Swisscom Managed K8s Hands-on with Camptocamp

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

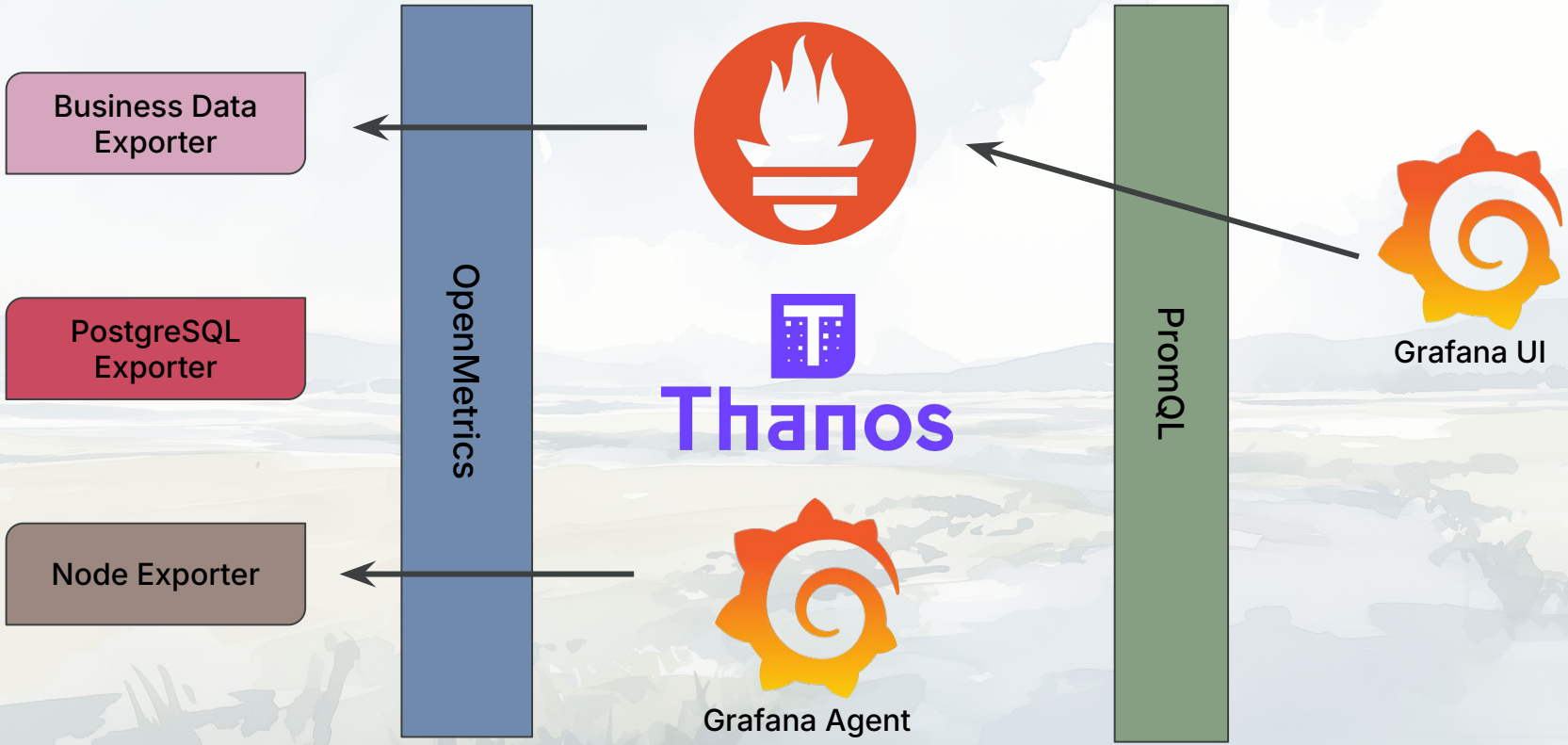
00	Warm up	–	OpenMetrics
01	Bootstrap	–	Create the Kubernetes Cluster
02	Init	–	Install tools
03	Story	–	Python based Particle Accelerator
04	Run	–	Deploy the lab and play with the metrics
05	Discover	–	Observability tools using EBPF

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
  - Counter: only increase, cumulative
  - Histogram, Summary

```
# 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

```python
view_metric = {}

@app.route('/view/<id>')
def view_product(id):
    # Update metric
    if product not in view_metric:
        view_metric[id] = 1
    else:
        view_metric[id] += 1
  return "View %s" % id
```

