

5G 5GC

5G SA services/5gc.yml 5gc_nf NF NF NF NF

11

OmniCore 5gc_nf 5GC NF NF NF
nf_package Debian systemd Jinja2 NF

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

--	--	--	--

```
nf systemd
roles/5gc_nf/templates/<nf_package>.j2
```

NF	nf_package	名前	ファイル名	機能
コア機能	omninrf	nrf	omninrf.j2	コア / NF- 
コア機能	omniamf	amf	omniamf.j2	N1/N2コア機能
コア機能	omniausf	ausf	omniausf.j2	5G-AKA / EAP-AKA'コア
コア機能	omnibsf	bsf	omnibsf.j2	PDUセッションPCF
コア機能	omnichf	chf	omnichf.j2	コア/コア
コア機能	omnissf	nssf	omnissf.j2	NSSAI / コア
コア機能	omnipcf	pcf	omnipcf.j2	コアQoS
コア機能	omniscp	scp	omniscp.j2	SBIコア / コア
コア機能	omnismf	smf	omnismf.j2	PDUセッションUPF/N4
コア機能	omniudm	udm	omniudm.j2	コアSUCI
コア機能	omniudr	udr	omniudr.j2	UDM/PCFコア

5gc_nfインストール

5GC NFインストール手順

1. `group_vars_omnicore` ファイルに `<inventory>/group_vars/` を追加する
2. `license_server_api_urls` に `license_server` を追加し
`https://<host>:8443/api` を指定する
3. `lsb-release`、`curl`、`gpg`、`unzip`、`libglu1-mesa`、`libsctp1` をインストールする
4. **OmniTouch APT** をインストールし `nf_version` を指定する

5. `templates/<nf_package>.j2` `/etc/<nf_package>/runtime-<deb_version>.exs`
6. `runtime.exs` → `/opt/<nf_package>/releases/*`
7. **NF**

Debian `runtime-<deb_version>.exs` `runtime.exs`

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

`nf_version`

UDM HNET SUCI

UDM `nf_package: omniudm` SUCI TS 33.501 §6.12 UE SUCI SUPI USIM

- Ansible `group_vars/hnet_udm/` `<curve>-<id>.key` PEM `<curve>` `curve25519` Profile A, X25519 `secp256r1` Profile B, `prime256v1` `<id>`
- `curve25519-1.key`
- `curve25519-*.key` 32 `.pub` USIM `.pub` `*.key` UDM `/etc/omniudm/hnet/` `0600` `root`

USIM SUCIs `.pub` USIM SIM HNET

8

□□□□

- □□□□ - □□□□□□□□□□□□
- □□□□□□ - □□CS□□BSC□MSC□SS7□
- □□□□□□ - □□NF□□□□□HNET□□
- □□□□ - □□□□□□□□□□
- **OmniCore**□□□ <https://docs.omnitech.com.au/docs/repos/OmniCore> - NF□□□□□□□□

OmniCore IP

OmniCore IP IP IP

IP

/24

OmniCore

1. - /24
2. - /24
3. **IMS** - /24
4. **IMS** - /24

OmniCore

-
- MME HSS
-
- **IP** RFC 1918 IP

IP

Network

1. Core (10.179.1.0/24)

Core Network

Core

- OmniMME (MME)
- OmniSGW (SGW)
- OmniPGW-C (PDN GW)
- OmniUPF/PGW-U (UPF / PDN GW)

Core 10.179.1.0/24

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

2. Edge (10.179.1.0/24)

Edge Diameter Network

Edge

- OmniHSS (HSS)
- OmniCharge OCS (OCS)
- OmniHSS PCRF (PCRF)
- OmniDRA DRA (Diameter DRA)
- DNS 10.179.1.0/24 DNS - 10.179.1.0/24 DNS 10.179.1.0/24
- TAP3/CDR 10.179.1.0/24
- 10.179.1.0/24/OAM

- SIP
-
- RAN
- Omnitouch CBC () -

10.179.2.0/24

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

3. IMS (/24)

IMS SIP

- OmniCSCF S-CSCF ()
- OmniCSCF I-CSCF ()
- OmniTAS (/)
- OmniMessage (SMPP IMS)

10.179.3.0/24

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

4. IMS 网络 (10.179.4.0/24)

网络 网络 IMS 网络 DNS 网络 SS7/网络

网络

- OmniCSCF P-CSCF (网络)
- XCAP 网络
- 网络
- 网络 DNS
- OmniSS7 STP (SS7 网络)
- OmniSS7 HLR (网络) - 网络 2G/3G
- OmniSS7 IP-SM-GW (MAP SMSs)
- OmniSS7 CAMEL 网络
- MSC (网络)

网络 SS7/HLR 网络 SIGTRAN/M3UA 网络 2G/3G 网络
网络 IMS SIP

网络 10.179.4.0/24

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

网络

OmniCore 网络

VLAN 400: 10.x.1.0/24 (HSS)
VLAN 401: 10.x.2.0/24 (HSS)
VLAN 402: 10.x.3.0/24 (IMS)
VLAN 403: 10.x.4.0/24 (IMS)

Network IP

OmniCore Network IP

- **DRA** -
- **SGW/PGW** - GTP
- **ePDG** - WiFi UE IPsec
- **SMSC** - SMS SMPP
- **P-CSCF** - UE SIP

Network IP

IP `ansible_host` IP

IP **SGW/PGW**

```

sgw:
  hosts:
    # 00 SGW 000000
    customer-site-sgw01:
      ansible_host: 10.4.1.25
      gateway: 10.4.1.1
      host_vm_network: "v400-omni-packet-core"

    # 0000 IP 000 SGW
    customer-site-roaming-sgw01:
      ansible_host: 203.0.113.10
      gateway: 203.0.113.9
      netmask: 255.255.255.248      # /29 00
      host_vm_network: "498-public-servers"
      in_pool: False
      cdrs_enabled: True

smf: # PGWs
  hosts:
    # 0000 IP 000 PGW
    customer-site-roaming-pgw01:
      ansible_host: 203.0.113.20
      gateway: 203.0.113.17
      netmask: 255.255.255.240      # /28 00
      host_vm_network: "497-public-services-LTE"
      in_pool: False
      ip_pools:
        - '100.64.24.0/22'

```

00000000 IP 0 DRA

```

dra:
  hosts:
    customer-site-dra01:
      ansible_host: 198.51.100.50
      gateway: 198.51.100.49
      netmask: 255.255.255.240      # /28 00
      host_vm_network: "497-public-services-LTE"

```

00000000 IP 0 ePDG

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

IP

IP

- SGW GTP
- SGW
- PGW-C SGW

OmniCore - IP

Omnitouch 与 Ansible 结合

结合

结合

Omnitouch 结合 Ansible 可以实现对网络设备的自动化配置管理，提高配置效率，减少人为错误。支持4G/5G网络设备的配置管理。 Ansible 可以实现对网络设备的自动化配置管理。

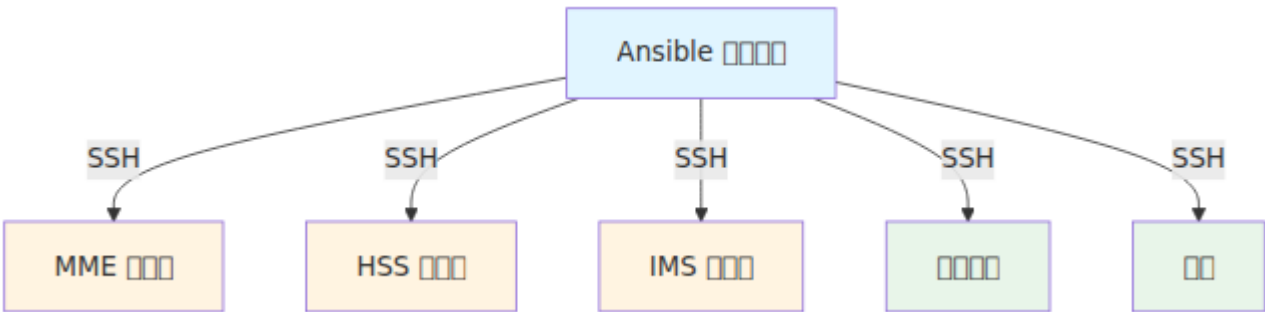
结合 Ansible

Ansible 可以实现对网络设备的自动化配置管理。

- 配置管理
- 设备管理
- 网络配置
- 设备维护

Ansible 可以实现对网络设备的自动化配置管理 - 网络设备 配置Ansible 可以实现对网络设备的自动化配置管理。

Omnitouch 结合 Ansible



结合

1. 配置管理

이 단락을 완료하면 다음 단락을 진행합니다

- 네트워크 구성
- IP 주소 할당
- 라우팅
- 방화벽 설정

이 단락을 완료하면 다음 단락을 진행합니다

이 단락을 완료하면

2. 준비

이 단락을 완료하면 다음 단락을 진행합니다

- 네트워크 구성
- IP 주소 할당
- 라우팅
- 방화벽 설정

OmniCore 구성을 완료하면 `omnihss`, `omnisgw`, `omnipgw`, `omnidra` 구성

ONS 구성을 완료하면 다음 단락을 진행합니다

3. 설치

이 단락을 완료하면 다음 단락을 진행합니다

```
- name: OmniCore MME
  hosts: mme
  roles:
    - omnimme
```

이 단락을 완료하면 다음 단락을 진행합니다

이 단락을 완료하면 다음 단락을 진행합니다 `services/epc.yml` 파일을 `import_playbook`로 `common.yml` 파일을 포함합니다

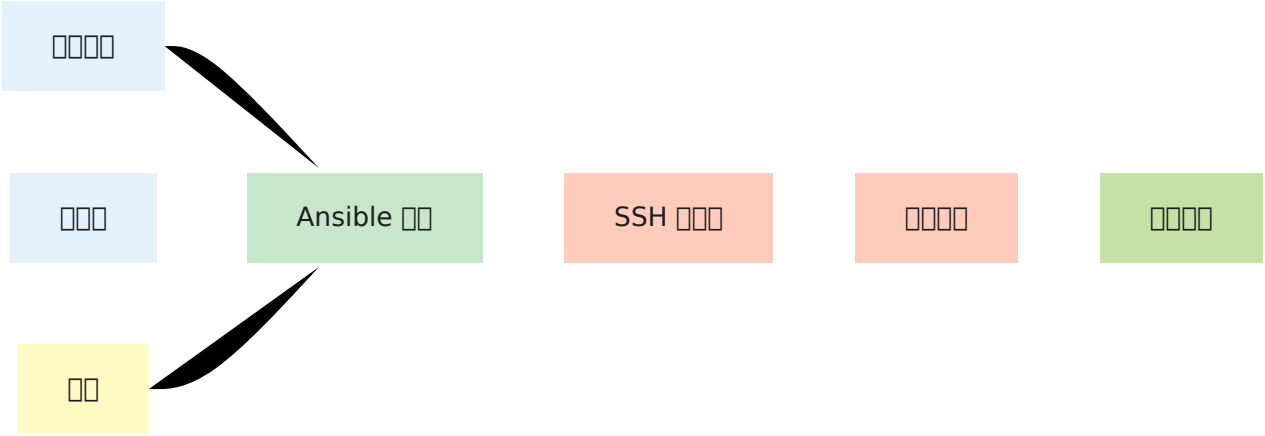
```
- name: 000000
  import_playbook: common.yml

- name: 00 MME 00
  import_playbook: omnimme.yml
```

4.

□ □ □

100/100



□ □ □ □

1. □□□□□□

[illegible]

□□□□□ □□□□□□□□□□□ IP □□□□□□□□□□□□ IP □□□□□□□□□□□□

Proxmox VM/LXC VM

□□□□□□□□ □ □□□□

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

2. 設定

group_vars 設定

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

設定 - 設定

3. 実行

実行

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

4. 結果

Ansible 結果

- 設定/設定 Proxmox/VMware 設定
- 設定
- APT 設定
- 設定
- 設定
- 設定
- 設定
- 設定

OmniCore

OmniCore 4G/5G 組件

- **OmniHSS** - HSS 組件
- **OmniSGW** - S-GW 組件
- **OmniPGW** - P-GW 組件
- **OmniUPF** - UPF 組件
- **OmniDRA** - Diameter 組件
- **OmniTWAG** - WLAN 組件

詳情請參閱 <https://docs.omnitech.com.au/docs/repos/OmniCore>

OmniCall 組件

- **OmniCall CSCF** - P-CSCF/I-CSCF/S-CSCF 組件
- **OmniTAS** - IMS VoLTE/VoNR 組件
- **OmniMessage** - SMS-C 組件
- **OmniMessage SMPP** - SMPP 組件
- **OmniSS7** - SS7 STP/HLR/CAMEL 組件
- **VisualVoicemail** - 語音郵件組件

詳情請參閱 <https://docs.omnitech.com.au/docs/repos/OmniCall>

OmniCharge/OmniCRM

- **CRM** 組件 - CRM 組件

詳情請參閱 <https://docs.omnitech.com.au/docs/repos/OmniCharge>

組件

- **DNS** - DNS 組件
- **組件** - 組件
- **組件** - Prometheus/Grafana

📄📄📄📄📄

📄📄📄📄

📄📄📄📄📄📄📄📄

📄📄📄 **APT** 📄📄

📄📄 Omnitouch 📄📄 Debian 📄📄 `.deb` 📄📄📄📄

- 📄📄📄 CI/CD 📄📄📄📄
- 📄📄📄📄
- 📄📄📄📄📄📄

APT 📄📄📄

📄📄📄📄📄

1. 📄📄 **APT** 📄📄 - 📄📄📄📄📄📄📄📄📄📄📄
2. 📄📄📄 - 📄📄📄 Omnitouch 📄📄📄📄📄

📄📄📄📄📄 `omnicore` `.deb`📄📄 `packaging/` 📄📄📄📄📄📄📄📄📄📄📄📄📄📄📄📄📄📄

📄📄**APT** 📄📄📄

📄📄📄📄

📄📄 Omnitouch 📄📄📄📄📄📄📄📄📄📄📄📄📄📄📄📄

- 📄📄📄📄📄📄📄📄
- 📄📄📄📄📄/📄📄📄
- 📄📄📄📄📄📄📄📄📄📄

📄📄📄📄📄📄

Ansible 入門

Ansible

Ansible について

- 簡単にインストール
- 簡単に操作
- 簡単に設定
- 簡単に実行

インストール

Ansible のインストール方法

基本操作

Ansible の基本操作

- 接続先を指定
- 実行コマンド
- 実行結果の確認

変数とタスク

変数 `group_vars` を利用してタスクを実行

まとめ

Ansible の基本操作と変数の使い方

環境構築

前提条件

Ansible をインストールする前に **uv** という Python のパッケージ管理ツール **uv** を

1. uv のインストール

uv をインストールするには **uv** をインストールする

```
pipx install uv # pipx を利用してインストール
curl -LsSf https://astral.sh/uv/install.sh | sh # シェルスクリプトを実行してインストール
```

2. 仮想環境の作成

仮想環境を作成する

```
uv sync
```

作成された `.venv/` フォルダに Ansible と Omnitouch の依存関係を Python の `uv.lock` ファイルで

3. 仮想環境のアクティベーション

Ansible のインストールを実行する前に

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

4. Ansible のインストール

Ansible のインストールには `requirements.yml` ファイル

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

5. Ansible の実行

如何安装 Ansible 呢

```
ansible --version
```

如何安装 `deactivate` 呢

如何

1. 安装 `deactivate` 包
2. 安装 `deactivate` 包
3. 安装 `APT` 包
4. 安装 `deactivate` 包
5. 安装

如何

- `IP` 包 - 安装 `IP` 包
- `deactivate` 包 - 安装 `deactivate` 包
- `APT` 包 - 安装 `APT` 包
- `deactivate` 包 - 安装 `deactivate` 包
- `deactivate` 包 - 安装 `deactivate` 包
- `deactivate` 包 - 安装 `deactivate` 包
- `deactivate` 包 - 安装 `deactivate` 包

APT 部署架构图

简介

Omnitouch APT 部署架构图如下所示：

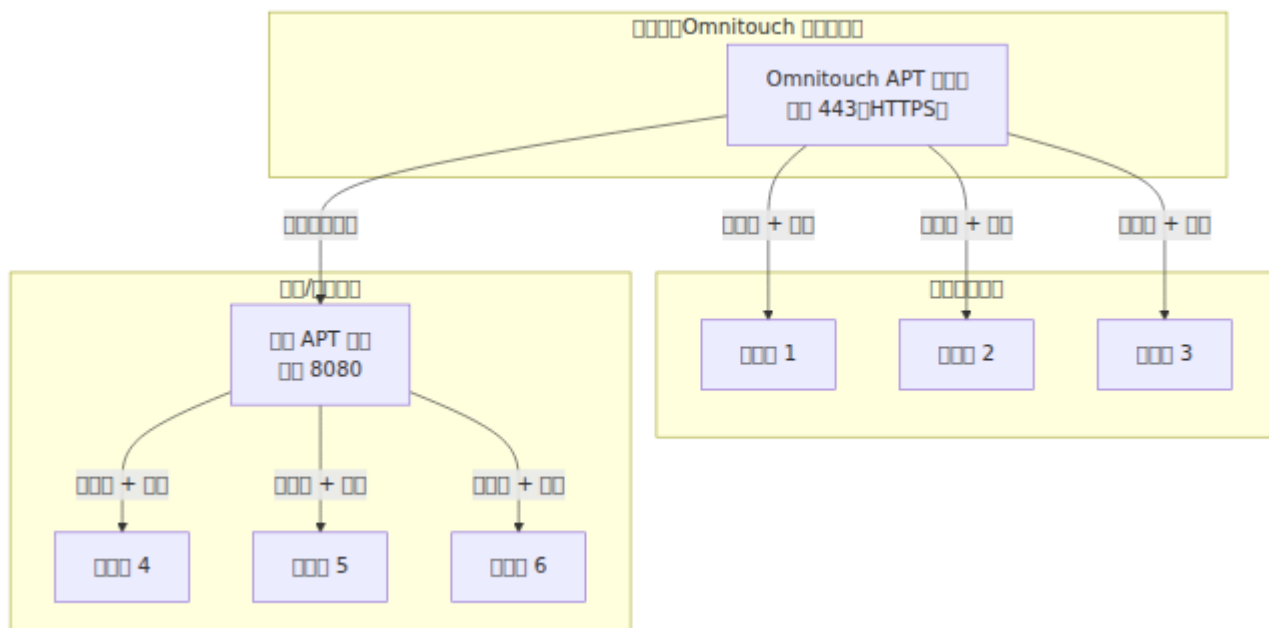
1. **APT** 部署在 `apt install` 的 Debian 系统
2. 部署在 Prometheus 监控系统

APT 部署在 **OmniCore** 部署在 `omnicore` 的 `/pool/main/o/omnicore/` 部署在 Ansible 部署在

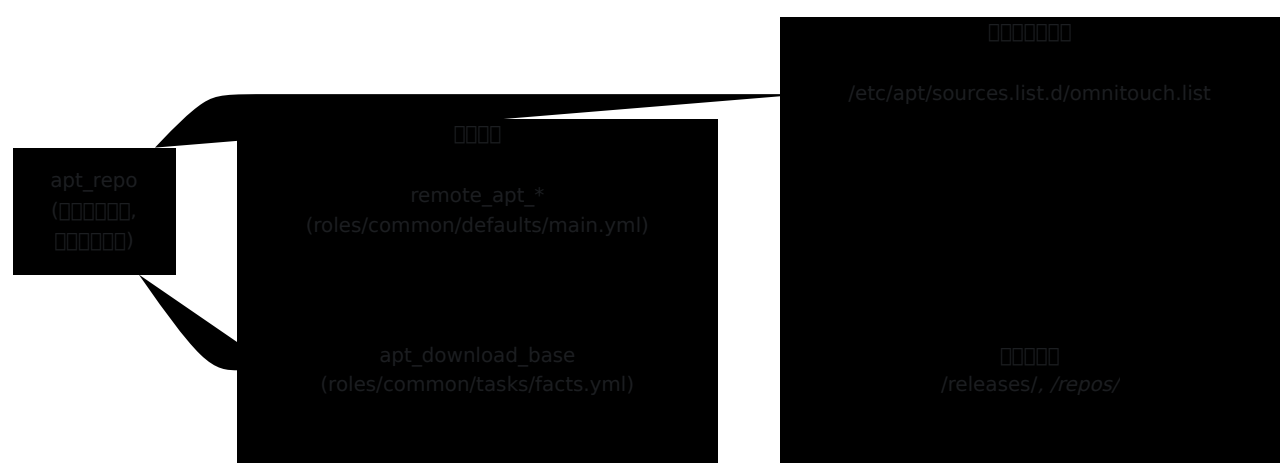
部署在

1. 部署在 Omnitouch APT 部署在 `apt.<region>.omnitou.ch`
2. 部署在 Omnitouch 部署在

部署



</



Table

Role	File	Description
apt_repo		Role that defines the APT repository
remote_apr_*	roles/common/defaults/main.yml	Role that defines the APT repository URL and the APT repository name. It defines the APT repository URL as apt.us-fl.omnitou.ch / 443 / https and the APT repository name as apt_repo.
apt_download_base	roles/common/tasks/facts.yml	Role that defines the APT repository URL and the APT repository name. It defines the APT repository URL as apt.us-fl.omnitou.ch / 443 / https and the APT repository name as apt_repo.

Role that defines the APT repository URL and the APT repository name. It defines the APT repository URL as apt.us-fl.omnitou.ch / 443 / https and the APT repository name as apt_repo.

Role that defines the APT repository URL and the APT repository name. It defines the APT repository URL as apt.us-fl.omnitou.ch / 443 / https and the APT repository name as apt_repo.

443 → https → http apt_repo.apt_repo_port

- 443 HTTPS
- 8080 HTTP

	APT	apt_download_base
use_apt_cache: true	http://<cache>:8080/...	http://<cache>:8080/...
use_apt_cache: false	https://<user>:<pass>@<server>/...	https://<user>:<pass>@<server>:443/...

Omnitouch APT

IP Omnitouch APT Omnitouch

- HTTP
- APT FQDN apt.<region>.omnitou.ch

Omnitouch IP

項目	値
IPv4	144.79.167.0/24
IPv4	160.22.43.0/24
IPv6	2001:df3:dec0::/48
ASN	AS152894

項目 Omnitouch 項目項目項目

項目	項目	項目	項目
APT 項目	443	TCP	項目項目HTTPS
APT 項目	53	TCP/UDP	APT FQDN 項目 DNS 項目
項目項目項目	123	UDP	項目項目項目 NTP 項目項目
項目項目項目	53	TCP/UDP	項目項目項目 DNS 項目

項目 HTTPS項目TCP/443項目NTP項目UDP/123項目 DNS項目TCP+UDP/53項目項目項目 Omnitouch IP 項目項目

変数	参照	値
remote_apt_server	apt_repo.apt_server	apt.us-fl.omnitou.ch
remote_apt_port	apt_repo.apt_repo_port	443
remote_apt_protocol	ポート 443 → https	https
remote_apt_user	apt_repo.apt_repo_username	" "
remote_apt_password	apt_repo.apt_repo_password	" "

変数

変数	型	値	説明
use_apt_cache	bool	-	キャッシュを使用するかどうか (false)

URL 変数

APT 変数 `/etc/apt/sources.list.d/omnitouch.list`

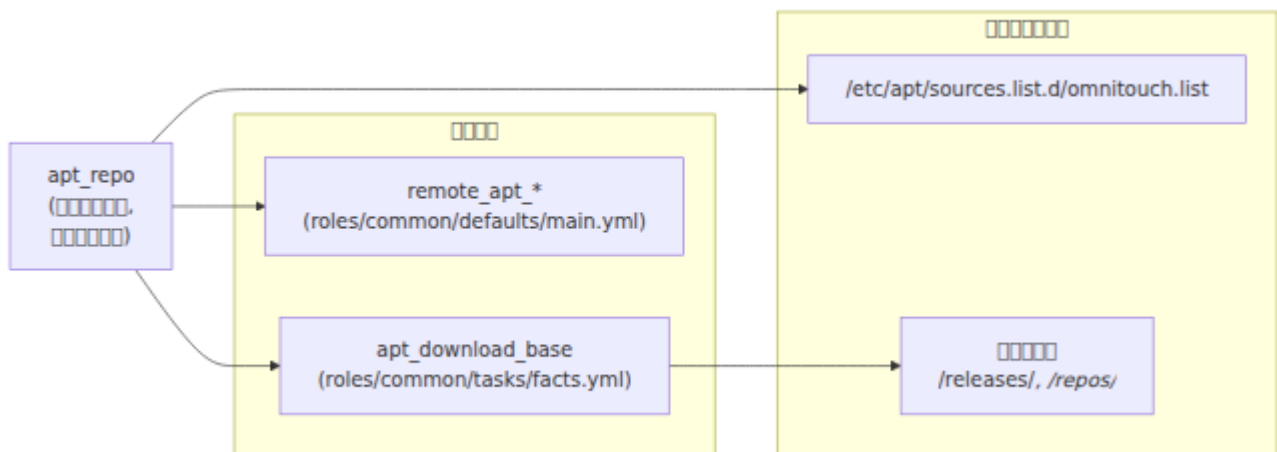
```
deb [trusted=yes] https://{username}:{password}@{apt_server}/noble main
```

