5G Core (5GC)

O núcleo 5G Standalone (SA) é implantado pelo playbook `services/5gc.yml`. Um único papel compartilhado, `5gc_nf`, instala cada função de rede (NF). O papel é parametrizado por NF. Esta página explica esse papel, as NFs que ele constrói e a configuração que cada uma lê.

Visão Geral

Em vez de um papel por NF, o OmniCore usa **um papel (`5gc_nf`) para todas as onze NFs do 5GC**. O playbook invoca o papel uma vez por NF e passa uma variável `nf_package` que seleciona o pacote Debian, o serviço systemd e o template de configuração Jinja2. Isso mantém a lógica de instalação, versionamento de configuração, licenciamento e reinicialização idêntica em cada NF.

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

Funções de Rede

Cada NF tem seu próprio grupo de inventário, pacote, serviço systemd e template de configuração (`roles/5gc_nf/templates/<nf_package>.j2`).

NF	`nf_package`	Grupo de inventário	Template de configuração	Papel
Função de Repositório de Rede	`omninrf`	`nrf`	`omninrf.j2`	Descoberta d serviços / Registro de N
Gerenciamento de Acesso e Mobilidade	`omniamf`	`amf`	`omniamf.j2`	Sinalização N1/N2, regist gerenciamen de conexão
Servidor de Autenticação	`omniausf`	`ausf`	`omniausf.j2`	Autenticação 5G-AKA / EAP AKA'
Função de Suporte de Vinculação	`omnibsf`	`bsf`	`omnibsf.j2`	Vinculação P para uma sessão PDU
Função de Cobrança	`omnichf`	`chf`	`omnichf.j2`	Cobrança online/offline convergente
Seleção de Fatiamento de Rede	`omnissf`	`nssf`	`omnissf.j2`	NSSAI / seleç de fatiament
Função de Controle de Políticas	`omnipcf`	`pcf`	`omnipcf.j2`	Regras de política e Qo
Proxy de Comunicação de Serviço	`omniscp`	`scp`	`omniscp.j2`	Roteamento mensagens S / comunicaçõ indiretas

NF	`nf_package`	Grupo de inventário	Template de configuração	Papel
Função de Gerenciamento de Sessão	`omnismf`	`smf`	`omnismf.j2`	Controle de sessão PDU e UPF (N4)
Gerenciamento de Dados Unificado	`omniudm`	`udm`	`omniudm.j2`	Dados de assinatura, desocultação SUCI
Repositório de Dados Unificado	`omniudr`	`udr`	`omniudr.j2`	Armazenamento de dados para UDM/PCF

Como o papel `5gc_nf` funciona

Para cada NF, o papel executa a mesma sequência:

1. **Resolver `group_vars_omnicore`**: definido como `<inventory>/group_vars/`, para que as buscas de configuração sejam resolvidas em relação ao inventário ativo.
2. **Descobrir servidores de licença**: se `license_server_api_urls` não estiver definido, ele é derivado do grupo de inventário `license_server` (`https://<host>:8443/api`).
3. **Instalar pré-requisitos**: `lsb-release`, `curl`, `gpg`, `unzip`, `libglu1-mesa`, `libsctp1`.
4. **Instalar o pacote NF** do repositório APT do Omnitouch. Último por padrão; fixe com `nf_version` (veja Version pinning).
5. **Template da configuração de runtime (versionada)**: renderiza `templates/<nf_package>.j2` para `/etc/<nf_package>/runtime-<deb_version>.exs`.
6. **Symlink `runtime.exs`** → a configuração da versão ativa, e espelhe esse symlink em cada diretório `/opt/<nf_package>/releases/*`.

7. **Reiniciar a NF** (handler) sempre que a configuração ativa mudar.

Version pinning e rollback

A configuração é escrita por versão Debian (`runtime-<deb_version>.exs`) e `runtime.exs` é um symlink para a ativa. Configurações anteriores são preservadas no disco, então um rollback é apenas reinstalar a versão anterior do pacote:

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

Omitir `nf_version` para rastrear o pacote mais recente.

Chaves HNET do UDM (desocultação de SUCI)

O UDM (`nf_package: omniudm`) tem etapas extras para suportar a desocultação de SUCI (TS 33.501 §6.12). A desocultação recupera o SUPI do SUCI oculto que um UE apresenta no registro. Ele precisa das chaves **privadas** da Rede Domiciliar que emparelham com as chaves **públicas** da Rede Domiciliar nos USIMs.

- As chaves vivem em `group_vars/hnet_udm/` no controlador Ansible como `<curve>-<id>.key` (PEM), onde `<curve>` é `curve25519` (Perfil A, X25519) ou `secp256r1` (Perfil B, prime256v1), e `<id>` é o Identificador da Chave Pública da Rede Domiciliar.
- Se o diretório estiver **vazio**, uma chave padrão `curve25519-1.key` é gerada automaticamente para que a desocultação funcione imediatamente. Esta etapa é idempotente. Uma chave existente nunca é sobrescrita.
- Para cada `curve25519-*.key`, o valor público bruto correspondente de 32 bytes é exportado (hex) para um arquivo `.pub` para provisionamento do USIM. Os arquivos `.pub` permanecem no inventário; apenas os arquivos

`*.key` são implantados em `/etc/omniudm/hnet/` no UDM (modo `0600`, root).

> Uma chave gerada automaticamente é aleatória e apenas desoculta SUCIs de USIMs programados com sua chave pública **correspondente**. Pegue o `.pub` exportado e provisioná-lo como a Chave Pública da Rede Domiciliar no USIM. SIMs usando o esquema nulo não precisam de chave HNET e podem deixar o diretório vazio.

Detalhes completos e exemplos de layout de arquivos: Configuração de Variáveis de Grupo: Exemplo 8.

Documentação Relacionada

- Playbooks de Serviço - Todos os playbooks de serviço e a hierarquia de implantação
- Núcleo com Comutação de Circuito - O domínio CS legado (BSC, MSC, SS7)
- Configuração de Variáveis de Grupo - Templates de configuração por NF e chaves HNET
- Arquitetura de Implantação - Como os domínios se encaixam
- **Configuração do OmniCore**: https://docs.omnitouch.com.au/docs/repos/OmniCore - O que vai nos arquivos de configuração da NF

Padrão de Planejamento de IP do OmniCore

Visão Geral

Este documento descreve a abordagem padrão de planejamento de IP para implantações do OmniCore. A arquitetura requer **quatro sub-redes internas** para segmentar adequadamente as funções de rede para segurança, desempenho e clareza operacional.

Requisitos de Alocação de IP

Alocação Padrão: Quatro Sub-redes /24

Cada implantação do OmniCore requer quatro sub-redes distintas para a rede interna:

1. **Rede de Núcleo de Pacotes** - Primeira /24
2. **Rede de Sinalização** - Segunda /24
3. **Rede Interna IMS** - Terceira /24
4. **Rede Pública IMS** - Quarta /24

Importante: Estas são Recomendações, Não Requisitos

A alocação de sub-rede descrita neste documento é uma **prática recomendada** para organizar implantações do OmniCore. No entanto, a arquitetura é **completamente flexível**:

- **Todos os hosts em uma sub-rede**: Você pode colocar todos os componentes em uma única sub-rede se isso atender às suas necessidades de implantação
- **Cada tipo de host em sua própria sub-rede**: Você pode criar sub-redes separadas para cada tipo de componente (uma para MMEs, uma para HSS, etc.)
- **Agrupamentos personalizados**: Você pode organizar os hosts em qualquer estrutura de sub-rede que faça sentido para seus requisitos específicos
- **Misturar IPs internos e públicos**: Alguns hosts podem usar endereços internos (RFC 1918) enquanto outros usam IPs públicos, todos dentro da mesma implantação

A abordagem recomendada de quatro sub-redes fornece **isolamento de segurança, gerenciamento de tráfego e clareza operacional** ideais, razão pela qual sugerimos para implantações em produção. No entanto, você deve adaptar o plano de IP para se adequar à sua topologia de rede específica, espaço de endereços disponível e requisitos operacionais.

Detalhamento dos Segmentos de Rede

1. Rede de Núcleo de Pacotes (Primeira /24)

Propósito: Elementos do plano de usuário e do plano de controle central

Componentes:

- OmniMME (Entidade de Gerenciamento de Mobilidade)
- OmniSGW (Gateway de Serviço)
- OmniPGW-C (Plano de Controle do Gateway PDN)
- OmniUPF/PGW-U (Função do Plano de Usuário / Gateway PDN do Plano de Usuário)

Exemplo: `10.179.1.0/24`

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

---

2. Rede de Sinalização (Segunda /24)

Propósito: Funções de sinalização Diameter, política, cobrança e gerenciamento

Componentes:

- OmniHSS (Servidor de Assinante Residencial)
- OmniCharge OCS (Sistema de Cobrança Online)
- OmniHSS PCRF (Função de Regras de Política e Cobrança)
- OmniDRA DRA (Agente de Roteamento Diameter)
- Servidores DNS (DNS interno/infraestrutura - distinto do DNS voltado para o cliente no segmento Público IMS; ambos são intencionais)
- Servidores TAP3/CDR
- Monitoramento/OAM
- Captura SIP
- Servidor de Licença
- Monitor de RAN
- Omnitouch Warning Link CBC (Centro de Transmissão de Células) - se implantado

Exemplo: `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. Rede Interna IMS (Terceira /24)

Propósito: Sinalização e serviços centrais do IMS (sinalização SIP interna)

Componentes:

- OmniCSCF S-CSCF (Função de Controle de Sessão de Chamada de Serviço)
- OmniCSCF I-CSCF (Função de Controle de Sessão de Chamada de Interrogatório)
- OmniTAS (Servidor de Aplicação de Telefonia / Servidor de Aplicação)
- OmniMessage (Controlador de SMS, SMPP, IMS)

Exemplo: `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. Rede Pública IMS (Quarta /24)

Propósito: Serviços IMS voltados para o usuário, DNS voltado para o cliente e interconexão SS7/legada

Componentes:

- OmniCSCF P-CSCF (Função de Controle de Sessão de Chamada Proxy)
- Servidores XCAP
- Servidores de Correio de Voz Visual
- DNS do Cliente
- OmniSS7 STP (Ponto de Transferência de Sinalização SS7)
- OmniSS7 HLR (Registro de Localização Residencial) - para 2G/3G
- OmniSS7 IP-SM-GW (MAP SMSc)
- OmniSS7 CAMEL Gateway
- MSC (Centro de Comutação Móvel)

> **Nota:** Funções de interconexão SS7/HLR e legadas relacionadas costumam estar no segmento público, uma vez que se conectam a redes externas 2G/3G via SIGTRAN/M3UA em vez de IMS SIP interno.

Exemplo: `10.179.4.0/24`

```yaml
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 Implementação

O OmniCore suporta dois métodos principais para implementar essa segmentação de rede:

Método 1: Interfaces de Rede Físicas/ Virtuais (Recomendado para Produção)

Use NICs separadas ou pontes virtuais para cada segmento de rede. Isso proporciona o maior isolamento e é a abordagem recomendada para

implantações em produção. A convenção é nomear cada ponte de acordo com seu segmento (`v400-omni-packet-core`, `v401-omni-signalling`, `v402-omni-ims-internal`, `v403-omni-ims-public`), embora uma única ponte compartilhada (por exemplo, `vmbr0`) também seja válida onde a segmentação é tratada em outro lugar.

Exemplo:

```yaml
# Núcleo de Pacotes - 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"

# Sinalização - 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: Segmentação Baseada em VLAN

Use uma única interface física com marcação VLAN para separar redes. Isso é adequado para implantações menores ou quando as NICs físicas são limitadas.

Exemplo:

```yaml
# Todos os componentes compartilham uma ponte (ovsbr1); cada segmento usa uma tag VLAN diferente.
# applicationserver (TAS) é um componente Interno 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 e dns são componentes de Sinalização -> 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"
```

Configuração da Rede:

- Configure VLANs no switch físico
- Marque o tráfego adequadamente no nível do hipervisor

- Roteie entre VLANs no gateway/firewall

Exemplo de Mapeamento de VLAN:

```
VLAN 400: 10.x.1.0/24 (Núcleo de Pacotes)
VLAN 401: 10.x.2.0/24 (Sinalização)
VLAN 402: 10.x.3.0/24 (IMS Interno)
VLAN 403: 10.x.4.0/24 (IMS Público)
```

Trabalhando com Endereços IP Públicos

Visão Geral

Muitas implantações do OmniCore requerem que alguns componentes tenham endereços IP públicos para conectividade externa, como:

- **DRA** - Para sinalização diameter de roaming com operadoras externas
- **SGW/PGW de Roaming** - Para tráfego GTP de parceiros de roaming
- **ePDG** - Para chamadas WiFi (túneis IPsec de UEs)
- **Gateway SMSC** - Para conexões SMPP com agregadores de SMS externos
- **P-CSCF** (em algumas implantações) - Para registro SIP direto de UE

Como Atribuir IPs Públicos

IPs públicos são tratados **exatamente da mesma forma que IPs internos** em seus arquivos de inventário de hosts. Basta especificar o endereço IP público no campo `ansible_host`, juntamente com o gateway e a máscara de rede apropriados.

Exemplo: SGW/PGW de Roaming com IPs Públicos

```yaml
sgw:
  hosts:
    # SGWs internos na rede 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 com IPs públicos
    customer-site-roaming-sgw01:
      ansible_host: 203.0.113.10
      gateway: 203.0.113.9
      netmask: 255.255.255.248       # /29 subnet
      host_vm_network: "498-public-servers"
      in_pool: False
      cdrs_enabled: True

smf:  # PGWs
  hosts:
    # PGW de Roaming com IP público
    customer-site-roaming-pgw01:
      ansible_host: 203.0.113.20
      gateway: 203.0.113.17
      netmask: 255.255.255.240       # /28 subnet
      host_vm_network: "497-public-services-LTE"
      in_pool: False
      ip_pools:
        - '100.64.24.0/22'
```

Exemplo: DRA com IP Público

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

Exemplo: ePDG com IP Público

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

Misturando IPs Internos e Públicos

Uma mistura de IPs internos e públicos dentro do mesmo grupo de componentes é comum. Por exemplo:

- SGWs internos para sites locais usando GTP
- SGWs públicos especificamente para tráfego de roaming de operadoras externas
- O mesmo PGW-C pode gerenciar tanto SGWs internos quanto externos

A arquitetura do OmniCore lida com isso de forma transparente - basta configurar cada host com seu endereçamento IP apropriado.

Introdução ao Deployment do Ansible na Omnitouch

Visão Geral

Os Serviços de Rede Omnitouch utilizam o Ansible como sua plataforma de automação de infraestrutura para implantar soluções completas de rede celular (4G/5G) de maneira consistente, repetível e automatizada. Este documento fornece uma visão geral de como aproveitamos o Ansible para orquestrar implantações complexas de telecomunicações.

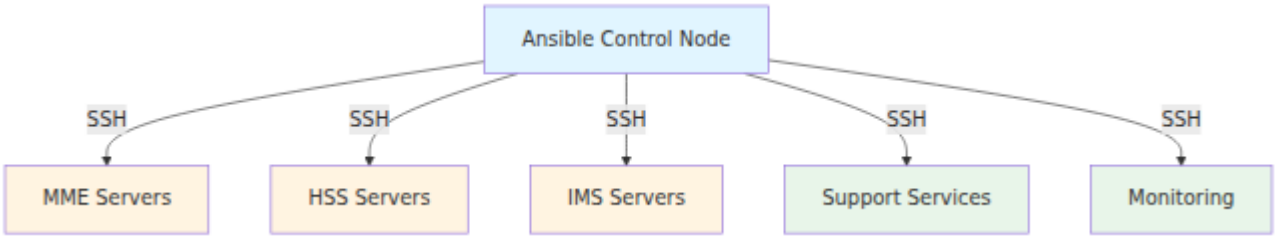
O que é Ansible?

Ansible é uma ferramenta de automação de código aberto que permite que você:

- Configure sistemas
- Implemente software
- Orquestre fluxos de trabalho complexos
- Gerencie infraestrutura como código

Ansible utiliza uma abordagem declarativa - você descreve o **estado desejado** de seus sistemas, e o Ansible garante que eles alcancem esse estado.

Como a Omnitouch Usa o Ansible



Conceitos Chave

1. Inventário (Arquivos de Hosts)

Define **quais** sistemas gerenciar. Cada implantação de cliente tem um arquivo de hosts que descreve:

- Todas as máquinas virtuais na rede
- Seus endereços IP
- Configuração de rede
- Parâmetros específicos de serviço

Os arquivos de hosts são com os quais você trabalhará para definir sua rede.

Veja: Configuração do Arquivo de Hosts

2. Funções

Define **como** configurar cada componente. As funções são unidades reutilizáveis que contêm:

- Tarefas (etapas a serem executadas)
- Modelos (modelos de arquivos de configuração)
- Manipuladores (ações acionadas por alterações)
- Variáveis (valores de configuração padrão)

Exemplos de funções para componentes do OmniCore: `omnihss`, `omnisgwc`, `omnipgwc`, `omnidra`, etc.

A equipe ONS define essas funções. Você pode editá-las. No entanto, geralmente há maneiras mais limpas de fazer quaisquer ajustes que você precise a partir do seu arquivo de hosts.

3. Playbooks

Orquestra **quando** e **onde** as funções são aplicadas:

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

Usamos isso essencialmente como grupos para as funções.

Na prática, cada playbook de serviço aplica uma única função, e a configuração compartilhada é puxada separadamente. Um pacote de serviço como `services/epc.yml` usa `import_playbook` para primeiro executar `common.yml` e depois cada playbook por serviço:

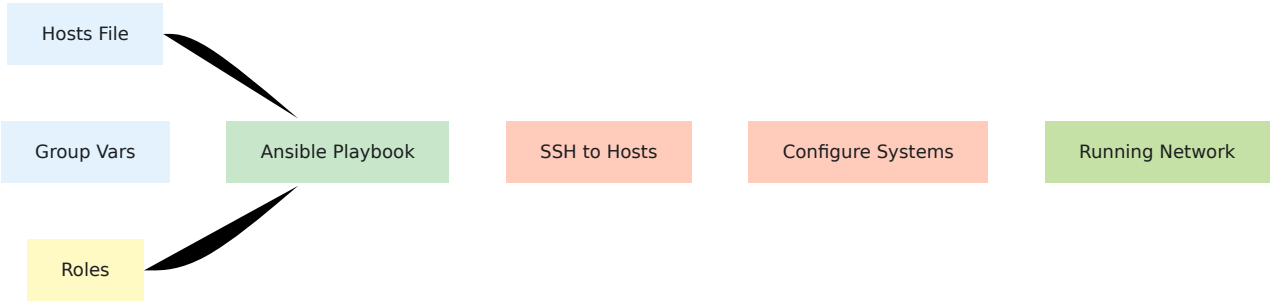
```
- name: Install Common
  import_playbook: common.yml

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

4. Variáveis de Grupo

Fornece **configuração específica do cliente** que substitui os padrões das funções. É aqui que a personalização do cliente acontece sem modificar as funções base.

Veja: Variáveis de Grupo e Configuração

Arquitetura de Implantação



O Processo de Implantação

1. Definir Infraestrutura

Crie um arquivo de hosts descrevendo sua topologia de rede:

Nota de Planejamento: Antes de definir a infraestrutura, revise o Padrão de Planejamento de IP para orientações sobre segmentação de rede, alocação de endereços IP e organização de sub-redes.

Usuários do Proxmox: Se implantando no Proxmox, veja Implantação de VM/LXC no Proxmox para provisionamento automatizado de VM/conteiner.

Veja: Configuração do Arquivo de Hosts e Referência de Configuração

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

2. Personalizar Configuração

Defina variáveis específicas do cliente em `group_vars`:

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

#ToDo - Adicionar link aqui para referência de configuração para lista completa

3. Executar Playbooks

Implante a rede:

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

4. Implantação Automatizada

O Ansible irá:

- Criar/provisionar VMs (se usando integração Proxmox/VMware)
- Configurar rede
- Instalar pacotes de software do cache APT
- Implantar código de aplicação
- Configurar serviços com configurações do cliente
- Iniciar serviços
- Validar implantação

Componentes Chave que Implantamos

OmniCore (Plataforma de Core de Pacotes 4G/5G)

- **OmniHSS** - Servidor de Assinante Residencial

- **OmniSGW** - Gateway de Serviço (plano de controle)
- **OmniPGW** - Gateway de Pacotes (plano de controle)
- **OmniUPF** - Função de Plano do Usuário
- **OmniDRA** - Agente de Roteamento Diameter
- **OmniTWAG** - Gateway de Acesso WLAN Confiável

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

OmniCall (Plataforma de Voz e Mensagens)

- **OmniCall CSCF** - Função de Controle de Sessão de Chamada (P-CSCF, I-CSCF, S-CSCF)
- **OmniTAS** - Servidor de Aplicação IMS (serviços VoLTE/VoNR)
- **OmniMessage** - Centro de SMS (SMS-C)
- **OmniMessage SMPP** - Suporte ao protocolo SMPP
- **OmniSS7** - Componentes de sinalização SS7 (STP, HLR, CAMEL)
- **VisualVoicemail** - Funcionalidade de correio de voz

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

OmniCharge/OmniCRM

- **Plataforma CRM** - Gestão de relacionamento com o cliente, auto-inscrição, faturamento

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

Serviços de Suporte

- **DNS** - Resolução de DNS da rede
- **Servidor de Licenças** - Gestão de licenças
- **Monitoramento** - Prometheus, Grafana

Veja: Visão Geral da Arquitetura de Implantação

Gestão de Pacotes

Usamos um modelo híbrido de distribuição de pacotes:

Pacotes APT Pré-compilados

Todo o software da Omnitouch é distribuído como pacotes Debian (`.deb`):

- Construídos a partir do código-fonte em nosso pipeline CI/CD
- Versionados e testados
- Hospedados em repositórios de pacotes

Sistema de Cache APT

Os clientes podem escolher entre:

1. **Cache APT Local** - Espelho dos pacotes necessários no local para implantação offline
2. **Repositório Público** - Acesso direto ao repositório de pacotes hospedado pela Omnitouch

Há também um `.deb` unificado `omnicore` (construído a partir de `packaging/`) que agrupa os playbooks e ferramentas para auto-provisionamento de uma implantação.

Veja: Sistema de Cache APT

Gestão de Licenças

Todos os componentes de software da Omnitouch requerem licenças válidas gerenciadas através de um servidor de licenças central:

- Os componentes verificam a validade da licença na inicialização
- Recursos são ativados/desativados com base na licença
- O servidor de licenças pode ser local ou hospedado na nuvem

Veja: Servidor de Licenças

Benefícios Desta Abordagem

Repetibilidade

Os mesmos playbooks do Ansible podem implantar:

- Laboratórios de desenvolvimento
- Ambientes de teste
- Redes de produção
- Sites de clientes

Consistência

Cada implantação usa as mesmas configurações testadas, reduzindo erros humanos.

Controle de Versão

A infraestrutura é definida como código no Git:

- Rastreie todas as alterações
- Revise antes da implantação
- Reverta se necessário

Personalização Sem Complexidade

Os clientes podem personalizar sua implantação através de `group_vars` sem modificar funções principais.

Implantação Rápida

Implante uma rede celular completa em horas em vez de dias ou semanas.

Começando

Pré-requisitos

Antes de executar os playbooks do Ansible, você precisa instalar as dependências necessárias. Este projeto usa uv para gerenciar o ambiente Python, então **uv é necessário**.

1. Instalar uv

Se o uv ainda não estiver instalado, instale-o usando o método que melhor se adequa ao seu nó de controle. Veja a documentação de instalação do uv. Por exemplo:

```
pipx install uv                                        # via pipx
curl -LsSf https://astral.sh/uv/install.sh | sh    # instalador
autônomo
```

2. Instalar Dependências

A partir da raiz do repositório, execute:

```
uv sync
```

Isso cria um `.venv/` isolado e instala o Ansible mais todos os pacotes Python necessários (fixados em `uv.lock`) para automação de implantação da Omnitouch.

3. Ativar o Ambiente

Ative o ambiente virtual e mantenha-o ativado para todos os comandos do Ansible:

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

4. Instalar Coleções do Ansible

Instale as coleções do Ansible das quais os playbooks dependem (a partir de `requirements.yml`):

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

5. Executar Comandos do Ansible

Com o ambiente ativado, execute comandos do Ansible diretamente:

```
ansible --version
```

Nota: Desative o ambiente quando terminar executando `deactivate`.

Etapas de Implantação

1. Revise a Configuração do Arquivo de Hosts para entender como definir sua rede
2. Aprenda sobre Variáveis de Grupo para personalização
3. Entenda o Sistema de Cache APT para gestão de pacotes
4. Revise a Arquitetura de Implantação para ver como tudo se encaixa
5. Implemente!

Próximos Passos

- Padrão de Planejamento de IP - **Planeje sua arquitetura de rede e alocação de IP**
- Configuração do Arquivo de Hosts - Aprenda como definir sua topologia de rede
- Sistema de Cache APT - Entenda a distribuição de pacotes
- Servidor de Licenças - Aprenda sobre gestão de licenças
- Visão Geral da Arquitetura de Implantação - Veja o quadro completo
- Configuração de Variáveis de Grupo - Personalize sua implantação
- Playbooks de Utilidade - Ferramentas operacionais para verificações de saúde, backups e manutenção

Repositório APT e Distribuição de Pacotes

Visão Geral

O sistema APT do Omnitouch fornece distribuição de pacotes para todas as implantações. Dois tipos de conteúdo são servidos:

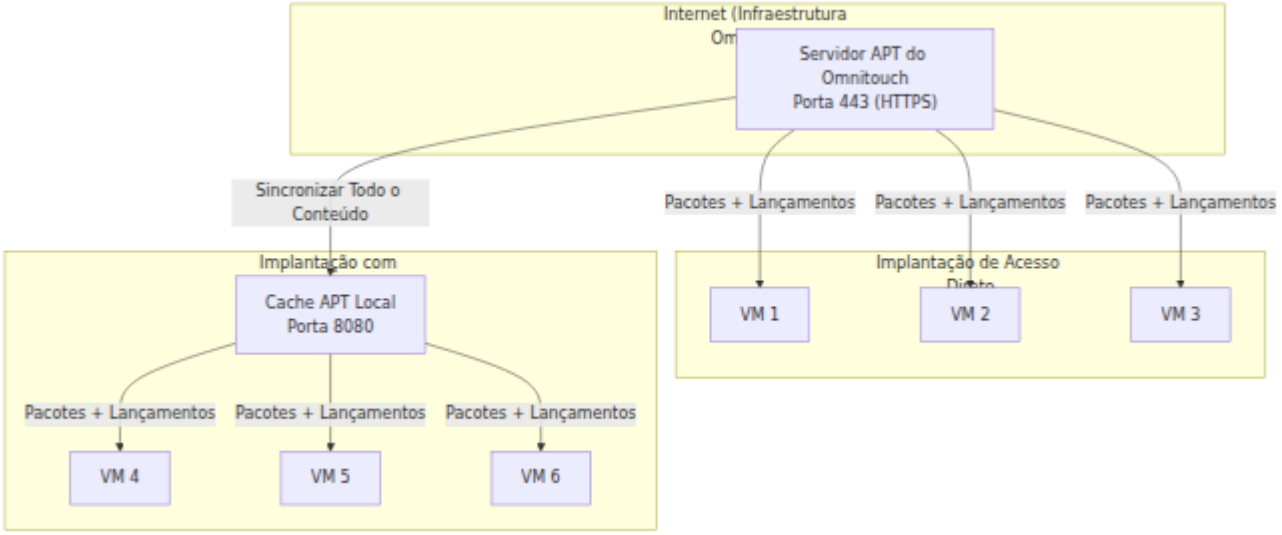
1. **Pacotes APT**: pacotes Debian instalados via `apt install`
2. **Lançamentos Binários**: binários pré-compilados baixados diretamente (exportadores Prometheus, agentes, etc.)

A página de pacotes no servidor APT também inclui uma aba **Auto-Configuração (OmniCore)**, que fornece o pacote `omnicore` (servido em `/pool/main/o/omnicore/`) contendo o conjunto completo de ferramentas de automação Ansible para clientes que desejam implantar e gerenciar sua rede por conta própria.

Dois modelos de implantação são suportados:

1. **Acesso Direto**: VMs puxam pacotes diretamente do servidor APT do Omnitouch (por exemplo, `apt.<region>.omnitou.ch`)
2. **Espelho de Cache Local**: um servidor local sincroniza com o Omnitouch e serve pacotes para VMs (para implantações offline/isoladas)

Arquitetura



Conteúdo Servido

O servidor APT hospeda todo o conteúdo necessário para implantações:

Tipo de Conteúdo	Descrição	Caminho
Pacotes Omnitouch	Pacotes `.deb` personalizados (omnihss, omnimme, etc.)	`/dists/<distro>/`
Conjunto de Ferramentas OmniCore	O pacote `omnicore`: automação completa do Ansible (funções, playbooks, dependências) para implantação self-service	`/pool/main/o/omnicore/`
Pacotes Ubuntu	Pacotes Ubuntu em cache com todas as dependências	`/<distro>/pool/main/`
Lançamentos do GitHub	Binários pré-compilados (Prometheus, Grafana, VoIPmonitor, etc.)	`/releases/<org>/<repo>/`
Tarballs de Fonte	Arquivos de origem para aplicativos web (CGrateS_UI, speedtest)	`/repos/`
Pacotes de Terceiros	Galera, FRR, InfluxDB, KeyDB, etc.	`/releases/<vendor>/`

Variáveis de Configuração

`apt_repo` é a única fonte de verdade para distribuição de pacotes. Defina-a uma vez em seu inventário. Todo o resto é derivado dela. Você não configura a URL de download binário, esquema, porta ou credenciais separadamente.

apt_repo
(única fonte de verdade,
definida no inventário)
Derivado
remote_apt_*
(roles/common/defaults/main.yml)
apt_download_base
(roles/common/tasks/facts.yml)
O que eles configuram
/etc/apt/sources.list.d/omnitouch.list
Downloads binários
/releases/, /repos/

Como a derivação funciona

Variável	Onde definida	Função
`apt_repo`	Seu inventário	O único bloco que você configura. Templado diretamente na linha de origem do APT.
`remote_apt_*`	`roles/common/defaults/main.yml`	Derivado de `apt_repo`. Último recurso quando `apt_repo` está ausente: `apt.us-fl.omnitou.ch` / `443` / `https`.
`apt_download_base`	`roles/common/tasks/facts.yml`	URL base única computada que cada função usa para downloads binários, de modo que a linha de origem do APT e os downloads binários sempre concordem em host, esquema,

Variável	Onde definida	Função
		porta e credenciais.

Como ambos os valores derivados são construídos a partir do mesmo bloco `apt_repo`, um inventário normal **precisa apenas de `apt_repo`**. Você não define `remote_apt_*` por conta própria.

> **Exceção: o servidor de espelho de cache.** Um host no grupo `apt_cache_servers` é o único lugar onde `remote_apt_*` ainda é definido explicitamente. Seu próprio `apt_repo` aponta para o *cache local*. Os valores `remote_apt_*` então descrevem o servidor Omnitouch *upstream* do qual ele sincroniza (veja Opção 2).

Esquema e porta

O esquema segue a porta automaticamente: **porta `443` → `https`, qualquer outra porta → `http`**. Quando `apt_repo.apt_repo_port` é omitido:

- **Acesso direto** padrão para **`443` (HTTPS)**
- **Modo de cache** padrão para **`8080` (HTTP)**

Quando cada modo é usado

Cenário	Linha de origem APT	Downloads binários (`apt_download_base`)
`use_apt_cache: true` (cache)	`http://<cache>:8080/...` (sem autenticação)	`http://<cache>:8080/...` (sem autenticação)
`use_apt_cache: false` (direto)	`https://<user>:<pass>@<server>/...`	`https://<user>:<pass>@<server>:443/...`

Opção 1: Acesso Direto

Para implantações com conectividade à internet, as VMs puxam pacotes diretamente do servidor APT do Omnitouch.

Requisitos de Rede

Lista de Permissão de IP de Origem: Seu endereço IP público deve estar na lista de permissão no servidor APT do Omnitouch. Durante a configuração, forneça suas sub-redes de origem ao Omnitouch. Em troca, você receberá:

- **Nome de usuário** e **senha** para Autenticação Básica HTTP
- **FQDN** para o servidor APT (por exemplo, `apt.<region>.omnitou.ch`)

Requisitos de Firewall: O acesso de saída para os seguintes intervalos de IP do Omnitouch deve ser permitido:

Rede	Intervalo
IPv4	`144.79.167.0/24`
IPv4	`160.22.43.0/24`
IPv6	`2001:df3:dec0::/48`
ASN	`AS152894`

Serviços que requerem acesso à infraestrutura do Omnitouch:

Serviço	Porta	Protocolo	Propósito
Servidor APT	443	TCP	Downloads de pacotes (HTTPS)
Servidor APT	53	TCP/UDP	Resolução DNS para o FQDN do APT
Servidor de Licença	123	UDP	Sincronização de tempo NTP para validação de licença
Servidor de Licença	53	TCP/UDP	Resolução DNS para validação de licença

Certifique-se de que o tráfego HTTPS (TCP/443), NTP (UDP/123) e DNS (TCP+UDP/53) seja permitido para os intervalos de IP do Omnitouch.

Configuração

```yaml
all:
  vars:
    use_apt_cache: false

    # Única fonte de verdade. Configuração de download binário (remote_apt_*) e
    # a URL apt_download_base são derivadas automaticamente - você NÃO
    # as define separadamente.
    apt_repo:
      apt_server: "apt.<region>.omnitou.ch"   # FQDN fornecido pelo Omnitouch
      apt_repo_username: "seu-usuario"
      apt_repo_password: "sua-senha"
      # apt_repo_port: 443                     # opcional; padrão para 443 (HTTPS)
```

Parâmetros

Fontes de Pacote APT (`apt_repo`)

Parâmetro	Tipo	Requerido	Padrão	Descriçã
`apt_repo.apt_server`	String	Sim	-	Nome do host ou endereço do servido APT
`apt_repo.apt_repo_username`	String	Sim	-	Nome de usuário da Autentica Básica HT
`apt_repo.apt_repo_password`	String	Sim	-	Senha da Autentica Básica HT
`apt_repo.apt_repo_port`	Inteiro	Não	`443`	Porta do servidor. C esquema segue a porta (`443` HTTPS, ca contrário HTTP).

Downloads Binários (`remote_apt_*`)

Esses são derivados automaticamente de `apt_repo`. Não os defina em um inventário de acesso direto. Eles existem apenas para que o servidor de espelho de cache possa apontar para um upstream diferente (veja a Opção 2). Para referência, eles têm os seguintes padrões:

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

Geral

Parâmetro	Tipo	Requerido	Padrão	Descrição
`use_apt_cache`	Booleano	Sim	-	Deve ser `false` para acesso direto

Padrões de URL (Acesso Direto)

Fontes de Pacote APT (configuradas em `/etc/apt/sources.list.d/omnitouch.list`):

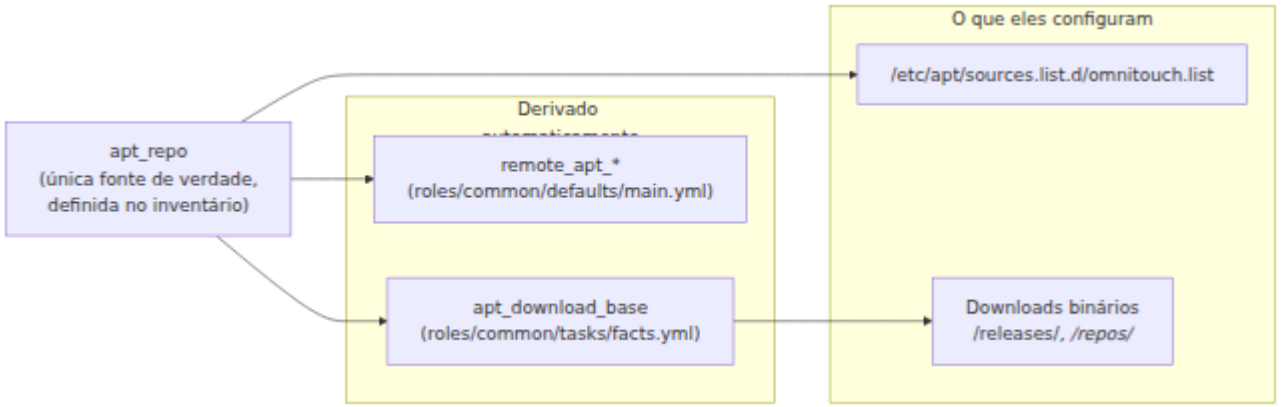
```
deb [trusted=yes] https://{username}:{password}@{apt_server}/
noble main
```

Downloads Binários (construídos a partir de `apt_download_base` por tarefas `get_url` do Ansible):

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

> A porta `443` é omitida da URL (é o padrão HTTPS). Uma porta não padrão é anexada como `:<port>` e troca o esquema para HTTP, a menos que seja `443`.

Como Funciona



As VMs se autenticam com Autenticação Básica HTTP para pacotes APT e downloads binários. Pacotes de sistema Ubuntu também são servidos a partir do servidor Omnitouch (pré-cacheados), portanto, as VMs não precisam de acesso a espelhos do Ubuntu.

Assinatura de Pacote (GPG)

O servidor APT do Omnitouch gera uma chave de assinatura de repositório e serve a chave pública armada em `/omnitouch.gpg`. Para hosts de acesso direto, o papel `common` busca automaticamente essa chave e a instala em `/etc/apt/trusted.gpg.d/omnitouch.gpg` para que `apt-get update` verifique os metadados do repositório sem avisos `NO_PUBKEY`. Isso acontece automaticamente; nenhuma gestão manual de chaves é necessária.

> O modo de espelho de cache não usa verificação GPG. Hosts que puxam de um cache local usam `[trusted=yes]` em vez disso (veja a Opção 2).

Opção 2: Cache Local

Para implantações offline, isoladas ou com largura de banda restrita, implante um cache APT local que sincroniza todo o conteúdo do Omnitouch.

Arquitetura

Configuração

Defina o servidor de cache em seu arquivo de hosts. É o único lugar onde `remote_apt_*` é definido explicitamente. Essas variáveis descrevem o servidor Omnitouch **upstream** do qual o cache sincroniza:

```
apt_cache_servers:
  hosts:
    example-apt-cache:
      ansible_host: 192.168.1.100
      gateway: 192.168.1.1
  vars:
    # O servidor de cache sincroniza conteúdo DO repositório
Omnitouch upstream
    remote_apt_server: "apt.<region>.omnitou.ch"
    remote_apt_port: 443
    remote_apt_protocol: "https"
    remote_apt_user: "seu-usuario"
    remote_apt_password: "sua-senha"

all:
  vars:
    # use_apt_cache: true  # Definido automaticamente quando o
grupo apt_cache_servers existe
    # apt_repo.apt_server: auto-derivado para 192.168.1.100
(primeiro servidor de cache)
```

Como funciona:

- **Servidor de cache** (`192.168.1.100`): Usa credenciais `remote_apt_*` para sincronizar conteúdo do servidor Omnitouch upstream via HTTPS.
- **Todos os outros hosts**: Derivam automaticamente `apt_repo.apt_server: "192.168.1.100"` e puxam do cache na porta `8080` sem credenciais.

Parâmetros

Fontes de Pacote APT (`apt_repo`)

Parâmetro	Tipo	Requerido	Padrão	Des
`apt_repo.apt_server`	String	Sim	Auto-derivado	IP do serv cache loc Derivado automati do prime `apt_cach` se não especifica
`apt_repo.apt_repo_username`	String	Não	-	Não é ne ao usar c autentica necessár
`apt_repo.apt_repo_password`	String	Não	-	Não é ne ao usar c autentica necessár
`apt_repo.apt_repo_port`	Inteiro	Não	`8080`	Porta que usam par o servido local. (Su `apt_cach` obsoleto.

Sincronização do Servidor de Cache (`remote_apt_*`)

Essas variáveis configuram como o servidor de cache sincroniza conteúdo do Omnitouch. Este é o único cenário onde `remote_apt_*` é definido manualmente:

Parâmetro	Tipo	Requerido	Padrão	Descrição
`remote_apt_server`	String	Sim	-	Servidor APT Omnitouch upstream para sincronizar
`remote_apt_port`	Inteiro	Não	`443`	Porta do servidor APT Omnitouch upstream
`remote_apt_protocol`	String	Não	`https`	Protocolo para conexão de sincronização
`remote_apt_user`	String	Sim	-	Credenciais para sincronização do Omnitouch
`remote_apt_password`	String	Sim	-	Credenciais para sincronização do Omnitouch

Geral

Parâmetro	Tipo	Requerido	Padrão	Descrição
`use_apt_cache`	Booleano	Não	`true`	Definido automaticamente como `true` quando o grupo `apt_cache_servers` existe

> **Obsoleto:** `apt_cache_port` não é mais usado. A porta que os hosts conectam agora é definida via `apt_repo.apt_repo_port` (padrão para `8080` quando omitido).

Padrões de URL (Modo de Cache)

Fontes de Pacote APT (configuradas em `/etc/apt/sources.list.d/omnitouch.list`):

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

Downloads Binários (construídos 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
```

O acesso ao cache não precisa de credenciais. Ele usa a configuração APT `[trusted=yes]`.

