Brane

A Framework for Programmable Orchestration of Multi-Site Applications

Onno Valkering, Reginald Cushing, Adam Belloum
o.a.b.valkering@uva.nl

Multiscale Networked Systems (MNS)
University of Amsterdam

https://mns-research.nl https://epi-project.nl https://github.com/onnovalkering/brane

Table of Contents

- **Multi-site applications**
    - *Context*
    - *Challenges*
- **The Brane framework**
    - *Seperation of concerns*
    - *Runtime system*
- **Programming Model**
    - *Packages*
    - *DSLs*
- **Future work**

Multi-site applications

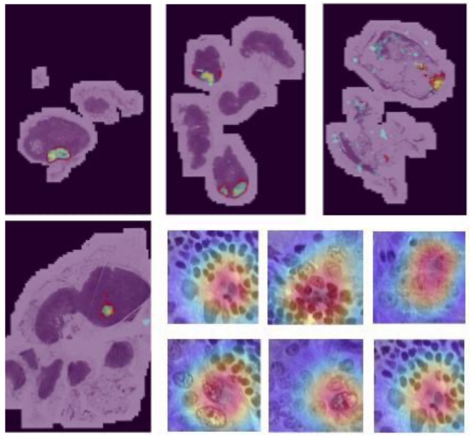
Context

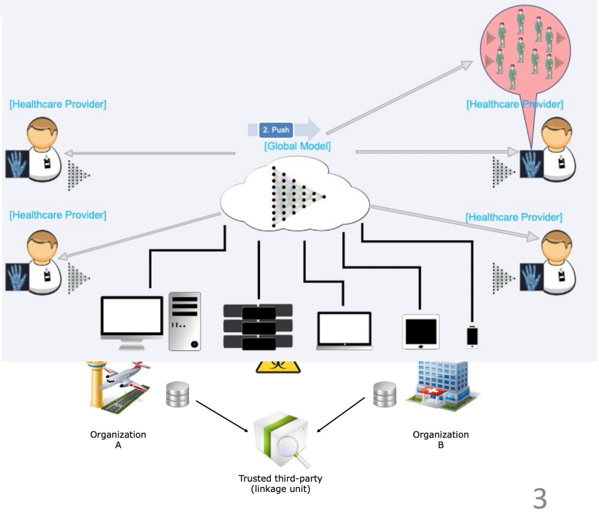
https://process-project.eu

A single infrastructure provider might not be able to fulfill all project's **compute and storage** requirements.

https://epi-project.nl

Privacy or **security** constraints might restrict centralized aggregation and demand federated data processing.





Multi-site applications

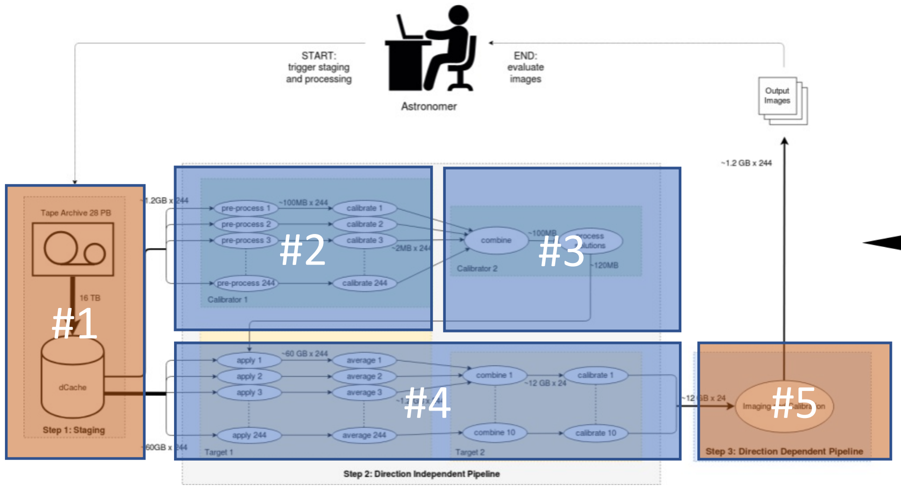
Challenges

- Brane aims to **streamline** and **simplify** development and deployment of complex multi-site scientific applications.
  - How to develop locally, but run it distributed on **heterogenous infrastructures**?
  - How can globally dispersed research teams and organizations **collaborate**?
  - How to **monitor and control** applications that run on a dispersed resource pool?
  - How to appropriately and securely handle **sensitive data** in a multi-domain setting?
- If not addressed, this may lead to **delayed** or **missed** innovation!

The Brane framework

Seperation of concerns

- Focus on both **technical** and **organizational** challenges.
- Accommodate different roles with different tools.



1) Engineers: prepare building blocks

Domain-specific languages
(DSLs)

