



LF NETWORKING

THE  
LINUX  
FOUNDATION

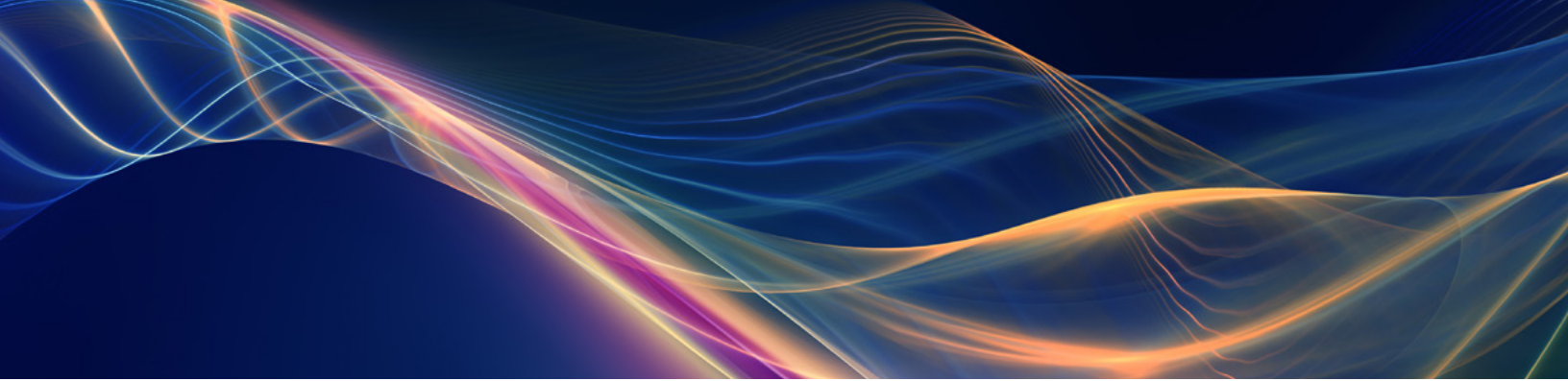
NEPHIO

# SPIFFE as a common glue for Large Scale Telco Deployments: A Nephio Rationale

Rahul Jadhav (AccuKnox), Nephio SIG-Security Chair

Prashant Mishra (AccuKnox)

A Linux Foundation Networking publication



## Contents

<b>Introduction .....</b>	<b>3</b>
<b>1. Terminology.....</b>	<b>4</b>
<b>2. Telco Deployments .....</b>	<b>5</b>
<b>3. Workload Identity vs User Identity.....</b>	<b>7</b>
<b>4. Why Nephio needs an Identity layer? .....</b>	<b>9</b>
<b>5. Why SPIFFE? .....</b>	<b>14</b>
<b>6. High level SPIFFE Reference Design for Nephio .....</b>	<b>16</b>
<b>7. Deciding the SPIFFE ID format.....</b>	<b>21</b>
<b>8. Next steps.....</b>	<b>21</b>
<b>9. SPIRE Limitations .....</b>	<b>21</b>
<b>10. Current state of SPIFFE integration in Nephio .....</b>	<b>22</b>
<b>11. References .....</b>	<b>22</b>
<b>12. Credits .....</b>	<b>23</b>

## Introduction

The majority traffic in any deployment is east-west traffic, i.e., inter-service traffic where applications talk to each other. Applications/Microservices typically have a fixed pattern of communication, for e.g., a web server will talk to a database server, a log server and take ingress traffic from a frontend server. In case of ORAN deployments, xApps can connect to Subscription Manager to listen to E2 nodes events. These fixed patterns of communications usually translate to a set of access control rules and setting up these access control/authorization rules requires one to “securely and uniquely identify” these applications aka workloads aka services (will be using these terms interchangeably). The [Principles of Least Privilege \(PoLP\)](#) which is the cornerstone for Zero Trust Security states that every workload must be able to access only the information and resources that are necessary for its legitimate operations.

Thus the workloads need to be identified at a granular level using a “unique and entire” set of attestable attributes so that the authorization frameworks have the flexibility to put in the access control rules as desired.



## Nephio Security Background

Nephio [SIG-Security was chartered in Oct 2023](#). The immediate action item as part of R3 release was to handle OpenSSF score improvement. But apart from that, the charter identified other security action items such as Holistic Secrets Management, Service Mesh, Network Security etc. Very soon it was realized that most of these advanced security action items cannot be handled without having a strong Identity layer especially given the distributed nature of Nephio’s operation. Thus the SIG-Security decided to take up the task of ensuring the right identity architecture and propose the design/architecture changes to the Nephio SIGs at large.

