Introducción al Despliegue de Ansible en Omnitouch

Resumen

Omnitouch Network Services utiliza Ansible como su plataforma de automatización de infraestructura para desplegar soluciones completas de red celular (4G/5G) de manera consistente, repetible y automatizada. Este documento proporciona una visión general de cómo aprovechamos Ansible para orquestar despliegues complejos de telecomunicaciones.

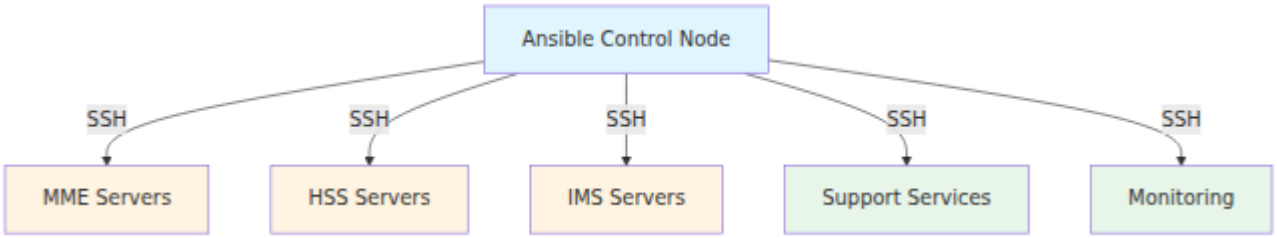
¿Qué es Ansible?

Ansible es una herramienta de automatización de código abierto que te permite:

- Configurar sistemas
- Desplegar software
- Orquestar flujos de trabajo complejos
- Gestionar infraestructura como código

Ansible utiliza un enfoque declarativo: describes el **estado deseado** de tus sistemas, y Ansible se asegura de que alcancen ese estado.

Cómo Usa Omnitouch Ansible



Conceptos Clave

1. Inventario (Archivos de Hosts)

Define **qué** sistemas gestionar. Cada despliegue de cliente tiene un archivo de hosts que describe:

- Todas las máquinas virtuales en la red
- Sus direcciones IP
- Configuración de red
- Parámetros específicos del servicio

Los archivos de hosts son con los que trabajarás para definir tu red.

Ver: Configuración del Archivo de Hosts

2. Roles

Define **cómo** configurar cada componente. Los roles son unidades reutilizables que contienen:

- Tareas (pasos a ejecutar)
- Plantillas (plantillas de archivos de configuración)
- Controladores (acciones desencadenadas por cambios)
- Variables (valores de configuración predeterminados)

Ejemplos de roles para componentes de OmniCore: `omnihss`, `omnisgwc`, `omnipgwc`, `omnidra`, etc.

Estos son definidos por el equipo de ONS, mientras que puedes editarlos, generalmente hay formas más limpias de hacer cualquier ajuste que puedas necesitar desde tu archivo de hosts.

3. Playbooks

Orquesta **cuándo** y **dónde** se aplican los roles:

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

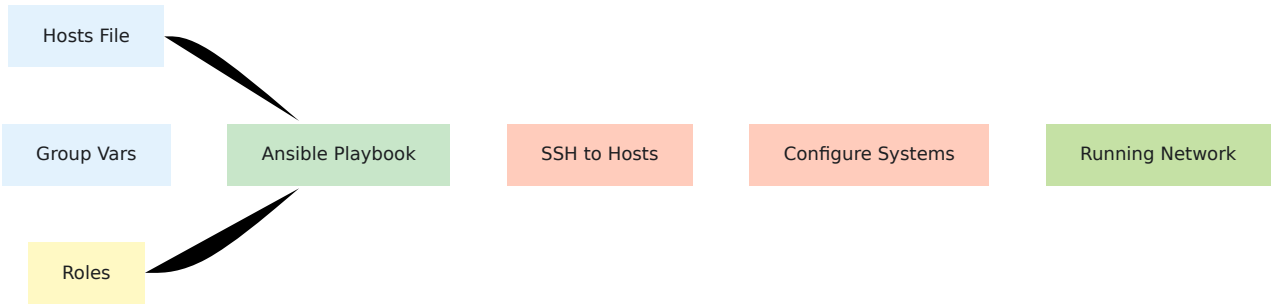
Los usamos esencialmente como grupos para los roles.

4. Variables de Grupo

Proporciona **configuración específica del cliente** que anula los valores predeterminados del rol. Aquí es donde ocurre la personalización del cliente sin modificar los roles base.

Ver: Variables de Grupo y Configuración

Arquitectura de Despliegue



El Proceso de Despliegue

1. Definir Infraestructura

Crea un archivo de hosts que describa tu topología de red:

Nota de Planificación: Antes de definir la infraestructura, revisa el Estándar de Planificación de IP para obtener orientación sobre segmentación de red, asignación de direcciones IP y organización de subredes.

Usuarios de Proxmox: Si despliegas en Proxmox, consulta Despliegue de VM/LXC en Proxmox para la provisión automatizada de VM/contenedores.

Ver: Configuración del Archivo de Hosts y Referencia de Configuración

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

2. Personalizar Configuración

Establece variables específicas del cliente en `group_vars`:

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

#ToDo - Agregar enlace aquí a la referencia de configuración para la lista completa

3. Ejecutar Playbooks

Desplegar la red:

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

4. Despliegue Automatizado

Ansible hará:

- Crear/provisionar VMs (si se utiliza la integración de Proxmox/VMware)
- Configurar la red
- Instalar paquetes de software desde la caché de APT
- Desplegar código de aplicación
- Configurar servicios con configuraciones del cliente
- Iniciar servicios
- Validar el despliegue

Componentes Clave que Desplegamos

OmniCore (Plataforma de Núcleo de Paquete 4G/5G)

- **OmniHSS** - Servidor de Suscriptores de Hogar
- **OmniSGW** - Puerta de Enlace de Servicio (plano de control)
- **OmniPGW** - Puerta de Enlace de Paquete (plano de control)
- **OmniUPF** - Función de Plano de Usuario
- **OmniDRA** - Agente de Enrutamiento Diameter
- **OmniTWAG** - Puerta de Enlace de Acceso WLAN de Confianza

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

OmniCall (Plataforma de Voz y Mensajería)

- **OmniCall CSCF** - Función de Control de Sesiones de Llamada (P-CSCF, I-CSCF, S-CSCF)
- **OmniTAS** - Servidor de Aplicaciones IMS (servicios VoLTE/VoNR)
- **OmniMessage** - Centro de SMS (SMS-C)
- **OmniMessage SMPP** - Soporte para el protocolo SMPP
- **OmniSS7** - Componentes de señalización SS7 (STP, HLR, CAMEL)
- **VisualVoicemail** - Funcionalidad de correo de voz

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

OmniCharge/OmniCRM

- **Plataforma CRM** - Gestión de relaciones con clientes, auto-registro, facturación

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

Servicios de Soporte

- **DNS** - Resolución DNS de red
- **Servidor de Licencias** - Gestión de licencias
- **Monitoreo** - Prometheus, Grafana

Ver: Resumen de Arquitectura de Despliegue

Gestión de Paquetes

Utilizamos un modelo híbrido de distribución de paquetes:

Paquetes APT Precompilados

Todo el software de Omnitouch se distribuye como paquetes Debian (`.deb`):

- Construidos a partir del código fuente en nuestra pipeline de CI/CD

- Versionados y probados
- Alojados en repositorios de paquetes

Sistema de Caché APT

Los clientes pueden elegir entre:

1. **Caché APT Local** - Espejo de paquetes requeridos en el sitio para despliegue fuera de línea
2. **Repositorio Público** - Acceso directo al repositorio de paquetes alojado por Omnitouch

Ver: Sistema de Caché APT

Gestión de Licencias

Todos los componentes de software de Omnitouch requieren licencias válidas gestionadas a través de un servidor de licencias central:

- Los componentes verifican la validez de la licencia al iniciar
- Las características se habilitan/deshabilitan según la licencia
- El servidor de licencias puede ser local o alojado en la nube

Ver: Servidor de Licencias

Beneficios de Este Enfoque

Repetibilidad

Los mismos playbooks de Ansible pueden desplegar:

- Laboratorios de desarrollo
- Entornos de prueba
- Redes de producción
- Sitios de clientes

Consistencia

Cada despliegue utiliza las mismas configuraciones probadas, reduciendo el error humano.

Control de Versiones

La infraestructura se define como código en Git:

- Rastrear todos los cambios
- Revisar antes del despliegue
- Revertir si es necesario

Personalización Sin Complejidad

Los clientes pueden personalizar su despliegue a través de `group_vars` sin modificar los roles centrales.

Despliegue Rápido

Desplegar una red celular completa en horas en lugar de días o semanas.

Comenzando

Requisitos Previos

Antes de ejecutar los playbooks de Ansible, necesitas configurar un entorno virtual de Python e instalar las dependencias requeridas.

1. Crear un Entorno Virtual de Python

Crea un entorno aislado de Python para el despliegue de Ansible:

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

2. Activar el Entorno Virtual

Activa el entorno virtual:

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

En Windows, usa:

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

3. Instalar Paquetes Requeridos

Instala todas las dependencias desde el archivo requirements.txt:

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

Esto instalará Ansible y todos los paquetes de Python necesarios para la automatización del despliegue de Omnitouch.

Nota: Mantén el entorno virtual activado siempre que ejecutes comandos de Ansible. Puedes desactivarlo cuando termines ejecutando `deactivate`.

Pasos de Despliegue

1. Revisa la Configuración del Archivo de Hosts para entender cómo definir tu red
2. Aprende sobre Variables de Grupo para personalización
3. Comprende el Sistema de Caché APT para la gestión de paquetes
4. Revisa la Arquitectura de Despliegue para ver cómo encaja todo
5. ¡Despliega!

Próximos Pasos

- Estándar de Planificación de IP - **Planifica tu arquitectura de red y asignación de IP**

- Configuración del Archivo de Hosts - Aprende cómo definir tu topología de red
- Sistema de Caché APT - Comprende la distribución de paquetes
- Servidor de Licencias - Aprende sobre la gestión de licencias
- Resumen de Arquitectura de Despliegue - Ve la imagen completa
- Configuración de Variables de Grupo - Personaliza tu despliegue
- Playbooks de Utilidad - Herramientas operativas para chequeos de salud, copias de seguridad y mantenimiento

Repositorio APT y Distribución de Paquetes

Descripción General

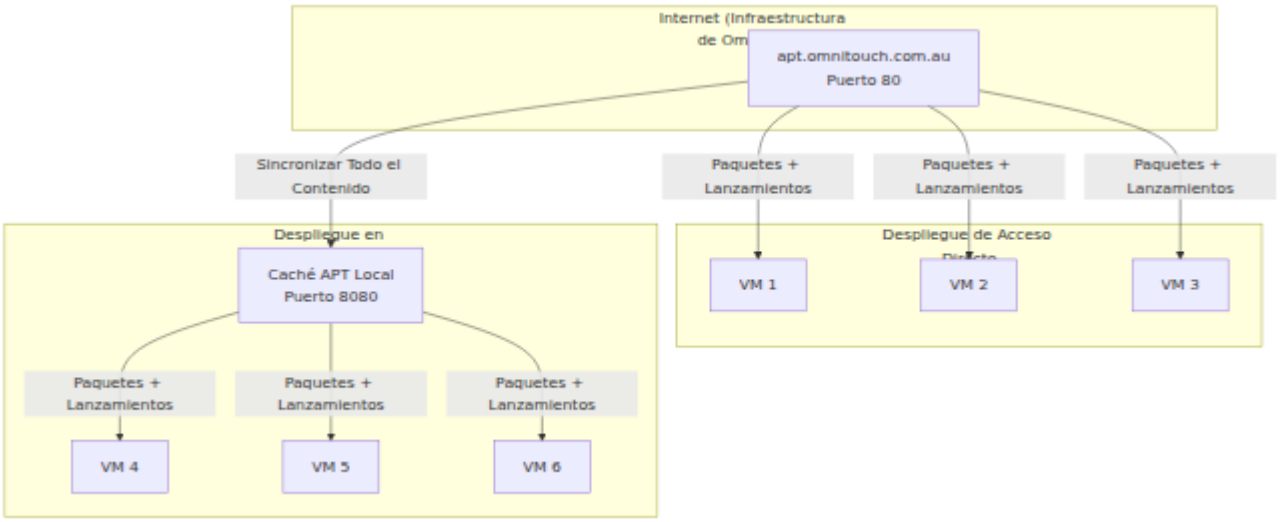
El sistema APT de Omnitouch proporciona distribución de paquetes para todos los despliegues. Se sirven dos tipos de contenido:

1. **Paquetes APT** — Paquetes Debian instalados a través de `apt install`
2. **Lanzamientos Binarios** — Binarios precompilados descargados directamente (exportadores de Prometheus, agentes, etc.)

