

sonata

agile service development and orchestration in 5G virtualized networks



SONATA SDK demo & hands-on

Steven Van Rossem

 **lmec**

HORIZON
2020



Agenda

- **SDK architecture**
- Building the Service Descriptor
- Testing the service
- Deploy the service in production



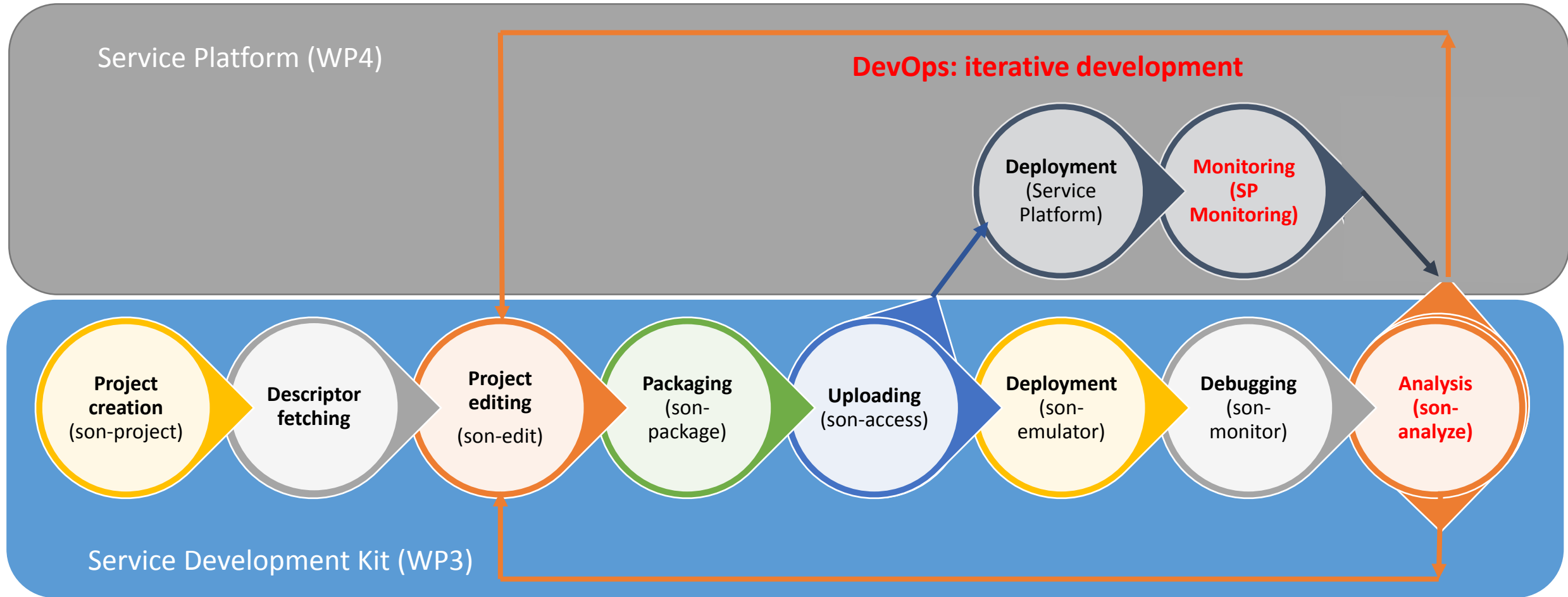
SONATA SDK : a sandbox for NFV-based services

The SONATA SDK:

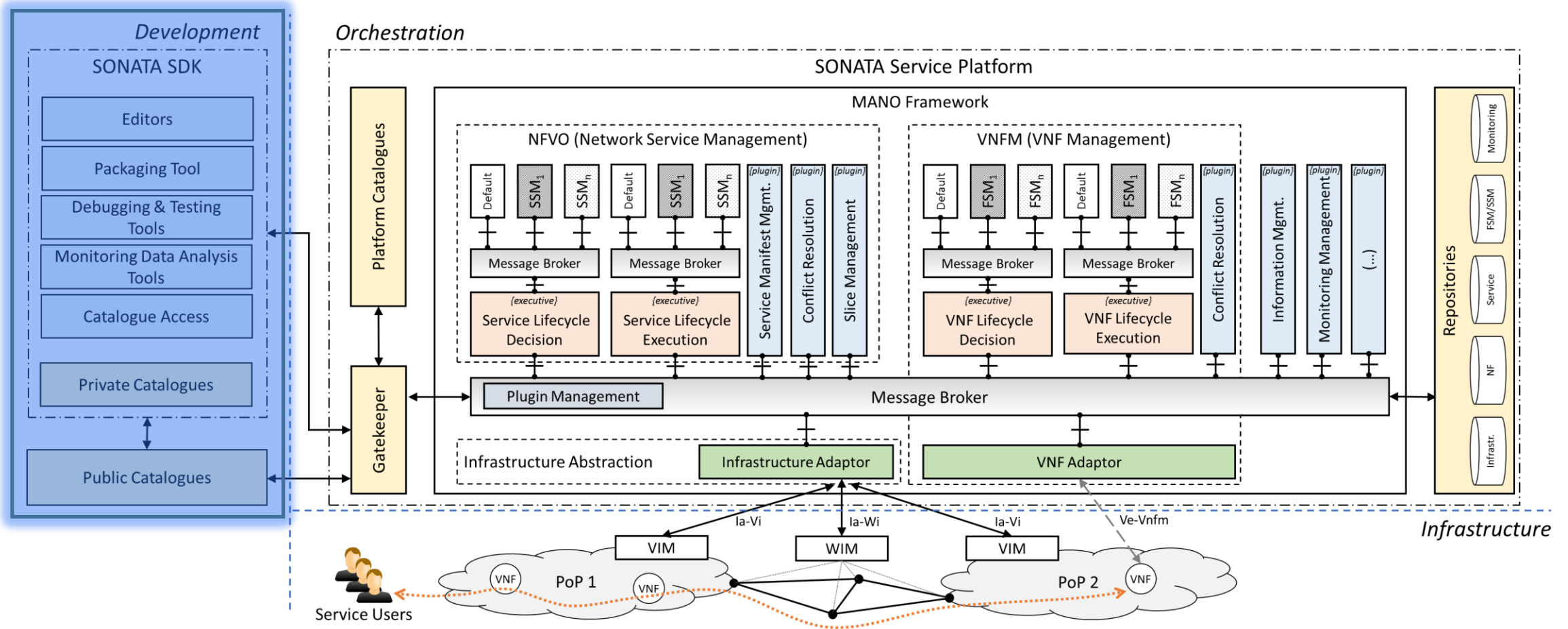
- Aim: an easy to use framework to try-out VNF-based service aspects such as: SFC, VNF configuration
- Modular approach
- Sandbox environment to give a service confidence the service will work, before it is deployed in production.



