



When 99.999% Isn't Enough: Exploring Swisscom Kubernetes and Particle Accelerator Observability

Thursday, 4
December 2025



KCD Suisse Romande
DECEMBER 4 & 5, 2025 – CERN, GENEVA

camptocamp



Federico Sismondi

Infrastructure developer



- Platform Developer and Operator
- I really like Kubernetes
- I work hosting Odoo and Geospatial apps
- I garden and grow pumpkins in my free time



Julien Acroute

Infrastructure developer

- Manage Kubernetes Platform
- Automate Deployment
- Passionate about Observability
- Trainer for K8s, Docker, Terraform, PostgreSQL
- PostgreSQL/PostGIS enthusiast



Workshop Agenda



00 Warm up – OpenMetrics

01 Bootstrap – Kubernetes Cluster

02 Init – Tools

03 Story – Python based Particle Accelerator

04 Run – The Lab and the Metrics

05 Discover – Observability and EBPF

00 – OpenMetrics

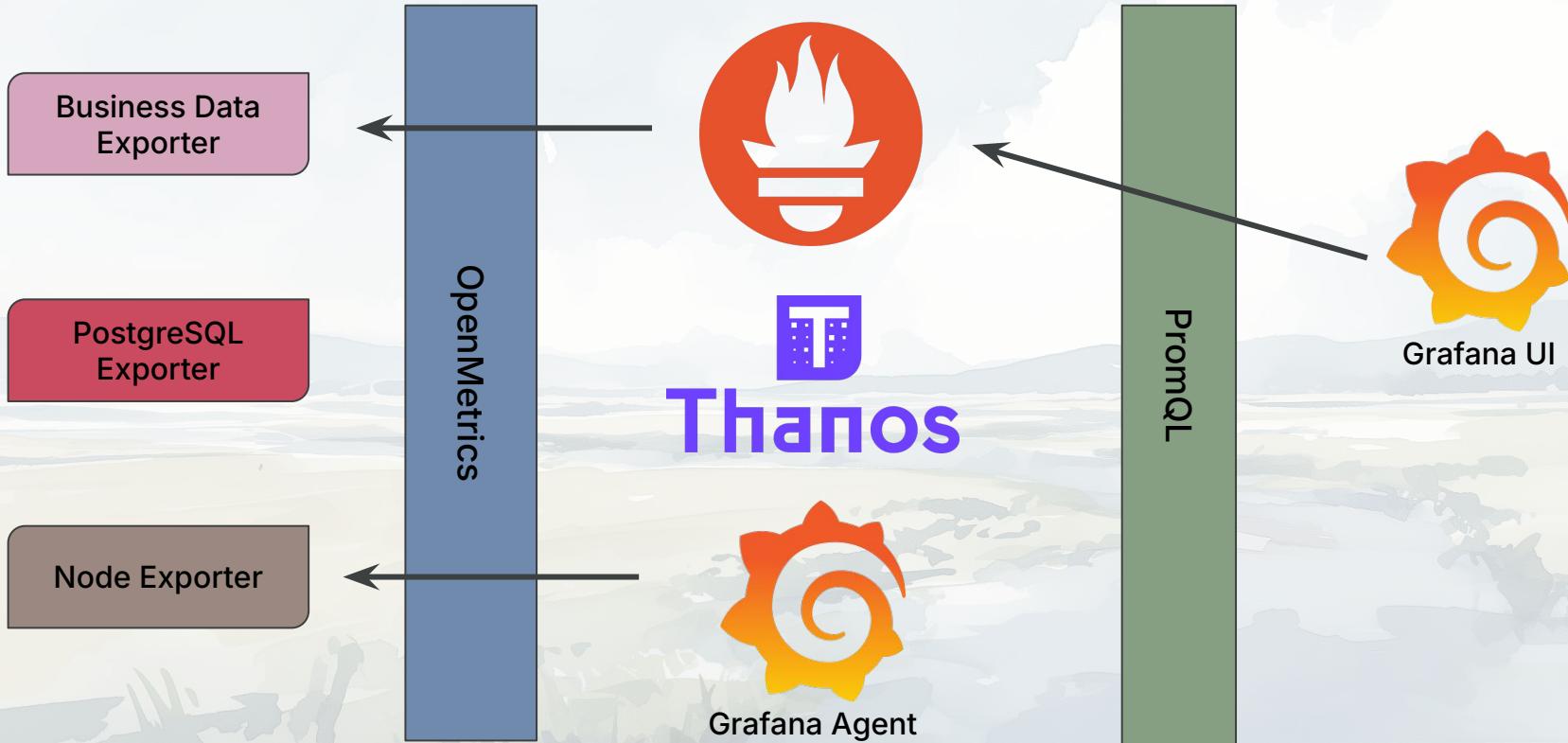


- A standard to expose numeric metrics
- Defined on openmetrics.io
- Extends Prometheus format
- Text based or Protocol Buffers
- CNCF Graduated project since 2018



OPEN METRICS

00 – OpenMetrics



00 – OpenMetrics



Exporter



Prometheus

```
# HELP disk_usage_percent Usage of disk in percent (0-100)
# TYPE disk_usage_percent gauge
disk_usage_percent{partition="/" 63.7
disk_usage_percent{partition="/var"} 37.2
disk_usage_percent{partition="/tmp"} 12.5
```

00 – OpenMetrics



Labels

- Label: metrics dimension
- A set of key/value pair
- Uniqueness: metric + labels + timestamp
- Shared between metrics

```
# HELP disk_usage_percent Usage of disk in percent (0-100)
# TYPE disk_usage_percent gauge
disk_usage_percent{partition="/" 63.7
disk_usage_percent{partition="/var"} 37.2
disk_usage_percent{partition="/tmp"} 12.5
```

```
# HELP disk_usage_percent Usage of disk in percent (0-100)
# TYPE disk_usage_percent gauge
disk_usage_percent{partition="/" 63.4
disk_usage_percent{partition="/var"} 37.6
disk_usage_percent{partition="/tmp"} 12.3

# HELP http_requests_total The total number of HTTP requests.
# TYPE http_requests_total counter
http_requests_total{method="GET", code="200"} 1234027
http_requests_total{method="POST", code="200"} 1027
http_requests_total{method="POST", code="400"} 3

# HELP sales_total The total number of sales.
# TYPE sales_total counter
sales_total{category="Tools", brand="Makita"} 163376
sales_total{category="Tools", brand="Bosh"} 298425
sales_total{category="Garden and Outdoor", brand="Weber"} 18346
sales_total{category="Garden and Outdoor", brand="Hyundai"} 163376

# HELP stock The number of stock items.
# TYPE stock gauge
stock{category="Tools", brand="Makita"} 234
stock{category="Tools", brand="Bosh"} 456
stock{category="Garden and Outdoor", brand="Weber"} 27
stock{category="Garden and Outdoor", brand="Hyundai"} 45
```

00 – OpenMetrics

Metrics Type



- HELP Description of the metrics
- TYPE Metrics type:
 - Gauge: can increase and decrease, instantaneous consumption, monthly bill
 - Counter: only increase, cumulative, electricity meter



```
# HELP disk_usage_percent Usage of disk in percent (0-100)
# TYPE disk_usage_percent gauge
disk_usage_percent{partition="/" 63.7
disk_usage_percent{partition="/var"} 37.2
disk_usage_percent{partition="/tmp"} 12.5
```

00 – OpenMetrics

Custom implementation



```
1 view_metric = {}
2
3 @app.route('/view/<id>')
4 def view_product(id):
5     # Update metric
6     if product not in view_metric:
7         view_metric[id] = 1
8     else:
9         view_metric[id] += 1
10    return "View %s" % id
```

```
1 @app.route('/metrics')
2 def metrics():
3     metrics = ""
4     for id in view_metric:
5         metrics += 'view_total{product="%s"} %s\n' %
6                     (id, view_metric[id])
7     for id in buy_metric:
8         metrics += 'buy_total{product="%s"} %s\n' %
9                     (id, buy_metric[id])
10    return metrics
```

```
view_total{product="p1"} 17
view_total{product="p2"} 25
buy_total{product="p1"} 5
buy_total{product="p2"} 12
```

00 – OpenMetrics

Prometheus Libraries



- Custom implementation
- Libraries: Python, Go, Java, Ruby, Rust, ...

```
1 from prometheus_client import Counter
2
3 view_metric = Counter('view', 'Product view', ['product'])
4
5 @app.route('/view/<id>')
6 def view_product(id):
7     # Update metric
8     view_metric.labels(product=id).inc()
9     return "View %s" % id
```

```
1 from prometheus_client import generate_latest
2
3 @app.route('/metrics')
4 def metrics():
5     return generate_latest()
```

00 – OpenMetrics

Helpers



- Exception counter
- Track duration

```
1 import time
2 import random
3 from prometheus_client import Summary
4
5 duration = Summary('duration_compute_seconds', 'Time spent in compute()')
6
7 @duration.time()
8 def compute():
9     time.sleep(random.uniform(0, 10))
```

00 – OpenMetrics

On-demand Metrics



- Use callback

```
1 from prometheus_client import Gauge
2
3 stock_metric = Gauge('stock', 'Stock count')
4 stock_metric.set_function(compute_stock)
5
6 def compute_stock():
7     res = query('SELECT count(*) FROM product_stock')
8     for line in res:
9         return line[0]
```

→01

01 – Create the Kubernetes Cluster

Get the Welcome card



<https://ze2zz7yjlt.ks.private.cloud.swisscom.ch/>

The welcome card contains:

- URL to the Swisscom Web UI:
- Username
- Password

The welcome page contains:

