Teradata® AppCenter 1.10

Customer Installation and Upgrade Guide for vSphere

Release Date: March 2020

Product ID: B035-1117-030K

Copyright Reference

Product Safety

Safety type	Description
Notice:	Indicates a situation which, if not avoided, could result in damage to property, such as to equipment or data, but not related to personal injury.
Caution:	Indicates a hazardous situation which, if not avoided, could result in minor or moderate personal injury.
Warning:	Indicates a hazardous situation which, if not avoided, could result in death or serious personal injury.

Warranty Disclaimer

Except as may be provided in a separate written agreement with Teradata or required by applicable law, the information available from the Teradata Documentation website or contained in Teradata information products is provided on an "as-is" basis, without warranty of any kind, either express or implied, including the implied warranties of merchantability, fitness for a particular purpose, or noninfringement.

The information available from the Teradata Documentation website or contained in Teradata information products may contain references or cross-references to features, functions, products, or services that are not announced or available in your country. Such references do not imply that Teradata Corporation intends to announce such features, functions, products, or services in your country. Please consult your local Teradata Corporation representative for those features, functions, products, or services available in your country.

The information available from the Teradata Documentation website or contained in Teradata information products may be changed or updated by Teradata at any time without notice. Teradata may also make changes in the products or services described in this information at any time without notice.

Feedback

To maintain the quality of our products and services, e-mail your comments on the accuracy, clarity, organization, and value of this document to: docs@teradata.com.

Any comments or materials (collectively referred to as "Feedback") sent to Teradata Corporation will be deemed nonconfidential. Without any payment or other obligation of any kind and without any restriction of any kind, Teradata and its affiliates are hereby free to (1) reproduce, distribute, provide access to, publish, transmit, publicly display, publicly perform, and create derivative works of, the Feedback, (2) use any ideas, concepts, know-how, and techniques contained in such Feedback for any purpose whatsoever, including developing, manufacturing, and marketing products and services incorporating the Feedback, and (3) authorize others to do any or all of the above.

Revision History

Revision/Version*	Date	Comments
1.0	February 2020	Initial Release

Table of Contents

1 Requirements ... 3
- 1.1 Minimum Recommended Resources ... 3
- 1.2 Downloads ... 4

2 Install Kubernetes on vSphere using Kubekit 2.0.14 ... 6
- 2.1 Initializing and Editing Cluster Configuration ... 6
- 2.2 Editing Cluster Configuration ... 6
  - 2.2.1 [Optional] Assigning Static IP Addresses ... 8
- 2.3 Logging in to the vSphere Cluster ... 8
- 2.4 Installing RPM and Creating Kubernetes Cluster ... 8
- 2.5 Validating the Kubernetes Cluster ... 9

3 Cluster Administration ... 10
  - 3.1.1 Backing up Cluster Configuration ... 10
  - 3.1.2 Deleting Kubernetes cluster ... 10

4 Teradata AppCenter 1.10.0 Installation ... 11
- 4.1 Considerations and Prerequisites ... 11
  - 4.1.1 NAS Requirements ... 11
- 4.2 Preparing the Client ... 12
- 4.3 Preparing the Host ... 12
- 4.4 Installing AppCenter ... 13
  - 4.4.1 Extracting Binary ... 13
  - 4.4.2 Setting Environment Variables for Your Cluster ... 14
  - 4.4.3 Providing Signed Certificate ... 14
  - 4.4.4 Initializing the Cluster ... 17
  - 4.4.5 Installing Services ... 18

5 Post-Installation LDAP Configuration ... 25
- 5.1 Adding Built-In OpenLDAP Configuration for Testing ... 25
- 5.2 Adding Corporate LDAP Configuration ... 26

6 Uninstall AppCenter ... 28

7 Back Up AppCenter ... 29

8 Restore AppCenter ... 30

9 Upgrading AppCenter (1.8.7 and Later) ... 31

10 Troubleshooting ... 32
- 10.1 Rotating Certificates (tls.key, tls.crt) ... 32

1 Requirements

- Functioning VMWare vSphere environment capable of running virtual machines (VMs) with the capacity described in Minimum Recommended Resources on page 3.
- VMWare vSphere with capacity to support the minimum requirements for AppCenter.
- Linux or OSX(Darwin) client machine from which you can perform installation.
- Fully qualified domain name (FQDN). For example: appcenter.example.com

  This domain name is used to access AppCenter from the browser and is required to create certificates for TLS communication with the cluster. Make sure the domain name is resolvable by the infrastructure DNS service.
- Signed certificate for the FQDN.

  AppCenter requires signed certificates that need to be ordered ahead of time. If self-signed certificates are used, each user will be required to accept a warning from the browser at first login.

  Note: This is a security requirement. Coordinate with your network security team for certificate configuration and standards for your environment.
- Network Attached Storage (NAS)

  NAS is required by Multi-Node cluster and is used for high availability and persistence. The minimum required size for NAS is 1500 GB.
- vSphere admin credentials.
- [Optional] An adequate number of IP addresses, allocated in advance if you plan to use static IP addresses.

1.1 Minimum Recommended Resources

Teradata recommends a three-master, three-worker, multi-node Kubernetes cluster. This is the minimum requirement for a production-grade setup with high availability and persistence.

Nodes	vCPUs	RAM (GB)	Disk (GB)
Master-1	15	30	500
Master-2	15	30	500
Master-3	15	30	500
Worker-1	15	30	500
Worker-2	15	30	500
Worker-3	15	30	500
Total	**90**	**180**	

AppCenter requires 65 CPU and approximately 100 GB RAM maximum to operate in high-availability mode. The remaining resources are used for apps/scripts. If you run all your apps/scripts with 0.1 CPU and 500 MB memory, you can run approximately 160 apps/scripts. If you want to run more than 160 apps/scripts, you can provision the cluster with higher resources.

1.2 Downloads

You must provide a client machine where the installation can be run. This can be a Linux/OSX (Darwin) machine/VM.

1. Create an installation directory on the client machine:

```
$ mkdir -p /var/opt/teradata/appcenter
```

2. To download the AppCenter, KubeKit, and KubeOS binaries to the installation directory, log on to https://access.teradata.com and select **Update Your Software** under **DOWNLOADS**.
3. Download the KubeOS binary:
   a. In **Search Downloads**, type `tdc-vmware-kube-os`.
   b. Select and download `tdc-vmware-kube-os-sles12-sp3-01.42-19.07.04.ova`.

