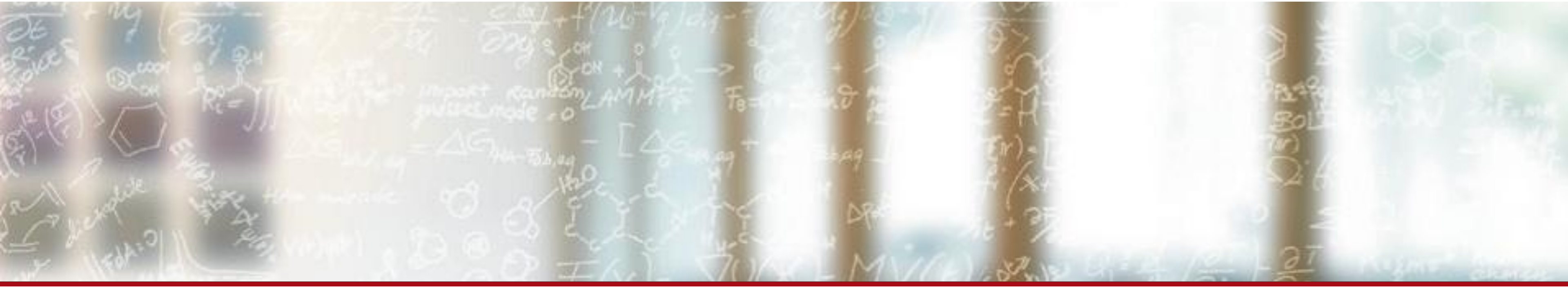
CSCS
Centro Svizzero di Calcolo Scientifico
Swiss National Supercomputing Centre

ETH zürich



Developing Cloud-native HPC Clusters at CSCS

Felipe Cruz
cruz@cscs.ch

Context

- Work on the successor to “Piz Daint”
- HPE Cray EX system
  - Software-defined infrastructure via API
- Single versatile infrastructure
  - **One system for different needs**
- Flexible
  - **Versatile** solutions will be defined in software, no longer via hardware
- **Operate, manage, and maintain** versatile clusters for customers?

Cloud Native Definition

“Cloud native technologies empower organizations to **build and run scalable applications** in modern, dynamic environments such as public, private, and hybrid clouds. Containers, service meshes, microservices, immutable infrastructure, and declarative APIs exemplify this approach.

These techniques enable **loosely coupled systems that are resilient, manageable, and observable**. Combined with robust automation, they allow engineers to make **high-impact changes frequently and predictably with minimal toil**.”

–

Who We Are. https://www.cncf.io/about/who-we-are/

Cloud Native Computing Foundation (CNCF)

How to build versatile clusters for customers?

- Leverage the cloud native pillars
  - **Containers**
  - **Microservices**
  - **CICD**
  - **DevOps**

Ava Babili
https://commons.wikimedia.org/wiki/File:Temple_of_Olympian_Zeus_Athens_Greece_9.jpg

How to build versatile clusters for customers?

- Leverage the cloud native pillars
  - Containers
  - Microservices
  - CICD
  - DevOps
- … adapted to HPC!
  - **Cloud-Native HPC Cluster**

DALLE-2 (https://labs.openai.com)
"Supercomputer Native to the Clouds"

Cloud-native HPC Cluster Overview

Cloud Native Definition

“Cloud native technologies empower organizations to **build and run scalable applications** in modern, dynamic environments such as public, private, and hybrid clouds. Containers, service meshes, microservices, immutable infrastructure, and declarative APIs exemplify this approach.

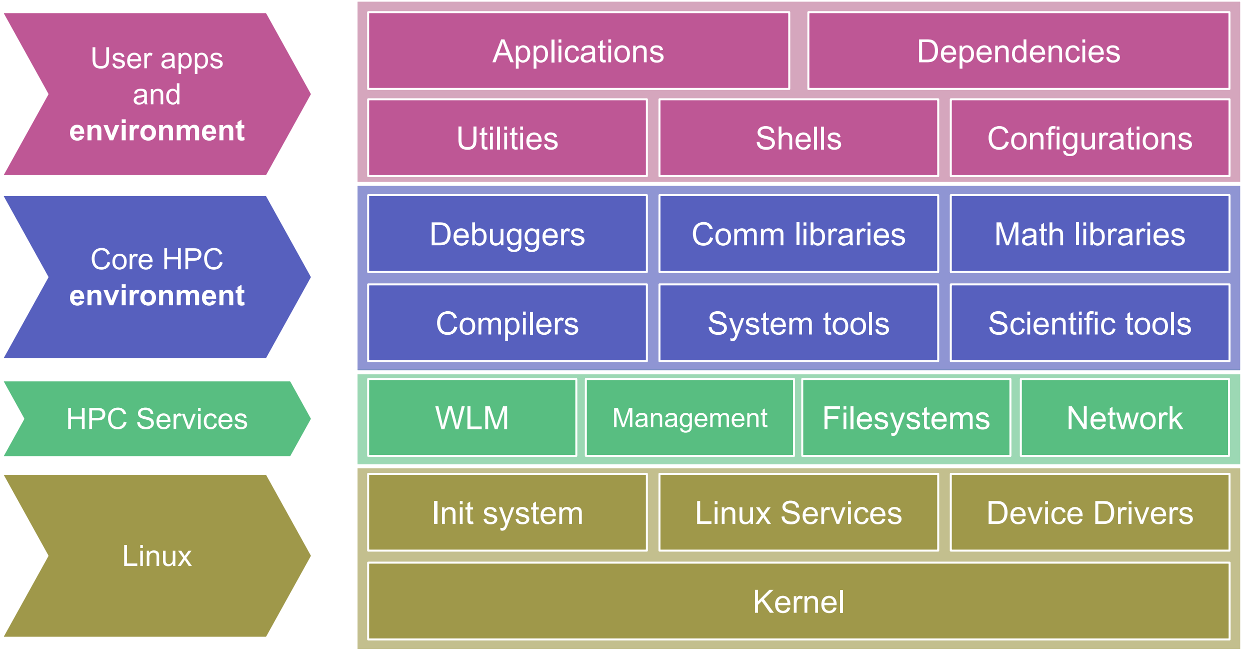
These techniques enable **loosely coupled systems** that are resilient, manageable, and observable. Combined with robust **automation**, they allow engineers to make **high-impact changes frequently and predictably with minimal toil**.”

