

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.

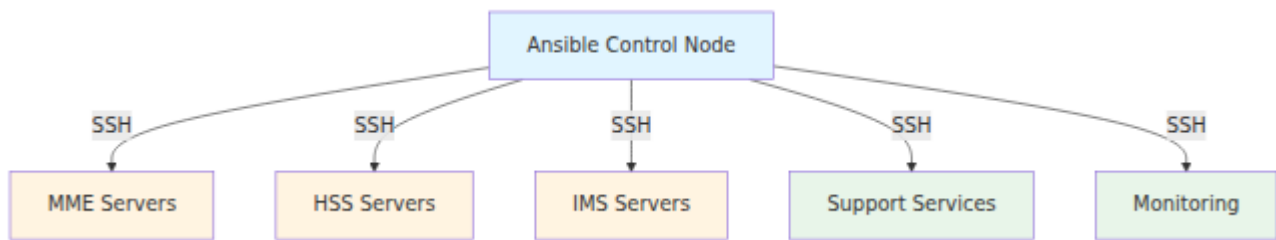
O que é Ansible?

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/container.

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.omnitech.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

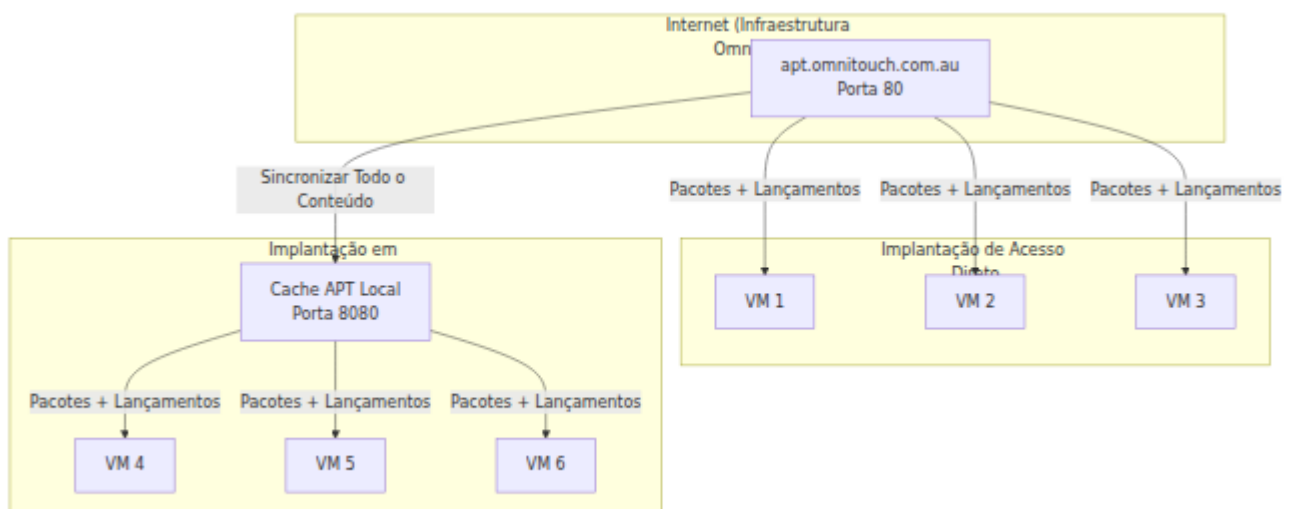
O sistema APT da Omnitouch fornece distribuição de pacotes para todas as implantações. Dois tipos de conteúdo são servidos:

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 <code>.deb</code> personalizados (omnihss, omnimme, etc.)	<code>/dists/<distro>/</code>
Pacotes Ubuntu	Pacotes Ubuntu em cache com todas as dependências	<code>/<distro>/pool/main/</code>
Lançamentos do GitHub	Binários pré-compilados (Prometheus, Grafana, Homer, etc.)	<code>/releases/<org>/<repo>/</code>
Tarballs de Fonte	Arquivos de origem para aplicativos web (CGrateS_UI, speedtest)	<code>/repos/</code>
Pacotes de Terceiros	Galera, FRR, InfluxDB, KeyDB, etc.	<code>/releases/<vendor>/</code>

Variáveis de Configuração

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

Variáveis de Configuração

`apt_repo`
(Fontes de pacotes APT)

`remote_apt_*`
(Downloads binários)

