PROGRAM GUIDE: **November 2025**

Migrate Tanzu Mission Control SaaS to SM

A Migration Guide

Table of contents

Version History

Date	Ver.	Author	Contributors	Description	Reviewers
2025-07-18	1.01	Multiple		Initial release	
2025-08-05	1.02		Steven Zou	Update scripts	
2025-08-08	1.03		Steven Zou	Replace code blocks with script path references	
2025-11-13	1.0.4		Willis Ren	Update the Harbor registry requirement and risk of cluster rolling update	
2025-11-20	1.0.5		Willis Ren	Updated the script links and added FAQs	

Purpose

Tanzu Mission Control SaaS (TMC-SaaS) is in deprecation and will reach end-of-life on November 15, 2025. TMC-SaaS will not be the target of further development or maintenance by Broadcom. This document describes a migration process that helps customers migrate from TMC-SaaS to TMC Self-Managed (TMC-SM). The migration process delivers a concrete solution for performing the migration using the TMC API (CLI) and some supporting scripts.

Goals

The primary objective of this document is to outline the comprehensive methodology for migrating resources from TMC-SaaS to SM. Specifically, this document will:

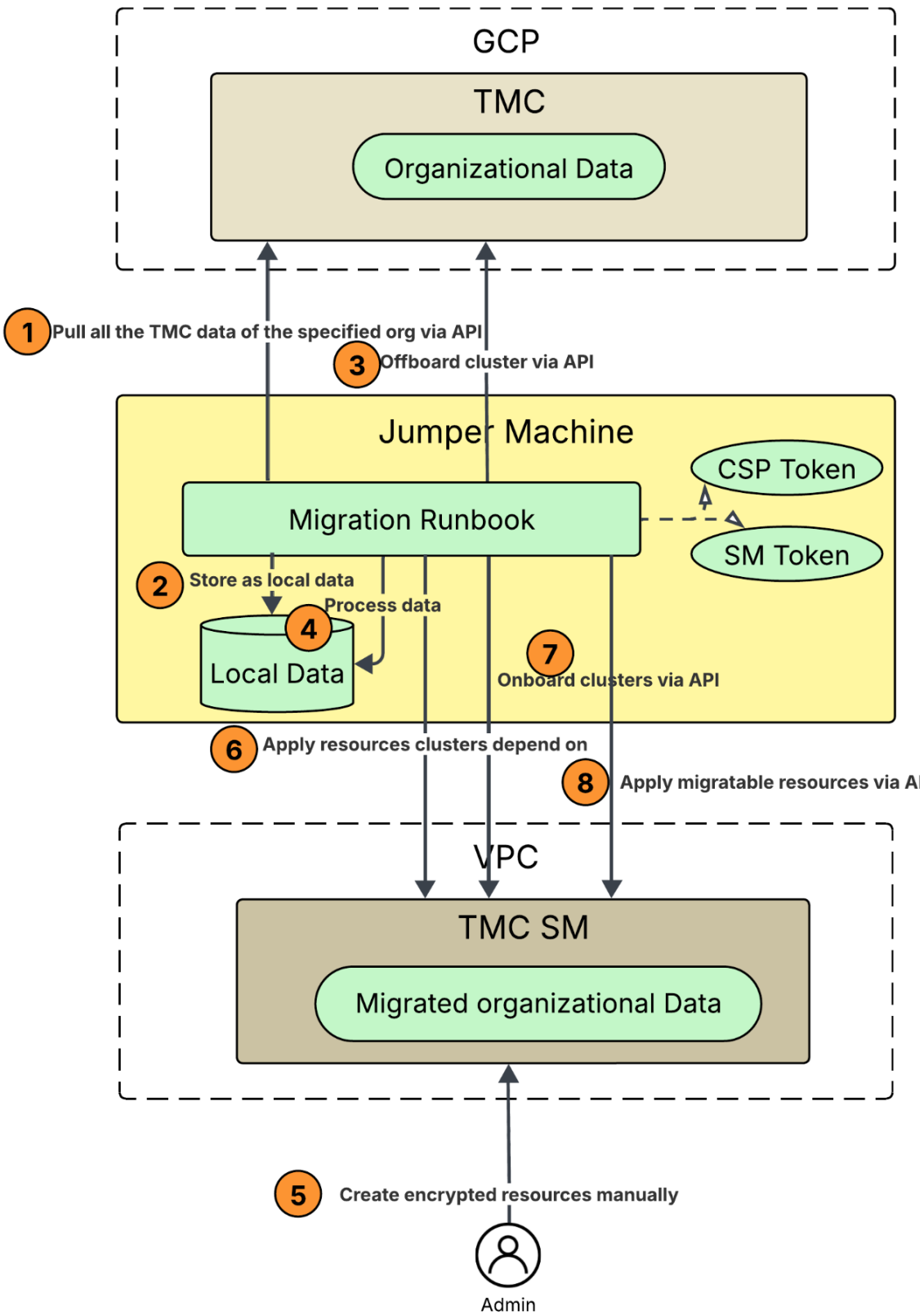
- Define the scope of the migration by identifying applicable clusters based on their type, cloud provider, and Kubernetes version.
- Detail the procedures for managing TMC resources, including:
  - Exporting resources from the SaaS environment.
  - Reapplying resources to the SM environment.
  - Analyzing resource dependencies.
- Establish the procedure for onboarding clusters from the SaaS environment to the SM environment.
- Describe the end-to-end migration execution process, specifying a logical sequence to mitigate potential resource dependency conflicts.

Prerequisites

- A jumper machine set up to run scripts with Tanzu CLI and tools like yq 4.48.2, jq 1.6 and curl installed, and all the supporting scripts from Appendix A
- Ensure the network settings of the jumper machine can reach the TMC-SaaS instance.
- TMC-SM installed following the official documentation.
  - Version v1.4.2+ is required.
  - The Harbor registry should NOT be deployed with the carvel package in a TKG cluster to be managed by TMC, either via the TMC SaaS Catalog or the Tanzu standard package. It should be a separate and independent Harbor installation.
  - The Cert-manager, Contour and Harbor should NOT be deployed in the cluster for TMC-SM installation.
- The CSP token of the organization admin for SaaS and the SM admin credentials is available.
- The related credentials stored in the SaaS account manager are available.
- If the jumper machine cannot reach the TMC-SM environment, a similar one must be set up and configured with proper network settings to ensure it can reach TMC-SM. The data exported to the jumper machine must be copied to the one that can reach TMC-SM before the import phase.

General Migration Process

The main process for migrations between TMC-SaaS and SM is via the related TMC APIs instead of data migration. The primary steps of migration are described in the diagram below.


GCP
TMC
Organizational Data
1 Pull all the TMC data of the specified org via API
3 Offboard cluster via API
Jumper Machine
CSP Token
Migration Runbook
SM Token
2 Store as local data
Process data
4
Local Data
7
Onboard clusters via API
6 Apply resources clusters depend on
8 Apply migratable resources via A
VPC
TMC SM
Migrated organizational Data
5 Create encrypted resources manually
Admin

Migration Steps

1. Extract the migratable resources from the SaaS environment.
2. Store the resource data as local data in YAML/JSON format.
3. Offboard (detach/unmanage/deregister) clusters from TMC-SaaS to avoid potential conflicts.
4. Process the data to clear system-annotated info and make it ready to reapply to TMC-SM.
5. Create resources containing encrypted information that cannot be directly processed by the migration script or CLI without customer interaction.
6. Apply the resources that clusters depend on before onboarding the clusters on SM
7. Onboard (register/manage/attach) the clusters to the TMC-SM instance.
8. Apply the migratable resources targeting the TMC-SM instance.

Scope

Clusters

The following cluster types and resources associated with them should be migrated:

Cluster Type	Version Restriction	Notes
TKGs/VKS	Infrastructure VC8 K8s version >=1.29	
TKGm	TKGm version >=2.5.0 K8s version >=1.29	
Attached (CNCF-conformant)	K8s version >=1.29 On vSphere infrastructure	

Table 1: Clusters to be migrated.

Other cluster types and clusters with unsupported versions will be excluded and are not candidates for migration, as listed below:

Cluster Type	Version Restriction	Notes
TKGs/VKs	Infrastructure VC7 K8s version <1.29	
TKGm	TKGm version < 2.5.0 K8s version < 1.29	
EKS (LCM)	All	TMC-SM does not support EKS LCM. Refer to this document to unmanage your EKS cluster prior to migrating from TMC-SaaS to SM.
AKS (LCM)	All	TMC-SM does not support AKS LCM. Refer to this document to unmanage your EKS cluster prior to migrating from TMC-SaaS to SM.

Table 2: Clusters that can not be migrated.

Resources

Migrations can be skipped for the resource types listed in the table.

Resource Type	Justifications	Notes
Audit logs	The audit data from SaaS is of little significance to SM.	
Events	Historical data	
Inspections	Re-run to generate the latest report.	
Integrations	Deprecated features	

Table 3: Resources that will not be migrated.

Resource Inventory

The first step of migration is to create an inventory of the resources in use in the source environment to prepare the correct processes for extraction. Refer to the TMC API categories, list all the migratable TMC resources in the source instance and identify what migration strategies are applicable for those resources based on the characteristics of the resources and the corresponding management mechanisms. Also, consider the resource dependencies and parent ownership for the next migration executions.

The available migration strategies include the following:

- **Cluster LCM:** transfer the management state of the cluster based on the applicable TMC Lifecycle Management (LCM) mechanism for the cluster.
- **Recreate on SM**: the resources can be recreated on the TMC-SM instance with the processed data copied from TMC-SaaS.
- **Manage back**: the resources are created on the physical clusters and can be managed back when the clusters are onboarded to TMC-SM.
- **Replacement on SM**: the resources are created on the physical clusters and cannot be managed back when the clusters are onboarded to TMC-SM. To restore the managed status, the existing resources on the physical clusters must be cleaned up before reinstalling the resources.
- **User operation needed**: resource recovery requires user intervention because there is some sensitive or encrypted data involved.