# 1. Terminology

Term	Details
SIG	Special Interest Group
SPIFFE	Secure Production Identity Framework For Everyone
SPIRE	SPIFFE Reference Implementation
ORAN	Open RAN
PoLP	Principles of Least Privilege
ZTA	Zero Trust Architecture
RAN	Radio Access Network
VAS	Value Added Services
FOCOM	Federated Open Cloud Orchestration & Management
IMS	Infrastructure Management System
IDP	IDentity Provider
SVID	SPIFFE Verifiable Identity Document
porch	Package Orchestration



## 2. Telco Deployments

Telco Deployments have evolved from:

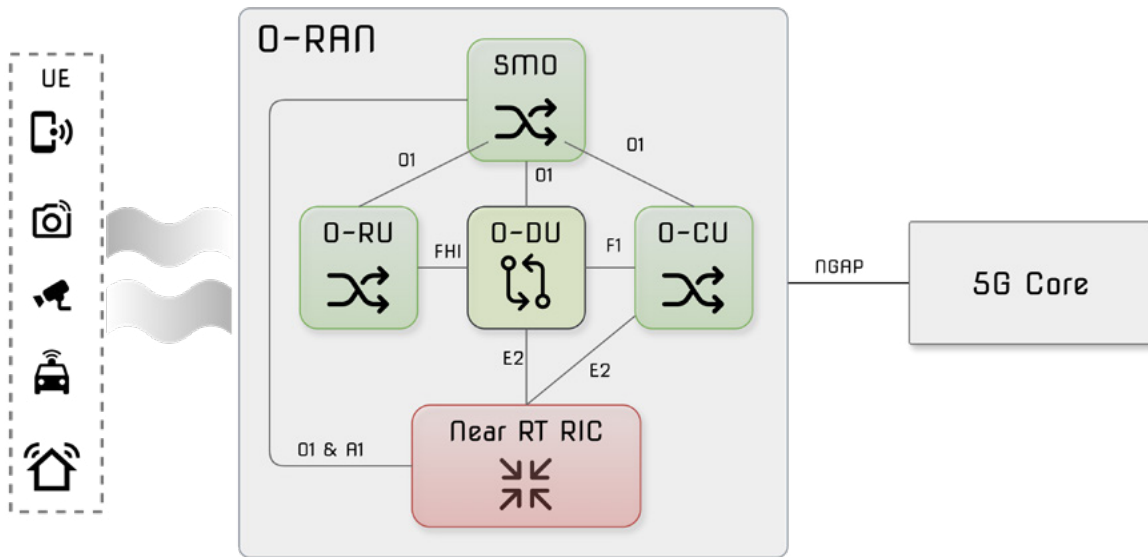
- SS7, that were only focussed on telephony services.
- SS7 over IP (aka SIGTRAN), that enabled native integration with IP networks and removed need for non-IP switches/routers.
- NGIN, that enabled VAS (Value Added Services) integrations with core telephony.
- Cloud Native Deployments. Disaggregation and democratization using cloud native deployments will help telcos adopt services beyond traditional telephony (voice and data).

One of the primary concern would be ensuring scalable authorization of services interconnect in such as dynamic, multi vendor, multi cloud deployments.

A strong Identity layer becomes a foundational concern for any Service-Centric Telco operator. If the services have to scale they need to trust each other in a secure way. If the trust is breached or compromised, then the Identity architecture decides the blast radius of the compromise.

### **ORAN context**

ORAN or (Open RAN) is a software-based transformation of RAN that allows operators to open the RAN interfaces and enables services to leverage the telemetry data in a highly scalable way. ORAN standardizes the interfaces such that the services can now be delivered in a vendor agnostic way. Any vendor can leverage the open interfaces and propose a service that improves telco operator's capability to provide advanced, improved services.



The Identity layer is at the core of this innovation since a secure and scalable Identity layer can enable secure onboarding and usage of such services.

ORAN disaggregates the RAN components and standardizes the interfaces between these components. The Identity of the components is the key to securely establishing the authorization between the components based on the shown connectivity graph. The components can be further broken down; for instance in near-RT RIC, there are subcomponents such as E2Term, E2Mgr, and xApps. These components essentially control the RAN and UE elements/groups and thus the authorization plays a central role in ensuring that unknown/unwanted components do not have access to such interfaces. Identity is a pre-requisite for any such secure authorization to happen.



### 3. Workload Identity vs User Identity


User Identity deals with identifying a user in a system and there is a human/person behind the Identity who can attest to the system by providing a secure key/password or any other bio print.

In case of workloads, a secure attestation becomes a challenge since this attestation should uniquely identify that workload alone without any manual intervention. The requirement for such attestations is that any other workload should not be able to spoof this attestation, even within the same operational domain.

#### Token based Access

Traditionally, there are two ways to provision an access to a service.

1. Using Identity, wherein a workload or user accesses the service by authentication of its Identity. The service enables authorization flow by enabling access to the service based on Identity.
2. Using Tokens, wherein a workload or a user is in possession of a token that is issued by the service that allows anyone with that token to access the service. Note that the Identity is not considered in such cases i.e., merely the possession of the token enables access to the service. Consider the case of GitHub Access Token which allows anyone holding that token to access the corresponding Github services.



Token based Access has following caveats:

1. Manual rotation of the token. For e.g., in case of Github, one can issue a token that is valid for a certain duration beyond which the token has to be rotated. The token might be used in more than one places and the user has to ensure that all these token are updated.
2. A token is associated with a fixed set of permissions. If there are any changes to the permissions spec, the token has to be updated to allow the updated permissions.

Consider the case where a token grants READ access to a FileService. There are two applications, App1 and App2 who needs this READ access and the user creates a single Token that enabled READ access to the FileService and then shares the token to App1 and App2. Any application in possession of the Token can now READ from the FileService. The FileService developer then gets a requirement to split the READ access into two parts READ-Metadata, READ-ALL enabling read of metadata-only, and metadata+content respectively of the files. Now lets assume that App1 needs READ-Metadata access only and App2 needs READ-ALL access. The issued tokens have to be updated to ensure appropriate access. Note that Token update usually is a more tedious process since all the impacted Applications have to be checked and updated.

