DB自動化ツールキット

NetApp Solutions

NetApp
April 10, 2024

目次

DB自動化ツールキット

Oracleの自動移行

NetAppソリューションエンジニアリングチーム

目的

FSx ONTAPストレージとEC2コンピューティングインスタンスをターゲットインフラとして使用し、オンプレミスからAWSクラウドへのOracleデータベースの移行を自動化するツールキットです。CDB / PDBモデルでは、お客様がオンプレミスのOracleデータベースをすでに導入していることを前提としています。このツールキットを使用すると、最大可用性オプションを指定してOracle PDB再配置手順を使用して、Oracleホスト上のコンテナデータベースから名前付きPDBを再配置できます。つまり、オンプレミスのストレージアレイ上のソースPDBは、サービスの中断を最小限に抑えて新しいコンテナデータベースに再配置されます。Oracle再配置手順は'データベースがオンラインのときにOracleデータ・ファイルを移動しますその後、すべてのデータファイルがAWSクラウドに移行されたときに、スイッチオーバー時にオンプレミスから再配置されたデータベースサービスにユーザセッションを再ルーティングします。下線のテクノロジは、実証済みのOracle PDBホットクローン手法です。

> 移行ツールキットは、AWSクラウドインフラで開発、検証されていますが、Oracleアプリケーションレベルのソリューションを基盤としています。そのため、このツールキットはAzureやGCPなどの他のパブリッククラウドプラットフォームにも適用できます。

この解決策 は、次のユースケースに対応します。

- 移行ユーザを作成し、必要な権限をオンプレミスのソースDBサーバに付与
- オンプレミスのCDBからクラウド内のターゲットCDBにPDBを再配置し、切り替えまでソースPDBをオンラインにします。

対象者

この解決策 は、次のユーザーを対象としています。

- オンプレミスからAWSクラウドにOracleデータベースを移行するDBA。
- オンプレミスからAWSクラウドへのOracleデータベースの移行に関心があるデータベース解決策アーキテクト。
- OracleデータベースをサポートするAWS FSx ONTAPストレージを管理するストレージ管理者。
- オンプレミスからAWSクラウドにOracleデータベースを移行したいと考えているアプリケーション所有者。

使用許諾

このGitHubリポジトリのコンテンツにアクセス、ダウンロード、インストール、または使用することにより、 "ライセンスファイル"。

このGitHubリポジトリ内のコンテンツとの派生物の作成および/または共有には、一定の制限があります。コンテンツを使用する前に、必ずライセンスの条件をお読みください。すべての条件に同意しない場合は、このリポジトリ内のコンテンツにアクセス、ダウンロード、または使用しないでください。

解決策 の導入

導入の前提条件

導入には、次の前提条件が必要です。

```
Ansible v.2.10 and higher
ONTAP collection 21.19.1
Python 3
Python libraries:
  netapp-lib
  xmltodict
  jmespath
```

```
Source Oracle CDB with PDBs on-premises
Target Oracle CDB in AWS hosted on FSx and EC2 instance
Source and target CDB on same version and with same options installed
```

```
Network connectivity
  Ansible controller to source CDB
  Ansible controller to target CDB
  Source CDB to target CDB on Oracle listener port (typical 1521)
```

ツールキットをダウンロード

```
git clone https://github.com/NetApp/na_ora_aws_migration.git
```

ホスト変数の設定

ホスト変数は、｛｛host_name｝｝.ymlという名前のhost_varsディレクトリに定義されています。一般的な設定を示すために、ホスト変数ファイルhost_name.ymlの例が含まれています。主な考慮事項は次のとおりです。

```
Source Oracle CDB - define host specific variables for the on-prem CDB
  ansible_host: IP address of source database server host
  source_oracle_sid: source Oracle CDB instance ID
  source_pdb_name: source PDB name to migrate to cloud
  source_file_directory: file directory of source PDB data files
  target_file_directory: file directory of migrated PDB data files
```

```
Target Oracle CDB - define host specific variables for the target CDB
including some variables for on-prem CDB
  ansible_host: IP address of target database server host
  target_oracle_sid: target Oracle CDB instance ID
  target_pdb_name: target PDB name to be migrated to cloud (for max
availability option, the source and target PDB name must be the same)
  source_oracle_sid: source Oracle CDB instance ID
  source_pdb_name: source PDB name to be migrated to cloud
  source_port: source Oracle CDB listener port
  source_oracle_domain: source Oracle database domain name
  source_file_directory: file directory of source PDB data files
  target_file_directory: file directory of migrated PDB data files
```