jupyter

2) Scientists: compose applications

netherlands eScience center
XENON
slurm
workload manager

HPC

Cloud

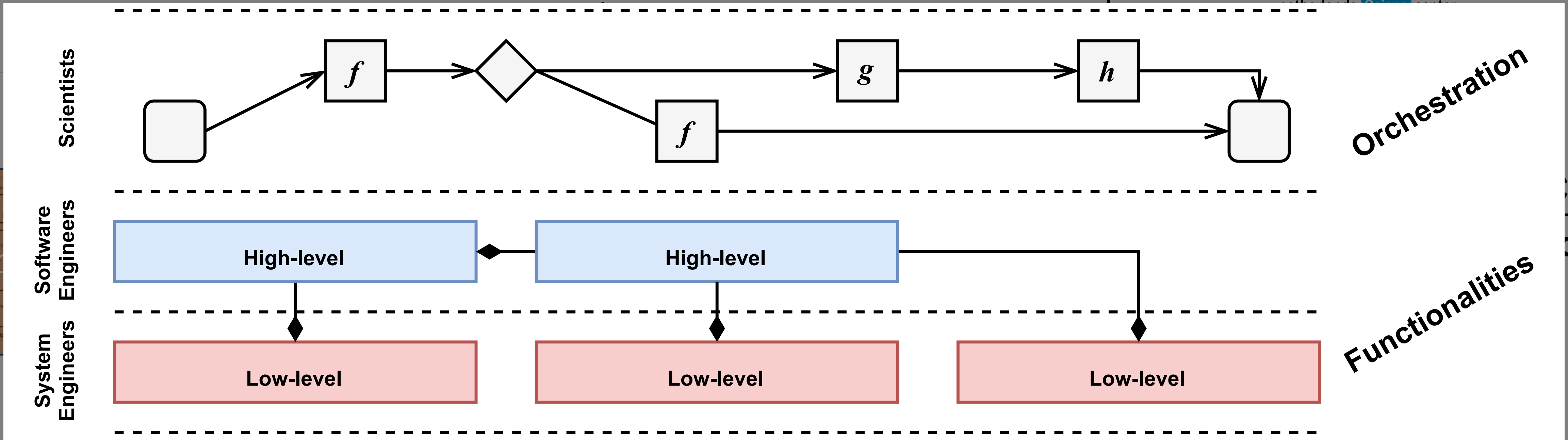
docker
kubernetes

3) Brane: deploy (*almost*) anywhere

The Brane framework

Seperation of concerns

- Focus on both **technical** and **organizational** challenges.
- Accommodate different roles with different tools.

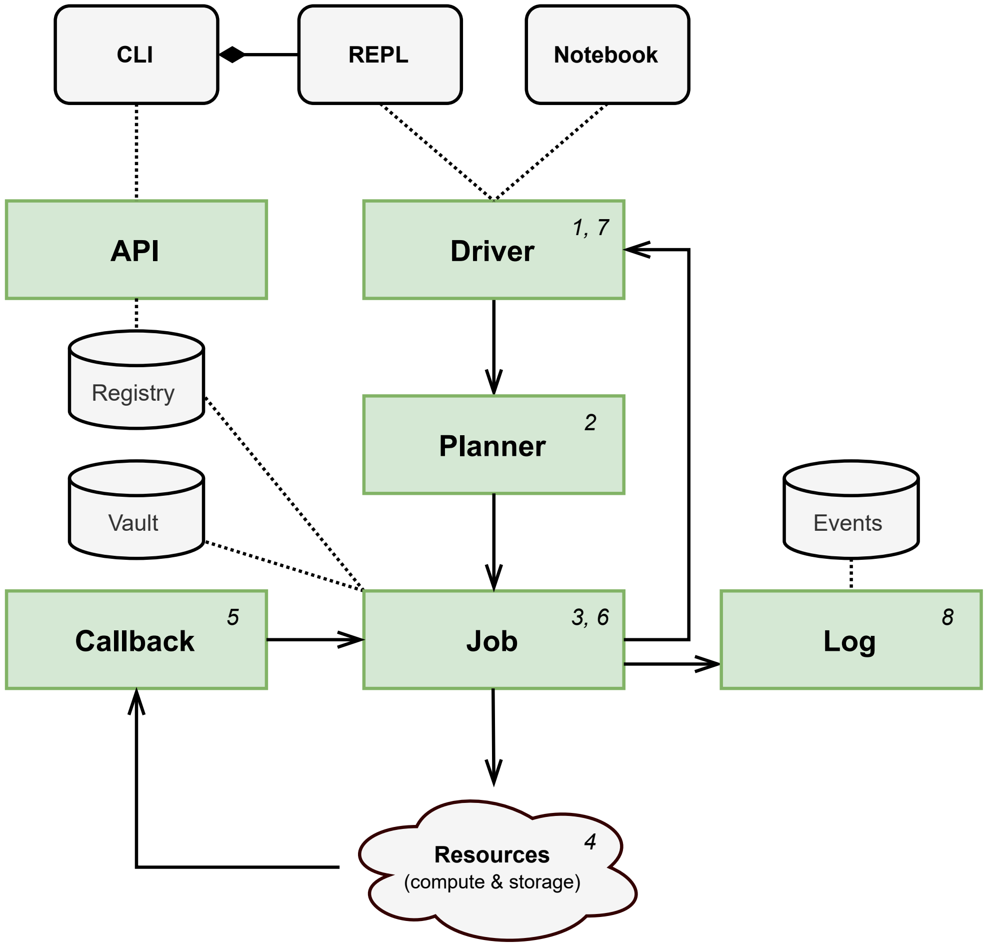


1) **Engineers:** prepare building blocks | 2) **Scientists:** compose applications | 3) **Brane**: deploy (*almost*) anywhere

