5G Core (5GC)

يتم نشر النواة المستقلة 5G (SA) بواسطة `services/5gc.yml` playbook. يقوم دور مشترك `5gc_nf`، واحد لكل 5 NFs. تشرح هذه الصفحة هذا الدور، والتكوين الذي يقرأه كل واحد منها، و التي يبنيها.

نظرة عامة

دور واحد (`5gc_nf`) لجميع الـ NFs في 5GC. يستدعي playbook الدور مرة واحدة لكل NF ويمرر متغير `nf_package` الذي يحدد حزمة Debian، خدمة systemd، وقالب تكوين Jinja2. هذا يحافظ على تثبيت، إصدار التكوين، الترخيص، ومنطق إعادة التشغيل متطابقًا عبر كل NF.

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

وظائف الشبكة

كل NF له مجموعة جرد، حزمة، خدمة systemd، وقالب تكوين خاص به (`roles/5gc_nf/templates/<nf_package>.j2`).

NF	`nf_package`	مجموعة الجرد	قالب التكوين	الدور
وظيفة مستودع الشبكة	`omninrf`	`nrf`	`omninrf.j2`	اكتشاف الخدمة / تسجيل NF
إدارة الوصول والتنقل	`omniamf`	`amf`	`omniamf.j2`	إشارات N1/N2، التسجيل، إدارة الاتصال
خادم المصادقة	`omniausf`	`ausf`	`omniausf.j2`	مصادقة 5G-AKA / EAP-AKA'
وظيفة دعم الربط	`omnibsf`	`bsf`	`omnibsf.j2`	ربط PCF لجلسة PDU
وظيفة الشحن	`omnichf`	`chf`	`omnichf.j2`	شحن متقارب عبر الإنترنت / غير متصل
اختيار شريحة الشبكة	`omnissf`	`nssf`	`omnissf.j2`	اختيار NSSAI / الشريحة
وظيفة التحكم في السياسة	`omnipcf`	`pcf`	`omnipcf.j2`	قواعد السياسة وجودة الخدمة
وكيل الاتصال بالخدمة	`omniscp`	`scp`	`omniscp.j2`	توجيه رسائل SBI / الاتصالات غير المباشرة
وظيفة إدارة الجلسات	`omnismf`	`smf`	`omnismf.j2`	جلسة PDU والتحكم في UPF (N4)
إدارة البيانات الموحدة	`omniudm`	`udm`	`omniudm.j2`	بيانات الاشتراك، فك تشفير SUCI

NF	`nf_package`	مجموعة الجرد	قالب التكوين	الدور
مستودع البيانات الموحد	`omniudr`	`udr`	`omniudr.j2`	تخزين بيانات داعم لـ UDM/PCF

5 كيفية عمل دور `5gc_nf`

لكل NF، يقوم الدور بنفس التسلسل:

1. **حل `group_vars_omnicore`**: يتم تعيينه إلى `<inventory>/group_vars/`، بحيث يتم حل عمليات البحث عن التكوين بالنسبة إلى الجرد النشط.
2. **اكتشاف خوادم الترخيص**: إذا لم يتم تعيين `license_server_api_urls`، يتم اشتقاقه من مجموعة جرد `license_server` (`https://<host>:8443/api`).
3. **تثبيت المتطلبات الأساسية**: `lsb-release`، `curl`، `gpg`، `unzip`، `libglu1-mesa`، `libsctp1`.
4. **تثبيت حزمة NF** من مستودع Omnitouch APT. الأحدث بشكل افتراضي؛ قم بتثبيته مع `nf_version` (انظر تثبيت الإصدار).
5. **تكوين القالب (المُعَلم)**: يقوم بتجسيد `templates/<nf_package>.j2` إلى `/etc/<nf_package>/runtime-<deb_version>.exs`.
6. **إنشاء رابط رمزي لـ `runtime.exs`** → تكوين الإصدار النشط، ونسخ ذلك الرابط الرمزي إلى كل دليل `/opt/<nf_package>/releases/*`.
7. **إعادة تشغيل NF** (المعالج) كلما تغير التكوين النشط.

تثبيت الإصدار والعودة

يتم كتابة التكوين لكل إصدار Debian (`runtime-<deb_version>.exs`) و`runtime.exs` هو رابط رمزي للإصدار النشط. يتم الاحتفاظ بالتكوينات السابقة على القرص، لذا فإن العودة هي مجرد إعادة تثبيت إصدار الحزمة الأقدم:

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

تجاهل `nf_version` لتتبع أحدث حزمة.

مفاتيح HNET لـ UDM (فك تشفير SUCI)

يمتلك UDM (`nf_package: omniudm`) خطوات إضافية لدعم فك تشفير SUCI (TS 33.501 §6.12). يستعيد فك التشفير SUPI من SUCI المخفية التي يقدمها UE عند التسجيل. يحتاج إلى مفاتيح الشبكة المنزلية **السرية** التي تتزاوج مع المفاتيح العامة للشبكة المنزلية على USIMs.

- تعيش المفاتيح في `group_vars/hnet_udm/` على وحدة التحكم كـ Ansible `<curve>-<id>.key` (PEM)، حيث `<curve>` هو `curve25519` (Profile A، X25519) أو `secp256r1` (Profile B، prime256v1)، و`<id>` هو معرف المفتاح العام للشبكة المنزلية.
- افتراضيًا بحيث يعمل فك `curve25519-1.key` إذا كان الدليل **فارغًا**، يتم إنشاء التشفير من الصندوق. هذه الخطوة غير متغيرة. لا يتم الكتابة فوق مفتاح موجود.
- لكل `curve25519-*.key`، يتم تصدير القيمة العامة الخام المطابقة 32 بايت (hex) إلى ملف `.pub` لتوفير USIM. تبقى ملفات `.pub` فقط في الجرد؛ يتم `*.key` نشرها إلى `/etc/omniudm/hnet/` على UDM (الجذر، `0600` الوضع).

> المفتاح الذي تم إنشاؤه تلقائيًا عشوائي ويقوم فقط بفك تشفير SUCIs من USIMs المبرمجة بمفتاحها العام `.pub` على المنزلية الشبكة المصدرة وقدمها كمفتاح عام **المطابق**. خذ ويمكنها ترك HNET التي تستخدم المخطط الفارغ إلى مفتاح SIM. لا تحتاج بطاقات USIM الدليل فارغًا.

تفاصيل كاملة وأمثلة تخطيط الملفات: المثال 8 تكوين المتغيرات الجماعية.

الوثائق ذات الصلة

- Service Playbooks - جميع playbooks الخدمة وهيكل النشر
- مجال - النواة الدائرية CS القديم (BSC، MSC، SS7)
- قوالب تكوين لكل - تكوين المتغيرات الجماعية NF ومفاتيح HNET
- كيف تتناسب المجالات معًا - معمارية النشر
- **تكوين OmniCore**: https://docs.omnitouch.com.au/docs/repos/OmniCore - ما يتم تضمينه في ملفات تكوين NF

معيار تخطيط IP لـ OmniCore

نظرة عامة

توضح هذه الوثيقة نهج تخطيط IP القياسي لنشر OmniCore. تتطلب البنية المعمارية **أربعة** **شبكات فرعية داخلية** لتقسيم وظائف الشبكة بشكل صحيح من أجل الأمان والأداء ووضوح العمليات.

متطلبات تخصيص IP

التخصيص القياسي: أربع شبكات فرعية /24

يتطلب كل نشر لـ OmniCore أربع شبكات فرعية متميزة للتواصل الداخلي:

1. **شبكة النواة الحزمية** - أول /24
2. **شبكة الإشارة** - ثاني /24
3. **شبكة IMS الداخلية** - ثالث /24
4. **شبكة IMS العامة** - رابع /24

مهم: هذه توصيات، وليست متطلبات

تخصيص الشبكات الفرعية الموضح في هذه الوثيقة هو **أفضل ممارسة موصى بها** لتنظيم نشرات OmniCore. ومع ذلك، فإن البنية المعمارية **مرنة تمامًا**:

- **جميع المضيفين في شبكة فرعية واحدة**: يمكنك وضع جميع المكونات في شبكة فرعية واحدة إذا كان ذلك يناسب احتياجات نشراتك
- **كل نوع مضيف في شبكته الفرعية الخاصة**: يمكنك إنشاء شبكات فرعية منفصلة لكل نوع من المكونات (واحدة لـ MMEs، واحدة لـ HSS، إلخ)
- **تجميعات مخصصة**: يمكنك تنظيم المضيفين في أي هيكل شبكة فرعية يتناسب مع متطلباتك المحددة

- **مزج IPs الداخلية والعامة**: يمكن لبعض المضيفين استخدام عناوين داخلية (RFC 1918) بينما يستخدم الآخرون IPs عامة، كل ذلك ضمن نفس النشر

يوفر نهج الشبكات الفرعية الأربعة الموصى به **عزل أمني، إدارة حركة المرور، ووضوح عمليات** IP مثالي، ولهذا السبب نقترحه للنشرات الإنتاجية. ومع ذلك، يجب عليك تكييف خطة لتناسب طوبولوجيا الشبكة الخاصة بك، ومساحة العناوين المتاحة، ومتطلبات التشغيل.

تحليل تقسيم الشبكة

1. شبكة النواة الحزمية (أول /24)

الغرض: عناصر خطة المستخدم وخطة التحكم الأساسية

المكونات:

- OmniMME (كيان إدارة التنقل)
- OmniSGW (البوابة الخدمية)
- OmniPGW-C (خطة التحكم في بوابة PDN)
- OmniUPF/PGW-U (وظيفة خطة المستخدم / بوابة PDN)

مثال: `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. شبكة الإشارة (ثاني /24)

الغرض: إشارات قطرية، سياسة، وظائف شحن وإدارة

المكونات:

- OmniHSS (خادم المشتركين المنزلي)

- OmniCharge OCS (نظام الشحن عبر الإنترنت)
- OmniHSS PCRF (وظيفة قواعد السياسة والشحن)
- OmniDRA DRA (وكيل توجيه قطرية)
- خوادم DNS (DNS داخلي/بنية تحتية - متميز عن DNS الموجه للعملاء في قسم IMS العامة؛ كلاهما مقصود)
- خوادم TAP3/CDR
- المراقبة/OAM
- التقاط SIP
- خادم الترخيص
- مراقب RAN
- رابط تحذير Omnitouch CBC (مركز بث الخلايا) - إذا تم نشره

مثال: `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. شبكة IMS (24/ ثالث) الداخلية

الغرض: إشارات وخدمات IMS الأساسية (إشارات SIP الداخلية)

المكونات:

- OmniCSCF S-CSCF (وظيفة التحكم في جلسة المكالمات الخدمية)
- OmniCSCF I-CSCF (وظيفة التحكم في جلسة المكالمات الاستجوابية)
- OmniTAS (خادم تطبيقات الهاتف / خادم التطبيقات)
- OmniMessage (وحدة التحكم في SMS، SMPP، IMS)

مثال: `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. شبكة IMS العامة (رابع /24)

الغرض: خدمات IMS الموجهة للمستخدم، DNS والتوصيل للعملاء، SS7/القديمة الموجه

المكونات:

- OmniCSCF P-CSCF (وظيفة التحكم في جلسة المكالمات الوكيلة)
- خوادم XCAP
- خوادم البريد الصوتي المرئي
- DNS العملاء
- OmniSS7 STP (نقطة نقل إشارات SS7)
- OmniSS7 HLR (سجل الموقع المنزلي) - 2 لـ 3G/2G
- OmniSS7 IP-SM-GW (MAP SMSc)
- OmniSS7 بوابة CAMEL
- MSC (مركز التحويل المحمول)

> **ملاحظة:** عادةً ما توجد وظائف SS7/HLR، والتوصيل القديمة ذات الصلة في القسم العام. حيث تتصل بشبكات 2G/3G الخارجية عبر SIGTRAN/M3UA بدلاً من SIP IMS الداخلية.

مثال: `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"
```

طرق التنفيذ

يدعم OmniCore طريقتين رئيسيتين لتنفيذ هذا التقسيم الشبكي:

الطريقة 1: واجهات الشبكة الفيزيائية/الافتراضية (موصى بها للإنتاج)

استخدم NICs منفصلة أو جسور افتراضية لكل قسم شبكة. يوفر هذا أقوى عزل وهو النهج الموصى به للنشرات الإنتاجية. القاعدة هي تسمية كل جسر بعد قسمه (`v400-omni-packet-core`، `v401-omni-signalling`، `v402-omni-ims-internal`، `v403-omni-ims-public`)، صالح أيضًا حيث يتم التعامل مع (مثل `vmbr0`) على الرغم من أن جسر مشترك واحد التقسيم في مكان آخر.

مثال:

```yaml
# النواة الحزمية - v400-omni-packet-core
mme:
  hosts:
    omni-site-mme01:
      ansible_host: 10.179.1.15
      gateway: 10.179.1.1
      host_vm_network: "v400-omni-packet-core"

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

# IMS الداخلية - v402-omni-ims-internal
icscf:
  hosts:
    omni-site-icscf01:
      ansible_host: 10.179.3.55
      gateway: 10.179.3.1
      host_vm_network: "v402-omni-ims-internal"

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

الطريقة 2: تقسيم قائم على VLAN

استخدم واجهة فيزيائية واحدة مع وسم VLAN لفصل الشبكات. هذا مناسب للنشرات الأصغر أو عندما تكون NICs الفيزيائية محدودة.

مثال:

```
# كل قسم يستخدم وسم VLAN ؛ جميع المكونات تشترك في جسر واحد (ovsbr1)
مختلف.
# applicationserver (TAS) هو مكون IMS داخلي -> VLAN 402 / 10.178.3.x
applicationserver:
  hosts:
    omni-site-sbc01:
      ansible_host: 10.178.3.213
      gateway: 10.178.3.1
      host_vm_network: "ovsbr1"
      vlanid: "402"

# dra و dns هما مكونات الإشارة -> VLAN 401 / 10.178.2.x
dra:
  hosts:
    omni-site-dra01:
      ansible_host: 10.178.2.211
      gateway: 10.178.2.1
      host_vm_network: "ovsbr1"
      vlanid: "401"

dns:
  hosts:
    omni-site-dns01:
      ansible_host: 10.178.2.178
      gateway: 10.178.2.1
      host_vm_network: "ovsbr1"
      vlanid: "401"
```

تكوين الشبكة:

- قم بتكوين VLANs على المفتاح الفيزيائي
- وسم حركة المرور بشكل مناسب على مستوى المحاكي
- قم بتوجيه حركة المرور بين VLANs عند البوابة/جدار الحماية

مثال على تخطيط VLAN:

```
VLAN 400: 10.x.1.0/24 (النواة الحزمية)
VLAN 401: 10.x.2.0/24 (الإشارة)
VLAN 402: 10.x.3.0/24 (IMS الداخلية)
VLAN 403: 10.x.4.0/24 (IMS العامة)
```

العامة IP العمل مع عناوين

نظرة عامة

تتطلب العديد من نشرات OmniCore أن تحتوي بعض المكونات على عناوين IP عامة للاتصال الخارجي، مثل:

- **DRA** - ل��شارات قطرية التجوال مع شركات النقل الخارجية
- **SGW/PGW** **للتجوال** - لحركة مرور GTP من الشركاء المتجولين
- **ePDG** - لمكالمات WiFi (نفق IPsec من UEs)
- **بوابة SMSC** - لاتصالات SMPP مع مجمعات SMS الخارجية
- **P-CSCF** (في بعض النشرات) - للتسجيل المباشر لـ UE SIP

كيفية تخصيص IPs العامة

تُعالج IPs **الداخلية** في ملفات جرد **IPs** العامة **بنفس الطريقة تمامًا كما يتم التعامل مع** العام في حقل IP المضيفين لديك. ببساطة حدد عنوان `ansible_host` مع البوابة المناسبة وقناع الشبكة.

مثال: IPs العامة للتجوال مع SGW/PGW

```yaml
sgw:
  hosts:
    # SGWs الداخلية على الشبكة الخاصة
    customer-site-sgw01:
      ansible_host: 10.4.1.25
      gateway: 10.4.1.1
      host_vm_network: "v400-omni-packet-core"

    # SGWs للتجوال مع IPs العامة
    customer-site-roaming-sgw01:
      ansible_host: 203.0.113.10
      gateway: 203.0.113.9
      netmask: 255.255.255.248       # /29 subnet
      host_vm_network: "498-public-servers"
      in_pool: False
      cdrs_enabled: True

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

مثال: DRA مع IP عام

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

مثال: ePDG مع IP عام

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

مزج IPs الداخلية والعامة

من الشائع وجود مزيج من IPs الداخلية والعامة ضمن نفس مجموعة المكونات. على سبيل المثال:

- GTP الداخلية للمواقع المحلية باستخدام SGWs
- العامة خصيصًا لحركة مرور التجوال من شركات النقل الخارجية SGWs
- الداخلية والخارجية SGWs كل من PGW-C يمكن أن تدير نفس

تتعامل بنية OmniCore مع هذا ب��لاسة - فقط قم بتكوين كل مضيف مع عناوين IP المناسبة له.

مقدمة في نشر Ansible في Omnitouch

نظرة عامة

كمنصة لأتمتة البنية التحتية لنشر حلول الشبكة Omnitouch Ansible تستخدم خدمات شبكة (4G/5G) الخلوية الكاملة بطريقة متسقة وقابلة للتكرار وأوتوماتيكية. توفر هذه الوثيقة نظرة عامة حول كيفية استغلالنا لـ Ansible لتنظيم نشرات الاتصالات المعقدة.

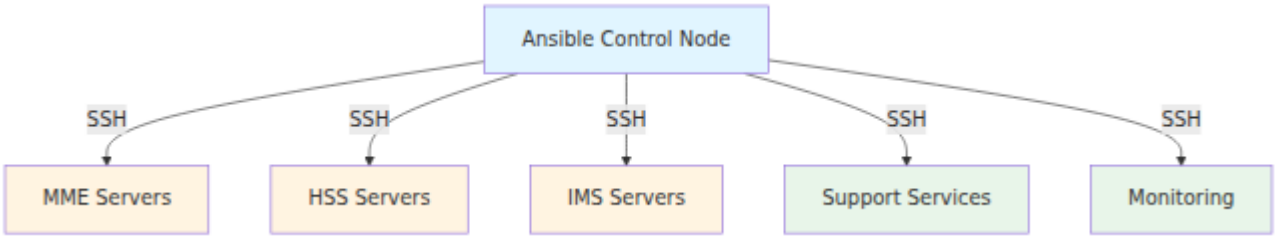
ما هو Ansible؟

Ansible هو أداة أتمتة مفتوحة المصدر تتيح لك:

- تكوين الأنظمة
- نشر البرمجيات
- تنظيم سير العمل المعقد
- إدارة البنية التحتية ككود

من أنها Ansible نهجًا تصريحيًا - حيث تصف **الحالة المرغوبة** لأنظمتك، ويتأكد Ansible يستخدم تصل إلى تلك الحالة.

كيف تستخدم Omnitouch Ansible



المفاهيم الأساسية

1. الجرد (ملفات المضيفين)

يحدد **ما** الأنظمة التي يجب إدارتها. تحتوي كل نشرية عميل على ملف مضيف يصف:

- جميع الآلات الافتراضية في الشبكة
- عناوين IP الخاصة بها
- تكوين الشبكة
- معلمات محددة للخدمة

ملفات المضيفين هي ما ستعمل معه لتعريف شبكتك.

انظر: تكوين ملف المضيفين

2. الأدوار

يحدد **كيف** يتم تكوين كل مكون. الأدوار هي وحدات قابلة لإعادة الاستخدام تحتوي على:

- مهام (خطوات للتنفيذ)
- قوالب (قوالب ملفات التكوين)
- معالجات (إجراءات يتم تشغيلها بواسطة التغييرات)
- متغيرات (قيم التكوين الافتراضية)

أمثلة على الأدوار لمكونات OmniCore: `omnihss`, `omnisgwc`, `omnipgwc`, `omnidra`, إلخ

تحدد فريق ONS هذه الأدوار. يمكنك تعديلها. ومع ذلك، هناك عمومًا طرق أنظف لإجراء أي تعديلات تحتاجها من داخل ملف المضيفين الخاص بك.

3. دفاتر التشغيل

تنظم **متى** و **أين** يتم تطبيق الأدوار:

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

نستخدم هذه بشكل أساسي كمجموعات للأدوار.

في الممارسة العملية، يطبق كل دفتر تشغيل خدمة دورًا واحدًا، ويتم سحب الإعداد المشترك لتشغيل `import_playbook` تستخدم `services/epc.yml` بشكل منفصل. حزمة الخدمة مثل `common.yml` أولاً ثم كل دفتر تشغيل خاص بالخدمة:

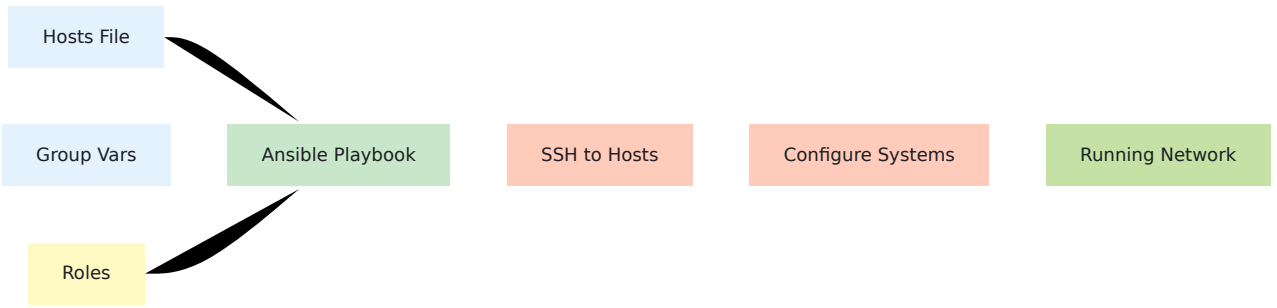
```yaml
- name: Install Common
  import_playbook: common.yml

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

4. متغيرات المجموعة

يوفر **تكوينًا محددًا للعميل** يتجاوز القيم الافتراضية للدور. هنا يحدث تخصيص العميل دون تعديل الأدوار الأساسية.

انظر: متغيرات المجموعة والتكوين

هندسة النشر



عملية النشر

1. تعريف البنية التحتية

إنشاء ملف مضيف يصف طوبولوجيا شبكتك:

ملاحظة تخطيط: قبل تعريف البنية التحتية، راجع معيار تخطيط IP للحصول على إرشادات حول تقسيم الشبكة، وتخصيص عناوين IP، وتنظيم الشبكات الفرعية.

مستخدمو Proxmox: إذا كنت تنشر على Proxmox، انظر نشر VM/LXC على Proxmox لتوفير VMs/حاويات بشكل أوتوماتيكي.

انظر: تكوين ملف المضيفين و مرجع التكوين

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

2. تخصيص التكوين

تعيين المتغيرات المحددة للعميل في `group_vars`:

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

#ToDo - إضافة رابط هنا إلى مرجع التكوين للقائمة الكاملة

3. تشغيل دفاتر التشغيل

نشر الشبكة:

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

4. النشر الآلي

سوف يقوم Ansible بـ:

- إنشاء/توفير VMs (إذا كنت تستخدم تكامل Proxmox/VMware)
- تكوين الشبكات
- تثبيت حزم البرمجيات من ذاكرة التخزين المؤقت APT
- نشر كود التطبيق
- تكوين الخدمات بإعدادات العميل
- بدء الخدمات
- التحقق من النشر

المكونات الرئيسية التي نقوم بنشرها

OmniCore (منصة النواة الحزمية 4G/5G)

- **OmniHSS** - خادم المشتركين المنزلي
- **OmniSGW** - بوابة الخدمة (طبقة التحكم)
- **OmniPGW** - بوابة الحزم (طبقة التحكم)
- **OmniUPF** - وظيفة طبقة المستخدم
- **OmniDRA** - وكيل توجيه Diameter
- **OmniTWAG** - بوابة الوصول اللاسلكي الموثوق

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

OmniCall (منصة الصوت والرسائل)

- **OmniCall CSCF** - وظيفة التحكم في جلسة المكالمات (P-CSCF، I-CSCF، S-CSCF)
- **OmniTAS** - خادم تطبيق IMS (خدمات VoLTE/VoNR)
- **OmniMessage** - مركز الرسائل القصيرة (SMS-C)
- **OmniMessage SMPP** - دعم بروتوكول SMPP
- **OmniSS7** - مكونات الإشارة SS7 (STP، HLR، CAMEL)
- **VisualVoicemail** - وظيفة البريد الصوتي

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

OmniCharge/OmniCRM

- **منصة CRM** - إدارة علاقات العملاء، التسجيل الذاتي، الفوترة

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

خدمات الدعم

- **DNS** - حل DNS الشبكة
- **خادم الترخيص** - إدارة الترخيص

- **المراقبة** - Prometheus، Grafana

انظر: نظرة عامة على هندسة النشر

إدارة الحزم

نستخدم نموذج توزيع حزم هجين:

حزم APT مسبقة التجميع

توزع جميع برامج Omnitouch كحزم Debian (`.deb` files):

- تم بناؤها من المصدر في خط أنابيب CI/CD لدينا
- تم إصدارها واختبارها
- مستضافة على مستودعات الحزم

نظام ذاك��ة التخزين المؤقت APT

يمكن للعملاء الاختيار بين:

1. **ذاكرة التخزين المؤقت المحلية APT** - مرآة للحزم المطلوبة في الموقع للنشر غير المتصل
2. **المستودع العام** - الوصول المباشر إلى مستودع الحزم المستضاف من قبل Omnitouch

يوجد أيضًا `omnicore` موحد `.deb` (مبني من `packaging/`) الذي يجمع دفاتر التشغيل والأدوات لتوفير نشر ذاتي.

انظر: نظام ذاكرة التخزين المؤقت APT

إدارة الترخيص

تتطلب جميع مكونات برامج Omnitouch تراخيص صالحة يتم إدارتها من خلال خادم ترخيص مركزي:

- تتحقق المكونات من صلاحية الترخيص عند بدء التشغيل

- يتم تمكين/تعطيل الميزات بناءً على الترخيص
- يمكن أن يكون خادم الترخيص محليًا أو مستضافًا في السحابة

انظر: خادم الترخيص

فوائد هذا النهج

القابلية للتكرار

يمكن أن تقوم نفس دفاتر تشغيل Ansible بنشر:

- مختبرات التطوير
- بيئات الاختبار
- الشبكات الإنتاجية
- مواقع العملاء

الاتساق

يستخدم كل نشر نفس التكوينات المختبرة، مما يقلل من الأخطاء البشرية.

التحكم في الإصدار

يتم تعريف البنية التحتية ككود في Git:

- تتبع جميع التغييرات
- مراجعة قبل النشر
- التراجع إذا لزم الأمر

التخصيص بدون تعقيد

يمكن للعملاء تخصيص نشرهم من خلال `group_vars` دون تعديل الأدوار الأساسية.

النشر السريع

نشر شبكة خلوية كاملة في ساعات بدلاً من أيام أو أسابيع.

البدء

المتطلبات المسبقة

تحتاج إلى تثبيت التبعيات المطلوبة. يستخدم هذا المشروع، Ansible قبل تشغيل دفاتر تشغيل Python، **يتطلب** لذا **uv** لإدارة بيئة.

1. تثبيت uv

إذا لم يكن uv مثبتًا بالفعل، قم بتثبيته باستخدام أي طريقة تناسب عقد التحكم الخاص بك. انظر وثائق تثبيت uv. على سبيل المثال:

```
pipx install uv                                    # عبر pipx
curl -LsSf https://astral.sh/uv/install.sh | sh    # مثبت مستقل
```

2. تثبيت التبعيات

من جذر المستودع، قم بتشغيل:

```
uv sync
```

هذا ينشئ `.venv/` معزولة ويثبت Ansible بالإضافة إلى جميع حزم Python اللازمة (المثبتة في `uv.lock`) لأتمتة نشر Omnitouch.

3. تفعيل البيئة

قم بتفعيل البيئة الافتراضية، واحتفظ بها مفعلة لجميع أوامر Ansible:

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

4. تثبيت مجموعات Ansible

قم بتثبيت مجموعات Ansible التي تعتمد عليها دفاتر التشغيل (من `requirements.yml`):

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

5. تشغيل أوامر Ansible

مباشرة Ansible مع تفعيل البيئة، قم بتشغيل أوامر:

```
ansible --version
```

ملاحظة: قم بإلغاء تفعيل البيئة عند الانتهاء عن طريق تشغيل `deactivate`.

خطوات النشر

1. مراجعة تكوين ملف المضيفين لفهم كيفية تعريف شبكتك
2. تعلم عن متغيرات المجموعة للتخصيص
3. فهم نظام ذاكرة التخزين المؤقت APT لإدارة الحزم
4. مراجعة هندسة النشر لرؤية كيفية تناسق كل شيء
5. انشر!

الخطوات التالية

- **خطط لهيكل شبكتك وتخصيص IP** - معيار تخطيط IP
- تكوين ملف المضيفين - تعلم كيفية تعريف طوبولوجيا شبكتك
- فهم توزيع الحزم - نظام ذاكرة التخزين المؤقت APT
- خادم الترخيص - تعلم عن إدارة الترخيص
- نظرة عامة على هندسة النشر - رؤية الصورة الكاملة
- تكوين متغيرات المجموعة - تخصيص نشر الخاص بك
- دفاتر التشغيل المساعدة - أدوات تشغيلية لفحص الصحة، النسخ الاحتياطي، والصيانة

مستودع APT وتوزيع الحزم

نظرة عامة

يوفر نظام APT من Omnitouch توزيع الحزم لجميع عمليات النشر. يتم تقديم نوعين من المحتوى:

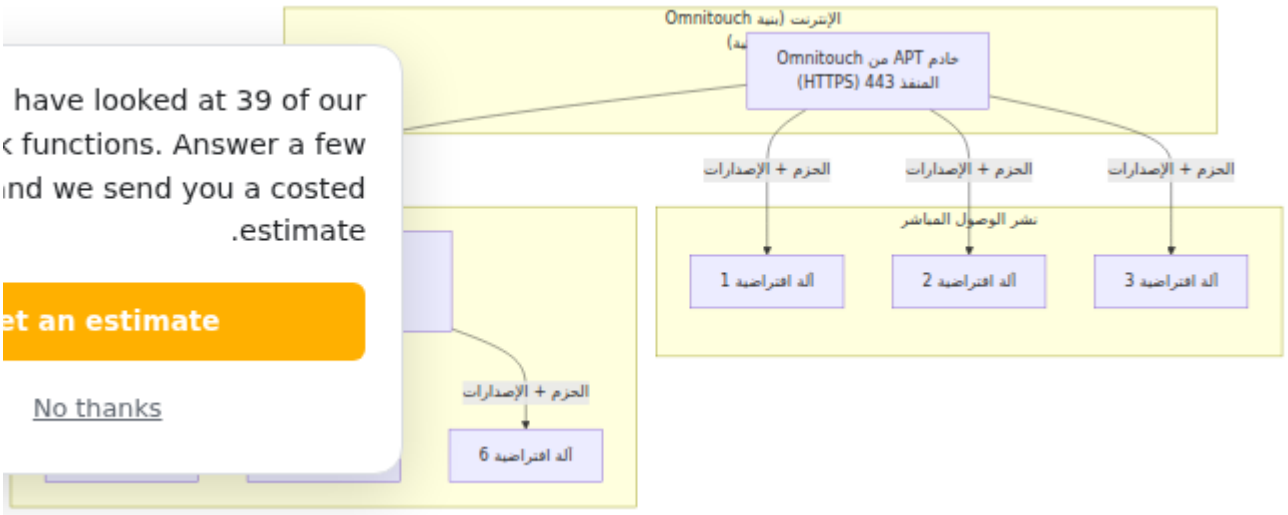
1. **حزم APT**: حزم Debian المثبتة عبر `apt install`
2. **الإصدارات الثنائية**: ثنائيات مسبقة البناء يتم تنزيلها مباشرة (مصدري Prometheus، الوكلاء، إلخ)

تتضمن صفحة الحزم على خادم APT **الإعداد الذاتي** أيضًا علامة **(OmniCore)**، والتي توفر حزمة `omnicore` (المقدمة في `/pool/main/o/omnicore/`) التي تحتوي على مجموعة أدوات أتمتة Ansible الكاملة للعملاء الذين يرغبون في نشر وإدارة شبكتهم بأنفسهم.

يدعم النظام نموذجين للنشر:

1. **الوصول المباشر** APT: تقوم الآلات الافتراضية بسحب الحزم مباشرة من خادم Omnitouch الخاص بـ (مثل `apt.<region>.omnitou.ch`)
2. **مرآة التخزين المؤقت المحلية** Omnitouch: يقوم خادم محلي بالمزا��نة من ويقدم الحزم للآلات الافتراضية (للنشر غير المتصل/المعزول)

الهيكلية



have looked at 39 of our
k functions. Answer a few
nd we send you a costed
.estimate
et an estimate
No thanks

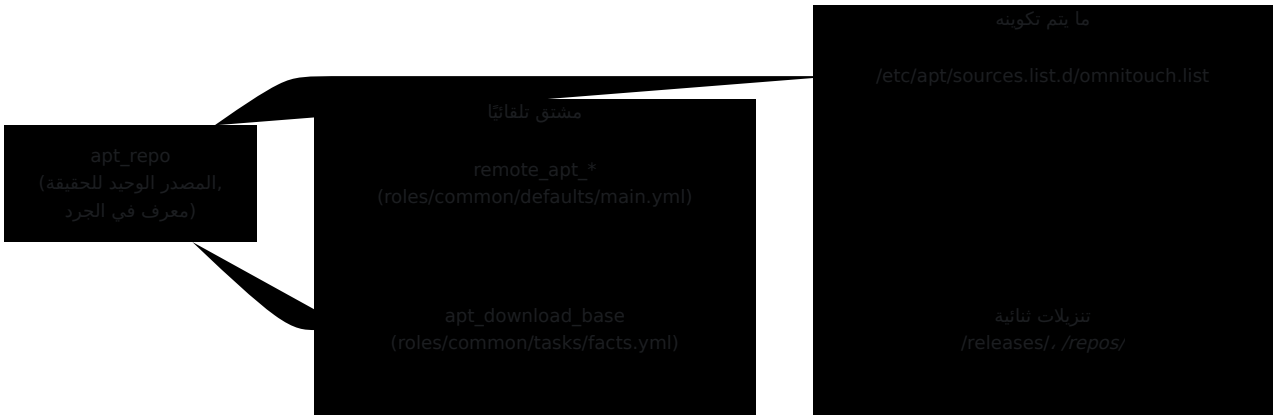
المحتوى المقدم

يستضيف خادم APT جميع المحتويات المطلوبة للنشر:

نوع المحتوى	الوصف	المسار
حزم Omnitouch	حزم `.deb` مخصصة (omnihss، omnimme، إلخ)	`/dists/<distro>/`
مجموعة أدوات OmniCore	حزمة `omnicore`: أتمتة Ansible الكاملة (الأدوار، كتب اللعب، التبعيات) للنشر الذاتي	`/pool/main/o/omnicore/`
حزم Ubuntu	حزم Ubuntu المخزنة مع جميع التبعيات	`/<distro>/pool/main/`
إصدارات GitHub	ثنائيات مسبقة البناء (Prometheus، Grafana، VoIPmonitor، إلخ)	`/releases/<org>/<repo>/`
أرشيفات المصدر	أرشيفات المصدر لتطبيقات الويب (CGrateS_UI، speedtest)	`/repos/`
حزم الطرف الثالث	Galera، FRR، InfluxDB، KeyDB، إلخ.	`/releases/<vendor>/`

متغيرات التكوين

هو المصدر الوحيد للحقيقة لتوزيع الحزم. قم بتعريفه مرة واحدة في جردك. `apt_repo` لتنزيل الثنائيات، أو المخطط، أو المنفذ، أو URL كل شيء آخر مشتق منه. لا تقوم بتكوين عنوان بيانات الاعتماد بشكل منفصل.


apt_repo
(المصدر الوحيد للحقيقة,
معرف في الجرد)
مشتق تلقائيًا
remote_apt_*
(roles/common/defaults/main.yml)
apt_download_base
(roles/common/tasks/facts.yml)
ما يتم تكوينه
/etc/apt/sources.list.d/omnitouch.list
تنزيلات ثنائية
/releases/، /repos/

كيف يعمل الاشتقاق

المتغير	مكان التعيين	الدور
`apt_repo`	جردك	الكتلة الوحيدة التي تقوم بتكوينها. يتم قالبتها مباشرة في سطر مصدر APT.
`remote_apt_*`	`roles/common/defaults/main.yml`	مشتق من `apt_repo`. الملاذ النهائي عندما يكون `apt_repo` غائبًا: `apt.us-fl.omnitou.ch` / `443` / `https`.
`apt_download_base`	`roles/common/tasks/facts.yml`	عنوان URL الأساسي المحسوب الوحيد الذي تستخدمه كل دور لتنزيلات الثنا��يات، بحيث يتفق سطر مصدر APT وتنزيلات الثنائيات دائمًا على المضيف، والمخطط، والمنفذ، وبيانات الاعتماد.

فإن جردًا عاديًا **يحتاج فقط إلى** `apt_repo`، نظرًا لأن كلا القيم المشتقة مبنية من نفس كتلة **`apt_repo`**. لا تقوم بتعيين `remote_apt_*` بنفسك.

> `apt_cache_servers` **استثناء: خادم مرآة التخزين المؤقت.** المضيف في مجموعة `apt_repo` بشكل صريح. تشير `remote_apt_*` هو المكان الوحيد الذي يتم فيه تعيين

> خادم Omnitouch `remote_apt_*` الخاصة به إلى *التخزين المؤقت المحلي*. ثم تصف قيم *العلوي* الذي يقوم بالمزامنة منه (انظر الخيار 2).

المخطط والمنفذ

عندما يتم حذف `http`. → **أي منفذ آخر**، `https` → يتبع المخطط المنفذ تلقائيًا: **المنفذ** `443` `apt_repo.apt_repo_port`:

- **الوصول المباشر (HTTPS)** يكون افتراضيًا إلى `443`
- **وضع التخزين المؤقت (HTTP)** يكون افتراضيًا إلى `8080`

متى يتم استخدام كل وضع

السيناريو	س��ر مصدر APT	تنزيلات ثنائية (`apt_download_base`)
`use_apt_cache: true` (تخزين مؤقت)	`http://<cache>:8080/...` (بدون مصادقة)	`http://<cache>:8080/...` (بدون مصادقة)
`use_apt_cache: false` (مباشر)	`https://<user>:<pass>@<server>/...`	`https://<user>:<pass>@<server>:443/...`

الخيار 1: الوصول المباشر

الخاص بـ APT للنشر مع الاتصال بالإنترنت، تقوم الآلات الافتراضية بسحب الحزم مباشرة من خادم Omnitouch.

متطلبات الشبكة

المصدر: يجب أن يكون عنوان IP في **القائمة البيضاء لعنوان IP** العام الخاص بك مدرجًا أثناء الإعداد، قدم الشبكات الفرعية الخاص بـ Omnitouch. القائمة البيضاء على خادم APT الخاصة بك إلى Omnitouch. في المقابل، ستتلقى:

- الأساسية HTTP لمصادقة **اسم المستخدم** و **كلمة المرور**

- **FQDN** لخادم APT (مثل `apt.<region>.omnitou.ch`)

متطلبات جدار الحماية: يجب السماح بالوصول الصادر إلى نطاقات IP التالية الخاصة بـ Omnitouch:

الشبكة	النطاق
IPv4	`144.79.167.0/24`
IPv4	`160.22.43.0/24`
IPv6	`2001:df3:dec0::/48`
ASN	`AS152894`

الخدمات التي تتطلب الوصول إلى بنية Omnitouch التحتية:

الخدمة	المنفذ	البروتوكول	الغرض
خادم APT	443	TCP	تنزيل الحزم (HTTPS)
خادم APT	53	TCP/UDP	حل DNS لـ FQDN الخاص بـ APT
خادم الترخيص	123	UDP	مزامنة الوقت عبر NTP للتحقق من الترخيص
خادم الترخيص	53	TCP/UDP	حل DNS للتحقق من الترخيص

تأكد من السماح بحركة المرور HTTPS (TCP/443)، NTP (UDP/123)، وDNS (TCP+UDP/53) إلى نطاقات IP الخاصة بـ Omnitouch.

التكوين

```yaml
all:
  vars:
    use_apt_cache: false

    # المصدر الوحيد للحقيقة. تكوين تنزيل الثنائيات (remote_apt_*) و
    # عنوان URL لـ apt_download_base مشتق من هذا تلقائيًا - لا تقوم بتعيينهم
    # بشكل منفصل.
    apt_repo:
      apt_server: "apt.<region>.omnitou.ch"   # FQDN المقدم من
Omnitouch
      apt_repo_username: "your-username"
      apt_repo_password: "your-password"
      # apt_repo_port: 443                    # اختياري؛ الافتراضي هو
443 (HTTPS)
```

المعلمات

مصادر حزم APT (`apt_repo`)

المعلمة	النوع	مطلوب	الافتراضي	الوصف
`apt_repo.apt_server`	سلسلة	نعم	-	اسم مضيف أو APT خادم IP عنوان
`apt_repo.apt_repo_username`	سلسلة	نعم	-	اسم مستخدم مصادقة HTTP الأساسية
`apt_repo.apt_repo_password`	سلسلة	نعم	-	كلمة مرور مصادقة HTTP الأساسية
`apt_repo.apt_repo_port`	عدد صحيح	لا	`443`	منفذ الخادم. يتبع المخطط المنفذ (`443` → HTTPS، وإلا HTTP).

تنزيلات ثنائية (`remote_apt_*`)

لا تقم بتعيينها في جرد الوصول المباشر. `apt_repo` توجد **هذه مشتقة تلقائيًا من** فقط حتى يتمكن خادم مرآة التخزين المؤقت من الإشارة إلى مصدر علوي مختلف (انظر الخيار 2:). كمرجع، يتم تعيينها افتراضيًا كما يلي

المعلمة	مشتقة من	الملاذ
`remote_apt_server`	`apt_repo.apt_server`	`apt.us-fl.omnitou.ch`
`remote_apt_port`	`apt_repo.apt_repo_port`	`443`
`remote_apt_protocol`	المنفذ (`443` → `https`)	`https`
`remote_apt_user`	`apt_repo.apt_repo_username`	`""`
`remote_apt_password`	`apt_repo.apt_repo_password`	`""`

عام

المعلمة	النوع	مطلوب	الافتراضي	الوصف
`use_apt_cache`	بولياني	نعم	-	يجب أن تكون `false` للوصول المباشر

أنماط URL (الوصول المباشر)

مصادر حزم APT (المعينة في `/etc/apt/sources.list.d/omnitouch.list`):

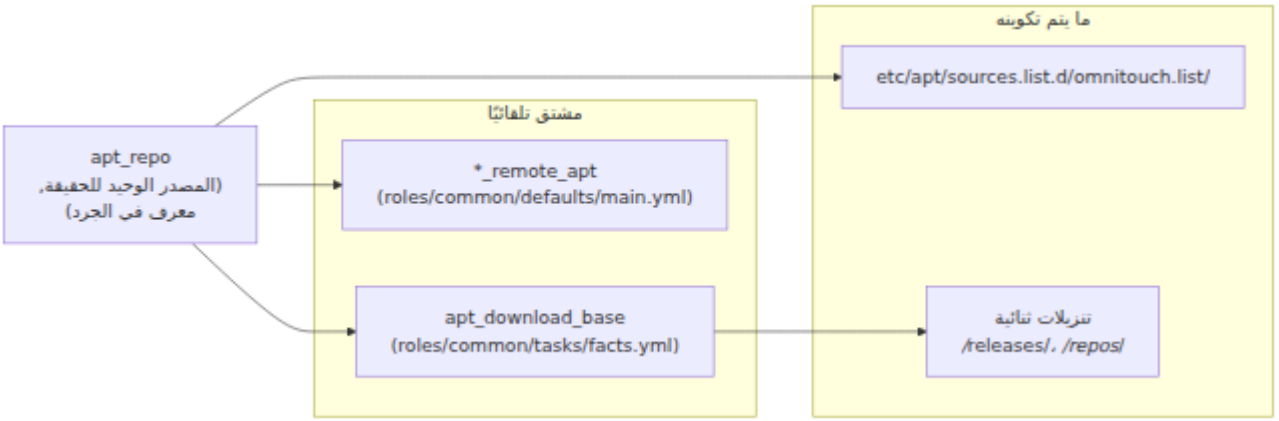
```
deb [trusted=yes] https://{username}:{password}@{apt_server}/
noble main
```

التنزيلات الثنائية (المبنية من `apt_download_base` بواسطة مهام Ansible `get_url`):

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

> يتم إلحاق منفذ غير HTTPS). إنه الافتراضي لـ URL يتم حذف المنفذ `443` من عنوان ما لم يكن `443` HTTP ويحول المخطط إلى قياسي كـ `:<port>`.

كيف يعمل



تقوم الآلات الافتراضية بالمصادقة باستخدام مصادقة HTTP الأساسية لكل من حزم APT والتنزيلات الثنائية. يتم أيضًا تقديم حزم نظام Ubuntu من خادم Omnitouch (مخزنة مسبقًا)، لذا لا تحتاج الآلات الافتراضية إلى الوصول إلى مرايا Ubuntu.

توقيع الحزم (GPG)

ينشئ خادم APT من Omnitouch مفتاح توقيع المستودع ويقدم المفتاح العام المحمي في `/omnitouch.gpg`. تلقائيًا بجلب `common` بالنسبة للمضيفات ذات الوصول المباشر، يقوم الدور بحيث يتحقق `/etc/apt/trusted.gpg.d/omnitouch.gpg` هذا المفتاح وتثبيته في `apt-get update`. يحدث هذا تلقائيًا؛ لا من بيانات التعريف الخاصة بالمستودع دون تحذيرات `NO_PUBKEY` حاجة لإدارة المفاتيح يدويًا.

> لا يستخدم وضع مرآة التخزين المؤقت التحقق من GPG. تستخدم المضيفات التي تسحب من التخزين المؤقت المحلي `[trusted=yes]` (انظر الخيار 2) بدلاً من ذلك.

الخيار 2: مرآة التخزين المؤقت المحلية

للنشر غير المتصل، المعزول، أو المقيد بالنطاق الترددي، قم بنشر ذاكرة تخزين مؤقت محلية لـ Omnitouch. تقوم بمزامنة جميع المحتويات من APT

الهيكلية



التكوين

قم بتعريف خادم التخزين المؤقت في ملف المضيفين الخاص بك. إنه المكان الوحيد الذي يتم فيه تعيين `remote_apt_*` بشكل صريح. تصف تلك المتغيرات خادم Omnitouch **العلوي** الذي يقوم التخزين المؤقت بالمزامنة منه:

```yaml
apt_cache_servers:
  hosts:
    example-apt-cache:
      ansible_host: 192.168.1.100
      gateway: 192.168.1.1
  vars:
    # Omnitouch العلوي يقوم خادم التخزين المؤقت بمزامنة المحتوى من مستودع
    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  # يتم تعيينه تلقائيًا عند وجود مجموعة
apt_cache_servers
    # apt_repo.apt_server: يتم اشتقاقه تلقائيًا إلى 192.168.1.100 (أول خادم
تخزين مؤقت)
```

كيف يعمل:

- **خادم التخزين المؤقت** (`192.168.1.100`): يستخدم بيانات اعتماد `remote_apt_*` لمزامنة المحتوى من خادم Omnitouch العلوي عبر HTTPS.
- **جميع المضيفات الأخرى**: تشتق تلقائيًا `apt_repo.apt_server: "192.168.1.100"` وتسحب من التخزين المؤقت على المنفذ `8080` دون بيانات اعتماد.

المعلمات

مصادر حزم APT (`apt_repo`)

المعلمة	النوع	مطلوب	الافتراضي	الوصف
`apt_repo.apt_server`	سلسلة	نعم	مشتق تلقائيًا	تخزين IP عنوان ت المحلي. يتم ه تلقائيًا من أول مضيف في `apt_cache_s` ا لم يتم تحديده
`apt_repo.apt_repo_username`	سلسلة	لا	-	غير مطلوب عند ستخدام التخزين مؤقت (لا حاجة لمصادقة)
`apt_repo.apt_repo_password`	سلسلة	لا	-	غير مطلوب عند ستخدام التخزين مؤقت (لا حاجة لمصادقة)
`apt_repo.apt_repo_port`	عدد صحيح	لا	`8080`	الذي تستخدمه ت للوصول إلى لتخزين المؤقت المحلي. (محل `apt_cache_p` المهجور).