Ansible `get_url` 変数 `apt_download_base`

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

ポート 443 が URL のプロトコルに HTTPS ではなく HTTP の場合、ポート 443 は

概要



このスクリプトは HTTP を介して APT リポジトリを Ubuntu リポジトリ Omnitech リポジトリに追加し、Ubuntu リポジトリ

必要なGPG

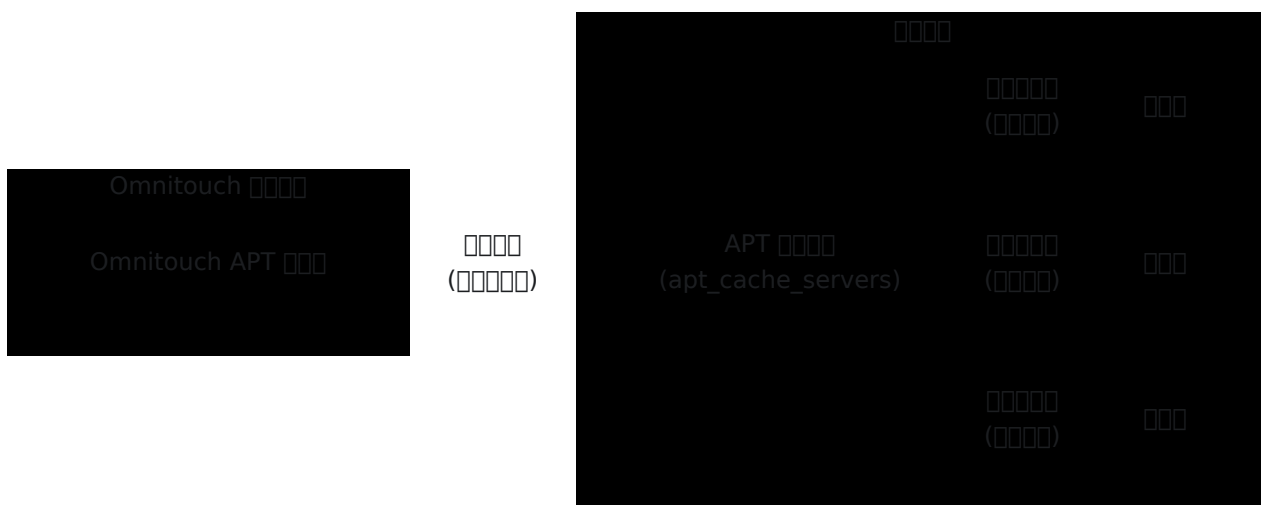
Omnitech APT リポジトリの GPG キーは `/omnitech.gpg` にあり、`common` リポジトリの `/etc/apt/trusted.gpg.d/omnitech.gpg` に追加します。`apt-get update` を実行するときに `NO_PUBKEY` エラーが発生する場合は

次のように GPG キーを信頼するように設定し、再度実行してください。

2. リポジトリの追加

このスクリプトは APT リポジトリ Omnitech リポジトリを

❏❏



❏❏

remote_apt_* Omnitouch

```
apt_cache_servers:
  hosts:
    example-apt-cache:
      ansible_host: 192.168.1.100
      gateway: 192.168.1.1
  vars:
    # Omnitouch
    remote_apt_server: "apt.<region>.omnitou.ch"
    remote_apt_port: 443
    remote_apt_protocol: "https"
    remote_apt_user: "your-username"
    remote_apt_password: "your-password"

all:
  vars:
    # use_apt_cache: true # apt_cache_servers
    # apt_repo.apt_server: 192.168.1.100
```

❏❏❏❏

- 192.168.1.100 remote_ap* HTTPS Omnitouch
- apt_repo.ap* "192.168.1.100"

APT apt_repo

apt_repo.ap*server				IP apt_cache_servers
apt_repo.ap*repo_username			-	
apt_repo.ap*repo_password			-	
apt_repo.ap*repo_port			8080	apt_cache_port

remote_ap*

Omnitouch remote_ap*

項目	項目	項目	項目	項目
remote_apt_server	項目	項目	-	項目 Omnitouch APT 項目
remote_apt_port	項目	項目	443	項目 Omnitouch APT 項目項目
remote_apt_protocol	項目	項目	https	項目項目項目
remote_apt_user	項目	項目	-	項目 Omnitouch 項目項目
remote_apt_password	項目	項目	-	項目 Omnitouch 項目項目

項目

項目	項目 項目	項目 項目	項目	項目
use_apt_cache	項目 項目	項目	true	項目 apt_cache_servers 項目項目項目 true

項目項目 apt_cache_port 項目項目項目項目項目項目項目 apt_repo.apt_repo_port 項目項目
項目項目 8080項目

URL 項目項目項目項目

APT 項目項目項目項目 /etc/apt/sources.list.d/omnitouch.list項目

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

項目項目項目項目 apt_download_base 項目項目

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

項目項目項目項目項目項目項目 [trusted=yes] APT 項目項目

環境

1. 仮想マシン 50+ GB の LXC 環境
2. 仮想マシン

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

3. 環境 `http://192.168.1.100:8080/` 環境

環境

環境 Omnitouch APT 環境 環境

- **APT** 環境 `/dists/` + `/pool/` 環境 Omnitouch 環境 Ubuntu 環境 Debian 環境
`curl` 環境 `Release` 環境 `Packages.gz` 環境
環境/SHA256/環境 SHA256 環境
SHA256 環境
- 環境 `/releases/` 環境 `/repos/` 環境 Debian 環境 `wget --recursive --timestamping` 環境 `Last-Modified` 環境

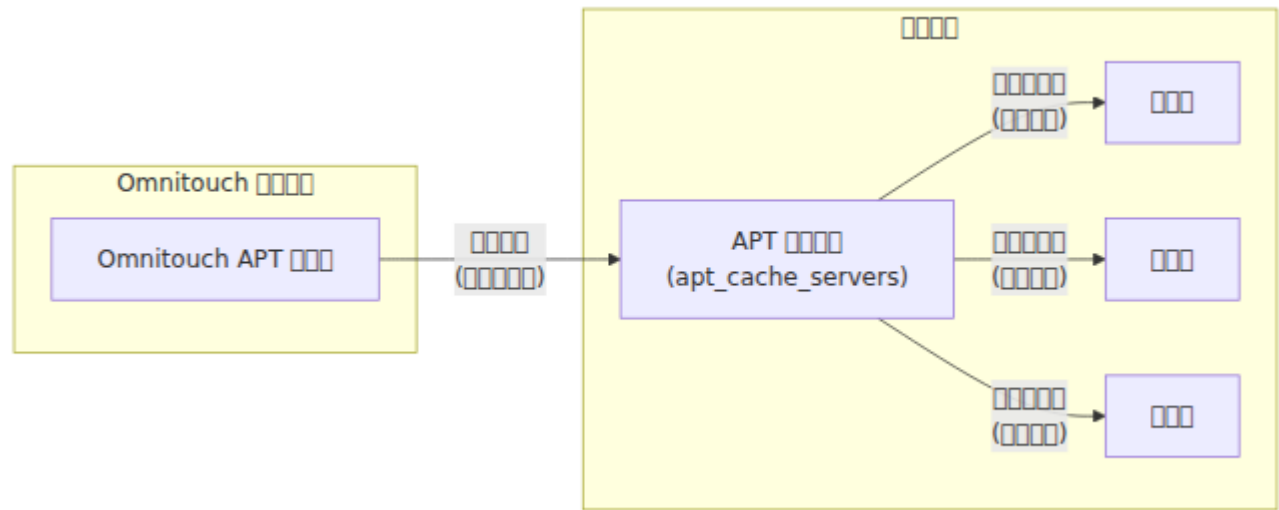
Omnitouch APT 環境				
Omnitouch .deb 環境 /pool/main/	Ubuntu 環境 + 環境 /noble/pool/main/	GitHub 環境 /releases/	環境 /repos/	APT 環境 /dists/
curl + index/SHA256	curl + index/SHA256	wget --recursive --timestamping	wget --recursive --timestamping	curl + index/SHA256
環境				
Omnitouch .deb 環境	Ubuntu 環境 + 環境	GitHub 環境	環境	APT 環境

環境

パス	説明
<code>/dists/<distro>/</code>	APT リポジトリのディレクトリ
<code>/pool/main/</code>	Omnitouch の .deb パッケージ
<code>/<distro>/pool/main/</code>	Ubuntu のディレクトリ
<code>/releases/</code>	GitHub の Prometheus, Grafana, Zabbix のリポジトリ
<code>/repos/</code>	Erlang, Elixir, CGrateS_UI のリポジトリ

リポジトリのディレクトリ

ディレクトリ



リポジトリのディレクトリ HTTP リポジトリのディレクトリ APT リポジトリ `curl` Debian リポジトリ `Release + Packages.gz` リポジトリのディレクトリ SHA256 リポジトリのディレクトリ `/releases/` `/repos/` リポジトリ `wget --recursive --timestamping` リポジトリのディレクトリ

ディレクトリ

ディレクトリ `apt_cache_servers` リポジトリのディレクトリ

1. `use_apt_cache: true`
2. `ansible_host` IP `apt_repo.apt_server`

```
apt_cache_servers:
  hosts:
    apt-cache-01:
      ansible_host: 192.168.1.100
      gateway: 192.168.1.1
  vars:
    # Omnitouch
    remote_apt_server: "apt.<region>.omnitou.ch"
    remote_apt_user: "your-username"
    remote_apt_password: "your-password"
```

- `use_apt_cache: true`
- `apt_repo.apt_server: "192.168.1.100"`
- `http://192.168.1.100:8080/`
- `https://your-username:your-password@apt.<region>.omnitou.ch/`

```
all:
  vars:
    use_apt_cache: false #
  apt_repo:
    apt_server: "apt.<region>.omnitou.ch"
    apt_repo_username: "user"
    apt_repo_password: "pass"
```

0000

00 100000 **APT** 000000000

0000000 APT 0000000000000000 apt_repo remote_ap*_ 00000000

```
all:
  vars:
    use_apt_cache: false

    # APT 0000000000000000
    apt_repo:
      apt_server: "apt.<region>.omnitou.ch"
      apt_repo_username: "user"
      apt_repo_password: "pass"
```

000 000000 deb [trusted=yes] https://user:pass@apt.
<region>.omnitou.ch/ noble main

00 2000000000 **APT** 0000000000

0000000000000000 Ansible 00/000

```

apt_cache_servers:
  hosts:
    cache-server:
      ansible_host: 192.168.1.100
      gateway: 192.168.1.1
  vars:
    # 設定するAPTサーバーのリスト
    remote_apt_server: "apt.<region>.omnitou.ch"
    remote_apt_port: 443
    remote_apt_protocol: "https"
    remote_apt_user: "user"
    remote_apt_password: "pass"

# 全ホストに vars を適用
# apt_cache_servers を適用

```

例

- 設定ファイルに `https://user:pass@apt.<region>.omnitou.ch/` を追加
- 設定ファイルに `deb [trusted=yes] http://192.168.1.100:8080/noble noble main` を追加

3. APT をインストール

Ansible を使って APT をインストール

```

all:
  vars:
    use_apt_cache: true

    # 設定するAPTサーバーのリスト
    apt_repo:
      apt_server: "192.168.1.100" # 設定する IP
      apt_repo_port: 8080 # 設定する 8080 ポート

# apt_cache_servers を適用
# remote_apt_* を適用

```

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

main

```

# APT
apt_cache_servers:
  hosts:
    example-apt-cache:
      ansible_host: 10.179.1.114
      gateway: 10.179.1.1
      host_vm_network: "vmbr0"
      num_cpus: 4
      memory_mb: 16384
      proxmoxLxcDiskSizeGb: 120
  vars:
    #
    remote_apt_server: "apt.<region>.omnitou.ch"
    remote_apt_port: 443
    remote_apt_protocol: "https"
    remote_apt_user: "customer-username"
    remote_apt_password: "customer-secure-token"

#
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

#
all:
  vars:
    #

```

```
# - use_apt_cache: true[] apt_cache_servers []
# - apt_repo.apt_server: "10.179.1.114"[]
```

[]

1. [] 10.179.1.114 []

- [] vars: [] remote_apt_*
- [] https://customer-username:customer-secure-token@apt.<region>.omnitou.ch/ []
- [] nginx [] 8080 []

2. [] customer-hss01 [] customer-mme01 [] customer-dns01 []

- [] apt_cache_servers []
- [] use_apt_cache: true
- [] apt_repo.apt_server: "10.179.1.114"
- [] deb [trusted=yes] http://10.179.1.114:8080/noble noble main
- []

[]

[]

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

[] Omnitouch APT []

- [] Omnitouch []
- [] Ubuntu []
- [] GitHub []
- []

APT SHA256 Debian Release
 /releases/ /repos/ wget --timestamping

Omnitouch APT Omnitouch
 services/apt_cache.yml

APT 401

```
https://10.179.1.115/noble/dists/noble/main/binary-  
amd64/Packages 401 
```

- apt_repo all: vars: apt_cache_servers: vars:
-
- apt_repo_username apt_repo_password
- IP Omnitouch APT
-

1. apt_repo apt_cache_servers: vars: all: vars:
2. 8080
3. /etc/apt/sources.list.d/omnitouch.list
 - deb [trusted=yes] http://10.179.1.114:8080/noble noble main
 - deb [trusted=yes] https://user:pass@10.179.1.115/noble noble main

4. 設定する
5. 設定した IP 上の Omnitouch を設定する

設定するZabbix

Ansible の /releases/ を設定する

設定する

- apt_repo を設定する apt_repo を設定する
- remote_apr_user / remote_apr_password を設定する
- 443 の HTTPS を設定する HTTP

設定する

1. apt_repo.apr_repo_username / apt_repo.apr_repo_password を設定する Omnitouch を設定する
2. remote_apr_* を設定する
3. apt_download_base を設定する ansible-playbook ... -vvv を設定する get_url を設定する

設定する

設定する

設定する

- 設定する
- remote_apr_* を設定する
- Omnitouch を設定する

設定する

1. 443 の Omnitouch APT を設定する
 2. remote_apr_* を設定する
 3. 設定する
-

□□□□

- □□□□□: □□□□□□
- □□□□: □□□□□
- □□□□: □□□□□
- Proxmox □□: □□□□□□□ LXC □□□□

목차

목차

OmniCore 모니터링 아키텍처 소개 Grafana 모니터링

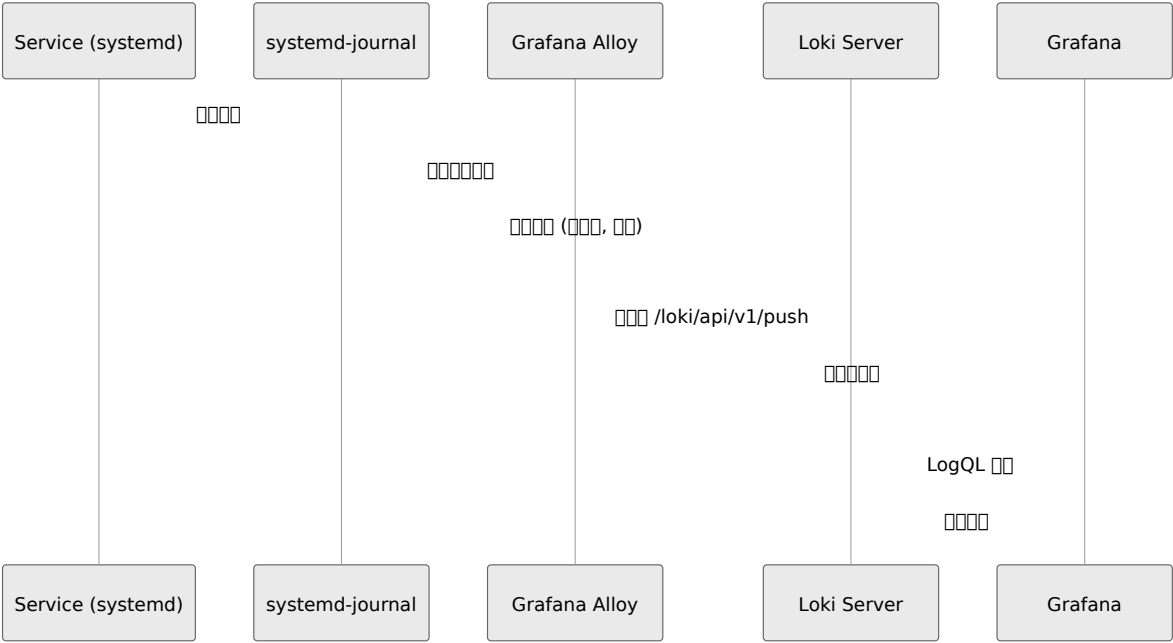
- **Grafana Alloy** 모니터링 에이전트
- **Grafana Loki** 로그 스토리지
- **Grafana** 모니터링 대시보드

목차



部署概要

ログフロー



ログフィールド

ログフィールドの定義

フィールド名	説明	値
job	ログジョブ名	customer-mme01
hostname	ホスト名 (job 単位で一意)	customer-mme01
component	コンポーネント名	mme, pcscf, ocs
unit	systemd ユニット	omnimme.service
level	ログレベル	info, warning, error
service	Syslog サービス	omnimme

目標

前提条件

Ansible 2.9.10 以上がインストールされていること

1. Ansible 2.9.10 以上がインストールされていること
2. Ansible 2.9.10 以上がインストールされていること

Ansible 2.9.10 以上がインストールされていること

作業内容

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

Ansible 2.9.10 以上がインストールされていること

- Ansible 2.9.10 以上がインストールされていること
- Ansible 2.9.10 以上がインストールされていること

作業内容

Loki 2.9.10 以上がインストールされていること

項目	内容	備考
Ansible 2.9.10 以上がインストールされていること	72 以上3 以上	Ansible 2.9.10 以上がインストールされていること

Ansible 2.9.10 以上がインストールされていること

```
all:
  vars:
    loki_retention_period: "168h" # 7 日間のログを保持
```

Alloy

Loki Alloy WAL

WAL	1	1

Loki

Grafana

--	--	--	--	--

☐❓❓☐	☐☐	☐☐
CSCF ☐☐	☐☐ → CSCF ☐☐	P-CSCF, S-CSCF, I-CSCF (OmniCSCF) ☐☐
MME ☐☐	☐☐ → MME ☐☐	MME ☐☐/☐☐☐☐☐☐☐
SGW ☐☐	☐☐ → SGW ☐☐	SGW ☐☐☐☐☐☐☐
PGW ☐☐	☐☐ → PGW ☐☐	PGW PDN ☐☐☐☐☐
HSS ☐☐	☐☐ → HSS ☐☐	HSS Diameter ☐☐☐☐☐☐☐
OmniMessage ☐☐	☐☐ → OmniMessage ☐☐	☐☐☐☐☐ SMPP ☐☐
OCS/CGateS ☐☐	☐☐ → OCS/CGateS ☐☐	☐☐☐☐☐ JSONRPC ☐☐
CGateS RPC ☐☐	☐☐ → CGateS RPC ☐☐	☐☐☐☐☐ RPC ☐☐/☐☐☐☐☐

--	--	--	--	--

□ □ □ □ □ □ □ □

- 0000000000000000
- 000000000000000000000000
- 00000000000P-CSCF, S-CSCF, I-CSCF
- 0000000000000000
- 0000000000010 0000

タスク

1. Grafanaに接続 `http://<monitoring-host>:3000`
 2. ログソース → ログ → ログ
 3. ログ ログ ログ
 4. ログログログ
-

タスク

LogQL ログ

Loki ログ LogQL ログログログ

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

タスク

ログログログログ

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

ログ **MME** ログ

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

ログ **CSCF** ログ

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

ログ **systemd** ログ

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

□□□□□

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

□□□□□□□

□□□□□□□□□□

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

CSCF □□□□□

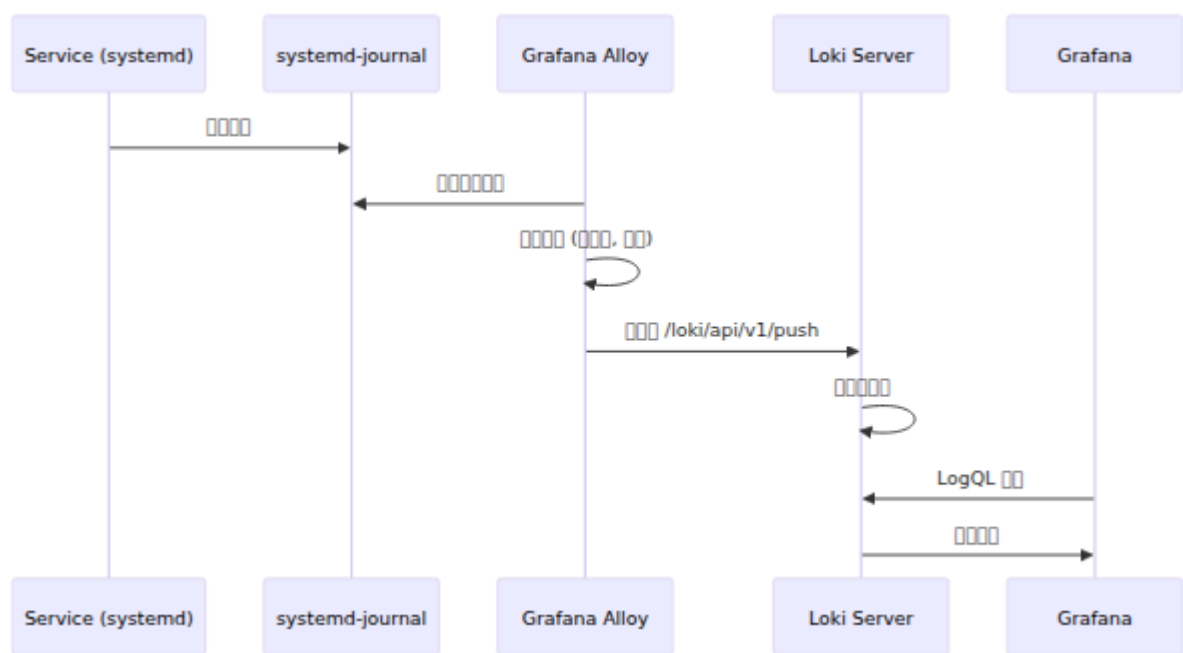
```
sum(rate({component=~"pcscf|scscf|icscf"} |~ "(?i)error" [5m]))
```

CGrateS JSONRPC □□□□

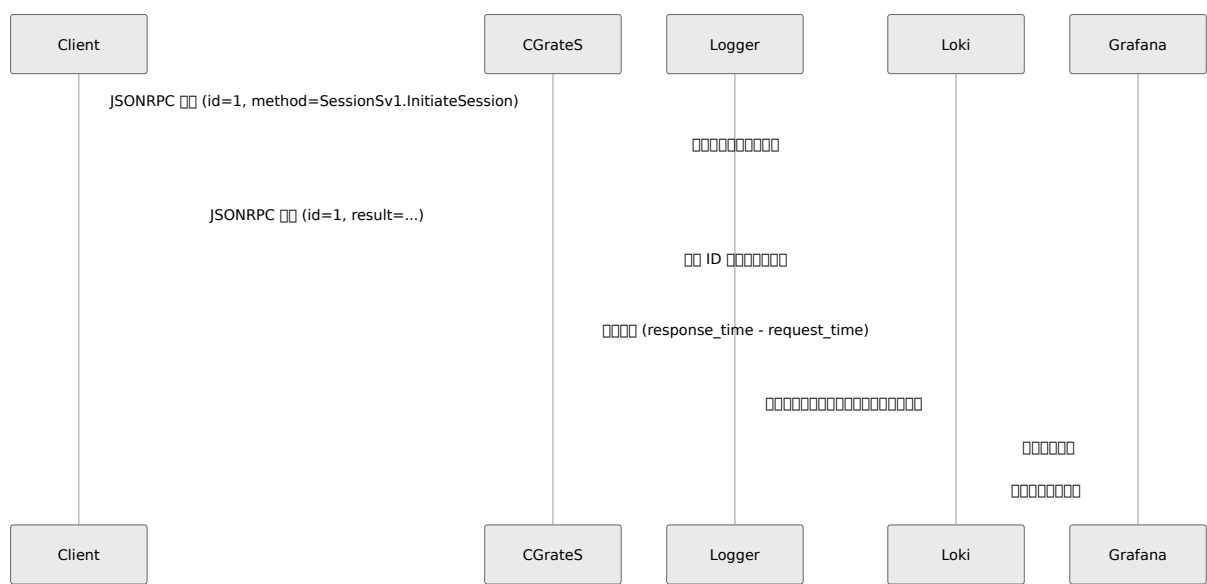
□□ OCS/CGrateS □□□JSONRPC □□□□□□□□□□□□/□□□□□□□□□□□□□□□□□□□□□□□□

JSONRPC □□□□ `cgrates` □□□□□□□□□□□□ (`/var/log/cgrates/jsonrpc.log`) □ Alloy
□ `ocs` □ `cgrates` □□□□□□□□

□□



□□/□□□□



□□□□

□□ CGrateS □□□□□□□□□□

Port	Service	Protocol	Notes
2012	rpc_json	TCP	JSONRPC over TCP
2080	http	HTTP	HTTP API endpoints: /metrics, /health

Configuration

- Prometheus scrape (GET /metrics)
- Health check (GET /health)
- Gzip compression

Log

JSONRPC logs: /var/log/cgrates/jsonrpc.log (format: timestamp / request details)

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

ログ

項目	説明	値
timestamp	ISO 8601 形式	2026-03-01T10:30:45.123456
port	CGrateS ポート	2012, 2080
service	サービス名	rpc_json, http
request_id	JSONRPC リクエスト ID	1, 42, (null)
method	JSONRPC メソッド	SessionSv1.InitiateSession
latency_ms	レイテンシー (ms)	2.45
status	ステータス	ok, error
error	エラーメッセージ	INSUFFICIENT_CREDIT
src	送信元 IP:ポート	10.10.1.50:45678
dst	宛先 IP:ポート	10.10.2.100:2012
request	送信された JSONRPC リクエストの JSON 形式	{"id":1,"method":"..."}
response	受信された JSONRPC レスポンスの JSON 形式	{"id":1,"result":{"..."}}

Grafana ログ

OCS / CGrateS ログは JSONRPC ログと一致する

ログは JSONRPC ログ

ログは JSONRPC ログと一致する

- 管理コンソール
- 監視ダッシュボード
- ログビュー

管理コンソール

JSONRPC 概要

管理コンソールは JSONRPC を利用して操作されます。

- 接続
- 認証
- 操作
- 監視

接続

管理コンソールは、管理コンソールと管理コンソール JSONRPC を利用して操作されます。
SessionSv1.InitiateSession

CGrateS RPC 概要

管理コンソールは CGraterS RPC を利用して操作されます。API を利用して操作されます。

接続

管理コンソール“管理コンソール 1234”

1. 管理コンソール → 管理コンソール → CGraterS RPC
2. 管理コンソール ID 1234
3. 管理コンソール 管理コンソール
4. RPC 管理コンソール 管理コンソール/管理コンソール

管理コンソールは管理コンソール管理コンソールCGraterS 管理コンソール管理コンソール

接続

□□□□□□

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

□□□□/□□□□□

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

□□□□

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

□□□□

□□□□□□□□□□□□□□□□

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

□□□□□□□□ > **100ms**□□

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

□□□□□□

CDR □□□

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

保存

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

保存

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

保存

JSONRPC サービス systemd サービス OCS/CGrateS サービス

確認

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

確認

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

確認

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

保存

JSONRPC サービス

JSONRPC サービス

確認

- 安裝依賴
- ngrep 安裝
- 安裝依賴
- Alloy 安裝

安裝

1. 安裝依賴

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

2. 安裝 ngrep

```
which ngrep
```

3. 安裝依賴

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

4. 安裝 Alloy

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

安裝

安裝JSONRPC 安裝

安裝

- 安裝JSONRPC 安裝
- 安裝JSONRPC 安裝

安裝

1. 安裝依賴

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

- 2. latency_ms
- 3.