      The vSphere admin needs to upload this KubeOS OVA to vSphere. KubeKit will use this OVA to spin up VMs as part of the installation process.
4. Download the KubeKit binary:
   a. In **Search Downloads** type `kubekit`.
   b. Select and download version 2.0.14.

      KubeKit is distributed as a go binary compiled for either Linux, OSX (Darwin) or Windows. Be sure to download and use the correct binary for the platform where you will be running KubeKit.
   c. Extract the contents of the file.

      The file contains the binary, an RPM file used during cluster configuration, and some informational text files.
   d. Copy the binary to the filename `kubekit` somewhere in the executable path.
   e. Verify you can run KubeKit and confirm the correct version is being used:

```
$ tar xvfz kubekit_2.0.14_linux_amd64.tgz
 kubekit_2.0.14_linux_amd64
 README.md
 KNOWN_ISSUES.md
 CONTRIBUTING.md
 USER_GUIDE.md
 KUBEKIT_CHANGELOG.md
 kubekit-2.0.14-20190808-150802.rpm

 # COPY KUBEKIT EXECUTABLE
 $ cp kubekit_2.0.14_linux_amd64 /usr/local/bin/kubekit

 # VERIFY THE BINARY IS REACHABLE
 $ which kubekit
 /usr/local/bin/kubekit

 # VERIFY THE VERSION
 $ kubekit version --verbose
 KubeKit v2.0.14
 Kubernetes version: 1.12.10
 Docker version: 18.06.1-ce
 etcd version: v3.3.13
```

5. Download the AppCenter binary:

   a. In **Search Downloads**, type `appcenter-all`.
   b. Select and download `appcenter-all__sles12_x8664.1.10.0.0.build.43.af0f8ae.tar.gz`.

   The package includes:

Package Name	Description
`appctl-1.10.0+build.19.057a2aa.linux.amd64.tgz`	Binary to install/upgrade/uninstall AppCenter Linux version.
`appctl-1.10.0+build.19.057a2aa.darwin.amd64.tgz`	Binary to install/upgrade/uninstall AppCenter macOS version.
`init-image-bundle-1.10.0+build.17.97cd1ca.tar.gz`	Docker images of registry, Minio and Tiller.
`platform-image-bundle-1.10.0+build.17.97cd1ca.tar.gz`	Docker images for AppCenter platform.
`appcenter-image-bundle-1.10.0+build.17.97cd1ca.tar.gz`	Docker images for AppCenter application services.
`init-1.10.0+build.78.83d5fe3.tgz`	Chart files to initialize AppCenter.
`platform-1.10.0+build.78.83d5fe3.tgz`	Chart files to install AppCenter platform.
`appcenter-1.10.0+build.78.83d5fe3.tgz`	Chart file to install AppCenter application services.
`appcenter-backup-1.10.0+build.78.83d5fe3.tgz`	Chart file to take on-demand backup.
`appcenter-restore-1.10.0+build.78.83d5fe3.tgz`	Chart file to restore a backup.
`Override template files`	Can be used as templates during installation: • `Init-override.yaml` • `platform-override.yaml` • `application-services-override.yaml`

6. Download Docker for the operating system of your client machine (Version 18.06.1-ce and above):

Docker Version	Download Location
Docker for Linux	https://docs.docker.com/install/linux/ubuntu/
Docker Mac	https://docs.docker.com/docker-for-mac/

7. Download kubectl (Version 1.12.3 and above):

Kubectl Version	Download Location
`Kubectl for Linux`	https://kubernetes.io/docs/tasks/tools/install-kubectl/#install-kubectl-on-linux
`Kubectl for Mac`	https://kubernetes.io/docs/tasks/tools/install-kubectl/#install-kubectl-on-macos

2 Install Kubernetes on vSphere using Kubekit 2.0.14

2.1 Initializing and Editing Cluster Configuration

1. Initialize the configuration by specifying the cluster name and platform, where `appcenter` is the cluster name and `vsphere` is the platform:

   ```
   $ kubekit init appcenter --platform vsphere
   ```

   A text editor opens to allow you to edit the default cluster configuration.

   Note: The cluster config file is in YAML format. You must maintain existing indentation.

2.2 Editing Cluster Configuration

1. Edit the following cluster configuration variables for your environment. For help identifying the values to use, contact your vSphere admin.

Variable	Variable Description and Command Example	Considerations
`datacenter`	Name of the VMware data center in which the cluster will be created. Example: `vagrant`	Virtual data center is a container for all inventory objects required for operating virtual machines (VMs).
`datastore`	Name of the datastore associated with the datacenter. Example: `vgrnt_dsc/vgrnt03`	Datastore is a logical container that stores virtual machine files and other files required for virtual-machine operations.
`resource_pool`	Name of the resource pool present in the datacenter. Example: `vgrnt_01/Resources/vagrant01` If you do not have a resource pool, use `datacenter/Resources` as the value.	You can use resource pools to hierarchically partition available CPU and memory resources of a standalone host or a cluster.
`vsphere_net`	Network with which the VMs need to be associated. Example: `dvpg_vm_550`	Networks provide a method of communication among virtual machines. Machines will be connected logically using this network so data can be sent and received among them.
`folder`	Name of the folder in which the VMs need to be deployed. Example: `Discovered virtual machine/folder_name`	If not provided, the VMs will be created in the root location.

Variable	Variable Description and Command Example	Considerations
`template_name`	Name of the KubeOS OVA to be used as a template for the VMs. Example:`<use the latest>`	
`cpus`	Number of CPUs to be allocated to each node.	For more information, see Requirements on page 3.
`memory`	Amount of memory to be allocated to each node.	For more information, see Requirements on page 3.
`root_vol_size`	Size of root volume for each node.	Minimum recommended configuration is 500 GB.
`private_key_file`	Location of a private key.	Private key will be used to set up the SSH access for the cluster. If you don't have a private key, generate one using `ssh-keygen`.
`public_key_file`	Location of a public key.	The public key will be needed for AppCenter installation. If you don't have a public key, generate one using `ssh-keygen`.

`node_pools:`
`master:`
`count:`
`worker`
`count:`
	Number of master and worker nodes you want to deploy in this environment.	Minimum recommended configuration is 3 masters and 3 workers.

`config:`
`host_timezone`
`controlplane_timezone`
	Time zone used when AppCenter runs scheduled jobs.
	Example: `host_timezone: "Asia/Tokyo"`
	`controlplane_timezone: "Asia/Tokyo"`
		If you do not provide a time zone, AppCenter defaults to UTC.

