Núcleo 5G (5GC)

El núcleo 5G Standalone (SA) se despliega mediante el playbook `services/5gc.yml`. Un único rol compartido, `5gc_nf`, instala cada función de red (NF). El rol está parametrizado por NF. Esta página explica ese rol, las NFs que construye y la configuración que cada una lee.

Visión general

En lugar de un rol por NF, OmniCore utiliza **un rol (`5gc_nf`) para las once NFs de 5GC**. El playbook invoca el rol una vez por NF y pasa una variable `nf_package` que selecciona el paquete Debian, el servicio systemd y la plantilla de configuración Jinja2. Esto mantiene la instalación, la versionado de configuración, la licencia y la lógica de reinicio idénticas en cada NF.

```
- name: Setup AMF
  hosts: amf
  roles:
    - {role: 5gc_nf, nf_package: omniamf, become: yes}
```

Funciones de Red

Cada NF tiene su propio grupo de inventario, paquete, servicio systemd y plantilla de configuración (`roles/5gc_nf/templates/<nf_package>.j2`).

NF	`nf_package`	Grupo de inventario	Plantilla de configuración	Rol
Función de Repositorio de Red	`omninrf`	`nrf`	`omninrf.j2`	Descubrimier de servicios / Registro de N
Gestión de Acceso y Movilidad	`omniamf`	`amf`	`omniamf.j2`	Señalización N1/N2, regist gestión de conexión
Servidor de Autenticación	`omniausf`	`ausf`	`omniausf.j2`	Autenticación 5G-AKA / EAP AKA'
Función de Soporte de Vinculación	`omnibsf`	`bsf`	`omnibsf.j2`	Vinculación P para una sesi PDU
Función de Carga	`omnichf`	`chf`	`omnichf.j2`	Carga en línea/fuera de línea convergente
Selección de Slice de Red	`omnissf`	`nssf`	`omnissf.j2`	NSSAI / selección de slice
Función de Control de Políticas	`omnipcf`	`pcf`	`omnipcf.j2`	Reglas de políticas y Qo
Proxy de Comunicación de Servicios	`omniscp`	`scp`	`omniscp.j2`	Enrutamiento mensajes SBI comunicacion indirectas

NF	`nf_package`	Grupo de inventario	Plantilla de configuración	Rol
Función de Gestión de Sesiones	`omnismf`	`smf`	`omnismf.j2`	Control de sesión PDU y UPF (N4)
Gestión de Datos Unificada	`omniudm`	`udm`	`omniudm.j2`	Datos de suscripción, c concealment SUCI
Repositorio de Datos Unificado	`omniudr`	`udr`	`omniudr.j2`	Almacenamie de datos de respaldo para UDM/PCF

Cómo funciona el rol `5gc_nf`

Para cada NF, el rol realiza la misma secuencia:

1. **Resolver `group_vars_omnicore`**: se establece en `<inventory>/group_vars/`, por lo que las búsquedas de configuración se resuelven en relación con el inventario activo.
2. **Descubrir servidores de licencia**: si `license_server_api_urls` no está configurado, se deriva del grupo de inventario `license_server` (`https://<host>:8443/api`).
3. **Instalar requisitos previos**: `lsb-release`, `curl`, `gpg`, `unzip`, `libglu1-mesa`, `libsctp1`.
4. **Instalar el paquete NF** desde el repositorio APT de Omnitouch. Última versión por defecto; fijar con `nf_version` (ver Fijación de versión).
5. **Plantillar la configuración de tiempo de ejecución (versionada)**: renderiza `templates/<nf_package>.j2` a `/etc/<nf_package>/runtime-<deb_version>.exs`.

6. **Crear un enlace simbólico** `runtime.exs` → la configuración de la versión activa, y reflejar ese enlace simbólico en cada directorio `/opt/<nf_package>/releases/*`.
7. **Reiniciar la NF** (manejador) cada vez que cambie la configuración activa.

Fijación de versión y retroceso

La configuración se escribe por versión de Debian (`runtime-<deb_version>.exs`) y `runtime.exs` es un enlace simbólico a la activa. Las configuraciones anteriores se preservan en disco, por lo que un retroceso es simplemente reinstalar la versión anterior del paquete:

```
- {role: 5gc_nf, nf_package: omniamf, nf_version: "1.2.2-1",
become: yes}
```

Omitir `nf_version` para rastrear el paquete más reciente.

Claves HNET de UDM (de-concealment de SUCI)

El UDM (`nf_package: omniudm`) tiene pasos adicionales para soportar la de-concealment de SUCI (TS 33.501 §6.12). La de-concealment recupera el SUPI del SUCI oculto que un UE presenta al registrarse. Necesita las claves **privadas** de la Red de Hogar que emparejan con las claves **públicas** de la Red de Hogar en los USIM.

- Las claves se encuentran en `group_vars/hnet_udm/` en el controlador Ansible como `<curve>-<id>.key` (PEM), donde `<curve>` es `curve25519` (Perfil A, X25519) o `secp256r1` (Perfil B, prime256v1), y `<id>` es el Identificador de Clave Pública de la Red de Hogar.
- Si el directorio está **vacío**, se genera automáticamente una `curve25519-1.key` por defecto para que la de-concealment funcione de inmediato. Este paso es idempotente. Una clave existente nunca se sobrescribe.
- Para cada `curve25519-*.key`, el valor público crudo de 32 bytes correspondiente se exporta (hex) a un archivo `.pub` para la provisión de

USIM. Los archivos `.pub` permanecen en el inventario; solo los archivos `*.key` se despliegan en `/etc/omniudm/hnet/` en el UDM (modo `0600`, root).

> Una clave generada automáticamente es aleatoria y solo de-conceals SUCIs de USIMs programados con su clave pública **coincidente**. Toma el `.pub` exportado y provéelo como la Clave Pública de la Red de Hogar en el USIM. Las SIM que utilizan el esquema nulo no necesitan clave HNET y pueden dejar el directorio vacío.

Detalles completos y ejemplos de diseño de archivos: Configuración de Variables de Grupo: Ejemplo 8.

Documentación Relacionada

- Playbooks de Servicio - Todos los playbooks de servicio y la jerarquía de despliegue
- Núcleo Conmutado por Circuito - El dominio CS legado (BSC, MSC, SS7)
- Configuración de Variables de Grupo - Plantillas de configuración por NF y claves HNET
- Arquitectura de Despliegue - Cómo se integran los dominios
- **Configuración de OmniCore**: https://docs.omnitouch.com.au/docs/repos/OmniCore - Qué se incluye en los archivos de configuración de NF

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 del Núcleo de Paquetes** - Primera /24
2. **Red de Señalización** - Segunda /24
3. **Red Interna de IMS** - Tercera /24
4. **Red Pública de IMS** - 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 adapta 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 **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 ajustarlo a tu topología de red específica, espacio de direcciones disponible y requisitos operativos.

Desglose de Segmentos de Red

1. Red del 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 la Puerta de Enlace de PDN)
- OmniUPF/PGW-U (Función del Plano de Usuario / Puerta de Enlace de PDN)

Ejemplo: `10.179.1.0/24`

```
mme:
  hosts:
    customer-mme01:
      ansible_host: 10.179.1.15
      gateway: 10.179.1.1
      host_vm_network: "v400-omni-packet-core"
```

---

2. Red de Señalización (Segunda /24)

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

Componentes:

- OmniHSS (Servidor de Suscriptores en el Hogar)
- OmniCharge OCS (Sistema de Cobro en Línea)
- OmniHSS PCRF (Función de Reglas de Políticas y Cobro)
- OmniDRA DRA (Agente de Enrutamiento Diameter)
- Servidores DNS (DNS interno/infraestructura - distinto del DNS orientado al cliente en el segmento Público de IMS; ambos son intencionales)
- Servidores TAP3/CDR
- Monitoreo/OAM
- Captura de SIP
- Servidor de Licencias
- Monitor de RAN
- Omnitouch Warning Link CBC (Centro de Difusión de Celdas) - si se implementa

Ejemplo: `10.179.2.0/24`

```yaml
hss:
  hosts:
    omni-site-hss01:
      ansible_host: 10.179.2.140
      gateway: 10.179.2.1
      host_vm_network: "v401-omni-signalling"
```

---

3. Red Interna de IMS (Tercera /24)

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

Componentes:

- OmniCSCF S-CSCF (Función de Control de Sesión de Llamada de Servicio)
- OmniCSCF I-CSCF (Función de Control de Sesión de Llamada de Interrogación)
- OmniTAS (Servidor de Aplicaciones de Telefonía / Servidor de Aplicaciones)
- OmniMessage (Controlador de SMS, SMPP, IMS)

Ejemplo: `10.179.3.0/24`

```yaml
scscf:
  hosts:
    omni-site-scscf01:
      ansible_host: 10.179.3.45
      gateway: 10.179.3.1
      host_vm_network: "v402-omni-ims-internal"
```

---

4. Red Pública de IMS (Cuarta /24)

Propósito: Servicios IMS orientados al usuario, DNS orientado al cliente y interconexión SS7/legado

Componentes:

- OmniCSCF P-CSCF (Función de Control de Sesión de Llamada Proxy)
- Servidores XCAP
- Servidores de Correo de Voz Visual
- DNS del Cliente
- 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
- MSC (Centro de Conmutación Móvil)

> **Nota:** Las funciones de interconexión SS7/HLR y relacionadas comúnmente se encuentran en el segmento público, ya que se conectan con redes externas 2G/3G a través de SIGTRAN/M3UA en lugar de IMS SIP interno.

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: "v403-omni-ims-public"
```

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 mayor aislamiento y es el enfoque recomendado para

implementaciones en producción. La convención es nombrar cada puente según su segmento (`v400-omni-packet-core`, `v401-omni-signalling`, `v402-omni-ims-internal`, `v403-omni-ims-public`), aunque un solo puente compartido (por ejemplo, `vmbr0`) también es válido donde la segmentación se maneje en otro lugar.

Ejemplo:

```
# Núcleo de Paquetes - v400-omni-packet-core
mme:
  hosts:
    omni-site-mme01:
      ansible_host: 10.179.1.15
      gateway: 10.179.1.1
      host_vm_network: "v400-omni-packet-core"

# Señalización - v401-omni-signalling
hss:
  hosts:
    omni-site-hss01:
      ansible_host: 10.179.2.140
      gateway: 10.179.2.1
      host_vm_network: "v401-omni-signalling"

# IMS Interno - v402-omni-ims-internal
icscf:
  hosts:
    omni-site-icscf01:
      ansible_host: 10.179.3.55
      gateway: 10.179.3.1
      host_vm_network: "v402-omni-ims-internal"

# IMS Público - v403-omni-ims-public
pcscf:
  hosts:
    omni-site-pcscf01:
      ansible_host: 10.179.4.165
      gateway: 10.179.4.1
      host_vm_network: "v403-omni-ims-public"
```

Método 2: Segmentación Basada en VLAN

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

Ejemplo:

```
# Todos los componentes comparten un puente (ovsbr1); cada
segmento utiliza una etiqueta VLAN diferente.
# applicationserver (TAS) es un componente interno de IMS -> VLAN
402 / 10.178.3.x
applicationserver:
  hosts:
    omni-site-sbc01:
      ansible_host: 10.178.3.213
      gateway: 10.178.3.1
      host_vm_network: "ovsbr1"
      vlanid: "402"

# dra y dns son componentes de Señalización -> VLAN 401 /
10.178.2.x
dra:
  hosts:
    omni-site-dra01:
      ansible_host: 10.178.2.211
      gateway: 10.178.2.1
      host_vm_network: "ovsbr1"
      vlanid: "401"

dns:
  hosts:
    omni-site-dns01:
      ansible_host: 10.178.2.178
      gateway: 10.178.2.1
      host_vm_network: "ovsbr1"
      vlanid: "401"
```

Configuración de Red:

- Configura las VLANs en el switch físico

- Etiqueta el tráfico apropiadamente a nivel del hipervisor
- Enruta entre VLANs en el gateway/firewall

Ejemplo de Mapeo de VLAN:

```
VLAN 400: 10.x.1.0/24 (Núcleo de Paquetes)
VLAN 401: 10.x.2.0/24 (Señalización)
VLAN 402: 10.x.3.0/24 (IMS Interno)
VLAN 403: 10.x.4.0/24 (IMS P��blico)
```

Trabajando con Direcciones IP Públicas

Descripción General

Muchas implementaciones de OmniCore requieren que algunos componentes tengan direcciones IP públicas para conectividad externa, tales 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
    customer-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
    customer-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
    customer-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:
    customer-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:
    customer-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 utilizando GTP
- SGWs públicas específicamente para tráfico de roaming de operadores externos
- La misma 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.

Introducción al Despliegue de Ansible en Omnitouch

Visión General

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

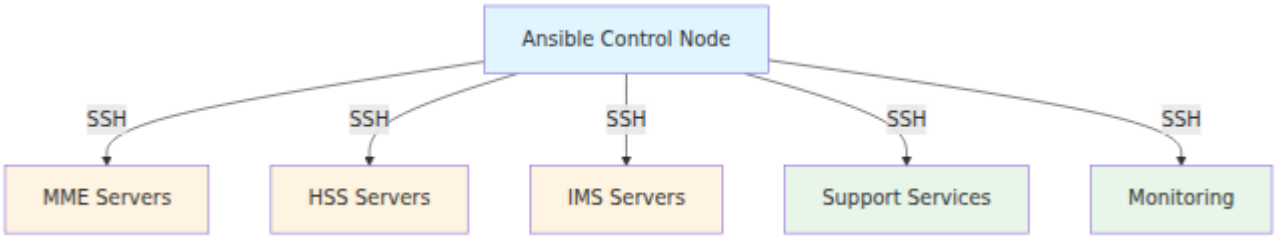
¿Qué es Ansible?

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.

El equipo de ONS define estos roles. Puedes editarlos. Sin embargo, generalmente hay formas más limpias de hacer cualquier ajuste que necesites desde tu archivo de hosts.

3. Playbooks

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

```
- name: Setup Omnicore MME
  hosts: mme
  roles:
    - omnimme
```

Usamos estos esencialmente como grupos para los roles.

En la práctica, cada playbook de servicio aplica un solo rol, y la configuración compartida se incluye por separado. Un paquete de servicio como `services/epc.yml` utiliza `import_playbook` para ejecutar primero `common.yml` y luego cada playbook por servicio:

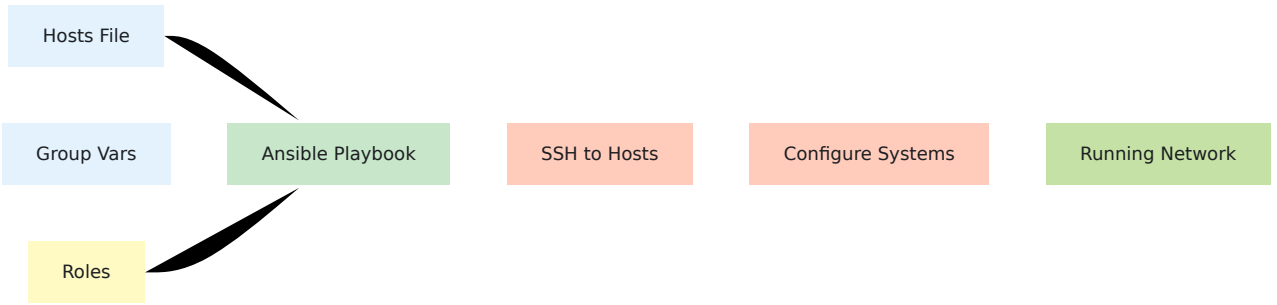
```
- name: Install Common
  import_playbook: common.yml

- name: Run MME Playbook
  import_playbook: omnimme.yml
```

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`:

```yaml
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 el código de la aplicación
- Configurar servicios con ajustes del cliente
- Iniciar servicios
- Validar el despliegue

Componentes Clave que Desplegamos

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

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

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

OmniCall (Plataforma de Voz y Mensajería)

- **OmniCall CSCF** - Función de Control de Sesiones de Llamadas (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: Visión General de la Arquitectura de Despliegue

Gestión de Paquetes

Utilizamos un modelo de distribución de paquetes híbrido:

Paquetes APT Precompilados

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

- Construidos a partir del código fuente en nuestra canalización 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

También hay un `.deb` unificado de `omnicore` (construido a partir de `packaging/`) que agrupa los playbooks y herramientas para la auto-provisión de un despliegue.

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 instalar las dependencias requeridas. Este proyecto utiliza uv para gestionar el entorno de Python, así que **uv es requerido**.

1. Instalar uv

Si uv no está ya instalado, instálalo utilizando el método que mejor se adapte a tu nodo de control. Consulta la documentación de instalación de uv. Por ejemplo:

```
pipx install uv                                    # a través de
pipx
curl -LsSf https://astral.sh/uv/install.sh | sh    # instalador
independiente
```

2. Instalar Dependencias

Desde la raíz del repositorio, ejecuta:

```
uv sync
```

Esto crea un `.venv/` aislado e instala Ansible más todos los paquetes de Python necesarios (fijados en `uv.lock`) para la automatización del despliegue

de Omnitouch.

3. Activar el Entorno

Activa el entorno virtual y mantenlo activado para todos los comandos de Ansible:

```
source .venv/bin/activate      # Windows: .venv\Scripts\activate
```

4. Instalar Colecciones de Ansible

Instala las colecciones de Ansible de las que dependen los playbooks (desde `requirements.yml`):

```
ansible-galaxy collection install -r requirements.yml
```

5. Ejecutar Comandos de Ansible

Con el entorno activado, ejecuta comandos de Ansible directamente:

```
ansible --version
```

Nota: Desactiva el entorno 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
- Visión General de la Arquitectura de Despliegue - Ve el panorama completo
- Configuración de Variables de Grupo - Personaliza tu despliegue
- Playbooks de Utilidad - Herramientas operativas para verificaciones de salud, copias de seguridad y mantenimiento

Repositorio APT y Distribución de Paquetes

Visión General

El sistema APT de Omnitouch proporciona distribución de paquetes para todas las implementaciones. Se sirven dos tipos de contenido:

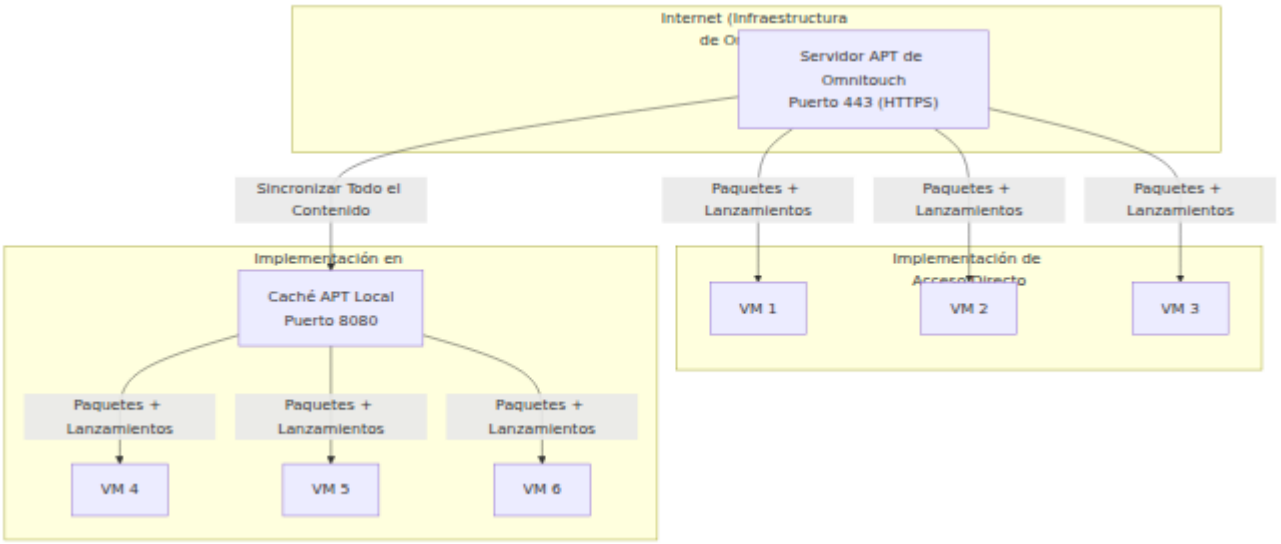
1. **Paquetes APT**: paquetes de Debian instalados a través de `apt install`
2. **Lanzamientos Binarios**: binarios preconstruidos descargados directamente (exportadores de Prometheus, agentes, etc.)

La página de paquetes en el servidor APT también incluye una pestaña de **Auto-configuración (OmniCore)**, que proporciona el paquete `omnicore` (servido en `/pool/main/o/omnicore/`) que contiene el conjunto completo de herramientas de automatización de Ansible para clientes que desean implementar y gestionar su red por sí mismos.

Se admiten dos modelos de implementación:

1. **Acceso Directo**: las máquinas virtuales obtienen paquetes directamente del servidor APT de Omnitouch (por ejemplo, `apt.<region>.omnitou.ch`)
2. **Espejo de Caché Local**: un servidor local se sincroniza desde Omnitouch y sirve paquetes a las máquinas virtuales (para implementaciones fuera de línea/aisladas)

Arquitectura



Contenido Servido

El servidor APT alberga todo el contenido requerido para las implementaciones:

Tipo de Contenido	Descripción	Ruta
Paquetes de Omnitouch	Paquetes `.deb` construidos a medida (omnihss, omnimme, etc.)	`/dists/<distro>/`
Kit de Herramientas OmniCore	El paquete `omnicore`: automatización completa de Ansible (roles, playbooks, deps) para implementación de autoservicio	`/pool/main/o/omnicore/`
Paquetes de Ubuntu	Paquetes de Ubuntu en caché con todas las dependencias	`/<distro>/pool/main/`
Lanzamientos de GitHub	Binarios preconstruidos (Prometheus, Grafana, VoIPmonitor, etc.)	`/releases/<org>/<repo>/`
Tarballs de Fuente	Archivos de origen para aplicaciones web (CGrateS_UI, speedtest)	`/repos/`
Paquetes de Terceros	Galera, FRR, InfluxDB, KeyDB, etc.	`/releases/<vendor>/`

Variables de Configuración

`apt_repo` es la única fuente de verdad para la distribución de paquetes. Defínelo una vez en tu inventario. Todo lo demás se deriva de él. No configuras la URL de descarga binaria, esquema, puerto o credenciales por separado.


apt_repo
(única fuente de verdad, definida en el inventario)
Derivado
remote_apt_*
(roles/common/defaults/main.yml)
apt_download_base
(roles/common/tasks/facts.yml)
Lo que Configuran
/etc/apt/sources.list.d/omnitouch.list
Descargas binarias
/releases/, /repos/

Cómo funciona la derivación

Variable	Donde se establece	Rol
`apt_repo`	Tu inventario	El único bloque que configuras. Se plantilla directamente en la línea de origen APT.
`remote_apt_*`	`roles/common/defaults/main.yml`	Derivado de `apt_repo`. Último recurso cuando `apt_repo` está ausente: `apt.us-fl.omnitou.ch` / `443` / `https`.
`apt_download_base`	`roles/common/tasks/facts.yml`	URL base única calculada que cada rol utiliza para descargas binarias, por lo que la línea de origen APT y las descargas binarias siempre concuerdan en host, esquema, puerto y credenciales.

Debido a que ambos valores derivados se construyen a partir del mismo bloque `apt_repo`, un inventario normal **solo necesita** **`apt_repo`**. No configuras `remote_apt_*` tú mismo.

> **Excepción: el servidor espejo de caché.** Un host en el grupo `apt_cache_servers` es el único lugar donde `remote_apt_*` todavía se establece explícitamente. Su propio `apt_repo` apunta a la *caché local*. Los valores `remote_apt_*` describen entonces el servidor Omnitouch *upstream* del que se sincroniza (ver Opción 2).

Esquema y puerto

El esquema sigue automáticamente el puerto: **puerto** **`443`** **→** **`https`**, **cualquier otro puerto →** **`http`**. Cuando se omite `apt_repo.apt_repo_port`:

- **Acceso directo** por defecto a **`443`** **(HTTPS)**
- **Modo de caché** por defecto a **`8080`** **(HTTP)**

Cuándo se usa cada modo

Escenario	Línea de origen APT	Descargas binarias (`apt_download_base`)
`use_apt_cache: true` (caché)	`http://<cache>:8080/...` (sin autenticación)	`http://<cache>:8080/...` (sin autenticación)
`use_apt_cache: false` (directo)	`https://<user>:<pass>@<server>/...`	`https://<user>:<pass>@<server>:443/...`

Opción 1: Acceso Directo

Para implementaciones 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 (por ejemplo, `apt.<region>.omnitou.ch`)

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	443	TCP	Descargas de paquetes (HTTPS)
Servidor APT	53	TCP/UDP	Resolución DNS para el FQDN de APT
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 HTTPS (TCP/443), NTP (UDP/123) y DNS (TCP+UDP/53) esté permitido a los rangos de IP de Omnitouch.

Configuración

```
all:
  vars:
    use_apt_cache: false

    # Única fuente de verdad. La configuración de descarga binaria (remote_apt_*) y
    # la URL apt_download_base se derivan de esto automáticamente - NO
    # las configuras por separado.
    apt_repo:
      apt_server: "apt.<region>.omnitou.ch"   # FQDN proporcionado por Omnitouch
      apt_repo_username: "tu-usuario"
      apt_repo_password: "tu-contraseña"
      # apt_repo_port: 443                    # opcional; por defecto a 443 (HTTPS)
```

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	-
`apt_repo.apt_repo_port`	Entero	No	`443`

Descargas Binarias (`remote_apt_*`)

Estos se derivan automáticamente de `apt_repo`. No los configures en un inventario de acceso directo. Existen solo para que el servidor espejo de caché pueda apuntar a un upstream diferente (ver Opción 2). Para referencia, se predeterminan de la siguiente manera:

Parámetro	Derivado de	Recaída
`remote_apt_server`	`apt_repo.apt_server`	`apt.us-fl.omnitou.ch`
`remote_apt_port`	`apt_repo.apt_repo_port`	`443`
`remote_apt_protocol`	puerto (`443` → `https`)	`https`
`remote_apt_user`	`apt_repo.apt_repo_username`	`""`
`remote_apt_password`	`apt_repo.apt_repo_password`	`""`

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.d/omnitouch.list`):

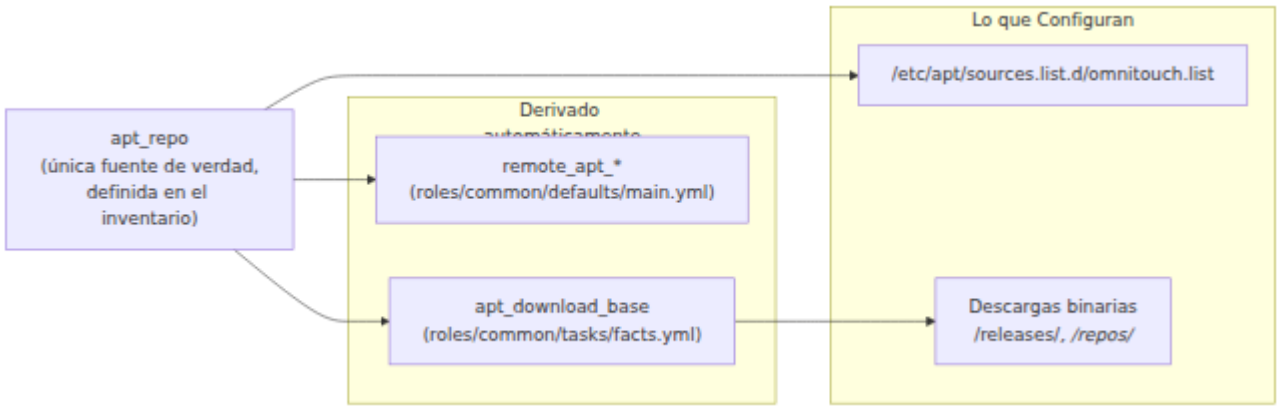
```
deb [trusted=yes] https://{username}:{password}@{apt_server}/
noble main
```

Descargas Binarias (construidas a partir de `apt_download_base` por tareas de Ansible `get_url`):

```
https://{username}:
{password}@{apt_server}/releases/prometheus/node_exporter/node_export
1.8.1.linux-amd64.tar.gz
```

> El puerto `443` se omite de la URL (es el predeterminado de HTTPS). Un puerto no estándar se agrega como `:<port>` y cambia el esquema a HTTP a menos que sea `443`.

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 Omnitouch (pre-caché), por lo que las máquinas virtuales no necesitan acceso a los espejos de Ubuntu.

Firma de Paquetes (GPG)

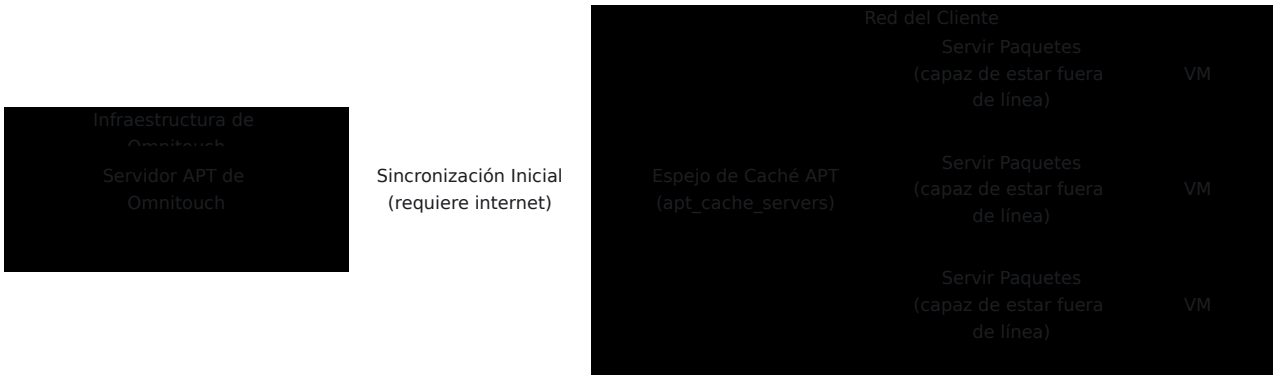
El servidor APT de Omnitouch genera una clave de firma de repositorio y sirve la clave pública encriptada en `/omnitouch.gpg`. Para los hosts de acceso directo, el rol `common` obtiene automáticamente esa clave e instala en `/etc/apt/trusted.gpg.d/omnitouch.gpg` para que `apt-get update` verifique los metadatos del repositorio sin advertencias de `NO_PUBKEY`. Esto sucede automáticamente; no se requiere gestión manual de claves.

> El modo de espejo de caché no utiliza verificación GPG. Los hosts que obtienen de una caché local utilizan `[trusted=yes]` en su lugar (ver Opción 2).

Opción 2: Espejo de Caché Local

Para implementaciones fuera de línea, aisladas o con ancho de banda limitado, implementa una caché APT local que sincroniza todo el contenido desde Omnitouch.

Arquitectura



Configuración

Define el servidor de caché en tu archivo de hosts. Es el único lugar donde `remote_apt_*` se establece explícitamente. Esas variables describen el servidor Omnitouch **upstream** del que la caché se sincroniza:

```yaml
apt_cache_servers:
  hosts:
    example-apt-cache:
      ansible_host: 192.168.1.100
      gateway: 192.168.1.1
  vars:
    # El servidor de caché sincroniza contenido DESDE el
repositorio Omnitouch upstream
    remote_apt_server: "apt.<region>.omnitou.ch"
    remote_apt_port: 443
    remote_apt_protocol: "https"
    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: auto-derivado a 192.168.1.100 (primer
servidor de caché)
```

Cómo funciona:

- **Servidor de caché** (`192.168.1.100`): Utiliza credenciales `remote_apt_*` para sincronizar contenido desde el servidor Omnitouch upstream a través de HTTPS.
- **Todos los demás hosts**: Derivan automáticamente `apt_repo.apt_server: "192.168.1.100"` y obtienen de 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	Auto-derivado
`apt_repo.apt_repo_username`	Cadena	No	-
`apt_repo.apt_repo_password`	Cadena	No	-
`apt_repo.apt_repo_port`	Entero	No	`8080`

Sincronización del Servidor de Caché (`remote_apt_*`)

Estas variables configuran cómo el servidor de caché sincroniza contenido desde Omnitouch. Este es el único escenario donde `remote_apt_*` se establece manualmente:

Parámetro	Tipo	Requerido	Predeterminado	Descr
`remote_apt_server`	Cadena	Sí	-	Servid Omnit upstre desde sincron
`remote_apt_port`	Entero	No	`443`	Puerto servid Omnit upstre
`remote_apt_protocol`	Cadena	No	`https`	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	Descrip
`use_apt_cache`	Booleano	No	`true`	Se establec automática en `true` cu existe el gr `apt_cache_`

> **Obsoleto:** `apt_cache_port` ya no se utiliza. El puerto al que se conectan los hosts ahora se establece a través de `apt_repo.apt_repo_port` (predeterminado a `8080` cuando se omite).

Patrones de URL (Modo Caché)

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

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

Descargas Binarias (construidas a partir de `apt_download_base`):

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

El acceso a la caché no necesita credenciales. 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 playbook de configuración de caché:**

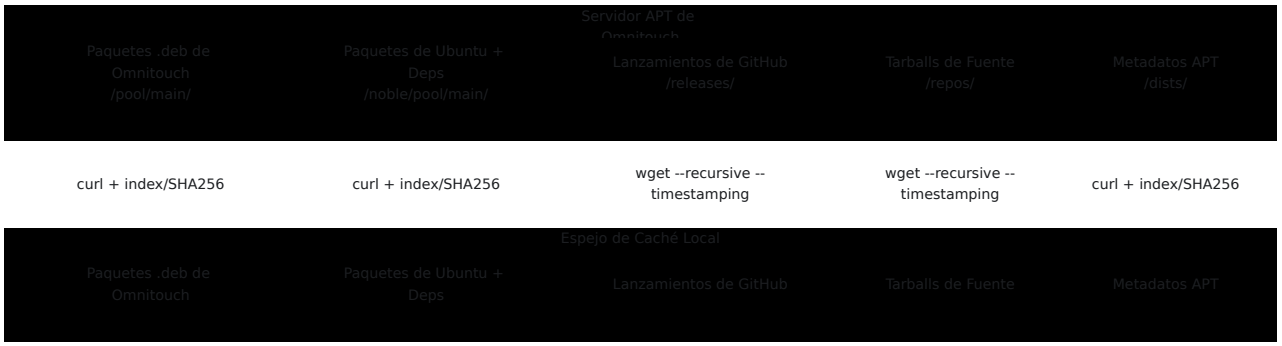
```
ansible-playbook -i hosts/customer/production.yml
services/apt_cache.yml
```

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. El mecanismo difiere según el tipo de contenido:

- **Paquetes APT** (`/dists/` + `/pool/` tanto para Omnitouch como para Ubuntu) se sincronizan con `curl` impulsado por el índice del repositorio de Debian. El espejo obtiene el archivo `Release` upstream (y se salta si no ha cambiado), analiza `Packages.gz` para el nombre de archivo/SHA256/tamaño autoritativo de cada paquete, y descarga solo archivos que faltan o cuyo SHA256 no coincide con la copia local. Cada archivo descargado se verifica con SHA256 antes de ser comprometido.
- **Lanzamientos binarios y tarballs de fuente** (`/releases/`, `/repos/`) no tienen índice al estilo de Debian, por lo que recurren a `wget --recursive --timestamping`, que solo vuelve a descargar archivos con un `Last-Modified` upstream más reciente.

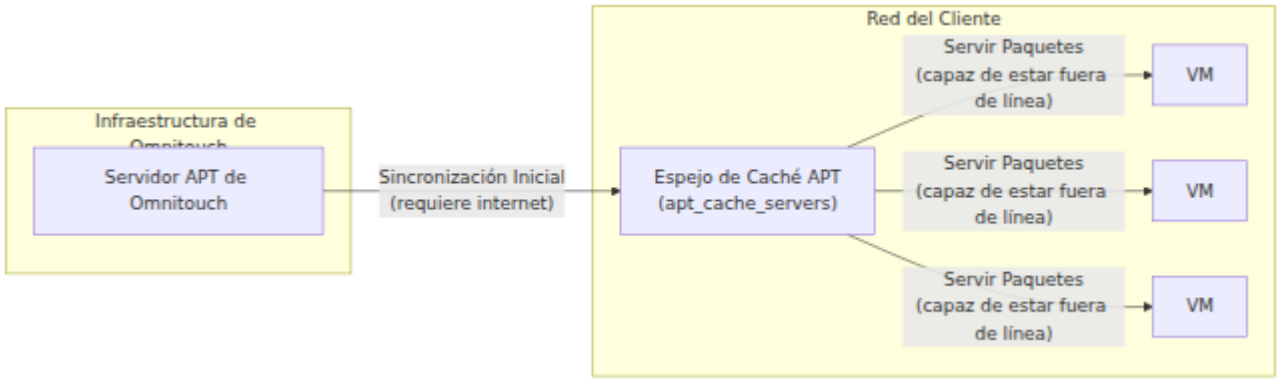


Directorios de contenido sincronizados:

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 de 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 se autentica con autenticación básica HTTP para todo el contenido. Los paquetes APT se sincronizan con `curl` contra el índice del repositorio de Debian (`Release` + `Packages.gz`), descargando solo archivos cuyo SHA256 difiere de la copia local y verificando cada uno después de la descarga; `/releases/` y `/repos/` recurren a `wget --recursive --timestamping`. De cualquier manera, las sincronizaciones posteriores solo obtienen archivos nuevos o cambiados.