DBサーバのホストファイル構成

AWS EC2インスタンスは、デフォルトでホスト名にIPアドレスを使用します。Ansibleのhostsファイルに異なる名前を使用する場合は、ソースサーバとターゲットサーバの両方について、/etc/hostsファイルにホストの名前解決を設定します。次に例を示します。

```
127.0.0.1   localhost localhost.localdomain localhost4
localhost4.localdomain4
::1         localhost localhost.localdomain localhost6
localhost6.localdomain6
172.30.15.96 source_db_server
172.30.15.107 target_db_server
```

Playbookの実行**-**順番に実行

1. Ansibleコントローラの前提条件をインストールする。

```
ansible-playbook -i hosts requirements.yml
```

```
ansible-galaxy collection install -r collections/requirements.yml
--force
```

2. オンプレミスサーバに対して移行前のタスクを実行（adminがsshユーザで、sudo権限でオンプレミスのOracleホストに接続する場合）

```
ansible-playbook -i hosts ora_pdb_relocate.yml -u admin -k -K -t
ora_pdb_relo_onprem
```

3. オンプレミスCDBからAWS EC2インスタンスのターゲットCDBへのOracle PDB再配置を実行します（EC2 DBインスタンス接続にはEC2-USER、EC2-USER sshキーペアを使用するdb1.pemを想定）。

```
ansible-playbook -i hosts ora_pdb_relocate.yml -u ec2-user --private
-key db1.pem -t ora_pdb_relo_primary
```

追加情報の参照先

NetApp 解決策 自動化の詳細については、次のWebサイトを参照してください。 "NetApp 解決策の自動化"

AWS FSx ONTAPでのOracle HA / DRの自動化

NetAppソリューションエンジニアリングチーム

目的

このツールキットを使用すると、AWSクラウドに導入されたOracleデータベース（FSx for ONTAPストレージとEC2コンピューティングインスタンス）に対して、高可用性とディザスタリカバリ（HR / DR）環境のセットアップと管理のタスクを自動化できます。

この解決策 は、次のユースケースに対応します。

- ソースサーバホストと一致するように、HA/DRターゲットホスト-カーネル構成とOracle構成をセットアップします。
- FSx ONTAP -クラスタピアリング、SVMピアリング、OracleボリュームSnapMirror関係をソースからターゲットに設定します。
- スナップショットによるOracleデータベースデータのバックアップ- crontabの実行

- SnapshotによるOracleデータベースアーカイブログのバックアップ- execute off crontab
- HA / DRホストでフェイルオーバーとリカバリを実行- HA / DR環境をテストして検証
- フェイルオーバーテスト後に再同期を実行-データベースボリュームの再確立HA / DRモードでのSnapMirror関係

対象者

この解決策 は、次のユーザーを対象としています。

- 高可用性、データ保護、ディザスタリカバリを実現するためにAWSにOracleデータベースをセットアップするDBAです。
- AWSクラウドのストレージレベルのOracle HA / DR解決策に関心をお持ちのデータベース解決策アーキテクト。
- OracleデータベースをサポートするAWS FSx ONTAPストレージを管理するストレージ管理者。
- AWS FSX/EC2環境でHA / DR用にOracleデータベースを立ち上げたいと考えているアプリケーション所有者。