`config:` `rook_enabled`	Example: `rook_enabled: "false"`	Set the variable to `false` to disable Rook. You must set the variable to false for vSphere installations.
`config:` `master_schedulable_enabled`	Enable master scheduling. Example: `master_schedulable_enabled :true`	Required for AppCenter installation. Schedules pods on the master node.

2. Save and close the cluster configuration.

2.2.1 [Optional] Assigning Static IP Addresses

If you are not using DHCP and want to use static IP addresses for your VMs, you can add the IP information. For this option, KubeKit version 2.0.14 or later is required.

Note: Specifying hostname is optional.

1. Add the IP information in the following format:

```
node_pools:
  master:
    count: 3
    contents:
    - ip: 10.25.37.21
      hostname: master1
    - ip: 10.25.37.22
      hostname: master2
    - ip: 10.25.37.23
      hostname: master3
    ip_netmask: 23
    ip_gateway: 10.25.37.254
  worker:
    count: 3
    contents:
    - ip: 10.25.37.24
    - ip: 10.25.37.25
    - ip: 10.25.37.26
    ip_netmask: 23
    ip_gateway: 10.25.37.254
```

2.3 Logging in to the vSphere Cluster

1. Get the login information from your vSphere admin, then log in:

```
$ kubekit login appcenter
Enter server []: <vSphere URL>
Enter username []: username@vsphere.local
Enter password []: very-secure-password
```

2.4 Installing RPM and Creating Kubernetes Cluster

Kubernetes cluster setup will take at least 15 minutes.

When running the `kubekit apply` command, you will specify the RPM file that was included in the distribution file that was previously downloaded and expanded.

If you see errors in the output, or KubeKit exits before indicating the environment has been successfully configured, search for `FAILED` or `ERROR` in the log file to see what went wrong. The file exists in the same location from which you run the `kubekit apply` command.

1. Set up the Kubernetes cluster:

```
$ kubekit apply appcenter -f kubekit-2.0.14-20190808-150802.rpm --log appcenter.log --
debug
```

2.5 Validating the Kubernetes Cluster

1. If the KubeKit installation completes without errors, confirm cluster access and verify that the nodes are in `Ready` state and pods are in `Running` state:

```
$ eval $(kubekit get env appcenter)

$ kubectl get nodes
NAME                  STATUS   ROLES    AGE   VERSION
appcenter-master-01   Ready    master   16m   v1.12.10
appcenter-master-02   Ready    master   16m   v1.12.10
appcenter-master-03   Ready    master   16m   v1.12.10
appcenter-worker-01   Ready    worker   16m   v1.12.10
appcenter-worker-02   Ready    worker   16m   v1.12.10
appcenter-worker-03   Ready    worker   16m   v1.12.10

$ kubectl get pods --all-namespaces
```

2. Make sure all Kubernetes master nodes are schedulable:
   a. Get all nodes:

```
$ kubectl get nodes
```

   b. For all nodes that have ROLES as master:

```
$ kubectl describe node <NODE_NAME> | grep Taints
```

   Taints should be set to: `"".` If master is not schedulable, Taints will have this value:

   `node.kubernetes.io/not-ready:NoSchedule`

   Note: If not all Kubernetes master nodes are schedulable, you must reapply KubeKit with the master_schedulable_enabled flag set to true. For more information, see Install Kubernetes on vSphere using Kubekit 2.0.14 on page 6.

   c. Make sure that Rook is not installed:

```
$ kubectl get ns | grep rook
$ kubectl get pods -n <namespace from above list>
```

   If Rook is installed, you need to reapply KubeKit with the `rook_enabled` flag set to `false`. For more information, see Initializing and Editing Cluster Configuration on page 6.

   If multiple Rook namespaces are returned, make sure no pods are deployed in any of them.

   Rook resources are not considered for resource estimation.

   d. Make sure every node has minimum recommended resources as described in Requirements on page 3.
   e. From each Kubernetes node, check node capacity:

```
kubectl describe node <NODE_NAME> | grep -a6 Capacity
```

3. If errors are encountered or KubeKit exits prior to a successful configuration, open `appcenter.log` to search for the reasons.

3 Cluster Administration

3.1.1 Backing up Cluster Configuration

You should always keep a backup copy of the cluster information in a directory. This information can be valuable if you encounter issues during KubeKit installation.

Note: Treat the backup file with great care because it contains credentials for the cluster you created.

1. Run the export command:

```
$ kubekit get env appcenter
export KUBECONFIG=/Users/foo/.kubekit.d/clusters/f355ee04-8f79-4f43-53b1-
700196970a48/certificates/kubeconfig
# Run this command to configure your shell:
# eval "$(kubekit get env appcenter)"
```

2. Run one of the following backup commands, without `certificates/kubeconfig`:

Available Utility	Command
TAR	`$ tar cvfz myclustername-backup.tgz /Users/foo/.kubekit.d/clusters/f355ee04-8f79-4f43-53b1-700196970a48/`
Zip	`$ zip -r myclustername-backup.zip /Users/foo/.kubekit.d/clusters/f355ee04-8f79-4f43-53b1-700196970a48/`

3.1.2 Deleting Kubernetes cluster

You can delete the cluster if you no longer need it or want to create a new one. Deleting the cluster removes the VMs.

Warning: This cannot be undone. The cluster will be destroyed.

1. Run the delete command:

```
$ kubekit delete appcenter
Do you want to destroy cluster "appcenter" [type 'yes']?: yes
```

4 Teradata AppCenter 1.10.0 Installation

4.1 Considerations and Prerequisites

- Install AppCenter from the same client machine from which you installed KubeKit. That client machine should have all required tools referenced in the installation instructions.
- Use the Kubernetes configuration file, `remote-kubeconfig` to connect to the cluster and perform Kubernetes operations remotely. The admin who created the cluster should have a copy of `remote-kubeconfig`. Upon successful installation of Kubernetes, Kubekit stores `remote-kubeconfig` in the KubeKit home directory on your machine (`~/.kubekit/`).

  If you have access to the cluster, you can use the following command to generate `remote-kubeconfig` from one of the nodes inside the cluster:

  ```
  $ kubectl config view --flatten > remote-kubeconfig
  ```
- You have the SSH key that was used during KubeKit installation.
- The time-zone on the Kubernetes nodes is correct. The time-zone is used by AppCenter to start apps that are based on a schedule. The time-zone must be provided during KubeKit configuration. If the time-zone is not correct, reapply KubeKit configuration and select the desired time-zone. You can get the time-zone by running the `date` command from any node in the cluster.
- You have a fully qualified domain name and the signed certificates for that domain. If you want to provide signed certificates for the domain of your choice, see Providing Signed Certificate on page 14. The certificates must be available before installation is started. Signed certificates for the domain of your choice often take longer to obtain, so start that process well in advance.
- You have network-attached storage (NAS).