O Que Eles Configuram

`/etc/apt/sources.list`

Downloads binários
`/releases/*`

Propósitos das Variáveis

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

Quando Cada Conjunto de Variáveis É Usado

Cenário	Fontes APT (<code>apt_repo</code>)	Downloads Binários (<code>remote_apt_*</code>)
<code>use_apt_cache:</code> <code>true</code>	Usa <code>apt_repo.apt_server</code>	Usa <code>apt_repo.apt_server</code>
<code>use_apt_cache:</code> <code>false</code>	Usa <code>apt_repo.*</code> com credenciais	Usa <code>remote_apt_*</code> 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

```
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	Descrição
<code>apt_repo.apt_server</code>	String	Sim	-	Nome do host ou endereço IP do servidor APT
<code>apt_repo.apt_repo_username</code>	String	Sim	-	Nome de usuário da Autenticação Básica HTTP para fonte APT
<code>apt_repo.apt_repo_password</code>	String	Sim	-	Senha da Autenticação Básica HTTP para fonte APT

Downloads Binários (`remote_apt_*`)

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

Geral

Parâmetro	Tipo	Requerido	Padrão	Descrição
<code>use_apt_cache</code>	Booleano	Sim	-	Deve ser <code>false</code> para acesso direto

Padrões de URL (Acesso Direto)

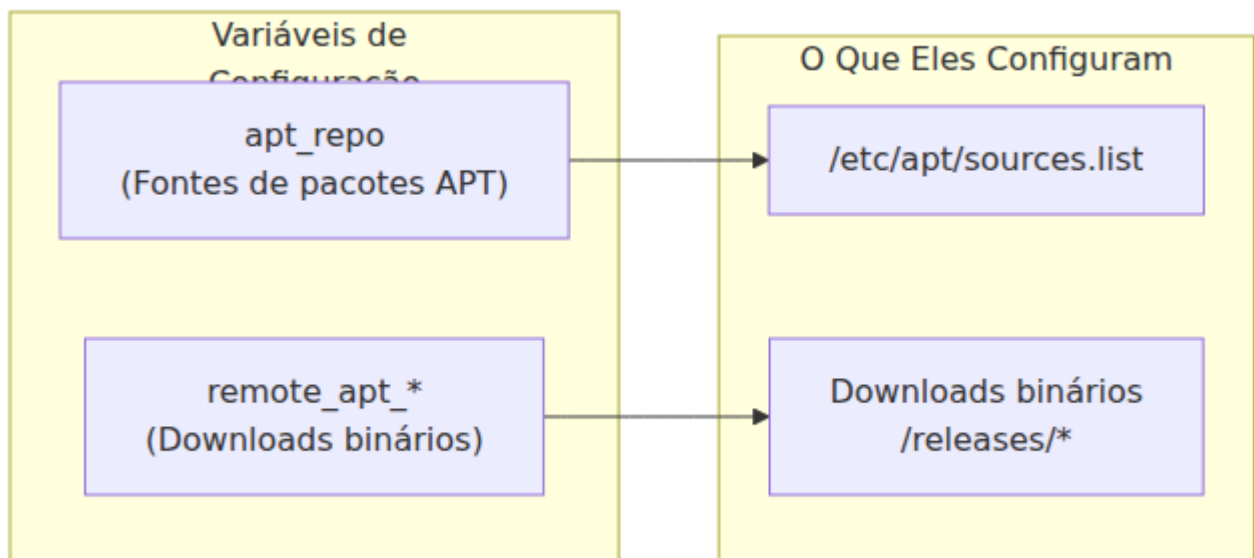
Fontes de Pacotes APT (configuradas em `/etc/apt/sources.list`):

```
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:

```
apt_cache_servers:
  hosts:
    customer-apt-cache:
      ansible_host: 192.168.1.100
      gateway: 192.168.1.1
  vars:
    # 0 servidor de cache sincroniza pacotes do repositório
    # autenticado
    remote_apt_server: "apt.omnitech.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.omnitech.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	Descrição
<code>apt_repo.apt_server</code>	String	Sim	Auto-derivado	IP do servidor de cache local. Derivado automaticamente do primeiro endereço IP disponível se não especificado.
<code>apt_repo.apt_repo_username</code>	String	Não	-	Não necessário para autenticação.
<code>apt_repo.apt_repo_password</code>	String	Não	-	Não necessário para autenticação.

Sincronização do Servidor de Cache (`remote_apt_*`)

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

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

Geral

Parâmetro	Tipo	Requerido	Padrão	Descrição
<code>use_apt_cache</code>	Booleano	Não	<code>true</code>	Definido automaticamente como <code>true</code> quando o grupo <code>apt_cache_servers</code> existe
<code>apt_cache_port</code>	Inteiro	Não	<code>8080</code>	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_exporter-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

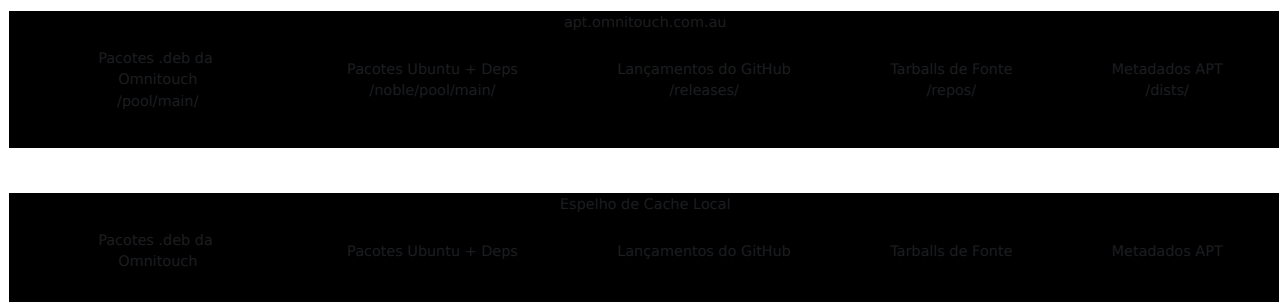
1. **Provisione o servidor de cache** (VM ou contêiner LXC com disco de 50+ GB)
2. **Execute o playbook de configuração do cache:**


