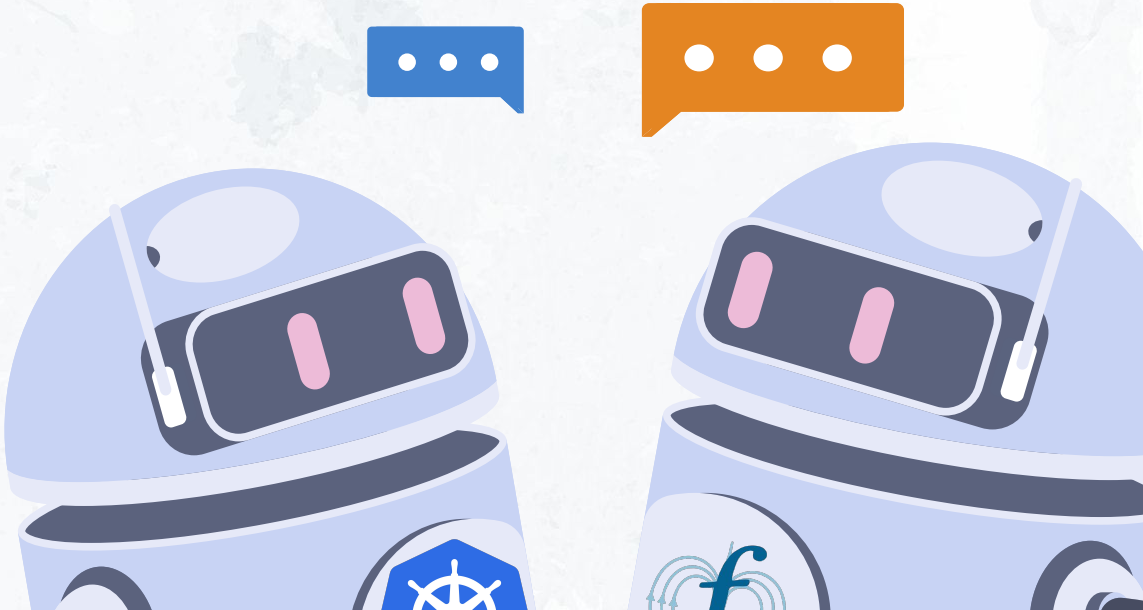
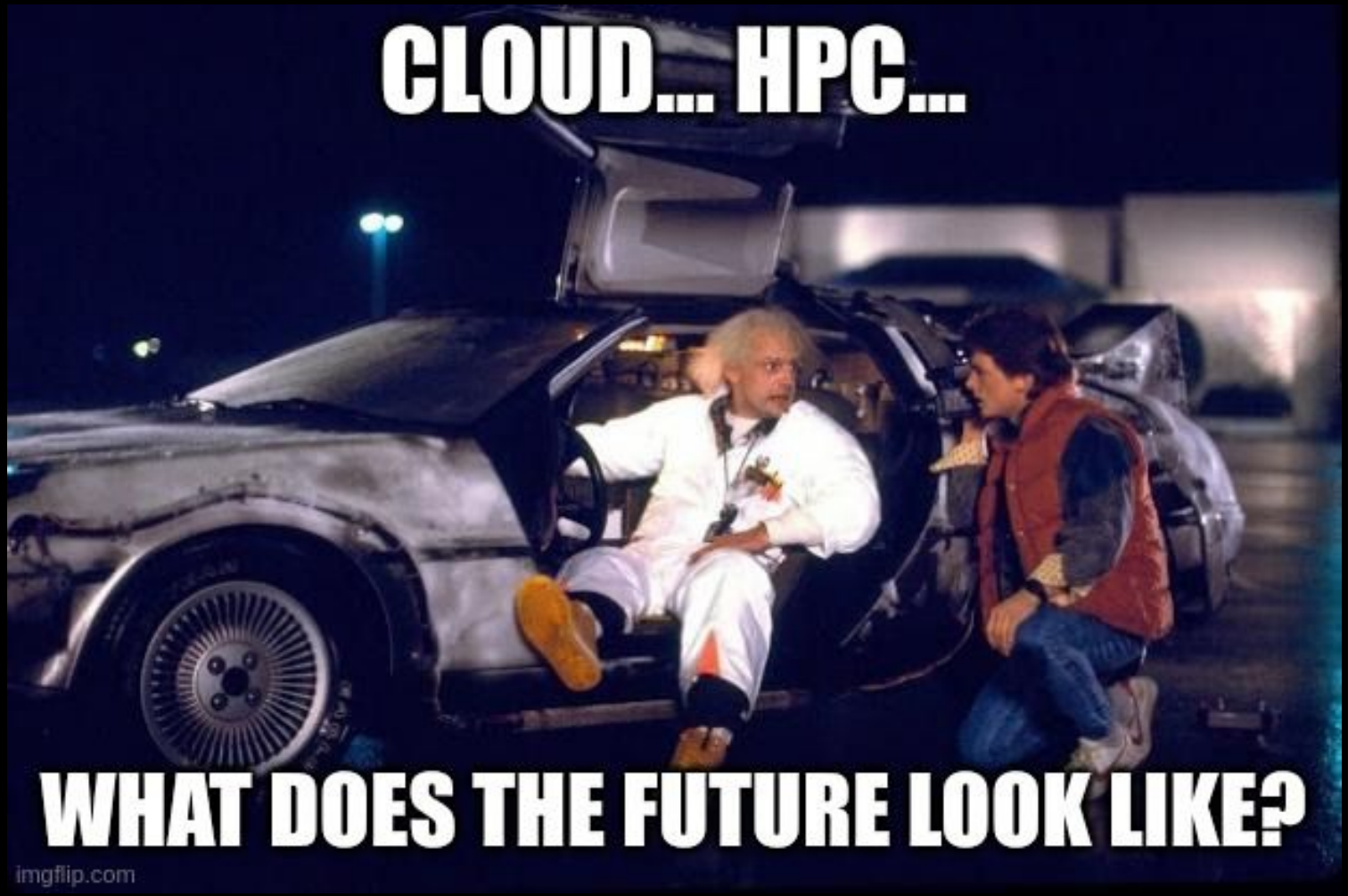
Kubernetes and HPC

Bare Metal Bros



Vanessa Sochat, Dave Fox, Daniel Milroy
Lawrence Livermore National Laboratory
LLNL-PRES-859106

Cloud HPC


CLOUD... HPC...
WHAT DOES THE FUTURE LOOK LIKE?
imgflip.com

Where is the money going?

Where is the money going?

Cloud: Projected to reach >$1.1 trillion revenue by 2027, 20% CAGR[1]

HPC: Projected to reach $40 billion revenue by 2026, 6.4% CAGR[2]

CAGR: "Compound Annual Growth Rate"

[1]Gartner February 2022 Report,
[2]Hyperion Research ISC Breakfast Briefing, 2023

Where is the money going?

Cloud: Projected to reach >$1.1 trillion revenue by 2027, 20% CAGR[1]

HPC: Projected to reach $40 billion revenue by 2026, 6.4% CAGR[2]

CAGR: "Compound Annual Growth Rate"

[1]Gartner February 2022 Report,
[2]Hyperion Research ISC Breakfast Briefing, 2023

Who gets left behind?

The Decline of Computers as a General Purpose Technology, CACM March 2021[1]
HPC Forecast: Cloudy and Uncertain, CACM February 2023[2]

Who gets left behind?

Reed, Gannon, Dongarra 2023[2] identified trends:

The Decline of Computers as a General Purpose Technology, CACM March 2021[1]
HPC Forecast: Cloudy and Uncertain, CACM February 2023[2]

Who gets left behind?

Reed, Gannon, Dongarra 2023[2] identified trends:

- **HPC: The way we design our systems won't continue to work**
  - We can't depend on Dennard scaling and Moore's law
  - Rising costs for improved semiconductors make it harder
  - Increasingly more expensive and laborious to deploy new systems
  - Nonrecurring engineering costs (NREs) for every new system

The Decline of Computers as a General Purpose Technology, CACM March 2021[1]
HPC Forecast: Cloudy and Uncertain, CACM February 2023[2]

Who gets left behind?

Reed, Gannon, Dongarra 2023[2] identified trends:

- **HPC: The way we design our systems won't continue to work**
  - We can't depend on Dennard scaling and Moore's law
  - Rising costs for improved semiconductors make it harder
  - Increasingly more expensive and laborious to deploy new systems
  - Nonrecurring engineering costs (NREs) for every new system

- **Cloud: Leading the space of innovation**
  - Massive expansion of large-scale, commercial clouds
  - Much less dependence on computing vendors
  - Deploying own software / hardware at scale, are cash rich
  - Easily attracting the talent pool

The Decline of Computers as a General Purpose Technology, CACM March 2021[1]
HPC Forecast: Cloudy and Uncertain, CACM February 2023[2]

Who gets left behind?

Reed, Gannon, Dongarra 2023[2] identified trends:

- **HPC: The way we design our systems won't continue to work**

  "endothermic" : requiring absorption of heat

- **Cloud: Leading the space of innovation**

  "exothermic" : accompanied by the release of heat

The Decline of Computers as a General Purpose Technology, CACM March 2021[1]
HPC Forecast: Cloudy and Uncertain, CACM February 2023[2]

Who gets left behind?

Reed, Gannon, Dongarra 2023[2] identified trends:

HPC

- **HPC: The way we design our systems won't continue to work**

  "endothermic" : requiring absorption of heat

- **Cloud: Leading the space of innovation**

  "exothermic" : accompanied by the release of heat

The Decline of Computers as a General Purpose Technology, CACM March 2021[1]
HPC Forecast: Cloudy and Uncertain, CACM February 2023[2]

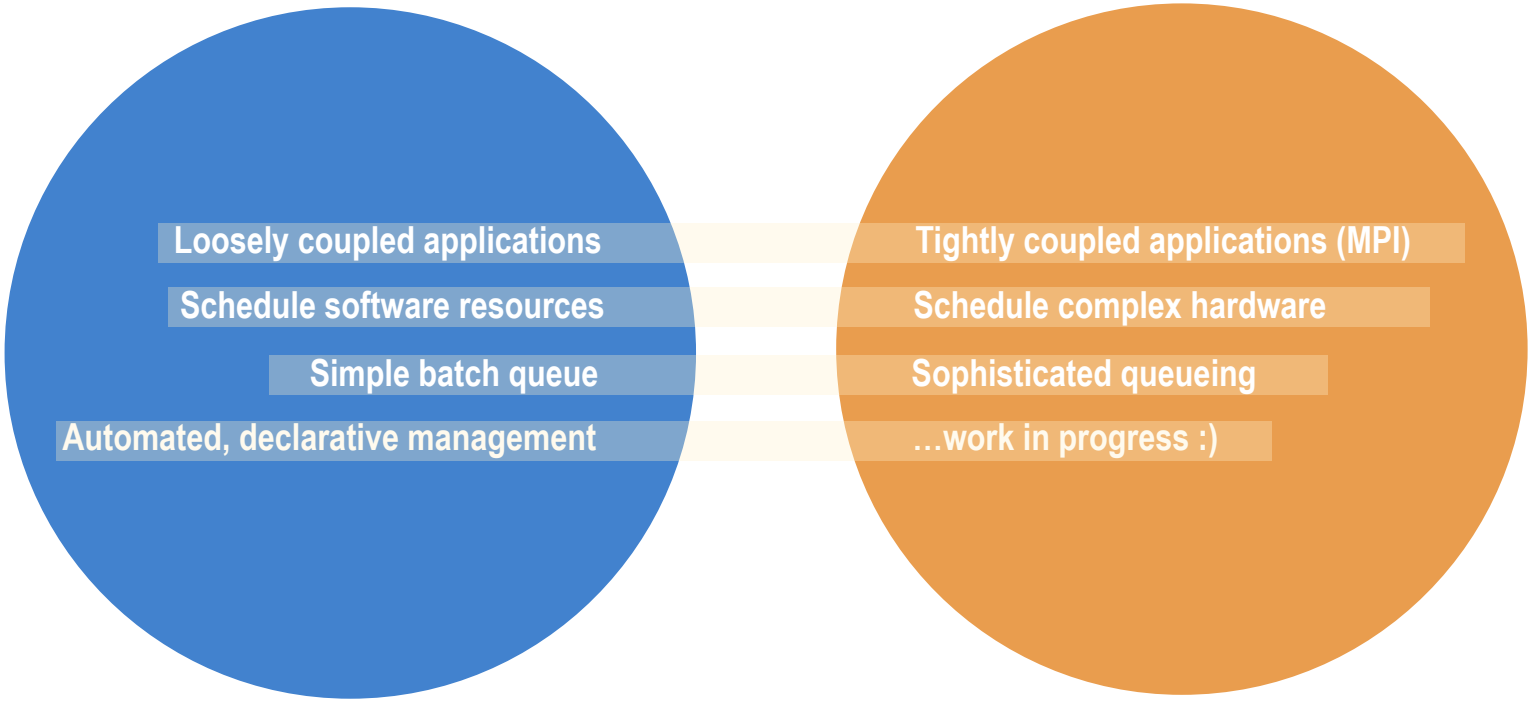
Converged Computing

is about ensuring that our needs are represented in this new environment.

The success of our science depends on our ability to be collaborative.

Cloud

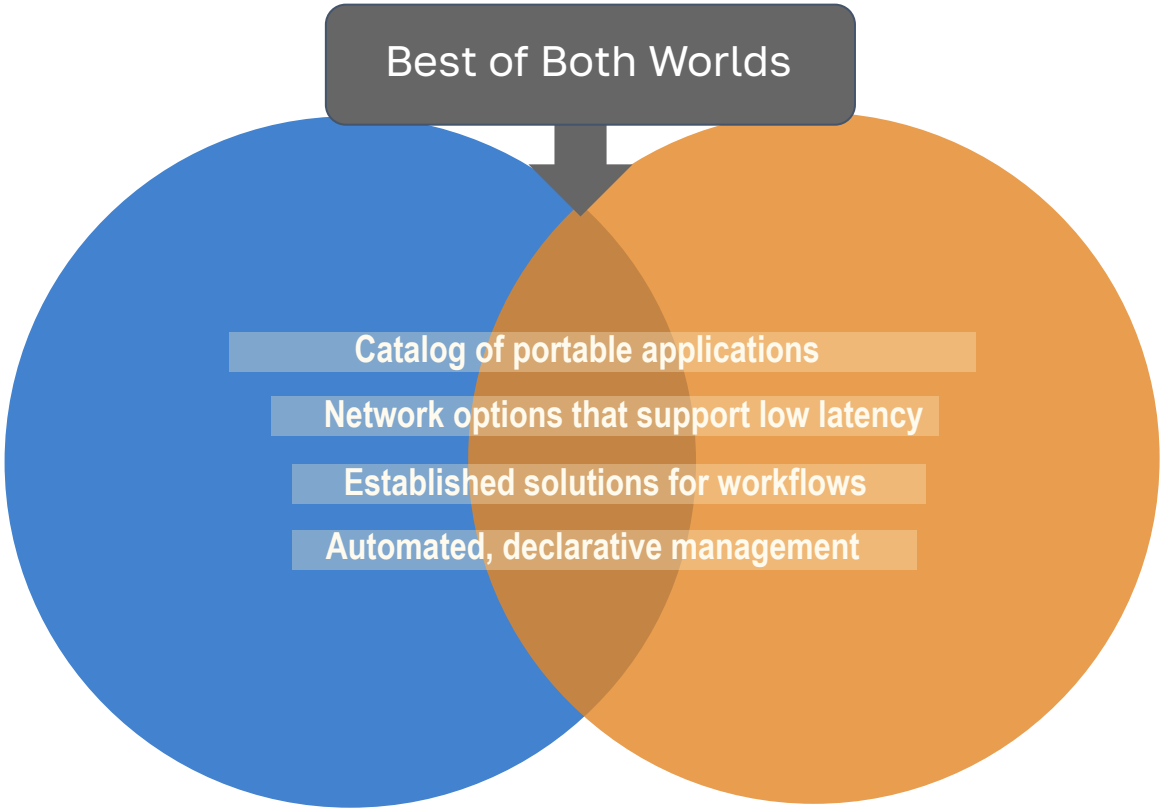
HPC



Converged Computing

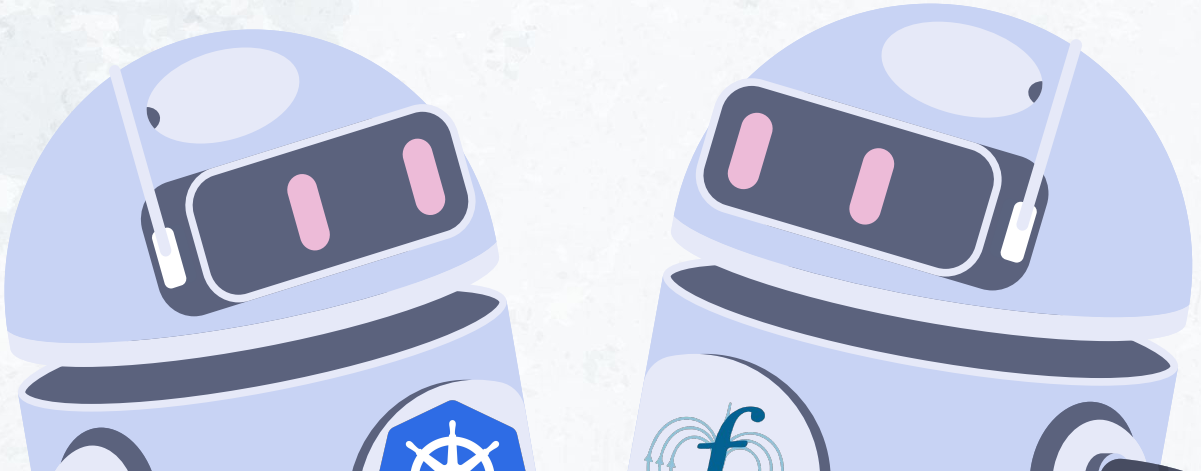
Cloud

HPC



Converged Computing

Where do we start?



What are we talking about today?

What are we talking about today?

(a) Models for Convergence

Patterns for bringing together disparate environments

What are we talking about today?

(a) Models for Convergence

Patterns for bringing together disparate environments

(b) Strategies for Convergence

Designs that allow for movement between spaces

What are we talking about today?

(a) Models for Convergence

Patterns for bringing together disparate environments

(b) Strategies for Convergence

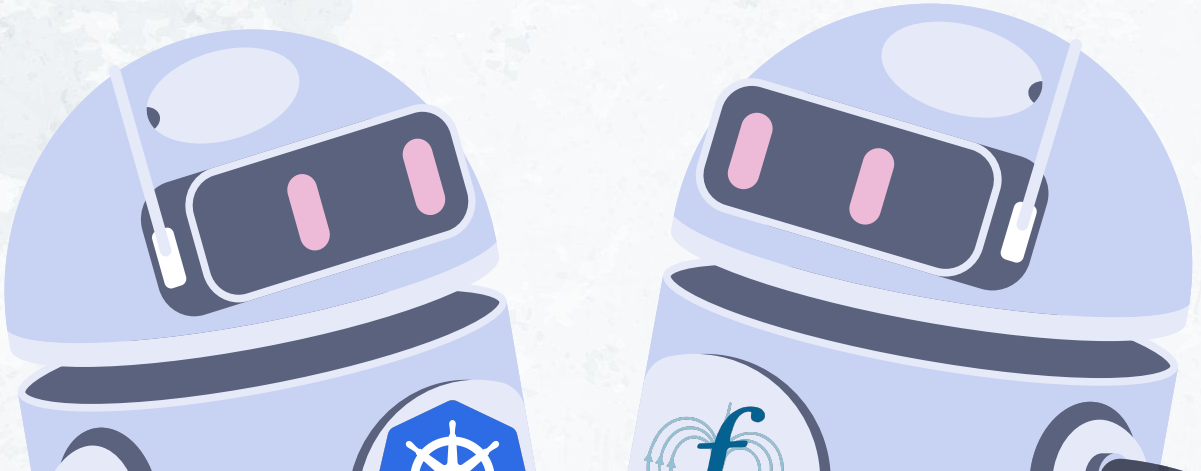
Designs that allow for movement between spaces

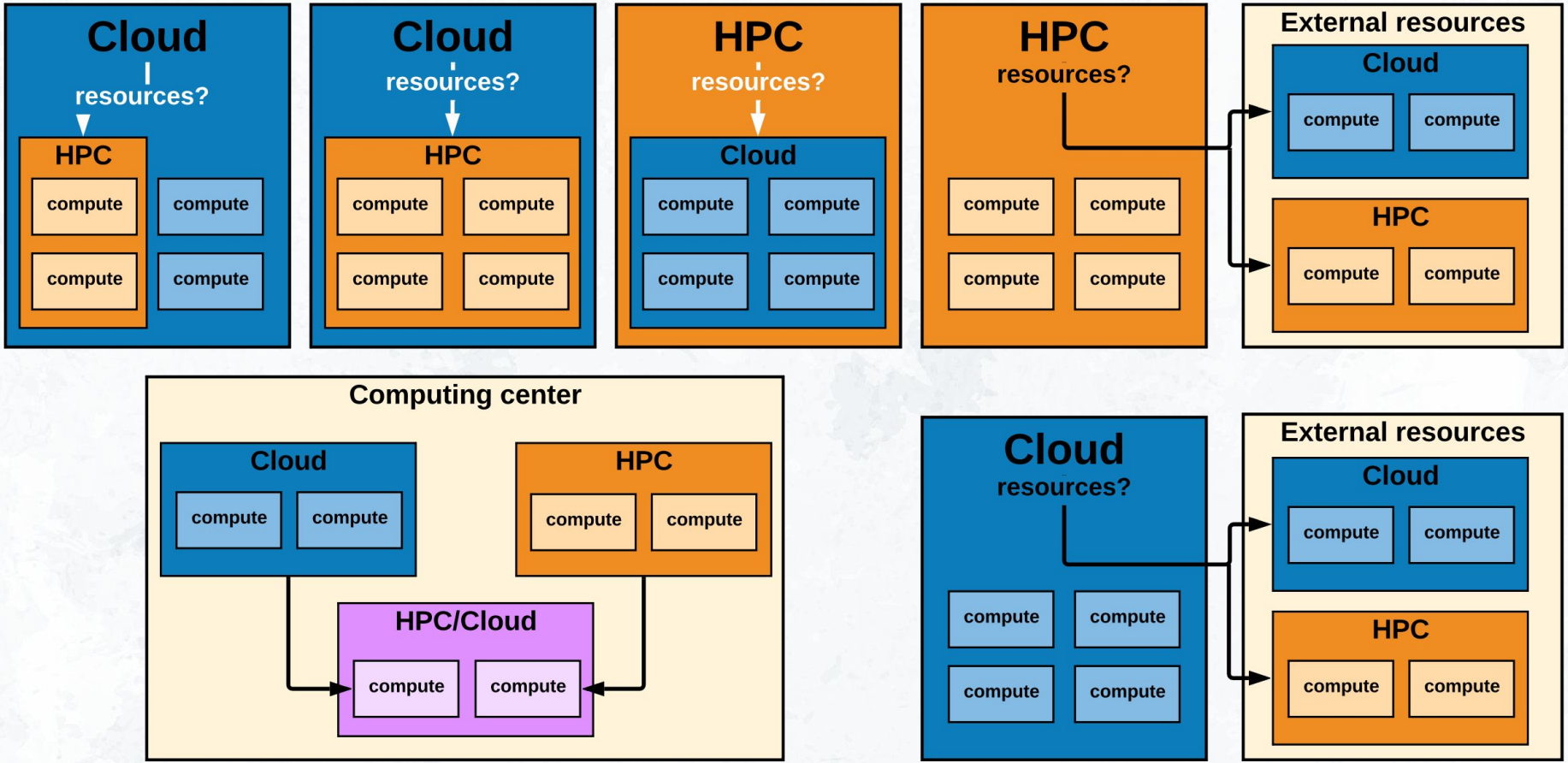
(c) Examples of Convergence

Combining models with strategies to enable converged computing.

Where do we start?

Let's talk about models of convergence.



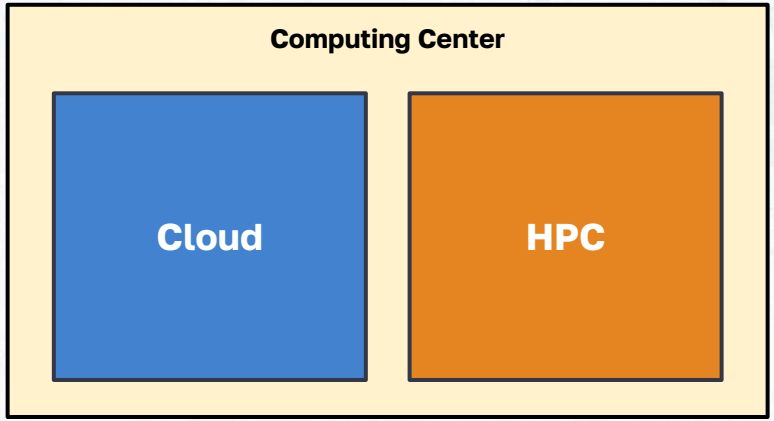


Daniel Milroy, CDO Presentation, July 2023

"I want cloud **AND** HPC"

Cloud & HPC

I am going to try and split my limited resources between two setups.