Implantando o Cache

1. **Provisione o servidor de cache** (VM ou contêiner LXC com disco de 50+ GB)

2. **Execute o playbook de configuração do cache:**

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

3. **Verifique o cache** navegando para `http://192.168.1.100:8080/`

O que é Sincronizado

O espelho de cache sincroniza **todo o conteúdo** do servidor APT do Omnitouch. O mecanismo difere por tipo de conteúdo:

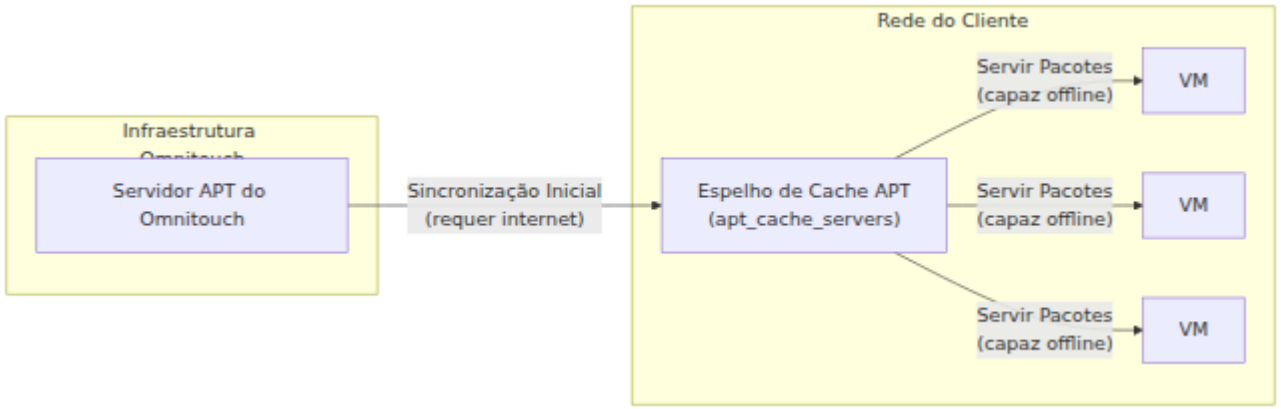
- **Pacotes APT** (`/dists/` + `/pool/` para Omnitouch e Ubuntu) são sincronizados com `curl` impulsionado pelo índice do repositório Debian. O espelho busca o arquivo `Release` upstream (e interrompe se não houver alterações), analisa `Packages.gz` para o nome de arquivo/SHA256/tamanho autoritativo de cada pacote e baixa apenas arquivos que estão faltando ou cujo SHA256 não corresponde à cópia local. Cada arquivo baixado é verificado por SHA256 antes de ser comprometido.
- **Lançamentos binários e tarballs de fonte** (`/releases/`, `/repos/`) não têm índice no estilo Debian, portanto, recorrem ao `wget --recursive --timestamping`, que apenas re-baixa arquivos com um `Last-Modified` upstream mais recente.

Diretórios de conteúdo sincronizados:

Caminho	Conteúdo
`/dists/<distro>/`	Metadados do repositório APT (Pacotes, arquivos de Lançamento)
`/pool/main/`	Pacotes .deb personalizados do Omnitouch
`/<distro>/pool/main/`	Pacotes Ubuntu e todas as dependências
`/releases/`	Lançamentos do GitHub (Prometheus, Grafana, Zabbix, etc.)
`/repos/`	Tarballs de fonte (Erlang, Elixir, CGrateS_UI, etc.)

Após a sincronização inicial, o cache pode servir todos os pacotes sem conectividade à internet.

Como Funciona



O espelho de cache se autentica com Autenticação Básica HTTP para todo o conteúdo. Pacotes APT são sincronizados com `curl` contra o índice do repositório Debian (`Release` + `Packages.gz`), baixando apenas arquivos cujo SHA256 difere da cópia local e verificando cada um após o download; `/releases/` e `/repos/` recorrem ao `wget --recursive --timestamping`. De qualquer forma, sincronizações subsequentes apenas buscam novos arquivos ou arquivos alterados.

Configuração Automática

Quando um grupo `apt_cache_servers` existe em seu inventário, o sistema automaticamente:

1. Define `use_apt_cache: true` para todos os hosts (a menos que explicitamente substituído)
2. Deriva `apt_repo.apt_server` do IP `ansible_host` do primeiro servidor de cache

Exemplo Mínimo de Configuração

```
apt_cache_servers:
  hosts:
    apt-cache-01:
      ansible_host: 192.168.1.100
      gateway: 192.168.1.1
  vars:
    # O servidor de cache sincroniza conteúdo do repositório
Omnitouch upstream
    remote_apt_server: "apt.<region>.omnitou.ch"
    remote_apt_user: "seu-usuario"
    remote_apt_password: "sua-senha"
```

O que acontece automaticamente:

- Todos os hosts (exceto o servidor de cache) recebem `use_apt_cache: true`
- Todos os hosts (exceto o servidor de cache) recebem `apt_repo.apt_server: "192.168.1.100"`
- Todos os hosts puxam de `http://192.168.1.100:8080/` sem credenciais
- O servidor de cache sincroniza pacotes de `https://seu-usuario:sua-senha@apt.<region>.omnitou.ch/`

Substituir Comportamento Automático

Para forçar acesso direto mesmo com servidores de cache definidos:

```yaml
all:
  vars:
    use_apt_cache: false  # Forçar acesso direto mesmo com
servidores de cache definidos

    apt_repo:
      apt_server: "apt.<region>.omnitou.ch"
      apt_repo_username: "usuario"
      apt_repo_password: "senha"
```

Resumo da Configuração

Cenário 1: Acesso Direto ao Servidor APT (Sem Cache)

Todos os hosts puxam pacotes diretamente do servidor de repositório APT. Apenas `apt_repo` é necessário. Os valores `remote_apt_*` são derivados dele.

```yaml
all:
  vars:
    use_apt_cache: false

    # Única fonte de verdade para fontes APT e downloads binários
    apt_repo:
      apt_server: "apt.<region>.omnitou.ch"
      apt_repo_username: "usuario"
      apt_repo_password: "senha"
```

Resultado: Todos os hosts geram `deb [trusted=yes] https://usuario:senha@apt.<region>.omnitou.ch/ noble main`

Cenário 2: Servidor de Cache APT Definido no Arquivo de Hosts (Automático)

O servidor de cache está em seu inventário e será implantado/sincronizado pelo Ansible.

```yaml
apt_cache_servers:
  hosts:
    cache-server:
      ansible_host: 192.168.1.100
      gateway: 192.168.1.1
  vars:
    # O servidor de cache sincroniza pacotes do repositório autenticado upstream
    remote_apt_server: "apt.<region>.omnitou.ch"
    remote_apt_port: 443
    remote_apt_protocol: "https"
    remote_apt_user: "usuario"
    remote_apt_password: "senha"

# Nenhuma configuração necessária em all: vars:
# Tudo auto-derivado do grupo apt_cache_servers
```

Resultado:

- **Servidor de cache**: Sincroniza de `https://user:pass@apt.<region>.omnitou.ch/`
- **Todos os outros hosts**: Gera `deb [trusted=yes] http://192.168.1.100:8080/noble noble main` (sem credenciais)

Cenário 3: Cache APT Remoto NÃO no Arquivo de Hosts (Manual)

O servidor de cache existe em outro lugar e já está configurado (não gerenciado pelo seu Ansible).

```yaml
all:
  vars:
    use_apt_cache: true

    # Aponta todos os hosts para o servidor de cache externo
    apt_repo:
      apt_server: "192.168.1.100"  # IP do servidor de cache
externo
      apt_repo_port: 8080           # O cache normalmente roda na
porta 8080

# Nenhum grupo apt_cache_servers necessário
# Nenhum remote_apt_* necessário (o cache já está configurado
externamente)
```

Resultado: Todos os hosts geram `deb [trusted=yes] http://192.168.1.100:8080/noble noble main` (sem credenciais)

Exemplo Completo

Este é um exemplo completo e funcional. Ele mostra a configuração do servidor de cache com múltiplos hosts de aplicação:

```yaml
# Grupo de Servidores de Cache 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:
    # O servidor de cache sincroniza pacotes do repositório
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 Aplicação
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

# Configuração Global
all:
  vars:
    # Auto-configuração (nenhuma configuração manual necessária):
    # - use_apt_cache: true (auto-ativado quando apt_cache_servers
```

```
existe)
    # - apt_repo.apt_server: "10.179.1.114" (auto-derivado do
servidor de cache)
```

O que acontece durante a implantação:

1. **Servidor de cache (`10.179.1.114`):**
   - Usa `remote_apt_*` da sua seção `vars:`
   - Baixa todos os pacotes de `https://customer-username:customer-secure-token@apt.<region>.omnitou.ch/`
   - Serve pacotes na porta 8080 via nginx
2. **Hosts de aplicação (`customer-hss01`, `customer-mme01`, `customer-dns01`):**
   - Detecta automaticamente que o grupo `apt_cache_servers` existe
   - Define automaticamente `use_apt_cache: true`
   - Deriva automaticamente `apt_repo.apt_server: "10.179.1.114"`
   - Gera: `deb [trusted=yes] http://10.179.1.114:8080/noble noble main`
   - Puxa todos os pacotes do servidor de cache (sem credenciais necessárias)

Atualizando o Cache

Para sincronizar novos pacotes ou atualizações:

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

Isso sincroniza incrementalmente todo o conteúdo do servidor APT Omnitouch:

- Novas versões de pacotes Omnitouch
- Novos pacotes e dependências do Ubuntu

- Novos lançamentos do GitHub
- Tarballs de origem atualizados

As re-sincronizações são incrementais: os pacotes APT são comparados com o índice Debian upstream por SHA256 (e são completamente ignorados se o arquivo `Release` upstream não tiver mudado), enquanto `/releases/` e `/repos/` usam `wget --timestamping`. Arquivos existentes que não mudaram são ignorados, tornando a re-sincronização rápida.

Nota: O servidor APT Omnitouch é a única fonte de verdade para todos os pacotes. Omnitouch compila e publica pacotes nele; executar `services/apt_cache.yml` em espelhos de cache então sincroniza o que está atualmente publicado.

Solução de Problemas

Atualização APT Falha com 401 Não Autorizado

Sintomas:

```
Falha ao buscar
https://10.179.1.115/noble/dists/noble/main/binary-amd64/Packages
401  Não Autorizado
```

Possíveis causas:

- Configuração `apt_repo` definida em `all: vars:` em vez de `apt_cache_servers: vars:`
- Hosts tentando acessar repositório autenticado diretamente em vez de pelo cache
- `apt_repo_username` ou `apt_repo_password` incorretos
- IP de origem não está na lista de permissões no servidor APT Omnitouch
- Usando credenciais de cache para acesso direto ou vice-versa

Resolução:

1. **Verifique o escopo da configuração**: Certifique-se de que `apt_repo` com credenciais está definido em `apt_cache_servers: vars:`, NÃO em `all: vars:`
2. **Verifique o modo de cache**: Ao usar cache, os hosts devem se conectar ao servidor de cache (porta 8080), não ao repositório upstream
3. **Verifique as fontes geradas**: No host com falha, verifique `/etc/apt/sources.list.d/omnitouch.list`
   - **Correto (modo de cache)**: `deb [trusted=yes] http://10.179.1.114:8080/noble noble main`
   - **Incorreto (tem credenciais no lugar errado)**: `deb [trusted=yes] https://user:pass@10.179.1.115/noble noble main`
4. Verifique se as credenciais estão corretas para seu modo de implantação
5. Confirme se seu IP público está na lista de permissões com Omnitouch (se estiver usando acesso direto)

Falhas de Downloads Binários (Node Exporter, Zabbix, etc.)

Sintomas: O playbook do Ansible falha ao baixar arquivos do caminho `/releases/`

Possíveis causas:

- Credenciais `apt_repo` incorretas (downloads binários derivam sua autenticação de `apt_repo` em modo direto)
- Em um servidor de espelho de cache, `remote_apt_user` / `remote_apt_password` incorretos
- Esquema ou porta errados. Lembre-se de que a porta `443` implica HTTPS e qualquer outra porta implica HTTP.

Resolução:

1. Em modo direto, verifique se `apt_repo.apt_repo_username` / `apt_repo.apt_repo_password` correspondem aos fornecidos pela Omnitouch
2. Em um servidor de espelho de cache, verifique as credenciais upstream `remote_apt_*`

3. Confirme se o `apt_download_base` derivado usa o host/esquema/porta esperados (verifique com `ansible-playbook ... -vvv` ou inspecionando a tarefa `get_url` que falhou)

O Servidor de Cache Não Consegue Sincronizar

Sintomas: O playbook do servidor de cache falha ao baixar pacotes

Possíveis causas:

- O servidor de cache não tem acesso à internet
- Credenciais `remote_apt_*` incorretas
- Firewall bloqueando conexões de saída para Omnitouch

Resolução:

1. Verifique se o servidor de cache pode alcançar o servidor APT Omnitouch upstream na porta 443
2. Verifique as credenciais `remote_apt_*`
3. Revise as regras de firewall para acesso de saída

Documentação Relacionada

- Configuração do Arquivo de Hosts: Inventário e configuração de variáveis
- Referência de Configuração: Referência completa de parâmetros
- Arquitetura de Implantação: Arquitetura geral do sistema
- Implantação Proxmox: Implantando o servidor de cache como contêiner LXC

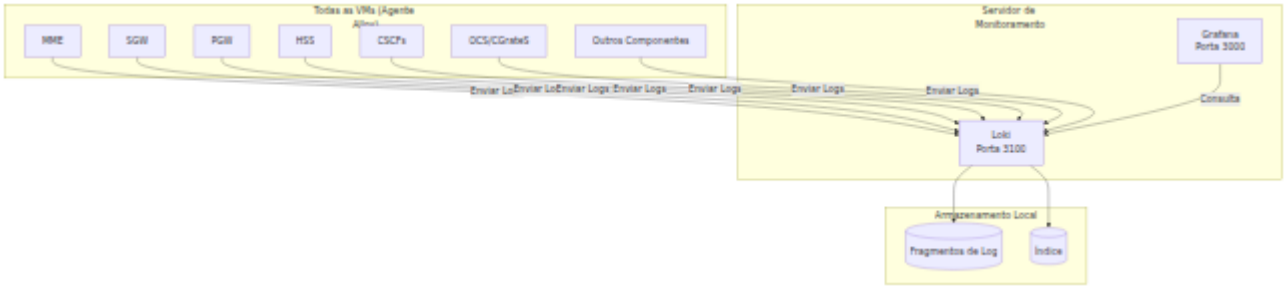
Registro Centralizado

Visão Geral

OmniCore inclui um sistema de registro centralizado que coleta logs de todos os componentes e os torna pesquisáveis através do Grafana. O sistema utiliza:

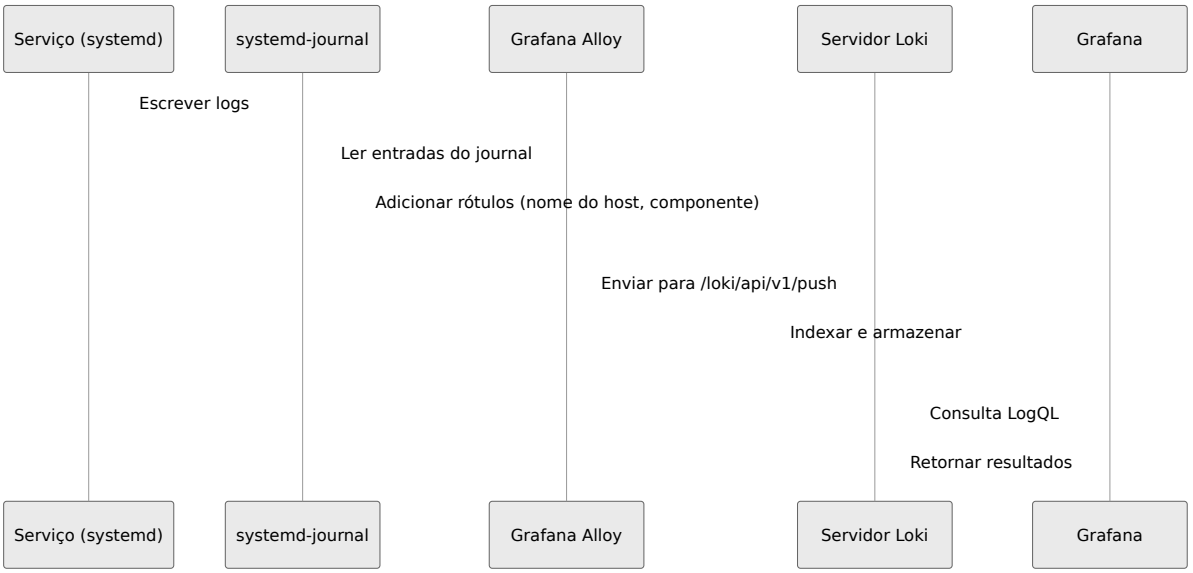
- **Grafana Alloy**: Agente de coleta de logs implantado em todas as VMs
- **Grafana Loki**: Servidor de agregação e armazenamento de logs nos hosts de monitoramento
- **Grafana Dashboards**: Dashboards pré-construídos para cada tipo de componente

Arquitetura



Como Funciona

Fluxo de Coleta de Logs



Rótulos de Log

Cada entrada de log inclui estes rótulos para filtragem e consulta:

Rótulo	Descrição	Exemplo
`job`	Nome do host da VM de origem	`customer-mme01`
`hostname`	Mesmo que job (para compatibilidade)	`customer-mme01`
`component`	Tipo de componente do grupo de inventário	`mme`, `pcscf`, `ocs`
`unit`	Nome da unidade systemd	`omnimme.service`
`level`	Severidade do log	`info`, `warning`, `error`
`service`	Identificador do Syslog	`omnimme`

Configuração

Implantação Automática

O registro é configurado automaticamente quando:

1. Um grupo `monitoring` existe em seu inventário
2. O papel comum é executado nos hosts de destino

Nenhuma configuração adicional é necessária para funcionalidade básica.

Requisitos de Inventário

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

Quando o grupo `monitoring` é definido:

- **Hosts de Monitoramento**: Executam o servidor Loki, que recebe e armazena os logs. Eles também executam o Alloy, que coleta os próprios logs do host e termina o fluxo de syslog do dispositivo de rede.
- **Todos os outros hosts**: Executam o agente Alloy, que coleta os logs e os envia para o Loki.

Configuração de Retenção

O Loki é configurado com retenção baseada em tempo. O compactador exclui logs mais antigos do que o período de retenção:

Configuração	Padrão	Descrição
Retenção baseada em tempo	72 horas (3 dias)	Logs mais antigos do que o período de retenção são excluídos

Para personalizar a retenção, substitua esta variável em seu inventário:

```yaml
all:
  vars:
    loki_retention_period: "168h"  # por exemplo, 7 dias em horas
```

Configuração do Buffer Alloy

O Alloy armazena logs localmente em um log de escrita antecipada (WAL) quando o Loki não está acessível. Segmentos armazenados são descartados

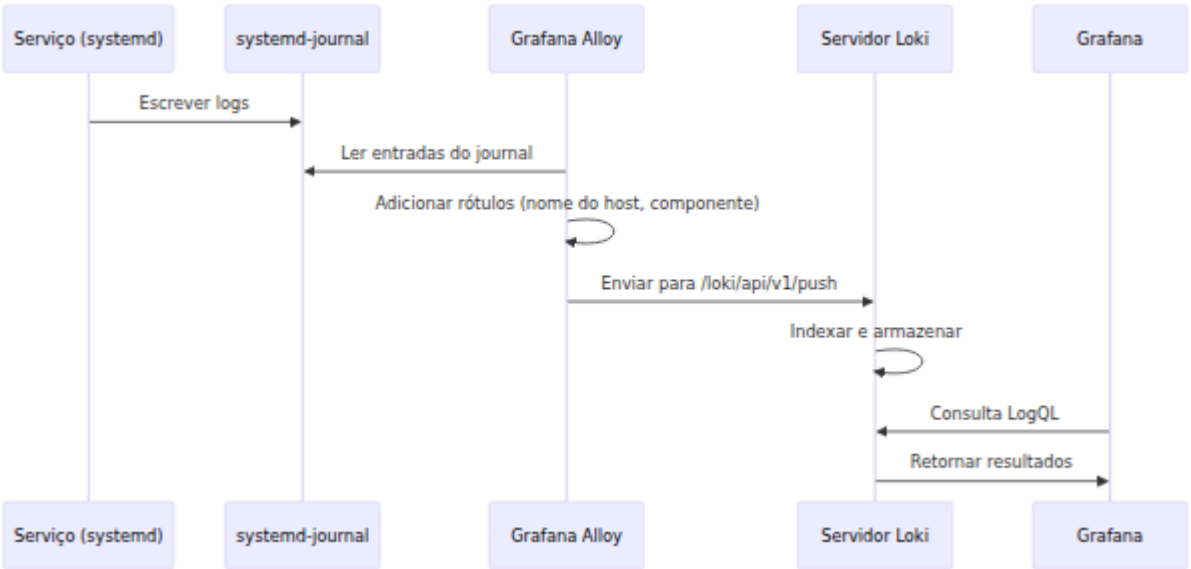
para evitar crescimento indefinido do disco:

Configuração	Padrão	Descrição
Idade máxima do segmento WAL	1 hora	Segmentos de log armazenados mais antigos que 1 hora são descartados

Quando o Loki não está acessível por mais tempo que a idade do segmento, os logs armazenados mais antigos são descartados.

Syslog de Dispositivos de Rede

Roteadores, switches e outros dispositivos não podem executar o Alloy. Esses dispositivos enviam syslog BSD para o host de monitoramento. O Alloy recebe o fluxo e o escreve no mesmo Loki, sob `job="network_syslog"`. Os logs do dispositivo compartilham a pesquisa, intervalo de tempo e retenção dos logs do host.



O ouvinte é parte da construção padrão de monitoramento. Um site não define uma flag para receber syslog de dispositivos. O Alloy nomeia cada dispositivo a partir do endereço de origem do datagrama, que o inventário do site mantém nos mapas de `devices:` do fornecedor.

Para os parâmetros do coletor, a configuração do dispositivo para RouterOS e outros remetentes, a referência de rótulos e os passos de solução de problemas, consulte Syslog de Dispositivos de Rede.

Dashboards do Grafana

Dashboards pré-construídos são provisionados automaticamente para cada tipo de componente.

Dashboards Disponíveis

Dashboard	Localização	Descrição
Logs do CSCF	Logs → Logs do CSCF	Logs do P-CSCF, S-CSCF, I-CSCF (OmniCSCF)
Logs do MME	Logs → Logs do MME	Eventos de anexação/desanexação do MME e erros
Logs do SGW	Logs → Logs do SGW	Eventos de sessão e bearer do SGW
Logs do PGW	Logs → Logs do PGW	Eventos de PDN e sessão do PGW
Logs do HSS	Logs → Logs do HSS	Mensagens de Diameter do HSS e eventos de autenticação
Logs do OmniMessage	Logs → Logs do OmniMessage	Eventos de entrega de SMS e SMPP
Logs do OCS/CGrateS	Logs → Logs do OCS/CGrateS	Eventos de cobrança e tráfego JSONRPC
Chamadas RPC do CGrateS	Logs → Chamadas RPC do CGrateS	Pesquisar e correlacionar solicitações/respostas RPC com latência
Syslog de Dispositivos de Rede	Logs → Syslog de Dispositivos de Rede	Syslog de MikroTik / equipamentos de rede, por dispositivo e severidade

Recursos do Dashboard

Cada dashboard inclui:

- **Caixa de pesquisa**: Pesquisa em texto livre em todos os logs
- **Gráficos de taxa de log**: Volume ao longo do tempo por host e severidade
- **Seções de componente**: Visualizações agrupadas (por exemplo, P-CSCF, S-CSCF, I-CSCF)
- **Filtragem de erros**: Visualizações pré-filtradas para erros e falhas
- **Transmissão ao vivo**: Streaming de log em tempo real (atualização a cada 10 segundos)

Usando os Dashboards

1. Navegue até o Grafana (tipicamente `http://<monitoring-host>:3000`)
2. Vá para **Dashboards** → **Logs** → selecione o componente
3. Use a variável **Search** para filtrar logs
4. Ajuste o intervalo de tempo conforme necessário

Consultando Logs

Fundamentos do LogQL

O Loki usa LogQL para consultas. Sintaxe básica:

```
{label="value"} |= "search string"
```

Consultas Comuns

Todos os logs de um host específico:

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

Todos os logs do componente MME:

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

Pesquisar por erros em todos os CSCFs:

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

Filtrar por unidade systemd:

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

Combinar filtros:

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

Consultas de Taxa de Log

Volume de log por componente:

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

Taxa de erro para CSCF:

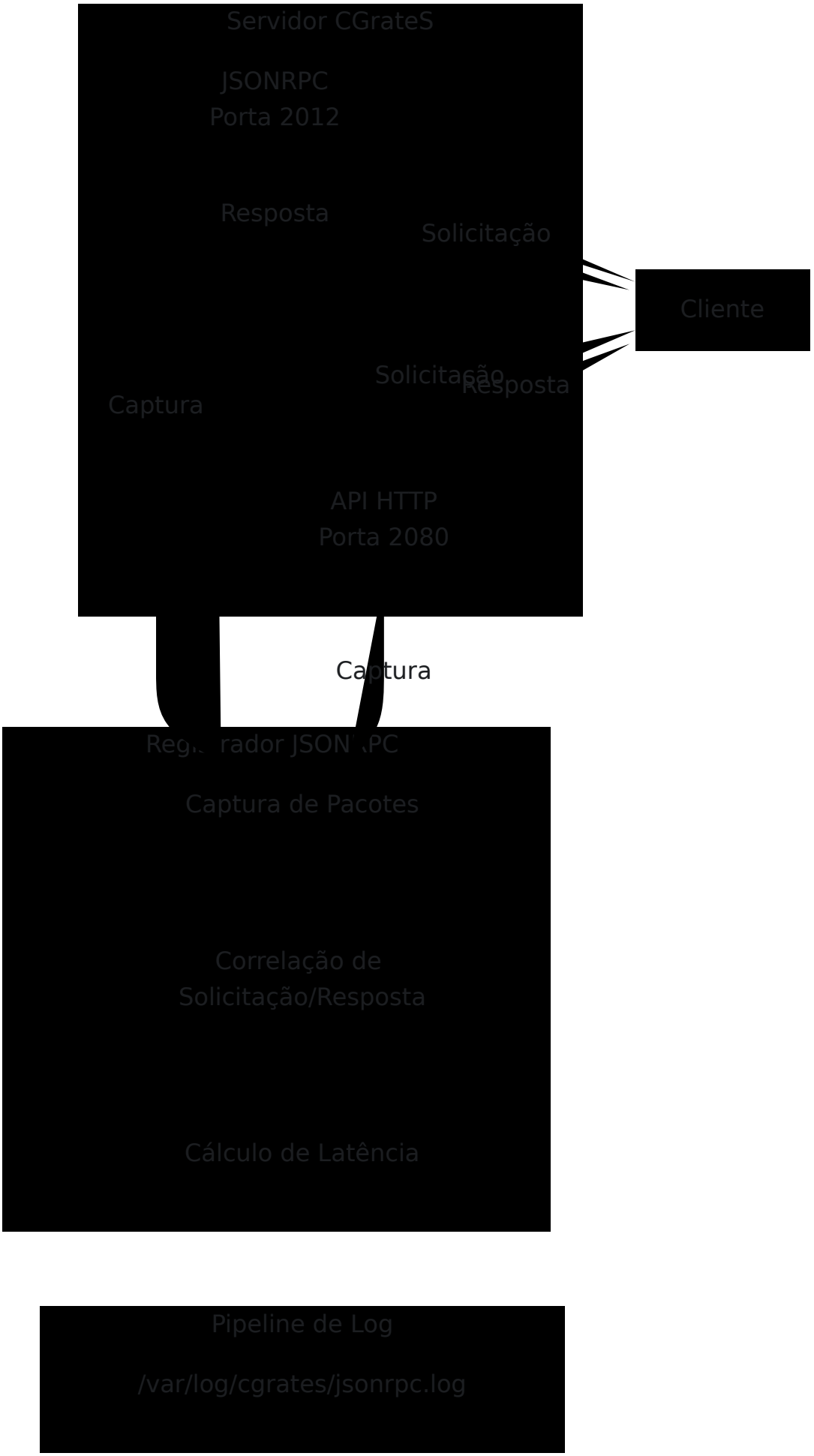
```
sum(rate({component=~"pcscf|scscf|icscf"} |~ "(?i)error" [5m]))
```

Registro JSONRPC do CGrateS

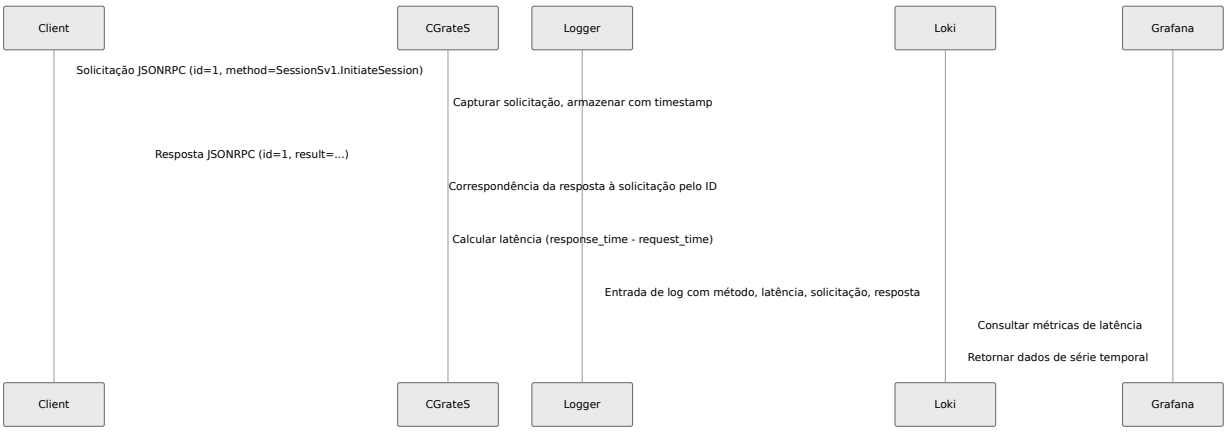
Para implantações OCS/CGrateS, o tráfego JSONRPC é capturado e registrado com correlação de solicitação/resposta e rastreamento de latência para análise de desempenho e depuração.

O registrador JSONRPC é definido no papel `cgrates` e seu arquivo de log (`/var/log/cgrates/jsonrpc.log`) é coletado pelo Alloy em ambos os grupos de hosts `ocs` e `cgrates`.

Arquitetura


Servidor CGrateS
JSONRPC
Porta 2012
Resposta
Solicitação
Cliente
Solicitação
Resposta
Captura
API HTTP
Porta 2080
Captura
Captura de Pacotes
Correlação de
Solicitação/Resposta
Cálculo de Latência
Pipeline de Log
/var/log/cgrates/jsonrpc.log

Fluxo de Solicitação/Resposta



O Que é Capturado

O tráfego nessas portas do CGrateS é capturado:

Porta	Nome do Serviço	Protocolo	Descrição
2012	`rpc_json`	TCP	JSONRPC bruto sobre TCP
2080	`http`	HTTP	API HTTP (filtra `/metrics` e `/health`)

O registrador filtra automaticamente:

- Coletas de métricas do Prometheus (`GET /metrics`)
- Solicitações de verificação de saúde (`GET /health`)
- Respostas comprimidas em gzip (dados binários)

Formato de Log

Os logs JSONRPC são escritos em `/var/log/cgrates/jsonrpc.log`. Cada entrada de log representa um **par de solicitação/resposta completo** com medição de latência:

```
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 Log

Campo	Descrição	Exemplo
`timestamp`	Timestamp ISO 8601 quando a resposta foi recebida	`2026-03-01T10:30:45.123456`
`port`	Porta do CGrateS para a qual a solicitação foi feita	`2012`, `2080`
`service`	Nome do serviço para a porta	`rpc_json`, `http`
`request_id`	ID da solicitação JSONRPC para correlação	`1`, `42`, (vazio para nulo)
`method`	Nome do método JSONRPC	`SessionSv1.InitiateSession`
`latency_ms`	Latência de ida e volta em milissegundos	`2.45`
`status`	Status do resultado	`ok`, `error`
`error`	Mensagem de erro se o status for erro	`INSUFFICIENT_CREDIT`
`src`	IP:porta de origem (cliente)	`10.10.1.50:45678`
`dst`	IP:porta de destino (CGrateS)	`10.10.2.100:2012`
`request`	Payload completo da solicitação JSONRPC (JSON compactado)	`{"id":1,"method":"..."}`

Campo	Descrição	Exemplo
`response`	Payload completo da resposta JSONRPC (JSON compactado)	`{"id":1,"result":{...}}`

Dashboard do Grafana

O dashboard **Logs OCS / CGrateS** inclui painéis dedicados para análise JSONRPC.

Latência JSONRPC por Método

Um gráfico de série temporal mostrando a latência para cada método JSONRPC ao longo do tempo. Útil para identificar:

- Métodos lentos que podem precisar de otimização
- Picos de latência indicando problemas no sistema
- Tendências de desempenho ao longo do tempo

A legenda exibe a latência média e máxima por método.

Tráfego JSONRPC

Um painel de logs mostrando todos os pares de solicitação/resposta JSONRPC com:

- Timestamp
- Nome do método
- Latência
- Payloads completos de solicitação e resposta (formatados)

Análise por Método

Filtrar o tráfego JSONRPC por um método específico usando a variável **Method** na parte superior do dashboard. Insira o nome do método (por exemplo, `SessionSv1.InitiateSession`) para ver apenas o tráfego para esse método.

Dashboard de Chamadas RPC do CGrateS

O dashboard dedicado **Chamadas RPC do CGrateS** fornece ferramentas focadas para solução de problemas de chamadas de API e correlação de solicitações com respostas.

Caso de Uso: Encontrando uma Chamada Falhada

Quando um usuário relata "Tentei ligar para 1234 e falhou":

1. Abra **Dashboards** → **Logs** → **Chamadas RPC do CGrateS**
2. No campo **Search**, insira o número de telefone ou ID da conta: `1234`
3. Defina **Status** como **Error** para ver apenas chamadas falhadas
4. O painel **Pesquisa de Chamadas RPC** mostra todas as chamadas correspondentes com solicitação/resposta completas

Cada entrada de log mostra o payload completo da solicitação (o que foi enviado) e o payload da resposta (o que o CGrateS retornou), facilitando a identificação do erro exato.

Variáveis do Dashboard

Variável	Propósito	Exemplo
Search	Pesquisa em texto livre nos payloads de solicitação/resposta	`1234`, `Account123`, `INSUFFICIENT_CREDIT`
Method	Filtrar por nome do método RPC	`SessionSv1.InitiateSession`, `CDRsV1.ProcessEvent`
Status	Filtrar por status de resultado	Todos, Sucesso, Erro

Painéis do Dashboard

Estatísticas Resumidas (linha superior):

- **Total de Chamadas RPC**: Contagem de todas as chamadas no intervalo de tempo
- **Erros**: Contagem de chamadas falhadas (codificação por cores: verde=0, amarelo=1-9, vermelho=10+)
- **Latência Média**: Tempo médio de ida e volta (limiares codificados por cores)
- **Latência Máxima**: Chamada com maior latência

Latência RPC por Método: Série temporal mostrando a latência para cada método. Passe o mouse para ver valores específicos.

Pesquisa de Chamadas RPC: Painel de log principal mostrando chamadas correspondentes. Clique em qualquer entrada para expandir e ver o JSON completo de solicitação/resposta.

Chamadas RPC Falhadas: Visualização pré-filtrada mostrando apenas chamadas `status=error`.

Distribuição por Método: Gráficos de pizza mostrando a distribuição de chamadas e a distribuição de erros por método.

Consultando Logs JSONRPC

Consultas Básicas

Todo o tráfego JSONRPC:

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

Filtrar por método:

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

Pesquisar payloads de solicitação/resposta:

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

Somente erros:

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

Análise de Latência

Extrair latência 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 solicitações lentas (latência > 100ms):

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

Consultas Específicas de Método

Processamento de CDR:

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

Gerenciamento de sessão:

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

Operações de conta:

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

Gerenciamento de Serviço

O registrador JSONRPC é executado como um serviço systemd em hosts OCS/CGrateS.

Verificar status do serviço:

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

Ver logs do serviço:

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

Reiniciar o serviço:

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

Solução de Problemas

Nenhum Log JSONRPC Aparecendo

Sintomas: O painel de Tráfego JSONRPC mostra nenhum dado

Possíveis causas:

- Serviço de registrador não está em execução
- ngrep não está instalado
- Desconexão da interface de rede
- Alloy não está coletando o arquivo de log

Resolução:

1. Verifique o status do serviço de registrador:

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

2. Verifique se o ngrep está instalado:

```
which ngrep
```

3. Verifique se o arquivo de log está sendo escrito:

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

4. Verifique se o Alloy está coletando o arquivo:

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

Gráfico de Latência Mostra Nenhum Dado

Sintomas: O painel de Latência JSONRPC por Método mostra "Sem dados"

Possíveis causas:

- Nenhum tráfego JSONRPC no intervalo de tempo selecionado
- Incompatibilidade de formato de log (regex não correspondendo)

Resolução:

1. Verifique se existem logs para o intervalo de tempo:

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

2. Verifique se o campo `latency_ms` está presente nos logs
3. Expanda o intervalo de tempo

Alta Contagem de Solicitações Pendentes

Sintomas: Logs de depuração mostram muitas solicitações armazenadas, mas poucas conclusões registradas

Possíveis causas:

- Solicitações sem respostas (cliente desconectado)
- Resposta capturada antes da solicitação (ordenação de pacotes)
- Incompatibilidade de ID entre solicitação e resposta

Resolução:

1. O registrador limpa automaticamente solicitações pendentes mais antigas que 60 segundos
2. Verifique se há problemas de rede entre o cliente e o CGrateS
3. Verifique se o CGrateS está respondendo às solicitações

Logs de Arquivo

Além dos logs do journal systemd, o Alloy coleta arquivos de log específicos:

Logs do FreeSWITCH (Servidores de Aplicação)

Caminho	Rótulo do Job
`/var/log/freeswitch/freeswitch.log`	`freeswitch`
`/var/log/omnitas-freeswitch.log`	`freeswitch-structured`

Logs do Nginx (OmniCRM)

Caminho	Rótulo do Job	Rótulo de Tipo
`/var/log/nginx/access.log`	`nginx`	`access`
`/var/log/nginx/error.log`	`nginx`	`error`

JSONRPC do CGrateS (OCS)

Caminho	Rótulo do Job
`/var/log/cgrates/jsonrpc.log`	`cgrates-jsonrpc`

Solução de Problemas

Logs Não Aparecendo nos Dashboards do Grafana

Sintomas: Nenhum log visível nos dashboards do Loki

Possíveis causas:

- Serviço Alloy não está em execução no host de origem
- Serviço Loki não está em execução no host de monitoramento
- Conectividade de rede entre hosts bloqueada
- Alloy não consegue alcançar o Loki na porta 3100

Resolução:

1. Verifique o status do Alloy no host de origem:

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

2. Verifique o status do Loki no host de monitoramento:

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

3. Teste a conectividade do host de origem para o monitoramento:

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

4. Verifique se o firewall permite TCP 3100 de todos os hosts para monitoramento

Alto Uso de Disco do Alloy

Sintomas: `/var/lib/alloy` consumindo espaço significativo em disco

Possíveis causas:

- Loki inacessível por um período prolongado
- Buffer WAL enchendo

Resolução:

1. Verifique a conectividade com o Loki (veja acima)
2. Se o Loki estiver fora do ar, os logs são armazenados localmente no WAL
3. Assim que o Loki estiver acessível, os logs armazenados são enviados automaticamente
4. Segmentos armazenados mais antigos que 1 hora são descartados (por design)

Armazenamento do Loki Cheio

Sintomas: Loki para de aceitar logs, disco cheio no servidor de monitoramento

Possíveis causas:

- Volume de logs excede o disco disponível antes que a retenção exclua logs antigos
- Compactação de retenção não está em execução

Resolução:

1. Verifique o uso de armazenamento do Loki:

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

2. Verifique se o compactador está em execução (verifique os logs do Loki)

3. Acione manualmente a compactação se necessário reiniciando o Loki
4. Considere aumentar a alocação de disco ou reduzir o período de retenção

Rótulos de Componente Faltando

Sintomas: Logs aparecem, mas o rótulo `component` é `infrastructure` em vez do valor esperado

Possíveis causas:

- Host não está no grupo de inventário esperado
- Configuração do Alloy não foi regenerada após a mudança no inventário

Resolução:

1. Verifique se o host está no grupo de inventário correto
2. Execute novamente o Ansible para regenerar a configuração do Alloy:

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

3. Reinicie o Alloy:

   ```
   systemctl restart alloy
   ```

Portas de Serviço

Serviço	Porta	Protocolo	Descrição
Loki	3100	HTTP	Ingestão de logs e API de consulta
Loki	9096	gRPC	gRPC interno (não usado externamente)
Alloy	12345	HTTP	Métricas e UI do Alloy
Grafana	3000	HTTP	Acesso ao dashboard

Documentação Relacionada

- Monitoramento & Observabilidade: Grafana, Prometheus, dashboards e alertas
- Arquitetura de Implantação: Arquitetura geral do sistema
- Configuração do Arquivo de Hosts: Configuração do inventário
- Referência de Configuração: Referência completa de parâmetros

Núcleo Comutado por Circuito (CS)

O domínio legado comutado por circuito (voz 2G/3G e SMS) é implantado pelo playbook `services/cs.yml`. Ele provisiona o BSC voltado para rádio, o softswitch OmniMSC e a pilha de sinalização SS7/SIGTRAN.

Visão Geral

`cs.yml` executa três funções, cada uma contra seu próprio grupo de inventário:

Componente	Função	Grupo de inventário	Pacotes	De
BSC	`circuit_switched/bsc`	`bsc`	BSC, MGW	Con de I Bas Gat Míd test vali 2G/
MSC	`circuit_switched/omnimsc`	`msc`	`omnimsc`	Soft do (Con Móv
SS7	`omniss7`	`ss7`	`omniss7`	Pilh SS7 (STI SMS CAM test

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

BSC (`circuit_switched/bsc`)

Instala um **BSC** e **Gateway de Mídia (MGW)** do servidor Omnitouch APT para testes e validação comutados por circuito. A função configura ambos os serviços.

- Renderiza os arquivos de configuração do BSC e MGW a partir dos templates da função.
- Instala uma sobreposição systemd para o serviço MGW que **desabilita o agendamento de CPU em tempo real**. Limpa `CPUSchedulingPolicy`, `Priority` e `AmbientCapabilities`. Máquinas virtuais não suportam agendamento em tempo real.
- Manipuladores reiniciam cada serviço. Cada manipulador então **verifica se o serviço atinge `ativo`** (6 tentativas, 5 s de intervalo) antes de continuar.

MSC (`circuit_switched/omnimsc`)

Instala o pacote `omnimsc` e o configura de forma semelhante a um NF 5GC:

- Resolve `group_vars_omnicore` e deriva `license_server_api_urls` do grupo `license_server`.
- Renderiza o `runtime.exs.j2` embutido para `/etc/omnimsc/`. Se `omnimsc_template_config` estiver definido, renderiza um template personalizado de `group_vars/` em vez disso.
- Garante um diretório de saída CDR (`omnimsc_cdr_output_dir`, padrão `/var/lib/omnimsc/cdr`).
- Gera um certificado TLS autoassinado para a interface web / API.
- Usa o padrão de symlink de configuração versionada + `runtime.exs`, reiniciando em caso de alteração.

SS7 (`omniss7`)

Uma única função que instala o pacote `omniss7` e renderiza **um template de configuração por função SS7**, selecionado pela variável de host `ss7_role`:

`ss7_role`	Template	Função
`stp`	`stp.j2`	Ponto de Transferência de Sinal (roteamento SS7)
`hlr`	`hlr.j2`	Registro de Localização do Usuário
`smsc`	`ipsmgw.j2`	SMSC / IP-SM-GW
`camel`	`camel.j2`	Controle de serviço CAMEL
`tester`	`tester.j2`	Agente de teste/carga de sinalização

Seletores adicionais:

- Definindo `ss7_protocol: m2pa` renderiza `m2pa.j2` (um endpoint M2PA) em vez do template `tester`.
- Definindo `ss7_template_config` (um caminho sob `group_vars/`) substitui **todos** os templates embutidos por uma configuração específica do host.

Como os outros componentes Elixir, `omniss7` usa o padrão de symlink de configuração versionada + `runtime.exs` e espelha o symlink em `/opt/omniss7/releases/*`.

Documentação Relacionada

- Playbooks de Serviço - Todos os playbooks de serviço e a hierarquia de implantação
- Núcleo 5G - O núcleo 5G Autônomo e a função compartilhada `5gc_nf`
- Configuração de Variáveis de Grupo - Templates de configuração personalizados (por exemplo, `ss7_template_config`, `omnimsc_template_config`)
- **Configuração do OmniCore**: https://docs.omnitouch.com.au/docs/repos/OmniCore - O que vai nos arquivos de configuração do componente

configuration-reference

```
---
last_update:
  date: "2026-09-13T09:23:02Z"
---

# Referência de Configuração

## Visão Geral

Este documento fornece uma referência abrangente para configurar
implantações do OmniCore através de arquivos de hosts. A
configuração é definida principalmente em arquivos de inventário
de hosts, com mínimas substituições de group_vars necessárias para
implantações modernas.

