Introduction au déploiement Ansible chez Omnitouch

Vue d'ensemble

Omnitouch Network Services utilise Ansible comme plateforme d'automatisation de l'infrastructure pour déployer des solutions complètes de réseau cellulaire (4G/5G) de manière cohérente, répétable et automatisée. Ce document fournit un aperçu de la manière dont nous exploitons Ansible pour orchestrer des déploiements télécom complexes.

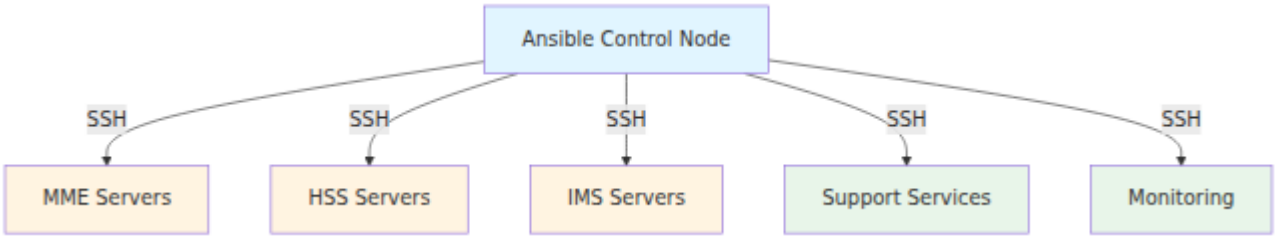
Qu'est-ce qu'Ansible ?

Ansible est un outil d'automatisation open-source qui vous permet de :

- Configurer des systèmes
- Déployer des logiciels
- Orchestrer des flux de travail complexes
- Gérer l'infrastructure en tant que code

Ansible utilise une approche déclarative - vous décrivez l'**état souhaité** de vos systèmes, et Ansible s'assure qu'ils atteignent cet état.

Comment Omnitouch utilise Ansible



Concepts clés

1. Inventaire (Fichiers d'hôtes)

Définit **quels** systèmes gérer. Chaque déploiement client a un fichier d'hôtes qui décrit :

- Toutes les machines virtuelles dans le réseau
- Leurs adresses IP
- La configuration réseau
- Les paramètres spécifiques aux services

Les fichiers d'hôtes sont ce avec quoi vous allez travailler pour définir votre réseau.

Voir : Configuration du fichier d'hôtes

2. Rôles

Définit **comment** configurer chaque composant. Les rôles sont des unités réutilisables qui contiennent :

- Tâches (étapes à exécuter)
- Modèles (modèles de fichiers de configuration)
- Gestionnaires (actions déclenchées par des changements)
- Variables (valeurs de configuration par défaut)

Exemples de rôles pour les composants OmniCore : `omnihss`, `omnisgwc`, `omnipgwc`, `omnidra`, etc.

Ceux-ci sont définis par l'équipe ONS, bien que vous puissiez les modifier, il existe généralement des moyens plus propres d'apporter les ajustements nécessaires depuis votre fichier d'hôtes.

3. Playbooks

Orchestre **quand** et **où** les rôles sont appliqués :

```
- name: Deploy EPC Core
  hosts: mme
  roles:
    - common
    - omnimme
```

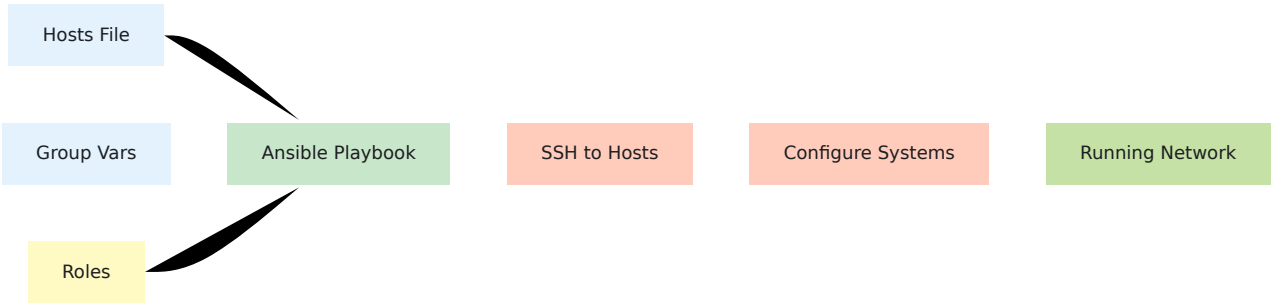
Nous les utilisons essentiellement comme groupes pour les rôles.

4. Variables de groupe

Fournit une **configuration spécifique au client** qui remplace les valeurs par défaut des rôles. C'est ici que la personnalisation du client se produit sans modifier les rôles de base.

Voir : Variables de groupe et configuration

Architecture de déploiement



Le processus de déploiement

1. Définir l'infrastructure

Créez un fichier d'hôtes décrivant votre topologie réseau :

Remarque de planification : Avant de définir l'infrastructure, consultez la Norme de planification IP pour des conseils sur la segmentation du réseau, l'allocation des adresses IP et l'organisation des sous-réseaux.

Utilisateurs de Proxmox : Si vous déployez sur Proxmox, consultez Déploiement de VM/LXC Proxmox pour l'approvisionnement automatisé de VM/conteneurs.

Voir : Configuration du fichier d'hôtes et Référence de configuration

```
mme:
  hosts:
    customer-mme01:
      ansible_host: 10.10.1.15
      mme_code: 1
```

2. Personnaliser la configuration

Définissez des variables spécifiques au client dans `group_vars` :

```
plmn_id:
  mcc: '001'
  mnc: '01'
customer_name_short: customer
```

#ToDo - Ajouter un lien ici vers la référence de configuration pour la liste complète

3. Exécuter des playbooks

Déployez le réseau :

```
ansible-playbook -i hosts/customer/host_files/production.yml
services/epc.yml
```

4. Déploiement automatisé

Ansible va :

- Créer/provisionner des VM (si vous utilisez l'intégration Proxmox/VMware)
- Configurer le réseau
- Installer des paquets logiciels à partir du cache APT
- Déployer le code de l'application
- Configurer les services avec les paramètres du client
- Démarrer les services
- Valider le déploiement

Composants clés que nous déployons

OmniCore (Plateforme de cœur de paquet 4G/5G)

- **OmniHSS** - Serveur d'abonnés à domicile
- **OmniSGW** - Passerelle de service (plan de contrôle)
- **OmniPGW** - Passerelle de paquet (plan de contrôle)
- **OmniUPF** - Fonction de plan utilisateur
- **OmniDRA** - Agent de routage Diameter
- **OmniTWAG** - Passerelle d'accès WLAN de confiance

Voir : https://docs.omnitouch.com.au/docs/repos/OmniCore

OmniCall (Plateforme de voix et de messagerie)

- **OmniCall CSCF** - Fonction de contrôle de session d'appel (P-CSCF, I-CSCF, S-CSCF)
- **OmniTAS** - Serveur d'application IMS (services VoLTE/VoNR)
- **OmniMessage** - Centre SMS (SMS-C)
- **OmniMessage SMPP** - Support du protocole SMPP
- **OmniSS7** - Composants de signalisation SS7 (STP, HLR, CAMEL)
- **VisualVoicemail** - Fonctionnalité de messagerie vocale

Voir : https://docs.omnitouch.com.au/docs/repos/OmniCall

OmniCharge/OmniCRM

- **Plateforme CRM** - Gestion de la relation client, auto-inscription, facturation

Voir : https://docs.omnitouch.com.au/docs/repos/OmniCharge

Services de support

- **DNS** - Résolution DNS réseau
- **Serveur de licences** - Gestion des licences
- **Surveillance** - Prometheus, Grafana

Voir : Aperçu de l'architecture de déploiement

Gestion des paquets

Nous utilisons un modèle de distribution de paquets hybride :

Paquets APT précompilés

Tous les logiciels Omnitouch sont distribués sous forme de paquets Debian (`.deb` files) :

- Construits à partir de la source dans notre pipeline CI/CD
- Versionnés et testés
- Hébergés sur des dépôts de paquets

Système de cache APT

Les clients peuvent choisir entre :

1. **Cache APT local** - Miroir des paquets requis sur site pour un déploiement hors ligne
2. **Dépôt public** - Accès direct au dépôt de paquets hébergé par Omnitouch

Voir : Système de cache APT

Gestion des licences

Tous les composants logiciels Omnitouch nécessitent des licences valides gérées via un serveur de licences central :

- Les composants vérifient la validité de la licence au démarrage
- Les fonctionnalités sont activées/désactivées en fonction de la licence
- Le serveur de licences peut être local ou hébergé dans le cloud

Voir : Serveur de licences

Avantages de cette approche

Répétabilité

Les mêmes playbooks Ansible peuvent déployer :

- Laboratoires de développement
- Environnements de test
- Réseaux de production
- Sites clients

Cohérence

Chaque déploiement utilise les mêmes configurations testées, réduisant ainsi les erreurs humaines.

Contrôle de version

L'infrastructure est définie comme du code dans Git :

- Suivre tous les changements
- Réviser avant le déploiement
- Revenir en arrière si nécessaire

Personnalisation sans complexité

Les clients peuvent personnaliser leur déploiement via `group_vars` sans modifier les rôles de base.

Déploiement rapide

Déployez un réseau cellulaire complet en quelques heures au lieu de jours ou de semaines.

Pour commencer

Prérequis

Avant d'exécuter des playbooks Ansible, vous devez configurer un environnement virtuel Python et installer les dépendances requises.

1. Créer un environnement virtuel Python

Créez un environnement Python isolé pour le déploiement Ansible :

```
python3 -m venv .venv
```

2. Activer l'environnement virtuel

Activez l'environnement virtuel :

```
source .venv/bin/activate
```

Sous Windows, utilisez :

```
.venv\Scripts\activate
```

3. Installer les paquets requis

Installez toutes les dépendances à partir du fichier requirements.txt :

```
pip install -r requirements.txt
```

Cela installera Ansible et tous les paquets Python nécessaires pour l'automatisation du déploiement Omnitouch.

Remarque : Gardez l'environnement virtuel activé chaque fois que vous exécutez des commandes Ansible. Vous pouvez le désactiver lorsque vous avez terminé en exécutant `deactivate`.

Étapes de déploiement

1. Consultez la Configuration du fichier d'hôtes pour comprendre comment définir votre réseau
2. Apprenez à connaître les Variables de groupe pour la personnalisation
3. Comprenez le Système de cache APT pour la gestion des paquets
4. Consultez l'Architecture de déploiement pour voir comment tout s'assemble
5. Déployez !

Prochaines étapes

- Norme de planification IP - **Planifiez votre architecture réseau et votre allocation IP**
- Configuration du fichier d'hôtes - Apprenez à définir votre topologie réseau
- Système de cache APT - Comprenez la distribution des paquets
- Serveur de licences - Découvrez la gestion des licences
- Aperçu de l'architecture de déploiement - Voir l'ensemble du tableau
- Configuration des variables de groupe - Personnalisez votre déploiement
- Playbooks utilitaires - Outils opérationnels pour les vérifications de santé, les sauvegardes et la maintenance

Dépôt APT & Distribution de Paquets

Vue d'ensemble

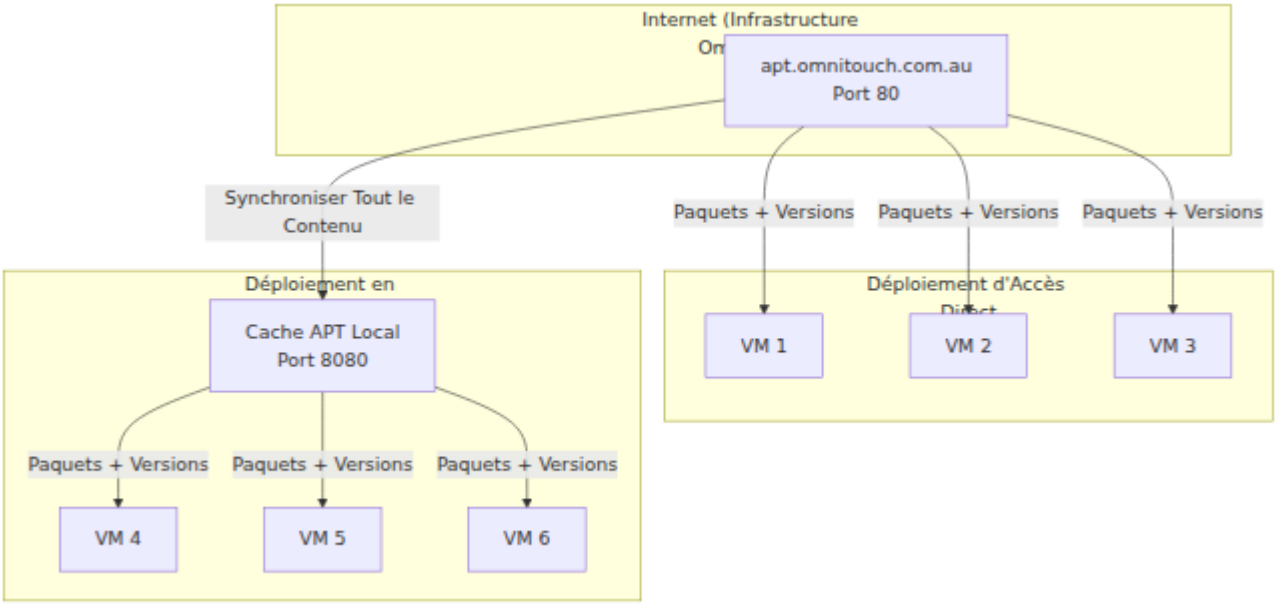
Le système APT d'Omnitouch fournit une distribution de paquets pour tous les déploiements. Deux types de contenu sont servis :

1. **Paquets APT** — Paquets Debian installés via `apt install`
2. **Versions Binaires** — Binaires pré-construits téléchargés directement (exportateurs Prometheus, agents, etc.)

Deux modèles de déploiement sont supportés :

1. **Accès Direct** — Les VM récupèrent les paquets directement depuis `apt.omnitouch.com.au`
2. **Miroir de Cache Local** — Un serveur local se synchronise avec Omnitouch et sert des paquets aux VM (pour des déploiements hors ligne/isolés)

Architecture



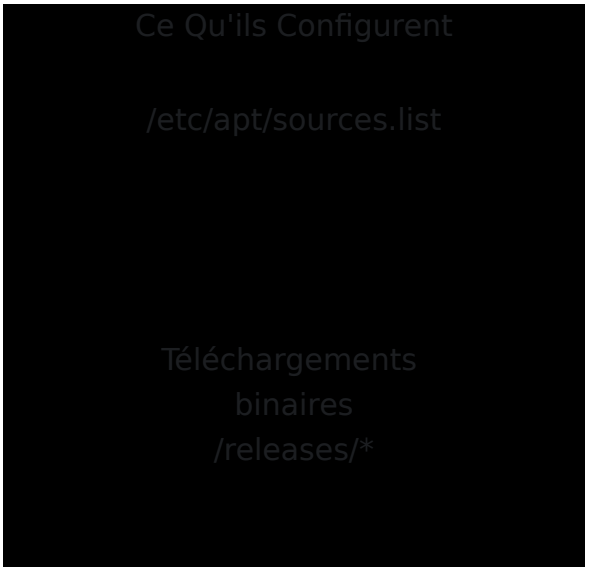
Contenu Servi

Le serveur APT héberge tout le contenu requis pour les déploiements :

Type de Contenu	Description	Chemin
Paquets Omnitouch	Paquets `.deb` construits sur mesure (omnihss, omnimme, etc.)	`/dists/<distro>/`
Paquets Ubuntu	Paquets Ubuntu mis en cache avec toutes les dépendances	`/<distro>/pool/main/`
Versions GitHub	Binaires pré-construits (Prometheus, Grafana, Homer, etc.)	`/releases/<org>/<repo>/`
Archives Source	Archives source pour les applications web (CGrateS_UI, speedtest)	`/repos/`
Paquets Tiers	Galera, FRR, InfluxDB, KeyDB, etc.	`/releases/<vendor>/`

Variables de Configuration

Deux ensembles de variables distincts contrôlent la distribution des paquets. Comprendre leurs objectifs est essentiel pour une configuration correcte.



Objectifs des Variables

Ensemble de Variables	Objectif	Utilisé Pour
`apt_repo`	Configure les sources de paquets APT	`/etc/apt/sources.list` et `/etc/apt/sources.list.d/*.list`
`remote_apt_*`	Configure les URL de téléchargement binaire	Téléchargement de fichiers depuis le chemin `/releases/` (Node Exporter, Zabbix, Nagios, etc.)

