ORACLE

Deploying Oracle True Cache on Kubernetes with an Oracle RAC Database Primary

Maximum Availability Architecture team

November, 2026, Version [1.0]
Copyright © 2026, Oracle and/or its affiliates
Public

Purpose statement

This document shows how Oracle True Cache can become part of a Kubernetes-based application architecture. Oracle Database Operator for Kubernetes simplifies this deployment model by managing much of the lifecycle work that would otherwise be manual. Using Kubernetes-native resources, teams can define, deploy, configure, and operate Oracle True Cache in a declarative way, aligned with the same workflows they already use for other workloads in the cluster.

The deployment uses a production-style architecture: an Oracle Real Application Clusters (Oracle RAC) database runs outside the Kubernetes cluster as the primary database, while the application tier runs inside Kubernetes. This reflects a common pattern where the database remains on a dedicated platform, and the application layer runs in a Kubernetes environment.

This document implements the architecture on Oracle Kubernetes Engine (OKE) in Oracle Cloud Infrastructure (OCI). Oracle Database Operator for Kubernetes can use OCI service integrations to automate deployment tasks, such as creating a Network Load Balancer and configuring Oracle Cloud Infrastructure DNS.

Copyright © 2026, Oracle and/or its affiliates / Public

ORACLE

Contents

Introduction **4**

What this document covers **5**

Summary of components and their requirements **7**

Deployment **8**

1. Create OKE Cluster on OCI 8
2. Configure network rules 8
3. Prepare the primary Oracle RAC database 8
4. Prepare the prerequisites: registry access, secrets, and configuration files 9
5. Deploy Oracle Database Operator for Kubernetes 9
6. Prepare DNS 10
7. Use Oracle Database Operator for Kubernetes to deploy Oracle True Cache 15

Verifications **22**

Connect from the application **28**

References **29**

List of figures

Figure 1 Oracle Kubernetes Engine (OKE) cluster, Oracle True Cache running in the same Kubernetes cluster as the application 6

Figure 3 Private DNS zone for the True Cache external hostname 10

Figure 4 OCI Private DNS View including the private zone for the True Cache hostname 11

Figure 5 Private View added to the VCN's resolver 11

Figure 6 DNS record resolving to ClusterIP address 28

List of tables

Table 1. List of the components and requirements 7

Table 2 Required communications 8

Copyright © 2026, Oracle and/or its affiliates / Public

ORACLE

Introduction

Oracle True Cache is an in-memory, read-only cache for Oracle AI Database that helps applications reduce query latency and offload read traffic from the primary database. It keeps frequently accessed data close to the application while remaining transparently synchronized with the source database, making it especially useful for workloads with high read demand, low-latency requirements, or geographically distributed application tiers. Key benefits of Oracle True Cache include:

- Automatically refreshed and database-consistent cached data
- SQL access across relational, JSON, and other Oracle data types
- Reduced need for custom cache loading and invalidation logic
- Read offload and horizontal scalability using multiple cache instances
- Oracle Database security and enterprise manageability
- Deploying Oracle True Cache closer to applications can improve response times.
- Support for larger cache volumes through flash cache and partitioned configurations

Oracle True Cache is bundled natively with the Oracle AI Database Enterprise Edition license, and no additional licensing fees are required compared with other caching products. Beyond licensing costs, traditional caches require developers to manually write complex cache-loading and invalidation logic, which is not required with Oracle True Cache. It can process full SQL queries and transparently support complex data types, including JSON and vector search for AI, which can significantly reduce implementation costs.

Deploying Oracle True Cache in a Kubernetes cluster places the cache closer to the applications running there. Instead of routing every read request to the primary database, applications can access a nearby cache instance within the same cluster. This can improve response times, reduce load on the primary database, and make the architecture more scalable for read-intensive services.

This document shows how Oracle True Cache can become part of a Kubernetes-based application architecture. Oracle Database Operator for Kubernetes simplifies this deployment model by managing much of the lifecycle work that would otherwise be manual. Using Kubernetes-native resources, teams can define, deploy, configure, and operate Oracle True Cache in a declarative way, aligned with the same workflows they already use for other workloads in the cluster.

The deployment uses a production-style architecture: an Oracle Real Application Clusters (Oracle RAC) database runs outside the Kubernetes cluster as the primary database, while the application tier runs inside Kubernetes. This reflects a common pattern where the database remains on a dedicated platform, and the application layer runs in a Kubernetes environment.

This document implements the architecture on Oracle Kubernetes Engine (OKE) in Oracle Cloud Infrastructure (OCI). Oracle Database Operator for Kubernetes can use OCI service integrations to automate deployment tasks, such as creating a Network Load Balancer and configuring Oracle Cloud Infrastructure DNS.

Copyright © 2026, Oracle and/or its affiliates / Public

What this document covers

- ✓ **Create the OKE cluster and prepare networking.**
- ✓ **Prepare the primary Oracle RAC database.**
- ✓ **Deploy Oracle Database Operator for Kubernetes.**
- ✓ **Create required secrets, wallets, blobs, and config maps.**
- ✓ **Prepare DNS for the Oracle True Cache service and primary database reachability.**
- ✓ **Deploy Oracle True Cache and run validation checks.**

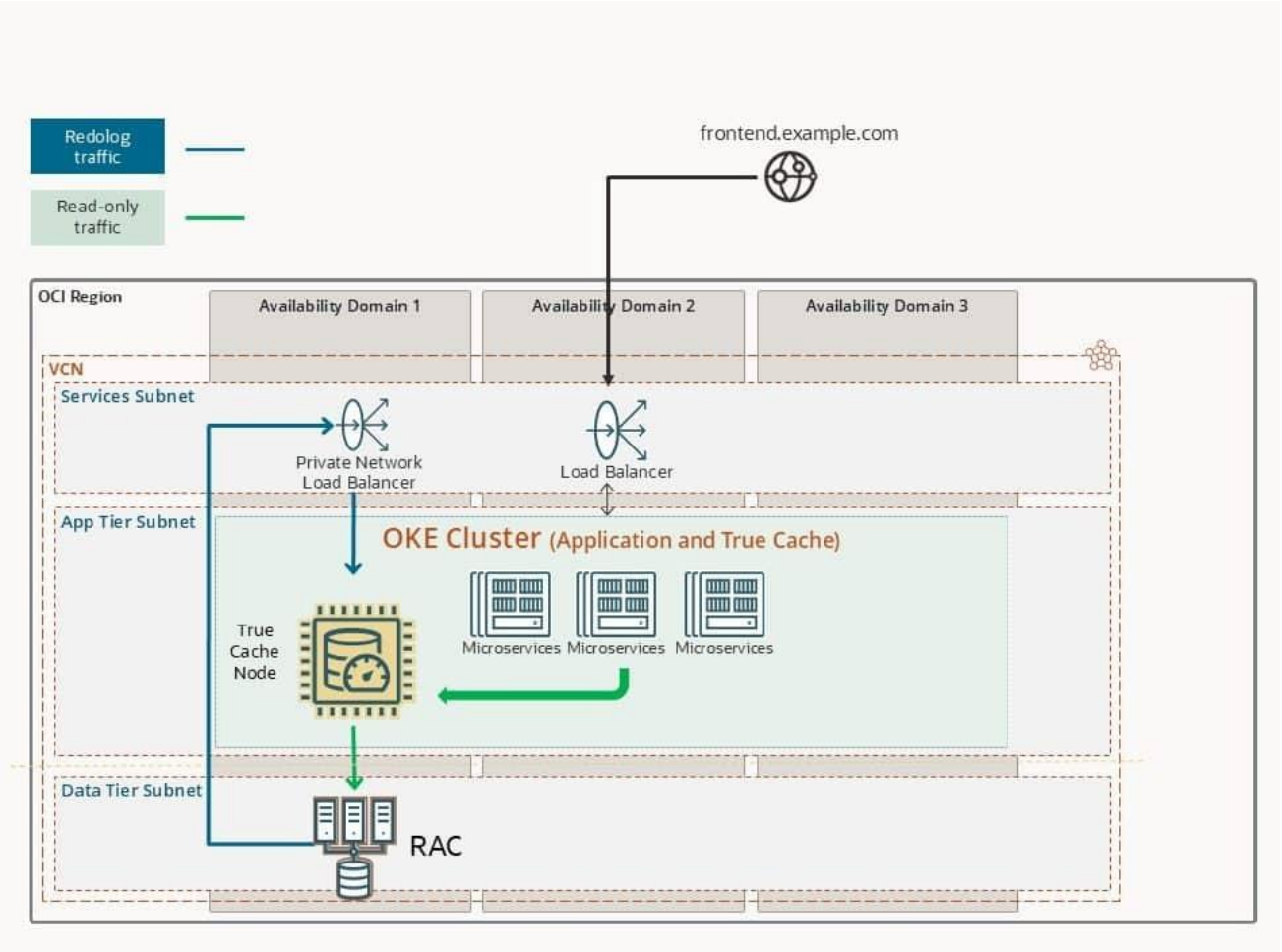
Environment description

The steps in this document apply to an environment with the following components:

- A VCN in an OCI region with separate subnets for each tier: an application subnet for OKE worker nodes, a services subnet for OKE services, and a database subnet for the database.
- An OKE cluster that runs the application.
- A Java application running on Helidon. It uses Oracle Universal Connection Pool (UCP) with JDBC to connect to the database. It defines two separate data source pools: a general data source, which can be used for any type of operation, and a read-only data source, which is intended for read-only operations. Both data sources use UCP, but they have different names and are separate entities, so they can use different connection strings and pool sizing.
- Oracle Base Database Service with Oracle RAC for Oracle AI Database 26aiA primary Oracle AI Database 26ai system using Oracle Real Application Clusters (Oracle RAC) runs on Oracle Base Database Service in the database subnet, outside the Kubernetes cluster.
- Oracle True Cache runs in the same OKE cluster as the application. Because OKE runs on OCI, Oracle Database Operator for Kubernetes can use OCI service integrations to automate parts of the deployment. The operator creates:
    - A pod running the Oracle True Cache instance.
    - An OCI private Network Load Balancer that exposes Oracle True Cache to external components, such as the primary database. The OCI Network Load Balancer operates at Layers 3 and 4 of the OSI model. It provides high availability, high throughput, and ultra-low latency, and is designed for long-lived connections. These characteristics make it well suited for exposing database services.
      Do not use an OCI Load Balancer to expose the Oracle True Cache listener. The OCI Load Balancer is designed for application traffic, such as HTTP/HTTPS, while an OCI Network Load Balancer is better suited to the long-lived connections required by database services.
    - A ClusterIP address for access from within the Kubernetes cluster.
    - An OCI Block Volume used as persistent storage for the Oracle True Cache pod.

Copyright © 2026, Oracle and/or its affiliates / Public

Figure 1 Oracle Kubernetes Engine (OKE) cluster, Oracle True Cache running in the same Kubernetes cluster as the application



The same steps apply when the OKE cluster and primary Oracle RAC database are in different regions or virtual cloud networks (VCNs). In that case, connect the networks through a Dynamic Routing Gateway and enable DNS resolution between them.

Copyright © 2026, Oracle and/or its affiliates / Public

ORACLE

Summary of components and their requirements

The following table lists the components of the deployment, the example value used in this document, and the requirements if they apply.

Table 1. List of the components and requirements

Component	Value used in this example	Requirement
Virtual Cloud Network	*vcnfra80*	
Subnets for database, application and services	*dbsubnet* for the database *appsubnet* for OKE cluster worker nodes *svcsubnet* for load balancer	
Network security lists and rules	Security lists defined in each subnet with the required security rules explained in next steps.	
DNS Private zone and View	*tcinternal.example.com* private zone *tcinternal.example.com_PRIV_VIEW_TC_IN_K8* private view	
Database	Oracle Base Database Service with Oracle RAC for Oracle AI Database 26ai	26ai RU 23.26.4 or higher (for 23.26.1/2/3 install the one-off patch for 38769688)
Oracle Kubernetes Engine (OKE) cluster	OKE Cluster	
Kubernetes namespace for Oracle True Cache	*tcdboper*	
Oracle Database Operator for Kubernetes	Oracle Database Operator for Kubernetes v2.2.0	V2.2.0 or higher
True Cache Image	Oracle AI Database 26ai container image	23.26.1 or higher
Network Load Balancer for exposing the True Cache	OCI Private Network Load Balancer. Hostname for this Network Load Balancer: *tck8node.tcinternal.example.com*	It must operate at Layer 4 of the OSI model and support long-lived TCP connections.
Application	Helidon application using Oracle Universal Connection Pool (UCP) through JDBC	
Load Balancer for exposing the application	OCI Private Load Balancer	

In addition to the example values above, all bold values in this document are customizable**.**

Copyright © 2026, Oracle and/or its affiliates / Public

ORACLE

Deployment

1. Create OKE Cluster on OCI

The steps assume that an OKE cluster already exists, and that an application is already running in the cluster. If you do not have an OKE cluster yet, create one in your VCN by following the instructions in the Oracle documentation: https://docs.oracle.com/en-us/iaas/Content/ContEng/Tasks/create-cluster.htm

2. Configure network rules

Allow the following traffic between the primary Oracle RAC database and the OKE cluster:

Table 2 Required communications

Source	Destination	Port/Range	Purpose/Notes
Kubernetes worker node subnet (application subnet)	Primary DB subnet	1521,6200	Allow Kubernetes nodes to communicate with the primary database listener and ONS service.
Primary DB subnet	Kubernetes Services subnet	1521	Allow the primary RAC database to reach True Cache through a Network Load Balancer.
Kubernetes Services subnet	Kubernetes node subnet	30000-32767	Allow the NLB to forward traffic to Kubernetes NodePort services. Note that a rule for each NodePort may be automatically created by OKE when the load balancer is provisioned.

Since the application is in the same Oracle Kubernetes Engine (OKE) cluster as the True Cache, the application can communicate with the Oracle True Cache listener over the cluster network.

3. Prepare the primary Oracle RAC database

- Ensure that the primary database has the patch for bug 38769688, which enables a primary database licensed for EE High Performance (EEHP) or EE Extreme Performance (EEEP) to communicate with Oracle True Cache running with Oracle Database Enterprise Edition in a Kubernetes cluster. The fix is included in the October 2026 Release Update (23.26.4). One-off patches are available through My Oracle Support for earlier Oracle AI Database 26ai Release Updates (23.26.1, 23.26.2, and 23.26.3).
- Ensure that the primary database is running in ARCHIVELOG mode on the primary node.
- Applications should always connect to a RAC database through a custom service defined for the application. If this service does not already exist, create one for the primary PDB by following the steps below. If you already have a custom service configured in this way in the primary RAC, you do not need to create a new one. Example to create and start the application service “myappsvc.example.com” in a RAC:

```
srvctl add service -db $ORACLE_UNQNAME -service myappsvc.example.com -preferred PRIDB1,PRIDB2
-pdb PDB1
srvctl modify service -db $ORACLE_UNQNAME -service myappsvc.example.com -rlbgoal SERVICE_TIME
-clbgoal SHORT
srvctl start service -d $ORACLE_UNQNAME -s myappsvc.example.com
srvctl status service -d $ORACLE_UNQNAME -s myappsvc.example.com
```


Copyright © 2026, Oracle and/or its affiliates / Public

4. Prepare the prerequisites: registry access, secrets, and configuration files

- Create a Kubernetes namespace for Oracle True Cache. In this example, the namespace is tcdboper:

```
kubectl create namespace tcdboper
```

- To pull the Oracle True Cache image, create an image-pull secret for Oracle Container Registry. Example:

```
kubectl create secret docker-registry ocr-pull \
--namespace tcdboper \
--docker-server=container-registry.oracle.com \
--docker-username=username@example.com \
--docker-password='XXXXXXXXXXXXXXXXXX' \
--docker-email=username@example.com
```

