ORACLE
TimesTen

Oracle Cloud InfrastructureでのTimesTen Scaleoutのデプロイ

クイックスタート・ホワイト・ペーパー

Oracleホワイト・ペーパー | 2018年10月V2

ORACLE

免責事項

下記事項は、弊社の一般的な製品の方向性に関する概要を説明するものです。また、情報提供を唯一の目的とするものであり、いかなる契約にも組み込むことはできません。マテリアルやコード、機能の提供をコミットメント（確約）するものではなく、購買を決定する際の判断材料になさらないで下さい。オラクルの製品に関して記載されている機能の開発、リリース、および時期については、弊社の裁量により決定されます。

目次

「OLTP ワークロードに適した新しいスケールアウト・インメモリ・データベースである TimesTen Scaleout のリリースを発表できることをうれしく思います。TimesTen Scaleout は長年の実績を持つ成熟した TimesTen In-Memory Database をベースとする製品で、非常に高度な機能と優れたパフォーマンスとを兼ね備えています。このスケールアウト・アーキテクチャは、非常に高性能な OLTP ワークロード向けに設計されており、インメモリ・データベース・テクノロジーにおけるオラクルのリーダーシップをさらに強化します。」

Oracle Database、Executive Vice President、
Andrew Mendelsohn
Oracle Corp

はじめに

Oracle TimesTen In-Memory Database は、業界を代表する OLTP アプリケーション向けのインメモリ・データベースで、さまざまな業界で数千のお客様に使用されています。リリース 18.1 の TimesTen は、完全に透過的なシェアード・ナッシングのスケールアウト・データベースへと進化し、TimesTen Scaleout という新しいアーキテクチャを備えています。このアーキテクチャにより、TimesTen では数十台のホストへのスケーリング、テラバイト単位の大規模なサイズの実現、1 秒当たり数億のトランザクション・スループットというほぼ直線的なスケーリングのサポートが可能になっています。これらはすべて、標準 SQL を使用して利用可能であり、手動のデータベース・シャーディングやアプリケーションのパーティショニングは不要です。

Oracle Cloud Infrastructure は、TimesTen Scaleout のデプロイに最適な、高性能のコンピュート、ネットワーク、およびストレージを備えています。本書では、Oracle Cloud Infrastructure（OCI）で TimesTen Scaleout をプロビジョニングするのに使用できる Terraform および Ansible[1]のスクリプトのセットについて説明します。TimesTen Scaleout についての詳しい情報は、ドキュメントを参照してください。

OCIでのTimesTen Scaleout

このスクリプトは、OCI に TimesTen Scaleout をプロビジョニングする例です。OCI 向けの Terraform プロバイダの例をモデルとしています。実行される操作の概要は以下のとおりです。

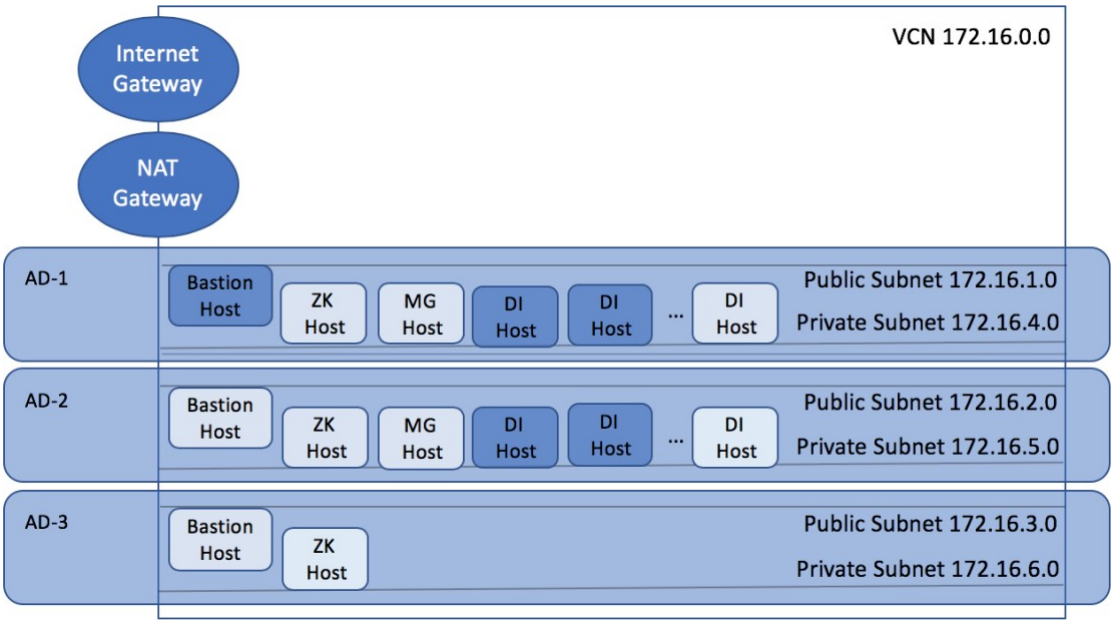
» Terraform を使用して、Virtual Compute Network（VCN）、セキュリティ・リスト、パブリックおよびプライベート・サブネット、ベア・メタルまたは VM のコンピュート・インスタンスなどのクラウド・リソースをプロビジョニングする

» Ansible を使用してリソースを構成し、TimesTen Scaleout のデプロイメントで利用できるようにする

[1] Hashicorp Terraform および Ansible Core Project は、Oracle Linux 7 Yum リポジトリで入手可能です。

» 次に Ansible を使用して、単一データベースを構成する ZooKeeper サーバー、管理インスタンス、データ・インスタンスをデプロイする

以下の図は、デプロイメントの例を示しています。Bastion ホストは、SSH アクセスが可能なパブリック・サブネットで稼働しています。ZooKeeper（ZK）、管理（MG）、データ（DI）コンピュート・インスタンスは、プライベート・サブネットで稼働しています。すべてのサーバーで NAT ゲートウェイ構成を使用しているため、ホストから外部にアクセスすることは可能ですが、パブリック・インターネットからアクセスできるのは Bastion ホストのみです。顧客は OCI VPN または DRG ゲートウェイを追加し、VCN ピアリングまたは FastConnect を使用して、コンパートメントの外部からデータベースにアクセスできます。デプロイメント図の詳細は以下で説明されています。