مزامنة خادم التخزين المؤقت (`remote_apt_*`)

تقوم هذه المتغيرات بتكوين كيفية مزامنة خادم التخزين المؤقت للمحتوى من Omnitouch. هذه `remote_apt_*` هي الحالة الوحيدة التي يتم فيها تعيين يدويًا:

المعلمة	النوع	مطلوب	الافتراضي	الوصف
`remote_apt_server`	سلسلة	نعم	-	خادم APT العلوي من Omnitouch للمزامنة منه
`remote_apt_port`	عدد صحيح	لا	`443`	منفذ خادم APT العلوي من Omnitouch
`remote_apt_protocol`	سلسلة	لا	`https`	البروتوكول لاتصال المزامنة
`remote_apt_user`	سلسلة	نعم	-	بيانات الاعتماد للمزامنة من Omnitouch
`remote_apt_password`	سلسلة	نعم	-	بيانات الاعتماد للمزامنة من Omnitouch

عام

المعلمة	النوع	مطلوب	الافتراضي	الوصف
`use_apt_cache`	بولياني	لا	`true`	عند `true` يتم تعيينه تلقائيًا إلى وجود مجموعة `apt_cache_servers`

> **مهجور:** لم يعد يستخدم `apt_cache_port`. يتم الآن تعيين المنفذ الذي تتصل به المضيفات عبر `apt_repo.apt_repo_port` (الافتراضي هو `8080` عند حذفه).

أنماط URL (وضع التخزين المؤقت)

مصادر حزم APT (المعينة في `/etc/apt/sources.list.d/omnitouch.list`):

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

التنزيلات الثنائية (المبنية من `apt_download_base`):

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

لا تحتاج الوصول إلى التخزين المؤقت إلى بيانات اعتماد. تستخدم تكوين APT `[trusted=yes]`.

نشر التخزين المؤقت

1. **قم بتوفير خادم التخزين المؤقت** (آلة افتراضية أو حاوية LXC +50 بسعة قرص جيجابايت)
2. **قم بتشغيل برنامج إعداد التخزين المؤقت:**

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

3. **تحقق من التخزين المؤقت** عن طريق التصفح إلى `http://192.168.1.100:8080/`

ما الذي يتم مزامنته

تقوم مرآة التخزين المؤقت بمزامنة **جميع المحتويات** من خادم APT الخاص بـ Omnitouch. تختلف الآلية حسب نوع المحتوى:

- **حزم APT** (`/dists/` + `/pool/` لكل من Omnitouch وUbuntu) تتم مزامنتها باستخدام `curl` مدفوعة بفهرس مستودع Debian. تقوم المرآة بجلب ملف `Release` للحصول على اسم `Packages.gz` العلوي (وتتخطى إذا لم يتغير)، وتحلل الملف/SHA256 لا/التي تقوم بتنزيل الملفات المفقودة فقط أو حجم كل حزمة،

مع النسخة المحلية. يتم التحقق من كل ملف تم تنزيله باستخدام SHA256 تتطابق قبل الالتزام به SHA256.

- **الإصدارات الثنائية وأرشيفات المصدر** (`/releases/`، `/repos/`) ليس لديها فهرس على نمط Debian، لذا تتراجع إلى `wget --recursive --timestamping`، الذي يقوم بإعادة تنزيل الملفات فقط التي تحتوي على `Last-Modified` أحدث.

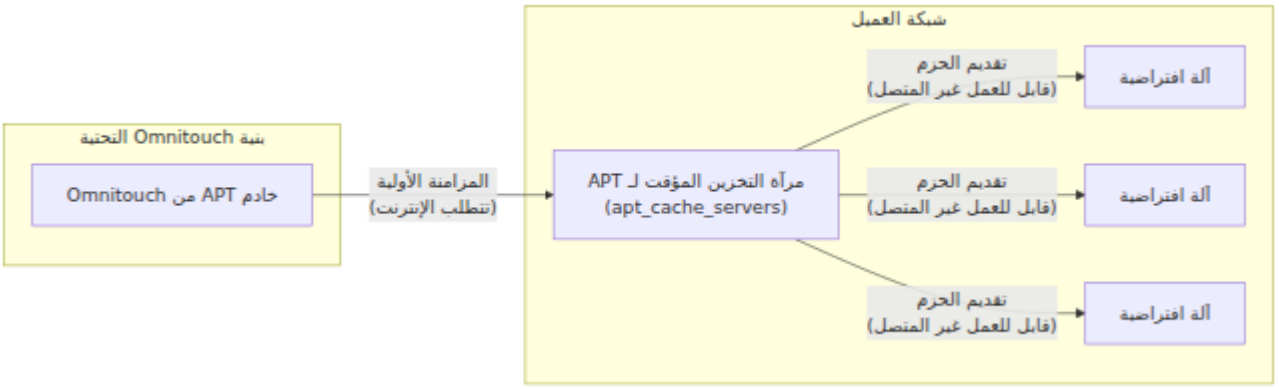


المسارات التي تم مزامنتها:

المحتوى	المسار
بيانات تعريف مستودع APT (ملفات Packages، Release)	`/dists/<distro>/`
حزم .deb المخصصة من Omnitouch	`/pool/main/`
حزم Ubuntu وجميع التبعيات	`/<distro>/pool/main/`
إصدارات GitHub (Prometheus، Grafana، Zabbix، إلخ)	`/releases/`
أرشيفات المصدر (Erlang، Elixir، CGrateS_UI، إلخ)	`/repos/`

بعد المزامنة الأولية، يمكن للتخزين المؤقت تقديم جميع الحزم دون الاتصال بالإنترنت.

كيف يعمل



الأساسية لكل المحتوى. تتم HTTP تقوم مرآة التخزين المؤقت بالمصادقة باستخدام مصادقة مزامنة حزم APT باستخدام `curl` ضد فهرس مستودع Debian (`Release` + `Packages.gz`)، الخاص بها عن النسخة المحلية والتحقق من كل منها SHA256 وتنزيل الملفات فقط التي يختلف `wget --recursive --timestamping`. بعد التنزيل؛ تتراجع `/releases/` و`/repos/` إلى في كلتا الحالتين، تقوم المزامنات اللاحقة بجلب الملفات الجديدة أو المتغيرة فقط.

التكوين التلقائي

في جردك، يقوم النظام تلقائيًا `apt_cache_servers` عندما توجد مجموعة:

1. لجميع المضيفات (ما لم يتم تجاوز ذلك صراحة) `use_apt_cache: true` بتعيين
2. الخاص بأول خادم تخزين مؤقت IP من عنوان `apt_repo.apt_server` باشتقاق

مثال على التكوين الأدنى

```yaml
apt_cache_servers:
  hosts:
    apt-cache-01:
      ansible_host: 192.168.1.100
      gateway: 192.168.1.1
  vars:
    # يقوم خادم التخزين المؤقت بمزامنة المحتوى من مستودع Omnitouch العلوي
    remote_apt_server: "apt.<region>.omnitou.ch"
    remote_apt_user: "your-username"
    remote_apt_password: "your-password"
```

ما يحدث تلقائيًا:

- تحصل جميع المضيفات (باستثناء خادم التخزين المؤقت) على `use_apt_cache: true`
- تحصل جميع المضيفات (باستثناء خادم التخزين المؤقت) على `apt_repo.apt_server: "192.168.1.100"`
- تسحب جميع المضيفات من `http://192.168.1.100:8080/` دون بيانات اعتماد
- يقوم خادم التخزين المؤقت بمزامنة الحزم من `https://your-username:your-password@apt.<region>.omnitou.ch/`

تجاوز السلوك التلقائي

لإجبار الوصول المباشر حتى مع تعريف خوادم التخزين المؤقت:

```
all:
  vars:
    use_apt_cache: false  # إجبار الوصول المباشر حتى مع تعريف خوادم التخزين المؤقت

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

ملخص التكوين

السيناريو 1: الوصول المباشر إلى خادم APT (بدون تخزين مؤقت)

تقوم جميع المضيفات بسحب الحزم مباشرة من خادم مستودع APT. فقط `apt_repo` مطلوب. يتم اشتقاق قيم `remote_apt_*` منها.

```yaml
all:
  vars:
    use_apt_cache: false

    # المصدر الوحيد للحقيقة لكل من مصادر APT والتنزيلات الثنائية
    apt_repo:
      apt_server: "apt.<region>.omnitou.ch"
      apt_repo_username: "user"
      apt_repo_password: "pass"
```

النتيجة: تقوم جميع المضيفات بإنشاء `deb [trusted=yes] https://user:pass@apt.<region>.omnitou.ch/ noble main`

السيناريو 2: تم تعريف خادم تخزين مؤقت في ملف المضيفين (تلقائي)

خادم التخزين المؤقت موجود في جردك وسيتم نشره/مزامنته بواسطة Ansible.

```yaml
apt_cache_servers:
  hosts:
    cache-server:
      ansible_host: 192.168.1.100
      gateway: 192.168.1.1
  vars:
    # يقوم خادم التخزين المؤقت بمزامنة الحزم من المستودع المعتمد العلوي
    remote_apt_server: "apt.<region>.omnitou.ch"
    remote_apt_port: 443
    remote_apt_protocol: "https"
    remote_apt_user: "user"
    remote_apt_password: "pass"

# لا حاجة لتكوين في all: vars:
# كل شيء يتم اشتقاقه تلقائيًا من مجموعة apt_cache_servers
```

النتيجة:

- **خادم التخزين المؤقت**: يتزامن من `https://user:pass@apt.<region>.omnitou.ch/`

- **جميع المضيفين الآخرين**: توليد `deb [trusted=yes] http://192.168.1.100:8080/noble noble main` (بدون بيانات اعتماد)

السيناريو 3: ذاكرة التخزين المؤقت عن بُعد غير APT موجودة في ملف المضيفين (يدوي)

يوجد خادم التخزين المؤقت في مكان آخر وتم إعداده بالفعل (غير مُدار بواسطة Ansible الخاص بك).

```yaml
all:
  vars:
    use_apt_cache: true

    # توجيه جميع المضيفين إلى خادم التخزين المؤقت الخارجي
    apt_repo:
      apt_server: "192.168.1.100"  # عنوان IP لخادم التخزين المؤقت الخارجي
      apt_repo_port: 8080           # عادةً ما يعمل التخزين المؤقت على المنفذ 8080

# لا حاجة لمجموعة apt_cache_servers
# لا حاجة لـ remote_apt_* (التخزين المؤقت مُعد بالفعل خارجيًا)
```

النتيجة: جميع المضيفين يولدون `deb [trusted=yes] http://192.168.1.100:8080/noble noble main` (بدون بيانات اعتماد)

مثال كامل

هذا مثال كامل يعمل. يُظهر تكوين خادم التخزين المؤقت مع عدة مضيفات تطبيق:

```yaml
# مجموعة خادم التخزين المؤقت 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:
    # خادم التخزين المؤقت يتزامن مع الحزم من المستودع المعتمد
    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"

# خوادم التطبيقات
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

# التكوين العالمي
all:
  vars:
    # التكوين التلقائي (لا حاجة لتكوين يدوي):
    # - use_apt_cache: true (يتم تمكينه تلقائيًا عند وجود
apt_cache_servers)
```

```
    # - apt_repo.apt_server: "10.179.1.114" (يتم اشتقاقه تلقائيًا من خادم التخزين المؤقت)
```

ما يحدث أثناء النشر:

1. **خادم التخزين المؤقت (`10.179.1.114`):**
   - يستخدم `remote_apt_*` من قسم `vars:` الخاص به
   - يقوم بتنزيل جميع الحزم من `https://customer-username:customer-secure-token@apt.<region>.omnitou.ch/`
   - يقدم الحزم على المنفذ 8080 عبر nginx
2. **مضيفات التطبيقات (`customer-hss01`، `customer-mme01`، `customer-dns01`):**
   - يتم اكتشاف مجموعة `apt_cache_servers` تلقائيًا
   - يتم تعيين `use_apt_cache: true` تلقائيًا
   - يتم اشتقاق `apt_repo.apt_server: "10.179.1.114"` تلقائيًا
   - توليد: `deb [trusted=yes] http://10.179.1.114:8080/noble noble main`
   - سحب جميع الحزم من خادم التخزين المؤقت (لا حاجة لبيانات الاعتماد)

تحديث التخزين المؤقت

لمزامنة الحزم الجديدة أو التحديثات:

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

هذا يقوم بمزامنة جميع المحتويات من خادم APT الخاص بـ Omnitouch بشكل تدريجي:

- إصدارات جديدة من حزم Omnitouch
- حزم Ubuntu جديدة والاعتماديات
- إصدارات جديدة من GitHub

- ملفات tarball المصدر المحدثة

تكون إعادة المزامنة تدريجية: يتم مقارنة حزم APT بفهرس Debian العلوي بواسطة SHA256 (وتُتخطى تمامًا إذا كان ملف `Release` العلوي غير متغير)، بينما تستخدم `/releases/` و `/repos/` `wget --timestamping`. يتم تخطي الملفات غير المتغيرة الموجودة، مما يجعل إعادة المزامنة سريعة.

ملاحظة: خادم APT الخاص بـ Omnitouch هو المصدر الوحيد للحقيقة لجميع الحزم. يقوم `services/apt_cache.yml` ببناء ونشر الحزم إليه؛ تشغيل Omnitouch على مرايا التخزين المؤقت ثم يقوم بمزامنة أي شيء تم نشره حاليًا.

استكشاف الأخطاء وإصلاحها

فشل تحديث APT مع 401 غير مصرح به

الأعراض:

```
فشل في جلب https://10.179.1.115/noble/dists/noble/main/binary-amd64/Packages  401  غير مصرح به
```

الأسباب المحتملة:

- تم تعريف تكوين `apt_repo` في `all: vars:` بدلاً من `apt_cache_servers: vars:`
- تحاول المضيفات الوصول إلى المستودع المعتمد مباشرة بدلاً من التخزين المؤقت
- بيانات اعتماد `apt_repo_username` أو `apt_repo_password` غير صحيحة
- عنوان IP المصدر غير مدرج في القائمة البيضاء على خادم APT الخاص بـ Omnitouch
- استخدام بيانات اعتماد التخزين المؤقت للوصول المباشر أو العكس

الحل:

1. **تحقق من نطاق التكوين**: تأكد من تعريف `apt_repo` مع بيانات الاعتماد في `apt_cache_servers: vars:`، وليس في `all: vars:`
2. **تحقق من وضع التخزين المؤقت**: عند استخدام التخزين المؤقت، يجب على المضيفين الاتصال بخادم التخزين المؤقت (المنفذ 8080)، وليس المستودع العلوي

3. **تحقق من المصادر المُولدة**: على المضيف الفاشل، تحقق من `/etc/apt/sources.list.d/omnitouch.list`
    - **صحيح (وضع التخزين المؤقت)**: `deb [trusted=yes] http://10.179.1.114:8080/noble noble main`
    - **غير صحيح (لديه بيانات اعتماد في المكان الخطأ)**: `deb [trusted=yes] https://user:pass@10.179.1.115/noble noble main`
4. تحقق من أن بيانات الاعتماد صحيحة لوضع النشر الخاص بك
5. تأكد من أن عنوان IP العام الخاص بك مدرج في القائمة البيضاء مع Omnitouch (إذا كنت تستخدم الوصول المباشر)

فشل تنزيلات ثنائية (Node Exporter، Zabbix، إلخ.)

الأعراض: فشل playbook الخاص بـ Ansible في تنزيل الملفات من مسار `/releases/`

الأسباب المحتملة:

- تستمد التنزيلات الثنائية مصادقتها من `apt_repo` (غير صحيحة بيانات اعتماد `apt_repo` في الوضع المباشر)
- `remote_apt_user` / `remote_apt_password` على خادم مرآة التخزين المؤقت، بيانات غير صحيحة
- HTTP. وأي منفذ آخر يعني HTTPS مخطط أو منفذ خاطئ. تذكر أن المنفذ `443` يعني

الحل:

1. في الوضع المباشر، تحقق من أن `apt_repo.apt_repo_username` / `apt_repo.apt_repo_password` تتطابق مع تلك المقدمة من Omnitouch
2. على خادم مرآة التخزين المؤقت، تحقق من بيانات اعتماد `remote_apt_*` العلوية
3. تأكد من أن `apt_download_base` المشتق يستخدم المضيف/المخطط/المنفذ المتوقع (تحقق باستخدام `ansible-playbook ... -vvv` أو عن طريق فحص مهمة `get_url` الفاشلة)

خادم التخزين المؤقت لا يمكنه المزامنة

الأعراض: فشل playbook الخاص بخادم التخزين المؤقت في تنزيل الحزم

الأسباب المحتملة:

- خادم التخزين المؤقت ليس لديه وصول إلى الإنترنت
- بيانات `remote_apt_*` غير صحيحة
- جدار الحماية يمنع الاتصالات الصادرة إلى Omnitouch

الحل:

1. تحقق من أن خادم التخزين المؤقت يمكنه الوصول إلى خادم APT العلوي الخاص بـ Omnitouch على المنفذ 443
2. تحقق من بيانات `remote_apt_*`
3. راجع قواعد جدار الحماية للوصول الصادر

الوثائق ذات الصلة

- تكوين ملف المضيفين: تكوين الجرد والمتغيرات
- مرجع التكوين: مرجع كامل للمعلمات
- معمارية النشر: المعمارية العامة للنظام
- نشر Proxmox: نشر خادم التخزين المؤقت كحاوية LXC

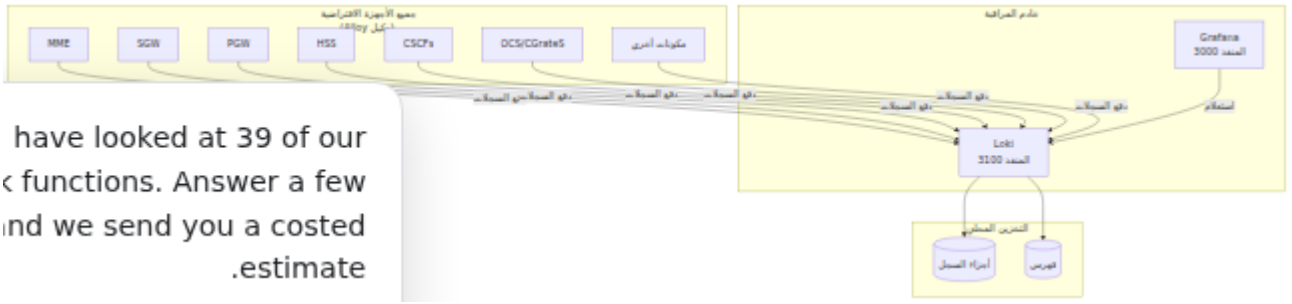
تسجيل مركزي

نظرة عامة

نظام تسجيل مركزي يجمع السجلات من جميع المكونات ويجعلها قابلة للبحث OmniCore يتضمن
من خلال Grafana. يستخدم النظام:

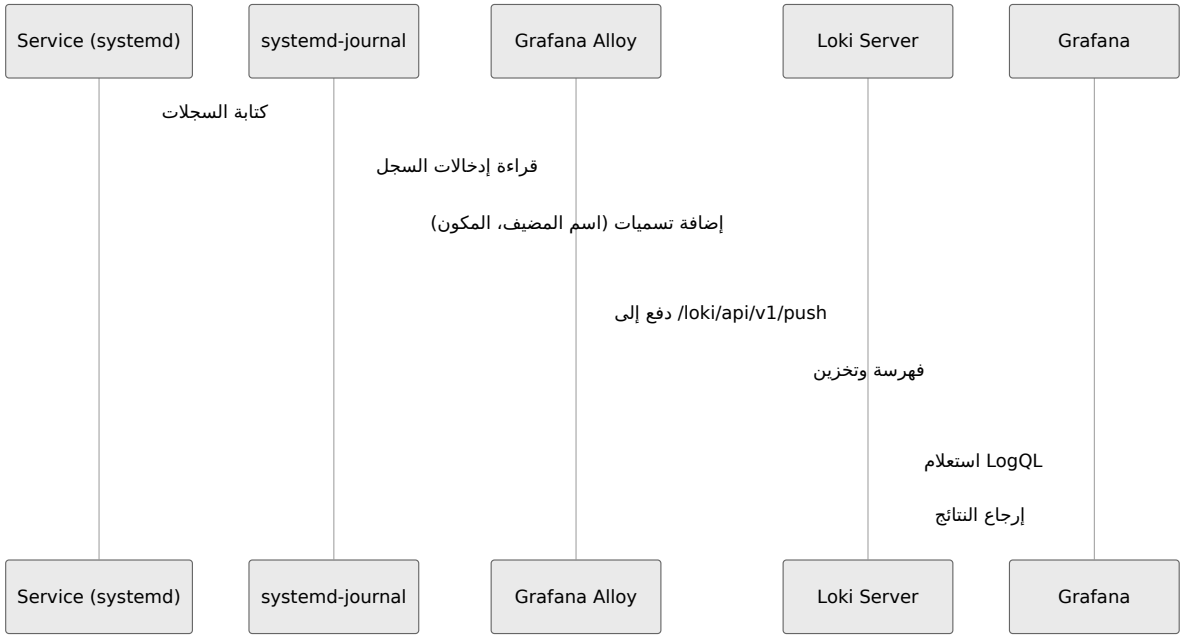
- **Grafana Alloy**: وكيل جمع السجلات يتم نشره على جميع الأجهزة الافتراضية
- **Grafana Loki**: خادم تجميع وتخزين السجلات على مضيفي المراقبة
- **لوحات Grafana**: لوحات جاهزة لكل نوع من المكونات

الهيكلية



كيف يعمل

تدفق جمع السجلات



تسميات السجل

تتضمن كل إدخال سجل هذه التسميات للتصفية والاستعلام:

التسمية	الوصف	المثال
`job`	اسم المضيف للجهاز الافتراضي المصدر	`customer-mme01`
`hostname`	نفس اسم الوظيفة (للتوافق)	`customer-mme01`
`component`	نوع المكون من مجموعة الجرد	`mme`, `pcscf`, `ocs`
`unit`	اسم وحدة systemd	`omnimme.service`
`level`	شدة السجل	`info`, `warning`, `error`
`service`	معرف Syslog	`omnimme`

التكوين

النشر التلقائي

يتم تكوين التسجيل تلقائيًا عندما:

1. توجد مجموعة `monitoring` في الجرد الخاص بك
2. يتم تشغيل الدور المشترك على المضيفين المستهدفين

لا يتطلب الأمر أي تكوين إضافي للوظائف الأساسية.

متطلبات الجرد

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

عند تعريف مجموعة `monitoring`:

- **مضيفو المراقبة**: يقومون بتشغيل خادم Loki، الذي يستقبل ويخزن السجلات. كما يقومون بتشغيل Alloy، الذي يجمع سجلات المضيف الخاصة به وينهي تدفق Syslog لجهاز الشبكة.
- **جميع المضيفين الآخ��ين**: يقومون بتشغيل وكيل Alloy، الذي يجمع السجلات ويرسلها إلى Loki.

تكوين الاحتفاظ

يتم تكوين Loki مع الاحتفاظ القائم على الوقت. يقوم الضاغط بحذف السجلات الأقدم من فترة الاحتفاظ:

الوصف	الافتراضي	الإعداد
يتم حذف السجلات الأقدم من فترة الاحتفاظ	72 ساعة (3 أيام)	الاحتفاظ القائم على الوقت

لتخصيص فترة الاحتفاظ، قم بتجاوز هذه المتغير في الجرد الخاص بك:

```
all:
  vars:
    loki_retention_period: "168h"  # على سبيل المثال، 7 أيام بالساعات
```

تكوين ذاكرة التخزين المؤقت لـ Alloy

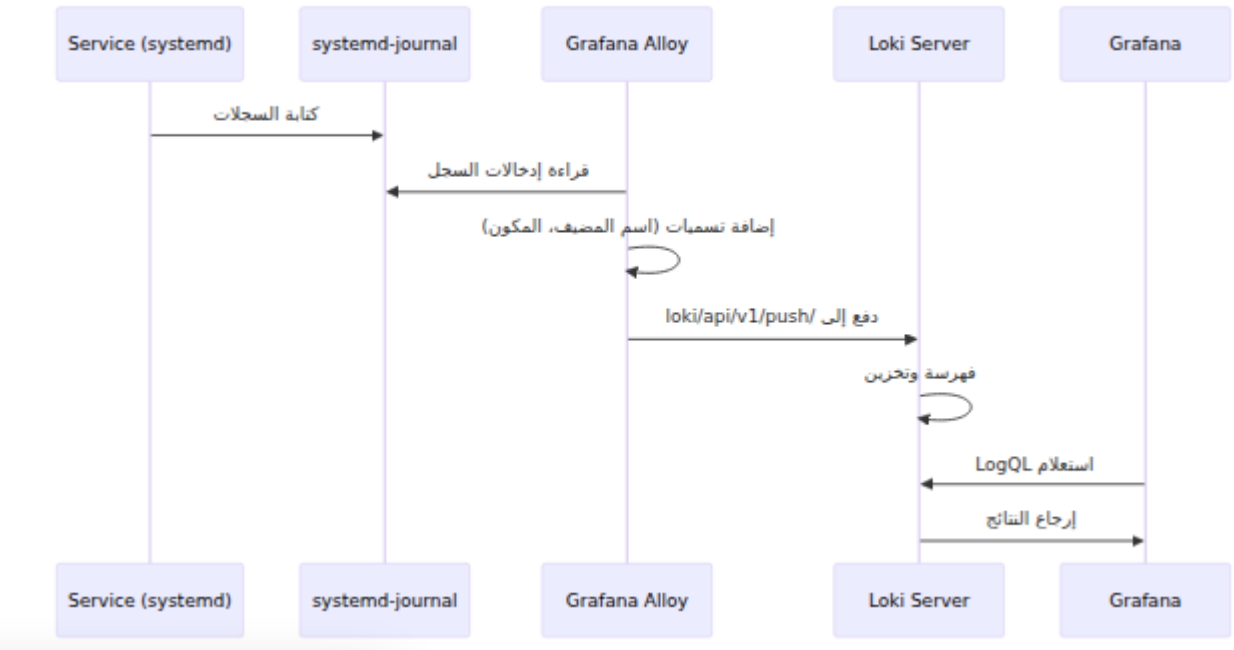
يقوم Alloy بتخزين السجلات محليًا في سجل كتابة مسبق (WAL) عندما يكون Loki غير متاح. يتم التخلص من الأجزاء المخزنة لمنع نمو القرص غير المحدود:

الوصف	الافتراضي	الإعداد
يتم إسقاط أجزاء السجل المخزنة التي تزيد عن 1 ساعة	1 ساعة	الحد الأقصى لعمر الجزء في WAL

غير متاح لفترة أطول من عمر الجزء، يتم إسقاط أقدم السجلات المخزنة Loki عندما يكون.

لجهاز الشبكة Syslog

ترسل هذه الأجهزة Syslog BSD لا يمكن لجهاز التوجيه والمبدلات والأجهزة الأخرى تشغيل Alloy. تحت Loki، التدفق ويكتبه إلى نفس Alloy إلى مضيف المراقبة بدلاً من ذلك. يستقبل `job="network_syslog"`. تشارك سجلات الجهاز بعد ذلك البحث، ونطاق الوقت، والاحتفاظ بسجلات المضيف.



الجهاز. Syslog المستمع هو جزء من بناء المراقبة القياسي. لا يقوم الموقع بتعيين علامة لاستقبال بتسمية كل جهاز من عنوان المصدر لبيانات الداتاجرام، الذي يحتفظ به جرد الموقع في Alloy يقوم خرائط البائع `devices:`.

والمُرسلين الآخرين، ومرجع RouterOS للحصول على معلما�ت الجامع، وتكوين الجهاز لجهاز التسمية، وخطوات استكشاف الأخطاء وإصلاحها، انظر Syslog لجهاز الشبكة.

لوحات Grafana

يتم توفير لوحات جاهزة تلقائيًا لكل نوع من المكونات.

اللوحات المتاحة

لوحة المعلومات	الموقع	الوصف
سجلات CSCF	السجلات ← سجلات CSCF	سجلات S-CSCF و P-CSCF و I-CSCF (OmniCSCF)
سجلات MME	السجلات ← سجلات MME	أحداث MME للاتصال والفصل والأخطاء
سجلات SGW	السجلات ← سجلات SGW	أحداث جلسة SGW وحامل
سجلات PGW	السجلات ← سجلات PGW	أحداث PDN وجلسة PGW
سجلات HSS	السجلات ← سجلات HSS	رسائل HSS Diameter وأحداث المصادقة
سجلات OmniMessage	السجلات ← سجلات OmniMessage	أحداث SMS وتسليم SMPP
سجلات OCS/CGrateS	السجلات ← سجلات OCS/CGrateS	أحداث الشحن وحركة مرور JSONRPC
استدعاءات CGrateS RPC	السجلات ← استدعاءات CGrateS RPC	البحث والتوافق بين طلبات/استجابات RPC مع زمن الانتظار
سجلات Syslog للشبكة	السجلات ← سجلات Syslog للشبكة	MikroTik / Syslog للشبكة، حسب الجهاز والشدة

ميزات لوحة المعلومات

تتضمن كل لوحة معلومات:

- **صندوق البحث**: بحث نصي حر عبر جميع السجلات
- **رسوم بيانية لمعدل السجل**: الحجم بمرور الوقت حسب المضيف والشدة
- **أقسام المكونات** (على سبيل المثال، S-CSCF و P-CSCF و I-CSCF): وجهات نظر مجمعة

- **تصفية الأخطاء**: وجهات نظر مسبقة التصفية للأخطاء والإخفاقات
- **تدفق مباشر**: تدفق سجلات في الوقت الحقيقي (تحديث كل 10 ثوانٍ)

استخدام لوحات المعلومات

1. انتقل إلى Grafana (عادةً `http://<monitoring-host>:3000`)
2. انتقل إلى **لوحات المعلومات** → **السجلات** → اختر المكون
3. استخدم متغير **البحث** لتصفية السجلات
4. اضبط نطاق الوقت حسب الحاجة

استعلام السجلات

أساسيات LogQL

يستخدم Loki LogQL للاستعلام. الصيغة الأساسية:

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

الاستعلامات الشائعة

جميع السجلات من مضيف معين:

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

جميع سجلات مكون MME:

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

البحث عن الأخطاء عبر جميع CSCFs:

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

تصفية حسب وحدة systemd:

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

دمج الفلاتر:

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

استعلامات معدل السجل

حجم السجل لكل مكون:

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

معدل الأخطاء لـ CSCF:

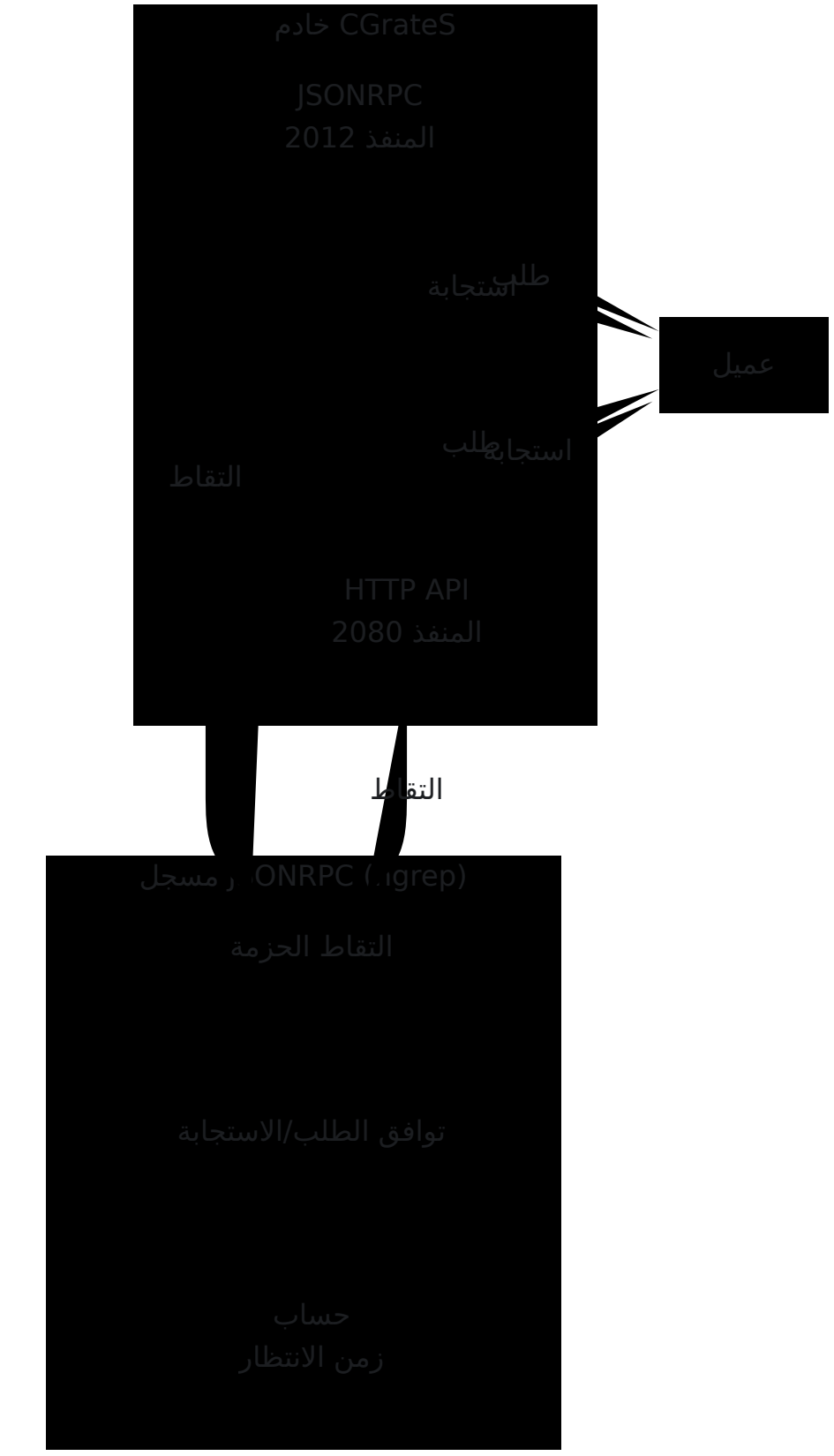
```
sum(rate({component=~"pcscf|scscf|icscf"} |~ "(?i)error" [5m]))
```

تسجيل JSONRPC لـ CGrateS

يتم التقاط حركة مرور JSONRPC /الطلب وتسجيلها مع توافق OCS/CGrateS، بالنسبة لنشر الاستجابة وتتبع زمن الانتظار لتحليل الأداء واستكشاف الأخطاء وإصلاحها.

يتم تعريف مسجل JSONRPC في دور `cgrates` ويتم جمع ملف السجل الخاص به (`/var/log/cgrates/jsonrpc.log`) بواسطة Alloy على كل من مجموعات مضيفي `ocs` و `cgrates`.

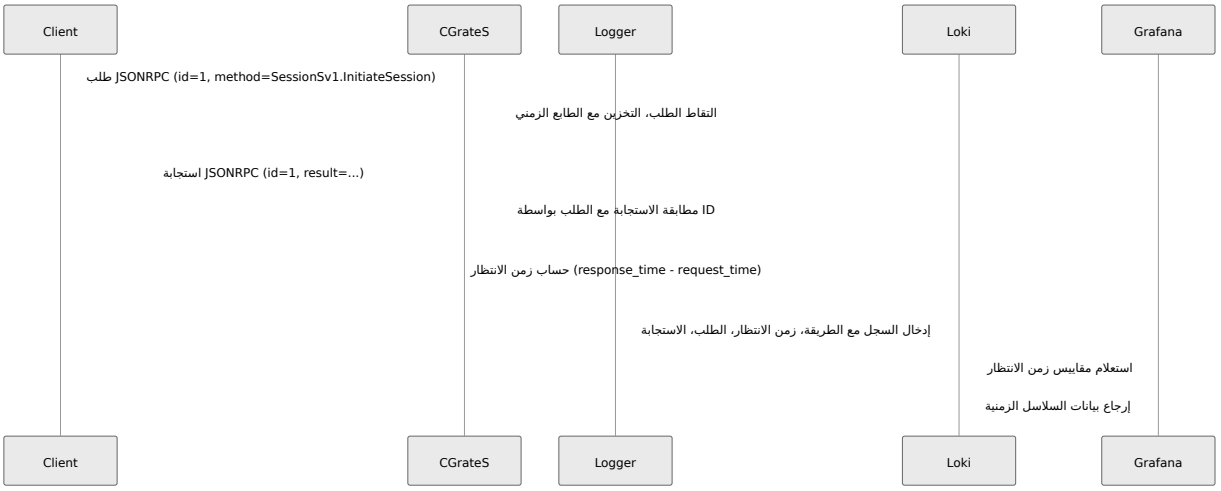
الهيكلية


خادم CGrateS
JSONRPC
المنفذ 2012
طلب
استجابة
عميل
طلب
استجابة
التقاط
HTTP API
المنفذ 2080
التقاط
ONRPC (.igrep) ر مسجل
التقاط الحزمة
توافق الطلب/الاستجابة
حساب
زمن الانتظار
خط أنابيب السجل
/var/log/cgrates/jsonrpc.log





تدفق الطلب/الاستجابة



ما الذي يتم التقاطه

يتم التقاط حركة المرور على هذه المنافذ لـ CGrateS:

المنفذ	اسم الخدمة	البروتوكول	الوصف
2012	`rpc_json`	TCP	JSONRPC الخام عبر TCP
2080	`http`	HTTP	واجهة برمجة التطبيقات HTTP (تستبعد `/health` و `/metrics`)

يقوم المسجل تلقائيًا بتصفية:

- عمليات سحب مقاييس Prometheus (`GET /metrics`)
- طلبات التحقق من الصحة (`GET /health`)
- الاستجابات المضغوطة باستخدام Gzip (بيانات ثنائية)

تنسيق السجل

تكتب سجلات JSONRPC إلى `/var/log/cgrates/jsonrpc.log`. يمثل كل إدخال سجل **زوج الطلب/الاستجابة المكتمل** مع قياس زمن الانتظار:

```
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":{...}}
```

حقول السجل

الحقل	الوصف	المثال
`timestamp`	الطابع الزمني ISO 8601 عند استلام الاستجابة	`2026-03-01T10:30:45.123456`
`port`	منفذ CGrateS الذي تم إجراء الطلب عليه	`2012`, `2080`
`service`	اسم الخدمة للمنفذ	`rpc_json`, `http`
`request_id`	معرف طلب JSONRPC للتوافق	`1`, `42`, (فارغ للنول)
`method`	اسم طريقة JSONRPC	`SessionSv1.InitiateSession`
`latency_ms`	زمن الانتظار في جولات في المللي ثانية	`2.45`
`status`	حالة النتيجة	`ok`, `error`
`error`	رسالة الخطأ إذا كانت الحالة خطأ	`INSUFFICIENT_CREDIT`
`src`	عنوان IP:المنفذ (العميل)	`10.10.1.50:45678`
`dst`	عنوان IP:المنفذ (CGrateS)	`10.10.2.100:2012`
`request`	الحمولة الكاملة لطلب JSONRPC (JSON مضغوط)	`{"id":1,"method":"..."}`
`response`	الحمولة الكاملة لاستجابة JSONRPC (JSON مضغوط)	`{"id":1,"result":{...}}`

لوحة معلومات Grafana

تتضمن لوحة **سجلات OCS / CGrateS** لوحات مخصصة لتحليل JSONRPC.

حسب الطريقة JSONRPC زمن الانتظار لـ

رسم بياني زمني يظهر زمن الانتظار لكل طريقة JSONRPC. مفيد لتحديد بمرور الوقت:

- الطرق البطيئة التي قد تحتاج إلى تحسين
- ارتفاعات زمن الانتظار التي تشير إلى مشاكل في النظام
- اتجاهات الأداء بمرور الوقت

يعرض الأسطورة متوسط وزمن الانتظار الأقصى لكل طريقة.

حركة مرور JSONRPC

لوحة سجلات تعرض JSONRPC مع أزواج الطلب/الاستجابة لـ ��مبع:

- الطابع الزمني
- اسم الطريقة
- زمن الانتظار
- الحمولة الكاملة للطلب والاستجابة (مُزينة)

تعمق في الطريقة

قم بتصفية حركة مرور JSONRPC بواسطة طريقة معينة باستخدام متغير **الطريقة** في أعلى لوحة المعلومات. أدخل اسم الطريقة (على سبيل المثال، `SessionSv1.InitiateSession`) لرؤية حركة المرور فقط لتلك الطريقة.

لوحة استدعاءات CGrateS RPC

توفر لوحة **CGrateS RPC** **استدعاءات** المخصصة أدوات مركزة لاستكشاف أخطاء استدعاءات API وتوافق الطلبات مع الاستجابات.

حالة الاستخدام: العثور على استدعاء فاشل

عندما يبلغ المستخدم "حاولت الاتصال بـ 1234 وفشل":

1. افتح **CGrateS RPC** **لوحات المعلومات** ← **السجلات** ← **استدعاءات**
2. في حقل **البحث**، أدخل رقم الهاتف أو معرف الحساب: `1234`
3. اضبط **الحالة** على **خطأ** لرؤية الاستدعاءات الفاشلة فقط
4. تعرض لوحة **RPC** **بحث استدعاء** جميع الاستدعاءات المطابقة مع الطلب/الاستجابة الكاملة

ما أعادته) يعرض كل إدخال سجل الحمولة الكاملة للطلب (ما تم إرساله) والحمولة الاستجابة CGrateS)، مما يسهل تحديد الخطأ بالضبط.

متغيرات لوحة المعلومات

المتغير	الغرض	المثال
البحث	بحث نصي حر في الحمولة الطلب/الاستجابة	`1234`, `Account123`, `INSUFFICIENT_CREDIT`
الطريقة	تصفية حسب اسم طريقة RPC	`SessionSv1.InitiateSession`, `CDRsV1.ProcessEvent`
الحالة	تصفية حسب حالة النتيجة	الكل، النجاح، الخطأ

لوحات المعلومات

إحصائيات ملخصة (الصف العلوي):

- **إجمالي استدعاءات RPC**: عدد جميع الاستدعاءات في نطاق الوقت
- **الأخطاء**: عدد الاستدعاءات الفاشلة (ملونة: أخضر=0، أصفر=1-9، أحمر=10+)
- **متوسط زمن الانتظار**: متوسط زمن الرحلة (عتبات ملونة)
- **أقصى زمن انتظار**: أعلى استدعاء زمن انتظار

حسب الطريقة: سلسلة زمنية تظهر زمن الانتظار لكل طريقة. مرر **زمن الانتظار لـ RPC** فوق لرؤية القيم المحددة.

بحث استدعاء RPC: لوحة السجل الرئيسية تعرض الاستدعاءات المطابقة. انقر على أي إدخال للتوسيع ورؤية JSON الطلب/الاستجابة الكامل.

الفاشلة: عرض مسبق التصفية يظهر فقط الاستدعاءات ذات **استدعاءات RPC** `status=error`.

تحليل الطريقة: مخططات دائرية تظهر توزيع الاستدعاءات وتوزيع الأخطاء حسب الطريقة.

استعلام سجلات JSONRPC

استعلامات أساسية

جميع حركة مرور JSONRPC:

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

تصفية حسب الطريقة:

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

البحث في الحمولة الطلب/الاستجابة:

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

الأخطاء فقط:

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

تحليل زمن الانتظار

استخراج زمن الانتظار كمقياس (للرسوم البيانية):

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

العثور على الطلبات البطيئة (زمن الانتظار > 100 مللي ثانية):

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

استعلامات محددة للطريقة

معالجة CDR:

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

إدارة الجلسات:

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

عمليات الحساب:

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

إدارة الخدمة

يعمل مسجل JSONRPC كخدمة systemd على مضيفي OCS/CGrateS.

تحقق من حالة الخدمة:

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

عرض سجلات الخدمة:

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

إعادة تشغيل الخدمة:

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

استكشاف الأخطاء وإصلاحها

عدم ظهور سجلات JSONRPC

لا تظهر أي بيانات JSONRPC **الأعراض**: لوحة حركة مرور

الأسباب المحتملة:

- عدم تشغيل خدمة المسجل
- عدم تثبيت ngrep
- عدم تطابق واجهة الشبكة
- عدم جمع Alloy لملف السجل

الحل:

1. تحقق من حالة خدمة المسجل:

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

2. تحقق من تثبيت ngrep:

```
which ngrep
```

3. تحقق من كتابة ملف السجل:

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

4. تحقق من أن Alloy يجمع الملف:

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

الرسم البياني لزمن الانتظار لا يظهر بيانات

لا توجد بيانات JSONRPC" **الأعراض**: لوحة زمن الانتظار لـ "حسب الطريقة تظهر

الأسباب المحتملة:

- لا توجد حركة مرور JSONRPC في نطاق الوقت المحدد
- عدم تطابق تنسيق السجل (لا يتطابق regex)

الحل:

1. تحقق من وجود السجلات لنطاق الوقت:

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

2. تحقق من أن حقل `latency_ms` موجود في السجلات
3. قم بتوسيع نطاق الوقت

عدد الطلبات المعلقة مرتفع

الأعرا: تظهر سجلات التصحيح العديد من الطلبات المخزنة ولكن القليل من الاكتمالات المسجلة

الأسباب المحتملة:

- طلبات بدون استجابات (تم فصل العميل)
- تم التقاط الاستجابة قبل الطلب (ترتيب الحزمة)
- عدم تطابق معرف الطلب بين الطلب والاستجابة

الحل:

1. يقوم المسجل تلقائيًا بتنظيف الطلبات المعلقة التي تزيد عن 60 ثانية
2. تحقق من وجود مشكلات في الشبكة بين العميل و CGrateS
3. تحقق من أن CGrateS تستجيب للطلبات

سجلات الملفات

بالإضافة إلى سجلات journal systemd، يقوم Alloy بجمع ملفات السجل المحددة:

سجلات FreeSWITCH (خوادم التطبيقات)

تسمية الوظيفة	المسار
`freeswitch`	`/var/log/freeswitch/freeswitch.log`
`freeswitch-structured`	`/var/log/omnitas-freeswitch.log`

سجلات Nginx (OmniCRM)

تسمية النوع	تسمية الوظيفة	المسار
`access`	`nginx`	`/var/log/nginx/access.log`
`error`	`nginx`	`/var/log/nginx/error.log`

JSONRPC لـ CGrateS (OCS)

تسمية الوظيفة	المسار
`cgrates-jsonrpc`	`/var/log/cgrates/jsonrpc.log`

استكشاف الأخطاء وإصلاحها

عدم ظهور السجلات في Grafana

الأعراض: لا توجد سجلات مرئية في لوحات Loki

الأسباب المحتملة:

- عدم تشغيل خدمة Alloy على المضيف المصدر
- عدم تشغيل خدمة Loki على مضيف المراقبة
- حظر الاتصال الشبكي بين المضيفين
- عدم قدرة Alloy على الوصول إلى Loki 3100 على المنفذ

الحل:

1. تحقق من حالة Alloy على المضيف المصدر:

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

2. تحقق من حالة Loki على مضيف المراقبة:

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

3. اختبار الاتصال من المصدر إلى المراقبة:

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

4. تحقق من أن جدار الحماية يسمح بـ TCP 3100 من جميع المضيفين إلى المراقبة

استخدام Alloy العالي للقرص

الأعراض: تستهلك مساحة قرص كبيرة `/var/lib/alloy`

ال��سباب المحتملة:

- عدم توفر Loki لفترة طويلة
- امتلاء ذاكرة التخزين المؤقت لـ WAL

الحل:

1. تحقق من اتصال Loki (انظر أعلاه)
2. إذا كان Loki في معطلاً، يتم تخزين السجلات محليًا WAL
3. بمجرد أن يصبح Loki متاحًا، يتم إرسال السجلات المخزنة تلقائيًا
4. يتم إسقاط الأجزاء المخزنة التي تزيد عن 1 ساعة (حسب التصميم)

امتلاء تخزين Loki

الأعراض: يتوقف Loki عن قبول السجلات، القرص ممتلئ على خادم المراقبة

الأسباب المحتملة:

- حجم السجل يتجاوز القرص المتاح قبل أن تحذف فترة الاحتفاظ السجلات القديمة
- عدم تشغيل ضغط الاحتفاظ

الحل:

1. تحقق من استخدام تخزين Loki:

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

2. تحقق من أن الضاغط يعمل (تحقق من سجلات Loki)
3. قم بتشغيل الضغط يدويًا إذا لزم الأمر عن طريق إعادة تشغيل Loki
4. ضع في اعتبارك زيادة تخصيص القرص أو تقليل فترة الاحتفاظ

فقدان تسميات المكونات

الأعراض: تظهر السجلات ولكن تسمية `component` هي `infrastructure` بدلاً من القيمة المتوقعة

الأسباب المحتملة:

- المضيف ليس في مجموعة الجرد المتوقعة
- لم يتم تجديد تكوين Alloy بعد تغيير الجرد

الحل:

1. تحقق من أن المضيف في مجموعة الجرد الصحيحة
2. أعد تشغيل Ansible لتجديد تكوين Alloy:

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

3. أعد تشغيل Alloy:

```
systemctl restart alloy
```

منافذ الخدمة

الخدمة	المنفذ	البروتوكول	الوصف
Loki	3100	HTTP	إدخال السجل وواجهة استعلام API
Loki	9096	gRPC	gRPC داخلي (غير مستخدم خارجيًا)
Alloy	12345	HTTP	واجهة المستخدم Alloy مقاييس
Grafana	3000	HTTP	الوصول إلى لوحة المعلومات

الوثائق ذات الصلة

- المراقبة والملاحظة: Grafana و Prometheus واللوحات والتنبيهات
- هيكلية النشر: الهيكلية العامة للنظام
- تكوين ملف المضيفين: تكوين الجرد
- مرجع التكوين: مرجع كامل للمعلمات

النواة المعتمدة على الدوائر (CS)

يتم نشر مجال الدوائر المعتمد على الدوائر القديمة (2 صوتG/3G و SMS) بواسطة playbook `services/cs.yml`. يقوم بتوفير BSC ومفتاح المواجه للراديو، وبوابة وسائط مستقلة، وإشارة SS7/SIGTRAN عبر STACK وOmniMSC.

نظرة عامة

تشغل `cs.yml` هذه الأدوار، كل منها ضد مجموعة جردها الخاصة:

المكون	الدور	مجموعة الجرد	الحزم	الوصف
BSC	`circuit_switched/bsc`	`bsc`	BSC, MGW	وحدة التحكم في المحطة الأساسية + بوابة الوسائط (لإجراء اختبارات وتحقق 2G/3G)
MGW	`circuit_switched/mgw`	`mgw` (المضيفون ليسوا أيضًا في `bsc`)	`osmo-mgw`	بوابة وسائط مستقلة للمواقع التي لا تحتوي على BSC
MSC	`circuit_switched/omnimsc`	`msc`	`omnimsc`	مفتاح مركز التحويل المحمول
SS7	`omniss7`	`ss7`	`omniss7`	STACK SS7/SIGTRAN (STP، HLR، SMSC، CAMEL، مختبر)

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

BSC (`circuit_switched/bsc`)

يثبت **BSC بوابة الوسائط** و **(MGW)** من خادم Omnitouch APT لاختبار الدوائر والتحقق منها. يقوم الدور بتكوين كلا الخدمتين.

- يقوم بإنشاء ملفات تكوين BSC و MGW من قوالب الدور.
- يقوم بتثبيت تجاوز systemd لخدمة MGW **التي تعطل جدولة وحدة المعالجة المركزية في الوقت الحقيقي**. يقوم بإزالة `CPUSchedulingPolicy` و `Priority` و `AmbientCapabilities`. لا تدعم الآلات الافتراضية جدولة الوقت الحقيقي.
- يقوم المعالجات بإعادة تشغيل كل خدمة. ثم **يتحقق كل معالج من أن الخدمة تصل إلى** `active` (6 محاولات، 5 ثوانٍ بين كل محاولة) قبل أن يستمر.

MGW (`circuit_switched/mgw`)

يثبت **بوابة الوسائط** المستقلة (`osmo-mgw`) من خادم Omnitouch APT. استخدمها في موقع MSC يحتوي على CS على MGW بتثبيت وسائط كل مكالمة ولكن لا يحتوي على BSC. يقوم MSC عبر MGCP.

- يقوم بإنشاء تكوين MGW. يرتبط MGCP بـ `mgw_bind_ip` على `mgw_port`. يرتبط RTP بـ `mgw_rtp_bind_ip` ويستخدم `mgw_rtp_port_range`. كلا العنوانين للمضيف. المنفذ الافتراضي هو 2427 والنطاق هو `ansible_host` الافتراضيين هما `16001 4002`.
- يقوم بتثبيت نفس تجاوز systemd مثل دور BSC. يقوم التجاوز بتعطيل جدولة وحدة المعالجة المركزية في الوقت الحقيقي، والتي لا تدعمها LXCs و VMs.
- يقوم المعالج بإعادة تشغيل الخدمة ثم يقوم بتشغيل فحص `verify_service` للتحقق من حدوث تعطل.
- يقوم بالفعل بتشغيل `bsc` المضيف الذي هو أيضًا في مجموعة BSC. من دور MGW هذا الدور على مثل هذا المضيف playbooks تتخطى.

MSC (`circuit_switched/omnimsc`)

يثبت حزمة `omnimsc` ويقوم بتكوينها مثل NF 5GC:

- يحل `group_vars_omnicore` ويشتق `license_server_api_urls` من مجموعة `license_server`.
- يقوم بإنشاء `runtime.exs.j2` المدمج إلى `/etc/omnimsc/`. إذا تم تعيين `omnimsc_template_config`، فإنه يقوم بإنشاء قالب مخصص من `group_vars/` بدلاً من ذلك.
- يضمن وجود دليل إخراج CDR (`omnimsc_cdr_output_dir`، الافتراضي `/var/lib/omnimsc/cdr`).
- ينشئ شهادة TLS موقعة ذاتيًا لواجهة الويب / API.
- يستخدم نمط الارتباط + `runtime.exs` المرتبط بالإصدار، مع إعادة التشغيل عند التغيير.
- يبني قائمة بوابات MGCP بهذا الترتيب:
  i. `omnimsc_mgcp_gateways`، إذا تم تعيينها (تجاوز صريح).
  ii. بوابة واحدة لكل مضيف في مجموعة `mgw` (العنوان `ansible_host`، المنفذ `mgw_port`، الافتراضي 2427).
  iii. بوابة واحدة لكل مضيف في مجموعة `bsc` (دور BSC يضع MGW في نفس المكان).

  إذا لم يكن أي من هذه موجودًا، يفشل الدور قبل تثبيت أي شيء. بوابة الوسائط المستخدمة هي `omnimsc_media_gateway` (الافتراضي `MGW-01`، أول بوابة في القائمة).

SS7 (`omniss7`)

يتم اختياره، **SS7** وينشئ **قالب تكوين واحد لكل دور** `omniss7` دور واحد يقوم بتثبيت حزمة `ss7_role` بواسطة متغير المضيف:

`ss7_role`	القالب	الوظيفة
`stp`	`stp.j2`	نقطة نقل الإشارة (SS7 توجيه)
`hlr`	`hlr.j2`	سجل الموقع المنزلي
`smsc`	`ipsmgw.j2`	SMSC / IP-SM-GW
`camel`	`camel.j2`	التحكم في خدمة CAMEL
`tester`	`tester.j2`	وكيل اختبار/تحميل الإشارة

محددات إضافية:

- تعيين `ss7_protocol: m2pa` يقوم بإنشاء `m2pa.j2` (نقطة نهاية M2PA) بدلاً من قالب `tester`.
- تعيين `ss7_template_config` (مسار تحت `group_vars/`) يتجاوز **جميع** القوالب المدمجة بتكوين محدد للمضيف.

مثل المكونات الأخرى في Elixir، يستخدم `omniss7` نمط الارتباط + `runtime.exs` المرتبط بالإصدار ويعكس الارتباط في `/opt/omniss7/releases/*`.

شحن CAMEL

من خلال CAMEL، بواسطة بوابة CAMEL لمشترك لديه بيانات CS تُفرض رسوم على مكالمات نفس OCS وtenant (`ocs_tenant`) الذي يستخدمه TAS لـ Ro:

متغير المضيف	المضيف	الوظيفة
`camel_gsmscf_global_title`	`ss7_role: camel`	العنوان العالمي gsmSCF لبوابة CAMEL. يضيف `stp.j2` مسار GT إليه، ويستخدمه `camel.j2` كـ GT المتصل لردود CAP.
`hlr_camel_subscriptions`	`ss7_role: hlr`	قائمة بقواعد `imsi_prefix` مع `t_csi` و/أو `o_csi` (`service_key`، `gsmscf_address`، `default_call_handling`: `release_call` أو `continue_call`)، و `mo_sms_csi` اختياري (`service_key`، `gsmscf_address`، `default_sms_handling`: `release_transaction` أو `continue_transaction`). يرسل HLR هذه إلى VLR في InsertSubscriberData. يرسل MSC InitialDP لمكالمة، أو InitialDPSMS لـ MO SMS، إلى `gsmscf_address`.

تدفق جدار الحماية، `ocs` (`:2080/jsonrpc`) إلى البوابة في أول مضيف `camel.j2` يشير MO ويشحن كل، TAS يقوم بشحن المكالمات في فترات زمنية مدتها 10 ثوانٍ، كما يفعل (31b. بتعيين متغير ran_simulator واحدة. عندما يتم تعيين كلا المتغيرين، يقوم دور `*sms` كوحدة SMS السيناريو `camel_origin_host`، مما يمكّن إعدادات شحن CAMEL.

USSD (`support/ussd_app`)

تذهب USSD المرسلة من الهاتف المحمول من MSC إلى HLR كـ MAP `processUnstructuredSS-Request`. عندما يحتوي الجرد على مجموعة `ussd`، يقوم

`hlr.j2` شغيل بوابة ب��omniss7 USSD إلى (`*`) مع مسار شامل واحد `http://<first ussd host>:8380/ussd`. بدون المجموعة، لا يتم تكوين البوابة.

بتثبيت تطبيق رد صغير (playbook `services/ussd.yml`) يقوم دور `support/ussd_app` يسجل سطرًا واحدًا لكل طلب إلى `ussd-app`. كخدمة systemd على مضيفات `ussd` journald.

الاتصال / الرد	الإجابة
`*100#`	النهاية: `رصيدك هو $12.34`
`*102* <digits>#`	النهاية: `أنت أرسلت <digits>`
`*101#`	الاستمرار: `1. الرصيد` / `2. البيانات` / `3. الخروج`
الرد `1`، `2`، `3`	النهاية: `الرصيد: $12.34`، `البيانات: 1.5 جيجا بايت متبقية`، `وداعًا`
أي رد آخر	النهاية: `خيار غير صالح`
أي رمز آخر	النهاية: `رمز غير معروف <code>`

بقراءته من `roles/omniss7` ينتمي إلى الدور. يقوم (8380، `ussd_app_port`) المنفذ للدور. تقوم `firewall_services.yml` إعدادات الدور الافتراضية، ويعلن جدار الحماية عنه في بنشر كل حالة أعلاه إلى التطبيق وتفشل إذا كانت الإجابة خاطئة playbook.

الوثائق ذات الصلة

- الخدمة وهيكل النشر playbooks - جميع Playbooks الخدمة
- المشترك `gc_nf` ودور 5 النواة المستقلة IG - 5 النواةIG 5
- قوالب تكوين مخصصة - تكوين المتغيرات الجماعية (مثل `ss7_template_config`، `omnimsc_template_config`)
- **تكوين OmniCore**: https://docs.omnitouch.com.au/docs/repos/OmniCore - ما يجب أن يحتويه ملفات تكوين المكونات

مرجع التكوين

نظرة عامة

يوفر هذا المستند مرجعًا شاملاً لتكوين نشرات OmniCore. يتم تعريف التكوين بشكل أساسي في ملفات جرد المضيفين مع الحد الأدنى من تجاوزات group_vars المطلوبة للنشرات الحديثة.

للوثائق الخاصة بالمنتج، انظر:

- **OmniCore**: https://docs.omnitouch.com.au/docs/repos/OmniCore
- **OmniCall**: https://docs.omnitouch.com.au/docs/repos/OmniCall
- **OmniCharge**: https://docs.omnitouch.com.au/docs/repos/OmniCharge

نهج التكوين

تستخدم نشرات OmniCore الحديثة نموذج تكوين مبسط:



المبدأ الأساسي: يتم تعريف معظم التكوين مباشرة في ملف المضيفين. تتعامل الافتراضات الخاصة بالدور مع العديد من الإعدادات. ومع ذلك، يتم تكوين بعض المكونات حصريًا من `group_vars` هو المثال البارز (انظر أدناه). في تلك OmniCRM. وليس لديها افتراضات دور الحالات، فإن القيمة المفقودة تفشل التشغيل بصوت عالٍ بدلاً من العودة إلى الافتراضي.

> **ملاحظة أمنية**: العديد من الافتراضات الخاصة بالدور هي عناصر غير آمنة مخصصة للاستخدام في المختبر فقط ويجب **تجاوزها** لأي نشر حقيقي. تشمل الأمثلة كلمات مرور وبيانات اعتماد مستودع (مثل قواعد البيانات `sql_pass: password`، `omnitouch2024`) group_vars تأكد دائمًا من تعيين أسرار حقيقية وفريدة في جردك أو APT (`omni`/`omni`). قبل النشر.

تخطيط الشبكة

للحصول على إرشادات حول IP **قبل تكوين المضيفين**، راجع معيار تخطيط:

- استراتيجيات تقسيم الشبكة
- تخصيص عنوان IP
- تنظيم الشبكة الفرعية
- التعامل مع IP العامة

معلمات المضيف الشائعة

للحصول على هذا hosts-file-configuration.md فقط قل للتحقق من - #ToDo

علامات الخدمة المحددة

```
cdrs_enabled: True                  # تمكين توليد CDR
in_pool: False                      # استبعاد من مجموعة التوازن
dra_peer_exclude: True              # استبعاد من قوائم الأقران Diameter
المولدة
gtpc_weight: 100                    # حصة SGW/PGW ضمن مستوى أولويته
(اختيار نظير MME)
gtpc_priority: 100                  # مستوى أولوية SGW/PGW، كلما كان أقل
كان مفضلًا
```

انظر استبعاد نظير Diameter لـ `dra_peer_exclude`.

عندما يقوم MME بإنشاء قائمة نظير GTP-C لذلك يمكن أن يكون النظير `100`، كلاهما افتراضي إلى `gtpc_priority` و `gtpc_weight` يتم قراءة من كل مضيف `sgwc` / `pgwc` الثابتة الخاصة به. مستوى MME مرجحًا أو مصنفًا فوق أو تحت بقية المجموعة دون تغيير كل مضيف آخر. يختار `gtpc_priority` عدديًا المتاح ويقوم بتقسيم الحركة داخله حسب *الأدنى/* `gtpc_weight` (RFC 2782)؛ المستويات الأعلى هي احتياطية وتأخذ الحركة فقط عندما لا يبقى أي نظير في مستوى أدنى. يتم تعيين سعة MME النسبية بنفس الطريقة، مع `relative_capacity` على مضيف MME (النطاق 0-255، افتراضي `100`).

المتغيرات العالمية (all:vars)

يحتوي قسم `all:vars` على إعدادات على مستوى النشر. تستخدم النشرات الحديثة متغيرات عالمية قليلة، مع تغطية العديد من الإعدادات بواسطة الافتراضات الخاصة بالدور. لاحظ الاستثناءات أدناه (OmniCRM، Grafana SMTP) حيث يجب تقديم التكوين بشكل صريح لأنه لا يوجد افتراض دور.

المتغيرات العالمية الأساسية

المصادقة والوصول

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

بدلاً من كلمات المرور SSH **بديل**: استخدم مفاتيح:

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

هوية العميل

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

تكوين PLMN

```
plmn_id:
  mcc: '001'                # رمز الدولة المحمول (3 أرقام)
  mnc: '01'                 # رمز الشبكة المحمولة (2-3 أرقام)
  mnc_longform: '001'       # MNC مع حشو بالأصفار (دائمًا 3 أرقام)

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

الغرض: يحدد شبكة الهاتف المحمول الخاصة بك بشكل فريد. يستخدم لبناء نطاق Diameter.

أسماء الشبكة

```
network_name_short: Omni
network_name_long: Omnitouch
tac_list: [10100,100]             # قائمة TAC الافتراضية (يمكن تجاوزها لكل
MME)
```

المعروض: أسماء الشبكة المعروضة على أجهزة UE المحمولة < الشبكة في الإعدادات.

تكوين DNS

```
manage_resolv_conf: True          # تعيينه إلى False لمنع Ansible من
إدارة /etc/resolv.conf
```

ملاحظة: عندما يتم تعيين `manage_resolv_conf` إلى `False`، لن يقوم Ansible بكتابة `/etc/resolv.conf` على ذلك المضيف. هذا مفيد للمضيفين الذين يحتاجون إلى تكوين DNS مخصص أو يتم إدارتهم بواسطة أنظمة خارجية. يمكن ت��يينه لكل مضيف في الجرد أو عالميًا في `all:vars`.

تكوين مستودع APT

هو المصدر الوحيد للحقيقة. قم بتعريفه مرة واحدة. يتم اشتقاق متغيرات تنزيل `apt_repo` منه تلقائيًا. لا تحتاج إلى تعيينها `apt_download_base` وعنوان URL (`remote_apt_*`) الثنائيات بشكل منفصل في جرد عادي.

الافتراضات التلقائية: عندما يتم تعريف مجموعة `apt_cache_servers` مع المضيفين:

- `use_apt_cache` افتراضيًا إلى `True` (ما لم يتم تعيينه صراحة إلى `False`)

- لأحد خوادم التخزين المؤقت الأولى IP افتراضيًا إلى عنوان `apt_repo.apt_server`

```
use_apt_cache: False              # الوصول المباشر إلى المستودع (True =
التخزين المؤقت المحلي)

apt_repo:
  apt_server: "apt.<region>.omnitou.ch"  # FQDN المقدم من Omnitouch
(أو IP التخزين المؤقت)
  apt_repo_username: "your-username"     # مطلوب للوصول المباشر
  apt_repo_password: "your-password"     # مطلوب للوصول المباشر
  # apt_repo_port: 443                   # 443 هو الافتراضي؛ اختياري
(HTTPS).
                                         # 443 :ينبع المخطط المنفذ ->
https، وإلا http.
```

في وضع **HTTP / 8080** للوصول المباشر و **HTTPS / 443** المخطط والمنفذ افتراضيان إلى (`apt_cache_servers` group) يقوم بتعيين التخزين المؤقت. فقط خادم مرآة التخزين المؤقت `remote_apt_*` يدويًا، لوصف خادم Omnitouch العلوي الذي يتزامن منه.

انظر: نظام تخزين APT

خادم الترخيص

```
# عند عدم تعيينه. `license_server` :اختياري يتم اشتقاقه تلقائيًا من مجموعة جرد
# يصبح كل مضيف في تلك المجموعة https://<ansible_host>:8443/api.
license_server_api_urls: ["https://10.10.2.150:8443/api"]
```

ملاحظة: إذا كانت مجموعة `license_server` موجودة في الجرد، يتم اشتقاق `license_server_api_urls` تلقائيًا من أعضائها؛ تحتاج فقط إلى تعيينه يدويًا لتجاوز هذا السلوك.

انظر: خادم الترخيص

إعدادات MME

باستخدام طريقة قابلة للتكوين، يتم تعيينها لكل بوابة تحت P-GW و S-GW أقران MME يختار `omnimme`. ي��كن أن تكون الطريقة قيمة واحدة أو مقسمة حسب حركة المرور `home`/`roaming`. القيم الصالحة هي `dns_lookup` (حل الأقران عبر DNS) و `random_peer` (اختيار من الأقران المكونة).

```yaml
omnimme:
  sgw:
    selection_method: random_peer        # أو "قيمة واحدة"...
    # selection_method:
    #   home: random_peer
    #   roaming: dns_lookup
  pgw:
    selection_method:
      home: random_peer
      roaming: dns_lookup
```

إعدادات AMF

```yaml
# قائمة tac_list تدفع أيضًا قائمة supported_ta_list الخاصة بـ AMF. كل TAC هنا يتم
# الإعلان عنه للأجهزة gNB المتصلة أثناء إعداد NG. إذا لم يتم تعيينه، فإن AMF الافتراضي
# هو TAC 1 فقط، مما يرفض أي gNB لا يبث TAC 1.
tac_list: [1, 3]

# تعطيل تشفير NAS (فرض NEA0). الافتراضي هو true لأن omniamf المشفر (إكمال NAS ولكن لا يمكنها فك تشفير NEA2 تتفاوض على >=20260727 وضع الأمان
# 5 والرسائل اللاحقة)، مما يتسبب في فشل كل تسجيل G. تعيينه إلى false بمجرد في الاتجاه الصاعد يعمل NEA2 أن يؤكد البائع أن فك تشفير
# ضد UEs الحقيقية NAS (NIA2). لا تتأثر سلامة
amf_force_null_ciphering: true
```

إعدادات SAEGW

```yaml
mtu: 1400                          # الحد الأقصى لوحدة النقل
```

إعدادات OmniHSS

يبني OmniHSS قائمة أقرانه Diameter تلقائيًا من الجرد. يستخدم مجموعات `mme` و `dra` (أي من تلك المستخدمة يعتمد `applicationserver` و `pcscf` و `icscf` و `scscf` و `pgwc` على `ims_dra_support` / `packet_core_dra_support`). يقوم `exclude_diameter_peers` بإسقاط مضيفين فرديين من تلك القائمة المولدة، للأقران الموجودة في الجرد ولكن لا ينبغي أن تكون أقران Diameter لـ HSS.

```
omnihss:
  exclude_diameter_peers:          # اختياري؛ الافتراضي هو [] (لا شيء مستبعد)
    - customer-as01
    - customer-as02
```

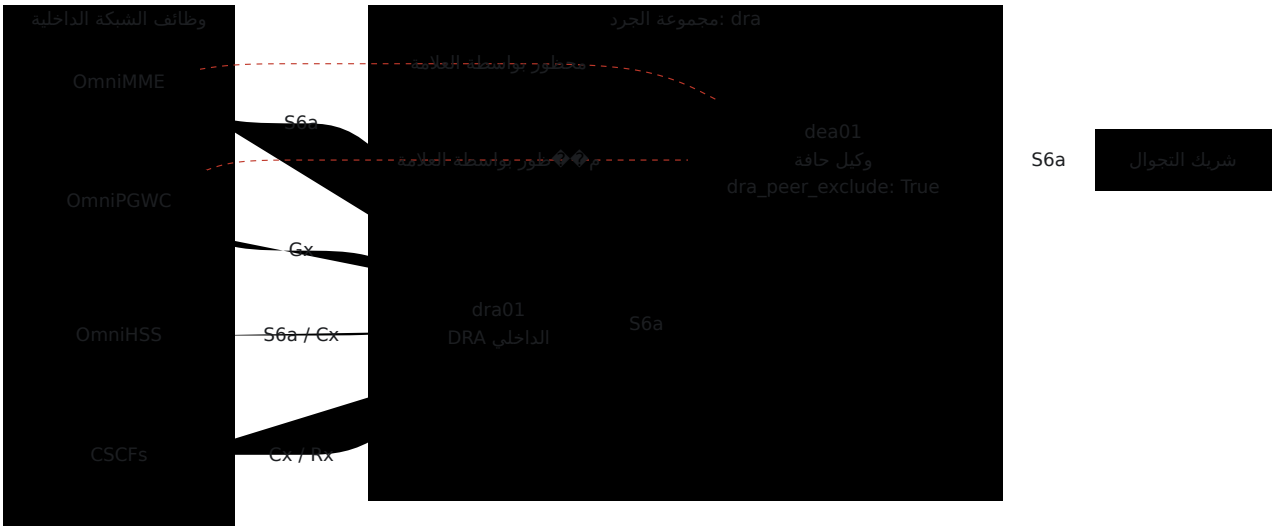
- يتم مطابقة الإدخالات ضد **`inventory_hostname`**، وليس FQDN أو IP Diameter.
- ينطبق الاستبعاد على كل مجموعة يتم اكتشافها تلقائيًا، لذا تغطي قائمة واحدة مضيفًا بغض النظر عن المجموعة التي يجلس فيها.
- يجب أن توجد الأسماء في الجرد. تؤكد الد��ر على ذلك قبل النشر، لذا فإن خطأ مطبعي يفشل التشغيل بصوت عالٍ بدلاً من ترك النظير في `/etc/omnihss/runtime.exs` بصمت.
- قم بتعيينه حيث يتم تعيين بقية قاموس `omnihss` (أو تحت `vars:` لكل مضيف، لأن Ansible يستبدل القواميس بدلاً من دمجها، يجب أن تعيش في نفس كتلة `omnihss` نظرًا لأن المجموعة). مثل المفاتيح الأخرى.
- يؤثر هذا فقط على قائمة أقران Diameter. قائمة أقران S-CSCF SIP تحت `config :hss, ims:` غير مرتبطة ولا يتم تصفيتها بواسطة هذه المفتاح.

> `exclude_diameter_peers` هو مفتاح من قاموس `omnihss`. يقوم بتصفية قائمة أقران OmniHSS فقط. لإزالة مضيف من قائمة الأقران لكل وظيفة شبكة، استخدم `dra_peer_exclude` بدلاً من ذلك. الآليتان مستقلتان.

استبعاد نظير Diameter

يبني OmniCore قائمة أقران Diameter لكل وظيفة شبكة من مجموعات الجرد. تضيف وظيفة `dra_peer_exclude` بإزالة مضيف الشبكة كل مضيف من المجموعات التي تتواصل معها. يقوم واحد من تلك القوا��م المولدة. قم بتعيين العلامة على المضيف الذي لا يجب أن يصبح نظير Diameter.

الحالة الرئيسية هي **وكيل حافة Diameter (DEA)**. DEA هو عضو في مجموعة `dra`، لأنه يحتاج إلى نفس النشر، جدار الحماية ومعالجة المراقبة. يحمل OmniDRA يشغل نفس برنامج الحركة إلى الشركاء الخارجيين فقط. يجب على كل وظيفة شبكة داخلية إرسال حركتها إلى DEA خارج قائمة DEA لإبقاء DEA على كل `dra_peer_exclude: True` الداخلي. قم بتعيين DRA الأقران لوظائف الشبكة الداخلية.



التكوين

قم بتعيين العلامة على إدخال المضيف نفسه، بجوار `ansible_host`.

```
dra:
  hosts:
    dra01:                  # DRA نظير لكل وظيفة شبكة داخلية .الداخلي.
      ansible_host: 10.4.2.150
    dea01:                  # وكيل الحافة . لا يكون أبدًا نظيرًا لوظيفة شبكة
داخلية.
      ansible_host: 203.0.113.10
      packet_core_dra_support: False
      ims_dra_support: False
      dra_peer_exclude: True
```

المعلمة	النوع	مطلوب	الافتراضي	سف
`ansible_host`	سلسلة	نعم	-	ضًا عنوان IP عنوان ، الأقران المولدة IP
`dra_peer_exclude`	منطقي	لا	`False`	`True`، عندما يكون عند بناء قائمة أقران ة الجرد Diameter
`packet_core_dra_support`	منطقي	لا	`False`	`True`، عندما يكون واة الحزم DRA مع Gx). عندما يكون `F` لمضيف مباشرة مع ينه إلى DEA، على مساراتها من DEA `dra_module_adv`
`ims_dra_support`	منطقي	لا	`False`	`True`، عندما يكون I لواجهات DRA مع ينه إلى DEA، على

النطاق

تعمل العلامة في اتجاهين. تتخطى وظيفة الشبكة مضيفًا مستبعدًا من مجموعة `dra`، ويتخطى مضيفًا مستبعدًا من أي مجموعة أخرى DRA.

وظائف الشبكة التي تصفي مجموعة `dra`:

الدور	قائمة الأقران
`omnimme`	أقران T6a و S6a
`omnipgwc`	أقران Gy و Gx
`omnihss`	أقران Sh و Cx و S6a
`omniepdg`	أقران S6b و SWm
`omnitwag`	أقران S6b
`omniscef`	أقران T6a
`omnisep`	أقران Sh
`omnilcs`	أقران SLh و SLg
`cscf`	أقران Cx و Rx و Dx لـ P-CSCF و I-CSCF و S-CSCF
SMSC و TAS (موقع `group_vars`)	أقران Sh و Ro

المجموعات التي يقوم DRA بتصفية: `hss` و `mme` و `pgwc` و `epdg` و `scef` و `twag` و `ocs` و `omnilcs`، `omnimessage` و `pcscf` و `icscf` و `scscf` و `xcap` و `applicationserver`.

قواعد السلوك

- قم بتعيين العلامة على المضيف للاستبعاد. لا تقم بتعيينها على المضيف الذي يبني قائمة الأقران.
- تنطبق العلامة فقط على قوائم الأقران المولدة. لا يتم تصفية النظير الثابت الصريح أبدًا. يغطي هذا وظيفة `diameter_static_peers` و DRA على `dra_peers` الشبكة.
- لا تغير العلامة قواعد جدار الحماية، أو خطة Prometheus، أو أهداف IP. يحتفظ المضيف المستبعد بوصوله إلى الشبكة ومراقبته.

- تدخل العلامة حيز التنفيذ في التشغيل التالي للدور. قم بتشغيل دور كل وظيفة شبكة المنفذ على ذلك المضيف بالنظير `runtime.exs` متأثرة بعد تعيينك العلامة. يحتفظ القديم حتى ذلك الحين.

أعراض الاستبعاد المفقود

تحقق من هذه العلامة أولاً. الأعراض هي، إذا اتصلت وظيفة شبكة داخلية بـ DEA:

- يفتح MME أو PGW-C اتصال SCTP إلى عنوان IP العام لـ DEA.
- يسجل DEA محاولة اتصال نظير غير مصرح بها، لأن `allow_undefined_peers_to_connect` هو `False`.
- تبلغ وظيفة الشبكة عن نظير في حلقة إعادة الاتصال الدائمة.

لتصحيح ذلك، قم بتعيين `dra_peer_exclude: True` على مضيف DEA. ثم قم بتشغيل دور المنفذ تحتوي على `runtime.exs` وظيفة الشبكة مرة أخرى. ثم تأكد من أن قائمة الأقران في DRAs الداخلية فقط.

إعدادات OmniCRM

يتم الحصول على تكوين OmniCRM من **حصريًا** `group_vars` الخاصة بالعميل (تحت `hosts/`); لا توجد افتراضات مدمجة في الدور لأسرار الموقع.

يقوم دور `omnicrm` بنشر OmniCRMv2 من المصدر: يقوم المتحكم بشحن حزمة من تفريغ OmniCRMv2، وواجهة المستخدم، ويقوم الدور بتهيئة venv يقوم المضيف ببناء، ومزامنة `/etc/omnicrm/crm_config.yaml` إلى `crm_config.yaml`، وتشغيل وحدات `omnicrm-v2-*` خلف nginx. يأتي تصميم واجهة المستخدم من كتلة `ui:` (runtime `GET /api/public/ui-config`).

```
crm_config.yaml                  # API + ui: config (group_vars; required)
deployment_assets/               # شعارات / أيقونات اختيارية لـ /etc/omnicrm/deployment_assets
```

إعدادات Grafana SMTP

تستخدمها دور المراقبة لإرسال بريد إلكتروني لتنبيه Grafana. `grafana_smtp_user` و `grafana_smtp_password` هما **مطلوبان** (لا يوجد افتراضي: يجب تعيينهما)؛ `grafana_smtp_host` افتراضيًا إلى `in-v3.mailjet.com:587`. انظر `roles/monitoring/templates/grafana/grafana.ini`.

```
grafana_smtp_user: "smtp-user"           # مطلوب، لا يوجد افتراضي
grafana_smtp_password: "smtp-password"   # مطلوب، لا يوجد افتراضي
grafana_smtp_host: "in-v3.mailjet.com:587"  # اختياري، هذا هو الافتراضي
```

إعدادات IMS

```
ims_dra_support: False           # توجيه IMS عبر DRA
sipcapture_client_enabled: False # إرسال SIP/RTP إلى صندوق التقاط
VoIPmonitor (مجموعة siptrace)
```

تكوين مراقب RAN

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

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

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

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

تكوين جدار الحماية

تم توثيق مجموعة المعلمات الكاملة، والمناطق الطائرة، وأنماط المراقبة / التنفيذ في جدار الحماية حسب الدور؛ فقط `allowed_oam_subnets` هو إلزامي. يتم تطبيقه بواسطة `services/firewall.yml`.

```
firewall:
  allowed_oam_subnets:            # مصدر الإدارة (SSH، واجهة المستخدم، سحب، واجهات برمجة التطبيقات الإدارية)
    - '10.0.1.0/24'
    - '10.0.0.0/24'
  allowed_ran_subnets: []         # S1AP / NGAP / GTP-U / Abis
  allowed_ue_voice_subnets:       # SIP UE الخارجي إلى P-CSCF
    - '10.0.1.0/24'
  allowed_carrier_voice_subnets:  # SIP الربط بين الناقلين
    - '10.0.1.0/24'
  zones:                          # المناطق الطائرة — انظر docs/firewall.md
    internal: ['10.0.0.0/8']
  ue_pools: []
```

خوادم DNS للتجوال

```
roaming_dns_servers:
  wildcard: ['10.0.99.1']
  # DNS محدد للناقل (استنادًا إلى PLMN)
  123456:                         # 1 مثال الناقل
    - '10.10.2.197'
  654321:                         # 2 مثال الناقل
    - '10.10.0.4'
```

المستخدمون المحليون (مفاتيح SSH)

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

تكوين Hypervisor

Proxmox

يتم نشر الموقع كـ VMs أو حاويات LXC، يتم اختيارها لكل إدخال عبر `proxmoxServers`. يجب أن تتفق جميع الإدخالات في الموقع؛ يؤدي الخلط إلى فشل التحقق. انظر نشر Proxmox للحصول على الشرح الكامل.

موقع VM

```
proxmoxServers:
  customer-prxmx01:
    # تم حذف deployment_type → هو الافتراضي "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        # اسم مستخدم cloud-init اختياري؛ الافتراضي هو أول مفتاح local_users
    proxmoxTemplatePassword: omnitouch    # كلمة مرور cloud-init اختيارية؛ الافتراضي هو اسم مستخدم cloud-init
    storage: SSD_RAID0            # اختياري، التخزين الافتراضي للـ VM
```

موقع LXC

```yaml
proxmox_lxc_nameserver: "1.1.1.1"  # عند LXCs اختياري، يتم حقنه في الإنشاء

proxmoxServers:
  customer-prxmx01:
    deployment_type: lxc
    proxmoxServerAddress: 10.10.0.100
    proxmoxServerPort: 8006
    proxmoxApiTokenName: AnsibleToken
    proxmoxApiTokenSecret: "token-secret"
    proxmoxNodeName: pve01
    proxmoxLxcOsTemplate: "local:vztmpl/ubuntu-24.04-standard_24.04-2_amd64.tar.zst"
    proxmoxLxcDefaultStorage: SSD_RAID0   # اختياري، النسخة الاحتياطية لجذر النظام
```

تجاوزات على مستوى المجموعة (كلا النوعين)

```yaml
dns:
  vars:
    proxmox_interface: vmbr0              # مطلوب: الجسر
    gateway: 10.10.0.1                    # مطلوب
    netmask: 255.255.255.0                # مطلوب
    vlanid: 100                           # اختياري
    proxmoxLxcCores: 2                    # اختياري (LXC فقط)
    proxmoxLxcMemoryMb: 4096              # اختياري (LXC فقط)
    proxmoxLxcDiskSizeGb: 30              # اختياري (LXC فقط)
    proxmoxLxcRootFsStorageName: SSD_RAID0 # اختياري (LXC فقط)
    host_vm_network: vmbr1                # تجاوز الجسر الاختياري
```

VMware vCenter

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

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

netmask: 255.255.255.0
```

الوثائق ذات الصلة

- معيار تخطيط IP - إرشادات بنية الشبكة وتخصيص IP
- تكوين ملف المضيفين - كيفية هيكلة ملفات المضيفين
- تكوين متغيرات المجموعة group_vars - متى وكيفية استخدام
- تكوين Netplan - عناوين IP متعددة وإعدادات NIC الثانوية
- بنية النشر - كيفية تفاعل المكونات
- نظام ذاكرة التخزين المؤقت APT - إدارة الحزم
- خادم الترخيص - تكوين الترخيص

وثائق المنتج

للحصول على أدلة تشغيل مفصلة وتكوين متقدم:

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

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

نظرة عامة على بنية النشر

نظرة عامة

يوفر هذا المستند عرضًا كاملاً لكيفية نشر برنامج الشبكة الخلوية لخدمات Omnitouch باستخدام Ansible، 4 موضحًا كيف تتناسب جميع المكونات معًا لإنشاء شبكة G/5G عاملة.

للحصول على تفاصيل حول موضع المكونات، وإرشادات تخصيص عنوان IP، راجع معيار تخطيط IP العامة ومعالجة IP.

مثال كامل على النشر

0. توفير البنية التحتية (اختياري)

قبل التكوين VMs/LXCs، قم بتوفير Proxmox بالنسبة لنشر:

```
# نشر VMs و/أو حاويات LXC على Proxmox (نوع النشر لكل مضيف يحدد
أيهما)
ansible-playbook -i hosts/Customer/hosts.yml
util_playbooks/proxmox.yml
```

راجع: نشر VM/LXC على Proxmox

1. تعريف البنية التحتية (ملف ا��مضيفين)

```yaml
# تعريف ما يجب نشره وأين
mme:
  hosts:
    customer-mme01:
      ansible_host: 10.10.1.15

hss:
  hosts:
    customer-hss01:
      ansible_host: 10.10.2.140

# ... جميع المكونات الأخرى
```

راجع: تكوين ملف المضيفين

2. التخصيص (group_vars)

يحتفظ مجلد `group_vars` بأي تجاوزات تكوين مطلوبة على مستوى المضيف أو الموقع أو الشبكة.

على سبيل المثال، يحتوي مجلد على تكوين OmniMessage SMSc الخاص بك. تعيش خطوط Diameter هنا أيضًا. كما تعيش جميع منطق توجيه TAS التي يتصل بها SIP هنا أيضًا.

تم تكوين OmniCRM هنا أيضًا. إعداداته خاصة بـ `group_vars` فقط. يتم استنساخ مصدره من GitHub على وحدة التحكم Ansible (`delegate_to: localhost`) باستخدام بيانات اعتماد وحدة التحكم الخاصة بها قبل مزامنتها مع الهدف، لذا لا تصل أي بيانات اعتماد أبدًا إلى المضيف الهدف.

راجع: تكوين المتغيرات الجماعية

3. توزيع الحزم (APT Cache)

```yaml
# تكوين مكان الحصول على الحزم
apt_repo:
  apt_server: "10.254.10.223"  # عنوان IP لخادم التخزين المؤقت أو خادم المستودع المباشر
use_apt_cache: false  # true = استخدام التخزين المؤقت المحلي، false = الوصول المباشر إلى المستودع
```

راجع: نظام APT Cache

4. تكوين الترخيص

```yaml
# توجيه المكونات إلى خادم الترخيص
license_server_api_urls: ["https://10.10.2.150:8443/api"]
```

راجع: خادم الترخيص

5. تنفيذ النشر

يمكنك نشر مكونات فردية. على سبيل المثال، قم بتشغيل `services/twag.yml` لتعامل مع كل شيء. استخدم `services/all.yml` أو `--limit=myhost` لتحديد المضيفين الذين تعمل عليهم مجموعة التشغيل `--limit=mmee,sgw`.

```
# نشر الشبكة الكاملة
ansible-playbook -i hosts/customer/host_files/production.yml
services/all.yml

# أو نشر مكونات محددة
ansible-playbook -i hosts/customer/host_files/production.yml
services/epc.yml
ansible-playbook -i hosts/customer/host_files/production.yml
services/ims.yml
```

نماذج النشر

هناك طريقتان للحصول على مجموعة أدوات OmniCore وتشغيله على وحدة الت��كم ��.

النموذج A: استنساخ المستودع على وحدة التحكم

النموذج التقليدي: استنساخ هذا المستودع على وحدة التحكم Ansible الخاصة بك، احتفظ بمخزونك تحت `hosts/`، وقم بتشغيل مجموعات التشغيل مباشرة من النسخة المستنسخة. هذا هو النموذج المفترض في مثال النشر أعلاه.

النموذج B: تثبيت `omnicore` .deb ذاتيًا

تم تعبئة OmniCore أيضًا كحزمة Debian ذاتية الاحتواء `omnicore`، بحيث يمكن لوحدة التحكم تثبيت مجموعة الأدوات من apt بدلاً من استنساخ المستودع.

- **ما الذي يتم تثبيته.** الحزمة (`name: omnicore`، انظر `packaging/nfpm.yaml`) تتضمن أدوار Ansible، ومجموعات التشغيل، ومجموعات *vendored*، ومترجم wheelhouse غير المتصل بالإنترنت و (`/opt/omnicore/python`) مضمن Python إلى `/opt/omnicore`. تقوم خطوة ما بعد التثبيت (`packaging/postinstall.sh`) ببناء venv ذاتي الاحتواء (`/opt/omnicore/venv`) من المترجم المضمن + wheelhouse وترتبط حيث تقوم وحدة ��، عبر `/usr/local/bin` المضمنة إلى `ansible*` ومكتبات Python الخاصة من Ansible `uv.lock` التحكم بتشغيل النسخ الدقيقة من نظامي، لا شبكة. يتم سحب الأدوات النظامية Python توزيعي، لا Ansible بها: لا الحقيقية (`git`، `rsync`، `openssh-client`) بواسطة apt. الحزمة هي **amd64** وتستهدف **Ubuntu 22.04+** (glibc ≥ 2.35).
- **كيف يتم بناؤها.** يقوم `packaging/build.sh` بتهيئة المستودع إلى `/opt/omnicore` (باستثناء `hosts/`، `.git/`، virtualenvs وملفات البناء)، يبيع + يجمع المترجم المستقل، `requirements.yml` من Ansible مجموعات المقفلة، يقوم بتشغيل فحص سري مغلق، ثم Python من اعتمادات wheelhouse يبني `.deb` باستخدام nfpm.
- **كيف يتم نشرها.** يقوم `.github/workflows/build-deb.yml` بتشغيل `build.sh` على كل دفع إلى `main` وينشر `.deb` (بالإضافة إلى `_metadata.json`) كإصدار (كملحق GitHub. يقوم برنامج `discover_github_debs.py` الخاص بخادم Omnitouch باكتشاف هذا الأصل الإصدار عبر منظمة apt ويضيفه إلى aptly. لا حاجة لتغييرات في دور apt.

تدفق العم��ل:

```
# 1. تثبيت مجموعة الأدوات من apt
apt-get install omnicore

# 2. توفير مخزونك الخاص
#    (لا تقوم الحزمة بشحن واحد - إنه يعيش خارج /opt/omnicore)
vi /etc/omnicore/inventory

# 3. تشغيل مجموعات التشغيل من مجموعة الأدوات المثبتة
cd /opt/omnicore
ansible-playbook -i /etc/omnicore/inventory services/all.yml
```

مفصول عن مجموعة الأدوات (`/etc/omnicore/inventory`) نظرًا لأن المخزون يعيش في المثبتة (`/opt/omnicore`)، فإن ترقية الحزمة لا تمس تعريف البنية التحتية أو التخصيص الخاص بك.

الوثائق ذات الصلة

- البدء - مقدمة في نشر Ansible
- مجموعات تشغيل الخدمة - **مرجع مجموعة التشغيل والتسلسل الهرمي**
- تكوين ملف المضيفين - تعريف البنية التحتية
- معيار تخطيط IP - **بنية الشبكة وتخصيص IP**
- تكوين المتغيرات الجماعية - التخصيص
- نظام APT Cache - إدارة الحزم
- خادم الترخيص - إدارة الترخيص
- المراق��ة والرصد - Grafana، Prometheus، ولوحات المعلومات، التنبيهات
- تسجيل مركزي Loki وAlloy - جمع سجلات

وثائق المنتج

للحصول على معلومات مفصلة حول تكوين كل مكون:

- **OmniCore** (4G/5G Packet Core): https://docs.omnitouch.com.au/docs/repos/OmniCore

    - OmniHSS، OmniSGW، OmniPGW، OmniUPF، OmniDRA، OmniTWAG

- **OmniCall** (الصوت والرسائل): https://docs.omnitouch.com.au/docs/repos/OmniCall
    - OmniTAS، OmniCall CSCF، OmniMessage، OmniSS7، VisualVoicemail
- **OmniCharge/OmniCRM** (الفوترة): https://docs.omnitouch.com.au/docs/repos/OmniCharge
- **الوثائق الرئيسية**: https://docs.omnitouch.com.au/

جدار الحماية حسب الدور

يوفر جدار الحماية حسب الدور لكل مضيف مجموعة قواعد `iptables` و `ip6tables` مستمدة من نموذج إعلاني واحد لحركة مرور الشبكة، بدلاً من قواعد مكتوبة يدوياً لكل مضيف. تصف ملف واحد كل تدفق مسموح به في النواة (من يمكنه الوصول إلى أي دور، وعلى أي بروتوكول ومنفذ، ولماذا). من هذا النموذج، تنتج المنصة شيئان يتفقان دائماً مع بعضهما البعض:

- **مجموعة القواعد المفروضة** المطبقة على كل مضيف، و
- **مصفوفة الاتصالات / المنافذ** (CSV) للمراجعة ولتسليمها إلى جدار حماية أعلى.

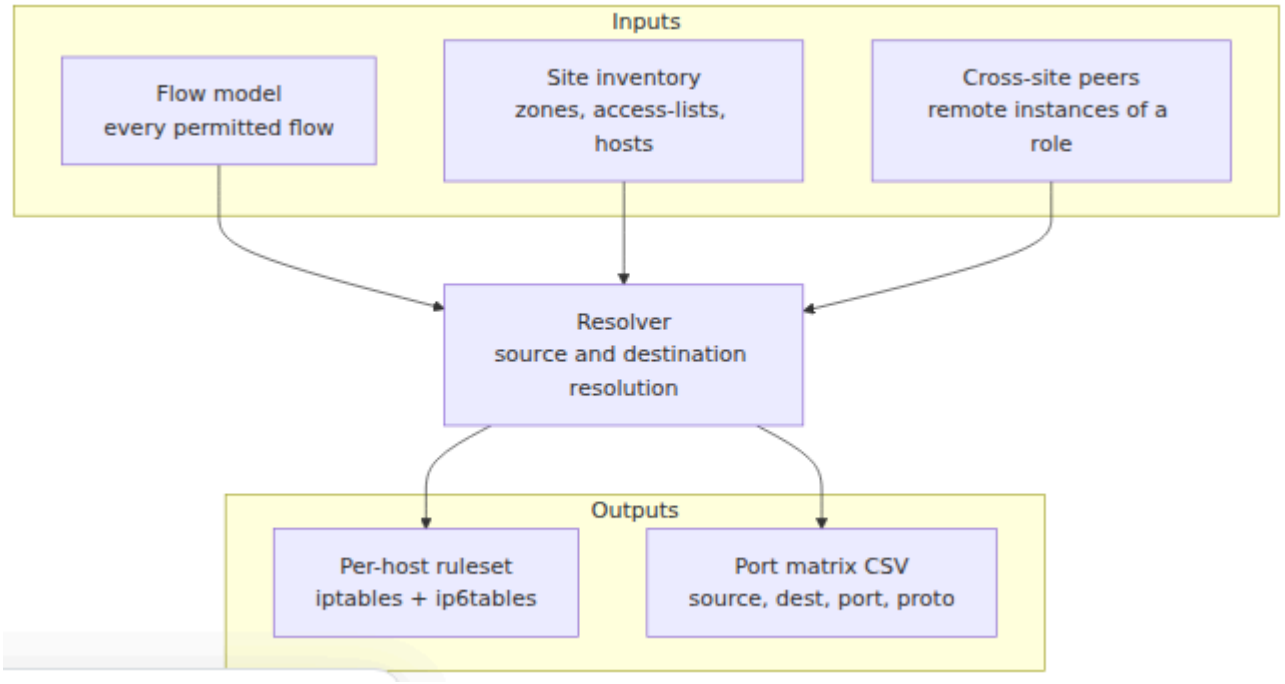
تعمل القواعد في أحد وضعين: **مراقبة** (تسجيل كل تدفق، عدم إسقاط أي شيء) أو **فرض** (رفض افتراضي). الحجب متعمد وآمن، وكل تطبيق محمي بواسطة استرجاع تلقائي على المضيف حتى لا تؤدي مجموعة قواعد خا��ئة إلى إغلاق المشغلين.

جدول المحتويات

- نظرة عامة على المعمارية
- نموذج التدفق
- أنواع المصادر
- فهرس الخدمات
- تكوين الموقع
- أوضاع التشغيل
- الأقران عبر المواقع
- حل العناوين
- مصفوفة الاتصالات / المنافذ
- حماية من الإغلاق
- والدعم المزدوج IPv6
- التسجيل
- تحذيرات تشغيلية

نظرة عامة على المعمارية

نموذج واحد بالإضافة إلى تغذية جرد كل موقع تغذي محللاً واحداً. ينتج المحلل كل من القواعد على المضيف ومصفوفة المنافذ، لذا لا يمكن أن تصف الوثائق شيئاً مختلفاً عما يتم فرضه فعلياً.



يتم تسجيل حركة المرور التي تم إسقاطها وتدقيقها من خلال خط أنابيب الرؤية القياسي:



نموذج التدفق

نموذج التدفق هو المصدر الوحيد للحقيقة حول **من يمكنه الوصول إلى ماذا**. يحتوي على جزئين: مجموعة من الافتراضات الهيكلية (التي لا تحمل أي عناوين موقع)، وقائمة التدفقات. لا تقول التدفق شيئاً عن أرقام المنافذ مباشرة - بل تسمي **خدمة** مكشوفة من قبل الدور الوجهة (انظر فهرس الخدمات)، ويتم البحث عن المنفذ من إعلان ذلك الدور:

```yaml
- id: '13'                          # stable label, appears in each
rule's comment
  dst: hss                          # inventory group the rule is
applied to
  service: diameter                 # named service exposed by the
hss role (-> 3868/tcp+sctp)
  purpose: 'Diameter'               # description, shown in the
matrix
  sources: [{ zone: internal }]     # who may reach it (list of
source tokens)
```

المعلمة	النوع	مطلوب	الافتراضي	الوصف
`id`	سلسلة	نعم	-	معرف ثابت للتدفق. يظهر في تعليق كل قاعدة تم إنشاؤها كـ `ans:<dst>:<id>` بحيث يمكن تتبع قاعدة حية إلى النموذج.
`dst`	سلسلة	نعم	-	مجموعة الجرد التي تقع عليها القاعدة، أو الوجهات الزائفة `all` (كل مضيف) و `five_g_sbi` (مجموعات وظائف الشبكة 5G). ينتج التدفق الذي لا يتم نشر مجموعة وجهته في موقع ما لا قواعد ولا صفوف في المصفوفة هناك.
`service`	سلسلة	نعم*	-	خدمة مسماة تم إعلانها من قبل الدور الوجهة (انظر فهرس الخدمات). يتم البحث عن المنفذ (المنافذ) من إعلان الدور، لذا لا تعيد التدفق ذكر رقم منفذ.
`proto_ports`	قائمة	نعم*	-	البديل الحرفي لـ `service`، للتدفقات غير المرتبطة بدور واحد (على سبيل المثال تدفقات `all`-host). كل إدخال هو `PORT/proto` (`tcp`، `udp`، `sctp`؛ قد يكون المنفذ نطاق `LO-HI`) أو بروتوكول عاري (`icmp`، `esp`، `proto 89`).
`purpose`	سلسلة	نعم	-	وصف إنساني للتدفق. يظهر في عمود `service` من مصفوفة المنافذ.
`sources`	قائمة	نعم	-	المصادر المسموح بها، كقائمة من رموز المصدر. انظر أنواع

المعلمة	النوع	مطلوب	الافتراضي	الوصف
				المصادر.

* **إما** يوفر التدفق `service` **أو** `proto_ports`، وليس كلاهما.

أنواع المصادر

كل إدخال في قائمة `sources` الخاصة بتدفق هو رمز ذو مفتاح واحد يتم حله ضد تكوين الموقع وجردها. لا يساهم الرمز الذي لا يتم حله (على سبيل المثال قائمة وصول فارغة) في أي قاعدة: لا يتم إصداره أبداً كعنصر نائب.

الرمز	يحل إلى
`{ zone: <name> }`	CIDRs لتلك المنطقة الطائرة (`firewall.zones.<name>`).
`{ list: <name> }`	قائمة وصول مشغل، على سبيل المثال `allowed_oam_subnets`.
`{ group: <name> }`	عناوين مضيف مجموعة الجرد، بالإضافة إلى أقران ذلك الدور عبر المواقع. انظر الأقران عبر المواقع.
`{ ue_pools: true }`	تجمعات UE / المشتركين في الموقع (`firewall.ue_pools`).
`{ dra_peers: true }`	كل عنوان تم تكوينه في قائمة الأقران الخاصة بدور DRA.
`{ net_devices: true }`	كل عنوان جهاز شبكة مُدار (`all.vars.<vendor>.devices`، على سبيل المثال `mikrotik.devices`).
`{ roaming: true }`	نطاقات التجوال (IR.21 / GRX-IPX). حالياً أي مصدر: انظر تحذيرات تشغيلية.
`{ literal: "..." }`	مصدر نصي حر غير مستمد من الجرد.

فهرس الخدمات

أرقام المنافذ **تعود إلى الدور الذي يفتحها**، وليس إلى النموذج المركزي. كل دور يكشف عن المنافذ يشحن ملف `firewall_services.yml` في دليل دوره، مفاتيح بواسطة مجموعة الجرد (يخدم عدة مجموعات، لذا يعلن كل منها `cscf` دور مثل) التي يخدمها هذا يحافظ على تعريف المنفذ بجوار الكود الذي يفتحه:

```
# 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 (queried by other
NFs)
    postgres: 5432/tcp                # logical replication
between HSS peers
```

يجمع المحلل كل `firewall_services.yml` لكل دور في فهرس واحد. يتم البحث عن `service:` الخاصة بذلك التدفق، لذا يعبر النموذج فقط عن `dst` للتدفق في الفهرس لمجموعة `hss` الخاصة بـ `diameter` من يمكنه الوصول إلى خدمة (**العلاقة**) بينما **المنفذ** يعيش مع الدور. أضف أو انقل مستمعاً في دور وقم بتحديث ملف `firewall_services.yml` الخاص به؛ لا يتغير أي شيء في النموذج المركزي.

يمتلك هذا الانقسام شبكة أمان تفتقر إليها النموذج المركزي فقط: **فحص الانجراف** يقارن ما تعلنه الأدوار ضد ما تستمع إليه المضيفات فعلياً.

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

لكل مضيف، يتم الإبلاغ عن:

- **غير معلن** (إما إضافته إلى ملف `firewall_services.yml` الخاص بالدور، أو أنه مستمع غير متوقع) - مستمع حي على منفذ لم يعلنه أي دور، و
- **غير مستمع** - خدمة معلنة غير موجودة فعلياً.

يتم تخطي المضيفات التي لا تعلن أي خدمات، لذا فإن الفحص مفيد أثناء ترحيل الأدوار إلى الفهرس تدريجياً بدلاً من جميعها دفعة واحدة.

MME **المستمعون المشروطون.** يتم فتح بعض المنافذ فقط عندما يتم نشر دور آخر: يفتح منافذ LCS الخاصة به (SBc-AP `29168/sctp`، SLs `9082/sctp`) فقط عندما يكون مضيف `omnilcs` في الجرد. لا يمكن أن تعبر إعلان ثابت عن هذا الشرط، لذا يتم ملاحظة هذه المنافذ في ملف `firewall_services.yml` الخاص بالدور ولكن لا يتم إعلانها. في موقع يقوم بتشغيل الدور المعتمد�� يقوم فحص الانجراف بالإبلاغ عنها كغير معلنة، وهو التحذير الصحيح لمراجعتها والسماح بها.

تكوين الموقع

في جرده. لا توجد بدائل للعناوين: يجب `all.vars` تحت `firewall` يعلن كل موقع عن كتلة على الموقع وصف شبكته الخاصة. أي شيء لا يعلنه الموقع يكون ببساطة فارغاً، ولا يتم وراثة عناوين أي بيئة أخرى.

المثال أدناه مكتمل بالكامل بحيث يظهر كل حقل قيمة تمثيلية. تستخدم العناوين نطاقات توثيق وتخطيط متماسك لكل طائرة (كل طائرة هي شبكتها الفرعية الخاصة ضمن السوبرنت)، على سبيل المثال، `internal`)، وهو ما يجعل المناطق مفيدة: إنها تسمح لتدفق واحد بالسماح، فقط لطائرة IMS بدلاً من النواة بأكملها.