Resource	Migration Strategy	Dependencies	Parent Ref	Data Exporting Guide	Data Reapplying Guide	Notes
TKG Clusters	Cluster LCM	• Cluster Group • Proxy/LIR objects	• Cluster Group • Provisioner	ES-30	ES-48	Management state transfer: WC:unmanage/MC:unregister -> MC:register/WC:manage
Attach Clusters	Cluster LCM	• Cluster Group • Proxy/LIR objects	• Cluster Group • Provisioner (attached)	ES-31	ES-49	Management state transfer: WC:detach -> WC:attach
Cluster Group	Recreate on SM	N/A	Organization	ES-02	ES-34	
Workspace	Recreate on SM	N/A	Organization	ES-03	ES-35	

Role	Recreate on SM	N/A	Organization	ES-04	ES-36	
Credential	User Operation needed	N/A	Organization	ES-05	ES-37	Only support: 1.TMC provisioned storage - AWS S3 2. Self-provisioned storage - AWS S3 or S3 protocol compatible - Azure blob Users provide credentials and CA
Access (Administration)	Recreate on SM	Credential - Proxy - Image Registry	Credential	ES-06	ES-60	Access is under the Administration page.
Proxy	User Operation needed	N/A	Organization	ES-07	ES-38	Users provide credentials and CA
Image Registry	User Operation needed	N/A	Organization	ES-08	ES-39	Users provide credentials and CA
Setting	Recreate on SM	• Organization • Cluster Group • Cluster	Organization	ES-09	ES-59	
Kubernetes secrets in Cluster Group	User Operation needed	Cluster Group	Cluster Group	ES-10	ES-40	
Kubernetes secrets export in Cluster Group	User Operation needed	Cluster Group	Cluster Group	ES-11	ES-41	
Continuous Delivery in Cluster Group	Recreate on SM	Cluster Group	Cluster Group	ES-12	ES-42	This resource enables Flux to be applied to the workload clusters within the cluster group
Repository credentials in Cluster Group	User Operation needed	Cluster Group	Cluster Group	ES-13	ES-43	
Git repository in Cluster Group	Recreate on SM	• Cluster Group • Continuous Delivery • Repository credentials	Cluster Group	ES-14	ES-44	
Kustomization in Cluster Group	Recreate on SM	• Cluster Group • Git repository	Cluster Group	ES-15	ES-45	
Helm in Cluster Group	Recreate on SM	Cluster Group	Cluster Group	ES-16	ES-46	This resource enables the Helm server to be applied to the target workload clusters within the cluster group
Helm release in Cluster Group	Recreate on SM	• Cluster Group • Git repository	Cluster Group	ES-17	ES-47	

Managed Namespace	Recreate on SM	Workspace	Cluster	ES-18	ES-50	
Kubernetes secrets in Cluster	User Operation needed	Cluster	Cluster	ES-19	ES-51	
Kubernetes secrets export in Cluster	User Operation needed	Cluster	Cluster	ES-20	ES-52	
Continuous Delivery in Cluster	Recreate on SM	Cluster	Cluster	ES-21	ES-53	This resource enables Flux to be applied to the target workload cluster
Repository credentials in Cluster	User Operation needed	• Cluster • Continuous Delivery	Cluster	ES-22	ES-54	
Git repository in Cluster	Recreate on SM	• Cluster • Continuous Delivery • Repository credentials	Cluster	ES-23	ES-55	
Kustomization in Cluster	Recreate on SM	• Cluster • Git repository	Cluster	ES-24	ES-56	
Helm in Cluster	Recreate on SM	Cluster	Cluster	ES-25	ES-57	This resource enables the Helm server to be applied to the target workload cluster
Helm release in Cluster	Recreate on SM	• Cluster • Git repository	Cluster	ES-26	ES-58	
Data Protection	Recreate on SM	• Organization • Cluster	Organization	ES-27	ES-64	
Access Policies	Recreate on SM	• Cluster Group • Cluster • Workspace • Namespace • Management cluster • Provisioner	Organization	ES-28	ES-61	
Policies Templates	Recreate on SM	• N/A		ES-29	ES-62	
Policies Assignments	Recreate on SM	• Policies Template • Cluster Group • Cluster • Workspace	Organization	ES-30	ES-63	

Tanzu repository in Cluster	Manage back	Cluster	Cluster	N/A	N/A	Nothing to do
Installed Tanzu Packages in Cluster	Manage back	Cluster	Cluster	N/A	N/A	Nothing to do

Table 4: Resources and migration strategies.

Migration Process

Migration Process

Step 1: Connect to SaaS

Execution Serial: ES-01

CLI | Connect to TMC-SaaS with the following script, after supplying values for the required variables listed at the top. Call the resource APIs to do operations with the access token.

SCRIPT: 001-base-saas_stack-connect.sh

Step 2: Export Resource Data

Follow the guide below to export each of the migratable resource data types from the TMC-SaaS instance to the local filesystem. Only the resources in use (as identified in the Resource Inventory) need to be exported.

Basic Resources

***Resource**: Cluster Group*

| **Strategy**: Recreate on SM | **Use tools**: CLI | **Output**: clustergroup/data/clustergroups.yaml | **Execution Serial**: ES-02

SCRIPT: 002-base-clustergroups-export.sh

***Resource**: Workspace*

| **Strategy**: Recreate on SM | **Use tools**: CLI | **Output**: workspace/data/workspaces.yaml | **Execution Serial**: ES-03

SCRIPT: 003-base-workspaces-export.sh

Administration Resources

***Resource**: Role*

| **Strategy**: Recreate on SM | **Use tools**: CLI | **Output**: role/data/roles.yaml | **Execution Serial**: ES-04

SCRIPT: 004-admin-roles-export.sh

***Resource**: Credential*

| **Strategy**: Recreate on SM | **Use tools**: CLI | **Output**: credential/data/credentials.yaml | **Execution Serial**: ES-05

SCRIPT: 005-admin-credentials-export.sh

Resource: Access (Under Administration)
| Strategy: Recreate on SM | Use tools: CLI | Output: credential-access/data/access---*.yaml | Execution Serial: ES-06

SCRIPT: 006-admin-access-export.sh

Resource: Proxy
| Strategy: Recreate on SM | Use tools: CLI | Output: proxy/data/proxies.yaml | Execution Serial: ES-07

SCRIPT: 007-admin-proxy-export.sh

Resource: ImageRegistry
| Strategy: Recreate on SM | Use tools: CLI | Output: image-registry/data/image-registries.yaml | Execution Serial: ES-08

SCRIPT: 008-admin-image-registry-export.sh

Resource: Settings
| Strategy: Recreate on SM | Use tools: CLI | Output: setting/data/$scope/settings.yaml | Execution Serial: ES-09

SCRIPT: 009-admin-settings-export.sh

Cluster Group Addon Resources

Resource: Kubernetes secrets in Cluster Group
| Strategy: Recreate on SM | Use tools: CLI API | Output: cluster_group_secrets/*.yaml | Execution Serial: ES-10

SCRIPT: 010-clustergroup-secrets-export.sh

Resource: Kubernetes secrets Export in Cluster Group
| Strategy: Recreate on SM | Use tools: CLI API | Output: cluster_group_secret_exports/*.yaml | Execution Serial: ES-11

SCRIPT: 011-clustergroup-secret-exports-export.sh

Resource: Continuous Delivery in Cluster Group
| Strategy: Recreate on SM | Use tools: CLI API | Output: cluster_group_continuousdelivery/*.yaml | Execution Serial: ES-12

SCRIPT: 012-clustergroup-continuous-deliveries-export.sh

Resource: Repository credentials in Cluster Group
| Strategy: Recreate on SM | Use tools: CLI | Output: cluster_group_repo_secrets/*.yaml | Execution Serial: ES-13

SCRIPT: 013-clustergroup-repository-credentials-export.sh

Resource: Git Repository in Cluster Group
| Strategy: Recreate on SM | Use tools: CLI | Output: cluster_group_git_repo/*.yaml | Execution Serial: ES-14

SCRIPT: 014-clustergroup-git-repositories-export.sh

Resource: Kustomization in Cluster Group
| Strategy: Recreate on SM | Use tools: CLI | Output: cluster_group_kustomization/*.yaml | Execution Serial: ES-15

SCRIPT: 015-clustergroup-kustomizations-export.sh

Resource: Helm in Cluster Group
| Strategy: Recreate on SM | Use tools: CLI | Output: cluster_group_helm/*.yaml | Execution Serial: ES-16

SCRIPT: 016-clustergroup-helms-export.sh

Resource: Helm Release in Cluster Group
| Strategy: Recreate on SM | Use tools: CLI | Output: cluster_group_helm_release/*.yaml | Execution Serial: ES-17

SCRIPT: 017-clustergroup-helm-releases-export.sh

Cluster Addon Resources

Resource: Managed namespace
| Strategy: Manage back | Use tools: CLI | Output: namespaces/*.yaml | Execution Serial: ES-18

SCRIPT: 018-cluster-namespaces-export.sh

Resource: Kubernetes secrets in Cluster
| Strategy: Recreate on SM | Use tools: CLI API | Output: cluster_secrets/*.yaml | Execution Serial: ES-19

SCRIPT: 019-cluster-secrets-export.sh

Resource: Kubernetes secrets Export in Cluster
| Strategy: Recreate on SM | Use tools: CLI API | Output: cluster_secret_exports/*.yaml | Execution Serial: ES-20

SCRIPT: 020-cluster-secret-exports-export.sh

Resource: Continuous Delivery in Cluster
| Strategy: Recreate on SM | Use tools: CLI | Output: cluster_continuousdelivery/*.yaml | Execution Serial: ES-21

SCRIPT: 021-cluster-continuous-deliveries-export.sh

Resource: Repository credentials in Cluster
| Strategy: Recreate on SM | Use tools: CLI | Output: cluster_repo_secrets/*.yaml | Execution Serial: ES-22