- Lab Instructions
- Commands for copy/paste
- <https://lab.campto.camp/>

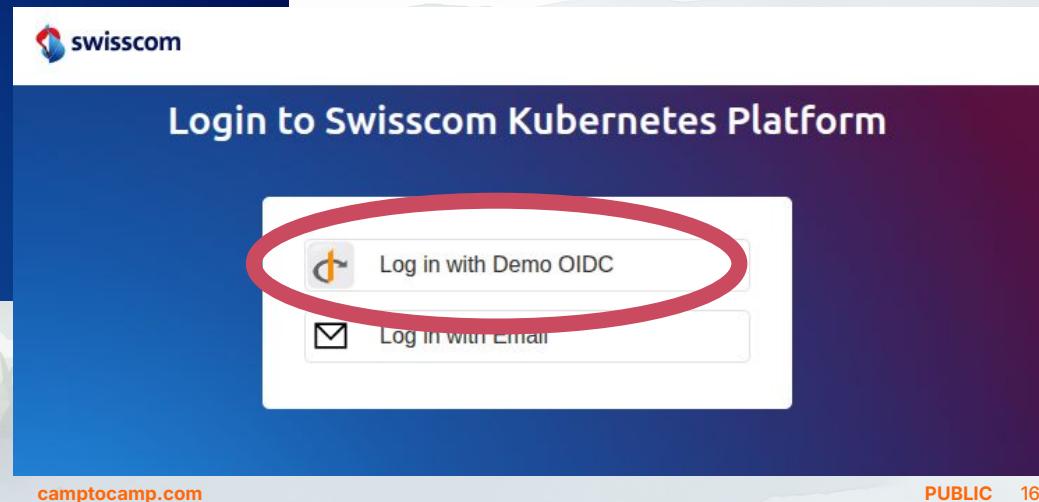
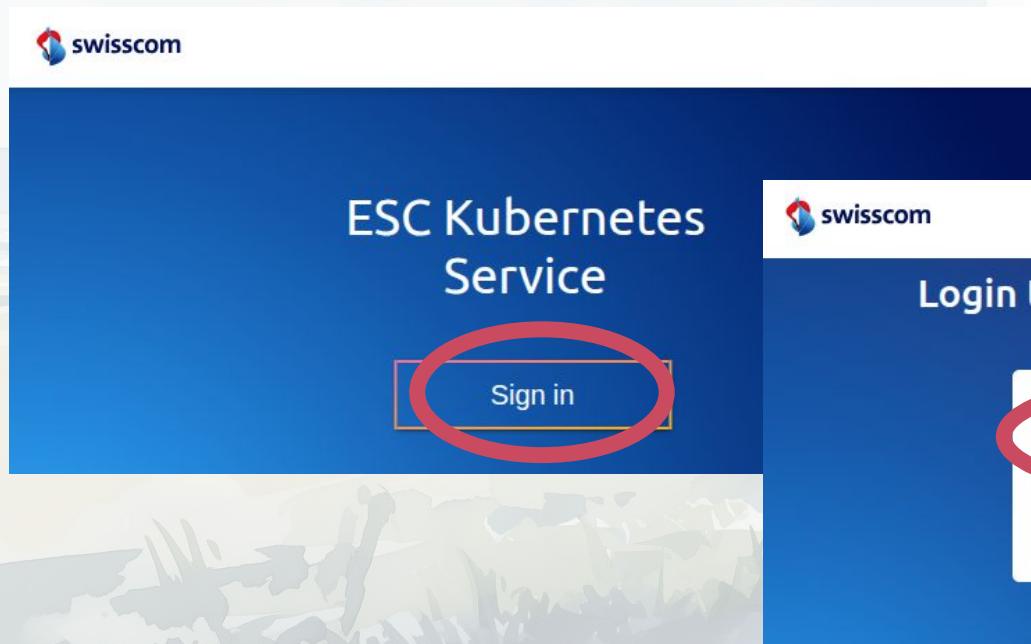


01 – Create the Kubernetes Cluster



Sign in

Using URL from the welcome card: <https://xxxx.ks.private.cloud.swisscom.ch/>



01 – Create the Kubernetes Cluster

Choose project



The screenshot shows a dark-themed user interface for managing projects. At the top, there is a header with the swisscom logo, a 'Projects' tab, and three icons: a gear, a bell, and a person. Below the header is a search bar with the placeholder 'Search'. To the right of the search bar are three buttons: a grid icon, a list icon, and a '+ Add Project' button.

The main area displays a list of projects. One project is highlighted with a green dot: 'camptocamp-01'. The details for this project are shown in a modal window:

- ID:** [REDACTED]
- Nodes:** 1
- Pods:** 0
- CI/CD:** +1
- Owner:** [REDACTED]
- Resources:** CPU, Memory, Disk

01 – Create the Kubernetes Cluster



Create a new cluster

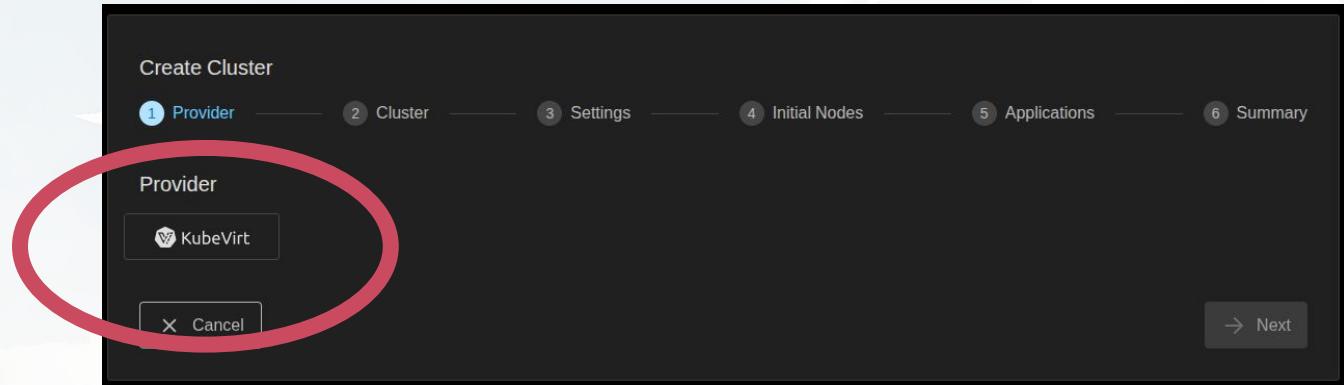
- Create Resource
 - Cluster

The screenshot shows the Camptocamp web interface for managing Kubernetes clusters. At the top, it displays the project name "campocamp-02". On the left, there's a sidebar with various icons for navigation. The main area is titled "Project Overview" and shows statistics for the current project: 0 Clusters, 0 External Clusters, 0 Cluster Templates, 0 Backups, and 0 SS. Below this, there are two main sections: "Clusters" and "Project Quota". The "Clusters" section indicates "No clusters available." and has a blue button labeled "Show All Clusters". The "Project Quota" section shows CPU usage at 0/16 and Memory usage. To the right of these sections is a vertical sidebar with options: "Cluster Template", "Automatic Backup", and "Snapshot". In the top right corner of the main area, there is a "Create Resource" button with a pencil icon, which is circled in red.

01 – Create the Kubernetes Cluster

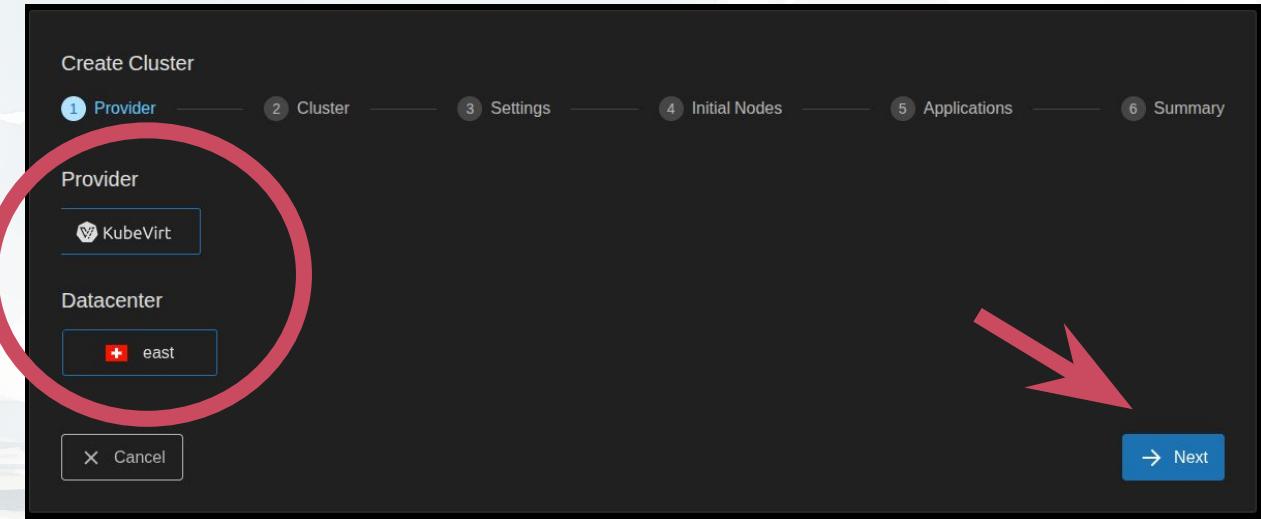
Provider

- Select Provider



01 – Create the Kubernetes Cluster Provider

- Select Provider
- Select Datacenter



01 – Create the Kubernetes Cluster

Cluster



- Cluster name

Create Cluster

Provider: ✓ Cluster: 2 Settings: 3 Initial Nodes: 4 Applications: 5 Summary: 6

Cluster

Name*

Network Configuration

cilium canal None

CNI Plugin Version: 1.18.2

Specification

Control Plane Version*: 1.33.5 Container Runtime*: containerd

Admission Plugins

Audit Logging custom metadata minimal recommended

Cluster Backup Disable CSI Driver

Annotations

Key Value

Back Next

01 – Create the Kubernetes Cluster

Settings

- Select settings

Create Cluster

Provider Cluster Settings Initial Nodes Applications Summary

Basic Settings

Provider Preset* swisscom-east

Kubeconfig*

No VPCs Available

Please enter your credentials first.