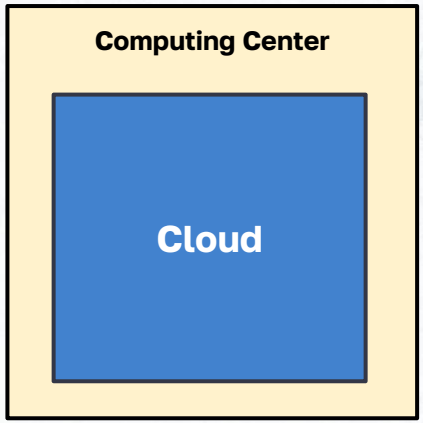


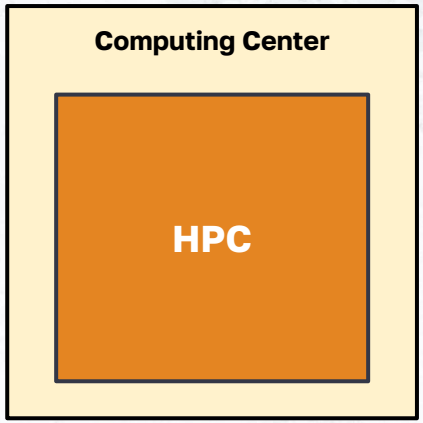

YOU CHOSE
TOO MUCH KUBERNETES

"I want cloud **XOR** HPC"

Cloud ^ HPC

I've realized I can't have my cake and eat it too, so I'm choosing just one.

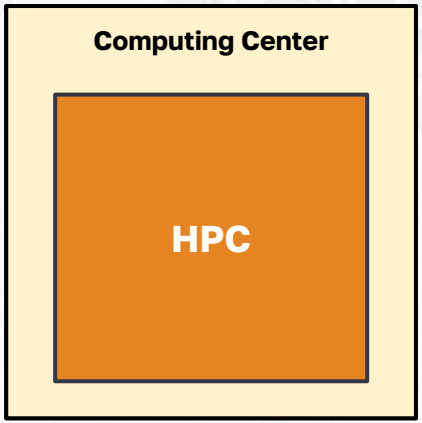




"I want cloud **XOR** HPC"

Cloud ^ HPC

I've realized I can't have my cake and eat it too, so I'm choosing just one.




YOU CHOSE
TOO LITTLE KUBERNETES

"I want to sneak it in!"

I have chosen poorly and now need a hack to add a "little more of this" to my setup.

"I want to sneak it in!"

I have chosen poorly and now need a hack to add a "little more of this" to my setup.

"Multi-cluster"

"Bursting"

"Fog Computing"

"Hybrid Cloud"

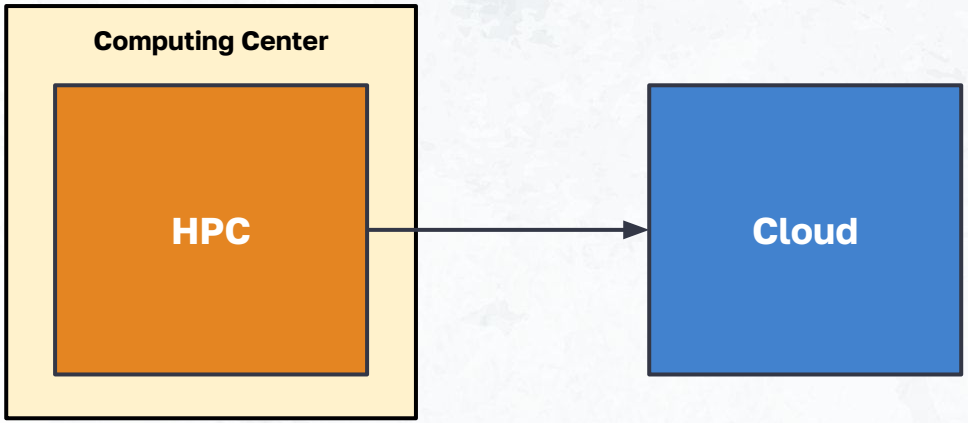
"I want to sneak it in!"

I have chosen poorly and now need a hack to add a "little more of this" to my setup.

"Multi-cluster"

"Bursting"

"Fog Computing"



"Hybrid Cloud"

"I want to sneak it in!"

I have chosen poorly and now need a hack to add a "little more of this" to my setup.

"Multi-cluster"

"Bursting"

"Fog Computing"

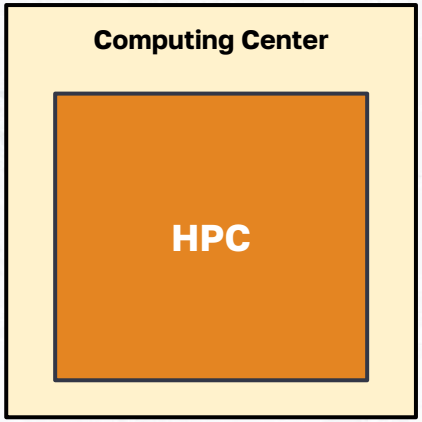
"Hybrid Cloud"

These approaches tend to be "snowflake" and complex.

"I want cloud **OR** HPC"

Cloud | HPC

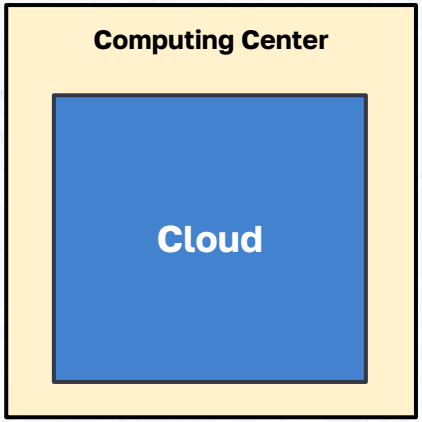
I can choose one or the other (or both) on the same resources.



"I want cloud **OR** HPC"

Cloud | HPC

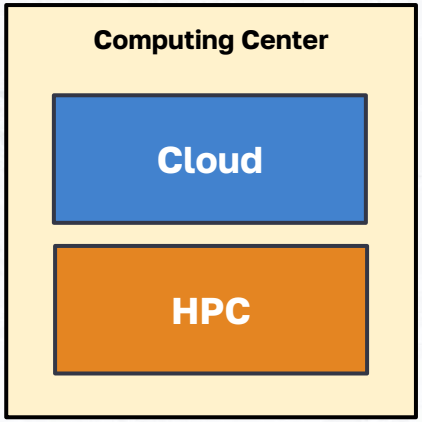
I can choose one or the other (or both) on the same resources.



"I want cloud **OR** HPC"

Cloud | HPC

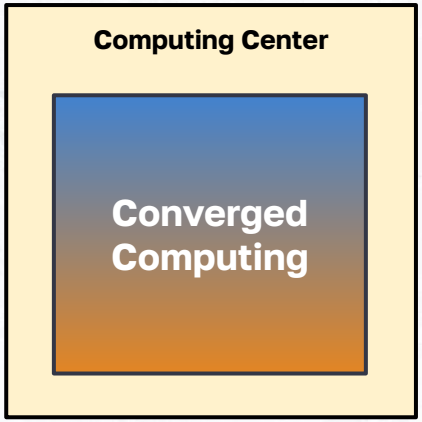
I can choose one or the other (or both) on the same resources.



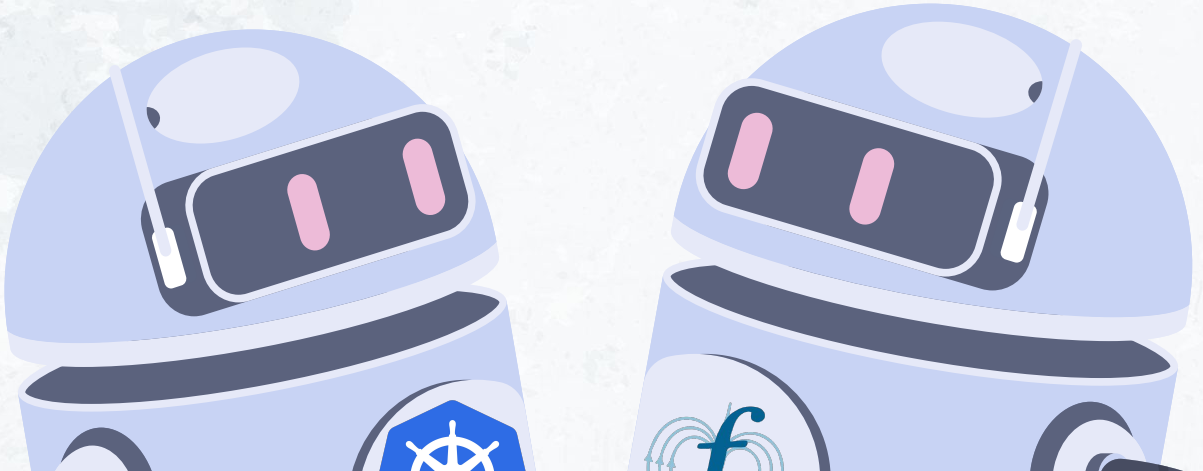
"I want cloud **OR** HPC"

Cloud | HPC

I can choose one or the other (or both) on the same resources.

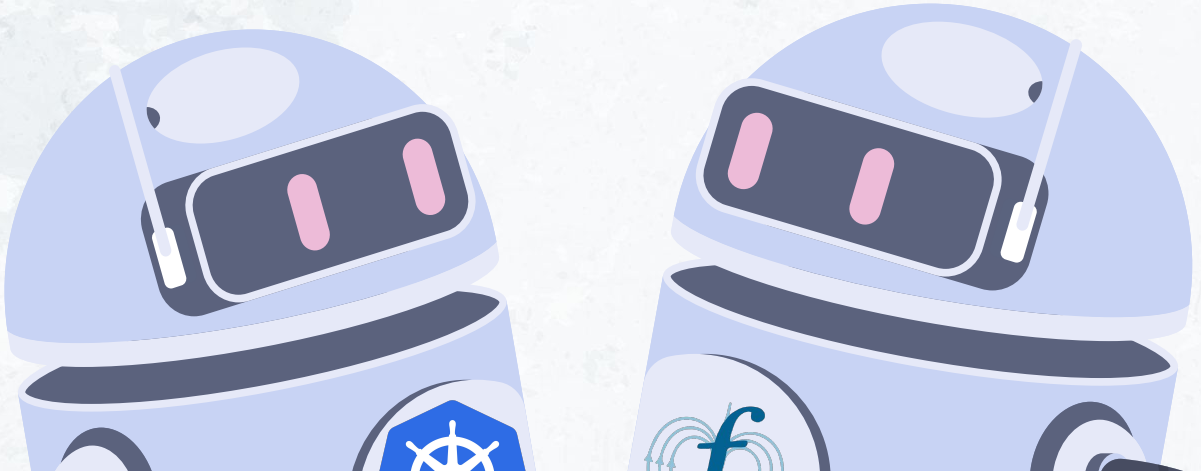


How do you do that?



How do you do that?

Now let's talk about strategies for convergence.



Converged Computing Strategies

What are strategies for convergence of technologies and people?

Converged Computing Strategies

What are strategies for convergence of technologies and people?

(a) Goals

Shared goals align incentives correctly for collaboration toward shared needs, and potentially mutual vision.

Converged Computing Strategies

What are strategies for convergence of technologies and people?

(a) Goals

Shared goals align incentives correctly for collaboration toward shared needs, and potentially mutual vision.

(b) Modularity

An application or infrastructure that is modular can have components used interchangeably.

Converged Computing Strategies

What are strategies for convergence of technologies and people?

(a) Goals

Shared goals align incentives correctly for collaboration toward shared needs, and potentially mutual vision.

(b) Modularity

An application or infrastructure that is modular can have components used interchangeably.

(c) Integration

Consumption or deploying components or entirety of one inside of another.

Converged Computing Strategies

What are strategies for convergence of technologies and people?

(a) Goals → batch workloads

Shared goals align incentives correctly for collaboration toward shared needs, and potentially mutual vision.

(b) Modularity

An application or infrastructure that is modular can have components used interchangeably.

(c) Integration

Consumption or deploying components or entirety of one inside of another.

Converged Computing Strategies

What are strategies for convergence of technologies and people?

(a) Goals → batch workloads

Shared goals align incentives correctly for collaboration toward shared needs, and potentially mutual vision.

(b) Modularity →

An application or infrastructure that is modular can have components used interchangeably.

(c) Integration

Consumption or deploying components or entirety of one inside of another.

Flux Framework

A workload manager that combines hierarchical management with graph based scheduling.

flux-core | flux-sched | flux-security | flux-accounting | flux-pmix

Kubernetes

The de-facto container orchestration system for automated deployment, scaling, and management of software.

kube-apiserver | kube-scheduler | kube-controller manager | kube-proxy | kubelet | container runtime

Converged Computing Strategies

What are strategies for convergence of technologies and people?

(a) Goals → batch workloads

Shared goals align incentives correctly for collaboration toward shared needs, and potentially mutual vision.

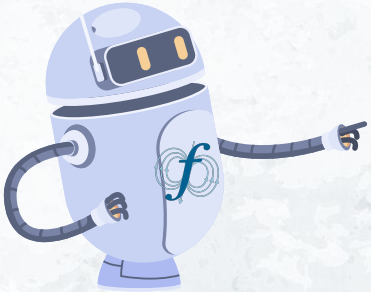
(b) Modularity → 

An application or infrastructure that is modular can have components used interchangeably.

(c) Integration → containers / language bindings

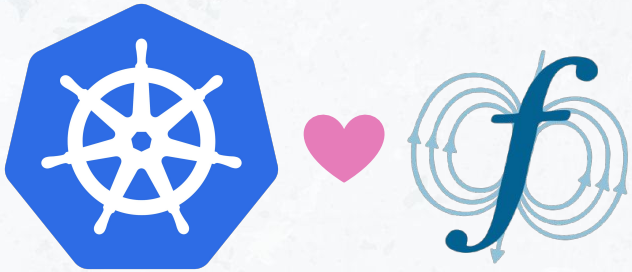
Consumption or deploying components or entirety of one inside of another.

What are some examples?



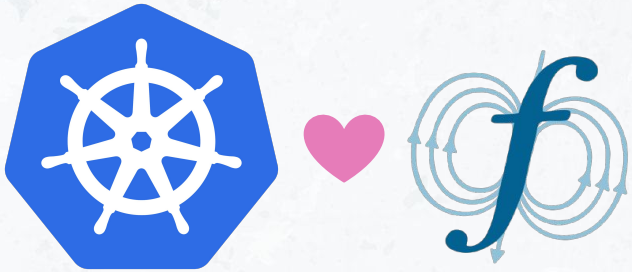
Converged Computing Projects

- **Fluence:** the Flux scheduler swapped with kube-scheduler



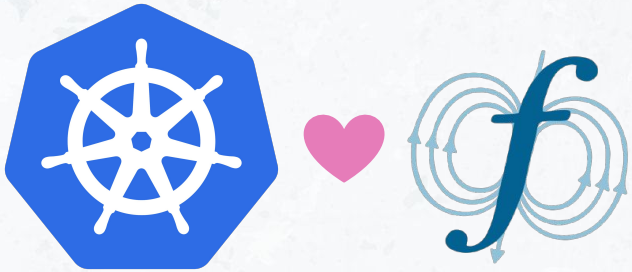
Converged Computing Projects

- **Fluence:** the Flux scheduler swapped with kube-scheduler
- **The Flux Operator:** Flux implemented inside of Kubernetes



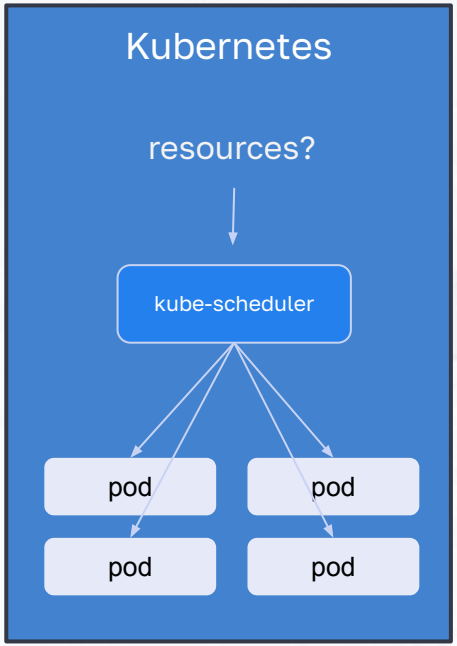
Converged Computing Projects

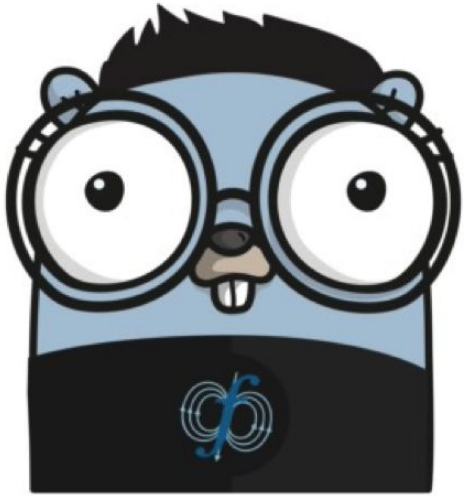
- **Fluence:** the Flux scheduler swapped with kube-scheduler
- **The Flux Operator:** Flux implemented inside of Kubernetes
- **Flux and Kubernetes:** "Bare Metal Bros" working side by side



1: The Flux Scheduler within Kubernetes

create job



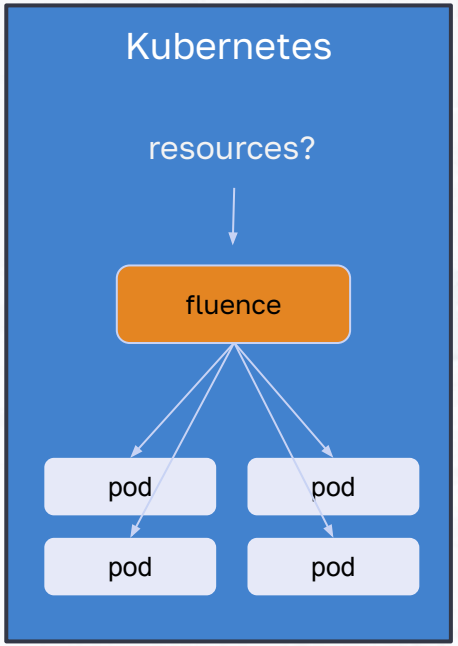


F

fluence

1: **Fluence**: The Flux Scheduler within Kubernetes

create job



“I’m using Kubernetes as I would typically, but the pod scheduling is being done by Flux”

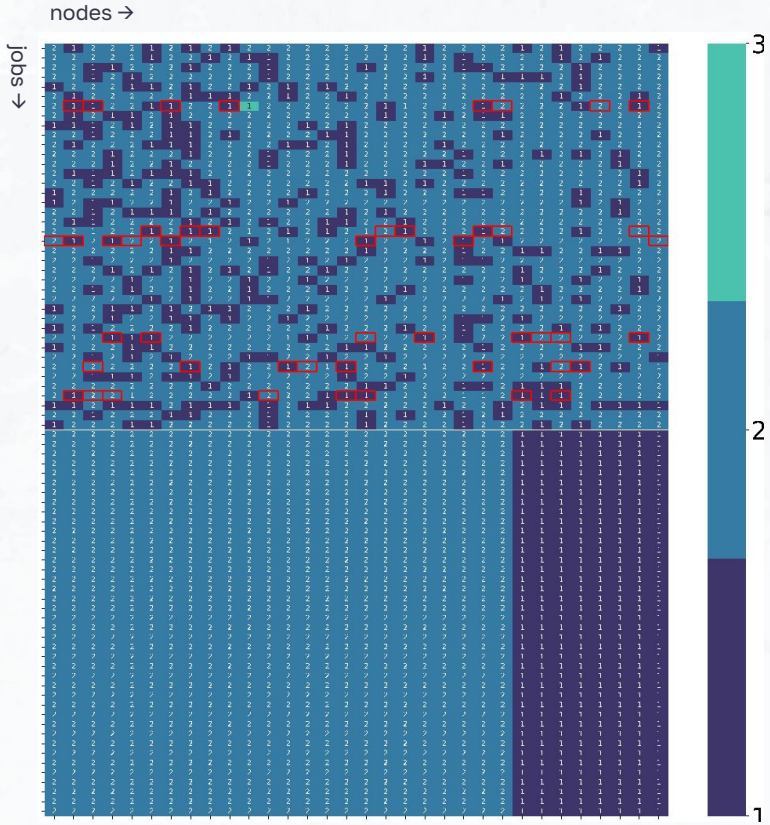
flux-framework/flux-k8s

Fluence scheduled workflows run 3x faster

with low variability and deterministic placement, as compared to kube-scheduler

kube-scheduler

Fluence



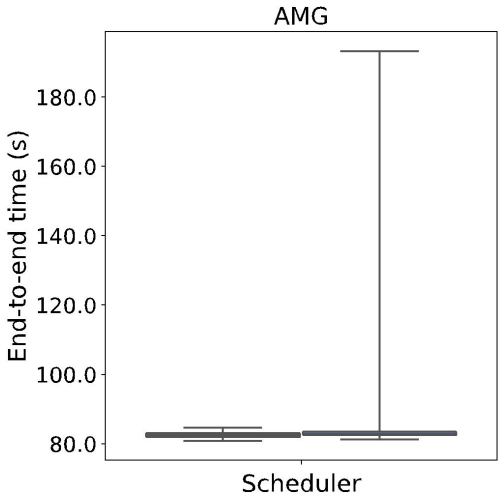
Fluence scheduled pods to nodes to avoid pathological scheduling that led to startup delay

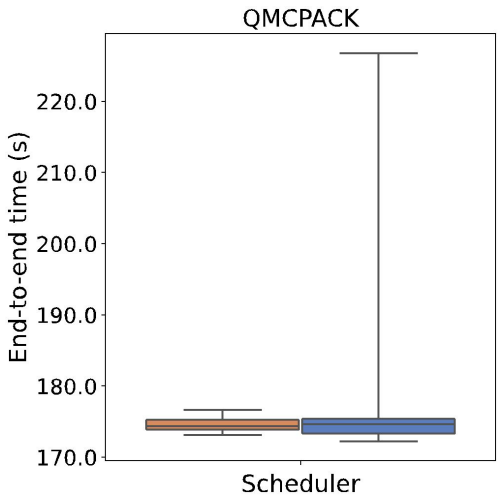
D. J. Milroy *et al.*, "One Step Closer to Converged Computing: Achieving Scalability with Cloud-Native HPC," *2022 CANOPIE-HPC*

https://github.com/flux-framework/flux-k8s

Fluence scheduled workflows run 3x faster

with low variability and deterministic placement, as compared to kube-scheduler





kube-scheduler

Fluence

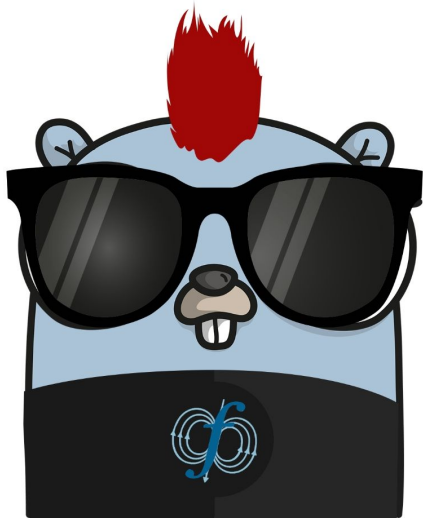
D. J. Milroy *et al.*, "One Step Closer to Converged Computing: Achieving Scalability with Cloud-Native HPC," *2022 CANOPIE-HPC*

(b) Modularity

Fluence exemplifies taking an HPC-oriented technology (the scheduler for Flux Framework) and swapping it into Kubernetes to improve upon an analogous component in a the cloud-native orchestrator, Kubernetes.

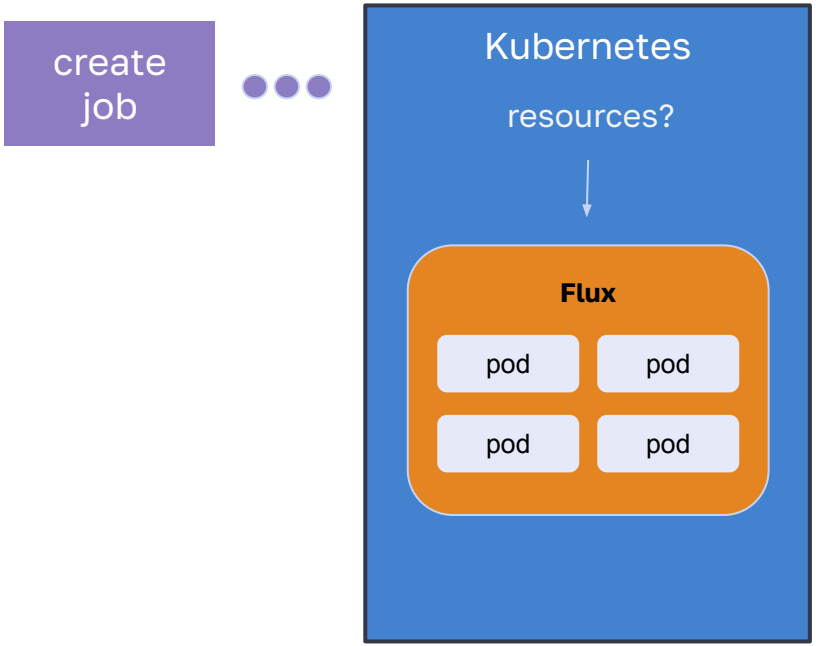
(c) Integration

Fluence exemplifies the importance of a feature like language bindings (in Go) to map a different language (C++) into a new space (Kubernetes and cloud-native projects are primarily in Go).