```
firewall:
  allowed_oam_subnets:           ['10.8.0.0/24']       #
management source (SSH, UI, scrape, mgmt APIs)
  allowed_ran_subnets:           ['10.20.0.0/16']      # RAN
signalling source (S1AP, NGAP, GTP-U, Abis)
  allowed_ue_voice_subnets:      ['100.64.0.0/16']     # external
UE SIP into the P-CSCF
  allowed_carrier_voice_subnets: ['203.0.113.0/24']    # carrier
interconnect SIP
  zones:                                               # plane
zones: each plane's real subnet(s)
    internal:    ['10.8.0.0/16']                       #   trusted
core supernet
    mobile_core: ['10.8.1.0/24']                       #   mobile
core NFs (EPC + 5GC + CS core)
    core_svc:    ['10.8.2.0/24']                       #   core
services plane (HSS, DRA, OCS, OAM)
    ims:       ['10.8.3.0/24']                         #   IMS
plane (CSCF, TAS)
    ue_svc:    ['10.8.4.0/24']                         #   UE-
reachable plane (P-CSCF, DNS, ePDG)
    core_v6:  ['2001:db8:0:1::/64']                    #   IPv6
addresses of core NFs (not UEs)
  ue_pools: ['100.64.0.0/16', '2001:db8:64::/48']      # UE /
subscriber IP ranges (v4 and v6)
  extra_rules:                                         # manual
accept rules (see below)
    - { proto: tcp, dport: '8443', source: '203.0.113.10/32',
comment: 'partner API' }
```

هو إلزامي. يمكن ترك كل قائمة أخرى فارغة (`[]`) في موقع `allowed_oam_subnets` فقط يفتقر إلى تلك الوظيفة: موقع بلا RAN يحدد `allowed_ran_subnets: []`، موقع بلا نواة IPv6 وموقع بلا UE يحدد `ue_pools: []`. تنتج القائمة الفا��غة يحدد `core_v6: []`، وموقع بلا تجمعات بسيطة عدم وجود قواعد للتدفقات التي تستخدمها (انظر عمود `Required` أدناه).

المعلمة	النوع	مطلوب	الافتراضي	الوصف
`allowed_oam_subnets`	قائمة	نعم	-	ات المصدر لإدارة. تحكم واجهة SSH، OmniWeb سحب Prometheu وواجهات برمجة التطبيقات الإدارية. إذا كانت فارغة، وقف المولد ع خطأ، لأن قيمة فارغة تترك الإدارة بلا مصدر مسموح به سبب إغلاقاً حت الفرض
`allowed_ran_subnets`	قائمة	لا	`[]`	ات المصدر لإشارات RA (S1AP، NGAP، GT U، Abis). كن أن تكون فارغة في موقع RA بلا
`allowed_ue_voice_subnets`	قائمة	لا	`[]`	ات المصدر لـ SIP UE خارجي إلى P-CSCF.

المعلمة	النوع	مطلوب	الافتراضي	الوصف
`allowed_carrier_voice_subnets`	قائمة	لا	`[]`	ات المصدر لربط SIP لـ ن الشركات
`zones`	خريطة	نعم	-	مناطق طائرات التي بط كل اسم ائرة بشبكتها الفرعية قيقية. انظر معلمات المنطقة.
`ue_pools`	قائمة	لا	`[]`	نطاقات IP UE / المشتركين. فارغة إذا لم يكن لدى الموقع أي منها.
`extra_rules`	قائمة	لا	`[]`	قواعد قبول دوية مطبقة على كل ضيف. انظر معلمات القاعدة اليدوية.

معلمات المنطقة

حدد فقط الطائرات التي يستخدمها الموقع؛ اترك .CIDRs تربط كل منطقة اسم طائرة بقائمة من البقية فارغة.

المنطقة	الوصف
`internal`	السوبرنت الأساسي الموثوق. أوسع مصدر إشارات داخلية.
`mobile_core`	وظائف شبكة النواة المتنقلة عبر الأجيال: EPC (MME، S/P-GW، UPF)، 5GC (AMF، SMF، وشبكة SBI)، ونواة CS (MSC، SS7/SIGTRAN).
`ims`	طائرة IMS (I-CSCF، S-CSCF، TAS).
`ue_svc`	طائرة الخدمات القابلة للوصول من UE (P-CSCF، DNS، ePDG).
`core_svc`	طائرة خدمات النواة (HSS، DRA، OCS، OAM).
`core_v6`	عناوين خدمات النواة IPv6. فارغة إذا لم يكن لدى الموقع نواة IPv6.

معلمات القاعدة اليدوية

تضيف `extra_rules` (عالمية، لكل المضيفات) و `firewall_extra_rules` لكل مضيف الذي يحتوي على `:` على `source` قواعد قبول خام للحالات التي لا يغطيها النموذج. يتم تطبيق IPv6.

المعلمة	النوع	مطلوب	الافتراضي	الوصف
`proto`	سلسلة	لا	-	البروتوكول، على سبيل المثال `tcp`، `udp`، `sctp`.
`dport`	سلسلة	لا	-	منفذ الوجهة أو النطاق.
`source`	سلسلة	لا	-	المصدر. يتم تطبيق قاعدة على CIDR عند وجود `:` في القيمة IPv6.
`comment`	سلسلة	لا	`override`	تسمية مسجلة في تعليق القاعدة.

أوضاع التشغيل

يتم التحكم في وضع التطبيق بواسطة `firewall_mode`.

الوضع	السلوك
`monitor`	يتم تثبيت كل قاعدة قبول، تبقى سياسة `INPUT` على `ACCEPT`، ويتم تسجيل حركة المرور التي تمر (كـ `FW-AUDIT`) ثم السماح بها. لا يتم إسقاط أي شيء. استخدم هذا للتحقق من النموذج ضد حركة المرور الحقيقية قبل فرضه.
`enforce`	يتم تسجيل حركة المرور التي تمر (كـ `FW-DROP`) ثم إسقاطها، ويتم تعيين سياسات `INPUT` و `FORWARD` إلى `DROP`.

تسقط حركة المرور. أي قيمة `enforce` **الحجب هو اختيار آمن: فقط القيمة الدقيقة** أخرى، بما في ذلك متغير غير محدد أو خطأ مطبعي، يتم اعتبارها مراقبة، لذا لا يمكن لمضيف أن يبدأ في إسقاط حركة المرور عن طريق الخطأ. هذا ينطبق على ثلاثة مستويات: الافتراضي للمنصة في جردها، ويعتبر المولد أي شيء `firewall_mode: monitor` تعلن المواقع، `monitor` هو ليس `enforce` كمراقبة. يتطلب التنفيذ تغييراً متعمداً.

```
all:
  vars:
    firewall_mode: monitor   # change to 'enforce' only after the
audit log confirms coverage
```

أو تجاوزها لكل مجموعة أو مضيف، `all.vars` لكل موقع تحت `firewall_mode` يمكن تعيين من خلال تسلسل المتغيرات العادي.

الأقران عبر المواقع

بإدراج (CSCF، TAS، OmniMessage، تكرار HSS، OCS) تقوم الأدوار التي تتعاون عبر المواقع على سبيل المثال) أقرانها البعيدين في ملف الأقران الخاص بالموقع `prod_master_peers.yml`). يتم دمج تلك العناوين البعيدة في حل `{ group: <role> }`، لذا فإن التدفق الذي يسمح بدور يغطي تلقائياً كل من مثيلات نفس الموقع والأقران عبر المواقع تحت سياسة واحدة متطابقة. يتم استخدام نفس الحل للقواعد والمصفوفة، لذا تظهر الأقران في كلاهما.

حل العناوين

يتم حل المضيف إلى **كل عنوان تم الإعلان عنه**، وليس فقط الأساسي. هذا يعني في الجرد. تربط `ansible_host` بالإضافة إلى كل `secondary_ips.<name>.ip_address` N3 على UPF الثانوية: يجيب NIC العقدة الخاصة بمسار البيانات وعقد الإشارات الخدمات ببطاقات على عنوانه العام. تغطي كل من القواعد SIGTRAN ويبدأ STP ويتلقى `n3` على عنوانه GTP-U والمصفوفة كل تلك العناوين، كمصادر وكوجهات، لذا فإن نظيراً يسمح لعقدة من خلال العنوان المقدم له هو السماح بالعنوان الذي تستخدمه حركة المرور فعلياً.

المضيف الذي ينتمي إلى **عدة مجموعات جرد** يتلقى **اتحاد** كل تدفقات كل مجموعة وخدماتها المعلنة. تكشف العقدة المتقاربة عن كل شيء لجميع الأدوار التي تعمل بها، ومجموعة المستمعين المتوقعة (المستخدمة بواسطة فحص الانجراف) هي الاتحاد عبر مجموعاتها.

يمكن أن تسرد المصفوفة NIC، لأن الحل يستخدم كل عنوان معلن بدلاً من تتبع أي خدمة تربط أي الخاص بـ N6 على عنوان N3 GTP-U على سبيل المثال، يظهر) خدمة على عنوان لا ترتبط به فعلياً UPF). هذا هو تقريبي متعمد: مصفوفة كاملة ولكنها واسعة قليلاً آمنة لتسليمها إلى جدار حماية إلى كسر حركة المرور SIGTRAN أعلى، بينما يؤدي حذف عنوان مسار البيانات الحقيقي أو عنوان الذي ترتبط به؛ NIC بصمت. إذا كانت الدقة حسب العنوان مطلوبة لاحقاً، يمكن أن تسمي الخدمة حتى ذلك الحين، يبقى الحل مستنداً إلى الحقائق وكاملاً.

مصفوفة الاتصالات / المنافذ

مصفوفة المنافذ هي نموذج التدفق المحلول ضد جرد الموقع، مسطحة إلى صف واحد لكل مصدر، مضيف وجهة، ومنفذ/بروتوكول واحد. إنها الشكل للمراجعة أو لتسليمها إلى جدار حماية أعلى. قم بإنشائها باستخدام:

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

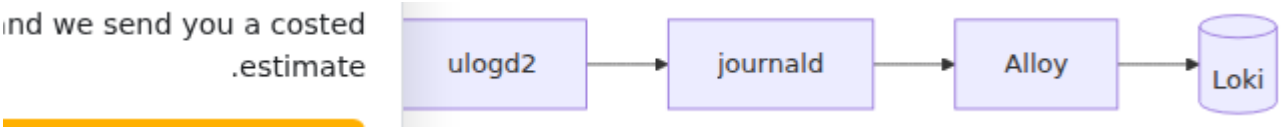
هذا يكتب `generated/firewall-port-matrix-exploded.csv` (لكل موقع، غير ملتزم). الأعمدة هي:

العمود	الوصف
`id`	معرف التدفق.
`service`	غرض التدفق.
`source`	CIDR أو عنوان مصدر محلول.
`destination_role`	مجموعة الجرد الوجهة.
`destination_host`	اسم المضيف الوجهة.
`destination_ip`	عنوان المضيف الوجهة.
`port`	منفذ الوجهة أو النطاق (فارغ للإدخالات التي تحتوي على بروتوكول فقط مثل ICMP أو ESP).
`protocol`	البروتوكول.

يظهر الصف فقط عندما يتم نشر وجهته في الموقع ويحل مصدره إلى عنوان حقيقي. تنتج الأدوار غير المنشورة والمصادر غير المحلولة لا صفوف.

حماية من الإغلاق

تتم حماية تطبيق ��لقواعد بواسطة استرجاع تلقائي، يتم التحكم فيه بواسطة `firewall_rollback_guard` (مفعل افتراضياً) و `firewall_rollback_timeout`:



1. يتم التقاط مجموعة القواعد الحية.
2. يتم تسليح مؤقت على المضيف سيستعيد اللقطة. يبقى هذا بعد قطع جلسة الإدارة.
3. يتم تطبيق القواعد الجديدة.
4. يعيد المتحكم إنشاء اتصال جديد ويجري فحصاً تافهاً. إذا كانت القواعد الجديدة قد أغلقت الوصول، فإن هذا يفشل ويتوقف التشغيل لذلك المضيف.

5. إذا تم تأكيد الاتصال، يتم إلغاء تسليح المؤقت. إذا لم يكن كذلك، يشتعل المؤقت ويستعيد مجموعة القواعد العاملة دون أي إجراء من المشغل.

المعلمة	النوع	مطلوب	الافتراضي	الوصف
`firewall_rollback_guard`	Boolean	لا	`true`	يمكّن اللقطة، المؤقت، وفحص إعادة الاتصال حول كل تطبيق.
`firewall_rollback_timeout`	عدد صحيح	لا	`120`	الثواني التي ينتظرها المؤقت قبل استعادة اللقطة تلقائياً.

والدعم المزدوج IPv6

ويتم تطبيق القاعدة على IPv6، يعني `:` في القيمة CIDR: يتم استنتاج عائلة العنوان لكل تنتج ببساطة عدم وجود CIDRs IPv4 لذلك، فإن منطقة أو قائمة تحتوي فقط على `ip6tables`. IPv6 المناطق والتجمعات IPv6 لذلك المصدر. تترك المواقع التي لا تحتوي على نواة IPv6 قواعد إلى IPv6 لـ `INPUT` تحت الفرض، يتم تعيين سياسة (`core_v6: []` على سبيل المثال) فارغة في وضع المراقبة قبل فرض مضيف يصل إليه فقط تدفق IPv6 لذا تحقق من حركة مرور `DROP`، فقط IPv4 الشرعي من خلال مصدر IPv6.

التسجيل

يتم تسجيل حركة المرور التي تم تدقيقها وإسقاطها وتحمل بواسطة خط الأنابيب القياسي إلى Loki. يتم استخدام ثلاثة بادئات:

البادئة	المعنى
`FW-FLOW` / `FW-FLOW6`	اتصال جديد تم رؤيته في وضع المراقبة. يبني خريطة التدفق الحية لحركة المرور المستخدمة فعلياً.
`FW-AUDIT` / `FW-AUDIT6`	حركة المرور التي تمر في وضع المراقبة: سيتم إسقاطها تحت الفرض، ولكنها مسموح بها هنا. راجع هذه قبل فرضها.
`FW-DROP` / `FW-DROP6`	حركة المرور التي تمر والتي تم إسقاطها فعلياً تحت الفرض.

هدف السجل

الذي يقوم الدور بتثبيته كمتطلب، `ulogd2` وتُقرأ بواسطة `NFLOG` تُكتب السجلات باستخدام `iptables-persistent`. يمكن قراءة NFLOG فقط أثناء تشغيل أساسي جنباً إلى جنب مع `ulogd2`، مستهلك — بدون يتخلص النواة من كل سجل، ويثبت وضع المراقبة بعد ذلك عدم إثبات أي شيء بينما يبدو أنه يعمل: يتم تثبيت القواعد، وتزداد العدادات، ولا يتم إنتاج أي سجل. تلك الفشل صامت، ولهذا السبب يكون الحزمة إلزامية بدلاً من الجهد الأفضل.

Omnitouch apt تم حمل كلا الحزمتين في مستودع
(`ubuntu_packages.networking`)، لذا فإن المضيف الذي لا يمكنه تثبيتهما لديه مستودع معطل ويتوقف التشغيل هناك.

تضع خط أنابيب سجل Alloy هذه `unit="ulogd2.service"` و `service="ulogd"`. لاحظ أنه لا يوجد تسمية `job="firewall"` تحمل — `job` على اختر `unit` اسم جرد المضيف، لذا لتغطية كل مضيف.

مثال على استعلامات لوكي:

```
# حركة المرور التي ستُحظر بمجرد تطبيق هذه المضيفات
{unit="ulogd2.service"} |= "FW-AUDIT"

# حركة المرور التي يتم إسقاطها تحت التطبيق، حسب المضيف
sum by (hostname) (count_over_time({unit="ulogd2.service"} |= "FW-DROP" [5m]))

# مضيف واحد فقط
{unit="ulogd2.service", hostname="omni-tampa-sbc01"} |= "FW-DROP"
```

على المضيف نفسه:

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

تحذيرات تشغيلية

- **المصادر المتنقلة والحرفية هي حاليًا أي مصدر.** يتم إصدار التدفقات التي بدون قيود على المصدر (`0.0.0.0/0`). اعتبرها `literal` أو `roaming` مصدرها مفتوحة حتى يتم حل نطاقات التجوال إلى CIDRs حقيقية.
- **يتم قبول ICMP من أي مصدر.** تسمح مجموعة القواعد المولدة بـ ICMP بدون شروط، وهو أوسع من نطاق ا��إدارة المعروض لتدفق ping.
- **لا تفرض مضيفًا في مستوى المستخدم دون قواعد توجيه.** تحت التطبيق، يتم إلى `DROP`، على ذلك فرض إن. توجيه قواعد أي إنشاء يتم ولا `FORWARD` سياسة تعيين دون إضافة قواعد توجيه سيكسر مستوى المستخدم. تحقق أولاً SGW أو PGW أو UPF.
- **قائمة الإدارة خشنة.** تتحكم `allowed_oam_subnets` في SSH وواجهة المستخدم وعمليات السحب، وواجهات برمجة التطبيقات الإدارية معًا. يحتاج الموقع الذي يقوم بالسحب من شبكة مختلفة عن تلك التي يديرها إلى إدخال `extra_rules` لفصلها.

تكوين متغيرات المجموعة

نظرة عامة

دليل `group_vars` هو المكان الذي تخزن فيه ملفات التكوين المخصصة التي تتجاوز القوالب الافتراضية.

هنا تعيش تكويناتك الخاصة بالعميل: خطوط SIP، منطق توجيه الرسائل، قواعد توجيه القطر القصيرة، خطط الاتصال، وأي تخصيصات أخرى حيث لا ترغب في استخدام التكوين الافتراضي. إنه موجود في `group_vars`.

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

تحل الأدوار هذا الدليل في وقت التشغيل كـ `group_vars_omnicore` - الدليل `group_vars/` الذي يقع بجوار دليل الجرد (`{{ inventory_dir.rsplit('/', 1)[0] }}/group_vars/`). في أي مكان يشير فيه مسار ملف أدناه إلى `{{ group_vars_omnicore }}`، فإنه يشير إلى هذا الدليل.

كيف يعمل

تمتلك أدوار Ansible قوالب تكوين ��فتراضية. لتخصيص نشر معين، ضع ملفاتك المخصصة في `group_vars` وارجع إليها في ملف المضيفين الخاص بك.

```
التكوين → group_vars (إذا تم تحديده) → قالب الدور الافتراضي → تجاوز المنشر
```

المثال 1: قالب تكوين مخصص (OmniMessage)

بعض المكونات تقبل قوالب تكوين Jinja2 مخصصة.

هيكل الملف

```
hosts/Customer/
└── group_vars/
    └── smsc_controller.exs        # قالب التكوين المخصص الخاص بك
```

الإشارة في ملف المضيفين

```yaml
omnimessage:
  hosts:
    customer-smsc-controller01:
      ansible_host: 10.10.3.219
      gateway: 10.10.3.1
      host_vm_network: "vmbr3"
      smsc_template_config: smsc_controller.exs   # الإشارة إلى اسم ملف القالب الخاص بك في group_vars
```

ماذا يحدث:

1. يجد Ansible `smsc_template_config: smsc_controller.exs`
2. يبحث في `hosts/Customer/group_vars/smsc_controller.exs`
3. يقوم بتهيئته باستخدام Jinja2 (يمكن استخدام `{{ inventory_hostname }}`، `{{ plmn_id.mcc }}`، إلخ.)
4. ينشر إلى `/etc/omnimessage/runtime.exs`
5. يعيد تشغيل الخدمة

بدون `smsc_template_config`، يتم استخدام القالب الافتراضي من الدور.

انظر **تفاصيل التكوين:** https://docs.omnitouch.com.au/docs/repos/OmniCall

---

المثال 2: مجموعات ملفات التكوين (وخطط الاتصال OmniTAS بوابات)

بعض المكونات تستخدم أدلة لملفات التكوين.

هيكل الملف

```
hosts/Customer/
└── group_vars/
    ├── gateways_prod/              # تكوينات بوابة SIP
    │   ├── gateway_carrier1.xml
    │   ├── gateway_carrier2.xml
    │   └── gateway_emergency.xml
    ├── gateways_lab/               # بوابات المختبر
    │   └── gateway_test.xml
    ├── dialplan/                   # قواعد توجيه المكالمات
(افتراضي)
    │   ├── mo_dialplan.xml         # مكالمات منشأة من الهاتف
(صادرة) المحمول
    │   ├── mt_dialplan.xml         # مكالمات مستلمة على الهاتف
(واردة) المحمول
    │   └── emergency.xml
    └── dialplan_lab/               # خطط الاتصال في المختبر
        └── mo_dialplan.xml
```

الإشارة في ملف المضيفين

```yaml
applicationserver:
  hosts:
    customer-tas01:
      ansible_host: 10.10.3.60
      gateway: 10.10.3.1
      host_vm_network: "vmbr3"
      gateways_folder: "gateways_prod"   # الإشارة إلى مجلد البوابة الخاص بك لاستخدامه على هذا المضيف
      dialplan_folder: "dialplan"         # اختياري - الافتراضي هو "dialplan" إذا لم يتم تعيينه
```

ماذا يحدث:

1. يجد Ansible `gateways_folder: "gateways_prod"`
2. ينسخ جميع الملفات من `hosts/Customer/group_vars/gateways_prod/` إلى `/etc/freeswitch/sip_profiles/`

3. ينسخ جميع الملفات من `hosts/Customer/group_vars/dialplan/` (أو المجلد المحدد بواسطة `dialplan_folder`) إلى دليل قوالب OmniTAS
4. تقوم الخدمات بتحميل التكوينات

بيئات مختلفة: استخدم مجلدات مختلفة لكل بيئة:

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

تفاصيل التكوين: انظر https://docs.omnitouch.com.au/docs/repos/OmniCall

---

المثال 3: قالب تكوين مخصص (OmniHSS)

يقبل خادم المشتركين ال��نزلي قوالب تكوين وقت التشغيل المخصصة.

هيكل الملف

```
hosts/Customer/
└── group_vars/
    └── hss_runtime.exs.j2        # قالب تكوين HSS الخاص بك المخصص
```

الإشارة في ملف المضيفين

```
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   # الإشارة إلى اسم ملف القالب الخاص بك في group_vars
```

ماذا يحدث:

1. يجد Ansible `hss_template_config: hss_runtime.exs.j2`
2. يبحث في `hosts/Customer/group_vars/hss_runtime.exs.j2`
3. يقوم بتهيئته باستخدام Jinja2 (يمكن استخدام `{{ inventory_hostname }}`، `{{ plmn_id.mcc }}`، إلخ.)
4. ينشر إلى `/etc/omnihss/runtime.exs`
5. يعيد تشغيل الخدمة

بدون `hss_template_config`، يتم استخدام القالب الافتراضي من الدور.

تفاصيل التكوين: انظر https://docs.omnitouch.com.au/docs/repos/OmniCore

---

المثال 4: قالب تكوين مخصص (OmniMME)

تقبل كيان إدارة الحركة قوالب تكوين وقت التشغيل المخصصة.

هيكل الملف

```
hosts/Customer/
└── group_vars/
    └── mme_runtime.exs.j2        # قالب تكوين MME المخصص الخاص بك
```

الإشارة في ملف المضيفين

```
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   # الإشارة إلى اسم ملف القالب الخاص بك في group_vars
```

ماذا يحدث:

1. يجد Ansible `mme_template_config: mme_runtime.exs.j2`
2. يبحث في `hosts/Customer/group_vars/mme_runtime.exs.j2`
3. يقوم بتهيئته باستخدام Jinja2 (يمكن استخدام `{{ inventory_hostname }}`، `{{ plmn_id.mcc }}`، إلخ.)
4. ينشر إلى `/etc/omnimme/runtime.exs`
5. يعيد تشغيل الخدمة

بدون `mme_template_config`، يتم استخدام القالب الافتراضي من الدور.

تفاصيل التكوين: انظر https://docs.omnitouch.com.au/docs/repos/OmniCore

---

المثال 5: ملفات التكوين المطلوبة (OmniCRM)

لا يوجد دور افتراضي للعودة إليه. تم - group_vars **حصريًا** من OmniCRM يتم الحصول على تصميم تكوين API بحيث لا يتم تضمينه في الدو حتى يفشل الملف المفقود في التشغيل بصوت عالٍ بدلاً من شحن أسرار موقع آخر بصمت.

هيكل الملف

```
hosts/Customer/
└── group_vars/
    ├── crm_config.yaml          # API + ui: (مطلوب) التكوين
    ├── deployment_assets/       # شعارات/أيقونات اختيارية →
/etc/omnicrm/deployment_assets
    └── omnicrm_plays/           # ألعاب توفير الموقع الاختيارية،
يتم نسخها فوق الخروج
```

الإشارة في ملف المضيفين

```yaml
crm:
  hosts:
    customer-crm01:
      ansible_host: 10.10.3.70
      gateway: 10.10.3.1
      host_vm_network: "vmbr3"
      ssl_fqdn: portal.example.com
```

ماذا يحدث:

1. يتم شحن OmniCRMv2 إلى `/opt/omnicrm-v2`، وبناء بيئة العمل الخاصة به، وقاعدة البيانات PostgreSQL وإنشاء دور.
2. يتم تهيئة `{{ group_vars_omnicore }}/crm_config.yaml` إلى `/etc/omnicrm/crm_config.yaml` (**مطلوب**).
3. يتم مزامنة `deployment_assets/` إلى `/etc/omnicrm/deployment_assets/`.
4. يتم ترحيل، وبناء واجهة المستخدم على المضيف، وبدء وحدات خلف `omnicrm-v2-*` nginx.

مهم: يجب أن يكون `crm_config.yaml` يحتوي على كتلة `ui:` مع موج��دًا `ui.assets_dir: '/etc/omnicrm/deployment_assets'`.

تفاصيل التكوين: انظر https://docs.omnitouch.com.au/docs/repos/OmniCore

---

المثال 6: دليل بيانات التوفير (OmniHSS)

تقرأ خدمة توفير OmniHSS (`services/provision_omnihss.yml`) بيانات المشتركين والملفات الشخصية من دليل مخصص `omnihss/` داخل group_vars.

هيكل الملف

```
hosts/Customer/
└── group_vars/
    └── omnihss/
        ├── subscribers.csv              # المشتركين للتوفير
        ├── 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
```

ماذا يحدث:

1. تقرأ الخدمة `{{ group_vars_omnicore }}/omnihss/subscribers.csv` لقائمة المشتركين للتوفير.
2. تقرأ الملفات JSON المصاحبة من نفس دليل `omnihss/` (`apn_identifiers.json`، `apn_qos_profiles.json`، `apn_profiles.json`، `eir_rules.json`، `ims_profiles.json`، `epc_profiles.json`، `roaming_rules.json`، `roaming_profiles.json`، `flow_information_rules.json`، `charging_rules.json`، `pcrf_profiles.json`) لتعريف بيانات APN، QoS، IMS، EPC، التجوال، التدفق، الشحن وبيانات PCRF المطبقة على هؤلاء المشتركين.

تفاصيل التكوين: انظر https://docs.omnitouch.com.au/docs/repos/OmniCore

---

المثال 7: بيانات اعتماد SMTP لـ Grafana (المراقبة)

تقوم قالب `grafana.ini` بتوصيل إعدادات البريد الإلكتروني لدور المراقبة (SMTP) لتنبيهات Grafana. تم - group_vars من وكلمة المرور لاسم المستخدم **قيمة افتراضية للدور** لا توجد تصميمها بشكل متعمد بحيث يفشل التشغيل بصوت عالٍ بدلاً من شحن بيانات الاعتماد.

الإشارة في ملف المضيفين

```
monitoring:
  hosts:
    customer-oam01:
      ansible_host: 10.10.3.135
      gateway: 10.10.3.1
      host_vm_network: "vmbr3"
      grafana_smtp_user: "smtp-user"          # مطلوب - لا يوجد افتراضي
      grafana_smtp_password: "smtp-password"  # مطلوب - لا يوجد افتراض
      grafana_smtp_host: "in-v3.mailjet.com:587"  # اختياري - الافتراضي هو in-v3.mailjet.com:587
```

ماذا يحدث:

1. يتم تهيئة `grafana_smtp_user` و `grafana_smtp_password` في قسم `[smtp]` من `/etc/grafana/grafana.ini`. لا يوجد بديل - كلاهما مطلوب.
2. `grafana_smtp_host` اختياري ويفترض القيمة الافتراضية `in-v3.mailjet.com:587` إذا لم يتم تعيينه.

تفاصيل التكوين: انظر https://docs.omnitouch.com.au/docs/repos/OmniCore

---

المثال 8: مفاتيح HNET لإزالة إخفاء SUCI لـ UDM

يقوم OmniUDM بإزالة إخفاء SUCI (TS 33.501 §6.12): يستعيد SUPI من SUCI المخفي الذي يقدمها UE عند التسجيل. يتطلب ذلك مفاتيح الشبكة المنزلية **الخاصة** التي تتزاوج مع مفاتيح الشبكة المنزلية **العامة** المخصصة على USIMs. تعيش هذه المفاتيح الخاصة في دليل مخصص `/etc/omniudm/hnet/` على مضيف UDM ويتم نشرها إلى group_vars داخل `hnet_udm/` بواسطة دور `gc_nf`5.

هيكل الملف

```
hosts/Customer/
└── group_vars/
    └── hnet_udm/
        ├── curve25519-1.key       # مفتاح خاص X25519، PEM (معرف المفتاح العام للشبكة المنزلية 1)
        ├── curve25519-1.pub       # قيمة عامة خام 32 بايت (hex) - لتوفير USIM
        ├── secp256r1-2.key        # مفتاح خاص EC (prime256v1)، PEM (معرف المفتاح العام HNET 2)
        └── ...
```

تسمية الملفات: `<curve>-<id>.key`، حيث `<id>` هو معرف المفتاح العام للشبكة المنزلية و`<curve>` هو `curve25519` (الملف الشخصي A، X25519) أو `secp256r1` (الملف الشخصي B، prime256v1). اسم الملف هو كيفية ربط المفتاح بمعرف مفتاح HNET على USIM الخاص به. يجب أن يتطابق مع المعرف المبرمج على USIM الذي يتطابق معه UDM ضد SUCI.

ماذا يحدث:

1. يضمن الدور أن `{{ group_vars_omnicore }}/hnet_udm/` موجود على وحدة التحكم Ansible (تم إنشاؤه باستخدام `become: false`: تبقى المفاتيح على وحدة التحكم ولا يتم إنشاؤها أبدًا على الجهاز البعيد).
2. يقوم بمسح الدليل بحثًا عن ملفات `*.key`.
3. **إذا لم يتم العثور على أي منها، يتم إنشاء مفتاح افتراضي `curve25519-1.key` (X25519) تلقائيًا ��**استخدام `openssl genpkey`، بحيث تعمل إزالة إخفاء SUCI من الصندوق. هذا هو idempotent (`creates:` guard): لا يتم الكتابة فوق مفتاح موجود.

4. يتم تصدير القيمة العامة الخام 32 بايت المطابقة إلى `curve25519-*.key`، لكل ملف `.pub` (hex) لتوفير USIM. تبقى ملفات `.pub` في الجرد ولا يتم نشرها إلى UDM.
5. يتم نسخ جميع ملفات `*.key` إلى `/etc/omniudm/hnet/` على UDM (الوضع `0600`، مملوكة للجذر)، ويتم إعادة تشغيل `omniudm`.

> **مهم:** المفتاح الذي تم إنشاؤه تلقائيًا عشوائي وسيقوم فقط بإزالة إخفاء SUCIs من USIMs المبرمجة بمفتاحها العام **المطابق**. بالنسبة لمختبر تتحكم فيه في برمجة SIM، خذ ملف `.pub` المصدّر وقم بتوفيرها كمفتاح عام للشبكة المنزلية (ID 1) على USIM. إذا كانت SIMs الخاصة بك تستخدم مخطط null (بدون إخفاء)، فلا حاجة لمفتاح HNET ويمكن أن يبقى هذا الدليل فارغًا.

لتصدير مفتاح عام من مفتاح خاص موجود يدويًا:

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

تفاصيل التكوين: انظر https://docs.omnitouch.com.au/docs/repos/OmniCore

---

مثال على هيكل الدليل في العالم

الحقيقي

```
hosts/Customer/
├── host_files/
│   └── production.yml          # يشير ملف المضيفين إلى ملفات
group_vars
└── group_vars/
    ├── smsc_controller.exs     # قالب مخصص لـ OmniMessage
    ├── smsc_smpp.exs           # قالب مخصص لـ OmniMessage SMPP
    ├── mmsc_runtime.exs        # قالب OmniMessage MMSc
    ├── tas_runtime.exs.j2      # قالب مخصص لـ TAS
    ├── hss_runtime.exs.j2      # قالب مخصص لـ HSS
    ├── mme_runtime.exs.j2      # قالب مخصص لـ MME
    ├── dra_runtime.exs.j2      # قالب مخصص لـ DRA
    ├── pgwc_runtime.exs.j2     # قالب مخصص لـ PGW
    ├── dea_runtime.exs.j2      # قالب مخصص لـ DEA
    ├── upf_config.yaml         # تكوين UPF
    ├── crm_config.yaml         # تكوين CRM
    ├── stp.j2                  # قالب SS7 STP
    ├── hlr.j2                  # قالب SS7 HLR
    ├── camel.j2                # قالب SS7 CAMEL
    ├── ipsmgw.j2               # قالب IP-SM-GW
    ├── omnicore_smsc_ims.yaml.j2  # تكوين SMSC IMS
    ├── pytap.yaml              # تكوين TAP3
    ├── sip_profiles/           # بوابات SIP (مجلد)
    │   └── gateway_otw.xml
    ├── dialplan/               # قواعد توجيه المكالمات (مجلد)
    │   ├── mo_dialplan.xml     # مكالمات منشأة من الهاتف المحمول
    │   ├── mt_dialplan.xml     # مكالمات مستلمة على الهاتف
المحمول
    │   └── mo_emergency.xml    # توجيه الطوارئ
    ├── omnihss/                # بيانات توفير OmniHSS (مجلد)
    │   ├── subscribers.csv        # المشتركين للتوفير
    │   ├── apn_identifiers.json
    │   ├── apn_qos_profiles.json
    │   ├── apn_profiles.json
    │   ├── eir_rules.json
    │   ├── ims_profiles.json
    │   ├── epc_profiles.json
    │   ├── roaming_rules.json
    │   ├── roaming_profiles.json
    │   ├── flow_information_rules.json
```

```
    │   ├── charging_rules.json
    │   └── pcrf_profiles.json
    └── hnet_udm/               # مفاتيح إزالة إخفاء SUCI لـ UDM
(مجلد)
        ├── curve25519-1.key        # مفتاح خاص X25519 (معرف
1 المفتاح العام للشبكة المنزلية)
        └── curve25519-1.pub        # قيمة عامة خام (hex) لتوفير
USIM
```

المعلمات الشائعة التي تشير إلى

group_vars

المعلمة	المكون	المراجع
`smsc_template_config`	omnimessage	ملف قالب Jinja2 (مثل، `smsc_controller.exs`)
`smsc_smpp_template_config`	omnimessage_smpp	ملف قالب Jinja2 (مثل، `smsc_smpp.exs`)
`mmsc_runtime_template`	mmsc	ملف قالب Jinja2 (مثل، `mmsc_runtime.exs`) - ب
`gateways_folder`	applicationserver	اسم المجلد (مثل، `gateways_prod`)
`dialplan_folder`	applicationserver	اسم المجلد (مثل، `dialpl` م `dialplan` الافتراضي هو يتم تعيينه
`tas_runtime_template`	applicationserver	ملف قالب Jinja2 (مثل، `tas_runtime.exs.j2`) الافتراضي هو `tas_runtime.exs.j2` م يتم تعيينه
`hss_template_config`	omnihss	ملف قالب Jinja2 (مثل، `hss_runtime.exs.j2`)
`mme_template_config`	omnimme	ملف قالب Jinja2 (مثل، `mme_runtime.exs.j2`)
`dra_template_config`	dra	ملف قالب Jinja2 (مثل، `dra_runtime.exs.j2`)

المعلمة	المكون	المراجع
`pgwc_template_config`	pgwc	ملف قالب Jinja2 (مثل، `pgwc_runtime.exs.j2`)
`frr_template_config`	omniupf	ملف قالب Jinja2 (مثل، `frr.conf.j2`)
قوالب SS7	ss7 (أدوار متنوعة)	ملفات قالب Jinja2 (مثل، `stp.j2`، `hlr.j2`، `camel.j2`)
`crm_config.yaml`	omnicrm	ملف تكوين API - يتم **لوب**، يتم الحصول عليه فقط من `{{ group_vars_omnicore }}/crm_config.yaml` (د دور افتراضي)
`grafana_smtp_user`	monitoring	مستخدم SMTP لـ Grafana **لوب**، لا يوجد دور افتراضي
`grafana_smtp_password`	monitoring	كلمة مرور SMTP لـ Grafa **لوب**، لا يوجد دور افتراضي
`grafana_smtp_host`	monitoring	مضيف SMTP لـ Grafana اختياري - الافتراضي هو `in-v3.mailjet.com:587`
توفير OmniHSS	omnihss	دليل `omnihss/` مع `subscribers.csv` + ت للملف الشخصي JSON
مفاتيح HNET	5gc_nf (omniudm)	دليل `hnet_udm/` خاصة يح `<curve>-<id>.key` زالة يتم إنشاء - SUCI إخفاء مف��اح افتراضي

المعلمة	المكون	المراجع
		`curve25519-1.key` إذا كان فارغًا
`secrets.yml`	common/secrets (omnihss وعملاء API الخاصين بها)	ملف Ansible Vault لمفاتيح وكلمات المرور، يتم إدارته باستخدام `scripts/vault_secret` - انظر أسرار العملاء
ملفات تكوين YAML	مكونات متنوعة	ملفات تكوين مباشرة (مثل، `upf_config.yaml`، `crm_config.yaml`)

النقاط الرئيسية

1. **تحتوي group_vars على تخصيصات** - تجاوزات للتكوينات الافتراضية
2. **الإشارة بالاسم** - استخدم معلمات مثل `smsc_template_config` أو `gateways_folder`
3. **تدعم القوالب Jinja2** - الوصول إلى أي متغير Ansible باستخدام `{{ variable_name }}`
4. **تقوم المجلدات بنشر كل شيء** - يتم نسخ جميع الملفات في المجلدات المشار إليها
5. **تحكم في الإصدار لكل شيء** - قم بالتزام جميع group_vars إلى Git

متى تستخدم group_vars

استخدم group_vars لـ:

- قوالب تكوين مخصصة للمكونات
- تعريفات بوابة SIP
- خطط توجيه المكالمات

- قواعد توجيه القطر
- إعدادات خاصة بالعميل تتجاوز الافتراضات

❌ لا تستخدم group_vars لـ:

- استخدم ملف المضيفين - (أسماء المضيفين، IP عناوين) تكوين المضيف الأساسي
- الاختبارات لمرة واحدة - استخدم متغيرات محددة للمضيف في ملف المضيفين
- إذا كانت دائمة group_vars التغييرات المؤقتة - قم بالتعديل على الهدف، والتزام إلى

الوثائق ذات الصلة

- مرجع التكوين - جميع معلمات المضيفين وما تفعله
- تكوين ملف المضيفين - كيفية هيكلة ملفات المضيفين
- ملف - أسرار العملاء `secrets.yml` المشفر في الخزنة وكيفية إدارته
- **تكوين OmniCall**: https://docs.omnitouch.com.au/docs/repos/OmniCall - ما يجب تضمينه في ملفات التكوين
- **تكوين OmniCore**: https://docs.omnitouch.com.au/docs/repos/OmniCore - تفاصيل تكوين المكونات

تكوين ملف المضيفين

نظرة عامة

ملف المضيفين (المعروف أيضًا بملف الجرد) هو الوثيقة المركزية للتكوين التي تحدد نشر شبكة الخلايا بالكامل. يحدد:

- وظائف الشبكة التي يجب نشرها
- أين تعمل (عناوين IP، شرائح الشبكة)
- كيفية تكوينها (معلمات محددة للخدمة)
- إعدادات محددة للعميل (PLMN، الميزات، بيانات الاعتماد)

موقع الملف

تقع ملفات المضيفين تحت الدليل الرئيسي `hosts/` (المحدد كـ `inventory` في `ansible.cfg`)، منظمة حسب العميل:

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

أسماء الملفات عشوائية. يوجد ملف واحد لكل نشر، وعادة ما يكون مسمى باسم النشر أو الموقع (مثل `Production.yml`، `Staging.yml`) بدلاً من نظام ثابت `production.yml`/`staging.yml`/`lab.yml`.

مثال على هيكل ملف المضيفين

يوضح هذا المثال المبسط الأقسام الرئيسية:

```yaml
# مكونات 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'

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

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

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

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

معلمات المضيفين الشائعة

تكوين الشبكة

يتضمن كل مضيف عادةً:

```
pcscf:
  hosts:
    customer-pcscf01:
      ansible_host: 10.10.1.15      # عنوان IP للوصول عبر SSH
      gateway: 10.10.1.1             # البوابة الافتراضية
      host_vm_network: "vmbr1"       # اسم NIC على Hypervisor للاستخدام
```

واستراتيجيات تقسيم الشبكة، انظر معيار IP **ملاحظة:** للحصول على إرشادات حول تخطيط عنوان تخطيط IP الذي يوضح هيكل الشبكة الموصى به بأربعة شرائح لنشر OmniCore.

مستخدمو Proxmox: يحدد معلمة `host_vm_network` الجسر الذي يجب استخدامه. انظر نشر Proxmox VM/LXC للتوفير الآلي.

