



ALMA MATER STUDIORUM
UNIVERSITÀ DI BOLOGNA



SLICES-RI Summer School 2026
11 June, Madrid, Spain

The SLICES Cloud Continuum Blueprint: Experimenting With Cloud Continuum Infrastructures

Armir Bujari, Andrea Sabbioni, Paolo Bellavista

Department of Computer Science and Engineering (DISI)
Alma Mater Studiorum - University of Bologna, Italy

Outline

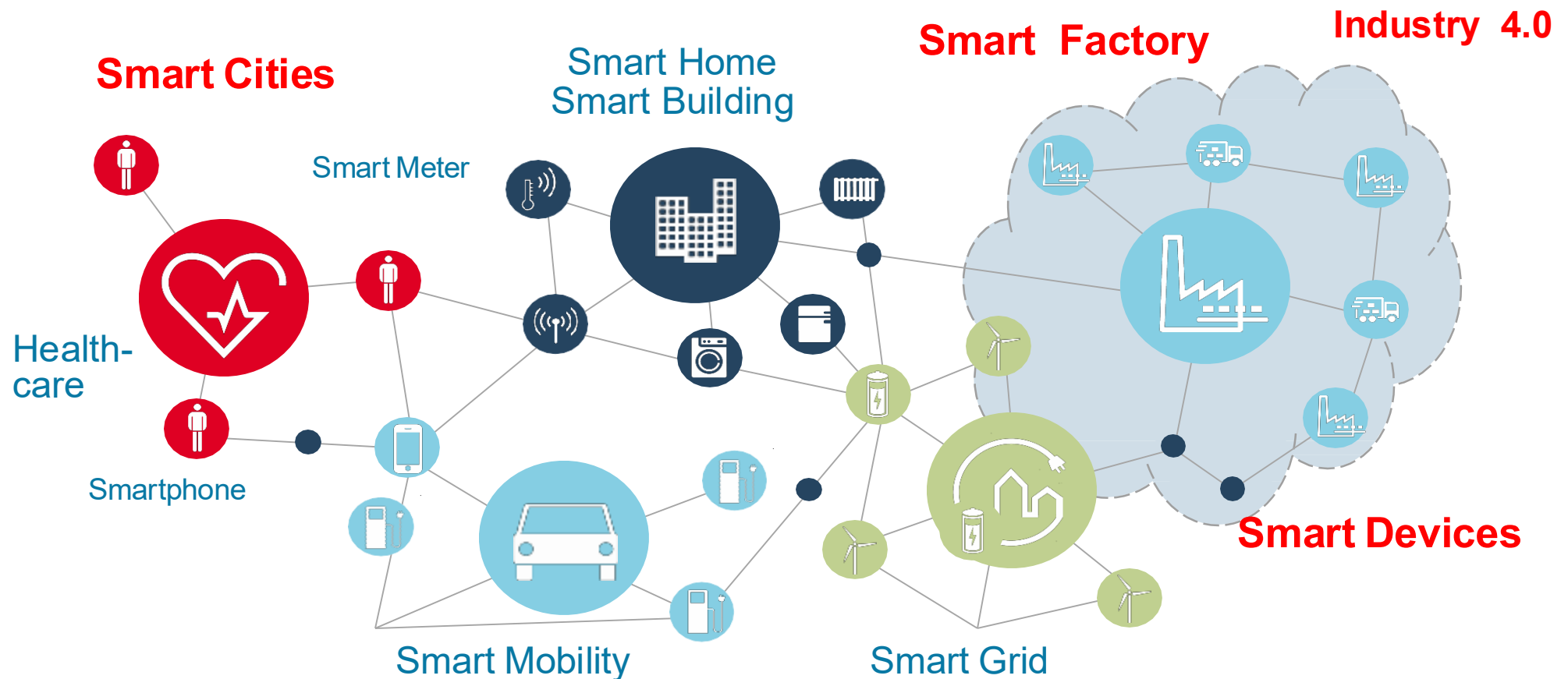
- Beyond the 5G-BP
- Definition & Rationale of the Cloud Continuum (CC)
- CC-BP Domain Model
- Hands-on Tutorial
 - Basic aaS model
 - Example Experiments
- Conclusion & roadmap



DISI – University of Bologna

Mobile Middleware Research Group

<https://site.unibo.it/middleware/>



Background: Blueprint methodology

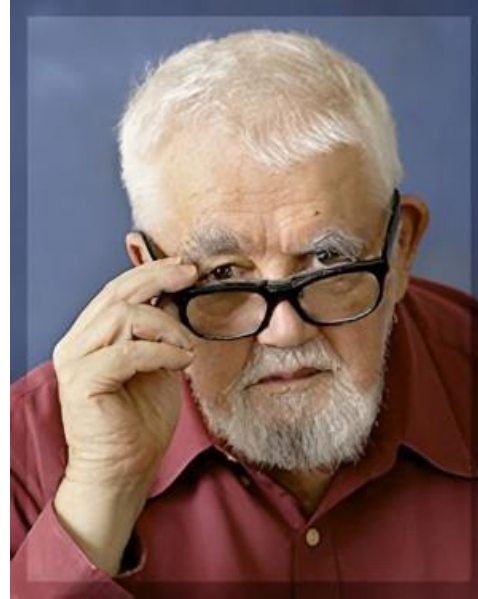
- Bottom up
 - Start **building** a series of post-5G testbeds
 - **Future proof**
 - Based on a well-defined and agreed architecture
 - Open components
 - Disaggregated
 - **Practical**
 - Use current (state-of-the-art, but viable and mature) technologies
- Results
 - B5G blueprint up and running
 - More and more nodes connected and compliant
- Desirable
 - Replicate in CC-BP what has been done in the 5G component
 - Guarantee compatibility (approach) with the 5G component

Extension Beyond 5G

- CC-BP target
 - Serve the research community working in the **IoT-Edge-Cloud (continuum)**, with **large-scale, distributed, and possibly federated deployment** environments in mind
- What it is **NOT**
 - Only a cloud platform to automatize/support the 5G component of the blueprint (and its deployment) in an easy way
- What it **is**
 - A component directed to **research communities other than 5G**
 - Covering
 - Cloud, edge, IoT, cloud continuum, ...
 - Compatible with what has been done already for the 5G component of the SLICES blueprint



A Little Historical Note on the Cloud



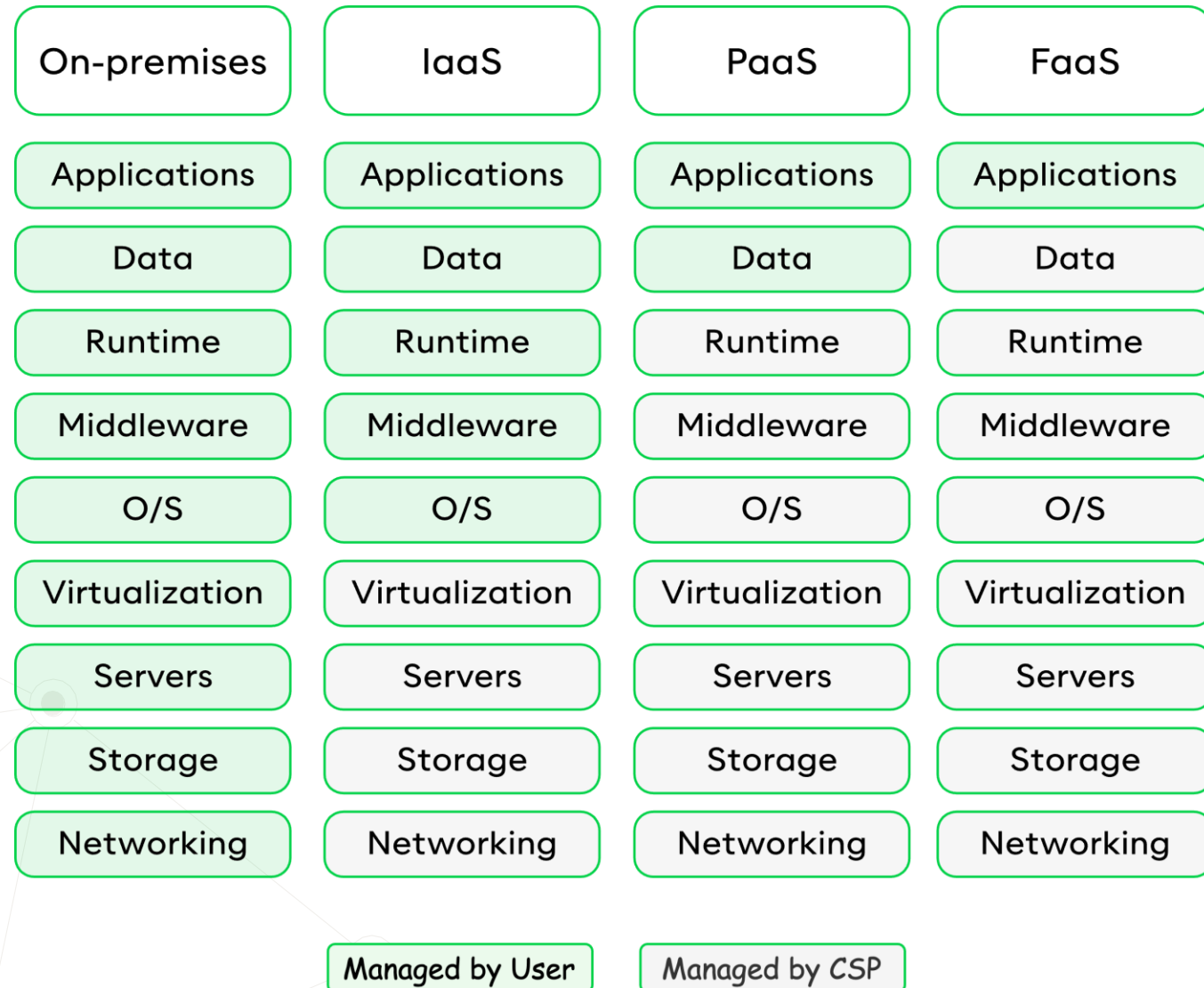
Computing may someday be organized as a public utility just as the telephone system is a public utility. Each subscriber needs to pay only for the capacity he actually uses, but he has access to all programming languages characteristic of a very large system ...

Certain subscribers might offer service to other subscribers.

(Professor John McCarthy, 1961)



Traditional vs. IaaS vs. PaaS vs. FaaS



Modern Computing Paradigms

Cloud computing

Mobile cloud computing

Fog computing

Edge computing

"A model for enabling convenient, on-demand network access to a shared pool of configurable computing resources (e.g., networks, servers, storage, applications and services) that can be rapidly provisioned and released with minimal management effort or service provider interaction".

P. Mell, T. Grance, et al., The NIST Definition of Cloud Computing, Computer Security Division, National Information Technology Laboratory, 2011.



Modern Computing Paradigms

Cloud computing

Mobile cloud computing (MCC)

Fog computing

Edge computing

“A mobile device that can execute a resource-intensive application on a distant high-performance compute server or compute cluster and support thin client user interactions with the application over the Internet.”

N.I.M. Enzai, M. Tang, A taxonomy of computation offloading in mobile cloud computing, in: 2014 2nd IEEE International Conference on Mobile Cloud Computing, Services, and Engineering, 2014, pp. 19-28.



Modern Computing Paradigms

Cloud computing

Mobile cloud computing

Fog computing

Edge computing

“The process of extending Cloud Computing capabilities at the edge of the network. Fog incorporates computing, storage and network resources close to the IoT layer to facilitate the data processing”

F. Bonomi, R. Mito, J. Zhu, S. Addepalli, Fog computing and its role in the Internet of Things, in; Proceedings of the First Edition of the MCC Workshop on Mobile Cloud Computing, 2012, pp. 13-16.



Modern Computing Paradigms

Cloud computing

Mobile cloud computing

Fog computing

Edge computing

“A key technology to assist wireless networks with Cloud Computing-like capabilities to provide low-latency and context-aware services directly from the network Edge.”

S. Kekki, W. Featherstone, Y. Fang, P. Kuure, A. Li, A. Ranjan, D. Purkayastha, F. Jiangping, D. Frydman, G. Verin, et al., MEC In 5G networks, ETSI White Paper 28 (2018) 1-28.





Cloud Continuum

An extension of the traditional Cloud towards multiple entities (e.g., Edge, Fog, IoT) that provide analysis, processing, storage, and data generation capabilities.



Characteristics



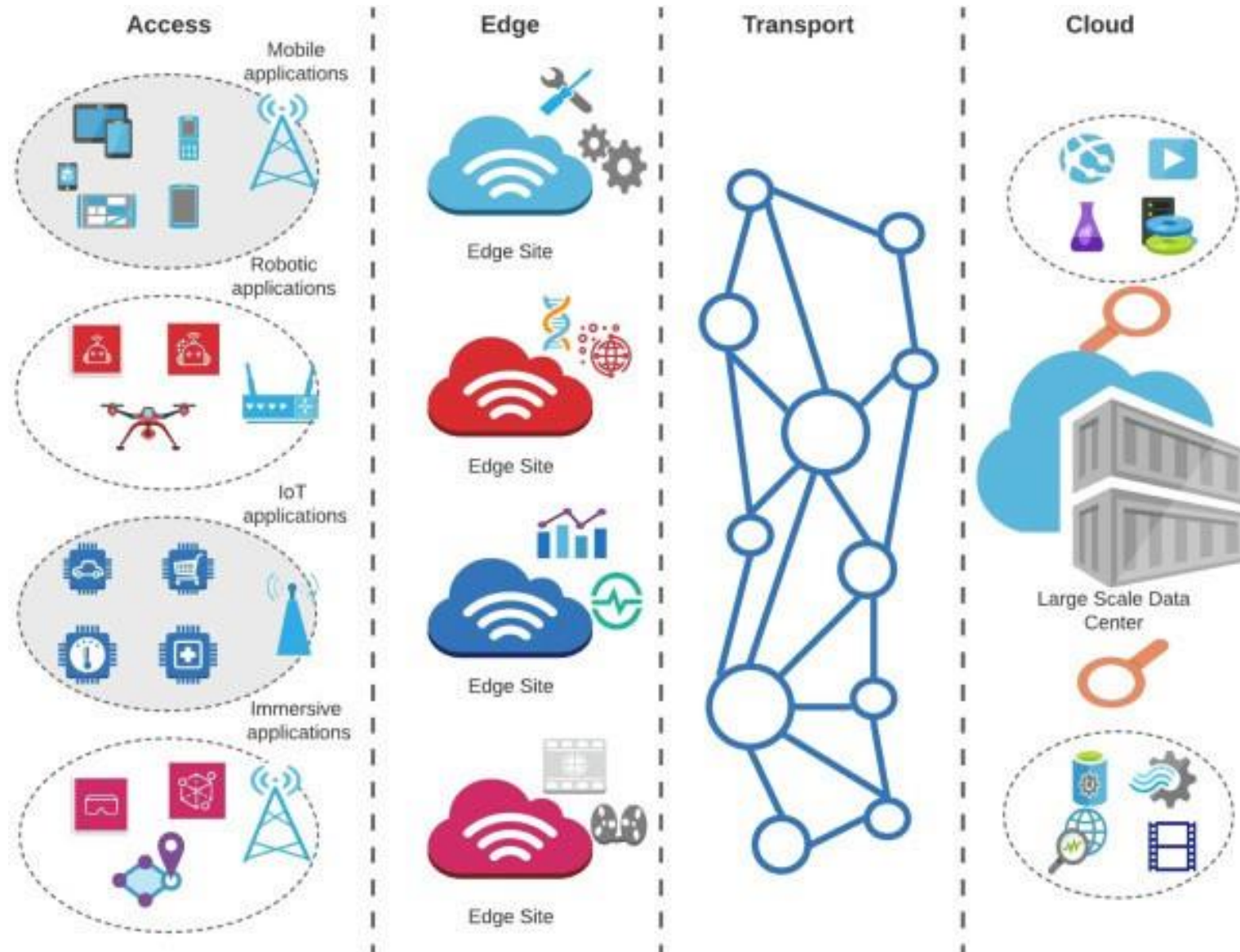
HETEROGENEOUS



DISTRIBUTED



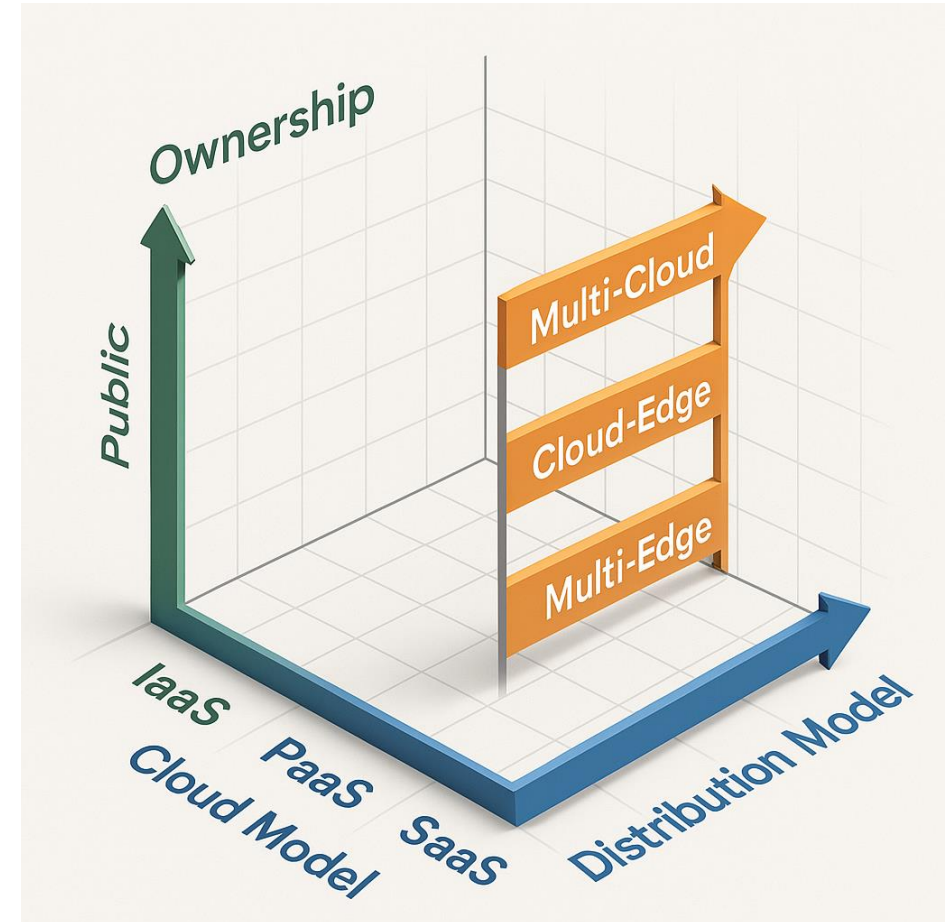
REQUIREMENTS



Cloud Continuum

The Cloud Continuum (CC) refers to the seamless integration of cloud technologies and environments across various deployment **models**, **locations**, and **services** [1]

- Heterogeneous abstractions (IaaS, PaaS, SaaS, FaaS, BaaS, ...) & resources (IoT, low-power nodes, Next Generation Networks, Tensor PU, AGVs, rovers, ...)
- Different types of distribution (Geo-replication, Multi-Cloud, Cloud-Edge, Multi-Edge)
- Different ownership and billing models (Private, Public, Federated)
- Support to **different verticals and requirements** (Distributed AI, Low-latency, Big Data processing, Resilient Services, Energy Efficiency, ...)



The Goal: Types of Studies/Experiments(1/2)

1. Architectural Studies

Focus: Investigate how various architectural paradigms impact performance, scalability, and resource utilization

Key experiments:

- **Hybrid and Multi-Cloud Architectures:** Experiment with workload distribution across private, public, and hybrid clouds
- **Edge-Cloud Integration:** Evaluate data processing and orchestration between edge devices and cloud platforms
- **Serverless Architectures:** Test function-as-a-service (FaaS) models for scalability and latency
- **Cloud-Native Microservices:** Assess the design of loosely coupled microservices optimized for distributed environments

Example: Researchers propose a dynamic workload allocation algorithm across the edge-cloud, where both the edge and cloud might be involved in data processing. The experimenter, selects an IoT use case, such as a smart factory monitoring system, with varying data volumes and processing requirements, assessing their proposal (end-to-end latency, energy consumption, network utilization, and cost) against a static allocation strategy in the targeted edge-cloud deployment

2. Performance and Optimization Studies

Focus: Optimize latency, bandwidth, compute, and storage for diverse applications

Key experiments:

- **Latency-Sensitive Applications:** Test real-time processing workloads (e.g., autonomous vehicles, AR/VR)
- **Dynamic Resource Scaling:** Evaluate autoscaling and cost optimization for fluctuating workloads
- **Energy Efficiency:** Measure energy consumption at different layers of the CC
- **Data Placement and Caching:** Investigate strategies for minimizing data transfer times while maintaining consistency

Example: Researchers proposing intelligent dynamic data placement & replication strategy would like to assess how their solution affects query performance, storage costs, and network overhead in a CC environment. The experimenter selects the geographically distributed infrastructure (storage) assets, a data partitioning strategy, measuring query latency (read/write operations), bandwidth usage between edge and cloud, storage costs for replication, etc. in a realistic infrastructure



The Goal : Types of Studies/Experiments(2/2)

3. Security and Compliance Studies

Focus: Ensure data integrity, confidentiality, and regulatory compliance across the CC

Key experiments:

- **Zero-Trust Architectures:** Test access controls and encryption methods in multi-tenant setups
- **Data Sovereignty:** Experiment with localization and compliance with GDPR, HIPAA, or similar regulations
- **Attack Simulations:** Measure resilience to DDoS, ransomware, and insider threats in hybrid deployments
- **Edge Security Mechanisms:** Evaluate lightweight security protocols suitable for IoT and edge devices

Example: Researchers proposing new security models in CC deployments would like to validate the effectiveness of zero-trust principles (e.g., least privilege access, continuous verification, etc.) in securing the CC deployment where multiple (simulated) users and application access resources across the continuum. The experimenter selects a threat simulation model and measures the time to detect/block attacks, system resilience, and user experience under restricted access

4. Application-Centric Studies

Focus: Tailor CC solutions to the needs of specific application domains

Key experiments:

- **IoT and Smart Cities:** Study the interplay of sensors, gateways, and cloud for real-time analytics
- **AI/ML Workloads:** Test distributed model training and inference pipelines
- **Industry 4.0:** Experiment with cloud/edge-based orchestration for manufacturing and automation
- **Content Delivery and Media Streaming:** Optimize delivery through edge caching and cloud-based encoding

Example: Researchers interested in testing the feasibility of real-time traffic AI analysis would like to assess a new vertical application optimizing city traffic flows. The setup involves cameras and sensors at intersections to monitor traffic density, using edge devices for initial image/video analysis and cloud servers for aggregation and advanced analytics. The experimenter, selects to compare the algorithmic approach relying on edge-only processing, cloud-only processing, and a hybrid edge-cloud setup