```python
@app.route('/metrics')
def metrics():
    metrics = ""
    for id in view_metric:
        metrics += 'view_total{product="%s"} %s\n'
                    % (id, view_metric[id])
    for id in buy_metric:
        metrics += 'buy_total{product="%s"} %s\n'
                    % (id, buy_metric[id])
    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, ...

```python
from prometheus_client import Counter

view_metric = Counter('view', 'Product view', ['product'])

@app.route('/view/<id>')
def view_product(id):
    # Update metric
    view_metric.labels(product=id).inc()
    return "View %s" % id
```

```python
from prometheus_client import generate_latest

@app.route('/metrics')
def metrics():
    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()')
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 Counter
2
3 stock_metric = Counter('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 – Create the Kubernetes Cluster

Get the Welcome card

The welcome card contains:

- URL to the Swisscom Web UI: https://xxxx.ks.private.cloud.swisscom.ch/
- Username
- Password

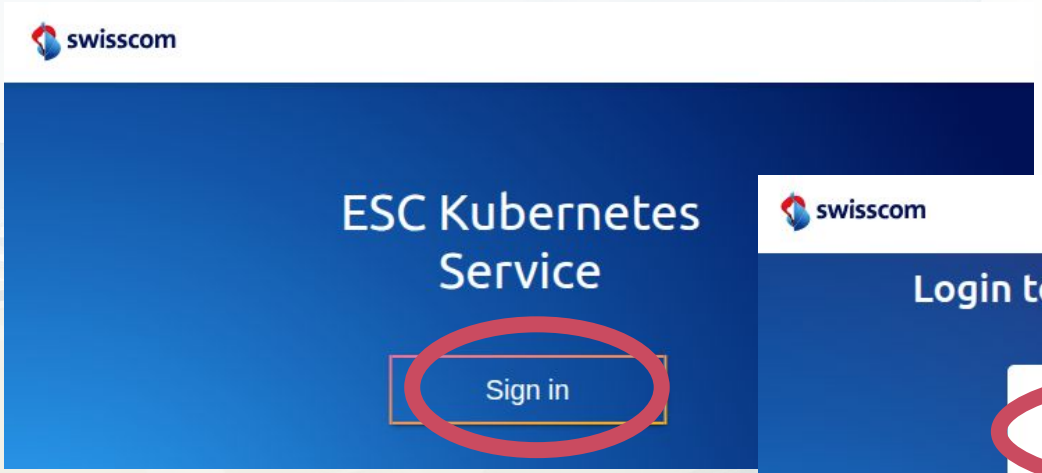
