

Deploying TimesTen Scaleout On Oracle Cloud Infrastructure

Quickstart White Paper

ORACLE WHITE PAPER | APRIL 2018





Disclaimer

The following is intended to outline our general product direction. It is intended for information purposes only, and may not be incorporated into any contract. It is not a commitment to deliver any material, code, or functionality, and should not be relied upon in making purchasing decisions. The development, release, and timing of any features or functionality described for Oracle's products remains at the sole discretion of Oracle.



ORACLE®



Table of Contents

Disclaimer	1
Introduction	3
TimesTen Scaleout on OCI	3
Planning Your TimesTen Scaleout Deployment	5
Preparing For Your TimesTen Scaleout Deployment	5
Software Downloads and Installation Steps	6
Setting Required Variables	9
Deploying Cloud Resources and TimesTen Scaleout	10
Running Terraform	10
Running Ansible	11
Accessing The Database	13
Client Connectivity	14
OCI Connectivity Options	14
Database Client Access	15
Customizing the Configuration	16
Oracle Linux Operating System Images	16
Availability Domains	16
Data Instances	16
Management Instances	17
ZooKeeper Servers	18
Bastion Hosts	18
Managing Your Configuration	18



Systemd	18
ZooKeeper	18
Management instances	19
Data Instances	19
Stop	19
Scaleout	20
Future Work	22

"We are proud to announce the release of TimesTen Scaleout, a new scaleout in-memory database for OLTP workloads. Since it is based on a mature and time-tested TimesTen In-Memory Database, TimesTen Scaleout has both extensive sophisticated functionality, as well as incredible performance. This scaleout architecture is designed for extreme performance OLTP workloads and further extends Oracle's in-memory database technology leadership."

ANDREW MENDELSON
EXECUTIVE VICE PRESIDENT, ORACLE DATABASE
ORACLE CORP

Introduction

Oracle TimesTen In-Memory Database is the industry-leading in-memory database for OLTP applications, with thousands of customers across many industries. In release 18.1 TimesTen evolves into a fully transparent, shared-nothing, scale-out database with a new architecture called TimesTen Scaleout. This architecture now enables TimesTen to scale across dozens of hosts, reach many terabytes in size and support near-linear scaling of transaction throughput to hundreds of millions per second. All of this is possible using standard SQL without the need for manual database sharding or application partitioning.

Oracle Cloud Infrastructure provides high performance compute, network and storage ideal for deploying TimesTen Scaleout. This paper describes a set of Terraform and Ansible¹ scripts that can be used to provision TimesTen Scaleout on Oracle Cloud Infrastructure (OCI). For detailed information on the TimesTen Scaleout please see the [documentation](#).

TimesTen Scaleout on OCI

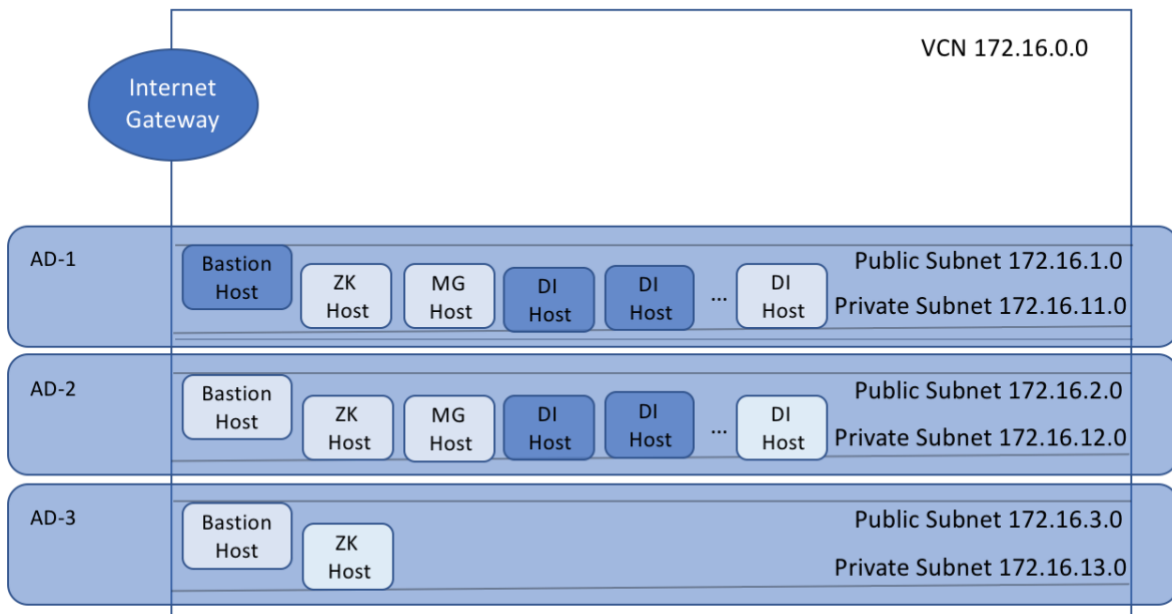
The scripts provide an example of provisioning TimesTen Scaleout on OCI. They are modeled on the [examples for the Terraform Provider for OCI](#). At a high level, the following operations are performed:

- » Using Terraform provision cloud resources such as a Virtual Compute Network (VCN), security lists, public and private subnets, Bare Metal or VM compute instances
- » Using Ansible configure the resources to prepare them for TimesTen Scaleout deployment.
- » Using Ansible then deploy the ZooKeeper servers, management and data instances that comprise a single database.

