atmosphere-eubrazil.eu @AtmosphereEUBR

ATMOSPHERE

Adaptive, Trustworthy, Manageable, Orchestrated, Secure Privacy-assuring Hybrid, Ecosystem for REsilient Cloud Computing

TMA Framework Usage Demo

José D'Abruzzo Pereira
Rui Silva

Co-funded by the European Commission
Horizon 2020 - Grant #777154

European Commission

Outline

1. Kubernetes Cluster Initialization
2. TMA_Monitor Deployment
3. Client Usage
4. Probe Development
5. Containers Metrics Reported

1.Kubernetes Cluster Initialization

Kubernetes Cluster Initialization

- Master Initialization

```
root@kubernetesMaster:/home/kubernetesmaster# kubeadm init --apiserver-advertise-address 192.168.1.1 --pod-network-cidr=10.244.0.0/16
```

- Master Configuration

```
root@kubernetesMaster:/home/kubernetesmaster#   mkdir -p $HOME/.kube
root@kubernetesMaster:/home/kubernetesmaster#   sudo cp -i /etc/kubernetes/admin.conf $HOME/.kube/config
cp: overwrite '/root/.kube/config'? y
root@kubernetesMaster:/home/kubernetesmaster# _ sudo chown $(id -u):$(id -g) $HOME/.kube/config
```

- Network Plugin Installation (Flannel)

```
root@kubernetesMaster:/home/kubernetesmaster/Desktop/tma-framework-m/development/dependency/kubernetes# sh network_kubernetes.sh
```

- Add static route for Kubernetes DNS

```
root@kubernetesMaster:/home/kubernetesmaster# ip route add  10.96.0.0/16 dev enp0s8
```

Kubernetes Cluster Initialization

- Join a worker node to Kubernetes Cluster

```
root@kubernetes-worker:~#   kubeadm join 192.168.1.1:6443 --token 9qcfqv.20kd8nk
j0e9d1db1 --discovery-token-ca-cert-hash sha256:465fb63e2aae196373dfe551bcc9534c
b98c2e4ae027390ab93dd4ccf63700f8
```

All previous commands are explained in more detail in this README:
https://github.com/eubr-atmosphere/tma-framework-m/tree/master/development/server#prerequisites

2. TMA_Monitor Deployment

TMA_Monitor Deployment

Worker Node

- Build base Docker image of Monitor

```
root@kubernetes-worker:/home/kubernetesworker/Desktop/tma-framework-m/development/dependency/python-base# sh build.sh
```

- Build Docker image of Monitor on Worker node

```
root@kubernetes-worker:/home/kubernetesworker/Desktop/tma-framework-m/development/server/monitor-server-python# sh build.sh
```

- Build Docker image of Apache Kafka

```
root@kubernetes-worker:/home/kubernetesworker/Desktop/tma-framework-m/development/server/kafka# sh build.sh
```

- Build Docker image of Apache Zookeeper

```
root@kubernetes-worker:/home/kubernetesworker/Desktop/tma-framework-m/development/server/zookeeper# sh build.sh
```

TMA_Monitor Deployment

Master Node

- Deployment of Apache Kafka, and Apache Zookeeper using **setup-testing-mode.sh** script
  - Deploys Apache Kafka and Apache Zookeeper persistent volumes;
  - Deploys Apache Kafka and Apache Zookeeper images;
  - Creates Apache Kafka topic topic-monitor.

TMA_Monitor Deployment

```
root@kubernetesMaster:/home/kubernetesmaster/Desktop/tma-framework-m/development/server# sh setup-testing-mode.sh
persistentvolume "datadir" created
persistentvolume "datadir-kafka" created
service "zk-hs" created
service "zk-cs" created
statefulset.apps "zk" created
service "kafka-hs" created
poddisruptionbudget.policy "kafka-pdb" created
statefulset.apps "kafka" created
Created topic "topic-monitor".
```

TMA_Monitor Deployment

Master Node

- Deployment of Monitor

```
root@kubernetesMaster:/home/kubernetesmaster/Desktop/tma-framework-m/development/server/monitor-server-python# kubectl create -f monitor-api-python.yaml
```

- With Monitor deployed, it can be accessed by the following endpoint:

https://IP_MASTER:32025/monitor

3. Client Usage

Client Usage

- Probes must be deployed to generate valid data and send them to Monitor endpoint.

