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 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 de UE** - Quarta /24

Importante: Estas são Recomendações, Não Requisitos

A alocação de sub-rede descrita neste documento é uma **melhor 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 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.

Desagregação do Segmento 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`

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

---

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)
- OminiHSS PCRF (Função de Regras de Política e Cobrança)
- OmniDRA DRA (Agente de Roteamento Diameter)
- Servidores DNS
- Servidores TAP3/CDR
- Monitoramento/OAM
- Captura de SIP
- Servidor de Licenças
- Monitor de RAN
- Omnitouch Warning Link CBC (Centro de Transmissão de Células) - se implantado
- Servidores de Cache APT - se implantado

Exemplo: `10.179.2.0/24`

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

---

3. 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 Servidora)
- OmniCSCF I-CSCF (Função de Controle de Sessão de Chamada Interrogante)
- OmniTAS (Servidor de Aplicação de Telefonia / Servidor de Aplicação)
- OmniMessage (Controlador de SMS, SMPP, IMS)
- 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

Exemplo: `10.179.3.0/24`

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

---

4. Rede Pública de UE (Quarta /24)

Propósito: Serviços voltados para o usuário, como IMS e DNS

Componentes:

- OmniCSCF P-CSCF (Função de Controle de Sessão de Chamada Proxy)
- Servidores XCAP
- Servidores de Correio de Voz Visual
- DNS do Cliente

Exemplo: `10.179.4.0/24`

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

Métodos de 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 fornece o isolamento mais forte e é a abordagem recomendada para implantações em produção.

Exemplo:

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

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

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

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

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 usam vmbr12 com diferentes VLANs
applicationserver:
  hosts:
    ons-lab08sbc01:
      ansible_host: 10.178.2.213
      gateway: 10.178.2.1
      host_vm_network: "ovsbr1"
      vlanid: "402"

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

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

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 10: 10.x.1.0/24 (Núcleo de Pacotes)
VLAN 20: 10.x.2.0/24 (Sinalização)
VLAN 30: 10.x.3.0/24 (IMS Interno)
VLAN 40: 10.x.4.0/24 (UE 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
    opt-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
    opt-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
    opt-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:
    opt-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:
    opt-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

É comum ter uma mistura de IPs internos e públicos dentro do mesmo grupo de componentes. 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 Desdobramento do Ansible na Omnitouch

Visão Geral

Os Serviços de Rede Omnitouch usam 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.

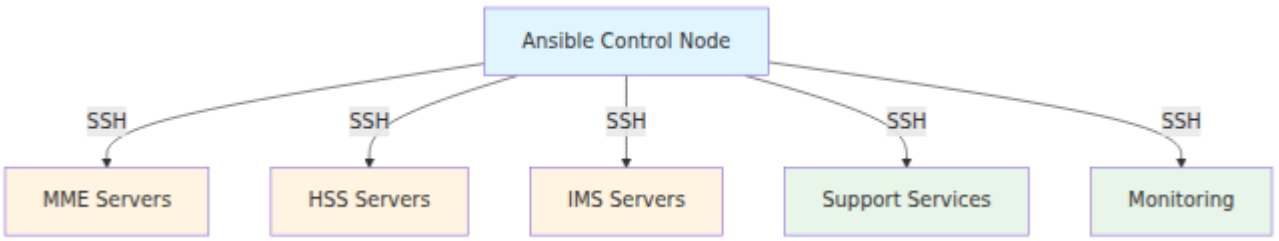
O que é Ansible?

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

- Configurar sistemas
- Implantar software
- Orquestrar fluxos de trabalho complexos
- Gerenciar infraestrutura como código

O Ansible usa 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.

Estas são definidas pela equipe ONS, enquanto você pode editá-las, geralmente há maneiras mais limpas de fazer quaisquer ajustes que você possa precisar a partir do seu arquivo de hosts.

3. Playbooks

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

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

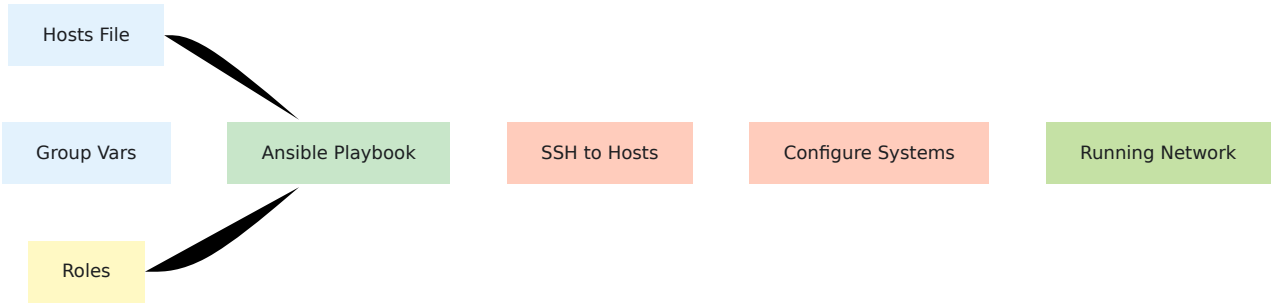
Usamos esses essencialmente como grupos para as funções.

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 do 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`:

```
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 aplicativo
- Configurar serviços com configurações do cliente
- Iniciar serviços
- Validar implantação

Componentes Chave que Implantamos

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

- **OmniHSS** - Servidor de Assinante Residencial
- **OmniSGW** - Gateway de Serviço (plano de controle)
- **OmniPGW** - Gateway de Pacote (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 de distribuição de pacotes híbrido:

Pacotes APT Pré-compilados

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

- Construído a partir do código-fonte em nosso pipeline CI/CD
- Versionado e testado
- Hospedado 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

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:

- Rastrear todas as alterações
- Revisar antes da implantação
- Reverter 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 configurar um ambiente virtual Python e instalar as dependências necessárias.

1. Criar um Ambiente Virtual Python

Crie um ambiente Python isolado para a implantação do Ansible:

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

2. Ativar o Ambiente Virtual

Ative o ambiente virtual:

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

No Windows, use:

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

3. Instalar Pacotes Necessários

Instale todas as dependências do arquivo requirements.txt:

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

Isso instalará o Ansible e todos os pacotes Python necessários para a automação de implantação da Omnitouch.

Nota: Mantenha o ambiente virtual ativado sempre que executar comandos do Ansible. Você pode desativá-lo 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. Implante!

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 Utilitários - Ferramentas operacionais para verificações de saúde, backups e manutenção

Repositório APT e Distribuição de Pacotes

Visão Geral

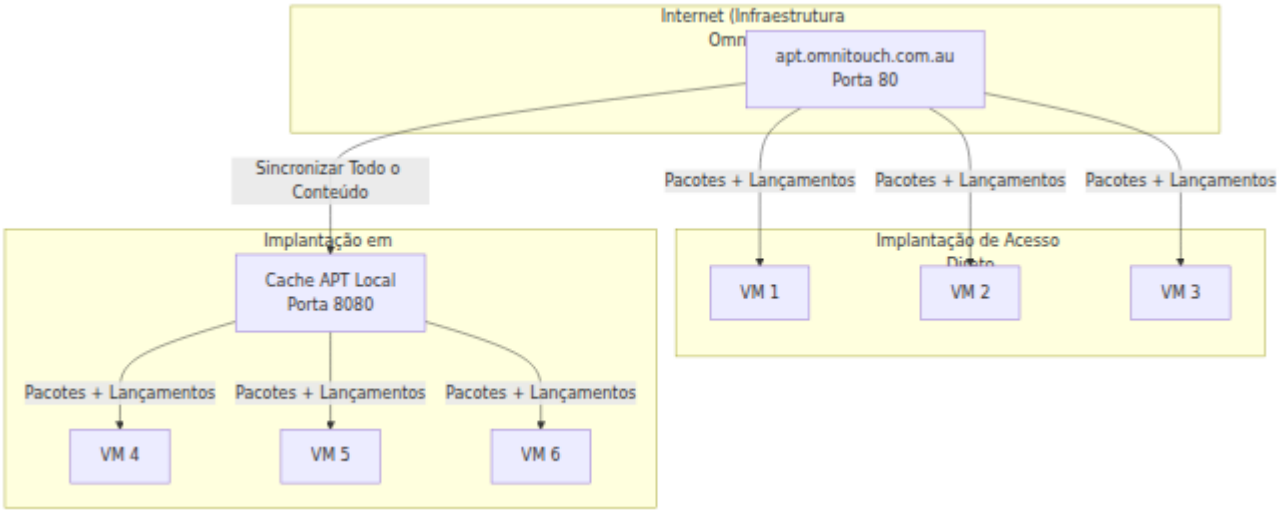
O sistema APT da Omnitouch fornece distribuição de pacotes para todas as implantações. Dois tipos de conteúdo são servidos:

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

Dois modelos de implantação são suportados:

1. **Acesso Direto** — VMs puxam pacotes diretamente de `apt.omnitouch.com.au`
2. **Espelho de Cache Local** — Um servidor local sincroniza com a 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>/`
Pacotes Ubuntu	Pacotes Ubuntu em cache com todas as dependências	`/<distro>/pool/main/`
Lançamentos do GitHub	Binários pré-compilados (Prometheus, Grafana, Homer, 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

Dois conjuntos de variáveis separadas controlam a distribuição de pacotes. Compreender seus propósitos é essencial para uma configuração correta.

Propósitos das Variáveis

Conjunto de Variáveis	Propósito	Usado Para
`apt_repo`	Configura fontes de pacotes APT	`/etc/apt/sources.list` e `/etc/apt/sources.list.d/*.list`
`remote_apt_*`	Configura URLs de download binário	Baixando arquivos do caminho `/releases/` (Node Exporter, Zabbix, Nagios, etc.)

Quando Cada Conjunto de Variáveis É Usado

Cenário	Fontes APT (`apt_repo`)	Downloads Binários (`remote_apt_*`)
`use_apt_cache: true`	Usa `apt_repo.apt_server`	Usa `apt_repo.apt_server`
`use_apt_cache: false`	Usa `apt_repo.*` com credenciais	Usa `remote_apt_*` com credenciais

Quando `use_apt_cache: false`, ambos os conjuntos de variáveis são necessários.

Opção 1: Acesso Direto

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

Requisitos de Rede

Whitelist de IP de Origem: Seu endereço IP público deve ser incluído na lista de permissões no servidor APT da Omnitouch. Durante a configuração, forneça suas sub-redes de origem para a Omnitouch. Em troca, você receberá:

- **Nome de usuário** e **senha** para Autenticação Básica HTTP
- **FQDN** para o servidor APT

Requisitos de Firewall: O acesso de saída aos seguintes intervalos de IP da 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 da Omnitouch:

Serviço	Porta	Protocolo	Propósito
Servidor APT	80	TCP	Downloads de pacotes
Servidor APT	53	TCP/UDP	Resolução DNS para `apt.omnitouch.com.au`
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 HTTP (TCP/80), NTP (UDP/123) e DNS (TCP+UDP/53) seja permitido para os intervalos de IP da Omnitouch.

Configuração