Quand Chaque Ensemble de Variables Est Utilisé

Scénario	Sources APT (`apt_repo`)	Téléchargements Binaires (`remote_apt_*`)
`use_apt_cache: true`	Utilise `apt_repo.apt_server`	Utilise `apt_repo.apt_server`
`use_apt_cache: false`	Utilise `apt_repo.*` avec des identifiants	Utilise `remote_apt_*` avec des identifiants

Lorsque `use_apt_cache: false`, les deux ensembles de variables sont requis.

Option 1 : Accès Direct

Pour les déploiements avec connectivité Internet, les VM récupèrent les paquets directement depuis le serveur APT d'Omnitouch.

Exigences Réseau

Liste Blanche des IP Sources : Votre adresse IP publique doit être ajoutée à la liste blanche sur le serveur APT d'Omnitouch. Lors de la configuration, fournissez vos sous-réseaux sources à Omnitouch. En retour, vous recevrez :

- **Nom d'utilisateur** et **mot de passe** pour l'authentification HTTP Basic
- **FQDN** pour le serveur APT

Exigences de Pare-feu : L'accès sortant aux plages IP suivantes d'Omnitouch doit être autorisé :

Réseau	Plage
IPv4	`144.79.167.0/24`
IPv4	`160.22.43.0/24`
IPv6	`2001:df3:dec0::/48`
ASN	`AS152894`

Services nécessitant un accès à l'infrastructure Omnitouch :

Service	Port	Protocole	Objectif
Serveur APT	80	TCP	Téléchargements de paquets
Serveur APT	53	TCP/UDP	Résolution DNS pour `apt.omnitouch.com.au`
Serveur de Licence	123	UDP	Synchronisation horaire NTP pour validation de licence
Serveur de Licence	53	TCP/UDP	Résolution DNS pour validation de licence

Assurez-vous que le trafic HTTP (TCP/80), NTP (UDP/123) et DNS (TCP+UDP/53) est autorisé vers les plages IP d'Omnitouch.

Configuration

```yaml
all:
  vars:
    use_apt_cache: false

    # Configuration des sources de paquets APT
    # Configure /etc/apt/sources.list pour les commandes apt
install
    apt_repo:
      apt_server: "apt.omnitouch.com.au"
      apt_repo_username: "your-username"
      apt_repo_password: "your-password"

    # Configuration des téléchargements binaires
    # Utilisé pour télécharger des fichiers depuis le chemin
/releases/
    remote_apt_server: "apt.omnitouch.com.au"
    remote_apt_port: 80
    remote_apt_protocol: "http"
    remote_apt_user: "your-username"
    remote_apt_password: "your-password"
```

Paramètres

Sources de Paquets APT (`apt_repo`)

Paramètre	Type	Requis	Par Défaut	Descriptio
`apt_repo.apt_server`	Chaîne	Oui	-	Nom d'hôte o adresse IP du serveur APT
`apt_repo.apt_repo_username`	Chaîne	Oui	-	Nom d'utilisateur pour l'authentificat HTTP Basic pc les sources AF
`apt_repo.apt_repo_password`	Chaîne	Oui	-	Mot de passe pour l'authentificat HTTP Basic pc les sources AF

Téléchargements Binaires (`remote_apt_*`)

Paramètre	Type	Requis	Par Défaut	Description
`remote_apt_server`	Chaîne	Oui	-	Nom d'hôte ou IP du serveur pour les téléchargements binaires
`remote_apt_port`	Entier	Non	`80`	Port du serveur pour les téléchargements binaires
`remote_apt_protocol`	Chaîne	Non	`http`	Protocole (`http` ou `https`)
`remote_apt_user`	Chaîne	Oui	-	Nom d'utilisateur pour l'authentification HTTP Basic pour les téléchargements
`remote_apt_password`	Chaîne	Oui	-	Mot de passe pour l'authentification HTTP Basic pour les téléchargements

Général

Paramètre	Type	Requis	Par Défaut	Description
`use_apt_cache`	Booléen	Oui	-	Doit être `false` pour un accès direct

Modèles d'URL (Accès Direct)

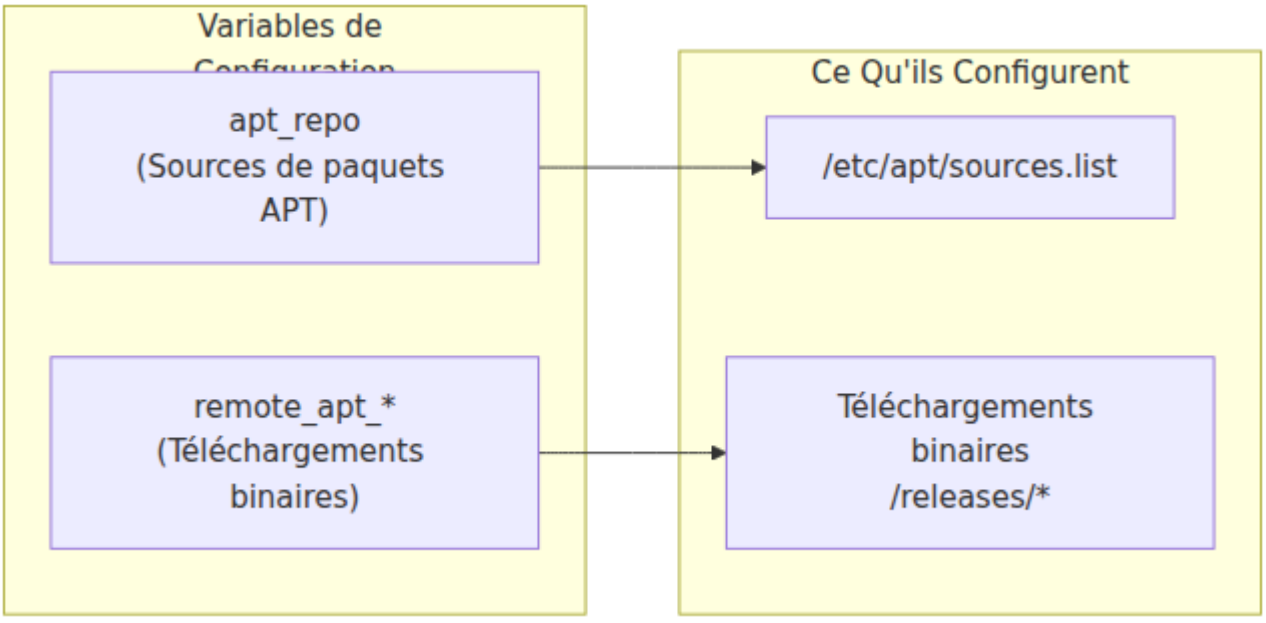
Sources de Paquets APT (configurées dans `/etc/apt/sources.list`) :

```
deb [trusted=yes] http://{apt_repo_username}:
{apt_repo_password}@{apt_server}/ noble main
```

Téléchargements Binaires (utilisés par les tâches `get_url` d'Ansible) :

```
http://{remote_apt_user}:
{remote_apt_password}@{remote_apt_server}:
{remote_apt_port}/releases/prometheus/node_exporter/node_exporter-
1.8.1.linux-amd64.tar.gz
```

Comment Cela Fonctionne



Les VM s'authentifient avec l'authentification HTTP Basic pour les paquets APT et les téléchargements binaires. Les paquets système Ubuntu sont également servis depuis le serveur Omnitouch (pré-mis en cache), donc les VM n'ont pas besoin d'accéder aux miroirs Ubuntu.

Option 2 : Miroir de Cache Local

Pour les déploiements hors ligne, isolés ou limités en bande passante, déployez un cache APT local qui synchronise tout le contenu d'Omnitouch.

Architecture

Configuration

Définissez le serveur de cache dans votre fichier hosts avec sa configuration de dépôt :

```yaml
apt_cache_servers:
  hosts:
    customer-apt-cache:
      ansible_host: 192.168.1.100
      gateway: 192.168.1.1
  vars:
    # Le serveur de cache synchronise les paquets depuis le dépôt
authentifié
    remote_apt_server: "apt.omnitouch.com.au"
    remote_apt_port: 80
    remote_apt_protocol: "http"
    remote_apt_user: "your-username"
    remote_apt_password: "your-password"

all:
  vars:
    # use_apt_cache: true  # Défini automatiquement lorsque le
groupe apt_cache_servers existe
    # apt_repo.apt_server: auto-dérivé à 192.168.1.100 (premier
serveur de cache)
```

Comment cela fonctionne :

- **Serveur de cache** (`192.168.1.100`) : Utilise les identifiants `remote_apt_*` pour synchroniser les paquets depuis `apt.omnitouch.com.au:80`
- **Tous les autres hôtes** : Dérivent automatiquement `apt_repo.apt_server: "192.168.1.100"` et récupèrent depuis le cache au port `8080` sans identifiants

Paramètres

Sources de Paquets APT (`apt_repo`)

Paramètre	Type	Requis	Par Défaut	Descripti
`apt_repo.apt_server`	Chaîne	Oui	Auto-dérivé	IP du serveur cache local. D automatiquen du premier hô `apt_cache_se` si non spécifié
`apt_repo.apt_repo_username`	Chaîne	Non	-	Non requis lor l'utilisation du cache (aucun authentificati nécessaire)
`apt_repo.apt_repo_password`	Chaîne	Non	-	Non requis lor l'utilisation du cache (aucun authentificati nécessaire)

Synchronisation du Serveur de Cache (`remote_apt_*`)

Ces variables configurent comment le serveur de cache synchronise le contenu d'Omnitouch :

Paramètre	Type	Requis	Par Défaut	Description
`remote_apt_server`	Chaîne	Oui	-	Serveur APT Omnitouch à synchroniser
`remote_apt_port`	Entier	Non	`80`	Port du serveur APT Omnitouch
`remote_apt_protocol`	Chaîne	Non	`http`	Protocole pour la connexion de synchronisation
`remote_apt_user`	Chaîne	Oui	-	Identifiants pour la synchronisation depuis Omnitouch
`remote_apt_password`	Chaîne	Oui	-	Identifiants pour la synchronisation depuis Omnitouch

Général

Paramètre	Type	Requis	Par Défaut	Description
`use_apt_cache`	Booléen	Non	`true`	Défini automatiquement sur `true` lorsque le groupe `apt_cache_servers` existe
`apt_cache_port`	Entier	Non	`8080`	Port sur lequel le serveur de cache local écoute

Modèles d'URL (Mode Cache)

Sources de Paquets APT (configurées dans `/etc/apt/sources.list`) :

```
deb [trusted=yes] http://192.168.1.100:8080/noble noble main
```

Téléchargements Binaires (utilisés par les tâches `get_url` d'Ansible) :

```
http://192.168.1.100:8080/releases/prometheus/node_exporter/node_expo
1.8.1.linux-amd64.tar.gz
```

Aucun identifiant requis pour l'accès au cache - il utilise la configuration APT `[trusted=yes]`.

Déploiement du Cache

1. **Provisionnez le serveur de cache** (VM ou conteneur LXC avec 50+ Go de disque)

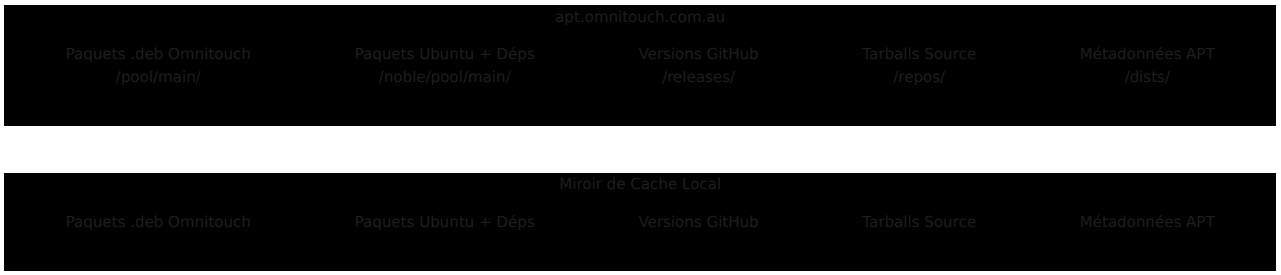
2. **Exécutez le playbook de configuration du cache :**

   ```
   ansible-playbook -i hosts/customer/production.yml
   services/apt_cache.yml
   ```

3. **Vérifiez le cache** en naviguant vers `http://192.168.1.100:8080/`

Ce Qui Est Synchronisé

Le miroir de cache synchronise **tout le contenu** depuis le serveur APT d'Omnitouch en utilisant un téléchargement wget récursif :

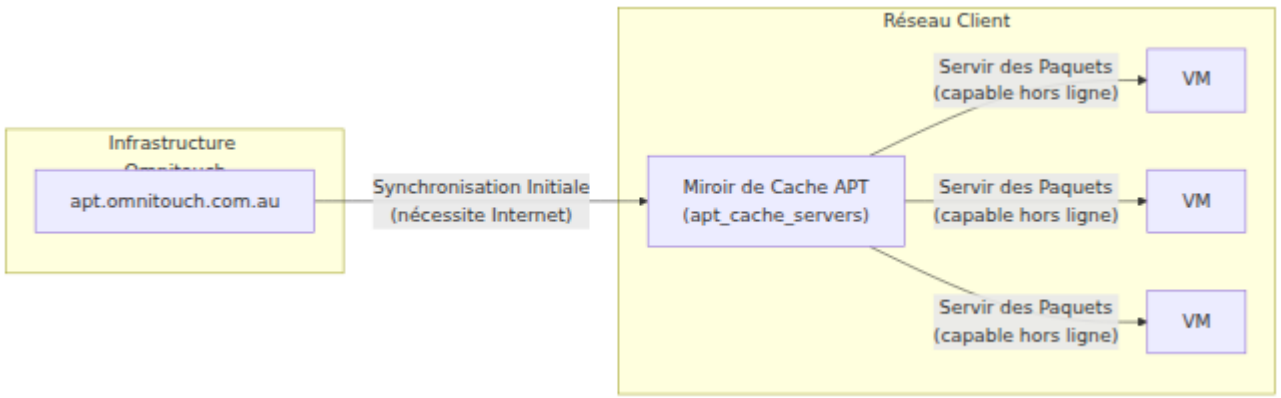


Répertoires de contenu synchronisés :

Chemin	Contenu
`/dists/<distro>/`	Métadonnées du dépôt APT (Fichiers Packages, Release)
`/pool/main/`	Paquets .deb personnalisés d'Omnitouch
`/<distro>/pool/main/`	Paquets Ubuntu et toutes les dépendances
`/releases/`	Versions GitHub (Prometheus, Grafana, Zabbix, etc.)
`/repos/`	Tarballs source (Erlang, Elixir, CGrateS_UI, etc.)

Après la synchronisation initiale, le cache peut servir tous les paquets sans connectivité Internet.

Comment Cela Fonctionne



Le miroir de cache utilise `wget --recursive` avec authentification HTTP Basic pour télécharger tout le contenu depuis le serveur APT d'Omnitouch. Les

synchronisations suivantes ne téléchargent que les fichiers nouveaux/modifiés (timestamping).