Worker Node

- Build Probe Docker base image

```
root@kubernetes-worker:/home/kubernetesworker/Desktop/tma-framework-m/development/dependency/python-probe-base# sh build.sh
```

- Build Probe Docker image

```
root@kubernetes-worker:/home/kubernetesworker/Desktop/tma-framework-m/development/probes/probe-python-demo# sh build.sh
```

Client Usage

Master Node

- Deployment of Probe

```
root@kubernetesMaster:/home/kubernetesmaster/Desktop/tma-framework-m/development# kubectl create -f probes/probe-python-demo/probe-python-demo.yaml
```

Testing

- Start an Apache Kafka consumer that receives all monitor data inside Apache Kafka pod.

```
root@kubernetesmaster-VirtualBox:/home/kubernetesmaster/Desktop/tma-framework-m/development/server# kubectl exec -ti kafka-0 -- bash
kafka@kafka-0:/$ kafka-console-consumer.sh --topic topic-monitor --bootstrap-server localhost:9093
```

Client Usage

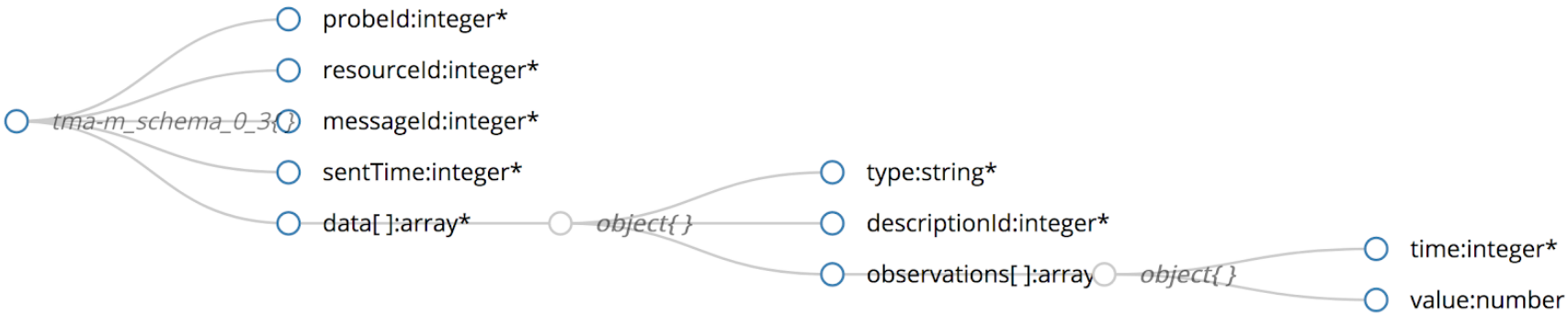
{"resourceId": 101098, "probeId": 0, "data": [{"descriptionId": 0, "type": "measurement", "observations": [{"value": 20000.00001, "time": 1535971455}]}, {"descriptionId": 0, "type": "event", "observations": [{"value": 10000.00001, "time": 1535971455}]}, {"descriptionId": 1, "type": "measurement", "observations": [{"value": 20001.00001, "time": 1535971455}]}, {"descriptionId": 1, "type": "event", "observations": [{"value": 10001.00001, "time": 1535971455}]}, {"descriptionId": 2, "type": "measurement", "observations": [{"value": 20002.00001, "time": 1535971455}]}, {"descriptionId": 2, "type": "event", "observations": [{"value": 10002.00001, "time": 1535971455}]}, {"descriptionId": 3, "type": "measurement", "observations": [{"value": 20003.00001, "time": 1535971455}]}, {"descriptionId": 3, "type": "event", "observations": [{"value": 10003.00001, "time": 1535971455}]}, {"descriptionId": 4, "type": "measurement", "observations": [{"value": 20004.00001, "time": 1535971455}]}, {"descriptionId": 4, "type": "event", "observations": [{"value": 10004.00001, "time": 1535971455}]}, {"descriptionId": 5, "type": "measurement", "observations": [{"value": 20005.00001, "time": 1535971455}]}, {"descriptionId": 5, "type": "event", "observations": [{"value": 10005.00001, "time": 1535971455}]}, {"descriptionId": 6, "type": "measurement", "observations": [{"value": 20006.00001, "time": 1535971455}]}, {"descriptionId": 6, "type": "event", "observations": [{"value": 10006.00001, "time": 1535971455}]}, {"descriptionId": 7, "type": "measurement", "observations": [{"value": 20007.00001, "time": 1535971455}]}, {"descriptionId": 7, "type": "event", "observations": [{"value": 10007.00001, "time": 1535971455}]}, {"descriptionId": 8, "type": "measurement", "observations": [{"value": 20008.00001, "time": 1535971455}]}, {"descriptionId": 8, "type": "event", "observations": [{"value": 10008.00001, "time": 1535971455}]}, {"descriptionId": 9, "type": "measurement", "observations": [{"value": 20009.00001, "time": 1535971455}]}, {"descriptionId": 9, "type": "event", "observations": [{"value": 10009.00001, "time": 1535971455}]}], "sentTime": 1535971455, "messageId": 0}


```
{
   "resourceId":101098,
   "probeId":0,
   "data":[
      {
         "descriptionId":0,
         "type":"measurement",
         "observations":[
            {
               "value":20000.00001,
               "time":1535971455
            }
         ]
      },
      {
         "descriptionId":0,
         "type":"event",
         "observations":[
            {
               "value":10000.00001,
               "time":1535971455
            }
         ]
      },
   ],
   "sentTime":1535971455,
   "messageId":0
}
```