SCRIPT: 022-cluster-repository-credentials-export.sh

Resource: Git Repository in Cluster

| Strategy: Recreate on SM | Use tools: CLI | Output: cluster_git_repo/*.yaml | Execution Serial: ES-23

SCRIPT: 023-cluster-git-repositories-export.sh

Resource: Kustomization in Cluster

| Strategy: Recreate on SM | Use tools: CLI | Output: cluster_kustomization/*.yaml | Execution Serial: ES-24

SCRIPT: 024-cluster-kustomizations-export.sh

Resource: Helm in Cluster

| Strategy: Recreate on SM | Use tools: CLI | Output: cluster_helm/*.yaml | Execution Serial: ES-25

SCRIPT: 025-cluster-helms-export.sh

Resource: Helm Release in Cluster

| Strategy: Recreate on SM | Use tools: CLI | Output: cluster_helm_release/*.yaml | Execution Serial: ES-26

SCRIPT: 026-cluster-helm-releases-export.sh

Data Protection

Resource: Data-protection

| Strategy: Recreate on SM | Use tools: CLI API | Output: data-protection-data/schedule.yaml data-protection-data/backup.yaml data-protection-data/restore.yamldata-protection-data/dataprotection_clustergroups.yaml data-protection-data/dataprotection_clusters.yaml | Execution Serial: ES-27

SCRIPT: 027-cluster-data_protection-export.sh

Access Policies

Resource: Access Policies

| Strategy: Recreate on SM | Use tools: API | Execution Serial: 1 | Output: iam/* | Execution Serial: ES-28

SCRIPT: 028-base-access-policies-export.sh

Policies

Resource: Policies Templates

| Strategy: Recreate on SM | Use tools: CLI | Execution Serial: 1 | Output: policies/* | Execution Serial: ES-29

SCRIPT: 029-base-policy-templates-export.sh

Resource: Policies Assignments

| Strategy: Recreate on SM | Use tools: CLI | Execution Serial: 1 | Output: policies/* | Execution Serial: ES-30

SCRIPT: 030-base-policy-assignments-export.sh

Step 3: Offboard Clusters from SaaS

After completing the data export, the clusters can be offboarded from the SaaS platform by following the steps outlined below.

Managed TKGm/VKS clusters

Execution Serial: ES-31

Export all the available management clusters and all the managed workload clusters under each management cluster with the following command.

Output: clusters/mc_list.yaml, clusters/wc_of_$name.yaml

SCRIPT: 031-base-managed_clusters-export.sh

Deregister the management clusters after unmanaging all the workload clusters under them. Management cluster deregistration can only be triggered after all the managed clusters have been successfully unmanaged. Otherwise, the deregistration request will be rejected.

SCRIPT: 031-base-managed_clusters-offboard.sh

Attached clusters

Execution Serial: ES-32

Export all the available attached clusters with the following command.

| **Output**: clusters/attached_clusters.yaml

SCRIPT: 032-base-attached_clusters-export.sh

With the exported data, detach all the attached clusters with the following command. Unhealthy attached clusters will try to be forcefully detached.

Input: clusters/attached_clusters.yaml

SCRIPT: 032-base-attached_clusters-offboard.sh

Step 4: Connect to the TMC-SM Instance

| **Execution Serial**: ES-33

Before performing the data import operation, it is necessary to establish a connection with the TMC-SM instance by following the steps outlined below.

CLI | Connect to the TMC-SM instance with the following command:

SCRIPT: 033-base-sm_stack-connect.sh

API | Get the access token with the following command. | **Output**: sm-api-call.sh

SCRIPT: sm-api-call.sh

```
# Source sm-api-call.sh
# Curl the TMC API when necessary
curl_api_call -X POST -d '{"data": "ok"}' <api_path>
curl_api_call -X PUT -d '{"data": "ok"}' <api_path>
curl_api_call <api_path>
# e.g.
curl_api_call v1alpha1/clustergroups
```

Step 5: Import Resource Data – Pre cluster onboarding

Follow the instructions below to recreate the applicable resources on the TMC-SM instance in the specified order (the order provided in this document). These resources are essential for successfully onboarding clusters to TMC-SM.

Basic Resources

***Resource**: Cluster Group*
Strategy: Recreate on SM | **Use tools**: CLI | **Input**: clustergroup/data/clustergroups.yaml | **Execution Serial**: ES-34

SCRIPT: 034-base-clustergroups-import.sh

***Resource**: Workspace*
Strategy: Recreate on SM | **Use tools**: CLI | **Input**: workspace/data/workspaces.yaml | **Execution Serial**: ES-35

SCRIPT: 035-base-workspaces-import.sh

Administration Resources

***Resource**: Role*
Strategy: Recreate on SM | **Use tools**: CLI | **Input**: role/data/roles.yaml | **Execution Serial**: ES-36

SCRIPT: 036-admin-roles-import.sh

***Resource**: Credential*
Strategy: Recreate on SM | **Use tools**: CLI | **Input**: credential/data/credentials.yaml | **Execution Serial**: ES-37

Credential Migration needs users to provide **Credentials** and related resources.
Two-phase scripts:

1. The first script will generate template files under the folder: credential/template.
   Users must then fill in the missing fields such as Certificate Authorities (CAs) and credentials.
2. The second script will create resources on TMC-SM based on the template files.

The template files generation script, produces: credential/template/*.yaml

---

SCRIPT: 037-admin-credentials-create-template.sh

Please Note: ***Check each file in the template folder, and fill in the missing credentials***

Field Name	Type	Format
spec.data.keyValue.data (**Self-provisioned: AWS S3 or S3-protocol compatible. The provider AWS_EC2 is not supported on TMC SM, use this template instead**)		
aws_access_key_id	string	base64
aws_secret_access_key	string	base64
spec.data.azureCredential.servicePrincipal (**Self-provisioned: Azure Blob**)		
subscriptionId	string	
tenantId	string	
resourceGroup	string	
clientId	string	
clientSecret	string	
azureCloudName	string	<AzurePublicCloud | AzureChinaCloud | AzureGermanCloud>

Credential resource recreation **script**

SCRIPT: 037-admin-credentials-import.sh

Resource: Proxy

Strategy: Recreate on SM | **Use tools**: CLI | **Input**: proxy/data/proxies.yaml | **Execution Serial**: ES-38

Proxy Migration needs users to provide **Credentials, CA and associated details**

Two-phase scripts:

1. The first script will generate template files under the folder: proxy/template.
   Then users need to fill in the missing fields such as CA, proxy credentials.
2. The second script will create resources on SM based on the template files.

The template files generation script:

SCRIPT: 038-admin-proxy-create-template.sh

Please Note: ***Check each file in the template folder, and fill in with the missing credentials and CA***

Field Name	Type	Format
spec.data.keyValue.data		
httpUserName	string	base64
httpPassword	string	base64
httpsUserName	string	base64
httpsPassword	string	base64
proxyCABundle	string	base64

Proxy resource recreation script

SCRIPT: 038-admin-proxy-import.sh

Resource: Image Registry
| **Strategy**: Recreate on SM | **Use tools**: CLI | **Execution Serial**: 1 | **Input**: image-registry/data/image-registries.yaml | **Execution Serial**: ES-39

Image Registry Migration requires users to provide **Credentials and CA** information.
Two-phase scripts:

1. The first script will generate template files under the folder: image-registry/template.
   Then users need to fill in the missing fields such as CA, credentials.
2. The second script will create resources on SM based on the template files.

The template files generation script:

SCRIPT: 039-admin-image-registry-create-template.sh

Please Note: *Check each file in the template folder, and fill in with the missing credentials and CA*

Field Name	Type	Format
spec.data.keyValue.data		

.dockerconfigjson	string	Base64
		Example script to generate base64 string:
		`accessId=${1-}`
		`accessSecret=${2-}`
		`registryUrl=${3-}`
		`data="{\"auths\":{\"$registryUrl\":{\"username\":\"$accessId\",\"password\":\"$accessSecret\"}}}"`
		`base64DockerConfigJsonString=`**`echo`** `$data
			`**`base64`**``

ca-cert	string	Base64

Image Registry resource recreation script

SCRIPT: 039-admin-image-registry-import.sh

Cluster Group Addon Resources

Resource: Kubernetes secrets in Cluster Group
Strategy: Recreate on SM | **Use tools**: CLI | **Input**: cluster_group_secrets/*.yaml | **Execution Serial**: ES-40

Kubernetes Secrets Migration requires users to fill secret data for the YAML files in cluster_group_secrets with the following template secrets

SECRET_TYPE_OPAQUE

```
spec:
  atomicSpec:
    data:
      key: base64-encoded-value
    secretType: SECRET_TYPE_OPAQUE
```

SECRET_TYPE_DOCKERCONFIGJSON

```
spec:
  atomicSpec:
    data:
      .dockerconfigjson: base64-encoded-dockerconfig-json-file
    secretType: SECRET_TYPE_DOCKERCONFIGJSON
```

After the secret data is filled, use the following script to recreate them

---

SCRIPT: 040-clustergroup-secrets-import.sh

***Resource**: Kubernetes secrets Export in Cluster Group*
| **Strategy**: Recreate on SM | **Use tools**: CLI | **Input**: cluster_group_secret_exports/*.yaml | **Execution Serial**: ES-41

SCRIPT: 041-clustergroup-secret-exports-import.sh

***Resource**: Continuous Delivery in Cluster Group*
Strategy: Recreate on SM | **Use tools**: CLI | **Input**: cluster_group_continuousdelivery/*.yaml | **Execution Serial**: ES-42

SCRIPT: 042-clustergroup-continuous-deliveries-import.sh

***Resource**: Repository credentials in Cluster Group*
Strategy: Recreate on SM | **Use tools**: CLI | **Input**: cluster_group_repo_secrets/*.yaml | **Execution Serial**: ES-43

