Kits d'outils d'automatisation DB

NetApp Solutions

NetApp
September 10, 2024

Sommaire

Kits d’outils d’automatisation DB

SnapCenter automatisation du cycle de vie des clones Oracle

Allen Cao, Niyaz Mohamed, NetApp

Cette solution fournit un kit d’outils d’automatisation basé sur Ansible pour configurer la haute disponibilité et la reprise d’activité (HA/DR) des bases de données Oracle avec AWS FSX ONTAP en tant que stockage de base de données Oracle et les instances EC2 en tant qu’instances de calcul dans AWS.

Objectif

Les clients apprécient la fonctionnalité FlexClone du stockage NetApp ONTAP pour les bases de données, car elle permet de réaliser d’importantes économies en termes de coûts de stockage. Ce kit Ansible automatise la configuration, le clonage et l’actualisation des bases de données Oracle clonées selon un calendrier défini à l’aide des utilitaires de ligne de commande NetApp SnapCenter qui simplifient la gestion du cycle de vie. Ce kit s’applique aux bases de données Oracle déployées sur un système de stockage ONTAP sur site ou dans le cloud public, et gérées par l’outil d’interface utilisateur NetApp SnapCenter.

Cette solution répond aux cas d’utilisation suivants :

- Configurez le fichier de configuration de la spécification de clonage de la base de données Oracle.
- Créez et actualisez la base de données Oracle clone selon un planning défini par l’utilisateur.

Public

Cette solution est destinée aux personnes suivantes :

- Administrateur de bases de données qui gère les bases de données Oracle avec SnapCenter.
- Administrateur du stockage qui gère le stockage ONTAP avec SnapCenter.
- Propriétaire d’application ayant accès à l’interface utilisateur de SnapCenter.

Licence

En accédant au contenu de ce référentiel GitHub, en le téléchargeant, en l’installant ou en l’utilisant, vous acceptez les conditions de la licence énoncées dans "Fichier de licence".

> Il existe certaines restrictions concernant la production et/ou le partage de travaux dérivés avec le contenu de ce référentiel GitHub. Assurez-vous de lire les termes de la Licence avant d’utiliser le contenu. Si vous n’acceptez pas toutes les conditions, n’accédez pas au contenu de ce référentiel, ne le téléchargez pas et ne l’utilisez pas.

Déploiement de la solution

Conditions préalables au déploiement

Le déploiement nécessite les conditions préalables suivantes.

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

```
SnapCenter server:
  version 5.0
  backup policy configured
  Source database protected with a backup policy
```

```
Oracle servers:
  Source server managed by SnapCenter
  Target server managed by SnapCenter
  Target server with identical Oracle software stack as source server
installed and configured
```

Téléchargez la boîte à outils

```
git clone https://bitbucket.ngage.netapp.com/scm/ns-
bb/na_oracle_clone_lifecycle.git
```

Configuration des fichiers des hôtes cibles Ansible

Le kit d'outils inclut un fichier hosts qui définit les cibles sur lesquelles s'exécute un PlayBook Ansible. Il s'agit généralement des hôtes clones Oracle cibles. Voici un exemple de fichier. Une entrée d'hôte comprend l'adresse IP de l'hôte cible ainsi que la clé ssh permettant à un utilisateur admin d'accéder à l'hôte pour exécuter la commande clone ou refresh.

#Hôtes de clonage Oracle

```
[clone_1]
ora_04.cie.netapp.com ansible_host=10.61.180.29
ansible_ssh_private_key_file=ora_04.pem
```

```
[clone_2]
[clone_3]
```

Configuration des variables globales

Les playbooks Ansible prennent des entrées variables à partir de plusieurs fichiers variables. Vous trouverez ci-dessous un exemple de fichier de variable globale vars.yml.

```
# ONTAP specific config variables
# SnapCtr specific config variables
```

```
snapctr_usr: xxxxxxxx
snapctr_pwd: 'xxxxxxxx'
```

```
backup_policy: 'Oracle Full offline Backup'
# Linux specific config variables
# Oracle specific config variables
```