Se admiten dos modelos de despliegue:

1. **Acceso Directo** — Las máquinas virtuales obtienen paquetes directamente de `apt.omnitouch.com.au`
2. **Espejo de Caché Local** — Un servidor local sincroniza desde Omnitouch y sirve paquetes a las máquinas virtuales (para despliegues fuera de línea/aislados)

Arquitectura



Contenido Servido

El servidor APT alberga todo el contenido requerido para los despliegues:

Tipo de Contenido	Descripción	Ruta
Paquetes Omnitouch	Paquetes `.deb` personalizados (omnihss, omnimme, etc.)	`/dists/<distro>/`
Paquetes de Ubuntu	Paquetes de Ubuntu en caché con todas las dependencias	`/<distro>/pool/main/`
Lanzamientos de GitHub	Binarios precompilados (Prometheus, Grafana, Homer, etc.)	`/releases/<org>/<repo>/`
Tarballs de Fuente	Archivos fuente para aplicaciones web (CGrateS_UI, speedtest)	`/repos/`
Paquetes de Terceros	Galera, FRR, InfluxDB, KeyDB, etc.	`/releases/<vendor>/`

Variables de Configuración

Dos conjuntos de variables separadas controlan la distribución de paquetes. Comprender sus propósitos es esencial para una configuración correcta.

Propósitos de las Variables

Conjunto de Variables	Propósito	Usado Para
`apt_repo`	Configura las fuentes de paquetes APT	`/etc/apt/sources.list` y `/etc/apt/sources.list.d/*.list`
`remote_apt_*`	Configura las URL de descarga de binarios	Descargando archivos de la ruta `/releases/` (Node Exporter, Zabbix, Nagios, etc.)

Cuándo Se Usa Cada Conjunto de Variables

Escenario	Fuentes APT (`apt_repo`)	Descargas Binarias (`remote_apt_*`)
`use_apt_cache: true`	Usa `apt_repo.apt_server`	Usa `apt_repo.apt_server`
`use_apt_cache: false`	Usa `apt_repo.*` con credenciales	Usa `remote_apt_*` con credenciales

Cuando `use_apt_cache: false`, ambos conjuntos de variables son requeridos.

Opción 1: Acceso Directo

Para despliegues con conectividad a Internet, las máquinas virtuales obtienen paquetes directamente del servidor APT de Omnitouch.

Requisitos de Red

Lista Blanca de IP de Origen: Tu dirección IP pública debe estar en la lista blanca en el servidor APT de Omnitouch. Durante la configuración, proporciona tus subredes de origen a Omnitouch. A cambio, recibirás:

- **Nombre de usuario** y **contraseña** para la autenticación básica HTTP
- **FQDN** para el servidor APT

Requisitos de Firewall: Se debe permitir el acceso saliente a los siguientes rangos de IP de Omnitouch:

Red	Rango
IPv4	`144.79.167.0/24`
IPv4	`160.22.43.0/24`
IPv6	`2001:df3:dec0::/48`
ASN	`AS152894`

Servicios que requieren acceso a la infraestructura de Omnitouch:

Servicio	Puerto	Protocolo	Propósito
Servidor APT	80	TCP	Descargas de paquetes
Servidor APT	53	TCP/UDP	Resolución DNS para `apt.omnitouch.com.au`
Servidor de Licencias	123	UDP	Sincronización de tiempo NTP para validación de licencias
Servidor de Licencias	53	TCP/UDP	Resolución DNS para validación de licencias

Asegúrate de que el tráfico HTTP (TCP/80), NTP (UDP/123) y DNS (TCP+UDP/53) esté permitido hacia los rangos de IP de Omnitouch.

Configuración

```yaml
all:
  vars:
    use_apt_cache: false

    # Configuración de fuentes de paquetes APT
    # Configura /etc/apt/sources.list para comandos apt install
    apt_repo:
      apt_server: "apt.omnitouch.com.au"
      apt_repo_username: "tu-usuario"
      apt_repo_password: "tu-contraseña"

    # Configuración de descargas binarias
    # Usado para descargar archivos de la ruta /releases/
    remote_apt_server: "apt.omnitouch.com.au"
    remote_apt_port: 80
    remote_apt_protocol: "http"
    remote_apt_user: "tu-usuario"
    remote_apt_password: "tu-contraseña"
```

Parámetros

Fuentes de Paquetes APT (`apt_repo`)

Parámetro	Tipo	Requerido	Predeterminado
`apt_repo.apt_server`	Cadena	Sí	-
`apt_repo.apt_repo_username`	Cadena	Sí	-
`apt_repo.apt_repo_password`	Cadena	Sí	-

Descargas Binarias (`remote_apt_*`)

Parámetro	Tipo	Requerido	Predeterminado	Descr
`remote_apt_server`	Cadena	Sí	-	Nombr host o servid descar binaria
`remote_apt_port`	Entero	No	`80`	Puerto servid descar binaria
`remote_apt_protocol`	Cadena	No	`http`	Protoc (`http` `https`
`remote_apt_user`	Cadena	Sí	-	Nombr usuari autent básica para descar
`remote_apt_password`	Cadena	Sí	-	Contra de autent básica para descar

General

Parámetro	Tipo	Requerido	Predeterminado	Descripció
`use_apt_cache`	Booleano	Sí	-	Debe ser `false` para acceso directo

Patrones de URL (Acceso Directo)

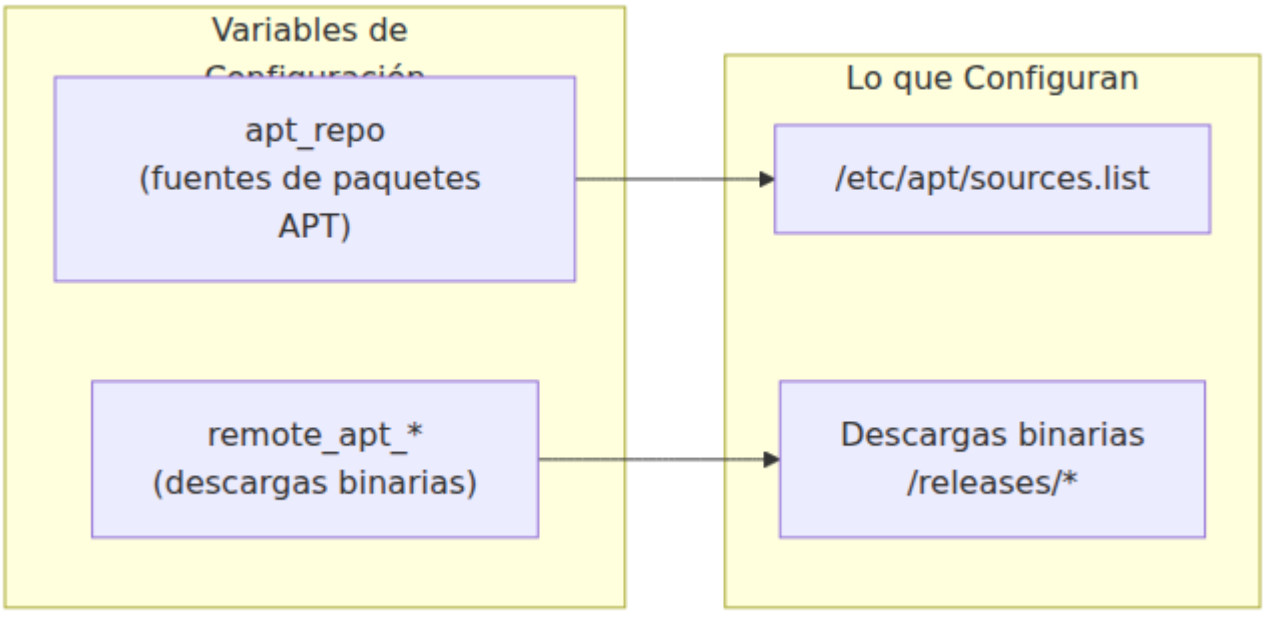
Fuentes de Paquetes APT (configuradas en `/etc/apt/sources.list`):

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

Descargas Binarias (usadas por tareas `get_url` de 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
```

Cómo Funciona



Las máquinas virtuales se autentican con autenticación básica HTTP tanto para paquetes APT como para descargas binarias. Los paquetes del sistema Ubuntu también se sirven desde el servidor de Omnitouch (pre-caché), por lo que las máquinas virtuales no necesitan acceso a los espejos de Ubuntu.

Opción 2: Espejo de Caché Local

Para despliegues fuera de línea, aislados o con limitaciones de ancho de banda, despliega una caché APT local que sincroniza todo el contenido desde Omnitouch.

Arquitectura

Configuración

Define el servidor de caché en tu archivo de hosts con su configuración de repositorio:

```yaml
apt_cache_servers:
  hosts:
    customer-apt-cache:
      ansible_host: 192.168.1.100
      gateway: 192.168.1.1
  vars:
    # El servidor de caché sincroniza paquetes desde el
repositorio autenticado
    remote_apt_server: "apt.omnitouch.com.au"
    remote_apt_port: 80
    remote_apt_protocol: "http"
    remote_apt_user: "tu-usuario"
    remote_apt_password: "tu-contraseña"

all:
  vars:
    # use_apt_cache: true  # Se establece automáticamente cuando
existe el grupo apt_cache_servers
    # apt_repo.apt_server: derivado automáticamente a
192.168.1.100 (primer servidor de caché)
```

Cómo funciona:

- **Servidor de caché** (`192.168.1.100`): Usa credenciales `remote_apt_*` para sincronizar paquetes desde `apt.omnitouch.com.au:80`
- **Todos los demás hosts**: Derivan automáticamente `apt_repo.apt_server: "192.168.1.100"` y obtienen desde la caché en el puerto `8080` sin credenciales

Parámetros

Fuentes de Paquetes APT (`apt_repo`)

Parámetro	Tipo	Requerido	Predeterminado
`apt_repo.apt_server`	Cadena	Sí	Derivado automáticamente
`apt_repo.apt_repo_username`	Cadena	No	-
`apt_repo.apt_repo_password`	Cadena	No	-

Sincronización del Servidor de Caché (`remote_apt_*`)

Estas variables configuran cómo el servidor de caché sincroniza contenido desde Omnitouch:

Parámetro	Tipo	Requerido	Predeterminado	Descr
`remote_apt_server`	Cadena	Sí	-	Servid de Om desde sincron
`remote_apt_port`	Entero	No	`80`	Puerto servid de Om
`remote_apt_protocol`	Cadena	No	`http`	Protoc para la conexi sincron
`remote_apt_user`	Cadena	Sí	-	Creder para sincron desde Omnit
`remote_apt_password`	Cadena	Sí	-	Creder para sincron desde Omnit

General

Parámetro	Tipo	Requerido	Predeterminado	Descri
`use_apt_cache`	Booleano	No	`true`	Se estable automática en `true` cu existe el g `apt_cache`
`apt_cache_port`	Entero	No	`8080`	Puerto en escucha el de caché l

Patrones de URL (Modo Caché)

Fuentes de Paquetes APT (configuradas en `/etc/apt/sources.list`):

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

Descargas Binarias (usadas por tareas `get_url` de Ansible):

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

No se requieren credenciales para el acceso a la caché; utiliza la configuración APT `[trusted=yes]`.

Desplegando la Caché

1. **Provisión del servidor de caché** (VM o contenedor LXC con disco de 50+ GB)

2. **Ejecuta el libro de jugadas de configuración de la caché:**

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

3. **Verifica la caché** navegando a `http://192.168.1.100:8080/`

Qué Se Sincroniza

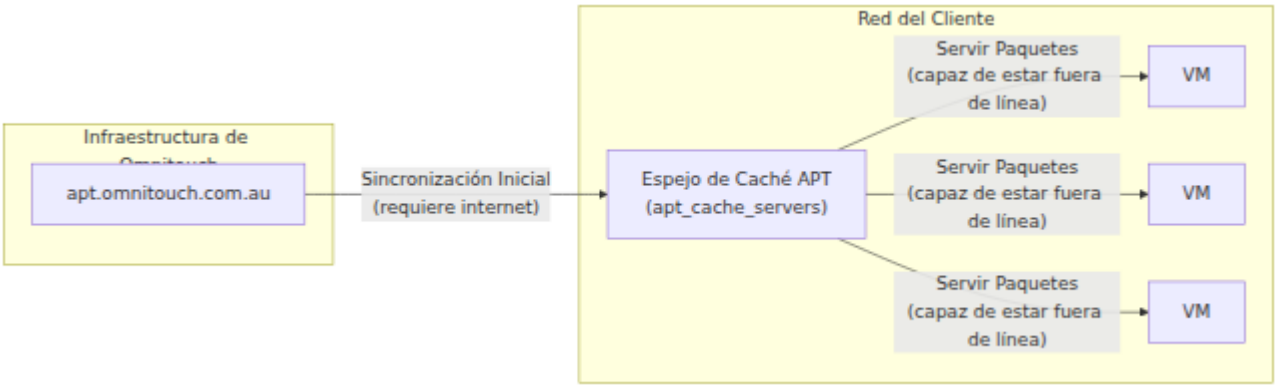
El espejo de caché sincroniza **todo el contenido** desde el servidor APT de Omnitouch utilizando descarga recursiva con wget:

Directorios de contenido sincronizados:

Ruta	Contenido
`/dists/<distro>/`	Metadatos del repositorio APT (archivos Packages, Release)
`/pool/main/`	Paquetes .deb personalizados de Omnitouch
`/<distro>/pool/main/`	Paquetes de Ubuntu y todas las dependencias
`/releases/`	Lanzamientos de GitHub (Prometheus, Grafana, Zabbix, etc.)
`/repos/`	Tarballs fuente (Erlang, Elixir, CGrateS_UI, etc.)

Después de la sincronización inicial, la caché puede servir todos los paquetes sin conectividad a Internet.

Cómo Funciona



El espejo de caché utiliza `wget --recursive` con autenticación básica HTTP para descargar todo el contenido desde el servidor APT de Omnitouch. Las sincronizaciones posteriores solo descargan archivos nuevos/cambiados (marcado de tiempo).

Configuración Automática

Cuando existe un grupo `apt_cache_servers` en tu inventario, el sistema automáticamente:

1. Establece `use_apt_cache: true` para todos los hosts (a menos que se anule explícitamente)
2. Deriva `apt_repo.apt_server` de la IP `ansible_host` del primer servidor de caché

Ejemplo de Configuración Mínima

