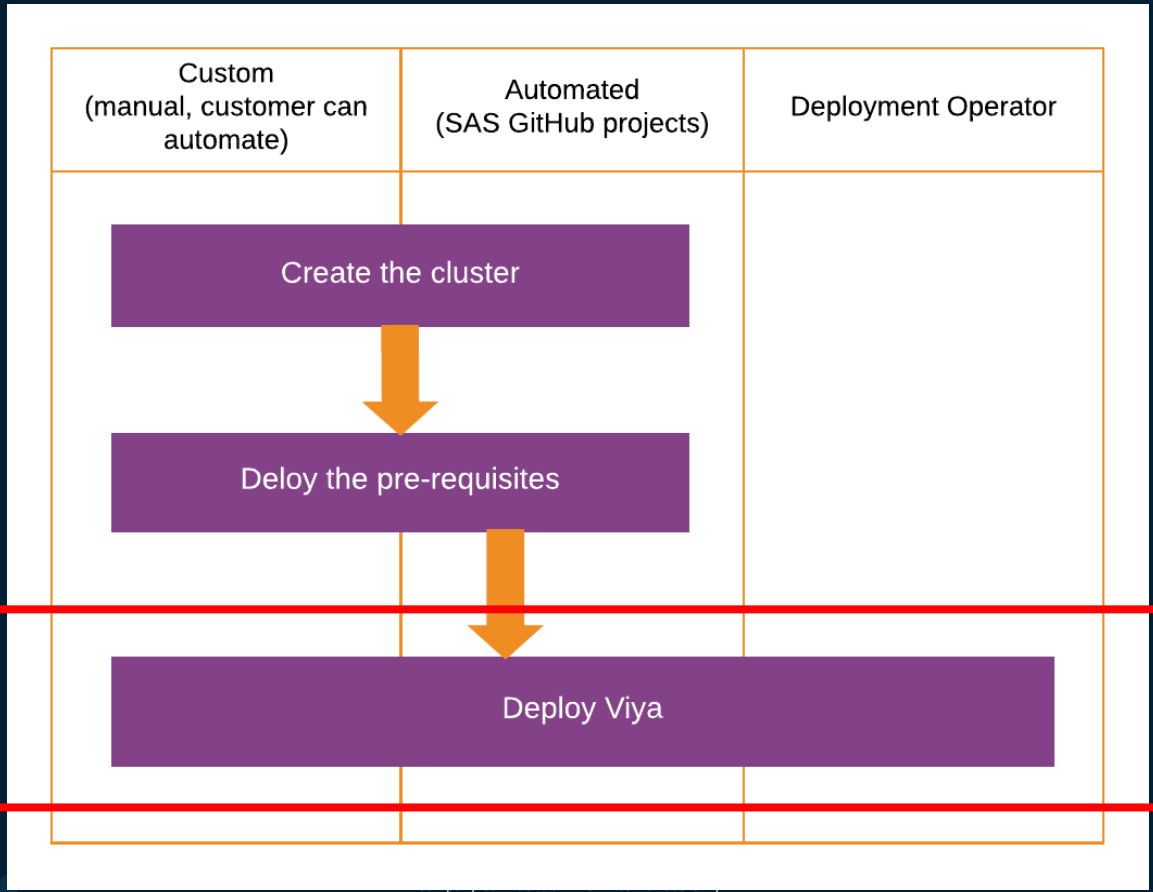
Deployment Operator - Introduction

SAS Viya 4

Fans September 29, 2021

Ole-Martin Hafslund

Deployment Choices



Overall Deployment Steps

Simplified list of steps – Manual Deployment

1. Get the required assets:
   - A Software Order is created
   - From the Software Order E-mail, follow the link to the my.sas.com portal
   - On a Kubernetes client (`kubectl`) machine, download the “Deployment Assets” and explode the .tgz file
2. Prepare the configuration
   - Create a "`kustomization.yaml`" file. (examples in the doc)
   - Build a Kubernetes Manifest file (`site.yaml`) with `kustomize`
3. Deploy SAS Viya
   - Apply the Kubernetes Manifest file to your Kubernetes Cluster
   - Wait for the environment to be fully started
4. Validate it