Para documentação específica do produto, veja:
- **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

## Abordagem de Configuração

Implantações modernas do OmniCore usam um modelo de configuração
simplificado:

```mermaid
graph LR
    A[Hosts File] --> B[Ansible Roles]
    B --> C[Component Configuration]
    C --> D[Deployed Services]

    style A fill:#e3f2fd
    style B fill:#c8e6c9
    style C fill:#fff3e0
    style D fill:#a5d6a7
```
```

Princípio Chave: A maior parte da configuração é definida diretamente no arquivo de hosts. Os padrões de função lidam com muitas configurações. No entanto, alguns componentes são configurados exclusivamente a partir de `group_vars` e não têm padrão de função. O OmniCRM é o exemplo notável (veja abaixo). Nesses casos, um valor ausente falha a execução de forma clara, em vez de voltar a um padrão.

> **Nota de Segurança**: Vários padrões de função são marcadores inseguros destinados apenas para uso em laboratório e **devem** ser substituídos para qualquer implantação real. Exemplos incluem senhas de banco de dados (por exemplo, `sql_pass: password`, `omnitouch2024`) e credenciais de repositório APT (`omni`/`omni`). Sempre defina segredos reais e únicos em seu inventário ou group_vars antes de implantar.

Planejamento de Rede

Antes de configurar os hosts, revise o Padrão de Planejamento de IP para orientações sobre:

- Estratégias de segmentação de rede
- Alocação de endereços IP
- Organização de sub-redes
- Tratamento de IPs públicos

Parâmetros Comuns de Host

#ToDo - Apenas diga para verificar hosts-file-configuration.md para isso

Flags Específicas de Serviço

```
cdrs_enabled: True                # Habilitar geração de CDR
in_pool: False                    # Excluir do pool de
balanceamento de carga
populate_crm: False               # Preencher o CRM com dados
iniciais
dra_peer_exclude: True            # Excluir de listas de pares
Diameter geradas
gtpc_weight: 100                  # Parte do SGW/PGW dentro de seu
nível de prioridade (seleção de pares MME)
gtpc_priority: 100                # Nível de prioridade do SGW/PGW,
menor é preferido
```

Veja Exclusão de Pares Diameter para `dra_peer_exclude`.

`gtpc_weight` e `gtpc_priority` são lidos de cada host `sgwc` / `pgwc` quando o MME constrói sua lista estática de pares GTP-C. Ambos têm padrão `100`, então um par pode ser ponderado ou classificado acima ou abaixo do restante do pool sem mudar cada outro host. O MME escolhe o nível de `gtpc_priority` *mais baixo* presente e divide o tráfego dentro dele por `gtpc_weight` (RFC 2782); níveis mais altos são standby e só recebem tráfego uma vez que nenhum par permaneça em um nível inferior. Uma capacidade relativa do MME é definida da mesma forma, com `relative_capacity` no host MME (padrão `100`, intervalo 0-255).

Variáveis Globais (all:vars)

A seção `all:vars` contém configurações em todo o ambiente de implantação. Implantações modernas usam variáveis globais mínimas, com muitas configurações cobertas por padrões de função. Note as exceções abaixo (OmniCRM, Grafana SMTP) onde a configuração deve ser fornecida explicitamente porque não existe padrão de função.

Variáveis Globais Essenciais

Autenticação e Acesso

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

Alternativa: Use chaves SSH em vez de senhas:

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

Identidade do Cliente

```
customer_name_short: omnitouch
customer_legal_name: "Customer Lab"
site_name: Customer-Site
region: AU
TZ: Australia/Melbourne
```

Configuração do PLMN

```
plmn_id:
  mcc: '001'              # Código do País Móvel (3 dígitos)
  mnc: '01'              # Código da Rede Móvel (2-3 dígitos)
  mnc_longform: '001'     # MNC com zeros à esquerda (sempre 3
dígitos)

diameter_realm: epc.mnc&#123;&#123; plmn_id.mnc_longform
&#125;&#125;.mcc&#123;&#123; plmn_id.mcc
&#125;&#125;.3gppnetwork.org
```

Propósito: Identifica exclusivamente sua rede móvel. Usado para construção do domínio Diameter.

Nomes de Rede

```
network_name_short: Omni
network_name_long: Omnitouch
tac_list: [10100,100]            # Lista TAC padrão (pode substituir por MME)
```

Exibido: Nomes de rede mostrados em dispositivos UE em Configurações > Rede Móvel.

Configuração de DNS

```
manage_resolv_conf: True         # Defina como False para impedir que o Ansible gerencie /etc/resolv.conf
```

Nota: Quando `manage_resolv_conf` está definido como `False`, o Ansible não sobrescreverá `/etc/resolv.conf` nesse host. Isso é útil para hosts que requerem configuração DNS personalizada ou são gerenciados por sistemas externos. Pode ser definido por host no inventário ou globalmente em `all:vars`.

Configuração do Repositório APT

`apt_repo` é a única fonte de verdade. Defina-o uma vez. As variáveis de download binário (`remote_apt_*`) e a URL `apt_download_base` são derivadas automaticamente dele. Você não as define separadamente em um inventário normal.

Padrões Automáticos: Quando um grupo `apt_cache_servers` é definido com hosts:

- `use_apt_cache` automaticamente tem padrão `True` (a menos que explicitamente definido como `False`)
- `apt_repo.apt_server` automaticamente tem padrão para o IP do primeiro servidor de cache

```yaml
use_apt_cache: False             # Acesso direto ao repositório
(True = cache APT local)

apt_repo:
  apt_server: "apt.&lt;region&gt;.omnitou.ch"  # FQDN fornecido
pela Omnitouch (ou IP do cache)
  apt_repo_username: "your-username"     # Necessário para acesso
direto
  apt_repo_password: "your-password"     # Necessário para acesso
direto
  # apt_repo_port: 443                   # Opcional; padrão para
443 (HTTPS).
                                         # O esquema segue a
porta: 443 -> https, senão http.
```

O esquema e a porta têm padrão para **HTTPS / 443** para acesso direto e **HTTP / 8080** em modo de cache. Apenas um servidor de espelho de cache (`apt_cache_servers` grupo) define `remote_apt_*` manualmente, para descrever o servidor upstream da Omnitouch do qual ele sincroniza.

Veja: Sistema de Cache APT

Servidor de Licença

```yaml
# Opcional: auto-derivado do grupo de inventário `license_server`
quando não definido.
# Cada host nesse grupo se torna
https://&lt;ansible_host&gt;:8443/api.
license_server_api_urls: ["https://10.10.2.150:8443/api"]
license_enforced: false          # O padrão efetivo é falso
(roles/omnihss/defaults/main.yml)
```

Nota: Se um grupo `license_server` existir no inventário, `license_server_api_urls` é derivado automaticamente de seus membros; você só precisa defini-lo manualmente para substituir esse comportamento.

Veja: Servidor de Licença

Configurações do MME

O MME seleciona pares S-GW e P-GW usando um método configurável, definido por gateway sob `omnimme`. O método pode ser um único valor ou dividido por tráfego `home`/`roaming`. Os valores válidos são `dns_lookup` (resolver pares via DNS) e `random_peer` (escolher entre pares configurados).

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

Configurações do AMF

```
# tac_list também impulsiona a supported_ta_list do AMF. Cada TAC
aqui é
# anunciado para gNBs conectados durante a Configuração NG. Se não
definido, o AMF tem padrão
# para apenas TAC 1, que rejeita qualquer gNB que não transmita
TAC 1.
tac_list: [1, 3]

# Desativar a cifragem NAS (forçar NEA0). Tem padrão para
verdadeiro porque omniamf
# >=20260727 negocia NEA2, mas não consegue decifrar NAS uplink
cifrado (Modo de Segurança
# Completo e mensagens subsequentes), causando a falha de cada
registro 5G. Defina como falso uma vez que o fornecedor confirme
que a decifragem de uplink NEA2 funciona
# contra UEs reais. A integridade NAS (NIA2) não é afetada.
amf_force_null_ciphering: true
```

Configurações do SAEGW

```
mtu: 1400                        # Unidade Máxima de Transmissão
```

Configurações do OmniHSS

O OmniHSS constrói sua lista de pares Diameter automaticamente a partir do inventário. Ele usa os grupos `dra`, `mme`, `pgwc`, `scscf`, `icscf`, `pcscf` e `applicationserver` (quais deles são usados depende de `packet_core_dra_support` / `ims_dra_support`). `exclude_diameter_peers` remove hosts individuais dessa lista gerada, para pares que estão no inventário, mas não devem ser pares Diameter do HSS.

```
omnihss:
  exclude_diameter_peers:          # opcional; padrão para []
(nada excluído)
    - customer-as01
    - customer-as02
```

- As entradas são correspondidas contra **`inventory_hostname`**, não o FQDN Diameter ou IP.
- A exclusão se aplica a cada grupo auto-descoberto, então uma lista cobre um host independentemente de qual grupo ele pertence.
- Os nomes devem existir no inventário. O papel afirma isso antes de implantar, então um erro de digitação falha a execução de forma clara, em vez de deixar silenciosamente o par em `/etc/omnihss/runtime.exs`.
- Defina onde o restante do dicionário `omnihss` é definido (por host, ou sob `vars:` de um grupo). Como o Ansible substitui em vez de mesclar dicionários, ele deve viver no mesmo bloco `omnihss` que as outras chaves.
- Isso afeta apenas a lista de pares Diameter. A lista de pares SIP S-CSCF sob `config :hss, ims:` é não relacionada e não é filtrada por esta chave.

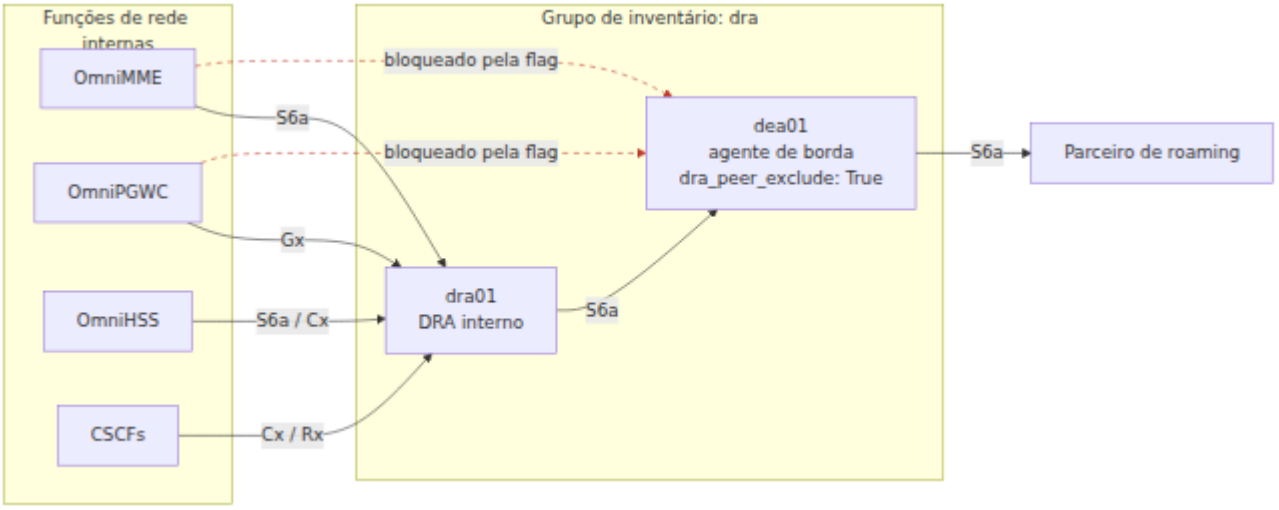
> `exclude_diameter_peers` é uma chave do dicionário `omnihss`. Ele filtra apenas a lista de pares do OmniHSS. Para remover um host da lista de pares de cada função de rede, use `dra_peer_exclude` em vez disso. Os dois mecanismos são independentes.

Exclusão de Pares Diameter

O OmniCore constrói a lista de pares Diameter de cada função de rede a partir dos grupos de inventário. Uma função de rede adiciona cada host dos grupos

com os quais ela se conecta. `dra_peer_exclude` remove um host dessas listas geradas. Defina a flag no host que não deve se tornar um par Diameter.

O caso de uso principal é um **Agente de Borda Diameter (DEA)**. Um DEA é um membro do grupo `dra`, porque ele executa o mesmo software OmniDRA e precisa do mesmo tratamento de implantação, firewall e monitoramento. Um DEA transporta tráfego apenas para parceiros de roaming externos. Cada função de rede interna deve enviar seu tráfego para o DRA interno. Defina `dra_peer_exclude: True` em cada DEA para manter o DEA fora da lista de pares das funções de rede internas.



Configuração

Defina a flag na entrada do host em si, ao lado de `ansible_host`.

```
dra:
  hosts:
    dra01:                  # DRA interno. Um par de cada função de
rede interna.
      ansible_host: 10.4.2.150
    dea01:                  # Agente de borda. Nunca um par de uma
função de rede interna.
      ansible_host: 203.0.113.10
      packet_core_dra_support: False
      ims_dra_support: False
      dra_peer_exclude: True
```

Parâmetro	Tipo	Requerido	Padrão	
`ansible_host`	String	Sim	-	Endereço I do host. Ta do par em gerada.
`dra_peer_exclude`	Booleano	Não	`False`	Quando `Tr` ignora este uma lista d partir de u inventário.
`packet_core_dra_support`	Booleano	Não	`False`	Quando `Tr` conecta ac interfaces (S6a, Gx). host se co HSS e ao P defina com DEA obtém `dra_modul`
`ims_dra_support`	Booleano	Não	`False`	Quando `Tr` conecta ac interfaces um DEA, d

Escopo

A flag funciona em duas direções. Uma função de rede ignora um host excluído do grupo `dra`, e o DRA ignora um host excluído de qualquer outro grupo.

Funções de rede que filtram o grupo `dra`:

Papel	Lista de pares
`omnimme`	Pares S6a e T6a
`omnipgwc`	Pares Gx e Gy
`omnihss`	Pares S6a, Cx e Sh
`omniepdg`	Pares SWm e S6b
`omnitwag`	Pares S6b
`omniscef`	Pares T6a
`omnisep`	Pares Sh
`omnilcs`	Pares SLg e SLh
`cscf`	Pares Cx, Rx e Dx do P-CSCF, I-CSCF e S-CSCF
TAS e SMSC (site `group_vars`)	Pares Sh e Ro

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

Regras de Comportamento

- Defina a flag no host a ser excluído. Não a defina no host que constrói a lista de pares.
- A flag se aplica apenas a listas de pares geradas. Um par estático explícito nunca é filtrado. Isso cobre `dra_peers` no DRA e `diameter_static_peers` em uma função de rede.
- A flag não altera as regras de firewall, os alvos do Prometheus ou o plano de IP. Um host excluído mantém seu acesso à rede e seu monitoramento.

- A flag entra em vigor na próxima execução do papel. Execute o papel de cada função de rede afetada após definir a flag. O `runtime.exs` implantado nesse host mantém o par antigo até então.

Sintomas de uma exclusão ausente

Se uma função de rede interna se conectar a um DEA, verifique esta flag primeiro. Os sintomas são:

- O MME ou o PGW-C abre uma conexão SCTP para o endereço IP público do DEA.
- O DEA registra uma tentativa de conexão de par não autorizada, porque `allow_undefined_peers_to_connect` é `False`.
- A função de rede relata um par em um loop de reconexão permanente.

Para corrigir isso, defina `dra_peer_exclude: True` no host DEA. Em seguida, execute o papel da função de rede novamente. Depois, confirme que a lista de pares no `runtime.exs` implantado contém apenas os DRAs internos.

Configurações do OmniCRM

A configuração do OmniCRM é obtida **exclusivamente** dos `group_vars` do cliente (sob `hosts/`); nenhum padrão é incorporado ao papel para segredos do site.

Caminho preferido (`omnicrm_deploy_mode: package`, o padrão do papel): instalar pacotes genéricos `omnicrm-api` / `omnicrm-ui` do `apt_repo`, modelar `crm_config.yaml` para `/etc/omnicrm/crm_config.yaml`, sincronizar `deployment_assets/`, e `systemctl restart omnicrm-api`. A marcação da UI vem do bloco `ui:` (runtime `GET /crm/public/ui-config`). Este caminho não requer **nenhuma** construção yarn no host e **nenhum** `crm_env_override`.

Caminho legado (`omnicrm_deploy_mode: source`): git archive + venv do host + construção yarn; `crm_env_override` é necessário apenas para esse modo.

```yaml
omnicrm_deploy_mode: package     # package (padrão) | source
omnicrm_package_version: ""      # versão opcional do apt para
omnicrm-api/ui
crm_config.yaml                  # API + ui: config (group_vars;
necessário)
deployment_assets/               # logos/icones opcionais para
/etc/omnicrm/deployment_assets
crm_env_override: my-site.env    # apenas necessário para modo
source / fallback local da UI
cell_sites: cellSites.txt        # opcional: dados do site celular
para a UI
message_templates: constants.js  # opcional: modelos de mensagens
de transmissão celular
site_data: site_data.json        # opcional: limites geográficos
da transmissão celular
```

Configurações SMTP do Grafana

Usado pelo papel de monitoramento para o email de alerta do Grafana. `grafana_smtp_user` e `grafana_smtp_password` são **necessários** (sem padrão: você deve defini-los); `grafana_smtp_host` tem padrão para `in-v3.mailjet.com:587`. Veja `roles/monitoring/templates/grafana/grafana.ini`.

```yaml
grafana_smtp_user: "smtp-user"           # NECESSÁRIO, sem padrão
grafana_smtp_password: "smtp-password"   # NECESSÁRIO, sem padrão
grafana_smtp_host: "in-v3.mailjet.com:587"  # opcional, este é o
padrão
```

Configurações IMS

```yaml
ims_dra_support: False           # Roteia IMS através do DRA
sipcapture_client_enabled: False # Enviar SIP/RTP para a caixa de
captura VoIPmonitor (grupo siptrace)
```

Configuração do 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
```

Configuração de Firewall

Aplicado por `services/firewall.yml`. O conjunto completo de parâmetros, as zonas de plano e os modos de monitoramento/aplicação estão documentados em Firewall por Papel; apenas `allowed_oam_subnets` é obrigatório.

```yaml
firewall:
  allowed_oam_subnets:            # fonte de gerenciamento (SSH,
UI, scrape, APIs de mgmt)
    - '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 para o P-
CSCF
    - '10.0.1.0/24'
  allowed_carrier_voice_subnets:  # SIP de interconexão de
operadora
    - '10.0.1.0/24'
  zones:                          # zonas de plano — veja
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 específico da operadora (baseado em PLMN)
  123456:                         # Exemplo Operadora 1
    - '10.10.2.197'
  654321:                         # Exemplo Operadora 2
    - '10.10.0.4'
```

Usuários Locais (Chaves SSH)

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

Configuração do Hypervisor

Proxmox

Um site é implantado como VMs ou contêineres LXC, selecionados por entrada `proxmoxServers` via `deployment_type` (padrão `vm`). Todas as entradas em um site devem concordar; misturar falha na validação. Veja Implantação Proxmox para o guia completo.

Site VM

```yaml
proxmoxServers:
  customer-prxmx01:
    # deployment_type omitido → padrão para "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        # usuário cloud-init
opcional; padrão para a primeira chave local_users
    proxmoxTemplatePassword: omnitouch    # senha cloud-init
opcional; padrão para o nome de usuário cloud-init
    storage: SSD_RAID0            # opcional, armazenamento padrão
da VM
```

Site LXC

```yaml
proxmox_lxc_nameserver: "1.1.1.1"  # opcional, injetado nos LXCs
na criação

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 do
rootfs
```

Substituições em nível de grupo (ambos os tipos)

```yaml
dns:
  vars:
    proxmox_interface: vmbr0              # required: bridge
    gateway: 10.10.0.1                    # required
    netmask: 255.255.255.0                # required
    vlanid: 100                           # optional
    proxmoxLxcCores: 2                    # optional (LXC only)
    proxmoxLxcMemoryMb: 4096              # optional (LXC only)
    proxmoxLxcDiskSizeGb: 30              # optional (LXC only)
    proxmoxLxcRootFsStorageName: SSD_RAID0 # optional (LXC only)
    host_vm_network: vmbr1                # optional bridge
override
```

VMware vCenter

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

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

netmask: 255.255.255.0
```

Documentação Relacionada

- Padrão de Planejamento de IP - Arquitetura de rede e diretrizes de alocação de IP
- Configuração do Arquivo Hosts - Como estruturar arquivos hosts
- Configuração de Variáveis de Grupo - Quando e como usar group_vars
- Configuração do Netplan - IPs secundários e configuração de múltiplas NICs
- Arquitetura de Implantação - Como os componentes interagem
- Sistema de Cache APT - Gerenciamento de pacotes
- Servidor de Licença - Configuração de licença

Documentação do Produto

Para guias operacionais detalhados e configuração avançada:

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

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

Visão Geral da Arquitetura de Implantação

Visão Geral

Este documento fornece uma visão completa de como o software de rede celular da Omnitouch Network Services é implantado usando Ansible, mostrando como todos os componentes se encaixam para criar uma rede 4G/5G funcional.

Consulte o Padrão de Planejamento de IP para diretrizes detalhadas sobre a colocação de componentes, atribuição de endereços IP e manuseio de IPs públicos.

Exemplo Completo de Implantação

0. Provisionamento de Infraestrutura (Opcional)

Para implantações Proxmox, provisionar VMs/LXCs antes da configuração:

```
# Implantar VMs e/ou contêineres LXC no Proxmox (o tipo de
implantação por host decide qual)
ansible-playbook -i hosts/Customer/hosts.yml
util_playbooks/proxmox.yml
```

Veja: Implantação de VM/LXC no Proxmox

1. Definição da Infraestrutura (Arquivo de Hosts)

```yaml
# Definir o que implantar e onde
mme:
  hosts:
    customer-mme01:
      ansible_host: 10.10.1.15

hss:
  hosts:
    customer-hss01:
      ansible_host: 10.10.2.140

# ... todos os outros componentes
```

Veja: Configuração do Arquivo de Hosts

2. Personalização (group_vars)

A pasta `group_vars` armazena quaisquer substituições de configuração necessárias em nível de host, site ou rede.

Por exemplo, uma pasta contém sua configuração do OmniMessage SMSc. Os troncos SIP aos quais seu TAS se conecta estão aqui. Toda a sua lógica de Roteamento Diameter também está aqui.

O OmniCRM também é configurado aqui. Suas configurações são apenas `group_vars`. Seu código-fonte do GitHub é clonado no controlador Ansible (`delegate_to: localhost`) com as credenciais do próprio controlador antes de ser sincronizado com o destino, para que nenhuma credencial chegue ao host de destino.

Veja: Configuração de Variáveis de Grupo

3. Distribuição de Pacotes (Cache APT)

```yaml
# Configurar de onde obter pacotes
apt_repo:
  apt_server: "10.254.10.223"  # IP do servidor de cache ou
servidor de repositório direto
use_apt_cache: false  # true = usar cache local, false = acesso
direto ao repositório
```

Veja: Sistema de Cache APT

4. Configuração de Licença

```yaml
# Apontar componentes para o servidor de licença
license_server_api_urls: ["https://10.10.2.150:8443/api"]
license_enforced: true
```

Veja: Servidor de Licença

5. Executar Implantação

Você pode implantar componentes individuais. Por exemplo, execute `services/twag.yml`. O playbook `services/all.yml` cuida de tudo. Use `--limit=myhost` ou `--limit=mmee,sgw` para limitar os hosts em que você trabalha.

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

# Ou implantar 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 Implantação

Existem duas maneiras de obter o kit de ferramentas OmniCore em um controlador e executá-lo.

Modelo A: Clonar o repositório no controlador

O modelo tradicional: clone este repositório no seu controlador Ansible, mantenha seu inventário em `hosts/` e execute os playbooks diretamente do checkout. Este é o modelo assumido pelo exemplo de implantação acima.

Modelo B: `omnicore` .deb auto-provisionado

O OmniCore também é empacotado como um pacote Debian `omnicore` autônomo, para que um controlador possa instalar o kit de ferramentas a partir do apt em vez de clonar o repositório.

- **O que ele instala.** O pacote (`name: omnicore`, veja `packaging/nfpm.yaml`) fornece os papéis Ansible, playbooks, coleções *vendored*, um interpretador Python empacotado (`/opt/omnicore/python`) e um wheelhouse offline para `/opt/omnicore`. A etapa pós-instalação (`packaging/postinstall.sh`) constrói um venv autônomo (`/opt/omnicore/venv`) a partir do interpretador empacotado + wheelhouse e cria links simbólicos dos `ansible*` empacotados no `PATH` via `/usr/local/bin`, para que o controlador execute as versões exatas de `uv.lock` do Ansible e suas bibliotecas Python: sem Ansible de distribuição, sem Python do sistema, sem rede. Apenas ferramentas genuínas do sistema (`git`, `rsync`, `openssh-client`) são puxadas pelo apt. O pacote é **amd64** e tem como alvo **Ubuntu 22.04+** (glibc ≥ 2.35).
- **Como é construído.** `packaging/build.sh` organiza o repositório em `/opt/omnicore` (excluindo `hosts/`, `.git/`, virtualenvs e arquivos de construção), fornece as coleções Ansible de `requirements.yml`, empacota o interpretador autônomo + um wheelhouse das dependências Python bloqueadas, executa uma varredura de segredos com falha fechada e, em seguida, constrói o `.deb` com nfpm.
- **Como é publicado.** `.github/workflows/build-deb.yml` executa `build.sh` em cada push para `main` e publica o `.deb` (mais um

`_metadata.json` como lado) como um Release do GitHub. O script `discover_github_debs.py` do servidor apt descobre automaticamente esse ativo de release em toda a organização Omnitouch e o adiciona ao aptly. Nenhuma alteração no papel apt é necessária.

Fluxo do cliente:

```
# 1. Instalar o kit de ferramentas a partir do apt
apt-get install omnicore

# 2. Fornecer seu próprio inventário
#    (o pacote não fornece um - ele vive fora de /opt/omnicore)
vi /etc/omnicore/inventory

# 3. Executar playbooks do kit de ferramentas instalado
cd /opt/omnicore
ansible-playbook -i /etc/omnicore/inventory services/all.yml
```

Como o inventário vive em `/etc/omnicore/inventory` (separado do kit de ferramentas instalado em `/opt/omnicore`), atualizar o pacote nunca toca sua definição de infraestrutura ou personalização.

Documentação Relacionada

- Introdução à Implantação Ansible - Começando
- Playbooks de Serviço - **Referência e hierarquia de playbooks**
- Configuração do Arquivo de Hosts - Definindo infraestrutura
- Padrão de Planejamento de IP - **Arquitetura de rede e alocação de IP**
- Configuração de Variáveis de Grupo - Personalização
- Sistema de Cache APT - Gerenciamento de pacotes
- Servidor de Licença - Gerenciamento de licença
- Monitoramento e Observabilidade - Grafana, Prometheus, alertas e painéis
- Registro Centralizado - Coleta de logs do Loki e Alloy

Documentação do Produto

Para informações detalhadas sobre a configuração de cada componente:

- **OmniCore** (Núcleo de Pacotes 4G/5G): https://docs.omnitouch.com.au/docs/repos/OmniCore
  - OmniHSS, OmniSGW, OmniPGW, OmniUPF, OmniDRA, OmniTWAG
- **OmniCall** (Voz e Mensagens): https://docs.omnitouch.com.au/docs/repos/OmniCall
  - OmniTAS, OmniCall CSCF, OmniMessage, OmniSS7, VisualVoicemail
- **OmniCharge/OmniCRM** (Faturamento): https://docs.omnitouch.com.au/docs/repos/OmniCharge
- **Documentação Principal**: https://docs.omnitouch.com.au/

Firewall por Papel

O firewall por papel fornece a cada host um conjunto de regras `iptables` e `ip6tables` que é derivado de um único modelo declarativo do tráfego da rede, em vez de regras escritas à mão por host. Um arquivo descreve cada fluxo permitido no núcleo (quem pode alcançar qual papel, em qual protocolo e porta, e por quê). A partir desse modelo, a plataforma produz duas coisas que sempre concordam entre si:

- o **conjunto de regras aplicadas** em cada host, e
- uma **matriz de comunicações / portas** (CSV) para revisão e para entrega a um firewall upstream.

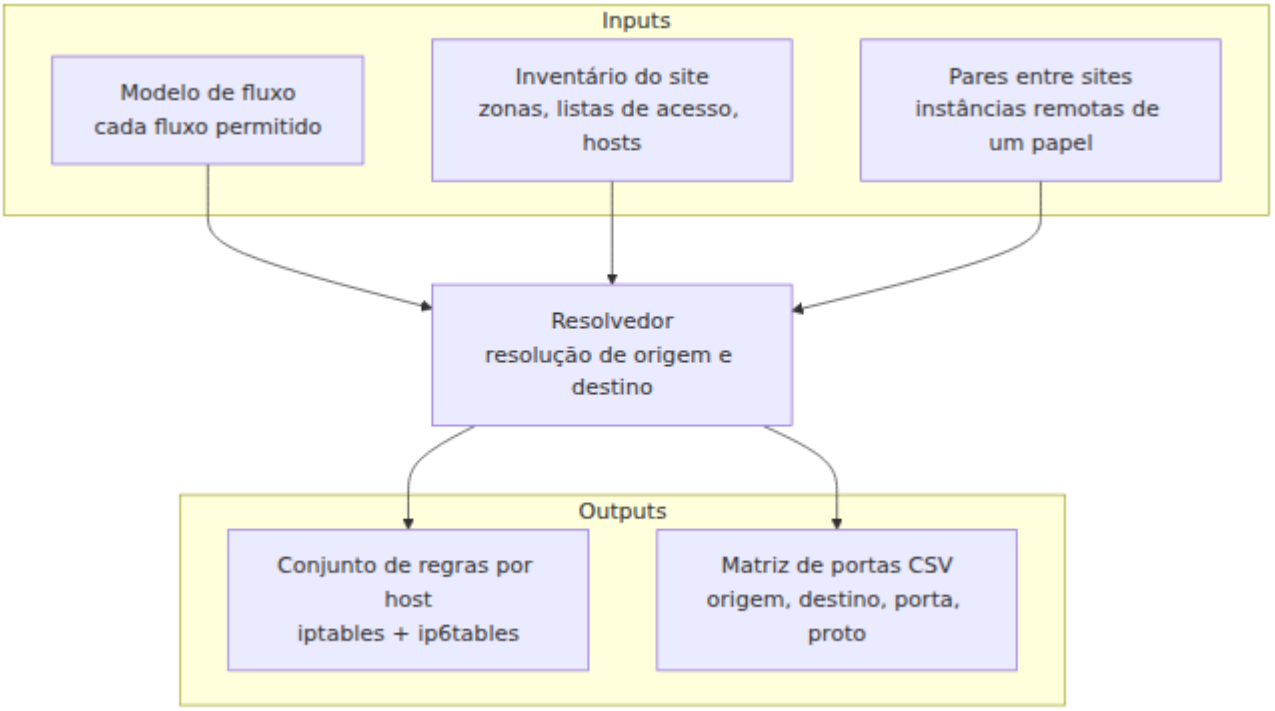
As regras funcionam em um de dois modos: **monitorar** (registrar cada fluxo, não bloquear nada) ou **aplicar** (padrão de negação). O bloqueio é deliberado e à prova de falhas, e cada aplicação é protegida por um rollback automático no host, de modo que um conjunto de regras incorreto não possa trancar os operadores.

Índice

- Visão Geral da Arquitetura
- O Modelo de Fluxo
- Tipos de Fonte
- Catálogo de Serviços
- Configuração do Site
- Modos Operacionais
- Pares entre Sites
- Resolução de Endereços
- Matriz de Comunicações / Portas
- Proteção contra Bloqueio
- IPv6 e Dual-Stack
- Registro
- Cautelas Operacionais

Visão Geral da Arquitetura

Um único modelo mais o feed de inventário de cada site alimentam um resolvedor. O resolvedor produz tanto as regras no host quanto a matriz de portas, de modo que a documentação nunca pode descrever algo diferente do que é realmente aplicado.



O tráfego descartado e auditado é registrado através do pipeline de observabilidade padrão:

O Modelo de Fluxo

O modelo de fluxo é a única fonte de verdade para **quem pode alcançar o quê**. Ele possui duas partes: um conjunto de padrões estruturais (que não carregam endereçamento do site) e a lista de fluxos. Um fluxo não diz nada sobre números de porta diretamente - ele nomeia um **serviço** exposto pelo papel de destino (veja Catálogo de Serviços), e a porta é consultada na declaração desse papel:

```yaml
- id: '13'                        # rótulo estável, aparece no
comentário de cada regra
  dst: hss                        # grupo de inventário ao qual a
regra é aplicada
  service: diameter               # serviço nomeado exposto pelo
papel hss (-> 3868/tcp+sctp)
  purpose: 'Diameter'             # descrição, mostrada na matriz
  sources: [{ zone: internal }]   # quem pode alcançá-lo (lista de
tokens de origem)
```

Parâmetro	Tipo	Obrigatório	Padrão	Descrição
`id`	String	Sim	-	Identificador estável para o fluxo. Aparece no comentário de cada regra gerada como `ans:<dst>:<id>` para que uma regra ativa possa ser rastreada de volta ao modelo.
`dst`	String	Sim	-	O grupo de inventário em que a regra se aplica, ou os pseudo-destinos `all` (todos os hosts) e `five_g_sbi` (os grupos de funções de rede 5G). Um fluxo cujo grupo de destino não está implantado em um site não produz regras nem linhas na matriz.
`service`	String	Sim*	-	Um serviço nomeado declarado pelo papel de destino (veja Catálogo de Serviços). As porta(s) são consultadas na declaração do papel, portanto, o fluxo nunca repete um número de porta.
`proto_ports`	Lista	Sim*	-	A alternativa literal a `service`, para fluxos não vinculados a um

Parâmetro	Tipo	Obrigatório	Padrão	Descrição
				único papel (por exemplo, fluxos de `all`-host). Cada entrada é `PORT/proto` (`tcp`, `udp`, `sctp`; a porta pode ser um intervalo `LO-HI`) ou um protocolo nu (`icmp`, `esp`, `proto 89`).
`purpose`	String	Sim	-	Descrição humana do fluxo. Mostrada na coluna `service` da matriz de portas.
`sources`	Lista	Sim	-	As fontes permitidas, como uma lista de tokens de origem. Veja Tipos de Fonte.

* Um fluxo fornece **ou** `service` **ou** `proto_ports`, não ambos.

Tipos de Fonte

Cada entrada na lista `sources` de um fluxo é um token de chave única que é resolvido contra a configuração e o inventário do site. Um token que não resolve em nada (por exemplo, uma lista de acesso vazia) não contribui com nenhuma regra: ele nunca é emitido como um espaço reservado.

Token	Resolve para
`{ zone: <name> }`	Os CIDRs daquela zona de plano (`firewall.zones.<name>`).
`{ list: <name> }`	Uma lista de acesso do operador, por exemplo `allowed_oam_subnets`.
`{ group: <name> }`	Os endereços dos hosts do grupo de inventário, além dos pares entre sites desse papel. Veja Pares entre Sites.
`{ ue_pools: true }`	Os pools de UE / assinantes do site (`firewall.ue_pools`).
`{ dra_peers: true }`	Cada endereço configurado na própria lista de pares Diameter do host DRA.
`{ net_devices: true }`	Cada endereço de dispositivo de rede gerenciado (`all.vars.<vendor>.devices`, por exemplo `mikrotik.devices`).
`{ roaming: true }`	Intervalos de roaming (IR.21 / GRX-IPX). Atualmente, qualquer fonte: veja Cautelas Operacionais.
`{ literal: "..." }`	Uma fonte de texto livre que não é derivada do inventário.

Catálogo de Serviços

Os números de porta são **de propriedade do papel que os abre**, não do modelo central. Cada papel que expõe portas envia um `firewall_services.yml` em seu diretório de papel, chaveado pelos grupos de inventário que atende (um papel como `cscf` atende a vários grupos, então ele declara cada um). Isso mantém a definição da porta próxima ao código que a 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
outros NFs)
    postgres: 5432/tcp                # replicação lógica entre
pares HSS
```

O resolvedor agrupa o `firewall_services.yml` de cada papel em um único catálogo. O `service:` de um fluxo é consultado no catálogo para o grupo `dst` desse fluxo, de modo que o modelo expressa apenas a **relação** (quem pode alcançar o serviço `diameter` do `hss`) enquanto a **porta** vive com o papel. Adicione ou mova um listener em um papel e atualize seu `firewall_services.yml`; nada no modelo central muda.

Essa divisão tem uma rede de segurança que um modelo apenas central não possui: uma **verificação de desvio** compara o que os papéis declaram com o que os hosts estão realmente ouvindo.

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

Para cada host, ele relata:

- **não declarado** - um listener ativo em uma porta que nenhum papel declarou (ou adicioná-lo ao `firewall_services.yml` do papel, ou é um listener inesperado), e
- **não ouvindo** - um serviço declarado que não está realmente ativo.

Hosts cujos grupos não declaram serviços são ignorados, portanto, a verificação é útil enquanto os papéis são migrados para o catálogo de forma incremental, em vez de todos de uma vez.

Listeners condicionais. Algumas portas são abertas apenas quando outro papel está implantado: o MME abre suas portas LCS (SBc-AP `29168/sctp`, SLs `9082/sctp`) apenas quando um host `omnilcs` está no inventário. Uma

declaração estática não pode expressar essa condição, então tais portas são anotadas no `firewall_services.yml` do papel, mas não declaradas. Em um site que executa o papel dependente, a verificação de desvio as relata como não declaradas, que é o aviso correto para revisar e permitir.