-
-
- ID

- 1. 60
- 2. CGrateS
- 3. CGrateS

systemd Alloy

FreeSWITCH

/var/log/freeswitch/freeswitch.log	freeswitch
/var/log/omnitas-freeswitch.log	freeswitch-structured

Nginx

CGrateS JSONRPC

Grafana

Loki

- Alloy
- Loki
-
- Alloy 3100 Loki

- Alloy

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

2. 確認する Loki ログ

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

3. 確認する Loki ログ

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

4. 確認する Loki ログ TCP 3100

Alloy 確認する

確認する /var/lib/alloy 確認する

確認する

- Loki 確認する
- WAL 確認する

確認する

1. 確認する Loki 確認する
2. 確認する Loki 確認する WAL 確認する
3. 確認する Loki 確認する
4. 確認する 1 確認する

Loki 確認する

確認する Loki 確認する

確認する

- 確認する
- 確認する

確認する

1. 確認する Loki 確認する

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

2. 確認する Loki ログ
3. 確認する Loki ログ
4. 確認する

確認する

確認する component ログ infrastructure ログ

確認する

- 確認する
- Alloy ログ

確認する

1. 確認する
2. ログ Ansible ログ Alloy ログ

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

3. ログ Alloy

```
systemctl restart alloy
```

部署

组件	端口	协议	说明
Loki	3100	HTTP	日志查询 API
Loki	9096	gRPC	日志 gRPC 接口
Alloy	12345	HTTP	Alloy 配置 UI
Grafana	3000	HTTP	仪表盘

配置

- 配置 Grafana 连接 Prometheus
- 配置 Loki 日志源
- 配置 Alloy 数据源
- 配置 Grafana 仪表盘

核心网 (CS)

核心网 (2G/3G 网络 SMS) 在 `services/cs.yml` 中配置 BSC 和 OmniMSC 以及 SS7/SIGTRAN 接口

配置

`cs.yml` 中配置核心网

名称	配置	接口	设备	说明
BSC	<code>circuit_switched/bsc</code>	<code>bsc</code>	BSC, MGW	基站 + 网关 (2G/3G 网络)
MSC	<code>circuit_switched/omnimsc</code>	<code>msc</code>	<code>omnimsc</code>	移动交换中心
SS7	<code>omniss7</code>	<code>ss7</code>	<code>omniss7</code>	SS7/SIGTRAN 接口 (STP, HLR, SMSC, CAMEL, 等等)

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

BSC (`circuit_switched/bsc`)

在 Omnitouch APT 中配置 BSC 和网关 (MGW) 接口

- 配置 BSC 和 MGW 接口
- 在 MGW 中配置 systemd 服务 `cpu` 以及 `CPU Scheduling Policy` `Priority` 和 `AmbientCapabilities`
- 配置 `active` 为 6 的接口

MSC (circuit_switched/omnimsc)

omnimsc 5GC NF

- group_vars_omnicore license_server license_server_api_urls
- runtime.exs.j2 /etc/omnimsc/ omnimsc_template_config group_vars/
- CDR (omnimsc_cdr_output_dir /var/lib/omnimsc/cdr)
- Web UI / API TLS
- runtime.exs

SS7 (omniss7)

omniss7 SS7 ss7_role

ss7_role		
stp	stp.j2	(SS7)
hlr	hlr.j2	
smsc	ipsmgw.j2	SMSC / IP-SM-GW
camel	camel.j2	CAMEL
tester	tester.j2	/

- ss7_protocol: m2pa m2pa.j2 M2PA tester
- ss7_template_config group_vars/

Elixir omniss7 + runtime.exs /opt/omniss7/releases/*

□□□□

- □□□□ - □□□□□□□□□□□□
- 5G □□ - 5G □□□□□□□□ 5gc_nf □□
- □□□□□□ - □□□□□□□□□□ ss7_template_config□ omnimsc_template_config□
- **OmniCore** □□: <https://docs.omnitech.com.au/docs/repos/OmniCore> - □□□□□□□□□□□□

Quickstart

Prerequisites

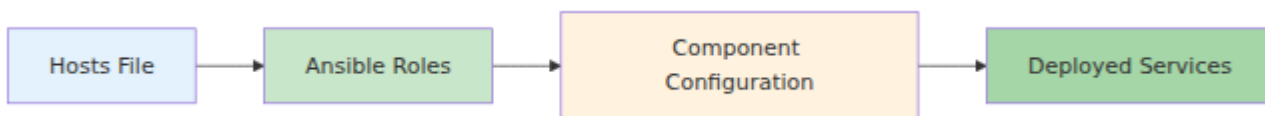
Before installing OmniCore, you must have the following prerequisites:

Operating System:

- **OmniCore:** <https://docs.omnitech.com.au/docs/repos/OmniCore>
- **OmniCall:** <https://docs.omnitech.com.au/docs/repos/OmniCall>
- **OmniCharge:** <https://docs.omnitech.com.au/docs/repos/OmniCharge>

Installation

OmniCore is installed using Ansible.



OmniCore uses Ansible to manage the deployment of services. The `group_vars` directory contains variables for the deployment. OmniCRM is installed using the `omnicrm` role.

```
group_vars:
  all:
    sql_pass: password, omnitech2024
    apt_packages:
      - omni/omni
```

Configuration

OmniCore is configured using the `IP` variable.

- `IP`
- `IP`
- `IP`
- `IP`

Ansible

hosts - hosts-file-configuration.md

Configuration

```
cdrs_enabled: True           # Enable CDR
in_pool: False               # In pool
populate_crm: False         # Populate CRM
```

Variables (all:vars)

all:vars defines the variables for the entire playbook. In this example, we define the variables for the OmniCRM and Grafana SMTP configurations.

Variables

Ansible

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

SSH Private Key

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

Summary

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

PLMN

```
plmn_id:
  mcc: '001'          # 00000003 0000
  mnc: '01'           # 00000002-3 0000
  mnc_longform: '001' # 0000 MNC000 3 0000

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

□ □

□□□□

```
network_name_short: Omni
network_name_long: Omnitouch
tac_list: [10100,100] # 00 TAC 000000 MME 0000
```

□□□□ UE □□□□□ > □□□□□□□□□□□□□□□□

DNS ☐

```
manage_resolv_conf: True           # False Ansible no  
/etc/resolv.conf
```

```

- name: manage_resolv_conf
  ansible.builtin.file:
    path: /etc/resolv.conf
    state: absent
  all:vars

```

APT □□□□□

`apt_repo` 設定ファイルに `remote_apt_*` `apt_download_base` URL 設定

設定ファイルに `apt_cache_servers` 設定

- `use_apt_cache` 設定 `True` 設定 `False`
- `apt_repo.apt_server` 設定 IP

```
use_apt_cache: False          # 設定 True = apt APT 設定

apt_repo:
  apt_server: "apt.<region>.omnitouch.ch" # Omnitouch 設定 FQDN 設定
  IP
  apt_repo_username: "your-username"      # 設定
  apt_repo_password: "your-password"      # 設定
  # apt_repo_port: 443                    # 設定 443 HTTPS
  # 設定 443 -> https

http
```

設定 **HTTPS / 443** 設定 **HTTP / 8080** 設定

`apt_cache_servers` 設定 `remote_apt_*` 設定 Omnitouch 設定

設定 APT 設定

設定

```
# 設定 `license_server` 設定
# 設定 https://<ansible_host>:8443/api
license_server_api_urls: ["https://10.10.2.150:8443/api"]
license_enforced: false      # 設定 false
(roles/omnihss/defaults/main.yml)
```

設定 `license_server` 設定 `license_server_api_urls` 設定

設定

MME 設定

MME `omnimme` `home/roaming` `dns_lookup` DNS `random_peer`

```
omnimme:
  sgw:
    selection_method: random_peer      # ...
    # selection_method:
    #   home: random_peer
    #   roaming: dns_lookup
  pgw:
    selection_method:
      home: random_peer
      roaming: dns_lookup
```

AMF

```
# tac_list AMF supported_ta_list TAC
# NG gNB AMF
# TAC 1 gNB
tac_list: [1, 3]

# NAS NEA0 true omniamf
# >=20260727 NEA2 NAS
# 5G
# NEA2 UE false NAS NIA2
amf_force_null_cipherring: true
```

SAEGW

```
mtu: 1400      #
```

OmniHSS

OmniHSS `dra` `mme` `pgwc` `scscf` `icscf` `pcscf` `applicationserver` `packet_core_dra_support` / `ims_dra_support` `exclude_diameter_peers` HSS

```
omnihss:
  exclude_diameter_peers:          # 指定したピアを除外
    - customer-as01
    - customer-as02
```

- `inventory_hostname` FQDN or IP
- `ansible_ssh_host`
- `ansible_ssh_port`
`/etc/omnihss/runtime.exs`
- `omnihss` vars: Ansible `omnihss`
- `config :hss, ims:` S-CSCF SIP

DRA 00000000000000000000 dra_peer_exclude: True 0000 OmniDRA 00000000
000000000000 HSS 000000000000 DRA 0000000000

OmniCRM

OmniCRM group_vars hosts/

```

omnicrm_deploy_mode: package
apt_repo omnicrm-api /
omnicrm-ui
crm_config.yaml /etc/omnicrm/crm_config.yaml
deployment_assets/
systemctl restart omnicrm-api
UI
ui:
GET /crm/public/ui-config
yarn
crm env override

```

```
omnicrm_deploy_mode: source[git] + [venv + yarn [crmlib]]
crm env override
```

```
omnicrm_deploy_mode: package      # package[ ] | source
omnicrm_package_version: ""      # apt [ ] omnicrm-api/ui
crm_config.yaml                  # API + ui: [ ]group_vars[ ]
deployment_assets/               # [ ]/[ ]
/etc/omnicrm/deployment_assets
crm_env_override: my-site.env    # [ ]/[ ] UI [ ]
cell_sites: cellSites.txt        # [ ]UI [ ]
message_templates: constants.js  # [ ]
site_data: site_data.json        # [ ]
```

Grafana SMTP

Configure Grafana SMTP with `grafana_smtp_user` and `grafana_smtp_password` and `grafana_smtp_host` in `in-v3.mailjet.com:587` in `roles/monitoring/templates/grafana/grafana.ini`

```
grafana_smtp_user: "smtp-user"          # SMTP user
grafana_smtp_password: "smtp-password"  # SMTP password
grafana_smtp_host: "in-v3.mailjet.com:587" # SMTP host
```

IMS

```
ims_dra_support: False      # DRA support in IMS
enable_homer: False        # Homer SIP support
```

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
```

□□□□□

```

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'

```

DNS

```

roaming_dns_servers:
  wildcard: ['10.0.99.1']
  # DNS PLMN
  123456: # 1
    - '10.10.2.197'
  654321: # 2
    - '10.10.0.4'

```

SSH

```

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

```

Proxmox

Proxmox

Proxmox VM or LXC configuration `deployment_type` `vm` `proxmoxServers` `Proxmox`

VM

```
proxmoxServers:
  customer-prmx01:
    # deployment_type vm → vm
    proxmoxServerAddress: 10.10.0.100
    proxmoxServerPort: 8006
    proxmoxApiTokenName: AnsibleToken
    proxmoxApiTokenSecret: "token-secret"
    proxmoxNodeName: pve01
    proxmoxTemplateName: ubuntu-24.04-cloud-init-template
    proxmoxTemplateId: 9000
    proxmoxTemplateUser: omnitouch # cloud-init user
    local_users:
      proxmoxTemplatePassword: omnitouch # cloud-init password
    cloud-init:
      storage: SSD_RAID0 # VM storage
```

LXC

```

proxmox_lxc_nameserver: "1.1.1.1" # 00000000 LXCs

proxmoxServers:
  customer-prxm01:
    deployment_type: lxc
    proxmoxServerAddress: 10.10.0.100
    proxmoxServerPort: 8006
    proxmoxApiTokenName: AnsibleToken
    proxmoxApiTokenSecret: "token-secret"
    proxmoxNodeName: pve01
    proxmoxLxcOsTemplate: "local:vztmpl/ubuntu-24.04-
standard_24.04-2_amd64.tar.zst"
    proxmoxLxcDefaultStorage: SSD_RAID0 # 000rootfs 00

```

000000000000

```

dns:
  vars:
    proxmox_interface: vmbr0 # 000000
    gateway: 10.10.0.1 # 00
    netmask: 255.255.255.0 # 00
    vlanid: 100 # 00
    proxmoxLxcCores: 2 # 0000 LXC
    proxmoxLxcMemoryMb: 4096 # 0000 LXC
    proxmoxLxcDiskSizeGb: 30 # 0000 LXC
    proxmoxLxcRootFsStorageName: SSD_RAID0 # 0000 LXC
    host_vm_network: vmbr1 # 000000

```

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
```

□□□□

- IP □□□□ - □□□□ IP □□□□
- □□□□□□ - □□□□□□□□
- □□□□□ - □□□□□□□□ group_vars
- Netplan □□ - □□ IP □□ NIC □□
- □□□□ - □□□□□□
- APT □□□□ - □□□
- □□□□□□ - □□□□□□

□□□□

□□□□□□□□□□□□□□□□

- **OmniCore** □□: <https://docs.omnitouch.com.au/docs/repos/OmniCore>
- **OmniCall** □□: <https://docs.omnitouch.com.au/docs/repos/OmniCall>

- **OmniCharge/OmniCRM:**

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

環境構築

概要

OmniTouch環境構築にAnsibleを用いて環境構築を行う。
4G/5G環境

IPアドレスを指定してIPを設定する。

環境構築

0. 環境構築

Proxmox環境構築VM/LXC

```
# Proxmox環境構築VM/LXC環境構築deployment_type環境構築  
ansible-playbook -i hosts/Customer/hosts.yml  
util_playbooks/proxmox.yml
```

Proxmox VM/LXC環境構築

1. 設定ファイル

```
# 設定ファイル
mme:
  hosts:
    customer-mme01:
      ansible_host: 10.10.1.15

hss:
  hosts:
    customer-hss01:
      ansible_host: 10.10.2.140

# ... 設定
```

設定ファイル

2. group_vars

group_vars 設定ファイル

OmniMessage SMS Scenarios TAS SIP Diameter

OmniCRM group_vars Ansible delegate_to: localhost

設定ファイル

3. APT

```
# 設定ファイル
apt_repo:
  apt_server: "10.254.10.223" # IP アドレス repo
  use_apt_cache: false # true = false = repo
```

APT

4. 設定

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

設定完了

5. 実行

ansible-playbook services/twag.yml services/all.yml --limit=myhost --limit=mme,sgw

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

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

完了

OmniCoreインストール完了

6. A. Ansible repo

Ansibleインストール hosts/

6. B. omnicores .deb

OmniCoreインストール omnicores Debian apt repo

- `name: omnicore` `packaging/nfpm.yaml` `Ansible` `vendored` `Python` `/opt/omnicore/python` `wheelhouse` `/opt/omnicore` `packaging/postinstall.sh` `+` `wheelhouse` `venv` `/opt/omnicore/venv` `/usr/local/bin` `ansible*` `PATH` `uv.lock` `Ansible` `Python` `Ansible` `Python` `git` `rsync` `openssh-client` `apt` `amd64` **Ubuntu 22.04+** `glibc ≥ 2.35`
- `packaging/build.sh` `repo` `/opt/omnicore` `hosts/` `.git/` `requirements.yaml` `Ansible` `+` `Python` `wheelhouse` `nfpm` `.deb`
- `.github/workflows/build-deb.yml` `main` `build.sh` `.deb` `_metadata.json` `GitHub` `apt` `discover_github_debs.py` `aptly` `apt`

📄

```
# 1. apt
apt-get install omnicore

# 2. 
#   - /opt/omnicore
vi /etc/omnicore/inventory

# 3. 
cd /opt/omnicore
ansible-playbook -i /etc/omnicore/inventory services/all.yml
```

`/etc/omnicore/inventory` `/opt/omnicore`

📄

- **Ansible** -
- -
- -
- **IP** - **IP**

- [Kubernetes](#) - [Tutorial](#)
- [APT](#) - [Tutorial](#)
- [Kubernetes](#) - [Tutorial](#)
- [Kubernetes](#) - Grafana Prometheus [Tutorial](#)
- [Kubernetes](#) - Loki Alloy [Tutorial](#)

[Kubernetes](#)

[Kubernetes](#)

- **OmniCore** 4G/5G <https://docs.omnitouch.com.au/docs/repos/OmniCore>
 - OmniHSS OmniSGW OmniPGW OmniUPF OmniDRA OmniTWAG
- **OmniCall** <https://docs.omnitouch.com.au/docs/repos/OmniCall>
 - OmniTAS OmniCall CSCF OmniMessage OmniSS7 VisualVoicemail
- **OmniCharge/OmniCRM** <https://docs.omnitouch.com.au/docs/repos/OmniCharge>
- <https://docs.omnitouch.com.au/>

目錄

iptables 與 ip6tables 的安裝與設定
iptables 的規則與表
ip6tables 的規則與表

- iptables 的安裝與設定
- iptables 的規則與表

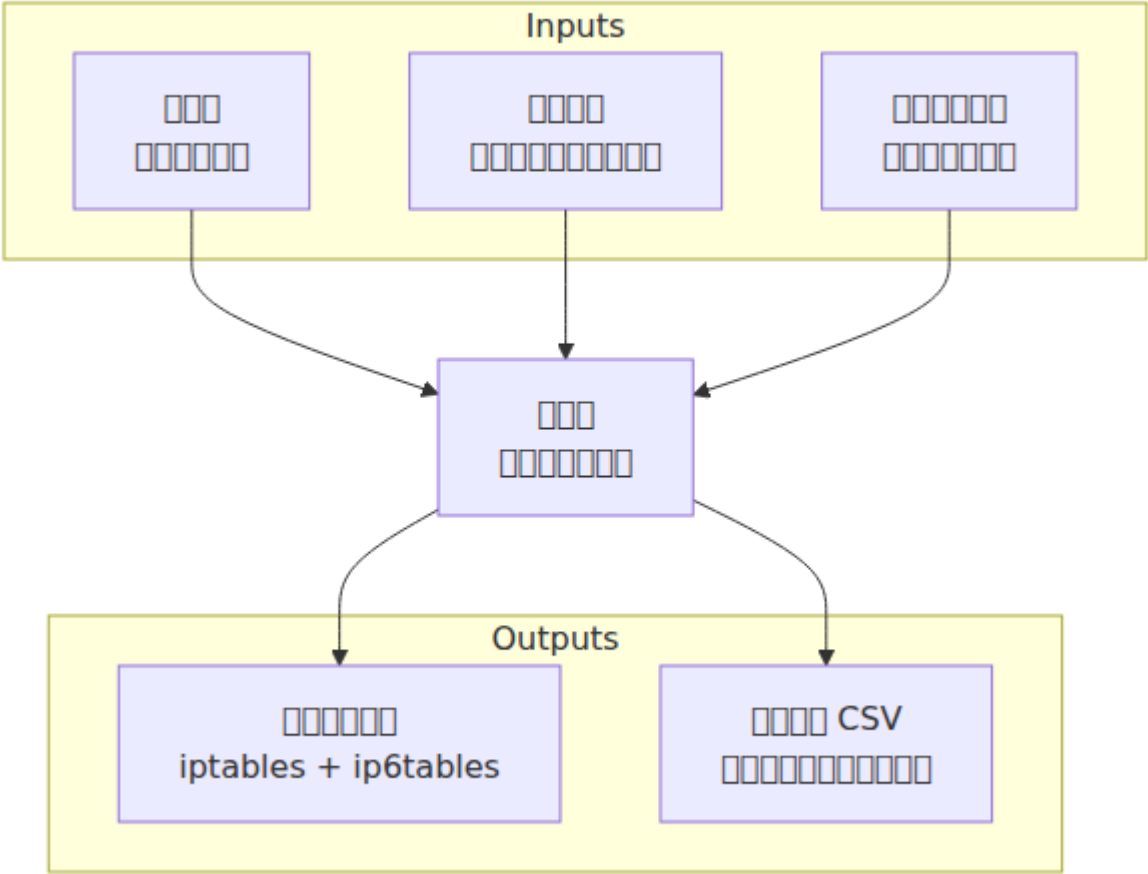
ip6tables 的安裝與設定
ip6tables 的規則與表

索引

- iptables
- ip6tables
- iptables 的安裝與設定
- ip6tables 的安裝與設定
- iptables 的規則與表
- ip6tables 的規則與表
- iptables 與 ip6tables 的差異
- iptables 的表與鏈
- iptables 的規則
- iptables 的 IP 地址
- iptables 的端口
- iptables 的協議
- iptables 的動作

附錄

iptables 的安裝與設定
ip6tables 的安裝與設定



이러한 구조를 구현하기 위해서는



다음과

같이 구성하면 iptables NFLOG가 ulogd2를 통해 journald에 로그를 기록하고, Alloy가 Loki에 로그를 전송하는 구조가 됩니다.

```
- id: '13' # iptables NFLOG
  dst: hss # hss
  service: diameter # hss
  purpose: 'Diameter' # Diameter
  sources: [{ zone: internal }] # internal
```

項目	種別	単位	注	説明
id	文字列	文字	-	一意に識別するためのID。ans:<dst>: <id>の形式で指定する。
dst	文字列	文字	-	宛先IPアドレス。all、five_g_sbi 5G などのキーワードも指定可能。
service	文字列	文字	*	サービス名。指定しない場合は すべてのサービスを含む。
proto_ports	文字列	文字	*	サービス名、all、PORT/proto、tcp、udp、sctp、 LO-HI、icmp、esp、proto 89など指定可能。
purpose	文字列	文字	-	目的。serviceと併用する。
sources	文字列	文字	-	送信元IPアドレス。指定しない場合は すべての送信元を含む。

* 単位は service、proto_ports 単位

検索

例 sources 10.0.0.0/24 からの接続をすべて取得する

項目	説明
<code>{ zone: <name> }</code>	指定した CIDR を <code>firewall.zones.<name></code> に設定
<code>{ list: <name> }</code>	指定したリストを <code>allowed_oam_subnets</code> に設定
<code>{ group: <name> }</code>	指定したグループを <code>allowed_oam_subnets</code> に設定
<code>{ ue_pools: true }</code>	UE / 指定した <code>firewall.ue_pools</code> に設定
<code>{ dra_peers: true }</code>	DRA 指定した <code>firewall.dra_peers</code> に設定
<code>{ roaming: true }</code>	IR.21 / GRX-IPX 指定した <code>firewall.roaming</code> に設定
<code>{ literal: "..."</code>	指定した literal を <code>firewall.literal</code> に設定

設定

この設定は、HSS のファイアウォールサービスを設定するための設定ファイルです。

`firewall_services.yml` に指定した設定は、`cscf` の `firewall_services` に適用されます。

```
# roles/omnihss/firewall_services.yml
firewall_services:
  hss:
    diameter: [3868/tcp, 3868/sctp] # S6a / Cx-Dx / Sh
    api_oam: 8080/tcp # HSS API (OAM)
    api_nf: 8443/tcp # HSS API (NF 向け)
    postgres: 5432/tcp # HSS データベース
```

この設定は、`firewall_services.yml` に指定した設定は、`service:` に指定した `dst` に適用されます。

指定した `hss` の `diameter` は、指定した `dst` に適用されます。

`firewall_services.yml` に指定した設定は、`firewall_services` に適用されます。

この設定は、HSS のファイアウォールサービスを設定するための設定ファイルです。

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

注意事項

- 注意 - 防火牆服務審計 playbook 位於 `firewall_services.yaml` 目錄下
- 注意 - 防火牆服務審計 playbook

防火牆服務審計 playbook 位於 `firewall_services.yaml` 目錄下

防火牆服務審計 playbook 位於 `firewall_services.yaml` 目錄下
MME 位於 `omnilcs` 目錄下
LCS 位於 `SBc-AP`
`29168/sctp` 位於 `SLs` 目錄下
`9082/sctp` 位於 `firewall_services.yaml` 目錄下
防火牆服務審計 playbook 位於 `firewall_services.yaml` 目錄下

防火牆

防火牆服務審計 playbook 位於 `all.vars` 目錄下
防火牆服務審計 playbook 位於 `firewall` 目錄下

防火牆服務審計 playbook 位於 `internal` 目錄下
防火牆服務審計 playbook 位於 `IMS` 目錄下