The following diagram shows deployment possibilities. Bastion hosts run on public subnets allowing ssh access. ZooKeeper (ZK), management (MG) and data (DI) compute instances run on private subnets. All servers use a NAT configuration so the outside world can be reached from the hosts, however only Bastion

¹ [Hashicorp Terraform](#) and [Ansible Core Project](#) are available in Oracle Linux 7 yum repositories

hosts are reachable from the public internet. Customers can add OCI VPN or DRG Gateways, use VCN Peering or [FastConnect](#) to access the database from outside the compartment. The deployment diagram is explained in more detail below.



Oracle TimesTen Scaleout Deployment Possibilities. Dark shading indicates default host deployment.

In the diagram above, the dark boxes represent the default configuration produced by the scripts, consisting of a bastion host and two hosts for data instances. The lighter boxes depict optional components. In the default configuration shown with the dark boxes, the ZooKeeper servers and two management instances configured as an active / standby pair are co-located with the data instances. The compute instances comprise an N x K TimesTen Scaleout configuration where N represents the number of replica sets of the database and K is the K-safety or number of replicas. The default is to create a 2x2 configuration, where there are 2 replica sets and 2 copies of the data.

Through script variables, the configuration can be changed to offload ZooKeeper and management instances to their own hosts (as depicted using lighter boxes in the diagram). Further larger values of N can be used to distribute the database across more hosts. At this time, only a K-safety value of 2 is supported.



Planning Your TimesTen Scaleout Deployment

The discussion below assumes that you are familiar with Oracle Cloud Infrastructure and TimesTen Scaleout. A good introduction to OCI can be found in the, [“OCI Getting Started Guide”](#). A good introduction for TimesTen Scaleout is available, [here](#).

To get started in planning, you need to choose the number of replica sets in the database (N in NxK) and the compute instance shape based on your workload. The default data instance compute shape uses VM.DenseIO.1.4 shapes, with 4 cores (OCPUs), 60 GB RAM, 1.2 Gbps network bandwidth and 3.2 TB NVMe SSD. It is recommended to leave about 10 GB memory for file system and other operating systems overhead. For example, the total shared memory used for the database on the VM.Dense.1.4 shape (PermSize+TempSize+LogBufMB+20 MB) should not exceed ~50 GB. For direct linked applications, it is necessary to plan to use shapes with sufficient memory for both application and database requirements.


The data instances must use shapes with NVMe; HighIO or DenseIO shapes are supported. On shapes with more than 4 NVMe devices only up to 4 devices are configured. On shapes with fewer devices, all the NVMe disk is used. With 4 devices you can choose an *mdraid*, RAID 10 striped and mirrored configuration. By default, the disks are striped using LVM in a RAID 0 configuration. Since the database uses a value of K-safety = 2, data loss in the event of a failed disk is avoided as there is a redundant copy of the data on another host.

The shapes chosen affect the network bandwidth available to the database as shown in the [OCI Components For Launching Images table](#). TimesTen Scaleout workloads with lots of connections that do not consider locality of reference can use up the network bandwidth. The utility *dstat* is installed on the data instances so that network bandwidth (along with other resources) can be easily monitored.

If ZooKeeper and management instances are offloaded to their own compute instances, the default shape for those instances is chosen to be a VM.Standard.1.1 shape. Larger shapes can be chosen by setting script variables if customer applications are to be co-located on these compute hosts.

Preparing For Your TimesTen Scaleout Deployment

Terraform is a tool for managing infrastructure in code. It allows for provisioning, modifying, versioning and managing infrastructure components. Terraform examines the existing state of the infrastructure based on configuration files and generates an execution plan. The plan can then be executed to achieve the desired state, here, provisioning, deleting or modifying the networking components and server hosts for deploying TimesTen Scaleout. For basic information about Terraform, see the following sites:

- 
- » <https://github.com/oracle/terraform-provider-oci>
 - » <https://community.oracle.com/community/oracle-cloud/cloud-infrastructure/blog/2017/02/15/terraform-and-oracle-bare-metal-cloud-services>

The scripts described here are intended to follow other Terraform examples for OCI located in the examples folder of the terraform-provider-oci Git repository at

<https://github.com/oracle/terraform-provider-oci/tree/master/docs/examples>

Ansible is a tool, similar to Chef or Puppet, used for automating system management operations (DevOps). Ansible core project scripts are used to configure the operating system, software and install the database. The *ansible* binary is installed on the OCI Linux servers by the provided scripts using *yum* from Oracle repositories. Ansible was chosen since it doesn't require access to external servers nor repositories.

Software Downloads and Installation Steps

The components you must download and installation steps are described below. You must be able to access your tenancy and the compartment in the Oracle Cloud from the host where you wish to run the scripts to deploy your service. The host can be on premises or in the Oracle Cloud. Running the scripts requires an Oracle Linux 6 or 7 system or a system running MacOS High Sierra 10.13.4, with python version 2.6 or greater. The following components are necessary.

- » **Terraform Provider for Oracle Cloud Infrastructure**
- » **Terraform Binary**
- » **OCI Credentials and OCIDs**
- » **TimesTen Scaleout OCI Deployment Scripts**
- » **TimesTen Scaleout Distribution**
- » **JDK 8 Distribution**

Each of these is described in more detail below.