With Identity based solution, this becomes an authorization problem, wherein the user/application identifies itself and then accesses the service. Thus any change in the permission spec can be dynamically applied. In this case, after the READ access is split between READ-Metadata and READ-ALL, the deployment changes the authz rule stating App1 needs READ-Metadata access and App2 needs READ-ALL. There are no changes required on the Applications since their Identity remains the same.





## 4. Why Nephio needs an Identity layer?

Nephio's mission is to deliver carrier-grade, open, Kubernetes based cloud native intent automation that simplifies the deployment and management of multi-vendor cloud infrastructure and network functions across large scale edge deployments.

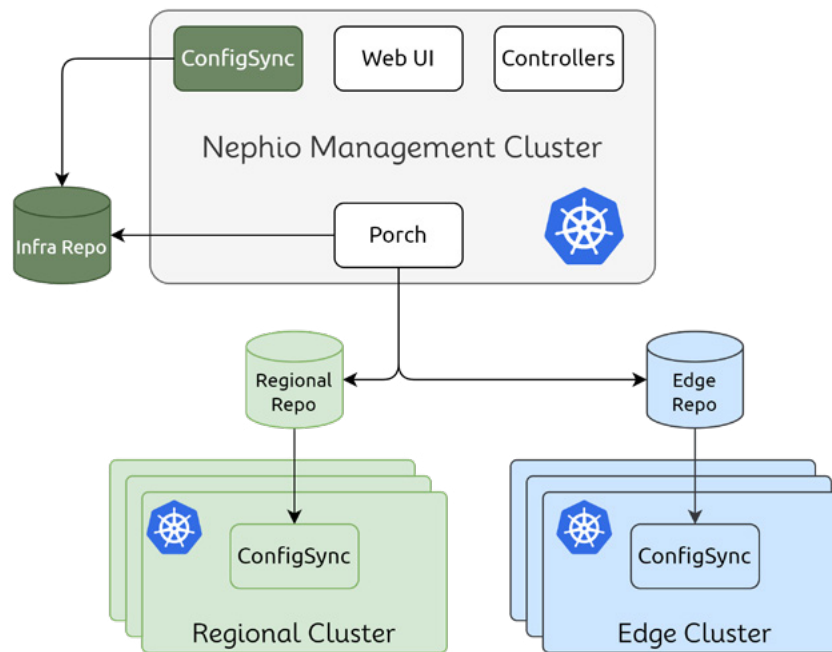
Workloads operating at such scales need to communicate to each other and secure authorization needs to be in place to handle this communication. Identity is a pre-requisite for any authorization solution to scale. Lets look at some of the examples today, why Nephio needs Identity.

**The problem Nephio wants to solve start only once we try to operate at scale. "Scale" here does not simply mean "large number of sites". It can be across many different dimensions: number of sites, number of services/workloads, size of the individual workloads, number of machine needed to operate the workloads, complexity of the org running the workloads, and other factors. The fact that our infrastructure, workloads, and workload configurations are all interconnected dramatically increases the difficulty in managing these architectures at scale.**

## Sharing resources between Management and Regional/Edge clusters

Nephio Management cluster is dedicated to manage the deployment and lifecycle of network functions that will be deployed on workload clusters. Workload cluster is where the actual network function workloads are deployed and running.

Requirement REQ\_MGMT\_CLUSTER\_COMM: Nephio Management Cluster currently



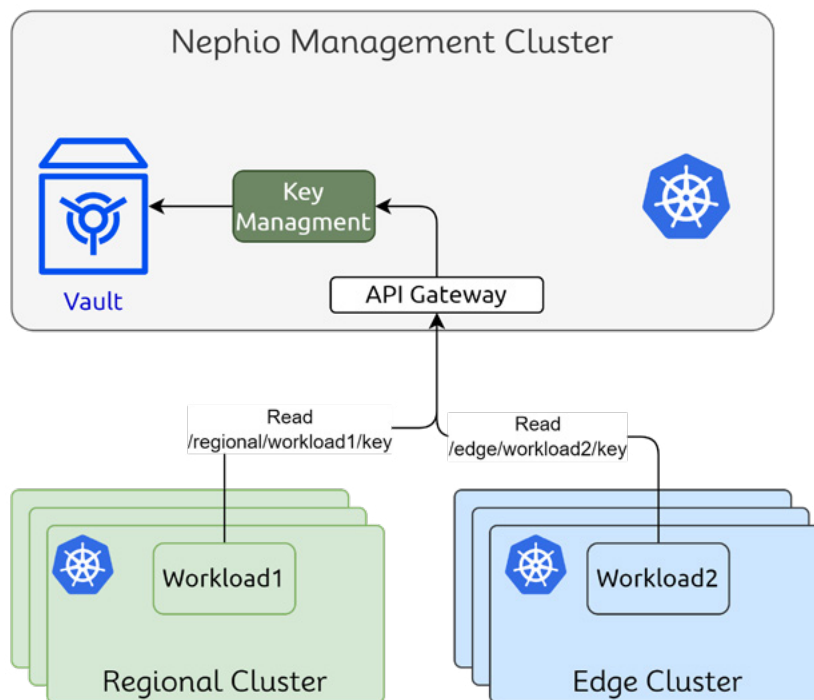
connects to the Regional/Edge cluster by creating a key as part of Cluster API operations. The use of this key allows any operations to be done at regional/edge cluster. Nephio Management cluster in the future might require somehow to create authorization policies such that only limited set of operations could be done through certain workloads from the management cluster.

