From Zero to Orchestrated

Introduction

Agenda

- **8:00 – 8:30 Introduction**
- 8:30 – 9:00 Bootstrapping of the development environment
- 9:00 – 9:30 Details of the Orchestrator domain models
- 9:30 – 10:00 Break
- 10:00 – 11:30 Development of your first Orchestrator workflow
- 11:30 – 13:00 Lunch
- 13:00 – 14:30 Integration of OSS and BSS to your workflow
- 14:30 – 16:00 Tailoring the Orchestrator to your needs (Discussion)

From Zero to Orchestrated

Introduction

Who do we have in the room?

To get to know each other please do the following:

- State your name
- Affiliation
- What you hope to achieve during this workshop.
- What knowledge and/or experience you bring with you from your home institution.

What is the goal of this workshop?

Have a reasonable understanding of the following concepts in the workfloworchestrator:

- Service modelling: Subscriptions, Products and Domain models
- The workflow engine and the anatomy of a workflow and it's steps
- Have an idea of how to integrate an OSS/BSS software and how you can define sources of truth
- Learn from other NREN's and discuss Automation and Orchestration use cases
- Create an understanding of how the workfloworchestrator ecosystem of software could help achieve an NREN's orchestration goals.

Automation

The automatic operation or control of equipment, a process, or a system. This often encompasses a linear process.

Orchestration

The execution of (multiple) automations to achieve the desired state of a process or system.

Why Automation & Orchestration?

Eliminate repetitive
& time consuming tasks

Prevent human
mistakes

Customer dashboard

Up-to-date
service lifecycle

Enable self service

Orchestration by example….

What does the orchestrator solve in the Real World®? A SURF Example:

- Reliable data administration
- Fast data administration
- Accurate data administration
- Reliable service provisioning
- Fast service provisioning
- Accurate service provisioning
- Single pane of glass on all services provisioned on SURF’s (inter)national network.

Orchestration in the context of the workfloworchestrator

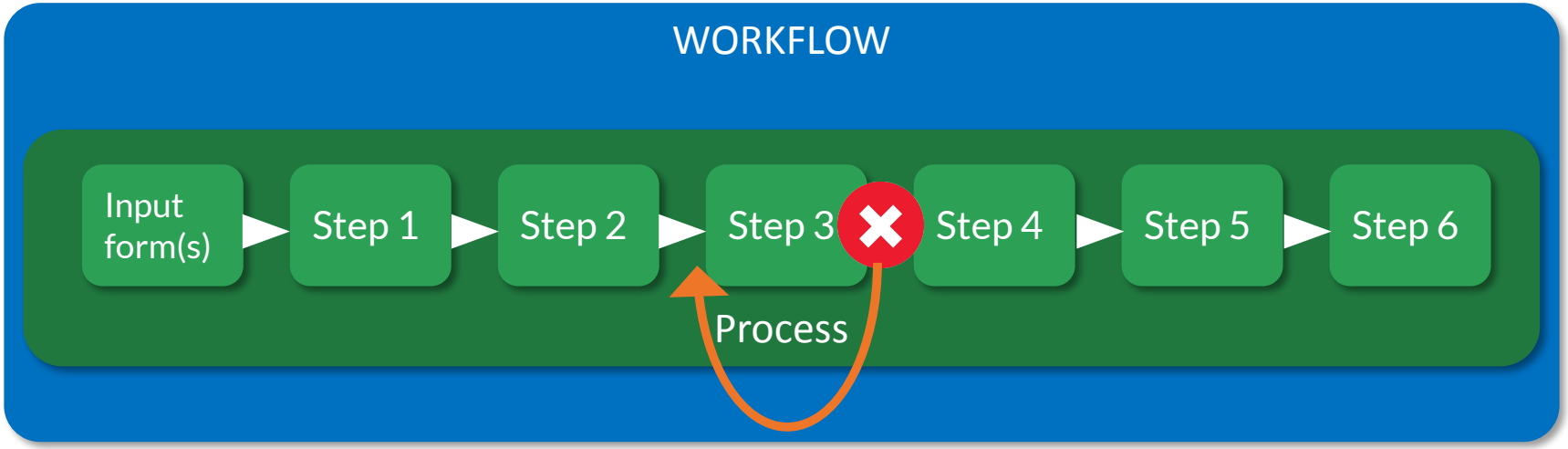
- It executes arbitrary python functions on objects and stores the result of each function in the database. These functions are called steps.
- A collection steps that follow on each other are called workflows
- Workflows can be run to execute arbitrary tasks, but…
  - are usually run on products and/or subscriptions.
- Workflows; create, modify, terminate and validate subscriptions and automate lifecycle tasks
- This creates an audit-trail for each subscription so you can see all actions that have been executed on each subscription. We call these processes.