THE OPERATOR

2: The Flux Operator: HPC Workload Manager inside K8s



I want my own HPC cluster inside of Kubernetes!

Sochat *et al.*, "The Flux Operator," *manuscript submitted*
https://github.com/flux-framework/flux-operator

2: The Flux Operator: HPC Workload Manager inside K8s

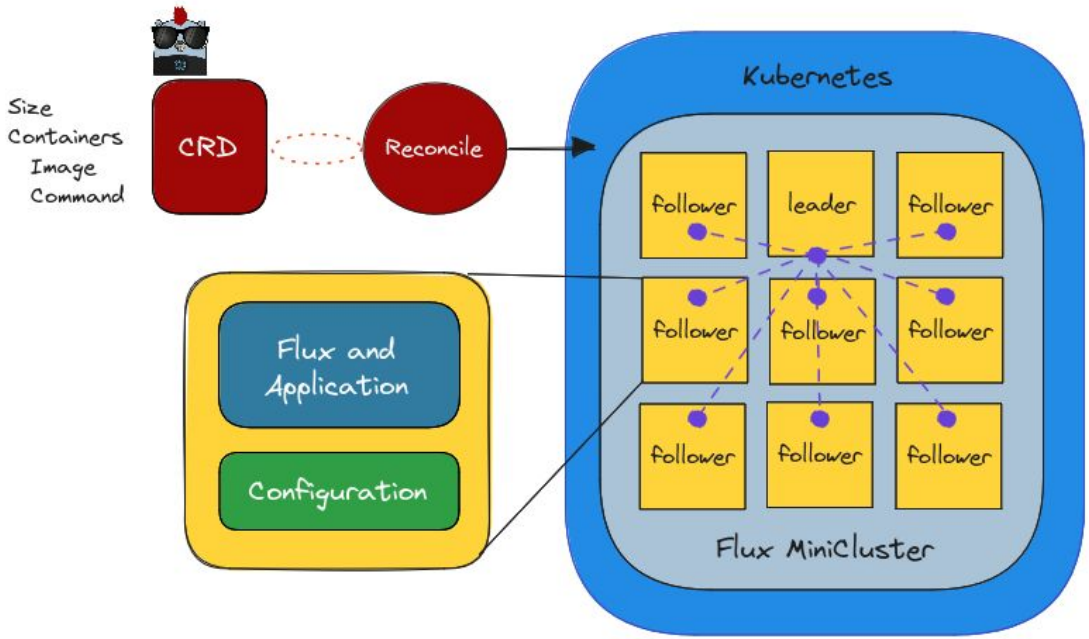
Custom Resource Definition "CRD"

```yaml
apiVersion: flux-framework.org/v1alpha2
kind: MiniCluster
metadata:
  name: flux-sample
spec:
  size: 4
  containers:
    - image: ghcr.io/converged-computing/metric-lammps:latest
      workingDir: /opt/lammps/examples/reaxff/HNS
      command: lmp -v x 2 -v y 2 -v z 2 -in in.reaxc.hns -nocite
```

Create a "MiniCluster" inside of Kubernetes with hierarchical, graph-based scheduling and fine-granted resource mapping by Flux.

Sochat *et al.*, "The Flux Operator," *manuscript submitted*
https://github.com/flux-framework/flux-operator

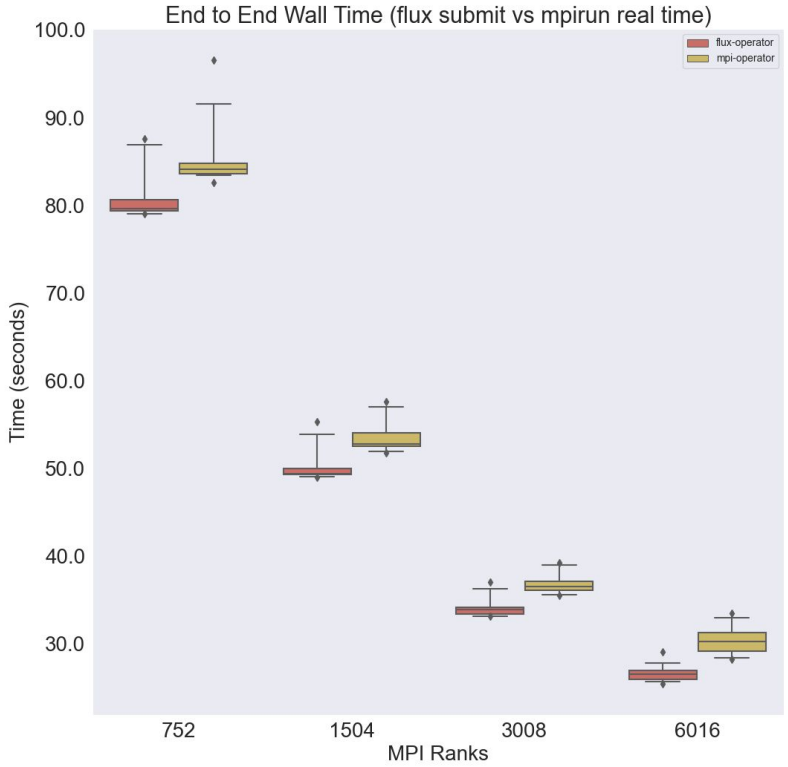
2: The Flux Operator: HPC Workload Manager inside K8s



Sochat *et al.*, "The Flux Operator," *manuscript submitted*
https://github.com/flux-framework/flux-operator

2: The Flux Operator: HPC Workload Manager inside K8s

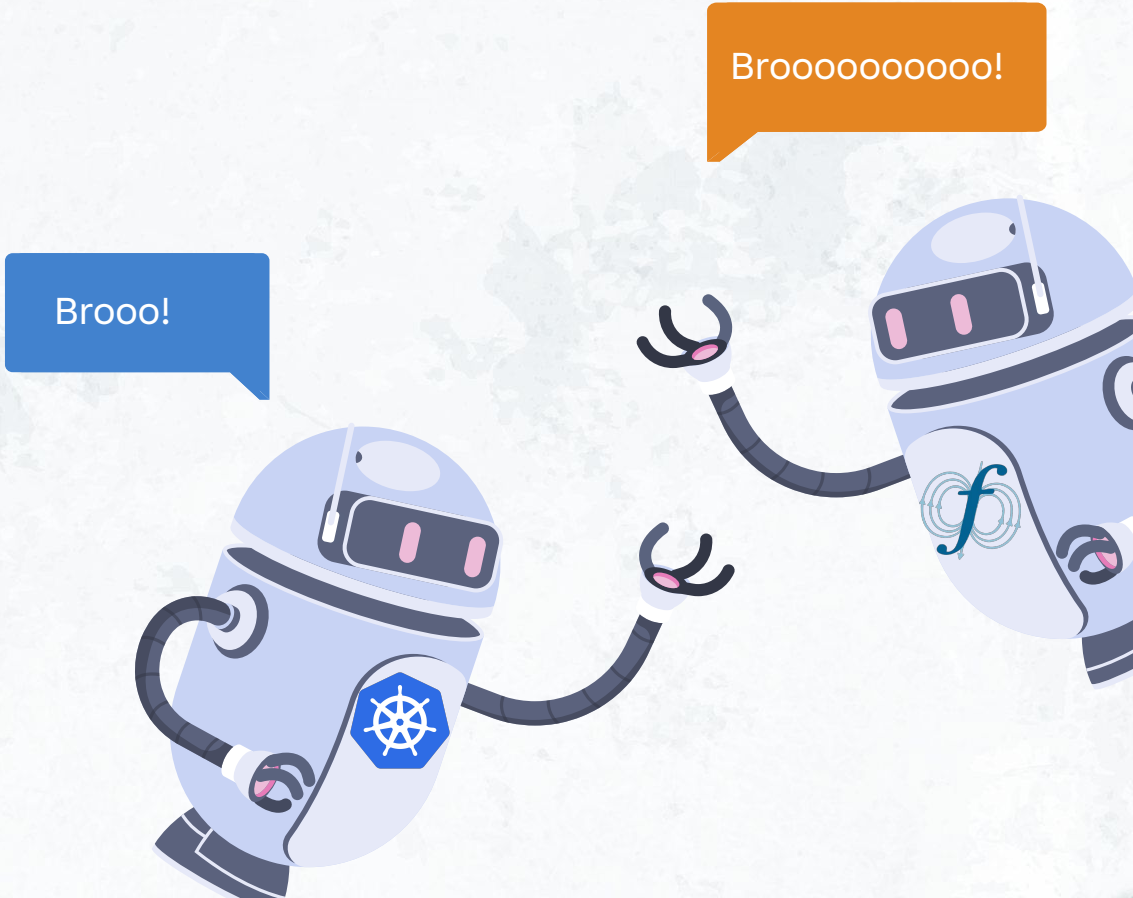
Outperformed the best in the space, the MPI Operator



Sochat *et al.*, "The Flux Operator," *Kubecon EU, 2023*
https://github.com/flux-framework/flux-operator

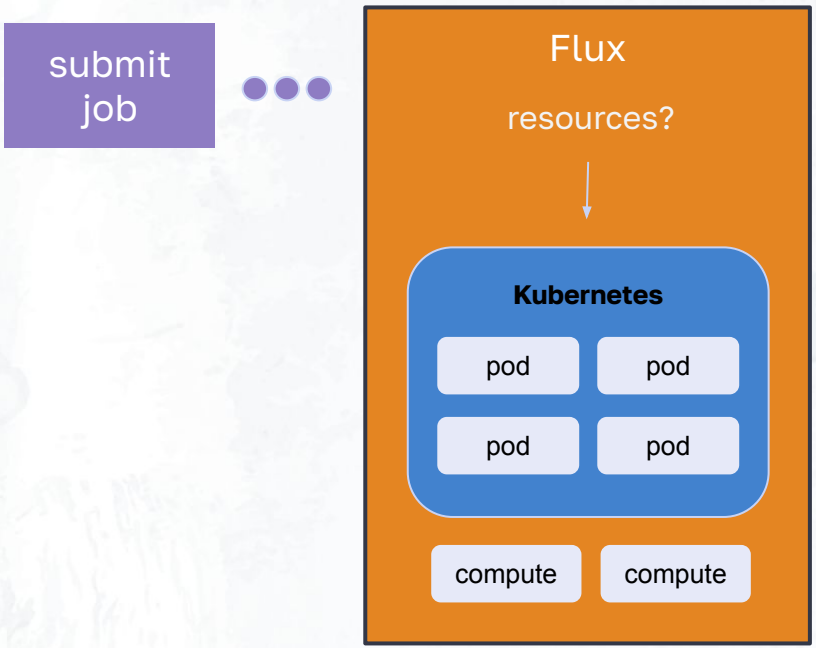
(c) Integration

The Flux Operator exemplifies taking the entirety of Flux Framework and implementing it inside of Kubernetes, made possible by the operator framework, containers, and design.


Brooooooooooo!
Brooo!

WARNING: virtual machines are used as a prototype for bare metal, proceed at own risk!

3: **Bare Metal Bros:** Flux as an external orchestrator



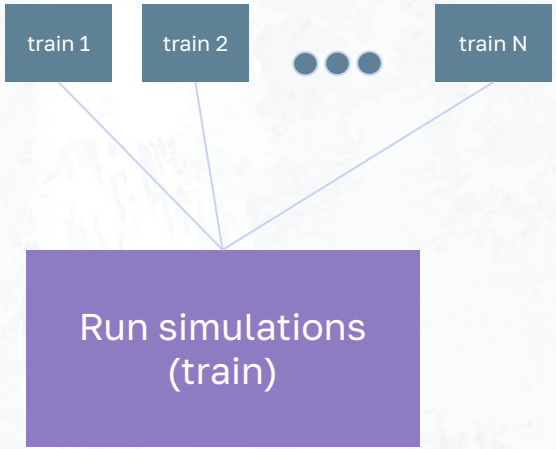
“I want the best of both worlds, Kubernetes running on the same resources as my workload manager."

Complex workflows require HPC and services

WARNING: Vanessa is not a scientist and is terrible at science.

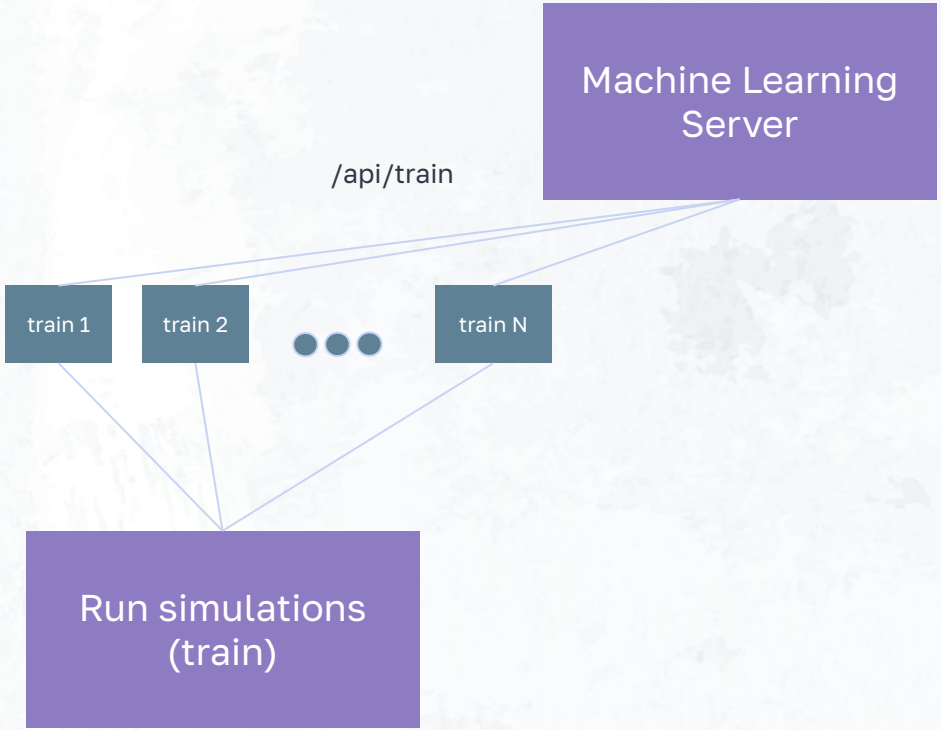
Complex workflows require HPC and services

1. Running a simulation on bare metal HPC alongside a service



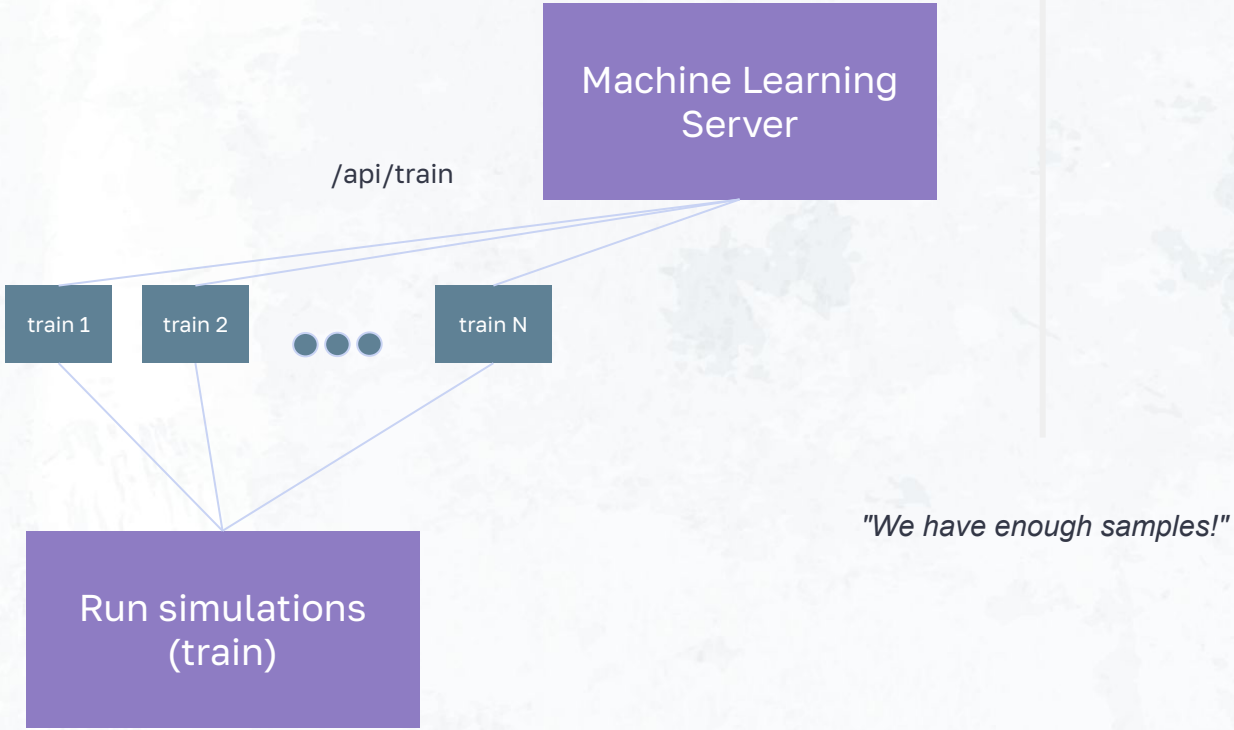
Complex workflows require HPC and services

2. Sending results to the service as you go (in this case, ML training points)



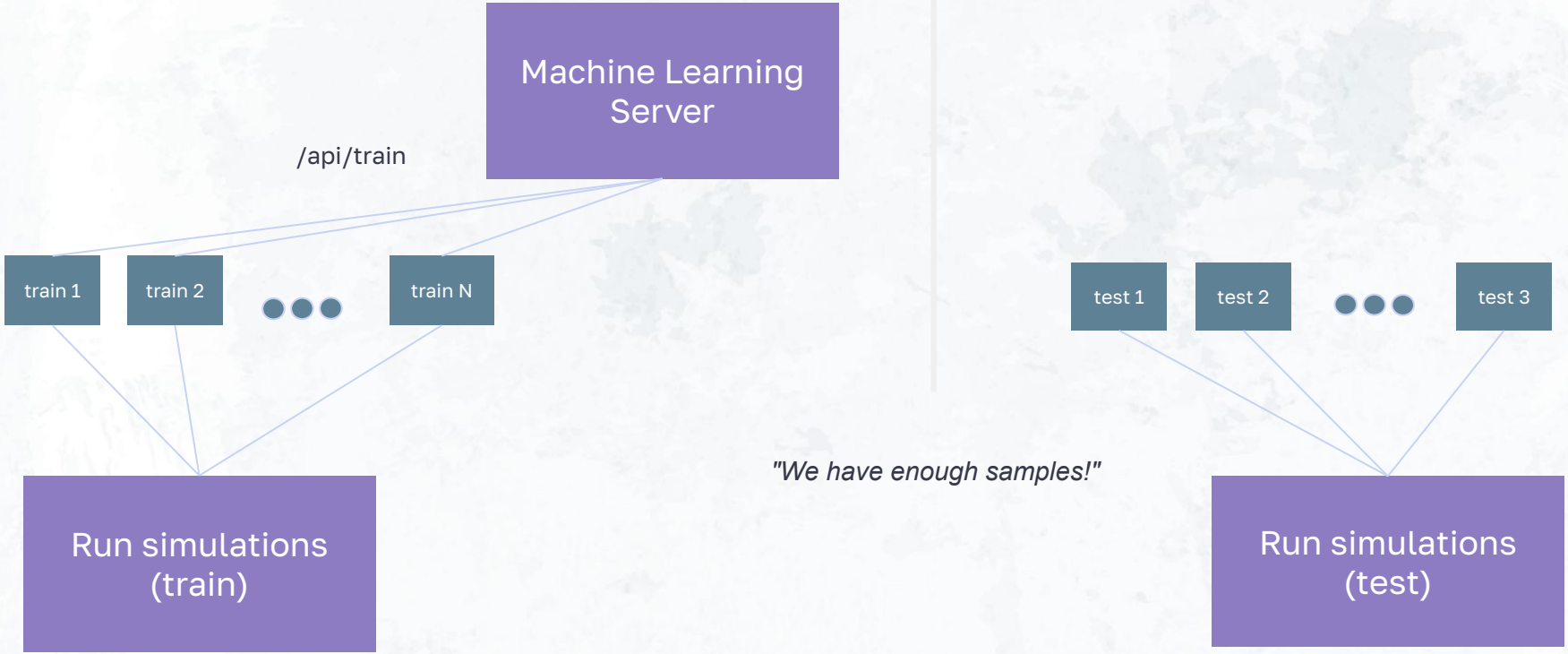
Complex workflows require HPC and services

3. Using the service to get updated info about the model on demand



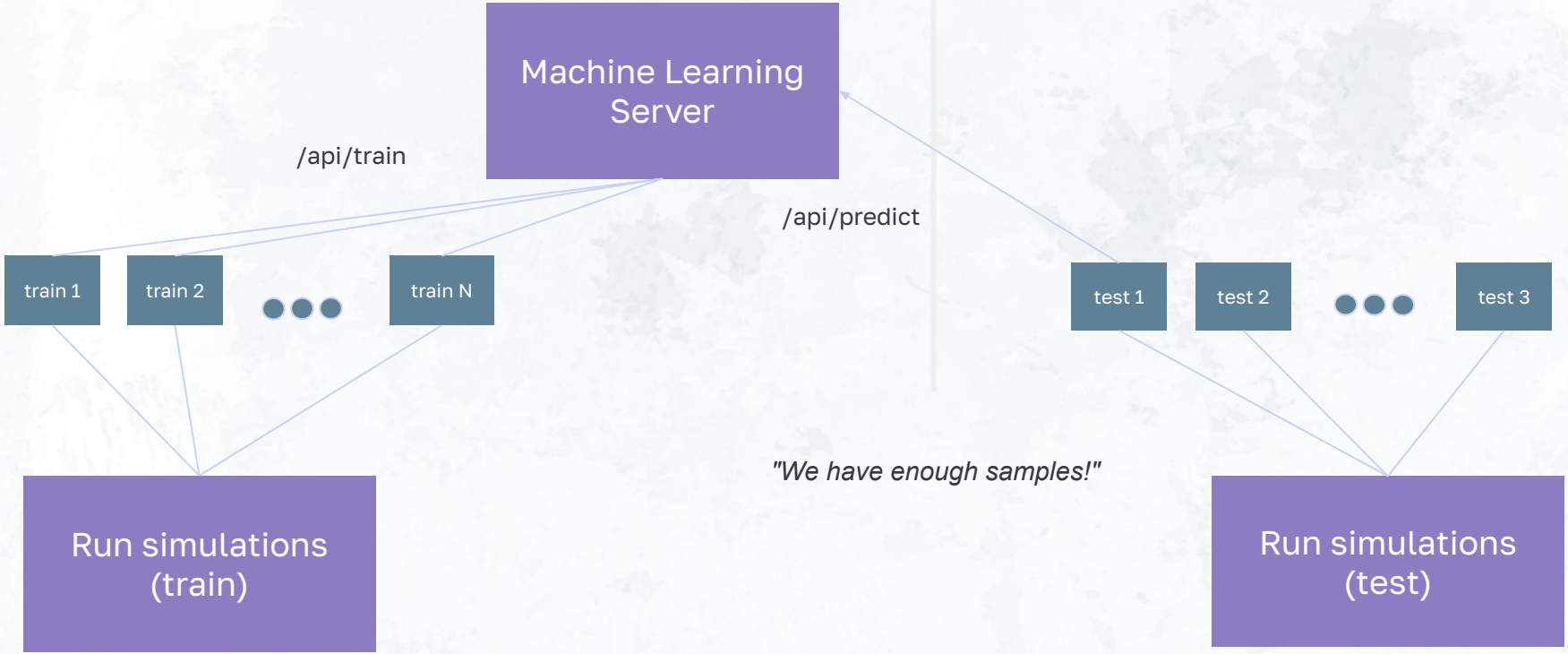
Complex workflows require HPC and services