Requirement REQ\_WORKLOAD\_CLUSTER\_COMM: Currently, Nephio regional/edge clusters do not communicate back to the Nephio Management cluster. However, there would be requirements to communicate back in the near future. In this case, it should be possible to authorize appropriate access for services on regional/edge clusters to Nephio management cluster resources.

## Secrets Management

Currently, if the management cluster has to provide any secrets access to the edge/regional cluster it has to create the secrets are created in the individual clusters. This has multiple issues:

- The secrets have to be synced across multiple clusters and the time duration to do this can be high.
- Causes secrets sprawl i.e., multiple clusters may end up having the same secret and thus the possibility of compromise increases manifold.
- Using k8s secrets may not be the best way for any sensitive secrets. Use of secrets managers is common, and the secrets manager can be issued with access policies for which the identity is a pre-requisite.



Requirement REQ\_SECRETS\_MGMT: Edge/Regional cluster workloads should be able to request access to certain keys from the management cluster. It should be possible to set the access control to read/write based on the workload type.

## Observability/Monitoring for Nephio subsystems

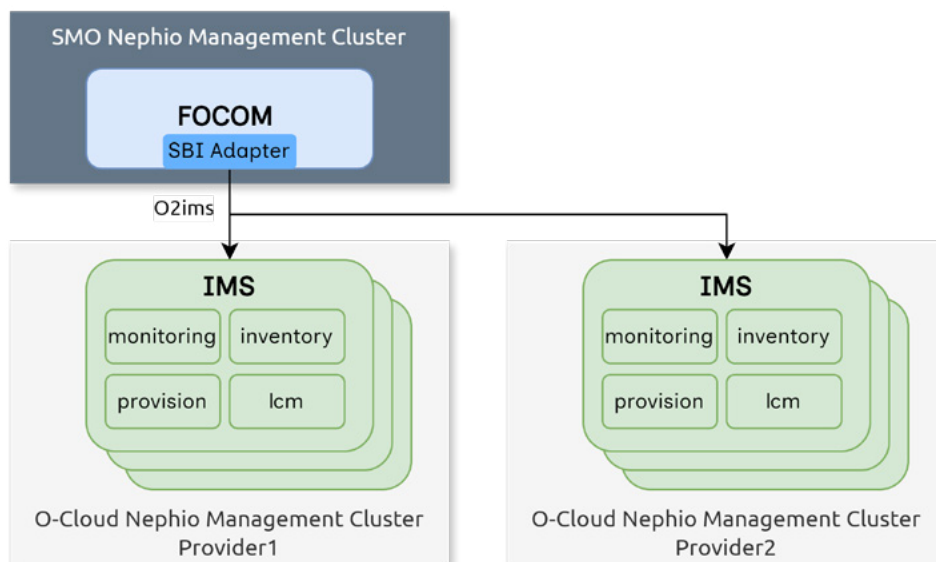
Nephio intends to use workload/cluster observability/monitoring solutions. This requires that the observability data be pushed to a common storage most likely on the management cluster. It is necessary to enable authorization of which edge/regional workloads should be allowed to push the observability data and at what points in the management cluster.

Requirement REQ\_OBS\_MON: Edge/Regional cluster should be able to call APIs on the management cluster that should allow only appropriate clusters to send observability data at the given API endpoints.

## ORAN FOCOM <> IMS Interaction

The primary role of the FOCOM and IMS services is to provide for the lifecycle management of the resources exposed by an O-Cloud.

The role of the FOCOM function is to provide federated orchestration and management across multiple O-Clouds using the O2ims interface.



Requirements REQ\_FOCOM\_IMS\_AUTHZ: FOCOM essentially operates as a client for IMS requesting IMS services to orchestrate the infrastructure. The IMS service could be provided by different providers and the FOCOM workload needs to identify itself for the IMS to grant access to the infrastructure operations.

Note that there are many other use-cases within the scope of ORAN based deployments but the FOCOM <> IMS interaction is the primary use-case of interest currently. Any Identity solution though should equally apply well to other cases.



## Inter-NF (Network Functions) comm across multiple clusters

Requirement REQ\_INTER\_WORKLOAD\_COMM: The identity layer should also enable authz for communication between workloads. For example, only UPF and SMF should be allowed to be connected on the N4 interface.

## Enabling Zero Trust Deployment

Requirement REQ\_ZERO\_TRUST: A deployment should consist of authorization rules that denies all and allows only specific communications between the set of endpoints.

## Why isn't Kubernetes Native Identity good enough?

For one simple reason; k8s identity is bound to a specific cluster. The namespace, service account, selector labels, scope is tied to the k8s cluster the workload belongs to. There needs to be an abstraction layer on top of these k8s constructs that expands the scope across the deployments constituting multiple clusters.

Further there is a need for Identity Federation i.e., Nephio entities will interact with third party providers to fulfill some of its tasks and thus would need Identity Federation. Kubernetes native Identity does not provide a common substrate to operate across heterogeneous deployments.