Repository Credentials Migration requires users to fill secret data for the YAML files in cluster_group_repo_secrets with the following template secrets

Username/Password

```
spec:
  atomicSpec:
    data:
      data:
        username: bas64-encoded-username
        password: base64-encoded-password
    sourceSecretType: USERNAME_PASSWORD
```

SSH Authentication

```
spec:
  atomicSpec:
    data:
      data:
        identity: bas64-encoded-ssh-identity
        known_hosts: base64-encoded-ssh-known-hosts
    sourceSecretType: SSH
```

CA Certificate

```
spec:
  atomicSpec:
    data:
      data:
        ca.crt: bas64-encoded-ca-crt
        username: base64-encoded-username  # username and password are optional
        password: base64-encoded-password  # username and password are optional
    sourceSecretType: CACert
```

After the secret data is filled, use the following script to recreate them

---

SCRIPT: 043-clustergroup-repository-credentials-import.sh

***Resource**: Git Repository in Cluster Group*

| **Strategy**: Recreate on SM | **Use tools**: CLI | **Input**: cluster_group_git_repo/*.yaml | **Execution Serial**: ES-44

SCRIPT: 044-clustergroup-git-repositories-import.sh

***Resource**: Kustomization in Cluster Group*

| **Strategy**: Recreate on SM | **Use tools**: CLI | **Input**: cluster_group_kustomization/*.yaml | **Execution Serial**: ES-45

SCRIPT: 045-clustergroup-kustomizations-import.sh

***Resource**: Helm in Cluster Group*

| **Strategy**: Recreate on SM | **Use tools**: CLI | **input**: cluster_group_helm/*.yaml | **Execution Serial**: ES-46

SCRIPT: 046-clustergroup-helms-import.sh

***Resource**: Helm Release in Cluster Group*

Strategy: Recreate on SM | **Use tools**: CLI | **Input**: cluster_group_helm_release/*.yaml | **Execution Serial**: ES-47

SCRIPT: 047-clustergroup-helm-releases-import.sh

Step 6: Onboard Clusters to SM

The clusters can now be onboarded to the TMC-SM instance by following the steps outlined below.

TKGm/VKS Clusters

Execution Serial: ES-48

The registration operations rely on the admin kubeconfig of the clusters. To fully automate the registration process for multiple clusters, an admin kubeconfig index file can be created first with the below command. After generating the index file, replace the path placeholders with the real paths of the kubeconfig files of the corresponding clusters.

| **Input**: clusters/mc_list.yaml | **Output**: clusters/mc-kubeconfig-index-file

SCRIPT: 048-base-managed_clusters-input_from_user.sh

With the exported YAML file that includes the clusters ready to be onboarded, run the following command to register the clusters to TMC-SM.

- The admin kubeconfigs of the management clusters are required for full automation.

Input: clusters/mc_list.yamlclusters/mc-kubeconfig-index-file | **Output**: clusters/mc_registered.txt | **Dependent resources**: Proxy object LIR object (TKGm-only)

Run the following script to do some pre-checks and ensure the configurations and agents from TMC SaaS are cleaned up.
SCRIPT: 048-base-managed_clusters-ensure-cleanup.sh

> **Notes:**
>
> It must NOT be rerun after running 048-base-managed_clusters-onboard.sh, since agents could be cleaned up.

SCRIPT: 048-base-managed_clusters-onboard.sh

> **Notes:**
>
> - Here, only register the management clusters with a HEALTHY status at export time. If an unhealthy management cluster has been restored to HEALTHY, the condition `"$health" == "HEALTHY"` in the script can be removed or dealt with separately.
> - If a management cluster with the same name is detected on TMC-SM, reregister will be used with the assumption that it is the same cluster.
> - It will trigger cluster rolling update when the cluster is being managed to TMC SM, the cluster nodes will be replaced one by one.

> **Notes:**
>
> Two variables can be set to control the onboarding cluster process:
>
> - CLUSTERS_ONBOARD_BATCH_SIZE: set the concurrency of onboarding clusters, default: 1.
> - CLUSTERS_ONBOARD_TIMEOUT: the timeout of onboarding a cluster, default 10 mins. It depends on the performance of underlying infrastructure and the node counts, it can be accumulated approximately 3 mins per node.

Clusters to be Attached
Execution Serial: ES-49

The attach operations rely on the kubeconfigs of the clusters. To support fully automated batch attach operations for the exported clusters, a kubeconfig index file can be created with the below command first. After generating the index file, replace the path placeholder with real paths of the kubeconfig files of the corresponding clusters.

Input: clusters/attached_clusters.yaml | **Output**: clusters/attached-wc-kubeconfig-index-file

SCRIPT: 049-base-attached_clusters-input_from_user.sh

Attach the healthy clusters to the TMC-SM instance from the exported data file with the following command.

| Input: clusters/attached_clusters.yaml clusters/attached-wc-kubeconfig-index-file| Output: clusters/onboared-clusters-name-index| Dependent resources: Proxy object LIR object

SCRIPT: 049-base-attached_clusters-onboard.sh

Run the following script to check the readiness of all the onboarded clusters.
SCRIPT: 049-base-whole_clusters-check_readiness.sh

Step 7: Import Resource Data – Post cluster onboarding

After completing the cluster onboarding process, follow the steps below to restore the relevant resources to the clusters.

Cluster Addon Resources

Resource: Managed namespace

| Strategy: Recreate on SM | Use tools: CLI | Input: namespaces/*.yaml | Execution Serial: ES-50

SCRIPT: 050-cluster-namespaces-import.sh

Resource: Kubernetes secrets in Cluster

| Strategy: Recreate on SM | Use tools: CLI | Input: cluster_secrets/*.yaml | Execution Serial: ES-51

Kubernetes Secrets Migration needs users to fill secret data for the yaml files in cluster_group_secrets with the following template secrets

SECRET_TYPE_OPAQUE

```
spec:
  data:
    key: base64-encoded-value
  secretType: SECRET_TYPE_OPAQUE
```

SECRET_TYPE_DOCKERCONFIGJSON

```
spec:
  data:
    .dockerconfigjson: base64-encoded-dockerconfig-json-file
  secretType: SECRET_TYPE_DOCKERCONFIGJSON
```

After the secret data is filled, use the following script to recreate them

SCRIPT: 051-cluster-secrets-import.sh

Resource: Kubernetes secrets Export in Cluster

Strategy: Recreate on SM | Use tools: CLI | Input: cluster_secret_exports/*.yaml | Execution Serial: ES-52

SCRIPT: 052-cluster-secret-exports-import.sh

Resource: *Continuous Delivery in Cluster*
| **Strategy**: Recreate on SM | **Use tools**: CLI | **Input**: cluster_continuousdelivery/*.yaml | **Execution Serial**: ES-53

SCRIPT: 053-cluster-continuous-deliveries-import.sh

Resource: *Repository credentials in Cluster*
Strategy: Recreate on SM | **Use tools**: CLI | **Input**: cluster_repo_secrets/*.yaml | **Execution Serial**: ES-54

Repository Credentials Migration requires users to fill secret data for the YAML files in cluster_repo_secrets with the following template secrets

Username/Password

```
spec:
  data:
    data:
      username: bas64-encoded-username
      password: base64-encoded-password
  sourceSecretType: USERNAME_PASSWORD
```

SSH Authentication

```
spec:
  data:
    data:
      identity: bas64-encoded-ssh-identity
      known_hosts: base64-encoded-ssh-known-hosts
  sourceSecretType: SSH
```

CA Certificate

```
spec:
  data:
    data:
      ca.crt: bas64-encoded-ca-crt
      username: base64-encoded-username  # username and password are optional
      password: base64-encoded-password  # username and password are optional
  sourceSecretType: CACert
```

After the secret data is filled, use the following script to recreate them

SCRIPT: 054-cluster-repository-credentials-import.sh

Resource: *Git Repository in Cluster*
Strategy: Recreate on SM | Use tools: CLI | Input: cluster_git_repo/*.yaml | Execution Serial: ES-55

SCRIPT: 055-cluster-git-repositories-import.sh

Resource: *Kustomization in Cluster*
Strategy: Recreate on SM | Use tools: CLI | Input: cluster_kustomization/*.yaml | Execution Serial: ES-56

SCRIPT: 056-cluster-kustomizations-import.sh

Resource: *Helm in Cluster*
Strategy: Recreate on SM | Use tools: CLI | Input: cluster_helm/*.yaml | Execution Serial: ES-57

SCRIPT: 057-cluster-helms-import.sh

Resource: *Helm Release in Cluster*
Strategy: Recreate on SM | Use tools: CLI | Input: cluster_helm_release/*.yaml | Execution Serial: ES-58

SCRIPT: 058-cluster-helm-releases-import.sh

Administration Resources

Resource: *Settings*
Strategy: Recreate on SM | Use tools: CLI | Input: setting/data/$scope/settings.yaml | Execution Serial: ES-59

SCRIPT: 059-admin-settings-import.sh

Resource: *Access*
| Strategy: Recreate on SM | Use tools: CLI | Input: credential-access/data/access---*.yaml | Execution Serial: ES-60

SCRIPT: 060-admin-access-import.sh

Access Policies

Resource: *Access Policies*
| Strategy: Recreate on SM | Use tools: API | Input: iam/* | Execution Serial: ES-61

Notes:

- Variables ADMIN_IDP_GROUP and MEMBER_IDP_GROUP MUST be set consistently with the configuration values idpGroupRoles.admin and idpGroupRoles.member of TMC SM deployment.
- User and group assignments remain the same in TMC-SM. It may be necessary to determine equivalent users and user groups in the TMC-SM environment and configure them again from the UI manually after importing policies.

Access policies must be loaded in three phases; a base access load which populates the role bindings for organizations, cluster groups, and workspaces, then a second phase load for cluster role bindings, at last clean up SaaS left resources and resync the role bindings and cluster role binding to clusters.