Configuration Automatique

Lorsqu'un groupe `apt_cache_servers` existe dans votre inventaire, le système :

1. Définit `use_apt_cache: true` pour tous les hôtes (à moins d'être explicitement remplacé)
2. Dérive `apt_repo.apt_server` de l'IP `ansible_host` du premier serveur de cache

Exemple de Configuration Minimale

```
apt_cache_servers:
  hosts:
    apt-cache-01:
      ansible_host: 192.168.1.100
      gateway: 192.168.1.1
  vars:
    # Le serveur de cache synchronise le contenu depuis le dépôt
Omnitouch
    remote_apt_server: "apt.omnitouch.com.au"
    remote_apt_user: "your-username"
    remote_apt_password: "your-password"
```

Ce qui se passe automatiquement :

- Tous les hôtes (sauf le serveur de cache) obtiennent `use_apt_cache: true`
- Tous les hôtes (sauf le serveur de cache) obtiennent `apt_repo.apt_server: "192.168.1.100"`
- Tous les hôtes récupèrent depuis `http://192.168.1.100:8080/` sans identifiants
- Le serveur de cache synchronise les paquets depuis `http://your-username:your-password@apt.omnitouch.com.au/`

Remplacer le Comportement Automatique

Pour forcer l'accès direct même avec des serveurs de cache définis :

```
all:
  vars:
    use_apt_cache: false  # Forcer l'accès direct même avec des
serveurs de cache définis

    apt_repo:
      apt_server: "apt.omnitouch.com.au"
      apt_repo_username: "user"
      apt_repo_password: "pass"

    remote_apt_server: "apt.omnitouch.com.au"
    remote_apt_user: "user"
    remote_apt_password: "pass"
```

Résumé de la Configuration

Scénario 1 : Accès Direct au Serveur APT (Pas de Cache)

Tous les hôtes récupèrent les paquets directement depuis le serveur de dépôt APT.

```yaml
all:
  vars:
    use_apt_cache: false

    # Sources de paquets APT - utilisées par tous les hôtes
    apt_repo:
      apt_server: "apt.omnitouch.com.au"
      apt_repo_username: "user"
      apt_repo_password: "pass"

    # Téléchargements binaires - utilisés par tous les hôtes
    remote_apt_server: "apt.omnitouch.com.au"
    remote_apt_port: 80
    remote_apt_protocol: "http"
    remote_apt_user: "user"
    remote_apt_password: "pass"
```

Résultat : Tous les hôtes génèrent `deb [trusted=yes] http://user:pass@apt.omnitouch.com.au/ noble main`

Scénario 2 : Serveur de Cache APT Défini dans le Fichier Hosts (Automatique)

Le serveur de cache est dans votre inventaire et sera déployé/synchronisé par Ansible.

```yaml
apt_cache_servers:
  hosts:
    cache-server:
      ansible_host: 192.168.1.100
      gateway: 192.168.1.1
  vars:
    # Le serveur de cache synchronise les paquets depuis le dépôt
authentifié
    remote_apt_server: "apt.omnitouch.com.au"
    remote_apt_port: 80
    remote_apt_protocol: "http"
    remote_apt_user: "user"
    remote_apt_password: "pass"

# Aucune configuration nécessaire dans all: vars:
# Tout est auto-dérivé du groupe apt_cache_servers
```

Résultat :

- **Serveur de cache** : Synchronise depuis `http://user:pass@apt.omnitouch.com.au:80/`
- **Tous les autres hôtes** : Génèrent `deb [trusted=yes] http://192.168.1.100:8080/noble noble main` (sans identifiants)

Scénario 3 : Cache APT DISTANT NON dans le Fichier Hosts (Manuel)

Le serveur de cache existe ailleurs et est déjà configuré (non géré par votre Ansible).

```yaml
all:
  vars:
    use_apt_cache: true

    # Pointer tous les hôtes vers le serveur de cache externe
    apt_repo:
      apt_server: "192.168.1.100"  # IP du serveur de cache
externe
      apt_repo_port: 8080           # Le cache fonctionne
généralement sur le port 8080

# Aucun groupe apt_cache_servers nécessaire
# Aucun remote_apt_* nécessaire (le cache est déjà configuré
externement)
```

Résultat : Tous les hôtes génèrent `deb [trusted=yes] http://192.168.1.100:8080/noble noble main` (sans identifiants)

Exemple Complet

Voici un exemple complet montrant la configuration du serveur de cache avec plusieurs hôtes d'application :

```yaml
# Groupe de Serveur de Cache APT
apt_cache_servers:
  hosts:
    customer-apt-cache:
      ansible_host: 10.179.1.114
      gateway: 10.179.1.1
      host_vm_network: "vmbr0"
      num_cpus: 4
      memory_mb: 16384
      proxmoxLxcDiskSizeGb: 120
  vars:
    # Le serveur de cache synchronise les paquets depuis le dépôt
authentifié
    remote_apt_server: "apt.omnitouch.com.au"
    remote_apt_port: 80
    remote_apt_protocol: "http"
    remote_apt_user: "customer-username"
    remote_apt_password: "customer-secure-token"

# Serveurs d'Application
hss:
  hosts:
    customer-hss01:
      ansible_host: 10.179.2.140
      gateway: 10.179.2.1

mme:
  hosts:
    customer-mme01:
      ansible_host: 10.179.1.15
      gateway: 10.179.1.1

dns:
  hosts:
    customer-dns01:
      ansible_host: 10.179.2.177
      gateway: 10.179.2.1

# Configuration Globale
all:
  vars:
    # Auto-configuration (aucune configuration manuelle
nécessaire) :
```

```
    # - use_apt_cache: true (auto-activé lorsque apt_cache_servers
existe)
    # - apt_repo.apt_server: "10.179.1.114" (auto-dérivé du
serveur de cache)
```

Ce qui se passe lors du déploiement :

1. **Serveur de cache (`10.179.1.114`) :**
   - Utilise `remote_apt_*` de sa section `vars:`
   - Télécharge tous les paquets depuis `http://customer-username:customer-secure-token@apt.omnitouch.com.au:80/`
   - Sert des paquets sur le port 8080 via nginx
2. **Hôtes d'application (`customer-hss01`, `customer-mme01`, `customer-dns01`) :**
   - Détectent automatiquement que le groupe `apt_cache_servers` existe
   - Définissent automatiquement `use_apt_cache: true`
   - Dérivent automatiquement `apt_repo.apt_server: "10.179.1.114"`
   - Génèrent : `deb [trusted=yes] http://10.179.1.114:8080/noble noble main`
   - Récupèrent tous les paquets depuis le serveur de cache (aucuns identifiants requis)

Mise à Jour du Cache

Pour synchroniser de nouveaux paquets ou mises à jour :

```
ansible-playbook -i hosts/customer/production.yml
services/apt_cache.yml
```

Cela synchronise de manière incrémentielle tout le contenu depuis le serveur APT d'Omnitouch :

- Nouvelles versions de paquets Omnitouch

- Nouveaux paquets Ubuntu et dépendances
- Nouvelles versions GitHub
- Tarballs source mis à jour

La synchronisation utilise `wget --timestamping`, donc les fichiers existants non modifiés sont ignorés, rendant la re-synchronisation rapide.

Remarque : Le serveur APT d'Omnitouch (`apt.omnitouch.com.au`) est la seule source de vérité pour tous les paquets. Exécutez `services/apt.yml` sur le serveur APT en premier pour construire/mettre à jour les paquets, puis exécutez `services/apt_cache.yml` sur les miroirs de cache pour synchroniser.

Dépannage

La Mise à Jour APT Échoue avec 401 Non Autorisé

Symptômes :

```
Échec de la récupération
http://10.179.1.115:80/noble/dists/noble/main/binary-
amd64/Packages  401  Non Autorisé
```

Causes possibles :

- Configuration `apt_repo` définie dans `all: vars:` au lieu de `apt_cache_servers: vars:`
- Hôtes essayant d'accéder directement au dépôt authentifié au lieu du cache
- Identifiant `apt_repo_username` ou `apt_repo_password` incorrect
- IP source non ajoutée à la liste blanche sur le serveur APT d'Omnitouch
- Utilisation des identifiants de cache pour un accès direct ou vice versa

Résolution :

1. **Vérifiez la portée de la configuration** : Assurez-vous que `apt_repo` avec des identifiants est défini dans `apt_cache_servers: vars:`, PAS dans `all: vars:`
2. **Vérifiez le mode cache** : Lors de l'utilisation du cache, les hôtes doivent se connecter au serveur de cache (port 8080), pas au dépôt (port 80)
3. **Vérifiez les sources générées** : Sur l'hôte en échec, vérifiez `/etc/apt/sources.list.d/omnitouch.list`
   - **Correct (mode cache)** : `deb [trusted=yes] http://10.179.1.114:8080/noble noble main`
   - **Incorrect (a des identifiants au mauvais endroit)** : `deb [trusted=yes] http://user:pass@10.179.1.115:80/noble noble main`
4. Vérifiez que les identifiants sont corrects pour votre mode de déploiement
5. Confirmez que votre IP publique est ajoutée à la liste blanche avec Omnitouch (si vous utilisez l'accès direct)

Les Téléchargements Binaires Échouent (Node Exporter, Zabbix, etc.)

Symptômes : Le playbook Ansible échoue à télécharger des fichiers depuis le chemin `/releases/`

Causes possibles :

- Variables `remote_apt_*` non configurées
- Identifiant `remote_apt_user` ou `remote_apt_password` incorrect
- `remote_apt_server` manquant lorsque `use_apt_cache: false`

Résolution :

1. Assurez-vous que toutes les variables `remote_apt_*` sont définies
2. Vérifiez que les identifiants correspondent à ceux fournis par Omnitouch
3. Vérifiez que `remote_apt_server` pointe vers l'hôte correct

Le Serveur de Cache Ne Peut Pas Synchroniser

Symptômes : Le playbook du serveur de cache échoue à télécharger des paquets

Causes possibles :

- Le serveur de cache n'a pas accès à Internet
- Identifiants `remote_apt_*` incorrects
- Pare-feu bloquant les connexions sortantes vers Omnitouch

Résolution :

1. Vérifiez que le serveur de cache peut atteindre `apt.omnitouch.com.au` sur le port 80
2. Vérifiez les identifiants `remote_apt_*`
3. Examinez les règles de pare-feu pour l'accès sortant

Documentation Connexe

- Configuration du Fichier Hosts — Configuration de l'inventaire et des variables
- Référence de Configuration — Référence complète des paramètres
- Architecture de Déploiement — Architecture globale du système
- Déploiement Proxmox — Déploiement du serveur de cache en tant que conteneur LXC

Référence de Configuration

Vue d'ensemble

Ce document fournit une référence complète pour la configuration des déploiements OmniCore via des fichiers d'hôtes. La configuration est principalement définie dans des fichiers d'inventaire d'hôtes avec des remplacements group_vars minimaux nécessaires pour les déploiements modernes.

Pour la documentation spécifique au produit, voir :

- **OmniCore** : https://docs.omnitouch.com.au/docs/repos/OmniCore
- **OmniCall** : https://docs.omnitouch.com.au/docs/repos/OmniCall
- **OmniCharge** : https://docs.omnitouch.com.au/docs/repos/OmniCharge

Approche de Configuration

Les déploiements modernes d'OmniCore utilisent un modèle de configuration simplifié :

Principe Clé : La plupart de la configuration est définie directement dans le fichier d'hôtes. Les valeurs par défaut des rôles gèrent la majorité des paramètres, avec group_vars utilisées uniquement pour des personnalisations spécifiques.

Planification Réseau

Avant de configurer les hôtes, consultez la Norme de Planification IP pour des conseils sur :

- Stratégies de segmentation réseau
- Allocation d'adresses IP
- Organisation des sous-réseaux
- Gestion des IP publiques

Paramètres Communs des Hôtes

#ToDo - Just say to check hosts-file-configuration.md for this

Drapeaux Spécifiques au Service

```
cdrs_enabled: True                # Activer la génération de CDR
in_pool: False                    # Exclure du pool d'équilibrage
de charge
online_charging_enabled: False    # Activer l'intégration OCS
recording: True                   # Activer l'enregistrement des
appels (AS)
populate_crm: False               # Peupler le CRM avec des données
initiales
```

Variables Globales (all:vars)

La section `all:vars` contient des paramètres à l'échelle du déploiement. Les déploiements modernes utilisent des variables globales minimales avec la plupart de la configuration dans les valeurs par défaut des rôles.

Variables Globales Essentielles

Authentification & Accès

```
ansible_connection: ssh
ansible_user: root
ansible_password: password
ansible_become_password: password
```

Alternative : Utilisez des clés SSH au lieu de mots de passe :

```
ansible_ssh_private_key_file: '/path/to/key.pem'
```

Identité du Client

```
customer_name_short: omnitouch
customer_legal_name: "YKTN Lab"
site_name: YKTN
region: AU
TZ: Australia/Melbourne
```

Configuration PLMN

```
plmn_id:
  mcc: '001'               # Code Pays Mobile (3 chiffres)
  mnc: '01'               # Code Réseau Mobile (2-3 chiffres)
  mnc_longform: '001'     # MNC avec zéros en tête (toujours 3
chiffres)

diameter_realm: epc.mnc{{ plmn_id.mnc_longform }}.mcc{{
plmn_id.mcc }}.3gppnetwork.org
```

But : Identifie de manière unique votre réseau mobile. Utilisé pour la construction du domaine Diameter.

Noms de Réseau

```
network_name_short: Omni
network_name_long: Omnitouch
tac_list: [10100,100]            # Liste TAC par défaut (peut être
remplacée par MME)
```

Affiché : Noms de réseau affichés sur les appareils UE dans Paramètres > Réseau Mobile.

Configuration DNS

```
netplan_DNS: False               # Utiliser systemd-resolved au
lieu de DNS netplan
```

Configuration du Dépôt APT

Valeurs par Défaut Automatiques : Lorsqu'un groupe `apt_cache_servers` est défini avec des hôtes :

- `use_apt_cache` par défaut à `True` (sauf s'il est explicitement défini sur `False`)
- `apt_repo.apt_server` par défaut à l'IP du premier serveur de cache

```
# Configuration manuelle (optionnelle si le groupe
apt_cache_servers existe)
use_apt_cache: True              # Utiliser le cache APT local vs
accès direct au dépôt

apt_repo:
  apt_server: "10.10.1.114"      # Serveur de cache APT ou serveur
de dépôt
  # Les identifiants ne sont nécessaires que lorsque
use_apt_cache: False
  # apt_repo_username: "omni"
  # apt_repo_password: "omni"

# Configuration des téléchargements binaires et de la
synchronisation du cache
# Utilisé pour : (1) télécharger des binaires depuis /releases/
lorsque use_apt_cache: false
#           (2) synchronisation du serveur de cache depuis
Omnitouch lorsque use_apt_cache: true
remote_apt_server: "apt.omnitouch.com.au"
remote_apt_user: "omni"
remote_apt_password: "omni"
```

Voir : Système de Cache APT

Serveur de Licences

```
license_server_api_urls: ["https://10.10.2.150:8443/api"]
license_enforced: true
```

Voir : Serveur de Licences

Paramètres MME

```
mme_dns: False                   # Activer la résolution DNS MME
```

Paramètres SAEGW

```
mtu: 1400                          # Unité de Transmission Maximum
```

Paramètres IMS

```
ims_dra_support: False             # Router IMS via DRA
enable_homer: False                # Activer la capture SIP Homer
```

Configuration du Moniteur RAN

```yaml
use_nokia_monitor: True
use_casa_monitor: True
install_influxdb: True

influxdb_user: monitor
influxdb_password: "secure-password"
influxdb_organisation_name: omnitouch
influxdb_nokia_bucket_name: nokia-monitor
influxdb_casa_bucket_name: casa-monitor
influxdb_operator_token: "generated-token"
influxdb_url: http://127.0.0.1:8086

enable_pm_collection: False
enable_alarm_collection: False
enable_location_collection: False
enable_ran_status_collection: True
enable_nokia_rectifier_collection: False
collection_interval_in_seconds: 120

ran_monitor:
  sql:
    user: ran_monitor
    password: "secure-password"
    database_host: 127.0.0.1
    database_name: ran_monitor
  influxdb:
    address: 10.10.2.135
    port: 8086
  nokia:
    airscales:
    - address: 10.7.15.66
      name: site-Lab-Airscale
      port: 8080
      web_password: nemuuser
      web_username: Nemuadmin
```

Configuration du Pare-feu

```yaml
firewall:
  allowed_ssh_subnets:
    - '10.0.1.0/24'
    - '10.0.0.0/24'
  allowed_ue_voice_subnets:
    - '10.0.1.0/24'
  allowed_carrier_voice_subnets:
    - '10.0.1.0/24'
  allowed_signaling_subnets:
    - '10.0.1.0/24'
```

Serveurs DNS de Roaming

```yaml
roaming_dns_servers:
  wildcard: ['10.0.99.1']
  # DNS spécifique à l'opérateur (basé sur PLMN)
  123456:                         # Exemple Opérateur 1
    - '10.10.2.197'
  654321:                         # Exemple Opérateur 2
    - '10.10.0.4'
```

Utilisateurs Locaux (Clés SSH)

```yaml
local_users:
  usera:
    name: Exemple Utilisateur A
    public_key: "ssh-rsa AAAAB3Nza..."
  userb:
    name: Exemple Utilisateur B
    public_key: "ssh-ed25519 AAAAC3..."
```