使用許諾

このGitHubリポジトリのコンテンツにアクセス、ダウンロード、インストール、または使用することにより、 "ライセンスファイル"。

> このGitHubリポジトリ内のコンテンツとの派生物の作成および/または共有には、一定の制限があります。コンテンツを使用する前に、必ずライセンスの条件をお読みください。すべての条件に同意しない場合は、このリポジトリ内のコンテンツにアクセス、ダウンロード、または使用しないでください。

解決策 の導入

導入の前提条件

導入には、次の前提条件が必要です。

```
Ansible v.2.10 and higher
ONTAP collection 21.19.1
Python 3
Python libraries:
  netapp-lib
  xmltodict
  jmespath
```

```
AWS FSx storage as is available
```

```
AWS EC2 Instance
  RHEL 7/8, Oracle Linux 7/8
  Network interfaces for NFS, public (internet) and optional management
  Existing Oracle environment on source, and the equivalent Linux
operating system at the target
```

ツールキットをダウンロード

```
git clone https://github.com/NetApp/na_ora_hadr_failover_resync.git
```

グローバル変数の設定

Ansibleのプレイブックは可変式です。一般的な構成を示すために、グローバル変数ファイルfsx_vars_example.ymlの例が含まれています。主な考慮事項は次のとおりです。

```
ONTAP - retrieve FSx storage parameters using AWS FSx console for both
source and target FSx clusters.
  cluster name: source/destination
  cluster management IP: source/destination
  inter-cluster IP: source/destination
  vserver name: source/destination
  vserver management IP: source/destination
  NFS lifs: source/destination
  cluster credentials: fsxadmin and vsadmin pwd to be updated in
roles/ontap_setup/defaults/main.yml file
```

```
Oracle database volumes - they should have been created from AWS FSx
console, volume naming should follow strictly with following standard:
  Oracle binary: {{ host_name }}_bin, generally one lun/volume
  Oracle data: {{ host_name }}_data, can be multiple luns/volume, add
additional line for each additional lun/volume in variable such as {{
host_name }}_data_01, {{ host_name }}_data_02 ...
  Oracle log: {{ host_name }}_log, can be multiple luns/volume, add
additional line for each additional lun/volume in variable such as {{
host_name }}_log_01, {{ host_name }}_log_02 ...
  host_name: as defined in hosts file in root directory, the code is
written to be specifically matched up with host name defined in host
file.
```

```
Linux and DB specific global variables - keep it as is.
  Enter redhat subscription if you have one, otherwise leave it black.
```

ホスト変数の設定

ホスト変数は、｛｛host_name｝｝.ymlという名前のhost_varsディレクトリに定義されています。一般的な設定を示すために、ホスト変数ファイルhost_name.ymlの例が含まれています。主な考慮事項は次のとおりです。

```
Oracle - define host specific variables when deploying Oracle in
multiple hosts concurrently
  ansible_host: IP address of database server host
  log_archive_mode: enable archive log archiving (true) or not (false)
  oracle_sid: Oracle instance identifier
  pdb: Oracle in a container configuration, name pdb_name string and
number of pdbs (Oracle allows 3 pdbs free of multitenant license fee)
  listener_port: Oracle listener port, default 1521
  memory_limit: set Oracle SGA size, normally up to 75% RAM
  host_datastores_nfs: combining of all Oracle volumes (binary, data,
and log) as defined in global vars file. If multi luns/volumes, keep
exactly the same number of luns/volumes in host_var file
```

```
Linux - define host specific variables at Linux level
  hugepages_nr: set hugepage for large DB with large SGA for
performance
  swap_blocks: add swap space to EC2 instance. If swap exist, it will
be ignored.
```

DBサーバのホストファイル構成

AWS EC2インスタンスは、デフォルトでホスト名にIPアドレスを使用します。Ansibleのhostsファイルに異なる名前を使用する場合は、ソースサーバとターゲットサーバの両方について、/etc/hostsファイルにホストの名前解決を設定します。次に例を示します。

```
127.0.0.1   localhost localhost.localdomain localhost4
localhost4.localdomain4
::1         localhost localhost.localdomain localhost6
localhost6.localdomain6
172.30.15.96 db1
172.30.15.107 db2
```