- Create Kubernetes secrets for the database administrator password and TDE wallet password, using the values configured for the primary database. Example:

```
kubectl create secret generic db-admin-secret \
--from-literal=oracle_pwd="XXXXXXXXX" -n tcdboper

kubectl create secret generic tde-wallet-secret \
--from-literal=tde_wallet_pwd="XXXXXXXXX" -n tcdboper
```

- Oracle True Cache provisioning requires a blob file containing the primary database's TDE wallet and password file. Run this command on the primary Oracle RAC database to create the file. Example:

```
$ORACLE_HOME/bin/dbca -configureDatabase
-prepareTrueCacheConfigFile \
-sourceDB $ORACLE_UNQNAME \
-trueCacheBlobLocation /tmp/primary_db_config_blob/ \
-silent
```

- Copy the generated file (for example DBPRIM_TIMESTAMP.tar.gz) to the host that runs the kubectl commands and create a ConfigMap in the Kubernetes cluster. Example:

```
kubectl create configmap orcl-primary-truecache-blob \
--from-file=DBPRIM_TIMESTAMP.tar.gz=./DBPRIM_TIMESTAMP.tar.gz \
-n tcdboper
```

5. Deploy Oracle Database Operator for Kubernetes

Deploy Oracle Database Operator for Kubernetes version 2.2.0 or later:

- Ensure that cert-manager is installed in the cluster. On OKE, you can install it as an OKE add-on.
- Download the operator from https://github.com/oracle/oracle-database-operator .
- Create the required RBAC resources and install Oracle Database Operator for Kubernetes. This example uses a namespace-scoped deployment. The following example commands generate the RBAC manifest that configures the operator to watch the *tcdboper* namespace, then install the operator:

```
scripts/generate-namespace-install.sh oracle-database-operator-system,tcdboper
kubectl apply -f dist/install/oracle-database-operator-rbac.yaml
kubectl apply -f dist/install/oracle-database-operator-system.yaml
```

- Confirm that the Oracle Database Operator for Kubernetes pods and services are running in the expected namespace. Example:

Copyright © 2026, Oracle and/or its affiliates / Public

```
kubectl get all -n oracle-database-operator-system
```

For additional information and installation options, visit https://github.com/oracle/oracle-database-operator.

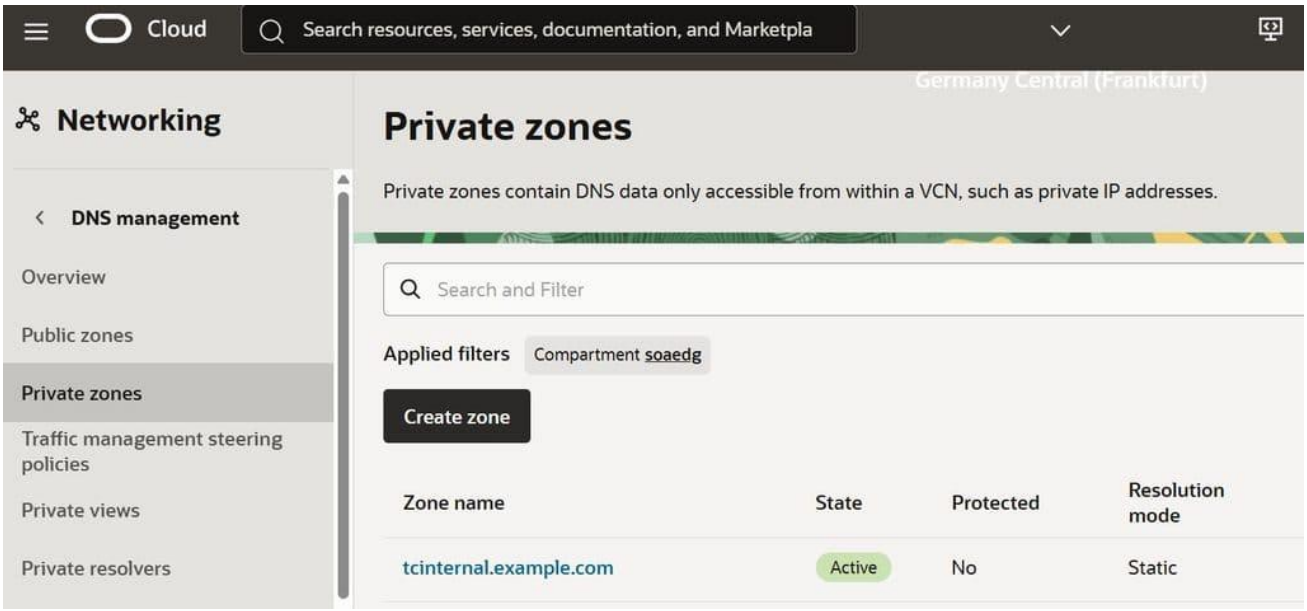
6. Prepare DNS

When Oracle True Cache is provisioned in an OKE cluster, it registers with the primary database using a configurable hostname. The primary database must be able to resolve that hostname and connect back to the True Cache instance. To enable this, Oracle Database Operator for Kubernetes exposes Oracle True Cache through a Network Load Balancer. The operator's ExternalDNS integration automatically registers the True Cache hostname in Oracle Cloud Infrastructure DNS and maps it to the load balancer's IP address.

Follow these steps to connect the OKE cluster to Oracle Cloud Infrastructure DNS so it can create records in a private DNS zone when the Oracle True Cache Network Load Balancer is provisioned. In this example, the Oracle True Cache hostname is *tck8node1.tcinternal.example.com*, and the private DNS zone is *tcinternal.example.com*.

- If it does not already exist, create a private DNS zone for the Oracle True Cache external hostname in the OCI region, for example, tcinternal.example.com. You do not need to add records manually.

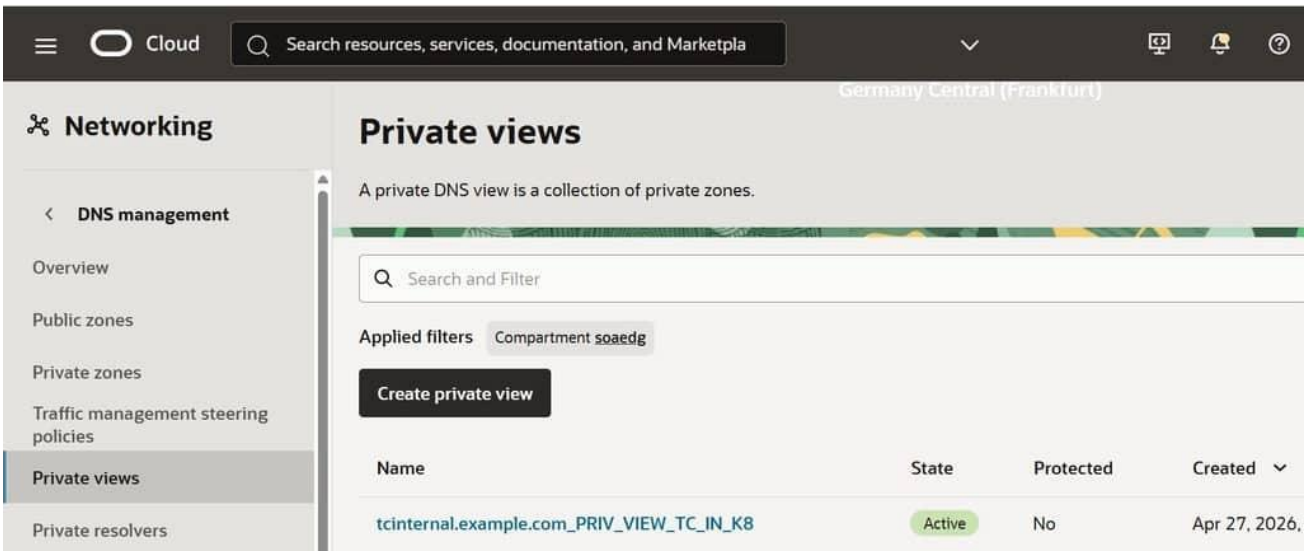
Figure 2 Private DNS zone for the True Cache external hostname



- Create a private DNS view that includes this zone.