```

firewall:
  allowed_oam_subnets:          [ '10.8.0.0/24' ]          # 允许SSH
UI 允许 API
  allowed_ran_subnets:          [ '10.20.0.0/16' ]          # RAN 网络
SIAP NGAP GTP-U Abis
  allowed_ue_voice_subnets:     [ '100.64.0.0/16' ]          # UE SIP
P-CSCF
  allowed_carrier_voice_subnets: [ '203.0.113.0/24' ]          # SIP
zones:                          # 网络区域
  internal:                      [ '10.8.0.0/16' ]          # 内部网络
  mobile_core:                   [ '10.8.1.0/24' ]          # NF
EPC + 5GC + CS
  core_svc:                      [ '10.8.2.0/24' ]          # 
HSS DRA OCS OAM
  ims:                          [ '10.8.3.0/24' ]          # IMS
CSCF TAS
  ue_svc:                        [ '10.8.4.0/24' ]          # UE 
P-CSCF DNS ePDG
  core_v6:                       [ '2001:db8:0:1::/64' ]          # NF 
IPv6 允许 UE
  ue_pools:                      [ '100.64.0.0/16', '2001:db8:64::/48' ] # UE / 
IP 允许 v4 和 v6
  extra_rules:                   # 额外规则
  - { proto: tcp, dport: '8443', source: '203.0.113.10/32',
comment: '允许 API' }

```

allowed_oam_subnets 允许 SSH 访问 RAN 网络
 allowed_ran_subnets: [] 允许 IPv6 访问 core_v6: [] 允许 UE
 ue_pools: [] 允许 UE 访问 Required

項目	必須	オプション	デフォルト	説明
<code>allowed_oam_subnets</code>	必須	必須	-	SSH、OmniWeb UI、Prometheus API、 管理用ネットワーク、 管理用ネットワーク、 管理用ネットワーク
<code>allowed_ran_subnets</code>	必須	必須	[]	RAN、S1AP、 NGAP、GTP-U、Abis、 RAN
<code>allowed_ue_voice_subnets</code>	必須	必須	[]	UE SIP、P-CSCF
<code>allowed_carrier_voice_subnets</code>	必須	必須	[]	SIP
<code>zones</code>	必須	必須	-	管理用ネットワーク、 管理用ネットワーク
<code>ue_pools</code>	必須	必須	[]	UE / IP
<code>extra_rules</code>	必須	必須	[]	管理用ネットワーク、 管理用ネットワーク

管理用ネットワーク

管理用ネットワーク CIDR

名前	説明
internal	内部ネットワーク
mobile_core	モバイルコアネットワークEPC、MME、S/P-GW、UPF、5GC、AMF、SMF、SBI、CS、MSC、SS7/SIGTRAN
ims	IMS、I-CSCF、S-CSCF、TAS
ue_svc	UEサービスP-CSCF、DNS、ePDG
core_svc	コアサービスHSS、DRA、OCS、OAM
core_v6	IPv6コアネットワークIPv6

ファイアウォール

extra_rules: firewall_extra_rules: source: IPv6

名前	タイプ	単位	値	説明
proto	文字列	必須	-	tcp、udp、sctp
dport	文字列	必須	-	ポート番号
source	文字列	必須	-	CIDR形式: IPv6
comment	文字列	任意	override	コメント

モード

firewall_mode: 1

モード	説明
monitor	パケットは INPUT として受け取り、ACCEPT して、FW-AUDIT して、 パケットはそのままパケットとしてパケットとしてパケットとしてパケットとして
enforce	パケットは FW-DROP として INPUT して、FORWARD して、DROP して

パケットは enforce としてパケットとしてパケットとしてパケットとしてパケットとしてパケットとして
パケットとしてパケットとしてパケットとして monitor パケットとして firewall_mode:
monitor パケットとして enforce パケットとしてパケットとしてパケットとして

```
all:  
  vars:  
    firewall_mode: monitor # パケットとしてパケットとして 'enforce'
```

firewall_mode パケットとして all.vars パケットとしてパケットとしてパケットとしてパケットとしてパケットとして

パケット? ? ? パケット

パケットとして CSCF TAS OmniMessage HSS パケットとして OCS パケットとしてパケットとしてパケットとして
prod_master_peers.yml パケットとしてパケットとして { group: <role> } パケットとしてパケットとして
パケットとしてパケットとしてパケットとしてパケットとしてパケットとしてパケットとして

パケット

パケットとしてパケットとしてパケットとしてパケットとして ansible_host パケットとして secondary_ips.
<name>.ip_address パケットとしてパケットとして NIC UPF パケットとして n3 パケットとして N3 GTP-U
STP パケットとしてパケットとして SIGTRAN パケットとしてパケットとしてパケットとしてパケットとして
パケットとしてパケットとして

パケットとしてパケットとしてパケットとしてパケットとしてパケットとしてパケットとしてパケットとして
パケットとして

パケットとしてパケットとしてパケットとしてパケットとして NIC パケットとしてパケットとしてパケットとして N3 GTP-U
パケットとして UPF パケットとして N6 パケットとしてパケットとしてパケットとしてパケットとしてパケットとして

SIGTRAN NIC

□□/□□□□

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

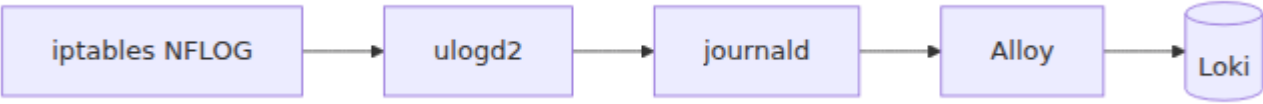
```
generated/firewall-port-matrix-exploded.csv
```

項目	説明
id	識別子
service	サービス
source	送信元 CIDR 範囲
destination_role	宛先役割
destination_host	宛先ホスト
destination_ip	宛先 IP
port	ポート番号 (ICMP は ESP 除外)
protocol	プロトコル

[illegible]

4444

```
firewall_rollback_guard
firewall_rollback_timeout
```



- 1. 配置 iptables
- 2. 配置 ulogd2
- 3. 配置 journald
- 4. 配置 Alloy
- 5. 配置 Loki

名称	类型	默认值	说明
firewall_rollback_guard	boolean	true	是否启用防火墙回滚保护
firewall_rollback_timeout	integer	120	防火墙回滚超时时间

IPv6 配置

配置 IPv6 CIDR 地址池 : 配置 IPv6 地址池 ip6tables 配置 IPv4 CIDR 地址池
配置 IPv6 地址池 IPv6 地址池 IPv6 地址池 core_v6: [] 配置 IPv6
配置 INPUT 配置 DROP 配置 IPv4 配置 IPv6 配置 IPv6
配置

配置

配置 NFLOG 配置 Loki

Ansible

Group vars

`group_vars` is a directory containing variable files for groups of hosts.

For example, you can create a file named `SIP` in the `group_vars` directory to define variables for all hosts in the `SIP` group. The file `group_vars` is located in the `hosts` directory.

For example, you can create a file named `hosts/{Customer}/group_vars/`.

For example, you can create a file named `group_vars_omnicore - group_vars/` in the `hosts/{Customer}/group_vars/` directory. The file `group_vars_omnicore` is located in the `hosts/{Customer}/group_vars/` directory.

Ansible

Ansible is a configuration management tool that can be used to manage infrastructure. It can be used to manage infrastructure in a variety of ways, including using `group_vars` to define variables for groups of hosts.

group_vars → group_vars → group_vars

1. Ansible

Ansible is a configuration management tool that can be used to manage infrastructure. It can be used to manage infrastructure in a variety of ways, including using Jinja2 to define variables for groups of hosts.

Ansible

```
hosts/Customer/  
└─ group_vars/  
    └─ smsc_controller.exs # group_vars
```

Ansible 設定

```
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 #  group_vars
```

実行方法

手順

1. Ansible 実行 `smsc_template_config: smsc_controller.exs`
2. `hosts/Custom/group_vars/smsc_controller.exs` 参照
3. Jinja2 テンプレート変数 `{{ inventory_hostname }}` `{{ plmn_id.mcc }}` 参照
4. 実行 `/etc/omnimessage/runtime.exs`
5. 確認

実行 `smsc_template_config` 確認

詳細情報 参照 <https://docs.omnitouch.com.au/docs/repos/OmniCall>

2. OmniTAS 設定

実行方法

目次

```
hosts/Customer/
├── group_vars/
│   ├── gateways_prod/                # SIP 設定
│   │   ├── gateway_carrier1.xml
│   │   ├── gateway_carrier2.xml
│   │   └── gateway_emergency.xml
│   ├── gateways_lab/                 # 実験用
│   │   └── gateway_test.xml
│   ├── dialplan/                     # ダイヤルプラン
│   │   ├── mo_dialplan.xml           # 発信ダイヤルプラン
│   │   ├── mt_dialplan.xml           # 着信ダイヤルプラン
│   │   └── emergency.xml
│   └── dialplan_lab/                 # 実験用
│       └── mo_dialplan.xml
```

Ansible playbook

```
applicationserver:
  hosts:
    customer-tas01:
      ansible_host: 10.10.3.60
      gateway: 10.10.3.1
      host_vm_network: "vmbr3"
      gateways_folder: "gateways_prod" # 発信ダイヤルプランのフォルダ
      dialplan_folder: "dialplan"      # 着信ダイヤルプランのフォルダ
      "dialplan"
```

手順

1. Ansible 上で `gateways_folder: "gateways_prod"`
2. `hosts/Customer/group_vars/gateways_prod/` フォルダに `/etc/freeswitch/sip_profiles/`
3. `hosts/Customer/group_vars/dialplan/` フォルダに `dialplan_folder` フォルダに `OmniTAS` フォルダ
4. 実行

関連するドキュメント

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

For more information, see <https://docs.omnitouch.com.au/docs/repos/OmniCall>

3. Deploying OmniHSS

Prerequisites

Tools

```
hosts/Custom
├── group_vars/
│   └── hss_runtime.exs.j2      # Deploy HSS runtime
```

Configuration

```
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  # Use group_vars file
  vars_files:
```

Steps

1. Ansible `hss_template_config: hss_runtime.exs.j2`
2. `hosts/Custom/group_vars/hss_runtime.exs.j2` file
3. Jinja2 template `{{ inventory_hostname }}` and `{{ plmn_id.mcc }}`

4. `/etc/omnihss/runtime.exs`

5. `omnimme`

`hss_template_config` `omnihss`

omnimme `https://docs.omnitouch.com.au/docs/repos/OmniCore`

4 `OmniMME`

omnimme `omnimme`

`omnimme`

```
hosts/Customr/  
└─ group_vars/  
    └─ mme_runtime.exs.j2          # omnimme omnimme
```

`omnimme`

```
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  # group_vars omnimme  
omnimme
```

`omnimme`

1. Ansible `mme_template_config: mme_runtime.exs.j2`

2. `hosts/Customr/group_vars/mme_runtime.exs.j2` `omnimme`

3. Jinja2 `{{ inventory_hostname }}` `{{ plmn_id.mcc }}` `omnimme`

4. `/etc/omnimme/runtime.exs`

5. `omnimme`

```
mme_template_config
```

□□□□ □□ <https://docs.omnitouch.com.au/docs/repos/OmniCore>

5 OmniCRM

OmniCRM 的 group_vars 目录 - 包含所有API 调用所需的配置信息

```
□□□□□□apt □
```

```

omnicrm_deploy_mode: package
Ansible apt_repo
omnicrm-api omnicrm-ui /etc/omnicrm/ systemctl
omnicrm-api UI crm_config.yaml ui: GET
/crm/public/ui-config

```

□□□□

```
hosts/
└─ group_vars/
    ├── crm_config.yaml           # API + ui: 000000
    └─ deployment_assets/       # 000000/00 →
/etc/omnicrm/deployment_assets
├   └─ logos/
└─ customer_crm.env             # 000000 omnicrm_deploy_mode:
source 000 UI 00
├─ cellSites.txt               # 00000000
├─ constants.js               # 000000000000
└─ site_data.json             # 000000000000
```

ansible-playbook

```
omnicrm:
  hosts:
    customer-crm01:
      ansible_host: 10.10.3.70
      gateway: 10.10.3.1
      host_vm_network: "vmbr3"
      omnicrm_deploy_mode: package          # source or package
      git+yarn
      # omnicrm_package_version: "20260810..." # apt pin
      logo_light: logo-light.png           #
      deployment_assets/logos/
      logo_dark: logo-dark.png
      login_background: login-bg.jpg        # →
      deployment_assets/login/auth-one-bg.jpg ( )
      login_splash: splash.jpg              # →
      deployment_assets/login/login-side.jpg ( )
      ssl_fqdn: portal.example.com
      # crm_env_override: customer_crm.env  #
      cell_sites: cellSites.txt             #
      message_templates: constants.js       #
      site_data: site_data.json             #
```

ansible-playbook

1. Apt `omnicrm-api` `omnicrm-ui`
2. `{{ group_vars_omnicore }}/crm_config.yaml` `/etc/omnicrm/crm_config.yaml`
3. `deployment_assets/` `logo_*` / `login_*` `/etc/omnicrm/deployment_assets/`
4. `omnicrm-api` `nginx` `/usr/share/omnicrm-ui` UI

`GET /crm/public/assets/login/...` `crm_config.yaml`
`ui.branding.login_splash / login_background` UI `.deb`
`splash.jpg` → `login/login-side.jpg` API `login-side.*`

ansible-playbook

1. `crm_config.yaml` `/etc/omnicrm/OmniCRM-API/`

2. `{{ crm_env_override }}` の UI `.env` をインストール

3. `yarn` をインストール UI

```
crm_config.yaml をインストールする ui: ui.assets_dir:  
'/etc/omnicrm/deployment_assets'
```

詳しくは <https://docs.omnitouch.com.au/docs/repos/OmniCore>

6 OmniHSS

OmniHSS をインストール `services/provision_omnihss.yml` の `group_vars` を `omnihss/` にインストール

インストール

```
hosts/Customr/  
└─ group_vars/  
    └─ omnihss/  
        ├── subscribers.csv          # インストール  
        ├── apn_identifiers.json  
        ├── apn_qos_profiles.json  
        ├── apn_profiles.json  
        ├── eir_rules.json  
        ├── ims_profiles.json  
        ├── epc_profiles.json  
        ├── roaming_rules.json  
        ├── roaming_profiles.json  
        ├── flow_information_rules.json  
        ├── charging_rules.json  
        └─ pcrf_profiles.json
```

インストール

1. `{{ group_vars_omnicore }}`/omnihss/subscribers.csv をインストール
2. `omnihss/` にインストール JSON `apn_identifiers.json`
`apn_qos_profiles.json` `apn_profiles.json` `eir_rules.json`

ims_profiles.json epc_profiles.json roaming_rules.json
roaming_profiles.json flow_information_rules.json
charging_rules.json pcrf_profiles.json APN
QoS IMS EPC PCRF

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

7 Grafana SMTP

grafana.ini group_vars Grafana SMTP

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

- grafana_smtp_user grafana_smtp_password
/etc/grafana/grafana.ini [smtp]
- grafana_smtp_host in-v3.mailjet.com:587

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

8 UDM SUCI HNET

OmniUDM SUCI TS 33.501 §6.12 UE SUCI SUPI
USIM group_vars hnet_udm/
5gc_nf UDM /etc/omniudm/hnet/

```
hosts/Customer/  
└─ group_vars/  
    └─ hnet_udm/  
        └─ curve25519-1.key      # X25519 PEM ID 1  
        └─ curve25519-1.pub      # 32 - USIM  
        └─ secp256r1-2.key      # ECprime256v1 PEM HNET  
ID 2  
    └─ ...
```

<curve>-<id>.key <id> UDM SUCI <curve>
curve25519 AX25519 secp256r1 Bprime256v1 HNET
ID USIM ID

1. {{ group_vars_omnicore }}/hnet_udm/ Ansible become:
false
2. *.key
3. curve25519-1.key X25519 SUCI
creates:
4. curve25519-*.key 32 .pub USIM
.pub UDM
5. *.key UDM /etc/omniudm/hnet/ 0600
omniudm

USIM SUCI SIM
.pub ID 1 USIM SIM HNET

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

📄📄📄📄 📄📄 <https://docs.omnitouch.com.au/docs/repos/OmniCore>

目录

```
hosts/Customer/
├─ host_files/
│   └─ production.yml          # 配置 group_vars 文件
└─ group_vars/
    ├─ smsc_controller.exs     # OmniMessage 配置
    ├─ smsc_smpp.exs           # OmniMessage SMPP 配置
    ├─ tas_runtime.exs.j2      # TAS 配置
    ├─ hss_runtime.exs.j2      # HSS 配置
    ├─ mme_runtime.exs.j2      # MME 配置
    ├─ dra_runtime.exs.j2      # DRA 配置
    ├─ pgwc_runtime.exs.j2     # PGW 配置
    ├─ dea_runtime.exs.j2      # DEA 配置
    ├─ upf_config.yaml         # UPF 配置
    ├─ crm_config.yaml         # CRM 配置
    ├─ stp.j2                  # SS7 STP 配置
    ├─ hlr.j2                  # SS7 HLR 配置
    ├─ camel.j2                # SS7 CAMEL 配置
    ├─ ipsmgw.j2               # IP-SM-GW 配置
    ├─ omnicore_smsc_ims.yaml.j2 # SMSC IMS 配置
    ├─ pytap.yaml              # TAP3 配置
    ├─ sip_profiles/           # SIP 配置
    │   └─ gateway_otw.xml
    ├─ dialplan/               # 拨号计划配置
    │   ├─ mo_dialplan.xml      # 移动拨号计划
    │   ├─ mt_dialplan.xml      # 移动终端拨号计划
    │   └─ mo_emergency.xml     # 移动紧急拨号计划
    ├─ omnihss/                # OmniHSS 配置
    │   ├─ subscribers.csv      # 订阅者数据
    │   ├─ apn_identifiers.json  # APN 标识符
    │   ├─ apn_qos_profiles.json # APN QoS 配置文件
    │   ├─ apn_profiles.json    # APN 配置文件
    │   └─ eir_rules.json       # EIR 规则
    │   └─ ims_profiles.json    # IMS 配置文件
    │   └─ epc_profiles.json    # EPC 配置文件
    │   └─ roaming_rules.json   # 漫游规则
    │   └─ roaming_profiles.json # 漫游配置文件
    │   └─ flow_information_rules.json # 流信息规则
    │   └─ charging_rules.json  # 计费规则
    │   └─ pcrf_profiles.json   # PCRF 配置文件
    └─ hnet_udm/               # UDM SUCI 配置
```

└─ curve25519-1.key	# X25519 000HNET 00 ID 10
└─ curve25519-1.pub	# 00000000000000 USIM 00

group_vars 目录

名称	目录	说明
smc_template_config	omnimessage	Jinja2 模板文件 smc_controller.exs[
smc_smpp_template_config	omnimessage_smpp	Jinja2 模板文件 smc_smpp.exs
gateways_folder	applicationserver	目录 gateways_prod
dialplan_folder	applicationserver	目录 dialplan - dialplan
tas_runtime_template	applicationserver	Jinja2 模板文件 tas_runtime.exs.j2 - tas_runtime.exs.j2
hss_template_config	omnihss	Jinja2 模板文件 hss_runtime.exs.j2
mme_template_config	omnimme	Jinja2 模板文件 mme_runtime.exs.j2
dra_template_config	dra	Jinja2 模板文件 dra_runtime.exs.j2
pgwc_template_config	pgwc	Jinja2 模板文件 pgwc_runtime.exs.j2[

項目	項目	項目
frr_template_config	omniupf	Jinja2 テンプレート frr.conf.j2
SS7 項目	ss7テンプレート	Jinja2 テンプレート stp.j2、hlr.j2 camel.j2
crm_config.yaml	omnicrm	API 呼び出し - テンプレート {{ group_vars_omnicore }}/crm_config.yaml の テンプレート
crm_env_override	omnicrm	UI .env テンプレート customer_crm.env - [テンプレート
cell_sites	omnicrm	テンプレート cellSites.txt
message_templates	omnicrm	テンプレート constants.js
site_data	omnicrm	テンプレート site_data.json
grafana_smtp_user	monitoring	Grafana SMTP 項目 - 項目 項目
grafana_smtp_password	monitoring	Grafana SMTP 項目 - 項目 項目
grafana_smtp_host	monitoring	項目 Grafana SMTP 項目 - 項目 in- v3.mailjet.com:587

項目	項目	項目
OmniHSS 項目	omnihss	omnihss/ 項目項目 subscribers.csv + 項目 項目 JSON
HNET 項目	5gc_nfomniudm項目	hnet_udm/ 項目項目項目 SUCI 項目項目 <curve>- <id>.key 項目 - 項目項目項目 項目項目項目 curve25519- 1.key
項目 YAML	項目項目	項目項目項目項目項目 upf_config.yaml項目 crm_config.yaml項目

項目項目

1. **group_vars** 項目項目 - 項目項目項目
2. 項目項目 - 項目項目 `smc_template_config` 項目 `gateways_folder`
3. 項目項目 **Jinja2** - 項目 `{{ variable_name }}` 項目項目 Ansible 項目
4. 項目項目項目項目項目 - 項目項目項目項目項目項目項目項目項目
5. 項目項目項目項目項目 - 項目項目 `group_vars` 項目項目 Git

項目項目 **group_vars**

項目項目 **group_vars** 項目項目

- 項目項目項目項目項目
- SIP 項目項目
- 項目項目項目項目項目
- Diameter 項目項目項目

- 環境変数

group_vars

- IPアドレス - 環境変数
 - 環境変数 - 環境変数
 - 環境変数 - 環境変数 group_vars
-

環境変数

- 環境変数 - 環境変数
- 環境変数 - 環境変数
- **OmniCall** 環境変数: <https://docs.omnitouch.com.au/docs/repos/OmniCall> - 環境変数
- **OmniCore** 環境変数: <https://docs.omnitouch.com.au/docs/repos/OmniCore> - 環境変数

Ansible

目的

Ansibleを用いて、ネットワーク機器の構成を自動化する。

- 構成の管理
- 構成のバックアップ/リストア
- 構成の検証
- 構成のPLMNごとの管理

環境

Ansibleは hosts/ ディレクトリに ansible.cfg という Ansible の inventory

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

Ansibleは Production.yml Staging.yml などの inventory
production.yml/staging.yml/lab.yml という

Ansibleのインストール

AnsibleはPythonで書かれているので、Pythonがインストールされている環境で実行できる。

```
# EPC
mme:
  hosts:
    customer-mme01:
      ansible_host: 10.10.1.15
      gateway: 10.10.1.1
      host_vm_network: "vmbr1"
      mme_code: 1
      network_name_short: Customer
      tac_list: [600, 601, 602]

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

pgwc:
  hosts:
    customer-pgw01:
      ansible_host: 10.10.1.21
      gateway: 10.10.1.1
      ip_pools:
        - '100.64.16.0/24'

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

#
license_server:
  hosts:
    customer-licenseserver:
      ansible_host: 10.10.2.150

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

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

--	--	--	--	--	--

--	--	--	--

--	--	--	--	--	--	--	--	--

```
pcscf:
  hosts:
    customer-pcscf01:
      ansible_host: 10.10.1.15      # SSH 接続 IP 指定
      gateway: 10.10.1.1           # ゲートウェイ
      host_vm_network: "vmbr1"     # 仮想ネットワーク NIC 指定
```

IP 10.10.10.10 IP 10.10.10.10 OmniCore 10.10.10.10

Proxmox `host_vm_network`  
VM/LXC 

VM ☐ ☐ ☐ ☐

□ □ □ □ □ □ □ □ □ □ □

```
num_cpus: 4 # CPU 00
memory_mb: 8192 # 8192 MB RAM
proxmoxLxcDiskSizeGb: 50 # 50 GB 0000
```

--	--	--	--	--	--	--	--

□ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □

MME:

```
mme_code: 1 # MME 00001-2550
mme_gid: 1 # MME 0 ID
network_name_short: Customer # 000000000000
network_name_long: Customer Network
tac_list: [600, 601, 602] # 000000
```

PGW:

```
ip_pools:                                     # 指定 IP 池
- '100.64.16.0/24'
- '100.64.17.0/24'
combined_CP_UP: false                       # 是否合并 CP 和 UP
```

Progress: 100%

□ □ □ □ □ □

```
online_charging_enabled: true # [] OCS []
tas_branch: "main" # []
gateways_folder: "gateways_prod" # SIP []
```

`all:vars` □□□□□□□□□□□□□□□□

```

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

    # 顧客
    customer_name_short: customer
    customer_legal_name: "Customer Inc."
    site_name: "Chicago DC1"
    region: US

    # PLMN
    plmn_id:
      mcc: '001'          # 
      mnc: '01'           # 
      mnc_longform: '001' #  MNC

    # ネットワーク
    network_name_short: Customer
    network_name_long: Customer Network

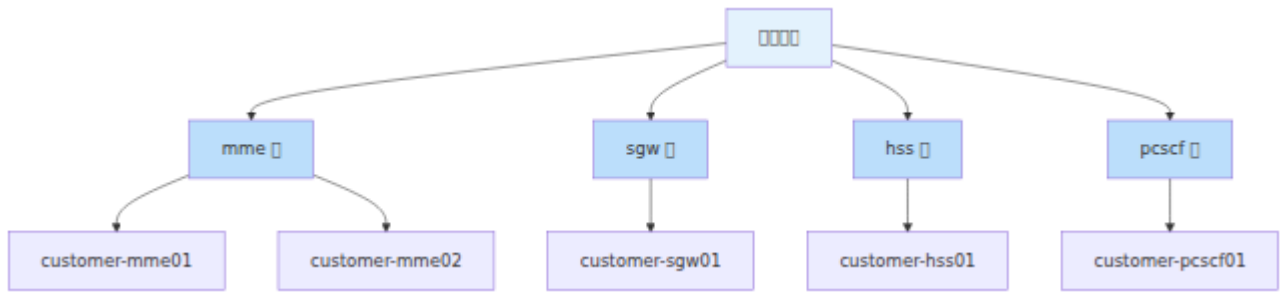
    # APT
    # apt_cache_servers 
    # use_apt_cache true apt_repo.apt_server
    # IP
    apt_repo:
      apt_server: "10.254.10.223"
      apt_repo_username: "customer"
      apt_repo_password: "secure-password"
    use_apt_cache: false

    # 
    TZ: America/Chicago