The Brane framework

Runtime system

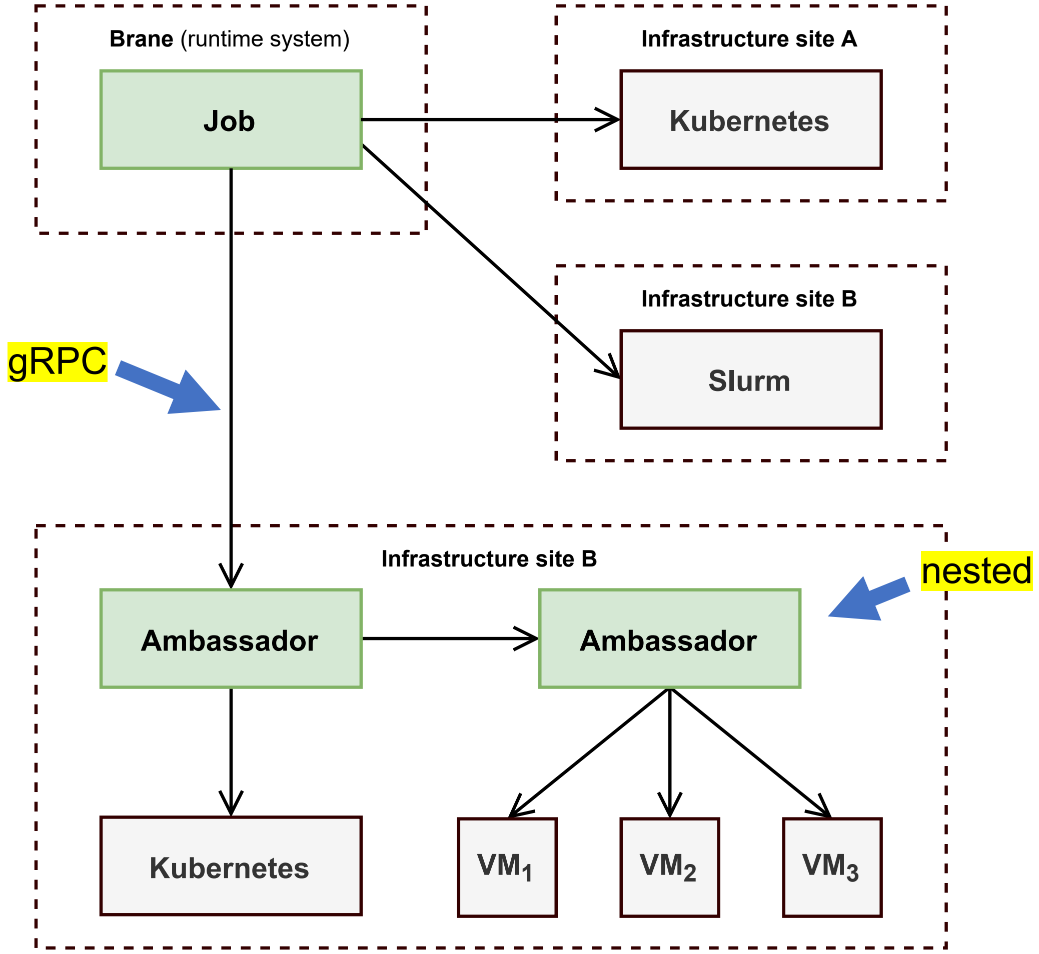
- The runtime system is a collection of loosely coupled **microservices.**
- The **centralized** runtime system relies, by default, on direct control of compute and storage resources.



The Brane framework

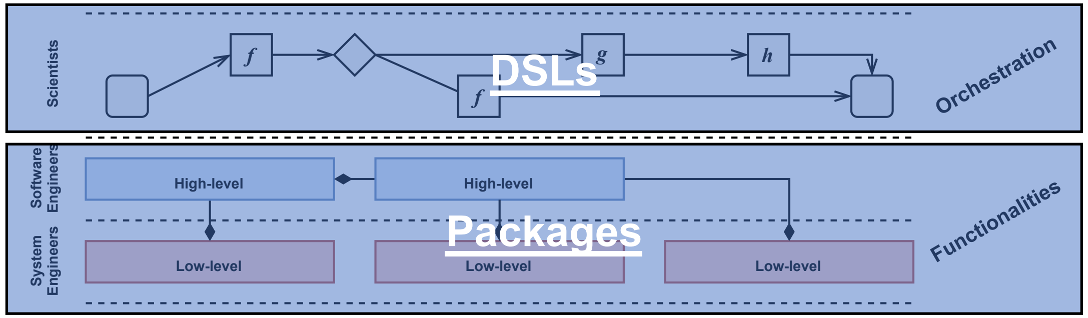
Runtime system

- The runtime system is a collection of loosely coupled **microservices.**
- The **centralized** runtime system relies, by default, on direct control of compute and storage resources.
- If direct control is not desired, an **ambassador** service can be used on-site as a **layer of indirection**.