تخصيص موارد VM

بالنسبة للخدمات التي تحتاج إلى موارد محددة:

```
num_cpus: 4                    # أنوية المعالج
memory_mb: 8192                # الذاكرة RAM بالميغابايت
proxmoxLxcDiskSizeGb: 50       # حجم القرص بالجيجابايت
```

معلمات محددة للخدمة

لكل وظيفة شبكة معلماتها الخاصة. أمثلة:

MME:

```yaml
mme_code: 1                      # معرف MME (1-255)
mme_gid: 1                       # معرف مجموعة MME
network_name_short: Customer     # اسم الشبكة (يظهر على الهواتف)
network_name_long: Customer Network
tac_list: [600, 601, 602]      # رموز منطقة التتبع
```

PGW:

```yaml
ip_pools:                        # تجمعات IP للمشتركين
  - '100.64.16.0/24'
  - '100.64.17.0/24'
combined_CP_UP: false            # فصل التحكم/خطة المستخدم
```

للحصول على شرح مفصل لما يتحكم فيه كل متغير، انظر: مرجع التكوين

خادم التطبيق:

```yaml
online_charging_enabled: true  # تمكين تكامل OCS
tas_branch: "main"             # فرع البرمجيات للنشر
gateways_folder: "gateways_prod"  # تكوين بوابة SIP
```

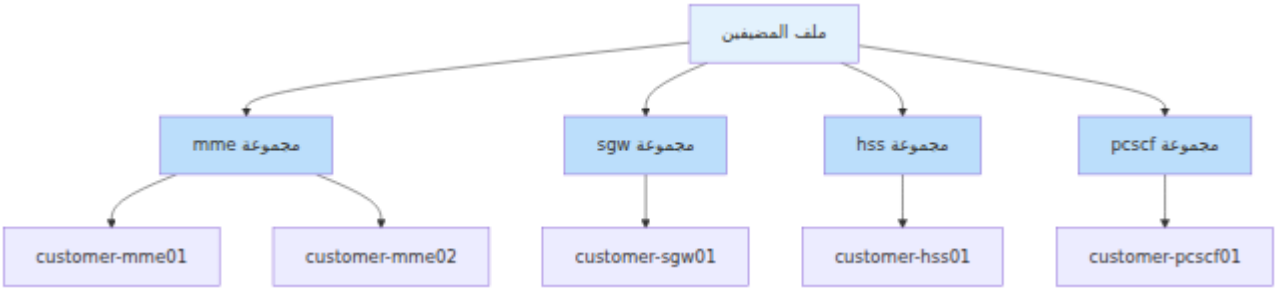
قسم المتغيرات العالمية

يحتوي قسم `all:vars` على إعدادات تنطبق على النشر بالكامل:

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

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

    # معرف PLMN (شبكة الهاتف المحمول)
    plmn_id:
      mcc: '001'              # رمز الدولة المحمول
      mnc: '01'               # رمز شبك الهاتف المحمول
      mnc_longform: '001'     # MNC مع صفر إضافي

    # أسماء الشبكات
    network_name_short: Customer
    network_name_long: Customer Network

    # مستودع APT
    # ملاحظة: إذا تم تعريف مجموعة apt_cache_servers في المضيفين،
    # فإن use_apt_cache يكون افتراضيًا true وapt_repo.apt_server
    # لأحد خوادم التخزين المؤقت IP افتراضيًا يكون عنوان
    apt_repo:
      apt_server: "10.254.10.223"
      apt_repo_username: "customer"
      apt_repo_password: "secure-password"
    use_apt_cache: false

    # المنطقة الزمنية
    TZ: America/Chicago
```

فهم مجموعات المضيفين

تنظم Ansible المضيفين في مجموعات تتوافق مع الأدوار:



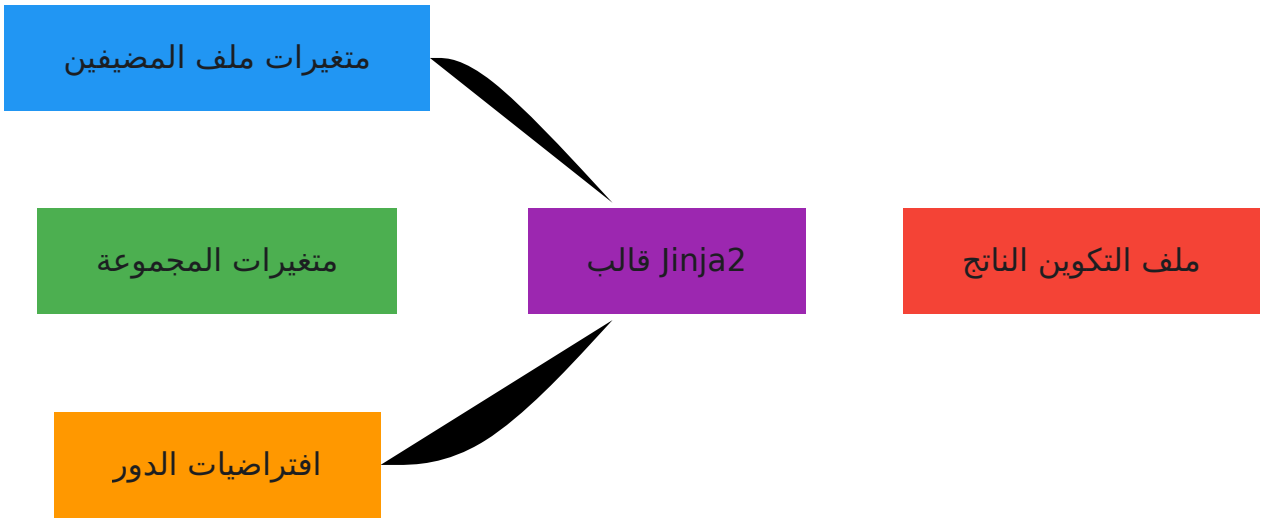
فإنه ينطبق على جميع المضيفين في قسم `mme`، عند تشغيل كتاب اللعب المستهدف `mme:hosts:`.

يوضح الرسم البياني أعلاه عددًا قليلاً فقط من المجموعات. هناك العديد من المجموعات الأخرى، بما في ذلك `ocs`، `crm`، `smsc`، `omnimessage`، `mmsc`، `dra`، `xcap`، `siptrace`، `cbc`، ومجموعات 5GC `amf`، `smf`، و`upf`، من بين آخرين.

التكوين باستخدام قوالب Jinja2

تستخدم Ansible **قوالب Jinja2** لإنشاء ملفات التكوين من المتغيرات المحددة في ملف المضيفين و group_vars.

كيفية عمل Jinja2



مثال على استخدام القالب

يعرف ملف المضيفين:

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

قالب Jinja2 (في الدور):

```
# 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
```

ملف التكوين الناتج:

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

أنماط Jinja2 الشائعة

الوصول إلى المتغيرات المتداخلة:

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

المنطق الشرطي:

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

الحلقات:

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

التنسيق:

```
# إضافة أصفار لـتصبح 3 أرقام
mnc{{ '%03d' | format(plmn_id.mnc|int) }}
```

group_vars تجاوز المتغيرات باستخدام

`group_vars` بينما يحدد ملف المضيفين بنية وإعدادات محددة للمضيف، يمكن أن تتجاوز الافتراضات لمجموعات المضيفين.

انظر: تكوين متغيرات المجموعة

مثال كامل على ملف المضيفين

يحتوي هذا المثال الأكثر اكتمالًا على بيانات حساسة مخفية:

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

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

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

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

# IMS Core
pcscf:
```

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

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

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

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

# خدمات الدعم
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"

# المتغيرات العالمية
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

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

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

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

    # تكوين الشحن
    charging:
```

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

    # انظر — قواعد جدار الحماية docs/firewall.md لمجموعة المعلمات الكاملة
    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']

    # تكوين Hypervisor (مثال Proxmox VM)
    proxmoxServers:
      customer-prxmx01:
        # تم حذف deployment_type → هو الافتراضي "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        # اسم مستخدم cloud-
init اختياري؛ الافتراضي هو أول مفتاح local_users
        proxmoxTemplatePassword: omnitouch    # كلمة مرور cloud-init
اختيارية؛ الافتراضي هو اسم مستخدم cloud-init

    # لموقع LXC، قم بتعيين deployment_type: lxc و
    # proxmoxLxcOsTemplate في كل إدخال proxmoxServers بدلاً من ذلك.
```

انظر نشر Proxmox VM/LXC للحصول على تفاصيل إعداد Proxmox والتكوين الكامل.

مراجع وثائق المنتج

للحصول على تكوين مفصل لكل مكون، يرجى الرجوع إلى الوثائق الرسمية للمنتج:

مكونات OmniCore:

- **وثائق OmniCore**: https://docs.omnitouch.com.au/docs/repos/OmniCore
- **OmniHSS** - خادم المشتركين المنزلي
- **OmniSGW** - بوابة الخدمة (خطة التحكم)
- **OmniPGW** - بوابة الحزمة (خطة التحكم)
- **OmniUPF** - وظيفة خطة المستخدم
- **OmniDRA** - وكيل توجيه قطرات
- **OmniTWAG** - بوابة الوصول إلى WLAN الموثوقة

مكونات OmniCall:

- **وثائق OmniCall**: https://docs.omnitouch.com.au/docs/repos/OmniCall
- **OmniTAS** - خادم تطبيق IMS (VoLTE/VoNR)
- **OmniCall CSCF** - وظائف التحكم في جلسات المكالمات
- **OmniMessage** - مركز الرسا��ل القصيرة
- **OmniMessage SMPP** - دعم بروتوكول SMPP
- **OmniSS7** - مجموعة إشارات SS7
- **VisualVoicemail** - البريد الصوتي

OmniCharge/OmniCRM:

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

الوثائق ذات الصلة

- مقدمة في نشر Ansible - عملية النشر العامة
- **دليل كامل لجميع متغيرات التكوين** - مرجع التكوين
- تكوين متغيرات المجموعة - تجاوز التكوينات الافتراضية

- معيار تخطيط IP - **إرشادات بنية الشبكة وتخصيص IP**
- تكوين Netplan - **عناوين IP المتقدم وتكوين الشبكة الثانوية**
- نظام تخزين APT - توزيع الحزم
- خادم الترخيص - إدارة الترخيص
- نظرة عامة على بنية النشر - عرض كامل للنظام

الخطوات التالية

1. أنشئ ملف المضيفين الخاص بك بناءً على هذا القالب
2. حدد PLMN وهويتك الشبكية
3. قم بتكوين الوصول إلى مستودع APT
4. إعداد خادم الترخيص
5. حسب الحاجة group_vars تخصيص باست��دام
6. نشر باستخدام كتب اللعب Ansible

خادم الترخيص

نظرة عامة

يدير خادم الترخيص تفعيل الميزات لجميع مكونات Omnitouch. يقوم كل مكون بالتحقق من ترخيصه عند بدء التشغيل وبشكل دوري أثناء التشغيل.

الإعداد

1. التعريف في ملف المضيفين

```
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: "اسم العميل"
    license_server_api_urls: ["https://10.10.2.150:8443/api"]
```

2. توفير ملف الترخيص

قم بوضع `license.json` (المقدم من Omnitouch) في `hosts/Customer/group_vars/`

3. النشر

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

مثل) تكون أدلة الجرد الحقيقية مكتوبة بحروف كبيرة ومسبوقة `hosts/Omnicore_Customer/`)، مثل) وملف المضيف ي��مى باسم الموقع `hosts/Omnicore_Customer/host_files/Production.yml`).

https://license_server . للتحقق من حالة جميع التراخيص، انتقل إلى

إزالة ترخيص

لإزالة ترخيص، استخدم `util_playbooks/delete_license.yml`.

متطلبات الشبكة

تكوين جدار الحماية

يجب تكوين جدران الحماية في مواقع العملاء للسماح بحركة مرور HTTPS (المنفذ 443) إلى خوادم التحقق من الترخيص الخاصة بـ Omnitouch:

الغرض	اسم المضيف
خادم التحقق من الترخيص 1	1.time.omnitouch.com.au
خادم التحقق من الترخيص 2	2.time.omnitouch.com.au
خادم التحقق من الترخيص 3	3.time.omnitouch.com.au

القواعد المطلوبة للخروج:

- البروتوكول: HTTPS (TCP/443)
- الوجهة: أسماء المضيفين أعلاه، تم حلها عبر DNS
- الاتجاه: خروج

> **لا تقم بتشفير عناوين IP. حل عبر DNS:** يجب السماح دائمًا لهذه المضيفات بواسطة FQDN (أو بواسطة العناوين التي يقوم جدار الحماية بحلها ديناميكيًا). يمكن أن تتغير عناوين IP الأساسية دون إشعار. القوائم الثابتة لعناوين IP غير مدعومة وستؤدي في النهاية إلى كسر التحقق من الترخيص.

متطلبات DNS

يتطلب خادم الترخيص الخاصة بـ DNS وظيفي للتواصل مع بنية التحقق من الترخيص وجود حل Omnitouch.

تكوين DNS المطلوب:

- يجب أن يكون لخادم الترخيص الوصول إلى خوادم DNS العامة
- قم بتكوين DNS لاستخدام أحد الخيارات التالية:
  - 1.1.1.1 (Cloudflare - يدعم DNS الآمن)
  - 8.8.8.8 (Google Public DNS)
- لا تستخدم خوادم DNS الخاصة بخادم الترخيص الداخلية/الشركات

ملاحظة: تستخدم خوادم ترخيص Omnitouch DNS الآمن (DoH/DoT). يضمن استخدام خوادم DNS بواسطة أجهزة DNS ويمنع المشكلات مع اعتراض DNSSEC العامة التحقق الصحيح من الأمان.

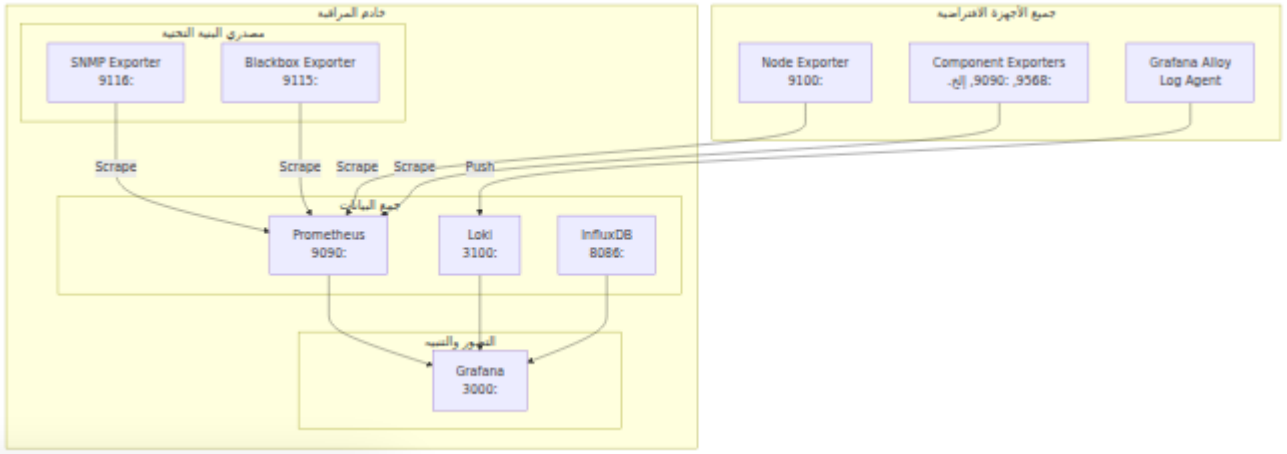
الوثائق ذات الصلة

- مرجع التكوين
- تكوين ملف المضيفين

المراقبة والرصد

نظرة عامة

يتضمن OmniCore مجموعة شاملة من أدوات المراقبة والرصد التي توفر جمع المقاييس، وتجميع السجلات، والتصور، والتنبيه. يتم نشر النظام تلقائيًا بواسطة دور المراقبة.



المكونات

المكون	الغرض	المنفذ	الاحتفاظ بالبيانات
Prometheus	تخزين مقاييس السلاسل الزمنية	9090	30 5 يومًا أو جيجابايت
Loki	تجميع السجلات	3100	50 7 أيام أو جيجابايت
InfluxDB	مقاييس RAN (نوكيا)	8086	30 يومًا
Grafana	التصور والتنبيه	3000	غير متاح
Node Exporter	مقاييس المضيف	9100	غير متاح
SNMP Exporter	مقاييس أجهزة الشبكة	9116	غير متاح
Blackbox Exporter	اختبارات ICMP/HTTP	9115	غير متاح

مصادر البيانات

Prometheus

يقوم Prometheus بجمع المقاييس من جميع مكونات OmniCore كل دقيقة واحدة.

معرف مصدر البيانات: `omnicore_prometheus`

الأهداف المجمعة

اسم الوظيفة	مجموعة الجرد	المنفذ	المقاييس
`Node Exporter`	`all`	9100	وحدة المعالجة المركزية للمضيف، الذاكرة، القرص، الشبكة
`MMEs`	`mme`	9568	عدد الجلسات، معدلات التوصيل والفصل
`HSS`	`hss`	9568	طلبات المصادقة، عمليات البحث عن المشتركين
`SGW-C`	`sgw`	9568	عدد الحاملات، رسائل GTP-C
`PGW-C`	`pgwc`	9090	اتصالات PDN، تخصيصات IP
`UPF Standalone`	`upf`	9090	عدد الحزم، معدل النقل
`CSCF`	`pcscf`, `scscf`, `icscf`	9090	التسجيلات، المعاملات
`ApplicationServer FreeSWITCH`	`applicationserver`	9090	عدد المكالمات، استخدام القنوات
`DRA`	`dra`	9568	قرارات التوجيه، حالة الأقران
`OmniMessage`	`omnimessage`	9568	تسليم الرسائل القصيرة، جلسات SMPP
`OmniMessage MMSc`	`mmsc`	9568	طلبات MM1، إشعارات MM4، تحويلات MMS

المقاييس	المنفذ	مجموعة الجرد	اسم الوظيفة
والداخلية، قرارات على/ خارج الشبكة			
مقاييس بوابة SS7/SIGTRAN	8080	`omniss7`	`OmniSS7`
استخدام الذاكرة، العمليات/ثانية	9121	`ocs`	`KeyDB`
التصنيف، الشحن، إحصائيات CDR	2080	`ocs`	`CGrateS`

مصدري البنية التحتية

متغير التكوين	الأهداف	المصدر
`mikrotik.hosts`	أجهزة التوجيه/المفاتيح	SNMP (MikroTik)
`idrac.hosts`	خوادم Dell	SNMP (iDRAC)
`synology.hosts`	أجهزة NAS	SNMP (Synology)
`saf.hosts`	روابط الميكروويف	SNMP (SAF)
`cisco.hosts`	المفاتيح	SNMP (Cisco)
تلقائي	جميع المضيفين + 8.8.8.8	Blackbox ICMP
`vcenter_ip`, `vcenter_password`	مجموعات vCenter	VMware
`proxmoxServers`	مجموعات PVE	Proxmox

Loki

يتلقى Loki السجلات من وكلاء Grafana Alloy الذين يعملون على جميع الأجهزة الافتراضية.

معرف مصدر البيانات: `omnicore_loki`

للحصول على تفاصيل تكوين Loki/Alloy انظر التسجيل المركزي.

تسميات السجلات

التسمية	الوصف	المثال
`hostname`	اسم مضيف الجهاز الافتراضي المصدر	`customer-mme01`
`component`	نوع المكون	`mme`, `cscf`, `ocs`
`unit`	اسم وحدة systemd	`omnimme.service`
`level`	شدة السجل	`info`, `error`

InfluxDB

تخزن InfluxDB بيانات السلاسل الزمنية لحالات استخدام محددة تتطلب احتفاظًا أطول أو أنماط استعلام مختلفة.

معرف مصدر البيانات: `omnicore_influxdb`

قواعد البيانات

قاعدة البيانات	الغرض	الاحتفاظ
`nokia-monitor`	مقاييس أداء RAN من نوكيا	30 يومًا
`dra`	إحصائيات توجيه DRA	365 يومًا
`Omnicharge_TAP3`	مقاييس ملف TAP للتجوال	90 يومًا

لوحات معلومات Grafana

تنظيم لوحات المعلومات

تُنظم لوحات المعلومات في مجلدات حسب المنطقة الوظيفية:

المجلد	المحتويات
BSS	CGrateS، KeyDB، لوحات معلومات الشحن
EPC	لوحات معلومات MME، SGW، PGW، UPF، HSS، DRA
IMS	لوحات معلومات CSCF، خادم التطبيقات، OmniMessage
Infrastructure	Node Exporter، SNMP، مراقبة الشبكة
RAN	Nokia/OmniRAN وحات معلومات أداء��
Logs	عارضو السجلات الخاصة بالمكونات

توفير لوحات المعلومات (الملفات على القرص)

بنسخ لوحات المعلومات الخاصة Ansible المستودع هو المصدر الوحيد للوحات المعلومات. يقوم الخاصة بالعميل إلى `group_vars/grafana/dashboards` بالدور وملفات `/var/lib/grafana/omnicore-dashboards/` على مضيف Grafana. يقوم ملف مزود في `/etc/grafana/provisioning/dashboards/` بجعل Grafana تقوم بتحميلها.

- لوحات المعلومات للقراءة فقط في واجهة Grafana.
- يتم تحديث لوحة المعلومات المعدلة خلال 30 ثانية.
- يتم حذف لوحة المعلومات المحذوفة.
- يصبح كل دليل مجلدًا بنفس الاسم.
- بعد إعادة التحميل، يقوم البرنامج النصي بحذف أي لوحة معلومات لم تأتِ من القرص.
- يستبدل ملف العميل ملف الدور في نفس المسار.

ملفات مصدر لوحة المعلومات

تُخزن ملفات JSON الخاصة بلوحات المعلومات في مستودع Ansible:

```
roles/monitoring/templates/grafana/dashboards/
├── BSS/
│   ├── KeyDB_Cluster.json
│   ├── cgrates_mysql.json
│   └── cgrates_stats.json
├── EPC/
│   ├── OmniMME.json
│   ├── OmniHSS.json
│   ├── OmniDRA.json
│   ├── OmniPGWC.json
│   ├── OmniUPF.json
│   └── ...
├── IMS/
│   ├── IMS_Status.json
│   ├── OmniMessage_(SMS-C).json
│   └── ...
├── Infrastructure/
│   ├── Node_Exporter_Full.json
│   ├── MikroTik_Dashboard.json
│   └── ...
├── RAN/
│   ├── Nokia_Overview.json
│   ├── Nokia_Detailed.json
│   └── ...
└── Logs/
    └── Logs.json
```

النسخ الاحتياطي للوحة المعلومات

قيد التشغيل إلى المستودع من أجل Grafana قم بتصدير لوحات المعلومات والتنبيهات من مثيل التحكم في الإصدار:

```
# من دليل roles/monitoring
python3 export_grafana.py --customer <CUSTOMER>
```

هذا يصدر:

- جميع لوحات المعلومات إلى `roles/monitoring/templates/grafana/dashboards/{folder}/*.json` (مشاركة بين العملاء)
- قواعد التنبيه إلى `hosts/Omnicore_{CUSTOMER}/group_vars/grafana/alerts/all_alert_rules.json`
- نقاط الاتصال إلى `hosts/Omnicore_{CUSTOMER}/group_vars/grafana/alerts/contact_points.json`
- سياسات الإشعار إلى `hosts/Omnicore_{CUSTOMER}/group_vars/grafana/alerts/notification_policies.json`



سلوك التصدير

- يكتب فقط الملفات التي تم تغييرها (متجاهلاً الحقول المتقلبة مثل `id` و `version`)
- يحافظ على هيكل المجلدات الذي يتطابق مع تنظيم Grafana
- يطابق عناوين لوحات المعلومات مع أسماء الملفات المتسقة

لوحات المعلومات المخصصة

يمكن وضع لوحات المعلومات الخاصة بالعميل في `group_vars/grafana/dashboards/` وسيتم تحميلها جنبًا إلى جنب مع لوحات المعلومات القياسية:

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

التنبيه

الهيكلية



تحميل التنبيه

تأتي قواعد التنبيه من الملفات. المستودع هو المصدر الوحيد للقواعد.

1. يقوم `load_grafana_from_export.py` بدمج قواعد الدور، وقواعد العميل، وتجاوزات الموقع. يكتب ملف توفير واحد Grafana.
2. يقوم Ansible بنسخ هذا الملف إلى `/etc/grafana/provisioning/alerting/omnicore-rules.yaml` ويعيد تحميل التنبيه.
3. قواعد التنبيه للقراءة فقط في واجهة Grafana.

الحذف: يقوم المحمل بحذف كل قاعدة على المضيف غير المجهزة عبر الملفات. يسرد كل قاعدة في الملف الجديد. لذلك، فإن إزالة ملف قاعدة `deleteRules` ملف تم إزالته من القرص تحت يؤدي إلى إزالة القاعدة في التشغيل التالي.

لا تزال نقاط الاتصال وسياسات الإشعار تُحمّل عبر API من `hosts/Omnicore_<CUSTOMER>/group_vars/grafana/alerts/`.

قواعد التنبيه الافتراضية

التنبيه	المجلد	الشرط	لمدة
استخدام مساحة القرص 90%	Infrastructure	نظام الملفات الجذر > 90%	5 دقائق
CPU 90 فوق%	Infrastructure	استخدام CPU > 90%	5 دقائق
حمل النظام فوق 90%	Infrastructure	متوسط الحمل > 90%	5 دقائق
المضيف متوقف	Infrastructure	الهدف في Prometheus غير متاح	5 دقائق
انخفاض مشتركي MME بنسبة 40%	EPC	انخفاض عدد الجلسات بنسبة 40%	30 ثانية
مشتركين على 0 MME	EPC	لا يوجد مشتركين متصلين	5 دقائق
انخفاض مشتركين IMS بنسبة 40%	IMS	انخفاض تسجيلات P-CSCF بنسبة 40%	30 ثانية
أقل من 4 MOS متوسط	IMS	تدهور جودة الصوت	5 دقائق
انخفاض عدد eNodeB	RAN	انخفاض عدد eNB المتصلة	1 دقيقة
تأخير استجابة Diameter مرتفع	EPC	ارتفاع زمن الاستجابة P95	5 دقائق

هيكل قاعدة التنبيه

قواعد التنبيه في `alerts.yaml`:

```yaml
apiVersion: 1
groups:
  - orgId: 1
    name: مجموعة التنبيهات
    folder: Infrastructure
    interval: 1m
    rules:
      - uid: alert-lowDiskSpace
        title: استخدام مساحة القرص على المضيف 90%
        condition: C
        data:
          - refId: A
            datasourceUid: omnicore_prometheus
            model:
              expr: |
                100 - ((node_filesystem_avail_bytes{job="Node
Exporter",mountpoint="/"} * 100)
                / node_filesystem_size_bytes{job="Node
Exporter",mountpoint="/"})
          # ... تعبيرات العتبة
        noDataState: OK
        execErrState: Error
        for: 5m
        annotations:
          summary: استخدام مساحة القرص فوق 90% على المضيف
        labels:
          type: resource
```

تكوين نقاط الاتصال

يتم تكوين نقاط الاتصال عبر متغيرات الجرد:

```yaml
# في group_vars/all.yml أو متغيرات خاصة بالعميل
monitoring_data:
  contactPoints:
    - orgId: 1
      name: الافتراضي
      receivers:
        # تكامل Slack
        - uid: slack-alerts
          type: slack
          disableResolveMessage: false
          settings:
            url: "https://hooks.slack.com/services/xxx/yyy/zzz"

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

تشمل أنواع نقاط الاتصال المدعومة Slack وGoogle Chat والبريد الإلكتروني وPushover (انظر `roles/monitoring/templates/grafana/contact-points.yaml.j2`). يتم تكوين Pushover عبر كتلة `notifications.pushover` (مع `enabled` و`api_token` و`user_key`).

سياسا�ت الإشعار

توجه سياسات الإشعار التنبيهات إلى نقاط الاتصال المناسبة:

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

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

مرجع التكوين

تكوين Prometheus

يقع في `/etc/prometheus/prometheus.yml` على خادم المراقبة.

المعامل	الافتراضي	الوصف
`scrape_interval`	1m	كم مرة يتم جمع البيانات من الأهداف
`evaluation_interval`	1m	كم مرة يتم تقييم قواعد التسجيل
`scrape_timeout`	50s	مهلة طلبات الجمع

تكوين Grafana

يقع في `/etc/grafana/grafana.ini` على خادم المراقبة.

الإعدادات الرئيسية التي يتم تكوينها بواسطة دور المراقبة:

الإعداد	القيمة	الوصف
`allow_embedding`	true	تمكين التضمين في iframes
`provisioning`	`/etc/grafana/provisioning`	دليل التوفير

`admin_password` يتم ترك سطر) `grafana.ini` لم يتم تعيين كلمة مرور المسؤ��ل في الذي يتم تمريره إلى ، `grafana_admin_password` بدلاً من ذلك، يتم تطبيقها عبر متغير .(معلقًا مهام محمل API (محملات لوحة المعلومات/التنبيه) أثناء النشر.

تنبيه البريد الإلكتروني (SMTP)

يتطلب تنبيه البريد الإلكتروني تعيين بيانات اعتماد SMTP في `group_vars` (انظر `roles/monitoring/templates/grafana/grafana.ini`):

المتغير	مطلوب	الافتراضي	الوصف
`grafana_smtp_user`	نعم	*(لا شيء)*	اسم مستخدم SMTP. لا يوجد افتراضي. إذا لم يتم تعيينه، يفشل النشر بصوت عالٍ بدلاً من شحن سر.
`grafana_smtp_password`	نعم	*(لا شيء)*	كلمة مرور SMTP. لا يوجد افتراضي. إذا لم يتم تعيينه، يفشل النشر.
`grafana_smtp_host`	لا	`in-v3.mailjet.com:587`	مضيف SMTP:المنفذ.
`grafana_smtp_enabled`	لا	`true`	ما إذا كان SMTP مفعلًا.

يتم اشتقاق `from_address` و`from_name` من `customer_legal_name` (مثل `alerts+<customer_legal_name>@omnitouch.com.au` و`Omnitouch Alerts - <customer_legal_name>`).

تكوين Loki

انظر التسجيل المركزي للحصول على إعدادات الاحتفاظ والتخزين الخاصة بـ Loki.

تكوين InfluxDB

يقع في `/etc/influxdb/influxdb.conf` على خادم المراقبة.

قاعدة البيانات	سياسة الاحتفاظ	المدة
`nokia-monitor`	`autogen`	30 يومًا
`dra`	`autogen`	365 يومًا
`Omnicharge_TAP3`	`autogen`	90 يومًا

منافذ الخدمة

الخدمة	المنفذ	البروتوكول	الوصف
Grafana	3000	HTTP	واجهة لوحة المعلومات و APIs
Prometheus	9090	HTTP	تخزين المقاييس واستعلامها
Loki	3100	HTTP	استيعاب السجلات واستعلامها
InfluxDB	8086	HTTP	واجهة API لـ InfluxDB
Node Exporter	9100	HTTP	مقاييس المضيف
SNMP Exporter	9116	HTTP	وكيل مقاييس SNMP
Blackbox Exporter	9115	HTTP	مقاييس الاختبار
Alloy	12345	HTTP	واجهة Alloy والمقاييس

العمليات الشائعة

��عرض المقايي

1. افتح Grafana على `http://<monitoring-ip>:3000`
2. انتقل إلى **لوحات المعلومات** واختر المجلد المناسب
3. استخدم محدد نطاق الوقت لضبط نافذة العرض

البحث في السجلات

1. افتح Grafana على `http://<monitoring-ip>:3000`
2. انتقل إلى **استكشاف** واختر مصدر بيانات **Loki**
3. استخدم استعلامات LogQL:

```
# جميع السجلات من مضيف معين
{hostname="customer-mme01"}

# الأخطاء من جميع مكونات MME
{component="mme"} |~ "(?i)error"

# البحث عن IMSI معين
{component="hss"} |= "123456789012345"
```

إنشاء تنبيهات مخصصة

أضف ملف قاعدة إلى دليل `group_vars/grafana/alerts/` الخاص بالعميل، أو إلى دليل `templates/grafana/rules/` الخاص بالدور للحصول على قاعدة يحصل عليها كل موقع. قم بتشغيل برنامج المراقبة. يتم حذف القاعدة التي تم إنشاؤها في واجهة Grafana في التشغيل التالي.

النسخ الاحتياطي للوح المعلومات

بعد إجراء تغييرات في واجهة Grafana، قم بتصديرها إلى المستودع:

```
cd roles/monitoring
python3 export_grafana.py --customer <CUSTOMER> --url
http://<monitoring-ip>:3000
git add ../../hosts/Omnicore_<CUSTOMER>/group_vars/grafana/
git commit -m "تحديث التنبيهات من الإنتاج"
```

مطلوب. يقوم هذا البرنامج النصي بتصدير قواعد التنبيه، ونقاط الاتصال، `--customer` الوسيط الخاص بالعميل؛ لم يعد يقوم `group_vars/grafana/alerts/` وسياسات الإشعار إلى دليل بتصدير لوحات المعلومات، التي يتم مشاركتها بين العملاء في دليل الدور (`roles/monitoring/templates/grafana/dashboards/`).

استكشاف الأخطاء وإصلاحها

متوقف Prometheus الهدف في

الأعراض: لوحة المعلومات تظهر "لا توجد بيانات" أو حالة الهدف تظهر متوقفة

الأسباب المحتملة:

- الخدمة غير قيد التشغيل على المضيف المستهدف
- جدار الحماية يحظر منفذ المصدر
- حل اسم المضيف غير صحيح

الحل:

1. تحقق مما إذا كانت الخدمة قيد التشغيل على الهدف:
   ```
   systemctl status <service>
   curl http://localhost:<port>/metrics
   ```
2. تحقق من الاتصال من خادم المراقبة:
   ```
   curl http://<target>:<port>/metrics
   ```
3. تحقق من صفحة أهداف Prometheus: `http://<monitoring-ip>:9090/targets`

لوحة المعلومات لا تُحمَل

الأعراض: لوحة المعلومات تظهر رمز تحميل أو خطأ

الأسباب المحتملة:

- فشل اتصال مصدر البيانات
- مهلة الاستعلام
- تلف JSON للوحة المعلومات

الحل:

1. اختبار مصدر البيانات في Grafana: **التكوين** → **مصادر البيانات** → **اختبار**
2. تحقق من سجلات Grafana: `journalctl -u grafana-server -f`
3. حاول إعادة تحميل لوحة المعلومات من المستودع

التنبيهات لا تعمل

الأعراض: الشرط مستوفى ولكن لم يتم استلام أي إشعار

الأسباب المحتملة:

- التنبيه متوقف
- نقطة الاتصال غير مكونة بشكل صحيح
- سياسة الإشعار لا تطابق

الحل:

1. تحقق من حالة التنبيه في Grafana: **التنبيه** → **قواعد التنبيه**
2. تحقق من اختبار نقطة الاتصال: **التنبيه** → **نقاط الاتصال** → **اخ��بار**
3. مراجعة توجيه سياسة الإشعار

Grafana لا يمكنها الاتصال بمصدر البيانات

الأعراض: Grafana في "بوابة سيئة" أو مهلة الاتصال

الأسباب المحتملة:

- خدمة Prometheus/Loki/InfluxDB متوقفة
- جدار الحماية يحظر اتصالات localhost
- الخدمة مرتبطة بواجهة خاطئة

الحل:

1. تحقق من حالة الخدمة:
   ```
   systemctl status prometheus loki influxdb
   ```
2. تحقق مما إذا كانت الخدمة تستمع:
   ```
   ss -tlnp | grep -E '9090|3100|8086'
   ```
3. اختبار الاتصال المحلي:
   ```
   curl http://localhost:9090/api/v1/status/config
   curl http://localhost:3100/ready
   ```

مراقبة المشتركين VIP

خدمات الطوارئ، المستشفيات، أجهزة) خدمة المشتركين المختارين VIP تتبع مراقبة المشتركين وقابلية الوصول إلى EPC، ارتباط IMS، تسجيل :(IP التوجيه الرئيسية للعملاء، والهواتف UE. يتم إدارة القائمة في **OmniWeb → المشتركين → المراقبة VIP**، لكل موقع، بواسطة فريق العمليات؛ لا يتعين تغيير أي شيء في هذا المستودع لإضافة أو إزالة مشترك.

الهيكلية



تسأل OmniWeb HSS لعنوان UE من خلال ping الموقع عن كل مشترك مراقب، وتقوم بعمل مصدر blackbox لمضيف الاختبار الذي يوجه إلى مجموعة UE (من المختارة في الصفحة وتعرض النتيجة. يقوم Prometheus، (`monitoring` و `sgwc` و `pgwc` و `pcscf` مجموعات الجرد بجمعها كوظيفة `vip_subscribers`.

أنواع الخدمات

النوع	مطلوب IMS	مطلوب EPC	حالة الاستخدام
الصوت + البيانات (`full`)	نعم	نعم	الأجهزة المحمولة
الصوت فقط (`voip_only`)	نعم	لا	أجهزة VoIP فقط، حسابات IMS بدون SIM (المعرف SIP = اسم مستخدم)
البيانات فقط (`data_only`)	لا	نعم	أجهزة التوجيه، IoT، M2M

`vip_subscriber_service_healthy` هو 1 عندما يتم استيفاء متطلبات النوع ويتم تمكين الحكم أبدًا ping لا يغير .HSS في الحساب.

التكوين

OAM على مضيف OmniWeb بجمع بيانات خدمة Prometheus لا شيء في المخزون. يقوم الخاص بـ nginx ؛ أسماء نقاط النهاية هي المشتركين، لذا فإن `vip_subscribers` كوظيفة OmniWeb يقبل فقط المتصلين من حلقة العودة إلى `/api/vip-subscribers/metrics`.

يتم تعيين مضيف الاستطلاع، وعدد عمليات البينغ لكل فحص، ووقت المهلة وفترة إعادة الفحص على الصفحة (لوحة الاستطلاع)، لكل موقع.

المقاييس

أسماء المقاييس لم تتغير عن المصدر السابق لكل موقع، لذا فإن لوحات المعلومات وقاعدة التنبيه تستمر.

المقياس	الوصف
`vip_subscriber_service_healthy`	1 إذا كان المشترك يفي بالقاعدة لنوعه، وإلا 0. تنبيه على هذا.
`vip_subscriber_ims_registered`	1 إذا تم تعيين S-CSCF
`vip_subscriber_epc_registered`	1 إذا كان MME يحتفظ بالمشترك في الساعة ULR (أو الأخيرة)
`vip_subscriber_hss_reachable`	1 إذا كانت HSS قد أجابت
`vip_subscriber_enabled`	1 إذا كان الحساب مفعلًا في HSS
`vip_subscriber_ims_registration_age_seconds` / `vip_subscriber_epc_registration_age_seconds`	ثوانٍ منذ آخر تسجيل / تحديث الموقع، -1 إذا لم يكن هناك
`vip_subscriber_ue_ip_reachable`, `vip_subscriber_ping_loss_percent`, `vip_subscriber_ping_rtt_{avg,min,max}_ms`, `vip_subscriber_ping_packets_{sent,received}`	نتيجة البينغ؛ تظهر فقط عندما يتم تعيين مضيف الاستطلاع ولدى UE عنوان
`vip_subscriber_info`	دائمًا 1؛ تحمل التسميات `assigned_scscf`, `last_seen_mme`, `ue_ip`, `ue_apn`, `error`
`vip_subscriber_last_check_timestamp_seconds`	الوقت اليونيكس لآخر فحص

التسميات على كل سلسلة: `customer`، `site`، `hss`، `imsi`، `label`، `msisdn` (من HSS)، `type`.

لوحات المعلومات والتنبيه

تأتي لوحات المعلومات **VIP IMS** و **VIP المشتركين** **المشتركين** مجلد) مع دور المراقبة IMS) **خدمة المشتركين** على كل موقع. القاعدة **VIP غير صحية** (`ims-vip-subscriber-service-unhealthy`) تطلق إنذارًا حرجًا عندما يكون `vip_subscriber_service_healthy == 0` على المشترك المتوقف لمدة دقيقة واحدة؛ الصفحة يخرج من المقاييس، لذا فإن انقطاع معروف لا يبقيه نشطًا.

المصدر القديم

كانت النشر السابقة تعمل على مصدر Python على مضيف OAM (`ims-subscriber-exporter`، المنفذ 9550) يتم تغذيته بواسطة `hosts/<customer>/group_vars/vip_subscribers.yaml`. لا يزال الدور يقوم بتثبيته طالما أن هذا الملف موجود ويتقاعد (يوقف الوحدة، يزيل الملفات) بمجرد إزالة الملف. لترحيل موقع: أدخل مشتركيه على صفحة OmniWeb، قم بتعيين مضيف الاستطلاع، أكد أن وظيفة `vip_subscribers` تعمل، ثم احذف ملف YAML وقم بتشغيل الدور.

الوثائق ذات الصلة

- تكوين: تسجيل مركزي بالتفصيل Alloy و Loki
- المعمارية العامة للنظام: معمارية النشر
- تكوين المخزون: تكوين ملف المضيفين

تكوين Netplan

نظرة عامة

يمكن لـ OmniCore تكوين واجهات الشبكة تلقائيًا على الأجهزة الافتراضية الموزعة باستخدام netplan. هذا مفيد لـ:

- إعداد واجهة الإدارة الأساسية (eth0)
- إضافة واجهات ثانوية لعناوين IP المخصصة لحركة المرور، أو اتصالات التبادل العامة
- تكوين المسارات الثابتة لوجهات محددة

تمكين تكوين Netplan

لتمكين تكوين netplan التلقائي لمضيف، أضف المتغير `netplan_config` الذي يشير إلى قالب Jinja2 في مجلد `group_vars` الخاص بك:

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

سيتم الحصول على القالب من `hosts/<customer>/group_vars/netplan.yaml.j2`.

مرجع القالب

لا يوجد `netplan.yaml.j2` مشترك أو قياسي. يقوم كل عميل بإنشاء خاص به وتخ��لف هذه القوالب - `netplan.yaml.j2` في `hosts/<customer>/group_vars/`، عند DNS، اسم الواجهة الأساسية، مجال البحث وتسمية الواجهة الثانوية تختلف حسب العميل. إدخال عميل جديد، أنشئ قالبًا خاصًا بالعميل يتناسب مع تخطيط واجهته بدلاً من نسخ قالب عميل آخر حرفيًا.

فيما يلي قالب عميل واحد كمثال. إنه توضيحي، وليس مواصفة - العملاء الآخرون يختلفون. تم إضافة تعليقات هنا للتوضيح:

```
network:
  version: 2
  ethernets:
    # تستخدم - الواجهة الأساسية ansible_host و gateway من الجرد
    eth0:
      addresses:
      - "{{ ansible_host }}/{{ mask_cidr | default(24) }}"
      nameservers:
        addresses:
{% if 'dns' in group_names %}
        # إذا كان هذا المضيف هو خادم DNS، استخدم DNS الخارجي لتجنب
الاعتماد الدائري
        - 8.8.8.8
{% else %}
        # خلاف ذلك، استخدم خوادم DNS من مجموعة 'dns' في الجرد
{% for dns_host in groups['dns'] | default([]) %}
        - {{ hostvars[dns_host]['ansible_host'] }}
{% endfor %}
{% endif %}
        # مجال البحث DNS - لا خاص بالعميل (يختلف حسب العميل؛ بعضهم
يحددون أي شيء)
        search:
        - customer.example
      routes:
      - to: "default"
        via: "{{ gateway }}"

{% if secondary_ips is defined %}
    # حلقة عبر الواجهات الثانوية dict secondary_ips من الجرد
    # هذا المثال يسميها ens19، ens20، ens21... (18 + loop.index).
هذا
    # خاص بالقالب؛ انظر "تسمية الواجهة" أدناه.
{% for nic_name, nic_config in secondary_ips.items() %}
    ens{{ 18 + loop.index }}:
      addresses:
      # القناع يأتي من mask_cidr الخاص بكل NIC (يكون افتراضيًا 24)
      - "{{ nic_config.ip_address }}/{{ nic_config.mask_cidr |
default(24) }}"
{% if nic_config.routes is defined %}
      # مسارات ثابتة لهذه الواجهة - كل مسار يستخدم بوابة هذه
الواجهة
      routes:
{% for route in nic_config.routes %}
```

```
      - to: "{{ route }}"
        via: "{{ nic_config.gateway }}"
{% endfor %}
{% endif %}
{% endfor %}
{% endif %}
```

النقاط الرئيسية:

- `ansible_host` و `gateway` تأتي من إدخال الجرد الخاص بالمضيف
- `dns` ديناميكيًا من المضيفين في مجموعة DNS يتم سحب خوادم
- مجال البحث DNS (يختلف؛ بعض العملاء يتجاهلونه خاص بالعمي)
- تختلف تسمية الواجهة الثانوية حسب العميل (بعضهم يستخدم `ens19`، `ens20`، ...؛ انظر أدناه)
- يمكن أن يكون لكل IP ثانوي قناع خاص به (`mask_cidr`)، وبوابة، ومسارات ثابتة

تكوين الواجهة الأساسية

يتم تكوين الواجهة الأساسية (eth0) تلقائيًا باستخدام:

- `ansible_host` - عنوان IP
- `gateway` - البوابة الافتراضية
- `mask_cidr` - قناع الشبكة (يكون افتراضيًا 24)

يتم تعيين خوادم DNS تلقائيًا إلى:

- المضيفين في مجموعة `dns` (تستخدم عناوين IP الخاصة بهم `ansible_host`)
- تتراجع إلى DNS `8.8.8.8` إذا كان المضيف هو نفسه خادم

الرابط MTU (`link_mtu`)

يمكن لدور netplan، **الرابط** للواجهة الأساسية بشكل مستقل عن قالب **MTU** تثبيت `common`. عند تعيينه، فإنه يقوم بإنشاء drop-in netplan عبر متغير الجرد `link_mtu` على الواجهة الأساسية للمضيف ويطبقه. هذا موجود (`/etc/netplan/999-link-mtu.yaml`) لأن الأجهزة الافتراضية تأتي من قالب VMware/Proxmox مع أي MTU تم شحنه من مجموعة

مما يؤدي إلى عدم تطابق مع بقية النسيج ويؤدي إلى فقدان، (أحيانًا jumbo/9000) المنفذ المصدر الحزم الكبيرة SCTP/SIP عندما يكون PMTUD معطلًا.

بدمج جميع netplan عمداً بـ `-999` بحيث يتم فرزها **أخيرًا**: يقوم drop-in تم تسمية بترتيب أبجدي مع فوز آخر ملف لكل مفتاح، ويمكن أن تترك عمليات `/etc/netplan/*.yaml` التهيئة ملفات متنافسة تحدد `mtu` على نفس الواجهة (`80` ملف تكوين قديم `-ansible-config.yaml` `99` وملف `-netcfg-vmware.yaml` لمحرك VMware). سيتم تجاوز drop-in ذو الرقم الأقل بهدوء بواسطة تلك.

```
all:
  vars:
    link_mtu: 1500   # تثبيت NIC 1500 على مضيف لكل الأساسي
```

لتطبيقه على مستوى الموقع، أو لكل مضيف للتجاوز. إذا لم يتم تعريف `all.vars` قم بتعيينه في `link_mtu`، فإن الدور يترك MTU الواجهة دون تغيير.

> `link_mtu` **ليس** `mtu`**.** المتغير `mtu` على المستوى الأعلى هو MTU الحمولة GTP/tunnel الذي تستهلكه أدوار packet-core، **ليس** MTU رابط. سيكون (مثل 1400/1430/1300) الواجهة MTU المخصص لـ `link_mtu` الفيزيائي لتلك القيم خاطئًا. استخدم دائمًا NIC تعيين.

الواجهات الثانوية

استخدم، (العامة، التبادل، إلخ IP عناوين) للمضيفين الذين يحتاجون إلى واجهات شبكة إضافية تكوين `secondary_ips`.

المخطط

```
secondary_ips:
  <logical_name>:
    ip_address: <ip_address>
    mask_cidr: <prefix_length>   # قناع لكل - اختياري NIC، 24 افتراضيًا
    gateway: <gateway_ip>
    host_vm_network: <proxmox_bridge>
    vlanid: <vlan_id>
    nic_name: <interface_name>   # تجاوز اسم الواجهة المحدد - اختياري
                                 # (مدعوم من القوالب التي تحترم ذلك)
    routes:                      # مسارات ثابتة عبر هذه الواجهة - اختياري
      - '<destination_cidr>'
      - '<destination_cidr>'
    table_routes:                # متقدم، خاص بالقالب - اختياري
      - { to: '<cidr>', via: '<gateway>', table: <table_id> }
    routing_policy:              # متقدم، خاص بالقالب - اختياري
      from: '<source_cidr>'
      via: '<gateway>'
      table: <table_id>
      priority: <priority>       # 100 افتراضيًا، اختياري
```

> يحدد طول البادئة لتلك الواجهة الثانوية الواحدة. ي��م قراءته من المفتاح `mask_cidr` على `mask_cidr`، وليس من (`nic_config.mask_cidr`) الثانوي IP داخل إدخال المستوى الأعلى. إذا تم تجاهله، فإنه يكون افتراضيًا إلى `/24`.

> تمكّن التوجيه حسب السياسة/المصدر ولكن يتم `routing_policy` و `table_routes` عرضها فقط بواسطة القوالب التي تنفذها. التوجيه المتقدم خاص بالقالب. تأكد من أن للعميل يحترم هذه المفاتيح قبل استخدامها `netplan.yaml.j2`.

تسمية الواجهة

تسمية الواجهة الثانوية خاصة بالقالب - ليست موحدة عبر العملاء:

- بعض العملاء يسميون الواجهات حسب الفهرس: `ens19`، `ens20`، `ens21`، ... (`18 + loop.index`). عند إضافة Proxmox يتطابق هذا مع أسماء الواجهات المعينة بواسطة NICs إضافية إلى VM.
- صراحة عند تعيينه، ويتراجعون إلى `nic_config.nic_name` يستخدم عملاء آخرون خلاف ذلك. (مثل اسم قائم على الفهرس `ens(34 + loop.index)`)

يمكنك تثبيت واجهة ثانوية على اسم محدد عن طريق تعيين `nic_name`، إذا كان القالب يحترم `nic_name` على إدخال IP الخاص بالقالب الثانوي؛ خلاف ذلك، يتم تطبيق مخطط الفهرس.

مثال على التكوين

```yaml
dra:
  hosts:
    <hostname>:
      ansible_host: 10.0.1.100
      gateway: 10.0.1.1
      host_vm_network: "ovsbr1"
      vlanid: "100"
      netplan_config: netplan.yaml.j2
      secondary_ips:
        public_ip:
          ip_address: 192.0.2.50
          mask_cidr: 28
          gateway: 192.0.2.1
          host_vm_network: "vmbr0"
          vlanid: "200"
          routes:
            - '198.51.100.0/24'
            - '203.0.113.0/24'
        peering_ip:
          ip_address: 172.16.50.10
          gateway: 172.16.50.1
          host_vm_network: "ovsbr2"
          vlanid: "300"
          routes:
            - '172.17.0.0/16'
```

Netplan الناتج الناتج عن

تم عرض التكوين أعلاه باستخدام القالب المثال، مما ينتج عنه (مجال البحث واسم DNS خاص بالعميل `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"
```

تظهر واجهة `public_ip` لأن إدخالها حدد `28/` كـ `mask_cidr: 28`، بينما تتراجع `peering_ip` إلى الافتراضي `24/`.