```
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 **tudo o conteúdo** do servidor APT da Omnitouch usando download recursivo com wget:

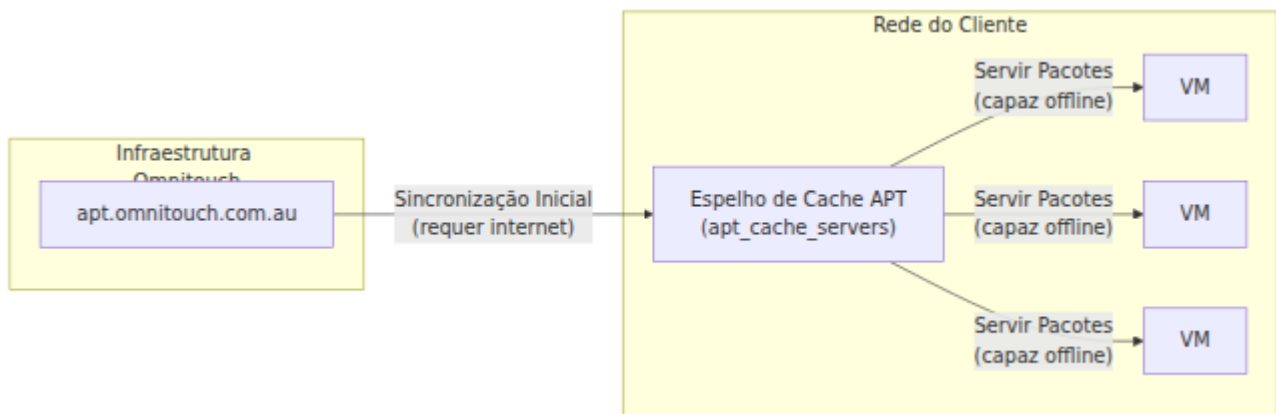


Diretórios de conteúdo sincronizados:

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

```
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:

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

    apt_repo:
      apt_server: "apt.omnitech.com.au"
      apt_repo_username: "usuario"
      apt_repo_password: "senha"

    remote_apt_server: "apt.omnitech.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.

```
all:
  vars:
    use_apt_cache: false

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

# Downloads binários - usados por todos os hosts
    remote_apt_server: "apt.omnitech.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.omnitech.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.omnitech.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.omnitech.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:

```
# 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: "vibr0"
      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.omnitech.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

Referência de Configuração

Visão Geral

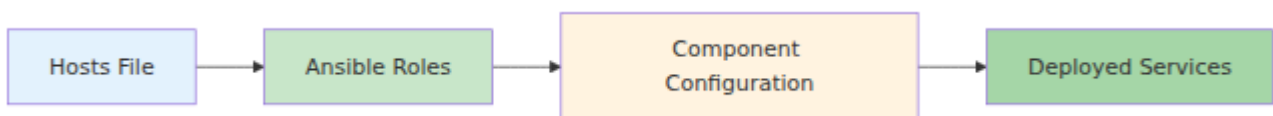
Este documento fornece uma referência abrangente para configurar implantações do OmniCore através de arquivos de hosts. A configuração é definida principalmente em arquivos de inventário de hosts, com mínimas substituições de `group_vars` necessárias para implantações modernas.

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
- Tratamento de IPs públicos

Parâmetros Comuns de Host

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

Flags Específicas de Serviço

```
cdrs_enabled: True           # Habilitar geração de CDR
in_pool: False               # Excluir do pool de
balanceamento de carga
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 mínimas variáveis globais, 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 de forma única 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 netplan
```

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 vs 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 de 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 Homer
```

