

  sudo service sas-viya-sasdatasvrc-serviceName-pgpool0 status

For the output, see “Understand Status Output” on page 16.

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 on Red Hat Enterprise Linux 6.x (or an equivalent distribution), where the data server service is named postgres2:

  sudo service sas-viya-sasdatasvrc-postgres2-pgpool0 status

Here is typical output:
Checking status of sas-viya-sasdatasvrc-postgres2-pgpool0...

PGPool is running with PID=11445
Checking Postgresql nodes status...

<table>
<thead>
<tr>
<th>node_id</th>
<th>hostname</th>
<th>port</th>
<th>status</th>
<th>lb_weight</th>
<th>role</th>
<th>select_cnt</th>
<th>load_balance_node</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>machine1</td>
<td>5452</td>
<td>up</td>
<td>0.250000</td>
<td>primary</td>
<td>1</td>
<td>false</td>
</tr>
<tr>
<td>1</td>
<td>machine2</td>
<td>5452</td>
<td>up</td>
<td>0.250000</td>
<td>standby</td>
<td>0</td>
<td>true</td>
</tr>
<tr>
<td>2</td>
<td>machine3</td>
<td>5452</td>
<td>up</td>
<td>0.250000</td>
<td>standby</td>
<td>0</td>
<td>false</td>
</tr>
<tr>
<td>3</td>
<td>machine4</td>
<td>5452</td>
<td>up</td>
<td>0.250000</td>
<td>standby</td>
<td>0</td>
<td>false</td>
</tr>
</tbody>
</table>

(4 rows)

3 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.

4 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 47.

5 Using the location of sds_env_var.sh noted in Step 3, run the script using the following command:

   ```shell
   sudo su - 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"
   ```

6 Enter the information that you collected in Step 4 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.

### Clean Up after a Hardware Failure (Linux)

If the machine stops unexpectedly that has high availability (HA) SAS Infrastructure Data Server cluster running, you might need to perform the 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 root-level privileges, sign in to the pgpool machine.

2 Stop the HA data server cluster. For more information, see Operate a Cluster on page 12.

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.

   - /tmp/.s.PGSQL.5430
   - /tmp/.s.PGSQL.5431
Start the HA data server cluster. For more information, see Operate a Cluster on page 12.

Remove a Persistent Lock on a Database Table (Linux)

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 root-level 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 (Linux”).

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 shows 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.
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.

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:

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

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.

Exit pgAdmin.

---

**Upgrading PostgreSQL in SAS Viya**

**Upgrade Overview**

You use the SAS Viya PostgreSQL upgrade utility to upgrade PostgreSQL databases. Upgrades are performed by using the PostgreSQL upgrade play for Ansible. The upgrade play launches the upgrade utility to run an upgrade on all machines that are running PostgreSQL databases, which cause all of them to upgrade at the same time. Each instance of the upgrade utility uses Consul to update its progress so that the instances do not interfere with each other.

Here is a summary of the upgrade process:

1. **Information about the PostgreSQL database is collected.**
   
   The utility contacts Consul and queries the databases to gather information for the upgrade. For example, the information might include the number of primary and standby nodes, the machine that hosts each PostgreSQL node, the version of each database, and the versions of PostgreSQL binaries that are installed on each machine.

2. **The primary PostgreSQL data is migrated.**
   
   Each PostgreSQL cluster in SAS Viya contains one primary server and $n$ standby servers. The standby servers enter a wait state while the primary server finishes the upgrade that is performed by the `pg_upgrade` CLI tool (default), which is provided by PostgreSQL.
   
   This step creates a second database in the upgrade version on the machine. The `pg_upgrade` tool streams the data from the current database into the upgrade database. At the end of this step, there are two data directories: the old version of PostgreSQL and the new database. Both contain the same data.

3. **The PostgreSQL configuration is migrated.**
The configuration information for PostgreSQL is defined by the \texttt{postgresql.conf} and \texttt{pg_hba.conf} files in the server's data directory. The configuration can change between major releases of PostgreSQL.

The \texttt{sas-pgupgrade-cli} tool accepts a configuration file, \texttt{postgresql_config_changes.json}, which defines all the changes that must be made to the following configuration files: \texttt{postgresql.conf} and \texttt{pg_hba.conf}. A version of the \texttt{postgresql_config_changes.json} file that contains the changes that are approved by SAS is shipped with the upgrade utility.

