



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

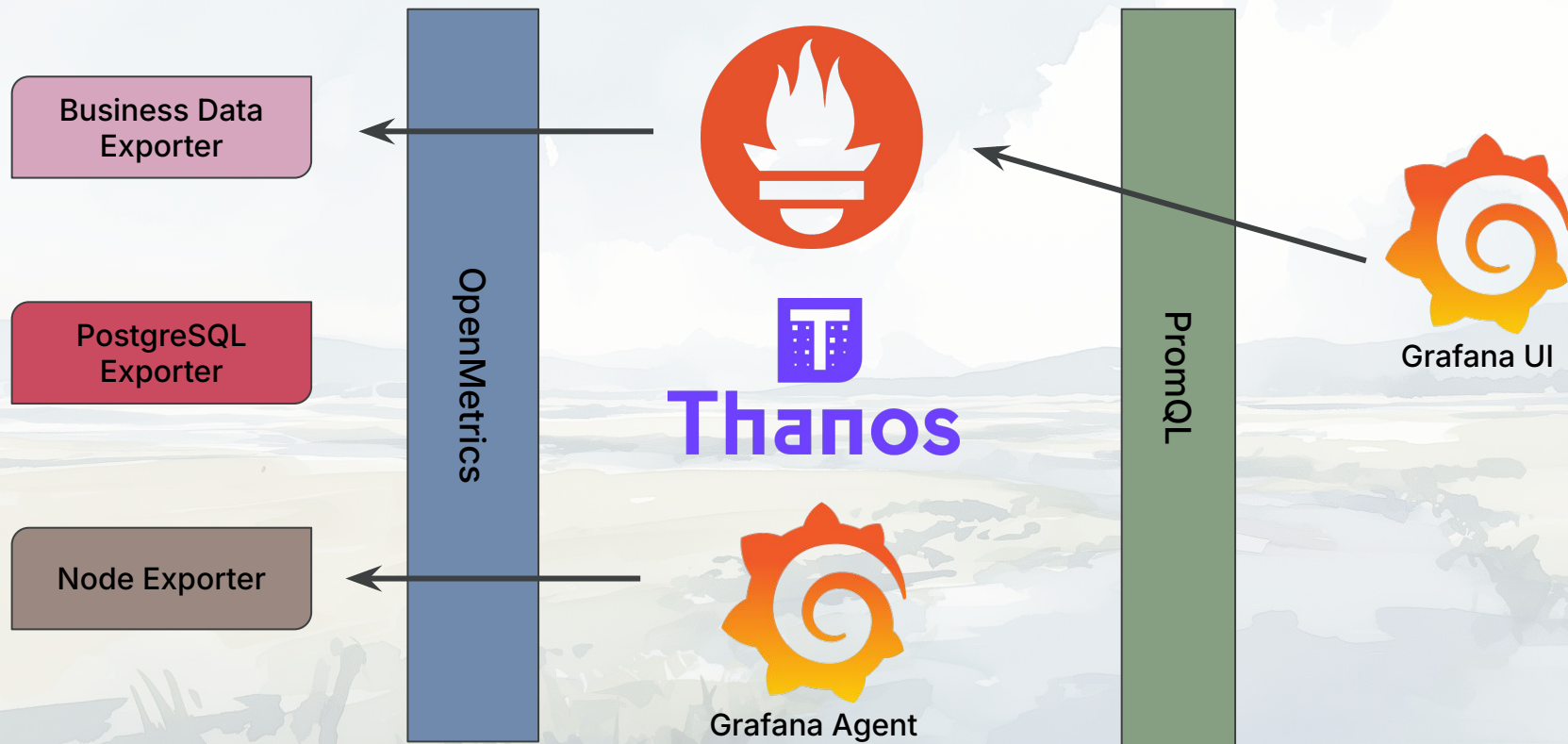


- A standard to expose numeric metrics
- Defined on openmetrics.io
- Extends Prometheus format
- Text based or Protocol Buffers
- CNCF Graduated project since 2018

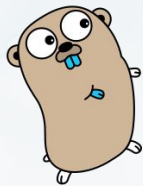


OPEN METRICS

00 – OpenMetrics



00 – OpenMetrics



Exporter

HTTP GET

/metrics



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
8 @duration.time()
9 def compute():
10     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/>

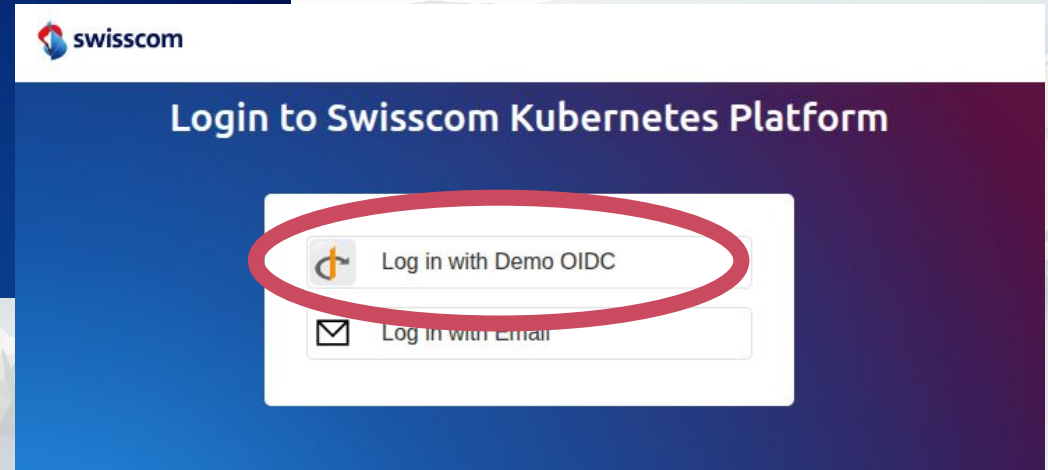
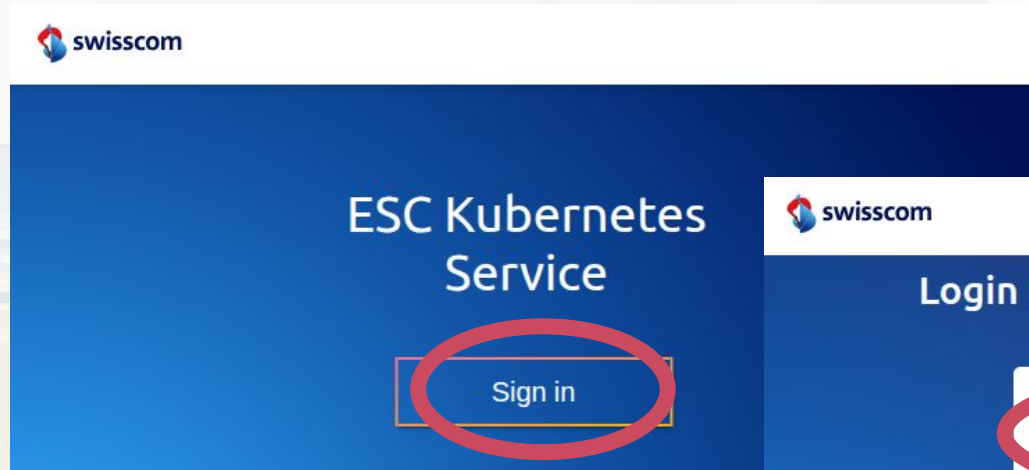