–

Who We Are. https://www.cncf.io/about/who-we-are/

Cloud Native Computing Foundation (CNCF)

HPC Software Stack: **Traditional view of full stack**



User apps and **environment**

Applications | Dependencies

Utilities | Shells | Configurations

Core HPC **environment**

Debuggers | Comm libraries | Math libraries

Compilers | System tools | Scientific tools

HPC Services

WLM | Management | Filesystems | Network

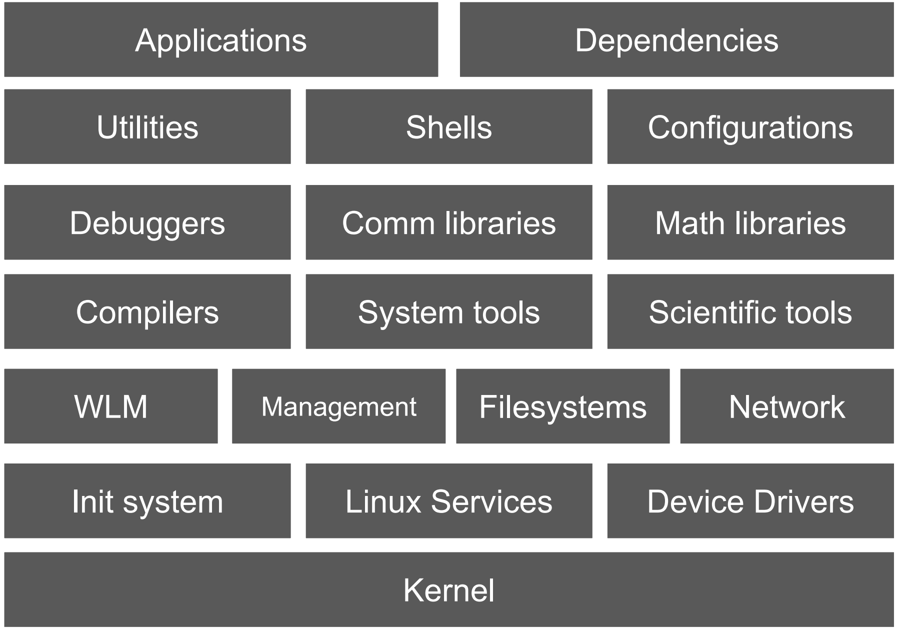
Linux

Init system | Linux Services | Device Drivers

Kernel

- Monolithic multi-layered stack, compromise of stability and flexibility
- Solution does not scale well with number of needs

HPC Software Stack: **reimagine the “stack”**



Applications | Dependencies

Utilities | Shells | Configurations

Debuggers | Comm libraries | Math libraries

Compilers | System tools | Scientific tools

WLM | Management | Filesystems | Network

Init system | Linux Services | Device Drivers

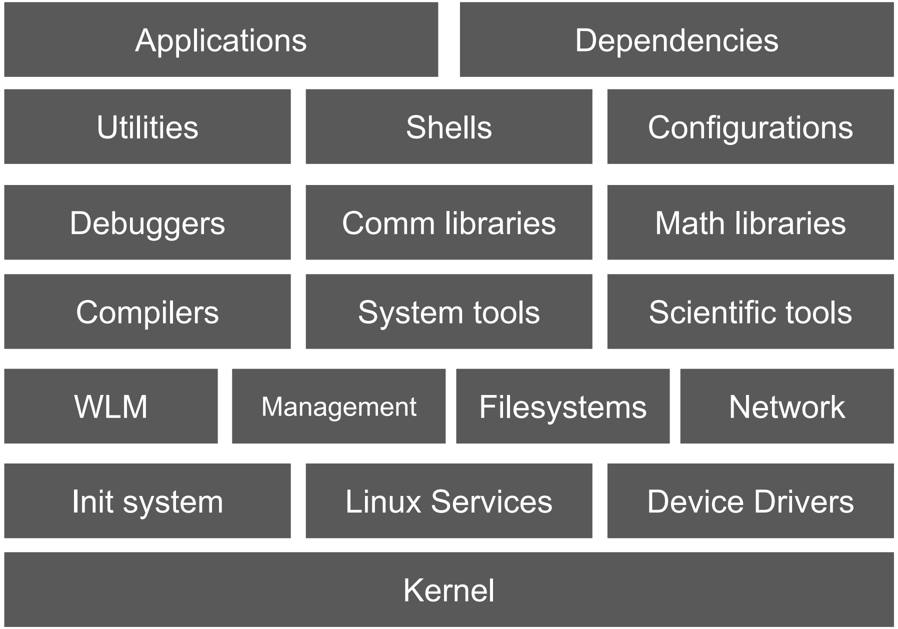
Kernel

- Decoupling the stack
- Frequent component deployment
  - with minimal effort to maintain

HPC Software Stack: **reimagine the “stack”**

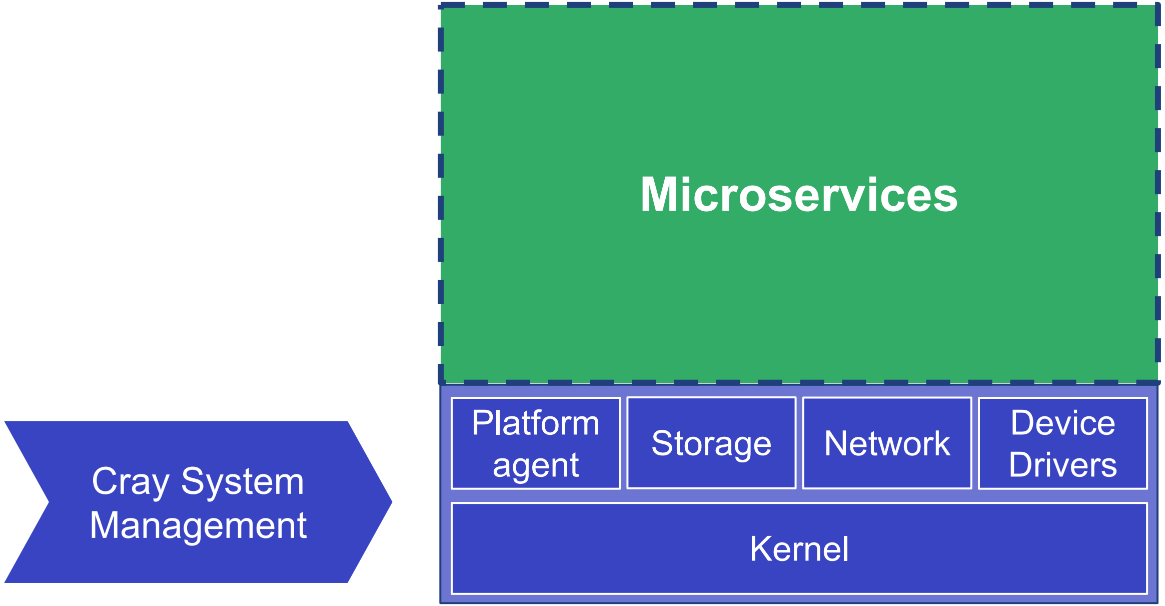
- Cloud native pillars
  - **Containers**
  - **Microservices**
  - **CI/CD**
  - **DevOps**