Overall Deployment Steps

Simplified list of steps – Using the Deployment Operator

1. Get the required assets:
   - A Software Order is created
   - From the Software Order E-mail, follow the link to the my.sas.com portal
   - On a Kubernetes client (`kubectl`) machine, download the “Deployment Assets” and explode the .tgz file
2. Deploy the Deployment Operator
   - Create a "`kustomization.yaml`" file. (examples in the doc)
   - Build a Kubernetes Manifest file with `kustomize`
   - Apply the Kubernetes Manifest file to your Kubernetes Cluster
3. Prepare the configuration
   - Create a "`kustomization.yaml`" file. (examples in the doc)
   - Create the SASDeployment custom resource definition file
4. Deploy SAS Viya
   - Apply the SASDeployment custom resource definition file
   - Wait for the environment to be fully started
5. Validate it

SAS Viya Deployment Operator

The basics

- The SAS Deployment Operator can facilitate some aspect of the deployment and maintenance of a SAS Viya environment

- Starting with Stable-2020.1.4 (and therefore, LTS-2021.1), it is the recommended method to install SAS Viya, but you can still do a manual deployment
  - It is not mandatory to use the Deployment Operator

SAS Viya Deployment Operator

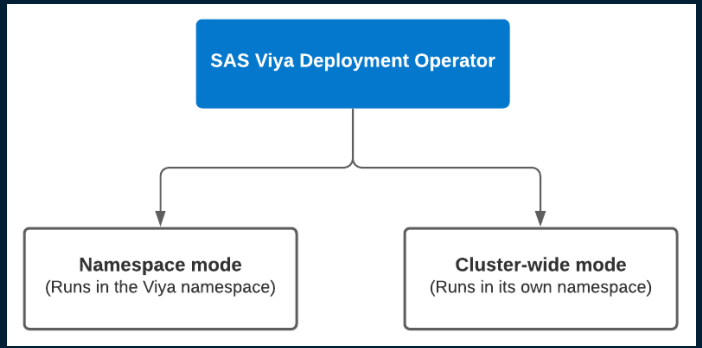
The basics

- Among other things, it can:
  - Build your manifest by running the '**kustomize build**' command and perform the deployment ('**kubectl apply**')
  - Manage and automate the deployment of multiple SAS Viya environments in multiple namespaces
  - Automatically grab the most up-to-date deployment assets within a CADENCE_VERSION, and apply them
  - Helps with going from one CADENCE_VERSION to the next

SAS Viya Deployment Operator

The basics

- The Deployment Operator can be installed (deployed) in the cluster in 2 ways – it can run in two modes:
  - Namespace mode:
    - Running in namespace mode the Deployment Operator can only manage the Viya environment in a single namespace
  - Cluster-wide mode:
    - When running in cluster-wide mode the Deployment Operator runs in its own namespace and can manage multiple Viya environments (in multiple namespaces)

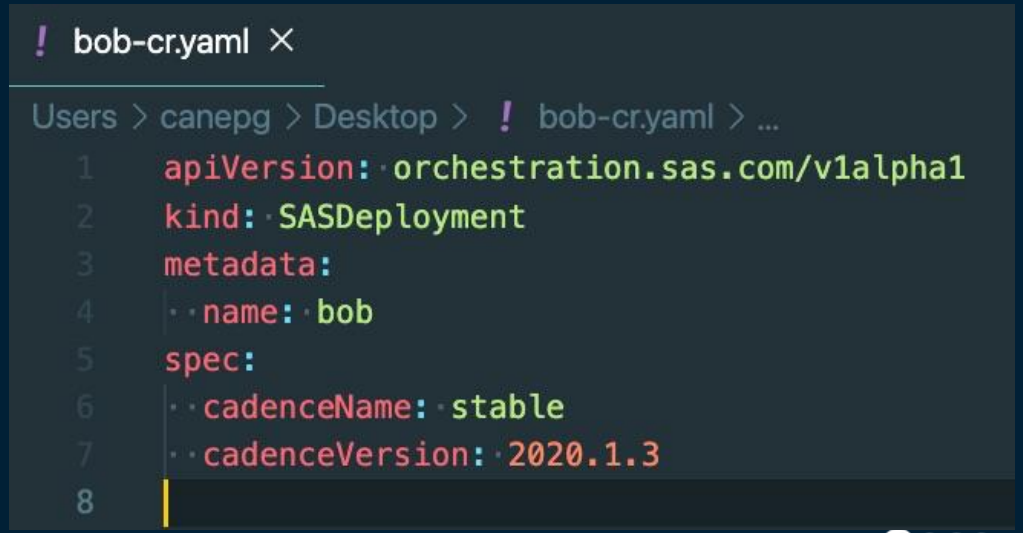


- The Deployment Operator keeps your Viya Environment in sync with the definition of the SASDeployment custom resource

