

5G 5GC

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

11

OmniCore5gc_nf5GC NFNFNF
nf_packageDebiansystemdJinja2
NF

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

--	--	--	--

```

NFsystemd
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- 4
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コア機能

コアNF機能

1. `group_vars_omnicore` コア `<inventory>/group_vars/` コア機能
2. コア機能 `license_server_api_urls` コア `license_server` コア
`https://<host>:8443/api`
3. コア機能 `lsb-release` `curl` `gpg` `unzip` `libglu1-mesa` `libsctp1`
4. **OmniTouch APT** コア **NF** コア機能 `nf_version` コア機能

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

/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 GW)
- DNS Core/Edge DNS - IMS Core/Edge DNS
- TAP3/CDR
- OAM/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 (MME)
VLAN 402: 10.x.3.0/24 (IMS S-CSCF)
VLAN 403: 10.x.4.0/24 (IMS SBC)

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:
    # [] SGW []
    customer-site-sgw01:
      ansible_host: 10.4.1.25
      gateway: 10.4.1.1
      host_vm_network: "v400-omni-packet-core"

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

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

```

[] IP [] DRA

```

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

```

[] IP [] 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 可以实现网络设备的自动化配置管理，提高网络运维效率。4G/5G 网络设备的配置管理，Ansible 可以实现自动化配置。

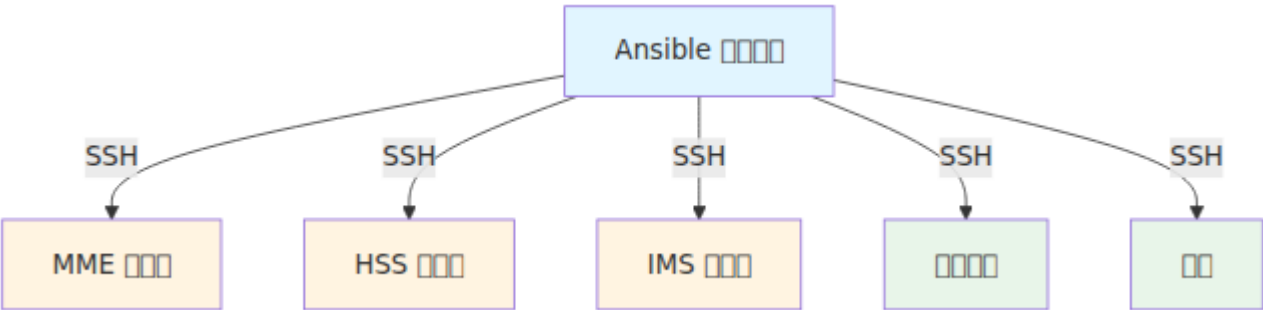
结合 Ansible

Ansible 可以实现网络设备的自动化配置管理。

- 配置管理
- 网络配置
- 网络设备的配置管理
- 网络设备的配置管理

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

Omnitouch 结合 Ansible



结合

1. 结合

이 단락을 완료한 후 다음 단락을 진행합니다.

- 네트워크 구성
- IP 주소 할당
- 서비스 설치
- 테스트 실행

이 단락을 완료한 후 다음 단락을 진행합니다.

이 단락을 완료한 후 다음 단락을 진행합니다.

2. 준비

이 단락을 완료한 후 다음 단락을 진행합니다.

- 네트워크 구성
- IP 주소 할당
- 서비스 설치
- 테스트 실행

OmniCore 구성을 완료한 후 `omnihss`, `omnisgw`, `omnipgw`, `omnidra` 구성을 완료합니다.

ONS 구성을 완료한 후 다음 단락을 진행합니다.

3. 설치

이 단락을 완료한 후 다음 단락을 진행합니다.

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

이 단락을 완료한 후 다음 단락을 진행합니다.

이 단락을 완료한 후 다음 단락을 진행합니다. `services/epc.yml` 파일을 `import_playbook`로 `common.yml` 파일을 호출합니다.

```

- name: 設定ファイル
  import_playbook: common.yml

- name: 全 MME 実行
  import_playbook: omnimme.yml

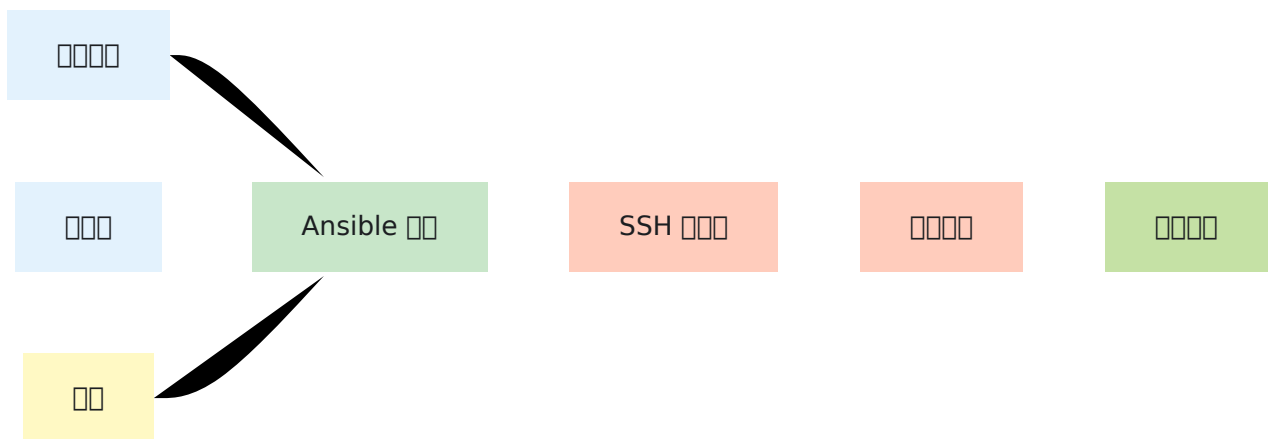
```

4. 実行

実行コマンドは以下の通りです。

実行コマンド

実行コマンド



実行コマンド

1. 環境構築

環境構築の手順は以下の通りです。

まず、Ansible のインストールと IP の設定を行います。

Proxmox 上で Ansible を実行する場合は Proxmox の VM/LXC 上で Ansible を実行する必要があります。

実行コマンドは以下の通りです。

```
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. 確認

Ansible 確認

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

OmniCall 組件

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

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

OmniCharge/OmniCRM

- **CRM** - CRM 組件

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

監控

- **DNS** - DNS 監控
- **網絡** - 網絡監控
- **監控** - 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をインストールする必要があります。

1. uv のインストール

uv をインストールするには、以下のコマンドを実行します。

```
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

仮想環境を作成します。

```
uv sync
```

作成された仮想環境 `.venv/` に Ansible と Omnitouch の依存関係を Python の `uv.lock` ファイルでロックします。

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、仮想環境をアクティベートします。

```
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` ファイルを使用します。

```
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. 如何 安装 系统
2. 如何 安装 系统
3. 如何 **APT** 安装 系统
4. 如何 安装 系统
5. 如何

如何

- **IP** 地址 - 如何安装 **IP** 地址
- 如何 - 如何安装
- **APT** 安装 - 如何安装
- 如何 - 如何安装
- 如何 - 如何安装
- 如何 - 如何安装
- 如何 - 如何安装

APT 環境構築

概要

Omnitouch APT 環境構築手順

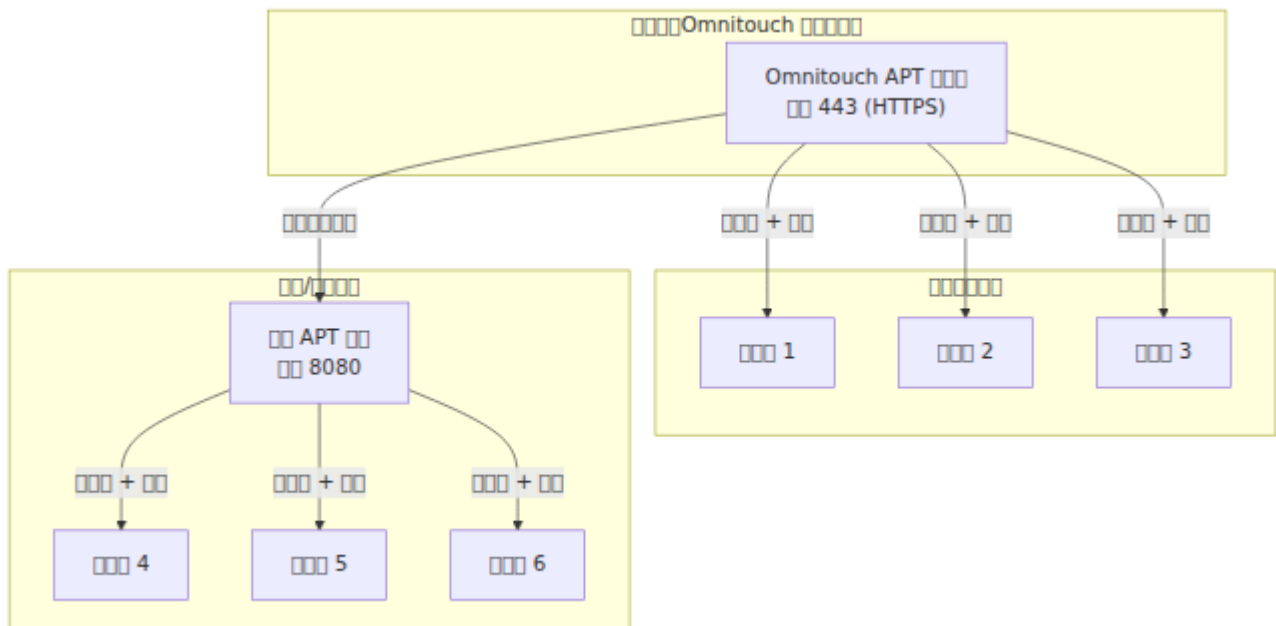
1. **APT** 環境構築 `apt install` 環境 Debian 環境
2. 環境構築環境構築環境構築Prometheus 環境構築

APT 環境構築環境構築 環境構築**OmniCore** 環境構築 `omnicore` 環境構築
`/pool/main/o/omnicore/` 環境構築 Ansible 環境構築環境構築環境構築

環境構築

1. 環境構築環境構築 Omnitouch APT 環境構築環境構築 `apt.<region>.omnitou.ch`
2. 環境構築環境構築 Omnitouch 環境構築環境構築環境構築/環境構築

環境構築



APT

Omnitouch 	.deb omnihss omnimme 	/dists/<distro>/
OmniCore 	omnicore Ansible 	/pool/main/o/omnicore/
Ubuntu 	Ubuntu 	/<distro>/pool/main/
GitHub 	Prometheus Grafana VoIPmonitor 	/releases/<org>/<repo>/
	Web CGrateS_UI speedtest 	/repos/
	Galera FRR InfluxDB KeyDB 	/releases/<vendor>/

apt_repo
URL

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/...

1

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

項目名 **Omnitouch** 項目名

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	123	UDP	項目項目 NTP 項目
項目項目	53	TCP/UDP	項目項目 DNS 項目

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

00

```
all:
  vars:
    use_apt_cache: false

    # 0000000000000000remote_apt_*00
    # apt_download_base URL 000000 - 0000
    # 00000000
    apt_repo:
      apt_server: "apt.<region>.omnitou.ch"      # Omnitouch 000 FQDN
      apt_repo_username: "your-username"
      apt_repo_password: "your-password"
      # apt_repo_port: 443                      # 000000 443
    (HTTPS)
```

00

APT 0000 (apt_repo)

00	00	0	00	00
apt_repo.apt_server	00	0	-	APT 00000000 IP 00
apt_repo.apt_repo_username	00	0	-	HTTP 0000000000
apt_repo.apt_repo_password	00	0	-	HTTP 0000000000
apt_repo.apt_repo_port	00	0	443	00000000000000 443 → HTTPS000000 HTTP00

00000 (remote_apt_*)

変数 `apt_repo` は、以下の値を持つ変数として定義されています。2つの値は、デフォルト値です。

変数	変数名	値
<code>remote_apt_server</code>	<code>apt_repo.apt_server</code>	<code>apt.us-fl.omnitou.ch</code>
<code>remote_apt_port</code>	<code>apt_repo.apt_repo_port</code>	<code>443</code>
<code>remote_apt_protocol</code>	変数 <code>443</code> → <code>https</code>	<code>https</code>
<code>remote_apt_user</code>	<code>apt_repo.apt_repo_username</code>	<code>" "</code>
<code>remote_apt_password</code>	<code>apt_repo.apt_repo_password</code>	<code>" "</code>

変数

変数	変数	変数	変数	変数
<code>use_apt_cache</code>	変数	変数	-	変数 <code>false</code>

URL 変数

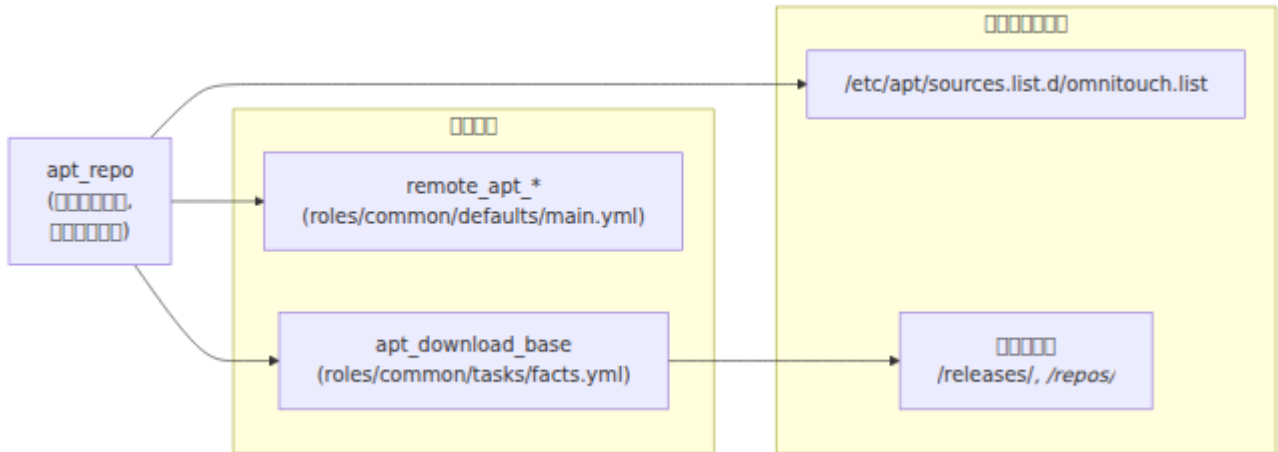
APT 変数 `/etc/apt/sources.list.d/omnitouch.list`

```
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
```

URL 443 HTTPS :<port> HTTP 443



HTTP APT Ubuntu Omnitouch Ubuntu

GPG

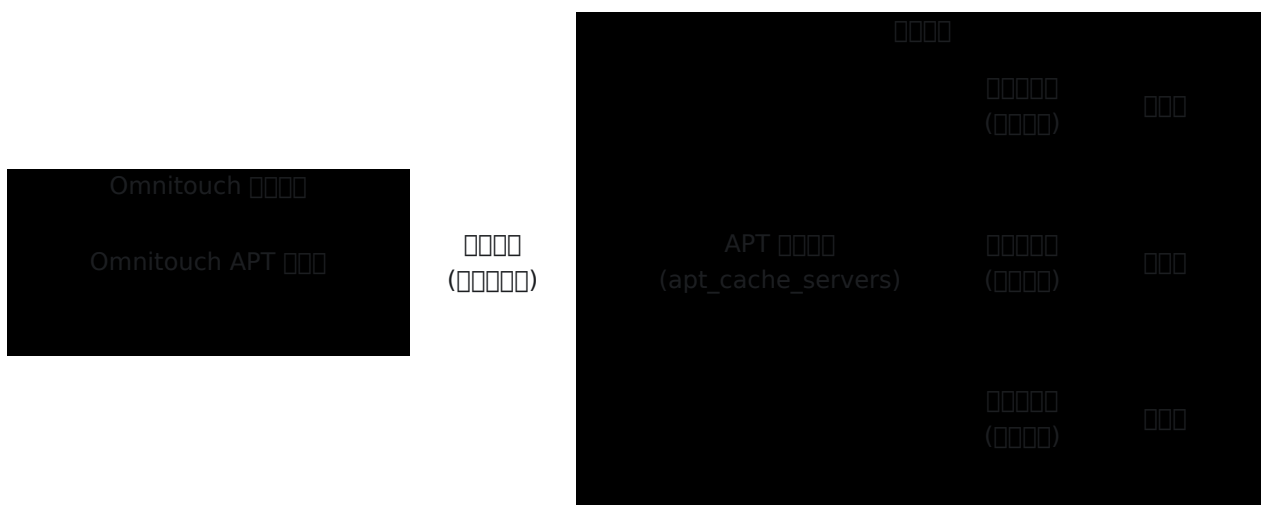
Omnitouch APT /omnitouch.gpg common /etc/apt/trusted.gpg.d/omnitouch.gpg apt-get update NO_PUBKEY

GPG [trusted=yes] 2

2

APT Omnitouch

❏❏



❏❏

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_apt_* HTTPS Omnitouch
- apt_repo.apt_server: "192.168.1.100" 8080

APT (apt_repo)

apt_repo.apt_server				IP apt_cache_servers
apt_repo.apt_repo_username			-	
apt_repo.apt_repo_password			-	
apt_repo.apt_repo_port			8080	apt_cache_port

(remote_apt_*)

Omnitouch remote_apt_*

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
use_apt_cache	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

