



# SAS<sup>®</sup> Viya<sup>®</sup> Platform Administration: Tuning

**2023.03 - 2023.09**

This document might apply to additional versions of the software. Open this document in [SAS Help Center](#) and click on the version in the banner to see all available versions.

<b><i>Tuning the SAS Viya Platform</i></b> .....	<b>2</b>
<b><i>Provisioning Hardware for Performance</i></b> .....	<b>3</b>
Storage Considerations for CAS and Compute .....	3
Machine Specifications .....	5
Machine Resource Monitoring Tools .....	5
Tuning Machine Resource Requests .....	6
<b><i>Managing Cluster Resources</i></b> .....	<b>6</b>
Understanding Key Components .....	6
Tuning Workload Allocations .....	7
SAS <sup>®</sup> Workload Management .....	8
<b><i>Reducing Pod Start-Up Times</i></b> .....	<b>8</b>
<b><i>Tuning the CAS Server</i></b> .....	<b>8</b>
Basic CAS Considerations .....	9
Change the Number of Workers for MPP CAS .....	9
Hardware Recommendations .....	10
Vendor-Specific Settings .....	10
Multi-Tenancy Considerations .....	10
<b><i>Tuning Programming Resources and Programs</i></b> .....	<b>11</b>
Storage for Programming Pods .....	11
Modifying Compute Resource Usage .....	12
Program Options to Optimize Performance .....	13
Compute Server Multithreading .....	13
Tuning Program Performance in the CAS Server .....	14

Managing Large or Long-Running Jobs .....	14
<b><i>Tuning the OpenSearch Component</i></b> .....	<b>15</b>
<b><i>Tuning the LDAP Connection Pool</i></b> .....	<b>16</b>
Determining When Tuning of LDAP Connections Is Needed .....	16
Recommendations .....	16
<b><i>Tuning the JDBC Connection Pool</i></b> .....	<b>17</b>
Deployment Size Definitions .....	17
Configure Deployment Size .....	18
Tuning Data Source Properties .....	18
Tuning the Data Source Scaling Factor .....	19
<b><i>Tuning the PostgreSQL Server</i></b> .....	<b>20</b>
Considerations .....	20
Tuning Memory and CPU Resources .....	21
Recommendations .....	22
<b><i>Tuning SAS Visual Analytics</i></b> .....	<b>26</b>
Modifying Performance Settings .....	26
Avoiding Out-of-Memory Issues .....	27
<b><i>Tuning SAS Viya with SingleStore</i></b> .....	<b>28</b>
Resource Considerations .....	28
Adjusting Resource Allocations .....	29

---

## Tuning the SAS Viya Platform

After you have deployed the SAS Viya platform, you might need to modify individual components for scalability and performance. This guide describes tuning methodologies that are appropriate for the SAS Viya platform in a Kubernetes environment.

Tuning for scalability often targets the ability of a component to adapt to a greater or lesser intensity of use or volume of use while meeting business requirements. Common objectives of scaling a component or system include increasing the capacity for growth, improving the speed or processing efficiency of the component, or rebalancing the load on components.

Performance requirements are usually identified in terms of transaction response time, number of transactions per second, throughput, resource utilization, total cost per transaction, availability, and more. When the performance of SAS components lags, you can often change some settings to address the situation.

Be aware that many of the tuning recommendations in this guide involve upgraded or additional resources. As a result, some of these recommendations are likely to result in increased costs from your cloud provider.

# Provisioning Hardware for Performance

Performance testing conducted by SAS indicates that your hardware can have an outsize impact on the performance of SAS Viya platform deployments. The main hardware-related factors that affect the performance of SAS deployments are storage and the resource provisioning of individual machines.

## Storage Considerations for CAS and Compute

The SAS Viya platform requires both temporary and persistent storage. Your assortment of SAS product offerings, the number of users, and the amount of data that is processed all affect storage sizing requirements for performance. The performance of various storage options is also highly vendor-dependent.

SAS Viya platform performance is affected most readily by the CAS server and compute (programming) nodes, which perform I/O-intensive work. Provision the CAS server and compute nodes with both high-performing temporary storage and shared storage, as described in the following table:

**Table 1** Required Storage by Component and Type

Component	Temporary (Ephemeral) Storage	Shared (Persistent) Storage
CAS server	<p>CAS disk cache: temporary storage for data that the CAS server is processing.</p> <p>The required amount depends on the size of table data and how the CAS server accesses it. However, inadequate space for the CAS disk cache can cause poor performance and node failure.</p>	<p>Default caslibs (cas-default-data): high-performing storage is recommended.</p> <p>CAS configuration data, caslib management privileges (cas-default-permstore): high-performing storage is not necessary.</p>
Compute server	<p>SASWORK directory: a storage location that is automatically defined by the compute server at the beginning of each SAS session or job; temporarily stores files that you create and files that SAS creates.</p> <p>This directory might require additional capacity that extends beyond the limits of local disk resources. Appropriate sizing for SASWORK involves multiple variables, including table sizes and job concurrency.</p>	<p>User private directories (home directories) and configuration data: high-performing storage is not necessary.</p>

Every CAS server pod requires a mount for the CAS disk cache. By default, CAS pods are backed by an emptyDir volume for this caching space, which provides overflow capacity for data that the CAS server processes in memory. This volume should be highly optimized for performance. SAS recommends using local ephemeral storage, such as an SSD volume that is memory-mapped to provide overflow capacity. However, the size of ephemeral storage might be limited. For a large deployment with heavy CAS usage, consider adding permanent attached storage to increase disk cache capacity.

Similarly, shared storage for caslibs should be optimized. By contrast, shared storage for configuration data does not need to be optimized for performance.

SAS Compute Server is part of the SAS Programming Run-Time Environment, which also includes SAS/CONNECT Server and SAS Batch Server. These components handle workloads in the compute workload class. By default, sas-programming-environment pods are backed by an emptyDir volume named `viya`. This volume is mounted automatically and uses storage on the node.

Using the default emptyDir volume is not recommended for compute workloads because SAS programming components can consume large amounts of storage quickly and cause nodes to shut down. Where feasible, SAS recommends provisioning local, non-NFS storage for SASWORK for improved performance. However, local disk resources might be inadequate. The tradeoffs to consider are as follows:

- whether persistent, external storage performs well enough to handle your compute workloads
- whether SAS Batch Server is running jobs in checkpoint/restart mode, which requires external storage
- whether high-performing local storage on the node can scale adequately for your expected compute workloads

In general, solid state drives (SSDs) perform better than hard disk drives (HDDs). In cloud deployments, local, ephemeral storage often performs better than persistent storage and is preferred over NFS-based storage. However, additional network bandwidth can improve the performance of network-based storage. As you select a file system option, consider both disk size and the number of disks. Speed is directly related to the amount of disk space that is provisioned. It is also constrained by VM network and I/O limits.

You should consult your cloud provider's storage documentation to ensure that the storage environment can provide the level of I/O that is required. SAS Technical Support has found that many performance issues reported by SAS customers can be directly attributed to insufficient levels of I/O throughput.

The use of hostPath volumes to mount files or directories from the host node's file system into pods is not recommended for most situations. Because these volumes present security risks, it is a best practice to avoid using hostPaths.

The SAS Viya platform supports encryption of data at rest; however, encryption typically causes degraded performance.

For more information about configuring storage for SAS programming components, see [“Tuning Programming Resources and Programs” on page 11](#).

The SAS Viya Platform System Requirements include hardware recommendations that are specific to each cloud provider. These recommendations are updated periodically as SAS Viya platform and third-party offerings evolve. Be sure to consult both the hardware requirements and the [“Resource Guidelines” in \*System Requirements for the SAS Viya Platform\*](#).

---

## Machine Specifications

The peak I/O throughput requirements of your SAS Viya platform deployment might exceed the capabilities of your storage configuration or of your machines themselves. Select VM instance types with the highest available I/O throughput levels. On Microsoft Azure, Premium storage is required in order to achieve these I/O throughput levels.

SAS recommends selecting hardware that offers the highest I/O levels for your budget. The I/O metric is especially important for the CAS and compute workloads. Machines should also be configured for maximum memory bandwidth.

CPU resources are critical. The CAS server is multi-threaded, and CAS procedures can take advantage of multi-core processors. The latest server generations are ideal for SAS applications, which demand faster CPUs, improved local disk performance, and increased memory resources.