Phase 1: Base Access Policies

SCRIPT: 061-base-access-policies-import.sh

Phase 2: Cluster Access Policies

SCRIPT: 061-cluster-access-policies-import.sh

Phase 3: Resync Cluster Access Polices

SCRIPT: 061-cluster-access-policies-resync.sh

Policies

Resource: Policies Templates
| **Strategy**: Recreate on SM | **Use tools**: CLI | **Input**: policies/* | **Execution Serial**: ES-62

SCRIPT: 062-base-policy-templates-import.sh

Resource: Policies Assignments
| **Strategy**: Recreate on SM | **Use tools**: CLI | **Input**: policies/* | **Execution Serial**: ES-63

Policy assignments must be loaded in two phases; a base policy load which populates policies for organizations, cluster groups, and workspaces, then a second phase load for cluster policies.

Phase 1: Base Policies

SCRIPT: 063-base-policy-assignments-import.sh

Phase 2: Cluster Policies

SCRIPT: 063-cluster-policy-assignments-import.sh

Data Protection

Resource: Data-protection
Strategy: Recreate on SM | **Use tools**: CLI | **Input**: data-protection-data/schedule.yaml data-protection-data/dataprotection_clustergroups.yaml data-protection-data/restore.yaml | **Dependent resources**: Storage Credential | **Execution Serial**: ES-64

SCRIPT: 064-cluster-data_protection-import.sh

> **Notes:**
> Credentials must be set up before importing.

Risks and Considerations

- The migration process will trigger rolling updates for the cluster managed by TMC while onboarding to TMC SM, the cluster nodes will be replaced one by one, which might cause transient downtime for the application workloads running on those clusters.
- The scripts listed in the documentation—for exporting TMC resources, offboarding the cluster, onboarding the cluster, and importing TMC resources—can be retried if any errors or failures occur during execution, until the process completes successfully. In some specific corner cases, execution may fail due to the presence of stale or dirty data causing conflicts. In such cases, manual intervention may be required to clean up the conflicting stale or dirty data before retrying the execution.
- TMC resource data will be exported into specific files, which essentially serve as API-based data backups. If these files are properly preserved and maintained, data loss should not occur.
- Access policies contain identity information for target users or groups, which is tied to the configured Identity Provider (IDP). If the IDP configured in the TMC-SM environment does not include this specific identity information, the imported access policies will not take effect. In such cases, you will need to manually update, adjust, or recreate the access policies to achieve the same access control capabilities as in the SaaS environment.

Appendix A: Support Library Scripts

The following supporting scripts contain common functions used by the scripts provided in this document. The scripts are intended to live in a subdirectory named `utils` under the directory in which you run these scripts.

Source File: **`utils/common.sh`**
utils/common.sh

Source File: **`utils/create-docker-config-json-base64.sh`**
utils/create-docker-config-json-base64.sh

Source File: **`utils/kubeconfig.sh`**
utils/**kubeconfig.sh**

Source File: **`utils/log.sh`**
utils/log.sh

Source File: **`utils/offboard-clusters.sh`**
utils/offboard-clusters.sh

Source File: **`utils/policy-helper.sh`**
utils/policy-helper.sh

Source File: **`utils/saas-api-call.sh.sh`**
utils/saas-api-call.sh

Source File: **`utils/sm-api-call.sh`**
utils/sm-api-call.sh

Appendix B: FAQs

Log collections

Q: How can I get the current configuration values from the TMC SM deployment?

A: You can get the configuration of TMC SM deployment from the secret tanzu-mission-control-tmc-local-values in the TMC namespace.

```
kubectl -n tmc-local get secret tanzu-mission-control-tmc-local-values -
ojsonpath='{.data.values\.yaml}' | base64 -d
```

Q: How can I collect support bundles for TMC SM deployment?

A: You can collect the resource configurations and logs for TMC services within the kubeconfig context of the TMC SM cluster.

```
# Download the crashd CLI
wget https://github.com/vmware-tanzu/crash-
diagnostics/releases/download/v0.4.3/crashd_0.4.3_linux_amd64.tar.gz
tar xvfz crashd_0.4.3_linux_amd64.tar.gz

# Download diagnostics.crsh
tmc-sm-installer/dependencies/imgpkg/imgpkg pull -i
projects.packages.broadcom.com/vsphere/tmc-sm/additional-resources:1.0.0 -o
/tmp/tmc-sm-support-bundle

mkdir /tmp/tmc-sm-logs
./crashd run --args="outdir=/tmp/tmc-sm-logs" /tmp/tmc-sm-support-bundle/crashd/diagnostics.crsh

# Copy the log bundle from /tmp/tmc-sm-logs/tmc-log-dump.tar.gz
```

Q: How can I collect the support bundles for the workload clusters and the supervisor cluster?

A: You can follow the KB article to collect the support bundle for the supervisor cluster and workload cluster: https://knowledge.broadcom.com/external/article/345464/gathering-logs-for-vsphere-with-tanzu.html.

Q: How can I execute the migration scripts with more debugging information?

A: You can execute the migration script with **-x** argument, it will show what each step of the shell script does and it failed in which step/which command, it would be helpful to manually check for a specific issue. e.g bash -x ./001-base-saas_stack-connect.sh

TMC SM Provisioning

Q: How can I skip the prechecks while using tmc-sm CLI to install or update the TMC SM deployment?

A: You can set the environment variable **SKIP_PRE_CHECK** to be "**true**" to skip the precheck while using **tmc-sm** CLI to install or update the TMC SM deployment.

For example, the tmc-sm 1.4.2 might panic when checking the available resources of the cluster in some unexpected saturation, you can set this variable to bypass the prechecks.

```
panic: cannot parse '': quantities must match the regular expression '^([+-]?[0-
9.]+)([eEinumkKMGTP]*[-+]?[0-9]*)$'
```

Q: Why do the TMC service pods(e.g auth-manager) fail to start with the error "tls: failed to verify certificate: x509: certificate signed by unknown authority"?

A: When your **trustedCAs** certificates might mismatch with the service certificates or the certificates are expired, the services will fail to start with the error like "**tls: failed to verify certificate: x509: certificate signed by unknown authority**"

example:

```
time="2024-09-13T06:43:38Z" level=error msg="Unable to retrieve metadata: Get \"https://pinniped-
supervisor.cn-tmc-pd-tanzu-ctl.example.local/provider/pinniped/.well-known/openid-configuration\":
tls: failed to verify certificate: x509: certificate signed by unknown authority" idp=oidc-pinniped
Error: Failed to get registered providers: failed to initialize IDP "pinniped", err=Error [117]:
OIDC Metadata Retrieval Error, detail: Get "https://pinniped-supervisor.cn-tmc-pd-tanzu-
ctl.example.local/provider/pinniped/.well-known/openid-configuration": tls: failed to verify
certificate: x509: certificate signed by unknown authority
```

You can validate the certificates with the command as below:

```
kubectl -n tmc-local get cm tls-ca-bundles -ojsonpath='{.data.letsencrypt\.pem}' > trusted_ca.crt
openssl s_client -connect https://pinniped-supervisor.cn-tmc-pd-tanzu-ctl.example.local:443 -
CAfile=trusted_ca.crt
```

In this case, you can update the TMC SM deployment with correct certificates.

Q: Why do the TMC service pods api-gateway fail to start with the "error reading server preface: remote error: tls: bad certificate"?

A: If the error message in the logs of the api-gateway pods shows below error to connect to gts.$TMC_FQDN,

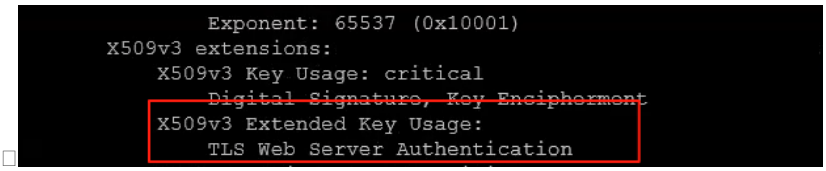
```
addrConn.createTransport failed to connect to {Addr: \"gts.tmc.lab.h2o-4-
23924.h2o.vmware.com:443\", ServerName: \"gts.tmc.lab.h2o-4-23924.h2o.vmware.com:443\", }. Err:
connection error: desc = \"error reading server preface: remote error: tls: bad certificate\""
subcomponent=grpc-runtime
```

this issue usually happens when the **extendedKeyUsage** is set to **serverAuth** only for the stack-tls certificate. It should be either unset or set to "**serverAuth, clientAuth**" since some service connections use mTLS. You can parse the certificates for **stack-tls** and **landing-service-tls** with below commands:

```
openssl s_client -showcerts -connect <TMC_FQDN>:443 </dev/null 2>/dev/null  | openssl x509 -text -
noout -in -

OR
kubectl -n tmc-local get secret stack-tls -ojsonpath='{.data.tls\.crt}' | base64 -d | openssl x509
-text -noout -in - | grep -A 1 'X509v3 Extended Key Usage'
kubectl -n tmc-local get secret landing-service-tls -ojsonpath='{.data.tls\.crt}' | base64 -d |
openssl x509 -text -noout -in - | grep -A 1 'X509v3 Extended Key Usage'

OR
openssl x509 -in $CERTIFICATE_FILE -text -noout
```



You should refer to the doc to regenerate the certificates **stack-tls** and **landing-service-tls** and set the **extendedKeyUsage = serverAuth, clientAuth** or unset it, then update the configuration of TMC SM.

Q: Why did the TMC service pods api-gateway fail to start with the errors "Failed to initialize org interceptor: rpc error: code = Unauthenticated desc = Missing access token"?