## 5. Why SPIFFE?

SPIFFE and SPIRE (reference implementation for SPIFFE) provides a uniform identity control plane across modern and heterogeneous infrastructure. In the case of Nephio, SPIFFE standard can help glue multiple workloads spread across multiple Nephio instantiated clusters and provide a consistent Identity standard. SPIFFE provides Identity federation across other IDPs thus able to operate in heterogeneous environments. **SPIRE** is a CNCF graduated project that provides a reference implementation for SPIFFE. Most importantly, the SPIRE provides a bunch of attestation plugins supported right out of the box that could be leveraged for Kubernetes, and bare-metal environments.

The most important differentiation provided by SPIRE is the attestation process by which it identifies the workload, before issuing it an identity document. This significantly improves security and reduces management complexity since no long-lived static tokens/credentials needs to be co-deployed with the workload itself.

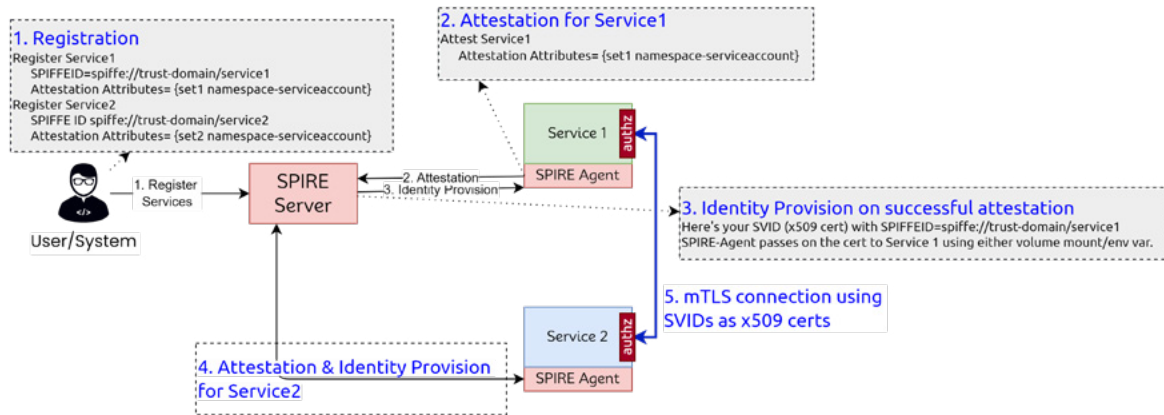
### SPIFFE 101

**SPIFFE** is a set of open-source specs for a framework capable of bootstrapping and issuing identity to services across heterogeneous environments and organizational boundaries. A short lived cryptographic identity document - called SVID (SPIFFE Verifiable Identity Document) plays a central role in these spec. [TODO add SVID to terms].

**SVID:** An SVID is a document with which a workload proves its identity to a resource. An SVID contains a single SPIFFE ID representing the identity of the service and the SVID is encoded in a cryptographically verifiable document; either as a X.509 certificate or a JWT token.

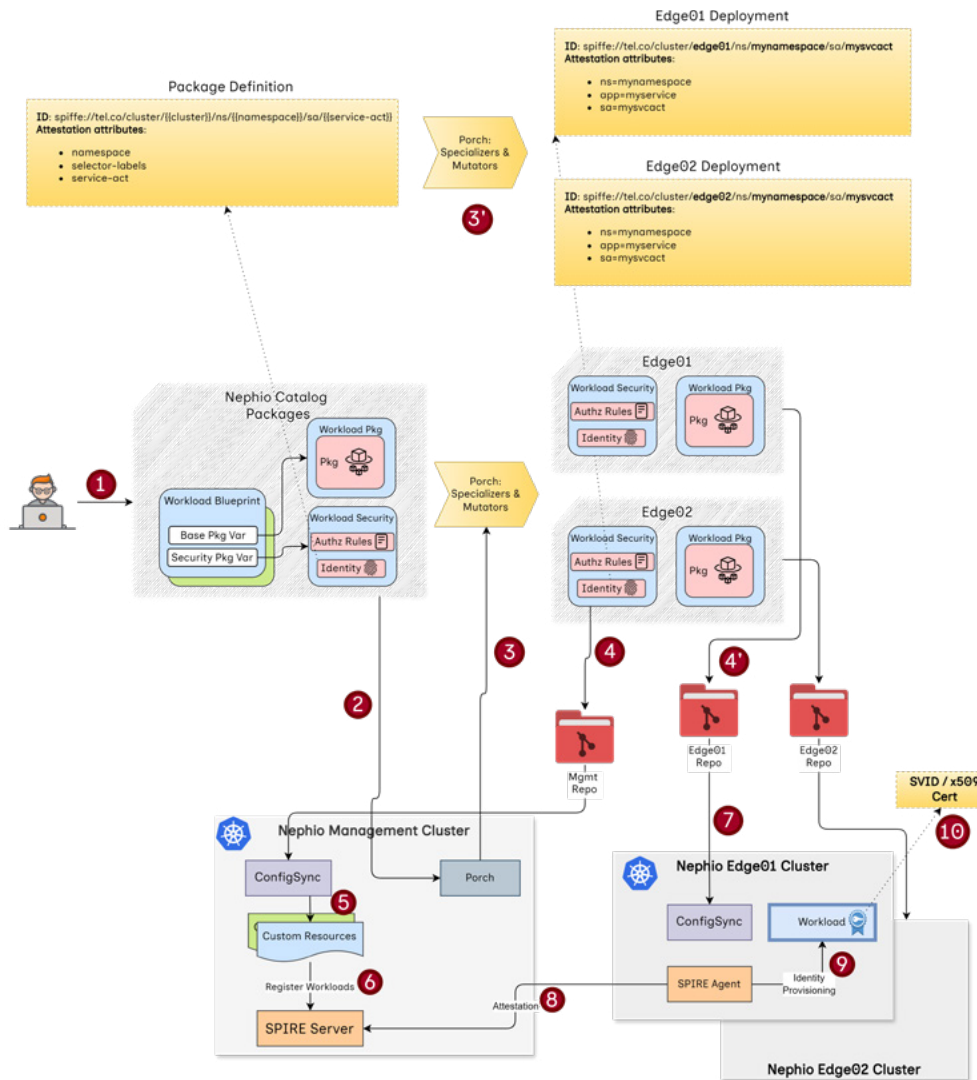
**SPIFFE ID:** A SPIFFE ID is a URI string of the format `spiffe://trust-domain/workload-identifier` that uniquely and specifically identifies a workload within a trust-domain.