Cancel Next

01 – Create the Kubernetes Cluster

Initial Nodes



- Deploy only one node

Create Cluster

Provider Cluster Settings **Initial Nodes** Applications Summary

Basic Settings

Leave this field blank to use automatic name generation.
Replicas*

1

Upgrade system on first boot

Operating System Profile
swisscom-ubuntu-22.04

Leave this field blank to use default operating system profile.

CPU 4/16 Memory 16/32 GB Disk 20/500 GB

x

01 – Create the Kubernetes Cluster

Initial Nodes



- Deploy only one node
- Increase CPU to 4
- Increase Memory to 16 GB
- Increase Disk size to 20 GB

KubeVirt Settings

No Instance Types Available No Preferences Available Please select an instance type first.

View

CPUs* 4

Memory (MB)* 16000

Subnet*

Use specific subnet for VMs. VPC needs to be selected first to use this option.

Primary Disk

Ubuntu System Image* Ubuntu 22.04 - <https://bin.ze2zz7yjt.ks.private.cloud.swisscom.ch/ubuntu-jammy-kkp/ubuntu-jammy-kkp-1.2.0-20251021.img>

Storage Class* swarm-standard-olt

Size (GB)* 20

camptocamp.com

01 – Create the Kubernetes Cluster

Initial Nodes

- Select any subnet
- Select a Primary disk

The screenshot shows a configuration interface for creating a Kubernetes cluster. A red circle highlights the 'Subnet' field, which contains the value 'draco-491-zvs1k1-qh2oxu-olt-01654f (10.50.5.0/24)'. Below it, a note says 'Use specific subnet for the machines. VPC needs to be selected first to use this option.' Another red circle highlights the 'Ubuntu System Image' field, which lists 'Ubuntu 22.04 - https://bin.ze2zz7yjlt.ks.private.cloud.swisscom.ch/ubuntu-jammy-kkp/ubuntu-jammy-kkp-1.2.0-20251021.img'. A large red arrow points from the bottom right towards the 'Next' button at the bottom right of the screen.

CPUs*
4

Memory (MB)*
16000

Subnet*
draco-491-zvs1k1-qh2oxu-olt-01654f (10.50.5.0/24)
Use specific subnet for the machines. VPC needs to be selected first to use this option.

Primary Disk

Ubuntu System Image*
Ubuntu 22.04 - https://bin.ze2zz7yjlt.ks.private.cloud.swisscom.ch/ubuntu-jammy-kkp/ubuntu-jammy-kkp-1.2.0-20251021.img

Storage Class*
san-infra-standard-olt

Size (GB)*
20

ADVANCED SCHEDULING SETTINGS ▾

Cancel

Back

Next

01 – Create the Kubernetes Cluster

Applications



- Add Nginx

Create Cluster

Provider Cluster Settings Initial Nodes Applications Summary

Applications to Install

No application selected to install on cluster creation, [learn more about Applications](#).

+ Add Application

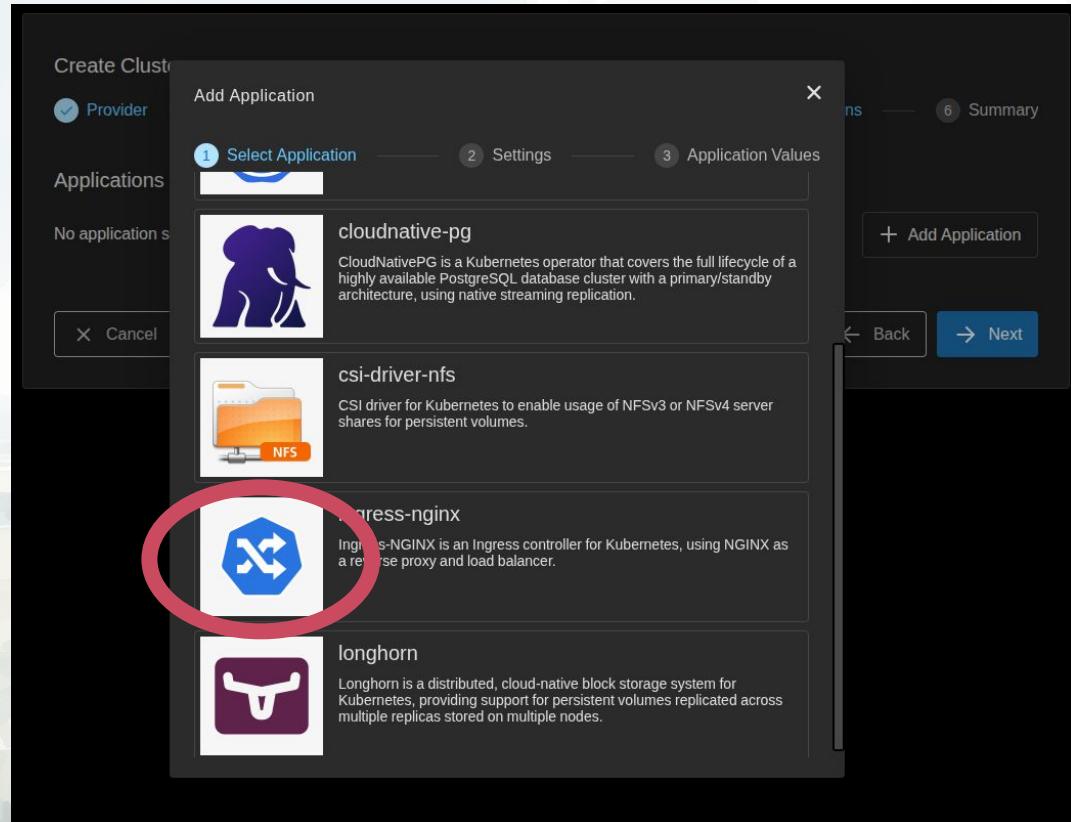
Cancel Back Next

The screenshot shows a dark-themed UI for creating a Kubernetes cluster. At the top, a progress bar indicates six steps: Provider (done), Cluster (done), Settings (done), Initial Nodes (done), Applications (selected), and Summary (not yet reached). The main area is titled 'Applications to Install' and contains a message stating 'No application selected to install on cluster creation'. Below this is a large red-circled button labeled '+ Add Application'. At the bottom, there are 'Cancel', 'Back', and 'Next' buttons.

01 – Create the Kubernetes Cluster

Applications

- Add Nginx



01 – Create the Kubernetes Cluster

Applications

- Add Nginx

The screenshot shows a dark-themed user interface for creating a Kubernetes cluster. On the left, a sidebar titled 'Create Cluster' has a 'Provider' section selected, showing 'Applications' and 'No application s'. A central modal window is open, titled 'Add Application', with a progress bar at the top showing steps 1 (Select Application), 2 (Settings), and 3 (Application Values). Step 1 is completed, indicated by a blue checkmark. The main content area displays the 'ingress-nginx' application details. It includes a logo of two arrows inside a hexagon, a brief description stating 'Ingress-NGINX is an Ingress controller for Kubernetes, using NGINX as a reverse proxy and load balancer.', a 'Version*' dropdown set to '4.13.3', and two 'Method' and 'Source' buttons, both of which have 'HELM' icons above them. Below these are fields for 'Application Installation Namespace*' (set to 'ingress-nginx') and 'Name*' (set to 'ingress-nginx'). Further down are fields for 'Application Resources Namespace*' (set to 'ingress-nginx') and a note about deploying resources. At the bottom of the modal are 'Back' and 'Next' buttons, with the 'Next' button being highlighted with a red circle.

01 – Create the Kubernetes Cluster

Applications



- Add Nginx

Create Cluster

Provider

Add Application

Applications

No application s

Cancel

Optional: Use custom values to override the default configuration for this application.

values.yaml

```
1 global:
2   image:
3     registry: registry.ze2zz7yjlt.ks.private.cloud.swisscom.ch
4 controller:
5   metrics:
6     enabled: true
7   service:
8     type: LoadBalancer
9     loadBalancerClass: kubelb
10    annotations:
11      prometheus.io/scrape: "true"
12      prometheus.io/port: "10254"
13    ingressClassResource:
14      default: true
15    replicaCount: 2
16
```

Input should be valid YAML

Back Next

+ Add Application

ns 6 Summary

+ Add Application

Back Next

01 – Create the Kubernetes Cluster

Applications



- Add Nginx
- Add Openmetrics
Particle
Accelerator Lab

Create Cluster

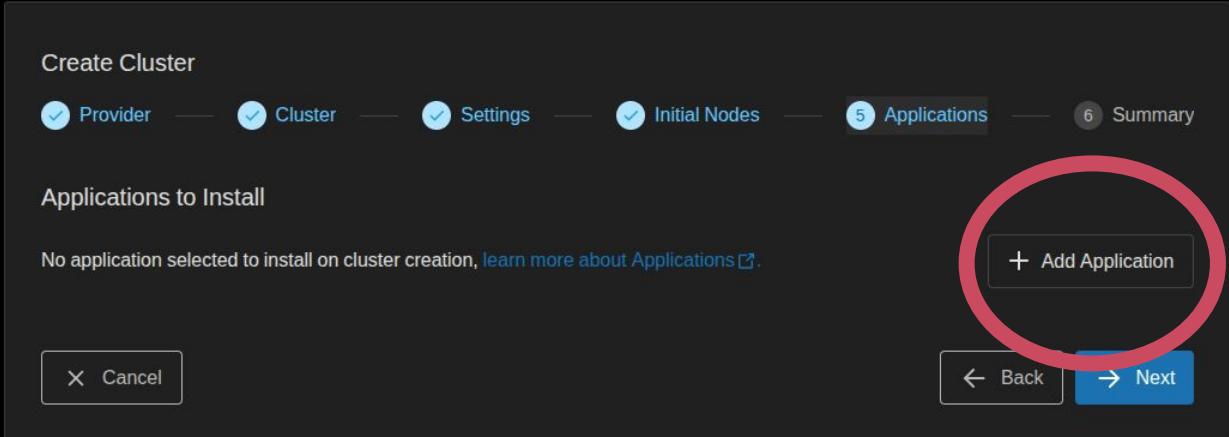
Provider Cluster Settings Initial Nodes Applications Summary

Applications to Install

No application selected to install on cluster creation, [learn more about Applications](#).