前提条件

1. 50+ GB の LXC を作成
2. 環境構築

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

3. `http://192.168.1.100:8080/` にアクセス

環境構築

OmniTouch APT をインストールするための環境構築

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

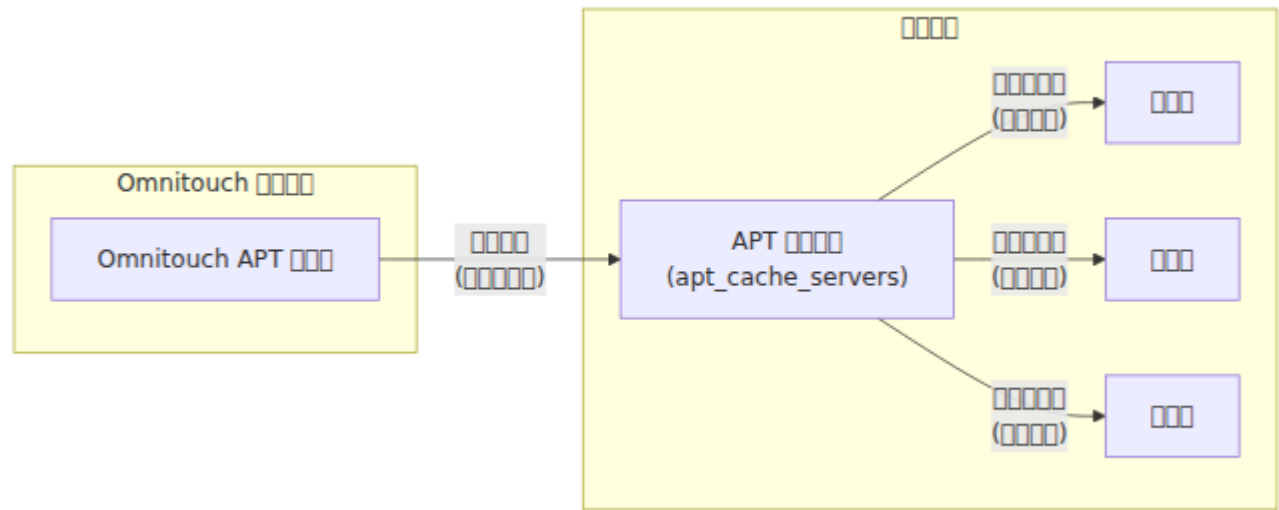
OmniTouch APT				
OmniTouch .deb	Ubuntu +	GitHub		APT
/pool/main/	/noble/pool/main/	/releases/	/repos/	/dists/
curl + index/SHA256	curl + index/SHA256	wget --recursive --timestamping	wget --recursive --timestamping	curl + index/SHA256
OmniTouch .deb	Ubuntu +	GitHub		APT

環境構築

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 ファイル
<code>/<distro>/pool/main/</code>	Ubuntu のディレクトリ
<code>/releases/</code>	GitHub の Prometheus, Grafana, Zabbix など
<code>/repos/</code>	Erlang, Elixir, CGrateS_UI など

ディレクトリ構造は以下の通りです。

ディレクトリ



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

ディレクトリ

ディレクトリ `apt_cache_servers` のディレクトリ

1. 設定 `use_apt_cache: true`
2. 設定 `ansible_host` IP として `apt_repo.apt_server`

設定

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

設定

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

設定

設定

```
all:
  vars:
    use_apt_cache: false # 設定
    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*_ 0000000

```
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 200000000000 **APT** 00000000000

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

```

実行

- 実行コマンド `https://user:pass@apt.<region>.omnitou.ch/` 実行
- 実行コマンド `deb [trusted=yes] http://192.168.1.100:8080/noble noble main` 実行

第3回 APT のインストールと設定

APT のインストールと設定 Ansible を使う

```

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 []
- [] tarball

OmniTouch APT 使用 SHA256 验证 Debian 仓库的 Release 文件
通过 `/releases/` 或 `/repos/` 使用 `wget --timestamping` 验证文件

OmniTouch APT 配置文件 `services/apt_cache.yml` 示例

配置

APT 仓库 401 错误

问题

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

原因

- `apt_repo` 配置 `all: vars:` 未正确设置 `apt_cache_servers: vars:`
- 配置文件中缺少必要的变量
- `apt_repo_username` 或 `apt_repo_password` 未正确设置
- IP 地址或 OmniTouch APT 服务未正确配置
- 配置文件中缺少必要的变量

解决方案

- 配置 `apt_repo` 和 `apt_cache_servers: vars:` 为 `all: vars:`
- 配置文件中添加必要的变量 `8080`
- 配置文件中添加必要的变量 `/etc/apt/sources.list.d/omnitouch.list`
 - 配置文件中添加必要的变量 `deb [trusted=yes] http://10.179.1.114:8080/noble noble main`
 - 配置文件中添加必要的变量 `deb [trusted=yes] https://user:pass@10.179.1.115/noble noble main`
- 配置文件中添加必要的变量

5. 設定する IP 上の Omnitouch 環境を構築する

Node ExporterとZabbix 設定

Ansible playbook を `/releases/` に配置する

実行する

- `apt_repo` をインストールする `apt_repo` をインストール
- ??? `remote_apt_user` / `remote_apt_password` 設定
- ポート `443` を HTTPS から HTTP に

実行する

- 設定 `apt_repo.apt_repo_username` / `apt_repo.apt_repo_password` を Omnitouch 設定
- 設定 `remote_apt_*` を設定
- 実行 `apt_download_base` を `ansible-playbook ... -vvv` で実行 `get_url` を実行

Node Exporter 設定

Ansible playbook を実行する

実行する

- Node Exporter をインストール
- `remote_apt_*` を設定
- 設定 Omnitouch を実行

実行する

- ポート `443` を Omnitouch APT に
 - 設定 `remote_apt_*` を設定
 - Node Exporter をインストール
-

□□□□

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

목표

요구사항

OmniCore 모니터링 통합 Grafana 모니터링

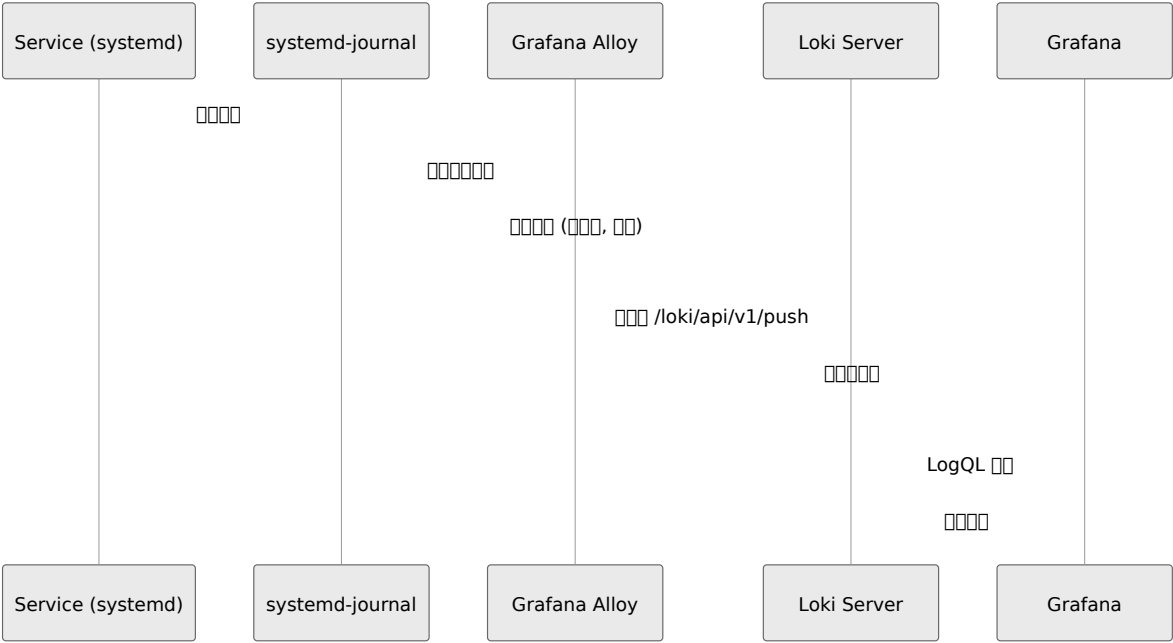
- **Grafana Alloy** 모니터링 통합
- **Grafana Loki** 로그 통합
- **Grafana** 모니터링 통합

구성



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	ログのジョブ名	customer-mme01
hostname	ホスト名 (job 名から取得)	customer-mme01
component	コンポーネント名	mme, pcscf, ocs
unit	systemd ユニット名	omnimme.service
level	ログレベル	info, warning, error
service	Syslog サービス名	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 Alloy 2.9.10 以上がインストールされていること syslog 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 Alloy 2.9.10 以上がインストールされていること Loki 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 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

```
all:
  vars:
    loki_retention_period: "168h" # 7 日間保存
```

Alloy ☐☐☐☐

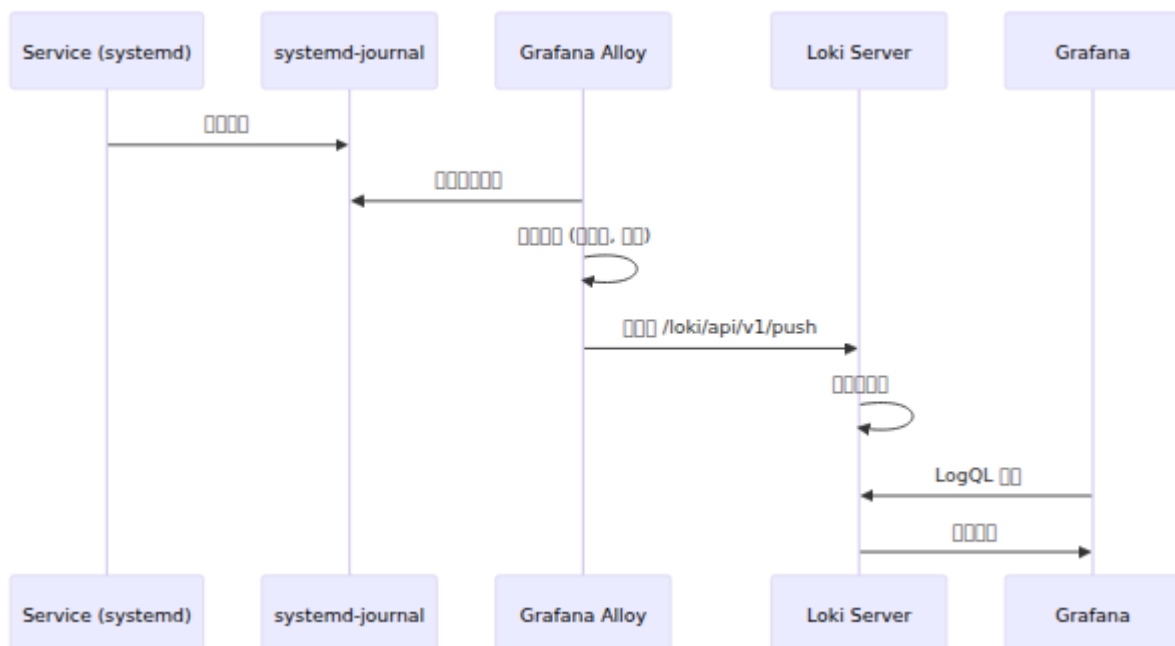
☐ Loki ☒ Alloy WAL ☐

DB	DBMS	DBA
WAL 000000	1 DB	DB 1 0000000000000000

[illegible]

□□□□ Syslog

Alloy BSD syslog Alloy Loki
job="network_syslog"



```

syslogAlloy devices:

```

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/CGrateS	→ OCS/CGrateS	JSONRPC
CGrateS RPC	→ CGrateS RPC	RPC /
Syslog	→ Syslog	MikroTik / syslog

- 監視対象のサービス/アプリケーション
- 監視対象のインフラ/ネットワーク
- 監視対象のサービス/アプリケーションP-CSCF, S-CSCF, I-CSCF
- 監視対象のサービス/アプリケーション
- 監視対象のサービス/アプリケーション10 分間隔

監視対象

1. GrafanaのURL `http://<monitoring-host>:3000`
2. 監視対象 → サービス → インフラ
3. サービスの監視対象
4. インフラの監視対象

監視対象

LogQL 検索

Loki の LogQL 検索構文

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

監視対象

監視対象のサービス/アプリケーション

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

監視対象 MME サービス

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

監視対象 CSCF サービス

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

□ **systemd** □□□□

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

□□□□□□

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

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

□□□□□□□□

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

CSCF □□□□□

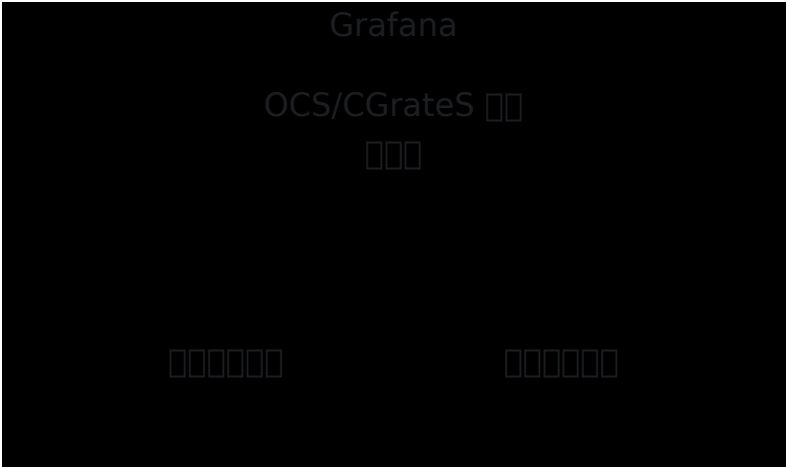
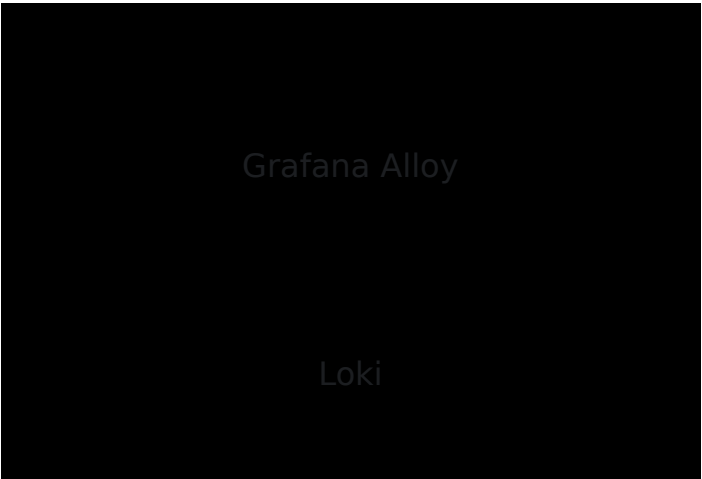
```
sum(rate({component=~"pcscf|scscf|icscf"} |~ "(?i)error" [5m]))
```

CGrateS JSONRPC ☐☐☐☐

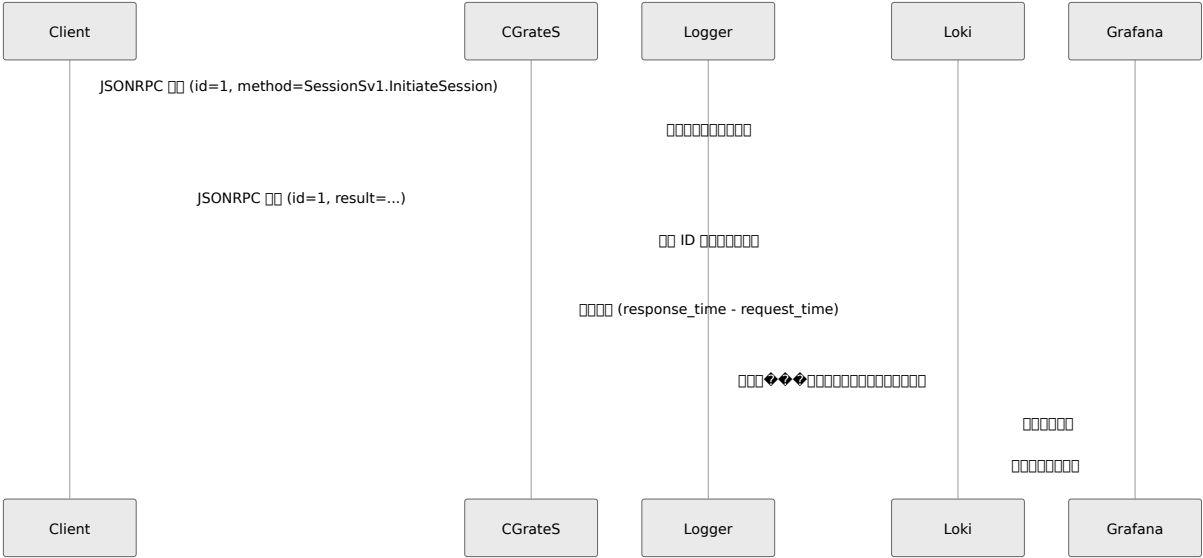
[OCS/CGrateS](#) [JSONRPC](#) [/](#)

```
JSONRPC 0000 cgrates 000000000000 (/var/log/cgrates/jsonrpc.log)  Alloy
000000 ocs 0 cgrates 0000
```





CGrateS/Prometheus



部署方法

CGrateS 設定方法

ポート	サービス	プロトコル	説明
2012	rpc_json	TCP	JSONRPC over TCP
2080	http	HTTP	HTTP API endpoints: /metrics, /health

監視方法

- Prometheus (GET /metrics)
- Health check (GET /health)
- Gzip 対応

ログ

JSONRPC ログ: /var/log/cgrates/jsonrpc.log (JSON形式)

```
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	CGateS 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 / CGateS 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

- 
- 
- 

□ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □

JSONRPC ☐

JSONRPC 与/与

- 
- 
- 
- 

□□□□□□

JSONRPC

SessionSv1.InitiateSession

CGrateS RPC ☐☐☐☐☐

CGRateS RPC API

[illegible]

□□□□□“□□□□□ 1234□□□□□”□□

1. 连接池 → 连接 → **CGrateS RPC** 连接
2. 连接池 连接池 ID 1234
3. 连接池 连接池 连接池
4. **RPC** 连接池 连接池/连接池

CGrateS

□ □ □ □ □

□□□□□□

```
{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

- 安裝預覽器
- ngrep 安裝
- 安裝預覽器
- Alloy 安裝預覽器

安裝預覽器

1. 安裝預覽器

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

2. 安裝 ngrep 安裝預覽器

```
which ngrep
```

3. 安裝預覽器

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

4. 安裝 Alloy 安裝預覽器

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

安裝預覽器

安裝JSONRPC 安裝預覽器“安裝”

