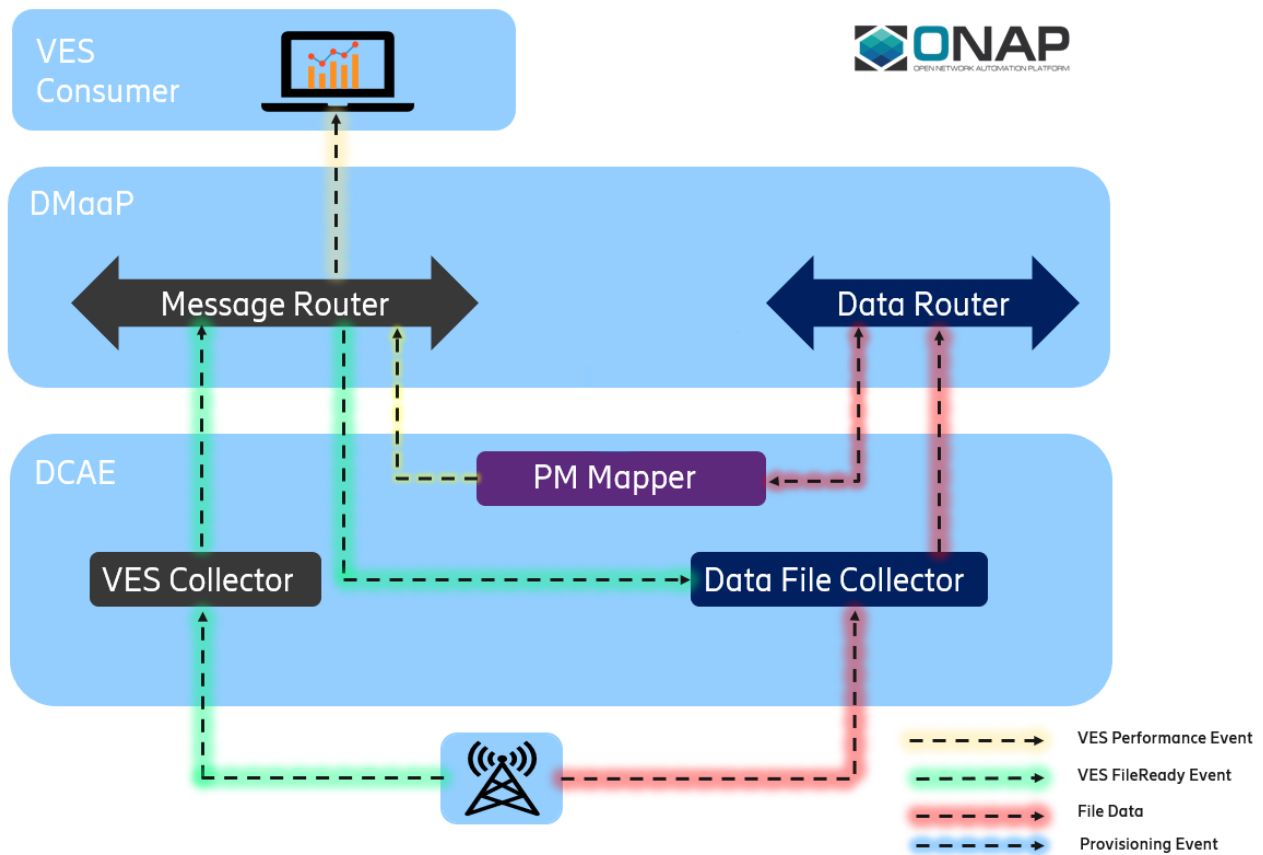


PM-Mapper (5G Usecase)

