

Paper SAS3322-2019
Multi-tenancy in SAS® Viya®: Considerations and Implementation

Eric Davis, SAS Institute Inc.

ABSTRACT

This paper introduces SAS® Viya® 3.4 multi-tenancy and considerations for its deployment. It delves into the steps required to deploy and configure multi-tenancy successfully. It highlights the reasons to use multi-tenancy, changes to make during deployment, how to onboard tenants, and pitfalls to avoid during the configuration.

INTRODUCTION

A multi-tenant deployment of SAS Viya allows for a single deployment to serve multiple customers. These customers can share some physical resources while remaining logically separated. A multi-tenant deployment allows for these distinct groups to share IT resources in a secure and cost-effective manner.

Multi-tenancy can add flexibility and depth to a SAS Viya deployment, but it also introduces some added complexity. This complexity should not be surfaced to the end users, but for administrators and IT stakeholders, this paper will highlight considerations around using multi-tenancy and communicate details and decisions that will need to be made. The paper will also provide an overview of the steps to complete when deploying and configuring a multi-tenant deployment.

CONSIDERATIONS FOR USING MULTI-TENANCY

In a SAS Viya multi-tenant deployment, many components are shared among all the tenants, and only a few components are tenant-specific. One of the best ways to visualize this is as an apartment building. The plumbing and electricity are common to all the tenants living there; as a tenant, you cannot say that you own the pipes in the walls. The apartment building would also have shared areas like a pool, gym, or laundry room that everyone uses. Every tenant in the apartment will have their personal unit that they lease, and that other tenants cannot access. This is the heart of multi-tenancy: allowing for sharing of common resources while keeping tenant-specific things like data secured to only the tenant.

WHEN TO USE MULTI-TENANCY

Multi-tenancy can be a very useful way to deploy SAS Viya. However, not every situation requires multi-tenancy. Careful consideration should be given to determining whether a multi-tenant deployment is right for you. Multi-tenancy results in architecture and administrative complexity, and depending on how you plan to deploy it, it might require more hardware. For example, if you want to segregate tenants to use separate hardware for processing, or if you require a highly available system, additional considerations and resources are required for a multi-tenant deployment as compared to a traditional SAS Viya deployment. However, a multi-tenant deployment is very beneficial in some scenarios.

Security and Segregation

One of the best use cases for multi-tenancy is for increased security and segregation. In a traditional SAS Viya deployment, segregating users, data, and reports among multiple business units requires a lot of thought and design for the authorization model. You would need to configure permissions for content folders, reports, tables, caslibs, and other objects

between business units. However, if you were to instead set up a multi-tenant deployment, you could configure each business unit as a tenant. This would allow for all users to share the common components of the platform, while inherently segregating and securing their data, reports, and other content. Using multi-tenancy in this way would have the constraint that content and data from one tenant would not be accessible by any other tenant. Thus, there would be no sharing of content across tenants.

Chargeback

Another reason you might choose to implement multi-tenancy is if you require a chargeback model for usage of an environment. For example, we have two business units: Marketing and Sales. Marketing needs a much smaller footprint than Sales. You can deploy the multi-tenant environment so that each business unit has a separate number of distinct workers on separate hardware. This type of deployment is shown in Figure 1 below.

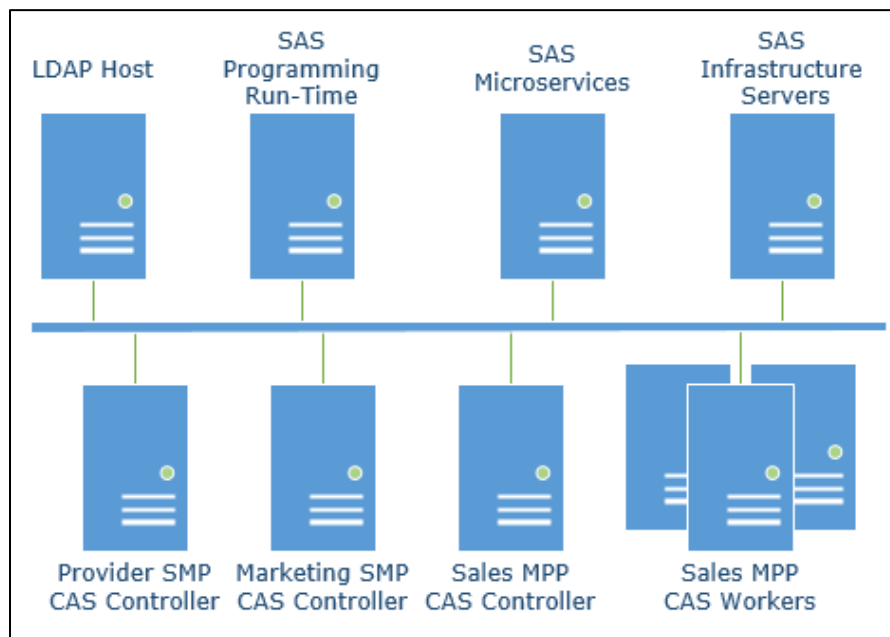


Figure 1: Multi-Tenant Deployment for Different Business Units

In this diagram, the Marketing tenant is provisioned with a single SAS® Cloud Analytic Services (CAS) controller on hardware that Marketing is paying for, along with a share of the SAS licenses. Sales is using a distributed CAS server with three CAS workers. Sales is also paying for their hardware and a greater share of the software licenses. Both Marketing and Sales are using the shared components of the deployment while having their own distinct tenant environment.

Multiple LDAP Connections

Another use case that fits well is if your environment needs to connect to separate LDAP providers. A current constraint in a traditional SAS Viya deployment enables you to connect only to a single LDAP provider for your users, groups, and memberships. However, if you require the ability to connect to multiple LDAP providers, you could set up multiple tenants and configure each tenant to connect to a different LDAP provider. A multi-tenant deployment with multiple LDAP providers would also add to the complexity of the host layer, as each host would need to be configured for the additional domains.

When Multi-tenancy Might Not Be the Right Fit

The three use cases that are described above are the most common scenarios where a multi-tenant deployment has proven business value. There are a few use cases where you might think multi-tenancy would add value, but in practice, it might not be the best fit.

In SAS Viya, you can set up security between business units without a multi-tenant deployment. You can define business groups and apply permissions between groups on caslibs and tables. You can also segregate content, folders, and reports. In addition, having a standard deployment might be the best choice if you have a lot of crossover of data and content that is shared between business units or users. For example, if you have enterprise reports that everyone can view, you can make these available to all users in a standard SAS Viya deployment. However, you would have to replicate both the reports and data for each tenant in a multi-tenant deployment. A standard SAS Viya deployment would also eliminate the administration overhead that comes with multi-tenancy.

Another potential use case that might be considered is a paradigm of development, test, and production environments, where every environment is a tenant. This sounds great in practice, as the data, reports, and content can be segregated and promoted between tenants/environments. However, there is a downside to this layered approach. A primary reason for using this tiered environment paradigm would be to test software updates and hotfixes in a lower environment before doing the same in a production environment. But because of all the shared resources, a software upgrade would not be isolated to a tenant; it would apply to all the tenants in the SAS Viya environment at the same time. However, there is value in this as a deployment scenario if the limitations are taken into consideration. A better solution might be to have development and test environments sharing a multi-tenant deployment, with production on its own dedicated infrastructure.

