PHYSICS

PHYSICS DESIGN ENVIRONMENT FOR FAAS

PHYSICS Webinar, 3/3/2022

Dr. Georgios Kousiouris
Department of Informatics and Telematics
Harokopio University of Athens

This project has received funding from the European Union's horizon 2020 research and innovation programme under grant agreement no 101017047

The Application Developer Saga…

- The Cloud Era
  - Ephemeral and scalable resources
  - Applications need to get more distributed
- The Microservice Era
  - Applications need to get more modular, scalable and manageable
- The Serverless Era
  - Built-in scalability in the computing model
  - Forces to follow a cloud native design
  - Applications need to break down into functions executed on demand
- Every 4-6 years

Your app needs to adapt!

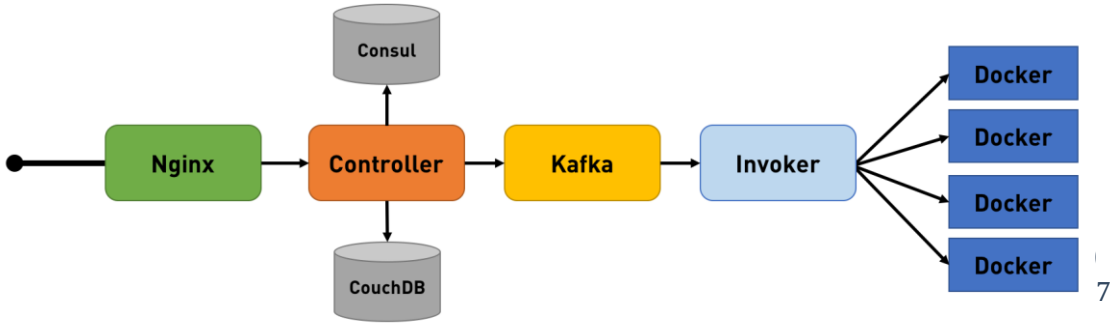
Your app needs to adapt!

Surprise! Your app needs to adapt!

This project has received funding from the European Union's horizon 2020 research and innovation programme under grant agreement no 101017047

Characteristics of Function as a Service

- Application logic is split into smaller chunks
- Function execution should be stateless
- Function execution is performed inside a container, launched on demand upon request
  - External events, cron jobs, calls by other functions etc.
- Incorporation of arbitrary and legacy application components as an arbitrary container
- Costs are based on function invocations, runtime and memory used for the function



7

Challenges of application development in the FaaS model[1,2]

- How can functions become manageable, shared, grouped and reused
  - Express increasing complexity in the functionality
- How to test , debug, deploy and manage serverless systems and applications?
  - What tools and IDE features are needed? DevOps processes etc.
- Abstracted application design and porting to the serverless paradigm
  - How to consider and support the legacy part of serverless applications?
- Link between design and dictation of deployment and runtime needs

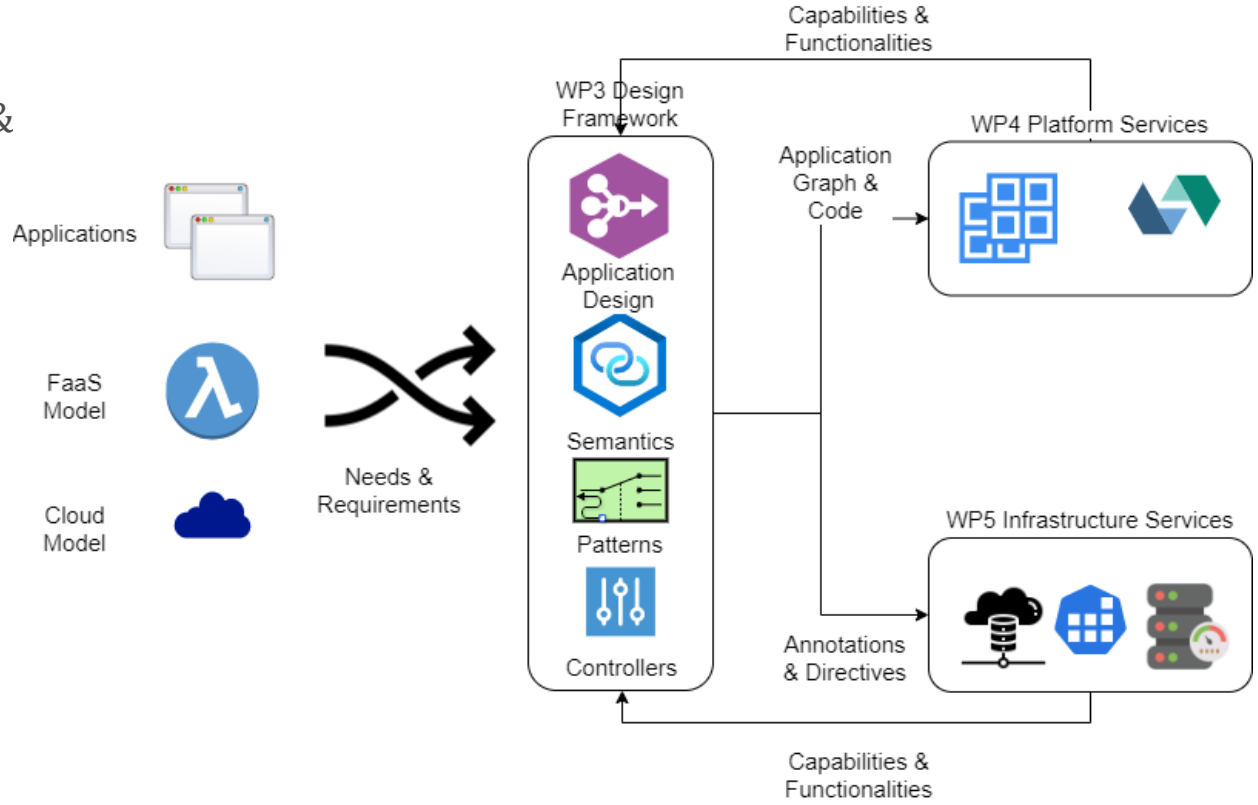
[1]Dagstuhl Seminar 21201 on Serverless Computing
Report available at:
https://drops.dagstuhl.de/opus/volltexte/2021/14798/pdf/dagrep_v011_i004_p034_21201.pdf

[2]Kousiouris, G. and Kyriazis, D., 2021. Functionalities, Challenges and Enablers for a Generalized FaaS based Architecture as the Realizer of Cloud/Edge Continuum Interplay. In *CLOSER* (pp. 199-206).

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement no 101017047

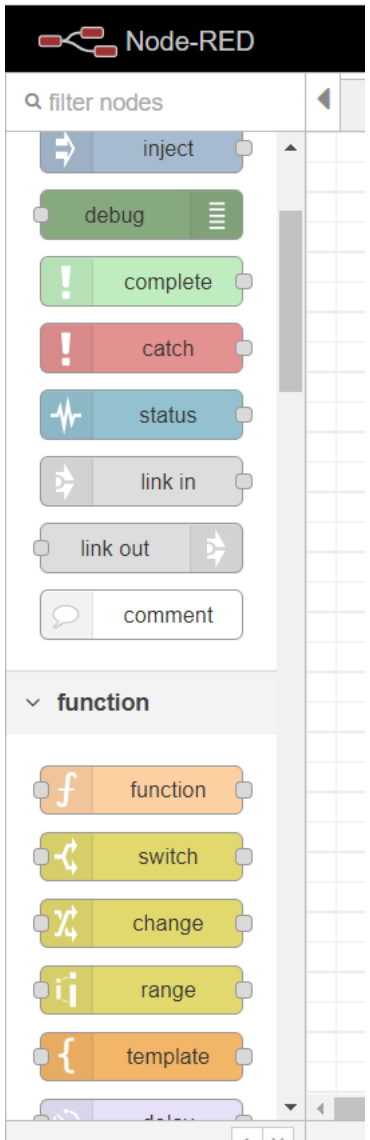
Goals of PHYSICS Design Environment: Versatility and Abstraction

- Visual Environment to easily create function flows
  - Exploiting readymade functionality (patterns) in a drag & drop manner
  - Ability to test locally without having to deploy
  - Translation to the necessary FaaS platform specification
- Multiple execution modes through a modular DevOps process
  - Logic as function, as sequence/workflow, as service
- Ability to include diverse annotations
  - guidelines for management components down the stack

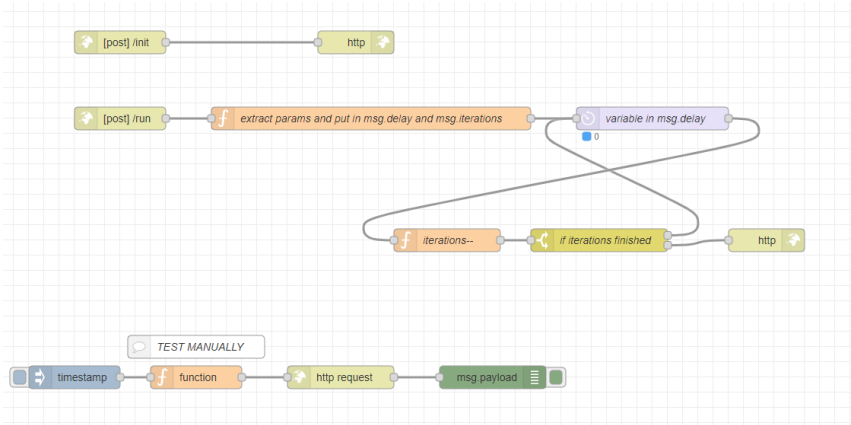


This project has received funding from the European Union's horizon 2020 research and innovation programme under grant agreement no 101017047

Node-RED as the main design element- Why?