SAS strongly recommends consulting with a sizing expert to obtain an official hardware recommendation that is based on your deployment type, the estimated SAS workload, and the number of users. The SAS World-Wide Sizing Team can help you plan for adequate hardware resources, based on your deployment size and usage patterns. To request sizing expertise, contact your SAS account representative. If you need assistance in determining your SAS account representative, send an email to [contactcenter@sas.com](mailto:contactcenter@sas.com).

For more information about recommended hardware for the CAS server, see [Tuning the CAS Server: Hardware Recommendations on page 10](#).

---

## Machine Resource Monitoring Tools

SAS Viya Monitoring for Kubernetes provides optional monitoring tools that are designed specifically for the SAS Viya platform. These tools can help you to monitor resource usage of key components. SAS Viya Monitoring for Kubernetes deploys standard open-source tools to collect and display the collected metric data and to manage alerts that are triggered by metric values. SAS Viya Monitoring for Kubernetes includes the following tools:

- Prometheus for data collection
- Alertmanager to handle alerts
- Grafana for the display of graphical reports

SAS provides documentation to help you deploy and use these tools with the SAS Viya platform. See [“About SAS Viya Monitoring for Kubernetes” in SAS Viya Platform: Log and Metric Monitoring](#) for more information.

Multiple pre-defined dashboards are included to display collected data related to the CAS server, the PostgreSQL cluster and server, Opensearch, SASRabbitMQ, and SAS Viya platform services. A set of dashboards is also included to help the Kubernetes cluster administrator gauge resource utilization by the cluster and networking resources.

SAS strongly recommends using these tools to gauge resource consumption as users begin to use the SAS Viya platform. You should plan for ongoing monitoring so that resources can be used efficiently to support optimal performance.

---

## Tuning Machine Resource Requests

Both the SAS administrator and the cluster administrator have opportunities to adjust properties that affect pod and client requests for CPU and RAM resources. A SAS administrator typically uses SAS Environment Manager to specify values for CPU requests, CPU limits, memory requests, and memory limits that the launcher service provides. As an alternative, a cluster administrator can set these values in pod templates by changing the values for properties in a pod template annotation.

When a client fails to provide a CPU or RAM request value for a job, default values determine the number of CPU cores or the amount of RAM to which the client has access. Changing these values can help to ensure that SAS applications receive a higher quality of service from the infrastructure. Specifying a value for maximum CPU or RAM requests can prevent applications from requesting a greater amount of resources than is appropriate for the typical number of incoming requests and the available resource levels on your nodes.

For more information about settings that affect resource requests, see [“Manage Requests and Limits for CPU and Memory”](#) in *SAS Viya Platform: Programming Run-Time Servers*.

---

## Managing Cluster Resources

SAS Viya platform components perform tasks that are associated with [multiple workload classes](#). Each class of workload has a unique set of attributes that you must consider when planning your deployment. When planning the placement of each workload, it is helpful to understand individual platform components and their roles. You should also think beyond the initial deployment and consider the Kubernetes maintenance life cycle.

---

## Understanding Key Components

The CAS server is discussed in detail in [“Tuning the CAS Server”](#) on page 8. The cas workload class is the only one that must have a dedicated node pool; other node pools can potentially be shared among multiple workloads, or perhaps between separate deployments. However, other workload classes also merit special consideration during resource planning.

Components that contribute to the compute workload class are discussed in detail in [“Tuning Programming Resources and Programs”](#) on page 11. This workload class consists of non-CAS processing and analytics tasks. If a large number of SAS Viya platform users are using SAS Studio or SAS Model Studio, you might need to allocate more nodes to the compute server, which runs in sas-programming-environment pods.

Compute processes are stateful processes that run to completion. They are not long-running services. They handle batch jobs, SAS Studio sessions, parallel compute processes, and more when requested by the Launcher service.

Consider dedicating a node pool to compute workloads. If you instead provision a single compute node pool for multiple SAS Viya platform deployments, size it appropriately. Select appropriate node sizes, but also consider the number of nodes in the node pool (Kubernetes minimum and maximum values) and the max\_pods value for the nodes. Setting the max\_pods can prevent the nodes from being overloaded, but it can result in higher costs for the cluster.

Pod resource management is also important. Base SAS and the SAS Viya platform compute servers perform I/O through the operating system file cache by default. In previous versions of the SAS Viya platform, sufficient swap space was enough to ensure that all jobs would run to completion without out-of-memory errors. The SAS Viya platform deployed in a Kubernetes cluster introduces an additional layer of memory management: Kubernetes pod memory request and limit values.

## Tuning Workload Allocations

For optimal performance of the SAS Viya platform, SAS strongly recommends labeling and tainting all your nodes, especially the CAS nodes. Providing taints, tolerations, and labels can enable SAS Viya platform deployments to scale in a way that allows for optimal node usage while managing workloads effectively. When labeling and tainting all the nodes for SAS components, it is important to reserve a “default node pool” to host components such as the ingress controller, monitoring tools, and any other applications that are required.

If you are managing a large SAS Viya platform deployment with 50–75 end users, keep in mind that Kubernetes is designed to accommodate a maximum number of nodes, typically with a maximum of 110 pods per node. (Each Kubernetes distribution and cloud provider has different maximum values.) Once you have performed the recommended steps to label and taint nodes for the CAS server, you might find that more pods land on your remaining nodes.

One way to avoid hitting pod limits on these nodes is to deploy a cluster autoscaler. An autoscaler enables Kubernetes to request additional nodes from the infrastructure provider as needed to handle additional workloads. In many environments, such as Microsoft Azure or GCP, cluster autoscaling is deployed by default. In AWS environments, additional steps are required to install and configure the cluster autoscaler.

It is often a good strategy to create a custom cluster topology to accommodate the workloads for your unique deployment. Among the tools that you can use to create such a topology are custom labels and preferred scheduling, including taints and tolerations that enable pods to run on tainted nodes if other resources are not available. Consider the following tips as you plan your topology:

- Preferred scheduling can be more flexible and robust than strict placement of workloads. Using a `preferredDuringSchedulingIgnoredDuringExecution` `nodeAffinity` for your pods means that the pods might run on a node outside of the target node pool. The advantage of preferred scheduling is that pods will run somewhere, even in situations of resource scarcity.
- Avoid extending the default workload definitions ([workload.sas.com](https://workload.sas.com)) for your custom labels and taints. Instead, define your own labels. If you try to reuse SAS workload definitions in a custom topology, you might create conflicts with the SAS software, and it might be more challenging to discuss your environment with SAS Technical Support.
- Create custom labels that can be stacked, enabling you to have multiple labels on each node. If you reuse SAS labels, they are not stackable. For example, a node can have either `workload.sas.com/class=stateful` or `workload.sas.com/class=stateless`, but not both.
- Use dedicated node pools carefully. If you place too many components in dedicated node pools or add too many labels and taints to nodes, administration becomes challenging. Consider whether a better strategy would be to use a dedicated Kubernetes cluster for distinct SAS Viya platform deployments.
- Depending on how components are used, it might be preferable to use larger node pools or nodes with additional resources rather than to deploy additional node pools. For example, in a multi-tenant deployment, SAS Infrastructure Data Server (in the stateful workload class) is used more heavily.
- The SAS product offerings in your environment influence service usage. For example, if you have deployed offerings that use SAS Micro Analytic Service, you might want to create dedicated node



pools for the stateless workload. This ensures that SAS Micro Analytic Service does not compete for resources with other SAS Viya platform services.

For general information about how to tune SAS Micro Analytic Service, see “[SAS Micro Analytic Service Tuning Guidelines](#)” in *SAS Micro Analytic Service: Programming and Administration Guide*.

Cloud vendors apply different sets of requirements and restrictions to node pools. When you consider increasing the size of your Kubernetes cluster, verify that your vendor of choice allows you to add nodes to the node pools that you have configured and to add node pools to your cluster.

---

## SAS® Workload Management

Kubernetes provides many workload management capabilities for a SAS Viya platform deployment, such as managing node and resource availability. SAS Workload Management extends the workload management capabilities of Kubernetes. The SAS Workload Orchestrator service of SAS Workload Management improves the performance of the SAS Viya platform by adding priority-based queues for jobs in the SAS Programming Runtime Environment.

SAS Workload Orchestrator jobs are pod requests that run only on nodes where the SAS Workload Orchestrator daemonset has been deployed. The daemonset is automatically deployed on nodes that have the label `workload.sas.com/class=compute`. You can add nodes with this label as a post-deployment step. You can also monitor and manage SAS Workload Orchestrator jobs and queues and view logs using the Workload Orchestrator page in SAS Environment Manager. For more information, see [SAS Viya Platform: Workload Management](#).