SLICES CC BluePrint Domain Model

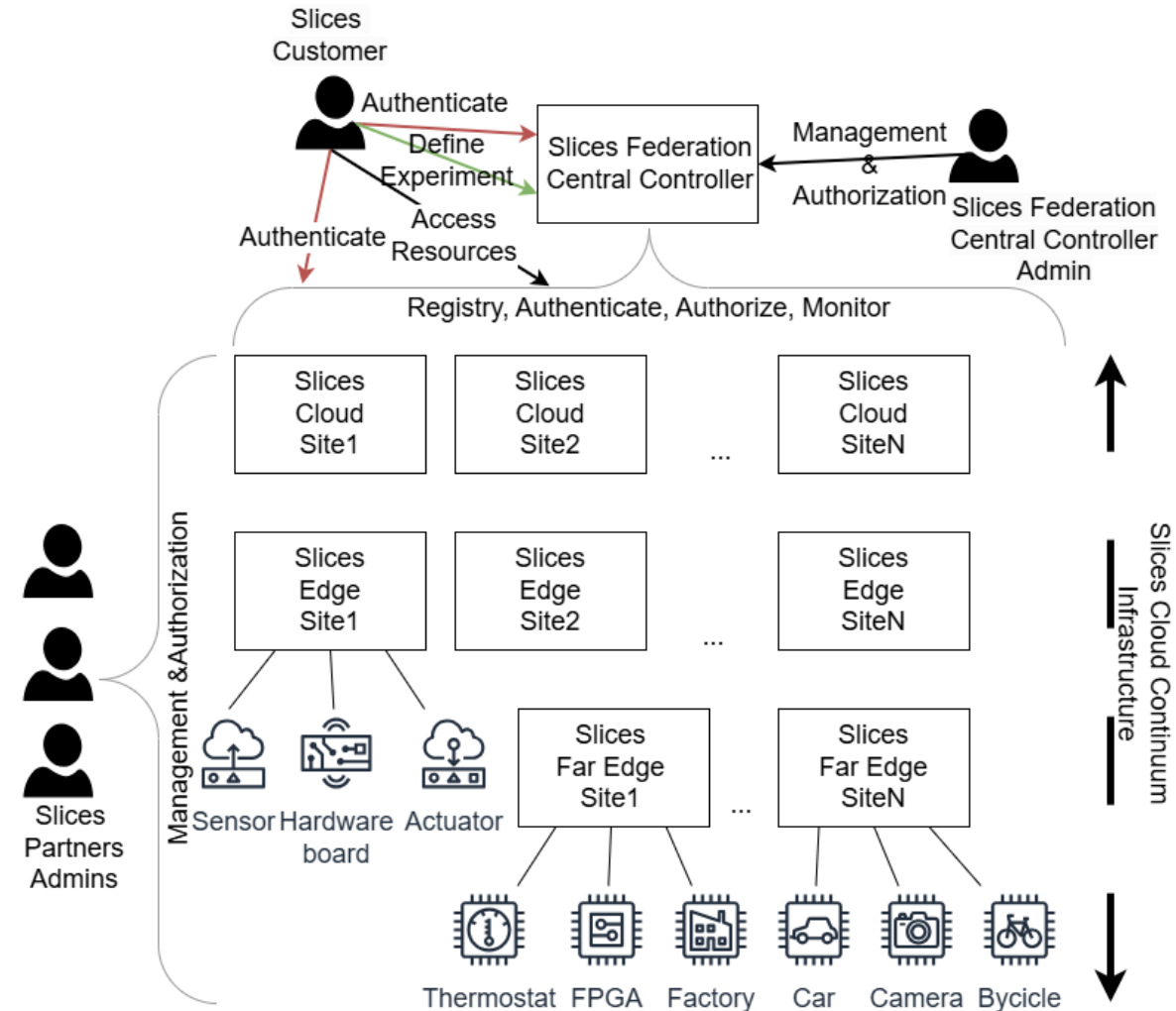
The **SLICES CC BluePrint (CCBP)** is a **modular** and **extensible** federated CC s/w infrastructure tailored to support CC experiments

Objective of the CCBP:

- Provide a **simplified** and **reproducible** testing environment
- Establish a guiding architecture and approach for the creation of an EU-wide infrastructure supporting CC experiments

Key design guidelines of the CCBP include:

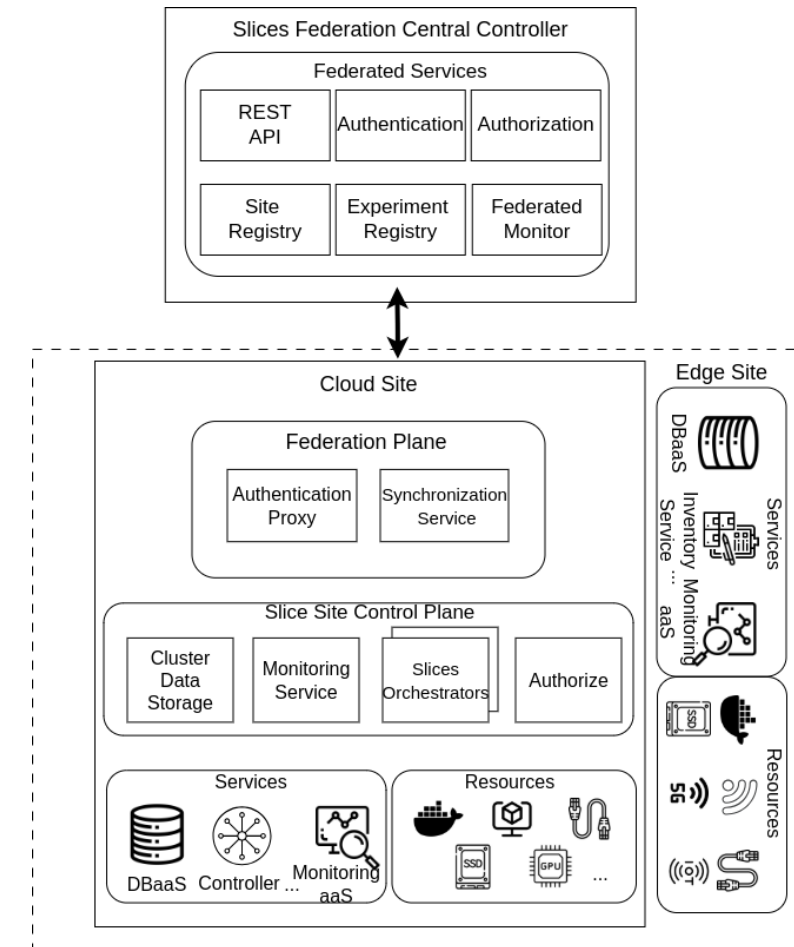
- Dynamic join/leave of sites (resources)
- **Resiliency** to failures and network partitions
- Support to **different cloud models** and heterogeneous h/w
- Support to **high-level programming interfaces**
- Also, **application-oriented design**



The CCBP 1st release (1/2)

The first Release of the CCBP has proposed a model to:

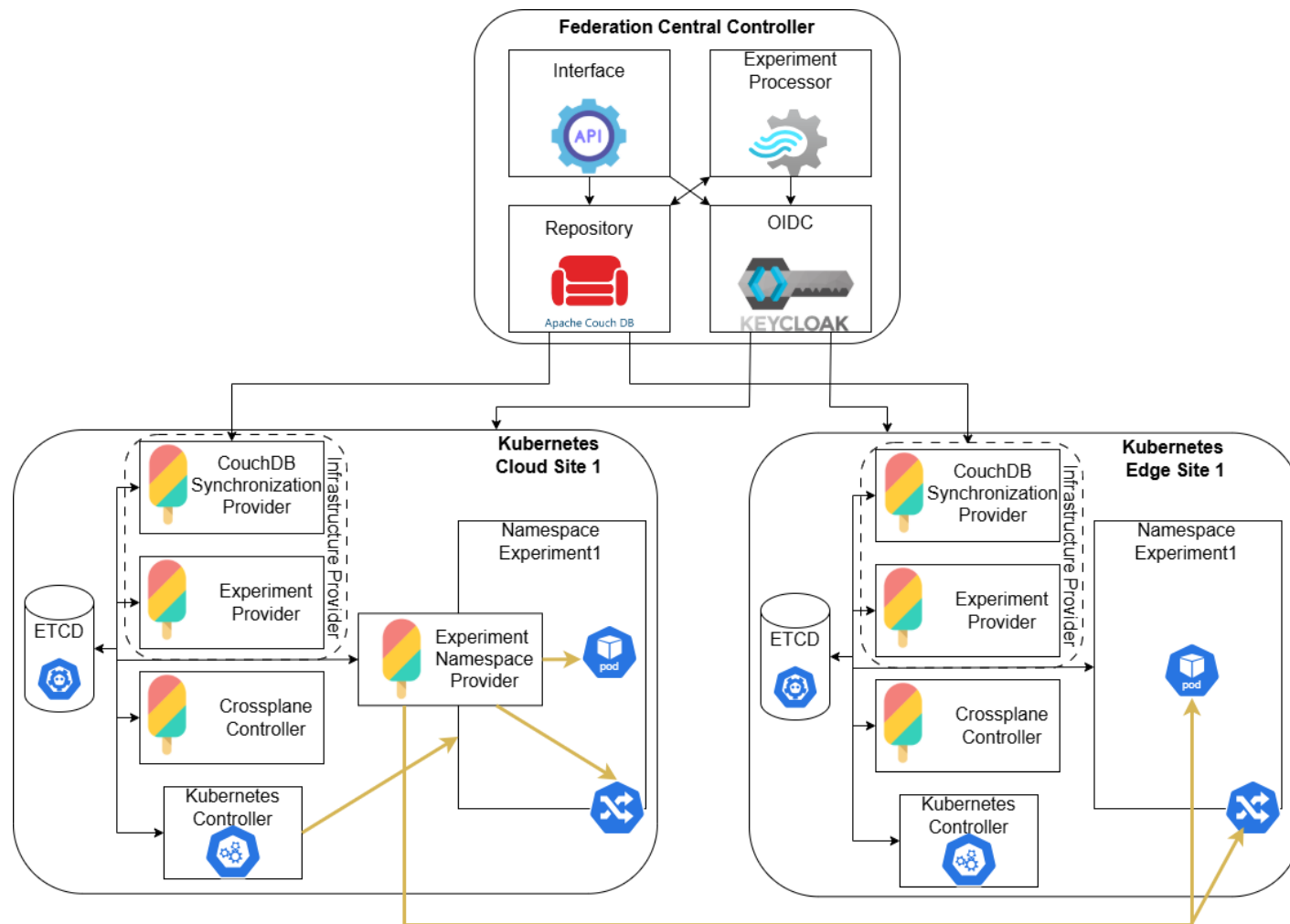
- Federate sites with heterogeneous capabilities
- Guarantee resiliency of the federation in case of failures
- Maintain independence of federated sites in:
 - Policy
 - Resource Provisioning
- Standardized resource provisioning across continuum
- Enable the creation and assignment of resource partitions o
- Support experimenters with high level orchestration over cloud continuum resources



The CCBP 1st release (2/2)

First Release Implementation:

- Kubernetes-first Resource representation
- Slice = Cloud Continuum Namespace
- Operators to provision resources
- Crossplane Compositions
- Asynchronous, fault-tolerant synchronization of SFCC and sites
- Federated Login through OIDC protocol
- Multi-Cluster Operator



Deployment Options:

- Quick start on local machine
- Self Deployment

CCBP 2nd release

From 1st Release → 2nd Release

- From infrastructure blueprint → Cloud Continuum Blueprint-as-a-Service (CCBP-aaS)
- From manual/self deployment → integrated, hosted SLICES-BI environment
- From resource federation focus → towards full experiment lifecycle support (Dev ↔ Infra ↔ Experiment ↔ Monitoring)

New CCBP Components:

- CC-BP CLI
- SFCC All-in-One
- Instrumented Slices Sites



Tutorial



Hands on objectives

1. Create multiple Kubernetes Clusters on the Slices BI
2. Deploy services across multiple Kubernetes Cluster
3. Monitor the state and logs of deployed services
4. Observe changes in monitored metrics when infrastructure is modified

Dev/Ops workflow for experiments with the CCBP

Development Environment

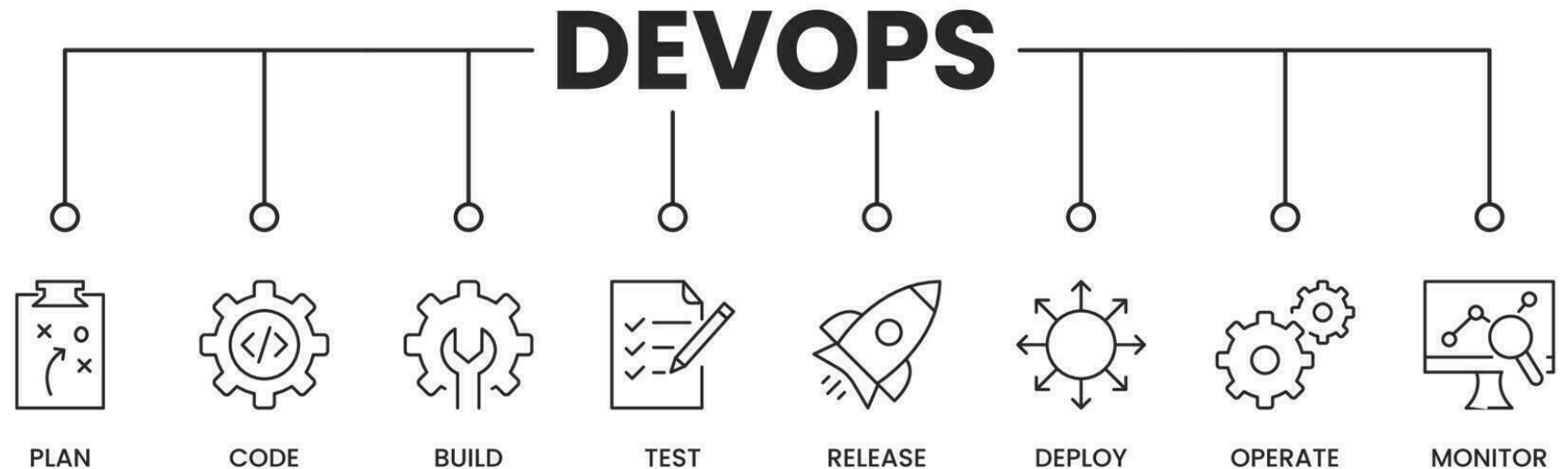
- CCBP-CLI
- VPN-aaS

Infrastructure Setup

- VPN-aaS
- SlicesBi Operator
- Kubernetes Operator

Experiment Execution

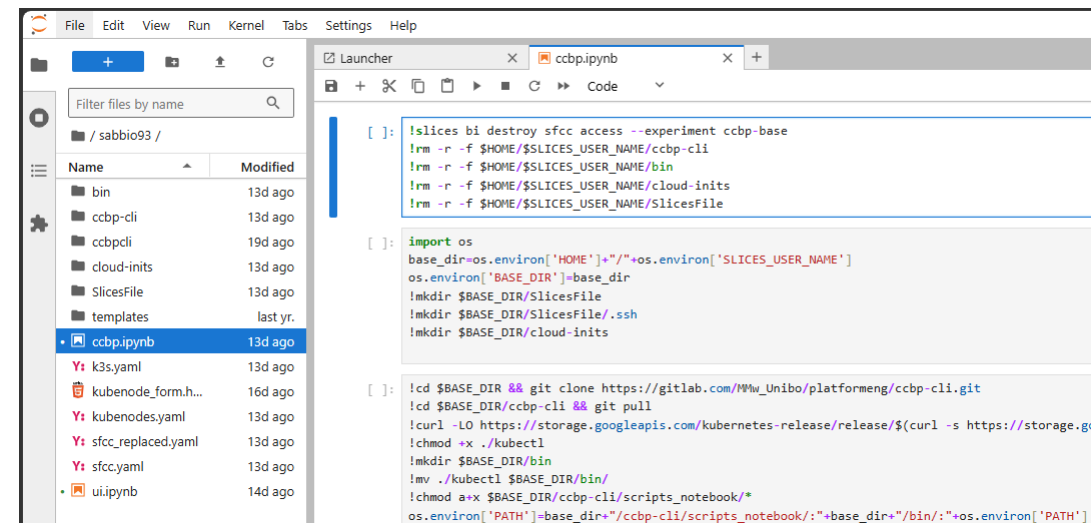
- Monitoring-aaS
- Extended Sync Operator



The CCBP-Command Line Interface

The CCBP command line interfaces(CLI) is a utility environment that:

- Simplifies development environment setup
- Automates the provisioning of the CCBP on the SLICES-B infrastructure
- Supports the experimenters with repetitive tasks
- Stateless and portable
- Multi-modal:
 - Local development(Docker container)
 - Remote VM(on SLICES-BI or any other Cloudinit compliant provisioner)
 - SLICES-BI Notebook
- Includes dev/ops tools (slices bi cli, Kubectl, vim)



https://gitlab.com/MMw_Unibo/platformeng/ccbp-cli

The Files of the CCBPCLI

In every configuration the CCBPCLI creates three directories which are preserved and portable across invocations and different installation.

SlicesFile/.ssh

Contains the ccbpkey automatically created and registered at initial setup of ccbp

SlicesFile/.slices

Contains authentication files used by the SLICES BI CLI

SlicesFile/VPNConfigs

Default directory where the CCBPCLI downloads configuration to connect to the VPN

SlicesFile/kubeconfigs

Default directory where the CCBPCLI stores Kubernetes configuration file used to interact with the SFCC and created clusters



The CCBPCLI: Local Development

Advantages:

- Direct access to all needed tools
- Local coding and configuration of resources
- Direct communication with
 - Services
 - Clusters
 - UIs

Needed Software:

- Docker
- Kubectl
- Wireguard

```
mkdir -p $HOME/SlicesFile/.ssh  
mkdir -p $HOME/SlicesFile/VPNConfigs  
mkdir -p $HOME/SlicesFile/kubeconfigs
```

```
docker pull  
registry.gitlab.com/mmw_unibo/platformeng/c  
cbp-cli:latest && docker tag  
registry.gitlab.com/mmw_unibo/platformeng/c  
cbp-cli:latest ccbpcli:latest
```

EXPERIMENTAL!

For Everyday use: ./install.sh



The CCBPCLI: Remote VM

With the Remote VM option we configure a SLICES BI VM with all the dependencies and the CCBPCLI already setuped. The VM is in the same network of the CCBP services and resource created on the BI

```
$ wget gitlab.com/MMw Unibo/platformeng/ccbp-cli/-/raw/main/cloud-  
inits/ccbpcli.yaml?ref type=heads -O ccbpcli.yaml  
$ slices bi --infra-id be-gent1-bi-vm1 create dev --flavor medium --user-data  
./ccbpcli.yaml --experiment dev --wait  
$ slices bi ssh dev --experiment dev
```

Suggested setup:

- Setup the SFCC
- Use remote development extensions (e.g. in Visual Studio Code)
- Consider to connect to the VPN to directly access UI or use SSH tunnels

Needed Software:

- Wireguard?



The CCBPCLI: Notebook

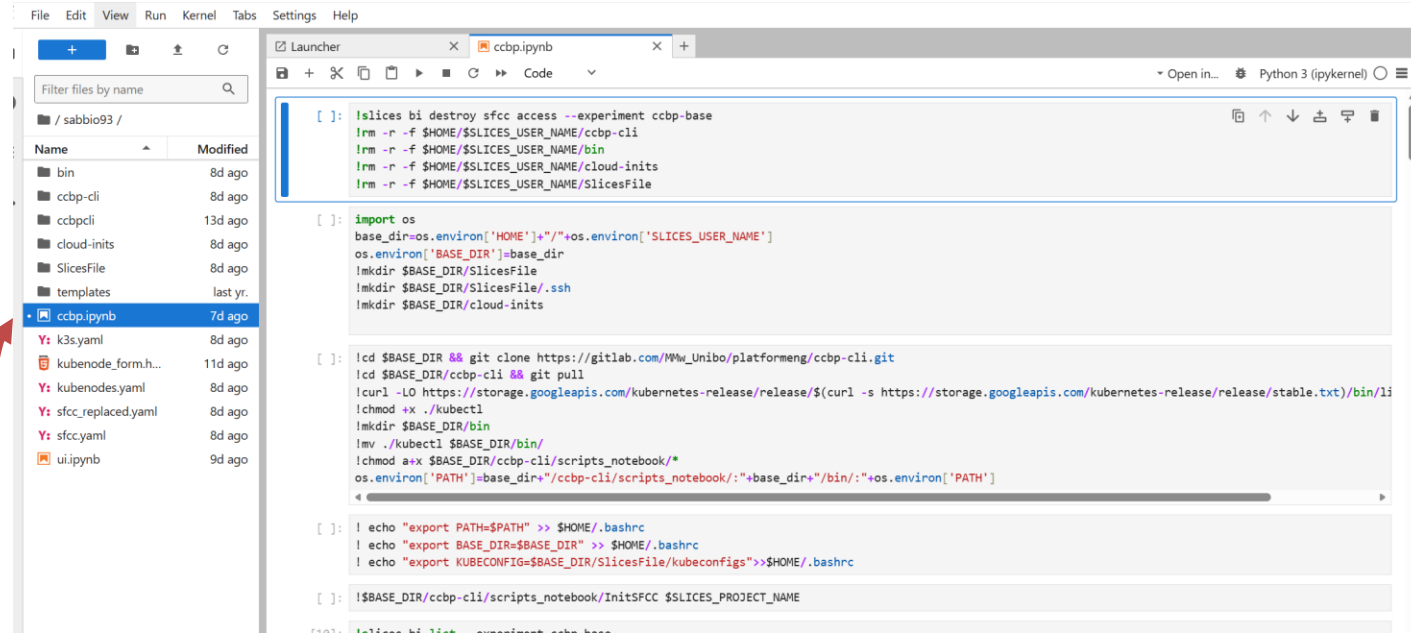
The CCBPCLI is also available integrated as scripts executed inside the SLICES-BI Notebook Service

Advantages:

- No software needed
- Extremely cross platform
- VPN not needed

Disadvantages:

- Limited development environment
- Setup slower
- No access to UIs



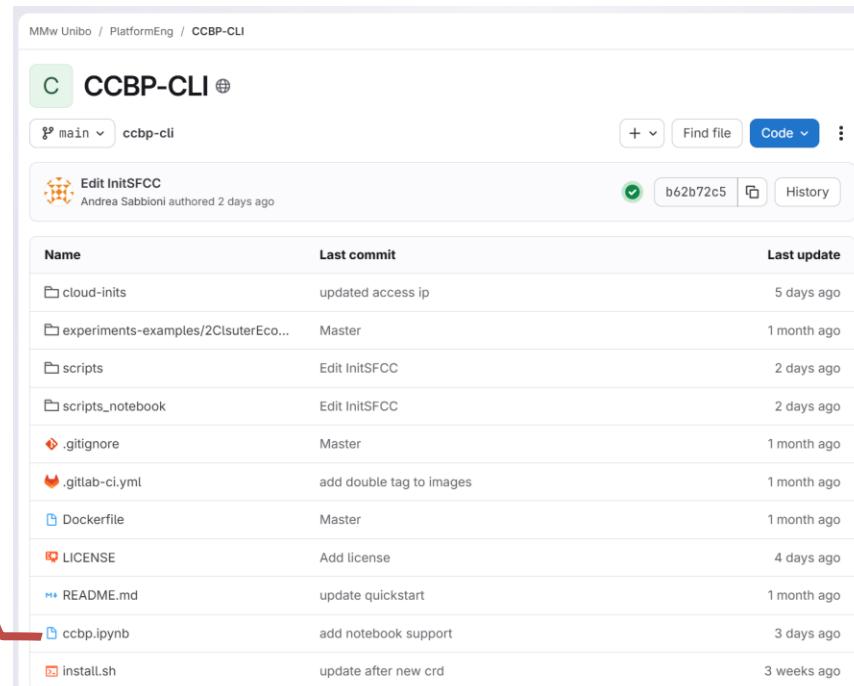
```
[ ]: !slices bi destroy sfcc access --experiment ccbp-base
!rm -r -f $HOME/$SLICES_USER_NAME/ccbp-cli
!rm -r -f $HOME/$SLICES_USER_NAME/bin
!rm -r -f $HOME/$SLICES_USER_NAME/cloud-inits
!rm -r -f $HOME/$SLICES_USER_NAME/SlicesFile

[ ]: import os
base_dir=os.environ['HOME']+"/"+os.environ['SLICES_USER_NAME']
os.environ['BASE_DIR']=base_dir
mkdir $BASE_DIR/SlicesFile
mkdir $BASE_DIR/SlicesFile/.ssh
mkdir $BASE_DIR/cloud-inits

[ ]: !cd $BASE_DIR && git clone https://gitlab.com/MMw_Unibo/platformeng/ccbp-cli.git
!cd $BASE_DIR/ccbp-cli && git pull
!curl -LO https://storage.googleapis.com/kubernetes-release/release/$(curl -s https://storage.googleapis.com/kubernetes-release/release/stable.txt)/bin/linux-amd64/kubectl
!chmod +x ./kubectl
!mkdir $BASE_DIR/bin
!mv ./kubectl $BASE_DIR/bin/
!chmod +x $BASE_DIR/ccbp-cli/scripts_notebook/*
os.environ['PATH']=base_dir+"/ccbp-cli/scripts_notebook/"+base_dir+"/bin/"+os.environ['PATH']

[ ]: !echo "export PATH=$PATH" >> $HOME/.bashrc
!echo "export BASE_DIR=$BASE_DIR" >> $HOME/.bashrc
!echo "export KUBECONFIG=$BASE_DIR/SlicesFile/kubeconfigs">>$HOME/.bashrc

[ ]: !$BASE_DIR/ccbp-cli/scripts_notebook/InitSFCC $SLICES_PROJECT_NAME
```