4. Doing a second phase (e.g., hold out testing) with your trained model.



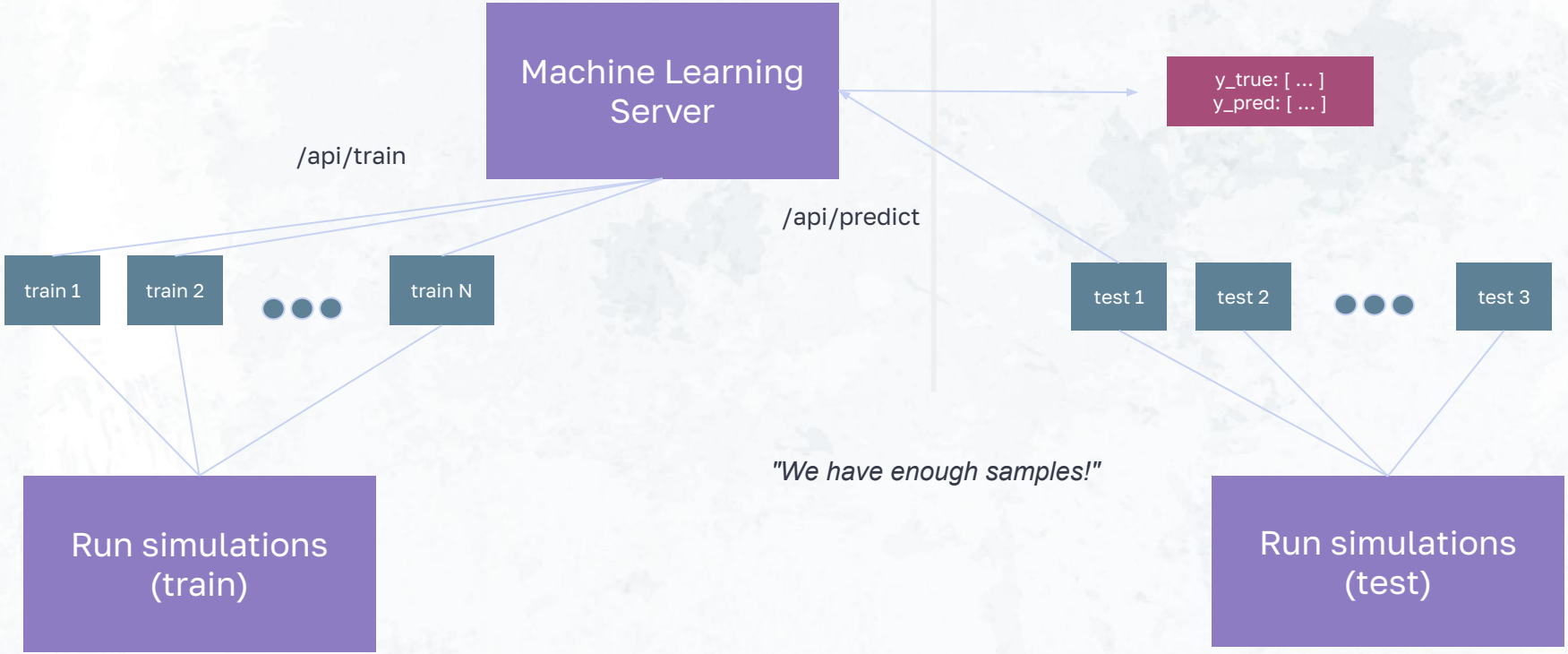
Complex workflows require HPC and services

4. Doing a second phase (e.g., hold out testing) with your trained model.

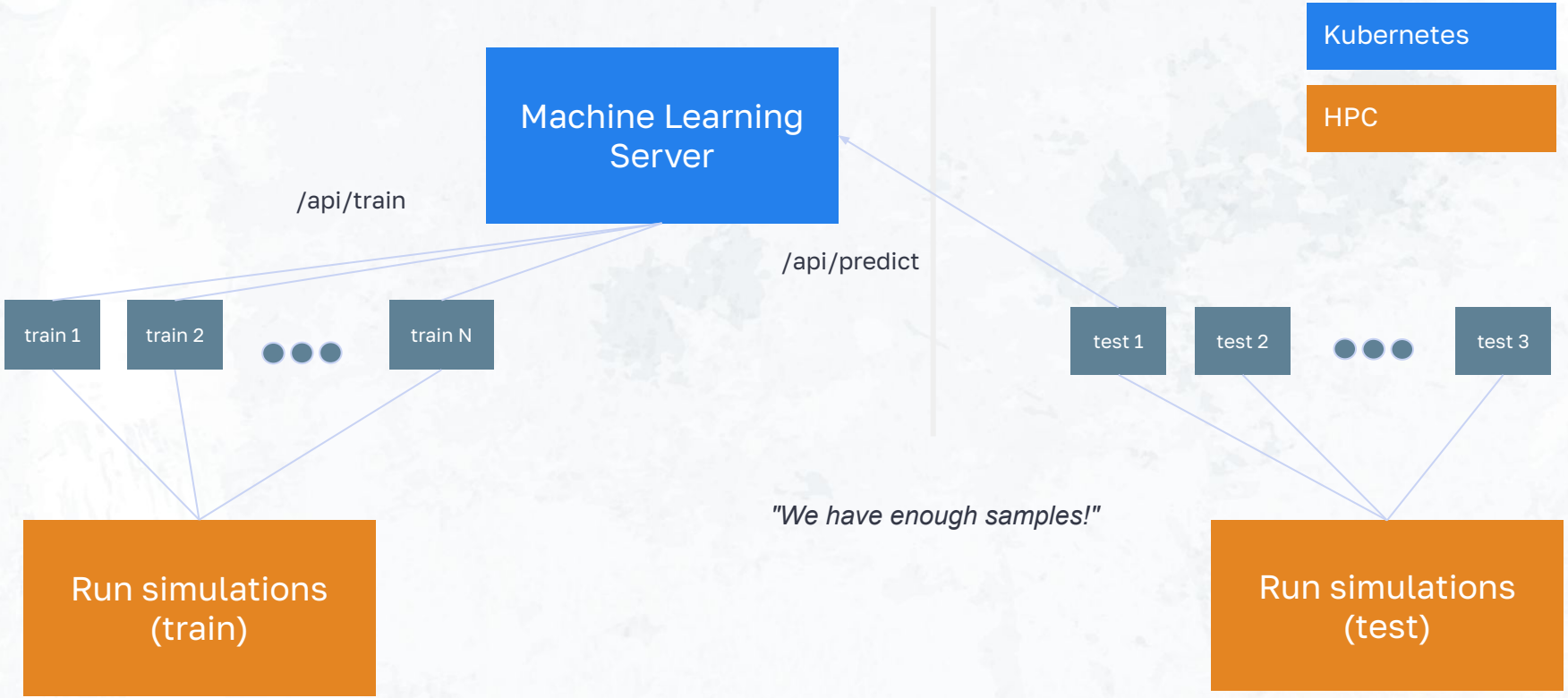


Complex workflows require HPC and services

4. Doing a second phase (e.g., hold out testing) with your trained model.



Complex workflows require HPC and services



Kubelets in user namespace "Usernetes"

- Kubernetes Enhancement Proposal (KEP) proposed in mid 2022, Akihiro Suda

Kubelets in user namespace "Usernetes"

- Kubernetes Enhancement Proposal (KEP) proposed in mid 2022, Akihiro Suda

Kubelets in user namespace "Usernetes"

- Kubernetes Enhancement Proposal (KEP) proposed in mid 2022, Akihiro Suda

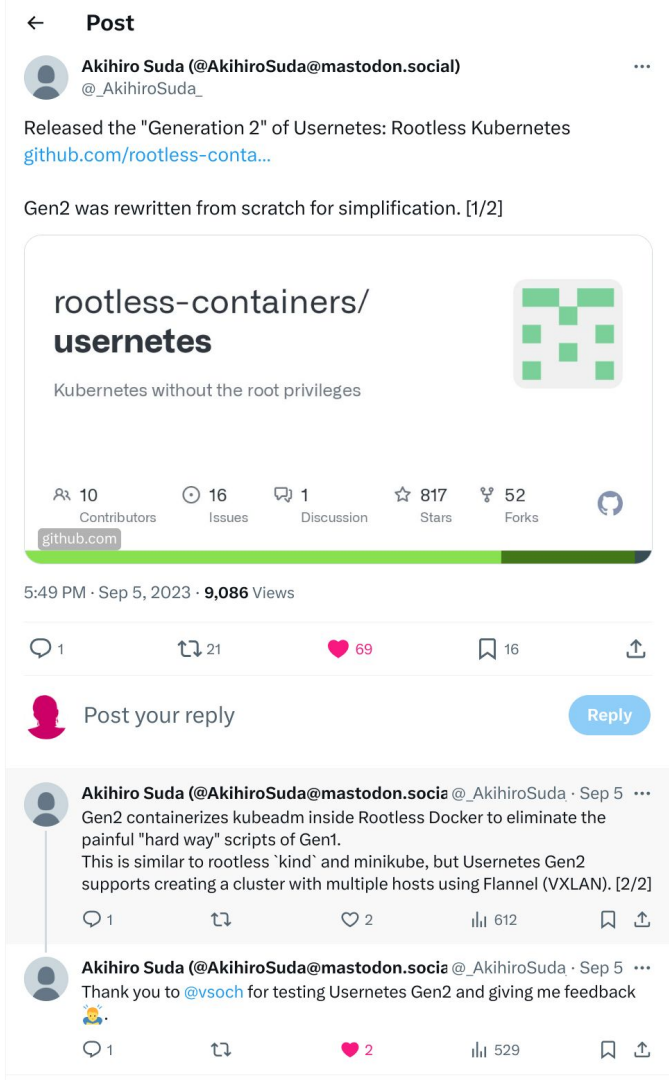
Usernetes
containerd
kernel
runc
buildkit
rootless
Lima (VMs)
lazy pulling
more!

Usernetes

- KEP proposed in mid 2022
  - Gen-2 of Usernetes - using containers!

Usernetes

- KEP proposed in mid 2022
  - Gen-2 of Usernetes - using containers!
  - Components:
    - Cluster configuration: kubeadm
    - CRI: containerd
    - OCI: runc
    - CNI: Flannel



Usernetes: in practice

1. Building VMs with cgroups v2 (I recommend Lima, Linux Virtual Machines)

Usernetes: in practice

1. Building VMs with cgroups v2 (I recommend Lima, Linux Virtual Machines)
2. Enable kernel modules
    a. br_netfilter
    b. vxlan

Usernetes: in practice

1. Building VMs with cgroups v2 (I recommend Lima, Linux Virtual Machines)
2. Enable kernel modules
    a. br_netfilter: allows you to apply iptables rules to bridged traffic
    b. vxlan

Usernetes: in practice

1. Building VMs with cgroups v2 (I recommend Lima, Linux Virtual Machines)
2. Enable kernel modules
    a. br_netfilter: allows you to apply iptables rules to bridged traffic
    b. vxlan: connect vxlan devices on different hosts to standalone bridge

Usernetes: in practice

1. Building VMs with cgroups v2 (I recommend Lima, Linux Virtual Machines)
2. Enable kernel modules
   a. br_netfilter: allows you to apply iptables rules to bridged traffic
   b. vxlan: connect vxlan devices on different hosts to standalone bridge
3. Rootless docker

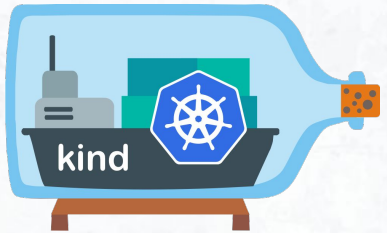
Usernetes: in practice

1. Building VMs with cgroups v2 (I recommend Lima, Linux Virtual Machines)
2. Enable kernel modules
    a. br_netfilter: allows you to apply iptables rules to bridged traffic
    b. vxlan: connect vxlan devices on different hosts to standalone bridge
3. Rootless docker
4. docker compose "make up" run via a Makefile for two contexts:
    a. Both: Build/start base image (same as kind) with CNI plugins
    b. Control plane: install flannel, run "kubeadm init" create the join command
    c. Worker: join cluster

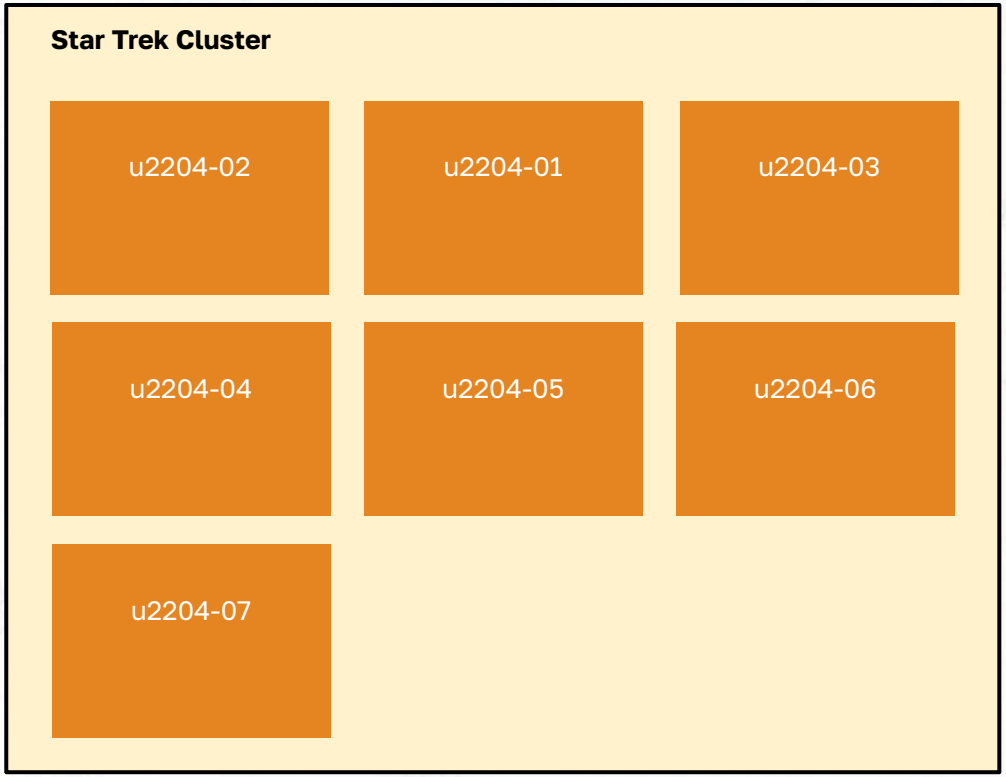
Usernetes: in practice

1. Building VMs with cgroups v2 (I recommend Lima, Linux Virtual Machines)
2. Enable kernel modules
   a. br_netfilter: allows you to apply iptables rules to bridged traffic
   b. vxlan: connect vxlan devices on different hosts to standalone bridge
3. Rootless docker
4. docker compose "make up" run via a Makefile for two contexts:
   a. Both: Build/start base image (same as kind) with CNI plugins
   b. Control plane: install flannel, run "kubeadm init" create the join command
   c. Worker: join cluster

Kubernetes **IN** **D**ocker

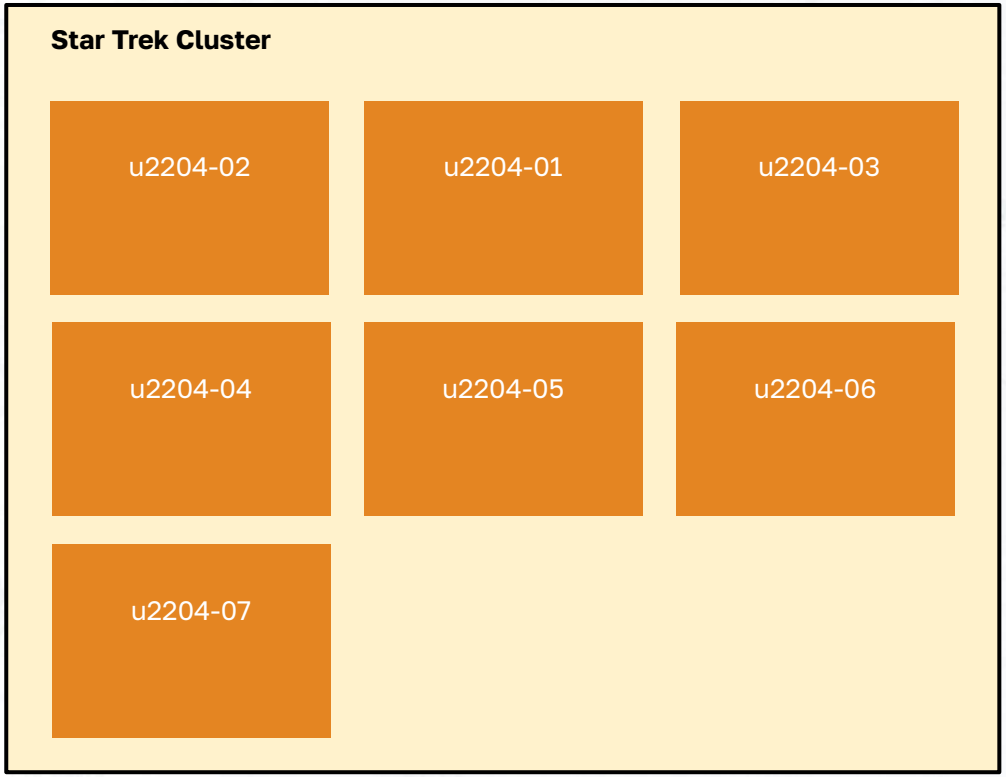


We created a VM cluster to test Usernetes | Flux



1. Star Trek Cluster:
   a. oVirt and Ansible
   b. 7 x ubuntu 22.04
   c. 8 cores / each
   d. 32 MB RAM
   e. Ethernet (10GB)

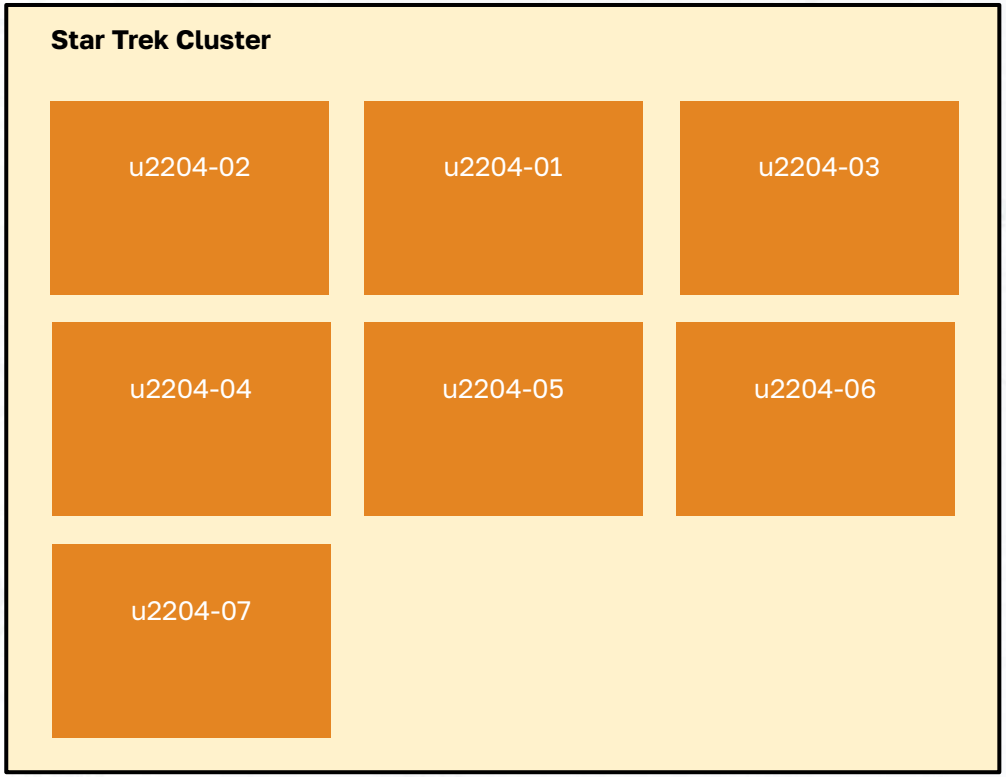
We created a VM cluster to test Usernetes | Flux



1. Star Trek Cluster:
   a. oVirt and Ansible
   b. 7 x ubuntu 22.04
   c. 8 cores / each
   d. 32 MB RAM
   e. Ethernet (10GB)

WARNING: virtual machines are used as a prototype for bare metal, proceed at own risk!

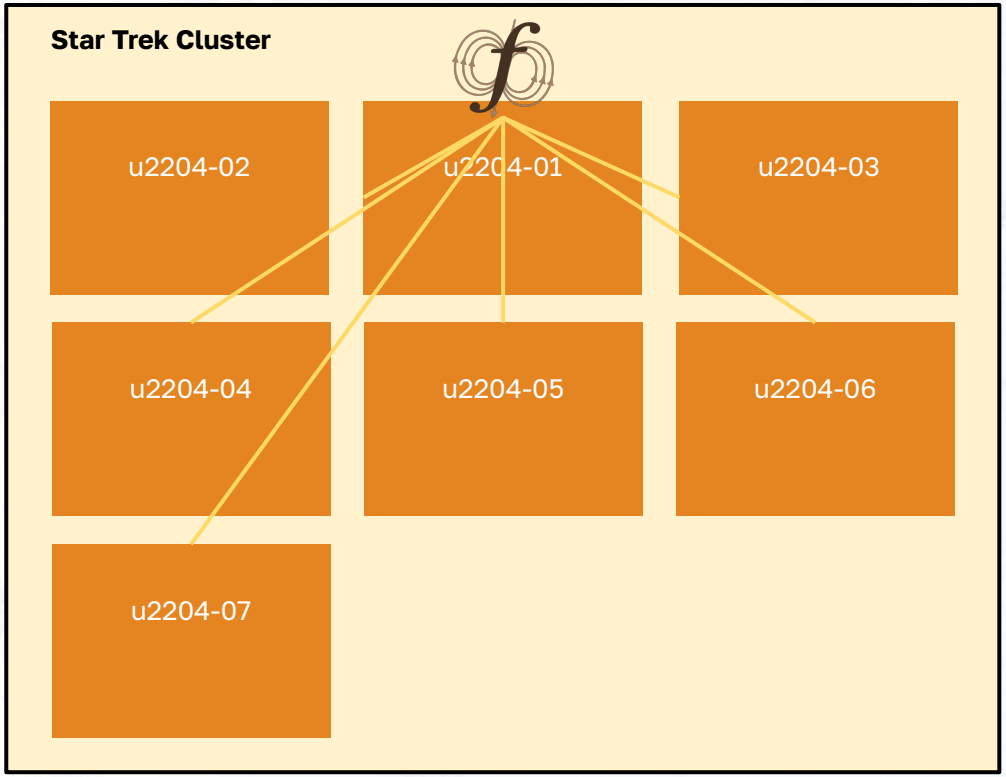
We created a VM cluster to test Usernetes | Flux



1. Star Trek Cluster:
   a. oVirt and Ansible
   b. 7 x ubuntu 22.04
   c. 8 cores / each
   d. 32 MB RAM
   e. Ethernet (10GB)

2. Setup includes:
   a. System install of Flux
   b. Singularity
   c. LAMMPS
   d. Usernetes

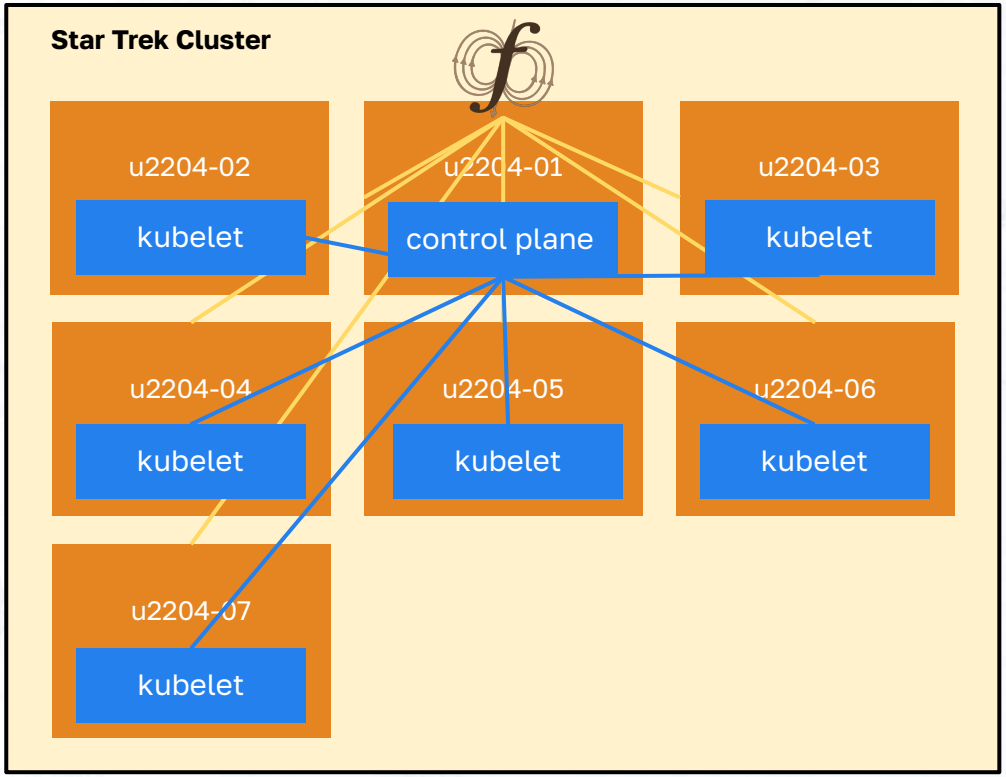
We created a VM cluster to test Usernetes | Flux