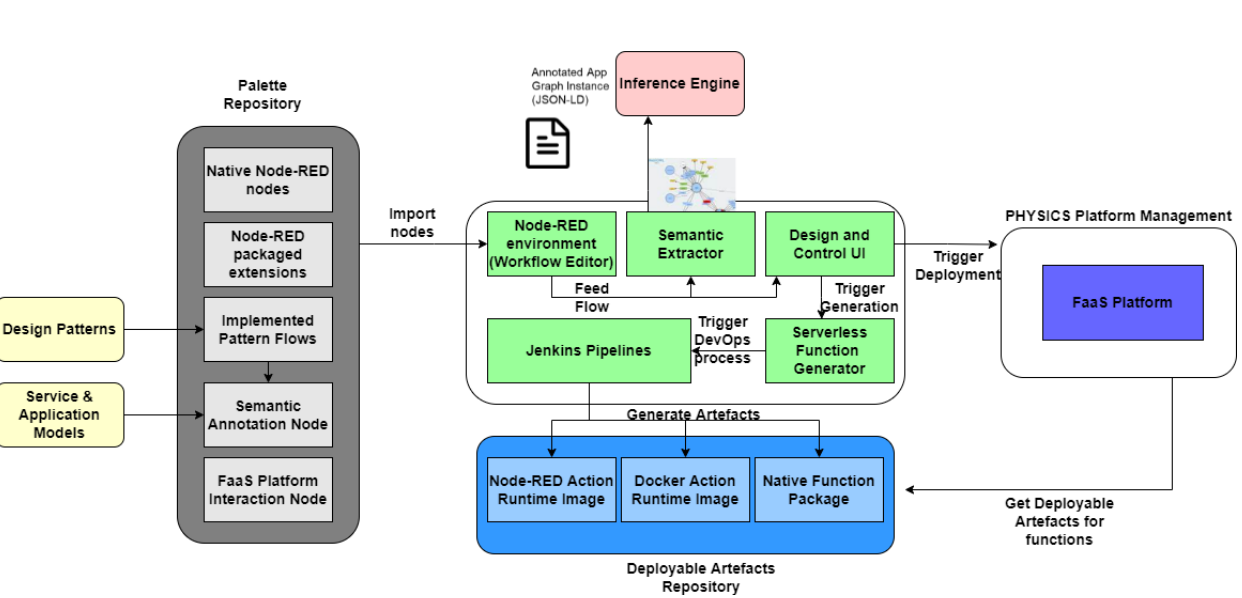
- Functional programming environment for event driven applications
  - Fits to the FaaS function concept
- Pallete of available built-in nodes
  - Variety of operations
    - Protocol inputs
    - Format transformation
    - Clients for external systems
      - DBs, storage etc.
    - Etc.
  - Extended by
    - Personally created nodes and subflows
    - Available flows and nodes in external repositories
      - https://flows.nodered.org/
    - Functions can be grouped arbitrarily in subflows and reused
  - Dragged and dropped in the workflow
- Deployed operationally on a Node-RED server
- Workflow orchestration and function execution abilities
- So why not use that as the baseline design environment for FaaS?

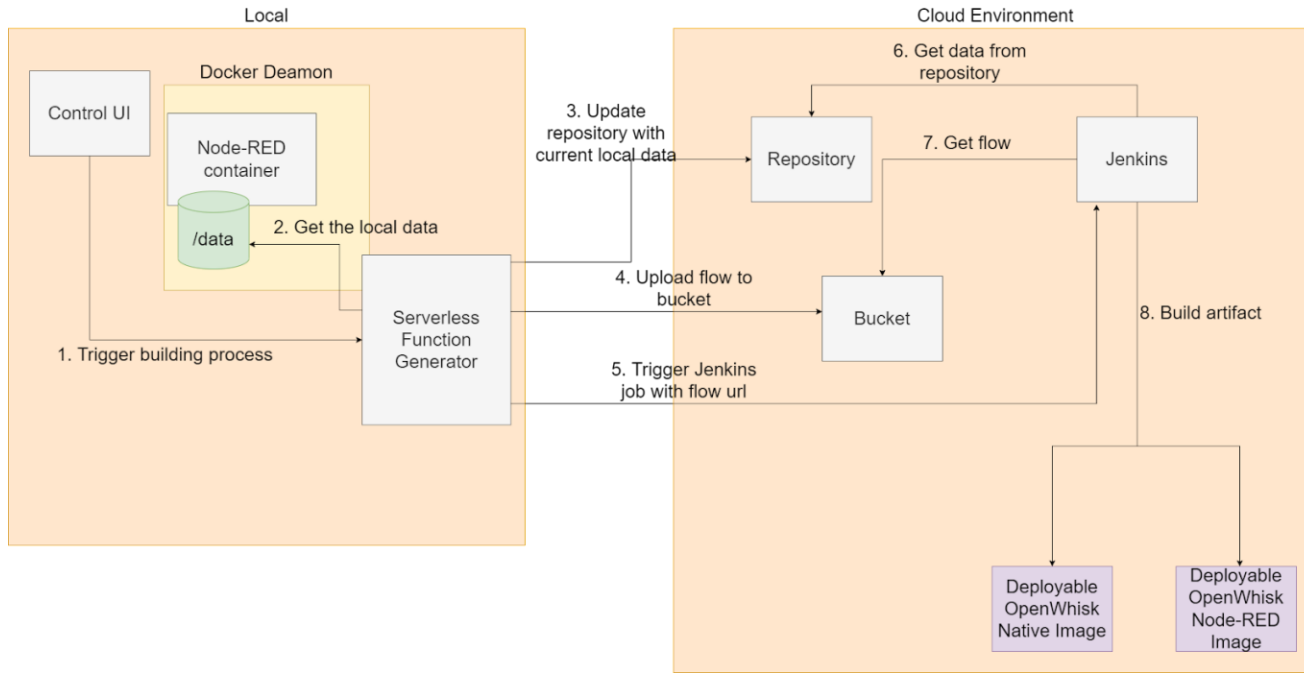


This project has received funding from the European Union's horizon 2020 research and innovation programme under grant agreement no 101017047

Entry Point: PHYSICS Design Environment

- Locally deployed Control UI and PHYSICS provided Node-RED container
- Platform based elements for flow processing and repository management

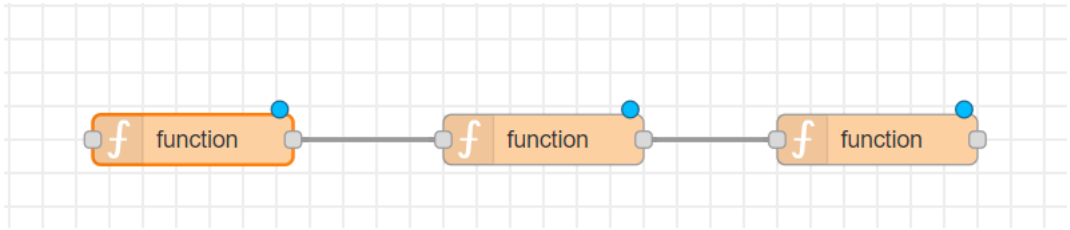




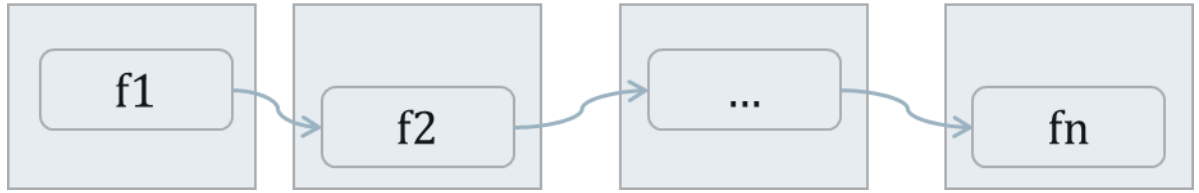
This project has received funding from the European Union's horizon 2020 research and innovation programme under grant agreement no 101017047

Three ways of using Node-RED (1/3)

- One as a function editor and deployer to the OW case
  - Faster
  - Limited at the moment to native Javascript functions
  - Limited in terms of embedded node-red nodes



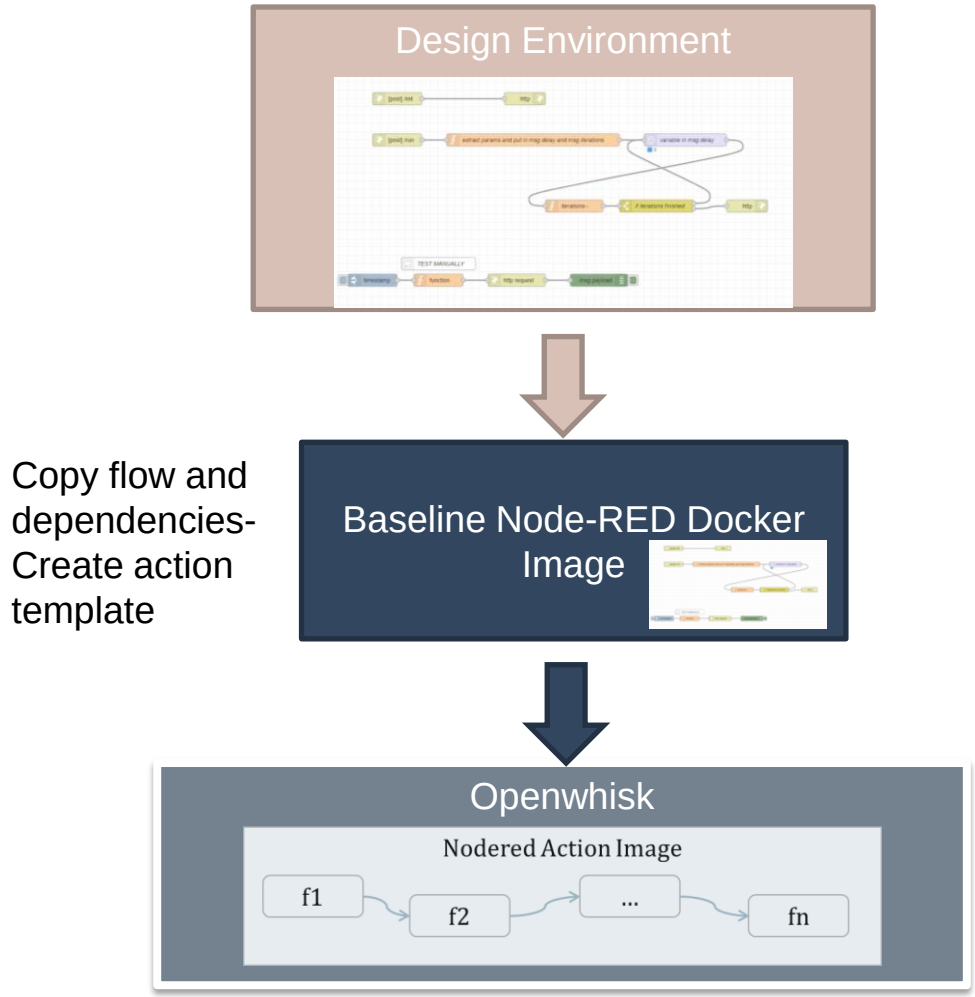
Typical Node.js Openwhisk Runtime Image



This project has received funding from the European Union's horizon 2020 research and innovation programme under grant agreement no 101017047

Three ways of using Node-RED (2a/3): Node-RED as a function execution runtime

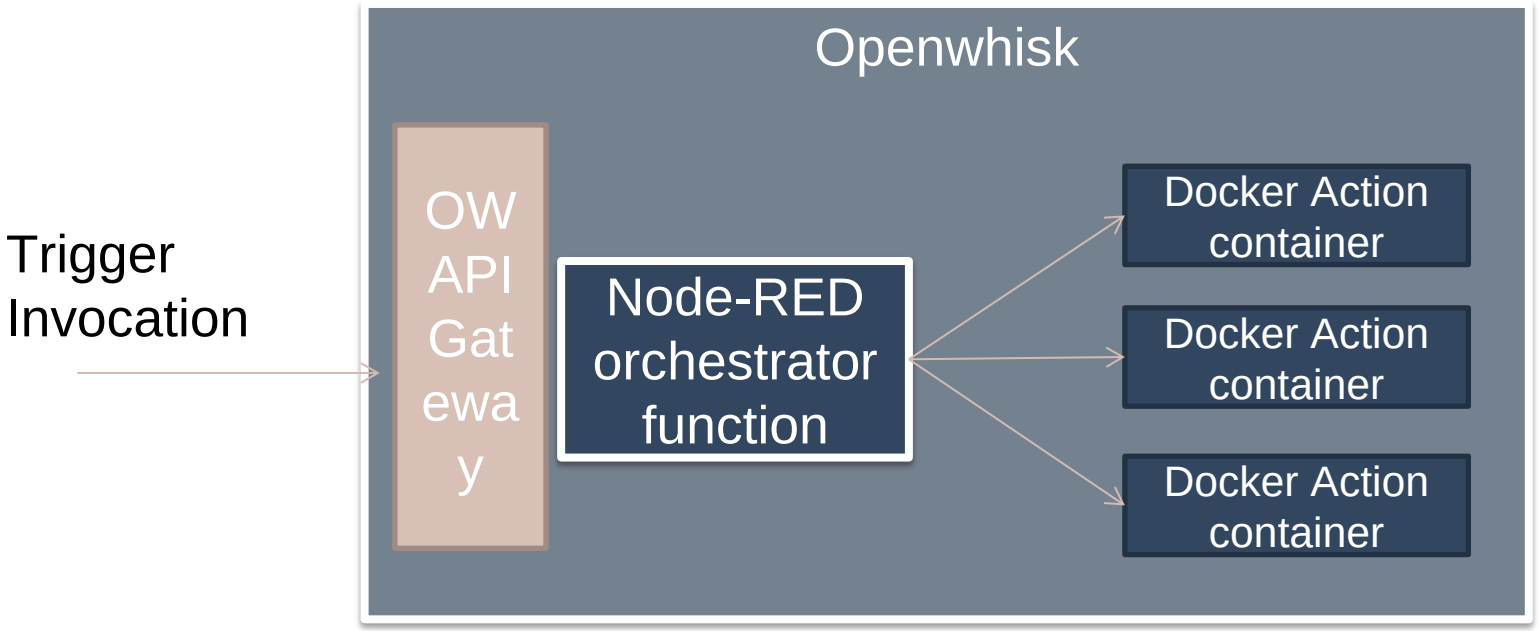
- Create an overall flow in Node-RED that can be directly executed as a function in Openwhisk
  - Exploiting the abundance of nodes and logic
  - Local Testing
- Create a customized Docker image with the specific flow
  - Addition of NR Runtime in the supported OW runtimes
  - Registered to OW through the any-image-as-a-function feature
- The flow can now be directly executed as a function in OW



This project has received funding from the European Union's horizon 2020 research and innovation programme under grant agreement no 101017047

Three ways of using Node-RED (2b/3): Node-RED as serverless orchestrator

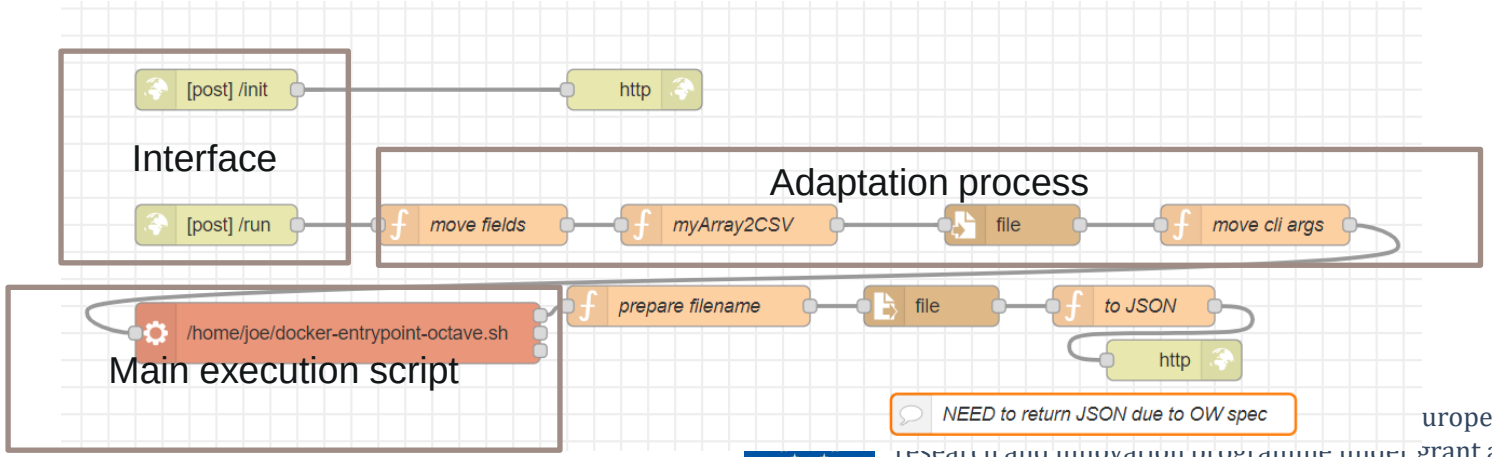
- Enabling the use of Node-RED as an application level orchestrator, running as a function

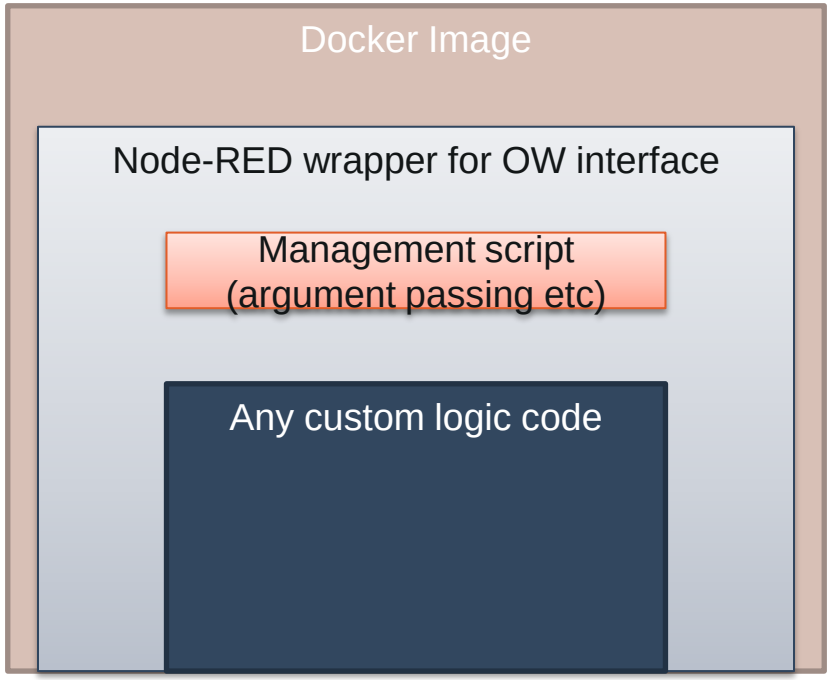


This project has received funding from the European Union's horizon 2020 research and innovation programme under grant agreement no 101017047

Three ways of using Node-RED (3/3): Node-RED wrapper around legacy logic

- 3 layers
  - Wrapper service in Node-RED
    - Needed by the OW specification for the any-image-as-function feature
  - Adaptation layer for management purposes (argument passing etc)
  - Custom Logic related scripts (migrate existing code in a more or less coarse grained manner)

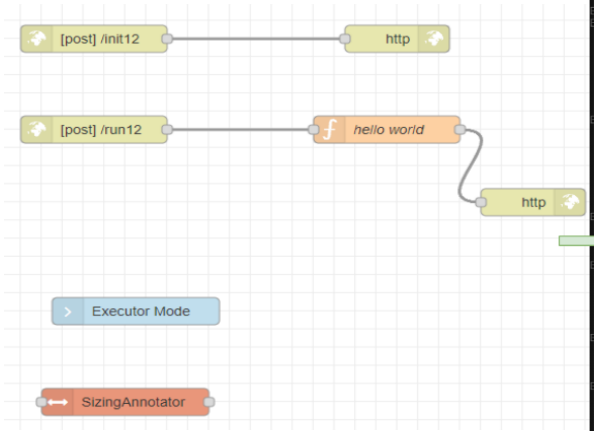


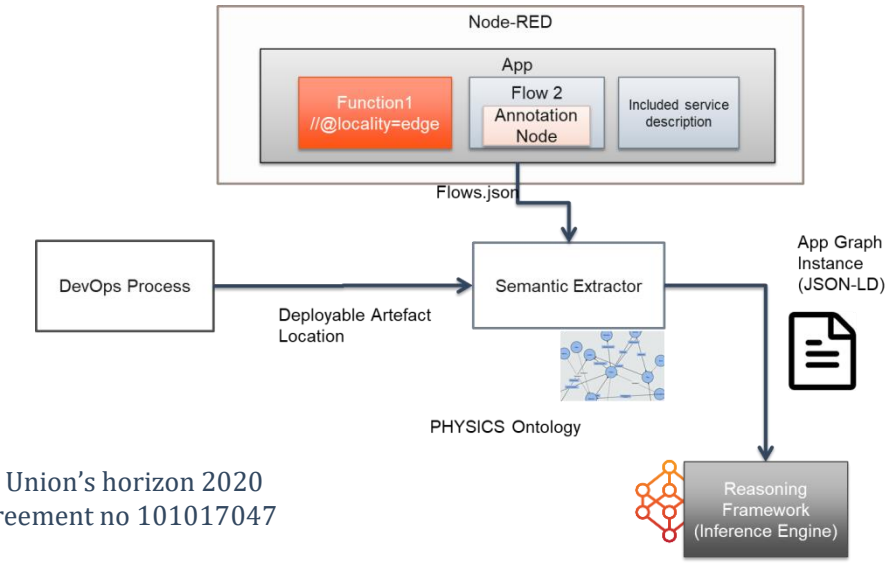


uropean Union's horizon 2020
research and innovation programme under grant agreement no 101017047

Annotations and semantics

- Annotation mechanisms through the Node-RED logic
  - Custom nodes
  - Code level specification for annotations at the function level
- Post-processing of the annotated flow to extract semantics
  - Mapping to a relevant Ontology
  - https://service.tib.eu/webvowl/#iri=https://drive.google.com/u/0/uc?id=1-9dnKP3Qr0oa9dEarp-hH2QfYvbXuDN4&export=download
- Semantics propagated to the underlying management layers
  - Schedulers, K8S placement etc.



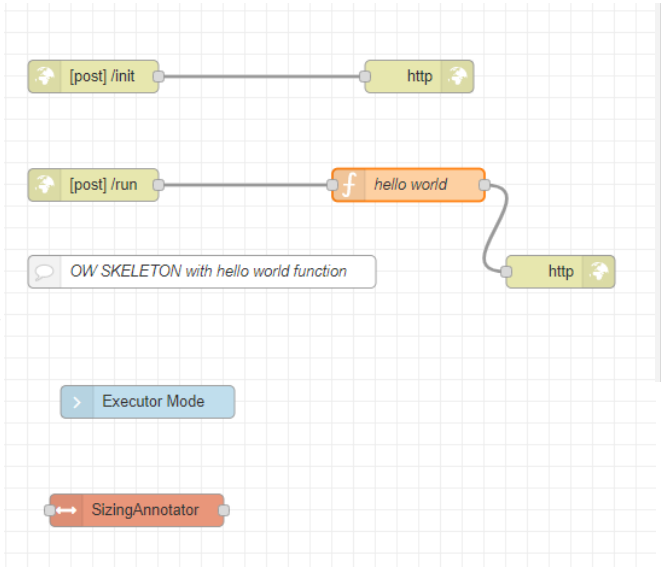


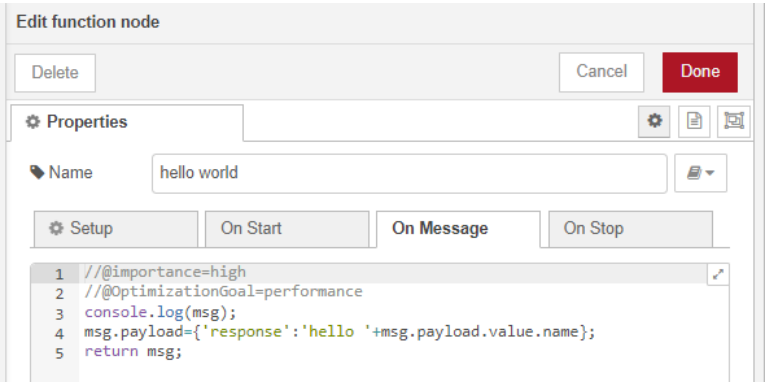
This project has received funding from the European Union's horizon 2020 research and innovation programme under grant agreement no 101017047

Ways of annotating

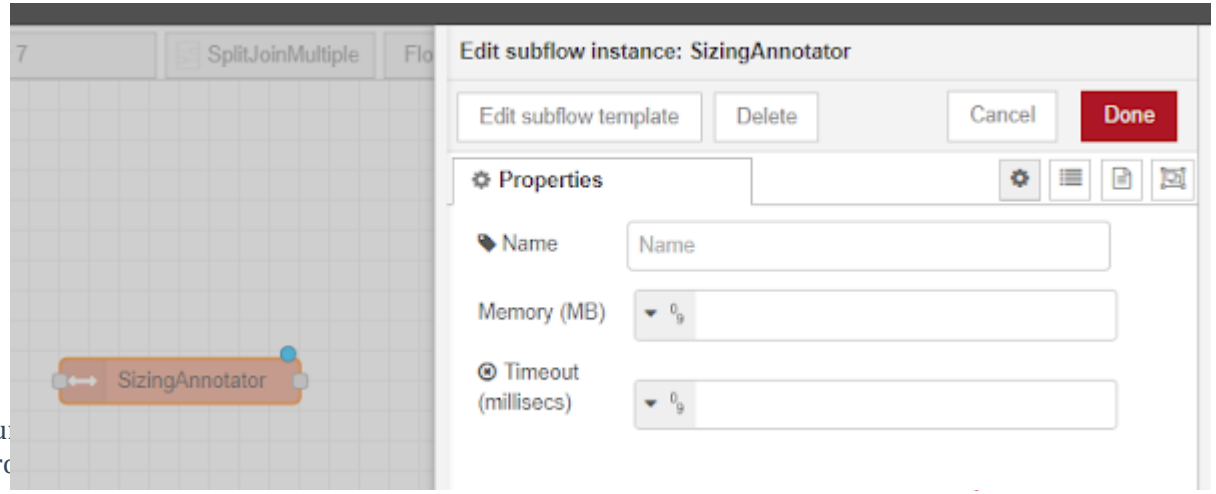
- Function level annotation
  - //@*<key or property>*=*<value>* for when the function is the subject
  - //$<local-definition>@*<property>*=*<value>* for local triple
    - If the property is not enough to infer the type/class of the definition, add:

      $<local-definition>@type=*<class>*





- Flow level annotation
  - Semantic nodes



This project has received fu
research and innovation pro

Examples of Annotations

- Function relative placement
  - Affinity, Antiaffinity
- Function absolute placement
  - Locality (e.g. Cloud/Edge), Replication (in multiple Cloud/Edge locations)
- Function scheduling considerations
  - Importance (High/Medium/Low), maximizeReusableState, OptimizationGoal (Performance, Energy etc)
- Function Sizing and QoS
  - QoSConstraint, sizingGB, sizingCores

This project has received funding from the European Union's horizon 2020 research and innovation programme under grant agreement no 101017047

Patterns to support reusability, manageability and abstracted functionality

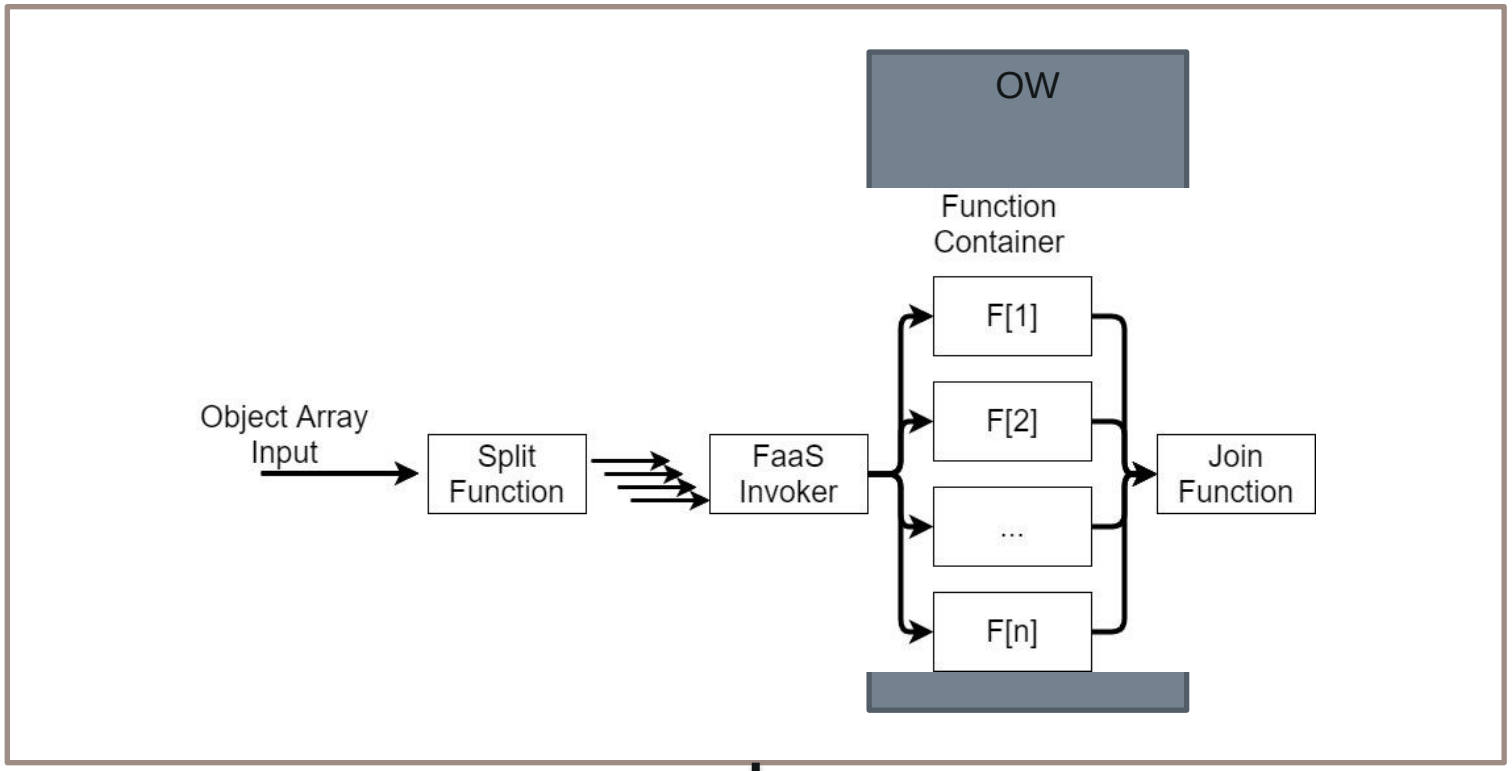
- What is a Pattern
  - *a proven **series of activities** which are supposed to overcome a **recurring problem** in a certain **context**, particular **objective**, and specific **initial condition**[1]*

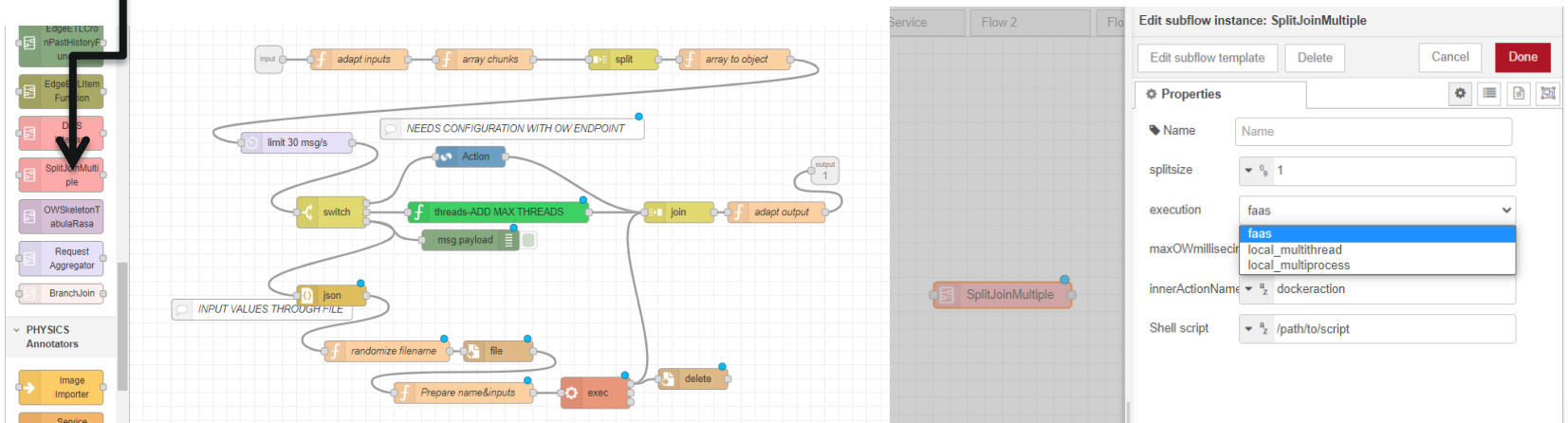
[1]*Agung Wahyudi, George Kuk and Marijn Janssen. A Process Pattern Model for Tackling and Improving Big Data Quality. Information Systems Frontiers (2018) 20:457–469. Springer*

- Exploit the grouping of functions to (sub)flows and their offering through the Node-RED pallete
  - Create reusable function collections, grouped into subflows, to offer generic functionality addressing a specific issue

This project has received funding from the European Union's horizon 2020 research and innovation programme under grant agreement no 101017047

Example of Fork-Join parallelization pattern





Example of Fork Join

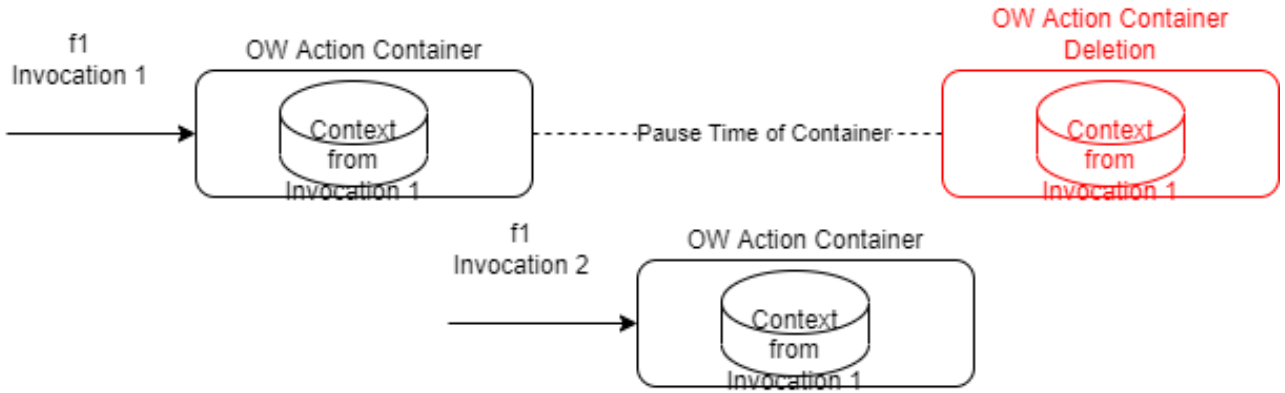
- Can implement parallelism based on different processing elements
  - Multiple spawned OW actions on a cluster
  - Local multiple threads inside the NR environment of the runtime
  - Local multiple processes spawned inside the container of the NR runtime

- Split size of array input dynamically set
  - Needed to reduce interference overheads from too many spawned processing elements

This project has received funding from the European Union's horizon 2020 research and innovation programme under grant agreement no 101017047

Context Management Patterns

- The issue of context reuse due to hot containers creates the opportunity for having multiple ways of handling context

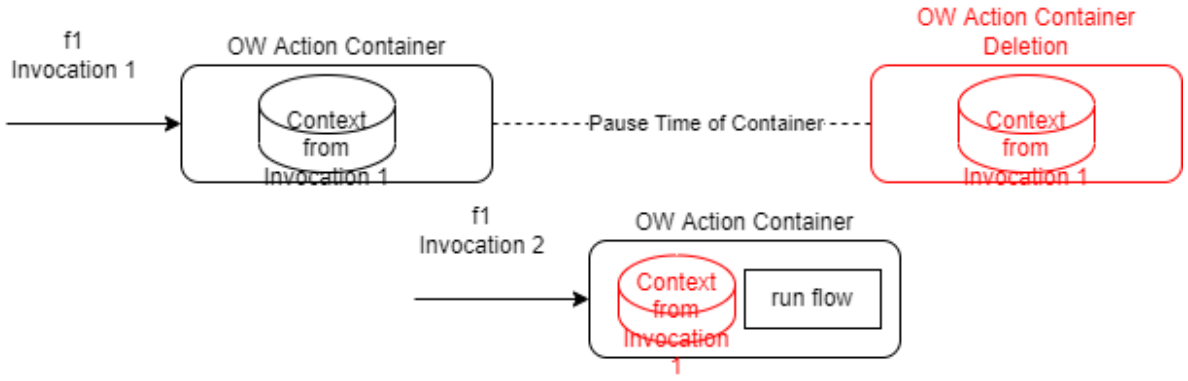


- ***Opportunistic Context Reuse*** (OW Skeleton Runtime)- Default behaviour
  - Context reuse can speed-up function
    - E.g. token retrieval for external access is maintained function invocation if no context is reused
    - Depends on the application needs and scope- Can be dangerous
    - Not guaranteed to be re-used since no hot container may exist-> opportunistic

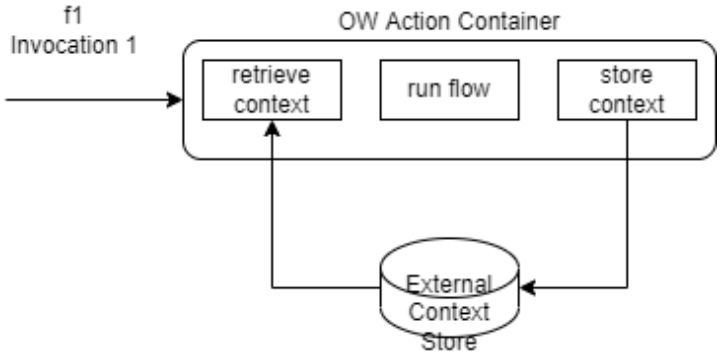
This project has received funding from the European Union's horizon 2020 research and innovation programme under grant agreement no 101017047

Context Management Patterns

- ***Deterministic Context Purge*** (OW Skeleton Tabula Rasa)
  - Context is maintained in the local filesystem (option of Node-RED)
  - Deleted in the beginning of the template flow



- ***Deterministic Context Reuse*** (OW Skeleton Externalized State)- WIP
  - Context is maintained in an external service
  - Brought in during each invocation

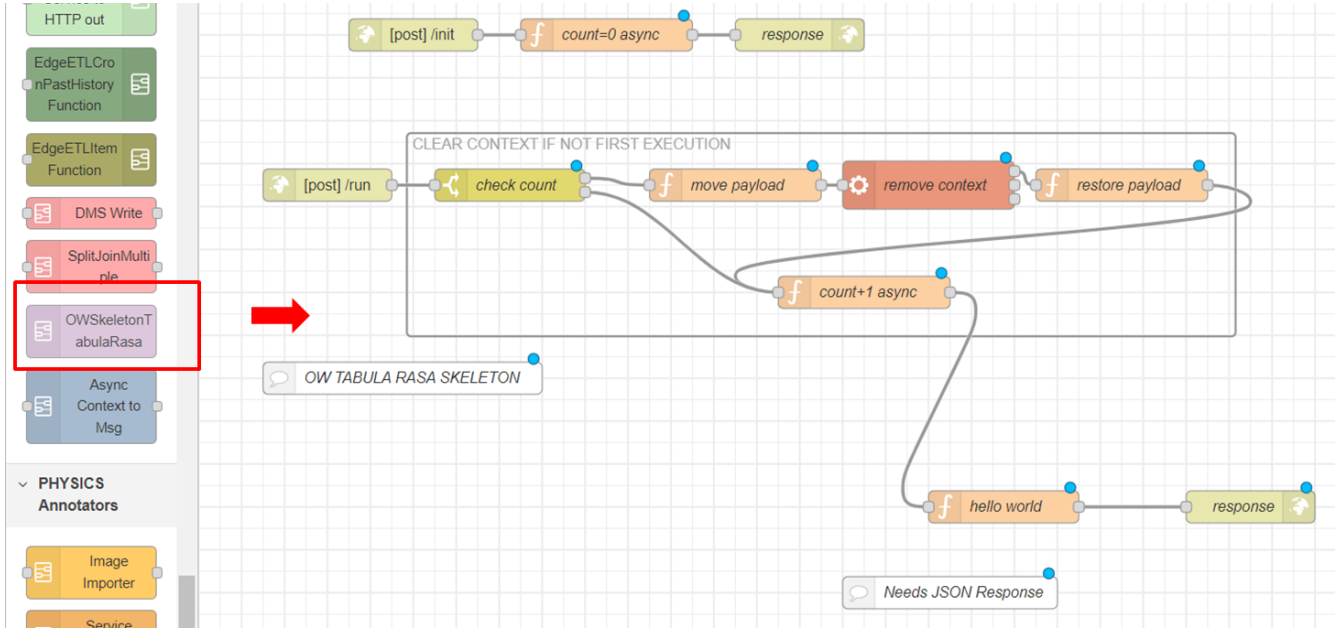


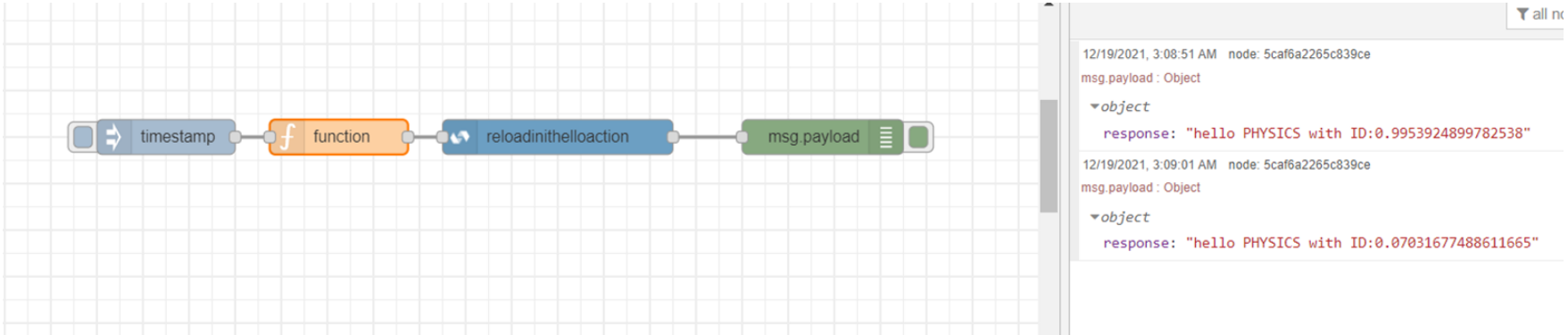
This project has received funding from the European Union's horizon 2020 research and innovation programme under grant agreement no 101017047

OW Skeleton Tabula Rasa

- Node-RED context storage in local filesystem option

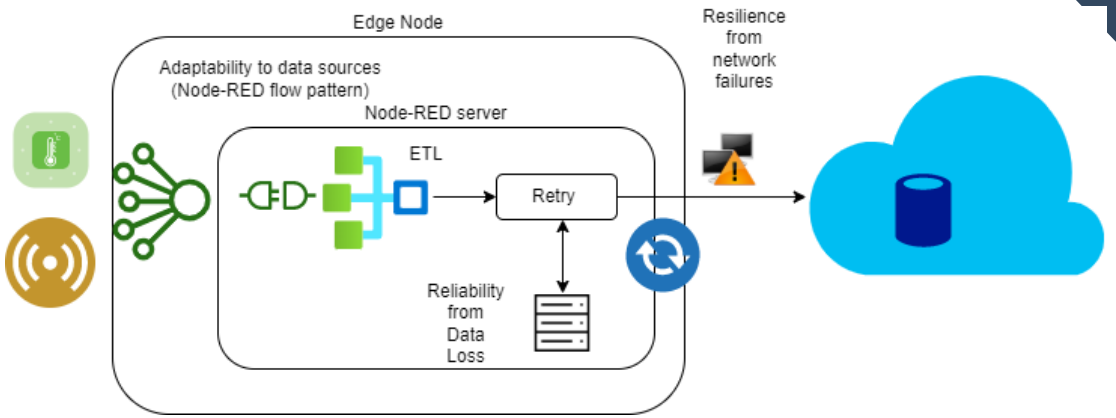
```
// Context Storage
// The following property can be used to
// provided here will enable file-based
// Refer to the documentation for furthe
//
contextStorage: {
    default: {
        module:"localfilesystem",
        config: {
            flushInterval:0.05,
            dir:"/data",
            base:"context",
            cache:false

        }
    },
},
```

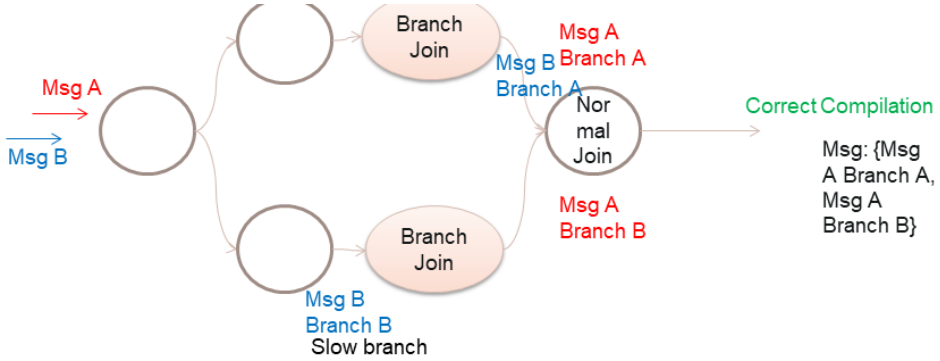


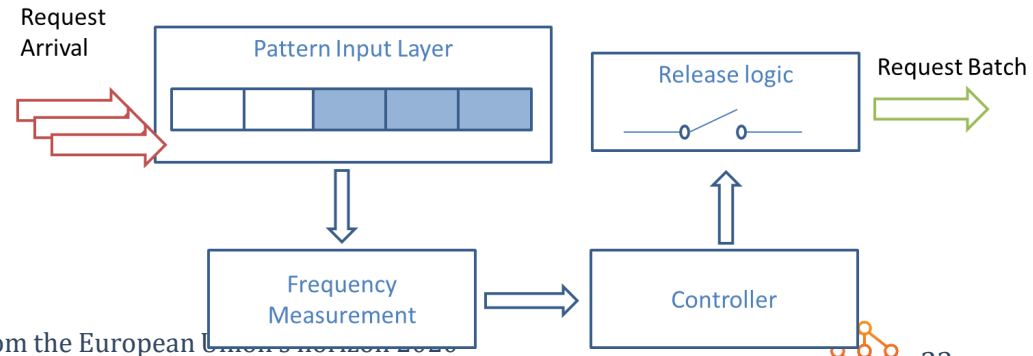


Other types of patterns

- Edge Extract-Transform-Load processes
  - Reliable data collection at the edge and pushing to a central storage
  - Function or Service mode
- Workflow primitives assistance
  - BranchJoin
- Request Aggregation
- Async read/writes
- Etc.







This project has received funding from the European Union's HORIZON 2020 research and innovation programme under grant agreement no 101017047

PHYSICS

Thank you for the attention!
George Kousiouris, HUA

www.physics-faas.eu

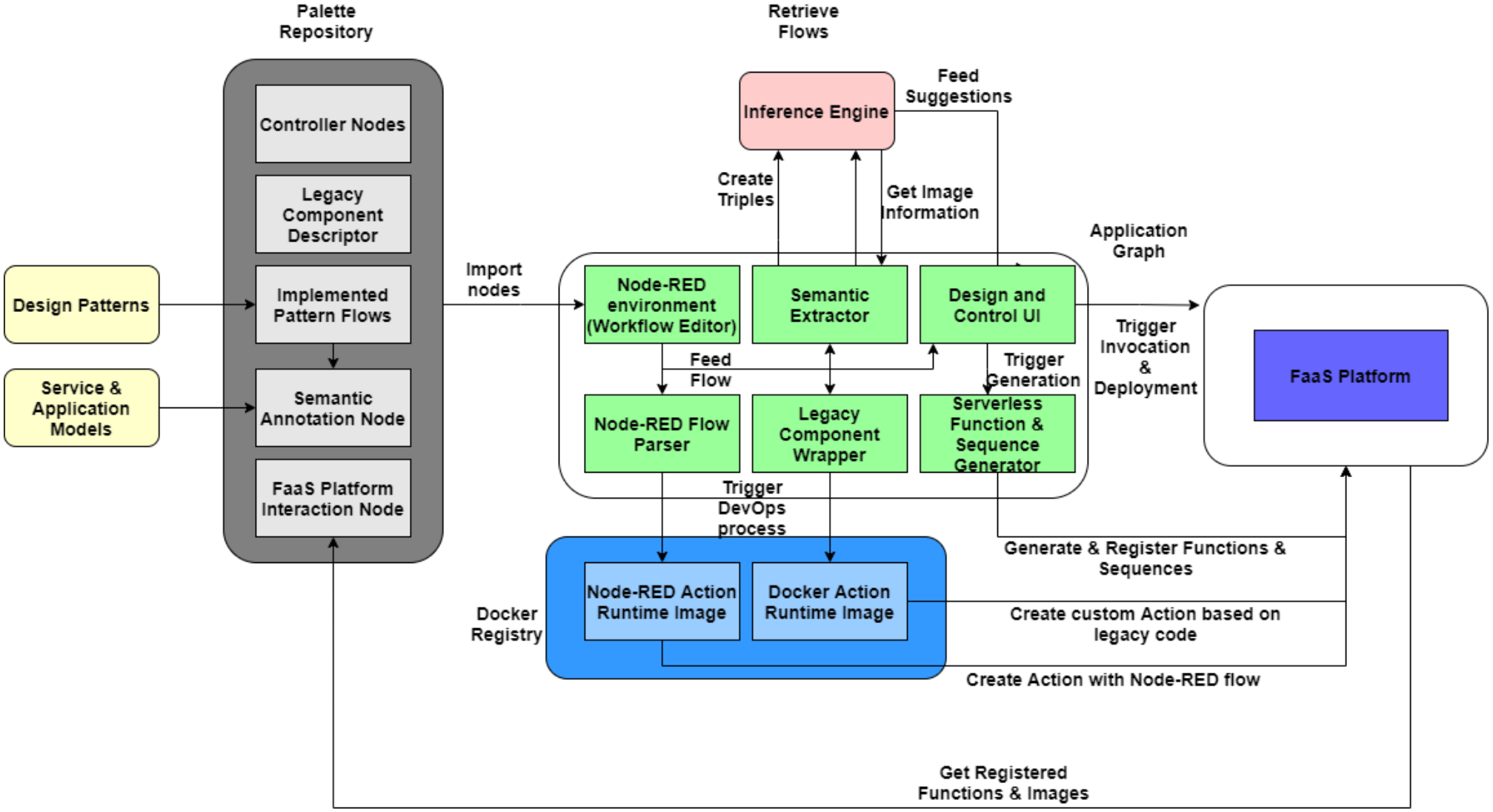
linkedin.com/company/physicsh2020

https://twitter.com/H2020Physics

info@physics-faas.eu

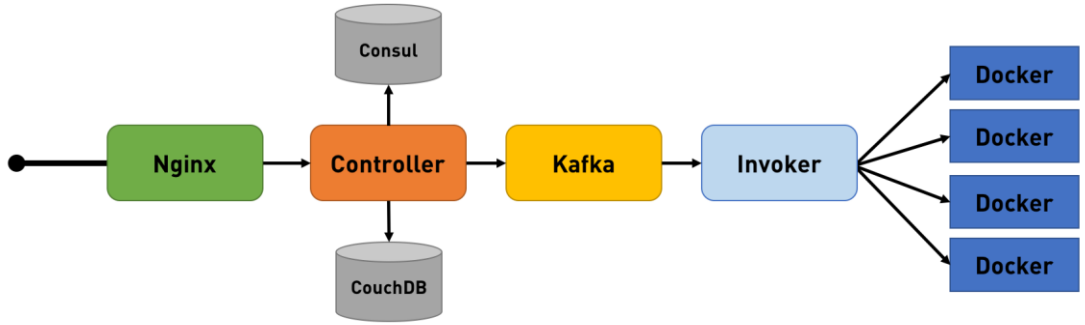
This project has received funding from the European Union's horizon 2020 research and innovation programme under grant agreement no 101017047

Design Environment Architecture



Backup slides: Openwhisk architecture and execution model

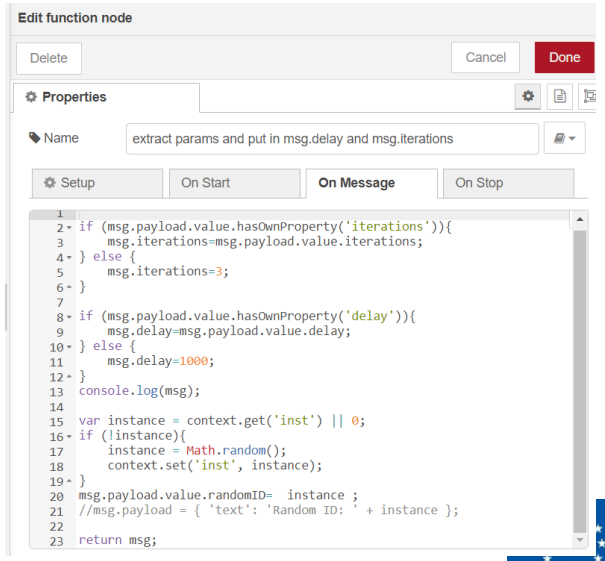
- Each function is based on a runtime image template
  - E.g. nodejs, java, arbitrary docker image
- Each runtime may have a number of prewarm containers
  - Containers that have been launched , are idle and waiting for incoming requests
- Each function may have a concurrency factor parameter
  - Maximum number of concurrent invocations of the function, the following ones are queued
- Whenever a function gets executed, the used container lingers in the system for some time
  - New invocation reuses that container
- If a new function invocation comes, if an existing runtime container is available it is reused otherwise a new one is spawned (assuming that the max concurrency factor has not been reached)
  - If it has, it waits until one of the executing containers finish

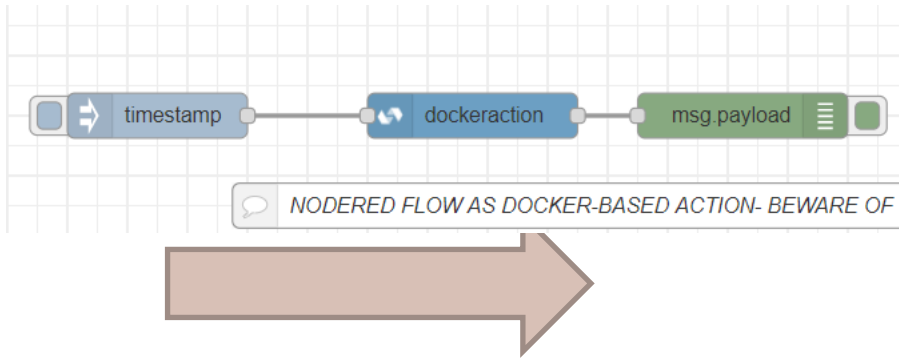


This project has received funding from the European Union's horizon 2020 research and innovation programme under grant agreement no 101017047

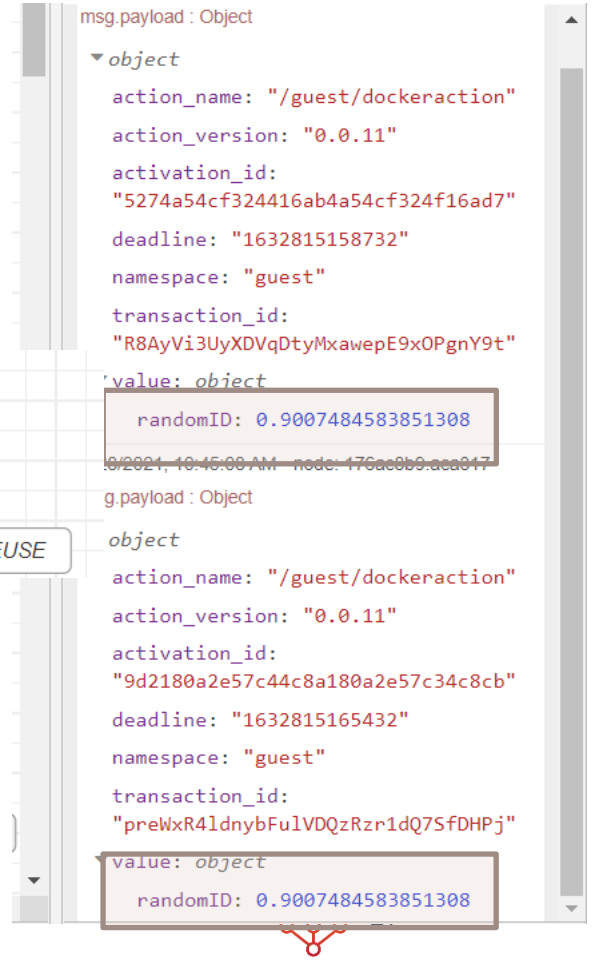
Example: Beware of Context Reuse!! Risk or Opportunity?

- Executed functions maintain the container for a period of time after execution (~15 minutes)
- Subsequent function invocations may use the same container (hot starts)
- This means that any variable that is initialized in the first invocation will maintain the value if not checked or refreshed!!!
- Example of random id (if not exists) variable in delay flow in two consecutive executions
  - Same environment





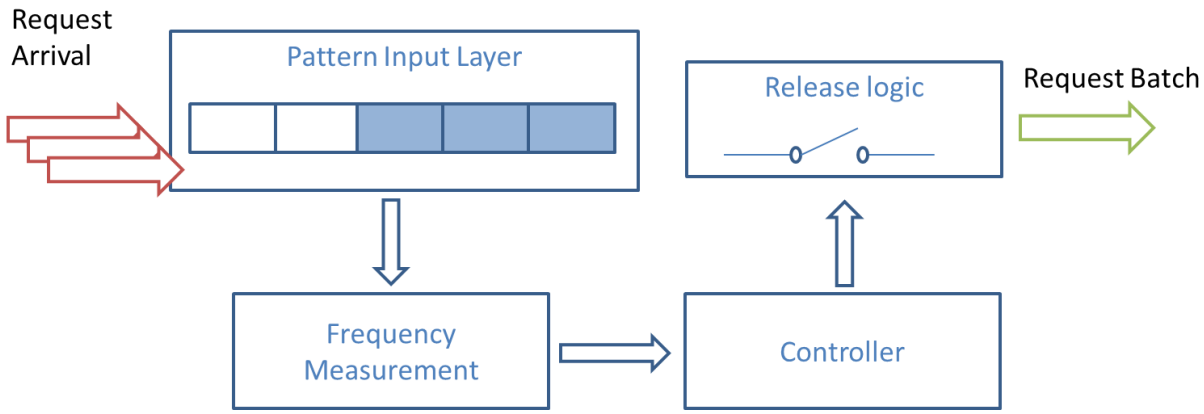
Invoking two times with a sufficient delay so that the first one has finished when the second arrives, but below the lingering warm container period

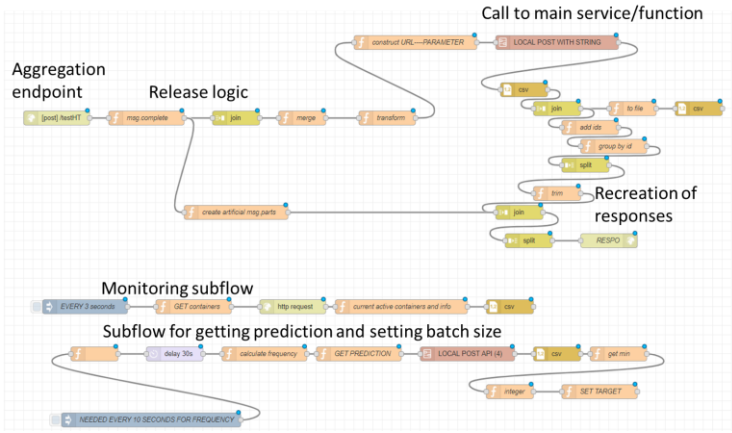


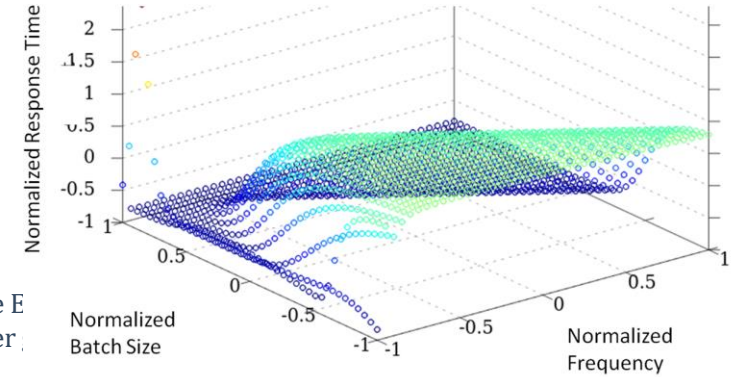
This project has received funding from the European Union's horizon 2020 research and innovation programme under grant agreement no 101017047

Batch Request Aggregation Pattern

- Use a **preprocessing** layer to batch incoming requests and launch one serving environment
- **Implemented as a reusable flow**
- **How and when** to release the batch
  - In an adapted and self-regulating manner
- Benefits
  - Minimize number of threads or containers needed compared to typical usages
  - Minimize costs in FaaS based on function invocation
  - Improve system performance **without increase** in resources in **varying traffic conditions**
- Full description in:
  - G. Kousiouris, "*A self-adaptive batch request aggregation pattern for improving resource management, response time and costs in microservice and serverless environments*", to appear in the 40th IEEE International Performance Computing and Communications Conference (IPCCC 2021)







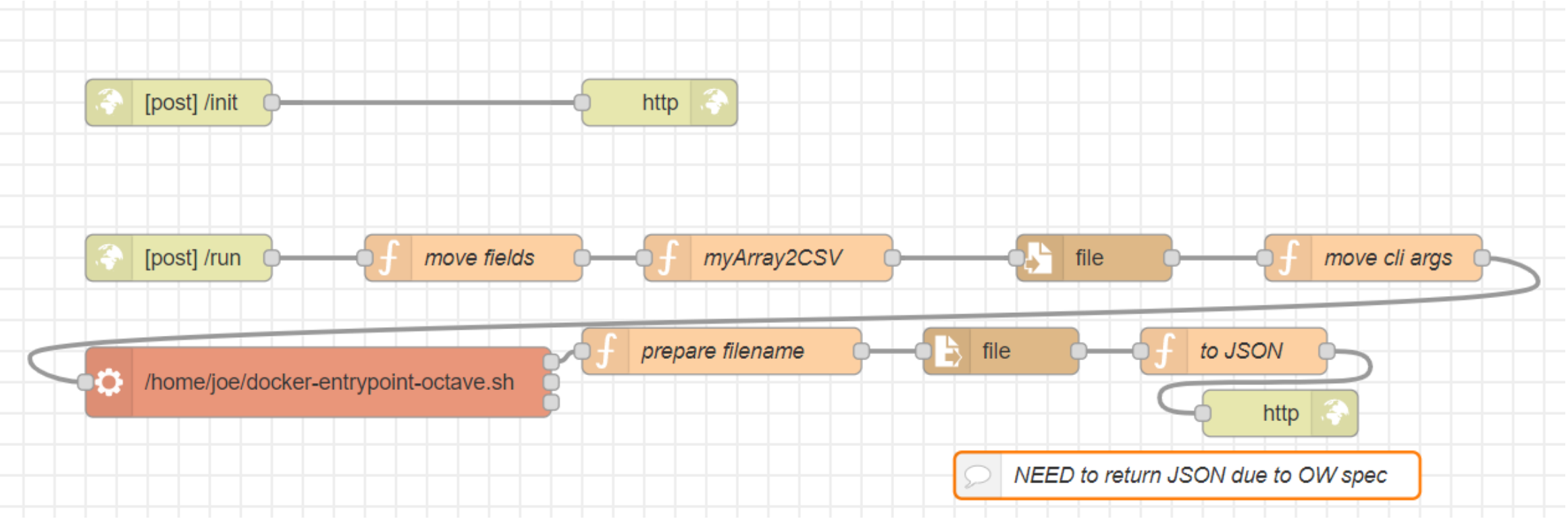
This project has received funding from the E
research and innovation programme under

Application structure

- What is an application
  - A collection of flows and services (support for microservices inclusion)
  - Flows consist of functions and subflows
    - Functions and subflows instantiate generic functions of Node-RED or new functions can be defined by developer
    - Functions can use any imported docker image
      - Has some OW requirements (POST /init and POST /run methods implemented)
  - Services can be internal services (to be deployed by the PHYSICS platform) or external static services (are not controlled by the PHYSICS platform, can not be changed and should be included for placement optimization purposes)
    - Examples of services
      - Storage
      - Eventing
      - External APIs etc.
      - Nodered flows created by developer but deployed as internal services
    - Services for which the image is not generated through PHYSICS DevOps are declared through the design environment (link to compose file) and included in the final app graph

This project has received funding from the European Union's horizon 2020 research and innovation programme under grant agreement no 101017047

Example 2: AI model prediction as a function



This project has received funding from the European Union's horizon 2020 research and innovation programme under grant agreement no 101017047