+ Add Application

Cancel Back Next



01 – Create the Kubernetes Cluster

Applications



- Add Nginx
- Add Openmetrics
Particle
Accelerator Lab

The screenshot shows a 'Create Cluster' interface with the 'Add Application' step selected. The process is divided into six steps: 1. Select Application, 2. Settings, 3. Application Values, 4. Summary, 5. Create Cluster, and 6. Summary. Step 1 is completed, indicated by a checked blue circle. The 'Select Application' screen displays three options:

- NFS**: An icon of a screwdriver and wrench.
- ingress-nginx**: An icon of a blue hexagon with white arrows. Description: Ingress-NGINX is an Ingress controller for Kubernetes, using NGINX as a reverse proxy and load balancer.
- longhorn**: An icon of a white bull's head on a purple background. Description: Longhorn is a distributed, cloud-native block storage system for Kubernetes, providing support for persistent volumes replicated across multiple replicas stored on multiple nodes.
- Openmetrics Particle Accelerator Lab**: An icon of a blue hexagon with a white circle and a red cross. Description: This project is a small experimental lab designed to demonstrate how easily OpenMetrics metrics can be integrated into a business application for Prometheus monitoring.

A large red oval highlights the 'Openmetrics Particle Accelerator Lab' entry. Navigation buttons include 'Cancel', 'Back', and 'Next'.

01 – Create the Kubernetes Cluster

Applications



- Add Nginx
- Add Openmetrics
Particle
Accelerator Lab

Add Application

Create Cluster

Provider

Applications

No application selected

Cancel

Select Application

Settings

Application Values

Openmetrics Particle Accelerator Lab

This project is a small experimental lab designed to demonstrate how easily OpenMetrics metrics can be integrated into a business application for Prometheus monitoring.

Documentation - </> Source

Version*

0.1.0

Method Source

Application Installation Namespace*

openmetrics-accelerator-lab

Namespace where application installation will be created.

Name*

openmetrics-accelerator-lab

Application Resources Namespace*

openmetrics-accelerator-lab

Back

Next

camptocamp.com

01 – Create the Kubernetes Cluster Applications



- Add Nginx
- Add Openmetrics
Particle
Accelerator Lab

The screenshot shows a 'swisscom' application interface for creating a Kubernetes cluster. The main title is 'Projects campocamp-01'. A central modal window is titled 'Add Application' and is on step 3: 'Application Values'. It shows a 'values.yaml' file with the content:

```
1 # empty
2
```

An error message at the bottom left of the modal says 'Input should be valid YAML'. At the bottom right of the modal is a blue 'Add Application' button. To the right of the modal, there's a sidebar with tabs like 'Overview', 'Logs', 'Events', 'Metrics', 'Summary', and 'Logs'. Below the sidebar are buttons for '+ Add Application', 'Back', and 'Next'.

01 – Create the Kubernetes Cluster

Applications



- Add Nginx
- Add Openmetrics
Particle
Accelerator Lab

Create Cluster

Provider Cluster Settings Initial Nodes Applications Summary

Applications to Install

+ Add Application

Search

Application	Version	Chart Version	Source
ingress-nginx	4.13.3	4.13.3	ingress-nginx
Openmetrics Particle Accelerator Lab	0.1.0	0.1.0	openmetrics-accelerator-lab

Cancel Back Next



01 – Create the Kubernetes Cluster

Summary



Create Cluster

Provider Cluster Settings Initial Nodes Applications Summary

① Provider
Provider: KubeVirt Datacenter: east (CH)

② Cluster

④ Initial Nodes
GENERAL
Name: Autogenerated name Replicas: 1 Operating System: Ubuntu
Operating System Profile: swisscom-ubuntu-22.04 Node Annotations:

③ Settings
Preset: swisscom-east

⑤ Applications
ingress-nginx
Deployment icon Helm icon Version: 4.13.3
↳ ingress-nginx



Cancel Back Save Cluster Template Create Cluster

01 – Create the Kubernetes Cluster

Get Kubeconfig



- Download Kubeconfig
- Remember the location of the file for next steps

The screenshot shows the KubeSphere interface for managing clusters. At the top, there's a cluster summary for 'particle-accelerator':

- Control Plane: 1.33.5
- CNI: cilium 1.18.2
- Region: CH (east)
- Provider: KubeVirt
- Preset: swisscom-east
- Container Runtime: containerd
- SSH Keys: No assigned keys

Below this, under 'ADDITIONAL CLUSTER INFORMATION ^', are several configuration details:

Control Plane	Networking	OPA
API Server	Proxy Mode ebpf	Policy Control
etcd	Expose Strategy Tunneling	
Scheduler	Pods CIDR 172.26.0.0/16	
Controller	Services CIDR 10.241.0.0/20	
Machine Controller	Node CIDR Mask Size 24	
Operating System Manager		
User Controller Manager		
Kubernetes Dashboard	✓ Node Local DNS Cache	
Kubermatic KubeLB	✓ Konnectivity	

01 – Create the Kubernetes Cluster

Summary

- Sign-in using OIDC
- Create Resource → Cluster
- 1. Provider: Choose default provider and datacenter
- 2. Cluster: Set cluster name
- 3. Settings: Choose default settings
- 4. Initial Nodes: One node with 4 CPUs, 16 GB of RAM and 20 GB of disk space
- 5. Applications: Deploy Nginx and the Lab

instructions on
lab.campto.camp



→02

02 – Install tools

kubectl



- Cluster nodes are not reachable



Kubernetes Nodes



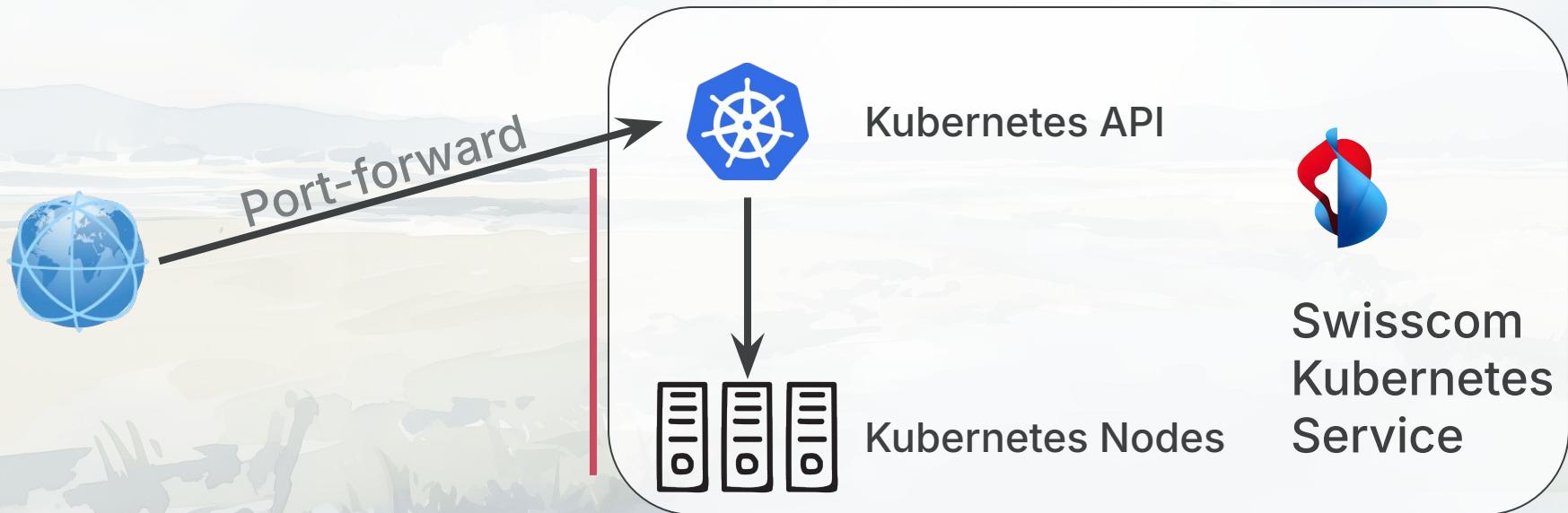
Swisscom
Kubernetes
Service

02 – Install tools

kubectl



- Cluster nodes are not reachable
- Port forward to an ingress controller pod



02 – Install tools

kubectl



- Follow [official instructions](#)
- Phase 1 – "Install Kubectl" on [lab.campto.camp](#)
- API version v1.33.5
- Kubernetes version skew policy -1/+1:
 - 1.32
 - 1.33
 - 1.34 (latest)