A: This issue usually happens when the api-gateway sync organization and tenancy information and tenancy-service failed to validate the CommonName in the subject of the client certificate, if the CommonName does match the TMC SM domain name, it rejected the API requests with the error "**code = Unauthenticated desc = Missing access token**".
You can parse the **stack-tls** certificate and check the CN in the subject should be the FQDN of TMC.

```
kubectl -n tmc-local get secret stack-tls -ojsonpath='{.data.tls\.crt}' | base64 -d | openssl x509 -text -noout -in -
----
Certificate:
    Data:
        Version: 3 (0x2)
        Serial Number:
            31:71:f7:7c:b1:65:bd:d7:2a:28:b8:d3:59:73:d0:77:f9:62:4f:4f
        Signature Algorithm: sha256WithRSAEncryption
        Issuer: C = US, ST = California, L = Palo Alto, O = "VMware, Inc.", OU = IT, CN = ca.tanzu.io
        Validity
            Not Before: Sep 11 11:52:25 2025 GMT
            Not After : Sep  9 11:52:25 2035 GMT
        Subject: C = US, ST = California, L = Palo Alto, O = "VMware, Inc.", OU = IT, CN = tmc.tanzu.com
```

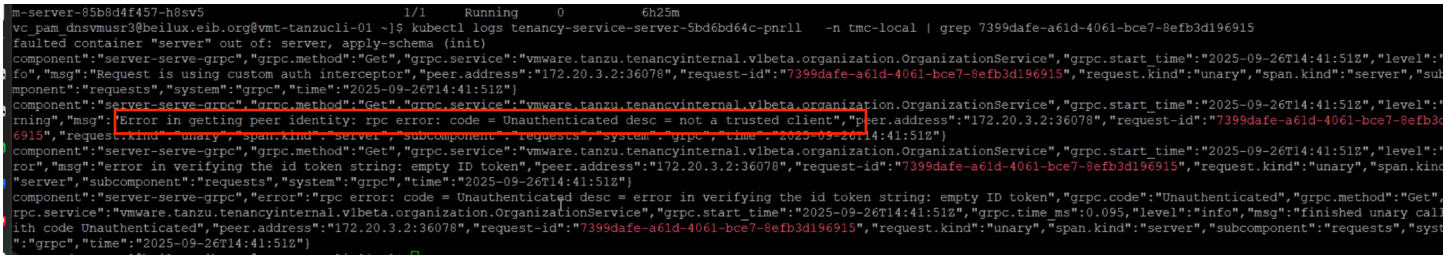
You should refer to the doc to regenerate certificates for stack-tls with correct CN.

Q: Why did the TMC UI fail with "errcode: 4030 errmsg: Internal Server Error requestid: xxx" while redirecting to the landing page?

A: You can search the errcode **4030** in the logs of the pods **landing-service** in tmc-local namespace if it has similar error as below or not:

{"component":"server-serve-http","http.host":"landing.tmc.example.org","http.proto_major":2,"http.request.length_bytes":0,"http.request.method":"GET","http.request.referer":"","http.request.user_agent":"Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/140.0.0.0 Safari/537.36","http.url.path":"/callback","level":"error","msg":"**errcode: 4030 errmsg: Internal Server Error requestid: 7399dafe-a61d-4061-bce7-8efb3d196915 cause: cannot exchange tenant id for org details**","peer.address":"172.20.7.3","peer.port":"47356","request-id":"7399dafe-a61d-4061-bce7-8efb3d196915","span.kind":"server","system":"http","time":"2025-09-26T14:41:51Z"}

Then check the logs of tenancy-service see if there is error message like "**code = Unauthenticated desc = not a trusted client"** as below or not



If you can see the above error message in above two services, this issue should be caused by an incorrect CommonName for the **landing-service** TLS certificate, you can inspect the certificate with below command to see if the CN in Subject is **landing.$TMC_FQDN** or not.

```
kubectl -n tmc-local get secret landing-service-tls -ojsonpath='{.data.tls\.crt}' | base64 -d |
openssl x509 -text -noout -in -
----
Certificate:
    Data:
        Version: 3 (0x2)
        Serial Number:
            31:71:f7:7c:b1:65:bd:d7:2a:28:b8:d3:59:73:d0:77:f9:62:4f:4f
        Signature Algorithm: sha256WithRSAEncryption
        Issuer: C = US, ST = California, L = Palo Alto, O = "VMware, Inc.", OU = IT, CN =
ca.tanzu.io
        Validity
            Not Before: Sep 11 11:52:25 2025 GMT
            Not After : Sep  9 11:52:25 2035 GMT
        Subject: C = US, ST = California, L = Palo Alto, O = "VMware, Inc.", OU = IT, CN =
landing.tmc.tanzu.com
```

This issue occurs when the tenancy-service checks the CommonName in the subject of the client certificate while the landing service get organization information from tenancy-service, if the CommonName does match the TMC SM domain name, it rejects the API requests with the error "**code = Unauthenticated desc = not a trusted client**".
You should refer to the doc to regenerate the certificate for **landing-service-tls** with correct CN.

Q: Why did the TMC SM UI fail to launch with the error "Fatal error loading application configuration (see console)."?

A: You can check the certificate configuration for TMC SM if using single TLS certificates for all services. Contour doesn't support HTTP/2 well when using a single TLS certificate, either with wildcard or all DNS, and it will cause redirects to access the wrong services. You should refer to the doc to regenerate a dedicated certificate for each DNS.

Authentication

Q: How should I configure pinniped-supervisor with an identity provider?

A: You can refer to https://pinniped.dev/docs/howto/supervisor/ to get hints on how to configure a specific identity provider to work with pinniped-supervisor

- With Auth0 OIDC
- With Azure AD
- With Dex OIDC
- With Entra ID
- With GitHub

- With Okta OIDC
- With Workspace ONE Access
- With GitLab OIDC
- With OpenLDAP
- With JumpCloud LDAP
- With Active Directory

Q: How can I sanity check the basic configuration issue when I failed to login to TMC UI?

A: You can check the status of IdentityProvider first, below command can show some basic configuration issue:

- ODIC: kubectl -n tmc-local describe OIDCIdentityProvider
- LDAP: kubectl -n tmc-local describe LDAPIdentityProvider
- ActiveDirectory: kubectl -n tmc-local describe ActiveDirectoryIdentityProvider

Q: How can I verify if the OIDC issuerURL is correct?

A: You can use the "**curl**" command to query the well-known configuration endpoint to validate the issuer URL. The well-known configuration endpoint looks like https://{{ISSUER FQDN}}/.well-known/openid-configuration. **Note** that the "**oidc.issuerURL**" in the TMC SM values file must **EXACTLY** match the "**issuer**" key in the configuration endpoint response. **Watch out for a trailing slash!** They should be exactly the same! should look like this example: https://auth0.com/docs/get-started/applications/configure-applications-with-oidc-discovery#sample-response

Q: Why can't I login to the UI with an error "Unauthorized requestid": errcode: 2004 errmsg: Unauthorized requestid: xxx""?

A: You can check the logs of landing-service pods if you can see below errors which complain the ID token is not found.

```
"errcode: 2004 errmsg: Unauthorized requestid: XXX  cause: id token not found"
```

Then, you can check the logs of auth-manager pods if you can see below error which complain the token is too long.

```
"Could not save idtokenCookie" X-Request-ID=XXX error="securecookie: the value is too long"
```

this is likely due to the large token size and requires additional configuration. We can see this if the user belongs to hundreds of groups.

- For LDAP, the group search query may need to be refined.
- For OIDC, the identity provider may allow a way to filter the response in the group claim

Q: Why can't I login to the UI with an error "3012 errmsg: Forbidden requestid: xxx"?

A: You can check the logs of **landing-service** pods if you can see below error which complain "**insufficient access: admin role not found**", this is likely because the login user is not in the group configured in "idpGroupRoles.admin". It's required that the first login user must be in the admin group in order to make some initialization.


```
{"component":"server-serve-
http","http.host":"landing.tmc.example.com","http.proto_major":2,"http.request.length_bytes":0,"htt
p.request.method":"GET","http.request.referer":"","http.request.user_agent":"Mozilla/5.0 (Windows
NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/140.0.0.0
Safari/537.36","http.url.path":"/callback","level":"error","msg":"errcode: 3012 errmsg: Forbidden
requestid: 449f1b89-fd1f-4e6c-9660-0a24e1b20074 cause: insufficient access: admin role not
found","peer.address":"192.168.2.3","peer.port":"36122","request-id":"449f1b89-fd1f-4e6c-9660-
0a24e1b20074","span.kind":"server","system":"http","time":"2025-10-22T20:59:39Z"}
```


Q: How can I get the username and groups of current login user in the token?

A: There are three approaches to get the current login user groups from the token:

1. You can inspect the token of login user from the secret with below command:

```
kubectl -n tmc-local get secret  --no-headers -o custom-columns=":metadata.name" | grep pinniped-storage-access-token | xargs -i kubectl -n tmc-local get secret {} -ojsonpath='{.data.pinniped-storage-data}' | base64 -d | jq .
```

2. You can inspect the token from the tmc context of tanzu CLI:

```
tanzu tmc context create --idp pinniped --endpoint <TMC SM INSTALL DOMAIN>:443 --basic-auth

# Once the login is completed tanzu context file is stored in ~/.config/tanzu/config-ng.yaml
# Decode the id token in the context using jwt.io
```

3. You can also enable the debug logs for pinniped supervisor to print out the current login username and groups:

```
# Need to pause the Pinniped package reconciliation so that your changes don't get overwritten
kctrl package installed pause \
  --package-install pinniped \
  --namespace tmc-local

kubectl edit configmap/pinniped-supervisor -n tmc-local

# Add the following highlighted lines
data:
  pinniped.yaml: |
    audit:
      logUsernamesAndGroups: enabled
# Save the file

kubectl rollout restart deployments/pinniped-supervisor \
  --namespace tmc-local

# Now perform a login while tailing the Pinniped Supervisor logs
```

Q: How can I debug LDAP filters?