---

## Reducing Pod Start-Up Times

Earlier versions of Kubernetes include default kubelet settings that might be too low for the SAS products in your deployment. The queries per second (QPS) to kubelet parameter is set to 5 until Kubernetes 1.27, when it is increased to 50. Typically, 5 is too low. In addition, the number of pods that can start simultaneously on each node might be too low in these earlier versions of Kubernetes.

Kubernetes 1.27 is supported for a deployment of the SAS Viya platform in 2023.09 and later. SAS recommends that you upgrade your cluster in order to reduce pod start-up times and increase the number of pods that can be started.

SAS Compute servers have a default time-out of 60 seconds. The higher QPS value improves pod start-up times. When faster pod start-up is combined with an increased number of pods that can be started simultaneously, upgrading to Kubernetes 1.27 might enable more pods to start within the 60-second time-out period.

---

## Tuning the CAS Server

The CAS server provides the SAS Viya platform with the in-memory run-time environment in which data management and analytics take place. A CAS server consists of at least one CAS controller and at least one worker, which are deployed on the same (single) node by default. A CAS server can also



be deployed across multiple nodes. Using multiple worker nodes can improve CAS server performance.

---

## Basic CAS Considerations

Deploying the CAS server on a single node facilitates symmetric multi-processing of data (SMP CAS). A single-node CAS server performs serial loads of data into memory from a supported data source. The in-memory analytic features of a distributed CAS server are available to the single-node CAS server.

Distributing the CAS server across multiple nodes enables massively parallel processing (MPP). An advantage of MPP is that, whenever possible, data is loaded into memory in parallel, which can produce a faster load time. However, MPP CAS does not always improve performance. SMP CAS might be preferable if your SAS Viya platform deployment generally handles relatively small data sets. With MPP CAS, the additional communications flows among CAS nodes can add latency.

By default, CAS is deployed as SMP. To deploy MPP CAS, add a reference to the cas-server overlay in the `resources` block of the `kustomization.yaml` file. The README file in `$deploy/sas-bases/overlays/cas-server/` (for Markdown) or `$deploy/sas-bases/docs/mpp_cas_server_for_sas_viya.htm` (for HTML) contains instructions.

Because CAS was designed with performance and availability as primary objectives, it should be hosted on dedicated resources, with each CAS pod placed on its own dedicated node. The SAS Viya platform includes an operator to help you manage CAS resources. By default, the CAS operator uses auto-resourcing to utilize the full resources of a node. However, the CAS operator enables node resources for supplementary containers to increase when auto-resourcing is configured, and when a burst of activity is required. For example, the SAS Viya platform server is backed up by default every Sunday morning, when the CAS server typically is not running. The backup agent and other containers in the pod can autoscale to temporarily use a larger share of the CPU resources that are available on the node and to complete the backup more rapidly.

You can disable this auto-resourcing feature and set RAM and CPU resources yourself. If you manually allocate resources, you must set both the RAM and CPU resources manually. An example YAML file has been provided to help you disable auto-resourcing. For more information, see the README file located at `sas-bases/overlays/cas-server/auto-resources/README.md` (for Markdown format) or `$deploy/sas-bases/docs/auto_resources_for_cas_server_for_sas_viya.htm` (for HTML format).

However, when auto-resourcing is disabled, you should consider using guaranteed QoS by configuring the manage CPU and RAM transformer, located in `$deploy/sas-bases/examples/cas/configure/cas-manage-cpu-and-memory.yaml`. Guaranteed QoS places the CAS pods in the category of pods that are the last to be evicted by Kubernetes when available resources become insufficient on the node. Another option, node tainting, prevents other (non-CAS) pods from being scheduled on a node.

---

## Change the Number of Workers for MPP CAS

On an MPP CAS server, the number of workers helps determine the processing power of your cluster. By default, MPP CAS has two workers. You can change the number of workers either before or after the initial deployment of the SAS Viya platform.

Verify that your cluster has adequate resources for the worker nodes, and add nodes if necessary. Then perform the steps that are described in [“Change the Number of Workers for MPP CAS” in SAS Viya Platform: Deployment Guide](#).

Adding workers does not require a restart. Existing SAS sessions will not reallocate or load balance to use the new workers, but new sessions should take advantage of the new workers.

Removing workers after the initial deployment requires deleting the CAS deployment, modifying the YAML file, restarting the CAS server, reloading your data, and starting new SAS sessions.

---

## Hardware Recommendations

CAS machines require adequate CPU and memory resources for your deployment size and usage patterns. For a large deployment, three CAS nodes with 24 vCPUs each and a total of 192 GB of RAM performed well in SAS testing.

Every CAS pod requires a mount for the CAS disk cache. This storage provides overflow capacity for data that the CAS server processes in memory. The mount is an empty directory that points to `/cas/cache`. It is configured automatically by the deployment process.

Performance improves if you use an SSD storage volume that is memory-mapped for the CAS disk cache. The `CASENV_CAS_DISK_CACHE` environment variable enables you to modify the default setting. For more information, see [“Tune CAS\\_DISK\\_CACHE” in SAS Viya Platform: Deployment Guide](#).

In a cloud deployment, ephemeral storage is recommended. The required amount depends on the size of table data and how the CAS server accesses it. For a large deployment, SAS recommends at least 750 GB for this volume. The following guidance applies to most deployments: provision local storage to be two or three times larger than the amount of memory on the node.

Some CAS procedures are GPU-enabled, meaning that a CUDA-capable GPU provides additional functionality. MPP CAS does not inevitably provide better performance of models that invoke these procedures. Some models perform better when they run on a single machine that has multiple GPUs, or even a single, powerful and highly optimized GPU. However, some of the large-scale deep learning models need many GPUs on dozens of (MPP) CAS worker nodes. Testing with various hardware configurations is therefore required.

---

## Vendor-Specific Settings

In some environments, unique tuning parameters might affect CAS server performance. For example, testing on Red Hat OpenShift for VMware vSphere found that the performance of the CAS server was significantly degraded until a default setting was modified. SAS recommends that you increase the process ID (PID) limit setting for the container runtime daemon named CRI-O well beyond its default value (which is 1024).

---

## Multi-Tenancy Considerations

The deployment of the SAS Viya platform with multi-tenancy creates a provider tenant with a CAS server and tools to onboard tenants. The provider CAS instance is deployed automatically, with

default settings. When provisioning nodes for the provider tenant, keep in mind that the provider CAS instance must be able to support the SAS Viya services that are shared by all tenants.

After tenants have been onboarded, a CAS server is onboarded into each new tenant as a manual step. A newly onboarded tenant requires an instance of CAS before that tenant can be configured or used.

CAS server configuration should be customized per tenant to meet the specific requirements of or typical workloads for that tenant.

---

## Tuning Programming Resources and Programs

The SAS Programming Run-Time Environment includes the SAS Compute Server, SAS/CONNECT Server, and SAS Batch Server. These components produce workloads in the compute workload class. Multiple factors can affect the performance of compute jobs.

Most programs that you created in SAS®9 run by default in the SAS Programming Run-Time Environment, but they deploy Base SAS procedures that have been enhanced to execute in multiple threads for optimal performance. Beginning with SAS 9.4M5, when you license the SAS Viya platform, you can also run these programs in the CAS server, which natively supports multithreaded, in-memory processing. Consider your options for enhancing your SAS programs with multithreading, as well as the resources that will be required in order to support the additional processing overhead.

---

## Storage for Programming Pods

SAS programming components must be able to create temporary files and data sets in a storage volume. By default, all sas-programming-environment pods are backed by an `emptyDir` volume named `viya`. This volume is mounted automatically and uses storage on the node. As discussed in [“Storage Considerations for CAS and Compute” on page 3](#), using the default `emptyDir` volume is not recommended. Make sure that the SASWORK directory, which is allocated in the `viya` volume, is pointing to a file system with adequate resources:

- enough I/O throughput to support the performance requirements of end-users
- enough disk space to support all the temporary files that are created during each SAS session

The amount and type of storage that you select are likely to affect the performance of compute workloads. High-performance storage is important in order to avoid degraded SAS Viya platform performance. Local, ephemeral storage on the node is typically high-performing and is an option for these servers, but you must provide adequate disk space. Otherwise, SAS processes compete for space with the node's kubelet and can cause node failures.