$ flux resource list

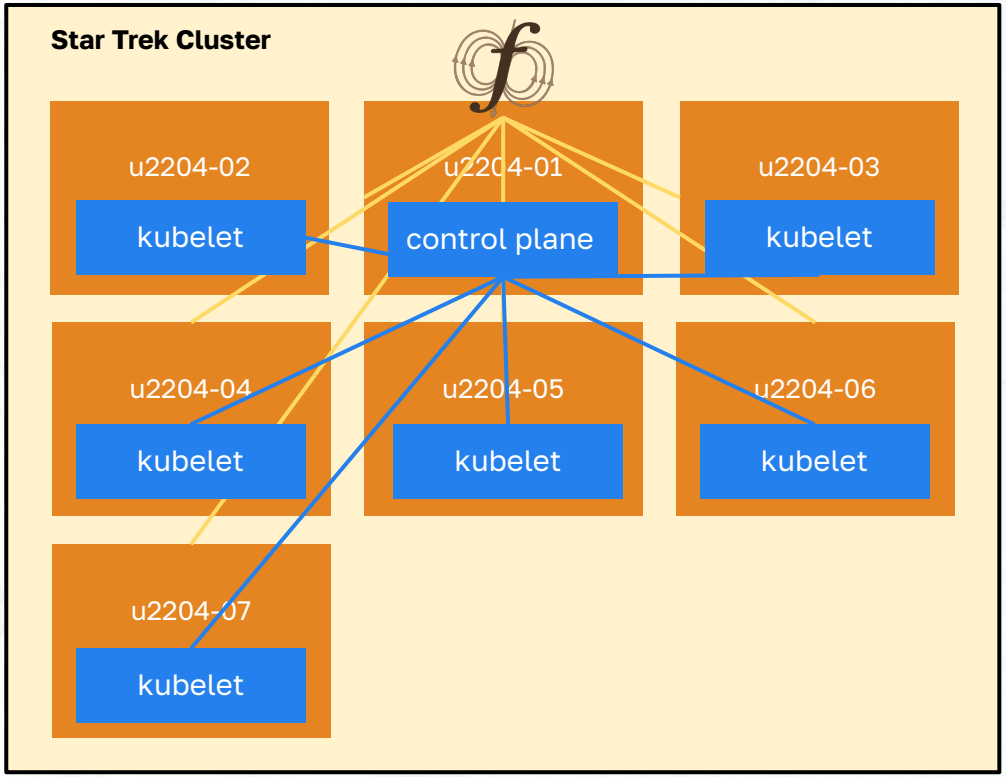
STATE	NNODES	NCORES	NGPUS	NODELIST
free	6	48	0	u2204-[02-07]
allocated	0	0	0	
down	0	0	0	

We created a VM cluster to test Usernetes | Flux



Usernetes

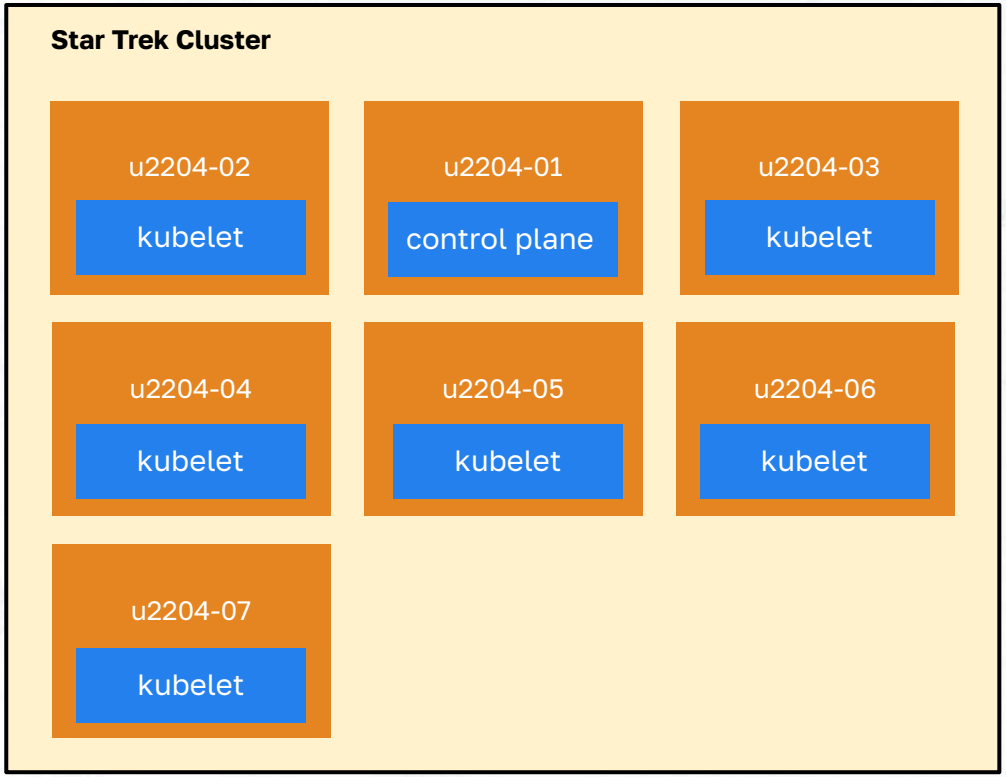
We created a VM cluster to test Usernetes | Flux



Usernetes

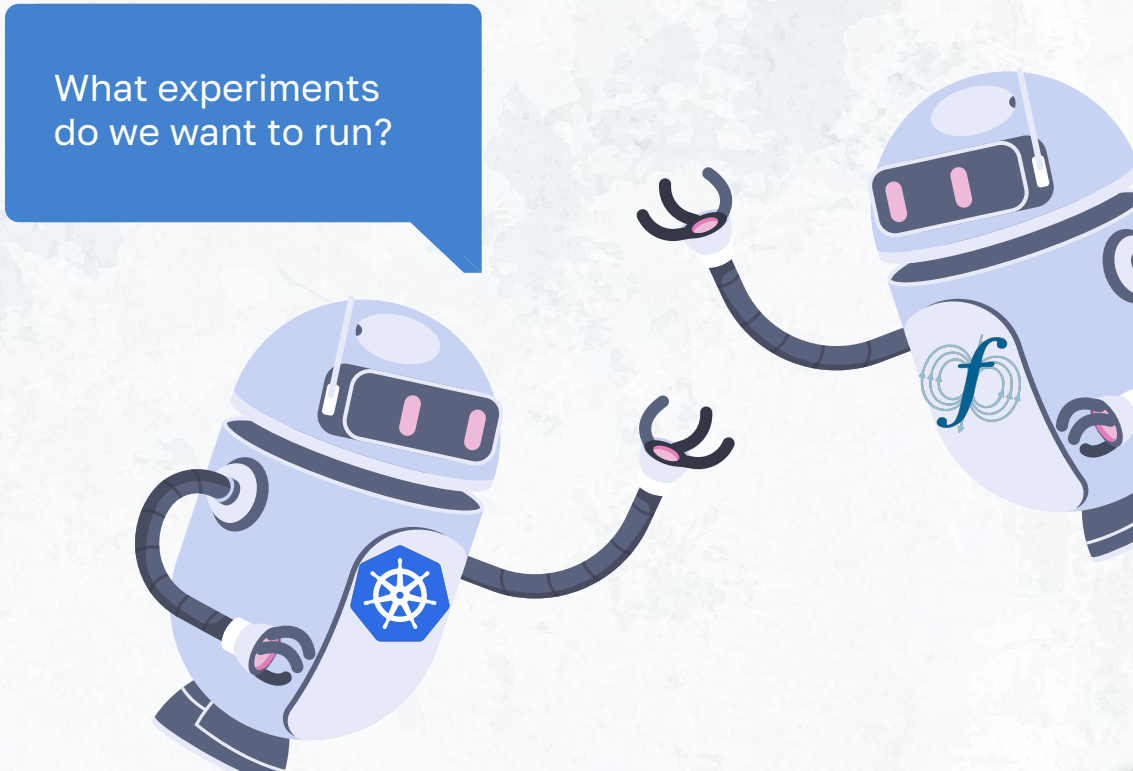
NAME	STATUS	ROLES	AGE	VERSION
u7s-u2204-01	Ready	control-plane	8m44s	v1.29.0
u7s-u2204-02	Ready	<none>	6m58s	v1.29.0
u7s-u2204-03	Ready	<none>	2m3s	v1.29.0
u7s-u2204-04	Ready	<none>	100s	v1.29.0
u7s-u2204-05	Ready	<none>	70s	v1.29.0
u7s-u2204-06	Ready	<none>	52s	v1.29.0
u7s-u2204-07	Ready	<none>	14s	v1.29.0

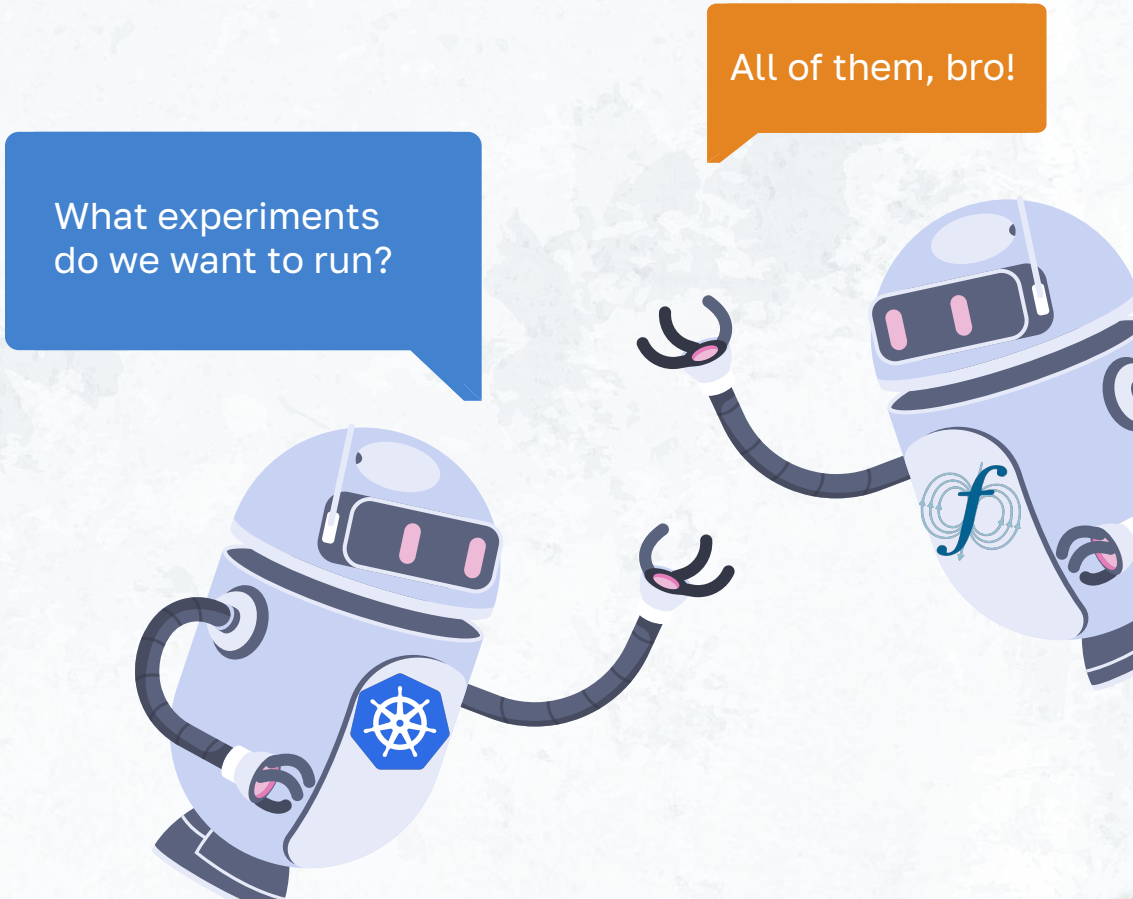
We created a VM cluster to test Usernetes | Flux



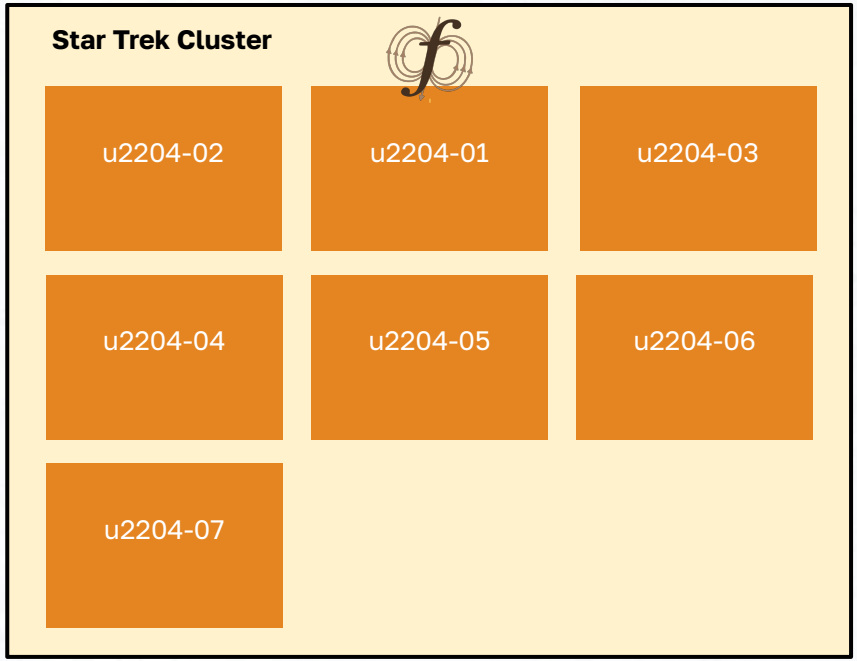
Usernetes

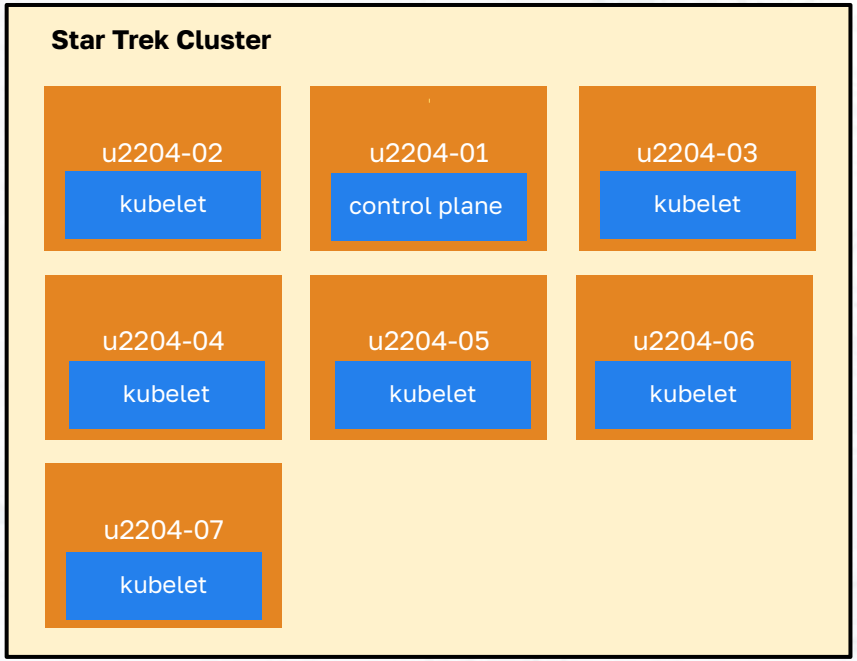
Flux


What experiments
do we want to run?


What experiments do we want to run?
All of them, bro!

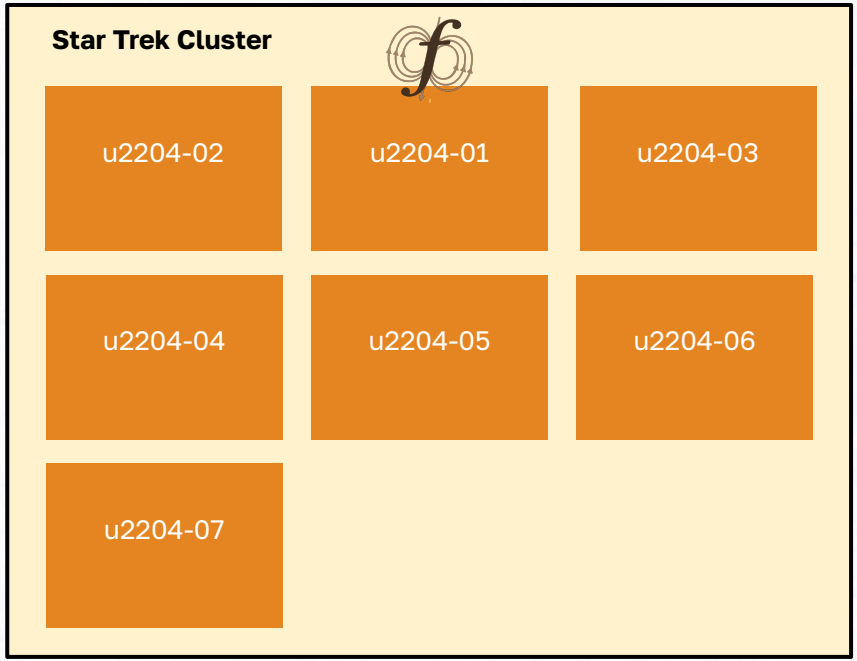
Sanity check we've chosen the right execution strategy

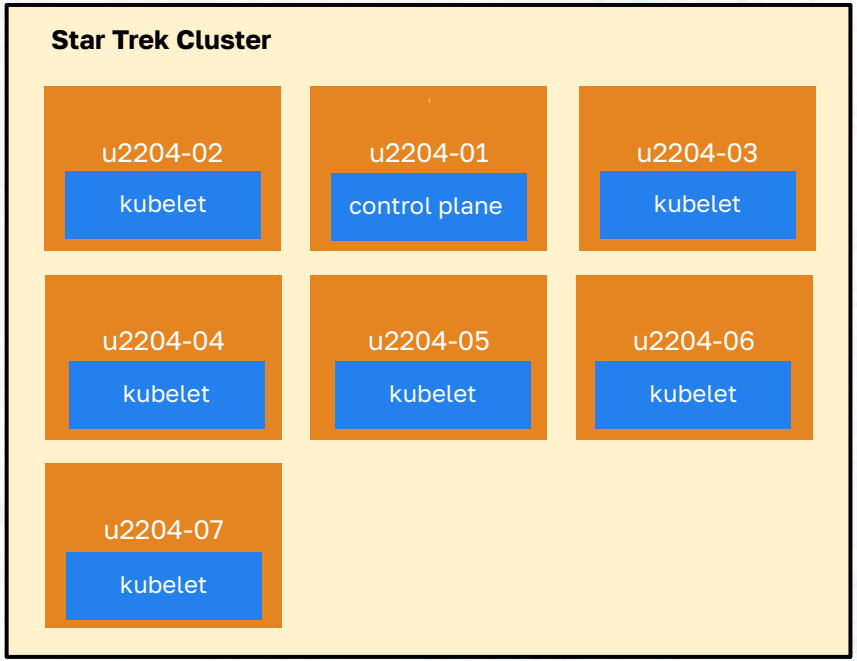




"Will the same LAMMPS (container, MPI) be faster with Usernetes or bare metal with Flux?"

1. Application performance between Flux | Usernetes





"Will the same LAMMPS (container, MPI) be faster with Usernetes or bare metal with Flux?"

1. Application performance between Flux | Usernetes

1. LAMMPS on bare metal with Flux

"Will the same LAMMPS (container, MPI) be faster with Usernetes or bare metal with Flux?"

1. Application performance between Flux | Usernetes

1. LAMMPS on bare metal with Flux
2. LAMMPS on bare metal Singularity container with Flux

"Will the same LAMMPS (container, MPI) be faster with Usernetes or bare metal with Flux?"

1. Application performance between Flux | Usernetes

1. LAMMPS on bare metal with Flux
2. LAMMPS on bare metal Singularity container with Flux
3. LAMMPS in Usernetes with the Flux Operator

"Will the same LAMMPS (container, MPI) be faster with Usernetes or bare metal with Flux?"

1. Application performance between Flux | Usernetes

1. LAMMPS on bare metal with Flux
2. LAMMPS on bare metal Singularity container with Flux
3. LAMMPS in Usernetes with the Flux Operator
4. Cases 1 and 2, but with Usernetes still running

"Will the same LAMMPS (container, MPI) be faster with Usernetes or bare metal with Flux?"

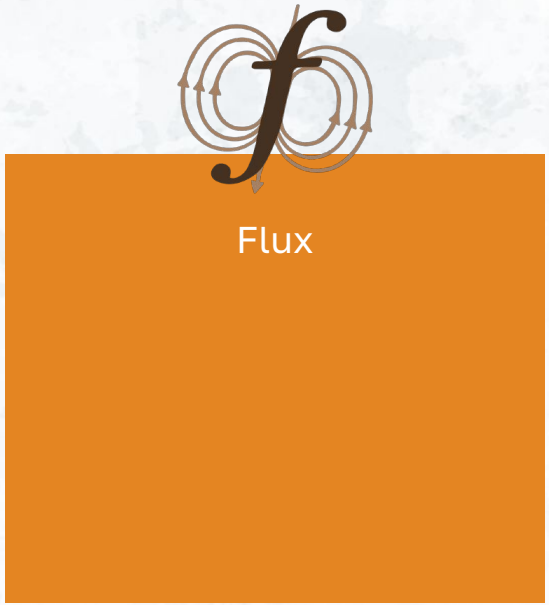
1. Application performance between Flux | Usernetes

1. LAMMPS on bare metal with Flux
2. LAMMPS on bare metal Singularity container with Flux
3. **LAMMPS in Usernetes with the Flux Operator**
4. Cases 1 and 2, but with Usernetes still running

"Will the same LAMMPS (container, MPI) be faster with Usernetes or bare metal with Flux?"

1. Application performance between Flux | Usernetes

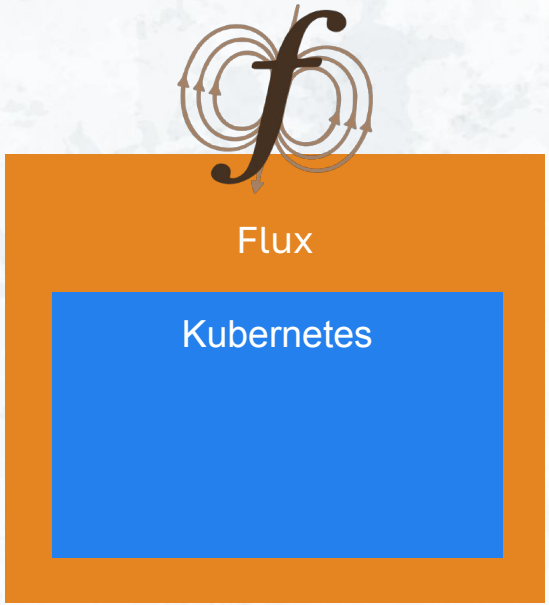
1. LAMMPS on bare metal with Flux
2. LAMMPS on bare metal Singularity container with Flux
3. **LAMMPS in Usernetes with the Flux Operator**
4. Cases 1 and 2, but with Usernetes still running



"Will the same LAMMPS (container, MPI) be faster with Usernetes or bare metal with Flux?"

1. Application performance between Flux | Usernetes

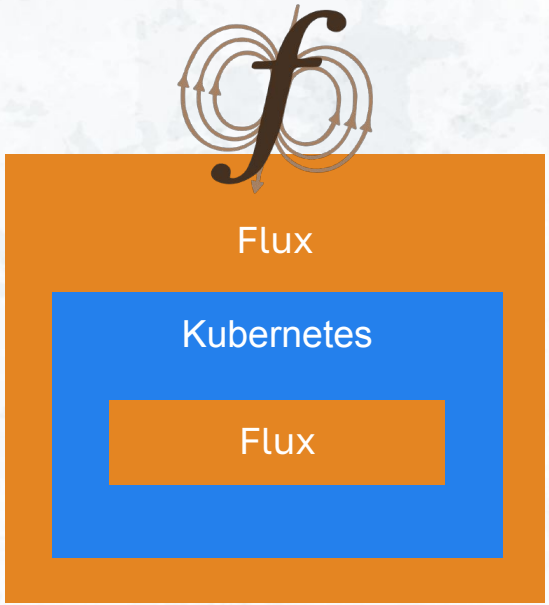
1. LAMMPS on bare metal with Flux
2. LAMMPS on bare metal Singularity container with Flux
3. **LAMMPS in Usernetes with the Flux Operator**
4. Cases 1 and 2, but with Usernetes still running



"Will the same LAMMPS (container, MPI) be faster with Usernetes or bare metal with Flux?"

1. Application performance between Flux | Usernetes

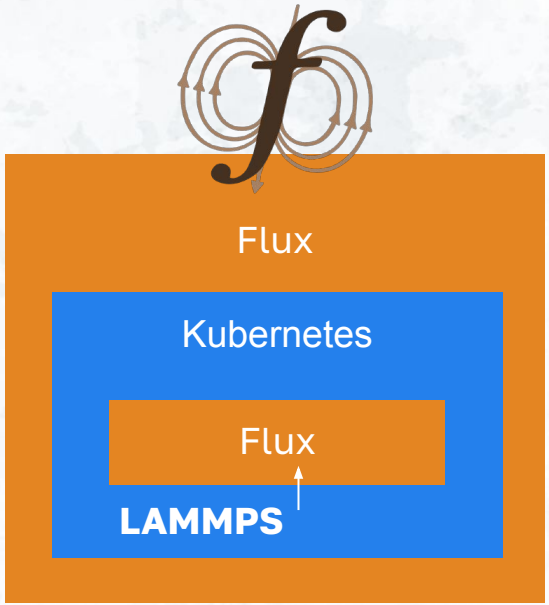
1. LAMMPS on bare metal with Flux
2. LAMMPS on bare metal Singularity container with Flux
3. **LAMMPS in Usernetes with the Flux Operator**
4. Cases 1 and 2, but with Usernetes still running



"Will the same LAMMPS (container, MPI) be faster with Usernetes or bare metal with Flux?"