The welcome page contains:

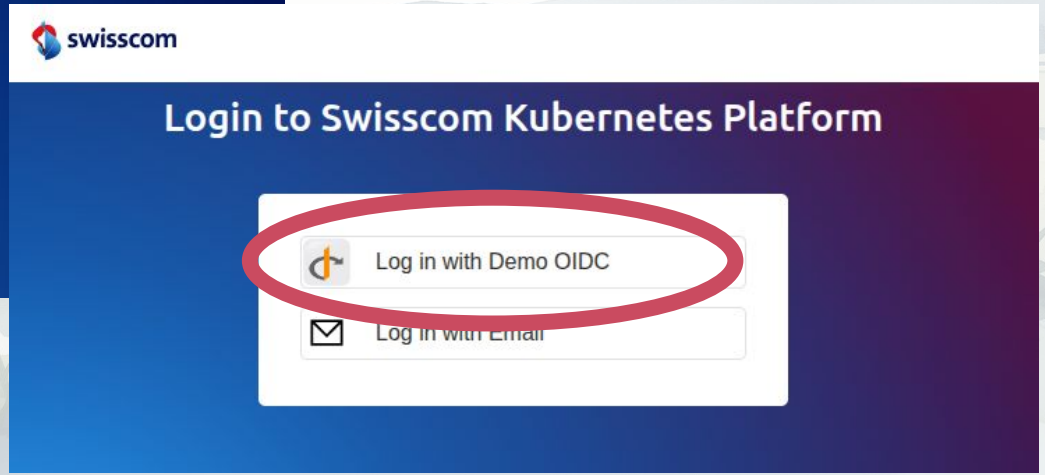
- Lab Instructions
- Commands for copy/paste
- https://lab.campto.camp/

01 – Create the Kubernetes Cluster

Sign in

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





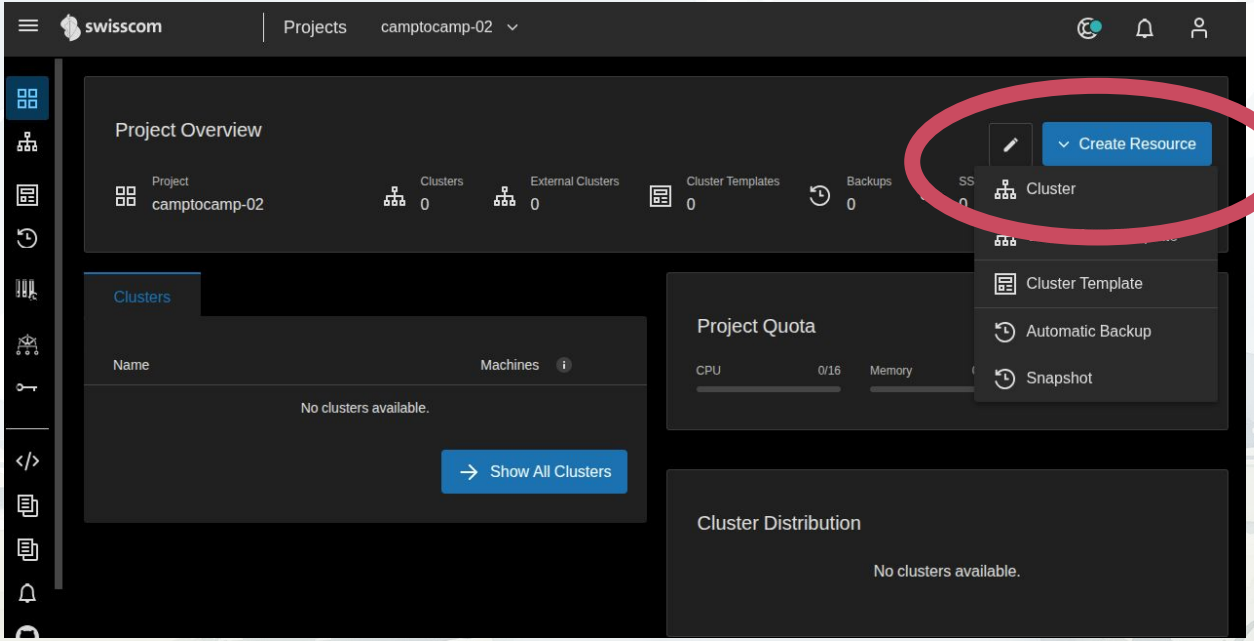
01 – Create the Kubernetes Cluster

Choose project

01 – Create the Kubernetes Cluster

Create a new cluster

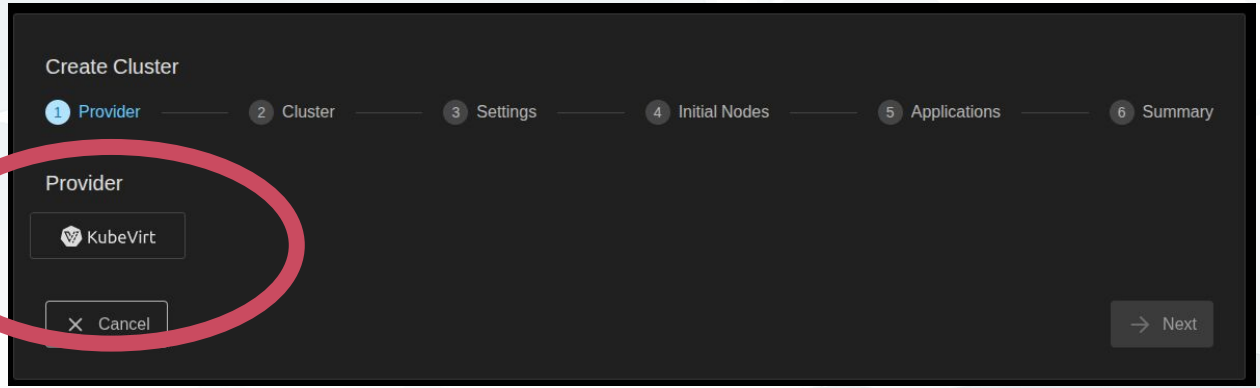
- Create Resource
  - Cluster



01 – Create the Kubernetes Cluster

Provider

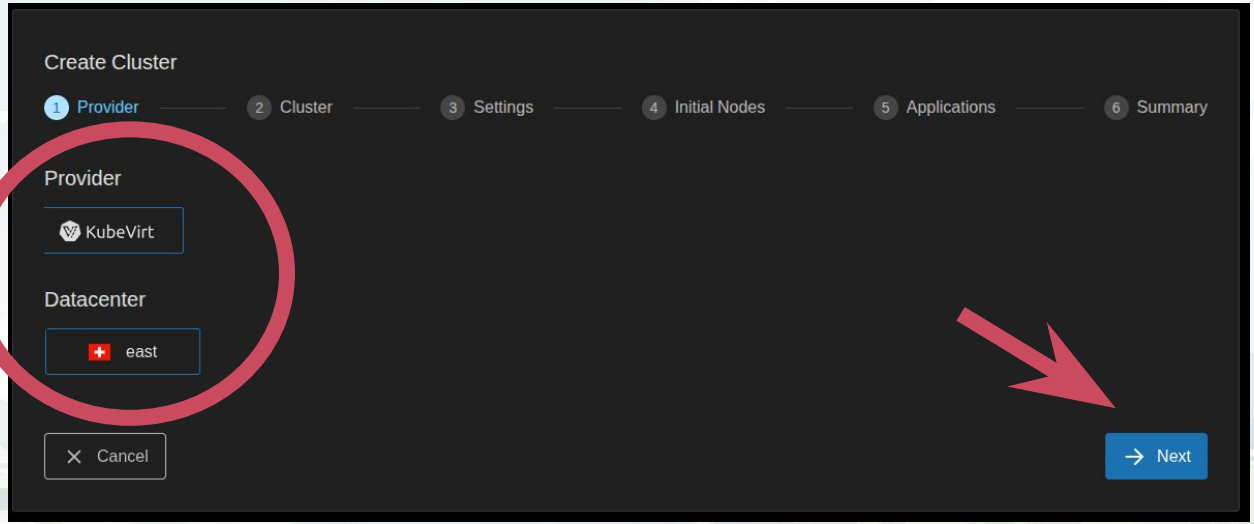
- Select Provider



01 – Create the Kubernetes Cluster

Provider

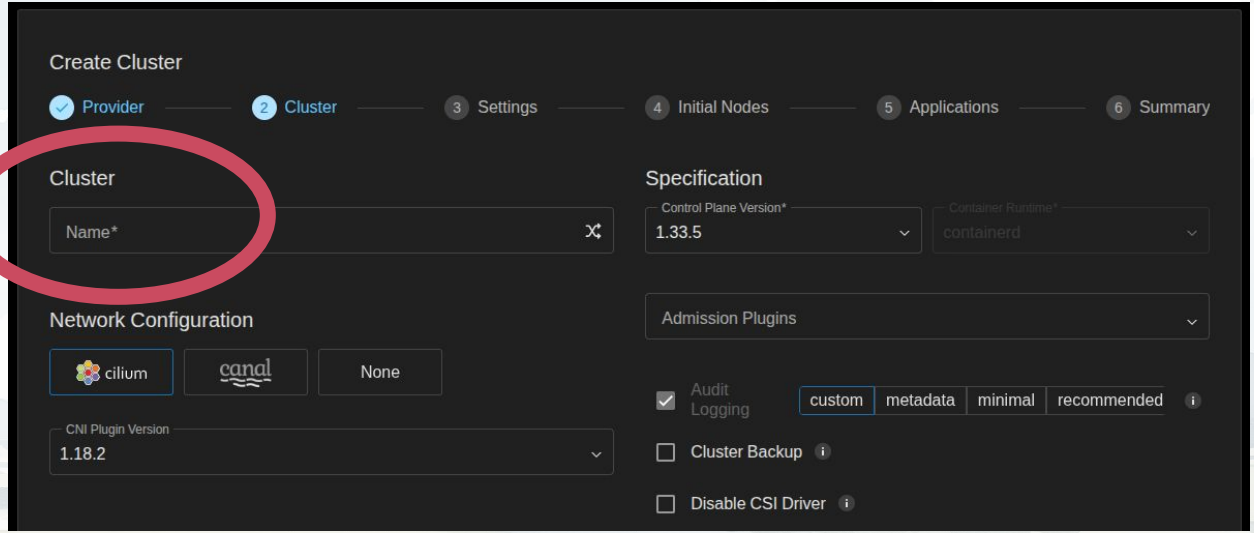
- Select Provider
- Select Datacenter

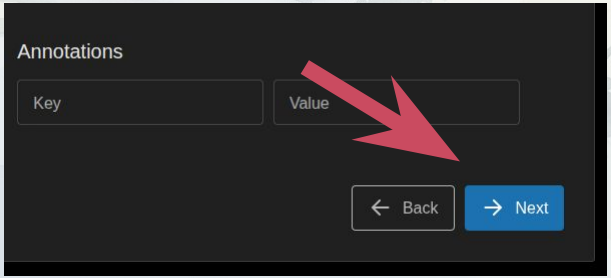


01 – Create the Kubernetes Cluster

Cluster

- Cluster name

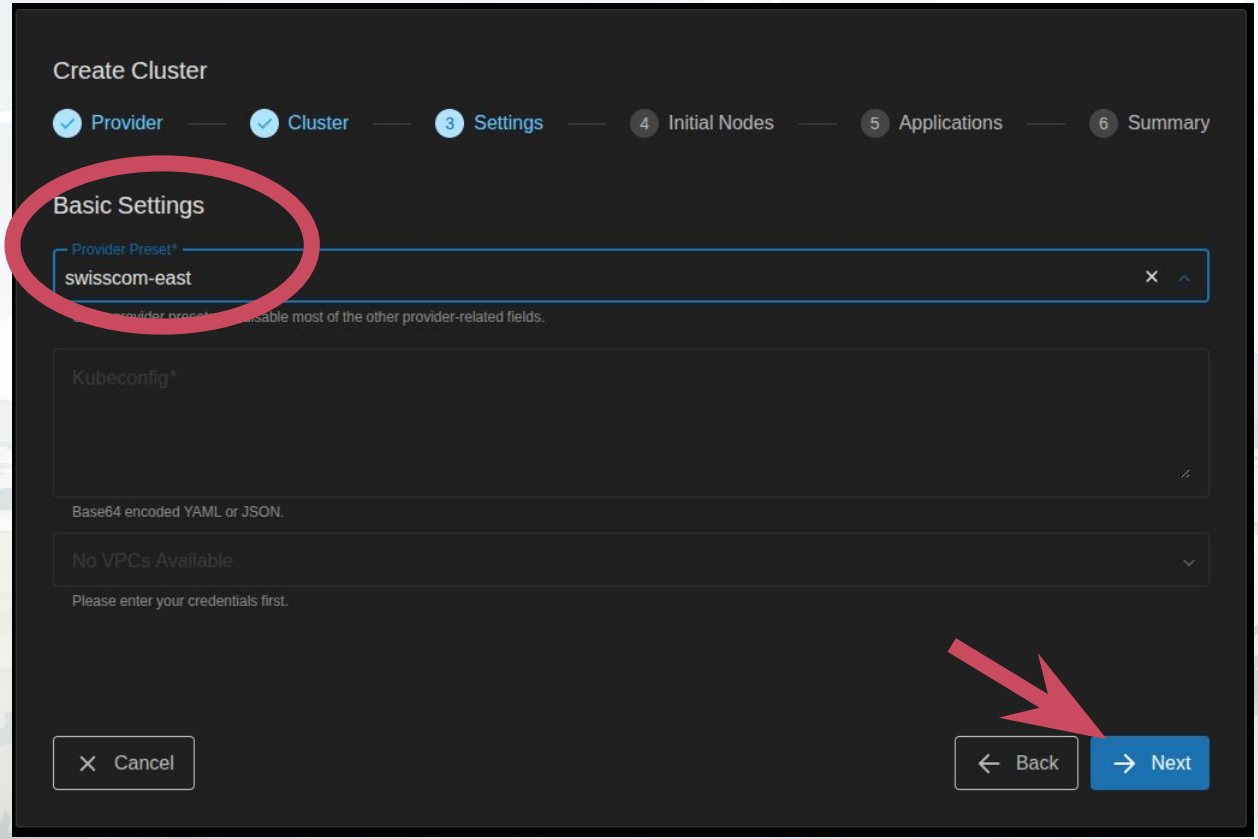




01 – Create the Kubernetes Cluster

Settings

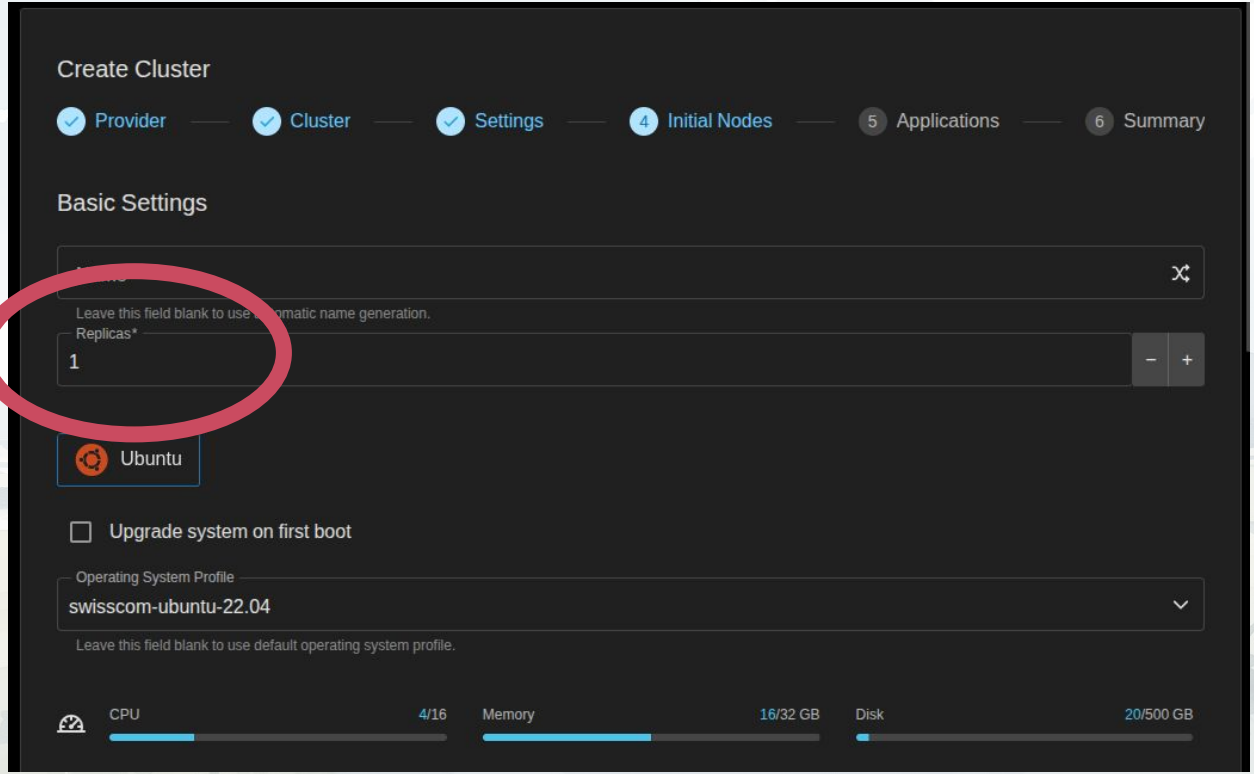
- Select settings



01 – Create the Kubernetes Cluster

Initial Nodes

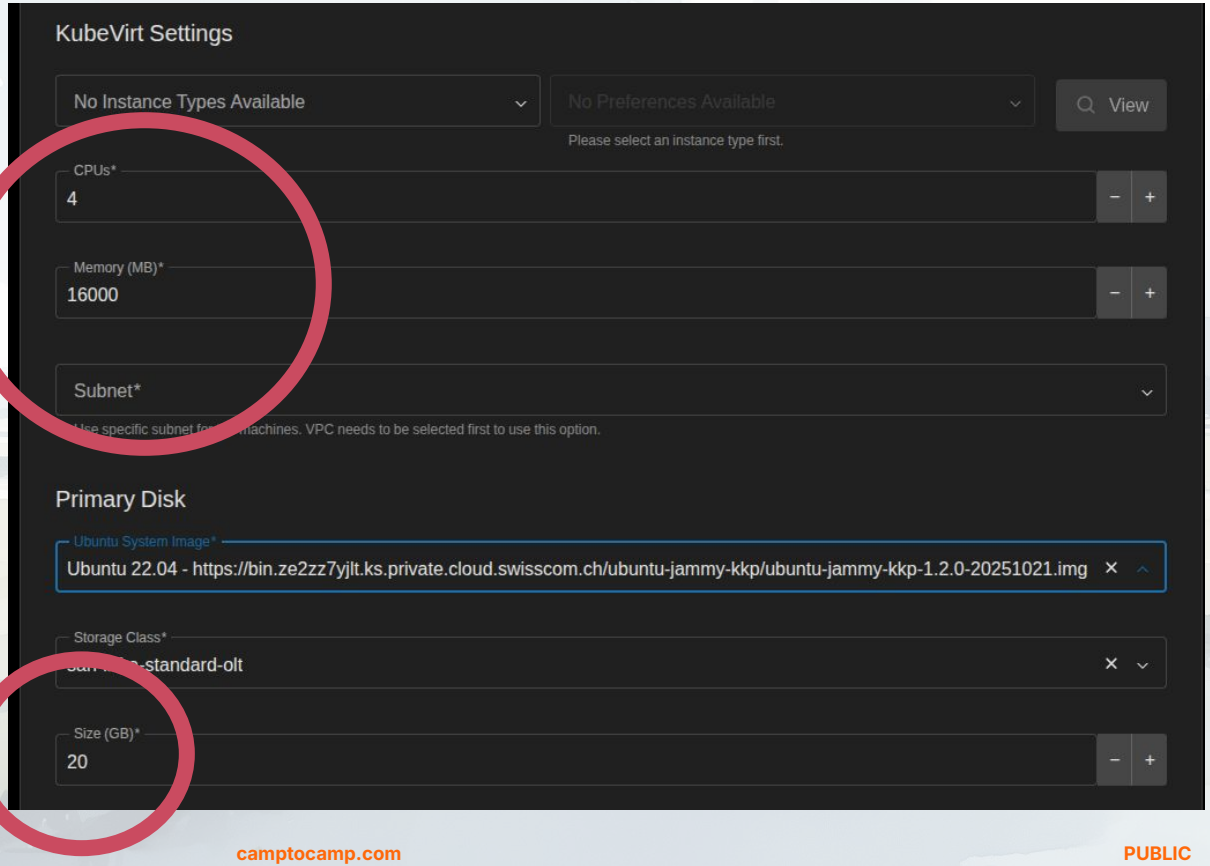
- Deploy only one node



01 – Create the Kubernetes Cluster

Initial Nodes

