Orquestación, automatización y virtualización (OAV) en el *core* de la red de GÉANT

GÉANT Automation and Orchestration Team (GOAT)

Carolina Fernández, i2CAT

Participante en GN5-1/WP7T2

XXXI Jornadas Técnicas RedIRIS, Zaragoza, España

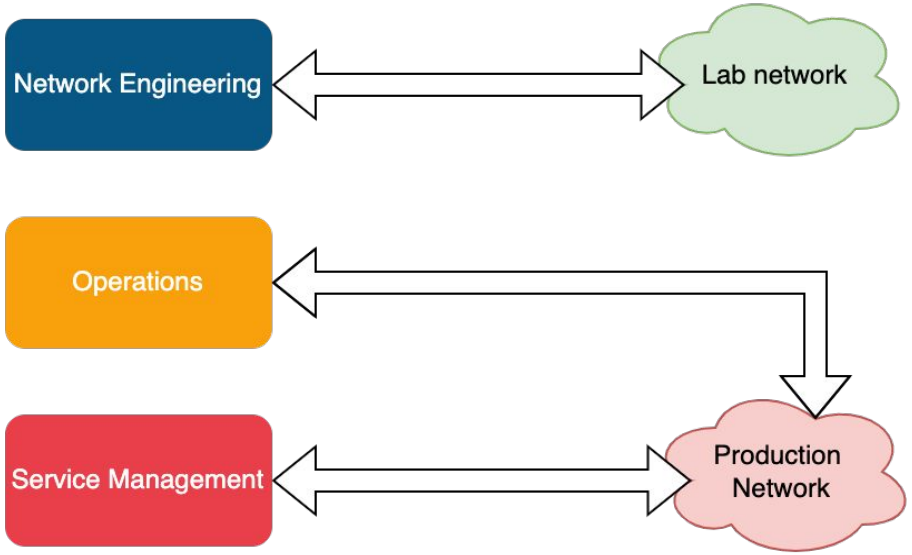
14 de junio de 2023

Agenda

- El *core* de la red: situación actual y deseada
- GÉANT Automation Platform (GAP)
  1. Pilares
  2. Modelos
  3. Ejemplo
  4. Conclusiones
- Equipo y referencias

El *core* de la red: situación actual

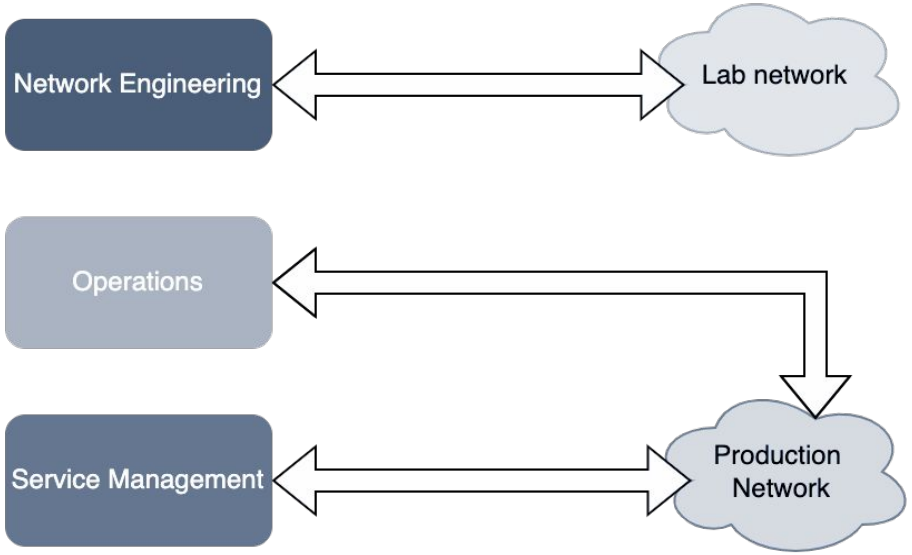
- Diferentes equipos
- Diferentes procedimientos y herramientas
- Transmisión de conocimiento manual
- Documentación esparcida por varios entornos (nodos), documentos (Word, wikis, *scripts*), etc

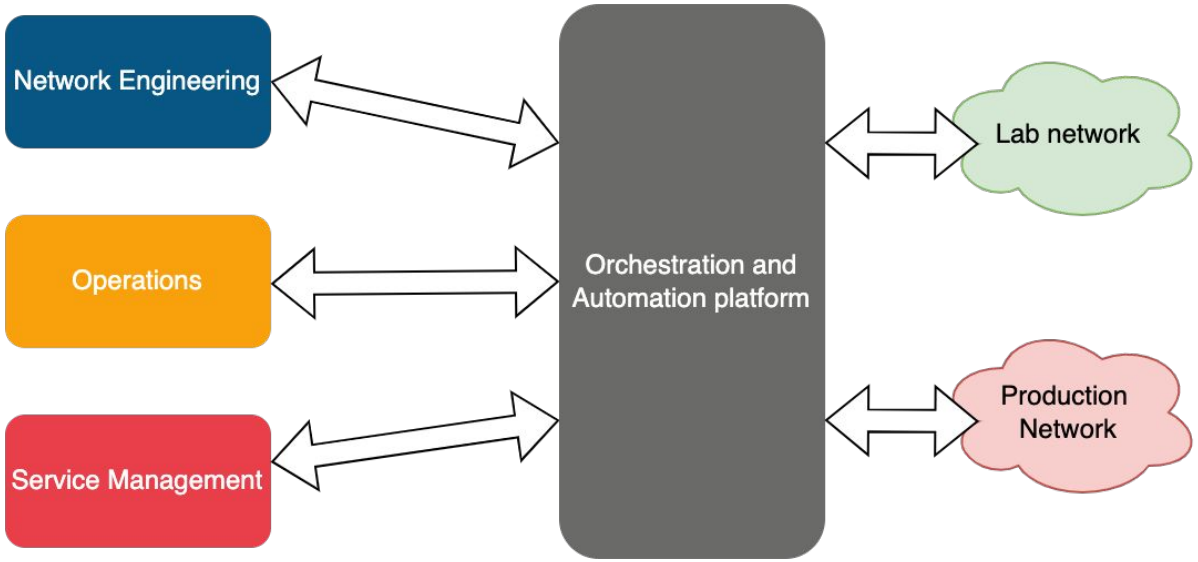


↑ *heterogeneidad* ⇒ ↑ *variabilidad*

El *core* de la red: situación deseada

- Diferentes equipos
- Diferentes procedimientos y herramientas
- Transmisión de conocimiento manual
- Documentación esparcida por varios entornos (nodos), documentos (Word, wikis, *scripts*), etc