4. Probe Development

Probe Development

- TMA_Monitor supports any probe that sends the collected data according the following schema:



- Any programming language is supported.

Probe Development

- There are base Docker images that support probes written both in Java and Python that can be used;
- All base images already have the Monitor needed certificate to establish a session with it;
- All base images are in dependency folder of the tma-framework-m repository;
- All Docker images of probes to be developed must be built from the respective Docker base image.

Probe Development

Example: Probe Python Demo

- It generates random valid data;
- All files are presented in this directory:

```
root@kubernetesMaster:/home/kubernetesmaster/Desktop/tma-framework-m/development/probes/probe-python-demo# ls
build.sh  communication.py  data.py  Dockerfile  message.py  observation.py  probe-python-demo.py  probe-python-demo.yaml  README.md
```

- This probe is composed by:
  - probe-python-demo.py;
  - communication.py;
  - data.py;
  - message.py;
  - observation.py.

Probe Development

Example: Probe Python Demo

- probe-python-demo.py
  - Main file of the probe;
  - Generates random values for:
    - descriptionId;
    - value;
  - Time field gets the value of timestamp.
- communication.py
  - Class that sends message to Monitor endpoint.

Probe Development

Example: Probe Python Demo

- data.py
  - Class that builds data arrays with values of:
    - type;
    - descriptionId;
    - observations arrays.

Probe Development

Example: Probe Python Demo

- message.py
  - Class that builds the message to send to Monitor with values of:
    - probeId;
    - resourceId;
    - messageId;
    - sentTime;
    - data object defined in data.py file.

Probe Development

Example: Probe Python Demo

- observation.py
  - Class that builds observation arrays with values of:
    - time;
    - value.

