TKG Automation with Cloud Assembly

Assaf Sauer, Tim Gretler

Vmware Inc.

May 2020

Contents

1 Introduction

This is a Guide on how to automate Tanzu Kubernetes Grid with Cloud Assembly and integrate it with Tanzu Mission Control, which is part of the VMware modern application portfolio [4]. It is thought to be a proof of concept to show how you can automate the control plane creation of Tanzu Kubernetes Grid. This will give you the ability to deploy a TKG management plane with a few clicks on AWS and vSphere.

1.1 Tanzu Kubernetes Grid

Tanzu Kuberenetes Grid is a certified Kubernetes distribution developed by VMware [5]. It strives to build a simplified infrastructure across a multi-cloud environment. This will improve ease of development and also the speed of development, while still providing production-grade stability. New automation capabilities provide a much faster and easier setup and therefore higher flexibility in a diverse working environment.

1.2 Cloud Assembly

Cloud Assembly is a VMware automation product [1]. It gives you a simple interface to your multi-cloud environment. With Cloud Assembly you have Infrastructure-as-Code and you can deploy templates to multiple cloud from a single interface. This provides the benefit of single source of truth for your deployments independent of the underlying cloud.

1.3 Tanzu Mission Control

Tanzu Mission Control is the central control hub for your Kubernetes clusters [6]. Tanzu Mission Control gives you insights into your fleet of clusters. One can also apply policies across all clusters from Tanzu Mission Control. You can attach any CNCF certified cluster and manage its lifecycle.

2 Manual

2.1 Prerequisites

- Access to Cloud Assembly

- vSphere Environment
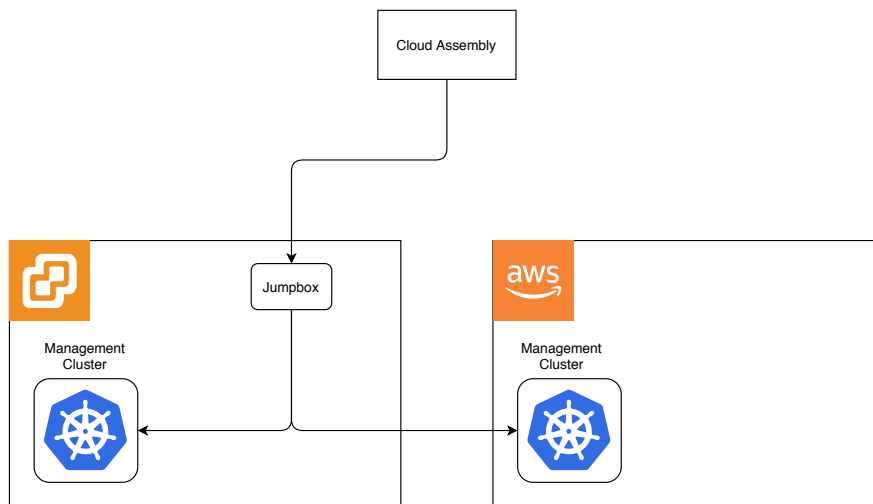
- AWS Account

2.2 General Overview

Figure 1: Architecture created by automation

Figure 1 provides an illustration of the created architecture by the automation. Cloud Assembly will deploy a Virtual Machine inside your vCenter and install all the software needed for TKG. After the software installation, it automatically deploys a management cluster to vSphere and AWS. When the process is complete you will have access to both management clusters and you are able to create clusters as you wish

2.3 Setup Cloud Assembly

Instructions to setup Cloud Assembly with vCenter. Follow all steps. This guide assumes that you have no configurations in cloud assembly. You can skip certain steps if you already have a working setup.

2.3.1 Connect Cloud Assembly and vCenter

1. Log into Cloud Assembly

2. Switch to infrastructure tab of Cloud Assembly

3. In the menu on the left choose Cloud Accounts

4. Click "Add Account" and choose vCenter

5. Click "New Cloud Proxy" and deploy a proxy by following the given instruction if you haven't already

6. Name your Cloud Account and provide login credentials of your vCenter

7. Click add at the bottom

2.3.2   Add Cloud Zone

1. Log into Cloud Assembly

2. Switch to infrastructure tab of Cloud Assembly

3. In the menu on the left choose Cloud Zones

4. If you have correctly setup the vCenter connection you should see your vCenter under Account/Region. Choose your vCenter

5. Name your Cloud Zone and add it

2.3.3   Create Ubuntu 18.04 Template in vCenter

1. Log into your vCenter

2. Right Click on your Datacenter and select "Deploy OVF Template"

3. As source you can add following url `https://cloud-images.ubuntu.com/bionic/current/bionic-server-cloudimg-amd64.ova` . This will get you the ubuntu cloud image

4. You can just click through the remaining options

5. Click "Finish"

6. Wait until you see Ubuntu ova deployed to your vCenter

7. Right Click on the deployed VM -¿ choose "Template" -¿ click convert to Template

8. Now you should see a Ubuntu template under your templates

2.3.4   Add Image mapping

1. Log into Cloud Assembly

2. Switch to infrastructure tab of Cloud Assembly

3. In the menu on the left choose Image Mapping

4. Click "New Image Mapping"

5. Name it "tkg-template"

6. Select your Account/Region

7. If you correctly created the Ubuntu template you should now see it under Image. Select it!

8. Click create

2.3.5   Add Flavor mapping

1. Log into Cloud Assembly

2. Switch to infrastructure tab of Cloud Assembly

3. In the menu on the left choose Flavor Mapping

4. Click "New Flavor Mapping" if a small Mapping doesn't already exist. If it already exists make sure you have a mapping for your account!