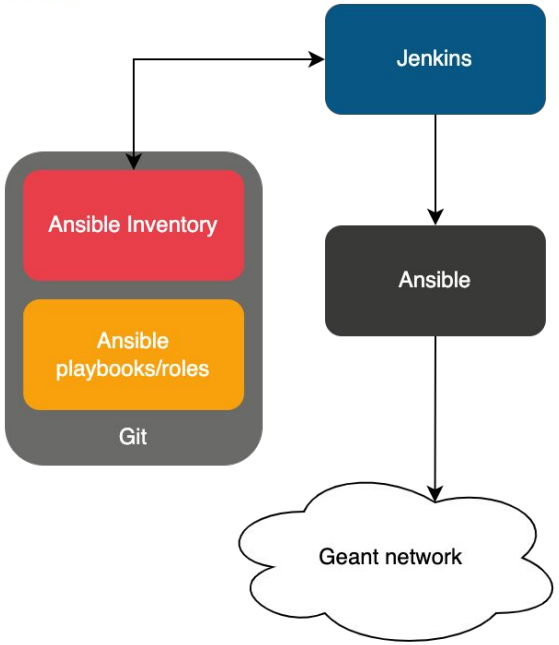


- Diferentes equipos
- Mismos procedimientos y herramientas
- Transmisión de conocimiento automática e instantánea
- Documentación dentro de los procesos y en puntos centrales de la misma capa compartida

↓ *heterogeneidad* ⇒ ↓ *variabilidad*

GÉANT Automation Platform (GAP): iteración #1 – pilares

- Especialmente basada en Ansible
  - Inventario: Ansible (YAML)
  - Modelado: roles y *playbooks* relacionados con los servicios (YAML)
- Jenkins como orquestador / *workflow engine* interactuando con:
  - Ansible *playbooks*
- Operaciones realizadas:
  - Aprovisionamiento de nodos
  - Aprovisionamiento de servicios IPTrunk (+pruebas)
  - Actualización del OS del nodo (+validaciones pre- & post-)
  - Inserción de un nodo en la red (*iBGP/MSDP mesh update*)
  - Otras operaciones estándar – p.ej., despliegue de TWAMP *probes* para medir el rendimiento entre nodos



GÉANT Automation Platform (GAP): iteración #1 – modelos (1)

Uso de capas

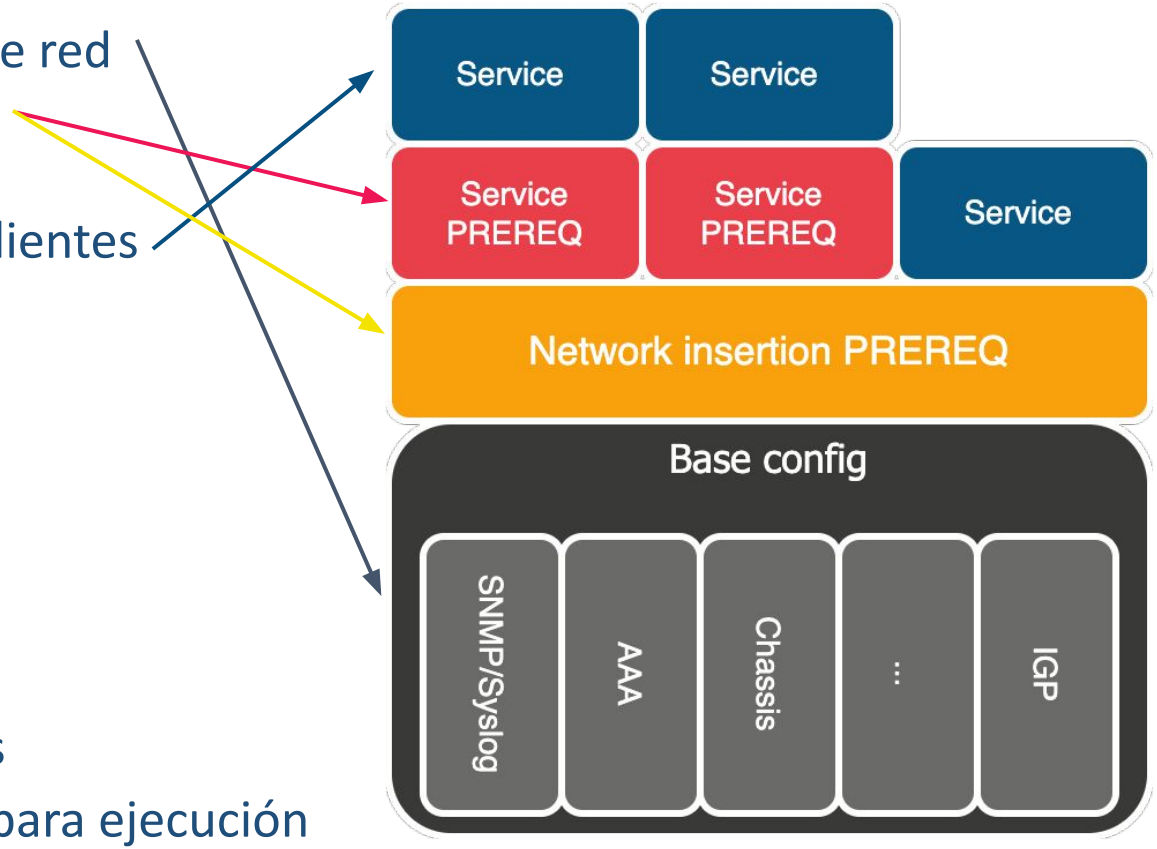
- Configuración base – similar para todo nodo de red
- Prerrequisitos de servicio & red – p.ej., *Virtual Routing & Forwarding*
- Servicios – con datos sobre infraestructura y clientes

Procesamiento secuencial

- Cada tipo de cambio se hace en un paso (fase)
- Un *workflow* (flujo) contiene múltiples pasos