Some customers have asked about using multi-tenancy to create a sort of high-availability solution. The thought was that if resources for one tenant go down, users can hop over to another tenant. The simple answer is no; this is not how multi-tenancy will work. Because SAS Viya includes many shared components, all tenants are impacted if those shared components go down.

TERMINOLOGY

Before moving on to the more technical considerations of a multi-tenant deployment, below are key terms that you should know. Most of these are discussed in SAS documentation and are needed for context and understanding of multi-tenancy.

- **Tenant**—A subset of users that requires segregation of data and content from all other users in the system. These users are isolated, with no visibility into other tenants. They share many common resources across the platform, but have some dedicated resources deployed and secured to themselves. A tenant has access to all licensed software but cannot access another tenant's data or content.
- **Tenant administrator**—A user who performs SAS administrative tasks at the tenant level in the environment. They manage access to software applications and control data access within the constraints of a single tenant. They are insular to their tenant environment and cannot administer or access other tenants. Tenant administrators manage tenant-specific configurations and deal with the creation of user groups and assignment of permissions to users and groups.
- **Provider**—The first deployed environment, which is used to manage one or more tenants within a single SAS Viya deployment. The provider could be considered the first tenant, or "Tenant Zero," as this environment is created when you first deploy the software in a multi-tenant environment. The provider is set up exclusively for

administration and governance of the multi-tenant environment. It is not intended to be a stand-alone tenant, or to be used in place of a standard deployment.

- **Provider administrator**—The administrator for the provider environment, who manages SAS administration tasks for the provider. In addition, they are responsible for the tasks of onboarding and offboarding tenants, managing global configurations for all tenants, implementing the host-level configuration of individual tenants, tenant monitoring, and troubleshooting of the multi-tenant environment.
- **Onboarding**—The process of creating and configuring tenants.
- **Offboarding**—The process of decommissioning and deleting tenants.
- **SAS Cloud Analytic Services (CAS)**—A server that provides the cloud-based run-time environment for data management and analytics with SAS. It can be deployed as a stand-alone controller that handles symmetric multiprocessing (**SMP**), or it can be deployed across multiple machines for massively parallel processing (**MPP**). When deployed as MPP, it consists of a CAS controller and multiple machines assigned as CAS workers.
- The **sasboot** account—An internal user account that is created during the deployment process. It is the initial administrator of a traditional deployment, and in a multi-tenant deployment, this account is the initial administrator of only the provider tenant.
- The **sasprovider** account—A distinct internal user account that is the tenant’s initial administrator. When tenants are onboarded, they each get a sasprovider account. This internal account is how the provider administrators can access and configure the tenant before granting tenant users and tenant administrators access. Although each tenant gets an account called sasprovider, each account is set up independently when onboarding the tenant.
- The **inventory.ini** file—SAS Viya uses this file to specify the machines to be included in a deployment and the type of software to be installed on them. In this document, a name shown in brackets, such as [CoreServices], is a reference to a specific section of the inventory.ini file.

DEPLOYMENT CONSIDERATIONS

If you determine that multi-tenancy is the right fit for your organization, or if you think it might be at some point in the future, additional architecture and deployment decisions are required.

First and foremost, SAS Viya 3.4 multi-tenant deployments are supported only in the Linux operating environment. This constraint narrows both the scope of licensing and hardware when planning a multi-tenant environment.

Deployment Time or Bust

When deciding if you want to use multi-tenancy, you must make that choice at deployment time. The initial deployment is the only time that you can select the multi-tenant option. Moreover, there is no way to convert a traditional deployment to a multi-tenant deployment, either through an update or an upgrade. This fact might raise the question whether it would be a good idea to always do a multi-tenant deployment. In this scenario, you would deploy with the multi-tenancy option, which would create a provider as tenant zero. This provider would then onboard a tenant that would be leveraged by the users of the environment. If you think that you ever might want to use the multi-tenancy features, this would be the best path because it would enable you to grow your tenants later if needed. To

change to a multi-tenant deployment after the fact, you must re-deploy a new environment and migrate your content.

SAS® Infrastructure Data Server

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

When deploying a multi-tenant environment, a decision must be made regarding the configuration of the SAS Infrastructure Data Server. This is a deployment-time decision and cannot be changed later. There are two ways to deploy the data server. This first (and default) way to configure multi-tenancy uses a single database but isolates the tenants with a separate database schema per tenant within the server. This allows for the partition and separation of the tenant's content, such as authorization rules and custom groups, which is stored in the tenant's schema. The *SAS Viya 3.4 for Linux: Deployment Guide* states the following in its section titled "[SAS Viya and Multi-tenancy](#)":

This mode is useful when database connection resources are limited because this mode uses a single connection pool for all tenants. A disadvantage is that data for all tenants is secured by a single credential. In addition, connections for all tenants come from a single connection pool, which means that a single tenant can consume all connection resources and deprive other tenants of resources.

Even with the limitations listed, this is a common multi-tenant deployment path to take. However, there are caveats; some software will not work correctly when using the schema per tenant option. For example, SAS® Visual Investigator does not support a multi-tenant environment when a single SAS Infrastructure Data Server is shared by tenants. This software and other SAS solutions require a different configuration style.

The second, and less common, way to configure the SAS Infrastructure Data Server is to create an entire database per tenant. The same section of the documentation quoted previously states,

This mode is useful for users who prefer maximum isolation of tenant data. In this mode, unique database credentials per tenant enhance security and isolation. However, a disadvantage is that each tenant must have a unique database connection pool, which can significantly increase the total connection count that the back-end database server must support. Additional tuning is required for this mode.

The value with this configuration is the truly segregated nature of the tenant content. There are industries like health care, banking, and government where everything that can be separated must be separated. This option requires additional configuration, tuning, and architecture considerations for defining a database per tenant. Regardless, the extra complexity might be worth the added value of separation for some organizations.

What Components Are Tenant-Specific versus Shared?

A multi-tenant deployment has some components that are tenant-specific, such as data and content. It also includes many shared components that are leveraged by all tenants. You can categorize the SAS Viya deployed components into three categories: programming run-times, CAS, and the services layer. With multi-tenancy, the general rule is that the services layer is shared among all tenants. This layer includes all the microservices, web applications, infrastructure servers (for example, the SAS Cache Server, SAS Secret

Manager, and SAS Message Broker), and the Apache HTTP Server. The reason that this is only a general rule is that some service-layer components can be configured with one service per tenant, such as the SAS Infrastructure Data Server.

SAS Viya is designed to enable many of these shared service-layer resources to be scaled up as you see fit. You can re-run the deployment and create redundant services to improve performance or availability. This is especially useful in a multi-tenant deployment, because as you grow the number of tenants, you can also grow these shared components, and probably need to do so.

When it comes to the CAS server however, each tenant gets their own distinct CAS server instance. Each tenant's CAS server is isolated to the tenant, and there is no crossover between tenants or even to the provider. Technically, you do have the ability to deploy these CAS servers to the same hardware – as either SMP or MPP architecture. However, each tenant has its own CAS controller. Even though a single machine can host multiple CAS configurations, SAS recommends that you dedicate a single CAS controller per machine in a production environment. Even when the same hardware hosts multiple CAS instances, each tenant's CAS server is secured to the tenant.