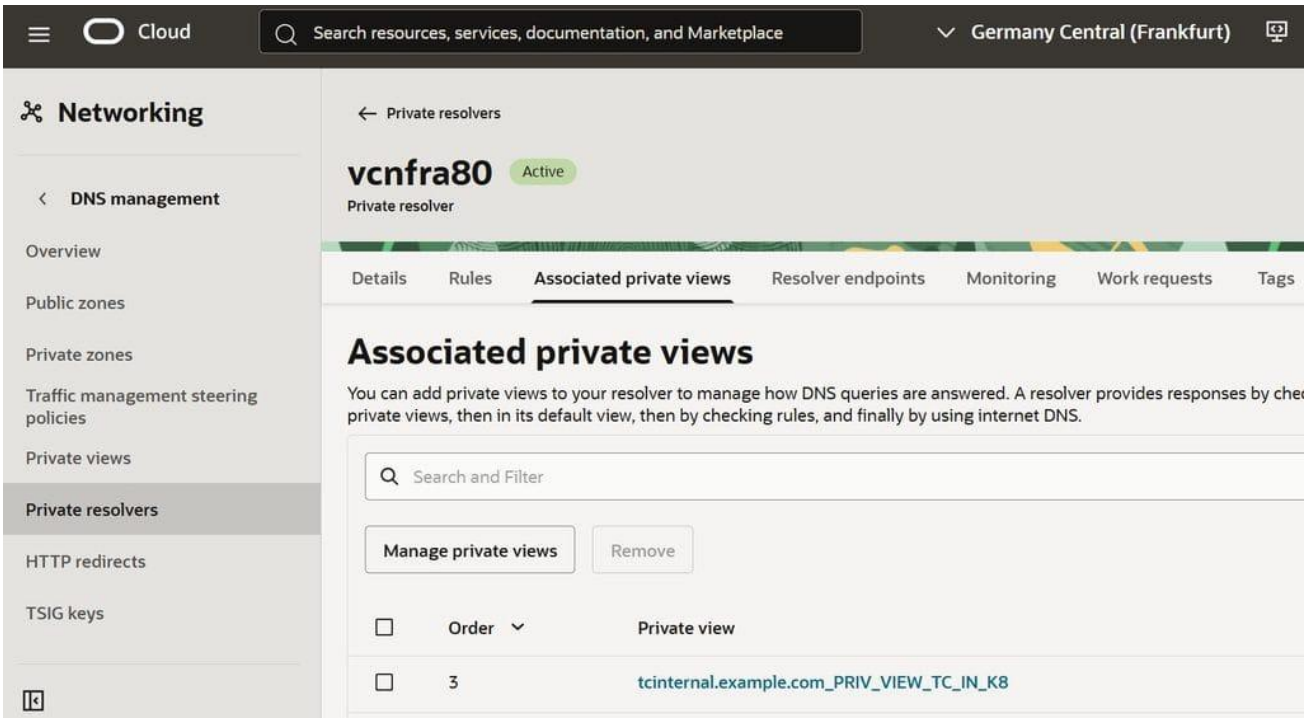
Copyright © 2026, Oracle and/or its affiliates / Public

Figure 3 OCI Private DNS View including the private zone for the True Cache hostname



- To make the zone resolvable by the primary database, add the private view to the resolver for the database VCN.

Figure 4 Private View added to the VCN's resolver



> ✓ NOTE: The following sample yamls are available in https://github.com/oracle/oracle-database-operator/tree/main/docs/sidb/external-dns

- Create a dedicated namespace and service account for ExternalDNS. Example:

```
kubectl apply -f 01-namespace-serviceaccount.yaml
```

Where *01-namespace-serviceaccount.yaml*

Copyright © 2026, Oracle and/or its affiliates / Public

```
apiVersion: v1
kind: Namespace
metadata:
  name: external-dns
---
apiVersion: v1
kind: ServiceAccount
metadata:
  name: external-dns
  namespace: external-dns
```

- Grant permissions to the ExternalDNS service account. Example:

```
kubectl apply -f 02-rbac.yaml
```

Where *02-rbac.yaml* contains:

Copyright © 2026, Oracle and/or its affiliates / Public

```
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRole
metadata:
  name: external-dns
rules:
  - apiGroups: [""]
    resources:
      - services
      - endpoints
      - pods
      - nodes
    verbs:
      - get
      - list
      - watch
  - apiGroups: ["discovery.k8s.io"]
    resources:
      - endpointslices
    verbs:
      - get
      - list
      - watch
  - apiGroups: ["networking.k8s.io"]
    resources:
      - ingresses
    verbs:
      - get
      - list
      - watch
  - apiGroups: [""]
    resources:
      - events
    verbs:
      - create
      - patch
---
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRoleBinding
metadata:
  name: external-dns
roleRef:
  apiGroup: rbac.authorization.k8s.io
  kind: ClusterRole
  name: external-dns
subjects:
  - kind: ServiceAccount
    name: external-dns
    namespace: external-dns
```

- In the OCI Console, create a narrowly scoped IAM policy. Example (enter it on a single line):

Copyright © 2026, Oracle and/or its affiliates / Public

```
Allow any-user to manage dns in compartment <compartment_name> where all {
request.principal.type='workload', request.principal.namespace='external-dns',
request.principal.service_account='external-dns', request.principal.cluster_id='<oke-
cluster-ocid>'}
```

- Create an OCI configuration secret for ExternalDNS, applying the *03-external-dns-secret-workload-identity.yaml* file. Replace the region and compartment OCID with your values. Example:

```
kubectl apply -f 03-external-dns-secret-workload-identity.yaml
```

Where *03-external-dns-secret-workload-identity.yaml* contains:

```
apiVersion: v1
kind: Secret
metadata:
name: external-dns-config
namespace: external-dns
type: Opaque
stringData:
  oci.yaml: |
   auth:
     region: eu-frankfurt-1
     useWorkloadIdentity: true
   compartment: ocid1.compartment.oc1..XXXXXXXXXXXXXX
```

- Deploy ExternalDNS in the OKE cluster. In the YAML file shown below, set txt-owner-id to a unique value. You can also use the optional *–domain-filter* and *–zone-id-filter* arguments to restrict changes to a specific domain or zone. Example:

```
kubectl apply -f 04-external-dns-deployment-workload-identity.yaml
```

Where *04-external-dns-deployment-workload-identity.yaml*

Copyright © 2026, Oracle and/or its affiliates / Public

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: external-dns
  namespace: external-dns
spec:
  strategy:
    type: Recreate
  selector:
    matchLabels:
      app: external-dns
  template:
    metadata:
      labels:
        app: external-dns
    spec:
      serviceAccountName: external-dns
      containers:
        - name: external-dns
          image: registry.k8s.io/external-dns/external-dns:v0.21.0
          args:
            - --source=service
            - --source=ingress
            - --provider=oci
            - --policy=upsert-only
            - --txt-owner-id=TrueCacheTest
            - --oci-zone-scope=PRIVATE
            - --domain-filter=tcinternal.example.com (OPTIONAL)
            - --zone-id-filter=ocid1.dns-zone.oc1.xxxxxxxxxxxx (OPTIONAL)
            - --publish-internal-services
            - --oci-config-file=/etc/kubernetes/oci.yaml
          volumeMounts:
            - name: config
              mountPath: /etc/kubernetes/
      volumes:
        - name: config
          secret:
            secretName: external-dns-config
```

To use the ExternalDNS integration, add the hostname annotation "*external-dns.alpha.kubernetes.io/hostname*" to the Load Balancer service in the Oracle True Cache deployment YAML, as shown later. The ExternalDNS feature will then add the hostname to the OCI private DNS zone.

7. Use Oracle Database Operator for Kubernetes to deploy Oracle True Cache

Use Oracle Database Operator to deploy True Cache in the Kubernetes cluster.

- Copy the *configure-primary-truecache-service.sh* script to the primary Oracle RAC database nodes. The script is required only when *autoTCServiceRegistration* is set to true in the provisioning YAML. The default value is false. When enabled, Oracle Database Operator for Kubernetes associates the primary database service with an Oracle True Cache service. This association is required when the application

Copyright © 2026, Oracle and/or its affiliates / Public

uses the single-logical-connection method. It is not needed with other approaches, such as the two-physical-connections method. For more details, see Methods for Connecting to Oracle True Cache. In this example, *autoTCServiceRegistration* is set to true for demonstration purposes, hence:

  - Download the script from GitHub: https://github.com/oracle/docker-images/tree/main/OracleDatabase/SingleInstance/extensions/truecache
  - Place it in /home/oracle on the Oracle RAC database nodes. If you place it in another folder, set the PRIMARY_TC_SERVICE_SCRIPT_PATH variable in the YAML manifest to the script's full path.
  - Make oracle:oinstall the owner and give it execution permissions.

- Create a yaml descriptor for the True Cache and deploy Oracle True Cache with Oracle Database Operator for Kubernetes. Example:

```
kubectl apply -f true-cache-node.yaml
```

The *true-cache-node.yaml* for this example:

Copyright © 2026, Oracle and/or its affiliates / Public

```
# Copyright (c) 2026, Oracle and/or its affiliates.
# Licensed under the Universal Permissive License v 1.0 as shown at
http://oss.oracle.com/licenses/upl.
#

apiVersion: database.oracle.com/v4
kind: SingleInstanceDatabase
metadata:
name: tck8node1
namespace: tcdboper
spec:
## The SID for the True Cache. Use only alphanumeric characters.
  sid: TCK8DB1
  ## DB edition
  edition: enterprise
  ## DB Type
  createAs: truecache
  ## Use a primary endpoint reachable from this cluster.
  primarySource:
    connectString: "drdbrac15a-
scan.dbsubnet.vcnfra80.oraclevcn.com:1521/DB15_FRA.dbsubnet.vcnfra80.oraclevcn.com"
    dbName: "DB15"
  ## Generated blob ConfigMap from the primary database.
  trueCache:
    blobConfigMapRef: orcl-primary-truecache-blob
    blobConfigMapKey: DBPRIM_TIMESTAMP.tar.gz
    blobMountPath: /stage/DBPRIM_TIMESTAMP.tar.gz
    truedbUniqueName: "TCK8DB1_FRA"
    ## Service mapping metadata.
    ## If set to true, keep the sample helper script installed
    ## on the same path on every RAC primary node.
    autoTCServiceRegistration: true
    ## PRIMARY_PDB_NAME:PRIMARY_PDB_SERVICE_NAME:TRUECACHE_PDB_SERVICE_NAME
    trueCacheServices:
      - "PDB1:myappsvc.example.com:tcokenodes.example.com"

  # Optionally, you can provide your custom node selector rules
  nodeSelector:
    role: TC
  ## Grouped security settings.
  security:
    secrets:
      admin:
        secretName: db-admin-secret
        secretKey: oracle_pwd
        keepSecret: true
      tde:
        secretName: tde-wallet-secret
        secretKey: tde_wallet_pwd
  ## Database image details
  image:
    pullFrom: container-registry.oracle.com/database/enterprise_ru:23.26.3.0
```


Copyright © 2026, Oracle and/or its affiliates / Public

```
    prebuiltDB: false
    imagePullPolicy: IfNotPresent
    pullSecrets: ocr-pull
  initParams:
    sgaTarget: 127000
    pgaAggregateTarget: 32000
    processes: 4500
  shmSize: 130Gi
  resources:
    requests:
      memory: "256Gi"
      cpu: "32"
    limits:
      memory: "256Gi"
      cpu: "32"
  ## Storage details
  persistence:
    oradata:
      size: 60Gi
      storageClass: "oci-bv"
      accessMode: "ReadWriteOnce"

  ## Optional operator-managed external service for the truecache database.
  ## This renders a Service named <sidb-name>-lb
  services:
    endpoints:
      - name: loadbalancer
        type: LoadBalancer
        ## Keep Service across delete/recreate so NLB / assigned IP can be reused.
        isKeep: true
        externalTrafficPolicy: Local
        annotations:
          oci.oraclecloud.com/load-balancer-type: "nlb"
          oci-network-load-balancer.oraclecloud.com/internal: "true"
          oci-network-load-balancer.oraclecloud.com/assigned-private-ipv4: 10.4.81.245
          oci-network-load-balancer.oraclecloud.com/subnet: "ocid1.subnet.oc1.eu-
xxxxljna"
          external-dns.alpha.kubernetes.io/hostname: "tck8node1.tcinternal.example.com"
          oci-network-load-balancer.oraclecloud.com/is-preserve-source: "false"
        tcp:
          enabled: true
          port: 1521
  ## Count of Database Pods. Should be 1 for True Cache.
  replicas: 1
```

- Check the resources created. Example:

Copyright © 2026, Oracle and/or its affiliates / Public

```
kubectl get all -n tcdboper

NAME                   READY   STATUS    RESTARTS   AGE
pod/tck8node1-71q0u    0/1     Running   0          70s
NAME                  TYPE        CLUSTER-IP    EXTERNAL-IP  PORT(S)              AGE
service/tck8node1     ClusterIP    10.96.225.107 <none>      1521/TCP             20s
service/tck8node1-lb LoadBalancer 10.96.242.4   10.4.81.245 5500:30470/TCP,1521:31815/TCP
139m
```

- Check the output of the pod. Example:

```
kubectl logs -f pod/tck8node1-71q0u -n tcdboper
```


Copyright © 2026, Oracle and/or its affiliates / Public

```
ORACLE EDITION: ENTERPRISE
07-30-2026 11:23:39 UTC : createDB.truecache-wrapper.sh : Using caller-provided
PRIMARY_DB_NAME=DB15.
07-30-2026 11:23:39 UTC : createDB.truecache-wrapper.sh : Routing INIT_PROCESSES through the True
Cache DBCA shim.
07-30-2026 11:23:39 UTC : createDB.truecache-wrapper.sh : Starting listener LISTENER before DBCA
startup to warm the external callback path.
07-30-2026 11:23:39 UTC : createDB.truecache-wrapper.sh : Waiting for external callback target
//tck8node1.tcinternal.example.com:1521/TCK8DB1_FRA.dbsubnet.vcnfra80.oraclevcn.com to become
connectable.
07-30-2026 11:23:44 UTC : createDB.truecache-wrapper.sh : External callback target
//tck8node1.tcinternal.example.com:1521/TCK8DB1_FRA.dbsubnet.vcnfra80.oraclevcn.com is connectable.
07-30-2026 11:23:44 UTC : createDB.truecache-wrapper.sh : Stopping temporary listener LISTENER before
DBCA so DBCA can bind the callback listener on 1521.
07-30-2026 11:23:45 UTC : createDB.truecache-wrapper.sh : Listener port 1521 is free for DBCA.
07-30-2026 11:23:45 UTC : createDB.truecache-wrapper.sh : No True Cache DB credentials wallet was
found. DBCA will fall back to ORACLE_PWD if it is set.
07-30-2026 11:23:45 UTC : createDB.truecache-wrapper.sh : Installed True Cache DBCA shim at
/tmp/truecache-dbca.K8xFeA/dbca.
Check and wait for the existence of /stage/DB15_2026-04-27_07-38-34AM_67403.tar.gz...
/stage/DB15_2026-04-27_07-38-34AM_67403.tar.gz found!
07-30-2026 11:23:45 UTC : createDB.truecache-wrapper.sh : Passing source TDE wallet password to DBCA
with -sourceTdeWalletPassword.
07-30-2026 11:23:45 UTC : createDB.truecache-wrapper.sh : Removed legacy TDE wallet password stdin
line because -sourceTdeWalletPassword is enabled.
Enter Remote DB SYS user
password:                                                                    **********

[WARNING] [DBT-11217] Unable to check available shared memory on specified node(s) ([tck8node1]).
Session ID of the current execution is: 1
-----------------
Running Extract_password_file_from_blob_file job
Completed Extract_password_file_from_blob_file job
25% complete
-----------------
Running Create_static_listener job
Completed Create_static_listener job
38% complete
-----------------
Running Register_listener job
Completed Register_listener job
50% complete
-----------------
Running Extract_tde_wallet_from_blob_file job
Completed Extract_tde_wallet_from_blob_file job
54% complete
-----------------
Running Setup_required_directories job
Skipping. Job is detected as not applicable.
57% complete
-----------------
Running Create_pfile job
```


Copyright © 2026, Oracle and/or its affiliates / Public