Enfoque conservador

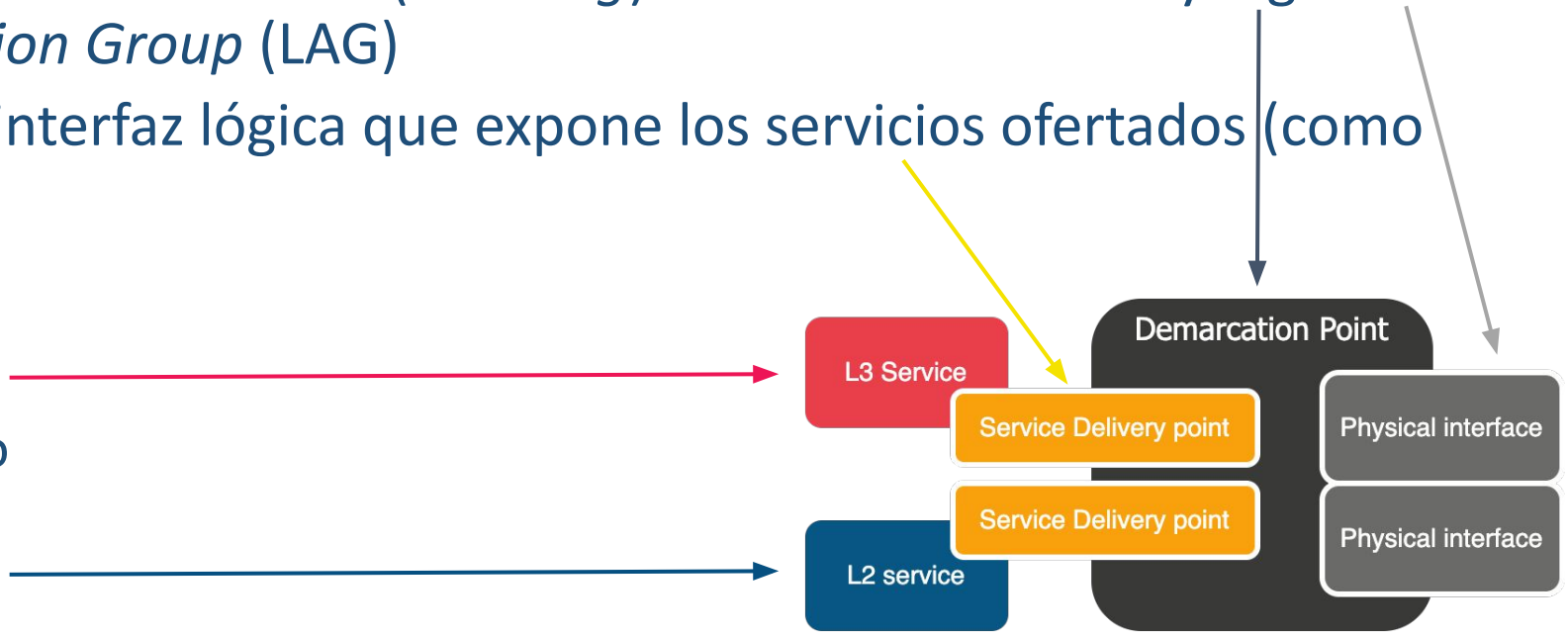
- Mínimos cambios (“quirúrgicos”) y justificados
- Dry-run de base + confirmación del operador para ejecución



GÉANT Automation Platform (GAP): iteración #1 – modelos (2)

Descomposición de servicios

- *Demarcation Point*: el punto de unión (*stitching*) de interfaces físicas y lógicas – p.ej., un *Link Aggregation Group* (LAG)
- *Service Delivery Point*: interfaz lógica que expone los servicios ofertados (como una VLAN)
- Tipos de servicio:
  - L3 vía BGP
  - L3 con acceso directo
  - L2 PTP
  - L2 multi-punto



GÉANT Automation Platform (GAP): iteración #1 – modelos (3)

Ejemplo básico: servicio IPTrunk

- Conecta dos puntos (lados A y B)
- LAG y miembros de interfaces dedicados
- Atributos específicos al link (p.ej., velocidad)

```
{% for trunk in trunks %}
    {% if inventory_hostname == trunk.config.nodeA.name  %}
        {% set local= trunk.config.nodeA %}
        {% set remote= trunk.config.nodeB %}
        {% set common= trunk.config.common %}
    {% endif %}
    {% if inventory_hostname == trunk.config.nodeB.name  %}
        {% set local= trunk.config.nodeB %}
        {% set remote= trunk.config.nodeA %}
        {% set common= trunk.config.common %}
    {% endif %}
…
```

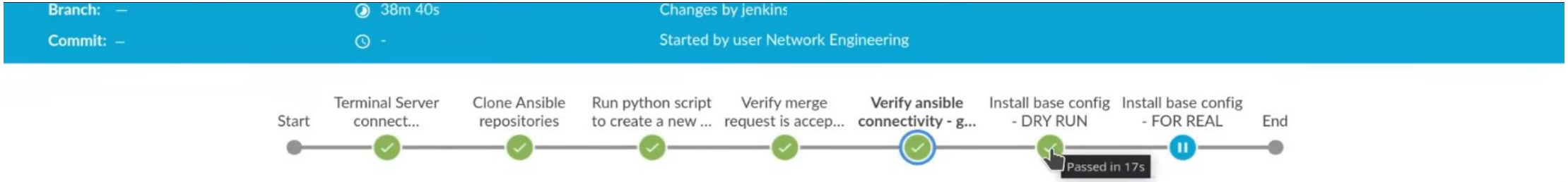
Playbook de Ansible

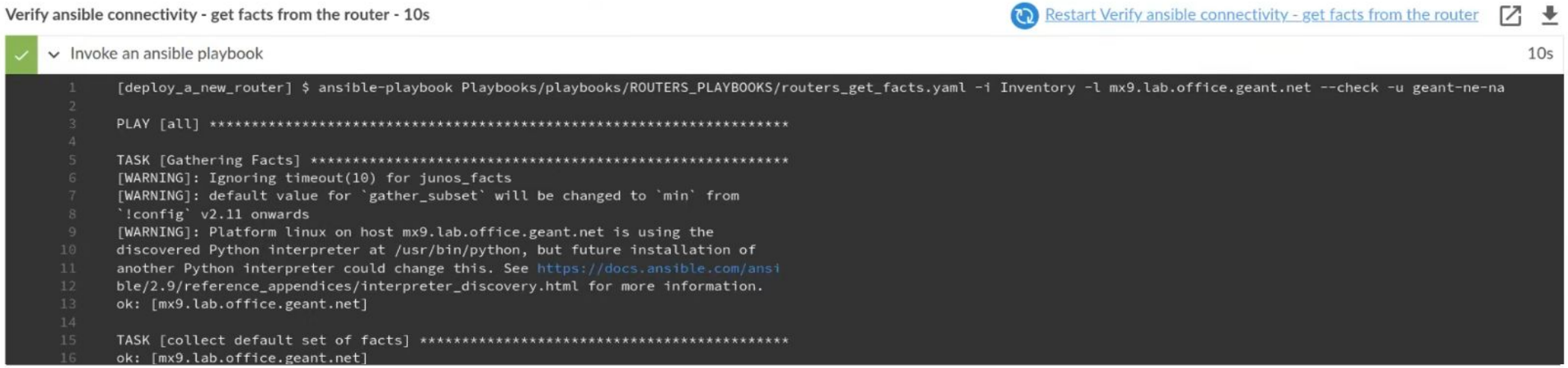
```
- id: GS-00007
    config:
      common:
        link_speed: '100'
        minimum_links: 1
        isis_metric: 9000
        is_leased_line: false
      nodeA:
        name: rt2.ams.nl.geant.net
        ae_name: ae1
        port_sid: GA-02033
        members:
          - et-0/0/2
        ipv4_address: 62.40.98.200/31
        ipv6_address: 2001:798:cc::51/126
      nodeB:
        name: mx1.ams.nl.geant.net
        ae_name: ae1
        port_sid: GA-02013
        members:
          - et-7/0/2
        ipv4_address: 62.40.98.201/31
        ipv6_address: 2001:798:cc::52/126
```