Ava Babili
https://commons.wikimedia.org/wiki/File:Temple_of_Olympian_Zeus_Athens_Greece_9.jpg

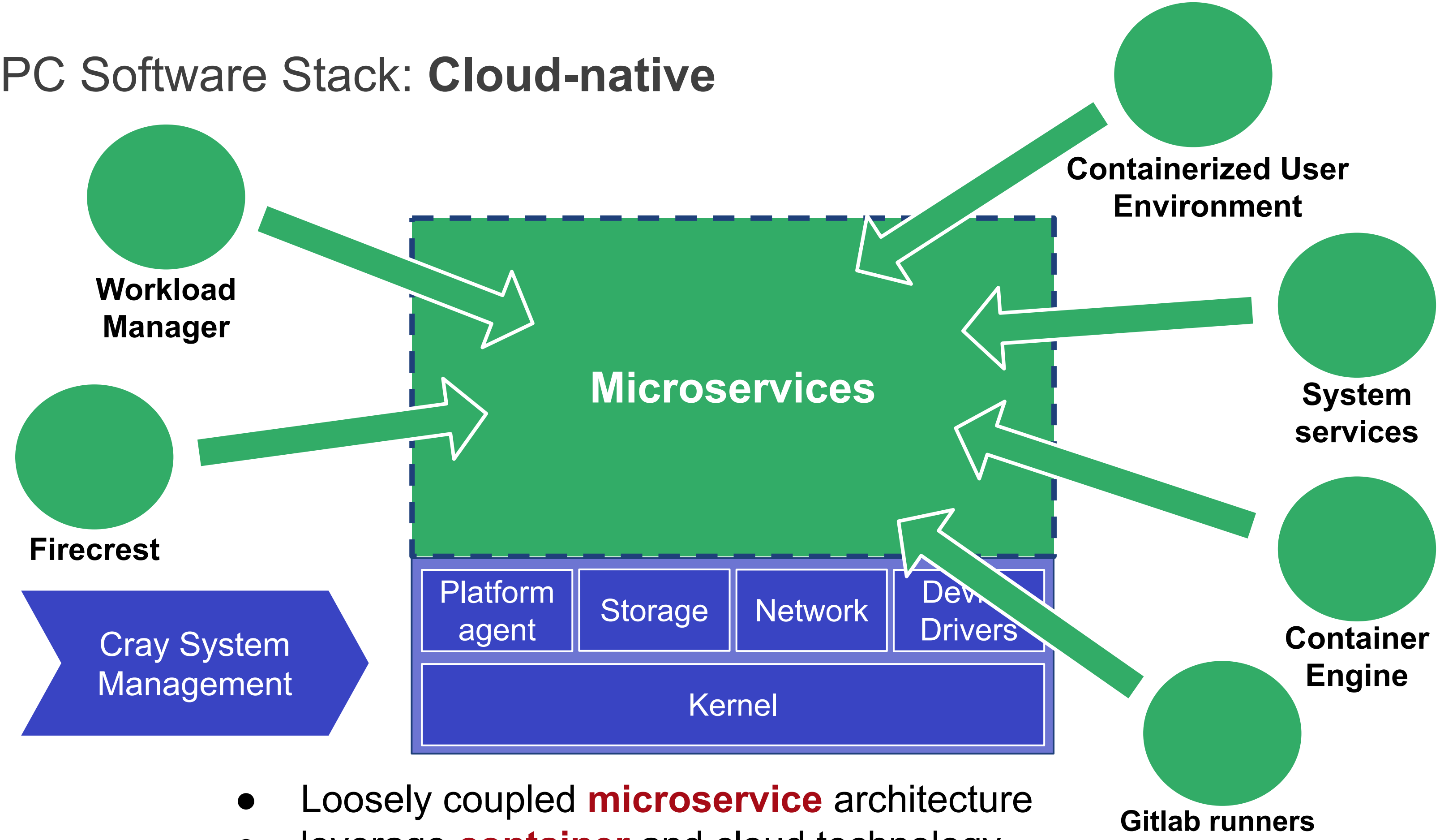


Applications		Dependencies	
Utilities	Shells	Configurations	
Debuggers	Comm libraries	Math libraries	
Compilers	System tools	Scientific tools	
WLM	Management	Filesystems	Network
Init system	Linux Services	Device Drivers	
Kernel			

HPC Software Stack: **Cloud-native**

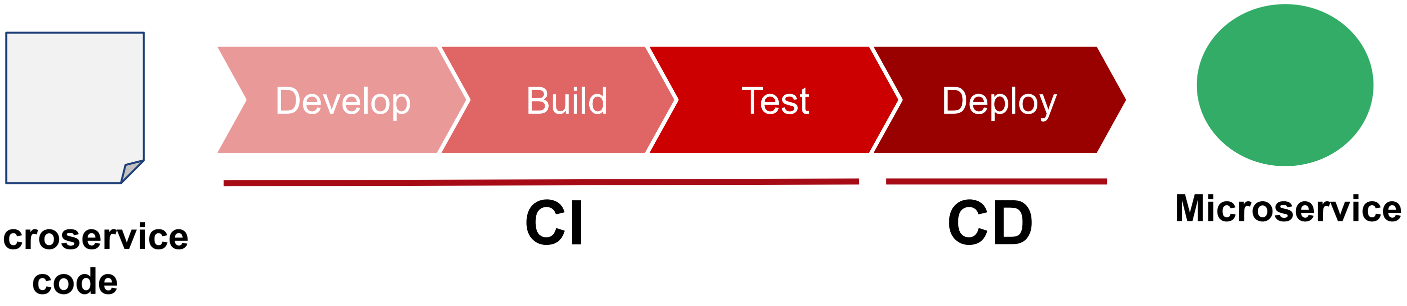


HPC Software Stack: **Cloud-native**



- Loosely coupled **microservice** architecture
- leverage **container** and cloud technology