- » **Terraform Provider for Oracle Cloud Infrastructure and Terraform binary.**

The OCI provider is a Terraform plugin created by the Oracle OCI Cloud team to operate directly with OCI REST and native control plane endpoints. The Terraform binary is the command line utility that uses the OCI plugin and the configuration scripts to plan, create or destroy the cloud infrastructure.

The terraform-oci-provider installation documentation describes how to download and install both the oci terraform provider and the terraform binary. The documentation can be found here:

<https://github.com/oracle/terraform-provider-oci>.

- » The Terraform OCI provider can be downloaded from:
<https://github.com/oracle/terraform-provider-oci/releases>.



Download the linux_amd64.tar.gz for Oracle Linux or the darwin_amd64.tar.gz file for MacOS as is appropriate.

» The Terraform binary can be downloaded from:

<https://www.terraform.io/downloads.html>

To install the provider unpack the tar file

```
% mkdir -p ~/.terraform.d/plugins
% tar -xzf linux_amd64.tar.gz
% cp -r linux_amd64 ~/.terraform.d/plugins
```

To install the terraform binary,

```
% mkdir -p ~/.oci
% unzip terraform_0.11.7_linux_amd64.zip # your version may be different than 0.11.7
% cp terraform ~/.oci
```

» **OCI Credentials and OCIDs.**

If you haven't done so already, it's necessary to upload a public key in PEM format for the user that will run the scripts to provision the cloud infrastructure and TimesTen Scaleout. For detailed instructions, please see:

<https://docs.us-phoenix-1.oraclecloud.com/Content/API/Concepts/apisigningkey.htm>

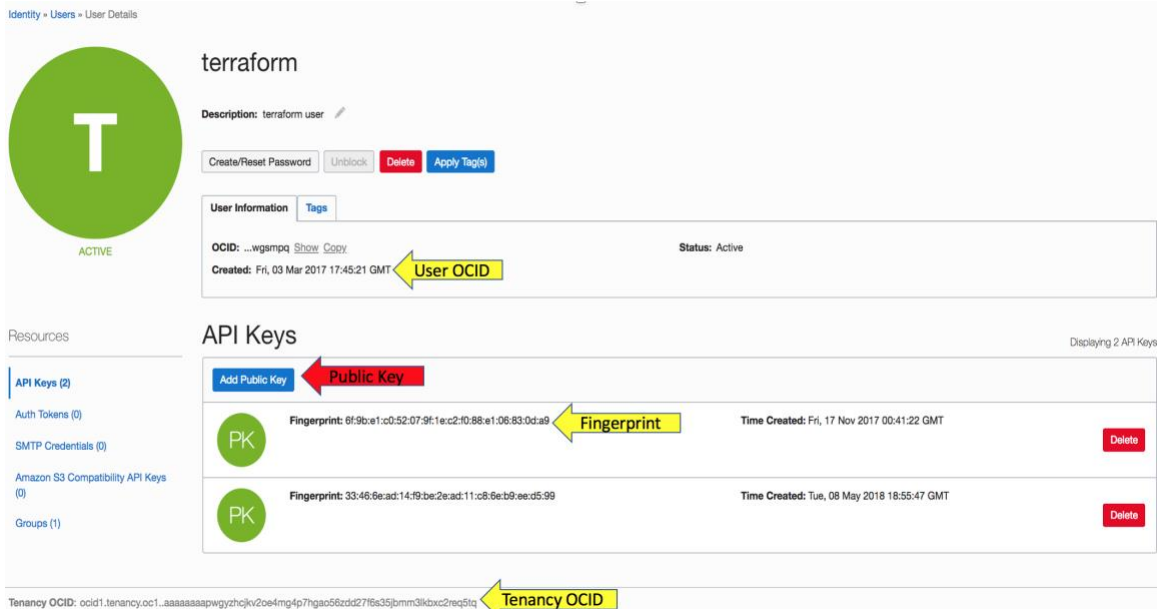
A brief outline of the steps necessary would be to run the following (note these instructions may be superseded by those at the link above)

```
% mkdir -p ~/.oci/keys
% cd ~/.oci/keys
# generate keys that do not require a passphrase
% openssl genrsa -out oci_api_key.pem 2048
# remove potential group or other user promiscuity
% chmod go-rwx oci_api_key.pem
# generate public key in PEM format
% openssl rsa -pubout -in oci_api_key.pem -out oci_api_key_public.pem
# generate signature key fingerprint necessary for use with scripts
% openssl rsa -pubout -outform DER -in oci_api_key.pem | openssl md5 -c
```

The public key in PEM format generated above, *oci_api_key_public.pem*, needs to be uploaded to OCI. This is done through the [OCI Console](#) under Identity->Users or by clicking on your user account link at the top center right of the console and navigating to User Settings. Below is a screen shot for a user named "terraform", however, this needs to be done for the user that will own the OCI resources provisioned. The red arrow shows where to upload the public key.



Resources in the Oracle Cloud are described by identifiers known as OCIDs. For use with the scripts, you'll need the OCIDs for the tenancy and user, and the key fingerprint². These are shown with the yellow arrows in the screen shot. Also required is the OCID for the compartment where resources are to be created. Use the compartment assigned by your cloud administrator. The OCID can be found in the OCI console under Identity->Compartments.



OCI Screen Capture. The red arrow shows where to upload the public key. The yellow arrows show where to find the user OCID, tenancy OCID, and the key fingerprint (available after upload), that will be needed later.