```yaml
all:
  vars:
    use_apt_cache: false

    # Configuração das fontes de pacotes APT
    # Configura /etc/apt/sources.list para comandos apt install
    apt_repo:
      apt_server: "apt.omnitouch.com.au"
      apt_repo_username: "seu-usuario"
      apt_repo_password: "sua-senha"

    # Configuração de downloads binários
    # Usado para baixar arquivos do caminho /releases/
    remote_apt_server: "apt.omnitouch.com.au"
    remote_apt_port: 80
    remote_apt_protocol: "http"
    remote_apt_user: "seu-usuario"
    remote_apt_password: "sua-senha"
```

Parâmetros

Fontes de Pacotes APT (`apt_repo`)

Parâmetro	Tipo	Requerido	Padrão	Descriçã
`apt_repo.apt_server`	String	Sim	-	Nome do host ou endereço I do servido APT
`apt_repo.apt_repo_username`	String	Sim	-	Nome de usuário da Autenticaç Básica HT para fonte APT
`apt_repo.apt_repo_password`	String	Sim	-	Senha da Autenticaç Básica HT para fonte APT

Downloads Binários (`remote_apt_*`)

Parâmetro	Tipo	Requerido	Padrão	Descrição
`remote_apt_server`	String	Sim	-	Nome do host ou IP do servidor para downloads binários
`remote_apt_port`	Inteiro	Não	`80`	Porta do servidor para downloads binários
`remote_apt_protocol`	String	Não	`http`	Protocolo (`http` ou `https`)
`remote_apt_user`	String	Sim	-	Nome de usuário da Autenticação Básica HTTP para downloads
`remote_apt_password`	String	Sim	-	Senha da Autenticação Básica HTTP para downloads

Geral

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

Padrões de URL (Acesso Direto)

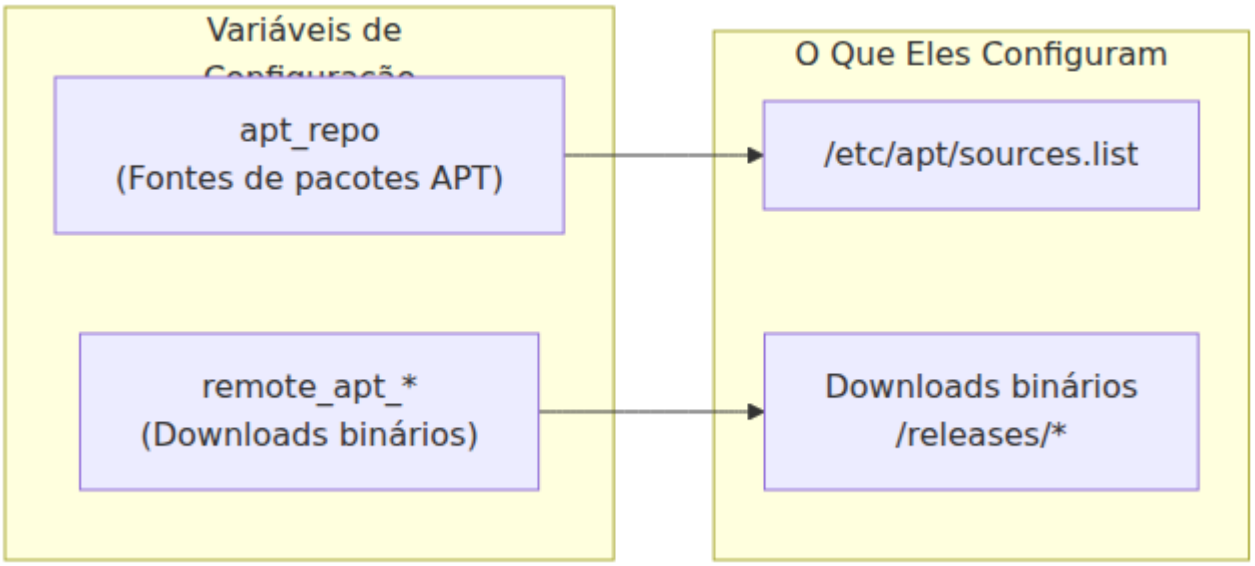
Fontes de Pacotes APT (configuradas em `/etc/apt/sources.list`):

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

Downloads Binários (usados por tarefas Ansible `get_url`):

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

Como Funciona



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

Opção 2: Espelho de Cache Local

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

Arquitetura

Configuração

Defina o servidor de cache em seu arquivo de hosts com sua configuração de repositório:

```yaml
apt_cache_servers:
  hosts:
    customer-apt-cache:
      ansible_host: 192.168.1.100
      gateway: 192.168.1.1
  vars:
    # O servidor de cache sincroniza pacotes do repositório
autenticado
    remote_apt_server: "apt.omnitouch.com.au"
    remote_apt_port: 80
    remote_apt_protocol: "http"
    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 pacotes de `apt.omnitouch.com.au:80`
- **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 Pacotes APT (`apt_repo`)

Parâmetro	Tipo	Requerido	Padrão	Desc
`apt_repo.apt_server`	String	Sim	Auto-derivado	IP do serv cache loc Derivado automatic do primei `apt_cach` se não especifica
`apt_repo.apt_repo_username`	String	Não	-	Não nece usar cach autentica necessári
`apt_repo.apt_repo_password`	String	Não	-	Não nece usar cach autentica necessári

Sincronização do Servidor de Cache (`remote_apt_*`)

Essas variáveis configuram como o servidor de cache sincroniza conteúdo da Omnitouch:

Parâmetro	Tipo	Requerido	Padrão	Descrição
`remote_apt_server`	String	Sim	-	Servidor APT da Omnitouch para sincronização
`remote_apt_port`	Inteiro	Não	`80`	Porta do servidor APT da Omnitouch
`remote_apt_protocol`	String	Não	`http`	Protocolo para conexão de sincronização
`remote_apt_user`	String	Sim	-	Credenciais para sincronização da Omnitouch
`remote_apt_password`	String	Sim	-	Credenciais para sincronização da Omnitouch

Geral

Parâmetro	Tipo	Requerido	Padrão	Descrição
`use_apt_cache`	Booleano	Não	`true`	Definido automaticamente como `true` quando o grupo `apt_cache_servers` existe
`apt_cache_port`	Inteiro	Não	`8080`	Porta que o servidor de cache local escuta

Padrões de URL (Modo Cache)

Fontes de Pacotes APT (configuradas em `/etc/apt/sources.list`):

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

Downloads Binários (usados por tarefas Ansible `get_url`):

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

Nenhuma credencial é necessária para acesso ao cache—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

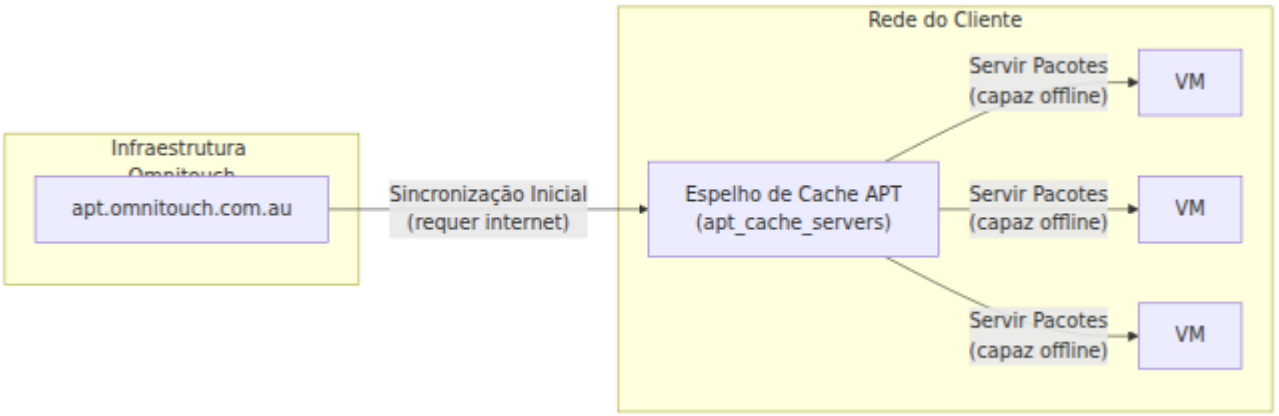
O espelho de cache sincroniza **todo o conteúdo** do servidor APT da Omnitouch usando download recursivo com wget:

Diretórios de conteúdo sincronizados:

Caminho	Conteúdo
`/dists/<distro>/`	Metadados do repositório APT (Pacotes, arquivos de Lançamento)
`/pool/main/`	Pacotes .deb personalizados da 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 usa `wget --recursive` com Autenticação Básica HTTP para baixar todo o conteúdo do servidor APT da Omnitouch. Sincronizações subsequentes baixam apenas arquivos novos/alterados (com base em timestamp).

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 sobrescrito)
2. Deriva `apt_repo.apt_server` do IP `ansible_host` do primeiro servidor de cache

Exemplo de Configuração Mínima

```yaml
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
    remote_apt_server: "apt.omnitouch.com.au"
    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 `http://seu-usuario:sua-senha@apt.omnitouch.com.au/`

Sobrescrever Comportamento Automático

Para forçar o 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.omnitouch.com.au"
      apt_repo_username: "usuario"
      apt_repo_password: "senha"

    remote_apt_server: "apt.omnitouch.com.au"
    remote_apt_user: "usuario"
    remote_apt_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.

```yaml
all:
  vars:
    use_apt_cache: false

    # Fontes de pacotes APT - usadas por todos os hosts
    apt_repo:
      apt_server: "apt.omnitouch.com.au"
      apt_repo_username: "usuario"
      apt_repo_password: "senha"

    # Downloads binários - usados por todos os hosts
    remote_apt_server: "apt.omnitouch.com.au"
    remote_apt_port: 80
    remote_apt_protocol: "http"
    remote_apt_user: "usuario"
    remote_apt_password: "senha"
```

Resultado: Todos os hosts geram `deb [trusted=yes] http://usuario:senha@apt.omnitouch.com.au/ 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.