```

Ansible

Ansible



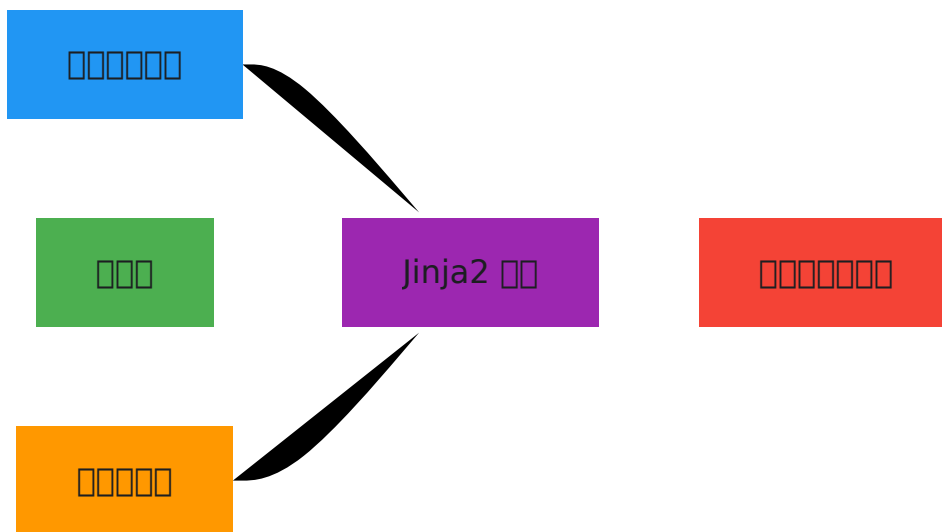
core mme mme:hosts:

ocs crm smsc omnimessage dra xcap
homer cbc 5GC amf smf upf

Jinja2

Ansible Jinja2 group_vars

Jinja2



template

vars

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

Jinja2 テンプレート

```
# 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
```

テンプレート

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

Jinja2

テンプレート

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

テンプレート

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

□□□

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

□□□□

```
# □□□□ 3 □□□
mnc{{ '%03d' | format(plmn_id.mnc|int) }}
```

□□ **group_vars** □□□□

□□□□□□□□□□□□□□□□□□□□ `group_vars` □□□□□□□□□□□□

□□□□□□□□□□

□□□□□□□□□□

□□□□□□□□□□□□□□□□

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

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

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

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

# IMS []
pcscf:
```

```
hosts:
  customer-pcscf01:
    ansible_host: 10.10.4.165
    gateway: 10.10.4.1
    host_vm_network: "vmbr4"

icscf:
  hosts:
    customer-icscf01:
      ansible_host: 10.10.3.55
      gateway: 10.10.3.1
      host_vm_network: "vmbr3"

scscf:
  hosts:
    customer-scscf01:
      ansible_host: 10.10.3.45
      gateway: 10.10.3.1
      host_vm_network: "vmbr3"

applicationserver:
  hosts:
    customer-as01:
      ansible_host: 10.10.3.60
      gateway: 10.10.3.1
      host_vm_network: "vmbr3"
      online_charging_enabled: false
      gateways_folder: "gateways_prod"

# 计费
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"

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

    customer_name_short: customer
    customer_legal_name: "Customer Network Inc."
    site_name: "Primary DC"
    region: US
    TZ: America/Chicago

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

# 网络
network_name_short: Customer
network_name_long: Customer Network
tac_list: [600, 601]

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

# 网络
charging:

```

```

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

# 設定
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'

# 設定 Proxmox VM 設定
proxmoxServers:
  customer-prxm01:
    # deployment_type 設定 → 設定 "vm"
    proxmoxServerAddress: 10.10.0.100
    proxmoxServerPort: 8006
    proxmoxApiTokenName: Customer
    proxmoxApiTokenSecret: "token-secret"
    proxmoxNodeName: pve01
    proxmoxTemplateName: ubuntu-24.04-cloud-init-template
    proxmoxTemplateId: 9000
    proxmoxTemplateUser: omnitouch # 設定 cloud-init 設定
    # 設定 local_users 設定
    proxmoxTemplatePassword: omnitouch # 設定 cloud-init 設定
    # 設定 cloud-init 設定

# 設定 LXC 設定 deployment_type: lxc 設定
# 設定 proxmoxServers 設定 proxmoxLxc0sTemplate

```

Proxmox VM/LXC 設定 Proxmox 設定

□□□□□□

□□□□□□□□□□□□□□□□□□

OmniCore □□□

- **OmniCore** □□□ <https://docs.omnitech.com.au/docs/repos/OmniCore>
- **OmniHSS** - □□□□□□□□
- **OmniSGW** - □□□□□□□□□□
- **OmniPGW** - □□□□□□□□□□
- **OmniUPF** - ◆◆◆□□□□□
- **OmniDRA** - Diameter □□□□
- **OmniTWAG** - □□ WLAN □□□□

OmniCall □□□

- **OmniCall** □□□ <https://docs.omnitech.com.au/docs/repos/OmniCall>
- **OmniTAS** - IMS □□□□□□VoLTE/VoNR□
- **OmniCall CSCF** - □□□□□□□□
- **OmniMessage** - □□□□
- **OmniMessage SMPP** - SMPP □□□□
- **OmniSS7** - SS7 □□□
- **VisualVoicemail** - □□□□

OmniCharge/OmniCRM□

- **OmniCharge** □□□ <https://docs.omnitech.com.au/docs/repos/OmniCharge>

□□□□

- **Ansible** □□□□ - □□□□□□
- □□□□ - □□□□□□□□□□
- □□□□□ - □□□□□□
- **IP** □□□□ - □□□□□ **IP** □□□□
- **Netplan** □□ - □□ **IP** □□□□□□□□

- APT 0000 - 00000
- 000000 - 00000
- 000000 - 000000

000

1. 00000000000000
2. 0000 PLMN 00000
3. 00 APT 00000
4. 00000000
5. 00000 00 group_vars 00000
6. 00 Ansible 000000

部署指南

简介

本指南旨在帮助您快速部署 Omnitouch 设备，并配置相关的网络和服务。

环境

1. 网络配置

```
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: "HSS"
    license_server_api_urls: ["https://10.10.2.150:8443/api"]
```

在 `license_enforced` 中配置 **HSS** 设备，并指定 `omnihss` 角色。
在 `roles/omnihss/defaults/main.yml` 中配置 `false`，并指定 HLD 设备。
在 `omnihss` 设备中配置。

2. 部署步骤

在 `license.json` 中配置 Omnitouch 设备，并指定 `hosts/Custom/group_vars/`。

3. 設定

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

hosts/Omnicoe_Customer/hosts/Omnicoe_Customer/host_files/Production.yml

https://license_server

util_playbooks/delete_license.yml

HTTPS 443 Omnitouch

名前	値
1.time.omnitouch.com.au	1
2.time.omnitouch.com.au	2
3.time.omnitouch.com.au	3

- HTTPS (TCP/443)
- DNS
-

DNS 係 IP 係 FQDN 係 IP 係
IP 係

DNS

DNS Omnitouch

DNS

- DNS
- DNS
 - 1.1.1.1 Cloudflare - DNS
 - 8.8.8.8 Google DNS
- DNS

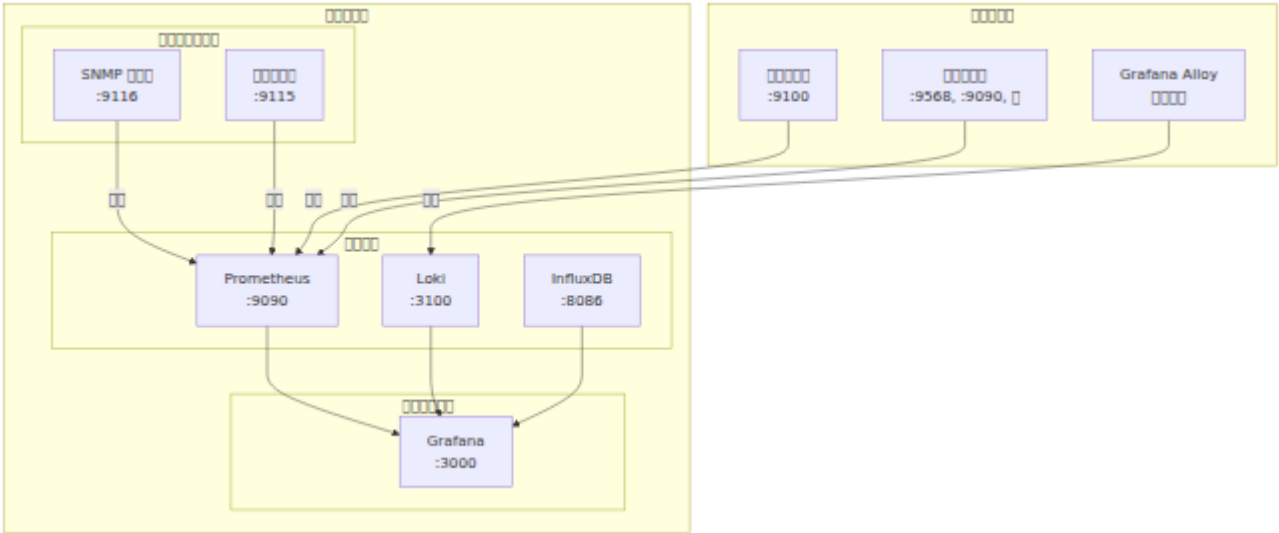
Omnitouch DNS DoH/DoT DNS DNSSEC DNS

-
-

環境構築

概要

OmniCore 環境構築の概要図



表

名称	描述	端口	资源需求
Prometheus	时序数据库	9090	30 核 5GB
Loki	日志聚合	3100	7 核 50GB
InfluxDB	RAN 数据 (时序)	8086	30 核
Grafana	可视化	3000	1核
SNMP 代理	网络监控	9100	1核
SNMP 设备	网络监控	9116	1核
ICMP/HTTP 代理	网络监控	9115	1核

部署

Prometheus

Prometheus 在 1 个节点上部署 OmniCore 节点

容器 UID: `omnicore_prometheus`

配置

名前	名前	ポート	説明
すべて	all	9100	すべての CPU 機能
MMEs	mme	9568	移動性管理/コア
HSS	hss	9568	ホームサブスクリプションシステム
SGW-C	sgw	9568	サービスゲートウェイ GTP-C 機能
PGW-C	pgwc	9090	PDN ゲートウェイ IP 機能
UPF 機能	upf	9090	パケットコア機能
CSCF	pcscf, scscf, icscf	9090	IP 多メディアサブシステム
FreeSWITCH	applicationserver	9090	アプリケーションサーバー
DRA	dra	9568	データローミングアーキテクチャ
OmniMessage	omnimessage	9568	メッセージング SMPP 機能
OmniSS7	omniss7	8080	SS7/SIGTRAN 機能
KeyDB	ocs	9121	データベース/キー
CGrates	ocs	2080	CDR データベース

名前

項目	項目	項目
SNMP (MikroTik)	項目/項目	<code>mikrotik.hosts</code>
SNMP (iDRAC)	項目項目	<code>idrac.hosts</code>
SNMP (Synology)	NAS 項目	<code>synology.hosts</code>
SNMP (SAF)	項目項目	<code>saf.hosts</code>
SNMP (Cisco)	項目	<code>cisco.hosts</code>
項目 ICMP	項目項目 + 8.8.8.8	項目
VMware	vCenter 項目	<code>vcenter_ip</code> , <code>vcenter_password</code>
Proxmox	PVE 項目	<code>proxmoxServers</code>

Loki

Loki 項目項目項目項目 Grafana Alloy 項目項目項目

項目 **UID:** `omnicore_loki`

項目 項目項目項目 項目項目 Loki/Alloy 項目

項目項目

項目	説明	値
hostname	ホスト名	customer-mme01
component	コンポーネント	mme, cscf, ocs
unit	systemd ユニット	omnimme.service
level	ログレベル	info, error

InfluxDB

InfluxDB への接続情報

UID: omnicores_influxdb

接続

名前	説明	単位
nokia-monitor	RAN モニタリング	30 分
dra	DRA モニタリング	365 分
Omnicharge_TAP3	TAP モニタリング	90 分

Grafana 接続

接続情報

接続情報

名前	説明
BSS	CGrateSとKeyDBのデータベース
EPC	MME、SGW、PGW、UPF、HSS、DRA の名前
IMS	CSCF、OmniMessage の名前
名前	名前、SNMPの名前
RAN	名前/OmniRAN の名前
名前	名前

名前 (名前 API)

名前 Grafana API の名前

- Grafana UI の名前
- API の名前
- Ansible の名前



成功

Ansible 成功

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

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

JSON Ansible

```

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

```

部署脚本

部署脚本 Grafana 部署脚本部署脚本部署脚本部署脚本

```

# 在 roles/monitoring 下
python3 export_grafana.py --customer <CUSTOMER>

```

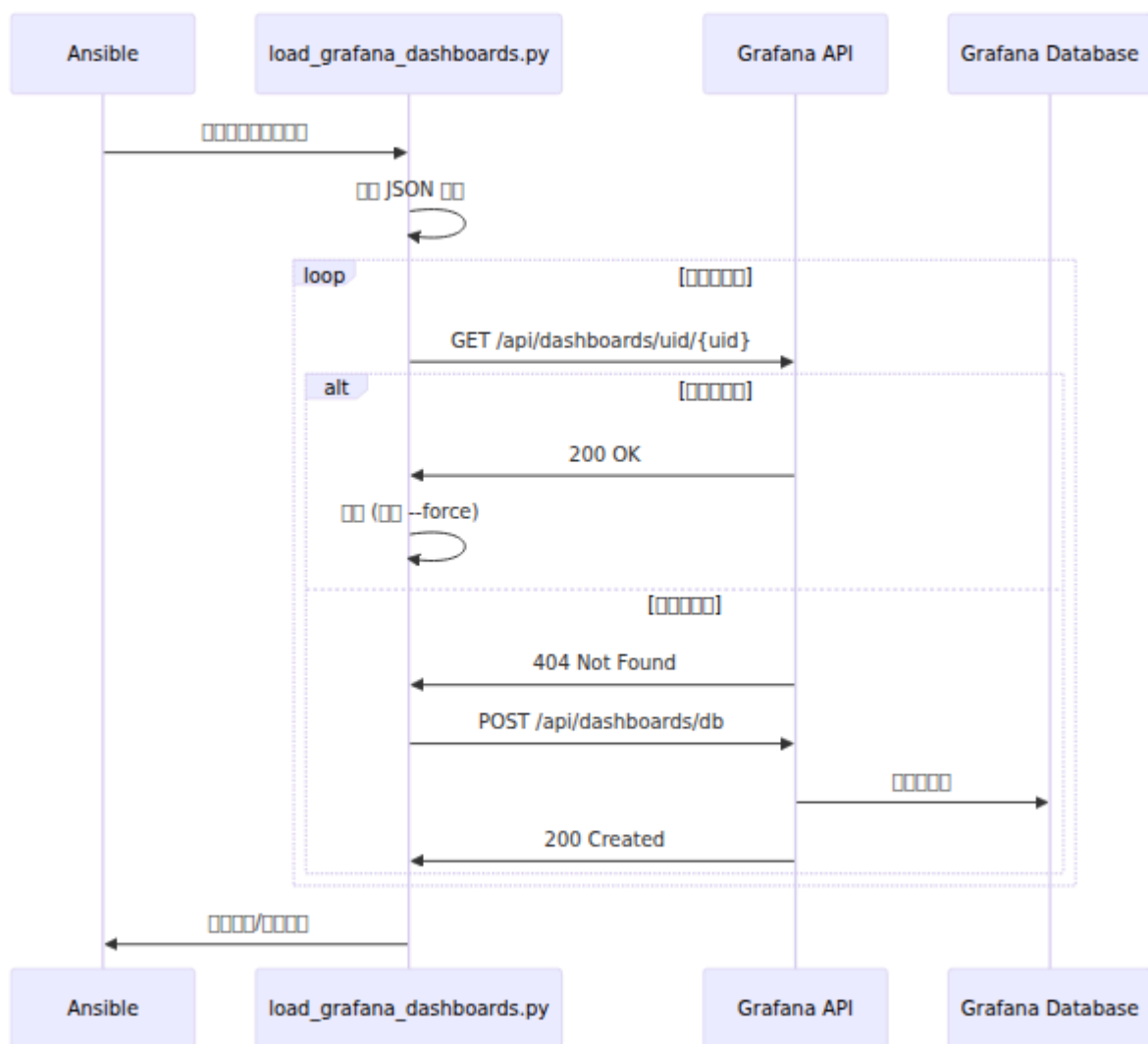
部署目录

- 部署目录

roles/monitoring/templates/grafana/dashboards/{folder}/*.json (网元)

)

- `hosts/Omnicores_{CUSTOMER}/group_vars/grafana/alerts/all_alert_rules.json`
- `hosts/Omnicores_{CUSTOMER}/group_vars/grafana/alerts/contact_points.json`
- `hosts/Omnicores_{CUSTOMER}/group_vars/grafana/alerts/notification_policies.json`



)

- `version` `id`
- Grafana

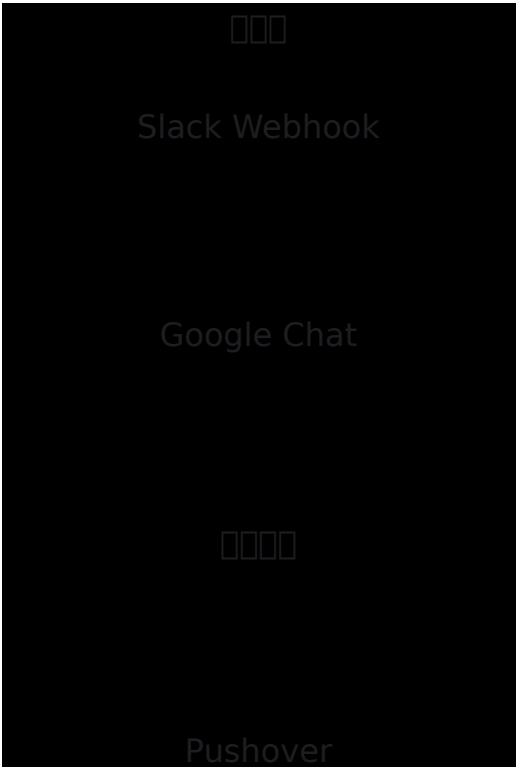
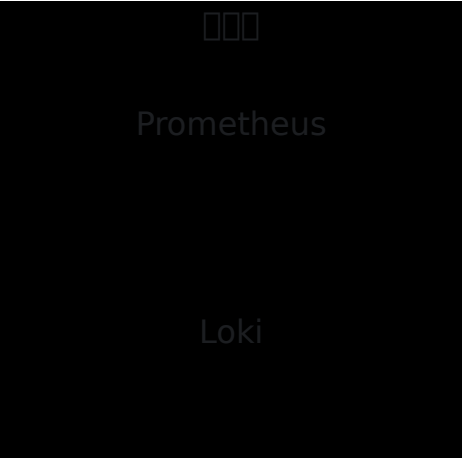
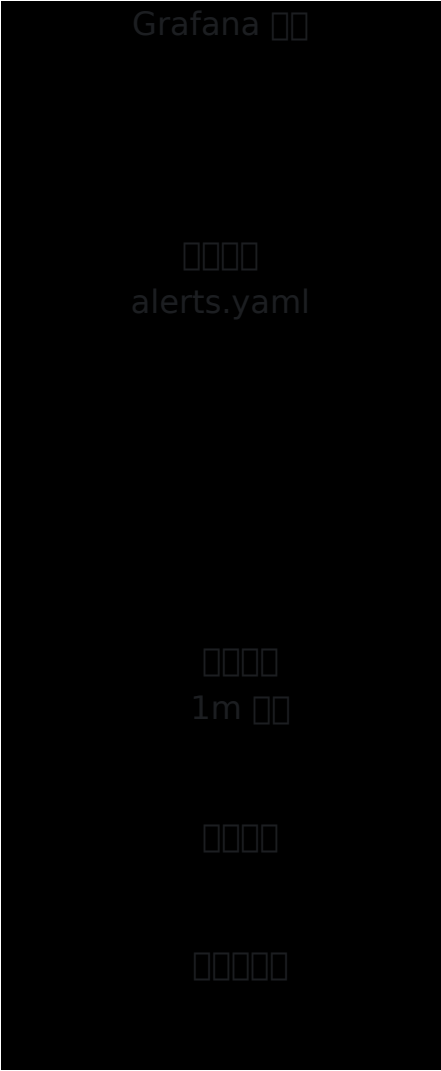
- 0000000000000000

0000000

0000000000000000 group_vars/grafana/dashboards/ 0000000000000000

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





概要

このスクリプトは Grafana API を利用して、Grafana の YAML を読み込み、Grafana の UI/API を呼び出す。

このスクリプトは `hosts/0mnicore_<CUSTOMER>/group_vars/grafana/alerts/` にある `load_grafana_from_export.py` を

```
python3 roles/monitoring/files/load_grafana_from_export.py \
  --url http://<monitoring-ip>:3000 \
  --user admin \
  --password <grafana_admin_password> \
  --alerts-dir
hosts/0mnicore_<CUSTOMER>/group_vars/grafana/alerts \
  --dashboards-dir roles/monitoring/templates/grafana/dashboards
\
  --force
```

このスクリプトは `load_grafana_alerts.py` を利用して `alerts.yaml` を

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

Alerts

Alert	Category	Expression	Duration
Alert: CPU usage 90%	Alert	node_cpu_usage > 90%	5 min
CPU usage 90%	Alert	CPU usage > 90%	5 min
Alert: Memory usage 90%	Alert	node_memory_usage > 90%	5 min
Alert: Prometheus	Alert	Prometheus	5 min
MME connection 40%	EPC	MME connection 40%	30 min
MME 0	EPC	MME 0	5 min
IMS connection 40%	IMS	P-CSCF connection 40%	30 min
MOS 4	IMS	MOS 4	5 min
eNodeB	RAN	eNodeB	1 min
P95	EPC	P95	5 min

Alerts

alerts.yaml

```

apiVersion: 1
groups:
  - orgId: 1
    name: []
    folder: []
    interval: 1m
    rules:
      - uid: alert-lowDiskSpace
        title: [] 90%
        condition: C
        data:
          - refId: A
            datasourceUid: omnicores-prometheus
            model:
              expr: |
                100 - ((node_filesystem_avail_bytes{job="Node
Exporter",mountpoint="/" } * 100)
                  / node_filesystem_size_bytes{job="Node
Exporter",mountpoint="/"})
                # ... []
            noDataState: OK
            execErrState: Error
            for: 5m
            annotations:
              summary: [] 90%
            labels:
              type: resource

```

[]

[]

```
#  group_vars/all.yml  
```

```
monitoring_data:
  contactPoints:
    - orgId: 1
      name: 
      receivers:
        # Slack 
        - uid: slack-alerts
          type: slack
          disableResolveMessage: false
          settings:
            url: "https://hooks.slack.com/services/xxx/yyy/zzz"

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

Slack Google Chat Pushover

```
roles/monitoring/templates/grafana/contact-points.yaml.j2
notifications.pushover
enabled
api_token
user_key
```

□ □ □ □

□ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □

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

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

部署

Prometheus 部署

配置文件 `/etc/prometheus/prometheus.yml`

配置项	值	说明
<code>scrape_interval</code>	<code>1m</code>	抓取间隔
<code>evaluation_interval</code>	<code>1m</code>	评估间隔
<code>scrape_timeout</code>	<code>50s</code>	抓取超时

Grafana 部署

配置文件 `/etc/grafana/grafana.ini`

配置文件 `/etc/grafana/provisioning`

配置项	值	说明
<code>allow_embedding</code>	<code>true</code>	允许 iframe 嵌入
<code>provisioning</code>	<code>/etc/grafana/provisioning</code>	配置目录

配置文件 `grafana.ini` 中的 `admin_password` 配置项，用于设置 Grafana 的 API 访问密码。

配置 (SMTP) 项

配置项 `group_vars` 中的 SMTP 配置项，位于 `roles/monitoring/templates/grafana/grafana.ini`。

項目	型	値	説明
grafana_smtp_user	文字列	(空)	SMTP 送信者アドレス 宛先メールアドレス 宛先
grafana_smtp_password	文字列	(空)	SMTP 送信者パスワード 宛先パスワード
grafana_smtp_host	文字列	in- v3.mailjet.com:587	SMTP 送信先:宛先
grafana_smtp_enabled	ブール値	true	SMTP 送信

```
from_address = from_name 宛先 customer_legal_name宛先 alerts+  
<customer_legal_name>@omnitouch.com.au 宛 Omnitouch Alerts -  
<customer_legal_name>宛
```

Loki 宛

宛先 宛先 Loki 宛先

InfluxDB 宛

宛先 /etc/influxdb/influxdb.conf

項目	型	値
nokia-monitor	autogen	30 宛
dra	autogen	365 宛
Omnicharge_TAP3	autogen	90 宛

部署

组件	端口	协议	说明
Grafana	3000	HTTP	可视化 UI 及 API
Prometheus	9090	HTTP	时序数据库
Loki	3100	HTTP	日志聚合
InfluxDB	8086	HTTP	InfluxDB API
节点导出器	9100	HTTP	节点指标
SNMP 导出器	9116	HTTP	SNMP 指标
日志导出器	9115	HTTP	日志数据
Alloy	12345	HTTP	Alloy UI 界面

部署

部署

- 部署 Grafana 访问 `http://<monitoring-ip>:3000`
- 部署 节点 时序数据库
- 部署 日志聚合

部署

- 部署 Grafana 访问 `http://<monitoring-ip>:3000`
- 部署 节点 日志 **Loki** 聚合
- 部署 LogQL 查询

```
# 主机名称
{hostname="customer-mme01"}

# MME 告警
{component="mme"} |~ "(?i)error"

# IMSI
{component="hss"} |= "123456789012345"
```

部署 Grafana

部署 Grafana UI 和告警模板 `group_vars/grafana/alerts/` 和告警规则 `roles/grafana/alerts/`

部署告警规则

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

部署

`roles/monitoring/templates/grafana/provisioning/alerting/alerts.yaml` 和告警规则

部署 Grafana

部署 Grafana UI 和告警模板

```
cd roles/monitoring
python3 export_grafana.py --customer <CUSTOMER> --url
http://<monitoring-ip>:3000
git add ../../hosts/Omnicores/<CUSTOMER>/group_vars/grafana/
git commit -m "XXXXXXXX"
```

```
--customer XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX group_vars/grafana/alerts/
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
roles/monitoring/templates/grafana/dashboards/XX
```