4 PostgreSQL performance is tuned.

To boost PostgreSQL performance after an upgrade, the upgrade utility provides the \texttt{vacuumdb} command that is run against the database.

5 The standby PostgreSQL data is migrated.

After the primary server finishes the upgrade, the standby servers start their upgrade using the \texttt{pg_basebackup} utility, which is provided by PostgreSQL.

6 The standby PostgreSQL configuration is migrated.

Using the same steps as for the primary server, the configuration information is upgraded for the standby server.

7 Pgpool is reconfigured.

Pgpool might require some configuration changes in order to communicate with the new version of PostgreSQL. For example, the changes for PostgreSQL servers are defined by the \texttt{postgresql_config_changes.json} file.

Upgrade PostgreSQL

You use the SAS Viya PostgreSQL upgrade utility, \texttt{sas-pgupgrade-cli}, a command-line tool, to upgrade PostgreSQL for a SAS Viya deployment.

This utility automatically scans a SAS Viya deployment for all running PostgreSQL databases and migrates them to the latest version of PostgreSQL that was installed by SAS on the physical machines. If there are two versions of PostgreSQL in your deployment, you must select the version to be upgraded. The tool performs additional tasks such as migrating PostgreSQL configuration data in Consul to the new version.

Pre-upgrade Steps

1 Shut down your SAS Viya deployment before you run the upgrade utility. By shutting down, you prevent errors that could occur when SAS microservices are not able to connect to the PostgreSQL database during the upgrade.

2 However, in order for the upgrade utility to work, the SAS Configuration Server (Consul), SAS Secret Manager (Vault), and Consul template services for PostgreSQL must be running. Therefore, after you shut down your SAS Viya deployment, start these services only. Use the commands that are appropriate for your environment. See \texttt{Start and Stop Servers and Services} for more information.

The Consul service should be running so that the utility gathers metadata about one or more PostgreSQL servers. If the Consul service is not running, log on to the machine (one or more) that hosts your Consul server (one or more) and the machine (one or more) that hosts either PostgreSQL or PGPool nodes.

Check the status of the Consul service. Start the service if it is not running. See information about how to run the \texttt{sas-viya-consul-default} script in \texttt{SAS Configuration Server: Operate (Linux)} on page 4.
On Windows, make sure that the SAS Consul Service service is running. See “Operate (Windows)” on page 5 for more information.

3 Make sure that SAS Secrets Manager (Vault) is running. Log on to the machine (one or more) that hosts the Vault server (one or more). Check the status of the service. Start the service if it is not running. See information about how to run the sas-viya-vault-default script in SAS Secrets Manager: Operate on page 8.

Note: Vault is not supported on Windows.

4 Make sure that the Consul template services are running. There are two consul template services for every PostgreSQL and PGPool node in your SAS Viya deployment. The services are as follows:

- sas-viya-product-servicename-nodename-consul-template-operation_node
- sas-viya-product-servicename-nodename-consul-template-pg_hba

To operate these services, see “Consul Template” on page 14.

Note: Make sure that the Consul and Vault services are started before attempting to start Consul template services. Consul template services are not supported on Windows.

5 The upgrade utility is located in the following directory on any machine that contains a PGPool or a PostgreSQL server.

Table 4 Upgrade Utility Paths

<table>
<thead>
<tr>
<th>Platform</th>
<th>Path</th>
</tr>
</thead>
<tbody>
<tr>
<td>Linux</td>
<td>/opt/sas/viya/home/libexec/sasdatasvrc/utilities/sas-pgupgrade-cli</td>
</tr>
<tr>
<td>Windows</td>
<td>C:\Program Files\SAS\Viya\libexec\sasdatasvrc\utilities\sas-pgupgrade-cli.exe</td>
</tr>
</tbody>
</table>

6 To see the options that are available for the utility tool, run the following command:

./sas-pgupgrade-cli --help

7 There are two commands provided by the tool: analyze and upgrade. For example, to see the help on upgrade command, run the following command:

./sas-pgupgrade-cli upgrade --help

Log Files
Log files for the upgrade utility are located at the following paths:

- Linux:
  
  `%SASHOME%/var/log/sasdatasvrc/pgupgrade`

  The path typically is `/opt/sas/viya/config/var/log/sasdatasvrc/pgupgrade`. By default, the utility runs in minimal logging mode.

- Windows:
  
  `%SASConfig%/var/log/sasdatasvrc/pgupgrade`
By default, the path is C:\ProgramData\SAS\Viya\var\log\sasdatasvrc\pgupgrade. During deployment, if the %SASConfig% path is set differently than the default path, then the utility selects the new path that is set.