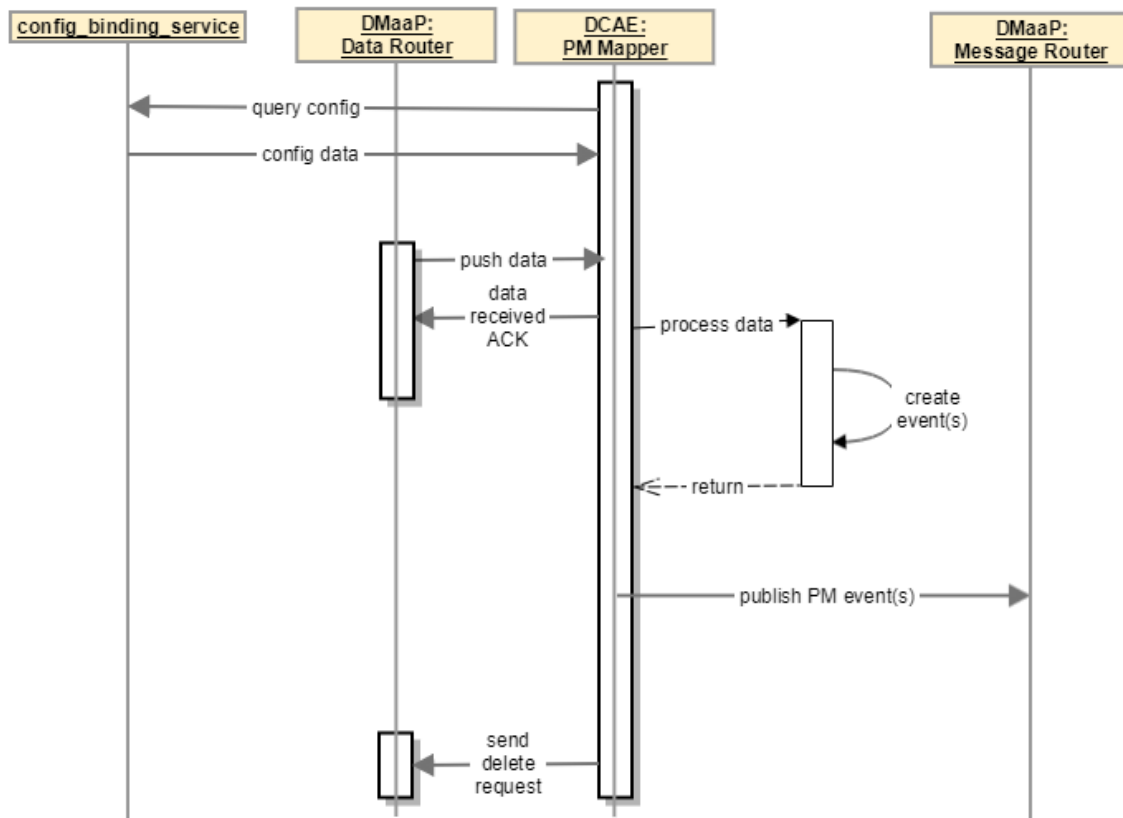
- [Overview](#)
- [Blueprint/model/image](#)
- [Deployment Prerequisite/dependencies](#)
- [Deployment Steps](#)
- [Validation](#)
- [Verification](#)
- [Configuration](#)
- [Offline ONAP Install workarounds](#)

Overview

PM-Mapper provides ONAP Operators with the ability to define flexible, customizable Performance events. **VES Events** are generated from **PM XML** Files and are targeted for analytics applications in ONAP.




PM Mapper Summary Sequence Diagram



Bulk PM Flow:

1. NF notifies *VES Collector* that a PM file is available for upload.
2. *Data File Collector* downloads PM Files from the NF and publishes them to *Data Router*.
3. **PM-Mapper** creates custom Performance events (VES) from the PM File data according to PM Mapping File.
4. Analytics Applications use these performance events for targeted analysis of network operations.

Blueprint/model/image

blocked URL pm-mapper blueprint	 pm-mapper models	blocked URL pm-mapper image: (TBD)
--	---	--

Deployment Prerequisite/dependencies

1. DCAE and DMaaP pods should be up and running.
2. DMaaP Bus Controller PostInstalls job should have completed successfully (executed as part of an OOM install).
3. Make sure that **cfy** is installed and configured to work with the Cloudify deployment.