```yaml
apt_cache_servers:
  hosts:
    apt-cache-01:
      ansible_host: 192.168.1.100
      gateway: 192.168.1.1
  vars:
    # El servidor de caché sincroniza contenido desde el
repositorio de Omnitouch
    remote_apt_server: "apt.omnitouch.com.au"
    remote_apt_user: "tu-usuario"
    remote_apt_password: "tu-contraseña"
```

Lo que sucede automáticamente:

- Todos los hosts (excepto el servidor de caché) obtienen `use_apt_cache: true`
- Todos los hosts (excepto el servidor de caché) obtienen `apt_repo.apt_server: "192.168.1.100"`
- Todos los hosts obtienen desde `http://192.168.1.100:8080/` sin credenciales
- El servidor de caché sincroniza paquetes desde `http://tu-usuario:tu-contraseña@apt.omnitouch.com.au/`

Anular Comportamiento Automático

Para forzar el acceso directo incluso con servidores de caché definidos:

```yaml
all:
  vars:
    use_apt_cache: false  # Forzar acceso directo incluso con
servidores de caché definidos

    apt_repo:
      apt_server: "apt.omnitouch.com.au"
      apt_repo_username: "usuario"
      apt_repo_password: "contraseña"

    remote_apt_server: "apt.omnitouch.com.au"
    remote_apt_user: "usuario"
    remote_apt_password: "contraseña"
```

Resumen de Configuración

Escenario 1: Acceso Directo al Servidor APT (Sin Caché)

Todos los hosts obtienen paquetes directamente del servidor del repositorio APT.

```yaml
all:
  vars:
    use_apt_cache: false

    # Fuentes de paquetes APT - usadas por todos los hosts
    apt_repo:
      apt_server: "apt.omnitouch.com.au"
      apt_repo_username: "usuario"
      apt_repo_password: "contraseña"

    # Descargas binarias - usadas por todos los hosts
    remote_apt_server: "apt.omnitouch.com.au"
    remote_apt_port: 80
    remote_apt_protocol: "http"
    remote_apt_user: "usuario"
    remote_apt_password: "contraseña"
```

Resultado: Todos los hosts generan `deb [trusted=yes] http://usuario:contraseña@apt.omnitouch.com.au/ noble main`

Escenario 2: Servidor de Caché APT Definido en el Archivo de Hosts (Automático)

El servidor de caché está en tu inventario y será desplegado/sincronizado por Ansible.

```yaml
apt_cache_servers:
  hosts:
    cache-server:
      ansible_host: 192.168.1.100
      gateway: 192.168.1.1
  vars:
    # El servidor de caché sincroniza paquetes desde el
repositorio autenticado
    remote_apt_server: "apt.omnitouch.com.au"
    remote_apt_port: 80
    remote_apt_protocol: "http"
    remote_apt_user: "usuario"
    remote_apt_password: "contraseña"

# No se necesita configuración en all: vars:
# Todo se deriva automáticamente del grupo apt_cache_servers
```

Resultado:

- **Servidor de caché**: Sincroniza desde `http://usuario:contraseña@apt.omnitouch.com.au:80/`
- **Todos los demás hosts**: Generan `deb [trusted=yes] http://192.168.1.100:8080/noble noble main` (sin credenciales)

Escenario 3: Caché APT Remota NO en el Archivo de Hosts (Manual)

El servidor de caché existe en otro lugar y ya está configurado (no gestionado por tu Ansible).

```yaml
all:
  vars:
    use_apt_cache: true

    # Apuntar todos los hosts al servidor de caché externo
    apt_repo:
      apt_server: "192.168.1.100"  # IP del servidor de caché
externo
      apt_repo_port: 8080           # La caché generalmente se
ejecuta en el puerto 8080

# No se necesita grupo apt_cache_servers
# No se necesita remote_apt_* (la caché ya está configurada
externamente)
```

Resultado: Todos los hosts generan `deb [trusted=yes] http://192.168.1.100:8080/noble noble main` (sin credenciales)

Ejemplo Completo

Aquí hay un ejemplo completo que muestra la configuración del servidor de caché con múltiples hosts de aplicación:

```yaml
# Grupo del Servidor de Caché 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:
    # El servidor de caché sincroniza paquetes desde el
repositorio autenticado
    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"

# Servidores de Aplicación
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

# Configuración Global
all:
  vars:
    # Auto-configuración (no se necesita configuración manual):
    # - use_apt_cache: true (auto-habilitado cuando existe
```

```
apt_cache_servers)
    # - apt_repo.apt_server: "10.179.1.114" (auto-derivado del
servidor de caché)
```

Lo que sucede durante el despliegue:

1. **Servidor de caché (`10.179.1.114`):**
   - Usa `remote_apt_*` de su sección `vars:`
   - Descarga todos los paquetes desde `http://customer-username:customer-secure-token@apt.omnitouch.com.au:80/`
   - Sirve paquetes en el puerto 8080 a través de nginx
2. **Hosts de aplicación (`customer-hss01`, `customer-mme01`, `customer-dns01`):**
   - Detectan automáticamente que existe el grupo `apt_cache_servers`
   - Establecen automáticamente `use_apt_cache: true`
   - Derivan automáticamente `apt_repo.apt_server: "10.179.1.114"`
   - Generan: `deb [trusted=yes] http://10.179.1.114:8080/noble noble main`
   - Obtienen todos los paquetes desde el servidor de caché (sin credenciales requeridas)

Actualizando la Caché

Para sincronizar nuevos paquetes o actualizaciones:

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

Esto sincroniza de manera incremental todo el contenido desde el servidor APT de Omnitouch:

- Nuevas versiones de paquetes de Omnitouch

- Nuevos paquetes de Ubuntu y dependencias
- Nuevos lanzamientos de GitHub
- Tarballs de fuente actualizados

La sincronización utiliza `wget --timestamping`, por lo que se omiten los archivos existentes sin cambios, haciendo que la re-sincronización sea rápida.

Nota: El servidor APT de Omnitouch (`apt.omnitouch.com.au`) es la única fuente de verdad para todos los paquetes. Ejecuta `services/apt.yml` en el servidor apt primero para construir/actualizar paquetes, luego ejecuta `services/apt_cache.yml` en los espejos de caché para sincronizar.

Solución de Problemas

La Actualización de APT Falla con 401 No Autorizado

Síntomas:

```
Failed to fetch
http://10.179.1.115:80/noble/dists/noble/main/binary-
amd64/Packages  401  Unauthorized
```

Causas posibles:

- Configuración de `apt_repo` definida en `all: vars:` en lugar de `apt_cache_servers: vars:`
- Hosts intentando acceder al repositorio autenticado directamente en lugar de a la caché
- `apt_repo_username` o `apt_repo_password` incorrectos
- IP de origen no en la lista blanca en el servidor APT de Omnitouch
- Usando credenciales de caché para acceso directo o viceversa

Resolución:

1. **Verifica el alcance de la configuración**: Asegúrate de que `apt_repo` con credenciales esté definido en `apt_cache_servers: vars:`, NO en `all: vars:`
2. **Verifica el modo de caché**: Al usar caché, los hosts deben conectarse al servidor de caché (puerto 8080), no al repositorio (puerto 80)
3. **Verifica las fuentes generadas**: En el host que falla, verifica `/etc/apt/sources.list.d/omnitouch.list`
   - **Correcto (modo caché)**: `deb [trusted=yes] http://10.179.1.114:8080/noble noble main`
   - **Incorrecto (tiene credenciales en el lugar equivocado)**: `deb [trusted=yes] http://usuario:contraseña@10.179.1.115:80/noble noble main`
4. Verifica que las credenciales sean correctas para tu modo de despliegue
5. Confirma que tu IP pública esté en la lista blanca con Omnitouch (si usas acceso directo)

Las Descargas Binarias Fallan (Node Exporter, Zabbix, etc.)

Síntomas: El libro de jugadas de Ansible falla al descargar archivos de la ruta `/releases/`

Causas posibles:

- Variables `remote_apt_*` no configuradas
- `remote_apt_user` o `remote_apt_password` incorrectos
- Falta `remote_apt_server` cuando `use_apt_cache: false`

Resolución:

1. Asegúrate de que todas las variables `remote_apt_*` estén definidas
2. Verifica que las credenciales coincidan con las proporcionadas por Omnitouch
3. Verifica que `remote_apt_server` apunte al host correcto

El Servidor de Caché No Puede Sincronizar

Síntomas: El libro de jugadas del servidor de caché falla al descargar paquetes

Causas posibles:

- El servidor de caché no tiene acceso a Internet
- Credenciales `remote_apt_*` incorrectas
- Firewall bloqueando conexiones salientes a Omnitouch

Resolución:

1. Verifica que el servidor de caché pueda alcanzar `apt.omnitouch.com.au` en el puerto 80
2. Verifica las credenciales `remote_apt_*`
3. Revisa las reglas del firewall para el acceso saliente

Documentación Relacionada

- Configuración del Archivo de Hosts — Configuración de inventario y variables
- Referencia de Configuración — Referencia completa de parámetros
- Arquitectura de Despliegue — Arquitectura general del sistema
- Despliegue en Proxmox — Desplegando el servidor de caché como contenedor LXC

Referencia de Configuración

Descripción General

Este documento proporciona una referencia completa para configurar implementaciones de OmniCore a través de archivos de hosts. La configuración se define principalmente en archivos de inventario de hosts con mínimas sobreescrituras de group_vars necesarias para implementaciones modernas.

Para documentación específica del producto, consulte:

- **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

Enfoque de Configuración

Las implementaciones modernas de OmniCore utilizan un modelo de configuración simplificado:

Principio Clave: La mayor parte de la configuración se define directamente en el archivo de hosts. Los valores predeterminados de los roles manejan la mayoría de las configuraciones, con group_vars utilizados solo para personalizaciones específicas.

Planificación de Red

Antes de configurar los hosts, revise el Estándar de Planificación de IP para obtener orientación sobre:

- Estrategias de segmentación de red
- Asignación de direcciones IP
- Organización de subredes
- Manejo de IP públicas

Parámetros Comunes de Host

#ToDo - Solo diga que consulte hosts-file-configuration.md para esto

Flags Específicos del Servicio

```
cdrs_enabled: True                # Habilitar generación de CDR
in_pool: False                    # Excluir del grupo de balanceo
de carga
online_charging_enabled: False    # Habilitar integración OCS
recording: True                   # Habilitar grabación de llamadas
(AS)
populate_crm: False               # Población inicial de CRM
```

Variables Globales (all:vars)

La sección `all:vars` contiene configuraciones a nivel de implementación. Las implementaciones modernas utilizan variables globales mínimas con la mayor parte de la configuración en los valores predeterminados de los roles.

Variables Globales Esenciales

Autenticación y Acceso

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

Alternativa: Use claves SSH en lugar de contraseñas:

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

Identidad del Cliente

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

Configuración de PLMN

```
plmn_id:
  mcc: '001'               # Código de País Móvil (3 dígitos)
  mnc: '01'               # Código de Red Móvil (2-3 dígitos)
  mnc_longform: '001'     # MNC con ceros a la izquierda (siempre
3 dígitos)

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

Propósito: Identifica de manera única su red móvil. Se utiliza para la construcción del dominio Diameter.