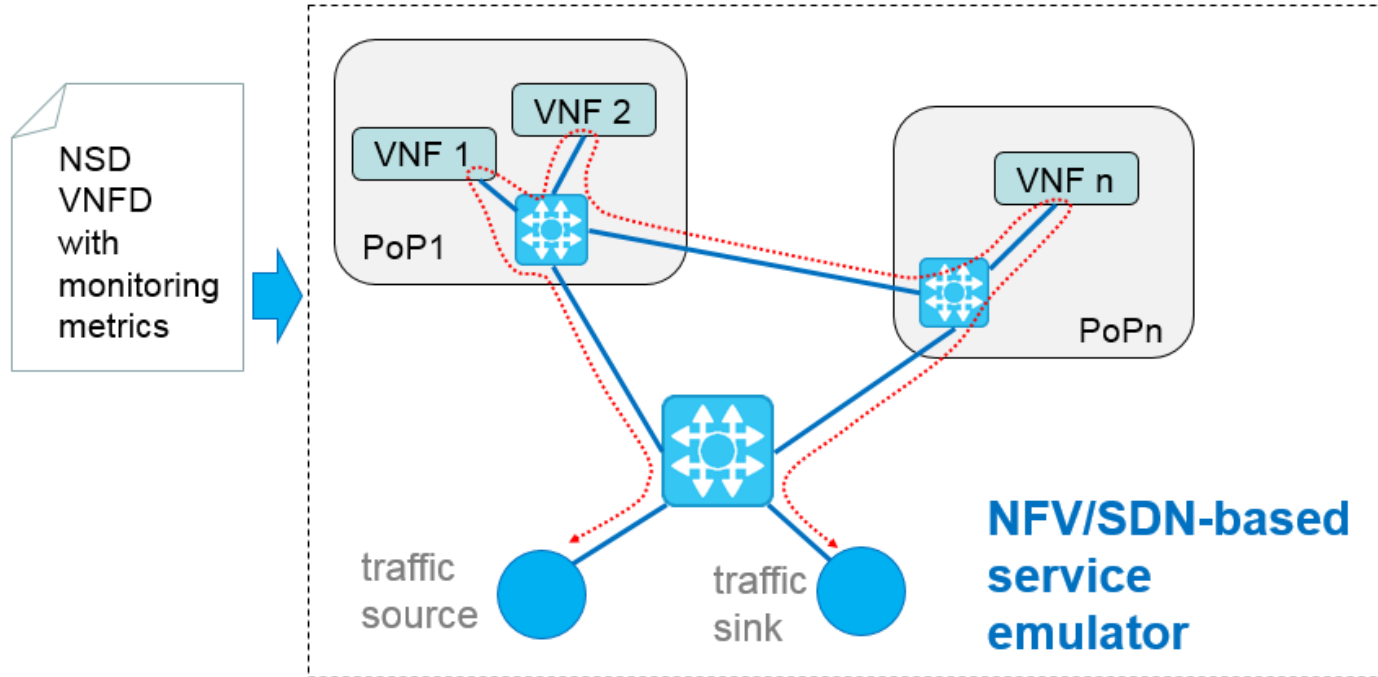
Service Development Workflow



SDK in the global SONATA architecture



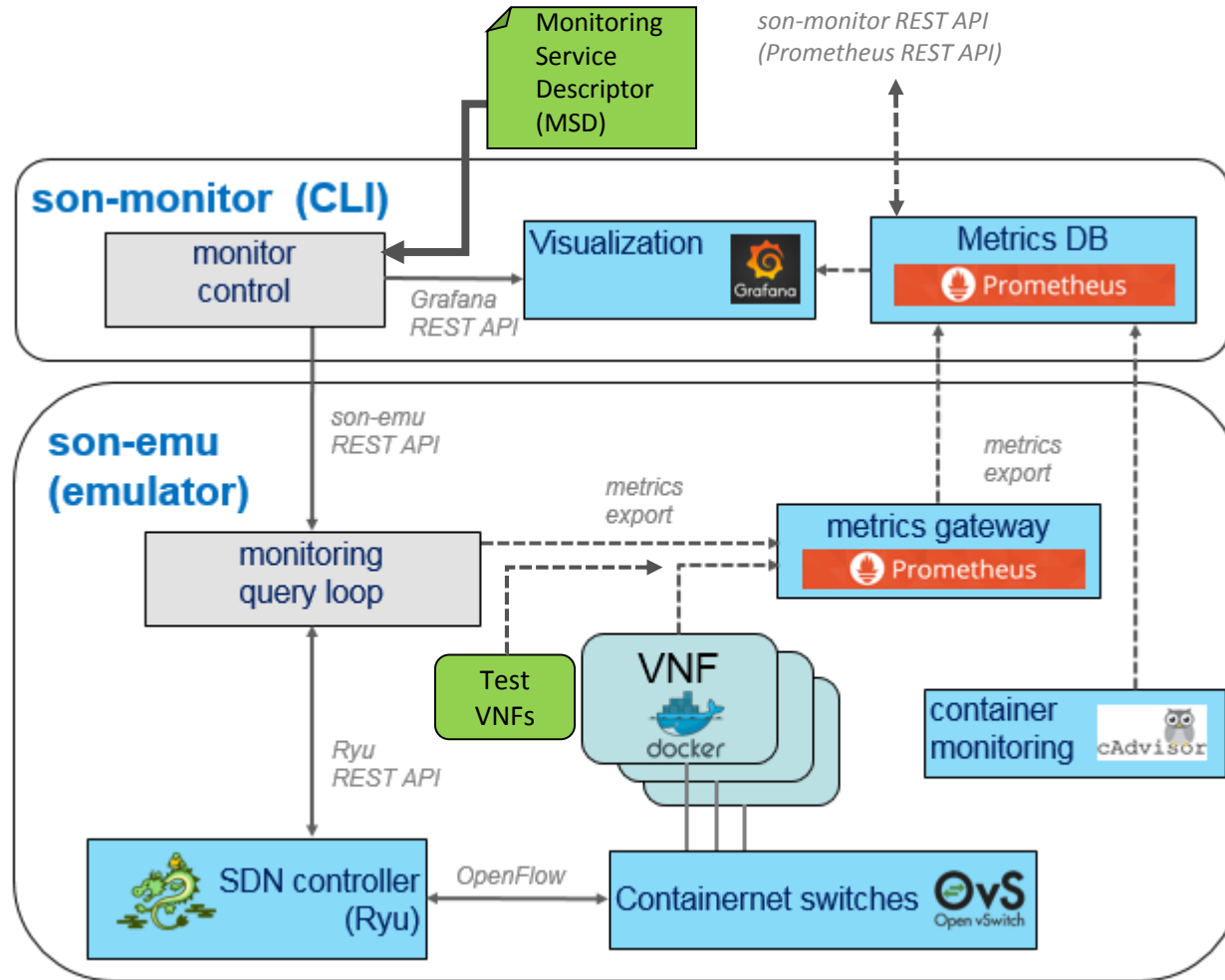
SONATA SDK Emulator



An SDK environment for NFV/SDN based services:

- Rapid prototyping of network services in a multi-PoP environment
- A descriptor format to define the service
- Custom Service Function Chaining (SFC)
- User-defined monitoring and traffic generation
- Sandbox for fast deployment, configuration and debugging of production-ready network functions

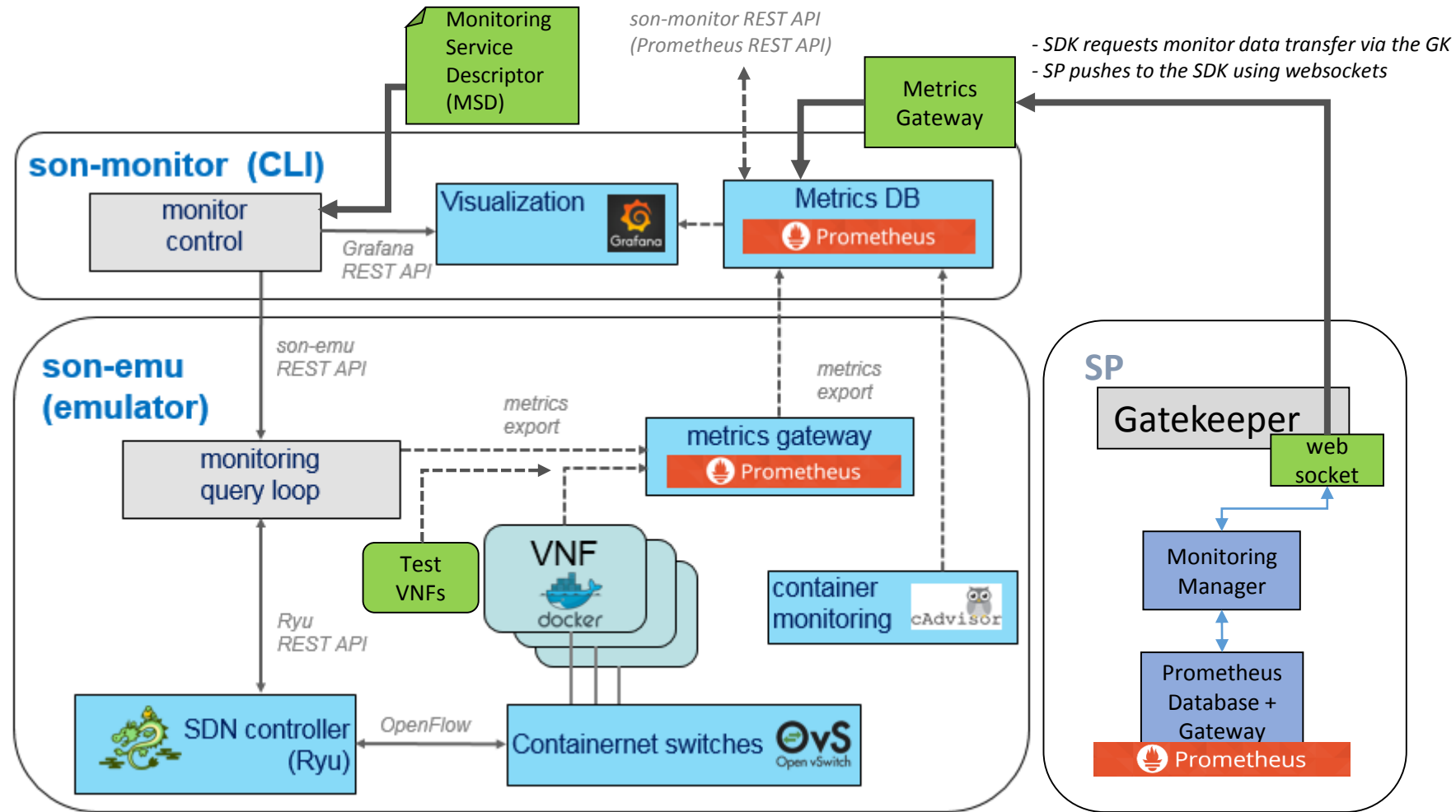
SONATA SDK architecture



Implementation based on:

- Combination of proven NFV/SDN related technologies (Mininet, OvS, Ryu, ...)
- Modular architecture with Python-based code that parses YAML-based service descriptors and deploys the chained VNFs.
- Can be deployed as isolated VM, combination of docker containers, debian install packages.

SONATA SDK architecture



SONATA SDK Virtual Machine

- use VirtualBox image to deploy the SONATA SDK VM
 - ssh sonata@localhost -p2222
 - several interfaces are exposed by the VM:
- This VM was created using Vagrant.
All related info will be shared via:
<https://github.com/sonata-nfv/son-tutorials>

son-editor (web gui)