## SPIFFE OPERATION PHASES



- 1. Workload Registration:** Workloads that will be using SPIFFE need to be registered with the SPIFFE/SPIRE server. The registration informs the server about the mapping between Identity (SPIFFE ID) and the corresponding set of attestation attributes. Server will use these attestation attributes to ascertain the claimed identity.
- 2. Service Attestation:** When the service starts, it needs to attest itself to the SPIRE server proving its identity.
- 3. Identity Provisioning:** Once the attestation is successful, the SPIRE server will issue the identity document (SVID) to the service. This SVID is either a x.509 certificate or a JWT that can subsequently be made use of in the data plane for authentication. The SPIFFE ID that is part of the SVID can be used for authorization purposes.
- 4. Data Plane Communication:** Data plane communication over transport protocols or tunnels require a credential or a token that can be used for security/authentication purpose. For e.g., in case of mTLS, both the client and server need to present their x.509 certificate for authentication purpose. In case of IP tunnels, x.509 certs will be needed to authenticate before establishing the tunnels. The SVID issued as part of identity provisioning procedure is put to use in all such cases.

## 6. High level SPIFFE Reference Design for Nephio



### SPIRE Control Plane Deployment

SPIRE Control Plane component has two aspects:

1. SPIRE Server
2. SPIRE Agents

In the case of Nephio, SPIRE server will be deployed on the Nephio Management Cluster and the SPIRE Agents would be deployed on all the clusters including management and workload clusters. SPIRE Agents are needed on all the clusters that has workloads that needs to attest to the SPIRE control plane server.



## Lifecycle of SPIFFE Workload Identity in context to Nephio

Nephio uses porch (Package Orchestration) toolkit that allows overall lifecycle of packages. These packages could be thought of as a Blueprint that encompasses both the workload definition itself and all the associated state such as Identity, Authz rules, etc.

The flow shown in the above diagram explains how Nephio manages the lifecycle of the workload blueprint and how the Identity related aspects could be plugged in.

### 1. DEV SPECIFIES THE WORKLOAD BLUEPRINT

Typically, the developer specifies the workload blueprint that includes

1. the workload related packages
2. the workload deployment related aspects
3. For workload identity, the blueprint will also additionally contain the identity and corresponding authz rules
  - Identity contains the SPIFFE ID that should be allocated to the workload provided it can attest to the given attributes. The attributes are extensible, but to begin with Nephio will use, Workload Namespace, Workload Selector Labels, and Workload Service Account as the attributes for attestation.
  - (Future) Authz rules will include all the authorization policies that this workload needs to use to access different resources.