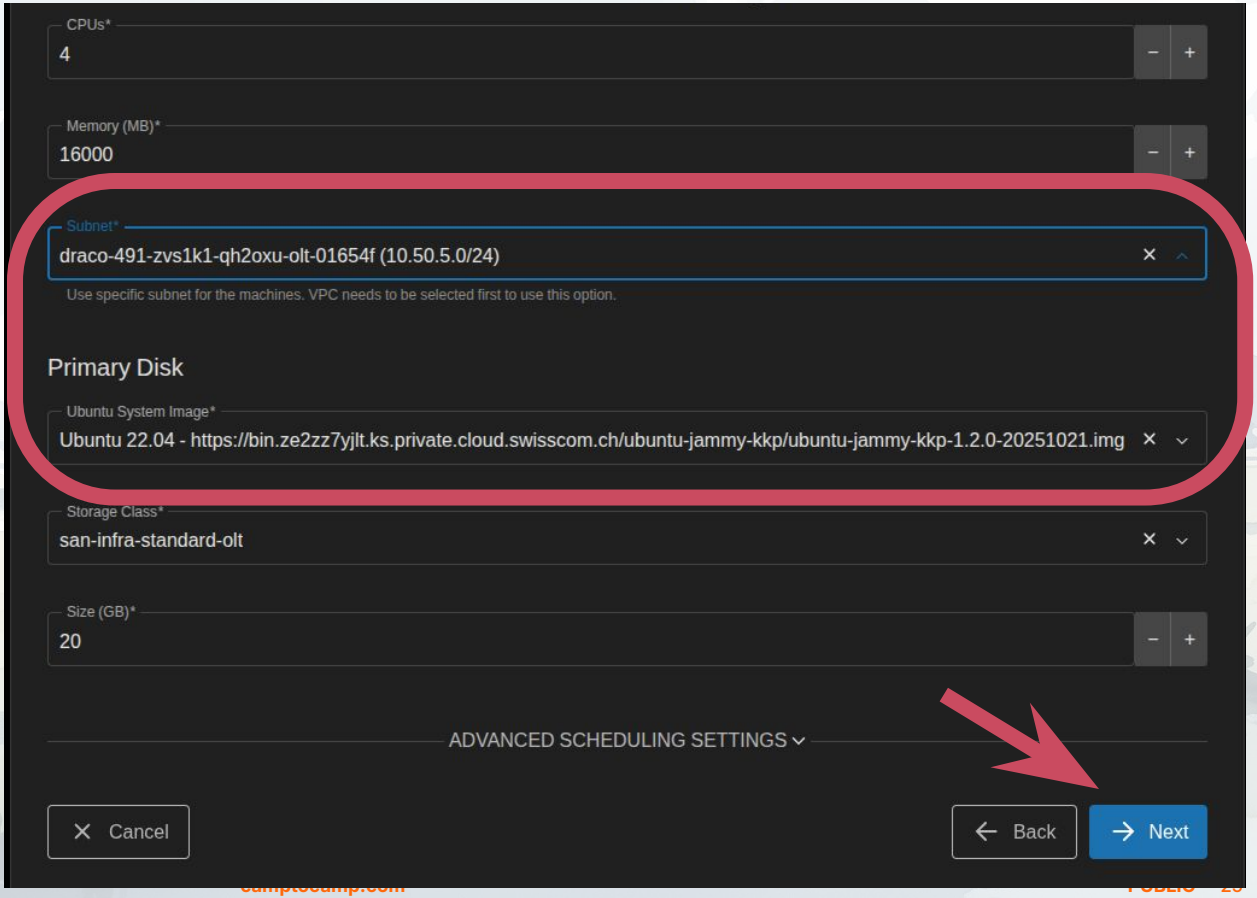
- Deploy only one node
- Increase CPU to 4
- Increase Memory to 16 GB
- Increase Disk size to 20 GB



01 – Create the Kubernetes Cluster

Initial Nodes

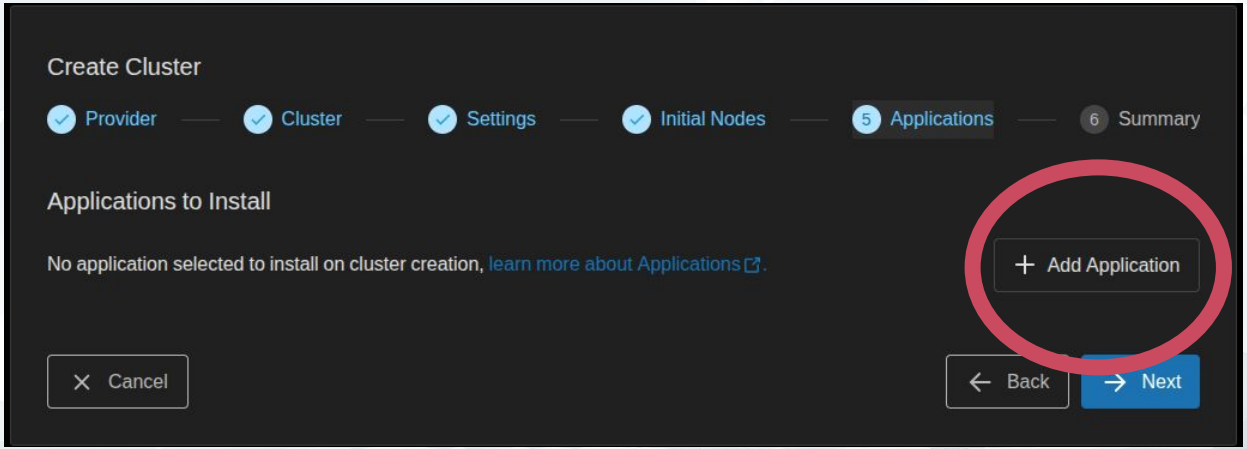
- Select any subnet
- Select a Primary disk



01 – Create the Kubernetes Cluster

Applications

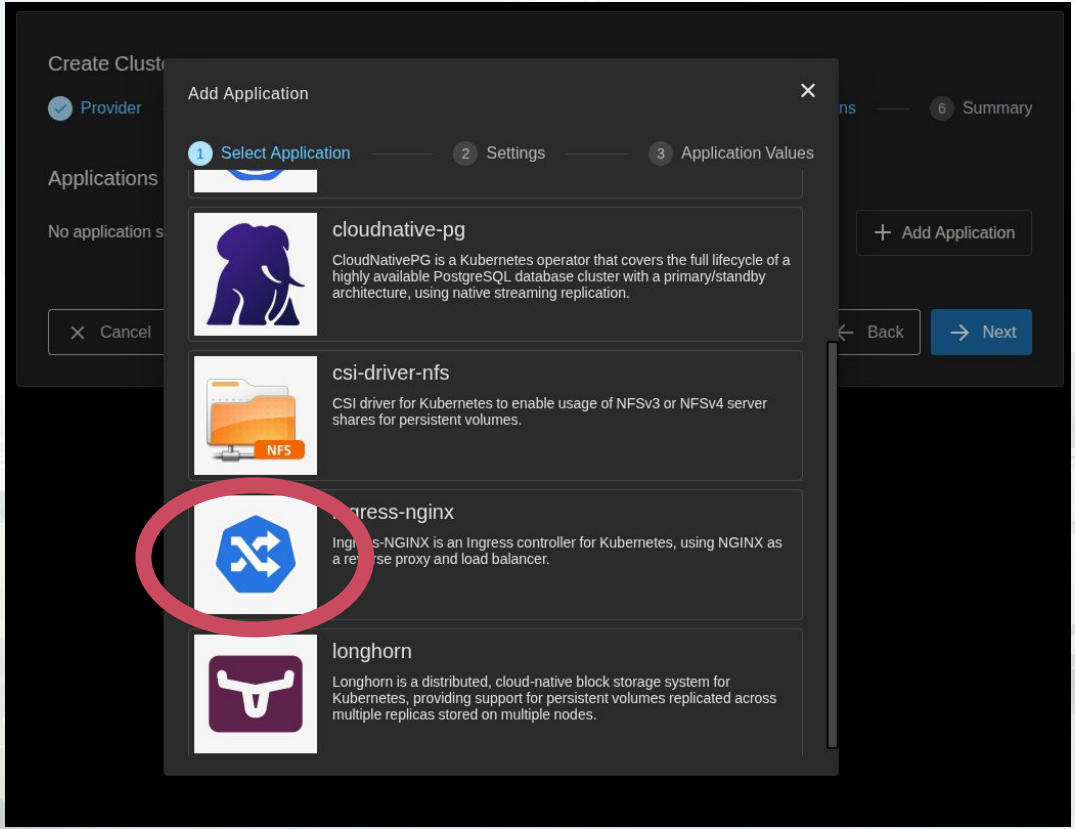
- Add Nginx



01 – Create the Kubernetes Cluster

Applications

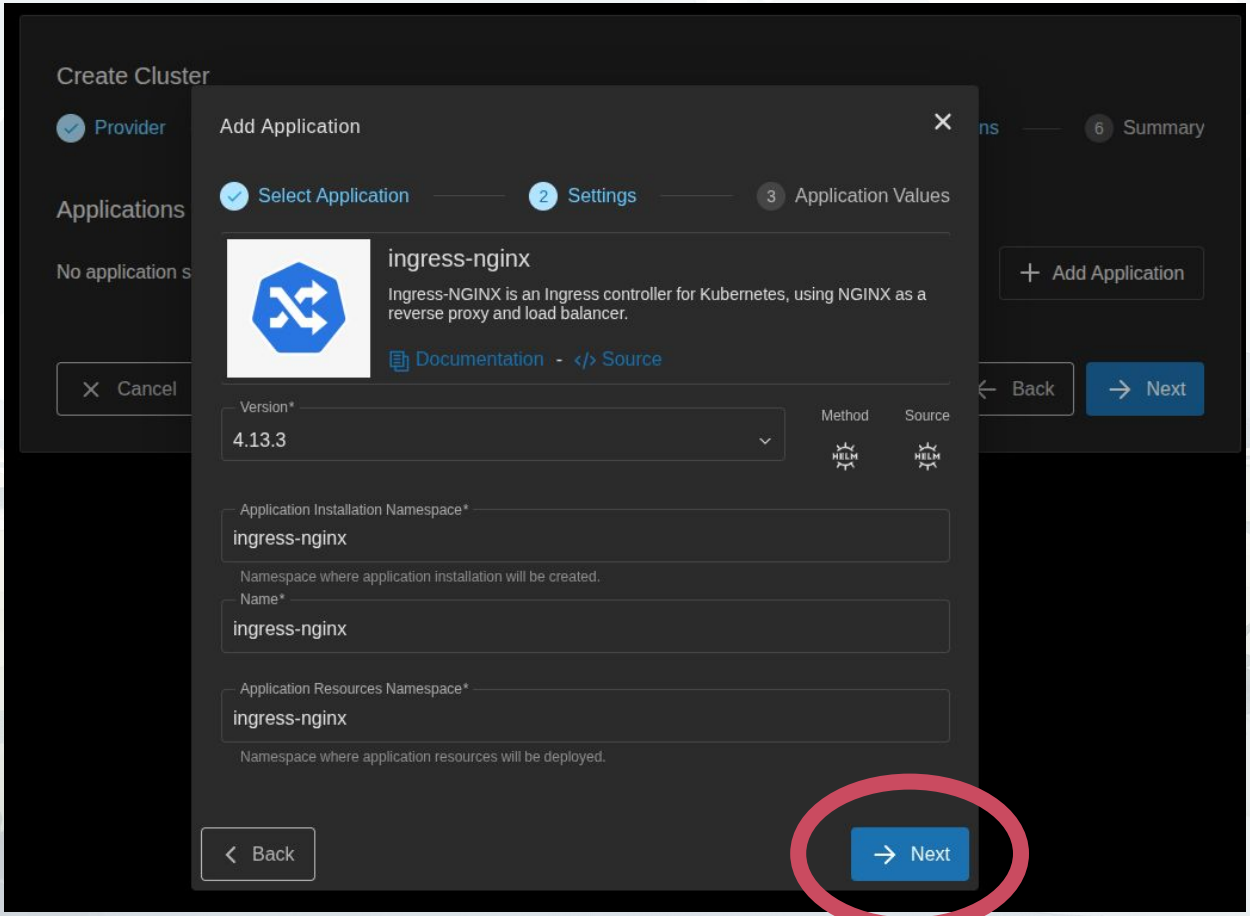
- Add Nginx



01 – Create the Kubernetes Cluster

Applications

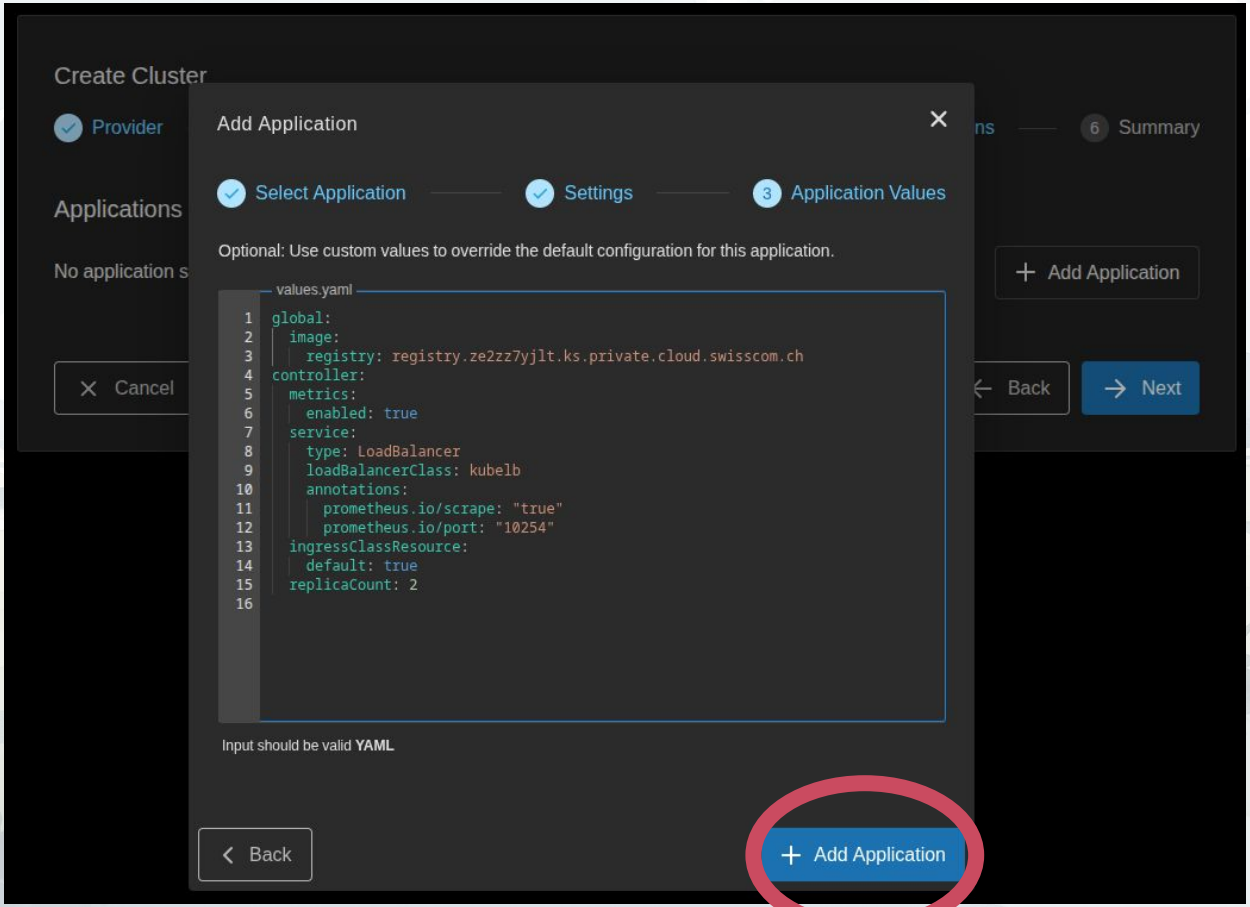
- Add Nginx



01 – Create the Kubernetes Cluster

Applications

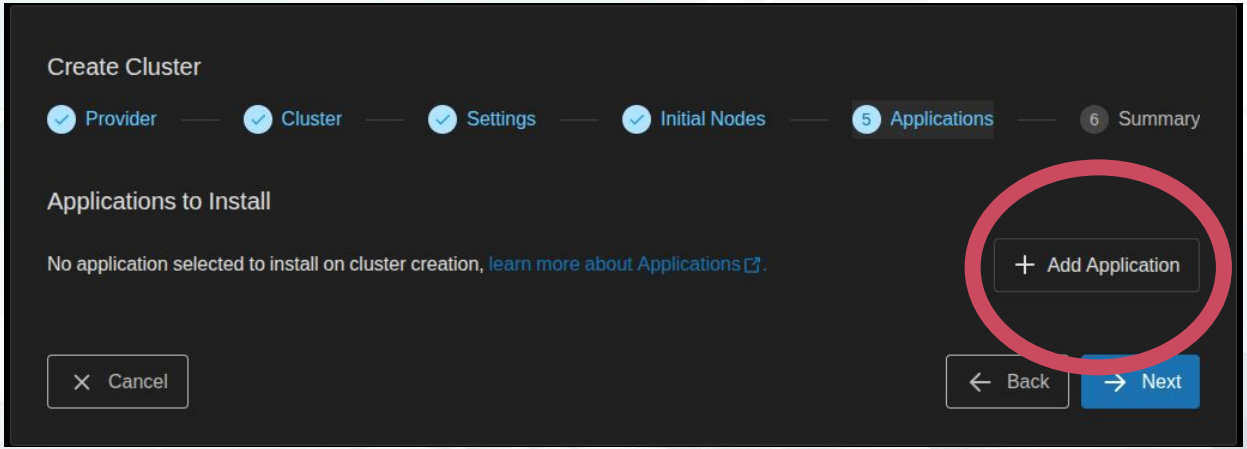
- Add Nginx



01 – Create the Kubernetes Cluster

Applications

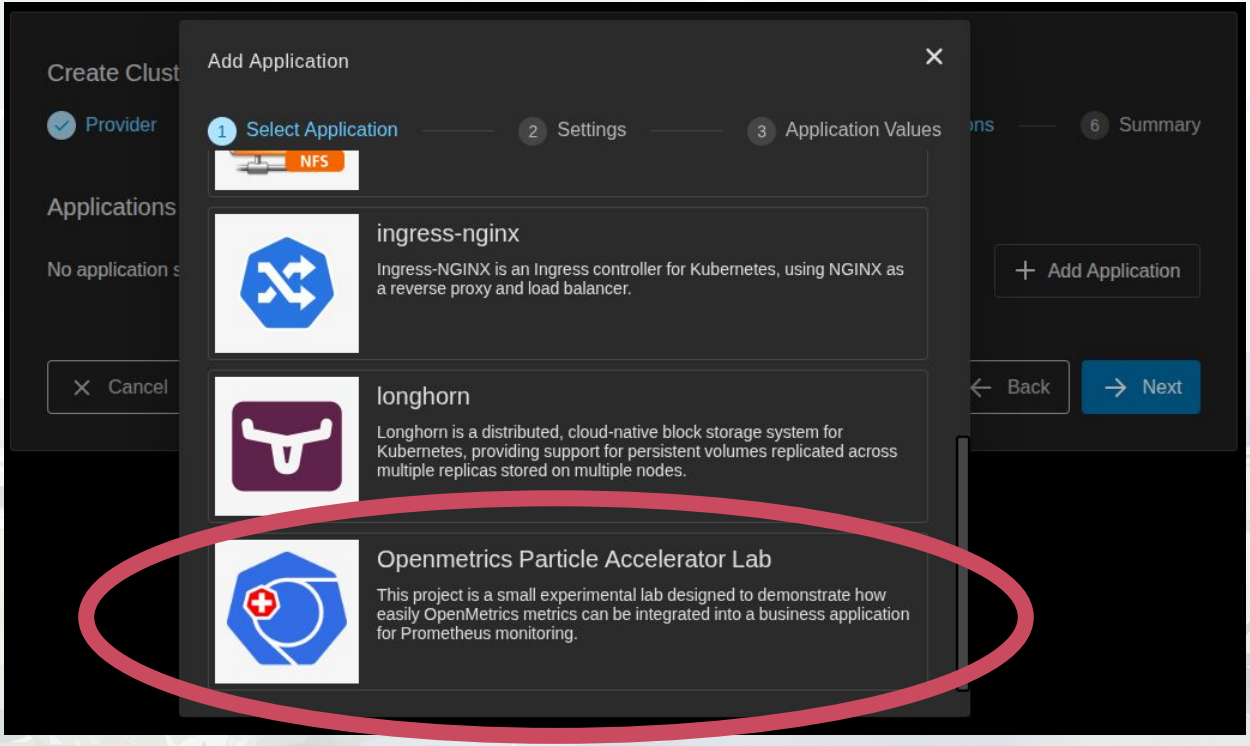
- Add Nginx
- Add Openmetrics
- Add Particle Accelerator Lab



01 – Create the Kubernetes Cluster

Applications

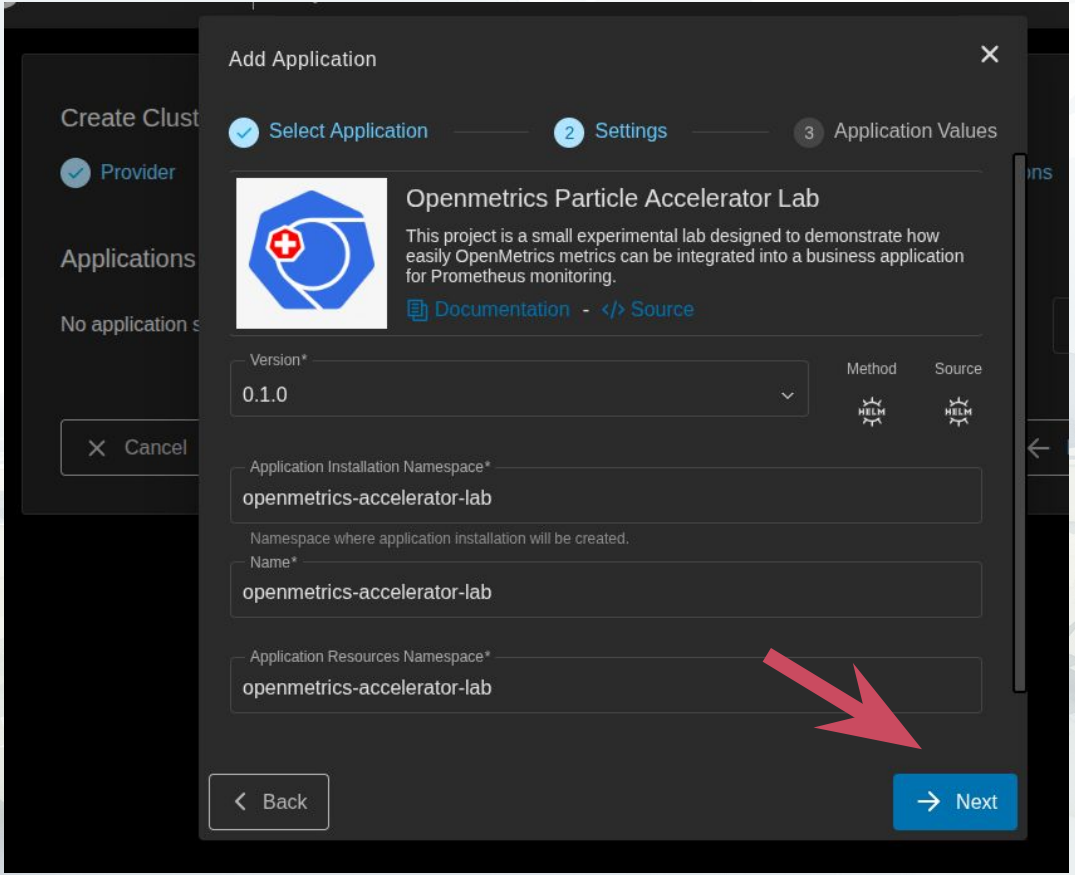
- Add Nginx
- Add Openmetrics Particle Accelerator Lab



01 – Create the Kubernetes Cluster

Applications

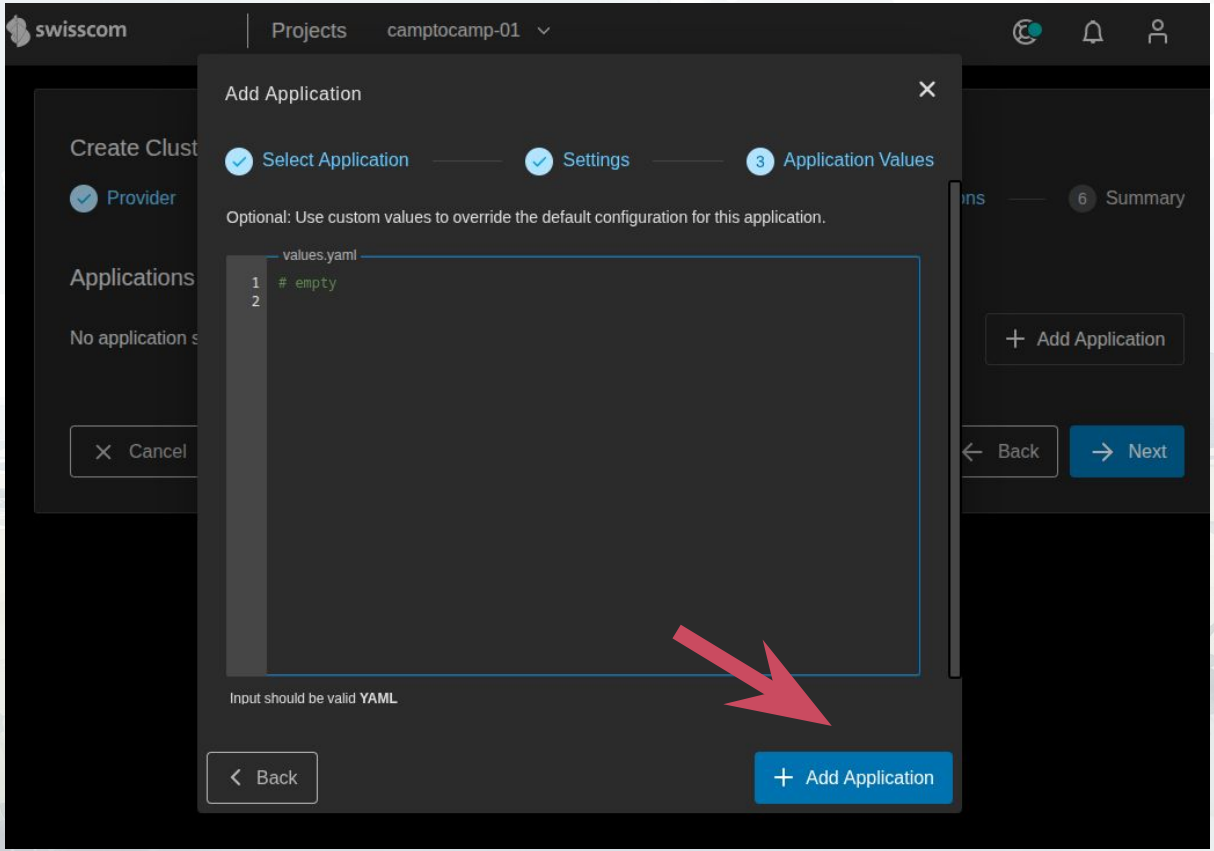
- Add Nginx
