Updated NFDeployment model

This User Story proposes to update the current NFDeployment CR and its API to expose topology and status of workloads which are aligned with ORAN specifications and supporting various controller implementation models (centralized, distributed, generic, custom).

Feature Overview

For Nephio users it is important to understand the status of workloads during deployment, upgrade and termination as well as during normal operations. This includes the status of the underlying resources created by deployment controllers (e.g. Flux, Argo, ConfigSync and NF specific controllers etc.). Nephio has introduced the NF Deployment CR that can be used both for input data configuration as well as NF Deployment status exposure but there is currently no support in Nephio for reporting workload cluster local NF Deployment status centrally up to the central NFO Nephio management cluster.

In addition, the current NFDeployment model in Nephio lacks information about the cluster and namespace topology for the workload, which is important for ORAN topology, supporting use cases for service orchestration and assurance. On the other hand, it currently defines configuration input for capacity and networking that rather should be exposed as day-1 application configuration data for the whole Cloudified NF or alternatively be defined in a generic way through the parameterRefs attribute.

The proposed extensions seek to (a) provide additional topology information for the deployed workload (i.e. the cluster and namespace),(b) provide a reference implementation of the NFDeployment status with a centralized Flux controller(s), (c) propose a generic architecture for NF Deployment status exposure using a centralized GitOps Engine and NF specific deployment controllers and (d) remove the day-1 related configuration attributes from the NF Deployment CR.

The proposed model updates are described in Appendix A. In summary the proposed changes are: -

- Extend NFDeployment *status* with the following:
  - *deployedCluster* and *deployedNamespace*
    - *deployedCluster* would refer to WorkloadCluster CR in Nephio
  - *nfoNfDeploymentStatus*
    - Overall status of the workload
  - *localNfDeploymentStatus*
    - *Generic status information reported from a NF specific deployment controller*
  - *gitOpsSpecificStatus containing fluxConditions*
    - Flux resource (Kustomize, HelmRelease) conditions
- Extend NFDeployment *metadata* with finalizer(s) for workloads

        - Finalizers keep the NFDeployment object during the termination process, so that users can monitor the status of the termination process via the NFDeployment object
    - Remove the *capacity*, *interfaces* and *networkInstances* items from the NFDeployment *spec* part

Stakeholders

- Nephio NBI consumers primarily Network Operators

Design and architectural considerations

- Backward compatibility with previous NFDeployment model
- Can be extended to support other controller types in the future (e.g. Argo)
- Synchronization of NFDeployment status will be done via a new NFO NFDeployment Controller (will be described in separate User Story)
- The generic architecture for NFDeployment status exposure is described with a centralized GitOps engine but would apply also for a distributed GitOps engine with no additional changes in the NFDeployment model

Prerequisites

- Depends on the current NFDeployment model of Nephio

Requirements

- Core functional requirements:
    - Updated NFDeployment model and API with the new parameters.

Acceptance criteria (Definition of done)

- List of deliverables:
    - Updated NFDeployment API in the Topology and Networking API Specifications
    - Updated NFDeployment CRD
    - Updated NFDeployment API implementation specification
- List of outcomes
    - Extended NFDeployment model