Configuração do Site

Cada site declara um bloco `firewall` sob `all.vars` em seu inventário. Não há fallbacks de endereço: um site deve descrever sua própria rede. Qualquer coisa que um site não declarar é simplesmente vazia, e nenhum endereçamento de outro ambiente é herdado.

O exemplo abaixo está totalmente preenchido para que cada campo mostre um valor representativo. Os endereços usam intervalos de documentação e um layout coerente por plano (cada plano é sua própria sub-rede dentro da super-rede `internal`), que é o que torna as zonas úteis: elas permitem que um fluxo permita, digamos, apenas o plano IMS em vez de todo o núcleo.

```yaml
firewall:
  allowed_oam_subnets:           ['10.8.0.0/24']       # fonte de
gerenciamento (SSH, UI, scrape, APIs de gerenciamento)
  allowed_ran_subnets:           ['10.20.0.0/16']      # fonte de
sinalização RAN (S1AP, NGAP, GTP-U, Abis)
  allowed_ue_voice_subnets:      ['100.64.0.0/16']     # SIP de UE
externo para o P-CSCF
  allowed_carrier_voice_subnets: ['203.0.113.0/24']    # SIP de
interconexão de operadora
  zones:                                               # zonas de
plano: sub-rede(s) real(is) de cada plano
    internal:    ['10.8.0.0/16']                       #   super-
rede de núcleo confiável
    mobile_core: ['10.8.1.0/24']                       #   NFs de
núcleo móvel (EPC + 5GC + núcleo CS)
    core_svc:    ['10.8.2.0/24']                       #   plano
de serviços centrais (HSS, DRA, OCS, OAM)
    ims:      ['10.8.3.0/24']                          #   plano
IMS (CSCF, TAS)
    ue_svc:   ['10.8.4.0/24']                          #   plano
acessível por UE (P-CSCF, DNS, ePDG)
    core_v6:  ['2001:db8:0:1::/64']                    #
endereços IPv6 de NFs centrais (não UEs)
  ue_pools: ['100.64.0.0/16', '2001:db8:64::/48']      #
intervalos de IP de UE / assinantes (v4 e v6)
  extra_rules:                                         # regras de
aceitação manuais (veja abaixo)
    - { proto: tcp, dport: '8443', source: '203.0.113.10/32',
comment: 'API do parceiro' }
```

Apenas `allowed_oam_subnets` é obrigatória. Cada outra lista pode ser deixada vazia (`[]`) em um site que não possui essa função: um site sem RAN define `allowed_ran_subnets: []`, um site sem núcleo IPv6 define `core_v6: []`, e um site sem pools de UE define `ue_pools: []`. Uma lista vazia simplesmente não produz regras para os fluxos que a utilizam (veja a coluna `Obrigatório` abaixo).

Parâmetro	Tipo	Obrigatório	Padrão	Des
`allowed_oam_subnets`	Lista	Sim	-	Sub-re fonte gerenc Gover interfa OmniV scrape Prome as API gerenc Se vaz gerad com u porque valor deixar gerenc sem u permit causar bloque aplica
`allowed_ran_subnets`	Lista	Não	`[]`	Sub-re fonte sinaliz RAN (S NGAP, Abis). vazia site se
`allowed_ue_voice_subnets`	Lista	Não	`[]`	Sub-re fonte

Parâmetro	Tipo	Obrigatório	Padrão	Des
				de UE para o
`allowed_carrier_voice_subnets`	Lista	Não	`[]`	Sub-re fonte de interc de ope
`zones`	Mapa	Sim	-	Zonas mapea cada r plano sua(s) rede(s Veja Parâm Zona.
`ue_pools`	Lista	Não	`[]`	Interv de UE assina Vazio não ti nenhu
`extra_rules`	Lista	Não	`[]`	Regra aceita manua aplica cada Parâm Regra Manua

Parâmetros de Zona

Cada zona mapeia um nome de plano para uma lista de CIDRs. Defina apenas os planos que o site utiliza; deixe os demais vazios.

Zona	Descrição
`internal`	Super-rede de núcleo confiável. Fonte de sinalização interna mais ampla.
`mobile_core`	Funções de rede de núcleo móvel através de gerações: EPC (MME, S/P-GW, UPF), 5GC (AMF, SMF e a malha SBI), e núcleo CS (MSC, SS7/SIGTRAN).
`ims`	Plano IMS (I-CSCF, S-CSCF, TAS).
`ue_svc`	Plano de serviços acessíveis por UE (P-CSCF, DNS, ePDG).
`core_svc`	Plano de serviços centrais (HSS, DRA, OCS, OAM).
`core_v6`	Endereços de serviços centrais IPv6. Vazio se o site não tiver núcleo IPv6.

Parâmetros de Regras Manuais

`extra_rules` (global, todos os hosts) e as `firewall_extra_rules` por host anexam regras de aceitação brutas para casos que o modelo não cobre. Um `source` contendo `:` é aplicado em IPv6.

Parâmetro	Tipo	Obrigatório	Padrão	Descrição
`proto`	String	Não	-	Protocolo, por exemplo `tcp`, `udp`, `sctp`.
`dport`	String	Não	-	Porta ou intervalo de destino.
`source`	String	Não	-	CIDR de origem. Um `:` no valor aplica a regra em IPv6.
`comment`	String	Não	`override`	Rótulo registrado no comentário da regra.

Modos Operacionais

O modo de aplicação é controlado por `firewall_mode`.

Modo	Comportamento
`monitor`	Cada regra de aceitação é instalada, a política `INPUT` permanece `ACCEPT`, e o tráfego que passa é registrado (como `FW-AUDIT`) e, em seguida, permitido. Nada é bloqueado. Use isso para validar o modelo contra o tráfego real antes de aplicar.
`enforce`	O tráfego que passa é registrado (como `FW-DROP`) e, em seguida, descartado, e as políticas `INPUT` e `FORWARD` são definidas como `DROP`.

O bloqueio é opt-in e à prova de falhas: apenas o valor exato `enforce` realmente descarta tráfego. Qualquer outro valor, incluindo uma variável não definida ou um erro de digitação, é tratado como monitor, de modo que um host não pode começar a descartar tráfego por acidente. Isso se aplica em três níveis: o padrão da plataforma é `monitor`, os sites declaram

`firewall_mode: monitor` em seu inventário, e o gerador trata qualquer coisa que não seja `enforce` como monitor. Aplicar requer uma mudança deliberada.

```
all:
  vars:
    firewall_mode: monitor   # mude para 'enforce' apenas depois que o log de auditoria confirmar a cobertura
```

`firewall_mode` pode ser definido por site sob `all.vars`, ou substituído por grupo ou por host através da precedência normal de variáveis.

Pares entre Sites

Papéis que se federam entre sites (CSCF, TAS, OmniMessage, replicação HSS, OCS) listam seus pares remotos no arquivo de pares do site (por exemplo, `prod_master_peers.yml`). Esses endereços remotos são incorporados na resolução `{ group: <role> }`, de modo que um fluxo que permite um papel cobre automaticamente tanto instâncias do mesmo site quanto entre sites sob uma política idêntica. A mesma resolução é usada para as regras e para a matriz, então os pares aparecem em ambos.

Resolução de Endereços

Um host é resolvido para **cada endereço com o qual é declarado**, não apenas seu primário. Isso significa `ansible_host` mais cada `secondary_ips.<name>.ip_address` no inventário. Nós de datapath e sinalização vinculam serviços a NICs secundárias: o UPF responde N3 GTP-U em seu endereço `n3`, e o STP recebe e origina SIGTRAN em seu endereço público. Tanto as regras quanto a matriz cobrem todos esses endereços, como fontes e como destinos, de modo que um par que permite um nó pelo endereço fornecido está permitindo o endereço que o tráfego realmente usa.

Um host que pertence a **vários grupos de inventário** recebe a **união** de todos os fluxos e serviços declarados de cada grupo. Um nó convergente expõe tudo para todos os papéis que executa, e seu conjunto esperado de listeners (usado pela verificação de desvio) é a união entre seus grupos.

Como a resolução usa todos os endereços declarados em vez de rastrear qual serviço vincula qual NIC, a matriz pode listar um serviço em um endereço que não está realmente vinculado (por exemplo, N3 GTP-U mostrado no endereço N6 do UPF). Isso é uma sobre-aproximação deliberada: uma matriz completa, mas ligeiramente ampla, é segura para entregar a um firewall upstream, enquanto omitir um endereço real de datapath ou SIGTRAN quebra silenciosamente o tráfego. Se a precisão por endereço for necessária mais tarde, um serviço pode nomear a NIC à qual se vincula; até lá, a resolução permanece baseada em fatos e completa.

Matriz de Comunicações / Portas

A matriz de portas é o modelo de fluxo resolvido contra o inventário de um site, achatado para uma linha por fonte, host de destino e única porta/protocolo. É a forma para revisar ou entregar a um firewall upstream. Gere-a com:

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

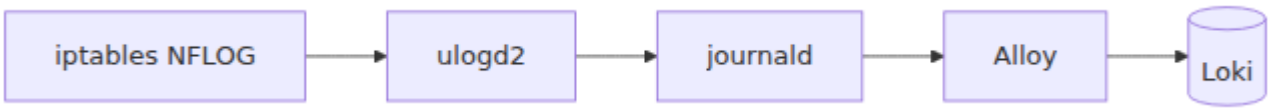
Isso escreve `generated/firewall-port-matrix-exploded.csv` (por site, não comprometido). As colunas são:

Coluna	Descrição
`id`	O identificador do fluxo.
`service`	O propósito do fluxo.
`source`	Um CIDR ou endereço de origem resolvido.
`destination_role`	O grupo de inventário de destino.
`destination_host`	O nome do host de destino.
`destination_ip`	O endereço do host de destino.
`port`	A porta ou intervalo de destino (vazio para entradas apenas de protocolo, como ICMP ou ESP).
`protocol`	O protocolo.

Uma linha aparece apenas quando seu destino está implantado no site e sua origem resolve para um endereço real. Papéis não implantados e fontes não resolvidas não produzem linhas.

Proteção contra Bloqueio

A aplicação de regras é protegida por um rollback automático, controlado por `firewall_rollback_guard` (padrão ativado) e `firewall_rollback_timeout`:



1. O conjunto de regras ativo é capturado.
2. Um temporizador é armado no host que restaurará a captura. Ele sobrevive à sessão de gerenciamento sendo cortada.
3. As novas regras são aplicadas.

4. O controlador restabelece uma nova conexão e executa uma verificação trivial. Se as novas regras o trancaram, isso falha e a execução para para aquele host.
5. Se a conectividade for confirmada, o temporizador é desarmado. Se não, o temporizador dispara e restaura o conjunto de regras de trabalho sem ação do operador.

Parâmetro	Tipo	Obrigatório	Padrão	Des
`firewall_rollback_guard`	Booleano	Não	`true`	Habilita captura tempor verifica recone torno d aplicaç
`firewall_rollback_timeout`	Inteiro	Não	`120`	Segund tempor espera restaur automa a captu

IPv6 e Dual-Stack

A família de endereços é inferida por CIDR: um `:` no valor significa IPv6, e a regra é aplicada em `ip6tables`. Uma zona ou lista contendo apenas CIDRs IPv4, portanto, não produz regras IPv6 para essa fonte. Sites sem núcleo IPv6 deixam as zonas e pools IPv6 vazias (por exemplo, `core_v6: []`). Sob aplicação, a política `INPUT` IPv6 é definida como `DROP`, então valide o tráfego IPv6 em modo de monitoramento antes de aplicar um host cuja única entrada IPv6 legítima chega através de uma fonte apenas IPv4.

Registro

O tráfego auditado e descartado é registrado e transportado pelo pipeline padrão para Loki. Três prefixos são usados:

Prefixo	Significado
`FW-FLOW` / `FW-FLOW6`	Uma nova conexão vista em modo de monitoramento. Constrói o mapa de fluxo ativo do tráfego realmente em uso.
`FW-AUDIT` / `FW-AUDIT6`	Tráfego que passa em modo de monitoramento: seria descartado sob aplicação, mas é permitido aqui. Revise isso antes de aplicar.
`FW-DROP` / `FW-DROP6`	Tráfego que realmente foi descartado sob aplicação.

Alvo de Log

Os registros são escritos com `NFLOG` e lidos por `ulogd2`, que o papel instala como uma exigência dura ao lado de `iptables-persistent`. NFLOG só é legível enquanto um consumidor está executando — sem `ulogd2`, o kernel descarta cada registro, e o modo de monitoramento então não prova nada enquanto parece funcionar: as regras estão instaladas, os contadores sobem, e nenhum log é nunca produzido. Essa falha é silenciosa, razão pela qual o pacote é obrigatório em vez de melhor esforço.

Ambos os pacotes são transportados no repositório apt do Omnitouch (`ubuntu_packages.networking`), portanto, um host que não pode instalá-los tem um repositório quebrado e a execução para ali.

O pipeline do diário Alloy rotula esses como `unit="ulogd2.service"` e `service="ulogd"`. Note que não há rótulo `job="firewall"` — `job` carrega o nome do inventário do host, então selecione em `unit` para abranger todos os hosts.

Exemplo de consultas Loki:

```
# Tráfego que seria bloqueado uma vez que esses hosts sejam
aplicados
{unit="ulogd2.service"} |= "FW-AUDIT"

# Tráfego sendo descartado sob aplicação, por host
sum by (hostname) (count_over_time({unit="ulogd2.service"} |= "FW-
DROP" [5m]))

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

No próprio host:

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

Cautelas Operacionais

- **Fontes roaming e literais são atualmente de qualquer fonte.** Fluxos cuja fonte é `roaming` ou `literal` são emitidos sem restrição de fonte (`0.0.0.0/0`). Trate-os como abertos até que os intervalos de roaming sejam resolvidos para CIDRs reais.
- **ICMP é aceito de qualquer fonte.** O conjunto de regras gerado permite ICMP incondicionalmente, o que é mais amplo do que o escopo de gerenciamento mostrado para o fluxo de ping.
- **Não aplique um host de plano de usuário sem regras de encaminhamento.** Sob aplicação, a política `FORWARD` é definida como `DROP`, e nenhuma regra de encaminhamento é gerada. Aplicar em um UPF, PGW ou SGW sem adicionar regras de encaminhamento quebraria o plano de usuário. Valide primeiro.
- **A lista de gerenciamento é grosseira.** `allowed_oam_subnets` governa SSH, a interface, coleta, e as APIs de gerenciamento juntas. Um site que coleta de uma rede diferente daquela de onde administra precisa de uma entrada `extra_rules` para separá-las.

Configuração de Variáveis de Grupo

Visão Geral

O diretório `group_vars` é onde você armazena arquivos de configuração personalizados que substituem os modelos padrão.

É aqui que suas configurações específicas do cliente residem: troncos SIP, regras de roteamento Diameter, lógica de roteamento SMS, planos de discagem e quaisquer outras personalizações onde você não deseja a configuração padrão. Ele reside em `group_vars`.

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

Os papéis resolvem este diretório em tempo de execução como `group_vars_omnicore` - o diretório `group_vars/` que fica adjacente ao diretório de inventário (`{{ inventory_dir.rsplit('/', 1)[0] }}/group_vars/`). Sempre que um caminho de arquivo abaixo faz referência a `{{ group_vars_omnicore }}`, ele aponta para este diretório.

Como Funciona

Os papéis do Ansible têm modelos de configuração padrão. Para personalizar para uma implantação específica, coloque seus arquivos personalizados em `group_vars` e faça referência a eles em seu arquivo de hosts.

```
Modelo Padrão do Papel → Substituição do group_vars (se
especificado) → Configuração Implantada
```

Exemplo 1: Modelo de Configuração Personalizado (OmniMessage)

Alguns componentes aceitam modelos de configuração Jinja2 personalizados.

Estrutura do Arquivo

```
hosts/Customer/
└── group_vars/
    └── smsc_controller.exs        # Seu modelo de configuração
personalizado
```

Referência no Arquivo 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   # Referencie o
nome do seu arquivo de modelo em group_vars
```

O que acontece:

1. O Ansible encontra `smsc_template_config: smsc_controller.exs`
2. Procura em `hosts/Customer/group_vars/smsc_controller.exs`
3. Faz o template com Jinja2 (pode usar `{{ inventory_hostname }}`, `{{ plmn_id.mcc }}`, etc.)
4. Implanta em `/etc/omnimessage/runtime.exs`
5. Reinicia o serviço

Sem `smsc_template_config`, o modelo padrão do papel é usado.

Detalhes da configuração: Veja https://docs.omnitouch.com.au/docs/repos/OmniCall

---

Exemplo 2: Coleções de Arquivos de Configuração (OmniTAS Gateways & Planos de Discagem)

Alguns componentes usam diretórios de arquivos de configuração.

Estrutura do Arquivo

```
hosts/Customer/
└── group_vars/
    ├── gateways_prod/              # Configurações do gateway SIP
    │   ├── gateway_carrier1.xml
    │   ├── gateway_carrier2.xml
    │   └── gateway_emergency.xml
    ├── gateways_lab/               # Gateways de laboratório
    │   └── gateway_test.xml
    ├── dialplan/                   # Regras de roteamento de
chamadas (padrão)
    │   ├── mo_dialplan.xml         # Origem Móvel (saída)
    │   ├── mt_dialplan.xml         # Terminado Móvel (entrada)
    │   └── emergency.xml
    └── dialplan_lab/               # Planos de discagem de
laboratório
        └── mo_dialplan.xml
```

Referência no Arquivo de Hosts

```yaml
applicationserver:
  hosts:
    customer-tas01:
      ansible_host: 10.10.3.60
      gateway: 10.10.3.1
      host_vm_network: "vmbr3"
      gateways_folder: "gateways_prod"   # Referencie sua pasta de gateways a ser usada neste host
      dialplan_folder: "dialplan"         # Opcional - padrão para "dialplan" se não definido
```

O que acontece:

1. O Ansible encontra `gateways_folder: "gateways_prod"`
2. Copia todos os arquivos de `hosts/Customer/group_vars/gateways_prod/` para `/etc/freeswitch/sip_profiles/`
3. Copia todos os arquivos de `hosts/Customer/group_vars/dialplan/` (ou a pasta especificada por `dialplan_folder`) para o diretório de modelos do OmniTAS
4. Os serviços carregam as configurações

Ambientes diferentes: Use pastas diferentes por ambiente:

- `gateways_folder: "gateways_lab"`
- `gateways_folder: "gateways_prod"`
- `gateways_folder: "gateways_customer_specific"`
- `dialplan_folder: "dialplan_lab"`
- `dialplan_folder: "dialplan_prod"`

Detalhes da configuração: Veja https://docs.omnitouch.com.au/docs/repos/OmniCall

---

Exemplo 3: Modelo de Configuração Personalizado (OmniHSS)

O Servidor de Assinantes Domiciliares aceita modelos de configuração em tempo de execução personalizados.

Estrutura do Arquivo

```
hosts/Customer/
└── group_vars/
    └── hss_runtime.exs.j2        # Seu modelo de configuração HSS
personalizado
```

Referência no Arquivo 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   # Referencie o
nome do seu arquivo de modelo em group_vars
```

O que acontece:

1. O Ansible encontra `hss_template_config: hss_runtime.exs.j2`
2. Procura em `hosts/Customer/group_vars/hss_runtime.exs.j2`
3. Faz o template com Jinja2 (pode usar `{{ inventory_hostname }}`, `{{ plmn_id.mcc }}`, etc.)
4. Implanta em `/etc/omnihss/runtime.exs`
5. Reinicia o serviço

Sem `hss_template_config`, o modelo padrão do papel é usado.

Detalhes da configuração: Veja https://docs.omnitouch.com.au/docs/repos/OmniCore

---

Exemplo 4: Modelo de Configuração Personalizado (OmniMME)

A Entidade de Gerenciamento de Mobilidade aceita modelos de configuração em tempo de execução personalizados.

Estrutura do Arquivo

```
hosts/Customer/
└── group_vars/
    └── mme_runtime.exs.j2        # Seu modelo de configuração MME
personalizado
```

Referência no Arquivo 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   # Referencie o
nome do seu arquivo de modelo em group_vars
```

O que acontece:

1. O Ansible encontra `mme_template_config: mme_runtime.exs.j2`
2. Procura em `hosts/Customer/group_vars/mme_runtime.exs.j2`
3. Faz o template com Jinja2 (pode usar `{{ inventory_hostname }}`, `{{ plmn_id.mcc }}`, etc.)

4. Implanta em `/etc/omnimme/runtime.exs`
5. Reinicia o serviço

Sem `mme_template_config`, o modelo padrão do papel é usado.

Detalhes da configuração: Veja https://docs.omnitouch.com.au/docs/repos/OmniCore

---

Exemplo 5: Arquivos de Configuração Necessários (OmniCRM)

O OmniCRM é obtido **exclusivamente** de group_vars - não há padrão de papel para retornar. A configuração da API não é intencionalmente incorporada ao papel, de modo que um arquivo ausente falhe a execução de forma clara, em vez de enviar silenciosamente os segredos de outro site.

Modo de implantação preferido: pacotes apt

Com `omnicrm_deploy_mode: package` (padrão do papel), o Ansible instala pacotes genéricos `omnicrm-api` e `omnicrm-ui` do `apt_repo`, faz o template da configuração em `/etc/omnicrm/`, e reinicia `omnicrm-api` via systemctl. A UI **não** é reconstruída no host; a marca vem do bloco `ui:` em `crm_config.yaml` (servido como `GET /crm/public/ui-config`).

Estrutura do Arquivo

```
hosts/Customer/
└── group_vars/
    ├── crm_config.yaml          # API + ui: configuração
(obrigatório)
    ├── deployment_assets/       # Logotipos/ícones opcionais →
/etc/omnicrm/deployment_assets
    │   └── logos/
    ├── customer_crm.env         # Opcional; apenas para
omnicrm_deploy_mode: source ou desenvolvimento local da UI
    ├── cellSites.txt            # Dados de sites de células
opcionais
    ├── constants.js             # Modelos de mensagens de
transmissão de células opcionais
    └── site_data.json           # Limites geográficos de
transmissão de células opcionais
```

Referência no Arquivo de Hosts

```yaml
omnicrm:
  hosts:
    customer-crm01:
      ansible_host: 10.10.3.70
      gateway: 10.10.3.1
      host_vm_network: "vmbr3"
      omnicrm_deploy_mode: package          # padrão; use source para git+yarn legados
      # omnicrm_package_version: "20260810...."  # versão apt opcional
      logo_light: logo-light.png            # copiado para deployment_assets/logos/
      logo_dark: logo-dark.png
      login_background: login-bg.jpg        # → deployment_assets/login/auth-one-bg.jpg (tempo de execução)
      login_splash: splash.jpg              # → deployment_assets/login/login-side.jpg (mantém a extensão de origem)
      ssl_fqdn: portal.example.com
      # crm_env_override: customer_crm.env  # apenas necessário para o modo source
      cell_sites: cellSites.txt             # Opcional
      message_templates: constants.js       # Opcional
      site_data: site_data.json             # Opcional
```

O que acontece (modo pacote):

1. O Apt instala `omnicrm-api` e `omnicrm-ui`.
2. Faz o template de `{{ group_vars_omnicore }}/crm_config.yaml` para `/etc/omnicrm/crm_config.yaml` (**obrigatório**).
3. Sincroniza `deployment_assets/` (e opcionais `logo_*` / `login_*`) para `/etc/omnicrm/deployment_assets/`.
4. Reinicia `omnicrm-api`; o nginx serve a UI de `/usr/share/omnicrm-ui`.

A tela de login / fundo de autenticação é servida em tempo de execução via `GET /crm/public/assets/login/…` (veja `ui.branding.login_splash` / `login_background` em `crm_config.yaml`). Eles não são incorporados no `.deb`

da UI. A cópia mantém a extensão de origem (`splash.jpg` → `login/login-side.jpg`); a API então serve qualquer arquivo `login-side.*` que exista.

O que acontece (modo source, legado):

1. Faz o template de `crm_config.yaml` para `/etc/omnicrm/OmniCRM-API/`.
2. Faz o template de `{{ crm_env_override }}` para o UI `.env` (**obrigatório** no modo source).
3. Constrói a UI no host com yarn.

Importante: `crm_config.yaml` deve estar presente. Para o modo pacote, inclua um bloco `ui:` e defina `ui.assets_dir: '/etc/omnicrm/deployment_assets'`.

Detalhes da configuração: Veja https://docs.omnitouch.com.au/docs/repos/OmniCore

---

Exemplo 6: Diretório de Dados de Provisionamento (OmniHSS)

O serviço de provisionamento OmniHSS (`services/provision_omnihss.yml`) lê dados de assinantes e perfis de um diretório dedicado `omnihss/` dentro de group_vars.

Estrutura do Arquivo

```
hosts/Customer/
└── group_vars/
    └── omnihss/
        ├── subscribers.csv             # Assinantes a serem
provisionados
        ├── 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
```

O que acontece:

1. O serviço lê `{{ group_vars_omnicore }}/omnihss/subscribers.csv` para a lista de assinantes a serem provisionados.
2. Ele lê os arquivos JSON acompanhantes do mesmo diretório `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 os dados de APN, QoS, IMS, EPC, roaming, fluxo, cobrança e PCRF aplicados a esses assinantes.

Detalhes da configuração: Veja https://docs.omnitouch.com.au/docs/repos/OmniCore

---

Exemplo 7: Credenciais SMTP do Grafana (Monitoramento)

O modelo `grafana.ini` do papel de monitoramento conecta as configurações de email de alerta do Grafana (SMTP) a partir de group_vars. O usuário e a senha **não têm padrão de papel** - eles não são intencionalmente incorporados, para que um valor ausente falhe a execução de forma clara, em vez de enviar credenciais.

Referência no Arquivo de Hosts

```yaml
monitoring:
  hosts:
    customer-oam01:
      ansible_host: 10.10.3.135
      gateway: 10.10.3.1
      host_vm_network: "vmbr3"
      grafana_smtp_user: "smtp-user"          # OBRIGATÓRIO - sem
padrão
      grafana_smtp_password: "smtp-password"  # OBRIGATÓRIO - sem
padrão
      grafana_smtp_host: "in-v3.mailjet.com:587"  # Opcional -
padrão para in-v3.mailjet.com:587
```

O que acontece:

1. `grafana_smtp_user` e `grafana_smtp_password` são feitos em template na seção `[smtp]` de `/etc/grafana/grafana.ini`. Ambos são obrigatórios - não há fallback.
2. `grafana_smtp_host` é opcional e padrão para `in-v3.mailjet.com:587` se não definido.

Detalhes da configuração: Veja https://docs.omnitouch.com.au/docs/repos/OmniCore

---

Exemplo 8: Chaves HNET para Desconcelamento de SUCI do UDM

O OmniUDM realiza o desconcelamento de SUCI (TS 33.501 §6.12): ele recupera o SUPI do SUCI oculto que um UE apresenta no registro. Isso requer as chaves **privadas** da Rede Domiciliar que emparelham com as chaves **públicas** da Rede Domiciliar provisionadas nos USIMs. Essas chaves privadas residem em um diretório dedicado `hnet_udm/` dentro de group_vars e são implantadas em `/etc/omniudm/hnet/` no host UDM pelo papel `5gc_nf`.

Estrutura do Arquivo

```
hosts/Customer/
└── group_vars/
    └── hnet_udm/
        ├── curve25519-1.key       # Chave privada X25519, PEM (ID
da Chave Pública da Rede Domiciliar 1)
        ├── curve25519-1.pub       # Valor público bruto de 32
bytes (hex) - para provisionamento de USIM
        ├── secp256r1-2.key        # Chave privada EC
(prime256v1), PEM (ID da Chave Pública HNET 2)
        └── ...
```

Nomeação de Arquivos: `<curve>-<id>.key`, onde `<id>` é o Identificador da Chave Pública da Rede Domiciliar que o UDM combina com o SUCI, e `<curve>` é `curve25519` (Perfil A, X25519) ou `secp256r1` (Perfil B, prime256v1). O nome do arquivo é como a chave é vinculada ao seu ID de chave HNET. Deve corresponder ao ID programado no USIM.

O que acontece:

1. O papel garante que `{{ group_vars_omnicore }}/hnet_udm/` exista no controlador Ansible (criado com `become: false`: as chaves permanecem no controlador e nunca são geradas na caixa remota).
2. Ele escaneia o diretório em busca de arquivos `*.key`.
3. **Se nenhum for encontrado, uma chave padrão `curve25519-1.key` (X25519) é gerada automaticamente** com `openssl genpkey`, para que

o desconcelamento de SUCI funcione imediatamente. Isso é idempotente (`creates:` guard): uma chave existente nunca é sobrescrita.

4. Para cada `curve25519-*.key`, o valor público bruto correspondente de 32 bytes é exportado para um arquivo `.pub` (hex) para provisionamento de USIM. Os arquivos `.pub` permanecem no inventário e **não** são implantados no UDM.
5. Todos os arquivos `*.key` são copiados para `/etc/omniudm/hnet/` no UDM (modo `0600`, propriedade root), e `omniudm` é reiniciado.

> **Importante:** Uma chave gerada automaticamente é aleatória e só desconcelará SUCIs de USIMs programados com sua chave pública **correspondente**. Para um laboratório onde você controla a programação do SIM, pegue o `.pub` exportado e provisioná-lo como a Chave Pública da Rede Domiciliar (ID 1) no USIM. Se seus SIMs usarem o esquema nulo (sem ocultação), nenhuma chave HNET é necessária e este diretório pode permanecer vazio.

Para exportar uma chave pública de uma chave privada existente manualmente:

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

Detalhes da configuração: Veja https://docs.omnitouch.com.au/docs/repos/OmniCore

---

Exemplo de Estrutura de Diretório

do Mundo Real

```
hosts/Customer/
├── host_files/
│   └── production.yml          # O arquivo de hosts faz
referência aos arquivos de group_vars
└── group_vars/
    ├── smsc_controller.exs     # Modelo personalizado do
OmniMessage
    ├── smsc_smpp.exs           # Modelo SMPP personalizado do
OmniMessage
    ├── tas_runtime.exs.j2      # Modelo personalizado do TAS
    ├── hss_runtime.exs.j2      # Modelo personalizado do HSS
    ├── mme_runtime.exs.j2      # Modelo personalizado do MME
    ├── dra_runtime.exs.j2      # Modelo personalizado do DRA
    ├── pgwc_runtime.exs.j2     # Modelo personalizado do PGW
    ├── dea_runtime.exs.j2      # Modelo personalizado do DEA
    ├── upf_config.yaml         # Configuração do UPF
    ├── crm_config.yaml         # Configuração do CRM
    ├── stp.j2                  # Modelo SS7 STP
    ├── hlr.j2                  # Modelo SS7 HLR
    ├── camel.j2                # Modelo SS7 CAMEL
    ├── ipsmgw.j2               # Modelo IP-SM-GW
    ├── omnicore_smsc_ims.yaml.j2  # Configuração SMSC IMS
    ├── pytap.yaml              # Configuração TAP3
    ├── sip_profiles/           # Gateways SIP (pasta)
    │   └── gateway_otw.xml
    ├── dialplan/               # Regras de roteamento de chamadas
(pasta)
    │   ├── mo_dialplan.xml     # Origem Móvel
    │   ├── mt_dialplan.xml     # Terminado Móvel
    │   └── mo_emergency.xml    # Roteamento de emergência
    ├── omnihss/                # Dados de provisionamento do
OmniHSS (pasta)
    │   ├── subscribers.csv         # Assinantes a serem
provisionados
    │   ├── 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/               # Chaves de desconcelamento SUCI
do UDM (pasta)
        ├── curve25519-1.key        # Chave privada X25519 (ID da
Chave Pública HNET 1)
        └── curve25519-1.pub        # Valor público bruto (hex)
para provisionamento de USIM
```

Parâmetros Comuns que

Referenciam group_vars

Parâmetro	Componente	Referências
`smsc_template_config`	omnimessage	Arquivo de modelo Jinja2 (ex: `smsc_controller.exs`)
`smsc_smpp_template_config`	omnimessage_smpp	Arquivo de modelo Jinja2 (ex: `smsc_smpp.exs`)
`gateways_folder`	applicationserver	Nome da pasta (ex: `gateways_prod`)
`dialplan_folder`	applicationserver	Nome da pasta (ex: `dialplan`) - padrão para `dialplan` se não definido
`tas_runtime_template`	applicationserver	Arquivo de modelo Jinja2 (ex: `tas_runtime.exs.j2`) - padrão para `tas_runtime.exs.j2` se não definido
`hss_template_config`	omnihss	Arquivo de modelo Jinja2 (ex: `hss_runtime.exs.j2`)
`mme_template_config`	omnimme	Arquivo de modelo Jinja2 (ex: `mme_runtime.exs.j2`)
`dra_template_config`	dra	Arquivo de modelo Jinja2 (ex:

Parâmetro	Componente	Referências
		`dra_runtime.exs.j2`)
`pgwc_template_config`	pgwc	Arquivo de modelo Jinja2 (ex: `pgwc_runtime.exs.j2`)
`frr_template_config`	omniupf	Arquivo de modelo Jinja2 (ex: `frr.conf.j2`)
Modelos SS7	ss7 (vários papéis)	Arquivos de modelo Jinja2 (ex: `stp.j2`, `hlr.j2`, `camel.j2`)
`crm_config.yaml`	omnicrm	Arquivo de configuração da API - **obrigatório**, obtido apenas de `{{ group_vars_omnicore }}/crm_config.yaml` (sem padrão de papel)
`crm_env_override`	omnicrm	Nome do arquivo .env da UI (ex: `customer_crm.env`) - **obrigatório**, sem padrão de papel
`cell_sites`	omnicrm	Arquivo de dados de sites de células opcionais (ex: `cellSites.txt`)
`message_templates`	omnicrm	Modelos de mensagens de transmissão de células

Parâmetro	Componente	Referências
		opcionais (ex: `constants.js`)
`site_data`	omnicrm	Limites geográficos de transmissão de células opcionais (ex: `site_data.json`)
`grafana_smtp_user`	monitoring	Usuário SMTP do Grafana - **obrigatório**, sem padrão de papel
`grafana_smtp_password`	monitoring	Senha SMTP do Grafana - **obrigatório**, sem padrão de papel
`grafana_smtp_host`	monitoring	Host SMTP opcional do Grafana - padrão para `in-v3.mailjet.com:587`
Provisionamento OmniHSS	omnihss	Diretório `omnihss/` com `subscribers.csv` + arquivos JSON de perfil

Parâmetro	Componente	Referências
Chaves HNET	5gc_nf (omniudm)	Diretório `hnet_udm/` de chaves privadas `<curve>-<id>.key` para desconcelamento de SUCI - gera automaticamente uma `curve25519-1.key` padrão se vazio
Configurações YAML	Vários componentes	Arquivos de configuração diretos (ex: `upf_config.yaml`, `crm_config.yaml`)

Pontos Chave

1. **group_vars contém personalizações** - Substituições para configurações padrão
2. **Referencie pelo nome** - Use parâmetros como `smsc_template_config` ou `gateways_folder`
3. **Modelos suportam Jinja2** - Acesse qualquer variável do Ansible com `{{ variable_name }}`
4. **Pastas implantam tudo** - Todos os arquivos nas pastas referenciadas são copiados
5. **Controle de versão de tudo** - Comite todos os group_vars no Git

Quando Usar group_vars

Use group_vars para:

- Modelos de configuração de componentes personalizados

- Definições de gateways SIP
- Planos de discagem para roteamento de chamadas
- Regras de roteamento Diameter
- Configurações específicas do cliente que substituem os padrões

 Não use group_vars para:

- Configuração básica de host (IPs, nomes de host) - Use o arquivo de hosts
- Testes únicos - Use variáveis específicas do host no arquivo de hosts
- Alterações temporárias - Edite no alvo, comite em group_vars se for permanente

---

Documentação Relacionada

- Referência de Configuração - Todos os parâmetros de host e o que eles fazem
- Configuração do Arquivo de Hosts - Como estruturar arquivos de hosts
- **Configuração do OmniCall**: https://docs.omnitouch.com.au/docs/repos/OmniCall - O que vai nos arquivos de configuração
- **Configuração do OmniCore**: https://docs.omnitouch.com.au/docs/repos/OmniCore - Detalhes de configuração do componente

Configuração do Arquivo Hosts

Visão Geral

O arquivo hosts (também chamado de arquivo de inventário) é o documento de configuração central que define toda a sua implantação de rede celular. Ele especifica:

- Quais funções de rede implantar
- Onde elas são executadas (endereços IP, segmentos de rede)
- Como elas são configuradas (parâmetros específicos de serviço)
- Configurações específicas do cliente (PLMN, credenciais, recursos)

Localização do Arquivo

Os arquivos hosts estão localizados no diretório de nível superior `hosts/` (definido como o `inventory` do Ansible em `ansible.cfg`), organizados por cliente:

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

Os nomes dos arquivos são arbitrários. Há um arquivo por implantação, tipicamente nomeado de acordo com a implantação ou local (por exemplo, `Production.yml`, `Staging.yml`) em vez de um esquema fixo `production.yml`/`staging.yml`/`lab.yml`.

Estrutura de Exemplo do Arquivo Hosts

Este exemplo simplificado mostra as seções principais:

```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

# Serviços de Suporte
license_server:
  hosts:
    customer-licenseserver:
      ansible_host: 10.10.2.150

# Variáveis Globais
all:
  vars:
    ansible_connection: ssh
    ansible_password: password
```

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

Parâmetros Comuns de Host

Configuração de Rede

Cada host normalmente inclui:

```
pcscf:
  hosts:
    customer-pcscf01:
      ansible_host: 10.10.1.15      # Endereço IP para acesso SSH
      gateway: 10.10.1.1             # Gateway padrão
      host_vm_network: "vmbr1"       # nome da NIC a ser usada no
Hypervisor
```

Nota: Para orientações sobre planejamento de endereços IP e estratégias de segmentação de rede, consulte o Padrão de Planejamento de IP que descreve a arquitetura recomendada de quatro sub-redes para implantações do OmniCore.

Usuários do Proxmox: O parâmetro `host_vm_network` especifica qual ponte usar. Consulte Implantação de VM/LXC do Proxmox para provisionamento automatizado.