4.1.1 NAS Requirements

- NAS is required for a multi-node cluster. It is used for high availability and persistence.
- NAS can be mounted using a network file system (NFS). It should be mounted on all the nodes before you install AppCenter.
- The minimum required size for NFS is 1500 GB. NFS should be mounted on `/var/lib/docker/teradata/data` on each node.

4.1.1.1 Mounting NAS Example

The following is an example for mounting NAS using NFS on all nodes in the AppCenter cluster:

```
$ mkdir -p /var/lib/docker/teradata/data
$ sudo mount -t nfs <IP-ADDRESS-OF-NFS-SERVER-HERE>:<LOCATION-OF-FOLDER-ON-NFS-SERVER>
/var/lib/docker/teradata/data
```

4.2 Preparing the Client

1. Make sure `remote-kubeconfig` has been copied to `/var/opt/teradata/appcenter` on the client machine from which you are going to run installation.
2. On the client machine, set the path environment variable for `kubeconfig`, `appctl`, and `kubectl`:

```
$ export KUBECONFIG=/var/opt/teradata/appcenter/remote-kubeconfig
```

Note: Setting the environment variable must be done for each logged-in SSH session. Previous settings are not maintained once the session is terminated.

4.3 Preparing the Host

Perform all steps from the client machine.

1. Label the nodes:
   a. List nodes in your cluster:

```
kubectl get nodes -o wide
```

   b. For each node in the cluster, run the following commands. All nodes should have the same labels.

```
kubectl label node <NODE_NAME> appcenter/consul-0=true
kubectl label node <NODE_NAME> appcenter/consul-1=true
kubectl label node <NODE_NAME> appcenter/consul-2=true
kubectl label node <NODE_NAME> appcenter/minio-0=true
kubectl label node <NODE_NAME> appcenter/minio-1=true
kubectl label node <NODE_NAME> appcenter/minio-2=true
kubectl label node <NODE_NAME> appcenter/minio-3=true
kubectl label node <NODE_NAME> appcenter/elastic-data-0=true
kubectl label node <NODE_NAME> appcenter/elastic-data-1=true
kubectl label node <NODE_NAME> appcenter/elastic-data-2=true
kubectl label node <NODE_NAME> appcenter/elastic-master-0=true
kubectl label node <NODE_NAME> appcenter/elastic-master-1=true
kubectl label node <NODE_NAME> appcenter/elastic-master-2=true
kubectl label node <NODE_NAME> appcenter/postgres=true
kubectl label node <NODE_NAME> appcenter/grafana=true
kubectl label node <NODE_NAME> appcenter/prometheus-0=true
kubectl label node <NODE_NAME> appcenter/prometheus-1=true
```

   c. Verify labels have been applied to all nodes:

```
$ kubectl get nodes --show-labels
```

2. Create the following folders in the NAS-mounted location for persistent volumes:

```
ssh root@<KUBE_NODE_IP> -i <SSH_KEY_USED_TO_INSTALL_KUBERNETES_CLUSTER> \
"mkdir -p /var/lib/docker/teradata/data && \
cd /var/lib/docker/teradata/data && \
mkdir -p minio/minio-0 minio/minio-1 minio/minio-2 minio/minio-3 \
consul/consul-0 consul/consul-1 consul/consul-2 \
elasticsearch/elastic-master-0 elasticsearch/elastic-master-1 \
elasticsearch/elastic-master-2 elasticsearch/elastic-data-0 \
elasticsearch/elastic-data-1 elasticsearch/elastic-data-2  \
postgres/postgres-0 grafana/grafana-0 \
prometheus/prometheus-0 prometheus/prometheus-1 \
alertmanager/alertmanager-0 alertmanager/alertmanager-1"
```

Note: To get the node IP address required for folders, run `$ kubectl get nodes -o wide`, then replace `<KUBE_NODE_IP>` with the IP of any node.

3. Make sure all the folders are properly created by doing the following:
   a. Using the SSH key, run:

```
ssh root@< KUBE_NODE_IP> -i <SSH_KEY_USED_TO_INSTALL_KUBERNETES_CLUSTER>
```

   b. Go to `/var/lib/docker/teradata/data`.

4.4 Installing AppCenter

Perform all installation steps from the same client machine from which you installed KubeKit.

4.4.1 Extracting Binary

1. Extract tar.gz in /var/opt/teradata/appcenter binary:

```
appcenter-all__sles12_x8664.1.10.0.0.build.43.af0f8ae.tar.gz
```

2. Make sure all files mentioned in Downloads on page 4 are present.
3. Make `$chmod +x appctl-<version>` executable:

```
$ chmod +x appctl
```

4. Do one of the following:
   - Add `appctl` to `PATH`.
   - Copy `appctl` to `/usr/local/bin`:

```
$ cp appctl-<version> /usr/local/bin/appctl
```

4.4.2 Setting Environment Variables for Your Cluster

You must set the environment variable for each logged-in SSH session. Previous settings are not maintained when the session is terminated.

1. Set the following variables:

Variable	Action
KUBECONFIG	a. Make sure KUBECONFIG is set as described in Preparing the Host on page 12. If not, run: `$ export KUBECONFIG=/var/opt/teradata/appcenter/remote-kubeconfig`

APPCTL_HOME	a. Make a directory for the cluster:
	`$ mkdir -p ~/.appctl/<ANY UNIQUE CLUSTER NAME>`
	b. Export APPCTL_HOME:
	`$ export APPCTL_HOME=~/.appctl/<ANY UNIQUE CLUSTER NAME>`
	If there are multiple clusters and you are using the same client machine to run the installation, Teradata recommends you export a different value for `APPCTL_HOME` for each cluster.

APPCTL_DOMAIN	Domain name is required to create self-signed certificates for Transport Layer Security (TLS) communication with the cluster. This is the domain name you put in your browser to access the AppCenter UI components. A self-signed certificate will be automatically generated using this domain name during the installation process. The browser will display a warning if certificates are self-signed. a. Make sure that the domain name is resolvable by your infrastructure's DNS service. Adding an entry in `/etc/hosts` will not work. **Note: AppCenter will not work without a valid DNS configured. DNS should have entries for all the master nodes. For example, if there are three master nodes, there should be three IP address entries added to DNS for the chosen APPCTL_DOMAIN.**

4.4.3 Providing Signed Certificate

Teradata recommends providing signed certificates for the domain of your choice.