Deployment Steps

1. Execute bash on the bootstrap Kubernetes pod.

```
kubectl -n onap exec -it <dcaegen2-dcae-bootstrap> bash
```

2. Download the [pm-mapper blueprint](#) and save to `/blueprints` directory.
3. Check that the `tag_version` of pm-mapper in the pm-mapper blueprint is correct for the release of ONAP that it is being installed on see [Nexus link for pm-mapper](#) for tag_versions
4. Create an inputs file (see the [configuration section](#) for more information).
5. Run the Cloudify Install command to install the pm-mapper with the recently downloaded blueprints, and the newly created inputs files.

```
cfy install --blueprint-id pm-mapper --deployment-id pm-mapper -i inputs.yaml <pm-mapper-blueprint-path>
```

```
$ cfy install --blueprint-id pm-mapper --deployment-id pm-mapper -i inputs.yaml blueprints/k8s-pm-mapper.yaml
```

```
Uploading blueprint k8s-pm-mapper.yaml...
k8s-pm-mapper.yaml |#####| 100.0%
Blueprint uploaded. The blueprint's id is pm-mapper
Creating new deployment from blueprint pm-mapper...
Deployment created. The deployment's id is pm-mapper
Executing workflow install on deployment pm-mapper [timeout=900 seconds]
Deployment environment creation is pending...
2019-04-11 16:00:37.497 CFY <pm-mapper> Starting 'create_deployment_environment' workflow execution
2019-04-11 16:00:38.692 CFY <pm-mapper> Installing deployment plugins
2019-04-11 16:00:38.692 CFY <pm-mapper> [,] Sending task 'cloudify_agent.operations.install_plugins'
2019-04-11 16:00:38.692 CFY <pm-mapper> [,] Task started 'cloudify_agent.operations.install_plugins'
2019-04-11 16:00:38.354 LOG <pm-mapper> [,] INFO: Installing plugin: k8s
2019-04-11 16:00:39.654 LOG <pm-mapper> [,] INFO: Using existing installation of managed plugin: 29023340-17d9-4737-886a-abba002334cd
[package_name: k8splugin, package_version: 1.4.5, supported_platform: linux_x86_64, distribution: centos, distribution_release: core]
2019-04-11 16:00:39.810 CFY <pm-mapper> [,] Task succeeded 'cloudify_agent.operations.install_plugins'
2019-04-11 16:00:40.717 CFY <pm-mapper> Skipping starting deployment policy engine core - no policies defined
2019-04-11 16:00:41.725 CFY <pm-mapper> Creating deployment work directory
2019-04-11 16:00:41.725 CFY <pm-mapper> 'create_deployment_environment' workflow execution succeeded
2019-04-11 16:00:45.447 CFY <pm-mapper> Starting 'install' workflow execution
2019-04-11 16:00:46.733 CFY <pm-mapper> [pm-mapper_twszn] Creating node
2019-04-11 16:00:46.733 CFY <pm-mapper> [pm-mapper_twszn.create] Sending task 'k8splugin.create_for_platforms'
2019-04-11 16:00:46.733 CFY <pm-mapper> [pm-mapper_twszn.create] Task started 'k8splugin.create_for_platforms'
2019-04-11 16:00:47.563 LOG <pm-mapper> [pm-mapper_twszn.create] INFO: Added config for dcae-pm-mapper