Probe Development

Example: Probe Python Demo

- After writing all code files, it is needed to build probe Docker image in Worker node. In this case:

```
root@kubernetes-worker:/home/kubernetesworker/Desktop/tma-framework-m/development/probes/probe-python-demo# sh build.sh
```

- Finally, deploy the probe in Kubernetes executing the following command in Kubernetes Master:

```
root@kubernetesMaster:/home/kubernetesmaster/Desktop/tma-framework-m/development# kubectl create -f probes/probe-python-demo/probe-python-demo.yaml
```

Example: Probe Python Demo

- For testing purposes, execute an Apache Kafka consumer in Apache Kafka pod:

```
root@kubernetesmaster-VirtualBox:/home/kubernetesmaster/Desktop/tma-framework-m/development/server# kubectl exec -ti kafka-0 -- bash
kafka@kafka-0:/$ kafka-console-consumer.sh --topic topic-monitor --bootstrap-server localhost:9093
```

- The data is received:


```
iptionId": 5, "type": "event", "observations": [{"value": 10005.00001, "time": 1
535878544}]}, {"descriptionId": 6, "type": "measurement", "observations": [{"val
ue": 20006.00001, "time": 1535878544}]}, {"descriptionId": 6, "type": "event", "
observations": [{"value": 10006.00001, "time": 1535878544}]}, {"descriptionId":
7, "type": "measurement", "observations": [{"value": 20007.00001, "time": 153587
8544}]}, {"descriptionId": 7, "type": "event", "observations": [{"value": 10007.
00001, "time": 1535878544}]}, {"descriptionId": 8, "type": "measurement", "obser
vations": [{"value": 20008.00001, "time": 1535878544}]}, {"descriptionId": 8, "t
ype": "event", "observations": [{"value": 10008.00001, "time": 1535878544}]}, {"
descriptionId": 9, "type": "measurement", "observations": [{"value": 20009.00001
, "time": 1535878544}]}, {"descriptionId": 9, "type": "event", "observations": [
{"value": 10009.00001, "time": 1535878544}]}], "sentTime": 1535878544, "messageI
d": 0}
```

Probe Development

Example: Probe Java Demo

- The development of Java probes can be done by using a library which is available through Maven:

```
mvn clean install
```

Reference: https://github.com/eubr-atmosphere/tma-framework-m/tree/master/development/libraries

- To use the library, add the following reference to the probe:

```
<dependency>
    <groupId>eu.atmosphere.tmaf</groupId>
    <artifactId>monitor-client</artifactId>
    <version>0.1</version>
</dependency>
```

Probe Development

Example: Probe Java Demo

- The following java code starts a client, and send the measurements to the monitor

```
BackgroundClient client = new BackgroundClient();
client.authenticate(1098, "pass".getBytes());

Message message;

boolean start = client.start();
message = client.createMessage();
message.setResourceId(101098);

long now = (new Date()).getTime();
message.addData(new Data(Data.Type.EVENT, 33,
        new Observation(now, 724.0)));
message.addData(new Data(Data.Type.MEASUREMENT, 34,
        new Observation(now, 342.0)));
client.send(message);
```

5. Container Metrics Reported

Container Metrics Reported

- Probe that collects some metrics of a Kubernetes pod;
- K8s probe is in this directory:

```
root@kubernetesmaster-VirtualBox:/home/kubernetesmaster/Desktop/tma-framework-m/development/probes/probe-k8s-docker# ls
build.sh  cert.pem  data.py  dockerAPI.py  Dockerfile  message.py  observation.py  probe-k8s-docker.yaml  README.md
```

- K8s probe is composed by four python files:
  - dockerAPI.py;
  - message.py;
  - data.py;
  - observation.py.

Container Metrics Reported

- dockerAPI.py
  - Metrics such as cpu values, memory usage and disk accesses are collected;
  - It sends these metrics in a format of json to TMA-Monitor;
  - The format of the messages respects the schema of this project.
- message.py
  - Builds the message to send to Monitor API.

Container Metrics Reported

- data.py
  - Class that represents data object of schema, specifies its type, descriptionId and observations.
- observation.py
  - Constructs observation object with values of time (timestamp) and value.