Oracle TimesTen Scaleoutデプロイメントの例。色の濃い部分はデフォルトのホスト・デプロイメントを指す。

上記の図では、色の濃いボックスはスクリプトで生成されるデフォルトの構成を表しており、Bastion のホスト 1 台と、データ・インスタンスのホスト 4 台で構成されています。明るい色のボックスは、オプションのコンポーネントを表しています。濃い色のボックスで示されているデフォルトの構成では、ZooKeeper サーバーと 2 つの管理インスタンスがアクティブ/スタンバイのペアとして構成され、データ・インスタンスと一緒に配置されています。コンピュート・インスタンスは、N x K の TimesTen Scaleout 構成で、N はデータベースのレプリカ・セットの数、K は K-safety またはレプリカ数です。デフォルトでは 2x2 構成で作成されます。この場合、レプリカ・セットが 2 つと、データのコピーが 2 つ（データ領域が 2 つ）あります。

スクリプトの変数を通して構成を変更し、ZooKeeper と管理インスタンスをそれらの独自のホストにオフロードできます（図では明るい色のボックスで示されています）。データベースをもっと多くのホストに分散させるために、N の値を大幅に増やすこともできます。K-safety の値は 1 と 2 がサポートされています。

TimesTen Scaleoutのデプロイメントの計画

この後の記述は、読者が Oracle Cloud Infrastructure と TimesTen Scaleout に慣れ親しんでいることを前提としています。OCI については、『OCI Getting Started Guide』にビギナー向けの情報が掲載されています。TimesTen Scaleout については、こちらにビギナー向けの情報が掲載されています。

計画に取り掛かるには、ワークロードに基づいて、データベースのレプリカ・セット数（NxK の N）とコンピュート・インスタンスのシェイプを選択する必要があります。データ・インスタンスのデフォルトのコンピュート・シェイプでは、4 コア（OCPU）、60 GB の RAM、1.2 Gbps のネットワーク帯域幅、および 3.2 TB の NVMe SSD を備える VM.DenseIO1.4 シェイプが使用されます。ファイル・システムとオペレーティング・システムの他のオーバーヘッドのために、約 10 GB のメモリを残しておくことを推奨します。たとえば、VM.DenseIO1.4 シェイプのデータベースに使用される共有メモリ（PermSize+TempSize+LogBufMB+20 MB）は、50 GB を超えないようにします。直接リンクされているアプリケーションについては、アプリケーションとデータベースの両方の要件を十分に満たすメモリを備えるシェイプの使用を計画する必要があります。

K-safety は 2 で、データ・インスタンス用に NVMe ストレージを備えるシェイプの使用をお勧めします。DenseIO シェイプまたは HighIO シェイプは、ブロック・ストレージのある標準シェイプと比べて高性能です。標準シェイプではブロック・ストレージが必要です。5 台以上の NVMe デバイスがあるシェイプでは、最大 4 台のデバイスのみが構成されます。デバイス数がそれより少ないシェイプでは、すべてのディスクが使用されます。デバイスが 4 台の場合は、RAID 10 のストライプおよびミラー構成である mdraid を選択できます。デフォルトでは、ディスクは RAID 0 構成で、LVM を使用してストライプ化されています。K-safety の値が 2 で、NVMe シェイプを使用する場合は、他方のホストでデータの冗長コピーが保持されているため、ディスクの故障によるデータ損失が回避されます。K-safety の値が 1 の場合、トランザクションはディスクに永続的にコミットしていますが、NVMe ストレージのあるホスト・シェイプがないと、データ損失が発生します。

選択したシェイプは、イメージ起動のための OCI コンポーネントの表にあるように、データベースで利用できるネットワーク帯域幅に影響します。TimesTen Scaleout のワークロードに、参照局所性を考慮していない接続が多数あると、ネットワーク帯域幅が使い尽くされる場合があります。通常は、ネットワークの帯域幅が向上すれば、ピーク時のスループットが改善します。ユーティリティの dstat がデータ・インスタンスにインストールされるため、ネットワーク帯域幅（および他のリソース）を簡単に監視できます。

ZooKeeper と管理インスタンスが独自のコンピュート・インスタンスにオフロードされている場合は、それらのインスタンスのデフォルトのシェイプが VM.Standard.1.1 シェイプとして選択されます。カスタマー・アプリケーションをこれらのコンピュート・ホストと一緒に配置する場合は、スクリプトの変数を設定することで、より大きなシェイプを選択できます。

TimesTen Scaleoutのデプロイメントの準備

Terraform は、コードでインフラストラクチャを管理するためのツールです。インフラストラクチャのコンポーネントのプロビジョニング、変更、バージョニング、および管理に対応しています。Terraform は、構成ファイルに基づいてインフラストラクチャの現在の状態を確認し、実行プランを生成します。そのプランを実行して、ネットワーク・コンポーネントおよびサーバー・ホストをプロビジョニング、削除、または変更し、TimesTen Scaleout をデプロイするのに適した状態を実現できます。Terraform の基本的な情報については、以下のサイトを参照してください。

» 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

ここで説明されているスクリプトは、以下の terraform-provider-oci という Git リポジトリの examples フォルダにある、OCI 向けの他の Terraform のサンプルに沿って使用できます。

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

Ansible は Chef や Puppet のように、システムの管理運用（DevOps）を自動化するのに使用されるツールです。Ansibleのコア・プロジェクトのスクリプトは、オペレーティング・システムとソフトウェアの構成、およびデータベースのインストールに使用されます。OCI Linux サーバーには、提供されているスクリプトによって、Oracle リポジトリから yum を使用して ansible バイナリがインストールされます。Ansible が選択されたのは、外部サーバーやリポジトリへのアクセスが不要なためです。