安裝預覽器

- 安裝預覽器 JSONRPC 安裝
- 安裝預覽器安裝預覽器

安裝預覽器

1. 安裝預覽器

```
{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 配置OmniCRM

文件	进程名	日志名
<code>/var/log/nginx/access.log</code>	nginx	access
<code>/var/log/nginx/error.log</code>	nginx	error

CGrateS JSONRPC配置OCS

文件	进程名
<code>/var/log/cgrates/jsonrpc.log</code>	cgrates-jsonrpc

配置

配置 Grafana 面板

配置 Loki 数据源

配置

- 配置 Alloy 代理
- 配置 Loki 数据源
- 配置 Grafana 面板
- Alloy 配置 3100 端口 Loki

配置

1. 配置 Alloy 代理

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

2. Loki 서비스 시작

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

3. 모니터링 서버 확인

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

4. 모니터링 서버에 TCP 3100 포트 열기

Alloy 설치

Alloy를 `/var/lib/alloy` 디렉토리에 설치합니다.

설치 방법

- Loki를 모니터링합니다.
- WAL을 사용합니다.

설치 순서

1. Loki를 설치합니다.
2. Loki를 모니터링합니다. WAL을 사용합니다.
3. Loki를 모니터링합니다.
4. Loki 1을 모니터링합니다.

Loki 설치

Loki를 모니터링합니다.

설치 방법

- Loki를 모니터링합니다.
- WAL을 사용합니다.

설치 순서

1. Loki를 설치합니다.

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

2. `du -sh /var/lib/loki/`
3. `du -sh /var/lib/loki/`
4. `du -sh /var/lib/loki/`

部署 Loki

部署 Loki 需要 `component` 和 `infrastructure` 两个组件

部署步骤

- 部署 Loki
- Alloy 部署 Loki

部署步骤

1. 部署 Loki
2. 部署 Ansible 部署 Alloy

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

3. 部署 Alloy

```
systemctl restart alloy
```

部署环境

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

部署步骤

- 部署 Grafana: Grafana, Prometheus, Loki
- 部署 Alloy: 配置 UI
- 部署 Loki: 日志存储
- 部署 Prometheus: 指标存储

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` 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 MGW

- 配置 BSC 和 MGW 接口
- 在 MGW 中配置 `systemd` 的 `CPU` 限制 `CPU Scheduling Policy` `Priority` 和 `Ambient Capabilities`
- 配置 `active` 为 6 个进程

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.omnitouch.com.au/docs/repos/OmniCore> - □□□□□□□□□□

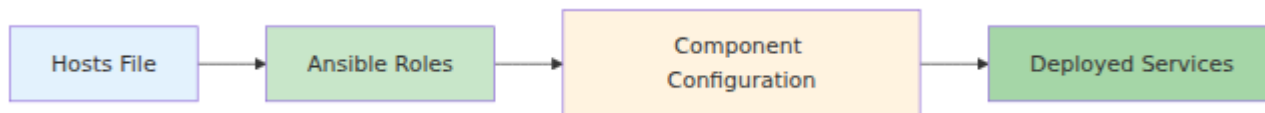
11

□ □ □ □ □ □ □ □ □ □ □ □ □

- **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>

□ □ □ □

OmniCore



OmniCRM group_vars

[illegible]

```
sql_pass: password, omnitouch2024 APT omni/omni
group vars
```

□ □ □ □

□□□□□□□□□□ IP □□□□ □□□□□□□□□□

- 000000
- IP 0000
- 0000

- IP

- hosts-file-configuration.md

cdrs_enabled: True

in_pool: False

populate_crm: False

dra_peer_exclude: True

gtpc_weight: 100

gtpc_priority: 100

CDR

#

CRM

#

SGW/PGW MME

SGW/PGW

Diameter Peer Exclusion dra_peer_exclude

gtpc_weight gtpc_priority MME GTP-C sgwc / pgwc
100MME // gtpc_priority
gtpc_weight RFC 2782
MME MME relative_capacity 100 0-255

(all:vars)

all:vars OmniCRM
Grafana SMTP

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

SSH 接続

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

顧客

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

PLMN

```
plmn_id:
  mcc: '001'          # 3桁のMCC
  mnc: '01'           # 2-3桁のMNC
  mnc_longform: '001' # MNC 3桁の長形式

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

ネットワーク名

ネットワーク

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

UE 設定 > ネットワーク

DNS

```
manage_resolv_conf: True          # 如果 False 那么 Ansible 会
/etc/resolv.conf
```

如果 `manage_resolv_conf` 为 `False` 那么 Ansible 不会修改
`/etc/resolv.conf` 文件 DNS 配置
`all:vars` 文件

APT

`apt_repo` 配置项 `remote_apt_*` `apt_download_base`
URL 配置项

配置项 `apt_cache_servers`

- `use_apt_cache` 配置项 `True` 或 `False`
- `apt_repo.apt_server` 配置项 IP

```
use_apt_cache: False              # 如果 True = 那么 APT 会
apt_repo:
  apt_server: "apt.<region>.omnitouch" # Omnitouch 的 FQDN
  apt_repo_username: "your-username"  # 用户名
  apt_repo_password: "your-password"  # 密码
  # apt_repo_port: 443                 # 端口 443 HTTPS
  # 443 -> https
  http
```

配置项 **HTTPS / 443** 配置项 **HTTP / 8080**
`apt_cache_servers` 配置项 `remote_apt_*` Omnitouch

配置项 **APT**

配置项

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

license_serverのAPI URLをlicense_server_api_urlsで設定する。

設定ファイル

MME 設定

MME 設定はS-GW と P-GW の設定で omnimme の設定は home/roaming の設定で dns_lookup と random_peer の設定で

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

AMF 設定

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

# NAS 設定 NEA0 true omniarf >=20260727 NEA2
NAS 5G NEA2 UE
false NAS NIA2
amf_force_null_cipherng: true
```

SAEGW

```
mtu: 1400
```

```
# 00000000
```

OmniHSS

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

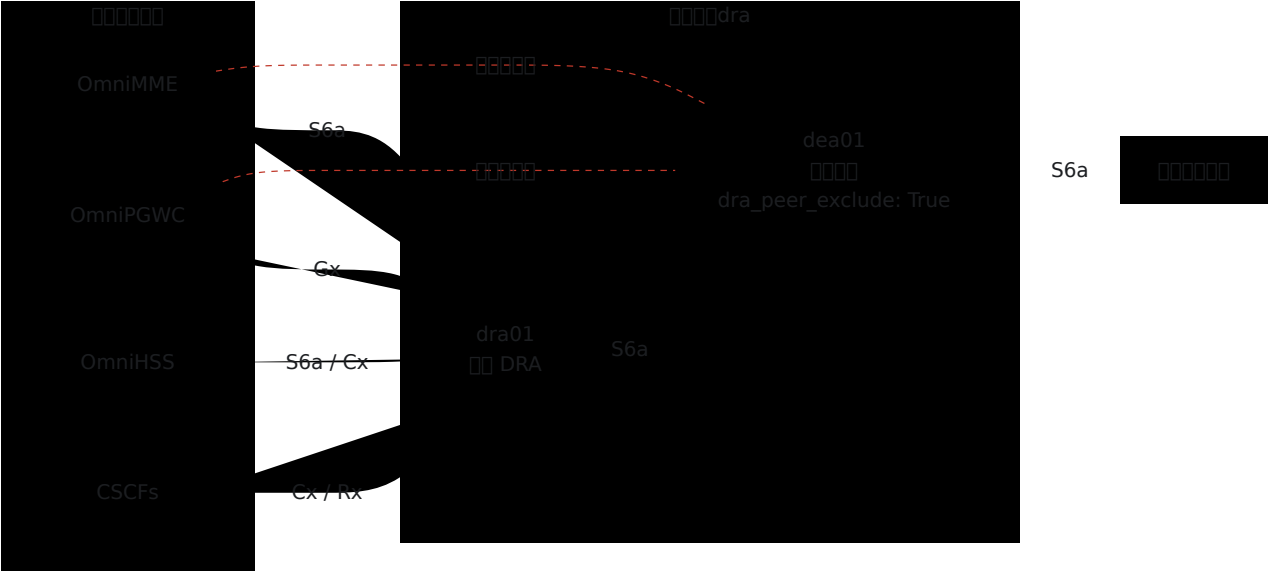
```
omnihss:
  exclude_diameter_peers:
    - customer-as01
    - customer-as02
```

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

`exclude_diameter_peers` `omnihss` `dra_peer_exclude`

OmniCore `dra_peer_exclude`

DEA `dra` OmniDRA `dra_peer_exclude: True` DEA



##

ansible_host

```
dra:
  hosts:
    dra01:
      # dra01 DRA
      ansible_host: 10.4.2.150
    dea01:
      # dea01
      ansible_host: 203.0.113.10
      packet_core_dra_support: False
      ims_dra_support: False
      dra_peer_exclude: True
```

項目	項目名	項目値	項目単位	項目説明
ansible_host	ansible_host	-		ansible_host IP アドレスを指定します。 IP アドレス
dra_peer_exclude	dra_peer_exclude	False		True の場合は、DRA のピアを除外します。 False の場合は、除外しません。
packet_core_dra_support	packet_core_dra_support	False		True の場合は、DRA をサポートします。 S6a、Gx、HSS、PCRF、DEA のサポートは、 dra_module_advanced_routing の設定で指定します。
ims_dra_support	ims_dra_support	False		True の場合は、DRA を IMS の Cx、Rx、Sh のサポートは、 DEA のサポートは、dra_module_advanced_routing の設定で指定します。

項目

ansible_host IP アドレスを指定します。dra DRA のピアを除外します。

項目 dra DRA のピアを除外します。

名前	説明
<code>omnimme</code>	S6a と T6a 間のインターフェース
<code>omnipgwc</code>	Gx と Gy 間のインターフェース
<code>omnihss</code>	S6a と Cx と Sh 間のインターフェース
<code>omniepdg</code>	SWm と S6b 間のインターフェース
<code>omnitwag</code>	S6b 間のインターフェース
<code>omniscef</code>	T6a 間のインターフェース
<code>omnisep</code>	Sh 間のインターフェース
<code>omnilcs</code>	SLg と SLh 間のインターフェース
<code>cscf</code>	P-CSCF と I-CSCF と S-CSCF と Cx と Rx と Dx 間のインターフェース
TAS と SMSC と <code>group_vars</code>	Sh と Ro 間のインターフェース

DRA は `hss`、`mme`、`pgwc`、`epdg`、`scef`、`twag`、`ocs`、`omnilcs`、`omnimessage`、`pcscf`、`icscf`、`scscf`、`xcap`、`applicationserver` と接続する。

インストール

- 必要なパッケージをインストールする。
- 必要なパッケージをインストールする。DRA は `dra_peers` と `diameter_static_peers` を定義する。
- 必要なパッケージをインストールする。Prometheus は IP アドレスを指定する。
- 必要なパッケージをインストールする。runtime.exs を実行する。

デバッグ

デバッグには DEA を使用する。

- MME → PGW-C → DEA → IP → SCTP →
- DEA → allow_undefined_peers_to_connect → False
- →

→ DEA → dra_peer_exclude: True → runtime.exs → DRA

OmniCRM

OmniCRM → group_vars → hosts/ →

→ omnictm_deploy_mode: package → apt_repo → omnictm-api / omnictm-ui → crm_config.yaml → /etc/omnictm/crm_config.yaml → deployment_assets/ → systemctl restart omnictm-api UI → ui: → GET /crm/public/ui-config → yarn → ❗❗ → crm_env_override

→ omnictm_deploy_mode: source → git → + → venv + yarn → crm_env_override

```
omnictm_deploy_mode: package      # package ( ) | source
omnictm_package_version: ""      # apt omnictm-api/ui
crm_config.yaml                  # API + ui: group_vars
deployment_assets/               # /
/etc/omnictm/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

→ Grafana → grafana_smtp_user → grafana_smtp_password →
→ grafana_smtp_host → in-v3.mailjet.com:587 →
roles/monitoring/templates/grafana/grafana.ini

```
grafana_smtp_user: "smtp-user"          # 邮箱用户名
grafana_smtp_password: "smtp-password"  # 邮箱密码
grafana_smtp_host: "in-v3.mailjet.com:587" # 邮箱服务器地址
```

IMS 配置

```
ims_dra_support: False          # 是否支持 DRA 功能
sipcapture_client_enabled: False # 是否启用 SIP/RTP 抓包功能
                                # siptrace 是否启用
```

RAN 配置

```

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

```

□□□□□

□ services/firewall.yml □□□□□□□□□□□□□□□□/□□□□□ □□□□□□ □□□□□□

allowed_oam_subnets □□□□□□

```

firewall:
  allowed_oam_subnets:          # [] []SSH []UI [] [] [] API []
    - '10.0.1.0/24'
    - '10.0.0.0/24'
  allowed_ran_subnets: []      # SIAP / NGAP / GTP-U / Abis
  allowed_ue_voice_subnets:    # [] UE SIP [] P-CSCF
    - '10.0.1.0/24'
  allowed_carrier_voice_subnets: # [] [] SIP
    - '10.0.1.0/24'
  zones:                        # [] [] - [] docs/firewall.md
    internal: ['10.0.0.0/8']
  ue_pools: []

```

[] 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 deployment_type vm proxmoxServers Proxmox

VM

```
proxmoxServers:
  customer-prmx01:
    # deployment_type vm → 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
    local_users:
      proxmoxTemplatePassword: omnitouch # cloud-init
    cloud-init:
      storage: SSD_RAID0 # VM
```

LXC

```

proxmox_lxc_nameserver: "1.1.1.1" # 1.1.1.1 LXC
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 # rootfs

```

proxmox_lxc_nameserver

```

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

```

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>

目錄

序

本書介紹OmniTouch如何透過Ansible實現自動化部署，並提供4G/5G網路。

本書IP地址使用IP地址，IP地址。

目錄

0. 環境準備

本書Proxmox環境準備VM/LXC。

```
# 在Proxmox環境VM/LXC環境部署deployment_type環境  
ansible-playbook -i hosts/Customer/hosts.yml  
util_playbooks/proxmox.yml
```

本書Proxmox VM/LXC環境。

1. 設定ファイルの作成

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

hss:
  hosts:
    customer-hss01:
      ansible_host: 10.10.2.140

# ... 設定ファイル
```

設定ファイルの作成

2. group_varsの作成

group_varsの作成

設定ファイルの作成
OmniMessage SMS Scenarios TAS SIP Diameter

OmniCRMの作成
group_varsの作成
delegate_to: localhost

設定ファイルの作成

3. APTのインストール

```
# APTのインストール
apt_repo:
  apt_server: "10.254.10.223" # IPアドレスrepo
  use_apt_cache: false # true = false = repo
```

設定ファイルの作成

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 --limit=myhost
ansible-playbook services/all.yml --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 repo

Ansible hosts/

7. B omnicore .deb

OmniCore omnicore 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](#)
- [Docker](#) - [Tutorial](#)
- [Grafana](#) - [Prometheus](#) - [Tutorial](#)
- [Loki](#) - [Alloy](#) - [Tutorial](#)

OmniCore

OmniCore is a 4G/5G core network solution.

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

iptables

iptables 是 Linux 内核中实现防火墙规则的工具，它通过配置 iptables 规则表来管理网络流量。iptables 规则表包含链（chain）和规则（rules），链用于分组规则，规则用于匹配数据包并执行相应的动作。

- 配置 iptables 规则表
- 使用 iptables 规则表进行网络流量控制

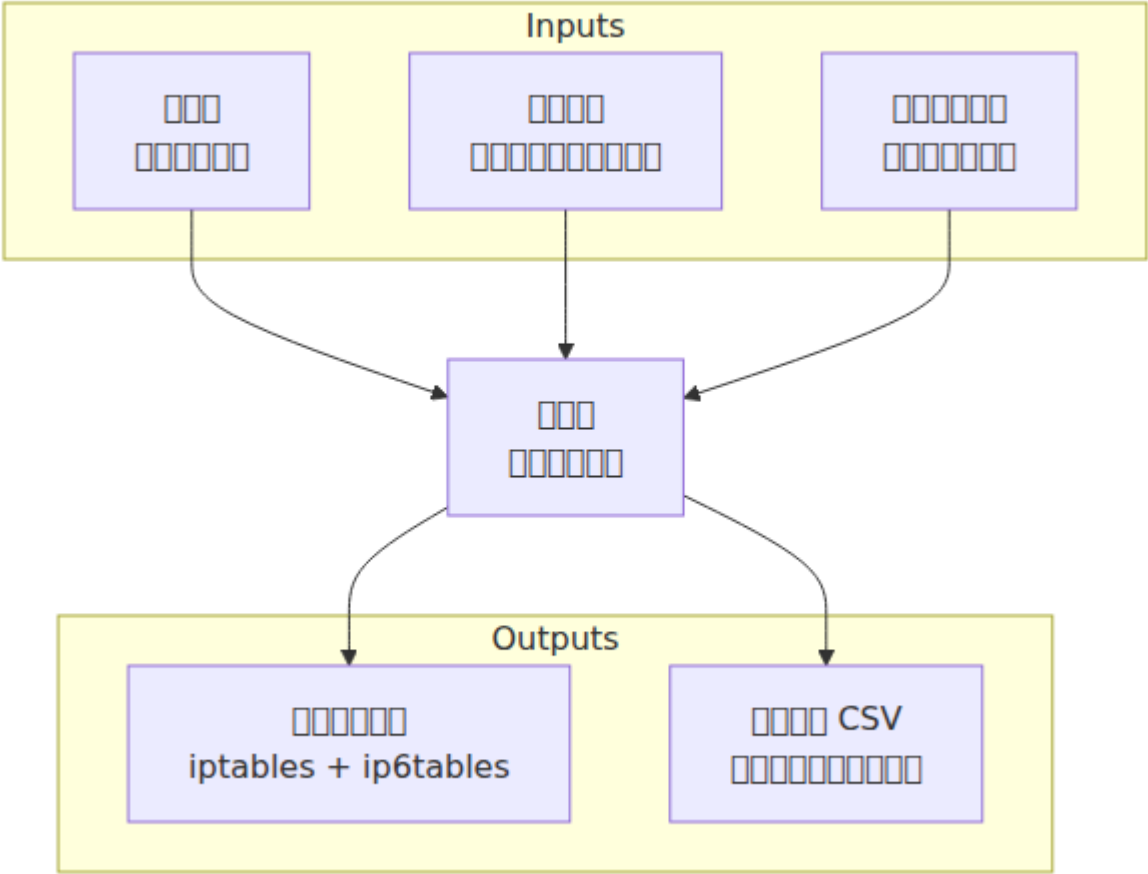
iptables 规则表包含链（chain）和规则（rules），链用于分组规则，规则用于匹配数据包并执行相应的动作。

配置

- 配置 iptables 规则表
- 配置 iptables 规则表
- 配置 iptables 规则表
- 配置 iptables 规则表
- 配置 iptables 规则表
- 配置 iptables 规则表
- 配置 iptables 规则表
- 配置 iptables 规则表
- 配置 iptables 规则表
- 配置 iptables 规则表
- 配置 iptables 规则表
- 配置 iptables 规则表
- 配置 iptables 规则表
- 配置 iptables 规则表

配置

iptables 规则表包含链（chain）和规则（rules），链用于分组规则，规则用于匹配数据包并执行相应的动作。



kernel+userspace



kernel

kernel+userspace kernel+userspace kernel+userspace kernel+userspace kernel+userspace - kernel+userspace kernel+userspace kernel+userspace kernel+userspace kernel+userspace