For programming run-times, this gets more complex as there is a split between which components are shared and which are tenant-specific. Programming run-times are essentially processes where SAS code gets executed, and these sessions where code runs are owned by the user who is executing the code. For programming run-times, tenants get their own components, which include the following:

- SAS Workspace Server configuration at `/opt/sas/tenantID/config/etc/workspaceserver/default`
- SAS Object Spawner configuration `/opt/sas/tenantID/config/etc/spawner/default`
- SAS Object Spawner process called `sas-tenantID-spawner-default`
- Compute Context for launching SAS Compute Servers
- Launcher Context for launching SAS Launcher Servers

These tenant-specific configurations and contexts are used during session initialization to get tenant-specific programming run-times. These configurations allow for lockdown options, and force tenants to be independent of each other.

Host-Level Considerations

With a standard SAS Viya deployment, host-layer considerations for filesystem access and process ownership come into play. For example, when you start a compute session in SAS Viya, the user that is attempting to launch the session needs to be known on the host because the compute process is launched as the end user. These host-level considerations also extend to the CAS server, depending on the identity that is running the CAS session process. Although all these considerations are still applicable for a multi-tenant deployment, there are additional considerations when using multi-tenancy.

When deploying SAS Viya 3.4 as multi-tenant, you still need to fulfill the user and group requirements for a traditional SAS Viya deployment, including:

- user account that deploys the software
- cas user account
- sas group
- consistent UID and GID for all users and groups across the hosts

In addition, the following requirements apply to each tenant that is onboarded:

- tenant_admin – This account is the owner of all tenant-specific files and directories on disk as well as the process owner of the tenant-specific services. The tenant's CAS server process will be running as this account. It is analogous to a combination of the deployment owner and the cas account from a standard deployment. This account could be from an LDAP provider or local to the host.
- tenant_admin_group – Must be the primary group for the tenant administrator. It is used for directory and file permissions on admin-restricted, tenant-specific resources. This group is like the tenant-specific version of the sas group. This group could be from an LDAP provider or local to the host.
- tenant_users_group – The group for all non-administrator tenant users. It is used for directory and file permissions on tenant-specific resources. This differs from the tenant_admin_group, as these tenant-specific locations are intended for end-user access. It could be from an LDAP provider or local to the host.

Another requirement for a multi-tenant deployment is that all provider users must be members of the sas group. This is because in a multi-tenant deployment, access control lists (ACLs) are applied to certain directories for all hosts within the Compute Server host group and programming host group within the SAS Viya configuration directory, `/opt/sas/viya/config/`. To ensure that the provider users can access critical services (including SAS Studio and the SAS Compute Server), they must be members of the sas group. This also requires the file system mount to allow for ACLs.

LDAP Requirements and Nuances

Beginning with the SAS Viya 3.4 release, there are two configuration options in a multi-tenant deployment for binding SAS Logon Manager and the Identities microservice to LDAP: a single configuration that applies to all tenants, or a separate configuration per tenant. The single configuration for all tenants was the only multi-tenant option available with SAS Viya 3.3 and has persisted. It requires a fixed LDAP tree structure that most organizations will not have in place. This tree structure has all tenants, including the provider, share the same LDAP domain. It requires the provider and all tenants to be at the same level within the tree, with the user and group organization units (OU) below the tenant level, and with the same structure mirrored across all tenants. In addition, it is not just membership within the OU that is required, the object must also be at that point in the directory information tree (DIT) and have the tenant as a part of its distinguished name (DN). Figure 2 below shows this hierarchical structure with the provider, tenantA, and tenantB in the LDAP tree.

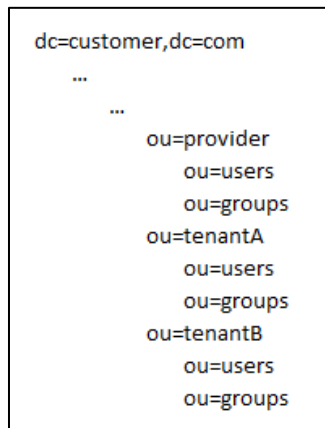


Figure 2: Required LDAP Structure for Single LDAP Configuration

LDAP configuration has lost its rigidity with the advent of SAS Viya 3.4 and its additional configuration options. You can now implement a separate LDAP configuration per tenant. This allows for much greater flexibility within your multi-tenant deployment and allows for some great usage patterns. For example, you can now configure tenants to all use the same LDAP server, but start your search for users or groups at different base distinguished names within the tree.

Another way to leverage this configuration is to change your object filter for tenant users or groups. For each tenant, you could start at the top of the LDAP tree as your base DN, and then have an object filter that brings in your tenant users and the members of your administrators group. As an example, for the provider you could use your object filter to bring in only the members of an "Administrators" LDAP group. This would let them be the only users who would be known to the provider tenant, and thus the only ones who are able to log on to the provider. You could then update the object filter for the Marketing tenant to allow for membership within "Administrators" or membership in the "Marketing" LDAP group. If you used this paradigm for all tenants, your SAS Administrators would have access to the provider and all tenants and could act as both your provider and tenant administrators.

Another use case for this new configuration option would be to connect to multiple distinct LDAP domains. Using this configuration can significantly increase the complexity of your solution. For example, if you have one set of users on your Hong Kong domain, and another set of users on your Europe domain, you could set each tenant's LDAP configuration to point to each domain. With these multiple domains there is now added complexity at the host level, as users from both domains need to be known and unique to host. Potentially, you would need to configure your hosts to connect to both domains. You would also need to make sure that the UIDs and GIDs that are used on the hosts are unique across domains. As the tenant administrators would be coming from the LDAP server that is configured for each tenant, they would need to be self-sufficient SAS administrators because the provider would (probably) not be in that domain.

When deploying multi-tenancy, there is no reason to stick to the rigid structure of a single LDAP configuration that applies to all tenants because each tenant can be configured separately. Even if you are connecting all tenants to the same LDAP provider, having a configuration per tenant exempts tenants from the strict LDAP tree requirements.

SAS also recommends that you enter the LDAP information about your multi-tenant deployment with SAS® Environment Manager after installing your software. To enable this functionality, remove the `sas.identities.providers.ldap.connection` block from your `sitedefault.yml` file. This will be covered later in the Implementation section.

Resource Considerations

Multi-tenancy requires proper planning for available resources. In general, SAS Viya requires careful host-level monitoring of resources like memory and disk space. For deployment-level monitoring, the focus is on availability of resources like the microservices or computing environment. By choosing a multi-tenant deployment, this required monitoring of resources is even more important. An example of this would be if you have multiple tenants sharing CAS workers, you need to carefully monitor how much memory is available because one tenant could monopolize the worker and impact other tenants.

In addition, each tenant that you onboard will require available resources, like CAS controllers and CAS workers. A new tenant could also necessitate additional disk space on the Compute tier. Other resources that you will want to consider scaling are the microservices and infrastructure servers. Each tenant that you onboard could place a strain on the availability of these resources. A simple example would be the SAS Message Broker.

As you onboard additional tenants, the message broker could become overwhelmed, resulting in delayed response times for end users and services. However, to mitigate this situation, you could scale out your services as you grow your environment.

Overall, the scaling of these resources is no different in a multi-tenant environment than in a standard deployment. You can add resources to the inventory file and re-run the original playbook. This is a highly useful strategy, because as you onboard tenants, additional service resources will probably be needed to support the higher load request.

Another resource to be mindful of is your license. You always want to make sure you are license-compliant as you expand your deployment and onboard tenants. Remember, each tenant that you onboard gets its own CAS server. A large component of designing a multi-tenant deployment is planning how the environment will share the license.