<http://localhost:8080>

installation instructions on: <https://github.com/sonata-nfv/son-emu>

son-emu

installation instructions on: <https://github.com/sonata-nfv/son-emu>

<http://localhost:5001/dashboard/index.html>

<http://localhost:5000/restapi>

<http://localhost:8081>

<http://localhost:9091>

dashboard

rest api

cAdvisor

Prometheus Pushgateway

son-monitor

installation instructions on: <https://github.com/sonata-nfv/son-cli>

<http://localhost:3000>

<http://localhost:9090>

Grafana

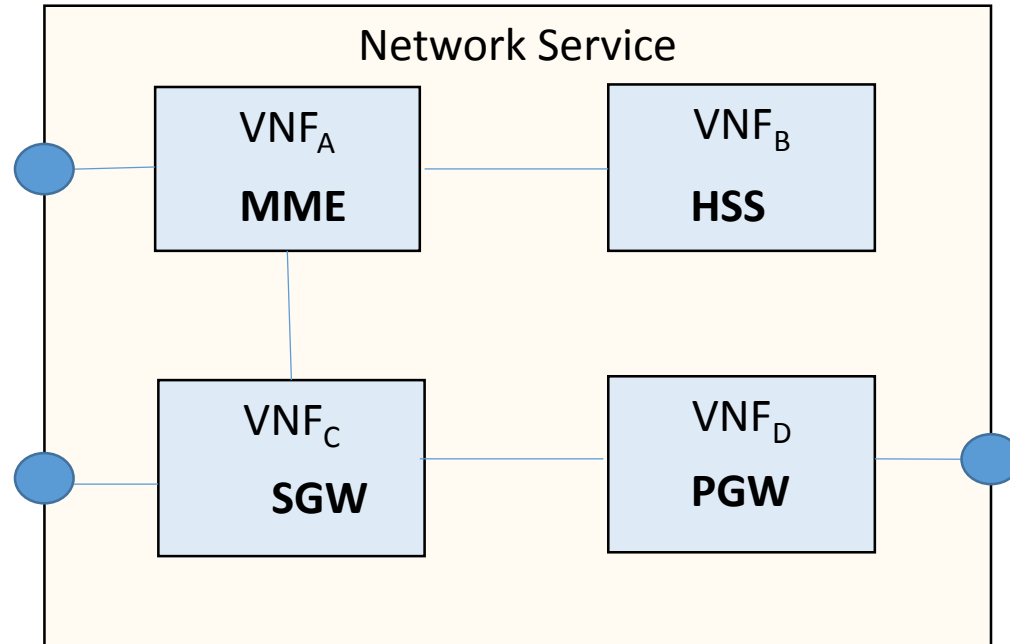
Prometheus

Agenda

- SDK architecture
- **Building the Service Descriptor**
- Testing the service
- Deploy the service in production



SONATA Descriptors



The JSON format of the SONATA descriptors can be seen in the editor:

<http://localhost:8080>

The schema and yaml format can be found on our GitHub:

<https://github.com/sonata-nfv/son-schema>

- A **Network Service** is specified in the Network Service Descriptor (NSD)



- A **Virtual Network Function** is described in a Virtual Network Service Descriptor (VNFD)



- The VNFD contains one or multiple **Virtual Deployment Unit (VDU)** templates that describe the Virtual Machines that run the virtual network function

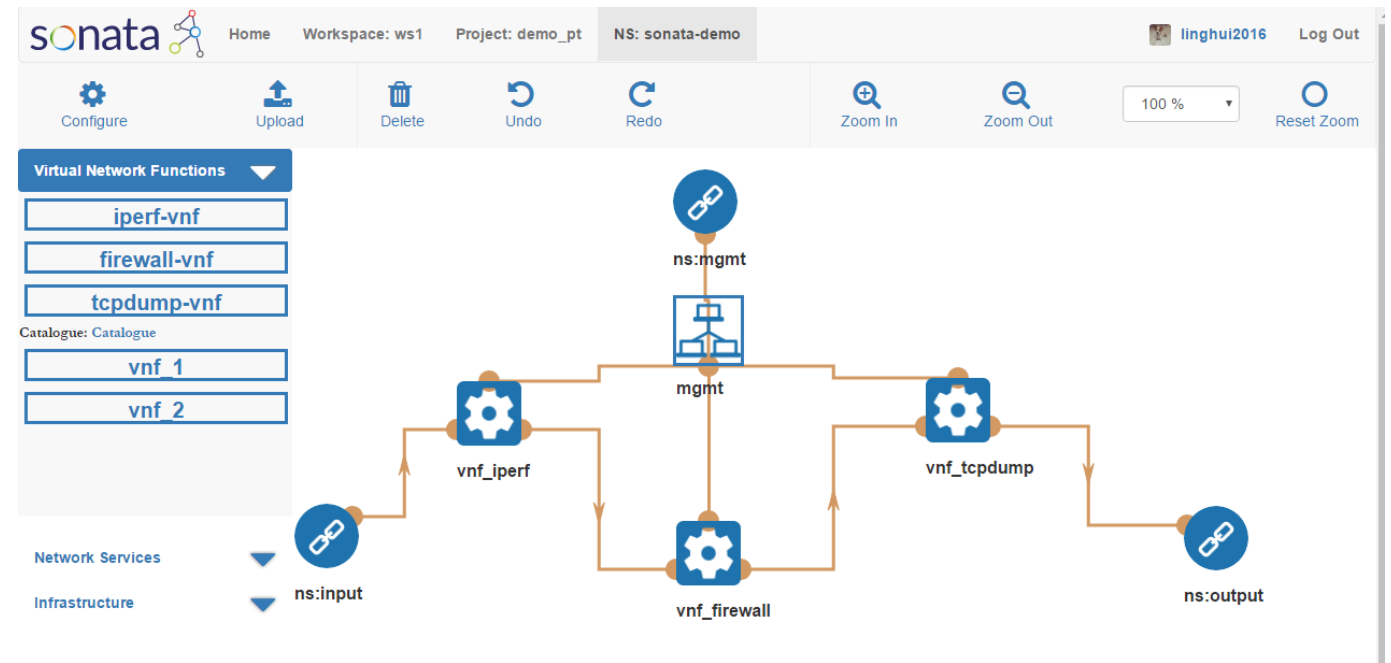
SONATA Descriptors

Open the editor:
<http://localhost:8080>

you can login with SONATA demo github user:
sonatademo
s0natademo

The editor allows to:

- create, edit descriptor files
- graphically check the descriptors
- upload the created service for deployment



Agenda

- SDK architecture
- Building the Service Descriptor
- **Testing the service**
- Deploy the service in production



SONATA SDK: service testing and debugging

start the emulator in terminal window:

```
ssh sonata@localhost -p2222
```

```
cd /home/ubuntu/son-emu
```

```
sudo python src/emuvim/examples/demo_topo_1pop.py
```

deploy a service from the editor

re-initialize the emulator with a new infrastructure topology


```
sudo python src/emuvim/examples/demo_topo_3pop.py
```

deploy a service from the editor

check the dashboard:

<http://localhost:5001/dashboard/index.html>

Emulator Dashboard



http://localhost:5001

Connect

Disconnect

Emulated Datacenters

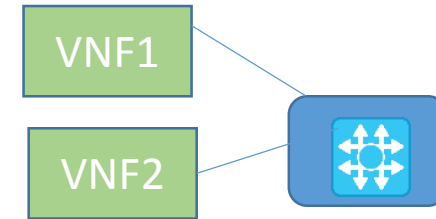
Label	Int. Name	Switch	Num. Containers	VNFs
dc1	dc1	dc1.s1	1	SDN, sgw1, pgw1, lme1, hss1