Configuration des variables hôte

Les variables hôtes sont définies dans le répertoire host_vars nommé {{ host_name }}.yml. Vous trouverez ci-dessous un exemple de fichier de variable hôte Oracle cible ora_04.cie.netapp.com.yml qui montre une configuration typique.

```
# User configurable Oracle clone db host specific parameters
```

```
# Source database to clone from
source_db_sid: NTAP1
source_db_host: ora_03.cie.netapp.com
```

```
# Clone database
clone_db_sid: NTAP1DEV
```

```
snapctr_obj_id: '{{ source_db_host }}\{{ source_db_sid }}'
```

Configuration du serveur Oracle cible de clone supplémentaire

La même pile logicielle Oracle doit être installée et corrigée pour le serveur Oracle cible de clone. $ORACLE_BASE et $ORACLE_HOME sont configurés pour l'utilisateur ORACLE .bash_profile. De plus, la variable $ORACLE_HOME doit correspondre au paramètre du serveur Oracle source. Voici un exemple.

```
# .bash_profile
```

```
# Get the aliases and functions
if [ -f ~/.bashrc ]; then
        . ~/.bashrc
fi
```

```
# User specific environment and startup programs
export ORACLE_BASE=/u01/app/oracle
export ORACLE_HOME=/u01/app/oracle/product/19.0.0/NTAP1
```

Exécution de PlayBook

Au total, trois playbooks permettent d'exécuter le cycle de vie des clones d'une base de données Oracle avec les utilitaires de l'interface de ligne de commande SnapCenter.

1. Installez les prérequis du contrôleur Ansible, une seule fois.

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

2. Fichier de spécification de clone de configuration - une seule fois.

```
ansible-playbook -i hosts clone_1_setup.yml -u admin -e
@vars/vars.yml
```

3. Créez et actualisez régulièrement la base de données de clones à partir de crontab avec un script shell pour appeler un PlayBook d'actualisation.

```
0 */4 * * * /home/admin/na_oracle_clone_lifecycle/clone_1_refresh.sh
```

Pour une base de données clone supplémentaire, créez un clone_n_setup.yml et un clone_n_refresh.yml et un clone_n_refresh.sh. Configurez les hôtes cibles Ansible et le fichier hostname.yml dans le répertoire host_vars en conséquence.

Où trouver des informations complémentaires

Pour en savoir plus sur l'automatisation de la solution NetApp, consultez ce site Web "Automatisation des solutions NetApp"

Migration Oracle automatisée

Équipe d'ingénierie des solutions NetApp

Cette solution fournit un kit d'outils d'automatisation basé sur Ansible pour la migration d'une base de données Oracle à l'aide du déplacement PDB avec une méthodologie de disponibilité maximale. La migration peut être n'importe quelle combinaison d'environnements sur site et cloud, tant en tant que source qu'en tant que cible.

Objectif

Ce kit automatise la migration des bases de données Oracle d'un environnement sur site vers le cloud AWS avec le stockage FSX ONTAP et l'instance de calcul EC2 en tant qu'infrastructure cible. Il suppose que le client dispose déjà d'une base de données Oracle sur site déployée dans le modèle CDB/PDB. La boîte à outils permet au client de déplacer un PDB nommé à partir d'une base de données de conteneurs sur un hôte Oracle à l'aide de la procédure de déplacement du PDB Oracle avec une option de disponibilité maximale. Cela signifie que le boîtier de distribution électrique source de toute baie de stockage sur site est relocalise dans une nouvelle base de données de conteneurs avec une interruption de service minimale. La procédure de déplacement Oracle déplace les fichiers de données Oracle pendant que la base de données est en ligne.

Il réachemine ensuite les sessions utilisateur depuis les installations sur site vers les services de base de données transférés au moment de le basculement, lorsque tous les fichiers de données sont transférés vers le cloud AWS. La technologie soulignée est la méthodologie éprouvée de clone à chaud du PDB Oracle.

