UUR

Sandia National Laboratories

Exceptional service in the national interest

K-Foundry: Using Kubernetes-Like Control Planes with Custom Resource Definitions to Deploy Containerized HPC Applications Across Multiple Computing Platforms

Presented by: Kevin Pedretti

Scalable System Software
ktpedre@sandia.gov

CANOPIE Workshop @ SC'24

Collaborators: Angel M. Beltre, Carlos Eduardo Arango Gutierrez, Sylvain Bernard, John M. Linebarger, Stephen L. Olivier, Andrew J. Younge, and Cory Leuninghoener

Sandia National Laboratories is a multimission laboratory managed and operated by National Technology and Engineering Solutions of Sandia, LLC, a wholly owned subsidiary of Honeywell International Inc., for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-NA0003525

SAND2024-15730PE

OUTLINE

- Introduction – my take on containers in HPC
- Computing Platform Engineering @ Sandia
  - Kubernetes / OpenShift deployments
  - Software-as-a-Service
  - Problems we have – new workload types, auth*, and integration across computing envs
- K-Foundry Overview
  - Goals, architecture, status
- Conclusion

WHO AM I?

- Filling in for Angel Beltre, primary author and leader of K-foundry R&D

- My Background
  - Computer Engineer at Sandia (2001 – Present)
  - Focused on Scalable System Software and HPC systems – OS geek
  - Got interested in ***Virtualization in HPC*** ~ 2008 – 2010 timeframe, Kitten & Palacios
    - Mix advantages of Linux with ability to run custom HPC operating systems on demand
    - Showed could scale virtualized tightly-coupled physics sims to > 10K nodes with minimal overhead (< 1%)
    - Today: Server virtualization hasn't caught on in HPC, *still niche*
  - Got interested in ***Containers in HPC*** ~ 2015 timeframe
    - Docker was all the rage in the cloud & web application communities
    - Today: Lots of excellent work on containers in HPC, *still niche* but better uptake than VMs
  - Presently leading Sandia's ***"Computing-as-a-Service"*** (CaaS) thrust in our new **"Computing Platform Engineering"** (CPE) initiative

Kitten OS

Informal Poll:
What percentage of HPC workloads at your site are containerized?

Best Guess, Greater Than:

0%

10%

20%

30%

40%

50%

60%

70%

80%

90%

100%

Informal Poll:
What percentage of HPC workloads at your site are containerized?

Best Guess, Greater Than:

0%
10%
20%
30%
40%
50%
60%
70%
80%
90%
100%

Sandia Today

CI Testing
Researchers
Code teams who build containers

Aspirational Goal

To get there, would likely require system-level approach.

Users run in a containerized environment by default, invisible.

Kubernetes – 100%
Openshift – 100%

Core OS – 100%

CEA Pcocc – 100%
Warewulf4 – ~100%

WHAT IS DRIVING CONTAINER ADOPTION AT SANDIA

- Enterprise and other computing (not HPC)
- HPC software packaging and distribution
- Digital Engineering Transformation (minds are open to trying new things)
  - Modeling & Simulation ***"Software-as-a-Service"***
  - New ***"Computing Platform Engineering"*** initiative
- Artificial intelligence / Large Language Model related workloads
  - vLLM, TensorRT-LLM, Chainlit, Grobid, MySQL, Postgres, Milvus, Prometheus, Grafana, Jupyter Hub, …

OUTLINE

- Introduction – my take on containers in HPC
- Computing Platform Engineering @ Sandia
  - Kubernetes / OpenShift deployments
  - Software-as-a-Service
  - Problems we have – new workload types and integration across systems
- K-Foundry Overview
  - Goals, architecture, status
- Conclusion

COMPUTING PLATFORM ENGINEERING @ SANDIA

- Streamline Modeling & Simulation delivery to end users
  - Software-as-a-Service, zero install, easy to use, …
  - Rapid and interactive design space exploration, early in design process
- Rearchitect computing infrastructure for integrated HPC & Cloud / AI
- Build reusable components and improve self-service capabilities
- Enable “Programming the Datacenter”

CLOUD VS. HPC – DIFFERENT USAGE MODELS, CUSTOMS, AND PRACTICES

1. They use the same underlying technology – servers, storage, and networks
2. Cloud has 100's of services, HPC has ~ 1 **(HPC is the service)**
3. Cloud has APIs for managing all infrastructure and services, **(HPC APIs are ad hoc)**
4. Cloud uses token-based authentication, **(HPC uses passwords)**
5. Cloud runs the customer's software stack, **(HPC runs the facility's SW stack)**
6. Cloud charges by the hour (encouraging paranoia), **(HPC cycles are free)**

Cross-Pollination of Cloud & HPC Mutually Beneficial

CLOUD VS. HPC – DIFFERENT USAGE MODELS, CUSTOMS, AND PRACTICES

1. They use the same underlying technology – servers, storage, and networks
2. Cloud has 100's of services, HPC has ~ 1 **(HPC is the service)**
3. Cloud has APIs for managing all infrastructure and services, **(HPC APIs are ad hoc)**
4. Cloud uses token-based authentication, **(HPC uses passwords)**
5. Cloud runs the customer's software stack, **(HPC runs the facility's SW stack)**
6. Cloud charges by the hour (encouraging paranoia), **(HPC cycles are free)**

Cross-Pollination of Cloud & HPC Mutually Beneficial

THE ROAD TO KUBERNETES & “COMPUTING-AS-A-SERVICE”