```
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
    remote_apt_server: "apt.omnitouch.com.au"
    remote_apt_port: 80
    remote_apt_protocol: "http"
    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 `http://usuario:senha@apt.omnitouch.com.au:80/`
- **Todos os outros hosts**: Geram `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).

```
all:
  vars:
    use_apt_cache: true

    # Apontar 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 funciona
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

Aqui está um exemplo completo de trabalho mostrando a configuração do servidor de cache com múltiplos hosts de aplicativo:

```yaml
# Grupo do Servidor de Cache APT
apt_cache_servers:
  hosts:
    customer-apt-cache:
      ansible_host: 10.179.1.114
      gateway: 10.179.1.1
      host_vm_network: "vmbr0"
      num_cpus: 4
      memory_mb: 16384
      proxmoxLxcDiskSizeGb: 120
  vars:
    # O servidor de cache sincroniza pacotes do repositório
autenticado
    remote_apt_server: "apt.omnitouch.com.au"
    remote_apt_port: 80
    remote_apt_protocol: "http"
    remote_apt_user: "usuario-cliente"
    remote_apt_password: "token-seguro-cliente"

# 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_*` de sua seção `vars:`
   - Baixa todos os pacotes de `http://usuario-cliente:token-seguro-cliente@apt.omnitouch.com.au:80/`
   - Serve pacotes na porta 8080 via nginx
2. **Hosts de Aplicação (`customer-hss01`, `customer-mme01`, `customer-dns01`):**
   - Detectam automaticamente que o grupo `apt_cache_servers` existe
   - Definem automaticamente `use_apt_cache: true`
   - Derivam automaticamente `apt_repo.apt_server: "10.179.1.114"`
   - Geram: `deb [trusted=yes] http://10.179.1.114:8080/noble noble main`
   - Puxam 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 da Omnitouch:

- Novas versões de pacotes da Omnitouch

- Novos pacotes Ubuntu e dependências
- Novos lançamentos do GitHub
- Tarballs de fonte atualizados

A sincronização usa `wget --timestamping`, portanto, arquivos existentes não alterados são pulados, tornando a re-sincronização rápida.

Nota: O servidor APT da Omnitouch (`apt.omnitouch.com.au`) é a única fonte de verdade para todos os pacotes. Execute `services/apt.yml` no servidor apt primeiro para construir/atualizar pacotes, em seguida, execute `services/apt_cache.yml` nos espelhos de cache para sincronizar.

Solução de Problemas

Atualização APT Falha com 401 Não Autorizado

Sintomas:

```
Falha ao buscar
http://10.179.1.115:80/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 o repositório autenticado diretamente em vez do cache
- `apt_repo_username` ou `apt_repo_password` incorretos
- IP de origem não incluído na lista de permissões no servidor APT da 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 esteja 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 (porta 80)
3. **Verifique as fontes geradas**: No host com falha, verifique `/etc/apt/sources.list.d/omnitouch.list`
   - **Correto (modo cache)**: `deb [trusted=yes] http://10.179.1.114:8080/noble noble main`
   - **Incorreto (tem credenciais no lugar errado)**: `deb [trusted=yes] http://usuario:senha@10.179.1.115:80/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 a Omnitouch (se estiver usando acesso direto)

Downloads Binários Falham (Node Exporter, Zabbix, etc.)

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

Possíveis causas:

- Variáveis `remote_apt_*` não configuradas
- `remote_apt_user` ou `remote_apt_password` incorretos
- `remote_apt_server` ausente quando `use_apt_cache: false`

Resolução:

1. Certifique-se de que todas as variáveis `remote_apt_*` estejam definidas
2. Verifique se as credenciais correspondem às fornecidas pela Omnitouch
3. Verifique se `remote_apt_server` aponta para o host correto

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 a Omnitouch

Resolução:

1. Verifique se o servidor de cache pode acessar `apt.omnitouch.com.au` na porta 80
2. Verifique as credenciais `remote_apt_*`
3. Revise as regras do 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

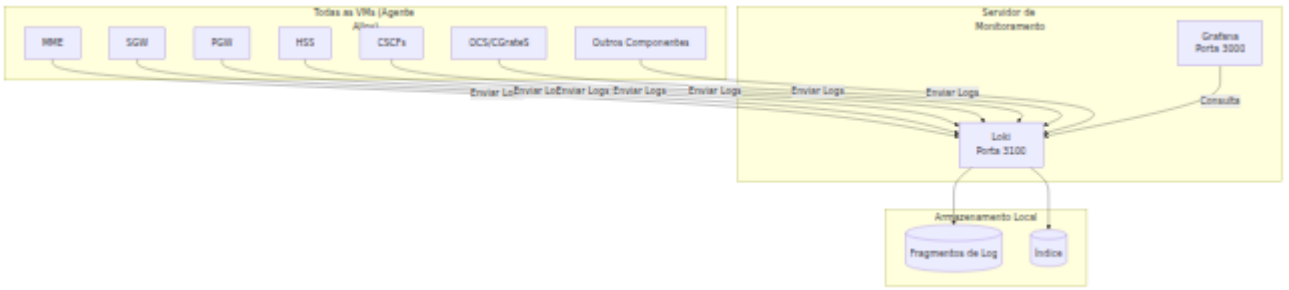
Registro Centralizado

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:

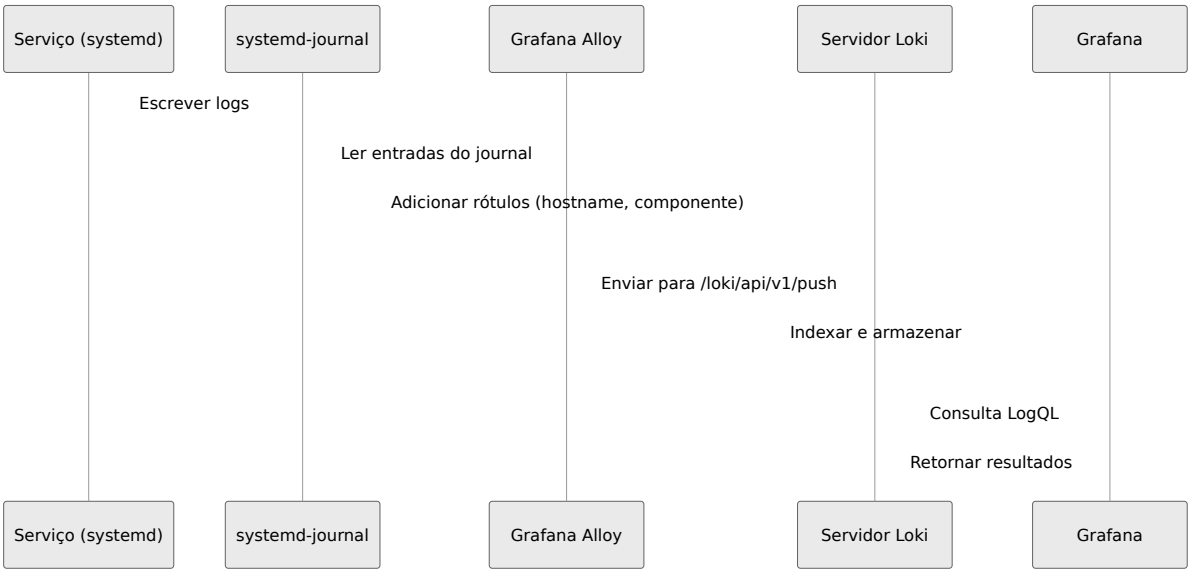
- **Grafana Alloy** — Agente de coleta de logs implantado em todas as VMs
- **Grafana Loki** — Servidor de agregação e armazenamento de logs em hosts de monitoramento
- **Grafana Dashboards** — Dashboards pré-construídos para cada tipo de componente

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`, `cscf`, `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 (recebem e armazenam logs)
- **Todos os outros hosts**: Executam o agente Alloy (coletam e enviam logs)

Configuração de Retenção

O Loki é configurado com limites de retenção dupla:

Configuração	Padrão	Descrição
Retenção baseada em tempo	7 dias	Logs mais antigos que 7 dias são excluídos
Retenção baseada em tamanho	50 GB	Logs mais antigos excluídos quando o armazenamento excede 50 GB

Qualquer limite alcançado primeiro aciona a exclusão de logs.

Para personalizar a retenção, sobrescreva estas variáveis em seu inventário:

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

Configuração do Buffer Alloy

O Alloy armazena logs localmente quando o Loki está inacessível. O buffer é limitado para evitar a exaustão do disco:

Configuração	Padrão	Descrição
Tamanho máximo do WAL	500 MB	Espaço máximo em disco para logs armazenados
Idade máxima do segmento do WAL	1 hora	Logs armazenados mais antigos são descartados após 1 hora

Quando o buffer enche, os logs mais antigos são descartados para abrir espaço para novas entradas.

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 portadora do SGW
Logs do PGW	Logs → Logs do PGW	Eventos de PDN e sessão do PGW
Logs do HSS	Logs → Logs do HSS	Mensagens Diameter do HSS e eventos de autenticação
Logs do OmniMessage	Logs → Logs do OmniMessage	Entrega de SMS e eventos 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

Recursos do Dashboard

Cada dashboard inclui:

- **Caixa de pesquisa** — Pesquisa de texto livre em todos os logs
- **Gráficos de taxa de log** — Volume ao longo do tempo por host e severidade

- **Seções de componente** — Visões agrupadas (por exemplo, P-CSCF, S-CSCF, I-CSCF)
- **Filtragem de erros** — Visões 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="cscf"} |~ "(?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]))
```