01 – Create the Kubernetes Cluster

Sign in



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



01 – Create the Kubernetes Cluster

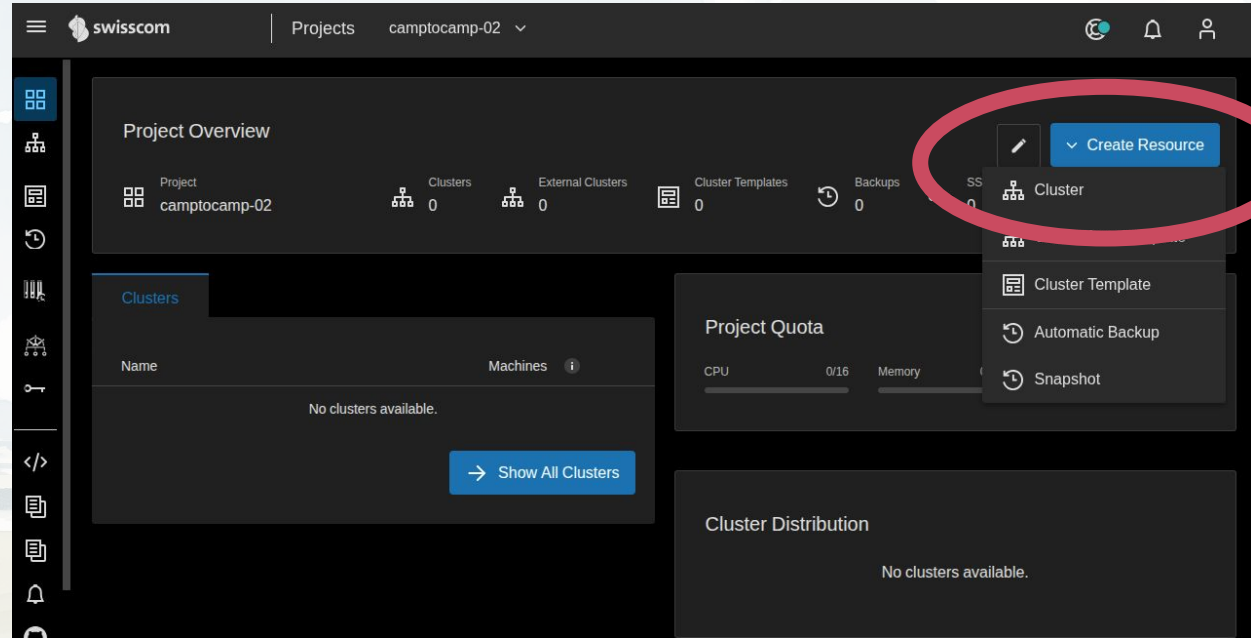
Choose project



01 – Create the Kubernetes Cluster

Create a new cluster

- Create Resource
 - Cluster



01 – Create the Kubernetes Cluster

Provider



- Select Provider

Create Cluster

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

Provider

KubeVirt

Cancel

→ Next

01 – Create the Kubernetes Cluster

Provider

- Select Provider
- Select Datacenter

Create Cluster

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

Provider

KubeVirt

Datacenter

east

× Cancel

→ Next

01 – Create the Kubernetes Cluster



- Cluster name

Create Cluster

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

Cluster

Name*

Specification

Control Plane Version* 1.33.5

Container Runtime* containerd

Admission Plugins

Audit Logging ☒ custom metadata minimal recommended

Cluster Backup ☐

Disable CSI Driver ☐

Network Configuration

☒ cilium ☐ canal ☐ None

CNI Plugin Version 1.18.2

Annotations

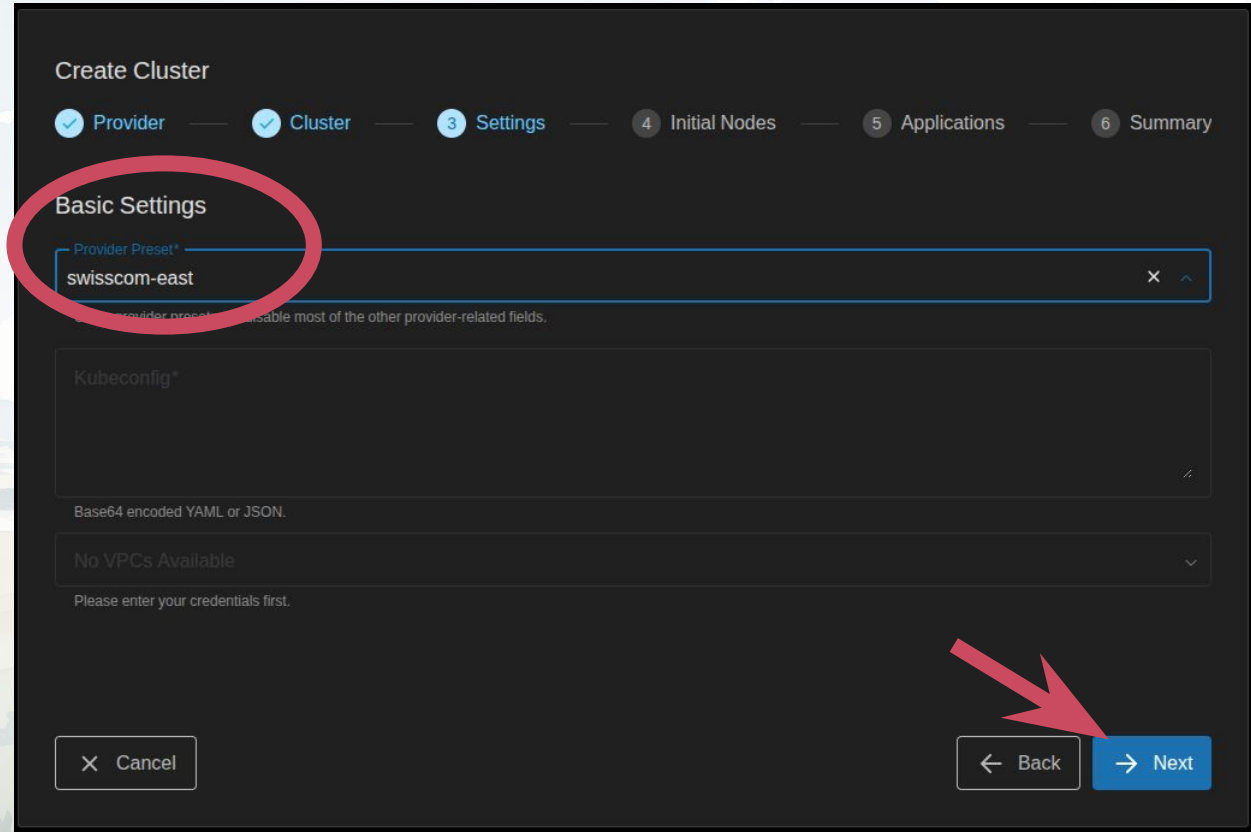
Key Value

← Back → Next

01 – Create the Kubernetes Cluster

Settings

- Select settings



Create Cluster

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

Basic Settings

Provider Preset*
swisscom-east

Kubeconfig*

Base64 encoded YAML or JSON.

No VPCs Available

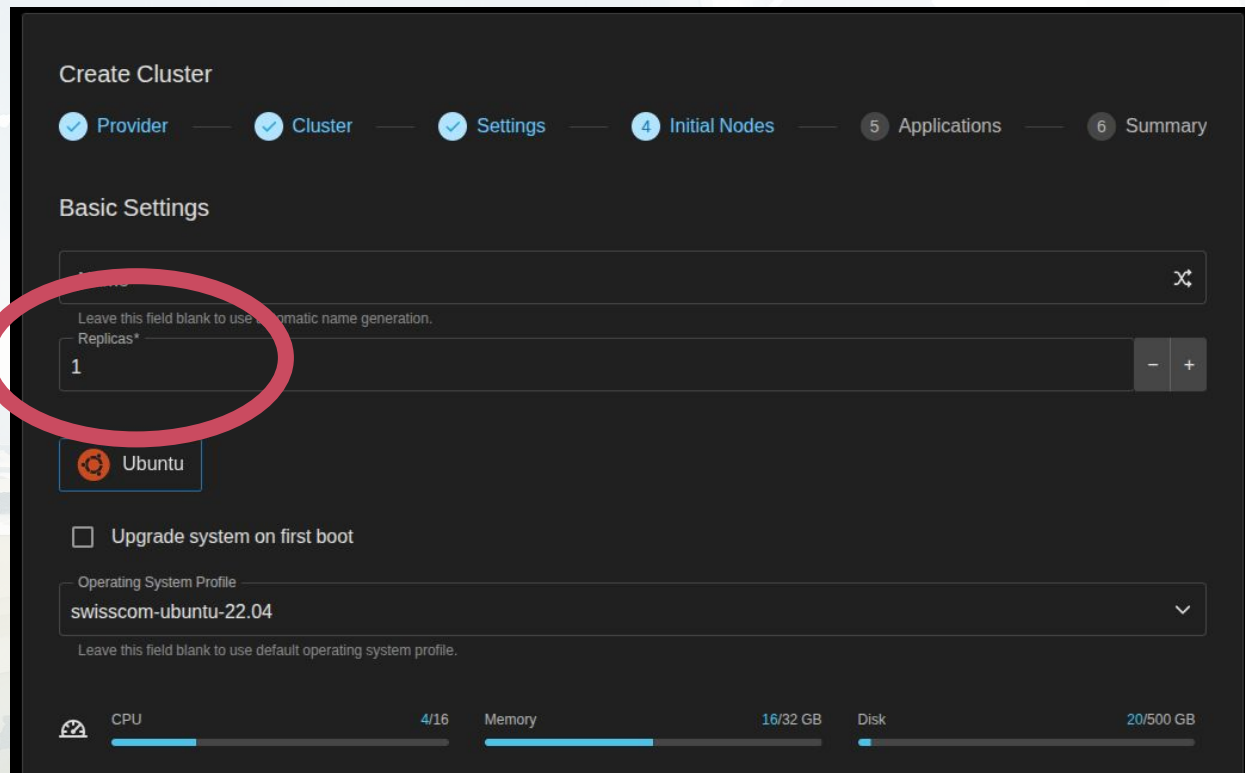
Please enter your credentials first.

Cancel Back 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

Ubuntu

☐ Upgrade system on first boot

Operating System Profile

swisscom-ubuntu-22.04

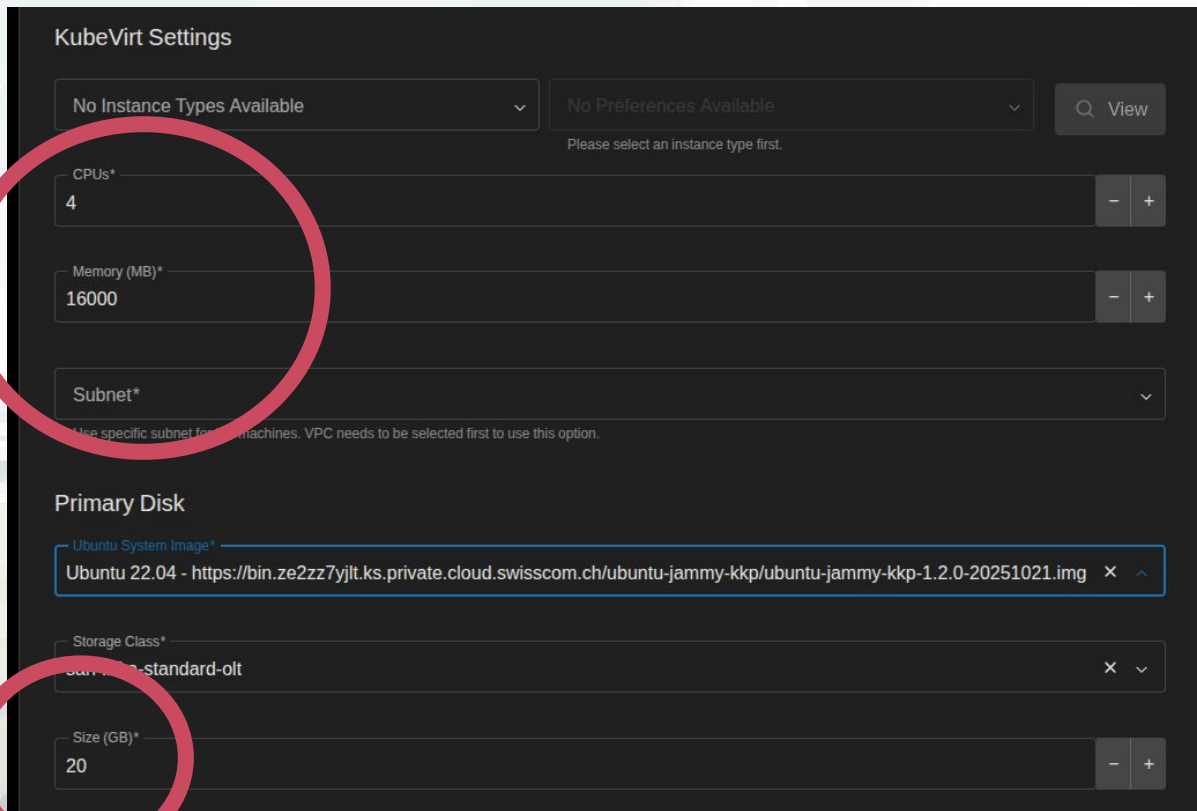
Leave this field blank to use default operating system profile.

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

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 View

Please select an instance type first.

CPU* 4 - +

Memory (MB)* 16000 - +

Subnet* Use specific subnet for machines. VPC needs to be selected first to use this option.