ソフトウェアのダウンロードとインストールの手順

ここからは、ダウンロードが必要なコンポーネントと、インストールの手順を説明します。サービスをデプロイするためのスクリプトを実行するホストから、ご使用のテナントと、Oracle Cloud のコンパートメントにアクセスできる必要があります。ホストはオンプレミスでも、Oracle Cloud にあるものでもかまいません。スクリプトを実行するには、Oracle Linux 6 か 7 のシステム、または MacOS High Sierra 10.13.4 を実行するシステムで、Python バージョン 2.6 以降を使用する必要があります。以下のコンポーネントが必要です。

» **Oracle Cloud Infrastructure（OCI）向けの Terraform プロバイダ**

» **Terraform バイナリ**

» **OCI 資格証明と OCID**

» **TimesTen Scaleout OCI デプロイメント・スクリプト**

» **TimesTen Scaleout ディストリビューション**

» **JDK/JRE 8 ディストリビューション**

ここからは、各コンポーネントについて詳しく説明します。

» **Oracle Cloud Infrastructure（OCI）向けの Terraform バイナリと Terraform プロバイダ**

OCI 向けの Terraform プロバイダは、Oracle Cloud チームが作成したプラグインで、OCI REST およびネイティブ・コントロール・プレーン・エンドポイントで直接操作します。Terraform バイナリは、クラウド・インフラストラクチャを計画、作成、または破棄するための、OCI プラグインと構成スクリプトを使用するコマンドライン・ユーティリティです。terraform-provider-oci のインストール・ドキュメントでは、OCI 向けの Terraform プロバイダと Terraform バイナリ双方のダウンロードおよびインストール方法が説明されていますが、ダウンロードが必要なのはバイナリだけです。プロバイダは、Terraform の初回実行時にダウンロードされます。

Terraform バイナリは、https://www.terraform.io/downloads.html からダウンロードします。

ドキュメントは https://github.com/oracle/terraform-provider-oci から参照できます。

Terraform バイナリをインストールするには、以下を実行します。

```
% mkdir -p ~/.oci
% unzip terraform_0.11.8_linux_amd64.zip # your version may be different than 0.11.8
% cp terraform ~/.oci
```

以下で説明されているように、terraform init を実行すると、terraform-provider-oci がインストールされます。

» **OCI 資格証明と OCID**

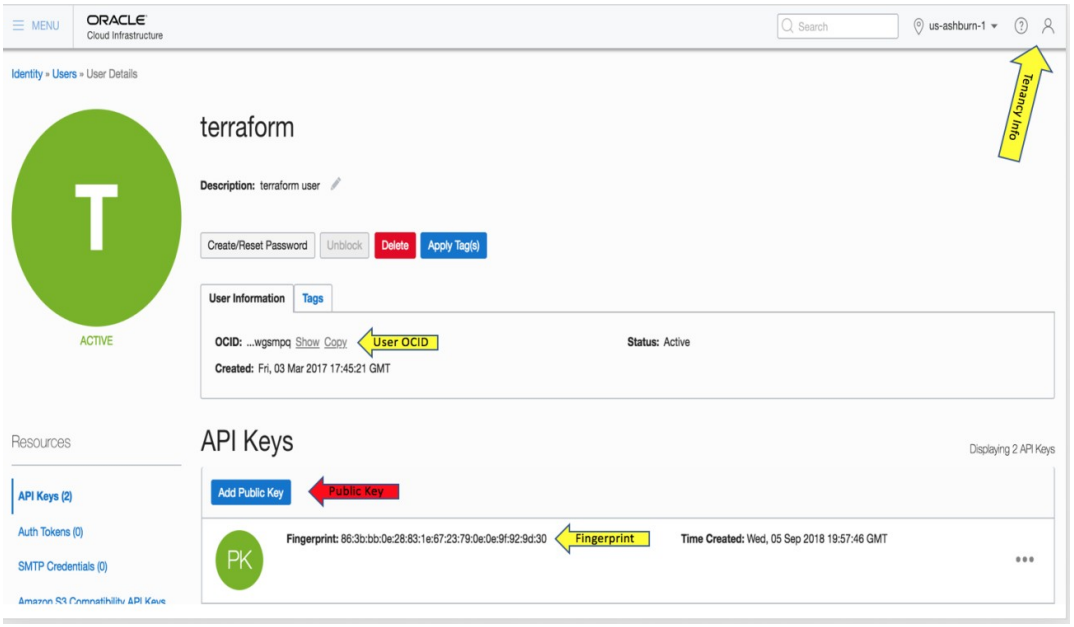
クラウド・インフラストラクチャと TimesTen Scaleout をプロビジョニングするためのスクリプトを実行するユーザーは、PEM 形式で公開鍵をアップロードする必要があります（まだアップロードしていない場合）。詳しい手順については、https://docs.us-phoenix-1.oraclecloud.com/Content/API/Concepts/apisigningkey.htm を参照してください。
必要な手順を簡単に表すと、以下のようになります（上記のリンクは以下のインストラクションより優先されます）。

```
% 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
```

上記で生成された PEM 形式の公開鍵 oci_api_key_public.pem を、OCI にアップロードする必要があります。これは、OCI コンソールから「Identity」→「Users」で実行するか、コンソールの上部中央右寄りにあるユーザー・アカウントのリンクをクリックし、「User Settings」に移動します。下のスクリーン・ショットではユーザー名が"terraform"ですが、この手順はプロビジョニングされ

る OCI リソースのオーナーとなるユーザーで実行する必要があります。赤い矢印のところから、公開鍵をアップロードします。

Oracle Cloud のリソースは、OCID という識別子で表示されます。スクリプトを使用するには、テナントとユーザーの OCID、および鍵のフィンガープリントが必要になります [2]。これらはスクリーン・ショットでは黄色の矢印で示されています。また、リソースが作成されるコンパートメントの OCID も必要です。クラウド管理者から割り当てられたコンパートメントを使用してください。OCID は、OCI コンソールの「Identity」→「Compartments」で確認できます。