Priority

- High

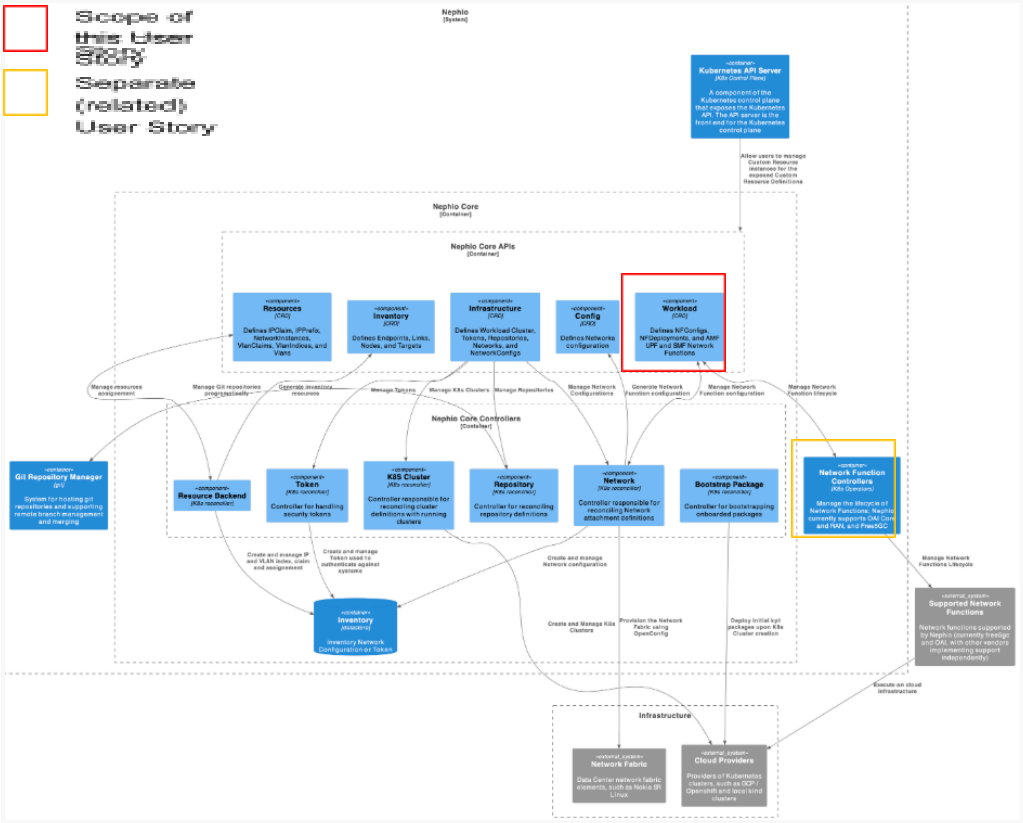
What's not in scope

- Changes to any other Nephio Topology and Networking APIs
- Updates for other GitOps engine specific conditions (e.g. Argo CD)
- Centralized Flux GitOps engine implementation (separate User Story)
- NFO NF Deployment Controller implementation for synchronization with Flux resources (separate User Story)
- Multi-cluster proxy implementation (separate User Story)

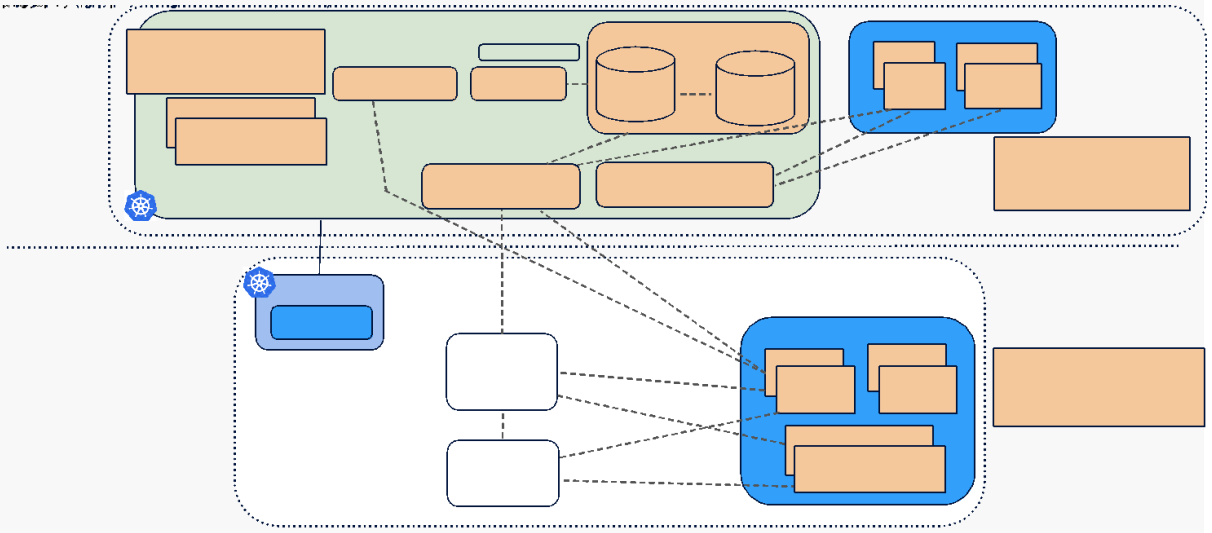
References

- Nephio Topology and Networking API Specifications
- NFDeployment CRD
- NFDeployment API specification
- Using Finalizers to Control Deletion (Kubernetes Blog)
- Flux Kustomize Health checks
- kstatus