Alocação de Recursos de VM

Para serviços que precisam de recursos específicos:

```
num_cpus: 4                     # Núcleos de CPU
memory_mb: 8192                 # RAM em megabytes
proxmoxLxcDiskSizeGb: 50      # Tamanho do disco em GB
```

Parâmetros Específicos de Serviço

Cada função de rede tem seus próprios parâmetros. Exemplos:

MME:

```
mme_code: 1                     # Identificador do MME (1-255)
mme_gid: 1                      # ID do Grupo MME
network_name_short: Customer    # Nome da rede (exibido nos
telefones)
network_name_long: Customer Network
tac_list: [600, 601, 602]     # Códigos de Área de Rastreamento
```

PGW:

```
ip_pools:                       # Pools de IP para assinantes
  - '100.64.16.0/24'
  - '100.64.17.0/24'
combined_CP_UP: false           # Plano de controle/plano de
usuário separados
```

Para uma explicação detalhada sobre o que cada variável controla, consulte: Referência de Configuração

Servidor de Aplicação:

```
online_charging_enabled: true  # Habilitar integração OCS
tas_branch: "main"             # Branch de software a ser
implantada
gateways_folder: "gateways_prod"  # Configuração do gateway SIP
```

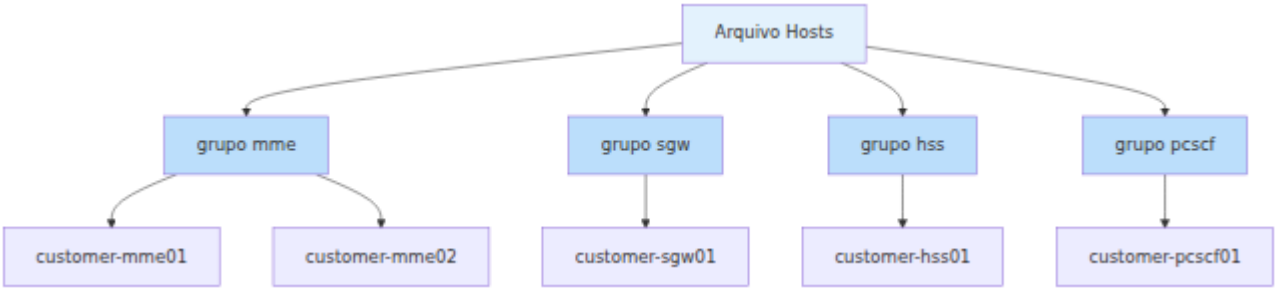
Seção de Variáveis Globais

A seção `all:vars` contém configurações que se aplicam a toda a implantação:

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

    # Identidade do Cliente
    customer_name_short: customer
    customer_legal_name: "Customer Inc."
    site_name: "Chicago DC1"
    region: US

    # Identificador PLMN (Rede Móvel)
    plmn_id:
      mcc: '001'              # Código do País Móvel
      mnc: '01'               # Código da Rede Móvel
      mnc_longform: '001'     # MNC com zeros à esquerda

    # Nomes de Rede
    network_name_short: Customer
    network_name_long: Customer Network

    # Repositório APT
    # Nota: Se o grupo apt_cache_servers estiver definido com
hosts,
    # use_apt_cache padrão para true e apt_repo.apt_server
    # padrão para o IP do primeiro servidor de cache
automaticamente
    apt_repo:
      apt_server: "10.254.10.223"
      apt_repo_username: "customer"
      apt_repo_password: "secure-password"
    use_apt_cache: false

    # Fuso Horário
    TZ: America/Chicago
```

Entendendo Grupos de Hosts

O Ansible organiza hosts em grupos que correspondem a funções:



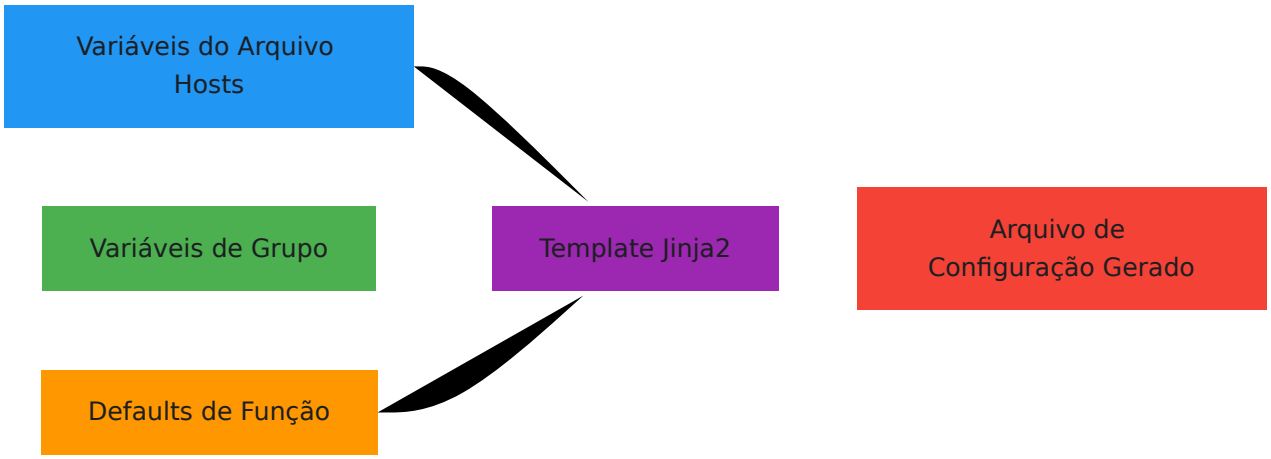
Quando você executa um playbook direcionado a `mme`, ele se aplica a todos os hosts na seção `mme:hosts:`.

O diagrama acima mostra apenas um punhado de grupos. Muitos outros existem, incluindo `ocs`, `crm`, `smsc`, `omnimessage`, `dra`, `xcap`, `siptrace`, `cbc`, e os grupos 5GC `amf`, `smf`, e `upf`, entre outros.

Configuração com Templates Jinja2

O Ansible usa **templating Jinja2** para gerar arquivos de configuração a partir das variáveis definidas no seu arquivo hosts e group_vars.

Como Funciona o Jinja2



Exemplo de Uso de Template

Arquivo hosts define:

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

Template Jinja2 (na função):

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

Arquivo de configuração gerado:

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

Padrões Comuns do Jinja2

Acessando variáveis aninhadas:

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

Lógica condicional:

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

Laços:

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

Formatação:

```
# Preencher com zeros até 3 dígitos
mnc{{ '%03d' | format(plmn_id.mnc|int) }}
```

Sobrescrevendo Variáveis com group_vars

Enquanto o arquivo hosts define a infraestrutura e as configurações específicas do host, `group_vars` pode sobrescrever os padrões para grupos de hosts.

Veja: Configuração de Variáveis de Grupo

Exemplo Completo de Arquivo Hosts

Este exemplo mais completo tem dados sensíveis ocultos:

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

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

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

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

# IMS Core
pcscf:
```

```yaml
  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"

# Serviços de Suporte
license_server:
  hosts:
    customer-licenseserver:
      ansible_host: 10.10.2.150
      gateway: 10.10.2.1
      host_vm_network: "vmbr2"

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

```yaml
      memory_mb: 8192

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

# Variáveis Globais
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

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

    # Nomes de Rede
    network_name_short: Customer
    network_name_long: Customer Network
    tac_list: [600, 601]

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

    # Configuração de Cobrança
    charging:
```

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

    # Regras de Firewall — veja docs/firewall.md para o conjunto
completo de parâmetros
    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']

    # Configuração do Hypervisor (exemplo de VM Proxmox)
    proxmoxServers:
      customer-prxmx01:
        # deployment_type omitido → padrão para "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        # nome de usuário
cloud-init opcional; padrão para a primeira chave local_users
        proxmoxTemplatePassword: omnitouch    # senha cloud-init
opcional; padrão para o nome de usuário cloud-init

    # Para um site LXC, defina deployment_type: lxc e
    # proxmoxLxcOsTemplate em cada entrada de proxmoxServers em
vez disso.
```

Veja Implantação de VM/LXC do Proxmox para detalhes completos de configuração e configuração do Proxmox.

Referências da Documentação do Produto

Para configuração detalhada de cada componente, consulte a documentação oficial do produto:

Componentes OmniCore:

- **Documentação do OmniCore**: https://docs.omnitouch.com.au/docs/repos/OmniCore
- **OmniHSS** - Servidor de Assinante Residencial
- **OmniSGW** - Gateway de Serviço (Plano de Controle)
- **OmniPGW** - Gateway de Pacote (Plano de Controle)
- **OmniUPF** - Função do Plano de Usuário
- **OmniDRA** - Agente de Roteamento Diameter
- **OmniTWAG** - Gateway de Acesso WLAN Confiável

Componentes OmniCall:

- **Documentação do OmniCall**: https://docs.omnitouch.com.au/docs/repos/OmniCall
- **OmniTAS** - Servidor de Aplicação IMS (VoLTE/VoNR)
- **OmniCall CSCF** - Funções de Controle de Sessão de Chamada
- **OmniMessage** - Centro de SMS
- **OmniMessage SMPP** - Suporte ao Protocolo SMPP
- **OmniSS7** - Pilha de Sinalização SS7
- **VisualVoicemail** - Correio de Voz

OmniCharge/OmniCRM:

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

Documentação Relacionada

- Introdução à Implantação do Ansible - Processo geral de implantação
- Referência de Configuração - **Guia completo para todas as variáveis de configuração**
- Configuração de Variáveis de Grupo - Sobrescrevendo configurações padrão
- Padrão de Planejamento de IP - **Arquitetura de rede e diretrizes de alocação de IP**
- Configuração do Netplan - **IPs secundários e configuração avançada de rede**
- Sistema de Cache APT - Distribuição de pacotes
- Servidor de Licença - Gerenciamento de licenças
- Visão Geral da Arquitetura de Implantação - Visão completa do sistema

Próximos Passos

1. Crie seu arquivo hosts com base neste template
2. Defina seu PLMN e identidade de rede
3. Configure o acesso ao repositório APT
4. Configure o servidor de licença
5. Personalize com group_vars conforme necessário
6. Implemente com playbooks do Ansible

Servidor de Licenças

Visão Geral

O Servidor de Licenças gerencia a ativação de recursos para todos os componentes do Omnitouch. Cada componente valida sua licença na inicialização e periodicamente durante a operação.

Configuração

1. Definir no Arquivo de Hosts

```yaml
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: "Nome do Cliente"
    license_server_api_urls: ["https://10.10.2.150:8443/api"]
```

> **Nota:** `license_enforced` é **escopado para HSS**, não é um toggle de aplicação global. Ele é definido apenas para o papel `omnihss` (`roles/omnihss/defaults/main.yml`, onde o padrão é `false`) e atualmente é lido apenas pelo gerador HLD; nenhum outro papel o consome. Defina-o sob as variáveis de grupo omnihss se precisar.

2. Fornecer Arquivo de Licença

Coloque `license.json` (fornecido pelo Omnitouch) em `hosts/Customer/group_vars/`

3. Implantar

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

Os diretórios de inventário reais são capitalizados e prefixados (por exemplo, `hosts/Omnicore_Customer/`), e o arquivo de hosts é nomeado de acordo com o site (por exemplo, `hosts/Omnicore_Customer/host_files/Production.yml`).

Para verificar o status de todas as licenças, acesse https://license_server .

Removendo uma Licença

Para remover uma licença, use `util_playbooks/delete_license.yml`.

Requisitos de Rede

Configuração do Firewall

Os firewalls do site do cliente devem ser configurados para permitir tráfego HTTPS (porta 443) para os servidores de validação de licença do Omnitouch:

Nome do Host	Propósito
1.time.omnitouch.com.au	Servidor de validação de licença 1
2.time.omnitouch.com.au	Servidor de validação de licença 2
3.time.omnitouch.com.au	Servidor de validação de licença 3

Regras de saída necessárias:

- Protocolo: HTTPS (TCP/443)
- Destino: os nomes dos hosts acima, resolvidos via DNS
- Direção: Saída

> **Resolver por DNS: não codifique IPs.** Sempre permita esses hosts por FQDN (ou pelos endereços que seu firewall resolve dinamicamente). Os IPs subjacentes podem mudar sem aviso. Listas de permissão de IPs estáticos não são suportadas e eventualmente quebrarão a validação da licença.

Requisitos de DNS

O servidor de licença requer resolução DNS funcional para se comunicar com a infraestrutura de validação de licença do Omnitouch.

Configuração de DNS necessária:

- O servidor de licença deve ter acesso a servidores DNS públicos
- Configure o DNS para usar um dos seguintes:
  - 1.1.1.1 (Cloudflare - suporta DNS seguro)
  - 8.8.8.8 (Google Public DNS)
- Não use servidores DNS internos/corporativos para o servidor de licença

Nota: Os servidores de licença do Omnitouch usam DNS seguro (DoH/DoT). Usar servidores DNS públicos garante validação adequada de DNSSEC e previne problemas com interceptação de DNS por dispositivos de segurança.

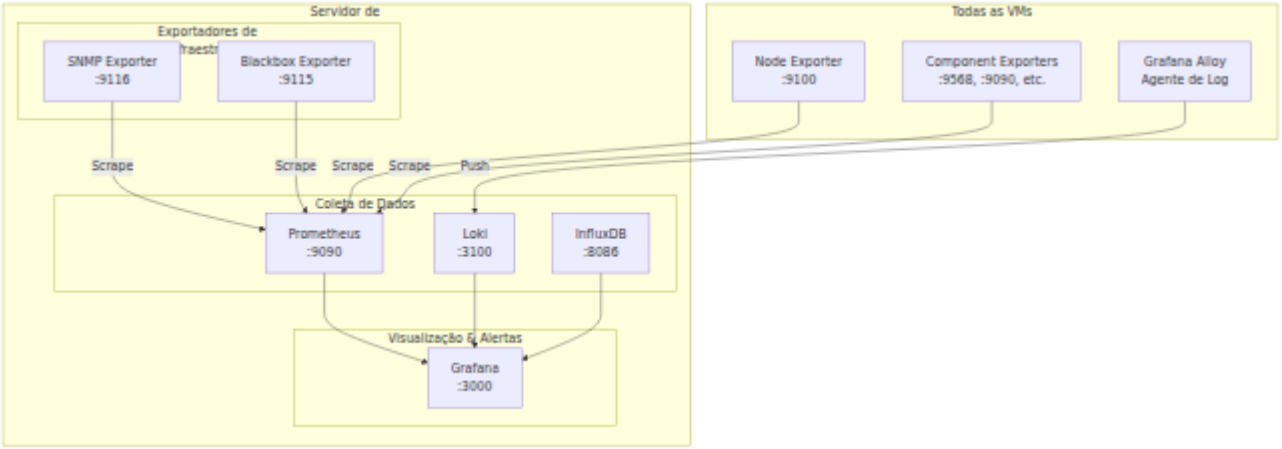
Documentação Relacionada

- Referência de Configuração
- Configuração do Arquivo de Hosts

Monitoramento & Observabilidade

Visão Geral

OmniCore inclui uma pilha abrangente de monitoramento e observabilidade que fornece coleta de métricas, agregação de logs, visualização e alertas. O sistema é implantado automaticamente pelo papel de monitoramento.



Componentes

Componente	Propósito	Porta	Retenção de Dados
Prometheus	Armazenamento de métricas de séries temporais	9090	30 dias ou 5GB
Loki	Agregação de logs	3100	7 dias ou 50GB
InfluxDB	Métricas RAN (Nokia)	8086	30 dias
Grafana	Visualização e alertas	3000	N/A
Node Exporter	Métricas do host	9100	N/A
SNMP Exporter	Métricas de dispositivos de rede	9116	N/A
Blackbox Exporter	Provas ICMP/HTTP	9115	N/A

Fontes de Dados

Prometheus

Prometheus coleta métricas de todos os componentes do OmniCore a cada 1 minuto.

UID da Fonte de Dados: `omnicore_prometheus`

Alvos Coletados

Nome do Trabalho	Grupo de Inventário	Porta	Métricas
`Node Exporter`	`all`	9100	CPU do host, memória, disco, rede
`MMEs`	`mme`	9568	Contagem de sessões, taxas de anexação/desanexação
`HSS`	`hss`	9568	Solicitações de autenticação, pesquisas de assinantes
`SGW-C`	`sgw`	9568	Contagem de portadores, mensagens GTP-C
`PGW-C`	`pgwc`	9090	Conexões PDN, alocações de IP
`UPF Standalone`	`upf`	9090	Contagem de pacotes, throughput
`CSCF`	`pcscf`, `scscf`, `icscf`	9090	Registros, transações
`ApplicationServer FreeSWITCH`	`applicationserver`	9090	Contagem de chamadas, uso de canais
`DRA`	`dra`	9568	Decisões de roteamento, status de pares

Nome do Trabalho	Grupo de Inventário	Porta	Métricas
`OmniMessage`	`omnimessage`	9568	Entrega de SMS, sessões SMPP
`OmniSS7`	`omniss7`	8080	Métricas de gateway SS7/SIGTRAN
`KeyDB`	`ocs`	9121	Uso de memória, operações/segundo
`CGrateS`	`ocs`	2080	Classificação, cobrança, estatísticas CDR

Exportadores de Infraestrutura

Exportador	Alvos	Variável de Configuração
SNMP (MikroTik)	Roteadores/ switches	`mikrotik.hosts`
SNMP (iDRAC)	Servidores Dell	`idrac.hosts`
SNMP (Synology)	Dispositivos NAS	`synology.hosts`
SNMP (SAF)	Links de micro-ondas	`saf.hosts`
SNMP (Cisco)	Switches	`cisco.hosts`
Blackbox ICMP	Todos os hosts + 8.8.8.8	Automático
VMware	Clusters vCenter	`vcenter_ip`, `vcenter_password`
Proxmox	Clusters PVE	`proxmoxServers`

Loki

Loki recebe logs dos agentes Grafana Alloy em execução em todas as VMs.

UID da Fonte de Dados: `omnicore_loki`

Veja Registro Centralizado para configuração detalhada do Loki/Alloy.

Rótulos de Log

Rótulo	Descrição	Exemplo
`hostname`	Nome do host da VM de origem	`customer-mme01`
`component`	Tipo de componente	`mme`, `cscf`, `ocs`
`unit`	Nome da unidade systemd	`omnimme.service`
`level`	Severidade do log	`info`, `error`

InfluxDB

InfluxDB armazena dados de séries temporais para casos de uso específicos que requerem retenção mais longa ou padrões de consulta diferentes.

UID da Fonte de Dados: `omnicore_influxdb`

Bancos de Dados

Banco de Dados	Propósito	Retenção
`nokia-monitor`	Métricas de desempenho RAN da Nokia	30 dias
`dra`	Estatísticas de roteamento DRA	365 dias
`Omnicharge_TAP3`	Métricas de arquivos TAP de roaming	90 dias

Painéis do Grafana

Organização dos Painéis

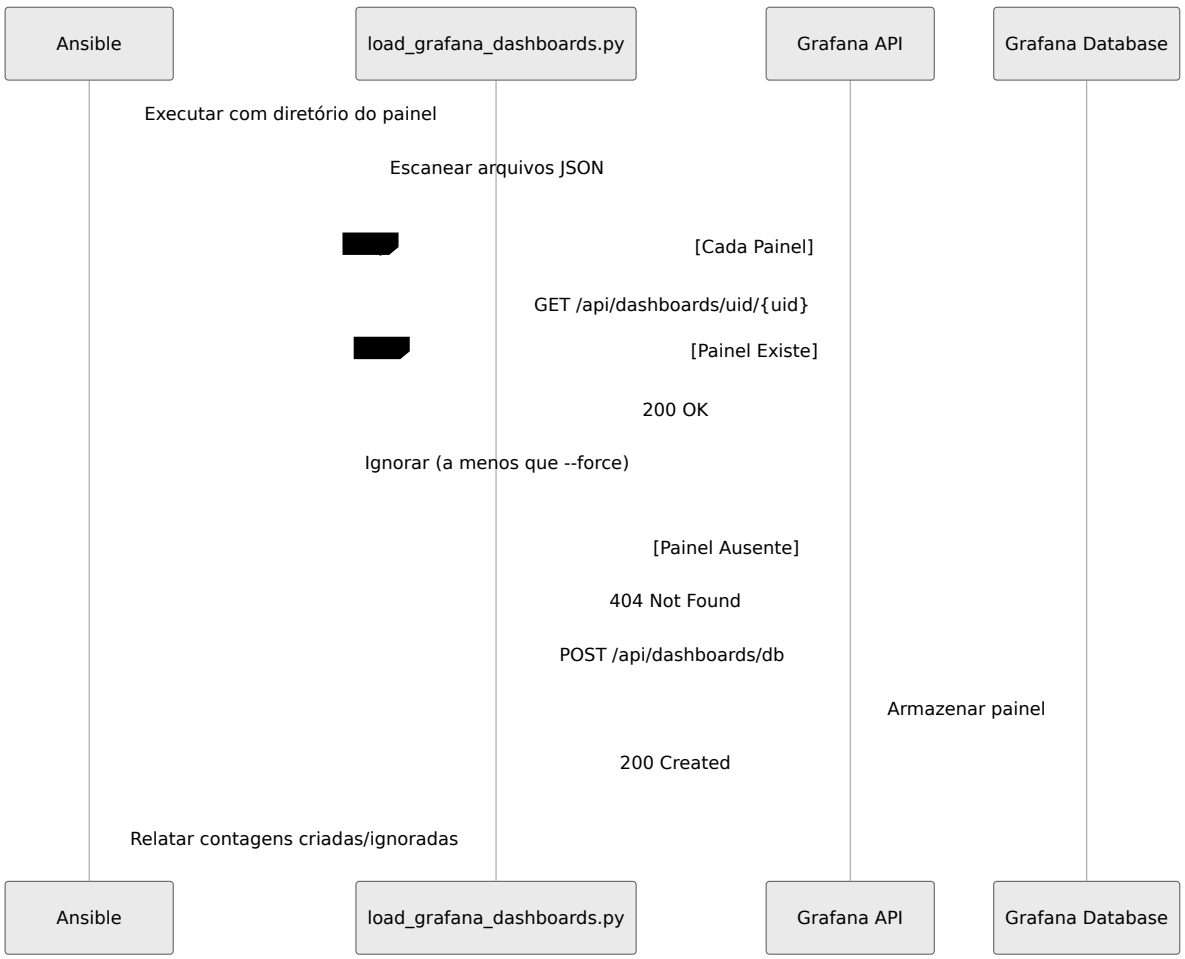
Os painéis são organizados em pastas por área funcional:

Pasta	Conteúdos
BSS	Painéis de CGrateS, KeyDB, cobrança
EPC	Painéis de MME, SGW, PGW, UPF, HSS, DRA
IMS	Painéis de CSCF, Servidor de Aplicação, OmniMessage
Infraestrutura	Node Exporter, SNMP, monitoramento de rede
RAN	Painéis de desempenho Nokia/OmniRAN
Logs	Visualizadores de logs específicos de componentes

Carregamento de Painéis (Baseado em API)

Os painéis são carregados via API do Grafana em vez de provisionamento baseado em arquivos. Isso permite que os painéis sejam:

- Editados através da UI do Grafana
- Modificados via API
- Controlados por versão no repositório Ansible



Carregamento de Painéis

Os painéis são carregados automaticamente durante a implantação do Ansible. Para recarregar manualmente:

```
# Do controlador Ansible
python3 roles/monitoring/files/load_grafana_dashboards.py \
    --url http://<monitoring-ip>:3000 \
    --user admin \
    --password <grafana_admin_password> \
    --dashboards-dir roles/monitoring/templates/grafana/dashboards

# Forçar atualização de painéis existentes
python3 roles/monitoring/files/load_grafana_dashboards.py \
    --url http://<monitoring-ip>:3000 \
    --user admin \
    --password <grafana_admin_password> \
    --dashboards-dir roles/monitoring/templates/grafana/dashboards
\
    --force
```

Arquivos Fonte dos Painéis

Os arquivos JSON dos painéis são armazenados no repositório Ansible:

```
roles/monitoring/templates/grafana/dashboards/
├── BSS/
│   ├── KeyDB_Cluster.json
│   ├── cgrates_mysql.json
│   └── cgrates_stats.json
├── EPC/
│   ├── MME_Dashboard.json
│   ├── OmniHSS.json
│   ├── OmniDRA.json
│   ├── SGW.json
│   ├── PGWs.json
│   └── ...
├── IMS/
│   ├── SMSc.json
│   ├── MMSc.json
│   └── ...
├── Infraestrutura/
│   ├── Node_Exporter_Full.json
│   ├── MikroTik_Dashboard.json
│   └── ...
├── RAN/
│   ├── Nokia_Overview.json
│   ├── Nokia_Detailed.json
│   └── ...
└── Logs/
    ├── CSCF_Logs.json
    ├── MME_Logs.json
    └── ...
```

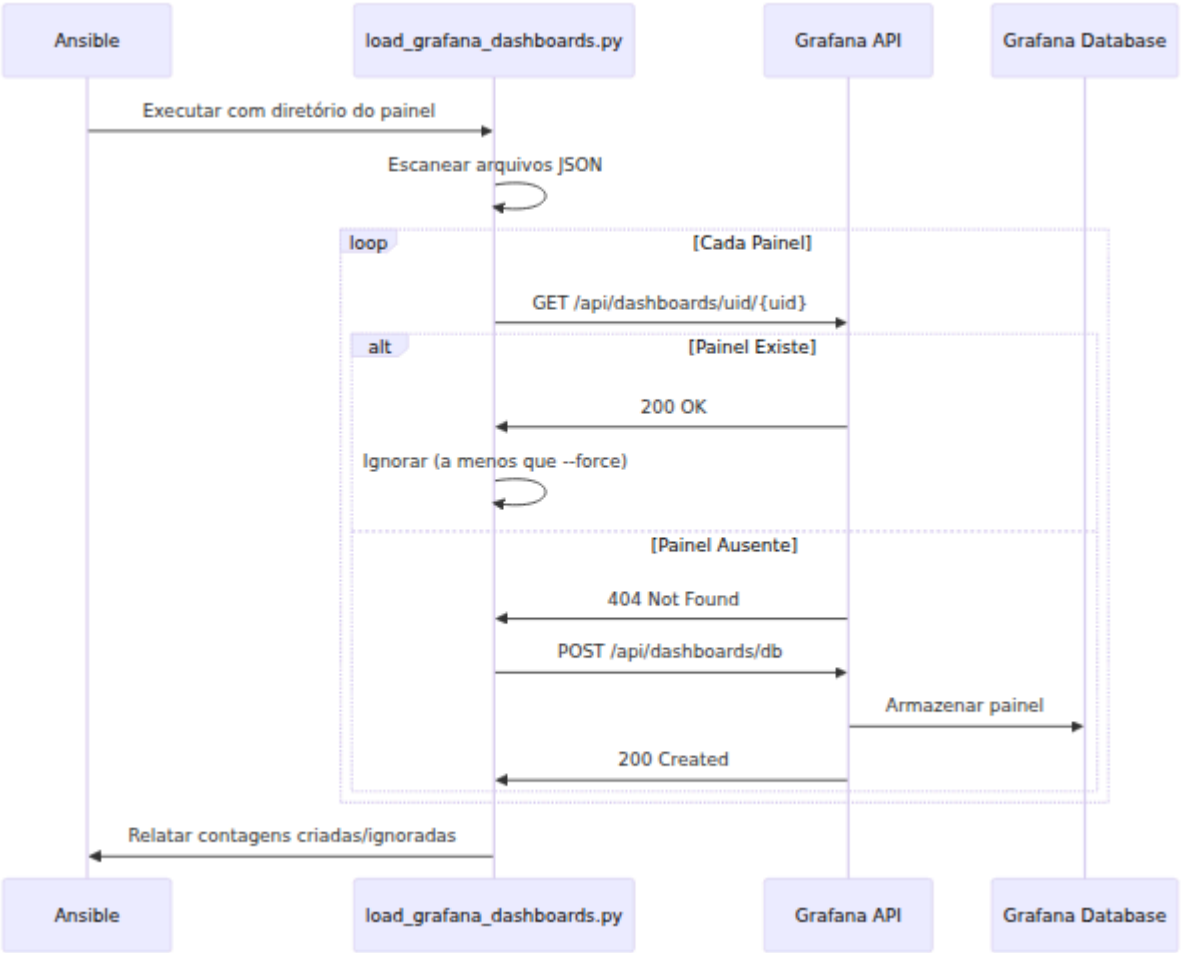
Backup de Painéis

Exporte painéis e alertas de uma instância Grafana em execução para o repositório para controle de versão:

```
# Do diretório roles/monitoring
python3 export_grafana.py --customer <CUSTOMER>
```

Isso exporta:

- Todos os painéis para `roles/monitoring/templates/grafana/dashboards/{folder}/*.json` (compartilhados entre clientes)
- Regras de alerta para `hosts/Omnicore_{CUSTOMER}/group_vars/grafana/alerts/all_alert_rules.json`
- Pontos de contato para `hosts/Omnicore_{CUSTOMER}/group_vars/grafana/alerts/contact_points.json`
- Políticas de notificação para `hosts/Omnicore_{CUSTOMER}/group_vars/grafana/alerts/notification_policies.json`



Comportamento de Exportação

- Apenas escreve arquivos que foram alterados (ignorando campos voláteis como `version` e `id`)

- Preserva a estrutura de pastas correspondente à organização do Grafana
- Mapeia títulos de painéis para nomes de arquivos consistentes

Painéis Personalizados

Painéis específicos do cliente podem ser colocados em `group_vars/grafana/dashboards/` e serão carregados juntamente com os painéis padrão:

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

Alertas

Arquitetura

Alertas do Grafana
Regras de Alerta
alerts.yaml
Mecanismo de Avaliação
intervalo de 1m
Alert triggers
Roteador de
Notificações
Fontes de Dados
Prometheus
Loki
Pontos de Contato
Webhook do Slack
Google Chat
Email

[redacted]

Carregamento de Alertas

Os alertas são carregados via API do Grafana a partir de exportações por cliente, em vez de um único arquivo YAML estático. No momento da implantação, o arquivo de provisionamento de alerta estático (`/etc/grafana/provisioning/alerting/alerts.yaml`) é removido para que os alertas possam ser editados via UI/API.

O caminho principal carrega regras de alerta, pontos de contato e políticas de notificação por cliente de `hosts/Omnicore_<CUSTOMER>/group_vars/grafana/alerts/` usando `load_grafana_from_export.py`:

```
python3 roles/monitoring/files/load_grafana_from_export.py \
    --url http://<monitoring-ip>:3000 \
    --user admin \
    --password <grafana_admin_password> \
    --alerts-dir
hosts/Omnicore_<CUSTOMER>/group_vars/grafana/alerts \
    --dashboards-dir roles/monitoring/templates/grafana/dashboards
\
    --force
```

Se não existirem alertas específicos do cliente, o papel recorre ao carregamento do modelo de papel `alerts.yaml` via `load_grafana_alerts.py`:

```
python3 roles/monitoring/files/load_grafana_alerts.py \
    --url http://<monitoring-ip>:3000 \
    --user admin \
    --password <grafana_admin_password> \
    --alerts-file
roles/monitoring/templates/grafana/provisioning/alerting/alerts.yaml
```

Regras de Alerta Padrão

Alerta	Pasta	Condição	Por Duração
Espaço em Disco 90% Usado	Infraestrutura	Sistema de arquivos raiz > 90%	5 minutos
CPU Acima de 90%	Infraestrutura	Uso de CPU > 90%	5 minutos
Carga do Sistema acima de 90%	Infraestrutura	Média de carga > 90%	5 minutos
Host Offline	Infraestrutura	Alvo do Prometheus inacessível	5 minutos
MME Subs Perdidos 40%	EPC	Contagem de sessões cai 40%	30 segundos
0 Subs no MME	EPC	Nenhum assinante anexado	5 minutos
Subs IMS em queda de 40%	IMS	Registros P-CSCF caem 40%	30 segundos
MOS Médio abaixo de 4	IMS	Qualidade de voz degradada	5 minutos
Contagem de eNodeB Reduzida	RAN	eNBs conectados diminuíram	1 minuto
Atraso de Resposta do Diameter Alto	EPC	Pico de latência P95	5 minutos

Estrutura da Regra de Alerta

Regras de alerta em `alerts.yaml`:

```yaml
apiVersion: 1
groups:
  - orgId: 1
    name: Grupo de Alertas
    folder: Infraestrutura
    interval: 1m
    rules:
      - uid: alert-lowDiskSpace
        title: Espaço em Disco 90% Usado no 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="/"})
          # ... expressões de limite
        noDataState: OK
        execErrState: Error
        for: 5m
        annotations:
          summary: Espaço em disco acima de 90% no host
        labels:
          type: resource
```

Configuração de Pontos de Contato

Os pontos de contato são configurados via variáveis de inventário:

```yaml
# Em group_vars/all.yml ou vars específicas do cliente
monitoring_data:
  contactPoints:
    - orgId: 1
      name: padrão
      receivers:
        # Integração com Slack
        - uid: slack-alerts
          type: slack
          disableResolveMessage: false
          settings:
            url: "https://hooks.slack.com/services/xxx/yyy/zzz"

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

Os tipos de pontos de contato suportados incluem Slack, Google Chat, Email e Pushover (veja `roles/monitoring/templates/grafana/contact-points.yaml.j2`). O Pushover é configurado via um bloco `notifications.pushover` (com `enabled`, `api_token` e `user_key`).

Políticas de Notificação

As políticas de notificação roteiam alertas para os pontos de contato apropriados:

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

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

Referência de Configuração

Configuração do Prometheus

Localizado em `/etc/prometheus/prometheus.yml` no servidor de monitoramento.

Parâmetro	Padrão	Descrição
`scrape_interval`	1m	Com que frequência coletar alvos
`evaluation_interval`	1m	Com que frequência avaliar regras de gravação
`scrape_timeout`	50s	Tempo limite para solicitações de coleta

Configuração do Grafana

Localizado em `/etc/grafana/grafana.ini` no servidor de monitoramento.

Configurações principais configuradas pelo papel de monitoramento:

Configuração	Valor	Descrição
`allow_embedding`	true	Habilitar incorporação em iframes
`provisioning`	`/etc/grafana/provisioning`	Diretório de provisionamento

A senha do administrador não é definida em `grafana.ini` (a linha `admin_password` é deixada comentada). Em vez disso, é aplicada via variável `grafana_admin_password`, que é passada para as tarefas do carregador da API (carregadores de painel/alerta) durante a implantação.

Alertas por Email (SMTP)

Alertas por email requerem credenciais SMTP a serem configuradas em `group_vars` (veja `roles/monitoring/templates/grafana/grafana.ini`):

Variável	Necessária	Padrão	Descrição
`grafana_smtp_user`	Sim	*(nenhum)*	Nome de usuário SMTP. Sem padrão. Se não definido, a implantação falha de forma explícita em vez de enviar um segredo.
`grafana_smtp_password`	Sim	*(nenhum)*	Senha SMTP. Sem padrão. Se não definido, a implantação falha.
`grafana_smtp_host`	Não	`in-v3.mailjet.com:587`	Host:porta do servidor SMTP.
`grafana_smtp_enabled`	Não	`true`	Se o SMTP está habilitado.

O `from_address` e `from_name` são derivados de `customer_legal_name` (por exemplo, `alerts+<customer_legal_name>@omnitouch.com.au` e `Omnitouch Alerts - <customer_legal_name>`).

Configuração do Loki

Veja Registro Centralizado para configurações de retenção e armazenamento do Loki.

Configuração do InfluxDB

Localizado em `/etc/influxdb/influxdb.conf` no servidor de monitoramento.

Banco de Dados	Política de Retenção	Duração
`nokia-monitor`	`autogen`	30 dias
`dra`	`autogen`	365 dias
`Omnicharge_TAP3`	`autogen`	90 dias

Portas de Serviço

Serviço	Porta	Protocolo	Descrição
Grafana	3000	HTTP	UI do Painel e API
Prometheus	9090	HTTP	Armazenamento e consulta de métricas
Loki	3100	HTTP	Ingestão e consulta de logs
InfluxDB	8086	HTTP	API do InfluxDB
Node Exporter	9100	HTTP	Métricas do host
SNMP Exporter	9116	HTTP	Proxy de métricas SNMP
Blackbox Exporter	9115	HTTP	Métricas de prova
Alloy	12345	HTTP	UI e métricas do Alloy

Operações Comuns

Visualizando Métricas

1. Abra o Grafana em `http://<monitoring-ip>:3000`
2. Navegue até **Painéis** e selecione a pasta apropriada
3. Use o seletor de intervalo de tempo para ajustar a janela de visualização

Pesquisando Logs

1. Abra o Grafana em `http://<monitoring-ip>:3000`
2. Vá para **Explorar** e selecione a fonte de dados **Loki**

3. Use consultas LogQL:

```
# Todos os logs de um host específico
{hostname="customer-mme01"}

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

# Pesquisar por IMSI específico
{component="hss"} |= "123456789012345"
```

Criando Alertas Personalizados

O fluxo de trabalho recomendado é criar/editar alertas na UI do Grafana e, em seguida, exportá-los para o diretório `group_vars/grafana/alerts/` do cliente (veja Backup de Painéis) para que sejam controlados por versão e reaplicados na próxima implantação.

Para carregar alertas específicos do cliente manualmente:

```
python3 roles/monitoring/files/load_grafana_from_export.py \
    --url http://<monitoring-ip>:3000 \
    --user admin --password <password> \
    --alerts-dir
hosts/Omnicore_<CUSTOMER>/group_vars/grafana/alerts \
    --dashboards-dir roles/monitoring/templates/grafana/dashboards
\
    --force
```

O modelo de papel `roles/monitoring/templates/grafana/provisioning/alerting/alerts.yaml` é usado apenas como um fallback quando um cliente não tem alertas exportados.

Fazendo Backup de Painéis

Após fazer alterações na UI do Grafana, exporte para o repositório:

```
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 "Atualizar alertas da produção"
```

O argumento `--customer` é necessário. Este script exporta regras de alerta, pontos de contato e políticas de notificação para o diretório `group_vars/grafana/alerts/` do cliente; não exporta mais painéis, que são compartilhados entre clientes no diretório de papel (`roles/monitoring/templates/grafana/dashboards/`).

Resolução de Problemas

Alvo do Prometheus Offline

Sintomas: O painel mostra "Sem Dados" ou o status do alvo mostra DOWN

Possíveis causas:

- Serviço não está em execução no host alvo
- Firewall bloqueando a porta do exportador
- Resolução de nome de host incorreta

Resolução:

1. Verifique se o serviço está em execução no alvo:

   ```
   systemctl status <service>
   curl http://localhost:<port>/metrics
   ```

2. Verifique a conectividade a partir do servidor de monitoramento:

   ```
   curl http://<target>:<port>/metrics
   ```

3. Verifique a página de alvos do Prometheus: `http://<monitoring-ip>:9090/targets`

Painel Não Carregando

Sintomas: O painel mostra um spinner de carregamento ou erro

Possíveis causas:

- Conexão da fonte de dados falhou
- Tempo limite de consulta
- Corrupção do JSON do painel

Resolução:

1. Teste a fonte de dados no Grafana: **Configuração** → **Fontes de Dados** → **Testar**
2. Verifique os logs do Grafana: `journalctl -u grafana-server -f`
3. Tente recarregar o painel do repositório

Alertas Não Disparando

Sintomas: A condição é atendida, mas nenhuma notificação recebida

Possíveis causas:

- Alerta está pausado
- Ponto de contato mal configurado
- Política de notificação não correspondendo

Resolução:

1. Verifique o estado do alerta no Grafana: **Alertas** → **Regras de Alerta**
2. Verifique se o teste do ponto de contato funciona: **Alertas** → **Pontos de Contato** → **Testar**
3. Revise o roteamento da política de notificação

Grafana Não Consegue Conectar à Fonte de Dados

Sintomas: "Bad Gateway" ou tempo limite de conexão no Grafana

Possíveis causas:

- Serviço Prometheus/Loki/InfluxDB offline
- Firewall bloqueando conexões localhost
- Serviço vinculado à interface errada

Resolução:

1. Verifique o status do serviço:

   ```
   systemctl status prometheus loki influxdb
   ```

2. Verifique se o serviço está ouvindo:

   ```
   ss -tlnp | grep -E '9090|3100|8086'
   ```

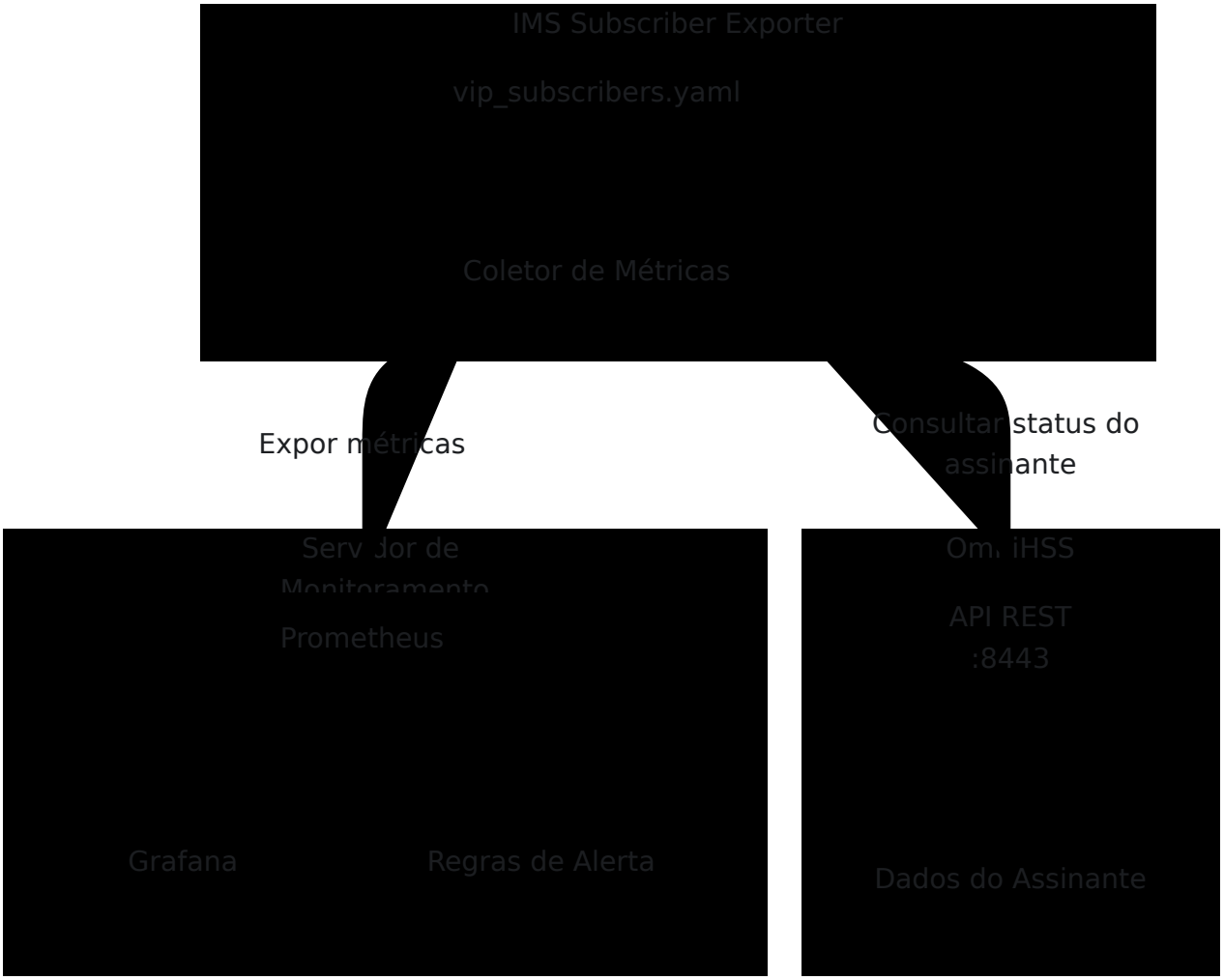
3. Teste a conectividade local:

   ```
   curl http://localhost:9090/api/v1/status/config
   curl http://localhost:3100/ready
   ```

Monitoramento de Assinantes VIP

O Monitoramento de Assinantes VIP fornece rastreamento de status em tempo real para assinantes críticos. O sistema monitora o registro IMS, a anexação EPC e a alcançabilidade do UE para assinantes de alta prioridade designados, como serviços de emergência, hospitais e infraestrutura crítica.

Arquitetura



Tipos de Serviço

Os assinantes são categorizados por tipo de serviço esperado, o que determina a avaliação de saúde:

Tipo	IMS Necessário	EPC Necessário	Caso de Uso
`full`	Sim	Sim	Assinantes móveis padrão com voz e dados
`voip_only`	Sim	Não	Dispositivos apenas VoIP (telefones IP, softphones)
`data_only`	Não	Sim	Dispositivos apenas de dados (roteadores, IoT, M2M)

Avaliação de saúde:

- **full**: Saudável quando ambos IMS registrados (`assigned_scscf` não nulo) E EPC anexado (`last_seen_mme` não nulo)
- **voip_only**: Saudável quando IMS registrado (`assigned_scscf` não nulo)
- **data_only**: Saudável quando EPC anexado (`last_seen_mme` não nulo)

Configuração

Os assinantes VIP são configurados nas variáveis de grupo do host:

Arquivo: `hosts/<customer>/group_vars/vip_subscribers.yaml`