- Add Openmetrics Particle Accelerator Lab



01 – Create the Kubernetes Cluster

Applications

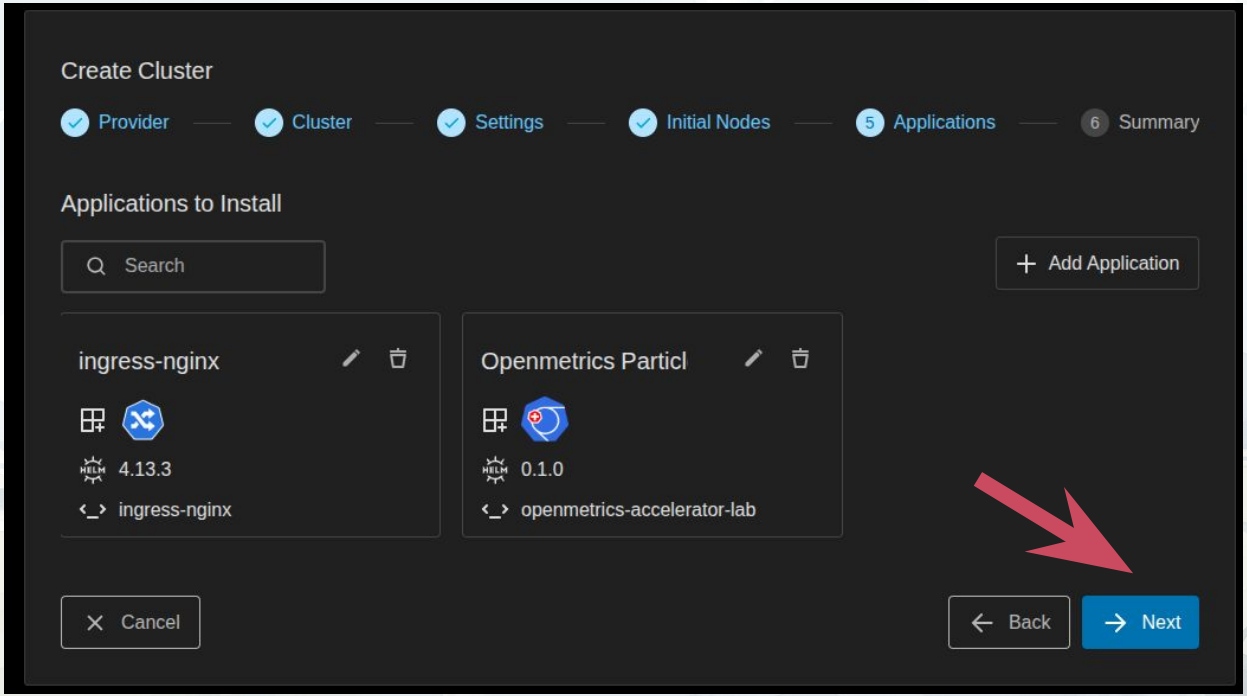
- Add Nginx
- Add Openmetrics
- Add Particle Accelerator Lab



01 – Create the Kubernetes Cluster

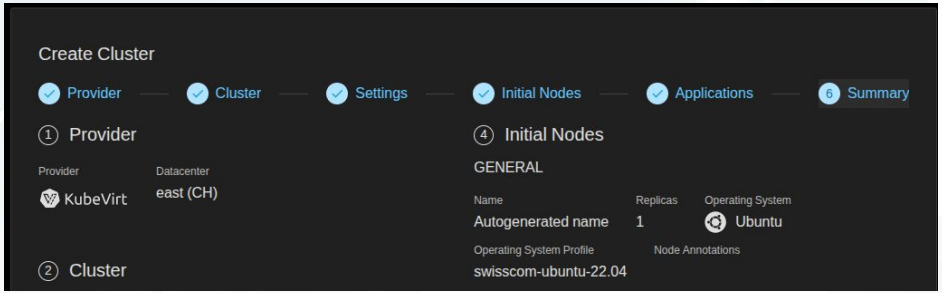
Applications

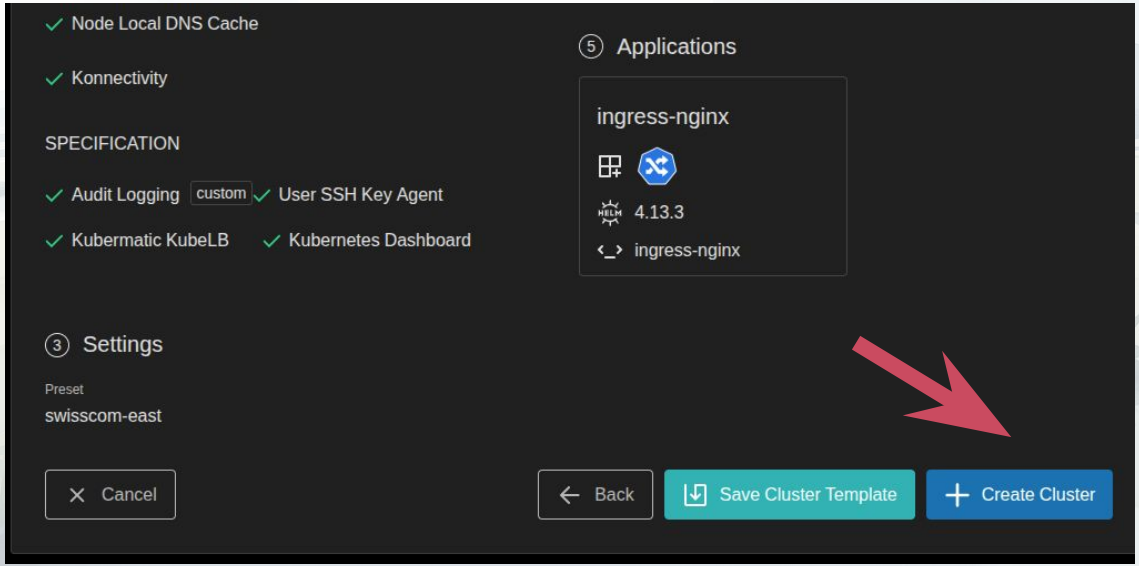
- Add Nginx
- Add Openmetrics Particle Accelerator Lab



01 – Create the Kubernetes Cluster

Summary

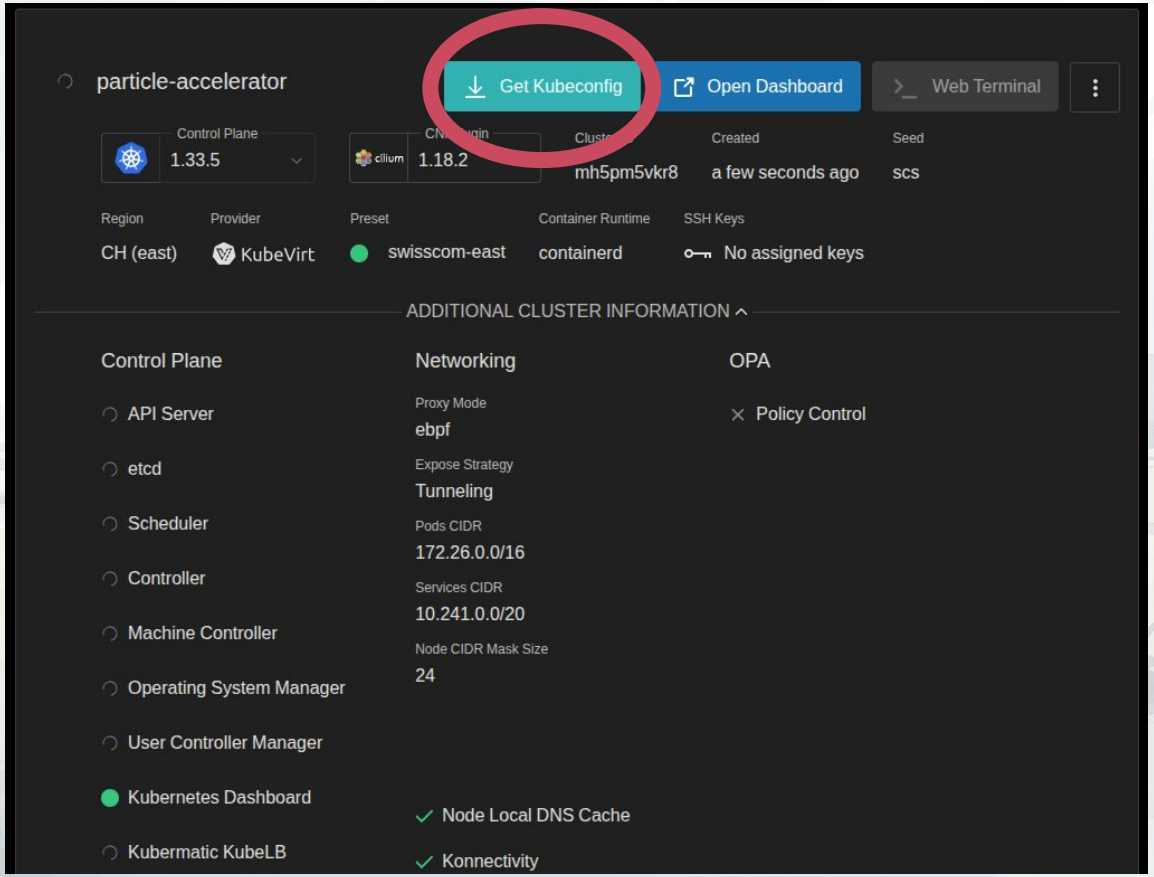




01 – Create the Kubernetes Cluster

Get Kubeconfig

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



01 – Create the Kubernetes Cluster

Summary

Instructions on lab.campto.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 – Install tools

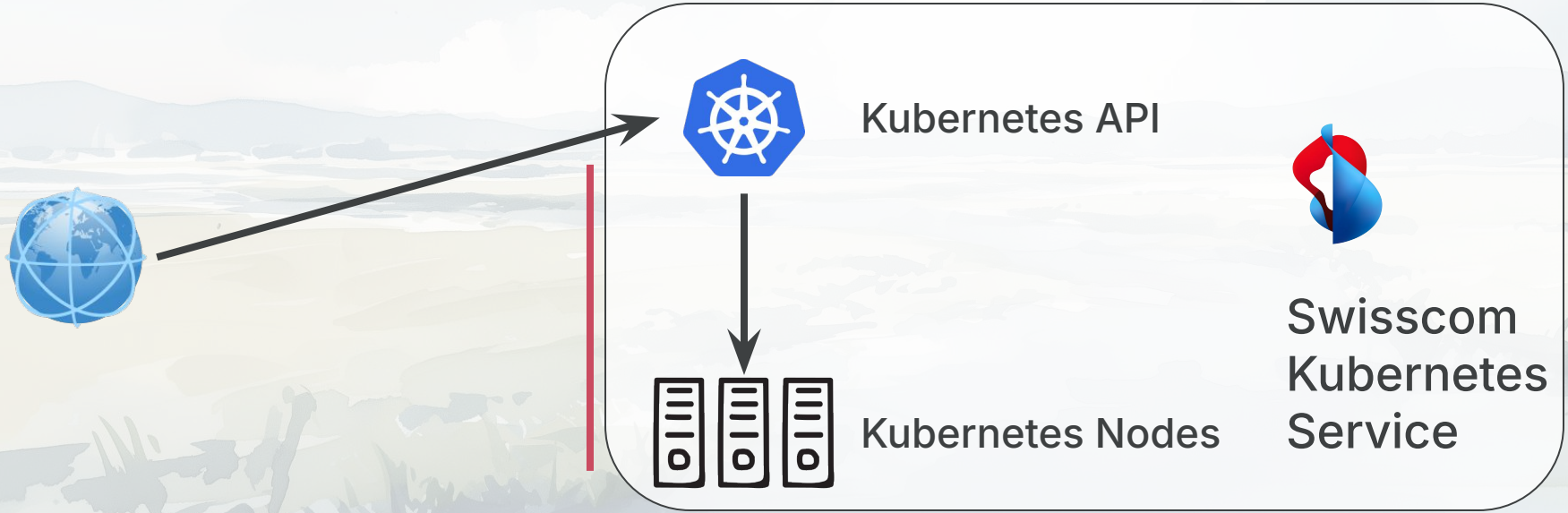
kubectl

- Cluster nodes are not reachable

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:

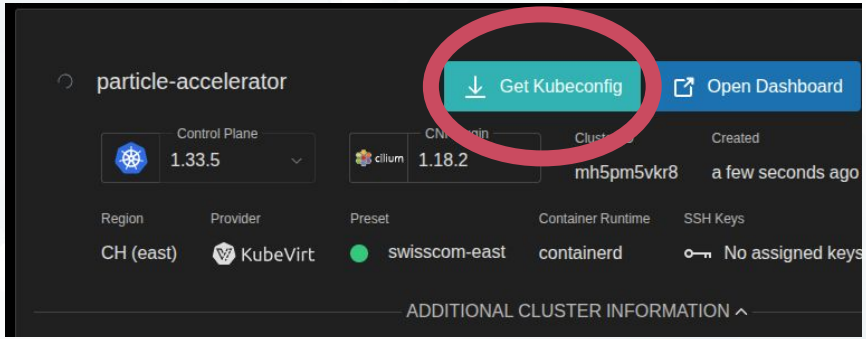
```
curl.exe -LO "https://dl.k8s.io/release/v1.34.2/bin/windows/amd64/kubectl.exe"
```

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

02 – Install tools

Configure API Access

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