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

```
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 Omnitouch upstream
    remote_apt_server: "apt.<region>.omnitou.ch"
    remote_apt_user: "tu-usuario"
    remote_apt_password: "tu-contraseña"
```

Qué 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 de `http://192.168.1.100:8080/` sin credenciales
- El servidor de caché sincroniza paquetes desde `https://tu-usuario:tu-contraseña@apt.<region>.omnitou.ch/`

Anular el 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.<region>.omnitou.ch"
      apt_repo_username: "usuario"
      apt_repo_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. Solo se necesita `apt_repo`. Los valores `remote_apt_*` se derivan de él.

```yaml
all:
  vars:
    use_apt_cache: false

    # Única fuente de verdad para tanto las fuentes APT como las descargas binarias
    apt_repo:
      apt_server: "apt.<region>.omnitou.ch"
      apt_repo_username: "usuario"
      apt_repo_password: "contraseña"
```

Resultado: Todos los hosts generan `deb [trusted=yes] https://usuario:contraseña@apt.<region>.omnitou.ch/ 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.

```
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 upstream
    remote_apt_server: "apt.<region>.omnitou.ch"
    remote_apt_port: 443
    remote_apt_protocol: "https"
    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 `https://user:pass@apt.<region>.omnitou.ch/`
- **Todos los demás hosts**: Genera `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

    # Apunta 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           # El caché normalmente se
ejecuta en el puerto 8080

# No se necesita grupo apt_cache_servers
# No se necesita remote_apt_* (el caché ya está configurado
externamente)
```

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

Ejemplo Completo

Este es un ejemplo completo y funcional. Muestra la configuración del servidor de caché con múltiples hosts de aplicación:

```yaml
# Grupo de Servidores de Caché APT
apt_cache_servers:
  hosts:
    example-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 upstream
    remote_apt_server: "apt.<region>.omnitou.ch"
    remote_apt_port: 443
    remote_apt_protocol: "https"
    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 existen
```

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

Qué sucede durante el despliegue:

1. **Servidor de caché (`10.179.1.114`):**
   - Utiliza `remote_apt_*` de su sección `vars:`
   - Descarga todos los paquetes desde `https://customer-username:customer-secure-token@apt.<region>.omnitou.ch/`
   - Sirve paquetes en el puerto 8080 a través de nginx
2. **Hosts de aplicación (`customer-hss01`, `customer-mme01`, `customer-dns01`):**
   - Detecta automáticamente que existe el grupo `apt_cache_servers`
   - Establece automáticamente `use_apt_cache: true`
   - Deriva automáticamente `apt_repo.apt_server: "10.179.1.114"`
   - Genera: `deb [trusted=yes] http://10.179.1.114:8080/noble noble main`
   - Obtiene todos los paquetes del 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 y dependencias de Ubuntu
- Nuevos lanzamientos de GitHub
- Tarballs de origen actualizados

Las re-sincronizaciones son incrementales: los paquetes APT se comparan con el índice Debian upstream por SHA256 (y se omiten por completo si el archivo `Release` upstream no ha cambiado), mientras que `/releases/` y `/repos/` utilizan `wget --timestamping`. Los archivos existentes que no han cambiado se omiten, haciendo que la re-sincronización sea rápida.

Nota: El servidor APT de Omnitouch es la única fuente de verdad para todos los paquetes. Omnitouch construye y publica paquetes en él; ejecutar `services/apt_cache.yml` en los espejos de caché luego sincroniza lo que actualmente está publicado.

Solución de Problemas

La Actualización de APT Falla con 401 No Autorizado

Síntomas:

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

Posibles causas:

- 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 al caché
- `apt_repo_username` o `apt_repo_password` incorrectos
- IP de origen no permitida 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 upstream
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 incorrecto)**: `deb [trusted=yes] https://user:pass@10.179.1.115/noble noble main`
4. Verifica que las credenciales sean correctas para tu modo de despliegue
5. Confirma que tu IP pública esté permitida con Omnitouch (si usas acceso directo)

Fallos en Descargas Binarias (Node Exporter, Zabbix, etc.)

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

Posibles causas:

- Credenciales de `apt_repo` incorrectas (las descargas binarias derivan su autenticación de `apt_repo` en modo directo)
- En un servidor espejo de caché, `remote_apt_user` / `remote_apt_password` incorrectos
- Esquema o puerto incorrectos. Recuerda que el puerto `443` implica HTTPS y cualquier otro puerto implica HTTP.

Resolución:

1. En modo directo, verifica que `apt_repo.apt_repo_username` / `apt_repo.apt_repo_password` coincidan con los proporcionados por Omnitouch
2. En un servidor espejo de caché, verifica las credenciales `remote_apt_*` upstream
3. Confirma que el `apt_download_base` derivado utilice el host/esquema/puerto esperado (verifica con `ansible-playbook ... -vvv` o inspeccionando la tarea `get_url` que falla)

El Servidor de Caché No Puede Sincronizar

Síntomas: El playbook del servidor de caché falla al descargar paquetes

Posibles causas:

- 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 el servidor APT upstream de Omnitouch en el puerto 443
2. Verifica las credenciales `remote_apt_*`
3. Revisa las reglas del firewall para el acceso saliente

Documentación Relacionada

- Configuración del Archivo de Hosts: Inventario y configuración de 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

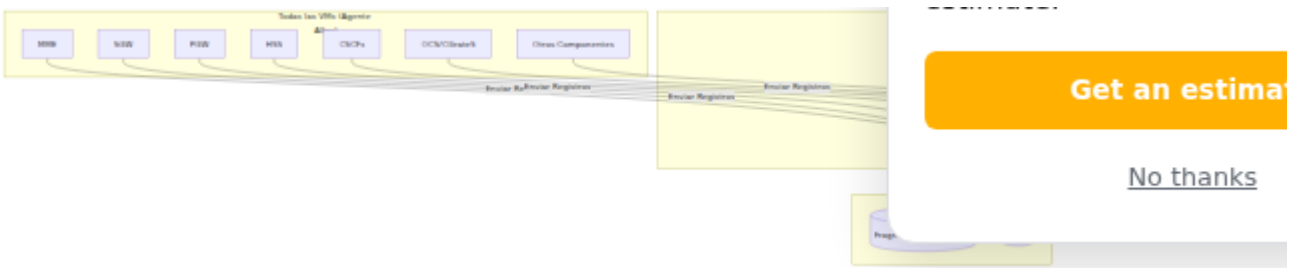
Registro Centralizado

Descripción General

OmniCore incluye un sistema de registro centralizado que recopila registros de todos los componentes y los hace buscables a través de Grafana. El sistema utiliza:

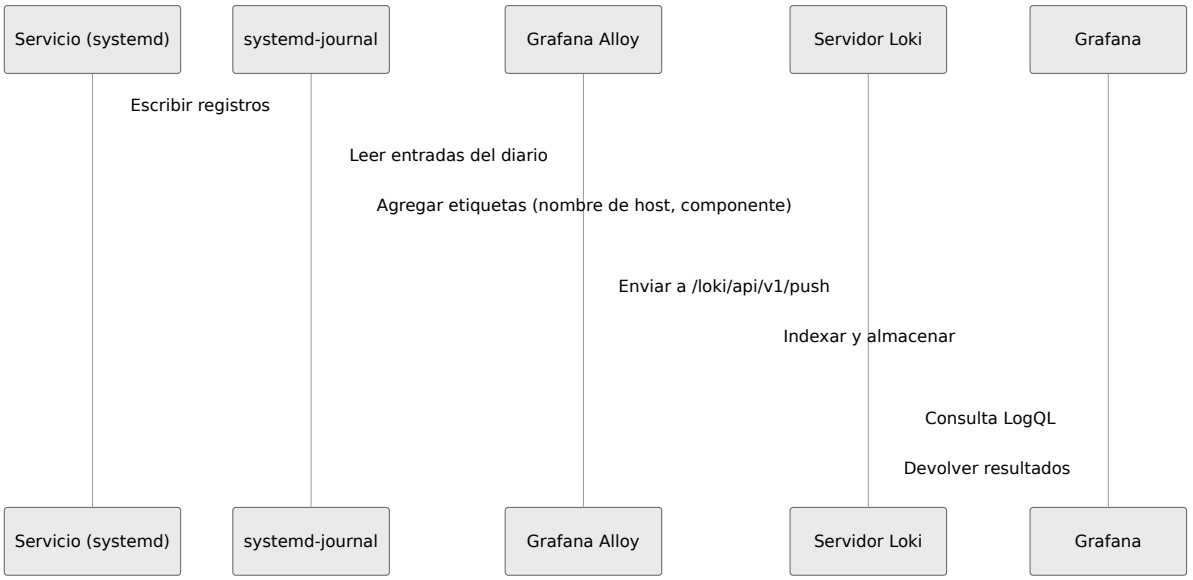
- **Grafana Alloy**: Agente de recopilación de registros desplegado en todas las VMs
- **Grafana Loki**: Servidor de agregación y almacenamiento de registros en los hosts de monitoreo
- **Dashboards de Grafana**: Dashboards preconstruidos para cada tipo de componente

Arquitectura



Cómo Funciona

Flujo de Recopilación de Registros



Etiquetas de Registro

Cada entrada de registro incluye estas etiquetas para filtrado y consulta:

Etiqueta	Descripción	Ejemplo
`job`	Nombre de host de la VM fuente	`customer-mme01`
`hostname`	Igual que job (por compatibilidad)	`customer-mme01`
`component`	Tipo de componente del grupo de inventario	`mme`, `pcscf`, `ocs`
`unit`	Nombre de unidad de systemd	`omnimme.service`
`level`	Severidad del registro	`info`, `warning`, `error`
`service`	Identificador de Syslog	`omnimme`

Configuración

Despliegue Automático

El registro se configura automáticamente cuando:

1. Un grupo `monitoring` existe en su inventario
2. El rol común se ejecuta en los hosts de destino

No se requiere configuración adicional para la funcionalidad básica.

Requisitos de Inventario

```yaml
monitoring:
  hosts:
    customer-monitoring01:
      ansible_host: 10.10.2.200
      gateway: 10.10.2.1
```

Cuando se define el grupo `monitoring`:

- **Hosts de Monitoreo**: Ejecutan el servidor Loki, que recibe y almacena los registros. También ejecutan Alloy, que recopila los propios registros del host y termina el flujo de syslog del dispositivo de red.
- **Todos los demás hosts**: Ejecutan el agente Alloy, que recopila los registros y los envía a Loki.

Configuración de Retención

Loki está configurado con retención basada en el tiempo. El compactador elimina registros más antiguos que el período de retención:

Configuración	Predeterminado	Descripción
Retención basada en el tiempo	72 horas (3 días)	Los registros más antiguos que el período de retención se eliminan

Para personalizar la retención, sobrescriba esta variable en su inventario:

```
all:
  vars:
    loki_retention_period: "168h"  # por ejemplo, 7 días en horas
```

Configuración del Búfer de Alloy

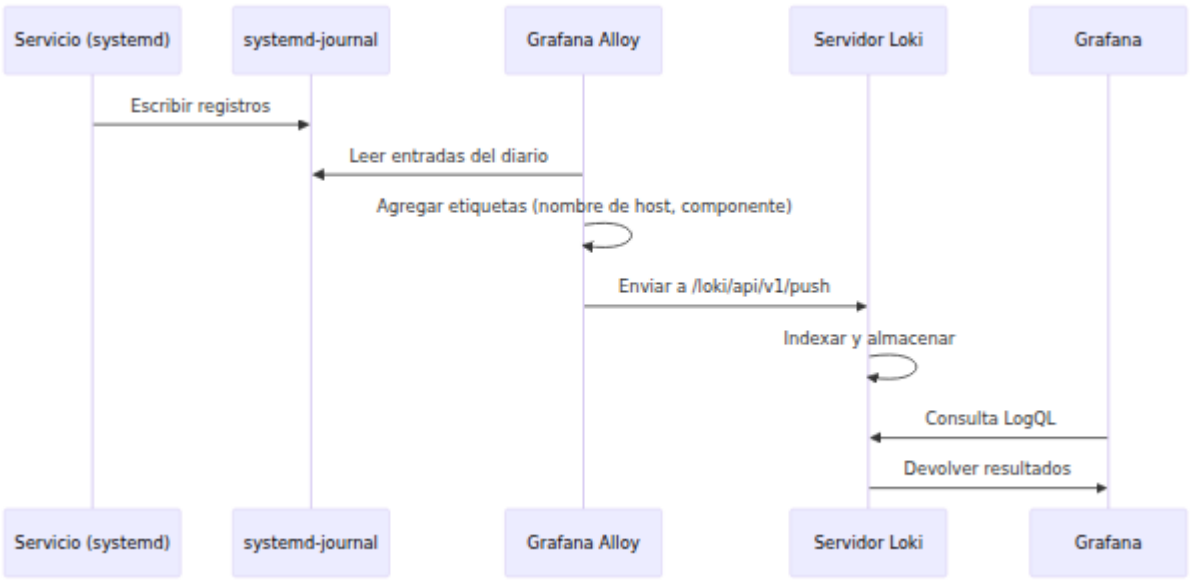
Alloy almacena registros localmente en un registro de escritura anticipada (WAL) cuando Loki no está disponible. Los segmentos en búfer se eliminan para evitar el crecimiento indefinido del disco:

Configuración	Predeterminado	Descripción
Edad máxima del segmento WAL	1 hora	Los segmentos de registro en búfer más antiguos que 1 hora se eliminan

Cuando Loki no está disponible durante más tiempo que la edad del segmento, se eliminan los registros en búfer más antiguos.

Syslog del Dispositivo de Red

Los enrutadores, conmutadores y otros dispositivos no pueden ejecutar Alloy. Estos dispositivos envían syslog BSD al host de monitoreo en su lugar. Alloy recibe el flujo y lo escribe en el mismo Loki, bajo `job="network_syslog"`. Los registros del dispositivo comparten la búsqueda, el rango de tiempo y la retención de los registros del host.



El escuchador es parte de la construcción estándar de monitoreo. Un sitio no establece una bandera para recibir syslog de dispositivos. Alloy nombra cada dispositivo a partir de la dirección fuente del datagrama, que el inventario del sitio mantiene en los mapas de `devices:` del proveedor.

Para los parámetros del colector, la configuración del dispositivo para RouterOS y otros remitentes, la referencia de etiquetas y los pasos de solución de problemas, consulte Syslog del Dispositivo de Red.

Dashboards de Grafana

Los dashboards preconstruidos se provisionan automáticamente para cada tipo de componente.

Dashboards Disponibles

Dashboard	Ubicación	Descripción
Registros de CSCF	Registros → Registros de CSCF	Registros de P-CSCF, S-CSCF, I-CSCF (OmniCSCF)
Registros de MME	Registros → Registros de MME	Eventos y errores de conexión/desconexión de MME
Registros de SGW	Registros → Registros de SGW	Eventos de sesión y portador de SGW
Registros de PGW	Registros → Registros de PGW	Eventos de PDN y sesión de PGW
Registros de HSS	Registros → Registros de HSS	Mensajes de diámetro de HSS y eventos de autenticación
Registros de OmniMessage	Registros → Registros de OmniMessage	Entrega de SMS y eventos de SMPP
Registros de OCS/CGrateS	Registros → Registros de OCS/CGrateS	Eventos de carga y tráfico JSONRPC
Llamadas RPC de CGrateS	Registros → Llamadas RPC de CGrateS	Buscar y correlacionar solicitudes/respuestas RPC con latencia
Syslog del Dispositivo de Red	Registros → Syslog del Dispositivo de Red	Syslog de MikroTik / equipo de red, por dispositivo y severidad

Características del Dashboard

Cada dashboard incluye:

- **Caja de búsqueda**: Búsqueda de texto libre en todos los registros
- **Gráficos de tasa de registros**: Volumen a lo largo del tiempo por host y severidad
- **Secciones de componentes**: Vistas agrupadas (por ejemplo, P-CSCF, S-CSCF, I-CSCF)
- **Filtrado de errores**: Vistas prefiltradas para errores y fallos
- **Transmisión en vivo**: Transmisión de registros en tiempo real (actualización cada 10 segundos)

Uso de los Dashboards

1. Navegue a Grafana (típicamente `http://<monitoring-host>:3000`)
2. Vaya a **Dashboards** → **Logs** → seleccione el componente
3. Utilice la variable **Search** para filtrar registros
4. Ajuste el rango de tiempo según sea necesario

Consultando Registros

Fundamentos de LogQL

Loki utiliza LogQL para las consultas. Sintaxis básica:

```
{label="value"} |= "cadena de búsqueda"
```

Consultas Comunes

Todos los registros de un host específico:

```
{hostname="customer-mme01"}
```

Todos los registros del componente MME:

```
{component="mme"}
```

Buscar errores en todos los CSCFs:

```
{component=~"pcscf|scscf|icscf"} |~ "(?i)error"
```

Filtrar por unidad de systemd:

```
{unit="omnicscf.service"}
```

Combinar filtros:

```
{component="hss", hostname="customer-hss01"} |= "ULR" |=
"DIAMETER_SUCCESS"
```

Consultas de Tasa de Registros

Volumen de registros por componente:

```
sum by (component) (rate({job=~".+"} [5m]))
```

Tasa de errores para CSCF:

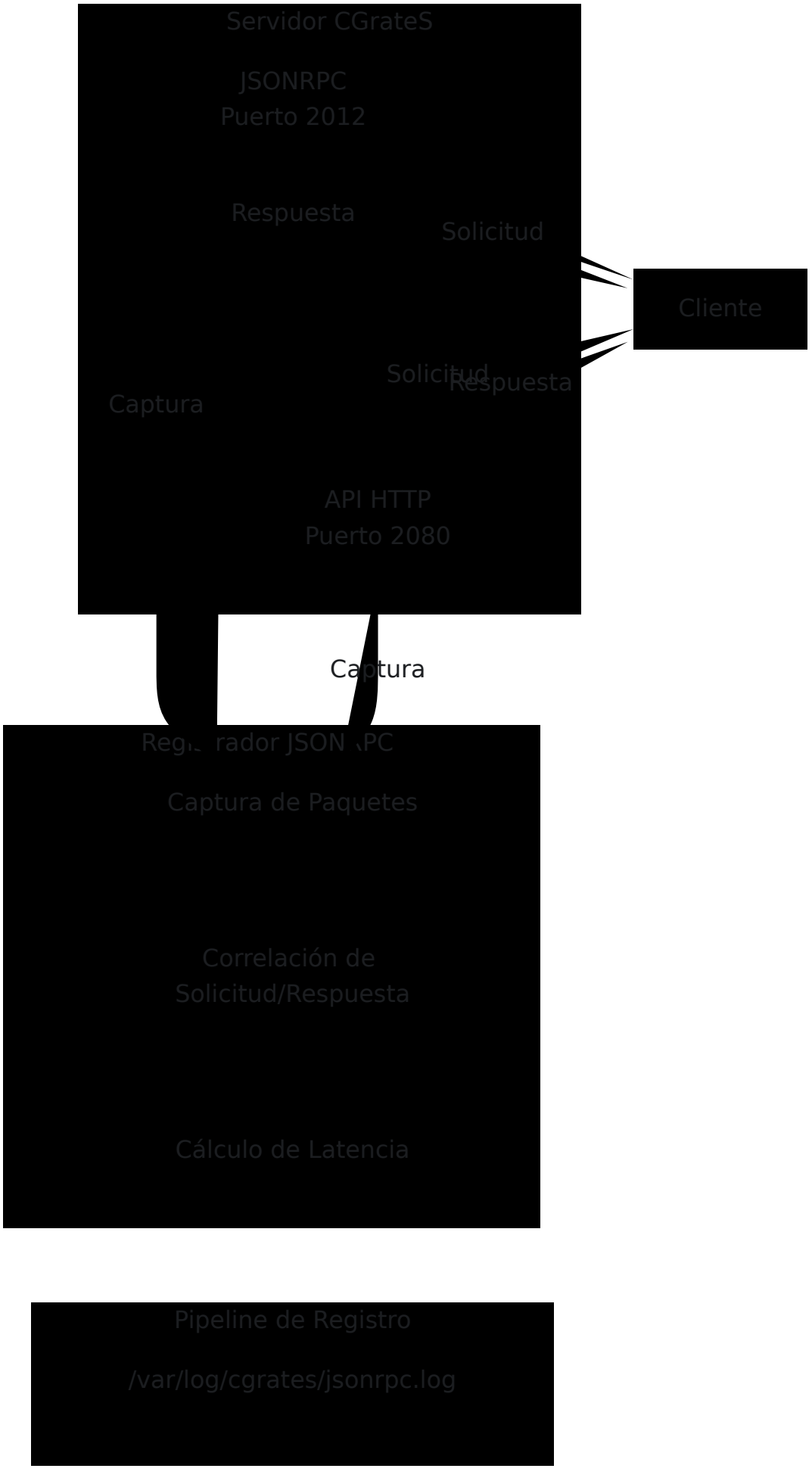
```
sum(rate({component=~"pcscf|scscf|icscf"} |~ "(?i)error" [5m]))
```

Registro JSONRPC de CGrateS

Para implementaciones de OCS/CGrateS, el tráfico JSONRPC se captura y registra con correlación de solicitud/respuesta y seguimiento de latencia para análisis de rendimiento y depuración.

El registrador JSONRPC se define en el rol `cgrates` y su archivo de registro (`/var/log/cgrates/jsonrpc.log`) es recopilado por Alloy en ambos grupos de hosts `ocs` y `cgrates`.

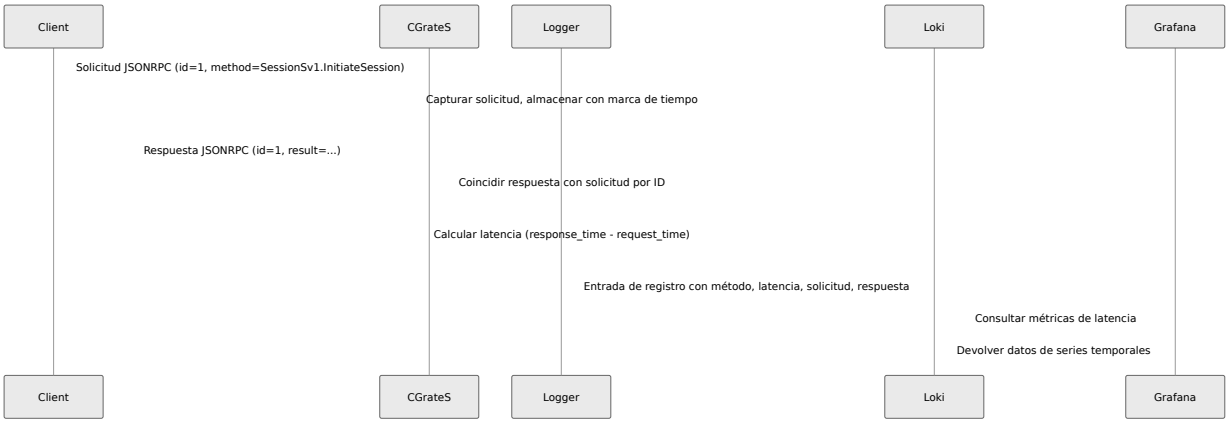
Arquitectura


Servidor CGrateS
JSONRPC
Puerto 2012
Respuesta
Solicitud
Cliente
Solicitud
Respuesta
Captura
API HTTP
Puerto 2080
Captura
Captura de Paquetes
Correlación de
Solicitud/Respuesta
Cálculo de Latencia
Pipeline de Registro
/var/log/cgrates/jsonrpc.log





Flujo de Solicitud/Respuesta



Qué se Captura

El tráfico en estos puertos de CGrateS se captura:

Puerto	Nombre del Servicio	Protocolo	Descripción
2012	`rpc_json`	TCP	JSONRPC en bruto sobre TCP
2080	`http`	HTTP	API HTTP (filtra `/metrics` y `/health`)

El registrador filtra automáticamente:

- Capturas de métricas de Prometheus (`GET /metrics`)
- Solicitudes de verificación de estado (`GET /health`)
- Respuestas comprimidas en gzip (datos binarios)

Formato de Registro

Los registros JSONRPC se escriben en `/var/log/cgrates/jsonrpc.log`. Cada entrada de registro representa un **par de solicitud/respuesta completado** con medición de latencia:

```
2026-03-01T10:30:45.123456 port=2012 service=rpc_json request_id=1
method=SessionSv1.InitiateSession latency_ms=2.45 status=ok error=
src=10.10.1.50:45678 dst=10.10.2.100:2012 request=
{"id":1,"method":"SessionSv1.InitiateSession",...} response=
{"id":1,"result":{...}}
```

Campos de Registro

Campo	Descripción	Ejemplo
`timestamp`	Marca de tiempo ISO 8601 cuando se recibió la respuesta	`2026-03-01T10:30:45.123456`
`port`	Puerto de CGrateS al que se realizó la solicitud	`2012`, `2080`
`service`	Nombre del servicio para el puerto	`rpc_json`, `http`
`request_id`	ID de solicitud JSONRPC para correlación	`1`, `42`, (vacío para nulo)
`method`	Nombre del método JSONRPC	`SessionSv1.InitiateSession`
`latency_ms`	Latencia de ida y vuelta en milisegundos	`2.45`
`status`	Estado del resultado	`ok`, `error`
`error`	Mensaje de error si el estado es error	`INSUFFICIENT_CREDIT`
`src`	IP:puerto de origen (cliente)	`10.10.1.50:45678`
`dst`	IP:puerto de destino (CGrateS)	`10.10.2.100:2012`
`request`	Carga útil completa de la solicitud JSONRPC (JSON compactado)	`{"id":1,"method":"..."}`

Campo	Descripción	Ejemplo
`response`	Carga útil completa de la respuesta JSONRPC (JSON compactado)	`{"id":1,"result":{...}}`

Dashboard de Grafana

El dashboard **Registros OCS / CGrateS** incluye paneles dedicados para el análisis de JSONRPC.

Latencia JSONRPC por Método

Un gráfico de series temporales que muestra la latencia para cada método JSONRPC a lo largo del tiempo. Útil para identificar:

- Métodos lentos que pueden necesitar optimización
- Picos de latencia que indican problemas en el sistema
- Tendencias de rendimiento a lo largo del tiempo

La leyenda muestra la latencia media y máxima por método.

Tráfico JSONRPC

Un panel de registros que muestra todos los pares de solicitud/respuesta JSONRPC con:

- Marca de tiempo
- Nombre del método
- Latencia
- Cargas útiles completas de solicitud y respuesta (formateadas)

Desglose por Método

Filtrar el tráfico JSONRPC por un método específico utilizando la variable **Method** en la parte superior del dashboard. Ingrese el nombre del método (por ejemplo, `SessionSv1.InitiateSession`) para ver solo el tráfico para ese método.

Dashboard de Llamadas RPC de CGrateS

El dashboard dedicado **Llamadas RPC de CGrateS** proporciona herramientas enfocadas para solucionar problemas de llamadas API y correlacionar solicitudes con respuestas.

Caso de Uso: Encontrar una Llamada Fallida

Cuando un usuario informa "Intenté llamar al 1234 y falló":

1. Abra **Dashboards** → **Logs** → **Llamadas RPC de CGrateS**
2. En el campo **Search**, ingrese el número de teléfono o ID de cuenta: `1234`
3. Establezca **Status** en **Error** para ver solo llamadas fallidas
4. El panel **Búsqueda de Llamadas RPC** muestra todas las llamadas coincidentes con la carga útil de solicitud/respuesta completa

Cada entrada de registro muestra la carga útil completa de la solicitud (lo que se envió) y la carga útil de respuesta (lo que devolvió CGrateS), lo que facilita identificar el error exacto.

Variables del Dashboard

Variable	Propósito	Ejemplo
Search	Búsqueda de texto libre en las cargas útiles de solicitud/respuesta	`1234`, `Account123`, `INSUFFICIENT_CREDIT`
Method	Filtrar por nombre de método RPC	`SessionSv1.InitiateSession`, `CDRsV1.ProcessEvent`
Status	Filtrar por estado de resultado	Todos, Éxito, Error

Paneles del Dashboard

Estadísticas Resumidas (fila superior):

- **Total de Llamadas RPC**: Conteo de todas las llamadas en el rango de tiempo
- **Errores**: Conteo de llamadas fallidas (codificación por colores: verde=0, amarillo=1-9, rojo=10+)
- **Latencia Promedio**: Tiempo promedio de ida y vuelta (umbral codificado por colores)
- **Latencia Máxima**: Llamada con mayor latencia

Latencia RPC por Método: Series temporales que muestran la latencia para cada método. Pase el mouse para ver valores específicos.

Búsqueda de Llamadas RPC: Panel principal de registros que muestra las llamadas coincidentes. Haga clic en cualquier entrada para expandir y ver el JSON completo de solicitud/respuesta.

Llamadas RPC Fallidas: Vista prefiltrada que muestra solo llamadas `status=error`.

Desglose por Método: Gráficos circulares que muestran la distribución de llamadas y la distribución de errores por método.

Consultando Registros JSONRPC

Consultas Básicas

Todo el tráfico JSONRPC:

```
{job="cgrates-jsonrpc"}
```

Filtrar por método:

```
{job="cgrates-jsonrpc"} |= "method=SessionSv1.InitiateSession"
```

Buscar cargas útiles de solicitud/respuesta:

```
{job="cgrates-jsonrpc"} |= "Account\":\"12345"
```

Solo errores:

```
{job="cgrates-jsonrpc"} |= "status=error"
```

Análisis de Latencia

Extraer latencia como métrica (para gráficos):

```
max_over_time(
  {job="cgrates-jsonrpc"}
  |= "method="
  | regexp "method=(?P<method>[^ ]+) latency_ms=(?P<latency>[0-
9.]+)"
  | __error__=""
  | unwrap latency [$__auto]
) by (method)
```

Encontrar solicitudes lentas (latencia > 100ms):

```
{job="cgrates-jsonrpc"}
| regexp "latency_ms=(?P<latency>[0-9.]+)"
| latency > 100
```

Consultas Específicas de Método

Procesamiento de CDR:

```
{job="cgrates-jsonrpc"} |= "method=CDRsV1"
```

Gestión de sesiones:

```
{job="cgrates-jsonrpc"} |~ "method=SessionSv1"
```

Operaciones de cuenta:

```
{job="cgrates-jsonrpc"} |~ "method=ApierV"
```

Gestión del Servicio

El registrador JSONRPC se ejecuta como un servicio systemd en los hosts de OCS/CGrateS.

Ver estado del servicio:

```
systemctl status cgrates-jsonrpc-logger
```

Ver registros del servicio:

```
journalctl -u cgrates-jsonrpc-logger -f
```

Reiniciar el servicio:

```
systemctl restart cgrates-jsonrpc-logger
```

Solución de Problemas

No Aparecen Registros JSONRPC

Síntomas: El panel de Tráfico JSONRPC muestra no datos

Causas posibles:

- Servicio de registrador no en ejecución
- ngrep no instalado
- Desajuste de interfaz de red
- Alloy no está recopilando el archivo de registro

Resolución:

1. Verifique el estado del servicio de registrador:

```
systemctl status cgrates-jsonrpc-logger
journalctl -u cgrates-jsonrpc-logger -n 50
```

2. Verifique que ngrep esté instalado:

```
which ngrep
```

3. Verifique que el archivo de registro se esté escribiendo:

```
tail -f /var/log/cgrates/jsonrpc.log
```

4. Verifique que Alloy esté recopilando el archivo:

```
journalctl -u alloy | grep jsonrpc
```

El Gráfico de Latencia No Muestra Datos

Síntomas: El panel de Latencia JSONRPC por Método muestra "Sin datos"

Causas posibles:

- No hay tráfico JSONRPC en el rango de tiempo seleccionado
- Desajuste de formato de registro (regex no coincide)

Resolución:

1. Verifique que existan registros para el rango de tiempo:

```
{job="cgrates-jsonrpc"} |= "method="
```

2. Verifique que el campo `latency_ms` esté presente en los registros
3. Amplíe el rango de tiempo

Alto Conteo de Solicitudes Pendientes

Síntomas: Los registros de depuración muestran muchas solicitudes almacenadas pero pocas finalizadas

Causas posibles:

- Solicitudes sin respuestas (cliente desconectado)
- Respuesta capturada antes de la solicitud (orden de paquetes)
- Desajuste de ID de solicitud entre solicitud y respuesta

Resolución:

1. El registrador limpia automáticamente las solicitudes pendientes que tienen más de 60 segundos
2. Verifique si hay problemas de red entre el cliente y CGrateS
3. Verifique que CGrateS esté respondiendo a las solicitudes

Registros de Archivos

Además de los registros del diario de systemd, Alloy recopila archivos de registro específicos:

Registros de FreeSWITCH (Servidores de Aplicaciones)

Ruta	Etiqueta de Trabajo
`/var/log/freeswitch/freeswitch.log`	`freeswitch`
`/var/log/omnitas-freeswitch.log`	`freeswitch-structured`

Registros de Nginx (OmniCRM)

Ruta	Etiqueta de Trabajo	Etiqueta de Tipo
`/var/log/nginx/access.log`	`nginx`	`access`
`/var/log/nginx/error.log`	`nginx`	`error`

JSONRPC de CGrateS (OCS)

Ruta	Etiqueta de Trabajo
`/var/log/cgrates/jsonrpc.log`	`cgrates-jsonrpc`

Solución de Problemas

Registros No Aparecen en Grafana

Síntomas: No se ven registros en los dashboards de Loki

Causas posibles:

- Servicio Alloy no en ejecución en el host fuente
- Servicio Loki no en ejecución en el host de monitoreo
- Conectividad de red entre hosts bloqueada
- Alloy no puede alcanzar a Loki en el puerto 3100

Resolución:

1. Verifique el estado de Alloy en el host fuente:

   ```
   systemctl status alloy
   journalctl -u alloy -f
   ```

2. Verifique el estado de Loki en el host de monitoreo:

   ```
   systemctl status loki
   journalctl -u loki -f
   ```

3. Pruebe la conectividad desde el origen hasta el monitoreo:

   ```
   curl http://<monitoring-ip>:3100/ready
   ```

4. Verifique que el firewall permita TCP 3100 desde todos los hosts hacia el monitoreo

Alto Uso de Disco de Alloy

Síntomas: `/var/lib/alloy` consume un espacio significativo en disco

Causas posibles:

- Loki inalcanzable por un período prolongado
- Búfer WAL llenándose

Resolución:

1. Verifique la conectividad de Loki (ver arriba)
2. Si Loki está inactivo, los registros se almacenan localmente en el WAL
3. Una vez que Loki esté disponible, los registros en búfer se envían automáticamente
4. Los segmentos en búfer más antiguos que 1 hora se eliminan (por diseño)

Almacenamiento de Loki Lleno

Síntomas: Loki deja de aceptar registros, disco lleno en el servidor de monitoreo

Causas posibles:

- El volumen de registros excede el disco disponible antes de que la retención elimine registros antiguos
- La compactación de retención no se está ejecutando

Resolución:

1. Verifique el uso de almacenamiento de Loki:

```
du -sh /var/lib/loki/
```

2. Verifique que el compactador esté en ejecución (verifique los registros de Loki)
3. Active manualmente la compactación si es necesario reiniciando Loki
4. Considere aumentar la asignación de disco o reducir el período de retención

Faltan Etiquetas de Componente

Síntomas: Los registros aparecen pero la etiqueta `component` es `infrastructure` en lugar del valor esperado

Causas posibles:

- Host no en el grupo de inventario esperado
- Configuración de Alloy no regenerada después del cambio de inventario

Resolución:

1. Verifique que el host esté en el grupo de inventario correcto
2. Vuelva a ejecutar Ansible para regenerar la configuración de Alloy:

   ```
   ansible-playbook -i hosts/customer/hosts.yml services/all.yml -
   -limit <hostname>
   ```

3. Reinicie Alloy:

   ```
   systemctl restart alloy
   ```

Puertos de Servicio

Servicio	Puerto	Protocolo	Descripción
Loki	3100	HTTP	Ingesta de registros y API de consulta
Loki	9096	gRPC	gRPC interno (no utilizado externamente)
Alloy	12345	HTTP	Métricas y UI de Alloy
Grafana	3000	HTTP	Acceso al dashboard

Documentación Relacionada

- Monitoreo y Observabilidad: Grafana, Prometheus, dashboards y alertas
- Arquitectura de Despliegue: Arquitectura general del sistema
- Configuración del Archivo de Hosts: Configuración del inventario
- Referencia de Configuración: Referencia completa de parámetros

Núcleo conmutado por circuitos (CS)

El dominio legado conmutado por circuitos (voz y SMS 2G/3G) se implementa mediante el playbook `services/cs.yml`. Proporciona el BSC orientado a la radio, una puerta de enlace de medios independiente, el softswitch OmniMSC y la pila de señalización SS7/SIGTRAN.

Descripción general

`cs.yml` ejecuta estos roles, cada uno contra su propio grupo de inventario:

Componente	Rol	Grupo de inventario	Paquetes	D
BSC	`circuit_switched/bsc`	`bsc`	BSC, MGW	Co de Ba de M pr va 2(
MGW	`circuit_switched/mgw`	`mgw` (hosts que no están también en `bsc`)	`osmo-mgw`	Pu En M in pa ur
MSC	`circuit_switched/omnimsc`	`msc`	`omnimsc`	Sc Ce Co M
SS7	`omniss7`	`ss7`	`omniss7`	Pi SS (S SM CA te

```
- name: Setup MSC
  hosts: msc
  roles:
    - {role: circuit_switched/omnimsc, become: yes}
```

BSC (`circuit_switched/bsc`)

Instala un **BSC** y **Puerta de Enlace de Medios (MGW)** desde el servidor APT de Omnitouch para pruebas y validación conmutadas por circuitos. El rol configura ambos servicios.

- Renderiza los archivos de configuración del BSC y MGW a partir de las plantillas del rol.
- Instala una anulación de systemd para el servicio MGW que **desactiva la programación de CPU en tiempo real**. Limpia `CPUSchedulingPolicy`, `Priority` y `AmbientCapabilities`. Las máquinas virtuales no admiten la programación en tiempo real.
- Los controladores reinician cada servicio. Cada controlador luego **verifica que el servicio alcance `active`** (6 reintentos, 5 s de separación) antes de continuar.

MGW (`circuit_switched/mgw`)

Instala una **Puerta de Enlace de Medios** independiente (`osmo-mgw`) desde el servidor APT de Omnitouch. Úsala en un sitio que tenga un MSC pero no un BSC. El MSC ancla el medio de cada llamada CS en un MGW a través de MGCP.

- Renderiza la configuración del MGW. MGCP se vincula a `mgw_bind_ip` en `mgw_port`. RTP se vincula a `mgw_rtp_bind_ip` y utiliza `mgw_rtp_port_range`. Ambas direcciones se configuran por defecto a `ansible_host` del host. El puerto se configura por defecto a 2427 y el rango a `4002 16001`.
- Instala la misma anulación de systemd que el rol BSC. La anulación desactiva la programación de CPU en tiempo real, que las VMs y LXCs no admiten.
- El controlador reinicia el servicio y luego ejecuta la verificación de bucle de fallos `verify_service`.
- Un host que también está en el grupo `bsc` ya ejecuta el MGW del rol BSC. Los playbooks omiten este rol en dicho host.

MSC (`circuit_switched/omnimsc`)

Instala el paquete `omnimsc` y lo configura de manera similar a un NF de 5GC:

- Resuelve `group_vars_omnicore` y deriva `license_server_api_urls` del grupo `license_server`.
- Renderiza el `runtime.exs.j2` incorporado en `/etc/omnimsc/`. Si `omnimsc_template_config` está configurado, renderiza una plantilla personalizada de `group_vars/` en su lugar.
- Asegura un directorio de salida CDR (`omnimsc_cdr_output_dir`, por defecto `/var/lib/omnimsc/cdr`).
- Genera un certificado TLS autofirmado para la interfaz web / API.
- Utiliza el patrón de enlace de configuración versionada + `runtime.exs`, reiniciando en caso de cambio.
- Construye la lista de puertas de enlace MGCP en este orden:

  i. `omnimsc_mgcp_gateways`, si está configurado (una anulación explícita).
  ii. Una puerta de enlace por host en el grupo `mgw` (dirección `ansible_host`, puerto `mgw_port`, por defecto 2427).
  iii. Una puerta de enlace por host en el grupo `bsc` (el rol BSC co-localiza un MGW).