1. Application performance between Flux | Usernetes

1. LAMMPS on bare metal with Flux
2. LAMMPS on bare metal Singularity container with Flux
3. **LAMMPS in Usernetes with the Flux Operator**
4. Cases 1 and 2, but with Usernetes still running



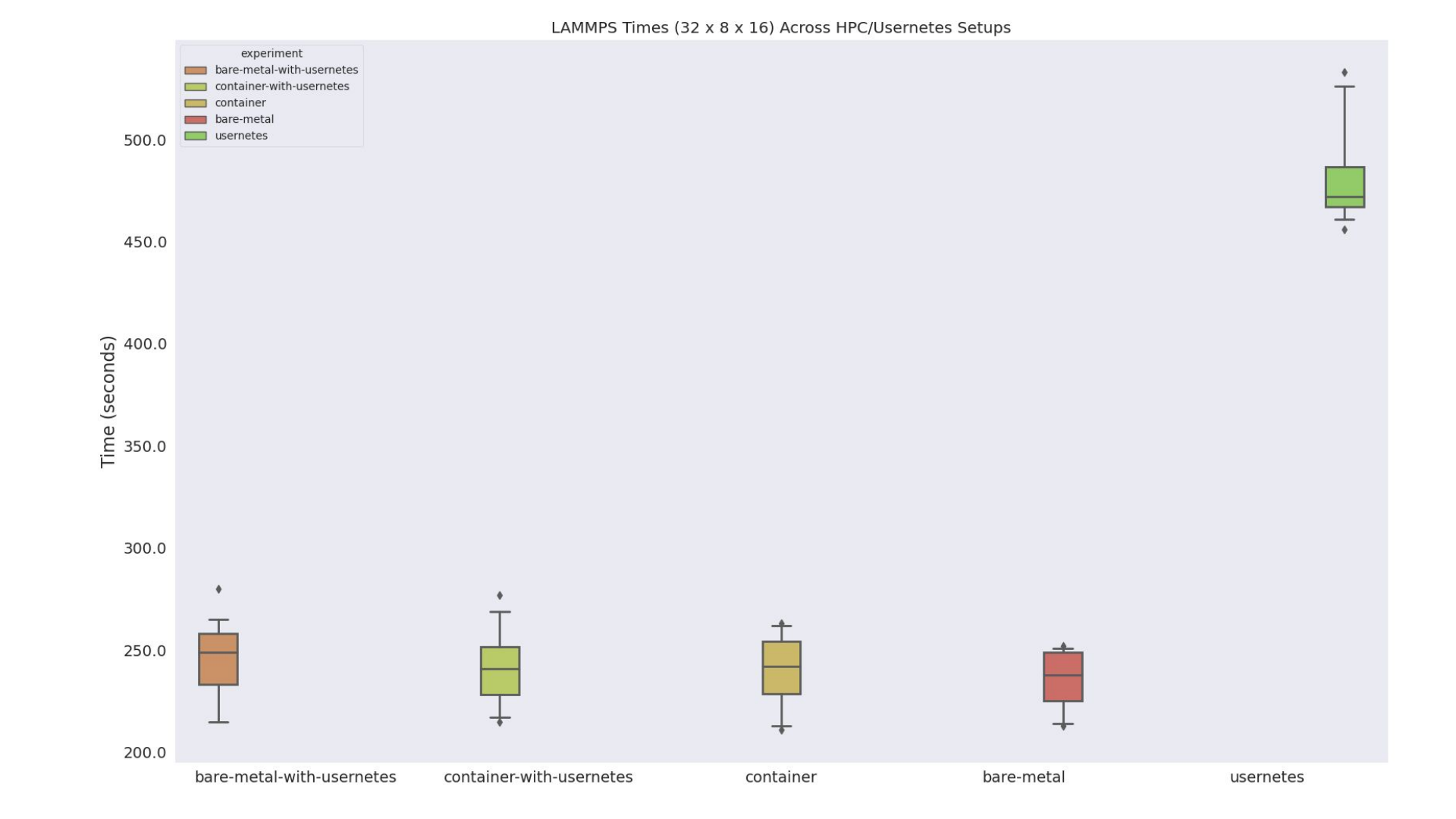
"Will the same LAMMPS (container, MPI) be faster with Usernetes or bare metal with Flux?"

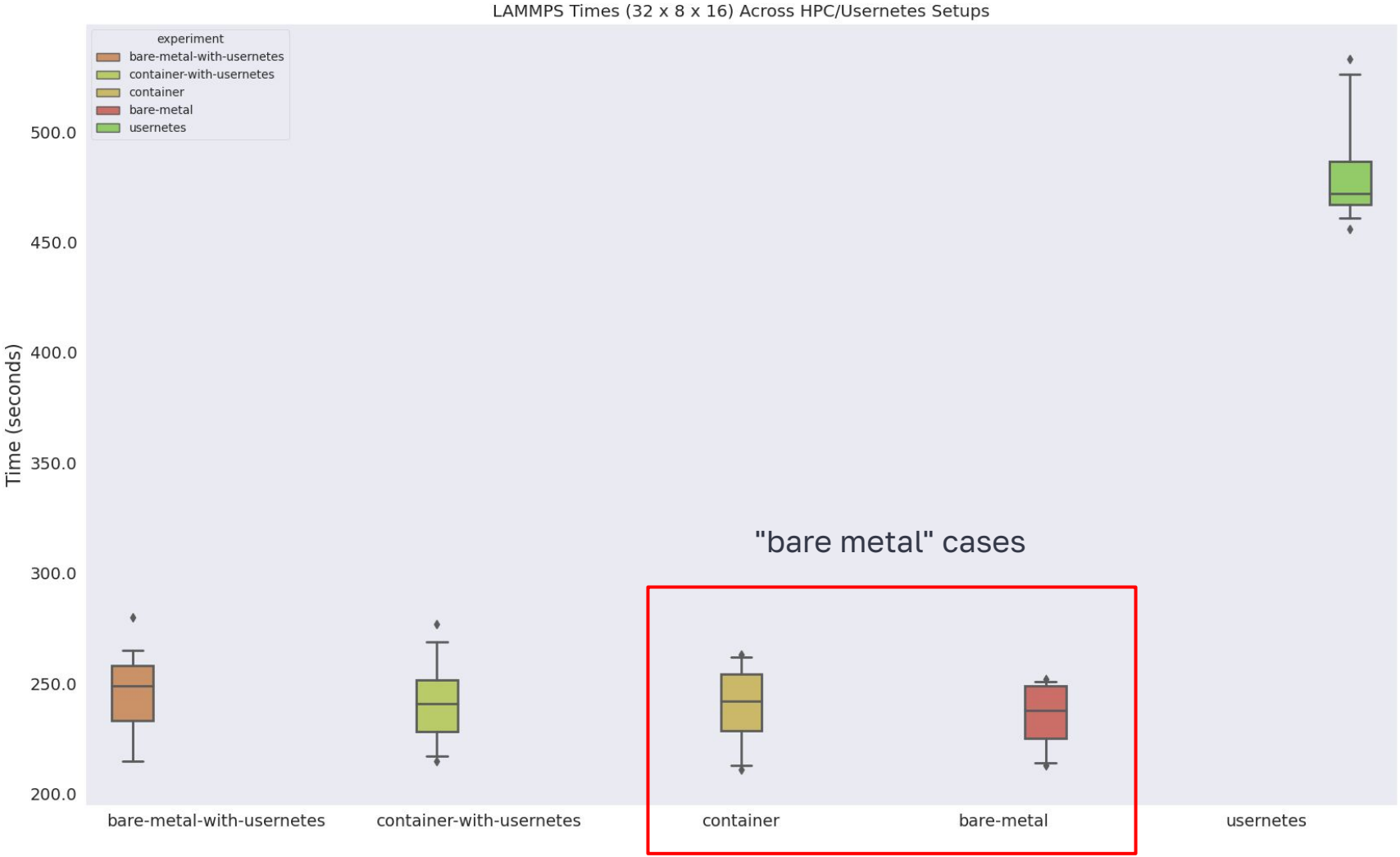
1. Application performance between Flux | Usernetes

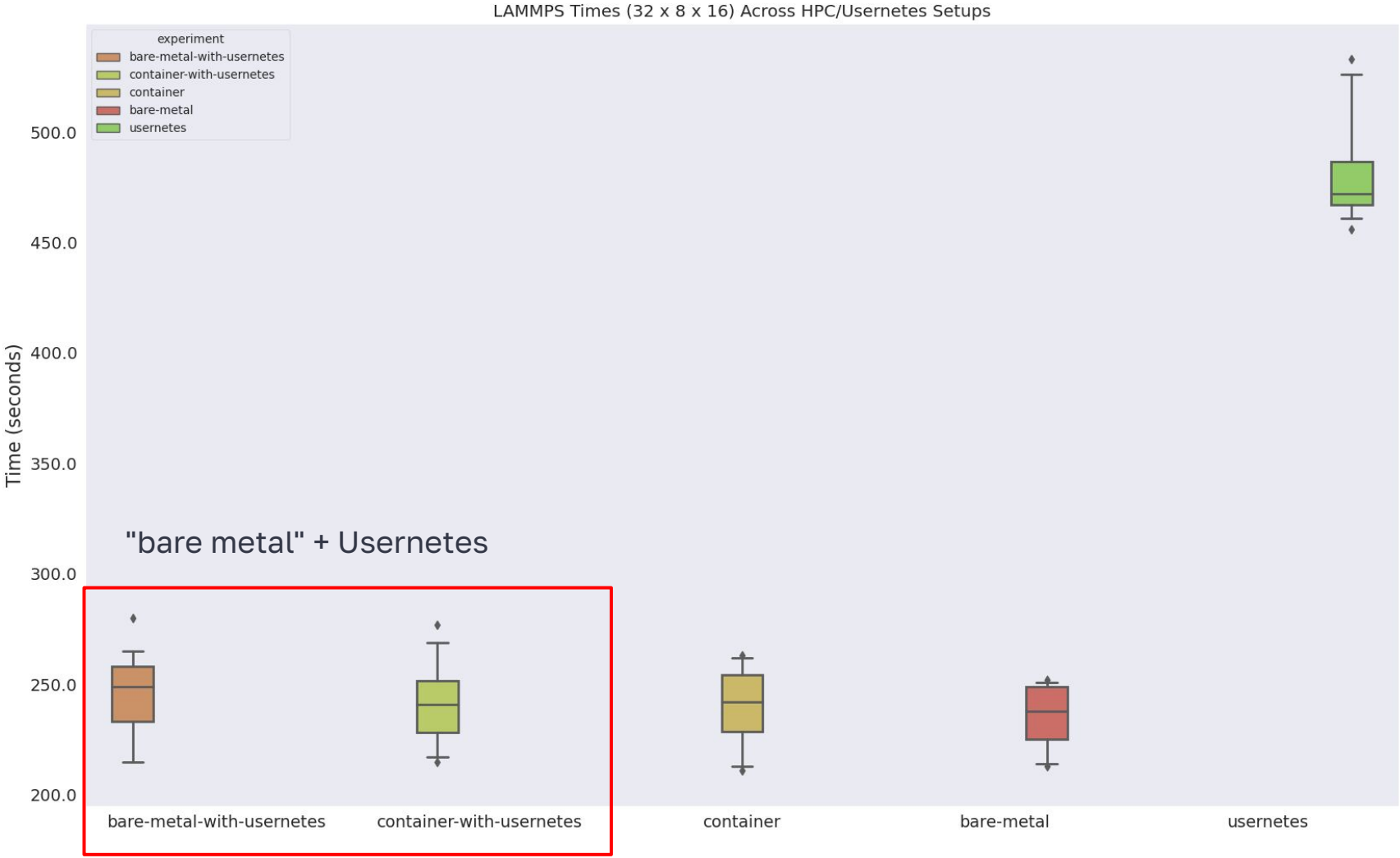
1. LAMMPS on bare metal with Flux
2. LAMMPS on bare metal Singularity container with Flux
3. **LAMMPS in Usernetes with the Flux Operator**
4. Cases 1 and 2, but with Usernetes still running

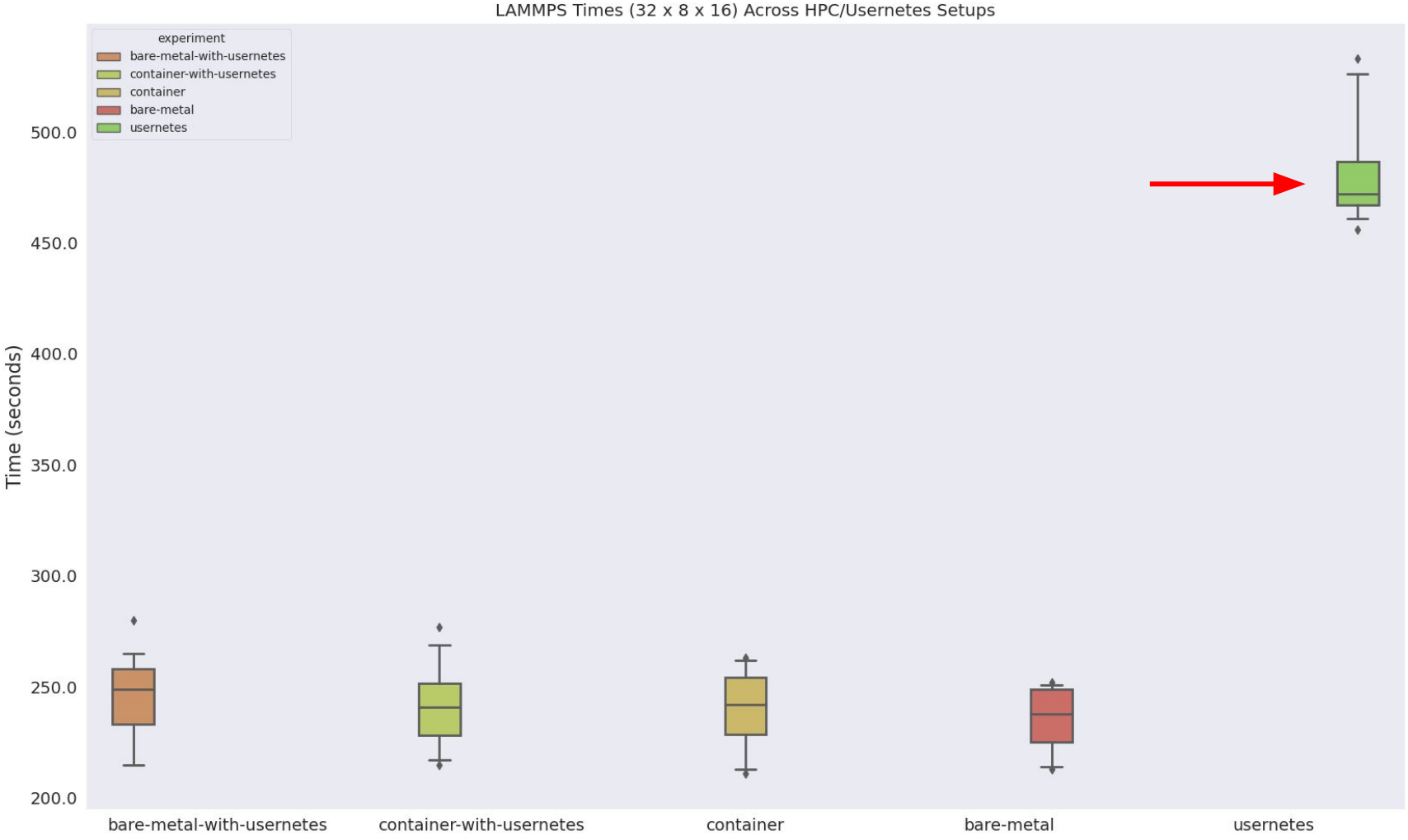
We expect LAMMPS to be slower in Usernetes because it makes MPI collective calls, and uses slirp4netns (additional processing of packets with a tap device).

"Will the same LAMMPS (container, MPI) be faster with Usernetes or bare metal with Flux?"

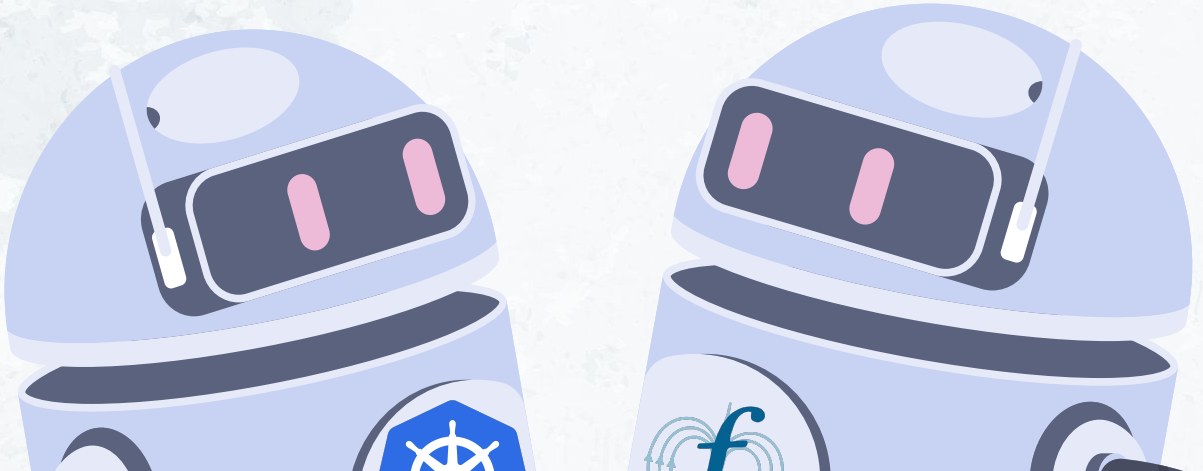

LAMMPS Times (32 x 8 x 16) Across HPC/Usernetes Setups
experiment
bare-metal-with-usernetes
container-with-usernetes
container
bare-metal
usernetes
Time (seconds)
500.0
450.0
400.0
350.0
300.0
250.0
200.0
bare-metal-with-usernetes
container-with-usernetes
container
bare-metal
usernetes


LAMMPS Times (32 x 8 x 16) Across HPC/Usernetes Setups
experiment
bare-metal-with-usernetes
container-with-usernetes
container
bare-metal
usernetes
Time (seconds)
500.0
450.0
400.0
350.0
300.0
250.0
200.0
"bare metal" cases
bare-metal-with-usernetes
container-with-usernetes
container
bare-metal
usernetes


LAMMPS Times (32 x 8 x 16) Across HPC/Usernetes Setups
experiment
bare-metal-with-usernetes
container-with-usernetes
container
bare-metal
usernetes
Time (seconds)
500.0
450.0
400.0
350.0
300.0
250.0
200.0
"bare metal" + Usernetes
bare-metal-with-usernetes
container-with-usernetes
container
bare-metal
usernetes

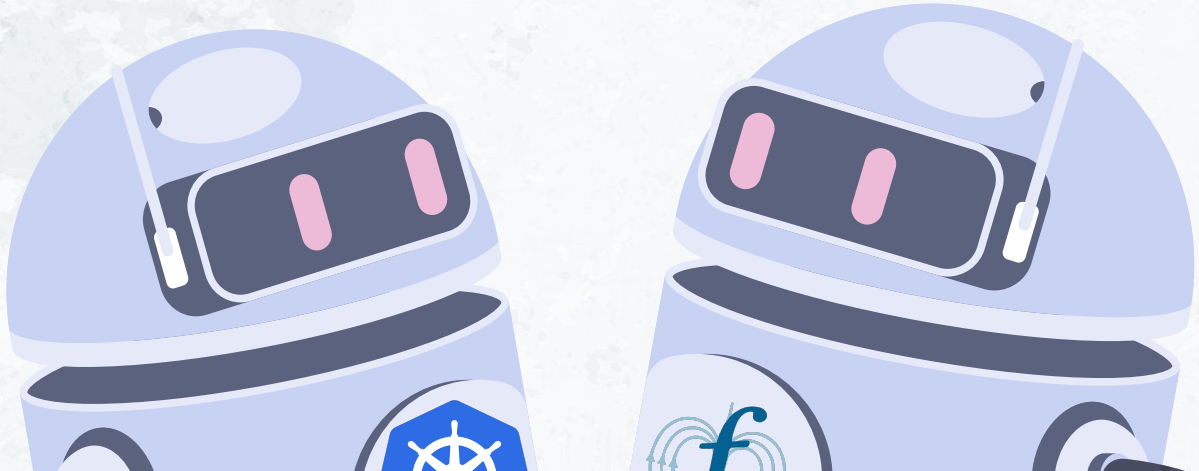

LAMMPS Times (32 x 8 x 16) Across HPC/Usernetes Setups
experiment
bare-metal-with-usernetes
container-with-usernetes
container
bare-metal
usernetes
Time (seconds)
500.0
450.0
400.0
350.0
300.0
250.0
200.0
bare-metal-with-usernetes
container-with-usernetes
container
bare-metal
usernetes

What did we learn?



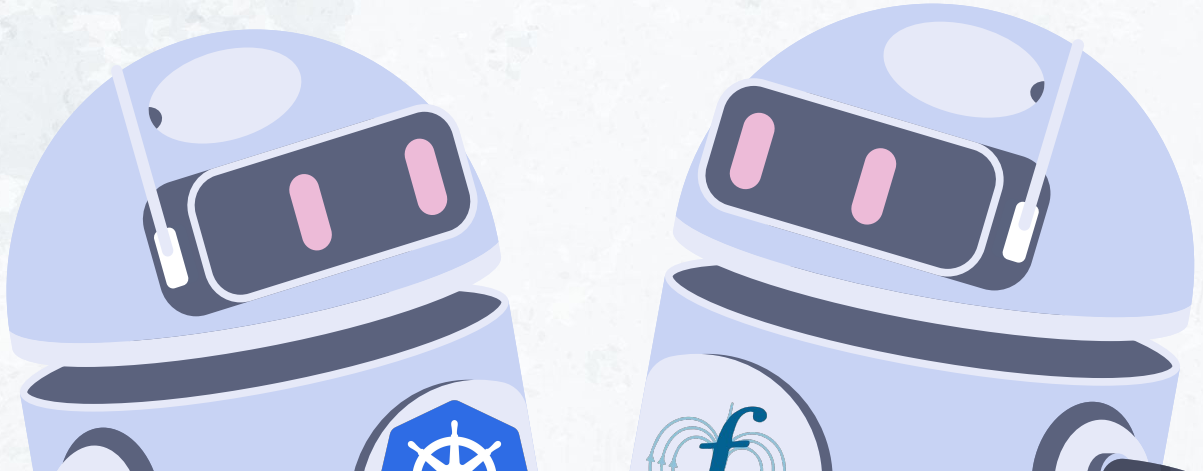
What did we learn?

For a setup like this, network sensitive stuff should run on HPC



What did we learn?

There is opportunity for improving this in Usernetes!