Configuração do Monitor RAN

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

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

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

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

Configuração de Firewall

```
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

```
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)

```
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

```
proxmoxServers:
  customer-prxm01:
    proxmoxServerAddress: 10.10.0.100
    proxmoxServerPort: 8006
    proxmoxRootPassword: password
    proxmoxApiTokenName: AnsibleToken
    proxmoxApiTokenSecret: "token-secret"
    proxmoxTemplateName: ubuntu-24.04-cloud-init-template
    proxmoxTemplateId: 9000
    proxmoxNodeName: pve01

# Configurações padrão do Proxmox
proxmoxServerAddress: 10.10.0.100
proxmoxServerPort: 8006
proxmoxNodeName: 'pve01'
proxmoxLxc0sTemplate: 'local:vztmpl/ubuntu-24.04-standard_24.04-2_amd64.tar.zst'
proxmoxApiTokenName: DocsTest
proxmoxLxcCores: 8
proxmoxLxcDiskSizeGb: 20
proxmoxLxcMemoryMb: 64000
proxmoxLxcRootFsStorageName: SSD_RAID0
proxmoxLxcBridgeName: vmbr0
proxmoxTemplateName: "ubuntu-24.04-cloud-init-template"
proxmoxStorage: SSD_RAID0
vLabNetmask: 24
PROXMOX_API_TOKEN: "token-secret"
vlabRootPassword: password
vLabPublicKey: "ssh-rsa AAAAB3..."
mask_cidr: 24
```

VMware vCenter

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

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

netmask: 255.255.255.0
```

Documentação Relacionada

- **Padrão de Planejamento de IP** - Diretrizes de arquitetura de rede e alocação de IP
- **Configuração do 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.omnitech.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 services/proxmox.yml

# Ou implantar contêineres LXC (apenas laboratório/teste)
ansible-playbook -i hosts/Customer/hosts.yml
services/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 Ansible](#) - Começando
- [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

Documentação do Produto

Para informações detalhadas sobre a configuração de cada componente:

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

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 vivem - 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 vivem 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

```
omnimessage:  
  hosts:  
    customer-smsc-controller01:  
      ansible_host: 10.10.3.219  
      gateway: 10.10.3.1  
      host_vm_network: "vmbr3"  
      smsc_template_config: smsc_controller.exs  # Referencie o  
nome do seu arquivo de modelo em group_vars
```

O que acontece:

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

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

Detalhes da configuração: Veja

<https://docs.omnitouch.com.au/docs/repos/OmniCall>

Exemplo 2: Coleções de Arquivos de Configuração (OmniTAS)

Gateways & Dialplans)

Alguns componentes usam diretórios de arquivos de configuração.

Estrutura de Arquivos

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

Referência no Arquivo de Hosts

```
applicationserver:
  hosts:
    customer-tas01:
      ansible_host: 10.10.3.60
      gateway: 10.10.3.1
      host_vm_network: "vmbr3"
      gateways_folder: "gateways_prod"  # Referencie sua pasta de
gateways a ser usada neste host
```

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/` 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"`

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 em 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. Faz o template com Jinja2 (pode usar `{{ inventory_hostname }}`, `{{ plmn_id.mcc }}`, etc.)
4. Implanta em `/etc/omnihss/runtime.exs`
5. Reinicia o serviço

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

Detalhes da configuração: Veja

<https://docs.omnitech.com.au/docs/repos/OmniCore>

Exemplo 4: Modelo de Configuração Personalizado (OmniMME)

O Mobility Management Entity aceita modelos de configuração em 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

```
omnimme:  
  hosts:  
    customer-mme01:  
      ansible_host: 10.10.3.51  
      gateway: 10.10.3.1  
      host_vm_network: "vmbr3"  
      mme_template_config: mme_runtime.exs.j2  # Referencie o  
nome do seu arquivo de modelo em group_vars
```

O que acontece:

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

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

Detalhes da configuração: Veja

<https://docs.omnitech.com.au/docs/repos/OmniCore>

Exemplo de Estrutura de Diretório do Mundo Real

```
hosts/Customer/
├── host_files/
│   └── production.yml           # 0 arquivo de hosts referencia os
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         # Originadas por Móvel
    ├── mt_dialplan.xml         # Terminadas por Móvel
    └── mo_emergency.xml        # Roteamento de emergência
```