```
# Linux/Mac
export KUBECONFIG=~/Downloads/kubeconfig-xxxxxxxxxx


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

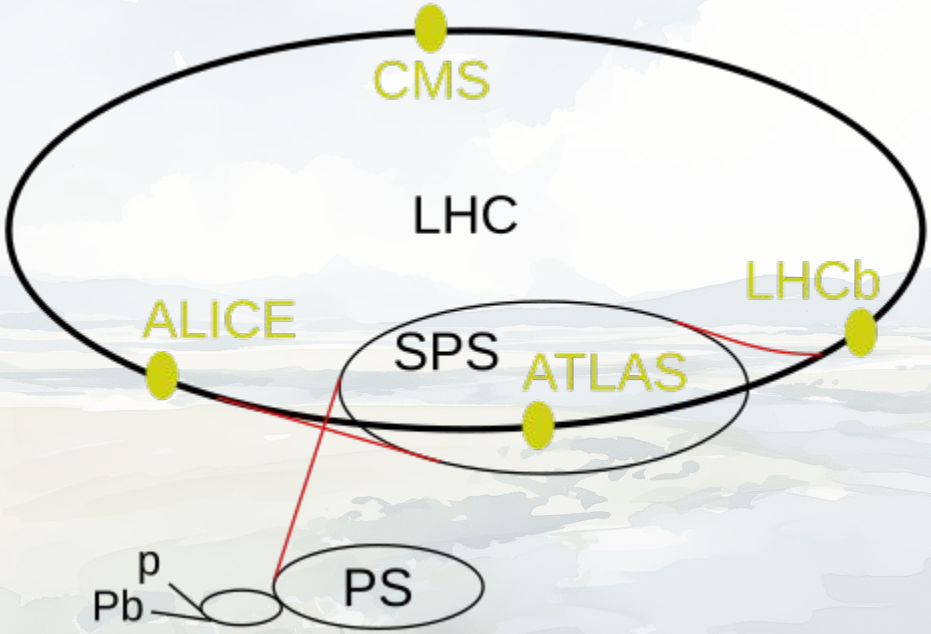
02 – Install tools

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     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
...
```

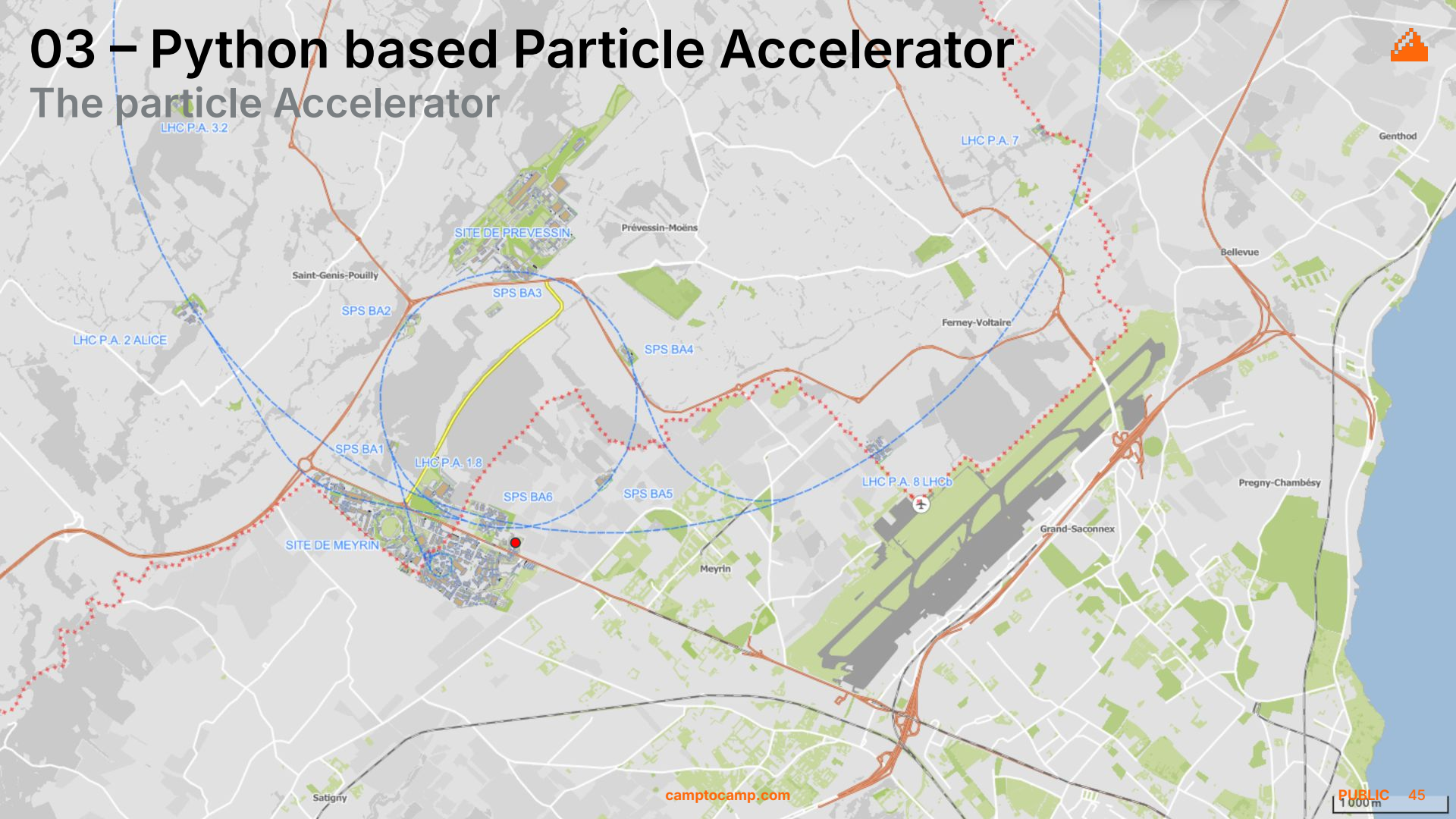
03 – Python based Particle Accelerator

The particle Accelerator



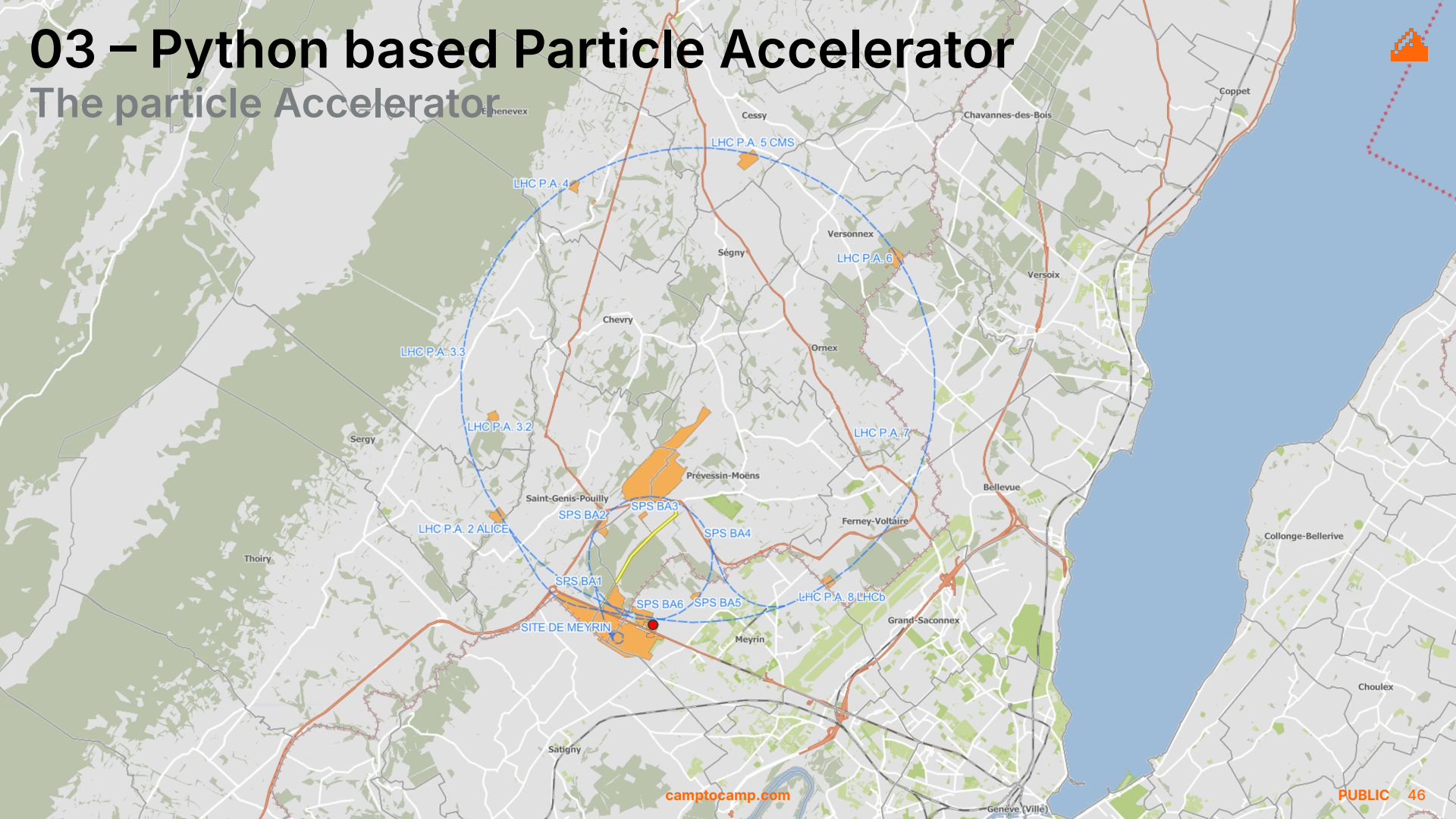
03 – Python based Particle Accelerator

The particle Accelerator



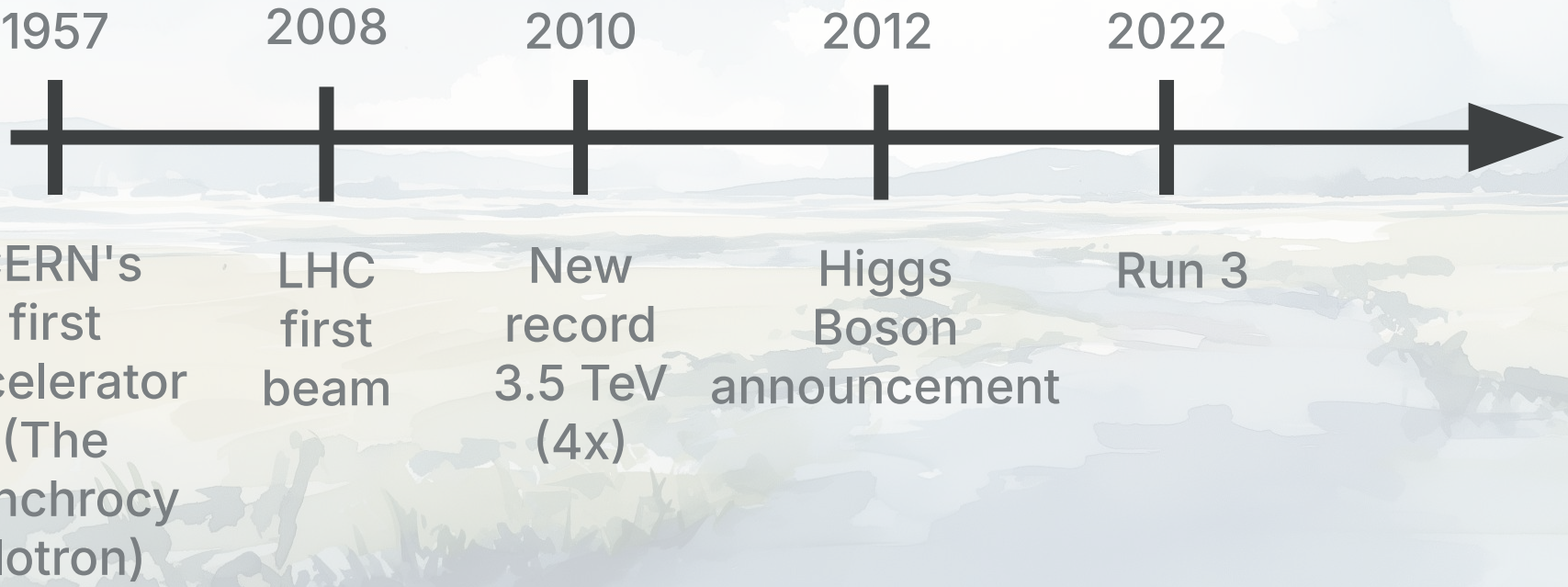
03 – Python based Particle Accelerator

The particle Accelerator

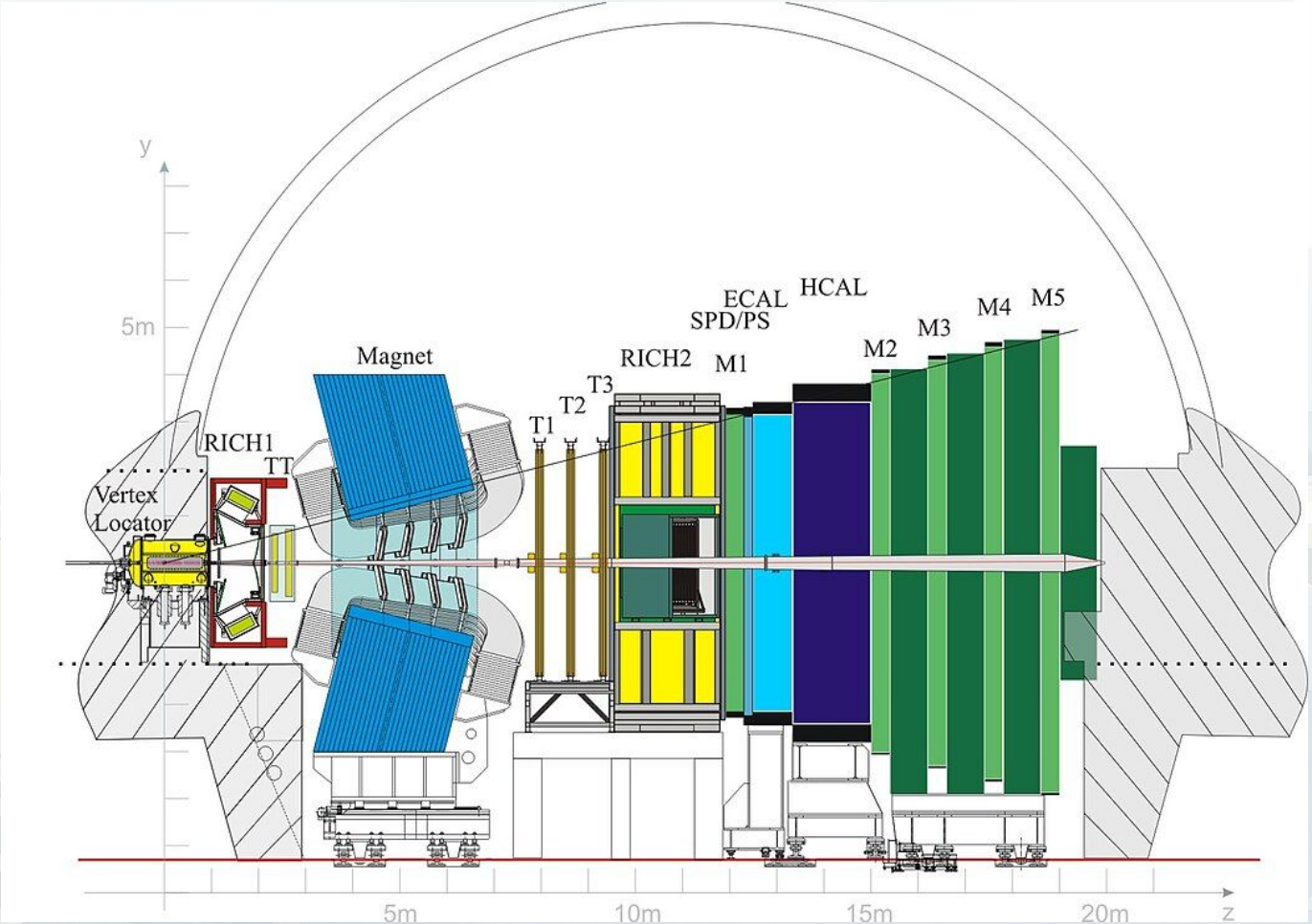


03 – Python based Particle Accelerator

The particle Accelerator

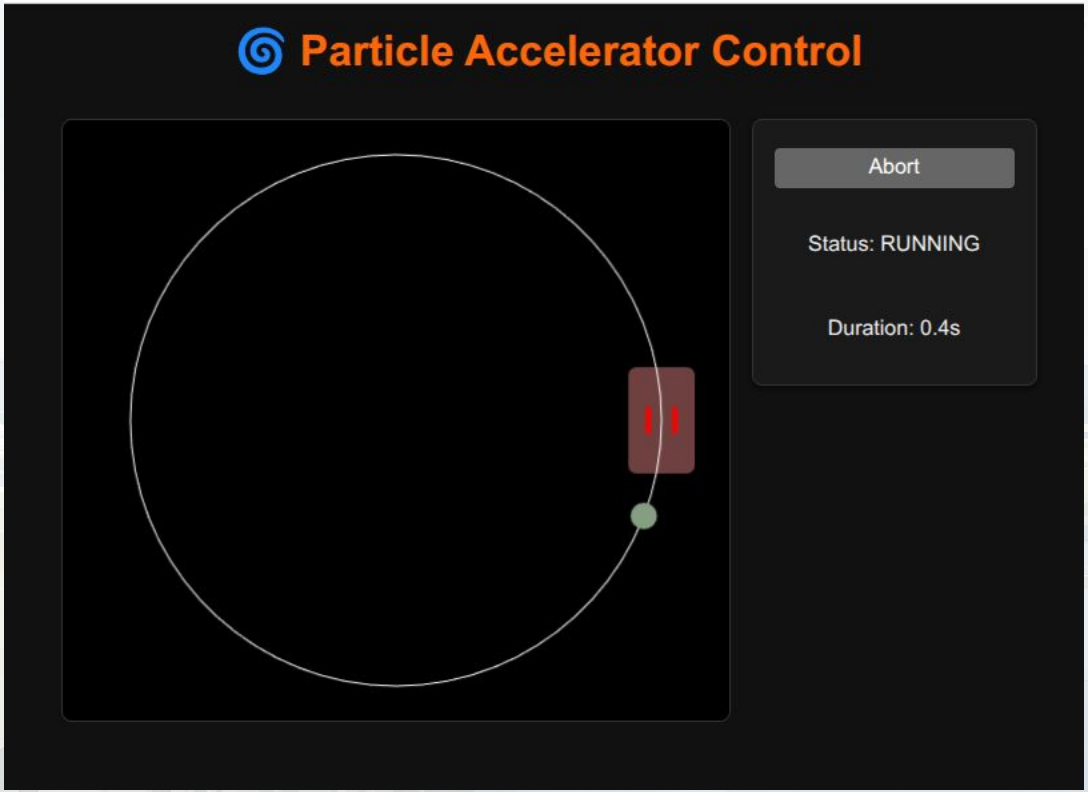


03 – Python based Particle Accelerator



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 – Deploy 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 – Deploy 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 – Deploy the lab

The Feedback Loop

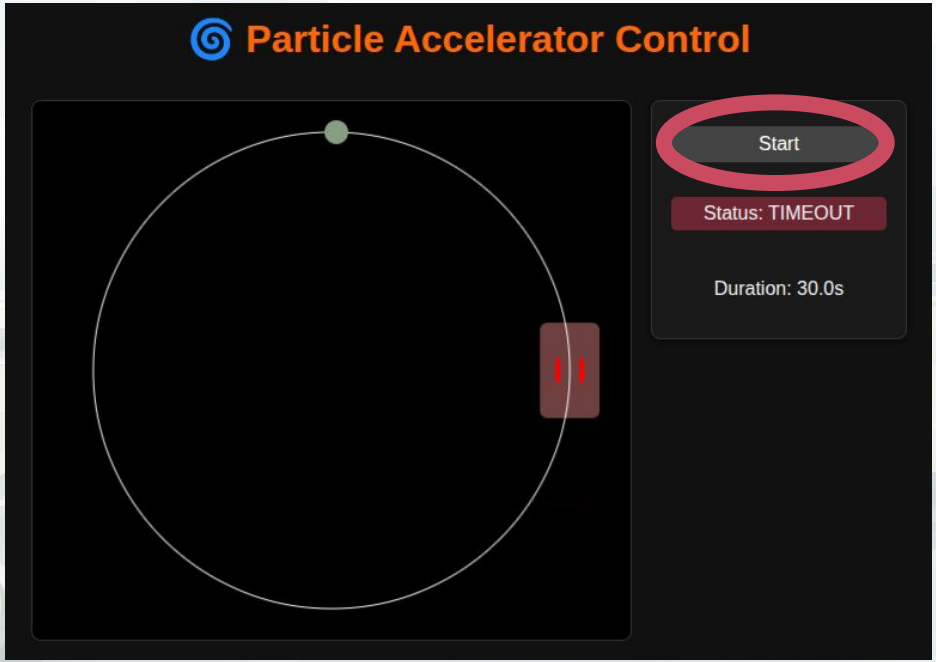
- Launch simulator
- Watch Grafana Dashboards.
- Change Code in Browser (VSCode)
- Save (Auto-reload).
- React to the data.

04 – Deploy the lab

The Feedback Loop

- Launch simulator

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



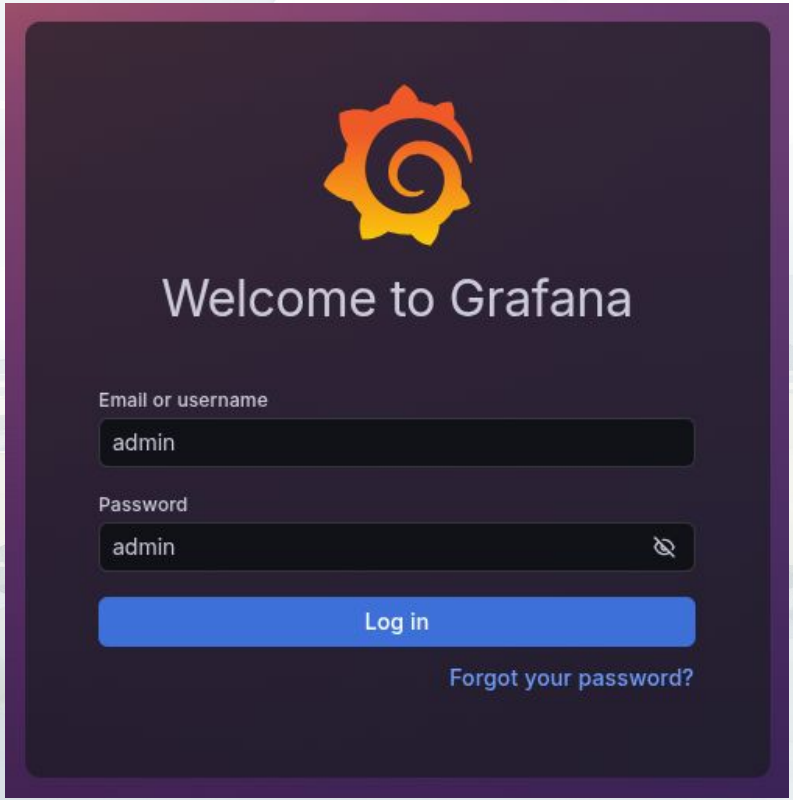
Simulation pauses when the tab is not visible.

04 – Deploy the lab

The Feedback Loop

- Launch simulator
- Watch Grafana