2019-04-11 16:00:48.675 LOG <pm-mapper> [pm-mapper_twszn.create] INFO: Done setting up: dcae-pm-mapper
2019-04-11 16:00:48.470 CFY <pm-mapper> [pm-mapper_twszn.create] Task succeeded 'k8splugin.create_for_platforms'
2019-04-11 16:00:48.745 CFY <pm-mapper> [pm-mapper_twszn] Configuring node
2019-04-11 16:00:49.759 CFY <pm-mapper> [pm-mapper_twszn] Starting node
2019-04-11 16:00:49.759 CFY <pm-mapper> [pm-mapper_twszn.start] Sending task 'k8splugin.create_and_start_container_for_platforms'
2019-04-11 16:00:49.759 CFY <pm-mapper> [pm-mapper_twszn.start] Task started 'k8splugin.create_and_start_container_for_platforms'
2019-04-11 16:00:51.693 LOG <pm-mapper> [pm-mapper_twszn.start] INFO: Passing k8sconfig: {'tls': {'u'cert_path': 'u/opt/tls/shared', 'u'image': 'u/nexus3.onap.org:10001/onap/org.onap.dcaegen2.deployments.tls-init-container:1.0.1-STAGING-latest'}, 'consul_host': 'consul-server:8500', 'consul_dns_name': 'u'consul-server.onap', 'image_pull_secrets': ['u'onap-docker-registry-key'], 'namespace': 'u'onap', 'filebeat': {'u'config_map': 'u'dcae-filebeat-configmap', 'u'config_path': 'u'/usr/share/filebeat/filebeat.yml', 'u'log_path': 'u'/var/log/onap', 'u'image': 'u'docker.elastic.co/beats/filebeat:5.5.0', 'u'data_path': 'u'/usr/share/filebeat/data', 'u'config_subpath': 'u'filebeat.yml'}}
2019-04-11 16:00:51.339 LOG <pm-mapper> [pm-mapper_twszn.start] INFO: Starting k8s deployment for dcae-pm-mapper, image: nexus3.onap.org:10001/onap/org.onap.dcaegen2.services.pm-mapper:1.0.0, env: {'CONSUL_HOST': 'u'consul-server.onap', 'CONFIG_BINDING_SERVICE': 'config-binding-service'}, kwargs: {'tls_info': {}, 'replicas': 1, 'labels': {'cfydeployment': 'u'pm-mapper', 'cfynodeinstance': 'u'pm-mapper_twszn', 'cfynode': 'u'pm-mapper'}, 'ctx': <cloudify.context.CloudifyContext object at 0x7f5fb5bb7710>, 'always_pull_image': False, 'u'ports': ['u'6162:0']}
2019-04-11 16:00:52.705 LOG <pm-mapper> [pm-mapper_twszn.start] INFO: k8s deployment initiated successfully for dcae-pm-mapper: {'services': ['dcae-pm-mapper'], 'namespace': 'u'onap', 'deployment': 'dep-dcae-pm-mapper'}
2019-04-11 16:00:52.705 LOG <pm-mapper> [pm-mapper_twszn.start] INFO: Waiting up to 300 secs for dcae-pm-mapper to become ready
2019-04-11 16:02:18.873 LOG <pm-mapper> [pm-mapper_twszn.start] INFO: k8s deployment ready for: dcae-pm-mapper
2019-04-11 16:02:19.621 CFY <pm-mapper> [pm-mapper_twszn.start] Task succeeded 'k8splugin.create_and_start_container_for_platforms'
2019-04-11 16:02:20.893 CFY <pm-mapper> 'install' workflow execution succeeded
Finished executing workflow install on deployment pm-mapper
* Run 'cfy events list -e 37da3f5f-a06b-4ce8-84d3-8b64ccd81c33' to retrieve the execution's events/logs
```