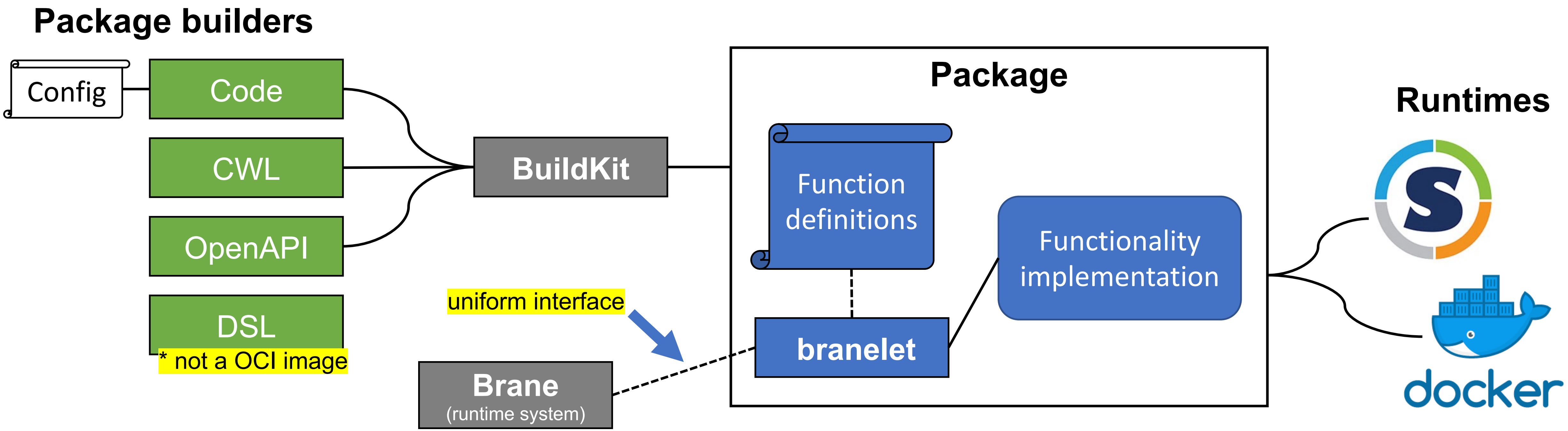


Programming model

Packages

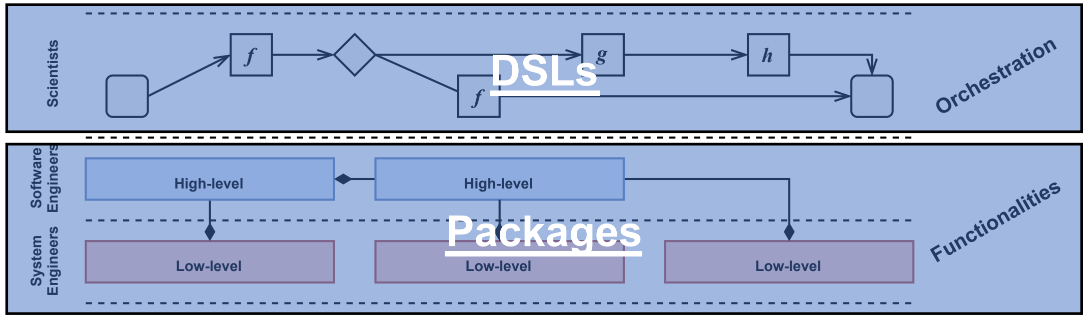


- Brane currently offers four **package builders**.
- Packages are **self-contained** and in the OCI image format.