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

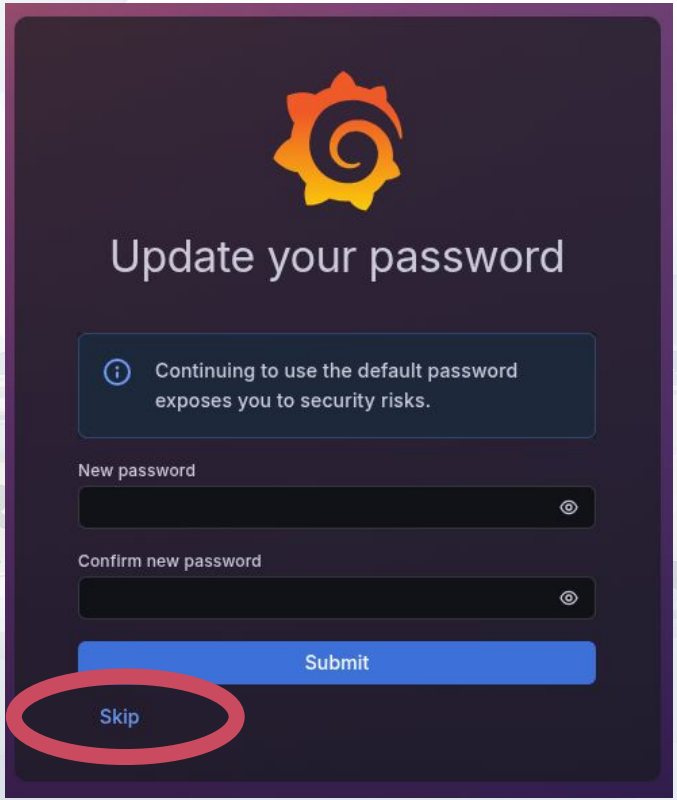


04 – Deploy the lab

The Feedback Loop

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

- Launch simulator
- Watch Grafana

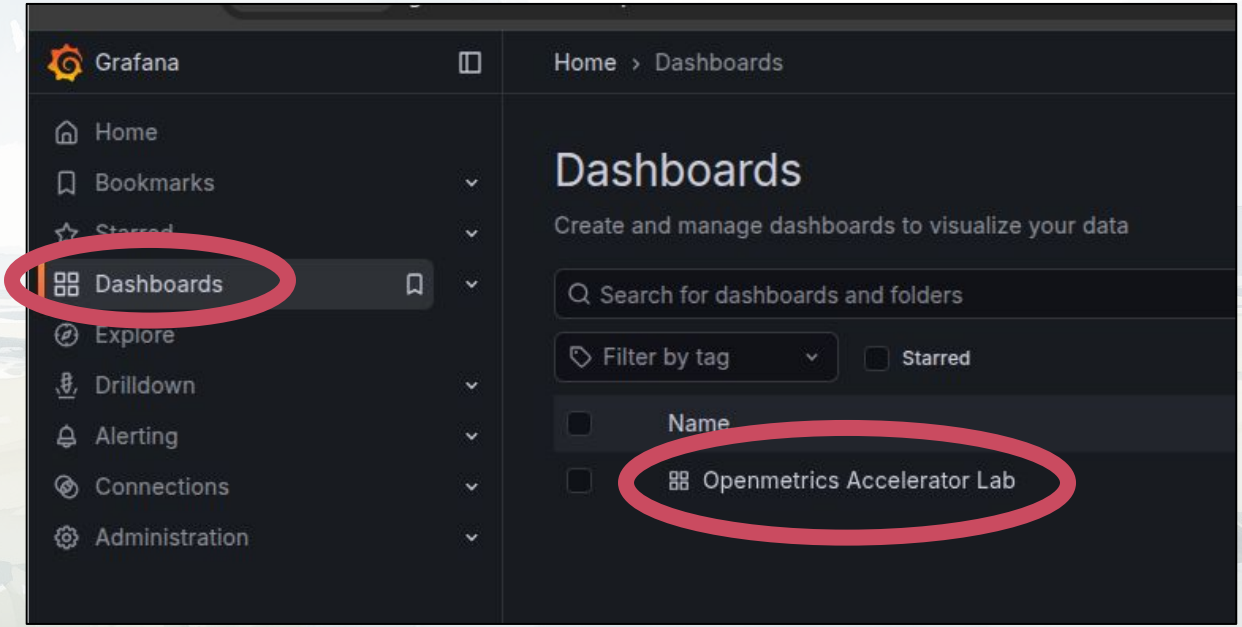


04 – Deploy the lab

The Feedback Loop

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

- Launch simulator
- Watch Grafana

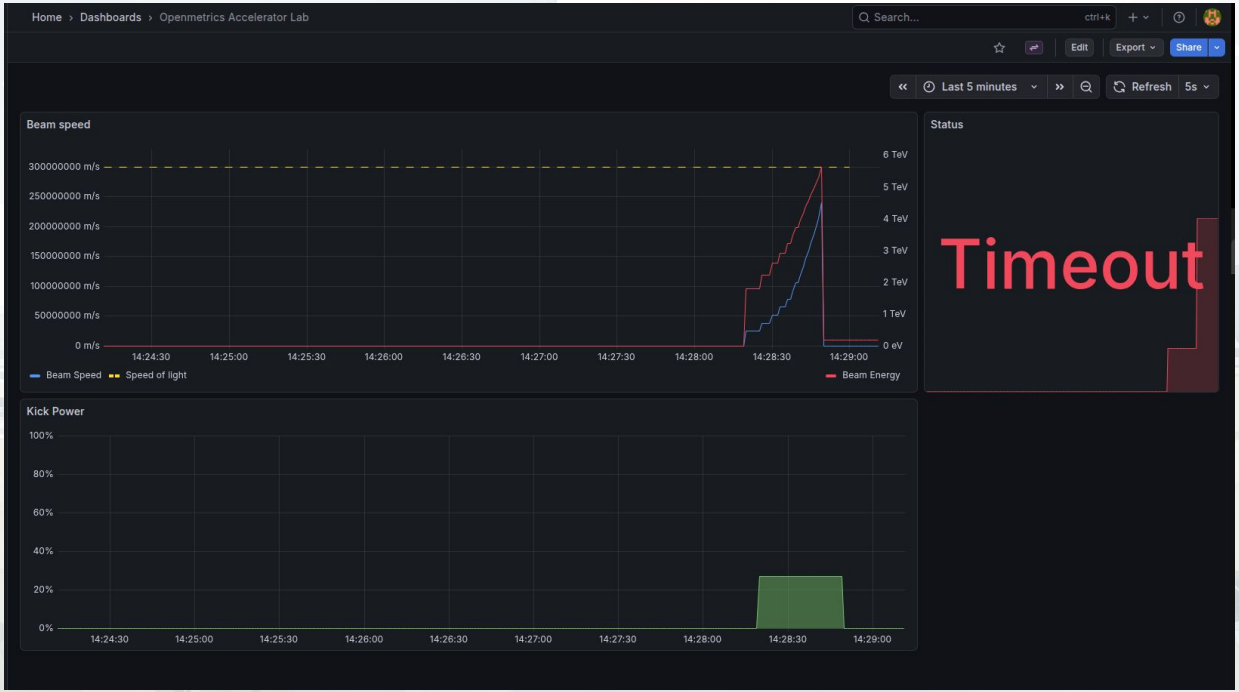


04 – Deploy the lab

The Feedback Loop

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

- Launch simulator
- Watch Grafana

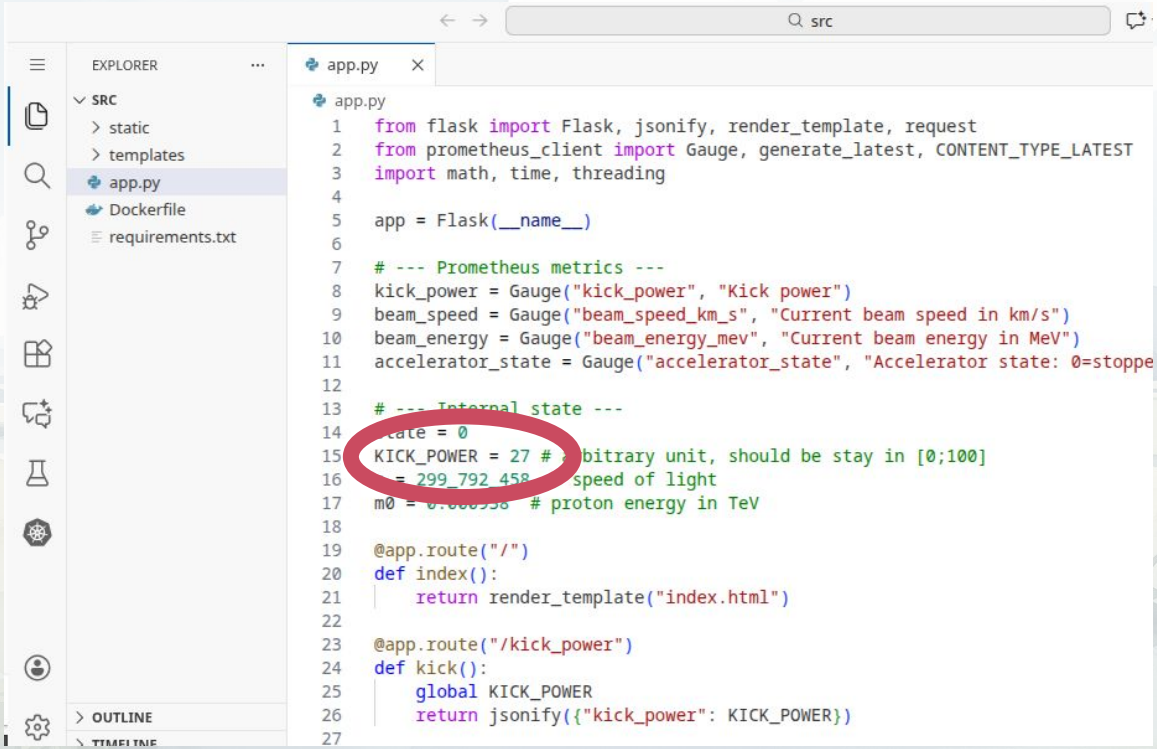


04 – Deploy the lab

The Feedback Loop

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

- Launch simulator
- Watch Grafana
- Change Code



04 – Deploy the lab

Command Center

The particle view http://lab.127.0.0.1.nip.io:8080/

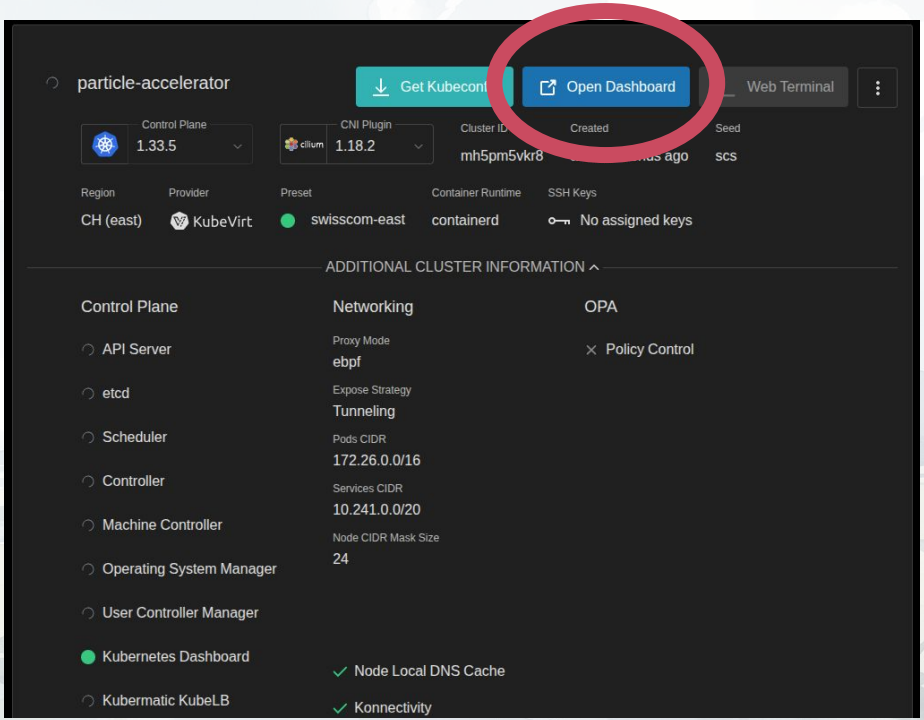
The Data View (Grafana) http://grafana.127.0.0.1.nip.io:8080/

The Code View (VSCode) http://vscode.127.0.0.1.nip.io:8080/

04 – Deploy the lab

Restart Lab

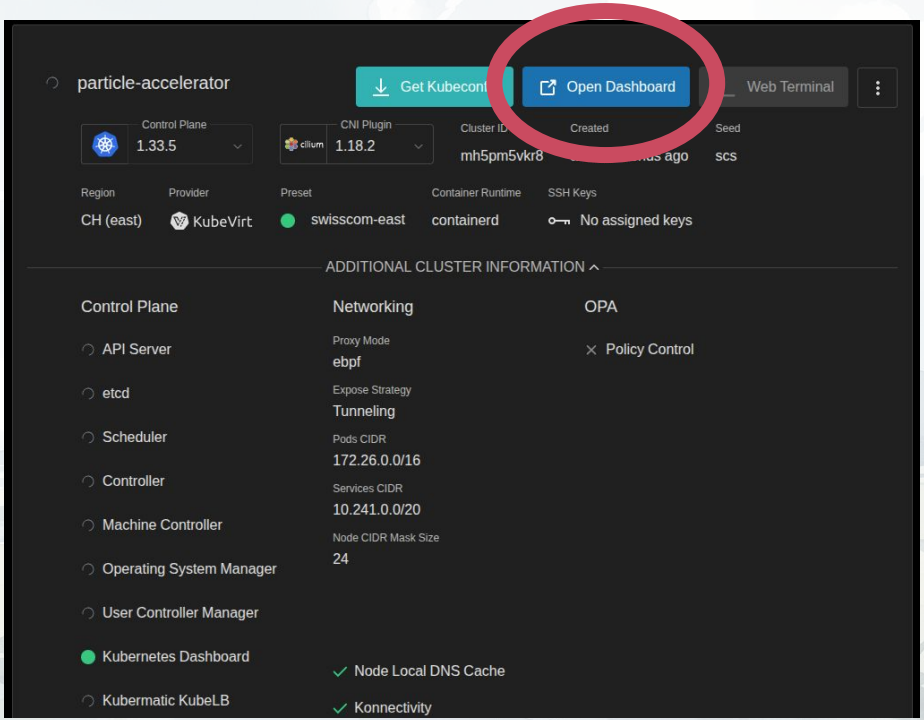
- Open Dashboard
- Select Deployments (left panel)
- Select 'openmetrics...' namespace (top drop list)
- Select 'openmetrics-accelerator-lab-xxx-lab'
- Click on restart



04 – Deploy the lab

Application observability

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



04 – Deploy the lab

Extra step – Implements a new metrics

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