Parâmetros Comuns que

Referenciam group_vars

Parâmetro	Componente	Referências
<code>smc_template_config</code>	omnimessage	Arquivo de modelo Jinja2 (ex: <code>smc_controller.exs</code>)
<code>smc_smpp_template_config</code>	omnimessage_smpp	Arquivo de modelo Jinja2 (ex: <code>smc_smpp.exs</code>)
<code>gateways_folder</code>	applicationserver	Nome da pasta (ex: <code>sip_profiles</code>)
Dialplans (automático)	applicationserver	Pasta <code>dialplan/</code> de XMLs de roteamento
<code>tas_template_config</code>	applicationserver	Arquivo de modelo Jinja2 (ex: <code>tas_runtime.exs.j2</code>)
<code>hss_template_config</code>	omnihss	Arquivo de modelo Jinja2 (ex: <code>hss_runtime.exs.j2</code>)
<code>mme_template_config</code>	omnimme	Arquivo de modelo Jinja2 (ex: <code>mme_runtime.exs.j2</code>)
<code>dra_template_config</code>	dra	Arquivo de modelo Jinja2 (ex: <code>dra_runtime.exs.j2</code>)

Parâmetro	Componente	Referências
<code>pgwc_template_config</code>	pgwc	Arquivo de modelo Jinja2 (ex: <code>pgwc_runtime.exs.j2</code>)
<code>frr_template_config</code>	omniupf	Arquivo de modelo Jinja2 (ex: <code>frr.conf.j2</code>)
Modelos SS7	ss7 (vários papéis)	Arquivos de modelo Jinja2 (ex: <code>stp.j2</code> , <code>hlr.j2</code> , <code>camel.j2</code>)
Configurações YAML	Vários componentes	Arquivos de configuração diretos (ex: <code>upf_config.yaml</code> , <code>crm_config.yaml</code>)

Pontos Chave

1. **group_vars contém personalizações** - Substituições para configurações padrão
2. **Referencie pelo nome** - Use parâmetros como `smc_template_config` ou `gateways_folder`
3. **Modelos suportam Jinja2** - Acesse qualquer variável do Ansible com `{{ variable_name }}`
4. **Pastas implantam tudo** - Todos os arquivos nas pastas referenciadas são copiados
5. **Controle de versão de tudo** - Faça commit de 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 os padrões

☐ Não use group_vars para:

- Configuração básica de host (IPs, nomes de host) - Use o arquivo de hosts
 - Testes únicos - Use variáveis específicas do host no arquivo de hosts
 - Alterações temporárias - Edite no alvo, faça commit 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/  
└─ Nome_Do_Cliente/  
    └─ host_files/  
        ├── production.yml  
        ├── staging.yml  
        └─ lab.yml
```

Estrutura de Exemplo do Arquivo Hosts

Aqui está um exemplo simplificado mostrando as seções principais:

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: Cliente

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 MME (1-255)
mme_gid: 1                 # ID do Grupo MME
network_name_short: Cliente # Nome da rede (exibido nos
                             telefones)
network_name_long: Rede do Cliente
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      # Controle/plano de usuário
                             separado
```

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

[Referência de Configuração](#)

Servidor de Aplicação:

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

Seção de Variáveis Globais

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

```

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

    # Identidade do Cliente
    customer_name_short: customer
    customer_legal_name: "Cliente 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: Cliente
    network_name_long: Rede do Cliente

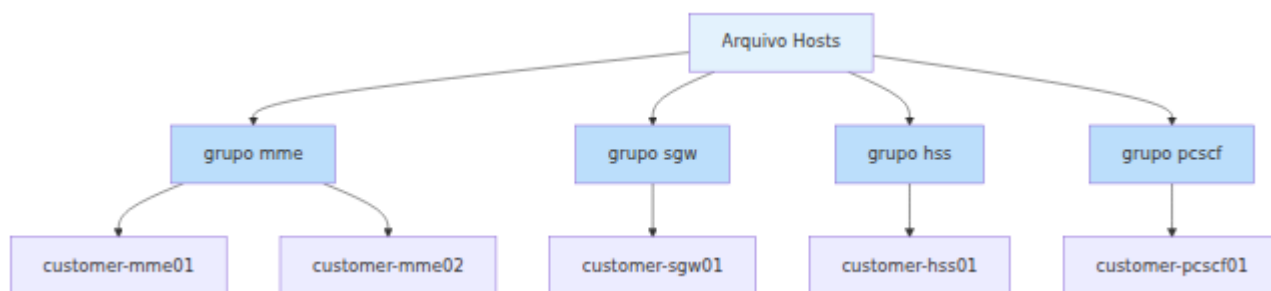
    # Repositório APT
    # Nota: Se o grupo apt_cache_servers estiver definido com
    hosts,
    # use apt_cache padrão é true e apt_repo.apt_server
    # padrão é o IP do primeiro servidor de cache automaticamente
    apt_repo:
      apt_server: "10.254.10.223"
      apt_repo_username: "customer"
      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:

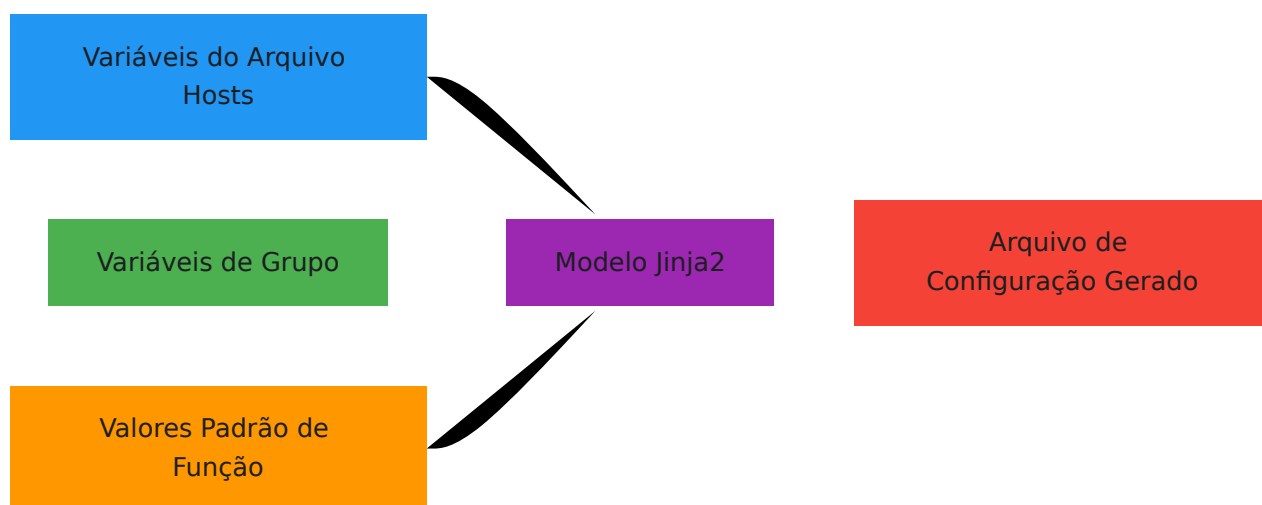


Quando você executa um playbook direcionado a `mme`, ele se aplica a todos os hosts na seção `mme:hosts:`.

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 Funciona o Jinja2



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 infraestrutura e configurações específicas de host, `group_vars` pode sobrescrever 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):

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: Cliente

network_name_long: Rede do Cliente

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"

cdns_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: "Rede do Cliente Inc."
    site_name: "DC Primário"
    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: Cliente
network_name_long: Rede do Cliente
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 Proxmox)
proxmoxServers:
  customer-prxm01:
    proxmoxServerAddress: 10.10.0.100
    proxmoxServerPort: 8006
    proxmoxApiTokenName: Cliente
    proxmoxApiTokenSecret: "token-secret"
    proxmoxTemplateName: ubuntu-24.04-cloud-init-template
    proxmoxNodeName: pve01
```

Consulte [Implantação de VM/LXC do Proxmox](#) para detalhes completos de configuração e configuração do Proxmox.

Referências da Documentação do Produto

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

Componentes OmniCore:

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

Componentes OmniCall:

- **Documentação do OmniCall:**
<https://docs.omnitech.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.omnitech.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. Implemente com playbooks do Ansible

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:

```
# 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:

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

```

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

```

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


```
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.

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/Custom/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](#)

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
  ethernet:
    # 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

```
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: "vibr0"
          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:

```
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

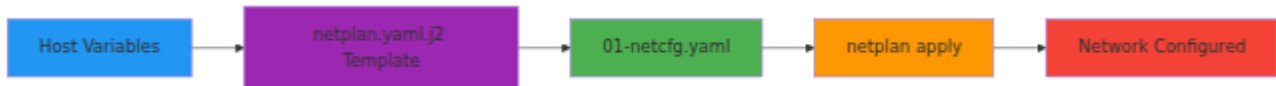
Ao usar o playbook `proxmox.yml`, NICs secundárias são criadas automaticamente na VM:

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 no Proxmox

A maioria dos nossos clientes executa a pilha OmniCore no Proxmox, este guia explica em detalhes como usar os plays `proxmox` para configurar seu ambiente usando o Proxmox.