When deploying or adding tenants, you always want to make sure you have considered the following factors:

- Do your hosts have appropriate memory?
- Do your hosts have appropriate disk space?
- Do you need additional services, infrastructure, or computing resources?
- Are you license-compliant?

If you are implementing a deployment and want a robust, highly available platform, multi-tenancy would also increase the complexity of your platform and the resources required. For example, if you want a highly available CAS server for your tenant, you now have a primary CAS controller, a backup CAS controller, and multiple CAS workers per tenant. Multi-tenancy can be configured with high availability; it is just more complex than a standard highly available deployment. Also, if you want a disaster recovery solution, you now need to consider each tenant's section of the platform, as they each have their own separate data, configurations, and content.

Tenant Access and Zones

Tenants are accessed via tenant-specific URLs that have the following format: `tenantID.hostname/applicationName`. The following is an example tenant-specific URL for navigating to SAS Drive for a tenant named Marketing: `marketing.sas.com/SASDrive`. All tenant URLs are built off a configuration called zones. In this configuration, you set the internal hostnames that are then prepended with a tenant name to allow for access. In the [SAS Viya 3.4 for Linux: Deployment Guide section on multi-tenancy](#), this list of hostnames is described as follows:

The comma-separated list of internal host names should specify any hosts that will be used to access the provider and any domain from which tenant subdomains will potentially be built. Machines that are included in the [httpproxy] host group in the inventory.ini file should be included in this list. Machines that are included in the [CoreServices] host group in the inventory.ini file should be included in this list if there are multiple domains and they are a subset of each other. Do not use wildcard characters in this list. A host name from this list with a prepended tenant name will be used as the URL to reach each tenant after each one has been onboarded.

Moreover, the documentation goes on to note that you must add wildcard versions of the host names to your DNS using the method that your administrator recommends. For example, if you added `hostname1.company.com` and `hostname2.company.com` to the `internal.hostnames` variable in your `sitedefault.yml` file, you would add `*.hostname1.company.com` and `*.hostname2.company.com` to your DNS that points to the

host name or IP address of the HTTP server of your SAS installation. This sitedefault.yml file is discussed further in the Implementation section.

If this DNS step is missed, the URLs will not resolve, and you will not be able to onboard or access your tenants. In testing deployments, you can temporarily add these domains and subdomains to the hosts file on the machines in the deployment so that the URL will resolve correctly. The same would need to be done from any users' host machines that connect to the environment because the URL must resolve in their browsers as well. This would be only a temporary solution, and you would still need to get the DNS updated appropriately.

Administrative Responsibilities: Provider versus Tenant

With multi-tenancy, there are now two types of administrators: provider administrators and tenant administrators. They each have different responsibilities and capabilities within the deployment. They even have different options and screens available within SAS Environment Manager. Certain activities can be performed only at the provider level, and certain tenant activities can be performed only at the tenant level.

Provider-level administrators have additional capabilities, permissions, and responsibilities compared to tenant administrators. As the provider can be considered the initial or default tenant, provider administrators also handle all the provider tenant-specific administrative tasks. Table 1 below contrasts the exclusive provider-level tasks with tenant administrator tasks (the tenant-specific tasks are generic SAS Administrators tasks):

| Provider Administrator Tasks | Tenant Administrator Tasks |
|---|---|
| <ul style="list-style-type: none"> • Perform tenant management tasks (onboarding, offboarding, etc.) • View or modify deployment resources For example: <ul style="list-style-type: none"> ❖ Add or remove resources to the deployment ❖ Manage SAS Configuration Server (Consul) key-value pairs ❖ Update system configuration properties ❖ Manage system component status • Manage global configurations, including global configuration for the connection to the LDAP identity provider • Manage product licenses • Access application and server logs • Perform machine-level monitoring • Perform provider-level backup and recovery • Administer the provider as a tenant | <ul style="list-style-type: none"> • Manage tenant-specific configurations, such as the tenant-specific configuration for the LDAP identity provider • Create and restore tenant-level backups • Add, remove, or modify caslibs • Manage access to data • Manage access to content • Manage access to functionality • Promote content • Assign users to custom groups • Manage mobile devices • Schedule jobs • Set the default QKB and the default locale |

Table 1: Comparison of Administrative Tasks

The documentation gives a great explanation of how these groups work together. below is an excerpt and image from "Multi-tenancy: Concepts" in the [SAS Viya 3.4 Administration documentation](#):

An identical set of predefined groups and principals exists in the provider and in each tenant. Membership in the provider's SAS Administrators group enables you to perform provider-level tasks, but does not enable you to sign in to any tenant. Membership in a tenant's SAS Administrators group enables you to administer that tenant. The following figure depicts the structure:

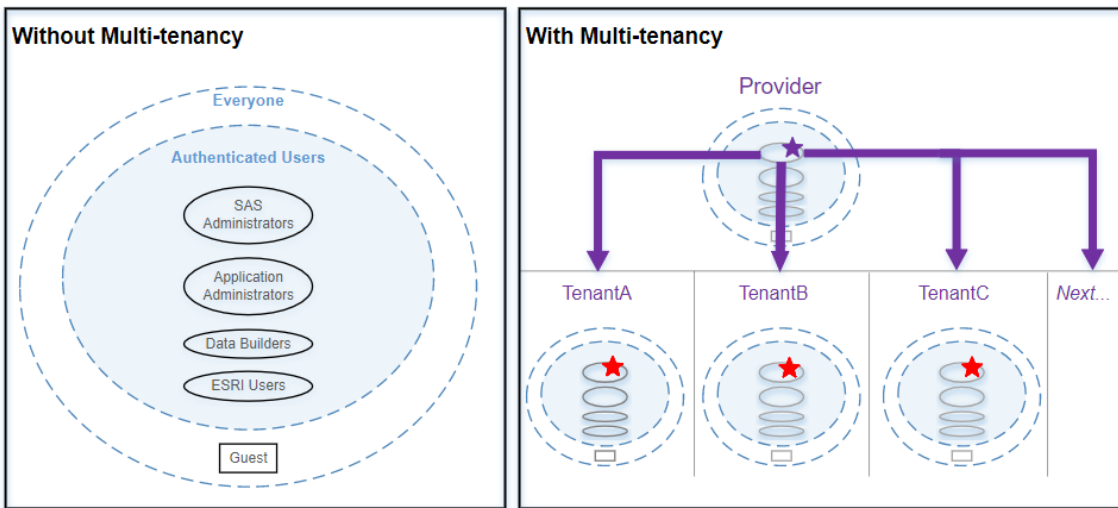


Figure 3: Predefined Groups and Principals

As shown in Figure 3, all tenants (including the provider) have the same predefined custom groups. The members of the provider's SAS Administrators custom group are the provider administrators. The provider might leverage other groups for purposes like validating or reporting. In the diagram, the provider administrator is designated with a purple star and cannot directly access the tenants; instead, provider administrators need to leverage the tenant's internal sasprovider account to access the tenant environment. These tenant-specific sasprovider accounts are designated with red stars in the diagram. While this internal account's name is the same for all tenants, each tenant gets its own version of the account, and its password is set when the tenant is being onboarded.