Inventario de Ansible

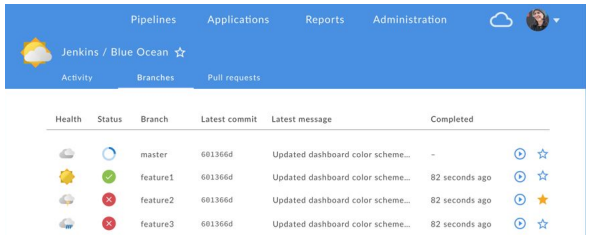
GÉANT Automation Platform (GAP): iteración #1 – ejemplo

Jenkins workflows (BlueOcean UI) como gestor de *workflows* + Ansible + Python *scripts* + GitLab





GÉANT Automation Platform (GAP): iteración #1 – conclusiones



Jenkins como orquestador de flujos

- Pros: interfaz web simple de operar (BlueOcean) y que abstrae Ansible
- Contras: gestión muy limitada, añadir funcionalidad con Groovy es complejo, soporte reducido, caso de uso no fácilmente extensible al entorno de operaciones de red

Modelado con YAML y versionado en Git

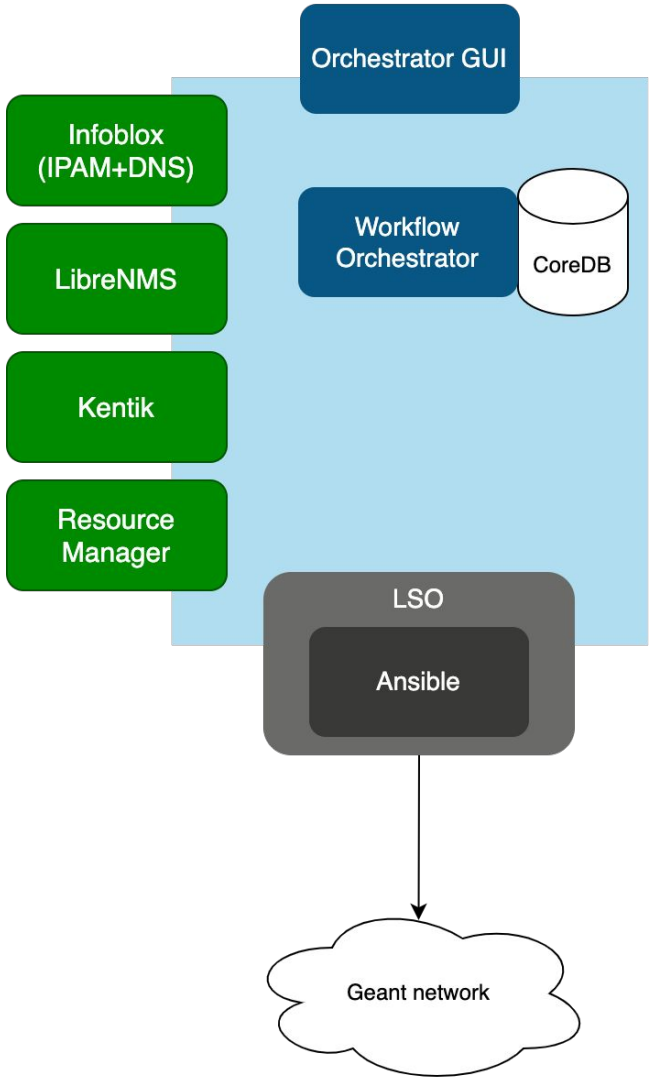
- Pros: se pueden expresar relaciones complejas con YAML
- Contras: los usuarios (operadores) no quieren tratar con IDEs y Git por cada cambio en la red, no escala (tratar con 100 trunks por archivo de inventario != 9000 sesiones BGP)

Operación

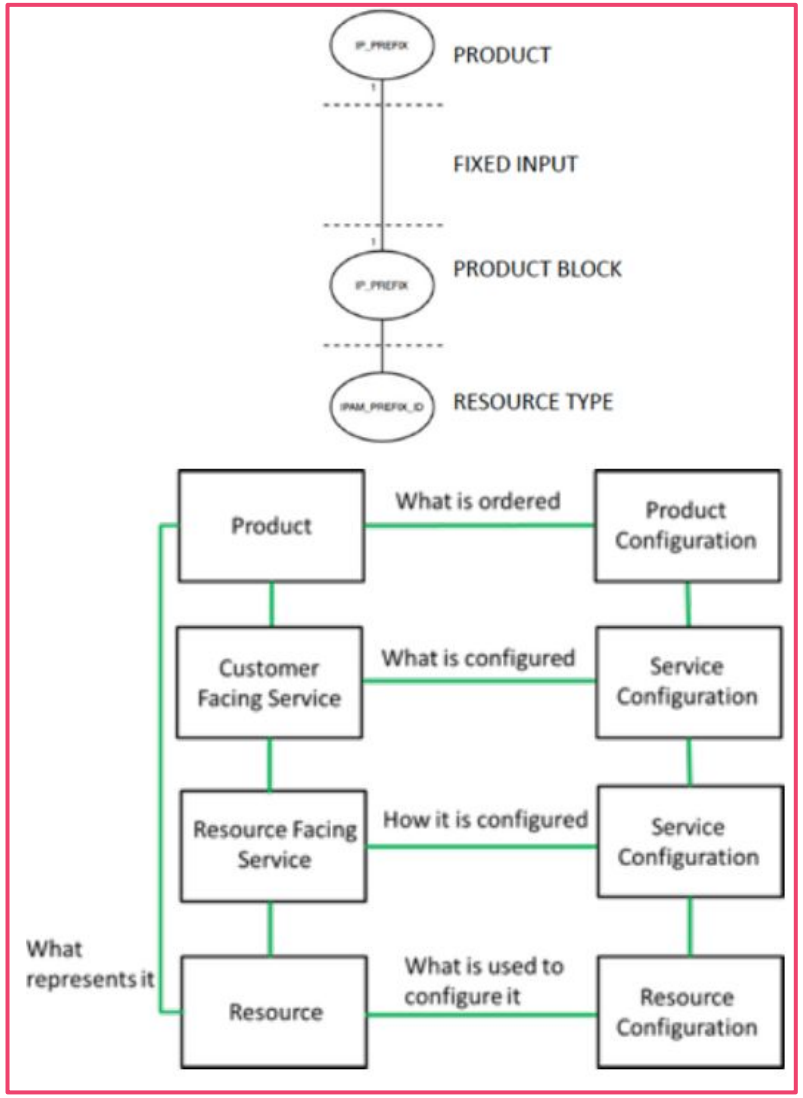
- Integración con otras herramientas mediante *scripts* individuales
- Identificación y evaluación de errores: Jenkins (centralizado) y *runners* (distribuido), *scripts* (según repo)