To generate debug logs, add the verbose and log-file arguments to the utility as follows:

`sas-pgupgrade-cli --verbose --log-file ~/viya_postgres_upgrade.log upgrade`

**Linux**

**Note:** You must run the utility as the sas installer user.

There are two ways to run the utility:

**Ansible Playbook**

SAS Viya on Linux generally contains multiple PostgreSQL nodes that are distributed across multiple machines. Therefore, it is recommended to use the Ansible play to run the upgrade utility.

To launch the upgrade play, log on to your Ansible machine. Make sure that you have the `inventory.ini` file that was used to create your current deployment. Run the following command:

```bash
ansible-playbook -i ${playbook} inventory.ini ${playbook}/utility/upgrade-postgresql.yml
```

**Note:** When the upgrade utility is running, it cannot report the current progress. Depending on the size of the data that is being upgraded, it might appear that the utility has stopped running. The status is reported after the upgrade is completed.

**Note:** Ansible can run the utility using only the default method for the upgrade. See Table 5 on page 40.

**Upgrade Using the CLI**

Log on as the sas installer user to each of your physical machines that contain any PGPool or PostgreSQL servers. Run the following command:

```
./sas-pgupgrade-cli upgrade
```

- Make sure that you upgrade the primary server (one or more) first. If you run the tool on a machine containing only a standby server (one or more) without upgrading the primary server, the tool waits until the primary server has been upgraded to the same version that the standby server intends to upgrade to. After the upgrade of the primary server has completed, the standby server stops waiting and the upgrade is resumed.

- Ensure that you do not start the PostgreSQL cluster until all nodes have been upgraded. Attempting to start the entire cluster with multiple nodes at different versions can cause unexpected behaviors.

If you are running multiple PostgreSQL clusters and want to upgrade one or more specific clusters, you must specify the `--cluster` argument. You might specify this argument multiple times to upgrade multiple clusters. Here is an example:

```
./sas-pgupgrade-cli upgrade --cluster postgres
```

**Windows**

The upgrade utility must be run as an administrator.

Specify the Windows account name where the PostgreSQL runs. You are prompted to enter the password for this user before the upgrade begins. Here is an example:

```
./sas-pgupgrade-cli.exe upgrade --postgres-account-username postgres
```
postgres password:

---

**Note:** The Ansible playbook is not supported on Windows.

---

**Primary Data Server Upgrade Types**

The upgrade utility provides three different methods of upgrading PostgreSQL primary servers. These methods are applicable only if you upgrade using the CLI.

*Table 5  Upgrade Type*

<table>
<thead>
<tr>
<th>Upgrade Method</th>
<th>CLI Command</th>
<th>Details</th>
</tr>
</thead>
<tbody>
<tr>
<td>Default</td>
<td><code>sas-pgupgrade-cli upgrade</code></td>
<td>This is the default option. This option performs the upgrade of the primary PostgreSQL server’s data using the <code>pg_upgrade</code> utility. This utility creates a copy of the user’s current database at the new version of PostgreSQL. This copy also serves as a backup of the old database, and gives an option to the user to switch to the old version, if necessary. Therefore, the machine where the utility is running must have extra free disk space that is equal to the space of the current primary data directory. For example, if you have a 100 GB database, free space of at least 100 GB is needed in order for this option to work.</td>
</tr>
<tr>
<td>Upgrade Method</td>
<td>CLI Command</td>
<td>Details</td>
</tr>
<tr>
<td>--------------------------------------</td>
<td>--------------------------------------------------</td>
<td>--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Upgrade with Linkage between Old and New Files</td>
<td><code>sas-pgupgrade-cli upgrade --link</code></td>
<td>This option is triggered by providing the <code>--link</code> argument to <code>sas-pgupgrade-cli</code> by using the upgrade utility. The upgrade is performed by linking PostgreSQL data files between the old version database and the new version database. More details are in the PostgreSQL documentation.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- In this upgrade, a copy of the old database is not maintained. The files are linked between the two versions of the data and modified only if necessary. Therefore, this method of upgrade requires less disk space than the default upgrade method.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- After the upgrade begins, you cannot revert to the old version because a copy of the old version is not maintained. If you want to revert to the old version of the database, create a backup of your current database or create a snapshot of your machines before starting the upgrade.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- This upgrade method is faster than the other methods because it does not copy the files.</td>
</tr>
<tr>
<td>Upgrade with Dump-Restore</td>
<td><code>sas-pgupgrade-cli upgrade --dump-restore-primaries</code></td>
<td>This is the conventional method to upgrade PostgreSQL that precedes the <code>pg_upgrade</code> tool. This is considered to be the slowest and safest upgrade method.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>This upgrade method creates an SQL dump of the current primary, saves the dump to the disk, and restores it at the new version. This upgrade typically requires free disk space that is four times the size of the current database. The upgrade might take multiple hours, depending on the size of the database.</td>
</tr>
</tbody>
</table>