Configuration de l'Hyperviseur

Proxmox

```yaml
proxmoxServers:
  customer-prxmx01:
    proxmoxServerAddress: 10.10.0.100
    proxmoxServerPort: 8006
    proxmoxRootPassword: password
    proxmoxApiTokenName: AnsibleToken
    proxmoxApiTokenSecret: "token-secret"
    proxmoxTemplateName: ubuntu-24.04-cloud-init-template
    proxmoxTemplateId: 9000
    proxmoxNodeName: pve01

# Paramètres par défaut de Proxmox
proxmoxServerAddress: 10.10.0.100
proxmoxServerPort: 8006
proxmoxNodeName: 'pve01'
proxmoxLxcOsTemplate: 'local:vztmpl/ubuntu-24.04-standard_24.04-2_amd64.tar.zst'
proxmoxApiTokenName: DocsTest
proxmoxLxcCores: 8
proxmoxLxcDiskSizeGb: 20
proxmoxLxcMemoryMb: 64000
proxmoxLxcRootFsStorageName: SSD_RAID0
proxmoxLxcBridgeName: vmbr0
proxmoxTemplateName: "ubuntu-24.04-cloud-init-template"
proxmoxStorage: SSD_RAID0
vLabNetmask: 24
PROXMOX_API_TOKEN: "token-secret"
vlabRootPassword: password
vLabPublicKey: "ssh-rsa AAAAB3..."
mask_cidr: 24
```

VMware vCenter

```yaml
vcenter_ip: "vcenter.example.com"
vcenter_username: "administrator@vsphere.local"
vcenter_password: "password"
vcenter_datacenter: "DC1"
vcenter_vm_template: ubuntu-24.04-model
vcenter_vm_disk_size: 50
vcenter_folder: "Omnicore"
host_vm_network: "Management"

vhosts:
  "10.0.0.23":
    vcenter_cluster_ip: 10.0.0.23
    vcenter_datastore: "datastore1 (3)"

netmask: 255.255.255.0
```

Documentation Connexe

- Norme de Planification IP - Architecture réseau et directives d'allocation IP
- Configuration du Fichier d'Hôtes - Comment structurer les fichiers d'hôtes
- Configuration des Variables de Groupe - Quand et comment utiliser group_vars
- Configuration de Netplan - IP secondaires et configuration multi-NIC
- Architecture de Déploiement - Comment les composants interagissent
- Système de Cache APT - Gestion des paquets
- Serveur de Licences - Configuration des licences

Documentation Produit

Pour des guides opérationnels détaillés et une configuration avancée :

- **Composants OmniCore** : https://docs.omnitouch.com.au/docs/repos/OmniCore

- **Composants OmniCall** : https://docs.omnitouch.com.au/docs/repos/OmniCall
- **OmniCharge/OmniCRM** : https://docs.omnitouch.com.au/docs/repos/OmniCharge

Vue d'ensemble de l'architecture de déploiement

Vue d'ensemble

Ce document fournit une vue complète de la manière dont le logiciel de réseau cellulaire d'Omnitouch Network Services est déployé à l'aide d'Ansible, montrant comment tous les composants s'assemblent pour créer un réseau 4G/5G fonctionnel.

Voir la Norme de planification IP pour des directives détaillées sur le placement des composants, l'attribution des adresses IP et la gestion des IP publiques.

Exemple complet de déploiement

0. Provisionnement de l'infrastructure (optionnel)

Pour les déploiements Proxmox, provisionnez les VMs/LXCs avant la configuration :

```
# Déployer des VMs sur Proxmox
ansible-playbook -i hosts/Customer/hosts.yml services/proxmox.yml

# Ou déployer des conteneurs LXC (laboratoire/test uniquement)
ansible-playbook -i hosts/Customer/hosts.yml
services/proxmox_lxc.yml
```

Voir : Déploiement de VM/LXC Proxmox

1. Définition de l'infrastructure (fichier hosts)

```yaml
# Définir quoi déployer et où
mme:
  hosts:
    customer-mme01:
      ansible_host: 10.10.1.15

hss:
  hosts:
    customer-hss01:
      ansible_host: 10.10.2.140

# ... tous les autres composants
```

Voir : Configuration du fichier hosts

2. Personnalisation (group_vars)

Le dossier `group_vars` est l'endroit où nous pouvons stocker toutes les substitutions de configuration nécessaires au niveau d'un hôte, d'un site ou d'un réseau.

Par exemple, vous auriez un dossier avec votre configuration SMSc OmniMessage, les trunks SIP auxquels votre TAS se connecte seraient ici, toute votre logique de routage Diameter, etc., etc.

Voir : Configuration des variables de groupe

3. Distribution des paquets (cache APT)

```yaml
# Configurer où obtenir les paquets
apt_repo:
  apt_server: "10.254.10.223"  # IP du serveur de cache ou serveur de repo direct
use_apt_cache: false  # true = utiliser le cache local, false = accès direct au repo
```

Voir : Système de cache APT

4. Configuration de la licence

```
# Pointer les composants vers le serveur de licence
license_server_api_urls: ["https://10.10.2.150:8443/api"]
license_enforced: true
```

Voir : Serveur de licence

5. Exécuter le déploiement

Des composants individuels peuvent être déployés en exécutant `services/twag.yml` par exemple, mais le `services/all.yml` gérera tout, et vous pouvez utiliser `--limit=myhost` ou `--limit=mmee,sgw`, etc., pour limiter les hôtes sur lesquels nous travaillons.

```
# Déployer le réseau complet
ansible-playbook -i hosts/customer/host_files/production.yml
services/all.yml

# Ou déployer des composants spécifiques
ansible-playbook -i hosts/customer/host_files/production.yml
services/epc.yml
ansible-playbook -i hosts/customer/host_files/production.yml
services/ims.yml
```

Documentation connexe

- Introduction au déploiement Ansible - Premiers pas
- Configuration du fichier hosts - Définir l'infrastructure
- Norme de planification IP - **Architecture réseau et allocation IP**
- Configuration des variables de groupe - Personnalisation
- Système de cache APT - Gestion des paquets
- Serveur de licence - Gestion des licences

Documentation produit

Pour des informations détaillées sur la configuration de chaque composant :

- **OmniCore** (Noyau de paquets 4G/5G) : https://docs.omnitouch.com.au/docs/repos/OmniCore
  - OmniHSS, OmniSGW, OmniPGW, OmniUPF, OmniDRA, OmniTWAG
- **OmniCall** (Voix et messagerie) : https://docs.omnitouch.com.au/docs/repos/OmniCall
  - OmniTAS, OmniCall CSCF, OmniMessage, OmniSS7, VisualVoicemail
- **OmniCharge/OmniCRM** (Facturation) : https://docs.omnitouch.com.au/docs/repos/OmniCharge
- **Documentation principale** : https://docs.omnitouch.com.au/

Configuration des Variables de Groupe

Vue d'ensemble

Le répertoire `group_vars` est l'endroit où vous stockez des fichiers de configuration personnalisés qui remplacent les modèles par défaut.

C'est ici que résident vos configurations spécifiques au client - trunks SIP, règles de routage Diameter, logique de routage SMS, plans d'appel, et toute autre personnalisation où vous ne souhaitez pas la configuration par défaut - Cela se trouve dans `group_vars`.

Emplacement : `hosts/{Customer}/group_vars/`

Comment ça fonctionne

Les rôles Ansible ont des modèles de configuration par défaut. Pour personnaliser pour un déploiement spécifique, placez vos fichiers personnalisés dans `group_vars` et référencez-les dans votre fichier hosts.

```
Modèle par défaut du rôle → Remplacement group_vars (si spécifié)
→ Configuration déployée
```

Exemple 1 : Modèle de Configuration Personnalisé (OmniMessage)

Certains composants acceptent des modèles de configuration Jinja2 personnalisés.

Structure des Fichiers

```
hosts/Customer/
└── group_vars/
    └── smsc_controller.exs        # Votre modèle de config
personnalisé
```

Référence dans le Fichier Hosts

```yaml
omnimessage:
  hosts:
    customer-smsc-controller01:
      ansible_host: 10.10.3.219
      gateway: 10.10.3.1
      host_vm_network: "vmbr3"
      smsc_template_config: smsc_controller.exs   # Référencez le
nom de votre modèle dans group_vars
```

Ce qui se passe :

1. Ansible trouve `smsc_template_config: smsc_controller.exs`
2. Cherche dans `hosts/Customer/group_vars/smsc_controller.exs`
3. Le modèle avec Jinja2 (peut utiliser `{{ inventory_hostname }}`, `{{ plmn_id.mcc }}`, etc.)
4. Déploie dans `/etc/omnimessage/runtime.exs`
5. Redémarre le service

Sans `smsc_template_config`, le modèle par défaut du rôle est utilisé.

Détails de configuration : Voir https://docs.omnitouch.com.au/docs/repos/OmniCall

---

Exemple 2 : Collections de Fichiers de Configuration (Passerelles &

Plans d'Appel OmniTAS)

Certains composants utilisent des répertoires de fichiers de configuration.

Structure des Fichiers

```
hosts/Customer/
└── group_vars/
    ├── gateways_prod/              # Configurations des
passerelles SIP
    │   ├── gateway_carrier1.xml
    │   ├── gateway_carrier2.xml
    │   └── gateway_emergency.xml
    ├── gateways_lab/               # Passerelles de laboratoire
    │   └── gateway_test.xml
    └── dialplan/                   # Règles de routage d'appels
        ├── mo_dialplan.xml         # Mobile Originate (sortant)
        ├── mt_dialplan.xml         # Mobile Terminate (entrant)
        └── emergency.xml
```

Référence dans le Fichier Hosts

```yaml
applicationserver:
  hosts:
    customer-tas01:
      ansible_host: 10.10.3.60
      gateway: 10.10.3.1
      host_vm_network: "vmbr3"
      gateways_folder: "gateways_prod"   # Référencez votre
dossier de passerelles à utiliser sur cet hôte
```

Ce qui se passe :

1. Ansible trouve `gateways_folder: "gateways_prod"`
2. Copie tous les fichiers de `hosts/Customer/group_vars/gateways_prod/` vers `/etc/freeswitch/sip_profiles/`

3. Copie tous les fichiers de `hosts/Customer/group_vars/dialplan/` vers le répertoire des modèles OmniTAS
4. Les services chargent les configurations

Différents environnements : Utilisez différents dossiers par environnement :

- `gateways_folder: "gateways_lab"`
- `gateways_folder: "gateways_prod"`
- `gateways_folder: "gateways_customer_specific"`

Détails de configuration : Voir https://docs.omnitouch.com.au/docs/repos/OmniCall

---

Exemple 3 : Modèle de Configuration Personnalisé (OmniHSS)

Le Home Subscriber Server accepte des modèles de configuration d'exécution personnalisés.

Structure des Fichiers

```
hosts/Customer/
└── group_vars/
    └── hss_runtime.exs.j2        # Votre modèle de config HSS
personnalisé
```

Référence dans le Fichier Hosts

```
omnihss:
  hosts:
    customer-hss01:
      ansible_host: 10.10.3.50
      gateway: 10.10.3.1
      host_vm_network: "vmbr3"
      hss_template_config: hss_runtime.exs.j2   # Référencez le
nom de votre modèle dans group_vars
```

Ce qui se passe :

1. Ansible trouve `hss_template_config: hss_runtime.exs.j2`
2. Cherche dans `hosts/Customer/group_vars/hss_runtime.exs.j2`
3. Le modèle avec Jinja2 (peut utiliser `{{ inventory_hostname }}`, `{{ plmn_id.mcc }}`, etc.)
4. Déploie dans `/etc/omnihss/runtime.exs`
5. Redémarre le service

Sans `hss_template_config`, le modèle par défaut du rôle est utilisé.

Détails de configuration : Voir https://docs.omnitouch.com.au/docs/repos/OmniCore

---

Exemple 4 : Modèle de Configuration Personnalisé (OmniMME)

L'Entity de Gestion de Mobilité accepte des modèles de configuration d'exécution personnalisés.

Structure des Fichiers

```
hosts/Customer/
└── group_vars/
    └── mme_runtime.exs.j2        # Votre modèle de config MME
personnalisé
```

Référence dans le Fichier Hosts

```yaml
omnimme:
  hosts:
    customer-mme01:
      ansible_host: 10.10.3.51
      gateway: 10.10.3.1
      host_vm_network: "vmbr3"
      mme_template_config: mme_runtime.exs.j2   # Référencez le
nom de votre modèle dans group_vars
```

Ce qui se passe :

1. Ansible trouve `mme_template_config: mme_runtime.exs.j2`
2. Cherche dans `hosts/Customer/group_vars/mme_runtime.exs.j2`
3. Le modèle avec Jinja2 (peut utiliser `{{ inventory_hostname }}`, `{{ plmn_id.mcc }}`, etc.)
4. Déploie dans `/etc/omnimme/runtime.exs`
5. Redémarre le service

Sans `mme_template_config`, le modèle par défaut du rôle est utilisé.

Détails de configuration : Voir https://docs.omnitouch.com.au/docs/repos/OmniCore

---

Exemple de Structure de Répertoire dans le Monde Réel

```
hosts/Customer/
├── host_files/
│   └── production.yml          # Le fichier hosts référence les
fichiers group_vars
└── group_vars/
    ├── smsc_controller.exs     # Modèle personnalisé OmniMessage
    ├── smsc_smpp.exs           # Modèle personnalisé SMPP
OmniMessage
    ├── tas_runtime.exs.j2      # Modèle personnalisé TAS
    ├── hss_runtime.exs.j2      # Modèle personnalisé HSS
    ├── mme_runtime.exs.j2      # Modèle personnalisé MME
    ├── dra_runtime.exs.j2      # Modèle personnalisé DRA
    ├── pgwc_runtime.exs.j2     # Modèle personnalisé PGW
    ├── dea_runtime.exs.j2      # Modèle personnalisé DEA
    ├── upf_config.yaml         # Configuration UPF
    ├── crm_config.yaml         # Configuration CRM
    ├── stp.j2                  # Modèle SS7 STP
    ├── hlr.j2                  # Modèle SS7 HLR
    ├── camel.j2                # Modèle SS7 CAMEL
    ├── ipsmgw.j2               # Modèle IP-SM-GW
    ├── omnicore_smsc_ims.yaml.j2  # Configuration SMSC IMS
    ├── pytap.yaml              # Configuration TAP3
    ├── sip_profiles/           # Passerelles SIP (dossier)
    │   └── gateway_otw.xml
    └── dialplan/               # Règles de routage d'appels
(dossier)
        ├── mo_dialplan.xml     # Mobile Originate
        ├── mt_dialplan.xml     # Mobile Terminate
        └── mo_emergency.xml    # Routage d'urgence
```

Paramètres Communs Qui

Référencent group_vars

Paramètre	Composant	Références
`smsc_template_config`	omnimessage	Fichier de modèle Jinja2 (par exemple, `smsc_controller.exs`)
`smsc_smpp_template_config`	omnimessage_smpp	Fichier de modèle Jinja2 (par exemple, `smsc_smpp.exs`)
`gateways_folder`	applicationserver	Nom du dossier (par exemple, `sip_profiles`)
Plans d'Appel (automatique)	applicationserver	Dossier `dialplan/` des XML de routage
`tas_template_config`	applicationserver	Fichier de modèle Jinja2 (par exemple, `tas_runtime.exs.j2`)
`hss_template_config`	omnihss	Fichier de modèle Jinja2 (par exemple, `hss_runtime.exs.j2`)
`mme_template_config`	omnimme	Fichier de modèle Jinja2 (par exemple, `mme_runtime.exs.j2`)
`dra_template_config`	dra	Fichier de modèle Jinja2 (par exemple, `dra_runtime.exs.j2`)
`pgwc_template_config`	pgwc	Fichier de modèle Jinja2 (par exemple,

Paramètre	Composant	Références
		`pgwc_runtime.exs.j2`)
`frr_template_config`	omniupf	Fichier de modèle Jinja2 (par exemple, `frr.conf.j2`)
Modèles SS7	ss7 (différents rôles)	Fichiers de modèle Jinja2 (par exemple, `stp.j2`, `hlr.j2`, `camel.j2`)
Configurations YAML	Divers composants	Fichiers de config directs (par exemple, `upf_config.yaml`, `crm_config.yaml`)

Points Clés

1. **group_vars contient des personnalisations** - Remplacements pour les configurations par défaut
2. **Référence par nom** - Utilisez des paramètres comme `smsc_template_config` ou `gateways_folder`
3. **Les modèles supportent Jinja2** - Accédez à n'importe quelle variable Ansible avec `{{ variable_name }}`
4. **Les dossiers déploient tout** - Tous les fichiers dans les dossiers référencés sont copiés
5. **Contrôlez tout par version** - Commitez tous les group_vars dans Git

Quand Utiliser group_vars

✅ Utilisez group_vars pour :

- Modèles de configuration de composants personnalisés
- Définitions de passerelles SIP
- Plans d'appel de routage
- Règles de routage Diameter
- Paramètres spécifiques au client qui remplacent les valeurs par défaut

N'utilisez pas group_vars pour :

- Configuration de base des hôtes (IPs, noms d'hôtes) - Utilisez le fichier hosts
- Tests ponctuels - Utilisez des variables spécifiques aux hôtes dans le fichier hosts
- Changements temporaires - Éditez sur la cible, commitez dans group_vars si permanent

---

Documentation Connexe

- Référence de Configuration - Tous les paramètres d'hôte et ce qu'ils font
- Configuration du Fichier Hosts - Comment structurer les fichiers hosts
- **Configuration OmniCall** : https://docs.omnitouch.com.au/docs/repos/OmniCall - Ce qui va dans les fichiers de configuration
- **Configuration OmniCore** : https://docs.omnitouch.com.au/docs/repos/OmniCore - Détails de configuration des composants

Playbooks Utilitaires

Vue d'ensemble

Ce dépôt comprend plusieurs playbooks utilitaires pour la maintenance, la surveillance et les tâches opérationnelles. Ceux-ci complètent les playbooks de déploiement principaux avec des capacités de gestion au quotidien.