Nombres de Red

```
network_name_short: Omni
network_name_long: Omnitouch
tac_list: [10100,100]            # Lista TAC predeterminada (se
puede sobrescribir por MME)
```

Mostrado: Nombres de red que se muestran en dispositivos UE en Configuración > Red Móvil.

Configuración de DNS

```
netplan_DNS: False                # Usar systemd-resolved en lugar
de DNS de netplan
```

Configuración del Repositorio APT

Valores Predeterminados Automáticos: Cuando se define un grupo `apt_cache_servers` con hosts:

- `use_apt_cache` se establece automáticamente en `True` (a menos que se establezca explícitamente en `False`)
- `apt_repo.apt_server` se establece automáticamente en la IP del primer servidor de caché

```
# Configuración manual (opcional si existe el grupo
apt_cache_servers)
use_apt_cache: True               # Usar caché APT local vs acceso
directo al repositorio

apt_repo:
  apt_server: "10.10.1.114"       # Servidor de caché APT o
servidor de repositorio
  # Credenciales solo necesarias cuando use_apt_cache: False
  # apt_repo_username: "omni"
  # apt_repo_password: "omni"

# Configuración de descargas de binarios y sincronización de caché
# Usado para: (1) descargar binarios de /releases/ cuando
use_apt_cache: false
#           (2) sincronización del servidor de caché desde
Omnitouch cuando use_apt_cache: true
remote_apt_server: "apt.omnitouch.com.au"
remote_apt_user: "omni"
remote_apt_password: "omni"
```

Consulte: Sistema de Caché APT

Servidor de Licencias

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

Consulte: Servidor de Licencias

Configuración de MME

```
mme_dns: False                    # Habilitar resolución DNS de MME
```

Configuración de SAEGW

```
mtu: 1400                         # Unidad de Transmisión Máxima
```

Configuración de IMS

```
ims_dra_support: False            # Rutar IMS a través de DRA
enable_homer: False               # Habilitar captura SIP de Homer
```

Configuración del Monitor RAN

```
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
```

Configuración del Cortafuegos

```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'
```

Servidores DNS de Roaming

```yaml
roaming_dns_servers:
  wildcard: ['10.0.99.1']
  # DNS específicos del operador (basados en PLMN)
  123456:                          # Ejemplo Operador 1
    - '10.10.2.197'
  654321:                          # Ejemplo Operador 2
    - '10.10.0.4'
```

Usuarios Locales (Claves SSH)

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

Configuración del Hipervisor

Proxmox

```
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

# Configuraciones predeterminadas 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

```
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
```

Documentación Relacionada

- Estándar de Planificación de IP - Arquitectura de red y directrices de asignación de IP
- Configuración del Archivo de Hosts - Cómo estructurar archivos de hosts
- Configuración de Variables de Grupo - Cuándo y cómo usar group_vars
- Configuración de Netplan - IPs secundarias y configuración de múltiples NIC
- Arquitectura de Implementación - Cómo interactúan los componentes
- Sistema de Caché APT - Gestión de paquetes
- Servidor de Licencias - Configuración de licencias

Documentación del Producto

Para guías operativas detalladas y configuración avanzada:

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

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

Descripción General de la Arquitectura de Despliegue

Descripción

Este documento proporciona una vista completa de cómo se despliega el software de red celular de Omnitouch Network Services utilizando Ansible, mostrando cómo todos los componentes encajan para crear una red 4G/5G funcional.

Consulte el Estándar de Planificación IP para obtener pautas detalladas sobre la ubicación de los componentes, la asignación de direcciones IP y el manejo de IP públicas.

Ejemplo Completo de Despliegue

0. Aprovisionamiento de Infraestructura (Opcional)

Para despliegues en Proxmox, aprovisione VMs/LXCs antes de la configuración:

```
# Desplegar VMs en Proxmox
ansible-playbook -i hosts/Customer/hosts.yml services/proxmox.yml

# O desplegar contenedores LXC (solo laboratorio/prueba)
ansible-playbook -i hosts/Customer/hosts.yml
services/proxmox_lxc.yml
```

Consulte: Despliegue de VM/LXC en Proxmox

1. Definición de Infraestructura (Archivo de Hosts)

```
# Definir qué desplegar y dónde
mme:
  hosts:
    customer-mme01:
      ansible_host: 10.10.1.15

hss:
  hosts:
    customer-hss01:
      ansible_host: 10.10.2.140

# ... todos los demás componentes
```

Consulte: Configuración del Archivo de Hosts

2. Personalización (group_vars)

La carpeta `group_vars` es donde podemos almacenar cualquier anulación de configuración necesaria a nivel de host, sitio o red.

Por ejemplo, tendría una carpeta con su configuración de SMSc de OmniMessage, los troncales SIP a los que se conecta su TAS vivirían aquí, toda su lógica de enrutamiento Diameter, etc., etc.

Consulte: Configuración de Variables de Grupo

3. Distribución de Paquetes (Cache APT)

```
# Configurar de dónde obtener paquetes
apt_repo:
  apt_server: "10.254.10.223"  # IP del servidor de caché o
servidor de repositorio directo
use_apt_cache: false  # true = usar caché local, false = acceso
directo al repositorio
```

Consulte: Sistema de Caché APT

4. Configuración de Licencia

```
# Apuntar componentes al servidor de licencias
license_server_api_urls: ["https://10.10.2.150:8443/api"]
license_enforced: true
```

Consulte: Servidor de Licencias

5. Ejecutar Despliegue

Los componentes individuales se pueden desplegar ejecutando `services/twag.yml`, por ejemplo, pero `services/all.yml` manejará todo, y puede usar `--limit=myhost` o `--limit=mmee,sgw`, etc., para limitar los hosts en los que estamos trabajando.

```
# Desplegar red completa
ansible-playbook -i hosts/customer/host_files/production.yml
services/all.yml

# O desplegar componentes específicos
ansible-playbook -i hosts/customer/host_files/production.yml
services/epc.yml
ansible-playbook -i hosts/customer/host_files/production.yml
services/ims.yml
```

Documentación Relacionada

- Introducción al Despliegue de Ansible - Introducción
- Configuración del Archivo de Hosts - Definición de infraestructura
- Estándar de Planificación IP - **Arquitectura de red y asignación de IP**
- Configuración de Variables de Grupo - Personalización
- Sistema de Caché APT - Gestión de paquetes
- Servidor de Licencias - Gestión de licencias

Documentación del Producto

Para obtener información detallada sobre la configuración de cada componente:

- **OmniCore** (Núcleo de Paquetes 4G/5G): https://docs.omnitouch.com.au/docs/repos/OmniCore
  - OmniHSS, OmniSGW, OmniPGW, OmniUPF, OmniDRA, OmniTWAG
- **OmniCall** (Voz y Mensajería): https://docs.omnitouch.com.au/docs/repos/OmniCall
  - OmniTAS, OmniCall CSCF, OmniMessage, OmniSS7, VisualVoicemail
- **OmniCharge/OmniCRM** (Facturación): https://docs.omnitouch.com.au/docs/repos/OmniCharge
- **Documentación Principal**: https://docs.omnitouch.com.au/

Configuración de Variables de Grupo

Descripción General

El directorio `group_vars` es donde almacenas archivos de configuración personalizados que sobrescriben las plantillas predeterminadas.

Aquí es donde viven tus configuraciones específicas del cliente: troncales SIP, reglas de enrutamiento Diameter, lógica de enrutamiento de SMS, planes de marcado y cualquier otra personalización donde no desees la configuración predeterminada - Vive en `group_vars`.

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

Cómo Funciona

Los roles de Ansible tienen plantillas de configuración predeterminadas. Para personalizar para un despliegue específico, coloca tus archivos personalizados en `group_vars` y haz referencia a ellos en tu archivo de hosts.

```
Plantilla Predeterminada del Rol → Sobrescritura de group_vars (si se especifica) → Configuración Desplegada
```

Ejemplo 1: Plantilla de Configuración Personalizada (OmniMessage)

Algunos componentes aceptan plantillas de configuración Jinja2 personalizadas.

Estructura de Archivos

```
hosts/Customer/
└── group_vars/
    └── smsc_controller.exs        # Tu plantilla de configuración
personalizada
```

Referencia en el Archivo de 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   # Referencia el
nombre de tu plantilla en group_vars
```

Qué sucede:

1. Ansible encuentra `smsc_template_config: smsc_controller.exs`
2. Busca en `hosts/Customer/group_vars/smsc_controller.exs`
3. Lo plantilla con Jinja2 (puede usar `{{ inventory_hostname }}`, `{{ plmn_id.mcc }}`, etc.)
4. Despliega en `/etc/omnimessage/runtime.exs`
5. Reinicia el servicio

Sin `smsc_template_config`, se utiliza la plantilla predeterminada del rol.

Detalles de configuración: Ver https://docs.omnitouch.com.au/docs/repos/OmniCall

---

Ejemplo 2: Colecciones de Archivos de Configuración (Pasarelas y

Planes de Marcado de OmniTAS)

Algunos componentes utilizan directorios de archivos de configuración.

Estructura de Archivos

```
hosts/Customer/
└── group_vars/
    ├── gateways_prod/              # Configuraciones de pasarelas
SIP
    │   ├── gateway_carrier1.xml
    │   ├── gateway_carrier2.xml
    │   └── gateway_emergency.xml
    ├── gateways_lab/               # Pasarelas de laboratorio
    │   └── gateway_test.xml
    └── dialplan/                   # Reglas de enrutamiento de
llamadas
        ├── mo_dialplan.xml         # Móvil Originado (saliente)
        ├── mt_dialplan.xml         # Móvil Terminado (entrante)
        └── emergency.xml
```

Referencia en el Archivo de 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"   # Referencia tu carpeta
de pasarelas a usar en este host
```

Qué sucede:

1. Ansible encuentra `gateways_folder: "gateways_prod"`
2. Copia todos los archivos de `hosts/Customer/group_vars/gateways_prod/` a `/etc/freeswitch/sip_profiles/`

3. Copia todos los archivos de `hosts/Customer/group_vars/dialplan/` al directorio de plantillas de OmniTAS
4. Los servicios cargan las configuraciones

Diferentes entornos: Usa diferentes carpetas por entorno:

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

Detalles de configuración: Ver https://docs.omnitouch.com.au/docs/repos/OmniCall

---

Ejemplo 3: Plantilla de Configuración Personalizada (OmniHSS)

El Servidor de Suscriptores en el Hogar acepta plantillas de configuración de tiempo de ejecución personalizadas.

Estructura de Archivos

```
hosts/Customer/
└── group_vars/
    └── hss_runtime.exs.j2        # Tu plantilla de configuración
HSS personalizada
```

Referencia en el Archivo de 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   # Referencia el
nombre de tu plantilla en group_vars
```

Qué sucede:

1. Ansible encuentra `hss_template_config: hss_runtime.exs.j2`
2. Busca en `hosts/Customer/group_vars/hss_runtime.exs.j2`
3. Lo plantilla con Jinja2 (puede usar `{{ inventory_hostname }}`, `{{ plmn_id.mcc }}`, etc.)
4. Despliega en `/etc/omnihss/runtime.exs`
5. Reinicia el servicio

Sin `hss_template_config`, se utiliza la plantilla predeterminada del rol.

Detalles de configuración: Ver https://docs.omnitouch.com.au/docs/repos/OmniCore

---

Ejemplo 4: Plantilla de Configuración Personalizada (OmniMME)

La Entidad de Gestión de Movilidad acepta plantillas de configuración de tiempo de ejecución personalizadas.