Container orchestration as a service has reached critical mass @ Sandia

2009 – Research

2022

Kubernetes

Sandia On-Premise
Azure Stack Hub

Production

2023

Red Hat OpenShift

Production OpenShift
+
GPUs for Sims & AI

Secure Envs

2024

Red Hat OpenShift

SCN OpenShift
&
Enterprise OpenShift

R&D

2023

ADVANCED ARCHITECTURE TESTBED PROGRAM

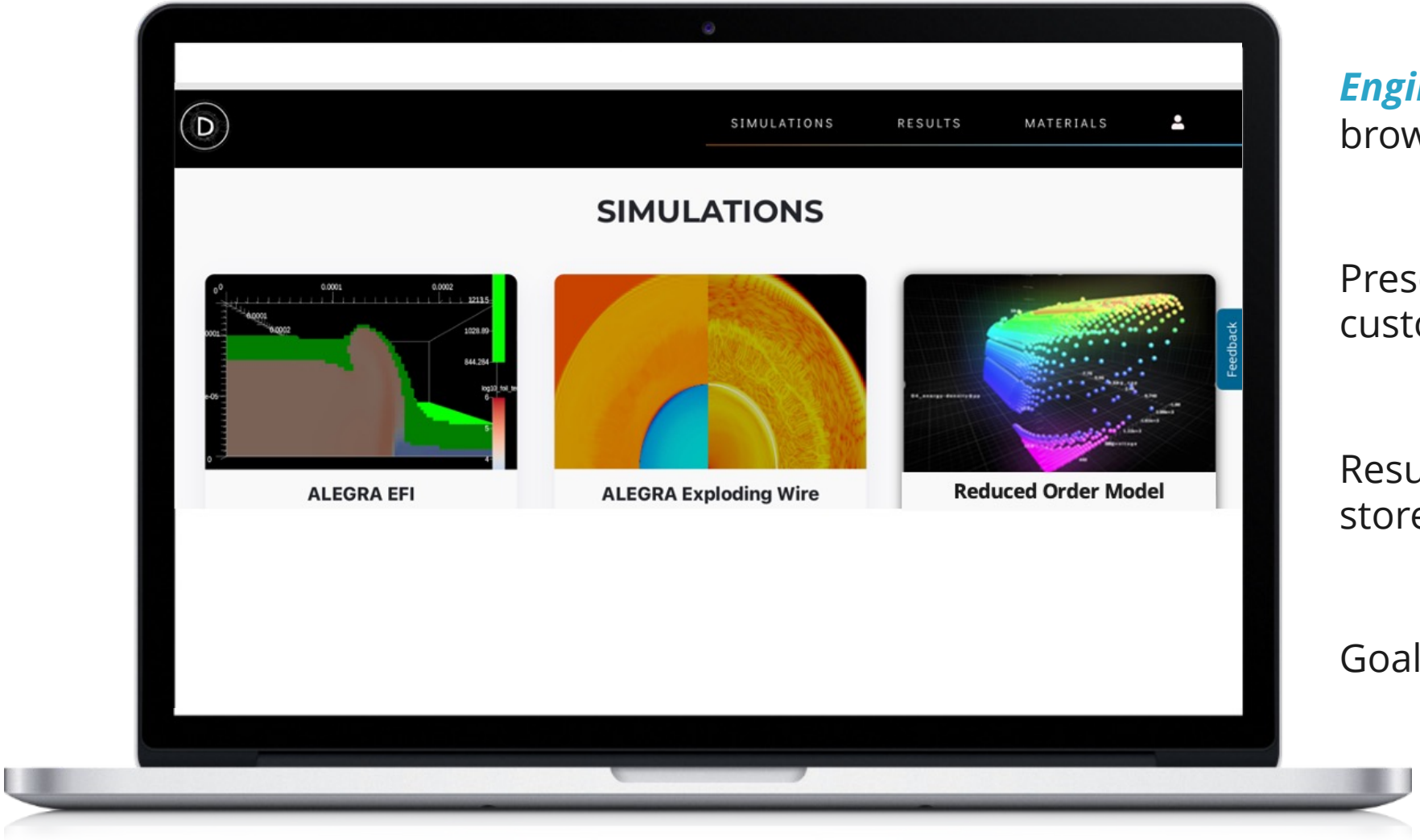
OpenShift Testbed
+
GPUs & InfiniBand

FY25
SNL
ASC
CaaS
Strategy

Early DetNet Protyping

Compute:	***Containers***
Storage:	***Persistent Volumes + S3***
Networking:	***Ingress Routes***
Authentication:	*GitLab OAUTH2*
Package Mgmt.:	*Helm*
Bridge 2 HPC:	*GitOps + Jacamar CI*