A: You can use ldapsearch to debug LDAP filters with the script https://github.com/vmware/pinniped/blob/main/hack/debug-ldapidentityprovider.sh

Migration scripts

Q: Can I rerun the migration scripts multiple times?
A: Yes. Almost all the scripts support rerunning, except below two cluster offboarding scripts:

- 031-base-managed_clusters-offboard.sh
- 032-base-attached_clusters-offboard.sh

The data exporting scripts can be rerun multiple times before the cluster offboard from SaaS, and the configuration importing scripts also can be rerun on TMC SM. You can ignore the "**AlreadyExists**" error while rerunning the data importing scripts.

Q: Do I have to execute the migration scripts according to index number one by one?
A: No, you can also run the scripts as below sequence, offboard clusters after importing non-cluster level configuration to TMC SM:

1. Run the script 001, which will create a context migration to connect to TMC SaaS.
2. Run the scripts 002 to 030 or run export-all-from-saas.sh (which is nothing but script 002 to 030 at once) to export configurations from TMC SaaS.
3. Run the script 033 which will create a context to connect to TMC-SM, which will be used to import the configuration in TMC-SM.
4. Run scripts 034 to 047, which will import some configurations on non-cluster levels to TMC SM.
5. Run the script 001 again to switch context to TMC SaaS and then run both the scripts 031 to 032, it will unmanage workload/detach clusters and de-register management cluster from TMC SaaS.
6. Run the script 033 again to switch the context to TMC SM, and then run both the script 048 which will register the management cluster and also manage your workload cluster, the rolling update of your workload clusters will be triggered, transient downtime might be expected, the impact depends on your application characteristics.
7. Run the scripts 049, and remaining scripts 050 to 064 to complete the migration process.

Q: Do I have to migrate all the clusters from TMC SaaS to TMC SM at once?

A: No, for the managed clusters on different supervisor clusters, for example you have different supervisor clusters for dev, stage and prod, you can set the variable **TMC_MC_FILTER** to specify the supervisor cluster names to be migrated, e.g export TMC_MC_FILTER=”dev-sc1,stage-sc2”, while running the script 031 which will only offboard the clusters on the supervisor clusters specified by TMC_MC_FILTER.
For the attached cluster, you can also set the variable **CLUSTER_NAME_FILTER** to specify the cluster to be offboarded, e.g export CLUSTER_NAME_FILTER=”cls1,cls2”.

Q: Can I onboard the cluster manually without the scripts?

A: Yes, you can register the supervisor clusters, manage workload clusters, or attach workload clusters manually from the TMC UI, but you should make sure the cluster name and cluster group are the same as on SaaS, and fill in the onboarded cluster list in the file data/clusters/onboarded-clusters-name-index.

The format of onboarded-clusters-name-index:
<ManagementClusterName>.<ProvisionerName>.<ClusterName>

Example:
dev-mgmt-1.testns1.wc-01
dev-mgmt-1.testns1.wc-02
attached.attached.wc-03

Cluster onboarding

Q: Is the cluster rolling update expected when the cluster is being managed to TMC SM?

A: Yes, the cluster will be reconfigured with new certificates of TMC SM while it’s being managed by TMC SM. Even though the cluster has been configured with the certificate of Harbor registry, the rolling update is still not avoidable.

Q: What should I do if the UI shows "API Error: Failed to register management cluster with configurations: Internal Server Error: please try again later (internal error)" when registering a management cluster?

A: You can check the logs of cluster-agent-service pods if you can see below errors

```
{"component":"server-serve-grpc","error":"harbor.example.com/tmc-sm/tap-tmc-docker-virtual.usw1.packages.broadcom.com/extensions/tmc-bootstrapper/supervisor/manifest:20250609161419230-2a1e223a256b31a12a6b294df6011b8ff7c88f31 was not
```

```
fetched: could not load image source: get image from the registry: GET
https://harbor.example.com/v2/tmc-sm/tap-tmc-docker-
virtual.usw1.packages.broadcom.com/extensions/tmc-
bootstrapper/supervisor/manifest/manifests/20250609161419230-
2a1e223a256b31a12a6b294df6011b8ff7c88f31:
NOT_FOUND: repository tmc-sm/tap-tmc-docker-virtual.usw1.packages.broadcom.com/extensions/tmc-
bootstrapper/supervisor/manifest not
found","grpc.method":"Create","grpc.service":"vmware.tanzu.manage.v1alpha1.managementcluster.Manage
mentClusterResourceService","grpc.start_time":"2025-11-11T04:13:06Z","level":"error","msg":"failed
to get spec for tmc-bootstrapper","peer.address":"10.233.67.18:36252","request-id":"c37f8b3f-a235-
4b4d-9163-
3a1af6a1076d","request.kind":"unary","span.kind":"server","subcomponent":"requests","system":"grpc"
,"time":"2025-11-11T04:13:07Z","uid":"917c4179-2a1f-466d-96b6-a7acd2a79aa7"}
```

It means the image tap-tmc-docker-virtual.usw1.packages.broadcom.com/extensions/tmc-bootstrapper/supervisor/manifest with the tag 20250609161419230-2a1e223a256b31a12a6b294df6011b8ff7c88f31 is missing in the harbor project. It's possible happens in below two situations:

1. The TMC SM installer tarball is extracted on Windows syetem or WSL, the colon in extenion image file name is coverted to underscore, e.g tap-tmc-docker-virtual.usw1.packages.broadcom.com/extensions/tmc-bootstrapper/supervisor/manifest:20250609161419230-2a1e223a256b31a12a6b294df6011b8ff7c88f31 -> tap-tmc-docker-virtual.usw1.packages.broadcom.com/extensions/tmc-bootstrapper/supervisor/manifest_20250609161419230-2a1e223a256b31a12a6b294df6011b8ff7c88f31, you should extract the tarbal on Linux system and re-push the packages again with CLI "tmc-sm".
2. The image is missing on the Harbor registry, you should re-push the packages again with CLI "tmc-sm"

Q: What should I do if the pod tmc-agent-installer-* in error state while registering the supervisor cluster to TMC?

A: The pod ***tmc-agent-installer-**** failed to start usually because it failed to download the manifests from TMC.
You can check the logs of pod ***tmc-agent-installer-**** in the namespace ***svc-tmc-c**** on the supervisor cluster if it contains the error message like "**download and apply registration link, attempt: 7: download manifest from link: get registration link: Get \"https://tmc.example.com/...\"**" which is usually caused by two kinds of issue:

- If you can also see the '**"msg": "using proxy"**' in the log message, it indicates the supervisor cluster is configured proxy, however it couldn't apply the no_proxy settings to the pod tmc-agent-installer, so the manifest failed to be downloaded through proxy server. Thus, you have to download and apply the manifest manually with below steps:
  1. Get the token and the registration link from the contents of AgentInstall CR on the TMC SM UI

  2. Download and generate the manifest. You should replace the correct values, **$bootstrap_token** is required be decoded with base64, **$svc_tmc** is the ***svc-tmc-c**** on the supervisor cluster.

curl -H "Authorization: Bearer **$bootstrap_token**" **$registration_link** | yq -p json '.manifest'| sed 's/{{.Namespace}}/**$svc_tmc**/' > agent-install.yaml

3. Install agents with manifest on the supervisor cluster. **Note that it requires the admin kubeconfig permission of the supervisor cluster.**

```
kubectl -n $svc_tmc apply -f agent-install.yaml
```

- If it's not using proxy, and you see the error message "**x509: certificate signed by unknown authority**" in the logs, you can validate the certificates in **spec.caCerts** in the AgentConfig CR, if it matches with the TMC SM configuration. If the certificate is invalid, please update the agentconfig with the correct certificate and try again.

```
kubectl -n $svc_tmc  get agentconfig tmc-agent-config -ojsonpath='{.spec.caCerts}' > ca.crt
openssl s_client -connect $TMCSM_FQDN:443 -CAfile=ca.crt
```

Q: What should I do if the pod tmc-bootstrapper in ImagePullBackOff state on the supervisor cluster?

A: You can describe the pod of tmc-bootstrapper in ImagePullBackOff state with command like ***"kubectl -n svc-tmc-c* describe pod tmc-bootstrapper-*"*** to show more details:

- If it shows the message like "https://harbor.example.com/tmc/tap-tmc-docker-virtual.usw1.packages.broadcom.com/extensions/tmc-bootstrapper/manager/manifests/sha256:…: **tls: failed to verify certificate: x509: certificate signed by unknown authority**", this issue is caused when the certificate of Harbor registry is not trusted on the supervisor cluster nodes, you should deregister the supervisor cluster from TMC and reregister it again to make the daemonset domain-local-ds to refresh the certificates.
- If it shows the message like "harbor.example.com/tmc//tap-tmc-docker-virtual.usw1.packages.broadcom.com/extenstions/tmc-bootstrapper/manager@sha…", the issue is caused by a misconfigured **harborProject** with trailing slash in the TMC SM configuration,e.g "harborProject: harbor.example.com/tmc/". You should update the configuration for the TMC SM by removing the extra trailing slash after the project name, e.g "harborProject: harbor.example.com/tmc".

Q: What should I do if the pod tmc-bootstrapper in ImagePoolBackOff state on the workload cluster?

A: You can describe the pod of tmc-bootstrapper in ImagePullBackOff state with command "kubectl -n vmware-system-tmc describe pod tmc-bootstrapper-*" to show more details.
If it shows the message like "https://harbor.example.com/tmc/tap-tmc-docker-virtual.usw1.packages.broadcom.com/extensions/tmc-bootstrapper/manager/manifests/sha256:…: **tls: failed to verify certificate: x509: certificate signed by unknown authority**", this issue likely occurs when the certificate of Harbor registry is not trusted by the cluster nodes.

- For an attaching cluster, you should follow the VKS doc to update the cluster to trust the certificate of Harbor registry.
- For a managed cluster, it might be a temporary failure during the cluster rolling update, the node might not be replaced and certificates have not been updated yet, the issue could be resolved after the cluster becomes ready.

Q: What should I do if the pod tmc-bootstrapper in CrashLoopBackOff state on the cluster?