OCI のスクリーン・キャプチャ。赤い矢印のところから、公開鍵をアップロードします。黄色の矢印は、ユーザーの OCID、テナントの OCID、鍵のフィンガープリント（アップロード後に使用可能）の場所を表しています。これらは後で必要になります。

» **TimesTen Scaleout BYOL スクリプト**

Terraform および Ansible のスクリプトは、GitHub で入手できる TimesTen の例のサンプルに含まれています。以下のページから oracle-timesten-samples.zip ファイルをダウンロードします。

https://github.com/oracle/oracle-timesten-samples/

ダウンロードしたらサンプルを解凍し、以下のコンポーネントを所定の場所に置きます。

unzip -q oracle-timesten-samples-master.zip
cd oracle-timesten-samples-master/cloud/ottscaleout

最上位のディレクトリの名前は ottscaleout です。

[2] OCI 内のホストからプロビジョニングするユーザーは、Instance Principal Authorization を使用することもできます。

» **TimesTen Scaleout ディストリビューション**

これは Bring Your Own License（BYOL）のソリューションです。TimesTen Scaleout ディストリビューションは、OTN から評価用にダウンロードできます。本番環境で使用する場合は、ARU または eDelivery から、最新のパッチ・リリースをダウンロードしてください。gzipped tar ファイルまたは zip ファイルは、ottscaleout/service/packages ディレクトリに配置します。

» **JDK/JRE 8 ディストリビューション**

TimesTen Scaleout は、JDK8/JRE8 をサポートしています。これは以下のページからダウンロードできます。
http://www.oracle.com/technetwork/java/javase/downloads/jre8-downloads-2133151.html
または
http://www.oracle.com/technetwork/java/javase/downloads/jdk8-downloads-2133151.html
linux-x64.tar.gz ファイルを選択して、ottscalout/service/packages ディレクトリに配置してください。

これで、必要なコンポーネントが集まりました。次に、スクリプトの起動に必要な変数を設定します。

必要な変数の設定

クラウドおよびデータベースのプロビジョニングを制御する変数は、最上位の ottscaleout ディレクトリにある 2 つのファイル、env-vars および variables.tf にあります。env-vars ファイルには、OCI へのアクセスを設定する環境変数があります。TimesTen Scaleout と、必要なデータベース構成に関連する変数は、variables.tf ファイルにあります。インストールに関する前述のセクションで収集した資格証明と OCID を使用して、env-vars ファイルを変更します。

```
# 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… "
```

上記の資格証明によって、OCI リソースにアクセスし、クラウド・リソースをプロビジョニングできます。OCI クラウドのコンピュート・インスタンスからプロビジョニングするユーザーは、Instance Principal Authorization を使用することもできますが、ここではその方法は説明しません。

プロビジョニングされた Bastion ホストへの ssh アクセスには、標準の rsa 形式の資格証明が必要です。以下に例を示します。

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

上記の他に、必要な変数はありません。その他にはすべてデフォルトの値があり、ttimdb1 という名前の NxK（2x2）データベースがプロビジョニングされます。

クラウド・リソースとTimesTen Scaleoutのデプロイ

Terraformの実行

変数が設定されたら、スクリプトを実行します。スクリプトを実行するユーザーは、上記で OCI 資格証明を確立したユーザーである必要があります。

```
% 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
```

Terraform の初回実行時には、インターネットから“oci”、“null_resource”、“template”プロバイダをダウンロードするために、プロキシ・サーバーを使用してローカルのファイアウォールをバイパスする必要がある場合があります。

```
# initialize terraform
% terraform init
Initializing provider plugins...
-   Checking for available provider plugins on
https://releases.hashicorp.com...
-   Downloading plugin for provider "oci" (3.4.0)...
-   Downloading plugin for provider "null" (1.0.0)...
...
```

~/.terraform.d または./.terraform ディレクトリにバージョン 3.4.0 以降 4.0 未満のプロバイダがある場合は、terraform-provider-oci はダウンロードされません。

Terraform がプロビジョニングするリソースを確認する、任意の手順があります。プロビジョニング後は必ずプランを確認して、Terraform で何が作成されるのか、またさらに重要な点として、何が破棄されるのかを理解しておくことをお勧めします。

% terraform plan

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

出力は詳細になるため、ファイルに保存してもよいかもしれません。プランに問題がなければ、クラウド・リソースを破棄します。–auto-approve オプション（ダッシュは 2 つ--）を追加すると、確認プロンプトで'yes'と入力せずに済みます。

% terraform apply --auto-approve

data.oci_identity_availability_domains.ADs:Refreshing state...

…

スピードテストに表示される待機時間が 80 ms のとき、デフォルトの 2x2 のクラウド・インフラストラクチャの割当てには 4 分かかります [3]。ここでも詳細な出力が表示されます。完了時に、Bastion ホストのアドレス（複数の場合あり）が出力されます。

```
Apply complete!Resources:23 added, 0 changed, 0 destroyed.
Outputs:
InstanceIPAddresses = [
    bastion host instances (public addresses):
    ssh opc@129.213.134.5 ,
    database [mgmt|zookeeper] hosts (private addresses):
    ttimdb1-di-001 172.16.48.4
    ttimdb1-di-002 172.16.64.4
    ttimdb1-di-003 172.16.48.2
    ttimdb1-di-004 172.16.64.2
    ,
client host instances (private addresses):
```

Terraform はデフォルトで'*terraform.tfstate*'というファイルを使用し、割り当てられたリソースの状態を追跡します。このファイルは、*terraform* を使って構成を破棄する場合や変更を加える場合に必要です。コピーを作成し、安全な場所に保存することをお勧めします。構成ファイルを使用して、Terraform 経由でリソースを破棄できます。

[3] 待機時間は測定基準の 1 つに過ぎません。操作に要する時間を正確に表すものではありません。

この時点で、クラウド・リソースはプロビジョニングされていますが、システムは構成されておらず、データベースも稼働していません。途中で問題が発生し、その問題を修正できると思われる場合は、Terraform を再度実行して、途中で変更された点を修正し、そこから再開します。