Estructura de Archivos

```
hosts/Customer/
└── group_vars/
    └── mme_runtime.exs.j2        # Tu plantilla de configuración
MME personalizada
```

Referencia en el Archivo de 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   # Referencia el
nombre de tu plantilla en group_vars
```

Qué sucede:

1. Ansible encuentra `mme_template_config: mme_runtime.exs.j2`
2. Busca en `hosts/Customer/group_vars/mme_runtime.exs.j2`
3. Lo plantilla con Jinja2 (puede usar `{{ inventory_hostname }}`, `{{ plmn_id.mcc }}`, etc.)
4. Despliega en `/etc/omnimme/runtime.exs`
5. Reinicia el servicio

Sin `mme_template_config`, se utiliza la plantilla predeterminada del rol.

Detalles de configuración: Ver https://docs.omnitouch.com.au/docs/repos/OmniCore

---

Ejemplo de Estructura de Directorios del Mundo Real

```
hosts/Customer/
├── host_files/
│   └── production.yml          # El archivo de hosts hace
referencia a los archivos de group_vars
└── group_vars/
    ├── smsc_controller.exs     # Plantilla personalizada de
OmniMessage
    ├── smsc_smpp.exs           # Plantilla personalizada SMPP de
OmniMessage
    ├── tas_runtime.exs.j2      # Plantilla personalizada de TAS
    ├── hss_runtime.exs.j2      # Plantilla personalizada de HSS
    ├── mme_runtime.exs.j2      # Plantilla personalizada de MME
    ├── dra_runtime.exs.j2      # Plantilla personalizada de DRA
    ├── pgwc_runtime.exs.j2     # Plantilla personalizada de PGW
    ├── dea_runtime.exs.j2      # Plantilla personalizada de DEA
    ├── upf_config.yaml         # Configuración de UPF
    ├── crm_config.yaml         # Configuración de CRM
    ├── stp.j2                  # Plantilla SS7 STP
    ├── hlr.j2                  # Plantilla SS7 HLR
    ├── camel.j2                # Plantilla SS7 CAMEL
    ├── ipsmgw.j2               # Plantilla IP-SM-GW
    ├── omnicore_smsc_ims.yaml.j2  # Configuración SMSC IMS
    ├── pytap.yaml              # Configuración TAP3
    ├── sip_profiles/           # Pasarelas SIP (carpeta)
    │   └── gateway_otw.xml
    └── dialplan/               # Reglas de enrutamiento de
llamadas (carpeta)
        ├── mo_dialplan.xml     # Móvil Originado
        ├── mt_dialplan.xml     # Móvil Terminado
        └── mo_emergency.xml    # Enrutamiento de emergencia
```

Parámetros Comunes que Hacen

Referencia a group_vars

Parámetro	Componente	Referencias
`smsc_template_config`	omnimessage	Archivo de plantilla Jinja2 (por ejemplo, `smsc_controller.exs`)
`smsc_smpp_template_config`	omnimessage_smpp	Archivo de plantilla Jinja2 (por ejemplo, `smsc_smpp.exs`)
`gateways_folder`	applicationserver	Nombre de carpeta (por ejemplo, `sip_profiles`)
Planes de Marcado (automático)	applicationserver	Carpeta `dialplan/` de XMLs de enrutamiento
`tas_template_config`	applicationserver	Archivo de plantilla Jinja2 (por ejemplo, `tas_runtime.exs.j2`)
`hss_template_config`	omnihss	Archivo de plantilla Jinja2 (por ejemplo, `hss_runtime.exs.j2`)
`mme_template_config`	omnimme	Archivo de plantilla Jinja2 (por ejemplo, `mme_runtime.exs.j2`)
`dra_template_config`	dra	Archivo de plantilla Jinja2 (por ejemplo, `dra_runtime.exs.j2`)
`pgwc_template_config`	pgwc	Archivo de plantilla Jinja2 (por ejemplo,

Parámetro	Componente	Referencias
		`pgwc_runtime.exs.j2`)
`frr_template_config`	omniupf	Archivo de plantilla Jinja2 (por ejemplo, `frr.conf.j2`)
Plantillas SS7	ss7 (varios roles)	Archivos de plantilla Jinja2 (por ejemplo, `stp.j2`, `hlr.j2`, `camel.j2`)
Configuraciones YAML	Varios componentes	Archivos de configuración directos (por ejemplo, `upf_config.yaml`, `crm_config.yaml`)

Puntos Clave

1. **group_vars contiene personalizaciones** - Sobrescrituras para configuraciones predeterminadas
2. **Referencia por nombre** - Usa parámetros como `smsc_template_config` o `gateways_folder`
3. **Las plantillas soportan Jinja2** - Accede a cualquier variable de Ansible con `{{ variable_name }}`
4. **Las carpetas despliegan todo** - Todos los archivos en las carpetas referenciadas son copiados
5. **Control de versiones de todo** - Comitea todos los group_vars a Git

Cuándo Usar group_vars

Usa group_vars para:

- Plantillas de configuración de componentes personalizados
- Definiciones de pasarelas SIP
- Planes de marcado de enrutamiento de llamadas
- Reglas de enrutamiento Diameter
- Configuraciones específicas del cliente que sobrescriben los valores predeterminados

No uses group_vars para:

- Configuración básica de hosts (IPs, nombres de host) - Usa el archivo de hosts
- Pruebas únicas - Usa variables específicas del host en el archivo de hosts
- Cambios temporales - Edita en el objetivo, comitea a group_vars si es permanente

---

Documentación Relacionada

- Referencia de Configuración - Todos los parámetros de hosts y lo que hacen
- Configuración del Archivo de Hosts - Cómo estructurar archivos de hosts
- **Configuración de OmniCall**: https://docs.omnitouch.com.au/docs/repos/OmniCall - Qué incluir en los archivos de configuración
- **Configuración de OmniCore**: https://docs.omnitouch.com.au/docs/repos/OmniCore - Detalles de configuración de componentes

Libretas de Utilidad

Descripción General

Este repositorio incluye varias libretas de utilidad para mantenimiento, monitoreo y tareas operativas. Estas complementan las libretas de despliegue principales con capacidades de gestión diaria.

Utilidad de Verificación de Salud

La utilidad de Verificación de Salud genera un informe HTML que muestra la salud del sistema, el estado del servicio, el tiempo de actividad y la información de la versión en todos los componentes de OmniCore.

Se ejecuta automáticamente como parte de la libreta `services/all.yml`.

Uso

Ejecución Manual

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

Salida

El informe se genera en `/tmp/health_check_YYYY-MM-DD HH:MM:SS.html`

Ábrelo en cualquier navegador web para verlo.

Contenidos del Informe

El informe HTML muestra:

Información del Host

- **Nombre del host y dirección IP**
- **Red/Subred** (de la variable `host_vm_network`, o N/A si no está configurada)
- **CPU** (número de vCPU)
- **RAM** (memoria total y libre)
- **Disco** (espacio total y libre de la partición raíz con porcentaje)
- **SO** (distribución y versión)

Estado del Servicio

- **Estado del servicio** (activo/inactivo con indicadores de color)
- **Tiempo de actividad**
- **Información de versión/liberación**

Pares de Diámetro HSS

- **Estado de conexión a la base de datos** (conectado/desconectado)
- **Conexiones de pares de Diámetro** (IP, host de origen, estado)
- Obtenido del endpoint de métricas HSS (puerto 9568)

Otras Utilidades Comunes

Configuración del Sistema Base

Rol Común (`services/common.yml`)

- Aplica la configuración base del sistema a todos los hosts
- Configura repositorios, claves SSH, zona horaria, NTP
- Configura la red y endurecimiento del sistema
- Ejecuta esto antes de desplegar servicios

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

Configurar Usuarios (`services/setup_users.yml`)

- Crea y configura cuentas de usuario en todos los hosts
- Gestiona claves SSH y privilegios sudo
- Asegura una configuración de usuario consistente

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

Reiniciar (`services/reboot.yml`)

- Reinicia de manera ordenada todos los hosts objetivo
- Espera a que los sistemas vuelvan a estar en línea (tiempo de espera de 5 minutos)
- Útil después de actualizaciones del kernel o cambios de configuración

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

Utilidades Operativas

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

- Genera un informe HTML de asignaciones de direcciones IP
- Muestra la topología de red completa desde el archivo de hosts
- Útil para documentación y solución de problemas

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

Respaldo de HSS (`util_playbooks/hss_backup.yml`)

- Respalda las tablas de la base de datos HSS
- Copia el volcado de MySQL a la máquina local de Ansible
- Solicitudes interactivas para la ruta de respaldo

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

Obtener Captura Local (`util_playbooks/getLocalCapture.yml`)

- Obtiene los dos archivos de captura de paquetes más recientes de todos los hosts
- Recupera archivos pcap de `/etc/localcapture/`
- Útil para depurar problemas de conectividad

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

Actualizar MTU (`util_playbooks/updateMtu.yml`)

- Actualiza la configuración de MTU de la interfaz de red
- Aplica cambios a través de netplan
- Útil para la configuración de tramas jumbo

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

Documentación Relacionada

- README Principal - Descripción general y cómo empezar
- Introducción al Despliegue de Ansible - Ejecución de libretas
- Configuración del Archivo de Hosts - Configura tu inventario
- Arquitectura de Despliegue - Visión general completa del sistema
- Sistema de Caché APT - Gestión de paquetes

Configuración del Archivo Hosts

Descripción General

El archivo hosts (también llamado archivo de inventario) es el documento de configuración central que define toda tu implementación de red celular. Especifica:

- Qué funciones de red desplegar
- Dónde se ejecutan (direcciones IP, segmentos de red)
- Cómo están configuradas (parámetros específicos del servicio)
- Configuraciones específicas del cliente (PLMN, credenciales, características)

Ubicación del Archivo

Los archivos hosts están organizados por cliente y entorno:

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

Ejemplo de Estructura del Archivo Hosts

Aquí hay un ejemplo simplificado que muestra las secciones clave:

```yaml
# Componentes 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'

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

# Servicios de Soporte
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'
```

Parámetros Comunes de Host

Configuración de Red

Cada host típicamente incluye:

```
pcscf:
  hosts:
    customer-pcscf01:
      ansible_host: 10.10.1.15      # Dirección IP para acceso SSH
      gateway: 10.10.1.1             # Puerta de enlace
predeterminada
      host_vm_network: "vmbr1"       # nombre de la NIC a usar en
el Hipervisor
```

Nota: Para obtener orientación sobre la planificación de direcciones IP y estrategias de segmentación de red, consulta el Estándar de Planificación IP que describe la arquitectura recomendada de cuatro subredes para implementaciones de OmniCore.

Usuarios de Proxmox: El parámetro `host_vm_network` especifica qué puente usar. Consulta Despliegue de VM/LXC en Proxmox para aprovisionamiento automatizado.

Asignación de Recursos de VM

Para servicios que necesitan recursos específicos:

```
num_cpus: 4                     # Núcleos de CPU
memory_mb: 8192                 # RAM en megabytes
proxmoxLxcDiskSizeGb: 50      # Tamaño del disco en GB
```

Parámetros Específicos del Servicio

Cada función de red tiene sus propios parámetros. Ejemplos:

MME:

```
mme_code: 1                    # Identificador de MME (1-255)
mme_gid: 1                     # ID del Grupo MME
network_name_short: Customer   # Nombre de la red (mostrado en
teléfonos)
network_name_long: Customer Network
tac_list: [600, 601, 602]     # Códigos de Área de Seguimiento
```

PGW:

```
ip_pools:                      # Grupos de IP para suscriptores
  - '100.64.16.0/24'
  - '100.64.17.0/24'
combined_CP_UP: false          # Plano de control/usuario separado
```

Para una explicación detallada de lo que controla cada variable, consulta: Referencia de Configuración

Servidor de Aplicaciones:

```
online_charging_enabled: true  # Habilitar integración OCS
tas_branch: "main"             # Rama de software a desplegar
gateways_folder: "gateways_prod"  # Configuración del gateway SIP
```

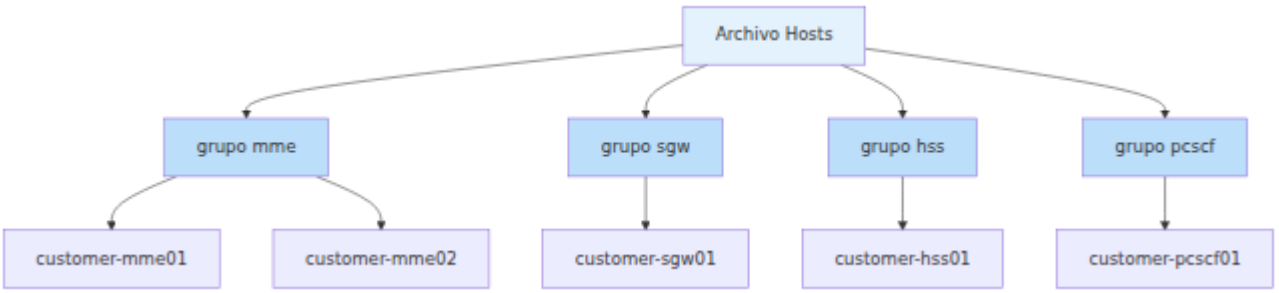
Sección de Variables Globales

La sección `all:vars` contiene configuraciones que se aplican a toda la implementación:

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

    # Identidad del Cliente
    customer_name_short: customer
    customer_legal_name: "Customer Inc."
    site_name: "Chicago DC1"
    region: US

    # Identificador PLMN (Red Móvil)
    plmn_id:
      mcc: '001'              # Código de País Móvil
      mnc: '01'               # Código de Red Móvil
      mnc_longform: '001'     # MNC con ceros a la izquierda

    # Nombres de Red
    network_name_short: Customer
    network_name_long: Customer Network

    # Repositorio APT
    # Nota: Si el grupo apt_cache_servers está definido con hosts,
    # use_apt_cache se establece en true de forma predeterminada y
apt_repo.apt_server
    # se establece automáticamente en la IP del primer servidor de
caché
    apt_repo:
      apt_server: "10.254.10.223"
      apt_repo_username: "customer"
      apt_repo_password: "secure-password"
    use_apt_cache: false

    # Zona Horaria
    TZ: America/Chicago
```