A: You can check the logs of the pod tmc-bootstrapper if it failed due to connection error through proxy, it's likely because the no_proxy list does not include the TMC domain, you can create new proxy with TMC domain in the no_proxy list and then manage/attach the cluster again.

```
{"level":"info","msg":"using explicit proxy for TMC communication","sub-component":"tmc-context","time":"2024-06-26T09:44:35Z"}
failed to build bootstrapper context failed to initialize gRPC connection: could not connect to TMC endpoint: context deadline exceeded
```

Q: What should I do if the supervisor cluster or workload cluster is in disconnected status and the logs of TMC agent pod extension-updater show "code = Unauthenticated desc = No valid authentication credentials"?

A: The error message "*code = Unauthenticated desc = No valid authentication credentials*" in extension-updater usually indicates the token for the agent extensions has expired.

- For supervisor cluster, you can reregister it with the tanzu CLI

`tanzu tmc mc reregister {{MANAGMENT_CLUSTER_NAME}} -k {{SUPERVISOR_CLUSTER_KUBECONFIG}}`

- For workload cluster, you can reattach it with tanzu CLI

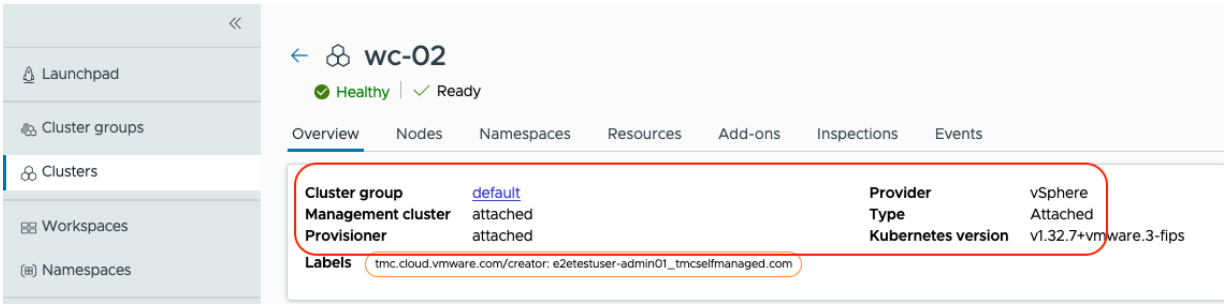
`tanzu tmc cluster reattach {{CLUSTER_NAME}} -m {{MANAGMENT_CLUSTER_NAME}} -p {{PROVISIONER_NAME}} -k {{CLUSTER_KUBECONFIG}}`

Q: How can I know a cluster is in managed or attached status?
A: You should exactly know how your cluster is onboarded to TMC before you are going to delete a cluster from TMC, because the behavior and effects of the 'Delete' operation are very different depending on whether the cluster is 'managed' or 'attached'.

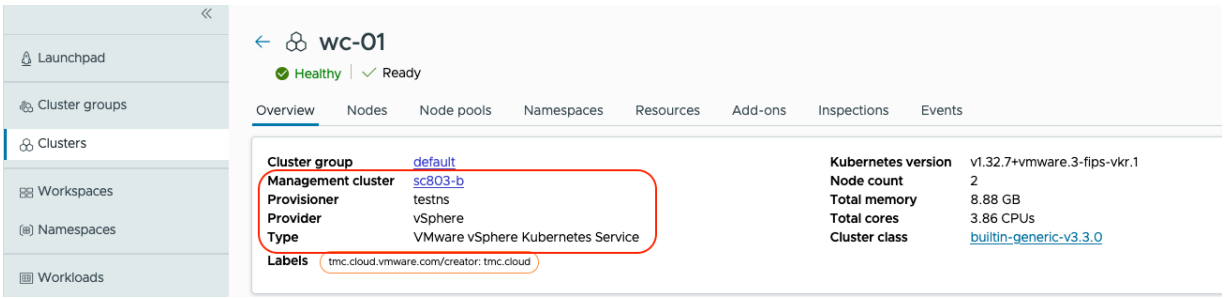
- Managed cluster: The cluster can be **deleted** from TMC as well as from **Supervisor cluster.** You can **unmanage** it if you only want to remove it from TMC.
- Attached cluster: The cluster is only **removed** from TMC and the TMC agents extensions are removed from VKS cluster.

You can identify the cluster status by two approaches:

- From the URL of cluster information: The format of URL is **https://{{TMC_FQDN}}/clusters/{{CLUSTE_NAME}}/{{MANAGEMENT_CLUSTER_NAME}}/{{PROVISIONER_NAME}}/overview**, e.g https://tmc.example.com/clusters/wc-01/sc803/testns/overvie
  - **{{MANAGEMENT_CLUSTER_NAME}}** and **{{PROVISIONER_NAME}}** are "**attached**" for the attached cluster.
  - **{{MANAGEMENT_CLUSTER_NAME}}** is the supervisor name and **{{PROVISIONER_NAME}}** is the cluster namespace for the managed cluster.
- From the cluster detailed information on the cluster page:
  - The fields "**Management cluster**", "**Provisioner**", "**Type**" all are "**attached**" for the attached cluster.



- The field "**Management cluster**" is the supervisor name, "**Provisioner**" is the cluster namespace, "**Type**" is "VMware vSphere Kubernetes Service" for managed cluster.

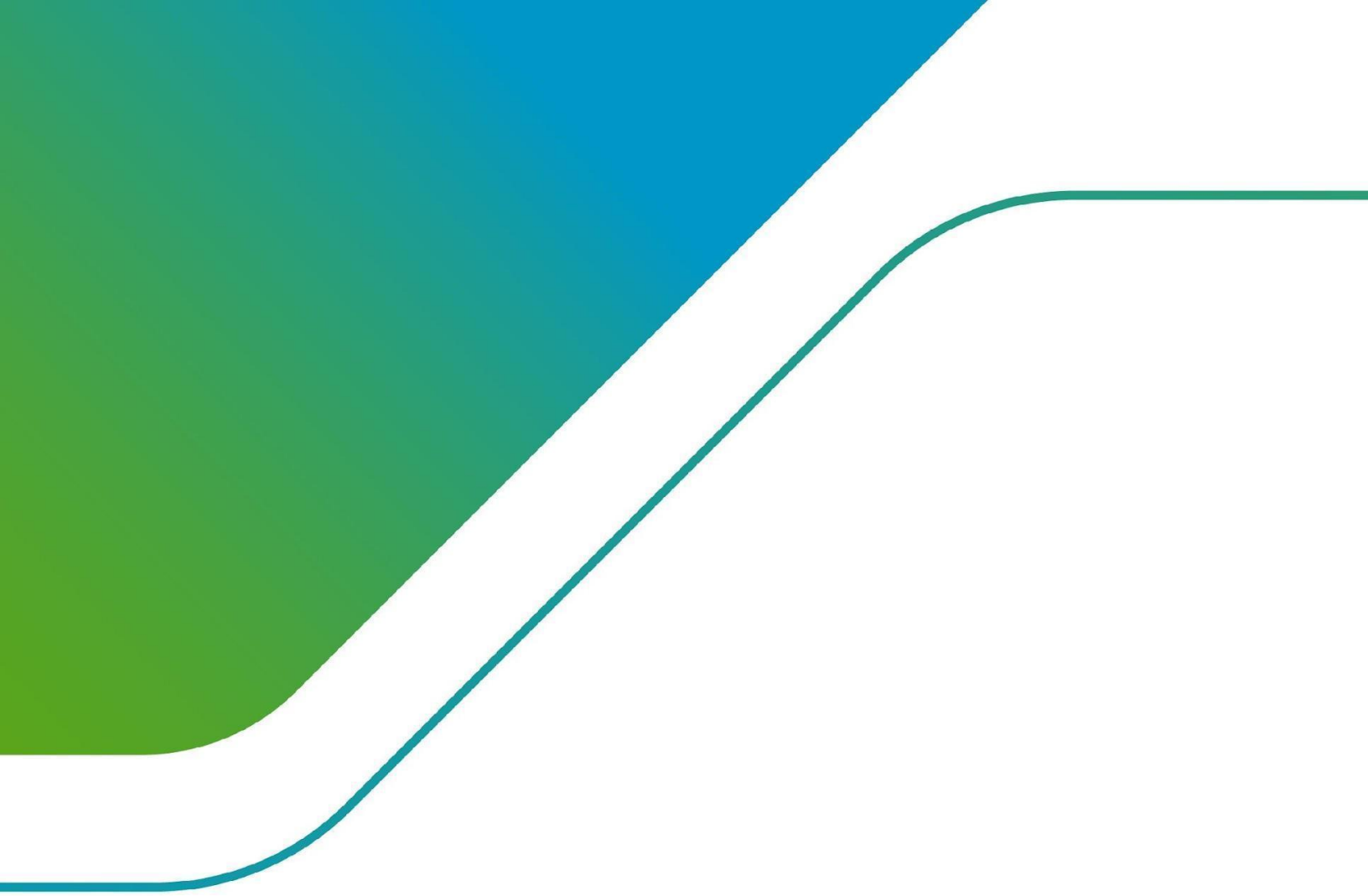


Q: How can I use a Tanzu repository for a certain cluster other than the default standard repository on the Harbor registry for TMC SM packages?

A: You can disable the currently default standard repository on the Harbor registry for TMC SM packages, then add a new standard repository from different registry, e.g projects.packages.broadcom.com/vsphere/tmc-sm/standard/repo/v2025.6.18:v2025.6.18, ensure the status get green, then use kubectl to restart the kapp-controller on the cluster to flush the packages cache, the packageinstalls will be updated to the new repository.

Resources

- README.md of migration scripts
- Various ways customers can install TMC SM
- The configuration keys for provisioning TMC SM
- The user guide of Tanzu TMC CLI Plugin-in



vmware by Broadcom

Item No: vmw-bc-pguide-temp-uslet-word-2025 Jul-25