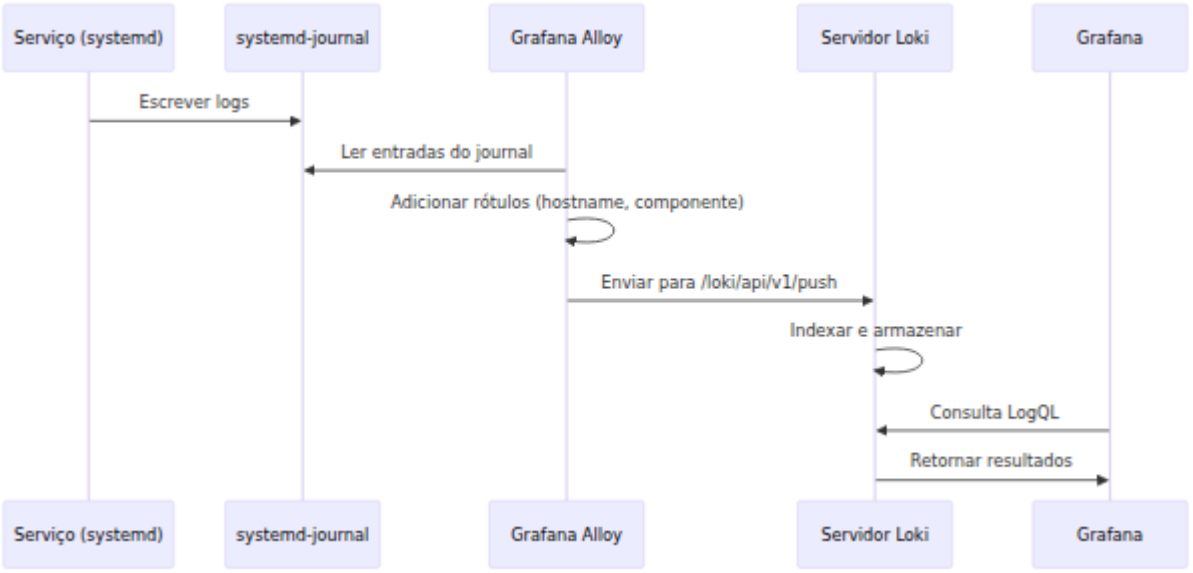
Taxa de erro para CSCF:

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

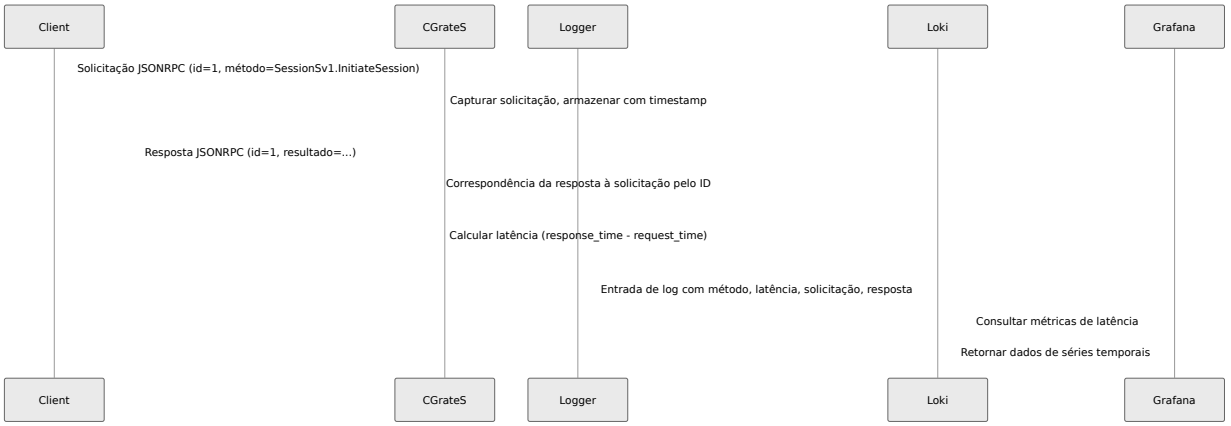
Registro JSONRPC do CGrateS

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.

Arquitetura



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 logger 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 concluído** 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 de JSONRPC.

Latência JSONRPC por Método

Um gráfico de séries temporais 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)

Detalhamento por Método

Filtre 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 solucionar chamadas de API e correlacionar solicitações com respostas.

Caso de Uso: Encontrando uma Chamada Falhada

Quando um usuário relata "Tentei chamar 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 de 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 cor: verde=0, amarelo=1-9, vermelho=10+)
- **Latência Média** — Tempo médio de ida e volta (limites codificados por cor)
- **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 principal de log mostrando chamadas correspondentes. Clique em qualquer entrada para expandir e ver o JSON completo de solicitação/resposta.

Chamadas RPC Falhadas: Visão 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 logger 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 logger 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 logger:

```
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 os logs existem para o intervalo de tempo:

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

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

Contagem de Solicitações Pendentes Alta

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 de solicitação entre solicitação e resposta

Resolução:

1. O logger 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 logs específicos:

Logs do FreeSWITCH (Servidores de Aplicação)

Caminho	Rótulo do Job
`/var/log/freeswitch/*.log`	`freeswitch`

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 no 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 origem para o monitoramento:

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

4. Verifique se o firewall permite TCP 3100 de todos os hosts para o 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 até 500 MB
3. Uma vez que o Loki esteja acessível, os logs armazenados são enviados automaticamente
4. Se o buffer estiver cheio, os logs mais antigos 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 limite de retenção de 50 GB
- 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 Ausentes

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 regenerada após alteração 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	API de ingestão e consulta de logs
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

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:

Princípio Chave: A maior parte da configuração é definida diretamente no arquivo de hosts. Os padrões de função lidam com a maioria das configurações, com group_vars usados apenas para personalizações específicas.

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
- Manipulação 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
online_charging_enabled: False    # Habilitar integração OCS
recording: True                   # Habilitar gravação de chamadas
(AS)
populate_crm: False               # Preencher CRM com dados
iniciais
```

Variáveis Globais (all:vars)

A seção `all:vars` contém configurações em todo o ambiente de implantação. Implantações modernas usam variáveis globais mínimas, com a maior parte da configuração nos padrões 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: "YKTN Lab"
site_name: YKTN
region: AU
TZ: Australia/Melbourne
```

Configuração 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{{ plmn_id.mnc_longform }}.mcc{{
plmn_id.mcc }}.3gppnetwork.org
```

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

Nomes de Rede

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

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

Configuração de DNS

```
netplan_DNS: False               # Usar systemd-resolved em vez de
DNS do netplan
manage_resolv_conf: True         # Defina como False para impedir
que o Ansible gerencie /etc/resolv.conf
```

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

Configuração do Repositório APT

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

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

```
# Configuração manual (opcional se o grupo apt_cache_servers
existir)
use_apt_cache: True              # Usar cache APT local em vez de
acesso direto ao repositório

apt_repo:
  apt_server: "10.10.1.114"      # Servidor de cache APT ou
servidor de repositório
  # Credenciais necessárias apenas quando use_apt_cache: False
  # apt_repo_username: "omni"
  # apt_repo_password: "omni"

# Configuração de downloads binários e sincronização de cache
# Usado para: (1) baixar binários de /releases/ quando
use_apt_cache: false
#           (2) sincronização do servidor de cache do Omnitouch
quando use_apt_cache: true
remote_apt_server: "apt.omnitouch.com.au"
remote_apt_user: "omni"
remote_apt_password: "omni"
```

Veja: Sistema de Cache APT

Servidor de Licença

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

Veja: Servidor de Licença

Configurações MME

```
mme_dns: False                   # Habilitar resolução DNS MME
```

Configurações SAEGW

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

Configurações IMS

```
ims_dra_support: False              # Roteia IMS através do DRA
enable_homer: False                 # Habilitar captura SIP do Homer
```

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 do Firewall

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

Servidores DNS de Roaming

```yaml
roaming_dns_servers:
  wildcard: ['10.0.99.1']
  # DNS específico do operador (baseado em PLMN)
  123456:                          # Exemplo Operador 1
    - '10.10.2.197'
  654321:                          # Exemplo Operador 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 walkthrough completo.

Site VM

```
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
rootfs
```

Substituições em nível de grupo (ambos os tipos)

```yaml
dns:
  vars:
    proxmox_interface: vmbr0              # obrigatório: bridge
    gateway: 10.10.0.1                    # obrigatório
    netmask: 255.255.255.0                # obrigatório
    vlanid: 100                           # opcional
    proxmoxLxcCores: 2                    # opcional (apenas LXC)
    proxmoxLxcMemoryMb: 4096              # opcional (apenas LXC)
    proxmoxLxcDiskSizeGb: 30              # opcional (apenas LXC)
    proxmoxLxcRootFsStorageName: SSD_RAID0 # opcional (apenas LXC)
    host_vm_network: vmbr1                # substituição de bridge
opcional
```

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 de Arquivo de Hosts - Como estruturar arquivos de 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 no Proxmox
ansible-playbook -i hosts/Customer/hosts.yml
util_playbooks/proxmox.yml

# Ou implantar contêineres LXC (apenas laboratório/teste)
ansible-playbook -i hosts/Customer/hosts.yml
util_playbooks/proxmox_lxc.yml
```

Veja: Implantação de VM/LXC no Proxmox

1. Definição de Infraestrutura (Arquivo de Hosts)

```
# 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` é onde podemos armazenar quaisquer substituições de configuração necessárias em nível de host, site ou rede.

Por exemplo, você teria uma pasta com sua configuração do OmniMessage SMSc, os troncos SIP aos quais seu TAS se conecta estariam aqui, toda a sua lógica de Roteamento Diameter, etc, etc.

Veja: Configuração de Variáveis de Grupo

3. Distribuição de Pacotes (Cache APT)

```
# 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

```
# 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

Componentes individuais podem ser implantados executando `services/twag.yml`, por exemplo, mas o `services/all.yml` cuidará de tudo, e você pode usar `--limit=myhost` ou `--limit=mmee,sgw`, etc, para limitar os hosts em que estamos trabalhando.

```
# 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
```

Documentação Relacionada

- Introdução à Implantação com 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 & 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** (Core de Pacote 4G/5G): https://docs.omnitouch.com.au/docs/repos/OmniCore
    - OmniHSS, OmniSGW, OmniPGW, OmniUPF, OmniDRA, OmniTWAG
- **OmniCall** (Voz & 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/

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 - Elas residem em `group_vars`.

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

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 de group_vars (se
especificado) → Configuração Implantada
```

Exemplo 1: Modelo de Configuração Personalizado (OmniMessage)

Alguns componentes aceitam modelos de configuração Jinja2 personalizados.

Estrutura de Arquivos

```
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. Modela 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 utilizado.

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

---

Exemplo 2: Coletas de Arquivos de Configuração (OmniTAS Gateways

& Planos de Discagem)

Alguns componentes usam diretórios de arquivos de configuração.

Estrutura de Arquivos

```
hosts/Customer/
└── group_vars/
    ├── gateways_prod/              # Configurações de 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         # Originado por Móvel (saída)
    │   ├── mt_dialplan.xml         # Terminado por 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 Home Subscriber Server aceita modelos de configuração de tempo de execução personalizados.

