FaaS: Beryllium: Release Plan

Contents

Introduction

During the Beryllium code sprint we plan to define the Fabric as a service API and implement these APIs through a Fabric Manager module and a OVS based VXLAN fabric. Those APIs intend to become a common abstraction layer between the applications and underneath network and abstract away the complexity of traditonal southbound API such as CLI, netconf, OVSDB, Openflow etc... To achieve the goal (becoming a common abstractio layer), we will integrate with GBP project and deliver a GBP renderer to demonstrate its capability in Be release.

API

FaaS feature or bundle exposes the following services to its external consumers. The consumers could be applications outside of ODL or modules within ODL. **In Beryllium release**, those APIs are limited to be used **within ODL only**. As FaaS matures, those service will be exposed to be part of ODL northbound.

Status - tentative
As the project progress, the APIs below are subject to change or removed or not implemented for this release according to our integration work with GBP or any internal projects for this release. They are not for ODL northbound used just yet.

Management API

Management API defines housekeeping interfaces to abstract, provision fabric objects as well as its life cycle management from initiation to decommission.

A fabric is an abstraction of a well configured underlay physical network or a portion of a physical network, which internally consists of a list of network nodes,a topology formed by those nodes as well as well configured underlay network ready for further virutulization.

Usally Those nodes supports the same l2 or l3 data path virtualization technology, such as VLAN, TRILL or VXLAN and share a underlay control plane. The control plane could be existing protocol running distributedly on those nodes or centralized provided by the fabric object within the controller.

The management API includes the following functions.

- To compose a fabric object based on a set of physical nodes

```
Fabric fabric = fabricManager.composeFabric(type, name, topology, typeSpecificOptions)
```

- To query a Fabric's internal topology, physical port list, capability and resource

```
ResourceDescriptor rd = fabric.queryResource(fabricID);
```

- Add a node to an existing fabric object.

```
fabric.addNode(Node node)
```

- Remove a node from an existing fabric object

```
fabric.removeNode(Node node)
```

- Add a link to a fabric topology.

```
fabric.addLink(Link link)
```

- Remove a link from a fabric Topology.

```
fabric.removeLink(linkID)
```

- Decompose a Fabric

```
fm.decomposeFabric(fabricID)
```

Core API

Core API defines the core services provided by a fabric object which are used to describe a logical network according to users' requirements. It is implemented by Fabric object and includes operations for the following key FaaS objects.

- Logical Port

A logical port is a logical counterpart of a physical port and provides layer 2 access point to a logical switch or a logical router. Depends on the variety of mappings to a physical port and a set of physical ports, it could be one of the following types

1. A physical port + logical ID
2. Bundling port + logical id

A bundling port is a object representing a port bundling of a set of physical ports which provides load balancing and HA merits

- L3 Interface

L3 Interface represents a layer 3 access point to a logical router. It could be

1. Unicast L3 interface
2. Multicast L3 interface

Also it could be either public or private.

- Tunnel *

Tunnel object represents a physical link abstraction. it provides Layer 1 connectivity, packet comes in one end of the tunnel and goes out from the other end without changed.

- Logical Switch

Logical Switch is an Layer 2 connectivity abstraction. It supports broadcast, could bind to a subnet, a gateway as well as DHCP services.

- Logical Router

Logical Switch is an Layer 3 connectivity abstraction.

- ACL(Access Control List)

ACL provides stateless flow control. It consists of a set of ACLEntries, each entry includes a pair of Classifier and Action

Port API

- To create a logical Port .

```
LogicalPort lport = fabric.createLogicalPort(phyiscalportID, logicalID);
```

- To remove a logical Port .

```
fabric.removeLogicalPort(lportID);
```

- To query logical port 's stats.

```
PortStatistics stats = lport.getStatistics()
```

- To create a l3 Interface.

```
L3Interface l3if = fabric.createL3Interface(IPAddress, public | private)
```

- To bind a l3 interface to a logical port.

```
l3If.bindLP(lpID);
```

- To bind a l3 interface to a logical switch.

```
l3if.bindLSW(lsw);
```

- To query a l3 interface's stats.

```
L3Statistics stats = L3if.getStatistics()
```

Layer 2 API

- Create a Logical Switch

```
LogicalSwitch lsw = fabric.createLSW(name, resourceID, … )
```

- Remove a logical Switch

```
fabric.removeLSW(lswID)
```

- Attach/remove a logical port to a logical switch

```
lsw.attachLogicalPort(lp);
```

- Dettach a logical port from a logical switch

```
lsw.dettachLogicalPort(lp);
```

- Query LSW stats

```
LSWStats stats = lsw.getStatistics();
```

- Query LSW Configuration attributes such as port list etc…

```
lsw.getAttributes();
```

Layer 3 API

- To create a Logical Router object

```
LogicalRouter lr = fabric.createLR(name);
```

- To remove a Logical Router object

```
fabric.deleteLR(lr);
```

- To query a Logical Router's routing table and its interface list.

```
Route[]  rs = fabric.lr.getRoutes();
L3Interfaces[]  rs = fabric.lr.getInterfaces ();
```

- To attach/dettach L3Interfaces to a Logical Router

```
lr.attachInterface(l3Interface);
```

- To update Logical Router's Routing Table

```
lr.addRoute(lr, Route);
lr.deleteRoute(lr, Route);
```

- To Check a logical router's status

```
lr.ping()
lr.traceroute()
```

Tunnel API

- Create a tunnel between two logical ports

```
Tunnel tunnel = fabric.createTunel(logicalPortPeer,  tunnelID, tunnel specific options);
```

- Remove a tunnel

```
fabric.removeTunel(tunnelID);
```

- Query a tunnel attributes

```
tunnel.getAttributes();
```

- Check a tunnel status

```
tunnel.ping()
tunnel.traceroute()
```

ACL API

ACL APIs allows applications to enforce stateless ACL control over logical ports and layer 3 interfaces. Its functionality includes

- add an ACL entry on a logical Port or a L3 Interface

```
ACLEntry entry = logicalPort.addACL(classifier, action, location);
ACLEntry entry = L3Interface.addACL(classifier, action, location);
```

- Delete an ACL entry on logical Port or L3 Interface

```
logicalPort.deleteACL(entryID)
l3Interface.deleteACL(entryID)
```

- Query ACLs on a logical Port or a L3 Interface

```
logicalPort.getACL()
l3Interface.getACL()
```

Resource Management API

Resource Managment API aims to support multi-tenancy. It does so by allocating and managing network resource within a Fabric including topology restrictions, ports, logical network id allocation and bandwidth into smaller fabric objects. The functionality includes

- To slice a fabric into smaller/children fabrics based on resource constraints

```
Fabric smallFabric = resourceManager.sliceFabric(fabric, resourceConstraint);
```

- To remove child fabric

```
resoureManager.removeSlice(smallFabricID);
```

- To define a tenant physical network based on a set of fabrics assigned to the tenant

```
resourceManager.assign(tenantID, Fabric[]);
```

- To query a tenant physical resource

```
TopologyGetTopologyByTenantID(tenantID)  // a fabric based abstacted topology object.
```

Release Deliverables

Name	Description
Fabric Manager	Fabric CRUD, network resource management, fabric based topology build as well as Rendering tenant logical network into services provided by fabrics
OVS Fabric	a fabric implementation based on OVS/VXLAN encapsulation

Release Milestones

Milestone	Offset 2 Date	Deliverables
M1	8/6/2015	<table><tr><th>Name</th><th>Status</th><th>Description</th></tr><tr><td>Intent to participate</td><td>Done</td><td>Intent to participate in Beryllium Simultaneous Release</td></tr><tr><td>Candidate Release Plan</td><td>Done</td><td>Candidate Release Plan</td></tr></table>
M2	9/3/2015	<table><tr><th>Name</th><th>Status</th><th>Description</th></tr><tr><td>Release Plan</td><td>Done</td><td>Final Release Plan</td></tr><tr><td>Project Checklist</td><td>Done</td><td>Project Checklist completed</td></tr><tr><td>FaaS Service definition</td><td>Done</td><td>The services provided by FaaS</td></tr><tr><td>Project acknowledged</td><td>Done</td><td>Project acknowledged from all projects that it depends on.</td></tr></table>

<table>
<tr><td>M3</td><td>10/15/2015</td><td>
<table>
<tr><th>Name</th><th>Status</th><th>Description</th></tr>
<tr><td>Feature Freeze</td><td>Done</td><td>Final list of externally consumable APIs defined and documented</td></tr>
<tr><td>Karaf Features defined</td><td>Done</td><td>Karaf Features defined</td></tr>
<tr><td>Integration & System Test</td><td>Done</td><td>Simple system test on a karaf distribution with the project's recommended features installed</td></tr>
</table>
</td></tr>
<tr><td>M4</td><td>11/12/2015</td><td>
<table>
<tr><th>Name</th><th>Status</th><th>Description</th></tr>
<tr><td>API Freeze</td><td></td><td>Fabric Manager/OVS fabric code complete</td></tr>
<tr><td>Datastore update</td><td></td><td>logical network to faas mapping stored</td></tr>
<tr><td>Integration & System Test</td><td></td><td>Run a simple system test on a karaf distribution with the project's recommended features installed on Code Merge</td></tr>
</table>
</td></tr>
<tr><td>M5</td><td>12/17/2015</td><td>
<table>
<tr><th>Name</th><th>Status</th><th>Description</th></tr>
<tr><td>Code Freeze</td><td></td><td>Code implementation to be completed working with GBP renderer.</td></tr>
<tr><td>Documentation</td><td></td><td>Update wiki documentation to reflect changes/new features.</td></tr>
<tr><td>Feature Test</td><td></td><td>Run system test for a feature.</td></tr>
</table>
</td></tr>
<tr><td>RC0</td><td>TBD</td><td>
<table>
<tr><th>Name</th><th>Status</th><th>Description</th></tr>
<tr><td>Deliverable Name</td><td></td><td>Deliverable Description</td></tr>
</table>
</td></tr>
<tr><td>RC1</td><td>TBD</td><td>
<table>
<tr><th>Name</th><th>Status</th><th>Description</th></tr>
<tr><td>Deliverable Name</td><td></td><td>Deliverable Description</td></tr>
</table>
</td></tr>
<tr><td>RC2</td><td>TBD</td><td>
<table>
<tr><th>Name</th><th>Status</th><th>Description</th></tr>
<tr><td>Release Review</td><td></td><td>Release Review Description</td></tr>
<tr><td>Deliverable Name</td><td></td><td>Deliverable Description</td></tr>
</table>
</td></tr>
<tr><td>Formal Release</td><td>TBD</td><td>
<table>
<tr><th>Name</th><th>Status</th><th>Description</th></tr>
<tr><td>Deliverable Name</td><td></td><td>Deliverable Description</td></tr>
</table>
</td></tr>
</table>

Externally Consumable APIs

Fabric as a Service API

Expected Dependencies on Other Projects

- controller
- odlparent
- yangtools
- md-sal
- OVSDB-plugin
- openflow-plugin

Expected Incompatibilities with Other Projects

No incompatibilities with other projects

Compatibility with Previous Releases

Removed APIs and/or Functionality

New Project. N/A

Deprecated APIs and/or Functionality

No deprecated APIs or functionality.

Changed APIs and/or Functionality

No APIs or functionality will be removed.

Themes and Priorities

Requests from Other Projects

A faas Based GBP renderer will be done within GBP projects, FaaS API definition needs to factor in GBP requirements

Test Tools Requirements

- Java unit and integration tests

Other