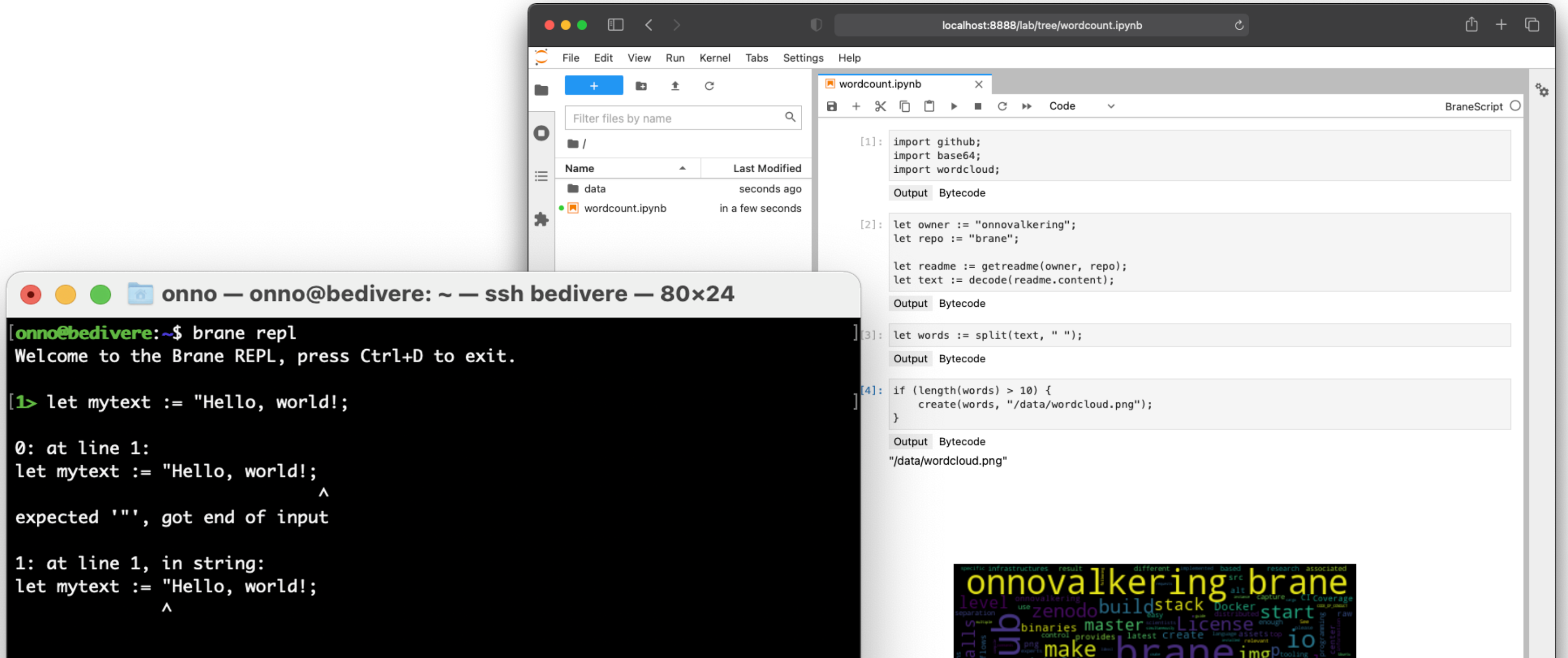


Programming model

DSLs



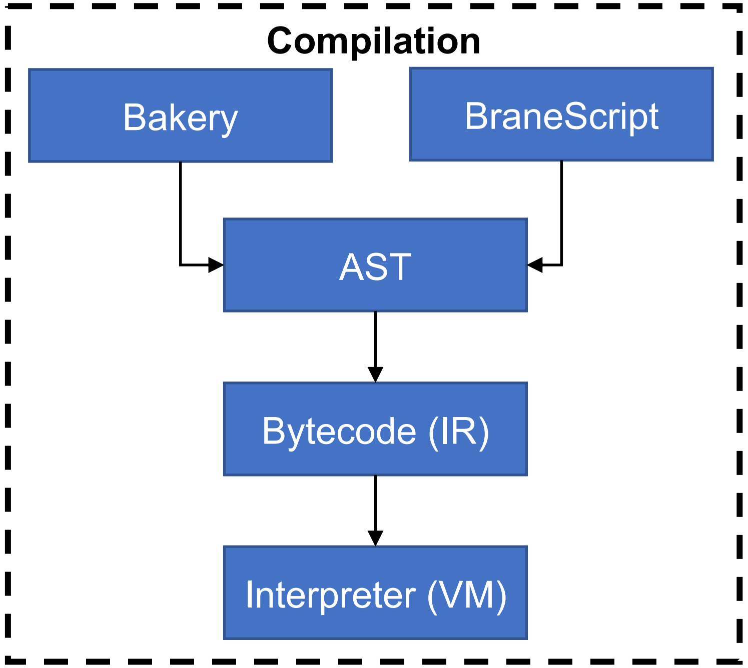
- Two domain-specific languages as **glue** between Brane packages.
  - **Bakery** requires limited to no programming experience.
  - **BraneScript** is a more complete and C-like language.



```
[1]: import github;
     import base64;
     import wordcloud;

[2]: let owner := "onnovalkering";
     let repo := "brane";

     let readme := getreadme(owner, repo);
     let text := decode(readme.content);

[3]: let words := split(text, " ");

[4]: if (length(words) > 10) {
         create(words, "/data/wordcloud.png");
     }
```

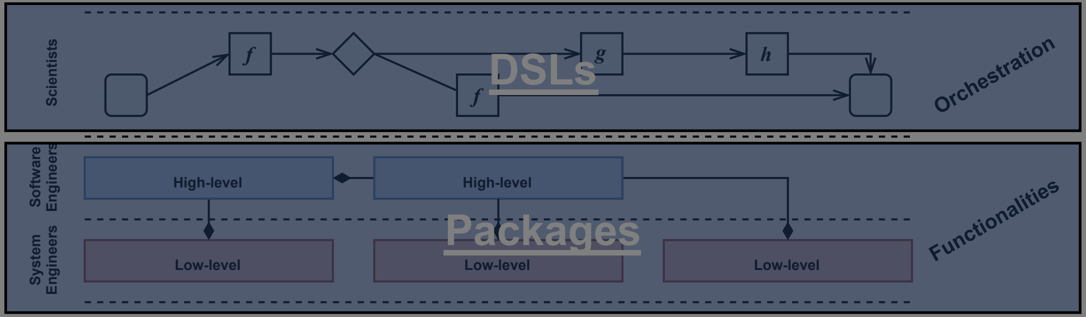
"/data/wordcloud.png"

```
onno@bedivere:~$ brane repl
Welcome to the Brane REPL, press Ctrl+D to exit.

1> let mytext := "Hello, world!;

0: at line 1:
let mytext := "Hello, world!;
                             ^
expected '"', got end of input

1: at line 1, in string:
let mytext := "Hello, world!;
               ^
```