Estrutura de Arquivos

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

Referência no Arquivo de Hosts

```
omnihss:
  hosts:
    customer-hss01:
      ansible_host: 10.10.3.50
      gateway: 10.10.3.1
      host_vm_network: "vmbr3"
      hss_template_config: hss_runtime.exs.j2   # 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. Modela 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 utilizado.

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

---

Exemplo 4: Modelo de Configuração Personalizado (OmniMME)

O Mobility Management Entity aceita modelos de configuração de tempo de execução personalizados.

Estrutura de Arquivos

```
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. Modela 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 utilizado.

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

---

Exemplo de Estrutura de Diretórios do Mundo Real

```
hosts/Customer/
├── host_files/
│   └── production.yml          # O arquivo de hosts referencia
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     # Originado por Móvel
        ├── mt_dialplan.xml     # Terminado por Móvel
        └── mo_emergency.xml    # Roteamento de emergência
```

Parâmetros Comuns que

Referenciam group_vars

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

Parâmetro	Componente	Referências
`pgwc_template_config`	pgwc	Arquivo de modelo Jinja2 (por exemplo, `pgwc_runtime.exs.j2`)
`frr_template_config`	omniupf	Arquivo de modelo Jinja2 (por exemplo, `frr.conf.j2`)
Modelos SS7	ss7 (vários papéis)	Arquivos de modelo Jinja2 (por exemplo, `stp.j2`, `hlr.j2`, `camel.j2`)
Configurações YAML	Vários componentes	Arquivos de configuração diretos (por exemplo, `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 em tudo** - Comite todos os group_vars no Git

Quando Usar group_vars

✅ Use group_vars para:

- Modelos de configuração de componentes personalizados
- Definições de gateway SIP
- Planos de discagem de roteamento de chamadas
- Regras de roteamento Diameter
- Configurações específicas do cliente que substituem padrões

❌ Não use group_vars para:

- Configuração básica de host (IPs, nomes de host) - Use o arquivo de hosts
- Testes pontuais - Use variáveis específicas de host no arquivo de hosts
- Mudanças 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 da configuração do componente

Playbooks de Utilidade

Visão Geral

Este repositório inclui vários playbooks de utilidade para manutenção, monitoramento e tarefas operacionais. Estes complementam os playbooks principais de implantação com capacidades de gerenciamento do dia a dia.

Utilitário de Verificação de Saúde

O utilitário de Verificação de Saúde gera um relatório HTML mostrando a saúde do sistema, status dos serviços, tempo de atividade e informações de versão em todos os componentes do OmniCore.

Executa automaticamente como parte do playbook `services/all.yml`.

Uso

Execução Manual

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

Saída

O relatório é gerado em `/tmp/health_check_YYYY-MM-DD HH:MM:SS.html`

Abra em qualquer navegador da web para visualizar.

Conteúdos do Relatório

O relatório HTML exibe:

Informações do Host

- **Nome do host e endereço IP**
- **Rede/Sub-rede** (do variável `host_vm_network`, ou N/A se não configurado)
- **CPU** (contagem de vCPU)
- **RAM** (memória total e livre)
- **Disco** (espaço total e livre da partição raiz com porcentagem)
- **SO** (distribuição e versão)

Status do Serviço

- **Status do serviço** (ativo/inativo com indicadores de cor)
- **Tempo de atividade**
- **Informações de versão/liberação**

Pares de Diâmetro HSS

- **Status da conexão do banco de dados** (conectado/desconectado)
- **Conexões de pares de diâmetro** (IP, host de origem, status)
- Obtido do endpoint de métricas HSS (porta 9568)

Outras Utilidades Comuns

Configuração do Sistema Base

Função Comum (`services/common.yml`)

- Aplica a configuração base do sistema a todos os hosts
- Configura repositórios, chaves SSH, fuso horário, NTP
- Configura rede e endurecimento do sistema
- Execute isso antes de implantar serviços

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

Configurar Usuários (`services/setup_users.yml`)

- Cria e configura contas de usuário em todos os hosts
- Gerencia chaves SSH e privilégios sudo
- Garante configuração consistente de usuários

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

Reiniciar (`services/reboot.yml`)

- Reinicia graciosamente todos os hosts alvo
- Aguarda os sistemas voltarem online (timeout de 5 minutos)
- Útil após atualizações de kernel ou alterações de configuração

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

Utilidades Operacionais

Gerador de Plano de IP (`util_playbooks/ip_plan_generator.yml`)

- Gera relatório HTML de atribuições de endereços IP
- Mostra a topologia completa da rede a partir do arquivo de hosts
- Útil para documentação e solução de problemas

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

Backup HSS (`util_playbooks/hss_backup.yml`)

- Faz backup das tabelas do banco de dados HSS
- Copia o dump do MySQL para a máquina Ansible local
- Prompts interativos para o caminho do backup

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

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

- Busca os dois arquivos de captura de pacotes mais recentes de todos os hosts
- Recupera arquivos pcap de `/etc/localcapture/`
- Útil para depuração de problemas de conectividade

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

Atualizar MTU (`util_playbooks/updateMtu.yml`)

- Atualiza as configurações de MTU da interface de rede
- Aplica as mudanças via netplan
- Útil para configuração de jumbo frames

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

Documentação Relacionada

- README Principal - Visão geral e como começar
- Introdução à Implantação Ansible - Executando playbooks
- Configuração do Arquivo de Hosts - Configure seu inventário
- Arquitetura de Implantação - Visão geral completa do sistema
- Sistema de Cache APT - Gerenciamento de pacotes

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 estão configuradas (parâmetros específicos do serviço)
- Configurações específicas do cliente (PLMN, credenciais, recursos)

Localização do Arquivo

Os arquivos hosts são organizados por cliente e ambiente:

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

Estrutura de Exemplo do Arquivo Hosts

Aqui está um exemplo simplificado mostrando 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 do 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"             # Ramo de software a ser implantado
gateways_folder: "gateways_prod"  # Configuração do gateway SIP
```

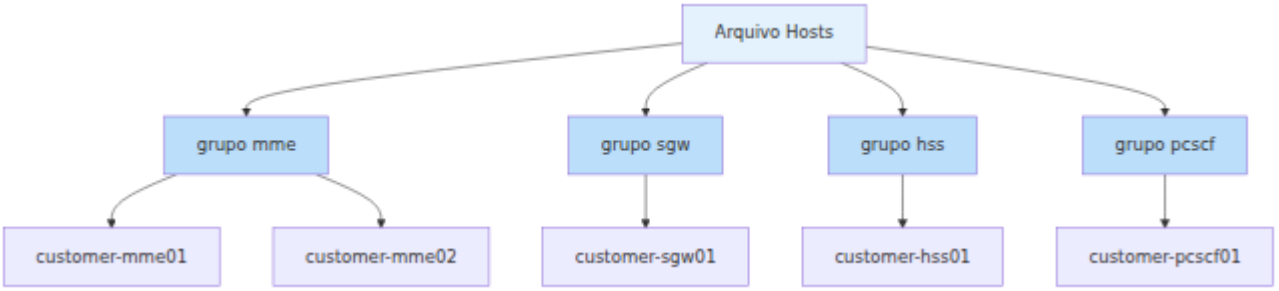
Seção de Variáveis Globais

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 for definido com hosts,
    # use_apt_cache padrão é verdadeiro e apt_repo.apt_server
    # padrão é 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
```

Compreendendo Grupos de Hosts

O Ansible organiza hosts em grupos que correspondem a funções:

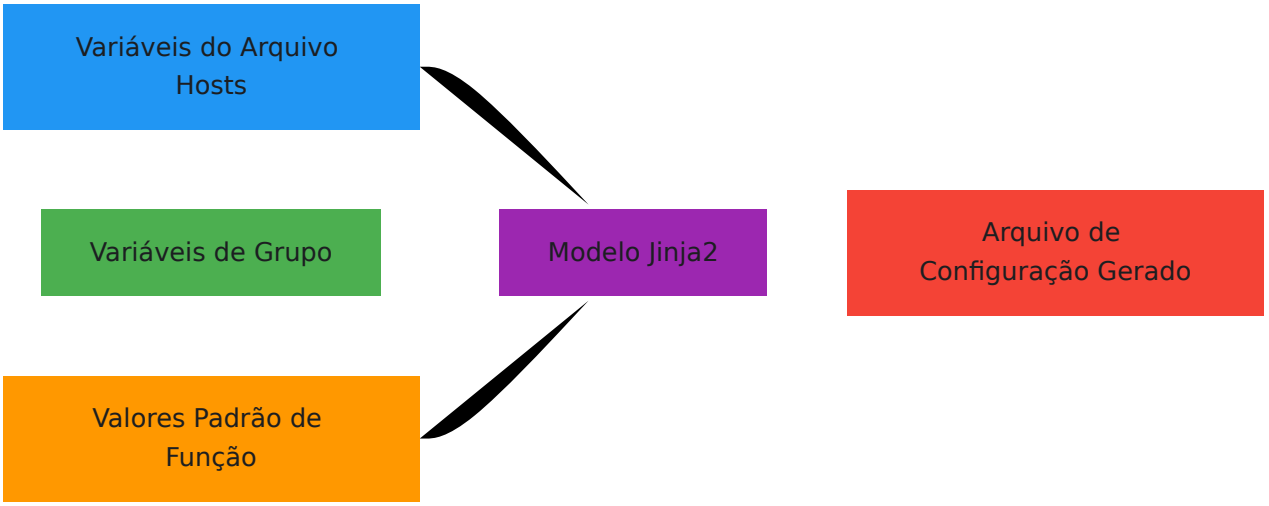


Quando você executa um playbook direcionado a `mme`, ele se aplica a todos os hosts na seção `mme:hosts:`.