> Même si le kit de migration est développé et validé sur l'infrastructure cloud AWS, il s'appuie sur les solutions Oracle au niveau des applications. Ce kit s'applique donc à d'autres plateformes de cloud public, comme Azure, GCP, etc

Cette solution répond aux cas d'utilisation suivants :

- Créez un utilisateur de migration et accordez les privilèges requis au serveur de base de données source sur site.
- Déplacer un PDB d'un CDB sur site vers un CDB cible dans le Cloud pendant que le PDB source est en ligne jusqu'au basculement.

Public

Cette solution est destinée aux personnes suivantes :

- Administrateur de bases de données qui migre les bases de données Oracle depuis une infrastructure sur site vers le cloud AWS.
- Architecte de solutions de bases de données qui souhaite migrer des bases de données Oracle d'un environnement sur site vers le cloud AWS.
- Administrateur du stockage qui gère le stockage AWS FSX ONTAP qui prend en charge les bases de données Oracle.
- Propriétaire d'applications qui aime migrer sa base de données Oracle d'une infrastructure sur site vers le cloud AWS.

Licence

En accédant au contenu de ce référentiel GitHub, en le téléchargeant, en l'installant ou en l'utilisant, vous acceptez les conditions de la licence énoncées dans "Fichier de licence".

> Il existe certaines restrictions concernant la production et/ou le partage de travaux dérivés avec le contenu de ce référentiel GitHub. Assurez-vous de lire les termes de la Licence avant d'utiliser le contenu. Si vous n'acceptez pas toutes les conditions, n'accédez pas au contenu de ce référentiel, ne le téléchargez pas et ne l'utilisez pas.

Déploiement de la solution

Conditions préalables au déploiement

Le déploiement nécessite les conditions préalables suivantes.

```
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)
```

Téléchargez la boîte à outils

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

Configuration des variables hôte

Les variables hôtes sont définies dans le répertoire host_vars nommé {{ host_name }}.yml. Un exemple de fichier de variable hôte nom_hôte.yml est inclus pour démontrer une configuration typique. Principaux éléments à prendre en compte :

```
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
```

Configuration du fichier hôte du serveur de BASE DE DONNÉES

Instance AWS EC2 utilise l'adresse IP pour la dénomination des hôtes par défaut. Si vous utilisez un nom différent dans le fichier hosts pour Ansible, configurez la résolution de dénomination des hôtes dans le fichier /etc/hosts pour les serveurs source et cible. Voici un exemple.

```
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
```

Exécution du manuel de vente - exécutée dans l'ordre

1. Installez les prérequis du contrôleur Ansible.

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

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

2. Exécutez des tâches de pré-migration sur un serveur sur site, en supposant que admin est un utilisateur ssh pour la connexion à l'hôte Oracle sur site avec l'autorisation sudo.

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

3. Exécutez le déplacement du PDB Oracle du CDB sur site vers le CDB cible dans l'instance AWS EC2, en supposant que l'utilisateur EC2 se connecte à l'instance de BD EC2 et que db1.pem possède des paires de clés SSH EC2-user.

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

Où trouver des informations complémentaires

Pour en savoir plus sur l'automatisation de la solution NetApp, consultez ce site Web "Automatisation des solutions NetApp"

Automatisez la haute disponibilité et la reprise après incident Oracle dans AWS FSX ONTAP

Équipe d'ingénierie des solutions NetApp

Cette solution fournit un kit d'outils d'automatisation basé sur Ansible pour configurer la haute disponibilité et la reprise d'activité (HA/DR) des bases de données Oracle avec AWS FSX ONTAP en tant que stockage de base de données Oracle et les instances EC2 en tant qu'instances de calcul dans AWS.