02 – Install tools

kubectl



- Linux:

```
curl -LO "https://dl.k8s.io/release/$(curl -L -s https://dl.k8s.io/release/stable.txt)/bin/linux/amd64/kubectl"
```

- macOS:

```
brew install kubectl
```

- Windows:

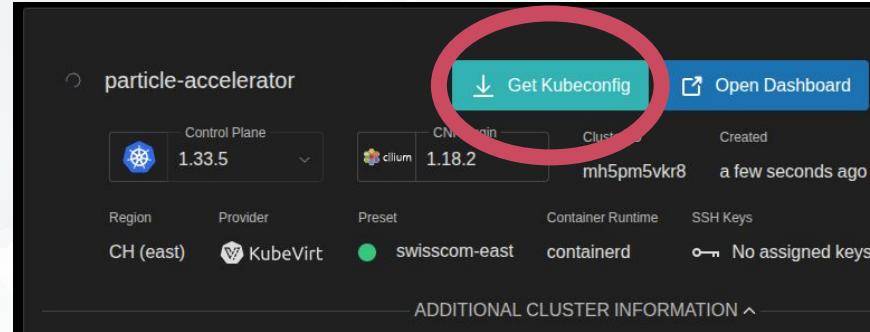
```
curl.exe -LO "https://dl.k8s.io/release/v1.34.2/bin/windows/amd64/kubectl.exe"
```

```
$ kubectl version --client  
Client Version: v1.34.2
```

02 – Install tools

Configure API Access

- Locate `kubeconfig-xxxxxxxxxxxx`
- Setup path to the `kubeconfig` file



```
# Linux/Mac
```

```
export KUBECONFIG=~/Downloads/kubeconfig-xxxxxxxxxxxx
```

```
# Windows PowerShell
```

```
$env:KUBECONFIG="C:\Users\YourUser\Downloads\kubeconfig-xxxxxx"
```

02 – Install tools

Validate Access



```
$ kubectl get pod -A
```

NAMESPACE	NAME	READY	STATUS	RESTARTS	AGE
ingress-nginx	ingress-nginx-controller-549886-cpg2d	1/1	Running	0	6h42m
ingress-nginx	ingress-nginx-controller-549886-p1bz5	1/1	Running	0	6h42m
kube-system	cilium-m7wf6	1/1	Running	0	10h
kube-system	cilium-operator-74668d995-hrpjh	1/1	Running	0	10h
kube-system	coredns-5d6489bb45-n4rrx	1/1	Running	0	10h
kube-system	coredns-5d6489bb45-v6mt2	1/1	Running	0	10h
kube-system	envoy-agent-jndgw	2/2	Running	0	10h
kube-system	hubble-relay-795945d857-xgqvw	1/1	Running	0	10h
kube-system	hubble-ui-556747b9c9-q6q29	2/2	Running	0	10h
kube-system	konnectivity-agent-7f7d9474dd-4pxjv	1/1	Running	0	10h
kube-system	konnectivity-agent-7f7d9474dd-gkp5s	1/1	Running	0	10h
kube-system	metrics-server-7bd6766f9d-5grc8	1/1	Running	0	10h

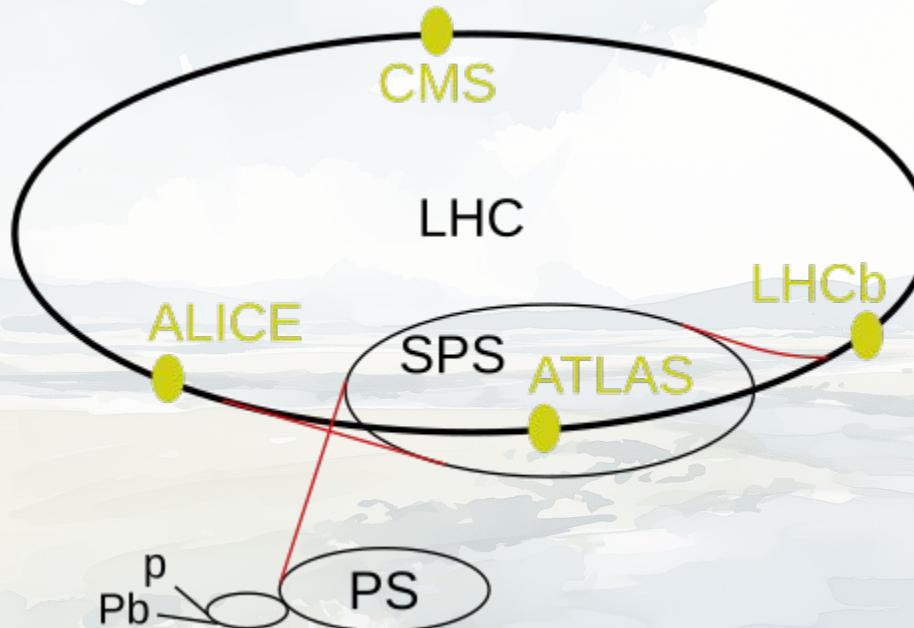
...

→03



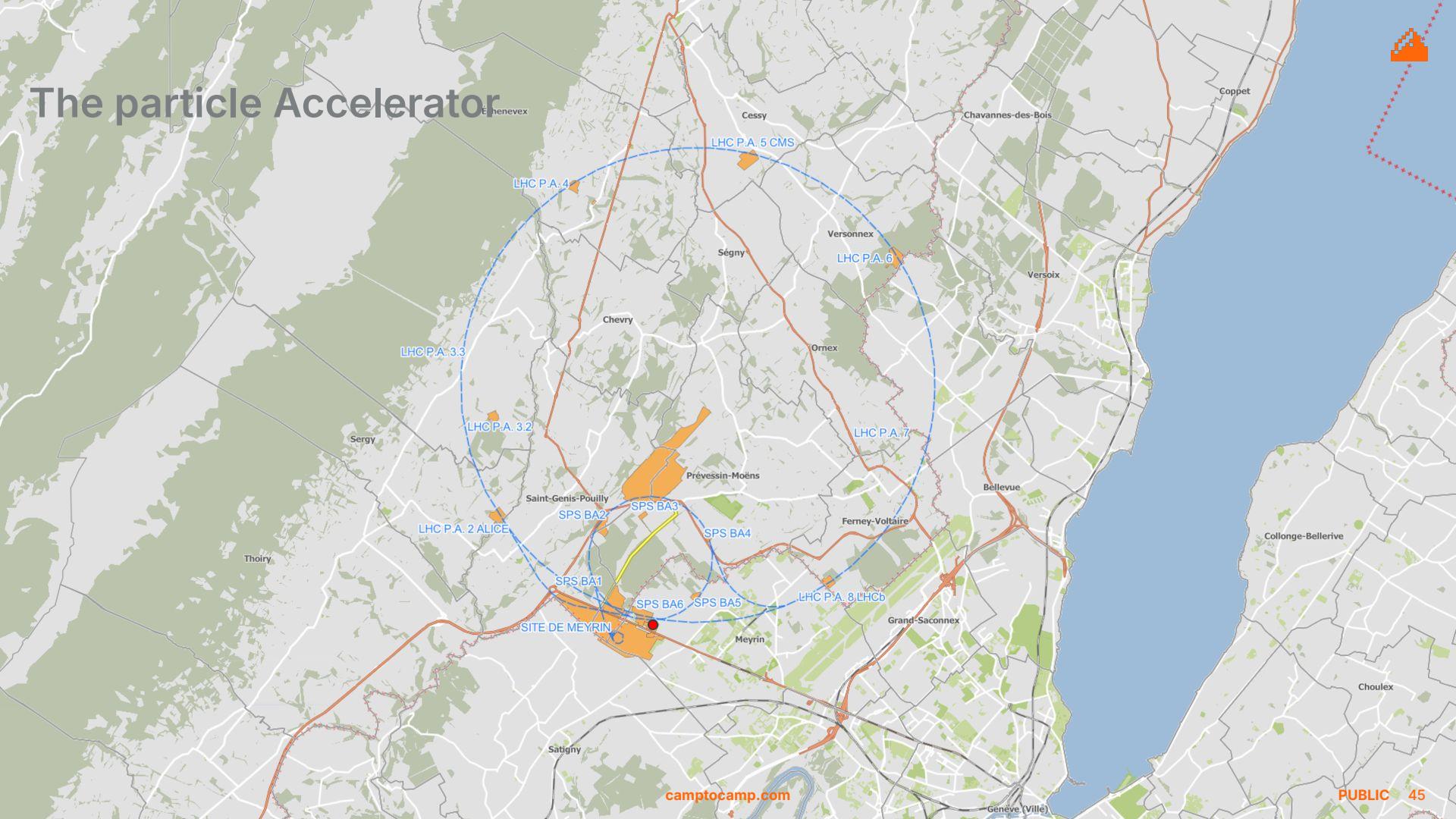
03 – Python based Particle Accelerator

The particle Accelerator





The particle Accelerator





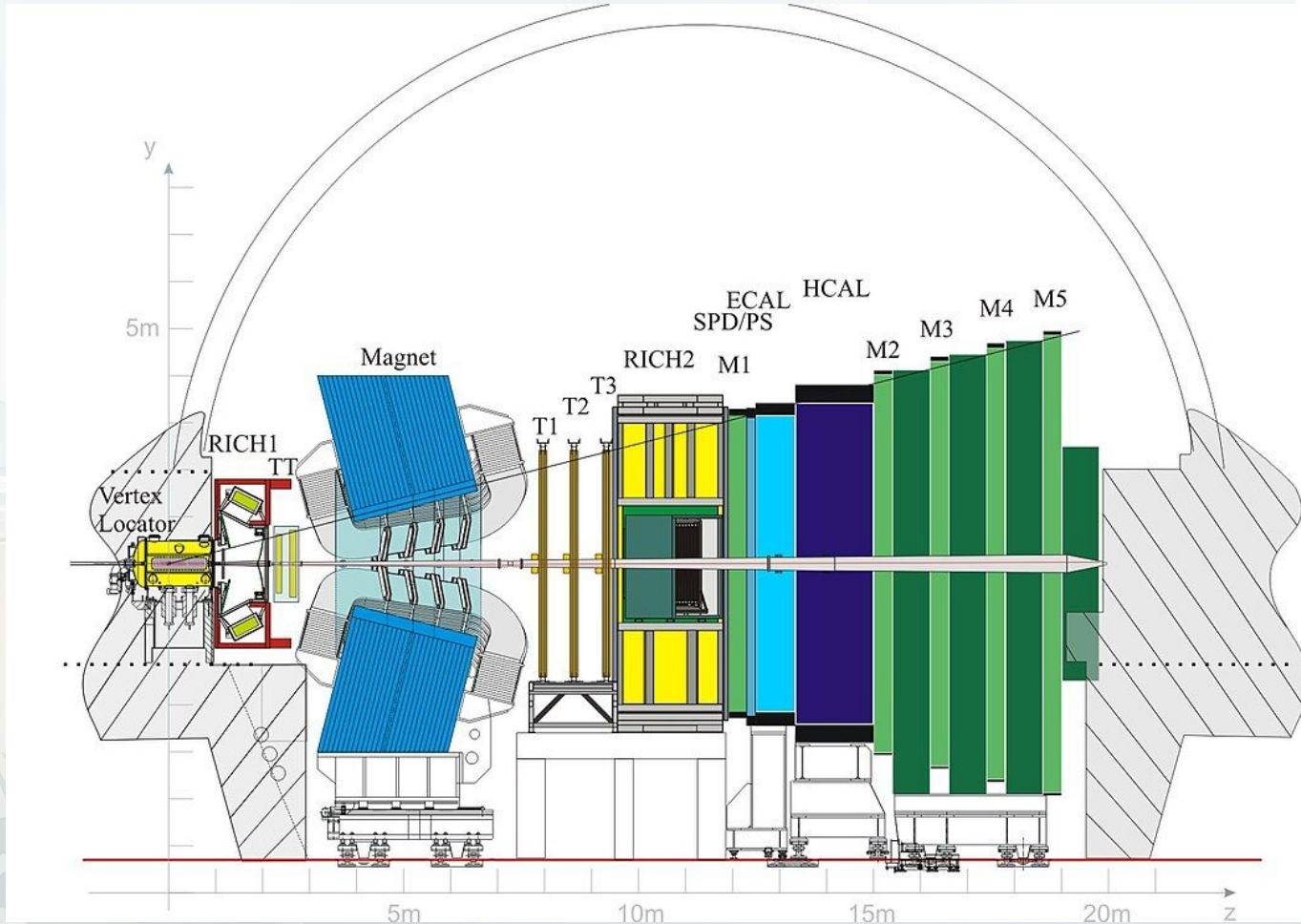
The particle Accelerator





The particle Accelerator

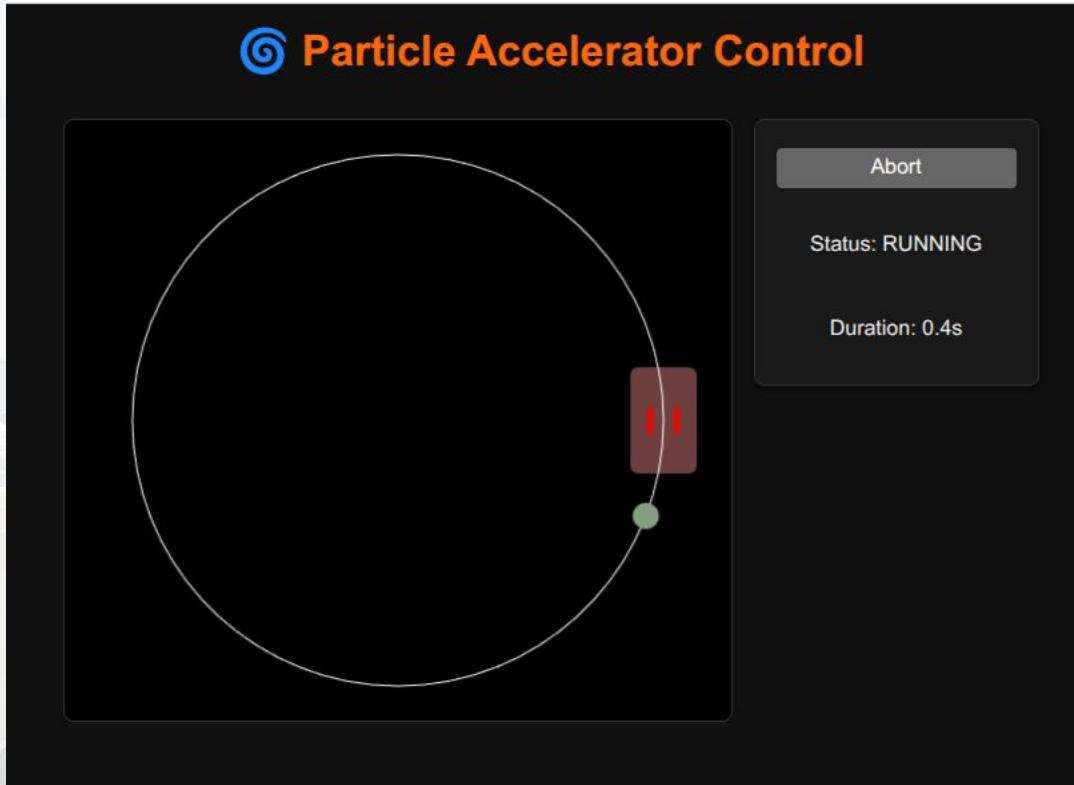






03 - Python based Particle Accelerator

The very simple python implementation



03 - Python based Particle Accelerator



- Goal:
 - Collide protons to find new sub particles
 - Accelerate protons faster
 - 0.99999999% speed of light (nine 9s)
- The "Kick":
 - Particles orbit the ring 11,000 times/second
 - RF cavity provides an electric "kick" to increase energy
- Accelerator challenge:
 - Too Weak: Beam won't reach collision energy (~6.8 TeV)
 - Too Strong: Superconducting magnets overheat

→04

04 – The Lab



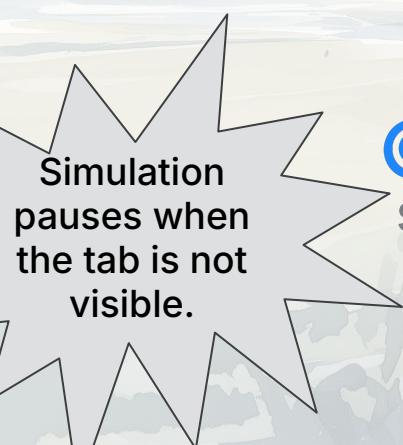
- The Stack:
 - Cluster: Ephemeral Kubernetes Environment
 - App: Python (Flask) simulating the accelerator physics
 - Observability: OpenMetrics (client library) → Prometheus → Grafana
- The Workflow:
 - Deploy: Launch the lab via the dashboard
 - Port Forward: Establish a secure tunnel to the cluster

04 – The Lab

The Feedback Loop



- Launch simulator.
- Watch Grafana Dashboards.
- Change Code in Browser (VSCode)
- Save (Auto-reload).
- React to the data.



Launch simulator



Watch dashboard



Auto reload



Edit
KICK_POWER

04 – The Lab

Establish Uplink (Port Forward)



- The Kubernetes cluster runs in an isolated network
- Single port forward to the ingress controller

```
$ kubectl -n ingress-nginx port-forward deploy/ingress-nginx-ingress-nginx-controller 8080:80
Forwarding from 127.0.0.1:8080 -> 80
Forwarding from [::1]:8080 -> 80
```

This exposes all workshop services to your local machine on port 8080.

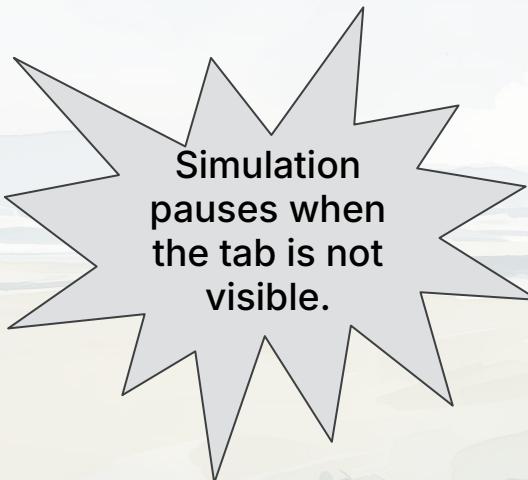
! If this command stops (Ctrl-C, terminal closed, laptop sleeps), you lose access to the lab.

04 – The Lab

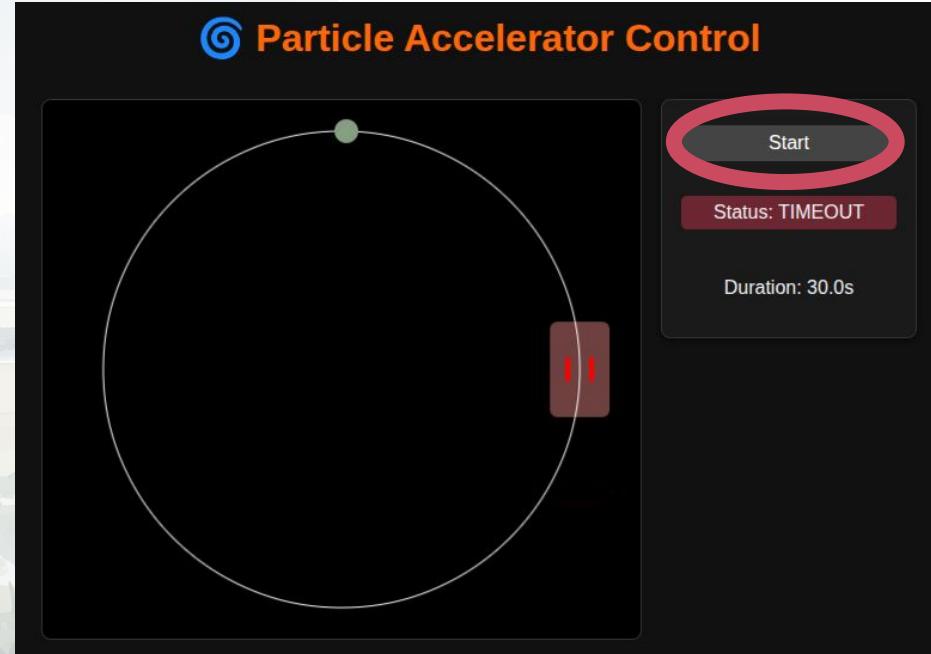
The Feedback Loop



- Launch simulator



<http://lab.127.0.0.1.nip.io:8080/>

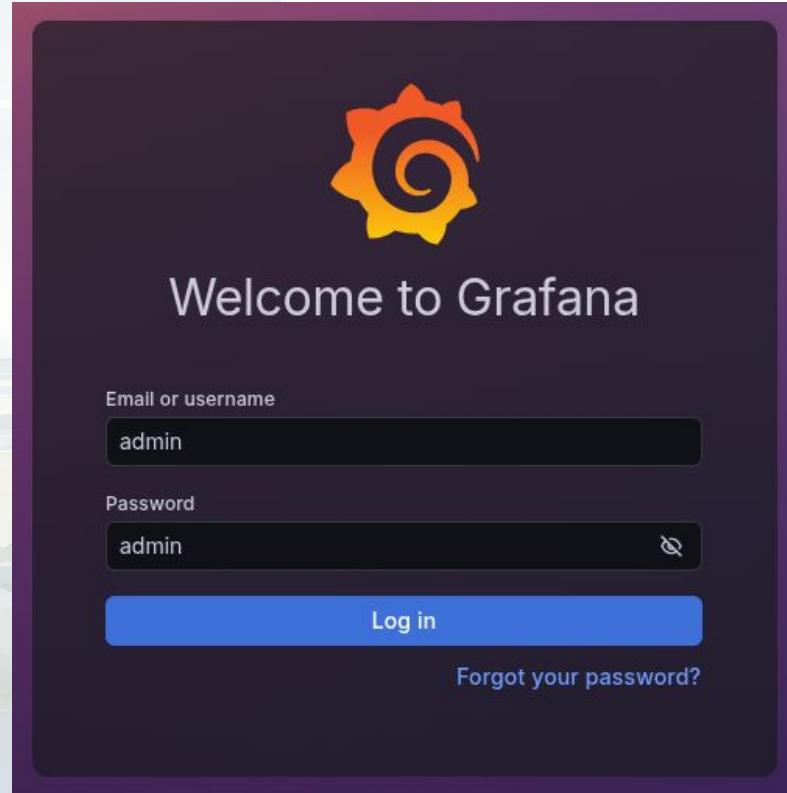


04 – The Lab

The Feedback Loop

- Launch simulator
- Watch Grafana

<http://grafana.127.0.0.1.nip.io:8080/>

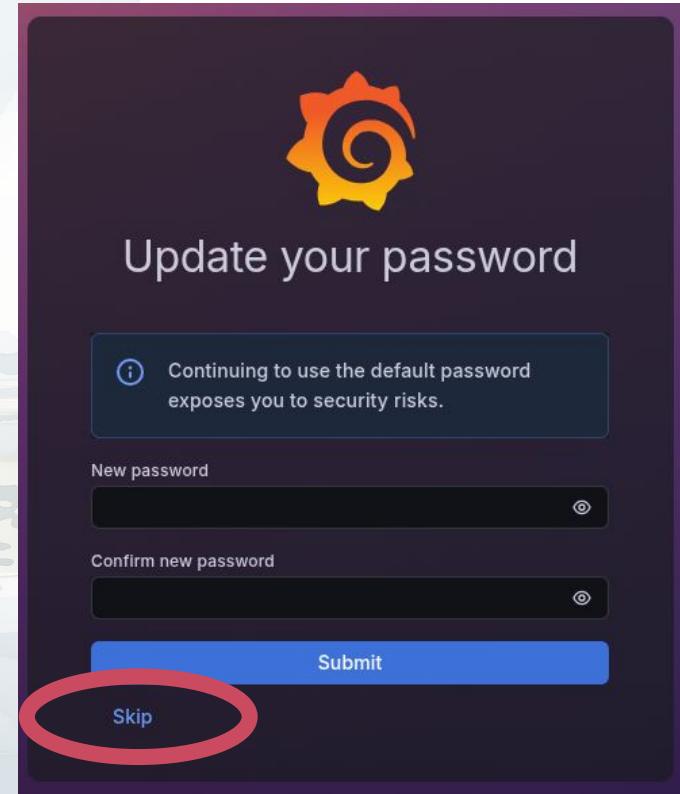


04 – The Lab

The Feedback Loop

- Launch simulator
- Watch Grafana

<http://grafana.127.0.0.1.nip.io:8080/>



04 – The Lab

The Feedback Loop



<http://grafana.127.0.0.1.nip.io:8080/>

- Launch simulator
- Watch Grafana

The screenshot shows the Grafana interface. On the left is a sidebar with the following menu items:

- Home
- Bookmarks
- Starred
- Dashboards** (highlighted with a red oval)
- Explore
- Drilldown
- Alerting
- Connections
- Administration

The main content area is titled "Dashboards" and contains the following text:
Create and manage dashboards to visualize your data

Search for dashboards and folders

Filter by tag

Starred

Name

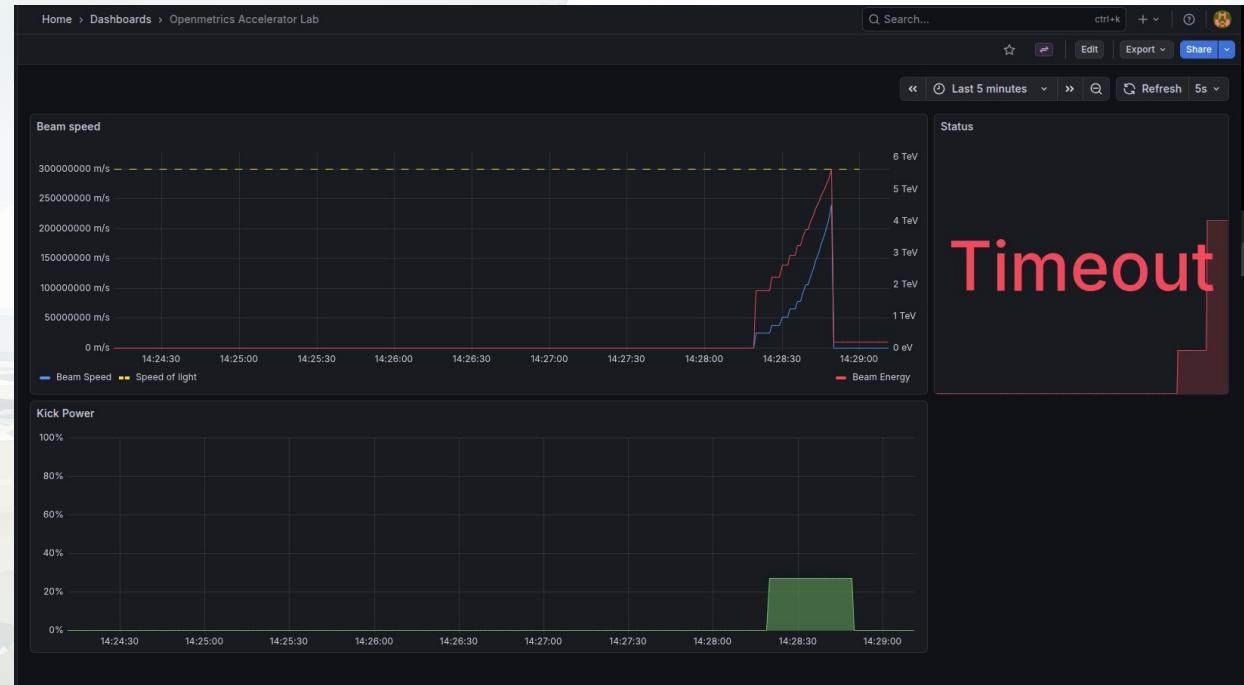
Openmetrics Accelerator Lab (highlighted with a red oval)