```
- id: '13' # kernel+userspace
  dst: hss # kernel+userspace
  service: diameter # [ hss kernel+userspace (->
3868/tcp+sctp)
  purpose: 'Diameter' # kernel+userspace
  sources: [{ zone: internal }] # kernel+userspace
```


配置	说明
<code>{ zone: <name> }</code>	指定 CIDR <code>firewall.zones.<name></code>
<code>{ list: <name> }</code>	指定 <code>allowed_oam_subnets</code>
<code>{ group: <name> }</code>	指定 <code>allowed_oam_subnets</code> 中的组
<code>{ ue_pools: true }</code>	启用 UE / 指定 <code>firewall.ue_pools</code>
<code>{ dra_peers: true }</code>	DRA 指定 Diameter 配置
<code>{ net_devices: true }</code>	指定 <code>all.vars.<vendor>.devices</code> 中的设备 <code>mikrotik.devices</code>
<code>{ roaming: true }</code>	启用 IR.21 / GRX-IPX 漫游
<code>{ literal: "..."</code> <code>}</code>	指定 literal 配置

配置

配置 `firewall_services.yml` 中的 `cscf` 配置

```
# 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)
    postgres: 5432/tcp # HSS 数据库
```

firewall_services.yml service: dst
hss diameter
firewall_services.yml

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

- - firewall_services.yml
- -

MME omnics LCS SBc-AP
29168/sctp SLs 9082/sctp
firewall_services.yml

all.vars firewall

internal
IMS

```

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
内部网络
  internal:                      ['10.8.0.0/16']           # 内部网络
移动核心
  mobile_core:                   ['10.8.1.0/24']           # NF
EPC + 5GC + CS 网络
  core_svc:                      ['10.8.2.0/24']           # 核心网络
HSS DRA OCS OAM
  ims:                           ['10.8.3.0/24']           # IMS 网络
CSCF TAS
  ue_svc:                        ['10.8.4.0/24']           # UE 网络
P-CSCF DNS ePDG
  core_v6:                       ['2001:db8:0:1::/64']      # NF IPv6
IPv6 网络 UE
  ue_pools:                      ['100.64.0.0/16', '2001:db8:64::/48'] # UE / 网络
IP 网络 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 [] RAN []
allowed_ran_subnets: [] IPv6 [] core_v6: [] UE []
ue_pools: [] 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 管理用サブネット 管理用サブネット 管理用サブネット
<code>allowed_ran_subnets</code>	必須	必須	[]	RAN 管理用サブネットS1AP NGAPGTP-UAbis RAN 管理用サブネット
<code>allowed_ue_voice_subnets</code>	必須	必須	[]	UE SIP P-CSCF 管理用サブネット
<code>allowed_carrier_voice_subnets</code>	必須	必須	[]	SIP 管理用サブネット
<code>zones</code>	必須	必須	-	管理用サブネット 管理用サブネット
<code>ue_pools</code>	必須	必須	[]	UE / IP 管理用サブネット 管理用サブネット
<code>extra_rules</code>	必須	必須	[]	管理用サブネット 管理用サブネット

管理用サブネット

管理用サブネット CIDR 管理用サブネット

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ネットワーク

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	必須	override	コメント

モード

firewall_mode: 1

モード	説明
monitor	パケットは INPUT によって ACCEPT され、FW-AUDIT によってパケットの送信元と宛先が記録される。
enforce	パケットは FW-DROP によって INPUT からの FORWARD によって DROP される。

デフォルトでは **enforce** モードでパケットの送信元と宛先が記録される。パケットの送信元と宛先が記録されるモードは **monitor** モードで `firewall_mode: monitor` と指定する。パケットの送信元と宛先が記録されるモードは **enforce** モードで `firewall_mode: enforce` と指定する。

```
all:
  vars:
    firewall_mode: monitor # デフォルトは 'enforce'
```

`firewall_mode` は `all.vars` から取得される。デフォルトは `enforce` である。

設定ファイル

パケットの送信元と宛先が記録されるモードは `prod_master_peers.yml` ファイルで `{ group: <role> }` と指定する。パケットの送信元と宛先が記録されるモードは `prod_master_peers.yml` ファイルで `{ group: <role> }` と指定する。

設定ファイル

パケットの送信元と宛先が記録されるモードは `ansible_host` と `secondary_ips.<name>.ip_address` から取得される。パケットの送信元と宛先が記録されるモードは `ansible_host` と `secondary_ips.<name>.ip_address` から取得される。パケットの送信元と宛先が記録されるモードは `ansible_host` と `secondary_ips.<name>.ip_address` から取得される。

パケットの送信元と宛先が記録されるモードは `ansible_host` と `secondary_ips.<name>.ip_address` から取得される。パケットの送信元と宛先が記録されるモードは `ansible_host` と `secondary_ips.<name>.ip_address` から取得される。

パケットの送信元と宛先が記録されるモードは `ansible_host` と `secondary_ips.<name>.ip_address` から取得される。パケットの送信元と宛先が記録されるモードは `ansible_host` と `secondary_ips.<name>.ip_address` から取得される。

SIGTRAN 環境構築環境構築環境構築環境構築 NIC環境構築環境構築環境構築

00 / 0000

環境構築環境構築環境構築環境構築環境構築環境構築/環境構築環境構築環境構築環境構築環境構築

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

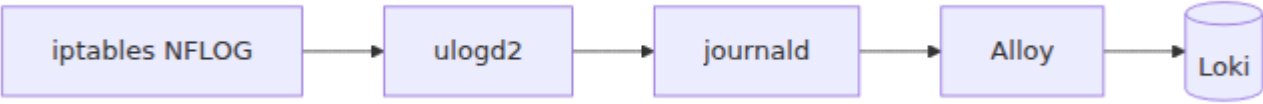
0000 generated/firewall-port-matrix-exploded.csv環境構築環境構築環境構築

項目	説明
id	000000
service	000000
source	000000 CIDR 0000
destination_role	000000
destination_host	00000000
destination_ip	00000000
port	0000000000000000 ICMP 0 ESP 0000
protocol	000

環境構築環境構築環境構築環境構築環境構築環境構築環境構築環境構築環境構築環境構築

0000

```
環境構築環境構築環境構築 firewall_rollback_guard環境構築環境構築
firewall_rollback_timeout 000
```



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

名称	类型	是否必填	默认值	说明
firewall_rollback_guard	字符串	否	true	防火墙回滚保护开关
firewall_rollback_timeout	整数	否	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 策略 DROP 配置 IPv4 地址池 IPv6 地址池 IPv6 地址池

配置

配置 Loki 配置

項目	説明
FW-FLOW / FW-FLOW6	パケットフローのログを記録する
FW-AUDIT / FW-AUDIT6	パケットの送信元・宛先・ポート番号などを記録する
FW-DROP / FW-DROP6	パケットの拒否理由を記録する

インストール

まず `NFLOG` を有効にする。 `ulogd2` をインストールし、 `iptables-persistent` に `NFLOG` を有効にする。 - `ulogd2` のインストールと設定は、 `ulogd2` のインストールガイドを参照してください。

また、 `Omnitouch apt` をインストールし、 `ubuntu_packages.networking` をインストールする。

Alloy のインストールは、 `unit="ulogd2.service"` の `service="ulogd"` を指定して、 `job="firewall"` の `job` を指定して、 `unit` を指定してインストールする。

Loki のインストール

```
# インストール
{unit="ulogd2.service"} |= "FW-AUDIT"

# インストール
sum by (hostname) (count_over_time({unit="ulogd2.service"} |= "FW-DROP" [5m]))

# インストール
{unit="ulogd2.service", hostname="omni-tampa-sbc01"} |= "FW-DROP"
```

ログの確認

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

□□□□□□

- □□□□□□□□□□ □□ `roaming` □ `literal` □□□□□□□□□□□□□□ (0.0.0.0/0)□□□□ □□□□□□ CIDR □□□□□□□□□□
- **ICMP** □□□□□□□□ □□□□□□□□□□ ICMP□□□□ ping □□□□□□□□□□□□
- □□□□□□□□□□□□□□□□□□□□ □□□□ `FORWARD` □□□□ `DROP`□□□□□□□□□□□□ UPF□ PGW □ SGW □□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□
- □□□□□□□□ `allowed_oam_subnets` □□ SSH□UI□□□□□□□□ API□□□□□□□□□□□□□□□□□□□□ □□□□□□ `extra_rules` □□□□□□□□□□

Ansible

Ansible

`group_vars` 目录结构

目录结构如下：SIP 目录 Diameter 目录 SMS 目录
目录结构 `group_vars` 目录

目录 `hosts/{Customer}/group_vars/`

目录结构如下：
`group_vars_omnicore - group_vars/` 目录结构如下：
`inventory_dir.rsplit('/', 1)[0] }}` `group_vars/` 目录结构如下：
`group_vars_omnicore }}` 目录结构如下：

Ansible

Ansible 目录结构如下：
目录 `group_vars` 目录结构如下：

目录结构 → `group_vars` 目录结构 → 目录

1. 目录结构 OmniMessage

目录结构 Jinja2 目录

目录

```
hosts/Customer/  
└─ group_vars/  
    └─ smsc_controller.exs      # 目录结构
```

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/                 # テスト用
│   │   └── 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/                 # テスト用
│       └── 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` フォルダを作成し、OmniTAS のダイヤルプランを配置
4. Ansible を実行

確認とトラブルシューティング

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

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

📄📄 3📄📄📄📄📄📄📄 OmniHSS📄

📄📄📄📄📄📄📄📄📄📄📄📄📄📄

📄📄📄📄

```
hosts/Customer/
└─ group_vars/
   └─ hss_runtime.exs.j2          # 📄📄📄📄 HSS 📄📄📄
```

📄📄📄📄📄📄📄

```
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  # 📄 📄 group_vars 📄📄
📄📄📄📄📄
```

📄📄📄📄📄

1. Ansible 📄📄 `hss_template_config: hss_runtime.exs.j2`
2. 📄 `hosts/Customer/group_vars/hss_runtime.exs.j2` 📄📄📄
3. 📄📄 Jinja2 📄📄📄📄📄📄📄📄 `{{ inventory_hostname }}`📄`{{ 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

```
hosts/Customér/  
└─ group_vars/  
   └─ mme_runtime.exs.j2          # omnimme MME 000
```

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

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

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

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

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

5. `omnimme`

```

00000000 mme_template_config 0000000000000000

```

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

5 OmniCRM

OmniCRM の group_vars を API によって取得する

```
□□□□□□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          # package source package
      git+yarn
      # omnicrm_package_version: "20260810...." # apt pin
      logo_light: logo-light.png           # logo
      deployment_assets/logos/
      logo_dark: logo-dark.png
      login_background: login-bg.jpg        # →
      deployment_assets/login/auth-one-bg.jpg (login_bg.jpg)
      login_splash: splash.jpg              # →
      deployment_assets/login/login-side.jpg (login_side.jpg)
      ssl_fqdn: portal.example.com
      # crm_env_override: customer_crm.env  # crm_env_override
      cell_sites: cellSites.txt             # cell_sites
      message_templates: constants.js       # message_templates
      site_data: site_data.json             # site_data
```

ansible-playbook

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

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

ansible-playbook

1. ansible crm_config.yaml package /etc/omnicrm/OmniCRM-API/ package

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.yaml` の `group_vars` を `omnihss/` にインストール

インストール

```
hosts/Custom/  
└─ 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

- インストール `{{ group_vars_omnicore }}/omnihss/subscribers.csv` をインストール
- インストール `omnihss/` の JSON を `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 表

項目	グループ	説明
smc_template_config	omnimessage	Jinja2 テンプレート smc_controller.exs
smc_smpp_template_config	omnimessage_smpp	Jinja2 テンプレート smc_smpp.exs
gateways_folder	applicationserver	ディレクトリ gateways_prod
dialplan_folder	applicationserver	ディレクトリ dialplan ディレクトリ dialplan
tas_runtime_template	applicationserver	Jinja2 テンプレート tas_runtime.exs.j2 - テンプレート tas_runtime.exs.j2
hss_template_config	omnihss	Jinja2 テンプレート hss_runtime.exs.j2
mme_template_config	omnimme	Jinja2 テンプレート mme_runtime.exs.j2
dra_template_config	dra	Jinja2 テンプレート dra_runtime.exs.j2
pgwc_template_config	pgwc	Jinja2 テンプレート 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

- 設定する環境変数

1. 環境変数 **group_vars** の設定

- 環境変数IPアドレス - 設定する
- 環境変数 - 設定する
- 環境変数 - 設定する group_vars

環境変数

- 環境変数 - 設定する
- 環境変数 - 設定する
- **OmniCall** の設定: <https://docs.omnitech.com.au/docs/repos/OmniCall> - 設定する
- **OmniCore** の設定: <https://docs.omnitech.com.au/docs/repos/OmniCore> - 設定する

Ansible

概要

Ansibleは、システム管理のための自動化ツールです。

- 簡単にインストール
- 簡単にIPを指定
- 簡単にタスクを定義
- 簡単にPLMNを定義

インストール

Ansibleは、`hosts/` ディレクトリに `ansible.cfg` ファイルを配置し、Ansible の `inventory` を指定します。

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

Ansibleは、`Production.yml`、`Staging.yml`、`lab.yml` を指定して実行します。

実行

Ansibleは、`ansible` コマンドで実行します。

```
# 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 □□ IP □□
      gateway: 10.10.1.1           # □□□□
      host_vm_network: "vmbr1"     # □□□□□□□□ NIC □□
```

□□ □ IP □□□□□□□□□□□□□□□□ IP □□□□□□□□□□ OmniCore □□□□□□□□□□

Proxmox □□□ `host_vm_network` □□□□□□□□□□□□□□□□□□□□□□□□ **Proxmox**
VM/LXC □□□

□□□□□□□□

□□□□□□□□□□

```
num_cpus: 4                # CPU □□□
memory_mb: 8192            # □□□□ MB □□□□
proxmoxLxcDiskSizeGb: 50   # □□□□□□ GB □□□□
```

□□□□□□□□

□□□□□□□□□□□□□□□□□□

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
```

□ □

□□□□□□

```
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

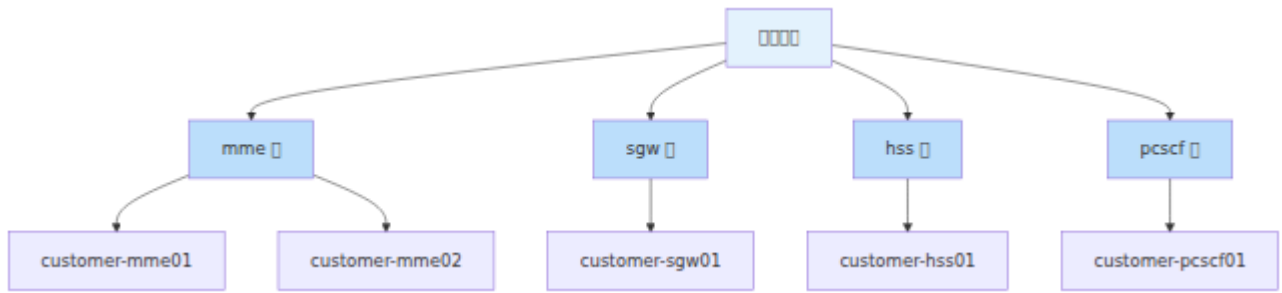
    # APT
    # apt_cache_servers
    # use_apt_cache true apt_repo.apt_server
    # IP
    apt_repo:
      apt_server: "10.254.10.223"
      apt_repo_username: "customer"
      apt_repo_password: "secure-password"
    use_apt_cache: false

    # 
    TZ: America/Chicago

```

Ansible

Ansible



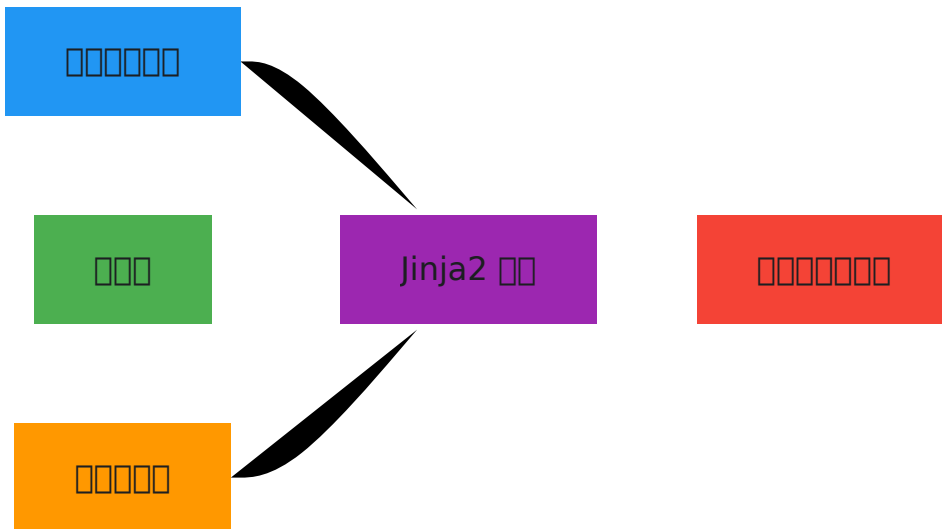
core mme mme:hosts:

ocs crm smsc omnimessage dra xcap siptrace 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"

# 例題 - 例題 docs/firewall.md
firewall:
  allowed_oam_subnets:
    - '10.0.0.0/8'
    - '192.168.0.0/16'
  allowed_ue_voice_subnets:
    - '10.0.0.0/8'
  zones:
    internal: ['10.0.0.0/8']