تكامل Proxmox

عند استخدام playbook `proxmox.yml`، يتم إنشاء NICs على الثانوية تلقائيًا VM:

1. تتم إضافة NICs الثانوية أثناء التهيئة الأولية **للأجهزة الافتراضية الجديدة**
2. تتم إضافة NICs الثانوية ويتم إعادة تشغيل VM **للأجهزة الافتراضية الموجودة** لتطبيق التغييرات

تستخدم تكوين Proxmox:

- `host_vm_network` - الجسر الذي يتم توصيل NIC به
- `vlanid` - علامة VLAN للواجهة

كيف يعمل



1. يتم عمل نسخة احتياطية من تكوينات netplan الموجودة في `/etc/netplan` ثم يتم إزالتها لتجنب التعارضات (`*.bak`)
2. يتم عرض قالب Jinja2 الخاص بالعميل إلى `/etc/netplan/01-netcfg.yaml` (وضع `0600`)
3. يقوم `netplan try --timeout 30` بتطبيق التكوين مع التراجع التلقائي إذا فقد الاتصال
4. يؤكد `wait_for_connection` أن المضيف ظل قابلًا للوصول
5. يقوم `netplan apply` بتأكيد التكوين

حالات الاستخدام الشائعة

Diameter Edge Agent (DEA) مع IP عام

```
<hostname>:
  ansible_host: 10.0.1.100          # IP الإدارة الداخلية
  gateway: 10.0.1.1
  netplan_config: netplan.yaml.j2
  secondary_ips:
    diameter_roaming:
      ip_address: 192.0.2.50        # IP عام لشركاء التجوال
      gateway: 192.0.2.1
      host_vm_network: "vmbr0"
      vlanid: "200"
      routes:
        - '198.51.100.0/24'         # شبكة شريك التجوال
```

PGW مع واجهة S5/S8

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

خادم متعدد الاتصال مع شبكات إدارة وبيانات منفصلة

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

الإشارة إلى IPs الثانوية في القوالب

يمكنك الإشارة إلى عناوين IP الثانوية في قوالب Jinja2 وملفات التكوين الأخرى.

على نفس المضيف

عند تكوين خدمة على نفس المضيف الذي يحتوي على IPs أو ثانوية، يمكنك الإشارة مباشرة استخدام `inventory_hostname`:

```
# الإشارة ��باشرة (الأبسط)
{{ secondary_ips.diameter_public_ip.ip_address }}

# أو صراحة عبر inventory_hostname (نفس النتيجة)
{{ hostvars[inventory_hostname]['secondary_ips']
['diameter_public_ip']['ip_address'] }}

# الوصول إلى خصائص أخرى
{{ secondary_ips.diameter_public_ip.gateway }}
{{ secondary_ips.diameter_public_ip.vlanid }}
```

من مضيف آخر

عندما تحتاج إلى الإشارة إلى IP ثانوي لمضيف *مختلف* (على سبيل المثال، تكوين اتصال نظير)، استخدم `hostvars` مع اسم المضيف المستهدف:

```
# الإشارة إلى المضيف الأول في مجموعة dra
{{ hostvars[groups['dra'][0]]['secondary_ips']
['diameter_public_ip']['ip_address'] }}

# حلقة عبر جميع مضيفي DRA والحصول على عناوين IP الخاصة بهم
{% for host in groups['dra'] %}
{% if hostvars[host]['secondary_ips'] is defined %}
  - {{ hostvars[host]['secondary_ips']['diameter_public_ip']
['ip_address'] }}
{% endif %}
{% endfor %}
```

مثال: تكوين نظير DRA

تكوين نظير diameter للربط إلى IP العام الخاص به:

```
# في dra_config.yaml.j2 - استخدم inventory_hostname للمضيف الحالي
peers:
  - name: external_peer
    # الربط إلى IP diameter العام لهذا المضيف
    local_ip: {{ hostvars[inventory_hostname]['secondary_ips']
['diameter_public_ip']['ip_address'] }}
    remote_ip: 198.51.100.50
    port: 3868
```

التحقق مما إذا كانت IPs الثانوية موجودة

تحقق دائمًا مما إذا كانت المتغيرات موجودة قبل استخدامها:

```
{% 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 %}
```

استكشاف الأخطاء وإصلاحها

تحقق من أسماء الواجهات

قم بتسجيل الدخول إلى VM وتحقق من أسماء الواجهات:

```
ip link show
```

الناتج المتوقع لجهاز افتراضي به واجهتين ثانويتين:

```
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> ...
```

تحقق من تكوين Netplan

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

تطبيق Netplan يدويًا

```
netplan apply
```

تصحيح Netplan

```
netplan --debug apply
```

تحقق من المسارات

```
ip route show
```

الوثائق ذات الصلة

- تكوين ملف المضيفين - إعداد جرد المضيف
- نشر Proxmox VM/LXC - تهيئة VM
- مرجع التكوين - جميع متغيرات التكوين

سجلات الأجهزة الشبكية

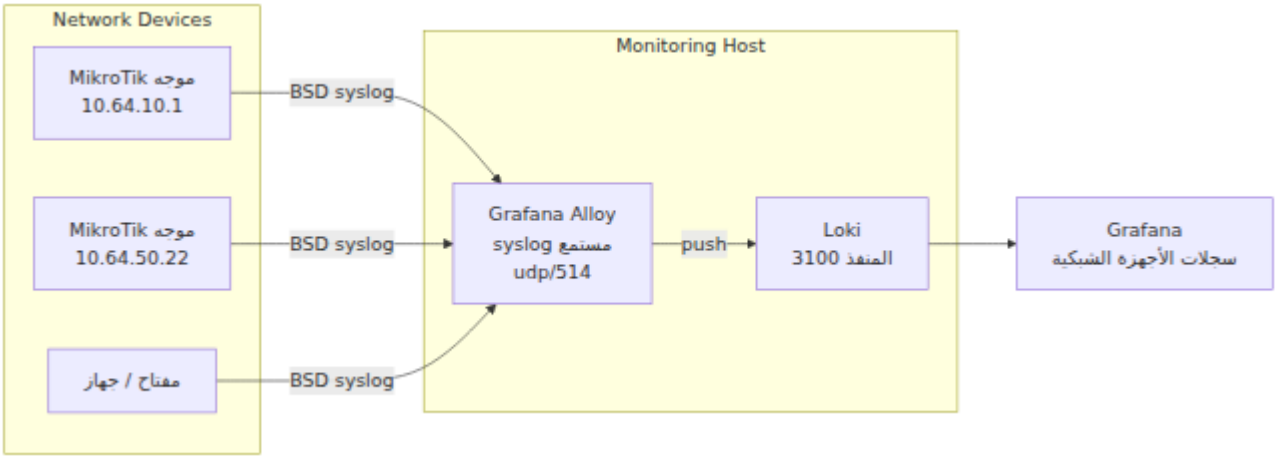
لا يمكن لجهاز التوجيه والمفاتيح وغيرها من الأجهزة الشبكية تشغيل عميل Grafana Alloy. بدلاً من ذلك، ترسل هذه الأجهزة سجلات syslog. يقوم كل مضيف مراقبة بتشغيل مستمع syslog المحلي. ثم تجلس سجلات الأجهزة بجانب سجلات Loki يكتب المستمع الرسائل في داخل Alloy. الشبكة الأساسية، وتشارك نفس البحث، ونطاق الوقت، والاحتفاظ.

جدول المحتويات

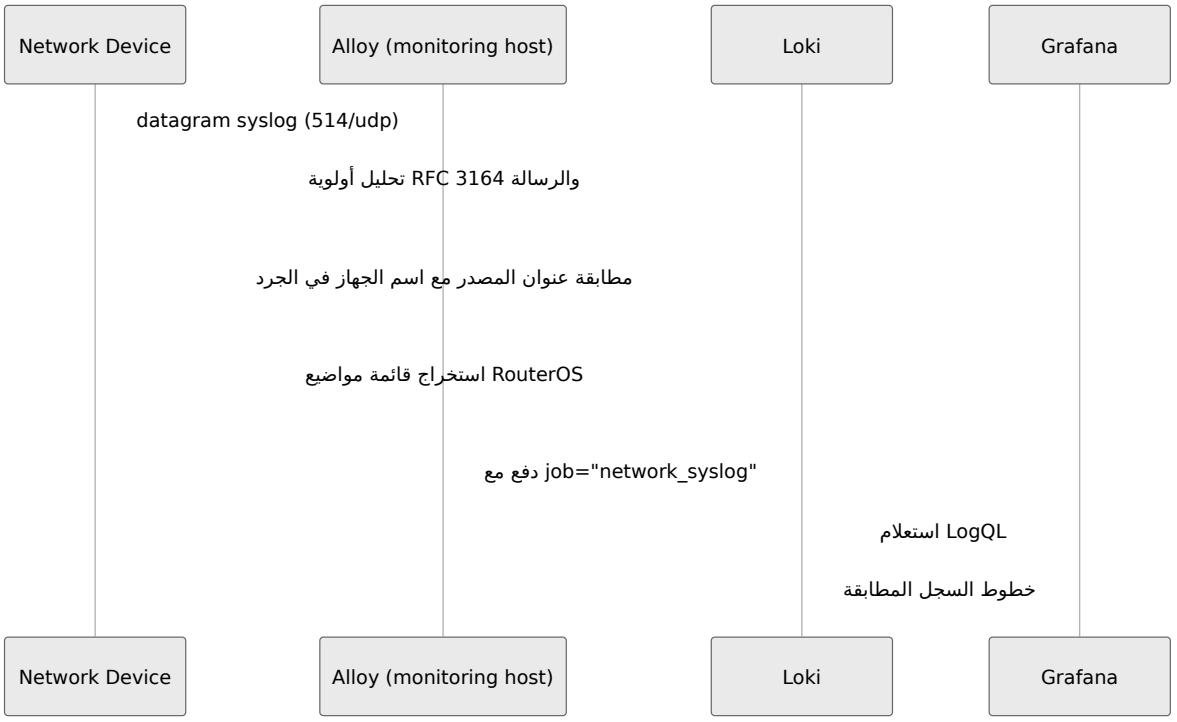
- العمارة
- كيفية تحديد الأجهزة
- تكوين المجمع
- جرد الأجهزة
- علامات السجل
- MikroTik RouterOS
- مرسلات syslog الأخرى
- جدار الحماية
- لوحة المعلومات
- استعلام السجلات
- استكشاف الأخطاء وإصلاحها
- المراجع

العمارة

المجمع هو جزء من بناء المراقبة القياسي. يفتح كل مضيف مراقبة المستمع. لا يقوم الموقع بتعيين علامة لاستقبال سجلات syslog للأجهزة.



يقوم Alloy بتحليل RFC 3164، وهو التنسيق الذي تحدده BSD syslog. يقبل المستمع سجلات العلامات التي تصفها الأقسام التالية Alloy الأولوية، والطابع الزمني، والرسالة. ثم يضيف.



كيفية تحديد الأجهزة

يسمي Alloy كل جهاز من عنوان المصدر للبيانات. يقارن Alloy هذا العنوان مع خرائط `devices:` في جرد الموقع. يحدد التطابق علامات `device` و `vendor`.

تتبع هذه التصميم نتيجتين:

- تكون علامة الجهاز صحيحة حتى عندما يرسل الجهاز هوية خاطئة، أو هوية فارغة، أو هوية مكررة في رأس الرسالة.

- يحتفظ الجهاز الذي يرسل من عنوان غير مدرج في الجرد بعلامة `device_ip` فقط. أضف العنوان إلى الجرد لإعطاء هذا الجهاز اسمًا.

إذا كان للجهاز أكثر من وا��هة واحدة، تأكد من أنه يرسل من العنوان الذي يسجله الجرد. قم بتعيين عنوان المصدر على الجهاز.

تكوين المجمع

قم بتجاوز معلمة في جرد `roles/common/defaults/main.yml`. تعيش معلمات المجمع في الموقع تحت `all.vars`.

```
all:
  vars:
    network_syslog_port: 514          # منفذ المستمع
    network_syslog_protocol: udp      # النقل
    network_syslog_vendors:           # كتل البائع التي تسمي مرسل
      - mikrotik
```

المعلمة	النوع	مطلوب	الافتراضي	الوصف
`network_syslog_port`	عدد صحيح	لا	`514`	المنفذ الذي يرتبط به المستمع على مضيف المراقبة. المنفذ أقل من 1024 هو مميز. انظر الملاحظة أدناه.
`network_syslog_protocol`	سلسلة	لا	`udp`	النقل للمستمع. استخدم `udp` أو `tcp`. ترسل معظم الأجهزة الشبكية UDP.
`network_syslog_vendors`	قائمة	لا	`[mikrotik]`	كتل البائع في الجرد التي تسمي مرسل syslog. يساهم كل بائع مدرج في علامات اسم الجهاز.

لذلك يحتاج المنفذ الأقل من 1024 إلى قدرة. يتلقى `alloy`. كم��تخدم غير مميز Alloy يعمل تلقائيًا عندما يكون `CAP_NET_BIND_SERVICE` على مضيف المراقبة Alloy وحدة `network_syslog_port` القدرة 1024 عن يزيد الذي المنفذ تتلقى لا .1024 من أقل.

قم بتطبيق تغيير على هذه المعلمات باستخدام دور `common`:

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

لهذا المستمع. لذلك، يرسل مضيف المراقبة أيضًا سجلات المجلة، Alloy تشغل مضيفات المراقبة وسجلات التدقيق إلى SSH، و Loki المحلي الخاص به.

جرد الأجهزة

أضف كل جهاز مرة واحدة. ثم تعطي SNMP. يقرأ المجمع نفس خرائط البائع التي يستخدمها مُصدِر SNMP، وقاعدة جدار حماية، الإدخال الجهاز علامة سجل، وهدف.

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

الوصف	الافتراضي	مطلوب	النوع	المعلمة
اسم الجهاز إلى عنوان IP. يصبح الاسم علامة `device`. يحدد العنوان المرسل ويفتح جدار الحماية.	-	نعم	خريطة	`devices`
مجتمع SNMP لمُصدِر SNMP الخاص بـ Prometheus. لا يستخدمه جامع syslog.	`public`	لا	سلسلة	`snmp_community`

علامات السجل

تصل سجلات syslog للأجهزة تحت `job="network_syslog"`. تنطبق العلامات التالية.

العلامة	الوصف	مثال
`job`	قيمة ثابتة لسجلات syslog للأجهزة.	`network_syslog`
`component`	قيمة ثابتة لسجلات syslog للأجهزة.	`network_device`
`device`	اسم الجهاز من الجرد. يطابق عنوان المرسل Alloy.	`nf-nids-rtr-core01`
`device_ip`	عنوان المصدر للبيانات.	`10.64.10.1`
`vendor`	كتلة بائع الجرد التي تحمل الجهاز.	`mikrotik`
`level`	شدة syslog.	`informational`, `warning`, `err`, `crit`
`facility`	مرفق syslog.	`daemon`
`topics`	قائمة مواضيع RouterOS الكاملة من رأس الرسالة.	`system,info,account`
`topic`	الموضوع الأول في تلك القائمة. استخدمه للتصفية الشائعة.	`system`, `firewall`, `dhcp`
`syslog_host`	حقل المضيف في رأس syslog، كما كتبه المرسل.	`system,info,account`

لا يرسل RouterOS هوية خاصة به. يكتب قائمة مواضيعه حيث ينتمي اسم المضيف، لذا يخزن من هذا الحقل. يحدد `topic` و `topics` يقرأ المجمع syslog. المحلل تلك القائمة كمضيف `syslog_host` المرسل الذي يكتب اسم مضيف حقيقي فقط، ولا ينشئ أي علامات موضوع.

MikroTik RouterOS

قم بتكوين كل جهاز من واجهة سطر الأوامر RouterOS. يشير الجهاز إلى إجراء التسجيل المدمج الخاص به إلى مضيف المراقبة. يحتاج الجهاز بعد ذلك إلى قاعدة تسجيل واحدة `remote` لكل موضوع.

```
/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
```

الخاصية	القيمة	الوصف
`target`	`remote`	يرسل السجل إلى خادم syslog عن بُعد بدلاً من الذاكرة أو القرص.
`remote`	عنوان مضيف المراقبة	وجهة تدفق syslog.
`remote-port`	`514`	المنفذ الوجهة. تطابق `network_syslog_port`.
`remote-log-format`	`bsd-syslog`	RouterOS 7 يرسل فقط. RFC 3164. انظر جدول النسخة أدناه.
`src-address`	عنوان الجهاز في الجرد	اختياري. قم بتعيينه عندما يكون للجهاز أكثر من واجهة واحدة، بحيث يتطابق عنوان المصدر مع إدخال الجرد.

RouterOS تنسيق الرسالة حسب إصدار

لذا يجب أن يرسل الجهاز ذلك التنسيق. تغيرت الخاصية التي RFC 3164، يقوم المجمع بتحليل تتحكم في التنسيق بين الإصدارات الرئيسية. تحقق من الإصدار باستخدام `/system resource print`.

الإصدار	الخاصية	الافتراضي	الإجراء
RouterOS 7	`remote-log-format`	`default`	قم بتعيين `remote-log-format=bsd-syslog`. القيمة `default` ترسل تأطير syslog، وهو بدون رأس أولو��ة MikroTik ما لا يمكن للمجمع تحليله. الخاصية `bsd-syslog` غير موجودة هنا، في RouterOS 6 وتفشل تلك الأمر مع `bad parameter bsd-syslog`.
RouterOS 6	`bsd-syslog`	`no`	قم بتعيين `bsd-syslog=yes`. القيمة الافتراضية ترسل رسالة عارية، وهو ما لا يمكن للمجمع تحليله.

```
# 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
```

تظهر التقاطات الحزم الفرق. يتم فك تشفير التنسيق الخاطئ كـ `(invalid)` في `tcpdump`، وتبدأ الحمولة باحتمالات ثنائية قبل نص الرسالة. يتم فك تشفير التنسيق الصحيح كـ `SYSLOG, Facility local0, Severity info`.

مواضيع RouterOS

الموضوع	الوصف
`critical`	الأعطال التي توقف خدمة على الجهاز.
`error`	العمليات الفاشلة، على سبيل المثال، فشل تأجير DHCP أو نفق فاشل.
`warning`	الظروف التي تحتاج إلى اهتمام ولكن لا توقف خدمة.
`info`	الأحداث العادية. تحمل هذه الموضوعات أحداث `account`، وهي تسجيلات دخول المستخدم، وتسجيلات الدخول الفاشلة، وتغييرات التكوين.

عندما تريد تدقيق تسجيل `info` لحجم منخفض. أضف `critical` و `warning` و `error` ابدأ بـ الدخول وتغييرات التكوين.

تحقق من النتيجة على الجهاز:

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

مرسلات syslog الأخرى

يمكن لأي جهاز يتحدث BSD syslog تنطبق ثلاث شروط. استخدام المجمع:

1. يرسل الجهاز BSD syslog، كما هو محدد في RFC 3164. لا يقوم المجمع بتحليل RFC 5424.
2. يرسل الجهاز UDP إلى المنفذ 514 على مضيف المراقبة، ما لم يغير الموقع `network_syslog_protocol` أو `network_syslog_port`.
3. يرسل الجهاز من عنوان مدرج في خريطة `devices:`. يسمح جدار الحماية بذلك العنوان، ويعطي العنوان الجهاز اسمه.

أضف بائعًا جديدًا إلى `network_syslog_vendors` لتسمية أجهزته. يقبل جدار الحماية بالفعل أجهزة كل بائع.

Cisco IOS

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

Linux rsyslog

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

احتفظ بذلك القالب. لا تحدد RFC 3164. هو القالب الافتراضي لـ rsyslog RFC 5424. لأن هذا القالب يرسل `RSYSLOG_SyslogProtocol23Format`.

الأجهزة ذات واجهة الويب

قم بتعيين خادم syslog عن بُعد والمنفذ. إذا كان الجهاز يقدم خيار تنسيق، اختر Legacy أو BSD.

جدار الحماية

تأتي `monitoring` إلى مجموعة `udp/` بـ `514` `b` يحمل نموذج التدفق القاعدة. يسمح التدفق `4` المصادر من الرمز `{ net_devices: true }`، الذي يحل كل عنوان في كل خريطة `all.vars.<vendor>.devices`.

يتم إسقاط الجهاز الذي لا يسجله الجرد عندما يقوم الموقع بتشغيل جدار الحماية في وضع `enforce`. أضف الجهاز إلى الجرد، ثم قم بتطبيق جدار الحماية:

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

انظر إلى دليل جدار الحماية حسب الدور لنموذج التدفق والأوضاع.

لوحة المعلومات

تحتفظ Grafana بلوحة المعلومات **سجلات الأجهزة الشبكية** في مجلد **السجلات**.

اللوحة	الوصف
بحث سجل الأجهزة الشبكية	بحث نصي حر عبر كل جهاز.
معدل السجل حسب الجهاز	الحجم لكل جهاز مع مرور الوقت. يظهر خط مستوي عند الصفر جهازًا توقف عن الإرسال.
معدل السجل حسب الشدة	الحجم لكل شدة مع مرور الوقت.
سجلات الأجهزة الشبكية	جميع سجلات الأجهزة للجهاز المحدد وسلسلة البحث.
الأخطاء، التحذيرات وحرجة	الشدة `warning` وما فوق.
تسجيلات الدخول، فشل المصادقة وتغييرات التكوين	أحداث الحساب والنظام للتدقيق.
أحداث الرابط، OSPF/BGP وDHCP	أحداث الواجهة، والتوجيه، وتأجير.

المتغير	الوصف
`الموقع (السجلات)`	مصدر بيانات Loki. اختر الموقع.
`الجهاز`	قائمة منسدلة للجهاز. تأتي القائمة من قيم علامة `device` في Loki. اختر جهازًا واحدًا، أو عدة أجهزة، أو `الكل`.
`البحث`	فلتر نصي حر للوحات السجل.

استعلام السجلات

```
# كل رسالة من جهاز واحد
{job="network_syslog", device="nf-nids-rtr-core01"}

# الشدة التحذيرية وما فوق، عبر الملكية
{job="network_syslog", level=~"emerg|alert|crit|err|warning"}

# تسجيلات الدخول وتغييرات التكوين
{job="network_syslog"} |~ `(?i)(logged in|login failure|changed
by)`

# موضوع واحد من RouterOS
{job="network_syslog", topic="firewall"}

# معدل الرسائل لكل جهاز
sum by (device) (rate({job="network_syslog"} [5m]))

# الأجهزة التي توقفت عن الإرسال في الساعة الماضية
sum by (device) (count_over_time({job="network_syslog"} [1h]))
```

استكشاف الأخطاء وإصلاحها

لا توجد سجلات أجهزة في Loki

الأعراض: استعلام عن `{job="network_syslog"}` لا يعيد أي نتائج.

الأسباب المحتملة:

- لا يعمل Alloy على مضيف المراقبة.
- لم يرتبط المستمع بالمنفذ.
- لا يرسل الجهاز.
- يقوم جدار الحماية بإسقاط البيانات.

الحل:

1. تحقق من حالة العميل على مضيف المراقبة باستخدام `systemctl status alloy`.

2. تحقق من أن Alloy يمتلك المنفذ باستخدام `ss -lunp | grep 514`.
3. التقاط حركة المرور باستخدام `tcpdump -n -i any udp port 514`. تثبت الحركة أن الجهاز يرسل وأن جدار الحماية يسمح بالبيانات.
4. إذا لم تكن هناك حركة، تحقق من إجراء التسجيل على الجهاز.

Alloy لا يبدأ

الأعراض: يتم إعادة تشغيل خدمة `alloy` في حلقة. تظهر المجلة خطأ ربط أو خطأ إذن للمنفذ 514.

الأسباب المحتملة:

- الوحدة لا تمتلك `CAP_NET_BIND_SERVICE`.
- يحتفظ عملية أخرى بالمنفذ، على سبيل المثال، `rsyslogd` محلي مع مستمع UDP.

الحل:

1. اقرأ الخطأ باستخدام `journalctl -u alloy -n 50`.
2. تأكد من القدرة باستخدام `systemctl cat alloy`.
3. ابحث عن المستمع الآخر باستخدام `ss -lunp | grep 514`. أوقف ذلك المستمع، أو قم بتعيين `network_syslog_port` لتتناسب Alloy إلى منفذ مجاني وقم بتكوين الأجهزة.

Loki شيئًا تصل البيانات ولكن لا يخزن

الأعراض: تظهر `tcpdump` حركة المرور على المنفذ 514. لا يحتفظ Loki بتدفق `network_syslog`.

الأسباب المحتملة:

- لا يرسل الجهاز سجلات BSD syslog. يحتاج جهاز RouterOS 7 إلى `remote-log-format=bsd-syslog`. يحتاج جهاز RouterOS 6 إلى `bsd-syslog=yes`.
- يرسل الجهاز RFC 5424.

الحل:

1. التقاط الحركة باستخدام `tcpdump -n -i any -A udp port 514`. الرسالة التي تميزها `tcpdump` كـ `(invalid)` ليس لديها رأس أولوية syslog، لذا فإن التنسيق خاطئ.

2. قم بتعيين خاصية التنسيق لإصدار الجهاز. انظر تنسيق الرسالة حسب إصدار RouterOS.
3. على جهاز آخر، اختر تنسيق syslog BSD أو Legacy.
4. تحقق من سجل Alloy لأخطاء التحليل باستخدام `journalctl -u alloy | grep -i syslog`.

علامة الجهاز فارغة

الأعراض: تحمل خطوط السجل `device_ip` ولكن لا توجد علامة `device`.

الأسباب المحتملة:

- عنوان المرسل غير موجود في خريطة `devices:`.
- يرسل الجهاز من واجهة ثانية، لذا يختلف عنوان المصدر عن عنوان الجرد.

الحل:

1. اقرأ عنوان المصدر من علامة `device_ip`.
2. أضف ذلك العنوان إلى خريطة `devices:` للبائع، أو قم بتعيين عنوان المصدر على الجهاز.
3. قم بتطبيق التغيير باستخدام دور `common`، ومع تشغيل جدار الحماية إذا كان الموقع يفرض جدار الحماية.

توقف جهاز واحد عن الإرسال

الأعراض: تظهر لوحة **معدل السجل حسب الجهاز** عدم وجود بيانات لجهاز واحد. تستمر الأجهزة الأخرى.

الأسباب المحتملة:

- أزال مشغل أو عطل قاعدة التسجيل على الجهاز.
- فقد الجهاز المسار إلى مضيف المراقبة.
- تغير عنوان الجرد، لذا يقوم جدار الحماية بإسقاط عنوان المصدر الجديد.

الحل:

1. تحقق من القواعد على الجهاز باستخدام `/system logging print`.
2. قم بعمل ping لمضيف المراقبة من الجهاز.

3. تأكد من أن العنوان الحالي للجهاز يتطابق مع إدخال الجرد.

المراجع

العنوان	المرجع
برو BSD syslog توكول	RFC 3164
بروتوكول Syslog. لا يقوم المجمع بتحليل هذا التنسيق.	RFC 5424
خط أنابيب سجل المضيف، الاحتفاظ، ولوحات المعلومات	التسجيل المركزي
نموذج التدفق، الرموز المصدر، والأوضاع	جدار الحماية حسب الدور
والقياسات Grafanaو Prometheus	المراقبة والملاحظة

عمليات النسخ المتماثل وعقد OmniHSS

تقوم OmniHSS بتخزين بيانات المشتركين، والمصادقة (AuC)، وبيانات IMS في قاعدة بيانات PostgreSQL محلية على كل عقدة. **النسخ المتماثل المنطقي ثنائي الاتجاه لـ PostgreSQL** متزامنة. يشكل هذا **شبكة نشطة-نشطة**: كل OmniHSS يبقي عدة عقد من عقدة هي رئيسية كاملة للقراءة/الكتابة، والكتابة على أي عقدة تنتشر إلى جميع العقد الأخرى. لا وقبول التهيئة في أي Sh وCx/Dx وS6a يوجد رئيس/احتياطي. يمكن لأي عقدة خدمة حركة مرور وقت.

وتحميلها OmniHSS يغطي هذا الدليل نموذج النسخ المتماثل. يشرح كيفية إضافة عقدة جديدة من بالبيانات. كما يشرح كيفية التحقق من الصحة، وإصلاح، وإزالة الأقران.

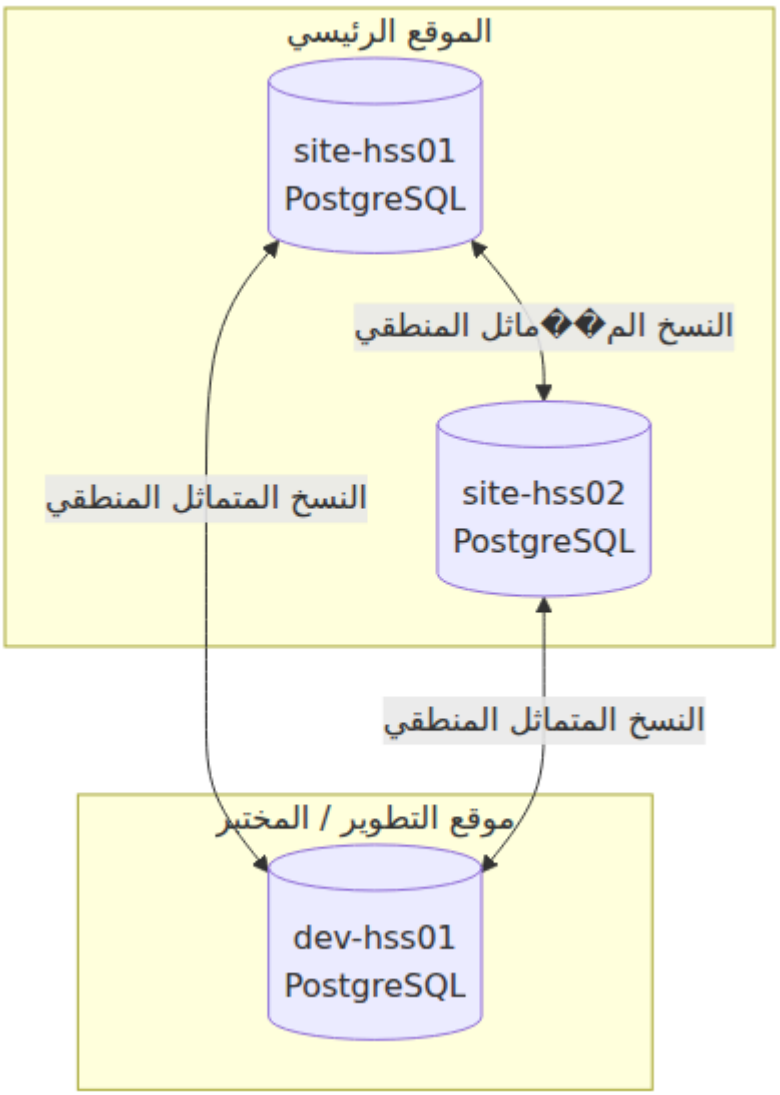
جدول المحتويات

- نظرة ��امة على العمارة
- كيفية اكتشاف الأقران
- التكوين
- إضافة عقدة جديدة من OmniHSS
- تحميل عقدة جديدة بالبيانات
- التحقق من النسخ المتماثل
- سلامة البيانات
- إزالة نظير
- النسخ الاحتياطي والاستعادة
- المقاييس
- استكشاف الأخطاء وإصلاحها

نظرة عامة على العمارة

تشغل كل عقدة من OmniHSS مثيل PostgreSQL خاص بها. كل عقدة **تنشر** جميع جداول التطبيق الخاصة بها و**تشترك** في نشر كل نظير. `origin = none` يمنع حلقات النسخ المتماثل.

من خلال **(LWW)** على جداول الحالة سريعة التغير، تحل الشبكة تعارضات الكتابة **الأحدث يفوز** مقارنة `updated_at`.

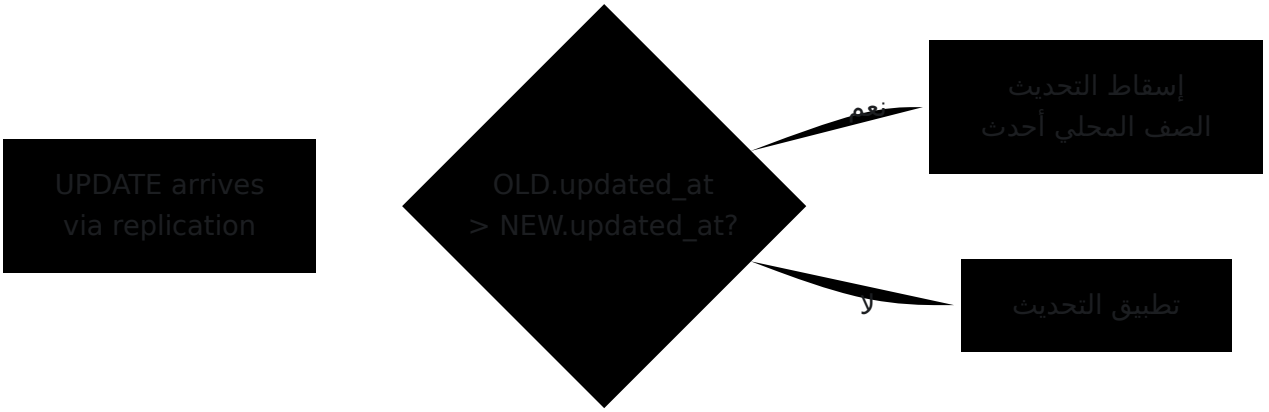


عقد، `N` كل رابط أعلاه هو اشتراك واحد الاتجاه (واحد لكل اتجاه). بالنسبة لشبكة مكونة من من فتحات النسخ المتماثل للنشر `N-1` اشتراكات. تستهلك الأقران `N-1` تحتفظ كل عقدة بـ الخاص بها.

ما الذي يتم نسخه متماثلًا

فئة الجدول	أمثلة	معالجة التعارض
البيانات الرئيسية / المهيأة	`subscriber`, `key_set` (AuC), `sim`, `msisdn`	لا حاجة: يتم تهيئتها من مكان واحد في كل مرة
الحالة الديناميكية	`subscriber_state`, `pdn_session`, `lte_call`, `ims_subscription`	يتم تفعيل الأحدث يفوز على `updated_at`
محاسبة الهجرة	`schema_migrations`	**مستبعدة** من النسخ المتماثل

تشغل كل عقدة) `schema_migrations` **باستثناء** `public` يغطي النشر كل جدول في مخطط عندما تقوم عقدتان بتحديث نفس صف الحالة في وقت .(ترحيلاتها الخاصة بشكل مستقل بدلاً من أي تحديث وصل آخر `updated_at` على أحدث LWW و��حد، تحافظ مشغلات.



كيفية اكتشاف الأقران

بناء قائمة الأقران تلقائيًا من مصدرين OmniHSS عندما يتم تمكين النسخ المتماثل، يقوم دور:

1. المعرفة OmniHSS **المحلية**: عقد `hss` **المضيفون الآخرون في مجموعة جرد** (لا `hss01` / `hss02` على سبيل المثال، زوج محلي) في *نفس* ملف الجرد/المضيف ؛ يكفي أن تكون عضواً في `connected_omnihss` تحتاج هذه إلى أن تكون مدرجة في المجموعة.

2. **قائمة** `connected_omnihss` في ملف الأقران البعيدة للموقع: عقد OmniHSS المعرفة في *ملف مضيف* مختلف (مواقع أخرى). لا يمكن اكتشاف الأقران عبر الملفات الخاصة بآخر `hss` إلا هنا، لأن جرد واحد ليس لديه معرفة بمجموعة.

يقوم الدور بتصفية الأقران من المجمو��ة المحلية إلى المضيفين الذين هم مشاركون في النسخ المتماثل لـ OmniHSS: أولئك الذين تكون قيمة `omnihss.replication.enabled` لديهم `true`. العقدة التي ليست علامتها `true` لا تُلتقط كقرين للنسخ المتماثل.

يستبعد الدور المضيف الحالي من قائمة الأقران الخاصة به تلقائيًا، لذا يمكن مشاركة نفس كتلة `connected_omnihss` من قبل كل موقع دون محاولة عقدة النسخ المتماثل من نفسها.

تسمية الكائنات

تسمي كل عقدة كائنات النسخ المتماثل الخاصة بها بحيث لا تتصادم شبكة بأي حجم:

الكائن	الاسم	تم إنشاؤه على	فريد لكل
النشر	`<self>_pub`	هذه العقدة	العقدة
الاشتراك	`sub_from_<source>`	هذه العقدة (المشترك)	(المشترك)
فتحة النسخ المتماثل	`<subscriber>_from_<source>`	المصدر (الناشر)	(المشترك، الزوج (المصدر

يجب أن يتضمن اسم الفتحة **المشترك**. فكر في ما يحدث إذا كان يتضمن فقط المصدر (حيث ترث الفتحة اسم الاشتراك، PostgreSQL الافت��اضي لـ) ستقوم عقدتان تشتركان في نفس المصدر بمحاولة إنشاء فتحة بنفس الاسم على ذلك المصدر. ستفشل الاشتراك الثاني بعد ذلك مع *"فتحة النسخ المتماثل موجودة بالفعل"*. إن تضمين المشترك يمنح كل رابط فتحة فريدة عالميًا.

المصالحة وأمان `--limit`

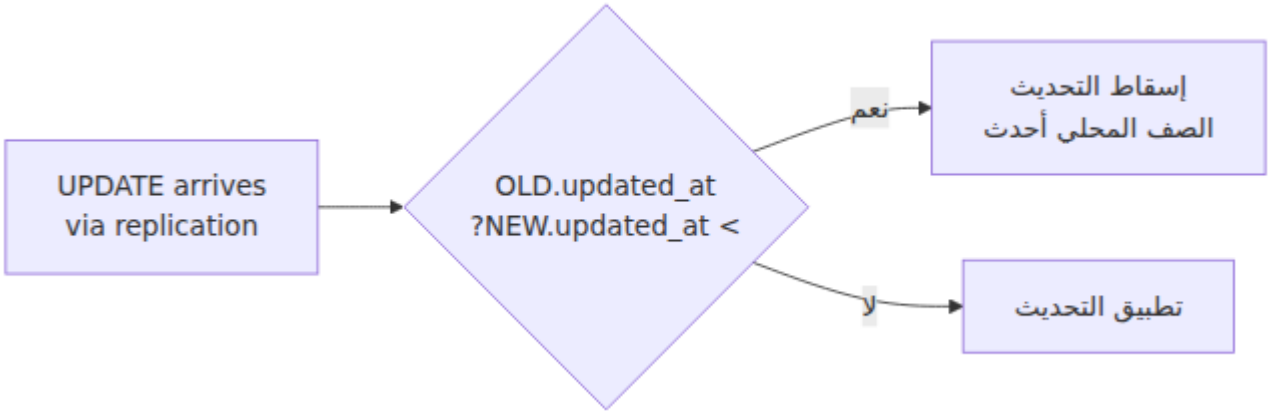
كل تشغيل للدور **يُصالح** كائنات النسخ المتماثل للعقدة مقابل مجموعة الأقران الصالحة الحالية (مجموعة `hss` المحلية + `connected_omnihss`، ناقص الذات):

- تقوم بتنظيف المنشورات القديمة (`<self>_pub` يتم إسقاط المنشورات الأخرى غير المعاد تسميتها تلقائيًا).
- يتم تعطيل الاشتراكات التي لم يعد نظيرها في المجموعة الصالحة، ويتم فصلها عن فتحتها البعيدة، وإسقاطها.
- يتم إسقاط الفتحات المنطقية غير النشطة التي لا تغذي نظيرًا صالحًا حاليًا. لا يتم لمس الفتحات النشطة أبدًا.

هي القائمة **السلطوية** للشبكة: يقوم التشغيل التالي بتقليم `connected_omnihss` هذا يجعل من كل عقدة. تمنع خاصيتان أمانيتان من (ذات الصلة `hss` ومن مجموعات) نظير تمت إزالته منها أن يكون هذا مدمرًا:

- **آمن.** يتم اشتقاق المجموعة الصالحة من `groups['hss']` و`--limit` والتي تعكس دائمًا عضوية الجرد *الكاملة* بغض النظر عن `connected_omnihss`، ترى `hss01` ولكن لا تزال `hss01`. تشغيل `--limit hss01` يُصالح فقط `hss01` أبدًا بتقليم `--limit` كصالحين وتحافظ على اشتراكاتهم. لا يقوم `hss02`/`hss03` الأقران المستبعدة.
- **المجموعات الفارغة لا تقلم أبدًا.** يتم تخطي خطوات التقليم تمامًا بدلاً من مسح كل النسخ المتماثل إذا كانت مجموعة الأقران الصالحة تحسب إلى فارغة (على سبيل المثال، جرد جزئي مع HSS واحد و `connected_omnihss` غير محملة).

> **تحذير القائمة السلطوية:** نظرًا لأن التقليم تلقائي، يقوم التشغيل بتقليم نظير عبر ملف `connected_omnihss` هو جزء حقيقي من الشبكة و**كنه مفقود من `connected_omnihss` مكتمل لكل نظير عبر ملف.



التكوين

تمكين النسخ المتماثل

يتم التحكم في النسخ المتماثل لكل نشر بواسطة علامة `omnihss.replication.enabled`. تكون `false` بشكل افتراضي. قم بتعيينها في جرد الموقع أو group_vars:

```
omnihss:
  database_port: 5432
  database_name: "omnihss"
  database_username: "hss"
  database_password: "password"
  replication:
    enabled: true
```

المعلمة	النوع	مطلوب	الافتراضي	الوصف
`database_port`	عدد صحيح	لا	`5432`	منفذ PostgreSQL، يُستخدم أيضًا كمنفذ افتراضي لاتصالات الأقران.
`database_name`	سلسلة	لا	`omnihss`	سم قاعدة�� البيانات. يجب أن يتطابق عبر جميع الأقران.
`database_username`	سلسلة	نعم	`hss`	دور تسجيل الدخول. تم إنشاؤه مع خاصية `REPLICATION` حتى يتمكن من دفع النسخ المتماثل المنطقي.
`database_password`	سلسلة	نعم	`password`	كلمة مرور لدور تسجيل الدخول. يجب أن تتطابق عبر جميع الأقران (تستخدم في سلاسل اتصال الأقران).
`replication.enabled`	منطقي	لا	`false`	مفتاح رئيسي. عندما يتم `true`، تكون تكوين PostgreSQL للنسخ المتماثل المنطقي ويتم إنشاء المنشورات/الاشتراكات.

عندما تكون `replication.enabled` هي `true`، يقوم الدور بتطبيق إعدادات PostgreSQL التالية على المثيل المحلي:

الإعداد	القيمة
`wal_level`	`logical`
`max_wal_senders`	`10`
`max_replication_slots`	`10`
`listen_addresses`	`*`
`max_slot_wal_keep_size`	`10GB` (يمكن تجاوزها عبر `omnihss.replication.max_slot_wal_keep_size`)

القيمة	الإعداد

يتم تحديث `pg_hba.conf` بعنوان IP للسماح لكل من اتصالات (`32/`) لكل نظير و `all` لكل `replication`.

إعلان الأقران عبر المواقع (`connected_omnihss`)

يتم إعلان عقد OmniHSS البعيدة في ملف الأقران البعيدة المشار إليه بواسطة متغير `remote_peers_file` الخاص بكل موقع (عادةً `prod_master_peers.yml`):

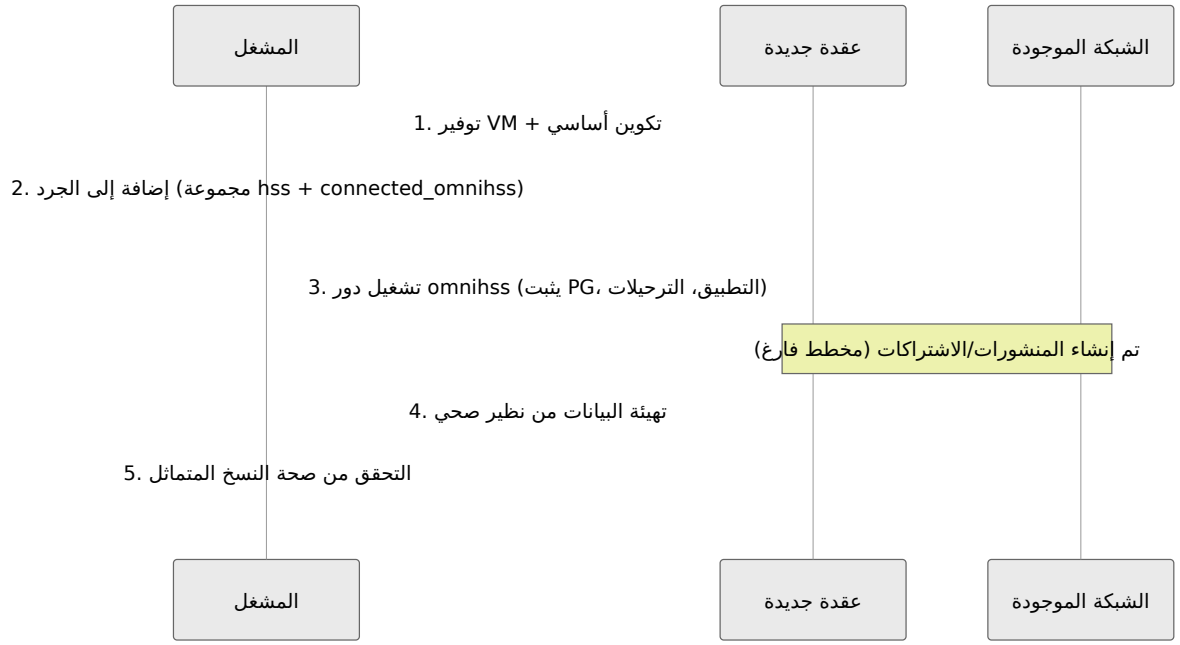
```
connected_omnihss:
  - name: site-hss01
    host: 10.4.2.142
    port: 5432
  - name: site-hss02
    host: 10.4.2.143
    port: 5432
  - name: dev-hss01
    host: 10.4.10.142
    port: 5432
```

المعلمة	النوع	مطلوب	الافتراضي	الوصف
`name`	سلسلة	نعم	-	للنظير. هذا `inventory_hostname` يسمي الاشتراك (`sub_from_<name>`) ويحل نشر النظير (`<name>_pub`)، لذا **يجب** أن يتطابق مع اسم المضيف الفعلي للنظير.
`host`	سلسلة	نعم	-	عنوان IP للنظير، يمكن الوصول إليه من هذه العقدة عبر شبكة الإشارة.
`port`	عدد صحيح	لا	`5432`	منفذ PostgreSQL للنظير. يعود إلى `database_port`.

نظرًا لأن ملف الأقران البعيدة يتم تحميله بواسطة كل موقع، يجب أن تحتوي القائمة الكاملة لـ `connected_omnihss` في الشبكة. تتخطى كل موقع تلقائيًا عقده OmniHSS على **كل** عقد (المحلية `hss`) لأنها مغطاة بالفعل بواسطة مجموعة المحلية الخاصة وتتخطى نفسها.

إضافة عقدة جديدة من OmniHSS

تتكون إضافة عقدة من مرحلتين متميزتين: **الانضمام إلى الشبكة** (حتى تتدفق التغييرات في كلا الاتجاهين) و**تحميل البيانات الموج��دة** على العقدة الجديدة. تهم المرحلة الثانية لأن العقدة الجديدة تمامًا تبدأ بقاعدة بيانات فارغة. انظر تحميل عقدة جديدة بالبيانات.



الخطوة 1: توفير المضيف

قم بتوفير VM في أي عقدة والشبكات الأساسية كما هو الحال في OmniCore (انظر نشر Proxmox). يجب أن تصل العقدة إلى منفذ PostgreSQL (`5432`) بشكل افتراضي عبر شبكة الإشارة، ويجب أن تتمكن الأقران من الوصول إليها.