SAS Viya Deployment Operator

The basics

- The Deployment Operator runs inside the K8s cluster as a pod
- A new Custom Resource (CR) is declared in the cluster, of the kind 'SASDeployment'
- You can then "declare" a new Custom Resource Definition (CRD), called "bob", of the kind, SASDeployment
- This would lead to a new Viya deployment being created in its own namespace



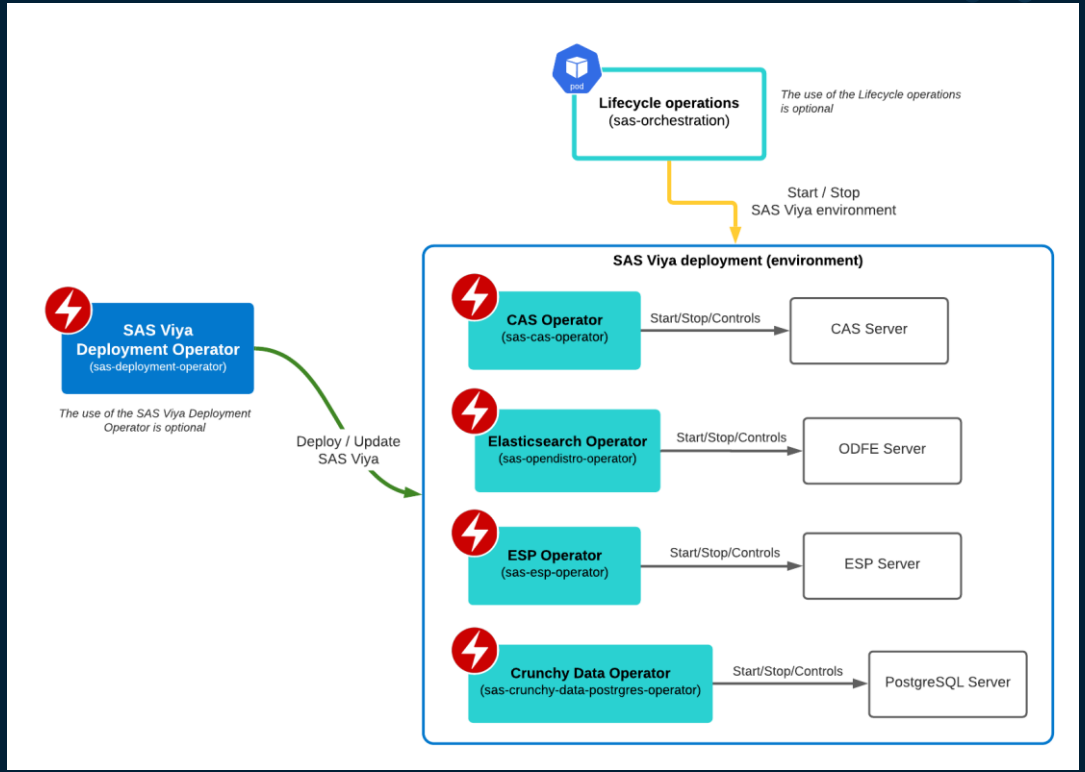
SAS Viya Deployment Operator

The Deployment Operator and Lifecycle operations

- The deployment operator is different from the Lifecycle operations
- The deployment operator is a SAS-built Kubernetes operator
  - Used to deploy and update the SAS Viya software
- The Lifecycle operations (the Orchestration Tool) is an application pod
  - Used to start and stop the SAS Viya environment
- Both can run in the Viya namespace or in a separate namespace