# 例題 Proxmox VM 例題
proxmoxServers:
  customer-prmx01:
    # deployment_type 例題 → 例題 "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 # 例題 cloud-init 例題
    local_users {
      proxmoxTemplatePassword: omnitouch # 例題 cloud-init 例題
    }
    cloud-init {
      # 例題 LXC 例題 proxmoxServers 例題 deployment_type: lxc {
      # proxmoxLxc0sTemplate{

```

例題 Proxmox VM/LXC 例題 例題 Proxmox 例題

□□□□□□

□□□□□□□□□□□□□□□□□□

OmniCore □□□

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

OmniCall □□□

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

OmniCharge/OmniCRM□

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

□□□□

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

- APT 0000 - 000
- 000000 - 00000
- 000000 - 000000

000

1. 0000000000000000
2. 0000 PLMN 00000
3. 00 APT 00000
4. 00000000 0000
5. 0000000 group_vars 000
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` 设备。
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/0mnicore_Customer/hosts/0mnicore_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

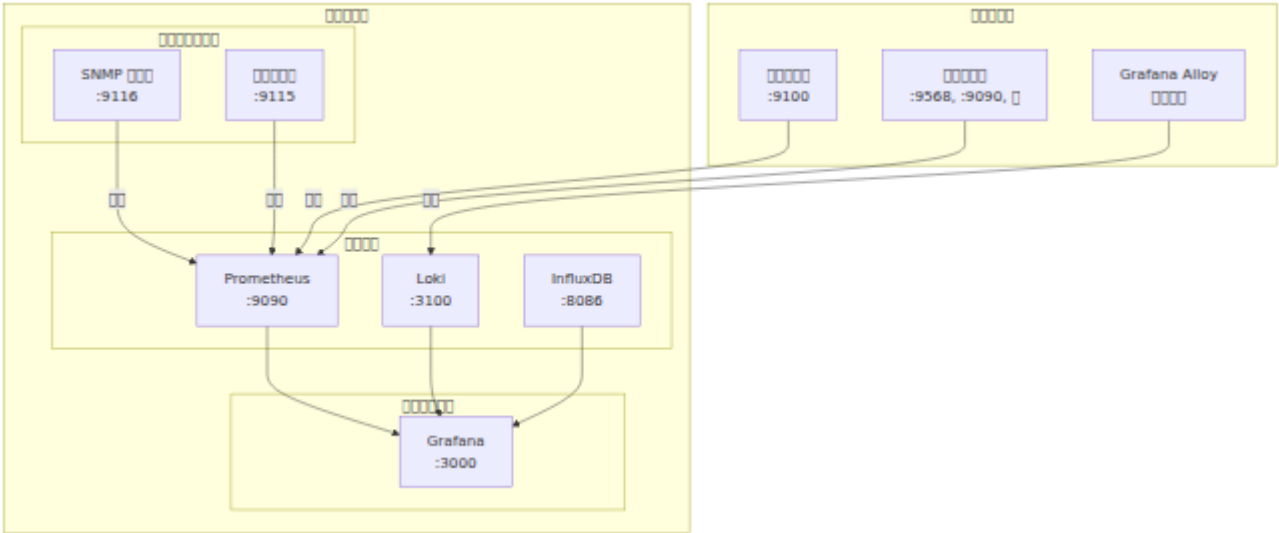
Omnitouch DNS DoH/DoT DNS DNSSEC DNS

-
-

部署架构图

部署

OmniCore 部署架构图



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 个虚拟机上部署 OmniCore 网络

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	コア/コア
HSS	hss	9568	コア
SGW-C	sgw	9568	コア GTP-C
PGW-C	pgwc	9090	PDN コア IP
UPF コア	upf	9090	コア
CSCF	pcscf, scscf, icscf	9090	コア
コア FreeSWITCH	applicationserver	9090	コア
DRA	dra	9568	コア
OmniMessage	omnimessage	9568	コア SMPP
OmniSS7	omniss7	8080	SS7/SIGTRAN コア
KeyDB	ocs	9121	コア/
CGrateS	ocs	2080	コア CDR

コア

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 は、タイムシリーズデータを保存するためのデータベースです。

UID: omnicores_influxdb

UID

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

ダッシュボード

ダッシュボードのリスト

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

```
├── BSS/
│   ├── KeyDB_Cluster.json
│   ├── cgrates_mysql.json
│   └── cgrates_stats.json
├── EPC/
│   ├── MME_Dashboard.json
│   ├── OmniHSS.json
│   ├── OmniDRA.json
│   ├── SGW.json
│   ├── PGWs.json
│   └── ...
├── IMS/
│   ├── SMS.json
│   ├── MM.json
│   └── ...
├── 应用/
│   ├── Node_Exporter_Full.json
│   ├── Mikrotik_Dashboard.json
│   └── ...
├── RAN/
│   ├── Nokia_Overview.json
│   ├── Nokia_Detailed.json
│   └── ...
└── 其他/
    ├── CSCF_Logs.json
    ├── MME_Logs.json
    └── ...
```

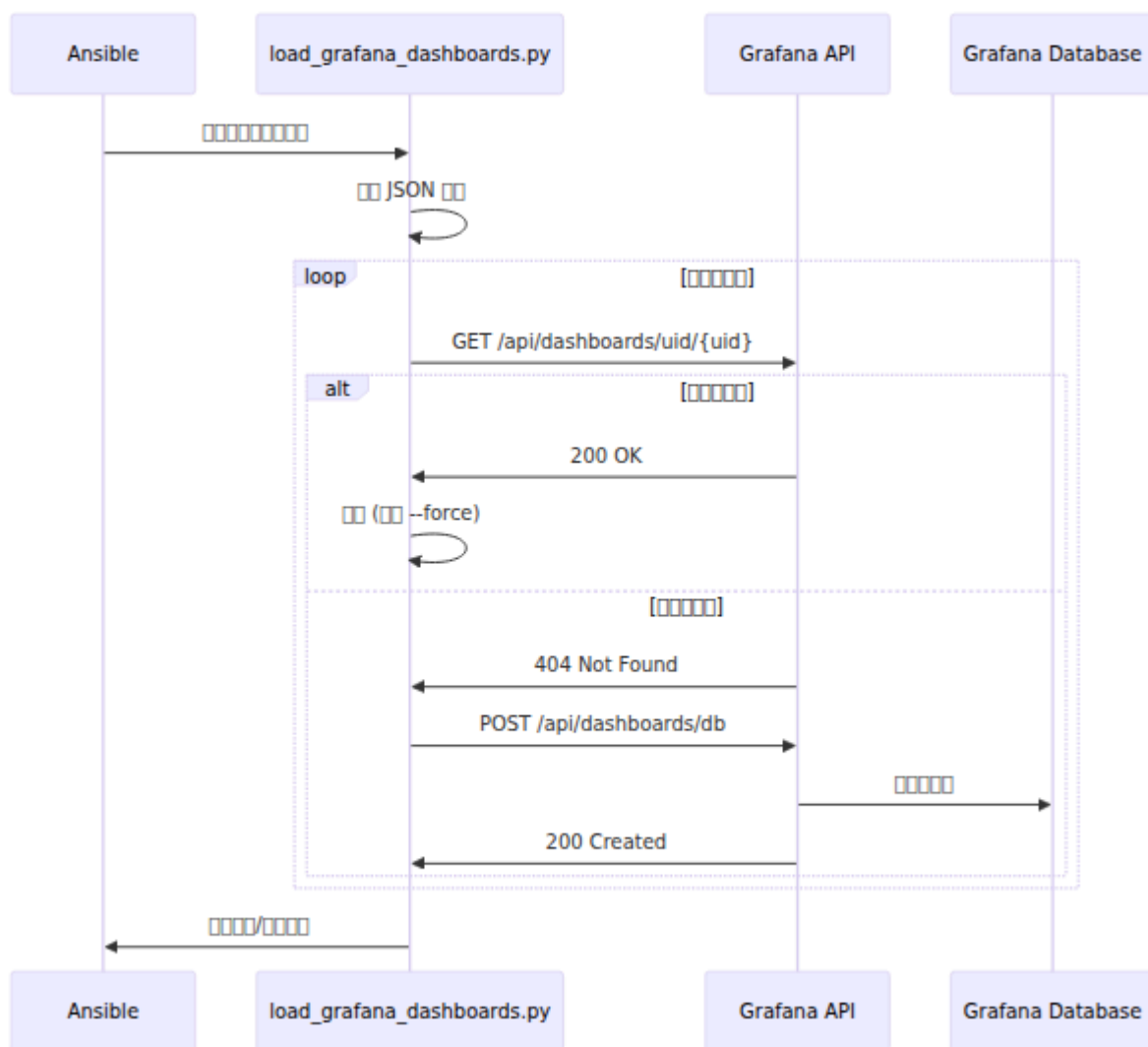
□□□□□ Grafana □□□□□□□□□□□□□□□□□□□□

□□□□□

- `roles/monitoring/templates/grafana/dashboards/{folder}/*.json` (grafana dashboard templates)

)

- `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` `id`
- Grafana `id`

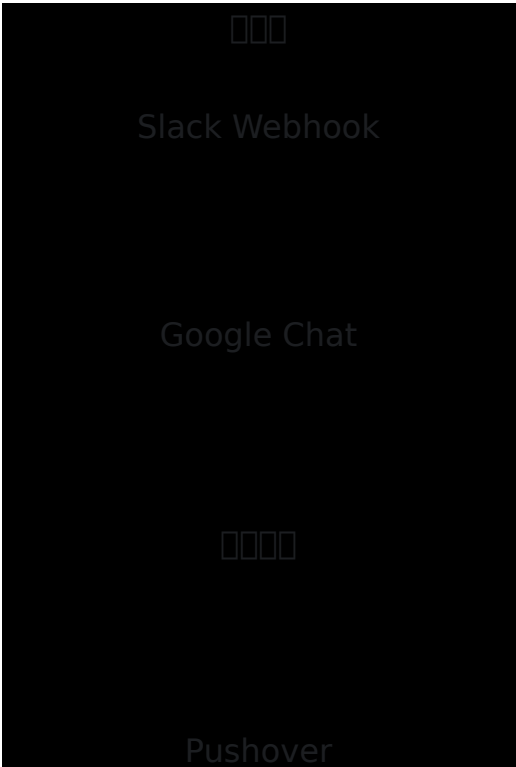
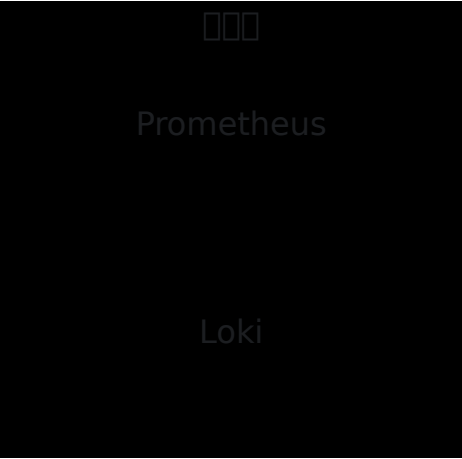
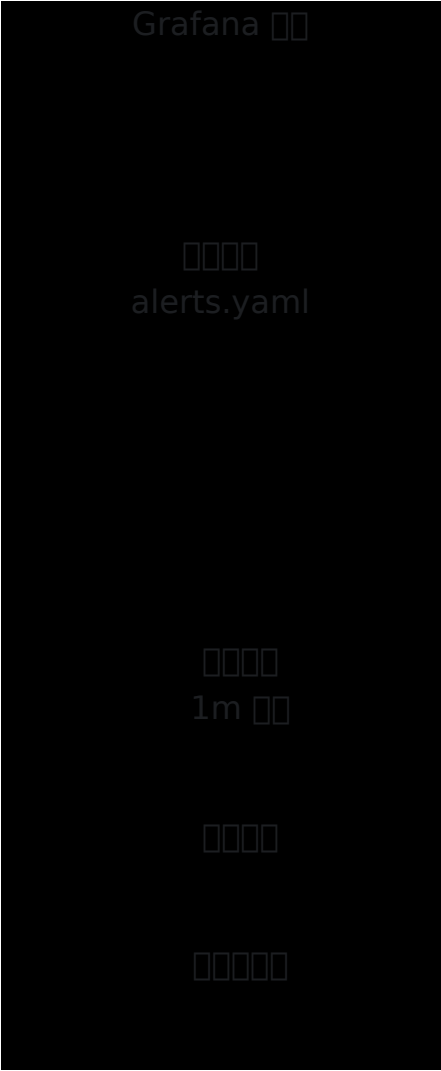
- 0000000000000000

0000000

0000000000000000 group_vars/grafana/dashboards/ 0000000000000000

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







概要

この章では Grafana API を利用して、Grafana の YAML を利用して Grafana の alerting (/etc/grafana/provisioning/alerting/alerts.yaml) を Grafana の UI/API から管理する方法について説明します。

この章では hosts/0mnicore_<CUSTOMER>/group_vars/grafana/alerts/ 以下の alerts.yaml を load_grafana_from_export.py を利用して Grafana の UI/API から管理する方法について説明します。

```
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
```

この章では roles/monitoring/files/load_grafana_alerts.py を利用して alerts.yaml を Grafana の UI/API から管理する方法について説明します。

```
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
High CPU usage 90%	Network	node_cpu{instance=~".*"} > 90%	5 min
CPU usage 90%	Network	node_cpu{instance=~".*"} > 90%	5 min
High memory usage 90%	Network	node_memory{instance=~".*"} > 90%	5 min
High network traffic	Network	Prometheus node_network_bytes{instance=~".*"} > 100MB	5 min
MME connection 40%	EPC	mme_connection{instance=~".*"} > 40%	30 min
MME connection 0	EPC	mme_connection{instance=~".*"} < 0	5 min
IMS connection 40%	IMS	p_cscf_connection{instance=~".*"} > 40%	30 min
IMS MOS 4	IMS	ims_mos{instance=~".*"} < 4	5 min
eNodeB connection	RAN	enb_connection{instance=~".*"} > 1	1 min
High network traffic	EPC	P95 network_traffic{instance=~".*"} > 100MB	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"
```

SlackGoogle ChatPushover

roles/monitoring/templates/grafana/contact-points.yaml.j2Pushover
 notifications.pushover enabledapi_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
```

0000

Prometheus 00

00000000 /etc/prometheus/prometheus.yml 0

00	00	00
scrape_interval	1m	00000000
evaluation_interval	1m	0000000000
scrape_timeout	50s	00000000

Grafana 00

00000000 /etc/grafana/grafana.ini 0

00000000000000

00	0	00
allow_embedding	true	00 iframe 00
provisioning	/etc/grafana/provisioning	0000

00000000 grafana.ini 0000 admin_password 0000000000000000
grafana_admin_password 0000000000000000 API 00000000/00000000

0000 (SMTP) 00

00000000 group_vars 000 SMTP 00000000
roles/monitoring/templates/grafana/grafana.ini 00

項目	型	値	説明
<code>grafana_smtp_user</code>	文字列	<code>(/)</code>	SMTP 送信者アドレス 送信者名を指定する オプション
<code>grafana_smtp_password</code>	文字列	<code>(/)</code>	SMTP 送信者パスワード 送信者パスワード
<code>grafana_smtp_host</code>	文字列	<code>in-v3.mailjet.com:587</code>	SMTP 送信先:ポート
<code>grafana_smtp_enabled</code>	ブール値	<code>true</code>	SMTP 送信を有効にする

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

Loki

Loki

InfluxDB

```
/etc/influxdb/influxdb.conf
```

名前	タイプ	値
<code>nokia-monitor</code>	<code>autogen</code>	30
<code>dra</code>	<code>autogen</code>	365
<code>Omnicharge_TAP3</code>	<code>autogen</code>	90

部署

组件	端口	协议	说明
Grafana	3000	HTTP	Grafana UI 和 API
Prometheus	9090	HTTP	Prometheus 服务
Loki	3100	HTTP	Loki 服务
InfluxDB	8086	HTTP	InfluxDB API
节点导出器	9100	HTTP	节点导出器
SNMP 导出器	9116	HTTP	SNMP 导出器
日志导出器	9115	HTTP	日志导出器
Alloy	12345	HTTP	Alloy UI 和 API

部署

部署

- 部署 Grafana 到 `http://<monitoring-ip>:3000`
- 部署 Prometheus 服务
- 部署 Loki 服务

部署

- 部署 Grafana 到 `http://<monitoring-ip>:3000`
- 部署 Prometheus 服务 **Loki** 服务
- 部署 LogQL 服务

```
# 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 Grafana 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 "XXXX" 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 -y libfontconfig libssl-dev`
2. 安裝 Grafana 包: `sudo apt-get install grafana`
3. 啟動 Grafana 服務

配置 Grafana

目標: 配置 Grafana 服務

步驟:

- 配置
- 安裝
- 配置

步驟:

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

Grafana 安裝與配置

目標: Grafana 安裝 "Grafana" 服務

步驟:

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

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'
```