If it is running jobs in SAS checkpoint/restart mode, SAS Batch Server requires persistent external storage for SASWORK (the `viya` volume) so that checkpoint information can be saved. With an additional configuration step, you can configure separate storage for the Batch Server so that slower-performing persistent storage does not cause performance degradation for the Compute Server or SAS/CONNECT server. For more information, see the README file at `$deploy/sas-bases/examples/sas-batch-server/storage/README.md` (for Markdown format) or `$deploy/sas-bases/docs/sas_batch_server_storage_task_for_checkpointrestart.htm` (for HTML format).

---

## Modifying Compute Resource Usage

Two services, the compute service and the launcher service, are involved in initializing compute pods. Tuning parameters are available for these server contexts in SAS Environment Manager, and you can also adjust settings for their Kubernetes pod templates. Server contexts are analogous to SAS®9 Application Servers and can apply to compute servers or batch servers.

In conditions of heavy usage, launching new compute server instances can cause performance degradation. Another tuning option for programming components is enabling compute servers to be reused when a user session is terminated.

By default, the SAS Compute Server launches pods as needed in response to user activity. Compute sessions run under the credentials of the requesting user. If you enable SAS compute servers to run under a shared service account, you can then edit the Compute context in SAS Environment Manager to set the `reuseServerProcesses` attribute to true. You can also set the length of the server time-out. The server continues to run until it either times out or is reused when a new session request is made from the same context.

When SAS Compute servers are configured for reuse, you can also set a value for the minimum number of servers that are available. This configuration enables the SAS compute service to maintain a pool of running Compute servers that can be reused. Whenever a SAS Viya platform deployment is started, the number of servers that are specified for `serverMinAvailable` is also started. Additional servers are pre-started whenever the minimum value of available servers is not met. New tasks are rapidly routed to an idle compute server with minimal wait time.

This setting can optimize performance in high-usage situations, where contention for compute resources can be an issue. If a resource-intensive job runs on a regular schedule in your deployment, this setting can avoid conflicts over access to shared resources while the job runs. For more information, see [“Configure Reusable Compute Servers” in SAS Viya Platform: Server Contexts](#).

You can also improve the performance of compute sessions by modifying properties that affect pod and client requests for CPU and RAM resources. For more information, see [“Tuning Machine Resource Requests” on page 6](#).

You can set resource limits at the level of the session using a context, at the level of the cluster by configuring node pools and allocating workloads, at the level of the individual pods by using a pod template, or in the VM operating system. The decision about where to set these limits depends on multiple factors. Your node pools and any contexts that are applied to sessions affect the computed value for memory limits. For example, if you have configured a separate node pool for compute workloads, you can use more VMs for that node pool and ensure that fewer pods run on them. You can set minimum and maximum values for the number of nodes in each node pool, and you can set a `max_pods` value for each node. Setting the `max_pods` can prevent node resource exhaustion, but it typically incurs higher costs for running the Kubernetes cluster.

If you instead change a resource minimum or maximum value in the SAS Launcher YAML file, it applies to all contexts that are configured for the Launcher service. You can also change resource settings for a single type of context, such as compute. For example, if multiple node pools are labeled for compute pods, compute context settings apply to all these node pools. SAS recommends that you test with different settings while taking the complexity into account.

---

## Program Options to Optimize Performance

Your individual SAS products' user documentation contains information about the available performance options. Consult the [SAS Viya Platform Programming Documentation](#) in SAS Help Center for information about tuning options, such as hyperparameter tuning and multithreading. Multithreading in particular provides performance gains.

Threading refers to the organization of computational work into multiple tasks, or processing units that can be scheduled by the operating system. *Multithreading* therefore refers to the concurrent execution of tasks. Threading technology provides multiple paths of execution within an operating environment. Each path of execution (or thread) can handle a program or data-transfer task. As a result, multiple program tasks and data I/O operations are performed at the same time, in parallel. Each thread requires a certain amount of memory and CPU time. An appropriate operating environment might have multiple CPUs, for example, or multiple CPUs with multiple cores per CPU and even multiple threads per core.

Many SAS components take advantage of threading technologies automatically. Threading is the default system option in all products so that multithreading can occur wherever performance is likely to improve. However, you can control the use of threading through environment variables and system options. Data set options, where available, can also be specified in order to adjust I/O or application processing in threads.

The system options `THREADS`, `CPUCOUNT`, and `NOTHREADS` influence execution throughout the SAS environment where threading is not automatic. If `THREADS` is set, `CPUCOUNT` defaults to the number of threads that are available for processing on the client. `NOTHREADS` disables threading in Base SAS or SAS clients and in products that execute in an environment that does not support multithreading. Procedure statement options are provided to override the system options when necessary.

---

## Compute Server Multithreading

SAS 9® Foundation is the term that SAS applies to the superset of SAS software that can be deployed with Base SAS, such as SAS/ACCESS and SAS/GRAPH. SAS 9® Foundation offerings, such as SAS/BASE, SAS/STAT, or SAS/ETS, include statements and procedures that use the SAS Compute Server for execution. The SAS Compute Server was the precursor to the CAS Server.

In SAS product documentation, the term *Massively Parallel Processing (MPP)* is most often used to describe the SAS LASR Analytic Server or the CAS server. Those technologies can take advantage of multiple machines and their additional resources to complete large jobs more rapidly. By contrast, the term *Symmetric Multiprocessing (SMP)* applies to SAS compute servers, and also to the CAS server or LASR Analytic server when deployed on a single machine.

An SMP server might contain multiple single-core CPUs, or even multiple multi-core CPUs. However, each SMP CPU executes different programs or works on different data sets. SAS®9 Foundation procedures have been enhanced so that they can execute in multiple threads while still using the (SMP) SAS Compute Server.

When multithreading is supported, substantial performance gains can be realized compared to sequential (single-threaded) task execution. However, using a `MULTITHREAD` option has an impact on SAS Compute Server requirements for CPU and RAM resources and should be taken into consideration when you set values for requests and limits. If the CPU limits that are set for the compute pods are lower than the number of CPU cores on the host node, these pods can use only a portion of the available host resources.

---

## Tuning Program Performance in the CAS Server

Newer SAS product offerings are optimized for the SAS Viya platform and use the CAS server for execution, including SAS Viya, SAS Viya Advanced, SAS Visual Forecasting, and offerings that support the CASL Programming Language. When you integrate SAS@9 and SAS Viya platform deployments, the CAS server can execute SAS@9 programs using multithreaded, in-memory processing.

The CAS server can be deployed in either SMP or MPP mode. In MPP mode, which uses multiple CAS worker nodes, multiple threads are always used for task execution, and the `NOTHREADS` programming option has no effect. Some SAS products have additional options for controlling threading, such as `DBSLICE` in SAS/ACCESS. Selected procedures in SAS products, including SAS/STAT, SAS/OR, SAS/ETS, SAS Enterprise Miner, and SAS High-Performance Analytics Server, can execute in either SMP mode or MPP mode using the CAS server. In most of these products, thread controls and execution mode are specified in the `PERFORMANCE` statement.

Although the DATA step can run on the SAS Compute Server in SMP mode, it can also run in CAS. If data tables are very large, executing the DATA step in the CAS server is recommended for improved performance because the CAS server processes tables using multiple threads, executed in multiple pods in parallel.

CAS-based processing can be managed by options like `NWORKERS` and the `Single=YES/NO` option on a data statement. The `NWORKERS` session option lets you specify the number of CAS workers to use in new CAS sessions. Use the `SINGLE=` option in the DATA statement to run the DATA step in a single thread. DATA step processing is then performed by a single server, regardless of the number of available CAS workers.

For more information about threading options for improved performance, see [“Threading in Base SAS” in SAS Programmer’s Guide: Essentials](#). For more information about DATA step processing options, see [“CAS DATA Step Basics” in SAS Cloud Analytic Services: DATA Step Programming](#).


---

## Managing Large or Long-Running Jobs

Some default settings are not appropriate for environments where large jobs are run in batch mode, or where jobs take multiple hours to complete. Batch jobs with large data sets can time out unless the default settings that affect session length are adjusted. In addition, you can avoid errors during the processing of large batch jobs by allocating additional memory resources to the sas-programming-environment container.

A batch job might be launched by a SAS program that connects to the SAS Compute Server for processing. By default, the OAuth token that authorizes the connection to the SAS Compute Server is valid for one hour. When the token expires, running jobs are stopped, and the pods where they are running are destroyed, leaving no useful information in logs. If you think that some batch jobs will take longer than one hour to complete, try increasing the length of time for the authorization token validity.

You can use SAS Environment Manager to modify settings for the authorization token.