Appctl accepts user-provided certificates that can be used to access the AppCenter UI from a browser. If you do not provide the certificates, appctl generates a self-signed certificate.

Note: This is a security requirement. Coordinate with your network security team for certificate configuration and standards for your environment.

4.4.3.1 Generating a Signed Certificate

Generating a signed certificate involves creating a certificate signing request (CSR).

1. Create a `san.cnf` CSR config file:

```
[ req ]
default_bits       = 2048
distinguished_name = req_distinguished_name
req_extensions     = req_ext
[ req_distinguished_name ]
countryName                = Country Name (2 letter code)
stateOrProvinceName        = State or Province Name (full name)
localityName               = Locality Name (eg, city)
organizationName           = Organization Name (eg, company)
commonName                 = Common Name (e.g. server FQDN or YOUR name)
[ req_ext ]
subjectAltName = @alt_names
[alt_names]
DNS.1   = appcenter.example.com
DNS.2   = *.appcenter.example.com
```

Note: Make sure the value of DNS.1 is the same as APPCTL_DOMAIN. The value of DNS.2 should be *.APPCTL_DOMAIN.

2. From your client machine, generate the server.key:

```
$ openssl req -out server.csr -newkey rsa:2048 -nodes -keyout server.key -config san.cnf
```

3. Enter the following CSR details when prompted:

CSR Detail	Description
`Common Name`	Fully qualified domain name (FQDN) you want to secure with the certificate. For example: `appcenter.example.com`.
`Organization`	Full legal name of your organization, including the corporate identifier.
`Organization Unit (OU)`	Your department. For example: `Information Technology` or `Website Security`.
`City or Locality`	Locality or city where your organization is legally incorporated. Do not abbreviate.
`Country`	Official two-letter country code where your organization is legally incorporated. For example: US or CH
`State or Province`	State or province where your organization is legally incorporated. Do not abbreviate.

For example:

```
$ openssl req -out server.csr -newkey rsa:2048 -nodes -keyout server.key -config san.cnf
Generating a 2048 bit RSA private key
.....................+++
.............................+++
writing new private key to 'private.key'
-----
You are about to be asked to enter information that will be incorporated
into your certificate request.
What you are about to enter is what is called a Distinguished Name or a DN.
There are quite a few fields but you can leave some blank
For some fields there will be a default value,
If you enter '.', the field will be left blank.
-----
Country Name (2 letter code) []:AA
State or Province Name (full name) []:BB
Locality Name (eg, city) []:CC
Organization Name (eg, company) []:Example
Common Name (e.g. server FQDN or YOUR name) []:appcenter.example.com
```

Note: You are not required to enter a password or passphrase. This optional field is for applying additional security to your key pair.

4. Provide the `server.csr` to the certificate authority to get a signed certificate. Make sure that you get the root and intermediate certificates from the certificate authority. The certificate needs to have the following order:

```
-----BEGIN MY CERTIFICATE-----
-----END MY CERTIFICATE-----
-----BEGIN INTERMEDIATE CERTIFICATE-----
-----END INTERMEDIATE CERTIFICATE-----
-----BEGIN ROOT CERTIFICATE-----
-----END ROOT CERTIFICATE-----
```

5. Rename the certificate received from certificate authority to `tls.crt`.
6. `Rename the server.key` generated previously to `tls.key`.
7. Make sure the certificates have the right permissions:

```
$ chmod 400 tls.key
$ chmod 400 tls.crt
```

8. Place `tls.crt` and `tls.key` in the `APPCTL_HOME` directory:

```
$ cp tls.crt $APPCTL_HOME
$ cp tls.key $APPCTL_HOME
```

4.4.4 Initializing the Cluster

1. In the override files, add the keys, passwords, and configurations.
2. Using any standard text editor, open `init-override.yaml` that was downloaded as part of the AppCenter bundle and make sure the required entries exist:

```
consul:
  storageClass: manual
  persistence:
    pv:
      location: /var/lib/docker/teradata/data/consul
  client:
    hostPath: /var/lib/docker/teradata/consul/consul-agent

global:
  nodeSelector:
    nodetype: appcenter
  enableNodeSelector: false
  rook:
    rookEnabled: false
  accessKey: # Refer Table 1.0 - This is a mandatory value
  secretKey: # Refer Table 1.0 - This is a mandatory value

minio:
  persistence:
    pv:
      location: /var/lib/docker/teradata/data/minio
  resources:
    limits:
      cpu: 800m
      memory: 1500Mi
    requests:
      cpu: 500m
      memory: 800Mi

tags:
  minio: true
```

Note: The template files shown in this init-override.yaml file are examples and might not match the files downloaded as part of the actual bundle. The files in the actual bundle are the most up-to-date.

3. Add the following key values:

Table 1.0

Key	Description
`accessKey`	Needed for Minio's S3 bucket authentication. Alphanumeric string. Minimum size is 3.
`secretKey`	Needed for Minio's S3 bucket authentication. Alphanumeric string. Minimum size is 8.

Note: Examples are included in the tar ball.

4. Initialize the cluster:

```
$ appctl init --chart init-1.10.0+build.78.83d5fe3.tgz --images init-image-bundle-
1.10.0+build.17.97cd1ca.tar.gz -f init-override.yaml --reset-values
```

5. Make sure all the pods are in "Running"/"Completed" state:

```
$ kubectl get pods --all-namespaces
```

use `appctl init -h` for more details on parameters that can be used as part of the `init` command.

6. Upload the Docker images that will be used to bring up all AppCenter services:

```
$ appctl upload --images platform-image-bundle-1.10.0+build.17.97cd1ca.tar.g
$ appctl upload --images appcenter-image-bundle-1.10.0+build.17.97cd1ca.tar.gz
```

The images are pushed to the Docker registry that was part of initializing the cluster.