構成を破棄するには、以下を実行します。

% terraform destroy -force

これですべてのリソースが解放されます。途中で失敗した場合や、やり直したい場合も、破棄を実行できます。作成されたものを使用する場合、次の手順で Bastion ホストにユーザーopc として ssh 接続し、ホストと TimesTen Scaleout を構成します。

Ansibleの実行

Terraform で Ansible のスクリプトを実行することは可能ですが、大規模な構成の場合、インストールの完了が Terraform でタイムアウトになる場合があります。代わりに、ssh ストリングを Terraform の出力からカット&ペーストして、Bastion ホストにログインし、スクリプトを実行できます。

% 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] ***

…

プレイブック全体を実行するには、数分かかります（スピードテストの待機時間が 80 ms の場合は 15 分以下）。出力は、特に任意の-v オプションが付いている場合は詳細になります。完了時には、TimesTen Scaleout の構成とアクセス情報が出力に表示されます。複数のコンピュート・イメージのホスト名には、サービス名に基づくプリフィックス、ホストの使用法に応じた 2 文字のインジケータ（zk|mg|di|cl）、3 桁のサフィックスが付きます。たとえば、ttimdb1-di-001 のようになります。

```
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      "
        ]
    }
```

上記の出力では、*ttimdb1-di-001～ttimdb1-di-004* で 2x2 データベースが実行されていることが分かります。このデータベースには instance1～instance 4 の 4 つのインスタンスがあり、オープンで、すぐに使用できる状態です。その後、データベースのアクセス情報が続きます。

```
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"
  ]
}
```

上記の例では、ttimdb1-di-001～ttimdb1-di-004 で実行される管理インスタンスおよびデータ・インスタンスと、それらに接続するためのコマンドが表示されています。途中で問題が発生しないことが望ましいですが、もし問題が発生し、修正可能だと思われる場合は、ansible を再度実行するだけで続行できます。

データベースへのアクセス

Bastion ホストから、構成内の他のどのホストにも ssh 接続できます。多くの場合、データベースを使用するのに最初に必要なのは、管理ユーザーを作成することです。これは、インスタンスの管理ユーザーである oracle として、データ・インスタンスで実行する必要があります。

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

上記の出力を使用すると、*ttimdb1-di-001* のインスタンスは以下の場所に配置されます。

/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.
```

上記のコマンドで、appuser という管理ユーザーが、パスワード appuser で作成されます。ttStatus を実行して、デーモン・ポート（ここでのデフォルトは 46464）を記録しておくこともできます。

$ ~]$ /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
```

後ほど、テナントの外側からコマンド 1 つの呼び出しでプライベート・ネットワーク上のホストにアクセスするには、ssh -J（ジャンプ・プロキシ）オプションを使用します。

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

たとえば、管理インスタンスまたはデータ・インスタンスに直接接続するために、Terraform が実行されていたシステムでこれを実行するとします。ssh コマンドでまず Bastion ホストに接続し、次にターゲットに接続します。ここでは ttimdb1-di-001 です。

これで、アプリケーションを実行する準備がほぼ完了しました。

クライアント接続

OCI接続オプション

アプリケーションをデータベースに接続するには、いくつかの方法があります。TimesTen Scaleoutに直接リンクされたアプリケーションは、データベース自体と同じコンピュート・インスタンスにデプロイできます。デプロイメントのために選択されたシェイプが、アプリケーションとデータベースの両方のリソース（ネットワーク帯域幅、CPU 数、メモリ容量、ディスク容量、およびディスク帯域幅など）に十分に対応していることを確認してください。

OCI では、VCN ピアリング、FastConnect、VPN アクセスなど、クラウド・テナントのデータベースに外部から接続するためのオプションを他にも多数備えています。こうしたオプションにより、テナント外部からの高帯域幅、短い待機時間でのアクセスが可能です。

テナント内部では、指定されたプライベート・サブネットに、複数のクライアント・システムをデプロイできます。パブリック・ネットワークにクライアントをインストールすることは推奨されていません。プライベート・ネットワークにインストールすることで、クライアント・システムのコンパートメントが同じになるため、待機時間が短くなり、高帯域幅でのアクセスが可能になります。管理サーバーや ZooKeeper サーバーが独立した専用のコンピュート・インスタンスで実行されている場合、クライアント・アプリケーションを実行するのに十分なリソースがプロビジョニングされているという前提で、それらのインスタンスがクライアントとして使用される場合があります。

スクリプトを使用して、クライアント向けの専用のコンピュート・インスタンスをデプロイできます。そこでは、TimesTen のクライアント・インストールおよびインスタンスが作成されます。“clInstanceCount”と clInstanceShape”の変数で、クライアント使用向けにプロビジョニングされる数とコンピュート・シェイプを制御します。

variable "clInstanceCount" { default = "2" }

variable "clInstanceShape" { default = "VM.Standard1.1" }

たとえば、上記の変数設定では、プライベート・サブネットの VM.Standard1.1 システムにプロビジョニングし、データベースへの接続がすでに構成されている TimesTen Scaleout クライアント・ソフトウェアをインストールします。管理インスタンスや Zookeeper サーバーが独自のコンピュート・インスタンスにオフロードされている場合は、“clInstanceCount”の変数の設定にかかわらず、クライアント・インストールとインスタンスがそこに作成されます。クライアント・インスタンスがユーザー“oracle”アカウントの<service_name>-client ディレクトリに作成されます。

上記の例のように、Terraform の出力の変数設定で、プロビジョニングされたクライアント・システムが分かります。

client host instances (private addresses):
ttimdb1-cl-001 172.16.48.3
ttimdb1-cl-002 172.16.64.3

プロビジョニングされたクライアントを使用する、またはオフロードされた管理インスタンスや Zookeeper サーバー上のクライアントを使用するには、ユーザーoracle として、適切なコンピュート・インスタンスに ssh 接続します。前述のとおり、appuser というユーザーが作成されているものとします。

```
ssh -tt ttimdb1-cl-001 sudo su – oracle
. ttimdb1-client/bin/ttenv.sh
ttisqlcs dsn="ttimdb1cs;uid=appuser;pwd=appuser" [4]
…
connect "dsn=ttimdb2";
Connection successful:DSN=ttimdb2;
… Command>
```