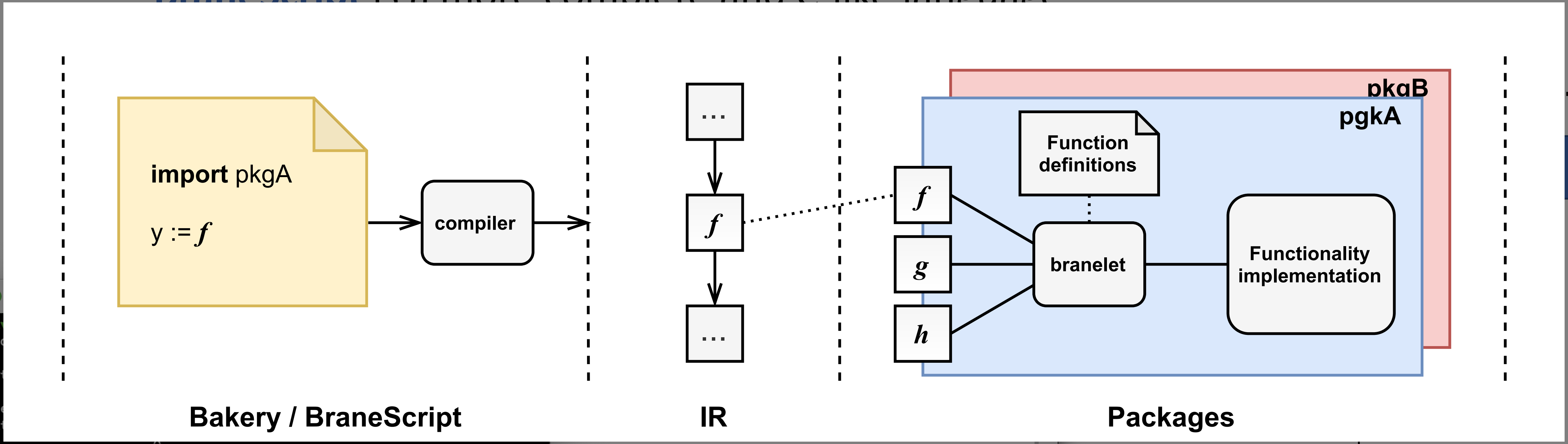


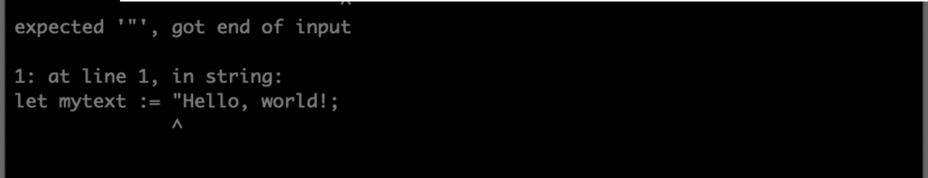
Programming model

DSLs

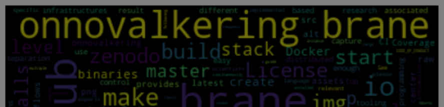


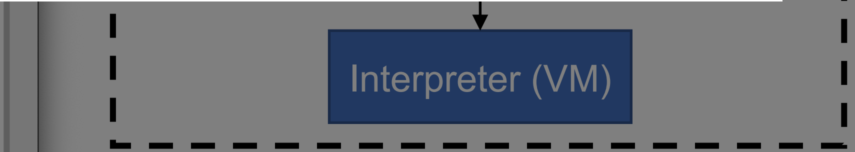
- Two domain-specific languages as **glue** between Brane packages.
  - **Bakery** requires limited to no programming experience.
  - **BraneScript** is a more complete and C-like language





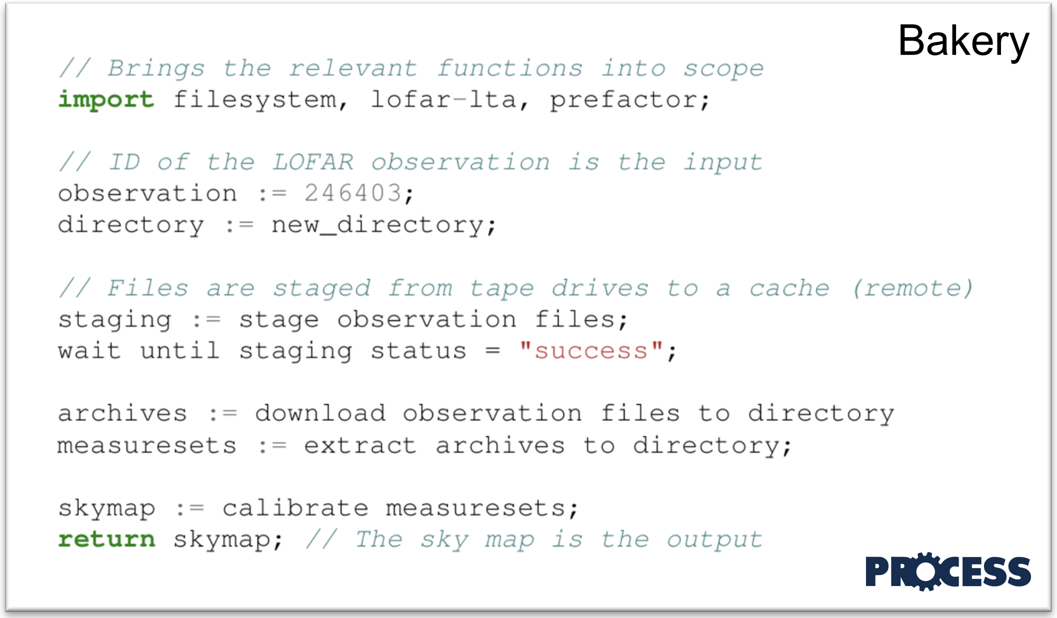
```
[onno@bedi
Welcome t
[1> let my
0: at line
let mytext
expected '"', got end of input

1: at line 1, in string:
let mytext := "Hello, world!;
               ^
```