الخطوة 2: إضافة العقدة إلى الجرد

أضف المضيف الجديد إلى ملف مضيف الموقع تحت مجموعة `hss:`:

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

في ملف الأقران البعيدة حتى تتعلم **المواقع الأخرى** عنها `connected_omnihss` ثم أضفها إلى أيضًا:

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

الخطوة 3: تشغيل دور OmniHSS

قم بتشغيل كتاب خدمة PostgreSQL ضد المضيف الجديد. يقوم هذا بتثبيت OmniHSS، ويقوم بتكوين النسخ المتماثل المنطقي، وينشئ النشر بالإضافة إلى اشتراك لكل نظير:

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

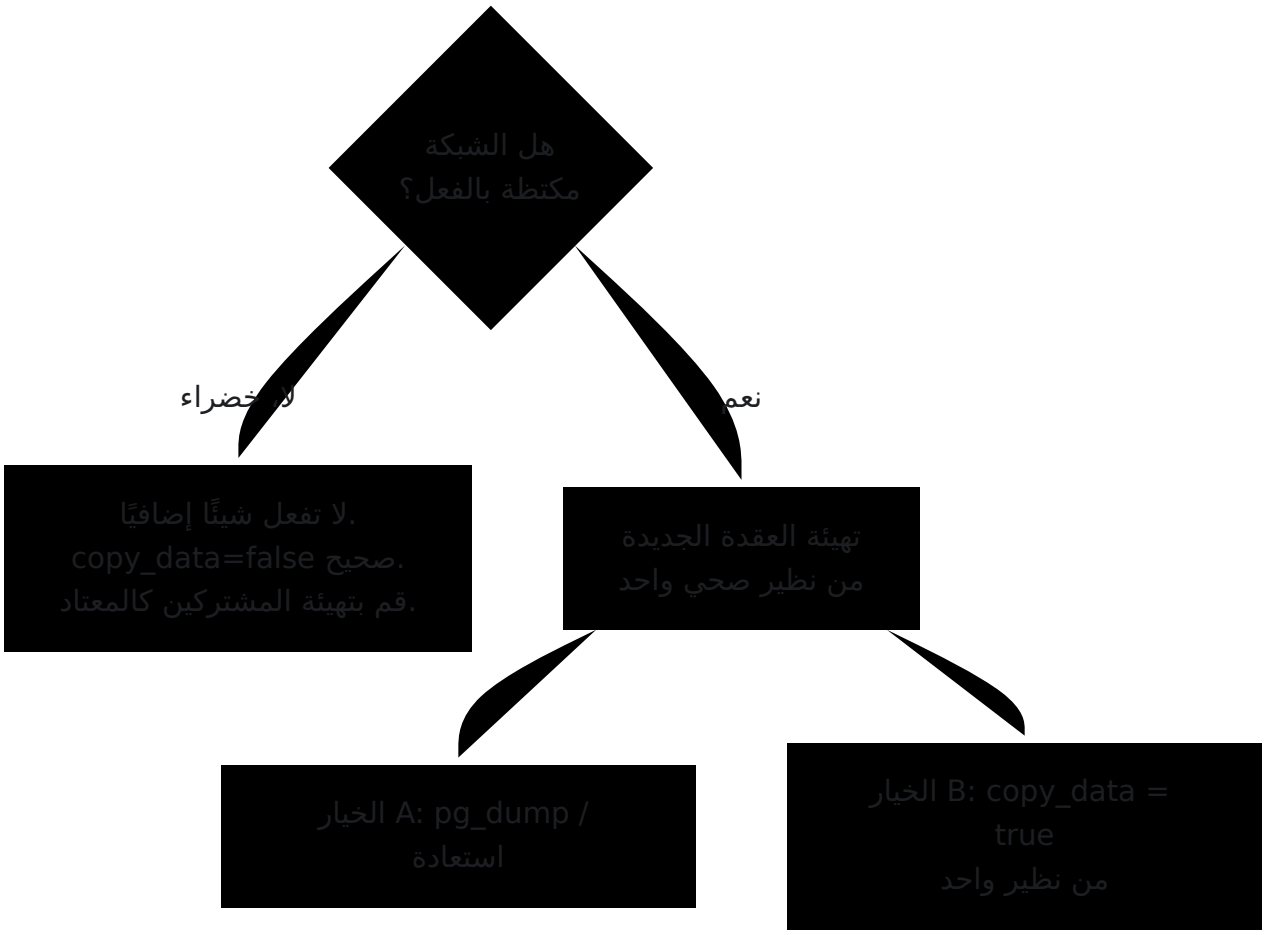
أعد تشغيل الدور على الأقران الموجودة (أو دع التشغي�المجدول التالي يتعامل معها) حتى تفتح `pg_hba.conf` للعقدة الجديدة وتقوم بإنشاء اشتراكها مرة أخرى. الاشتراكات غير متكررة: يكتشف الدور ويخطي الاشتراكات الموجودة، ويصدر `ALTER SUBSCRIPTION ... REFRESH PUBLICATION` حتى تلتقط أي جداول جديدة.

في هذه المرحلة، أصبحت العقدة الجديدة عضوًا في الشبكة وتستقبل **جميع الكتابات المستقبلية**. لكنها لا تحتوي بعد على قاعدة المشتركين الموجودة. تابع لتحميل بياناتها.

تحميل عقدة جديدة بالبيانات

هذا صحيح لشبكة **خضراء** (تبدأ كل `copy_data = false`. ينشئ الدور الاشتراكات الآلية مع عقدة فارغة في نفس الوقت: لا يوجد شيء لنسخه، والنسخ سيسبب تعارضات في الصفوف المكررة). ولكن يعني أن العقدة التي تنضم إلى شبكة **مكتظة بالفعل** تبقى فارغة باستثناء التغييرات التي تحدث بعد انضمامها.

اختر المسار الذي يتناسب مع وضعك:



شبكة خضراء

معًا دون بيانات مسبقة، فلا تحتاج إلى تهيئة. قم OmniHSS إذا كنت تقوم بتشغيل جميع عقد بتمكين النسخ المتماثل، وشغل الدور عبر جميع العقد، وتهيئة المشتركين بشكل طبيعي. كل كتابة تتكرر إلى جميع الأعضاء.

الخيار A: تهيئة عبر التفريغ والاستعادة (موصى به)

خذ لقطة من نظير صحي واستعدها على العقدة الجديدة. ثم دع النسخ المتماثل يحمل التغييرات المستمرة. يعيد هذا استخدام أدوات النسخ الاحتياطي/الاستعادة القياسية وهو النهج الأكثر قابلية للتنبؤ، بما في ذلك لمجموعات البيانات الكبيرة.

1. لقاعدة بيانات `pg_dump` **قم بعمل نسخة احتياطية من نظير صحي.** ينتج هذا `--no-publications --no-subscriptions`، لذا لا يتم تم أخذه مع OmniHSS) على آلة التحكم (نقل كائنات النسخ المتماثل:

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

يتم حفظ التفريغ كـ `backups/<Site_Name>/hss_dump_<hostname>_<timestamp>.sql`.

2. **استعادة على العقدة الجديدة.** يقوم كتاب الاستعادة بإيقاف OmniHSS، وإسقاط وإعادة إنشاء قاعدة البيانات، واستعادة التفريغ، وإعادة تشغيل الخدمة. يأخذ نسخة احتياطية قبل الاستعادة للهدف أولاً:

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

أضف `-e skip_pre_restore_backup=true` لتخطي تلك النسخة الاحتياطية قبل الاستعادة — مفيد عندما يكون الهدف عقدة جديدة أو فارغة بالفعل، أو عندما يكون التفريغ بطيئًا ومحتوياته الحالية معروفة بأنها غير ذات قيمة. لا يتم حفظ أي شيء، لذا لا يمكن استرداد الب��انات الحالية للعقدة.

من الخطوة 1 واترك مسار التكوين فارغًا عند المطالبة، قدم المسار إلى تفريغ SQL (لأن العقدة الجديدة لديها تكوينها الخاص من تشغيل الدور).

3. **تأكيد تدفق النسخ المتماثل.** الآن تنطبق الاشتراكات الموجودة للعقدة الجديدة (`copy_data = false`) أدناه التحقق. قم بتشغيل التغييرات فوق اللقطة المستعادة.

> **نافذة الصيانة:** هناك نافذة صغيرة بين التفريغ واللحظة التي يلحق فيها النسخ المتماثل. خلال ذلك، لم تعد تغييرات التهيئة على الأقران موجودة بعد على العقدة الجديدة. تتغير بيانات المشترك الرئيسية بشكل غير متكرر، وتتعافى الحالة الديناميكية (`subscriber_state`, `pdn_session`, `lte_call`, `ims_subscription`) من حركة المرور الحية، لذا فإن نافذة النشاط المنخفضة كافية. لا تقم بتهيئة مشتركين جدد خلال التهيئة.

الخيار B: تهيئة باستخدام كتاب التهيئة (موصى به)

يقوم كتاب المرافق `seed_hss_replication.yml` بالتهيئة من **نظير** مختار واحد دون عيوب النسخ الضخم الخام. يعيد إنشاء الاشتراك كاشتراك فقط للبث (`copy_data = false`، لا يوجد جداول متزامنة). ثم يقوم بملء الصفوف الموجودة في **جدول الرئيسي** بتحميل آمن من التعارض. `subscriber_state` 1:1 يشير كل مشترك إلى) **هو** جزء من الملء: إنه بيانات هيكلية

تركه يعني أن كل مشترك تم ملؤه يصبح .(تم إنشاؤه مرة واحدة عند التهيئة، FK واحد بواسطة على الهدف. لا يمكن للبث إصلاح ذلك، لأن النسخ المنطقي يسقط S6a وAPI يتيمًا، مما يكسر بهدوء تحديثات الصفوف التي لا توجد. لا يقوم الكتاب عمدًا بنسخ الجداول المؤقتة حقًا لتلك الجداول الساخنة (مفتاح مكرر) مع الكتابات `COPY` يتصادم (`pdn_session`, `lte_call`). المتزامنة على نظام حي ويؤدي إلى دورات تحطم لجداول التزامن. يتم إعادة ملئها من النسخ المتماثل البث وحركة المرور الحية بدلاً من ذلك.

تفاعلي (موصى به): اترك المصدر. سيجد الكتاب بعد ذلك الأهداف الفارغة، ويستفسر عن الأقران التي تحمل بيانات، ويطلب منك الاختيار:

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

```
الأقران المتاحة للتهيئة منها:
  1) dev-hss01 (10.4.10.142) - 838 مشترك
أدخل رقم المصدر الذي تريد التهيئة منه
```

صريح: قم بتسمية المصدر مباشرة. هذا يجبر على إعادة التهيئة حتى لو كانت الهدف تحتوي بالفعل على بيانات (استخدم لاسترداد):

```
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"
```

للحصول على مرجع كامل للمعلمات والسلوك HSS انظر تهيئة النسخ المتماثل لـ.

أيضًا **بتوصيل الشبكة الكاملة** بعد التهيئة: يقوم بتشغيل `seed_hss_replication.yml` يقوم بتشغيل المصالحة النسخ المتماثل عل كل عقدة (المنشورات، الاشتراكات، الفتحات) والتحقق من صحتها. يُفضل تشغيله بدون `--limit` — والعقد يتم اكتشاف العقد الفارغة تلقائيًا كأهداف تهيئة، المملوءة كمصادر، ويتم توصيل كل عقدة في تمريرة واحدة. يضيق `--limit` كل من التهيئة والتوصيل، لذا تترك عقدة واحدة محدودة الشبكة نصف موصولة؛ استخدمه فقط لإعادة تهيئة مستهدفة، ثم أعد التشغيل بدون `--limit` لتوصيل الباقي. قم بالتهيئة من مصدر واحد فقط. يُفضل A الخيار لمجموعات البيانات الكبيرة جدًا.

لاحظ) تتم الخطوات اليدوية المعادلة على العقدة المستهدفة. أولاً، قم بإنشاء الاشتراك البث
`slot_name` تسمية الكائنات ثم قم بملء جداول الرئيسية بأمان من .(الصريح لكل رابط، انظر
التعارض مع تخطي جداول الحالة الساخنة:

```
-- 1. بث جميع الجداول، بدون نسخ ضخم
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. ملء جداول الرئيسية فقط (تشغيل على الهدف؛ كل جدول باستثناء -t
#    schema_migrations، pdn_session، lte_call - يجب تضمين
subscriber_state،
#    يشير :subscriber_state_id أو يصبح كل مشترك تم ملؤه يتيمًا
الخاص به إلى صف لا يصل أبدًا، مما يكسر
#    API/S6a، ولا يمكن للبث إصلاحه لأن النسخ المنطقي
#    يسقط بهدوء تحديثات الصفوف المفقودة).
#    يجب أن يتم تحميلها مع أصل النسخ المتماثل: origin=none
#    تقوم الاشتراكات بتصفية التغييرات الموقعة عند الناشر، لذا فإن
مثل WAL الملء غير مرئي للشبكة. يتم فك تشفير التحميل غير الموقّع من
أي
#    كتابة محلية ويعيد صدى الإدراج إلى الأقران الذين يحملون الصفوف
بالفعل،
#    مما يؤدي إلى توقف عمال تطبيقهم بسبب مفتاح مكرر.
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');"
```

التحقق من النسخ المتماثل

تقوم الوظيفة بإجراء فحص صحي تلقائيًا بعد الإعداد، ويمكن أيضًا تشغيلها بمفردها. يؤكد التحقق أن:

- كل اشتراك لديه عملية عامل حية.
- كل فتحة نسخ منطقية نشطة.
- تأخير WAL لكل فتحة أقل من 100 ميجابايت.
- حالة WAL لكل فتحة هي `محجوزة` أو `موسعة` (وليس `مفقودة`).

للتحقق من الصحة عند الطلب، أعد تشغيل وظيفة OmniHSS (يتم تشغيل التحقق في النهاية)، أو تفقد مباشرة على عقدة:

```
# يجب أن يكون - عمال الاشتراك pid غير فارغ لكل واحد
sudo -u postgres psql -d omnihss -c \
  "SELECT subname, pid IS NOT NULL AS worker_alive FROM
pg_stat_subscription;"

# يجب أن تكون نشطة - فتحات النسخ true، حالة wal_status /محجوزة
موسعة
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';"
```

العرض	المعنى
`worker_alive = false`	عامل الاشتراك متوقف: النظير غير قابل للوصول أو بيانات الاعتماد/سلسلة الاتصال خاطئة.
`active = false` (فتحة)	لا يوجد مشترك يستهلك الفتحة: اشتراك النظير متوقف أو تم إسقاطه.
تأخير متزايد `lag`	النظير بطيء أو الرابط متدهور. يتراكم ويستهلك القرص WAL.
`wal_status = lost`	الفتحة تأخرت كثيرًا وتمت إزالة WAL بواسطة PostgreSQL. يجب عليك **إعادة تهيئة** النظير (انظر تحميل عقدة جديدة بالبيانات)

PostgreSQL HSS أيضًا لوحة **نسخ** (`health_check.yml`) يتضمن تقرير الفحص الصحي للعقد التي تم تمكين النسخ فيها، مما يظهر حالة عامل كل اشتراك وحالة كل فتحة نشطة وحالة WAL في لمحة.

سلامة البيانات

تطبيق النسخ المنطقي لا يفرض المفاتيح الخارجية. يعمل عامل التطبيق مع `session_replication_role = replica`، مما يقمع المشغلات الداخلية التي تنفذ قيود RI `FOREIGN KEY`. لذلك، لا يقوم PostgreSQL من صف يصل عبر النسخ **بإعادة التحقق أبدًا** مقابل الوالد الخاص به. تعطي قيود FK *سلامة لكل عقدة*، وليس *سلامة على مستوى الشبكة* لك.

في شبكة نشطة-نشطة تفتح ثغرة لا يمكن لعقدة واحدة التقاطها:

- تقوم العقدة A بحذف `epc_profile` X. تمر `ON DELETE RESTRICT` لأن A ليس لديها مشترك محلي يشير إلى X في تلك اللحظة.
- تشير العقدة B (أو خطوة التزويد) إلى المشترك 203 إلى X. تمر هذه لأن X لا يزال موجودًا على B.
- **و** تتقارب كل عقدة على "X → 203". تتكرر كلا التغييرات وتطبق دون فحوصات FK. مرجع عالق لم تنتهك FK الخاصة بأي عقدة محليًا: "تم حذف X"

غير قادر على بناء بيانات الاشتراك HSS العالق يجعل `subscriber.epc_profile_id` مرجع ULR/AIR، على `DIAMETER_ERROR_USER_UNKNOWN (5001)` بـ HSS لذلك المشترك. ثم يجيب

على الرغم من أن صف المشترك موجود ومفعل. تجعل نافذة المخطط قبل البيانات (أدناه) الأمر أكثر احتمالًا، لأن المعاملات المتخطاة يمكن أن تفصل إنشاء/حذف الوالد عن إعادة توجيه الطفل.

الكشف (ليس التنفيذ)

على مسار التطبيق إلى إيقاف الشبكة عند أي شذوذ، لذا بدلاً من ذلك نحن FKs سوف يؤدي فرض **نكتشف وننبه**. تقوم كل عقدة بتثبيت وظيفة `SECURITY DEFINER`، `omnihss_fk_orphan_count()`، ذات عمود واحد تشير قيمتها إلى والد مفقود FK التي تعد كل عندما تكون أكبر من Grafana باختيارها، وتنبه `postgres_exporter` عبر جميع الجداول. يقوم صفر (انظر المقاييس).

قم بتشغيل التنظيف يدويًا في أي وقت:

```
sudo -u postgres psql -d omnihss -tAc "SELECT
omnihss_fk_orphan_count();"   # 0 = نظيف
```

غير EPC مثال: المشتركين الذين يشيرون إلى ملف) لتحديد الصفوف المخالفة لمرجع معين موجود):

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

قم بإصلاح ذلك ن طريق إعادة توجيه الصف إلى والد صالح (تحديث على صف موجود يتكرر بشكل نظيف عبر الشبكة) أو إزالته.

منع انحراف المخطط

كان المحفز الجذري للحادث الذي حفز هذا القسم هو **ترحيل المخطط الذي تم تطبيقه على عقدة واحدة بينما كانت نظيراتها لا تزال على المخطط القديم**، ثم كتابة بيانات جديدة ضد الحقول الجديدة. نشر المصدر صفوف لم تتمكن نظيرات المخطط القديم من تطبيقها، مما تسبب في أخطاء تطبيق و**معاملات متخطاة**، وبالتالي تباين.

- OmniHSS **قم بترحيل الشبكة بأكملها معًا.** قم بتطبيق الترحيلات على كل عقدة قبل التزويد ضد الحقول الجديدة. لا تكتب بيانات المخطط الجديد بينما أي نظير متخلف.

- تقوم العقدة بتصدير `omnihss_schema_max_version` / `omnihss_schema_migration_count` لكل عقدة. قارنها عبر الشبكة. عقدة متخلفة عن نظيراتها هي ترحيل نصف مطبق. **راقب الانحراف.**
- يعني ارتفاع `omnihss_subscription_error_apply_error_count` / `_sync_error_count` أن عامل التطبيق يفشل (وربما يتخطى). يتم تنبيه "أخطاء تطبيق/تزامن نسخ HSS" على هذا. **راقب عدادات الأخطاء.**

إزالة نظير

تؤدي إزالة نظير إلى إسقاط الاشتراك **من** ذلك النظير وإزالة إدخاله في `pg_hba.conf`. تبقى البيانات التي تم نسخها بالفعل؛ فقط التغييرات المستقبلية تتوقف عن التدفق. يجب تأكيد الإزالة بشكل صريح.

على العقدة التي تقوم بإزالة النظير *منها*:

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

ثم قم بإزالة سطر (أسطر) النظير من `pg_hba.conf` وأعد تحميل PostgreSQL:

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

لا يزال النظير الذي تتم إزالته يحمل اشتراكه إلى هذه العقدة. إذا كان يجب إزالة الرابط بالكامل، قم بإزالة ذلك الاشتراك هناك أيضًا. ثم قم بإزالة العقدة من مجموعة `connected_omnihss` `hss` حتى لا تعيد تشغيل الوظيفة التالية إنشائها.

النس�� الاحت��اطي والاستعادة

تستخدم النسخ الاحتياطي والاستعادة في OmniHSS تفريغ PostgreSQL القياسي:

العملية	دفتر التشغيل	الملاحظات
النسخ الاحتياطي	`services/backup.yml`	ينتج `hss_dump_<host>_<ts>.sql` (قاعدة البيانات) و `hss_<host>_<ts>.tar.gz` (تكوين `/etc/omnihss`) تحت `backups/<Site_Name>/`. يستثني التفريغ المنشورات/الاشتراكات.
الاستعادة	`util_playbooks/restore_hss.yml`	يطلب تفريغ SQL و/أو أرشيف التكوين. يوقف OmniHSS، يأخذ نسخة احتياطية قبل الاستعادة، **يمزق العقدة من الشبكة** (يسقط اشتراكاتها/منشوراتها/فتحاتها ويعطل اشتراكات النظيرات لها)، يسقط/يعيد إنشاء قاعدة البيانات، يستعيد، يرحل، ويعيد التشغيل. يرفض استهداف أكثر من عقدة واحدة في وقت واحد.

الاستعادة والشبكة

بينما `DROP DATABASE` قبل المسح، تزيل الاستعادة العقدة من الشبكة النشطة-النشطة. تفشل (`omnihss.replication.enabled`، توجد اشتراك أو فتحة منطقية. عندما يتم تمكين النسخ تعيد الاستعادة بعد ذلك بناء الشبكة بهذا الترتيب:

1. **نشر.** يقوم دفتر التشغيل بإنشاء المنشور على العقدة المستعادة. لا يشترك العقدة المستعادة في نظيراتها بعد.
2. **إعادة الاتصال وتهيئة النظيرات المحلية.** يقوم دفتر التشغيل بإعادة إنشاء `hss` الاشتراك من كل نظير محلي إلى العقدة المستعادة، مع `copy_data = false`. ثم يملأ كل نظير فارغ بـ `pg_dump --data-only --inserts --on-conflict-do-nothing`. يتخطى `schema_migrations` و `pdn_session` و `lte_call`، كما

يحتفظ النظير الذي يحمل البيانات ببياناته. `seed_hss_replication.yml` يفعل ويستأنف البث فقط.

3. للوظيفة `replication.yml`. **الاشتراك مرة أخرى.** يقوم دفتر التشغيل بتشغيل ينشئ هذا الاشتراكات من العقدة المستعادة إلى نظيراتها.

قم بتنفيذ الخطوة 3 فقط بعد الخطوة 2. إذا اشتركت العقدة المستعادة في نظير قبل التهيئة، فإن ودورة تحطم عامل `duplicate key` صفوف التهيئة تتدفق مرة أخرى إليها. هذا يسبب أخطاء التطبيق.

لذا فإنه، `TRUNCATE` على نظير. يقوم النسخ المنطقي بتكرار `COPY` أو `TRUNCATE` لا تستخدم يمسح العقدة المستعادة وبقية الشبكة.

يجب أن تكون جميع النظيرات المحلية قابلة للوصول. إذا لم يكن النظير قابلاً للوصول، تفشل الاستعادة.

قم بتهيئتها باستخدام `connected_omnihss`). لا تقوم الاستعادة بتهيئة النظيرات عبر المواقع بعد الاستعادة، قم بتشغيل `seed_hss_replication.yml`. `util_playbooks/audit_hss_drift.yml` للتأكد من أن الشبكة قد تقاربت.

انظر دفاتر التشغيل المساعدة لعملية النسخ الاحتياطي/الاستعادة الأوسع.

استعادة عقدة في شبكة نشطة-نشطة

بإيقاف النسخ على العقدة التي تستعيدها: قبل أن تسقط وتعيد `restore_hss.yml` تقوم إنشاء قاعدة البيانات، تسقط اشتراكات تلك العقدة، ومنشوراتها، وفتحاتها. لذا بعد الاستعادة، لا تحتوي العقدة على **أشياء نسخ** — يجب إعادة توصيل الشبكة، وإعادة توصيل *النظير* وحده ليس كافيًا (لا تزال العقدة المستعادة بحاجة إلى منشورها الخاص مرة أخرى). قاعدتان:

1. **استعادة عقدة واحدة بالضبط.** الاستعادة على عقدتين أو أكثر في وقت واحد تدمر الشبكة (`restore_hss.yml` يحذر من ذلك). استعد واحدة، ثم هيئ البقية منها.
2. **قبل توصيل النسخ. OmniHSS يجب أن تعمل جميع العقد بنفس إصدار** من `CREATE SUBSCRIPTION` يحمل النسخ المنطقي البيانات، وليس المخطط. تتحقق أن **كل جدول في منشور الناشر موجود بالفعل على المشترك**، لذا لا يمكن الذي يتضمن منشوره جدولًا أضافه ترحيل لاحق،) أن تشترك عقدة على إصدار أحدث في نظير لا يزال على المخطط القديم — تفشل الاشتراك (`ims_subscription` مثل ولا تظهر تلك الاتجاه. يجب `relation "public.<table>" does not exist` مع أن يتطابق مخطط **تفريغ** الاستعادة أيضًا مع الإصدار المنشر، أو تنتهي العقدة المستعادة متقدمة على نظيراتها.

إعادة بناء زوج نشط-نشط من الصفر:

```
# 1. قم بتزويد كلا الجهازين الافتراضيين، ثم قم بتثبيت OmniHSS +
المخطط على كليهما بنفس
#    الإصدار (مخطط متطابق، فارغ، تم ترحيله على كل منهما).
ansible-playbook -i hosts/<inv>/host_files/<site>.yml omnihss.yml

# 2. استعادة البيانات على عقدة واحدة فقط (يجب أن يتطابق مخطط
التفريغ مع إصدار الخطوة 1).
#    هذا يمسح أشياء النسخ الخاصة بتلك العقدة؛ تعيد الخطوة 3 بناء
الشبكة.
ansible-playbook -i hosts/<inv>/host_files/<site>.yml \
  util_playbooks/restore_hss.yml --limit <site>-hss01

# 3. هيئ العقدة الأخرى (العقد الأخرى) وقم بتوصيل الشبكة الثنائية
الاتجاه بالكامل – لا تستخدم --limit.
#    يتم ملء العقد الفارغة من العقدة المستعادة؛ يتم (إعادة) إنشاء
المنشورات،
#    الاشتراكات والفتحات على كل عقدة، ثم يتم التحقق منها.
ansible-playbook -i hosts/<inv>/host_files/<site>.yml \
  util_playbooks/seed_hss_replication.yml
```

يحدد أيضًا `--limit`: لا تنتهي باستخدام `seed_hss_replication.yml --limit <peer>` التوصيل، لذا فإنه ينشئ فقط اشتراك النظير ويترك العقدة المستعادة بدون منشورها — شبكة الآن أن (التي تعمل تلقائيًا) `validate_replication.yml` نصف موصلة، أحادية الاتجاه. تؤكد كل عقدة تشترك في كل نظير، لذا فإن نتيجة نصف موصلة تفشل بصوت عالٍ بدلاً من الإبلاغ عن صحة.

المقاييس

تشغل كل عقدة OmniHSS `postgres_exporter`، مما يكشف عن مقاييس PostgreSQL (بما في ذلك النسخ) إلى Prometheus. السلاسل المفيدة لصحة النسخ:

المقياس: `pg_replication_slots_active` **النوع**: Gauge **الوصف**: ما إذا كانت كل فتحة نسخ منطقية لديها مستهلك نشط. **التسميات**:

- `slot_name`: (واحدة لكل نظير مشترك) فتحة النسخ

الوصف: تأخير موضع Gauge: تأخير الفتحة / **النوع**: `pg_stat_replication_*` **المقياس**
WAL لكل فتحة. يظهر مدى تأخر النظير المستهلك.

استعلامات المثال:

```
# (0 < تنبيه إذا كانت) أي فتحة نسخ منطقية غير نشطة
count(pg_replication_slots_active == 0)

# يمكن التحقق من الاشتراكات التي يجب أن توجد مقابل العمال الذين
يعملون
# ضد مخرجات دفتر التشغيل للتحقق أثناء النشر.
```

مقاييس OmniHSS المخصصة

يصدر `postgres_exporter` على كل عقدة أيضًا سلاسل صحة النسخ وسلامة البيانات الخاصة بـ OmniHSS عبر ملف استعلام مخصص (`roles/omnihss/files/postgres_exporter_queries.yaml`). تقوم Prometheus بجمع هذه تحت وظيفة `HSS Postgres`. **تنبيهات** تدعم قواعد التنبيه في مجموعة **HSS** (`roles/monitoring/templates/grafana/rules/`):

المقياس	النوع	ف
`omnihss_subscription_worker_alive{subname}`	Gauge	ن 1 امل بيق عمل
`omnihss_replication_slot_active{slot_name}`	Gauge	ن 1 ظير هلك تحة
`omnihss_replication_slot_wal_status_code{slot_name}`	Gauge	ر=0 سعة =غير جوز **ودة**
`omnihss_replication_slot_lag_bytes{slot_name}`	Gauge	يتات ل بها تحة
`omnihss_subscription_error_apply_error_count{subname}`	Counter	طاء كمية امل بيق
`omnihss_subscription_error_sync_error_count{subname}`	Counter	طاء كمية

المقياس	النوع	ف
		امنة اول.
`omnihss_integrity_fk_orphans`	Gauge	وف فتاح رجي جب ن 0)
`omnihss_schema_max_version` / `omnihss_schema_migration_count`	Gauge	مياه الية حيل بق / لعدد

انظر المراقبة والرصد لخط أنابيب المقاييس واللوحات.

استكشاف الأخطاء وإصلاحها

تبقى العقدة الجديدة فارغة بعد الانضمام

الأعراض: عمال النسخ مفعّلون والفتحات نشطة، لكن العقدة الجديدة ليس لديها (أو لديها القليل من) المشتركين.

الأسباب المحتملة:

- انضمت العقدة إلى شبكة مكتملة بالفعل ولم يتم تهيئتها أبدًا. تستخدم الاشتراكات `copy_data = false`، لذا تصل فقط التغييرات بعد الانضمام.

الحل:

1. هيئ العقدة من نظير صحي. انظر تحميل عقدة جديدة بالبيانات.

عامل الاشتراك غير قيد التشغيل

الأعراض: `pid IS NULL` لاشتراك في `pg_stat_subscription`.

الأسباب المحتملة:

- النظير غير قابل للوصول على منفذ PostgreSQL (جدار ناري / توجيه).
- عنوان IP للنظير مفقود من `pg_hba.conf` لهذه العقدة.
- بيانات اعتماد خاطئة في سلسلة الاتصال (`database_username` / `database_password` تختلف بين العقد).

الحل:

1. تأكد من إمكانية الوصول عبر TCP `5432` إلى النظير على.
2. تحقق من ظهور النظير في `connected_omnihss` (أو مجموعة `hss` المحلية) وأعد تشغيل الوظيفة حتى يتم تحديث `pg_hba.conf`.
3. تأكد من أن `database_username`/`database_password` متطابقة عبر جميع العقد.

تظهر فتحة النسخ `wal_status = lost`

الأعراض: حالة `wal_status` لفتحة هي `lost`؛ النظير المقابل متأخر جدًا.

الأسباب المحتملة:

- كانت النظير متوقفة أو غير قابلة للوصول لفترة طويلة بحيث تمت إزالة WAL المطلوبة.

الحل:

1. أعد تهيئة النظير المتأثر من عقدة صحية. انظر تحميل عقدة جديدة بالبيانات.

أخطاء مفتاح مكرر أثناء النسخ الأولي

الأعراض: أخطاء في تطبيق التغييرات مباشرة بعد إنشاء اشتراك مع `copy_data = true`.

الأسباب المحتملة:

- تم تشغيل النسخ الأولي من أكثر من نظير، مما أدى إلى نسخ نفس الصفوف مرتين.

الحل:

1. احتفظ بجميع الاشتراكات؛ `copy_data = true` هيئ من **نظير واحد فقط** مع `copy_data = false` الأخرى عند.

تحديثات حالة متضاربة بين المواقع

الأعراض: يبدو أن صف الحالة (مثل `subscriber_state`) "يتأرجح" بين القيم المحددة في مواقع مختلفة.

الأسباب المحتملة:

- تحديثات متزامنة لنفس الصف في موقعين.

الحل: هذا متوقع ويتم التعامل معه. يحافظ مشغل آخر كتابة على التحديث مع أحدث `updated_at`. تأكد من مزامنة الساعات (NTP) عبر جميع المواقع بحيث يكون ترتيب `updated_at` ذا معنى.

نشر Proxmox VM/LXC

يقوم كتاب اللعب Proxmox. على OmniCore تقوم الغالبية العظمى من عملائنا بتشغيل مجموعة بناءً على LXC وحاويات VMs QEMU بتوفير كل من ، `util_playbooks/proxmox.yml`، الواحد (`proxmoxServers` على إدخال) يمكنك تعيين التبديل لكل موقع. `deployment_type` تبديل بينما تبقى المضيفات الفردية LXC وتجاوزه لكل مضيف. يتيح ذلك لموقع ما أن يعمل في الغالب كـ كاملة VMs كـ (مثل UPF).

نحن لا نزال ندعم VMware وHyperV والسحابة (Vultr / AWS / GCP) للنشر.

انظر أيضًا:

- تعريف: تكوين ملف المضيفين VMs/LXCs للنشر
- معيار تخطيط IP
- تكوين Netplan
- معمارية النشر

مقابل VM LXC

حاويات LXC:

- خفيفة الوزن، تشارك نواة المضيف
- بدء تشغيل سريع، تكلفة منخفضة
- عزل محدود
- لا يمكن تشغيل نوى مخصصة أو وحدات نواة
- **غير مناسبة لنشر EPC/IMS الإنتاجي**
- **لا يمكن تشغيل UPF** (يتطلب وحدات نواة / أجهزة TUN)

VMs (KVM):

- افتراضية كاملة مع نواة مخصصة
- عزل كامل، تشغيل وحدات نواة / شبكات مخصصة
- تكلفة موارد أعلى
- **موصى بها للإنتاج**
- **مطلوبة لنشر UPF**

حالات الاستخدام:

- **VMs**: مواقع الإنتاج، UPF، جميع وظائف الشبكة
- **LXC**: بيئات مختبر / اختبار، خدمات خفيفة الوزن (dns، مراقبة، apt-cache)

اختيار نوع النشر

يتم حل `deployment_type` لكل مضيف بهذا الترتيب:

1. **لكل مضيف** `deployment_type` (المعين تحت اسم المضيف): يفوز إذا كان موجودًا.
2. **لكل موقع** `deployment_type` على إدخال `proxmoxServers` الذي يختاره التوزيع الدائري للمضيف.
3. **افتراضي** `vm` إذا لم يتم تعيين أي منهما.

النمط الشائع هو تعيين الافتراضي للموقع على إدخال `proxmoxServers` (مثل `deployment_type: lxc`) وتجاوز فقط المضيفات التي يجب أن تختلف. المثال القياسي هو مختبر LXC باستثناء UPF، الذي يحتاج إلى وحدات نواة / TUN ويجب أن يكون يعمل كل شيء كـ VM:

```
all:
  vars:
    proxmoxServers:
      pve-node-01:
        deployment_type: lxc      # LXC الافتراضي للموقع ← كل شيء هو
        # ...

upf:
  hosts:
    site-upf01:
      ansible_host: 192.168.1.24
      deployment_type: vm         # VM تجاوز لكل مضيف ← هذا المضيف هو
```

كقاعدة، حافظ على الافتراضي للموقع متسقًا وتجاوز فقط الاستثناءات. عندما تقوم بتوزيع VM عبر العديد من المضيفات، يعتمد وضع العقد على فهرس المضيف ومن الصعب التفكير فيه وLXC.

يؤدي كتاب اللعب فقط إلى الفشل الصارم في حالة حدوث تصادمات نوع مع الموارد الموجودة: إذا كانت هناك مورد مع اسم المضيف المستهدف موجود بالفعل كنوع خاطئ (مثل LXC ولكن وجود)

فإنه يتوقف مع خطأ واضح. انظر، `deployment_type: vm` (أو العكس) المضيف يحل إلى حراس النوع.

إعداد Proxmox

1. إنشاء رمز API

```
# واجهة Proxmox: رموز ← الأذونات ← مركز البيانات API
# إنشاء رمز: root@pam!<TokenName>
# انسخ سر الرمز (يظهر مرة واحدة)
# قم بإلغاء تحديد مربع `فصل الامتيازات`
```

2a. إنشاء قالب VM Cloud-Init (فقط للمواقع VM)

يقوم بتنزيل صورة سحابية لأوبونتو ويخلق Proxmox. قم بتشغيل هذا البرنامج النصي على مضيف قالبًا نظيفًا. يقوم `proxmox.yml` بحقن بيانات اعتماد cloud-init ومفاتيح SSH في وقت النسخ (انظر الملاحظات أدناه).

```
#!/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
```

ملاحظات:

- بتعيين `proxmox.yml` القالب لا يحتوي على مستخدمين أو كلمات مرور مشفرة. يقوم `sshkeys` و`cipassword` و`ciuser` في وقت النسخ.
- أولوية مستخدم cloud-init: `proxmoxTemplateUser` (على إدخال `proxmoxServers` المطابق) → أول مفتاح `local_users`.
- أولوية كلمة مرور cloud-init: `proxmoxTemplatePassword` (على إدخال `proxmoxServers` المطابق) → مستخدم cloud-init (أي القيمة المختارة أعلاه).
- يتم حقن مفاتيح SSH من كل إدخال `local_users.*.public_key` (المصادقة القائمة على المفاتيح هي مسار تسجيل الدخول المقصود).
- قم بتعيين `proxmoxTemplateUser` (ويمكن اختيارياً `proxmoxTemplatePassword`) على كل إدخال `proxmoxServers` كلما تم إعداد القالب مع مستخدم cloud-init ليس هو أول إدخال `local_users`، بحيث يتم اختيار الحساب الصحيح.
- `--vga std` يضمن أن وحدة التحكم في الويب Proxmox تعمل.

- علامة `-N` على wget تقوم بتنزيل فقط إذا كانت أحدث من النسخة المحلية.

بديل: قالب يدوي من ISO

إذا لم تكن الصور السحابية متاحة أو كنت بحاجة إلى تثبيت مخصص:

الخطوة 1: إنشاء VM عبر واجهة الويب

- إنشاء VM → جديدة VM 9000، الاسم: ubuntu-2404-template
- نظام التشغيل: تحميل ISO → خادم أوبونتو أو استخدام ISO موجود
- النظام: الافتراضي (وحدة التحكم SCSI: VirtIO SCSI)
- الأقراص: 32GB، الناقل: SCSI
- وحدة المعالجة المركزية: 2 نوى
- الذاكرة: 2048 ميجابايت
- الشبكة: VirtIO، جسر vmbr0
- بدء VM → خادم أوبونتو وتثبيت

الخطوة 2: داخل VM - تنظيف وتحضير

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

# تنظيف البيانات الخاصة بالآلة
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

# مسح تاريخ bash وإيقاف التشغيل
history -c
sudo poweroff
```

الخطوة 3: إضافة Cloud-Init وتحويل إلى قالب

- اخ��ر VM → محرك → إضافة → الأجهزة CloudInit (اختر التخزين مثل local-lvm)
- وكيل → الخيارات → الأجهزة QEMU → تمكين
- تحويل إلى قالب → انقر بزر الماوس الأيمن على VM

- لا تقم **بتعيين** مستخدم / كلمة مرور cloud-init. يقوم `proxmox.yml` على القالب بحقن هذه في وقت النسخ من `proxmoxTemplateUser` / `proxmoxTemplatePassword` (مع تراجع `local_users` لاسم المستخدم).

2b. تنزيل قالب نظام تشغيل LXC (فقط LXC للمواقع)

```
# على قشرة عقدة Proxmox:
pveam update
pveam download local ubuntu-24.04-standard_24.04-2_amd64.tar.zst
```

مسار الحجم الناتج (مثل `local:vztmpl/ubuntu-24.04-standard_24.04-2_amd64.tar.zst`) هو ما تقوم بتكوينه كـ `proxmoxLxcOsTemplate` أدناه.

تكوين ملف المضيفين

نشر VM

```yaml
all:
  vars:
    # تم حذف deployment_type → هو الافتراضي "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        # اسم مستخدم cloud-init اختياري؛ الافتراضي هو أول مفتاح local_users
        proxmoxTemplatePassword: omnitouch    # كلمة مرور cloud-init اختياري؛ الافتراضي هو اسم مستخدم cloud-init
        storage: local-lvm  # اختياري
      pve-node-02:
        # ... تكوين العقدة الثانية

    local_users:
      admin_user:
        name: Admin User
        public_key: "ssh-rsa AAAA..."

mme:
  hosts:
    site-mme01:
      ansible_host: 192.168.1.10
  vars:
    proxmox_interface: vmbr0
    gateway: 192.168.1.1
    mask_cidr: 24  # 24 اختياري، الافتراضي
    vlanid: 100  # اختياري
```

نشر LXC

```yaml
all:
  vars:
    proxmox_lxc_nameserver: "1.1.1.1"   # 1.1.1.1 اختياري، الافتراضي
    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  # تراجع اختياري، rootfs
      pve-node-02:
        deployment_type: lxc
        # ... نفس الشكل

    local_users:
      admin_user:
        name: Admin User
        public_key: "ssh-rsa AAAA..."

dns:
  hosts:
    site-dns01:
      ansible_host: 192.168.1.20
  vars:
    proxmox_interface: vmbr0
    gateway: 192.168.1.1
    mask_cidr: 24                        # 24 اختياري، الافتراضي
    vlanid: 100                         # اختياري
    proxmoxLxcCores: 2                  # تجاوز اختياري
    proxmoxLxcMemoryMb: 4096            # تجاوز اختياري
    proxmoxLxcDiskSizeGb: 30            # تجاوز اختياري
    proxmoxLxcRootFsStorageName: local-lvm  # اختياري، يتجاوز التخزين
الافتراضي للمضيف proxmoxLxcDefaultStorage
```

الاستخدام

توفير (VM أو LXC)

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

حذف

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

يتعامل كتاب اللعب للحذف مع كل من VMs وLXCs تلقائيًا، بغض النظر عن `deployment_type`.

سلوك كتاب اللعب

حراس النوع

- إذا كان هناك VM ولكن موجود مع اسم المضيف المستهدف `deployment_type: lxc` → يفشل كتاب اللعب مع خطأ واضح.
- الحالة العكسية (LXC موجود، الموقع مُعد كـ VM) → نفس الشيء.
- مجموعة UPF على موقع LXC → تحذير فقط (لن يعمل UPF ليس فشلًا صارمًا، لكن).

مورد جديد

- حسب فهرس المضيف داخل `proxmoxServers` يقوم بالتوزيع الدائري عبر إدخالات المجموعة، ما لم يقم المضيف بتعيين `proxmox_server:` (انظر لتثبيت إدخال معين التوزيع عبر العقد).
- يستعلم عن واجهة برمجة تطبيقات Proxmox للحصول على VMID التالي المتاح.
- **VM**: يتم استنساخه من `proxmoxTemplateId`، ويقوم بتعيين مستخدم cloud-init / من `proxmoxTemplateUser` (مستخدم cloud-init = كلمة المرور / مفاتيح SSH ؛ كلمة مرور cloud-init = `local_users` إدخال الخادم إذا تم تعيينه، وإلا أول مفتاح

`proxmoxTemplatePassword` إذا تم تعيينها، وإلا اسم مستخدم cloud-init)، يقوم بتغيير حجم `scsi0`، ويبدأ.

- **LXC**: يتم إنشاؤه من `proxmoxLxcOsTemplate`، rootfs على `proxmoxLxcRootFsStorageName` (المجموعة) ← `proxmoxLxcDefaultStorage` (الخادم) ← `storage` ← `local-lvm` (التراجع النهائي)، يقوم بحقن جميع `local_users.*.public_key` عبر `ssh-public-keys`، `0` ويبدأ. كلمة مرور الجذر مشفرة كـ `mn1Touch@`. جميع LXCs غير مميزة (`unprivileged: 1`) مع تمكين التعشيش (`features: nesting=1`).

مورد موجود

تقوم كل من VM وLXC بتحديث المسارات بمقارنة الحالية مقابل المرغوبة وتطبيق التغييرات:

- تكوين الشبكة (الجسر، علامة VLAN، IP لـ LXC)
- النوى / الذاكرة
- حجم القرص (VM `scsi0` / LXC `rootfs`)
- NICs الثانوية من `secondary_ips` أو يتم إعادة تشغيل المورد إذا تم تغيير الشبكة أو الموارد.

> **OmniUPF:** تعالج الأدوات إدخال `secondary_ips` المسمى `n3` (أو `data` / `gtp`) كـ NIC مخصص لطائرة المستخدم. يقوم دور `omniupf` بربط N3/N9 + XDP به ويخصص وحدة معالجة مركزية للواجهة الرئيسية، بحيث لا يمكن أن يؤدي الحمل الثقيل على مسار البيانات إلى تجويع PFCP/API/SSH (التي تبقى على IP الرئيسي).

وضع العلامات

يتم وضع علامة ع��ى كل مورد بأسماء مجموعته في Ansible + `site_name`.

التوزيع عبر العقد

بشكل افتراضي، تقوم الأدوات بتوزيع المضيفات عبر إدخالات `proxmoxServers` بالتوزيع الدائري، مع نفس المنطق لـ VMs وLXCs. **داخل مجموعة دوره**، الفهرس هو موضع المضيف في `proxmoxServers` (بترتيب الإعلان):

```
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)
```

> **إضافة عقدة ثانية تعيد توزيع كل شيء.** نظرًا لأن التوزيع يعتمد على الموضع، فإن الانتقال من إدخال `proxmoxServers` واحد إلى اثنين يغير مكان وجود مجموعات متعددة المضيفين الموجودة (المضيف-0 → الإدخال-0، المضيف-1 → الإدخال-1، ...). قم بتثبيت المضيفات التي لا تريد نقلها (أدناه) قبل إعادة التشغيل ضد أسطول مختلط.

تثبيت مضيف على عقدة معينة

لفرض `proxmoxServers` إلى **اسم إدخال** `proxmox_server:` قم بتعيين المفتاح الاختياري مضيف على تلك العقدة بغض النظر عن فهرس مجموعته. عند عدم تعيينه، ينطبق التوزيع الدائري تفشل التشغيل مع خطأ يسرد الأسماء الصالحة (لذا `proxmoxServers` أعلاه. قيمة ليست مفتاح لا يمكن أن يؤدي خطأ مطبعي إلى التوزيع الدائري في مكان آخر).

```yaml
device_tester:
  hosts:
    site-device-tester01:
      ansible_host: 10.179.2.50
      proxmox_server: nicklab-prxmx01   # تثبيت على هذا الإدخال من proxmoxServers (→ عقدته)
      # إذا كان تخزين القرص لتلك العقدة يختلف عن اسم التخزين العالمي proxmoxLxcRootFsStorageName،
      # (يتم قراءة التخزين عالميًا، وليس من إدخال الخادم) قم بتجاوزه هنا أيضًا:
      proxmoxLxcRootFsStorageName: local-lvm
```

للمجموعة لتثبيت كل مضيف في `vars:` على مستوى `proxmox_server` يمكن أيضًا تعيين مجموعة على عقدة واحدة.

تجاوزات لكل مضيف

(تثبيت العقدة: انظر التوزيع عبر العقد) يمكن تعيين `proxmox_server` و`deployment_type` ع��ى مستوى المجموعة LXC وأي من مفاتيح (`proxmoxLxcCores`، `proxmoxLxcMemoryMb`، `proxmoxLxcDiskSizeGb`، `proxmoxLxcRootFsStorageName`، `proxmoxLxcOsTemplate`، `host_vm_network`) لكل مضيف تحت اسم المضيف.