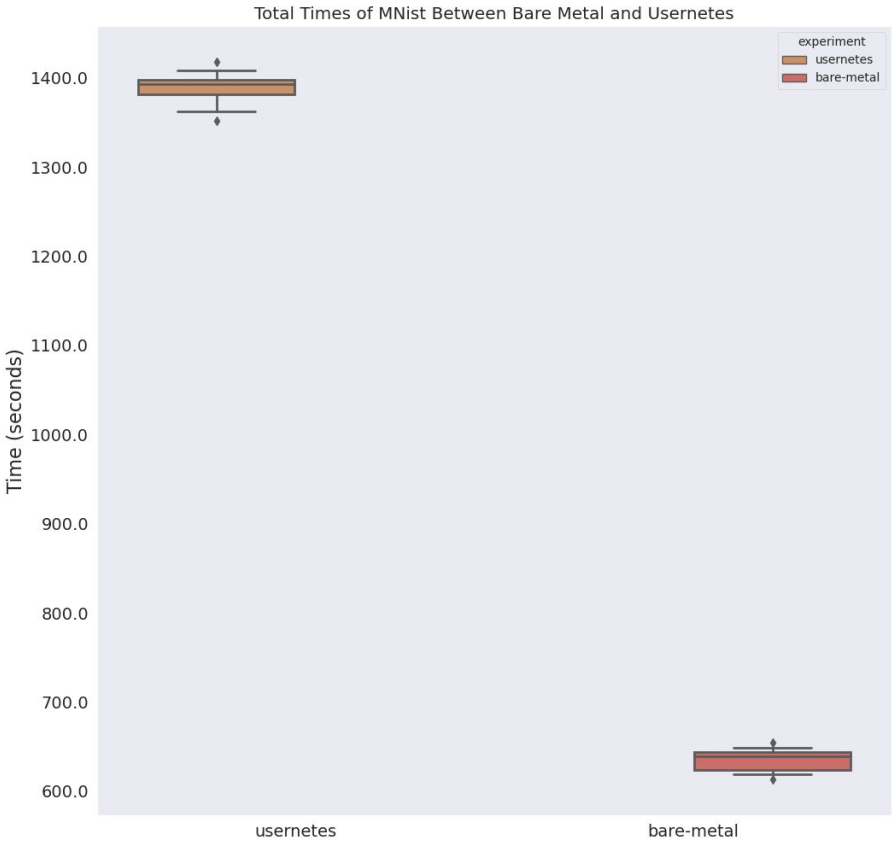
2. Distributed ML (mnist) between Flux | Usernetes

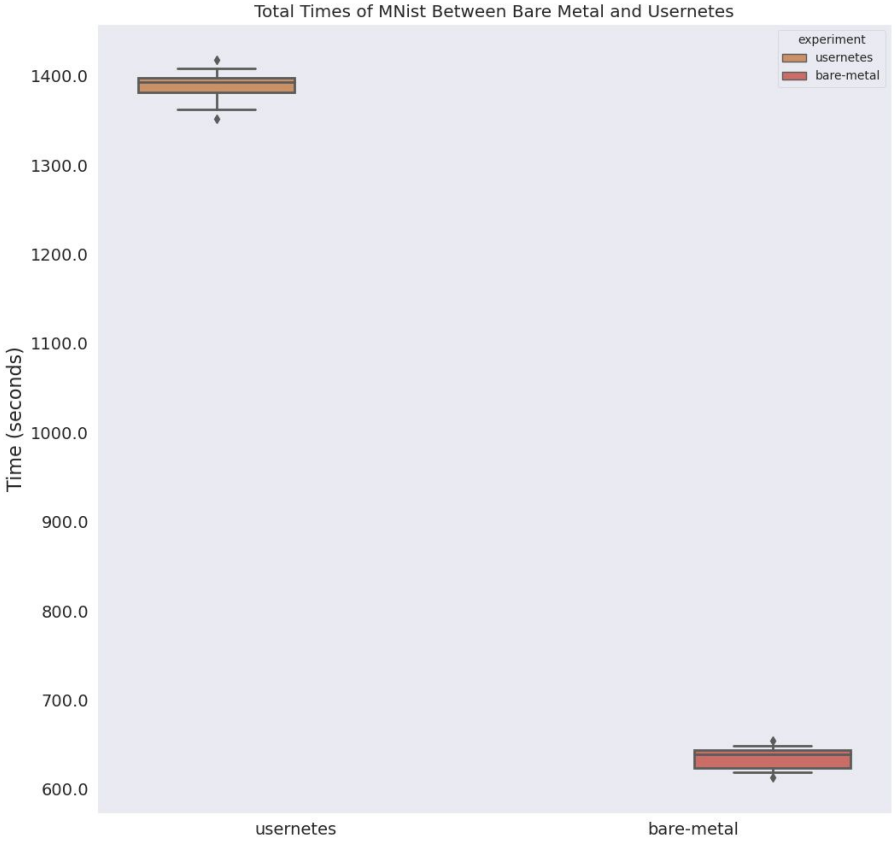
How does machine learning training compare between the two environments?

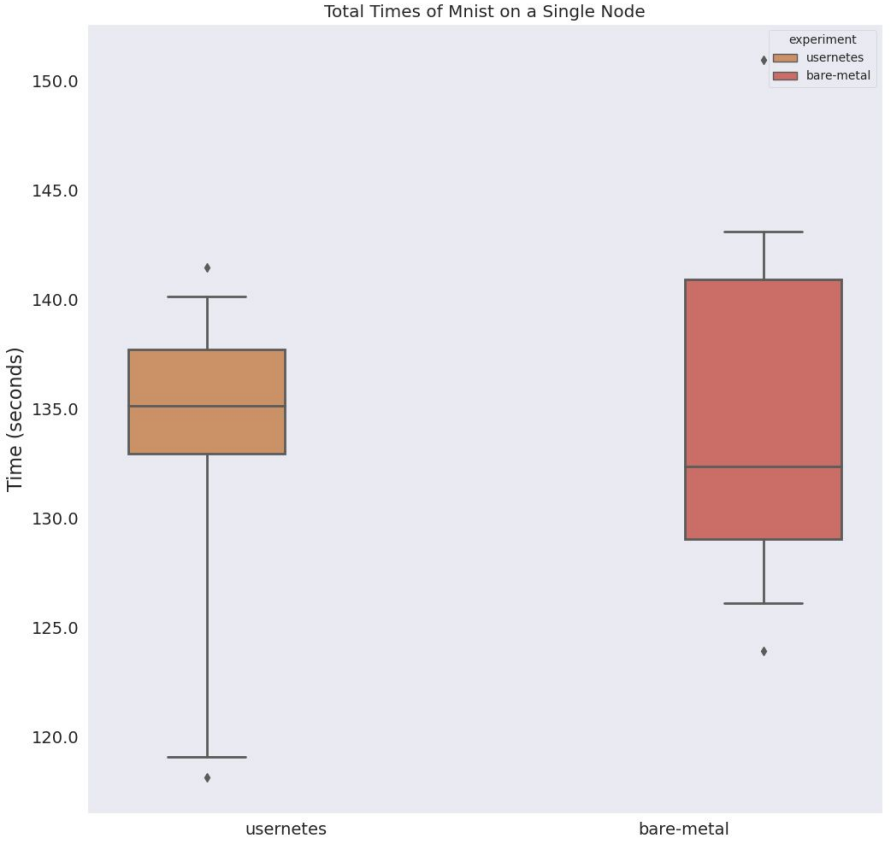
2. Distributed ML (mnist) between Flux | Usernetes

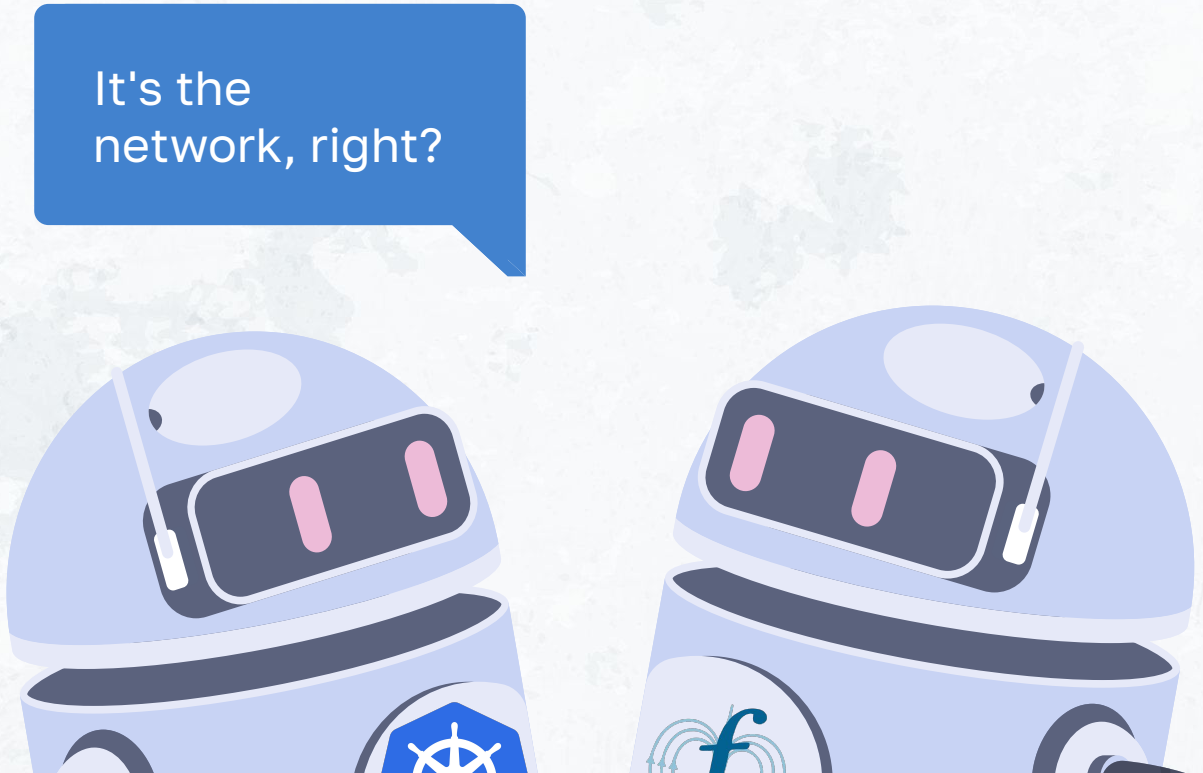
How does machine learning training compare between the two environments?

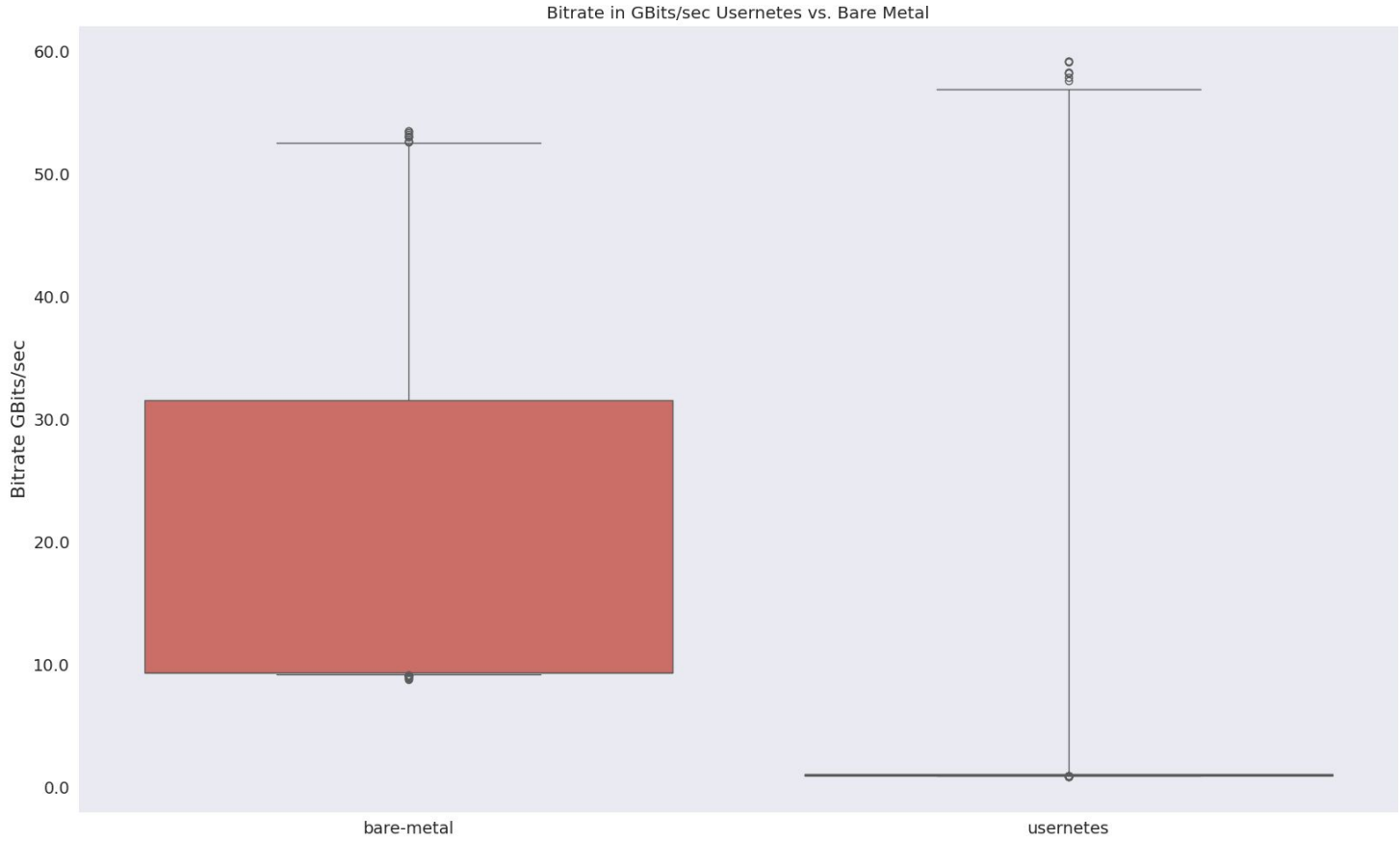
1. Mnist on bare metal with Flux vs the same in Usernetes with the Flux Operator
   a. **Distributed**: 6 nodes, 1 epoch (network **is** a variable)
   b. **One node:** 1 node, 5 epochs (network **not** a variable)


Total Times of MNist Between Bare Metal and Usernetes
experiment
usernetes
bare-metal
Time (seconds)
1400.0
1300.0
1200.0
1100.0
1000.0
900.0
800.0
700.0
600.0
usernetes
bare-metal

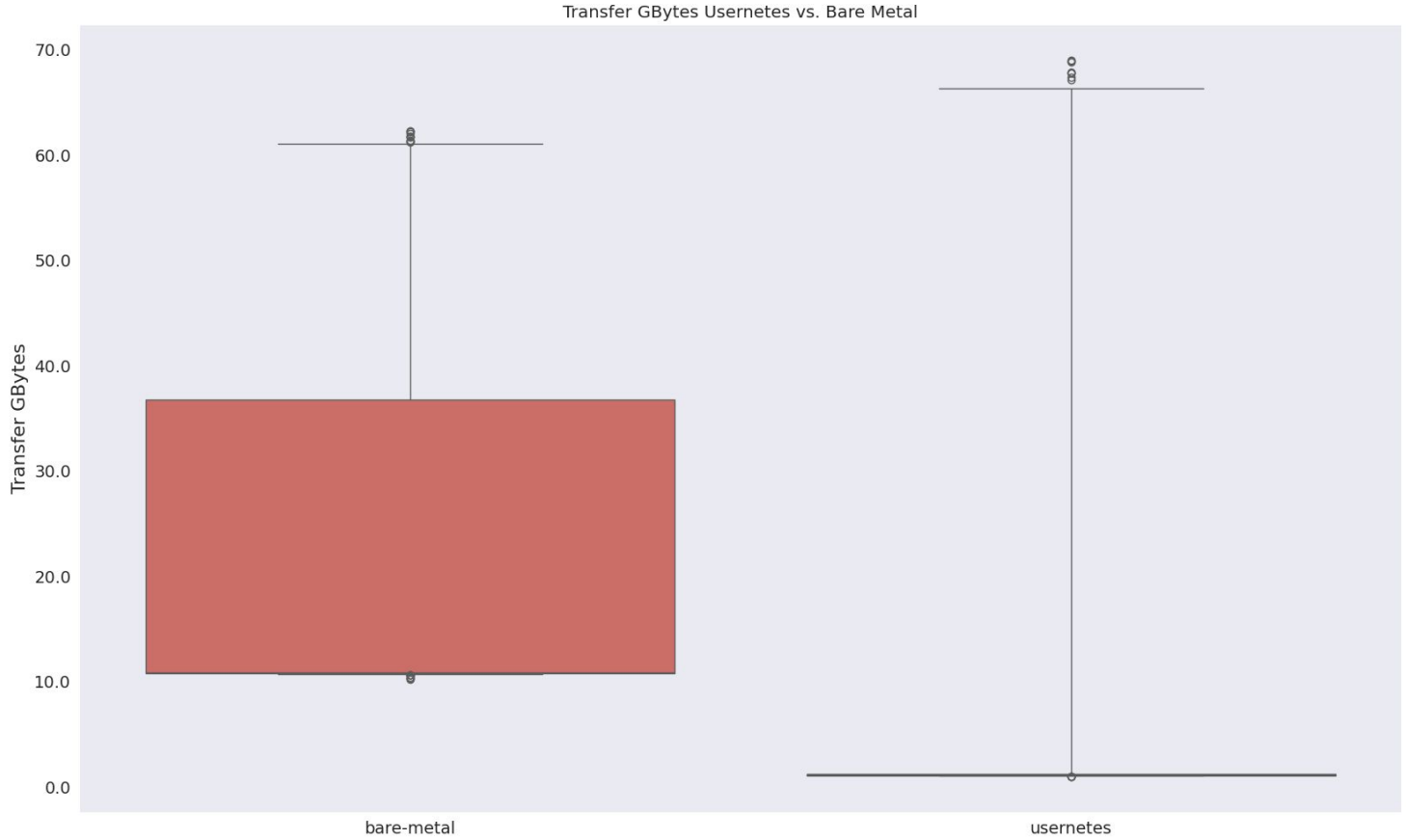

Total Times of MNist Between Bare Metal and Usernetes
experiment
usernetes
bare-metal
Time (seconds)
1400.0
1300.0
1200.0
1100.0
1000.0
900.0
800.0
700.0
600.0
usernetes
bare-metal


Total Times of Mnist on a Single Node
experiment
usernetes
bare-metal
Time (seconds)
150.0
145.0
140.0
135.0
130.0
125.0
120.0
usernetes
bare-metal

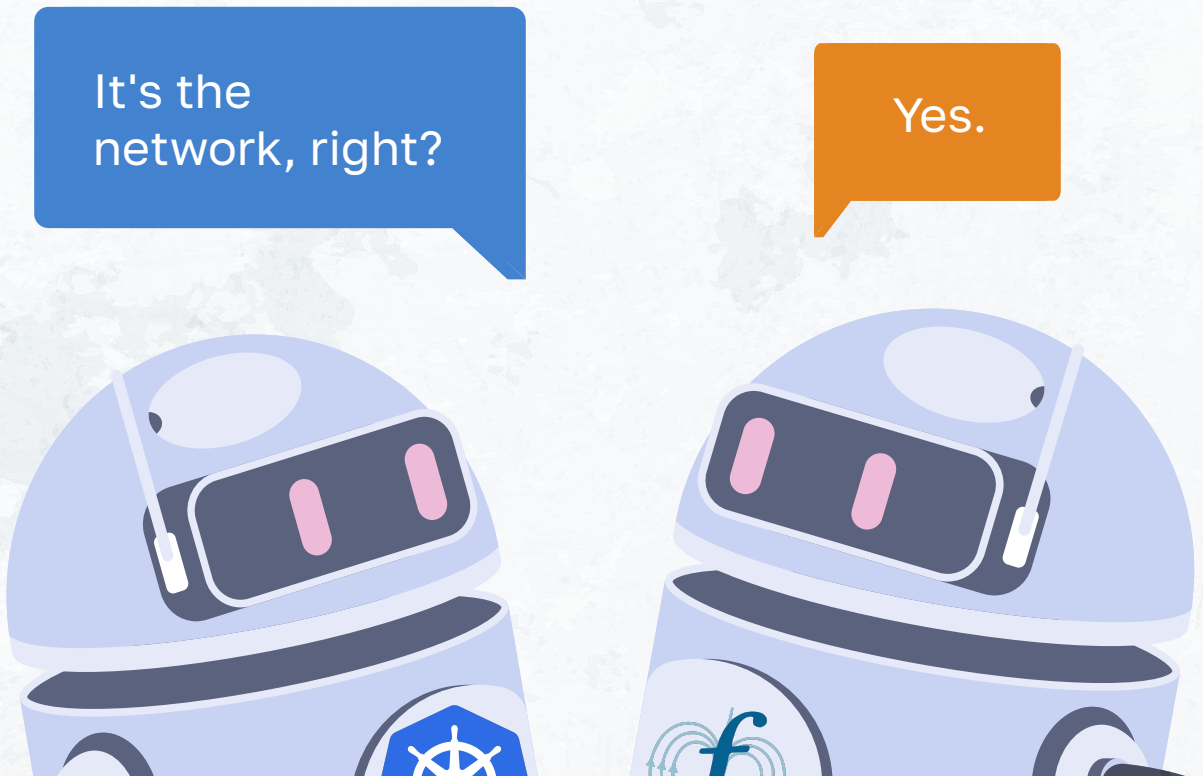

It's the
network, right?

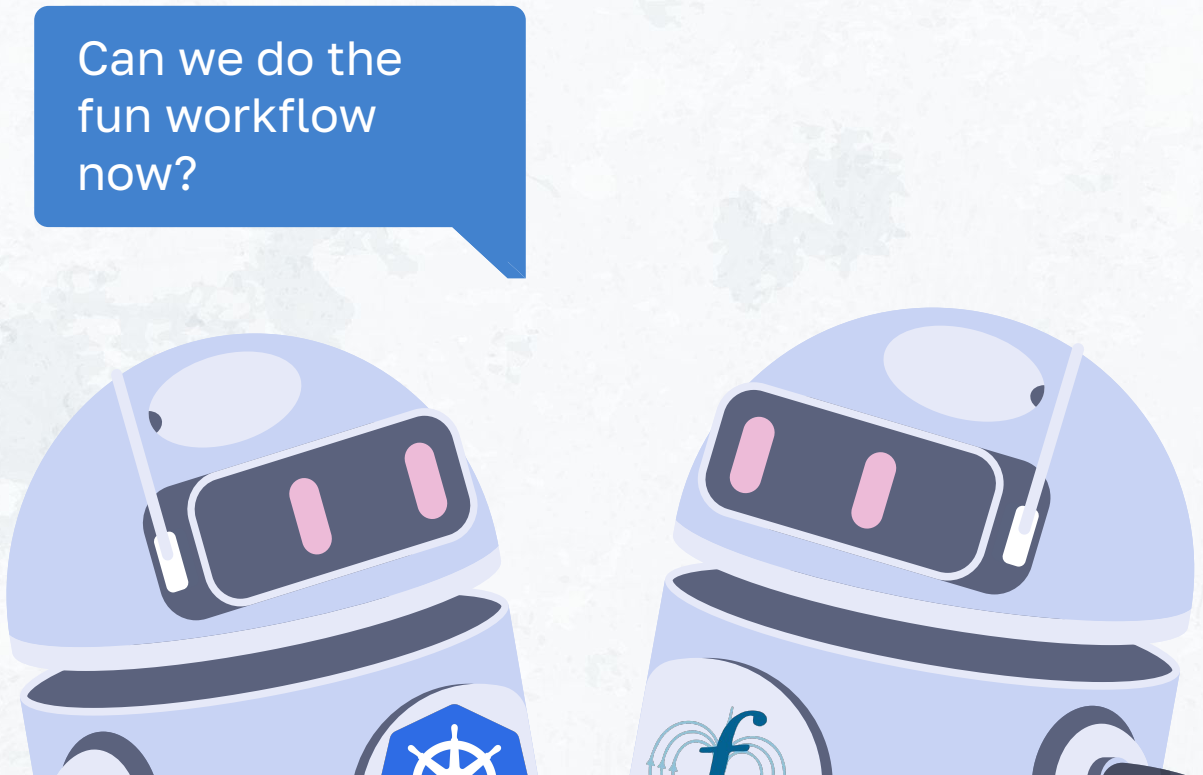


iperf3 - ~1 minute transfer from each node as a client to each other node as a server



iperf3 - ~1 minute transfer from each node as a client to each other node as a server

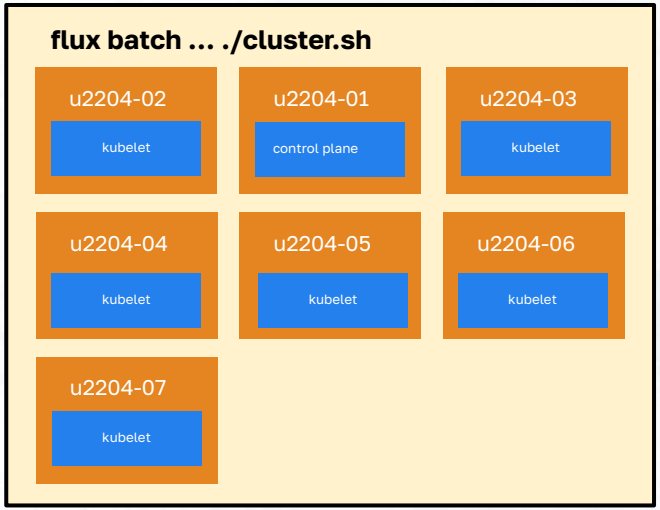

It's the network, right?
Yes.


Can we do the fun workflow now?