GÉANT Automation Platform (GAP): iteración #2 – pilares

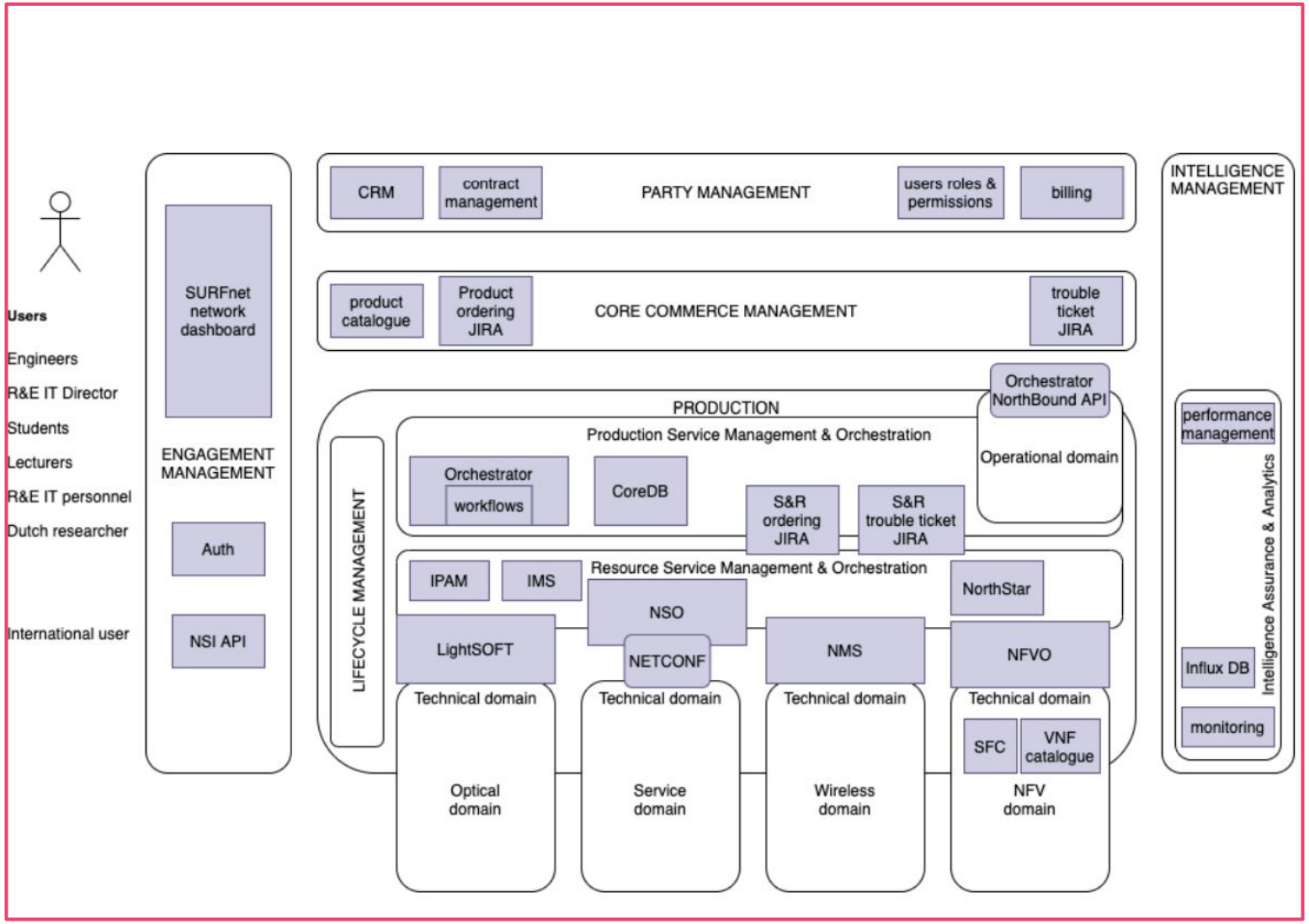
- Ansible deja de ser el punto central para el modelado e inventariado
  - Inventario y relaciones entre objetos: Workflow Orchestrator (WFO)
  - Variables individuales *(host vars)*: WFO → argumento a Ansible
  - Variables comunes *(group vars)*: roles en Ansible
- WFO como orquestador / *workflow engine* interactuando con:
  - Gestión IP y DNS: Infoblox
  - Inventario: LibreNMS, Resource Manager
  - Monitorización de dispositivos: LibreNMS, Kentik
  - Observación y anomalías: Kentik
  - Despliegue remoto de *playbooks* de Ansible: LSO (ansible-runner)
- Operaciones realizadas: ~80% de las de Jenkins (pero más mantenibles)



GÉANT Automation Platform (GAP): iteración #2 – WFO (1)