これで、クライアント・インスタンスが接続されました。クライアント・アクセス向けのコンピュート・インスタンスを手動で構成する場合は、TimesTen Scaleoutのドキュメントを参照してください。

構成のカスタマイズ

初期のプロビジョニング設定および構成を制御できる変数は多数あります。変数は variables.tf ファイルにあります。こうした変数については、次の項で説明します。

Oracle Linuxオペレーティング・システムのイメージ

プロビジョニングされたすべてのコンピュート・インスタンスに、同じ Oracle Linux イメージがインストールされます。本書の執筆時点では、Oracle Linux 7.5 イメージです。インストールされるイメージを設定するには、variables.tfの"InstanceImageOCID"マップを更新します。使用できるイメージについては、https://docs.us-phoenix-1.oraclecloud.com/images/を参照してください。

variables.tf の *securityupdates* 変数が *true* の**場合**、インストール時に yum がイメージのすべてのパッケージ（セキュリティの正誤表があるもの）を更新します。デフォルトでは、イメージは更新***されません***。cronjob では、CVE ごとに利用できるセキュリティ更新がリスト化され、*/home/opc/latest-cves* ファイルに出力が書き込まれます。プロビジョニング時にセキュリティ更新を適用する場合は、より新しいイメージを選択すると、適用が必要になる正誤表が少なくなることが多いため、より短時間で開始できます。プロビジョニングされたホストを管理するために必要な更新を適用するかどうかは、ユーザーに委ねられています。

可用性ドメイン

デフォルトでは、2x2 構成のデータ・コンピュート・インスタンスは、AD-1 および AD-2 の可用性ドメインに作成されます。プロビジョニングで AD-2 および AD-3 を使用するように変更するには、variables.tf の環境変数を設定します。

variable" initialAD" { default = "2"}

デフォルトの設定は initialAD=1 です。initialAD=3 を設定すると、AD-3 と AD-1 にプロビジョニングされます。この変数では、管理インスタンスが独自の VM にオフロードされている場合に、それらがプロビジョニングされる AD も制御します。ZooKeeper サーバーがオフロードされている場合は、それらの VM は 3 つすべての可用性ドメインに分散されます。

[4] 当然のことですが、コマンドラインでパスワードを指定するのは最善策ではありません。

データおよび管理コンピュート・インスタンスを単一の可用性ドメインに割り当てるには、以下のように設定します。

variable "singleAD" { default="true" }.

使用される AD は上記の"initialAD"によって指定されます。

K-Safety

variables.tf の *ksafety* の値をそれぞれ 1 または 2 に設定することで、1 つまたは複数のデータのコピーを作成するように TimesTen Scaleout を構成できます。デフォルトでは、ksafety は 2 に設定されており、データのコピーが複数作成されます。

```
# The K in NxK
variable "ksafety" { default = "2" }
```

ksafety==1 で、データベース属性の Durability が設定されて Durability=1 となっている場合、トランザクション・マネージャが永続的に、コミットの準備ができたすべてのレコードを書き込みます。

データ・インスタンス

前述の可用性ドメインの項で説明されているように、singleAD==true である場合を除き、可用性を高めるために、データベースのレプリカ・セットは可用性ドメイン全体に分散されています。したがって通常は、TimesTen Scaleout のデータ領域は、1 つが AD-1 に、もう 1 つが AD-2 にあります。データ・インスタンスの数とシェイプは、variables.tf の環境変数で制御されます。

```
# the N in NxK
variable "diInstanceCount" { default = "2" }

# Compute instance shape for data instances
# N*K VMs/BMs are provisioned for data instances.
# Recommended to use NVMe shape(DenseIO or HighIO) for best performance
variable "diInstanceShape" { default = "VM.DenseIO1.4"}
```

データ・インスタンスは VM.Standard2.4 などの標準シェイプを使用することもできますが、標準シェイプを使用する場合は、ブロック・ボリュームも使用する必要があります。下記のブロック・ボリュームの項を参照してください。

データベースの名前は、env_vars で以下のように設定されます。

export TF_VAR_service_name="ttimdb1"

この変数は、VCN 名とディレクトリ構造も制御します。たとえばこの場合、インストールおよびインスタンス・ディレクトリは、ttimdb1 という名前の最上位ディレクトリのサブディレクトリです。名前を変更すると、もう 1 つのリソース・セットが別の VCN とデータベースで有効になり、同じコンパートメントでプロビジョニングされます。

*PermSize*および *DatabaseCharacterSet*などの他のデータベース属性は、*variables.tf*ファイルにある"*timesten*"マップを使用して変更できます。

データベースは、ユーザー*oracle* で実行されます。つまり、ユーザー*oracle* はデータベースのインスタンス管理者です。ユーザー*oracle* にパスワードはありません。*oracle* アカウントにアクセスするには、*opc* ユーザー・アカウントから *sudo* を使用します。

sudo su – oracle

*oracle*としてログインすると、*ssh*を使用して他のホストの *oracle*アカウントにアクセスできます。*opc* アカウントからユーザーに *sudo* を使用させたくない場合は、*/home/oracle/.ssh* の ssh キーをユーザーにエクスポートできます。

TimesTen Scaleout のインストールの場所は、デフォルトでは */home/oracle/<service_name>/<version>;/home/oracle/ttimdb1/tt18.1.1.2.0* です。データ・インスタンスのディレクトリ、データベース・ファイル、トランザクション・ログ・ファイルの場所は、デフォルトでは*/<fsname>/TimesTen/<service-name>, /u10/TimesTen/ttimdb1* です。ファイル・システム名の"*fsname*"は、*variables.tf* で変更できます。