Note:
- For the dump-restore upgrade method to work successfully, the `/etc/hosts` file needs to map the host name to the loop back IP address (127.0.0.1). Otherwise, the upgrade fails with the error `timed out waiting for node to start`. To resolve this error, get the fully qualified domain name (FQDN) and the short host name of your machine by running the following commands:

  ```bash
  hostname -f
  hostname -s
  ```

  Add the following line to the top of your `/etc/hosts` file:
Run the upgrade utility. After the upgrade is complete, remove the preceding line from your /etc/hosts file.

- If you have a manually clustered PGPool and a non-clustered PGPool in your environment, make sure that all the PGPools are clustered before you upgrade.

**Standby Data Server Upgrade Types**

The upgrade utility provides only one method of upgrading the standby server (one or more).

For the standby server (one or more), the utility waits until it detects that the primary server (one or more) has been upgraded to the expected version of PostgreSQL. On completion, the standby server upgrades by connecting to the newly upgraded primary server and performing a `pg_basebackup`.

**Post-upgrade Steps**

After the upgrade is complete, your PostgreSQL clusters are in a shutdown state except the primary server (one or more) that is in a running state. Run the `startall` command to start your upgraded cluster. See Operate a Cluster on page 12 for more information.

If you have shut down your SAS Viya deployment, start all your SAS Viya services. See “Start and Stop All Servers and Services” in SAS Viya Administration: General Servers and Services for more information.

**CAUTION**

Do not restore your old backups after you upgrade PostgreSQL. This might cause the system to go in an inconsistent state. For details, see “Best Practices for Performing Restores” in SAS Viya Administration: Backup and Restore.

---

**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, and auto vacuum on 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-2018-10915 security patch and CVE-2017-7547 security patch on page 45.

**Re-Index and Auto Vacuum Database Tables (Linux)**

A utility script is provided to re-index and auto vacuum each table in the SAS Infrastructure Data Server databases. SAS recommends that you run this script 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.

The script is `sas_reindex_vac.sh` and is located in `/opt/sas/viya/home/libexec/sasdatasvrc/script/maintenance` directory.

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

2. Run the script
   
   `/opt/sas/viya/home/libexec/sasdatasvrc/script/maintenance/sas_reindex_vac.sh -s servicename|clustername`

   In this example, the cluster name is postgres:

   `/opt/sas/viya/home/libexec/sasdatasvrc/script/maintenance/sas_reindex_vac.sh -s postgres`

3. A log file is created and located in `/opt/sas/viya/config/var/log/sasdatasvrc/postgres/pgpool0` For example: `/opt/sas/viya/config/var/log/sasdatasvrc/postgres/pgpool0/sas_reindex_vac_20191024142228.log`

4. If there were no errors in the previous step, then you are done.
   
   If you had stopped all 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 (Linux)

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:

   ```sql
   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.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 root-level 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.

```bash
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-2018-10915 Security Patch**

A new security patch, CVE-2018-10915, fixes a vulnerability that was found in libpq, the default PostgreSQL client library.

Sites that deploy SAS Viya 3.4 on Windows have a newer version of PostgreSQL (version 9.4.19) that contains the fix for this security issue. However, sites running SAS Viya 3.4 or earlier on Red Hat Enterprise Linux and SUSE Linux Enterprise Server, must manually apply patch SAS Infrastructure Data Server with patch CVE-2018-10915.

1. As a SAS install user (sas) or with root-level privileges, sign in to the PGPool server machine.
2. Stop all SAS Viya services.
3. Perform an update using `yum`, or run Ansible with the appropriate playbook.
4. Restart all the SAS Viya services.

**Apply CVE-2017-7547 Security Patch**

PostgreSQL 9.4.13 fixes a password security issue in the database. However, existing PostgreSQL 9.4.9 databases that are upgraded to 9.4.13 need to be patched manually.