  Si ninguno de estos existe, el rol falla antes de instalar cualquier cosa. La puerta de enlace de medios en uso es `omnimsc_media_gateway` (por defecto `MGW-01`, la primera puerta de enlace en la lista).

SS7 (`omniss7`)

Un único rol que instala el paquete `omniss7` y renderiza **una plantilla de configuración por rol SS7**, seleccionada por la variable de host `ss7_role`:

`ss7_role`	Plantilla	Función
`stp`	`stp.j2`	Punto de Transferencia de Señales (enrutamiento SS7)
`hlr`	`hlr.j2`	Registro de Ubicación del Hogar
`smsc`	`ipsmgw.j2`	SMSC / IP-SM-GW
`camel`	`camel.j2`	Control de servicio CAMEL
`tester`	`tester.j2`	Agente de prueba/carga de señalización

Selectores adicionales:

- Configurar `ss7_protocol: m2pa` renderiza `m2pa.j2` (un punto final M2PA) en lugar de la plantilla `tester`.
- Configurar `ss7_template_config` (una ruta bajo `group_vars/`) anula **todas** las plantillas incorporadas con una configuración específica del host.

Al igual que los otros componentes de Elixir, `omniss7` utiliza el patrón de enlace de configuración versionada + `runtime.exs` y refleja el enlace en `/opt/omniss7/releases/*`.

Carga CAMEL

Las llamadas CS de un suscriptor con datos CAMEL son cobradas por la puerta de enlace CAMEL, a través del mismo OCS de CGrateS y arrendatario (`ocs_tenant`) que utiliza el TAS para Ro:

Variable de host	Host	Función
`camel_gsmscf_global_title`	`ss7_role: camel`	El título global gsmSCF de la puerta de enlace CAMEL. `stp.j2` agrega una ruta GT a él, y `camel.j2` lo utiliza como el GT de llamada de las respuestas CAP.
`hlr_camel_subscriptions`	`ss7_role: hlr`	Una lista de reglas `imsi_prefix` con un `o_csi` y/o `t_csi` (`service_key`, `gsmscf_address`, `default_call_handling`: `release_call` o `continue_call`), y un `mo_sms_csi` opcional (`service_key`, `gsmscf_address`, `default_sms_handling`: `release_transaction` o `continue_transaction`). El HLR los envía al VLR en InsertSubscriberData. El MSC envía InitialDP para una llamada, o InitialDPSMS para un SMS MO, a `gsmscf_address`.

`camel.j2` apunta la puerta de enlace al primer host `ocs` (`:2080/jsonrpc`, flujo de firewall 31b). Cobra llamadas en bloques de 10 s, como lo hace el TAS, y cobra cada SMS MO como una unidad `*sms`. Cuando ambas variables están configuradas, el rol ran_simulator establece la variable de escenario `camel_origin_host`, que habilita los preajustes de carga CAMEL.

USSD (`support/ussd_app`)

El USSD originado en el móvil va del MSC al HLR como MAP `processUnstructuredSS-Request`. Cuando el inventario tiene un grupo `ussd`, `hlr.j2` activa la puerta de enlace USSD omniss7 con una ruta de captura general (`*`) a `http://<primer host ussd>:8380/ussd`. Sin el grupo, la puerta de enlace no se configura.

El rol `support/ussd_app` (playbook `services/ussd.yml`) instala una pequeña aplicación de callback en los hosts `ussd` como el servicio systemd `ussd-app`. Registra una línea por solicitud en journald.

Marcación / respuesta	Respuesta
`*100#`	Fin: `Tu saldo es $12.34`
`*102*<dígitos>#`	Fin: `Enviaste <dígitos>`
`*101#`	Continuar: `1. Saldo` / `2. Datos` / `3. Salir`
Respuesta `1`, `2`, `3`	Fin: `Saldo: $12.34`, `Datos: 1.5 GB restantes`, `Adiós`
Cualquier otra respuesta	Fin: `Opción no válida`
Cualquier otro código	Fin: `Código desconocido <código>`

El puerto (`ussd_app_port`, 8380) pertenece al rol. `roles/omniss7` lo lee de los valores predeterminados del rol, y el firewall lo declara en el `firewall_services.yml` del rol. El playbook publica cada caso anterior en la aplicación y falla si una respuesta es incorrecta.

Documentación relacionada

- Playbooks de servicio - Todos los playbooks de servicio y la jerarquía de implementación
- Núcleo 5G - El núcleo 5G independiente y el rol compartido `5gc_nf`
- Configuración de variables de grupo - Plantillas de configuración personalizadas (por ejemplo, `ss7_template_config`, `omnimsc_template_config`)
- **Configuración de OmniCore**: https://docs.omnitouch.com.au/docs/repos/OmniCore - Lo que va en los archivos de configuración de componentes

Referencia de Configuración

Visión 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 muchas configuraciones. Sin embargo, algunos componentes se configuran exclusivamente desde `group_vars` y no tienen un valor predeterminado de rol. OmniCRM es el ejemplo notable (ver más abajo). En esos casos, un valor faltante falla la ejecución de manera ruidosa en lugar de retroceder a un valor predeterminado.

> **Nota de Seguridad**: Varios valores predeterminados de rol son marcadores de posición inseguros destinados solo para uso en laboratorio

> y **deben** ser sobreescritos para cualquier implementación real. Ejemplos incluyen contraseñas de bases de datos (por ejemplo, `sql_pass: password`, `omnitouch2024`) y credenciales de repositorio APT (`omni`/`omni`). Siempre establezca secretos reales y únicos en su inventario o `group_vars` antes de implementar.

Planificación de Red

Antes de configurar 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
dra_peer_exclude: True            # Excluir de listas de pares
Diameter generadas
gtpc_weight: 100                  # Participación de SGW/PGW dentro
de su nivel de prioridad (selección de pares MME)
gtpc_priority: 100                # Nivel de prioridad de SGW/PGW,
menor es preferido
```

Consulte Exclusión de Pares Diameter para `dra_peer_exclude`.

`gtpc_weight` y `gtpc_priority` se leen de cada host `sgwc` / `pgwc` cuando el MME construye su lista de pares GTP-C estática. Ambos tienen un valor

predeterminado de `100`, por lo que un par puede ser ponderado o clasificado por encima o por debajo del resto del grupo sin cambiar cada otro host. El MME elige el nivel de `gtpc_priority` presente numéricamente *más bajo* y divide el tráfico dentro de él por `gtpc_weight` (RFC 2782); niveles más altos son de reserva y solo toman tráfico una vez que no queda ningún par en un nivel inferior. Una capacidad MME relativa se establece de la misma manera, con `relative_capacity` en el host MME (valor predeterminado `100`, rango 0-255).

Variables Globales (all:vars)

La sección `all:vars` contiene configuraciones a nivel de implementación. Las implementaciones modernas utilizan variables globales mínimas, con muchas configuraciones cubiertas por valores predeterminados de rol. Tenga en cuenta las excepciones a continuación (OmniCRM, Grafana SMTP) donde la configuración debe ser proporcionada explícitamente porque no existe un valor predeterminado de rol.

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: "Customer Lab"
site_name: Customer-Site
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 (puede
sobreescribirse por MME)
```

Mostrado: Nombres de red mostrados en dispositivos UE en Configuración > Red Móvil.

Configuración de DNS

```
manage_resolv_conf: True         # Establecer en False para evitar
que Ansible gestione /etc/resolv.conf
```

Nota: Cuando `manage_resolv_conf` está configurado en `False`, Ansible no sobrescribirá `/etc/resolv.conf` en ese host. Esto es útil para hosts que

requieren una configuración DNS personalizada o son gestionados por sistemas externos. Puede configurarse por host en el inventario o globalmente en `all:vars`.

Configuración del Repositorio APT

`apt_repo` es la única fuente de verdad. Defínalo una vez. Las variables de descarga binaria (`remote_apt_*`) y la URL `apt_download_base` se derivan automáticamente de él. No las configure por separado en un inventario normal.

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 configure explícitamente en `False`)
- `apt_repo.apt_server` se establece automáticamente en la IP del primer servidor de caché

```
use_apt_cache: False              # Acceso directo al repositorio
(True = caché APT local)

apt_repo:
  apt_server: "apt.<region>.omnitou.ch"  # FQDN proporcionado por
Omnitouch (o IP de caché)
  apt_repo_username: "your-username"     # Requerido para acceso
directo
  apt_repo_password: "your-password"     # Requerido para acceso
directo
  # apt_repo_port: 443                   # Opcional;
predeterminado en 443 (HTTPS).
                                         # El esquema sigue el
puerto: 443 -> https, de lo contrario http.
```

El esquema y el puerto se establecen en **HTTPS / 443** para acceso directo y **HTTP / 8080** en modo caché. Solo un servidor de espejo de caché (`apt_cache_servers` grupo) establece `remote_apt_*` manualmente, para describir el servidor Omnitouch upstream del que se sincroniza.

Vea: Sistema de Caché APT

Servidor de Licencias

```
# Opcional: auto-derivado del grupo de inventario `license_server`
cuando no está establecido.
# Cada host en ese grupo se convierte en
https://<ansible_host>:8443/api.
license_server_api_urls: ["https://10.10.2.150:8443/api"]
```

Nota: Si existe un grupo `license_server` en el inventario, `license_server_api_urls` se deriva automáticamente de sus miembros; solo necesita configurarlo manualmente para anular ese comportamiento.

Vea: Servidor de Licencias

Configuraciones de MME

El MME selecciona pares S-GW y P-GW utilizando un método configurable, establecido por gateway bajo `omnimme`. El método puede ser un único valor o dividido por tráfico `home`/`roaming`. Los valores válidos son `dns_lookup` (resolver pares a través de DNS) y `random_peer` (elegir de pares configurados).

```
omnimme:
  sgw:
    selection_method: random_peer        # valor único, o...
    # selection_method:
    #   home: random_peer
    #   roaming: dns_lookup
  pgw:
    selection_method:
      home: random_peer
      roaming: dns_lookup
```

Configuraciones de AMF

```
# tac_list también impulsa la supported_ta_list del AMF.  Cada TAC
aquí es
# anunciado a los gNB conectados durante la Configuración NG.  Si
no se establece, el AMF predetermina
# solo TAC 1, que rechaza cualquier gNB que no transmita TAC 1.
tac_list: [1, 3]

# Deshabilitar cifrado NAS (forzar NEA0).  Predeterminado en
verdadero porque omniamf
# >=20260727 negocia NEA2 pero no puede decodificar NAS cifrado
ascendente (Modo de Seguridad
# Completo y mensajes posteriores), causando que cada registro 5G
falle.  Establezca en falso una vez que el proveedor confirme que
el descifrado de NEA2 ascendente funciona
# contra UEs reales.  La integridad NAS (NIA2) no se ve afectada.
amf_force_null_ciphering: true
```

Configuraciones de SAEGW

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

Configuraciones de OmniHSS

OmniHSS construye su lista de pares Diameter automáticamente a partir del inventario. Utiliza los grupos `dra`, `mme`, `pgwc`, `scscf`, `icscf`, `pcscf` y `applicationserver` (cuáles de esos se utilizan depende de `packet_core_dra_support` / `ims_dra_support`). `exclude_diameter_peers` elimina hosts individuales de esa lista generada, para pares que están en el inventario pero no deberían ser pares Diameter del HSS.

```
omnihss:
  exclude_diameter_peers:          # opcional; predeterminado en
[] (nada excluido)
    - customer-as01
    - customer-as02
```

- Las entradas se comparan contra **`inventory_hostname`**, no el FQDN o IP Diameter.

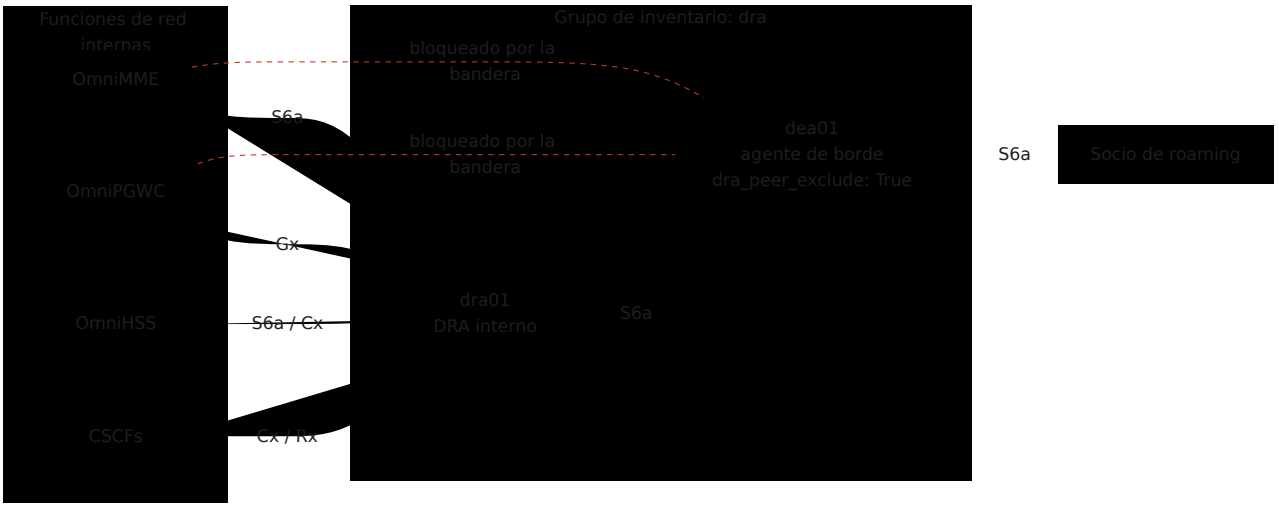
- La exclusión se aplica a cada grupo auto-descubierto, por lo que una lista cubre un host independientemente de en qué grupo se encuentre.
- Los nombres deben existir en el inventario. El rol asegura esto antes de implementar, por lo que un error tipográfico falla la ejecución de manera ruidosa en lugar de dejar silenciosamente al par en `/etc/omnihss/runtime.exs`.
- Establezca esto donde se establece el resto del diccionario `omnihss` (por host, o bajo `vars:` de un grupo). Debido a que Ansible reemplaza en lugar de fusionar diccionarios, debe vivir en el mismo bloque `omnihss` que las otras claves.
- Esto afecta solo la lista de pares Diameter. La lista de pares SIP S-CSCF bajo `config :hss, ims:` no está relacionada y no es filtrada por esta clave.

> `exclude_diameter_peers` es una clave del diccionario `omnihss`. Filtra solo la lista de pares de OmniHSS. Para eliminar un host de la lista de pares de cada función de red, use `dra_peer_exclude` en su lugar. Los dos mecanismos son independientes.

Exclusión de Pares Diameter

OmniCore construye la lista de pares Diameter de cada función de red a partir de los grupos de inventario. Una función de red agrega cada host de los grupos con los que hace pares. `dra_peer_exclude` elimina un host de esas listas generadas. Establezca la bandera en el host que no debe convertirse en un par Diameter.

El caso de uso principal es un **Agente de Borde Diameter (DEA)**. Un DEA es miembro del grupo `dra`, porque ejecuta el mismo software OmniDRA y necesita el mismo tratamiento de implementación, firewall y monitoreo. Un DEA transporta tráfico solo a socios de roaming externos. Cada función de red interna debe enviar su tráfico al DRA interno. Establezca `dra_peer_exclude: True` en cada DEA para mantener al DEA fuera de la lista de pares de las funciones de red internas.



Configuración

Establezca la bandera en la entrada del host, junto a `ansible_host`.

```
dra:
  hosts:
    dra01:                  # DRA interno. Un par de cada función
de red interna.
      ansible_host: 10.4.2.150
    dea01:                  # Agente de borde. Nunca un par de una
función de red interna.
      ansible_host: 203.0.113.10
      packet_core_dra_support: False
      ims_dra_support: False
      dra_peer_exclude: True
```

Parámetro	Tipo	Requerido	Predeterminado	
`ansible_host`	Cadena	Sí	-	
`dra_peer_exclude`	Booleano	No	`False`	
`packet_core_dra_support`	Booleano	No	`False`	
`ims_dra_support`	Booleano	No	`False`	

Alcance

La bandera funciona en dos direcciones. Una función de red omite un host excluido del grupo `dra`, y el DRA omite un host excluido de cualquier otro grupo.

Funciones de red que filtran el grupo `dra`:

Rol	Lista de pares
`omnimme`	Pares S6a y T6a
`omnipgwc`	Pares Gx y Gy
`omnihss`	Pares S6a, Cx y Sh
`omniepdg`	Pares SWm y S6b
`omnitwag`	Pares S6b
`omniscef`	Pares T6a
`omnisep`	Pares Sh
`omnilcs`	Pares SLg y SLh
`cscf`	Pares Cx, Rx y Dx del P-CSCF, I-CSCF y S-CSCF
TAS y SMSC (variables de grupo del sitio)	Pares Sh y Ro

Grupos que el DRA filtra: `hss`, `mme`, `pgwc`, `epdg`, `scef`, `twag`, `ocs`, `omnilcs`, `omnimessage`, `pcscf`, `icscf`, `scscf`, `xcap` y `applicationserver`.

Reglas de Comportamiento

- Establezca la bandera en el host a excluir. No la establezca en el host que construye la lista de pares.
- La bandera se aplica solo a listas de pares generadas. Un par estático explícito nunca se filtra. Esto cubre `dra_peers` en el DRA y `diameter_static_peers` en una función de red.
- La bandera no cambia las reglas del firewall, los objetivos de Prometheus o el plan de IP. Un host excluido mantiene su acceso a la red y su monitoreo.

- La bandera entra en efecto en la próxima ejecución del rol. Ejecute el rol de cada función de red afectada después de establecer la bandera. El `runtime.exs` desplegado en ese host mantiene el viejo par hasta entonces.

Síntomas de una exclusión faltante

Si una función de red interna se conecta a un DEA, verifique esta bandera primero. Los síntomas son:

- El MME o el PGW-C abren una conexión SCTP a la dirección IP pública del DEA.
- El DEA registra un intento de conexión de par no autorizado, porque `allow_undefined_peers_to_connect` es `False`.
- La función de red informa un par en un bucle de reconexión permanente.

Para corregir esto, establezca `dra_peer_exclude: True` en el host DEA. Luego ejecute el rol de la función de red nuevamente. Luego confirme que la lista de pares en el `runtime.exs` desplegado contenga solo los DRAs internos.

Configuraciones de OmniCRM

La configuración de OmniCRM se obtiene **exclusivamente** de los `group_vars` del cliente (bajo `hosts/`); no hay valores predeterminados incorporados en el rol para secretos del sitio.

El rol `omnicrm` despliega OmniCRMv2 desde la fuente: el controlador envía un tarball de la verificación de OmniCRMv2, el host construye un venv y la UI, y el rol plantilla `crm_config.yaml` en `/etc/omnicrm/crm_config.yaml`, sincroniza `deployment_assets/`, y ejecuta las unidades `omnicrm-v2-*` detrás de nginx. La marca de la UI proviene del bloque `ui:` (tiempo de ejecución `GET /api/public/ui-config`).

```
crm_config.yaml                  # API + ui: config (group_vars; requerido)
deployment_assets/               # logotipos/iconos opcionales para /etc/omnicrm/deployment_assets
```

Configuraciones SMTP de Grafana

Utilizado por el rol de monitoreo para el correo electrónico de alerta de Grafana. `grafana_smtp_user` y `grafana_smtp_password` son **requeridos** (sin valor predeterminado: debe configurarlos); `grafana_smtp_host` tiene un valor predeterminado de `in-v3.mailjet.com:587`. Consulte `roles/monitoring/templates/grafana/grafana.ini`.

```
grafana_smtp_user: "smtp-user"           # REQUERIDO, sin valor predeterminado
grafana_smtp_password: "smtp-password"   # REQUERIDO, sin valor predeterminado
grafana_smtp_host: "in-v3.mailjet.com:587"  # opcional, este es el valor predeterminado
```

Configuraciones IMS

```
ims_dra_support: False           # Enrutar IMS a través de DRA
sipcapture_client_enabled: False # Enviar SIP/RTP a la caja de captura VoIPmonitor (grupo siptrace)
```

Configuración del Monitor RAN

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

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

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

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

Configuración del Firewall

Aplicado por `services/firewall.yml`. El conjunto completo de parámetros, las zonas de plano y los modos de monitoreo/aplicación están documentados en Firewall por Rol; solo `allowed_oam_subnets` es obligatorio.

```yaml
firewall:
  allowed_oam_subnets:            # fuente de gestión (SSH, UI,
scrape, APIs de gestión)
    - '10.0.1.0/24'
    - '10.0.0.0/24'
  allowed_ran_subnets: []         # S1AP / NGAP / GTP-U / Abis
  allowed_ue_voice_subnets:       # SIP de UE externo en el P-CSCF
    - '10.0.1.0/24'
  allowed_carrier_voice_subnets:  # SIP de interconexión de
operadores
    - '10.0.1.0/24'
  zones:                          # zonas de plano — ver
docs/firewall.md
    internal: ['10.0.0.0/8']
  ue_pools: []
```

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

Configuración del Hipervisor

Proxmox

Un sitio se despliega como VMs o contenedores LXC, seleccionados por entrada `proxmoxServers` a través de `deployment_type` (predeterminado `vm`). Todas las entradas en un sitio deben coincidir; mezclar falla la validación. Consulte Implementación de Proxmox para el recorrido completo.

Sitio VM

```
proxmoxServers:
  customer-prxmx01:
    # deployment_type omitido → predeterminado en "vm"
    proxmoxServerAddress: 10.10.0.100
    proxmoxServerPort: 8006
    proxmoxApiTokenName: AnsibleToken
    proxmoxApiTokenSecret: "token-secret"
    proxmoxNodeName: pve01
    proxmoxTemplateName: ubuntu-24.04-cloud-init-template
    proxmoxTemplateId: 9000
    proxmoxTemplateUser: omnitouch        # usuario opcional de
cloud-init; predeterminado en la primera clave de local_users
    proxmoxTemplatePassword: omnitouch    # contraseña opcional de
cloud-init; predeterminada en el nombre de usuario de cloud-init
    storage: SSD_RAID0            # opcional, almacenamiento
predeterminado de VM
```

Sitio LXC

```yaml
proxmox_lxc_nameserver: "1.1.1.1"  # opcional, inyectado en LXCs
al crear

proxmoxServers:
  customer-prxmx01:
    deployment_type: lxc
    proxmoxServerAddress: 10.10.0.100
    proxmoxServerPort: 8006
    proxmoxApiTokenName: AnsibleToken
    proxmoxApiTokenSecret: "token-secret"
    proxmoxNodeName: pve01
    proxmoxLxcOsTemplate: "local:vztmpl/ubuntu-24.04-
standard_24.04-2_amd64.tar.zst"
    proxmoxLxcDefaultStorage: SSD_RAID0   # opcional, fallback de
rootfs
```

Sobrescrituras a Nivel de Grupo (ambos tipos)

```yaml
dns:
  vars:
    proxmox_interface: vmbr0              # requerido: puente
    gateway: 10.10.0.1                    # requerido
    netmask: 255.255.255.0                # requerido
    vlanid: 100                           # opcional
    proxmoxLxcCores: 2                    # opcional (solo LXC)
    proxmoxLxcMemoryMb: 4096              # opcional (solo LXC)
    proxmoxLxcDiskSizeGb: 30              # opcional (solo LXC)
    proxmoxLxcRootFsStorageName: SSD_RAID0 # opcional (solo LXC)
    host_vm_network: vmbr1                # sobrescritura de
puente opcional
```

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

- IP Planning Standard - Arquitectura de red y directrices de asignación de IP
- Hosts File Configuration - Cómo estructurar archivos de hosts
- Group Variables Configuration - Cuándo y cómo usar group_vars
- Netplan Configuration - IPs secundarias y configuración de múltiples NIC
- Deployment Architecture - Cómo interactúan los componentes
- APT Cache System - Gestión de paquetes
- License Server - Configuración de licencias

Documentación del Producto

Para guías operativas detalladas y configuración avanzada:

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

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

Visión General de la Arquitectura de Despliegue

Visión General

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 IPs públicas.

Ejemplo Completo de Despliegue

0. Aprovisionamiento de Infraestructura (Opcional)

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

```
# Desplegar VMs y/o contenedores LXC en Proxmox (el tipo de
despliegue por host decide cuál)
ansible-playbook -i hosts/Customer/hosts.yml
util_playbooks/proxmox.yml
```

Consulte: Despliegue de VM/LXC en Proxmox

1. Definición de Infraestructura (Archivo de Hosts)

```yaml
# 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` almacena cualquier anulación de configuración necesaria a nivel de host, sitio o red.

Por ejemplo, una carpeta contiene su configuración de OmniMessage SMSc. Los troncos SIP a los que se conecta su TAS se encuentran aquí. Toda su lógica de enrutamiento Diameter también reside aquí.

OmniCRM también se configura aquí. Sus configuraciones son solo de `group_vars`. Su fuente de GitHub se clona en el controlador de Ansible (`delegate_to: localhost`) con las credenciales del propio controlador antes de que se sincronice con el objetivo, por lo que ninguna credencial llega al host objetivo.

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"]
```

Consulte: Servidor de Licencias

5. Ejecutar Despliegue

Puede desplegar componentes individuales. Por ejemplo, ejecute `services/twag.yml`. El playbook `services/all.yml` maneja todo. Use `--limit=myhost` o `--limit=mmee,sgw` para limitar los hosts en los que trabaja.

```
# 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
```

Modelos de Despliegue

Hay dos formas de obtener el kit de herramientas de OmniCore en un controlador y ejecutarlo.

Modelo A: Clonar el repositorio en el controlador

El modelo tradicional: clone este repositorio en su controlador de Ansible, mantenga su inventario bajo `hosts/`, y ejecute los playbooks directamente desde la copia. Este es el modelo asumido por el ejemplo de despliegue anterior.

Modelo B: Autoaprovisionamiento de `omnicore` .deb

OmniCore también se empaqueta como un paquete Debian `omnicore` autónomo, por lo que un controlador puede instalar el kit de herramientas desde apt en lugar de clonar el repositorio.

- **Qué instala.** El paquete (`name: omnicore`, consulte `packaging/nfpm.yaml`) incluye los roles de Ansible, playbooks, colecciones *vendidas*, un intérprete de Python empaquetado (`/opt/omnicore/python`) y un wheelhouse fuera de línea en `/opt/omnicore`. El paso posterior a la instalación (`packaging/postinstall.sh`) construye un venv autónomo (`/opt/omnicore/venv`) a partir del intérprete empaquetado + wheelhouse y enlaza simbólicamente los `ansible*` empaquetados en `PATH` a través de `/usr/local/bin`, de modo que el controlador ejecute las versiones exactas de `uv.lock` de Ansible y sus bibliotecas de Python: sin Ansible de distribución, sin Python del sistema, sin red. Solo se utilizan herramientas genuinas del sistema (`git`, `rsync`, `openssh-client`) que son traídas por apt. El paquete es **amd64** y está dirigido a **Ubuntu 22.04+** (glibc ≥ 2.35).
- **Cómo se construye.** `packaging/build.sh` organiza el repositorio en `/opt/omnicore` (excluyendo `hosts/`, `.git/`, entornos virtuales y archivos de construcción), vende las colecciones de Ansible desde `requirements.yml`, empaqueta el intérprete independiente + un wheelhouse de las dependencias de Python bloqueadas, ejecuta un

escaneo de secretos que falla en cerrado, y luego construye el `.deb` con nfpm.
- **Cómo se publica.** `.github/workflows/build-deb.yml` ejecuta `build.sh` en cada push a `main` y publica el `.deb` (más un `_metadata.json` como acompañante) como una Release de GitHub. El script `discover_github_debs.py` del servidor apt descubre automáticamente ese activo de lanzamiento en toda la organización Omnitouch y lo agrega a aptly. No se necesitan cambios en el rol de apt.

Flujo del cliente:

```
# 1. Instalar el kit de herramientas desde apt
apt-get install omnicore

# 2. Proporcionar su propio inventario
#    (el paquete no incluye uno - vive fuera de /opt/omnicore)
vi /etc/omnicore/inventory

# 3. Ejecutar playbooks desde el kit de herramientas instalado
cd /opt/omnicore
ansible-playbook -i /etc/omnicore/inventory services/all.yml
```

Debido a que el inventario vive en `/etc/omnicore/inventory` (separado del kit de herramientas instalado en `/opt/omnicore`), actualizar el paquete nunca toca su definición de infraestructura o personalización.

Documentación Relacionada

- Introducción al Despliegue de Ansible - Comenzando
- Playbooks de Servicio - **Referencia y jerarquía de playbooks**
- Configuración del Archivo de Hosts - Definiendo 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
- Monitoreo y Observabilidad - Grafana, Prometheus, alertas y paneles

- Registro Centralizado - Recolección de registros de Loki y Alloy

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/

Firewall por Rol

El firewall por rol proporciona a cada host un conjunto de reglas `iptables` e `ip6tables` que se deriva de un único modelo declarativo del tráfico de la red, en lugar de reglas escritas a mano por host. Un archivo describe cada flujo permitido en el núcleo (quién puede alcanzar qué rol, en qué protocolo y puerto, y por qué). A partir de ese modelo, la plataforma produce dos cosas que siempre están de acuerdo entre sí:

- el **conjunto de reglas aplicadas** en cada host, y
- una **matriz de comunicaciones / puertos** (CSV) para revisión y para entregar a un firewall ascendente.

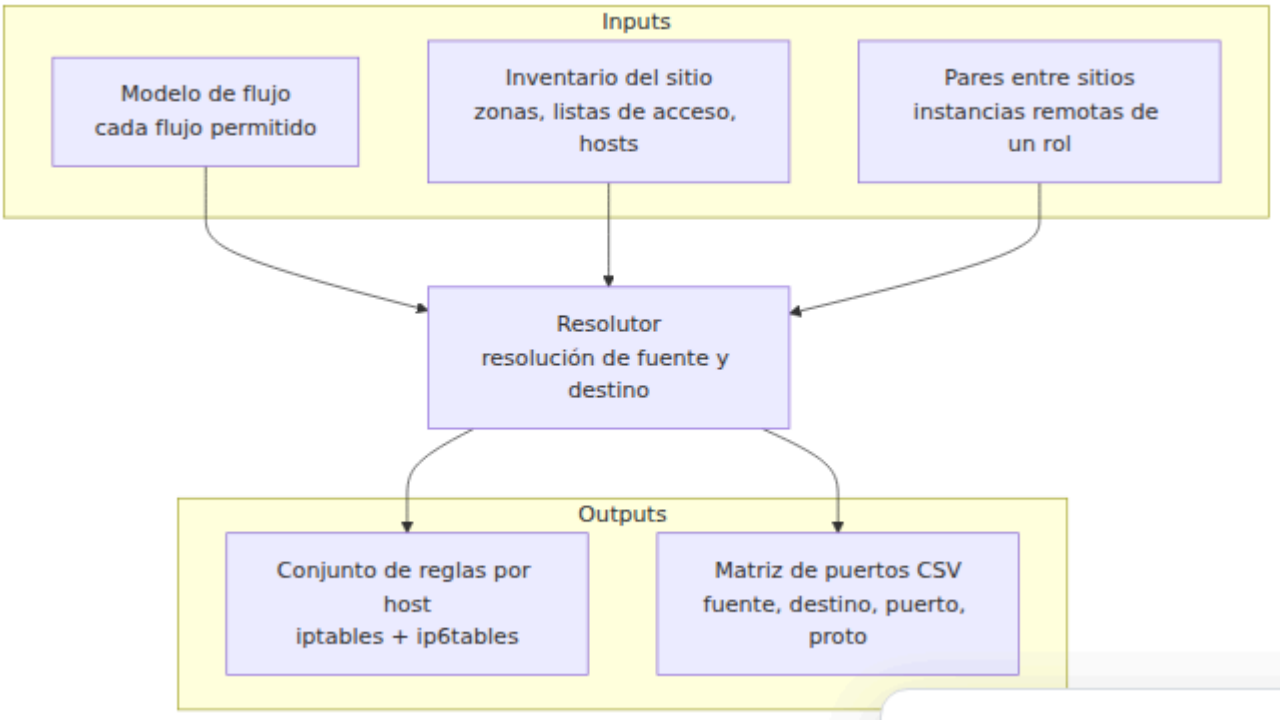
Las reglas se ejecutan en uno de dos modos: **monitor** (registrar cada flujo, no bloquear nada) o **hacer cumplir** (denegación por defecto). El bloqueo es deliberado y a prueba de fallos, y cada aplicación está protegida por un retroceso automático en el host para que un conjunto de reglas erróneo no pueda bloquear a los operadores.

Tabla de Contenidos

- Descripción General de la Arquitectura
- El Modelo de Flujo
- Tipos de Fuentes
- Catálogo de Servicios
- Configuración del Sitio
- Modos de Operación
- Pares entre Sitios
- Resolución de Direcciones
- Matriz de Comunicaciones / Puertos
- Protección contra Bloqueos
- IPv6 y Doble Pila
- Registro
- Precauciones Operativas

Descripción General de la Arquitectura

Un único modelo más el inventario propio de cada sitio alimentan un resolutor. El resolutor produce tanto las reglas en el host como la matriz de puertos, por lo que la documentación nunca puede describir algo diferente de lo que realmente se aplica.



El tráfico descartado y auditado se registra a través de la tubería de observabilidad estándar:



El Modelo de Flujo

El modelo de flujo es la única fuente de verdad para **quién puede alcanzar qué**. Contiene dos partes: un conjunto de valores predeterminados estructurales (que no llevan dirección del sitio) y la lista de flujos. Un flujo no dice nada sobre números de puerto directamente: nombra un **servicio**

expuesto por el rol de destino (ver Catálogo de Servicios), y el puerto se busca en la declaración de ese rol:

```
- id: '13'                          # etiqueta estable, aparece en el comentario de cada regla
  dst: hss                          # grupo de inventario al que se aplica la regla
  service: diameter                 # servicio nombrado expuesto por el rol hss (-> 3868/tcp+sctp)
  purpose: 'Diameter'               # descripción, mostrada en la matriz
  sources: [{ zone: internal }]     # quién puede alcanzarlo (lista de tokens de fuente)
```

Parámetro	Tipo	Requerido	Por Defecto	Descripción
`id`	String	Sí	-	Identificador estable para el flujo. Aparece en el comentario de cada regla generada como `ans:<dst>:<id>` para que una regla activa pueda ser rastreada de vuelta al modelo.
`dst`	String	Sí	-	El grupo de inventario en el que se aplica la regla, o los pseudo-destinos `all` (cada host) y `five_g_sbi` (los grupos de funciones de red 5G). Un flujo cuyo grupo de destino no está desplegado en un sitio no produce reglas ni filas en la matriz allí.
`service`	String	Sí*	-	Un servicio nombrado declarado por el rol de destino (ver Catálogo de Servicios). Los puertos se buscan en la declaración del rol, por lo que el flujo nunca vuelve a declarar un número de puerto.
`proto_ports`	Lista	Sí*	-	La alternativa literal a `service`, para flujos no vinculados a un solo rol

Parámetro	Tipo	Requerido	Por Defecto	Descripción
				(por ejemplo, flujos de `all`-host). Cada entrada es `PORT/proto` (`tcp`, `udp`, `sctp`; el puerto puede ser un rango `LO-HI`) o un protocolo desnudo (`icmp`, `esp`, `proto 89`).
`purpose`	String	Sí	-	Descripción humana del flujo. Mostrada en la columna `service` de la matriz de puertos.
`sources`	Lista	Sí	-	Las fuentes permitidas, como una lista de tokens de fuente. Ver Tipos de Fuentes.

* Un flujo proporciona **ya sea** `service` **o** `proto_ports`, no ambos.

Tipos de Fuentes

Cada entrada en la lista `sources` de un flujo es un token de una sola clave que se resuelve contra la configuración y el inventario del sitio. Un token que no se resuelve a nada (por ejemplo, una lista de acceso vacía) no contribuye a ninguna regla: nunca se emite como un marcador de posición.

Token	Se Resuelve a
`{ zone: <name> }`	Los CIDRs de esa zona de plano (`firewall.zones.<name>`).
`{ list: <name> }`	Una lista de acceso de operador, por ejemplo `allowed_oam_subnets`.
`{ group: <name> }`	Las direcciones de host del grupo de inventario, más los pares entre sitios de ese rol. Ver Pares entre Sitios.
`{ ue_pools: true }`	Los grupos de UE / suscriptores del sitio (`firewall.ue_pools`).
`{ dra_peers: true }`	Cada dirección configurada en la propia lista de pares de Diameter del host DRA.
`{ net_devices: true }`	Cada dirección de dispositivo de red gestionado (`all.vars.<vendor>.devices`, por ejemplo `mikrotik.devices`).
`{ roaming: true }`	Rangos de roaming (IR.21 / GRX-IPX). Actualmente cualquier fuente: ver Precauciones Operativas.
`{ literal: "..." }`	Una fuente de texto libre que no se deriva del inventario.

Catálogo de Servicios

Los números de puerto son **propiedad del rol que los abre**, no del modelo central. Cada rol que expone puertos envía un `firewall_services.yml` en su directorio de rol, claveado por los grupos de inventario que sirve (un rol como `cscf` sirve a varios grupos, por lo que declara cada uno). Esto mantiene la definición del puerto junto al código que lo abre:

```yaml
# roles/omnihss/firewall_services.yml
firewall_services:
  hss:
    diameter: [3868/tcp, 3868/sctp]   # S6a / Cx-Dx / Sh
    api_oam:  8080/tcp                # API HSS (OAM)
    api_nf:   8443/tcp                # API HSS (consultada por
otros NFs)
    postgres: 5432/tcp                # replicación lógica entre
pares HSS
```

El resolutor agrupa cada `firewall_services.yml` de rol en un solo catálogo. El `service:` de un flujo se busca en el catálogo para el grupo `dst` de ese flujo, por lo que el modelo expresa solo la **relación** (quién puede alcanzar el servicio `diameter` de `hss`) mientras que el **puerto** vive con el rol. Agrega o mueve un listener en un rol y actualiza su `firewall_services.yml`; nada en el modelo central cambia.