```yaml
vip_subscriber_monitoring:
  hss_url: "https://10.80.12.140:8443"

  # Configurações de monitoramento
  scrape_interval_seconds: 30
  ping_timeout_seconds: 2
  ping_count: 2

  # Assinantes a serem monitorados
  subscribers:
    # Assinantes de serviço completo (IMS + EPC)
    - imsi: "001010000000001"
      label: "Serviços de Emergência"
      type: "full"

    - imsi: "001010000000002"
      label: "Infraestrutura Crítica"
      type: "full"

    # Serviços apenas de dados (roteadores, IoT - sem IMS
esperado)
    - imsi: "001010000000003"
      label: "Roteador do Site"
      type: "data_only"

    # Serviços apenas VoIP (IMS esperado, sem portador de dados)
    - imsi: "001010000000004"
      label: "Telefones IP"
      type: "voip_only"
```

Parâmetros de Configuração

Parâmetro	Tipo	Necessário	Padrão	Descr
`hss_url`	String	Sim	-	URL da API R OmniHSS (H
`scrape_interval_seconds`	Inteiro	Não	30	Com que fre consultar o s assinante
`ping_timeout_seconds`	Inteiro	Não	2	Tempo limite testes de pin UE
`ping_count`	Inteiro	Não	2	Número de p ping a serem
`blackbox_exporter_url`	String	Não	-	URL do expo blackbox ren testes de pin exemplo, `http://pcsc` Usa a prova remota em v local.
`subscribers`	Lista	Sim	-	Lista de assi serem monit

Parâmetros do Assinante

Parâmetro	Tipo	Necessário	Padrão	Descrição
`imsi`	String	Sim	-	IMSI do assinante (15 dígitos)
`label`	String	Não	IMSI	Nome legível por humanos para exibição
`type`	String	Não	`full`	Tipo de serviço: `full`, `voip_only` ou `data_only`

Nota: O MSISDN é recuperado automaticamente da resposta da API do HSS e não precisa ser configurado.

Métricas

O exportador expõe métricas na porta 9550 em `/metrics`.

Métricas de Status

Métrica	Tipo	Descrição
`vip_subscriber_service_healthy`	Gauge	1 se o assinante atende aos critérios de saúde para seu tipo de serviço, 0 caso contrário
`vip_subscriber_ims_registered`	Gauge	1 se o assinante tem S-CSCF atribuído, 0 caso contrário
`vip_subscriber_epc_registered`	Gauge	1 se o assinante tem anexação MME ativa, 0 caso contrário
`vip_subscriber_ue_ip_reachable`	Gauge	1 se o IP do UE responde ao ping, 0 caso contrário
`vip_subscriber_hss_reachable`	Gauge	1 se a API do HSS retornou dados válidos, 0 caso contrário
`vip_subscriber_enabled`	Gauge	1 se a conta do assinante está habilitada no HSS, 0 caso contrário

Métricas de Idade

Métrica	Tipo	Descrição
`vip_subscriber_ims_registration_age_seconds`	Gauge	Segundos desde o último registro IMS (-1 se não registrado)
`vip_subscriber_epc_registration_age_seconds`	Gauge	Segundos desde a última atualização de localização EPC (-1 se não anexado)

Métrica de Informação

Métrica	Tipo	Descrição
`vip_subscriber_info`	Gauge	Sempre 1, carrega todo o status e informações como rótulos

Rótulos em todas as métricas:

Rótulo	Descrição	Exemplo
`imsi`	IMSI do assinante	`001010000000002`
`label`	Nome legível por humanos	`Infraestrutura Crítica`
`msisdn`	Número de telefone (do HSS)	`61400000002`
`type`	Tipo de serviço	`full`, `voip_only`, `data_only`

Rótulos adicionais em `vip_subscriber_info`:

Rótulo	Descrição	Exemplo
`healthy`	Status de saúde do serviço	`1` ou `0`
`ims`	Status de registro IMS	`1` ou `0`
`epc`	Status de anexação EPC	`1` ou `0`
`ping`	Alcançabilidade do IP do UE	`1` ou `0`
`assigned_scscf`	Nome do host S-CSCF	`scscf01.ims.example.com`
`last_seen_mme`	Nome do host MME	`mme02.epc.example.com`
`ue_ip`	Endereço IP atribuído ao UE	`100.72.83.20`

Consultas de Exemplo

```
# Contagem de assinantes VIP saudáveis
count(vip_subscriber_service_healthy == 1)

# Contagem de assinantes VIP não saudáveis
count(vip_subscriber_service_healthy == 0)

# Status de registro IMS para serviços VoIP
vip_subscriber_ims_registered{type=~"voip_only|full"}

# Status de anexação EPC para serviços de dados
vip_subscriber_epc_registered{type=~"data_only|full"}

# Assinantes com registro IMS obsoleto (>1 hora)
vip_subscriber_ims_registration_age_seconds > 3600
```

Painel

O painel **Assinantes VIP IMS** fornece visibilidade em tempo real sobre o status dos assinantes VIP.

Localização: Grafana → IMS → Assinantes VIP IMS

URL do Painel: `/d/ims-vip-subscribers/`

Painéis

Painel	Descrição
Serviços Saudáveis	Contagem de assinantes atendendo aos critérios de saúde
Serviços Não Saudáveis	Contagem de assinantes falhando nos critérios de saúde
IMS Registrados	Contagem de assinantes com registro IMS ativo
EPC Registrados	Contagem de assinantes com anexação EPC ativa
UE Acessível	Contagem de assinantes com IP do UE pingável
Total Monitorado	Número total de assinantes VIP configurados
Status do Assinante VIP	Tabela mostrando todos os assinantes com colunas de status
Saúde do Serviço ao Longo do Tempo	Linha do tempo de estado mostrando o histórico de saúde

Alertas

As regras de alerta disparam quando os serviços de assinantes VIP se tornam não saudáveis.

Serviço de Assinante VIP Não Saudável

Alerta: `Serviço de Assinante VIP Não Saudável` **Severidade**: Crítica **Por**: 1 minuto **Condição**: `vip_subscriber_service_healthy == 0`

Anotações:

- **Resumo**: `Assinante VIP {{ $labels.label }} está não saudável`
- **Descrição**: Inclui tipo de serviço e identificador apropriado:
  - Serviços VoIP: Nome e MSISDN
  - Serviços de dados: Nome e IMSI
  - Serviços completos: Nome, MSISDN e IMSI

Exemplos de descrições de alerta:

- `Serviço de Telefone IP (voip_only) está DOWN. MSISDN: 61400000001`
- `Serviço de Roteador do Site (data_only) está DOWN. IMSI: 001010000000003`
- `Serviço de Infraestrutura Crítica (full) está DOWN. MSISDN: 61400000002 IMSI: 001010000000002`

Adicionando Novos Assinantes VIP

1. Edite o arquivo de configuração:

```yaml
# hosts/<customer>/group_vars/vip_subscribers.yaml
subscribers:
  - imsi: "123456789012345"
    label: "Novo Assinante VIP"
    type: "full"  # ou voip_only, data_only
```

2. Implemente a configuração atualizada:

```
scp vip_subscribers.yaml <monitoring-server>:/tmp/
ssh <monitoring-server> "sudo cp /tmp/vip_subscribers.yaml
/etc/prometheus/vip_subscribers.yaml && sudo systemctl restart
ims-subscriber-exporter"
```

3. Verifique se o novo assinante aparece nas métricas:

```
curl -s http://<monitoring-server>:9550/metrics | grep "Novo
Assinante VIP"
```

Documentação Relacionada

- Registro Centralizado: Configuração detalhada do Loki e Alloy
- Arquitetura de Implantação: Arquitetura geral do sistema
- Configuração do Arquivo de Hosts: Configuração de inventário

Configuração do Netplan

Visão Geral

OmniCore pode configurar automaticamente interfaces de rede em VMs implantadas usando netplan. Isso é útil para:

- Configurar a interface de gerenciamento principal (eth0)
- Adicionar interfaces secundárias para IPs públicos, conexões de peering ou tráfego dedicado
- Configurar rotas estáticas para destinos específicos

Habilitando a Configuração do Netplan

Para habilitar a configuração automática do netplan para um host, adicione a variável `netplan_config` apontando para um template Jinja2 na sua pasta `group_vars`:

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

O template será obtido de `hosts/<customer>/group_vars/netplan.yaml.j2`.

Referência do Template

Não existe um `netplan.yaml.j2` compartilhado e canônico. Cada cliente cria seu próprio `netplan.yaml.j2` em `hosts/<customer>/group_vars/`, e esses templates divergem - o nome da interface principal, o domínio de pesquisa DNS e a nomenclatura da interface secundária variam por cliente. Ao integrar um novo cliente, crie um template por cliente que corresponda ao layout de interface desse cliente em vez de copiar o de outro cliente literalmente.

O seguinte é o template de um cliente como um **exemplo**. Ele é ilustrativo, não um especificação - outros clientes diferem. Comentários são adicionados aqui para explicação:

```
network:
  version: 2
  ethernets:
    # Interface principal - usa ansible_host e gateway do
inventário
    eth0:
      addresses:
      - "{{ ansible_host }}/{{ mask_cidr | default(24) }}"
      nameservers:
        addresses:
{% if 'dns' in group_names %}
        # Se este host É um servidor DNS, use DNS externo para
evitar dependência circular
        - 8.8.8.8
{% else %}
        # Caso contrário, use servidores DNS do grupo 'dns' no
inventário
{% for dns_host in groups['dns'] | default([]) %}
        - {{ hostvars[dns_host]['ansible_host'] }}
{% endfor %}
{% endif %}
        # Domínio de pesquisa DNS - específico do cliente (varia
por cliente; alguns não definem)
        search:
        - customer.example
      routes:
      - to: "default"
        via: "{{ gateway }}"

{% if secondary_ips is defined %}
    # Interfaces secundárias - percorre o dicionário secondary_ips
do inventário
    # Este exemplo as nomeia ens19, ens20, ens21... (18 +
loop.index). Isso é
    # específico do template; veja "Nomenclatura de Interface"
abaixo.
{% for nic_name, nic_config in secondary_ips.items() %}
    ens{{ 18 + loop.index }}:
      addresses:
      # A máscara vem da mask_cidr por NIC (padrão é 24)
      - "{{ nic_config.ip_address }}/{{ nic_config.mask_cidr |
default(24) }}"
{% if nic_config.routes is defined %}
```

```
      # Rotas estáticas para esta interface - cada rota usa o
gateway desta interface
      routes:
{% for route in nic_config.routes %}
      - to: "{{ route }}"
        via: "{{ nic_config.gateway }}"
{% endfor %}
{% endif %}
{% endfor %}
{% endif %}
```

Pontos-chave:

- `ansible_host` e `gateway` vêm da entrada do inventário do host
- Servidores DNS são puxados dinamicamente de hosts no grupo `dns`
- O domínio de pesquisa DNS é específico do cliente (varia; alguns clientes o omitem)
- A nomenclatura da interface secundária varia por cliente (alguns usam `ens19`, `ens20`, ...; veja abaixo)
- Cada IP secundário pode ter sua própria máscara (`mask_cidr`), gateway e rotas estáticas

Configuração da Interface Principal

A interface principal (eth0) é configurada automaticamente usando:

- `ansible_host` - O endereço IP
- `gateway` - O gateway padrão
- `mask_cidr` - Máscara de rede (padrão é 24)

Os servidores DNS são definidos automaticamente como:

- Hosts no grupo `dns` (usa seus IPs `ansible_host`)
- Reverte para `8.8.8.8` se o host for ele mesmo um servidor DNS

MTU do Link (`link_mtu`)

O papel `common` pode fixar o **link MTU** da interface principal independentemente do template do netplan, através da variável de inventário `link_mtu`. Quando definido, ele gera um drop-in do netplan (`/etc/netplan/999-link-mtu.yaml`) na interface principal do host e o aplica. Isso existe porque VMs vêm do template VMware/Proxmox com qualquer MTU que o grupo de porta de origem enviou (às vezes jumbo/9000), o que não combina com o resto da malha e causa buracos negros para SCTP/SIP oversized quando o PMTUD está quebrado.

O drop-in é deliberadamente nomeado `999-` para que ele seja ordenado **por último**: o netplan mescla todos `/etc/netplan/*.yaml` em ordem lexicográfica com o último arquivo ganhando por chave, e o provisionamento pode deixar arquivos concorrentes que definem `mtu` na mesma interface (um legado `80-ansible-config.yaml` e o `99-netcfg-vmware.yaml` do mecanismo VMware). Um drop-in com número mais baixo seria silenciosamente sobrescrito por esses.

```yaml
all:
  vars:
    link_mtu: 1500   # fixa o NIC primário de cada host em 1500
```

Defina-o em `all.vars` para aplicar em todo o site, ou por host para sobrescrever. Se `link_mtu` não for definido, o papel deixa o MTU da interface inalterado.

> **`link_mtu` NÃO é `mtu`.** A variável `mtu` de nível superior é o MTU do payload GTP/túnel (por exemplo, 1400/1430/1300) consumido pelos papéis de núcleo de pacotes, **não** um link MTU. Definir o NIC físico para esses valores seria errado. Sempre use o dedicado `link_mtu` para o MTU da interface.

Interfaces Secundárias

Para hosts que requerem interfaces de rede adicionais (IPs públicos, peering, etc.), use a configuração `secondary_ips`.

Esquema

```
secondary_ips:
  <logical_name>:
    ip_address: <ip_address>
    mask_cidr: <prefix_length>   # Opcional - máscara por NIC,
padrão é 24
    gateway: <gateway_ip>
    host_vm_network: <proxmox_bridge>
    vlanid: <vlan_id>
    nic_name: <interface_name>   # Opcional - substituição
explícita do nome da interface
                                 # (suportado por templates que o
respeitam)
    routes:                      # Opcional - rotas estáticas via
esta interface
      - '<destination_cidr>'
      - '<destination_cidr>'
    table_routes:                # Opcional - avançado, específico
do template
      - { to: '<cidr>', via: '<gateway>', table: <table_id> }
    routing_policy:              # Opcional - avançado, específico
do template
      from: '<source_cidr>'
      via: '<gateway>'
      table: <table_id>
      priority: <priority>       # Opcional, padrão é 100
```

> A chave `mask_cidr` define o comprimento do prefixo para aquela interface secundária. Ela é lida de dentro da entrada do IP secundário (`nic_config.mask_cidr`), não de um `mask_cidr` de nível superior. Se omitida, o padrão é `/24`.

> `table_routes` e `routing_policy` habilitam roteamento por política/fonte, mas são apenas renderizados por templates que os implementam.

> Roteamento avançado é específico do template. Confirme se o `netplan.yaml.j2` do cliente respeita essas chaves antes de usá-las.

Nomenclatura de Interface

A nomenclatura da interface secundária é específica do template - não é padronizada entre os clientes:

- Alguns clientes nomeiam interfaces por índice: `ens19`, `ens20`, `ens21`, ... (`18 + loop.index`). Isso corresponde aos nomes das interfaces atribuídas pelo Proxmox ao adicionar NICs adicionais a uma VM.
- Outros clientes usam um `nic_config.nic_name` explícito quando definido, revertendo para um nome baseado em índice (por exemplo, `ens(34 + loop.index)`) caso contrário.

Se um template respeitar `nic_name`, você pode fixar uma interface secundária a um nome específico definindo `nic_name` nessa entrada de IP secundário; caso contrário, o esquema baseado em índice do template se aplica.

Exemplo de Configuração

```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'
```

Saída do Netplan Gerada

Renderizada com o template de exemplo, a configuração acima gera (o domínio de pesquisa DNS e a nomenclatura `ens19`/`ens20` são específicos do cliente):

```yaml
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"
```

A interface `public_ip` é renderizada como `/28` porque sua entrada definiu `mask_cidr: 28`, enquanto `peering_ip` reverte para o padrão `/24`.

Integração com Proxmox

Ao usar o playbook `proxmox.yml`, NICs secundárias são criadas automaticamente na VM:

1. **Novas VMs**: NICs secundárias são adicionadas durante o provisionamento inicial

2. **VMs Existentes**: NICs secundárias são adicionadas e a VM é reiniciada para aplicar as alterações

A configuração do Proxmox usa:

- `host_vm_network` - A ponte para anexar a NIC
- `vlanid` - Tag VLAN para a interface

Como Funciona

1. As configurações existentes do netplan em `/etc/netplan` são feitas backup (`*.bak`) e depois removidas para evitar conflitos
2. O template Jinja2 do cliente é renderizado para `/etc/netplan/01-netcfg.yaml` (modo `0600`)
3. `netplan try --timeout 30` aplica a configuração com rollback automático se a conectividade for perdida
4. `wait_for_connection` confirma que o host permaneceu acessível
5. `netplan apply` confirma a configuração

Casos de Uso Comuns

Diameter Edge Agent (DEA) com IP Público

```yaml
<hostname>:
  ansible_host: 10.0.1.100          # IP de gerenciamento interno
  gateway: 10.0.1.1
  netplan_config: netplan.yaml.j2
  secondary_ips:
    diameter_roaming:
      ip_address: 192.0.2.50        # IP público para parceiros
de roaming
      gateway: 192.0.2.1
      host_vm_network: "vmbr0"
      vlanid: "200"
      routes:
        - '198.51.100.0/24'         # Rede do parceiro de roaming
```

PGW com Interface S5/S8

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

Servidor Multi-homed com Redes de Gerenciamento e Dados Separadas

```yaml
<hostname>:
  ansible_host: 10.0.1.100           # Rede de gerenciamento
  gateway: 10.0.1.1
  netplan_config: netplan.yaml.j2
  secondary_ips:
    data_network:
      ip_address: 10.0.2.100         # Rede de dados
      gateway: 10.0.2.1
      host_vm_network: "ovsbr2"
      vlanid: "200"
    backup_network:
      ip_address: 10.0.3.100         # Rede de backup
      gateway: 10.0.3.1
      host_vm_network: "ovsbr3"
      vlanid: "300"
```

Referenciando IPs Secundários em Templates

Você pode referenciar endereços IP secundários em outros templates Jinja2 e arquivos de configuração.

No Mesmo Host

Ao configurar um serviço no mesmo host que possui IPs secundários, você pode referenciar diretamente ou usar `inventory_hostname`:

```
# Referência direta (mais simples)
{{ secondary_ips.diameter_public_ip.ip_address }}

# Ou explicitamente via inventory_hostname (mesmo resultado)
{{ hostvars[inventory_hostname]['secondary_ips']
['diameter_public_ip']['ip_address'] }}

# Acesse outras propriedades
{{ secondary_ips.diameter_public_ip.gateway }}
{{ secondary_ips.diameter_public_ip.vlanid }}
```

De Outro Host

Quando você precisa referenciar um IP secundário de um *host diferente* (por exemplo, configurando uma conexão de par), use `hostvars` com o nome do host de destino:

```
# Referência ao primeiro host no grupo dra
{{ hostvars[groups['dra'][0]]['secondary_ips']
['diameter_public_ip']['ip_address'] }}

# Percorra todos os hosts DRA e obtenha seus IPs públicos
{% for host in groups['dra'] %}
{% if hostvars[host]['secondary_ips'] is defined %}
  - {{ hostvars[host]['secondary_ips']['diameter_public_ip']
['ip_address'] }}
{% endif %}
{% endfor %}
```

Exemplo: Configuração de Par DRA

Configure um par de diâmetro para vincular ao seu próprio IP público:

```
# Em dra_config.yaml.j2 - use inventory_hostname para o host atual
peers:
  - name: external_peer
    # Vincule ao IP público de diâmetro deste host
    local_ip: {{ hostvars[inventory_hostname]['secondary_ips']
['diameter_public_ip']['ip_address'] }}
    remote_ip: 198.51.100.50
    port: 3868
```

Verificando se os IPs Secundários Existem

Sempre verifique se a variável existe antes de usá-la:

```
{% 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 %}
```

Solução de Problemas

Verifique os Nomes das Interfaces

SSH para a VM e verifique os nomes das interfaces:

```
ip link show
```

Saída esperada para uma VM com duas interfaces secundárias:

```
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> ...
```

Verifique a Configuração do Netplan

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

Aplique o Netplan Manualmente

```
netplan apply
```

Depure o Netplan

```
netplan --debug apply
```

Verifique as Rotas

```
ip route show
```

Documentação Relacionada

- Configuração do Arquivo de Hosts - Configuração do inventário de hosts
- Implantação de VM/LXC no Proxmox - Provisionamento de VM
- Referência de Configuração - Todas as variáveis de configuração

Syslog de Dispositivos de Rede

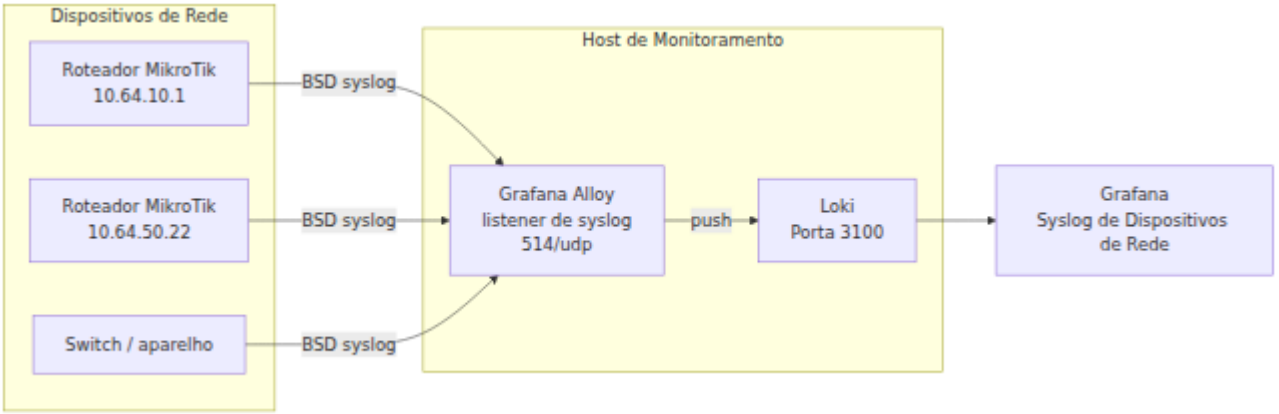
Roteadores, switches e outros aparelhos de rede não podem executar o agente Grafana Alloy. Esses dispositivos enviam syslog em vez disso. Cada host de monitoramento executa um listener de syslog dentro do Alloy. O listener grava as mensagens no Loki local. Os logs dos dispositivos ficam ao lado dos logs de rede principais e compartilham a mesma pesquisa, intervalo de tempo e retenção.

Índice

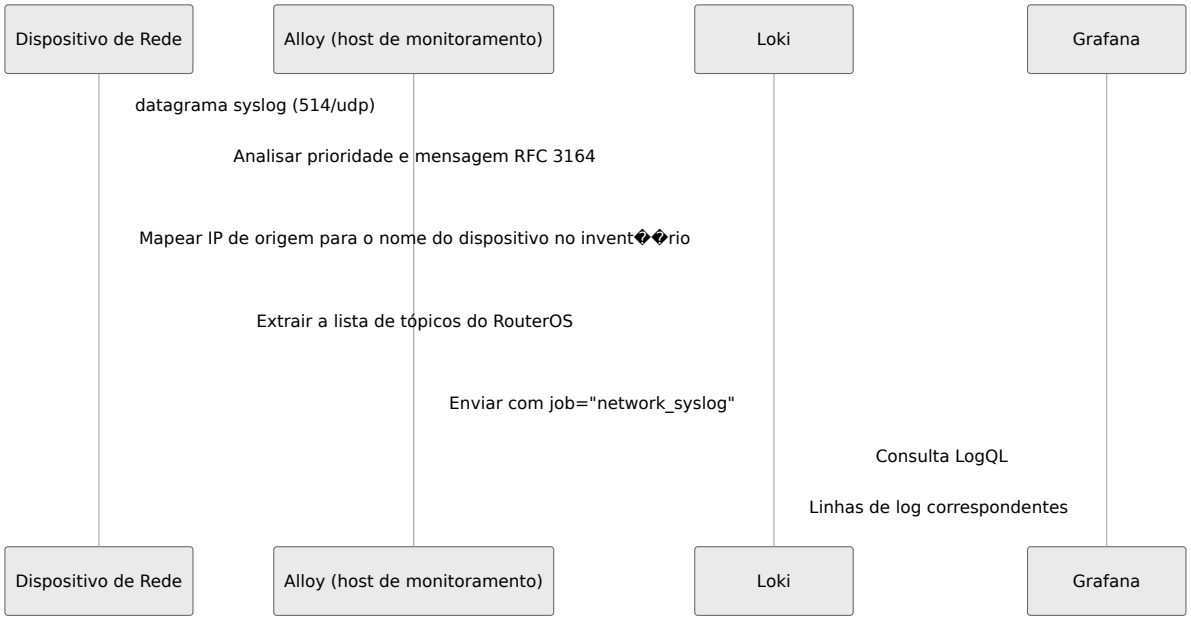
- Arquitetura
- Como os Dispositivos São Identificados
- Configuração do Coletor
- Inventário de Dispositivos
- Rótulos de Log
- MikroTik RouterOS
- Outros Enviadores de Syslog
- Firewall
- Dashboard
- Consultando os Logs
- Solução de Problemas
- Referências

Arquitetura

O coletor é parte da construção padrão de monitoramento. Cada host de monitoramento abre o listener. Um site não define uma flag para receber syslog de dispositivos.



O listener aceita BSD syslog, que é o formato definido pela RFC 3164. O Alloy analisa a prioridade, o timestamp e a mensagem. O Alloy então adiciona os rótulos que as próximas seções descrevem.



Como os Dispositivos São Identificados

O Alloy nomeia cada dispositivo a partir do endereço de origem do datagrama. O Alloy compara esse endereço com os mapas `devices:` no inventário do site. Uma correspondência define os rótulos `device` e `vendor`.

Dois resultados seguem desse design:

- O rótulo do dispositivo está correto mesmo quando o dispositivo envia uma identidade errada, uma identidade vazia ou uma identidade duplicada no

cabeçalho da mensagem.

- Um dispositivo que envia de um endereço que o inventário não lista mantém apenas o rótulo `device_ip`. Adicione o endereço ao inventário para dar um nome a esse dispositivo.

Se um dispositivo tiver mais de uma interface, certifique-se de que ele envie do endereço que o inventário lista. Defina o endereço de origem no dispositivo.

Configuração do Coletor

Os parâmetros do coletor estão em `roles/common/defaults/main.yml`. Substitua um parâmetro no inventário do site sob `all.vars`.

```
all:
  vars:
    network_syslog_port: 514        # porta do listener
    network_syslog_protocol: udp    # transporte
    network_syslog_vendors:         # blocos de fornecedores que
nomeiam um remetente
      - mikrotik
```

Parâmetro	Tipo	Requerido	Padrão	Descri
`network_syslog_port`	Inteiro	Não	`514`	Porta que listener v no host d monitora Uma port abaixo de é privileg Veja a no abaixo.
`network_syslog_protocol`	String	Não	`udp`	Transport o listener `udp` ou `t` maioria d dispositiv rede envi
`network_syslog_vendors`	Lista	Não	`[mikrotik]`	Blocos de forneced do invent cujo map `devices:` nomeia u remetent syslog. C forneced listado contribui os rótulos nome do dispositiv

O Alloy é executado como o usuário não privilegiado `alloy`. Portanto, uma porta abaixo de 1024 precisa de uma capacidade. A unidade Alloy em um host de monitoramento recebe `CAP_NET_BIND_SERVICE` automaticamente quando

`network_syslog_port` está abaixo de 1024. Uma porta acima de 1024 não recebe a capacidade.

Aplique uma alteração a esses parâmetros com o papel `common`:

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

Os hosts de monitoramento executam o Alloy para este listener. Portanto, um host de monitoramento também envia seu próprio journal, logs de SSH e de auditoria para seu Loki local.

Inventário de Dispositivos

O coletor lê os mesmos mapas de fornecedores que o exportador SNMP usa. Adicione cada dispositivo uma vez. A entrada então dá ao dispositivo um rótulo de log, um alvo SNMP e uma regra de firewall.

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

Parâmetro	Tipo	Requerido	Padrão	Descrição
`devices`	Mapa	Sim	-	Nome do dispositivo para endereço IP. O nome se torna o rótulo `device`. O endereço identifica o remetente e abre o firewall.
`snmp_community`	String	Não	`public`	Comunidade SNMP para o exportador SNMP do Prometheus. O coletor de syslog não a utiliza.

Rótulos de Log

O syslog de dispositivos chega sob `job="network_syslog"`. Os seguintes rótulos se aplicam.

Rótulo	Descrição	Exemplo
`job`	Valor fixo para syslog de dispositivos.	`network_syslog`
`component`	Valor fixo para syslog de dispositivos.	`network_device`
`device`	Nome do dispositivo do inventário. O Alloy combina o endereço do remetente.	`nf-nids-rtr-core01`
`device_ip`	Endereço de origem do datagrama.	`10.64.10.1`
`vendor`	Bloco de fornecedor do inventário que contém o dispositivo.	`mikrotik`
`level`	Severidade do syslog.	`informational`, `warning`, `err`, `crit`
`facility`	Instância do syslog.	`daemon`
`topics`	Lista completa de tópicos do RouterOS do cabeçalho da mensagem.	`system,info,account`
`topic`	Primeiro tópico nessa lista. Use-o para o filtro comum.	`system`, `firewall`, `dhcp`
`syslog_host`	Campo host do cabeçalho do syslog, como o remetente escreveu.	`system,info,account`

O RouterOS não envia identidade própria. Ele escreve sua lista de tópicos onde o nome do host pertence, então o parser armazena essa lista como o host do

syslog. O coletor lê `topics` e `topic` desse campo. Um remetente que escreve um nome de host real lá define apenas `syslog_host` e não cria rótulos de tópico.

MikroTik RouterOS

Configure cada dispositivo a partir do terminal do RouterOS. O dispositivo aponta sua ação de registro `remote` integrada para o host de monitoramento. O dispositivo então precisa de uma regra de registro para cada tópico.

```
/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
```

Propriedade	Valor	Descrição
`target`	`remote`	Envia o log para um servidor syslog remoto em vez de memória ou disco.
`remote`	Endereço do host de monitoramento	Destino do fluxo de syslog.
`remote-port`	`514`	Porta de destino. Combine com `network_syslog_port`.
`remote-log-format`	`bsd-syslog`	Apenas RouterOS 7. Envia RFC 3164. Veja a tabela de versões abaixo.
`src-address`	Endereço do dispositivo no inventário	Opcional. Defina-o quando o dispositivo tiver mais de uma interface, para que o endereço de origem corresponda à entrada do inventário.

Formato da Mensagem por Versão do RouterOS

O coletor analisa RFC 3164, então o dispositivo deve enviar esse formato. A propriedade que controla o formato mudou entre as versões principais. Verifique a versão com `/system resource print`.

Versão	Propriedade	Padrão	Ação
RouterOS 7	`remote-log-format`	`default`	Defina `remote-log-format=bsd-syslog`. O valor `default` envia uma estrutura MikroTik sem cabeçalho de prioridade syslog, que o coletor não pode analisar. A propriedade `bsd-syslog` do RouterOS 6 não existe aqui, e esse comando falha com `bad parameter bsd-syslog`.
RouterOS 6	`bsd-syslog`	`no`	Defina `bsd-syslog=yes`. O valor padrão envia uma mensagem simples, que o coletor não pode analisar.