- 1 In SAS Environment Manager, click **Configuration** and select the **Definitions** view. In the **Definitions** list, select **sas.logon.jwt**.
- 2 If the service has at least one configuration already, select the SAS Viya platform service whose logon settings you want to modify by clicking  next to that service.



If the service has no existing configuration, click **New Configuration**.

- 3 Scroll down to find the configuration for **policy.accessTokenValiditySeconds**.
- 4 In the **Value** field, specify a new value for the OAuth token in seconds. For example, specify 43200 for a token expiration of 12 hours.
- 5 Take the same steps to set a new value for **policy.global.accessTokenValiditySeconds**.
- 6 Click **Save** in the New sas.logon.jwt Configuration window.

Large data sets or long-running jobs also require additional memory resources. Otherwise, the sas-batch pod is subject to frequent restarts. Within the sas-batch-pod template, modify memory requests and limits and CPU requests for the sas-programming-environment container. Run the following command to modify the sas-batch-pod-template:

```
kubect1 edit podTemplates sas-batch-pod-template
```

Change the **resources** section to increase values for memory and CPU:

```
name: sas-programming-environment
resources:
  limits:
    cpu: 500m
    memory: 2Gi
  requests:
    cpu: 100m
    memory: 1Gi
```

---

## Tuning the OpenSearch Component

By default, the SAS Viya platform deployment process creates one OpenSearch node, which acts as both a controller and a data node. Although this topology is acceptable for initial small-scale deployment and testing environments, it is intended to minimize resources in a temporary situation, such as a testing environment. SAS recommends adding nodes for a production deployment.

The recommended production topology should consist of three or more controller nodes and at least three data storage nodes. This topology provides the following benefits:

- **Performance** — Query loads are shared among the available controllers.
- **High availability** — Failure of any one node does not bring down the service. Performance is slightly degraded while a failed node is brought back online.
- **Data preservation** — Data is redundantly shared among the data nodes. Even a catastrophic failure of one data node does not result in any loss of data. When the failed node is brought back up, the data is reconstructed from duplicate copies on the other nodes.

In order to migrate your deployment from the initial setup to a production-ready topology with multiple OpenSearch nodes, you can modify the cluster without data loss. SAS provides example YAML files to help you transform your topology through an intermediate state and into a final production state. Using the example files as instructed enables the cluster to copy the data that is stored on the existing node across to the data nodes. For more information, see the README file in your `$deploy/sas-bases/examples/configure-elasticsearch/internal/topology` directory.



---

## Tuning the LDAP Connection Pool

The Lightweight Directory Access Protocol (LDAP) service provider supports connection pooling. In LDAP connection pooling, the service provider maintains a pool of previously used connections. When a connection is closed or goes to garbage collection, it returns to the pool to be used again.

By default, no configuration is required for the LDAP service provider to use connection pooling. However, you might want to modify the default setting for optimal performance. For example, when your SAS Viya platform users send large quantities of batch jobs for processing within a small time frame, tuning LDAP connection pooling is required in order to accommodate the load.

---

## Determining When Tuning of LDAP Connections Is Needed

Signs that might indicate that the LDAP connection pool needs to be increased include the following:

- Users experience failures when they attempt to log on to a SAS Viya product interface.
- SAS Logon Manager performance is slow. User logon requests are delayed.

These symptoms can arise when a significant number of concurrent requests are made to the identities service, which queries the LDAP server for users and groups. Requests to the identities service can fill queues if connections are not available in the connection pool. Eventually, logon requests might time out, depending on the configured wait time.

Tuning of the LDAP connection pool might be needed after adjustments have been made to other components, especially the [PostgreSQL data server](#) or the [JDBC connection pool](#). For example, if you decide to increase the size of the JDBC pool because of a high volume of concurrent users or batch jobs, consider whether the LDAP connection pool should also be increased.

---

## Recommendations

To modify LDAP connection pool settings for your SAS Viya platform deployment, take the following steps:

- 1 In SAS Environment Manager, edit the **identities service**.
- 2 Navigate to the **sas.identities.providers.ldap.connection** configuration instance and modify the following settings:

*Table 2 Properties and Values*

Property	Recommended Value
pool.maxActive	30
pool.maxIdle	30

Testing with higher values is important. The value of 30 resulted in improved response times in SAS Viya platform testing. However, larger deployments with a higher number of concurrent users might need to increase the pool size even more.

Another option is to change the behavior of the identities service when the connection pool is inadequate for current usage. The **pool.whenExhaustedAction** determines how the identities service responds when the connection pool has been exhausted. Valid values are as follows:

- FAIL (0) — Throws an exception (No Such Element) when the pool is exhausted.

---

**Note:** This setting is typically the default so that it is easier to determine when to adjust the size of the LDAP connection pool.

---

- BLOCK (1) — Waits until a new object is available. If maxWait is positive, a No Such Element exception is thrown if no new object is available after the maxWait time expires.
- GROW (2) — Creates and returns a new connection object. This setting negates any effects of setting pool.maxActive.

If you decide to use a value of 2 for pool.whenExhaustedAction and neglect to set a value for pool.maxActive, your LDAP connection pool is unbounded.

For more information about tuning the LDAP provider settings for the identities service, see “[sas.identities.providers.ldap.connection](#)” in *SAS Viya Platform: Identity Management*.

The Spring LDAP documentation also contains helpful tips: <https://docs.spring.io/spring-ldap/docs/1.3.2.RELEASE/reference/html/pooling.html>.

---

## Tuning the JDBC Connection Pool

Java Database Connectivity (JDBC) connection pooling creates a memory cache of database connections. This connection pool is a collection of database connection objects that are available. Database connections can then be reused so that a new connection is not created for every request. The collection is maintained by a connection pooling module as a layer on top of the JDBC driver.

When SAS Viya platform users send large numbrs of batch jobs for processing within a small time frame, tuning JDBC connection pooling is required in order to accommodate the load.

---

## Deployment Size Definitions

The suggested configurations in this section are based on the following deployment size definitions:


- A small deployment receives more than one concurrent query or update, and creates hundreds of database records daily.
- A medium-sized deployment receives several concurrent queries or updates (most of which use indexes), and creates thousands of database records per hour.
- A large deployment receives many concurrent queries or updates, creates thousands of database records per minute, receives many queries to scan tables, and processes complex queries and regular bulk loads.

---

## Configure Deployment Size

The SAS Viya platform provides default JDBC connection pool settings for small, medium, and large deployments. By default, all services are configured to use the medium deployment settings.

To configure a different deployment size for a service, complete the following tasks:

- 1 In SAS Environment Manager, click **Configuration** and select the **Definitions** view.
- 2 In the **Definitions** list, select **jvm**.
- 3 Verify that the jvm service does not have an existing configuration. If nothing has been configured yet, a message states that `The selected item has no configurations`.
- 4 If the jvm service has at least one configuration already, select the SAS Viya platform service whose jvm settings you want to modify by clicking  next to that service.  
If the service has no existing configuration, click **New Configuration**.
- 5 Click **+Add Property**.
- 6 In the **Name** field, specify `java_option_springdatasource_default`.
- 7 In the **Value** field, specify `-Dsas.deployment.springdatasource.defaults=size`, where **size** is small, medium, or large.
- 8 Click **Save** in the Add Property window.
- 9 Click **Save** to save the configuration change.


---

## Tuning Data Source Properties

By default, SAS Environment Manager runs with predefined property settings for small, medium, and large deployments. You can override these values. For example, you can set the Preferences Service to use the default property values for a small system, but override the default value of the `spring.datasource.tomcat.maxIdle` property by changing it from 2 to 3.

## Override Default Property Values

To change the default property settings, complete the following tasks:

- 1 In SAS Environment Manager, click **Configuration** and select the **Definitions** view.
- 2 In the **Definitions** list, select **spring**.
- 3 Verify that the spring service does not have an existing configuration. If nothing has been configured yet, a message states that `The selected item has no configurations`.
- 4 If the spring service has a configuration already, select the SAS Viya platform service whose spring settings you want to modify by clicking  next to that service.  
If the spring service has no existing configuration, click **New Configuration**.

---

**Note:** Not all services use the default property settings. Instead, those services specify a scaling factor that enables them to have larger pool sizes, based on the deployment size that is specified in the `sas.deployment.springdatasource.defaults` property. For more information, see [“Tuning the Data Source Scaling Factor” on page 19](#).

---

- 5 Click **+Add Property**.
- 6 In the **Name** field, specify a property from the [Property Settings Table on page 19](#).
- 7 In the **Value** field, specify the new size that you want to set for the property.
- 8 Click **Save** in the Add Property window.
- 9 Click **Save** to save the configuration setting.