Esta división tiene una red de seguridad que un modelo solo central carece: un **chequeo de deriva** compara lo que declaran los roles con lo que realmente están escuchando los hosts.

```
ansible-playbook -i hosts/<Deployment>/host_files/<Site>.yml \
    util_playbooks/firewall_services_audit.yml
```

Para cada host informa:

- **no declarado** - un listener activo en un puerto que ningún rol declaró (ya sea añádelo al `firewall_services.yml` del rol, o es un listener inesperado), y
- **no escuchando** - un servicio declarado que no está realmente activo.

Los hosts cuyos grupos no declaran servicios se omiten, por lo que la verificación es útil mientras los roles se migran al catálogo de manera incremental en lugar de todo a la vez.

Listeners condicionales. Algunos puertos se abren solo cuando se despliega otro rol: el MME abre sus puertos LCS (SBc-AP `29168/sctp`, SLs `9082/sctp`) solo cuando hay un host `omnilcs` en el inventario. Una declaración estática no

puede expresar esa condición, por lo que tales puertos se anotan en el `firewall_services.yml` del rol pero no se declaran. En un sitio que sí ejecuta el rol dependiente, la verificación de deriva los informa como no declarados, lo cual es el aviso correcto para revisarlos y permitirlos.

Configuración del Sitio

Cada sitio declara un bloque `firewall` bajo `all.vars` en su inventario. No hay retrocesos de dirección: un sitio debe describir su propia red. Cualquier cosa que un sitio no declare simplemente está vacía, y ninguna dirección de otro entorno se hereda nunca.

El ejemplo a continuación está completamente poblado para que cada campo muestre un valor representativo. Las direcciones utilizan rangos de documentación y un diseño coherente por plano (cada plano es su propia subred dentro de la superred `internal`), que es lo que hace que las zonas sean útiles: permiten que un flujo permita, digamos, solo el plano IMS en lugar de todo el núcleo.

```yaml
firewall:
  allowed_oam_subnets:           ['10.8.0.0/24']       # fuente de
gestión (SSH, UI, scrape, APIs de gestión)
  allowed_ran_subnets:           ['10.20.0.0/16']      # fuente de
señalización RAN (S1AP, NGAP, GTP-U, Abis)
  allowed_ue_voice_subnets:      ['100.64.0.0/16']     # UE
externo SIP en el P-CSCF
  allowed_carrier_voice_subnets: ['203.0.113.0/24']    # SIP de
interconexión de operadores
  zones:                                               # zonas de
plano: subred(es) real(es) de cada plano
    internal:    ['10.8.0.0/16']                       #
superred de núcleo confiable
    mobile_core: ['10.8.1.0/24']                       #   NFs de
núcleo móvil (EPC + 5GC + núcleo CS)
    core_svc:    ['10.8.2.0/24']                       #   plano
de servicios centrales (HSS, DRA, OCS, OAM)
    ims:         ['10.8.3.0/24']                       #   plano
IMS (CSCF, TAS)
    ue_svc:      ['10.8.4.0/24']                       #   plano
alcanzable por UE (P-CSCF, DNS, ePDG)
    core_v6:     ['2001:db8:0:1::/64']                 #
direcciones IPv6 de NFs centrales (no UEs)
  ue_pools: ['100.64.0.0/16', '2001:db8:64::/48']      # rangos de
IP de UE / suscriptor (v4 y v6)
  extra_rules:                                         # reglas de
aceptación manual (ver más abajo)
    - { proto: tcp, dport: '8443', source: '203.0.113.10/32',
comment: 'API de socio' }
```

Solo `allowed_oam_subnets` es obligatorio. Cada otra lista puede dejarse vacía (`[]`) en un sitio que carezca de esa función: un sitio sin RAN establece `allowed_ran_subnets: []`, un sitio sin núcleo IPv6 establece `core_v6: []`, y un sitio sin grupos de UE establece `ue_pools: []`. Una lista vacía simplemente no produce reglas para los flujos que la utilizan (ver la columna `Requerido` a continuación).

Parámetro	Tipo	Requerido	Por Defecto	Descr
`allowed_oam_subnets`	Lista	Sí	-	Subred fuente gestión Gobier SSH, la interfa usuario OmniW recopil de Promet las API gestión está va genera detiene un erro porque valor v dejaría gestión fuente permit causar bloque cumpli
`allowed_ran_subnets`	Lista	No	`[]`	Subred fuente señaliz RAN (S NGAP, Abis). I estar v

Parámetro	Tipo	Requerido	Por Defecto	Descr
				un siti RAN.
`allowed_ue_voice_subnets`	Lista	No	`[]`	Subred fuente UE ext SIP en CSCF.
`allowed_carrier_voice_subnets`	Lista	No	`[]`	Subred fuente SIP de interco de operad
`zones`	Mapa	Sí	-	Zonas plano c mapea nombr plano a subred real(es Parám de Zon
`ue_pools`	Lista	No	`[]`	Rangos de UE suscrip Vacío s sitio nc ningun
`extra_rules`	Lista	No	`[]`	Reglas acepta

Parámetro	Tipo	Requerido	Por Defecto	Descr
				manua aplicac cada h Ver Paráme de Reg Manua

Parámetros de Zona

Cada zona mapea un nombre de plano a una lista de CIDRs. Establezca solo los planos que utiliza el sitio; deje el resto vacío.

Zona	Descripción
`internal`	Superred de núcleo confiable. Fuente de señalización interna más amplia.
`mobile_core`	Funciones de red de núcleo móvil a través de generaciones: EPC (MME, S/P-GW, UPF), 5GC (AMF, SMF y la malla SBI), y núcleo CS (MSC, SS7/SIGTRAN).
`ims`	Plano IMS (I-CSCF, S-CSCF, TAS).
`ue_svc`	Plano de servicios alcanzables por UE (P-CSCF, DNS, ePDG).
`core_svc`	Plano de servicios centrales (HSS, DRA, OCS, OAM).
`core_v6`	Direcciones de servicio central IPv6. Vacío si el sitio no tiene núcleo IPv6.

Parámetros de Regla Manual

`extra_rules` (global, todos los hosts) y las `firewall_extra_rules` por host añaden reglas de aceptación en bruto para casos que el modelo no cubre. Una `source` que contenga `:` se aplica en IPv6.

Parámetro	Tipo	Requerido	Por Defecto	Descripción
`proto`	String	No	-	Protocolo, por ejemplo `tcp`, `udp`, `sctp`.
`dport`	String	No	-	Puerto o rango de destino.
`source`	String	No	-	CIDR de fuente. Un `:` en el valor aplica la regla en IPv6.
`comment`	String	No	`override`	Etiqueta registrada en el comentario de la regla.

Modos de Operación

El modo de aplicación se controla mediante `firewall_mode`.

Modo	Comportamiento
`monitor`	Cada regla de aceptación se instala, la política `INPUT` permanece en `ACCEPT`, y el tráfico que pasa se registra (como `FW-AUDIT`) y luego se permite. No se bloquea nada. Utilice esto para validar el modelo contra el tráfico real antes de hacer cumplir.
`enforce`	El tráfico que pasa se registra (como `FW-DROP`) y luego se bloquea, y las políticas `INPUT` y `FORWARD` se establecen en `DROP`.

El bloqueo es optativo y a prueba de fallos: solo el valor exacto `enforce` bloquea el tráfico. Cualquier otro valor, incluyendo una variable no establecida o un error tipográfico, se trata como monitor, por lo que un host no puede comenzar a bloquear tráfico por accidente. Esto se mantiene en tres niveles: el valor predeterminado de la plataforma es `monitor`, los sitios declaran `firewall_mode: monitor` en su inventario, y el generador trata cualquier cosa que no sea `enforce` como monitor. Hacer cumplir requiere un cambio deliberado.

```
all:
  vars:
    firewall_mode: monitor   # cambiar a 'enforce' solo después de
que el registro de auditoría confirme la cobertura
```

`firewall_mode` puede establecerse por sitio bajo `all.vars`, o anularse por grupo o por host a través de la precedencia normal de variables.

Pares entre Sitios

Los roles que se federan entre sitios (CSCF, TAS, OmniMessage, replicación HSS, OCS) enumeran sus pares remotos en el archivo de pares del sitio (por ejemplo, `prod_master_peers.yml`). Esas direcciones remotas se incorporan en la resolución `{ group: <role> }`, por lo que un flujo que permite un rol cubre automáticamente tanto las instancias del mismo sitio como las de otro sitio

bajo una política idéntica. La misma resolución se utiliza para las reglas y para la matriz, por lo que los pares aparecen en ambas.

Resolución de Direcciones

Un host se resuelve a **cada dirección con la que se declara**, no solo a su primaria. Eso significa `ansible_host` más cada `secondary_ips.<name>.ip_address` en el inventario. Los nodos de datapath y señalización vinculan servicios a NICs secundarias: el UPF responde N3 GTP-U en su dirección `n3`, y el STP recibe y origina SIGTRAN en su dirección pública. Tanto las reglas como la matriz cubren todas esas direcciones, como fuentes y como destinos, por lo que un par que permite un nodo por la dirección que se le entrega está permitiendo la dirección que realmente utiliza el tráfico.

Un host que pertenece a **varios grupos de inventario** recibe la **unión** de los flujos y servicios declarados de cada grupo. Un nodo convergente expone todo para todos los roles que ejecuta, y su conjunto de listeners esperados (utilizado por la verificación de deriva) es la unión a través de sus grupos.

Debido a que la resolución utiliza cada dirección declarada en lugar de rastrear qué servicio vincula qué NIC, la matriz puede listar un servicio en una dirección que realmente no vincula (por ejemplo, N3 GTP-U mostrado en la dirección N6 del UPF). Esto es una sobre-aproximación deliberada: una matriz completa pero ligeramente amplia es segura para entregar a un firewall ascendente, mientras que omitir una dirección real de datapath o SIGTRAN rompe el tráfico en silencio. Si se necesita precisión por dirección más adelante, un servicio puede nombrar la NIC que vincula; hasta entonces, la resolución permanece basada en hechos y completa.

Matriz de Comunicaciones / Puertos

La matriz de puertos es el modelo de flujo resuelto contra el inventario de un sitio, aplanado a una fila por fuente, host de destino y único puerto/protocolo. Es la forma de revisar o entregar a un firewall ascendente. Genérelo con:

```
ansible-playbook -i hosts/<Deployment>/host_files/<Site>.yml \
    util_playbooks/render_port_matrix.yml
```

Esto escribe `generated/firewall-port-matrix-exploded.csv` (por sitio, no comprometido). Las columnas son:

Columna	Descripción
`id`	El identificador del flujo.
`service`	El propósito del flujo.
`source`	Un CIDR o dirección de fuente resuelta.
`destination_role`	El grupo de inventario de destino.
`destination_host`	El nombre del host de destino.
`destination_ip`	La dirección del host de destino.
`port`	El puerto o rango de destino (vacío para entradas solo de protocolo como ICMP o ESP).
`protocol`	El protocolo.

Una fila aparece solo cuando su destino está desplegado en el sitio y su fuente se resuelve a una dirección real. Los roles no desplegados y las fuentes no resueltas no producen filas.

Protección contra Bloqueos

La aplicación de reglas está protegida por un retroceso automático, controlado por `firewall_rollback_guard` (predeterminado en on) y `firewall_rollback_timeout`:



1. Se captura el conjunto de reglas activo.
2. Se arma un temporizador en el host que restaurará la captura. Sobrevive a la sesión de gestión que se corta.
3. Se aplican las nuevas reglas.
4. El controlador restablece una nueva conexión y ejecuta una verificación trivial. Si las nuevas reglas lo bloquearon, esto falla y la ejecución se detiene para ese host.
5. Si se confirma la conectividad, el temporizador se desarma. Si no, el temporizador se activa y restaura el conjunto de reglas de trabajo sin acción del operador.

Parámetro	Tipo	Requerido	Por Defecto	Desc
`firewall_rollback_guard`	Booleano	No	`true`	Habilita captura tempor verifica reconex alreded cada ap
`firewall_rollback_timeout`	Entero	No	`120`	Segund espera tempor antes d restaur automá la captu

IPv6 y Doble Pila

La familia de direcciones se infiere por CIDR: un `:` en el valor significa IPv6, y la regla se aplica en `ip6tables`. Una zona o lista que contenga solo CIDRs IPv4, por lo tanto, no produce reglas IPv6 para esa fuente. Los sitios sin núcleo IPv6 dejan las zonas y grupos IPv6 vacíos (por ejemplo, `core_v6: []`). Bajo cumplimiento, la política `INPUT` de IPv6 se establece en `DROP`, por lo que valide el tráfico IPv6 en modo monitor antes de hacer cumplir un host cuya única entrada IPv6 legítima llega a través de una fuente solo IPv4.

Registro

El tráfico auditado y descartado se registra y se transporta por la tubería estándar a Loki. Se utilizan tres prefijos:

Prefijo	Significado
`FW-FLOW` / `FW-FLOW6`	Una nueva conexión vista en modo monitor. Construye el mapa de flujo activo del tráfico realmente en uso.
`FW-AUDIT` / `FW-AUDIT6`	Tráfico que pasa en modo monitor: se bloquearía bajo cumplimiento, pero se permite aquí. Revise estos antes de hacer cumplir.
`FW-DROP` / `FW-DROP6`	Tráfico que realmente se bloqueó bajo cumplimiento.

Objetivo del Registro

Los registros se escriben con `NFLOG` y se leen por `ulogd2`, que el rol instala como un requisito obligatorio junto con `iptables-persistent`. NFLOG solo es legible mientras un consumidor está en ejecución: sin `ulogd2`, el kernel descarta cada registro, y el modo monitor no prueba nada mientras parece funcionar: las reglas están instaladas, los contadores aumentan y nunca se

produce un registro. Ese fallo es silencioso, por lo que el paquete es obligatorio en lugar de un esfuerzo adicional.

Ambos paquetes se transportan en el repositorio apt de Omnitouch (`ubuntu_packages.networking`), por lo que un host que no puede instalarlos tiene un repositorio roto y la ejecución se detiene allí.

La tubería del diario de Alloy etiqueta estos `unit="ulogd2.service"` y `service="ulogd"`. Tenga en cuenta que no hay una etiqueta `job="firewall"`: `job` lleva el nombre del inventario del host, por lo que seleccione en `unit` para abarcar cada host.

Ejemplos de consultas de Loki:

```
# Tráfico que sería bloqueado una vez que se apliquen estos hosts
{unit="ulogd2.service"} |= "FW-AUDIT"

# Tráfico que se está descartando bajo aplicación, por host
sum by (hostname) (count_over_time({unit="ulogd2.service"} |= "FW-
DROP" [5m]))

# Solo un host
{unit="ulogd2.service", hostname="omni-tampa-sbc01"} |= "FW-DROP"
```

En el host mismo:

```
journalctl -t ulogd -f | grep FW-AUDIT
```

Precauciones Operativas

- **Las fuentes roaming y literal son actualmente cualquier fuente.** Los flujos cuya fuente es `roaming` o `literal` se emiten sin restricción de fuente (`0.0.0.0/0`). Trátalos como abiertos hasta que los rangos de roaming se resuelvan en CIDRs reales.
- **ICMP es aceptado de cualquier fuente.** El conjunto de reglas generado permite ICMP incondicionalmente, lo cual es más amplio que el alcance de gestión mostrado para el flujo de ping.

- **No apliques un host de plano de usuario sin reglas de reenvío.** Bajo aplicación, la política `FORWARD` se establece en `DROP`, y no se generan reglas de reenvío. Aplicar en un UPF, PGW o SGW sin agregar reglas de reenvío rompería el plano de usuario. Valida primero.
- **La lista de gestión es imprecisa.** `allowed_oam_subnets` rige SSH, la interfaz de usuario, scrape, y las APIs de gestión juntas. Un sitio que scrapea desde una red diferente a la que administra necesita una entrada `extra_rules` para separarlas.

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. Se encuentra en `group_vars`.

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

Los roles resuelven este directorio en tiempo de ejecución como `group_vars_omnicore` - el directorio `group_vars/` que se encuentra adyacente al directorio de inventario (`{{ inventory_dir.rsplit('/', 1)[0] }}/group_vars/`). Dondequiera que una ruta de archivo a continuación haga referencia a `{{ group_vars_omnicore }}`, apunta a este directorio.

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 OmniTAS y Planes de Marcado)

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 (predeterminado)
    │   ├── mo_dialplan.xml         # Móvil Originado (saliente)
    │   ├── mt_dialplan.xml         # Móvil Terminado (entrante)
    │   └── emergency.xml
    └── dialplan_lab/               # Planes de marcado de
laboratorio
        └── mo_dialplan.xml
```

Referencia en el Archivo de Hosts

```
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 para usar en este host
      dialplan_folder: "dialplan"         # Opcional -
predeterminado a "dialplan" si no se establece
```

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/` (o la carpeta especificada por `dialplan_folder`) 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"`
- `dialplan_folder: "dialplan_lab"`
- `dialplan_folder: "dialplan_prod"`

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

---

Ejemplo 3: Plantilla de Configuración Personalizada (OmniHSS)

El Servidor de Suscriptores en Casa acepta plantillas de configuración en 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

```yaml
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 en 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 5: Archivos de Configuración Requeridos (OmniCRM)

OmniCRM se obtiene **exclusivamente** de group_vars - no hay un valor predeterminado de rol al que recurrir. La configuración de la API no está intencionalmente integrada en el rol, por lo que un archivo faltante falla la ejecución de manera ruidosa en lugar de enviar en silencio los secretos de otro sitio.

Estructura de Archivos

```
hosts/Customer/
└── group_vars/
    ├── crm_config.yaml          # API + ui: configuración (requerido)
    ├── deployment_assets/       # Logotipos/iconos opcionales → /etc/omnicrm/deployment_assets
    └── omnicrm_plays/           # Juegos de aprovisionamiento de sitios opcionales, copiados sobre el checkout
```

Referencia en el Archivo de Hosts

```
crm:
  hosts:
    customer-crm01:
      ansible_host: 10.10.3.70
      gateway: 10.10.3.1
      host_vm_network: "vmbr3"
      ssl_fqdn: portal.example.com
```

Qué sucede:

1. Envía el checkout de OmniCRMv2 a `/opt/omnicrm-v2`, construye su venv y crea el rol y la base de datos de PostgreSQL.
2. Plantilla `{{ group_vars_omnicore }}/crm_config.yaml` a `/etc/omnicrm/crm_config.yaml` (**requerido**).
3. Sincroniza `deployment_assets/` a `/etc/omnicrm/deployment_assets/`.
4. Migra, construye la UI en el host y comienza las unidades `omnicrm-v2-*` detrás de nginx.

Importante: `crm_config.yaml` debe estar presente e incluir un bloque `ui:` con `ui.assets_dir: '/etc/omnicrm/deployment_assets'`.

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

---

Ejemplo 6: Directorio de Datos de Aprovisionamiento (OmniHSS)

El servicio de aprovisionamiento de OmniHSS (`services/provision_omnihss.yml`) lee datos de suscriptores y perfiles de un directorio dedicado `omnihss/` dentro de group_vars.

Estructura de Archivos

```
hosts/Customer/
└── group_vars/
    └── omnihss/
        ├── subscribers.csv             # Suscriptores a
aprovisionar
        ├── apn_identifiers.json
        ├── apn_qos_profiles.json
        ├── apn_profiles.json
        ├── eir_rules.json
        ├── ims_profiles.json
        ├── epc_profiles.json
        ├── roaming_rules.json
        ├── roaming_profiles.json
        ├── flow_information_rules.json
        ├── charging_rules.json
        └── pcrf_profiles.json
```

Qué sucede:

1. El servicio lee `{{ group_vars_omnicore }}/omnihss/subscribers.csv` para la lista de suscriptores a aprovisionar.
2. Lee los archivos JSON acompañantes del mismo directorio `omnihss/` (`apn_identifiers.json`, `apn_qos_profiles.json`, `apn_profiles.json`, `eir_rules.json`, `ims_profiles.json`, `epc_profiles.json`, `roaming_rules.json`, `roaming_profiles.json`, `flow_information_rules.json`, `charging_rules.json`, `pcrf_profiles.json`) para definir los datos de APN, QoS, IMS, EPC, roaming, flujo, carga y PCRF aplicados a esos suscriptores.

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

---

Ejemplo 7: Credenciales SMTP de Grafana (Monitoreo)

La plantilla `grafana.ini` del rol de monitoreo conecta la configuración de correo electrónico de alerta de Grafana (SMTP) desde group_vars. El usuario y la contraseña **no tienen un valor predeterminado de rol** - están intencionalmente no integrados, por lo que un valor faltante falla la ejecución de manera ruidosa en lugar de enviar credenciales.

Referencia en el Archivo de Hosts

```
monitoring:
  hosts:
    customer-oam01:
      ansible_host: 10.10.3.135
      gateway: 10.10.3.1
      host_vm_network: "vmbr3"
      grafana_smtp_user: "smtp-user"          # REQUERIDO - sin
predeterminado
      grafana_smtp_password: "smtp-password"  # REQUERIDO - sin
predeterminado
      grafana_smtp_host: "in-v3.mailjet.com:587"  # Opcional -
predeterminado a in-v3.mailjet.com:587
```

Qué sucede:

1. `grafana_smtp_user` y `grafana_smtp_password` se plantillan en la sección `[smtp]` de `/etc/grafana/grafana.ini`. Ambos son requeridos - no hay respaldo.
2. `grafana_smtp_host` es opcional y predeterminado a `in-v3.mailjet.com:587` si no se establece.

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

---

Ejemplo 8: Claves HNET para la De-concealment de SUCI de UDM

El OmniUDM realiza la de-concealment de SUCI (TS 33.501 §6.12): recupera el SUPI de la SUCI oculta que un UE presenta en el registro. Esto requiere las claves **privadas** de la Red de Hogar que emparejan con las claves **públicas** de la Red de Hogar aprovisionadas en los USIMs. Estas claves privadas viven en un directorio dedicado `hnet_udm/` dentro de group_vars y se despliegan en `/etc/omniudm/hnet/` en el host UDM por el rol `5gc_nf`.

Estructura de Archivos

```
hosts/Customer/
└── group_vars/
    └── hnet_udm/
        ├── curve25519-1.key       # Clave privada X25519, PEM (ID
de Clave Pública de la Red de Hogar 1)
        ├── curve25519-1.pub       # Valor público bruto de 32
bytes (hex) - para aprovisionamiento de USIM
        ├── secp256r1-2.key        # Clave privada EC
(prime256v1), PEM (ID de Clave Pública HNET 2)
        └── ...
```

Nombramiento de archivos: `<curve>-<id>.key`, donde `<id>` es el Identificador de Clave Pública de la Red de Hogar que el UDM empareja contra la SUCI, y `<curve>` es `curve25519` (Perfil A, X25519) o `secp256r1` (Perfil B, prime256v1). El nombre del archivo es cómo la clave está vinculada a su ID de clave HNET. Debe coincidir con el ID programado en el USIM.

Qué sucede:

1. El rol asegura que `{{ group_vars_omnicore }}/hnet_udm/` exista en el controlador de Ansible (creado con `become: false`: las claves permanecen en el controlador y nunca se generan en la caja remota).
2. Escanea el directorio en busca de archivos `*.key`.
3. **Si no se encuentran, se genera automáticamente una clave predeterminada `curve25519-1.key` (X25519)** con `openssl genpkey`,

para que la de-concealment de SUCI funcione desde el principio. Esto es idempotente (`creates:` guardia): una clave existente nunca se sobrescribe.

4. Para cada `curve25519-*.key`, el valor público bruto de 32 bytes correspondiente se exporta a un archivo `.pub` (hex) para el aprovisionamiento de USIM. Los archivos `.pub` permanecen en el inventario y **no** se despliegan en el UDM.
5. Todos los archivos `*.key` se copian a `/etc/omniudm/hnet/` en el UDM (modo `0600`, propiedad de root), y se reinicia `omniudm`.

> **Importante:** Una clave generada automáticamente es aleatoria y solo de-concealará SUCIs de USIMs programados con su clave pública **coincidente**. Para un laboratorio donde controlas la programación de SIM, toma el `.pub` exportado y aprovisiónalo como la Clave Pública de la Red de Hogar (ID 1) en el USIM. Si tus SIMs utilizan el esquema nulo (sin ocultación), no se necesita ninguna clave HNET y este directorio puede permanecer vacío.

Para exportar una clave pública de una clave privada existente manualmente:

```
openssl pkey -in curve25519-1.key -pubout -outform DER | tail -c
32 | xxd -p -c 32
```

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

---

Ejemplo de Estructura de

Directorio 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
    ├── mmsc_runtime.exs        # Plantilla de OmniMessage MMSc
    ├── 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
    ├── omnihss/                # Datos de aprovisionamiento de
OmniHSS (carpeta)
    │   ├── subscribers.csv         # Suscriptores a aprovisionar
    │   ├── apn_identifiers.json
    │   ├── apn_qos_profiles.json
    │   ├── apn_profiles.json
    │   ├── eir_rules.json
    │   ├── ims_profiles.json
    │   ├── epc_profiles.json
```

```
    │   ├── roaming_rules.json
    │   ├── roaming_profiles.json
    │   ├── flow_information_rules.json
    │   ├── charging_rules.json
    │   └── pcrf_profiles.json
    └── hnet_udm/               # Claves de de-concealment de SUCI
de UDM (carpeta)
        ├── curve25519-1.key        # Clave privada X25519 (ID de
Clave Pública de la Red de Hogar 1)
        └── curve25519-1.pub        # Valor público bruto (hex)
para aprovisionamiento de USIM
```

Parámetros Comunes que Hacen

Referencia a group_vars

Parámetro	Componente	Referencias
`smsc_template_config`	omnimessage	Archivo de plantilla Jinj (por ejemplo, `smsc_controller.exs`)
`smsc_smpp_template_config`	omnimessage_smpp	Archivo de plantilla Jinj (por ejemplo, `smsc_smpp.exs`)
`mmsc_runtime_template`	mmsc	Archivo de plantilla Jinj (por ejemplo, `mmsc_runtime.exs`) - requerido
`gateways_folder`	applicationserver	Nombre de carpeta (pc ejemplo, `gateways_pro`
`dialplan_folder`	applicationserver	Nombre de carpeta (pc ejemplo, `dialplan`) - predeterminado a `dialplan` si no se establece
`tas_runtime_template`	applicationserver	Archivo de plantilla Jinj (por ejemplo, `tas_runtime.exs.j2`) predeterminado a `tas_runtime.exs.j2` s se establece
`hss_template_config`	omnihss	Archivo de plantilla Jinj (por ejemplo, `hss_runtime.exs.j2`)

Parámetro	Componente	Referencias
`mme_template_config`	omnimme	Archivo de plantilla Jinj (por ejemplo, `mme_runtime.exs.j2`)
`dra_template_config`	dra	Archivo de plantilla Jinj (por ejemplo, `dra_runtime.exs.j2`)
`pgwc_template_config`	pgwc	Archivo de plantilla Jinj (por ejemplo, `pgwc_runtime.exs.j2`)
`frr_template_config`	omniupf	Archivo de plantilla Jinj (por ejemplo, `frr.con`
Plantillas SS7	ss7 (varios roles)	Archivos de plantilla Jir (por ejemplo, `stp.j2`, `hlr.j2`, `camel.j2`)
`crm_config.yaml`	omnicrm	Archivo de configuraci API - **requerido**, obter solo de `{{ group_vars_omnicore }}/crm_config.yaml` (s valor predeterminado rol)
`grafana_smtp_user`	monitoring	Usuario SMTP de Grafa **requerido**, sin valor predeterminado de rol
`grafana_smtp_password`	monitoring	Contraseña SMTP de Grafana - **requerido**, s valor predeterminado

Parámetro	Componente	Referencias
`grafana_smtp_host`	monitoring	Host SMTP de Grafana opcional - predetermin a `in-v3.mailjet.com:`
Aprovisionamiento de OmniHSS	omnihss	Directorio `omnihss/` cc `subscribers.csv` + archivos JSON de perfil
Claves HNET	5gc_nf (omniudm)	Directorio `hnet_udm/` c claves privadas `<curve <id>.key` para de-concealment de SUCI - genera automáticamer una `curve25519-1.key` predeterminada si está vacío
`secrets.yml`	common/secrets (omnihss y sus clientes de API)	Archivo de Ansible Vau claves y contraseñas d API, gestionado con `scripts/vault_secret` - ver Secretos del Clier
Config YAMLs	Varios componentes	Archivos de configurac 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 se copian
5. **Control de versiones de todo** - Confirma todos los group_vars en 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 de clientes 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 de host en el archivo de hosts
- Cambios temporales - Edita en el objetivo, confirma en group_vars si es permanente

---

Documentación Relacionada

- Referencia de Configuración - Todos los parámetros de host y lo que hacen
- Configuración del Archivo de Hosts - Cómo estructurar archivos de hosts
- Secretos del Cliente - El archivo `secrets.yml` cifrado en vault y cómo gestionarlo

- **Configuración de OmniCall**: https://docs.omnitouch.com.au/docs/repos/OmniCall - Qué va en los archivos de configuración
- **Configuración de OmniCore**: https://docs.omnitouch.com.au/docs/repos/OmniCore - Detalles de configuración del componente

Configuración del Archivo de Hosts

Descripción General

El archivo de 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 de hosts se encuentran en el directorio de nivel superior `hosts/` (establecido como el `inventory` de Ansible en `ansible.cfg`), organizados por cliente:

```
hosts/
└── Customer_Name/
    └── host_files/
        ├── Production.yml
        ├── Staging.yml
        └── Customer_Lab.yml
```

Los nombres de archivo son arbitrarios. Hay un archivo por implementación, típicamente nombrado según la implementación o el sitio (por ejemplo, `Production.yml`, `Staging.yml`) en lugar de un esquema fijo `production.yml`/`staging.yml`/`lab.yml`.

Ejemplo de Estructura del Archivo de Hosts

Este ejemplo simplificado 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 de 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 Implementación de Proxmox VM/LXC 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 de Grupo de 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
```

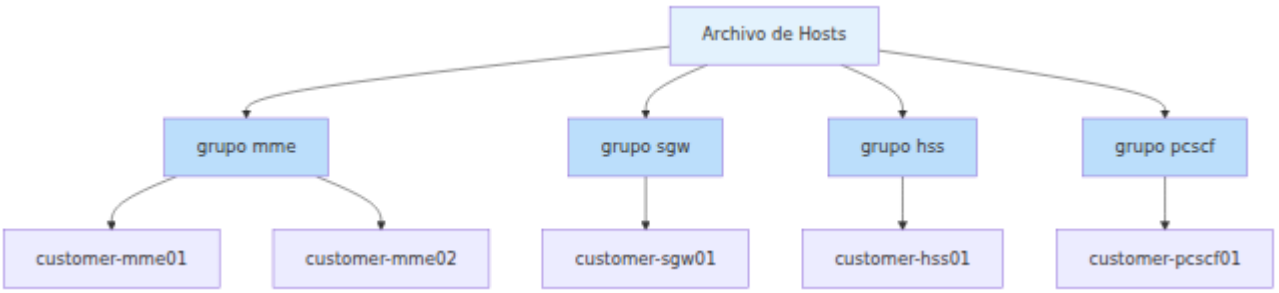
Sección de Variables Globales

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 Redes
    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:



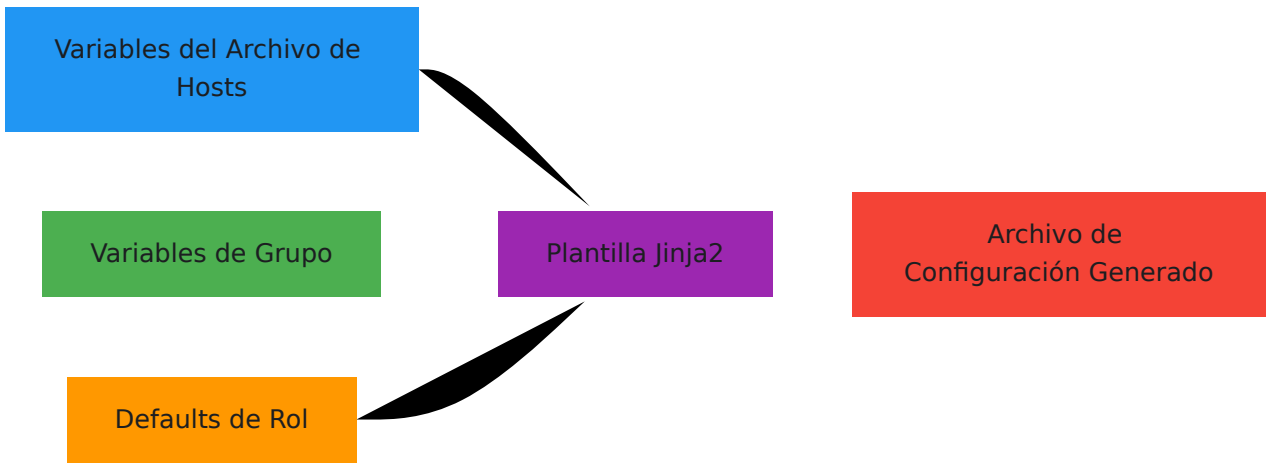
Cuando ejecutas un playbook dirigido a `mme`, se aplica a todos los hosts en la sección `mme:hosts:`.

El diagrama anterior muestra solo un puñado de grupos. Existen muchos más, incluidos `ocs`, `crm`, `smsc`, `omnimessage`, `mmsc`, `dra`, `xcap`, `siptrace`, `cbc`, y los grupos 5GC `amf`, `smf`, y `upf`, entre otros.