Configuração com Modelos Jinja2

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

Como o Jinja2 Funciona



Exemplo de Uso de Modelo

Arquivo hosts define:

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

Modelo Jinja2 (na função):

```
# 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:

```
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

Aqui está um exemplo mais completo (com 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:
```

```
  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
```

```
      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
    firewall:
      allowed_ssh_subnets:
        - '10.0.0.0/8'
        - '192.168.0.0/16'
      allowed_ue_voice_subnets:
        - '10.0.0.0/8'
      allowed_signaling_subnets:
        - '10.0.0.0/8'

    # 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.
```

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

Referências de Documentação do Produto

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

Componentes do 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 do 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 modelo
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. Implante com playbooks do Ansible

Servidor de Licença

Visão Geral

O Servidor de Licença 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

```
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"]
    license_enforced: true
```

2. Fornecer o 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
```

Você pode verificar o status de todas as licenças acessando https://license_server .

Requisitos de Rede

Configuração do Firewall

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

Nome do Host	Endereço IP	Propósito
time.omnitouch.com.au	160.22.43.18	Servidor de validação de licença 1
time.omnitouch.com.au	160.22.43.66	Servidor de validação de licença 2
time.omnitouch.com.au	160.22.43.114	Servidor de validação de licença 3

Regras de saída necessárias:

- Protocolo: HTTPS (TCP/443)
- Destino: 160.22.43.18, 160.22.43.66, 160.22.43.114
- Direção: Saída

Requisitos de DNS

O servidor de licença requer resolução de 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 do DNSSEC e previne problemas com interceptação de DNS por dispositivos de segurança.

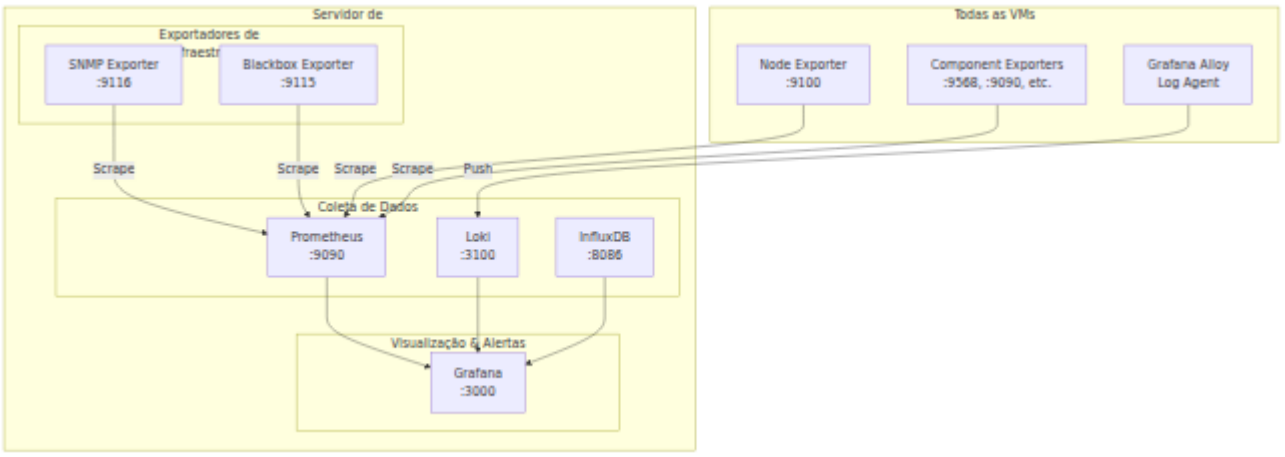
Documentação Relacionada

- 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	15 dias (padrão)
Loki	Agregação de logs	3100	7 dias ou 50GB
InfluxDB	Métricas de 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.

Datasource UID: `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, consultas 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, taxa de transferência
`OmniCSCF`	`pcscf`, `scscf`, `icscf`	9090	Registros, transações
`ApplicationServer FreeSWITCH`	`applicationserver`	9090	Contagem de chamadas, uso de canal
`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	Avaliação, cobrança, estatísticas de CDR

Exportadores de Infraestrutura

Exportador	Alvos	Variável de Configuração
SNMP (MikroTik)	Roteadores/Comutadores	`mikrotik.hosts`
SNMP (iDRAC)	Servidores Dell	`idrac.hosts`
SNMP (Synology)	Dispositivos NAS	`synology.hosts`
SNMP (SAF)	Links de micro-ondas	`saf.hosts`
SNMP (Cisco)	Comutadores	`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 executando em todas as VMs.

Datasource UID: `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.

Datasource UID: `omnicore_influxdb`

Bancos de Dados

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

Painéis do Grafana

Organização dos Painéis

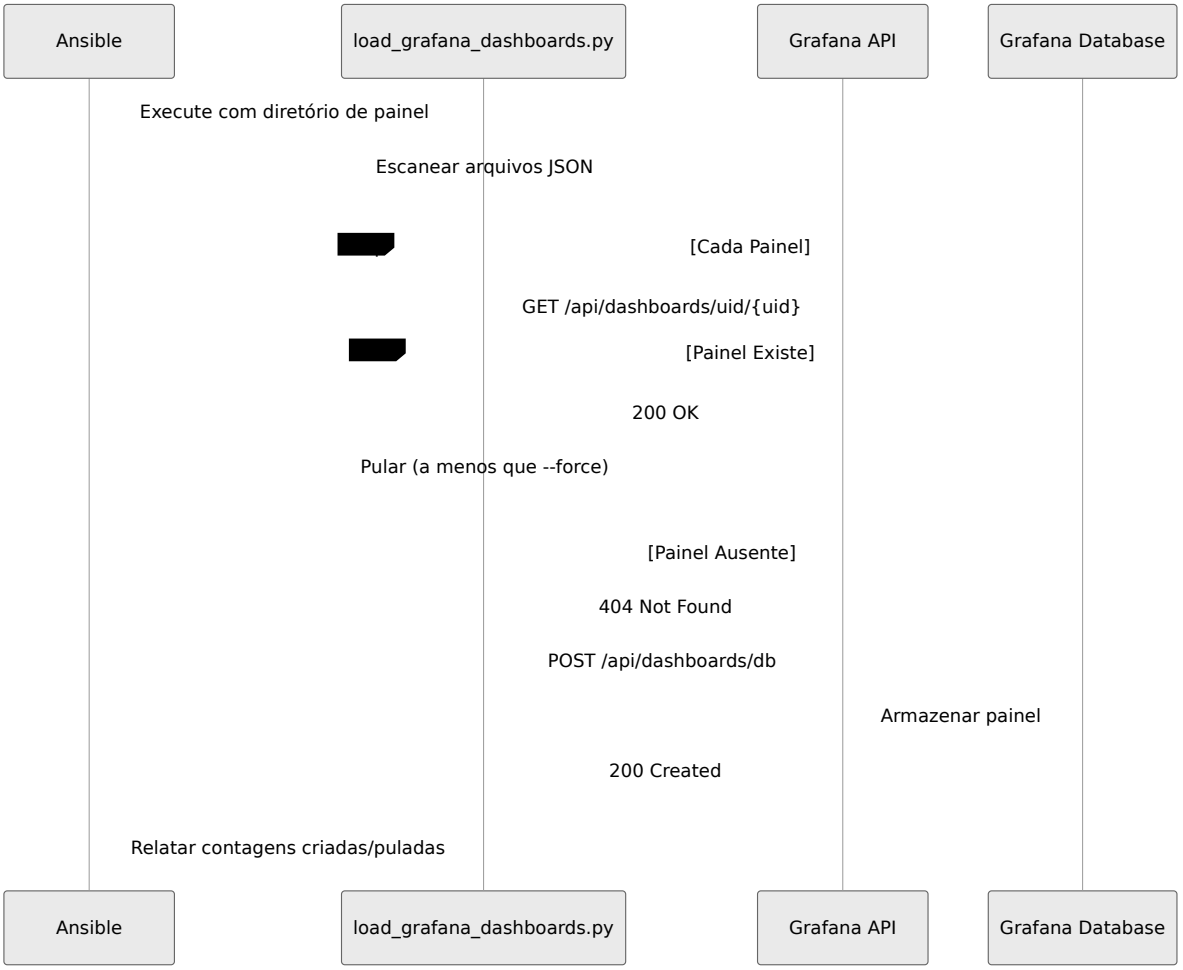
Os painéis são organizados em pastas por área funcional:

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

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 interface do Grafana
- Modificados via API
- Controlados por versão no repositório Ansible



Carregando 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 de Fonte de 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
    └── ...
```

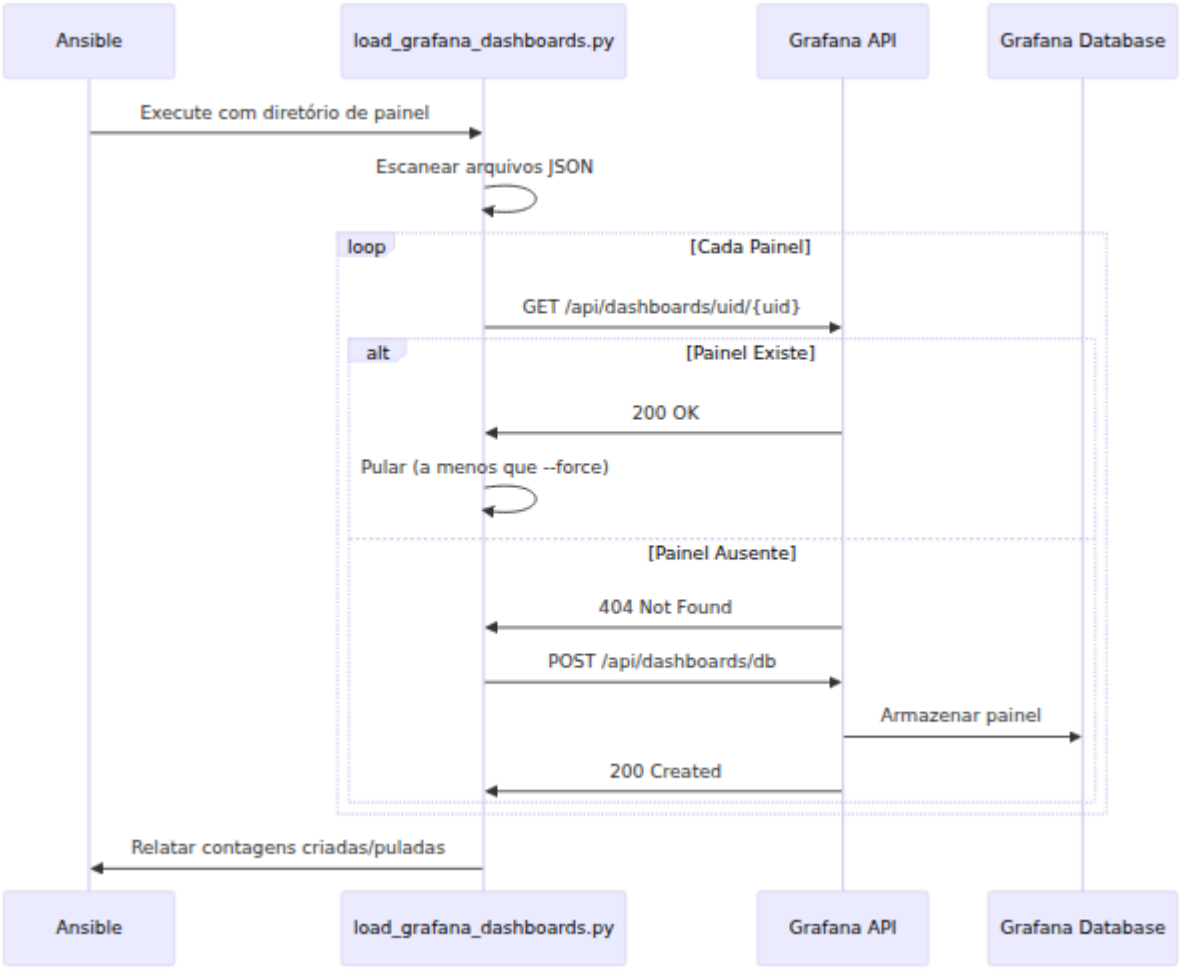
Backup de Painéis

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` (compartilhado 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
Módulo de Avaliação
intervalo de 1m
Alert triggers
Roteador de
Notificações
Fontes de Dados
Prometheus
Loki
Pontos de Contato
Webhook do Slack
Google Chat
Email

Carregamento de Alertas

Os alertas são definidos em YAML e carregados via API, semelhante aos painéis:

```
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 Caiu	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: default
      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"
```

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: default
    group_by: ['alertname', 'instance']
    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
`admin_password`	`grafana_admin_password` var	Senha do usuário administrador
`allow_embedding`	true	Habilitar incorporação em iframes
`provisioning`	`/etc/grafana/provisioning`	Diretório de provisionamento

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	Interface 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	Interface 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

1. Crie a regra de alerta em `roles/monitoring/templates/grafana/provisioning/alerting/alerts.yaml`
2. Execute o playbook de monitoramento ou carregue manualmente:

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

Fazendo Backup de Painéis

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

```
cd roles/monitoring
python3 export_grafana.py --url http://<monitoring-ip>:3000
git add templates/grafana/
git commit -m "Atualizar painéis da produção"
```

Solução 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 indicador de carregamento ou erro

Possíveis causas:

- Conexão da fonte de dados falhou
- Tempo limite da 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 Disparam

Sintomas: A condição é atendida, mas nenhuma notificação recebida

Possíveis causas:

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

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'
```

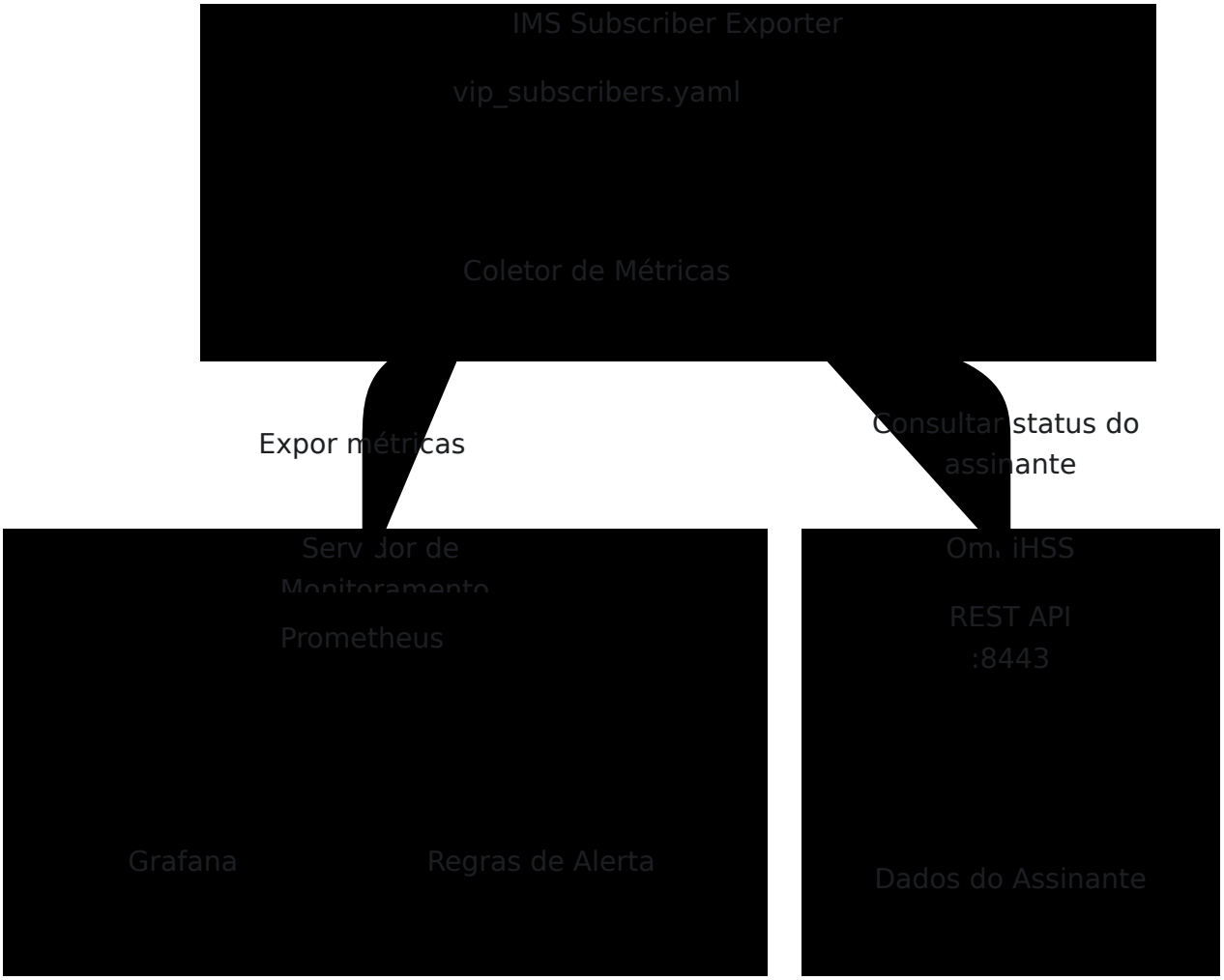
3. Teste a conectividade local:

```
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 UE para assinantes de alta prioridade designados, como serviços de emergência, hospitais e infraestrutura chave.

Arquitetura



Tipos de Serviço

Os assinantes são categorizados pelo 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: "313380930011949"
      label: "Resgate Marítimo"
      type: "full"

    - imsi: "313380930011948"
      label: "Polícia"
      type: "full"

    # Serviços apenas de dados (roteadores, IoT - sem IMS
esperado)
    - imsi: "313380930010064"
      label: "Roteador do Hospital"
      type: "data_only"

    # Serviços apenas VoIP (IMS esperado, sem portador de dados)
    - imsi: "896468419011262"
      label: "Telefones do Hospital"
      type: "voip_only"
```

Parâmetros de Configuração

Parâmetro	Tipo	Obrigatório	Padrão	Desc
`hss_url`	String	Sim	-	URL da API F 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 pi
`ping_count`	Inteiro	Não	2	Número de p ping a seren
`blackbox_exporter_url`	String	Não	-	URL do expo blackbox ren testes de pi exemplo, `http://pcs` Usa a prova remota em v local.
`subscribers`	Lista	Sim	-	Lista de assi serem moni

Parâmetros do Assinante

Parâmetro	Tipo	Obrigatório	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 da 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	`313380930011948`
`label`	Nome legível por humanos	`Polícia`
`msisdn`	Número de telefone (do HSS)	`24724748250`
`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 da 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 à 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	Descri��ão
Serviços Saudáveis	Contagem de assinantes que atendem aos critérios de saúde
Serviços Não Saudáveis	Contagem de assinantes que falham 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 da 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 do 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 Telefones do Hospital (voip_only) está OFFLINE. MSISDN: 24724766000`
- `Serviço de Roteador do Hospital (data_only) está OFFLINE. IMSI: 313380930010064`
- `Serviço de Polícia (full) está OFFLINE. MSISDN: 24724748250 IMSI: 313380930011948`

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. Implante 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 do 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 primária (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

Aqui está o template completo `netplan.yaml.j2` com comentários explicando cada seção:

```
network:
  version: 2
  ethernets:
    # Interface primária - 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 %}
        search:
        - slice
      routes:
      - to: "default"
        via: "{{ gateway }}"

{% if secondary_ips is defined %}
    # Interfaces secundárias - percorre o dicionário secondary_ips
do inventário
    # Nomeação da interface: ens19, ens20, ens21... (18 +
loop.index)
{% for nic_name, nic_config in secondary_ips.items() %}
    ens{{ 18 + loop.index }}:
      addresses:
      - "{{ nic_config.ip_address }}/{{ mask_cidr | default(24)
}}"
{% if nic_config.routes is defined %}
      # 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`
- Interfaces secundárias são nomeadas `ens19`, `ens20`, etc. para corresponder à nomenclatura de NIC do Proxmox
- Cada IP secundário pode ter seu próprio gateway e rotas estáticas

Configuração da Interface Primária

A interface primária (eth0) é configurada automaticamente usando:

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

Os servidores DNS são configurados automaticamente para:

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

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>
    gateway: <gateway_ip>
    host_vm_network: <proxmox_bridge>
    vlanid: <vlan_id>
    routes:                    # Opcional - rotas estáticas via
esta interface
      - '<destination_cidr>'
      - '<destination_cidr>'
```

Nomeação de Interfaces

As interfaces secundárias são nomeadas automaticamente usando o esquema de nomeação previsível do Ubuntu:

- Primeira interface secundária: `ens19`
- Segunda interface secundária: `ens20`
- Terceira interface secundária: `ens21`
- E assim por diante...

Isso corresponde aos nomes das interfaces atribuídos pelo Proxmox ao adicionar NICs adicionais a uma VM.