## Default Data Source Property Settings

The following table provides the default property settings for the JDBC connection pool, based on the deployment size:

**Table 3** *Property Settings Table*

Property	Small Deployment	Medium Deployment	Large Deployment
<code>datasource.tomcat.initialSize</code>	2	2	2
<code>datasource.tomcat.maxActive</code>	6	10	20
<code>datasource.tomcat.maxIdle</code>	2	2	2
<code>datasource.tomcat.minIdle</code>	2	2	2

A service can also provide default files for small, medium, and large deployments in the service resource directory.

---

## Tuning the Data Source Scaling Factor

Not all SAS Viya platform services use the default property settings. Instead, those services specify a scaling factor that enables them to have larger pool sizes, based on the deployment size that is specified in the `sas.deployment.springdatasource.defaults` property. For example, the authorization service specifies a scaling factor of 10. Therefore, its `maxActive` value is 60 for small, 100 for medium, and 200 for large deployments. For a list of the default property values, see [Table 3 on page 19](#).

## Configure the Scaling Factor

To define a scaling factor for a service, complete the following tasks:

- 1 In SAS Environment Manager, click **Configuration** and select the **Definitions** view.
- 2 In the **Definitions** list, select **jvm**.
- 3 Follow the steps that are described above in [“Configure Deployment Size” on page 18](#) to add a new configuration or to modify a service definition to add a configuration property.
- 4 In the **Name** field, specify the `java_option_datasource_factor` property.
- 5 In the **Value** field, specify `-Dsas.datasource.custom.factor=multiplier`, where *multiplier* is the multiplier factor by which the property in the [Property Settings Table on page 19](#) is multiplied.
- 6 Click **Save** in the Add Property window.
- 7 Click **Save** to save the configuration change.
- 8 Restart SAS Viya platform services.

## Scaling Factor Example

You can specify a multiplier factor for a service by using the `sas.datasource.custom.factor` property. The default value for a property is multiplied by the value that you specify. For example, for a medium deployment, the default value for the `spring.datasource.tomcat.maxActive` property is 10. If you set the multiplier factor to 5, the new `maxActive` value is 50. The factor must be greater than 0. The resulting `maxActive` value will be no less than `spring.datasource.tomcat.initialsize` and no more than 200.

---

# Tuning the PostgreSQL Server

The SAS Infrastructure Data Server component is supported by a PostgreSQL database server. SAS Infrastructure Data Server provides a transactional store that is used by the SAS Viya platform. The server is configured automatically during deployment. However, you have multiple options for optimizing its performance, which can have an outsize impact on overall SAS Viya platform performance.

---

## Considerations

To support the required SAS Infrastructure Data Server component, the SAS Viya platform requires either an internal PostgreSQL instance, which is the default option and is deployed automatically, or an external instance that you configure and maintain. If you deploy with the default option, SAS provides tools to configure and maintain the deployment for you. If you instead deploy an external PostgreSQL instance, you are responsible for configuring and maintaining it.

For the internal PostgreSQL database server, SAS uses Crunchy Data PostgreSQL. Two operators manage SAS Infrastructure Data Server:

- SAS Data Server Operator – Manages all PostgreSQL instances for the SAS Viya platform (both internal and external PostgreSQL).
- SAS Crunchy Data PostgreSQL Operator – Provisions internal PostgreSQL instances only.

This operator is not included in your deployment by default, but when you add an overlay to the `kustomization.yaml` file, it creates a `sas-crunchy-platform-postgres` or `sas-crunchy-cds-postgres` cluster with default settings. You might want to tune these default settings for improved SAS Viya platform performance.

Providing your own instance enables you to restrict the ability of SAS services to access the database when a single PostgreSQL instance houses multiple databases. A README file in your deployment assets, `$deploy/sas-bases/examples/postgres/README.md` (Markdown format) or `$deploy/sas-bases/docs/configure_postgresql.htm` (for HTML format), documents the required steps to configure an external PostgreSQL instance.

Your deployment assets include example transformers to assist you in tuning PostgreSQL settings. The files and an explanatory README are located in `$deploy/sas-bases/examples/crunchydata/tuning`.

---

## Tuning Memory and CPU Resources

Crunchy Data offers the following rough formula for determining memory resource levels for the PostgreSQL container:

```
(work_mem * avg_active_session) + (max_connections * 14mb) + shared_buffer =
avgMemory
```

However, usage spikes might trigger the Out-Of-Memory killer in the operating system. In order to reduce the risk of launching the OOM killer, Crunchy Data recommends that the `avgMemory` not exceed 70% of the actual pod resource limit for memory. For more information, see <https://blog.crunchydata.com/blog/deep-postgresql-thoughts-the-linux-assassin>.

For CPU resource limits, you can tune the CPU requests value, which represents a minimum number of CPU cores that are allocated to a pod. Crunchy Data provides the following tuning formulas that might help you determine a minimum value for CPU requests:

Minimum CPU = `avg_concurrent_sessions / 2`

---

**Note:** Multiply CPU by 1000 to get the value in millicores.

---

Minimum CPU = `avg_concurrent_sessions / 2.5`

This formula suggests setting a higher limit, with the trade-off that it uses more resources but is less likely to be throttled. If latency or disk contention limit performance, consider adding additional PostgreSQL instances or applying application pooling to `pgbouncer` or `pgpool`.

SAS provides example files to help you modify these settings. For more information, see the README file: `$deploy/sas-bases/examples/crunchydata/pod-resources/README.md` (for Markdown format) or `$deploy/sas-bases/docs/configuration_settings_for_postgresql_pod_resources.htm` (for HTML format).

# Recommendations

## Relevant Tuning Parameters

Specialized solutions or use cases might require further configuration tuning. If you need to experiment with parameters in order to optimize system performance, the most important PostgreSQL tuning parameters are as follows:

### `shared_buffers`

Specifies the amount of memory to be used for caching data. PostgreSQL also benefits from the file system cache, so `shared_buffers` should not be so large that they interfere with the file system cache. For a large database, set this parameter to a value ranging from 1 GB up to 25% of the total system memory.

### `work_mem`

Specifies the amount of memory to be used for sorts, hashing, and materialization, before writing to temporary disk files. Several running sessions can perform operations concurrently. Therefore, the total memory that is used might be many times the value of `work_mem`. Keep this in mind when selecting a value for this parameter. Set this parameter to a value between 16 MB and 64 MB or more, for a specialized use case (for example, frequent, very large sorts).

### `max_connections`

Specifies the maximum number of database user connections that can be open simultaneously. The internal PostgreSQL instance is set to 1280 connections by default. An external PostgreSQL server should support `max_connections` of at least 1024. SAS recommends setting this value to be the same as the value for `max_prepared_transactions`.

Microsoft provides guidance for setting the maximum connections on Azure Database for PostgreSQL in [Limits in Azure Database for PostgreSQL – Single Server](#).

### `max_prepared_transactions`

Specifies the maximum number of transactions with a status of `prepared` that can exist simultaneously. The default value is 0, which means that no transactions are "prepared." SAS recommends setting `max_prepared_transactions` to the same value as `max_connections`.

### `maintenance_work_mem`

Specifies the maximum amount of memory to be used for vacuuming (reclaiming storage that is used by rows that are marked for deletion) and index builds. For a large database, set this parameter to 256 MB or more.

### `temp_buffers`

A database session allocates temporary buffers as needed, up to the limit that is specified by this parameter. These per-session buffers facilitate access to temporary tables. This setting can be changed only within individual sessions, and only before the first use of temporary tables within those sessions.

You can set the `synchronous_commit` parameter to **Off** for faster updates, but if an outage occurs, transactions might be lost.

See the README file at `$deploy/sas-bases/examples/crunchydata/tuning` (for Markdown format) or at `$deploy/sas-bases/docs/configuration_settings_for_postgresql_database_tuning.htm` (for HTML format) for instructions about how to set or change those parameters.



## Tune Wait Times

The PostgreSQL server from Crunchy Data offers a Patroni option that can be tuned to improve performance. The Patroni option is used by the SAS Crunchy Data PostgreSQL Operator to manage the performance of the `sas-crunchy-platform-postgres` or `sas-crunchy-cds-postgres` cluster.

Tuning Patroni settings is recommended in cases where the Kubernetes cluster experiences slow response times, which can occur due to resource contention or network latency. The Patroni option attempts to address performance degradation on the cluster side by increasing the wait times.