The workflow engine orchestrates automations.

Orchestration Concepts

- Orchestration is executed on higher order abstractions
- The goal is to not only achieve valid network configuration, but to make it possible to define relations between all the data that is needed to provision an arbitrary service
  - Inventory
  - Customer data
  - IP address management
  - Ticketing
  - Planning
- Modelling of abstractions tries create logical relationships between resources that are necessary to provision a service (of any type). The orchestration then makes it possible to define the state of each resource during the lifecycle of a subscription.

Workflow Engine

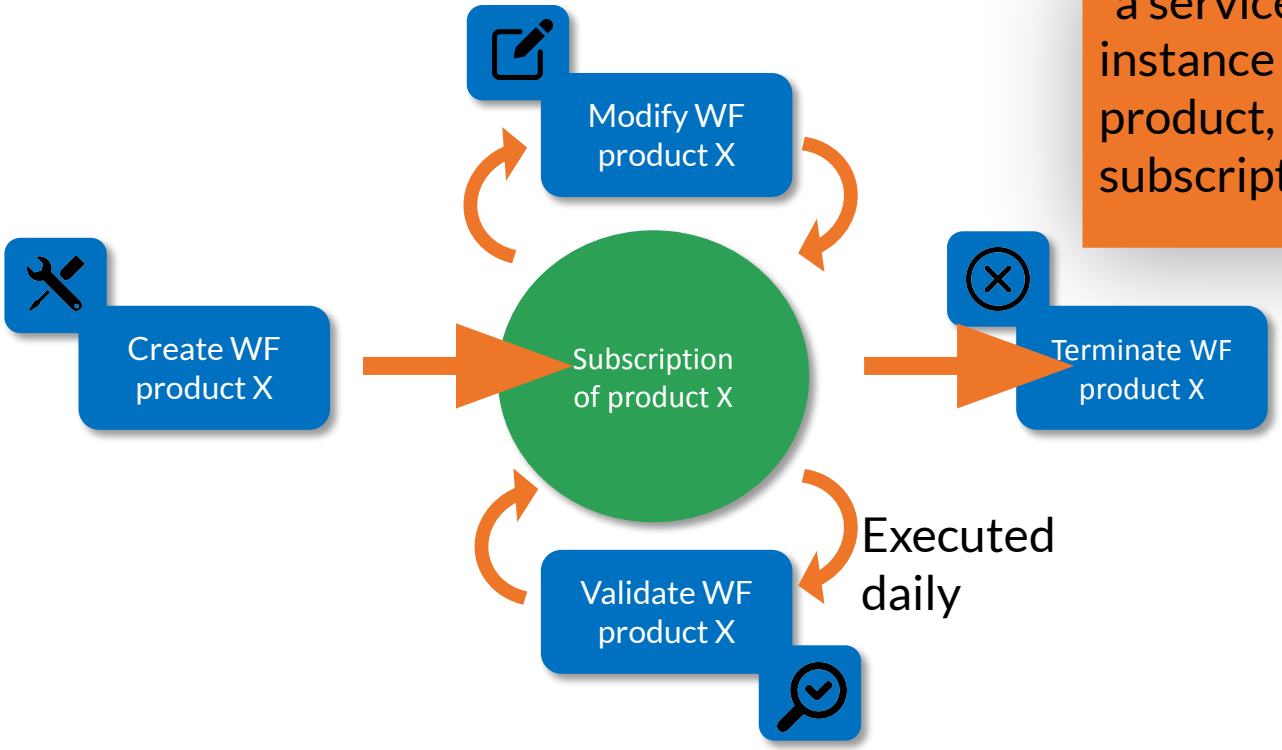


- Each Step writes the state to the database and is used as input for the next step
  - Each (atomic) Step can be retried, making the workflow robust

Workflow Code

```python
@create_workflow("Create SURFnet8 L2VPN", initial_input_form=initial_input_form_generator)
def create_sn8_l2vpn() -> StepList:
    return (
        begin
        >> construct_l2vpn_model
        >> store_process_subscription(Target.CREATE)
        >> create_ims_circuit
        >> create_nso_service_model
        >> re_deploy_nso
        >> take_ims_circuit_in_service(is_redundant=False)
        >> send_confirmation_email()
    )
```