The Deployment operator and Lifecycle operations

There are also several operators within the Viya deployment



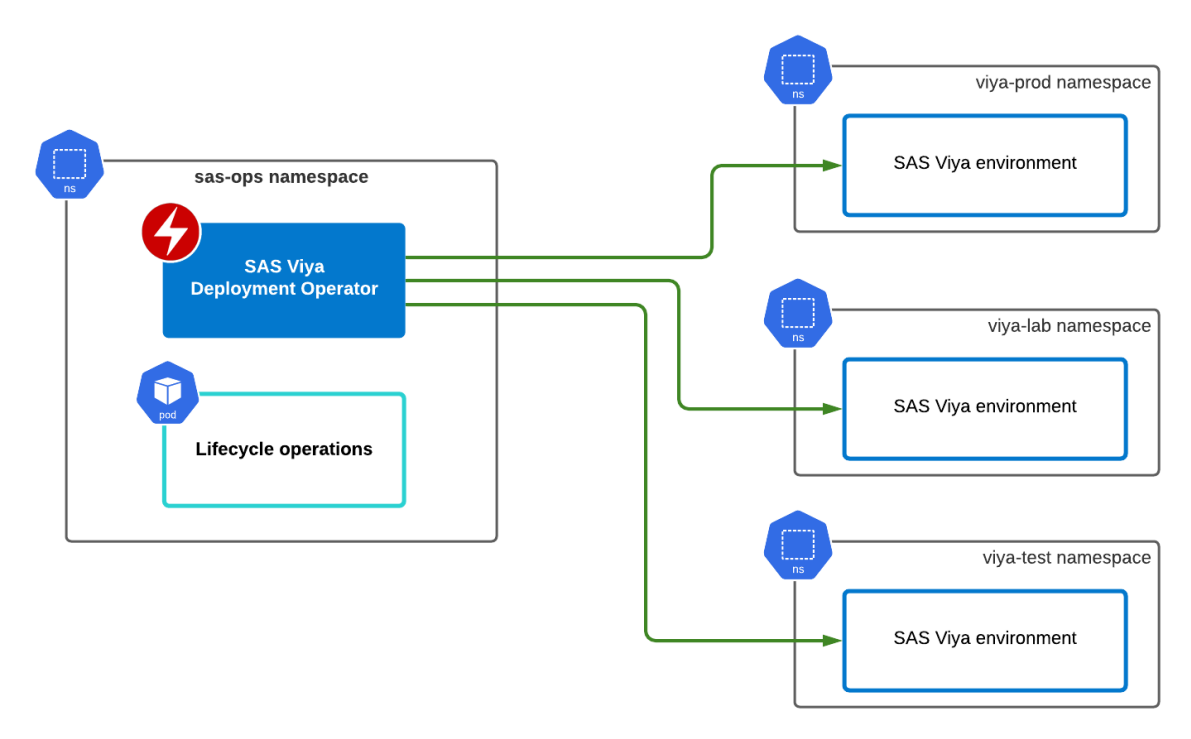
The Deployment Operator and Lifecycle operations

Using a dedicated namespace

- It is recommended that both the Deployment Operator and the Lifecycle operations run in a separate namespace from the Viya namespace(s)
  - For example, using a "**sas-ops**" namespace
- This allows both to control multiple Viya deployments
- The Viya namespace(s) can be deleted without affecting the Deployment Operator or the Lifecycle operations

The Deployment operator and Lifecycle operations

Using a dedicated namespace



SAS Viya Deployment Operator

Can be used to deploy multiple Viya cadence versions

- The Deployment Operator is independent of the SAS Viya cadence version being deployed
  - It is not tied to a specific Stable or LTS cadence
  - The operator is backwards compatible
    - It is not locked to a specific SAS Viya version
    - For example, you can use the operator that you pulled at Stable-2021.1.2 to deploy LTS-2021.1 and older Stable cadence versions
  - You need a process to keep the operator up to date, it doesn't automatically update itself
    - It is recommended that you deploy the latest operator before performing any new deployments and/or upgrading an environment

What does the CR definition include?

SAS Viya Deployment Operator

What does the CR definition include

- The custom resource definition contains information on:
  - The SAS Viya cadence
  - The license and entitlement certificates
  - The SAS Viya configuration
- For example…

```yaml
apiVersion: orchestration.sas.com/v1alpha1
kind: SASDeployment
metadata:
  annotations:
    operator.sas.com/checksum: ""
  creationTimestamp: null
  name: lab-sasdeployment
spec:
  license:
    secretKeyRef:
      key: license
      name: sas-viya
  clientCertificate:
    secretKeyRef:
      key: cert
      name: sas-viya
  caCertificate:
    secretKeyRef:
      key: cacert
      name: sas-viya
  cadenceName: lts
  cadenceVersion: "2021.1"
  cadenceRelease: ""
```

SAS Viya Deployment Operator

Understanding the cadenceRelease and updatePolicy

- The value of cadenceRelease in the SASDeployment CR can be set in one of three ways:
  1. Explicitly by the user when they apply the CR. For example, cadenceRelease: "20210406.1617742339205“
  2. Set to “” by the user, implying the cadence release should auto-pinned to the latest
  3. Set by the updatePolicy (Never or Releases)
- SAS Deployment Operator will reconcile anytime there is a change to the SASDeployment CR spec
  - This includes cadenceRelease, among other fields
  - The operator doesn’t know how this field gets updated but is notified by Kubernetes when there is a change it might care about