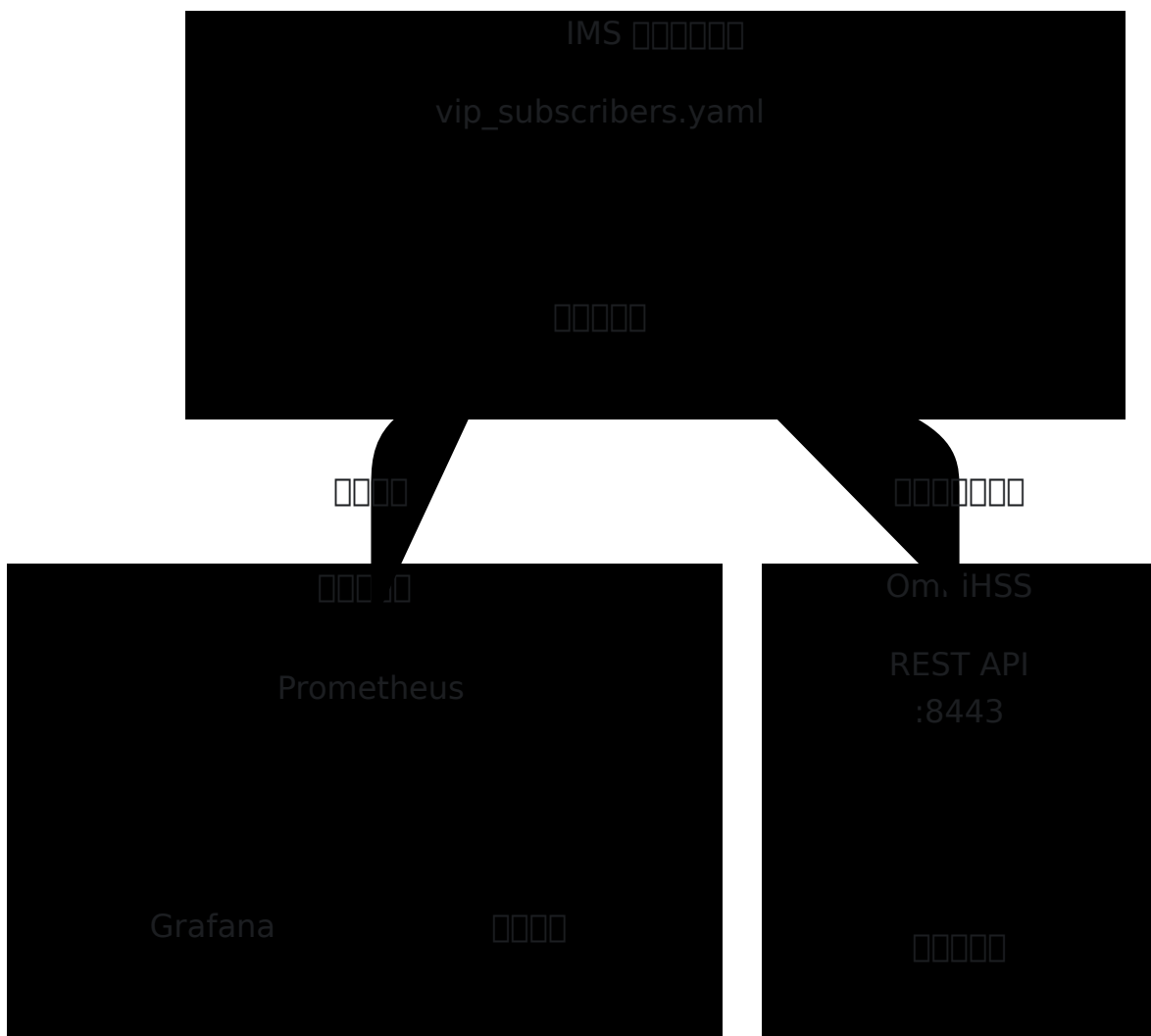
3. 检查配置

```
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 部署需要部署 IMS 和 EPC 和 UE

11



--	--	--	--

□ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □

プロファイル	IMS プロファイル	EPC プロファイル	サービス
full	有	有	音声、データ、SMS、MMS
voip_only	有	有	VoIP、データ、IP 音声、SMS
data_only	有	有	データ、SMS、M2M

0000

- **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
<code>hss_url</code>	URL	-	OmniHSS REST API URL (HTTPS)
<code>scrape_interval_seconds</code>	秒	30	スクレイピング間隔
<code>ping_timeout_seconds</code>	秒	2	UE IP ping タイムアウト
<code>ping_count</code>	回	2	ping 回数
<code>blackbox_exporter_url</code>	URL	-	ping 結果を出力する URL (例: <code>http://pcscf01:9115</code>)。ICMP ping 結果を出力。
<code>subscribers</code>	数	-	サブスクリプション数

出力

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

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	000 IMS 000000 00 (-1 000000)
<code>vip_subscriber_epc_registration_age_seconds</code>	Gauge	000 EPC 00000 000000 (-1 0000 00)

接口

接口	类型	说明
<code>vip_subscriber_info</code>	Gauge	当前 1 分钟内 VIP 用户数

接口参数:

参数	描述	示例
<code>imsi</code>	IMSI	<code>001010000000002</code>
<code>label</code>	标签	<code>label</code>
<code>msisdn</code>	手机号 (HSS)	<code>61400000002</code>
<code>type</code>	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 地址 IP 地址	<code>100.72.83.20</code>

接口

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

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

# VoIP 是否 IMS 注册
vip_subscriber_ims_registered{type=~"voip_only|full"}

# 是否 EPC 注册
vip_subscriber_epc_registered{type=~"data_only|full"}

# IMS 注册时间 (>1 分钟)
vip_subscriber_ims_registration_age_seconds > 3600
```

图例

IMS VIP 是否健康状态 VIP 是否注册状态

图例: Grafana → IMS → IMS VIP 图例

图例 **URL:** `/d/ims-vip-subscribers/`

图例

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

如何 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. 查询 VIP 数据

```
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/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/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 地址
        - 8.8.8.8
{% else %}
        # 主机 'dns' 组 DNS 地址
{% for dns_host in groups['dns'] | default([]) %}
        - {{ hostvars[dns_host]['ansible_host'] }}
{% endfor %}
{% endif %}
        # DNS 搜索 - 搜索域
        search:
        - customer.example
      routes:
        - to: "default"
          via: "{{ gateway }}"

{% if secondary_ips is defined %}
  # 主机 - 主机名 secondary_ips 组
  # 主机名 ens19-ens20-ens21...-18 + loop.index-1
  # 主机名 "主机名"
{% for nic_name, nic_config in secondary_ips.items() %}
  ens{{ 18 + loop.index }}:
    addresses:
      # 主机 NIC 掩码 mask_cidr-1 24-1
      - "{{ nic_config.ip_address }}/{{ nic_config.mask_cidr |
default(24) }}"
{% if nic_config.routes is defined %}
      # 主机 - 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
- DNS 指定 `dns` 指定 DNS
- DNS `search` 指定 DNS 搜索域
- 指定网卡 `ens19` `ens20` ...
- 指定 IP 地址 `mask_cidr` 指定子网掩码

#####

网卡 (eth0) 配置

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

DNS 配置

- `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 NIC 00000000000000

Proxmox 000000

- `host_vm_network` - 物理 NIC の名前
- `vlanid` - 物理 VLAN の ID

手順



1. `/etc/netplan` ディレクトリに `netplan` のバックアップファイルを作成する
2. Jinja2 テンプレート `/etc/netplan/01-netcfg.yaml` を生成する
3. `netplan try --timeout 30` を実行して、設定を確認する
4. `wait_for_connection` を実行して、接続を確認する
5. `netplan apply` を実行して、設定を適用する

結果

IP の設定 (Diameter Edge Agent (DEA))

```

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

DRAs

DRAs 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 📄 - 📄📄📄📄
- 📄📄📄 - 📄📄📄📄

□□□□ Syslog

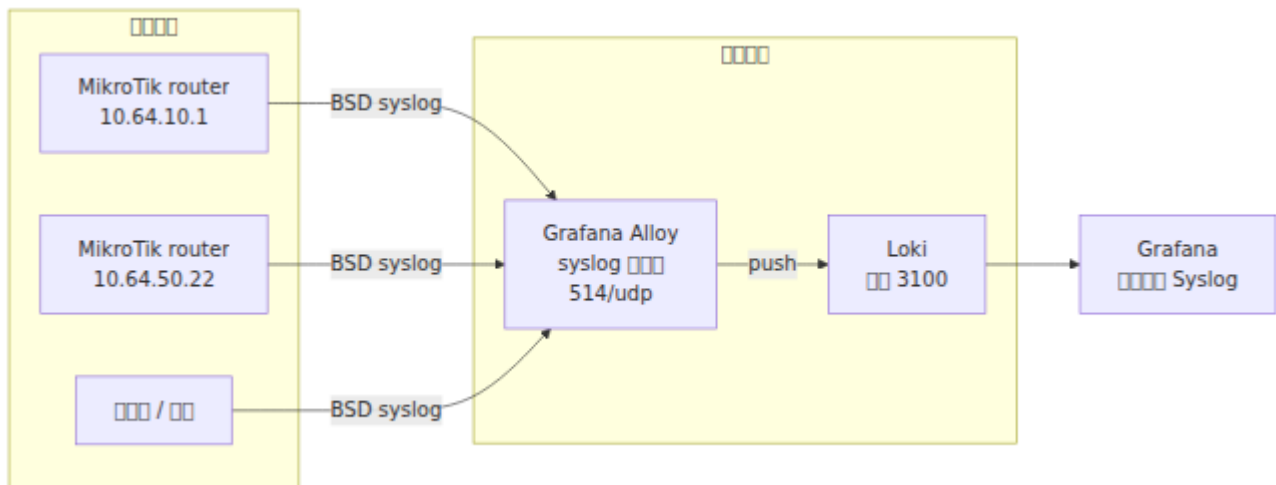
Grafana Alloy syslog Alloy syslog Loki

□ □

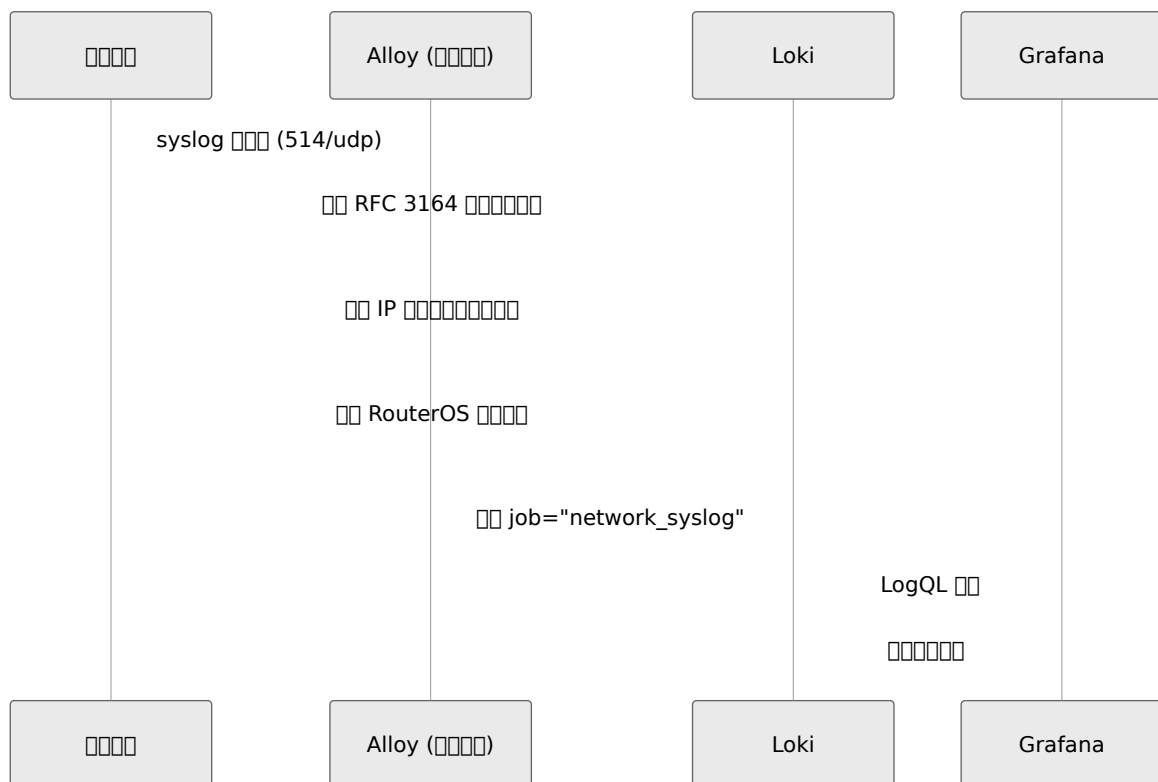
- 配置
- 配置策略路由
- 配置策略路由
- 配置策略路由
- 配置策略路由
- MikroTik RouterOS
- 配置 Syslog 配置
- 配置
- 配置
- 配置策略路由
- 配置策略路由
- 配置策略路由

11

syslog



使用 `BSD syslog` [RFC 3164](#) 格式的设备使用 `Alloy` 接收 syslog 消息，`Alloy` 接收消息后
 再推送到 `Loki`，最后由 `Grafana` 展示。



配置

`Alloy` 接收 syslog 消息的配置如下：`Alloy` 接收消息后推送到 `Loki`，最后由 `Grafana` 展示。

配置 `Alloy` 接收 syslog 消息。

- 所有设备都使用相同的配置
- 所有设备都使用相同的 `device_ip` 配置

所有设备都使用相同的配置

配置

所有设备都使用 `roles/common/defaults/main.yml` 配置 `all.vars` 配置

```
all:
  vars:
    network_syslog_port: 514          # 端口
    network_syslog_protocol: udp      # 协议
    network_syslog_vendors:          # 厂商
      - mikrotik
```

配置	设备	配置	配置
<code>network_syslog_port</code>	设备	<code>514</code>	所有设备都使用相同的配置 1024 端口
<code>network_syslog_protocol</code>	设备	<code>udp</code>	所有设备都使用相同的配置 <code>udp</code> 或 <code>tcp</code> 或 <code>UDP</code>
<code>network_syslog_vendors</code>	设备	<code>[mikrotik]</code>	所有设备都使用相同的配置 <code>devices:</code> 或 <code>syslog</code> 或 <code>syslog</code>

Alloy 配置 `alloy` 配置 1024 端口 `network_syslog_port` 配置 1024 端口 Alloy 配置 `CAP_NET_BIND_SERVICE` 配置 1024 端口

配置 `common` 配置

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

AlloySSH Loki

SNMP SNMP

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

devices			-	IP device
snmp_community			public	Prometheus SNMP syslog

syslog job="network_syslog"

項目	説明	値
job	syslog 送信先	network_syslog
component	syslog 送信先	network_device
device	送信先Alloy の名前	nf-nids-rtr-core01
device_ip	送信先IP	10.64.10.1
vendor	送信先ベンダー	mikrotik
level	Syslog レベル	informational, warning, err, crit
facility	Syslog ファシリティ	daemon
topics	RouterOS のトピック	system,info,account
topic	トピック	system, firewall, dhcp
syslog_host	syslog 送信先ホスト	system,info,account

RouterOS の syslog 送信先を network_syslog とし、送信先を network_device とし、送信先IPを 10.64.10.1 とし、送信先ベンダーを mikrotik とし、送信先レベルを informational, warning, err, crit とし、送信先ファシリティを daemon とし、送信先トピックを system,info,account とし、トピックを system, firewall, dhcp とし、syslog 送信先ホストを system,info,account とする。

MikroTik RouterOS

RouterOS の syslog 送信先を remote とし、送信先を network_device とし、送信先IPを 10.64.10.1 とし、送信先ベンダーを mikrotik とし、送信先レベルを informational, warning, err, crit とし、送信先ファシリティを daemon とし、送信先トピックを system,info,account とし、トピックを system, firewall, dhcp とし、syslog 送信先ホストを system,info,account とする。

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

項目	値	説明
target	remote	syslog 送信先
remote	syslog	syslog 送信先
remote-port	514	network_syslog_port
remote-log-format	bsd-syslog	RouterOS 7 RFC 3164 形式
src-address		

RouterOS 形式

```
system resource /system resource print
```

項目	値	説明
RouterOS 7	remote-log-format	remote-log-format=bsd-syslog default Mikrotik syslog RouterOS 6 bsd-syslog bad parameter bsd-syslog
RouterOS 6	bsd-syslog	bsd-syslog=yes

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

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

tcpdump (invalid) SYSLOG, Facility local0, Severity info

RouterOS

critical	
error	DHCP
warning	
info	account

error warning critical info

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

Syslog

BSD syslog

1. BSD syslog RFC 3164 RFC 5424

2. UDP 514 network_syslog_protocol network_syslog_port
3. devices:

network_syslog_vendors

Cisco IOS

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

Linux rsyslog

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

rsyslog RFC 3164 RSYSLG_SyslogProtocol23Format RFC 5424

Web

syslog BSD

4b 514/udp monitoring { net_devices: true } all.vars.<vendor>.devices

enforce

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

111

Grafana Logs Syslog

項目	内容
ネットワーク構成	ネットワーク構成図を添付する
接続機器	接続機器の型番、ファームウェアバージョンを添付する
設定ファイル	設定ファイルのバックアップを添付する
エラーログ	エラーログの出力方法を説明する
接続テスト	接続テストの結果を添付する
セキュリティ	セキュリティ対策の取組状況を説明する
バックアップ	バックアップの取組方法を説明する
OS/DB	OS/DBのバージョン、パッチ適用状況を説明する
OSPF/BGP	OSPF/BGPの設定を説明する
DHCP	DHCPの設定を説明する

項目	内容
Site (logs)	Loki ログ検索
Device	ログを生成したデバイス名で Loki から device フィールドで絞り込む All
Search	ログ本文の検索

配置

```
# 配置 syslogd 服务
{job="network_syslog", device="nf-nids-rtr-core01"}

# 配置 warning 级别
{job="network_syslog", level=~"emerg|alert|crit|err|warning"}

# 配置 syslogd 服务
{job="network_syslog"} |~ `(?i)(logged in|login failure|changed by)`

# 配置 RouterOS 服务
{job="network_syslog", topic="firewall"}

# 配置 syslogd 服务
sum by (device) (rate({job="network_syslog"} [5m]))

# 配置 syslogd 服务
sum by (device) (count_over_time({job="network_syslog"} [1h]))
```

部署

Loki 部署

配置 Loki 服务

配置 Loki 服务

- Alloy 配置
- 配置 Loki 服务
- 配置 Loki 服务
- 配置 Loki 服务

配置 Loki 服务

1. 配置 `systemctl status alloy` 配置 Loki 服务

2. `ss -ltnp | grep 514` Alloy 確認
3. `tcpdump -n -i any udp port 514` 確認
4. 確認

Alloy 確認

Alloy 確認 514 確認

確認

- `CAP_NET_BIND_SERVICE`
- `rsyslogd` UDP 確認

確認

1. `journalctl -u alloy -n 50` 確認
2. `systemctl cat alloy` 確認
3. `ss -ltnp | grep 514` `network_syslog_port` 確認

確認 Loki 確認

`tcpdump` 514 Loki `network_syslog`

確認

- `BSD syslog` RouterOS 7 `remote-log-format=bsd-syslog`
RouterOS 6 `bsd-syslog=yes`
- `RFC 5424`

確認

1. `tcpdump -n -i any -A udp port 514` `tcpdump` `(invalid)` `syslog` 確認
2. RouterOS 確認
3. `BSD` `syslog` 確認
4. `journalctl -u alloy | grep -i syslog` Alloy 確認

環境構築

環境構築時に `device_ip` と `device` を指定

環境構築

- 環境構築時に `devices:` を指定
- 環境構築時に `devices:` を指定

環境構築

1. `device_ip` を指定
2. 環境構築時に `devices:` を指定
3. `common` を指定

環境構築

環境構築時に `devices:` を指定

環境構築

- 環境構築時に `devices:` を指定
- 環境構築時に `devices:` を指定
- 環境構築時に `devices:` を指定

環境構築

1. `/system logging print` を指定
2. `ping` を指定
3. 環境構築時に `devices:` を指定

📄📄📄📄

📄📄	📄📄
RFC 3164	BSD syslog 📄📄
RFC 5424	Syslog 📄📄📄📄📄📄📄📄📄📄
📄📄📄📄📄📄	📄📄📄📄📄📄📄📄📄📄
📄📄📄📄📄	📄📄📄📄📄📄📄📄
📄📄📄📄📄📄	Prometheus📄Grafana 📄📄📄