HPC Software Stack: **Cloud-native microservices**



Microservice code

Develop

Build

Test

Deploy

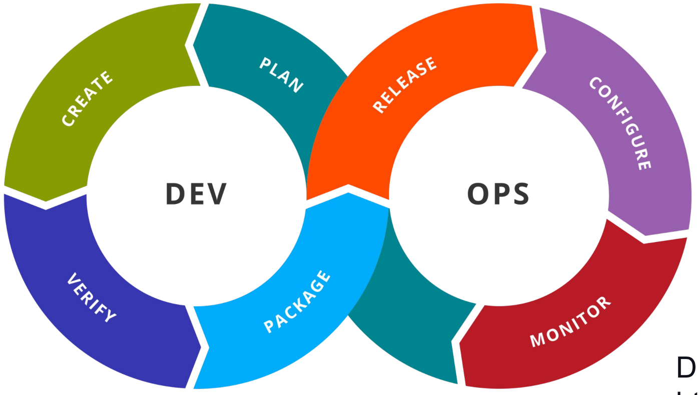
CI

CD

Microservice

CI/CD

- Continuous integration
- SW defined
- Automation
- Tools



DevOps

- Practices
- Processes
- Roles

DevOps toolchain. (2023, April 27). In *Wikipedia*. https://en.wikipedia.org/wiki/DevOps_toolchain

Cloud-Native HPC Cluster summary

- **Dynamic Infrastructure** with CSM
- **Loosely coupled** microservices architecture
- Leverage cloud **automation**

Towards HPC clusters

- Resilient
- Manageable
- Observable

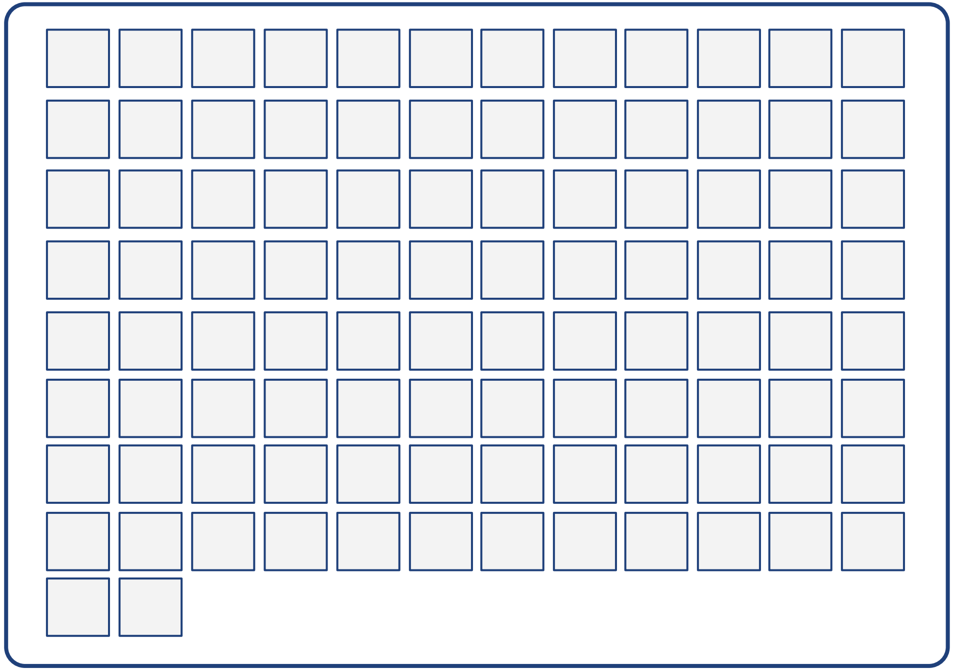
While enabling

- Teams of engineers
- Frequent changes
- Minimal toil

DALLE-2 (https://labs.openai.com)
“Supercomputer Native to the Clouds”

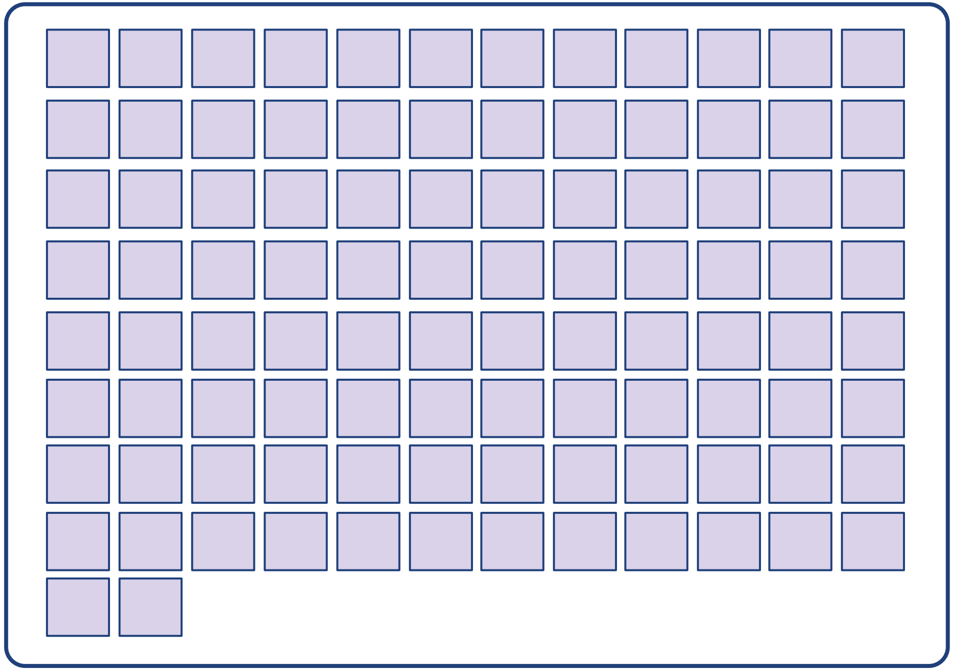
Anatomy of a Cloud-native HPC cluster (bottom-up)

Cloud-Native HPC: **Infrastructure**



- Dynamically provision nodes using HPE Cray Shasta
    - Provision nodes
    - Base config of resources
- Software-defined infrastructure