MULTI-TENANCY IMPLEMENTATION

SAS Viya multi-tenancy documentation is embedded as sections in various documents and creates a slightly disjointed experience when you are trying to plan and execute your deployment. The following sections outline the general steps for deploying SAS Viya in a multi-tenant fashion and describe where specific details can be found in the documentation.

MULTI-TENANT PRE-INSTALLATION TASKS

Multi-tenancy is just a specific way to deploy SAS Viya. The deployment still has the all the same pre-work that is required of a traditional SAS Viya deployment. Listed below are some specific pre-installation items that are needed for a multi-tenant deployment.

Mirror Repository

When adding new tenants to an existing SAS Viya deployment, the system downloads and installs the latest software available from the SAS software repository. Therefore, make sure

you are using a mirror repository. A mirror repository ensures that the software that you might deploy later is the same software that was installed during the original deployment. If a mirror repository is not used, then a tenant that you onboard 6 months after your original deployment could possibly end up with a different set of binaries, as the SAS repository might have been updated during that time. Another reason to use a mirror repository would be if you are adding resources to a tenant. For example, a tenant has run out of space and needs to grow their CAS server. When adding MPP CAS workers, you want to make sure you are deploying the same software as the original deployment to keep your environment in sync.

For more information, see [SAS Viya 3.4 for Linux Deployment Guide: Create a Mirror Repository](#).

Multi-Tenant Package Requirements

Multi-tenancy requires two extra packages that need to be installed before you can onboard tenants to your environment. The original deployment will succeed, but if these packages are not installed on the correct host, tenant onboarding will fail.

To support multi-tenancy, the jq-1.5 package (or a later version) must be installed on any hosts where the command line interface is installed before you begin onboarding. This package enables the system to easily process JSON structured data and is required. If the jq package is not installed on the hosts in the [CommandLine] host group from your inventory file, the onboarding playbook fails with the message "jq: command not found".

In addition, the access control list (acl) package must be installed on any hosts in the [ComputeServer] host group and the [programming] host group from your inventory file before you begin onboarding. These access control lists are used in securing some directories when a tenant is onboarded. If the acl package is not installed in the two host groups, the onboarding playbook fails with the message "setfacl: command not found".

DNS Resolution

The Ansible scripts that are used in onboarding tenants use the subdomain tenantID.hostname, and this can cause issues if the host name's subdomain cannot be accessed. Issues are typically caused by a subdomain name that is not correctly resolving. An error might be seen like the following example, shown in Figure 4, from the tenant-specific onboarding log at

/opt/sas/viya/config/var/log/multitenant/tenant/tenant_onboard.log:

```
...
INFO: Creating sasprovider in the tenant has returned a code of 1.
ERROR: sasprovider.sh returned 1.
INFO: Often this error is caused by missing DNS setup or an incorrect value
for zones.internal.hostnames in sitedefault.yml
...
```

Figure 4: Log Output from a Failed Onboarding

Before onboarding, SAS recommends pinging the tenant-specific URL that you will be using to access your tenant from each host. This will enable you to see whether the name is resolving as expected.

DEPLOYING A MULTI-TENANT ENVIRONMENT

Deploying SAS Viya with multi-tenancy is the same as a traditional SAS Viya deployment, with a few extra steps as noted in the following sections.

SAS® Studio 4 Configuration Addition in vars.yml

SAS Studio 4 is a programming client that uses the Object Spawner and Workspace Server paradigm to launch a programming run-time. The UI of SAS Studio 4 provides a way to navigate the filesystem. This is problematic for a multi-tenant environment, as there is potential for the tenant to navigate outside of the tenant-specific area (for example `/tmp`) and to write or read something that should be tenant-specific. Therefore, before running the Ansible playbook to deploy SAS Viya, make a small change in the vars.yml file by adding the following highlighted line in the figure below under the STUDIO_CONFIGURATION block:

```
STUDIO_CONFIGURATION:
  init:
    #sasstudio.appserver.port_comment: '# Port that Studio is listening on'
    #sasstudio.appserver.port: 7080
    #sasstudio.appserver.https.port: 7443
    #webdms.workspaceServer.hostName: localhost
    #webdms.workspaceServer.port: 8591
    webdms.showSystemRoot: false
```

Figure 5: Update to vars.yml

Information about this configuration update comes from the “Enable Multi-tenancy” section of the [SAS Viya 3.4 for Linux: Deployment Guide](#). By setting this parameter to *false*, you are limiting what is seen in the Folders tree within SAS Studio 4. This changes the Files navigation to start at the user’s home directory (`$HOME`) instead of at the system root (`/`). Additional information about this setting can be found at [SAS Viya 3.4 Administration: Configuration Properties: How to Configure SAS Studio 4](#). This is the only addition needed for the vars.yml file.

Creating the sitedefault.yml File

Modifications in the sitedefault.yml file are the main drivers of multi-tenancy. This file includes configuration information to enable multi-tenancy, configure SAS Infrastructure Data Server, and identify zones declared for the tenants. This file, in YAML syntax, is used by the Ansible playbook to drive the deployment. Special care needs to be taken when editing this file, as spacing, indenting, and spelling are extremely important. After you set a value with sitedefault.yml, you cannot re-run the playbook with the updated sitedefault.yml to change the value. You will have to use other tools, such as SAS Environment Manager or the sas-bootstrap-config CLI, to make post-deployment changes to the values that are set in this file. SAS recommends starting with a copy of the sitedefault_sample.yml file and naming the copy sitedefault.yml. The file sitedefault_sample.yml is in the directory where the playbook was uncompressed.

This was mentioned previously but bears repeating: SAS recommends that you manually enter the LDAP information about your multi-tenant deployment within SAS Environment Manger. This would mean not including any variables from the `sas.identities.providers.ldap.connection` block in the sitedefault.yml file.

Select a Multi-Tenant Deployment

This is one of the easiest modifications to `sitedefault.yml`. Simply add the line highlighted in the image below:

```
config:
  application:
    sas.multi.tenancy.enabled: true
```

Figure 6: Multi-Tenant Deployment Settings

This property simply indicates that you want a multi-tenant deployment. It is an option that can be set only at deployment time. There is no way to convert a traditional deployment to a multi-tenant deployment, either through an update or an upgrade. If this option is not set, the deployment will be a standard deployment.

SAS Infrastructure Data Server

Next, select the configuration for SAS Infrastructure Data Server. The two options discussed previously were a single database, where each tenant gets a schema, or a unique database per tenant. The first option is the default, with a schema per tenant in a single database. If you do not add a configuration value, this option is selected for deployment.

However, if you do want a deployment with a database per tenant, you can set `sas.multi.tenancy.db.mode: databasePerTenant`, as shown in the image below. Some software solutions, such as SAS Visual Investigator, require you to deploy the databases in this manner. If you specify `databasePerTenant`, you must add additional parameters for the connection pool of the databases. The applicable parameters are highlighted in Figure 7:

```
config:
  application:
    sas.multi.tenancy.enabled: true
    sas.multi.tenancy.db.mode: databasePerTenant
  spring:
    datasource.initialSize: ${spring.datasource.tomcat.initialSize}
    datasource.maxIdle: ${spring.datasource.tomcat.maxIdle}
    datasource.maxActive: ${spring.datasource.tomcat.maxActive}
    datasource.minEvictableIdleTimeMillis: ${spring.datasource.tomcat.minEvictableTimeMillis}
    datasource.minIdle: ${spring.datasource.tomcat.minIdle}
    datasource.tomcat.initialSize: 0
    datasource.tomcat.maxIdle: 0
    datasource.tomcat.minEvictableTimeMillis: 30000
    datasource.tomcat.minIdle: 0
    datasource.tomcat.maxActive: 100
```

Figure 7: Parameters Needed for a Database per Tenant

The parameters for a database per tenant need to be tuned to your environment and usage. These parameters can be adjusted later if necessary. However, the choice of database mode is final and can be set only at deployment time. If you want to change how you configured SAS Infrastructure Data Server, then you must uninstall and redeploy with the other option selected. This would force you to migrate content and configurations that you wish to preserve.

