Custom CAs enhancement for TKGs WC

Author(s): [Xianping Zhang](mailto:zxianping@vmware.com)  
Reviewer(s): [Willis Ren](mailto:renw@vmware.com)[Lu Wang](mailto:luwang@vmware.com)[Weiwei He](mailto:hweiwei@vmware.com)

# Introduction

This doc is initiated to improve custom CAs UX in TMC Self-Managed for TKGs workload cluster.

The initial feature request [ticket](https://jira.eng.vmware.com/browse/OLYMP-37521) for TKGs.

There are 2 topics here:

1. TMC image registry topologies supported for TKGs.
2. Detailed design of CA certificates auto-injection to TKGs workload cluster.

# Goal

* TMC image registry topologies supported for TKGs.
* Custom CAs support when provisioning TKGs workload clusters.
* Custom CAs support when managing/upgrading TKGs workload clusters.

# Non-Goal

* CA certs rotation.

# Overview

Custom CAs support matrix on TKGs

| Workload cluster | v1.6 | v2.0 | v2.1+ |
| --- | --- | --- | --- |
| Legacy | Yes | NA | NA |
| CC | NA | Yes | NA |

# Details

## TKGs Image registry [topology](https://www.google.com.hk/url?q=https://miro.com/app/board/uXjVMTlCg1E%3D/&sa=D&source=docs&ust=1684146049897587&usg=AOvVaw2qGyAuc6iV0pkAYilVvYAF)

TKGs support 3 different types of image registry:

1. T1: External container registry.
2. T2: Embedded harbor registry.
3. T3: External container registry with embedded harbor registry enabled on SV.

### T1: External container registry

Use an [external container registry](https://docs.vmware.com/en/VMware-vSphere/7.0/vmware-vsphere-with-tanzu/GUID-376FCCD1-7743-4202-ACCA-56F214B6892F.html) with Tanzu Kubernetes cluster pods. This is an alternative to using the embedded Harbor Registry.

### T2: Embedded harbor registry

Embedded harbor registry, [enabled the Harbor Registry on Supervisor](https://docs.vmware.com/en/VMware-vSphere/7.0/vmware-vsphere-with-tanzu/GUID-AE24CF79-3C74-4CCD-B7C7-757AD082D86A.html) by user, used as TMC image registry.

### T3: External container registry with embedded harbor registry enabled on SV

Both the registries, external private registry and the embedded harbor registry on SV, are deployed.

1. There is an external independent private registry deployed, it also could be an embedded harbor registry which is shared by other supervisor clusters.
2. The supervisor clusters enabled the embedded harbor registry.

Normally, this is not recommended, because the embedded harbor registry may impact the private registry CA certificates injection, when provisioning workload clusters. While there do have the scenario as below:

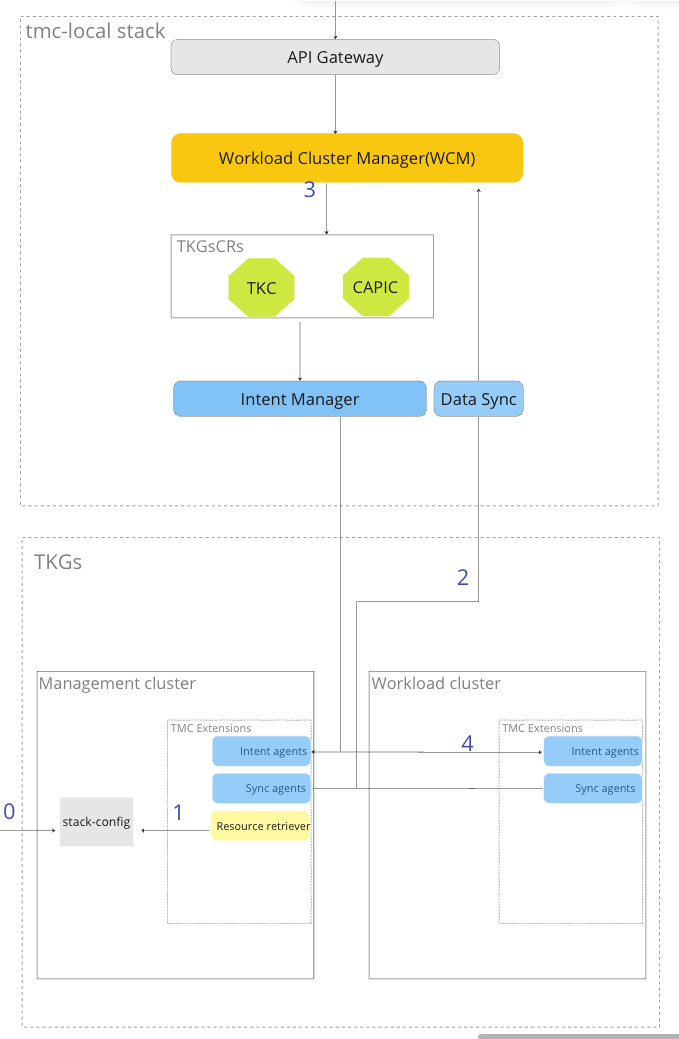
1. Firstly, customer deployed a independent private registry for TMC-local
2. Then, customers enabled the embedded registry for some reasons.