7. [Optional] For more details on parameters that can be used as part of the upload command, run:

```
use appctl upload -h
```

4.4.5 Installing Services

1. Using any standard text editor, open `platform-override.yaml` that was downloaded as part of the AppCenter bundle and make sure the required entries exist:

```
elasticsearch:
  master:
    resources:
      limits:
        cpu: 500m
        memory: 2Gi
      requests:
        cpu: 250m
        memory: 2Gi
    heapMemory: 1024m
  data:
    resources:
      limits:
        cpu: 500m
        memory: 2Gi
      requests:
        cpu: 250m
        memory: 2Gi
    heapMemory: 1024m
  client:
    resources:
      limits:
        cpu: 500m
        memory: 2Gi
      requests:
        cpu: 250m
        memory: 2Gi
    heapMemory: 1024m
```

```
filebeat:
  hostPath: /var/lib/docker/teradata/elasticsearch/filebeat
  resources:
    limits:
      cpu: 1000m
      memory: 1500Mi
    requests:
      cpu: 300m
      memory: 200Mi

global:
  nodeSelector:
    nodetype: appcenter
  enableNodeSelector: false
  rook:
    rookEnabled: false
  backup:
    encKey:  # Refer Table 2.0 - This is a mandatory value

kibana:
  basicAuthUsername: admin
  basicAuthPassword: # Refer Table 2.0 - This is a mandatory value

grafana:
  basicAuthUsername: admin
  basicAuthPassword: # Refer Table 2.0 - This is a mandatory value
  storageClass: manual
  persistence:
    location: /var/lib/docker/teradata/data/grafana


alert-manager:
  basicAuthUsername: admin
  basicAuthPassword: # Refer Table 2.0 - This is a mandatory value
  storageClass: manual
  persistence:
    pv:
      location: /var/lib/docker/teradata/data/alertmanager

postgres:
  resources:
    requests:
      cpu: 300m
      memory: 200Mi
  storageClass: manual
  persistence:
    location: /var/lib/docker/teradata/data/postgres
```

```
prometheus:
  resources:
    requests:
      cpu: 300m
      memory: 512Mi
  basicAuthUsername: admin
  basicAuthPassword: # Refer Table 2.0 - This is a mandatory value
  storageClass: manual
  persistence:
    pv:
      location: /var/lib/docker/teradata/data/prometheus
  nodeExporter:
    resources:
      limits:
        cpu: 300m
        memory: 512Mi
      requests:
        cpu: 100m
        memory: 100Mi

teradataExternalService:
  enabled: false
```

Note: The template files shown in this platform-override.yaml file are examples and might not match the files downloaded as part of the actual bundle. The files in the actual bundle are the most up-to-date.

2. Add the following key values:

Table 2.0

Key	Description
`global.backup.encKey`	Key for backup encryption. Same value is needed in `appcenter-override.yaml`. Alphanumeric string. Minimum size is 3.
`grafana.basicAuthPassword`	Grafana password for authentication. Username is `admin`. Alphanumeric string. Minimum length is 3.
`kibana.basicAuthPassword`	Kibana password needed for authentication. Username is `admin`. Alphanumeric string. Minimum length is 3.
`prometheus.basicAuthPassword`	Prometheus password needed for authentication. Username is `admin`. Alphanumeric string. Minimum length is 3.
`alert-manager.basicAuthPassword`	Alert Manger password needed for authentication. Username is `admin`. Alphanumeric string. Minimum length is 3.

3. [Optional] Use this command to generate random char strings for any of the passwords/keys:

```
$head /dev/urandom | LC_CTYPE=C tr -dc A-Za-z0-9 | head -c <NUMBER-OF-CHARACTERS> ; echo
''
```

4. [Optional] Encrypt the override file, which contains passwords:

```
$ appctl encrypt platform-override.yaml
Enter password:
Confirm password: encrypting platform-override.yaml...
```

Note: If you encrypt the override file, the password cannot be retrieved after installation. Make sure you save the password in a secure location so you can view the contents of override file after installation. To decrpyt the override file, use: `appctl decrypt <path/to/file>`

5. Using any standard text editor, open `application-services-override.yaml` that was downloaded as part of the AppCenter bundle and make sure the required entries exist:

```
grafana-init:
  dockerMount: /dev/sda1
  rootMount: /dev/sda1

global:
  accessKey: # Refer Table 3.0 - This is a mandatory value, use same value from init
override file
  secretKey: # Refer Table 3.0 - This is a mandatory value, use same value from init
override file
  nodeSelector:
    nodetype: appcenter
  enableNodeSelector: false
  rook:
    rookEnabled: false

  backup:
    encKey: # Refer Table 3.0 - This is a mandatory value, use same value from platform
override file

  appService:
    pgUsername: appservice
    pgPassword: # Refer Table 3.0 - This is a mandatory value
    applicationKey: appservice
    clientSecret: # Refer Table 3.0 - This is a mandatory value

  auditService:
    pgUsername: auditservice
    pgPassword: # Refer Table 3.0 - This is a mandatory value
    applicationKey: auditservice
    clientSecret: # Refer Table 3.0 - This is a mandatory value

  dictionaryService:
    pgUsername: dictionaryservice
    pgPassword: # Refer Table 3.0 - This is a mandatory value
    applicationKey: dictionaryservice
    clientSecret: # Refer Table 3.0 - This is a mandatory value

  systemService:
    pgUsername: systemservice
    pgPassword: # Refer Table 3.0 - This is a mandatory value
    applicationKey: systemservice
    clientSecret: # Refer Table 3.0 - This is a mandatory value

  userService:
    pgUsername: userservice
    pgPassword: # Refer Table 3.0 - This is a mandatory value
    applicationKey: userservice
    clientSecret: # Refer Table 3.0 - This is a mandatory value
```

```
  notificationService:
    pgUsername: notificationservice
    pgPassword: # Refer Table 3.0 - This is a mandatory value
    applicationKey: notificationservice
    clientSecret: # Refer Table 3.0 - This is a mandatory value

  jobService:
    pgUsername: jobservice
    pgPassword: # Refer Table 3.0 - This is a mandatory value
    applicationKey: jobservice
    clientSecret: # Refer Table 3.0 - This is a mandatory value

  keycloak:
    pgUsername: keycloak
    pgPassword: # Refer Table 3.0 - This is a mandatory value

  root_password: # Refer Table 3.0 - This is a mandatory value

app-service:
  appcenterByocDisabled: false
  migration:
    enable: true

keycloak:
  keycloak:
    password: # Refer Table 3.0 - This is a mandatory value
```

Note: The template files shown in this application-services-override.yaml file are examples and might not match the files downloaded as part of the actual bundle. The files in the actual bundle are the most up-to-date.