Name	Last commit	Last update
cloud-inits	updated access ip	5 days ago
experiments-examples/2CIsuterEco...	Master	1 month ago
scripts	Edit InitSFCC	2 days ago
scripts_notebook	Edit InitSFCC	2 days ago
.gitignore	Master	1 month ago
.gitlab-ci.yml	add double tag to images	1 month ago
Dockerfile	Master	1 month ago
LICENSE	Add license	4 days ago
README.md	update quickstart	1 month ago
ccbp.ipynb	add notebook support	3 days ago
install.sh	update after new crd	3 weeks ago



The CCBPCLI:

1. Local
2. Jupyter Notebook
3. Remote Machine

	Local	Remote Machine	Jupyter Notebook
Dependencies	Docker, Kubectl, Wireguard	Wireguard?	Wireguard?
Services, UI	Direct access	Trough ssh Tunnel	Only Rest

Interoperability among CCBPCLIs options:

- The CCBPCLI is “stateless”
- Is always possible to download SlicesFile folder and change CCBPCLI mode
- Recovery options trough Slices PK, SFCC kubeconfig



SFCC All-in-one Solution

The SFCC All-in-One integrates the capabilities of the SFCC and a Site in a **single** deployment

- Acts as the central entry point for experiment definition and management
- Coordinates configuration, automation, and monitoring across all Slice Sites
- Provides a logically centralized **control plane** over distributed resources

Unified Abstraction Layer

- Exposes resources and services as Kubernetes-compliant objects
- Enables a **declarative** model for experiment specification
- Ensures consistency and portability across heterogeneous environments

Automation & DevOps Integration

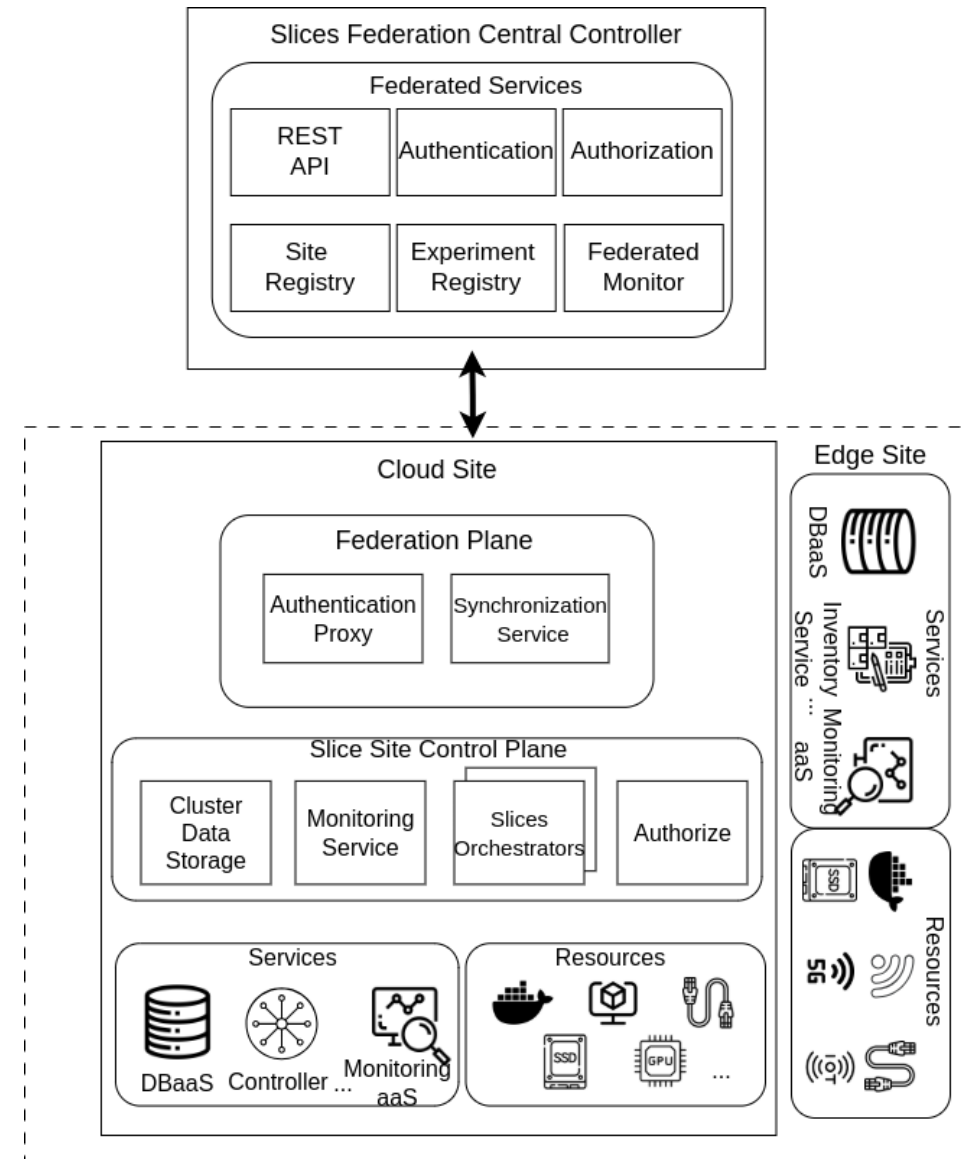
- CLI-driven interaction for programmatic control
- Seamless integration with:
 -  GitLab CI/CD pipelines
 -  Terraform infrastructure provisioning
 -  Helm-based application deployment
- Enables Infrastructure-as-Code (IaC) and **reproducible** experiments



SFCC All-in-one: Services

SLICES Federation Central Controller services include:

- Site & Experiment Registry
- Authentication Service
- REST API
- VPN-aaS
- SLICESBI-Operator
- Monitoring-aaS
- Secret/Configuration Store
- DNS-aaS
- Multi-Site Controllers



Initialize the CCBP

The script InitSFCC <project_name> initializes the CCBPCLI by creating two virtual machines:

- SFCC: A deployment all in one of the SFCC
- Access: the VPN server exposing a public IP

By default, if not passed as 2 argument to the script, the VMs are create in the *ccbp-base* experiment

```
andrea@SURFACESVENTURA:~$ ccbpcli
root@5355a454bac0:/# slices project^C
root@5355a454bac0:/# slices bi list --experiment ccbp-base
You must be logged in to use this command ✖
root@5355a454bac0:/# InitSFCC ccbpcli
Please go to https://portal.slices-ri.eu/oauth/authorize\_device?user\_code=QVXH-SRCR and login to continue.
Successfully logged in as 'sabbio93'
📁 Saved to /root/.slices/auth.json
The following projects are available: ccbp, ccbptronics, ccbpcli
👉 Now select the project that you want to use with slices project use <project name>
The current project was set to 'ccbpcli', in which you are a lead and which expires on 2027-04-28 00:00 UTC.
The current project was set to 'ccbpcli', in which you are a lead and which expires on 2027-04-28 00:00 UTC.
Experiment ccbp-base already exists. Skipping creation.
Generating public/private rsa key pair.
Your identification has been saved in /root/.ssh/ccbp_key
Your public key has been saved in /root/.ssh/ccbp_key.pub
The key fingerprint is:
SHA256:D66U0dNmICwtBeUQWuSNkpupSrvV0+Qd9Mho0F1aqA root@5355a454bac0
The key's randomart image is:
+---[RSA 4096]-----+
|..0=0 o o..|
|.00.00 = o. .|
|.B.=E o.o o .|
|= B. .0+.00 .|
|. + 00S.+ .|
|o .+.*|
|o .00. .|
|.. .. .|
|.o. .|
+---[SHA256]-----+
Registered public key. ✅
🔥 Created access with ID r_be-gent1-bi-vm1_01ks5dq2nyf63tjeb44zmfjhf1
% Total % Received % Xferd Average Speed Time Time Time Current
Dload Upload Total Spent Left Speed
100 4518 100 4518 0 0 10378 0 --:--:-- --:--:-- --:--:-- 10386
🔥 Created sfcc with ID r_be-gent1-bi-vm1_01ks5dr4nhejgbhmzjm51et1sv
```

Initialize the CCBP: Notebook

```
: ! echo "export PATH=$PATH" >> $HOME/.bashrc
! echo "export BASE_DIR=$BASE_DIR" >> $HOME/.bashrc
! echo "export KUBECONFIG=$BASE_DIR/SlicesFile/kubeconfigs">>$HOME/.bashrc
```

```
: !$BASE_DIR/ccbp-cli/scripts_notebook/InitSFCC $SLICES_PROJECT_NAME
```

```
: !slices bi list --experiment ccbp-base
```

Resources in experiment exp_expauth.ilabt.imec.be_01kqyfx4w0fwksk9042j3t1ykm

ID	Name	Status	Created At	Expires At	Public IPv4	Private IPv4
r_be-gent1-bi-vm1_01krg46625fbc914g40...	access	up	2026-05-13 09:35 CEST	2026-05-13 19:50 CEST	193.191.169.35	
r_be-gent1-bi-vm1_01krg477b8fjnak7v9x...	sfcc	up	2026-05-13 09:36 CEST	2026-05-13 19:50 CEST		10.10.222.66

```
: !slices bi extend -d 10h access --experiment ccbp-base
!slices bi extend -d 10h sfcc --experiment ccbp-base
```

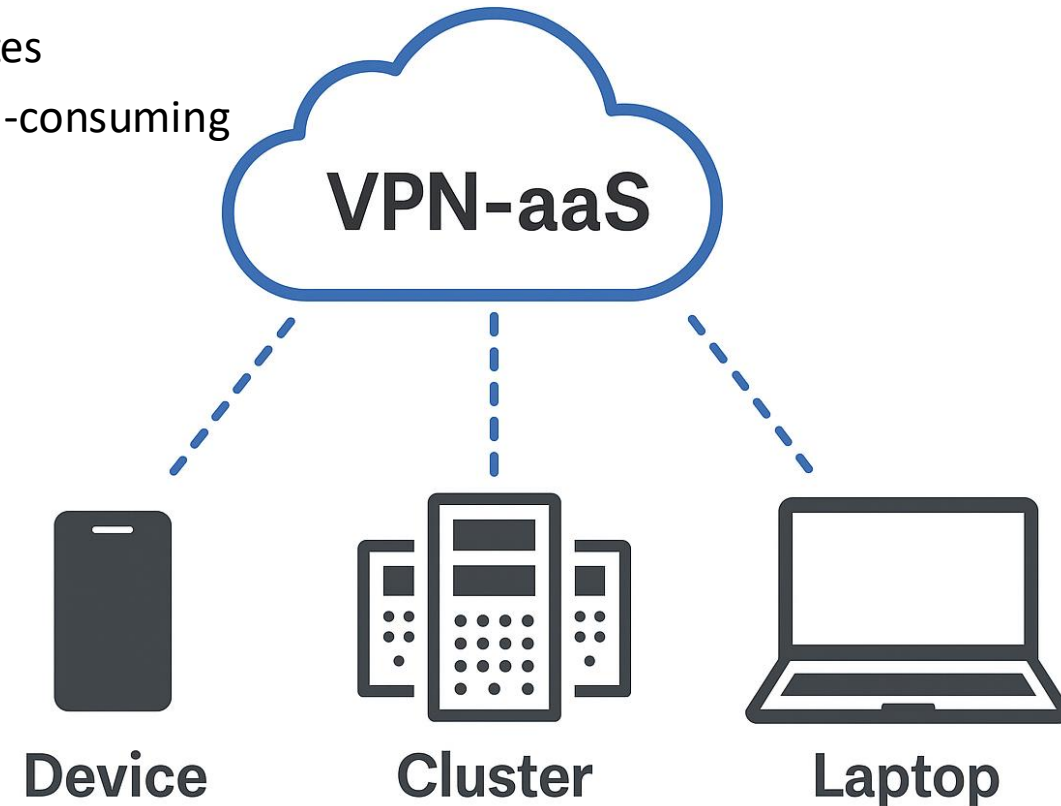
VPN-aaS

Motivation

- Cloud Continuum experiments span distributed, heterogeneous sites
- Network configuration and secure access can be complex and time-consuming
- Researchers need **transparent** and unified connectivity

CCBP VPN-aaS:

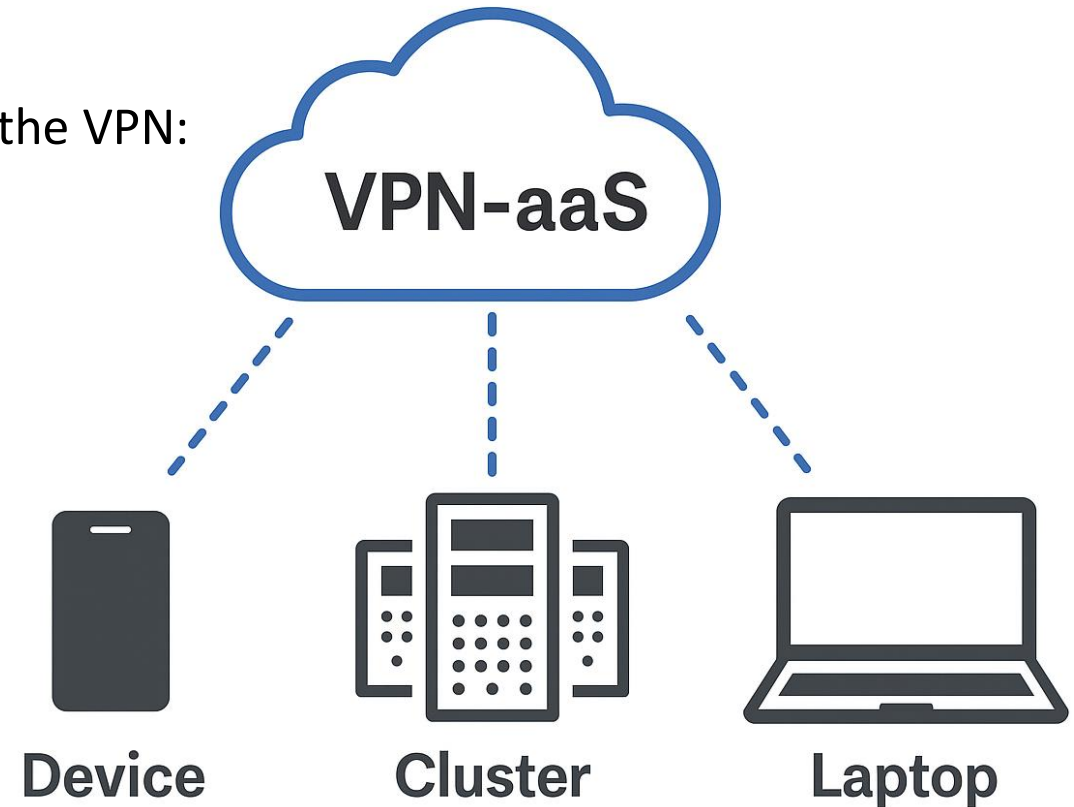
- Created by default at CCBP startup
- Reduces the requests for Public IPs on the BI
- Support future port forwarding
- Facilitate **remote development** on CCBP and BI resources
- Enable to “**connect**” remote resources, infrastructures and devices
- Implemented through Wireguard:
 - Open-Source
 - Supports different topology
 - Multi-platform



The VPN

The Access VM at startups automatically:

1. Creates 15 ready to use configurations for connecting to the VPN:
 1. Dev environments
 2. Single IoT, Device, Service
 3. Remote clusters
2. Starts the VPN server



VPN-aaS: local CCBPCLI

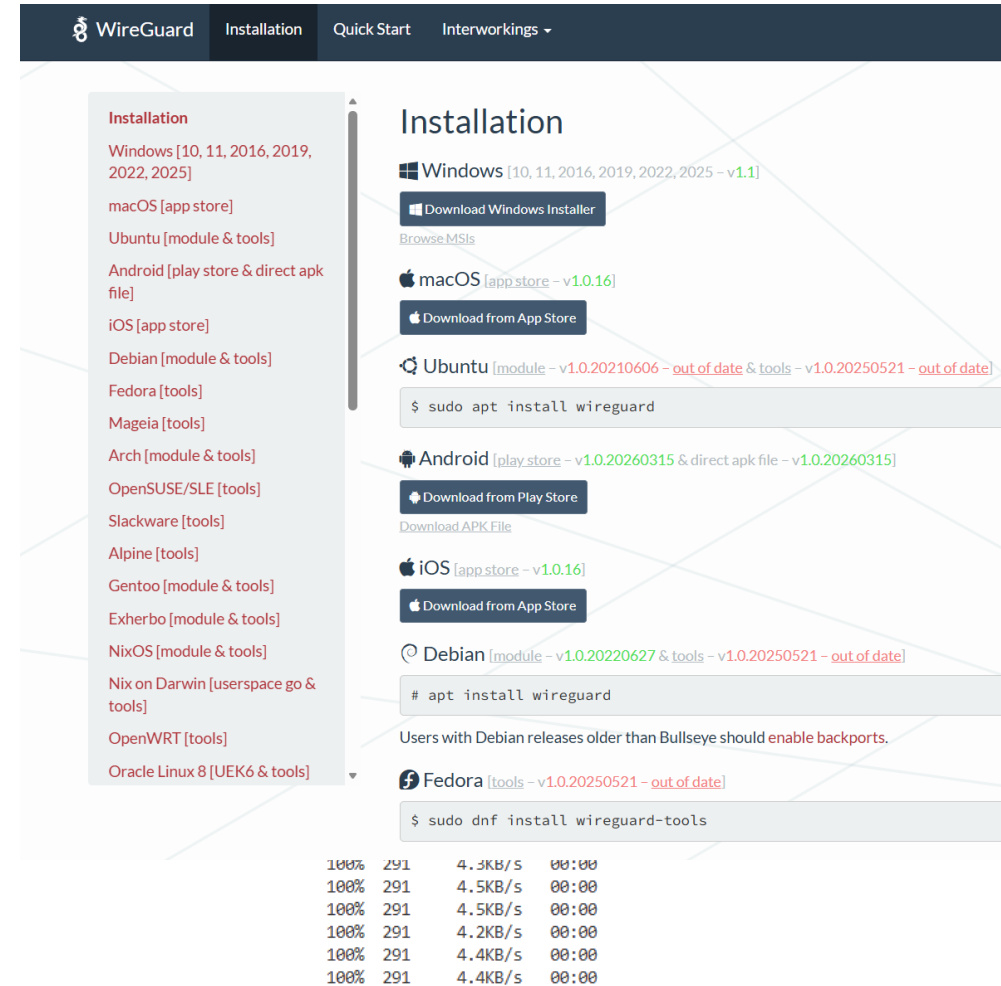
To automatically download VPN configurations:

```
$ ccbpcli getVpnConfigs <project_name
```

Then you can connect to the VPN through one of the available clients:

- Windows
- Unix systems install=> wg-quick
- Android Wireguard on play store

```
root@5355a454bac0:/# getVpnConfigs ccbpcli
The current project was set to 'ccbpcli', in which you are a lead and which expires on 2027-04-28 00:00 UTC.
Warning: Permanently added '193.191.169.51' (ED25519) to the list of known hosts.
client10.conf
client11.conf
client12.conf
client13.conf
client14.conf
client15.conf
client2.conf
client3.conf
client4.conf
client5.conf
client6.conf
client7.conf
client8.conf
client9.conf
```



The screenshot shows the WireGuard website's installation page. The left sidebar lists various operating systems and their installation methods. The main content area provides detailed instructions for Windows, macOS, Ubuntu, Android, iOS, Debian, and Fedora. At the bottom, there is a table showing download progress for several files.

Progress	File Name	Size	Speed	Time
100%	291	4.3KB/s	00:00	
100%	291	4.5KB/s	00:00	
100%	291	4.5KB/s	00:00	
100%	291	4.2KB/s	00:00	
100%	291	4.4KB/s	00:00	
100%	291	4.4KB/s	00:00	

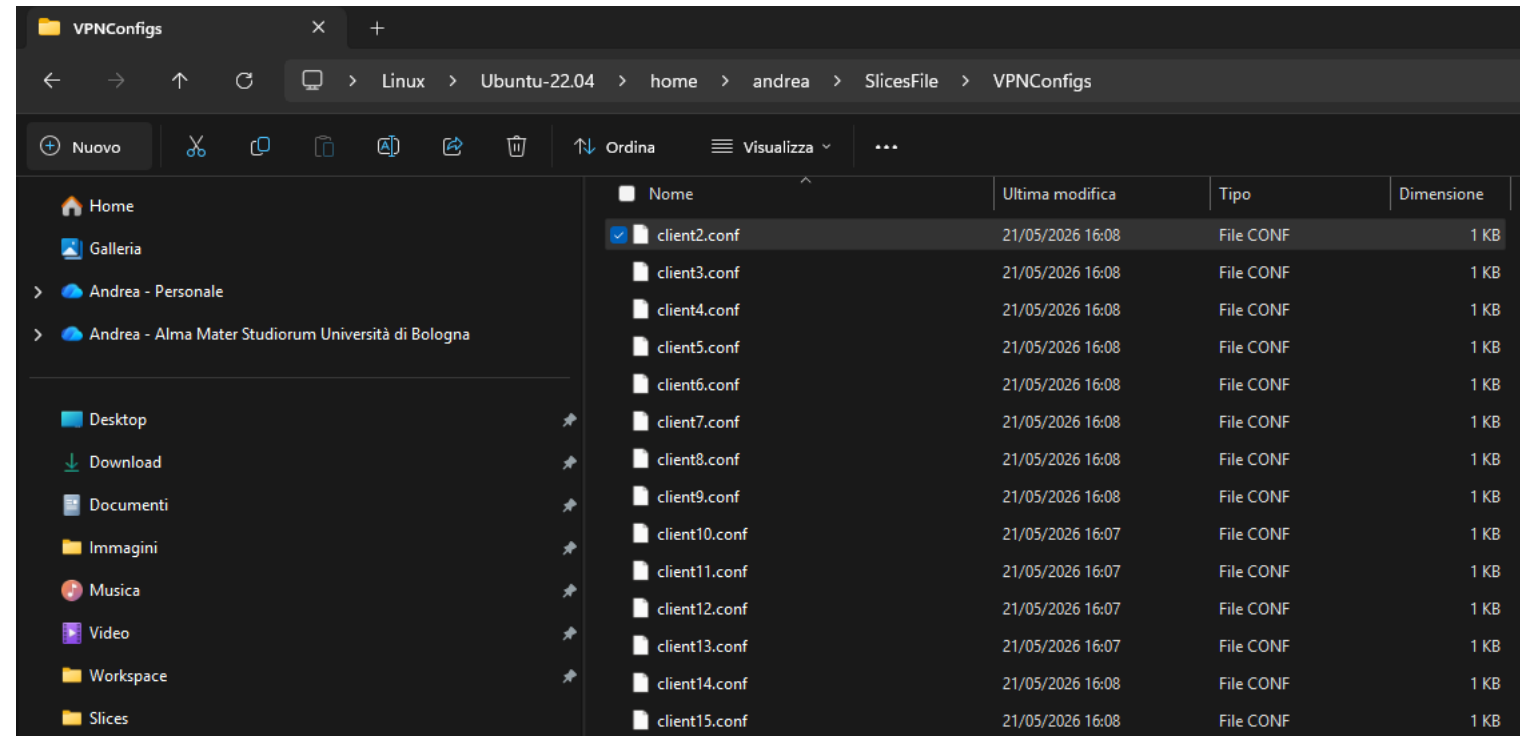
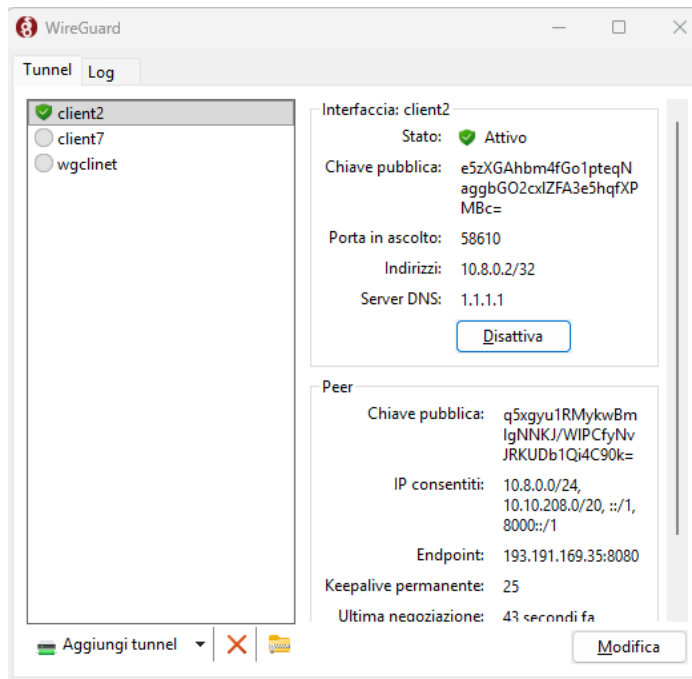


Connect to the VPN: Windows+WSL

If you are running the CCBPCLI under WSL you can:

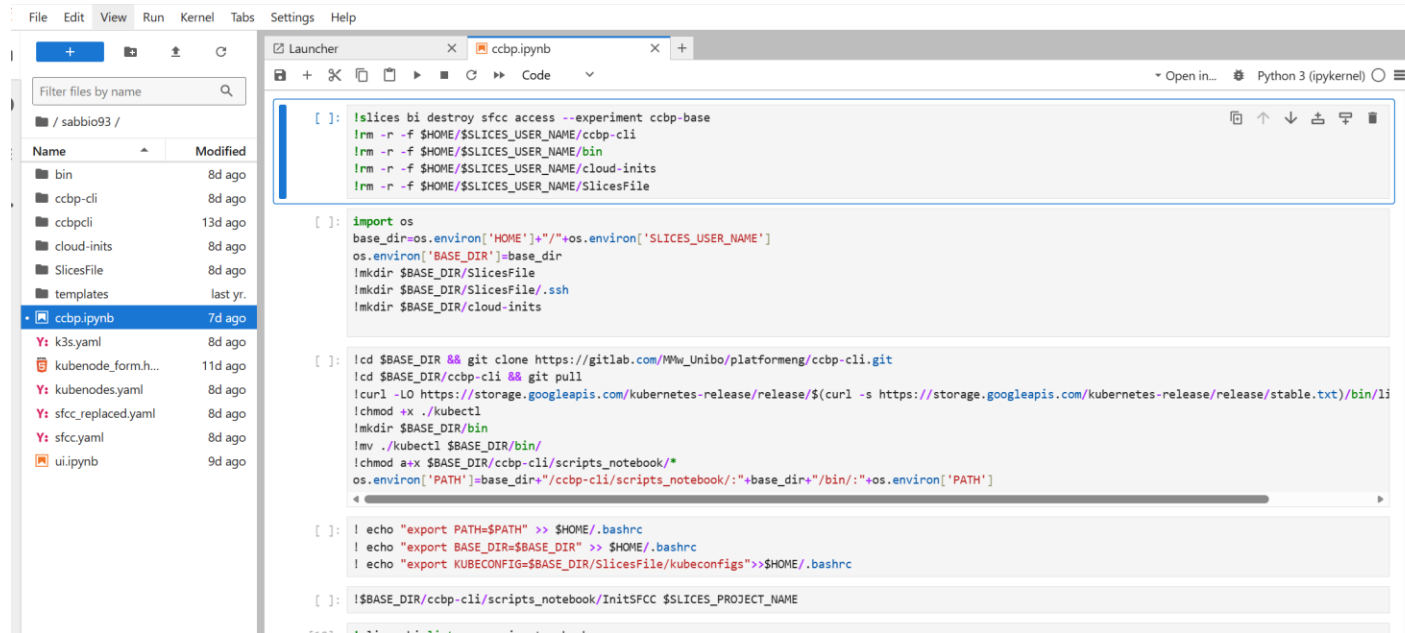
- Install the wireguard client on Windows
- Navigate to SlicesFile directory inside WSL to get the certificate

Both windows and WSL are connected through the VPN



VPN-aaS: Notebook CCBPCLI

If you are working on the python notebook and want to connect your pc or another device, you can download a VPN configuration from the folder SlicesFile/VPNConfigs trough the left panel of the notebook.