We need to support this, and test on this scenario.

## Custom CAs design

### Possible solutions

**Solution1**:

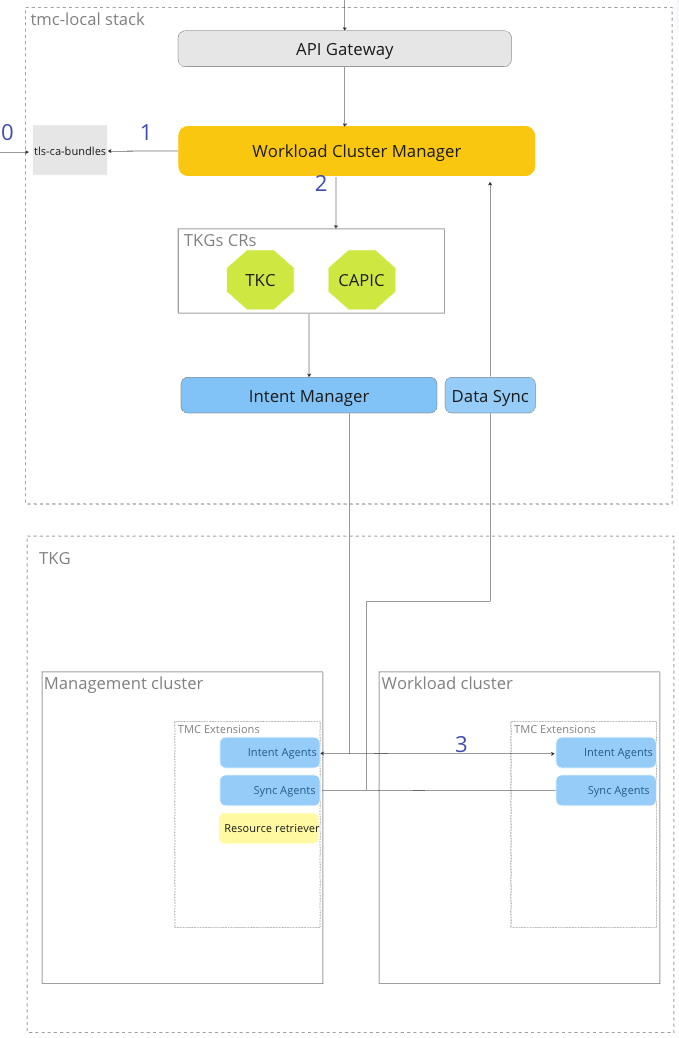


Main workflows:

1. Prerequisite, the configmap stack-config, with required TLS CA bundles, will be created when mgmt cluster registers.
2. The resource-retriever collects the CA bundles.
3. Sync agents sync the CA bundles back into WCM.
4. Required caCerts will be injected into cluster spec when provisioning workload clusters.
5. The caCerts will be injected into workload cluster nodes.

Drawbacks: The TLS CA bundles come from tls-ca-bundles, it seems redundant to retrieve and sync back to WCM.