SAS Viya Deployment Operator

Understanding the cadenceRelease and updatePolicy

- The **updatePolicy** impacts the behavior of the autoupdate CronJob that is deployed into any Viya namespace that is managed by SAS Deployment Operator
  - The CronJob will update the status of the CR with the latest information (about what new stuff is available to the user)
- If the updatePolicy is set to Releases, it will also update the cadenceRelease in the spec with what it determines to be the latest release for the customer assigned cadence version (e.g. LTS-2021.1)
- As described earlier, the operator will be notified of this spec change and re-reconcile the deployment
- While the user can apply a CR with cadenceRelease set to “”, it is always filled in automatically with the latest value at the time of the reconciliation

Using an inline configuration

SAS Viya Deployment Operator

Using an inline definition

- The SASDeployment custom resource definition contains the Viya configuration (hence, the “inline” name)
- This includes the kustomization.yaml and all other configuration that is required for the Viya deployment
- Can quickly become very unwieldy (cumbersome) and is fraught with YAML syntax (indentation) errors
- The SAS Orchestration Tool is the best option for building an inline CR

SAS Viya Deployment Operator

Example of using an inline definition

```
  cadenceName: "stable"
  cadenceVersion: "2020.1.5"
  cadenceRelease: ""
  # The following is an example of how to specify inline user content.
  # See documentation for specifying license, client certificate,
  # and certificate authority certificate.
  userContent:
    files:
      "kustomization.yaml": |-
        ---
        namespace: lab
        resources:
          - sas-bases/base
          - sas-bases/overlays/cert-manager-issuer     # TLS
          - sas-bases/overlays/network/ingress
          - sas-bases/overlays/network/ingress/security   # TLS
          - sas-bases/overlays/internal-postgres
          - sas-bases/overlays/crunchydata
          - sas-bases/overlays/cas-server
          - sas-bases/overlays/update-checker       # added update checker
          #- sas-bases/overlays/cas-server/auto-resources    # CAS-related
        configurations:
          - sas-bases/overlays/required/kustomizeconfig.yaml  # required for 0.6
        transformers:
          - sas-bases/overlays/network/ingress/security/transformers/product-tls-transformers
          - sas-bases/overlays/network/ingress/security/transformers/ingress-tls-transformers
          - sas-bases/overlays/network/ingress/security/transformers/backend-tls-transformers
          - sas-bases/overlays/required/transformers.yaml
          - sas-bases/overlays/internal-postgres/internal-postgres-transformer.yaml
          - site-config/security/cert-manager-provided-ingress-certificate.yaml     # TLS
          #- sas-bases/overlays/cas-server/auto-resources/remove-resources.yaml    # CAS-rela
          #- sas-bases/overlays/scaling/zero-scale/phase-0-transformer.yaml
          #- sas-bases/overlays/scaling/zero-scale/phase-1-transformer.yaml
        patches:
          - path: site-config/storageclass.yaml
            target:
              kind: PersistentVolumeClaim
              annotationSelector: sas.com/component-name in (sas-cas-operator,sas-backup-job,
e-data-deploy-utilities,sas-data-quality-services,sas-model-publish,sas-commonfiles)
        configMapGenerator:
          - name: ingress-input
            behavior: merge
            literals:
              - INGRESS_HOST=lab.snzmgo.gelsandbox.aks.unx.sas.com
            name: sas-shared-config
```

Create the kustomization file

```
        generators:
          - postgres-custom-config.yaml
      "site-config/storageclass.yaml": |-
        ---
        kind: PersistentStorageClass
        metadata:
          name: wildcard
        spec:
          storageClassName: sas-azurefile #azurefile with sas UID/GID
      "sitedefault.yaml": |-
        ---
        config:
          application:
            sas.logon.initial:
              user: sasboot
              password: lnxsas
      "postgres-custom-config.yaml": |-
        ---
        apiVersion: builtin
        kind: ConfigMapGenerator
        metadata:
          name: postgresql-custom
        behavior: merge
        literals:
        - |
          postgres-ha.yaml=
          ---
          bootstrap:
            dcs:
              loop_wait: 10                                    # Added through SAS provide
              ttl: 30                                          # Added through SAS provide
              master_start_timeout: 0                          # Added through SAS provide
              postgresql:
                parameters:
                  archive_timeout: 60                          # Added through SAS provide
                  checkpoint_completion_target: 0.9            # Added through SAS provide
```