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

Veja Também:

- [Configuração do Arquivo Hosts](#) - Definir VMs a serem implantadas
- [Padrão de Planejamento de IP](#) - Diretrizes de atribuição de endereços IP
- [Configuração do Netplan](#) - IPs secundários e configuração de múltiplas NICs
- [Arquitetura de Implantação](#) - Fluxo de trabalho completo de implantação

LXC vs VM

Contêineres LXC:

- Leves, compartilham o kernel do host
- Inicialização rápida, baixa sobrecarga
- Isolamento limitado
- Não podem executar kernels ou módulos de kernel personalizados
- **Não adequado para implantações em produção**
- **Não pode executar UPF** (requer módulos de kernel/dispositivos TUN)

VMs (KVM):

- Virtualização completa com kernel dedicado
- Isolamento completo
- Pode executar módulos de kernel e 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)

Configuração do Proxmox

1. Criar Token de API

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

2. Criar Modelo de VM Cloud-Init (apenas para VMs)

Execute este script no host Proxmox. Ele baixa a imagem de nuvem do Ubuntu e cria um modelo com as credenciais do usuário do cloud-init.

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

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

echo "=== Baixando imagem de nuvem do Ubuntu ==="
cd /var/lib/vz/template/iso
wget -N "$IMAGE_URL"

echo "=== Limpando modelo antigo ==="
qm destroy $TEMPLATE_ID --purge 2>/dev/null || true

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

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

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

```
qm template $TEMPLATE_ID
```

```
echo "=== Modelo $TEMPLATE_ID criado com sucesso ==="
```

Notas:

- O modelo fornece um login de fallback: `omnitech` / `password` (para acesso ao console se o cloud-init falhar)
- Ao clonar via Ansible, as credenciais são substituídas de `local_users` em seu arquivo hosts:
 - Nome de usuário: Chave do primeiro usuário de `local_users`
 - Senha: Campo `password` do primeiro usuário (padrão para 'password' se não definido)
 - Chave SSH: Campo `public_key` do primeiro usuário
- `--vga std` garante que o console da 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 UI da Web

- Criar Nova VM → ID da VM 9000, Nome: ubuntu-2404-template
- SO: Carregar ISO do Ubuntu Server ou usar ISO existente
- Sistema: Padrão (Controlador SCSI: VirtIO SCSI)
- Discos: 32GB, Barramento: SCSI
- CPU: 2 núcleos
- Memória: 2048 MB
- Rede: VirtIO, bridge vmbr0
- Iniciar VM e instalar o Ubuntu Server

Passo 2: Dentro da VM - Limpar e preparar

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

# Limpar dados específicos da máquina
sudo cloud-init clean
sudo rm -f /etc/machine-id /var/lib/dbus/machine-id
sudo rm -f /etc/ssh/ssh_host_*
sudo truncate -s 0 /etc/hostname
sudo truncate -s 0 /etc/hosts

# Limpar histórico do bash e desligar
history -c
sudo poweroff
```

Passo 3: Adicionar Cloud-Init e Converter em Modelo

- Selecionar VM → Hardware → Adicionar → CloudInit Drive (selecionar armazenamento, por exemplo, local-lvm)
- Cloud-Init → Usuário: `omnitech`, Senha: `password`
- Hardware → Opções → Agente QEMU → Habilitar
- Clique com o botão direito na VM → Converter em Modelo

3. Baixar Modelo LXC (apenas para LXC)

```
# No shell do nó Proxmox:
pveam update
pveam download local ubuntu-24.04-standard_24.04-2_amd64.tar.zst
```

Configuração do Arquivo Hosts

Para Implantação de VM (proxmox.yml)

```
all:
  vars:
    proxmoxServers:
      pve-node-01:
        proxmoxServerAddress: 192.168.1.100
        proxmoxServerPort: 8006
        proxmoxRootPassword: YourPassword
        proxmoxApiTokenName: ansible
        proxmoxApiTokenSecret: "your-token-secret-uuid"
        proxmoxTemplateName: ubuntu-2404-template
        proxmoxTemplateId: 9000
        proxmoxNodeName: pve-node-01
        storage: local-lvm # opcional
      pve-node-02:
        # ... configuração do segundo nó

    # Credenciais do usuário - o primeiro usuário é usado para o
    # cloud-init da VM
    local_users:
      admin_user:
        name: Admin User
        public_key: "ssh-rsa AAAA..."
        password: "optional-password" # padrão para 'password' se
        não definido

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

Para Implantação de LXC (proxmox_lxc.yml)

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

    # Credenciais do usuário - o primeiro usuário é usado para
    acesso inicial à VM/LXC
    local_users:
      admin_user:
        name: Admin User
        public_key: "ssh-rsa AAAA..."
        password: "optional-password" # padrão para 'password' se
        não definido