» TimesTen Scaleout BYOL Scripts

The Terraform and Ansible scripts can be downloaded from the blog at <https://github.com/oracle/oracle-timesten-samples/tree/master/cloud/ottscaleout> Once downloaded, untar the distribution so that the components below can be put in place.

```
tar -xzf TimesTen ottscaleout-byol-oci.tar.gz
```

The top level directory is named *ottscaleout*.

TimesTen Scaleout Distribution

This is a bring-your-own-license (BYOL) solution. A TimesTen Scaleout distribution can be downloaded for evaluation from [OTN](#). For production use, please download the most recent patch release from ARU or eDelivery. The zipped tar file or zip file should be placed in the *ottscaleout/service/packages* directory.

² Users provisioning from a host within OCI can use [Instance Principal Authorization](#) instead.

» JDK 8 Distribution.

TimesTen Scaleout supports JDK8 which can be downloaded from

<http://www.oracle.com/technetwork/java/javase/downloads/jdk8-downloads-2133151.html>

Please select the *linux-x64.tar.gz* file placing it in the *ottscalout/service/packages* directory.

At this point the necessary components have been gathered, and it's time to set the variables necessary to launch the scripts.

Setting Required Variables

Variables that control the cloud and database provisioning are located in two files found in the top level *ottscaleout* directory, *env-vars* and *variables.tf*. The *env-vars* file contains environment variables that set access to OCI, while variables related to TimesTen Scaleout and the desired database configuration are found in *variables.tf* file.

Modify the *env-vars* file using the credentials and OCIDs gathered in the Installation section above.

```
# Region assigned by your cloud administrator (top center of OCI Console page)
export TF_VAR_region="us-phoenix-1"
# Tenancy, User OCIDs collected above from Identity->Users
export TF_VAR_tenancy_ocid="ocid1.tenancy. ..."
export TF_VAR_user_ocid="ocid1.user..."
export TF_VAR_fingerprint="1f:2b:..."
# Private key corresponding to public key uploaded to console
export TF_VAR_private_key_path="~/oci/keys/oci_api_key.pem"
# Compartment OCID assigned by your cloud administrator from Identity->Compartments
export TF_VAR_compartment_ocid="ocid1.compartment..."
```

The credentials above allow access to the OCI resources for provisioning cloud resources. Users provisioning from a compute instance in the OCI cloud can use [Instance Principal Authorization](#) instead, however that approach is not described here.

For *ssh* access to the bastion hosts provisioned, credentials in regular *rsa* format are required. For example,

```
export TF_VAR_ssh_public_key=$(cat ${HOME}/.ssh/id_rsa.pub)
export TF_VAR_ssh_private_key=$(cat ${HOME}/.ssh/id_rsa)
```

Beyond the above, there are no other required variables, rather everything else has default values that result in provisioning an NxK (2x2) database named *ttimdb1*.

Deploying Cloud Resources and TimesTen Scaleout

Running Terraform

Once variables have been set, it's time to run the scripts. The user running the scripts should be the one for whom the OCI credentials were established above.

```
% cd ottscaleout
```

```
# Before using terraform, to plan, apply or destroy, make sure to source the environment file. Following other terraform examples a bash compatible shell is required.
```

```
% ./env-vars
```

When run the first time, Terraform may need to use a proxy server to bypass local firewalls in order to download the "null_resource" and "template" providers from the internet.

```
# initialize terraform
```

```
% terraform init
```

```
Initializing provider plugins...
- Checking for available provider plugins on https://releases.hashicorp.com...
- Downloading plugin for provider "null" (1.0.0)...
- Downloading plugin for provider "template" (1.0.0)...
...
```

An optional step is view the resources terraform will provision. After provisioning, it is always a good idea to view the plan to understand what terraform will create or destroy.

```
% terraform plan
```

```
Refreshing Terraform state in-memory prior to plan...
...
Plan: 50 to add, 0 to change, 0 to destroy.
```

The output is extensive so you may wish to save it in a file. If the plan looks OK, then deploy the cloud resources. Adding the `--auto-approve` option (two dashes --) avoids having to type 'yes' at a confirmation prompt.

```
% terraform apply --auto-approve
```

```
data.oci_identity_availability_domains.ADs: Refreshing state...
...
```

Allocating the default 2x2 cloud infrastructure takes about 4 minutes when the [speedtest](#) shows latency of 80 ms³. Again there is extensive output. When complete, the output provides the address(es) of the Bastion host(s).

Apply complete! Resources: 23 added, 0 changed, 0 destroyed.

Outputs:

```
InstanceIPAddresses = [
  bastion host instances (public addresses):
  ssh opc@129.213.40.166 ,
  database [mgmt|zookeeper] hosts (private addresses):
  ttimdb1-di-001 172.16.10.2
  ttimdb1-di-002 172.16.11.2
  ttimdb1-di-003 172.16.10.3
  ttimdb1-di-004 172.16.11.3
```

Terraform uses a file, 'terraform.tfstate' by default, to keep track of the state of allocated resources. The file is necessary to destroy the configuration with *terraform* or to make changes. It's a good idea to make a copy and save it in a safe place. The configure file can be used to destroy the resources via terraform. At this point cloud resources have been provisioned, however, the systems have not been configured and the database is not yet running. If there has been a problem along the way, assuming you can correct the issue, you can rerun terraform and it will correct any changes along the way, then continue where it left off.

To destroy your configuration, run:

```
% terraform destroy -force
```

which frees all resources. Destroy can also be used if there was a failure along the way and you want to start over. Assuming you want to use what's been created, the next step is to *ssh* to the Bastion host as user *opc* to configure the hosts and TimesTen Scaleout.