```
# 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
```

Uma captura de pacotes mostra a diferença. O formato errado decodifica como `(invalid)` no `tcpdump`, e o payload começa com bytes binários antes do texto da mensagem. O formato correto decodifica como `SYSLOG, Facility local0, Severity info`.

Tópicos do RouterOS

Tópico	Descrição
`critical`	Falhas que interrompem um serviço no dispositivo.
`error`	Operações falhadas, por exemplo, um lease DHCP falhado ou um túnel falhado.
`warning`	Condições que precisam de atenção, mas não interrompem um serviço.
`info`	Eventos normais. Este tópico carrega os eventos de `account`, que são os logins de usuários, os logins falhados e as mudanças de configuração.

Comece com `error`, `warning` e `critical` para um volume baixo. Adicione `info` quando quiser auditoria de login e mudanças de configuração.

Verifique o resultado no dispositivo:

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

Outros Enviadores de Syslog

Qualquer dispositivo que fale BSD syslog pode usar o coletor. Três condições se aplicam:

1. O dispositivo envia BSD syslog, conforme definido na RFC 3164. O coletor não analisa a RFC 5424.
2. O dispositivo envia UDP para a porta 514 no host de monitoramento, a menos que o site tenha alterado `network_syslog_protocol` ou `network_syslog_port`.

3. O dispositivo envia de um endereço que um mapa `devices:` lista. O firewall permite esse endereço, e o endereço dá ao dispositivo seu nome.

Adicione um novo fornecedor a `network_syslog_vendors` para rotular seus dispositivos. O firewall já aceita os dispositivos de cada fornecedor.

Cisco IOS

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

Linux rsyslog

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

O template padrão do rsyslog é RFC 3164. Mantenha esse template. Não selecione `RSYSLOG_SyslogProtocol23Format`, porque esse template envia RFC 5424.

Aparelhos com uma interface web

Defina o servidor syslog remoto e a porta. Se o dispositivo oferecer uma escolha de formato, selecione BSD ou legado.

Firewall

O modelo de fluxo carrega a regra. O fluxo `4b` permite `514/udp` para o grupo `monitoring`. As fontes vêm do token `{ net_devices: true }`, que resolve cada endereço em cada mapa `all.vars.<vendor>.devices`.

Um dispositivo que não está listado no inventário é descartado quando o site executa o firewall em modo `enforce`. Adicione o dispositivo ao inventário e, em seguida, aplique o firewall:

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

Veja o guia Firewall por Papel para o modelo de fluxo e os modos.

Dashboard

O Grafana contém o dashboard **Syslog de Dispositivos de Rede** na pasta **Logs**.

Painel	Descrição
Pesquisa de Log de Dispositivo de Rede	Pesquisa de texto livre em todos os dispositivos.
Taxa de Log por Dispositivo	Volume para cada dispositivo ao longo do tempo. Uma linha plana em zero mostra um dispositivo que parou de enviar.
Taxa de Log por Severidade	Volume para cada severidade ao longo do tempo.
Logs de Dispositivos de Rede	Todos os logs de dispositivos para o dispositivo e string de pesquisa selecionados.
Erros, Avisos e Críticos	Severidade `warning` e acima.
Logins, Falhas de Autenticação e Mudanças de Configuração	Eventos de conta e sistema para auditoria.
Eventos de Link, OSPF/BGP e DHCP	Eventos de interface, roteamento e lease.

Variável	Descrição
`Site (logs)`	Fonte de dados Loki. Selecione o site.
`Device`	Dropdown de Dispositivo. A lista vem dos valores de rótulo `device` no Loki. Selecione um dispositivo, vários dispositivos ou `Todos`.
`Search`	Filtro de texto livre para os painéis de log.

Consultando os Logs

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

# Severidade warning e acima, em toda a propriedade
{job="network_syslog", level=~"emerg|alert|crit|err|warning"}

# Logins e mudanças de configuração
{job="network_syslog"} |~ `(?i)(logged in|login failure|changed
by)`

# Um tópico do RouterOS
{job="network_syslog", topic="firewall"}

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

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

Solução de Problemas

Sem Logs de Dispositivos no Loki

Sintomas: Uma consulta para `{job="network_syslog"}` não retorna resultados.

Possíveis causas:

- O Alloy não está em execução no host de monitoramento.
- O listener não vinculou a porta.
- O dispositivo não envia.
- O firewall descarta os datagramas.

Resolução:

1. Verifique o estado do agente no host de monitoramento com `systemctl status alloy`.
2. Verifique se o Alloy possui a porta com `ss -lunp | grep 514`.
3. Capture o tráfego com `tcpdump -n -i any udp port 514`. O tráfego prova que o dispositivo envia e que o firewall permite o datagrama.
4. Se não houver tráfego, verifique a ação de registro no dispositivo.

Alloy Não Inicia

Sintomas: O serviço `alloy` reinicia em um loop. O journal mostra um erro de vinculação ou um erro de permissão para a porta 514.

Possíveis causas:

- A unidade não possui `CAP_NET_BIND_SERVICE`.
- Outro processo ocupa a porta, por exemplo, um `rsyslogd` local com um listener UDP.

Resolução:

1. Leia o erro com `journalctl -u alloy -n 50`.

2. Confirme a capacidade com `systemctl cat alloy`.
3. Encontre o outro listener com `ss -lunp | grep 514`. Pare esse listener ou defina `network_syslog_port` para uma porta livre e configure os dispositivos para corresponder.

Datagramas Chegam, mas o Loki Não Armazena Nada

Sintomas: `tcpdump` mostra tráfego na porta 514. O Loki não possui fluxo `network_syslog`.

Possíveis causas:

- O dispositivo não envia BSD syslog. Um dispositivo RouterOS 7 precisa de `remote-log-format=bsd-syslog`. Um dispositivo RouterOS 6 precisa de `bsd-syslog=yes`.
- O dispositivo envia RFC 5424.

Resolução:

1. Capture o tráfego com `tcpdump -n -i any -A udp port 514`. Uma mensagem que o `tcpdump` marca como `(invalid)` não possui cabeçalho de prioridade syslog, então o formato está errado.
2. Defina a propriedade de formato para a versão do dispositivo. Veja Formato da Mensagem por Versão do RouterOS.
3. Em outro dispositivo, selecione o formato de syslog BSD ou legado.
4. Verifique o log do Alloy para erros de análise com `journalctl -u alloy | grep -i syslog`.

O Rótulo do Dispositivo Está Vazio

Sintomas: As linhas de log carregam `device_ip`, mas nenhum rótulo `device`.

Possíveis causas:

- O endereço do remetente não está em um mapa `devices:`.

- O dispositivo envia de uma segunda interface, então o endereço de origem difere do endereço do inventário.

Resolução:

1. Leia o endereço de origem do rótulo `device_ip`.
2. Adicione esse endereço ao mapa `devices:` do fornecedor ou defina o endereço de origem no dispositivo.
3. Aplique a alteração com o papel `common` e com o play do firewall se o site aplicar o firewall.

Um Dispositivo Parou de Enviar

Sintomas: O painel **Taxa de Log por Dispositivo** não mostra dados para um dispositivo. Outros dispositivos continuam.

Possíveis causas:

- Um operador removeu ou desativou a regra de registro no dispositivo.
- O dispositivo perdeu a rota para o host de monitoramento.
- O endereço do inventário mudou, então o firewall descarta o novo endereço de origem.

Resolução:

1. Verifique as regras no dispositivo com `/system logging print`.
2. Faça um ping no host de monitoramento a partir do dispositivo.
3. Confirme que o endereço atual do dispositivo corresponde à entrada do inventário.

Referências

Referência	Título
RFC 3164	O Protocolo BSD syslog
RFC 5424	O Protocolo Syslog. O coletor não analisa esse formato.
Registro Centralizado	Pipeline de logs do host, retenção e dashboards
Firewall por Papel	Modelo de fluxo, tokens de origem e modos
Monitoramento & Observabilidade	Prometheus, Grafana e métricas

Operações de Replicação e Nó do OmniHSS

OmniHSS armazena seus dados de assinante, autenticação (AuC) e IMS em um banco de dados PostgreSQL local em cada nó. **A replicação lógica bidirecional do PostgreSQL** mantém múltiplos nós do OmniHSS sincronizados. Isso forma uma **rede ativa-ativa**: cada nó é um primário completo de leitura/gravação, e uma gravação em qualquer nó se propaga para todos os outros. Não há primário/standby. Qualquer nó pode atender o tráfego S6a, Cx/Dx e Sh e aceitar provisionamento a qualquer momento.

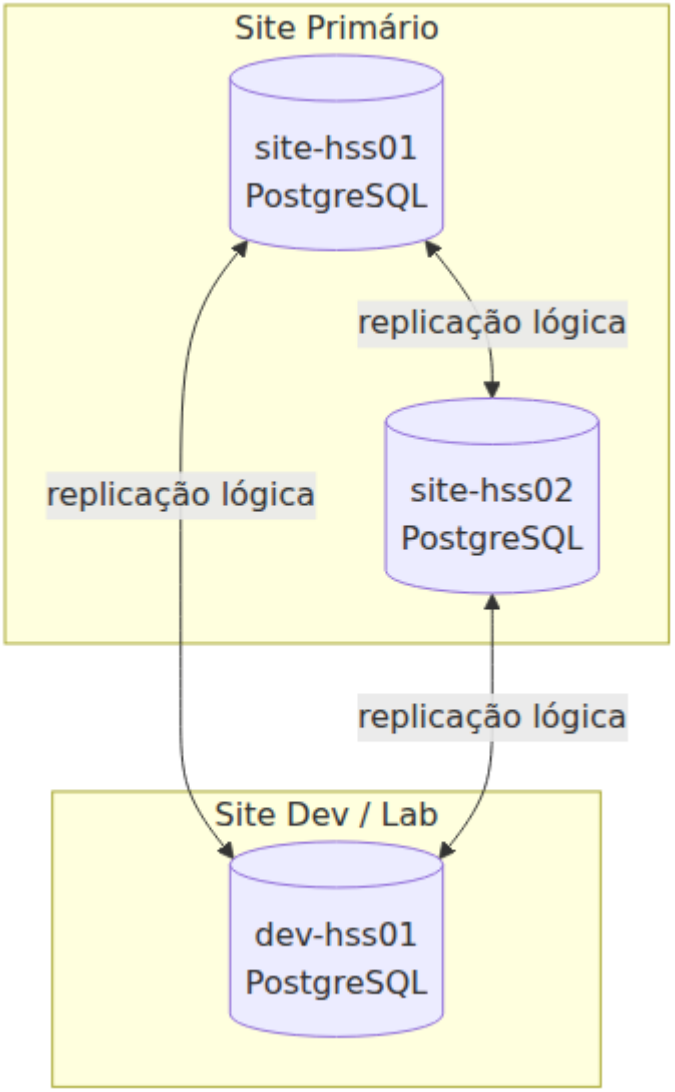
Este guia cobre o modelo de replicação. Ele explica como adicionar um novo nó do OmniHSS e carregá-lo com dados. Também explica como validar, reparar e remover pares.

Índice

- Visão Geral da Arquitetura
- Como os Pares São Descobertos
- Configuração
- Adicionando um Novo Nó do OmniHSS
- Carregando um Novo Nó com Dados
- Validando a Replicação
- Integridade dos Dados
- Removendo um Par
- Backup e Restauração
- Métricas
- Solução de Problemas

Visão Geral da Arquitetura

Cada nó do OmniHSS executa sua própria instância do PostgreSQL. Cada nó **publica** todas as suas tabelas de aplicação e **se inscreve** na publicação de cada par. `origin = none` previne loops de replicação. Nas tabelas de estado de rápida mudança, a rede resolve conflitos de gravação **last-write-wins (LWW)** comparando `updated_at`.

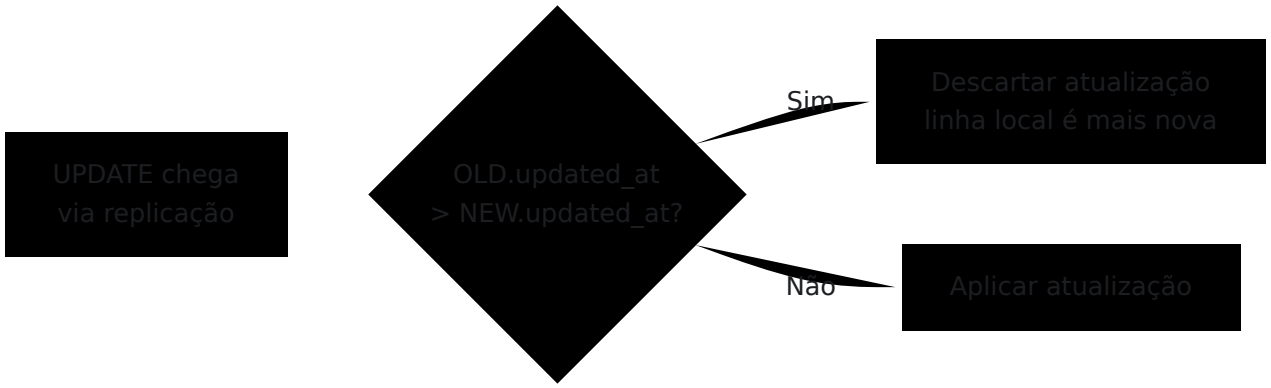


Cada link acima é duas inscrições unidirecionais (uma por direção). Para uma rede de `N` nós, cada nó possui `N-1` inscrições. Os pares consomem `N-1` dos slots de replicação da sua publicação.

O Que é Replicado

Categoria da Tabela	Exemplos	Tratamento de Conflitos
Dados mestres / provisionados	`subscriber`, `key_set` (AuC), `sim`, `msisdn`	Nenhum necessário: provisionado de um lugar de cada vez
Estado dinâmico	`subscriber_state`, `pdn_session`, `lte_call`, `ims_subscription`	Last-write-wins acionado em `updated_at`
Controle de migração	`schema_migrations`	**Excluído** da replicação

A publicação cobre todas as tabelas no esquema `public` **exceto** `schema_migrations` (cada nó executa suas próprias migrações de forma independente). Quando dois nós atualizam a mesma linha de estado simultaneamente, os acionadores LWW mantêm o `updated_at` mais recente em vez de qual atualização chegou por último.



Como os Pares São Descobertos

Quando a replicação está habilitada, o papel do OmniHSS constrói sua lista de pares automaticamente a partir de duas fontes:

1. **Outros hosts no grupo de inventário local `hss`**: os nós do OmniHSS definidos no *mesmo* arquivo de inventário/host (por exemplo, um par local `hss01` / `hss02`). Estes **não** precisam estar listados em `connected_omnihss`; a associação ao grupo é suficiente.
2. **A lista `connected_omnihss`** no arquivo de pares remotos do site: nós do OmniHSS definidos em um *arquivo* de host diferente (outros sites). Pares entre arquivos só são descobertos aqui, porque um inventário não tem conhecimento do grupo `hss` de outro.

O papel filtra pares do grupo local para hosts que são participantes da replicação do OmniHSS: aqueles cujo `omnihss.replication.enabled` é `true`. Um nó cuja flag não é `true` não é considerado como um par de replicação.

O papel exclui automaticamente o host atual de sua própria lista de pares, de modo que o mesmo bloco `connected_omnihss` pode ser compartilhado por cada site sem que um nó tente replicar de si mesmo.

Nomeação de Objetos

Cada nó nomeia seus objetos de replicação para que uma rede de qualquer tamanho nunca colida:

Objeto	Nome	Criado em	Único por
Publicação	`<self>_pub`	este nó	nó
Inscrição	`sub_from_<source>`	este nó (assinante)	(assinante)
Slot de Replicação	`<subscriber>_from_<source>`	a fonte (publicador)	par (assinante, fonte)

O nome do slot **deve** codificar o assinante. Considere o que acontece se ele codificar apenas a fonte (o padrão do PostgreSQL, onde o slot herda o nome da inscrição). Dois nós se inscrevendo na mesma fonte tentariam criar um slot com o mesmo nome nessa fonte. A segunda inscrição falharia com *"o slot de*

replicação já existe". Codificar o assinante dá a cada link um slot globalmente único.

Reconciliação e Segurança do `--limit`

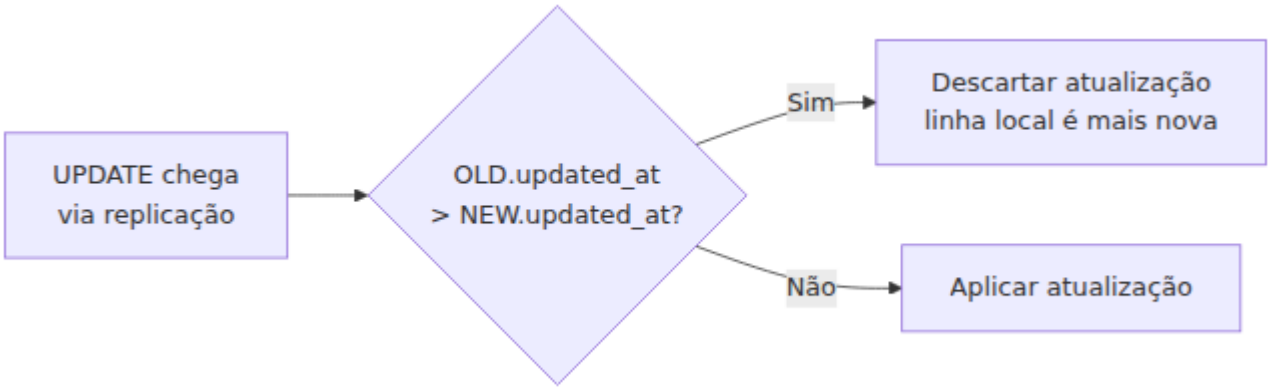
Cada execução de papel **reconcila** os objetos de replicação do nó contra o conjunto de pares válidos atual (grupo local `hss` + `connected_omnihss`, menos o próprio):

- Publicações diferentes de `<self>_pub` são descartadas (limpa automaticamente publicações obsoletas/renomeadas).
- Inscrições cujo par não está mais no conjunto válido são desativadas, destacadas de seu slot remoto e descartadas.
- Slots lógicos inativos que não alimentam um par válido atual são descartados. Slots ativos nunca são tocados.

Isso faz de `connected_omnihss` a lista de rede **autoritativa**: a próxima execução poda um par removido dela (e dos grupos `hss` relevantes) de cada nó. Duas propriedades de segurança impedem que isso seja destrutivo:

- **`--limit` é seguro.** O conjunto válido é derivado de `groups['hss']` e `connected_omnihss`, que sempre refletem a *totalidade* da associação ao inventário, independentemente de `--limit`. Executar `--limit hss01` reconcila apenas `hss01`. Mas `hss01` ainda vê `hss02`/`hss03` como válidos e mantém suas inscrições. `--limit` nunca poda os pares excluídos.
- **Conjuntos vazios nunca podam.** Os passos de poda são totalmente pulados em vez de apagar toda a replicação se o conjunto de pares válidos computar como vazio (por exemplo, um inventário parcial com um único HSS e `connected_omnihss` não carregado).

> **Advertência sobre a lista autoritativa:** como a poda é automática, a execução poda um par entre arquivos que é genuinamente parte da rede, mas está ausente de `connected_omnihss`. Sempre mantenha `connected_omnihss` completo para cada par entre arquivos.



Configuração

Habilitando a Replicação

A replicação é controlada por implantação pela flag `omnihss.replication.enabled`. Ela é `false` por padrão. Defina-a no inventário ou em group_vars do site:

```
omnihss:
  database_type: "postgres"
  database_host: "localhost"
  database_port: 5432
  database_name: "omnihss"
  database_username: "hss"
  database_password: "password"
  replication:
    enabled: true
```

Parâmetro	Tipo	Requerido	Padrão	Descr
`database_type`	String	Não	`postgres`	Backend do dados. A rep se aplica ap `postgres`.
`database_host`	String	Não	`localhost`	Host do ban dados. A rep configurada quando o ba dados é loca (`localhost`
`database_port`	Inteiro	Não	`5432`	Porta do Pos também usa a porta padr conexões de
`database_name`	String	Não	`omnihss`	Nome do ba dados. Deve corresponde todos os par
`database_username`	String	Sim	`hss`	Papel de log com o atribu `REPLICATIO` que possa c replicação lo
`database_password`	String	Sim	`password`	Senha para de login. De corresponde todos os par nas strings c conexão de

Parâmetro	Tipo	Requerido	Padrão	Descr
`replication.enabled`	Booleano	Não	`false`	Interruptor r Quando `tru` PostgreSQL configurado replicação lo publicações, são criadas.

Quando `replication.enabled` é `true`, o papel aplica as seguintes configurações do PostgreSQL na instância local:

Configuração	Valor
`wal_level`	`logical`
`max_wal_senders`	`10`
`max_replication_slots`	`10`
`listen_addresses`	`*`
`max_slot_wal_keep_size`	`10GB` (substituir via `omnihss.replication.max_slot_wal_keep_size`)

Configuração	Valor

`pg_hba.conf` é atualizado para permitir o IP de cada par (`/32`) para conexões de `all` e `replication`.

Declarando Pares Entre Sites (`connected_omnihss`)

Nós remotos do OmniHSS são declarados no arquivo de pares remotos referenciado pela variável `remote_peers_file` de cada site (comumente `prod_master_peers.yml`):

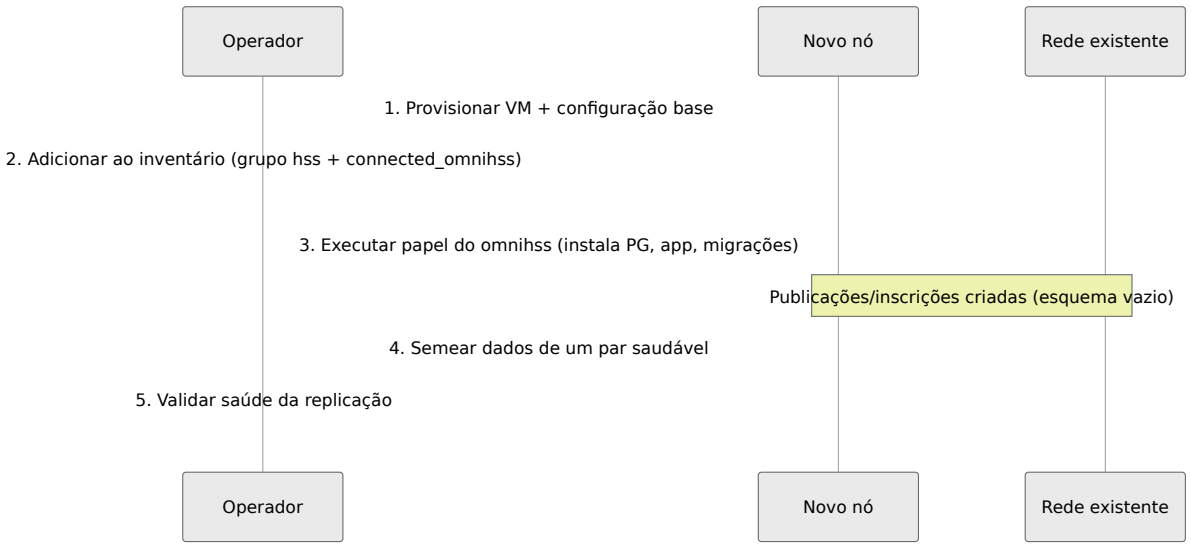
```yaml
connected_omnihss:
  - name: site-hss01
    host: 10.4.2.142
    port: 5432
  - name: site-hss02
    host: 10.4.2.143
    port: 5432
  - name: dev-hss01
    host: 10.4.10.142
    port: 5432
```

Parâmetro	Tipo	Requerido	Padrão	Descrição
`name`	String	Sim	-	O `inventory_hostname` do par. Isso nomeia a inscrição (`sub_from_<name>`) e resolve a publicação do par (`<name>_pub`), portanto, **deve** corresponder ao hostname real do inventário do par.
`host`	String	Sim	-	O endereço IP do par, acessível a partir deste nó pela rede de sinalização.
`port`	Inteiro	Não	`5432`	A porta do PostgreSQL do par. Reverte para `database_port`.

Como o arquivo de pares remotos é carregado por cada site, a lista completa de `connected_omnihss` deve conter **todos** os nós do OmniHSS na rede. Cada site automaticamente ignora seus próprios nós locais (eles já estão cobertos pelo grupo local `hss`) e ignora a si mesmo.

Adicionando um Novo Nó do OmniHSS

Adicionar um nó tem duas fases distintas: **entrar na rede** (para que as mudanças fluam em ambas as direções) e **carregar os dados existentes** no novo nó. A segunda fase é importante porque um nó completamente novo começa com um banco de dados vazio. Veja Carregando um Novo Nó com Dados.



Passo 1: Provisionar o host

Provisionar a VM e a rede base como para qualquer nó do OmniCore (veja Implantação Proxmox). O nó deve alcançar a porta do PostgreSQL de cada par (`5432` por padrão) pela rede de sinalização, e os pares devem conseguir alcançá-lo.

Passo 2: Adicionar o nó ao inventário

Adicione o novo host ao arquivo host do site sob o grupo `hss:`:

```
hss:
  hosts:
    site-hss01:
      ansible_host: 10.4.2.142
      gateway: 10.4.2.1
      host_vm_network: "v401-omni-signalling"
    site-hss03:        # novo nó
      ansible_host: 10.4.2.144
      gateway: 10.4.2.1
      host_vm_network: "v401-omni-signalling"
```

Em seguida, adicione-o a `connected_omnihss` no arquivo de pares remotos para que os **outros sites** também aprendam sobre ele:

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

Passo 3: Executar o papel do OmniHSS

Execute o playbook do serviço OmniHSS contra o novo host. Isso instala o PostgreSQL e o OmniHSS, executa as migrações do banco de dados (criando um esquema **vazio**), configura a replicação lógica e cria a publicação, além de uma inscrição para cada par:

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

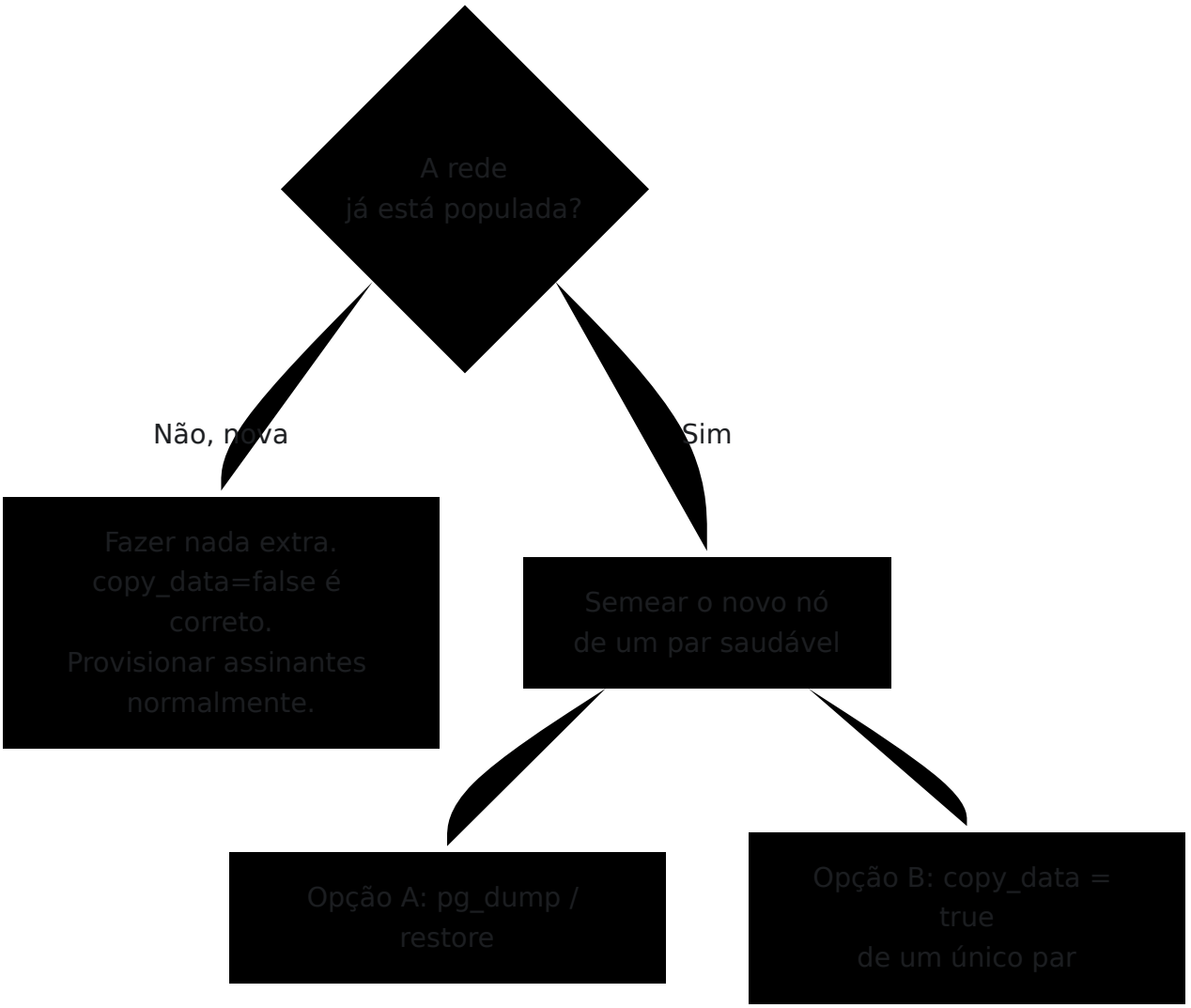
Re-execute o papel nos pares existentes (ou deixe a próxima execução agendada lidar com isso) para que eles abram `pg_hba.conf` para o novo nó e criem suas inscrições de volta para ele. As inscrições são idempotentes: o papel detecta e ignora as existentes e emite `ALTER SUBSCRIPTION ... REFRESH PUBLICATION` para que ele pegue quaisquer novas tabelas.

Neste ponto, o novo nó é um membro da rede e recebe **todas as futuras gravações**. Mas ele ainda **não** contém a base de assinantes existente. Prossiga para carregar seus dados.

Carregando um Novo Nó com Dados

O papel cria as inscrições automatizadas com `copy_data = false`. Isso é correto para uma **rede nova** (cada nó começa vazio ao mesmo tempo: não há nada para copiar, e copiar causaria conflitos de linhas duplicadas). Mas isso significa que um nó se juntando a uma **rede já populada** permanece vazio, exceto pelas mudanças que ocorrem após sua entrada.

Escolha o caminho que corresponde à sua situação:



Rede Nova

Se você está configurando todos os nós do OmniHSS juntos sem dados pré-existentes, não precisa semear. Habilite a replicação, execute o papel em todos os nós e provisionar assinantes normalmente. Cada gravação se replica para todos os membros.

Opção A: Semear via dump e restore (recomendado)

Tire uma cópia de um par saudável e restaure-a no novo nó. Em seguida, deixe a replicação carregar as mudanças contínuas. Isso reutiliza as ferramentas

padrão de backup/restauração e é a abordagem mais previsível, incluindo para grandes conjuntos de dados.

1. **Faça backup de um par saudável.** Isso produz um `pg_dump` do banco de dados OmniHSS (feito com `--no-publications --no-subscriptions`, para que objetos de replicação não sejam transferidos) na máquina de controle:

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

   O dump é salvo como `backups/<Site_Name>/hss_dump_<hostname>_<timestamp>.sql`.

2. **Restaure no novo nó.** O playbook de restauração para o OmniHSS, descarta e recria o banco de dados, restaura o dump e reinicia o serviço. Ele faz um backup antes da restauração do alvo:

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

   Adicione `-e skip_pre_restore_backup=true` para pular esse dump de pré-restauração — útil quando o alvo é um nó novo ou já vazio, ou quando o dump é lento e seu conteúdo atual é conhecido como inútil. Nada é então salvo, portanto, os dados atuais do nó não podem ser recuperados.

   Quando solicitado, forneça o caminho para o dump SQL do passo 1 e deixe o caminho de configuração vazio (o novo nó já tem sua própria configuração da execução do papel).

3. **Confirme que a replicação está fluindo.** As inscrições existentes do novo nó (`copy_data = false`) agora aplicam mudanças em cima do snapshot restaurado. Execute a validação abaixo.

> **Janela de manutenção:** Há uma pequena janela entre o dump e o momento em que a replicação se atualiza. Durante isso, mudanças de provisionamento em pares ainda não estão no novo nó. Os dados do

> assinante mestre mudam com pouca frequência, e o estado dinâmico (`subscriber_state`, `pdn_session`, `lte_call`, `ims_subscription`) se cura a partir do tráfego ao vivo, então uma janela de baixa atividade é suficiente. Não provisionar novos assinantes durante a semeadura.

Opção B: Semear com o playbook de semeadura (recomendado)

O playbook utilitário `seed_hss_replication.yml` semeia de **um** par fonte escolhido sem as armadilhas de uma cópia em massa bruta. Ele recria a inscrição como somente streaming (`copy_data = false`, sem tablesync). Ele então preenche as linhas existentes da **tabela mestre** com uma carga segura contra conflitos. `subscriber_state` **é** parte do preenchimento: é um dado estrutural 1:1 (cada assinante referencia um por FK, criado uma vez no provisionamento). Ignorá-lo deixa cada assinante preenchido órfão, o que quebra a API e S6a no alvo. Streaming não pode curar isso, porque a replicação lógica silenciosamente descarta UPDATEs para linhas que não existem. O playbook deliberadamente não faz cópia em massa das verdadeiras tabelas de sessão transitórias (`pdn_session`, `lte_call`). Um `COPY` dessas tabelas quentes colide (chave duplicada) com gravações concorrentes em um sistema ao vivo e causa loops de falha na sincronização de tabelas. Elas se repopulam a partir da replicação streaming e do tráfego ao vivo.

Interativo (recomendado): omita a fonte. O playbook então encontra quais alvos estão vazios, investiga os pares em busca de aqueles que possuem dados e solicita que você escolha:

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

```
Pares com dados disponíveis para semear:
  1) dev-hss01 (10.4.10.142) - 838 assinantes
Digite o NÚMERO da fonte para semear
```

Explícito: nomeie a fonte diretamente. Isso força uma nova semeadura mesmo que o alvo já tenha dados (use para recuperação):

```
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"
```

Veja Semente de Replicação HSS para a referência completa de parâmetros e comportamentos.

`seed_hss_replication.yml` também **conecta a rede completa** após a semeadura: ele executa a reconciliação de replicação em cada nó (publicações, inscrições, slots) e valida isso. Prefira executá-lo com **sem** `**--limit**` — nós vazios são detectados automaticamente como alvos de semeadura, nós populados como fontes, e cada nó é conectado em uma única passagem. `--limit` restringe tanto a semeadura quanto a conexão, então um único nó `--limit` deixaria a rede meio conectada; use-o apenas para uma nova semeadura direcionada, depois re-execute sem `--limit` para conectar o restante. Semear de exatamente uma fonte. Prefira a Opção A para conjuntos de dados muito grandes.

Os passos manuais equivalentes são executados no nó alvo. Primeiro, crie a inscrição de streaming (note o `slot_name` explícito por link, veja Nomeação de Objetos). Em seguida, preencha as tabelas mestres de forma segura contra conflitos, enquanto ignora as tabelas de estado quentes:

```
-- 1. transmitir todas as tabelas, sem cópia em massa
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. preencher apenas as tabelas mestres (executar no alvo; -t
cada tabela exceto
#    schema_migrations, pdn_session, lte_call - subscriber_state
DEVE ser
#    incluído, ou cada assinante preenchido ficará órfão: seu
#    subscriber_state_id aponta para uma linha que nunca chega,
quebrando a
#    API/S6a, e streaming não pode curá-lo porque replicação
lógica
#    silenciosamente descarta UPDATEs para linhas ausentes).
#    A carga DEVE ser marcada com uma origem de replicação:
origin=none
#    inscrições filtram mudanças marcadas no publicador, então o
preenchimento é
#    invisível para a rede. Uma carga não marcada é decodificada
do WAL como qualquer
#    gravação local e ecoa as inserções para pares que já possuem
as linhas,
#    atrasando seus trabalhadores de aplicação por chave
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 a Replicação

O papel executa uma verificação de saúde automaticamente após a configuração, e também pode ser executado por conta própria. A validação afirma que:

- Cada assinatura tem um processo de trabalho ativo.

- Cada slot de replicação lógica está ativo.
- O atraso do WAL está abaixo de 100 MB por slot.
- O status do WAL de cada slot é `reserved` ou `extended` (não `lost`).

Para verificar a saúde sob demanda, re-execute o papel OmniHSS (a validação é executada no final), ou inspecione diretamente em um nó:

```
# Trabalhos de assinatura - pid deve ser não nulo para cada um
sudo -u postgres psql -d omnihss -c \
  "SELECT subname, pid IS NOT NULL AS worker_alive FROM
pg_stat_subscription;"

# Slots de replicação - ativo deve ser verdadeiro, wal_status
reservado/estendido
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';"
```

Sintoma	Significado
`worker_alive = false`	O trabalhador de assinatura está inativo: o par está inacessível ou as credenciais/string de conexão estão erradas.
`active = false` (slot)	Nenhum assinante está consumindo o slot: a assinatura do par está inativa ou foi removida.
Aumento de `lag`	O par está lento ou o link está degradado. O WAL está se acumulando e consumindo disco.
`wal_status = lost`	O slot ficou muito atrasado e o PostgreSQL removeu o WAL. Você deve **re-seed** o par (veja Carregando um Novo Nó com Dados).

O relatório de verificação de saúde (`health_check.yml`) também inclui um painel de **Replicação HSS Postgres** para nós com replicação habilitada,

mostrando o estado do trabalhador de cada assinatura e o estado ativo de cada slot e status do WAL de relance.

Integridade dos Dados

A aplicação de replicação lógica não impõe chaves estrangeiras. O trabalhador de aplicação é executado com `session_replication_role = replica`, o que suprime os gatilhos internos de RI que implementam restrições de `FOREIGN KEY`. Portanto, o PostgreSQL **nunca re-verifica** uma linha que chega via replicação em relação ao seu pai. As restrições FK oferecem integridade *por nó*, não integridade *em toda a malha*.

Em uma malha ativa-ativa que abre um buraco que nenhum nó único pode capturar:

- O Nó A exclui `epc_profile` X. Seu `ON DELETE RESTRICT` passa porque A não tem assinante local referenciando X naquele instante.
- O Nó B (ou uma etapa de provisionamento) aponta o assinante 203 para X. Isso passa porque X ainda existe no B.
- Ambas as mudanças replicam e aplicam sem verificações de FK. Cada nó converge em "203 → X" **e** "X excluído": uma referência pendente que nenhuma FK de nó violou localmente.

Uma `subscriber.epc_profile_id` pendente torna o HSS incapaz de construir dados de assinatura para esse assinante. O HSS então responde `DIAMETER_ERROR_USER_UNKNOWN (5001)` no ULR/AIR, mesmo que a linha do assinante exista e esteja habilitada. A janela de esquema antes dos dados (abaixo) torna isso mais provável, porque transações puladas podem separar a criação/exclusão do pai do re-apontamento do filho.

Detecção (não imposição)

Impor FKs no caminho de aplicação interromperia a malha em qualquer anomalia, então, em vez disso, nós **detectamos e alarmamos**. Cada nó instala uma função `SECURITY DEFINER`, `omnihss_fk_orphan_count()`, que conta cada FK de coluna única cujo valor aponta para um pai ausente em todas

as tabelas. `postgres_exporter` seleciona isso, e o Grafana gera alarmes quando é maior que zero (veja Métricas).

Execute a varredura manualmente a qualquer momento:

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

Para localizar linhas problemáticas para uma referência específica (exemplo: assinantes apontando para um 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;
```

Remedie re-apontando a linha para um pai válido (um `UPDATE` em uma linha existente replica limpo através da malha) ou removendo-a.

Prevenindo a deriva do esquema

O gatilho raiz do incidente que motivou esta seção foi uma **migração de esquema aplicada em um nó enquanto seus pares ainda estavam no esquema antigo**, e novos dados escritos contra os novos campos. A fonte publicou linhas que os pares do esquema antigo não puderam aplicar, causando erros de aplicação e **transações puladas**, e, portanto, divergência.

- **Migre toda a malha junta.** Aplique migrações a cada nó OmniHSS antes de provisionar contra novos campos. Não escreva dados do novo esquema enquanto qualquer par estiver atrasado.
- **Fique de olho na deriva.** O nó exporta `omnihss_schema_max_version` / `omnihss_schema_migration_count` por nó. Compare-os através da malha. Um nó atrasado em relação aos seus pares é uma migração meio aplicada.
- **Fique de olho nos contadores de erro.** Um aumento em `omnihss_subscription_error_apply_error_count` / `_sync_error_count` significa que o trabalhador de aplicação está falhando (e possivelmente

pulando). O alerta "Erros de Aplicação/Sincronização de Replicação HSS" dispara sobre isso.

Removendo um Par

Remover um par exclui a assinatura **desse** par e remove sua entrada `pg_hba.conf`. Os dados já replicados permanecem; apenas as futuras alterações param de fluir. A remoção deve ser confirmada explicitamente.

No nó de onde você está removendo o par:

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

Em seguida, remova a(s) linha(s) do par de `pg_hba.conf` e recarregue o PostgreSQL:

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

O par que está sendo removido ainda mantém sua assinatura de volta para este nó. Se o link dever ser totalmente desmontado, remova essa assinatura lá também. Em seguida, remova o nó de `connected_omnihss` e do grupo `hss` para que a próxima execução do papel não o recrie.

Backup & Restauração

O backup e a restauração do OmniHSS usam dumps padrão do PostgreSQL:

Operação	Playbook	Notas
Backup	`services/backup.yml`	Produz `hss_dump_<host>_<ts>.s` (banco de dados) e `hss_<host>_<ts>.tar.gz` (a configuração de `/etc/omnihss`) sob `backups/<Site_Name>/`. (dump exclui publicações/assinaturas.
Restauração	`util_playbooks/restore_hss.yml`	Solicita o dump SQL e/ou arquivo de configuração. Para o OmniHSS, faz um backup pré-restauração, exclui/recria o banco de dados, restaura e reinicia

Veja Playbooks Utilitários para o fluxo de trabalho mais amplo de backup/restauração.

Restaurando um nó em uma malha ativa-ativa

`restore_hss.yml` **desmonta a replicação no nó que restaura**: antes de excluir e recriar o banco de dados, ele exclui as assinaturas, publicações e slots desse nó. Portanto, após uma restauração, o nó não possui **objetos de replicação** — a malha deve ser reconfigurada, e reconfigurar apenas o *par* não é suficiente (o nó restaurado ainda precisa de sua própria publicação de volta). Duas regras:

1. **Restaure exatamente UM nó.** Restaurar em dois ou mais nós ao mesmo tempo destrói a malha (`restore_hss.yml` alerta sobre isso). Restaure um, depois semeie o restante a partir dele.
2. **Todos os nós devem executar a MESMA versão do OmniHSS antes de você conectar a replicação.** A replicação lógica transporta dados, não esquema. `CREATE SUBSCRIPTION` verifica que **todas as tabelas na**

publicação do publicador já existem no assinante, então um nó em uma versão mais nova (cuja publicação inclui uma tabela que uma migração posterior adicionou, por exemplo, `ims_subscription`) não pode ser assinado por um par ainda no esquema mais antigo — a assinatura falha com `relation "public.<table>" does not exist` e essa direção nunca aparece. O esquema do **dump** de restauração também deve corresponder à versão implantada, ou o nó restaurado acaba à frente de seus pares.

Reconstrua um par ativo-ativo do zero:

```
# 1. Provisione ambas as VMs, depois instale OmniHSS + esquema em
AMBOS na MESMA
#    versão (esquema idêntico, vazio e migrado em cada um).
ansible-playbook -i hosts/<inv>/host_files/<site>.yml omnihss.yml

# 2. Restaure dados em APENAS UM nó (o esquema do dump deve
corresponder à versão do passo 1).
#    Isso apaga os objetos de replicação desse nó; o passo 3
reconstrói a malha.
ansible-playbook -i hosts/<inv>/host_files/<site>.yml \
  util_playbooks/restore_hss.yml --limit <site>-hss01

# 3. Semeie o(s) outro(s) nó(s) E conecte a malha bidirecional
completa — SEM --limit.
#    Nós vazios são preenchidos a partir do nó restaurado;
publicações,
#    assinaturas e slots são (re)criados em cada nó, e depois
validados.
ansible-playbook -i hosts/<inv>/host_files/<site>.yml \
  util_playbooks/seed_hss_replication.yml
```

Não termine com `seed_hss_replication.yml --limit <peer>`: `--limit` limita a fiação também, então cria apenas a assinatura do par e deixa o nó restaurado sem sua publicação — uma malha meio conectada e unidirecional. `validate_replication.yml` (executado automaticamente) agora afirma que cada nó se inscreve em cada par, então um resultado meio conectado falha de forma barulhenta em vez de relatar saúde.

Métricas

Cada nó OmniHSS executa `postgres_exporter`, expondo métricas do PostgreSQL (incluindo replicação) ao Prometheus. Séries úteis para a saúde da replicação:

Métrica: `pg_replication_slots_active` **Tipo**: Gauge **Descrição**: Se cada slot de replicação lógica tem um consumidor ativo. **Rótulos**:

- `slot_name`: o slot de replicação (um por par assinante)

Métrica: `pg_stat_replication_*` / atraso do slot **Tipo**: Gauge **Descrição**: Atraso da posição do WAL por slot. Mostra o quão atrasado um par consumidor está.

Consultas de exemplo:

```
# Qualquer slot de replicação lógica inativa (alerta se > 0)
count(pg_replication_slots_active == 0)

# Assinaturas que deveriam existir vs trabalhadores em execução
podem ser verificadas
# contra a saída do playbook de validação durante a implantação.
```

Métricas personalizadas do OmniHSS

`postgres_exporter` em cada nó também exporta séries de saúde e integridade de replicação específicas do OmniHSS via um arquivo de consulta personalizado (`roles/omnihss/files/postgres_exporter_queries.yaml`). O Prometheus coleta esses sob o trabalho `HSS Postgres`. Eles sustentam as regras de alerta no grupo **Alertas HSS** (`roles/monitoring/templates/grafana/rules/`):

Métrica	Tipo	Sig
`omnihss_subscription_worker_alive{subname}`	Gauge	1 se trab de de esti exe
`omnihss_replication_slot_active{slot_name}`	Gauge	1 se esti con o sl
`omnihss_replication_slot_wal_status_code{slot_name}`	Gauge	0=r 1=e 2=r res **3=**
`omnihss_replication_slot_lag_bytes{slot_name}`	Gauge	Byt WA par
`omnihss_subscription_error_apply_error_count{subname}`	Counter	Erro cun do trab de
`omnihss_subscription_error_sync_error_count{subname}`	Counter	Erro cun

Métrica	Tipo	Sig
		de sinc de
`omnihss_integrity_fk_orphans`	Gauge	Linl um estı per (de
`omnihss_schema_max_version` / `omnihss_schema_migration_count`	Gauge	Mar alta mig apli con

Veja Monitoramento & Observabilidade para o pipeline de métricas e painéis.

Solução de Problemas

Novo nó permanece vazio após a junção

Sintomas: Trabalhadores de replicação estão ativos e slots estão ativos, mas o novo nó não tem (ou poucos) assinantes.

Possíveis causas:

- O nó se juntou a uma malha já populada e nunca foi semeado. As assinaturas usam `copy_data = false`, então apenas as mudanças pós-junção chegam.

Resolução:

1. Semeie o nó a partir de um par saudável. Veja Carregando um Novo Nó com Dados.

Trabalhador de assinatura não está em execução

Sintomas: `pid IS NULL` para uma assinatura em `pg_stat_subscription`.

Possíveis causas:

- Par inacessível na porta PostgreSQL (firewall / roteamento).
- IP do par ausente no `pg_hba.conf` deste nó.
- Credenciais erradas na string de conexão (`database_username` / `database_password` diferem entre os nós).

Resolução:

1. Confirme a acessibilidade TCP para o par em `5432`.
2. Verifique se o par aparece em `connected_omnihss` (ou no grupo local `hss`) e re-execute o papel para que `pg_hba.conf` seja atualizado.
3. Certifique-se de que `database_username`/`database_password` sejam idênticos em todos os nós.

Slot de replicação mostra `wal_status = lost`

Sintomas: O `wal_status` de um slot é `lost`; o par correspondente está muito atrasado.

Possíveis causas:

- O par esteve inativo ou inacessível por tempo suficiente para que o WAL necessário fosse removido.

Resolução:

1. Re-seed o par afetado a partir de um nó saudável. Veja Carregando um Novo Nó com Dados.

Erros de chave duplicada durante a cópia inicial

Sintomas: Erros ao aplicar alterações logo após criar uma assinatura com `copy_data = true`.

Possíveis causas:

- A cópia inicial foi executada a partir de mais de um par, copiando as mesmas linhas duas vezes.

Resolução:

1. Semeie de **exatamente um** par com `copy_data = true`; mantenha todas as outras assinaturas em `copy_data = false`.

Atualizações de estado conflitantes entre sites

Sintomas: Uma linha de estado (por exemplo, `subscriber_state`) parece "flutuar" entre valores definidos em diferentes sites.

Possíveis causas:

- Atualizações concorrentes na mesma linha em dois sites.

Resolução: Isso é esperado e tratado. O gatilho de último a escrever vence mantém a atualização com o `updated_at` mais recente. Confirme que os relógios estão sincronizados (NTP) em todos os sites para que a ordenação de `updated_at` seja significativa.

Implantação de VM/LXC do Proxmox

A maioria de nossos clientes executa a pilha OmniCore no Proxmox. Um único playbook, `util_playbooks/proxmox.yml`, provisiona tanto VMs QEMU quanto contêineres LXC com base em um toggle `deployment_type`. Você pode definir o toggle por site (na entrada `proxmoxServers`) e substituí-lo por host. Isso permite que um site funcione principalmente como LXC enquanto hosts individuais (por exemplo, o UPF) permanecem como VMs completas.

Ainda suportamos VMware, HyperV e nuvem (Vultr / AWS / GCP) para implantações.

Veja Também:

- Configuração do Arquivo de Hosts: defina VMs/LXCs a serem implantados
- Padrão de Planejamento de IP
- Configuração do Netplan
- Arquitetura de Implantação

LXC vs VM

Contêineres LXC:

- Leves, compartilham o kernel do host
- Inicialização rápida, baixa sobrecarga
- Isolamento limitado
- Não pode executar kernels ou módulos de kernel personalizados
- **Não adequado para implantações EPC/IMS em produção**
- **Não pode executar UPF** (requer módulos de kernel/dispositivos TUN)

VMs (KVM):

- Virtualização completa com kernel dedicado

- Isolamento completo, executa módulos de kernel / rede personalizada
- Maior sobrecarga de recursos
- **Recomendado para produção**
- **Necessário para implantações UPF**

Casos de Uso:

- **VMs**: sites de produção, UPF, todas as funções de rede
- **LXC**: ambientes de laboratório/teste, serviços leves (apt-cache, monitoramento, dns)

Escolhendo o Tipo de Implantação

`deployment_type` é resolvido para cada host nesta ordem:

1. **Por host** `deployment_type` (definido sob o nome do host): vence se presente.
2. **Por site** `deployment_type` na entrada `proxmoxServers` que o round-robin seleciona para o host.
3. **Padrão** `vm` se nenhum dos dois estiver definido.

O padrão comum é definir o padrão do site na entrada `proxmoxServers` (por exemplo, `deployment_type: lxc`) e substituir apenas os hosts que devem diferir. O exemplo canônico é um laboratório que executa tudo como LXC, exceto o UPF, que precisa de módulos de kernel / TUN e deve ser uma VM:

```yaml
all:
  vars:
    proxmoxServers:
      pve-node-01:
        deployment_type: lxc     # padrão do site → tudo é LXC
        # ...

upf:
  hosts:
    site-upf01:
      ansible_host: 192.168.1.24
      deployment_type: vm         # substituição por host → este host é uma VM
```

Como convenção, mantenha o padrão do site consistente e substitua apenas as exceções. Quando você espalha VMs e LXC por muitos hosts, a colocação dos nós depende do índice do host e é difícil de raciocinar.

O playbook falha apenas em casos de colisão de tipo com recursos existentes: se um recurso com o nome do host de destino já existir como o tipo errado (por exemplo, um LXC existe, mas o host resolve para `deployment_type: vm`, ou vice-versa), ele aborta com um erro claro. Veja Guardiões de Tipo.

Configuração do Proxmox

1. Criar Token de API

```
# Proxmox UI: Datacenter → Permissões → Tokens de API
# Criar token: root@pam!<TokenName>
# Copie o segredo do token (mostrado uma vez)
# Desmarque a caixa `Separação de Privilégios`
```

2a. Criar Modelo de VM Cloud-Init (apenas sites de VM)

Execute este script no host Proxmox. Ele baixa a imagem em nuvem do Ubuntu e cria um modelo limpo. `proxmox.yml` injeta credenciais do cloud-init e chaves SSH no momento da clonagem (veja Notas abaixo).