MOD-SIM SOFTWARE-AS-A-SERVICE



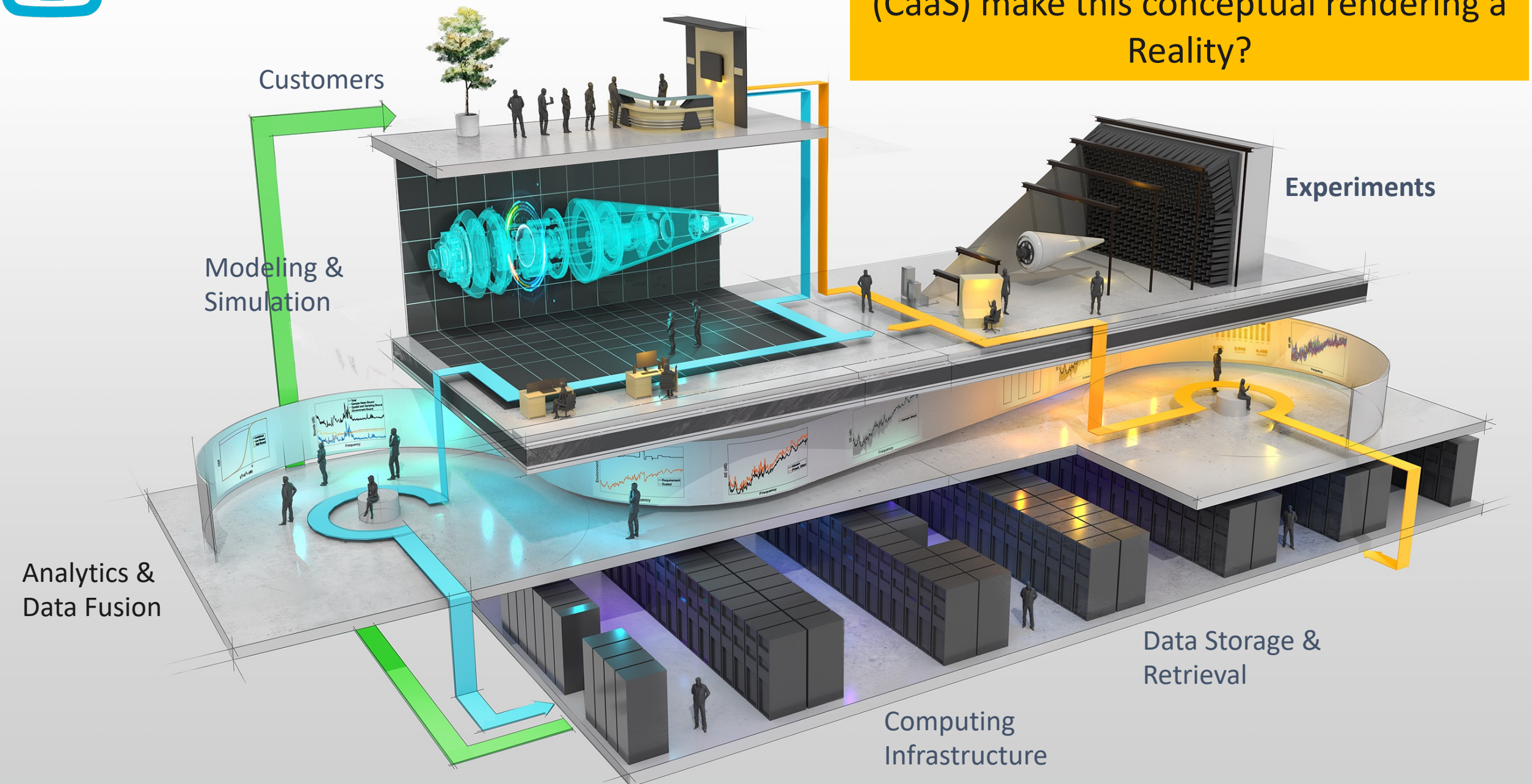
Engineers / designers navigate web browser to portal

Presented with ***menu of simulations***, customize as needed

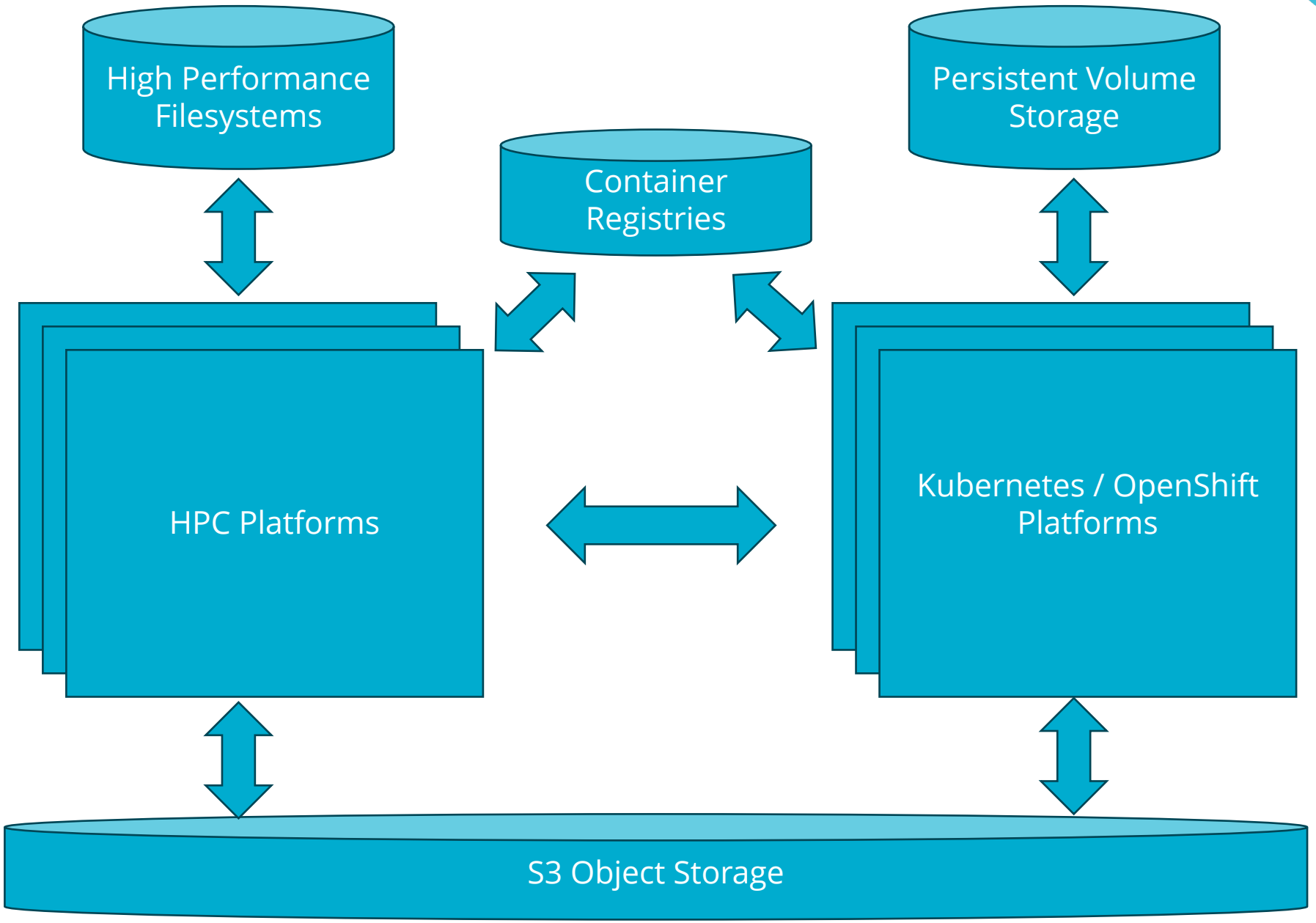
Results presented ***interactively*** and stored for later retrieval & analysis

Goal: Make the computing invisible

How can “Computing-as-a-Service” (CaaS) make this conceptual rendering a Reality?