Running Ansible

It is possible to have Terraform run the ansible scripts, however, Terraform may timeout completing the installation with a large configuration. Instead, cut and paste the *ssh* string from the terraform output to access the Bastion host to run the scripts.

```
% ssh opc@129.213.40.166
...
Are you sure you want to continue connecting (yes/no)? yes
% cd service/ansible
% ansible-playbook -i ./hosts rollout.yml
PLAY [bastion-hosts] *****
...
```

³ Latency is only one metric. It doesn't completely explain how long the operations will take.

It takes several minutes to run through the playbooks (~15 mins when [speedtest](#) shows 80 ms latency). Output is extensive, especially with the optional `-v` option. When complete, the output shows the TimesTen Scaleout configuration and access information. The host names for the different compute images use a prefix based on the service-name, a 2 letter indicator [zk|mg|di] depending on the host usage, and a 3 digit suffix, e.g. `ttimdb1-di-001`.

```
TASK [datainstances : dbstatus output *****
ok: [ttimdb1-di-001] => {
  "msg": [
    "Database ttimdb1 Replica Set status as of Fri Apr 27 17:15:34 GMT 2018",
    "",
    "RS DS Elem Host      Instance Status Date/Time of Event  Message ",
    "-----",
    " 1 1  1 ttimdb1-di-001 instance1 opened 2018-04-27 17:15:20  ",
    "  2  2 ttimdb1-di-002 instance2 opened 2018-04-27 17:15:20  ",
    " 2 1  3 ttimdb1-di-003 instance3 opened 2018-04-27 17:15:20  ",
    "  2  4 ttimdb1-di-004 instance4 opened 2018-04-27 17:15:20  "
  ]
}
```

Above the output shows a 2x2 database running on `ttimdb1-di-001` through `ttimdb1-di-004`. The database has four instances, *instance1* through *instance4* that are open and ready for use.

```
TASK [datainstances : ttgridrollout output] *****
ok: [ttimdb1-di-001] => {
  "msg": [
    "Management Instance Locations",
    "-----",
    "- ttimdb1-di-001:/u10/TimesTen/ttimdb1/iron_mgmt",
    "- ttimdb1-di-002:/u10/TimesTen/ttimdb1/iron_mgmt2",
    "",
    "Please source ttenv script under Management Instances for grid management via \"ttGridAdmin\" commands.",
    "",
    " For example, to use the first management instance, on ttimdb11-di-001:",
    " sh: ./u10/TimesTen/ttimdb1/iron_mgmt/bin/ttenv.sh",
    " csh: source /u10/TimesTen/ttimdb1/iron_mgmt/bin/ttenv.csh",
    "",
    "",
    "Data Instance Locations",
    "-----",
    "- ttimdb1-di-001.instance1 ==> ttimdb1-di-001:/u10/TimesTen/ttimdb1/instance1",
    "- ttimdb1-di-002.instance2 ==> ttimdb1-di-002:/u10/TimesTen/ttimdb1/instance2",
  ]
}
```

```

"- ttimdb1-di-003.instance1 ==> ttimdb1-di-003:/u10/TimesTen/ttimdb1/instance1",
"- ttimdb1-di-004.instance2 ==> ttimdb1-di-004:/u10/TimesTen/ttimdb1/instance2",
""
"Please source ttenv script under Data Instances for database operations.",
""
" For example, to use instance1, on ttimdb1-di-001:",
" sh: ./u10/TimesTen/ttimdb1/instance1/bin/ttenv.sh",
" csh: source /u10/TimesTen/ttimdb1/instance1/bin/ttenv.csh"
]
}

```

The above shows management and data instances running on `ttimdb1-di-001` – `ttimdb1-di-004` and the commands to connect to them. We hope that there hasn't been any issues along the way, but if there are, assuming you can correct them, you need merely rerun *ansible* to continue.

Accessing The Database

From the Bastion host, you can *ssh* to any other host in the configuration. Perhaps the first thing to do to use the database is to create an administrative user. It is necessary to run on a data instance as the instance administrator user, *oracle*.

```

[opc@ttimdb1-bs-001 ~]$ ssh ttimdb1-di-001
Last login: Fri Apr ...
[opc@ttimdb1-bs-001 ~]$ sudo su - oracle
[oracle@ttimdb1-di-001 ~]$

```

Using the output from running *ansible* above, the instance for `ttimdb1-di-001` is located at:

```


/u10/TimesTen/ttimdb1/instance1

```

```

[oracle@ttimdb1-di-001 ~]$ /u10/TimesTen/ttimdb1/instance1/bin/ttenv ttisql dsn=ttimdb1
...
connect "dsn=ttimdb1";
...
Command> create user appuser identified by appuser;
User created.
Command> grant admin to appuser;
Command> quit;
Disconnecting...
Done.

```



The command above creates an admin user named `appuser` with password `appuser`. You may also want to run `ttStatus` and make note of the daemon port (default here is 46464).

```
$ ~J$ /u10/TimesTen/ttimdb1/instance1/bin/ttenv ttstatus
TimesTen status report as of Fri Apr 27 21:49:53 2018
...
Daemon pid 31052 port 46464 instance instance1
```

Later, a trick for accessing hosts on the private networks from outside the tenancy with a single command invocation is to use the `ssh -J` option (jump proxy).

```
ssh -J opc@129.213.35.214 opc@ttimdb1-di-001
...
Are you sure you want to continue connecting (yes/no)? yes
...
```

For example, you'd run this on the system where terraform was run to go directly to a management or data instance. The `ssh` command first connects to the Bastion host, then to the target, in this case, `ttimdb1-di-001`.