Configuración con Plantillas Jinja2

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

Cómo Funciona Jinja2



Ejemplo de Uso de Plantillas

El archivo de hosts define:

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

Plantilla Jinja2 (en rol):

```
# 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:

```
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 %}
```

Formato:

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

Sobrescribiendo Variables con group_vars

Mientras que el archivo de 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 del Archivo de Hosts

Este ejemplo más completo tiene datos sensibles oscurecidos:

```yaml
# Núcleo EPC
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"

# Núcleo IMS
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 Redes
    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 — consulta docs/firewall.md para el
conjunto completo de parámetros
    firewall:
      allowed_oam_subnets:
        - '10.0.0.0/8'
        - '192.168.0.0/16'
      allowed_ue_voice_subnets:
        - '10.0.0.0/8'
      zones:
        internal: ['10.0.0.0/8']

    # Configuración del Hipervisor (ejemplo de VM Proxmox)
    proxmoxServers:
      customer-prxmx01:
        # deployment_type omitido → se establece de forma
predeterminada en "vm"
        proxmoxServerAddress: 10.10.0.100
        proxmoxServerPort: 8006
        proxmoxApiTokenName: Customer
        proxmoxApiTokenSecret: "token-secret"
        proxmoxNodeName: pve01
        proxmoxTemplateName: ubuntu-24.04-cloud-init-template
        proxmoxTemplateId: 9000
        proxmoxTemplateUser: omnitouch        # nombre de usuario
opcional de cloud-init; se establece de forma predeterminada en la
primera clave local_users
        proxmoxTemplatePassword: omnitouch    # contraseña
opcional de cloud-init; se establece de forma predeterminada en el
nombre de usuario de cloud-init

    # Para un sitio LXC, establece deployment_type: lxc y
    # proxmoxLxcOsTemplate en cada entrada de proxmoxServers en su
lugar.
```

Consulta Implementación de Proxmox VM/LXC para obtener detalles completos sobre la configuración y configuración 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 Local
- **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 a la Implementación de Ansible - Proceso de implementación general
- Referencia de Configuración - **Guía completa de todas las variables de configuración**
- Configuración de Variables de Grupo - Sobrescribiendo configuraciones predeterminadas
- Estándar de Planificación de IP - **Arquitectura de red y pautas 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 Implementación - Vista completa del sistema

Próximos Pasos

1. Crea tu archivo de 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

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 iniciarse 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: "Nombre del Cliente"
    license_server_api_urls: ["https://10.10.2.150:8443/api"]
```

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
```

Los directorios de inventario reales están en mayúsculas y tienen un prefijo (por ejemplo, `hosts/Omnicore_Customer/`), y el archivo de hosts se nombra según el sitio (por ejemplo, `hosts/Omnicore_Customer/host_files/Production.yml`).

Para verificar el estado de todas las licencias, navegue a https://license_server .

Eliminando una Licencia

Para eliminar una licencia, use `util_playbooks/delete_license.yml`.

Requisitos de Red

Configuración del Cortafuegos

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

Nombre de Host	Propósito
1.time.omnitouch.com.au	Servidor de validación de licencias 1
2.time.omnitouch.com.au	Servidor de validación de licencias 2
3.time.omnitouch.com.au	Servidor de validación de licencias 3

Reglas de salida requeridas:

- Protocolo: HTTPS (TCP/443)
- Destino: los nombres de host anteriores, resueltos a través de DNS
- Dirección: Salida

> **Resolver por DNS: no codificar IPs.** Siempre permita estos hosts por FQDN (o por las direcciones a las que su cortafuegos los resuelve dinámicamente). Las IP subyacentes pueden cambiar sin previo aviso. No

> se admiten listas de permitidos de IP estáticas y eventualmente romperán la validación de licencias.

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 - admite 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.

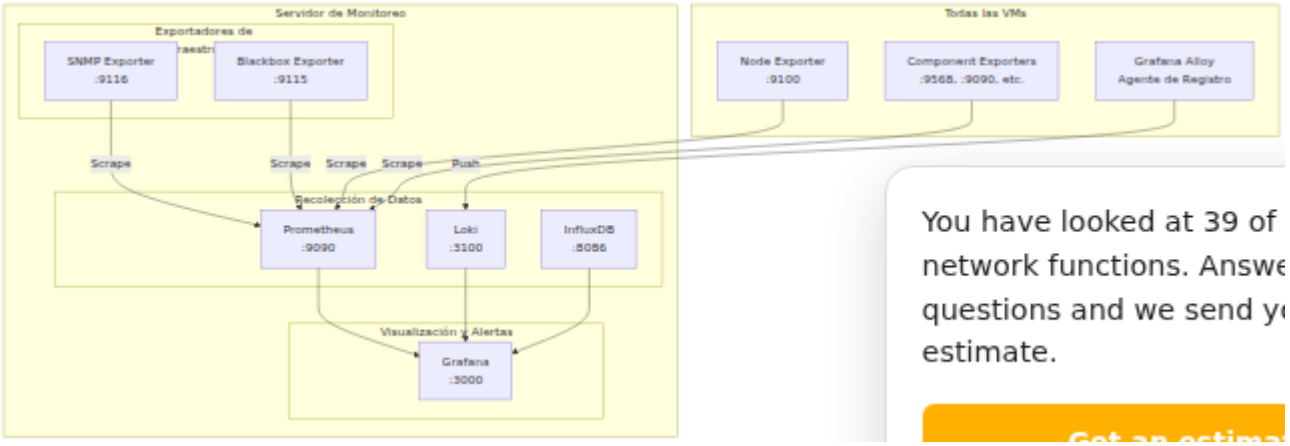
Documentación Relacionada

- Referencia de Configuración
- Configuración del Archivo de Hosts

Monitoreo y Observabilidad

Descripción general

OmniCore incluye un conjunto integral de monitoreo y observabilidad que proporciona recolección de métricas, agregación de registros, visualización y alertas. El sistema se despliega automáticamente mediante el rol de monitoreo.



Componentes

Componente	Propósito	Puerto	Retención de Datos
Prometheus	Almacenamiento de métricas de series temporales	9090	30 días o 5GB
Loki	Agregación de registros	3100	7 días o 50GB
InfluxDB	Métricas de RAN (Nokia)	8086	30 días
Grafana	Visualización y alertas	3000	N/A
Node Exporter	Métricas del host	9100	N/A
SNMP Exporter	Métricas de dispositivos de red	9116	N/A
Blackbox Exporter	Sondeos ICMP/HTTP	9115	N/A

Fuentes de Datos

Prometheus

Prometheus recopila métricas de todos los componentes de OmniCore cada 1 minuto.

Datasource UID: `omnicore_prometheus`

Objetivos Recopilados

Nombre del Trabajo	Grupo de Inventario	Puerto	Métricas
`Node Exporter`	`all`	9100	CPU del host, memoria, disco, red
`MMEs`	`mme`	9568	Conteos de sesiones, tasas de adjunto/desadjunto
`HSS`	`hss`	9568	Solicitudes de autenticación, búsquedas de suscriptores
`SGW-C`	`sgw`	9568	Conteos de portadores, mensajes GTP-C
`PGW-C`	`pgwc`	9090	Conexiones PDN, asignaciones de IP
`UPF Standalone`	`upf`	9090	Conteos de paquetes, rendimiento
`CSCF`	`pcscf`, `scscf`, `icscf`	9090	Registros, transacciones
`ApplicationServer FreeSWITCH`	`applicationserver`	9090	Conteos de llamadas, uso de canales
`DRA`	`dra`	9568	Decisiones de enrutamiento, estado de pares

Nombre del Trabajo	Grupo de Inventario	Puerto	Métricas
`OmniMessage`	`omnimessage`	9568	Entrega de SMS, sesiones SMPP
`OmniMessage MMSc`	`mmsc`	9568	Solicitudes MM1, notificaciones MMS, reenvíos y entradas MM4, decisiones en/red fuera
`OmniSS7`	`omniss7`	8080	Métricas de puerta de enlace SS7/SIGTRAN
`KeyDB`	`ocs`	9121	Uso de memoria, operaciones/segundo
`CGrateS`	`ocs`	2080	Calificación, cobro, estadísticas de CDR

Exportadores de Infraestructura

Exportador	Objetivos	Variable de Configuración
SNMP (MikroTik)	Routers/conmutadores	`mikrotik.hosts`
SNMP (iDRAC)	Servidores Dell	`idrac.hosts`
SNMP (Synology)	Dispositivos NAS	`synology.hosts`
SNMP (SAF)	Enlaces de microondas	`saf.hosts`
SNMP (Cisco)	Conmutadores	`cisco.hosts`
Blackbox ICMP	Todos los hosts + 8.8.8.8	Automático
VMware	Clústeres de vCenter	`vcenter_ip`, `vcenter_password`
Proxmox	Clústeres PVE	`proxmoxServers`

Loki

Loki recibe registros de los agentes de Grafana Alloy que se ejecutan en todas las VMs.

Datasource UID: `omnicore_loki`

Consulte Registro Centralizado para la configuración detallada de Loki/Alloy.

Etiquetas de Registro

Etiqueta	Descripción	Ejemplo
`hostname`	Nombre del host de la VM fuente	`customer-mme01`
`component`	Tipo de componente	`mme`, `cscf`, `ocs`
`unit`	Nombre de la unidad systemd	`omnimme.service`
`level`	Severidad del registro	`info`, `error`

InfluxDB

InfluxDB almacena datos de series temporales para casos de uso específicos que requieren una retención más prolongada o diferentes patrones de consulta.

Datasource UID: `omnicore_influxdb`

Bases de Datos

Base de Datos	Propósito	Retención
`nokia-monitor`	Métricas de rendimiento de RAN de Nokia	30 días
`dra`	Estadísticas de enrutamiento de DRA	365 días
`Omnicharge_TAP3`	Métricas de archivos TAP de roaming	90 días

Tableros de Grafana

Organización de Tableros

Los tableros están organizados en carpetas por área funcional:

Carpeta	Contenidos
BSS	CGrateS, KeyDB, tableros de cobro
EPC	Tableros de MME, SGW, PGW, UPF, HSS, DRA
IMS	Tableros de CSCF, Servidor de Aplicaciones, OmniMessage
Infraestructura	Node Exporter, SNMP, monitoreo de red
RAN	Tableros de rendimiento de Nokia/OmniRAN
Registros	Visores de registros específicos de componentes

Aprovisionamiento de Tableros (Archivos en Disco)

El repositorio es la única fuente de tableros. Ansible copia los tableros del rol y los `group_vars/grafana/dashboards` del cliente a `/var/lib/grafana/omnicore-dashboards/` en el host de Grafana. Un archivo de proveedor en `/etc/grafana/provisioning/dashboards/` hace que Grafana los cargue.

- Los tableros son de solo lectura en la interfaz de Grafana.
- Un archivo cambiado actualiza el tablero dentro de los 30 segundos.
- Un archivo eliminado elimina el tablero.
- Cada directorio se convierte en una carpeta con el mismo nombre.
- Después de la recarga, el playbook elimina cualquier tablero que no provenga del disco.
- Un archivo de cliente reemplaza un archivo de rol en la misma ruta.

Archivos Fuente de Tableros

Los archivos JSON de los tableros se almacenan en el repositorio de Ansible:

```
roles/monitoring/templates/grafana/dashboards/
├── BSS/
│   ├── KeyDB_Cluster.json
│   ├── cgrates_mysql.json
│   └── cgrates_stats.json
├── EPC/
│   ├── OmniMME.json
│   ├── OmniHSS.json
│   ├── OmniDRA.json
│   ├── OmniPGWC.json
│   ├── OmniUPF.json
│   └── ...
├── IMS/
│   ├── IMS_Status.json
│   ├── OmniMessage_(SMS-C).json
│   └── ...
├── Infraestructura/
│   ├── Node_Exporter_Full.json
│   ├── MikroTik_Dashboard.json
│   └── ...
├── RAN/
│   ├── Nokia_Overview.json
│   ├── Nokia_Detailed.json
│   └── ...
└── Logs/
    └── Logs.json
```

Respaldo de Tableros

Exportar tableros y alertas desde una instancia de Grafana en ejecución al repositorio para control de versiones:

```
# Desde el directorio roles/monitoring
python3 export_grafana.py --customer <CUSTOMER>
```

Esto exporta:

- Todos los tableros a `roles/monitoring/templates/grafana/dashboards/{folder}/*.json`

(compartido entre clientes)

- Reglas de alerta a `hosts/Omnicore_{CUSTOMER}/group_vars/grafana/alerts/all_alert_rules.json`
- Puntos de contacto a `hosts/Omnicore_{CUSTOMER}/group_vars/grafana/alerts/contact_points.json`
- Políticas de notificación a `hosts/Omnicore_{CUSTOMER}/group_vars/grafana/alerts/notification_policies.json`



Comportamiento de Exportación

- Solo escribe archivos que han cambiado (ignorando campos volátiles como `version` e `id`)
- Preserva la estructura de carpetas que coincide con la organización de Grafana
- Mapea títulos de tableros a nombres de archivos consistentes

Tableros Personalizados

Los tableros específicos del cliente se pueden colocar en `group_vars/grafana/dashboards/` y se cargarán junto con los tableros estándar:

```
hosts/customer/group_vars/
└── grafana/
    └── dashboards/
        ├── Custom_Dashboard_1.json
        └── Custom_Dashboard_2.json
```

Alertas

Arquitectura



Carga de Alertas

Las reglas de alerta provienen de archivos. El repositorio es la única fuente de reglas.

1. `load_grafana_from_export.py` fusiona las reglas del rol, las reglas del cliente y las sobreescrituras del sitio. Escribe un archivo de aprovisionamiento de Grafana.
2. Ansible copia ese archivo a `/etc/grafana/provisioning/alerting/omnicore-rules.yaml` y recarga las alertas.
3. Las reglas son de solo lectura en la interfaz de Grafana.

Eliminaciones: el cargador elimina cada regla en el host que no está aprovisionada por archivos. Enumera cada regla de archivo que ha desaparecido del disco bajo `deleteRules` en el nuevo archivo. Por lo tanto, eliminar un archivo de regla elimina la regla en la siguiente ejecución.

Los puntos de contacto y las políticas de notificación aún se cargan a través de la API desde `hosts/Omnicore_<CUSTOMER>/group_vars/grafana/alerts/`.

Reglas de Alerta Predeterminadas

Alerta	Carpeta	Condición	Por Duración
Espacio en Disco 90% Usado	Infraestructura	Sistema de archivos raíz > 90%	5 minutos
CPU Superior al 90%	Infraestructura	Uso de CPU > 90%	5 minutos
Carga del Sistema sobre 90%	Infraestructura	Promedio de carga > 90%	5 minutos
Host Caído	Infraestructura	Objetivo de Prometheus inalcanzable	5 minutos
Subs de MME Caídos 40%	EPC	Conteo de sesiones cae un 40%	30 segundos
0 Subs en MME	EPC	No hay suscriptores adjuntos	5 minutos
Subs de IMS caídos 40%	IMS	Registros de P-CSCF caen un 40%	30 segundos
MOS Promedio por debajo de 4	IMS	Calidad de voz degradada	5 minutos
Conteo de eNodeB Caído	RAN	eNBs conectados disminuidos	1 minuto
Retraso de Respuesta de Diámetro Alto	EPC	Pico de latencia P95	5 minutos

Estructura de Reglas de Alerta

Reglas de alerta en `alerts.yaml`:

```yaml
apiVersion: 1
groups:
  - orgId: 1
    name: Grupo de Alertas
    folder: Infraestructura
    interval: 1m
    rules:
      - uid: alert-lowDiskSpace
        title: Espacio en Disco 90% Usado en el host
        condition: C
        data:
          - refId: A
            datasourceUid: omnicore_prometheus
            model:
              expr: |
                100 - ((node_filesystem_avail_bytes{job="Node
Exporter",mountpoint="/"} * 100)
                / node_filesystem_size_bytes{job="Node
Exporter",mountpoint="/"})
          # ... expresiones de umbral
        noDataState: OK
        execErrState: Error
        for: 5m
        annotations:
          summary: El espacio en disco superó el 90% en el host
        labels:
          type: resource
```

Configuración de Puntos de Contacto

Los puntos de contacto se configuran a través de variables de inventario:

```yaml
# En group_vars/all.yml o vars específicas del cliente
monitoring_data:
  contactPoints:
    - orgId: 1
      name: default
      receivers:
        # Integración de Slack
        - uid: slack-alerts
          type: slack
          disableResolveMessage: false
          settings:
            url: "https://hooks.slack.com/services/xxx/yyy/zzz"

        # Integración de Google Chat
        - uid: gchat-alerts
          type: googlechat
          disableResolveMessage: false
          settings:
            url:
"https://chat.googleapis.com/v1/spaces/xxx/messages?key=yyy"
```

Los tipos de puntos de contacto admitidos incluyen Slack, Google Chat, Correo Electrónico y Pushover (ver `roles/monitoring/templates/grafana/contact-points.yaml.j2`). Pushover se configura a través de un bloque `notifications.pushover` (con `enabled`, `api_token` y `user_key`).

Políticas de Notificación

Las políticas de notificación dirigen alertas a los puntos de contacto apropiados:

```yaml
# templates/grafana/notification-policies.yaml.j2
apiVersion: 1

policies:
  - orgId: 1
    receiver: default
    group_by: ['grafana_folder', 'alertname']
    group_wait: 30s
    group_interval: 5m
    repeat_interval: 4h
```

Referencia de Configuración

Configuración de Prometheus

Ubicada en `/etc/prometheus/prometheus.yml` en el servidor de monitoreo.

Parámetro	Predeterminado	Descripción
`scrape_interval`	1m	Con qué frecuencia raspar objetivos
`evaluation_interval`	1m	Con qué frecuencia evaluar reglas de grabación
`scrape_timeout`	50s	Tiempo de espera para solicitudes de raspado

Configuración de Grafana

Ubicada en `/etc/grafana/grafana.ini` en el servidor de monitoreo.

Configuraciones clave configuradas por el rol de monitoreo:

Configuración	Valor	Descripción
`allow_embedding`	true	Habilitar incrustaciones en iframes
`provisioning`	`/etc/grafana/provisioning`	Directorio de aprovisionamiento

La contraseña de administrador no se establece en `grafana.ini` (la línea `admin_password` se deja comentada). En su lugar, se aplica a través de la variable `grafana_admin_password`, que se pasa a las tareas del cargador de API (cargadores de tableros/alertas) durante el despliegue.

Alertas por Correo Electrónico (SMTP)

Las alertas por correo electrónico requieren que se configuren credenciales SMTP en `group_vars` (ver `roles/monitoring/templates/grafana/grafana.ini`):

Variable	Requerido	Predeterminado	Descripci
`grafana_smtp_user`	Sí	*(ninguno)*	Nombre de usuario SMT Sin predetermin Si no se establece, e despliegue f ruidosament en lugar de enviar un secreto.
`grafana_smtp_password`	Sí	*(ninguno)*	Contraseña SMTP. Sin predetermin Si no se establece, e despliegue f
`grafana_smtp_host`	No	`in-v3.mailjet.com:587`	Host:puerto servidor SM
`grafana_smtp_enabled`	No	`true`	Si SMTP está habilitado.

La `from_address` y `from_name` se derivan de `customer_legal_name` (por ejemplo, `alerts+<customer_legal_name>@omnitouch.com.au` y `Omnitouch Alerts - <customer_legal_name>`).

Configuración de Loki

Consulte Registro Centralizado para la retención y configuración de almacenamiento de Loki.

Configuración de InfluxDB

Ubicada en `/etc/influxdb/influxdb.conf` en el servidor de monitoreo.

Base de Datos	Política de Retención	Duración
`nokia-monitor`	`autogen`	30 días
`dra`	`autogen`	365 días
`Omnicharge_TAP3`	`autogen`	90 días

Puertos de Servicio

Servicio	Puerto	Protocolo	Descripción
Grafana	3000	HTTP	Interfaz de Tablero y API
Prometheus	9090	HTTP	Almacenamiento y consulta de métricas
Loki	3100	HTTP	Ingesta y consulta de registros
InfluxDB	8086	HTTP	API de InfluxDB
Node Exporter	9100	HTTP	Métricas del host
SNMP Exporter	9116	HTTP	Proxy de métricas SNMP
Blackbox Exporter	9115	HTTP	Métricas de sondeo
Alloy	12345	HTTP	Interfaz y métricas de Alloy

Operaciones Comunes

Visualización de Métricas

1. Abra Grafana en `http://<monitoring-ip>:3000`
2. Navegue a **Tableros** y seleccione la carpeta apropiada
3. Use el selector de rango de tiempo para ajustar la ventana de vista

Búsqueda de Registros

1. Abra Grafana en `http://<monitoring-ip>:3000`
2. Vaya a **Explorar** y seleccione la fuente de datos **Loki**
3. Use consultas LogQL:

```
# Todos los registros de un host específico
{hostname="customer-mme01"}

# Errores de todos los componentes de MME
{component="mme"} |~ "(?i)error"

# Buscar un IMSI específico
{component="hss"} |= "123456789012345"
```

Creación de Alertas Personalizadas

Agregue un archivo de regla al directorio `group_vars/grafana/alerts/` del cliente, o al directorio `templates/grafana/rules/` del rol para una regla que todos los sitios obtienen. Ejecute el play de monitoreo. Una regla creada en la interfaz de Grafana se elimina en la siguiente ejecución.

Respaldo de Tableros

Después de realizar cambios en la interfaz de Grafana, exporte al repositorio:

```
cd roles/monitoring
python3 export_grafana.py --customer <CUSTOMER> --url
http://<monitoring-ip>:3000
git add ../../hosts/Omnicore_<CUSTOMER>/group_vars/grafana/
git commit -m "Actualizar alertas desde producción"
```

El argumento `--customer` es obligatorio. Este script exporta reglas de alerta, puntos de contacto y políticas de notificación al directorio `group_vars/grafana/alerts/` del cliente; ya no exporta tableros, que se comparten entre clientes en el directorio del rol (`roles/monitoring/templates/grafana/dashboards/`).

Solución de Problemas

Objetivo de Prometheus Caído

Síntomas: El tablero muestra "Sin Datos" o el estado del objetivo muestra CAÍDO

Causas posibles:

- Servicio no en ejecución en el host objetivo
- Firewall bloqueando el puerto del exportador
- Resolución de nombre de host incorrecta

Resolución:

1. Verifique si el servicio se está ejecutando en el objetivo:
   ```
   systemctl status <service>
   curl http://localhost:<port>/metrics
   ```
2. Verifique la conectividad desde el servidor de monitoreo:
   ```
   curl http://<target>:<port>/metrics
   ```

3. Verifique la página de objetivos de Prometheus: `http://<monitoring-ip>:9090/targets`

Tablero No Cargando

Síntomas: El tablero muestra un spinner de carga o error

Causas posibles:

- Conexión de fuente de datos fallida
- Tiempo de espera de consulta
- Corrupción del JSON del tablero

Resolución:

1. Pruebe la fuente de datos en Grafana: **Configuración → Fuentes de Datos → Probar**
2. Verifique los registros de Grafana: `journalctl -u grafana-server -f`
3. Intente volver a cargar el tablero desde el repositorio

Alertas No Disparando

Síntomas: La condición se cumple pero no se recibe notificación

Causas posibles:

- La alerta está en pausa
- Punto de contacto mal configurado
- Política de notificación no coincidente

Resolución:

1. Verifique el estado de la alerta en Grafana: **Alertas → Reglas de alerta**
2. Verifique que la prueba del punto de contacto funcione: **Alertas → Puntos de contacto → Probar**
3. Revise el enrutamiento de la política de notificación

Grafana No Puede Conectarse a la Fuente de Datos

Síntomas: "Puerta de Enlace Incorrecta" o tiempo de espera de conexión en Grafana

Causas posibles:

- Servicio de Prometheus/Loki/InfluxDB caído
- Firewall bloqueando conexiones locales
- Servicio vinculado a la interfaz incorrecta

Resolución:

1. Verifique el estado del servicio:
   ```
   systemctl status prometheus loki influxdb
   ```
2. Verifique que el servicio esté escuchando:
   ```
   ss -tlnp | grep -E '9090|3100|8086'
   ```
3. Pruebe la conectividad local:
   ```
   curl http://localhost:9090/api/v1/status/config
   curl http://localhost:3100/ready
   ```

Monitoreo de Suscriptores VIP

El Monitoreo de Suscriptores VIP rastrea el servicio de suscriptores seleccionados (servicios de emergencia, hospitales, enrutadores clave de clientes y teléfonos IP): registro IMS, adjunto EPC y accesibilidad de UE. La lista es gestionada en **OmniWeb → Monitoreo → Suscriptores VIP**, por sitio, por el equipo de operaciones; nada en este repositorio tiene que cambiar para agregar o eliminar un suscriptor.

Arquitectura



OmniWeb pregunta al HSS del sitio por cada suscriptor vigilado, hace ping a la dirección de UE a través del exportador blackbox de un host de sondeo que enruta al grupo de UE (elegido en la página de los grupos de inventario `pcscf`, `pgwc`, `sgwc` y `monitoring`), y expone el resultado. Prometheus lo raspa como trabajo `vip_subscribers`.

Tipos de Servicio

Tipo	IMS requerido	EPC requerido	Caso de uso
Voz + datos (`full`)	Sí	Sí	Dispositivos móviles
Solo voz (`voip_only`)	Sí	No	Dispositivos solo VoIP, cuentas IMS sin SIM (identificador = nombre de usuario SIP)
Solo datos (`data_only`)	No	Sí	Enrutadores, IoT, M2M

`vip_subscriber_service_healthy` es 1 cuando se cumplen los requisitos del tipo y la cuenta está habilitada en el HSS. El ping nunca cambia el veredicto.

Configuración

Nada en el inventario. Prometheus recopila el servicio OmniWeb en el host OAM como trabajo `vip_subscribers`; el nombre del endpoint es subscribers, por lo que el nginx de OmniWeb admite solo llamadas de loopback a `/api/vip-subscribers/metrics`.

El host de sondeo, pings por verificación, tiempo de espera e intervalo de re-verificación se configuran en la página (panel de sondeo), por sitio.

Métricas

Los nombres de las métricas no han cambiado respecto al exportador anterior por sitio, por lo que los paneles y la regla de alerta se mantienen.

Métrica	Descripción
`vip_subscriber_service_healthy`	1 si el suscriptor cumple con la regla para su tipo, de lo contrario 0. Alerta sobre esto.
`vip_subscriber_ims_registered`	1 si se asigna un S-CSCF
`vip_subscriber_epc_registered`	1 si un MME tiene al suscriptor (o un ULR en la última hora)
`vip_subscriber_hss_reachable`	1 si el HSS respondió
`vip_subscriber_enabled`	1 si la cuenta está habilitada en el HSS
`vip_subscriber_ims_registration_age_seconds` / `vip_subscriber_epc_registration_age_seconds`	Segundos desde la última registración / Actualización de ubicación, -1 si no hay ninguna
`vip_subscriber_ue_ip_reachable`, `vip_subscriber_ping_loss_percent`, `vip_subscriber_ping_rtt_{avg,min,max}_ms`, `vip_subscriber_ping_packets_{sent,received}`	Resultado del ping; presente solo cuando se establece un host de sondeo y el UE tiene una dirección

Métrica	Descripción
`vip_subscriber_info`	Siempre 1; las etiquetas llevan `assigned_scscf`, `last_seen_mme`, `ue_ip`, `ue_apn`, `error`
`vip_subscriber_last_check_timestamp_seconds`	Tiempo Unix de la última verificación

Etiquetas en cada serie: `customer`, `site`, `hss`, `imsi`, `label`, `msisdn` (del HSS), `type`.

Paneles y alertas

Los paneles **VIP Subscribers** y **IMS VIP Subscribers** se envían con el rol de monitoreo (carpeta IMS) en cada sitio. La regla **VIP Subscriber Service Unhealthy** (`ims-vip-subscriber-service-unhealthy`) se activa de forma crítica cuando `vip_subscriber_service_healthy == 0` durante un minuto; un suscriptor pausado en la página se elimina de las métricas, por lo que una interrupción conocida no lo mantiene activo.

Exportador legado

Las implementaciones anteriores ejecutaban un exportador de Python en el host OAM (`ims-subscriber-exporter`, puerto 9550) alimentado por `hosts/<customer>/group_vars/vip_subscribers.yaml`. El rol aún lo instala mientras ese archivo exista y lo retira (detiene la unidad, elimina los archivos) una vez que se elimina el archivo. Para migrar un sitio: ingrese sus suscriptores en la página de OmniWeb, configure el host de sondeo, confirme que el trabajo `vip_subscribers` está activo, luego elimine el archivo YAML y ejecute el rol.

Documentación relacionada

- Centralized Logging: Configuración detallada de Loki y Alloy
- Deployment Architecture: Arquitectura general del sistema
- Hosts File Configuration: Configuración del inventario

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

Habilitar 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 plantilla

No hay un `netplan.yaml.j2` compartido y canónico. Cada cliente crea su propio `netplan.yaml.j2` en `hosts/<customer>/group_vars/`, y estas plantillas divergen - el nombre de la interfaz principal, el dominio de búsqueda DNS y la nomenclatura de la interfaz secundaria varían según el cliente. Al incorporar un nuevo cliente, crea una plantilla por cliente que coincida con el diseño de interfaz de ese cliente en lugar de copiar la de otro cliente verbatim.

Lo siguiente es la plantilla de un cliente como **ejemplo**. Es ilustrativa, no una especificación - otros clientes son diferentes. Se añaden comentarios aquí para explicación:

```
network:
  version: 2
  ethernets:
    # Interfaz principal - 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, utiliza DNS externo
para evitar dependencia circular
        - 8.8.8.8
{% else %}
        # De lo contrario, utiliza servidores DNS del grupo 'dns'
en el inventario
{% for dns_host in groups['dns'] | default([]) %}
        - {{ hostvars[dns_host]['ansible_host'] }}
{% endfor %}
{% endif %}
        # Dominio de búsqueda DNS - específico del cliente (varía
por cliente; algunos no establecen ninguno)
        search:
        - customer.example
      routes:
      - to: "default"
        via: "{{ gateway }}"

{% if secondary_ips is defined %}
    # Interfaces secundarias - iterar a través del diccionario
secondary_ips del inventario
    # Este ejemplo las nombra ens19, ens20, ens21... (18 +
loop.index). Esto es
    # específico de la plantilla; ver "Nomenclatura de interfaz" a
continuación.
{% for nic_name, nic_config in secondary_ips.items() %}
    ens{{ 18 + loop.index }}:
      addresses:
      # La máscara proviene de la máscara por NIC mask_cidr
(predeterminado a 24)
      - "{{ nic_config.ip_address }}/{{ nic_config.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`
- El dominio de búsqueda DNS es específico del cliente (varía; algunos clientes lo omiten)
- La nomenclatura de la interfaz secundaria varía según el cliente (algunos usan `ens19`, `ens20`, ...; ver abajo)
- Cada IP secundaria puede tener su propia máscara (`mask_cidr`), puerta de enlace y rutas estáticas

Configuración de la interfaz principal

La interfaz principal (eth0) se configura automáticamente utilizando:

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

Los servidores DNS se configuran automáticamente a:

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

MTU de enlace (`link_mtu`)

El rol `common` puede fijar el **MTU de enlace** de la interfaz principal independientemente de la plantilla de netplan, a través de la variable de inventario `link_mtu`. Cuando se establece, genera un drop-in de netplan (`/etc/netplan/999-link-mtu.yaml`) en la interfaz principal del host y lo aplica. Esto existe porque las VMs provienen de la plantilla de VMware/Proxmox con cualquier MTU que el grupo de puertos de origen envió (a veces jumbo/9000), lo que no coincide con el resto de la red y provoca agujeros negros en SCTP/SIP sobredimensionados cuando PMTUD está roto.

El drop-in se nombra deliberadamente `999-` para que se ordene **último**: netplan fusiona todos `/etc/netplan/*.yaml` en orden lexicográfico con el último archivo que gana por clave, y la provisión puede dejar archivos en competencia que establecen `mtu` en la misma interfaz (un `80-ansible-config.yaml` heredado y el `99-netcfg-vmware.yaml` del motor de VMware). Un drop-in de número más bajo sería sobrescrito silenciosamente por esos.

```yaml
all:
  vars:
    link_mtu: 1500   # fijar el NIC primario de cada host a 1500
```

Establecerlo en `all.vars` para aplicar en todo el sitio, o por host para anular. Si `link_mtu` no está definido, el rol deja el MTU de la interfaz sin cambios.

> **`link_mtu` NO es `mtu`.** La variable `mtu` de nivel superior es el MTU de carga útil de GTP/túnel (por ejemplo, 1400/1430/1300) consumido por los roles de núcleo de paquetes, **no** un MTU de enlace. Establecer el NIC físico en esos valores sería incorrecto. Siempre utiliza el dedicado `link_mtu` para el MTU de la interfaz.

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>
    mask_cidr: <prefix_length>   # Opcional - máscara por NIC,
predeterminada a 24
    gateway: <gateway_ip>
    host_vm_network: <proxmox_bridge>
    vlanid: <vlan_id>
    nic_name: <interface_name>   # Opcional - anulación explícita
del nombre de la interfaz
                                 # (soportado por plantillas que
lo honran)
    routes:                      # Opcional - rutas estáticas a
través de esta interfaz
      - '<destination_cidr>'
      - '<destination_cidr>'
    table_routes:                # Opcional - avanzado, específico
de la plantilla
      - { to: '<cidr>', via: '<gateway>', table: <table_id> }
    routing_policy:              # Opcional - avanzado, específico
de la plantilla
      from: '<source_cidr>'
      via: '<gateway>'
      table: <table_id>
      priority: <priority>       # Opcional, predeterminado a 100
```

> La clave `mask_cidr` establece la longitud del prefijo para esa interfaz secundaria. Se lee desde dentro de la entrada de IP secundaria (`nic_config.mask_cidr`), no desde un `mask_cidr` de nivel superior. Si se omite, predetermina a `/24`.

> `table_routes` y `routing_policy` habilitan el enrutamiento por política/origen, pero solo se renderizan por plantillas que los implementan. El enrutamiento avanzado es específico de la plantilla. Confirma que el `netplan.yaml.j2` del cliente honre estas claves antes de usarlas.

Nomenclatura de interfaz

La nomenclatura de la interfaz secundaria es específica de la plantilla - no está estandarizada entre clientes:

- Algunos clientes nombran las interfaces por índice: `ens19`, `ens20`, `ens21`, ... (`18 + loop.index`). Esto coincide con los nombres de interfaz asignados por Proxmox al agregar NICs adicionales a una VM.
- Otros clientes utilizan un `nic_config.nic_name` explícito cuando se establece, recurriendo a un nombre basado en índice (por ejemplo, `ens(34 + loop.index)`) de lo contrario.

Si una plantilla honra `nic_name`, puedes fijar una interfaz secundaria a un nombre específico estableciendo `nic_name` en esa entrada de IP secundaria; de lo contrario, se aplica el esquema basado en índice de la plantilla.

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
          mask_cidr: 28
          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 de Netplan generada

Renderizada con la plantilla de ejemplo, la configuración anterior genera (el dominio de búsqueda DNS y la nomenclatura `ens19`/`ens20` son específicos del cliente):

```
network:
  version: 2
  ethernets:
    eth0:
      addresses:
      - "10.0.1.100/24"
      nameservers:
        addresses:
        - 10.0.1.53
        search:
        - customer.example
      routes:
      - to: "default"
        via: "10.0.1.1"
    ens19:
      addresses:
      - "192.0.2.50/28"
      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"
```

La interfaz `public_ip` se renderiza como `/28` porque su entrada estableció `mask_cidr: 28`, mientras que `peering_ip` recurre al valor predeterminado de `/24`.

Integración con Proxmox

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

1. **Nuevas VMs**: Los NICs secundarios se agregan durante la provisión inicial

2. **VMs existentes**: Se agregan NICs secundarios y la VM se reinicia para aplicar los cambios

La configuración de Proxmox utiliza:

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

Cómo funciona



1. Se respaldan las configuraciones de netplan existentes en `/etc/netplan` (`*.bak`) y luego se eliminan para prevenir conflictos
2. La plantilla Jinja2 del cliente se renderiza en `/etc/netplan/01-netcfg.yaml` (modo `0600`)
3. `netplan try --timeout 30` aplica la configuración con retroceso automático si se pierde la conectividad
4. `wait_for_connection` confirma que el host se mantuvo accesible
5. `netplan apply` confirma la configuración

Casos de uso comunes

Agente de borde de diámetro (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 multi-homed 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 necesites referenciar una IP secundaria de un host *diferente* (por ejemplo, configurando una conexión de par), 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'] }}

# Iterar a través de todos los hosts DRA y obtener 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 par DRA

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

```
# En dra_config.yaml.j2 - usar 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
```

Verificando 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 interfaz

SSH a la VM y verifica los nombres de interfaz:

```
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 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 VM/LXC en Proxmox - Provisión de VM
- Referencia de configuración - Todas las variables de configuración

Syslog de Dispositivos de Red

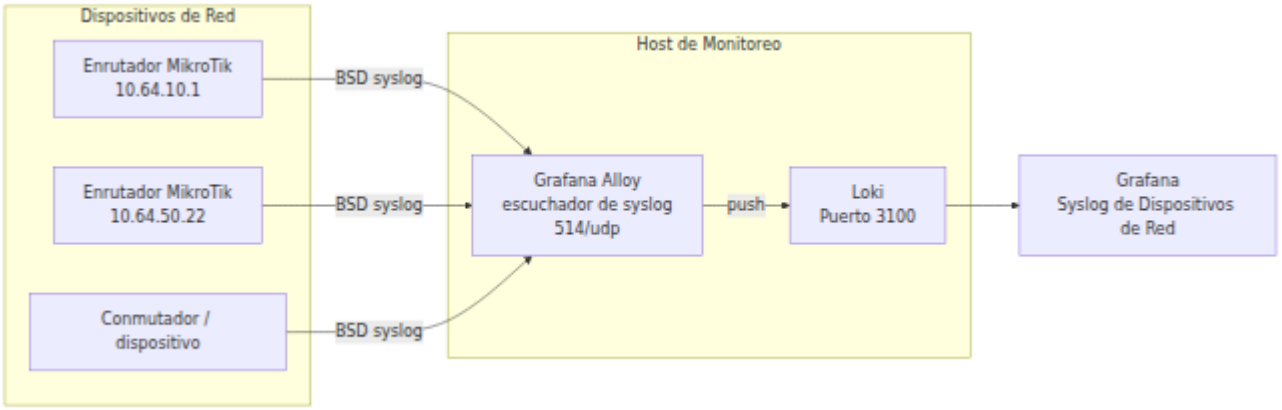
Los enrutadores, conmutadores y otros dispositivos de red no pueden ejecutar el agente Grafana Alloy. Estos dispositivos envían syslog en su lugar. Cada host de monitoreo ejecuta un escuchador de syslog dentro de Alloy. El escuchador escribe los mensajes en el Loki local. Los registros de dispositivos se sitúan junto a los registros de red centrales y comparten la misma búsqueda, rango de tiempo y retención.

Tabla de Contenidos

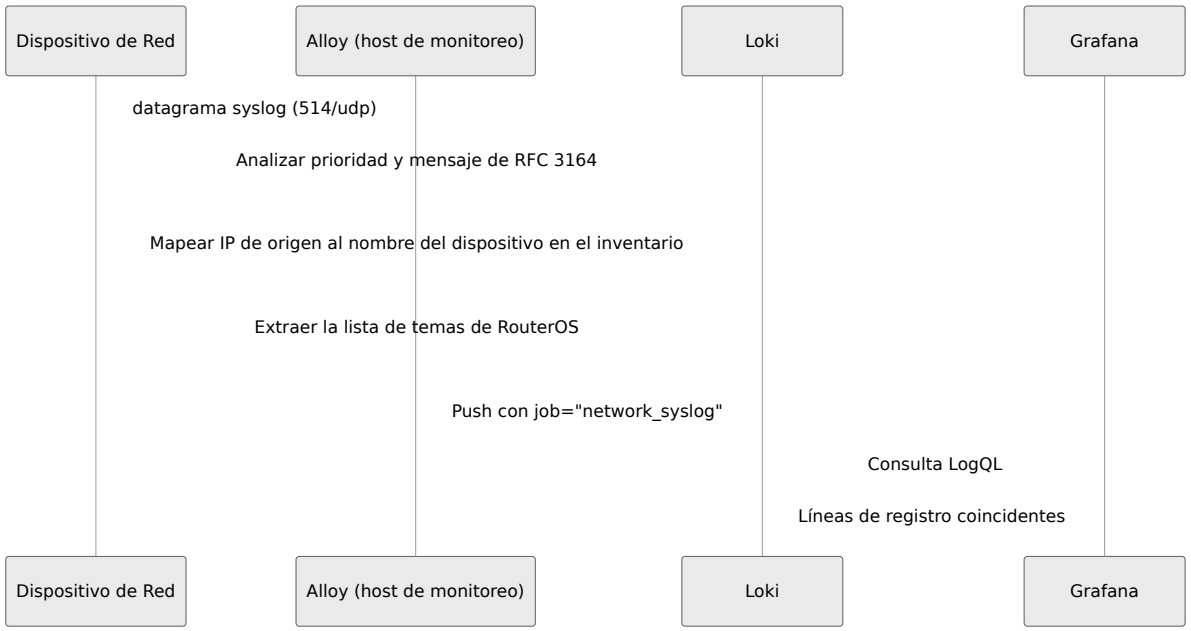
- Arquitectura
- Cómo se Identifican los Dispositivos
- Configuración del Colector
- Inventario de Dispositivos
- Etiquetas de Registro
- MikroTik RouterOS
- Otros Enviadores de Syslog
- Cortafuegos
- Tablero
- Consultando los Registros
- Resolución de Problemas
- Referencias

Arquitectura

El colector es parte de la construcción de monitoreo estándar. Cada host de monitoreo abre el escuchador. Un sitio no establece una bandera para recibir syslog de dispositivos.



El escuchador acepta BSD syslog, que es el formato que RFC 3164 define. Alloy analiza la prioridad, la marca de tiempo y el mensaje. Alloy luego agrega las etiquetas que las siguientes secciones describen.



Cómo se Identifican los Dispositivos

Alloy nombra cada dispositivo a partir de la dirección de origen del datagrama. Alloy compara esa dirección con los mapas `devices:` en el inventario del sitio. Una coincidencia establece las etiquetas `device` y `vendor`.

Dos resultados siguen de este diseño:

- La etiqueta del dispositivo es correcta incluso cuando el dispositivo envía una identidad incorrecta, una identidad vacía o una identidad duplicada en

el encabezado del mensaje.

- Un dispositivo que envía desde una dirección que el inventario no lista mantiene solo la etiqueta `device_ip`. Agrega la dirección al inventario para darle un nombre a ese dispositivo.

Si un dispositivo tiene más de una interfaz, asegúrate de que envíe desde la dirección que el inventario lista. Establece la dirección de origen en el dispositivo.

Configuración del Colector

Los parámetros del colector viven en `roles/common/defaults/main.yml`. Sobrescribe un parámetro en el inventario del sitio bajo `all.vars`.

```
all:
  vars:
    network_syslog_port: 514        # puerto del escuchador
    network_syslog_protocol: udp    # transporte
    network_syslog_vendors:         # bloques de proveedores que
nombran un emisor
      - mikrotik
```

Parámetro	Tipo	Requerido	Predeterminado	D
`network_syslog_port`	Entero	No	`514`	Pu se es er m Ur po de pr Co no co
`network_syslog_protocol`	Cadena	No	`udp`	Tr pa es Us `t` m lo di de er
`network_syslog_vendors`	Lista	No	`[mikrotik]`	Bl pr de in cu `d` no er sy pr lis

Parámetro	Tipo	Requerido	Predeterminado	D
				cc cc et nc di

Alloy se ejecuta como el usuario no privilegiado `alloy`. Por lo tanto, un puerto por debajo de 1024 necesita una capacidad. La unidad de Alloy en un host de monitoreo recibe `CAP_NET_BIND_SERVICE` automáticamente cuando `network_syslog_port` está por debajo de 1024. Un puerto por encima de 1024 no recibe la capacidad.