```
#!/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:

- O modelo não possui usuários ou senhas codificados. `proxmox.yml` define `ciuser`, `cipassword` e `sshkeys` no momento da clonagem.
- Precedência do usuário do cloud-init: `proxmoxTemplateUser` (na entrada correspondente `proxmoxServers`) → primeira chave `local_users`.
- Precedência da senha do cloud-init: `proxmoxTemplatePassword` (na entrada correspondente `proxmoxServers`) → o usuário do cloud-init (ou seja, o valor escolhido acima).

- As chaves SSH são injetadas de cada entrada `local_users.*.public_key` (a autenticação baseada em chave é o caminho de login pretendido).
- Defina `proxmoxTemplateUser` (e opcionalmente `proxmoxTemplatePassword`) em cada entrada `proxmoxServers` sempre que o modelo foi preparado com um usuário cloud-init que não é a primeira entrada `local_users`, para que a conta correta seja selecionada.
- `--vga std` garante que o console da web do Proxmox funcione.
- A flag `-N` no wget baixa apenas se for mais recente que a cópia local.

Alternativa: Modelo Manual a partir de ISO

Se as imagens em nuvem não estiverem disponíveis ou você precisar de uma instalação personalizada:

Passo 1: Criar VM via UI da Web

- Criar Nova VM → ID da VM 9000, Nome: ubuntu-2404-template
- SO: Carregar ISO do Ubuntu Server ou usar ISO existente
- Sistema: Padrão (Controlador SCSI: VirtIO SCSI)
- Discos: 32GB, Barramento: SCSI
- CPU: 2 núcleos
- Memória: 2048 MB
- Rede: VirtIO, ponte vmbr0
- Iniciar VM e instalar Ubuntu Server

Passo 2: Dentro da VM - Limpar e preparar

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

# Limpar dados específicos da 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

# Limpar histórico do bash e desligar
history -c
sudo poweroff
```

Passo 3: Adicionar Cloud-Init e Converter em Modelo

- Selecionar VM → Hardware → Adicionar → Unidade CloudInit (selecionar armazenamento, por exemplo, local-lvm)
- Hardware → Opções → Agente QEMU → Habilitar
- Clique com o botão direito na VM → Converter em Modelo
- **não** defina usuário/senha do cloud-init no modelo. `proxmox.yml` injeta esses valores no momento da clonagem a partir de `proxmoxTemplateUser` / `proxmoxTemplatePassword` (com fallback para `local_users` para o nome de usuário).

2b. Baixar Modelo de SO LXC (apenas sites de LXC)

```
# No shell do nó Proxmox:
pveam update
pveam download local ubuntu-24.04-standard_24.04-2_amd64.tar.zst
```

O caminho do volume resultante (por exemplo, `local:vztmpl/ubuntu-24.04-standard_24.04-2_amd64.tar.zst`) é o que você configura como `proxmoxLxcOsTemplate` abaixo.

Configuração do Arquivo de Hosts

Implantação de VM

```yaml
all:
  vars:
    # deployment_type omitido → padrão para "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        # nome de usuário
cloud-init opcional; padrão para a primeira chave local_users
        proxmoxTemplatePassword: omnitouch    # senha cloud-init
opcional; padrão para o nome de usuário cloud-init
        storage: local-lvm  # opcional
      pve-node-02:
        # ... configuração do segundo nó

    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, padrão 24
    vlanid: 100  # opcional
```

Implantação de LXC

```yaml
all:
  vars:
    proxmox_lxc_nameserver: "1.1.1.1"   # opcional, padrão 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, fallback
do rootfs
      pve-node-02:
        deployment_type: lxc
        # ... mesma estrutura

    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, padrão 24
    vlanid: 100                         # opcional
    proxmoxLxcCores: 2                  # substituição opcional
    proxmoxLxcMemoryMb: 4096            # substituição opcional
    proxmoxLxcDiskSizeGb: 30            # substituição opcional
    proxmoxLxcRootFsStorageName: local-lvm  # opcional, substitui
o armazenamento padrão do servidor proxmoxLxcDefaultStorage
```

Uso

Provisionar (VM ou LXC)

```
ansible-playbook -i hosts/Customer/site.yml
util_playbooks/proxmox.yml
```

Deletar

```
ansible-playbook -i hosts/Customer/site.yml
util_playbooks/proxmox_delete.yml
```

O playbook de exclusão lida automaticamente com VMs e LXCs, independentemente do `deployment_type`.

Comportamento do Playbook

Guardiões de Tipo

- Se uma VM com o nome do host de destino existir, mas `deployment_type: lxc` → o playbook falha com um erro claro.
- Caso inverso (LXC existe, site configurado como VM) → o mesmo.
- Grupo UPF em um site LXC → apenas aviso (não é uma falha crítica, mas o UPF não funcionará).

Novo Recurso

- Round-robins nas entradas `proxmoxServers` por índice de host dentro do grupo, a menos que o host defina `proxmox_server:` para fixar uma entrada específica (veja Distribuição Entre Nós).
- Consulta a API do Proxmox para o próximo VMID livre.
- **VM**: clona de `proxmoxTemplateId`, define usuário/senha/chaves SSH do cloud-init (usuário cloud-init = `proxmoxTemplateUser` da entrada do

servidor se definido, caso contrário, a primeira chave `local_users`; senha cloud-init = `proxmoxTemplatePassword` se definida, caso contrário, o nome de usuário do cloud-init), redimensiona `scsi0`, inicia.

- **LXC**: cria a partir de `proxmoxLxcOsTemplate`, rootfs em `proxmoxLxcRootFsStorageName` (grupo) → `proxmoxLxcDefaultStorage` (servidor) → `storage` → `local-lvm` (fallback final), injeta todas as `local_users.*.public_key` via `ssh-public-keys`, inicia. Senha root codificada como `0mn1Touch@`. Todos os LXCs são não privilegiados (`unprivileged: 1`) com aninhamento habilitado (`features: nesting=1`).

Recurso Existente

Tanto o caminho de atualização de VM quanto de LXC diferenciam o atual do desejado e aplicam as mudanças:

- Configuração de rede (ponte, tag VLAN, IP para LXC)
- Núcleos / memória
- Tamanho do disco (VM `scsi0` / LXC `rootfs`)
- NICs secundárias de `secondary_ips` Reinicia o recurso se a rede ou os recursos mudarem.

> **OmniUPF:** a ferramenta trata uma entrada `secondary_ips` chamada `n3` (ou `data`/`gtp`) como uma NIC dedicada do plano de usuário. O papel `omniupf` vincula N3/N9 + XDP a ele e reserva uma CPU para a interface primária, para que uma carga pesada de datapath não possa esgotar PFCP/API/SSH (que permanecem no IP primário).

Marcação

Cada recurso é marcado com seus nomes de grupos Ansible + `site_name`.

Distribuição Entre Nós

Por padrão, a ferramenta espalha hosts pelas entradas `proxmoxServers` por round-robin, com a mesma lógica para VMs e LXCs. O índice é a posição do

host **dentro de seu grupo de função**, tomado módulo do número de entradas `proxmoxServers` (na ordem de declaração):

```
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)
```

> **Adicionar um segundo nó redistribui tudo.** Como a colocação é posicional, passar de uma entrada `proxmoxServers` para duas muda onde grupos existentes de múltiplos hosts ficam (host-0 → entrada-0, host-1 → entrada-1, ...). Fixe os hosts que você não deseja mover (abaixo) antes de executar novamente contra uma frota mista.

Fixando um host a um nó específico

Defina a chave opcional `proxmox_server:` para o **nome de uma entrada** **`proxmoxServers`** para forçar um host a esse nó, independentemente de seu índice de grupo. Quando não definido, a lógica de round-robin acima se aplica. Um valor que não é uma chave `proxmoxServers` falha a execução com um erro que lista os nomes válidos (para que um erro de digitação não possa redirecionar silenciosamente para outro lugar).

```yaml
device_tester:
  hosts:
    site-device-tester01:
      ansible_host: 10.179.2.50
      proxmox_server: nicklab-prxmx01   # fixar a esta entrada proxmoxServers (→ seu nó)
      # Se o armazenamento em disco desse nó diferir do nome global proxmoxLxcRootFsStorageName,
      # substitua aqui também (o armazenamento é lido globalmente, não da entrada do servidor):
      proxmoxLxcRootFsStorageName: local-lvm
```

`proxmox_server` também pode ser definido no escopo de `vars:` do grupo para fixar cada host em um grupo a um nó.

Substituições por Host

`deployment_type`, `proxmox_server` (fixação de nó: veja Distribuição Entre Nós), e qualquer um dos parâmetros de nível de grupo LXC (`proxmoxLxcCores`, `proxmoxLxcMemoryMb`, `proxmoxLxcDiskSizeGb`, `proxmoxLxcRootFsStorageName`, `proxmoxLxcOsTemplate`, `host_vm_network`) podem ser definidos por host sob o nome do host.

```
dns:
  hosts:
    site-dns01:
      ansible_host: 192.168.1.20
      deployment_type: vm         # substituição por host do
padrão do site
      proxmoxLxcDiskSizeGb: 120   # substituição por host
      host_vm_network: vmbr1      # substituição de ponte por host
  vars:
    proxmox_interface: vmbr0
    gateway: 192.168.1.1
    mask_cidr: 24  # opcional, padrão 24
```

Playbooks de Serviço

Os playbooks de serviço implantam e configuram a infraestrutura do OmniCore. Localizados no diretório `services/`.

Hierarquia dos Playbooks

Os playbooks são estruturados hierarquicamente para permitir implantações rápidas e direcionadas:

```
all.yml
├── setup_users.yml
├── apt_cache.yml          # apenas quando o grupo
apt_cache_servers está definido
├── 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
```

Execute apenas o que você precisa. Implantar um único componente (por exemplo, `omnimme.yml`) leva segundos. Executar `all.yml` em 50 hosts leva mais tempo, mas gerencia tudo. Use `--limit` para restringir ainda mais o escopo.

Referência Rápida

Playbooks de Nível Superior

Playbook	Escopo	Descrição
`all.yml`	Rede completa	Implantação completa: usuários, DNS, monitoramento, testador de dispositivos, EPC, IMS, ePDG, TWAG, SS7, OCS, CRM, monitor de RAN, Core 5G e uma verificação de saúde final (do `util_playbooks/`)
`epc.yml`	Core de Pacotes	MME, SGW-C, PGW-C, UPF, HSS, DRA
`ims.yml`	Voz/IMS	P/I/S-CSCF, Servidor de Aplicação, Mensageria, OmniSEP (XCAP/Entitlements/BSF)
`ocs.yml`	Cobrança	CGrateS, cluster KeyDB, sincronização OCS
`monitoring.yml`	Observabilidade	Prometheus, Grafana, exporters

Playbooks de Infraestrutura

Playbook	Descrição
`common.yml`	Configuração base do SO, pacotes, NTP, agentes de logging
`setup_users.yml`	Contas de usuário local e chaves SSH
`dns.yml`	Implantação do servidor DNS
`license_server.yml`	Servidor de licença do OmniCore
`does_netplan_exist.yml`	Verifica se o netplan está instalado em cada host
`update_mtu.yml`	Define e aplica o MTU da interface via netplan
`firewall.yml`	Regras iptables/nftables
`apt_cache.yml`	Espelho APT local (se habilitado)

Componentes EPC

Playbook	Componente	Descrição
`omnimme.yml`	OmniMME	Entidade de Gerenciamento de Mobilidade (4G)
`omnisgwc.yml`	OmniSGW-C	Plano de Controle do Gateway de Serviço
`omnipgwc.yml`	OmniPGW-C	Plano de Controle do Gateway de PDN
`upf.yml` / `omniupf.yml`	OmniUPF	Função do Plano do Usuário (SGW-U/PGW-U combinados)
`omnihss.yml` / `hss.yml`	OmniHSS	Servidor de Assinante Residencial
`omnidra.yml`	OmniDRA	Agente de Roteamento Diameter
`omnitwag.yml`	OmniTWAG	Gateway de Acesso Sem Fio Confiável
`omniepdg.yml`	OmniEPDG	Gateway de Dados de Pacote Evoluído (Chamadas WiFi)
`gtp_proxy.yml`	GTP Proxy	Proxy de tráfego GTP

Componentes IMS

Playbook	Componente	Descrição
`pcscf.yml`	P-CSCF	Função de Controle de Sessão de Chamada Proxy
`icscf.yml`	I-CSCF	CSCF Interrogador
`scscf.yml`	S-CSCF	CSCF de Serviço
`as.yml`	OmniTAS	Servidor de Aplicação de Telefonia
`omnisep.yml`	OmniSEP	XCAP, Servidor de Direitos, BSF, Correio de Voz Visual
`xcap.yml`	OmniSEP	Alias legado de `omnisep.yml` (implanta o mesmo papel `omnisep`)
`omnimessage.yml`	OmniMessage / IMS SMSC	Controlador SMS sobre IP, SMPP, e o IMS SMSC (implantado via o papel `cscf` com `cscf_type: smsc_ims`)

Core 5G

Implantado via `5gc.yml`. Cada função de rede é instalada pelo papel compartilhado `5gc_nf` com um `nf_package` por NF. Veja Core 5G para os detalhes internos do papel, configuração por NF, fixação de versão e chaves UDM HNET/SUCI.

Playbook	Componente	Descrição
`5gc.yml`	Core 5G	Implanta todas as funções de rede 5G Standalone
	OmniNRF	Função de Repositório de Rede
	OmniAMF	Função de Gerenciamento de Acesso e Mobilidade
	OmniAUSF	Função do Servidor de Autenticação
	OmniBSF	Função de Suporte à Ligação
	OmniCHF	Função de Cobrança
	OmniNSSF	Função de Seleção de Fatiamento de Rede
	OmniPCF	Função de Controle de Políticas
	OmniSCP	Proxy de Comunicação de Serviço
	OmniSMF	Função de Gerenciamento de Sessão
	OmniUDM	Gerenciamento Unificado de Dados
	OmniUDR	Repositório Unificado de Dados

Comutação de Circuito

Implantado via `cs.yml` (papéis sob `circuit_switched/`). Veja Core de Comutação de Circuito para os detalhes dos papéis BSC, MSC e SS7.

Playbook	Componente	Descrição
`cs.yml`	Comutação de Circuito	Implanta o domínio CS legado
	BSC	Controlador de Estação Base (`circuit_switched/bsc`)
	OmniMSC	Centro de Comutação Móvel (`circuit_switched/omnimsc`)
	SS7	Pilha SS7/SIGTRAN (`omniss7` role)

Serviços de Suporte

Playbook	Descrição
`ocs.yml`	Sistema de Cobrança Online (CGrateS + KeyDB)
`crm.yml`	Portal de gerenciamento de clientes (OmniCRM; configuração agora vive em `group_vars/`)
`omniss7.yml`	Gateway SS7/SIGTRAN
`voipmonitor.yml`	Captura de pacotes SIP/RTP (VoIPmonitor) na caixa de captura `siptrace`, além de coletores thin-client opt-in
`grafana.yml`	Painéis e alertas do Grafana
`ran_monitor.yml`	Integração de monitoramento de RAN
`omniroam.yml`	Conjunto de roaming OmniRoam (TAP3/RAP UI, API Elixir, gerador)
`omnilcs.yml`	Serviços de Localização OmniLCS e Transmissão de Célula
`device_tester.yml`	Banco de testes do agente de telefone OmniWeb (Test Devices)

> **Provisionamento Proxmox** (`proxmox.yml`, `proxmox_delete.yml`) vive sob `util_playbooks/`. Veja Playbooks Utilitários e Implantação de VM/LXC Proxmox.

Operações

Playbook	Descrição
`backup.yml`	Backup de bancos de dados e configurações
`reboot.yml`	Reinicialização controlada de hosts
`shutdown.yml`	Desligamento gracioso
`apt_update.yml`	Atualizar pacotes
`apt_refresh_metadata.yml`	Atualizar metadados do cache APT
`speedtest.yml`	Teste de throughput de rede

Playbooks de Restauração

Playbook	Descrição
`restore_applicationserver.yml`	Restaurar OmniTAS a partir do backup
`restore_smsc.yml`	Restaurar SMSC a partir do backup: Controlador OmniMessage, gateway SMPP e nós IMS SMSc

> Nota: `health_check.yml`, `restore_hss.yml` e `restore_ocs.yml` **não** estão em `services/`. Eles vivem em `util_playbooks/`. Veja Playbooks Utilitários. `all.yml` executa a verificação de saúde como `../util_playbooks/health_check.yml`.

Uso

Implantar Tudo

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

Implantar Subsistema Específico

```
# Apenas o core de pacotes
ansible-playbook -i hosts/customer/host_files/production.yml
services/epc.yml

# Apenas IMS/voz
ansible-playbook -i hosts/customer/host_files/production.yml
services/ims.yml
```

Implantar Componente Único

```
# Atualizar apenas o MME
ansible-playbook -i hosts/customer/host_files/production.yml
services/omnimme.yml

# Atualizar apenas monitoramento
ansible-playbook -i hosts/customer/host_files/production.yml
services/monitoring.yml
```

Limitar a Hosts Específicos

```
# Executar all.yml, mas apenas em um host
ansible-playbook -i hosts/customer/host_files/production.yml
services/all.yml --limit mme01

# Executar em múltiplos hosts específicos
ansible-playbook -i hosts/customer/host_files/production.yml
services/all.yml --limit "mme01,hss01"

# Executar em um grupo
ansible-playbook -i hosts/customer/host_files/production.yml
services/all.yml --limit mme
```

Documentação Relacionada

- Arquitetura de Implantação - Fluxo de trabalho geral de implantação
- Configuração do Arquivo de Hosts - Definindo infraestrutura
- Configuração de Variáveis de Grupo - Personalização
- Playbooks Utilitários - Ferramentas operacionais (verificação de saúde, restauração, etc.)
- Monitoramento & Observabilidade - Grafana, Prometheus, alertas

Playbooks de Utilidade

Os playbooks de utilidade fornecem ferramentas operacionais para gerenciar a infraestrutura do OmniCore implantada. Esses playbooks estão localizados no diretório `util_playbooks/` e podem ser executados de forma independente para realizar tarefas comuns de manutenção e solução de problemas.

Referência Rápida

Playbook	Propósito
`proxmox.yml`	Provisionar VMs e/ou contêineres LXC no Proxmox (por `deployment_type`)
`proxmox_delete.yml`	Excluir VMs/LXCs no Proxmox
`proxmox_restart_hung.yml`	Forçar a reinicialização de VMs/LXCs via API do Proxmox quando o SSH não responde
`vmware.yml`	Provisionar VMs no VMware vSphere
`vmware_delete.yml`	Excluir VMs no VMware vSphere
`health_check.yml`	Gerar relatório de saúde abrangente para todos os serviços
`restore_hss.yml`	Restaurar o banco de dados HSS e/ou configuração a partir do backup
`seed_hss_replication.yml`	Semear um nó OmniHSS vazio a partir de um par populado (carregamento inicial de dados de replicação)
`audit_hss_drift.yml`	Auditar a malha OmniHSS para desvio de dados (contagem de linhas + resumo de conteúdo por tabela de referência em todos os nós); somente leitura
`reconcile_hss_drift.yml`	Reconciliar o desvio de dados da malha OmniHSS preenchendo as linhas ausentes entre os nós (execução a seco por padrão)

Playbook	Propósito
`restore_ocs.yml`	Restaurar OCS KeyDB, MySQL e configuração a partir do backup
`ip_plan_generator.yml`	Gerar documentação de rede com diagramas Mermaid
`get_ports.yml`	Auditar portas abertas e serviços em escuta em todos os hosts
`getLocalCapture.yml`	Recuperar arquivos de captura de pacotes dos hosts
`delete_local_user.yml`	Remover uma conta de usuário local de todos os hosts
`clear_mnesia.yml`	Limpar o banco de dados Mnesia para OmniSS7 / OmniMessage e reiniciar (destrutivo)
`delete_license.yml`	Excluir o arquivo de licença no servidor de licença e reiniciá-lo

Provisionamento de VM

Os playbooks de provisionamento de VM criam e gerenciam máquinas virtuais em plataformas de hipervisor. Todos os playbooks suportam `--limit` para direcionar hosts ou grupos específicos.

Veja Implantação do Proxmox para configuração detalhada.

Proxmox

Arquivos: `util_playbooks/proxmox.yml`, `util_playbooks/proxmox_delete.yml`

`proxmox.yml` provisiona tanto VMs quanto contêineres LXC; se um determinado host se torna uma VM ou um LXC é decidido pelo seu `deployment_type` resolvido (sobrescrição por host → entrada `proxmoxServers` → padrão `vm`). Veja Implantação do Proxmox.

```
# Provisionar (VMs e/ou contêineres LXC, por deployment_type)
ansible-playbook -i hosts/customer/host_files/production.yml
util_playbooks/proxmox.yml

# Provisionar apenas grupo específico
ansible-playbook -i hosts/customer/host_files/production.yml
util_playbooks/proxmox.yml --limit mme

# Excluir VMs/LXCs
ansible-playbook -i hosts/customer/host_files/production.yml
util_playbooks/proxmox_delete.yml --limit old-host
```

Reiniciar Hosts Proxmox Congelados

Arquivo: `util_playbooks/proxmox_restart_hung.yml`

Verifica a responsividade do SSH de cada host e força a reinicialização de qualquer VM ou contêiner LXC não responsivo através da API do Proxmox. VMs QEMU são paradas e iniciadas; contêineres LXC são iniciados (ou reiniciados se já estiverem em execução). Hosts que respondem ao SSH são ignorados.

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

# Restringir a um único host ou grupo
ansible-playbook -i hosts/customer/host_files/production.yml
util_playbooks/proxmox_restart_hung.yml --limit hss01
```

VMware vSphere

Arquivos: `util_playbooks/vmware.yml`, `util_playbooks/vmware_delete.yml`

Ambos os playbooks incluem os arquivos de tarefa por VM sob `util_playbooks/vm/vmware/` (`create_individual_vm.yml` e `delete_individual_vm.yml`). `vmware_delete.yml` é uma fina camada em torno da tarefa de exclusão; `vmware.yml` aguarda SSH após o provisionamento, remove arquivos de netplan obsoletos e aplica netplan na nova VM.

Pré-requisitos: Requer que as dependências do Python estejam instaladas (`uv sync`).

Uso:

```
# Provisionar VMs
ansible-playbook -i hosts/customer/host_files/production.yml
util_playbooks/vmware.yml

# Provisionar apenas grupo específico
ansible-playbook -i hosts/customer/host_files/production.yml
util_playbooks/vmware.yml --limit mme

# Excluir VMs
ansible-playbook -i hosts/customer/host_files/production.yml
util_playbooks/vmware_delete.yml --limit old-host
```

Variáveis Necessárias:

```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

Arquivos: `util_playbooks/vm/vultr/vmProvisionVultr.yml`, `util_playbooks/vm/vultr/vmTeardownVultr.yml`, `util_playbooks/vm/vultr/vmCommonSetup.yml`

Provisiona e desmonta instâncias no Vultr via API do Vultr. `vmProvisionVultr.yml` cria instâncias, `vmTeardownVultr.yml` as remove, e `vmCommonSetup.yml` realiza a configuração base pós-provisionamento (DNS, reinicialização, etc.). Todos os três são playbooks de ponto de entrada de nível superior, em vez de inclusões de tarefa. Eles leem o token da API da variável de ambiente `VULTR_API_TOKEN`, e o playbook de provisionamento faz referência a um `cloud-config.yaml` do mesmo diretório, portanto, execute-os a partir de `util_playbooks/vm/vultr/`.

```
export VULTR_API_TOKEN=<token>
ansible-playbook -i hosts/customer/host_files/production.yml
util_playbooks/vm/vultr/vmProvisionVultr.yml
```

Verificação de Saúde

Arquivo: `util_playbooks/health_check.yml`

Gera um relatório de saúde HTML abrangente cobrindo todos os serviços OmniCore e OmniCall implantados.

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

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

Informações Coletadas

Componente	Dados Coletados
Todos os serviços	Status do serviço, versão, tempo de atividade
OmniHSS	Status do banco de dados, conexões de pares Diameter
OmniDRA	Conexões e status de pares Diameter
OmniTAS	Chamadas ativas, sessões, registros, uso da CPU
OCS	Status de replicação do KeyDB

Restauração do HSS

Arquivo: `util_playbooks/restore_hss.yml`

Restaura o OmniHSS a partir de arquivos de backup. Suporta a restauração apenas do banco de dados, apenas da configuração ou de ambos.

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

Formatos de Arquivo de Backup

Tipo	Padrão de Nome de Arquivo	Conteúdos
Banco de Dados	`hss_dump_<hostname>_<timestamp>.sql`	Dump do banco de dados OmniHSS (PostgreSQL (pg_dump) do padrão `omnihss`)
Configuração	`hss_<hostname>_<timestamp>.tar.gz`	Arquivo do diretório `/etc/omnihss`

O estado atual do HSS (banco de dados e/ou configuração) é feito backup antes da execução da restauração.

Solicitações

Solicitação	Necessária	Descrição
Caminho do dump SQL do HSS	Não	Caminho completo para o arquivo de dump SQL do HSS (deixe em branco para pular a restauração do banco de dados)
Caminho do tar.gz de configuração do HSS	Não	Caminho completo para o arquivo tar.gz de configuração do HSS (deixe em branco para pular a restauração da configuração)
Confirmação	Sim	Digite `yes` para confirmar a operação de restauração nos host(s) listados

Semente de Replicação do HSS

Arquivo: `util_playbooks/seed_hss_replication.yml`

Realiza o carregamento inicial de dados uma única vez para um nó OmniHSS que está se juntando a uma malha de replicação ativa-ativa já populada. Um novo nó começa vazio (as assinaturas são criadas com `copy_data = false`); este playbook recria a assinatura para um único par fonte escolhido com `copy_data = true`, de modo que o PostgreSQL copie os dados existentes e continue a transmissão sem lacunas.

Ele é executado em um dos dois modos:

Interativo (padrão): omita as variáveis `seed_source_*`. O playbook detecta quais alvos limitados estão vazios. Ele então investiga os pares válidos em busca de aqueles que possuem dados, lista-os com contagens de assinantes e solicita que você escolha uma fonte. Ele semeia apenas alvos vazios.

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

```
Pares com dados disponíveis para semear:
  1) dev-hss01 (10.4.10.142) - 838 assinantes
Digite o NÚMERO da fonte para semear
```

Explícito: passe a fonte diretamente para pular o menu. Isso **força** uma nova semeadura mesmo que o alvo já possua dados (use para recuperação).

```
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"
```

Parâmetros

Parâmetro	Necessário	Padrão	Descriçã
`seed_source_name`	Não	(solicitado)	Nome do host inventário do fonte populac Usado para nomear a assinatura (`sub_from_<n` e resolver sua publicação (`<name>_pub`) Omitir para o modo interati
`seed_source_host`	Não	(solicitado)	Endereço IP d fonte, acessív em sua porta PostgreSQL. C para o modo interativo.
`seed_source_port`	Não	porta `omnihss.database_port` do alvo (5432)	Porta Postgre no par fonte.

> Tanto `seed_source_name` quanto `seed_source_host` devem ser fornecidos juntos para usar o modo explícito; não fornecer nenhum dos dois aciona o modo interativo.

Comportamento

1. Conta assinantes em cada alvo limitado para determinar quais estão vazios.

2. **Modo interativo:** investiga os pares válidos (grupo `hss` local + `connected_omnihss`) em busca de aqueles que possuem dados, lista-os e solicita uma escolha. **Modo explícito:** solicita `yes` para confirmar.
3. Garante que o IP da fonte seja permitido no `pg_hba.conf` do alvo.
4. Remove e recria a assinatura para a fonte `COM (copy_data = false)` (por link `slot_name` de `<target>_from_<source>`). Ele transmite **todas** as tabelas, mas não realiza nenhuma sincronização em massa de tabelas.
5. Preenche as linhas existentes da **tabela mestre** a partir da fonte com um carregamento seguro contra conflitos (`pg_dump --data-only --inserts --on-conflict-do-nothing --disable-triggers`). O `subscriber_state` **é** incluído: cada assinante referencia um por FK, então ignorá-lo orfana cada assinante preenchido. As tabelas de sessão transitórias (`pdn_session`, `lte_call`) **não** são preenchidas.
6. Relata a contagem de assinantes no nó semeado.

> **Por que as tabelas de sessão são ignoradas:** elas são tabelas quentes; um `COPY` em massa delas em um sistema ativo colide (chave duplicada) com gravações concorrentes e causa loops de falha no trabalhador de sincronização de tabelas. Elas se repopulam a partir da replicação em streaming e tráfego ativo. O `subscriber_state` é diferente: as linhas são apenas INSERTadas no momento do provisionamento (1:1 com o assinante) e apenas atualizadas posteriormente, então o preenchimento `--on-conflict-do-nothing` é seguro. Sem isso, os assinantes do alvo apontam para linhas de estado que nunca chegam, uma vez que a replicação lógica silenciosamente descarta UPDATEs para linhas ausentes.

> **Importante:** Sempre `--limit` para o(s) nó(s) alvo. No modo interativo, alvos não vazios são ignorados automaticamente. Veja Replicação e Operações de Nó do OmniHSS para o fluxo de trabalho completo.

Auditoria de Desvio do HSS

Arquivos: `util_playbooks/audit_hss_drift.yml` (independente) e `roles/omnihss/tasks/audit_drift.yml` (a implementação compartilhada).

Relata **desvio de dados** na malha ativa-ativa do OmniHSS. Assinaturas de replicação lógica são criadas com `copy_data = false`, então dados que divergiram antes do link da malha (diferenças de tempo de semeadura) ou via uma gravação direta/não replicada nunca são reconciliados e nunca alarmados. A replicação mostra 0 de atraso enquanto os dados diferem, e o Prometheus por site não consegue vê-lo porque a malha abrange dois sites.

A auditoria se conecta a cada membro da malha (grupo `hss` local + pares `connected_omnihss`) e, para cada tabela de referência/configuração, compara uma contagem exata de linhas **e** um md5 independente de ordem de todo o conteúdo. Ela também compara `schema_migrations` (contagem + versão máxima) para capturar desvio de esquema. Tabelas de estado quentes (`subscriber_state`, `pdn_session`, `lte_call`) são excluídas: elas diferem legitimamente sob tráfego ativo. Um **check de saúde** estrutural é incluído além da comparação entre nós: `__orphaned_subscribers__`, a contagem por nó de assinantes cuja linha `subscriber_state` está ausente. Uma contagem diferente de zero em **qualquer** nó falha a auditoria, mesmo quando todos os nós concordam. Uma semeadura que copia `subscriber` sem `subscriber_state` orfana todos os nós de forma idêntica, o que uma comparação pura entre nós nunca consegue ver. A comparação é centralizada em um executor que alcança o Postgres de cada par, então uma única execução cobre toda a malha entre sites a partir de qualquer inventário, e é segura para `--limit`.

Executa automaticamente após cada implantação do HSS. No final do papel `omnihss` (quando a replicação é habilitada), a mesma auditoria é executada como uma verificação de consistência pós-implantação e, por padrão, **FALHA o play** se desvio for encontrado, então uma malha quebrada é impossível de ser ignorada durante uma implantação. É não interativa (sem pausa): a auditoria deve ser segura sob a estratégia `free` da implantação agregada `services/all.yml`, onde uma pausa geraria erro. Nessa implantação agregada, a auditoria dentro do papel é suprimida e a auditoria é executada uma vez em um play **linear** dedicado no final de `services/deploy_nfs.yml`. O comportamento é controlado por duas variáveis de papel:

Variável	Padrão	Efeito
`omnihss_audit_drift`	`true`	executar a auditoria no final do papel
`omnihss_audit_fail_on_drift`	`true`	falhar o play em caso de desvio; definir `false` para relatório apenas (cron/CI não interativo)

A auditoria é **somente leitura** de qualquer forma. O playbook independente por padrão relata- então **falha** em caso de desvio (para que possa controlar cron/CI):

```
# sob demanda
ansible-playbook -i hosts/<inv>/host_files/<site>.yml
util_playbooks/audit_hss_drift.yml
# relatório apenas, sempre passa
ansible-playbook -i ... util_playbooks/audit_hss_drift.yml -e
audit_fail_on_drift=false
# desabilitar a auditoria automática para uma única implantação
ansible-playbook -i ... services/omnihss.yml -e
omnihss_audit_drift=false
```

Reconciliação de Desvio do HSS

Arquivo: `util_playbooks/reconcile_hss_drift.yml`

Trás tabelas de referência divergentes de volta à sincronização, fazendo com que cada nó mantenha a **união** das linhas: para cada tabela desviada, ele preenche as linhas que um nó está faltando de seus pares, usando o mesmo primitivo seguro contra conflitos que o playbook de semeadura (`pg_dump --data-only --inserts --on-conflict-do-nothing --disable-triggers`).

- **Não destrutivo**: as linhas são sempre apenas INSERTadas (ignorando conflitos). Nada é excluído ou atualizado. Ele resolve "um nó está faltando

linhas"; não resolve "mesma chave primária, valores diferentes" ou remove linhas que não deveriam existir. Essas precisam de uma decisão humana.

- **Interativo por padrão**: ele escaneia, imprime o plano e **pergunta** (`yes` para prosseguir) antes de escrever; qualquer outra coisa aborta sem alterações. Use `-e dry_run=true` para olhar apenas, ou `-e assume_yes=true` para execuções não interativas.
- As gravações são executadas localmente em cada alvo como superusuário do postgres (para `--disable-triggers`); pares são apenas lidos, como o usuário do aplicativo.
- **União simétrica, não autoritária de produção**: executá-lo contra um site puxa *as* linhas desse site para os outros. Ele não trata a produção como a mestre, então executá-lo contra o inventário do Lab propagaria linhas do Lab em direção à produção. Se a produção deve ser a fonte da verdade, dirija-o a partir do inventário de produção (ou solicite o modo autoritário).
- **Entre sites:** um nó só pode ser um *alvo* quando está no play, então execute isso contra **cada** inventário de site para cobrir todos os nós. O re-escanear no final relata qualquer coisa que ainda precise de execução.

```
# interativo (padrão): pergunta antes de escrever
ansible-playbook -i hosts/<inv>/host_files/<site>.yml
util_playbooks/reconcile_hss_drift.yml
# olhar apenas, nunca escreve
ansible-playbook -i ... util_playbooks/reconcile_hss_drift.yml -e
dry_run=true
# aplicação não interativa (CI)
ansible-playbook -i ... util_playbooks/reconcile_hss_drift.yml -e
assume_yes=true
```

> Execute `audit_hss_drift.yml` primeiro para ver o que desviou, e novamente após reconciliar (contra ambos os inventários) para confirmar a convergência. Veja Replicação e Operações de Nó do OmniHSS.

Restauração do OCS

Arquivo: `util_playbooks/restore_ocs.yml`

Restaura o OCS (CGrateS) a partir de arquivos de backup. Lida com replicação multi-mestre do KeyDB restaurando para um nó e permitindo que a replicação sincronize com outros.

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

Processo de Restauração

1. Para CGrateS e KeyDB em todos os nós do OCS
2. Limpa arquivos AOF para evitar conflitos com dados restaurados
3. Restaura o RDB do KeyDB para o primeiro nó, inicia o KeyDB, permite que a replicação sincronize
4. Restaura o StoreDB do MySQL (opcional)
5. Restaura a configuração de `/etc/cgrates` (opcional)
6. Inicia o CGrateS em todos os nós

Formatos de Arquivo de Backup

Tipo	Padrão de Nome de Arquivo	Conteúdos
Banco de Dados KeyDB	`keydb_dump_<hostname>_<timestamp>.rdb`	Snapshot RDB do KeyDB
StoreDB MySQL	`cgrates_dump_<hostname>_<timestamp>.sql`	Dump MySQL do banco de dados `cgrates`
Configuração	`cgrates_<hostname>_<timestamp>.tar.gz`	Arquivo do diretório `/etc/cgrates`

Solicitações

Solicitação	Necessária	Descrição
Caminho do dump RDB do KeyDB	Sim	Caminho completo para o arquivo de backup RDB do KeyDB
Caminho do dump SQL do MySQL	Não	Caminho completo para o dump SQL do CGrateS (pule para preservar o StoreDB atual)
Caminho do tar.gz de configuração	Não	Caminho completo para o backup de configuração (pule para preservar a configuração atual)

Gerador de Plano de IP

Arquivo: `util_playbooks/ip_plan_generator.yml`

Gera documentação de rede a partir do inventário, incluindo:

- Atribuições de IP de host (NICs primárias e secundárias)
- Visão geral do segmento de rede
- Diagramas de conectividade de interface (Diameter, GTP, PFCP, SIP, SS7)

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

Arquivos de Saída

Arquivo	Formato	Descrição
`/tmp/ip_plan_<customer>_<site>.md`	Markdown	Documentação baseada em texto
`/tmp/ip_plan_<customer>_<site>.html`	HTML	Diagrama interativo com camadas filtráveis

Auditoria de Portas

Arquivo: `util_playbooks/get_ports.yml`

Audita todas as portas em escuta em toda a implantação e gera documentação.

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

Arquivos de Saída

Arquivo	Descrição
`/tmp/all_ports.csv`	CSV com nome do host, IP, protocolo, porta, serviço
`./open_ports.rst`	tabela reStructuredText para documentação Sphinx

Dados Coletados

Campo	Descrição
Nome do Host	Nome do host do inventário
IP	Endereço IP `ansible_host` do host
Versão do IP	IPv4 ou IPv6
Transporte	TCP ou UDP
Porta	Número da porta em escuta
Serviço	Nome do processo

Recuperação de Captura Local

Arquivo: `util_playbooks/getLocalCapture.yml`

Recupera os dois arquivos de captura de pacotes mais recentes do diretório `/etc/localcapture` de cada host.

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

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

Gerenciamento de Usuários

Arquivo: `util_playbooks/delete_local_user.yml`

Remove uma conta de usuário local de todos os hosts no inventário.

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

Solicitação: Digite o nome de usuário a ser excluído quando solicitado.

Limpar Banco de Dados Mnesia

Arquivo: `util_playbooks/clear_mnesia.yml`

Para o serviço, exclui seu diretório de banco de dados Mnesia e o reinicia. Executa contra o grupo `ss7` (OmniSS7) e o grupo `omnimessage` (OmniMessage).

> **Aviso:** Isso é destrutivo. O banco de dados Mnesia é excluído permanentemente e reconstruído vazio na reinicialização.

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

# Limitar a um único serviço ou host
ansible-playbook -i hosts/customer/host_files/production.yml
util_playbooks/clear_mnesia.yml --limit ss7
```

Excluir Licença

Arquivo: `util_playbooks/delete_license.yml`

Exclui `/etc/license_server/license.json` no grupo `license_server` e reinicia o servidor de licença.

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

Executando Playbooks de Utilidade

Sintaxe Básica

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

Opções Comuns

Opção	Descrição
`-i <inventory>`	Especificar arquivo de inventário
`--limit <hosts>`	Limitar a hosts ou grupos específicos
`-v` / `-vv` / `-vvv`	Aumentar a verbosidade
`--check`	Execução a seco (mostrar o que mudaria)
`--diff`	Mostrar diferenças de arquivo

Exemplos

```
# Executar verificação de saúde na produção
ansible-playbook -i hosts/acme/host_files/production.yml
util_playbooks/health_check.yml

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

# Gerar plano de IP com saída detalhada
ansible-playbook -i hosts/acme/host_files/production.yml
util_playbooks/ip_plan_generator.yml -v
```