Zones

The next addition to the `sitedefault.yml` file is a property called `zones`. This property has only one value that you need to provide for multi-tenancy: `internal.hostnames`. This is a comma-separated list of internal host names that are used to access the provider zone or

are used in a subdomain to access other zones. This is also the field that is used to generate the subdomain access. For example, if the Apache HTTP server is on a host called `server.customer.com`, and if you are planning for tenants named Sales and Marketing, you would set this value as `internal.hostnames: server.customer.com`. This (combined with the DNS updates) would allow for your tenants to be accessed at `sales.server.customer.com` and `marketing.server.customer.com`.

This comma-separated list of internal host names should specify any hosts that will be used to access the provider and any domain from which tenant subdomains will potentially be built. Machines that are included in the `[httpproxy]` host group in the `inventory.ini` file should be included in this list. Machines that are included in the `[CoreServices]` host group in the `inventory.ini` file should be included if your environment includes multiple domains that are a subset of each other. Do not use wildcard characters in this list, and be sure to specify the base host name without any tenant prefixes.

If you have a highly available deployment with multiple Apache HTTP servers, you would want to include each one of those internal hostnames in this comma-separated list.

Below is an example of a full `sitedefault.yml` with multi-tenancy selected. Because no database mode was selected, it gets the standard schema per tenant:

```
config:
  application:
    sas.multi.tenancy.enabled: true
  zones:
    internal.hostnames: server.customer.com
```

Figure 8: Simple `sitedefault.yml`

INFORMATION ABOUT INVENTORY.INI

Ansible uses an inventory file to specify the machines to be included in a deployment and the software to be installed on them. Each inventory file consists of two parts: a deployment target definition, which is a list of each machine used in the deployment, and a host group assignment list, which maps the machines to the function that they will serve and the types of software that will be deployed on them. In a multi-tenant deployment, there are some extra considerations related to the inventory file.

The provider can be considered the initial tenant, or “tenant zero.” The provider gets its own CAS controller, and potentially a backup controller and CAS workers. These are all listed in the host group assignments of the inventory file as `[sas-casserver-primary]`, `[sas-casserver-secondary]`, and `[sas-casserver-worker]`.

In a multi-tenant deployment, the first server listed under `[sas-casserver-primary]` will be the CAS controller for the provider. Every other server underneath this host grouping will have software deployed to the host so that it acts as a CAS server, but it will not be used by the provider. The intent is that if you know the tenant architecture and configuration that you want, you can seed the servers that you want to designate as tenant controllers, backups, and workers. When onboarding, this allows you to avoid redeploying the software to these hosts later.

In the `inventory.ini` file, the server under `[sas-casserver-secondary]` will act as the provider’s backup CAS controller and must meet its requirements. Hosts that are listed under `[sas-casserver-worker]` will be the provider’s worker nodes and will deploy the provider’s CAS server in an MPP architecture.

CUSTOM CERTIFICATES

During the initial deployment, SAS Viya will replace the default self-signed certificates on the Apache HTTP Server with a self-signed certificate that includes a wildcard of subdomains from the fully qualified domain (FQDN) name of the machine. As a result, the default certificate is valid for your FQDN and *.FQDN. This updated certificate, combined with the DNS updates, allows tenants to securely access their tenant-specific URLs.

If you want to use your own custom certificate, make sure the certificate contains a wildcard for the subdomain or has the subject alternative name (SAN) for each tenant. You would then replace the self-signed certificate by following the instructions in the following SAS Viya Administration document: [Data in Motion: Replace Self-Signed Certificates](#).

PROVIDER'S LDAP CONFIGURATION

Once the deployment has completed, the provider administrator will need to log in to SAS Environment Manager using the sasboot account to update the provider's LDAP configuration. This is similar to a standard deployment, except that in a multi-tenant deployment, the provider has an extra option for these configurations. This extra option is present for the three configuration elements under Basic services for the Identities service: the sas.identities.providers.ldap connection, group, and user. In these three configuration instances, there is now a setting to **Apply configuration only to this tenant (provider)**, as shown below:

The screenshot displays the configuration page for 'sas.identities.providers.ldap.connection'. It features several sections:

- GUID:** 6201a217-15b3-4609-9a51-653d4625f3b1. Description: The globally unique identifier for the configuration instance.
- Services:** A dropdown menu showing 'Identities service, SAS Logon Manager'. Description: One or more services to which this configuration instance applies.
- Apply configuration only to th...:** A toggle switch that is currently turned on (blue). Description: When off, the configuration applies to all tenants including the provider. Each tenant can override the configuration from within its own environment.
- anonymousBind:** A toggle switch that is currently turned off (gray). Description: Defines whether Read-Only operations are performed using an anonymous (unauthenticated) context. If enabled, userDN and password will be ignored.

Figure 9: Provider LDAP Configuration

When this option is toggled in the Off position (a gray O), the provider's LDAP configuration for this category applies to all tenants. However, tenant administrators can override this configuration set at the provider level by configuring their own LDAP connection categories. This setting is exclusive to its category: connection, user, or group. For example, if you are connecting all tenants to the same LDAP provider, you might want to leave this setting in

the Off position to propagate the connection settings to all tenants, but leave it in the On position for users and groups, so that those settings will apply to the provider only. Then the provider and each tenant can have unique LDAP user and LDAP group configurations.

ONBOARDING TENANTS

Multi-tenancy enables you to grow your environment as you see fit. You do not have to onboard all tenants at the same time; additional tenants can be added later. The initial deployment of the software does create the provider tenant, which is used to administer the deployment. All subsequent tenants must be onboarded.

Onboarding Prerequisites

Here is a short list of questions to work through before onboarding:

- Is your deployment multi-tenant? This might sound silly, but you need to make sure your deployment is a successful multi-tenant deployment. You can check this in the provider's SAS Environment Manager by navigating to the **Tenants** option in the left panel. If the option is not there, then you must re-install the software with multi-tenancy enabled before onboarding tenants.
- Do you have enough resources in your environment to handle the new tenant? Is there enough space and memory available on your hosts? Do you need to add redundant microservices or infrastructure servers? Are the correct ports open on the hosts where you are deploying?
- Does the environment need to be tuned to operate better? More information can be found in the documentation at [SAS Viya 3.4 Administration: Tuning](#).
- Is the jq package installed on all hosts in the [CommandLine] host group in the inventory file?
- Is the acl package installed, and are ACLs enabled on the mount for the [ComputeServer] host group and [programming] host group?

Onboarding Steps

The first, and perhaps most important, step when onboarding a tenant is making sure you have a full binary backup before onboarding. You can get more information about this in the SAS documentation at the [SAS Viya 3.4 Administration: Backup and Restore](#).