CONVERGED HPC AND CLOUD COMPUTING ENVIRONMENT



Backend HPC Compute Jobs

UQ Studies

Unique HW Capabilities

Persistent Services

Web Frontends

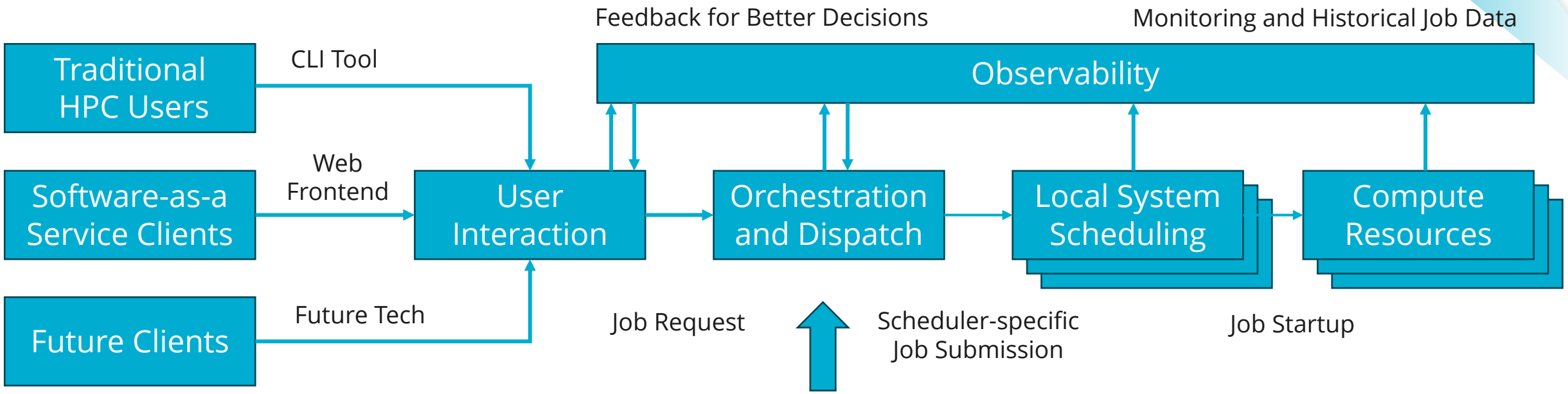
AI Inferencing

...

OUTLINE

- Introduction – my take on containers in HPC
- Computing Platform Engineering @ Sandia
  - Kubernetes / OpenShift deployments
  - Software-as-a-Service
  - Problems we have – new workload types and integration across systems
- K-Foundry Overview
  - Goals, architecture, status
- Conclusion