XXXX

Prometheus XXXX

XX: XXXXX "XXX" XXXXXXXX DOWN

XXXX:

- XXXXXXXXXXXX
- XXXXXXXXXXXX
- XXXXXXXX

XXXX:

1. XXXXXXXXXXXX

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

2. XXXXXXXXXXXX

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

3. XX Prometheus XXXX: http://<monitoring-ip>:9090/targets

安裝 Grafana

目標: 在 Ubuntu 系統上安裝 Grafana

步驟:

- 更新系統
- 安裝
- 配置 JSON 文件

步驟:

1. 安裝 Grafana 依賴包: `sudo apt-get install libfontconfig libssl-dev`
2. 安裝 Grafana 包: `sudo apt-get install grafana`
3. 啟動 Grafana 服務

安裝 Grafana

目標: 在 Ubuntu 系統上安裝 Grafana

步驟:

- 更新系統
- 安裝
- 配置 JSON 文件

步驟:

1. 安裝 Grafana 依賴包: `sudo apt-get install libfontconfig libssl-dev`
2. 安裝 Grafana 包: `sudo apt-get install grafana`
3. 啟動 Grafana 服務

Grafana 安裝指南

目標: Grafana 安裝 "Grafana" 包

步驟:

- Prometheus/Loki/InfluxDB 部署
- 部署 Prometheus
- 部署 Loki

部署:

1. 检查服务状态

```
systemctl status prometheus loki influxdb
```

2. 检查端口监听

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

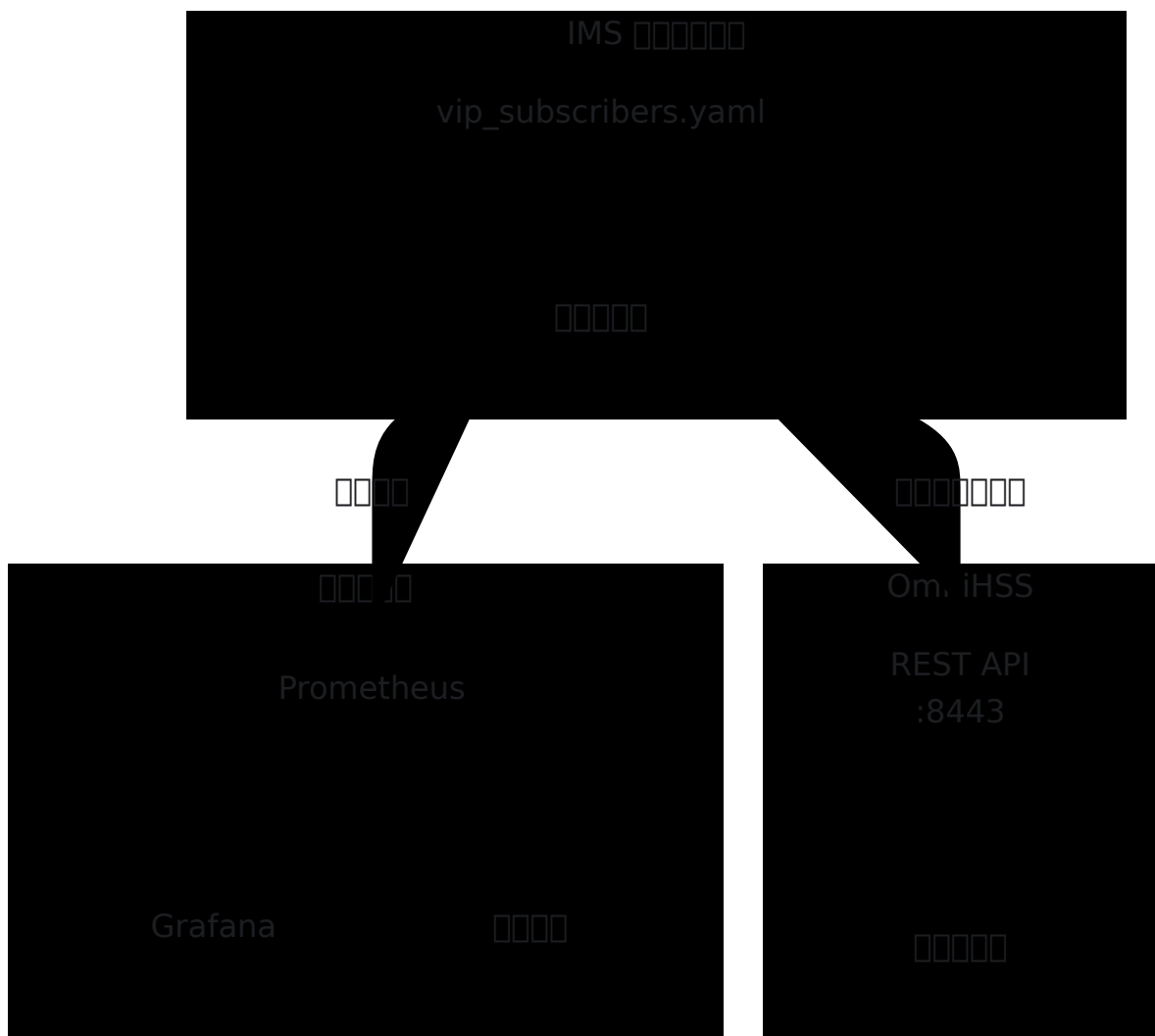
3. 检查配置

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

VIP 部署

VIP 部署需要部署 IMS 和 EPC 和 UE 部署需要部署 IMS 和 EPC 和 UE

11



□ □ □ □

□ □

모드	IMS 모드	EPC 모드	비고
full	지원	지원	모든 기능을 지원합니다.
voip_only	지원	지원	VoIP 서비스와 IP 서비스를 지원합니다.
data_only	지원	지원	데이터 서비스와 M2M을 지원합니다.

□□□□:

- **full:** IMS assigned_scscf null EPC last_seen_mme null
- **voip_only:** IMS assigned_scscf null
- **data_only:** EPC last_seen_mme null

VIP

hosts/<customer>/group_vars/vip_subscribers.yaml

```

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

#
scrape_interval_seconds: 30
ping_timeout_seconds: 2
ping_count: 2

#
subscribers:
  # (IMS + EPC)
  - imsi: "0010100000000001"
    label: ""
    type: "full"

  - imsi: "0010100000000002"
    label: ""
    type: "full"

  # ( - IMS)
  - imsi: "0010100000000003"
    label: ""
    type: "data_only"

  # VoIP (IMS)
  - imsi: "0010100000000004"
    label: "IP"
    type: "voip_only"

```

配置

項目	単位	デフォルト	説明
hss_url	URL	-	OmniHSS REST API URL (HTTPS)
scrape_interval_seconds	秒	30	スクレイピング間隔
ping_timeout_seconds	秒	2	UE IP ping タイムアウト
ping_count	回	2	ping 回数
blackbox_exporter_url	URL	-	ping 結果を出力する URL (例: http://pcscf01:9115) に ICMP ping 結果を送信
subscribers	数	-	サブスクリプション数

出力

項目	単位	デフォルト	説明
imsi	文字列	-	IMSI (15 文字)
label	文字列	IMSI	ラベル
type	文字列	full	タイプ: full, voip_only, data_only

例: MSISDN 例 HSS API 例

00

000000 9550 0 /metrics 000000

0000

00	00	00
<code>vip_subscriber_service_healthy</code>	Gauge	00000000000000000000 1000 0 0
<code>vip_subscriber_ims_registered</code>	Gauge	00000000 S-CSCF0000 10000 0
<code>vip_subscriber_epc_registered</code>	Gauge	00000000 MME 000000 10000 0
<code>vip_subscriber_ue_ip_reachable</code>	Gauge	00 UE IP 0 ping 000000 10000 0
<code>vip_subscriber_hss_reachable</code>	Gauge	00 HSS API 0000000000 10000 0
<code>vip_subscriber_enabled</code>	Gauge	00000000 HSS 0000000 10000 0

0000

00	00	00
<code>vip_subscriber_ims_registration_age_seconds</code>	Gauge	0000 IMS 000000 00 (-1 000000)
<code>vip_subscriber_epc_registration_age_seconds</code>	Gauge	0000 EPC 000000 000000 (-1 0000 00)

接口

接口	类型	说明
<code>vip_subscriber_info</code>	Gauge	当前 1 分钟内活跃用户数

接口参数:

参数	描述	示例
<code>imsi</code>	用户 IMSI	<code>001010000000002</code>
<code>label</code>	用户标识	<code>000000</code>
<code>msisdn</code>	手机号 (HSS)	<code>61400000002</code>
<code>type</code>	数据类型	<code>full</code> <code>voip_only</code> <code>data_only</code>

`vip_subscriber_info` 接口参数:

参数	描述	示例
<code>healthy</code>	健康状态	<code>1</code> <code>0</code>
<code>ims</code>	IMS 状态	<code>1</code> <code>0</code>
<code>epc</code>	EPC 状态	<code>1</code> <code>0</code>
<code>ping</code>	UE IP 地址	<code>1</code> <code>0</code>
<code>assigned_scscf</code>	S-CSCF 地址	<code>scscf01.ims.example.com</code>
<code>last_seen_mme</code>	MME 地址	<code>mme02.epc.example.com</code>
<code>ue_ip</code>	UE 设备 IP 地址	<code>100.72.83.20</code>

接口

```
# IMS VIP 健康状態
count(vip_subscriber_service_healthy == 1)

# IMS VIP 不健康状態
count(vip_subscriber_service_healthy == 0)

# VoIP 専用 IMS 健康状態
vip_subscriber_ims_registered{type=~"voip_only|full"}

# データ専用 EPC 健康状態
vip_subscriber_epc_registered{type=~"data_only|full"}

# IMS 登録状態 (>1 分)
vip_subscriber_ims_registration_age_seconds > 3600
```

ダッシュボード

IMS VIP 健康状態 IMS VIP 登録状態

パス: Grafana → IMS → IMS VIP ダッシュボード

ダッシュボード URL: </d/ims-vip-subscribers/>

ダッシュボード

項目	説明
概要	概要説明
構成	構成説明
IMS 設定	IMS 設定説明
EPC 設定	EPC 設定説明
UE 設定	UE ping 設定説明
その他	その他 VIP 設定説明
VIP 詳細	VIP 詳細説明
関連項目	関連項目説明

概要

VIP 概要説明

VIP 概要

項目: VIP 概要説明 項目: 項目: 1 項目: vip_subscriber_service_healthy == 0

項目:

- 項目: VIP 項目 {{ \$labels.label }} 項目
- 項目: 項目
 - VoIP 項目: 項目 MSISDN
 - 項目: 項目 IMSI
 - 項目: 項目MSISDN 項目 IMSI

項目項目:

- IP (voip_only) MSISDN: 61400000001
- (data_only) IMSI: 001010000000003
- (full) MSISDN: 61400000002 IMSI: 001010000000002

配置 VIP

1. 创建配置文件

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

2. 部署配置文件

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

3. 验证配置

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

部署

- : Loki Alloy
- :
- :

Netplan 簡介

簡介

OmniCore 透過 netplan 配置網路設備

- 網路介面 (eth0)
- 網路 IP 地址
- 網路子網

Netplan 配置

透過 netplan 配置 `group_vars` 透過 Jinja2 模板 `netplan_config` 生成

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

透過 `hosts/<customer>/group_vars/netplan.yaml.j2` 生成

配置

透過 `netplan.yaml.j2` 模板生成 `hosts/<customer>/group_vars/` 目錄下的 `netplan.yaml.j2` 模板 - 透過 DNS 配置網路設備

透過 `netplan.yaml.j2` 模板生成 `hosts/<customer>/group_vars/` 目錄下的 `netplan.yaml.j2` 模板 - 透過 DNS 配置網路設備

```

network:
  version: 2
  ethernet:
    # 主机 - 主机名 ansible_host 网关
    eth0:
      addresses:
        - "{{ ansible_host }}/{{ mask_cidr | default(24) }}"
      nameservers:
        addresses:
{% if 'dns' in group_names %}
        # 主机 DNS 服务器 DNS 服务器
        - 8.8.8.8
{% else %}
        # 主机 'dns' 主机 DNS 服务器
{% for dns_host in groups['dns'] | default([]) %}
        - {{ hostvars[dns_host]['ansible_host'] }}
{% endfor %}
{% endif %}
        # DNS 服务器 - 主机名
        search:
        - customer.example
      routes:
        - to: "default"
          via: "{{ gateway }}"

{% if secondary_ips is defined %}
  # 主机 - 主机名 secondary_ips 主机
  # 主机名 ens19ens20ens21...ens18 + loop.index主机名
  # 主机名主机名“主机名”
{% for nic_name, nic_config in secondary_ips.items() %}
  ens{{ 18 + loop.index }}:
    addresses:
      # 主机名 NIC 主机名 mask_cidr主机名 24
      - "{{ nic_config.ip_address }}/{{ nic_config.mask_cidr |
default(24) }}"
{% if nic_config.routes is defined %}
      # 主机名 - 主机名
      routes:
{% for route in nic_config.routes %}
      - to: "{{ route }}"
        via: "{{ nic_config.gateway }}"
{% endfor %}
{% endif %}

```

```
{% endfor %}
{% endif %}
```

#####

- `ansible_host` 与 `gateway` 指定网关地址
- DNS 指定 `dns` 指定 DNS 地址
- DNS `search` 指定 DNS 搜索域
- 指定网卡 `ens19` 与 `ens20` ... 指定网卡
- 指定 IP 地址 `mask_cidr` 指定子网掩码

#####

网卡 (eth0) 指定子网掩码

- `ansible_host` - IP 地址
- `gateway` - 网关
- `mask_cidr` - 子网掩码 24

DNS 指定 DNS 地址

- `dns` 指定 DNS 地址 `ansible_host` IP
- 指定 DNS 地址 `8.8.8.8`

MTU (`link_mtu`)

`common` 指定 `link_mtu` 指定 netplan 指定 **MTU** 指定 netplan drop-in (`/etc/netplan/999-link-mtu.yaml`) 指定 VMware/Proxmox 指定 MTU 指定 jumbo/9000 指定 PMTUD 指定 SCTP/SIP 指定

指定 drop-in 指定 `999-` 指定 netplan 指定 `/etc/netplan/*.yaml` 指定 `mtu` 指定 `80-ansible-config.yaml` 指定 VMware 指定 `99-netcfg-vmware.yaml` 指定 drop-in 指定

```
all:
  vars:
    link_mtu: 1500    # 物理 NIC の MTU 1500
```

link_mtu 物理 NIC の MTU

link_mtu 物理 NIC の MTU 1400/1430/1300
packet-core 物理 NIC の MTU 1500 link_mtu 物理 NIC の MTU

物理 NIC

物理 NIC IP secondary_ips

物理 NIC

```
secondary_ips:
  <logical_name>:
    ip_address: <ip_address>
    mask_cidr: <prefix_length>    # 物理 NIC の MTU 24
    gateway: <gateway_ip>
    host_vm_network: <proxmox_bridge>
    vlanid: <vlan_id>
    nic_name: <interface_name>    # 物理 NIC
    routes:                        # 物理 NIC
      - '<destination_cidr>'
      - '<destination_cidr>'
    table_routes:                  # 物理 NIC
      - { to: '<cidr>', via: '<gateway>', table: <table_id> }
    routing_policy:                # 物理 NIC
      from: '<source_cidr>'
      via: '<gateway>'
      table: <table_id>
      priority: <priority>        # 物理 NIC 100
```

mask_cidr 物理 NIC IP (nic_config.mask_cidr)
物理 NIC mask_cidr /24

□□□□

```
dra:
  hosts:
    <hostname>:
      ansible_host: 10.0.1.100
      gateway: 10.0.1.1
      host_vm_network: "ovsbr1"
      vlanid: "100"
      netplan_config: netplan.yaml.j2
      secondary_ips:
        public_ip:
          ip_address: 192.0.2.50
          mask_cidr: 28
          gateway: 192.0.2.1
          host_vm_network: "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'
```

□□□ Netplan □□

□□□□□□□□□□□□□□ □DNS search □□ ens19/ens20 □□□□□□□□□□

```

network:
  version: 2
  ethernets:
    eth0:
      addresses:
        - "10.0.1.100/24"
      nameservers:
        addresses:
          - 10.0.1.53
        search:
          - customer.example
      routes:
        - to: "default"
          via: "10.0.1.1"
    ens19:
      addresses:
        - "192.0.2.50/28"
      routes:
        - to: "198.51.100.0/24"
          via: "192.0.2.1"
        - to: "203.0.113.0/24"
          via: "192.0.2.1"
    ens20:
      addresses:
        - "172.16.50.10/24"
      routes:
        - to: "172.17.0.0/16"
          via: "172.16.50.1"

```

public_ip 000000 /28 0000000000 mask_cidr: 28 00 peering_ip 0000 /24 0000

Proxmox 00

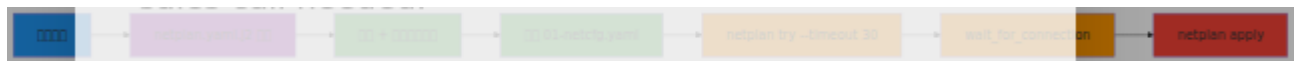
00 proxmox.yml 000000 NIC 000000000000

1. 00000000000000000000 NIC
2. 000000000000 NIC000000000000

Proxmox 000000

- `host_vm_network` - VM NIC 名称
- `vlanid` - VM VLAN ID

手順



1. `/etc/netplan` ディレクトリに `netplan` ディレクトリを作成し、`(*.bak)` のバックアップを作成する
2. Jinja2 テンプレート `/etc/netplan/01-netcfg.yaml` を `0600` のパーミッションで作成する
3. `netplan try --timeout 30` を実行して、ネットワーク設定を確認する
4. `wait_for_connection` を実行して、ネットワーク接続を確認する
5. `netplan apply` を実行して、ネットワーク設定を適用する

結果

IP 設定: Diameter Edge Agent (DEA)

```

<hostname>:
  ansible_host: 10.0.1.100          # VM IP
  gateway: 10.0.1.1
  netplan_config: netplan.yaml.j2
  secondary_ips:
    diameter_roaming:
      ip_address: 192.0.2.50        # 仮想マシン IP
      gateway: 192.0.2.1
      host_vm_network: "vmbr0"
      vlanid: "200"
      routes:
        - '198.51.100.0/24'        # 仮想マシン IP
  
```

📄 S5/S8 📄📄 PGW

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

📄📄📄📄📄📄📄📄📄📄📄📄📄📄

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

📄📄📄📄📄📄 IP

📄📄📄📄 Jinja2 📄📄📄📄📄📄📄 IP 📄📄

Inventory

Inventory IP addresses are defined in `inventory_hostname`

```
# Secondary IP
{{ secondary_ips.diameter_public_ip.ip_address }}
```

```
# Map inventory_hostname to secondary_ips
{{ hostvars[inventory_hostname]['secondary_ips']
  ['diameter_public_ip']['ip_address'] }}
```

```
# Gateway
{{ secondary_ips.diameter_public_ip.gateway }}
```

```
{{ secondary_ips.diameter_public_ip.vlanid }}
```

Groups

Groups are defined in `hostvars`

```
# Map dra to secondary_ips
{{ hostvars[groups['dra'][0]]['secondary_ips']
  ['diameter_public_ip']['ip_address'] }}
```

```
# Map DRA to secondary_ips IP
{% for host in groups['dra'] %}
{% if hostvars[host]['secondary_ips'] is defined %}
  - {{ hostvars[host]['secondary_ips']['diameter_public_ip']
    ['ip_address'] }}
{% endif %}
{% endfor %}
```

Groups **DRA** Groups

Groups are defined in `IP`

```
# dra_config.yaml.j2 - {{ inventory_hostname }} peers:
- name: external_peer
  # {{ ansible_host }} IP
  local_ip: {{ hostvars[inventory_hostname]['secondary_ips']
['diameter_public_ip']['ip_address'] }}
  remote_ip: 198.51.100.50
  port: 3868
```

Define IP

Define IP

```
{% 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 %}
```

Define IP

Define IP

SSH

```
ip link show
```

Define IP

```
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> ...
```

📄 Netplan 📄

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

📄📄📄 Netplan

```
netplan apply
```

📄 Netplan

```
netplan --debug apply
```

📄📄📄

```
ip route show
```

📄📄📄📄

- 📄📄📄📄 - 📄📄📄📄
- Proxmox VM/LXC 📄 - 📄📄📄📄
- 📄📄📄 - 📄📄📄📄

OmniHSS 開箱評測

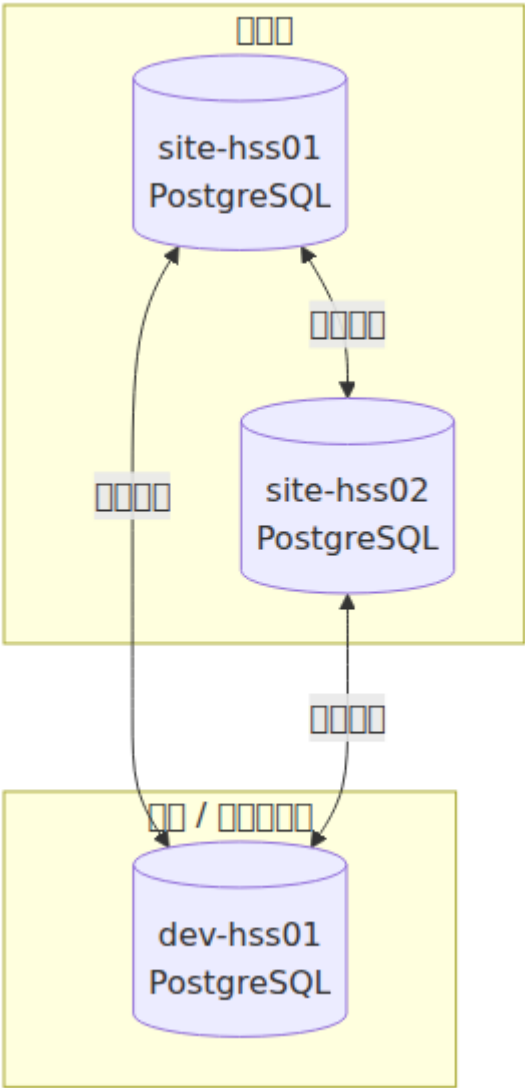
OmniHSS 是基於 OpenHSS 開發的 IMS 系統，基於 PostgreSQL 資料庫。
PostgreSQL 是基於 OmniHSS 開發的 S6a/Cx/Dx 及 Sh 系統。
OmniHSS 是基於 OpenHSS 開發的 IMS 系統，基於 PostgreSQL 資料庫。

簡介

- 基於 OpenHSS 開發
- 基於 PostgreSQL 資料庫
- 基於 S6a/Cx/Dx 及 Sh 系統
- 基於 OmniHSS 開發
- 基於 OpenHSS 開發
- 基於 PostgreSQL 資料庫
- 基於 S6a/Cx/Dx 及 Sh 系統
- 基於 OmniHSS 開發
- 基於 OpenHSS 開發
- 基於 PostgreSQL 資料庫
- 基於 S6a/Cx/Dx 及 Sh 系統

安裝

OmniHSS 是基於 PostgreSQL 資料庫，安裝時需要指定 origin
= none 安裝時需要指定 updated_at 安裝時需要指定 LWW 安裝

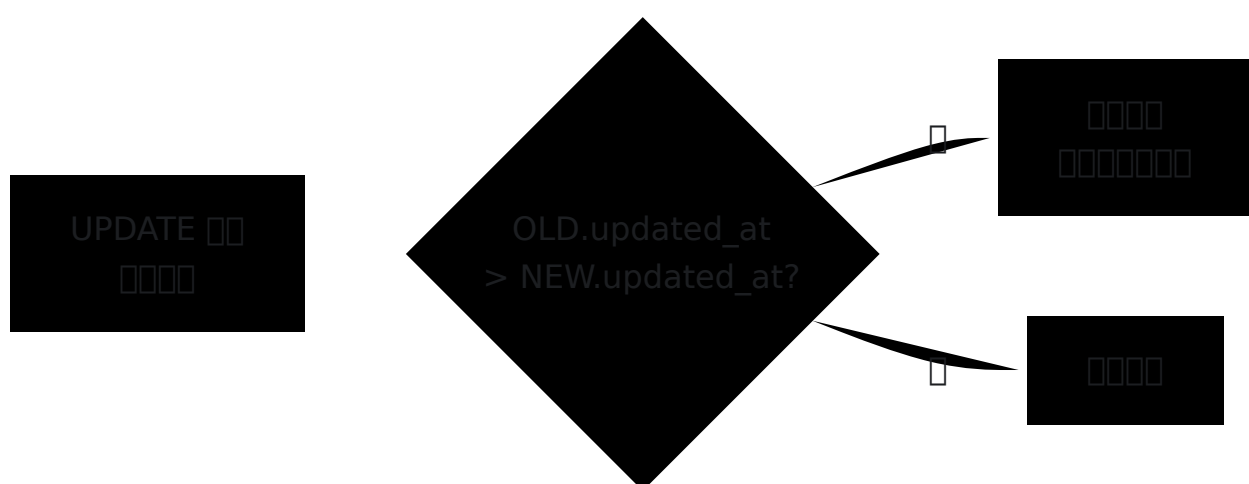


subscriber database (N) replicates data from the subscriber database (N-1) to the subscriber database (N-1) database.