```yaml
dns:
  hosts:
    site-dns01:
      ansible_host: 192.168.1.20
      deployment_type: vm          # تجاوز لكل مضيف للافتراضي للموقع
      proxmoxLxcDiskSizeGb: 120    # تجاوز لكل مضيف
      host_vm_network: vmbr1       # تجاوز جسر لكل مضيف
  vars:
    proxmox_interface: vmbr0
    gateway: 192.168.1.1
    mask_cidr: 24  # 24 اختياري، الافتراضي
```

سجلات تشغيل الكتاب

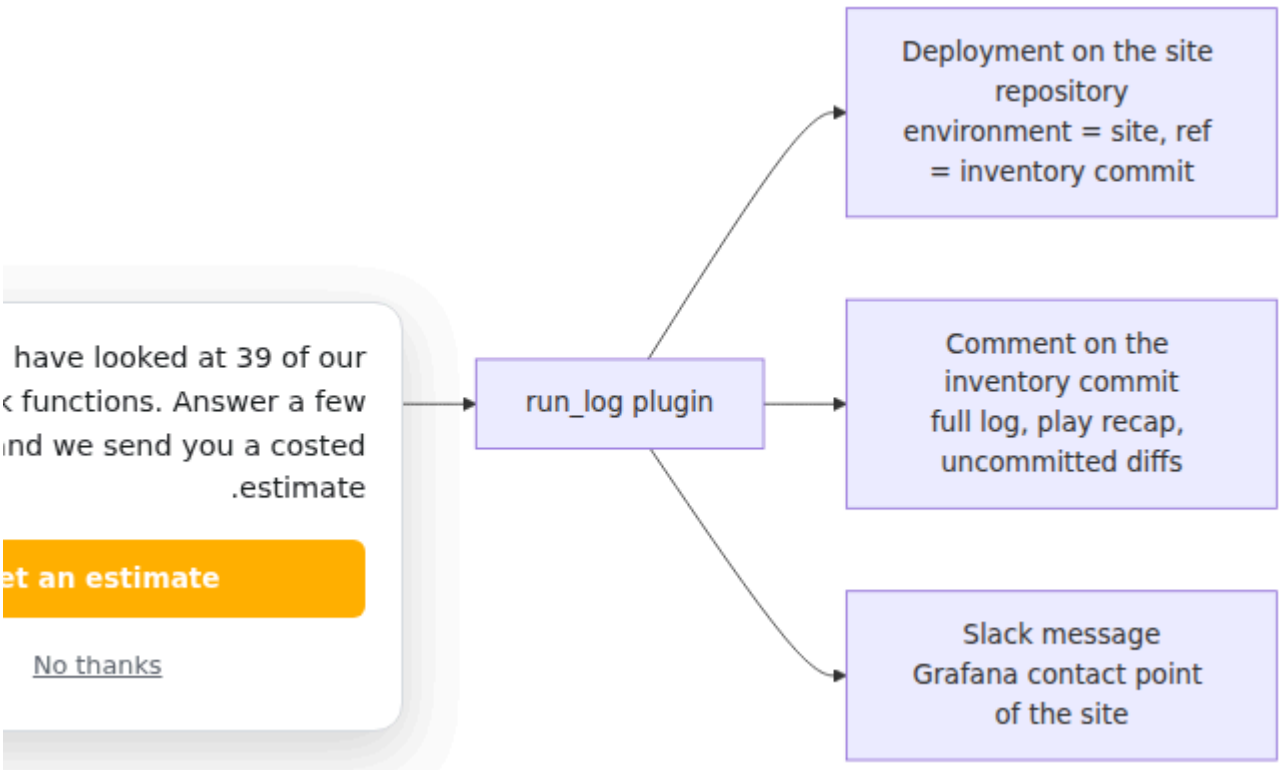
تقوم الإضافة `run_log` بتسجيل كل تشغيل لـ `ansible-playbook` على GitHub. يتم تفعيلها في `ansible.cfg` في استنساخ git عبر مجموعة الأدوات المثبتة وفي apt. يقوم المشغل بتشغيل الكتب كما كان من قبل:

```
ansible-playbook -i hosts/YourCustomer/host_files/production.yml
services/firewall.yml
```

في نهاية التشغيل، تقوم الإضافة بطباعة دليل التشغيل المحلي، والنشر ورابط السجل.

ما الذي يتم تسجيله

يتم تسجيل السجل في مستودع الموقع: `origin` لاستنساخ git الذي يحتوي على ملف الجرد المقدم مع `-i`. اسم الموقع هو اسم ملف الجرد بدون امتداده.



السجل	المحتوى
النشر	`success` أو `failure`. الحالة = التزام الجرد. المرجع = الموقع. البيئة = الحمولة = الأمر، المشغل، الآلة، وقت البدء، المدة، النتيجة، المضيفون الفاشلون، إصدار Ansible، وإصدار كلا الاستنساخين: الفرع والالتزام، أو إصدار حزمة `omnicore` من تثبيت apt، مع عدد الملفات غير الملتزم بها.
تعليق الالتزام	السجل الكامل للتشغيل، ملخص اللعب، والتغييرات غير الملتزم بها لكلا الاستنساخين. حالة النشر ترتبط به. يتم تقسيم السجل الطويل إلى عدة تعليقات.
رسالة Slack	النتيجة، المشغل، الأمر، كلا الالتزامين، وروابط إلى النشر، والالتزام والسجل. يتم نشرها إلى أول مستقبل في `monitoring_data.contactPoints`، `slack` نقطة الاتصال التي تنبه من خلالها Grafana.
النسخة المحلية	`~/.local/state/omnicore/runs/<id>/`: `ansible.log`, `run.json`, `playbooks.diff`, `inventory.diff`.

المتطلبات

- يجب أن يكون ملف الجرد في استنساخ git الذي يكون `origin` الخاص به على GitHub، وأن يكون التزام الجرد قد تم دفعه.
- يجب تثبيت `gh` (واجهة سطر الأوامر لـ GitHub) وتسجيل الدخول مع الوصول إلى مستودع الموقع.
- يجب تعيين `log_path` في `ansible.cfg`.

إذا لم يتم تلبية أي متطلب، تقوم الإضافة بطباعة خطأ `run_log:` في نهاية التشغيل. لا تتغير نتيجة الكتاب.

أسرار العملاء

كل مخزون عميل يحتوي على ملف مشفر للقيم التي يجب عدم تخزينها بشكل واضح: مفاتيح API، كلمات المرور، والتوكنات. الملف هو `hosts/<customer>/group_vars/secrets.yml`. يتم تشفيره باستخدام Ansible Vault. الأدوار التي تحتاج إلى سر تقوم بتحميل هذا الملف في وقت التشغيل. النص العادي لا يدخل Git أبداً.

كيف يعمل



1. يقوم المشغل بإضافة سر إلى الملف باستخدام `scripts/vault_secrets.py`. يقوم الأداة بتشفير الملف باستخدام `ansible-vault`.
2. يقوم الدور بفك تشفير `common/secrets` الدور الذي يحتاج إلى سر يتضمن الدور الملف مرة واحدة لكل مضيف ويحمّل كل اسم على مستوى أعلى كمتغير عادي.
3. يؤكد الدور أن الأسماء التي يحتاجها موجودة. إذا كان اسم مفقود، تفشل العملية مع الأمر الدقيق الذي ينشئه.

يتم فك تشفير الملف فقط على وحدة التحكم Ansible. فقط التكوين المعروض يصل إلى مضيف الهدف.

تنسيق الملف

الملف هو خريطة YAML من الأسماء إلى القيم. القيمة هي سلسلة أو خريطة من السلاسل.

```
omnihss_api_keys:
  omniweb: "<64 hex characters>"
  omnihlr: "<64 hex characters>"
  omnicrm: "<64 hex characters>"
  omniudr: "<64 hex characters>"
  omnieir: "<64 hex characters>"
  ran_simulator: "<64 hex characters>"
```

الاسم	النوع	مطلوب بواسطة	الوصف
`omnihss_api_keys`	خريطة	`omnihss_api_auth: rbac` (الافتراضي)	مفتاح API واحد لكل عميل في OmniHSS. كل مفتاح هو سر عشوائي لا يقل عن 32 حرفًا. يجب ألا يتشارك عميلان مفتاحًا. انظر مفاتيح OmniHSS API.

تقوم الأدوار الأخرى بتوثيق الأسماء التي تقرأها في أقسامها الخاصة من هذا الدليل عند اعتماد الملف.

إدارة الأسرار

استخدم `scripts/vault_secrets.py` باستخدام `ansible-vault edit`: لكل تغيير. لا تعدل الملف الأداة تحتفظ بالنص العادي في أنبوب ولا تكتبه على القرص.

```
scripts/vault_secrets.py hosts/<customer> list [vault options]
scripts/vault_secrets.py hosts/<customer> set NAME [--random-hex
N] [--map K1,K2,...] [--force] [vault options]
```

الوسيط	الوصف
`hosts/<customer>`	دليل مخزون العميل. يجب أن يحتوي على `host_files/`.
`list`	طباعة أسماء الأسرار. بالنسبة لخريطة، اطبع مفاتيحها أيضًا. لا تطبع قيمة أبدًا.
`set NAME`	إضافة `NAME` إذا لم ي��ن موجودًا إلى الملف. أنشئ الملف.
`--random-hex N`	تعيين القيمة إلى سلسلة عشوائية من `N` حرف `N`بايت، مكتوبة كـ `2` سداسي. يجب أن يكون `N` على الأقل 16.
`--map K1,K2,...`	تعيين القيمة إلى خريطة تحتوي على سلسلة، مع `--random-hex` عشوائية واحدة لكل مفتاح.
`--force`	استبدال `NAME` الذي موجود بالفعل. بدونها، ترفض الأداة.
خيارات vault	تمرر إلى `ansible-vault`، على سبيل المثال `--ask-vault-pass` أو `--vault-password-file FILE`.

تقرأ الأداة سطرًا واحدًا من الإدخال القياسي وتستخدمه، إذا لم يتم إعطاء `--random-hex` كقيمة.

تطلب الأداة كلمة المرور مرة واحدة. عند إنشاء ملف جديد، تطلب، مع `--ask-vault-pass` مرتين للتأكيد.

أمثلة

إنشاء الملف ومفاتيح OmniHSS API لعميل جديد:

```
scripts/vault_secrets.py hosts/ExampleCustomer set
omnihss_api_keys --random-hex 32 \
    --map omniweb,omnihlr,omnicrm,omniudr,omnieir,ran_simulator --
ask-vault-pass
```

إضافة سر من قيمة لديك بالفعل:

```
echo 'the-api-key' | scripts/vault_secrets.py
hosts/ExampleCustomer set example_api_key --ask-vault-pass
```

رؤية الأسرار التي يمتلكها العميل:

```
scripts/vault_secrets.py hosts/ExampleCustomer list --ask-vault-
pass
```

تدوير مفاتيح OmniHSS API:

```
scripts/vault_secrets.py hosts/ExampleCustomer set
omnihss_api_keys --random-hex 32 \
    --map omniweb,omnihlr,omnicrm,omniudr,omnieir,ran_simulator --
force --ask-vault-pass
```

ثم قم بتشغيل الـ playbooks لـ HSS ولكل عميل API، بحيث يحصل الجميع على المفاتيح الجديدة.

تشغيل الـ Playbooks

كل Playbook يصل إلى دور يحتوي على سر يحتاج إلى كلمة مرور vault. قدمها بإحدى طريقتين:

الاستخدام	الأمر	الطريقة
عمليات تفاعلية	`ansible-playbook ... --ask-vault-pass`	المطالبة
عمليات غير مراقبة، CI	`ansible-playbook ... --vault-password-file ~/.ansible/<customer>_vault_pass`	ملف كلمة المرور

في مدير كلمات المرور المشترك. احتفظ OmniCore لكل عميل هي إدخال vault كلمة مرور بملف كلمة المرور خارج المستودع مع وضع `0600`.

إلى مستودع مخزون العميل. من الآمن الالتزام: الملف هو نص مشفر. `secrets.yml` قم بالتزام تحقق من ذلك قبل الدفع:

```
head -c 14 hosts/ExampleCustomer/group_vars/secrets.yml   # prints
$ANSIBLE_VAULT
```

مفاتيح OmniHSS API

تحمي OmniHSS واجهة برمجة التطبيقات REST لكل عميل. لكل عميل الخاصة بها بمفتاح واحد لكل عميل، دور ثابت في OmniHSS. لذا فإن مفتاح مسروق يمنح فقط حقوق ذلك العميل.

المفتاح	العميل	دور OmniHSS
`omniweb`	OmniWeb	`oam`
`omnihlr`	OmniSS7 HLR interworking	`hlr_interworking`
`omnicrm`	OmniCRM	`crm_provisioning`
`omniudr`	OmniUDR (5G core)	`udr`
`omnieir`	OmniEIR (5G core)	`eir_lookup`
`ran_simulator`	RAN Simulator	`ran_simulator`

يقوم دور `omnihss` بعرض جميع المفاتيح الستة في تكوين HSS. يقوم كل دور عميل بعرض مفتاحه الخاص فقط. يجب أن يكون المفتاح لا يقل عن 32 حرفًا، ولا يجوز أن يتشارك عميلان مفتاحًا. تفشل العملية إذا تم كسر أي من القاعدتين.

المعامل	النوع	مطلوب	الافتراضي	الوصف
`omnihss_api_auth`	سلسلة	لا	`rbac`	`rbac`: كل عميل يرسل مفتاحه من `omnihss_api_keys`، و`secrets.yml` مطلوب. `disabled`: تقدم الخدمة لكل OmniHSS متصل بدون مفتاح، ولا حاجة لملف الأسرار. قم بتعيينه في ملف المضيف تحت `all.vars`.

استكشاف الأخطاء وإصلاحها

تفشل العملية مع "يجب أن توجد وتكون مشفرة باستخدام ansible-vault"

الأعراض: يتوقف دور عند "تأكيد وجود ملف الأسرار وأنه مشفر".

الأسباب المحتملة:

- العميل ليس لديه `group_vars/secrets.yml` بعد.
- تم كتابة الملف بنص عادي.

الحل:

1. قم بتشغيل الأمر `set` الموضح في رسالة الخطأ. يقوم بإنشاء الملف مشفرًا.
2. إذا كان هناك ملف نص عادي موجود، قم بحذفه. ثم أعد إنشائه باستخدام الأداة.

تفشل العملية مع "تعذر فك التشفير"

الأعراض: يتوقف دور عند "شرح فشل تحميل الأسرار".

الأسباب المحتملة:

- تم تشغيل الـ playbook بدون `--ask-vault-pass` أو `--vault-password-file`.
- لا تتطابق كلمة المرور مع إدخال OmniCore vault لهذا العميل.

الحل:

1. قم بتشغيل الـ playbook مرة أخرى مع كلمة مرور vault.
2. تحقق من كلمة المرور باستخدام `ansible-vault view hosts/<customer>/group_vars/secrets.yml > /dev/null && echo VAULT-OK`.

يحصل عميل على 401 من OmniHSS API

الأعراض: يسجل OmniWeb أو OmniCRM `401` أو عميل آخر `Unauthorized` من HSS بعد تدوير المفتاح.

الأسباب المحتملة:

- حصل HSS على المفاتيح الجديدة ولكن العميل لم يحصل عليها، أو العكس.

الحل:

1. قم بتشغيل Playbook لـ HSS وPlaybook العميل بنفس `secrets.yml`.
2. تأكد من أن كلا المضيفين قد أعادا تشغيل خدمتهما.

الوثائق ذات الصلة

- تكوين المتغيرات الجماعية `group_vars/` - ملفات أخرى لكل عميل تحت
- مرجع التكوين - معلمات المضيف
- نظرة عامة

كتيبات الخدمة

تقوم كتيبات الخدمة بنشر وتكوين بنية OmniCore التحتية. تقع في دليل `services/`.

تسلسل كتيبات اللعب

تكون كتيبات اللعب منظمة بشكل هرمي لتمكين عمليات النشر السريعة والمستهدفة:

```
all.yml
├── setup_users.yml
├── apt_cache.yml          # فقط عند تعريف مجموعة
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
```

ثوانٍ. يستغرق (`omnimme.yml` مثل) **قم بتشغيل فقط ما تحتاجه.** يستغرق نشر مكون واحد `--limit` عبر 50 مضيفًا وقتًا أطول ولكنه يتعامل مع كل شيء. استخدم `all.yml` تشغيل لتضييق النطاق أكثر.

مرجع سريع

كتيبات اللعب على المستوى الأعلى

كتيب اللعب	النطاق	الوصف
`all.yml`	الشبكة الكاملة	المستخدمون: نشر كامل، DNS، مختبر، المراقبة، الأجهزة، EPC، IMS، ePDG، TWAG، SS7، OCS، RAN، 5G Core، مراقب CRM، وفحص صحي نهائي (من `util_playbooks/`)
`epc.yml`	Packet Core	MME، SGW-C، PGW-C، UPF، HSS، DRA
`ims.yml`	الصوت/IMS	P/I/S-CSCF، خادم التطبيقات، الرسائل، OmniSEP (BSF/الاستحقاقات/XCAP)
`ocs.yml`	الشحن	OCS مزامنة، KeyDB مجموعة، CGrateS
`monitoring.yml`	القابلية للمراقبة	المصدرون، Grafana، Prometheus

كتيبات بنية تحتية

كتيب اللعب	الوصف
`common.yml`	تكوين نظام التشغيل الأساسي، الحزم، NTP، وكلاء التسجيل
`setup_users.yml`	حسابات المستخدمين المحليين ومفاتيح SSH
`dns.yml`	نشر خادم DNS
`license_server.yml`	خادم ترخيص OmniCore
`does_netplan_exist.yml`	تحقق مما إذا كان netplan مثبتًا على كل مضيف
`update_mtu.yml`	تعيين وتطبيق MTU الواجهة عبر netplan
`firewall.yml`	قواعد iptables/nftables
`apt_cache.yml`	مرآة APT محلية (إذا كانت مفعلة)

مكونات EPC

الوصف	المكون	كتيب اللعب
كيان إدارة الحركة (4G)	OmniMME	`omnimme.yml`
بوابة الخدمة لطبقة التحكم	OmniSGW-C	`omnisgwc.yml`
بوابة PDN لطبقة التحكم	OmniPGW-C	`omnipgwc.yml`
وظيفة مستوى المستخدم (SGW-U/PGW-U مجمعة)	OmniUPF	`upf.yml` / `omniupf.yml`
خادم المشتركين المنزليين	OmniHSS	`omnihss.yml` / `hss.yml`
وكيل توجيه Diameter	OmniDRA	`omnidra.yml`
بوابة الوصول اللاسلكي الموثوق	OmniTWAG	`omnitwag.yml`
بوابة بيانات الحزمة المتطورة (مكالمات WiFi)	OmniEPDG	`omniepdg.yml`
وكيل حركة GTP	GTP Proxy	`gtp_proxy.yml`

مكونات IMS

كتيب اللعب	المكون	الوصف
`pcscf.yml`	P-CSCF	وظيفة التحكم في جلسة المكالمات الوكيلة
`icscf.yml`	I-CSCF	CSCF الاستجواب
`scscf.yml`	S-CSCF	CSCF الخدمة
`as.yml`	OmniTAS	خادم تطبيقات الهاتف
`omnisep.yml`	OmniSEP	XCAP، خادم الاستحقاق، BSF، البريد الصوتي المرئي
`xcap.yml`	OmniSEP	اسم مستعار قديم لـ `omnisep.yml` (ينشر نفس دور `omnisep`)
`omnimessage.yml`	OmniMessage / IMS SMSC / MMSc	وحدة تحكم SMS-over-IP، SMPP، IMS SMSC (يتم نشرها عبر دور `cscf` مع `cscf_type: smsc_ims`)، وMMSc المستقل (الدور `mmsc`، المجموعة `mmsc`)

5G Core

المشترك مع `gc_nf` يتم تثبيت كل وظيفة شبكة بواسطة دور `5gc.yml`. يتم نشره عبر دور `5gc` لكل `nf_package`. انظر 5G Core لتفاصيل الدور، وتكوين كل NF، ومفاتيح وتثبيت الإصدار، وUDM HNET/SUCI.

الوصف	المكون	كتيب اللعب
ينشر جميع وظائف الشبكة المستقلة 5G	5G Core	`5gc.yml`
وظيفة مستودع الشبكة	OmniNRF	
وظيفة إدارة الوصول والحركة	OmniAMF	
وظيفة خادم المصادقة	OmniAUSF	
وظيفة دعم الربط	OmniBSF	
وظيفة الشحن	OmniCHF	
وظيفة اختيار شريحة الشبكة	OmniNSSF	
وظيفة التحكم في السياسة	OmniPCF	
وكيل الاتصال بالخدمة	OmniSCP	
وظيفة إدارة الجلسات	OmniSMF	
إدارة البيانات الموحدة	OmniUDM	
مستودع البيانات الموحد	OmniUDR	

الدائرة المبدلة

يتم نشره عبر `cs.yml` (الأدوار تحت `circuit_switched/`). انظر Circuit-Switched Core لتفاصيل دور BSC و MGW و MSC و SS7.

كتيب اللعب	المكون	الوصف
`cs.yml`	الدائرة المبدلة	ينشر مجال CS القديم
	BSC	وحدة التحكم في المحطة الأساسية (`circuit_switched/bsc`)
	MGW	بوابة الوسائط المستقلة (`circuit_switched/mgw`)
	OmniMSC	مركز التحويل المحمول (`circuit_switched/omnimsc`)
	SS7	كومة SS7/SIGTRAN (`omniss7` الدور)

الخدمات الداعمة

كتيب اللعب	الوصف
`ocs.yml`	نظام الشحن عبر الإنترنت (CGrateS + KeyDB)
`crm.yml`	بوابة إدارة العملاء (OmniCRMفي موجود الآن التكوين ؛ `group_vars/`)
`omniss7.yml`	بوابة SS7/SIGTRAN
`voipmonitor.yml`	التقاط حزم SIP/RTP (VoIPmonitor) على صندوق التقاط `siptrace`، بالإضافة إلى جامعي العملاء الخفيفين الاختياريين
`grafana.yml`	لوحات معلومات Grafana والتنبيهات
`ran_monitor.yml`	تكامل مراقبة RAN
`ussd.yml`	تطبيق رد الاتصال USSD لبوابة USSD الخاصة بـ HLR (`ussd` المجموعة). انظر Circuit-Switched.
`omniroam.yml`	مجموعة التجوال OmniRoam (واجهة TAP3/RAP، واجهة برمجة تطبيقات Elixir، مولد)
`omnilcs.yml`	خدمات الموقع OmniLCS وبث الخلايا
`device_tester.yml`	منصة اختبار وكيل الهاتف OmniWeb (أجهزة الاختبار)

توفير Proxmox (`proxmox.yml`، `proxmox_delete.yml`) موجود تحت `util_playbooks/`. انظر Utility Playbooks و نشر Proxmox VM/LXC.

العمليات

كتيب اللعب	الوصف
`backup.yml`	نسخ احتياطي لقواعد البيانات والتكوينات
`reboot.yml`	إعادة تشغيل مضبوطة للمضيفين
`shutdown.yml`	إيقاف تشغيل سلس
`apt_update.yml`	تحديث الحزم
`apt_refresh_metadata.yml`	تحديث بيانات التعريف الخاصة بذاكرة التخزين المؤقت APT
`speedtest.yml`	اختبار عرض النطاق الترددي للشبكة

كتيبات الاستعادة

كتيب اللعب	الوصف
`restore_applicationserver.yml`	استعادة OmniTAS من النسخة الاحتياطية
`restore_smsc.yml`	استعادة SMSC من النسخة الاحتياطية: وحدة تحكم OmniMessage، بوابة SMPP، وعقد IMS SMSc

> ملاحظة: `health_check.yml`، `restore_hss.yml`، و `restore_ocs.yml` في **ليست** Utility Playbooks. يقوم انظر `util_playbooks/`. إنها موجودة في `services/`. بتشغيل الفحص الصحي كـ `all.yml` `../util_playbooks/health_check.yml`.

الاستخدام

نشر كل شيء

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

نشر نظام فرعي محدد

```
# فقط النواة الحزم
ansible-playbook -i hosts/customer/host_files/production.yml
services/epc.yml

# فقط IMS/الصوت
ansible-playbook -i hosts/customer/host_files/production.yml
services/ims.yml
```

نشر مكون واحد

```
# تحديث فقط MME
ansible-playbook -i hosts/customer/host_files/production.yml
services/omnimme.yml

# تحديث فقط المراقبة
ansible-playbook -i hosts/customer/host_files/production.yml
services/monitoring.yml
```

الحد من المضيفين المحددين

```
# تشغيل all.yml ولكن فقط على مضيف واحد
ansible-playbook -i hosts/customer/host_files/production.yml
services/all.yml --limit mme01

# التشغيل على مضيفين محددين متعددين
ansible-playbook -i hosts/customer/host_files/production.yml
services/all.yml --limit "mme01,hss01"

# التشغيل على مجموعة
ansible-playbook -i hosts/customer/host_files/production.yml
services/all.yml --limit mme
```

الوثائق ذات الصلة

- معمارية النشر - سير العمل العام للنشر
- تكوين ملف المضيف - تعريف البنية التحتية
- تكوين المتغيرات الجماعية - التخصيص
- كتيبات الأدوات - أدوات التشغيل (فحص صحي، استعادة، إلخ.)
- المراقبة والقابلية للمراقبة - Grafana، Prometheus، التنبيهات

كتيبات الأدوات

توفر كتيبات الأدوات أدوات تشغيلية لإدارة بنية OmniCore التحتية المنفذة. تقع هذه الكتيبات في `util_playbooks/` ويمكن تشغيلها بشكل مستقل لأداء مهام الصيانة واستكشاف الأخطاء وإصلاح الشائعة.

مرجع سريع

الغرض	كتيب الأدوات
توفير الآلات الافتراضية و/أو حاويات LXC على Proxmox (حسب `deployment_type`)	`proxmox.yml`
حذف الآلات الافتراضية/حاويات LXC على Proxmox	`proxmox_delete.yml`
فرض إعادة تشغيل الآلات الافتراضية/حاويات LXC عبر واجهة برمجة تطبيقات Proxmox عندما لا تستجيب SSH	`proxmox_restart_hung.yml`
توفير الآلات الافتراضية على VMware vSphere	`vmware.yml`
حذف الآلات الافتراضية على VMware vSphere	`vmware_delete.yml`
إنشاء تقرير شامل عن صحة جميع الخدمات	`health_check.yml`
استعادة قاعدة بيانات HSS و/أو التكوين من النسخة الاحتياطية	`restore_hss.yml`
تهيئة عقدة OmniHSS فارغة من نظير مأهول (تحميل بيانات النسخ الأولية)	`seed_hss_replication.yml`
تدقيق شبكة OmniHSS لاكتشاف انحراف البيانات (عدد الصفوف + تجزئة المحتوى لكل جدول مرجعي عبر جميع العقد)؛ قراءة فقط	`audit_hss_drift.yml`
تسوية انحراف بيانات شبكة OmniHSS عن طريق ملء الصفوف المفقودة عبر العقد (تشغيل جاف افتراضي)	`reconcile_hss_drift.yml`
استعادة OCS KeyDB وMySQL والتكوين من النسخة الاحتياطية	`restore_ocs.yml`
إنشاء وثائق الشبكة مع مخططات Mermaid	`ip_plan_generator.yml`

كتيب الأدوات	الغرض
`get_ports.yml`	تدقيق المنافذ المفتوحة والخدمات المستمعة عبر جميع المضيفين
`getLocalCapture.yml`	استرجاع ملفات التقاط الحزم من المضيفين
`delete_local_user.yml`	إزالة حساب مستخدم محلي من جميع المضيفين
`clear_mnesia.yml`	مسح قاعدة بيانات Mnesia لـ OmniSS7 / OmniMessage وإعادة تشغيلها (تدميري)
`delete_license.yml`	حذف ملف الترخيص على خادم الترخيص وإعادة تشغيله

توفير الآلات الافتراضية

تقوم كتيبات توفير الآلات الافتراضية بإنشاء وإدارة الآلات الافتراضية على منصات المحاكاة الافتراضية. تدعم جميع الكتيبات `--limit` لاستهداف مضيفين أو مجموعات معينة.

للحصول على تكوين مفصل راجع نشر Proxmox.

Proxmox

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

يقوم `proxmox.yml` بتوفير كل من الآلات الافتراضية وحاويات LXC؛ يتم تحديد ما إذا كان `deployment_type` المحلولة (تجاوز → إدخال لكل مضيف → `proxmoxServers` → الافتراضي `vm`) بواسطة المضيف المعطى يصبح آلة افتراضية أو حاوية LXC. راجع نشر Proxmox.

```
# توفير (آلات افتراضية و/أو حاويات LXC، حسب deployment_type)
ansible-playbook -i hosts/customer/host_files/production.yml
util_playbooks/proxmox.yml

# توفير مجموعة معينة فقط
ansible-playbook -i hosts/customer/host_files/production.yml
util_playbooks/proxmox.yml --limit mme

# حذف الآلات الافتراضية/حاويات LXC
ansible-playbook -i hosts/customer/host_files/production.yml
util_playbooks/proxmox_delete.yml --limit old-host
```

إعادة تشغيل المضيفين المعلقين على Proxmox

الملف: `util_playbooks/proxmox_restart_hung.yml`

يتحقق من استجابة SSH لكل مضيف ويفرض إعادة تشغيل أي آلات افتراضية أو حاويات LXC غير مستجيبة عبر واجهة برمجة تطبيقات Proxmox. يتم إيقاف تشغيل آلات QEMU بشكل قسري (أو إعادة تشغيلها إذا كانت قيد التشغيل بالفعل) وإعادة تشغيلها؛ يتم بدء تشغيل حاويات LXC. يتم تخطي المضيفين الذين يستجيبون لـ SSH.

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

# تقييد إلى مضيف أو مجموعة واحدة
ansible-playbook -i hosts/customer/host_files/production.yml
util_playbooks/proxmox_restart_hung.yml --limit hss01
```

VMware vSphere

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

تتضمن كلا ال`util_playbooks/vm/vmware/` تتبين ملفات المهام لكل آلة افتراضية تحت (`create_individual_vm.yml` و `delete_individual_vm.yml`).
`vmware.yml` ينتظر أيضًا SSH هو غلاف رقيق حول مهمة الحذف؛ بينما `vmware_delete.yml` القديمة netplan ويطبق netplan بعد التوفير، ثم يزيل ملفات على الآلة الافتراضية الجديدة.

المتطلبات المسبقة: يتطلب تثبيت تبعيات Python (`uv sync`).

الاستخدام:

```
# توفير الآلات الافتراضية
ansible-playbook -i hosts/customer/host_files/production.yml
util_playbooks/vmware.yml

# توفير مجموعة معينة فقط
ansible-playbook -i hosts/customer/host_files/production.yml
util_playbooks/vmware.yml --limit mme

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

المتغيرات المطلوبة:

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

Vultr

الملفات: `util_playbooks/vm/vultr/vmProvisionVultr.yml`، `util_playbooks/vm/vultr/vmTeardownVultr.yml`، `util_playbooks/vm/vultr/vmCommonSetup.yml`

يوفر وينزع الحالات على Vultr عبر واجهة برمجة تطبيقات Vultr. يقوم `vmProvisionVultr.yml` بإنشاء الحالات، و `vmTeardownVultr.yml` بإزالتها، و `vmCommonSetup.yml` بأداء إعداد قاعدة بعد التوفير (DNS، إلخ، إعادة التشغيل). الثلاثة هي كتيبات نقطة دخول على مستوى عالٍ بدلاً من تضمينات المهام. تقرأ رمز API من متغير البيئة

من نفس الدليل، لذا قم `cloud-config.yaml` ويشير كتيب التوفير إلى `VULTR_API_TOKEN`، بتشغيلها من `util_playbooks/vm/vultr/`.

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

فحص الصحة

الملف: `util_playbooks/health_check.yml`

ينشئ تقرير صحة شامل بتنسيق HTML يغطي جميع خدمات OmniCore وOmniCall المنفذة.

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

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

المعلومات المجمعة

المكون	البيانات المجمعة
جميع الخدمات	حالة الخدمة، الإصدار، وقت التشغيل
OmniHSS	حالة قاعدة البيانات، اتصالات نظير Diameter
OmniDRA	اتصالات نظير Diameter والحالة
OmniTAS	المكالمات النشطة، الجلسات، التسجيلات، استخدام وحدة المعالجة المركزية
OCS	حالة تكرار KeyDB

استعادة HSS

الملف: `util_playbooks/restore_hss.yml`

يستعيد OmniHSS من ملفات النسخ الاحتياطي. يدعم استعادة قاعدة البيانات فقط، أو التكوين فقط، أو كليهما.

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

تنسيقات ملفات النسخ الاحتياطي

النوع	نمط اسم الملف	المحتويات
قاعدة البيانات	`hss_dump_<hostname>_<timestamp>.sql`	تفريغ قاعدة بيانات OmniHSS (الافتراضي `omnihss`)
التكوين	`hss_<hostname>_<timestamp>.tar.gz`	أرشيف من دليل `/etc/omnihss`

يتم عمل نسخة احتياطية من الحالة الحالية لـ HSS (قاعدة البيانات و/أو التكوين) قبل تشغيل الاستعادة.

المطالبات

المطالبة	مطلوب	الوصف
مسار تفريغ HSS SQL	لا	المسار الكامل إلى ملف تفريغ HSS SQL (اتركه فارغًا لتخطي استعادة قاعدة البيانات)
مسار ملف تكوين HSS tar.gz	لا	المسار الكامل إلى ملف تكوين HSS tar.gz (اتركه فارغًا لتخطي استعادة التكوين)
تأكيد	نعم	لتأكيد عملية الاستعادة على المضيفين المذكورين اكتب `yes`

تهيئة تكرار HSS

الملف: `util_playbooks/seed_hss_replication.yml`

تنضم إلى شبكة تكرار نشطة نشطة OmniHSS يقوم بتحميل البيانات الأولية لمرة واحدة لعقدة فارغة جديدة (مأهولة بالفعل. تظهر عقدة جديدة يتم إنشاء الاشتراكات مع `copy_data = false`)؛ يعيد هذا الكتيب إنشاء الاشتراك إلى نظير مصدر مختار مع ��احد `copy_data = true`، بحيث تقوم PostgreSQL بنسخ البيانات الموجودة ثم تواصل البث دون أي فجوة.

يعمل في أحد وضعين:

تفاعلي (افتراضي): اترك متغيرات `seed_source_*` فارغة. يكتشف الكتيب أي هدف محدود فارغ. ثم يستفسر عن الأقران الصالحين للعثور على الأقران التي تحتوي على بيانات، ويقوم بإدراجها مع عدد المشتركين، ويطلب منك اختيار مصدر. يقوم بتغذية الأهداف الفارغة فقط.

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

```
الأقران التي تحتوي على بيانات متاحة للتغذية منها:
  1) dev-hss01 (10.4.10.142) - 838 مشترك
أدخل رقم المصدر للتغذية منه
```

صريح: مرر المصدر مباشرة لتخطي القائمة. هذا **يجب** إعادة التغذية حتى لو كانت الهدف تحتوي بالفعل على بيانات (استخدم لاستعادة).

```
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"
```

المعلمات

المعلمة	مطلوب	الافتراضي	الوصف
`seed_source_name`	لا	(مطالب)	اسم مضيف الجرد للنظير المصدر المأهول. يستخدم لتسمية الاشتراك (`sub_from_<name>`) وحل نشره (`<name>_pub`). تركه في وضع التفاعل.
`seed_source_host`	لا	(مطالب)	عنوان IP للنظير المصدر، قابل للوصول على منفذ PostgreSQL الخاص به. اتركه في وضع التفاعل.
`seed_source_port`	لا	منفذ `omnihss.database_port` المستهدف (5432)	منفذ PostgreSQL على النظير المصدر.

> يجب تقديم كل من `seed_source_name` و `seed_source_host` معًا لاستخدام الوضع الصريح؛ عدم تقديم أي منهما يؤدي إلى تفعيل وضع التفاعل.

السلوك

1. يعد عدد المشتركين على كل هدف محدود لتحديد أي منها فارغ.
2. يستفسر عن الأقران الصالحين (مجموعة `hss` المحلية + `connected_omnihss`) للعثور على الأقران التي تحمل بيانات، ويقو�ّ بإدراجها. **وضع التفاعل:** يطلب تأكيد `yes` ويطلب اختيارًا. **وضع الصريح:** يطلب تأكيد.
3. يضمن أن عنوان IP المصدر مسموح به في `pg_hba.conf` الخاص بالهدف.
4. يقوم بإسقاط وإعادة إنشاء الاشتراك إلى المصدر `WITH (copy_data = false)` (اسم `slot_name` لكل رابط هو `<target>_from_<source>`). يقوم ببث **جميع** الجداول ولكن لا يقوم بأداء مزامنة جداول جماعية.
5. يقوم بملء الصفوف الموجودة في **جدول الماستر** من المصدر باستخدام تحميل آمن من حيث التعارض (`pg_dump --data-only --inserts --on-conflict-do-nothing --disable-triggers`). يتم تضمين `subscriber_state`: كل مشترك يشير إلى واحد بواسطة FK، لذا فإن تخطيه سيجعل كل مشترك تم ملؤه يتيمًا. يتم **عدم** ملء جداول الجلسات المؤقتة (`pdn_session`، `lte_call`).
6. يبلغ عن عدد المشتركين على العقدة التي تم تغذيتها.

> **لماذا يتم تخطي جداول الجلسات:** إنها جداول نشطة؛ قد يتصادم `COPY` الجماعي لها على نظام حي (مفتاح مكرر) مع كتابات متزامنة ويؤدي إلى تعطل جداول المزامنة. يتم إعادة ملئها من النسخ المتماثل المتدفق وحركة المرور الحية بدلاً من ذلك. `subscriber_state` مختلف: يتم إدراج الصفوف فقط في وقت التوفير (1:1 مع المشترك) ويتم تحديثها فقط آمن. بدونها، تشير مشتركات الهدف إلى `--on-conflict-do-nothing` لاحقًا، لذا فإن صفوف الحالة التي لا تصل أبدًا، حيث تقوم النسخ المنطقية بإسقاط تحديثات الصفوف المفقودة بصمت.

> **مهم:** دائمًا `--limit` إلى العقدة (أو العقد) المستهدفة. في وضع التفاعل، يتم تخطي الأهداف غير الفارغة تلقائيًا. راجع تكرار OmniHSS وعمليات العقد للحصول على سير العمل الكامل.

تدقيق انحراف HSS

الملفات: `util_playbooks/audit_hss_drift.yml` (مستقل) و `roles/omnihss/tasks/audit_drift.yml` (التنفيذ المشترك).

يبلغ عن **انحراف البيانات** عبر شبكة OmniHSS النشطة النشطة. يتم إنشاء اشتراكات النسخ المنطقية مع `copy_data = false`، لذا فإن البيانا�ّ التي انحرفت قب�ّ رابط الشبكة (اختلافات وقت التغذية) أو عبر كتابة مباشرة/غير مكررة لا تتم تسويتها أبدًا ولا يتم تنبيهها. تظهر

لكل موقع رؤيتها لأن الشبكة تمتد Prometheus النسخ 0 تأخير بينما تختلف البيانات، ولا يمكن لـ عبر موقعين.

(`connected_omnihss` المحلية + أقران `hss` مجموعة) يتصل التدقيق بكل عضو في الشبكة غير المعتمد على الترتيب md5 وبالنسبة لكل جدول مرجعي/تكوين، يقارن عدد الصفوف الدقيق **و** لالتقاط انحراف (عدد + أقصى إصدار) `schema_migrations` للمحتويات الكاملة. كما يقارن المخطط. يتم استبعاد جداول الحالة الساخنة (`subscriber_state`, `pdn_session`, `lte_call`): فهي تختلف بشكل مشروع تحت حركة المرور الحية. يتم تضمين فحص صحي هيكلي واحد فوق المقارنة عبر العقد: `__orphaned_subscribers__`، عدد المشتركين لكل عقدة يؤدي عدد غير صفري على **أي** عقدة إلى فشل. `subscriber_state` الذين تفتقر صفوفهم إلى `subscriber` بدون التدقيق حتى عندما تتفق كل عقدة. قد يؤدي تغذية نسخ `subscriber_state` إلى يتيم كل عقدة بشكل متطابق، وهو ما لا يمكن أن تراه المقارنة النقية للأقران، لذا Postgres عبر العقد. يتم مركز المقارنة على عداء واحد يصل إلى كل قاعدة بيانات `--limit`. فإن تشغيل واحد يغطي الشبكة الكاملة عبر الموقعين من أي جرد، وهو آمن لـ

(عندما يتم تمكين النسخ) `omnihss` في نهاية دور **HSS. يتم تشغيله تلقائيًا بعد كل نشر لـ** يتم تشغيل نفس التدقيق كتحقق من التناسق بعد النشر، وبشكل افتراضي **يفشل التشغيل** إذا تم العثور على انحراف، لذا فإن الشبكة المعطلة من المستحيل تفويتها أثناء النشر. إنه غير تفاعلي (لا المجمع، `services/all.yml` لنشر `free` توقف): يجب أن يكون التدقيق آمنًا تحت استراتيجية حيث ستؤدي التوقف إلى حدوث خطأ. في ذلك النشر المجمع، يتم كتم التدقيق داخل الدور ويتم تشغيل التدقيق بدلاً من ذلك مرة واحدة في كتيب **خطّي** مخصص في نهاية `services/deploy_nfs.yml`. يتم التحكم في السلوك بواسطة متغيري دور:

المتغير	الافتراضي	التأثير
`omnihss_audit_drift`	`true`	تشغيل التدقيق في نهاية الدور
`omnihss_audit_fail_on_drift`	`true`	فشل التشغيل في حالة الانحراف؛ قم بتعيين `false` للتقرير فقط (cron/CI غير تفاعلي)

التدقيق هو **قراءة فقط** في كلتا الحالتين. الافتراضي لكتيب المستقل هو التقرير ثم **الفشل** في حالة الانحراف (cron/CI حتى يمكنه التحكم في):

```
# عند الطلب
ansible-playbook -i hosts/<inv>/host_files/<site>.yml
util_playbooks/audit_hss_drift.yml
# تقرير فقط، دائمًا ينجح
ansible-playbook -i ... util_playbooks/audit_hss_drift.yml -e
audit_fail_on_drift=false
# تعطيل التدقيق التلقائي لنشر واحد
ansible-playbook -i ... services/omnihss.yml -e
omnihss_audit_drift=false
```

HSS تسوية انحراف

الملف: `util_playbooks/reconcile_hss_drift.yml`

يعيد جداول المرجع المتباينة إلى التزامن من خلال جعل كل عقدة تحمل **اتحاد** الصفوف: لكل جدول متباين يقوم بملء الصفوف التي تفتقر إليها العقدة من أقرانها، باستخدام نفس الأسلوب الآمن من حيث التعارض كما في كتيب التغذية (`pg_dump --data-only --inserts --on-conflict-do-nothing --disable-triggers`).

- **غير تدميري**: يتم إدراج الصفوف فقط (تخطي التعارضات). لا يتم حذف أي شيء أو تحديثه. يحل "تفتقر العقدة إلى صفوف"؛ لا يحل "نفس المفتاح الأساسي، قيم مختلفة" أو يزيل الصفوف التي يجب ألا توجد. تحتاج تلك إلى قرار بشري.
- **تفاعلي بشكل افتراضي**: يقوم بالمسح، ويطبع الخطة، و**يسأل** قبل (للمتابعة `yes`) الكتابة؛ أي شيء آخر يتسبب في الإلغاء دون تغييرات. استخدم `-e dry_run=true` للنظر فقط، أو `-e assume_yes=true` للتشغيلات غير التفاعلية.
- تتم الكتابات محليًا على كل هدف كـ superuser لـ postgres (لـ `--disable-triggers`); يتم قراءة الأقران فقط، كمستخدم التطبيق.
- **اتحاد متماثل، وليس موثوقًا للإنتاج**: تشغيله ضد موقع يسحب *صفوف ذلك* ال��وقع إلى الآخرين. لا يتعامل مع الإنتاج كالأستاذ، لذا فإن تشغيله ضد جرد المختبر سيؤدي إلى نشر صفوف المختبر نحو الإنتاج. إذا كان يجب أن يكون الإنتاج هو المصدر الحقيقي، قم بتشغيله من جرد الإنتاج (أو اطلب وضع الموثوقية).
- **عبر المواقع:** يمكن أن تكون العقدة هدفًا فقط عندما تكون في الكتيب، لذا قم بتشغيل هذا ضد **كل** جرد موقع لتغطية كل عقدة. يقوم إعادة المسح في النهاية بالإبلاغ عن أي شيء لا يزال يحتاج إلى تشغيل.

```
# تفاعلي (افتراضي): يسأل قبل الكتابة
ansible-playbook -i hosts/<inv>/host_files/<site>.yml
util_playbooks/reconcile_hss_drift.yml
# نظر فقط، لا يكتب أبدًا
ansible-playbook -i ... util_playbooks/reconcile_hss_drift.yml -e
dry_run=true
# تطبيق غير تفاعلي (CI)
ansible-playbook -i ... util_playbooks/reconcile_hss_drift.yml -e
assume_yes=true
```

> أولاً لرؤية ما انحرف، ومرة أخرى بعد التسوية (ضد `audit_hss_drift.yml`) قم بتشغيل كلا الجردين) لتأكيد التقارب. انظر تك��ار وعمليات العقد OmniHSS.

استعادة OCS

الملف: `util_playbooks/restore_ocs.yml`

يستعيد OCS (CGrateS) من ملفات النسخ الاحتياطي. يتعامل مع تكرار KeyDB متعدد الماستر من خلال الاستعادة إلى عقدة واحدة والسماح للنسخ بالتزامن إلى الآخرين.

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

عملية الاستعادة

1. يوقف CGrateS وKeyDB على جميع عقد OCS
2. يمسح ملفات AOF لمنع التعارض مع البيانات المستعادة
3. يستعيد KeyDB RDB إلى العقدة الأولى، يبدأ KeyDB، ويترك النسخ تتزامن
4. يستعيد MySQL StoreDB (اختياري)
5. يستعيد تكوين `/etc/cgrates` (اختياري)
6. يبدأ CGrateS على جميع العقد

تنسيقات ملفات النسخ الاحتياطي

النوع	نمط اسم الملف	المحتويات
قاعدة بيانات KeyDB	`keydb_dump_<hostname>_<timestamp>.rdb`	لقطة RDB لـ KeyDB
قاعدة بيانات MySQL StoreDB	`cgrates_dump_<hostname>_<timestamp>.sql`	تفريغ MySQL لقاعدة بيانات `cgrates`
التكوين	`cgrates_<hostname>_<timestamp>.tar.gz`	أرشيف من دليل `/etc/cgrates`

المطالبات

المطالبة	مطلوب	الوصف
مسار تفريغ KeyDB RDB	نعم	المسار الكامل إلى ملف النسخ الاحتياطي KeyDB RDB
مسار تفريغ MySQL SQL	لا	المسار الكامل إلى تفريغ SQL لـ CGrateS (تخطي للحفاظ على StoreDB الحالي)
مسار ملف تكوين tar.gz	لا	المسار الكامل إلى النسخة الاحتياطية للتكوين (تخطي للحفاظ على التكوين الحالي)

مولد خطة IP

الملف: `util_playbooks/ip_plan_generator.yml`

ينشئ وثائق الشبكة من الجرد، بما في ذلك:

- تعيينات IP للمضيف (NICs الأساسية والثانوية)
- نظرة عامة على مقطع الشبكة
- مخططات الاتصال بالواجهة (Diameter، GTP، PFCP، SIP، SS7)

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

ملفات الإخراج

الملف	التنسيق	الوصف
`/tmp/ip_plan_<customer>_<site>.md`	Markdown	وثائق نصية
`/tmp/ip_plan_<customer>_<site>.html`	HTML	مخطط تفاعلي مع طبقات قابلة للتصفية

تدقيق المنافذ

الملف: `util_playbooks/get_ports.yml`

يدقق جميع المنافذ المستمعة عبر النشر وينشئ وثائق.

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

ملفات الإخراج

الملف	الوصف
`/tmp/all_ports.csv`	CSV مع اسم المضيف، IP، الخدمة، المنفذ، البروتوكول
`./open_ports.rst`	جدول reStructuredText لوثائق Sphinx

البيانات المجمعة

الحقل	الوصف
اسم المضيف	اسم مضيف الجرد
IP	عنوان IP الخاص بـ `ansible_host`
إصدار IP	IPv4 أو IPv6
النقل	UDP أو TCP
المنفذ	رقم المنفذ المستمع
الخدمة	اسم العملية

استرجاع الالتقاط المحلي

الملف: `util_playbooks/getLocalCapture.yml`

يسترجع أحدث ملفي التقاط حزم من دليل `/etc/localcapture` لكل مضيف.

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

الإ��راج: `./localCapturePcaps/<hostname>/*.pcap`

إدارة المستخدمين

الملف: `util_playbooks/delete_local_user.yml`

يزيل حساب مستخدم محلي من جميع المضيفين في الجرد.

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

المطالبة: أدخل اسم المستخدم للحذف عند المطالبة.

مسح قاعدة بيانات Mnesia

الملف: `util_playbooks/clear_mnesia.yml`

يوقف الخدمة، يحذف دليل قاعدة بيانات Mnesia الخاصة بها، ويعيد تشغيلها. يعمل ضد مجموعة `ss7` (OmniSS7) ومجموعة `omnimessage` (OmniMessage).

> **تحذير:** هذا تدميري. يتم حذف قاعدة بيانات Mnesia بشكل دائم وإعادة بنائها فارغة عند إعادة التشغيل.

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

# تقييد إلى خدمة أو مضيف واحد
ansible-playbook -i hosts/customer/host_files/production.yml
util_playbooks/clear_mnesia.yml --limit ss7
```

حذف الترخيص

الملف: `util_playbooks/delete_license.yml`

يحذف `/etc/license_server/license.json` على مجموعة `license_server` ويعيد تشغيل خادم الترخيص.

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

تشغيل كتيبات الأدوات

الصياغة الأساسية

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

الخيارات الشائعة

الخيار	الوصف
`-i <inventory>`	تحديد ملف الجرد
`--limit <hosts>`	تقييد إلى مضيفين أو مجموعات معينة
`-v` / `-vv` / `-vvv`	زيادة مستوى التفاصيل
`--check`	تشغيل جاف (عرض ما سيتغير)
`--diff`	عرض اختلافات الملفات

الأمثلة

```
# تشغيل فحص الصحة على الإنتاج
ansible-playbook -i hosts/acme/host_files/production.yml
util_playbooks/health_check.yml

# استعادة HSS على مضيف معين
ansible-playbook -i hosts/acme/host_files/production.yml
util_playbooks/restore_hss.yml --limit hss01

# إنشاء خطة IP مع إخراج مفصل
ansible-playbook -i hosts/acme/host_files/production.yml
util_playbooks/ip_plan_generator.yml -v
```