"fsname"（*デフォルトで/u10*）ファイル・システムは、NVMe またはブロック・ボリュームのデバイスのマウント・ポイントです。前述の計画に関するセクションで取り上げられたように、このマウント・ポイントのもとでは、4 台以下の NVMe デバイスまたは 1 つのブロック・ボリュームが構成されます。複数のデバイスがある場合は、デフォルトでは LVM を使用してストライプ化されます。LVM によるストライプ化ではなく、NVMe デバイスで *mdraid* を使用して RAID 10 のストライプ化とミラー化を行うには、*variables.tf*の変数を *"storage"="MD-RAID-10"*に設定します。mdraid デバイスでファイル・システムを作成するにはかなりの時間がかかります。

ブロック・ボリューム

ブロック・ボリュームは標準シェイプで使用できます。各データ・コンピュート・インスタンスに付き、1 つのブロック・ボリュームがプロビジョニングされます。ブロック・ボリュームは、必要なストレージを GB 単位のサイズで指定することで構成します。ブロック・ボリュームには、2 つのチェックポイント・ファイル（2xPermSize）、トランザクション・ログ・ファイルが含まれ、さらにバックアップ用に一時使用されるため、メモリ・サイズの少なくとも 3 倍のサイズを使用することをお勧めします。ブロック・ボリュームにリポジトリを含める場合は、サイジングでそのストレージを考慮してください。ブロック・ボリュームをプロビジョニングするには、variables.tf で以下のように設定します。

```
# Minimum allocation is 50 GB
variable "diBlockVolumeSizeGB" { default = "50" }
```

上記の設定では、50 GB のブロック・ボリュームが設定されています。50 未満に設定すると、ブロック・ボリュームがプロビジョニングされなくなります。

管理インスタンス

デフォルトでは、管理インスタンスはデータ・インスタンスと同じ VM に配置できます。2 つの管理インスタンスが作成され、データベース・レプリカ・セットと同じように、複数の可用性ドメインに分散されます。管理インスタンスは、*variables.tf*の変数を *"mgInstanceCount" { default = "2"}* と設定することで、独自の VM にオフロードできます。オフロードされた管理インスタンスは、*VM.Standard.1.1* シェイプを使用します。*variables.tf* でシェイプも変更できます。各管理インスタンスでは、1 GB 未満の RAM、約 2 GB のストレージが必要であり、CPU やネットワーク帯域幅の使用はごくわずかです。独自の VM にオフロードされた管理インスタンスには、前述のクライアント接続の項で説明されているように、クライアント・インスタンスが含まれます。

ZooKeeperサーバー

TimesTen Scaleout は、メンバーシップ・サービスとして ZooKeeper を使用します。デフォルトでは、3 台の ZooKeeper サーバーの実行が設定され、データ・インスタンスと一緒に配置されます。データ・インスタンスが 3 つ以上ある場合は、そのうちの 3 つが ZooKeeper サーバーを実行します。別個に管理インスタンスがある場合は、ZooKeeper サーバーの 2 台が管理インスタンスのホストで実行され、3 台目がデータ・コンピュート・インスタンスで実行されます。ZooKeeper サーバーは、以下のように設定することで独自の VM にオフロードできます。

variable "zkInstanceCount" { default = "3" }

この場合、3 台の ZooKeeper サーバーが 3 つすべての可用性ドメインに分散されています。オフロードされたサーバーのデフォルトのシェイプは、*VM.Standard.1.1* シェイプです。ZooKeeper サーバーでは 1 GB 以下のメモリが必要ですが、CPU やネットワーク帯域幅はほとんど使われません。独自の VM にオフロードされた ZooKeeper サーバーには、前述のクライアント接続の項で説明されているように、クライアント・インスタンスが含まれます。

Bastionホスト

Bastion ホストにはインターネット・ファイアウォールの役割があり、残りの構成への ssh ゲートウェイとして使用されます。OCI では、Bastion ホストの使用がベスト・プラクティスであると考えられます。詳しくは、『Bastion Hosts: Protected Access for Virtual Cloud Networks』を参照してください。デフォルトでは、Bastion ホストが 1 台のみプロビジョニングされますが、variables.tf の bsInstanceCount 変数の値を増やすことで、OCI のホワイト・ペーパー『NAT Instance Configuration: Enabling Internet Access for Private Subnets』で説明されている高可用性構成の実装が可能です。ホワイト・ペーパーで説明されている H/A 構成のスクリプトは、Bastion ホストにデプロイされるのではありませんが、ホワイト・ペーパーからコピーを展開したり、service/scripts ディレクトリで探したりできます。Bastion ホストの数とそのシェイプは、*variables.tf* ファイルで構成されます。

構成の管理

Systemd

ZooKeeper サーバー、管理インスタンス、データ・インスタンスは、*systemd* のもとでサービスとして実行されます。こうしたコンピュート・インスタンスのいずれかがダウンすると、*systemd* が再起動しようとします。以下のコマンドのいずれかを opc ユーザー・アカウントから実行すると、サービスのステータスを全般的に確認できます。

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

ZooKeeper

ZooKeeper の *systemd* サービスは、*<service-name>-zk1.service* という名前です。トラブルシューティングのために、コンピュート・インスタンスには *ncat* がインストールされ、'*stat*'のような 4 文字のコマンドを ZooKeeper に送信できるようになっています。

