sonata

agile service development and orchestration in 5G virtualized networks

The SONATA SDK: Rapid prototyping of network services using an integrated DevOps toolchain

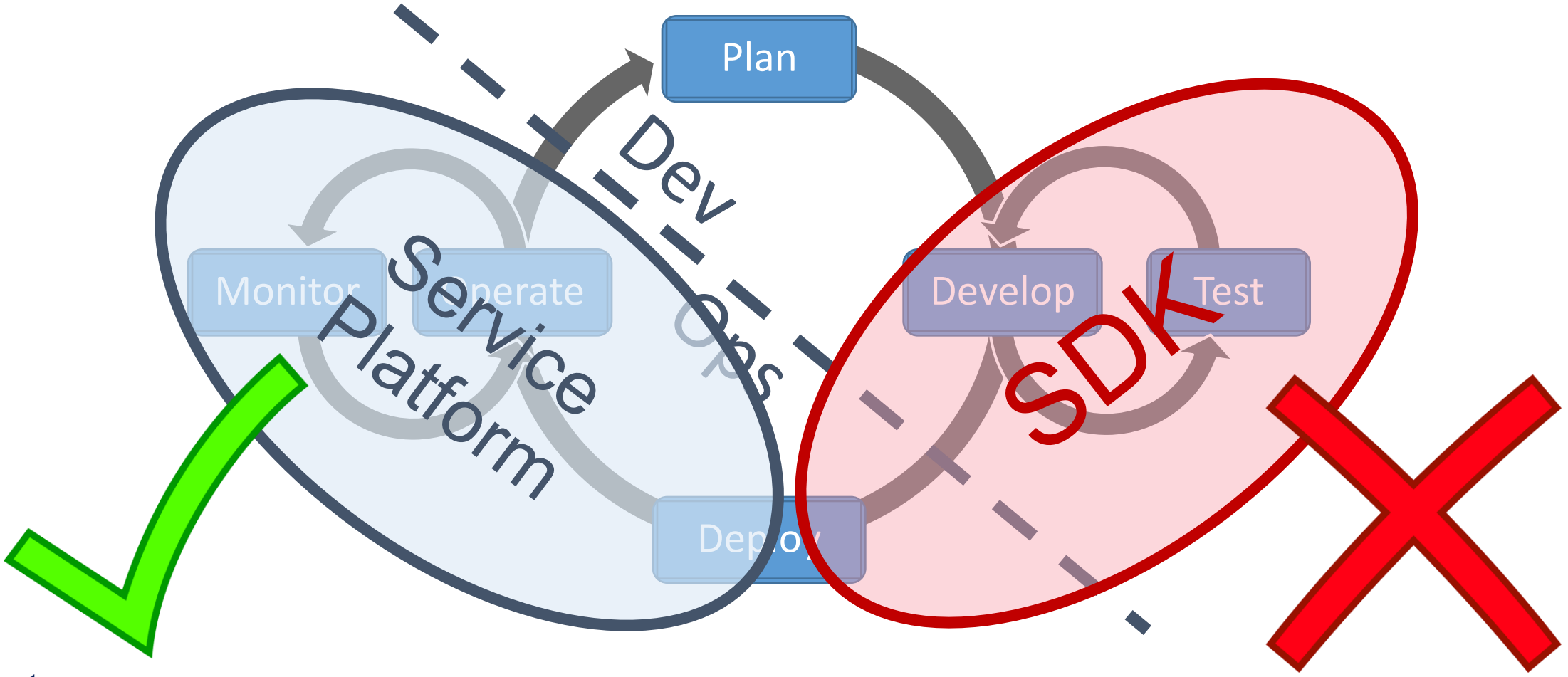
Manuel Peuster (Paderborn University)

PADERBORN UNIVERSITY

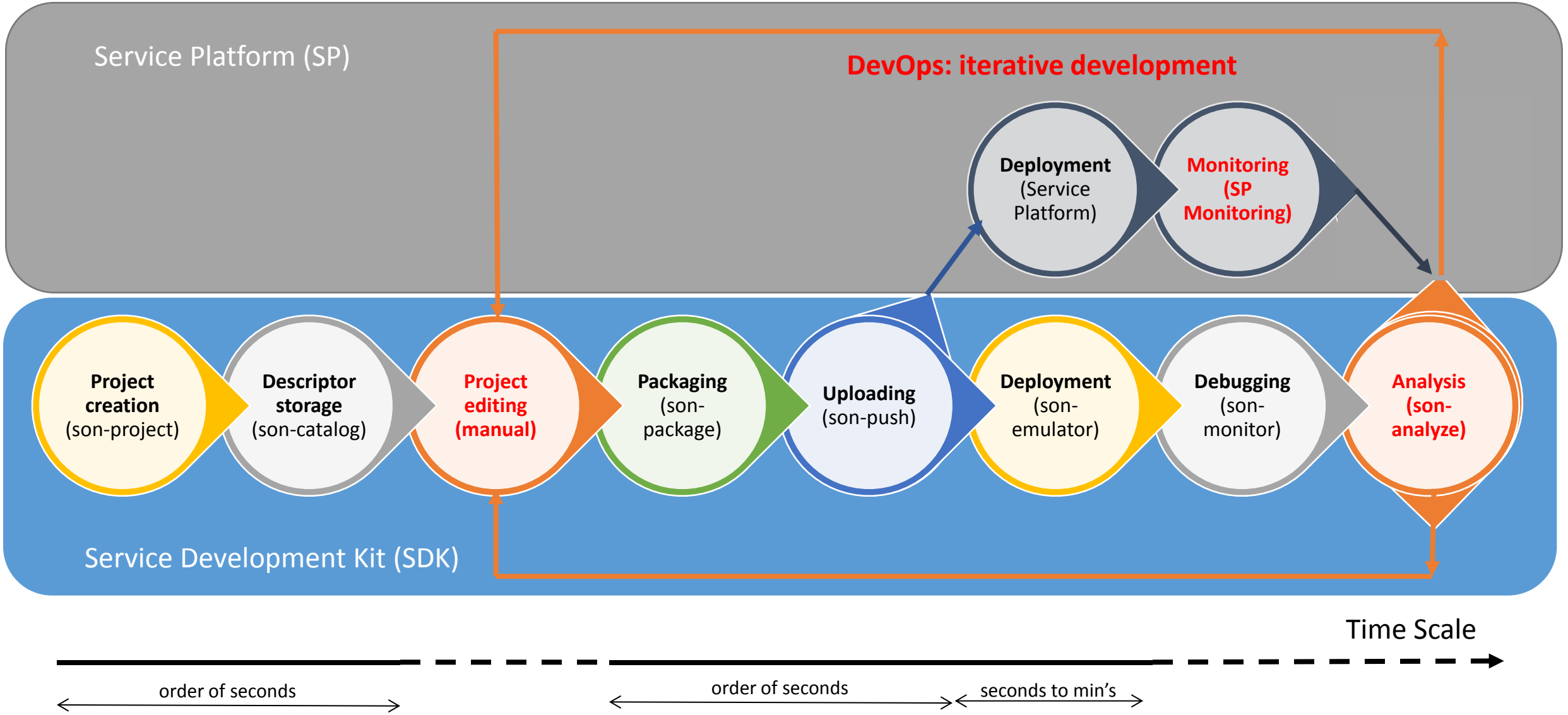
HORIZON 2020

5G PPP

SDK for NFV – Why?



Service Development Workflow



SONATA Descriptors

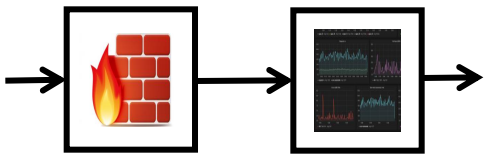
- Challenges addressed and solved by SONATA (so far)
  - **Identified** all the different **artifacts** that need to be references by all the different entities
    - VNFDs, PNFDs, VDUs, VLDs, etc.
  - Developed and implemented an **addressing scheme** that works across multiple domains
  - Incorporated other important features incl. **monitoring**

Virtual Network Function Descriptor

```
---
descriptor_version: "vnfd-schema-01"

vendor: "eu.sonata-nfv"
name: "firewall-vnf"
version: "0.3"

[...]

virtual_deployment_u
  - id: "vdu01"
    vm_image: "sonata
    vm_image_format:
    resource_requirem
      cpu:
        vcpus: 1

[...]
```

Network Service Descriptor

```
---
descriptor_version: "1.0"

vendor: "eu.sonata-nfv.service-descriptor"
name: "sonata-demo"
version: "0.2.1"

[...]

network_functions:
  - vnf_id: "vnf_firewall"
    vnf_vendor: "eu.sonata-nfv"
    vnf_name: "firewall-vnf"
    vnf_version: "0.3"

[...]
```



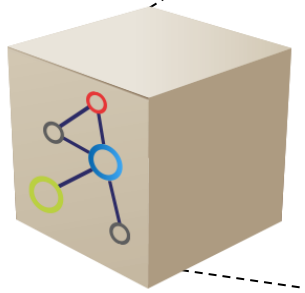
PADERBORN UNIVERSITY

sonata

Packaging – concept

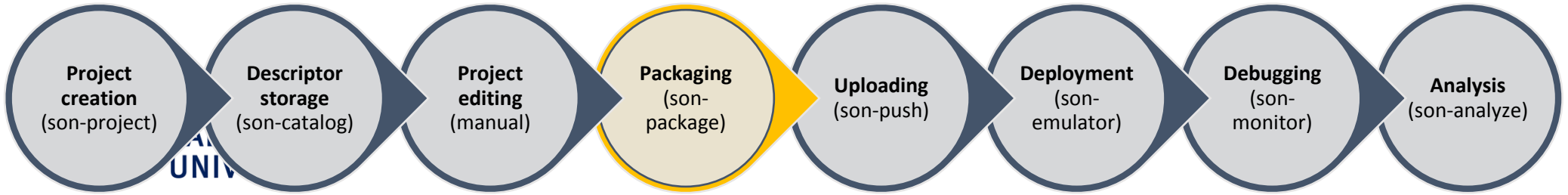
- Packages are the **primary artifact** in the SONATA ecosystem
  - Created using the SDK
  - Shipped to catalogs and the SONATA Service Platform

- SONATA supports slim and fat packages
  - A **slim** package contains references to artifacts
  - A **fat** package contains all the artifacs



Package Descriptor (Manifest.MF)

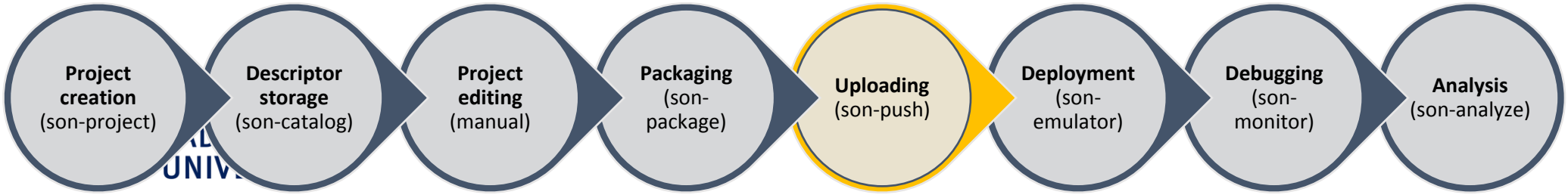
```
---
descriptor_version: "1.0"

vendor: "eu.sonata-nfv.service-descriptor"
name: "sonata-demo"
version: "0.2.1"

[...]

 package_content:
  - name: "/service_descriptors/sonata-demo.yml"
    content-type: "application/sonata.service_descriptor"
    md5: "0bb347f5bcd4986fccfc44ee87932c48"
[...]
```



Uploading – *son-push*

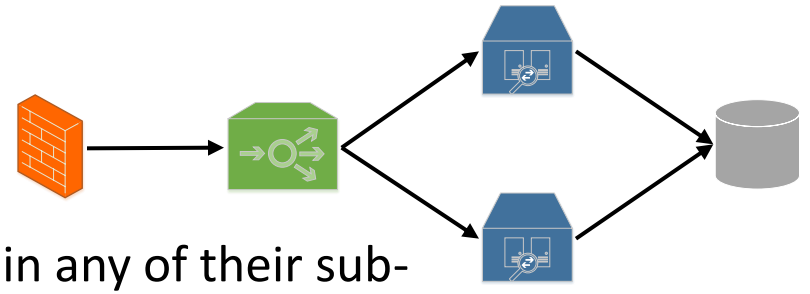
- **Uniform SDK interface** to either the Service Platform's Gatekeeper or the SDK emulator (emulated Service Platform)
- **Upload service packages** to Gatekeeper
- Obtain list of **available packages** from Gatekeeper
- View the list of **instantiated services**

Can be extended to support any Service Platform (using REST API)



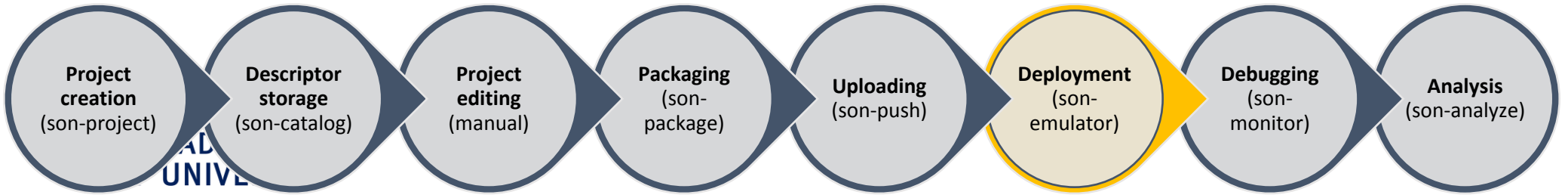
Deployment - *son-emu* motivation

- For a developer it is ...
  - **not easy** to validate their complex/distributed service chains
  - **non-trivial** to test them in realistic multi-PoP environments
  - **time consuming** to investigate effects caused by recent changes in any of their sub-components



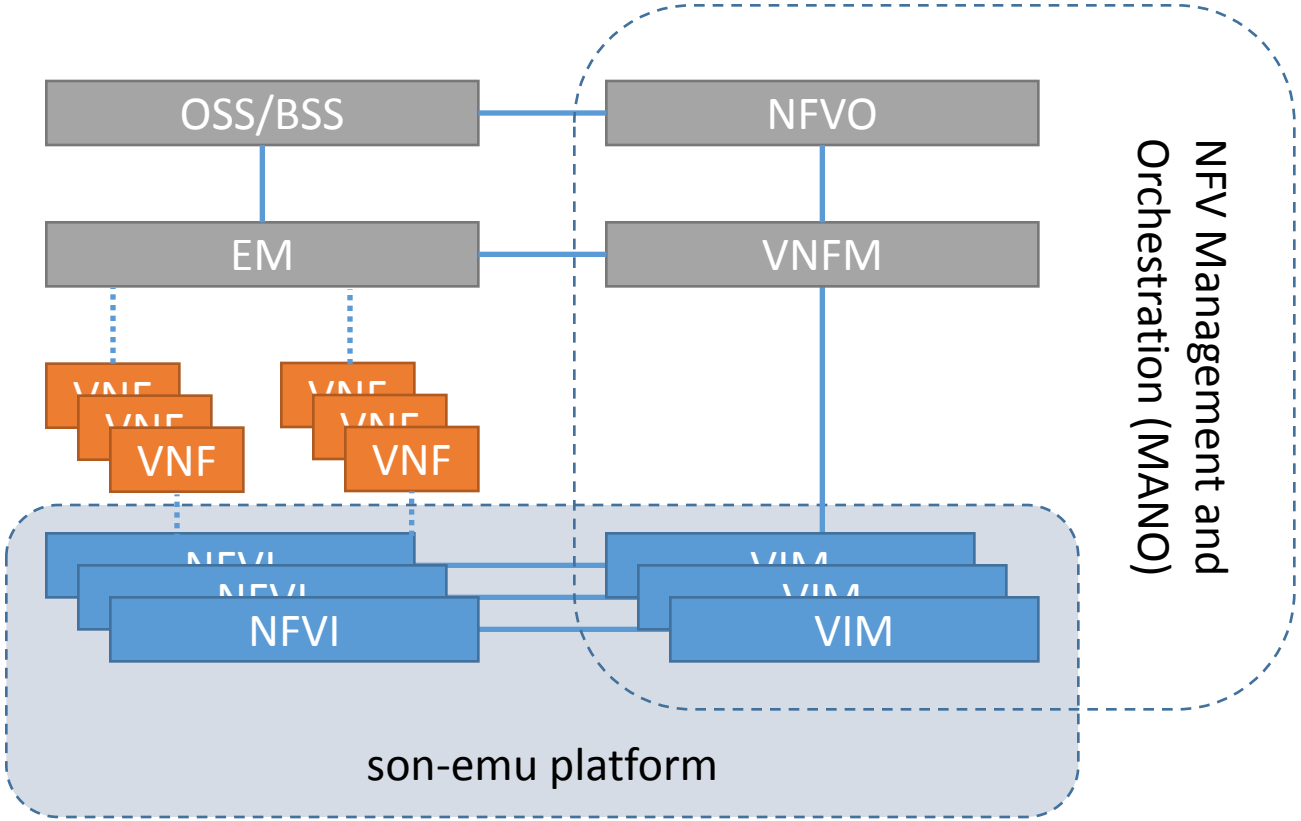
- Classic cloud **testbeds** are expensive and **require many resources**

- Develop platform **capable of emulating multiple NFVI PoPs on single laptop**
  - Executing VNFs as **containers**
  - **Enable quick iterations to test** complex service chains in well-defined setups
- Developers can **directly manipulate components** of the running service chain

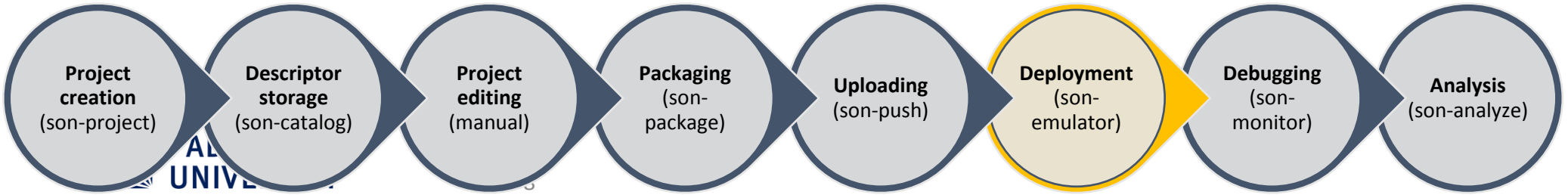


sonata

Deployment – *son-emu* scope vs. ETSI model

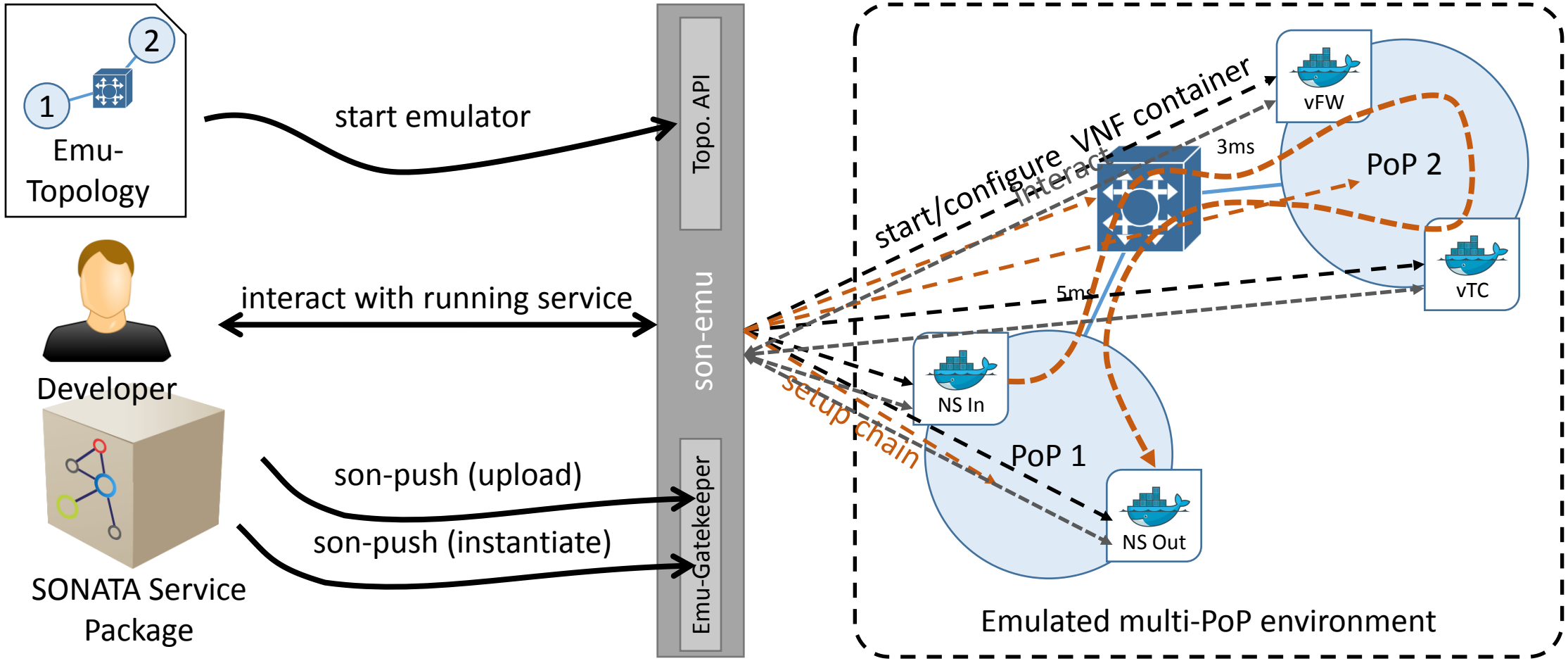


- Focus on **NFVI+VIM**
- **MANO *"batteries included"*** as "Dummy Gatekeeper"
  - Offers interface similar to WP4 service platform' gatekeeper
- Architecture:
  - **Modular** design based on Containernet[1]
  - **Flexible APIs** to extend/customize the platform
  - Interactive **CLI**



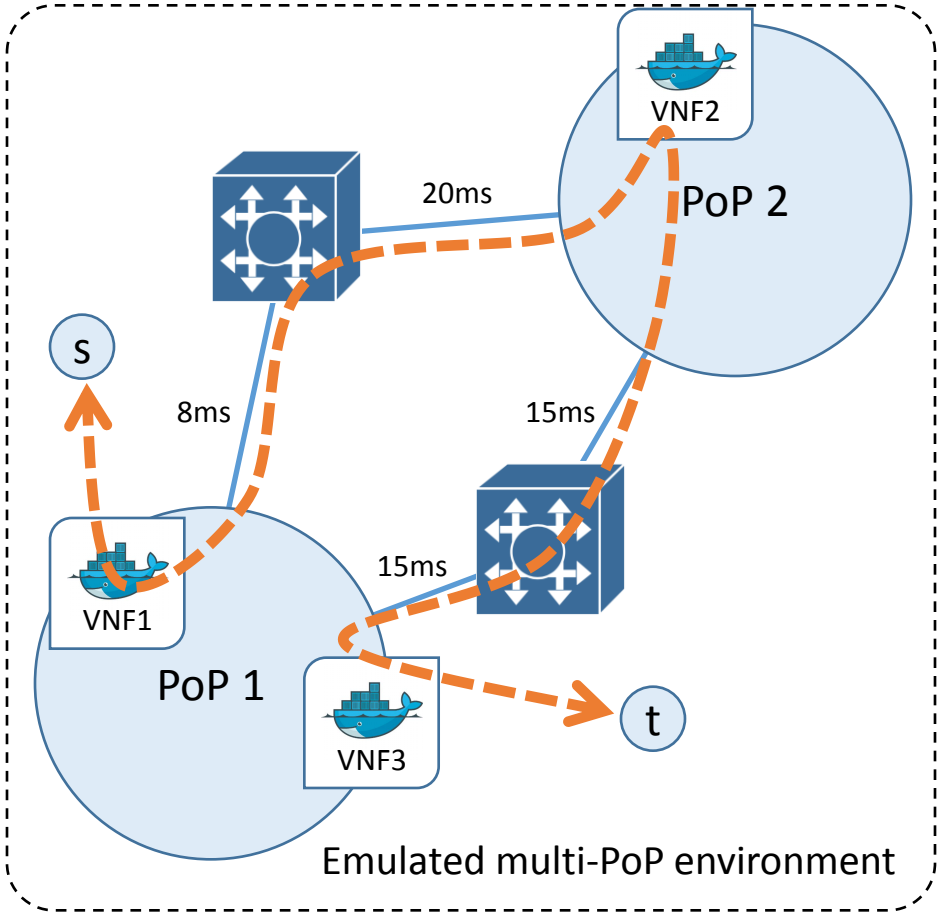
sonata

Deployment – *son-emu* workflow



Deployment – *son-emu* features

- Interactive, cloud-like **rapid prototyping** platform
- Executed on **developer's laptop**
- **Production-ready** network services
- Executes **container-based** VNFs
- Custom **multi-PoP** topologies
- **Service Chaining** support
- **MANO** system **integration**
- Integrates with **SONATA workflow** and tools

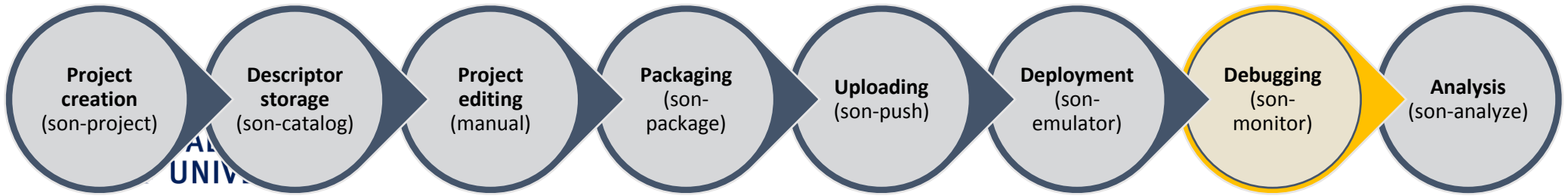


Project creation (son-project) → **Descriptor storage** (son-catalog) → **Project editing** (manual) → **Packaging** (son-package) → **Uploading** (son-push) → **Deployment** (son-emulator) → **Debugging** (son-monitor) → **Analysis** (son-analyze)

Debugging – SDK *son-monitor* objectives

- Newly developed service might contain **bugs**
  - Functionality bugs (not working as intended)
  - Performance bugs
- Newly **developed service might be not well understood**

- son-monitor = range of **debugging and profiling** features to investigate
- Enable a more **efficient development and deployment workflow** for a network service.
- Provide a **dataset of metrics** that further analysis (e.g., workload prediction, scaling thresholds). Metrics might be pre-processed (aggregated, etc.)



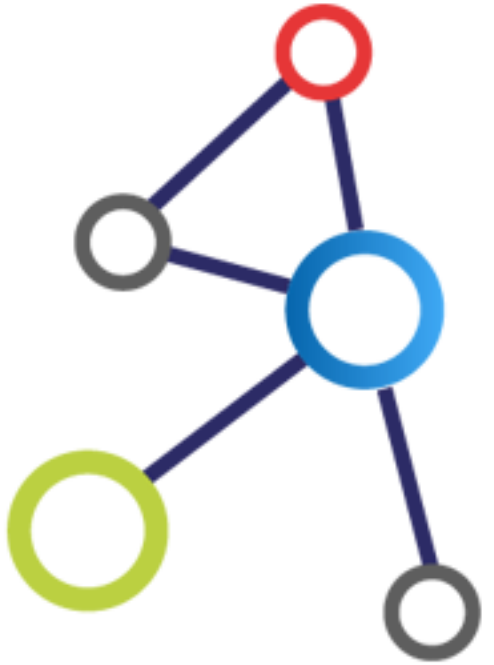
Conclusions

- One of the **first open-source**, vendor-independent SDK for NFV
- **SDK emulator environment** enabling developers to **rapidly validate and deploy** the developed service in multi-PoP setting on a limited local environment (e.g., laptop)
- **Working prototype, open-source code base** with clear **roadmap** available
- **Effective integration operations with service development (DevOps)** for 5G

https://github.com/sonata-nfv

sonata

agile service development and orchestration in 5G virtualized networks

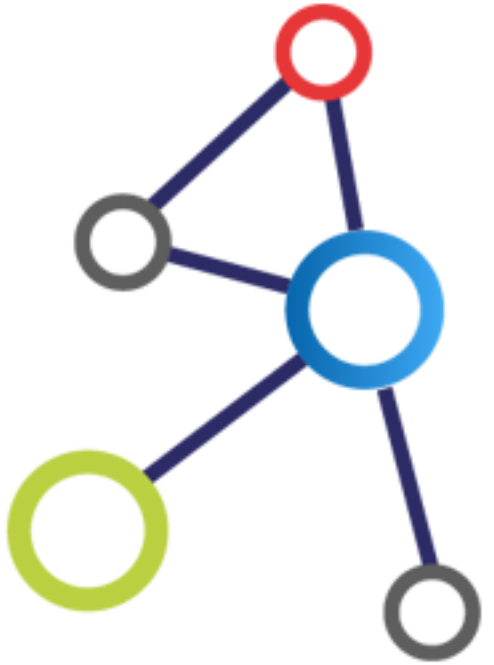


Thank you!

http://sonata-nfv.eu

sonata

agile service development and orchestration in 5G virtualized networks



Backup

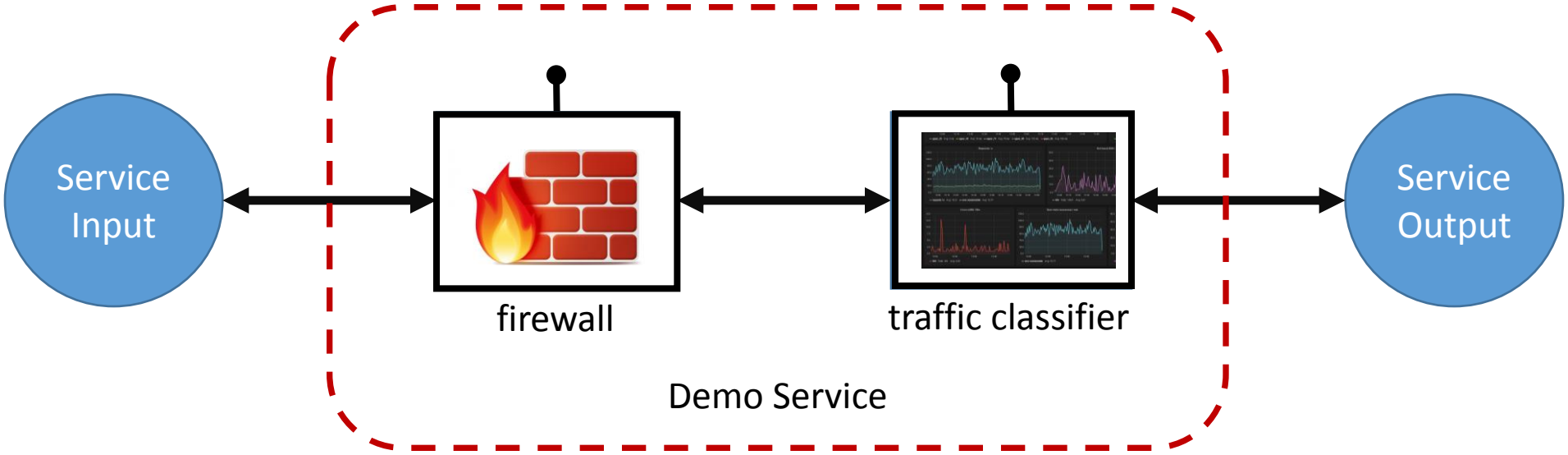
SDK demo overview

Perspective during this demo: Network service developer!

1. Show how a SONATA network service project looks like
2. Present network service and function descriptors
3. Show interaction with SDK catalogue and service packaging
4. Use SONATA's emulator to test the entire service locally
5. Manipulate the service at runtime and validate the results

Used demo service

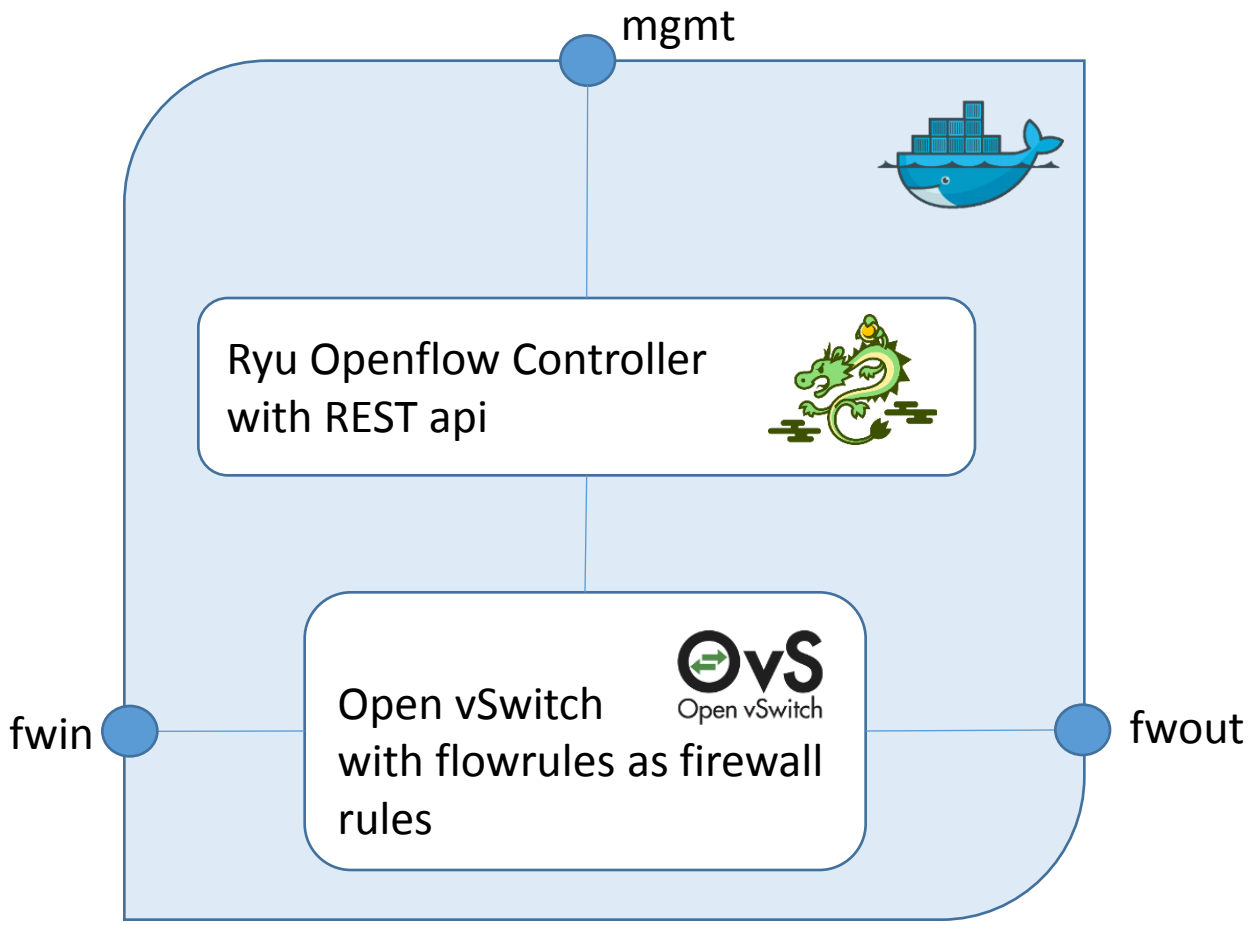
- Virtual firewall (vFW) + virtual traffic classifier (vTC)
- Traffic generator (input) + traffic sink (output)



https://github.com/sonata-nfv/son-examples/tree/master/service-projects/sonata-fw-vtc-service-emu

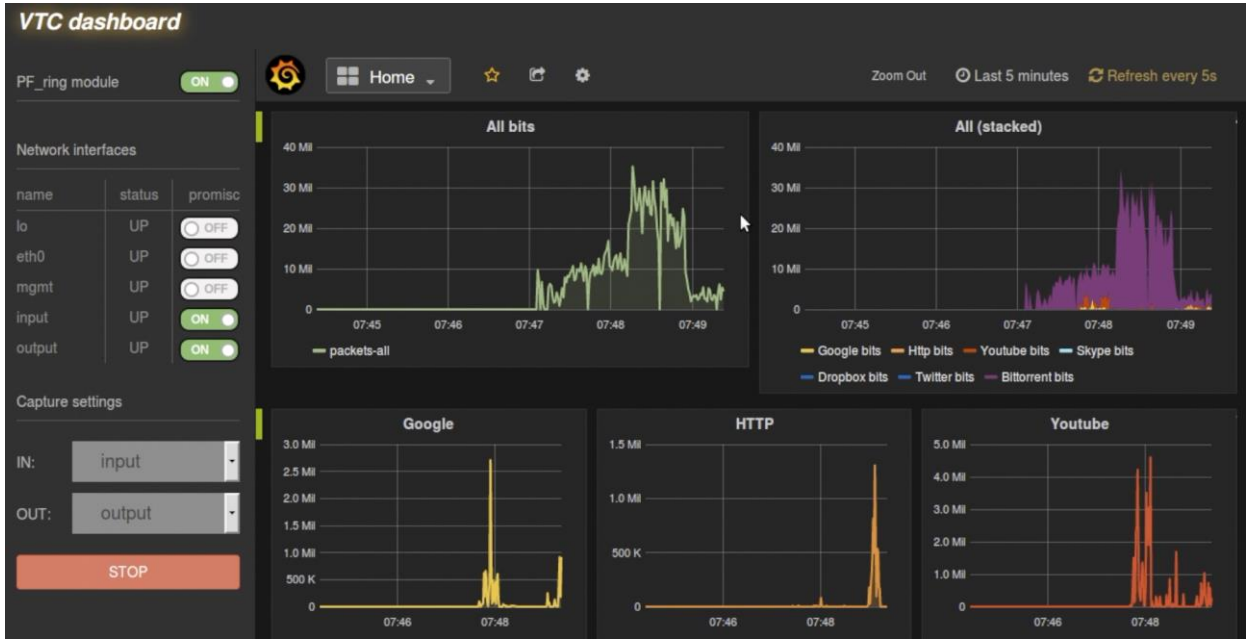
Virtual firewall (vFW)

- Based on:
  - Open vSwitch
  - Ryu controller
- Offers REST interface to reconfigure blocking rules
- Interfaces
  - Input (fwin)
  - Output (fwout)
  - Management (mgmt)



Virtual traffic classifier (vTC)

- DPI-based traffic classification
- Based on PF_RING module
- Offers interactive dashboard
- Interfaces:
  - Input
  - Output
  - Management

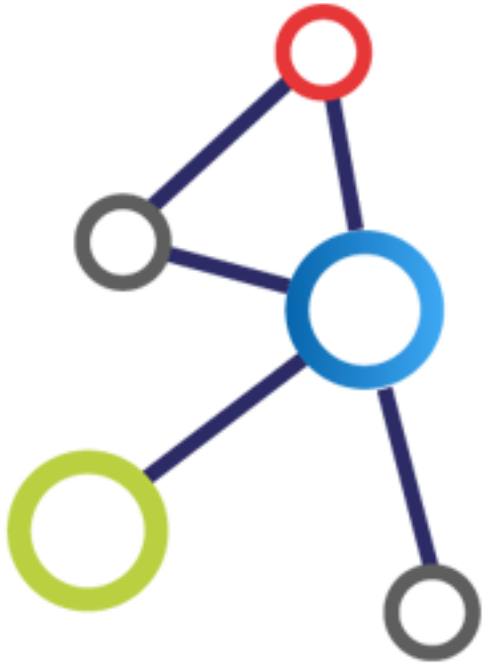


Involved SONATA components

- SONATA Example Services (available on GitHub)
- SONATA Descriptors
- SONATA Catalogues
- SONATA CLI
  - son-publish
  - son-package
  - son-push
  - son-monitor
- SONATA Emulator
  - Dummy Gatekeeper
  - Son-emu-cli tools
  - Interactive emulator interface

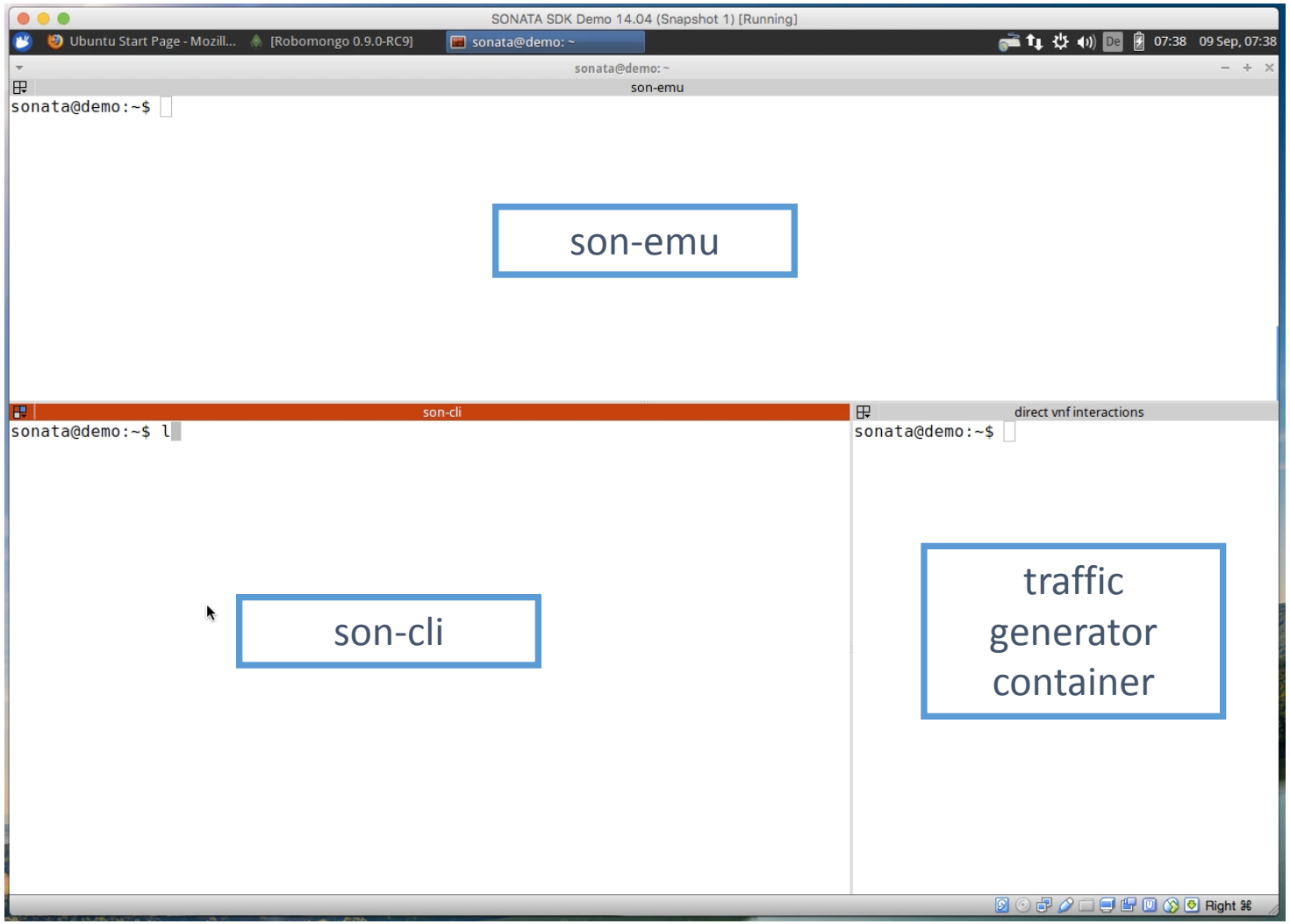
sonata

agile service development and orchestration in 5G virtualized networks

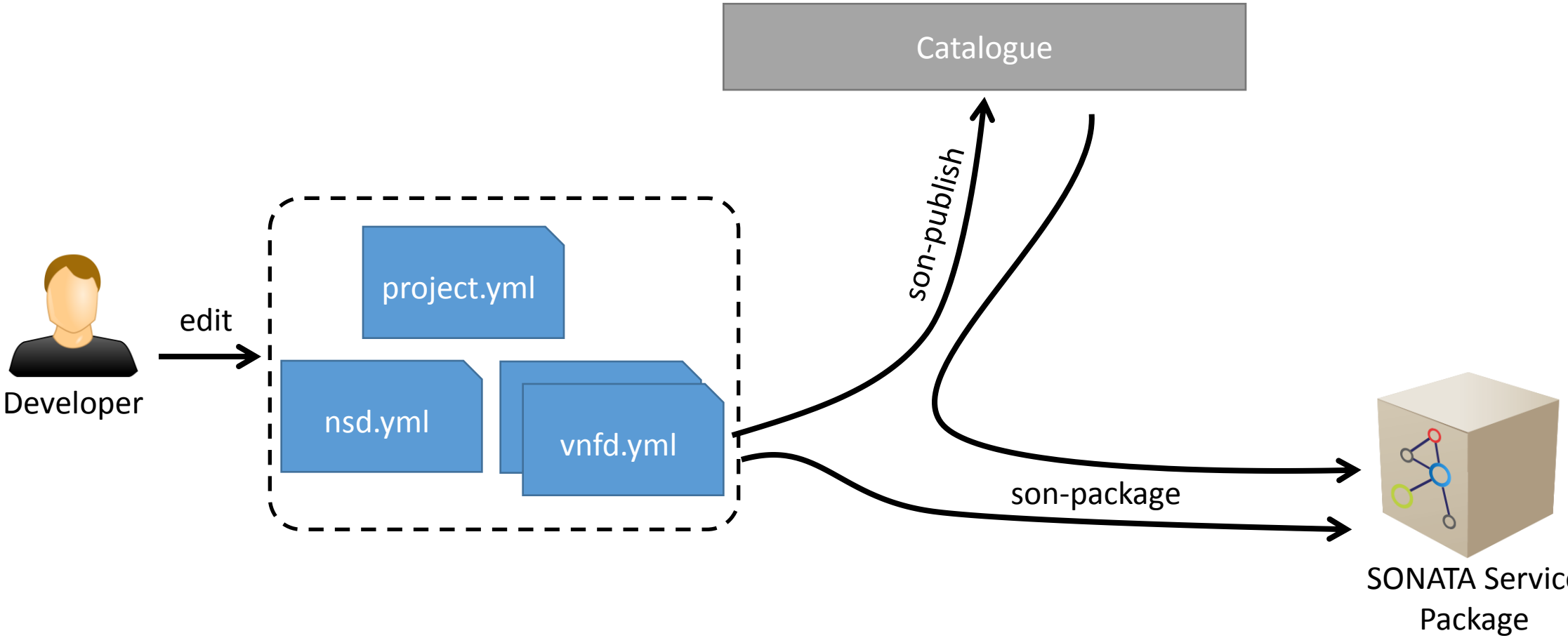


Demo

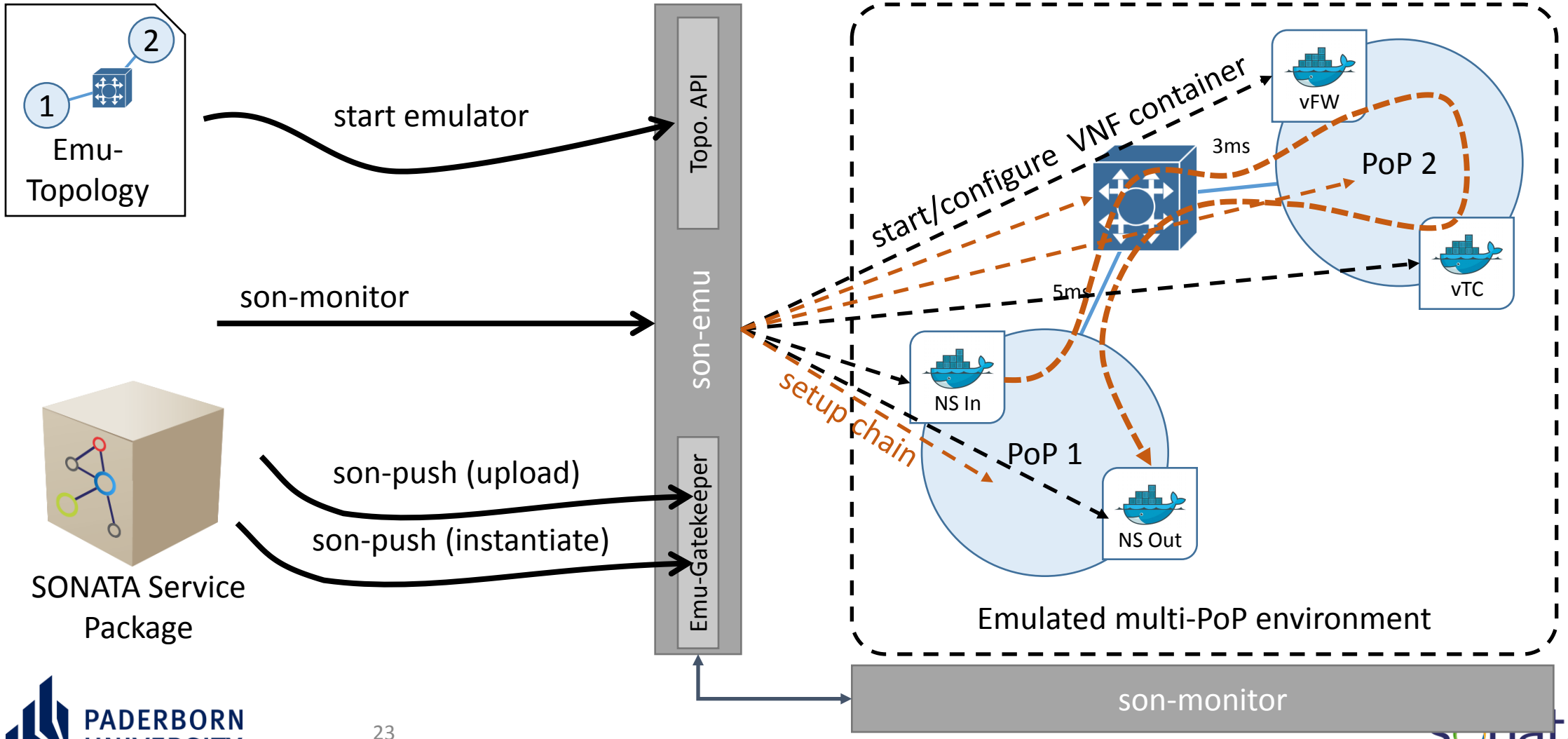
Terminal setup



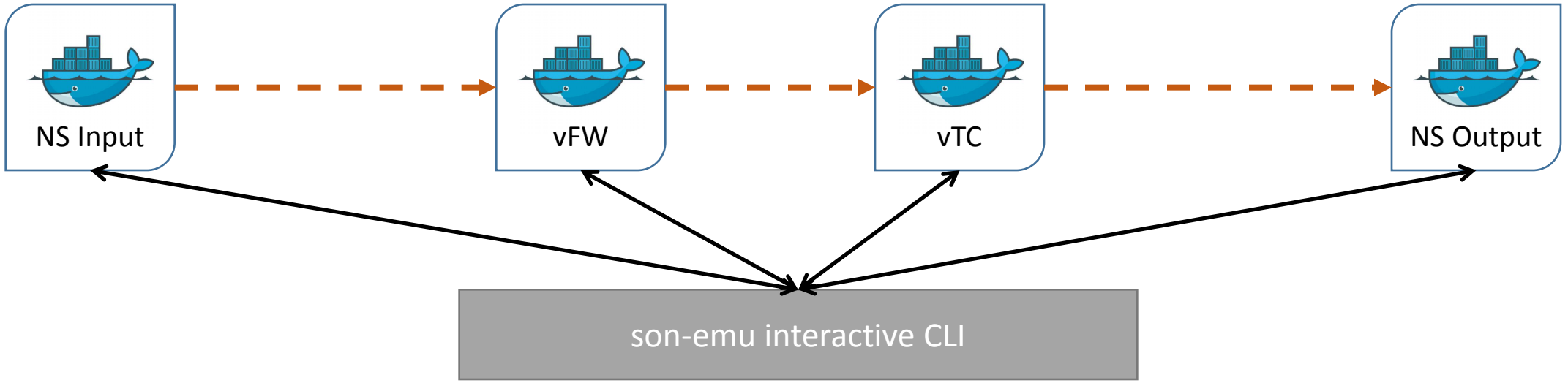
Part 1: Modify and package NS project



Part 2: Deploy on emulator

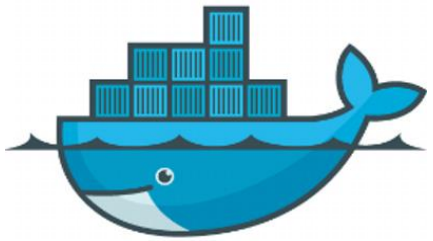


Part 3: Interact with running service

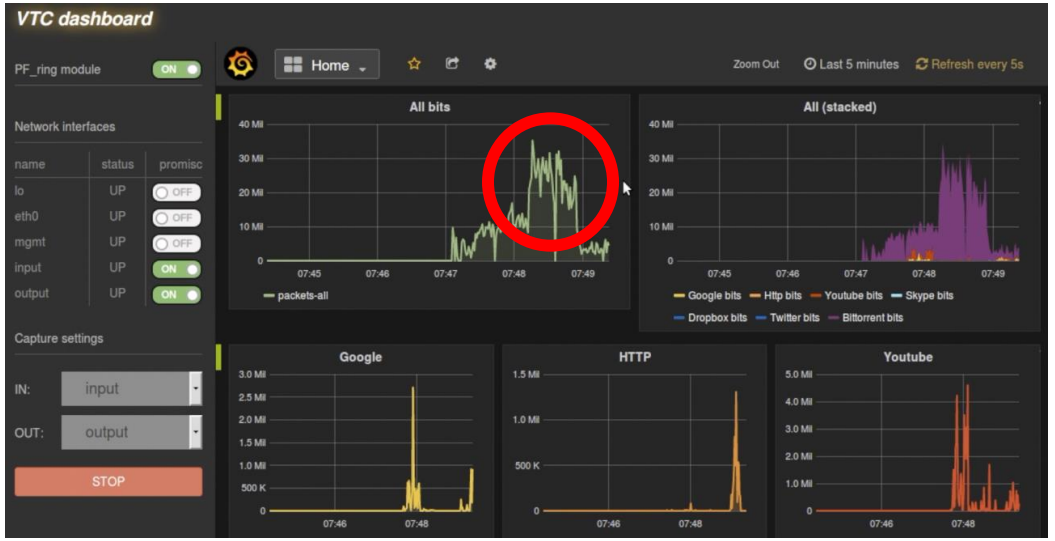


- Direct interaction with container's bash, e.g.:
  - fw_vnf ifconfig
  - vtc_vnf ifconfig
  - ns_servicein ping –c10 ns_serviceout

Part 4: Generate traffic and simulate attack



NS Input



- Attach to ns_input container
- Replay PCAP traffic trace
- Run iperf traffic to simulate attack

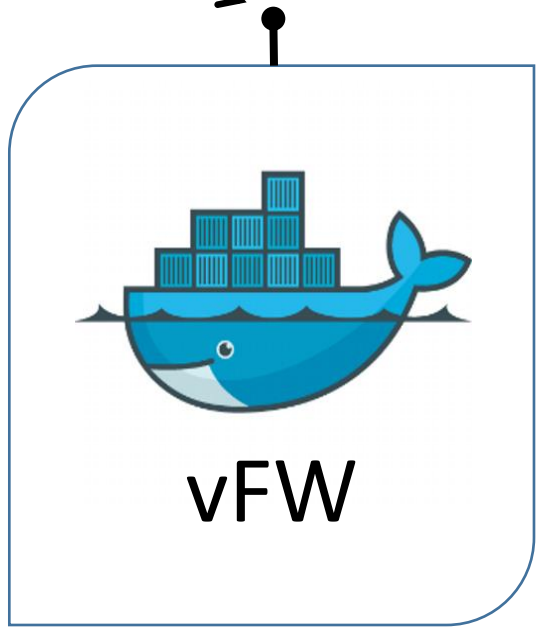
- Monitor traffic bursts in vTC dashboard and son-monitor dashboard

Part 5: Reconfigure vFW to block attack

```
curl -X POST http://172.17.0.10:8080/stats/flowentry/add \
  -v -d '{"dpid": 1, "cookie": 200, "priority": 1000, \
  "match": {"dl_type": 0x0800, "nw_proto": 17, "udp_dst": 5001}}'
```

- Reconfigure vFW at runtime
- Goal: Block iperf traffic

- Validate the outcome of the reconfiguration directly after it was applied
  - vTC dashboard
  - son-monitor dashboard



Part 6: Upload the package

SONATA Service Package

son-push (upload)

SONATA Service Platform (WP4)

- Service is fine! Let’s upload it to our production platform!
- Use son-push to upload it to the service platform
- Continuation in WP4 demo …

Key takeaways

- User friendly management of network service projects
- Integrated catalogue service to re-use 3rd party VNFs
- Fully automated packaging procedure
- Novel multi-PoP emulation platform
- Extensive monitoring support to simplify failure detection
- Rapid prototyping workflow incl. manipulation of VNFs at runtime

Fully integrated service development kit for NFV!

What is shown?

1. Open and edit an existing SONATA network service project
2. Publish descriptors to SDK catalogues
3. Package the network service project
4. Upload the service package to the emulation platform
5. Run the network service in an emulated environment with 2 PoPs

What is shown?

1. Access dashboard of vTC VNF to monitor it
2. Use son-monitor to monitor metrics of the entire network service
3. Replay PCAP traffic file to test the service chain
4. Simulate DDoS attack (additional iperf traffic)
5. Reconfigure vFW **at runtime** to block the unwanted traffic