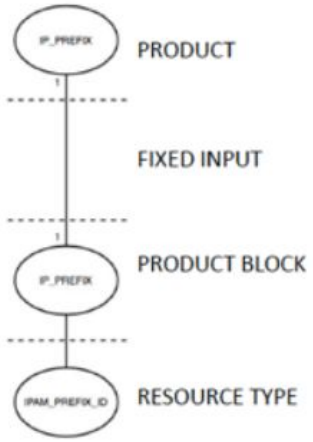
Modelos

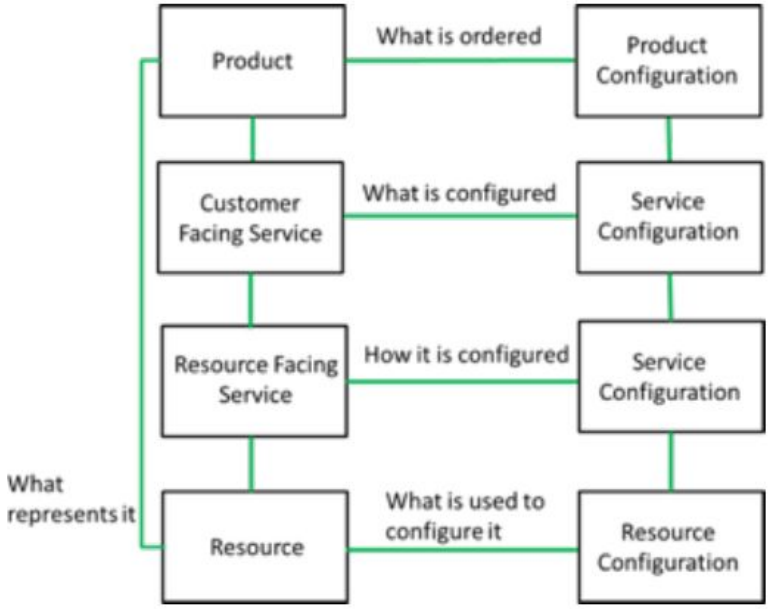


Arquitectura

Fuente: SURFnet OAV Architecture Analysis (GN4-3 whitepaper)

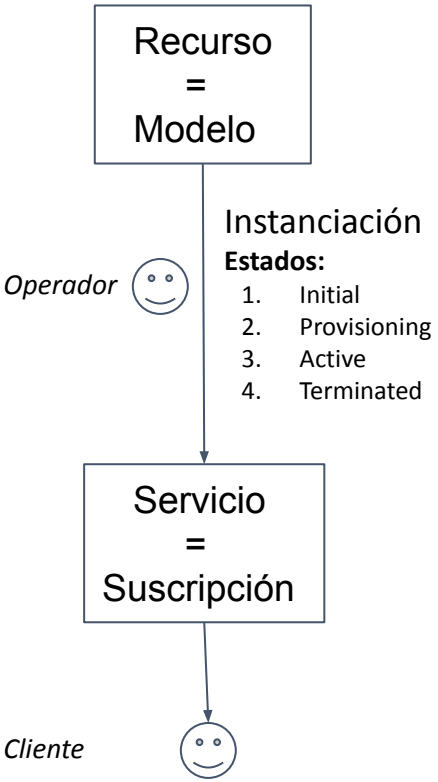
GÉANT Automation Platform (GAP): iteración #2 – WFO (2)






```
class IptrunkBlockInactive(ProductBlockModel,
                           lifecycle=[SubscriptionLifecycle.INITIAL],
                           product_block_name="IptrunkBlock"):
class DeviceBlockInactive(ProductBlockModel,
                          lifecycle=[SubscriptionLifecycle.INITIAL],
                          product_block_name="DeviceBlock"):
```

```
class IptrunkInactive(SubscriptionModel, is_base=True):
class DeviceInactive(SubscriptionModel, is_base=True):
```



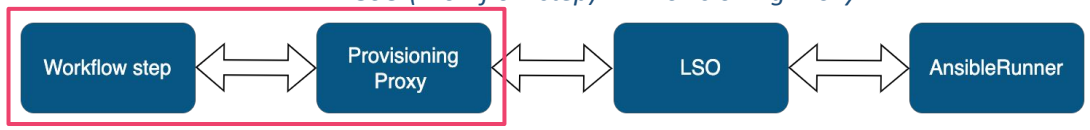
Fuente: SURFnet OAV Architecture Analysis (GN4-3 whitepaper)

GÉANT Automation Platform (GAP): iteración #2 – modelos (1)

```
@step('Provision device [FOR REAL]')
def provision_device_real(subscription: DeviceProvisioning,
                          process_id: UUIDstr) -> State:
    provisioning_proxy.provision_device(subscription, process_id,
False)

    return {'subscription': subscription}
...
            >> provision_device_real
            >> await_pp_results
            >> confirm_pp_results
```

GSO (Workflow step) → Provisioning Proxy




```
def provision_device(
        subscription: DeviceProvisioning,
        process_id: UUIDstr,
        dry_run: bool = True):

    parameters = {
        'dry_run': dry_run,
        'subscription': json.loads(json_dumps(subscription))
    }


    _send_request('device', parameters, process_id, CUDOperation.POST)
```

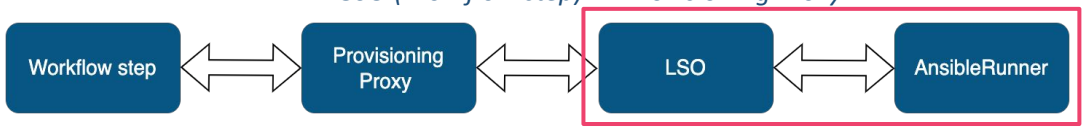
GSO (Workflow step) → Provisioning Proxy

GÉANT Automation Platform (GAP): iteración #2 – modelos (1)

```
@step('Provision device [FOR REAL]')
def provision_device_real(subscription: DeviceProvisioning,
                          process_id: UUIDstr) -> State:
    provisioning_proxy.provision_device(subscription, process_id,
False)

    return {'subscription': subscription}
...
            >> provision_device_real
            >> await_pp_results
            >> confirm_pp_results
```

GSO (Workflow step) → Provisioning Proxy




```
def provision_device(
        subscription: DeviceProvisioning,
        process_id: UUIDstr,
        dry_run: bool = True):

    parameters = {
        'dry_run': dry_run,
        'subscription': json.loads(json_dumps(subscription))
    }


    _send_request('device', parameters, process_id, CUDOperation.POST)
```

GSO (Workflow step) → Provisioning Proxy

```
@router.post('/')
async def provision_node(params: NodeProvisioningParams) \
        -> playbook.PlaybookLaunchResponse:
    extra_vars = {
        'wfo_device_json': params.subscription,
        'dry_run': str(params.dry_run),
        'verb': 'deploy',
        'commit_comment': 'Base config deployed with WFO/LSO &
Ansible'
    }


    if params.subscription['device_type'] == 'router':
        playbook_path = \
            os.path.join(config_params.ansible_playbooks_root_dir,
                         'base_config.yaml')
    else:
        raise ValueError(f'Cannot find playbook path for device type '
                         f"{params.subscription['device_type']}!!")


    return playbook.run_playbook(
        playbook_path=playbook_path,
        inventory=f"{params.subscription['device']['device_fqdn']}",
        extra_vars=extra_vars,
        callback=params.callback)
```