After you upgrade PostgreSQL servers from version 9.4.9 to 9.4.13, apply the security patch to the existing databases to prevent the vulnerability. See Changes E.12.2 in CVE-2017-7547. Do not apply the patch on the new deployment. Apply the patch only when a PostgreSQL server version update is performed. To apply the patch, log on to the machine that is running the Pgpool-II server and execute the `CVE-2017-7547.sh` script. You can accept the default settings, unless you have changed them in your deployment.

Make sure the following conditions are met, before running the script.
- You must have an account with sudo privileges to run the script.
- Stop all SAS Viya services and start only the SAS Infrastructure Data Server.
- You must know the database superuser (for example: dbmsowner) password.

Here is an example to execute the script on RHEL 6.x (or equivalent distribution):

```
Note: Use appropriate commands for other RHEL versions.
```
Retrieve the superuser password from Consul.
source /opt/sas/viya/config/consul.conf
export CONSUL_TOKEN=$(sudo cat /opt/sas/viya/config/etc/SASSecurityCertificateFramework/tokens/consul/default/client.token)
/opt/sas/viya/home/bin/sas-bootstrap-config kv read config/application/sas/database/postgres/password

# Make sure that only the SAS Infrastructure Data Server is running
sudo service sas-viya-all-services stop
sudo service sas-viya-sasdatasvrc-postgres start

# Run the CVE maintenance script
/opt/sas/viya/home/libexec/sasdatasvrc/script/maintenance/CVE-2017-7547.sh

# Stop the SAS Data Server and restart all services
sudo service sas-viya-sasdatasvrc-postgres stop
sudo service sas-viya-all-services start

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

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 the 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

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 (Linux)

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

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

Log Files

SAS Infrastructure Data Server log files are located in the following path, depending on your operating system:

- **Linux:**
  ```bash
  /opt/sas/viya/config/var/log/sasdatasvrc
  ```

- **Windows:**
  ```bash
  \ProgramData\SAS\Viya\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 20.

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

---

Operate (Linux)

SAS Viya provides a script in `/etc/init.d` that you use to stop, start, restart, and check the status of SAS Message Broker. The script is named, `sas-viya-rabbitmq-server-default`.

**Syntax**

How you run `sas-viya-rabbitmq-server-default` depends on your operating system:

- **Red Hat Enterprise Linux 7.x (or an equivalent distribution) and SUSE Linux Enterprise Server 12.x:**
  ```bash
  sudo systemctl status | stop | start | restart sas-viya-rabbitmq-server-default
  ```

- **Red Hat Enterprise Linux 6.x (or an equivalent distribution):**
  ```bash
  sudo service sas-viya-rabbitmq-server-default status | stop | start | restart
  ```
Usage Notes and Tips

- You must be logged on to the machine where SAS Message Broker resides. Also, you must have root-level privileges to run this script.

- If there are multiple instances of SAS Message Broker:
  - start them in the reverse sequence in which you stopped them.
  - stop them in the reverse sequence in which you started them.

- 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.

- On Linux systems that support systemd, use the systemctl command when running sas-viya-rabbitmq-server-default. The systemctl command maintains a record of service status that the service command and a direct call does not use.

CAUTION
On Red Hat Enterprise Linux 7.x (or an equivalent distribution) and SUSE Linux Enterprise Server 12.x, do not mix System V init and systemd commands. Mixing the System V init (service command) with the systemd (systemctl command) causes several issues. The systemctl command knows nothing about a SAS Viya service started with the service command. If you start sas-viya-rabbitmq-server-default on RHEL 7.x with the service command, and later attempt to shut down SAS Message Broker using the systemctl command, the configuration server stops responding and does not shut down.

Examples

- To check status of SAS Message Broker on Red Hat Enterprise Linux 7.x (or an equivalent distribution) and SUSE Linux Enterprise Server 12.x:
  
    sudo systemctl status sas-viya-rabbitmq-server-default

- To stop SAS Message Broker on Red Hat Enterprise Linux 6.x (or an equivalent distribution):
  
    sudo service sas-viya-rabbitmq-server-default stop

- To start SAS Message Broker on Red Hat Enterprise Linux 7.x (or an equivalent distribution) and SUSE Linux Enterprise Server 12.x:
  
    sudo systemctl start sas-viya-rabbitmq-server-default

- To restart SAS Message Broker on Red Hat Enterprise Linux 6.x (or an equivalent distribution):
  
    sudo service sas-viya-rabbitmq-server-default restart

Operate (Windows)

Using the Microsoft Management Console (MMC) Services snap-in, you can start, stop, and restart SAS Message Broker.
Because there is a particular sequence in which the servers and services must be started and stopped, the individual services are not configured to run automatically when the SAS Viya machine is booted.

**IMPORTANT** SAS Configuration Service (Consul), SAS Infrastructure Data Server (PostgreSQL), SAS HTTP Proxy Server (Apache HTTP Server), and SAS Message Broker (RabbitMQ) are dependencies for the other SAS Viya services. If you are operating one or more services individually, always start each of these four services first and stop them last.

**Note:** There is one service, SAS Services Manager, that you can use to start and stop all SAS Viya servers and services. SAS Services Manager recognizes the dependencies between services and starts and stops services in the correct sequence. For more information, see “Start and Stop All Servers and Services” in *SAS Viya Administration: General Servers and Services.*
### Add a Node

It is recommended that you use Ansible to add a node to the RabbitMQ server. Follow the steps to add a node to the RabbitMQ server:

1. **Edit the Ansible `inventory.ini` file.** Add the host at the top of the file. One or more hosts should already exist at the top of the file.

   For example:
   ```ini
   rabbitNode2 ansible_host=0.0.0.0  ansible_user=<ansible-user>
   ansible_ssh_private_key_file=~/.ssh/<ansible-ssh-key>
   ```

2. **Add `rabbitNode2` to the rabbitmq host group within the inventory.ini file.** The host group should look similar to the following:
   ```
   [rabbitmq]
   deployPrimary
   rabbitNode1
   rabbitNode2
   ```

3. **Re-run the ansible playbook.**

4. **Verify that the new node is added to the rabbitMQ cluster.** Run the following command:
   ```
   /opt/sas/viya/home/sbin/rabbitmqctl cluster_status
   ```

### Remove a Node

Follow the steps to remove a node from a RabbitMQ cluster:

1. **With root privileges or as a `sasrabbitmq` user,** log on to the node that you want to remove.

2. **Verify that the cluster is healthy.**

   ```
   /opt/sas/viya/home/sbin/rabbitmqctl cluster_status
   ```

   The status of all the nodes should be **up.** Make sure that the output does not have any partitions and alarms.

3. **Stop the RabbitMQ applications.**

   ```
   /opt/sas/viya/home/sbin/rabbitmqctl stop_app
   ```

4. **Reset the RabbitMQ node.**

   ```
   /opt/sas/viya/home/sbin/rabbitmqctl reset
   ```

5. **Start RabbitMQ applications.**

   ```
   /opt/sas/viya/home/sbin/rabbitmqctl start_app
   ```

6. **Run the following command on all RabbitMQ nodes and verify that the node was removed.**

   ```
   /opt/sas/viya/home/sbin/rabbitmqctl cluster_status
   ```
If the status returns the expected results, then stop RabbitMQ on the node that you are removing from the cluster. You should have root privileges to run the following command appropriate for your operation system.

For Red Hat Enterprise Linux 7.x (or equivalent distribution) and SUSE Linux Enterprise Server 12.x:

```bash
sudo systemctl stop sas-viya-rabbitmq-server-default
```

For Red Hat Enterprise Linux 6.x (or an equivalent distribution):

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

7 Disable RabbitMQ so that it will not start upon the next boot. You should have root privileges to run the following command appropriate for your operation system.

For Red Hat Enterprise Linux 7.x (or equivalent distribution) and SUSE Linux Enterprise Server 12.x:

```bash
sudo systemctl disable sas-viya-rabbitmq-server-default
```

For Red Hat Enterprise Linux 6.x (or an equivalent distribution):