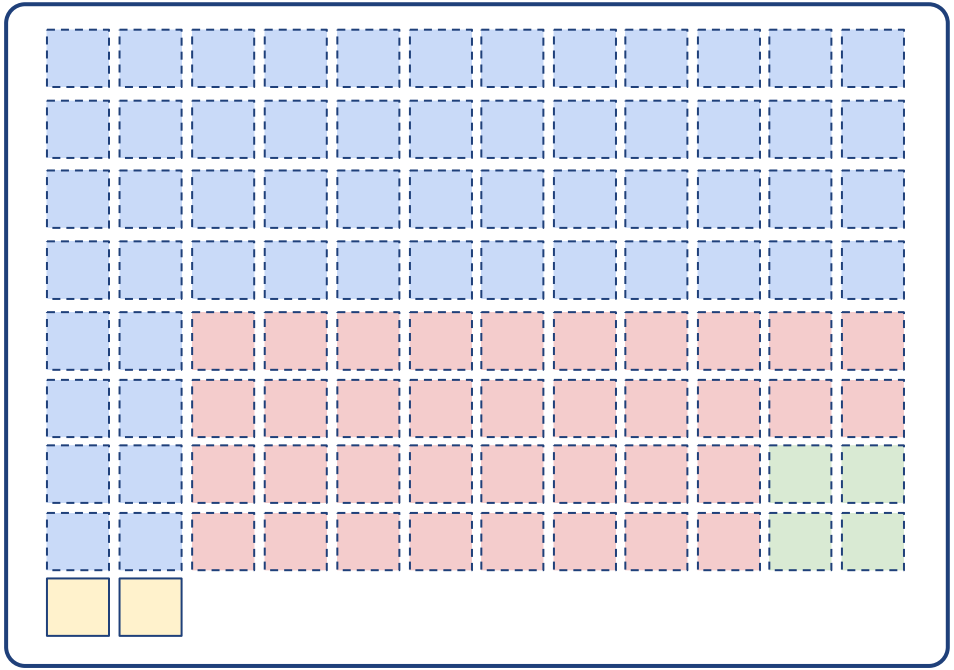
Cloud-Native HPC: **software-defined tenant**



Software defined via Cray CSM

- Base state
    - Minimal OS
    - Devices
    - Core FS
    - VLAN
    - Core services
        - Microservice Orchestrator
- Identical node state
    - **Interchangeable**
- We now have a **fleet of nodes**!