Container Metrics Reported

- This probe receives as input the name of the docker container to monitor, and the url of the TMA-Monitor;

Worker Node

- Build Probe Docker image on Worker node;

```
root@kubernetesworker-VirtualBox:/home/kubernetesworker/Desktop/tma-framework-m/development/probes/probe-k8s-docker# sh build.sh
```

- Docker container is executed that will be the managed system;

```
root@kubernetes-worker:~# docker run -d --name monitor-api tma-monitor/server-python:0.2
```

Container Metrics Reported

Master Node

- Probe deployment;

```
root@kubernetesmaster-VirtualBox:/home/kubernetesmaster/Desktop/tma-framework-m/development/probes/probe-k8s-docker# kubectl create -f probe-k8s-docker.yaml
```

Testing

- Start an Apache Kafka consumer that receives all monitor data inside Apache Kafka pod.

```
root@kubernetesmaster-VirtualBox:/home/kubernetesmaster/Desktop/tma-framework-m/development/server# kubectl exec -ti kafka-0 -- bash
kafka@kafka-0:/$ kafka-console-consumer.sh --topic topic-monitor --bootstrap-server localhost:9093
```

Container Metrics Reported

{"resourceId": 0, "probeId": 0, "data": [{"descriptionId": 0, "type": "measurement", "observations": [{"value": 8, "time": 1535980994}]}, {"descriptionId": 1, "type": "measurement", "observations": [{"value": 0, "time": 1535980994}]}, {"descriptionId": 2, "type": "measurement", "observations": [{"value": 24576, "time": 1535980994}]}, {"descriptionId": 3, "type": "measurement", "observations": [{"value": 8, "time": 1535980994}]}, {"descriptionId": 4, "type": "measurement", "observations": [{"value": 0, "time": 1535980994}]}, {"descriptionId": 5, "type": "measurement", "observations": [{"value": 0, "time": 1535980994}]}, {"descriptionId": 6, "type": "measurement", "observations": [{"value": 8, "time": 1535980994}]}, {"descriptionId": 7, "type": "measurement", "observations": [{"value": 0, "time": 1535980994}]}, {"descriptionId": 8, "type": "measurement", "observations": [{"value": 24576, "time": 1535980994}]}, {"descriptionId": 9, "type": "measurement", "observations": [{"value": 8, "time": 1535980994}]}, {"descriptionId": 10, "type": "measurement", "observations": [{"value": 0, "time": 1535980994}]}, {"descriptionId": 11, "type": "measurement", "observations": [{"value": 0, "time": 1535980994}]}, {"descriptionId": 12, "type": "measurement", "observations": [{"value": 8, "time": 1535980994}]}, {"descriptionId": 13, "type": "measurement", "observations": [{"value": 0, "time": 1535980994}]}, {"descriptionId": 14, "type": "measurement", "observations": [{"value": 24576, "time": 1535980994}]}, {"descriptionId": 15, "type": "measurement", "observations": [{"value": 8, "time": 1535980994}]}, {"descriptionId": 16, "type": "measurement", "observations": [{"value": 0, "time": 1535980994}]}, {"descriptionId": 17, "type": "measurement", "observations": [{"value": 6, "time": 1535980994}]}, {"descriptionId": 18, "type": "measurement", "observations": [{"value": 8, "time": 1535980994}]}, {"descriptionId": 19, "type": "measurement", "observations": [{"value": 0, "time": 1535980994}]}, {"descriptionId": 20, "type": "measurement", "observations": [{"value": 0, "time": 1535980994}]}, {"descriptionId": 21, "type": "measurement", "observations": [{"value": 8, "time": 1535980994}]}, {"descriptionId": 22, "type": "measurement",


```
{
    "resourceId":0,
    "probeId":0,
    "data":[
        {
            "descriptionId":0,
            "type":"measurement",
            "observations":[
                {
                    "value":8,
                    "time":1535980994
                }
            ]
        },
        {
            "descriptionId":1,
            "type":"measurement",
            "observations":[
                {
                    "value":0,
                    "time":1535980994
                }
            ]
        }
    ],
    "sentTime":1535980994,
    "messageId":0
}
```

Thank you very much for your attention!

Questions?

mvieira@dei.uc.pt
nmsa@dei.uc.pt
josep@dei.uc.pt
rfsilva@student.dei.uc.pt

ATMOSPHERE
Adaptive, Trustworthy, Manageable, Orchestrated, Secure Privacy-assuring Hybrid, Ecosystem for REsilient Cloud Computing

atmosphere-eubrazil.eu