Primary Disk

Ubuntu System Image* Ubuntu 22.04 - <https://bin.ze2zz7yilt.ks.private.cloud.swisscom.ch/ubuntu-jammy-kkp/ubuntu-jammy-kkp-1.2.0-20251021.img> x ^

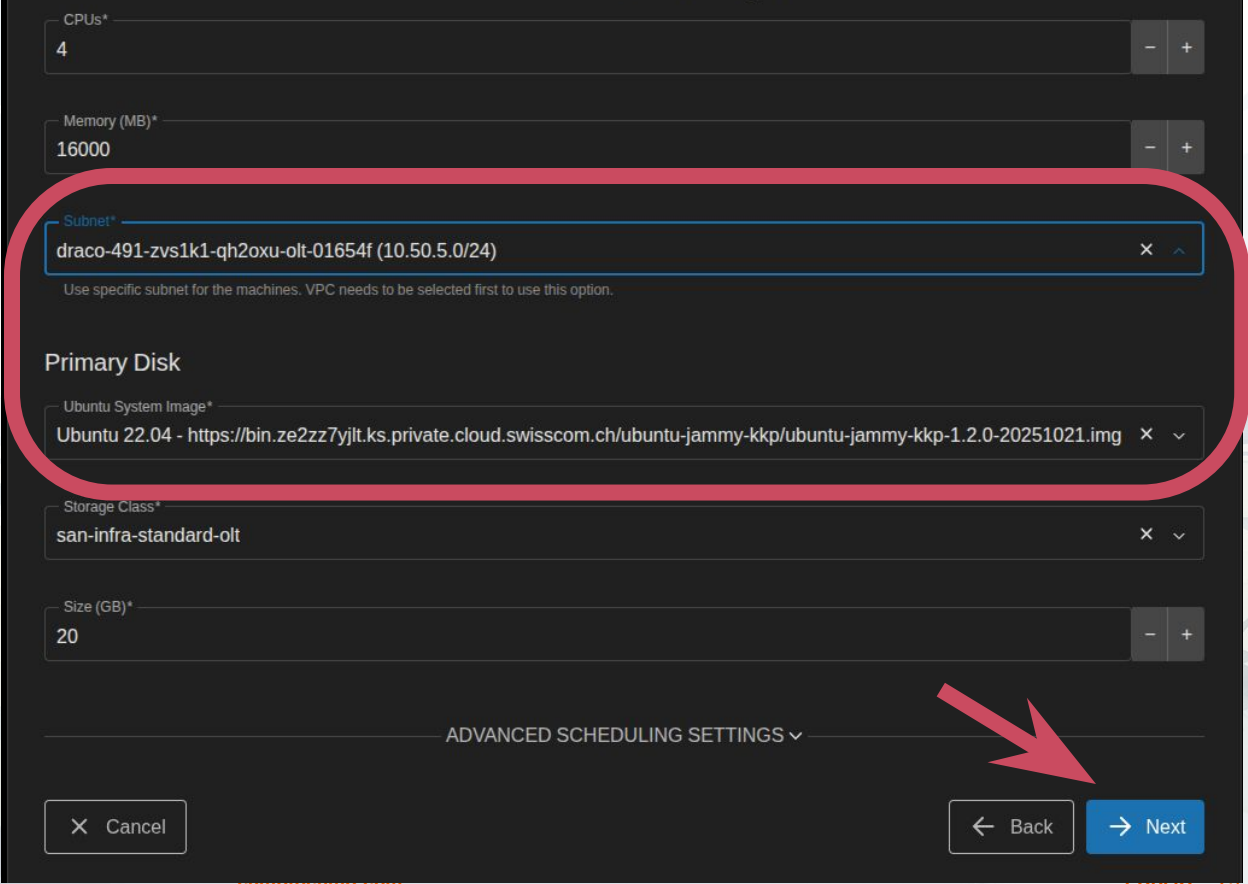
Storage Class* standard-olt x v

Size (GB)* 20 - +

01 – Create the Kubernetes Cluster

Initial Nodes

- Select any subnet
- Select a Primary disk



The screenshot shows the 'Create Instance' page in the AWS Management Console. A pink rounded rectangle highlights the 'Subnet' and 'Primary Disk' sections. The 'Subnet' dropdown is set to 'draco-491-zvs1k1-qh2oxu-olt-01654f (10.50.5.0/24)'. The 'Primary Disk' section shows 'Ubuntu System Image*' set to 'Ubuntu 22.04' and 'Storage Class*' set to 'san-infra-standard-olt'. The 'Size (GB)*' is set to '20'. At the bottom, there is a 'Cancel' button, a 'Back' button, and a 'Next' button. A pink arrow points to the 'Next' button.

CPU*s
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.ze2zz7yilt.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

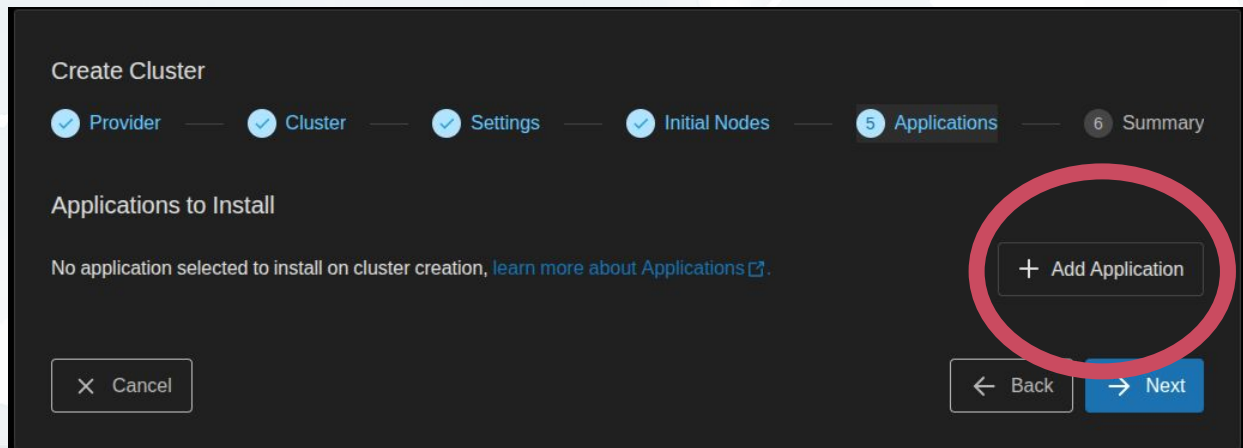
ADVANCED SCHEDULING SETTINGS ▾

Cancel Back Next

01 – Create the Kubernetes Cluster

Applications

- Add Nginx



Create Cluster

✓ Provider — ✓ Cluster — ✓ Settings — ✓ Initial Nodes — 5 Applications — 6 Summary