See the README file in `$deploy/sas-bases/examples/crunchydata/tuning` (Markdown) or see `$deploy/sas-bases/docs/configuration_settings_for_postgresql_database_tuning.htm` (HTML) for instructions. For more information about Patroni settings, see [Crunchy Data Patroni Configuration Settings](#).

## Tuning SAS Backup for PostgreSQL Database Backup Operations

If your PostgreSQL database is large, backup operations can take hours to complete. By default, compression is applied by the PostgreSQL utility as the backup proceeds for the PostgreSQL data source. The addition of compression means that the resulting size of the backup data is greatly reduced, but it slows the process significantly. You have the option to configure the backup job to disable compression for the PostgreSQL data source, to run the PostgreSQL dump in parallel by dumping a specified number of tables simultaneously, or both.

---

**Note:** The parallel jobs option might reduce the time that is required to perform the dump, but it also increases the load on the database server.

---

To reduce the overall time that is required in order to complete the backup operation, add options to the backup command using the `sas-backup-job-parameters` configMap. For more information, see the README file that is located at `$deploy/sas-bases/examples/crunchydata/backups/README.md` (for Markdown format) or `$deploy/sas-bases/docs/configuration_settings_for_postgresql_backups.htm` (for HTML).

Here are examples of configMap options that might improve the performance of the backup job. To run the PostgreSQL (`pg_dump`) backup command without the compression that is applied to the backup job by default, apply this YAML:

```
configMapGenerator:
- name: sas-backup-job-parameters
  behavior: merge
  literals:
  - SAS_DATA_SERVER_BACKUP_ADDITIONAL_OPTIONS=-Z0
```

To run the PostgreSQL (`pg_dump`) backup command with the parallel jobs option (in this example, while dumping three tables simultaneously), apply this YAML:

```
configMapGenerator:
- name: sas-backup-job-parameters
  behavior: merge
  literals:
  - SAS_DATA_SERVER_BACKUP_ADDITIONAL_OPTIONS=-j3
```

To run the PostgreSQL (pg\_dump) backup command with both options (that is, without compression and with the parallel jobs option), apply this YAML:

```
configMapGenerator:
- name: sas-backup-job-parameters
  behavior: merge
  literals:
  - SAS_DATA_SERVER_BACKUP_ADDITIONAL_OPTIONS=-Z0,-j3
```

SAS has tested with both options. The uncompressed files were almost three times larger than the compressed file that is created by default. Although the time to complete the backup operation depended on the size of the data that was backed up, the jobs completed approximately 70% faster when 8 threads were specified for the parallel jobs option.

If you decide to disable compression for backup jobs, consider also allocating additional disk space for the files that are created.

## Tuning SAS Backup for PostgreSQL Database Restore Operations

If your PostgreSQL database is large, restoring it from a backup package can take hours to complete. The time that is required to restore the database from backup is reduced when you run parallel jobs to restore the database objects. The optimal value for this option depends on the underlying hardware of the server and of the client, and it also depends on the network. For example, the number of CPU cores plays an important role. Refer to the following document for more information about parallel restore jobs using the pg\_restore utility: <https://www.postgresql.org/docs/12/app-pgrestore.html>.

To reduce the overall time that is required in order to complete the restore operation, add options to restore command using the `sas-restore-job-parameters` configMap. For more information, see the README file that is located at `$deploy/sas-bases/examples/restore/postgresql/README.md` (for Markdown format) or `$deploy/sas-bases/docs/uncommon_restore_customizations.htm` (for HTML).

Here are examples of configMap options that might improve the performance of the restore job. To run the pg\_restore command with the parallel jobs option (in this example, running eight jobs simultaneously), apply this YAML:

```
configMapGenerator:
- name: sas-restore-job-parameters
  behavior: merge
  literals:
  - SAS_DATA_SERVER_RESTORE_PARALLEL_JOB_COUNT=8
```

## Tuning the Crunchy Data PostgreSQL pgBackRest Utility

Crunchy Data provides a backup and restore utility that operates independently from the SAS Viya platform backup service. The pgBackRest utility takes a binary backup of the Crunchy Data PostgreSQL server that provides an internal PostgreSQL instance. You can modify the default pgBackRest backup schedule for full or incremental backups. You can modify the retention policy for those backups and also change the archive policy for the PostgreSQL WAL (Write-Ahead Log) data.

For more information about the Crunchy Data pgBackRest utility, see the [pgBackRest User Guide](#) and the [pgBackRest Command Reference](#).

## Multi-Tenancy Considerations

In a multi-tenant environment, the PostgreSQL server is very likely to require tuning. Before you change any parameters to try to improve the performance of SAS Viya platform components, take a few planning steps. First, attempt to determine the maximum number of connections that your database server is likely to receive simultaneously under typical operating conditions.

The database configuration that you select for your tenants has an effect on tuning requirements. The SAS Viya platform supports using one database for all tenants, separated by schemas, or a separate database for each tenant. Because the choice to use a single database for all tenants results in an increased workload for that PostgreSQL instance, you should consider increasing the resource limits for that configuration.

- a separate PostgreSQL database for each tenant

Each tenant has a unique database connection pool, which might significantly increase the total connection count that the back-end database server must support. SAS recommends tuning the `max_connections` and `max_prepared_transactions` parameters. In a multi-tenant environment, SAS recommends setting them to the same value.

- a shared PostgreSQL database that is partitioned to provide tenant isolation

This option is recommended when database connection resources are limited. A single connection pool is used for all tenants. Because connections for all tenants come from a single connection pool, a single tenant can consume all connection resources, depriving other tenants.

If you select the database-per-tenant option, SAS recommends adjusting CPU and memory resource limits for the PostgreSQL pods. An example file has been provided in your `$deploy/sas-bases/examples/crunchydata/pod-resources` directory. SAS recommends tuning both the minimum (request) size and the maximum (limit) size for CPU and memory. You can make separate resource adjustments for the pods that are deployed for PostgreSQL backup and restore operations.

For either option (separate databases per tenant or separate schemas per tenant), you should configure additional connections on the database server. Spikes in connection usage have been observed during tenant onboarding and when users log in and start using SAS products. The baseline recommendation for the SAS Viya platform is to set `max_connections` to a minimum of 1024 for an external data server. The internal data server is set to 1280 max connections by default.

For an external or internal PostgreSQL data server, you can use the following baseline formula to size your environment and tune the settings to improve performance:

$$(\text{number of tenants} + 1) * 1128 = \text{max\_connections}$$

For example, if you plan for three tenants,  $(3 \text{ tenants} + 1 \text{ provider}) * 1128 = 4512 \text{ max\_connections}$ .

In order to prepare for peak usage during tenant onboarding and SAS Viya platform usage, temporarily allocate additional connections:

- 20% more connections for the database-per-tenant option
- 25% more connections for the schema-per-tenant option

During tenant onboarding, the number of connection requests spikes to a peak before declining to a stable value. In a test of the schema-per-tenant option that onboarded 7 tenants, the spike was 1787 connections and the stable value was 1312, approximately a 26% spike during the peak. With the database-per-tenant option selected, the spike was 1881 connections and the stable value was 1528 (approximately a 20% spike), but only 3 tenants were onboarded. Consistently in SAS testing, the database-per-tenant option used more connections, with a higher spike and a higher stable value—even with fewer tenants.

For either database option, smaller spikes occurred again during system usage. The appropriate value for this setting is also partially dependent on the SAS offerings that you have purchased. If you plan to deploy multiple SAS offerings, plan for additional connection resources.

If your deployment of the SAS Viya platform includes an internal PostgreSQL database instance, a README file provides instructions for setting or changing connection parameters. It is available at `$deploy/sas-bases/examples/crunchydata/tuning` (for Markdown format) or at `$deploy/sas-bases/docs/configuration_settings_for_postgresql_database_tuning.htm` (for HTML format). For deployments with an external PostgreSQL instance, follow the instructions that apply to the PostgreSQL distribution that you have configured.

---

## Tuning SAS Visual Analytics

SAS Visual Analytics offers multiple opportunities to modify settings in order to improve report performance. The SAS Visual Analytics user interface includes Performance settings that individual users can tune if reports are loading more slowly than expected.

---

### Modifying Performance Settings

To modify SAS Visual Analytics Performance settings:

- 1 In the application bar, click the user name button, and then click **Settings**.
- 2 Expand the **SAS Visual Analytics** item in the side menu, and then select **Interface, Report Defaults, Performance**.
- 3 Update the settings that you want to change.