OmniHSS

OmniHSS ██████████AuC██ IMS ████████ PostgreSQL ████████

PostgreSQL 2024 10월 OmniHSS 2024.10.20-2024.10.26/2024.10.27
2024.10.28-2024.10.30/2024.10.31 S6a/Cx/Dx 1 Sh 2024.11.01

OmniHSS

11

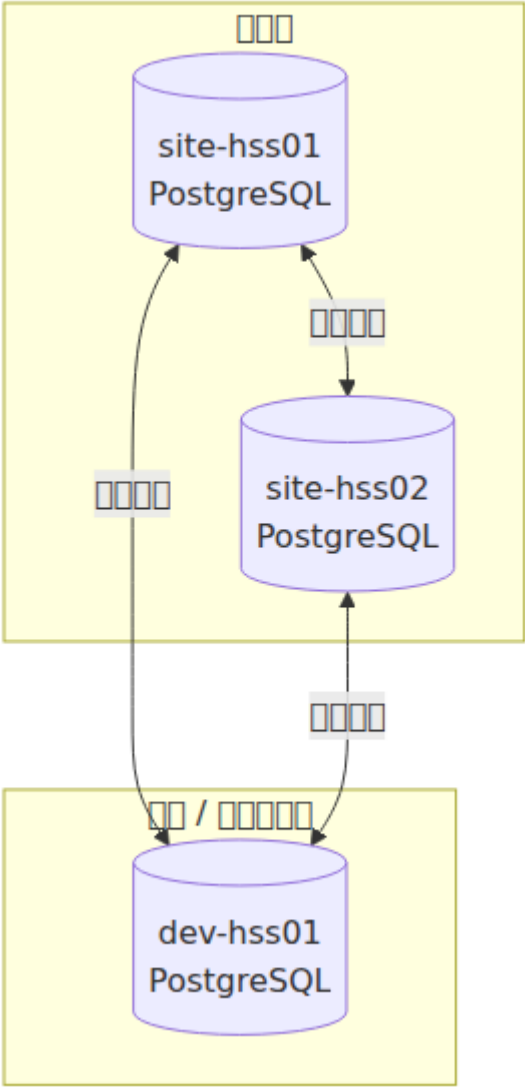
- 〇〇〇〇
- 〇〇〇〇〇〇〇〇〇〇
- 〇〇
- 〇〇〇〇 OmniHSS 〇〇
- 〇〇〇〇〇〇〇〇〇〇
- 〇〇〇〇
- 〇〇〇〇〇
- 〇〇〇〇〇〇〇
- 〇〇〇〇〇
- 〇〇
- 〇〇〇〇

□ □ □ □

```

-- OmniHSS 接続先 PostgreSQL 接続先を指定する。接続先が origin
= none 接続先が origin である場合は updated_at 列を LWW 列

```



データベースデータベースデータベースデータベース N データベースデータベース N-1 データベースデータベース N-1 データベースデータベース

数据库

表名	表结构	备注
用户信息表	subscriber, key_set (AuC), sim, msisdn	用户基本信息表
用户状态表	subscriber_state, pdn_session, lte_call, ims_subscription	用户状态信息表 updated_at
迁移表	schema_migrations	记录数据库迁移

数据库 public 数据库 schema_migrations 记录数据库迁移
记录数据库 LWW 记录数据库 updated_at 记录数据库



数据库连接

数据库 OmniHSS 数据库连接

1. 数据库 hss 数据库连接 // 数据库 OmniHSS 数据库连接 hss01 / hss02 数据库连接 connected_omnihss 数据库连接
2. 数据库 connected_omnihss 数据库 // 数据库 OmniHSS 数据库连接 hss 数据库连接

omnihss.replication.enabled 为 true 时，omnihss 才会启动。

connected_omnihss 为 true 时，omnihss 才会启动。

配置

omnihss 配置

名称	值	类型	说明
omnihss	<self>_pub	字符串	omnihss 名称
sub_from	sub_from_<source>	字符串	子源名称
subscriber_from	<subscriber>_from_<source>	字符串	订阅者名称

omnihss 使用 PostgreSQL 数据库，数据库名称为 omnihss。

omnihss --limit 配置

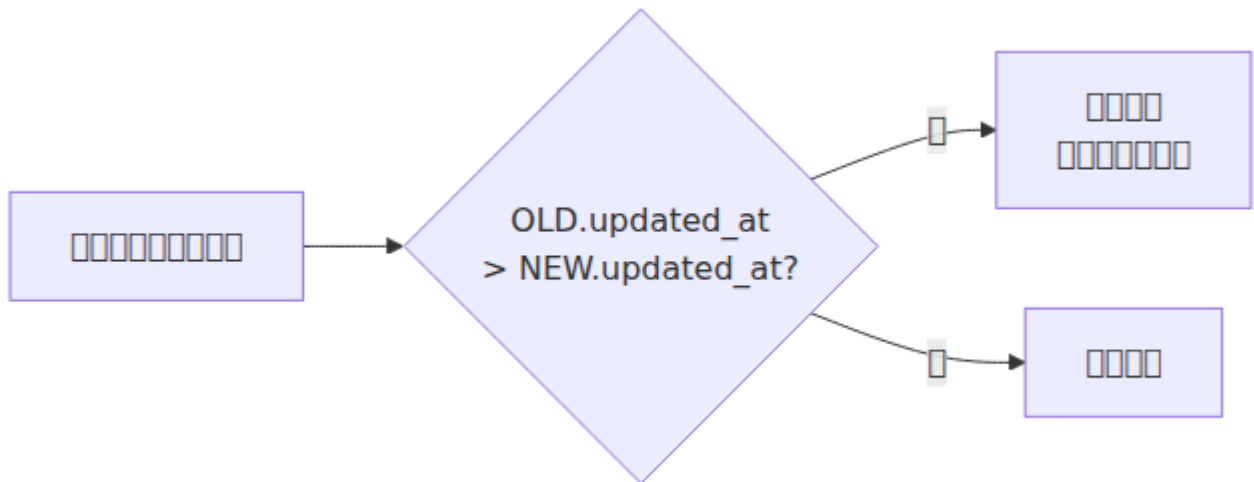
omnihss 配置 hss 和 connected_omnihss 配置

- omnihss 配置 <self>_pub 为 omnihss/omnihss
- omnihss 配置 sub_from 为 sub_from_omnihss
- omnihss 配置 subscriber_from 为 <subscriber>_from_omnihss

connected_omnihss 配置 为 hss 配置

- limit 配置 groups['hss'] 为 connected_omnihss 配置 --limit 为 hss01/hss02/hss03 配置 --limit 为 hss01/hss02/hss03 配置
- omnihss 配置 connected_omnihss 为 hss 配置

connected_omnihss
connected_omnihss



OLD

NEW

omnihss.replication.enabled false group_vars

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

項目	型	デフォルト値	説明	コメント
database_type	文字列		postgres	データベースの種類 postgres
database_host	文字列		localhost	データベースのホスト localhost
database_port	整数		5432	PostgreSQL のポート番号
database_name	文字列		omnihss	データベースの名前
database_username	文字列		hss	データベースのユーザー名 REPLICATION ユーザー
database_password	文字列		password	データベースのパスワード
replication.enabled	ブール値		false	replication.enabled true なら PostgreSQL のレプリケーションが有効

replication.enabled true なら PostgreSQL のレプリケーションが有効

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 1000

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

接続許可する (connected_omnihss)

OmniHSS 接続許可 remote_peers_file 接続許可する接続先
prod_master_peers.yml

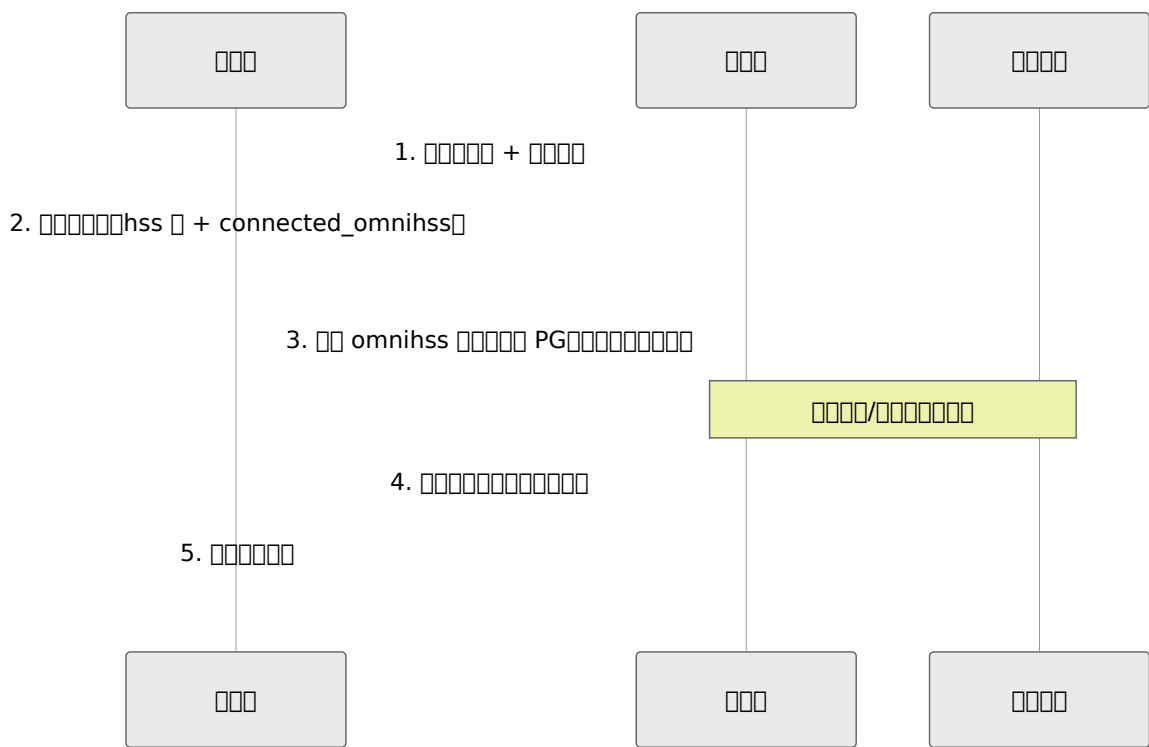
```
connected_omnihss:
- name: site-hss01
  host: 10.4.2.142
  port: 5432
- name: site-hss02
  host: 10.4.2.143
  port: 5432
- name: dev-hss01
  host: 10.4.10.142
  port: 5432
```

項目	項目名	項目値	項目単位	項目説明
name	name	-		inventory_hostname sub_from_<name> <name>_pub inventory_hostname
host	host	-		IP
port	port	5432		PostgreSQL database_port

connected_omnihss 項目は OmniHSS 項目
hss 項目

OmniHSS

OmniHSS 項目は
項目



1 10.4.2.142

10.4.2.142 OmniCore 10.4.2.142 10.4.2.142 Proxmox 10.4.2.142 10.4.2.142 PostgreSQL 10.4.2.142 5432 10.4.2.142 10.4.2.142

2 10.4.2.142

10.4.2.142 host_file 10.4.2.142 hss: 10.4.2.142

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

10.4.2.142 10.4.2.142 connected_omnihss 10.4.2.142 10.4.2.142 10.4.2.142

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

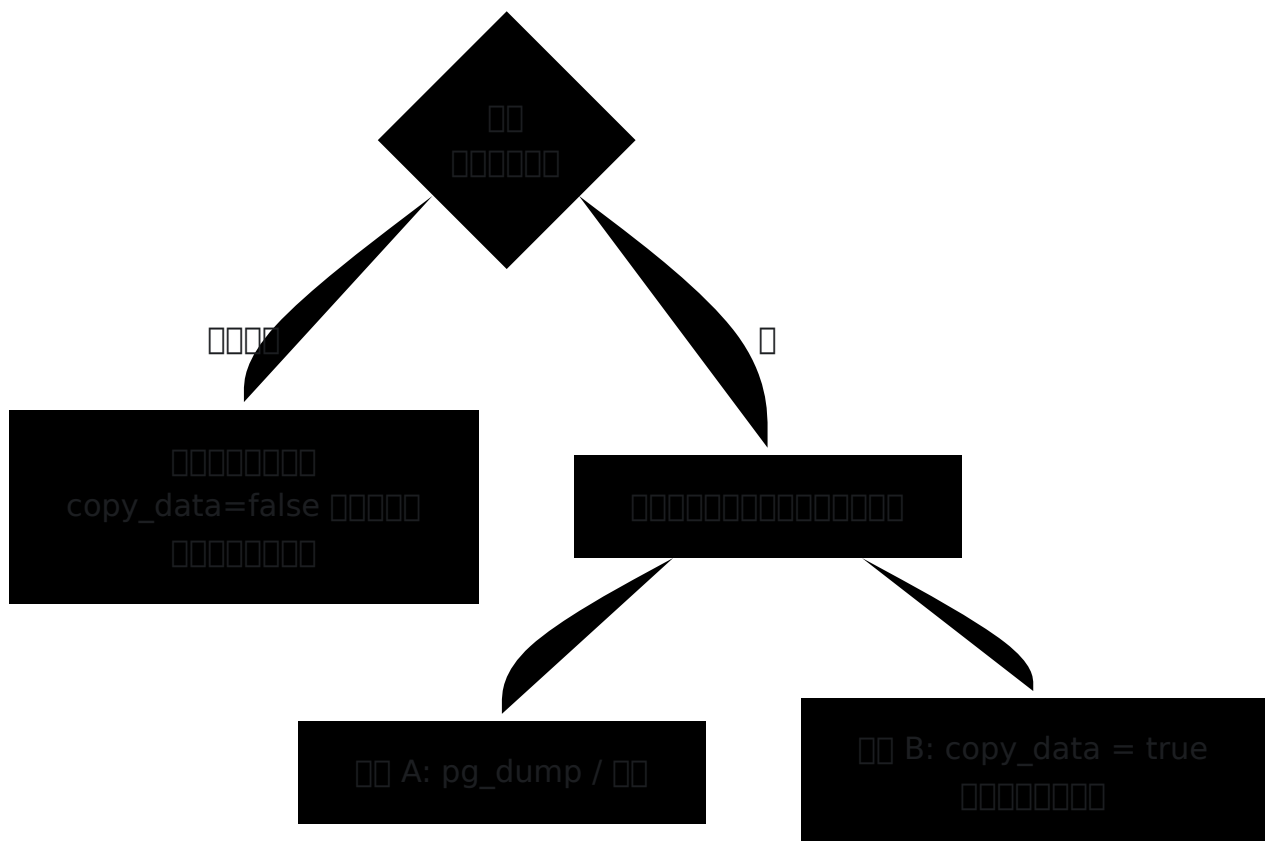
3 OmniHSS

OmniHSS PostgreSQL OmniHSS

```
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



Site

Customer OmniHSS Host

Host A

Host B

1. Host OmniHSS Host pg_dump Host --no-publications --no-subscriptions Host

```

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

Backups backups/<Site_Name>/hss_dump_<hostname>_<timestamp>.sql

2. 確認する 確認する OmniHSS 確認する確認する確認する確認する確認する確認する確認する確認する

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

-e skip_pre_restore_backup=true 確認する — 確認する確認する確認する確認する
確認する確認する確認する確認する確認する確認する確認する確認する確認する確認する

確認する 1 SQL 確認する確認する確認する確認する確認する確認する確認する

3. 確認する確認する copy_data = false 確認する確認する確認する 確認する

確認する 確認する確認する確認する確認する確認する確認する確認する確認する確認する確認する
 subscriber_state, pdn_session, lte_call, ims_subscription 確認する
確認する確認する確認する確認する確認する確認する

確認する B 確認する確認する確認する確認する

seed_hss_replication.yml 確認する 確認する確認する確認する確認する確認する確認する確認する
確認する copy_data = false 確認する確認する確認する確認する 確認する subscriber_state
 確認する 1:1 確認する確認する確認する確認する確認する確認する確認する確認する
確認する API S6a 確認する確認する確認する確認する確認する UPDATE 確認する確認する確認する
 pdn_session, lte_call 確認する COPY 確認する確認する確認する確認する確認する確認する
確認する確認する確認する