Utilitaire de Vérification de Santé

L'utilitaire de Vérification de Santé génère un rapport HTML montrant la santé du système, l'état des services, le temps de disponibilité et les informations de version sur tous les composants d'OmniCore.

S'exécute automatiquement dans le playbook `services/all.yml`.

Utilisation

Exécution Manuelle

```
ansible-playbook -i hosts/customer/host_files/production.yml
util_playbooks/health_check.yml
```

Sortie

Le rapport est généré à `/tmp/health_check_YYYY-MM-DD HH:MM:SS.html`

Ouvrez-le dans n'importe quel navigateur web pour le visualiser.

Contenu du Rapport

Le rapport HTML affiche :

Informations sur l'Hôte

- **Nom de l'hôte et adresse IP**
- **Réseau/Sous-réseau** (à partir de la variable `host_vm_network`, ou N/A si non défini)
- **CPU** (nombre de vCPU)
- **RAM** (mémoire totale et libre)
- **Disque** (espace total et libre de la partition racine avec pourcentage)
- **OS** (distribution et version)

État des Services

- **État du service** (actif/inactif avec indicateurs de couleur)
- **Temps de disponibilité**
- **Informations sur la version/la publication**

Pairs Diameter HSS

- **État de la connexion à la base de données** (connecté/déconnecté)
- **Connexions de pairs Diameter** (IP, hôte d'origine, état)
- Récupéré à partir du point de terminaison des métriques HSS (port 9568)

Autres Utilitaires Communs

Configuration de Base du Système

Rôle Commun (`services/common.yml`)

- Applique la configuration de base du système à tous les hôtes
- Configure les dépôts, les clés SSH, le fuseau horaire, NTP
- Configure le réseau et le renforcement du système
- Exécutez ceci avant de déployer des services

```
ansible-playbook -i hosts/customer/host_files/production.yml
services/common.yml
```

Configuration des Utilisateurs (`services/setup_users.yml`)

- Crée et configure des comptes utilisateurs sur tous les hôtes
- Gère les clés SSH et les privilèges sudo
- Assure une configuration utilisateur cohérente

```
ansible-playbook -i hosts/customer/host_files/production.yml
services/setup_users.yml
```

Redémarrage (`services/reboot.yml`)

- Redémarre gracieusement tous les hôtes ciblés
- Attend que les systèmes reviennent en ligne (délai de 5 minutes)
- Utile après des mises à jour de noyau ou des modifications de configuration

```
ansible-playbook -i hosts/customer/host_files/production.yml
services/reboot.yml
```

Utilitaires Opérationnels

Générateur de Plan IP (`util_playbooks/ip_plan_generator.yml`)

- Génère un rapport HTML des attributions d'adresses IP
- Montre la topologie complète du réseau à partir du fichier des hôtes
- Utile pour la documentation et le dépannage

```
ansible-playbook -i hosts/customer/host_files/production.yml
util_playbooks/ip_plan_generator.yml
```

Sauvegarde HSS (`util_playbooks/hss_backup.yml`)

- Sauvegarde les tables de la base de données HSS

- Copie le dump MySQL sur la machine Ansible locale
- Invite interactive pour le chemin de sauvegarde

```
ansible-playbook -i hosts/customer/host_files/production.yml
util_playbooks/hss_backup.yml
```

Obtenir la Capture Locale (`util_playbooks/getLocalCapture.yml`)

- Récupère les deux fichiers de capture de paquets les plus récents de tous les hôtes
- Récupère les fichiers pcap à partir de `/etc/localcapture/`
- Utile pour le débogage des problèmes de connectivité

```
ansible-playbook -i hosts/customer/host_files/production.yml
util_playbooks/getLocalCapture.yml
```

Mettre à Jour MTU (`util_playbooks/updateMtu.yml`)

- Met à jour les paramètres MTU de l'interface réseau
- Applique les modifications via netplan
- Utile pour la configuration des trames jumbo

```
ansible-playbook -i hosts/customer/host_files/production.yml
util_playbooks/updateMtu.yml
```

Documentation Associée

- README Principal - Vue d'ensemble et démarrage
- Introduction au Déploiement Ansible - Exécution des playbooks
- Configuration du Fichier des Hôtes - Configurez votre inventaire
- Architecture de Déploiement - Vue d'ensemble complète du système
- Système de Cache APT - Gestion des paquets

Configuration du fichier Hosts

Vue d'ensemble

Le fichier hosts (également appelé fichier d'inventaire) est le document de configuration central qui définit l'ensemble de votre déploiement de réseau cellulaire. Il spécifie :

- Quelles fonctions réseau déployer
- Où elles s'exécutent (adresses IP, segments de réseau)
- Comment elles sont configurées (paramètres spécifiques au service)
- Paramètres spécifiques au client (PLMN, identifiants, fonctionnalités)

Emplacement du fichier

Les fichiers hosts sont organisés par client et environnement :

```
services/hosts/
└── Customer_Name/
    └── host_files/
        ├── production.yml
        ├── staging.yml
        └── lab.yml
```

Exemple de structure de fichier Hosts

Voici un exemple simplifié montrant les sections clés :

```yaml
# Composants EPC
mme:
  hosts:
    customer-mme01:
      ansible_host: 10.10.1.15
      gateway: 10.10.1.1
      host_vm_network: "vmbr1"
      mme_code: 1
      network_name_short: Customer
      tac_list: [600, 601, 602]

sgw:
  hosts:
    customer-sgw01:
      ansible_host: 10.10.1.25
      gateway: 10.10.1.1
      cdrs_enabled: true

pgwc:
  hosts:
    customer-pgw01:
      ansible_host: 10.10.1.21
      gateway: 10.10.1.1
      ip_pools:
        - '100.64.16.0/24'

# Composants IMS
pcscf:
  hosts:
    customer-pcscf01:
      ansible_host: 10.10.4.165

# Services de support
license_server:
  hosts:
    customer-licenseserver:
      ansible_host: 10.10.2.150

# Variables globales
all:
  vars:
    ansible_connection: ssh
    ansible_password: password
```

```
    customer_name_short: customer
    plmn_id:
      mcc: '001'
      mnc: '01'
```

Paramètres d'hôte communs

Configuration réseau

Chaque hôte comprend généralement :

```
pcscf:
  hosts:
    customer-pcscf01:
      ansible_host: 10.10.1.15      # Adresse IP pour l'accès SSH
      gateway: 10.10.1.1             # Passerelle par défaut
      host_vm_network: "vmbr1"       # nom de la NIC à utiliser
sur l'hyperviseur
```

Remarque : Pour des conseils sur la planification des adresses IP et les stratégies de segmentation réseau, consultez la Norme de planification IP qui décrit l'architecture recommandée à quatre sous-réseaux pour les déploiements OmniCore.

Utilisateurs de Proxmox : Le paramètre `host_vm_network` spécifie quel pont utiliser. Consultez Déploiement Proxmox VM/LXC pour l'approvisionnement automatisé.

Allocation des ressources VM

Pour les services nécessitant des ressources spécifiques :

```
num_cpus: 4                     # Cœurs de CPU
memory_mb: 8192                 # RAM en mégaoctets
proxmoxLxcDiskSizeGb: 50      # Taille du disque en Go
```

Paramètres spécifiques au service

Chaque fonction réseau a ses propres paramètres. Exemples :

MME :

```
mme_code: 1                    # Identifiant MME (1-255)
mme_gid: 1                     # ID de groupe MME
network_name_short: Customer   # Nom du réseau (affiché sur les
téléphones)
network_name_long: Customer Network
tac_list: [600, 601, 602]     # Codes de zone de suivi
```

PGW :

```
ip_pools:                      # Pools IP pour les abonnés
  - '100.64.16.0/24'
  - '100.64.17.0/24'
combined_CP_UP: false          # Plan de contrôle/plan utilisateur
séparé
```

Pour une explication détaillée de ce que chaque variable contrôle, consultez : Référence de configuration

Serveur d'application :

```
online_charging_enabled: true  # Activer l'intégration OCS
tas_branch: "main"             # Branche logicielle à déployer
gateways_folder: "gateways_prod"  # Configuration du passerelle
SIP
```

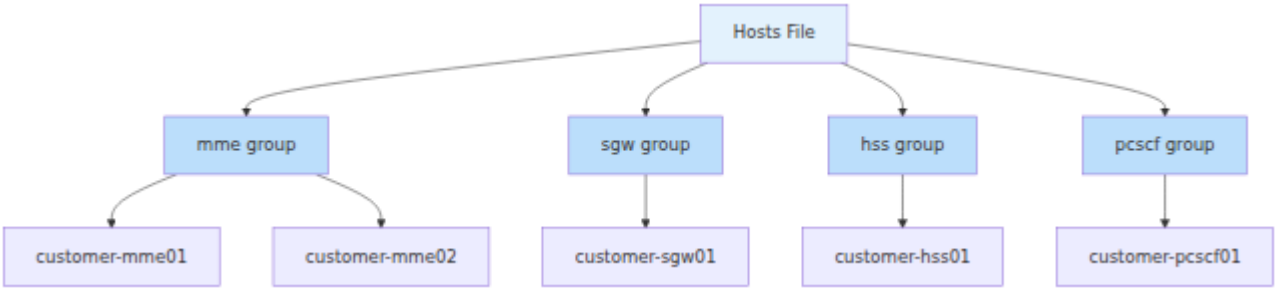
Section des variables globales

La section `all:vars` contient des paramètres qui s'appliquent à l'ensemble du déploiement :

```yaml
all:
  vars:
    # Authentification
    ansible_connection: ssh
    ansible_password: password
    ansible_become_password: password

    # Identité du client
    customer_name_short: customer
    customer_legal_name: "Customer Inc."
    site_name: "Chicago DC1"
    region: US

    # Identifiant PLMN (réseau mobile)
    plmn_id:
      mcc: '001'              # Code de pays mobile
      mnc: '01'               # Code de réseau mobile
      mnc_longform: '001'     # MNC avec zéros en tête

    # Noms de réseau
    network_name_short: Customer
    network_name_long: Customer Network

    # Dépôt APT
    # Remarque : Si le groupe apt_cache_servers est défini avec
des hôtes,
    # use_apt_cache par défaut est vrai et apt_repo.apt_server
    # par défaut est l'adresse IP du premier serveur de cache
automatiquement
    apt_repo:
      apt_server: "10.254.10.223"
      apt_repo_username: "customer"
      apt_repo_password: "secure-password"
    use_apt_cache: false

    # Fuseau horaire
    TZ: America/Chicago
```

Comprendre les groupes d'hôtes

Ansible organise les hôtes en groupes qui correspondent à des rôles :

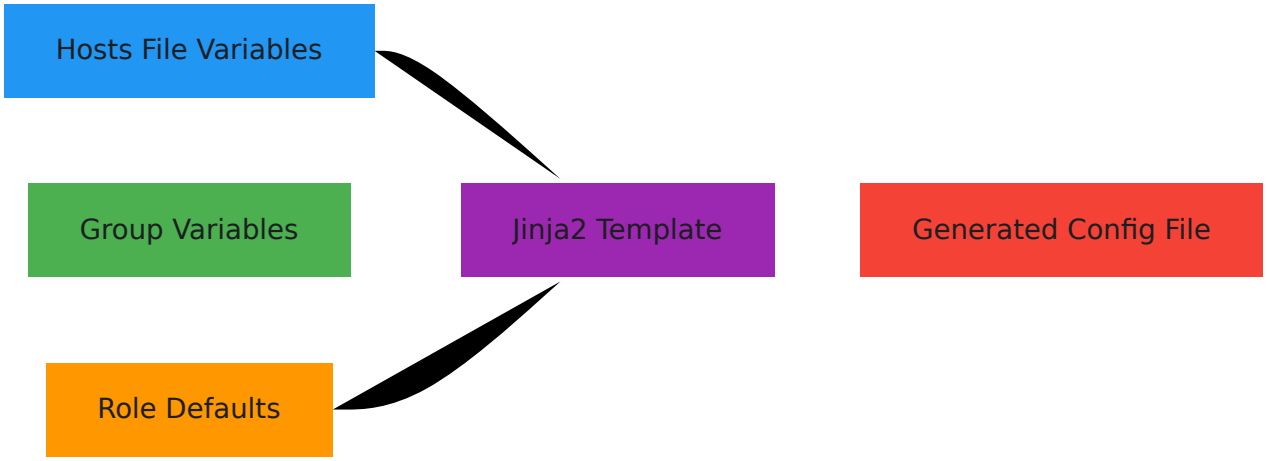


Lorsque vous exécutez un playbook ciblant `mme`, il s'applique à tous les hôtes de la section `mme:hosts:`.

Configuration avec des modèles Jinja2

Ansible utilise **le modèle Jinja2** pour générer des fichiers de configuration à partir des variables définies dans votre fichier hosts et group_vars.

Comment fonctionne Jinja2



Exemple d'utilisation de modèle

Le fichier hosts définit :

```
plmn_id:
  mcc: '001'
  mnc: '01'
customer_name_short: acme
```

Modèle Jinja2 (dans le rôle) :

```
# mme_config.yml.j2
network:
  plmn:
    mcc: {{ plmn_id.mcc }}
    mnc: {{ plmn_id.mnc }}
  operator: {{ customer_name_short }}
  realm: epc.mnc{{ plmn_id.mnc_longform }}.mcc{{ plmn_id.mcc
}}.3gppnetwork.org
```

Fichier de configuration généré :

```
network:
  plmn:
    mcc: 001
    mnc: 01
  operator: acme
  realm: epc.mnc001.mcc001.3gppnetwork.org
```

Modèles Jinja2 courants

Accéder aux variables imbriquées :

```
{{ plmn_id.mcc }}
{{ apt_repo.apt_server }}
```

Logique conditionnelle :

```
{% if online_charging_enabled %}
  charging:
    enabled: true
    ocs_ip: {{ ocs_ip }}
{% endif %}
```

Boucles :

```
tracking_areas:
{% for tac in tac_list %}
  - {{ tac }}
{% endfor %}
```

Formatage :

```
# Zéro-remplir à 3 chiffres
mnc{{ '%03d' | format(plmn_id.mnc|int) }}
```

Surcharge des variables avec group_vars

Alors que le fichier hosts définit les infrastructures et les paramètres spécifiques aux hôtes, `group_vars` peut remplacer les valeurs par défaut pour des groupes d'hôtes.