Aplica un cambio a estos parámetros con el rol `common`:

```
ansible-playbook -i hosts/<Customer>/host_files/<Site>.yml
services/common.yml \
  --limit <monitoring-host> --tags alloy
```

Los hosts de monitoreo ejecutan Alloy para este escuchador. Por lo tanto, un host de monitoreo también envía su propio registro, SSH y registros de auditoría a su Loki local.

Inventario de Dispositivos

El colector lee los mismos mapas de proveedores que utiliza el exportador SNMP. Agrega cada dispositivo una vez. La entrada luego le da al dispositivo una etiqueta de registro, un objetivo SNMP y una regla de cortafuegos.

```yaml
all:
  vars:
    mikrotik:
      devices:
        nf-nids-rtr-core01: 10.64.10.1
        nf-flgstf-rtr01: 10.64.50.22
      snmp_community: <community-string>
```

Parámetro	Tipo	Requerido	Predeterminado	Descripció
`devices`	Mapa	Sí	-	Nombre del dispositivo a dirección IP. El nombre s convierte en la etiqueta `device`. La dirección identifica al emisor y abre el cortafuegos.
`snmp_community`	Cadena	No	`public`	Comunidad SNMP para el exportador SNMP de Prometheus El colector de syslog no la utiliza.

Etiquetas de Registro

El syslog de dispositivos llega bajo `job="network_syslog"`. Las siguientes etiquetas se aplican.

Etiqueta	Descripción	Ejemplo
`job`	Valor fijo para syslog de dispositivos.	`network_syslog`
`component`	Valor fijo para syslog de dispositivos.	`network_device`
`device`	Nombre del dispositivo del inventario. Alloy coincide con la dirección del emisor.	`nf-nids-rtr-core01`
`device_ip`	Dirección de origen del datagrama.	`10.64.10.1`
`vendor`	Bloque de proveedor del inventario que contiene el dispositivo.	`mikrotik`
`level`	Severidad de syslog.	`informational`, `warning`, `err`, `crit`
`facility`	Instalación de syslog.	`daemon`
`topics`	Lista completa de temas de RouterOS del encabezado del mensaje.	`system,info,account`
`topic`	Primer tema en esa lista. Úsalo para el filtro común.	`system`, `firewall`, `dhcp`
`syslog_host`	Campo de host del encabezado de syslog, tal como lo escribió el emisor.	`system,info,account`

RouterOS no envía ninguna identidad propia. Escribe su lista de temas donde pertenece el nombre de host, por lo que el analizador almacena esa lista como

el host de syslog. El colector lee `topics` y `topic` de ese campo. Un emisor que escribe un nombre de host real allí establece solo `syslog_host` y no crea etiquetas de tema.

MikroTik RouterOS

Configura cada dispositivo desde la terminal de RouterOS. El dispositivo apunta su acción de registro `remote` incorporada al host de monitoreo. El dispositivo luego necesita una regla de registro para cada tema.

```
/system logging action set [find name=remote] target=remote
remote=<monitoring-host> remote-port=514 remote-log-format=bsd-
syslog
/system logging add topics=error action=remote
/system logging add topics=warning action=remote
/system logging add topics=critical action=remote
/system logging add topics=info action=remote
```

Propiedad	Valor	Descripción
`target`	`remote`	Envía el registro a un servidor syslog remoto en lugar de a la memoria o disco.
`remote`	Dirección del host de monitoreo	Destino del flujo de syslog.
`remote-port`	`514`	Puerto de destino. Coincide con `network_syslog_port`.
`remote-log-format`	`bsd-syslog`	Solo RouterOS 7. Envía RFC 3164. Consulta la tabla de versiones a continuación.
`src-address`	Dirección del dispositivo en el inventario	Opcional. Establecerlo cuando el dispositivo tiene más de una interfaz, para que la dirección de origen coincida con la entrada del inventario.

Formato de Mensaje por Versión de RouterOS

El colector analiza RFC 3164, por lo que el dispositivo debe enviar ese formato. La propiedad que controla el formato cambió entre las versiones principales. Verifica la versión con `/system resource print`.

Versión	Propiedad	Predeterminado	Acción
RouterOS 7	`remote-log-format`	`default`	Establecer `remote-log-format=bsd-syslog`. El valor `default` envía un formato de MikroTik sin encabezado de prioridad de syslog, que el colector no puede analizar. La propiedad `bsd-syslog` de RouterOS 6 no existe aquí, y ese comando falla con `bad parameter bsd-syslog`.
RouterOS 6	`bsd-syslog`	`no`	Establecer `bsd-syslog=yes`. El valor predeterminado envía un mensaje simple, que el colector no puede analizar.

```
# RouterOS 7
/system logging action set [find name=remote] remote-log-
format=bsd-syslog

# RouterOS 6
/system logging action set [find name=remote] bsd-syslog=yes
```

Una captura de paquetes muestra la diferencia. El formato incorrecto se decodifica como `(invalid)` en `tcpdump`, y la carga útil comienza con bytes binarios antes del texto del mensaje. El formato correcto se decodifica como `SYSLOG, Facility local0, Severity info`.

Temas de RouterOS

Tema	Descripción
`critical`	Fallos que detienen un servicio en el dispositivo.
`error`	Operaciones fallidas, por ejemplo, un arrendamiento DHCP fallido o un túnel fallido.
`warning`	Condiciones que necesitan atención pero no detienen un servicio.
`info`	Eventos normales. Este tema lleva los eventos de `account`, que son los inicios de sesión de usuarios, los inicios de sesión fallidos y los cambios de configuración.

Comienza con `error`, `warning` y `critical` para un volumen bajo. Agrega `info` cuando desees auditar inicios de sesión y cambios de configuración.

Verifica el resultado en el dispositivo:

```
/system logging print
/system logging action print
```

Otros Enviadores de Syslog

Cualquier dispositivo que hable BSD syslog puede usar el colector. Se aplican tres condiciones:

1. El dispositivo envía BSD syslog, como se define en RFC 3164. El colector no analiza RFC 5424.
2. El dispositivo envía UDP al puerto 514 en el host de monitoreo, a menos que el sitio haya cambiado `network_syslog_protocol` o `network_syslog_port`.

3. El dispositivo envía desde una dirección que un mapa `devices:` lista. El cortafuegos permite esa dirección, y la dirección le da al dispositivo su nombre.

Agrega un nuevo proveedor a `network_syslog_vendors` para etiquetar sus dispositivos. El cortafuegos ya acepta los dispositivos de cada proveedor.

Cisco IOS

```
logging host <monitoring-host>
logging trap informational
logging source-interface <management-interface>
```

Linux rsyslog

```
*.* @<monitoring-host>:514
```

El template predeterminado de rsyslog es RFC 3164. Mantén ese template. No selecciones `RSYSLOG_SyslogProtocol23Format`, porque ese template envía RFC 5424.

Dispositivos con una interfaz web

Establece el servidor syslog remoto y el puerto. Si el dispositivo ofrece una opción de formato, selecciona BSD o legado.

Cortafuegos

El modelo de flujo lleva la regla. El flujo `4b` permite `514/udp` al grupo `monitoring`. Las fuentes provienen del token `{ net_devices: true }`, que resuelve cada dirección en cada mapa `all.vars.<vendor>.devices`.

Un dispositivo que no está listado en el inventario se descarta cuando el sitio ejecuta el cortafuegos en modo `enforce`. Agrega el dispositivo al inventario, luego aplica el cortafuegos:

```
ansible-playbook -i hosts/<Customer>/host_files/<Site>.yml
services/firewall.yml \
  --limit <monitoring-host>
```

Consulta la guía de Cortafuegos por Rol para el modelo de flujo y los modos.

Tablero

Grafana contiene el tablero **Syslog de Dispositivos de Red** en la carpeta **Logs**.

Panel	Descripción
Búsqueda de Registros de Dispositivos de Red	Búsqueda de texto libre en todos los dispositivos.
Tasa de Registros por Dispositivo	Volumen para cada dispositivo a lo largo del tiempo. Una línea plana en cero muestra un dispositivo que dejó de enviar.
Tasa de Registros por Severidad	Volumen para cada severidad a lo largo del tiempo.
Registros de Dispositivos de Red	Todos los registros de dispositivos para el dispositivo y la cadena de búsqueda seleccionados.
Errores, Advertencias y Críticos	Severidad `warning` y superior.
Inicios de Sesión, Fallos de Autenticación y Cambios de Configuración	Eventos de cuenta y sistema para auditoría.
Eventos de Enlace, OSPF/BGP y DHCP	Eventos de interfaz, enrutamiento y arrendamiento.

Variable	Descripción
`Site (logs)`	Fuente de datos de Loki. Selecciona el sitio.
`Device`	Desplegable de dispositivos. La lista proviene de los valores de la etiqueta `device` en Loki. Selecciona un dispositivo, varios dispositivos o `Todos`.
`Search`	Filtro de texto libre para los paneles de registro.

Consultando los Registros

```
# Cada mensaje de un dispositivo
{job="network_syslog", device="nf-nids-rtr-core01"}

# Severidad de advertencia y superior, en toda la propiedad
{job="network_syslog", level=~"emerg|alert|crit|err|warning"}

# Inicios de sesión y cambios de configuración
{job="network_syslog"} |~ `(?i)(logged in|login failure|changed
by)`

# Un tema de RouterOS
{job="network_syslog", topic="firewall"}

# Tasa de mensajes para cada dispositivo
sum by (device) (rate({job="network_syslog"} [5m]))

# Dispositivos que dejaron de enviar en la última hora
sum by (device) (count_over_time({job="network_syslog"} [1h]))
```

Resolución de Problemas

No Hay Registros de Dispositivos en Loki

Síntomas: Una consulta para `{job="network_syslog"}` no devuelve resultados.

Causas posibles:

- Alloy no se ejecuta en el host de monitoreo.
- El escuchador no vinculó el puerto.
- El dispositivo no envía.
- El cortafuegos descarta los datagramas.

Resolución:

1. Verifica el estado del agente en el host de monitoreo con `systemctl status alloy`.
2. Verifica que Alloy posea el puerto con `ss -lunp | grep 514`.
3. Captura el tráfico con `tcpdump -n -i any udp port 514`. El tráfico prueba que el dispositivo envía y que el cortafuegos permite el datagrama.
4. Si no hay tráfico, verifica la acción de registro en el dispositivo.

Alloy No Inicia

Síntomas: El servicio `alloy` se reinicia en un bucle. El diario muestra un error de vinculación o un error de permiso para el puerto 514.

Causas posibles:

- La unidad no tiene `CAP_NET_BIND_SERVICE`.
- Otro proceso ocupa el puerto, por ejemplo, un `rsyslogd` local con un escuchador UDP.

Resolución:

1. Lee el error con `journalctl -u alloy -n 50`.
2. Confirma la capacidad con `systemctl cat alloy`.
3. Encuentra el otro escuchador con `ss -lunp | grep 514`. Detén ese escuchador o establece `network_syslog_port` a un puerto libre y configura los dispositivos para que coincidan.

Los Datagramas Llegan Pero Loki No Almacena Nada

Síntomas: `tcpdump` muestra tráfico en el puerto 514. Loki no tiene ningún flujo `network_syslog`.

Causas posibles:

- El dispositivo no envía BSD syslog. Un dispositivo RouterOS 7 necesita `remote-log-format=bsd-syslog`. Un dispositivo RouterOS 6 necesita `bsd-syslog=yes`.

- El dispositivo envía RFC 5424.

Resolución:

1. Captura el tráfico con `tcpdump -n -i any -A udp port 514`. Un mensaje que `tcpdump` marca como `(invalid)` no tiene encabezado de prioridad de syslog, por lo que el formato es incorrecto.
2. Establece la propiedad de formato para la versión del dispositivo. Consulta Formato de Mensaje por Versión de RouterOS.
3. En otro dispositivo, selecciona el formato de syslog BSD o legado.
4. Verifica el registro de Alloy en busca de errores de análisis con `journalctl -u alloy | grep -i syslog`.

La Etiqueta del Dispositivo Está Vacía

Síntomas: Las líneas de registro llevan `device_ip` pero no etiqueta `device`.

Causas posibles:

- La dirección del emisor no está en un mapa `devices:`.
- El dispositivo envía desde una segunda interfaz, por lo que la dirección de origen difiere de la dirección del inventario.

Resolución:

1. Lee la dirección de origen de la etiqueta `device_ip`.
2. Agrega esa dirección al mapa `devices:` del proveedor, o establece la dirección de origen en el dispositivo.
3. Aplica el cambio con el rol `common`, y con la jugada del cortafuegos si el sitio aplica el cortafuegos.

Un Dispositivo Dejó de Enviar

Síntomas: El panel **Tasa de Registros por Dispositivo** no muestra datos para un dispositivo. Otros dispositivos continúan.

Causas posibles:

- Un operador eliminó o desactivó la regla de registro en el dispositivo.
- El dispositivo perdió la ruta al host de monitoreo.
- La dirección del inventario cambió, por lo que el cortafuegos descarta la nueva dirección de origen.

Resolución:

1. Verifica las reglas en el dispositivo con `/system logging print`.
2. Haz ping al host de monitoreo desde el dispositivo.
3. Confirma que la dirección actual del dispositivo coincide con la entrada del inventario.

Referencias

Referencia	Título
RFC 3164	El Protocolo syslog BSD
RFC 5424	El Protocolo Syslog. El colector no analiza este formato.
Registro Centralizado	Canalización de registros del host, retención y tableros
Cortafuegos por Rol	Modelo de flujo, tokens de origen y modos
Monitoreo y Observabilidad	Prometheus, Grafana y métricas

Operaciones de Replicación y Nodo de OmniHSS

OmniHSS almacena sus datos de suscriptores, autenticación (AuC) e IMS en una base de datos local de PostgreSQL en cada nodo. **La replicación lógica bidireccional de PostgreSQL** mantiene sincronizados múltiples nodos de OmniHSS. Esto forma una **malla activa-activa**: cada nodo es un primario completo de lectura/escritura, y una escritura en cualquier nodo se propaga a todos los demás. No hay primario/respaldo. Cualquier nodo puede servir tráfico S6a, Cx/Dx y Sh y aceptar aprovisionamiento en cualquier momento.

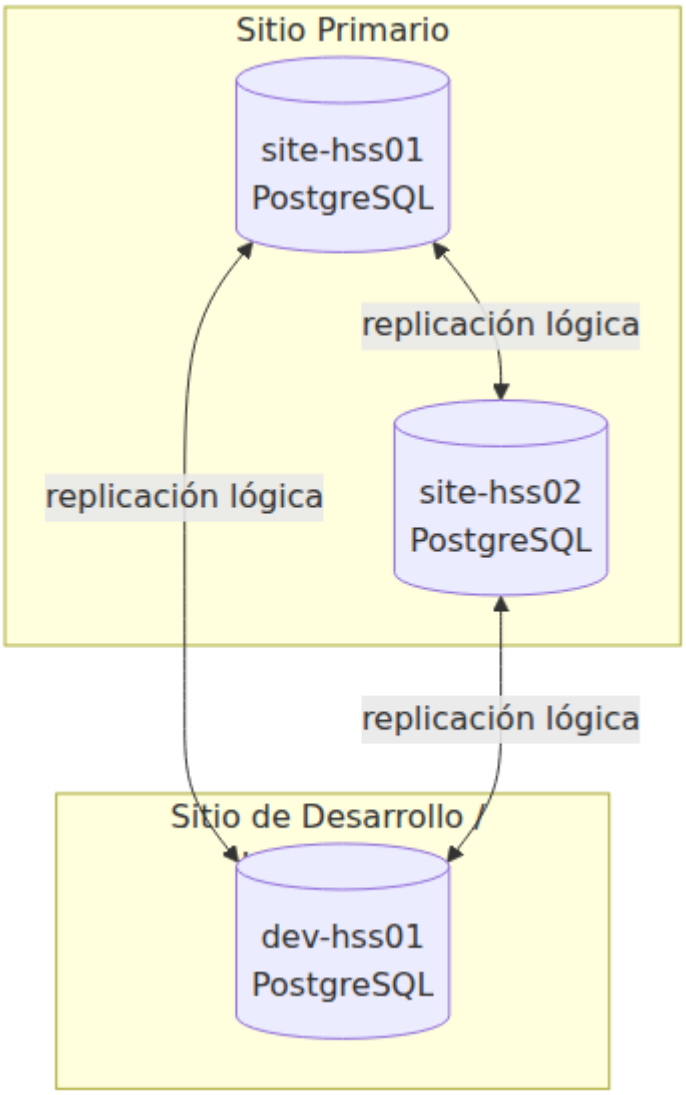
Esta guía cubre el modelo de replicación. Explica cómo agregar un nuevo nodo de OmniHSS y cargarlo con datos. También explica cómo validar, reparar y eliminar pares.

Tabla de Contenidos

- Descripción General de la Arquitectura
- Cómo se Descubren los Pares
- Configuración
- Agregar un Nuevo Nodo de OmniHSS
- Cargar un Nuevo Nodo con Datos
- Validar la Replicación
- Integridad de los Datos
- Eliminar un Par
- Copia de Seguridad y Restauración
- Métricas
- Solución de Problemas

Descripción General de la Arquitectura

Cada nodo de OmniHSS ejecuta su propia instancia de PostgreSQL. Cada nodo **publica** todas sus tablas de aplicación y **se suscribe** a la publicación de cada par. `origin = none` previene bucles de replicación. En las tablas de estado de rápido cambio, la malla resuelve conflictos de escritura **el último en escribir gana (LWW)** al comparar `updated_at`.

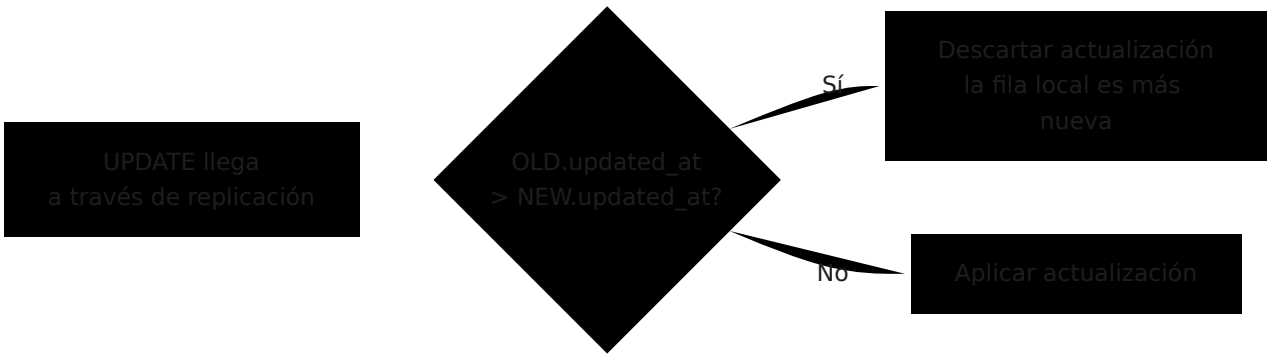


Cada enlace anterior es dos suscripciones unidireccionales (una por dirección). Para una malla de `N` nodos, cada nodo tiene `N-1` suscripciones. Los pares consumen `N-1` de los espacios de replicación de su publicación.

Qué se Replica

Categoría de tabla	Ejemplos	Manejo de conflictos
Datos maestros / aprovisionados	`subscriber`, `key_set` (AuC), `sim`, `msisdn`	No se necesita: aprovisionado desde un lugar a la vez
Estado dinámico	`subscriber_state`, `pdn_session`, `lte_call`, `ims_subscription`	El último en escribir gana se activa en `updated_at`
Contabilidad de migración	`schema_migrations`	**Excluido** de la replicación

La publicación cubre cada tabla en el esquema `public` **excepto** `schema_migrations` (cada nodo ejecuta sus propias migraciones de forma independiente). Cuando dos nodos actualizan la misma fila de estado de manera concurrente, los activadores LWW mantienen el `updated_at` más nuevo en lugar de la actualización que llegó más tarde.



Cómo se Descubren los Pares

Cuando la replicación está habilitada, el rol de OmniHSS construye su lista de pares automáticamente a partir de dos fuentes:

1. **Otros hosts en el grupo de inventario local `hss`**: los nodos de OmniHSS definidos en el *mismo* archivo de inventario/hosts (por ejemplo, un par local `hss01` / `hss02`). Estos **no** necesitan estar listados en `connected_omnihss`; la membresía del grupo es suficiente.
2. **La lista `connected_omnihss`** en el archivo de pares remotos del sitio: nodos de OmniHSS definidos en un *archivo* de hosts *diferente* (otros sitios). Los pares entre archivos solo se pueden descubrir aquí, porque un inventario no tiene conocimiento del grupo `hss` de otro.

El rol filtra los pares del grupo local a los hosts que son participantes de replicación de OmniHSS: aquellos cuyo `omnihss.replication.enabled` es `true`. Un nodo cuyo indicador no es `true` no se considera un par de replicación.

El rol excluye automáticamente el host actual de su propia lista de pares, por lo que el mismo bloque `connected_omnihss` puede ser compartido por cada sitio sin que un nodo intente replicarse a sí mismo.

Nomenclatura de Objetos

Cada nodo nombra sus objetos de replicación de manera que una malla de cualquier tamaño nunca colisione:

Objeto	Nombre	Creado en	Único por
Publicación	`<self>_pub`	este nodo	nodo
Suscripción	`sub_from_<source>`	este nodo (suscriptor)	(suscriptor)
Espacio de replicación	`<subscriber>_from_<source>`	la fuente (publicador)	par (suscriptor, fuente)

El nombre del espacio **debe** codificar el suscriptor. Considera lo que sucede si solo codifica la fuente (el valor predeterminado de PostgreSQL, donde el espacio hereda el nombre de la suscripción). Dos nodos suscribiéndose a la

misma fuente intentarían crear un espacio con el mismo nombre en esa fuente. La segunda suscripción fallaría con *"el espacio de replicación ya existe"*. Codificar el suscriptor le da a cada enlace un espacio globalmente único.

Reconciliación y Seguridad de `--limit`

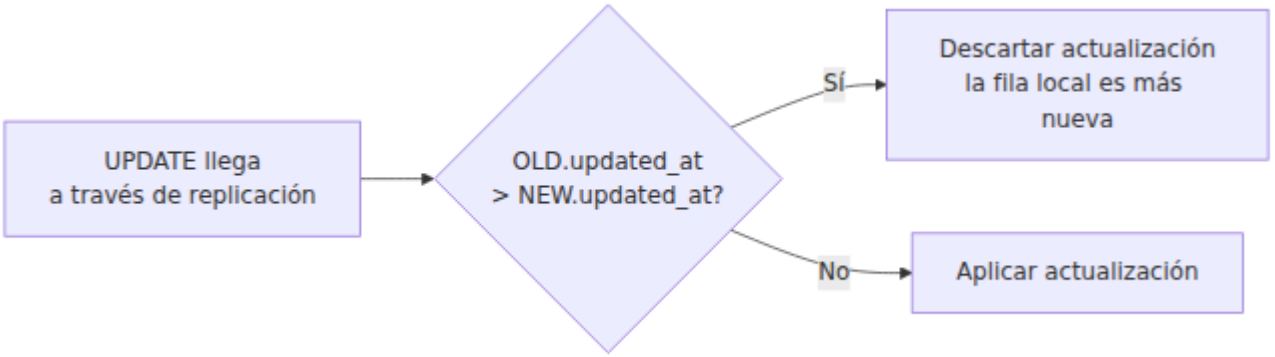
Cada ejecución de rol **reconciliará** los objetos de replicación del nodo contra el conjunto de pares válidos actual (grupo local `hss` + `connected_omnihss`, menos el propio):

- Las publicaciones distintas de `<self>_pub` se eliminan (limpia automáticamente publicaciones obsoletas/renombradas).
- Las suscripciones cuyo par ya no está en el conjunto válido se desactivan, se separan de su espacio remoto y se eliminan.
- Los espacios lógicos inactivos que no alimentan un par válido actual se eliminan. Los espacios activos nunca se tocan.

Esto hace que `connected_omnihss` sea la lista de malla **autoritativa**: la próxima ejecución poda un par eliminado de ella (y de los grupos `hss` relevantes) de cada nodo. Dos propiedades de seguridad evitan que esto sea destructivo:

- **`--limit` es seguro.** El conjunto válido se deriva de `groups['hss']` y `connected_omnihss`, que siempre reflejan la membresía del inventario *completa* independientemente de `--limit`. Ejecutar `--limit hss01` reconcilia solo `hss01`. Pero `hss01` todavía ve `hss02`/`hss03` como válidos y mantiene sus suscripciones. `--limit` nunca poda los pares excluidos.
- **Los conjuntos vacíos nunca podan.** Los pasos de poda se omiten por completo en lugar de eliminar toda la replicación si el conjunto de pares válidos se calcula como vacío (por ejemplo, un inventario parcial con un solo HSS y `connected_omnihss` no cargado).

> **Advertencia sobre la lista autoritativa:** debido a que la poda es automática, la ejecución poda un par entre archivos que es genuinamente parte de la malla pero que falta en `connected_omnihss`. Siempre mantén `connected_omnihss` completo para cada par entre archivos.



Configuración

Habilitar la Replicación

La replicación se controla por implementación mediante el indicador `omnihss.replication.enabled`. Por defecto, es `false`. Configúralo en el inventario del sitio o en group_vars:

```
omnihss:
  database_port: 5432
  database_name: "omnihss"
  database_username: "hss"
  database_password: "password"
  replication:
    enabled: true
```

Parámetro	Tipo	Requerido	Predeterminado	
`database_port`	Entero	No	`5432`	Puert tamb puert cone
`database_name`	Cadena	No	`omnihss`	Nom datos todos
`database_username`	Cadena	Sí	`hss`	Rol d Crea `REPL` pued repli
`database_password`	Cadena	Sí	`password`	Cont inicio coinc (utili cone
`replication.enabled`	Booleano	No	`false`	Inter Cuar Postg para crear publi

Cuando `replication.enabled` es `true`, el rol aplica la siguiente configuración de PostgreSQL en la instancia local:

Configuración	Valor
`wal_level`	`logical`
`max_wal_senders`	`10`
`max_replication_slots`	`10`
`listen_addresses`	`*`
`max_slot_wal_keep_size`	`10GB` (sobrescribir a través de `omnihss.replication.max_slot_wal_keep_size`)

Configuración	Valor

`pg_hba.conf` se actualiza para permitir la IP de cada par (`/32`) tanto para conexiones `all` como para `replication`.

Declarar Pares entre Sitios (`connected_omnihss`)

Los nodos remotos de OmniHSS se declaran en el archivo de pares remotos referenciado por la variable `remote_peers_file` de cada sitio (comúnmente `prod_master_peers.yml`):

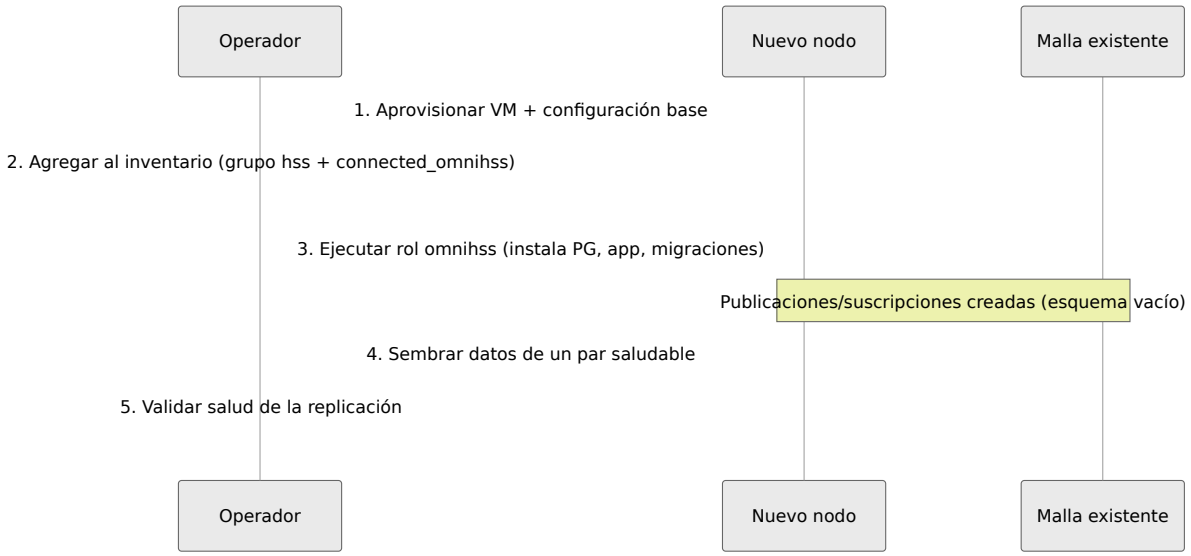
```
connected_omnihss:
  - name: site-hss01
    host: 10.4.2.142
    port: 5432
  - name: site-hss02
    host: 10.4.2.143
    port: 5432
  - name: dev-hss01
    host: 10.4.10.142
    port: 5432
```

Parámetro	Tipo	Requerido	Predeterminado	Descripción
`name`	Cadena	Sí	-	El `inventory_hostn` del par. Esto nom la suscripción (`sub_from_<name` resuelve la publicación del p (`<name>_pub`), pc que **debe** coincic con el nombre de host de inventari real del par.
`host`	Cadena	Sí	-	La dirección IP de par, accesible de este nodo a trave de la red de señalización.
`port`	Entero	No	`5432`	El puerto de PostgreSQL del p Se retrocede a `database_port`.

Debido a que el archivo de pares remotos se carga en cada sitio, la lista completa de `connected_omnihss` debe contener **cada** nodo de OmniHSS en la malla. Cada sitio omite automáticamente sus propios nodos locales (ya están cubiertos por el grupo local `hss`) y se omite a sí mismo.

Agregar un Nuevo Nodo de OmniHSS

Agregar un nodo tiene dos fases distintas: **unirse a la malla** (para que los cambios fluyan en ambas direcciones) y **cargar los datos existentes** en el nuevo nodo. La segunda fase es importante porque un nodo completamente nuevo comienza con una base de datos vacía. Consulta Cargar un Nuevo Nodo con Datos.



Paso 1: Aprovisionar el host

Aprovisiona la VM y la red base como para cualquier nodo de OmniCore (consulta Despliegue de Proxmox). El nodo debe alcanzar el puerto de PostgreSQL de cada par (`5432` por defecto) a través de la red de señalización, y los pares deben poder alcanzarlo.

Paso 2: Agregar el nodo al inventario

Agrega el nuevo host al archivo de host del sitio bajo el grupo `hss:`:

```yaml
hss:
  hosts:
    site-hss01:
      ansible_host: 10.4.2.142
      gateway: 10.4.2.1
      host_vm_network: "v401-omni-signalling"
    site-hss03:        # nuevo nodo
      ansible_host: 10.4.2.144
      gateway: 10.4.2.1
      host_vm_network: "v401-omni-signalling"
```

Luego, agrégalo a `connected_omnihss` en el archivo de pares remotos para que **otros sitios** también se enteren de él:

```yaml
connected_omnihss:
  - name: site-hss03
    host: 10.4.2.144
    port: 5432
  # ... entradas existentes ...
```

Paso 3: Ejecutar el rol de OmniHSS

Ejecuta el libro de jugadas del servicio OmniHSS contra el nuevo host. Esto instala PostgreSQL y OmniHSS, ejecuta las migraciones de la base de datos (creando un esquema **vacío**), configura la replicación lógica y crea la publicación más una suscripción a cada par:

```
ansible-playbook -i hosts/Customer_Name/host_files/Production.yml
\
  services/omnihss.yml --limit site-hss03
```

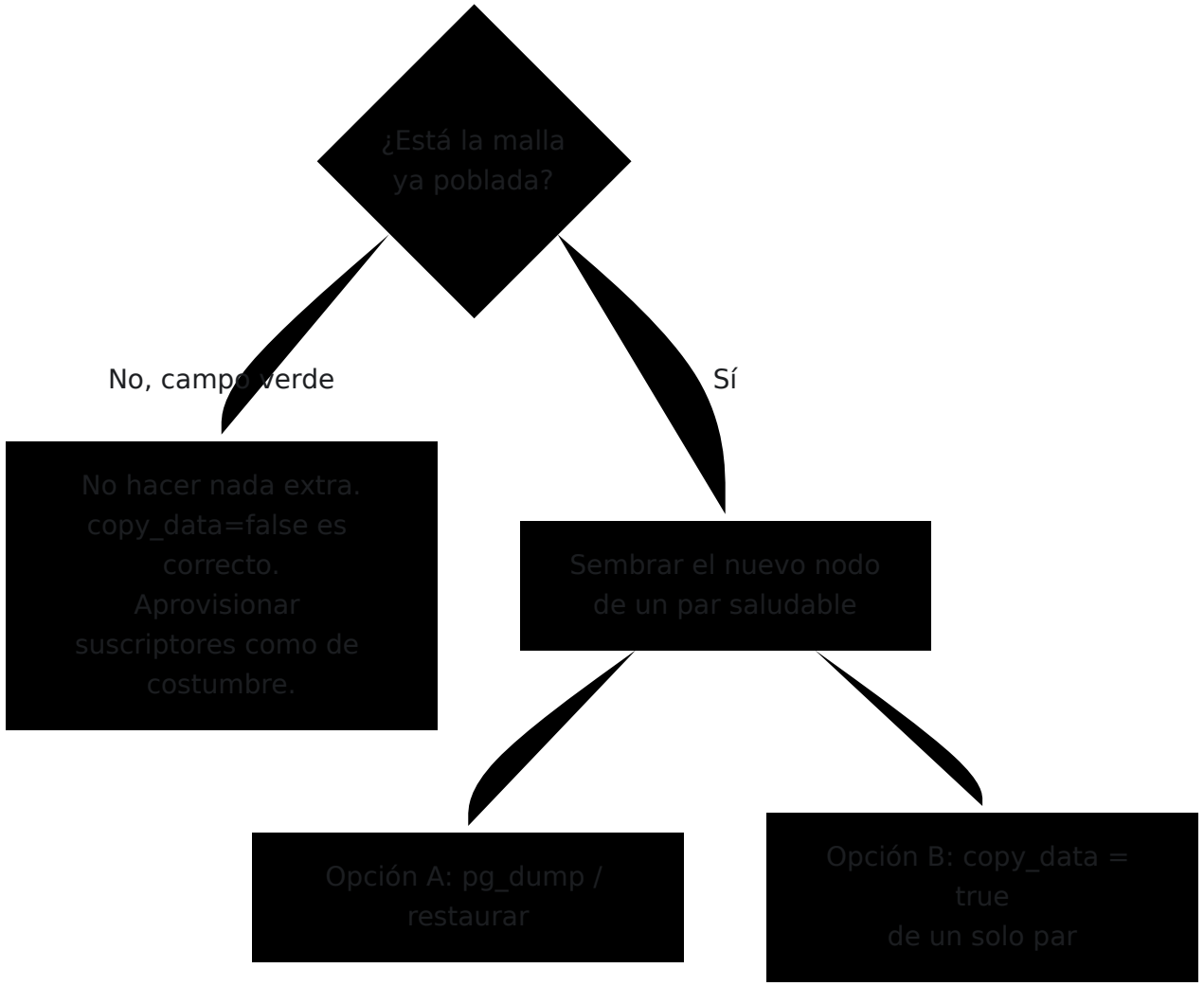
Vuelve a ejecutar el rol en los pares existentes (o deja que la próxima ejecución programada lo maneje) para que abran `pg_hba.conf` para el nuevo nodo y creen su suscripción de regreso a él. Las suscripciones son idempotentes: el rol detecta y omite las existentes, y emite `ALTER SUBSCRIPTION ... REFRESH PUBLICATION` para que recoja cualquier nueva tabla.

En este punto, el nuevo nodo es un miembro de la malla y recibe **todas las futuras escrituras**. Pero aún **no** contiene la base de suscriptores existente. Procede a cargar sus datos.

Cargar un Nuevo Nodo con Datos

El rol crea las suscripciones automatizadas con `copy_data = false`. Esto es correcto para una **malla de campo verde** (cada nodo comienza vacío al mismo tiempo: no hay nada que copiar, y copiar causaría conflictos de filas duplicadas). Pero significa que un nodo que se une a una malla **ya poblada** permanece vacío excepto por los cambios que ocurren después de que se une.

Elige el camino que coincida con tu situación:



Malla de Campo Verde

Si estás levantando todos los nodos de OmniHSS juntos sin datos preexistentes, no necesitas sembrar. Habilita la replicación, ejecuta el rol en todos los nodos y aprovisiona suscriptores normalmente. Cada escritura se replica a todos los miembros.

Opción A: Sembrar a través de volcado y restauración (recomendado)

Toma una instantánea de un par saludable y restáuralo en el nuevo nodo. Luego deja que la replicación lleve los cambios en curso. Esto reutiliza las herramientas estándar de copia de seguridad/restauración y es el enfoque más predecible, incluso para conjuntos de datos grandes.

1. **Hacer una copia de seguridad de un par saludable.** Esto produce un `pg_dump` de la base de datos de OmniHSS (tomado con `--no-publications --no-subscriptions`, por lo que los objetos de replicación no se transfieren) en la máquina de control:

   ```
   ansible-playbook -i
   hosts/Customer_Name/host_files/Production.yml \
     services/backup.yml --limit site-hss01
   ```

   El volcado se guarda como `backups/<Site_Name>/hss_dump_<hostname>_<timestamp>.sql`.

2. **Restaurar en el nuevo nodo.** El libro de jugadas de restauración detiene OmniHSS, elimina y recrea la base de datos, restaura el volcado y reinicia el servicio. Toma primero una copia de seguridad antes de la restauración:

   ```
   ansible-playbook -i
   hosts/Customer_Name/host_files/Production.yml \
     util_playbooks/restore_hss.yml --limit site-hss03
   ```

   Agrega `-e skip_pre_restore_backup=true` para omitir ese volcado previo a la restauración — útil cuando el destino es un nodo nuevo o ya vacío, o

cuando el volcado es lento y su contenido actual se sabe que no tiene valor. No se guarda nada, por lo que los datos actuales del nodo no pueden recuperarse.

Cuando se te pida, proporciona la ruta al volcado SQL del paso 1 y deja la ruta de configuración vacía (el nuevo nodo ya tiene su propia configuración de la ejecución del rol).

3. **Confirmar que la replicación está fluyendo.** Las suscripciones existentes del nuevo nodo (`copy_data = false`) ahora aplican cambios sobre la instantánea restaurada. Ejecuta la validación a continuación.