$ 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
```

管理インスタンス

管理インスタンス、とりわけアクティブな管理インスタンスとは、データベースが管理される場所です。TimesTen Scaleout での管理インスタンスの操作は、*ttgridadmin* ユーティリティによって実行されます。管理インスタンスを監視するために、*ttgridadmin mgmtexamine* が定期的に実行されることになっています。コンピュート・インスタンスが利用できなくなった場合、*mgmtexamine* の出力に、インスタンスをフェイルオーバーまたは再起動するための推奨措置が表示されます。*systemd* にはスクリプトがインストールされ、管理インスタンスの状態が定期的にチェックされます。また、インスタンスがダウンした場合には、*ttgridadmin mgmtexamine* から推奨措置が実行されます。管理データベースでは、*systemd* が、*'/home/oracle/bin/mgmtexamine.py'* という名前のスクリプトを実行することで、管理データベースの状態を定期的に確認するサービスを実行します。サービス・ファイルの名前は、*<service-name>-mgmt.service* および *<service-name>-mgmt.timer* です。提供されているスクリプトを使用せずに、管理インスタンスが破棄された場合、または別のコンピュート・ホストに移動した場合は、対応するサービスを必ずシャットダウンしてください。

データ・インスタンス

データベース・サービス・ファイルの名前は*<host>.service* です（例：*ttimdb1-di-001.service*）。データ・インスタンスはリブート時または異常な終了時に再起動されます。つまり、リターン・コードが 0 ではなく、SIGHUP、SIGINT、SIGTERM、SIGPIPE のいずれのシグナル経由でもない場合です。

停止

Ansible スクリプトで、データベース、管理サーバー、ZooKeeper サーバー、および対応する*systemd*サービスを、この順番で停止できます。

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

停止操作は、以下のコマンドを実行することでデータベースを終了し、メモリからアンロードします。

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

*<time-in-seconds>*パラメータを空の文字列にして、完了まで待機することができます。デフォルトでは、停止時間は 5 分に設定されていますが、*variables.tf* またはコマンドラインで上記のように構成できます。Ansibleスクリプトは、タイムアウトが発生すると機能しません。データベースをアンロードする前に、データベースへのすべてのクライアント・サーバー接続を終了する必要があります。

データベースがアンロードされた後に、デーモンとサービスが停止されます。次に、'*ttgridadmin mgmtstandbystop*'と'*mgmtactivestop*'で管理サービスが停止されます。最後に、ZooKeeper の'*zkServer.sh* stop'スクリプトが実行されて、ZooKeeper サーバーが停止します。

このスクリプトでは、使用されているコンピュート・リソースは停止されず、OCI コンソールを通して停止する必要があります。再起動スクリプトは現在のところ提供されていません。再起動する場合は、*systemd* を使用して、まず ZooKeeper サーバー、管理インスタンス、次いでデータ・インスタンスの順で再起動します。

スケールアウト

使用可能なデータ・インスタンスの変数（NxK の N）を増やすことで、***データ・インスタンス***数を限定された形でスケールアウトできます。たとえば、1x2 構成で始めた場合、2x2 構成にするには、N を 2 に増やします。

variables.tf を編集して、"diInstanceCount" { default = "2" }を設定する

Terraform プランに、2 つのデータ・インスタンスが追加されたことが表示されます。

% 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.
…
```

プランに問題がなければ、データ・インスタンスを作成します。–auto-approve オプションにはダッシュが 2 つ必要です。

```
% 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 がデータ・インスタンスをプロビジョニングし、Ansible ホスト・ファイルを更新して、そのファイルを Bastion ホストにコピーしました。この構成にインスタンスを追加するには、Bastion ホストにログインして、以下の操作を実行します。Ansible によってホスト、インストール、およびインスタンスが追加されます。次に dbDistribute が実行されて、要素が入力されます。スケールアウトに要する時間は、新しい要素への移動が必要なデータの量によって左右されます。

```
% ssh 129…
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 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 ttimdb1-di-004 instance2 opened 2018-06-20 17:39:16 "

インフラストラクチャのスケーリングには制限があります。2x2 を 1x2 構成にするというスケールインは、現在はスクリプトではサポートされていません。また、データ・インスタンスのシェイプを変更することで"スケールアップ"または"スケールダウン"することはできません。そうした操作を実行すると、Terraform が既存のデータ・インスタンスを破棄し、新しいものをプロビジョニングします。こうした操作を実行するための Ansible スクリプトが存在しないため、データベースは一貫性に欠けた状態になります。加えて、管理インスタンスや ZooKeeper サーバーの数のスケーリングは、スクリプトではサポートされていません。

データベースの再作成

データベースを破棄して再作成したいが、クラウド・リソースやインフラストラクチャの再プロビジョニングは不要という場合があるかもしれません。その場合は、Bastion ホストで *ansible* コマンドを実行します。

```
[opc@ttimdb1-bs-001 ~]$ ansible-playbook -i hosts dbdestroy.yml
[opc@ttimdb1-bs-001 ~]$ ansible-playbook -i hosts datainstances.yml mgmtinstances.yml status.yml
```

status.yml スクリプトは、単体でいつでも実行できます。Zookeeper サーバーをチェックする'*echo stat | nc …*'が実行され、次いで'*ttgridadmin dbstatus ttimdb1 -element* 'が実行されます。*ttGridRollout* からの事前の出力も表示されます。

クライアント

クライアントは、最初のロールアウトが終わり、すべてがセットアップされた後に、プロビジョニングされます。クライアントを追加するには、*variables.tf* の"*clInstanceCount*"の値を増やします。次に、Terraform と Ansible を実行して、追加のクライアントをプロビジョニングします。最初に

```
% terraform plan
```

を実行して、重要なものが破棄されないことを確かめます。Ansible ホスト・ファイルが破棄されてから再作成され、新しくプロビジョニングされるクライアントが含められます。問題がなければ、以下のようにします。

```
% terraform apply –auto-approve # (two dashes before auto-approve)
[opc@ttimdb1-bs-001 ~]$ ssh opc@129....
[opc@ttimdb1-bs-001 ~]$ cd service/ansible
[opc@ttimdb1-bs-001 ~]$ ansible-playbook -i hosts client.yml
```

将来的な作業

本日使用したスクリプトは、主にサービスをプロビジョニングするものです。これらのスクリプトは、TimesTen Scaleoutの可用性や管理を改善するために進化を遂げていく予定です。今後の作業により、ライフ・サイクル管理が改善され、OCI の機能をさらに活用し、可用性を向上させられるものと思われます。

ORACLE

Oracle Corporation, World Headquarters
500 Oracle Parkway
Redwood Shores, CA 94065, USA

海外からのお問い合わせ窓口
電話：+1.650.506.7000
ファクシミリ：+1.650.506.7200

Oracle Cloud Infrastructure での Oracle TimesTen Scaleout のデプロイ
2018 年 4 月
著者：steve.folkman@oracle.com
共著者：