Comprendiendo los Grupos de Hosts

Ansible organiza los hosts en grupos que corresponden a roles:

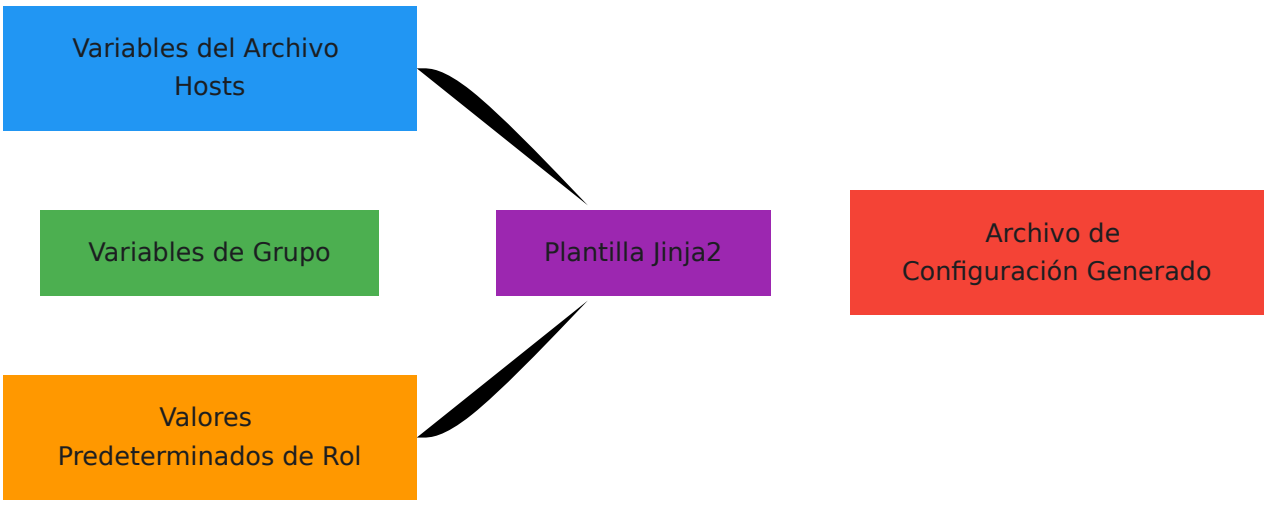


Cuando ejecutas un playbook dirigido a `mme`, se aplica a todos los hosts en la sección `mme:hosts:`.

Configuración con Plantillas Jinja2

Ansible utiliza **plantillas Jinja2** para generar archivos de configuración a partir de las variables definidas en tu archivo hosts y group_vars.

Cómo Funciona Jinja2



Ejemplo de Uso de Plantillas

El archivo hosts define:

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

Plantilla Jinja2 (en rol):

```yaml
# 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
```

Archivo de configuración generado:

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

Patrones Comunes de Jinja2

Accediendo a variables anidadas:

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

Lógica condicional:

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

Bucles:

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

Formateo:

```
# Rellenar con ceros a 3 dígitos
mnc{{ '%03d' | format(plmn_id.mnc|int) }}
```

Sobrescribiendo Variables con group_vars

Mientras que el archivo hosts define la infraestructura y configuraciones específicas del host, `group_vars` puede sobrescribir los valores predeterminados para grupos de hosts.

Consulta: Configuración de Variables de Grupo

Ejemplo Completo de Archivo Hosts

Aquí hay un ejemplo más completo (con datos sensibles oscurecidos):

```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"

# Servicios de Soporte
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
```

```
      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

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

    # Nombres de Red
    network_name_short: Customer
    network_name_long: Customer Network
    tac_list: [600, 601]

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

    # Configuración de Carga
    charging:
```

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

    # Reglas de Firewall
    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'

    # Configuración del Hipervisor (ejemplo de 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
```

Consulta Despliegue de VM/LXC en Proxmox para obtener detalles completos sobre la configuración y el establecimiento de Proxmox.

Referencias de Documentación del Producto

Para la configuración detallada de cada componente, consulta la documentación oficial del producto:

Componentes de OmniCore:

- **Documentación de OmniCore**: https://docs.omnitouch.com.au/docs/repos/OmniCore
- **OmniHSS** - Servidor de Suscriptores en Casa
- **OmniSGW** - Gateway de Servicio (Plano de control)
- **OmniPGW** - Gateway de Paquetes (Plano de control)
- **OmniUPF** - Función de Plano de Usuario
- **OmniDRA** - Agente de Enrutamiento Diameter
- **OmniTWAG** - Gateway de Acceso WLAN de Confianza

Componentes de OmniCall:

- **Documentación de OmniCall**: https://docs.omnitouch.com.au/docs/repos/OmniCall
- **OmniTAS** - Servidor de Aplicaciones IMS (VoLTE/VoNR)
- **OmniCall CSCF** - Funciones de Control de Sesiones de Llamadas
- **OmniMessage** - Centro de SMS
- **OmniMessage SMPP** - Soporte para el Protocolo SMPP
- **OmniSS7** - Pila de Señalización SS7
- **VisualVoicemail** - Buzón de Voz

OmniCharge/OmniCRM:

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

Documentación Relacionada

- Introducción al Despliegue de Ansible - Proceso general de despliegue
- Referencia de Configuración - **Guía completa de todas las variables de configuración**
- Configuración de Variables de Grupo - Sobrescribiendo configuraciones predeterminadas
- Estándar de Planificación IP - **Arquitectura de red y directrices de asignación de IP**

- Configuración de Netplan - **IPs secundarias y configuración avanzada de red**
- Sistema de Caché APT - Distribución de paquetes
- Servidor de Licencias - Gestión de licencias
- Descripción General de la Arquitectura de Despliegue - Vista completa del sistema

Próximos Pasos

1. Crea tu archivo hosts basado en esta plantilla
2. Define tu PLMN e identidad de red
3. Configura el acceso al repositorio APT
4. Configura el servidor de licencias
5. Personaliza con group_vars según sea necesario
6. Despliega con playbooks de Ansible

Estándar de Planificación de IP de OmniCore

Descripción general

Este documento describe el enfoque estándar de planificación de IP para implementaciones de OmniCore. La arquitectura requiere **cuatro subredes internas** para segmentar adecuadamente las funciones de red por seguridad, rendimiento y claridad operativa.

Requisitos de Asignación de IP

Asignación Estándar: Cuatro Subredes /24

Cada implementación de OmniCore requiere cuatro subredes distintas para la red interna:

1. **Red de Núcleo de Paquetes** - Primera /24
2. **Red de Señalización** - Segunda /24
3. **Red Interna IMS** - Tercera /24
4. **Red Pública de UE** - Cuarta /24

Importante: Estas son Recomendaciones, No Requisitos

La asignación de subredes descrita en este documento es una **mejor práctica recomendada** para organizar las implementaciones de OmniCore. Sin embargo, la arquitectura es **completamente flexible**:

- **Todos los hosts en una subred**: Puedes colocar todos los componentes en una sola subred si eso se ajusta a tus necesidades de implementación.
- **Cada tipo de host en su propia subred**: Puedes crear subredes separadas para cada tipo de componente (una para MMEs, una para HSS, etc.)
- **Agrupaciones personalizadas**: Puedes organizar los hosts en cualquier estructura de subred que tenga sentido para tus requisitos específicos.
- **Mezclar IPs internas y públicas**: Algunos hosts pueden usar direcciones internas (RFC 1918) mientras que otros usan IPs públicas, todo dentro de la misma implementación.

El enfoque recomendado de cuatro subredes proporciona una **aislamiento de seguridad, gestión de tráfico y claridad operativa** óptimos, por lo que lo sugerimos para implementaciones en producción. Sin embargo, debes adaptar el plan de IP para que se ajuste a tu topología de red específica, espacio de direcciones disponible y requisitos operativos.

Desglose de Segmentos de Red

1. Red de Núcleo de Paquetes (Primera /24)

Propósito: Elementos del plano de usuario y del plano de control central

Componentes:

- OmniMME (Entidad de Gestión de Movilidad)
- OmniSGW (Puerta de Enlace de Servicio)
- OmniPGW-C (Plano de Control de Puerta de Enlace de PDN)
- OmniUPF/PGW-U (Función de Plano de Usuario / Puerta de Enlace de PDN)

Ejemplo: `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. Red de Señalización (Segunda /24)

Propósito: Funciones de señalización Diameter, políticas, facturación y gestión

Componentes:

- OmniHSS (Servidor de Suscriptores Local)
- OmniCharge OCS (Sistema de Facturación en Línea)
- OminiHSS PCRF (Función de Reglas de Políticas y Facturación)
- OmniDRA DRA (Agente de Enrutamiento Diameter)
- Servidores DNS
- Servidores TAP3/CDR
- Monitoreo/OAM
- Captura de SIP
- Servidor de Licencias
- Monitor de RAN
- Omnitouch Warning Link CBC (Centro de Difusión Celular) - si se despliega
- Servidores de Caché APT - si se despliega

Ejemplo: `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. Red Interna IMS (Tercera /24)

Propósito: Señalización y servicios centrales de IMS (señalización SIP interna)

Componentes:

- OmniCSCF S-CSCF (Función de Control de Sesiones de Llamada de Servicio)
- OmniCSCF I-CSCF (Función de Control de Sesiones de Llamada Interrogante)
- OmniTAS (Servidor de Aplicaciones de Telefonía / Servidor de Aplicaciones)
- OmniMessage (Controlador de SMS, SMPP, IMS)
- OmniSS7 STP (Punto de Transferencia de Señalización SS7)
- OmniSS7 HLR (Registro de Ubicación del Hogar) - para 2G/3G
- OmniSS7 IP-SM-GW (MAP SMSc)
- OmniSS7 Puerta de Enlace CAMEL

Ejemplo: `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. Red Pública de UE (Cuarta /24)

Propósito: Servicios orientados al usuario como IMS y DNS

Componentes:

- OmniCSCF P-CSCF (Función de Control de Sesiones de Llamada Proxy)
- Servidores XCAP
- Servidores de Correo de Voz Visual
- DNS del Cliente

Ejemplo: `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"
```

Métodos de Implementación

OmniCore admite dos métodos principales para implementar esta segmentación de red:

Método 1: Interfaces de Red Físicas/Virtuales (Recomendado para Producción)

Utiliza NICs separadas o puentes virtuales para cada segmento de red. Esto proporciona el aislamiento más fuerte y es el enfoque recomendado para implementaciones en producción.

Ejemplo:

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

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

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

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

Método 2: Segmentación Basada en VLAN

Utiliza una única interfaz física con etiquetado de VLAN para separar redes. Esto es adecuado para implementaciones más pequeñas o cuando las NICs físicas son limitadas.

Ejemplo:

```yaml
# Todos los componentes usan vmbr12 con diferentes VLANs
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"
```

Configuración de Red:

- Configura VLANs en el switch físico
- Etiqueta el tráfico apropiadamente a nivel del hipervisor
- Rutea entre VLANs en el gateway/firewall

Ejemplo de Mapeo de VLAN:

```
VLAN 10: 10.x.1.0/24 (Núcleo de Paquetes)
VLAN 20: 10.x.2.0/24 (Señalización)
VLAN 30: 10.x.3.0/24 (IMS Interno)
VLAN 40: 10.x.4.0/24 (UE Pública)
```

Trabajando con Direcciones IP Públicas

Descripción general

Muchas implementaciones de OmniCore requieren que algunos componentes tengan direcciones IP públicas para conectividad externa, como:

- **DRA** - Para señalización diameter de roaming con operadores externos
- **SGW/PGW de Roaming** - Para tráfico GTP de socios de roaming
- **ePDG** - Para llamadas WiFi (túneles IPsec desde UEs)
- **Puerta de Enlace SMSC** - Para conexiones SMPP a agregadores de SMS externos
- **P-CSCF** (en algunas implementaciones) - Para registro SIP directo de UE

Cómo Asignar IPs Públicas

Las IPs públicas se manejan **exactamente de la misma manera que las IPs internas** en tus archivos de inventario de hosts. Simplemente especifica la dirección IP pública en el campo `ansible_host` junto con la puerta de enlace y la máscara de red apropiadas.