Cloud-Native HPC: **Automated service management**

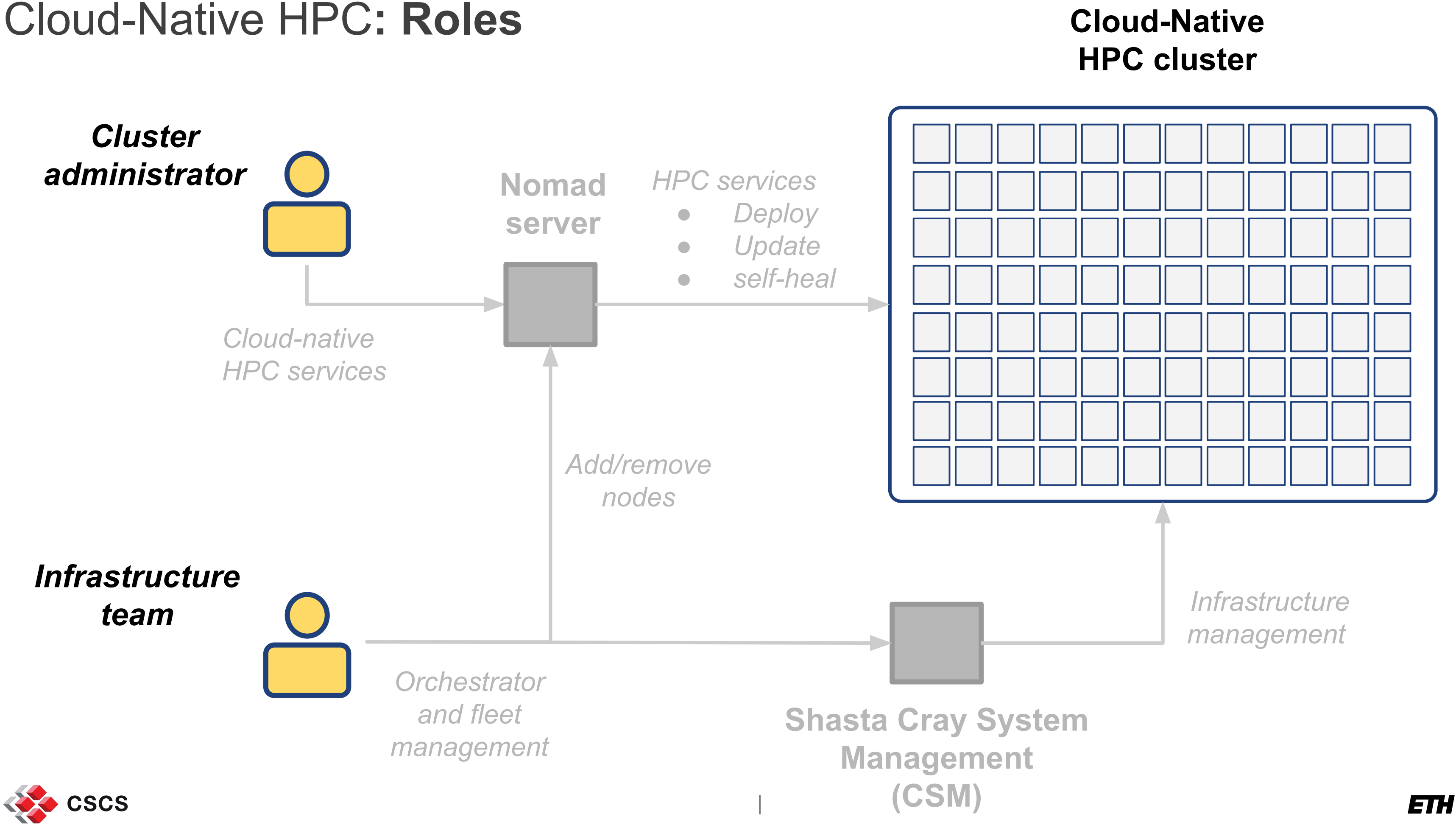


* figures not to scale

Service management

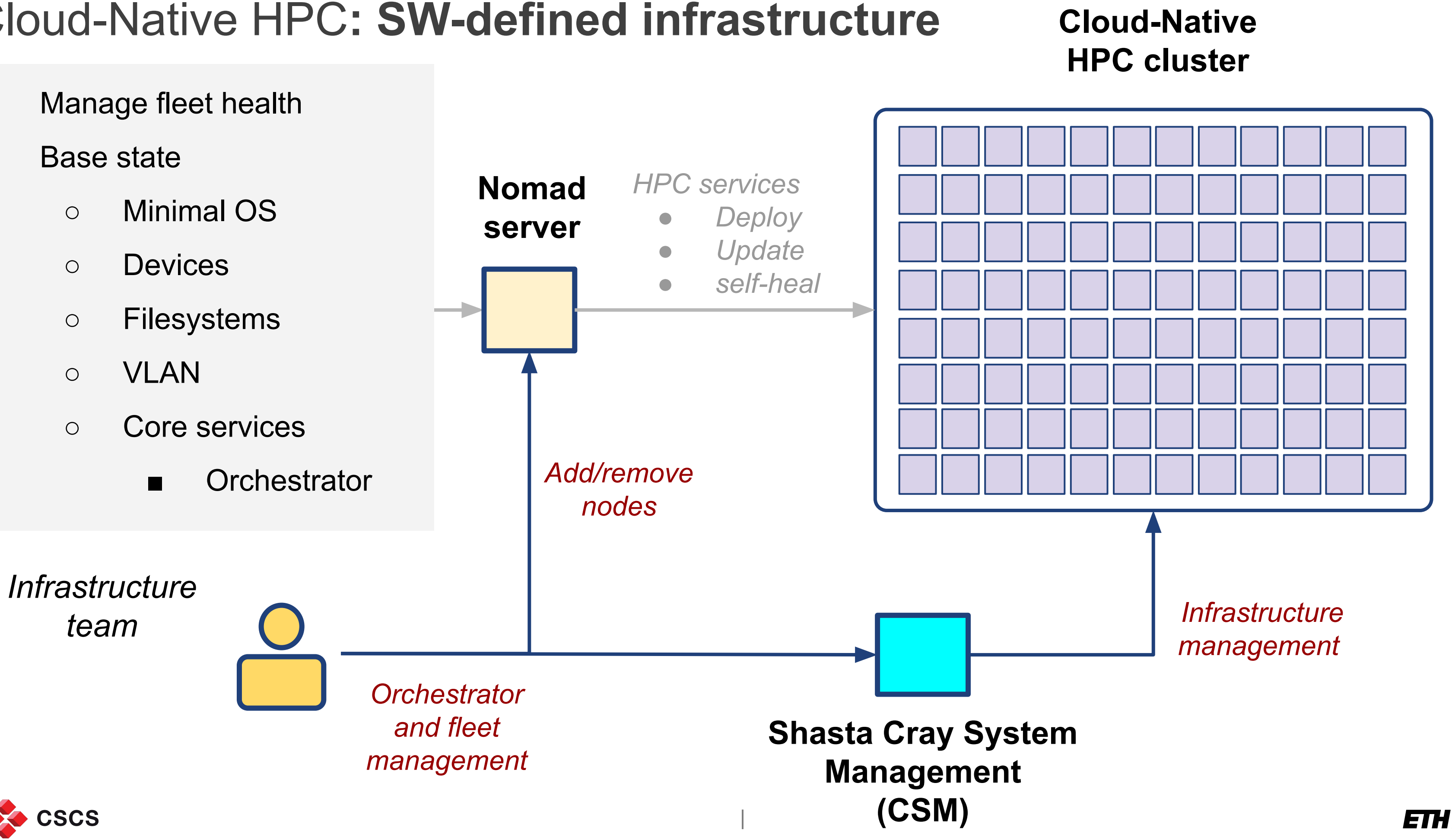
- 🟦 SLURM pool
- 🟩 Cluster services pool
    - Gitlab runner
    - HPC APIs (firecrest)
    - JupyterHub
- 🟥 HTC services pool
- 🟨 Nomad support pool

Cloud-Native HPC: **Roles**



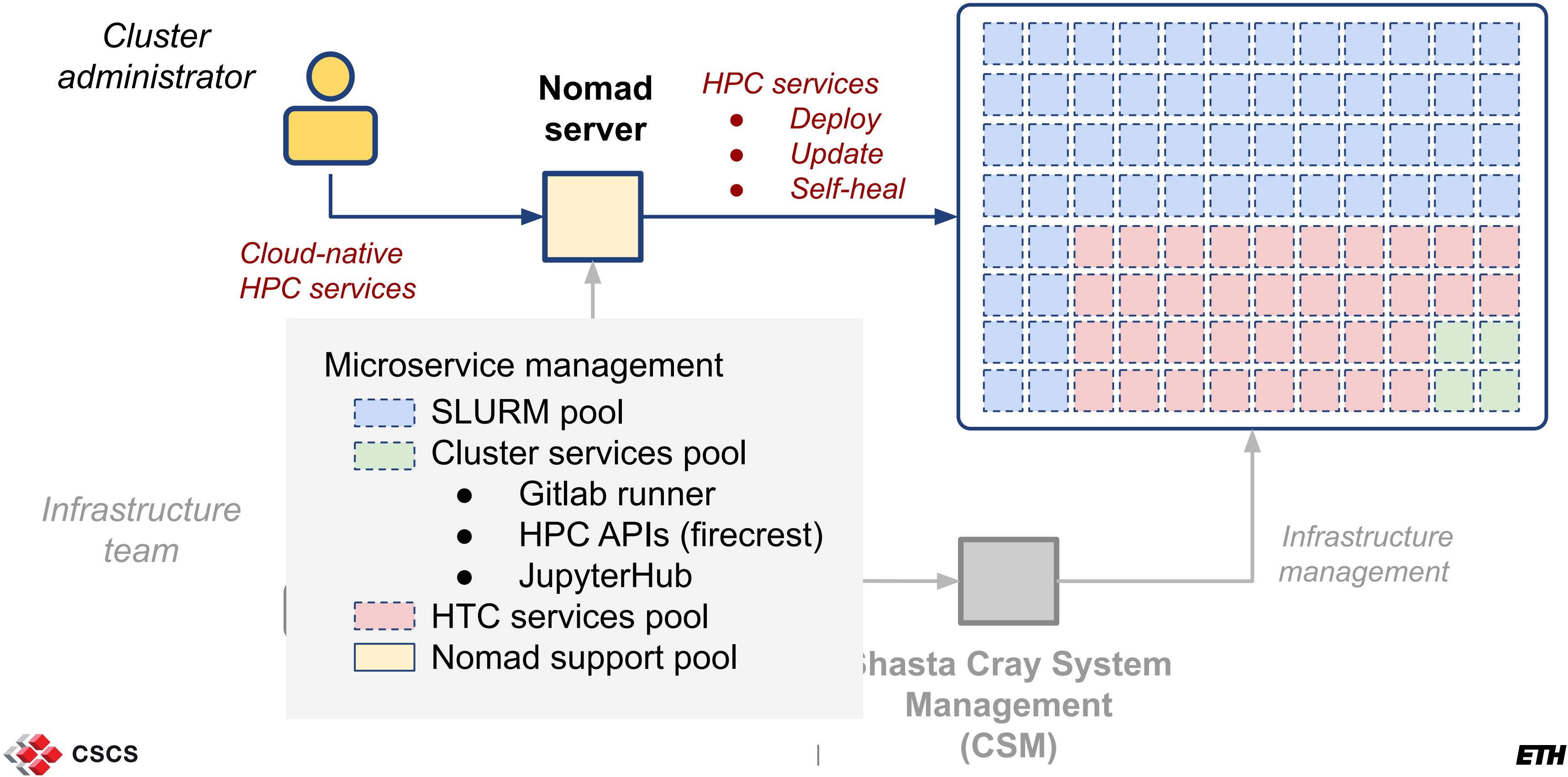
Cloud-Native HPC: **SW-defined infrastructure**