Component Architecture



Scenario 1: Centralized GitOps Engine with NF specific deployment controller



When the NF Deployment instance resources are created by a Nephio compliant NF specific deployment controller running in the workload cluster, the deployment will be triggered by the creation of a Nephio NF Deployment CR instance and referred NF Config CR instances created locally in the workload cluster. Both the NF specific deployment controller as well as each NF Deployment instance specific local NF Deployment and NF Config CRs will be deployed by the “Centralized GitOps Engine”

running in the Nephio Management cluster that realizes the SMO NFO. Then the NF specific deployment controller will create the required native K8s resources based on the local NF Deployment CR. Once the required resources have been successfully deployed, the NF specific deployment controller shall report the status of the deployment in the local NF Deployment CR in two parts:

- <kstatus compliant status/conditions>
  Contains generic/common status information reported in the same way by each NF specific deployment controller. Proposal is that the reported status shall be according to the "kstatus" definition
- <application specific status/conditions>
  Contains any additional/extended status information reported that is unique for a certain NF specific deployment controller

The Centralized GitOps Engine will be able to read up the kstatus compliant information from the local NF Deployment CR and store it as extended information in the GitOps Engine specific CR. For instance with Flux as the GitOps Engine, this can be defined as health checks in the Kustomization CR and will be reported back as e.g. "Current" (status OK), "Failed" or "In Progress" in the same Kustomization CR. The Kustomize CR will also store any Flux GitOps Engine specific status information related to the reconciliation status of the local NF Deployment and NF Config CR instances.

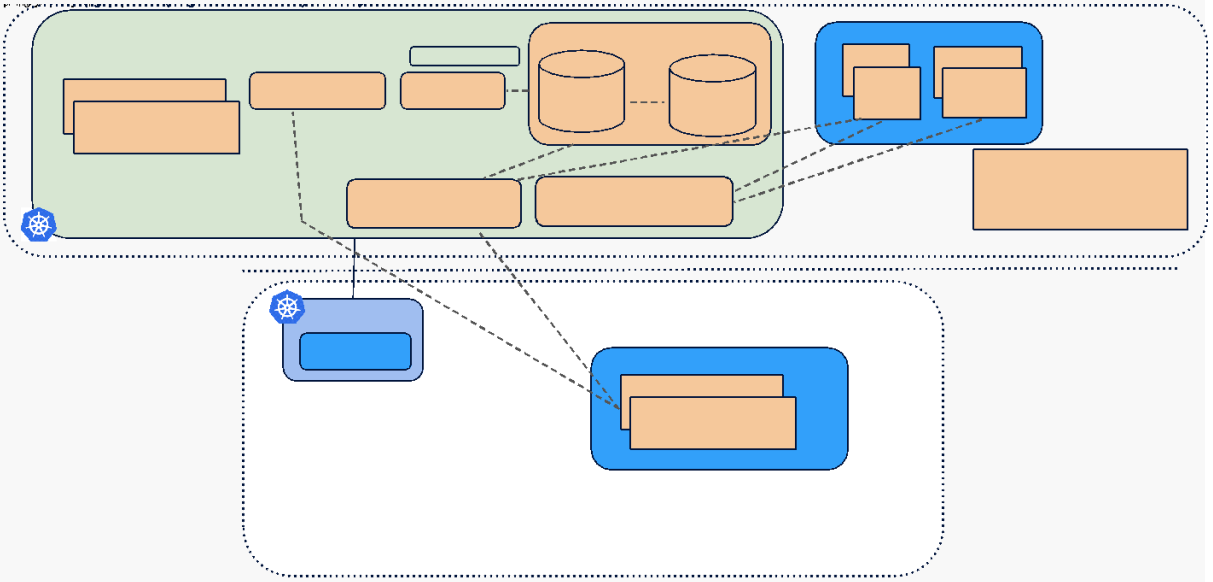
On the SMO NFO level the NF Deployment status shall be exposed in the NFO Nephio Management cluster in an implementation agnostic way. To do this, an "NFO NF Deployment Controller" deployed in the NFO Nephio Management cluster will read the information from the GitOps Engine specific CR and store it in a "NFO NF Deployment CR". There three types of information will be exposed:

- nfoNfDeploymentStatus
  A logical aggregation of the two statuses below, to provide a first-glance health reporting of the NF Deployment
- localNfDeploymentStatus
  A copy of the generic/common kstatus status information reported from the local NF Deployment CR
- gitOpsSpecificStatus
  GitOps Engine specific status information copied from e.g. the Flux Kustomize status

The application specific status/conditions reported by the NF specific deployment controller in the local NF Deployment CR will not be exposed through the NFO NF Deployment CR due to that it may contain any NF Deployment specific status information (e.g. more detailed status than the kstatus Current/Failed/InProgress status), and thus it cannot be queried in a generic way by the Centralized GitOps Engine. Instead the proposal is to support a "Multi-cluster proxy" component in the NFO Nephio Management cluster that can query the content directly from the local NF Deployment CR in the workload cluster. The Multi-cluster proxy would also be able to query and expose the status for any native K8s resource as well. During normal working conditions it would only be necessary to access the information exposed centrally in the NFO NF

Deployment CR, but in a failure scenario the Multi-cluster proxy would provide low level detailed status information about the local NF Deployment instance that could be useful for trouble-shooting. The Multi-cluster proxy would need to provide applicable authorization control to limit the access to the workload cluster information to approved users.

Scenario 2: Centralized GitOps Engine without NF specific deployment controller



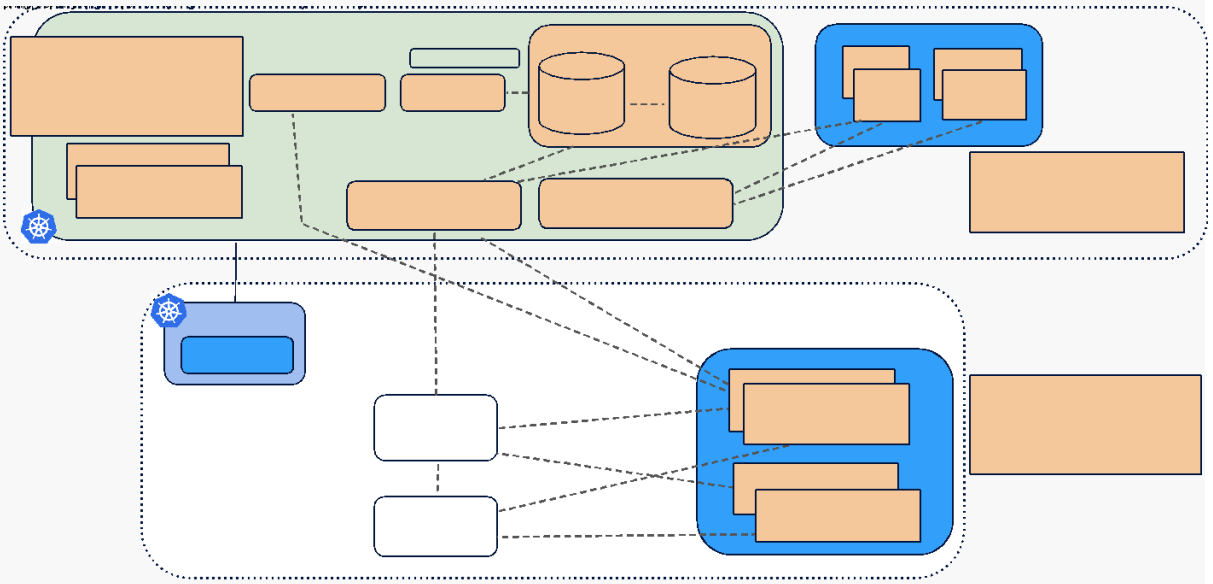
When the native K8s resources required for the NF Deployment instance are directly reconciled by the Centralized GitOps Engine there will be no local controller or local NF Deployment CR in the workload cluster. Instead the Centralized GitOps Engine will monitor the individual native K8s resources (e.g. Pod Deployments, StatefulSets etc.) and report GitOps Engine specific status information in the GitOps Engine specific CR, e.g. Flux kustomize and helmrelease CR status information.