LSO → AnsibleRunner

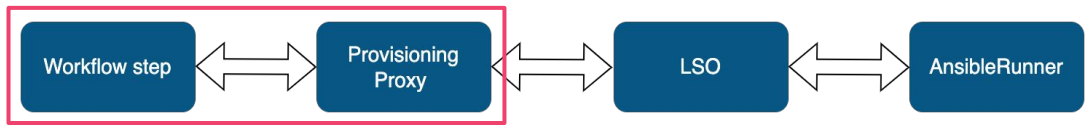
GÉANT Automation Platform (GAP): iteración #2 – modelos (2)

```python
class IptrunkBlock(IptrunkBlockProvisioning,
                   lifecycle=[SubscriptionLifecycle.ACTIVE]):
    geant_s_sid: str
    iptrunk_description: str
    iptrunk_type: IptrunkType
    iptrunk_speed: str
    iptrunk_isis_metric: int
    iptrunk_minimum_links: int
    iptrunk_ipv4_network: ipaddress.IPv4Network
    iptrunk_ipv6_network: ipaddress.IPv6Network

    iptrunk_sideA_node: DeviceBlock
    iptrunk_sideA_ae_iface: str
    iptrunk_sideA_ae_geant_a_sid: str
    iptrunk_sideA_ae_members: list[str] = Field(default_factory=list)
    iptrunk_sideA_ae_members_description: list[str] = \
            Field(default_factory=list)

    iptrunk_sideB_node: DeviceBlock
    iptrunk_sideB_ae_iface: str
    iptrunk_sideB_ae_geant_a_sid: str
    iptrunk_sideB_ae_members: list[str] = Field(default_factory=list)
    iptrunk_sideB_ae_members_description: list[str] = \
            Field(default_factory=list)
```

GSO (modelo)

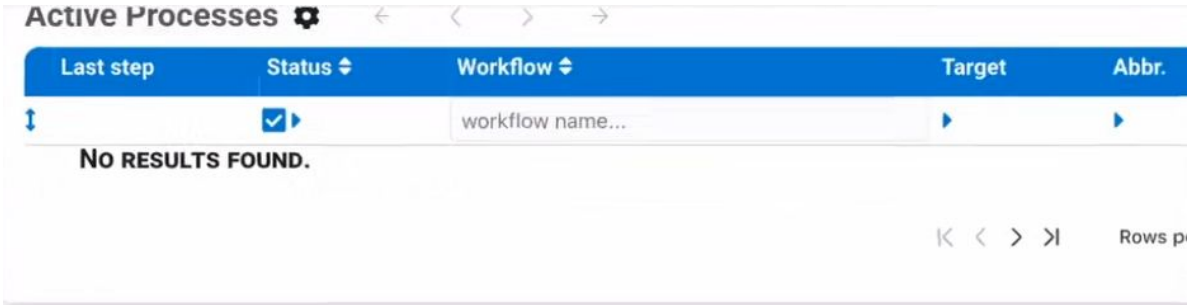


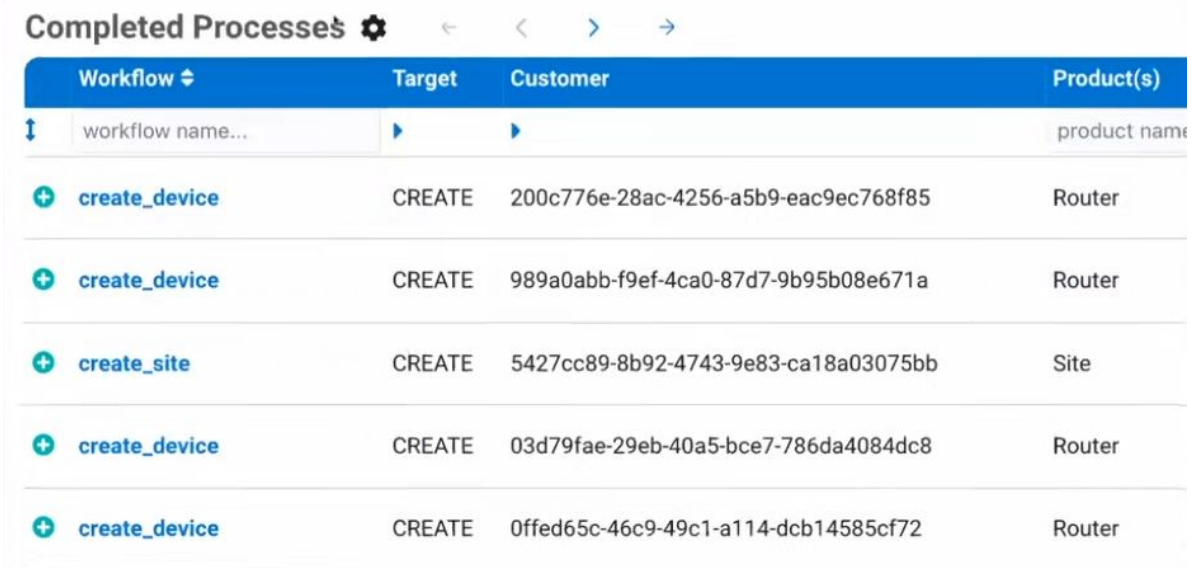

```python
@step('Initialize subscription')
def initialize_subscription(
        subscription: IptrunkInactive,
        geant_s_sid: str,
        iptrunk_type: IptrunkType,
        iptrunk_description: str,
        iptrunk_speed: str,
        iptrunk_minimum_links: int,
        iptrunk_sideA_node_id: str,
        iptrunk_sideA_ae_iface: str,
        iptrunk_sideA_ae_geant_a_sid: str,
        iptrunk_sideA_ae_members: list[str],
        iptrunk_sideA_ae_members_descriptions: list[str],
        iptrunk_sideB_node_id: str,
        iptrunk_sideB_ae_iface: str,
        iptrunk_sideB_ae_geant_a_sid: str,
        iptrunk_sideB_ae_members: list[str],
        iptrunk_sideB_ae_members_descriptions: list[str]
) -> State:
    subscription.iptrunk.geant_s_sid = geant_s_sid
    ...
    subscription.iptrunk.iptrunk_sideA_node = Device.from_subscription(
        iptrunk_sideA_node_id[0]).device
    ...
    subscription = IptrunkProvisioning.from_other_lifecycle(
        subscription, SubscriptionLifecycle.PROVISIONING
    )
    return {'subscription': subscription}
```