```
Completed Create_pfile job
61% complete
-----------------
Running Start_nomount_instance job
Completed Start_nomount_instance job
64% complete
-----------------
Running Create_TDE_wallet job
Completed Create_TDE_wallet job
68% complete
-----------------
Running Create_truecache_instance job
Completed Create_truecache_instance job
71% complete
-----------------
Running Add_oratab_entry job
Completed Add_oratab_entry job
75% complete
-----------------
Running Reopen_wallet job
Completed Reopen_wallet job
100% complete
Look at the log file "/opt/oracle/cfgtoollogs/dbca/TCK8DB1_FRA/TCK8DB1_FRA.log" for further details.
The Oracle base remains unchanged with value /opt/oracle
Executing user defined scripts
/opt/oracle/scripts/base/runUserScripts.sh: running
/opt/oracle/scripts/extensions/setup/registerService.sh
07-30-2026 11:26:09 UTC :  : PRIMARY_DB_NAME requested as DB15
07-30-2026 11:26:09 UTC :  : Resolved PRIMARY_DB_NAME=DB15 db_unique_name=DB15_FRA
cluster_database=true
07-30-2026 11:26:09 UTC :  : PDB_TC_SVCS_STR=PDB1:myappsvc.example.com:tcokenodes.example.com
07-30-2026 11:26:09 UTC :  : # of PDB_TC_VALUES=1
07-30-2026 11:26:09 UTC :  : setting connect str for pdb PDB1
07-30-2026 11:26:09 UTC :  : Using primary CDB connect target drdbrac15a-
scan.dbsubnet.vcnfra80.oraclevcn.com:1521/DB15_FRA.dbsubnet.vcnfra80.oraclevcn.com.
07-30-2026 11:26:09 UTC :  : Using primary PDB connect target drdbrac15a-
scan.dbsubnet.vcnfra80.oraclevcn.com:1521/myappsvc.example.com for PDB PDB1.
07-30-2026 11:26:09 UTC :  : validate if the PDB1 pdb exists
OPEN_MODE output=[READ WRITE]
07-30-2026 11:26:10 UTC :  : Resolved True Cache listener service from
service_names=TCK8DB1_FRA.tcdboper.svc.cluster.local.
07-30-2026 11:26:10 UTC :  : True Cache listener target
localhost:1521/TCK8DB1_FRA.tcdboper.svc.cluster.local is connectable.
07-30-2026 11:26:10 UTC :  : running primary-host registration helper for
PRIMARY_SERVICE=myappsvc.example.com TRUECACHE_SERVICE=tcokenodes.example.com
07-30-2026 11:26:11 UTC :  : Using cached True Cache listener service
TCK8DB1_FRA.tcdboper.svc.cluster.local.
07-30-2026 11:26:11 UTC :  : Executing primary-host True Cache registration script
/home/oracle/configure-primary-truecache-service.sh via DBMS_SCHEDULER on the primary database
machine
07-30-2026 11:26:11 UTC :  : Executing sql script using connect string
07-30-2026 11:26:12 UTC :  : Received Output:
```


Copyright © 2026, Oracle and/or its affiliates / Public

```
PL/SQL procedure successfully completed.
07-30-2026 11:26:13 UTC :  : Primary transport to True Cache TCK8DB1_FRA is active despite missing
association metadata. transport=[DEST_ID=3 STATUS=VALID TARGET=STANDBY
DESTINATION=TCK8NODE1.TCINTERNAL.EXAMPLE.C OM:1521/TCK8DB1_FRA.TCDBOPER.SVC.CLUSTER.LOCAL]
07-30-2026 11:26:13 UTC :  : Waiting for True Cache PDB PDB1 to report READ ONLY. current=[MOUNTED]
elapsed=0s
07-30-2026 11:26:28 UTC :  : Waiting for True Cache PDB PDB1 to report READ ONLY. current=[MOUNTED]
elapsed=15s
07-30-2026 11:26:33 UTC :  : True Cache PDB PDB1 is open READ ONLY.
07-30-2026 11:26:33 UTC :  : True Cache service TCOKENODES.EXAMPLE.COM already exists in PDB PDB1.
07-30-2026 11:26:34 UTC :  : True Cache start service output:
07-30-2026 11:26:34 UTC :  : True Cache service TCOKENODES.EXAMPLE.COM is active locally.
07-30-2026 11:26:35 UTC :  : True Cache Service TCOKENODES.EXAMPLE.COM is associated with primary
service MYAPPSVC.EXAMPLE.COM and active on True Cache. final_association=[TCOKENODES.EXAMPLE.COM]
final_active=[TCOKENODES.EXAMPLE.COM]
DONE: Executing user defined scripts
The Oracle base remains unchanged with value /opt/oracle
#########################
DATABASE IS READY TO USE!
#########################
```

- Check the deployment status through Oracle Database Operator for Kubernetes; it should be Healthy. Example:

```
kubectl get SingleInstanceDatabase -n tcdboper
NAME        EDITION      STATUS    ROLE         VERSION       CONNECT
STR                TCPS CONNECT STR   OEM EXPRESS URL
tck8node1   Enterprise   Healthy   TRUE_CACHE   23.26.1.0.0   10.4.81.245:1521/TCK8DB1   N
ot enabled        Unavailable
```

- In case you need to remove the True Cache, this is the command to remove the Oracle True Cache deployment and its related resources. Example:

```
kubectl delete SingleInstanceDatabase tck8node1 -n tcdboper
```

Verifications

- In the primary database:
    - Verify that the True Cache appears as one log archive destination. Example:

Copyright © 2026, Oracle and/or its affiliates / Public

```
sqlplus /as sysdba
SQL> show parameter log_archive_dest
NAME                                 TYPE        VALUE
------------------------------------ ----------- ------------------------------
log_archive_dest                     string
log_archive_dest_1                   string
log_archive_dest_10                  string
(…)
log_archive_dest_3                   string      service=, "tck8node1.tcinterna
                                                 l.example.com:1521/TCK8DB1_FRA
                                                 .tcdboper.svc.cluster.local",
                                                 LGWR ASYNC NET_TIMEOUT=30 VALI
                                                 D_FOR=(ONLINE_LOGFILE,PRIMARY_
                                                 ROLE) DB_UNIQUE_NAME="TCK8DB1_
                                                 FRA" REOPEN=15 MAX_FAILURE=20
                                                 ROLE="TRUE_CACHE"
log_archive_dest_30                  string
(…)
```

- (if *autoTCServiceRegistration* was set to true), check whether the active services show an association between the primary service and the True Cache service. This association is created during provisioning when *autoTCServiceRegistration*: true. It is required only when the application uses the "One logical connection" method. Example:

```
sqlplus /as sysdba
SQL> SET LINESIZE 32767;
SQL> SELECT service_id, name, true_cache_service FROM v$active_services;

SERVICE_ID NAME                  TRUE_CACHE_SERVICE
---------------             --------------------------------------------------
4 export_soaedg.example.com
3 DB15_CFG
6 DB15_FRA.dbsubnet.vcnfra80.oraclevcn.com
7 myappsvc.example.com          TCOKENODES.EXAMPLE.COM
1 SYS$BACKGROUND
2 SYS$USERS
(...)
12 rows selected.
```

- On the Oracle True Cache instance
  - Check the role and open mode of Oracle True Cache. The role must be "TRUE CACHE" and the open mode "READ ONLY WITH APPLY", which means that the True Cache instance is open for read-only queries and redo from the primary database is actively being applied. Example:

```
kubectl exec -i tck8node1-71q0u -n tcdboper  -- sqlplus / as sysdba

SQL> SELECT database_role, open_mode FROM v$database;
DATABASE_ROLE    OPEN_MODE
---------------- --------------------
TRUE CACHE       READ ONLY WITH APPLY
```

  - Verify that the SCN increases over time. Example:

Copyright © 2026, Oracle and/or its affiliates / Public