- Manage fleet health
- Base state
  - Minimal OS
  - Devices
  - Filesystems
  - VLAN
  - Core services
    - Orchestrator



Cloud-Native HPC: **SW-defined microservices**

Cloud-Native HPC cluster

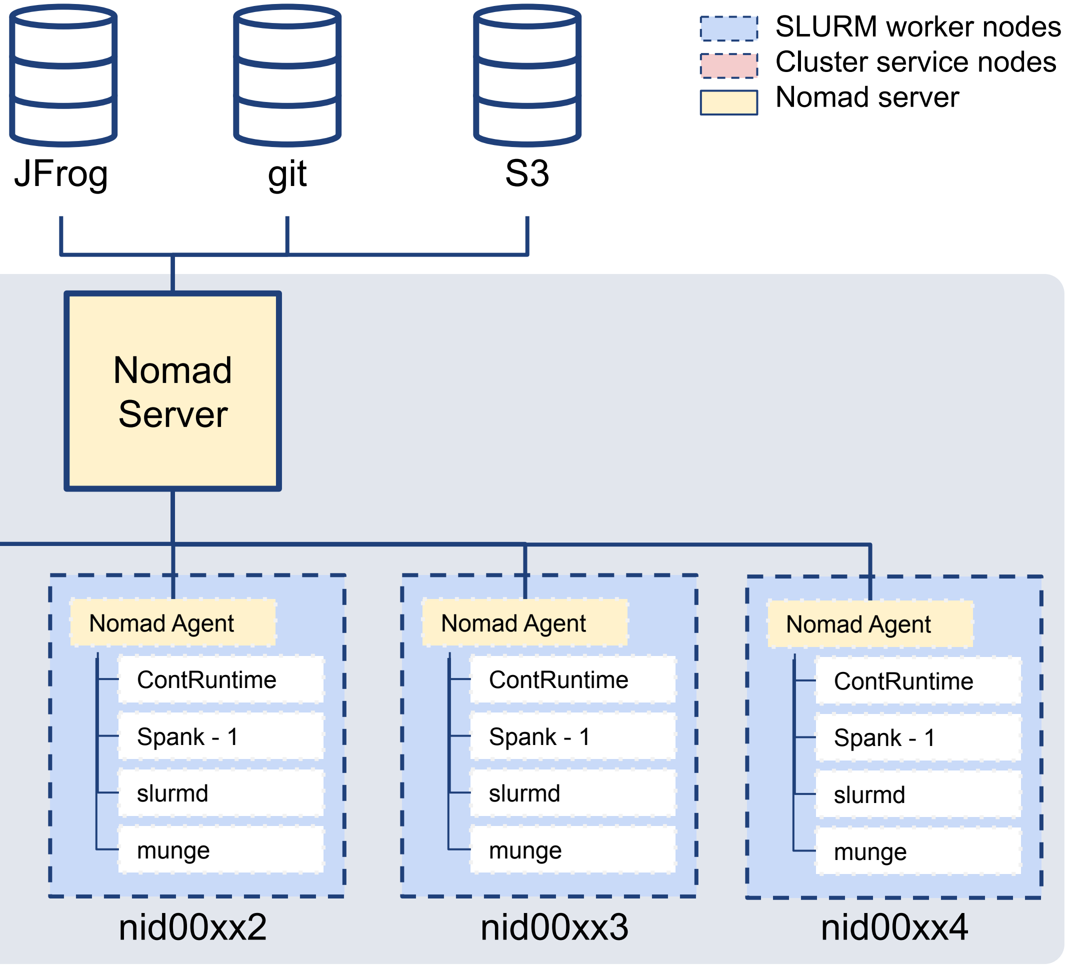


Microservice management

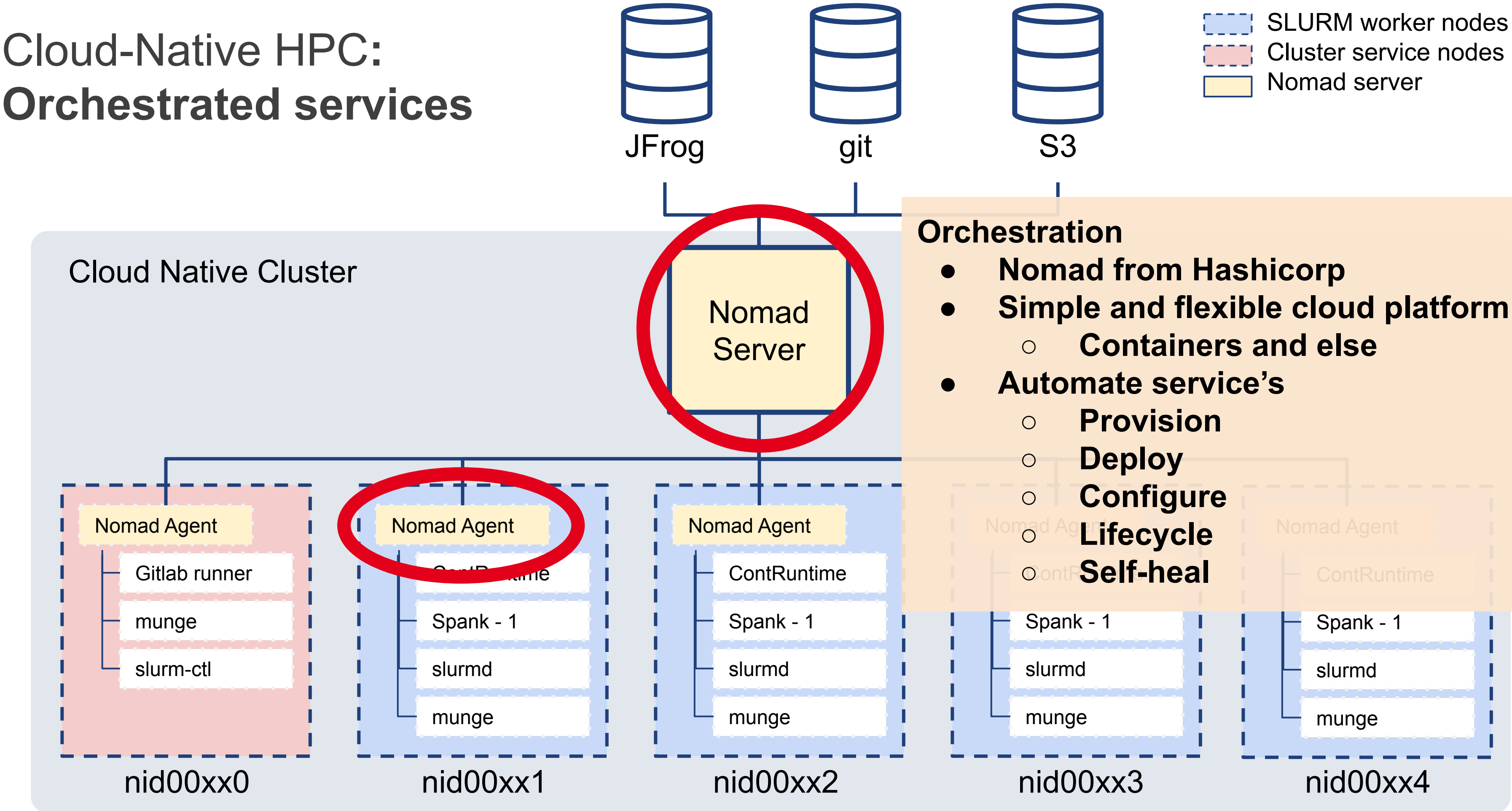
- 🟦 SLURM pool
- 🟩 Cluster services pool
  - Gitlab runner
  - HPC APIs (firecrest)
  - JupyterHub
- 🟥 HTC services pool
- 🟨 Nomad support pool