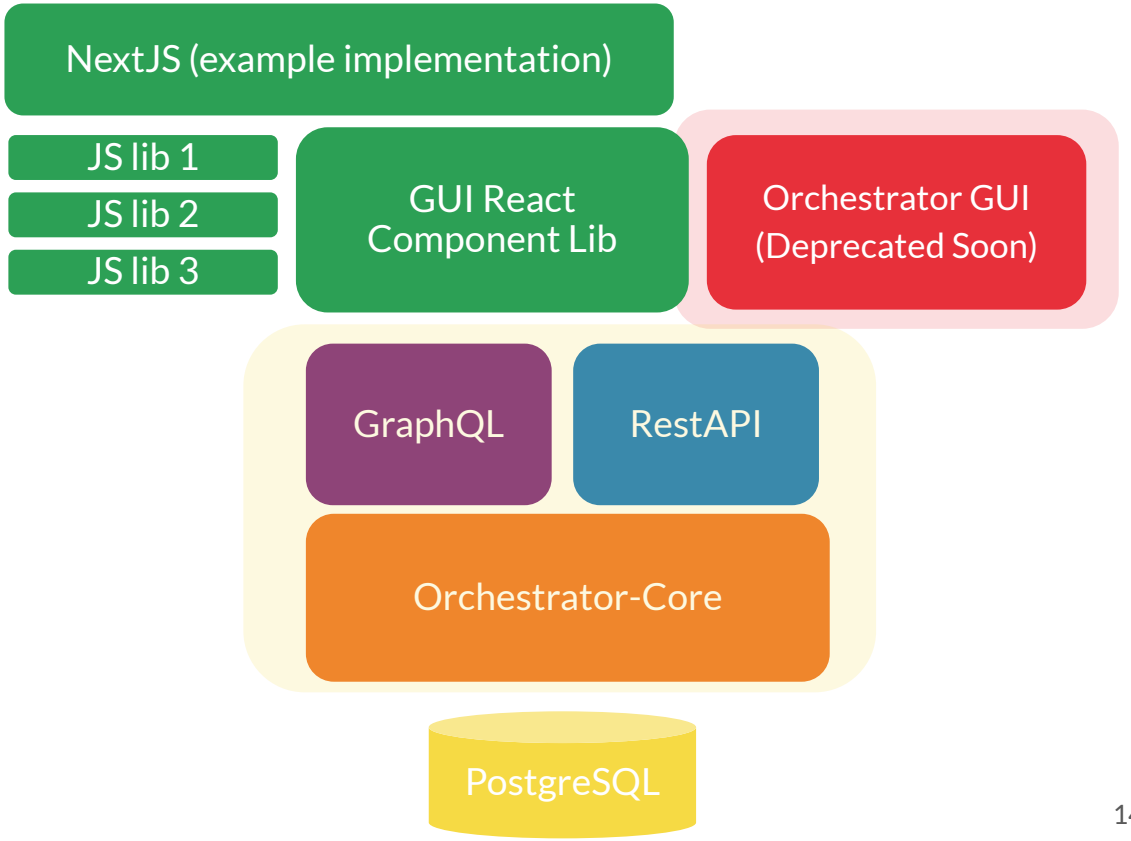
Lifecycle of a Service



"a service is an instance of a product, and called subscription"

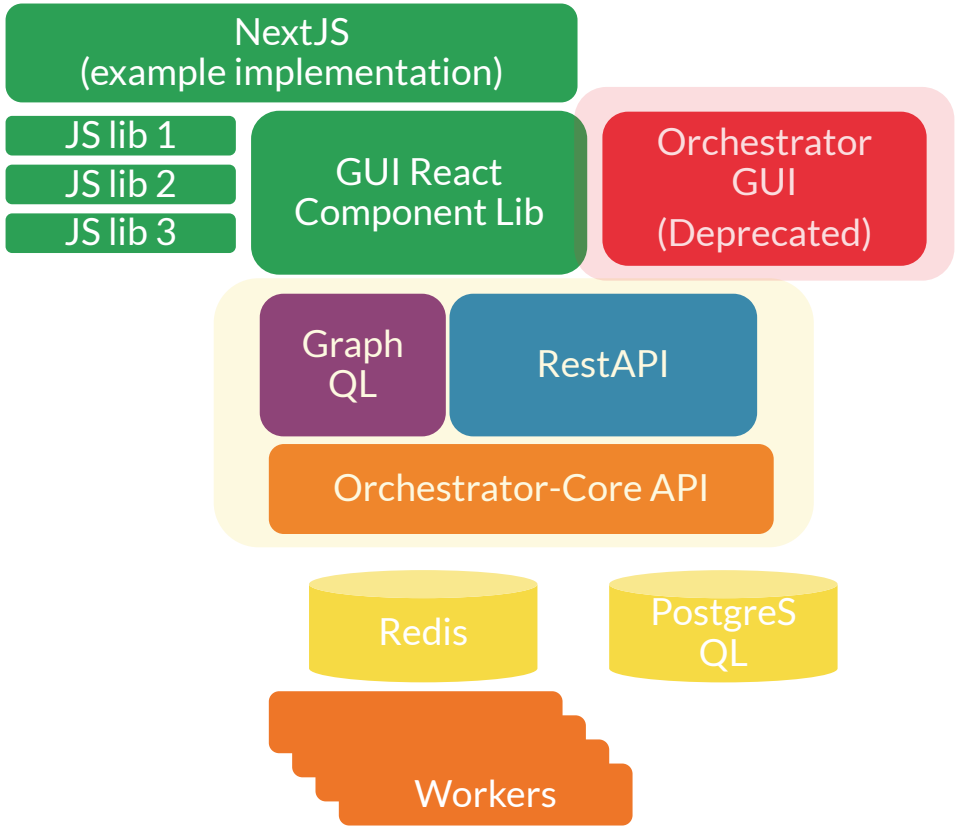
The Orchestrator Application Architecture (basic)

- Python API based on FastAPI and Pydantic
- REST API
- GraphQL
- PostgreSQL database
- React application
- EUI components
- NextJS
- Uniforms



The Orchestrator Application Architecture (at scale)

- Python API based on FastAPI and Pydantic
- REST API
- GraphQL
- **Celery**
- PostgreSQL database
- **Redis**
- React application
- EUI components
- NextJS
- Uniforms



The Orchestrator Application Architecture (Workshop)

- Python API based on FastAPI and Pydantic
- REST API
- PostgreSQL database
- React application
- EUI components
- Uniforms