Next, establish your tenant ID. This ID is used by the tenant in many places, including URLs, the file system, and in the LDAP server configuration. The tenant ID must meet the following requirements:

- Must be unique among your tenants.
- Can be 1-16 characters in length and must begin with a letter.
- Can contain only lowercase letters (only a - z in the English alphabet) and numbers.
- The following identities are reserved: default, provider, shared, sharedservices, spre, uaa, viya, and any identity beginning with "sas".

Before you begin to onboard a tenant, you need to set up your tenant users and groups. The bare minimum consists of a valid user that can be configured as the initial tenant's administrator, a tenant administrators group, and the tenant's user group. These are used during the onboarding process to set host-level ownership of tenant-specific files and directories. These could be from LDAP or exist only at the host level, but they must exist and be known at the host level before onboarding.

Create a Tenant-Specific Onboarding File

When executing the initial SAS Viya multi-tenant deployment, you updated the `vars.yml` file to add a property for SAS Studio 4. This `vars.yml` contains deployment variables that enable you to customize the deployment to meet your needs. When you onboard a tenant, you create a tenant-specific `vars.yml` to uniquely configure the tenant. In SAS documentation, this file is referred to as `tenantID_vars.yml` (where `tenantID` is an ID that fits the tenant naming restrictions).

This file is where you list the specific configurations for your tenant. These configuration settings include how you want your CAS servers configured, the filesystem permissions on the tenant-specific directories and files, the tenant ID, and more.

To begin, SAS recommends that you copy the `sample_tenant_vars.yml` file from the `samples` directory that is provided in your deployment to the `sas_viya_playbook` directory. This file will be named `tenantID_vars.yml` at that location; for example, you could run the following command to initiate the configuration of a Marketing tenant:

```
cp samples/sample_tenant_vars.yml marketing_vars.yml
```

Once the sample has been copied, you can start updating the fields in the `tenantID_vars.yml` file.

The `sample_tenant_vars.yml` file includes some great descriptions of what the configuration values control. These explanations are helpful when configuring these tenant-specific values for onboarding. Some descriptions explicitly call out `*** Do not edit this value ***`. These fields are prepopulated with Ansible references and should not be changed.

Throughout the file, many fields are labeled with "`<replace me>`" indicating where you must add tenant-specific details. In these fields, be sure to retain the quotation marks, as those are important to the processing. Other fields are stubbed out with only quotation marks. These indicate optional configurations, such as declaring which host from `inventory.ini` you want to serve as the backup CAS controller. If you are not updating these fields, leave them unchanged.

The first property to set in the file is the tenant ID. In the file, this field is noted as `sas_tenant: "<replace me>"`. As in the previous example, if this was the `marketing_vars.yml`, you would set this value as `sas_tenant: "marketing"`.

Another field to pay attention to is the `skip_ldap_config`. This field can have a value of `true` or `false`, and it determines whether the tenant will use the provider's configuration for LDAP connection, or will instead use its own. If you have chosen to configure an LDAP connection per tenant, then this value should be `skip_ldap_config: "true"`. This will prevent the onboarding process from picking up the provider's LDAP connection details and will enable you to set your tenant's specific LDAP configurations in SAS Environment Manager after onboarding.

The next three fields are the `tenant_admin`, `tenant_admin_group`, and `tenant_users_group`. As discussed previously, these will be used for filesystem permissions on tenant-specific directories and files.

Next you need to set the password for the tenant-specific `sasprovider` account in the field `tenant_provider_pwd`. This account is the initial tenant administrator, and natively is a member of SAS Administrators. This account resembles `sasboot`, and the `sasprovider` account is used by the provider to finish configuring the tenant for access. This tenant-specific `sasprovider` account has administrative access within the tenant and cannot be disabled by a tenant administrator. Because the password is set in the file, restricting

access to this file or removing this value from the tenant's vars.yml file after the tenant has been onboarded is highly recommended.

The next two properties enable the tenant to be onboarded. They are the ID of a provider administrator and its password; these values are used to programmatically onboard the tenant. In the fields of *provider_admin* and *provider_admin_pwd*, set the admin's ID and password for the account. This ID can be any account that is in the provider's SAS Administrators group (sasboot is used by default in the sample_tenant_vars.yml file). It is highly recommended that you restrict access to this file or remove this value from the tenant's vars.yml file after the tenant has been onboarded.

If SAS/CONNECT® is licensed, the following values must be set for the tenant-specific SAS/CONNECT configuration:

- *sasenv_connect_port* – requires a unique port number for SAS/CONNECT Spawner. This process typically runs on port 17551, and it is typically incremented starting here as you onboard tenants.
- *sasenv_connect_mgmt_port* – requires a unique port number for monitoring the SAS/CONNECT Spawner. This process typically runs on port 17541 and is also normally incremented.

These fields are in the sample_tenant_vars.yml file that you copied earlier. If SAS/CONNECT is not licensed, these lines can be commented out.

The next configuration to set is the unique port for the tenant's SAS Object Spawner. This is a tenant-specific process that is used to launch a tenant's SAS Workspace Server processes. This property is set under the SPAWNER_CONFIGURATION block as the variable *sasPort*. The port that is used must be unique and, given the SAS Object Spawner by default runs on port 8591, it is typically started at 8592 and incremented as you onboard tenants.

The largest block of configuration values in the tenant's vars.yml file is for the tenant's CAS configuration. The CAS configuration from the provider is not automatically enabled for the tenant, and therefore these values orchestrate how the tenant's CAS server will be deployed. These properties let you select hosts from the inventory.ini file to use for the tenant's CAS hosts, the ports that CAS will use, and additional tenant-specific CAS configurations. The fields for *primary_host*, *secondary_host*, and *worker_hosts* declare how you want your hosts to be used for this tenant's CAS server. The values from your inventory.ini file that can be used for these configuration variables need to be listed in the [sas-casserver-primary] host group. Figure 10 shows an example of values for these variables. The hosts are mapped in the inventory.ini file and are in the [sas-casserver-primary] host group:

```

CLUSTER_DEFINITIONS:
  cas:
    default:
      # The primary host value is the Ansible alias name from a host in the
      # sas-casserver-primary group in the inventory.ini file. There should only be
      # one entry for the primary_host. This host will be the main CAS controller
      # for the CAS grid

      primary_host: "sharedControler"

      # The secondary host value is the Ansible alias name from a host in the
      # sas-casserver-primary group in the inventory.ini file. There should only be
      # one entry for the secondary_host. This host will be the backup CAS controller
      # for the CAS grid

      secondary_host: "sales_backup"

      # The worker hosts value is a comma-separated list of the Ansible alias names
      # from the sas-casserver-primary group in the inventory.ini file.
      # The listed hosts will be the workers for the CAS grid

      worker_hosts: "sales_worker_1, sales_worker_2, sales_worker3"

```

Figure 10: Example Tenant CAS Configuration

In a multi-tenant environment, you could have many varied combinations of CAS configurations. You could have all tenants sharing a CAS controller, or you could have workers shared across tenants. Also, within this CLUSTER DEFINITIONS block, you set the tenant-specific CAS_DISK_CACHE, identify a load balancer for CAS, and add any CAS settings.

To avoid collisions of the CAS process on a shared host, update the CAS *port* and *httpport*. You can leave these as the defaults if you do not have tenants that share CAS hosts. If you do have CAS server processes sharing a host, then you must use unique ports for each tenant that is using the host.