> **Ventana de mantenimiento:** Hay una pequeña ventana entre el volcado y el momento en que la replicación se pone al día. Durante ella, los cambios de aprovisionamiento en los pares aún no están en el nuevo nodo. Los cambios en los datos de suscriptores maestros son infrecuentes, y el estado dinámico (`subscriber_state`, `pdn_session`, `lte_call`, `ims_subscription`) se auto-repara a partir del tráfico en vivo, por lo que una ventana de baja actividad es suficiente. No aprovisionar nuevos suscriptores durante la siembra.

Opción B: Sembrar con el libro de jugadas de siembra (recomendado)

El libro de jugadas utilitario `seed_hss_replication.yml` siembra desde **un** par fuente elegido sin los inconvenientes de una copia masiva en bruto. Recrea la suscripción como solo de transmisión (`copy_data = false`, sin tablas de sincronización). Luego llena las filas de la **tabla maestra** existente con una carga segura de conflictos. `subscriber_state` **es** parte de la carga: es datos estructurales 1:1 (cada suscriptor referencia uno por FK, creado una vez al aprovisionar). Omitirlo deja a cada suscriptor rellenado huérfano, lo que rompe la API y S6a en el destino. La transmisión no puede reparar eso, porque la replicación lógica elimina silenciosamente las actualizaciones a filas que no existen. El libro de jugadas deliberadamente **no** copia en masa las verdaderas tablas de sesión transitorias (`pdn_session`, `lte_call`). Un `COPY` de esas tablas calientes colisiona (clave duplicada) con escrituras concurrentes en un sistema en vivo y provoca bucles de choque en la sincronización de tablas. Se repoblan a partir de la replicación en streaming y el tráfico en vivo en su lugar.

Interactivo (recomendado): omite la fuente. El libro de jugadas luego encuentra qué destinos están vacíos, sondea a los pares en busca de aquellos que tienen datos y te solicita que elijas:

```
ansible-playbook -i hosts/Customer_Name/host_files/Production.yml
\
  util_playbooks/seed_hss_replication.yml \
  --limit site-hss01,site-hss02
```

```
Pares con datos disponibles para sembrar desde:
  1) dev-hss01 (10.4.10.142) - 838 suscriptores
Ingresa el NÚMERO de la fuente desde la cual sembrar
```

Explícito: nombra la fuente directamente. Esto fuerza una nueva siembra incluso si el destino ya tiene datos (usar para recuperación):

```
ansible-playbook -i hosts/Customer_Name/host_files/Production.yml
\
  util_playbooks/seed_hss_replication.yml \
  --limit site-hss01 \
  -e "seed_source_name=dev-hss01 seed_source_host=10.4.10.142"
```

Consulta HSS Replication Seed para la referencia completa de parámetros y comportamiento.

`seed_hss_replication.yml` también **conecta toda la malla** después de sembrar: ejecuta la reconciliación de replicación en cada nodo (publicaciones, suscripciones, espacios) y la valida. Prefiere ejecutarlo sin **`--limit`** — los nodos vacíos se detectan automáticamente como destinos de siembra, los nodos poblados como fuentes, y cada nodo se conecta en una sola pasada. `--limit` reduce tanto la siembra como la conexión, por lo que un solo nodo `--limit` deja la malla medio conectada; úsalo solo para una nueva siembra dirigida, luego vuelve a ejecutar sin `--limit` para conectar el resto. Siembra desde exactamente una fuente. Prefiere Opción A para conjuntos de datos muy grandes.

Los pasos manuales equivalentes se ejecutan en el nodo de destino. Primero crea la suscripción de transmisión (nota el `slot_name` explícito por enlace, consulta Nomenclatura de Objetos). Luego llena las tablas maestras de manera segura frente a conflictos mientras omites las tablas de estado calientes:

```
-- 1. transmitir todas las tablas, sin copia masiva
DROP SUBSCRIPTION IF EXISTS sub_from_dev_hss01;
CREATE SUBSCRIPTION sub_from_dev_hss01
  CONNECTION 'host=10.4.10.142 port=5432 dbname=omnihss user=hss
password=<omnihss.database_password>'
  PUBLICATION dev_hss01_pub
  WITH (copy_data = false, origin = none,
        slot_name = 'site_hss01_from_dev_hss01');
```

```
# 2. llenar solo tablas maestras (ejecutar en el destino; -t cada
tabla excepto
#    schema_migrations, pdn_session, lte_call - subscriber_state
DEBE ser
#    incluido, o cada suscriptor rellenado queda huérfano: su
#    subscriber_state_id apunta a una fila que nunca llega,
rompiendo la
#    API/S6a, y la transmisión no puede repararlo porque la
replicación
#    lógica elimina silenciosamente las actualizaciones a filas
faltantes).
#    La carga DEBE estar estampada con un origen de replicación:
origin=none
#    las suscripciones filtran los cambios estampados en el
publicador, por lo que la carga es
#    invisible para la malla. Una carga no estampada se decodifica
desde WAL como cualquier
#    escritura local y eco los inserts a pares que ya tienen las
filas,
#    deteniendo sus trabajadores de aplicación por clave
duplicada.
sudo -u postgres psql -d omnihss -c \
  "SELECT pg_replication_origin_create('omnihss_seed_load');"
{ echo "SELECT
pg_replication_origin_session_setup('omnihss_seed_load');"
  PGPASSWORD=<pw> pg_dump -h 10.4.10.142 -U hss -d omnihss \
    --data-only --inserts --on-conflict-do-nothing --disable-
triggers \
    -t subscriber -t subscriber_state -t key_set -t sim -t msisdn
... ; } \
  | sudo -u postgres psql -d omnihss
sudo -u postgres psql -d omnihss -c \
  "SELECT pg_replication_origin_drop('omnihss_seed_load');"
```

Validando la Replicación

El rol ejecuta una verificación de salud automáticamente después de la configuración, y también se puede ejecutar por sí solo. La validación afirma que:

- Cada suscripción tiene un proceso de trabajo activo.
- Cada ranura de replicación lógica está activa.
- El retraso de WAL está por debajo de 100 MB por ranura.
- El estado de WAL de cada ranura es `reservado` o `extendido` (no `perdido`).

Para verificar la salud bajo demanda, vuelve a ejecutar el rol de OmniHSS (la validación se ejecuta al final), o inspecciona directamente en un nodo:

```
# Trabajadores de suscripción - pid debería ser no nulo para cada
uno
sudo -u postgres psql -d omnihss -c \
  "SELECT subname, pid IS NOT NULL AS worker_alive FROM
pg_stat_subscription;"

# Ranuras de replicación - active debería ser verdadero,
wal_status reservado/extendido
sudo -u postgres psql -d omnihss -c \
  "SELECT slot_name, active, wal_status, \
     pg_size_pretty(pg_wal_lsn_diff(pg_current_wal_lsn(),
confirmed_flush_lsn)) AS lag \
   FROM pg_replication_slots WHERE slot_type = 'logical';"
```

Síntoma	Significado
`worker_alive = false`	El trabajador de suscripción está caído: el par no es accesible o las credenciales/cadena de conexión son incorrectas.
`active = false` (ranura)	Ningún suscriptor está consumiendo la ranura: la suscripción del par está caída o fue eliminada.
Retraso creciente `lag`	El par es lento o el enlace está degradado. WAL se está acumulando y consumiendo disco.
`wal_status = lost`	La ranura se quedó demasiado atrás y PostgreSQL eliminó el WAL. Debes **resembrar** el par (ver Cargando un Nuevo Nodo con Datos).

El informe de verificación de salud (`health_check.yml`) también incluye un panel de **Replicación de HSS Postgres** para nodos con replicación habilitada, mostrando el estado del trabajador de cada suscripción y el estado activo y estado de WAL de cada ranura de un vistazo.

Integridad de Datos

La aplicación de replicación lógica no impone claves foráneas. El trabajador de aplicación se ejecuta con `session_replication_role = replica`, lo que suprime los desencadenadores internos de RI que implementan restricciones de `FOREIGN KEY`. PostgreSQL, por lo tanto, **nunca vuelve a verificar** una fila que llega a través de replicación contra su padre. Las restricciones de FK te dan integridad *por nodo*, no integridad *a través de la malla*.

En una malla activa-activa que abre un agujero que ningún nodo único puede atrapar:

- El nodo A elimina `epc_profile` X. Su `ON DELETE RESTRICT` pasa porque A no tiene un suscriptor local que haga referencia a X en ese instante.
- El nodo B (o un paso de aprovisionamiento) apunta al suscriptor 203 a X. Esto pasa porque X todavía existe en B.
- Ambos cambios se replican y aplican sin verificaciones de FK. Cada nodo converge en "203 → X" **y** "X eliminado": una referencia colgante que ninguna FK de nodo violó localmente.

Un `subscriber.epc_profile_id` colgante hace que el HSS no pueda construir datos de suscripción para ese suscriptor. El HSS entonces responde `DIAMETER_ERROR_USER_UNKNOWN (5001)` en ULR/AIR, aunque la fila del suscriptor existe y está habilitada. La ventana de esquema antes de datos (a continuación) hace que sea más probable, porque las transacciones omitidas pueden separar la creación/eliminación de un padre de la re-apuntación del hijo.

Detección (no imposición)

Imponer FKs en la ruta de aplicación detendría la malla ante cualquier anomalía, así que en su lugar **detectamos y alarmamos**. Cada nodo instala una función `SECURITY DEFINER`, `omnihss_fk_orphan_count()`, que cuenta cada FK de columna única cuyo valor apunta a un padre faltante en todas las tablas. `postgres_exporter` la selecciona, y Grafana alarma cuando es mayor que cero (ver Métricas).

Ejecuta la barrida manualmente en cualquier momento:

```
sudo -u postgres psql -d omnihss -tAc "SELECT
omnihss_fk_orphan_count();"   # 0 = limpio
```

Para localizar filas problemáticas para una referencia específica (ejemplo: suscriptores apuntando a un perfil EPC inexistente):

```
SELECT s.imsi, s.epc_profile_id
FROM subscriber s
LEFT JOIN epc_profile e ON e.id = s.epc_profile_id
WHERE s.epc_profile_id IS NOT NULL AND e.id IS NULL;
```

Remedia apuntando la fila a un padre válido (una `UPDATE` en una fila existente se replica limpiamente a través de la malla) o eliminándola.

Previniendo la deriva del esquema

El desencadenante raíz del incidente que motivó esta sección fue una **migración de esquema aplicada en un nodo mientras sus pares aún estaban en el viejo esquema**, luego nuevos datos escritos contra los nuevos campos. La fuente publicó filas que los pares del viejo esquema no pudieron aplicar, causando errores de aplicación y **transacciones omitidas**, y por lo tanto divergencia.

- **Migra toda la malla junta.** Aplica migraciones a cada nodo de OmniHSS antes de aprovisionar contra nuevos campos. No escribas datos de nuevo esquema mientras cualquier par esté atrasado.

- **Observa la deriva.** El nodo exporta `omnihss_schema_max_version` / `omnihss_schema_migration_count` por nodo. Compáralos a través de la malla. Un nodo atrasado respecto a sus pares es una migración medio aplicada.
- **Observa los contadores de errores.** Un aumento en `omnihss_subscription_error_apply_error_count` / `_sync_error_count` significa que el trabajador de aplicación está fallando (y posiblemente omitiendo). La alerta "Errores de Aplicación/Sincronización de Replicación de HSS" se activa en esto.

Eliminando un Par

Eliminar un par elimina la suscripción **de** ese par y quita su entrada de `pg_hba.conf`. Los datos ya replicados permanecen; solo los cambios futuros dejan de fluir. La eliminación debe ser confirmada explícitamente.

En el nodo del que estás eliminando el par *de*:

```
DROP SUBSCRIPTION IF EXISTS sub_from_<peer_name>;
```

Luego elimina la(s) línea(s) del par de `pg_hba.conf` y recarga PostgreSQL:

```
sudo -u postgres psql -c 'SELECT pg_reload_conf();'
```

El par que se está eliminando aún mantiene su suscripción hacia este nodo. Si el enlace debe ser completamente desmantelado, elimina esa suscripción allí también. Luego elimina el nodo de `connected_omnihss` y del grupo `hss` para que la próxima ejecución del rol no lo vuelva a crear.

Respaldo y Restauración

El respaldo y la restauración de OmniHSS utilizan volcado estándar de PostgreSQL:

Operación	Playbook	Notas
Respaldo	`services/backup.yml`	Produce `hss_dump_<host>_<ts>.sc` de datos) y `hss_<host>_<ts>.tar.gz` configuración de `/etc/omn` bajo `backups/<Site_Name>` volcado excluye publicaciones/suscripcione
Restaurar	`util_playbooks/restore_hss.yml`	Solicita el volcado SQL y/o de configuración. Detiene OmniHSS, toma un respal a la restauración, **desmar nodo de la malla** (elimin suscripciones/publicacione y desactiva las suscripcion pares hacia él), elimina/re base de datos, restaura, n reinicia. Se niega a apunta de un nodo a la vez.

Restaurar y la malla

Antes del borrado, la restauración elimina el nodo de la malla activa-activa. `DROP DATABASE` falla mientras exista una suscripción o una ranura lógica. Cuando la replicación está habilitada (`omnihss.replication.enabled`), la restauración reconstruye la malla en este orden:

1. **Publicar.** El playbook crea la publicación en el nodo restaurado. Aún no suscribe el nodo restaurado a sus pares.
2. **Reconectar y sembrar los pares locales.** El playbook recrea la suscripción de cada par local `hss` al nodo restaurado, con `copy_data = false`. Luego llena cada par vacío con `pg_dump --data-only --inserts --on-conflict-do-nothing`. Omite `schema_migrations`, `pdn_session` y

`lte_call`, como lo hace `seed_hss_replication.yml`. Un par que tiene datos mantiene sus datos y solo reanuda la transmisión.

3. **Suscribirse de nuevo.** El playbook ejecuta el `replication.yml` del rol. Esto crea las suscripciones del nodo restaurado a sus pares.

Haz el paso 3 solo después del paso 2. Si el nodo restaurado se suscribe a un par antes de la siembra, las filas de siembra se transmiten de nuevo a él. Esto causa errores de `clave duplicada` y un bucle de falla del trabajador de aplicación.

No uses `TRUNCATE` o `COPY` en un par. La replicación lógica replica `TRUNCATE`, así que borra el nodo restaurado y el resto de la malla.

Todos los pares locales deben ser accesibles. Si un par no es accesible, la restauración falla.

La restauración no siembra pares cruzados (`connected_omnihss`). Siémbralos con `seed_hss_replication.yml`. Después de una restauración, ejecuta `util_playbooks/audit_hss_drift.yml` para asegurarte de que la malla convergió.

Consulta Playbooks de Utilidad para el flujo de trabajo más amplio de respaldo/restauración.

Restaurando un nodo en una malla activa-activa

`restore_hss.yml` **desmantela la replicación en el nodo que restaura**: antes de eliminar y recrear la base de datos, elimina las suscripciones, publicaciones y ranuras de ese nodo. Así que después de una restauración, el nodo tiene **ningún objeto de replicación** — la malla debe ser reconfigurada, y reconfigurar solo al *par* no es suficiente (el nodo restaurado aún necesita su propia publicación de vuelta). Dos reglas:

1. **Restaura exactamente UN nodo.** Restaurar en dos o más nodos a la vez destruye la malla (`restore_hss.yml` advierte sobre esto). Restaura uno, luego siembra el resto desde él.

2. **Todos los nodos deben ejecutar la MISMA versión de OmniHSS antes de conectar la replicación.** La replicación lógica transporta datos, no esquema. `CREATE SUBSCRIPTION` verifica que **cada tabla en la publicación del publicador ya existe en el suscriptor**, así que un nodo en una versión más nueva (cuyo publicación incluye una tabla que una migración posterior agregó, por ejemplo, `ims_subscription`) no puede ser suscrito por un par que aún esté en el esquema más antiguo — la suscripción falla con `relation "public.<table>" does not exist` y esa dirección nunca se presenta. El esquema del **volcado** de la restauración también debe coincidir con la versión desplegada, o el nodo restaurado termina por delante de sus pares.

Reconstruir un par activo-activo desde cero:

```
# 1. Aprovisiona ambas VMs, luego instala OmniHSS + esquema en
AMBAS a la MISMA
#    versión (esquema idéntico, vacío y migrado en cada una).
ansible-playbook -i hosts/<inv>/host_files/<site>.yml omnihss.yml

# 2. Restaura datos en SOLO UN nodo (el esquema del volcado debe
coincidir con la versión del paso 1).
#    Esto elimina los objetos de replicación de ese nodo; el paso
3 reconstruye la malla.
ansible-playbook -i hosts/<inv>/host_files/<site>.yml \
  util_playbooks/restore_hss.yml --limit <site>-hss01

# 3. Siembra el/los otros nodo(s) Y conecta la malla bidireccional
completa — SIN --limit.
#    Los nodos vacíos se rellenan desde el nodo restaurado; las
publicaciones,
#    suscripciones y ranuras se (re)crean en cada nodo, luego se
validan.
ansible-playbook -i hosts/<inv>/host_files/<site>.yml \
  util_playbooks/seed_hss_replication.yml
```

No **termines** con `seed_hss_replication.yml --limit <peer>`: `--limit` también limita la conexión, así que solo crea la suscripción del par y deja al nodo restaurado sin su publicación — una malla medio conectada y unidireccional. `validate_replication.yml` (ejecutado automáticamente) ahora

afirma que cada nodo se suscribe a cada par, por lo que un resultado medio conectado falla ruidosamente en lugar de reportar saludable.

Métricas

Cada nodo de OmniHSS ejecuta `postgres_exporter`, exponiendo métricas de PostgreSQL (incluyendo replicación) a Prometheus. Series útiles para la salud de la replicación:

Métrica: `pg_replication_slots_active` **Tipo**: Medidor **Descripción**: Si cada ranura de replicación lógica tiene un consumidor activo. **Etiquetas**:

- `slot_name`: la ranura de replicación (una por par suscriptor)

Métrica: `pg_stat_replication_*` / retraso de ranura **Tipo**: Medidor **Descripción**: Retraso de posición de WAL por ranura. Muestra cuán atrás está un par consumidor.

Consultas de ejemplo:

```
# Cualquier ranura de replicación lógica inactiva (alerta si > 0)
count(pg_replication_slots_active == 0)

# Las suscripciones que deberían existir frente a los trabajadores
en ejecución se pueden verificar
# contra la salida del playbook de validación durante la
implementación.
```

Métricas personalizadas de OmniHSS

`postgres_exporter` en cada nodo también exporta series de salud de replicación e integridad específicas de OmniHSS a través de un archivo de consulta personalizado (`roles/omnihss/files/postgres_exporter_queries.yaml`). Prometheus las recopila bajo el trabajo `HSS Postgres`. Respaldan las reglas de alerta en el grupo **Alertas de HSS** (`roles/monitoring/templates/grafana/rules/`):

Métrica	Tipo	S
`omnihss_subscription_worker_alive{subname}`	Medidor	1 s tra ap er er
`omnihss_replication_slot_active{slot_name}`	Medidor	1 s es co la
`omnihss_replication_slot_wal_status_code{slot_name}`	Medidor	0= 1= 2= re **3=**
`omnihss_replication_slot_lag_bytes{slot_name}`	Medidor	By re pa ra
`omnihss_subscription_error_apply_error_count{subname}`	Contador	Er ac de de
`omnihss_subscription_error_sync_error_count{subname}`	Contador	Er ac de si de
`omnihss_integrity_fk_orphans`	Medidor	Fil cl

Métrica	Tipo	S
		co (d 0)
`omnihss_schema_max_version` / `omnihss_schema_migration_count`	Medidor	Ma ag m ap co

Consulta Monitoreo y Observabilidad para el pipeline de métricas y paneles.

Solución de Problemas

Nuevo nodo permanece vacío después de unirse

Síntomas: Los trabajadores de replicación están vivos y las ranuras están activas, pero el nuevo nodo no tiene (o pocos) suscriptores.

Causas posibles:

- El nodo se unió a una malla ya poblada y nunca fue sembrado. Las suscripciones utilizan `copy_data = false`, así que solo llegan cambios posteriores a la unión.

Resolución:

1. Siembra el nodo desde un par saludable. Ver Cargando un Nuevo Nodo con Datos.

Trabajador de suscripción no en ejecución

Síntomas: `pid IS NULL` para una suscripción en `pg_stat_subscription`.

Causas posibles:

- Par inalcanzable en el puerto de PostgreSQL (cortafuegos / enrutamiento).
- IP del par faltante en el `pg_hba.conf` de este nodo.
- Credenciales incorrectas en la cadena de conexión (`database_username` / `database_password` difieren entre nodos).

Resolución:

1. Confirma la accesibilidad TCP al par en `5432`.
2. Verifica que el par aparezca en `connected_omnihss` (o el grupo local `hss`) y vuelve a ejecutar el rol para que `pg_hba.conf` se actualice.
3. Asegúrate de que `database_username`/`database_password` sean idénticos en todos los nodos.

La ranura de replicación muestra `wal_status = lost`

Síntomas: El `wal_status` de una ranura es `lost`; el par correspondiente está muy atrasado.

Causas posibles:

- El par estuvo caído o inalcanzable el tiempo suficiente para que se eliminara el WAL requerido.

Resolución:

1. Re-siembra el par afectado desde un nodo saludable. Ver Cargando un Nuevo Nodo con Datos.

Errores de clave duplicada durante la copia inicial

Síntomas: Errores al aplicar cambios justo después de crear una suscripción con `copy_data = true`.

Causas posibles:

- La copia inicial se ejecutó desde más de un par, copiando las mismas filas dos veces.

Resolución:

1. Siembra desde **exactamente un** par con `copy_data = true`; mantén todas las demás suscripciones en `copy_data = false`.

Actualizaciones de estado en conflicto entre sitios

Síntomas: Una fila de estado (por ejemplo, `subscriber_state`) parece "fluctuar" entre valores establecidos en diferentes sitios.

Causas posibles:

- Actualizaciones concurrentes a la misma fila en dos sitios.

Resolución: Esto es esperado y manejado. El desencadenador de último escritor gana mantiene la actualización con el `updated_at` más reciente. Confirma que los relojes estén sincronizados (NTP) a través de todos los sitios para que el orden de `updated_at` tenga sentido.

Implementación de VM/LXC en Proxmox

La mayoría de nuestros clientes ejecutan la pila de OmniCore en Proxmox. Un solo playbook, `util_playbooks/proxmox.yml`, aprovisiona tanto VMs QEMU como contenedores LXC basándose en un interruptor `deployment_type`. Puedes establecer el interruptor por sitio (en la entrada `proxmoxServers`) y anularlo por host. Esto permite que un sitio funcione principalmente como LXC mientras que hosts individuales (por ejemplo, el UPF) se mantengan como VMs completas.

Todavía soportamos VMware, HyperV y la nube (Vultr / AWS / GCP) para implementaciones.

Ver También:

- Configuración del Archivo de Hosts: define VMs/LXCs a desplegar
- Estándar de Planificación IP
- Configuración de Netplan
- Arquitectura de Implementación

LXC vs VM

Contenedores LXC:

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

VMs (KVM):

- Virtualización completa con núcleo dedicado
- Aislamiento completo, ejecuta módulos del núcleo / redes personalizadas
- Mayor consumo 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, dns)

Elegir el Tipo de Implementación

`deployment_type` se resuelve para cada host en este orden:

1. **Por-host** `deployment_type` (establecido bajo el nombre del host): gana si está presente.
2. **Por-sitio** `deployment_type` en la entrada `proxmoxServers` que el round-robin selecciona para el host.
3. **Por defecto** `vm` si ninguno está establecido.

El patrón común es establecer el valor por defecto del sitio en la entrada `proxmoxServers` (por ejemplo, `deployment_type: lxc`) y anular solo los hosts que deben diferir. El ejemplo canónico es un laboratorio que ejecuta todo como LXC excepto el UPF, que necesita módulos del núcleo / TUN y debe ser una VM:

```yaml
all:
  vars:
    proxmoxServers:
      pve-node-01:
        deployment_type: lxc     # valor por defecto del sitio → todo es LXC
        # ...

upf:
  hosts:
    site-upf01:
      ansible_host: 192.168.1.24
      deployment_type: vm         # anulación por-host → este host es una VM
```

Como convención, mantén el valor por defecto del sitio consistente y solo anula las excepciones. Cuando distribuyes VM y LXC a través de muchos hosts, la colocación de nodos depende del índice del host y es difícil de razonar.

El playbook solo falla de forma crítica en colisiones de tipo con recursos existentes: si un recurso con el nombre de host objetivo ya existe como el tipo incorrecto (por ejemplo, un LXC existe pero el host se resuelve a `deployment_type: vm`, o viceversa), aborta con un error claro. Ver Guardias de Tipo.

Configuración de Proxmox

1. Crear Token de API

```
# Interfaz de Proxmox: Centro de datos → Permisos → Tokens de API
# Crear token: root@pam!<TokenName>
# Copia el secreto del token (se muestra una vez)
# Desmarca la casilla `Separación de Privilegios`
```

2a. Crear Plantilla de VM Cloud-Init (solo sitios de VM)

Ejecuta este script en el host de Proxmox. Descarga la imagen de nube de Ubuntu y crea una plantilla limpia. `proxmox.yml` inyecta credenciales de cloud-init y claves SSH en el momento de clonar (ver Notas a continuación).

```
#!/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"

cd /var/lib/vz/template/iso
wget -N "$IMAGE_URL"

qm destroy $TEMPLATE_ID --purge 2>/dev/null || true

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 template $TEMPLATE_ID
```

Notas:

- La plantilla no tiene usuarios o contraseñas codificados. `proxmox.yml` establece `ciuser`, `cipassword` y `sshkeys` en el momento de clonar.
- Precedencia del usuario de cloud-init: `proxmoxTemplateUser` (en la entrada `proxmoxServers` correspondiente) → primera clave `local_users`.
- Precedencia de la contraseña de cloud-init: `proxmoxTemplatePassword` (en la entrada `proxmoxServers` correspondiente) → el usuario de cloud-init (es decir, el valor elegido arriba).

- Las claves SSH se inyectan desde cada entrada `local_users.*.public_key` (la autenticaci��n basada en clave es el camino de inicio de sesión previsto).
- Establece `proxmoxTemplateUser` (y opcionalmente `proxmoxTemplatePassword`) en cada entrada `proxmoxServers` siempre que la plantilla se haya preparado con un usuario de cloud-init que no sea la primera entrada `local_users`, para que se seleccione la cuenta correcta.
- `--vga std` asegura que la consola web de Proxmox funcione.
- La bandera `-N` en wget solo descarga si es más reciente que la copia local.

Alternativa: Plantilla Manual desde ISO

Si las imágenes de nube no están disponibles o necesitas 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 el 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)
- Hardware → Opciones → Agente QEMU → Habilitar
- Hacer clic derecho en la VM → Convertir a Plantilla
- No establecer usuario/contraseña de cloud-init en la plantilla. `proxmox.yml` inyecta estos en el momento de clonar desde `proxmoxTemplateUser` / `proxmoxTemplatePassword` (con `local_users` como respaldo para el nombre de usuario).

2b. Descargar Plantilla de OS LXC (solo sitios de LXC)

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

La ruta del volumen resultante (por ejemplo, `local:vztmpl/ubuntu-24.04-standard_24.04-2_amd64.tar.zst`) es lo que configuras como `proxmoxLxcOsTemplate` a continuación.

Configuración del Archivo de Hosts

Implementación de VM

```yaml
all:
  vars:
    # deployment_type omitido → por defecto es "vm"
    proxmoxServers:
      pve-node-01:
        proxmoxServerAddress: 192.168.1.100
        proxmoxServerPort: 8006
        proxmoxApiTokenName: ansible
        proxmoxApiTokenSecret: "your-token-secret-uuid"
        proxmoxNodeName: pve-node-01
        proxmoxTemplateName: ubuntu-2404-template
        proxmoxTemplateId: 9000
        proxmoxTemplateUser: omnitouch        # nombre de usuario de cloud-init opcional; por defecto es la primera clave local_users
        proxmoxTemplatePassword: omnitouch    # contraseña de cloud-init opcional; por defecto es el nombre de usuario de cloud-init
        storage: local-lvm  # opcional
      pve-node-02:
        # ... configuración del segundo nodo

    local_users:
      admin_user:
        name: Admin User
        public_key: "ssh-rsa AAAA..."

mme:
  hosts:
    site-mme01:
      ansible_host: 192.168.1.10
  vars:
    proxmox_interface: vmbr0
    gateway: 192.168.1.1
    mask_cidr: 24  # opcional, por defecto 24
    vlanid: 100  # opcional
```

Implementación de LXC

```yaml
all:
  vars:
    proxmox_lxc_nameserver: "1.1.1.1"   # opcional, por defecto
1.1.1.1
    proxmoxServers:
      pve-node-01:
        deployment_type: lxc
        proxmoxServerAddress: 192.168.1.100
        proxmoxServerPort: 8006
        proxmoxApiTokenName: ansible
        proxmoxApiTokenSecret: "your-token-secret-uuid"
        proxmoxNodeName: pve-node-01
        proxmoxLxcOsTemplate: "local:vztmpl/ubuntu-24.04-
standard_24.04-2_amd64.tar.zst"
        proxmoxLxcDefaultStorage: local-lvm  # opcional, respaldo
de rootfs
      pve-node-02:
        deployment_type: lxc
        # ... misma forma

    local_users:
      admin_user:
        name: Admin User
        public_key: "ssh-rsa AAAA..."

dns:
  hosts:
    site-dns01:
      ansible_host: 192.168.1.20
  vars:
    proxmox_interface: vmbr0
    gateway: 192.168.1.1
    mask_cidr: 24                        # opcional, por defecto
24
    vlanid: 100                         # opcional
    proxmoxLxcCores: 2                  # anulación opcional
    proxmoxLxcMemoryMb: 4096            # anulación opcional
    proxmoxLxcDiskSizeGb: 30            # anulación opcional
    proxmoxLxcRootFsStorageName: local-lvm  # opcional, anula el
almacenamiento por defecto del servidor proxmoxLxcDefaultStorage
```

Uso

Aprovisionar (VM o LXC)

```
ansible-playbook -i hosts/Customer/site.yml
util_playbooks/proxmox.yml
```

Eliminar

```
ansible-playbook -i hosts/Customer/site.yml
util_playbooks/proxmox_delete.yml
```

El playbook de eliminación maneja automáticamente tanto VMs como LXCs, independientemente de `deployment_type`.

Comportamiento del Playbook

Guardias de Tipo

- Si existe una VM con el nombre de host objetivo pero `deployment_type: lxc` → el playbook falla con un error claro.
- Caso inverso (LXC existe, sitio configurado como VM) → lo mismo.
- Grupo UPF en un sitio LXC → solo advertencia (no es un fallo crítico, pero UPF no funcionará).

Nuevo Recurso

- Se distribuye de manera round-robin a través de las entradas `proxmoxServers` por índice de host dentro del grupo, a menos que el host establezca `proxmox_server:` para fijar una entrada específica (ver Distribución a Través de Nodos).
- Consulta la API de Proxmox para el siguiente VMID libre.

- **VM**: clona desde `proxmoxTemplateId`, establece usuario/contraseña/ claves SSH de cloud-init (usuario de cloud-init = `proxmoxTemplateUser` de la entrada del servidor si está establecido, de lo contrario la primera clave `local_users`; contraseña de cloud-init = `proxmoxTemplatePassword` si está establecida, de lo contrario el nombre de usuario de cloud-init), redimensiona `scsi0`, inicia.
- **LXC**: crea desde `proxmoxLxcOsTemplate`, rootfs en `proxmoxLxcRootFsStorageName` (grupo) → `proxmoxLxcDefaultStorage` (servidor) → `storage` → `local-lvm` (respaldo final), inyecta todas las `local_users.*.public_key` a través de `ssh-public-keys`, inicia. La contraseña raíz está codificada como `0mn1Touch@`. Todos los LXCs son no privilegiados (`unprivileged: 1`) con anidamiento habilitado (`features: nesting=1`).

Recurso Existente

Tanto las rutas de actualización de VM como de LXC difieren entre el actual y el deseado y aplican cambios:

- Configuración de red (puente, etiqueta VLAN, IP para LXC)
- Núcleos / memoria
- Tamaño del disco (VM `scsi0` / LXC `rootfs`)
- NICs secundarias de `secondary_ips` Reinicia el recurso si la red o los recursos cambiaron.

> **OmniUPF:** la herramienta trata una entrada `secondary_ips` llamada `n3` (o `data`/`gtp`) como una NIC de plano de usuario dedicada. El rol `omniupf` vincula N3/N9 + XDP a ella y reserva una CPU para la interfaz principal, de modo que una carga pesada de datapath no pueda agotar PFCP/API/SSH (que permanecen en la IP principal).

Etiquetado

Cada recurso se etiqueta con sus nombres de grupo de Ansible + `site_name`.

Distribución a Través de Nodos

Por defecto, la herramienta distribuye hosts a través de las entradas `proxmoxServers` de manera round-robin, con la misma lógica para VMs y LXCs. El índice es la posición del host **dentro de su grupo de rol**, tomado módulo el número de entradas `proxmoxServers` (en orden de declaración):

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

> **Agregar un segundo nodo redistribuye todo.** Debido a que la colocación es posicional, pasar de una entrada `proxmoxServers` a dos cambia dónde aterrizan los grupos multi-host existentes (host-0 → entrada-0, host-1 → entrada-1, ...). Fija los hosts que no deseas mover (a continuación) antes de volver a ejecutar contra una flota mixta.

Fijar un host a un nodo específico

Establece la clave opcional `proxmox_server:` al **nombre de una entrada `proxmoxServers`** para forzar un host en ese nodo independientemente de su índice de grupo. Cuando no está establecida, se aplica el round-robin anterior. Un valor que no es una clave de `proxmoxServers` falla la ejecución con un error que lista los nombres válidos (por lo que un error tipográfico no puede redirigir silenciosamente a otro lugar).

```yaml
device_tester:
  hosts:
    site-device-tester01:
      ansible_host: 10.179.2.50
      proxmox_server: nicklab-prxmx01   # fijar a esta entrada de
proxmoxServers (→ su nodo)
      # Si el almacenamiento de disco de ese nodo difiere del
global proxmoxLxcRootFsStorageName,
      # anúlalo aquí también (el almacenamiento se lee
globalmente, no desde la entrada del servidor):
      proxmoxLxcRootFsStorageName: local-lvm
```

`proxmox_server` también se puede establecer en el ámbito de `vars:` del grupo para fijar cada host en un grupo a un nodo.

Anulaciones por Host

`deployment_type`, `proxmox_server` (fijación de nodo: ver Distribución a Través de Nodos), y cualquiera de los controles de nivel de grupo de LXC (`proxmoxLxcCores`, `proxmoxLxcMemoryMb`, `proxmoxLxcDiskSizeGb`, `proxmoxLxcRootFsStorageName`, `proxmoxLxcOsTemplate`, `host_vm_network`) se pueden establecer por host bajo el nombre del host.

```yaml
dns:
  hosts:
    site-dns01:
      ansible_host: 192.168.1.20
      deployment_type: vm         # anulación por-host del valor
por defecto del sitio
      proxmoxLxcDiskSizeGb: 120   # anulación por-host
      host_vm_network: vmbr1      # anulación por-host del puente
  vars:
    proxmox_interface: vmbr0
    gateway: 192.168.1.1
    mask_cidr: 24  # opcional, por defecto 24
```

Registros de Ejecución del Playbook

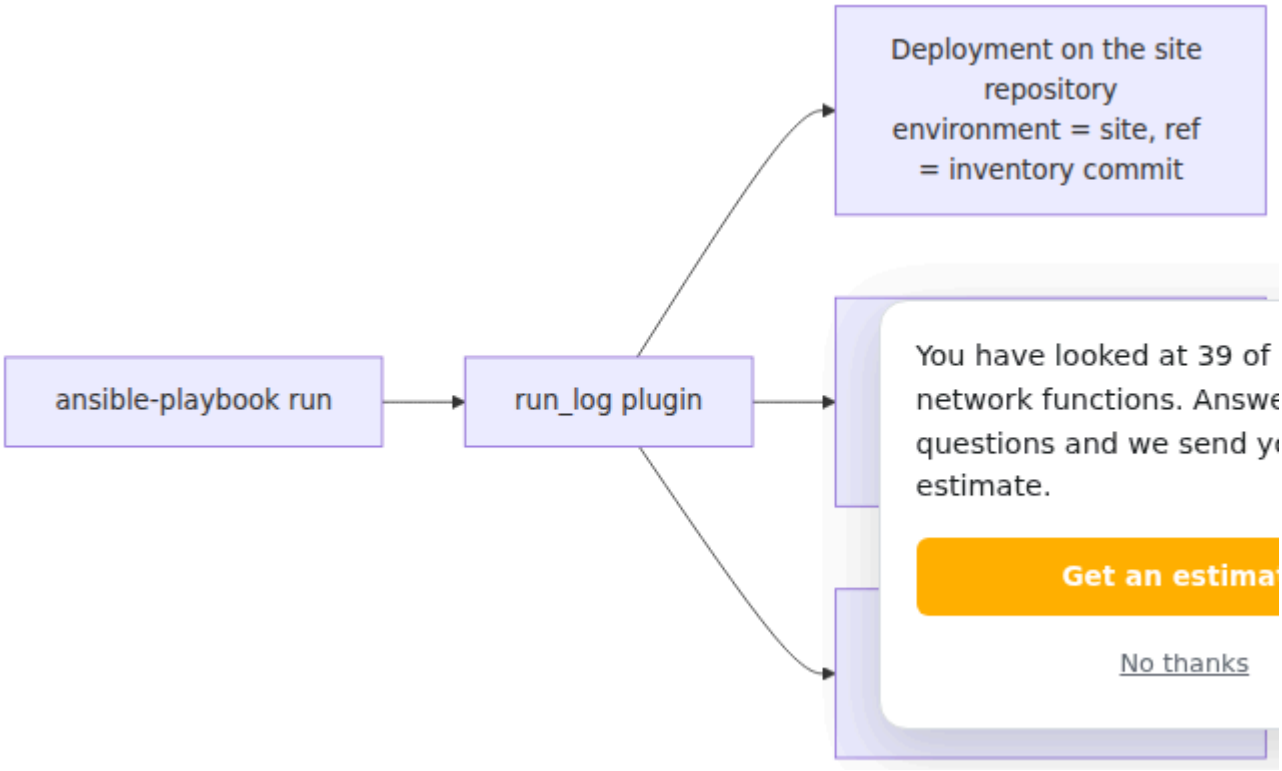
El plugin de callback `run_log` registra cada ejecución de `ansible-playbook` en GitHub. Está habilitado en `ansible.cfg` en la copia de git y en el kit de herramientas instalado con apt. El operador ejecuta los playbooks como antes:

```
ansible-playbook -i hosts/YourCustomer/host_files/production.yml
services/firewall.yml
```

Al final de la ejecución, el plugin imprime el directorio local de ejecución, el Despliegue y el enlace del registro.