6. Add the following key values:

Table 3.0

Key	Description
`global.backup.encKey`	Key for backup encryption. Copy `global.backup.encKey` from `platform-override.yaml`. The key for backup encryption needs to be the same as the key from `platform-override.yaml`.
`global.appService.pgPassword`	Postgres password for `app-service`. Alphanumeric string. Minimum size is 3.
`global.appService.clientSecret`	Postgres client secret for `app-service`. Alphanumeric string. Minimum size is 3.
`global.auditService.pgPassword`	Postgres password for `audit-service`. Alphanumeric string. Minimum size is 3.
`global.auditService.clientSecret`	Postgres client secret for `audit-service`. Alphanumeric string. Minimum size is 3.
`global.dictionaryService.pgPassword`	Postgres password for `dictionary-service`. Alphanumeric string. Minimum size is 3.

Table 3.0

Key	Description
`global.dictionaryService.clientSecret`	Postgres client secret for `dictionary-service`. Alphanumeric string. Minimum size is 3.
`global.systemService.pgPassword`	Postgres password for `system-service`. Alphanumeric string. Minimum size is 3.
`global.systemService.clientSecret`	Postgres client secret for `system-service`. Alphanumeric string. Minimum size is 3.
`global.userService.pgPassword`	Postgres password for `user-service`. Alphanumeric string. Minimum size is 3.
`global.userService.clientSecret`	Postgres client secret for `user-service`. Alphanumeric string. Minimum size is 3.
`global.notificationService.pgPassword`	Postgres password for `notification-service`. Alphanumeric string. Minimum size is 3.
`global.notificationService.clientSecret`	Postgres client secret for `notification-service`. Alphanumeric string. Minimum size is 3.
`global.root_password`	AppCenter root login password. Alphanumeric string. Minimum size is 3.
`global.keycloak.pgPassword`	Postgres password for `keycloak`. Alphanumeric string. Minimum size is 3.
`keycloak.keycloak.password`	Keycloak admin login password. Alphanumeric string. Minimum size is 5.
`global.accessKey`	Copy value from `init` override file.
`global.secretKey`	Copy value from `init` override file.

7. [Optional] Use this command to generate random char strings for any of the passwords/keys:

```
head /dev/urandom | LC_CTYPE=C tr -dc A-Za-z0-9 | head -c <NUMBER-OF-CHARACTERS> ; echo ''
```

8. [Optional] Encrypt the override file, which contains passwords:

```
$ appctl encrypt application-services-override.yaml
Enter password:
Confirm password: encrypting application-services-override.yaml...
```

Note: If you encrypt the override file, the password cannot be retrieved after installation. Make sure you save the password in a secure location so you can view the contents of override file after installation. To decrpyt the override file, use: `appctl decrypt <path/to/file>`

4.4.5.1 Installing Platform Services

1. Run:

```
$ appctl platform install --namespace td-platform --name platform platform-
1.10.0+build.78.83d5fe3.tgz -f platform-override.yaml --reset-values
```

2. If the override files are encrypted, provide the password:

```
$ appctl platform install --namespace td-platform --name platform platform-
1.10.0+build.78.83d5fe3.tgz -f platform-override.yaml --reset-value
Enter password:
```

4.4.5.2 Installing Application Services

1. Make sure all the pods are in a running or completed state:

```
$ kubectl get pods -n td-platform
```

2. When all pods are in running or completed state, install the services:

```
$ appctl platform install --namespace appcenter --name appcenter appcenter-
1.10.0+build.78.83d5fe3.tgz -f application-services-override.yaml --reset-values
```

3. If the override files are encrypted, provide the password:

```
$ appctl platform install --namespace appcenter --name appcenter appcenter-
1.10.0+build.78.83d5fe3.tgz -f application-services-override.yaml --reset-values
Enter password:
```

4. After installation of services, confirm all pods are in a running or completed state:

```
$ kubectl get pods --all-namespaces
```

5 Post-Installation LDAP Configuration

AppCenter includes built-in OpenLDAP with two users that you can use to get started with AppCenter before adding corporate LDAP users. The built-in LDAP should not be used for production environments.

AppCenter uses your corporate LDAP for user authentication and supports LDAP groups. You can add multiple LDAP configurations. You cannot delete users that you add. If a user is no longer part of a configured LDAP, AppCenter retains only the username in the app-service database to maintain their associated apps or scripts.

5.1 Adding Built-In OpenLDAP Configuration for Testing

AppCenter includes built-in OpenLDAP with two users that you can use to test AppCenter before adding your corporate LDAP. The built-in OpenLDAP configuration is optional and is not for production purposes. If you add corporate LDAP, the OpenLDAP configuration will still work with your instance of AppCenter.

Only the root user can add built-in OpenLDAP configuration.

1. Select  > **Settings** > **Authentication** >  , then complete the LDAP configuration fields with the following entries:

Setting	OpenLDAP Entry or Option
Server	`openldap`
URL	`openldap.appcenter.svc.cluster.local`
Port	`389`
Encryption	`None`
Base domain	`dc=example,dc=org`
Domain search user	`cn=admin,dc=example,dc=org`
Domain search password	`admin`
Vendor	**`Other`**
User object classes	`top, posixAccount, person`
Id field	`uid`
Name field	`uid`
Member of field	`uid`
Member field	`uid`
Email field	`uid`

Note: LDAP group fields are not applicable to built-in OpenLDAP.

The built-in LDAP includes the following two users for login:

Built-In Username	Password
`user1`	`user1`
`user2`	`user2`

Note: For the built-in OpenLDAP, you cannot add additional users.

5.2 Adding Corporate LDAP Configuration

User authentication is based on configured domains. AppCenter supports both LDAP and LDAPS and multiple LDAP domains and LDAP groups. If you added an OpenLDAP configuration, it will still work with your instance of AppCenter after you add corporate LDAP.

If a user is part of a configured LDAP domain, they can log into AppCenter and are added as a user automatically at that time.

Only the root user can add corporate LDAP configuration.

1. Select  > **Settings** > **Authentication** >  , then complete the LDAP configuration:

Setting	Required	Description	Example
Server	•	Display name of server.	`LDAP Server`
URL	•	LDAP or LDAPS server hostname.	`ldap.yourcompany.com` `ldaps.yourcompany.com`

Port	•	LDAP or LDAPS server port.	`389 (LDAP)`
			`636 (LDAPS)`
			`3268 (Active Directory Global Catalog, LDAP)`
			`3269 (Active Directory Global Catalog, LDAPS)`

Encryption		**None** provides no encryption for connectivity. **LDAPS** incorporates SSL for greater security.	
Base domain	•	Base DN for your tree. Can be shifted to restrict users from a single tree from logging in.	`DC=YOURCOMPANY,DC=COM`
Domain search user	•	User DN for user to connect to LDAP server.	`CN=AppCenter-User,OU=Service Accounts,DC=YOURCOMPANY,DC=COM`
Domain search password	•	Password for search user.	
Vendor	•	Vendor of your LDAP directory.	OpenLDAP would be `Other`, which includes OpenLDAP, Novell eDirectory, Red Hat's 389 Directory Service, or ApacheDS.
User object classes	•	LDAP object classes that identify your users.	The default, `person, OrganizationalPerson, user` should match most organizations.
Id field	•	Unique LDAP attribute within the directory to identify user accounts.	For Active Directory, this attribute could be `objectGUID` or `sAMAccountName`. For eDirectory, this attribute could be `GUID`. It could also be `CN` if `CN` is unique for your entire directory. If you are using `posixAccount`, it could be `posixAccount`.