Ejemplo: SGW/PGW de Roaming con IPs Públicas

```yaml
sgw:
  hosts:
    # SGWs internos en red privada
    opt-site-sgw01:
      ansible_host: 10.4.1.25
      gateway: 10.4.1.1
      host_vm_network: "v400-omni-packet-core"

    # SGWs de roaming con IPs públicas
    opt-site-roaming-sgw01:
      ansible_host: 203.0.113.10
      gateway: 203.0.113.9
      netmask: 255.255.255.248       # subred /29
      host_vm_network: "498-public-servers"
      in_pool: False
      cdrs_enabled: True

smf:  # PGWs
  hosts:
    # PGW de roaming con IP pública
    opt-site-roaming-pgw01:
      ansible_host: 203.0.113.20
      gateway: 203.0.113.17
      netmask: 255.255.255.240       # subred /28
      host_vm_network: "497-public-services-LTE"
      in_pool: False
      ip_pools:
        - '100.64.24.0/22'
```

Ejemplo: DRA con IP Pública

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

Ejemplo: ePDG con IP Pública

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

Mezclando IPs Internas y Públicas

Es común tener una mezcla de IPs internas y públicas dentro del mismo grupo de componentes. Por ejemplo:

- SGWs internos para sitios locales usando GTP
- SGWs públicas específicamente para tráfico de roaming de operadores externos
- El mismo PGW-C puede gestionar tanto SGWs internas como externas

La arquitectura de OmniCore maneja esto sin problemas: simplemente configura cada host con su direccionamiento IP apropiado.

Servidor de Licencias

Descripción General

El Servidor de Licencias gestiona la activación de características para todos los componentes de Omnitouch. Cada componente valida su licencia al iniciar y periódicamente durante su operación.

Configuración

1. Definir en el Archivo de 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: "Customer Name"
    license_server_api_urls: ["https://10.10.2.150:8443/api"]
    license_enforced: true
```

2. Proporcionar el Archivo de Licencia

Coloque `license.json` (proporcionado por Omnitouch) en `hosts/Customer/group_vars/`

3. Desplegar

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

Puede verificar el estado de todas las licencias navegando a https://license_server .

Requisitos de Red

Configuración del Cortafuegos

Los cortafuegos del sitio del cliente deben estar configurados para permitir tráfico HTTPS (puerto 443) a los siguientes servidores de validación de licencias de Omnitouch:

Nombre de Host	Dirección IP	Propósito
time.omnitouch.com.au	160.22.43.18	Servidor de validación de licencias 1
time.omnitouch.com.au	160.22.43.66	Servidor de validación de licencias 2
time.omnitouch.com.au	160.22.43.114	Servidor de validación de licencias 3

Reglas de salida requeridas:

- Protocolo: HTTPS (TCP/443)
- Destino: 160.22.43.18, 160.22.43.66, 160.22.43.114
- Dirección: Saliente

Requisitos de DNS

El servidor de licencias requiere resolución DNS funcional para comunicarse con la infraestructura de validación de licencias de Omnitouch.

Configuración de DNS requerida:

- El servidor de licencias debe tener acceso a servidores DNS públicos
- Configure DNS para usar uno de los siguientes:
  - 1.1.1.1 (Cloudflare - soporta DNS seguro)
  - 8.8.8.8 (Google Public DNS)
- No use servidores DNS internos/corporativos para el servidor de licencias

Nota: Los servidores de licencias de Omnitouch utilizan DNS seguro (DoH/DoT). Usar servidores DNS públicos asegura una validación adecuada de DNSSEC y previene problemas con la interceptación de DNS por dispositivos de seguridad.

Documentación Relacionada

- Referencia de Configuración
- Configuración del Archivo de Hosts

Configuración de Netplan

Descripción general

OmniCore puede configurar automáticamente las interfaces de red en las VMs desplegadas utilizando netplan. Esto es útil para:

- Configurar la interfaz de gestión principal (eth0)
- Agregar interfaces secundarias para IPs públicas, conexiones de emparejamiento o tráfico dedicado
- Configurar rutas estáticas para destinos específicos

Habilitación de la Configuración de Netplan

Para habilitar la configuración automática de netplan para un host, agrega la variable `netplan_config` que apunte a una plantilla Jinja2 en tu carpeta `group_vars`:

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

La plantilla se obtendrá de `hosts/<customer>/group_vars/netplan.yaml.j2`.

Referencia de la Plantilla

Aquí está la plantilla completa `netplan.yaml.j2` con comentarios que explican cada sección:

```
network:
  version: 2
  ethernets:
    # Interfaz primaria - utiliza ansible_host y gateway del
inventario
    eth0:
      addresses:
      - "{{ ansible_host }}/{{ mask_cidr | default(24) }}"
      nameservers:
        addresses:
{% if 'dns' in group_names %}
        # Si este host ES un servidor DNS, usa DNS externo para
evitar dependencia circular
        - 8.8.8.8
{% else %}
        # De lo contrario, usa servidores DNS del grupo 'dns' en
el inventario
{% 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 secundarias - recorre el diccionario
secondary_ips del inventario
    # Nomenclatura de 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 %}
      # Rutas estáticas para esta interfaz - cada ruta utiliza la
puerta de enlace de esta interfaz
      routes:
{% for route in nic_config.routes %}
      - to: "{{ route }}"
```

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

Puntos clave:

- `ansible_host` y `gateway` provienen de la entrada del inventario del host
- Los servidores DNS se obtienen dinámicamente de los hosts en el grupo `dns`
- Las interfaces secundarias se nombran `ens19`, `ens20`, etc. para coincidir con la nomenclatura de NIC de Proxmox
- Cada IP secundaria puede tener su propia puerta de enlace y rutas estáticas

Configuración de la Interfaz Primaria

La interfaz primaria (eth0) se configura automáticamente utilizando:

- `ansible_host` - La dirección IP
- `gateway` - La puerta de enlace predeterminada
- `mask_cidr` - Máscara de red (predeterminado a 24)

Los servidores DNS se configuran automáticamente a:

- Hosts en el grupo `dns` (utiliza sus IPs `ansible_host`)
- Se retrocede a `8.8.8.8` si el host es él mismo un servidor DNS

Interfaces Secundarias

Para hosts que requieren interfaces de red adicionales (IPs públicas, emparejamiento, etc.), utiliza la configuración `secondary_ips`.

Esquema

```
secondary_ips:
  <logical_name>:
    ip_address: <ip_address>
    gateway: <gateway_ip>
    host_vm_network: <proxmox_bridge>
    vlanid: <vlan_id>
    routes:                    # Opcional - rutas estáticas a
través de esta interfaz
      - '<destination_cidr>'
      - '<destination_cidr>'
```

Nomenclatura de Interfaces

Las interfaces secundarias se nombran automáticamente utilizando el esquema de nomenclatura predecible de Ubuntu:

- Primera interfaz secundaria: `ens19`
- Segunda interfaz secundaria: `ens20`
- Tercera interfaz secundaria: `ens21`
- Y así sucesivamente...

Esto coincide con los nombres de las interfaces asignados por Proxmox al agregar NICs adicionales a una VM.

Ejemplo de Configuración

```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'
```

Salida Generada de Netplan

La configuración anterior genera:

```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"
```

Integración con Proxmox

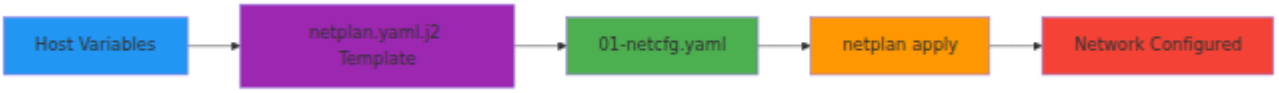
Al utilizar el playbook `proxmox.yml`, las NICs secundarias se crean automáticamente en la VM:

1. **Nuevas VMs**: Las NICs secundarias se agregan durante la provisión inicial
2. **VMs Existentes**: Las NICs secundarias se agregan y la VM se reinicia para aplicar los cambios

La configuración de Proxmox utiliza:

- `host_vm_network` - El puente al que se adjunta la NIC
- `vlanid` - Etiqueta VLAN para la interfaz

Cómo Funciona



1. Las variables del archivo de hosts se pasan a la plantilla Jinja2
2. La plantilla se renderiza en `/etc/netplan/01-netcfg.yaml`
3. Cualquier configuración de netplan existente se elimina para evitar conflictos
4. `netplan apply` activa la configuración
5. Las direcciones IP se verifican con `ip addr show`

Casos de Uso Comunes

Diameter Edge Agent (DEA) con IP Pública

```
<hostname>:
  ansible_host: 10.0.1.100           # IP de gestión interna
  gateway: 10.0.1.1
  netplan_config: netplan.yaml.j2
  secondary_ips:
    diameter_roaming:
      ip_address: 192.0.2.50         # IP pública para socios de
roaming
      gateway: 192.0.2.1
      host_vm_network: "vmbr0"
      vlanid: "200"
      routes:
        - '198.51.100.0/24'          # Red de socios de roaming
```

PGW con Interfaz S5/S8

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

Servidor Multihomed con Redes de Gestión y Datos Separadas

```
<hostname>:
  ansible_host: 10.0.1.100           # Red de gestión
  gateway: 10.0.1.1
  netplan_config: netplan.yaml.j2
  secondary_ips:
    data_network:
      ip_address: 10.0.2.100         # Red de datos
      gateway: 10.0.2.1
      host_vm_network: "ovsbr2"
      vlanid: "200"
    backup_network:
      ip_address: 10.0.3.100         # Red de respaldo
      gateway: 10.0.3.1
      host_vm_network: "ovsbr3"
      vlanid: "300"
```

Referenciando IPs Secundarias en Plantillas

Puedes referenciar direcciones IP secundarias en otras plantillas Jinja2 y archivos de configuración.

En el Mismo Host

Al configurar un servicio en el mismo host que tiene IPs secundarias, puedes referenciar directamente o usar `inventory_hostname`:

```
# Referencia directa (más simple)
{{ secondary_ips.diameter_public_ip.ip_address }}

# O explícitamente a través de inventory_hostname (mismo
resultado)
{{ hostvars[inventory_hostname]['secondary_ips']
['diameter_public_ip']['ip_address'] }}

# Acceder a otras propiedades
{{ secondary_ips.diameter_public_ip.gateway }}
{{ secondary_ips.diameter_public_ip.vlanid }}
```

Desde Otro Host

Cuando necesitas referenciar una IP secundaria de un host *diferente* (por ejemplo, configurando una conexión de emparejamiento), utiliza `hostvars` con el nombre del host de destino:

```
# Referencia al primer host en el grupo dra
{{ hostvars[groups['dra'][0]]['secondary_ips']
['diameter_public_ip']['ip_address'] }}

# Recorre todos los hosts DRA y obtiene sus IPs públicas
{% for host in groups['dra'] %}
{% if hostvars[host]['secondary_ips'] is defined %}
  - {{ hostvars[host]['secondary_ips']['diameter_public_ip']
['ip_address'] }}
{% endif %}
{% endfor %}
```

Ejemplo: Configuración de Emparejamiento DRA

Configura un par de diámetro para vincularse a su propia IP pública:

```
# En dra_config.yaml.j2 - usa inventory_hostname para el host
actual
peers:
  - name: external_peer
    # Vincular a la IP pública de diámetro de este host
    local_ip: {{ hostvars[inventory_hostname]['secondary_ips']
['diameter_public_ip']['ip_address'] }}
    remote_ip: 198.51.100.50
    port: 3868
```

Comprobando si Existen IPs Secundarias

Siempre verifica si la variable existe antes de usarla:

```
{% 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 %}
```

Solución de Problemas

Verificar Nombres de Interfaces

SSH a la VM y verifica los nombres de las interfaces:

```
ip link show
```

Salida esperada para una VM con dos interfaces secundarias:

```
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> ...
```

Verificar la Configuración de Netplan

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

Aplicar Netplan Manualmente

```
netplan apply
```

Depurar Netplan

```
netplan --debug apply
```

Verificar Rutas

```
ip route show
```

Documentación Relacionada

- Configuración del Archivo de Hosts - Configuración del inventario de hosts
- Despliegue de Proxmox VM/LXC - Provisión de VM
- Referencia de Configuración - Todas las variables de configuración

Implementación de VM/LXC en Proxmox

La mayoría de nuestros clientes ejecutan la pila de OmniCore en Proxmox, esta guía explica en detalle cómo usar los plays de `proxmox` para configurar su entorno utilizando Proxmox.

Seguimos apoyando VMware, HyperV y la nube (Actualmente Vultr / AWS / GCP) para implementaciones.

Ver También:

- Configuración del Archivo de Hosts - Definir VMs a implementar
- Estándar de Planificación de IP - Directrices para la asignación de direcciones IP
- Configuración de Netplan - IPs secundarias y configuración de múltiples NIC
- Arquitectura de Implementación - Flujo de trabajo completo de implementación

LXC vs VM

Contenedores LXC:

- Livianos, comparten el núcleo del host
- Inicio rápido, bajo overhead
- Aislamiento limitado
- No pueden ejecutar núcleos o módulos de núcleo personalizados
- **No son adecuados para implementaciones en producción**
- **No pueden ejecutar UPF** (requiere módulos de núcleo/dispositivos TUN)

VMs (KVM):

- Virtualización completa con núcleo dedicado
- Aislamiento completo

- Pueden ejecutar módulos de núcleo y redes personalizadas
- Mayor overhead de recursos
- **Recomendado para producción**
- **Requerido para implementaciones de UPF**

Casos de Uso:

- **VMs**: Sitios de producción, UPF, todas las funciones de red
- **LXC**: Entornos de laboratorio/prueba, servicios livianos (apt-cache, monitoreo)

Configuración de Proxmox

1. Crear Token de API