```bash
chkconfig sas-viya-rabbitmq-server-default off
```

In addition, you can also place `sas-viya-rabbitmq-server-default` in `/opt/sas/viya/config/etc/viya-svc-mgr/svc-ignore`.

---

### 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
SAS Message Broker configuration files are located in /opt/sas/viya/config/etc/rabbitmq-server/.

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

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.

See Also
SAS Viya Administration: Tuning
Operate (Linux)

SAS Viya provides scripts in `/etc/init.d` that you use to stop, start, restart, and check the status of SAS Cache Locator and SAS Cache Server. The scripts are named `sas-viya-cachelocator-default` and `sas-viya-cacheserver-default`, respectively.

Syntax

How you run `sas-viya-cachelocator-default` and `sas-viya-cacheserver-default` depends on your operating system:

- **Red Hat Enterprise Linux 7.x (or an equivalent distribution) and SUSE Linux Enterprise Server 12.x:**
  
  `sudo systemctl status | stop | start | restart sas-viya-cachelocator-default`
  
  `sudo systemctl status | stop | start | restart sas-viya-cacheserver-default`

- **Red Hat Enterprise Linux 6.x (or an equivalent distribution):**
  
  `sudo service sas-viya-cachelocator-default status | stop | start | restart`
  
  `sudo service sas-viya-cacheserver-default status | stop | start | restart`

Usage Notes and Tips

- You must be logged on to the machine where the SAS Cache Locator and SAS Cache Server services reside. Also, you must have root-level privileges to run these scripts.

- 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.

- On Linux systems that support systemd, use the `systemctl` command when running the individual service and server scripts. The `systemctl` command maintains a record of service status that the `service` command and a direct call does not use.

**CAUTION**

On Red Hat Enterprise Linux 7.x (or an equivalent distribution) and SUSE Linux Enterprise Server 12.x, do not mix System V init and systemd commands. Mixing the System V init (service command) with the systemd (systemctl command) causes several issues. The `systemctl` command knows nothing about a SAS Viya service started with the `service` command. If you start `sas-viya-cachelocator-default` on RHEL 7.x with the `service` command, and later attempt to shut down SAS Cache Locator using the `systemctl` command, the cache locator stops responding and does not shut down.

Examples

- To check status of SAS Cache Locator on Red Hat Enterprise Linux 7.x (or an equivalent distribution) and SUSE Linux Enterprise Server 12.x:
  
  `sudo systemctl status sas-viya-cachelocator-default`

- To stop SAS Cache Server on Red Hat Enterprise Linux 6.x (or an equivalent distribution):
  
  `sudo service sas-viya-cacheserver-default stop`

- To start SAS Cache Locator on Red Hat Enterprise Linux 7.x (or an equivalent distribution) and SUSE Linux Enterprise Server 12.x:
  
  `sudo systemctl start sas-viya-cachelocator-default`

- To restart CAS Cache Server on Red Hat Enterprise Linux 6.x (or an equivalent distribution):
**Operate (Windows)**

Using the Microsoft Management Console (MCC) Services snap-in, you can start, stop, and restart SAS Cache Locator and the SAS Cache Server.

*Figure 8*  **SAS Cache Locator in the Services Snap-In**

Because there is a particular sequence in which the servers and services must be started and stopped, the individual services are not configured to run automatically when the SAS Viya machine is booted.

**IMPORTANT**  SAS Configuration Service (Consul), SAS Infrastructure Data Server (PostgreSQL), SAS HTTP Proxy Server (Apache HTTP Server), and SAS Message Broker (RabbitMQ) are dependencies for the other SAS Viya services. If you are operating one or more services individually, always start each of these four services first and stop them last.
Note: There is one service, SAS Services Manager, that you can use to start and stop all SAS Viya servers and services. SAS Services Manager recognizes the dependencies between services and starts and stops services in the correct sequence. For more information, see “Start and Stop All Servers and Services” in SAS Viya Administration: General Servers and Services.

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.

SAS Viya supports the following versions of Apache HTTP Server:

- Red Hat Linux 6.x uses Apache HTTP Server upstream v2.2.
- Red Hat Linux 7.x uses Apache HTTP Server upstream v2.4.
- SUSE Linux Enterprise Server 12.x uses Apache HTTP Server upstream v2.4.
- Windows uses Apache HTTP Server v2.4.34.

For more information, see “Apache httpd” in SAS Viya for Linux: Deployment Guide.

How To

Operate (Linux)

Note: You must be logged on to the machine where SAS HTTP Proxy Server resides, and you must have root-level privileges to run this script.

Note: For complete information about Apache httpd arguments, see Apache HTTP Options.

To run `apache httpd`:

```bash
sudo apachectl status | stop | start | restart
```

To run `sas-viya-httpproxy-default` scripts, use the command that is appropriate for your operating system:

- On Red Hat Enterprise Linux 6.x (or an equivalent distribution):
  ```bash
  sudo service sas-viya-httpproxy-default status | stop | start | restart
  ```

- On Red Hat Enterprise Linux 7.x (or an equivalent distribution):
  ```bash
  sudo systemctl status | stop | start | restart sas-viya-httpproxy-default
  ```

- On SUSE Linux Enterprise Server 12.x:
  ```bash
  sudo systemctl status | stop | start | restart sas-viya-httpproxy-default
  ```
Operate (Windows)

Using the Microsoft Management Console (MMC) Services snap-in, you can start, stop, and restart SAS HTTP Proxy Server.

Figure 9  SAS Apache HTTP Server in the Services Snap-In

Because there is a particular sequence in which the servers and services must be started and stopped, the individual services are not configured to run automatically when the SAS Viya machine is booted.

**IMPORTANT**  SAS Configuration Service (Consul), SAS Infrastructure Data Server (PostgreSQL), SAS HTTP Proxy Server (Apache HTTP Server), and SAS Message Broker (RabbitMQ) are dependencies for the other SAS Viya services. If you are operating one or more services individually, always start each of these four services first and stop them last.

**Note:** There is one service, SAS Services Manager, that you can use to start and stop all SAS Viya servers and services. SAS Services Manager recognizes the dependencies between services and starts and stops services in the correct sequence. For more information, see “Start and Stop All Servers and Services” in SAS Viya Administration: General Servers and Services.
Configure External Reverse Proxy

To configure SAS Viya for an external reverse proxy, including load balancers that act as a reverse proxy:

1. Add properties to ensure that applications that generate links to SAS Viya objects (such as SAS Visual Analytics reports) are aware of the external reverse proxy.

   From the machine where the SAS Viya internal Apache HTTP Server resides, run the following commands to add the appropriate property values. Here is an example:

   ```
   source /opt/sas/viya/config/consul.conf
   /opt/sas/viya/home/bin/sas-bootstrap-config kv write
   config/viya/sas.httpproxy.external.hostname reverseproxy_hostname
   /opt/sas/viya/home/bin/sas-bootstrap-config kv write
   config/viya/sas.httpproxy.external.port reverseproxy_port
   ```

   **Note:** Specify each of these commands on a single line. Multiple lines are used here to improve readability.

2. On the external reverse proxy, set the X-Forwarded-Proto and X-Forwarded-Port headers to the protocol and port that the client is using to connect to the external reverse proxy.

3. From the SAS Viya internal Apache HTTP server machine, comment out the X-Forwarded-Proto and X-Forwarded-Port lines in the petrichor.conf file as follows:

   a. Locate the petrichor.conf file according to the platform used:

      ```
      UNIX           /etc/http/conf.d
      SUSE Linux     /etc/apache2/conf.d
      Windows        C:\ProgramData\SAS\Viya\etc\httpd\conf.d
      ```

   b. Make a backup copy of the petrichor.conf file.

   c. Locate the following lines in the petrichor.conf file:

      ```
      # Default the X-Forwarded-Proto header to http
      RequestHeader set X-Forwarded-Proto http

      # Set the X-Forwarded-Proto header if request is over HTTPS
      RewriteCond "\%{HTTPS}" "on"
      RewriteRule ^.*$ - [ENV=HTTPS:true]
      RequestHeader set X-Forwarded-Proto https env=HTTPS
      RequestHeader set X-Forwarded-Port 443 env=HTTPS
      ```

   d. Comment out these lines so that they appear as follows:

      ```
      # Default the X-Forwarded-Proto header to http
      #RequestHeader set X-Forwarded-Proto http
      ```
# Set the X-Forwarded-Proto header if request is over HTTPS
#RewriteCond %{HTTPS} "on"
#RewriteRule ^.*$ - [ENV=HTTPS:true]
#RequestHeader set X-Forwarded-Proto https env=HTTPS
#RequestHeader set X-Forwarded-Port 443 env=HTTPS

4  Start (or restart) the SAS Viya internal Apache HTTP Server.

**Note:**

- The external reverse proxy must use HTTPS.
- The external reverse proxy must forward requests through the SAS Viya internal Apache HTTP Server without changing the URL path.
- In environments with multiple SAS Viya internal Apache HTTP servers, you should configure the external reverse proxy to route requests to an active HTTPD instance. Round-robin routing or load-balanced routing is recommended.

## Concepts

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

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

## Log Files

**Note:** You must be logged on with root-level privileges to the machine where the service resides in order to view log files.

By default, Apache HTTP Server log files are located in the following directory, as appropriate for your operating system:

- On Red Hat Enterprise Linux and equivalent distributions:
  
  /var/log/httpd

- On SUSE Linux Enterprise Server 12.x:

  /var/log/apache2