Applications to Install

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

+ Add Application

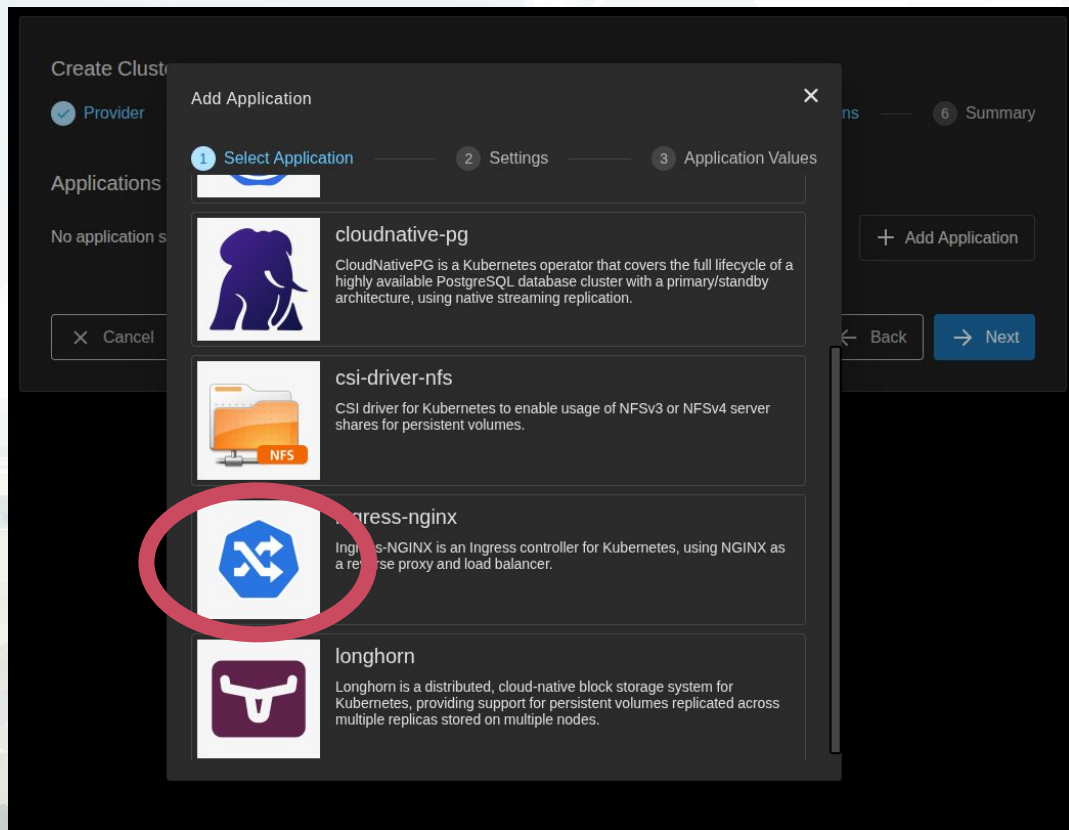
× Cancel

← Back → Next

01 – Create the Kubernetes Cluster

Applications

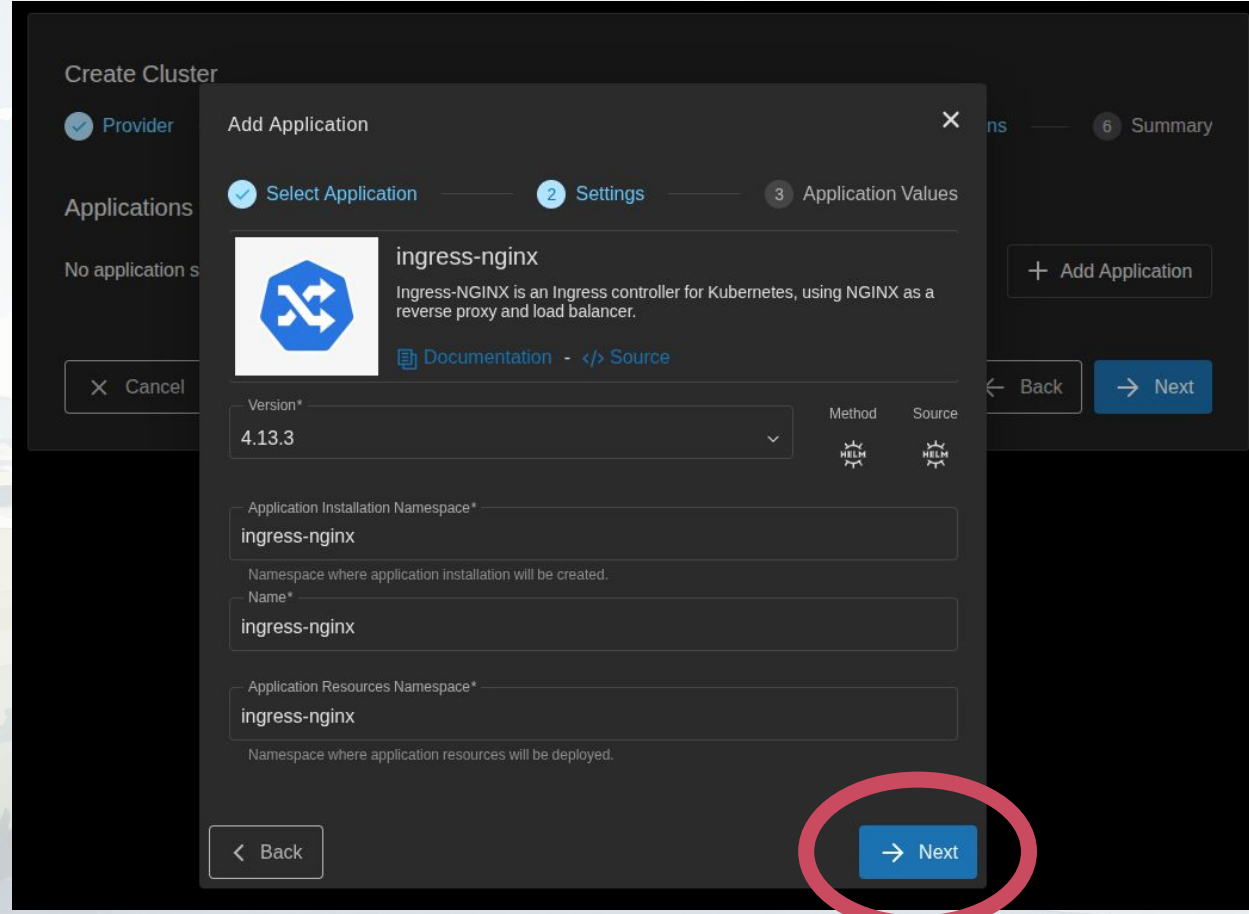
- Add Nginx



01 – Create the Kubernetes Cluster

Applications

- Add Nginx



The screenshot shows a 'Create Cluster' wizard with a modal 'Add Application' dialog. The dialog has three steps: '1 Select Application' (active), '2 Settings', and '3 Application Values'. The 'ingress-nginx' application is selected, with a description: 'Ingress-NGINX is an Ingress controller for Kubernetes, using NGINX as a reverse proxy and load balancer.' Below this, there are links for 'Documentation' and 'Source'. The 'Version' dropdown is set to '4.13.3'. There are radio buttons for 'Method' (Helm) and 'Source' (Helm). Three text input fields are present, all containing 'ingress-nginx': 'Application Installation Namespace*', 'Name*' (with a subtext 'Namespace where application installation will be created.'), and 'Application Resources Namespace*' (with a subtext 'Namespace where application resources will be deployed.'). At the bottom, there are 'Back' and 'Next' buttons. The 'Next' button is highlighted with a red circle.

Create Cluster

✓ Provider

Applications

No application s

Cancel

Add Application

✓ Select Application — 2 Settings — 3 Application Values

 **ingress-nginx**
Ingress-NGINX is an Ingress controller for Kubernetes, using NGINX as a reverse proxy and load balancer.
[Documentation](#) - [Source](#)

Version*
4.13.3

Method Source
 

Application Installation Namespace*
ingress-nginx

Namespace where application installation will be created.
Name*
ingress-nginx

Application Resources Namespace*
ingress-nginx

Namespace where application resources will be deployed.

Back Next

01 – Create the Kubernetes Cluster

Applications

- Add Nginx

Create Cluster

✓ Provider

Applications

No application s

✕ Cancel

ns — 6 Summary

✕ Add Application

✓ Select Application — ✓ Settings — 3 Application Values

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

values.yaml

```
1 global:
2   image:
3     registry: registry.ze2zz7yjlts.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

← Back + Add Application

01 – Create the Kubernetes Cluster

Applications



- Add Nginx
- Add Openmetrics Particle Accelerator Lab

Create Cluster

✓ Provider — ✓ Cluster — ✓ Settings — ✓ Initial Nodes — 5 Applications — 6 Summary

Applications to Install

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

+ Add Application

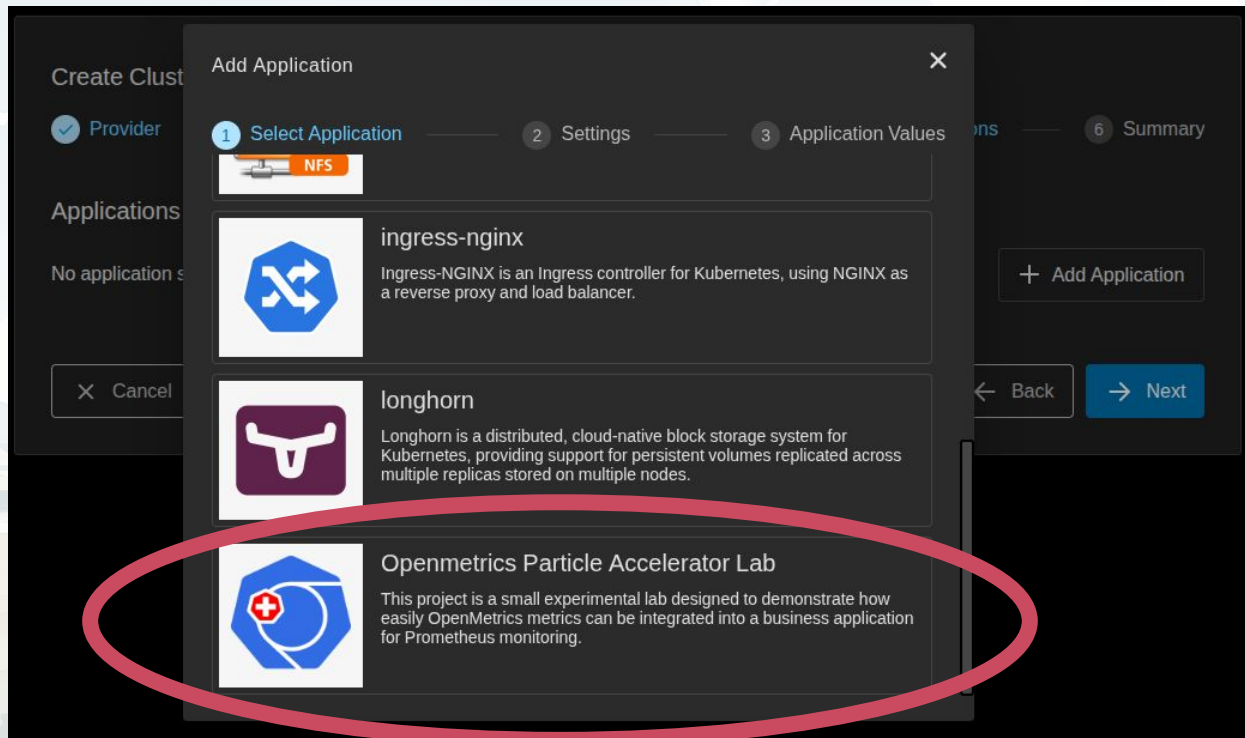
× Cancel

← Back → Next

01 – Create the Kubernetes Cluster

Applications

- Add Nginx
- Add Openmetrics Particle Accelerator Lab



01 – Create the Kubernetes Cluster

Applications

- Add Nginx
- Add Openmetrics Particle Accelerator Lab

Add Application

1 Select Application 2 Settings 3 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  

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](#)

01 – Create the Kubernetes Cluster Applications

- Add Nginx
- Add Openmetrics Particle Accelerator Lab

The screenshot shows the 'Add Application' dialog in the camptocamp-01 project interface. The dialog has a title bar with 'swisscom' and 'Projects camptocamp-01'. It features a progress bar with three steps: 'Select Application' (checked), 'Settings' (checked), and '3 Application Values' (active). Below the progress bar, it says 'Optional: Use custom values to override the default configuration for this application.' A text area labeled 'values.yaml' contains the following content:

```
1 # empty
2
```

Below the text area, it says 'Input should be valid YAML'. At the bottom of the dialog, there are two buttons: '< Back' and '+ Add Application'. A red arrow points to the '+ Add Application' button. In the background, the 'Create Cluster' dialog is visible, showing a progress bar with six steps: 'Provider' (checked), 'Applications' (checked), 'No application s' (checked), 'Cancel' (checked), 'Summary' (checked), and 'Add Application' (checked).

01 – Create the Kubernetes Cluster

Applications

- Add Nginx
- Add Openmetrics Particle Accelerator Lab

Create Cluster

✓ Provider — ✓ Cluster — ✓ Settings — ✓ Initial Nodes — 5 Applications — 6 Summary

Applications to Install

Q Search + Add Application

ingress-nginx  

 4.13.3

 ingress-nginx

Openmetrics Particl  

 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 Datacenter
KubeVirt east (CH)

2 Cluster

4 Initial Nodes

GENERAL

Name	Replicas	Operating System
Autogenerated name	1	Ubuntu
Operating System Profile	Node Annotations	
swisscom-ubuntu-22.04		

✓ Node Local DNS Cache

✓ Konnectivity

SPECIFICATION

✓ Audit Logging ✓ User SSH Key Agent

✓ Kubermatic KubeLB ✓ Kubernetes Dashboard

3 Settings

Preset
swisscom-east

5 Applications

ingress-nginx

 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

particle-accelerator

Control Plane: 1.33.5

CN: 1.18.2

Cluster: mh5pm5vkr8

Created: a few seconds ago

Seed: scs

Region: CH (east)

Provider: KubeVirt

Preset: swisscom-east

Container Runtime: containerd

SSH Keys: No assigned keys

Open Dashboard

Web Terminal

ADDITIONAL CLUSTER INFORMATION

Control Plane

- API Server
- etcd
- Scheduler
- Controller
- Machine Controller
- Operating System Manager
- User Controller Manager
- Kubernetes Dashboard
- Kubermatic KubeLB

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

OPA

- Policy Control

Node Local DNS Cache

Konnektivity

01 – Create the Kubernetes Cluster

Summary



instructions on
lab.camptocamp.com

- 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

→02

02 – Install tools

kubectl

- Cluster nodes are not reachable



Kubernetes Nodes

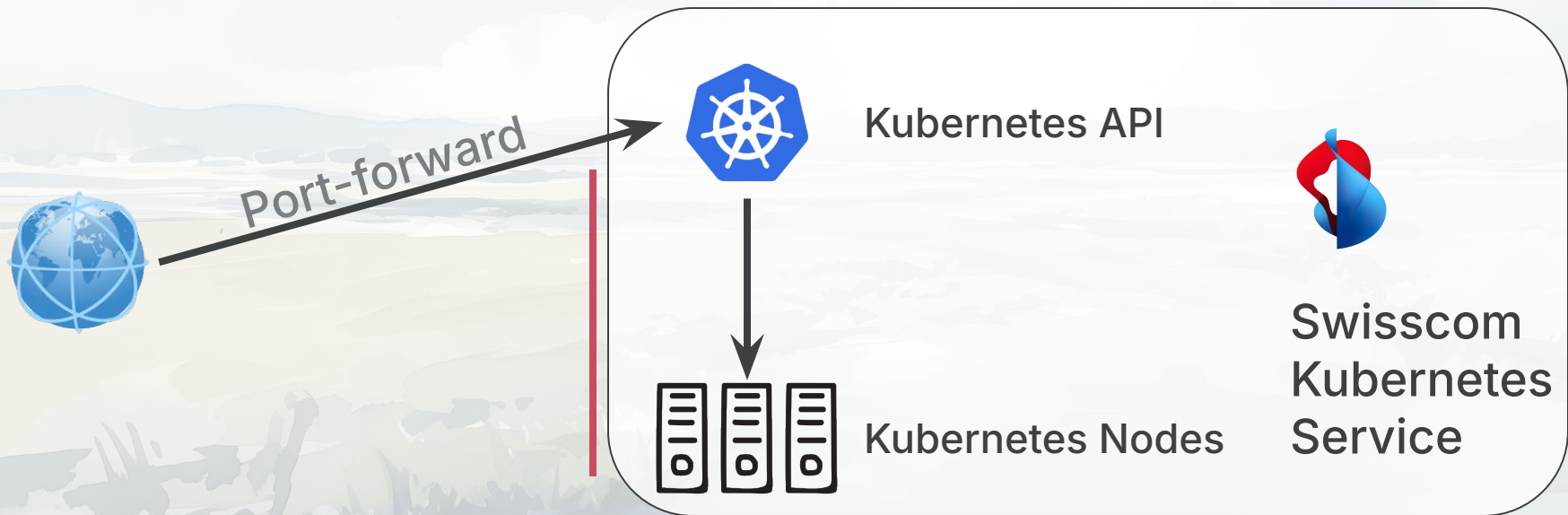


Swisscom
Kubernetes
Service

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.camptocamp.com
- 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:

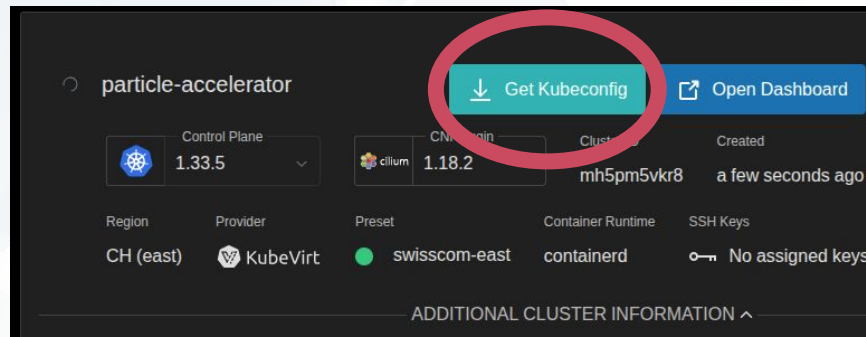
```
curl.exe -LO "https://dl.k8s.io/release/v1.34.2/bin/windows/amd64/kubectl.exe"
```

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

02 – Install tools

Configure API Access

- Locate kubeconfig-xxxxxxxxx
- Setup path to the kubeconfig file



```
# Linux/Mac
```

```
export KUBECONFIG=~/.Downloads/kubeconfig-xxxxxxxxx
```

```
# Windows PowerShell
```

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

02 – Install tools

Validate Access



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

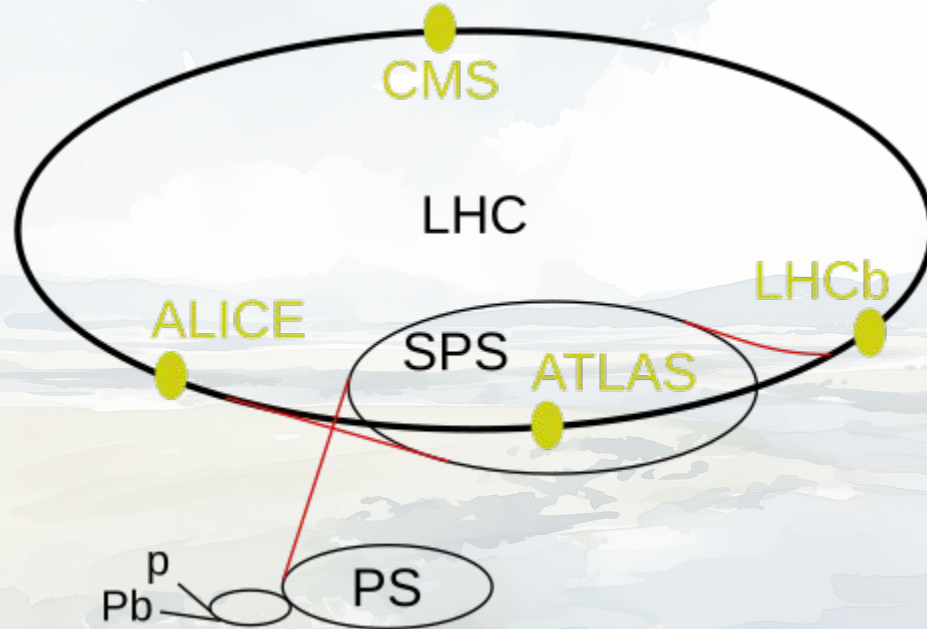
NAMESPACE	NAME	READY	STATUS	RESTARTS	AGE
ingress-nginx	ingress-nginx-controller-549886-cpg2d	1/1	Running	0	6h42m
ingress-nginx	ingress-nginx-controller-549886-plbz5	1/1	Running	0	6h42m
kube-system	cilium-m7wf6	1/1	Running	0	10h
kube-system	cilium-operator-74668d995-hrpjh	1/1	Running	0	10h
kube-system	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	konnnectivity-agent-7f7d9474dd-4pxjv	1/1	Running	0	10h
kube-system	konnnectivity-agent-7f7d9474dd-gkp5s	1/1	Running	0	10h
kube-system	metrics-server-7bd6766f9d-5grc8	1/1	Running	0	10h