Computing-as-a-Service Architecture

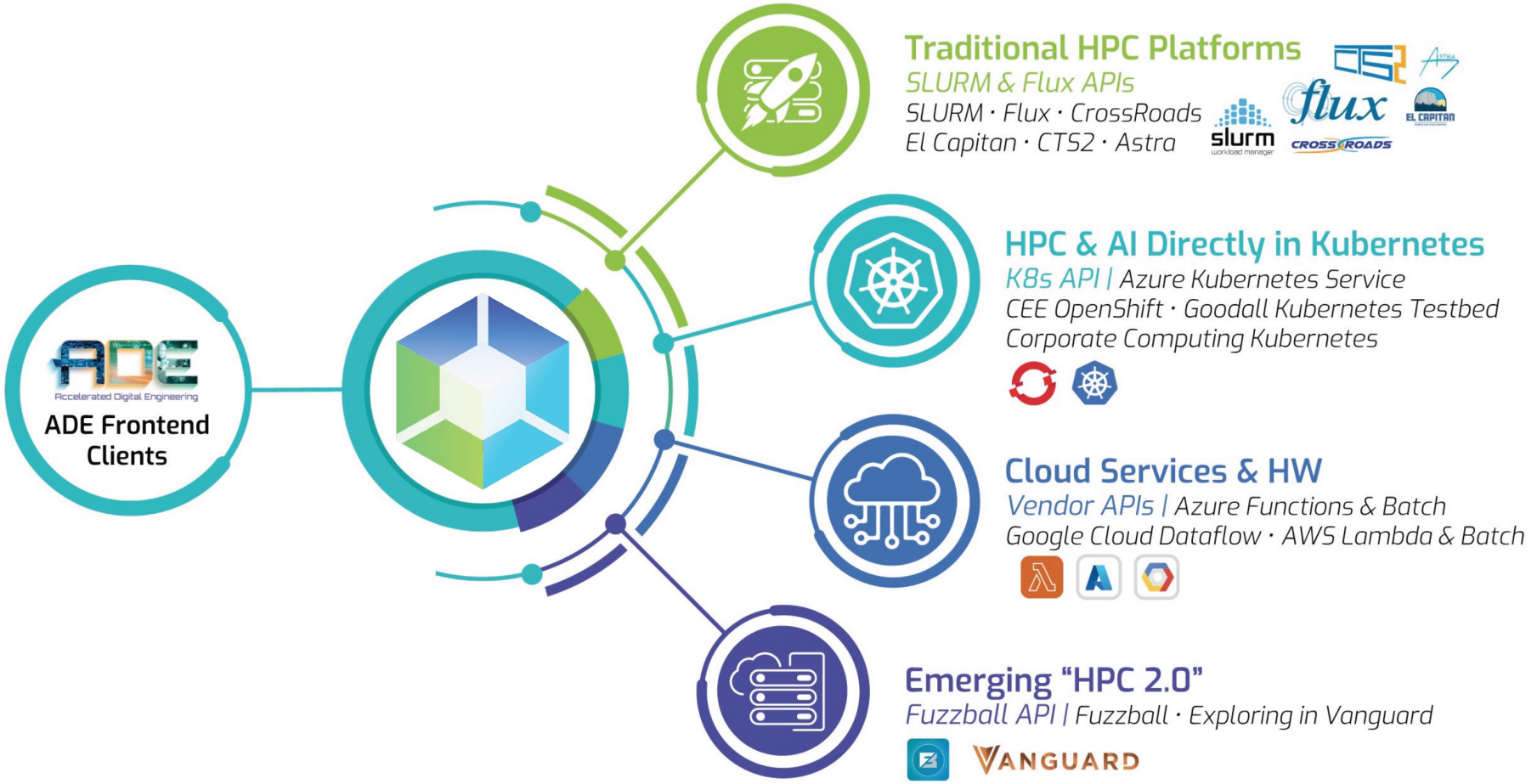


K-Foundry Focusing on Job Orchestration and Dispatch Layer

A NEW WAY TO INTERACT WITH DIVERSE SET OF COMPUTE

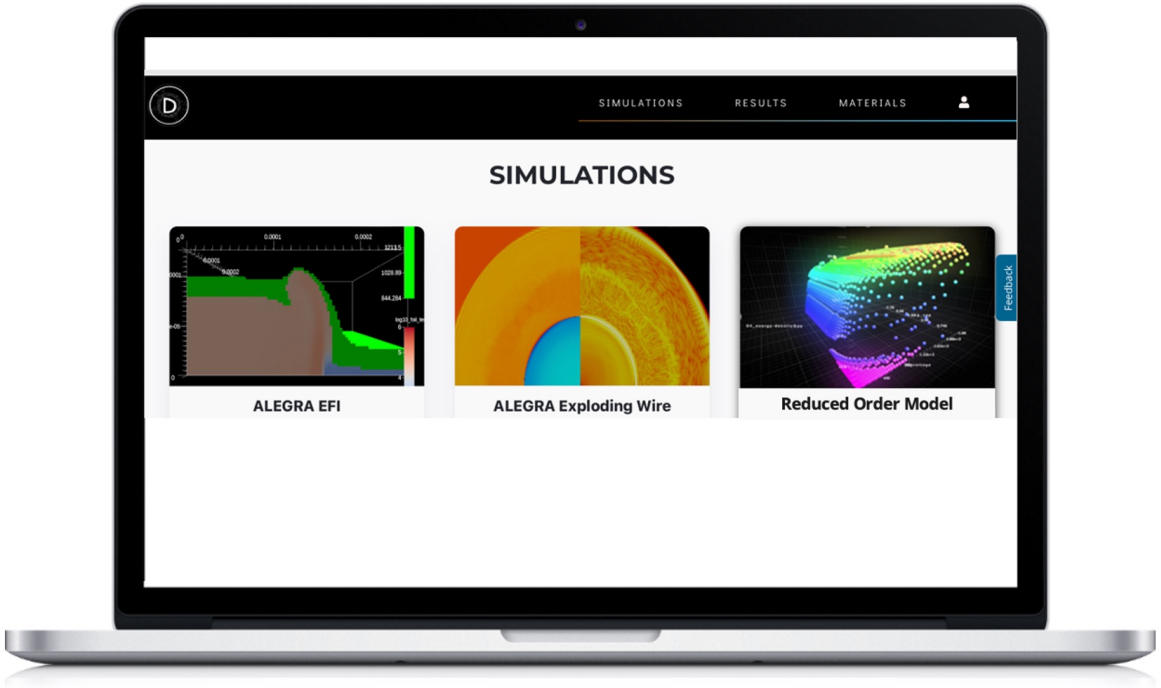
A routing layer for execution of interactive short-lived and long running jobs.

- **Evolving Expectations**
- **Complex Workloads**
- **Collaborative Growth**



INTRODUCTION TO K-FOUNDRY

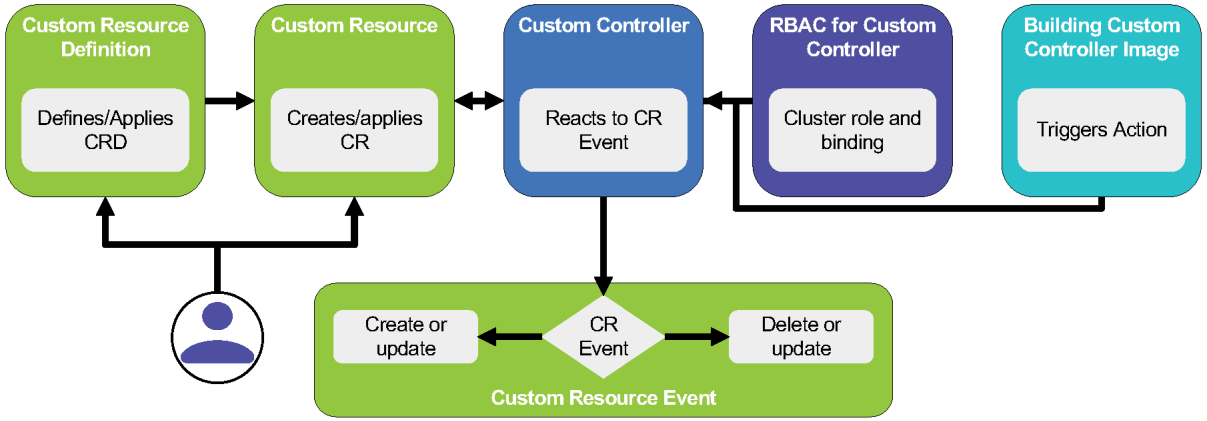
- **Overview:** A framework integrating SLURM in an HPC system with Kubernetes (K8s) to enable a unified communication and scheduling layer. **The framework is geared towards unifying communication for a diverse set of computing platforms and resource managers (i.e., Flux resource manager, Fuzzball workflows, different K8s clusters).**
- **Converged Execution:** Leverages K8s-like control planes to seamlessly schedule, manage, and monitor HPC and Microservices workloads through a unified configuration.
- **Goal/Objective:**
  - To improve infrastructure interaction to meet computation demands in containerized HPC workloads.
  - To improve user interaction with diverse computing platform and to abstract infrastructure resources from the end user.
  - To abstract compute resources and underlying infrastructure from users.