Latencies

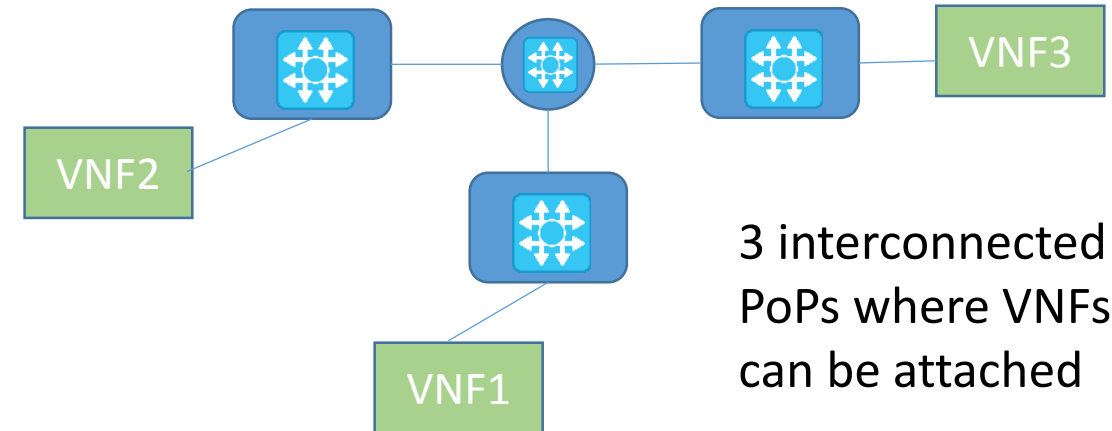
Running Containers

Datacenter	Container	Image	dockerID	Status
dc1	SDN	sdn-v2	375.371.8.8	Running
dc1	sgw1	sgw-v2	375.371.8.8	Running
dc1	pgw1	pgw-v2	375.371.8.5	Running
dc1	lme1	lme-v2	375.371.8.4	Running
dc1	hss1	hss-v2	375.371.8.7	Running

Latencies



1 PoP where VNFs can be attached



3 interconnected PoPs where VNFs can be attached

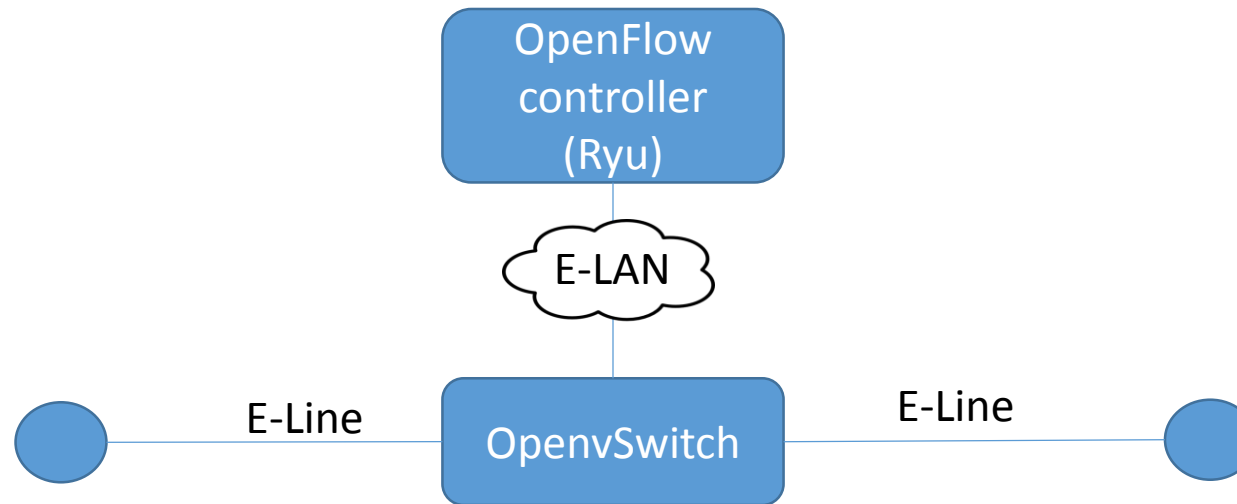
The emulator allows to:

- emulate a custom defined infrastructure topology
- deploy a SONATA service descriptor
- placement can be customized

SONATA SDK: simple example

project: **sonata-ovs-user-service-project**

(coming from <https://github.com/sonata-nfv/son-examples>)



Verify if the controller is really connected to the ovs instance in this service?

check the ip of the controller

check in the ovs instance:

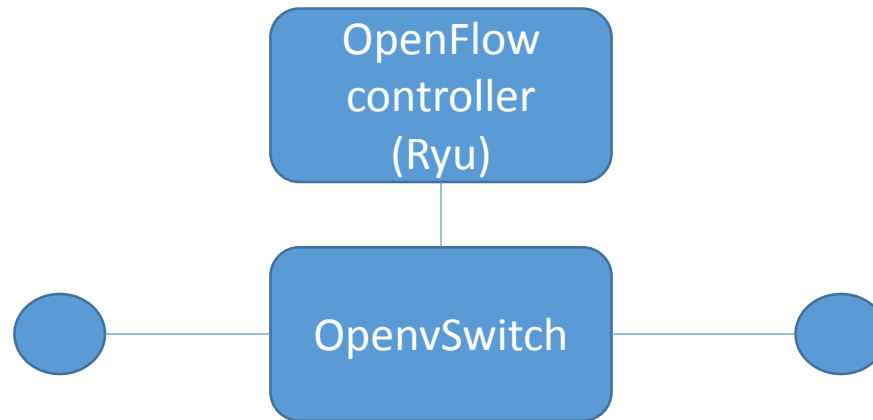
```
ovs1 ovs-vsctl show
```

```
ovs1 ovs-vsctl set-controller 'ovs1' tcp:<ip of controller>:6633
```

SONATA SDK: simple example

project: **sonata-ovs-user-service-project**

(coming from <https://github.com/sonata-nfv/son-examples>)



Test with generated traffic and monitor:

```
cd home/ubuntu/son-examples/service-projects/sonata-ovs-user-service-emu/
```

```
nano msd.yml
```

```
son-monitor msd -f msd.yml
```


SONATA SDK: service testing and debugging

Test scripts should be provided by the developer to test the service, eg. create traffic.