```
# En la interfaz de Proxmox: Datacenter → Permisos → Tokens de API
# Crear token: root@pam!<TokenName>
# Copiar el secreto del token (se muestra una vez)
```

2. Crear Plantilla de VM Cloud-Init (solo para VMs)

Ejecute este script en el host de Proxmox. Descarga la imagen de nube de Ubuntu y crea una plantilla con las credenciales de usuario de 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 "=== Descargando imagen de nube de Ubuntu ==="
cd /var/lib/vz/template/iso
wget -N "$IMAGE_URL"

echo "=== Limpiando plantilla antigua ==="
qm destroy $TEMPLATE_ID --purge 2>/dev/null || true

echo "=== Habilitando almacenamiento de snippets ==="
pvesm set local --content images,vztmpl,iso,backup,snippets

echo "=== Creando datos de usuario de 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 "=== Creando plantilla 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 "=== Plantilla $TEMPLATE_ID creada con éxito ==="
```

Notas:

- La plantilla proporciona un inicio de sesión de respaldo: `omnitouch` / `password` (para acceso a la consola si cloud-init falla)
- Al clonar a través de Ansible, las credenciales se sobrescriben desde `local_users` en su archivo de hosts:
  - Nombre de usuario: Clave del primer usuario de `local_users`
  - Contraseña: Campo `password` del primer usuario (por defecto es 'password' si no se establece)
  - Clave SSH: Campo `public_key` del primer usuario
- `--vga std` asegura que la consola web de Proxmox funcione
- La bandera `-N` en wget solo descarga si es más nueva que la copia local

Alternativa: Plantilla Manual desde ISO

Si las imágenes de nube no están disponibles o necesita una instalación personalizada:

Paso 1: Crear VM a través de la Interfaz Web

- Crear Nueva VM → ID de VM 9000, Nombre: ubuntu-2404-template
- SO: Subir ISO de Ubuntu Server o usar ISO existente
- Sistema: Predeterminado (Controlador SCSI: VirtIO SCSI)
- Discos: 32GB, Bus: SCSI
- CPU: 2 núcleos
- Memoria: 2048 MB
- Red: VirtIO, puente vmbr0
- Iniciar VM e instalar Ubuntu Server

Paso 2: Dentro de la VM - Limpiar y preparar

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

# Limpiar datos específicos de la máquina
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

# Limpiar historial de bash y apagar
history -c
sudo poweroff
```

Paso 3: Agregar Cloud-Init y Convertir a Plantilla

- Seleccionar VM → Hardware → Agregar → Unidad CloudInit (seleccionar almacenamiento, por ejemplo, local-lvm)
- Cloud-Init → Usuario: `omnitouch`, Contraseña: `password`
- Hardware → Opciones → Agente QEMU → Habilitar
- Hacer clic derecho en la VM → Convertir a Plantilla

3. Descargar Plantilla LXC (solo para LXC)

```
# En la shell del nodo Proxmox:
pveam update
pveam download local ubuntu-24.04-standard_24.04-2_amd64.tar.zst
```

Configuración del Archivo de Hosts

Para Implementación 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  # opcional
      pve-node-02:
        # ... configuración del segundo nodo

    # Credenciales de usuario - el primer usuario se utiliza para
cloud-init de VM
    local_users:
      admin_user:
        name: Admin User
        public_key: "ssh-rsa AAAA..."
        password: "optional-password"  # por defecto es 'password'
si no se establece

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

Para Implementación de LXC (proxmox_lxc.yml)

```yaml
all:
  vars:
    proxmoxServerAddress: 192.168.1.100
    proxmoxServerPort: 8006
    proxmoxNodeName: ['pve-node-01', 'pve-node-02']  # único o
lista
    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

    # Credenciales de usuario - el primer usuario se utiliza para
el acceso inicial a VM/LXC
    local_users:
      admin_user:
        name: Admin User
        public_key: "ssh-rsa AAAA..."
        password: "optional-password"  # por defecto es 'password'
si no se establece

apt_cache_servers:
  hosts:
    site-cache:
      ansible_host: 192.168.1.20
      gateway: 192.168.1.1
      vlanid: "100"  # opcional
      proxmoxLxcDiskSizeGb: 120  # anulación por host
```

Uso

Implementar VMs

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

Implementar Contenedores LXC

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

Eliminar VMs/LXCs

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

Comportamiento

proxmox.yml

- Verifica si ya existe una VM con el mismo nombre en Proxmox
- Distribuye VMs entre nodos usando round-robin
- Clona desde la plantilla
- Configura IP estática, etiquetas y cloud-init
- **Establece las credenciales de usuario de cloud-init desde la primera entrada de `local_users`**
- Soporta etiquetado VLAN

proxmox_lxc.yml

- Verifica que el contenedor no exista por nombre o IP

- Distribuye LXCs entre nodos usando round-robin
- Crea contenedor con IP estática
- **Crea automáticamente la primera cuenta de `local_users` con acceso sudo y clave SSH**
- Configura netplan para la red
- Inicia automáticamente los contenedores
- Excluye hosts UPF

proxmox_delete.yml

- Detiene y elimina VM/LXC que coincidan con el nombre de host del inventario
- Busca en todos los nodos configurados
- Fuerza la detención después de 20 segundos

Distribución y Etiquetado de VM/LXC

Distribución Round-Robin

Las VMs y LXCs se distribuyen automáticamente entre los nodos de Proxmox utilizando lógica de round-robin (módulo):

Ejemplo con 3 hipervisores y 5 MMEs:

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

Cómo funciona:

1. El playbook identifica el grupo de roles del host (por ejemplo, `mme`, `sgw`, `hss`)

2. Calcula el índice del host dentro de ese grupo (basado en 0)
3. Utiliza la operación de módulo: `host_index % number_of_nodes`
4. Selecciona el hipervisor según el resultado

Configuración:

```
# Para VMs (proxmox.yml) - definir múltiples servidores
proxmoxServers:
  pve-node-01: { ... }
  pve-node-02: { ... }
  pve-node-03: { ... }

# Para LXCs (proxmox_lxc.yml) - listar múltiples nodos
proxmoxNodeName: ['pve-node-01', 'pve-node-02', 'pve-node-03']
```

Etiquetado Automático

Las VMs y LXCs se etiquetan automáticamente con:

- **Nombres de Rol/Grupo**: Todos los grupos de Ansible a los que pertenece el host
- **Nombre del Sitio**: La variable `site_name`

Ejemplo:

```
site_name: "melbourne-prod"

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

Resultado: VM/LXC etiquetada con: `mme`, `melbourne-prod`

Las etiquetas son visibles en la interfaz de Proxmox y son útiles para filtrado/organización.

Anulaciones por Host

Anule los valores predeterminados en hosts específicos:

```yaml
hosts:
  high-spec-host:
    ansible_host: 192.168.1.50
    gateway: 192.168.1.1
    proxmoxLxcCores: 8        # anular núcleos
    proxmoxLxcMemoryMb: 16384  # anular memoria
    proxmoxLxcDiskSizeGb: 100  # anular disco
```

Libretas de Utilidad

Las libretas de utilidad proporcionan herramientas operativas para gestionar la infraestructura de OmniCore desplegada. Estas libretas se encuentran en el directorio `util_playbooks/` y se pueden ejecutar de forma independiente para realizar tareas comunes de mantenimiento y resolución de problemas.

Referencia Rápida

Libreta	Propósito
`health_check.yml`	Generar un informe de salud completo para todos los servicios
`restore_hss.yml`	Restaurar la base de datos HSS y/o la configuración desde una copia de seguridad
`ip_plan_generator.yml`	Generar documentación de red con diagramas de Mermaid
`get_ports.yml`	Auditar puertos abiertos y servicios en escucha en todos los hosts
`getLocalCapture.yml`	Recuperar archivos de captura de paquetes de los hosts
`delete_local_user.yml`	Eliminar una cuenta de usuario local de todos los hosts
`updateMtu.yml`	Establecer MTU en 9000 (tramas jumbo) en las interfaces de red
`systemctl status.yml`	Verificar el estado del servicio en los componentes EPC

Verificación de Salud

Archivo: `util_playbooks/health_check.yml`

Genera un informe de salud HTML completo que cubre todos los servicios de OmniCore y OmniCall desplegados.

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

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

Información Recopilada

Componente	Datos Recopilados
Todos los servicios	Estado del servicio, versión, tiempo de actividad
OmniHSS	Estado de la base de datos, conexiones de pares Diameter
OmniDRA	Conexiones de pares Diameter y estado
OmniTAS	Llamadas activas, sesiones, registros, uso de CPU
OCS	Estado de replicación de KeyDB

Restauración de HSS

Archivo: `util_playbooks/restore_hss.yml`

Restaura OmniHSS a partir de archivos de copia de seguridad. Soporta la restauración solo de la base de datos, solo de la configuración, o de ambos.

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

Formatos de Archivos de Copia de Seguridad

Tipo	Patrón de Nombre de Archivo	Contenidos
Base de datos	`hss_dump_<hostname>_<timestamp>.sql`	Volcado de MySQL de la base de datos `omnihss`
Configuración	`hss_<hostname>_<timestamp>.tar.gz`	Archivo del directorio `/etc/omnihss`

Generador de Plan de IP

Archivo: `util_playbooks/ip_plan_generator.yml`

Genera documentación de red a partir del inventario, incluyendo:

- Asignaciones de IP de hosts (NICs primarias y secundarias)
- Visión general del segmento de red
- Diagramas de conectividad de interfaces (Diameter, GTP, PFCP, SIP, SS7)

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

Archivos de Salida

Archivo	Formato	Descripción
`/tmp/ip_plan_<customer>_<site>.md`	Markdown	Documentación basada en texto
`/tmp/ip_plan_<customer>_<site>.html`	HTML	Diagrama interactivo con capas filtrables

Auditoría de Puertos

Archivo: `util_playbooks/get_ports.yml`

Audita todos los puertos en escucha a través del despliegue y genera documentación.

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

Archivos de Salida

Archivo	Descripción
`/tmp/all_ports.csv`	CSV con nombre de host, IP, protocolo, puerto, servicio
`./open_ports.rst`	Tabla reStructuredText para documentación Sphinx

Datos Recopilados

Campo	Descripción
Nombre de Host	Nombre de host del inventario
IP	Dirección IP `ansible_host` del host
Versión de IP	IPv4 o IPv6
Transporte	TCP o UDP
Puerto	Número de puerto en escucha
Servicio	Nombre del proceso

Recuperación de Captura Local

Archivo: `util_playbooks/getLocalCapture.yml`

Recupera los dos archivos de captura de paquetes más recientes del directorio `/etc/localcapture` de cada host.

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

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

Gestión de Usuarios

Archivo: `util_playbooks/delete_local_user.yml`

Elimina una cuenta de usuario local de todos los hosts en el inventario.

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

Solicitud: Ingrese el nombre de usuario a eliminar cuando se le solicite.

Configuración de MTU

Archivo: `util_playbooks/updateMtu.yml`

Establece el MTU en 9000 (tramas jumbo) en la interfaz `ens160` en todos los hosts.

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

Nota: Esta libreta está codificada para la interfaz `ens160`. Modifique la libreta si su entorno utiliza nombres de interfaz diferentes.

Ejecución de Libretas de Utilidad

Sintaxis Básica

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

Opciones Comunes

Opción	Descripción
`-i <inventory>`	Especificar archivo de inventario
`--limit <hosts>`	Limitar a hosts o grupos específicos
`-v` / `-vv` / `-vvv`	Aumentar la verbosidad
`--check`	Ejecución en seco (mostrar lo que cambiaría)
`--diff`	Mostrar diferencias de archivos

Ejemplos

```
# Ejecutar verificación de salud en producción
ansible-playbook -i hosts/acme/host_files/production.yml
util_playbooks/health_check.yml

# Restaurar HSS en un host específico
ansible-playbook -i hosts/acme/host_files/production.yml
util_playbooks/restore_hss.yml --limit hss01

# Generar plan de IP con salida detallada
ansible-playbook -i hosts/acme/host_files/production.yml
util_playbooks/ip_plan_generator.yml -v
```