```
...
```

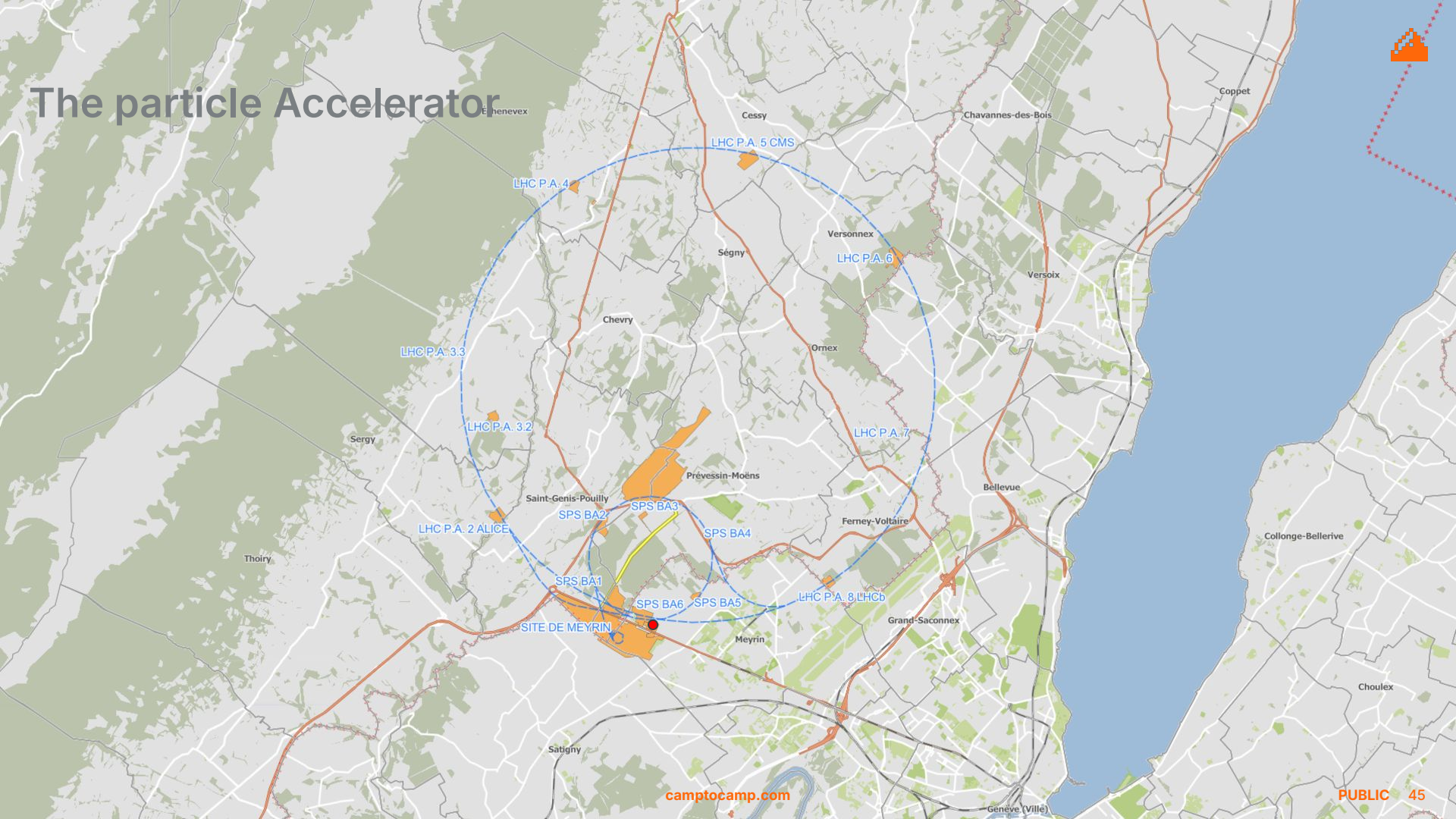
→03

03 – Python based Particle Accelerator

The particle Accelerator



The particle Accelerator

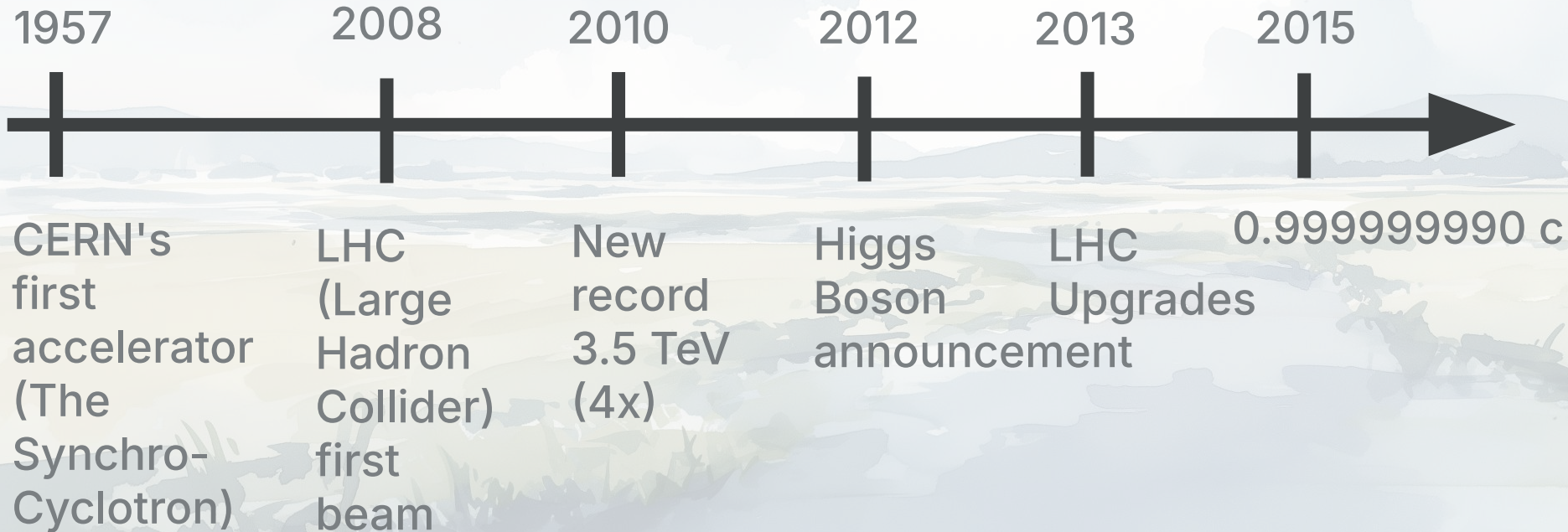


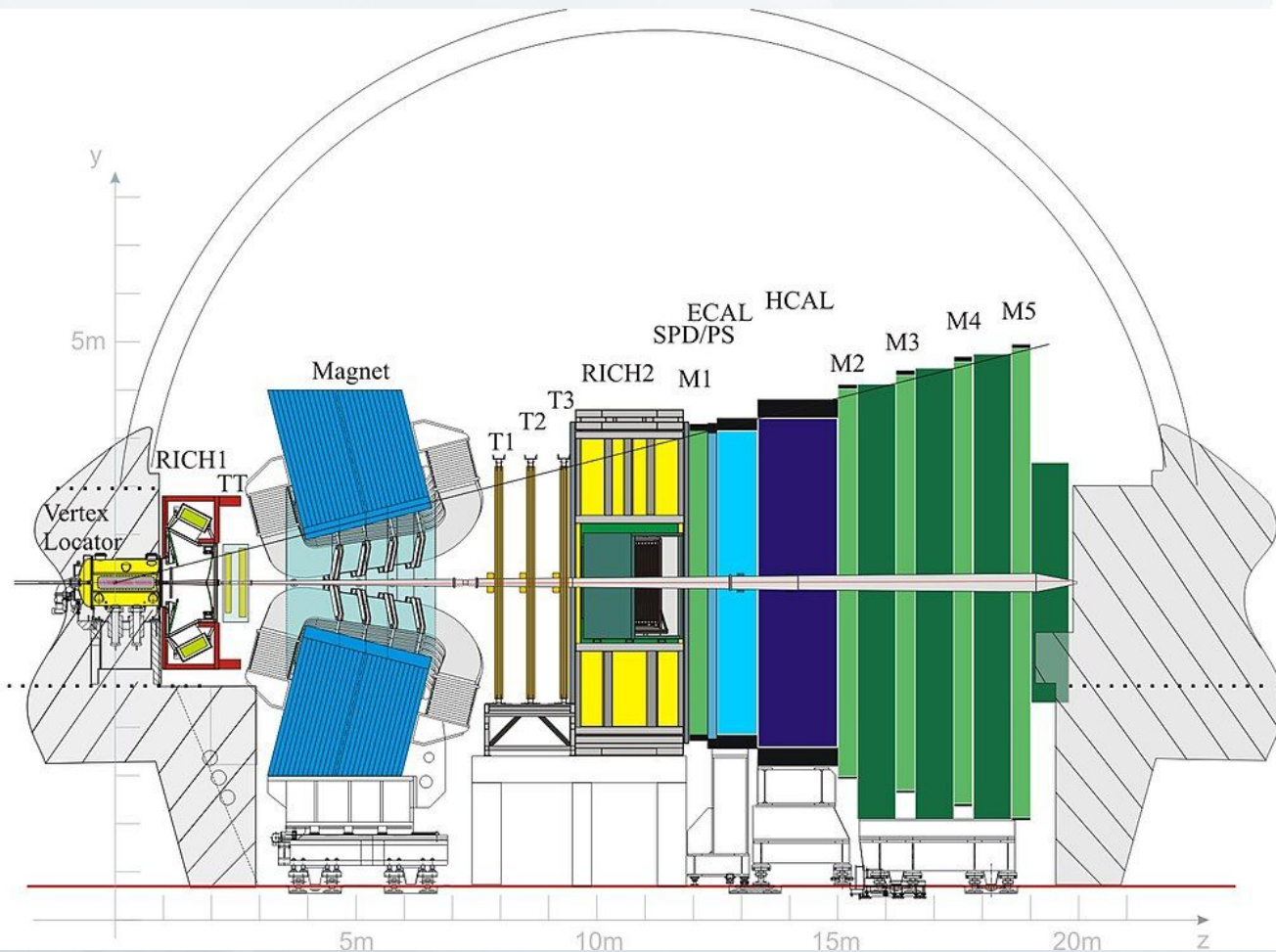
The particle Accelerator





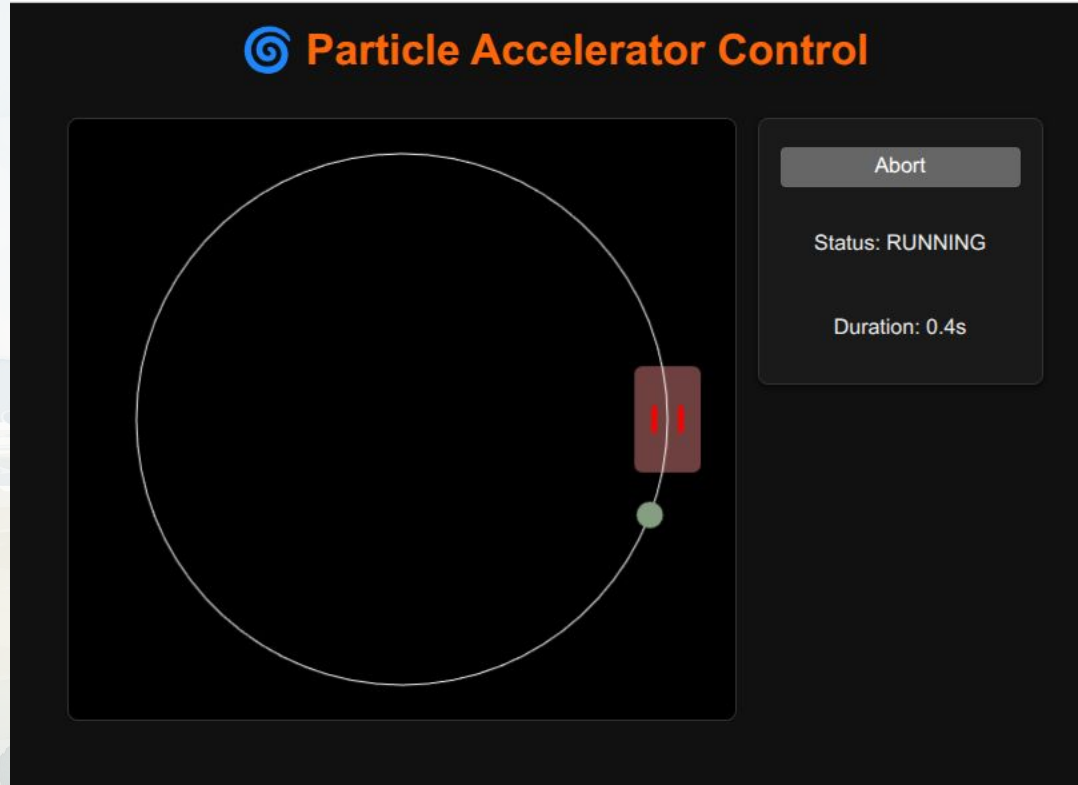
The particle Accelerator





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
 - 0.999999999% 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

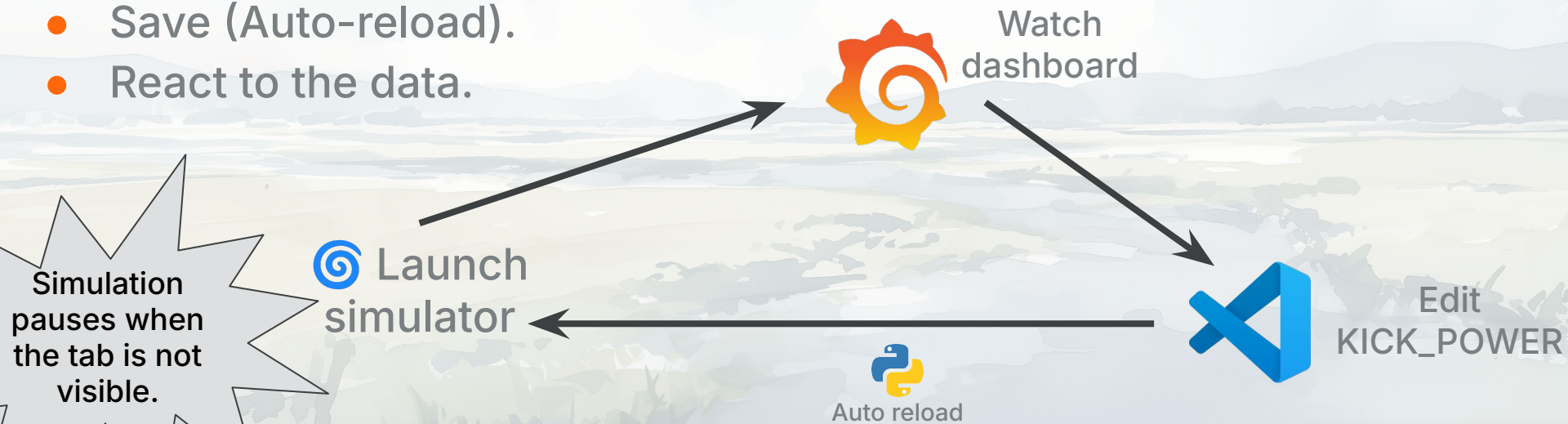


- The Stack:
 - Cluster: Ephemeral Kubernetes Environment
 - App: Python (Flask) simulating the accelerator physics
 - Observability: OpenMetrics (client library) → Prometheus → Grafana
- The Workflow:
 - Deploy: Launch the lab via the dashboard
 - Port Forward: Establish a secure tunnel to the cluster

04 – The Lab

The Feedback Loop

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



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.

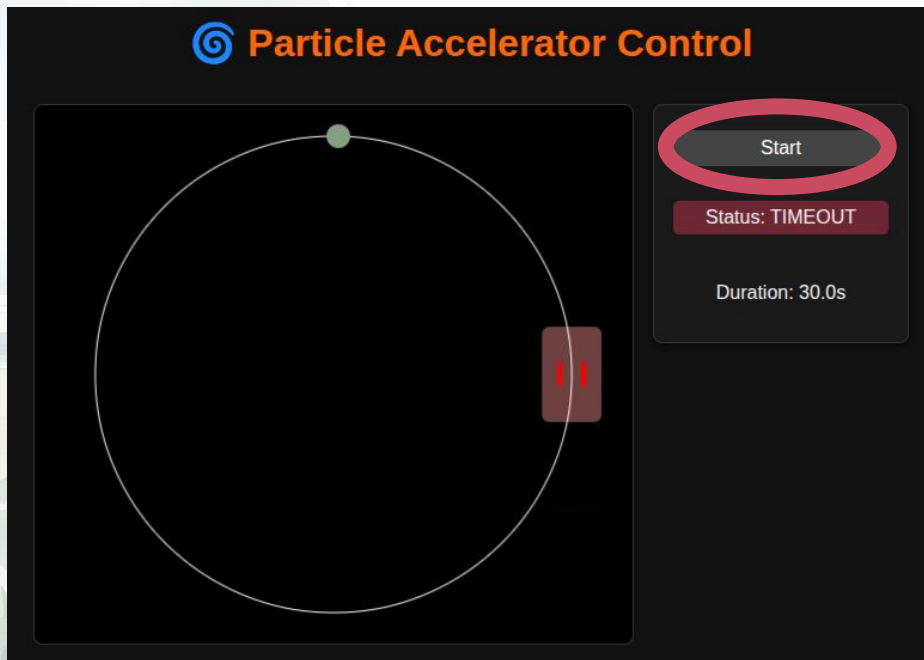
04 – The Lab

The Feedback Loop

- Launch simulator

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

Simulation
pauses when
the tab is not
visible.

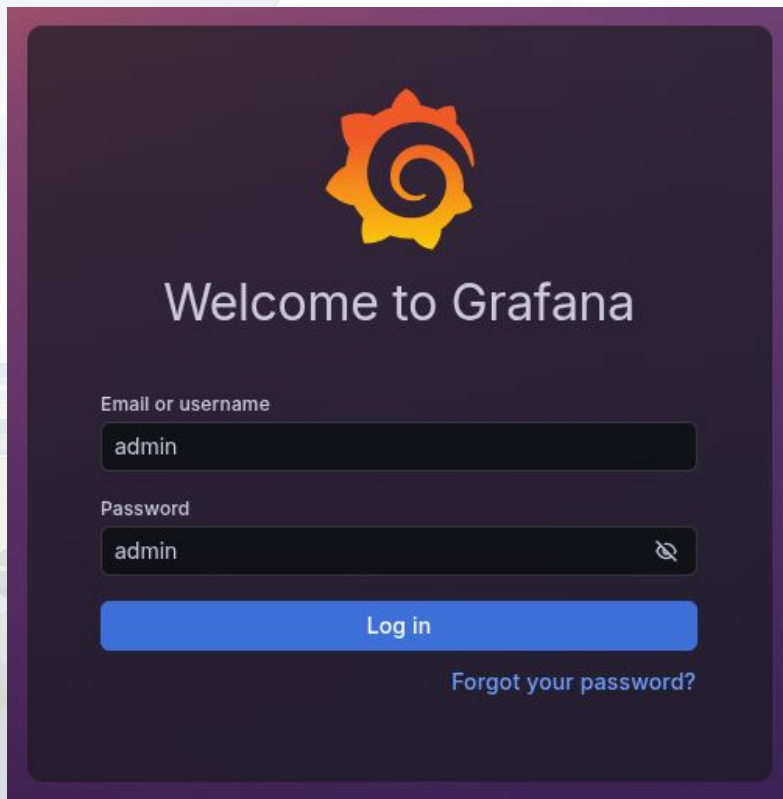


04 – The Lab

The Feedback Loop

- Launch simulator
- Watch Grafana

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



The image shows the Grafana login interface. At the top center is the Grafana logo, a stylized orange and yellow gear with a spiral inside. Below the logo, the text "Welcome to Grafana" is displayed in a white, sans-serif font. Underneath this, there are two input fields. The first is labeled "Email or username" and contains the text "admin". The second is labeled "Password" and also contains the text "admin", with a small eye icon to its right for toggling visibility. Below these fields is a prominent blue button with the text "Log in" in white. At the bottom right of the login area, there is a link that says "Forgot your password?". The entire login form is set against a dark purple background with a subtle gradient.

04 – The Lab

The Feedback Loop

- Launch simulator
- Watch Grafana

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



Update your password

 Continuing to use the default password exposes you to security risks.

New password

Confirm new password

Submit

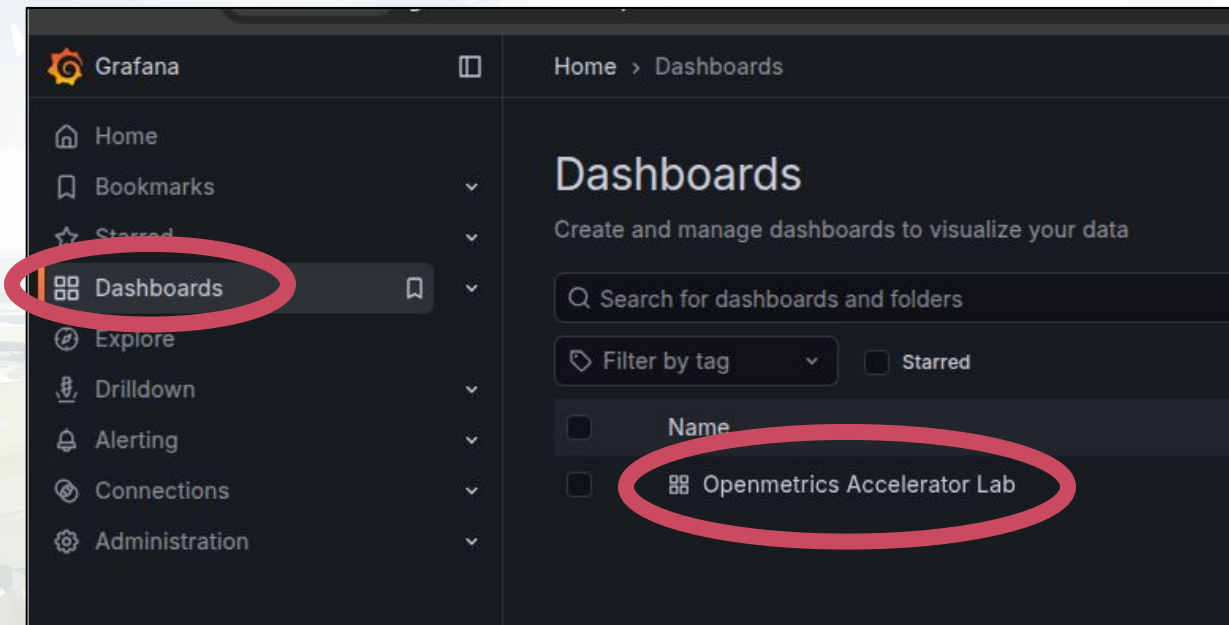
Skip

04 – The Lab

The Feedback Loop

- Launch simulator
- Watch Grafana

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



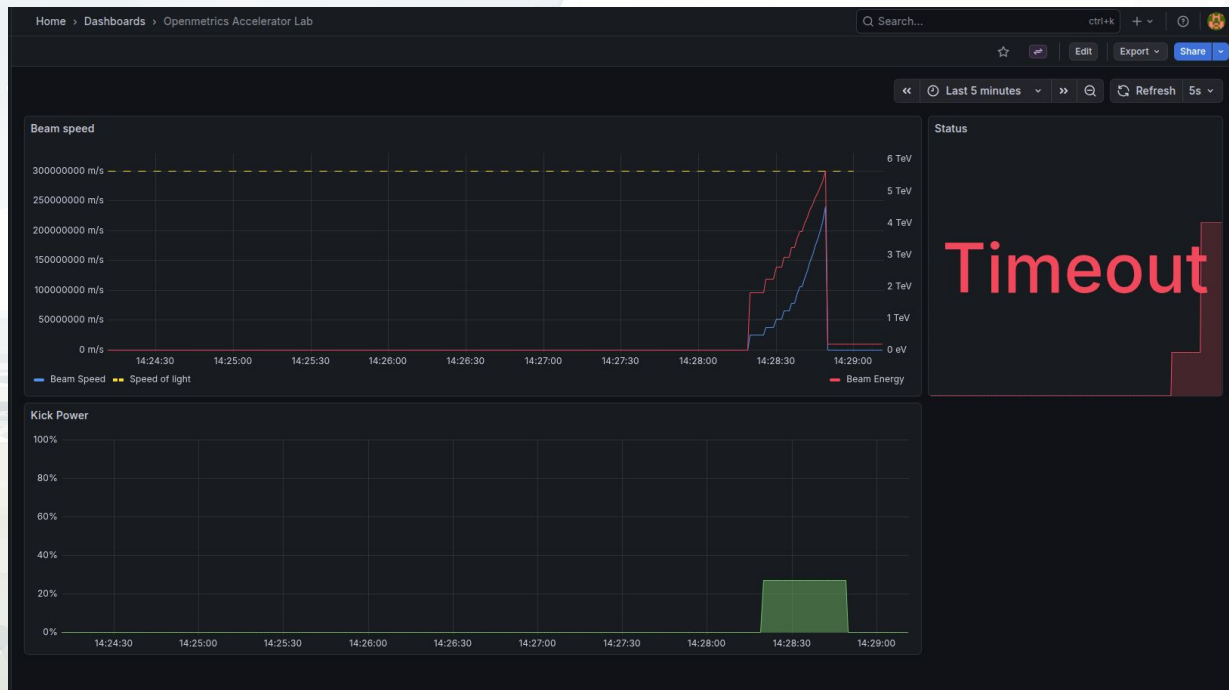
04 – The Lab

The Feedback Loop



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

- Launch simulator
- Watch Grafana

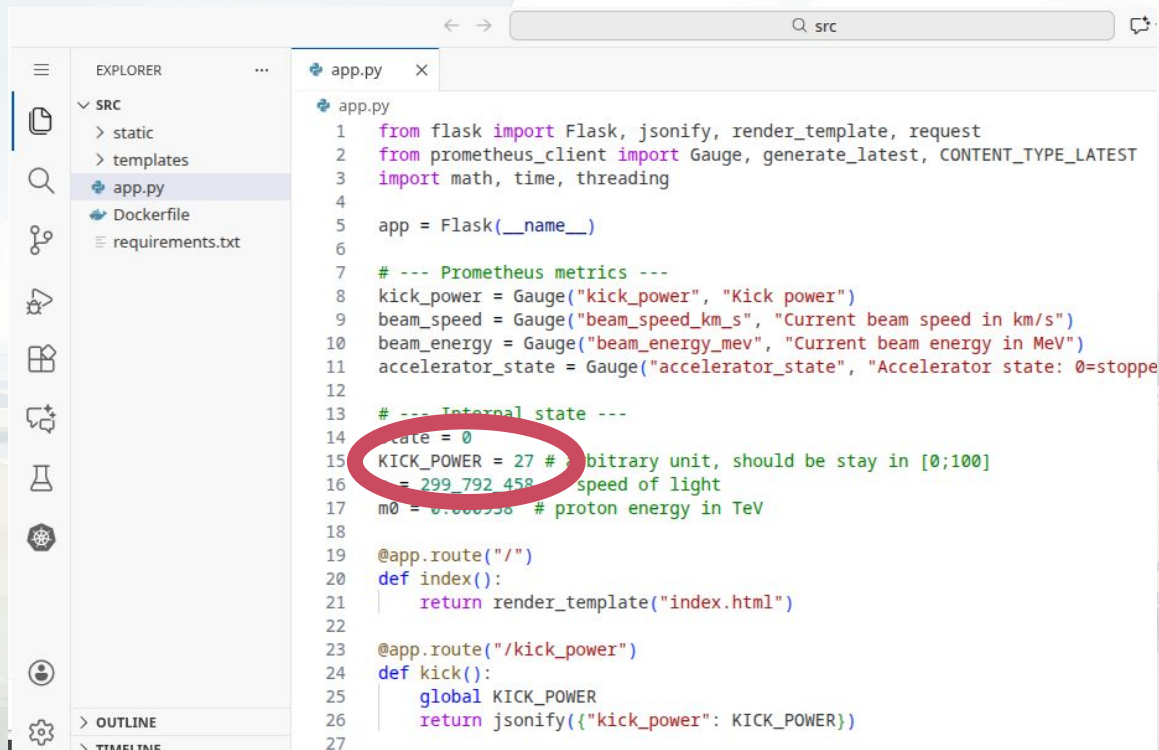


04 – The Lab

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 = 0.000938 # 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)

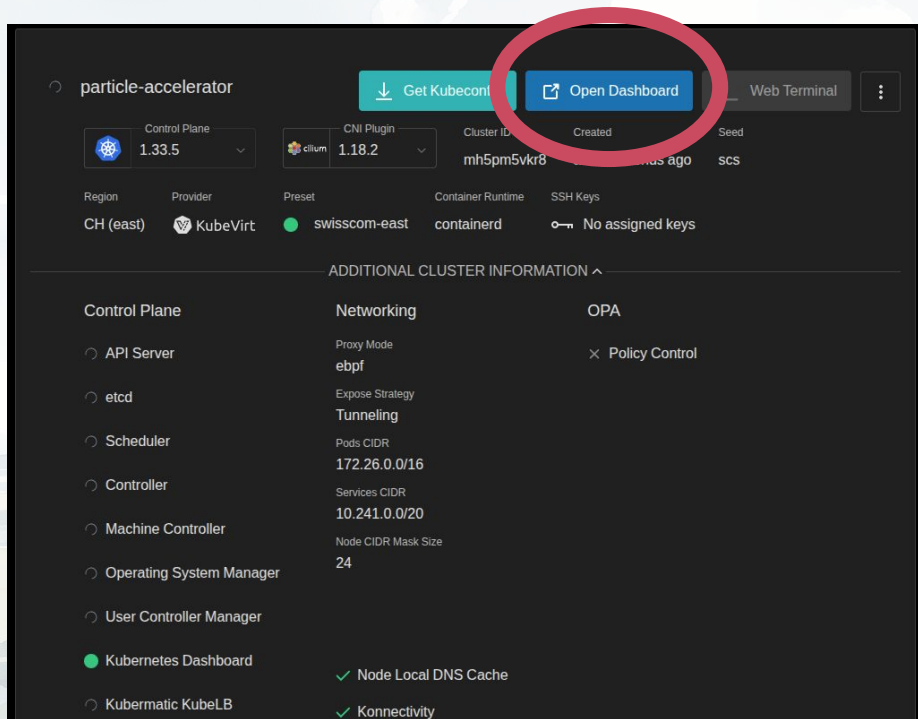