Voir : Configuration des variables de groupe

Exemple complet de fichier Hosts

Voici un exemple plus complet (avec des données sensibles obscurcies) :

```yaml
# EPC Core
mme:
  hosts:
    customer-mme01:
      ansible_host: 10.10.1.15
      gateway: 10.10.1.1
      host_vm_network: "vmbr1"
      mme_code: 1
      mme_gid: 1
      network_name_short: Customer
      network_name_long: Customer Network
      tac_list: [600, 601, 602, 603]
      omnimme:
        sgw_selection_method: "random_peer"
        pgw_selection_method: "random_peer"

sgw:
  hosts:
    customer-sgw01:
      ansible_host: 10.10.1.25
      gateway: 10.10.1.1
      host_vm_network: "vmbr1"
      cdrs_enabled: true

pgwc:
  hosts:
    customer-pgw01:
      ansible_host: 10.10.1.21
      gateway: 10.10.1.1
      host_vm_network: "vmbr1"
      ip_pools:
        - '100.64.16.0/24'
      combined_CP_UP: false

hss:
  hosts:
    customer-hss01:
      ansible_host: 10.10.2.140
      gateway: 10.10.2.1
      host_vm_network: "vmbr2"

# IMS Core
pcscf:
```

```
  hosts:
    customer-pcscf01:
      ansible_host: 10.10.4.165
      gateway: 10.10.4.1
      host_vm_network: "vmbr4"

icscf:
  hosts:
    customer-icscf01:
      ansible_host: 10.10.3.55
      gateway: 10.10.3.1
      host_vm_network: "vmbr3"

scscf:
  hosts:
    customer-scscf01:
      ansible_host: 10.10.3.45
      gateway: 10.10.3.1
      host_vm_network: "vmbr3"

applicationserver:
  hosts:
    customer-as01:
      ansible_host: 10.10.3.60
      gateway: 10.10.3.1
      host_vm_network: "vmbr3"
      online_charging_enabled: false
      gateways_folder: "gateways_prod"

# Services de support
license_server:
  hosts:
    customer-licenseserver:
      ansible_host: 10.10.2.150
      gateway: 10.10.2.1
      host_vm_network: "vmbr2"

monitoring:
  hosts:
    customer-oam01:
      ansible_host: 10.10.2.135
      gateway: 10.10.2.1
      host_vm_network: "vmbr2"
      num_cpus: 4
```

```yaml
      memory_mb: 8192

dns:
  hosts:
    customer-dns01:
      ansible_host: 10.10.2.177
      gateway: 10.10.2.1
      host_vm_network: "vmbr2"

# Variables globales
all:
  vars:
    ansible_connection: ssh
    ansible_password: password
    ansible_become_password: password

    customer_name_short: customer
    customer_legal_name: "Customer Network Inc."
    site_name: "Primary DC"
    region: US
    TZ: America/Chicago

    # Configuration PLMN
    plmn_id:
      mcc: '001'
      mnc: '01'
      mnc_longform: '001'
    diameter_realm: epc.mnc{{ plmn_id.mnc_longform }}.mcc{{
plmn_id.mcc }}.3gppnetwork.org

    # Noms de réseau
    network_name_short: Customer
    network_name_long: Customer Network
    tac_list: [600, 601]

    # Configuration APT
    apt_repo:
      apt_server: "10.254.10.223"
      apt_repo_username: "customer"
      apt_repo_password: "secure-password"
    use_apt_cache: false

    # Configuration de facturation
    charging:
```

```
      data:
        online_charging:
          enabled: false
      voice:
        online_charging:
          enabled: true
          domain: "mnc{{ plmn_id.mnc_longform }}.mcc{{ plmn_id.mcc
}}.3gppnetwork.org"

    # Règles de pare-feu
    firewall:
      allowed_ssh_subnets:
        - '10.0.0.0/8'
        - '192.168.0.0/16'
      allowed_ue_voice_subnets:
        - '10.0.0.0/8'
      allowed_signaling_subnets:
        - '10.0.0.0/8'

    # Configuration de l'hyperviseur (exemple Proxmox)
    proxmoxServers:
      customer-prxmx01:
        proxmoxServerAddress: 10.10.0.100
        proxmoxServerPort: 8006
        proxmoxApiTokenName: Customer
        proxmoxApiTokenSecret: "token-secret"
        proxmoxTemplateName: ubuntu-24.04-cloud-init-template
        proxmoxNodeName: pve01
```

Voir Déploiement Proxmox VM/LXC pour des détails complets sur la configuration et la mise en place de Proxmox.

Références de documentation produit

Pour une configuration détaillée de chaque composant, consultez la documentation produit officielle :

Composants OmniCore :

- **Documentation OmniCore** : https://docs.omnitouch.com.au/docs/repos/OmniCore
- **OmniHSS** - Serveur d'abonnés domicile
- **OmniSGW** - Passerelle de service (plan de contrôle)
- **OmniPGW** - Passerelle de paquets (plan de contrôle)
- **OmniUPF** - Fonction de plan utilisateur
- **OmniDRA** - Agent de routage Diameter
- **OmniTWAG** - Passerelle d'accès WLAN de confiance

Composants OmniCall :

- **Documentation OmniCall** : https://docs.omnitouch.com.au/docs/repos/OmniCall
- **OmniTAS** - Serveur d'application IMS (VoLTE/VoNR)
- **OmniCall CSCF** - Fonctions de contrôle de session d'appel
- **OmniMessage** - Centre SMS
- **OmniMessage SMPP** - Support du protocole SMPP
- **OmniSS7** - Pile de signalisation SS7
- **VisualVoicemail** - Messagerie vocale

OmniCharge/OmniCRM :

- **Documentation OmniCharge** : https://docs.omnitouch.com.au/docs/repos/OmniCharge

Documentation connexe

- Introduction au déploiement Ansible - Processus de déploiement global
- Référence de configuration - **Guide complet de toutes les variables de configuration**
- Configuration des variables de groupe - Remplacement des configurations par défaut
- Norme de planification IP - **Architecture réseau et directives d'allocation IP**

- Configuration Netplan - **IP secondaires et configuration réseau avancée**
- Système de cache APT - Distribution de paquets
- Serveur de licences - Gestion des licences
- Vue d'ensemble de l'architecture de déploiement - Vue complète du système

Prochaines étapes

1. Créez votre fichier hosts basé sur ce modèle
2. Définissez votre PLMN et votre identité réseau
3. Configurez l'accès au dépôt APT
4. Configurez le serveur de licences
5. Personnalisez avec group_vars si nécessaire
6. Déployez avec des playbooks Ansible

Norme de Planification IP d'OmniCore

Aperçu

Ce document décrit l'approche standard de planification IP pour les déploiements d'OmniCore. L'architecture nécessite **quatre sous-réseaux internes** pour segmenter correctement les fonctions réseau pour la sécurité, la performance et la clarté opérationnelle.

Exigences d'Allocation IP

Allocation Standard : Quatre Sous-Réseaux /24

Chaque déploiement d'OmniCore nécessite quatre sous-réseaux distincts pour le réseau interne :

1. **Réseau de Noyau de Paquet** - Premier /24
2. **Réseau de Signalisation** - Deuxième /24
3. **Réseau Interne IMS** - Troisième /24
4. **Réseau Public UE** - Quatrième /24

Important : Ce sont des Recommandations, Pas des Exigences

L'allocation de sous-réseaux décrite dans ce document est une **meilleure pratique recommandée** pour organiser les déploiements d'OmniCore. Cependant, l'architecture est **complètement flexible** :

- **Tous les hôtes dans un sous-réseau** : Vous pouvez placer tous les composants dans un seul sous-réseau si cela convient à vos besoins de déploiement

- **Chaque type d'hôte dans son propre sous-réseau** : Vous pouvez créer des sous-réseaux séparés pour chaque type de composant (un pour les MMEs, un pour les HSS, etc.)
- **Groupements personnalisés** : Vous pouvez organiser les hôtes dans n'importe quelle structure de sous-réseau qui a du sens pour vos exigences spécifiques
- **Mélanger les IP internes et publiques** : Certains hôtes peuvent utiliser des adresses internes (RFC 1918) tandis que d'autres utilisent des IP publiques, le tout dans le même déploiement

L'approche recommandée des quatre sous-réseaux offre une **isolation de sécurité, une gestion du trafic et une clarté opérationnelle** optimales, c'est pourquoi nous la suggérons pour les déploiements en production. Cependant, vous devez adapter le plan IP pour correspondre à votre topologie réseau spécifique, à l'espace d'adresses disponible et à vos exigences opérationnelles.

Répartition des Segments Réseau

1. Réseau de Noyau de Paquet (Premier /24)

Objectif : Éléments du plan de données utilisateur et du plan de contrôle central

Composants :

- OmniMME (Entité de Gestion de Mobilité)
- OmniSGW (Passerelle de Service)
- OmniPGW-C (Plan de Contrôle de Passerelle PDN)
- OmniUPF/PGW-U (Fonction de Plan Utilisateur / Passerelle PDN Plan Utilisateur)

Exemple : `10.179.1.0/24`

```
mme:
  hosts:
    omni-site-mme01:
      ansible_host: 10.179.1.15
      gateway: 10.179.1.1
      host_vm_network: "vmbr1"
```

---

2. Réseau de Signalisation (Deuxième /24)

Objectif : Signalisation Diameter, politique, facturation et fonctions de gestion

Composants :

- OmniHSS (Serveur d'Abonnés Local)
- OmniCharge OCS (Système de Facturation en Ligne)
- OminiHSS PCRF (Fonction de Règles de Politique et de Facturation)
- OmniDRA DRA (Agent de Routage Diameter)
- Serveurs DNS
- Serveurs TAP3/CDR
- Surveillance/OAM
- Capture SIP
- Serveur de Licences
- Moniteur RAN
- Omnitouch Alerte Lien CBC (Centre de Diffusion de Cellule) - si déployé
- Serveurs de Cache APT - si déployé

Exemple : `10.179.2.0/24`

```
hss:
  hosts:
    omni-site-hss01:
      ansible_host: 10.179.2.140
      gateway: 10.179.2.1
      host_vm_network: "vmbr2"
```

3. Réseau Interne IMS (Troisième /24)

Objectif : Signalisation et services de cœur IMS (signalisation SIP interne)

Composants :

- OmniCSCF S-CSCF (Fonction de Contrôle de Session d'Appel de Service)
- OmniCSCF I-CSCF (Fonction de Contrôle de Session d'Appel Interrogative)
- OmniTAS (Serveur d'Application de Téléphonie / Serveur d'Application)
- OmniMessage (Contrôleur SMS, SMPP, IMS)
- OmniSS7 STP (Point de Transfert de Signalisation SS7)
- OmniSS7 HLR (Registre de Localisation Local) - pour 2G/3G
- OmniSS7 IP-SM-GW (MAP SMSc)
- OmniSS7 Passerelle CAMEL

Exemple : `10.179.3.0/24`

```
scscf:
  hosts:
    omni-site-scscf01:
      ansible_host: 10.179.3.45
      gateway: 10.179.3.1
      host_vm_network: "vmbr3"
```

---

4. Réseau Public UE (Quatrième /24)

Objectif : Services orientés utilisateur tels que IMS et DNS

Composants :

- OmniCSCF P-CSCF (Fonction de Contrôle de Session d'Appel Proxy)
- Serveurs XCAP
- Serveurs de Messagerie Vocale Visuelle
- DNS Client

Exemple : `10.179.4.0/24`

```
pcscf:
  hosts:
    omni-site-pcscf01:
      ansible_host: 10.179.4.165
      gateway: 10.179.4.1
      host_vm_network: "vmbr4"
```

Méthodes de Mise en Œuvre

OmniCore prend en charge deux méthodes principales pour mettre en œuvre cette segmentation réseau :

Méthode 1 : Interfaces Réseau Physiques/Virtualisées (Recommandé pour la Production)

Utilisez des NIC séparés ou des ponts virtuels pour chaque segment réseau. Cela offre la plus forte isolation et est l'approche recommandée pour les déploiements en production.

Exemple :

```yaml
# Noyau de Paquet - vmbr1
mme:
  hosts:
    omni-lab07-mme01:
      ansible_host: 10.179.1.15
      gateway: 10.179.1.1
      host_vm_network: "vmbr1"

# Signalisation - vmbr2
hss:
  hosts:
    omni-lab07-hss01:
      ansible_host: 10.179.2.140
      gateway: 10.179.2.1
      host_vm_network: "vmbr2"

# IMS Interne - vmbr3
icscf:
  hosts:
    omni-lab07-icscf01:
      ansible_host: 10.179.3.55
      gateway: 10.179.3.1
      host_vm_network: "vmbr3"

# UE Public - vmbr4
pcscf:
  hosts:
    omni-lab07-pcscf01:
      ansible_host: 10.179.4.165
      gateway: 10.179.4.1
      host_vm_network: "vmbr4"
```

Méthode 2 : Segmentation Basée sur VLAN

Utilisez une seule interface physique avec un balisage VLAN pour séparer les réseaux. Cela convient aux déploiements plus petits ou lorsque les NIC physiques sont limitées.

Exemple :

```yaml
# Tous les composants utilisent vmbr12 avec différents VLAN
applicationserver:
  hosts:
    ons-lab08sbc01:
      ansible_host: 10.178.2.213
      gateway: 10.178.2.1
      host_vm_network: "ovsbr1"
      vlanid: "402"

dra:
  hosts:
    ons-lab08dra01:
      ansible_host: 10.178.2.211
      gateway: 10.178.2.1
      host_vm_network: "ovsbr1"
      vlanid: "402"

dns:
  hosts:
    ons-lab08dns01:
      ansible_host: 10.178.2.178
      gateway: 10.178.2.1
      host_vm_network: "ovsbr1"
      vlanid: "402"
```

Configuration Réseau :

- Configurez les VLAN sur le commutateur physique
- Taguer le trafic de manière appropriée au niveau de l'hyperviseur
- Router entre les VLAN au niveau de la passerelle/firewall

Exemple de Mappage VLAN :

```
VLAN 10 : 10.x.1.0/24 (Noyau de Paquet)
VLAN 20 : 10.x.2.0/24 (Signalisation)
VLAN 30 : 10.x.3.0/24 (IMS Interne)
VLAN 40 : 10.x.4.0/24 (UE Public)
```

Travailler avec des Adresses IP Publiques

Aperçu

De nombreux déploiements d'OmniCore nécessitent que certains composants aient des adresses IP publiques pour la connectivité externe, telles que :

- **DRA** - Pour la signalisation diameter en itinérance avec des opérateurs externes
- **SGW/PGW en itinérance** - Pour le trafic GTP des partenaires d'itinérance
- **ePDG** - Pour les appels WiFi (tunnels IPsec des UE)
- **Passerelle SMSC** - Pour les connexions SMPP aux agrégateurs SMS externes
- **P-CSCF** (dans certains déploiements) - Pour l'enregistrement SIP direct des UE

Comment Assigner des IP Publiques

Les IP publiques sont gérées **exactement de la même manière que les IP internes** dans vos fichiers d'inventaire d'hôtes. Il suffit de spécifier l'adresse IP publique dans le champ `ansible_host` avec la passerelle et le masque de sous-réseau appropriés.

Exemple : SGW/PGW en itinérance avec des IP Publiques

```yaml
sgw:
  hosts:
    # SGWs internes sur réseau privé
    opt-site-sgw01:
      ansible_host: 10.4.1.25
      gateway: 10.4.1.1
      host_vm_network: "v400-omni-packet-core"

    # SGWs en itinérance avec des IP publiques
    opt-site-roaming-sgw01:
      ansible_host: 203.0.113.10
      gateway: 203.0.113.9
      netmask: 255.255.255.248       # sous-réseau /29
      host_vm_network: "498-public-servers"
      in_pool: False
      cdrs_enabled: True

smf:  # PGWs
  hosts:
    # PGW en itinérance avec IP publique
    opt-site-roaming-pgw01:
      ansible_host: 203.0.113.20
      gateway: 203.0.113.17
      netmask: 255.255.255.240       # sous-réseau /28
      host_vm_network: "497-public-services-LTE"
      in_pool: False
      ip_pools:
        - '100.64.24.0/22'
```

Exemple : DRA avec IP Publique

```yaml
dra:
  hosts:
    opt-site-dra01:
      ansible_host: 198.51.100.50
      gateway: 198.51.100.49
      netmask: 255.255.255.240       # sous-réseau /28
      host_vm_network: "497-public-services-LTE"
```

Exemple : ePDG avec IP Publique

```yaml
epdg:
  hosts:
    opt-site-epdg01:
      ansible_host: 198.51.100.51
      gateway: 198.51.100.49
      netmask: 255.255.255.240       # sous-réseau /28
      host_vm_network: "497-public-services-LTE"
```

Mélanger des IP Internes et Publiques

Il est courant d'avoir un mélange d'IP internes et publiques au sein du même groupe de composants. Par exemple :

- SGWs internes pour des sites locaux utilisant GTP
- SGWs publiques spécifiquement pour le trafic d'itinérance des opérateurs externes
- Le même PGW-C peut gérer à la fois des SGWs internes et externes

L'architecture d'OmniCore gère cela sans problème - il suffit de configurer chaque hôte avec son adressage IP approprié.

Serveur de Licence

Vue d'ensemble

Le Serveur de Licence gère l'activation des fonctionnalités pour tous les composants Omnitouch. Chaque composant valide sa licence au démarrage et périodiquement pendant son fonctionnement.

Configuration

1. Définir dans le Fichier Hosts

```
license_server:
  hosts:
    customer-licenseserver:
      ansible_host: 10.10.2.150
      gateway: 10.10.2.1
      host_vm_network: "vmbr2"

all:
  vars:
    customer_legal_name: "Nom du Client"
    license_server_api_urls: ["https://10.10.2.150:8443/api"]
    license_enforced: true
```

2. Fournir le Fichier de Licence

Placez `license.json` (fourni par Omnitouch) dans `hosts/Customer/group_vars/`

3. Déployer

```
ansible-playbook -i hosts/customer/host_files/production.yml
services/license_server.yml
```

Vous pouvez vérifier l'état de toutes les licences en accédant à https://license_server .

Exigences Réseau

Configuration du Pare-feu

Les pare-feu du site client doivent être configurés pour autoriser le trafic HTTPS (port 443) vers les serveurs de validation de licence Omnitouch suivants :

Nom d'hôte	Adresse IP	Objectif
time.omnitouch.com.au	160.22.43.18	Serveur de validation de licence 1
time.omnitouch.com.au	160.22.43.66	Serveur de validation de licence 2
time.omnitouch.com.au	160.22.43.114	Serveur de validation de licence 3

Règles sortantes requises :

- Protocole : HTTPS (TCP/443)
- Destination : 160.22.43.18, 160.22.43.66, 160.22.43.114
- Direction : Sortante

Exigences DNS

Le serveur de licence nécessite une résolution DNS fonctionnelle pour communiquer avec l'infrastructure de validation de licence Omnitouch.

Configuration DNS requise :

- Le serveur de licence doit avoir accès à des serveurs DNS publics
- Configurez DNS pour utiliser l'un des suivants :
  - 1.1.1.1 (Cloudflare - prend en charge DNS sécurisé)
  - 8.8.8.8 (Google Public DNS)
- Ne pas utiliser de serveurs DNS internes/corporatifs pour le serveur de licence

Remarque : Les serveurs de licence Omnitouch utilisent DNS sécurisé (DoH/DoT). L'utilisation de serveurs DNS publics garantit une validation DNSSEC appropriée et empêche les problèmes d'interception DNS par des appareils de sécurité.

Documentation Connexe

- Référence de Configuration
- Configuration du Fichier Hosts

Configuration de Netplan

Vue d'ensemble

OmniCore peut configurer automatiquement les interfaces réseau sur les VMs déployées en utilisant netplan. Cela est utile pour :

- Configurer l'interface de gestion principale (eth0)
- Ajouter des interfaces secondaires pour des IP publiques, des connexions de peering ou du trafic dédié
- Configurer des routes statiques pour des destinations spécifiques

Activation de la Configuration Netplan

Pour activer la configuration automatique de netplan pour un hôte, ajoutez la variable `netplan_config` pointant vers un modèle Jinja2 dans votre dossier `group_vars` :

```
dra:
  hosts:
    <hostname>:
      ansible_host: 10.0.1.100
      gateway: 10.0.1.1
      netplan_config: netplan.yaml.j2
```

Le modèle sera récupéré depuis `hosts/<customer>/group_vars/netplan.yaml.j2`.

Référence du Modèle

Voici le modèle complet `netplan.yaml.j2` avec des commentaires expliquant chaque section :

```
network:
  version: 2
  ethernets:
    # Interface principale - utilise ansible_host et gateway de
l'inventaire
    eth0:
      addresses:
      - "{{ ansible_host }}/{{ mask_cidr | default(24) }}"
      nameservers:
        addresses:
{% if 'dns' in group_names %}
        # Si cet hôte EST un serveur DNS, utilisez un DNS externe
pour éviter une dépendance circulaire
        - 8.8.8.8
{% else %}
        # Sinon, utilisez les serveurs DNS du groupe 'dns' dans
l'inventaire
{% for dns_host in groups['dns'] | default([]) %}
        - {{ hostvars[dns_host]['ansible_host'] }}
{% endfor %}
{% endif %}
        search:
        - slice
      routes:
      - to: "default"
        via: "{{ gateway }}"

{% if secondary_ips is defined %}
    # Interfaces secondaires - boucle à travers le dictionnaire
secondary_ips de l'inventaire
    # Nommage des interfaces : ens19, ens20, ens21... (18 +
loop.index)
{% for nic_name, nic_config in secondary_ips.items() %}
    ens{{ 18 + loop.index }}:
      addresses:
      - "{{ nic_config.ip_address }}/{{ mask_cidr | default(24)
}}"
{% if nic_config.routes is defined %}
      # Routes statiques pour cette interface - chaque route
utilise la passerelle de cette interface
      routes:
{% for route in nic_config.routes %}
      - to: "{{ route }}"
```

```
        via: "{{ nic_config.gateway }}"
{% endfor %}
{% endif %}
{% endfor %}
{% endif %}
```

Points clés :

- `ansible_host` et `gateway` proviennent de l'entrée d'inventaire de l'hôte
- Les serveurs DNS sont extraits dynamiquement des hôtes dans le groupe `dns`
- Les interfaces secondaires sont nommées `ens19`, `ens20`, etc. pour correspondre à la nomenclature des NIC Proxmox
- Chaque IP secondaire peut avoir sa propre passerelle et des routes statiques

Configuration de l'Interface Principale

L'interface principale (eth0) est configurée automatiquement en utilisant :

- `ansible_host` - L'adresse IP
- `gateway` - La passerelle par défaut
- `mask_cidr` - Masque réseau (par défaut 24)

Les serveurs DNS sont automatiquement définis sur :

- Hôtes dans le groupe `dns` (utilise leurs IP `ansible_host`)
- Revertit à `8.8.8.8` si l'hôte est lui-même un serveur DNS

Interfaces Secondaires

Pour les hôtes nécessitant des interfaces réseau supplémentaires (IP publiques, peering, etc.), utilisez la configuration `secondary_ips`.

Schéma

```
secondary_ips:
  <logical_name>:
    ip_address: <ip_address>
    gateway: <gateway_ip>
    host_vm_network: <proxmox_bridge>
    vlanid: <vlan_id>
    routes:                    # Optionnel - routes statiques via
cette interface
      - '<destination_cidr>'
      - '<destination_cidr>'
```

Nommage des Interfaces

Les interfaces secondaires sont automatiquement nommées en utilisant le schéma de nommage prévisible d'Ubuntu :

- Première interface secondaire : `ens19`
- Deuxième interface secondaire : `ens20`
- Troisième interface secondaire : `ens21`
- Et ainsi de suite...

Cela correspond aux noms d'interface attribués par Proxmox lors de l'ajout de NIC supplémentaires à une VM.

Exemple de Configuration

```yaml
dra:
  hosts:
    <hostname>:
      ansible_host: 10.0.1.100
      gateway: 10.0.1.1
      host_vm_network: "ovsbr1"
      vlanid: "100"
      netplan_config: netplan.yaml.j2
      secondary_ips:
        public_ip:
          ip_address: 192.0.2.50
          gateway: 192.0.2.1
          host_vm_network: "vmbr0"
          vlanid: "200"
          routes:
            - '198.51.100.0/24'
            - '203.0.113.0/24'
        peering_ip:
          ip_address: 172.16.50.10
          gateway: 172.16.50.1
          host_vm_network: "ovsbr2"
          vlanid: "300"
          routes:
            - '172.17.0.0/16'
```

Sortie Netplan Générée

La configuration ci-dessus génère :

```yaml
network:
  version: 2
  ethernets:
    eth0:
      addresses:
      - "10.0.1.100/24"
      nameservers:
        addresses:
        - 10.0.1.53
        search:
        - slice
      routes:
      - to: "default"
        via: "10.0.1.1"
    ens19:
      addresses:
      - "192.0.2.50/24"
      routes:
      - to: "198.51.100.0/24"
        via: "192.0.2.1"
      - to: "203.0.113.0/24"
        via: "192.0.2.1"
    ens20:
      addresses:
      - "172.16.50.10/24"
      routes:
      - to: "172.17.0.0/16"
        via: "172.16.50.1"
```

Intégration Proxmox

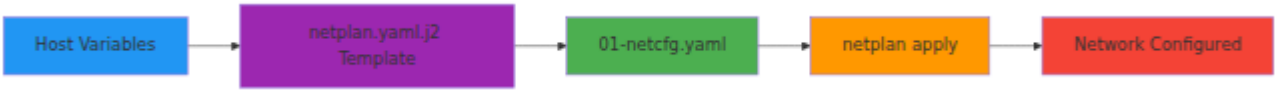
Lors de l'utilisation du playbook `proxmox.yml`, les NIC secondaires sont automatiquement créées sur la VM :

1. **Nouvelles VMs** : Les NIC secondaires sont ajoutées lors du provisionnement initial
2. **VMs Existantes** : Les NIC secondaires sont ajoutées et la VM est redémarrée pour appliquer les changements

La configuration Proxmox utilise :

- `host_vm_network` - Le pont auquel attacher la NIC
- `vlanid` - Tag VLAN pour l'interface

Comment Ça Marche



1. Les variables du fichier d'hôtes sont passées au modèle Jinja2
2. Le modèle est rendu dans `/etc/netplan/01-netcfg.yaml`
3. Toute configuration netplan existante est supprimée pour éviter les conflits
4. `netplan apply` active la configuration
5. Les adresses IP sont vérifiées avec `ip addr show`

Cas d'utilisation Courants

Diameter Edge Agent (DEA) avec IP Publique

```
<hostname>:
  ansible_host: 10.0.1.100           # IP de gestion interne
  gateway: 10.0.1.1
  netplan_config: netplan.yaml.j2
  secondary_ips:
    diameter_roaming:
      ip_address: 192.0.2.50         # IP publique pour les
partenaires de roaming
      gateway: 192.0.2.1
      host_vm_network: "vmbr0"
      vlanid: "200"
      routes:
        - '198.51.100.0/24'          # Réseau des partenaires de
roaming
```

PGW avec Interface S5/S8

```
<hostname>:
  ansible_host: 10.0.2.20            # IP interne
  gateway: 10.0.2.1
  netplan_config: netplan.yaml.j2
  secondary_ips:
    s5s8_interface:
      ip_address: 203.0.113.17       # IP publique S5/S8
      gateway: 203.0.113.1
      host_vm_network: "vmbr0"
      vlanid: "50"
```

Serveur Multi-hébergé avec Réseaux de Gestion et de Données Séparés

```
<hostname>:
  ansible_host: 10.0.1.100           # Réseau de gestion
  gateway: 10.0.1.1
  netplan_config: netplan.yaml.j2
  secondary_ips:
    data_network:
      ip_address: 10.0.2.100         # Réseau de données
      gateway: 10.0.2.1
      host_vm_network: "ovsbr2"
      vlanid: "200"
    backup_network:
      ip_address: 10.0.3.100         # Réseau de sauvegarde
      gateway: 10.0.3.1
      host_vm_network: "ovsbr3"
      vlanid: "300"
```

Référencement des IPs Secondaires dans les Modèles

Vous pouvez référencer les adresses IP secondaires dans d'autres modèles Jinja2 et fichiers de configuration.

Sur le Même Hôte

Lors de la configuration d'un service sur le même hôte qui a des IP secondaires, vous pouvez référencer directement ou utiliser `inventory_hostname` :

```
# Référence directe (la plus simple)
{{ secondary_ips.diameter_public_ip.ip_address }}

# Ou explicitement via inventory_hostname (même résultat)
{{ hostvars[inventory_hostname]['secondary_ips']
['diameter_public_ip']['ip_address'] }}

# Accéder à d'autres propriétés
{{ secondary_ips.diameter_public_ip.gateway }}
{{ secondary_ips.diameter_public_ip.vlanid }}
```

D'un Autre Hôte

Lorsque vous devez référencer une IP secondaire d'un *autre* hôte (par exemple, configurer une connexion de pair), utilisez `hostvars` avec le nom d'hôte cible :

```
# Référence du premier hôte dans le groupe dra
{{ hostvars[groups['dra'][0]]['secondary_ips']
['diameter_public_ip']['ip_address'] }}

# Boucle à travers tous les hôtes DRA et obtenez leurs IP
publiques
{% for host in groups['dra'] %}
{% if hostvars[host]['secondary_ips'] is defined %}
  - {{ hostvars[host]['secondary_ips']['diameter_public_ip']
['ip_address'] }}
{% endif %}
{% endfor %}
```

Exemple : Configuration de Pair DRA

Configurer un pair de diamètre pour se lier à sa propre IP publique :

```
# Dans dra_config.yaml.j2 - utilisez inventory_hostname pour
l'hôte actuel
peers:
  - name: external_peer
    # Lier à l'IP publique de diamètre de cet hôte
    local_ip: {{ hostvars[inventory_hostname]['secondary_ips']
['diameter_public_ip']['ip_address'] }}
    remote_ip: 198.51.100.50
    port: 3868
```

Vérification de l'Existence des IPs Secondaires

Vérifiez toujours si la variable existe avant de l'utiliser :

```
{% if secondary_ips is defined and
secondary_ips.diameter_public_ip is defined %}
public_ip: {{ secondary_ips.diameter_public_ip.ip_address }}
{% else %}
public_ip: {{ ansible_host }}
{% endif %}
```

Dépannage

Vérifier les Noms des Interfaces

SSH sur la VM et vérifiez les noms des interfaces :

```
ip link show
```

Sortie attendue pour une VM avec deux interfaces secondaires :

```
1: lo: <LOOPBACK,UP,LOWER_UP> ...
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> ...
3: ens19: <BROADCAST,MULTICAST,UP,LOWER_UP> ...
4: ens20: <BROADCAST,MULTICAST,UP,LOWER_UP> ...
```

Vérifier la Configuration de Netplan

```
cat /etc/netplan/01-netcfg.yaml
```

Appliquer Netplan Manuellement

```
netplan apply
```

Déboguer Netplan

```
netplan --debug apply
```

Vérifier les Routes

```
ip route show
```

Documentation Connexe

- Configuration du Fichier Hosts - Configuration de l'inventaire des hôtes
- Déploiement Proxmox VM/LXC - Provisionnement de VM
- Référence de Configuration - Toutes les variables de configuration

Déploiement de VM/LXC Proxmox

La majorité de nos clients exécutent la pile OmniCore sur Proxmox, ce guide explique en détail comment utiliser les playbooks `proxmox` pour configurer leur environnement en utilisant Proxmox.

Nous continuons à prendre en charge VMware, HyperV et le cloud (actuellement Vultr / AWS / GCP) pour les déploiements.