Qué se registra

El registro se almacena en el repositorio del sitio: el `origin` de la copia de git que contiene el archivo de inventario proporcionado con `-i`. El nombre del sitio es el nombre del archivo de inventario sin su extensión.



Registro	Contenido
Despliegue	Entorno = site. Ref = el commit del inventario. Estado = `success` o `failure`. Carga útil = el comando, operador, máquina, hora de inicio, duración, resultado, hosts fallidos, versión de Ansible, y la versión de ambas copias: rama y commit, o la versión del paquete `omnicore` de una instalación apt, con el número de archivos no confirmados.
Comentario del commit	El registro completo de la ejecución, el resumen del play, y los cambios no confirmados de ambas copias. El estado del Despliegue se vincula a esto. Un registro largo se divide en varios comentarios.
Mensaje de Slack	El resultado, operador, comando, ambos commits, y enlaces al Despliegue, al commit y al registro. Publicado al primer receptor de `slack` en `monitoring_data.contactPoints`, el punto de contacto a través del cual Grafana envía alertas.
Copia local	`~/.local/state/omnicore/runs/<id>/`: `ansible.log`, `run.json`, `playbooks.diff`, `inventory.diff`.

Requisitos

- El archivo de inventario está en una copia de git cuyo `origin` está en GitHub, y el commit del inventario está enviado.
- `gh` (GitHub CLI) está instalado y ha iniciado sesión con acceso al repositorio del sitio.
- `log_path` está configurado en `ansible.cfg`.

Si no se cumple un requisito, el plugin imprime un error `run_log:` al final de la ejecución. El resultado del playbook no cambia.

Secretos del Cliente

Cada inventario de cliente tiene un archivo encriptado para los valores que no deben almacenarse en texto claro: claves de API, contraseñas y tokens. El archivo es `hosts/<customer>/group_vars/secrets.yml`. Está encriptado con Ansible Vault. Los roles que necesitan un secreto cargan este archivo en tiempo de ejecución. El texto plano nunca entra en Git.

Cómo Funciona



1. Un operador agrega un secreto al archivo con `scripts/vault_secrets.py`. La herramienta encripta el archivo con `ansible-vault`.
2. Un rol que necesita un secreto incluye el rol `common/secrets`. El rol desencripta el archivo una vez por host y carga cada nombre de nivel superior como una variable ordinaria.
3. El rol verifica que los nombres que necesita estén presentes. Si falta un nombre, la ejecución falla con el comando exacto que lo crea.

El archivo solo se desencripta en el controlador de Ansible. Solo la configuración renderizada llega a un host objetivo.

Formato del Archivo

El archivo es un mapeo YAML de nombres a valores. Un valor es una cadena o un mapeo de cadenas.

```
omnihss_api_keys:
  omniweb: "<64 hex characters>"
  omnihlr: "<64 hex characters>"
  omnicrm: "<64 hex characters>"
  omniudr: "<64 hex characters>"
  omnieir: "<64 hex characters>"
  ran_simulator: "<64 hex characters>"
```

Nombre	Tipo	Requerido por	Descripción
`omnihss_api_keys`	Mapeo	`omnihss_api_auth: rbac` (el predeterminado)	Una clave de API REST de OmniHSS por cliente. Cada clave es un secreto aleatorio de al menos 32 caracteres. Dos clientes no deben compartir una clave. Ver Claves de API de OmniHSS.

Otros roles documentan los nombres que leen en sus propias secciones de esta guía a medida que adoptan el archivo.

Gestión de Secretos

Utilice `scripts/vault_secrets.py` para cada cambio. No edite el archivo con `ansible-vault edit`: la herramienta mantiene el texto plano en un pipe y nunca lo escribe en disco.

```
scripts/vault_secrets.py hosts/<customer> list [opciones de vault]
scripts/vault_secrets.py hosts/<customer> set NAME [--random-hex
N] [--map K1,K2,...] [--force] [opciones de vault]
```

Argumento	Descripción
`hosts/<customer>`	El directorio del inventario del cliente. Debe contener `host_files/`.
`list`	Imprimir los nombres de los secretos. Para un mapeo, imprima también sus claves. Nunca imprima un valor.
`set NAME`	Agregar `NAME` al archivo. Cree el archivo si no existe.
`--random-hex N`	Establecer el valor en una cadena aleatoria de `N` bytes, escrita como `2N` caracteres hexadecimales. `N` debe ser al menos 16.
`--map K1,K2,...`	Con `--random-hex`, establecer el valor en un mapeo con una cadena aleatoria por clave.
`--force`	Reemplazar un `NAME` que ya existe. Sin él, la herramienta se niega.
opciones de vault	Pasadas a `ansible-vault`, por ejemplo `--ask-vault-pass` o `--vault-password-file FILE`.

Si no se proporciona `--random-hex`, la herramienta lee una línea de la entrada estándar y la utiliza como valor.

Con `--ask-vault-pass`, la herramienta pide la contraseña una vez. Cuando crea un nuevo archivo, pide dos veces para confirmar.

Ejemplos

Crear el archivo y las claves de API de OmniHSS para un nuevo cliente:

```
scripts/vault_secrets.py hosts/ExampleCustomer set
omnihss_api_keys --random-hex 32 \
    --map omniweb,omnihlr,omnicrm,omniudr,omnieir,ran_simulator --
ask-vault-pass
```

Agregar un secreto de un valor que ya tiene:

```
echo 'the-api-key' | scripts/vault_secrets.py
hosts/ExampleCustomer set example_api_key --ask-vault-pass
```

Ver qué secretos tiene un cliente:

```
scripts/vault_secrets.py hosts/ExampleCustomer list --ask-vault-
pass
```

Rotar las claves de API de OmniHSS:

```
scripts/vault_secrets.py hosts/ExampleCustomer set
omnihss_api_keys --random-hex 32 \
    --map omniweb,omnihlr,omnicrm,omniudr,omnieir,ran_simulator --
force --ask-vault-pass
```

Luego ejecute los playbooks para el HSS y para cada cliente API, para que todos ellos obtengan las nuevas claves.

Ejecución de Playbooks

Cada playbook que llega a un rol con un secreto necesita la contraseña del vault. Proporciónela de una de dos maneras:

Método	Comando	Uso
Solicitud	`ansible-playbook ... --ask-vault-pass`	Ejecuciones interactivas
Archivo de contraseña	`ansible-playbook ... --vault-password-file ~/.ansible/<customer>_vault_pass`	Ejecuciones no atendidas, CI

La contraseña del vault para cada cliente es la entrada de OmniCore en el gestor de contraseñas compartido. Mantenga un archivo de contraseña fuera del repositorio con modo `0600`.

Confirme `secrets.yml` en el repositorio del inventario del cliente. Es seguro confirmar: el archivo es texto cifrado. Verifique esto antes de enviar:

```
head -c 14 hosts/ExampleCustomer/group_vars/secrets.yml   #
imprime $ANSIBLE_VAULT
```

Claves de API de OmniHSS

OmniHSS protege su API REST con una clave por cliente. Cada cliente tiene un rol fijo de OmniHSS, por lo que una clave robada solo otorga los derechos de ese cliente.

Clave	Cliente	Rol de OmniHSS
`omniweb`	OmniWeb	`oam`
`omnihlr`	Interoperabilidad de HLR de OmniSS7	`hlr_interworking`
`omnicrm`	OmniCRM	`crm_provisioning`
`omniudr`	OmniUDR (núcleo 5G)	`udr`
`omnieir`	OmniEIR (núcleo 5G)	`eir_lookup`
`ran_simulator`	Simulador RAN	`ran_simulator`

El rol `omnihss` renderiza las seis claves en la configuración del HSS. Cada rol de cliente renderiza solo su propia clave. Una clave debe tener al menos 32 caracteres, y no dos clientes pueden compartir una clave. La ejecución falla si se rompe alguna de estas reglas.

Parámetro	Tipo	Requerido	Predeterminado	Descri
`omnihss_api_auth`	Cadena	No	`rbac`	`rbac`: cad envía su c `omnihss_a` y `secrets` requerido. `disabled` OmniHSS cada llam una clave necesita a secretos. Establezca el archivo bajo `all.`

Solución de Problemas

La ejecución falla con "debe existir y estar encriptado con ansible-vault"

Síntomas: Un rol se detiene en "Afirmar que el archivo de secretos existe y está encriptado en vault".

Causas posibles:

- El cliente no tiene `group_vars/secrets.yml` aún.
- El archivo fue escrito en texto plano.

Resolución:

1. Ejecute el comando `set` que se muestra en el mensaje de error. Esto crea el archivo encriptado.
2. Si existe un archivo en texto plano, elimínelo. Luego, vuelva a crearlo con la herramienta.

La ejecución falla con "No se pudo desencriptar"

Síntomas: Un rol se detiene en "Explicar la falla de carga de secretos".

Causas posibles:

- El playbook se ejecutó sin `--ask-vault-pass` o `--vault-password-file`.
- La contraseña no coincide con la entrada de vault de OmniCore para este cliente.

Resolución:

1. Ejecute el playbook nuevamente con la contraseña del vault.
2. Verifique la contraseña con `ansible-vault view hosts/<customer>/group_vars/secrets.yml > /dev/null && echo VAULT-OK`.

Un cliente recibe 401 de la API de OmniHSS

Síntomas: OmniWeb, OmniCRM u otro cliente registra `401 Unauthorized` desde el HSS después de una rotación de claves.

Causas posibles:

- El HSS obtuvo las nuevas claves pero el cliente no, o viceversa.

Resolución:

1. Ejecute el playbook del HSS y el playbook del cliente con el mismo `secrets.yml`.
2. Confirme que ambos hosts reiniciaron su servicio.

Documentación Relacionada

- Configuración de Variables de Grupo - Otros archivos por cliente bajo `group_vars/`
- Referencia de Configuración - Parámetros de host

- Resumen

Libretos de Servicio

Los libretos de servicio despliegan y configuran la infraestructura de OmniCore. Se encuentran en el directorio `services/`.

Jerarquía de Libretos

Los libretos están estructurados jerárquicamente para permitir despliegues rápidos y específicos:

```
all.yml
├── setup_users.yml
├── apt_cache.yml          # solo cuando se define el grupo
apt_cache_servers
├── dns.yml
├── common.yml
├── license_server.yml
├── monitoring.yml
│   └── grafana.yml
├── device_tester.yml
├── epc.yml
│   ├── common.yml
│   ├── omnimme.yml
│   ├── omnisgwc.yml
│   ├── omnipgwc.yml
│   ├── upf.yml
│   ├── omnihss.yml
│   └── omnidra.yml
├── ims.yml
│   ├── pcscf.yml
│   ├── icscf.yml
│   ├── scscf.yml
│   ├── as.yml
│   ├── omnimessage.yml
│   └── omnisep.yml
├── omniepdg.yml
├── omnitwag.yml
├── omniss7.yml
├── ocs.yml
├── crm.yml
├── ran_monitor.yml
├── 5gc.yml
└── ../util_playbooks/health_check.yml
```

Ejecuta solo lo que necesitas. Desplegar un solo componente (por ejemplo, `omnimme.yml`) toma segundos. Ejecutar `all.yml` en 50 hosts toma más tiempo pero maneja todo. Usa `--limit` para reducir aún más el alcance.

Referencia Rápida

Libretos de Nivel Superior

Libreto	Alcance	Descripción
`all.yml`	Red completa	Despliegue completo: usuarios, DNS, monitoreo, probador de dispositivos, EPC, IMS, ePDG, TWAG, SS7, OCS, CRM, monitor RAN, núcleo 5G y una verificación de salud final (desde `util_playbooks/`)
`epc.yml`	Núcleo de Paquetes	MME, SGW-C, PGW-C, UPF, HSS, DRA
`ims.yml`	Voz/IMS	P/I/S-CSCF, Servidor de Aplicaciones, Mensajería, OmniSEP (XCAP/Entitlements/BSF)
`ocs.yml`	Facturación	CGrateS, clúster KeyDB, sincronización OCS
`monitoring.yml`	Observabilidad	Prometheus, Grafana, exportadores

Libretos de Infraestructura

Libreto	Descripción
`common.yml`	Configuración base del sistema operativo, paquetes, NTP, agentes de registro
`setup_users.yml`	Cuentas de usuario locales y claves SSH
`dns.yml`	Despliegue del servidor DNS
`license_server.yml`	Servidor de licencias de OmniCore
`does_netplan_exist.yml`	Verifica si netplan está instalado en cada host
`update_mtu.yml`	Establece y aplica MTU de interfaz a través de netplan
`firewall.yml`	Reglas de iptables/nftables
`apt_cache.yml`	Espejo APT local (si está habilitado)

Componentes de EPC

Libreto	Componente	Descripción
`omnimme.yml`	OmniMME	Entidad de Gestión de Movilidad (4G)
`omnisgwc.yml`	OmniSGW-C	Plano de Control del Gateway de Servicio
`omnipgwc.yml`	OmniPGW-C	Plano de Control del Gateway de PDN
`upf.yml` / `omniupf.yml`	OmniUPF	Función de Plano de Usuario (SGW-U/PGW-U combinados)
`omnihss.yml` / `hss.yml`	OmniHSS	Servidor de Suscriptores Locales
`omnidra.yml`	OmniDRA	Agente de Enrutamiento Diameter
`omnitwag.yml`	OmniTWAG	Gateway de Acceso Inalámbrico de Confianza
`omniepdg.yml`	OmniEPDG	Gateway de Datos de Paquetes Evolucionados (Llamadas WiFi)
`gtp_proxy.yml`	Proxy GTP	Proxy de tráfico GTP

Componentes de IMS

Libreto	Componente	Descripción
`pcscf.yml`	P-CSCF	Función de Control de Sesiones de Llamadas Proxy
`icscf.yml`	I-CSCF	CSCF Interrogador
`scscf.yml`	S-CSCF	CSCF de Servicio
`as.yml`	OmniTAS	Servidor de Aplicaciones de Telefonía
`omnisep.yml`	OmniSEP	XCAP, Servidor de Derechos, BSF, Buzón de Voz Visual
`xcap.yml`	OmniSEP	Alias heredado de `omnisep.yml` (despliega el mismo rol `omnisep`)
`omnimessage.yml`	OmniMessage / IMS SMSC / MMSc	Controlador de SMS sobre IP, SMPP, el IMS SMSC (desplegado a través del rol `cscf` con `cscf_type: smsc_ims`), y el MMSc independiente (`grupo mmsc`, `rol mmsc`)

Núcleo 5G

Desplegado a través de `5gc.yml`. Cada función de red se instala mediante el rol compartido `5gc_nf` con un `nf_package` por NF. Consulta 5G Core para los detalles internos del rol, configuración por NF, fijación de versiones y claves HNET/SUCI de UDM.

Libreto	Componente	Descripción
`5gc.yml`	Núcleo 5G	Despliega todas las funciones de red 5G Autónomas
	OmniNRF	Función de Repositorio de Red
	OmniAMF	Función de Gestión de Acceso y Movilidad
	OmniAUSF	Función de Servidor de Autenticación
	OmniBSF	Función de Soporte de Vinculación
	OmniCHF	Función de Facturación
	OmniNSSF	Función de Selección de Cortes de Red
	OmniPCF	Función de Control de Políticas
	OmniSCP	Proxy de Comunicación de Servicio
	OmniSMF	Función de Gestión de Sesiones
	OmniUDM	Gestión Unificada de Datos
	OmniUDR	Repositorio Unificado de Datos

Conmutación de Circuitos

Desplegado a través de `cs.yml` (roles bajo `circuit_switched/`). Consulta Circuit-Switched Core para los detalles de los roles BSC, MGW, MSC y SS7.

Libreto	Componente	Descripción
`cs.yml`	Conmutación de Circuitos	Despliega el dominio CS heredado
	BSC	Controlador de Estación Base (`circuit_switched/bsc`)
	MGW	Gateway de Medios Independiente (`circuit_switched/mgw`)
	OmniMSC	Centro de Conmutación Móvil (`circuit_switched/omnimsc`)
	SS7	Pila SS7/SIGTRAN (`rol omniss7`)

Servicios de Soporte

Libreto	Descripción
`ocs.yml`	Sistema de Facturación en Línea (CGrateS + KeyDB)
`crm.yml`	Portal de gestión de clientes (OmniCRM; la configuración ahora se encuentra en `group_vars/`)
`omniss7.yml`	Gateway SS7/SIGTRAN
`voipmonitor.yml`	Captura de paquetes SIP/RTP (VoIPmonitor) en la caja de captura `siptrace`, además de recolectores de cliente delgados optativos
`grafana.yml`	Tableros y alertas de Grafana
`ran_monitor.yml`	Integración de monitoreo RAN
`ussd.yml`	Aplicación de callback USSD para el gateway USSD del HLR (`grupo ussd`). Consulta Circuit-Switched
`omniroam.yml`	Suite de roaming OmniRoam (TAP3/RAP UI, API de Elixir, generador)
`omnilcs.yml`	Servicios de Localización OmniLCS y Difusión de Celdas
`device_tester.yml`	Banco de pruebas del agente de teléfono OmniWeb (Dispositivos de Prueba)

> **Provisionamiento de Proxmox** (`proxmox.yml`, `proxmox_delete.yml`) se encuentra bajo `util_playbooks/`. Consulta Utility Playbooks y Despliegue de VM/LXC de Proxmox.

Operaciones

Libreto	Descripción
`backup.yml`	Respaldo de bases de datos y configuraciones
`reboot.yml`	Reinicio controlado de hosts
`shutdown.yml`	Apagado controlado
`apt_update.yml`	Actualizar paquetes
`apt_refresh_metadata.yml`	Actualizar metadatos de caché APT
`speedtest.yml`	Pruebas de rendimiento de red

Libretos de Restauración

Libreto	Descripción
`restore_applicationserver.yml`	Restaurar OmniTAS desde respaldo
`restore_smsc.yml`	Restaurar SMSC desde respaldo: Controlador OmniMessage, gateway SMPP y nodos IMS SMSc

> Nota: `health_check.yml`, `restore_hss.yml` y `restore_ocs.yml` **no** están en `services/`. Se encuentran en `util_playbooks/`. Consulta Utility Playbooks. `all.yml` ejecuta la verificación de salud como `../util_playbooks/health_check.yml`.

Uso

Desplegar Todo

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

Desplegar Subsistema Específico

```
# Solo el núcleo de paquetes
ansible-playbook -i hosts/customer/host_files/production.yml
services/epc.yml

# Solo IMS/voz
ansible-playbook -i hosts/customer/host_files/production.yml
services/ims.yml
```

Desplegar Componente Único

```
# Actualizar solo el MME
ansible-playbook -i hosts/customer/host_files/production.yml
services/omnimme.yml

# Actualizar solo monitoreo
ansible-playbook -i hosts/customer/host_files/production.yml
services/monitoring.yml
```

Limitar a Hosts Específicos

```
# Ejecutar all.yml pero solo en un host
ansible-playbook -i hosts/customer/host_files/production.yml
services/all.yml --limit mme01

# Ejecutar en múltiples hosts específicos
ansible-playbook -i hosts/customer/host_files/production.yml
services/all.yml --limit "mme01,hss01"

# Ejecutar en un grupo
ansible-playbook -i hosts/customer/host_files/production.yml
services/all.yml --limit mme
```

Documentación Relacionada

- Arquitectura de Despliegue - Flujo de trabajo general de despliegue
- Configuración del Archivo de Hosts - Definición de infraestructura
- Configuración de Variables de Grupo - Personalización
- Utility Playbooks - Herramientas operativas (verificación de salud, restauración, etc.)
- Monitoreo y Observabilidad - Grafana, Prometheus, alertas

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
`proxmox.yml`	Provisionar VMs y/o contenedores LXC en Proxmox (según `deployment_type`)
`proxmox_delete.yml`	Eliminar VMs/LXCs en Proxmox
`proxmox_restart_hung.yml`	Forzar el ciclo de energía de VMs/LXCs a través de la API de Proxmox cuando SSH no responde
`vmware.yml`	Provisionar VMs en VMware vSphere
`vmware_delete.yml`	Eliminar VMs en VMware vSphere
`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
`seed_hss_replication.yml`	Sembrar un nodo OmniHSS vacío desde un par poblado (carga de datos de replicación inicial)
`audit_hss_drift.yml`	Auditar la malla de OmniHSS para detectar desviaciones de datos (conteo de filas + resumen de contenido por tabla de referencia en todos los nodos); solo lectura
`reconcile_hss_drift.yml`	Conciliar la desviación de datos de la malla de OmniHSS llenando las filas faltantes en los nodos (ejecución en seco por defecto)

Libreta	Propósito
`restore_ocs.yml`	Restaurar KeyDB de OCS, MySQL y 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
`clear_mnesia.yml`	Limpiar la base de datos Mnesia para OmniSS7 / OmniMessage y reiniciar (destructivo)
`delete_license.yml`	Eliminar el archivo de licencia en el servidor de licencias y reiniciarlo

Provisionamiento de VM

Las libretas de provisionamiento de VM crean y gestionan máquinas virtuales en plataformas de hipervisor. Todas las libretas admiten `--limit` para dirigir hosts o grupos específicos.

Consulte Despliegue de Proxmox para obtener una configuración detallada.

Proxmox

Archivos: `util_playbooks/proxmox.yml`, `util_playbooks/proxmox_delete.yml`

`proxmox.yml` provisiona tanto VMs como contenedores LXC; si un host dado se convierte en una VM o un LXC se decide por su `deployment_type` resuelto (sobrescritura por host → entrada `proxmoxServers` → `vm` por defecto). Consulte Despliegue de Proxmox.

```
# Provisionar (VMs y/o contenedores LXC, según deployment_type)
ansible-playbook -i hosts/customer/host_files/production.yml
util_playbooks/proxmox.yml

# Provisionar solo un grupo específico
ansible-playbook -i hosts/customer/host_files/production.yml
util_playbooks/proxmox.yml --limit mme

# Eliminar VMs/LXCs
ansible-playbook -i hosts/customer/host_files/production.yml
util_playbooks/proxmox_delete.yml --limit old-host
```

Reiniciar Hosts Colgados de Proxmox

Archivo: `util_playbooks/proxmox_restart_hung.yml`

Verifica la capacidad de respuesta de SSH de cada host y fuerza el ciclo de energía de cualquier VM o contenedor LXC que no responda a través de la API de Proxmox. Las VMs de QEMU se detienen y se inician; los contenedores LXC se inician (o reinician si ya están en ejecución). Se omiten los hosts que responden a SSH.

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

# Restringir a un solo host o grupo
ansible-playbook -i hosts/customer/host_files/production.yml
util_playbooks/proxmox_restart_hung.yml --limit hss01
```

VMware vSphere

Archivos: `util_playbooks/vmware.yml`, `util_playbooks/vmware_delete.yml`

Ambas libretas incluyen los archivos de tarea por VM bajo `util_playbooks/vm/vmware/` (`create_individual_vm.yml` y `delete_individual_vm.yml`). `vmware_delete.yml` es un envoltorio delgado alrededor de la tarea de eliminación; `vmware.yml` además espera por SSH después de la provisión, luego elimina archivos de netplan obsoletos y aplica netplan en la nueva VM.

Requisitos previos: Se requiere que las dependencias de Python estén instaladas (`uv sync`).

Uso:

```
# Provisionar VMs
ansible-playbook -i hosts/customer/host_files/production.yml
util_playbooks/vmware.yml

# Provisionar solo un grupo específico
ansible-playbook -i hosts/customer/host_files/production.yml
util_playbooks/vmware.yml --limit mme

# Eliminar VMs
ansible-playbook -i hosts/customer/host_files/production.yml
util_playbooks/vmware_delete.yml --limit old-host
```

Variables Requeridas:

```yaml
all:
  vars:
    vcenter_ip: "vcenter.example.com"
    vcenter_username: "administrator@vsphere.local"
    vcenter_password: "password"
    vcenter_datacenter: "Datacenter"
    vcenter_folder: "OmniCore"
    vcenter_vm_template: "ubuntu-2404-template"
    vhosts:
      esxi-01:
        vcenter_cluster_ip: "192.168.1.10"
        vcenter_datastore: "datastore1"
```

Vultr

Archivos: `util_playbooks/vm/vultr/vmProvisionVultr.yml`, `util_playbooks/vm/vultr/vmTeardownVultr.yml`, `util_playbooks/vm/vultr/vmCommonSetup.yml`

Provisiona y desmantela instancias en Vultr a través de la API de Vultr. `vmProvisionVultr.yml` crea instancias, `vmTeardownVultr.yml` las elimina, y `vmCommonSetup.yml` realiza la configuración base posterior a la provisión (DNS, reinicio, etc.). Las tres son libretas de entrada de nivel superior en lugar de incluir tareas. Lee el token de API de la variable de entorno `VULTR_API_TOKEN`, y la libreta de provisión hace referencia a un `cloud-config.yaml` del mismo directorio, así que ejecútalas desde `util_playbooks/vm/vultr/`.

```
export VULTR_API_TOKEN=<token>
ansible-playbook -i hosts/customer/host_files/production.yml
util_playbooks/vm/vultr/vmProvisionVultr.yml
```

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 y estado de pares Diameter
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 desde archivos de copia de seguridad. Soporta restaurar solo la base de datos, solo la configuración, o ambas.

```
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	Contenido
Base de Datos	`hss_dump_<hostname>_<timestamp>.sql`	Volcado de la base de datos OmniHSS (por defecto `omnihss`)
Configuración	`hss_<hostname>_<timestamp>.tar.gz`	Archivo del directorio `/etc/omnihss`

El estado actual de HSS (base de datos y/o configuración) se respalda antes de que se ejecute la restauración.

Solicitudes

Solicitud	Requerido	Descripción
Ruta del volcado SQL de HSS	No	Ruta completa al archivo de volcado SQL de HSS (dejar vacío para omitir la restauración de la base de datos)
Ruta del archivo tar.gz de configuración de HSS	No	Ruta completa al archivo tar.gz de configuración de HSS (dejar vacío para omitir la restauración de la configuración)
Confirmación	Sí	Escriba `yes` para confirmar la operación de restauración en el/los host(s) listados

Semilla de Replicación de HSS

Archivo: `util_playbooks/seed_hss_replication.yml`

Realiza la carga inicial de datos única para un nodo de OmniHSS que se une a una malla de replicación activa-activa ya poblada. Un nuevo nodo comienza vacío (las suscripciones se crean con `copy_data = false`); esta libreta recrea la suscripción a un único par fuente elegido con `copy_data = true`, de modo que PostgreSQL copie los datos existentes y luego continúe transmitiendo sin interrupciones.

Se ejecuta en uno de dos modos:

Interactivo (por defecto): omitir las variables `seed_source_*`. La libreta detecta cuáles son los objetivos limitados que están vacíos. Luego sondea los pares válidos para aquellos que tienen datos, los lista con conteos de suscriptores y le solicita que elija una fuente. Solo siembra objetivos vacíos.

```
ansible-playbook -i hosts/customer/host_files/production.yml \
  util_playbooks/seed_hss_replication.yml \
  --limit site-hss01,site-hss02
```

```
Pares con datos disponibles para sembrar:
  1) dev-hss01 (10.4.10.142) - 838 suscriptores
Ingrese el NÚMERO de la fuente de la que sembrar
```

Explícito: pase la fuente directamente para omitir el menú. Esto **fuerza** una re-siembra incluso si el objetivo ya tiene datos (usar para recuperación).

```
ansible-playbook -i hosts/customer/host_files/production.yml \
  util_playbooks/seed_hss_replication.yml \
  --limit site-hss01 \
  -e "seed_source_name=dev-hss01 seed_source_host=10.4.10.142"
```

Parámetros

Parámetro	Requerido	Predeterminado	Descripció
`seed_source_name`	No	(solicitado)	Nombre de ho del inventario par fuente poblado. Se ut para nombrar suscripción (`sub_from_<na` y resolver su publicación (`<name>_pub`). Omitir para el modo interact
`seed_source_host`	No	(solicitado)	Dirección IP de par fuente, accesible en s puerto de PostgreSQL. O para el modo interactivo.
`seed_source_port`	No	puerto `omnihss.database_port` del objetivo (5432)	Puerto de PostgreSQL en par fuente.

> Tanto `seed_source_name` como `seed_source_host` deben ser proporcionados juntos para usar el modo explícito; no proporcionar ninguno activa el modo interactivo.

Comportamiento

1. Cuenta los suscriptores en cada objetivo limitado para determinar cuáles están vacíos.

2. **Modo interactivo:** sondea los pares válidos (grupo local `hss` + `connected_omnihss`) para aquellos que tienen datos, los lista y solicita una elección. **Modo explícito:** solicita `yes` para confirmar.
3. Asegura que la IP de la fuente esté permitida en el `pg_hba.conf` del objetivo.
4. Elimina y recrea la suscripción a la fuente `WITH (copy_data = false)` (nombre de `slot_name` por enlace de `<target>_from_<source>`). Transmite **todas** las tablas pero no realiza ninguna sincronización masiva de tablas.
5. Rellena las filas de la **tabla maestra** existente desde la fuente con una carga a prueba de conflictos (`pg_dump --data-only --inserts --on-conflict-do-nothing --disable-triggers`). `subscriber_state` **se incluye**: cada suscriptor referencia uno por FK, por lo que omitirlo huérfana a cada suscriptor rellenado. Las tablas de sesión transitorias (`pdn_session`, `lte_call`) **no** se rellenan.
6. Informa el conteo de suscriptores en el nodo sembrado.

> **Por qué se omiten las tablas de sesión:** son tablas activas; un `COPY` masivo de ellas en un sistema en vivo colisiona (clave duplicada) con escrituras concurrentes y provoca bucles de falla en el trabajador de sincronización de tablas. Se repoblan a partir de la replicación en streaming y el tráfico en vivo. `subscriber_state` es diferente: las filas solo se INSERTan en el momento de la provisión (1:1 con el suscriptor) y solo se ACTUALIZAN después, por lo que el relleno a prueba de conflictos es seguro. Sin él, los suscriptores del objetivo apuntan a filas de estado que nunca llegan, ya que la replicación lógica omite silenciosamente las ACTUALIZACIONES a filas faltantes.

> **Importante:** Siempre `--limit` a los nodos objetivo. En modo interactivo, los objetivos no vacíos se omiten automáticamente. Consulte Replicación de OmniHSS y Operaciones de Nodos para el flujo de trabajo completo.

Auditoría de Desviación de HSS

Archivos: `util_playbooks/audit_hss_drift.yml` (independiente) y `roles/omnihss/tasks/audit_drift.yml` (la implementación compartida).

Informa sobre **desviaciones de datos** en la malla activa-activa de OmniHSS. Las suscripciones de replicación lógica se crean con `copy_data = false`, por lo que los datos que divergieron antes del enlace de la malla (diferencias de tiempo de siembra) o a través de una escritura directa/no replicada nunca se concilian y nunca se alarman. La replicación muestra 0 de retraso mientras los datos difieren, y Prometheus por sitio no puede verlo porque la malla abarca dos sitios.

La auditoría se conecta a cada miembro de la malla (grupo local `hss` + pares `connected_omnihss`) y, para cada tabla de referencia/configuración, compara un conteo exacto de filas **y** un md5 independiente del orden de todo el contenido. También compara `schema_migrations` (conteo + versión máxima) para detectar desviaciones de esquema. Se excluyen las tablas de estado activas (`subscriber_state`, `pdn_session`, `lte_call`): difieren legítimamente bajo tráfico en vivo. Se incluye una **verificación de salud** estructural sobre la comparación entre nodos: `__orphaned_subscribers__`, el conteo por nodo de suscriptores cuya fila `subscriber_state` falta. Un conteo distinto de cero en **cualquier** nodo falla la auditoría incluso cuando cada nodo está de acuerdo. Una siembra que copia `subscriber` sin `subscriber_state` huérfana a cada nodo de manera idéntica, lo que la comparación pura entre nodos nunca puede ver. La comparación se centraliza en un ejecutor que alcanza el Postgres de cada par, por lo que una sola ejecución cubre toda la malla entre sitios desde cualquier inventario, y es segura para `--limit`.

Se ejecuta automáticamente después de cada despliegue de HSS. Al final del rol `omnihss` (cuando la replicación está habilitada) se ejecuta la misma auditoría como una verificación de consistencia posterior al despliegue, y por defecto **FALLA la ejecución** si se encuentra desviación, por lo que es imposible pasar por alto una malla rota durante un despliegue. No es interactiva (sin pausa): la auditoría debe ser segura bajo la estrategia `free` del despliegue agregado `services/all.yml`, donde una pausa generaría un error. En ese despliegue agregado, la auditoría en el rol se suprime y la auditoría se ejecuta una vez en una libreta **lineal** dedicada al final de `services/deploy_nfs.yml`. El comportamiento se controla mediante dos variables de rol:

Variable	Predeterminado	Efecto
`omnihss_audit_drift`	`true`	ejecutar la auditoría al final del rol
`omnihss_audit_fail_on_drift`	`true`	fallar la ejecución en caso de desviación; establecer `false` para solo informar (no interactivo cron/CI)

La auditoría es **solo lectura** de cualquier manera. La libreta independiente por defecto informa y luego **falla** en caso de desviación (para que pueda controlar cron/CI):

```
# bajo demanda
ansible-playbook -i hosts/<inv>/host_files/<site>.yml
util_playbooks/audit_hss_drift.yml
# solo informar, siempre pasar
ansible-playbook -i ... util_playbooks/audit_hss_drift.yml -e
audit_fail_on_drift=false
# deshabilitar la auditoría automática para un solo despliegue
ansible-playbook -i ... services/omnihss.yml -e
omnihss_audit_drift=false
```

Conciliación de Desviación de HSS

Archivo: `util_playbooks/reconcile_hss_drift.yml`

Devuelve las tablas de referencia divergentes a la sincronización haciendo que cada nodo contenga la **unión** de las filas: para cada tabla desviada, rellena las filas que un nodo está faltando de sus pares, utilizando el mismo primitivo a prueba de conflictos que la libreta de siembra (`pg_dump --data-only --inserts --on-conflict-do-nothing --disable-triggers`).

- **No destructivo**: las filas solo se INSERTan (saltando conflictos). No se elimina ni se actualiza nada. Resuelve "un nodo está faltando filas"; no resuelve "mismo clave primaria, diferentes valores" o elimina filas que no deberían existir. Esas necesitan una decisión humana.
- **Interactivo por defecto**: escanea, imprime el plan y **pregunta** (`yes` para proceder) antes de escribir; cualquier otra cosa aborta sin cambios. Use `-e dry_run=true` para solo mirar, o `-e assume_yes=true` para ejecuciones no interactivas.
- Las escrituras se ejecutan localmente en cada objetivo como superusuario de postgres (para `--disable-triggers`); los pares solo se leen, como usuario de la aplicación.
- **Unión simétrica, no autoritativa de producción**: ejecutarlo contra un sitio trae las filas de *ese* sitio a los otros. No trata a la producción como la maestra, por lo que ejecutarlo contra el inventario de Lab propagaría las filas de Lab hacia la producción. Si la producción debe ser la fuente de verdad, diríjalo desde el inventario de producción (o solicite el modo autoritativo).
- **Entre sitios:** un nodo solo puede ser un *objetivo* cuando está en la libreta, así que ejecute esto contra **cada** inventario de sitio para cubrir cada nodo. La reexploración al final informa cualquier cosa que aún necesite una ejecución.

```
# interactivo (por defecto): pregunta antes de escribir
ansible-playbook -i hosts/<inv>/host_files/<site>.yml
util_playbooks/reconcile_hss_drift.yml
# solo mirar, nunca escribe
ansible-playbook -i ... util_playbooks/reconcile_hss_drift.yml -e
dry_run=true
# aplicación no interactiva (CI)
ansible-playbook -i ... util_playbooks/reconcile_hss_drift.yml -e
assume_yes=true
```

> Ejecute `audit_hss_drift.yml` primero para ver qué se desvió, y nuevamente después de conciliar (contra ambos inventarios) para confirmar la convergencia. Consulte Replicación de OmniHSS y Operaciones de Nodos.

Restauración de OCS

Archivo: `util_playbooks/restore_ocs.yml`

Restaura OCS (CGrateS) desde archivos de copia de seguridad. Maneja la replicación de KeyDB multi-maestro restaurando en un nodo y permitiendo que la replicación se sincronice con otros.

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

Proceso de Restauración

1. Detiene CGrateS y KeyDB en todos los nodos de OCS
2. Limpia los archivos AOF para prevenir conflictos con los datos restaurados
3. Restaura KeyDB RDB al primer nodo, inicia KeyDB, permite que la replicación se sincronice
4. Restaura MySQL StoreDB (opcional)
5. Restaura la configuración de `/etc/cgrates` (opcional)
6. Inicia CGrateS en todos los nodos

Formatos de Archivos de Copia de Seguridad

Tipo	Patrón de Nombre de Archivo	Contenido
Base de Datos KeyDB	`keydb_dump_<hostname>_<timestamp>.rdb`	Instantánea RDB de KeyDB
Base de Datos MySQL	`cgrates_dump_<hostname>_<timestamp>.sql`	Volcado de MySQL de la base de datos `cgrates`
Configuración	`cgrates_<hostname>_<timestamp>.tar.gz`	Archivo del directorio `/etc/cgrates`

Solicitudes

Solicitud	Requerido	Descripción
Ruta del volcado RDB de KeyDB	Sí	Ruta completa al archivo de copia de seguridad RDB de KeyDB
Ruta del volcado SQL de MySQL	No	Ruta completa al volcado SQL de CGrateS (omitir para preservar el StoreDB actual)
Ruta del archivo tar.gz de configuración	No	Ruta completa a la copia de seguridad de configuración (omitir para preservar la configuración actual)

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 host (NICs primarias y secundarias)
- Resumen 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 de Sphinx

Datos Recopilados

Campo	Descripción
Nombre de Host	Nombre de host del inventario
IP	Dirección IP `ansible_host` del host
Versión 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.

Limpiar Base de Datos Mnesia

Archivo: `util_playbooks/clear_mnesia.yml`

Detiene el servicio, elimina su directorio de base de datos Mnesia y lo reinicia. Se ejecuta contra el grupo `ss7` (OmniSS7) y el grupo `omnimessage` (OmniMessage).

> **Advertencia:** Esto es destructivo. La base de datos Mnesia se elimina permanentemente y se reconstruye vacía al reiniciar.

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

# Limitar a un solo servicio o host
ansible-playbook -i hosts/customer/host_files/production.yml
util_playbooks/clear_mnesia.yml --limit ss7
```

Eliminar Licencia

Archivo: `util_playbooks/delete_license.yml`

Elimina `/etc/license_server/license.json` en el grupo `license_server` y reinicia el servidor de licencias.

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

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
```