Programming model

DSLs

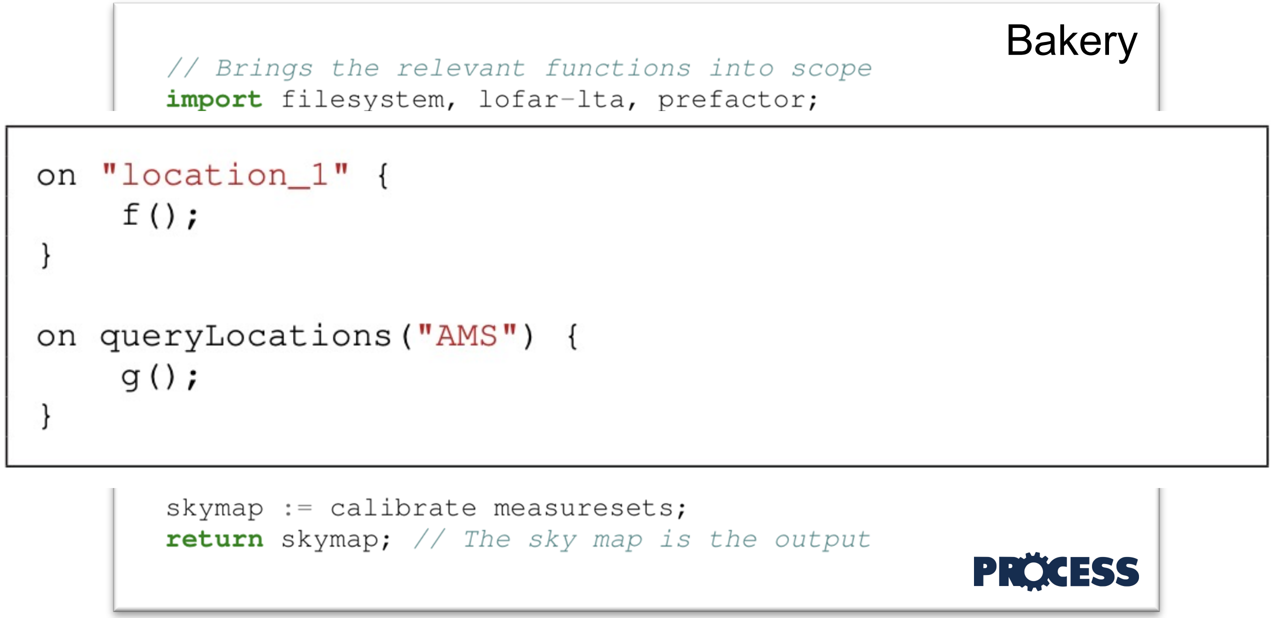


Bakery

```
// Brings the relevant functions into scope
import filesystem, lofar-lta, prefactor;

// ID of the LOFAR observation is the input
observation := 246403;
directory := new_directory;

// Files are staged from tape drives to a cache (remote)
staging := stage observation files;
wait until staging status = "success";

archives := download observation files to directory
measuresets := extract archives to directory;

skymap := calibrate measuresets;
return skymap; // The sky map is the output
```

Programming model

DSLs



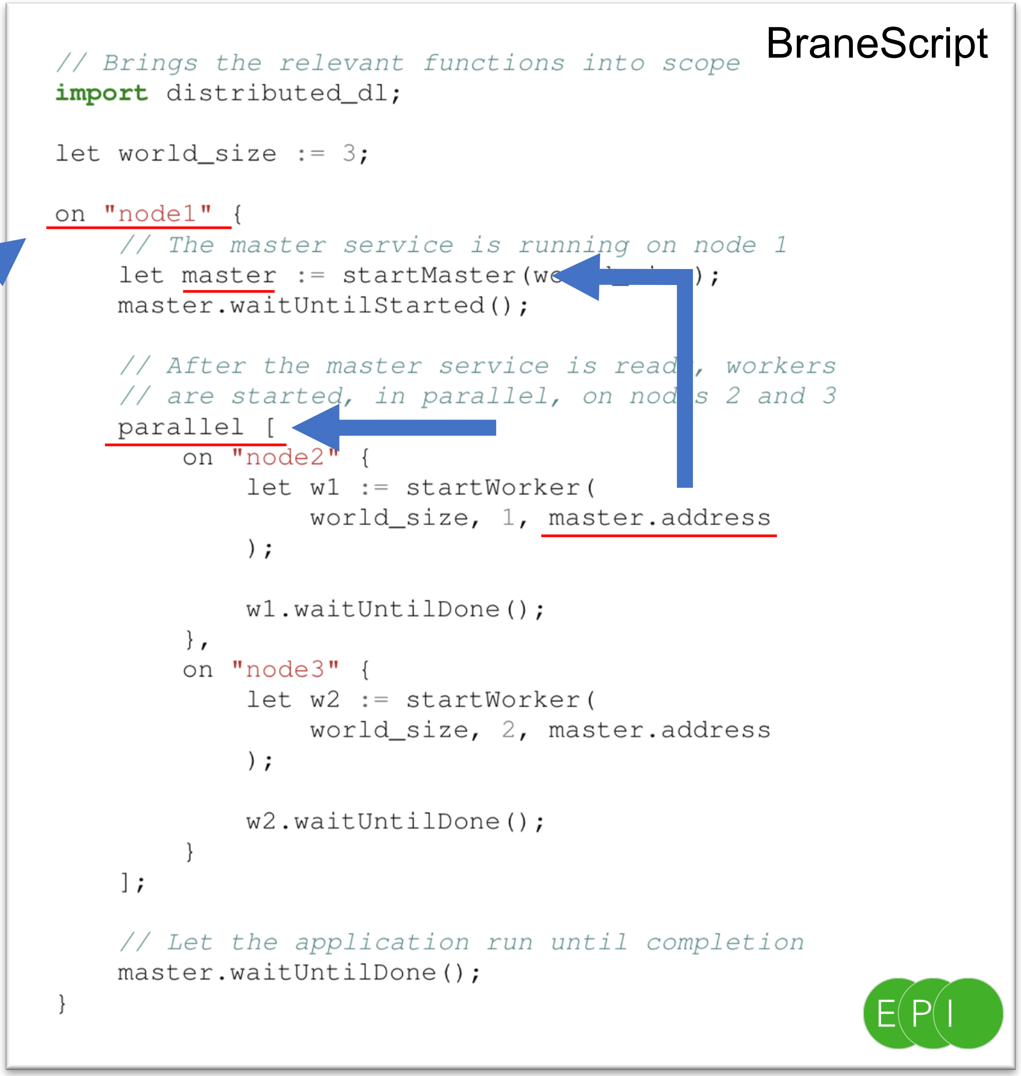
Bakery

```
// Brings the relevant functions into scope
import filesystem, lofar-lta, prefactor;
```

```
on "location_1" {
    f();
}

on queryLocations("AMS") {
    g();
}
```

```
skymap := calibrate measuresets;
return skymap; // The sky map is the output
```