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
          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

A configuração acima gera:

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

Integração com Proxmox

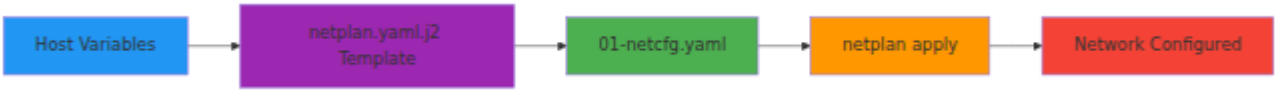
Ao usar o playbook `proxmox.yml`, NICs secundárias são criadas automaticamente na VM:

1. **Novas VMs**: NICs secundárias são adicionadas durante a provisão 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. Variáveis do arquivo de hosts são passadas para o template Jinja2
2. O template é renderizado para `/etc/netplan/01-netcfg.yaml`
3. Quaisquer configurações de netplan existentes são removidas para evitar conflitos
4. `netplan apply` ativa a configuração
5. Endereços IP são verificados com `ip addr show`

Casos de Uso Comuns

Diameter Edge Agent (DEA) com IP Público

```
<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 de parceiros de
roaming
```

PGW com Interface S5/S8

```
<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

```
<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 peer), 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 Peer DRA

Configure um peer 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 na 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 - Provisão de VM
- Referência de Configuração - Todas as variáveis de configuração

Implantação de VM/LXC Proxmox

A maioria dos nossos clientes executa a pilha OmniCore no Proxmox. Um único playbook — `services/proxmox.yml` — provisiona tanto VMs QEMU quanto contêineres LXC com base em um alternador de `deployment_type` em todo o site.

Ainda continuamos a oferecer suporte ao VMware, HyperV e nuvem (Vultr / AWS / GCP) para implantações.

Veja Também:

- Configuração do Arquivo Hosts — defina VMs/LXCs a serem implantadas
- 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)

Um Site = Um Tipo

Todas as entradas `proxmoxServers.*.deployment_type` em um site **devem concordar**. Misturar VM e LXC no mesmo arquivo de host não é suportado e falha na validação. Escolha um por site.

Configuração do Proxmox

1. Criar Token de API

```
# Proxmox UI: Datacenter → Permissions → API Tokens
# Criar token: root@pam!<TokenName>
# Copie o segredo do token (mostrado uma vez)
```

2a. Criar Modelo de VM Cloud-Init (apenas sites de VM)

Execute este script no host Proxmox. Ele baixa a imagem de nuvem do Ubuntu e cria um modelo limpo — as credenciais do cloud-init e as chaves SSH são injetadas no momento da clonagem pelo `proxmox.yml` (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 web do Proxmox funcione.
- A flag `-N` no wget só baixa se for mais recente que a cópia local.

Alternativa: Modelo Manual a partir de ISO

Se as imagens de nuvem não estiverem disponíveis ou você precisar de uma instalação personalizada:

Passo 1: Criar VM via Web UI

- 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 para Modelo

- Selecionar VM → Hardware → Adicionar → CloudInit Drive (selecionar armazenamento, por exemplo, local-lvm)
- Hardware → Opções → QEMU Guest Agent → Habilitar
- Clique com o botão direito na VM → Converter para Modelo
- **Não** defina usuário/senha do cloud-init no modelo — `proxmox.yml` injeta esses dados no momento da clonagem a partir de `proxmoxTemplateUser` / `proxmoxTemplatePassword` (com fallback de `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 Hosts

Implantação de VM

```
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        # 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
    netmask: 255.255.255.0
    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
    netmask: 255.255.255.0
    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 services/proxmox.yml
```

Deletar

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

O playbook de exclusão lida automaticamente com VMs e LXCs, independentemente do `deployment_type`.

Comportamento do Playbook

Guardas de Tipo

- Se uma VM com o nome de host alvo 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 entre as entradas `proxmoxServers` por índice de host dentro do grupo.
- 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

do 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 privilegiados (`unprivileged: 0`) com aninhamento habilitado.

Recurso Existente

Tanto os caminhos 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 mudaram.

Marcação

Cada recurso é marcado com seus nomes de grupo Ansible + `site_name`.

Distribuição Entre Nós

Mesma lógica de round-robin para VMs e LXCs:

```
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)
```

Substituições por Host

Qualquer um dos parâmetros de LXC em nível de grupo (`proxmoxLxcCores`, `proxmoxLxcMemoryMb`, `proxmoxLxcDiskSizeGb`, `proxmoxLxcRootFsStorageName`, `proxmoxLxcOsTemplate`, `host_vm_network`) também pode ser definido por host sob o nome do host.

```
dns:
  hosts:
    site-dns01:
      ansible_host: 192.168.1.20
      proxmoxLxcDiskSizeGb: 120   # substituição por host
      host_vm_network: vmbr1      # substituição de ponte por host
  vars:
    proxmox_interface: vmbr0
    gateway: 192.168.1.1
    netmask: 255.255.255.0
```

`util_playbooks/proxmox_lxc.yml` Legado

Depreciado. Usa a antiga forma de variável `proxmoxServerAddress`/`PROXMOX_API_TOKEN` e não está conectado ao fluxo de implantação. Todos os novos sites devem usar `services/proxmox.yml` com `deployment_type: lxc`.

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
├── dns.yml
├── common.yml
├── license_server.yml
├── monitoring.yml
│   └── grafana.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
│   ├── smsc.yml
│   └── omnisep.yml
├── omniepdg.yml
├── omniss7.yml
├── ocs.yml
├── crm.yml
├── ran_monitor.yml
└── 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 cobre 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, EPC, IMS, OCS, CRM
`epc.yml`	Núcleo de Pacotes	MME, SGW-C, PGW-C, UPF, HSS, DRA
`ims.yml`	Voz/IMS	P/I/S-CSCF, Servidor de Aplicação, XCAP, Mensagens
`ocs.yml`	Cobrança	CGrateS, cluster KeyDB, sincronização OCS
`monitoring.yml`	Observabilidade	Prometheus, Grafana, exporters, HOMER

Playbooks de Infraestrutura

Playbook	Descrição
`common.yml`	Configuração base do OS, pacotes, NTP, agentes de log
`setup_users.yml`	Contas de usuário locais e chaves SSH
`dns.yml`	Implantação do servidor DNS
`license_server.yml`	Servidor de licença OmniCore
`netplan.yml`	Configuração da interface de rede
`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 PDN
`upf.yml` / `omniupf.yml`	OmniUPF	Função do Plano do Usuário (SGW-U/PGW-U combinado)
`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
`omnimessage.yml`	OmniMessage	Controlador de SMS sobre IP
`smsc.yml`	SMSC	Centro de Serviço de Mensagens Curtas

Serviços de Suporte

Playbook	Descrição
`ocs.yml`	Sistema de Cobrança Online (CGrateS + KeyDB)
`crm.yml`	Portal de gerenciamento de clientes
`omniss7.yml`	Gateway SS7/SIGTRAN
`homer.yml`	Captura de pacotes SIP/Diameter
`grafana.yml`	Painéis e alertas do Grafana
`promtail.yml`	Envio de logs para o Loki
`ran_monitor.yml`	Integração de monitoramento RAN

Provisionamento de VM

Playbook	Descrição
`proxmox.yml`	Criar VMs no Proxmox VE
`proxmox_lxc.yml`	Criar contêineres LXC (laboratório/teste)
`proxmox_delete.yml`	Excluir VMs/LXCs do Proxmox

Operações

Playbook	Descrição
`backup.yml`	Backup de bancos de dados e configurações
`reboot.yml`	Reinicialização controlada de hosts
`shutdown.yml`	Desligamento controlado
`apt_update.yml`	Atualizar pacotes
`apt_refresh_metadata.yml`	Atualizar metadados do cache APT
`speedtest.yml`	Teste de throughput da rede

Playbooks de Restauração

Playbook	Descrição
`restore_applicationserver.yml`	Restaurar OmniTAS a partir do backup
`restore_omnimessage_controller.yml`	Restaurar OmniMessage a partir do backup
`restore_smsc.yml`	Restaurar SMSC a partir do backup

Uso

Implantar Tudo

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

Implantar Subsistema Específico

```
# Apenas o núcleo 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 no Proxmox
`proxmox_delete.yml`	Excluir VMs/LXCs no Proxmox
`proxmox_lxc.yml`	Provisionar contêineres LXC no Proxmox
`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
`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

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 Proxmox para configuração detalhada.

Proxmox

Arquivos: `util_playbooks/proxmox.yml`, `util_playbooks/proxmox_lxc.yml`, `util_playbooks/proxmox_delete.yml`

```
# Provisionar VMs
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

# Provisionar contêineres LXC
ansible-playbook -i hosts/customer/host_files/production.yml
util_playbooks/proxmox_lxc.yml

# Excluir VMs/LXCs
ansible-playbook -i hosts/customer/host_files/production.yml
util_playbooks/proxmox_delete.yml --limit old-host
```

VMware vSphere

Arquivos: `util_playbooks/vmware.yml`, `util_playbooks/vmware_delete.yml`

Pré-requisitos: Requer que as dependências do `requirements.txt` sejam instaladas.

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"
```

Verificação de Saúde

Arquivo: `util_playbooks/health_check.yml`

Gera um relatório de saúde abrangente em HTML cobrindo todos os serviços implantados do OmniCore e OmniCall.

```
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 de pares Diameter e status
OmniTAS	Chamadas ativas, sessões, registros, uso de 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 ambos.

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

Formatos de Arquivo de Backup

Tipo	Padrão do Nome do Arquivo	Conteúdos
Banco de Dados	`hss_dump_<hostname>_<timestamp>.sql`	Dump MySQL do banco de dados `omnihss`
Configuração	`hss_<hostname>_<timestamp>.tar.gz`	Arquivo do diretório `/etc/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 os 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 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 MySQL StoreDB (opcional)
5. Restaura a configuração de `/etc/cgrates` (opcional)
6. Inicia o CGrateS em todos os nós

Formatos de Arquivo de Backup

Tipo	Padrão do Nome do Arquivo	Conteúdos
KeyDB DataDB	`keydb_dump_<hostname>_<timestamp>.rdb`	Snapshot RDB do KeyDB
MySQL StoreDB	`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 dos hosts (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 hostname, IP, protocolo, porta, serviço
`./open_ports.rst`	Tabela reStructuredText para documentação Sphinx

Dados Coletados

Campo	Descrição
Hostname	Nome do host no 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: Insira o nome de usuário a ser excluído quando solicitado.

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 de teste (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
```