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

Uso

Implantar VMs

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

Implantar Contêineres LXC

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

Deletar VMs/LXCs

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

Comportamento

proxmox.yml

- Verifica se uma VM com o mesmo nome já existe no Proxmox
- Distribui VMs entre os nós usando round-robin
- Clona do modelo
- Configura IP estático, tags e cloud-init
- **Define credenciais do usuário do cloud-init a partir da primeira entrada de `local_users`**
- Suporta tagging VLAN

proxmox_lxc.yml

- Verifica se o contêiner não existe pelo nome ou IP

- Distribui LXCs entre os nós usando round-robin
- Cria contêiner com IP estático
- **Cria automaticamente a primeira conta de `local_users` com acesso sudo e chave SSH**
- Configura netplan para rede
- Inicia contêineres automaticamente
- Exclui hosts UPF

proxmox_delete.yml

- Para e deleta VM/LXC correspondente ao nome do host do inventário
- Pesquisa em todos os nós configurados
- Para forçadamente após 20 segundos

Distribuição e Tagging de VM/LXC

Distribuição Round-Robin

VMs e LXCs são automaticamente distribuídos entre os nós do Proxmox usando lógica de round-robin (módulo):

Exemplo com 3 hipervisores e 5 MMEs:

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

Como funciona:

1. O playbook identifica o grupo de funções do host (por exemplo, `mme`, `sgw`, `hss`)
2. Calcula o índice do host dentro desse grupo (base 0)
3. Usa a operação de módulo: `host_index % number_of_nodes`

4. Seleciona o hipervisor com base no resultado

Configuração:

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

# Para LXCs (proxmox_lxc.yml) - liste vários nós
proxmoxNodeName: ['pve-node-01', 'pve-node-02', 'pve-node-03']
```

Tagging Automático

VMs e LXC são automaticamente marcados com:

- **Nomes de Função/Grupo:** Todos os grupos Ansible aos quais o host pertence
- **Nome do Site:** A variável `site_name`

Exemplo:

```
site_name: "melbourne-prod"

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

Resultado: VM/LXC marcada com: `mme`, `melbourne-prod`

As tags são visíveis na interface do Proxmox e úteis para filtragem/organização.

Substituições por Host

Substitua os padrões em hosts específicos:

```
hosts:
  high-spec-host:
    ansible_host: 192.168.1.50
    gateway: 192.168.1.1
    proxmoxLxcCores: 8          # substituição de núcleos
    proxmoxLxcMemoryMb: 16384  # substituição de memória
    proxmoxLxcDiskSizeGb: 100  # substituição de disco
```

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
<code>health_check.yml</code>	Gerar relatório de saúde abrangente para todos os serviços
<code>restore_hss.yml</code>	Restaurar banco de dados HSS e/ou configuração a partir do backup
<code>ip_plan_generator.yml</code>	Gerar documentação de rede com diagramas Mermaid
<code>get_ports.yml</code>	Auditar portas abertas e serviços em escuta em todos os hosts
<code>getLocalCapture.yml</code>	Recuperar arquivos de captura de pacotes dos hosts
<code>delete_local_user.yml</code>	Remover uma conta de usuário local de todos os hosts
<code>updateMtu.yml</code>	Definir MTU para 9000 (frames jumbo) nas interfaces de rede
<code>systemctl status.yml</code>	Verificar o status do serviço nos componentes EPC

Verificação de Saude

Arquivo: util_playbooks/health_check.yml

Gera um relatório de saúde HTML abrangente 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 da CPU
OCS	Status de replicação do KeyDB

Restauração do HSS

Arquivo: util_playbooks/restore_hss.yml

Restaura o OmniHSS a partir de arquivos de backup. Suporta a restauração apenas do banco de dados, apenas da configuração ou de ambos.

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

Formatos de Arquivo de Backup

Tipo	Padrão de Nome de Arquivo	Conteúdos
Banco de dados	<code>hss_dump_<hostname>_<timestamp>.sql</code>	Dump MySQL do banco de dados <code>omnihss</code>
Configuração	<code>hss_<hostname>_<timestamp>.tar.gz</code>	Arquivo do diretório <code>/etc/omnihss</code>

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 na 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 do <code>ansible_host</code> 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
```

Prompt: Digite o nome de usuário a ser excluído quando solicitado.

Configuração de MTU

Arquivo: util_playbooks/updateMtu.yml

Define o MTU para 9000 (frames jumbo) na interface `ens160` em todos os hosts.

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

Nota: Este playbook é codificado para a interface `ens160`. Modifique o playbook se seu ambiente usar nomes de interface diferentes.

Executando Playbooks de Utilidade

Sintaxe Básica

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

Opções Comuns

Opção	Descrição
<code>-i <inventory></code>	Especificar arquivo de inventário
<code>--limit <hosts></code>	Limitar a hosts ou grupos específicos
<code>-v</code> / <code>-vv</code> / <code>-vvv</code>	Aumentar a verbosidade
<code>--check</code>	Execução de teste (mostrar o que mudaria)
<code>--diff</code>	Mostrar diferenças de arquivos

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
```