PROCESS



BraneScript

```
// Brings the relevant functions into scope
import distributed_dl;

let world_size := 3;

on "node1" {
    // The master service is running on node 1
    let master := startMaster(world_size);
    master.waitUntilStarted();

    // After the master service is ready, workers
    // are started, in parallel, on nodes 2 and 3
    parallel [
        on "node2" {
            let w1 := startWorker(
                world_size, 1, master.address
            );

            w1.waitUntilDone();
        },
        on "node3" {
            let w2 := startWorker(
                world_size, 2, master.address
            );

            w2.waitUntilDone();
        }
    ];

    // Let the application run until completion
    master.waitUntilDone();
}
```

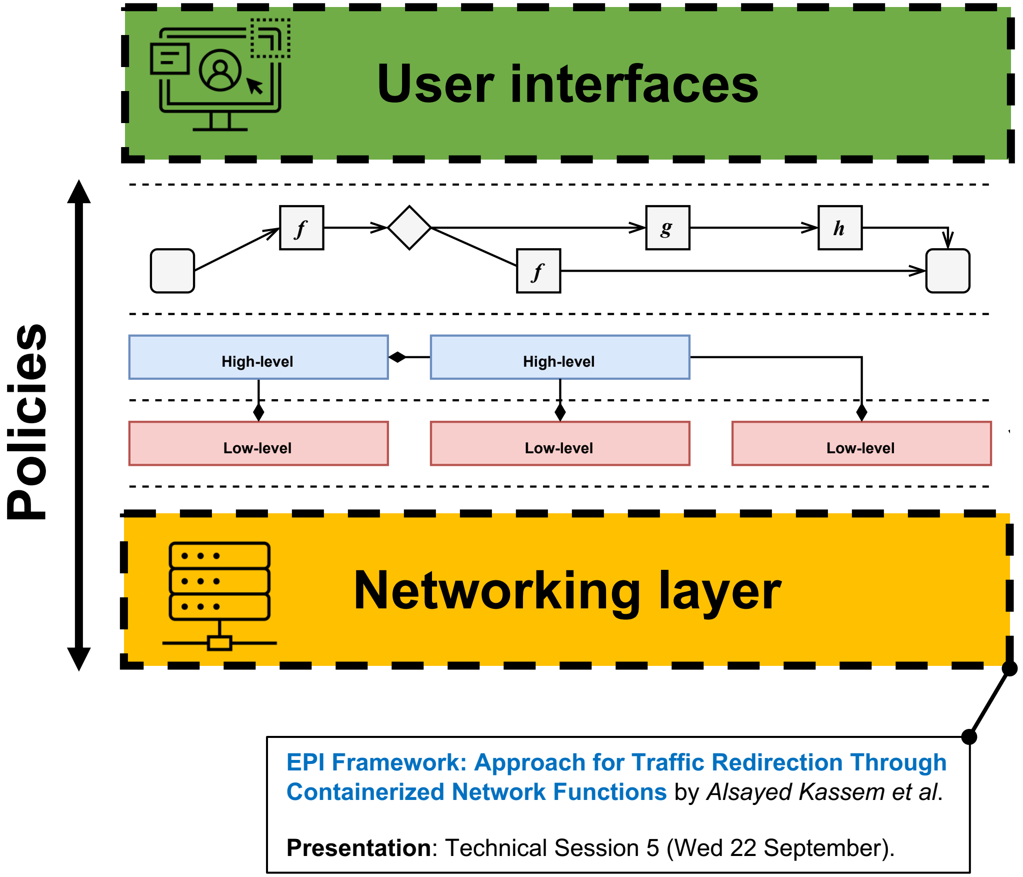
EPI

Future work

- The goal is **complete full-stack** programmable orchestration.
  - Attaching UIs to the runtime system.
  - Inter-domain networking with NFs.
  - Checking applications against policies.

- A **hub** for sharing Brane packages.
- Investigating a **decentralized** setup.

- Additional and deeper **integrations**.
  - Interactive (rich) JupyterLab notebook widgets.
  - Integration with WfMS, Spark, AsyncAPI, …



EPI Framework: Approach for Traffic Redirection Through Containerized Network Functions by *Alsayed Kassem et al.*

Presentation: Technical Session 5 (Wed 22 September).

Currently hiring a scientific programmer to work on Brane. Contact: **Adam Belloum**. a.s.z.belloum@uva.nl

Thank you!

Onno Valkering, Reginald Cushing, Adam Belloum
o.a.b.valkering@uva.nl

Multiscale Networked Systems (MNS)
University of Amsterdam

Multiscale Networked Systems
https://mns-research.nl

https://epi-project.nl

https://github.com/onnovalkering/brane

Currently hiring a scientific programmer to work on Brane. Contact: **Adam Belloum**. a.s.z.belloum@uva.nl