Playbookの実行-順番に実行

1. Ansibleコントローラの前提条件をインストールします。

```
ansible-playbook -i hosts requirements.yml
```

```
ansible-galaxy collection install -r collections/requirements.yml
--force
```

2. ターゲットEC2 DBインスタンスをセットアップします。

```
ansible-playbook -i hosts ora_dr_setup.yml -u ec2-user --private-key
db2.pem -e @vars/fsx_vars.yml
```

3. ソースデータベースボリュームとターゲットデータベースボリューム間にFSx ONTAP SnapMirror関係を設定します。

```
ansible-playbook -i hosts ontap_setup.yml -u ec2-user --private-key
db2.pem -e @vars/fsx_vars.yml
```

4. crontabのスナップショットを使用して、Oracleデータベースのデータボリュームをバックアップします。

```
10 * * * * cd /home/admin/na_ora_hadr_failover_resync &&
/usr/bin/ansible-playbook -i hosts ora_replication_cg.yml -u ec2-
user --private-key db1.pem -e @vars/fsx_vars.yml >>
logs/snap_data_`date +"%Y-%m%d-%H%M%S"`.log 2>&1
```

5. crontabのSnapshotを使用して、Oracleデータベースのアーカイブログボリュームをバックアップします。

```
0,20,30,40,50 * * * * cd /home/admin/na_ora_hadr_failover_resync &&
/usr/bin/ansible-playbook -i hosts ora_replication_logs.yml -u ec2-
user --private-key db1.pem -e @vars/fsx_vars.yml >>
logs/snap_log_`date +"%Y-%m%d-%H%M%S"`.log 2>&1
```

6. フェイルオーバーを実行し、ターゲットEC2 DBインスタンスでOracleデータベースをリカバリします。テストを行い、HA / DR構成を検証します。

```
ansible-playbook -i hosts ora_recovery.yml -u ec2-user --private-key
db2.pem -e @vars/fsx_vars.yml
```

7. フェイルオーバーテスト後に再同期を実行し、レプリケーションモードでデータベースボリュームのSnapMirror関係を再確立します。

```
ansible-playbook -i hosts ontap_ora_resync.yml -u ec2-user --private
-key db2.pem -e @vars/fsx_vars.yml
```

追加情報の参照先

NetApp 解決策 自動化の詳細については、次のWebサイトを参照してください。 "NetApp 解決策の自動化"

AWS FSx ONTAPクラスタとEC2インスタンスのプロビジョニング

NetAppソリューションエンジニアリングチーム

目的

このツールキットを使用すると、AWS FSx ONTAPストレージクラスタとEC2コンピューティングインスタンスのプロビジョニングタスクを自動化できます。EC2コンピューティングインスタンスは、あとからデータベースの導入に使用できます。

この解決策 は、次のユースケースに対応します。

- 事前定義されたVPCサブネット内のAWSクラウドでEC2コンピューティングインスタンスをプロビジョニングし、EC2インスタンスアクセス用のSSHキーをEC2-userとして設定します。
- 必要なアベイラビリティゾーンにAWS FSx ONTAPストレージクラスタをプロビジョニングし、ストレージSVMを設定し、クラスタ管理ユーザfsxadminパスワードを設定します。

対象者

この解決策 は、次のユーザーを対象としています。

- AWS EC2環境でデータベースを管理するDBA。
- AWS EC2エコシステムへのデータベース導入に関心のあるデータベース解決策アーキテクト。
- データベースをサポートするAWS FSx ONTAPストレージを管理するストレージ管理者。
- AWS EC2エコシステムでデータベースを立ち上げたいと考えているアプリケーション所有者。

使用許諾

このGitHubリポジトリのコンテンツにアクセス、ダウンロード、インストール、または使用することにより、 "ライセンスファイル"。

このGitHubリポジトリ内のコンテンツとの派生物の作成および/または共有には、一定の制限があります。コンテンツを使用する前に、必ずライセンスの条件をお読みください。すべての条件に同意しない場合は、このリポジトリ内のコンテンツにアクセス、ダウンロード、または使用しないでください。

解決策 の導入

導入の前提条件

導入には、次の前提条件が必要です。

```
An Organization and AWS account has been setup in AWS public cloud
  An user to run the deployment has been created
  IAM roles has been configured
  IAM roles granted to user to permit provisioning the resources
```

```
VPC and security configuration
  A VPC has been created to host the resources to be provisioned
  A security group has been configured for the VPC
  A ssh key pair has been created for EC2 instance access
```

```
Network configuration
  Subnets has been created for VPC with network segments assigned
  Route tables and network ACL configured
  NAT gateways or internet gateways configured for internet access
```

ツールキットをダウンロード

```
git clone https://github.com/NetApp/na_aws_fsx_ec2_deploy.git
```

接続と認証

このツールキットはAWSクラウドシェルから実行されることになっています。AWSクラウドシェルは、AWSリソースの安全な管理、探索、操作を容易にするブラウザベースのシェルです。CloudShellは、コンソールのクレデンシャルで事前に認証されます。一般的な開発ツールと運用ツールが事前にインストールされているため、ローカルでのインストールや設定は必要ありません。

Terraformプロバイダの.tfファイルとmain.tfファイルの構成

provider.tfは、TerraformがAPI呼び出しを介してリソースをプロビジョニングするプロバイダを定義します。main.tfは、プロビジョニングされるリソースのリソースと属性を定義します。以下に詳細を示します。

```
provider.tf:
  terraform {
    required_providers {
      aws = {
        source  = "hashicorp/aws"
        version = "~> 4.54.0"
      }
    }
  }
```

```
main.tf:
  resource "aws_instance" "ora_01" {
    ami                           = var.ami
    instance_type                 = var.instance_type
    subnet_id                     = var.subnet_id
    key_name                      = var.ssh_key_name
    root_block_device {
      volume_type                 = "gp3"
      volume_size                 = var.root_volume_size
    }
    tags = {
      Name                        = var.ec2_tag
    }
  }
  ....
```

Terraform変数.tfおよびterraform.tfvarsの設定

variables.tfは、main.tfで使用する変数を宣言します。terraform.tfvarsには、変数の実際の値が含まれています。次に例を示します。

```
variables.tf:
  ### EC2 instance variables ###
```

```
variable "ami" {
  type        = string
  description = "EC2 AMI image to be deployed"
}
```

```
variable "instance_type" {
  type        = string
  description = "EC2 instance type"
}
```

```

```

terraform.tfvars:
EC2 instance variables

```

```

ami = "ami-06640050dc3f556bb" //RedHat 8.6 AMI
instance_type = "t2.micro"
ec2_tag = "ora_01"
subnet_id = "subnet-04f5fe7073ff514fb"
ssh_key_name = "sufi_new"
root_volume_size = 30

```

```

ステップバイステップのプロシージャ-順番に実行

1. AWSクラウドシェルにTerraformをインストールする。

```
git clone https://github.com/tfutils/tfenv.git ~/.tfenv
```

```
mkdir ~/bin
```

```
ln -s ~/.tfenv/bin/* ~/bin/
```

```
tfenv install
```

```
tfenv use 1.3.9
```

2. NetApp GitHubパブリックサイトからツールキットをダウンロード

```
git clone https://github.com/NetApp-
Automation/na_aws_fsx_ec2_deploy.git
```

3. initを実行してterraformを初期化する

```
terraform init
```

4. 実行計画の出力

```
terraform plan -out=main.plan
```

5. 実行計画の適用

```
terraform apply "main.plan"
```

6. 完了したらdestroyを実行してリソースを削除します

```
terraform destroy
```

追加情報の参照先

NetApp 解決策 自動化の詳細については、次のWebサイトを参照してください。 "NetApp 解決策の自動化"