Objectif

Ce kit automatise les tâches de configuration et de gestion d'un environnement haute disponibilité et de reprise d'activité pour la base de données Oracle déployée dans le cloud AWS avec FSX pour le stockage ONTAP et les instances de calcul EC2.

Cette solution répond aux cas d'utilisation suivants :

- Configurez l'hôte cible HA/DR : configuration du noyau, configuration Oracle pour qu'il corresponde à l'hôte du serveur source.
- Setup FSX ONTAP : peering de cluster, peering de vServers, configuration des relations snapmirror des volumes Oracle de la source à la cible.
- Sauvegardez les données de la base de données Oracle via snapshot - exécutez-les hors crontab
- Sauvegarder le journal d'archivage de la base de données Oracle via snapshot - exécuter hors crontab
- Exécuter le basculement et la restauration sur un hôte HA/DR et tester et valider l'environnement HA/DR
- Exécutez la resynchronisation après le test de basculement pour rétablir la relation snapmirror des volumes de base de données en mode HA/DR

Public

Cette solution est destinée aux personnes suivantes :

- Administrateur de bases de données qui a configuré une base de données Oracle dans AWS pour bénéficier de la haute disponibilité, de la protection des données et de la reprise après incident.
- Architecte de solutions de bases de données et s'intéresse à la solution de haute disponibilité/reprise après incident Oracle au niveau du stockage dans le cloud AWS.
- Administrateur du stockage qui gère le stockage AWS FSX ONTAP qui prend en charge les bases de données Oracle.
- Un propriétaire d'applications qui aime créer une base de données Oracle pour la haute disponibilité/reprise dans l'environnement AWS FSX/EC2.

Licence

En accédant au contenu de ce référentiel GitHub, en le téléchargeant, en l'installant ou en l'utilisant, vous acceptez les conditions de la licence énoncées dans "Fichier de licence".

> Il existe certaines restrictions concernant la production et/ou le partage de travaux dérivés avec le contenu de ce référentiel GitHub. Assurez-vous de lire les termes de la Licence avant d'utiliser le contenu. Si vous n'acceptez pas toutes les conditions, n'accédez pas au contenu de ce référentiel, ne le téléchargez pas et ne l'utilisez pas.

Déploiement de la solution

Conditions préalables au déploiement

Le déploiement nécessite les conditions préalables suivantes.

```
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
```

Téléchargez la boîte à outils

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

Configuration des variables globales

Les playbooks Ansible sont basés sur des variables. Un exemple de fichier de variables globales fsx_vars_example.yml est inclus pour démontrer une configuration typique. Principaux éléments à prendre en compte :

```
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.
```

Configuration des variables hôte

Les variables hôtes sont définies dans le répertoire host_vars nommé {{ host_name }}.yml. Un exemple de fichier de variable hôte nom_hôte.yml est inclus pour démontrer une configuration typique. Principaux éléments à prendre en compte :

```
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.
```

Configuration du fichier hôte du serveur de BASE DE DONNÉES

Instance AWS EC2 utilise l'adresse IP pour la dénomination des hôtes par défaut. Si vous utilisez un nom différent dans le fichier hosts pour Ansible, configurez la résolution de dénomination des hôtes dans le fichier /etc/hosts pour les serveurs source et cible. Voici un exemple.

```
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
```

Exécution du manuel de vente - exécutée dans l'ordre

1. Installez les versions préalables du contrôleur Ansible.

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

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

2. Configurez l'instance de base de données EC2 cible.

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

3. Configurez la relation ONTAP FSX snapmirror entre les volumes de base de données source et cible.

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

4. Sauvegardez les volumes de données de la base de données Oracle via snapshot à partir de crontab.

```
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. Sauvegarde des volumes du journal d'archivage de la base de données Oracle via snapshot à partir de crontab.

```
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. Exécutez le basculement et restaurez la base de données Oracle sur l'instance de base de données EC2 cible. Testez et validez la configuration HA/DR.

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

7. Exécuter la resynchronisation après le test de basculement - rétablir la relation snapmirror des volumes de base de données en mode de réplication.