```python
from prometheus_client import Counter

kick_count = Counter('kick_count', 'Kick Count')

@app.route('/kick_power')
def kick():
    ...
    kick_count.inc()
    ...
```

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

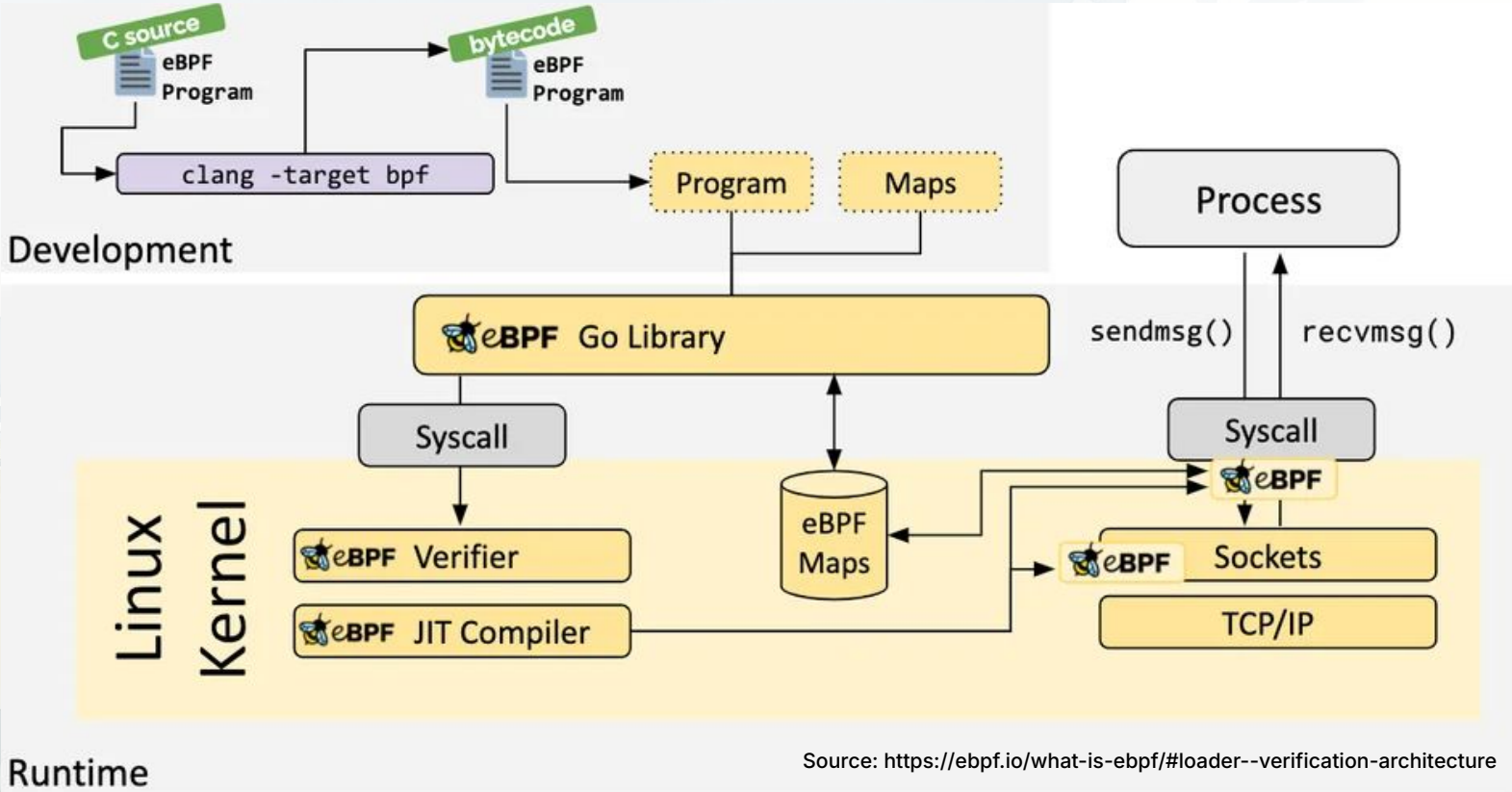
05 – Observability tools using EBPF

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



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
```

camptocamp

Federico Sismondi
federico.sismondi@camptocamp.com

Julien Acroute
julien.acroute@camptocamp.com

Thanks for your attention.