Voir aussi :

- Configuration du fichier Hosts - Définir les VMs à déployer
- Norme de planification IP - Directives d'attribution d'adresses IP
- Configuration de Netplan - IP secondaires et configuration multi-NIC
- Architecture de déploiement - Flux de travail complet de déploiement

LXC vs VM

Conteneurs LXC :

- Léger, partage le noyau de l'hôte
- Démarrage rapide, faible surcharge
- Isolation limitée
- Ne peut pas exécuter de noyaux ou de modules de noyau personnalisés
- **Pas adapté aux déploiements en production**
- **Ne peut pas exécuter UPF** (nécessite des modules de noyau / dispositifs TUN)

VMs (KVM) :

- Virtualisation complète avec noyau dédié
- Isolation complète
- Peut exécuter des modules de noyau et un réseau personnalisé

- Surcharge de ressources plus élevée
- **Recommandé pour la production**
- **Nécessaire pour les déploiements UPF**

Cas d'utilisation :

- **VMs** : Sites de production, UPF, toutes les fonctions réseau
- **LXC** : Environnements de laboratoire/test, services légers (apt-cache, surveillance)

Configuration de Proxmox

1. Créer un jeton API

```
# Dans l'UI Proxmox : Datacenter → Permissions → Jetons API
# Créer un jeton : root@pam!<TokenName>
# Copier le secret du jeton (affiché une seule fois)
```

2. Créer un modèle de VM Cloud-Init (pour les VMs uniquement)

Exécutez ce script sur l'hôte Proxmox. Il télécharge l'image cloud d'Ubuntu et crée un modèle avec les informations d'identification utilisateur cloud-init.

```
#!/bin/bash
set -e

TEMPLATE_ID=9000
IMAGE_URL="https://cloud-images.ubuntu.com/noble/current/noble-
server-cloudimg-amd64.img"
IMAGE="noble-server-cloudimg-amd64.img"

echo "=== Téléchargement de l'image cloud d'Ubuntu ==="
cd /var/lib/vz/template/iso
wget -N "$IMAGE_URL"

echo "=== Nettoyage de l'ancien modèle ==="
qm destroy $TEMPLATE_ID --purge 2>/dev/null || true

echo "=== Activation du stockage des extraits ==="
pvesm set local --content images,vztmpl,iso,backup,snippets

echo "=== Création des données utilisateur cloud-init ==="
mkdir -p /var/lib/vz/snippets
cat > /var/lib/vz/snippets/user-data.yml << 'USERDATA'
#cloud-config
ssh_pwauth: true
users:
  - name: omnitouch
    plain_text_passwd: password
    lock_passwd: false
    shell: /bin/bash
    sudo: ALL=(ALL) NOPASSWD:ALL
    groups: sudo
USERDATA

echo "=== Création du modèle de VM ==="
qm create $TEMPLATE_ID --name ubuntu-2404-template --memory 2048 -
-cores 2 --net0 virtio,bridge=vmbr0
qm importdisk $TEMPLATE_ID $IMAGE local-lvm
qm set $TEMPLATE_ID --scsihw virtio-scsi-pci --scsi0 local-
lvm:vm-${TEMPLATE_ID}-disk-0
qm set $TEMPLATE_ID --ide2 local-lvm:cloudinit
qm set $TEMPLATE_ID --boot c --bootdisk scsi0
qm set $TEMPLATE_ID --vga std
qm set $TEMPLATE_ID --agent enabled=1
qm set $TEMPLATE_ID --cicustom user=local:snippets/user-data.yml
```

```
qm template $TEMPLATE_ID

echo "=== Modèle $TEMPLATE_ID créé avec succès ==="
```

Remarques :

- Le modèle fournit un accès de secours : `omnitouch` / `password` (pour l'accès console si cloud-init échoue)
- Lors du clonage via Ansible, les informations d'identification sont remplacées par `local_users` dans votre fichier hosts :
  - Nom d'utilisateur : Clé du premier utilisateur de `local_users`
  - Mot de passe : Champ `password` du premier utilisateur (par défaut 'password' si non défini)
  - Clé SSH : Champ `public_key` du premier utilisateur
- `--vga std` garantit que la console web Proxmox fonctionne
- Le drapeau `-N` sur wget ne télécharge que si le fichier est plus récent que la copie locale

Alternative : Modèle manuel à partir d'un ISO

Si les images cloud ne sont pas disponibles ou si vous avez besoin d'une installation personnalisée :

Étape 1 : Créer une VM via l'UI Web

- Créer une nouvelle VM → ID VM 9000, Nom : ubuntu-2404-template
- OS : Télécharger l'ISO d'Ubuntu Server ou utiliser un ISO existant
- Système : Par défaut (Contrôleur SCSI : VirtIO SCSI)
- Disques : 32 Go, Bus : SCSI
- CPU : 2 cœurs
- Mémoire : 2048 Mo
- Réseau : VirtIO, pont vmbr0
- Démarrer la VM et installer Ubuntu Server

Étape 2 : À l'intérieur de la VM - Nettoyer et préparer

```
# Installer cloud-init
sudo apt update
sudo apt install cloud-init qemu-guest-agent -y

# Nettoyer les données spécifiques à la machine
sudo cloud-init clean
sudo rm -f /etc/machine-id /var/lib/dbus/machine-id
sudo rm -f /etc/ssh/ssh_host_*
sudo truncate -s 0 /etc/hostname
sudo truncate -s 0 /etc/hosts

# Effacer l'historique bash et éteindre
history -c
sudo poweroff
```

Étape 3 : Ajouter Cloud-Init et convertir en modèle

- Sélectionner VM → Matériel → Ajouter → Disque CloudInit (sélectionner le stockage par exemple, local-lvm)
- Cloud-Init → Utilisateur : `omnitouch`, Mot de passe : `password`
- Matériel → Options → Agent invité QEMU → Activer
- Clic droit sur la VM → Convertir en modèle

3. Télécharger le modèle LXC (pour LXC uniquement)

```
# Dans le shell du nœud Proxmox :
pveam update
pveam download local ubuntu-24.04-standard_24.04-2_amd64.tar.zst
```

Configuration du fichier Hosts

Pour le déploiement de VM (proxmox.yml)

```
all:
  vars:
    proxmoxServers:
      pve-node-01:
        proxmoxServerAddress: 192.168.1.100
        proxmoxServerPort: 8006
        proxmoxRootPassword: YourPassword
        proxmoxApiTokenName: ansible
        proxmoxApiTokenSecret: "your-token-secret-uuid"
        proxmoxTemplateName: ubuntu-2404-template
        proxmoxTemplateId: 9000
        proxmoxNodeName: pve-node-01
        storage: local-lvm  # optionnel
      pve-node-02:
        # ... configuration du deuxième nœud

    # Informations d'identification utilisateur - le premier
utilisateur est utilisé pour cloud-init de la VM
    local_users:
      admin_user:
        name: Admin User
        public_key: "ssh-rsa AAAA..."
        password: "optional-password"  # par défaut 'password' si
non défini

mme:
  hosts:
    site-mme01:
      ansible_host: 192.168.1.10
      gateway: 192.168.1.1
      vlanid: "100"  # optionnel
```

Pour le déploiement de LXC (proxmox_lxc.yml)

```yaml
all:
  vars:
    proxmoxServerAddress: 192.168.1.100
    proxmoxServerPort: 8006
    proxmoxNodeName: ['pve-node-01', 'pve-node-02']  # unique ou
liste
    proxmoxApiTokenName: ansible
    PROXMOX_API_TOKEN: "your-token-secret-uuid"
    proxmoxLxcOsTemplate: 'local:vztmpl/ubuntu-24.04-
standard_24.04-2_amd64.tar.zst'
    proxmoxLxcCores: 2
    proxmoxLxcMemoryMb: 4096
    proxmoxLxcDiskSizeGb: 30
    proxmoxLxcRootFsStorageName: local-lvm
    mask_cidr: 24
    host_vm_network: vmbr0

    # Informations d'identification utilisateur - le premier
utilisateur est utilisé pour l'accès initial à la VM/LXC
    local_users:
      admin_user:
        name: Admin User
        public_key: "ssh-rsa AAAA..."
        password: "optional-password"  # par défaut 'password' si
non défini

apt_cache_servers:
  hosts:
    site-cache:
      ansible_host: 192.168.1.20
      gateway: 192.168.1.1
      vlanid: "100"  # optionnel
      proxmoxLxcDiskSizeGb: 120  # remplacement par hôte
```

Utilisation

Déployer des VMs

```
ansible-playbook -i hosts/Customer/hosts.yml services/proxmox.yml
```

Déployer des conteneurs LXC

```
ansible-playbook -i hosts/Customer/hosts.yml
services/proxmox_lxc.yml
```

Supprimer des VMs/LXCs

```
ansible-playbook -i hosts/Customer/hosts.yml
services/proxmox_delete.yml
```

Comportement

proxmox.yml

- Vérifie si une VM avec le même nom existe déjà dans Proxmox
- Distribue les VMs sur les nœuds en utilisant un round-robin
- Clone à partir du modèle
- Configure l'IP statique, les balises et cloud-init
- **Définit les informations d'identification de l'utilisateur cloud-init à partir de la première entrée `local_users`**
- Prend en charge le balisage VLAN

proxmox_lxc.yml

- Vérifie que le conteneur n'existe pas par nom ou IP

- Distribue les LXCs sur les nœuds en utilisant un round-robin
- Crée un conteneur avec une IP statique
- **Crée automatiquement le premier compte `local_users` avec accès sudo et clé SSH**
- Configure netplan pour le réseau
- Démarre automatiquement les conteneurs
- Exclut les hôtes UPF

proxmox_delete.yml

- Arrête et supprime la VM/LXC correspondant au nom d'hôte de l'inventaire
- Recherche sur tous les nœuds configurés
- Force l'arrêt après 20 secondes

Distribution et balisage des VM/LXC

Distribution Round-Robin

Les VMs et LXCs sont automatiquement distribués sur les nœuds Proxmox en utilisant une logique de round-robin (modulo) :

Exemple avec 3 hyperviseurs et 5 MMEs :

```
mme01 → pve-node-01 (index 0 % 3 = 0)
mme02 → pve-node-02 (index 1 % 3 = 1)
mme03 → pve-node-03 (index 2 % 3 = 2)
mme04 → pve-node-01 (index 3 % 3 = 0)
mme05 → pve-node-02 (index 4 % 3 = 1)
```

Comment cela fonctionne :

1. Le playbook identifie le groupe de rôle de l'hôte (par exemple, `mme`, `sgw`, `hss`)
2. Calcule l'index de l'hôte au sein de ce groupe (basé sur 0)

3. Utilise l'opération modulo : `host_index % number_of_nodes`
4. Sélectionne l'hyperviseur en fonction du résultat

Configuration :

```
# Pour les VMs (proxmox.yml) - définir plusieurs serveurs
proxmoxServers:
  pve-node-01: { ... }
  pve-node-02: { ... }
  pve-node-03: { ... }

# Pour les LXCs (proxmox_lxc.yml) - lister plusieurs nœuds
proxmoxNodeName: ['pve-node-01', 'pve-node-02', 'pve-node-03']
```

Balisage Automatique

Les VMs et LXCs sont automatiquement balisés avec :

- **Noms de rôle/groupe** : Tous les groupes Ansible auxquels appartient l'hôte
- **Nom du site** : La variable `site_name`

Exemple :

```
site_name: "melbourne-prod"

mme:
  hosts:
    melbourne-mme01: { ... }
```

Résultat : VM/LXC balisé avec : `mme`, `melbourne-prod`

Les balises sont visibles dans l'UI Proxmox et utiles pour le filtrage/l'organisation.

Remplacements par Hôte

Remplacer les valeurs par défaut sur des hôtes spécifiques :

```
hosts:
  high-spec-host:
    ansible_host: 192.168.1.50
    gateway: 192.168.1.1
    proxmoxLxcCores: 8        # remplacement des cœurs
    proxmoxLxcMemoryMb: 16384  # remplacement de la mémoire
    proxmoxLxcDiskSizeGb: 100  # remplacement du disque
```

Playbooks Utilitaires

Les playbooks utilitaires fournissent des outils opérationnels pour gérer l'infrastructure OmniCore déployée. Ces playbooks se trouvent dans le répertoire `util_playbooks/` et peuvent être exécutés indépendamment pour effectuer des tâches courantes de maintenance et de dépannage.

Référence Rapide

Playbook	Objectif
`health_check.yml`	Générer un rapport de santé complet pour tous les services
`restore_hss.yml`	Restaurer la base de données HSS et/ou la configuration à partir d'une sauvegarde
`ip_plan_generator.yml`	Générer une documentation réseau avec des diagrammes Mermaid
`get_ports.yml`	Auditer les ports ouverts et les services à l'écoute sur tous les hôtes
`getLocalCapture.yml`	Récupérer les fichiers de capture de paquets des hôtes
`delete_local_user.yml`	Supprimer un compte utilisateur local de tous les hôtes
`updateMtu.yml`	Définir le MTU à 9000 (trames jumbo) sur les interfaces réseau
`systemctl status.yml`	Vérifier l'état des services sur les composants EPC

Vérification de la Santé

Fichier : `util_playbooks/health_check.yml`

Génère un rapport de santé HTML complet couvrant tous les services OmniCore et OmniCall déployés.

```
ansible-playbook -i hosts/customer/host_files/production.yml
util_playbooks/health_check.yml
```

Sortie : `/tmp/health_check_YYYY-MM-DD HH:MM:SS.html`

Informations Collectées

Composant	Données Collectées
Tous les services	État du service, version, temps de fonctionnement
OmniHSS	État de la base de données, connexions de pairs Diameter
OmniDRA	Connexions de pairs Diameter et état
OmniTAS	Appels actifs, sessions, enregistrements, utilisation du CPU
OCS	État de la réplication KeyDB

Restauration HSS

Fichier : `util_playbooks/restore_hss.yml`

Restaure OmniHSS à partir de fichiers de sauvegarde. Prend en charge la restauration uniquement de la base de données, uniquement de la

configuration, ou les deux.

```
ansible-playbook -i hosts/customer/host_files/production.yml
util_playbooks/restore_hss.yml
```

Formats de Fichiers de Sauvegarde

Type	Modèle de Nom de Fichier	Contenu
Base de données	`hss_dump_<hostname>_<timestamp>.sql`	Dump MySQL de la base de données `omnihss`
Configuration	`hss_<hostname>_<timestamp>.tar.gz`	Archive du répertoire `/etc/omnihss`

Générateur de Plan IP

Fichier : `util_playbooks/ip_plan_generator.yml`

Génère une documentation réseau à partir de l'inventaire, y compris :

- Attributions IP des hôtes (NICs primaires et secondaires)
- Vue d'ensemble des segments réseau
- Diagrammes de connectivité des interfaces (Diameter, GTP, PFCP, SIP, SS7)

```
ansible-playbook -i hosts/customer/host_files/production.yml
util_playbooks/ip_plan_generator.yml
```

Fichiers de Sortie

Fichier	Format	Description
`/tmp/ip_plan_<customer>_<site>.md`	Markdown	Documentation basée sur du texte
`/tmp/ip_plan_<customer>_<site>.html`	HTML	Diagramme interactif avec des couches filtrables

Audit des Ports

Fichier : `util_playbooks/get_ports.yml`

Audite tous les ports à l'écoute sur le déploiement et génère de la documentation.

```
ansible-playbook -i hosts/customer/host_files/production.yml
util_playbooks/get_ports.yml
```

Fichiers de Sortie

Fichier	Description
`/tmp/all_ports.csv`	CSV avec nom d'hôte, IP, protocole, port, service
`./open_ports.rst`	Table reStructuredText pour la documentation Sphinx

Données Collectées

Champ	Description
Nom d'hôte	Nom d'hôte de l'inventaire
IP	Adresse IP `ansible_host` de l'hôte
Version IP	IPv4 ou IPv6
Transport	TCP ou UDP
Port	Numéro de port à l'écoute
Service	Nom du processus

Récupération de Capture Locale

Fichier : `util_playbooks/getLocalCapture.yml`

Récupère les deux fichiers de capture de paquets les plus récents du répertoire `/etc/localcapture` de chaque hôte.

```
ansible-playbook -i hosts/customer/host_files/production.yml
util_playbooks/getLocalCapture.yml
```

Sortie : `./localCapturePcaps/<hostname>/*.pcap`

Gestion des Utilisateurs

Fichier : `util_playbooks/delete_local_user.yml`

Supprime un compte utilisateur local de tous les hôtes de l'inventaire.

```
ansible-playbook -i hosts/customer/host_files/production.yml
util_playbooks/delete_local_user.yml
```

Invite : Entrez le nom d'utilisateur à supprimer lorsqu'il est demandé.

Configuration MTU

Fichier : `util_playbooks/updateMtu.yml`

Définit le MTU à 9000 (trames jumbo) sur l'interface `ens160` de tous les hôtes.

```
ansible-playbook -i hosts/customer/host_files/production.yml
util_playbooks/updateMtu.yml
```

Remarque : Ce playbook est codé en dur pour l'interface `ens160`. Modifiez le playbook si votre environnement utilise des noms d'interface différents.

Exécution des Playbooks Utilitaires

Syntaxe de Base

```
ansible-playbook -i <inventory_file> util_playbooks/<playbook>.yml
```

Options Courantes

Option	Description
`-i <inventory>`	Spécifier le fichier d'inventaire
`--limit <hosts>`	Limiter à des hôtes ou groupes spécifiques
`-v` / `-vv` / `-vvv`	Augmenter la verbosité
`--check`	Exécution à blanc (afficher ce qui changerait)
`--diff`	Afficher les différences de fichiers

Exemples

```
# Exécuter la vérification de santé en production
ansible-playbook -i hosts/acme/host_files/production.yml
util_playbooks/health_check.yml

# Restaurer HSS sur un hôte spécifique
ansible-playbook -i hosts/acme/host_files/production.yml
util_playbooks/restore_hss.yml --limit hss01

# Générer un plan IP avec une sortie verbeuse
ansible-playbook -i hosts/acme/host_files/production.yml
util_playbooks/ip_plan_generator.yml -v
```