```
kubectl exec -i tck8node1-71q0u -n tcdboper  -- sqlplus / as sysdba

SQL> SELECT current_scn FROM v$database;
CURRENT_SCN
-----------
 1811247240
SQL> SELECT current_scn FROM v$database;
CURRENT_SCN
-----------
 1811247244
```

- Verify that Oracle True Cache is retrieving data from the primary database by connecting to the True Cache PDB and running queries. This example uses the HR schema to query the EMPLOYEES and JOBS tables. The results should match the data in the primary database. Example:

```
kubectl exec -i tck8node1-71q0u -n tcdboper  -- sqlplus / as sysdba

SQL> alter session set container=PDB1;
Session altered.
SQL> select count(*) from HR.employees;
  COUNT(*)
----------
  11730969
SQL> SELECT EMPLOYEE_ID from HR.employees ORDER BY EMPLOYEE_ID FETCH FIRST 5 ROWS ONLY;
EMPLOYEE_ID
-----------
          5
        100
        101
        102
        103
SQL> SELECT JOB_TITLE from HR.jobs ORDER BY JOB_ID FETCH FIRST 5 ROWS ONLY;
JOB_TITLE
-----------------------------------
Public Accountant
Accounting Manager
Administration Assistant
President
Administration Vice President
```

- Query the V$TRUE_CACHE_STAT view To view Oracle True Cache statistics on the True Cache instance. For more information, see Monitoring True Cache with the V$TRUE_CACHE_STAT View. Example:

Copyright © 2026, Oracle and/or its affiliates / Public

```
kubectl exec -i tck8node1-71q0u -n tcdboper  -- sqlplus / as sysdba

SQL> SET LINESIZE 32767;
COLUMN VALUE FORMAT A20;
COLUMN TIME_COMPUTED FORMAT A20;
COLUMN DATA_TIME FORMAT A20;
SELECT * FROM v$true_cache_stat;
SQL> SQL> SQL> SQL>
MY_DGID REMOTE_DGID    DEST_ID
NAME   VALUE   UNIT    TIME_COMPUTED        DATA_TIME               CON_ID
---------- ----------- ---------- ------------------------------ -------------------- -----
3338574170 21633686 0 single block fetch latency 22739   micro seconds per block 06/29/2026 09:24:17 0
3338574170 216336869 0 multiblock fetch latency  32120 micro seconds per request 06/29/2026 09:24:17 0
3338574170 216336869 0 list of blocks fetch latency 33946 micro seconds per request 06/29/2026 09:24:17 0
3338574170 216336869 0 RAM buffer hit ratio    0.00     percentage ratio 06/29/2026 09:24:17 0
3338574170 216336869 0 Flash buffer hit ratio    0.00   percentage ratio 06/29/2026 09:24:17 0
3338574170 216336869 0 True Cache hit ratio   0.00      percentage ratio 06/29/2026 09:24:17 0
3338574170 216336869 0 prewarm progress     100.00      percentage completed 06/29/2026 09:24:17 0
       transport lag  +00 00:00:00 day(2) to second(0) interval 06/29/2026 09:24:17 06/29/2026 09:24:15 0
       apply lag  +00 00:00:00  day(2) to second(0) interval 06/29/2026 09:24:17 06/29/2026 09:24:15  0
       apply finish time  +00 00:00:00.000 day(2) to second(3) interval 06/29/2026 09:24:17 0
       estimated startup time                second  06/29/2026 09:24:17  0
```

- (if *autoTCServiceRegistration* was set to true), verify that the True Cache service provided is listed in the listener. Example:

Copyright © 2026, Oracle and/or its affiliates / Public

```
kubectl exec -i tck8node1-71q0u -n tcdboper -- lsnrctl status

LSNRCTL for Linux: Version 23.26.1.0.0 - Production on 30-JUL-2026 11:30:09
Copyright (c) 1991, 2026, Oracle.  All rights reserved.
Connecting to (ADDRESS=(PROTOCOL=TCP)(HOST=0.0.0.0)(PORT=1521))
STATUS of the LISTENER
------------------------
Alias                     LISTENER
Version                   TNSLSNR for Linux: Version 23.26.1.0.0 - Production
Start Date                30-JUL-2026 11:23:57
Uptime                    0 days 0 hr. 6 min. 11 sec
Trace Level               off
Security                  ON: Local OS Authentication
SNMP                      OFF
Listener Parameter File   /opt/oracle/product/26ai/dbhome_1/network/admin/listener.ora
Listener Log File         /opt/oracle/diag/tnslsnr/tck8node1-71q0u/listener/alert/log.xml
Listening Endpoints Summary...
  (DESCRIPTION=(ADDRESS=(PROTOCOL=tcp)(HOST=0.0.0.0)(PORT=1521)))
Services Summary...
Service "TCK8DB1_FRA.dbsubnet.vcnfra80.oraclevcn.com" has 1 instance(s).
  Instance "TCK8DB1", status UNKNOWN, has 1 handler(s) for this service...
Service "TCK8DB1_FRA.tcdboper.svc.cluster.local" has 2 instance(s).
  Instance "TCK8DB1", status UNKNOWN, has 1 handler(s) for this service...
  Instance "TCK8DB1", status READY, has 1 handler(s) for this service...
Service "TCOKENODES.EXAMPLE.COM" has 1 instance(s).
  Instance "TCK8DB1", status READY, has 1 handler(s) for this service...
Service "pdb1.tcdboper.svc.cluster.local" has 1 instance(s).
  Instance "TCK8DB1", status READY, has 1 handler(s) for this service...
The command completed successfully
```

- In both primary and True Cache,
    - Query the V$true_cache view to verify the status and confirm that the configuration is consistent. The status must be HEALTHY and the REMOTE_DG_ID in the True Cache must match the MY_DG_ID of the primary and viceversa.

      Example in primary:

```
SQL> SET LINESIZE 32767;
COLUMN STATUS FORMAT A30;
SELECT * FROM v$true_cache;
SQL> SQL>
  MY_DG_ID REMOTE_DG_ID    DEST_ID
TRUE_CACHE_NAME    PRIMARY_NAME    STATUS    REMOTE_VERSION     CURRENT_SCN    CON_ID
---------- ------------ ---------- ------------------------------ ---------------------------
--- ---------------------
 216336869   3338574170          3
TCK8DB1_FRA        DB15_FRA       HEALTHY   23.0.0.0.0          1811270210       0
```

      Example in the True Cache:

Copyright © 2026, Oracle and/or its affiliates / Public

```
kubectl exec -i tck8node1-71q0u -n tcdboper  -- sqlplus / as sysdba

SQL> SET LINESIZE 32767;
COLUMN STATUS FORMAT A30;
SELECT * FROM v$true_cache;
SQL> SQL>
MY_DG_ID REMOTE_DG_ID    DEST_ID
TRUE_CACHE_NAME    PRIMARY_NAME      STATUS       REMOTE_VERSION      CURRENT_SCN    CON_ID
---------- ------------ ---------- ------------------------------ --------------------------
--- ----------------------
3338574170    216336869          0
TCK8DB1_FRA      DB15_FRA          HEALTHY     23.0.0.0.0        1811268487        0
```

- Query V$SYSSTAT on the primary and True Cache to review the system statistics related to True Cache. These statistics can help confirm that the primary database is serving blocks to True Cache.
  Example in the primary database:

```
SQL> SET LINESIZE 32767;
COLUMN value FORMAT 9999999999999999990;
select * from V$SYSSTAT where name like 'True Cache%';
SQL> SQL>
STATISTIC# NAME                                             CLASS VALUE    STAT_ID    CON_ID
---------- ---------------------------------------------------------------- ----------
1751 True Cache: block requests to preferred primary      264     0     1218856049    0
1752 True Cache: block requests to primary                          264     0     2549190145
0
1753 True Cache: blocks fetched for prewarm                         264     0     1180400875
0
1900 True Cache: message roundtrip time data send                   256 11835403 30833870
0
1901 True Cache: message roundtrip time request send      256    0     3377926425    0
1902 True Cache: message count data send                  256  111020  4102656338    0
1903 True Cache: message count request send               256    0     4234527987    0
1904 True Cache: potentially current buffer made current   264    0     111948630     0
1905 True Cache: potentially current buffer made CR       264    0     331429815     0
1906 True Cache: blocks served from primary to True Cache  264   5132056 3795117908  0
1907 True Cache: primary current block cleanout done for True Cache 264 70123 4069949261 0
1908 True Cache: primary CR block cleanout done for True Cache    264 269 2971360615 0
1909 True Cache: proactive fetched blocks for high availability   264 0  358745601   0
13 rows selected.
```

Example in the True Cache:

```
kubectl exec -i tck8node1-71q0u -n tcdboper -- sqlplus / as sysdba
SQL> SET LINESIZE 32767;
COLUMN value FORMAT 9999999999999999990;
select * from V$SYSSTAT where name like 'True Cache%';
SQL> SQL>
STATISTIC#   NAME            CLASS                VALUE    STAT_ID    CON_ID
---------- ---------------------------------------------------------------- ----------
```


Copyright © 2026, Oracle and/or its affiliates / Public

```
1758 True Cache: block requests to preferred primary        264 515 1218856049          0
1759 True Cache: block requests to primary                  264 5019 2549190145          0
1760 True Cache: blocks fetched for prewarm                 264 16127  1180400875 0
1907 True Cache: message roundtrip time data send           256 0   30833870          0
1908 True Cache: message roundtrip time request send        256 59472606 3377926425 0
1909 True Cache: message count data send                    256 0 4102656338          0
1910 True Cache: message count request send                 256 5019 4234527987          0
1911 True Cache: potentially current buffer made current    264 0  111948630          0
1912 True Cache: potentially current buffer made CR         264 8  331429815          0
1913 True Cache: blocks served from primary to True Cache   264  0 3795117908          0
1914 True Cache: primary current block cleanout done for True Cache  264 0 4069949261 0
1915 True Cache: primary CR block cleanout done for True Cache      264 0 2971360615 0
1916 True Cache: proactive fetched blocks for high availability    264  0  358745601 0
13 rows selected.
```

Connect from the application

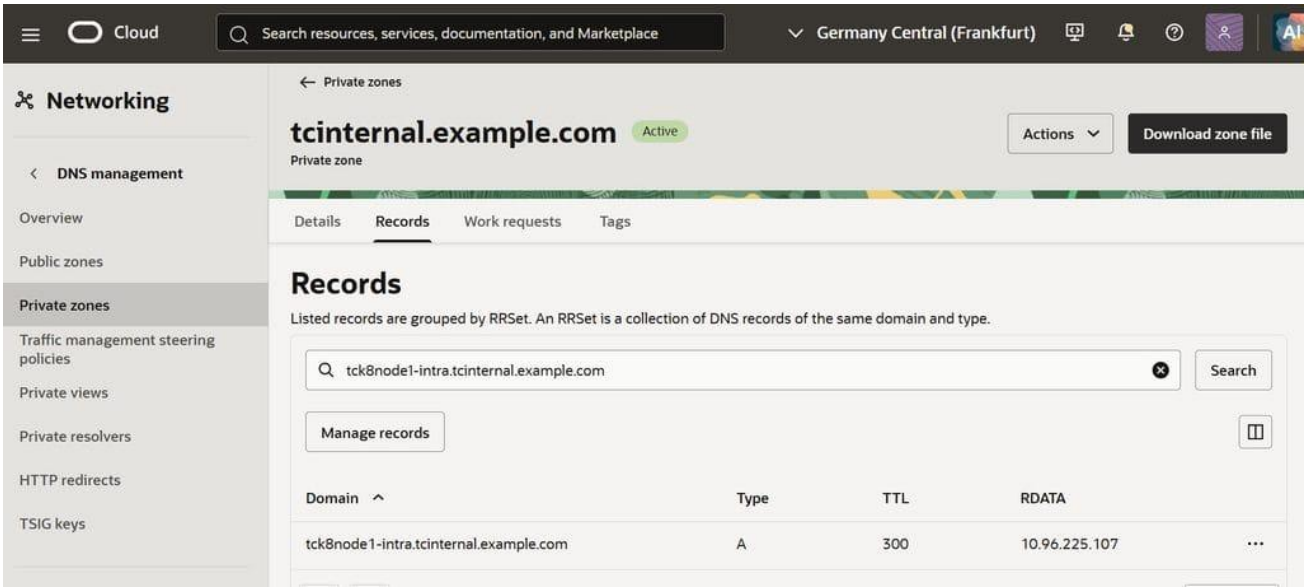
In this example, the application and Oracle True Cache run in the same Kubernetes cluster. The application can connect to Oracle True Cache through the internal Kubernetes service address and listener port. To find the ClusterIP address, list the services in the True Cache namespace:

```
kubectl get all -n tcdboper

NAME                    READY   STATUS    RESTARTS   AGE
pod/tck8node1-71q0u     0/1     Running   0          70s
NAME                  TYPE         CLUSTER-IP    EXTERNAL-IP  PORT(S)              AGE
service/tck8node1     ClusterIP    10.96.225.107 <none>       1521/TCP             7m16s
service/tck8node1-lb LoadBalancer 10.96.242.4   10.4.81.245 5500:30470/TCP,1521:31815/TCP 139m
```

When connecting from an application in the same Kubernetes cluster, use the service's ClusterIP address (for example, 10.96.225.107). For easier service discovery, create a DNS record that resolves to this address, as shown below:

Figure 5 DNS record resolving to ClusterIP address



Copyright © 2026, Oracle and/or its affiliates / Public

This example connects the application to Oracle True Cache using the two-physical-connections method and defines two separate data source pools:

- The general data source points to the primary database and can be used for any type of operation. In this example, its connection string is:

```
drdbrac15a-scan.dbsubnet.vcnfra80.oraclevcn.com:1521/myappsvc.example.com
```

- The read-only data source is intended for read operations and points to Oracle True Cache through a hostname that resolves to the service's ClusterIP address. In this example, its connection string is:

```
tck8node1-intra.tcinternal.example.com:1521/tcokenodes.example.com
```

Alternatively, use the full descriptor format. Example for the read-only data source in full descriptor format:

```
(DESCRIPTION=
  (ADDRESS=(PROTOCOL=TCP)(HOST=tck8node1-intra.tcinternal.example.com)(PORT=1521))
  (CONNECT_DATA=(SERVER=DEDICATED)(SERVICE_NAME=tcokenodes.example.com))
)
```

And that's it, you're ready to send your application's read workload to True Cache!!

References

- Oracle Kubernetes Engine – Creating a Cluster
- Oracle Database Operator for Kubernetes for Kubernetes – https://github.com/oracle/oracle-database-operator
- Oracle True Cache User's Guide – Methods for Connecting to True Cache
- Oracle True Cache User's Guide – Monitoring True Cache with the V$TRUE_CACHE_STAT View
- Oracle Cloud Infrastructure Documentation – Network Load Balancer
- ExternalDNS for Shared Private OCI DNS – https://github.com/oracle/oracle-database-operator/tree/main/docs/sidb/external-dns

Copyright © 2026, Oracle and/or its affiliates / Public

ORACLE

Connect with us

Call +**1.800.ORACLE1** or visit **oracle.com**. Outside North America, find your local office at: **oracle.com/contact**.

blogs.oracle.com facebook.com/oracle twitter.com/oracle

Copyright © 2026, Oracle and/or its affiliates. This document is provided for information purposes only, and the contents hereof are subject to change without notice. This document is not warranted to be error-free, nor subject to any other warranties or conditions, whether expressed orally or implied in law, including implied warranties and conditions of merchantability or fitness for a particular purpose. We specifically disclaim any liability with respect to this document, and no contractual obligations are formed either directly or indirectly by this document. This document may not be reproduced or transmitted in any form or by any means, electronic or mechanical, for any purpose, without our prior written permission.

Oracle, Java, MySQL, and NetSuite are registered trademarks of Oracle and/or its affiliates. Other names may be trademarks of their respective owners.