5. Name it "small"

6. Select your Account/Region

7. Configure your small Flavor by choosing an appropriate number of CPU and Memory

8. Click create

2.3.6   Create Project

1. Log into Cloud Assembly

2. Switch to infrastructure tab of Cloud Assembly

3. In the menu on the left choose Projects

4. Click "New Project"

5. Name the project

6. Switch tab to "user" and add yourself

7. Switch tab to "provisioning" and add your cloud zone

8. Click create

2.4 Create Tanzu Mission Control API Keys

1. Log into VMware Cloud Services

2. Click on your name on the top right and choose "My Account"

3. Switch to tab API Tokens

4. Click "Generate Token"

5. Name the token and choose "All Roles"

6. Generate Token and save it!

2.5 Deploy Tanzu Kubernetes Grid

Now we have everything setup to deploy TKG. As an attachment to this PDF you will find the Cloud Assembly template, which deploys TKG. You will also get AWS credentials to download the TKG binaries. You should keep them secret and don't publish them anywhere!

2.5.1 Import Template to Cloud Assembly

1. Log into Cloud Assembly

2. Switch to Design tab of Cloud Assembly

3. Click "Upload"

4. Name your design, select your project and upload the attached TKG deployment yaml.

5. Click "Upload"

2.5.2 Deploy TKG

1. Log into Cloud Assembly

2. Switch to Design tab of Cloud Assembly

3. Go to your design by selecting it

4. Click "Deploy" on the bottom left

5. Name your Deployment and choose current draft

6. Fill out prompted values

   - `env:` Your target cloud environment
   - `AWS_KEY_ID_S3:` ID of provided credentials to download binaries
   - `AWS_SECRET_KEY_S3:` Secret Key of provided credentials to download binaries

- `AWS_KEY_ID_COMPUTE`: Key ID of your AWS account on where to deploy TKG. Need to have administrator permissions
- `AWS_SECRET_KEY_COMPUTE`: Secret Key of your AWS account on where to deploy TKG
- `AWS_COMPUTE_REGION`: Region where to deploy
- `TMC_API_TOKEN`: Tanzu Mission Control Token created in 2.4
- `tkg-cli`: Name of tkg-cli file in s3. Can be left default
- `haproxy-ova`: Name of haproxy-ova file in s3. Can be left default
- `tkg-ova`: Name of tkg-ova file in s3. Can be left default
- `datastore`: Datastore to use
- `vcenterIP`: IP of your vCenter
- `vcenterUser`: User of your vCenter
- `vcenterPassword`: Password of your vCenter
- `resourcePool`: Resourcepool to deploy to
- `sshKey`: Public Key to connect to TKG nodes

Note: If you don't want to deploy to AWS/vSphere you can fill in an invalid value. This leads to not deploying this provider. The same goes for TMC.

Now you can see your deployment under the Cloud Assembly tab Deployments. You will see the IP of your newly created jumpbox. You can now connect to the jumpbox, but the TKG installation wil take aprox. 45-60min.

3 Tanzu Kubernets Grid Basics

Interaction with TKG happens through the command line. This sections provides a brief overview. TKG uses the cluster API [3] to create new clusters therefore you need a seperate management cluster for vCenter and AWS. In general you always have to set a context, which is either the vCenter or the AWS management cluster.

3.1 General Help

The command line interface is easy to use if you are familiar with the linux command line. If you don't know what to type you can always call for help like this:

```
$ tkg −h
```

 or

```
$ tkg <command> −h
```

3.2   Some TKG CLI commands

This command sets the management cluster. You can switch between management clusters as you like.

```
$ tkg set management−cluster <cluster_name>
```

This command lets you create a new cluster.

```
$ tkg create cluster <new_cluster_name>
```

4   Troubleshooting

4.1   Debugging Jumpbox

Access cloud-init logs:

```
$ cat /var/log/cloud−init−output.log
```

4.2   Cloud Assembly Deployment failed

To debug cloud assembly following steps are helpful.

1. Log into Cloud Assembly

2. Go to Cloud Assembly tab Infrastructure

3. In the panel on the left hand side choose "Requests" under "Activity"

4. Now you can find your deployment that failed

5. You now see the decision steps of Cloud Assembly including where it failed

5   Appendix

5.1   Cloud-init

Cloud init is a tool for customising cloud instances [2]. It automatically applies user data to each cloud instance. It supports all major linux distributions and all major public and private cloud providers. In this automation we heavily used cloud init to configure the jumpbox, which is created by Cloud Assembly. Cloud Assembly provides you the possibility to use cloud init, that allowed us to run the entire setup of the management clusters in cloud init.

References

[1] Cloud assembly. `https://cloud.vmware.com/cloud-assembly`.

[2] Cloud init. `https://cloud-init.io/`.

[3] Cluster api. `https://github.com/kubernetes-sigs/cluster-api`.

[4] Tanzu. `https://tanzu.vmware.com/tanzu`.

[5] Tanzu kubernetes grid. `https://tanzu.vmware.com/kubernetes-grid`.

[6] Tanzu mission control. `https://tanzu.vmware.com/mission-control`.