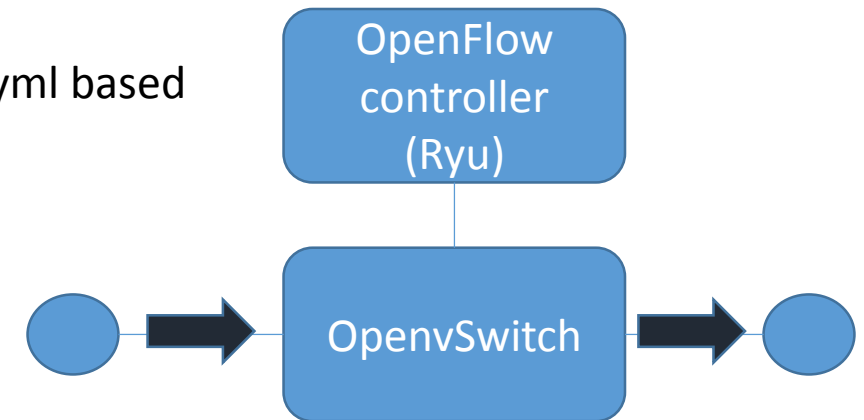
Each of the service's endpoints can be made available to external scripts.

A Service Access Point (SAP) in the emulator can be:

- A Docker container itself that does the traffic generation/analysis and is chained to the service
- A virtual interface on the host where traffic can be sent to and that is then chained to the service

During the testing a set of metrics to be monitored can be defined in a .yaml based descriptor file (msd.yaml)

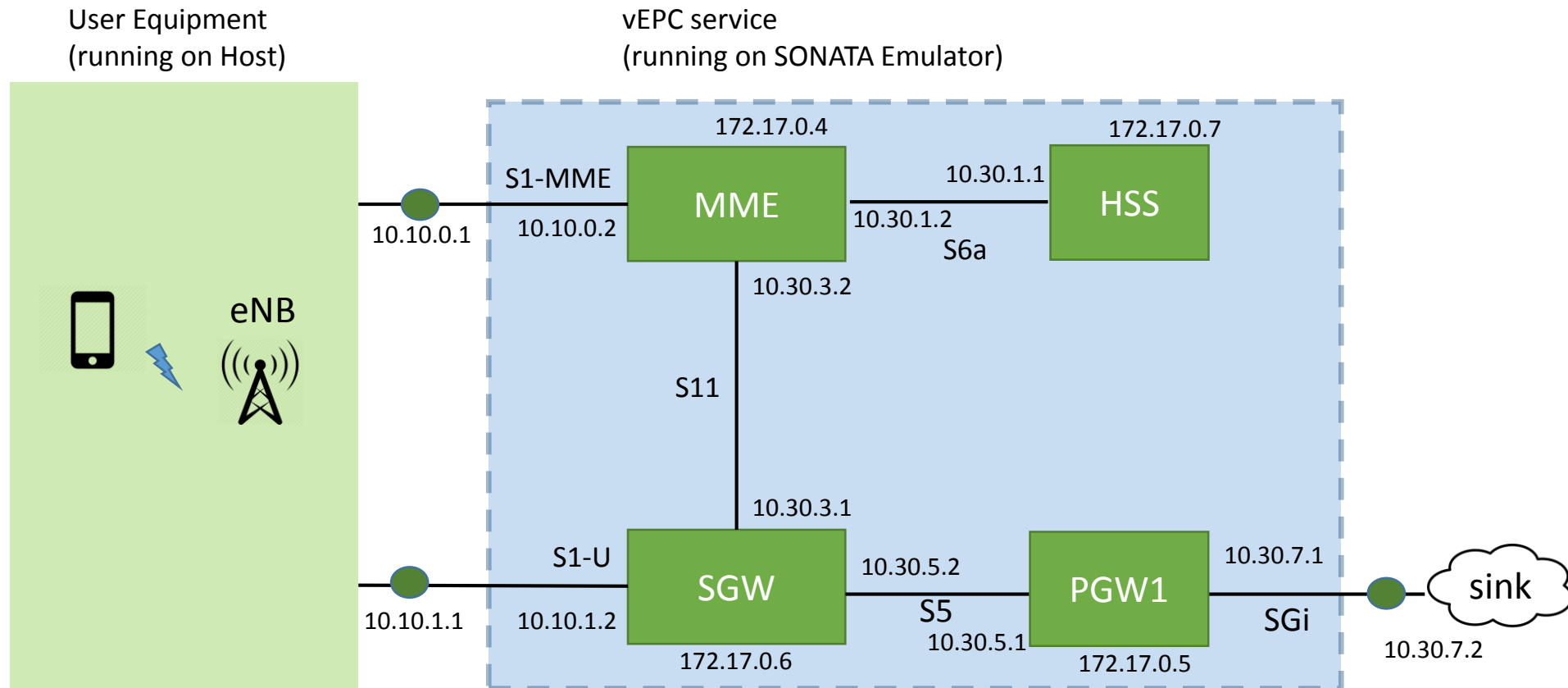
These metrics are exported and visualized in Grafana (*admin,admin*)



Test scripts and monitoring can also be combined and automated using a profile tool in the SONATA SDK

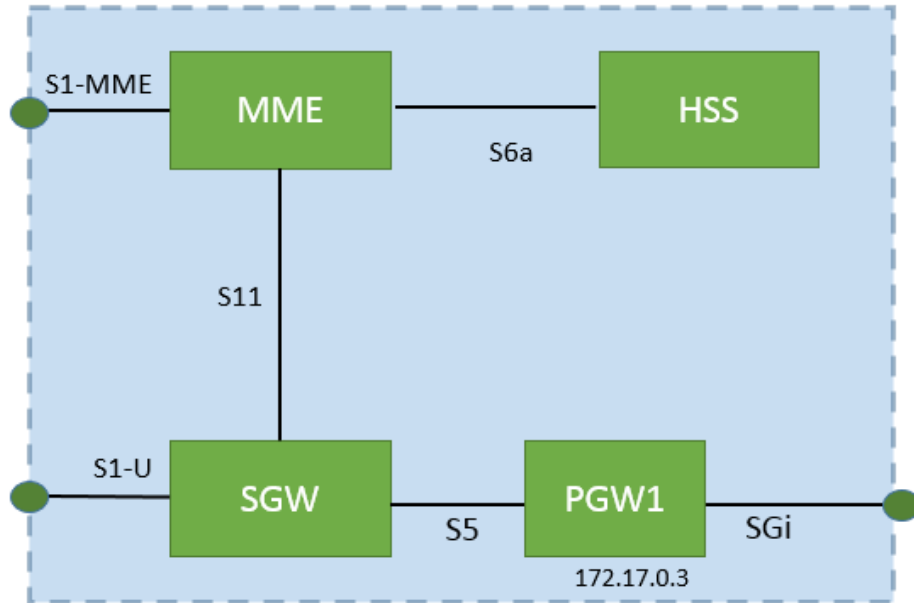
SONATA SDK: the vEPC service

Upload from the Editor:



SONATA SDK: service testing and debugging

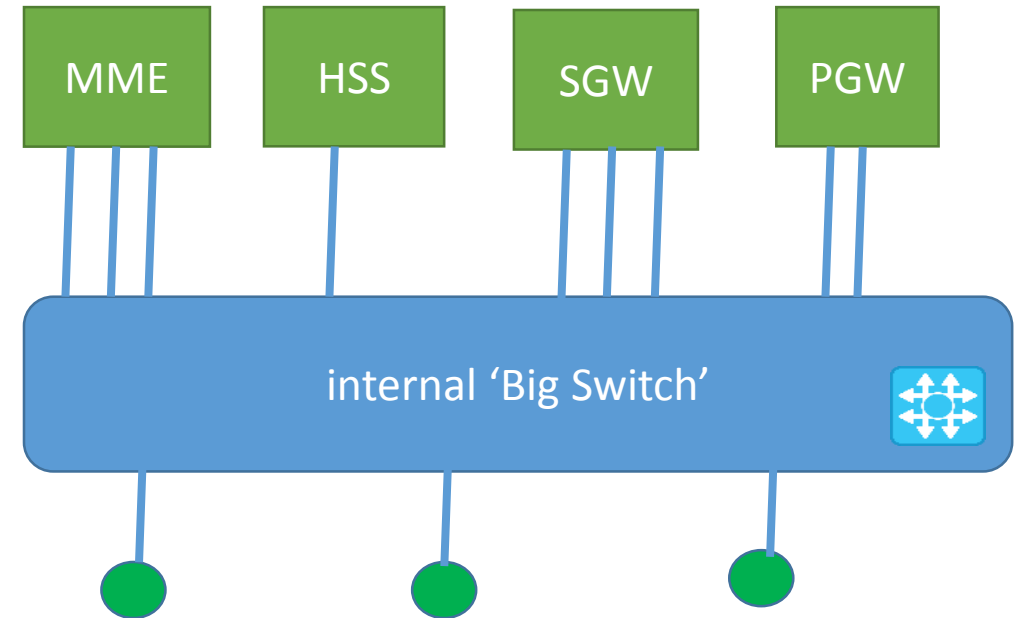
The deployed service is configured like this:



refresh the dashboard:

<http://localhost:5001/dashboard/index.html>

Mapped to the infrastructure that is emulated:



SONATA SDK: service testing and debugging

upload the service from the editor

or

via the terminal:

```
cd /home/Ubuntu/son-epc/
```

```
son-access -p emu push --upload vepc.son
```

The service is now deployed and ready to test.

What has happened:

- **the uploaded service descriptor is parsed by the emulator**
- **The VNFs are assumed to be available as Docker containers (check docker ps...)**
- **The VNFs get the number of interfaces as defined in their descriptors, and get an IP address**
- **The VNFs are chained together as defined in the service descriptor**

SONATA SDK: service testing and debugging

Test scripts and monitoring can be combined and automated using a profile tool in the SONATA SDK

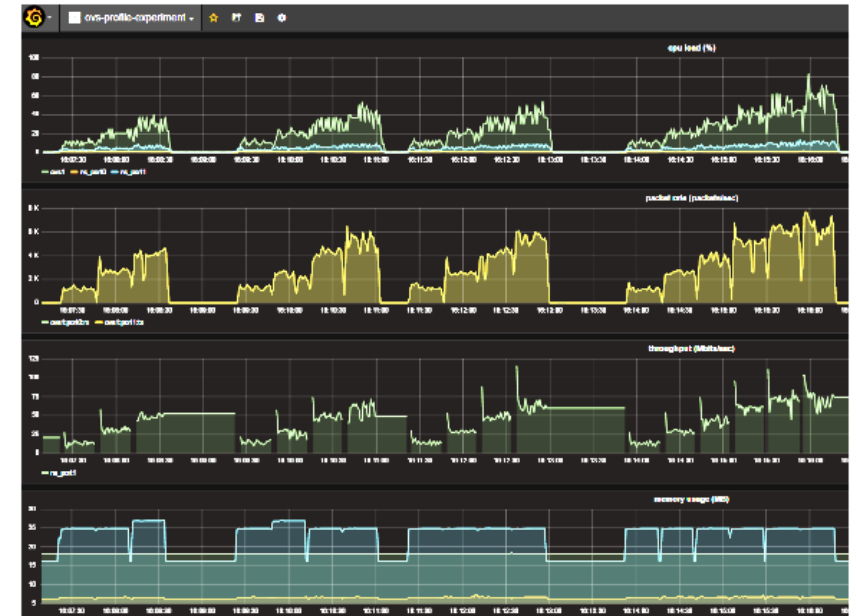
```
cd /home/ubuntu/son-epc/SOFTNETWORKING_vEPC/vepc-descriptor
```

emulate control traffic (user devices re-attaching to the vEPC):
`son-profile -c ped_ctrl.yml --no-generation --no-display`

emulate data traffic (users sending data):
`son-profile -c ped_data.yml --no-generation --no-display`

The monitor and profiling tools allow to:

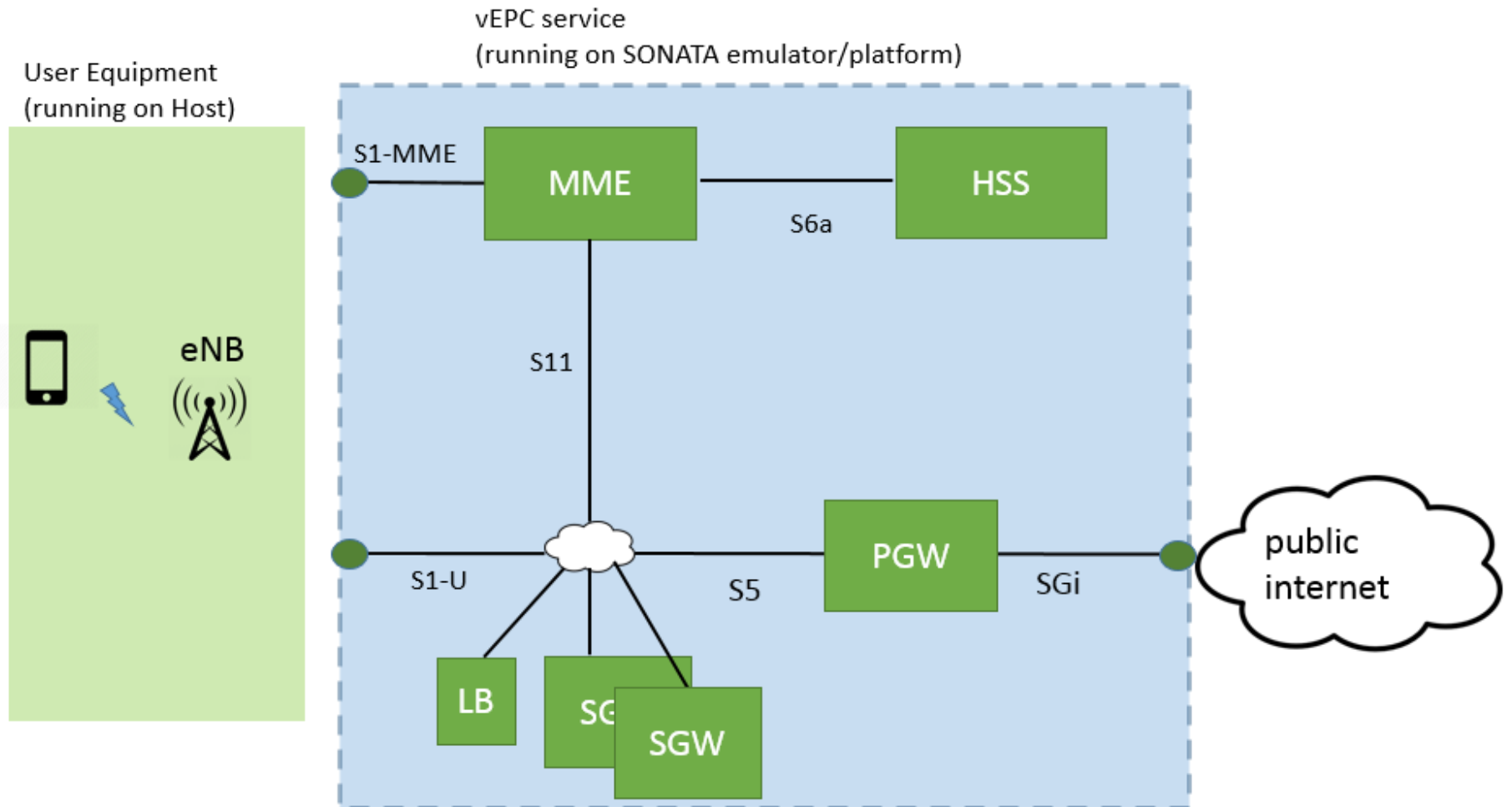
- automate the configuration of different traffic types/rates
- monitor a set of customizable metrics
- modify allocated resources (cpu/mem) to the VNFs



SONATA SDK: next steps for the vEPC service

The profiled performance of the VNFs in service allows a better resource and capacity planning.

-> optimized scaling in function of the loaded VNF



Agenda

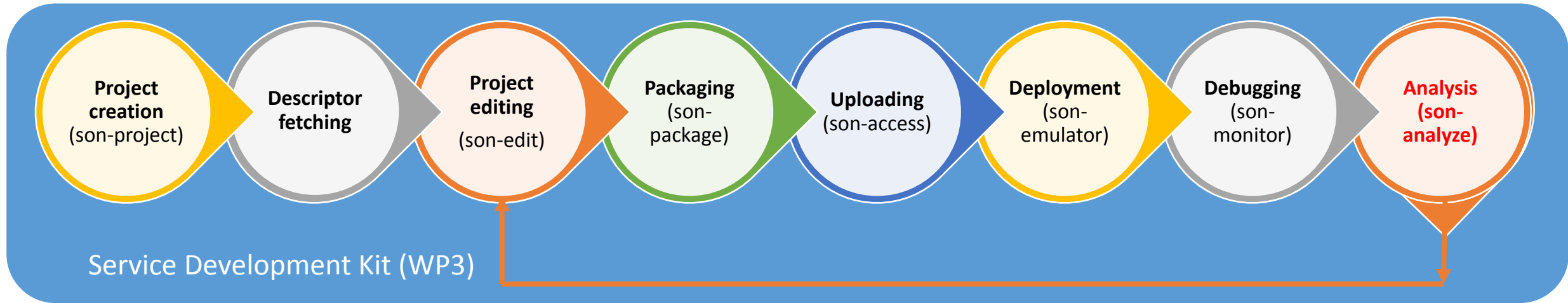
- SDK architecture
- Building the Service Descriptor
- Testing the service
- **Deploy the service in production!**



SONATA SDK: deploy the service to the SP

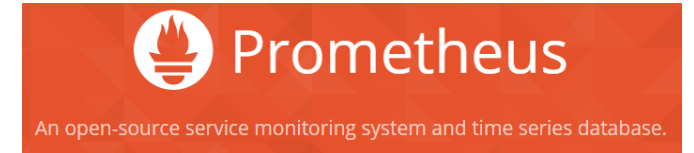
- The service developer has used the SONATA SDK to verify and debug the service:

A sandbox environment to try, test, debug... NFV-based services

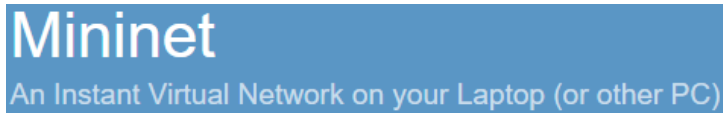


- A final package of the service can be generated and pushed to the Service Platform in production!

Technologies used



PostgreSQL



Where to find SONATA

sonata



Service Programming and Orchestration for
Virtualized Software Networks

The used software is open-source:

- Developed in the European SONATA research project
- Available on GitHub



<https://github.com/sonata-nfv>

Part of 5G-PPP
initiative:



HORIZON 2020:

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement no 671517.

Atos

Telefónica

UCL

SYNEXIS

NEC

NOKIA

iMinds

BT

PT INOVAÇÃO

UNIVERSITÄT PADERBORN

Optare Solutions

ubiwhere

THALES

Demokritos

i2cat



@sonataNFV

www.sonata-nfv.eu