You're ready to run applications! Almost.

Client Connectivity

OCI Connectivity Options

There are several ways to connect applications to the database. TimesTen Scaleout direct-linked applications can be deployed on the same compute instances as the database themselves. Please ensure that the shapes chosen for the deployment are sufficient to account for both application and database resources, including network bandwidth, number of CPUs, memory capacity, disk capacity and disk bandwidth.

Separate client systems can be deployed on the private subnets provided. It is not recommended to install clients on the public networks. Installing on the private networks provides low latency, high bandwidth access, since client systems are in the same compartment. If the management and or ZooKeeper servers are running on their own separate compute instances, those may be used as clients given they have been provisioned with sufficient resources to run the client applications.

OCI provides many other options for connecting to the database in your cloud tenancy from the outside including VCN peering, FastConnect and VPN access. These options provide high bandwidth, low latency access from outside the tenancy.

Database Client Access

Client connection information is gleaned from a management instance. In our case, the management instance is co-located on the same host system as the data instance. It is necessary to connect to a management instance. Again, using the ansible output from above:

```
[oracle@ttimdb1-di-001 ~]$ /u10/TimesTen/ttimdb1/iron_mgmt/bin/ttenv ttgridadmin gridclientexport client.odbc.ini
[ODBC Data Sources]
ttimdb1cs=TimesTen 18.1 Client Driver
[ttimdb1cs]
TTC_SERVER_DSN=TTIMDB1
# External address/port info for ttimdb1-di-002.instance2
TTC_SERVER1=ttimdb1-di-002/46465
# External address/port info for ttimdb1-di-001.instance1
TTC_SERVER2=ttimdb1-di-001/46465
TTC_Timeout=60...
```

To install a client, you'll need to create a client instance. These commands could be run on a separate VM you've allocated within the tenancy for running your application; from offloaded management or ZooKeeper VM's or from the bastion host (later not recommended except for testing).

First, unzip the distribution,

```
% unzip -q timesten181120.server.linux8664.zip # or tar -xzf timesten181120.server.linux8664.tgz
% ./tt18.1.1.2.0/bin/ttInstanceCreate -grid -name myclient -location /home/oracle -serverhost ttimdb1-di-001 -
  csport 46465 -clientonly
# Replace the sys.odbc.ini file with gridclientexport output
% mv /home/oracle/myclient/conf/sys.odbc.ini /tmp
% cp client.odbc.ini /home/oracle/myclient/conf/sys.odbc.ini
# source environment
% . /home/oracle/myclient/bin/ttenv.sh
...
# need to have created a user before trying to connect client
% ttisqlcs dsn=ttimdb1cs\;uid=appuser
...
```

You are good to go!

Customizing the Configuration

There are a number of variables that allow control over the initial provisioning settings and configuration. The variables are located in the *variables.tf* file. These variables are discussed in the next sections.

Oracle Linux Operating System Images

The same Oracle Linux Image is installed on all provisioned compute instances. As of this writing, this is an Oracle Linux 7.5 image. To set the image to be installed, update the “InstanceImageOCID” map in *variables.tf*. Please see <https://docs.us-phoenix-1.oraclecloud.com/images/> for available images.

Upon installation, *yum* updates all packages in the image for which there are security errata available if the *securityupdates* variable is *true* in *variables.tf*. By default, the image is **not** updated. A cronjob lists by CVE any security updates available, writing the output to the */home/opc/latest-cves* file. If you'd like to apply security updates upon provisioning, choosing more recent images results in faster startup time as there is generally less errata to apply. It is up to you to apply any updates necessary to maintain the provisioned hosts.

Availability Domains

By default, the data compute instances for the 2x2 configuration are created in availability domains AD-1 and AD-2. To change the provisioning to use AD-2 and AD-3, set the environment variable in *variables.tf*:

```
variable "initial_AD" { default = "2" }
```

The default setting is *initial_AD=1*. Setting *initial_AD=3*, provisions in AD-3 and AD-1. This variable also controls the AD's where management instances are provisioned if they are offloaded to their own VMs. If the ZooKeeper servers are offloaded, then their VMs span all three availability domains.

Data Instances

For high availability purposes, database replica sets span across availability domains. Only a K-safety value, K=2 is supported. So one TimesTen Scaleout data space is in AD-1, and the other is in AD-2. The number and shapes of the data instances are controlled by environment variables in *variables.tf*.

```
# the N in NxK
```

```
variable "diInstanceCount" { default = "2" }
```

```
# Compute instance shape for data instances
```

```
# N*K VMs/BMs are provisioned for data instances.
```

```
# Use NVMe shape(DenseIO or HighIO)
```

```
variable "diInstanceShape" { default = "VM.DenseIO1.4" }
```



The name of the database is set in `env_vars` with:

```
export TF_VAR_service_name="ttimdb1"
```

This variable also controls the vcn name, and the directory structure. For example, the installation, and instance directories are subdirectories of a top level directory named `ttimdb1`, in this case. Changing the name, enables another set of resources with a different VCN and database to be provisioned in the same compartment.

Other database attributes such as `PermSize` and `DatabaseCharacterSet` can be changed using the `"timesten"` map variable in the `variables.tf` file.