WHAT IS A CRD IN KUBERNETES?

CRDs provide a powerful way to extend Kubernetes.

- A **custom resource definition (CRD)** enables users to define custom object types and specify important attributes (e.g., objects name and scope).
- A ***custom resource (CR)*** *extends the Kubernetes API with user-defined API objects.*
- The ***Kubernetes API*** server leverages CRDs to create ***REST endpoints*** for managing a custom object with CRUD operations.



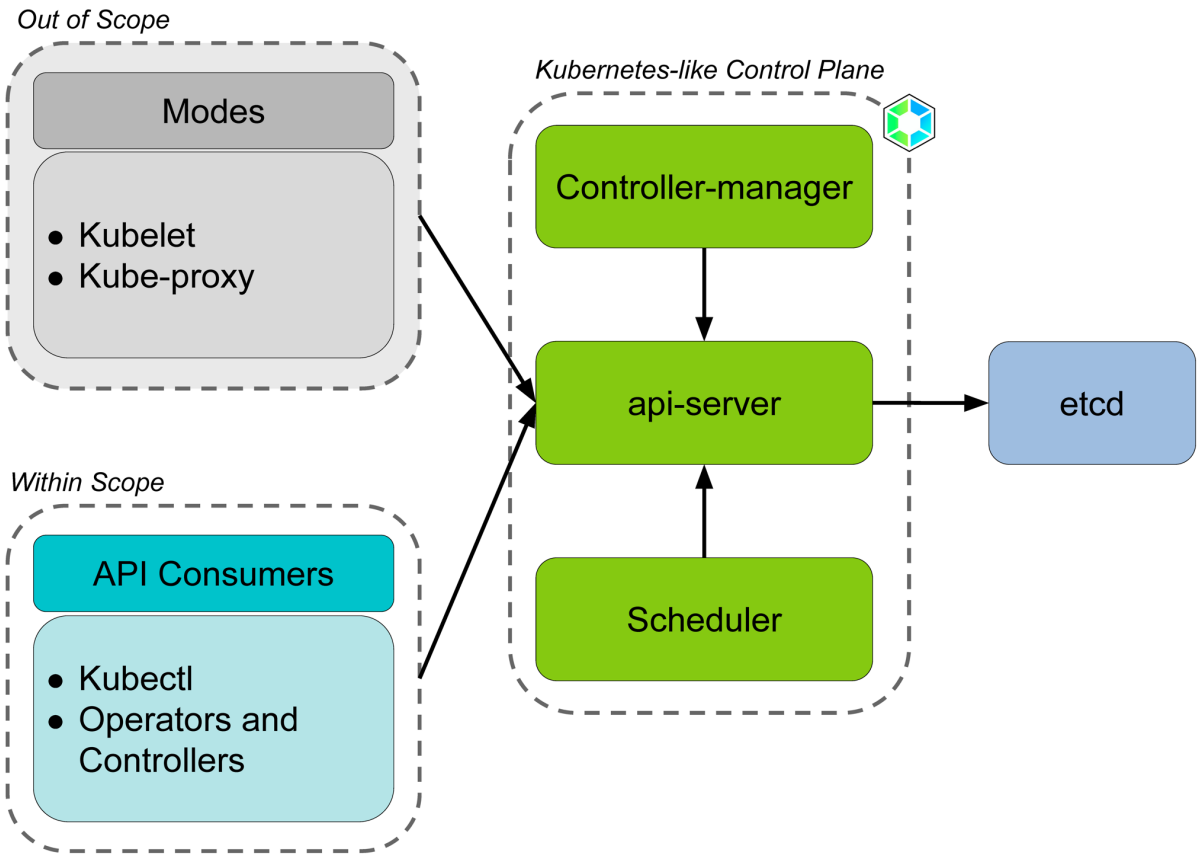
[1] Custom Resources, "Custom Resources," *Kubernetes*, Jul. 26, 2024. https://kubernetes.io/docs/concepts/extend-kubernetes/api-extension/custom-resources/?_gl=1 (accessed Nov. 02, 2024).

WHAT IS KUBERNETES-LIKE CONTROL PLANE (KCP)?

A high-level representation of KCP and its components

KCP is a lightweight, Kubernetes-like control plane.

- A control plane for many independent, isolated "clusters" known as workspaces.
- A **single binary** control plane that centralizes the control plane's functionalities, reducing complexity and overhead.
- It offers an API that is compatible with the Kubernetes API, allowing users to execute ***kubectl*** commands against it.
- Easy API consumption for users in their workspaces.



KCP'S SCOPE OF SUPPORT AND LIMITATIONS

Supported

- **Core Kubernetes Resources**: Supports ConfigMap, Secret, and RBAC types.
- ***Customizability**: Allows Custom Resource Definitions (CRDs) for extending capabilities.*
- **Advanced Solutions**: Offers built-in types for multi-tenancy and multi-cluster management.

Limitations

- No orchestration types (i.e., Node, Pod, Deployment, etc.) come with the vanilla KPC.

MINIMIZING THE ADOPTION BURDEN

A uniform API by modeling containerized workloads after K8s pods

- We can leverage existing standardized job fields.
- Batch jobs become K8s jobs.

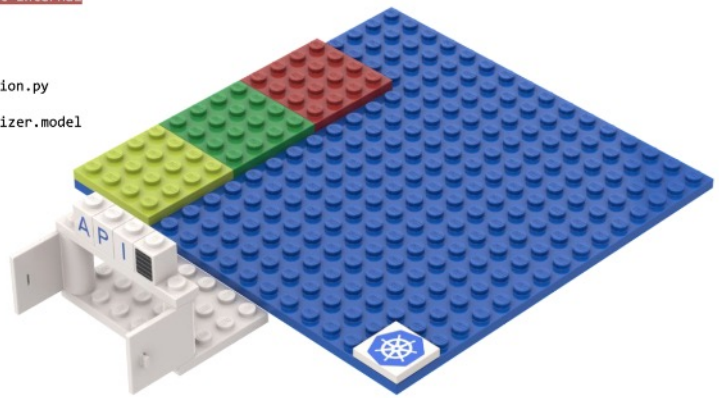
```yaml
apiVersion: kfoundry.io/v1alpha1
kind: Job
metadata:
  generateName: k-foundry-example
  namespace: default
spec:
  template:
    spec:
      containers:
      - name: pi
        image: perl:5.34.0
        command:["perl",  "-Mbignum=bpi", "-wle", "print bpi(2000)"]
        resources:
          limits:
            cpu: "4"
            memory: "16Gi"
            nvidia.com/gpu: "1"
          requests:
            cpu: "4"
            memory: "16Gi"
            nvidia.com/gpu: "1"
      restartPolicy: Never

status: {}
```

```yaml
kind: Model
apiVersion: training.faros.sh/v1alpha1
metadata:
  name: my-amazing-chat
  namespace: team1
spec:
  model: Llama2
  nProcPerNod: 1
  script: chat_completion.py
  ckptDir: /pvc/data
  tokenizerPath: tokenizer.model
  maxSeqLen: 512
  maxBatchSize: 6
```

```yaml
kind: Model
apiVersion: training.faros.sh/v1alpha1
metadata:
  name: my-amazing-chat-internal
  namespace: team3
spec:
  model: Llama2
  nProcPerNod: 2
  script: chat_completion.py
  ckptDir: /pvc/data
  tokenizerPath: tokenizer.model
  maxSeqLen: 512
  maxBatchSize: 3
```

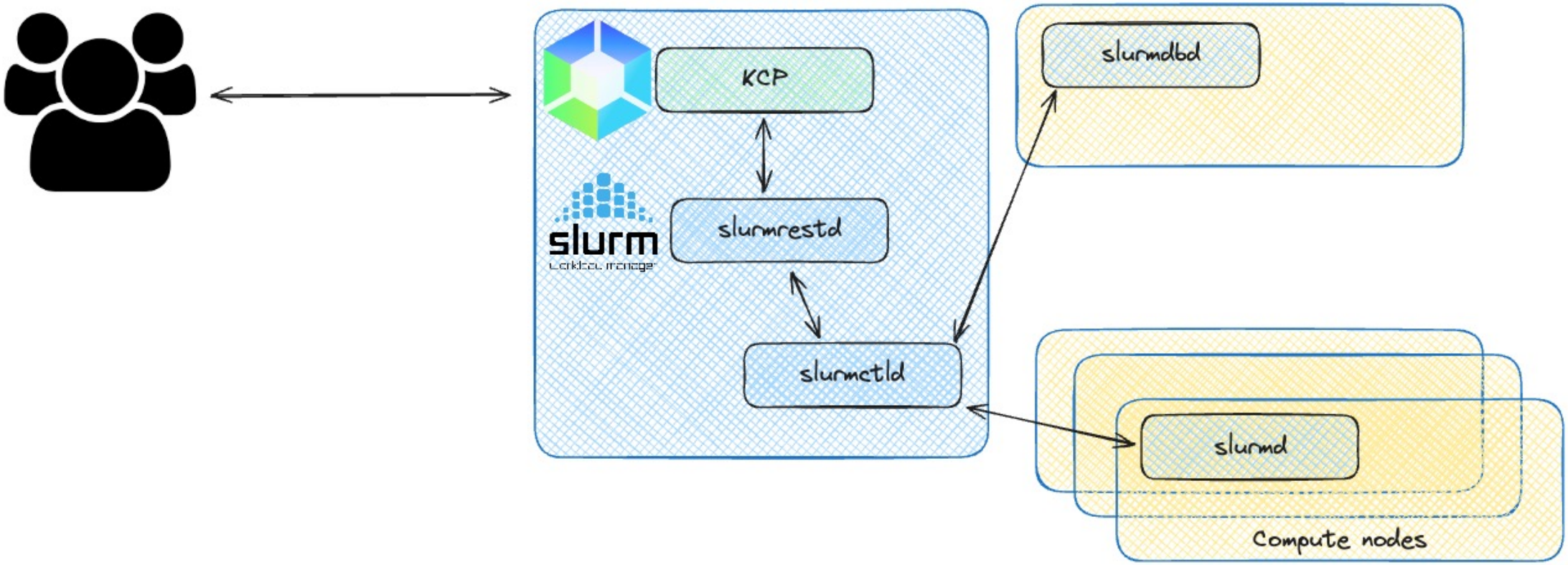
```yaml
kind: Model
apiVersion: training.faros.sh/v1alpha1
metadata:
  name: my-amazing-chat
  namespace: team2
spec:
  model: Llama2
  nProcPerNod: 2
  script: chat_completion.py
  ckptDir: /pvc/data
  tokenizerPath: tokenizer.model
  maxSeqLen: 512
  maxBatchSize: 6
```

with a **uniform API**.

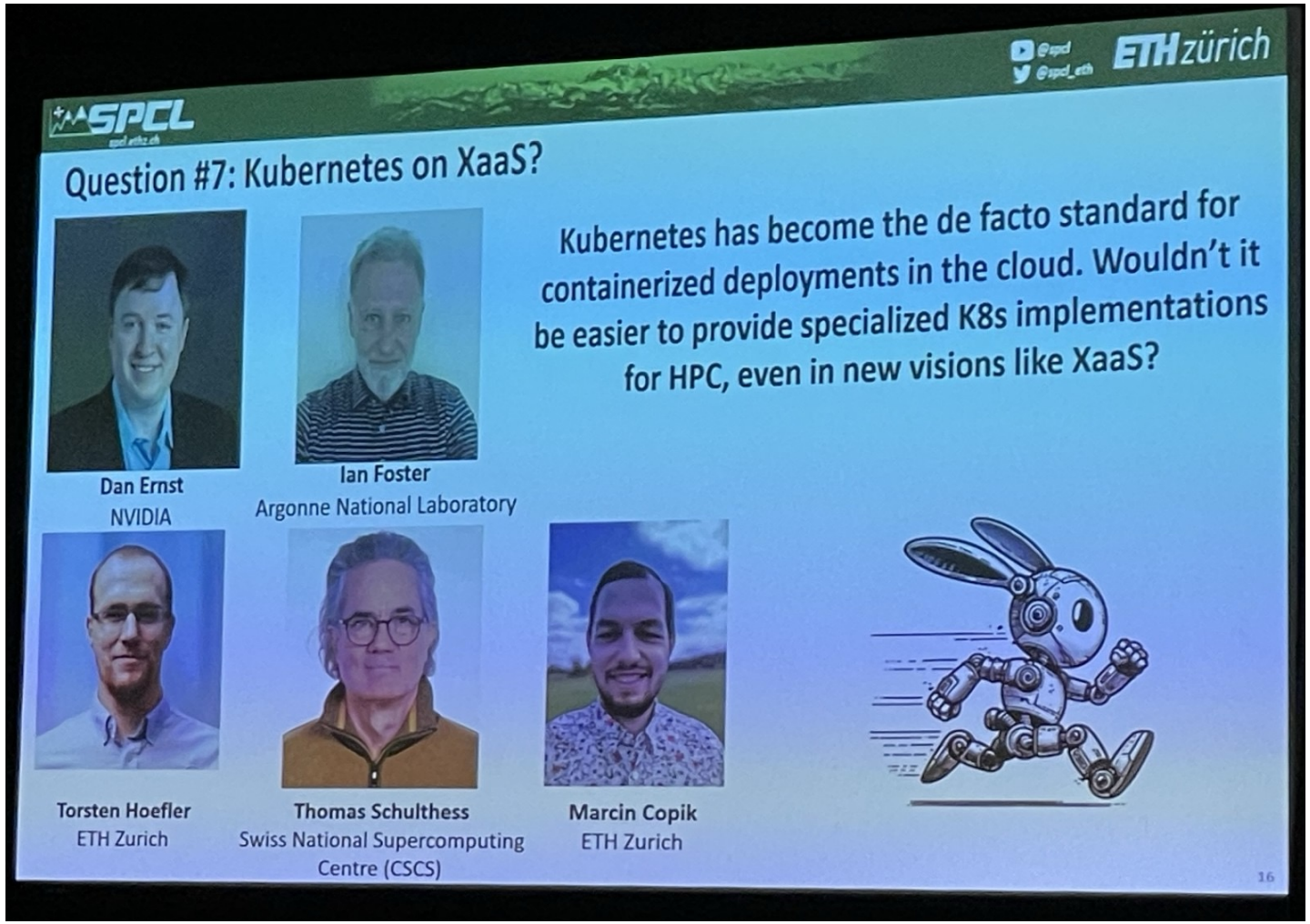


[1] kcp-dev, “contrib/20240321-kubecon-paris/slides.pdf at main · kcp-dev/contrib,” *GitHub*, 2024. https://github.com/kcp-dev/contrib/blob/main/20240321-kubecon-paris/slides.pdf (accessed Nov. 04, 2024).

USE CASE: SLURM CLOUD NATIVE WAY



K-FOUNDRY COULD BE THOUGHT OF AS A SPECIALIZED K8S API



K-Foundry isn't Kubernetes.

K-Foundry provides a K8s-like API for submitting containerized jobs to backend compute.

Supports interactive and rapid turn-around use cases, QoS requirements.

Federates together multiple computing platforms (HPC systems, K8s, standalone servers, specialized hardware).

OUTLINE

- Introduction – my take on containers in HPC
- Computing Platform Engineering @ Sandia
  - Kubernetes / OpenShift deployments
  - Software-as-a-Service
  - Problems we have – new workload types and integration across systems
- K-Foundry Overview
  - Goals, architecture, status
- Conclusion

CONCLUSION

- Many factors driving container adoption in HPC; containers in HPC still hard

- Computing Platform Engineering initiative @ Sandia is driving:
  - Modernized computing env integrating Cloud and HPC technologies
  - Streamlined Software-as-a-Service delivery of advanced modeling and simulation codes

- K-Foundry Framework
  - Execution engine for containerized jobs, provides Kubernetes-like API and CRD
  - Federates together multiple computing resources, intelligently routes jobs
  - Prototype targets SLURM and Kubernetes, in future Flux and Fuzzball

- This is a work in progress, interested in hearing your ideas and feedback