確認する確認する 確認する確認する確認する確認する確認する確認する確認する確認する

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

確認する確認する確認する

1) dev-hss01 (10.4.10.142) - 838 確認する
確認する確認する

####

```
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 ???

seed_hss_replication.yml ####
--limit #### — #### --
limit #### --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. 複製するデータベース -t 対象データベース
# schema_migrations, pdn_session, lte_call - 対象データベース
subscriber_state
# 複製するデータベース
# subscriber_state_id 列の値を API/S6a から取得する
# 複製するデータベース
# 複製するデータベース origin=none
# 複製するデータベース
# 複製するデータベース WAL 複製するデータベース
# 複製するデータベース
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

複製するデータベースを確認する

- 複製するデータベース
- 複製するデータベース
- 複製する WAL サイズ 100 MB
- 複製する WAL サイズ reserved extended lost

複製するデータベース OmniHSS から取得する

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

# 確認 - active 確認 wal_status 確認 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	確認項目
worker_alive = false	確認サブスクリプション/ワーカーの状態
active = false 確認	確認サブスクリプションのアクティブ状態
確認 lag	確認WALの遅延
wal_status = lost	確認PostgreSQLのWALステータス (黄色)

確認 health_check.yml 確認 HSS Postgres 確認 WAL

確認

確認 session_replication_role = replica 確認 FOREIGN KEY 確認 RI PostgreSQL 確認 FK

確認-確認

- 確認 A 確認 epc_profile X 確認 ON DELETE RESTRICT 確認 A 確認 X
- 確認 B 確認 203 確認 X 確認 B

準備

pg_hba.conf を編集して、接続を許可する。

接続を許可する。

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

pg_hba.conf を編集して、PostgreSQL を

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

connected_omnihss と hss を

準備

OmniHSS を PostgreSQL に

項目	設定	説明
services/backup.yml	hss_dump_<host>_<ts>.sql hss_<host>_<ts>.tar.gz /etc/omnihss backups/<Site_Name>/ /	
util_playbooks/restore_hss.yml	SQL/OmniHSS /	

準備完了

メトリック名	タイプ	単位
omnihss_subscription_worker_alive{subname}	Gauge	0=オフライン 1=オンライン
omnihss_replication_slot_active{slot_name}	Gauge	0=スロット非アクティブ 1=スロットアクティブ
omnihss_replication_slot_wal_status_code{slot_name}	Gauge	0=不明 1=エラー 2=正常 3=リプレケーションエラー
omnihss_replication_slot_lag_bytes{slot_name}	Gauge	バイト
omnihss_subscription_error_apply_error_count{subname}	Counter	エラー回数
omnihss_subscription_error_sync_error_count{subname}	Counter	エラー回数
omnihss_integrity_fk_orphans	Gauge	0=0 1=1 2=2 3=3
omnihss_schema_max_version / omnihss_schema_migration_count	Gauge	バージョン

メトリック名は、監視対象のデータベースに依存する

問題

問題の背景

データベースのバックアップと復元に関する問題

問題

- データベースのバックアップ時に `copy_data = false` の設定が有効になっている

問題

- データベースのバックアップ時に `copy_data = false` の設定が有効になっている

問題の解決

データベースのバックアップ時に `pg_stat_subscription` の設定が `pid IS NULL` の状態になっている

問題

- データベースのバックアップ時に PostgreSQL の設定が有効になっている
- データベースのバックアップ時に `pg_hba.conf` の設定が有効になっている
- データベースのバックアップ時に `database_username / database_password` の設定が有効になっている

問題

- データベースのバックアップ時に `5432` のポート番号が有効になっている
- データベースのバックアップ時に `connected_omnihss` の設定が `hss` の状態になっている `pg_hba.conf` の設定が有効になっている
- データベースのバックアップ時に `database_username / database_password` の設定が有効になっている

問題の解決

データベースのバックアップ時に `wal_status` の設定が `lost` の状態になっている

問題

- データベースのバックアップ時に WAL の設定が有効になっている

□□□□□

1. □□□□□□□□□□□□□□□□ □□□□□□□□□□

□□□□□□□□□□□□

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

□□□□□

- □□□□□□□□□□□□□□□□□□□□□□□□□□□□

□□□□□

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

□□□□□□□□□□□□

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

□□□□□

- □□□□□□□□□□□□□□□□

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

Proxmox VM/LXC 簡介

目前我們使用 Proxmox 部署 OmniCore 目前我們使用 `util_playbooks/proxmox.yml` 設定 `deployment_type` 為 QEMU 或 LXC 目前我們使用 `proxmoxServers` 設定 LXC 目前我們使用 UPF 設定

目前我們使用 VMware/HyperV 或 Vultr / AWS / GCP 部署

目前

- 目前我們使用 LXC
- IP 地址
- Netplan 設定
- 目前

LXC 與 VM

LXC 簡介

- 目前我們使用
- 目前我們使用
- 目前
- 目前我們使用
- 目前我們使用 **EPC/IMS** 設定
- 目前我們使用 **UPF** 設定/TUN 設定

目前我們使用 KVM

- 目前我們使用 ??? 設定
- 目前我們使用 / 設定
- 目前
- 目前
- **UPF** 設定

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
```

XXXXXXXXXXXXXXXXXXXX LXC XXXXXXXXXXXXXXXXXXXX

☐ 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
```

111

- `proxmox.yml` `ciuser` `cipassword` `sshkeys`
- Cloud-init `proxmoxTemplateUser` `proxmoxServers` `local_users`
- Cloud-init `proxmoxTemplatePassword` `proxmoxServers` `cloud-init`
- SSH `local_users.*.public_key`
- `local_users` `cloud-init` `proxmoxTemplateUser` `proxmoxTemplatePassword` `proxmoxServers`
- `--vga std` Proxmox Web
- `wget -N`

□□□□□ **ISO** □□□□□

[illegible]

1 Web UI

- `qemu-system-x86_64 -smp 2 -m 2048 -drive file=ubuntu-2404-template.img`
- `-cdrom ubuntu-server.iso`
- `-device virtio-scsi-pci`
- `-device scsi-hd,drive=sda`
- `-cpu qemu64`
- `-mem-path /dev/shm`
- `-device virtio-blk-pci,drive=vdb`
- `-device vhost-user-fs-pci`
- `-device vhost-user-fs-pci`

2. 2019 - 2020

```
# [] 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           # CPU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 → UPF → UPF

1111

- `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...

Ansible Playbook

proxmox_server: proxmoxServers
proxmoxServers proxmoxServers
proxmoxServers

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

proxmox_server vars: proxmoxServers

Ansible Playbook

deployment_type proxmox_server proxmoxServers LXC
proxmoxLxcCores proxmoxLxcMemoryMb proxmoxLxcDiskSizeGb
proxmoxLxcRootFsStorageName proxmoxLxcOsTemplate host_vm_network
proxmoxServers

```
dns:
  hosts:
    site-dns01:
      ansible_host: 192.168.1.20
      deployment_type: vm # proxmoxServers
      proxmoxLxcDiskSizeGb: 120 # proxmoxServers
      host_vm_network: vmbr1 # proxmoxServers
  vars:
    proxmox_interface: vmbr0
    gateway: 192.168.1.1
    mask_cidr: 24 # proxmoxServers 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
│   └─ omnisep.yml
├─ omniepdg.yml
├─ omnitwag.yml
├─ omniss7.yml
├─ ocs.yml
├─ crm.yml
├─ ran_monitor.yml
├─ 5gc.yml
└─ ../util_playbooks/health_check.yml

```

ansible-playbook omnimme.yml --limit 50 --limit all.yml

目录

目录

文件	目录	内容
all.yml	目录	包含所有组件的清单，包括DNS、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/Entitlements/BSF)
ocs.yml	目录	CGrateS、KeyDB、OCS、
monitoring.yml	目录	Prometheus、Grafana、exporters

📁📁📁📁📁

📁	📁
<code>common.yml</code>	📁📁📁📁📁📁📁📁NTP📁📁📁📁
<code>setup_users.yml</code>	📁📁📁📁📁 SSH 📁📁
<code>dns.yml</code>	DNS 📁📁📁📁
<code>license_server.yml</code>	OmniCore 📁📁📁📁📁
<code>does_netplan_exist.yml</code>	📁📁📁📁📁📁📁📁 netplan
<code>update_mtu.yml</code>	📁📁 netplan 📁📁📁📁📁 MTU
<code>firewall.yml</code>	iptables/nftables 📁📁
<code>apt_cache.yml</code>	📁📁 APT 📁📁📁📁📁📁

EPC 配置

配置ファイル	機能	説明
<code>omnimme.yml</code>	OmniMME	4G/LTE ネットワーク (4G)
<code>omnisgwc.yml</code>	OmniSGW-C	4G/LTE ネットワーク
<code>omnipgwc.yml</code>	OmniPGW-C	PDN ネットワーク
<code>upf.yml</code> / <code>omniupf.yml</code>	OmniUPF	4G/LTE ネットワーク SGW-U/PGW-U
<code>omnihss.yml</code> / <code>hss.yml</code>	OmniHSS	4G/LTE ネットワーク
<code>omnidra.yml</code>	OmniDRA	Diameter ネットワーク
<code>omnitwag.yml</code>	OmniTWAG	4G/LTE ネットワーク
<code>omniepdg.yml</code>	OmniEPDG	4G/LTE ネットワーク WiFi ネットワーク
<code>gtp_proxy.yml</code>	GTP ネットワーク	GTP ネットワーク

IMS 配置

配置文件	网元	说明
pcscf.yml	P-CSCF	公网P-CSCF
icscf.yml	I-CSCF	公网I-CSCF
scscf.yml	S-CSCF	公网S-CSCF
as.yml	OmniTAS	公网TAS
omnisep.yml	OmniSEP	XCAP、XCAP-BSF、XCAP-BSF
xcap.yml	OmniSEP	omnisep.yml 配置文件 omnisep 配置
omnimessage.yml	OmniMessage / IMS SMSC	公网 IP 地址、SMPP 与 IMS SMSC cscf 配置 cscf_type: smsc_ims

5G 配置

5gc.yml 配置文件 5gc_nf 配置 NF 包 nf_package 配置
NF 配置 UDM HNET/SUCI 配置 5G Core

名前	機能	説明
5gc.yml	5G ネットワーク	5G ネットワークの構成
	OmniNRF	ネットワーク機能
	OmniAMF	ネットワーク機能
	OmniAUSF	ネットワーク機能
	OmniBSF	ネットワーク機能
	OmniCHF	ネットワーク機能
	OmniNSSF	ネットワーク機能
	OmniPCF	ネットワーク機能
	OmniSCP	ネットワーク機能
	OmniSMF	ネットワーク機能
	OmniUDM	ネットワーク機能
	OmniUDR	ネットワーク機能

ネットワーク

cs.yml circuit_switched/ BSC MSC SS7 Circuit-Switched Core

파일	기능	설명
<code>cs.yml</code>	CS	CS (Circuit Switched) 모듈
	BSC	BSC (Base Station Controller) 모듈 (<code>circuit_switched/bsc</code>)
	OmniMSC	OmniMSC (Omni Mobile Switching Center) 모듈 (<code>circuit_switched/omnimsc</code>)
	SS7	SS7/SIGTRAN 모듈 (<code>omniss7</code> 모듈)

모듈 목록

파일	기능
<code>ocs.yml</code>	OCSS (Open Core Switching) (CGrateS + KeyDB)
<code>crm.yml</code>	CRM (Customer Relationship Management) (OmniCRM 모듈 포함 <code>group_vars/</code>)
<code>omniss7.yml</code>	SS7/SIGTRAN 모듈
<code>voipmonitor.yml</code>	SIP/RTP 모니터링 (VoIPmonitor) 모듈 <code>siptrace</code> 모듈 포함
<code>grafana.yml</code>	Grafana 모듈
<code>ran_monitor.yml</code>	RAN (Radio Access Network) 모듈
<code>omniroam.yml</code>	OmniRoam 모듈 (TAP3/RAP UI, Elixir API, generator)
<code>omnilcs.yml</code>	OmniLCS (Omni Location Services) 모듈
<code>device_tester.yml</code>	OmniWeb 모듈 (모듈)

Proxmox 모듈 (`proxmox.yml`, `proxmox_delete.yml`) 모듈 `util_playbooks/` 모듈
모듈 **Utility Playbooks** 모듈 **Proxmox VM/LXC** 모듈

00

00	00
backup.yml	0000000000
reboot.yml	0000000
shutdown.yml	00000
apt_update.yml	000000
apt_refresh_metadata.yml	00 APT 000000
speedtest.yml	000000000

00000

00	00
restore_applicationserver.yml	000000 OmniTAS
restore_smsc.yml	000000 SMSCOmniMessage 000000SMPP 0000 IMS SMS Sc 00

```
000 health_check.yml 0 restore_hss.yml 0 restore_ocs.yml 0 0
services/ 0000000 util_playbooks/ 000000 Utility Playbooks 0 all.yml 000
000000 ../util_playbooks/health_check.yml 0
```

□□

□□□□□□

```
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
```

コマンド

```
# 全ホストに all.yml を実行
ansible-playbook -i hosts/customer/host_files/production.yml
services/all.yml --limit mme01

# mme01 と hss01 の両方に all.yml を実行
ansible-playbook -i hosts/customer/host_files/production.yml
services/all.yml --limit "mme01,hss01"

# ホスト名を指定して all.yml を実行
ansible-playbook -i hosts/customer/host_files/production.yml
services/all.yml --limit mme
```

構成

- **Ansible** - 自動化ツール
- **Ansible Tower** - Ansible の管理・監視ツール
- **Ansible Galaxy** - Ansible のモジュール・ロールの共有プラットフォーム
- **Ansible** - Ansible のインストール・アップグレード
- **Ansible** - Grafana・Prometheus と連携



OmniCore `util_playbooks/`

目次

ファイル名	説明
proxmox.yml	Proxmox クラウド/ LXC のデプロイメント 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 のデプロイメント
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 のシードレプリケーション
audit_hss_drift.yml	OmniHSS の監査ドリフト + レポート
reconcile_hss_drift.yml	OmniHSS のドリフトを修正 ???
restore_ocs.yml	ocs KeyDB/MySQL のバックアップ
ip_plan_generator.yml	Mermaid の IP プラン
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

実行方法

実行コマンド: `--limit` 指定する

実行環境: Proxmox VM

Proxmox

実行コマンド: `util_playbooks/proxmox.yml, util_playbooks/proxmox_delete.yml`

`proxmox.yml` は LXC のデプロイメントタイプを指定する。デフォルトは `deployment_type` であり、`proxmoxServers` に指定された VM は Proxmox VM として実行される。

```
# LXC のデプロイメントタイプを指定する
ansible-playbook -i hosts/customer/host_files/production.yml
util_playbooks/proxmox.yml

# 実行
ansible-playbook -i hosts/customer/host_files/production.yml
util_playbooks/proxmox.yml --limit mme

# LXC のデプロイメントタイプを指定する
ansible-playbook -i hosts/customer/host_files/production.yml
util_playbooks/proxmox_delete.yml --limit old-host
```

Proxmox の再起動

実行コマンド: `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

```

- include: util_playbooks/vm/vultr/vmProvisionVultr.yml,
- include: util_playbooks/vm/vultr/vmTeardownVultr.yml,
- include: util_playbooks/vm/vultr/vmCommonSetup.yml

```

1. Vultr API 2. Vultr 3. vmProvisionVultr.yml 4. vmTeardownVultr.yml 5. vmCommonSetup.yml 6. DNS 7. VULTR_API_TOKEN 8. API 9. cloud-config.yml 10. util_playbooks/vm/vultr/

```

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

```

OmniCore

```

- include: util_playbooks/health_check.yml

```

OmniCore 1. OmniCall 2. HTML

```

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

```

例: /tmp/health_check_YYYY-MM-DD HH:MM:SS.html

項目名

項目	説明
項目名	項目名
OmniHSS	項目名Diameter 項目名
OmniDRA	Diameter 項目名
OmniTAS	項目名CPU 項目名
OCS	KeyDB 項目名

HSS 項目

例: util_playbooks/restore_hss.yml

項目名 OmniHSS項目名

```
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_<hostname>_<timestamp>.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_<hostname>_<timestamp>.tar.gz</code>	<code>/etc/omnihss</code> にバックアップする

環境構築完了後 HSS を再起動/インストール

確認

項目	確認	確認
HSS SQL をバックアップ	☐	HSS SQL をバックアップするかどうかを確認
HSS を tar.gz にバックアップ	☐	HSS を tar.gz にバックアップするかどうかを確認
確認	☐	<code>yes</code> を入力して確認

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かどうかを確認 `copy_data = false`
インストール時に `copy_data = true` を指定すると PostgreSQL をインストールする
インストールする

インストール完了

確認コマンド: `seed_source_*` を実行してインストールが完了しているかどうかを確認
インストールが完了しているかどうかを確認

```
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"
```

##

##	##	##	##
seed_source_name	##	####	##### #### [sub_from_<name>]## #####<name>_pub## #####
seed_source_host	##	####	##### IP ##### PostgreSQL ##### #####
seed_source_port	##	#### omnihss.database_port (5432)	##### PostgreSQL ## ##

| seed_source_name [seed_source_host #####

11

1. 创建数据库和表
2. 配置: 在 `hss` 表 + `connected_omnihss` 表中添加 `yes` 列
3. 设置 IP 地址 `pg_hba.conf`
4. 创建表 `WITH (copy_data = false)` 并指定 `slot_name` 为 `<target>_from_<source>`
5. 使用 `pg_dump --data-only --inserts --on-conflict-do-nothing --disable-triggers` 导出 `subscriber_state` 表并指定 `FK` 为 `pdn_session` 和 `lte_call`
6. 创建数据库和表

00000000: 000000000000000000000000 COPY 0000000000000000000000000000000000
 000000000000000000000000 subscriber_state 00000000000000000000 1:10000000
 0000 --on-conflict-do-nothing 0000000000000000000000000000000000000000
 000000000000000000000000 ???

`--limit`

HSS ☐ ☐ ☐ ☐

```

- ansible-playbook: util_playbooks/audit_hss_drift.yml
- ansible-playbook: roles/omnihss/tasks/audit_drift.yml

```

`copy_data = false`

[illegible]

```

$ cat /etc/hss/omnihss/omnihss.yaml
services:
  all:
    - services/all.yml
  free:
    - services/deploy_nfs.yml

```

項目	設定	説明
omnihss_audit_drift	true	ドリフト検出時の挙動
omnihss_audit_fail_on_drift	true	ドリフト検出時に失敗するかどうか (false なら 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
```

- `mysql -u root -p --insert --host 127.0.0.1 --port 3306 --database test --table test --data test.txt`
- `mysql -u root -p --insert --host 127.0.0.1 --port 3306 --database test --table test --data test.txt --dry-run -e assume yes=true`

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

実行 `audit_hss_drift.yaml` を実行して、`OmniHSS` の状態を確認する

OCS 実行

実行: `util_playbooks/restore_ocs.yaml`

実行 OCS (CGrates) の実行環境を設定する KeyDB の実行

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

実行

1. OCS の実行 CGrates の KeyDB
2. AOF の実行環境を設定する
3. KeyDB RDB の実行環境を設定する KeyDB の実行
4. MySQL StoreDB の実行
5. `/etc/cgrates` の実行

6. 部署 CGrateS

部署

項目	ファイル名	説明
KeyDB 保存先	<code>keydb_dump_<hostname>_<timestamp>.rdb</code>	KeyDB RDB 保存先
MySQL StoreDB	<code>cgrates_dump_<hostname>_<timestamp>.sql</code>	<code>cgrates</code> の MySQL 保存先
tar.gz 保存先	<code>cgrates_<hostname>_<timestamp>.tar.gz</code>	<code>/etc/cgrates</code> 保存先

設定

項目	設定	説明
KeyDB RDB 保存先	<code>keydb_rdb</code>	KeyDB RDB 保存先
MySQL SQL 保存先	<code>mysql_sql</code>	CGrateS SQL 保存先 MySQL StoreDB
tar.gz 保存先	<code>tar.gz</code>	tar.gz 保存先

IP 計画

例: `util_playbooks/ip_plan_generator.yml`

ネットワーク計画

- IP 計画 NIC 計画
- ネットワーク
- ネットワーク Diameter GTP PFCP SIP SS7

```
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	

```
util_playbooks/get_ports.yml
```

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

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

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 <inventory></code>	ホストのリスト
<code>--limit <hosts></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>	変更内容の差分を表示

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
```