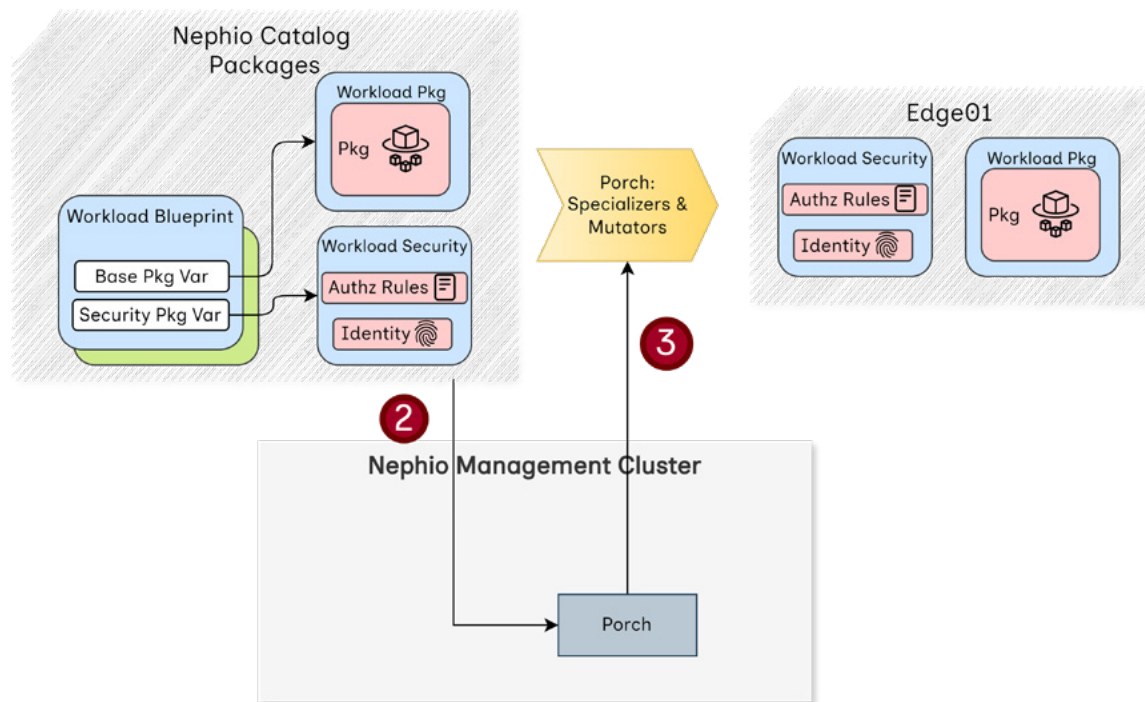
### 2. WORKLOAD BLUEPRINT AUTOMATION USING OF PORCH AND ASSOCIATED TOOLING

Porch/kpt/configsync are three tools that are heavily used for automation by Nephio.

- **kpt** automates Kubernetes configuration editing.
- **porch** provides a control plane for creating, modifying, updating, and deleting packages, and evaluating functions on package data. This enables operations on packaged resources similar to operations directly on the live state through the Kubernetes API.
- **configsync** synchronizes the generated resources on the target k8s clusters. Config Sync is built on top of git-sync and is used to automatically render manifests on the fly.

The workload blueprints are essentially packages that are managed by porch. Porch evaluates the blueprints and emits the final state in the repositories. ConfigSync picks up these updates from the repos and then synchronizes in the target k8s deployments.

### 3. USE OF PORCH SPECIALIZERS & MUTATORS

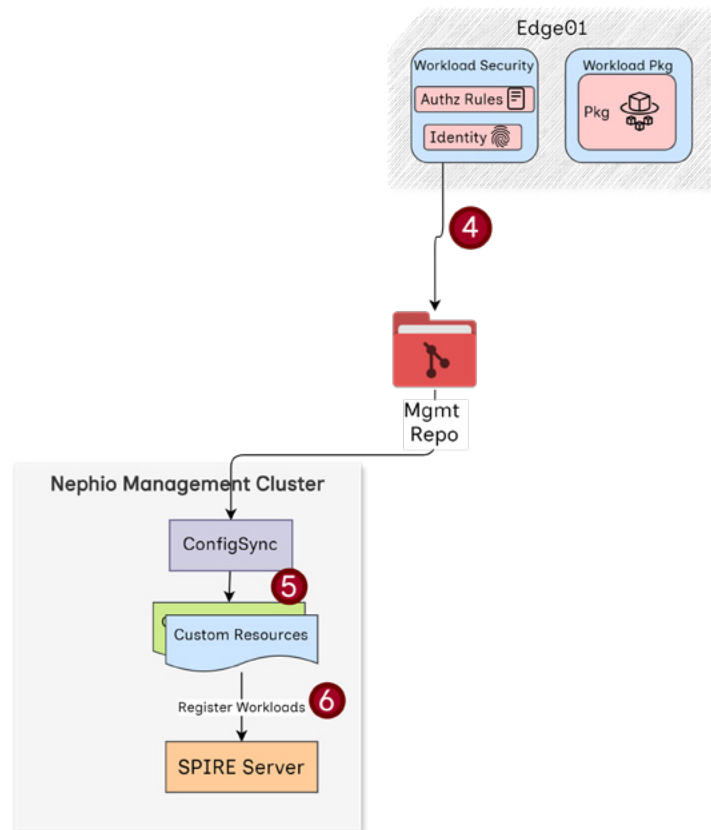


Porch enables use of [workflows](#) similar to those supported by kpt cli, but makes them available as a service.

Workflows enables Porch to use “specializers” and “mutators” that can enable in place updates of the resources customizing it for deployment specific needs.

In case of Identity resources, the specializers/mutators will adapt the k8s resources based on the given deployment. For e.g., when the Nephio decides to deploy the given workload in a specific namespace using a specific service account, the mutators would be used to replace the appropriate Identity parameters with the relevant details.

#### 4. SYNCHRONIZING THE RESOURCES IN THE TARGET K8S CLUSTERS



The process of porch mutation leads to workload identity resources to be pushed to the management repo, which are then picked up by the corresponding configsync in the Nephio Management Cluster.

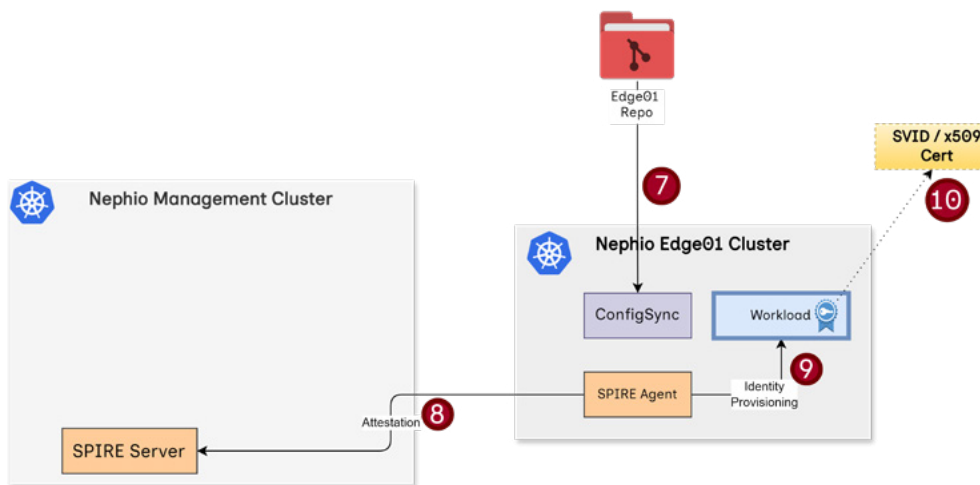
The workload identity resources provide identity registration details to the SPIRE server.

#### 5 & 6. REGISTRATION OF WORKLOAD IDENTITY WITH SPIRE SERVER

In the context of SPIFFE, the Workload Identity needs to be registered with the SPIRE server. The registration process tells the server what attestation to expect given the Identity. The configsync creates appropriate Identity resources in the k8s cluster which are then watched/picked up by the SPIRE server for registering the workloads.

## 7 & 8. WORKLOAD ATTESTATION

Once the workloads are deployed in the target k8s workload clusters, then the SPIRE agent will initiate a attestation procedure. Note that the SPIRE agent is deployed as a daemonset and it needs (read) access to all the workload resources using a ClusterRole.



The SPIRE Agent initiates a attestation procedure based on its configuration. In this case, we intend to use the [k8s built-in plugin](#) that enables attestation based on namespace and service account.

## 9 & 10. IDENTITY PROVISIONING

Using the attestation procedure, the SPIRE agent submits claims (namespace and service-account) on behalf of the workload to the SPIRE server. The SPIRE server independently investigates these claims using k8s control plane and if the claims are entirely validated, a Identity is provisioned.

The result of an Identity provisioning typically is the provisioning of a cryptographically secured token or credential. In this case, we assume that Nephio will use x.509 certs as the Identity token aka SVID. The x.509 certificate's SAN (Subject Alternative Name) field contains the Identity of the workload. Note that it is possible to provision multiple Identity documents to the same workload.

## 7. Deciding the SPIFFE ID format

SPIFFE IDs are a Uniform Resource Identifier (URI) which takes the following format: ***spiffe://trust domain/workload-identifier***. The ***workload identifier*** uniquely identifies a specific workload within a trust domain.

The end result of SPIFFE control plane operation is the provisioning of the Identity (ID) that uniquely identifies the workload. The SPIFFE ID is then eventually used for authorization purpose. For e.g., when the mTLS connection is established, the SPIFFE ID is validated at both ends.

Thus it is imperative that the SPIFFE ID is decided keeping in view the long term authorization needs.

## 8. Next steps...

- Handling Identity Federation.
- ORAN FOCOM <> IMS Interaction. FOCOM is a client to IMS services.
- Is there any change required in the existing apps to use SPIFFE?
- Should Nephio use x.509 or JWT as SVIDs?
- TODO: Explain Certificate Revocation strategy.

## 9. SPIRE Limitations

SPIRE is a reference implementation and has some limitations that one needs to be familiar with:

- SPIRE Agent is deployed as daemonset and thus won't work on k8s architecture not allowing to deploy daemonset (such as AWS Fargate, GKE AutoPilot).
  - Nephio Management and Workload clusters are completely managed by Nephio and we can assume Daemonsets to work.
- SPIRE helps you to provision Identity after attestation ... However, the authz procedures needs to be updated to use this Identity and requires additional work depending on where the authz is done.
- It is possible to handle federation with other providers, but each of it requires an integration of its own. (This cannot be called SPIRE limitation but one has to remember that SPIRE needs to work with ecosystem to handle Identity at scale).



## 10. Current state of SPIFFE integration in Nephio

Nephio SIG-Security has been focussing on getting the requirements, user-scenarios for Workload Identity. The efforts have led to multiple demos that were shown to the community.

- [14th May 2024: Initial Demo of Nephio-SPIRE Integration](#)
- [23rd July 2024: Second Demo of Nephio-SPIRE Integration](#)
- [13th Aug 2024: Integrated demo with Automated workload registration](#)
- [20th Aug 2024: Discussion on Nephio Hydration strategies for SPIFFE Identity](#)

## 11. References

- [Nephio Documentation](#)
- [SPIFFE Website](#)
- [SPIFFE Specification](#)
- [Solving the Bottom Turtle: SPIFFE eBook](#)
- [How to construct SPIFFE IDs?](#)
- [SPIFFE/SPIRE on Redhat OpenShift](#)
- [Kubernetes Workload Identity](#)
- [What are vRAN and Open RAN?](#)

## 12. Credits

Nephio SIG-Security (especially Shiv Bhagavatul, Wim Hendrickx, and Byung-Woo Jun for their continuous feedback/review)

Nephio SIG-Automation and SIG-NetArch members for providing their mind share.

This research was supported in part by NSF awards CNS-2112471, ITE-2226443 and ITE-2326882. Any opinions, findings, conclusions, or recommendations expressed are those of the authors and not necessarily of the NSF.