04 – The Lab

The Feedback Loop

- Launch simulator
- Watch Grafana

<http://grafana.127.0.0.1.nip.io:8080/>



04 – The Lab

The Feedback Loop



- Launch simulator
- Watch Grafana
- Change Code

<http://vscode.127.0.0.1.nip.io:8080/>

```
app.py
1 from flask import Flask, jsonify, render_template, request
2 from prometheus_client import Gauge, generate_latest, CONTENT_TYPE_LATEST
3 import math, time, threading
4
5 app = Flask(__name__)
6
7 # --- Prometheus metrics ---
8 kick_power = Gauge("kick_power", "Kick power")
9 beam_speed = Gauge("beam_speed_km_s", "Current beam speed in km/s")
10 beam_energy = Gauge("beam_energy_mev", "Current beam energy in MeV")
11 accelerator_state = Gauge("accelerator_state", "Accelerator state: 0=stoppe
12
13 # --- Internal state ---
14 state = 0
15 KICK_POWER = 27 # arbitrary unit, should be stay in [0;100]
16 c = 299_792_458 # speed of light
17 m0 = 0.000538 # proton energy in TeV
18
19 @app.route("/")
20 def index():
21     return render_template("index.html")
22
23 @app.route("/kick_power")
24 def kick():
25     global KICK_POWER
26     return jsonify({"kick_power": KICK_POWER})
27
```

04 – The Lab

Command Center



The particle view

<http://lab.127.0.0.1.nip.io:8080/>



The Data View (Grafana)

<http://grafana.127.0.0.1.nip.io:8080/>



The Code View (VSCode)

<http://vscode.127.0.0.1.nip.io:8080/>

04 – The Lab

Restart Lab



- Open Dashboard
- Select Deployments (left panel)
- Select 'openmetrics...' namespace (top drop list)
- Select 'openmetrics-accelerator-lab-x-xx-lab'
- Click on restart ↺

The screenshot shows the Kubernetes Dashboard interface for the 'particle-accelerator' cluster. At the top, there are dropdown menus for 'Control Plane' (set to 1.33.5), 'CNI Plugin' (set to cilium 1.18.2), and other cluster details like 'Region: CH (east)', 'Provider: KubeVirt', 'Preset: swisscom-east', and 'Container Runtime: containerd'. A red circle highlights the 'Open Dashboard' button in the top right. Below this, there's a section for 'ADDITIONAL CLUSTER INFORMATION' with tabs for 'Control Plane', 'Networking', and 'OPA'. Under 'Control Plane', several components are listed: API Server, etcd, Scheduler, Controller, Machine Controller, Operating System Manager, User Controller Manager, Kubernetes Dashboard (which is green), and Kubermatic KubeLB. Under 'Networking', it shows 'Proxy Mode: ebpf', 'Expose Strategy: Tunneling', 'Pods CIDR: 172.26.0.0/16', 'Services CIDR: 10.241.0.0/20', and 'Node CIDR Mask Size: 24'. Under 'OPA', there's a note about 'Policy Control'. At the bottom, there are two green checkmarks: 'Node Local DNS Cache' and 'Konnectivity'.

04 – The Lab

Application observability



- Open Dashboard
- Select Pod (left panel)
- Select 'openmetrics...' namespace (top drop list)
- Inject memory and cpu usage

The screenshot shows the KubeSphere UI for a cluster named 'particle-accelerator'. At the top, there are dropdown menus for Control Plane (1.33.5), CNI Plugin (cilium 1.18.2), Cluster ID (mh5pm5vkr8), and Seed (scs). Below these are fields for Region (CH (east)), Provider (KubeVirt), Preset (swisscom-east), Container Runtime (containerd), and SSH Keys (No assigned keys). A 'Get Kubeconfig' button is available. A prominent red circle highlights the 'Open Dashboard' button. To the right, there's a 'Web Terminal' button and a three-dot menu. The main area displays 'ADDITIONAL CLUSTER INFORMATION' with sections for Control Plane (API Server, etcd, Scheduler, Controller, Machine Controller, Operating System Manager, User Controller Manager) and Networking (Proxy Mode: ebpf, Expose Strategy: Tunneling, Pods CIDR: 172.26.0.0/16, Services CIDR: 10.241.0.0/20, Node CIDR Mask Size: 24). The 'Kubernetes Dashboard' is selected in the left sidebar. On the right, there are two green checkmarks: 'Node Local DNS Cache' and 'Konnectivity'.

04 – The Lab

Extra step – Implements a new metrics



- kick_count
- Type: counter
- Call inc() in the kick() function
- Check metrics on Prometheus

```
1 from prometheus_client import Counter  
2  
3 kick_count = Counter('kick_count', 'Kick Count')  
4  
5 @app.route('/kick_power')  
6 def kick():  
7     ...  
8     kick_count.inc()  
9     ...
```

→05

05 – Observability tools using EBPF

eBPF



- Small sandboxed program running in the Linux Kernel
- Custom programs injected at runtime in kernel
- No need to recompile or load modules
- Kernel feature started on 2011 with BPF: Berkeley Packet Filter
- 2014 Kernel 3.15 : Introduce eBPF support
- Many evolutions with Kernel 4, 5 and 6



05 – Observability tools using eBPF

eBPF

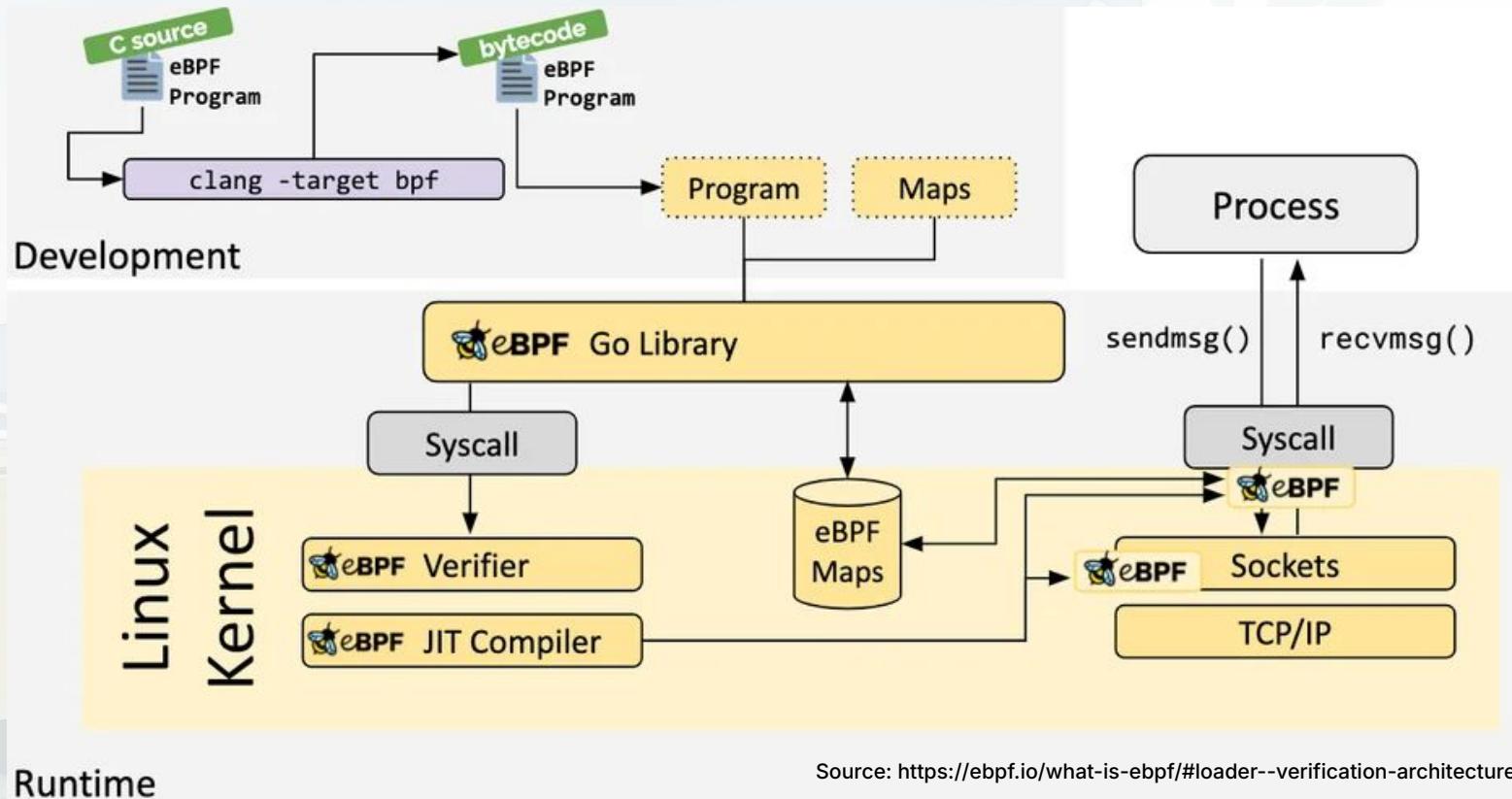


- Focus on stability and security
- User writes eBPF program with C language
- The verifier checks that:
 - user is allowed to inject eBPF programs
 - the program always run to completion
- Maps are used to exchange data between Kernel and user-space



05 – Observability tools using eBPF

eBPF



05 – Observability tools using EBPF



- BPTrace : <https://github.com/bpftrace/bpftrace/tree/master/tools>
- Inspektor Gadget <https://inspektor-gadget.io/docs/latest/gadgets/>

```
$ kubectl debug --profile=sysadmin $(kubectl get node -o name) -ti \  
  --image=ghcr.io/inspektor-gadget/ig:latest -- \  
  ig run trace_exec:latest \  
  --filter k8s.namespace==openmetrics-accelerator-lab
```



Federico Sismondi

federico.sismondi@camptocamp.com

Julien Acroute

julien.acroute@camptocamp.com

Thanks for your attention.