Subscriber

Subscriber	Subscriber	Subscriber
site / subscriber	subscriber, key_set (AuC), sim, msisdn	subscriber database
subscriber	subscriber_state, pdn_session, lte_call, ims_subscription	subscriber database updated_at
subscriber	schema_migrations	subscriber

public schema_migrations LWW updated_at



Schema Migration

OmniHSS

1. hss OmniHSS hss01 / hss02 connected_omnihss
2. connected_omnihss OmniHSS hss

OmniHSS omnihss.replication.enabled true

connected_omnihss

Schema Migration

Schema Migration

項目	名前	データ型	コメント
自己公開	<code><self>_pub</code>	boolean	
サブスクリプション元	<code>sub_from_source</code>	integer	
サブスクリプション元名	<code><subscriber>_from_source</code>	integer	

この表は PostgreSQL のデータベースに作成されたものです。
 テーブルのスキーマは `"table_schema"` のように表示されます。

項目 `--limit` の説明

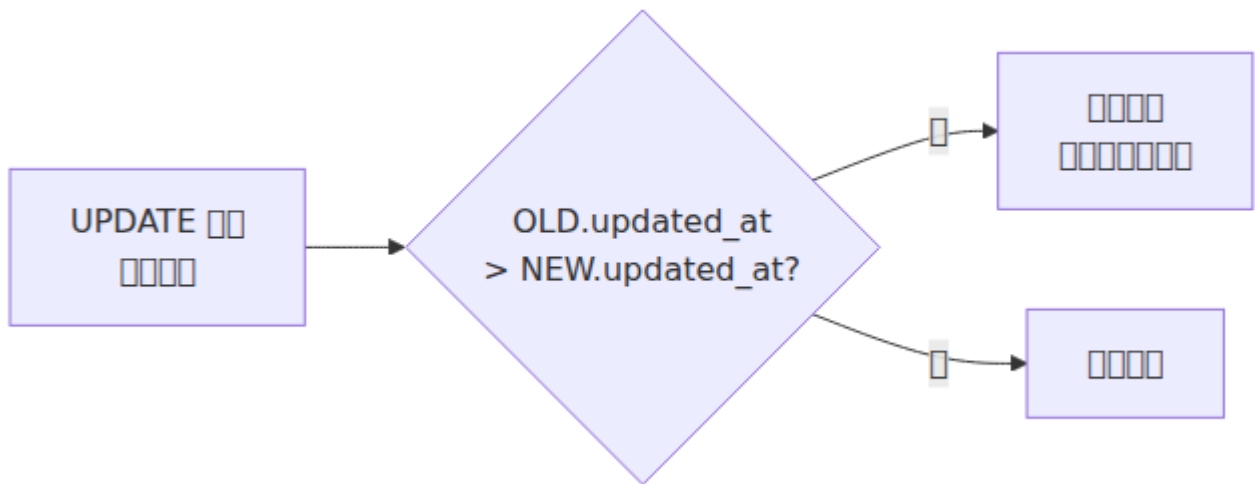
この項目は、`hss` と `connected_omnihss` の合計値を示します。

- `<self>_pub` は、自己公開の有無を示します。
- `sub_from_source` は、サブスクリプション元を示します。
- `<subscriber>_from_source` は、サブスクリプション元名を示します。

この `connected_omnihss` は、`hss` の値を示します。
 テーブルのスキーマは `"table_schema"` のように表示されます。

- `--limit` は、`groups['hss']` の `connected_omnihss` の値を `--limit` の値で制限します。
`--limit hss01` は、`hss01` の値を `hss01` の値で制限します。
`hss02/hss03` は、`--limit` の値を `hss02/hss03` の値で制限します。
- この項目は、`connected_omnihss` の値を示します。

この項目は、`connected_omnihss` の値を示します。
 テーブルのスキーマは `connected_omnihss` のように表示されます。



表

记录

omnihss.replication.enabled 记录组 group_vars 表

```

omnihss:
  database_type: "postgres"
  database_host: "localhost"
  database_port: 5432
  database_name: "omnihss"
  database_username: "hss"
  database_password: "password"
  replication:
    enabled: true
  
```

項目	必須 項目	オプション 項目	値	説明
database_type	必須	オプション	postgres	データベースの種類 postgres
database_host	必須	オプション	localhost	データベースのホスト localhost
database_port	必須	オプション	5432	PostgreSQL のポート番号
database_name	必須	オプション	omnihss	データベースの名前
database_username	必須	オプション	hss	データベースのユーザー名 REPLICATION ユーザー
database_password	必須	オプション	password	データベースのパスワード
replication.enabled	必須	オプション	false	レプリケーションの有効化 true PostgreSQL のレプリケーションの有効化

replication.enabled true PostgreSQL のレプリケーションの有効化

項目	値
wal_level	logical
max_wal_senders	10
max_replication_slots	10
listen_addresses	*
max_slot_wal_keep_size	10GB omnihss.replication.max_slot_wal_keep_size

pg_hba.conf 接続許可する IP /32 all replication

接続する (connected_omnihss)

OmniHSS remote_peers_file 接続する
prod_master_peers.yml

connected_omnihss:

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

| 項目   | 形式  | 単位 | 初期値  | 説明                                                                      |
|------|-----|----|------|-------------------------------------------------------------------------|
| name | 文字列 | 必須 | -    | inventory_hostname 形式で、sub_from_<name> 形式で、<name>_pub 形式で、または 形式で指定します。 |
| host | 文字列 | 必須 | -    | IP 形式で指定します。                                                            |
| port | 整数  | 必須 | 5432 | PostgreSQL データベースの database_port                                        |

```

connected_omnihss ##### OmniHSS
hss

```

## OmniHSS

[illegible]



```
connected_omnihss:
 - name: site-hss03
 host: 10.4.2.144
 port: 5432
 #
```

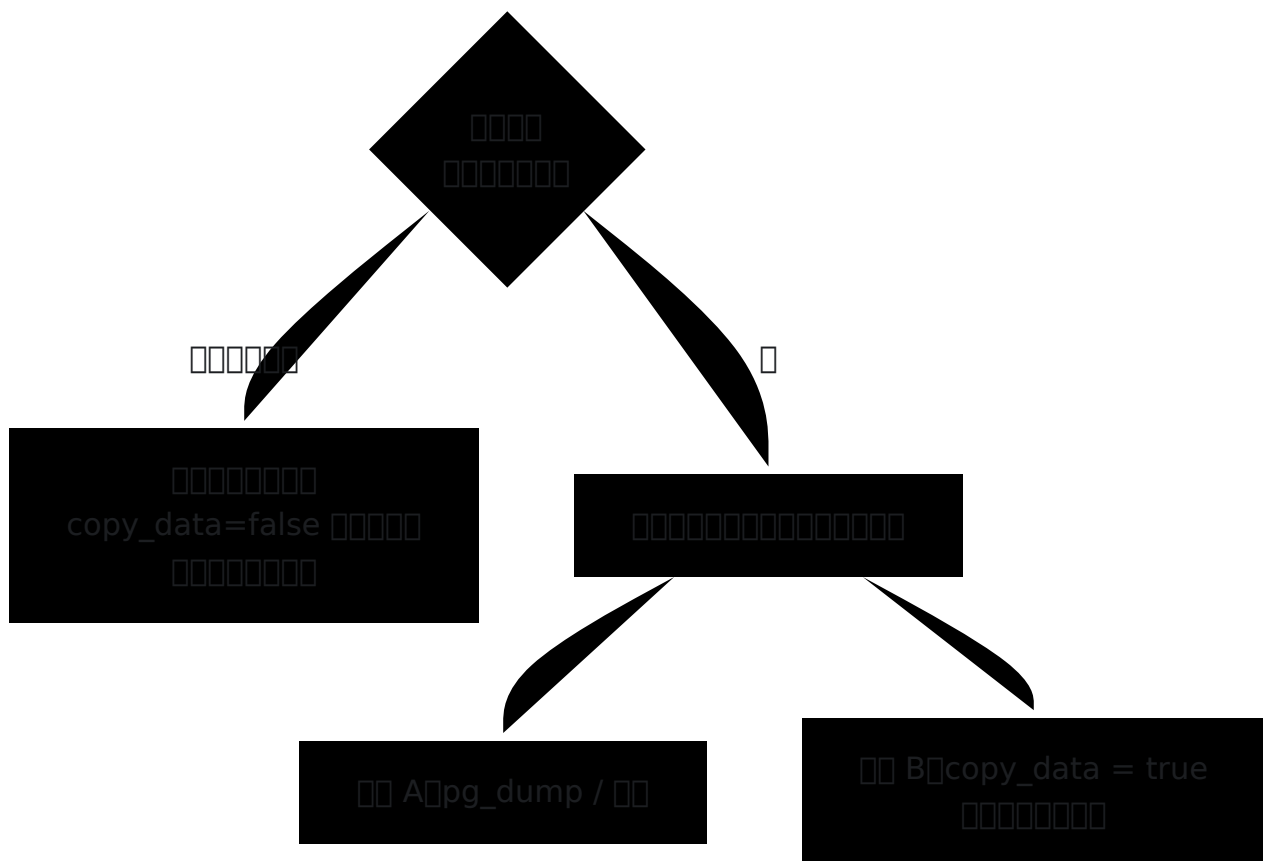
### 3. 3. OmniHSS

OmniHSS PostgreSQL OmniHSS PostgreSQL

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

pg\_hba.conf ALTER SUBSCRIPTION ... REFRESH PUBLICATION

copy\_data = false



|  |  |  |  |  |  |  |  |
|--|--|--|--|--|--|--|--|
|  |  |  |  |  |  |  |  |
|--|--|--|--|--|--|--|--|

OmniHSS

**Answer**

/

1. 创建数据库并安装 OmniHSS 数据库 `pg_dump` 数据库 `--no-publications --no-subscriptions`

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

```
backups/<Site Name>/hss dump <hostname> <timestamp>.sql
```

2. 確認する 確認する OmniHSSデータベースのバックアップが完了していることを確認する

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

データベース 1 の SQL データベースのバックアップが完了していることを確認する

3. 確認する 確認する `copy_data = false` データベースのバックアップが完了していることを確認する

データベースのバックアップが完了していることを確認する  
[subscriber\_state, pdn\_session, lte\_call, ims\_subscription] データベース  
データベースのバックアップが完了していることを確認する

## DB B データベースのバックアップ

`seed_hss_replication.yml` データベースのバックアップが完了していることを確認する  
データベース `copy_data = false` データベースのバックアップが完了していることを確認する  
データベース 1:1 データベースのバックアップが完了していることを確認する  
FK データベースのバックアップが完了していることを確認する  
API の S6a データベースのバックアップが完了していることを確認する  
UPDATE データベースのバックアップが完了していることを確認する  
[pdn\_session, lte\_call] データベースの COPY データベースのバックアップが完了していることを確認する  
データベースのバックアップが完了していることを確認する

データベースのバックアップが完了していることを確認する

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

データベースのバックアップが完了していることを確認する  
1) dev-hss01 (10.4.10.142) - 838 データベース  
データベースのバックアップが完了していることを確認する

データベースのバックアップが完了していることを確認する

```

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

```

### HSS #####

## --limit ##### A

##### slot\_name #####

```

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

```

```
2. 表を複製する -t オプション
schema_migrations, pdn_session, lte_call - 複製
subscriber_state
複製するテーブル
subscriber_state_id 列を API/S6a 列に複製する
#
テーブルを UPDATE
複製するテーブルの origin=none
複製するテーブルの WAL 複製する
#
テーブルを複製する
sudo -u postgres psql -d omnihss -c \
 "SELECT pg_replication_origin_create('omnihss_seed_load');"
{ echo "SELECT
pg_replication_origin_session_setup('omnihss_seed_load');"
 PGPASSWORD=<pw> pg_dump -h 10.4.10.142 -U hss -d omnihss \
 --data-only --inserts --on-conflict-do-nothing --disable-
triggers \
 -t subscriber -t subscriber_state -t key_set -t sim -t msisdn
... ; } \
| sudo -u postgres psql -d omnihss
sudo -u postgres psql -d omnihss -c \
 "SELECT pg_replication_origin_drop('omnihss_seed_load');"

```

## 確認

複製されたテーブルを確認する

- テーブルの複製を確認する
- テーブルの複製を確認する
- テーブルの WAL サイズを確認する (100 MB)
- テーブルの WAL サイズを確認する (reserved, extended, lost)

複製されたテーブルを確認する OmniHSS テーブルを確認する

```
pid - pid
sudo -u postgres psql -d omnihss -c \
 "SELECT subname, pid IS NOT NULL AS worker_alive FROM
pg_stat_subscription;"

active - active wal_status reserved/extended
sudo -u postgres psql -d omnihss -c \
 "SELECT slot_name, active, wal_status, \
 pg_size_pretty(pg_wal_lsn_diff(pg_current_wal_lsn(),
confirmed_flush_lsn)) AS lag \
 FROM pg_replication_slots WHERE slot_type = 'logical';"
```

| 項目                   | 説明                               |
|----------------------|----------------------------------|
| worker_alive = false | サブスクリプションがアクティブかどうかを示すフラグ        |
| active = false       | サブスクリプションがアクティブかどうかを示すフラグ        |
| lag                  | サブスクリプションとWALとの遅延時間              |
| wal_status = lost    | PostgreSQLのWALが失われた状態であることを示すフラグ |

health\_check.yml HSS Postgres の健康状態を確認するためのスクリプト。WALの状態を確認する。

## 環境構築

session\_replication\_role = replica を設定し、FOREIGN KEY を利用してPostgreSQLのFKを定義する。

環境構築 - 環境構築の完了を確認する。

- A epc\_profile X ON DELETE RESTRICT A X
- B 203 X B

- FK□□□□□□□□□□□□□□□□“203 → X” □ “X□□□”□□□□□□□□□□□□  
FK□□□□□□

```
subscriber.epc_profile_id HSSULR/AIR
DIAMETER_ERROR_USER_UNKNOWN (5001)
/
```

|  |  |  |  |  |  |  |  |
|--|--|--|--|--|--|--|--|
|  |  |  |  |  |  |  |  |
|--|--|--|--|--|--|--|--|

```
omnihss_fk_orphan_count() FK SECURITY DEFINER
postgres_exporter Grafana
```

□ □ □ □ □ □ □ □

```
sudo -u postgres psql -d omnihss -tAc "SELECT
omnihss fk orphan count();" # 0 = []
```

☐☐☐☐☐☐☐☐☐☐☐☐☐☐☐☐EPC☐☐☐☐☐☐

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

[illegible]

□ □ □ □ □ □

[illegible]

- 0000000000 000000000000000000OmniHSS000000000000000000000000
- 000000 00000 omnihss\_schema\_max\_version /  
omnihss\_schema\_migration\_count0000000000000000000000000000000000
- 0000000000 0000 omnihss\_subscription\_error\_apply\_error\_count /  
sync error count 000000000000000000000000“HSS0000/0000”0000

# 準備

pg\_hba.conf 設定を再読み込みする

接続を再試行する

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

pg\_hba.conf を再読み込みする

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

connected\_omnihss  
hss を再起動する

# バックアップ

OmniHSS をバックアップする

| 操作     | コマンド                           | 説明                                                                                                          |
|--------|--------------------------------|-------------------------------------------------------------------------------------------------------------|
| バックアップ | services/backup.yml            | hss_dump_<host>_<ts>.sql<br>hss_<host>_<ts>.tar.gz<br>/etc/omnihss<br>backups/<Site_Name>/ にバックアップ<br>を保存する |
| リストア   | util_playbooks/restore_hss.yml | SQL データベースを OmniHSS<br>にリストアする                                                                              |

バックアップを確認する

11

OmniHSS postgres\_exporter PostgreSQL Prometheus

```
📊: pg_replication_slots_active 📊: Gauge 📊: ████████████████████ 📊:
```

- slot\_name: □□□□□□□□□□□□□□□□

```
pg_stat_replication_* / Gauge pg_stat_replication_wal_lsn_diff
```

□□□□□

```
pg_replication_slots_active > 0
count(pg_replication_slots_active == 0)

pg_replication_slots_active < 0
```

# OmniHSS

```
postgres_exporter
```

```
roles/omnihss/files/postgres_exporter_queries.yaml OmniHSS
```

 Prometheus
  HSS
  Postgres
  HSS
  Postgres

```
roles/monitoring/templates/grafana/rules/
```

| メトリック名                                                         | タイプ     | 単位                       |
|----------------------------------------------------------------|---------|--------------------------|
| omnihss_subscription_worker_alive{subname}                     | Gauge   | 0=0<br>1=1               |
| omnihss_replication_slot_active{slot_name}                     | Gauge   | 0=0<br>1=1               |
| omnihss_replication_slot_wal_status_code{slot_name}            | Gauge   | 0=1<br>1=2<br>2=3<br>3=4 |
| omnihss_replication_slot_lag_bytes{slot_name}                  | Gauge   | 0=0<br>1=1               |
| omnihss_subscription_error_apply_error_count{subname}          | Counter | 0=0<br>1=1               |
| omnihss_subscription_error_sync_error_count{subname}           | Counter | 0=0<br>1=1               |
| omnihss_integrity_fk_orphans                                   | Gauge   | 0=0<br>1=1               |
| omnihss_schema_max_version /<br>omnihss_schema_migration_count | Gauge   | 0=0<br>1=1               |

メトリック名: 00000000 00000000000000

## 問題

### 問題の概要

問題: PostgreSQLのバックアップが失敗した

原因:

- PostgreSQLのバックアップ時に `copy_data = false` が設定されていた

解決:

1. PostgreSQLのバックアップを再実行する

### 解決方法

問題: `pg_stat_subscription` テーブルに `pid IS NULL` のレコードがある

原因:

- PostgreSQLのバージョンアップ/ダウン
- PostgreSQLの `pg_hba.conf` ファイルの修正
- PostgreSQLの `database_username / database_password` の設定が正しくない

解決:

1. PostgreSQLのポート番号を `5432` に変更する
2. PostgreSQLの `connected_omnihs` テーブルに `hss` のレコードを追加する `pg_hba.conf` ファイル
3. `database_username / database_password` の設定を確認する

### 問題の解決方法 `wal_status = lost`

問題: `wal_status` が `lost` になっている

原因:

- PostgreSQLのバックアップ時にWALが削除された

□□□□:

1. □□□□□□□□□□□□□□□□ □□□□□□□□

□□□□□□□□□□□□

□□: □□□ `copy_data = true` □□□□□□□□□□□□□□□□

□□□□:

- □□□□□□□□□□□□□□□□□□□□□□□□□□□□

□□□□:

1. □□□□□□□□□□□□ `copy_data = true` □□□□□□□□□□ `copy_data = false`□

□□□□□□□□□□□□

□□: □□□□□□ `subscriber_state` □□□□□□□□□□□□□□□□“□□”□

□□□□:

- □□□□□□□□□□□□□□□□

□□□□: □□□□□□□□□□□□□□□□□□□□□□□□ `updated_at` □□□□□□□□□□□□□□□□NTP□□□□□  
`updated_at` □□□□□□□□□□

# Proxmox VM/LXC 簡介

目前使用 Proxmox 部署 OmniCore 使用 `util_playbooks/proxmox.yml` 的 `deployment_type` 使用 QEMU 部署 LXC 使用 `proxmoxServers` 使用 LXC 部署 UPF 使用 `UPF`

目前使用 VMware/HyperV 部署 Vultr / AWS / GCP 使用

目前

- 部署 LXC
- IP 地址
- Netplan 配置
- 网络

## LXC 与 VM

LXC 使用

- 部署 LXC
- 部署 LXC
- 部署 LXC
- 部署 LXC
- 部署 EPC/IMS 使用
- 部署 UPF 使用/TUN 使用

使用 KVM 使用

- 部署 LXC
- 部署 LXC/部署 LXC
- 部署 LXC
- 部署 LXC
- UPF 使用

目前

- `UPF`
- **LXC** `/apt-cache dns`

|  |  |  |  |  |  |
|--|--|--|--|--|--|
|  |  |  |  |  |  |
|--|--|--|--|--|--|

deployment\_type ████████████████████

1. `deployment_type` `proxmoxServers`
2. `deployment_type` `proxmoxServers`
3. `vm`

proxmoxServers deployment\_type: lxc  
UPF LXC UPF/TUN

```
all:
 vars:
 proxmoxServers:
 pve-node-01:
 deployment_type: lxc # 物理機 → 虛擬機 LXC
 # ...

upf:
 hosts:
 site-upf01:
 ansible_host: 192.168.1.24
 deployment_type: vm # 物理機 → 虛擬機 VM
```

[illegible]

☐ ☐ ☐ ☐ ☐ ☐ ☐ ☐ ☐ ☐ ☐ ☐ ☐ ☐ ☐ ☐ ☐ ☐ ☐ ☐ ☐ ☐ ☐ ☐ ☐ LXC ☐ ☐ ☐ ☐

deployment\_type: vm

# Proxmox 部署

## 1. 部署 API 节点

```
Proxmox UI: 浏览器 → 节点 → API 节点
用户名: root@pam!<TokenName>
密码: 1234567890123456
节点名 `节点名` 节点
```

## 2a. 部署 Cloud-Init 模板

在 Proxmox 部署模板时，Ubuntu 模板需要指定 `proxmox.yml` 文件，cloud-init 通过 SSH 连接到模板

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

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

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

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

qm create $TEMPLATE_ID --name ubuntu-2404-template --memory 2048 --cores 2 --net0 virtio,bridge=vmbr0
qm importdisk $TEMPLATE_ID $IMAGE local-lvm
qm set $TEMPLATE_ID --scsihw virtio-scsi-pci --scsi0 local-lvm:vm-${TEMPLATE_ID}-disk-0
qm set $TEMPLATE_ID --ide2 local-lvm:cloudinit
qm set $TEMPLATE_ID --boot c --bootdisk scsi0
qm set $TEMPLATE_ID --vga std
qm set $TEMPLATE_ID --agent enabled=1
qm template $TEMPLATE_ID
```

000

- 0000000000000000 proxmox.yml 000000 ciuser cipassword sshkeys
- Cloud-init 000000 proxmoxTemplateUser 000000 proxmoxServers 0000→ 0000 local\_users 00
- Cloud-init 000000 proxmoxTemplatePassword 000000 proxmoxServers 0000→ cloud-init 00000000000000
- SSH 000000 local\_users. \*.public\_key 00000000000000000000000000000000
- 0000000000000000 local\_users 0000 cloud-init 0000000000 proxmoxTemplateUser 0000 proxmoxTemplatePassword 000000 proxmoxServers 000000000000000000
- --vga std 00 Proxmox Web 0000000000
- wget 0 -N 000000000000000000

000000 ISO 0000000

000000000000000000000000

00 1000 Web UI 000000

- 000000 → 0000 ID 90000000ubuntu-2404-template
- 00000000 Ubuntu Server ISO 000000 ISO
- 000000SCSI 0000VirtIO SCSI
- 000032GB0000SCSI
- CPU02 00
- 00002048 MB
- 0000VirtIO0000 vmbr0
- 0000000000 Ubuntu Server

00 200000000 - 000000

```
[] cloud-init
sudo apt update
sudo apt install cloud-init qemu-guest-agent -y

[]
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

[] bash []
history -c
sudo poweroff
```

### [] 3[] **Cloud-Init** []

- [] → [] → [] → CloudInit [] local-lvm
- [] → [] → QEMU [] → []
- [] → []
- [] cloud-init []/[] proxmox.yml [] proxmoxTemplateUser / proxmoxTemplatePassword [] local\_users []

## 2b. [] **LXC** [] **LXC** []

```
[] Proxmox [] shell []
pveam update
pveam download local ubuntu-24.04-standard_24.04-2_amd64.tar.zst
```

```
local:vztmpl/ubuntu-24.04-standard_24.04-2_amd64.tar.zst[]
[] proxmoxLxcOsTemplate []
```

ansible

proxmox

```
all:
 vars:
 # deployment_type [] → [] "vm"
 proxmoxServers:
 pve-node-01:
 proxmoxServerAddress: 192.168.1.100
 proxmoxServerPort: 8006
 proxmoxApiTokenName: ansible
 proxmoxApiTokenSecret: "your-token-secret-uuid"
 proxmoxNodeName: pve-node-01
 proxmoxTemplateName: ubuntu-2404-template
 proxmoxTemplateId: 9000
 proxmoxTemplateUser: omnitouch # [] cloud-init []
 proxmoxTemplatePassword: omnitouch # [] cloud-init []
 storage: local-lvm # []
 pve-node-02:
 # ... []

 local_users:
 admin_user:
 name: []
 public_key: "ssh-rsa AAAA..."

mme:
 hosts:
 site-mme01:
 ansible_host: 192.168.1.10
 vars:
 proxmox_interface: vmbr0
 gateway: 192.168.1.1
 mask_cidr: 24 # [] 24
 vlanid: 100 # []
```

## LXC 配置

```
all:
 vars:
 proxmox_lxc_nameserver: "1.1.1.1" # 指定DNS 1.1.1.1
 proxmoxServers:
 pve-node-01:
 deployment_type: lxc
 proxmoxServerAddress: 192.168.1.100
 proxmoxServerPort: 8006
 proxmoxApiTokenName: ansible
 proxmoxApiTokenSecret: "your-token-secret-uuid"
 proxmoxNodeName: pve-node-01
 proxmoxLxcOsTemplate: "local:vztmpl/ubuntu-24.04-
standard_24.04-2_amd64.tar.zst"
 proxmoxLxcDefaultStorage: local-lvm # 指定rootfs 位置
 pve-node-02:
 deployment_type: lxc
 # ... 其他配置

 local_users:
 admin_user:
 name: root
 public_key: "ssh-rsa AAAA..."

dns:
 hosts:
 site-dns01:
 ansible_host: 192.168.1.20
 vars:
 proxmox_interface: vmbr0
 gateway: 192.168.1.1
 mask_cidr: 24 # 子网掩码 24
 vlanid: 100 # VLAN ID
 proxmoxLxcCores: 2 # 核心数
 proxmoxLxcMemoryMb: 4096 # 内存大小
 proxmoxLxcDiskSizeGb: 30 # 磁盘大小
 proxmoxLxcRootFsStorageName: local-lvm # 指定rootfs 位置
 proxmoxLxcDefaultStorage
```

11

□□□□□□ **LXC**□

```
ansible-playbook -i hosts/Custommer/site.yml
util_playbooks/proxmox.yml
```

11