Create the storageclass config

Create the sitedefault config

Create the postgres config

Using a Git repository, HTTP file server or a file share

SAS Viya Deployment Operator

Using a Git repository, HTTP file server or a file share

- The operator uses go-getter to access the files
  - See HashiCorp's Go Getter "URL Format" documentation for details:
  - https://pkg.go.dev/github.com/hashicorp/go-getter@v1.4.1?tab=overview#url-format
- The configuration files are defined under the 'userContent' field
  - This is to access the root folder for the Viya configuration files
- The license and certificates must use explicit file references

SAS Viya Deployment Operator

Using a Git repository

- Go-getter file access
  - The ‘git’ tag support is only for reading at the folder level
  - The HTTP tag is used to directly access files
- Therefore, if you are use Git for all files
  - You must use the ‘git::’ parameter for the userContent
  - You must use the ‘HTTP’ access for the following:
    - License (*.jwt file)
    - The entitlements certificates
    - The CA certificate

Using a Git repository

Example

```
apiVersion: orchestration.sas.com/v1alpha1
kind: SASDeployment
:

userContent:
  url: git::http://git_server/username/project.git
license:
  url: http://git_server/username/project/-/raw/master/.../SASViyaV4_license.jwt
clientCertificate:
  url: http://git_server/username/project/-/raw/master/.../entitlement_certificate.pem
caCertificate:
  url: http://git_server/username/project/-/raw/master/.../SAS_CA_Certificate.pem
```

Updating the Viya deployment

Using the Deployment Operator

Updates to the Viya environment

Using the Deployment Operator

Let’s discuss two scenarios:

1. An administration update
   For example, changing from using an SMP CAS server to a MPP CAS server

2. Updating from one cadence version to another
   For example:
   - LTS-2021.1 to LTS-2021.2
   - Stable-2020.1.4 to Stable-2021.1.1
   - LTS-2021.1 to Stable-2021.1.2

Updates to the Viya environment

Using the Deployment Operator for an admin change

- The deployment operator is **ONLY** looking for changes to the SASDeployment CR objects
  - This WILL trigger an update to the Viya deployment
  - So, what happens if you are NOT using an inline configuration
- If the configuration change is in one of the other YAML files
  - For example, increasing the number of CAS Workers, or changing from SMP to MPP CAS
  - This does NOT automatically trigger an update to the Viya deployment
  - You need to reset the checksum on the SASDeployment custom resource object to trigger the update, as the SASDeployment CR definition has not changed

Updates to the Viya environment

Resetting the SASDeployment CR checksum

- There are two ways to reset the SASDeployment CR checksum
  1. Issue the following command:
     `kubectl -n {{namespace}} annotate sasdeployment {{name}} operator.sas.com/checksum-`
  2. Add the following annotation to the SASDeployment CR metadata
     `operator.sas.com/checksum: ""`

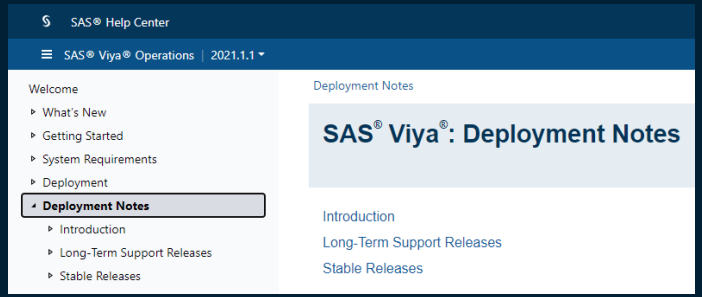
```
apiVersion: orchestration.sas.com/v1alpha1
kind: SASDeployment
metadata:
  annotations:
    operator.sas.com/checksum: ""
  creationTimestamp: null
  name: lab-sasdeployment
```

Updates to the Viya environment

Using the Deployment Operator for a version update

- You can use the operator to update from one cadence version to another
- The change must be within the supported change window
  - To a “newer” cadence version (you can’t move backwards)
  - Within the supported change (within 4 cadence versions)
- You need to have a valid Viya configuration for the new version

  For example:
  - Check the Deployment Notes for any required changes
  - The kustomization.yaml must be configured for that cadence version



End of module

sas.com