API SUMMARY

本文档梳理并阐明k8s-prometheus-onos前端所调用的后端api。后端部分包含若干api组，分别映射到k8s及ONOS的不同api服务。

一、后端API

/network

API映射

```
http://front-ip:port/network -> http://onos-ip:port/onos/v1
```

API资源

1. 获取网络设备列表

- Request

```
GET /network/devices
```

- Response

```
{
  "devices": [
    {
      "id": "device:s1",
      "type": "SWITCH",
      "available": true,
      "mfr": "Open Networking Foundation",
      "hw": "BMv2 simple_switch",
      "sw": "Stratum",
      "lastUpdate": "1667300925492",
      "annotations": {
        "driver": "stratum-bmv2",
        "name": "device:s1",
        "protocol": "P4Runtime, gNMI, gNOI"
      }
    },
    ...
  ]
}
```

2. 获取网络设备安装的流表项

- Request

```
GET /network/flows/{deviceId}
```

- Response

```
{
  "devices": [
    {
      "groupId": 0,
      "state": "ADDED",
      "life": 77137,
      "lastSeen": 1667378063754,
      "packets": 84,
      "bytes": 12306,
      "id": "281475604904862",
      "appId": "org.onosproject.core",
      "priority": 5,
      "timeout": 0,
      "isPermanent": true,
      "deviceId": "device:s1",
      "tableId": 2130487575,
      "tableName": "ingress.table0_control.table0",
      "treatment": {
        "instructions": [
          {
            "type": "OUTPUT",
            "port": "CONTROLLER"
          }
        ],
        "clearDeferred": true,
        "deferred": []
      },
      "selector": {
        "criteria": [
          {
            "type": "ETH_TYPE",
            "ethType": "0x800"
          }
        ]
      }
    },
    ...
  ]
}
```

3. 获取网络中的主机

- Request

```
GET /network/hosts
```

- Response


```
{
  "hosts": [
    {
      "id": "08:00:27:59:9E:67/None",
      "mac": "08:00:27:59:9E:67",
      "ipAddresses": [
        "172.1.113.58"
      ],
      "locations": [
        {
          "elementId": "device:s5",
          "port": "4"
        }
      ]
    }
  ]
}
```


4. 获取设备端口

- Request

```
GET /network/devices/{id}/ports
```

- Response


```
{
  "id": "device:s1",
  "type": "SWITCH",
  "available": true,
  "mfr": "Open Networking Foundation",
  "hw": "BMv2 simple_switch",
  "sw": "Stratum",
  "lastUpdate": "1667300925492",
  "annotations": {
    "driver": "stratum-bmv2",
    "name": "device:s1",
    "protocol": "P4Runtime, gNMI, gNOI"
  },
  "ports": [
    {
      "element": "device:s1",
```

```
      "port": "[s1-eth1](1)",
      "isEnabled": true,
      "type": "copper",
      "portSpeed": 10000
    },
    {
      "element": "device:s1",
      "port": "[eth1](2)",
      "isEnabled": true,
      "type": "copper",
      "portSpeed": 10000
    }
  ]
}
```

5. 获取拓扑设备

- Request

```
GET /network/topology/clusters/{id}/devices
```

- Response

```
{
  "devices": [
    "device:s1",
    "device:s2",
    "device:s3",
    "device:s4",
    "device:s5",
    "device:s6",
    "device:s7",
    "device:s8"
  ]
}
```

6. 获取拓扑链路

- Request

```
GET /network/topology/clusters/{id}/links
```

- Response

```

```

```
{
  "links": [
    {
      "src": {
        "port": "[s1-eth1](1)",
        "device": "device:s1"
      },
      "dst": {
        "port": "1",
        "device": "device:s2"
      },
      "type": "DIRECT",
      "state": "ACTIVE"
    }
  ]
}
```

/cluster

API映射

```
http://front-ip:port/cluster -> http://k8s-ip:port/api/v1
```

API资源

1. 获取集群中节点列表

- Request

```
GET /cluster/nodes
```

- Response

```
{
    "items": [
        {
            "metadata": {
                "name": "master",
                "creationTimestamp": "2022-11-01T12:46:43Z"
            },
            "status": {
                "addresses": [
                    {
                        "type": "InternalIP",
                        "address": "172.1.113.58"
                    }
```

```
                ],
                "conditions": [
                    {
                        "type": "Ready",
                        "status": "True"
                    }
                ]
            },
            "nodeInfo": {
                "operatingSystem": "linux",
                "architecture": "amd64"
            }
        },
        ...
    ]
}
```

/task

API映射

```
http://front-ip:port/task -> http://k8s-ip:port/apis/apps/v1
```

API资源

1. 获取集群中任务列表

- Request

```
GET task/namespaces/default/deployments
```

- Response

```
{
  "items": [
    {
      "metadata": {
        "name": "nginx-dp",
        "namespace": "default",
        "creationTimestamp": "2021-07-19T03:04:19Z",
        "annotations": {
          "kubernetes.io/ingress-bandwidth": "10M",
          "kubernetes.io/egress-bandwidth": "10M"
        }
      },
```

```
        "spec": {
          "replicas": 1,
          "selector": {
            "matchLabels": {
              "app": "nginx"
            }
          },
          "template": {
            "metadata": {
              "creationTimestamp": null,
              "labels": {
                "app": "nginx"
              }
            },
            "spec": {
              "containers": [
                {
                  "name": "nginx",
                  "image": "nginx:1.14",
                  "ports": [
                    {
                      "containerPort": 80,
                      "protocol": "TCP"
                    }
                  ],
                  "resources": {
                    "limits": {
                      "cpu": "500m",
                      "ephemeral-storage": "4Gi",
                      "memory": "128Mi"
                    },
                    "requests": {
                      "cpu": "250m",
                      "ephemeral-storage": "2Gi",
                      "memory": "64Mi"
                    }
                  }
                }
              ]
            }
          },
          "strategy": {
            "type": "RollingUpdate",
            "rollingUpdate": {
              "maxUnavailable": "25%",
              "maxSurge": "25%"
            }
          }
        },
        "status": {
          "observedGeneration": 1,
          "replicas": 1,
          "updatedReplicas": 1,
          "readyReplicas": 1,
```

```
        "availableReplicas": 1
      }
    }
  ]
}
```

2. 创建集群任务

- Request

```
POST task/namespaces/default/deployments
```

- Request Body

```
{
  "apiVersion": "apps/v1",
  "kind": "Deployment",
  "metadata": {
    "name": "nginx-dp"
  },
  "spec": {
    "replicas": 1,
    "revisionHistoryLimit": 10,
    "selector": {
      "matchLabels": {
        "app": "nginx"
      }
    },
    "template": {
      "metadata": {
        "labels": {
          "app": "nginx"
        },
        "annotations": {
          "kubernetes.io/ingress-bandwidth": "1M",
          "kubernetes.io/egress-bandwidth": "1M"
        }
      },
      "spec": {
        "containers": [
          {
            "name": "nginx",
            "image": "nginx:1.14",
            "resources": {
              "requests": {
                "memory": "64Mi",
                "cpu": "250m",
                "ephemeral-storage": "2Gi"
              },
```

```
                "limits": {
                  "memory": "128Mi",
                  "cpu": "500m",
                  "ephemeral-storage": "4Gi"
                }
              },
              "ports": [
                {
                  "containerPort": 80
                }
              ]
            }
          ]
        }
      }
    }
  }
```

/metrics

1. 映射

```
http://front-ip:port/metrics -> http://onos-ip:port/apis/metrics.k8s.io/v1beta1
```

2. 资源

/custom-metrics

1. 映射

```
http://front-ip:port/custom-metrics -> http://onos-
ip:port/apis/custom.metrics.k8s.io/v1beta1
```

2. 资源

二、前端页面

I. 系统首页

展示系统全局拓扑。

拓扑中存在四种节点：cluster节点、network节点、task节点和IoT节点。节点之间通过链路连接，task节点和IoT节点仅能连接到cluster节点，network节点可以连接其他network节点以及cluster节点。双击cluster节点以显示/隐藏接入的task节点和IoT节点。

- 全局拓扑图

实现待定

II. 资源感知首页

展示系统全局资源占用情况。

包含系统磁盘总量、内存总量、总运行时间、CPU总数。动态展示实时CPU负载，内存负载，磁盘负载。全局CPU使用情况，全局内存使用情况，全局磁盘使用情况。

- 资源感知页面

页面通过嵌入grafana Dashboard实现。

III. 资源感知列表

展示集群中cluster节点，network节点的列表。

- 云端资源列表

表项	资源
节点名	cluster/node.item[i].metadata.name
IP地址	cluster/node.item[i].status.addresses[x(type==InternalIP)].address
操作系统	cluster/node.item[i].nodeInfo.operatingSystem
创建时间	cluster/node.item[i].metadata.creationTimestamp
状态	cluster/node.item[i].status.conditions[x(type==Ready)].status==True?就绪:未就绪
查看详情	button(name)

- 边端资源列表

内容同云端资源列表

- 传输资源列表

表项	资源
设备名称	network/devices[i].annotations.name
设备类型	network/devices[i].type
端口数量	network/devices/{id}/ports.length
支持协议	network/devices[i].annotations.protocol
状态	network/devices[i].available==true?就绪:未就绪
查看详情	button(id)

子页面：Cluster节点资源详情页

跳转自资源感知列表-云端资源列表/边端资源列表中的查看详情按钮。

展示选中节点的资源占用情况。

包含节点磁盘总量、内存总量、总运行时间、CPU总数。动态展示实时CPU负载，内存负载，磁盘负载。节点CPU使用情况，全局内存使用情况，全局磁盘使用情况。

- Cluster节点资源详情页

页面通过嵌入grafana Dashboard实现。

子页面：Network节点资源详情页

跳转自资源感知列表中的查看详情按钮。

展示选中节点的网络数据统计。

包含节点安装的流表项目数量，每个端口的线速，状态，转发的报文数量。 每个端口实时带宽的折线图。

- Network节点资源详情页

实现待定

IV. 设备感知列表

展示系统Cluster节点接入的设备列表。

- 设备感知列表

实现待定

V. 任务首页

展示集群中部署任务的资源使用情况。

- 任务首页

页面通过嵌入grafana Dashboard实现。

VI. 任务列表

展示集群中部署任务的列表。

- 任务列表

表项	资源
任务名称	task/item[i].metadata.name
镜像版本	task/item[i].spec.template.spec.containers[0].image
创建时间	task/item[i].metadata.creationTimestamp
副本数	task/item[i].spec.replicas
CPU需求	task/item[i].spec.template.spec.containers[0].resources.limits/requests.cpu
内存需求	task/item[i].spec.template.spec.containers[0].resources.limits/requests.memory
磁盘需求	task/item[i].spec.template.spec.containers[0].resources.limits/requests.ephemeral-storage

表项	资源
带宽需求	task/item[i].metadata.annotations."kubernetes.io/ingress-bandwidth/kubernetes.io/egress-bandwidth
边云负载比	
查看详情	button(id)

VII. 任务部署

填写表单以在系统中部署任务。

- 任务部署表单

表项	资源
任务名称	task/item[i].metadata.name
镜像版本	task/item[i].spec.template.spec.containers[0].image
副本数	task/item[i].spec.replicas
CPU需求	task/item[i].spec.template.spec.containers[0].resources.limits/requests.cpu
内存需求	task/item[i].spec.template.spec.containers[0].resources.limits/requests.memory
磁盘需求	task/item[i].spec.template.spec.containers[0].resources.limits/requests.ephemeral-storage
带宽需求	task/item[i].metadata.annotations."kubernetes.io/ingress-bandwidth/kubernetes.io/egress-bandwidth

VIII. 虚拟网络首页

展示系统中虚拟网络的拓扑。

- 虚拟网络拓扑

实现待定

IX. 虚拟网络列表

展示系统中虚拟网络列表。

- 虚拟网络列表

实现待定

X. 创建虚拟网络

填写表单以在系统中部署虚拟网络。

- 虚拟网络表单

实现待定