```
ansible-playbook -i hosts/Custommer/site.yml
util_playbooks/proxmox_delete.yml
```

XXXXXXXXXX LXC?? deployment\_type XX

|  |  |  |  |
|--|--|--|--|
|  |  |  |  |
|--|--|--|--|

|  |  |  |  |
|--|--|--|--|
|  |  |  |  |
|--|--|--|--|

- `deployment_type: lxc` → `lxc`
- `LXC` → `lxc`
- `LXC` → `UPF` → `UPF`

111

- `proxmoxServers` `proxmox_server`: `proxmoxServer`
- `Proxmox API` `VMID`
- `proxmoxTemplateId` `cloud-init` `SSH` `cloud-init` = `proxmoxTemplateUser` `local_users` `cloud-init` = `proxmoxTemplatePassword` `cloud-init` `scsi0`
- **LXC** `proxmoxLxc0sTemplate` `rootfs` `proxmoxLxcRootFsStorageName` `proxmoxLxcDefaultStorage` `storage` `local-lvm` `ssh-public-keys` `local users.*.public key`

0mn1Touch@ LXC unprivileged: 1 features:  
nesting=1

## 

LXC

- VLAN LXC IP
- /
- scsi0 / LXC rootfs
- secondary\_ips NIC

**OmniUPF** n3 data/gtp secondary\_ips NIC  
omniupf N3/N9 + XDP CPU  
PFCP/API/SSH IP

## 

Ansible + site\_name

## 

proxmoxServers LXC  
proxmoxServers

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

proxmoxServers  
-0 → -0-1 → -1...

[illegible]

```
proxmox_server: proxmoxServers
proxmoxServers
```

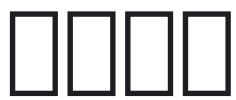
```
device_tester:
 hosts:
 site-device-tester01:
 ansible_host: 10.179.2.50
 proxmox_server: nicklab-prxmx01 # 指定 proxmoxServers 指定
 # 指定 proxmoxLxcRootFsStorageName 指定
 # 指定 proxmoxLxcRootFsStorageName
 proxmoxLxcRootFsStorageName: local-lvm
```

```
proxmox_server 000000 vars: 00000000000000000000000000000000
```

|  |  |  |  |  |
|--|--|--|--|--|
|  |  |  |  |  |
|--|--|--|--|--|

```
deployment_type proxmox_server 00000000 00000000 LXC 00
00 proxmoxLxcCores 00 proxmoxLxcMemoryMb 00 proxmoxLxcDiskSizeGb 00
proxmoxLxcRootFsStorageName 00 proxmoxLxcOsTemplate 00 host_vm_network 0000
000000000000
```

```
dns:
 hosts:
 site-dns01:
 ansible_host: 192.168.1.20
 deployment_type: vm # 虚拟机
 proxmoxLxcDiskSizeGb: 120 # 120GB
 host_vm_network: vmbr1 # 宿主机网络接口
vars:
 proxmox_interface: vmbr0
 gateway: 192.168.1.1
 mask_cidr: 24 # 子网掩码 24
```



OmniCore [services/](#)



```

all.yml
├─ setup_users.yml
├─ apt_cache.yml # apt_cache_servers []
├─ dns.yml
├─ common.yml
├─ license_server.yml
├─ monitoring.yml
│ └─ grafana.yml
├─ device_tester.yml
├─ epc.yml
│ ├── common.yml
│ ├── omnimme.yml
│ ├── omnisgwc.yml
│ ├── omnipgwc.yml
│ ├── upf.yml
│ ├── omnihss.yml
│ └─ omnidra.yml
├─ ims.yml
│ ├── pcscf.yml
│ ├── icscf.yml
│ ├── scscf.yml
│ ├── as.yml
│ ├── omnimessage.yml
│ └─ omniseq.yml
├─ omniepdg.yml
├─ omnitwag.yml
├─ omniss7.yml
├─ ocs.yml
├─ crm.yml
├─ ran_monitor.yml
├─ 5gc.yml
└─ ../util_playbooks/health_check.yml

```

〇〇〇〇〇〇〇〇〇〇 〇〇〇〇〇〇〇〇〇 omnimme.yml 〇〇〇〇〇〇〇〇 50 〇〇〇〇 all.yml 〇〇〇〇〇〇〇〇〇〇〇〇  
 〇〇〇〇〇〇〇〇〇 --limit 〇〇〇〇〇〇〇〇〇

目录

目录

| 文件             | 目录     | 内容                                                                    |
|----------------|--------|-----------------------------------------------------------------------|
| all.yml        | 目录     | 包含所有组件的DNS配置、EPC、IMS、ePDG、TWAG、SS7、OCS、CRM、RAN、5G、以及util_playbooks/目录 |
| epc.yml        | 目录     | MME、SGW-C、PGW-C、UPF、HSS、DRA                                           |
| ims.yml        | 目录/IMS | P/I/S-CSCF、OmniSEP、XCAP、BSF                                           |
| ocs.yml        | 目录     | CGrateS、KeyDB、OCS                                                     |
| monitoring.yml | 目录     | Prometheus、Grafana、HOMER                                              |

環境構築

| ファイル                   | 内容                          |
|------------------------|-----------------------------|
| common.yml             | 基本的な設定、NTPサーバー              |
| setup_users.yml        | SSH 接続のユーザー設定               |
| dns.yml                | DNS サーバー設定                  |
| license_server.yml     | OmniCore のライセンスサーバー         |
| does_netplan_exist.yml | netplan がインストールされているかどうかを確認 |
| update_mtu.yml         | netplan の MTU を更新           |
| firewall.yml           | iptables/nftables の設定       |
| apt_cache.yml          | APT のキャッシュを設定               |

# EPC 组件

| 配置文件                                            | 组件名称      | 功能描述                    |
|-------------------------------------------------|-----------|-------------------------|
| <code>omnimme.yml</code>                        | OmniMME   | 4G 核心网 (4G)             |
| <code>omnisgwc.yml</code>                       | OmniSGW-C | 4G 核心网 (4G)             |
| <code>omnipgwc.yml</code>                       | OmniPGW-C | PDN 连接管理                |
| <code>upf.yml</code> / <code>omniupf.yml</code> | OmniUPF   | 4G 核心网 (4G) SGW-U/PGW-U |
| <code>omnihss.yml</code> / <code>hss.yml</code> | OmniHSS   | 4G 核心网 (4G)             |
| <code>omnidra.yml</code>                        | OmniDRA   | Diameter 接口             |
| <code>omnitwag.yml</code>                       | OmniTWAG  | 4G 核心网 (4G)             |
| <code>omniepdg.yml</code>                       | OmniEPDG  | 4G 核心网 (4G) WiFi 接入     |
| <code>gtp_proxy.yml</code>                      | GTP 代理    | GTP 代理                  |

# IMS

| ファイル名           | 機能                        | 説明                                                                                    |
|-----------------|---------------------------|---------------------------------------------------------------------------------------|
| pcscf.yml       | P-CSCF                    | パブリックインターネット接続                                                                        |
| icscf.yml       | I-CSCF                    | インターネット接続                                                                             |
| scscf.yml       | S-CSCF                    | インターネット接続                                                                             |
| as.yml          | OmniTAS                   | VoLTEサービス                                                                             |
| omnisep.yml     | OmniSEP                   | XCAPサービスとBSFサービス                                                                      |
| xcap.yml        | OmniSEP                   | omnisep.yml サービス<br>omnisep サービス                                                      |
| omnimessage.yml | OmniMessage /<br>IMS SMSC | SMS-over-IP サービス<br>SMPP サービス<br>IMS<br>SMSCサービス cscf サービス cscf_type:<br>smc_ims サービス |

# 5G

5gc.yml サービス 5gc\_nf サービス NF サービス nf\_package サービス  
NF サービス UDM HNET/SUCI サービス 5G Core

| 項目      | 項目       | 項目             |
|---------|----------|----------------|
| 5gc.yml | 5G 項目    | 項目項目 5G 項目項目項目 |
|         | OmniNRF  | 項目項目項目         |
|         | OmniAMF  | 項目項目項目項目       |
|         | OmniAUSF | 項目項目項目         |
|         | OmniBSF  | 項目項目項目         |
|         | OmniCHF  | 項目項目           |
|         | OmniNSSF | 項目項目項目項目       |
|         | OmniPCF  | 項目項目項目         |
|         | OmniSCP  | 項目項目項目         |
|         | OmniSMF  | 項目項目項目         |
|         | OmniUDM  | 項目項目項目         |
|         | OmniUDR  | 項目項目項目         |

項目項目

項目 cs.yml 項目項目項目 circuit\_switched/ 項目項目 BSC項目MSC 項目 SS7 項目項目項目項目項目項目  
Circuit-Switched Core項目

| 名前                  | 説明      | ディレクトリ                                |
|---------------------|---------|---------------------------------------|
| <code>cs.yml</code> | CS      | <code>CS</code>                       |
|                     | BSC     | <code>circuit_switched/bsc</code>     |
|                     | OmniMSC | <code>circuit_switched/omnimsc</code> |
|                     | SS7     | <code>SS7/SIGTRAN omniss7</code>      |

## その他

| 名前                             | 説明                               |
|--------------------------------|----------------------------------|
| <code>ocs.yml</code>           | CGateS + KeyDB                   |
| <code>crm.yml</code>           | OmniCRM <code>group_vars/</code> |
| <code>omniss7.yml</code>       | SS7/SIGTRAN                      |
| <code>homer.yml</code>         | SIP/Diameter                     |
| <code>grafana.yml</code>       | Grafana                          |
| <code>ran_monitor.yml</code>   | RAN                              |
| <code>omniroam.yml</code>      | OmniRoam TAP3/RAP UI Elixir API  |
| <code>omnilcs.yml</code>       | OmniLCS                          |
| <code>device_tester.yml</code> | OmniWeb                          |

**Proxmox** `proxmox.yml` `proxmox_delete.yml` `util_playbooks/`   
 [Utility Playbooks](#) [Proxmox VM/LXC](#)

00

| 00                       | 00            |
|--------------------------|---------------|
| backup.yml               | 0000000000    |
| reboot.yml               | 00000000      |
| shutdown.yml             | 00000         |
| apt_update.yml           | 000000        |
| apt_refresh_metadata.yml | 00 APT 000000 |
| speedtest.yml            | 0000000000    |

00000

| 00                            | 00                                                      |
|-------------------------------|---------------------------------------------------------|
| restore_applicationserver.yml | 000000 OmniTAS                                          |
| restore_smsc.yml              | 000000 SMSCOmniMessage 000000SMPP<br>0000 IMS SMS Sc 00 |

```
000health_check.yml00restore_hss.yml 00restore_ocs.yml 00
services/ 00000000 util_playbooks/ 000000 Utility Playbooks00all.yml 00
../util_playbooks/health_check.yml 0000000000
```

□□

□□□□□□

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

□□□□□□□

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

□ IMS/□□
ansible-playbook -i hosts/customer/host_files/production.yml
services/ims.yml
```

□□□□□□

```
□□□ MME
ansible-playbook -i hosts/customer/host_files/production.yml
services/omnimme.yml

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

## 📁📁📁📁📁

```
📁📁📁📁📁📁 all.yml
ansible-playbook -i hosts/customer/host_files/production.yml
services/all.yml --limit mme01

📁📁📁📁📁📁
ansible-playbook -i hosts/customer/host_files/production.yml
services/all.yml --limit "mme01,hss01"

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

## 📁📁📁📁

- 📁📁📁 - 📁📁📁📁📁📁
- 📁📁📁📁 - 📁📁📁📁
- 📁📁📁 - 📁📁
- 📁📁📁📁📁 - 📁📁📁📁📁📁📁📁📁
- 📁📁📁📁 - Grafana📁Prometheus📁📁



OmniCore `util_playbooks/`

## 目次

| ファイル名                    | 説明                                           |
|--------------------------|----------------------------------------------|
| proxmox.yml              | Proxmox クラウド/ LXC のインストール<br>deployment_type |
| proxmox_delete.yml       | Proxmox クラウド/LXC の削除                         |
| proxmox_restart_hung.yml | SSH を介して Proxmox API を再起動/LXC                |
| vmware.yml               | VMware vSphere のインストール                       |
| vmware_delete.yml        | VMware vSphere の削除                           |
| health_check.yml         | ヘルスチェック                                      |
| restore_hss.yml          | HSS のバックアップ/リストア                             |
| seed_hss_replication.yml | OmniHSS の複製のシード                              |
| audit_hss_drift.yml      | OmniHSS のドリフトの監査 + レポート                      |
| reconcile_hss_drift.yml  | OmniHSS のドリフトの修正<br>???                      |
| restore_ocs.yml          | ocs KeyDB/MySQL のバックアップ/リストア                 |
| ip_plan_generator.yml    | Mermaid の IP プランジェネレーター                      |
| get_ports.yml            | ポートの取得                                       |
| getLocalCapture.yml      | ローカルキャプチャ                                    |
| delete_local_user.yml    | ローカルユーザーの削除                                  |

| 名前                              | 説明                                      |
|---------------------------------|-----------------------------------------|
| <code>clear_mnesia.yml</code>   | OmniSS7 / OmniMessage の Mnesia データを削除する |
| <code>delete_license.yml</code> | ライセンスを削除する                              |

## インストール

インストール時に、`--limit` オプションで制限を指定する

インストール時に **Proxmox** を指定する

## Proxmox

例: `util_playbooks/proxmox.yml`, `util_playbooks/proxmox_delete.yml`

`proxmox.yml` は、LXC コンテナのインストールと LXC コンテナの `deployment_type` を指定する。→ `proxmoxServers` に指定する → `vm` を指定する **Proxmox** を指定する

```
LXC コンテナのインストール
ansible-playbook -i hosts/customer/host_files/production.yml
util_playbooks/proxmox.yml

LXC コンテナのインストール
ansible-playbook -i hosts/customer/host_files/production.yml
util_playbooks/proxmox.yml --limit mme

LXC コンテナのインストール
ansible-playbook -i hosts/customer/host_files/production.yml
util_playbooks/proxmox_delete.yml --limit old-host
```

## Proxmox の再起動

例: `util_playbooks/proxmox_restart_hung.yml`

利用 SSH 利用 Proxmox API 利用 LXC 利用 QEMU 利用 LXC 利用 SSH

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

```

ansible-playbook -i hosts/customer/host_files/production.yml
util_playbooks/proxmox_restart_hung.yml --limit hss01
```

## VMware vSphere

利用: util\_playbooks/vmware.yml, util\_playbooks/vmware\_delete.yml

利用 util\_playbooks/vm/vmware/ 利用  
create\_individual\_vm.yml delete\_individual\_vm.yml  
vmware\_delete.yml vmware.yml SSH netplan  
netplan

利用: Python uv sync

利用:

```

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

```

ansible-playbook -i hosts/customer/host_files/production.yml
util_playbooks/vmware.yml --limit mme
```

```

ansible-playbook -i hosts/customer/host_files/production.yml
util_playbooks/vmware_delete.yml --limit old-host
```

利用:

```
all:
 vars:
 vcenter_ip: "vcenter.example.com"
 vcenter_username: "administrator@vsphere.local"
 vcenter_password: "password"
 vcenter_datacenter: "Datacenter"
 vcenter_folder: "OmniCore"
 vcenter_vm_template: "ubuntu-2404-template"
 vhosts:
 esxi-01:
 vcenter_cluster_ip: "192.168.1.10"
 vcenter_datastore: "datastore1"
```

## Vultr

```
##: util_playbooks/vm/vultr/vmProvisionVultr.yml,
util_playbooks/vm/vultr/vmTeardownVultr.yml,
util_playbooks/vm/vultr/vmCommonSetup.yml
```

## Vultr API ## Vultr ## ## vmProvisionVultr.yml ## ##  
vmTeardownVultr.yml ## ## vmCommonSetup.yml ## ## ## DNS ## ## ##  
## ## ## ## VULTR\_API\_TOKEN ## ## ## API ## ## ## ## cloud-  
config.yaml ## ## util\_playbooks/vm/vultr/ ## ##

```
export VULTR_API_TOKEN=<token>
ansible-playbook -i hosts/customer/host_files/production.yml
util_playbooks/vm/vultr/vmProvisionVultr.yml
```

## ## ##

```
##: util_playbooks/health_check.yml
```

## ## ## OmniCore ## OmniCall ## ## HTML ## ##

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

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

Table

| Item    | Description      |
|---------|------------------|
| Item    | Description      |
| OmniHSS | OmniHSS Diameter |
| OmniDRA | Diameter         |
| OmniTAS | OmniTAS CPU      |
| OCS     | KeyDB            |

HSS

```
curl: util_playbooks/restore_hss.yml
```

OmniHSS

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

環境構築

| 項目                                        | ファイル名                                                        | 説明                                                             |
|-------------------------------------------|--------------------------------------------------------------|----------------------------------------------------------------|
| OmniHSS から PostgreSQL (pg_dump) をバックアップする | <code>hss_dump_&lt;hostname&gt;_&lt;timestamp&gt;.sql</code> | OmniHSS から PostgreSQL (pg_dump) をバックアップする <code>omnihss</code> |
| OmniHSS から PostgreSQL (pg_dump) をバックアップする | <code>hss_&lt;hostname&gt;_&lt;timestamp&gt;.tar.gz</code>   | <code>/etc/omnihss</code> にバックアップ                              |

環境構築完了後 HSS を再起動/インストール

確認

| 項目                 | 確認                       | 説明                     |
|--------------------|--------------------------|------------------------|
| HSS SQL をバックアップ    | <input type="checkbox"/> | HSS SQL をバックアップする      |
| HSS tar.gz をバックアップ | <input type="checkbox"/> | HSS tar.gz をバックアップする   |
| 確認                 | <input type="checkbox"/> | <code>yes</code> を入力する |

HSS のインストール

コマンド: `util_playbooks/seed_hss_replication.yml`

インストール時に OmniHSS をインストールする場合は `copy_data = false` を指定する。  
インストール時に PostgreSQL をインストールする場合は `copy_data = true` を指定する。

インストール完了後

確認コマンド: `seed_source_*` を実行する。  
インストール完了後

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

```

1) dev-hss01 (10.4.10.142) - 838

```

□□: □□□□□□□□ □□ □□□□□□□□□□□□□□□□

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


 Türkiye Cumhuriyeti Millî Eğitim Bakanlığı  
 Ministry of National Education of the Republic of Turkey

| 項目               | 単位  | 説明                                             | デフォルト値                                |
|------------------|-----|------------------------------------------------|---------------------------------------|
| seed_source_name | 文字列 | データベース名                                        | sub_from_<name><br><name>_pub         |
| seed_source_host | 文字列 | データベースホスト名                                     | IP アドレス<br>PostgreSQL データベース<br>のホスト名 |
| seed_source_port | 整数  | データベースポート番号<br>omnihss.database_port<br>(5432) | PostgreSQL の<br>ポート番号                 |

```
seed_source_name | seed_source_host |
```

1. `CREATE TABLE subscriber_state (`
2. `hss VARCHAR(255) NOT NULL,`  
`connected_omnihss VARCHAR(255) NOT NULL,`  
`yes VARCHAR(255) NOT NULL,`
3. `IP VARCHAR(255) NOT NULL,`  
`pg_hba.conf`
4. `WITH (copy_data = false)`  
`slot_name`  
`<target>_from_<source>`
5. `pg_dump --data-only --inserts --on-conflict-`  
`do-nothing --disable-triggers`  
`subscriber_state`  
`FK`  
`pdn_session`  
`lte_call`
6. `CREATE TABLE pdn_session (`

```
00: 00000000 --limit 00000000000000000000000000000000 OmniHSS 0000
0000
```

```

[]: util_playbooks/audit_hss_drift.yml [] [] [] []
roles/omnihss/tasks/audit_drift.yml [] [] [] [] []

```

```

hss + connected_omnihss
md5(schema_migrations +
subscriber_state pdn_session lte_call
__orphaned_subscribers__ subscriber_state
subscriber
subscriber_state
Postgres --limit

```

```

$ cat /etc/passwd | grep hss | grep omnihss | grep services/all.yml | grep free | grep services/deploy_nfs.yml

```

| 項目                          | 設定   | 説明                                   |
|-----------------------------|------|--------------------------------------|
| omnihss_audit_drift         | true | 監視対象のドリフトを監視する                       |
| omnihss_audit_fail_on_drift | true | 監視対象のドリフトが発生した場合、監視失敗とみなす<br>cron/CI |

cron/CI

```
[]
ansible-playbook -i hosts/<inv>/host_files/<site>.yaml
util_playbooks/audit_hss_drift.yaml
[]
ansible-playbook -i ... util_playbooks/audit_hss_drift.yaml -e
audit_fail_on_drift=false
[]
ansible-playbook -i ... services/omnihss.yaml -e
omnihss audit drift=false
```

HSS ☐ ☐ ☐ ☐

```
❯❯: util playbooks/reconcile hss drift.yml
```

```
pg_dump --data-only --inserts --on-conflict-do-nothing --disable-triggers
```

- `INSERT` 语句使用 `"{}"` 来指定要插入的列名
- 使用 `yes` 选项来指定是否要插入数据 `-e dry_run=true` 选项来指定是否要插入数据 `-e assume_yes=true` 选项来指定是否要插入数据

- postgres 設定で `--disable-triggers` を指定する
- 設定ファイルのパス / 名前が不明な場合は、`ansible-playbook --help` を実行して確認する
- 設定ファイルのパス / 名前が不明な場合は、`ansible-playbook --help` を実行して確認する

```
実行コマンド
ansible-playbook -i hosts/<inv>/host_files/<site>.yaml
util_playbooks/reconcile_hss_drift.yaml
実行コマンド
ansible-playbook -i ... util_playbooks/reconcile_hss_drift.yaml -e
dry_run=true
実行コマンド
ansible-playbook -i ... util_playbooks/reconcile_hss_drift.yaml -e
assume_yes=true
```

実行コマンド `audit_hss_drift.yaml` を実行して、HSS の状態を確認する。

実行コマンド `OmniHSS` を実行する。

## OCS 実行

実行コマンド: `util_playbooks/restore_ocs.yaml`

実行コマンド OCS (CGrates) を実行して、KeyDB の状態を確認する。

```
ansible-playbook -i hosts/customer/host_files/production.yaml
util_playbooks/restore_ocs.yaml
```

## 実行手順

1. OCS を実行して CGrates の KeyDB
2. AOF を実行して KeyDB
3. KeyDB RDB を実行して KeyDB
4. MySQL StoreDB を実行して
5. `/etc/cgrates` を実行して

## 6. CGrateS 環境構築

### 環境構築

| 項目                | ファイル名                                                            | 説明                                |
|-------------------|------------------------------------------------------------------|-----------------------------------|
| KeyDB データ<br>ファイル | <code>keydb_dump_&lt;hostname&gt;_&lt;timestamp&gt;.rdb</code>   | KeyDB RDB ファイル                    |
| MySQL<br>StoreDB  | <code>cgrates_dump_&lt;hostname&gt;_&lt;timestamp&gt;.sql</code> | <code>cgrates</code> の MySQL データ  |
| バックアップ<br>ファイル    | <code>cgrates_&lt;hostname&gt;_&lt;timestamp&gt;.tar.gz</code>   | <code>/etc/cgrates</code> のバックアップ |

### 手順

| 項目                    | 操作 | 説明                                     |
|-----------------------|----|----------------------------------------|
| KeyDB RDB ファイル        | 取得 | KeyDB RDB データをバックアップ                   |
| MySQL SQL ファイル        | 取得 | CGrateS SQL データを MySQL StoreDB にバックアップ |
| バックアップ<br>ファイル tar.gz | 作成 | バックアップファイルを作成                          |

## IP 計画

例: `util_playbooks/ip_plan_generator.yml`

### 環境構築

- IP 計画 NIC 計画 NIC
- 環境構築
- 環境構築DiameterGTPPFPCPSIPSS7

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

???

| File                                | Format   | Output |
|-------------------------------------|----------|--------|
| /tmp/ip_plan_<customer>_<site>.md   | Markdown |        |
| /tmp/ip_plan_<customer>_<site>.html | HTML     |        |

Step 1

```
util_playbooks/get_ports.yml
```

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

Step 2

| File               | Format                       |
|--------------------|------------------------------|
| /tmp/all_ports.csv | IP addresses CSV file        |
| ./open_ports.rst   | Sphinx reStructuredText file |

필수사항

| 항목      | 설명                 |
|---------|--------------------|
| 필수      | 필수사항               |
| IP      | ansible_host IP 지정 |
| IP 프로토콜 | IPv4 또는 IPv6       |
| 포트      | TCP 또는 UDP         |
| 패킷      | 패킷                 |
| 파일      | 파일                 |

실행방법

1. util\_playbooks/getLocalCapture.yml

2. /etc/localcapture 디렉토리 생성

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

3. ./localCapturePcaps/<hostname>/\*.pcap

결과

1. util\_playbooks/delete\_local\_user.yml

2. 결과

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

00: 0000000000000000

## 00 Mnesia 000

00: util\_playbooks/clear\_mnesia.yml

00000000 Mnesia 000000000000 ss7 00OmniSS700 omnimessage 0  
0OmniMessage0000

00: 00000000Mnesia 0000000000000000000000000000

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

```
000000000000
ansible-playbook -i hosts/customer/host_files/production.yml
util_playbooks/clear_mnesia.yml --limit ss7
```

## 000000

00: util\_playbooks/delete\_license.yml

00 license\_server 000 /etc/license\_server/license.json 000000000000

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

# Ansible

## 実行

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

## オプション

| オプション                              | 説明                |
|------------------------------------|-------------------|
| <code>-i &lt;inventory&gt;</code>  | ホストのリスト           |
| <code>--limit &lt;hosts&gt;</code> | 実行するホストの制限        |
| <code>-v / -vv / -vvv</code>       | verbosity (デバッグ)  |
| <code>--check</code>               | 実行せずに確認 (dry-run) |
| <code>--diff</code>                | 変更内容の差分表示         |

## 例

```
健康チェックを実行
ansible-playbook -i hosts/acme/host_files/production.yaml
util_playbooks/health_check.yaml

HSS を復元
ansible-playbook -i hosts/acme/host_files/production.yaml
util_playbooks/restore_hss.yaml --limit hss01

IP 計画を生成
ansible-playbook -i hosts/acme/host_files/production.yaml
util_playbooks/ip_plan_generator.yaml -v
```