3. Complex workflow with HPC and services

3. Complex workflow with HPC and services

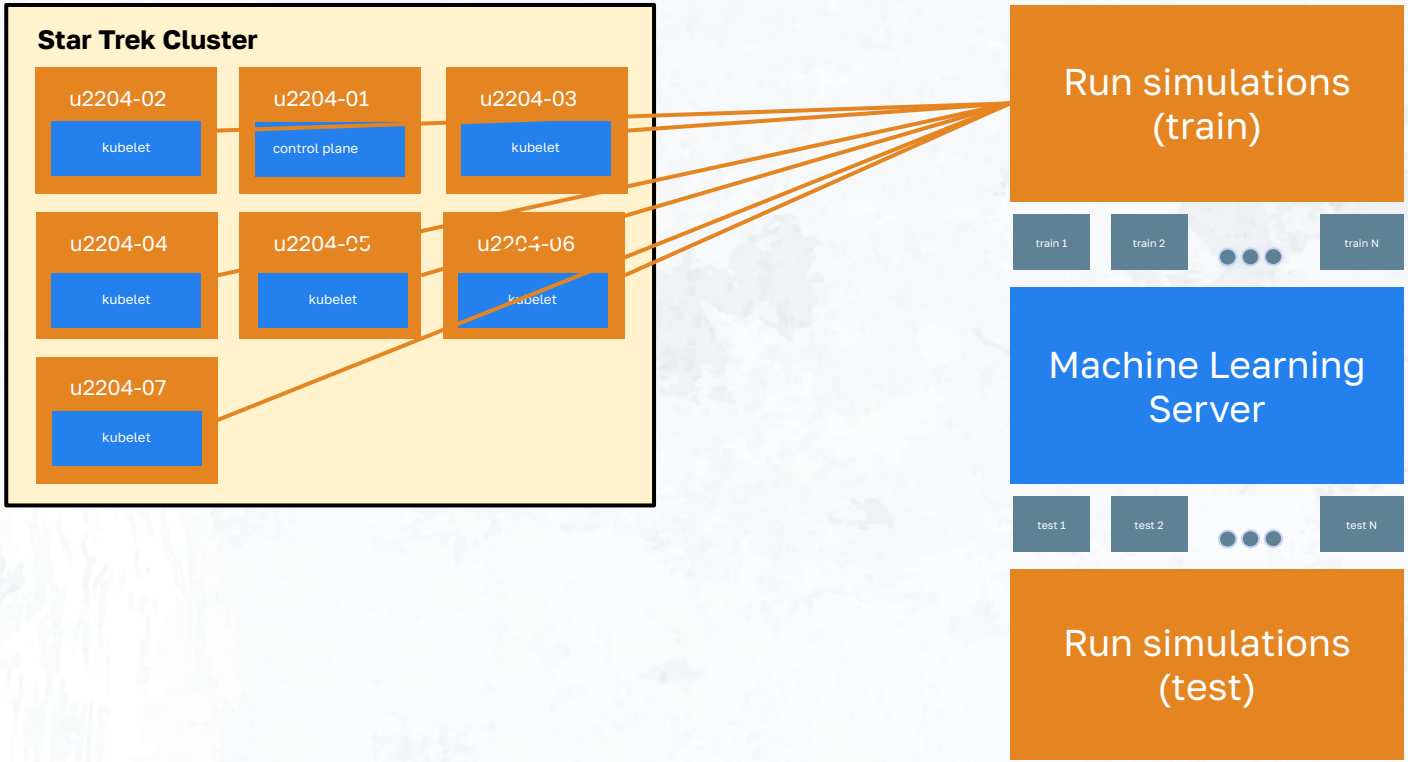
Orchestrating bare metal work and services (Usernetes) in the same batch job



1. Flux instance:
   a. Owned by running user
   b. Scoped resources (hwloc)
   c. Setup / teardown of Usernetes

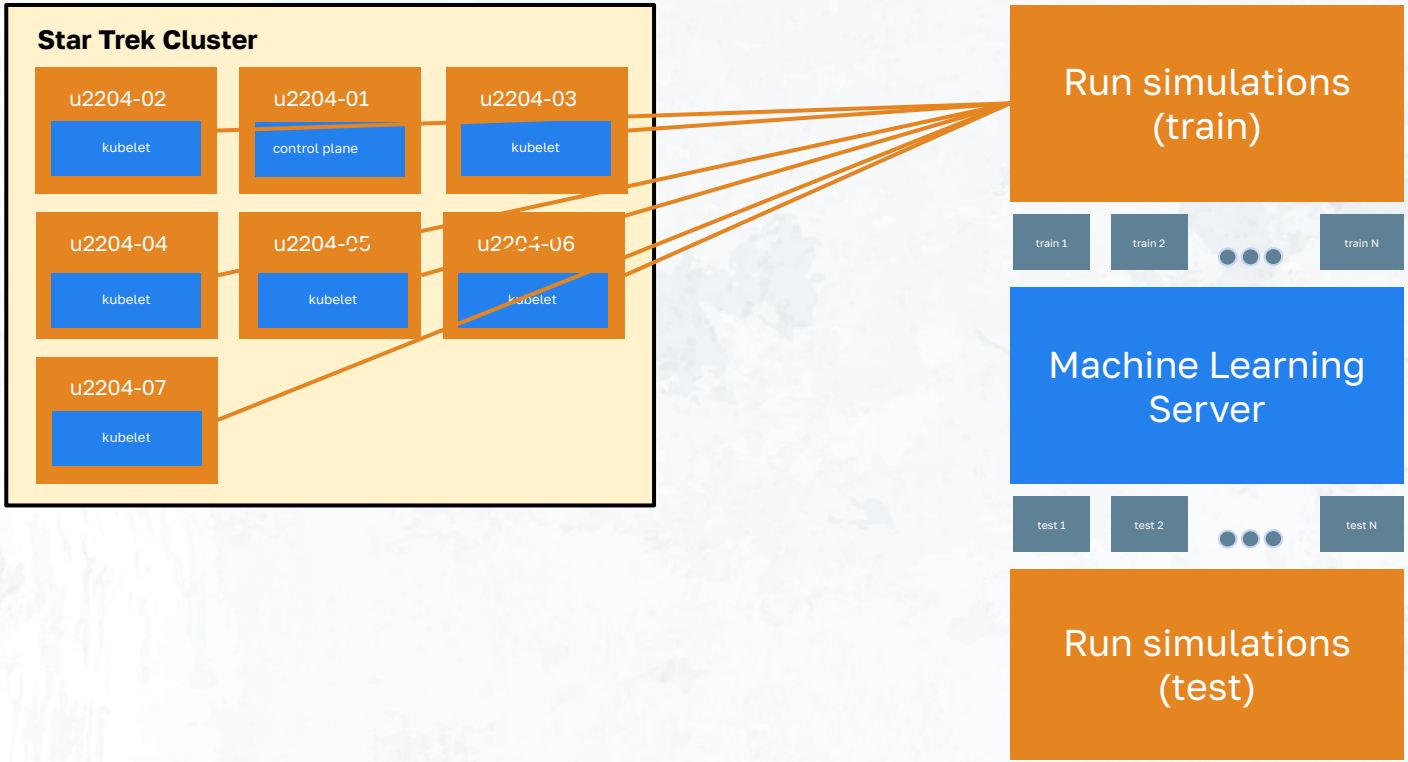
3. Complex workflow with HPC and services

Orchestrating bare metal work and services (Usernetes) in the same batch job



3. Complex workflow with HPC and services

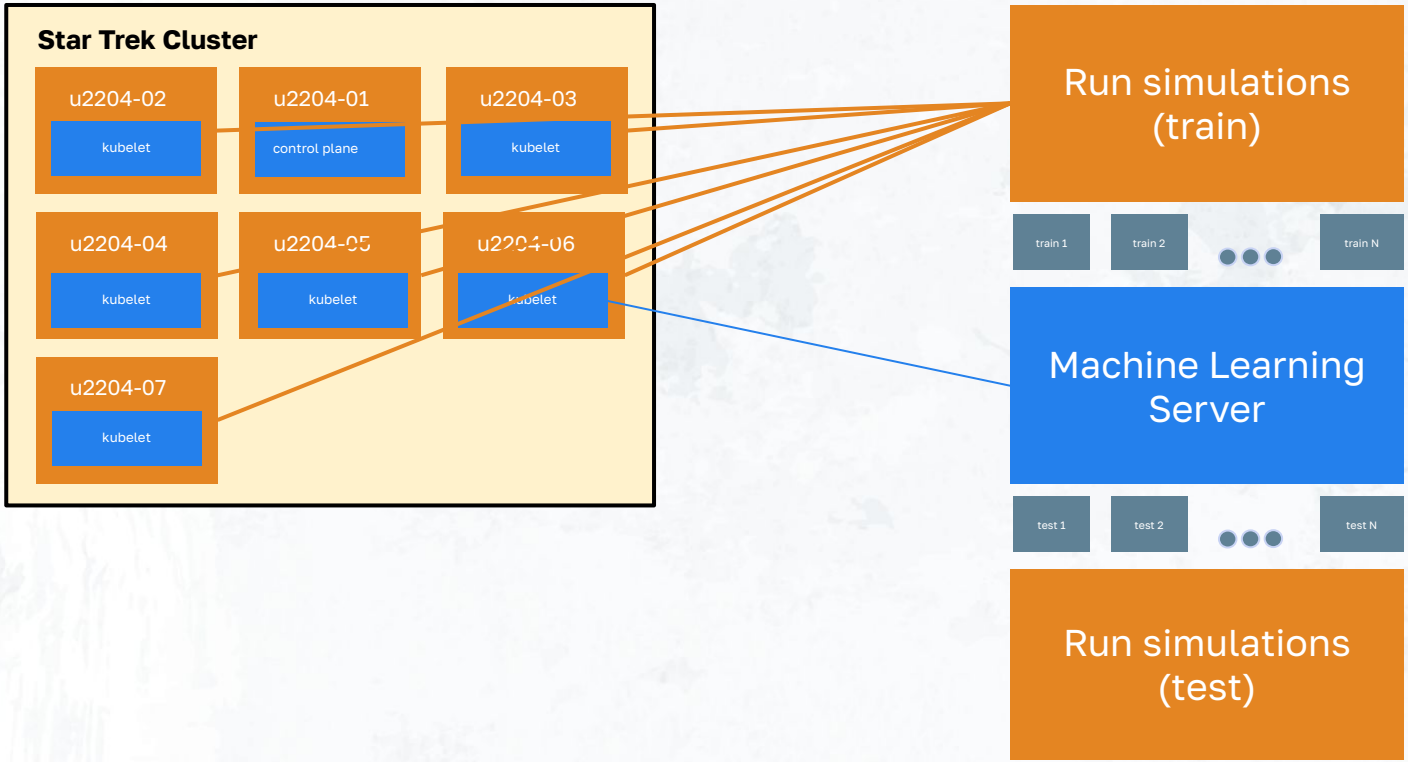
Orchestrating bare metal work and services (Usernetes) in the same batch job



LAMMPS problem sizes to predict wall time

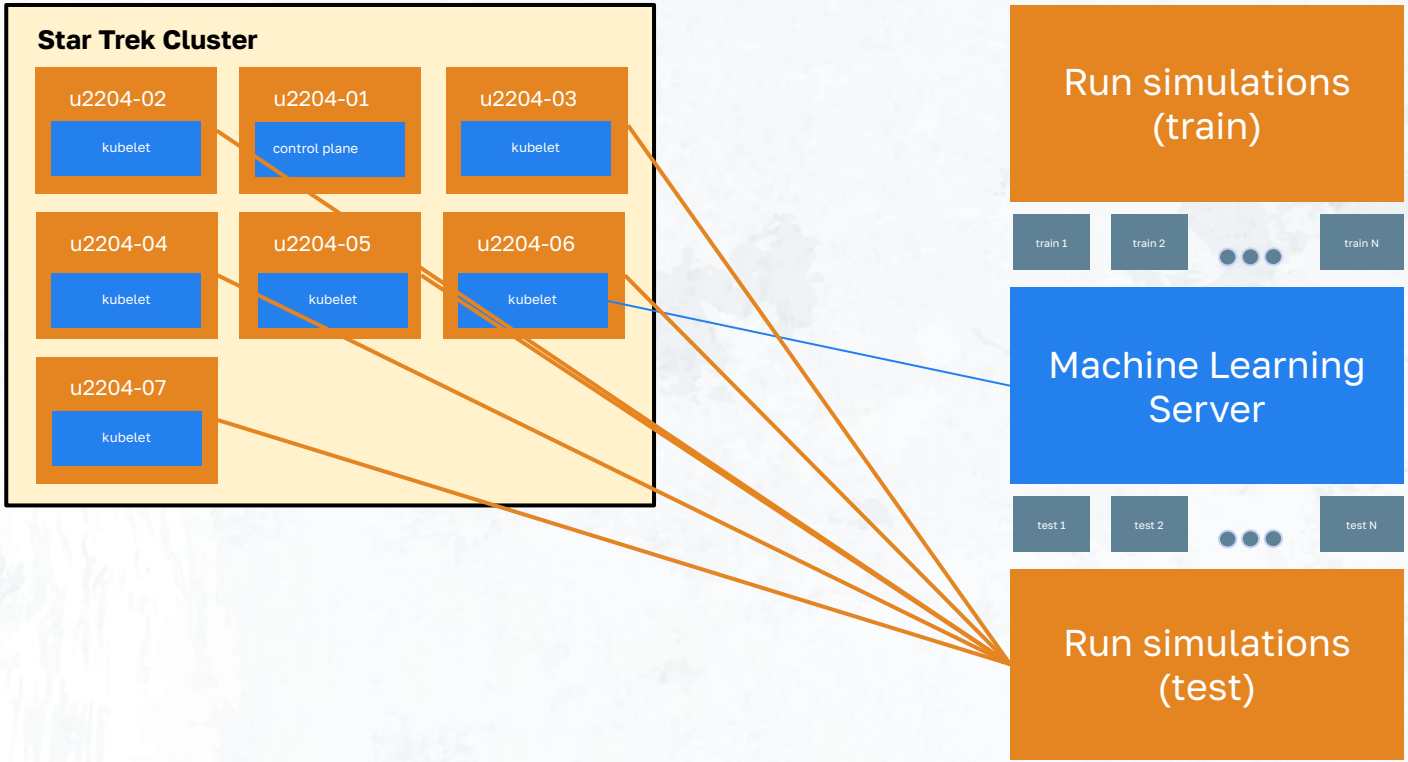
3. Complex workflow with HPC and services

Orchestrating bare metal work and services (Usernetes) in the same batch job



3. Complex workflow with HPC and services

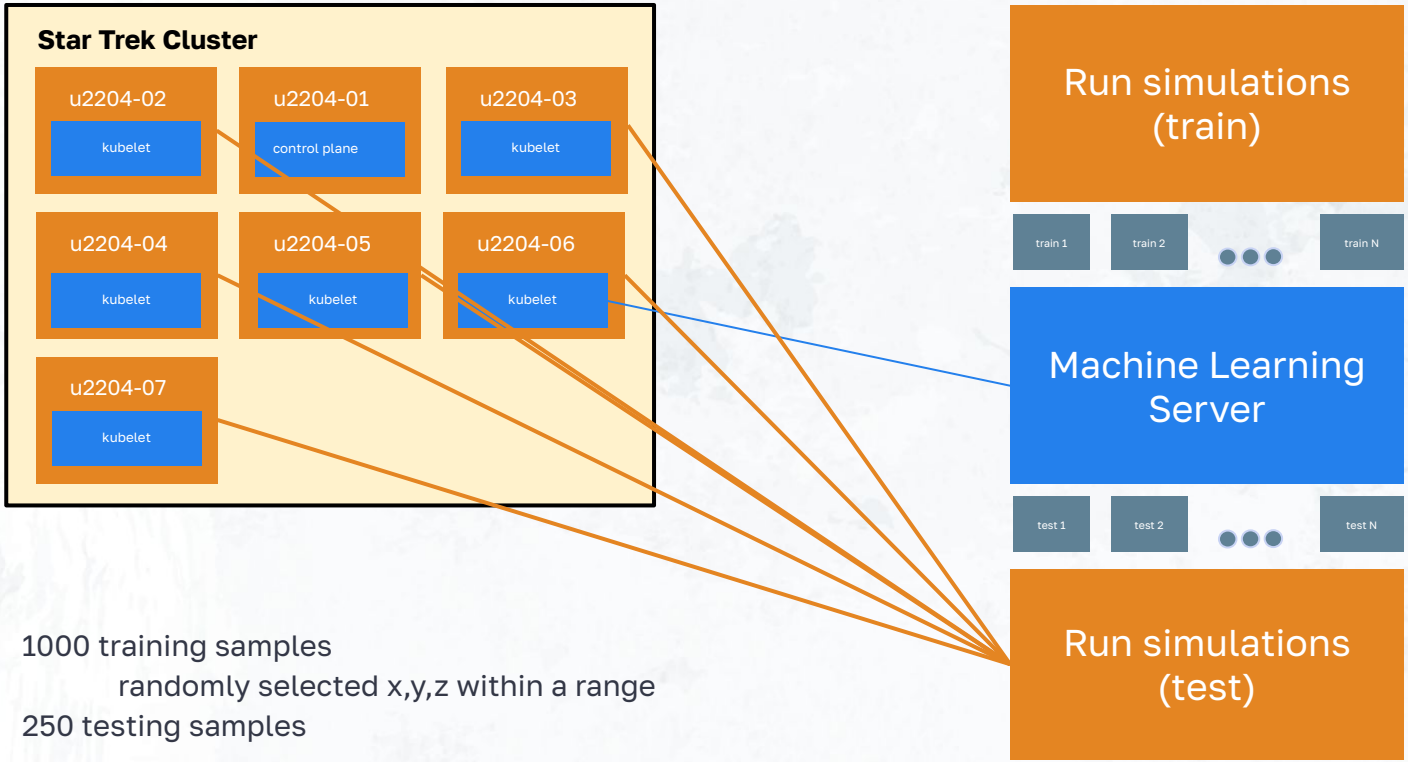
Orchestrating bare metal work and services (Usernetes) in the same batch job



LAMMPS problem sizes - what's the wall time?

3. Complex workflow with HPC and services

Orchestrating bare metal work and services (Usernetes) in the same batch job



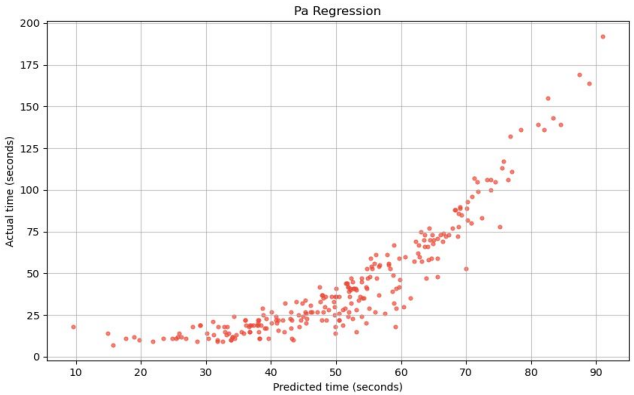
1000 training samples
randomly selected x,y,z within a range
250 testing samples

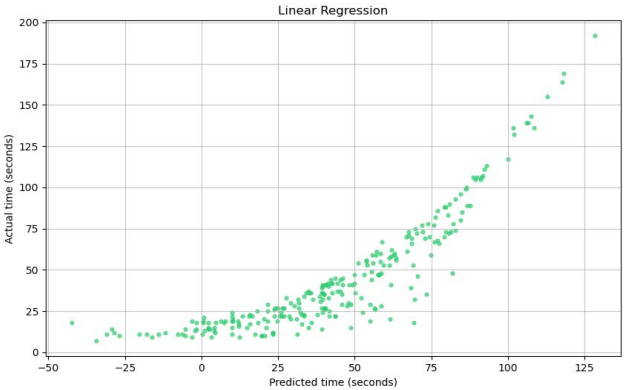
LAMMPS problem sizes - what's the wall time?

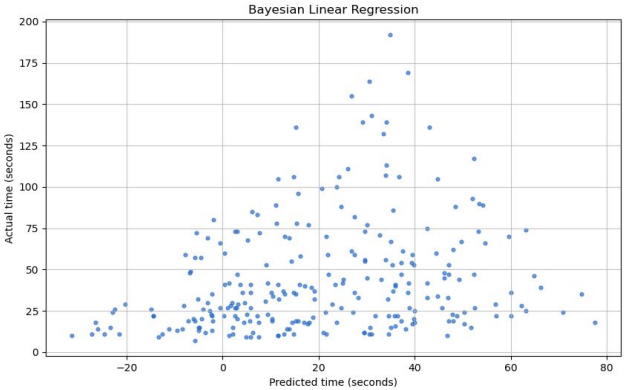
3. Complex workflow with HPC and services

The result? This prototype worked beautifully!

pa == "Passive-aggressive learning for regression"

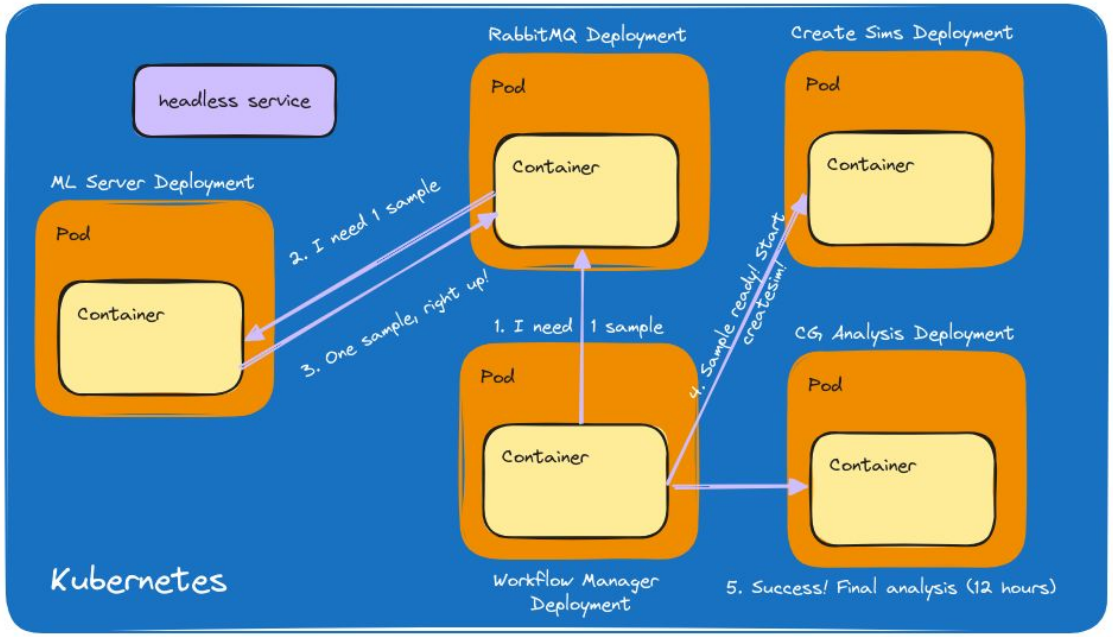






"Bare metal simulations can be run alongside services to do science!"

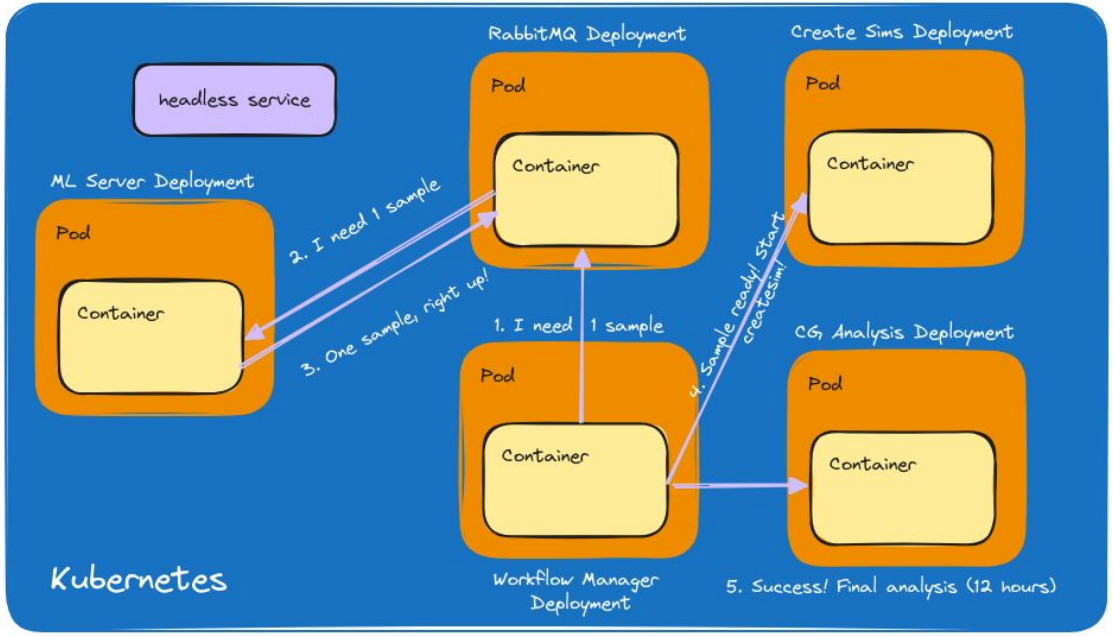
Real world heterogeneous workloads need this capability



Massively parallel Multiscale Machine-Learned Modeling Infrastructure "MuMMI"

simulate biological systems: dynamic interaction between RAS proteins and plasma membrane

Real world heterogeneous workloads need this capability



The Moomins
Finnish comic book / story series

Massively parallel Multiscale Machine-Learned Modeling Infrastructure "MuMMI"
simulate biological systems: dynamic interaction between RAS proteins and plasma membrane

(d) co-existence

Adopting technologies to make it possible for them to co-exist (and work together) in a common environment, allowing for creative collaboration and providing "the best of both worlds."

What should you remember from this talk?

What should you remember from this talk?

(a) Opportunities for collaboration

Look for shared, aligned goals that can be worked on through analogous components.

What should you remember from this talk?

(a) Opportunities for collaboration

Look for shared, aligned goals that can be worked on through analogous components.

(b) Provide handles for your components

Ensure your work is open for collaboration by way of language bindings, containers

What should you remember from this talk?

(a) Opportunities for collaboration

Look for shared, aligned goals that can be worked on through analogous components.

(b) Provide handles for your components

Ensure your work is open for collaboration by way of language bindings, containers

(c) Engagement

Show up to have a voice at the table for working groups, conferences that cross into "the other" community.

What should you remember from this talk?

(a) Opportunities for collaboration

Look for shared, aligned goals that can be worked on through analogous component.

(b) Provide handles for your components

Ensure your work is open for collaboration by way of language bindings, containers

(c) Engagement

Show up to have a voice at the table for working groups, conferences that cross into "the other" community.

(d) Mindset

Throw away adversarial mindsets for collaborative ones.

Cloud ♥ HPC

Thank you!

sochat1@llnl.gov

@vsoch

Flux Framework

https://flux-framework.org

Fluence

https://github.com/flux-framework/flux-k8s

Flux Operator

https://github.com/flux-framework/flux-operator