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

Où trouver des informations complémentaires

Pour en savoir plus sur l'automatisation de la solution NetApp, consultez ce site Web "Automatisation des solutions NetApp"

Provisionnement du cluster AWS FSX ONTAP et de l'instance EC2

Équipe d'ingénierie des solutions NetApp

Cette solution fournit un kit d'outils d'automatisation basé sur Terraform pour le provisionnement du cluster FSX ONTAP et de l'instance de calcul EC2.

Objectif

Ce kit automatise les tâches de provisionnement d'un cluster de stockage AWS FSX ONTAP et d'une instance de calcul EC2, qui peut ensuite être utilisée pour le déploiement de bases de données.

Cette solution répond aux cas d'utilisation suivants :

- Provisionnez une instance de calcul EC2 dans le cloud AWS dans un sous-réseau VPC prédéfini et définissez la clé ssh pour l'accès à l'instance EC2 en tant qu'utilisateur EC2.
- Provisionnez un cluster de stockage AWS FSX ONTAP dans les zones de disponibilité souhaitées, configurez un SVM de stockage et configurez un mot de passe utilisateur fsxadmin du cluster.

Public

Cette solution est destinée aux personnes suivantes :

- Administrateur de bases de données gérant les bases de données dans l'environnement AWS EC2.
- Architecte de solutions de bases de données qui s'intéresse au déploiement de bases de données dans l'écosystème AWS EC2.
- Administrateur du stockage qui gère le stockage AWS FSX ONTAP qui prend en charge les bases de données.
- Propriétaire d'applications qui aime standup base de données dans l'écosystème AWS EC2.

Licence

Il existe certaines restrictions concernant la production et/ou le partage de travaux dérivés avec le contenu de ce référentiel GitHub. Assurez-vous de lire les termes de la Licence avant d'utiliser le contenu. Si vous n'acceptez pas toutes les conditions, n'accédez pas au contenu de ce référentiel, ne le téléchargez pas et ne l'utilisez pas.

Déploiement de la solution

Conditions préalables au déploiement

Le déploiement nécessite les conditions préalables suivantes.

```
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
```

Téléchargez la boîte à outils

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

Connectivité et authentification

Le kit d'outils est censé être exécuté à partir d'un shell cloud AWS. Le shell cloud AWS est un shell basé sur un navigateur qui facilite la gestion, la découverte et l'interaction avec vos ressources AWS de manière sécurisée. CloudShell est pré-authentifié avec les informations d'identification de votre console. Les outils de développement et d'exploitation courants sont préinstallés. Aucune installation ou configuration locale n'est donc nécessaire.

Configuration des fichiers Terraform Provider.tf et main.tf

Le Provider.tf définit le fournisseur à partir duquel Terraform provisionne des ressources via des appels API. Le fichier main.tf définit les ressources et les attributs des ressources à provisionner. Voici quelques détails :

```
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
    }
  }
  ....
```

Configuration des variables Terraform.tf et terraform.tfvars

Variables.tf déclare les variables à utiliser dans main.tf. Le terraform.tfvars contient les valeurs réelles des variables. Voici quelques exemples :

```
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

```

```

Procédures étape par étape - exécutées dans l'ordre

1. Installez Terraform dans le shell cloud AWS.

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

```
mkdir ~/bin
```

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

```
tfenv install
```

```
tfenv use 1.3.9
```

2. Téléchargez le kit d'outils depuis le site public de NetApp GitHub

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

3. Exécutez init pour initialiser terraform

```
terraform init
```

4. Sortir le plan d'exécution

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

5. Appliquer le plan d'exécution

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

6. Exécutez détruire pour supprimer les ressources une fois l'opération terminée

```
terraform destroy
```

Où trouver des informations complémentaires

Pour en savoir plus sur l'automatisation de la solution NetApp, consultez ce site Web "Automatisation des solutions NetApp"