Setting	Required	Description	Example
Name field	•	Username attribute.	For Active Directory, this attribute could be `sAMAccountName`. It could also be `CN` if that is your login name. If you are using `posixAccount`, it could be `posixAccount`.
Member of field	•	Attribute in a user entry for group membership. Should contain a DN pointing to the groups the user is a member of. Not all directories support reverse membership.	For Active Directory, this attribute is typically `memberOf`. For eDirectory, it is typically `groupMembership`.
Member field	•	Attribute in a group entry for user membership. Should contain a DN pointing to the users who are members of the group.	For most directories, this attribute should be `member`.
Email field	•	Email attribute.	For most directories, this attribute should be `mail`.
Group Base	• If using LDAP groups	Base DN containing all of your groups.	`OU=Groups,DC=YOURCOMPANY,DC=COM`
Group Id	• If using LDAP groups	Unique identifier for your groups.	`cn` For Active Directory, this could also be the `sAMAccountName`.

Once you complete corporate LDAP configuration, all users who are part of that configured LDAP have access to AppCenter.

6 Uninstall AppCenter

1. Uninstall application services:

```
$ appctl platform uninstall --name appcenter
```

2. Uninstall platform services:

```
$ appctl platform uninstall --name platform
```

3. Uninstall init:

```
$ appctl platform uninstall --name init
```

4. Delete all PersistentVolumeClaims (PVCs):

```
$ kubectl delete pvc -n td-platform --all
```

5. Clean up namespaces, deployments, services, and secrets:

```
$ kubectl delete ns td-platform
$ kubectl delete deployment tiller-deploy -n kube-system
$ kubectl delete svc tiller-deploy -n kube-system
$ kubectl delete secrets tiller-secret -n kube-system
```

6. Make sure that there are no pods running in `appcenter` and the `td-platform` namespace.

```
$ kubectl get pods -n appcenter
$ kubectl get pods -n td-platform
```

Note: If you want to reinstall AppCenter, you need to uninstall the old one first.

7 Back Up AppCenter

Postgres metadata and Vault are backed up as part of the backup process.

1. From the client machine run:

```
$ appctl platform backup --chart appcenter-backup-1.10.0+build.21.149baa2.tgz --name
backup1 --namespace td-platform --set global.rook.rookEnabled=false
```

Note: To download the backup use the `--download` flag with the command. The backup will be downloaded to the client machine in the `/tmp` location.

Note: If you want to download it to a different location use `--downloadPath <path>` with the command.

8 Restore AppCenter

If you want to restore AppCenter 1.10 using a backup from a 1.10 cluster, you must install AppCenter services (`appcenter namespace`) before restoring AppCenter 1.10.

If you want to restore AppCenter versions earlier than 1.10, you must uninstall AppCenter services (`appcenter namespace`) before restoring AppCenter.

Postgres metadata and the vault are restored as part of the backup.

1. From the client machine run:

```
$ appctl platform restore --chart appcenter-restore-1.10.0+build.21.149baa2.tgz --
localPath <location-of-backup-file> --namespace td-platform  --set
global.rook.rookEnabled=false
```

Note: To restore from a file that is currently present within AppCenter, remove `localPath` and provide a name with `--name parameter`. The `--decKey VALUE` needs to be used if the password for encrypting the backup is different for the source cluster. You can find this key in `application-services-override.yaml`. This key is the same as the key used for encrypting the backup.

9 Upgrading AppCenter (1.8.7 and Later)

AppCenter 1.10.0 supports over-the-top upgrades

1. Download AppCenter 1.10.0 bits. See *Downloads* on page 4.
2. Prepare the client machine. See *Preparing the Client* on page 12.
3. Prepare the host. See *Preparing the Host* on page 12.
4. Install AppCenter for over-the-top upgrade. See *Installing AppCenter* on page 13.

   Note: While installing AppCenter 1.10.0, use the passwords and keys that were used in the override files during previous AppCenter installations.

10 Troubleshooting

In case of errors during installation, please run the following commands to capture important information for troubleshooting:

Command Example	Description
`appctl version`	Fetch the appctl version.
`appctl platform install --namespace foo --name bar mychart.tgz --debug`	Capture output of the CLI.
`kubectl get pods -n td-platform | grep CrashLoopBackOff`	List all pods in AppCenter Platform Services for any crashes.
`kubectl get pods -n appcenter | grep CrashLoopBackOff`	List all pods in AppCenter Application Services for crash loop.
`kubectl describe pod <pod_name> -n td-platform`	List details of specific errors.
`kubectl describe pod <pod_name> -n appcenter`	List details of specific errors in AppCenter.
`kubectl logs -f <pod_name> -n td-platform`	Get logs for platform components.
`kubectl logs -f <pod_name> -n appcenter`	Get logs for AppCenter components.

10.1 Rotating Certificates (tls.key, tls.crt)

From the client machine, do the following:

1. Update the domain (same as the one in certificates):

```
$ export APPCTL_DOMAIN=appcenter.example.com
```

2. Rename the certificate and key to tls.crt and tls.key:
3. Verify the certificates. Check for the DNS name, it should match the APPCTL_DOMAIN:

```
$ openssl x509 -in tls.crt -text -noout
```

4. Copy the certificates to $APPCTL_HOME (default - ~/.appctl/ on the client machine):
5. Upgrade platform services:

```
$ appctl platform install --namespace td-platform --name platform platform-
1.10.0+build.21.149baa2.tgz -f platform-override.yaml --reset-values
```

6. Upgrade application services:

```
$ appctl platform install --namespace appcenter --name appcenter appcenter-
1.10.0+build.21.149baa2.tgz -f appcenter-override.yaml --reset-values
```

7. Restart ambassador:

```
$ kubectl delete pod -n td-platform -l component=ambassador
```