<http://vscode.127.0.0.1.nip.io:8080/>

04 – The Lab

Restart Lab



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



04 – The Lab

Application observability



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

particle-accelerator

Get Kubeconfig Open Dashboard Web Terminal

Control Plane	CNI Plugin	Cluster ID	Created	Seed
1.33.5	cilium 1.18.2	mh5pm5vkr8	...	scs

Region: CH (east) Provider: KubeVirt Preset: swisscom-east Container Runtime: containerd SSH Keys: No assigned keys

ADDITIONAL CLUSTER INFORMATION

Control Plane	Networking	OPA
- API Server - etcd - Scheduler - Controller - Machine Controller - Operating System Manager - User Controller Manager - Kubernetes Dashboard - Kubermatic KubeLB	- 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 - Node Local DNS Cache: ✓ - Konnektivity: ✓	Policy Control: ×

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

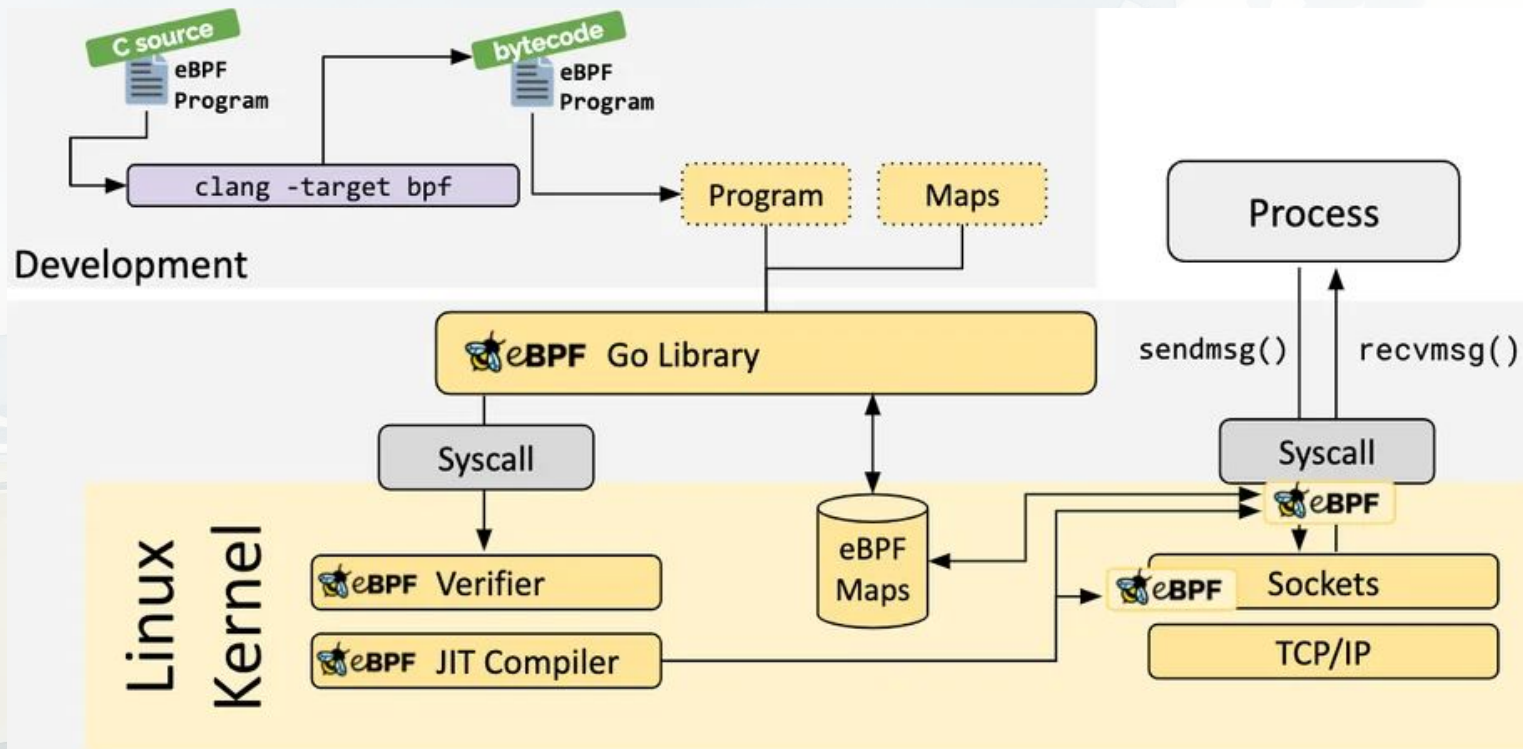


- Focus on stability and security
- User writes eBPF program with C language
- The verifier checks that:
 - user is allowed to inject eBPF programs
 - the program always run to completion
- Maps are used to exchange data between Kernel and user-space



05 – Observability tools using EBPF

eBPF



Runtime

Source: <https://ebpf.io/what-is-ebpf/#loader--verification-architecture>

05 – Observability tools using EBPF



- BPFTrace : <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.