Validation

1. curl `<dcaegen2-dcae-healthcheck>` and check if pm-mapper is in `'ready'` state.

```
$ curl 10.42.181.66 | jq '.items[] | select(.name | contains("pm-mapper"))'
{
  "name": "dev-dcae-gen2-dcae-pm-mapper",
  "ready": 1,
  "unavailable": 0
}
```

Verification

To verify that the PM-Mapper is working as it should, we can publish a 3GPP XML file to the PM-Mapper, and verify that the counters in the outputted VES match what we expect based on our filtering.

To publish a file to the PM-Mapper we can use the following example curl. Where <filename> Must begin with A or C and the file extension must be xml.

```
<?xml version="1.0" encoding="UTF-8"?>
<measCollecFile xmlns="http://www.3gpp.org/ftp/specs/archive/32_series/32.435#measCollec">
  <fileHeader dnPrefix="some dnPrefix" vendorName="Acme Ltd"
    fileFormatVersion="32.435 V10.0">
    <fileSender localDn="Dublin"/>
    <measCollec beginTime="2018-10-02T12:00:00+01:00"/>
  </fileHeader>
  <measData>
    <managedElement swVersion="r0.1" localDn="Dublin"/>
    <measInfo measInfoId="some measInfoId">
      <job jobId="some Job Id"/>
      <granPeriod endTime="2018-10-02T12:15:00Z" duration="PT900S"/>
      <repPeriod duration="PT900S"/>
      <measType p="1">a</measType>
      <measType p="2">b</measType>
      <measType p="3">c</measType>
      <measValue measObjLdn="some measObjLdn">
        <r p="1">86</r>
        <r p="2">67</r>
        <r p="3">14</r>
        <suspect>>false</suspect>
      </measValue>
    </measInfo>
  </measData>
  <fileFooter>
    <measCollec endTime="2018-10-02T12:15:00+01:00"/>
  </fileFooter>
</measCollecFile>
```

```
curl -k -X PUT https://dcae-pm-mapper:8443/delivery/<filename> -H 'X-DMAAP-DR-META:{ "productName":
"AcmeNode", "vendorName": "Acme", "lastEpochMicrosec": "1538478000000", "sourceName": "oteNB5309",
startEpochMicrosec": "1538478900000", "timeZoneOffset": "UTC+05:00", "location": "ftpes://127.0.0.1:22
/ftp/rop/A20161224.1045-1100.bin.gz", "compression": "gzip", "fileFormatType": "org.3GPP.32.435
#measCollec", "fileFormatVersion": "V9"}' -H "Content-Type:application/xml" --data-binary @<filename> -
H 'X-ONAP-RequestID: 12345' -H 'X-DMAAP-DR-PUBLISH-ID: 12345'
```



Note

The productName and vendorName specified in X-DMAAP-DR-META must match the nftype and vendor specified during [configuration](#) of the PM-Mapper

We can then check that the XML has been processed and published to Message Router as a VES event by curling the PM-Mapper topic on Message Router.

We first need to set up a subscriber to the PM-Mapper topic as this is not set up by default. We can do this with the following curl to Bus Controller:

```
curl -X POST http://dmaap-bc:8080/webapi/mr_clients -H "Content-Type:application/json" --data @sub-
client.json
```

<= R4 release

```
{
  "dcaeLocationName": "san-francisco",
  "fqtn": "org.onap.dmaap.mr.PM_MAPPER",
  "clientIdentity": "dcae@dcae.onap.org",
  "action": "sub"
}
```

El Alto release:

```
{
  "dcaeLocationName": "san-francisco",
  "fqtn": "org.onap.dmaap.mr.PERFORMANCE_MEASUREMENTS",
  "clientIdentity": "dcae@dcae.onap.org",
  "action": "sub"
}
```

Once the subscriber is set up, we can curl the topic on Message Router to retrieve the published event:

```
<= R4 release
curl -k https://message-router:3905/events/org.onap.dmaap.mr.PM_MAPPER -u 'dcae@dcae.onap.org:
<dcae_password>'

El Alto release:
curl -k https://message-router:3905/events/org.onap.dmaap.mr.PERFORMANCE_MEASUREMENTS/$ConsumerGroup
/$ID -u 'dcae@dcae.onap.org:<dcae_password>'
e.g.
curl -k https://message-router:3905/events/org.onap.dmaap.mr.PERFORMANCE_MEASUREMENTS/1/1 -u
'dcae@dcae.onap.org:<dcae_password>'
```

This VES event can then be compared to the filter specified during [configuration](#) of the PM-Mapper to verify that only the specified counters have been published.

```

{
  "pm-mapper-filter": {
    "filters": [
      {
        "pmDefVsn": "1.8",
        "nfType": "AcmeNode",
        "vendor": "Acme",
        "measTypes": [
          "a",
          "c"
        ]
      }
    ]
  },
  "trust_store_pass_path": "/opt/app/pm-mapper/etc/cert/trust.pass",
  "key_store_path": "/opt/app/pm-mapper/etc/cert/cert.jks.b64",
  "streams_subscribes": {
    "dmaap_subscriber": {
      "type": "data_router",
      "dmaap_info": {
        "username": "username",
        "password": "password",
        "location": "san-francisco",
        "delivery_url": "https://dcae-pm-mapper:8443/delivery",
        "subscriber_id": "1"
      }
    }
  },
  "trust_store_path": "/opt/app/pm-mapper/etc/cert/trust.jks.b64",
  "streams_publishes": {
    "dmaap_publisher": {
      "aaf_password": "password",
      "type": "message_router",
      "dmaap_info": {
        "topic_url": "http://message-router:3904/events/org.onap.dmaap.mr.PM_MAPPER",
        "client_role": "org.onap.dmaap.mr.PM_MAPPER.pub",
        "location": "san-francisco",
        "client_id": "dcae@dcae.onap.org"
      }
    },
    "aaf_username": "username"
  },
  "dmaap_dr_delete_endpoint": "https://dmaap-dr-node:8443/delete",
  "key_store_pass_path": "/opt/app/pm-mapper/etc/cert/jks.pass",
  "enable_http": false,
  "dmaap_dr_feed_name": "bulk_pm_feed"
}

```

Using the example type A file above, and the example key/value configuration in consul, the PM Mapper will produce the following VES output, note that only counter "a" and "c" were mapped, as specified in the filter configuration.

```

{
  "event": {
    "commonEventHeader": {
      "domain": "perf3gpp",
      "eventId": "f544c442-9dc1-41d4-8f0d-6567e1ea7729",
      "sequence": 0,
      "eventName": "perf3gpp_AcmeNode-Acme_pmMeasResult",
      "sourceName": "oteNB5309",
      "reportingEntityName": "",
      "priority": "Normal",
      "startEpochMicrosec": 1538478000000,
      "lastEpochMicrosec": 1538478900000,
      "version": "4.0",
      "vesEventListenerVersion": "7.1",
      "timeZoneOffset": "UTC+05:00"
    },
    "perf3gppFields": {
      "perf3gppFieldsVersion": "1.0",
      "measDataCollection": {
        "granularityPeriod": 1538482500000,
        "measuredEntityUserName": "",
        "measuredEntityDn": "Dublin",
        "measuredEntitySoftwareVersion": "r0.1",
        "measInfoList": [
          {
            "measInfoId": {
              "sMeasInfoId": "some measInfoId"
            },
            "measTypes": {
              "sMeasTypesList": [
                "a",
                "c"
              ]
            },
            "measValuesList": [
              {
                "measObjInstId": "some measObjLdn",
                "suspectFlag": "false",
                "measResults": [
                  {
                    "p": 1,
                    "sValue": "86"
                  },
                  {
                    "p": 3,
                    "sValue": "14"
                  }
                ]
              }
            ]
          }
        ]
      }
    }
  }
}

```

Configuration

Configuration of the service consists of generating an inputs file (YAML) which will be used as part of the Cloudify install. The PM-Mapper blueprints were designed with sane defaults for the majority of the fields.
Below you will find some examples of fields which *can* be configured, and some of the fields which **must** be configured (These may change depending on the version of ONAP).

Property	Sample Value	Description	Required
aaf_username	dcae@dcae.onap.org	In the Dublin release information about the AAF user must be provided to enable publishing to authenticated topics.	Yes
aaf_password	<dcae_password>	This is the password for the given user e.g. The <dcae_password> is dcae@dcae.onap.org 's password.	Yes
enable_http	true	By default, the PM-Mapper will only allow inbound queries over HTTPS. However, it is possible to configure it to enable HTTP also.	No
tag_version	nexus3.onap.org:10001/onap/org.onap.dcae2.services.pm-mapper:1.0.1	By default the latest Docker Images will be used when deploying the PM-Mapper. Replace <version> with the desired Docker image version.	No
pm-mapper-filter	{'filters': [{'pmDefVsn': 'targetVersion', 'nfType': 'targetNodeType', 'vendor': 'targetVendor', 'measTypes': ['targetMeasType']}]}	The default behavior of the PM-Mapper is to map all measType in the received PM XML files, however, it's possible to provide filtering configuration which will reduce the VES event to the counters that the designer has expressed interest in. In this provided example a VES event containing the "targetMeasType" will only be generated and published if the following conditions are true: <ul style="list-style-type: none"> The vendor of the node sending the file is "targetVendor" The type of the node sending the file is "targetNodeType" Additional Information on the filter model can be found here .	No

inputs.yaml

```
aaf_username: dcae@dcae.onap.org

aaf_password: demo123456!

enable_http: false

tag_version: nexus3.onap.org:10001/onap/org.onap.dcae2.services.pm-mapper:1.0.1

filter: {'filters': []}
```

inputs.yaml

```
client_password : demo123456!

filter: {'filters': []}
```

Offline ONAP Install workarounds

If you want to instantiate PM-Mapper (and Datafile Collector) they will need to be copied from a server with internet access to the ONAP deployment which doesn't.

[OOM-2243](#) when completed will include the PM-Mapper and Datafile Collector packages in the offline deployment environment.

Create the docker image on a server which has internet access i.e.

docker pull [nexus3.onap.org:10001/onap/org.onap.dcae2.services.pm-mapper:1.2.0](#)

1.2.0: Pulling from [onap/org.onap.dcae2.services.pm-mapper](#)

Digest: sha256:f6a9349b575d7a62525196fec4f610a773db0ec6ff49be0bc6d0b134fc52a76d

Status: Downloaded newer image for [nexus3.onap.org:10001/onap/org.onap.dcae2.services.pm-mapper:1.2.0](#)

[nexus3.onap.org:10001/onap/org.onap.dcae2.services.pm-mapper:1.2.0](#)

docker images

REPOSITORY TAG IMAGE ID CREATED SIZE

[nexus3.onap.org:10001/onap/org.onap.dcae2.services.pm-mapper](#) 1.2.0 2bb6a1b426ab 36 minutes ago 101MB

docker save 2bb6a1b426ab > 1.2.0.mapper.tar

On each of the K8s worker nodes copy the docker image for mapper (and DFC).
i.e.

```
cd /dockerdata-nfs/onap-dcaegen2/  
docker load -i 1.2.0.mapper.tar  
docker images  
docker tag 06776409bc03 dcaegen2/pmmapper.1.2.0
```

On on bootstrap pod. (Look at the imports that are required in the blueprints) i.e
imports:

```
- 'http://www.getcloudify.org/spec/cloudify/4.5.5/types.yaml'  
- 'https://nexus.onap.org/service/local/repositories/raw/content/org.onap.dcaegen2.platform.plugins/R5/k8splugin/1.6.0/k8splugin_types.yaml'  
- 'https://nexus.onap.org/service/local/repositories/raw/content/org.onap.ccsdk.platform.plugins/type_files/dmaap/dmaap.yaml'
```

For each of the above yaml files open them on a server that has internet access then create a corresponding file on the bootstrap pod and paste the contents into that yaml.

so it becomes

```
imports:  
- 455types.yaml  
- 1608splugin.yaml  
- dmaap.yaml
```

And also change

```
tag_version:  
type: string  
description: Docker image to be used  
default: dcaegen2/pmmapper.1.2.0
```