The screenshot shows a Jupyter Notebook interface with a file browser on the left and a code editor on the right. The file browser shows a directory structure with files like bin, ccbp-cli, ccbpcli, cloud-inits, SlicesFile, templates, and ccbp.ipynb. The code editor shows a series of commands to set up the environment, including cloning a git repository, installing dependencies, and setting environment variables.

```
[ ]: !lslices bi destroy sfcc access --experiment ccbp-base
!rm -r -f $HOME/$SLICES_USER_NAME/ccbp-cli
!rm -r -f $HOME/$SLICES_USER_NAME/bin
!rm -r -f $HOME/$SLICES_USER_NAME/cloud-inits
!rm -r -f $HOME/$SLICES_USER_NAME/SlicesFile

[ ]: import os
base_dir=os.environ['HOME']+"/"+os.environ['SLICES_USER_NAME']
os.environ['BASE_DIR']=base_dir
mkdir $BASE_DIR/SlicesFile
mkdir $BASE_DIR/SlicesFile/.ssh
mkdir $BASE_DIR/cloud-inits

[ ]: !cd $BASE_DIR && git clone https://gitlab.com/MWw_Unibo/platformeng/ccbp-cli.git
!cd $BASE_DIR/ccbp-cli && git pull
!curl -LO https://storage.googleapis.com/kubernetes-release/release/$(curl -s https://storage.googleapis.com/kubernetes-release/release/stable.txt)/bin/linux-amd64.tar.gz
!tar xzf linux-amd64.tar.gz
!mkdir $BASE_DIR/bin
!mv ./kubect1 $BASE_DIR/bin/
!chmod a+x $BASE_DIR/ccbp-cli/scripts_notebook/*
os.environ['PATH']=base_dir+"/ccbp-cli/scripts_notebook/"+base_dir+"/bin:"+os.environ['PATH']

[ ]: !echo "export PATH=$PATH" >> $HOME/.bashrc
!echo "export BASE_DIR=$BASE_DIR" >> $HOME/.bashrc
!echo "export KUBECONFIG=$BASE_DIR/SlicesFile/kubeconfigs">>$HOME/.bashrc

[ ]: !$BASE_DIR/ccbp-cli/scripts_notebook/InitSFCC $SLICES_PROJECT_NAME
```

/ ... / SlicesFile / VPNConfigs /	
Name	Modified
client10.conf	8d ago
client11.conf	8d ago
client12.conf	8d ago
client13.conf	8d ago
client14.conf	8d ago
client15.conf	8d ago
client2.conf	8d ago
client3.conf	8d ago
client4.conf	8d ago
client5.conf	8d ago
client6.conf	8d ago
client7.conf	8d ago
client8.conf	8d ago
client9.conf	8d ago



CCBPCLI: getVPNConfigUtils

getVPNConfigUtils is an utility that enables to download the Kubernetes configuration needed to interact with the cluster through the *kubectI* utils.

When invoked it will automatically download the requested configuration in the folders

- SlicesFile/kubeconfigs in Notebook and in mounted directories in local CCBPLI environment
- /SlicesFile/kubeconfigs inside the CCBPCLI container

Inside the ccbpcli

```
$ ccbpcli getVpnConfigUtils <project_name> <experiment_name> <vm_name>
```

e.g.

```
$ ccbpcli getVpnConfigUtils ccbp ccbp-base sfcc
```



The SFCC

Configurations and **services** of the SFCC are managed as Kubernetes objects and accessible through the kubectl tool or compliant third party tools.

This enables native integration with Dev/Ops tools such as:

- IaC (Helm, Terraform, Ansible...)
- Kubernetes Uis (Rancher,
- Pipeline tools (Argo CD, Github, Gitlab...)

```
root@d070e84dca17:/# getKubeconfigUtils ccbpcli ccbp-base sfcc
The current project was set to 'ccbpcli', in which you are a lead and which expires on 2027-04-28 00:00 UTC.
Warning: Permanently added '10.10.220.234' (ED25519) to the list of known hosts.
root@d070e84dca17:/# ls /SlicesFile/kubeconfigs/sfcc.yaml
/SlicesFile/kubeconfigs/sfcc.yaml
```

```
andrea@SURFACESVENTURA:~/SlicesFile/kubeconfigs$ kubectl get pods --kubeconfig=./sfcc.yaml
```

NAME	READY	STATUS	RESTARTS	AGE
couchdb-couchdb-0	1/1	Running	0	15d
couchdb-couchdb-1	1/1	Running	0	15d
couchdb-couchdb-2	1/1	Running	0	15d
example-kc-0	1/1	Running	0	15d
keycloak-operator-65f4b6cbc8-2kjdr	1/1	Running	0	15d
loki-0	2/2	Running	0	15d
loki-canary-66bfh	1/1	Running	0	15d
loki-chunks-cache-0	0/2	Pending	0	15d
loki-gateway-8569888448-swgjg	1/1	Running	0	15d
loki-minio-0	1/1	Running	0	15d
loki-results-cache-0	2/2	Running	0	15d
my-coredns-78c76db4b5-xv7xf	1/1	Running	0	15d
my-grafana-7df77989cd-52zh8	1/1	Running	0	15d
otel-collector-opentelemetry-collector-5b8b68dfb8-b47bd	1/1	Running	0	15d
postgresql-db-0	1/1	Running	0	15d
prometheus-6d67d99754-v4wdh	1/1	Running	0	15d
pushgateway-654c4b644d-wctb8	1/1	Running	0	15d
slicesbi-operator-75b59795c9-pv712	1/1	Running	0	15d



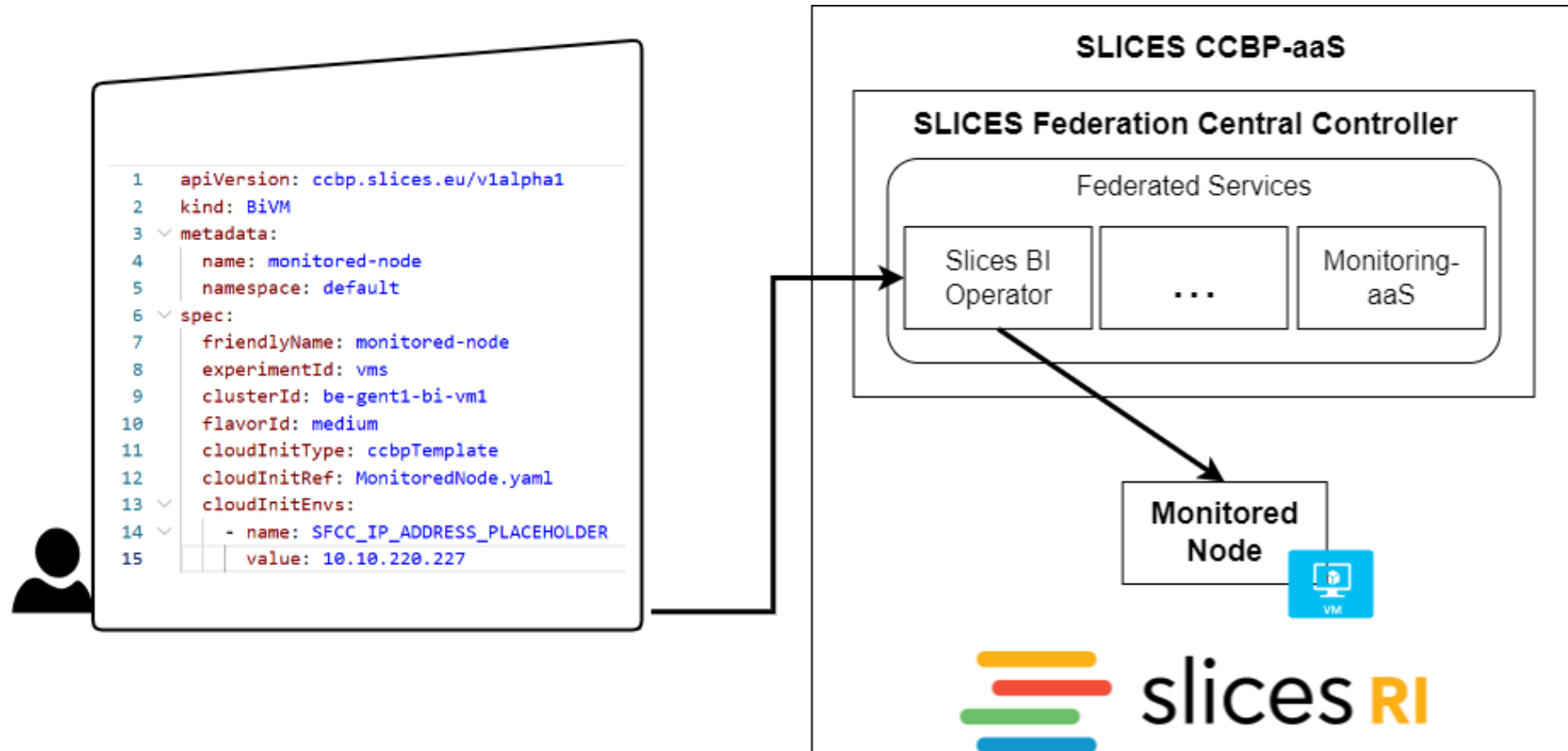
The Slices BI Operator & Presets

A Specialized Controller enabling provisioning of resources on the Slices BI from the CCBP

- Wraps the SLICES-BI APIs through Kubernetes APIs
- Strong validation and authorization checks
- Namespace-based assignment of resources
- Enables templated Cloudinit configurations
- Support URL references to Cloudinits
- Automatically retrieve secrets and configuration from nodes and store as Kubernetes Secrets, ConfigMaps
- Instrument created VMs with DNS rules, and service references
- Support the creation of **Presets**:
“Frequent requested and already tested environments generic enough to support multiple class and types of experiments such as: Kubernetes, Knative, Openstack...”



The Slices BI Operator & Presets



Install the SlicesBI Operator: Local Development & Remote VM

For local development only ensure to be connected to the VPN and able to reach the sfcc node

```
$ccbpcli
```

```
$getVPNConfigUtils ccbp ccbp-base sfcc
```

From inside the ccbpcli we can use the InstallSlicesBIOperator utility to automatically deploy and setup the SlicesBI Operator

```
$InstallSlicesBIOperator <project_name> <experiment_name>
```

e.g.

```
$InstallSlicesBIOperator ccbp ccbp-base
```



Install the SlicesBI Operator: Local Development & Remote VM

The tool setups the Operator and configure:

- Credentials to interact with the slices bi
- Authorization to allow the R/W of Kubernetes objects
- Custom Resource Definitions to support native definition of Slices BI resources

```
root@d070e84dca17:/# InstallSlicesBIOperator ccbcli ccbp-base
The current project was set to 'ccbcli', in which you are a lead and which expires
apiVersion: v1
clusters:
- cluster:
  certificate-authority-data: LS0tLS1CRUdJTiBDRVJUSUZJQ0FURSB0tLS0tCk1JSUJlRENDQ
  qVXdXaGNOTXpZd05URTRNVE15TWpVdWpXakFqTVNFd0h3WURWUWFEREJock0zTXRjMlZ5ZG1WeUxXTmhR
  d1OHhIaUYzOW5RWjZyRNFTE1GTDlZTDc5bUttUfVb0JJJaXNlZTNvME13UURBT0JnTlZIUTHCQWY4RQp
  VAXeTJXNUlRVHppVFdZcm9RZrZWnkM5R1c3a29BeXAKV3NyWkZor2sxSctSQWlFQXBEM01jNGJJNXJsRU
  server: https://0.0.0.0:6443
  name: default
contexts:
- context:
  cluster: default
  user: default
  name: default
current-context: default
kind: Config
users:
- name: default
  user:
    client-certificate-data: LS0tLS1CRUdJTiBDRVJUSUZJQ0FURSB0tLS0tCk1JSUJrRENDQVRl
    lNVEV6TWpJMU1Gb1hEVEkzTURVeQpNVEV6TWpJMU1Gb3dNREVYTUJVR0ExVUVDaE1PYzNsemRHVnRPbTF
    FBTR2xFd3pvR2JyNWQxMHhpcmdReFQvT2dsU0lXc29yYwlpazgybG9pWVNXRXc0eStNdDY5RDlLbgpxYX
    V0NDcXl0ekFLQmdncWhrak9QUVFEQWd0SEFEQkUKQWlBdm01WHNawi820ExDKy9VcGlkQW1Id2s5Zm1QZi
    JTiBDRVJUSUZJQ0FURSB0tLS0tCk1JSUJkekeNDQViyZ0F3SUJBZ0lCQURBS0JnZ3Foa2pPUFFRREFqQWpN
    h3WURWUWFEREJock0zTXRZMnhwWlclMExXTmhRREUzTnprek56TXp0ekF3V1RBVEJnY3Foa2pPClBRSUJ
    jJ5WmVncDlVME13UURBT0JnTlZIUTHCQWY4RQpCQU1DQXFRd0R3WURWUjBUQVFIL0JBVXdBd0VCL3pBZE
    Q0I3NHl3am9mUzJIQWlBL2t0T2VRQU9qRDRRUmpxVkhETk0rampZHZSSSTYwS1Ivd3lTMUtFdWFMZz09C
    client-key-data: LS0tLS1CRUdJTiBFQyBQUkIwQVRFIEtFWSB0tLS0tCk1IY0NBUEVFSUhVeERN
    NYVEldUJERlA4NkNWSWhheQppdG4yS1R6YVdpSmhLb1REakw0eTnyMFAwcWwczT4cFJ3PT0KLS0tLS1
    customresourcedefinition.apiextensions.k8s.io/bivms.ccbp.slices.eu created
    customresourcedefinition.apiextensions.k8s.io/kubenodes.ccbp.slices.eu created
    customresourcedefinition.apiextensions.k8s.io/filegetters.ccbp.slices.eu created
    customresourcedefinition.apiextensions.k8s.io/kubenodeas.ccbp.slices.eu created
```



Install The SlicesBI Operator: Notebook (1/2)

Trigger the configuration and deployment of the Operator from the notebook

```
!mkdir $BASE_DIR/SlicesFile/.slices/  
!$BASE_DIR/ccbp-cli/scripts_notebook/InstallSlicesBIOperator ccbp-base
```

```
customresourcedefinition.apiextensions.k8s.io/bivms.ccbp.slices.eu created  
customresourcedefinition.apiextensions.k8s.io/kubenodes.ccbp.slices.eu created  
customresourcedefinition.apiextensions.k8s.io/filegetters.ccbp.slices.eu created  
customresourcedefinition.apiextensions.k8s.io/kubenodeas.ccbp.slices.eu created  
serviceaccount/monitor-compute-acc created  
clusterrole.rbac.authorization.k8s.io/monitor-compute created  
clusterrole.rbac.authorization.k8s.io/monitor-kubenode created  
clusterrole.rbac.authorization.k8s.io/monitor-kubenodea created  
clusterrole.rbac.authorization.k8s.io/create-secret created  
clusterrole.rbac.authorization.k8s.io/create-configmap created  
clusterrole.rbac.authorization.k8s.io/create-filegetter created  
clusterrole.rbac.authorization.k8s.io/create-events created  
clusterrolebinding.rbac.authorization.k8s.io/monitor-compute created  
clusterrolebinding.rbac.authorization.k8s.io/monitor-kubenode created  
clusterrolebinding.rbac.authorization.k8s.io/monitor-kubenodea created  
clusterrolebinding.rbac.authorization.k8s.io/create-secret created  
clusterrolebinding.rbac.authorization.k8s.io/create-configmap created  
clusterrolebinding.rbac.authorization.k8s.io/create-filegetter created  
clusterrolebinding.rbac.authorization.k8s.io/create-events created  
secret/slices-auth created  
secret/slices-settings created  
Waiting CRDs to be available...  
Installing Slices BI Operator on the cluster...  
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current  
                                 Dload  Upload  Total   Spent    Left   Speed  
100 1405 100 1405    0     0  4030      0 --:--:-- --:--:-- --:--:-- 4037  
deployment.apps/slicesbi-operator created
```

Install The SlicesBI Operator: Notebook (2/2)

Authorize the Operator to create resource on the Slices BI

```
1) && echo "Pod: $podname" && kubectl exec -it $podname --kubeconfig=$BASE_DIR/SlicesFile/kubeconfigs/sfcc.yaml -- slices auth login
1) && echo "Pod: $podname" && kubectl exec -it $podname --kubeconfig=$BASE_DIR/SlicesFile/kubeconfigs/sfcc.yaml -- slices project use $SLICES_PROJECT_NAME
```

Pod: slicesbi-operator-5577bff757-71x97

Please go to

https://portal.slices-ri.eu/oauth/authorize_device?user_code=JGWR-XCJC and login
to continue.

∴ Waiting for authorizationon

Successfully logged in as 'sabbio93'

📁 Saved to /root/.slices/auth.json

Invalid project id '3010SLICES_PROJECT_NAME' ✖

Please check your configuration. You can set a valid project with `slices project`

use `<project name>`

command terminated with exit code 130

Pod: slicesbi-operator-5577bff757-71x97

The current project was set to 'ccbpcli', in which you are a lead and which
expires on 2027-04-28 00:00 UTC.



Deploy the first cluster

Connect to the SFCC

Launcher

ccbp.ipynb

jovyan@064c8d363ff2: ~/sak

Every 2.0s: slices bi list --experiment ccbpbase064c8d363ff2: Wed May 27 16:04:08 2026

Resources in experiment exp_expauth.ilabt.imec.be_01ksmvwv7sf17aa7mnzr3mq0b1

ID	Name	Status	Created At	Expires At	Public IPv4	Private IPv4
r_be-gent1-bi-vm1_01ksmvwv5web4r3b7hcnsnfjqm	master	up	2026-05-27 16:02 CEST	2026-05-27 19:02 CEST		10.10.218.148
r_be-gent1-bi-vm1_01ksmvzbx2f74rjy9d3mak43gx	worker1	booting	2026-05-27 16:03 CEST	2026-05-27 19:03 CEST		10.10.218.87

The SlicesBI Operator Kubenode

Kubernetes objects:

- BiVM
- FileGetter
- KubeNode

```
andrea@SURFACESVENTURA:~/SlicesFile/kubeconfigs$ kubectl describe crds kubenodes.ccbp.slices.eu
Name:          kubenodes.ccbp.slices.eu
Namespace:
Labels:        <none>
Annotations:   <none>
API Version:   apiextensions.k8s.io/v1
Kind:          CustomResourceDefinition
Metadata:
  Creation Timestamp:  2026-04-23T17:19:00Z
  Generation:         1
  Resource Version:    1539
  UID:                 35d005cf-e1a5-4d70-8400-c7d4158380aa
Spec:
  Conversion:
    Strategy:  None
  Group:      ccbp.slices.eu
  Names:
    Kind:      KubeNode
    List Kind: KubeNodeList
    Plural:    kubenodes
    Short Names:
      kn
    Singular:  kubnode
  Scope:      Namespaced
  Versions:
    Additional Printer Columns:
      Description: The user readable name of the ComputeResource
      Json Path:   .spec.friendlyName
      Name:        friendlyName
      Type:        string
      Description: The state of the ComputeResource in the backend
      Json Path:   .status.state
      Name:        state
      Type:        string
      Description: The creation time of the ComputeResource in the backend
      Json Path:   .status.createdAt
      Name:        createdAt
      Type:        date
      Description: The expiration time of the ComputeResource in the backend
      Format:      date-time
      Json Path:   .status.expiresAt
      Name:        expiresAt
      Type:        string
      Description: Time in seconds until expiration
      Json Path:   .status.expiresIn
      Name:        SecondsToExpire
      Type:        integer
      Description: The cluster ID of the ComputeResource
```

BiVM custom resource definition

```
apiVersion: ccbp.slices.eu/v1alpha1
kind: BiVM
metadata:
  name: server2
  namespace: default
spec:
  friendlyName: server02
  experimentId: cctest
  clusterId: be-gent1-bi-vm1
  flavorId: small
  cloudInitType: empty
```

```
kubectl apply -f vmemptyCloudInit.yaml --kubeconfig /SlicesFile/kubeconfigs/sfcc.yaml
```

```
root@22b8d20ba13f:/# slices bi list --experiment cctest
Resources in experiment exp_expauth.ilabt.imec.be_01kqygsw67f56bp0zsy9m1ccv3
```

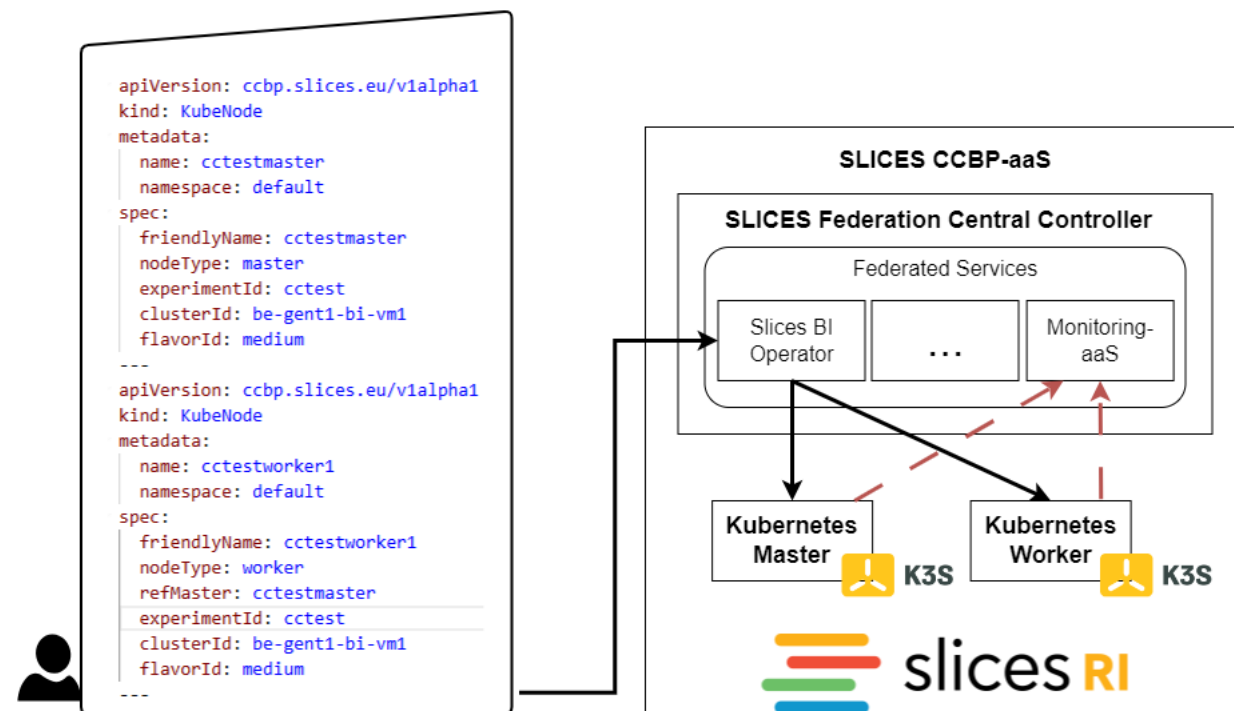
ID	Name	Status	Created At	Expires At	Public IPv4	Private IPv4
r_be-gent1-bi-vm1_01kt14wdk5e44r1sdyzy7zwg0h	server02	booting	2026-06-01 08:30 UTC	2026-06-01 09:30 UTC		10.10.208.172

The Kubernetes Operator

The Kubernetes Operator enables automatic and dynamic provisioning of Kubernetes clusters on the Slices BI infrastructures.

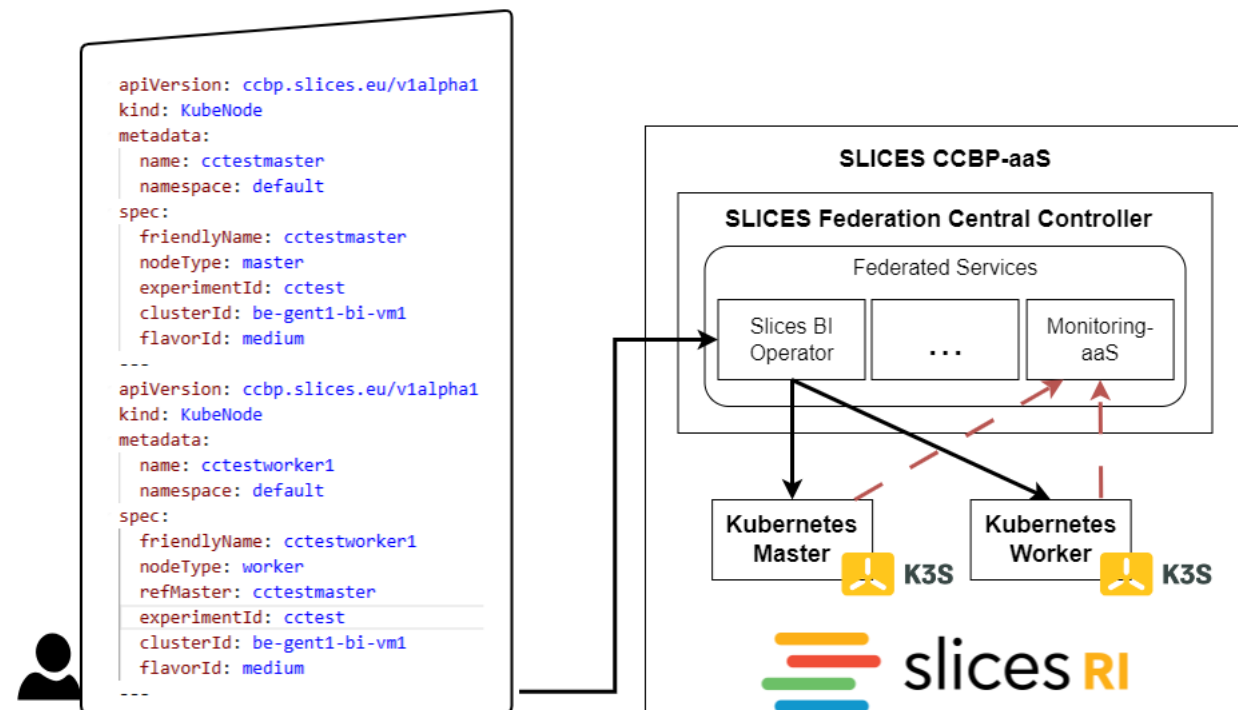
In particular it handles:

- The provisioning of the VMs
- Stores the secrets to join the cluster
- Stores of the Kubeconfig to access the cluster
- Checks and updates its status
- Configure the cluster to collect metrics and logs about:
 - Nodes
 - Every Single Pod instantiated
- Configure the endpoint for send metrics and logs to the Mon-aaS



The Kubernetes Operator

```
apiVersion: ccbp.slices.eu/v1alpha1
kind: KubeNode
metadata:
  name: servera
  namespace: default
spec:
  friendlyName: servera
  experimentId: ccbpbase
  clusterId: be-gent1-bi-vm1
  flavorId: medium
```



```
kubectl apply -f /experiments/kubenode.yaml -
-kubeconfig /SlicesFile/kubeconfigs/sfcc.yaml
```

```
root@22b8d20ba13f:/experiments# slices bi list --experiment ccbpbase
Resources in experiment exp_expauth.ilabt.imec.be_01kt1566ewfss92a67dnaqqth7
```

ID	Name	Status	Created At	Expires At	Public IPv4	Private IPv4
r_be-gent1-bi-vm1_01kt15677wf01bw26k3mb7ye2r	servera	up	2026-06-01 08:35 UTC	2026-06-01 11:35 UTC		10.10.213.6



SFCC: get secrets (1/2)

```
root@22b8d20ba13f:/experiments# getKubeconfigUtils ccbpcli ccbpbase servera
The current project was set to 'ccbpcli', in which you are a lead and which expires on 2027-04-28 00:00 UTC.
Warning: Permanently added '10.10.213.6' (ED25519) to the list of known hosts.
```

```
andrea@SURFACESVENTURA:~/workspace/slices/slices-bi-operator/test/testmanifest$ kubectl get secrets --kubeconfig /home/andrea/SlicesFile/kubeconfigs/sfcc.yaml
```

NAME	TYPE	DATA	AGE
couchdb-couchdb	Opaque	4	13m
example-kc-initial-admin	kubernetes.io/basic-auth	2	6m37s
example-tls-secret	kubernetes.io/tls	2	13m
keycloak-db-secret	Opaque	2	13m
kubeconfig-servera	Opaque	1	3m56s
loki-minio	Opaque	2	13m
my-grafana	Opaque	3	12m
node-token-servera	Opaque	1	3m56s
sh.helm.release.v1.couchdb.v1	helm.sh/release.v1	1	13m
sh.helm.release.v1.loki.v1	helm.sh/release.v1	1	13m
sh.helm.release.v1.my-grafana.v1	helm.sh/release.v1	1	12m
sh.helm.release.v1.otel-collector.v1	helm.sh/release.v1	1	12m
slices-auth	Opaque	1	11m
slices-settings	Opaque	1	11m

```
kubectl get secrets node-token-servera -o jsonpath='{.data.node-token}' --kubeconfig /home/andrea/SlicesFile/kubeconfigs/sfcc.yaml | base64 -d
```



SFCC: get secrets (2/2)

```
andrea@SURFACESVENTURA:~/workspace/slices/slices-bi-operator/test/testmanifest$ kubectl describe secrets kubeconfig-servera --kubeconfig /home/andrea/SlicesFile/kubeconfig/sfcc.yaml
```

```
Name:      kubeconfig-servera
Namespace:  default
Labels:     <none>
Annotations: <none>
```

```
Type: Opaque
```

```
Data
```

```
====
```

```
kubeconfig: 2943 bytes
```

-

```
kubectl get secrets kubeconfig-servera -o jsonpath='{.data.kubeconfig}' --kubeconfig /home/andrea/SlicesFile/kubeconfigs/sfcc.yaml | base64 -d
```

Kubernetes Operator: Multi-Cluster

```
apiVersion:
ccbp.slices.eu/v1alpha1
kind: KubeNode
metadata:
  name: master
  namespace: default
spec:
  friendlyName: master
  nodeType: master
  experimentId: ccbpbase
  clusterId: be-gent1-bi-vm1
  flavorId: medium
---
apiVersion:
ccbp.slices.eu/v1alpha1
kind: KubeNode
metadata:
  name: worker1
  namespace: default
spec:
  friendlyName: worker1
  nodeType: worker
  refMaster: master
  experimentId: ccbpbase
```

```
  clusterId: be-gent1-bi-vm1
  flavorId: medium
---
apiVersion:
ccbp.slices.eu/v1alpha1
kind: KubeNode
metadata:
  name: worker2
  namespace: default
spec:
  friendlyName: worker2
  nodeType: worker
  refMaster: master
  experimentId: ccbpbase
  clusterId: be-gent1-bi-vm1
  flavorId: medium
---
apiVersion:
ccbp.slices.eu/v1alpha1
kind: KubeNode
metadata:
  name: edge1
  namespace: default
spec:
```

```
  friendlyName: edge1
  nodeType: master
  experimentId: ccbpbase
  clusterId: be-gent1-bi-vm1
  flavorId: medium
  duration: 1h
---
apiVersion:
ccbp.slices.eu/v1alpha1
kind: KubeNode
metadata:
  name: edge2
  namespace: default
spec:
  friendlyName: edge2
  nodeType: master
  experimentId: ccbpbase
  clusterId: be-gent1-bi-vm1
  flavorId: small
  duration: 1h
```

Kubernetes Operator: Multi-Cluster

```
root@9aae743daf0d:/# kubectl apply -f /experiments/kubeCluster.yaml --kubeconfig /SlicesFile/kubeconfigs/sfcc.yaml
kubenode.ccbp.slices.eu/master created
kubenode.ccbp.slices.eu/worker1 created
kubenode.ccbp.slices.eu/worker2 created
kubenode.ccbp.slices.eu/edge1 created
kubenode.ccbp.slices.eu/edge2 created
```

Every 2.0s: kubectl get kubenodes --kubeconfig /SlicesFile/kubeconfigs/sfcc.yaml

NAME	FRIENDLYNAME	STATE	CREATEDAT	EXPIRESAT	SECONDDSTOEXPIRE	KUBECLUSTER	PRIVATEIPV4
edge1	edge1	up	2m35s	2026-05-26T11:07:00Z	10604		10.10.221.115
edge2	edge2	up	75s	2026-05-26T11:09:00Z	10723		10.10.217.87
master	master	up	6m53s	2026-05-26T11:03:00Z	10362		10.10.219.147
worker1	worker1	up	3m40s	2026-05-26T11:06:00Z	10541	master	10.10.222.24
worker2	worker2	up	3m7s	2026-05-26T11:07:00Z	10605	master	10.10.218.91

Kubernetes Operator: Multi-Cluster get secrets

```
root@9aae743daf0d:/# kubectl get secrets --kubeconfig /SlicesFile/kubeconfigs/sfcc.yaml
```

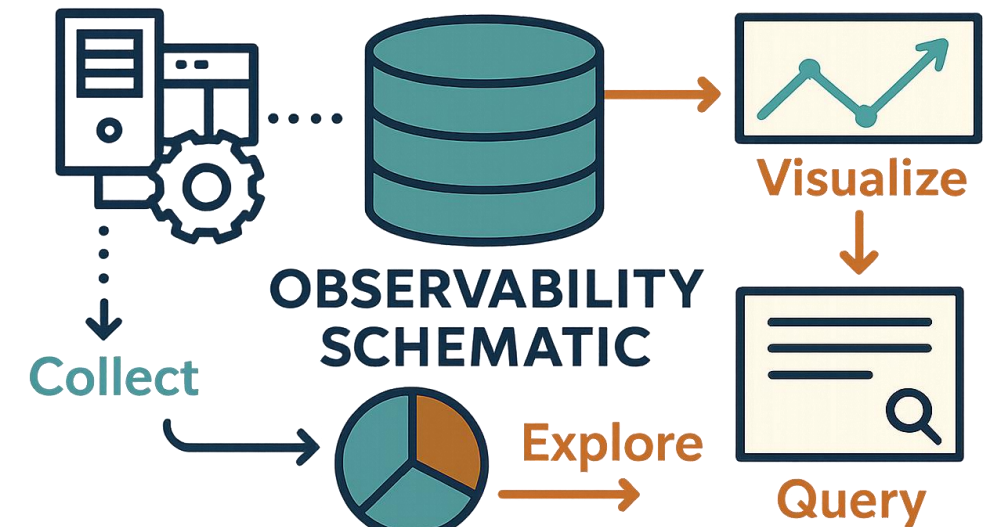
NAME	TYPE	DATA	AGE
couchdb-couchdb	Opaque	4	14m
example-kc-initial-admin	kubernetes.io/basic-auth	2	13m
example-tls-secret	kubernetes.io/tls	2	14m
keycloak-db-secret	Opaque	2	14m
kubeconfig-edge1	Opaque	1	2m51s
kubeconfig-edge2	Opaque	1	106s
kubeconfig-master	Opaque	1	5m17s
loki-minio	Opaque	2	14m
my-grafana	Opaque	3	13m
node-token-edge1	Opaque	1	2m51s
node-token-edge2	Opaque	1	106s
node-token-master	Opaque	1	5m17s



Monitoring-aaS & Auto instrumentation

Monitoring-as-a-Service (MaaS) provides centralized, real-time visibility and automated tracking of system performance and resources across distributed environments.

- Log and Metrics as signals (Release 2), Traces (planned for Release 2.5)
- **Open Telemetry standard** Compliant Stack
- **Auto-Instrumentation** of BI resources through automation
- Signals:
 - Collect
 - Store
 - Explore, Visualize and Query
- Infrastructure and Application-level Observability to support:
 - Debugging
 - Experiment results
 - Continuum infrastructure management



Monitoring-aaS Implementation

Prometheus

Scalable time-series monitoring and alerting for infrastructure and services

Loki

Efficient, label-based log aggregation tightly integrated with Grafana

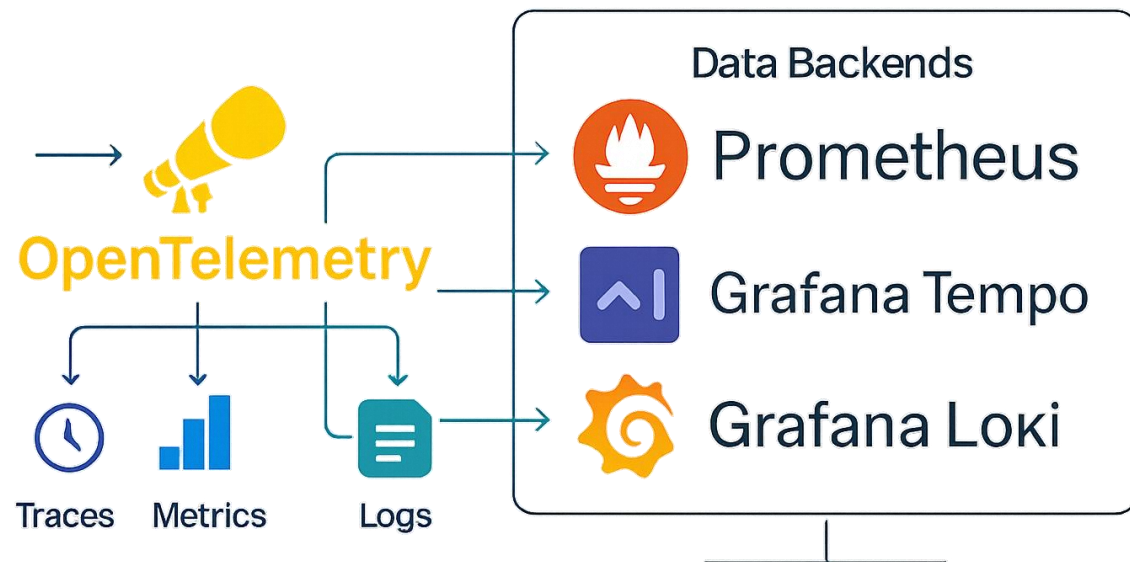
OpenTelemetry Collector

Vendor-neutral telemetry pipeline for collecting, processing, and exporting metrics, logs, and traces

Grafana

Unified visualization platform for real-time dashboards and analytics

- Gather signals from VMs, Clusters, devices
- Accessible from within the VPN and the BI infrastructure
- Supported SDKs, and REST API for query
- Enables the integration with the Slices BI S3



Monitoring-aaS Endpoints

```
andrea@SURFACESVENTURA:~/workspace/slices/slices-bi-operator/test/testmanifest$ kubectl get svc --kubeconfig /home/andrea/SlicesFile/kubeconfigs/sfcc.yaml
```

NAME	TYPE	CLUSTER-IP	EXTERNAL-IP	PORT(S)	AGE
couchdb-couchdb	ClusterIP	None	<none>	5984/TCP	39m
couchdb-svc-couchdb	NodePort	10.43.113.205	<none>	5984:376/TCP	39m
example-kc-discovery	ClusterIP	None	<none>	7800/TCP	32m
example-kc-service	ClusterIP	10.43.15.38	<none>	8443/TCP,9000/TCP	32m
key-serv	NodePort	10.43.95.249	<none>	9000:156/TCP,8443:325/TCP	39m
keycloak-operator	ClusterIP	10.43.36.168	<none>	80/TCP	39m
kubernetes	ClusterIP	10.43.0.1	<none>	443/TCP	40m
loki	ClusterIP	10.43.52.99	<none>	3100/TCP,9095/TCP	39m
loki-canary	ClusterIP	10.43.137.246	<none>	3500/TCP	39m
loki-chunks-cache	ClusterIP	None	<none>	11211/TCP,9150/TCP	39m
loki-gateway	ClusterIP	10.43.126.155	<none>	80/TCP	39m
loki-headless	ClusterIP	None	<none>	3100/TCP	39m
loki-memberlist	ClusterIP	None	<none>	7946/TCP	39m
loki-minio	ClusterIP	10.43.49.178	<none>	9000/TCP	39m
loki-minio-console	ClusterIP	10.43.217.21	<none>	9001/TCP	39m
loki-minio-svc	ClusterIP	None	<none>	9000/TCP	39m
loki-results-cache	ClusterIP	None	<none>	11211/TCP,9150/TCP	39m
my-coredns	NodePort	10.43.20.122	<none>	53:53/UDP,53:53/TCP	40m
my-grafana	ClusterIP	10.43.34.72	<none>	80/TCP	38m
otel-collector-opentelemetry-collector	ClusterIP	10.43.36.35	<none>	6831/UDP,14250/TCP,14268/TCP,8889/TCP,4317/TCP,4318/TCP,9411/TCP	38m
postgres-db	ClusterIP	10.43.111.160	<none>	5432/TCP	39m
prometheus	NodePort	10.43.135.106	<none>	9090:390/TCP	39m



How to access Monitoring-aaS

```
andrea@SURFACESVENTURA:~/workspace/slices/slices-bi-operator/test/testmanifest$ kubectl get secrets --kubeconfig /home/andrea/SlicesFile/kubeconfigs/sfcc.yaml
```

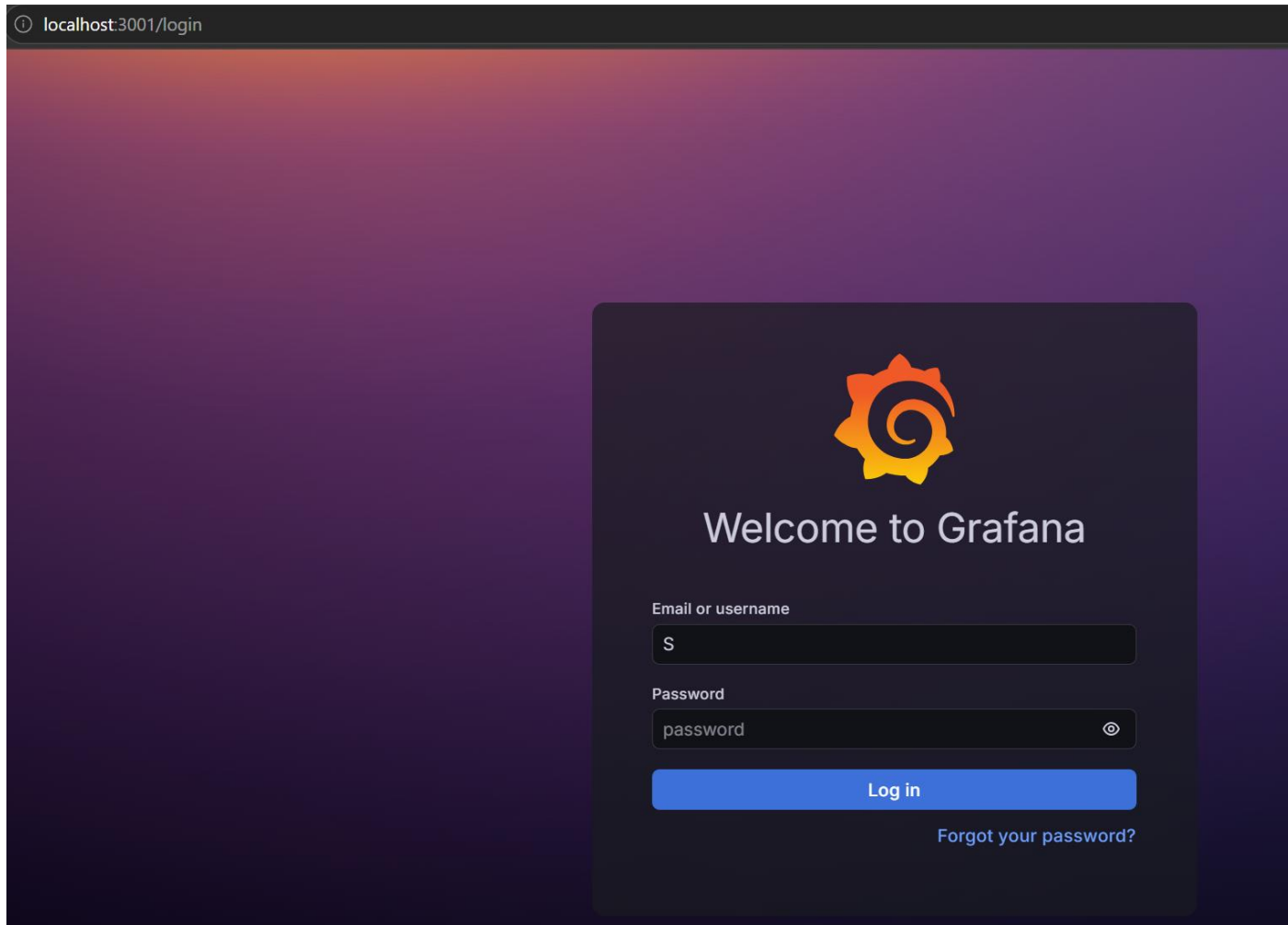
NAME	TYPE	DATA	AGE
couchdb-couchdb	Opaque	4	38m
example-kc-initial-admin	kubernetes.io/basic-auth	2	31m
example-tls-secret	kubernetes.io/tls	2	38m
keycloak-db-secret	Opaque	2	38m
kubeconfig-servera	Opaque	1	28m
loki-minio	Opaque	2	37m
my-grafana	Opaque	3	37m
node-token-servera	Opaque	1	28m
sh.helm.release.v1.couchdb.v1	helm.sh/release.v1	1	38m
sh.helm.release.v1.loki.v1	helm.sh/release.v1	1	37m
sh.helm.release.v1.my-grafana.v1	helm.sh/release.v1	1	37m
sh.helm.release.v1.otel-collector.v1	helm.sh/release.v1	1	37m
slices-auth	Opaque	1	36m
slices-settings	Opaque	1	36m

```
kubectl get secrets my-grafana -o jsonpath={.data.admin-password} --kubeconfig SlicesFile/kubeconfigs/sfcc.yaml | base64 -d
```

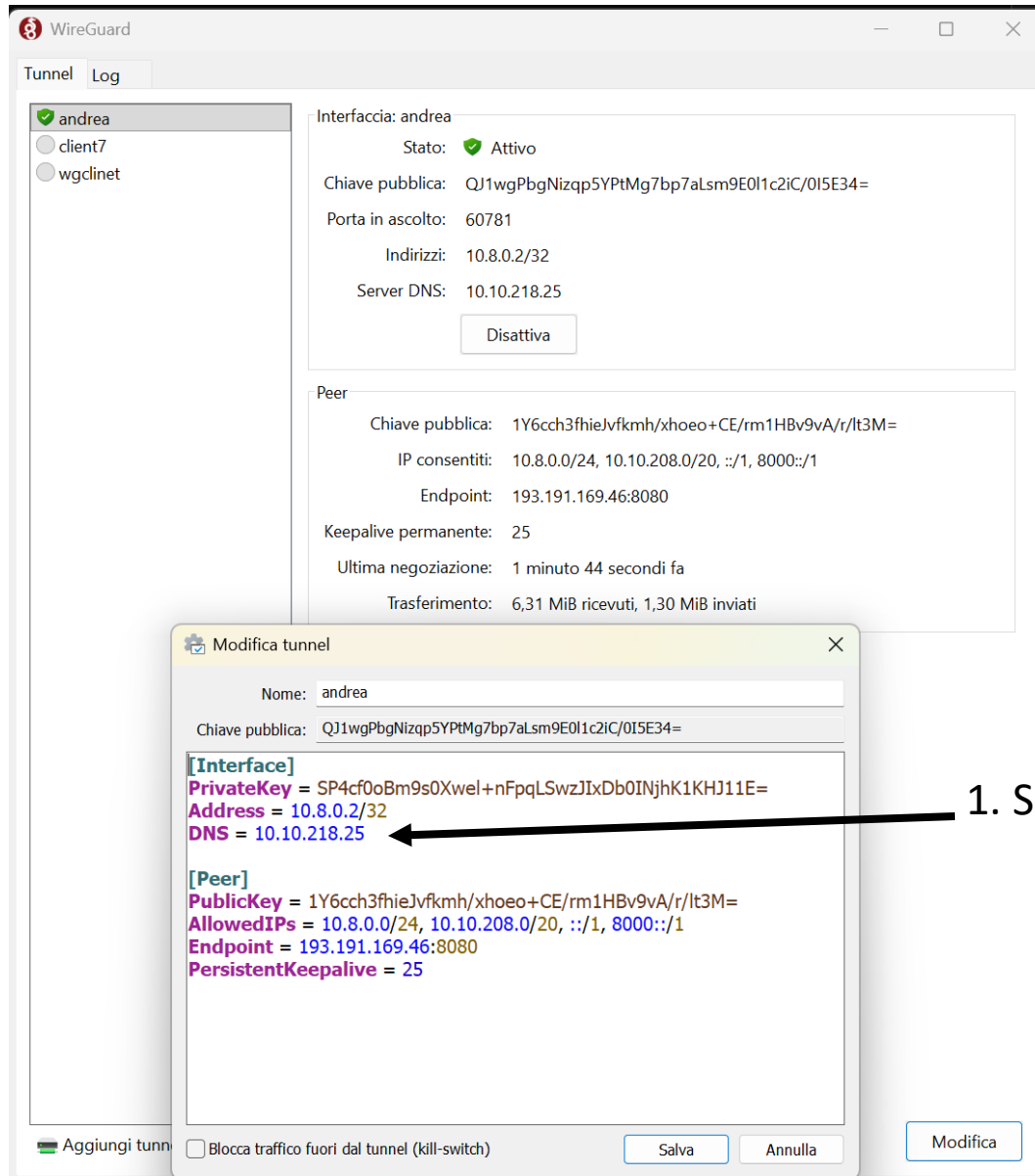


Accessing Monitoring-aaS: Kubeproxy and DNS-aaS (1/2)

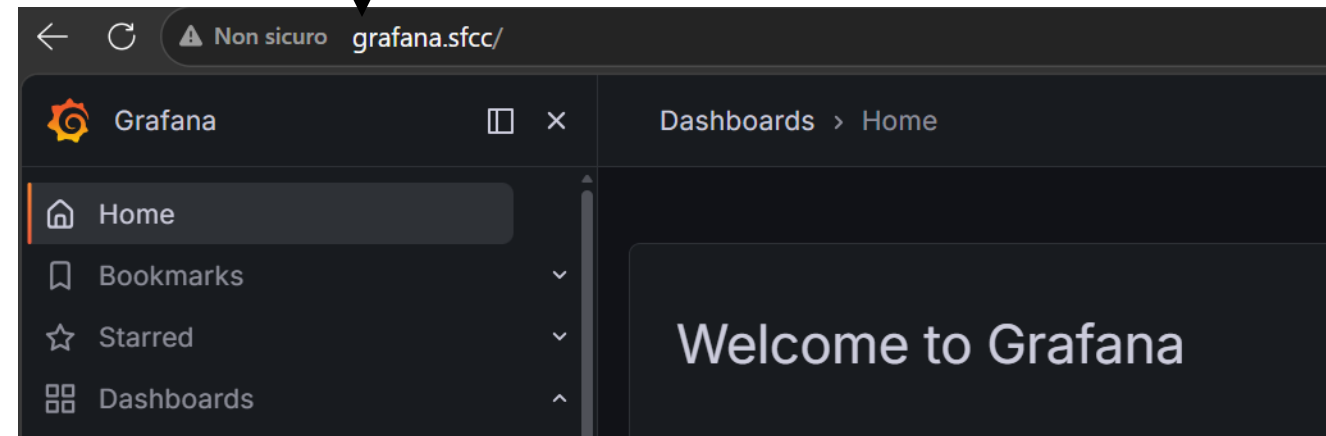
```
kubectl port-forward svc/my-grafana 3001:80 --kubeconfig SlicesFile/kubeconfigs/sfcc.yaml
```



Accessing Monitoring-aaS: Kubeproxy and DNS-aaS (1/2)

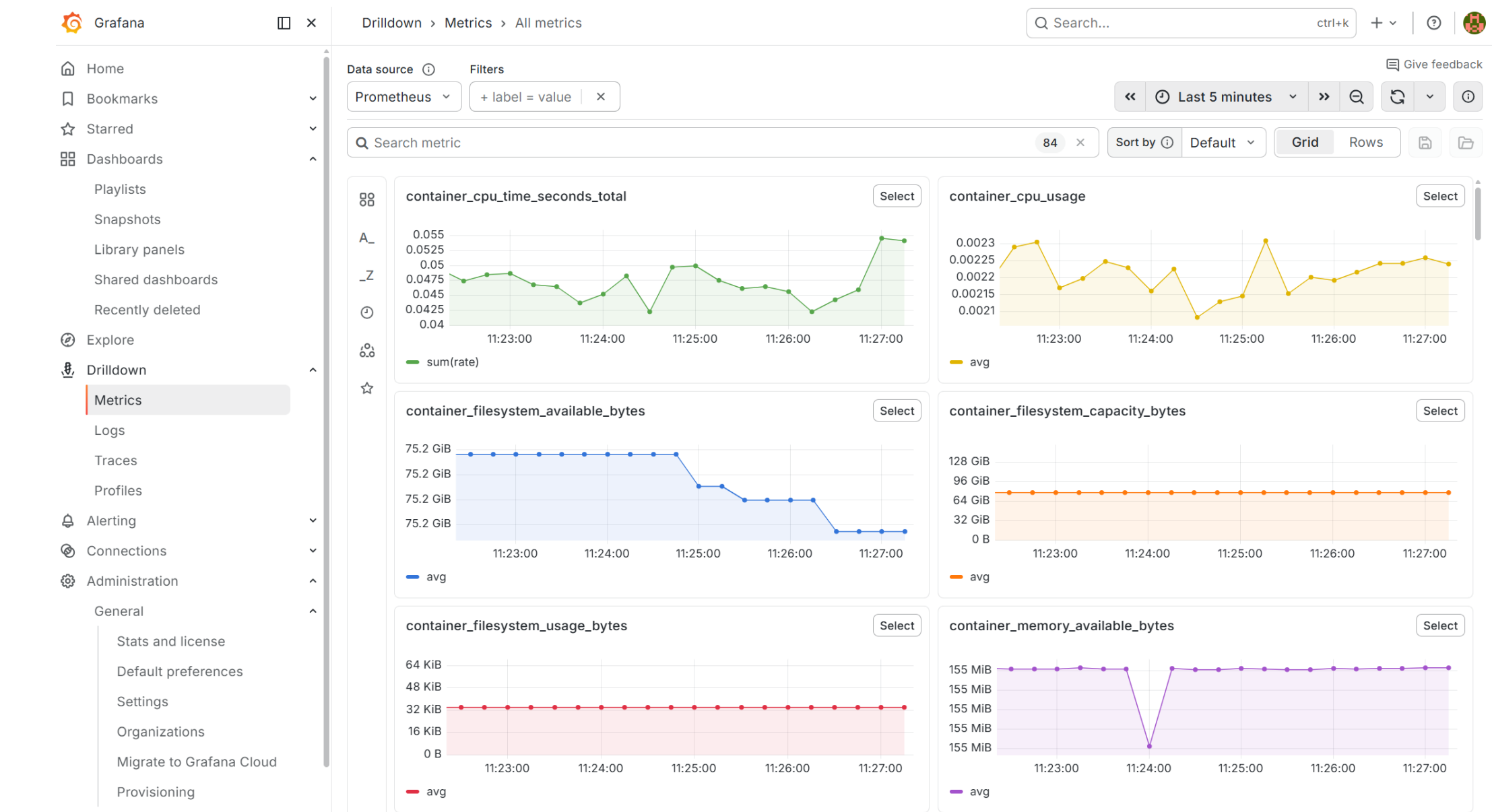


1. DNS record



1. SFCC IP

Monitoring-aaS: Grafana Drilldown metrics



66



Monitoring-aaS: Grafana APIs and Scripts/Notebook Integration

Grafana exposes a rich set of APIs that enable powerful queries on monitored sources

Steps to query Grafana:

1. Get an access token
2. Locate the datasource id
3. Query data

```
# ===== STEP 3: create token =====
token_url = f"{GRAFANA_URL}/api/serviceaccounts/{sa_id}/tokens"

token_payload = {"name": "notebook-token"}

resp = requests.post(token_url, headers=headers, auth=auth, data=json.dumps(token_payload))

if resp.status_code not in [200, 201]:
    print("❌ Token creation failed:", resp.text)
    raise SystemExit

token = resp.json().get("key")

print("\n🎉 BEARER TOKEN:")
print(token)
```

Monitoring-aaS: Grafana APIs and Notebook Integration

Grafana exposes a rich set of APIs that enable powerful queries on monitored sources

Steps to query Grafana:

1. Get an access token
2. Locate the datasource id
3. Query data

```
# ===== REQUEST =====  
url = f"{GRAFANA_URL}/api/datasources"  
  
headers = {  
    "Host": HOST_HEADER,  
    "Authorization": f"Bearer {TOKEN}"  
}  
  
response = requests.get(url, headers=headers)  
  
# ===== OUTPUT =====  
print("Status:", response.status_code)
```

Status: 200

✓ Available datasources:

- Name: Loki
UID: P8E80F9AEF21F6940
Type: loki
URL: <http://loki:3100>
- Name: Prometheus
UID: PBFA97CFB590B2093
Type: prometheus
URL: <http://prometheus:9090>



Monitoring-aaS: Grafana APIs and Notebook Integration

Grafana exposes a rich set of APIs that enable powerful queries on monitored sources

Steps to query Grafana:

1. Get an access token
2. Locate the datasource id
3. Query data

```
# ===== CONFIG =====
DATASOURCE_UID = "PBFA97CFB590B2093" # set this

# adjust depending on your Labels
PROM_QUERY = 'sum(rate(container_cpu_time_seconds_total{ }[$__rate_interval]))'

# ===== TIME RANGE =====
now = int(time.time() * 1000)
five_min_ago = int((time.time() - 300) * 1000)

# ===== REQUEST =====
url = f"{GRAFANA_URL}/api/ds/query"

headers = {
    "Host": HOST_HEADER,
    "Authorization": f"Bearer {TOKEN}",
    "Content-Type": "application/json"
}

payload = {
    "queries": [
        {
            "refId": "A",
            "expr": PROM_QUERY,
            "datasource": {
                "type": "prometheus",
                "uid": DATASOURCE_UID
            },
            "intervalMs": 5000,
            "maxDataPoints": 100
        }
    ],
    "from": str(five_min_ago),
    "to": str(now)
}

resp = requests.post(url, headers=headers, data=json.dumps(payload))
```



Monitoring-aaS: Grafana APIs and Notebook Integration

```
resp = requests.post(url, headers=headers, data=json.dumps(payload))

if resp.status_code != 200:
    print("❌ Query failed:", resp.text)
    raise SystemExit

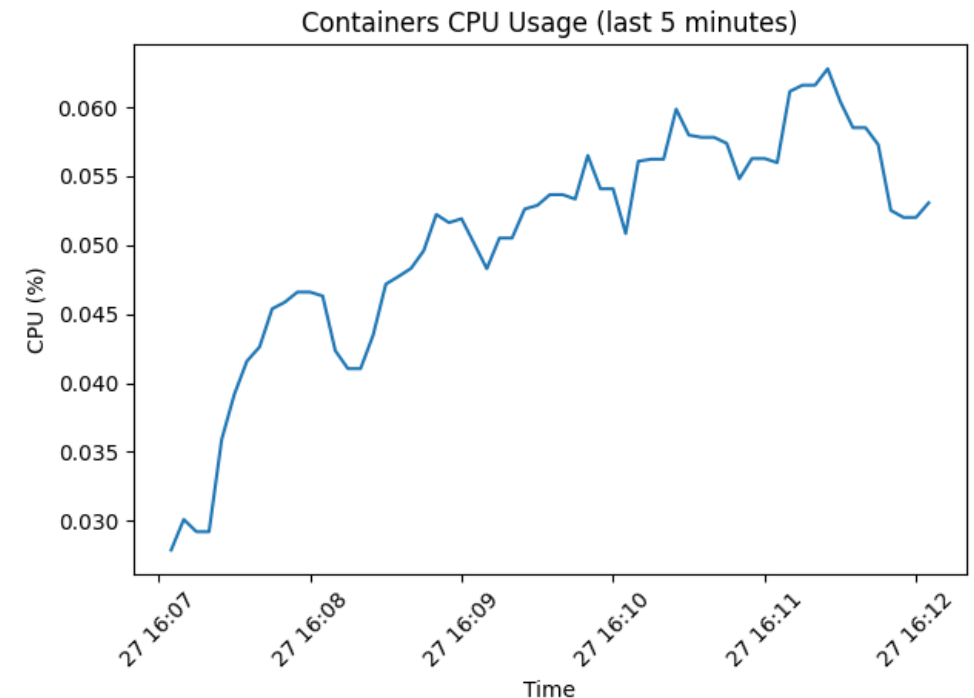
data = resp.json()

# ===== PARSE RESPONSE =====
frames = data["results"]["A"]["frames"]
values = frames[0]["data"]["values"]

timestamps = values[0]
cpu_values = values[1]

# convert timestamps → datetime
times = [datetime.fromtimestamp(t / 1000) for t in timestamps]

# ===== PLOT =====
plt.figure()
plt.plot(times, cpu_values)
plt.xlabel("Time")
plt.ylabel("CPU (%)")
plt.title("Containers CPU Usage (last 5 minutes)")
plt.xticks(rotation=45)
plt.tight_layout()
plt.show()
```



Ping Pong Cloud Continuum Experiment

Objective

- Demonstrate multi-cluster deployment across the Cloud Continuum
- Evaluate inter-cluster communication between distributed services
- Scenario
- Two simple services:
 - Ping → sends requests
 - Pong → receives and responds
- Deployed on different Kubernetes clusters (e.g., edge vs central)
- Focus of the experiment
- Cross-cluster connectivity
- Service exposure and routing
- Impact of network conditions on communication



https://gitlab.com/MMw_Unibo/platformeng/ping-pong-test



Ping Pong Cloud Continuum Experiment

For example, the deployment descriptor for **service B** (to one that receives Pings), defines two Kubernetes objects:

1) A Kubernetes deployment specifying:

- The container image of the service deployed
- A name
- The namespace
- And the port exposed to receive ping requests

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: answer-random
spec:
  replicas: 1
  selector:
    matchLabels:
      app: answer-random
  template:
    metadata:
      labels:
        app: answer-random
    spec:
      containers:
        - name: answer-random
          image: registry.gitlab.com/mmw_unibo/platformeng/ping-pong-test/answer-random
          ports:
            - containerPort: 5000
```

Ping Pong Cloud Continuum Experiment

For example, the deployment descriptor for **service B** (to one that receives Pings), defines two Kubernetes objects:

2) A Kubernetes Service describing:

- The deployment to which it is connected
- A NodePort ingress that instruments Kubernetes to redirect packets sent to node2 to that matching a specific port to the deployment specified

```
---
apiVersion: v1
kind: Service
metadata:
  name: answer-random
spec:
  selector:
    app: answer-random
  ports:
    - protocol: TCP
      port: 5000
      targetPort: 5000
      nodePort: 30007
  type: NodePort
```

Setup ping services

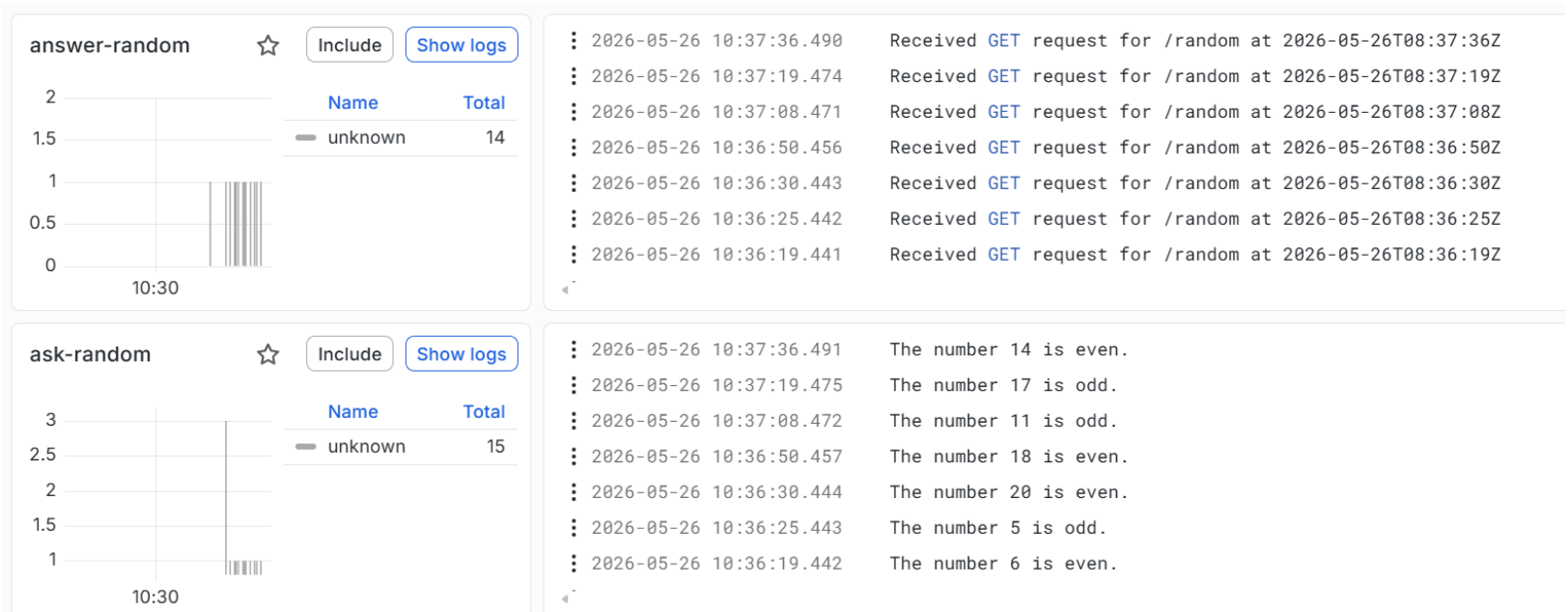
- Use Slices Bi or SFCC kubectl to get the IP of Master cluster
- Set the ip as ENV variable in the Ping service deployed on edge1

```
io.k8s.api.core.v1.Service (v1@service.json) | io.k8s.api.apps.v1.Deployment (v1@deployment.json)
1  ∨ apiVersion: apps/v1
2  kind: Deployment
3  ∨ metadata:
4    name: ask-random
5  ∨ spec:
6    replicas: 1
7    selector:
8      matchLabels:
9        app: ask-random
10   template:
11     metadata:
12       labels:
13         app: ask-random
14     spec:
15       containers:
16         - name: ask-random
17           image: registry.gitlab.com/mmw_unibo/platformeng/ping-pong-test/ask-random:latest
18           ports:
19             - containerPort: 8080
20           env:
21             - name: RANDOM_ADDRESS
22               value: "10.10.219.147"
23             - name: RANDOM_PORT
24               value: "30007"
```



Ping Pong Service: Monitoring-aaS

```
andrea@SURFACESVENTURA:~/workspace/slices/ping-pong-test$ kubectl logs -f ask-random-75cb57d9c7-89d7z --kubeconfig /home/andrea/SlicesFile/kubeconfigs/edge1.yaml
address: 10.10.219.147, port: 30007
Starting server on :8080
The number 19 is odd.
The number 19 is odd.
The number 3 is odd.
The number 4 is even.
```



Ping Pong Experiment: Interactive mode + TC

Objective: Start an interactive session inside a container hosted in one of the two clusters and experiment with packet loss and packet delay among the two Kubernetes clusters

Tools:

- *Tc*: Traffic control utility for shaping, scheduling, policing, and dropping network traffic
- *Curl pod*: container image with curl installed and enabling interactive bash sessions

To start an interactive session:

1) deploy a new pod with curl in node3

```
kubectrl apply -f deployments/curlpod.yaml --namespace=experiment1-namespace
```

2) And then execute a console inside the pod

...



Ping Pong Experiment: Interactive mode

```
andrea@SURFACESVENTURA:~/workspace/slices/ping-pong-test$ kubectl apply -f
curldeploy.yaml --kubeconfig /home/andrea/SlicesFile/kubeconfigs/edge2.yaml
pod/curl created
```

```
andrea@SURFACESVENTURA:~/workspace/slices/ping-pong-test$ kubectl get pods --kubeconfig
/home/andrea/SlicesFile/kubeconfigs/edge2.yaml
```

NAME	READY	STATUS	RESTARTS	AGE
curl	1/1	Running	0	34s
otel-collector-opentelemetry-collector-agent-vfdrl	1/1	Running	0	37m

```
andrea@SURFACESVENTURA:~/workspace/slices/ping-pong-test$ kubectl exec -it curl --
kubeconfig /home/andrea/SlicesFile/kubeconfigs/edge2.yaml -- /bin/sh
```

```
~ $
```

```
~ $
```

```
~ $
```

```
~ $
```

```
~ $
```

```
~ $ curl 10.10.219.147:30007/random
```

```
20~ $
```

```
~ $ curl -s -w ' Latency: %{time_total}s\n' 10.10.219.147:30007/random
```

```
10 Latency: 0.001308s
```



Ping Pong Cloud Continuum Experiment: Interactive mode + TC

3) In a second terminal connect to edge2 and set a 50% packet loss on incoming connections

TC Utilities (Linux Traffic Control)

- *Used to manage and shape network traffic*
- *Controls bandwidth, delay, priority, and packet loss*
- *Works with queuing disciplines (qdisc)*

Key components:

- *qdisc – traffic scheduling*
- *class – bandwidth allocation*
- *filter – packet classification*

```
ssh -i /root/.ssh/ccbp_key ubuntu@10.10.217.87
```

```
Welcome to Ubuntu 24.04.3 LTS (GNU/Linux 6.8.0-85-generic x86_
```

Common uses:

- *Bandwidth limiting*
- *Traffic prioritization (QoS)*
- *Network simulation (latency, loss)*

```
$ sudo tc qdisc add dev enp6s18 root netem loss 50%
```

Or

```
#30 ms average delay with ±5 ms variation
```

```
sudo tc qdisc add dev enp6s18 root netem delay 30ms 5ms
```



Ping Pong Cloud Continuum Experiment: Interactive mode + TC

```
$ ip a
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
        valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host noprefixroute
        valid_lft forever preferred_lft forever
2: enp6s18: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP group default qlen 1000
    link/ether bc:24:11:29:4f:65 brd ff:ff:ff:ff:ff:ff
    inet 10.10.217.87/20 metric 100 brd 10.10.223.255 scope global dynamic enp6s18
        valid_lft 3144sec preferred_lft 3144sec
    inet6 2001:6a8:1d80:2042:be24:11ff:fe29:4f65/64 scope global dynamic mngtmpaddr noprefixroute
        valid_lft 86396sec preferred_lft 14396sec
    inet6 fe80::be24:11ff:fe29:4f65/64 scope link
        valid_lft forever preferred_lft forever
```



Ping Pong Cloud Continuum Experiment: Interactive mode + TC

4) In the first terminal connected to the *curl* pod test the request latency when setting different loss conditions

```
curl -s -w ' Latency: %{time_total}s\n' 10.10.219.147:30007/random
17 Latency: 4.550513s
~ $ curl -s -w ' Latency: %{time_total}s\n' 10.10.219.147:30007/random
1 Latency: 0.001989s
~ $ curl -s -w ' Latency: %{time_total}s\n' 10.10.219.147:30007/random
17 Latency: 0.001492s
~ $ curl -s -w ' Latency: %{time_total}s\n' 10.10.219.147:30007/random
6 Latency: 0.001318s
~ $ curl -s -w ' Latency: %{time_total}s\n' 10.10.219.147:30007/random
8 Latency: 2.702756s
```



Advanced Services of the CCBP-aaS



Additional Services of the CCBP-aaS

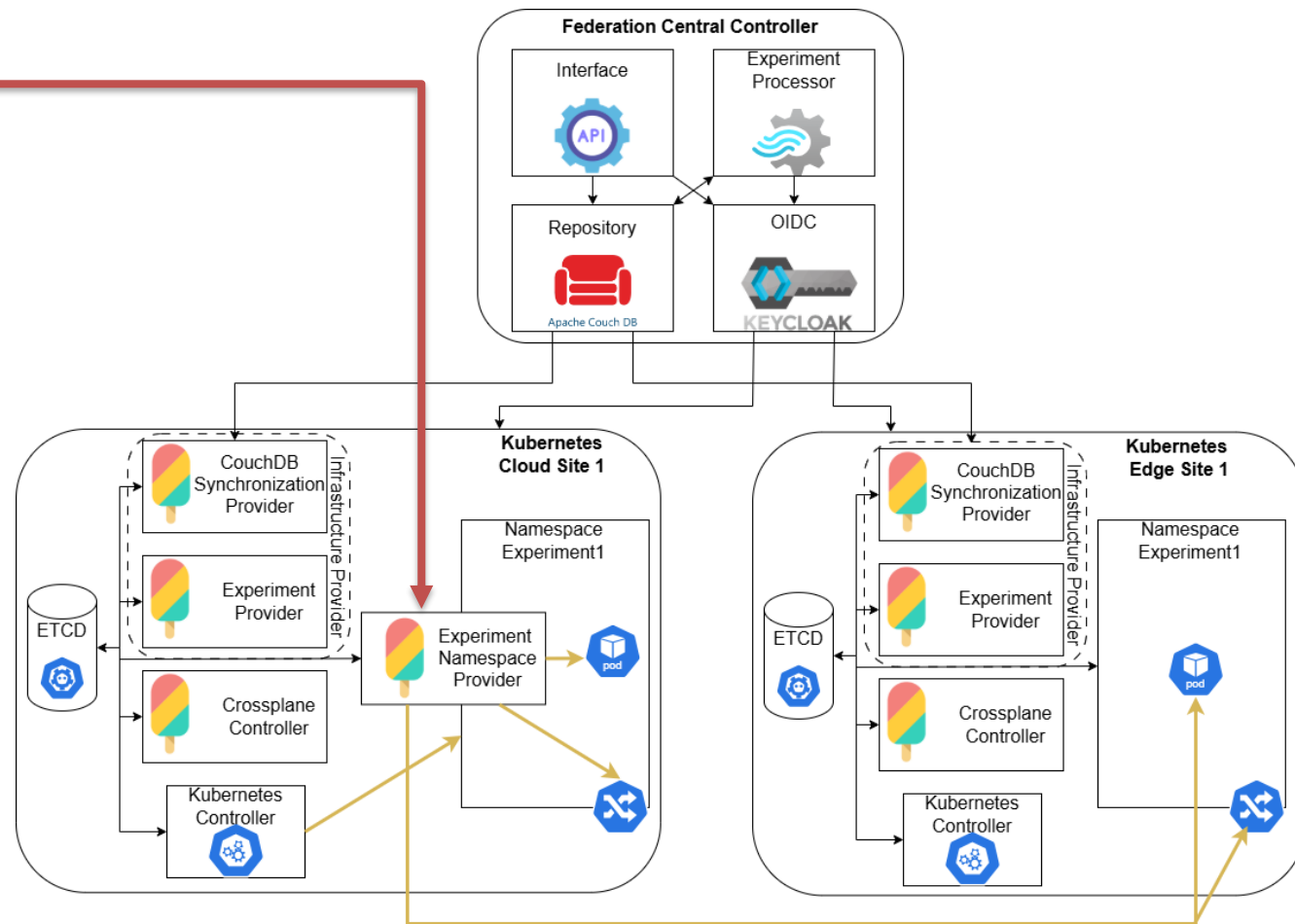
- DNS-aaS
- Openstack provisioner
- Openstack Controller
- Experiment Namespace Provider
- Federated authentication
- Extended Sync Operator



Experiment Namespace Provider

Experiment Namespace Provider: Simplify CC experiment provisioning by providing a *unique endpoint* for the provisioning and configuration of CC Deployments

- Embodies a CC Controller-as-Service model
- Interacts with remote cluster through “standard” Kubernetes API
- Enables the creation of any Kubernetes Object supported by remote clusters (including Custom Objects or SLICES Objects)
- Supports and integrates with Kubernetes:
 - Authorization, Authentication, Auditing
 - Monitoring and Logging
 - High Availability



Ping Pong Cloud Continuum Experiment

Second approach: make a deployment through the Experiment Namespace Provider

The podNGINX-claim.yaml create a Kubernetes object that triggers the **Experiment Namespace Provider** by defining a new Remote Resource Claim

The **Experiment Namespace Provider** extracts the inner definition of a Kubernetes object and apply it to the remote cluster identified by the provider config reg

```
apiVersion: experiment.mmwunibo.it/v1alpha1
kind: RemoteResourceClaim
metadata:
  ## insert here the manifest of the kubernetes resource
  name: test-pod-creato-da-host
spec:
  manifest:
    apiVersion: v1
    kind: Pod
    metadata:
      name: pod-created-by-host-node2
      namespace: experiment1-namespcae
      labels:
        app: example
    spec:
      containers:
        - name: nginx-container
          image: nginx:latest
          ports:
            - containerPort: 80
          resources:
            requests:
              memory: "64Mi"
              cpu: "250m"
            limits:
              memory: "128Mi"
              cpu: "500m"
      providerConfigRef:
        name: node3-experiment1-providerconfig
```

Federated Authentication (Keycloak + OpenID)

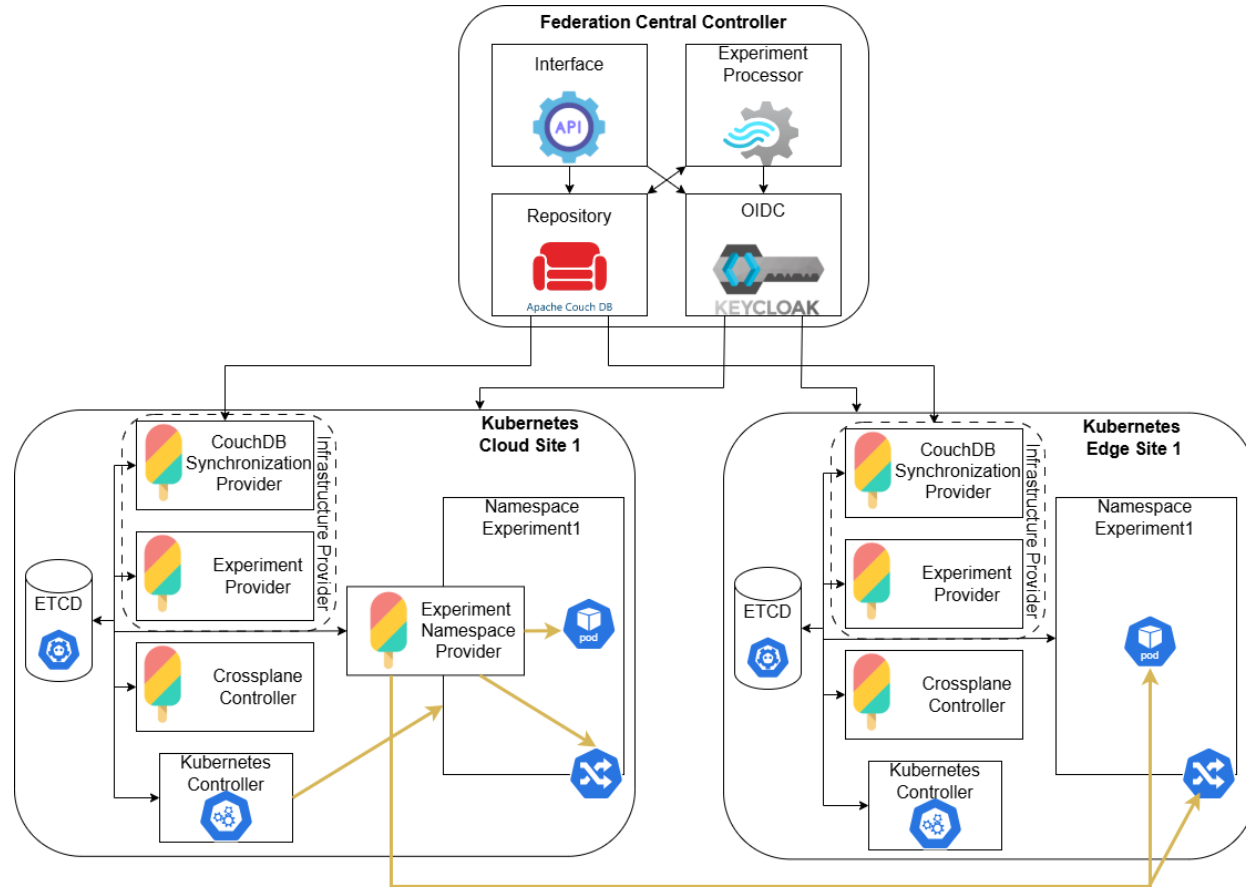
- Enable **secure, role-based access** to Cloud Continuum resources
- Support **multi-cluster management** and controlled experiment access

Implementation

- Based on **Keycloak** for identity and access management
- **Federation via OpenID Connect (OIDC)**
- Integration with **SLICES accounts** for unified login

User Roles

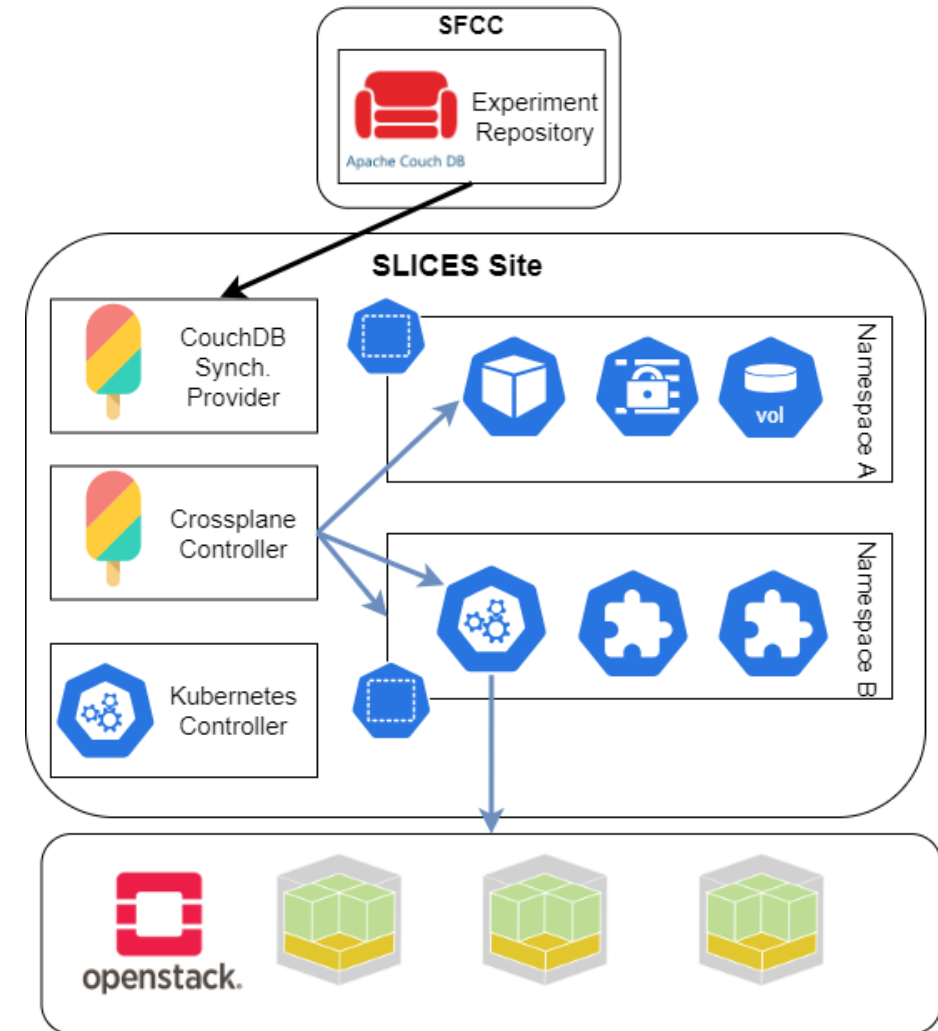
- **Administrators**
 - Full access to infrastructure
 - Manage **multi-cluster orchestration**
 - Handle **resource provisioning and configuration**
- **Experimenters**
 - Limited, scoped access
 - Operate on **assigned slices/namespaces**
 - Interact only with **allocated resources**



The Extended Sync Operator

The Sync Operator can natively provision any type of Kubernetes Resources defined in Couchdb including:

- To assign a Slices of available resources to an experiment
- Synchronize the configurations needed including authentication and authorization across sites to create the slice
- Configuration
- Services, Jobs
- Definition of resources to be provisioned on external systems (e.g. Openstack, Kubernetes)



CCBP-aaS Examples

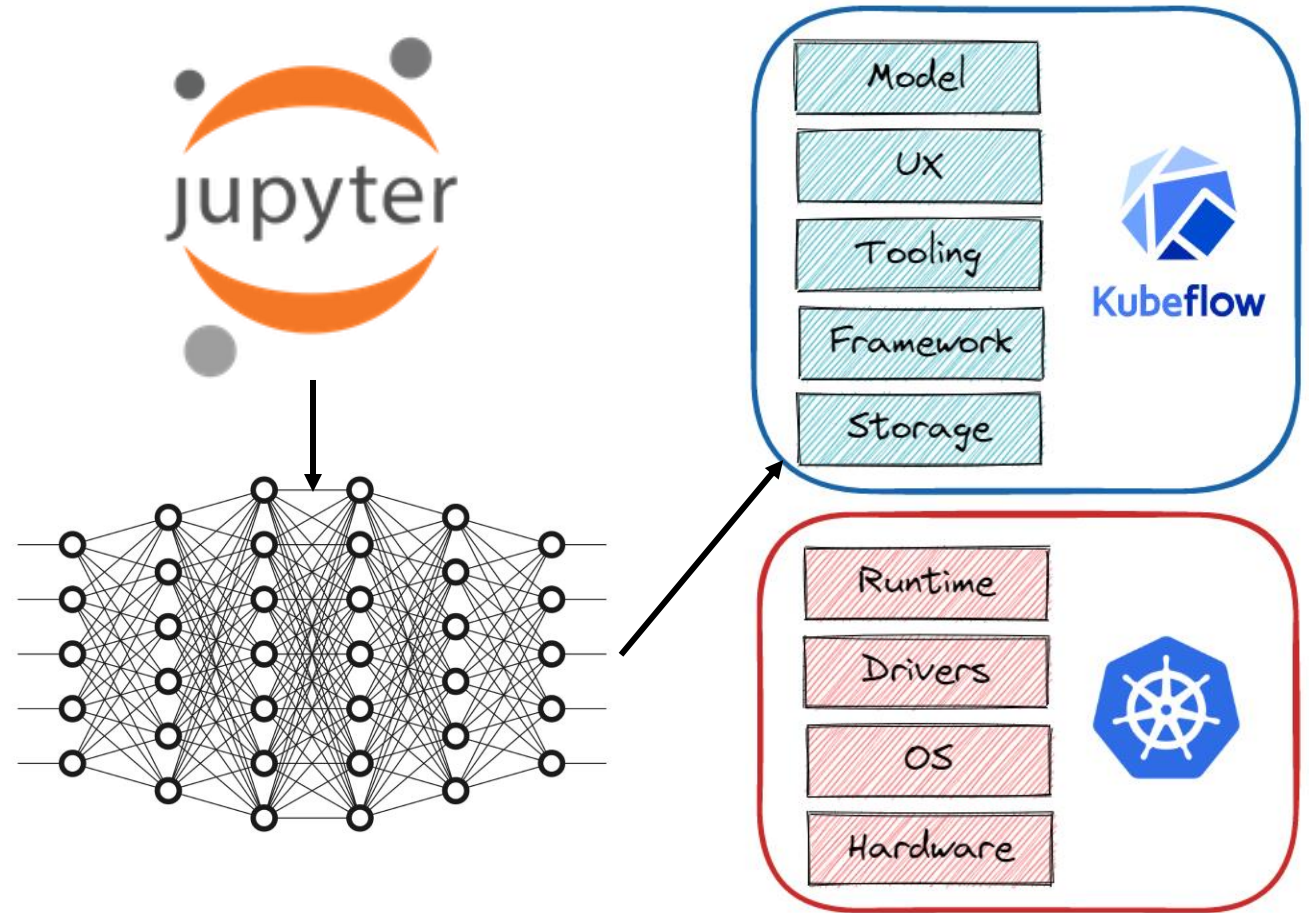
Federated Learning Blueprint Integration: MLOps

Scenario: MLOps

Testing the distribution and performance of trained machine learning model in a multi-cluster cloud edge environment:

Setup:

1. Use Federated Learning Blueprint to train the model
2. Export the model (e.g. ONNX)
3. Create a Cloud-Edge scenario with the CCBP
4. Install Kubeflow on each cluster with kubectl
5. Deploy the model
6. Observe results on Mon-aaS



Cloud Continuum LoRA Sensor Network

CCBP Application: LoRa Cloud Continuum Scenario

- Distributed LoRa devices deployed in the field (e.g., sensors, meters, IoT nodes)
- Devices communicate via LoRaWAN (low-power, long-range)
- Data collected by LoRa gateways and integrated into the Cloud Continuum
- Edge layer:
 - LoRa devices generate telemetry
 - Battery-powered, constrained, intermittent connectivity

Goal

- Integrate real IoT devices into CCBP multi-cluster infrastructure
- Enable reproducible IoT experimentation at scale



Cloud Continuum LoRA Sensor Network

Sensor Layer

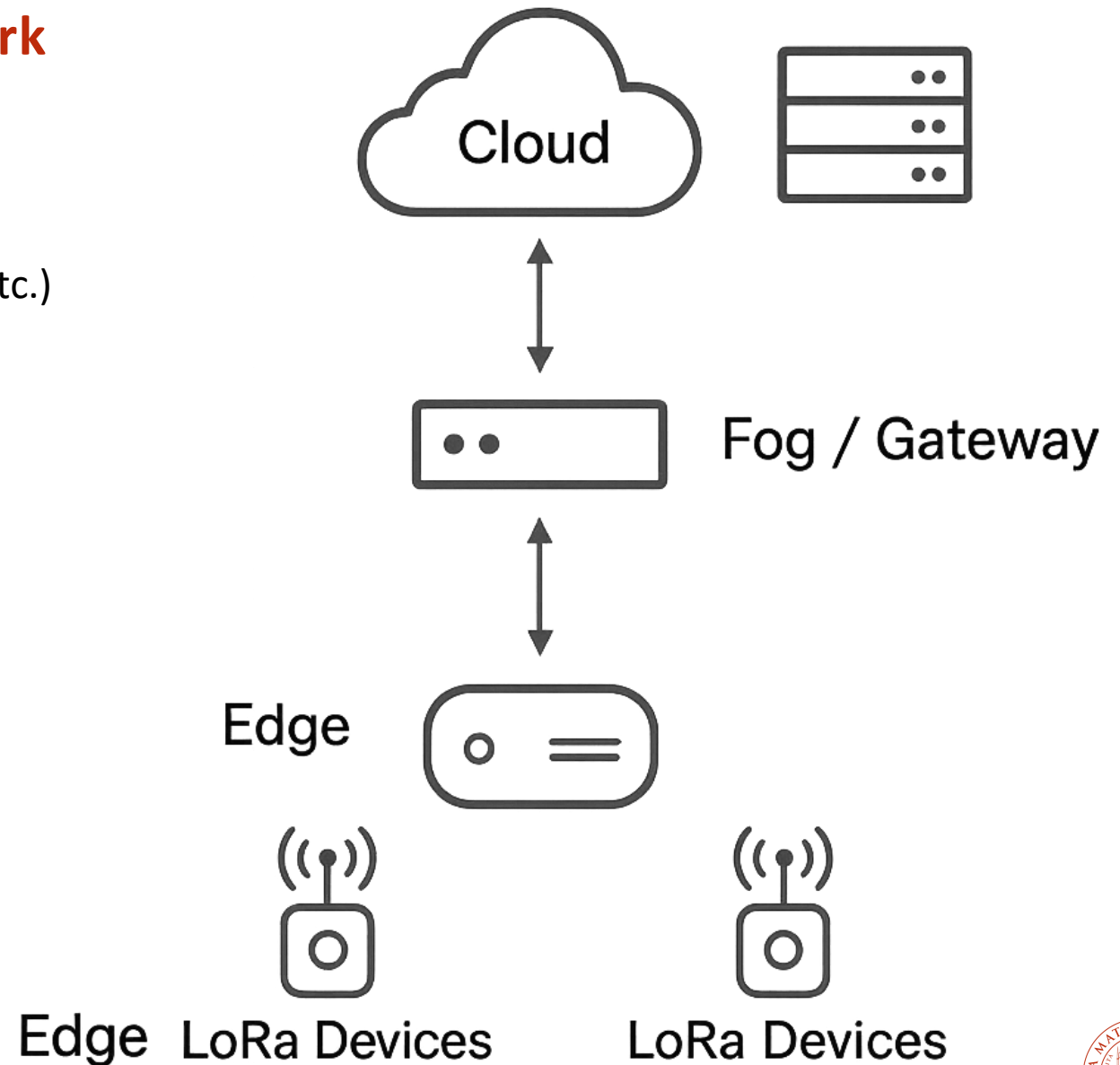
LoRa sensors (temperature, energy, environment, etc.)
Minimal processing, continuous data generation

Gateway Layer

LoRa gateways bridge LoRa → IP
Optional local buffering and filtering
Connected via VPN-aaS (CCBP)

Cloud layer

ChirpStack deployed on Kubernetes (CCBP clusters)
Services processing IoT data



Cloud Continuum Data Processing

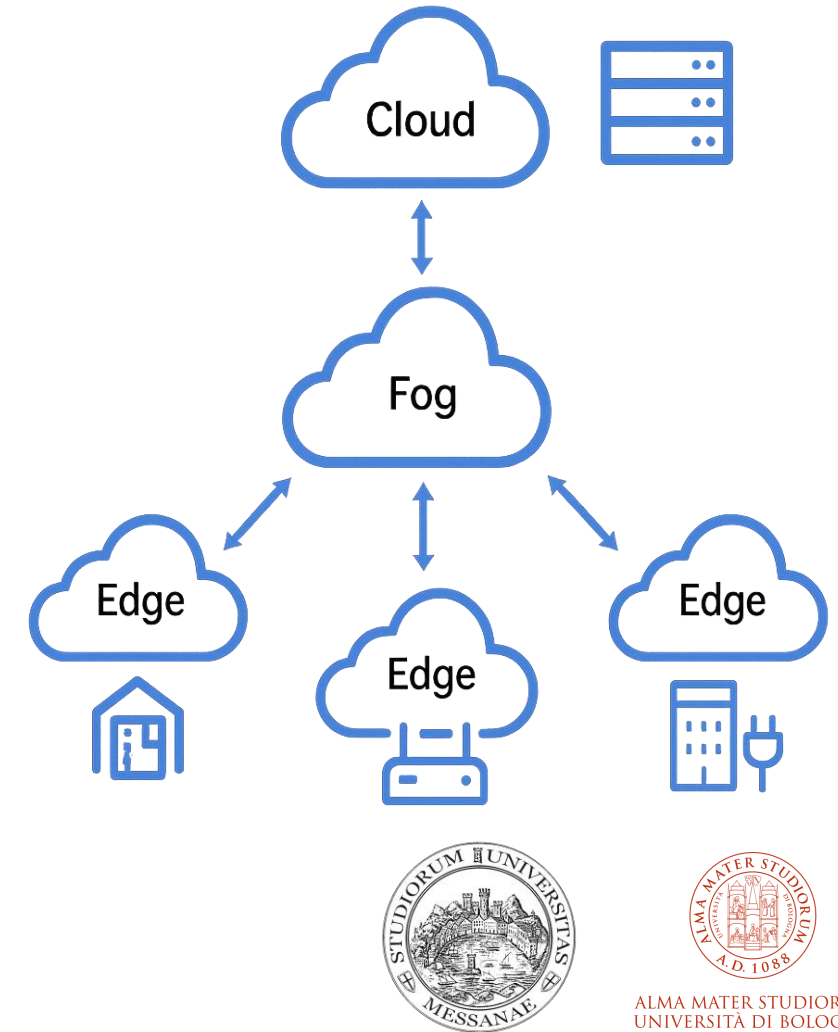
Team

1. University of Bologna
2. University of Messina

Fabio Orazio Mirto, Giuseppe Tricomi, Luca D'Agati, Andrea Sabbioni, Stefano Silvestri, Francesco Longo, Giovanni Merlino, Armir Bujari, Paolo Bellavista, Antonio Puliafito

Deployment:

- 1 VM and 2 IoT devices in Messina
- 1 Kubernetes Cluster in Bologna
- 3 Large Nodes Kubernetes Cluster on Slices BI
- Monitoring-aaS
- VPN-aaS
 - Messina<->Bologna: 10 ms
 - Bologna<->Slices BI: 30 ms
 - Messina<->Slices BI: 50 ms



Cloud Continuum Data Processing

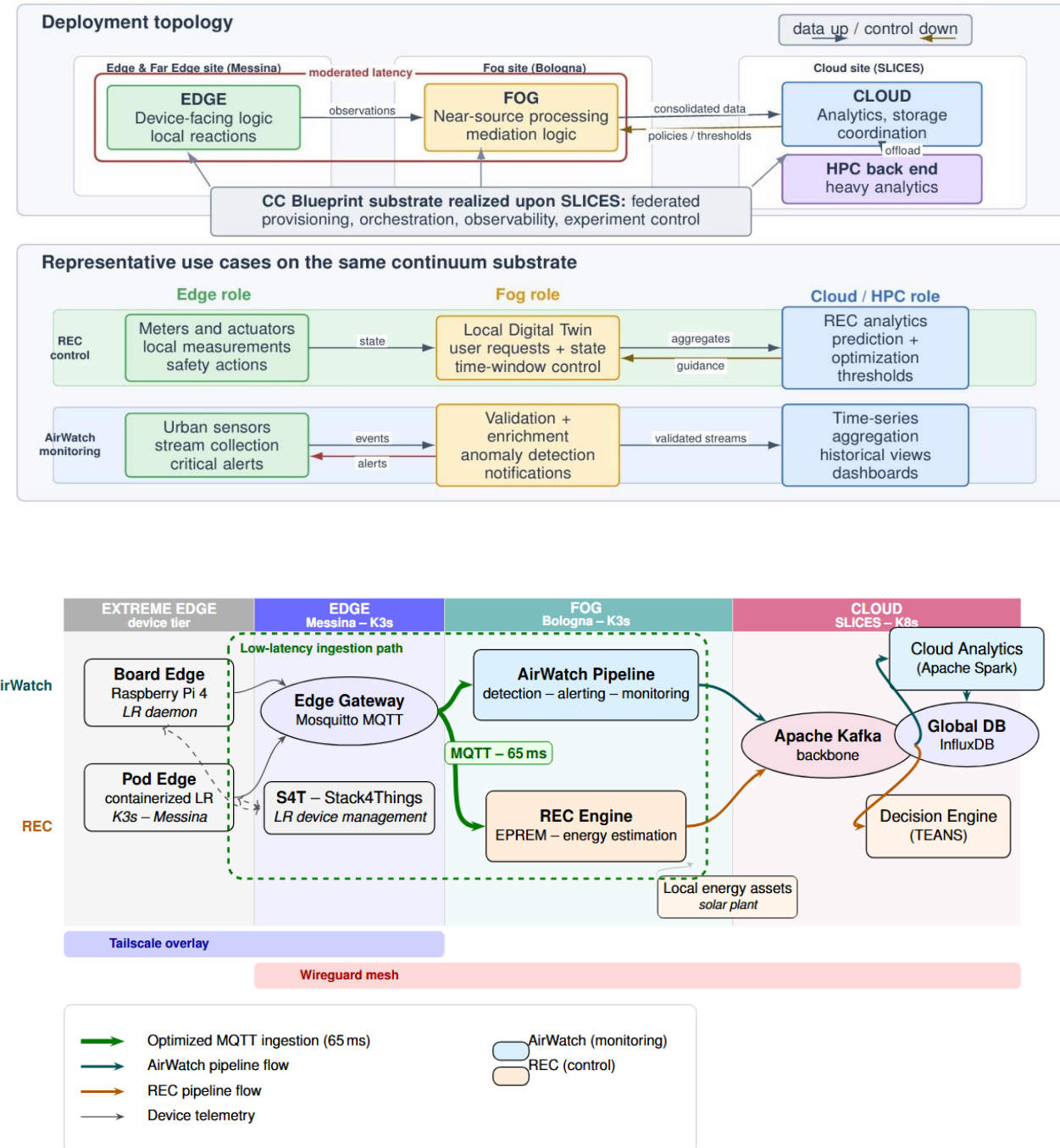
Use Cases:

Renewable Energy Community management:

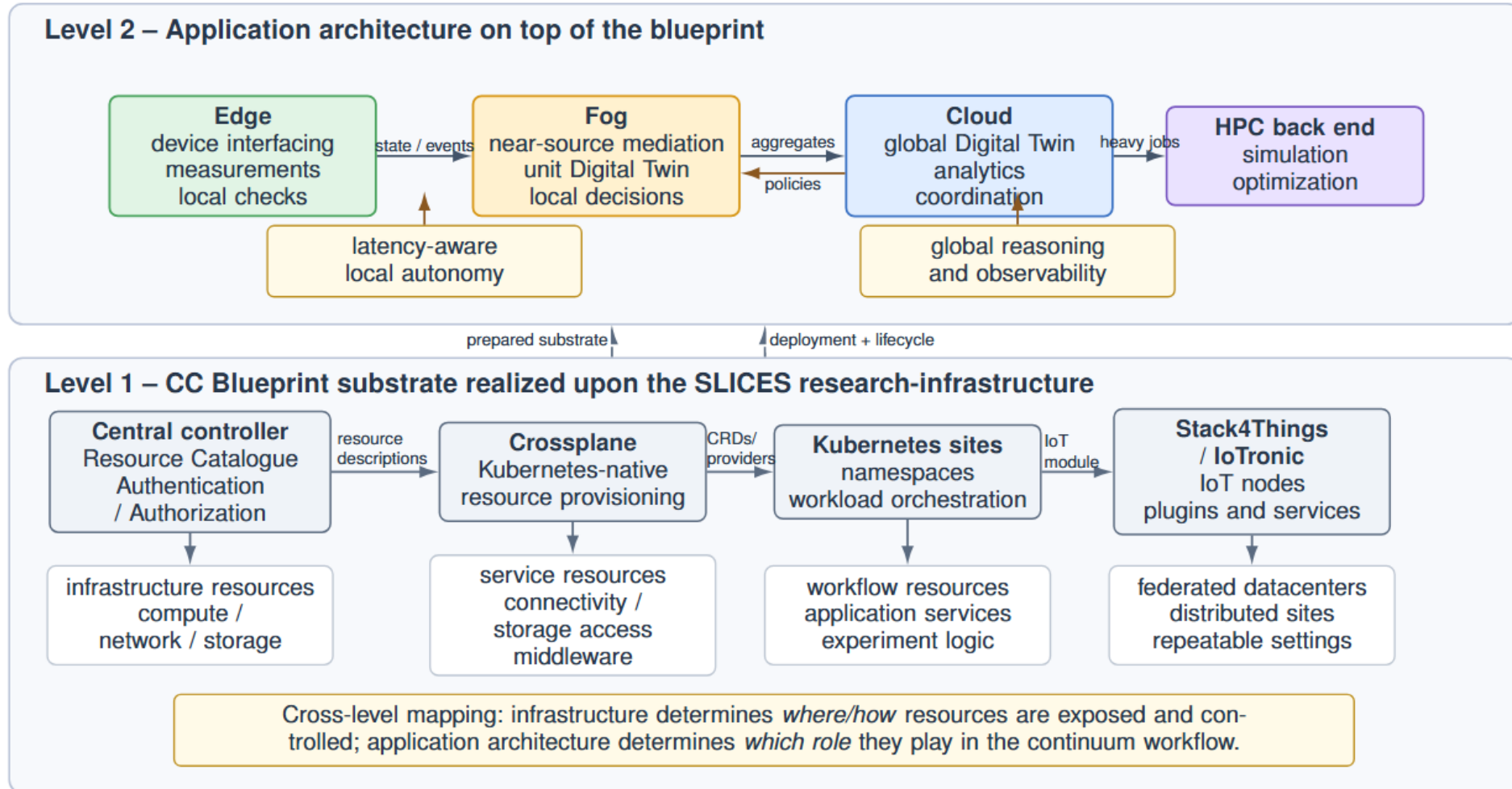
the continuum supports distributed Digital Twin coordination and time-window-based energy control

AirWatch:

Urban monitoring pipeline focused on anomaly detection, low-latency alerting, and cloud-side aggregation.



Cloud Continuum Data Processing: integration S4T in CC-Blueprint



Stack4Things Pipeline (3 Boards)

Operational View (current provider-s4t CRDs)

 ProviderConfig

s4t-provider-domain

(used by all resources)



Plugin (injected on board)



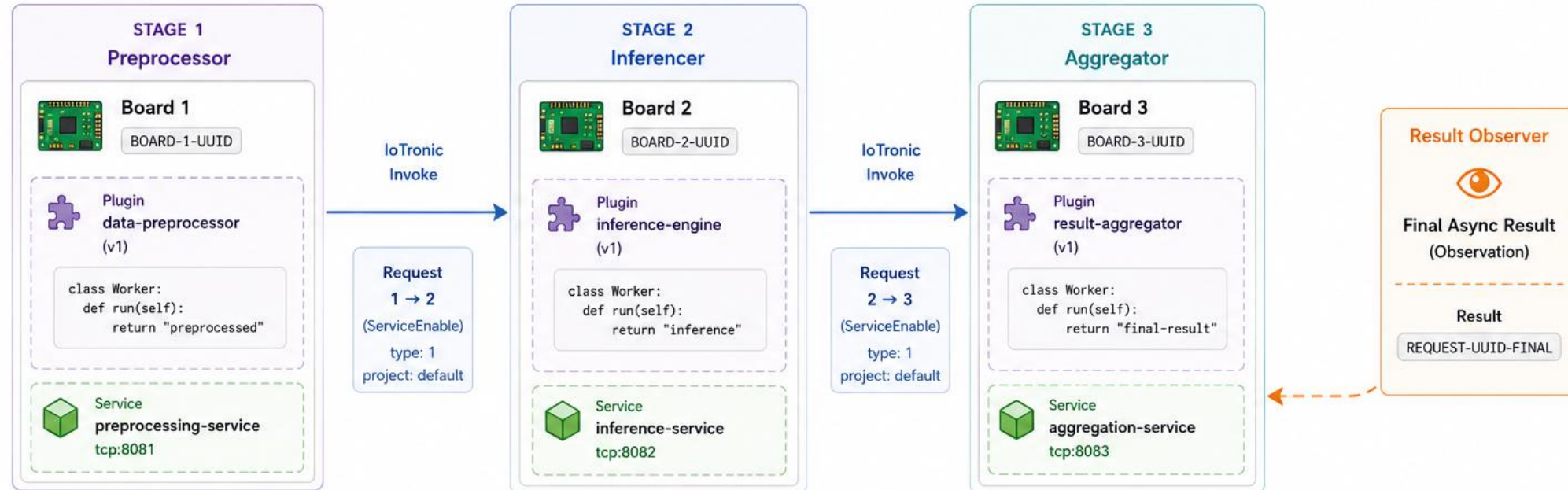
Service (exposed on board)



Invocation (IoTronic Request)



Async Result (Observation)



Pipeline Flow



preprocessor



inferencer



aggregator



final async result

Underlying Resources (current provider-s4t CRDs)

 Plugins

PLUGIN-UUID-DATA-PREPROCESSOR

PLUGIN-UUID-INFERENCER

PLUGIN-UUID-AGGREGATOR

 Services

SERVICE-UUID-PREPROCESSING
tcp:8081

SERVICE-UUID-INFERENCER
tcp:8082

SERVICE-UUID-AGGREGATION
tcp:8083

 BoardPluginInjection

BOARD-1-UUID → PLUGIN-UUID-DATA-PREPROCESSOR

BOARD-2-UUID → PLUGIN-UUID-INFERENCER

BOARD-3-UUID → PLUGIN-UUID-AGGREGATOR

 BoardServiceInjection

BOARD-1-UUID → SERVICE-UUID-PREPROCESSING

BOARD-2-UUID → SERVICE-UUID-INFERENCER

BOARD-3-UUID → SERVICE-UUID-AGGREGATION

 Requests (IoTronic)

REQUEST 1 → 2
(ServiceEnable, type: 1, project: default)

REQUEST 2 → 3
(ServiceEnable, type: 1, project: default)

 Result

REQUEST-UUID-FINAL
(Observation)

Future Composite API (draft)
XStack4ThingsPipeline



Single declarative object
describing the entire pipeline:

- stages[] (plugin + service + board)
- edges[] (fromStage → toStage, action)

Composition renders all underlying
CRDs automatically.

A Pipeline example

```
apiVersion: iot.s4t.crossplane.io/v1alpha1
kind: Plugin
metadata:
name: draft-plugin-data-preprocessor
labels:
draft.example: "true"
spec:
providerConfigRef:
name: s4t-provider-domain
forProvider:
name: data-preprocessor
version: v1
parameters:
mode: local
nextStage: inference-engine
code: |
# DRAFT example plugin code placeholder
class Worker:
def run(self):
return "preprocessed"
--
apiVersion: iot.s4t.crossplane.io/v1alpha1
kind: Service
metadata:
name: draft-service-preprocessing
labels:
draft.example: "true"
spec:
providerConfigRef:
name: s4t-provider-domain
forProvider:
name: preprocessing-service
port: 8081
protocol: tcp
--
```

```
apiVersion: iot.s4t.crossplane.io/v1alpha1
kind: BoardPluginInjection
metadata:
name: draft-inject-plugin-board-1
labels:
draft.example: "true"
spec:
providerConfigRef:
name: s4t-provider-domain
forProvider:
boardUuid: "BOARD-1-UUID"
pluginUuid: "PLUGIN-UUID-DATA-PREPROCESSOR"
--
apiVersion: iot.s4t.crossplane.io/v1alpha1
kind: Plugin
metadata:
name: draft-plugin-inference-engine
labels:
draft.example: "true"
spec:
providerConfigRef:
name: s4t-provider-domain
forProvider:
name: inference-engine
version: v1
parameters:
mode: local
nextStage: result-aggregator
code: |
# DRAFT example plugin code placeholder
class Worker:
def run(self):
return "inference"
--
```

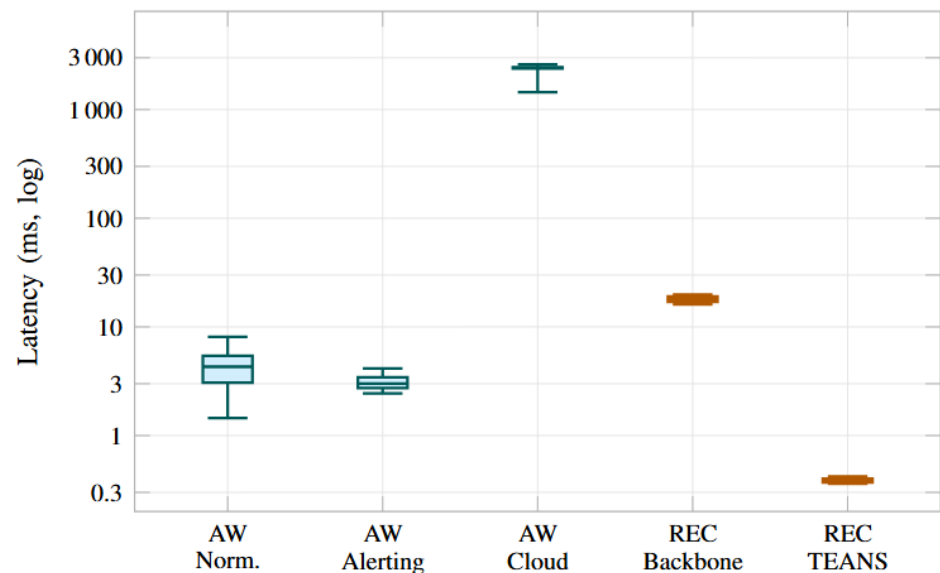
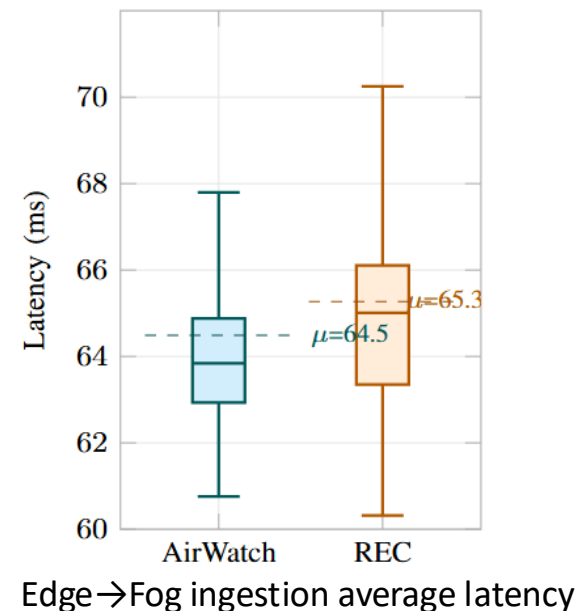
```
apiVersion: iot.s4t.crossplane.io/v1alpha1
kind: Service
metadata:
name: draft-service-inference
labels:
draft.example: "true"
spec:
providerConfigRef:
name: s4t-provider-domain
forProvider:
name: inference-service
port: 8082
protocol: tcp
--
apiVersion: iot.s4t.crossplane.io/v1alpha1
kind: BoardPluginInjection
metadata:
name: draft-inject-plugin-board-2
labels:
draft.example: "true"
spec:
providerConfigRef:
name: s4t-provider-domain
forProvider:
boardUuid: "BOARD-2-UUID"
pluginUuid: "PLUGIN-UUID-INFERENCE"
--
apiVersion: iot.s4t.crossplane.io/v1alpha1
kind: Plugin
metadata:
name: draft-plugin-result-aggregator
labels:
draft.example: "true"
spec:
providerConfigRef:
name: s4t-provider-domain
forProvider:
name: result-aggregator
version: v1
```

```
parameters:
mode: local
finalStage: true
code: |
# DRAFT example plugin code placeholder
class Worker:
def run(self):
return "final-result"
--
apiVersion: iot.s4t.crossplane.io/v1alpha1
kind: Service
metadata:
name: draft-service-aggregation
labels:
draft.example: "true"
spec:
providerConfigRef:
name: s4t-provider-domain
forProvider:
name: aggregation-service
port: 8083
protocol: tcp
--
apiVersion: iot.s4t.crossplane.io/v1alpha1
kind: BoardPluginInjection
metadata:
name: draft-inject-plugin-board-3
labels:
draft.example: "true"
spec:
providerConfigRef:
name: s4t-provider-domain
forProvider:
boardUuid: "BOARD-3-UUID"
pluginUuid: "PLUGIN-UUID-AGGREGATOR"
```

Cloud Continuum Data Processing: Results

The REC workflow evaluates coordination in a Renewable Energy Community using a low-latency push path. Data is streamed from edge prosumers to the Fog-side logic for real-time Digital Twin synchronization.

AirWatch stresses the continuum as a monitoring pipeline, where Fog services ingest environmental records via MQTT, normalize them, and forward them via Kafka toward Cloud-side Spark aggregation.



Average latency of each stages composing pipelines across cloud continuum

Conclusion

The CC Blueprint has shown to be an innovative architecture that facilitates the provisioning and management of CC resources and services for SLICES experimenters

Key Features:

- Simplicity
- Failure tolerance
- Easy extensibility
- Interoperability with existing tools
- Modular

The CC Blueprint is offered as **open-source project to be tested, deployed, AND EXTENDED** by researchers and practitioners.



https://gitlab.com/MMw_Unibo/platformeng/slices-blueprint



ALMA MATER STUDIORUM
UNIVERSITÀ DI BOLOGNA

Future Work

- DNS-aaS Complete Integration
- Extend Presets
 - Better Integration with federation stack
 - Ready to use environments (Serverless, ML-OPS...)
- New version of the Kubernetes Operator
- Advanced Monitoring-aaS:
 - Traces
 - Filters
 - Automated export of single datasets to S3
 - Automatic rotation of signals
- Integration with Ansible for provisioned VM automation
- GUI to support: Infrastructure Creation and Slices definition
- One-Click deployment of Experiments
- Extensive Documentation





ALMA MATER STUDIORUM
UNIVERSITÀ DI BOLOGNA

Q&A

Team

Paolo Bellavista, Armir Bujari, Andrea Sabbioni

Department of Computer Science and Engineering, University of Bologna, Italy

paolo.bellavista@unibo.it

armir.bujari@unibo.it

andrea.sabbioni5@unibo.it

www.unibo.it

References

- [1] L. Bittencourt et al., “The Internet of Things, Fog and Cloud continuum: Integration and challenges,” Internet of Things, vol. 3-4, pp. 134–155, 2018.
- [2] Slices-Blueprint. Mobile Middleware Research Group. Accessed: Apr. 16, 2025. [Online]. Available: <https://gitlab.com/MMwUnibo/platformeng/slices-blueprint>
- [6] F. B. Oliveira, M. Di Felice, and C. Kamienski, “IoTDeploy: Deployment of IoT Smart Applications over the Computing Continuum,” Internet of Things, vol. 28, p. 101348, 2024.
- [7] L. Osmani, T. Kauppinen, M. Komu, and S. Tarkoma, “Multi-Cloud Connectivity for Kubernetes in 5G Networks,” IEEE Communications Magazine, vol. 59, no. 10, pp. 42–47, 2021.
- [8] E. Song et al., “Canal Mesh: A Cloud-Scale Sidecar-Free Multi-Tenant Service Mesh Architecture,” in Proc. of the ACM SIGCOMM 2024 Conference. New York, NY, USA: Association for Computing Machinery, 2024, p. 860–875.
- [11] T. Goethals, F. De Turck, and B. Volckaert, “Extending kubernetes clusters to low-resource edge devices using virtual kubelets,” IEEE Transactions on Cloud Computing, vol. 10, no. 4, pp. 2623–2636, 2020.
- [12] M. Iorio, F. Risso, A. Palesandro, L. Camiciotti, and A. Manzalini, “Computing without borders: The way towards liquid computing,” IEEE Transactions on Cloud Computing, vol. 11, no. 3, pp. 2820–2838, 2023.
- [13] F. Faticanti, D. Santoro, S. Cretti, and D. Siracusa, “An Application of Kubernetes Cluster Federation in Fog Computing,” in Proc. of Conference on Innovation in Clouds, Internet and Networks and Workshops (ICIN), 2021, pp. 89–91.
- [14] L. Larsson, H. Gustafsson, C. Klein, and E. Elmroth, “Decentralized kubernetes federation control plane,” in Proc. of IEEE/ACM International Conference on Utility and Cloud Computing (UCC), 2020, pp. 354–359.