In the same way as for scenario 1 there is a need to expose the aggregated NF Deployment status centrally in the NFO Nephio Management cluster in an implementation agnostic way. This will be managed by the “NFO Generic NF Deployment Controller” deployed in the NFO Nephio Management cluster. It will read the information from the GitOps Engine specific CR and store it in the “NFO NF Deployment CR” in a similar way as for Scenario 1. In this scenario two types of information will be exposed:

- nfoNfDeploymentStatus
  A logical aggregation of the gitOpsSpecificStatus information, to provide a first-glance health reporting of the NF Deployment
- gitOpsSpecificStatus
  GitOps Engine specific status information

The trouble-shooting support through the Multi-cluster proxy is still applicable but in this case it would only be used to query the status for the native K8s resources in the workload cluster.

Scenario 3:



This is a special case of scenario 2 where the Centrailzed GitOps Engine deploys both native K8s resources but also a proprietary NF specific deployment controller and corresponding proprietary Custom Resources to be consumed by this controller. If the proprietary NF specific deployment controller CR supports the kstatus reporting the Centralized GitOps Engine can use the same health check reporting as described in scenario 1. If kstatus is not supported the alternative when using a Flux based Centralized GitOps Engine would be to define so called “health check expressions” in the Kustomization CR.

The exposure of the NFO NF Deployment status would follow the same pattern as for scenario 1 and 2, with the NFO NF Deployment Controller monitoring the GitOps specific CR and writing the aggregated status into the NFO NF Deployment CR.

For trouble shooting the multi-cluster proxy would expose the proprietary NF specific deployment controller CR as well as local K8s resource CRs.

Needed software development.

- Updated NFDeployment CRD Updated NFDeployment API
- Centralized Flux GitOps Engine (separate User Story)
- NFO NFDeployment Controller (separate User story)
- Multi-cluster proxy (separate user story)

API design

- Update the NFDeployment CR and API, step to version 2

E2E test environment

- Full e2e testing requires controller implementation (separate user story)

Appendix A – Updated +)

This table outlines the proposed updates to the NFDeployment model – proposed extensions in green and removed items in red.

Item	Type	Description
apiVersion:	string	workload.nephio.org/v1alpha2
kind:	string	“NFDeployment”
metadata:	object	
name	string	Name of NFDeployment
finalizers:	object	Finalizers of NFDeployment Finalizers keep the NFDeployment object during the termination process, so that users can monitor the status of the termination process via the NFDeployment object
spec:	object	Spec is optional [5] and not used for functionality described in current PSS. NFDeploymentSpec defines the characteristics of a deployment of a network function
selector:	object	Label selector for referencing helmrelease and kustomization CRs
capacity:	*object*	*Defines the capacity characteristics of the NF deployment*
maxDownlinkthroughput	*string*	*Defines the max downlink dataplane throughput*
maxNFConnections	*integer*	*Defines the max NF(s) that can be connected to this NF/device*
maxSessions	*integer*	*Defines the max sessions of the control plane expressed in unit of 1000s*
maxSubscribers	*integer*	*Defines the max subscribers expressed in unit of 1000s*
maxUplinkThroughput	*integer*	*Defines the max uplink dataplane throughput*

interfaces:	*Array of objects*	*Interfaces defines the interfaces associated with the NF deployment*
ipv4:	*object*	*Defines the ipv4 configuration of the interface*
address	*string*	*Defines the IPv4 address and prefix length in CIDR notation [IP prefix, range IPv4 with host bits]*
gateway	*string*	*Defines the IPv4 address associated to the interface as a gateway*
ipv6:	*object*	*Defines the ipv6 configuration of the interface*
address	*string*	*Defines the IPv6 address and prefix length in CIDR notation [IP prefix, range IPv6 with host bits]*
gateway	*string*	*Defines the IPv6 address associated to the interface as a gateway*
name	*string*	*Required.* *Defines the name of the interface*
vlanid	*integer*	*Defines the specific vlan id associated on this interface*
networkInstances:	*Array of objects*	*Defines the network instances associated with the NF deployment*
interfaces	*Array of string*	*Defines the interfaces associated with the network instance*
name	*string*	*Required.* *Defines the name of the network instance*
bgp:	*object*	*Defines the BGP configuration associated with the network instance*
routerID	*string*	*Defines the router ID of the bgp process*
autonomousSystem	*integer*	*Required.*