The database runs under user `oracle`. That is, the user `oracle` is the Instance Administrator for the database. There is no password for the `oracle` user. Access to the `oracle` account is through `sudo` from the `opc` user account.

```
sudo su - oracle
```

Once logged in as `oracle`, `ssh` can be used to access the `oracle` account on other hosts. If it is undesirable to have users `sudo` from the `opc` account, then the ssh keys under `/home/oracle/.ssh` can be exported to the user.

The `installation_location` for TimesTen Scaleout is under `/home/oracle/<service_name>/<version>`; `/home/oracle/ttimdb1/tt18.1.1.2.0` by default. The data instance directories, database files, and transaction log files are located under `/<fsname>/TimesTen/<service-name>`, `/u10/TimesTen/ttimdb1` by default. The filesystem name, `"fsname"`, can be changed in `variables.tf`.

The `"fsname"` (`/u10 default`) filesystem is the mount point for the NVMe devices. As discussed in 'Planning' above, no more than 4 NVMe devices are configured under this mount point. If more than one device is present, the default is to stripe the devices using LVM. To use `mdraid` to create a RAID 10 striped and mirrored configuration, instead of LVM striping, set the variable `"storage"="MD-RAID-10"` in `variables.tf`. It takes significant time to create a filesystem on an `mdraid` device.

Management Instances

By default, management instances are co-located on the same VMs as the data instances. Two management instances are created, that like the database replica sets, span availability domains. The management instances can be offloaded to their own VMs by setting, variable `"mgInstanceCount" { default = "2" }` in `variables.tf`. When offloaded, the management instances use a `VM.Standard.1.1` shape. The shape may also be changed in `variables.tf`. Each management instance requires less than 1 GB of RAM, about 2 GB storage, and uses very little CPU or network bandwidth.

ZooKeeper Servers

TimesTen Scaleout uses ZooKeeper as a membership service. By default, 3 ZooKeeper servers are set running, co-located with the data instances. If there are more than 2 data instances, then 3 of them will run ZooKeeper servers. If there are separate management instances, 2 of the ZooKeeper servers will run on the management instance hosts, and the third will run on a data compute instance. ZooKeeper servers may be offloaded to their own VMs, by setting

```
variable "zkInstanceCount" { default = "3" }
```

In this case, the 3 ZooKeeper Servers will span all 3 availability domains. The default shape for the offloaded servers is a *VM.Standard.1.1* shape. ZooKeeper servers require ~1 GB of memory but use little CPU or network bandwidth.

Bastion Hosts

Bastion hosts provide an internet firewall and are used as ssh gateways to the rest of the configuration. Use of a bastion host is considered best practice for OCI. For more information please read, [Bastion Hosts: Protected Access for Virtual Cloud Networks](#). By default, only 1 bastion host is provisioned, but the implementation allows for the highly available configuration described in the OCI white paper, [NAT Instance Configuration: Enabling Internet Access for Private Subnets](#), by increasing the *bsInstanceCount* variable in *variables.tf*. The scripts described in the paper for the H/A configuration are not deployed on the bastion hosts, however, copies can be extracted from the whitepaper or can be found in the *service/scripts* directory. The number of bastion hosts and their shape are configured in the *variables.tf* file.

Managing Your Configuration

Systemd

The ZooKeeper servers, management and data instances are running as services under *systemd*. If one of these compute instances goes down, then *systemd* attempts a restart. To generically check the status of the services, one of the commands can be used below from the *opc* user account.

```
% sudo systemctl status <service-name>
% sudo journalctl -u <service-name> # -f option can be added to follow updates dynamically.
```

ZooKeeper

The *systemd* service for ZooKeeper is named *<service-name>-zk1.service*. For troubleshooting purposes, *ncat* is installed on the compute instances to enable sending the ZooKeeper four letter commands such as *'stat'*.



```
$ echo stat | nc ttimdb1-di-002 2181
```

```
Zookeeper version: 3.4.10-39d3a4f269333c922ed3db283be479f9deacaa0f, built on 03/23/2017 10:13 GMT
Clients:
  /172.16.10.2:59199[0] (queued=0, recved=1, sent=0)
Latency min/avg/max: 0/0/0
Received: 2
Sent: 1
Connections: 1
Outstanding: 0
Zxid: 0x10000001d
Mode: leader
Node count: 21
```

Management instances

The management instances, in particular the active management instance, is where the database is managed. Operations on management instances under TimesTen Scaleout are performed by the *ttgridadmin* utility. To monitor a management instance, *ttgridadmin mgmtexamine* is expected to be run periodically. In the event a compute instance becomes unavailable, the output of *mgmtexamine* provides recommended actions to take to failover or restart the instances. Scripts are installed under *systemd* that periodically check the state of the management instances and run the recommended actions from *ttgridadmin mgmtexamine* in the event an instance goes down. For the management database, *systemd* runs a service that continuously checks the state of the management database by running a script named, `'/home/oracle/bin/mgmtexamine.py'` The service files are named, `<service-name>-mgmt.service` and `<service-name>-mgmt.timer`. If the management instances are destroyed, or moved to different compute hosts, without using the provided scripts, please ensure the corresponding services are shut down.

Data Instances

The database service files are named `<host>.service`, e.g `ttimdb1-di-001.service`. The data instances will be restarted on reboot or an abnormal exit, that is one where the return code is not 0 and not via one of the signals, `SIGHUP`, `SIGINT`, `SIGTERM`, nor `SIGPIPE`.