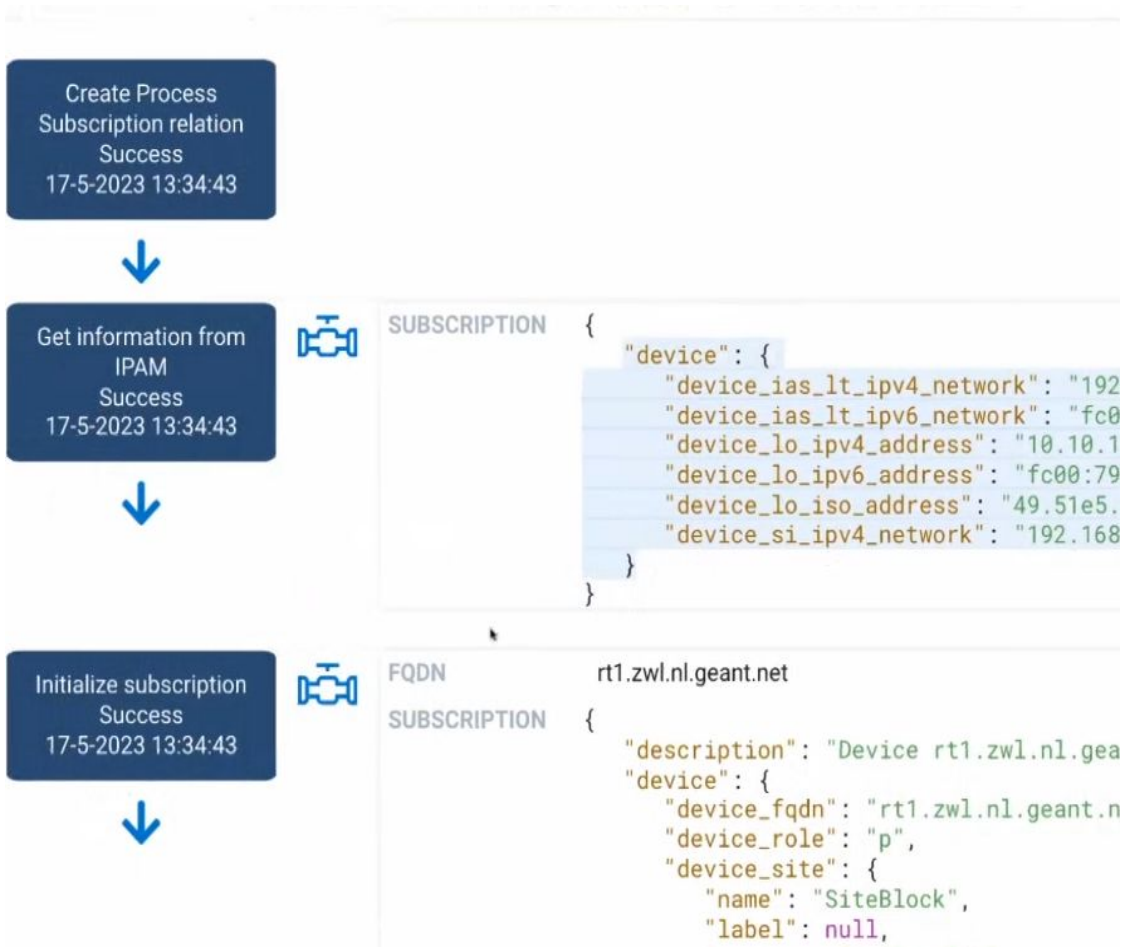
GSO (Workflow step) → Provisioning Proxy

GÉANT Automation Platform (GAP): iteración #2 – ejemplo

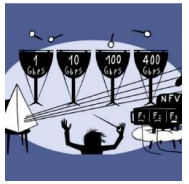
Workflow Orchestration (UI) como gestor de *workflows* + Ansible + Python *scripts*







GÉANT Automation Platform (GAP): iteración #2 – conclusiones 😀



Workflow Orchestrator (WFO) como orquestador de flujos

- Pros: integración con múltiples componentes, gestión *multi-tenant* y RBAC, mantenido por la comunidad NREN, despliegues en producción validados por SURFnet y ESnet
- Contras: despliegue inicial con recorrido por simplificar

Modelado mediante WFO y persistido en DB

- Pros: se pueden expresar mejor las relaciones complejas en el dominio de gestión de red
- Contras: curva de aprendizaje algo elevada (bloques de modelado), gestión de migraciones (DB)

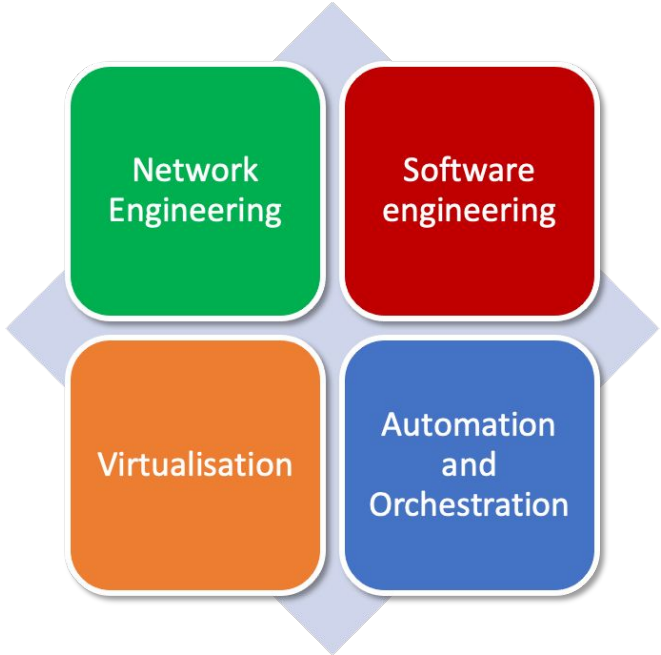
Operación

- Integración con otras herramientas como parte de la definición de *workflows* en WFO
- Identificación y evaluación de errores: WFO/GSO/LSO (centralizado), workflows (centralizado)

Equipo

GÉANT Orchestration & Automation

- 17 miembros, 9 organizaciones diferentes
- División de trabajo:
  - Red: modelado, definición de procesos
  - SW: diseño, desarrollo, pruebas
  - Adaptación y portabilidad de código a entornos virtuales
  - Despliegue: orquestación y automatización
- Colaboración con otros equipos y audiencias:
  - Documentación y diseminación
  - SURFnet: seminario WFO y acompañamiento



Material de referencia

Workflow Orchestrator (WFO)

- https://workfloworchestrator.org/
- https://workfloworchestrator.org/orchestrator-core/workshops/beginner/overview/
- https://github.com/workfloworchestrator

WFO en GÉANT

- https://gitlab.geant.org/goat
- Contacto con el equipo GOAT: <gn4-3-wp7-nat@lists.geant.org>

¿Preguntas?

/EOF
Gracias por vuestra atención

www.geant.org

Co-funded by
the European Union