		Defines the AS number of the bgp process
bgpNeigbors:	*Array of objects*	*Required.* *Defines the configuration of the BGP neighbor*
address	*string*	*Required.* *Defines the IPv4 or IPv6 address of the BGP neighbor*
name	*string*	*BGP interface name, MUST match the one use in InterfaceConfig*
peerAS	*integer*	*Required.* *Defines the AS number of the bgp peer*
dataNetworks:	*Array of objects*	*Defines the data networks assocated with the network instance*
name	*string*	*Defines the name of the data network*
pool:	*Array of objects*	*Defines the list of address pools associated with the data network*
prefix	*string*	*Required.* *Defines the ip cidr in prefix notation. It is defined as a subnet*
peers:	*Array of objects*	*Defines the peer configuration associated with the network instance*
name	*string*	*Defines the name of the data network*
ipv4:	*object*	*Defines the ipv4 configuration of the interface*
address	*string*	*Defines the IPv4 address and prefix length in CIDR notation [IP prefix, range IPv4 with host bits]*
gateway	*string*	*Defines the IPv4 address associated to the interface as a gateway*

ipv6:	*object*	*Defines the ipv6 configuration of the interface*
address	*string*	*Defines the IPv6 address and prefix length in CIDR notation [IP prefix, range IPv6 with host bits]*
gateway	*string*	*Defines the IPv6 address associated to the interface as a gateway*
parametersRefs:	Array of objects	Defines additional KRM parameter references the NFDepends upon. Target source is kustomization and helmrelease
apiVersion	string	APIVersion of the target resources
kind	string	Kind of the target resources
name	string	Name of the target resource
provider	string	Defines which provider implement this NFDeployment
status:	object	NFDeployment status defines the observed state of NFDeployment
deployedNamespace	*string*	*Defines namespace where the NFDeployment instance is deployed*
deployedCluster	*string*	*Defines workload cluster where the NFDeployment instance is deployed*
nfoNfDeploymentStatus	*string*	*Defines aggregated NFO level status of NFDeployment instance (e.g., Deploying, Not Ready, Failed, Ready, Stalled, Terminating, Unknown)* Present only in the NFO NF Deployment CR.
localNfDeploymentStatus:	*Array of objects*	*Local status information reported by the NF specific deployment controller. Present both in the Local NF Deployment CR and NFO NF Deployment CR when the deployment is managed by a NF specific deployment controller.*
gitOpsSpecificStatus:	*Array of objects*	*Status information specific for a certain GitOps Engine implementation. At this time specified for Flux only.*

		Present only in the Central NF Deployment CR.
fluxConditions:	*Array of objects*	*Conditions define the current state of target resource*
apiVersion	*string*	*APIVersion of the target resources*
kind	*string*	*Kind of the target resources*
name	*string*	*Name of the target resource*
lastTransitionTime	*string*	*Required.* *Specifies the last time the condition transitioned from one status to another of target source*
message	*string*	*Required.* *Message is a human readable message indicating details about the transition that taken from target source*
reason	*string*	*Required.* *Contains a programmatic identifier indicating the reason for the condition's last transition of target source*
status	*string*	*Required.* *Status of the condition from target source*
type	*string*	*Required.* *type of condition from target source*
conditions:	object	Conditions define the current state of the NF deployment
lastTransitionTime	string	Required. Specifies the last time the condition transitioned from one status to another. This should be when the underlying condition changed. If that is not known, then using the time when the API field changed is acceptable.
message	string	Required.

		Message is a human readable message indicating details about the transition. This may be an empty string.
observedGeneration	integer	Represents the metadata.generation that the condition was set based upon. For instance, if metadata.generation is currently 12, but the .status.conditions[x].observedGeneration is 9, the condition is out of date with respect to the current state of the instance.
reason	string	Required. Contains a programmatic identifier indicating the reason for the condition's last transition. Producers of specific condition types may define expected values and meanings for this field, and whether the values are considered a guaranteed API. The value should be a CamelCase string.
status	string	Required. Status of the condition, one of True, False, Unknown.
type	string	Required. type of condition in CamelCase or in foo.example.com/CamelCase.