Stop

Ansible scripts are provided to stop the database, management and ZooKeeper servers and their corresponding *systemd* services in that order

```
% ansible-playbook -i hosts stop.yml # or
% ansible-playbook -l hosts -e stoptime=<time-in-seconds> stop.yml
```



The stop operation closes and unloads the database from memory by running the commands:

```
ttgridadmin dbclose <dbname> -wait <time-in-seconds>
ttgridadmin dbunload <dbname> -wait <time-in-seconds>
```

The *<time-in-seconds>* parameter can be an empty string meaning wait until complete. By default, stoptime is set to 5 minutes but can be configured in *variables.tf* or on the command line as above. The ansible scripts will fail out if a timeout occurs. All client-server connections to the database need to be terminated before the database can be unloaded.

After the database is unloaded the daemons and services are stopped. Next, the management services are stopped with *'ttgridadmin mgmtstandbystop'* and *'mgmtactivestop'*. Finally, the ZooKeeper provided *'zkServer.sh stop'* script is run to halt the ZooKeeper servers.

The scripts do not turn off the compute resources used, rather that needs to be done through the OCI console. Restart scripts are not yet provided. To restart, use *systemd* to first restart the ZooKeeper servers, the management instances and then the data instances in that order.

Scaleout

A limited form of scaling out the number of **data instances** can be achieved by increasing the variable count of data instances available, the N in NxK. For example, if you started with a 1x2 configuration, to go to a 2x2 configuration, increase N to 2.

```
Edit variables.tf, setting "diInstanceCount" { default = "2" }
```

Terraform plan will show an addition of two data instances.

```
% terraform plan
Refreshing Terraform state in-memory prior to plan...
...
+ oci_core_instance.di_instance[2]
...
+ oci_core_instance.di_instance[3]
...
Plan: 4 to add, 0 to change, 2 to destroy.
...
```

Once the plan looks good, create the data instances. The *–auto-approve* option has two dashes.

```
% terraform apply –auto-approve
oci_core_virtual_network.CoreVCN: Refreshing state...
```

```

...
Apply complete! Resources: 4 added, 0 changed, 2 destroyed.
Outputs:
InstanceIPAddresses = [
  bastion host instances (public addresses):
  ssh opc@129.213.102.15 ,
  database [mgmt|zookeeper] hosts (private addresses):
  ttimdb1-di-001 172.16.10.2
  ttimdb1-di-002 172.16.11.2
  ttimdb1-di-003 172.16.10.3
  ttimdb1-di-004 172.16.11.3
]

```

Terraform has provisioned the data instances, updated the ansible hosts file and copied that file to the Bastion host. To add the instances to the configuration, login to the Bastion host to perform the following operations. Ansible will add the hosts, installations and instances, and then run dbDistribute to populate the elements. How long it takes to scale out depends on how much data needs to move to the new elements.

```

% ssh opc@129.213.102.15
Last login: ...
[opc@ttimdb1-bs-001 ~]$ cd service/ansible
[opc@ttimdb1-bs-001 ansible]$ ansible-playbook -i hosts scaleout.yml
...
PLAY [db-addresses]
...
"Host ttimdb1-di-003 created in Model",
  "Installation installation1 created in Model",
  "Instance instance1 created in Model",
  "",
  "Host ttimdb1-di-004 created in Model",
  "Installation installation1 created in Model",
  "Instance instance2 created in Model",
  "",
  "Creating new model version..."

```



If the scaleout has been successful, *ansible* displays the replica set information:

```
"RS DS Elem Host      Instance Status Date/Time of Event Message ",
"-----",
" 1 1 1 ttimdb1-di-001 instance1 opened 2018-06-20 17:39:15  ",
" 2 2 2 ttimdb1-di-002 instance2 opened 2018-06-20 17:39:16  ",
" 2 1 3 ttimdb1-di-003 instance1 opened 2018-06-20 17:39:16  ",
" 2 4 4 ttimdb1-di-004 instance2 opened 2018-06-20 17:39:16  "
```

Scaling the infrastructure has limitations. Scaling in, that is going from a 2x2 to a 1x2 configuration is not currently supported by the scripts. Nor is attempting to “scale up” or to “scale down” by changing the shape of the data instances. These operations result in terraform destroying the existing data instances, then provisioning new ones. Since there are no ansible scripts in place to perform these operations the database is left in an inconsistent state. In addition, scaling the number of management instances or ZooKeeper servers is not supported by the scripts.

Future Work

The scripts today primarily provision the service. It is the intention to evolve these scripts to improve availability and management of TimesTen Scaleout. Future work hopes to improve the life cycle management, take advantage of more OCI features, and improve availability.







Oracle Corporation, World Headquarters

500 Oracle Parkway
Redwood Shores, CA 94065, USA

Worldwide Inquiries

Phone: +1.650.506.7000
Fax: +1.650.506.7200

CONNECT WITH US

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

Integrated Cloud Applications & Platform Services

Copyright © 2018, Oracle and/or its affiliates. All rights reserved. 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 and Java are registered trademarks of Oracle and/or its affiliates. Other names may be trademarks of their respective owners.

Intel and Intel Xeon are trademarks or registered trademarks of Intel Corporation. All SPARC trademarks are used under license and are trademarks or registered trademarks of SPARC International, Inc. AMD, Opteron, the AMD logo, and the AMD Opteron logo are trademarks or registered trademarks of Advanced Micro Devices. UNIX is a registered trademark of The Open Group. 0116

Deploying Oracle TimesTen Scaleout On Oracle Cloud Infrastructure
April 2018
Author: steve.folkman@oracle.com
Contributing Authors: