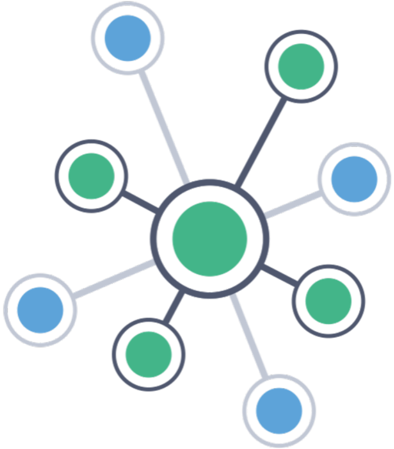
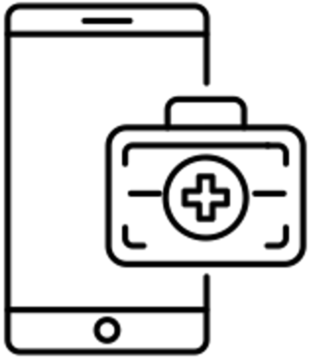
The EPI Framework:

A dynamic infrastructure to support healthcare use cases



Jamila Alsayed Kassem
MNS, UvA



Progress Report

- PoC deployed at UvA machines
  ⇒ Client – Server communication
  ⇒ via proxy and BFC's

- Implementation progress
  ⇒ Chaining of functions
  ⇒ Interface to change BFC's (command line)

- Experiments and evaluation
  ⇒ EPIF & BFC's resilience and adaptiveness
  ⇒ Stress test & scalability

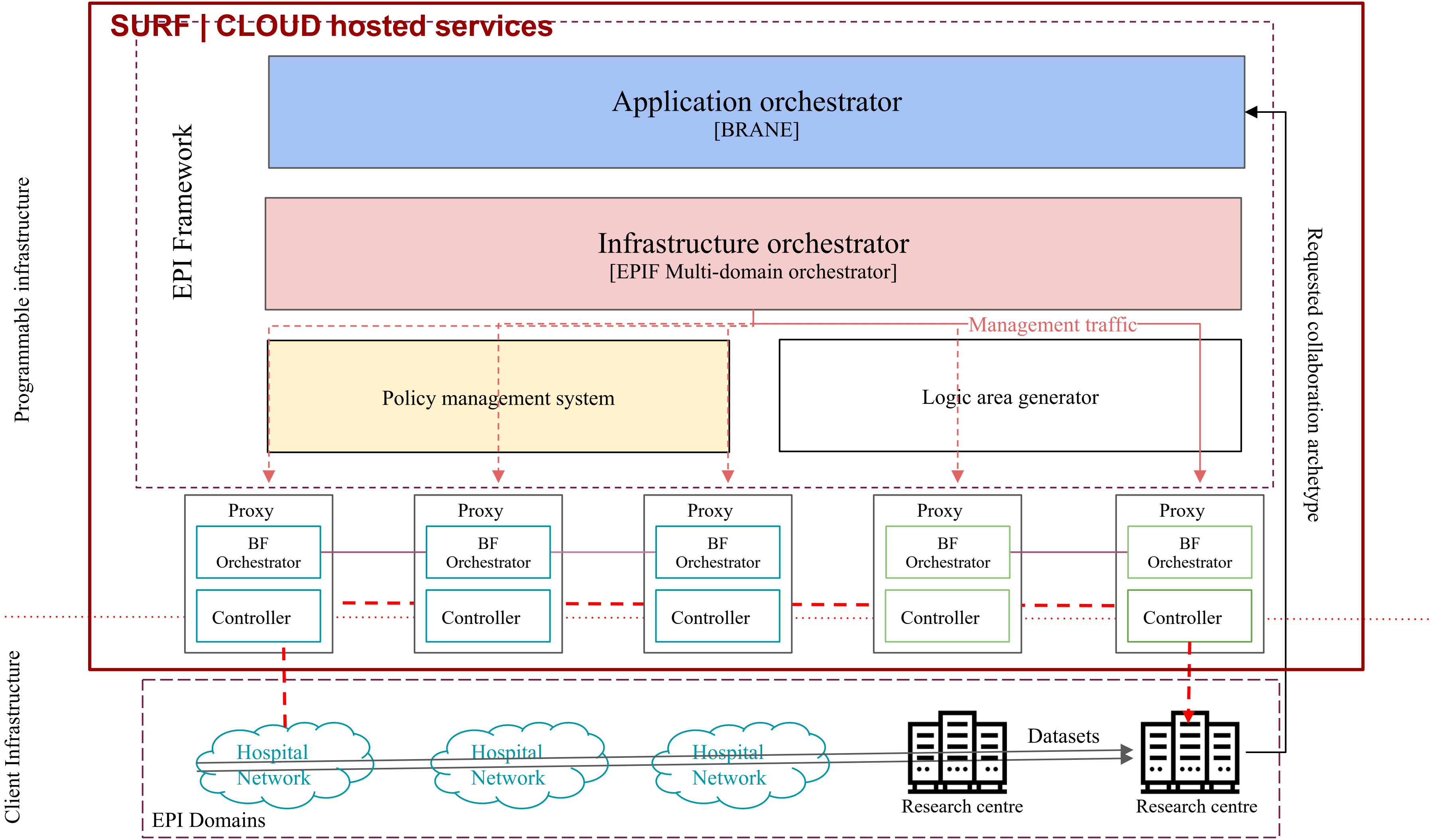
- Policy integration efforts
- Other discussions

EPIF: Background

Background & Challenges

- Data-sharing is key to enable EPI use-cases
- Challenges on many levels:
  ⇒ Policies and agreements Policy level
  ⇒ FAIR data Data level
  ⇒ Heterogeneous computing capabilities Application level
  ⇒ Heterogeneous security and networking Network level
- We focus on the last challenge
  ⇒ Network function virtualisation
  ⇒ Containers are efficient → bridging function chain
  ⇒ Dynamicity and programmability
- Setting security requirements to address this heterogeneity

EPIF: The Architecture



SURF | CLOUD hosted services

Programmable infrastructure

EPI Framework

Application orchestrator
[BRANE]

Infrastructure orchestrator
[EPIF Multi-domain orchestrator]

Management traffic

Policy management system

Logic area generator

Requested collaboration archetype

Proxy
BF Orchestrator
Controller

Proxy
BF Orchestrator
Controller

Proxy
BF Orchestrator
Controller

Proxy
BF Orchestrator
Controller

Proxy
BF Orchestrator
Controller

Client Infrastructure

Hospital Network

Hospital Network

Hospital Network

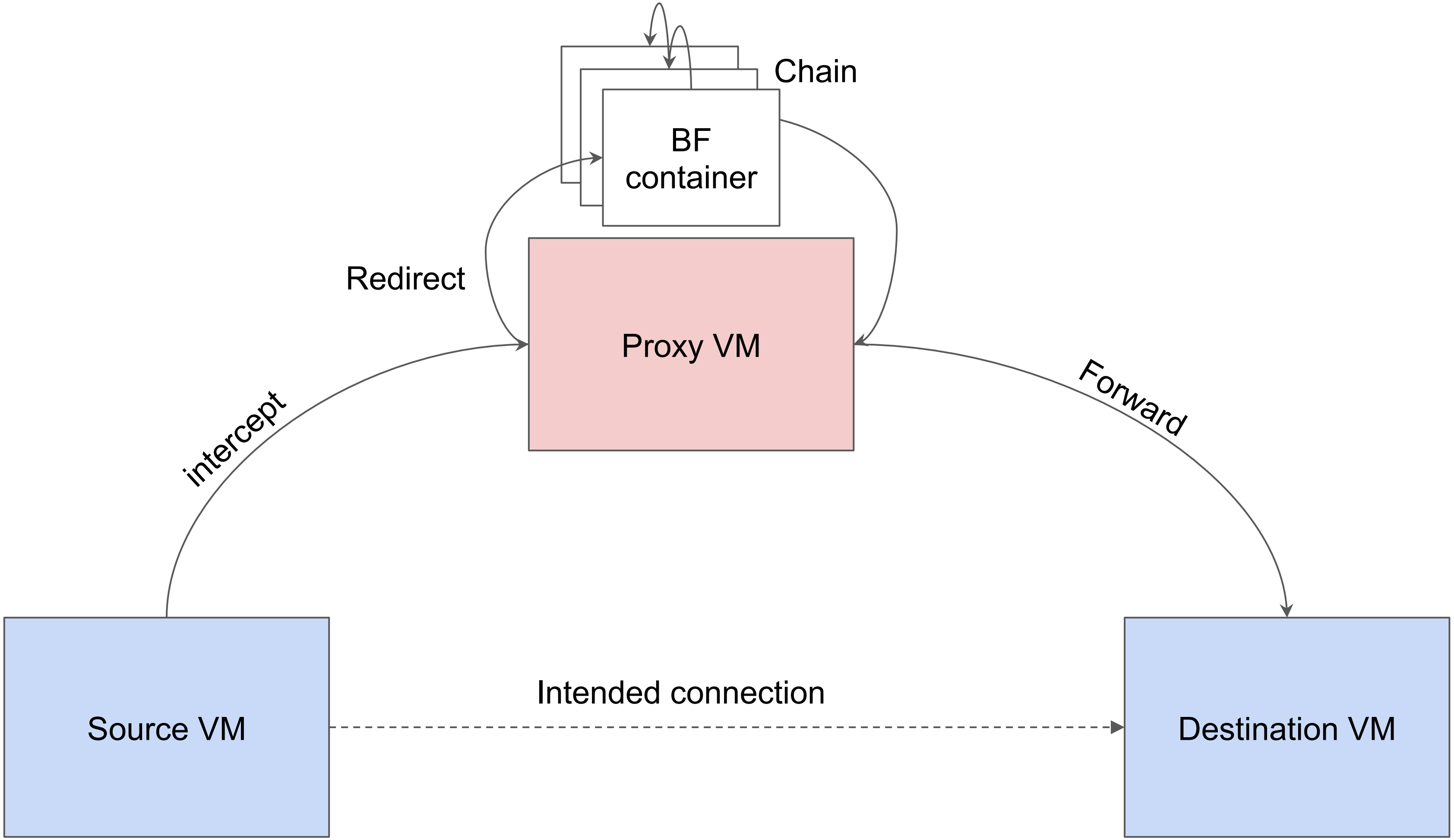
Datasets

Research centre

Research centre

EPI Domains

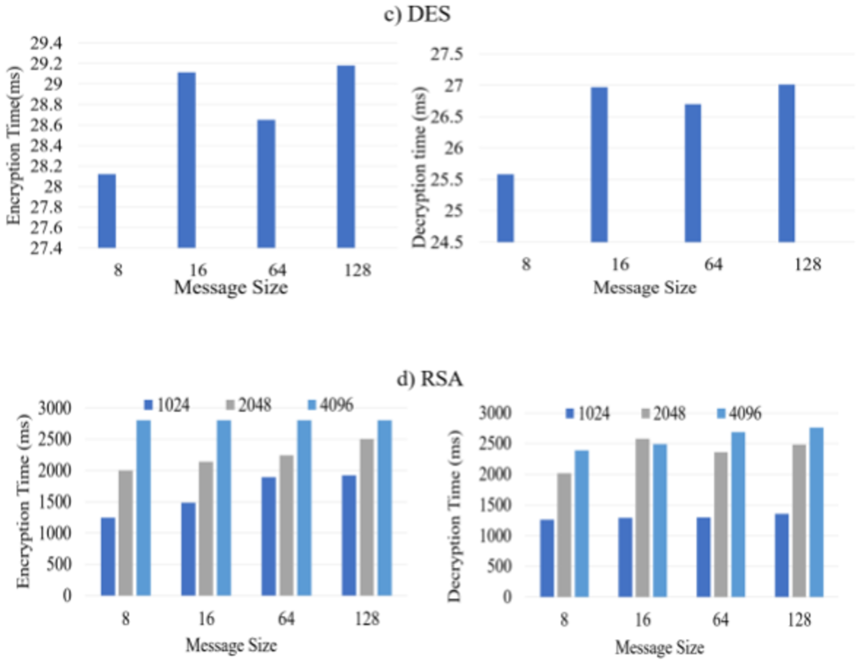
Proxy Implementations


Chain
BF container
Redirect
Proxy VM
intercept
Forward
Intended connection
Source VM
Destination VM

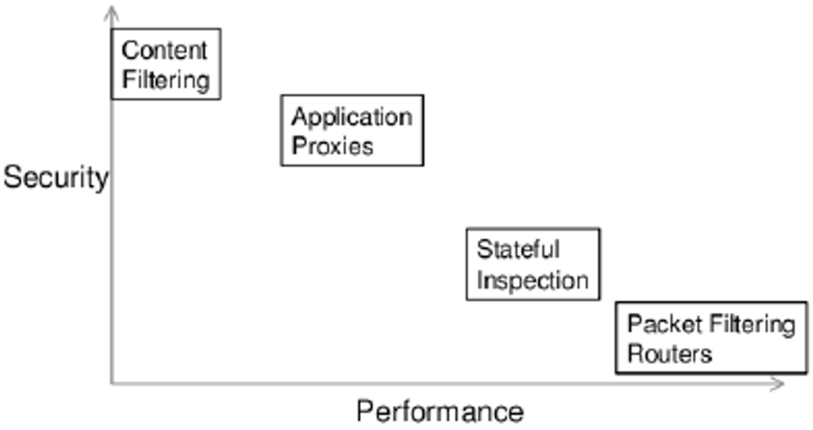
Experiments

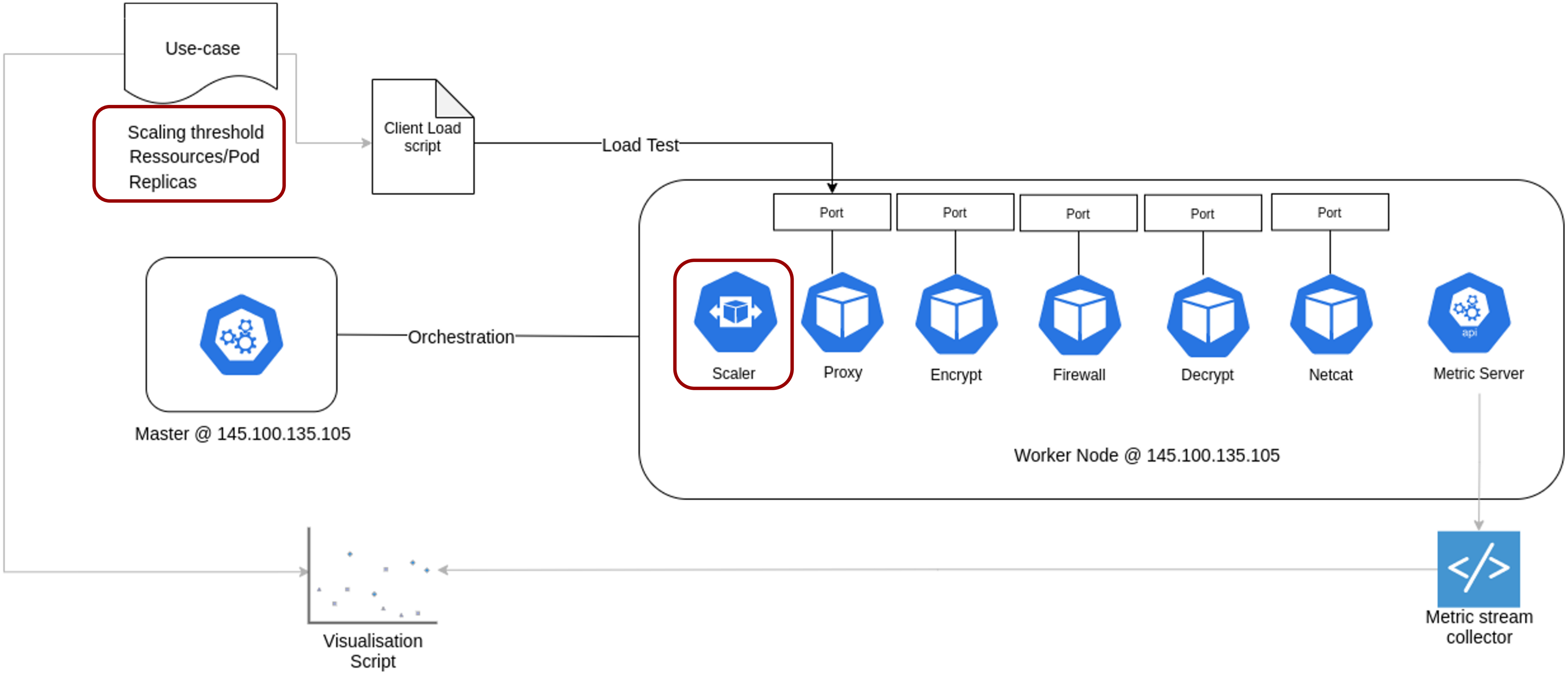
Experiment Questions

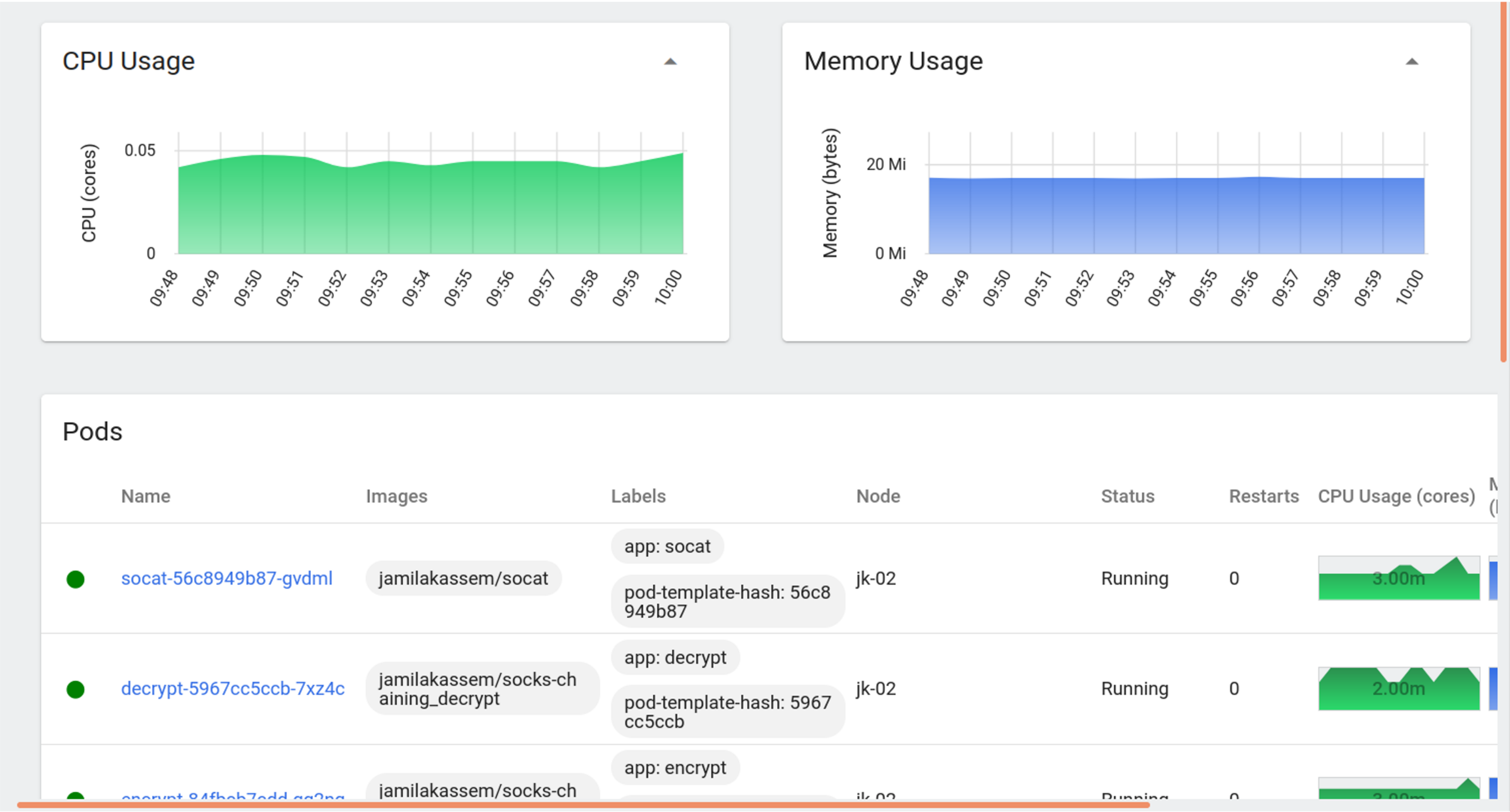
- Adding the VNF adds to the complexity and communication overhead, how to adapt the setup to avoid the EPIF being a bottleneck?
  - Scaling and load testing is the right approach
  - We need to provision and adapt BFC, but how?
    - Different scalers
    - Minimise time overhead
    - Maximise availability (no downtime)
- Identify how different BF’s scale in terms of CPU and memory, and how performance is utilised best under which scaling strategy?
  - Set up a link with BF chain in between, and place different scalers
  - It might be the case that setting up different performance requirements / BF is relevant



Performance vs. Security




Use-case
Scaling threshold
Ressources/Pod
Replicas
Client Load script
Load Test
Port
Port
Port
Port
Port
Orchestration
Scaler
Proxy
Encrypt
Firewall
Decrypt
Netcat
api
Metric Server
Master @ 145.100.135.105
Worker Node @ 145.100.135.105
Metric stream collector
Visualisation Script



CPU Usage

Memory Usage

Pods

	Name	Images	Labels	Node	Status	Restarts	CPU Usage (cores)
●	socat-56c8949b87-gvdml	jamilakassem/socat	app: socat pod-template-hash: 56c8949b87	jk-02	Running	0	3.00m
●	decrypt-5967cc5ccb-7xz4c	jamilakassem/socks-chaining_decrypt	app: decrypt pod-template-hash: 5967cc5ccb	jk-02	Running	0	2.00m
●	encrypt-84fbeb7edd-gg2ng	jamilakassem/socks-ch	app: encrypt	jk-02	Running	0	3.00m

Ongoing & Future Work

Experimentation wise:

- Deploy different scalers
  - Horizontal, Vertical, Reinforcement
- Solidify results
- Define strategy:
  - Reaction time
  - Wasted resources
  - Performance requirement
  - Violation time
- Targeted venue IEE cloud 2022
  - 1st March

Research and open questions:

- Security performance and experiments
- Policy – EPIF communication
- ***How do we deploy this within the hospital infrastructure?***
  - ***Acceptability?***
  - ***Hospital IT***
- ***Other feasible architecture implementation?***
  - ***Blockchain hosting policy***
  - ***Blockchain enforcement and audits***
- Deploy & apply use-cases
  - SURF