**Solution2**:



Keep it simple.

1. Prerequisite, the configmap should be created already, with required TLS CA bundles.
2. WCM reads CA bundles from configmap.
3. Required caCerts will be injected into cluster spec when provisioning workload clusters.
4. The caCerts will be injected into workload cluster nodes.

Limitation: Currently not found.

**Solution3**: leverage lcm-tkg-extension to support registry/repository and certs.

Limitation: lcm-tkg-extension is not supported in TKGs.

### Final Design

Considering the limitations of the above 3 solutions, we take solution2 as the final solution.

**Config** for TMC local, the type of OLYMPUS\_CELL\_TYPE: local, is the tag for tmc local.

**WCM** will read the caCerts from tls-ca-bundles configmap, and inject into the cluster spec as *trust*. The sample trust content for TKGs VC70/VC80

|  | Support API | Variables | comments |
| --- | --- | --- | --- |
| VC702 | Not support | Not support |  |
| VC703 | Create  Patch(Manage)  Update(Upgrade) | Add below settings when creating WC  trust:  additionalTrustedCAs:  - name: interophub2.eastus.cloudapp.azure.com.ca.crt  data: LS0LQo= | When managing/upgrading a cluster with self-signed CA for TMC-local, the cluster node will be rolling-updated, there may be risk on customer applications. (Only when certs add/remove/change) |
| VC80 | Create  Patch(Manage)  Update(Upgrade) | 1.Create a secret named %s-user-trusted-ca-secret, where %s is the name of the user's Cluster. This secret has to be under the same namespace as the Cluster  apiVersion: v1  data:  zx.io-ca: LSxxQo=  kind: Secret  metadata:  name: cluster-user-trusted-ca-secret  namespace: testns  type: Opaque    2.In your Cluster, put the trust variable as shown below  - name: trust  value:  additionalTrustedCAs:  - name: zx.io-ca | The cluster creation will be in pending status if the secret has not been created.  The cert of the Secret, should be base64 encoded twice.  Cluster nodes will be rolling-updated, only when certs add/remove/change. |

**UI** enhancement to support Manage/Upgrade workload with self-signed CA certs for TMC-local.

| Cluster operations | UI enhancement | comments |
| --- | --- | --- |
| Create | No UI change |  |
| Manage |  | The cluster will be rolling updated to support self-signed CA certs for TMC local  (Only when certs add/remove/change)  Add readonly label to workload clusters *"tmc.cloud.vmware.com/self-managed": "true"*  Add warning message: *"*The cluster may result in rolling update when it's going to be managed*"* |
| Upgrade |  | The cluster will be rolling updated to support self-signed CA certs for TMC local (Only when certs add/remove/change)  Warning message is Not needed for upgrade, since upgrade will cause cluster rolling update already. |

## Testing

### TKGs VC702 Not support

### TKGs VC703

|  | context | comments |
| --- | --- | --- |
| Create | Create cluster vc70-w1  Validata:   1. vc70-w1 create success 2. Check spec of settings/trust cert name/data is expected.   "trust": {  "additionalTrustedCAs": [  {  "name": "extensions.aws-usw2.tmc-dev.cloud.vmware.com",  "data": "xxxx="  }  ]  }   1. Ssh to cluster nodes, double confirm the /etc/ssl/certs/tkg-extensions.aws-usw2.tmc-dev.cloud.vmware.com-ca.pem is expected. |  |
| Update | 1. Normal upgrade(non-self-signed CA support), no special change 2. Self-signed CA support upgrade:   Create cluster without tmc-local caCerts  Upgrade, and tmc-local caCerts will be auto-injected. | Cluster upgrade – rolling-update |
| Patch | 1. Normal patch(non-self-signed CA support), no special change 2. Self-signed CA support upgrade:   Manual create a cluster with cli, without tmc-local caCerts.  Managed by TMC, and tmc-local caCerts will be auto-injected. | Cluster managing – rolling-update |
| Others | No change, and will be covered by the automation pipeline in case of regression. |  |

### TKGs VC80

|  | context | comments |
| --- | --- | --- |
| Create | Create cluster vc70-w1  Validata:   1. vc70-w1 create success 2. Check spec of settings/trust cert name/data is expected.   "trust": {  "additionalTrustedCAs": [  {  "name": "extensions.aws-usw2.tmc-dev.cloud.vmware.com",  "data": "xxxx="  }  ]  }   1. Ssh to cluster nodes, double confirm the /etc/ssl/certs/tkg-extensions.aws-usw2.tmc-dev.cloud.vmware.com-ca.pem is correct. |  |
| Update | Create a cluster without tmc-local caCerts.  Upgrade, and tmc-local caCerts will be auto-injected. | Cluster upgrade – rolling-update |
| Patch | Manual create a cluster with cli, without tmc-local caCerts.  Managed by TMC, and tmc-local caCerts will be auto-injected. | Cluster managing – rolling-update |

# Appendix

## Reference

Related [tickets](https://jira.eng.vmware.com/browse/OLYMP-37521) supported.

The [miroboard](https://miro.com/app/board/uXjVMTlCg1E=/) of the TKGs registry topology.

[TKG Documentation for VC8](https://confluence.eng.vmware.com/display/CNA/TKG+2.0+Documentation+for+8.0)0.

Detailed content of *tls-ca-bundle*

apiVersion: v1

data:

letsencrypt.pem: |-

-----BEGIN CERTIFICATE-----

BASE64CACerts==

-----END CERTIFICATE-----

kind: ConfigMap

metadata:

annotations:

meta.helm.sh/release-name: tls-ca-bundles

meta.helm.sh/release-namespace: tmc-local

creationTimestamp: "2023-04-10T09:43:49Z"

labels:

app: raw

app.kubernetes.io/managed-by: Helm

chart: raw-0.2.5

heritage: Helm

release: tls-ca-bundles

name: tls-ca-bundles

namespace: tmc-local

resourceVersion: "5162"

uid: e3aef7e8-f472-4216-8e1d-96ab48beef4c

Detailed content of *stack-config*

apiVersion: v1

data:

cluster\_rid: rid:mc:bc27608b-4809-4cac-9e04-778803963da2:tkgs-vc703

management\_cluster\_name: tkgs-vc703

org\_id: bc27608b-4809-4cac-9e04-778803963da2

resource\_uid: mc:01GYEH4HJND90XSNCP1RNHZXJ2

stack\_type: tmc

tls.crt: |

-----BEGIN CERTIFICATE-----

xxx

-----END CERTIFICATE-----

tmc\_host: zxp.tmc-dev.cloud.vmware.com

tmc\_url: https://zxp.tmc-dev.cloud.vmware.com

kind: ConfigMap

metadata:

annotations:

kubectl.kubernetes.io/last-applied-configuration: |

{"apiVersion":"v1"...}

labels:

tmc.cloud.vmware.com/managed: "true"

name: stack-config

namespace: svc-tmc-c8

selfLink: /api/v1/namespaces/svc-tmc-c8/configmaps/stack-config

Configmap *global* is stack level, the main content as below.

apiVersion: v1

data:

OLYMPUS\_ACCOUNT\_MANAGER\_ENCRYPTION\_TYPE: LOCALAES

OLYMPUS\_AUTHENTICATOR\_DOMAIN: auth.tmc.tanzu.io

OLYMPUS\_BI\_DATA\_COLLECT\_ENABLED: "false"

OLYMPUS\_CELL\_ENVIRONMENT: ""

OLYMPUS\_CELL\_NAME: local

OLYMPUS\_CELL\_REGION: aws-usw2

OLYMPUS\_CELL\_TYPE: local

OLYMPUS\_CELL\_ZONE: tmc.tanzu.io

OLYMPUS\_COLLECTOR\_HOST\_PORT: wavefront-proxy.wavefront-tanzu:14267

OLYMPUS\_COOKIE\_DOMAIN: tmc.tanzu.io

OLYMPUS\_CSP\_AUTHORIZE\_URL: CSP\_REFERENCE\_NOT\_APPLICABLE\_FOR\_LOCAL

OLYMPUS\_CSP\_BASE\_URL: CSP\_REFERENCE\_NOT\_APPLICABLE\_FOR\_LOCAL

OLYMPUS\_CSP\_CLIENT\_ID\_LANDING: CSP\_REFERENCE\_NOT\_APPLICABLE\_FOR\_LOCAL

OLYMPUS\_CSP\_CLIENT\_ID\_LANDING\_LOGIN: 2a1f3fa2-fc1b-4a18-863d-2a75db3e1744

OLYMPUS\_CSP\_DISCOVERY\_URL: CSP\_REFERENCE\_NOT\_APPLICABLE\_FOR\_LOCAL

OLYMPUS\_CSP\_IDP\_ENABLED: "false"

OLYMPUS\_CSP\_ISSUER\_URL: CSP\_REFERENCE\_NOT\_APPLICABLE\_FOR\_LOCAL

OLYMPUS\_CSP\_MESSENGER\_ORG: CSP\_REFERENCE\_NOT\_APPLICABLE\_FOR\_LOCAL

OLYMPUS\_CSP\_MESSENGER\_ORG\_ROLE: CSP\_REFERENCE\_NOT\_APPLICABLE\_FOR\_LOCAL

OLYMPUS\_CSP\_PUBLISHER\_ID: CSP\_REFERENCE\_NOT\_APPLICABLE\_FOR\_LOCAL

OLYMPUS\_CSP\_REDIRECT\_URL: ""

OLYMPUS\_CSP\_SERVICE\_ID: CSP\_REFERENCE\_NOT\_APPLICABLE\_FOR\_LOCAL

OLYMPUS\_CSP\_SERVICE\_ORG\_ID: CSP\_REFERENCE\_NOT\_APPLICABLE\_FOR\_LOCAL

OLYMPUS\_CUSTOMER\_DOMAIN\_ROOT: tmc.tanzu.io

OLYMPUS\_DEFAULT\_IDP: vcd

OLYMPUS\_DEV\_DOMAIN\_ROOT: tmc.tanzu.io

OLYMPUS\_DEV\_ENV\_ENABLED: "false"

OLYMPUS\_ENVIRONMENT\_SUFFIX: ""

OLYMPUS\_EU\_LOGICAL\_STACK\_ID\_FOR\_ONBOARDING: a426f7e1-59e6-4724-9322-a994e59a581b

OLYMPUS\_EXTENSIONS\_ENDPOINT\_FQDN: harbor.tanzu.io:8443/tmc/498533941640.dkr.ecr.us-west-2.amazonaws.com

OLYMPUS\_GAZ\_ISSUER: CSP\_REFERENCE\_NOT\_APPLICABLE\_FOR\_LOCAL

OLYMPUS\_GLOBAL\_AUTH\_ENDPOINT: https://auth.tmc.tanzu.io/api/v1/login

OLYMPUS\_GLOBAL\_AUTHENTICATOR\_ENDPOINT: https://auth.tmc.tanzu.io

OLYMPUS\_GLOBAL\_CERTIFICATE\_ISSUER\_KIND: ClusterIssuer

OLYMPUS\_GLOBAL\_CERTIFICATE\_ISSUER\_NAME: local-issuer

OLYMPUS\_GLOBAL\_LANDING\_ENDPOINT: https://landing.tmc.tanzu.io

OLYMPUS\_GLOBAL\_ROUTER\_DOMAIN: gr.tmc.tanzu.io

OLYMPUS\_GLOBAL\_TENANCY\_ENDPOINT: gts.tmc.tanzu.io

OLYMPUS\_GLOBAL\_TRUSTED\_CLIENT\_DOMAIN\_SUFFIXES: .tanzu.io

OLYMPUS\_GLOBAL\_UI\_ENDPOINT: https://onboarding.tmc.tanzu.io/callback

OLYMPUS\_LANDING\_SERVICE\_DOMAIN: landing.tmc.tanzu.io

OLYMPUS\_LANDING\_SERVICE\_GLOBAL\_DOMAIN: landing.tmc.tanzu.io

OLYMPUS\_OAUTH\_CLIENT\_ID: tmc-web-oauth-client

OLYMPUS\_OIDC\_CLIENT\_ID: 2a1f3fa2-fc1b-4a18-863d-2a75db3e1744

OLYMPUS\_OIDC\_ISSUER: https://atl1-vcd-static-100-186.eng.vmware.com/oidc

OLYMPUS\_OIDC\_SERVICE\_ID: 0f20a3f6-ab70-482e-bb3b-f94a064f3b9b

OLYMPUS\_OPERATOR\_ORG\_ID: ""

OLYMPUS\_PINNIPED\_IDP\_ENABLED: "false"

OLYMPUS\_PINNIPED\_ISSUER\_URL: ""

OLYMPUS\_PINNIPED\_OAUTH\_CLIENT\_ID: ""

OLYMPUS\_PRIMARY\_STACK\_DOMAIN: stacks.tmc.tanzu.io

OLYMPUS\_REGION: us-west-2

OLYMPUS\_RESOURCE\_MANAGER\_NUM\_WORKERS: "1"

OLYMPUS\_ROUTING\_ALIAS: main-tenancy.tmc.tanzu.io

OLYMPUS\_ROUTING\_HEALTH\_NODEPORT: "32430"

OLYMPUS\_ROUTING\_HOSTED\_ZONE\_ID: ZJ2FJT6C19GWT

OLYMPUS\_ROUTING\_HTTP\_NODEPORT: "30420"

OLYMPUS\_ROUTING\_HTTPS\_NODEPORT: "32420"

OLYMPUS\_S3\_BUCKET\_LOGS: tmc-local

OLYMPUS\_SERVICE\_ADMIN\_ROLE: service:admin

OLYMPUS\_SERVICE\_MEMBER\_ROLE: service:member

OLYMPUS\_SERVICE\_OAUTH\_CLIENT\_ID: YbECMBPJOONYxs8fCh4TaH7rX3CVcsBb5uK

OLYMPUS\_STACK\_CHANNEL: ""

OLYMPUS\_STACK\_ID: 128b7853-589d-4061-a445-44bdd7750f70

OLYMPUS\_STACK\_NAME: local

OLYMPUS\_STACK\_PUBLISHER\_DOMAIN: sp.tmc.tanzu.io

OLYMPUS\_SUPER\_COLLIDER\_ID\_PREFIX: ""

OLYMPUS\_TENANCY\_API\_DOMAIN: gts.tmc.tanzu.io

OLYMPUS\_TENANCY\_REST\_API\_DOMAIN: gts-rest.tmc.tanzu.io

OLYMPUS\_TRACING\_DISABLE: "true"

OLYMPUS\_TRACING\_HOST: nonexistant

OLYMPUS\_TRACING\_INSECURE: "true"

OLYMPUS\_TRACING\_PORT: "12345"

OLYMPUS\_TRACING\_RATE: "10"

OLYMPUS\_TRACING\_SAMPLING\_DISABLE: ""

OLYMPUS\_UI\_ONBOARDING\_GLOBAL\_DOMAIN: onboarding.tmc.tanzu.io

OLYMPUS\_UI\_ONBOARDING\_SERVICE\_DOMAIN: onboarding.tmc.tanzu.io

OLYMPUS\_UI\_PRODUCT: vcd

OLYMPUS\_UI\_REDIRECT\_URL: https://landing.tmc.tanzu.io/callback

OLYMPUS\_UNSAFE\_TRACING\_INSECURE: "true"

OLYMPUS\_US\_LOGICAL\_STACK\_ID\_FOR\_ONBOARDING: bdda398e-7292-481d-81dd-0db4bc747342

OLYMPUS\_VCD\_IDP\_ENABLED: "true"

OLYMPUS\_VCD\_ISSUER\_URL: https://atl1-vcd-static-100-186.eng.vmware.com/oidc

OLYMPUS\_VCD\_OAUTH\_CLIENT\_ID: 2a1f3fa2-fc1b-4a18-863d-2a75db3e1744

kind: ConfigMap

metadata:

...

name: global

namespace: tmc-local