**TIP** When you click **Reset**, the settings revert to their original values.

Here are some performance-related settings that users might want to change:

#### **Disable categorical distinct counts for the Data pane**

specifies that you do not want distinct count (cardinality) values to be displayed beside data items in the **Data** pane. If you enable this setting after the distinct counts are already displayed in the **Data** pane, SAS Visual Analytics will not retrieve the values in the future. If you close and reopen the report, the distinct counts are not displayed.

Enabling the **Disable categorical distinct counts for the Data pane** setting can improve performance. However, the **Suggestions** pane cannot be displayed if this setting is enabled.

#### **Disable automatic correlation detection for the Data pane**

specifies that you do not want correlation values to be displayed in the **Data** pane. If you enable this setting, you cannot select measures that are correlated with a selected measure.

Enabling the **Disable automatic correlation detection for the Data pane** setting can improve performance. However, the **Suggestions** pane cannot be displayed if this setting is enabled.

**Disable automatic outlier detection for the Data pane**

specifies that you do not want possible outlier values to be displayed in the **Data** pane.

Enabling the **Disable automatic outlier detection for the Data pane** setting can improve report performance.

**Insight mode (editing)**

specifies whether SAS Visual Analytics automatically checks the current report for insights while you are editing a report. Select **Auto** to check automatically, or select **Manual** to check only when you click the ⓘ icon in the report toolbar.

**Insight mode (viewing)**

specifies whether SAS Visual Analytics automatically checks the current report for insights while you are viewing a report. Select **Auto** to check automatically, or select **Manual** to check only when you click the ⓘ icon on the report toolbar.

- 4 Click **Close** to apply your changes.

---

## Avoiding Out-of-Memory Issues

Supporting a large number of concurrent users can cause the sas-cas-control service to throw an out-of-memory (OOM) error. User concurrency in SAS Visual Analytics is quite common, for example, because the offering supports many other SAS applications. If multiple users access SAS Visual Analytics reports around the same time, you might see a Linux cgroup OOM issue when the pod limit on the machine is exceeded. Each SAS Visual Analytics user can create 6 CAS sessions that are open simultaneously. In some situations, between 300 and 400 concurrent CAS sessions can be open on the same machine, which can cause the cgroup OOM error.

In addition to user concurrency, pod OOM errors can be triggered by the following factors:

- VM instance memory and VM I/O caps
- Values of VM settings: vm.dirty\_ratio, vm.dirty\_background\_ratio, and vm.min\_free\_kbytes
- I/O device speed
- Amount of data that is written concurrently
- The pod memory limit

A Linux cgroup can enforce a physical memory limit for all the session processes that are started on a CAS server. The cgroup is used to limit the physical memory use. It enables administrators to manage physical memory utilization instead of virtual memory. This is important because physical memory is the scarce resource and the subject of management and tuning. Because most tables are memory mapped from the CAS disk cache, the server does not use any system swap space. CAS configures the cgroup to prevent use of swap space to avoid poor performance.

To avoid having CAS sessions close unexpectedly when the pod limit is exceeded, be sure to include the sas-cas-control pod in your monitoring configuration. Setting an alert that fires when the memory consumption for that pod exceeds 80% of its memory limit (which is 2500 MiB by default) should help detect and prevent OOM problems with user concurrency.

---

# Tuning SAS Viya with SingleStore

SAS Viya with SingleStore provides a highly scalable distributed relational database that powers a platform for AI, analytics, and data management. The integration results in reduced data movement and seamless access to SingleStore features.

Resource provisioning must take into account the SingleStore architecture and the extent to which CAS utilization is likely to affect SingleStore performance. For the basic system requirements, see [“Requirements for SAS® Viya® with SingleStore” in \*System Requirements for the SAS Viya Platform\*](#).

---

## Resource Considerations

SingleStore partitions act as independent databases. As a best practice, provide uniform resources for each partition. The number of vCPUs in the SingleStore cluster leaf pods should be evenly divisible by the number of partitions, and the number of partitions should be evenly divisible by the number of leaf pods.

When selecting virtual machine types, consider that SingleStore recommends uniform resourcing and at least 30 MB of RAM per physical CPU core or vCPU. In Microsoft Azure, for example, the VMs within the Es-series can have different types of processor, or processors that span multiple generations, and thus they do not conform to the uniform-resourcing requirement.

When provisioning CPU and memory resources, take the following factors into consideration:

- CPU resources are required to support connections from CAS to SingleStore CAs. SAS recommends:
  - having at least 1 CA vCPU to every 2 CAS worker vCPUs.
  - limiting CAS worker vCPUs to no more than 3 per SingleStore leaf pod vCPU.
- RAM resources are required to enable SingleStore leaf nodes to support table metadata in memory. SAS recommends:
  - having at least 8 GiB of RAM per vCPU.

---

**Note:** Memory is also needed for uncompressing of columnstore data from storage to satisfy queries and to support the SAS Embedded Process.

---

- allocating 30 MB/s of I/O bandwidth and 300 IOPS per vCPU in each leaf pod.

SAS Viya with SingleStore resource requirements also depend on the following factors in your environment:

- the expected number of simultaneous CAS user sessions.
- the type of CAS workload. Solutions can start multiple CAS sessions per user.
- the number of CAS workers (with MPP CAS).
- the number of SingleStore child aggregators (CAs).
- the number of partitions per leaf node.

SAS recommends having multiple partitions per leaf node so that you can more easily add resources if necessary.

- the number of vCPUs per partition.

Either 4 vCPUs or 8 vCPUs per partition is recommended.

- the SingleStore HA policy that you have applied.
- VNet configuration.

SAS Viya platform and SingleStore VMs should be running within a single availability zone or proximity group in order to minimize latency. Be aware that stopping and restarting the deployment might cause the movement of some nodes into different proximity groups or availability zones.

- the compression ratio of columnstore tables.
- database or database table options.

The SingleStore Support Help Center includes guidance about provisioning your environment for the expected number of database partitions. For more information, see [Number of database partitions required](#).

For more SingleStore-specific guidance, see the "System Requirements and Recommendations" section of the SingleStore [documentation website](#).

---

## Adjusting Resource Allocations

Resource allocation for SAS Viya with SingleStore pods can be a complex undertaking. Multiple factors in the environment can have an effect. For example, resource requests that are specified in pod templates might be equal to or higher than the allocatable resources of any available nodes. When all nodes lack sufficient allocatable resources to meet a pod's resource request, the pod is stuck in a "Pending" state because Kubernetes cannot find a node on which to schedule it.

SingleStore uses a unique `--overpack-factor` option to ensure that SingleStore pods can be scheduled on Kubernetes nodes despite resource contention. The overpack value specifies a fraction of a node's resource limits to subtract from pod resource requests. You can use it to adjust CPU and RAM request settings for SingleStore containers. Ideally, the overpack factor (a positive number between 0 and 1) should be as close to 0 as possible.

However, it cannot always be 0. An overpack factor of 0 can result in situations like the following example: a SingleStore container request is for 8 vCPUs. As a result, that container cannot be scheduled on a VM with 8 vCPUs. The allocatable vCPU value for a VM is always slightly lower than the VM capacity.

In addition, other containers that are running on the VM consume some of the allocatable vCPU space with their own requests. The role that each node plays in the SingleStore cluster can therefore affect resource requirements. Each SingleStore leaf pod has a second container that is named *exporter*. Requests for vCPU and memory from exporter containers also count toward the total CPU and memory that your SingleStore VMs require.

In typical SAS Viya platform usage, CPU and memory requests are set for each container in PodTemplates, Deployments, StatefulSets, and the CASDeployment custom resource. Setting resource requests for SingleStore requires you to consider additional parameters. The SingleStore container's request is determined dynamically at run time by the SingleStore operator. The request value is a product of `limit * overpack factor`, where `overpack factor` is a parameter that is defined in the SingleStore operator. The SingleStore container's limit is also determined at run time by the operator as a product of `height * cpu-per-unit` or `height * memory-per-unit`. The `cpu-per-`



`unit` and `memory-per-unit` parameters are defined in the operator. However, `leaf` and `aggregator height` are parameters in the `memsqlcluster` custom resource.

Adjusting the overpack factor can automatically reduce container requests, preventing scheduling errors. However, the setting for `--overpack-factor` affects the value for `default_partitions_per_leaf`. As a result, you might have to modify that setting. For a full discussion of the SingleStore overpack factor and examples of how to adjust it, see [“Modify the Overpack Factor”](#) in *SAS Viya Platform with SingleStore: Administration and Configuration Guide*.