Run the Tenant Onboarding Script

Once the *tenantID_vars.yml* file is customized for your tenant, run the Ansible script to onboard the tenant. For example, if you have created and configured the *marketing_vars.yml* for the Marketing tenant, you would run the following:

```

ansible-playbook -i inventory.ini utility/multi-tenancy.yml -e
"@marketing_vars.yml" -vv

```

If any errors occur during onboarding, you will potentially see failures in the output. You can also view the tenant-specific onboarding log, which is created at the following location:

```

/opt/sas/viya/ config/var/log/multitenant/tenantID/tenant_onboard.log

```

If there are no errors, you can access the provider's instance of SAS Environment Manager, navigate to the **Tenants** page, and see the tenant with a status of Onboarded. Figure 11 shows an entry for multiple tenants and their status:

| Tenants | | | | |
|-----------|-------------|------------|---------------|------------|
| Name | Description | Status | Access Policy | Created By |
| acme | | Onboarded | Open | sasboot |
| marketing | | Onboarded | Open | sasboot |
| sales | | Onboarding | Limited | sasboot |

Figure 11: Tenant Status After Onboarding

Some services take some time to move from the state of Onboarding to Onboarded. You can verify the state of services by highlighting the tenant from the **Tenants** page and selecting **Service Status**. That action populates a pop-up window like the one shown below:

| Service Status | |
|---------------------|------------|
| Service | Status |
| SASVisualAnalytics | Onboarding |
| searchIndex | Onboarding |
| analyticsComponents | Onboarded |
| analyticsFlows | Onboarded |

Count: 114

Figure 12: Tenant Services Status with Mixed States

If a service’s status does not change to Onboarded after a reasonable amount of time, you will need to troubleshoot the issue. To investigate these individual services, find the appropriate log file and look for errors or warnings related to the onboarding process. In the example above, you would navigate to the SAS® Visual Analytics log on the microservices host. In this log, you would search for the word “onboarding” and examine any associated warnings or errors. These services will continuously retry the onboarding process, even after a failure, until they get a status of Onboarded.

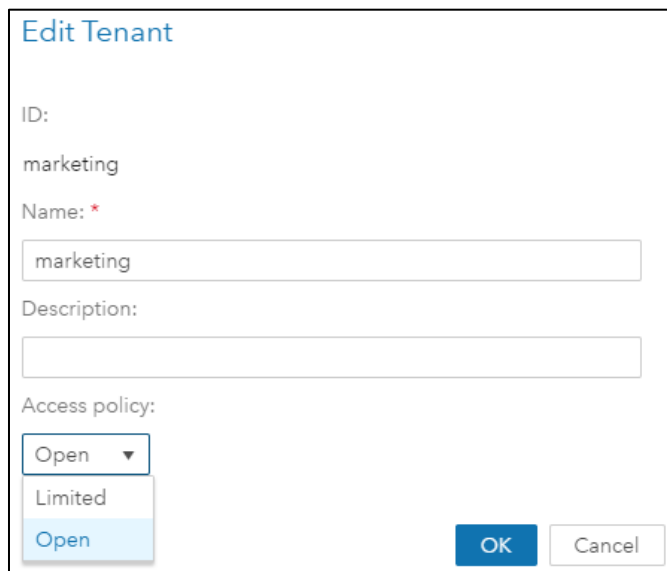
Configure LDAP for the Tenant

If you have selected `skip_ldap_config: "true"` in your tenant-specific vars.yml file, then once the tenant onboarding process has completed, the provider administrator will need to log on to the tenant using the sasprovider account and configure the tenant’s LDAP connection. The provider administrator must navigate to `tenantID.hostname/SASEnvironmentManager` and update the configuration for the Identities service’s `sas.identities.providers.ldap` for its connection, group, and user.

Once the users and groups are surfaced in the tenant’s environment, the provider administrator (still using the tenant’s sasprovider account) will define the initial tenant administrators by adding those users as members of the tenant’s SAS Administrators group.

Enable the Tenant

After the tenant has been onboarded and has its LDAP connection configured, the provider needs to enable the tenant. When the tenant is onboarded, its Access Policy is set to *Limited*. This setting means that only the tenant's sasprovider account can log on, but no one else can access the tenant environment. When all onboarding tasks have been completed, the provider administrator can set the Access Policy to *Open*.



The screenshot shows a dialog box titled "Edit Tenant". It contains the following fields and controls:

- ID:** A text field containing "marketing".
- Name: *** A text field containing "marketing".
- Description:** An empty text field.
- Access policy:** A dropdown menu with "Open" selected. The dropdown list shows "Open", "Limited", and "Open" (highlighted).
- Buttons:** "OK" and "Cancel" buttons at the bottom right.

Figure 13: Settings for Tenant Access Policy

This setting is accessed through the provider's **Tenants** page in SAS Environment Manager. It is shown in Figure 13, above, for the tenant named Marketing. This setting can be changed at any time, to allow or to prevent tenant users from logging in to the tenant environment. However, any users who currently have active sessions will not be interrupted and will continue to have access for the duration of their session.

What's Next: Task for the Tenant

Once the tenant's access policy is in the Open state, all users that belong to that tenant (that is, users that are returned from its LDAP configuration) will be able to log on and access the tenant environment. It is now the responsibility of the tenant administrators to set up their structure and security around data, content, and other SAS administrative tasks.

OFFBOARDING

Offboarding is the process of decommissioning and removing a tenant from the system. SAS provides specific Ansible playbooks that must be run to offboard a tenant. You can choose whether to delete or disable a tenant. For details regarding the offboarding process, reference the following SAS Viya Administration documentation: [Multi-Tenancy: Offboard a Tenant](#). This section of the documentation provides specific details about the required steps to offboard a tenant, disable a tenant, and re-onboard a tenant.

CONCLUSION

A SAS Viya 3.4 multi-tenant deployment can be a very useful way to leverage SAS Viya. It allows for segregation of data and content for discrete groups of end users. For complex deployments, multi-tenancy might be the only way to implement an important business requirement, such as connecting a single deployment of SAS Viya to multiple LDAP providers.

Multi-tenant deployments allow for flexibility of architecture, while allowing IT resources to be shared in a secure and cost-effective manner. If proper planning goes into the deployment, and deployment considerations are taken into account, multi-tenancy can drive business value in SAS Viya.

REFERENCES

SAS Institute Inc. 2018. *SAS Viya 3.4 Administration: Multi-tenancy*. Available at <https://go.documentation.sas.com/?cdcId=calcdc&cdcVersion=3.4&docsetId=caltentants&docsetTarget=titlepage.htm&locale=en>.

SAS Institute Inc. 2018. *SAS Viya 3.4 for Linux: Deployment Guide*. Available at <https://go.documentation.sas.com/?cdcId=calcdc&cdcVersion=3.4&docsetId=dplyml0phy0lax&docsetTarget=titlepage.htm&locale=en>.

ACKNOWLEDGMENTS

A huge thanks to Angie Hedberg, Philip Hopkins, Eric Bourn, Gilles Chrzaszcz, Wade Adkins, Tom Reichmeider, Alex Kubacki, Dara Davis, and Susan Pearsall for their help and support.

CONTACT INFORMATION

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

Eric Davis
SAS® Institute
eric.davis@sas.com

SAS and all other SAS Institute Inc. product or service names are registered trademarks or trademarks of SAS Institute Inc. in the USA and other countries. ® indicates USA registration.

Other brand and product names are trademarks of their respective companies.