Orchestrated HPC services

Cloud-Native HPC:
Orchestrated services



Cloud-Native HPC:
Orchestrated services



JFrog

git

S3

▨ SLURM worker nodes
▨ Cluster service nodes
□ Nomad server

Cloud Native Cluster

Nomad Server

nid00xx0: Nomad Agent, Gitlab runner, munge, slurm-ctl

nid00xx1: Nomad Agent, ContRuntime, Spank - 1, slurmd, munge

nid00xx2: Nomad Agent, ContRuntime, Spank - 1, slurmd, munge

nid00xx3: Nomad Agent, ContRuntime, Spank - 1, slurmd, munge

nid00xx4: Nomad Agent, ContRuntime, Spank - 1, slurmd, munge

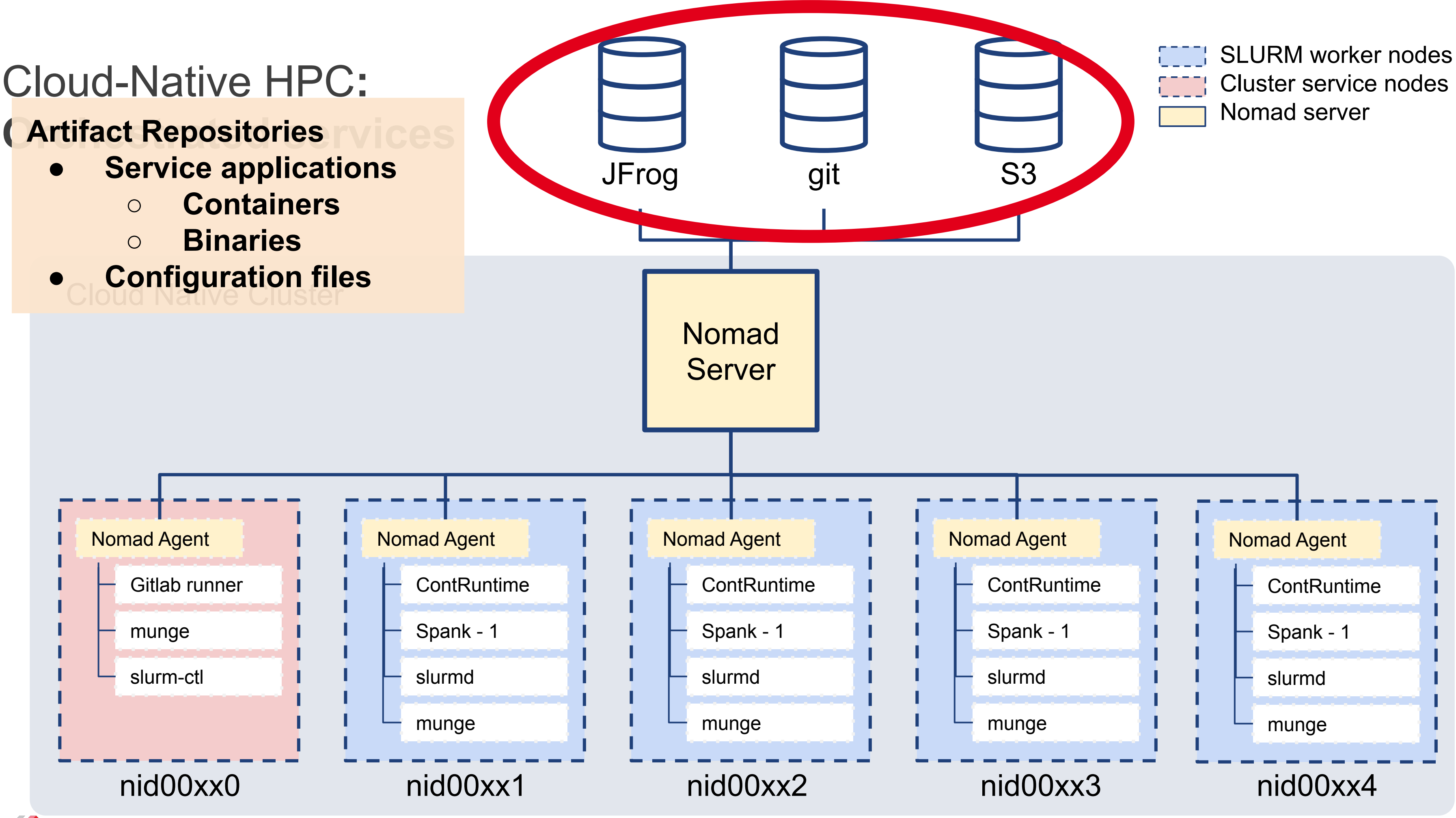
Orchestration

- **Nomad from Hashicorp**
- **Simple and flexible cloud platform**
  - **Containers and else**
- **Automate service's**
  - **Provision**
  - **Deploy**
  - **Configure**
  - **Lifecycle**
  - **Self-heal**

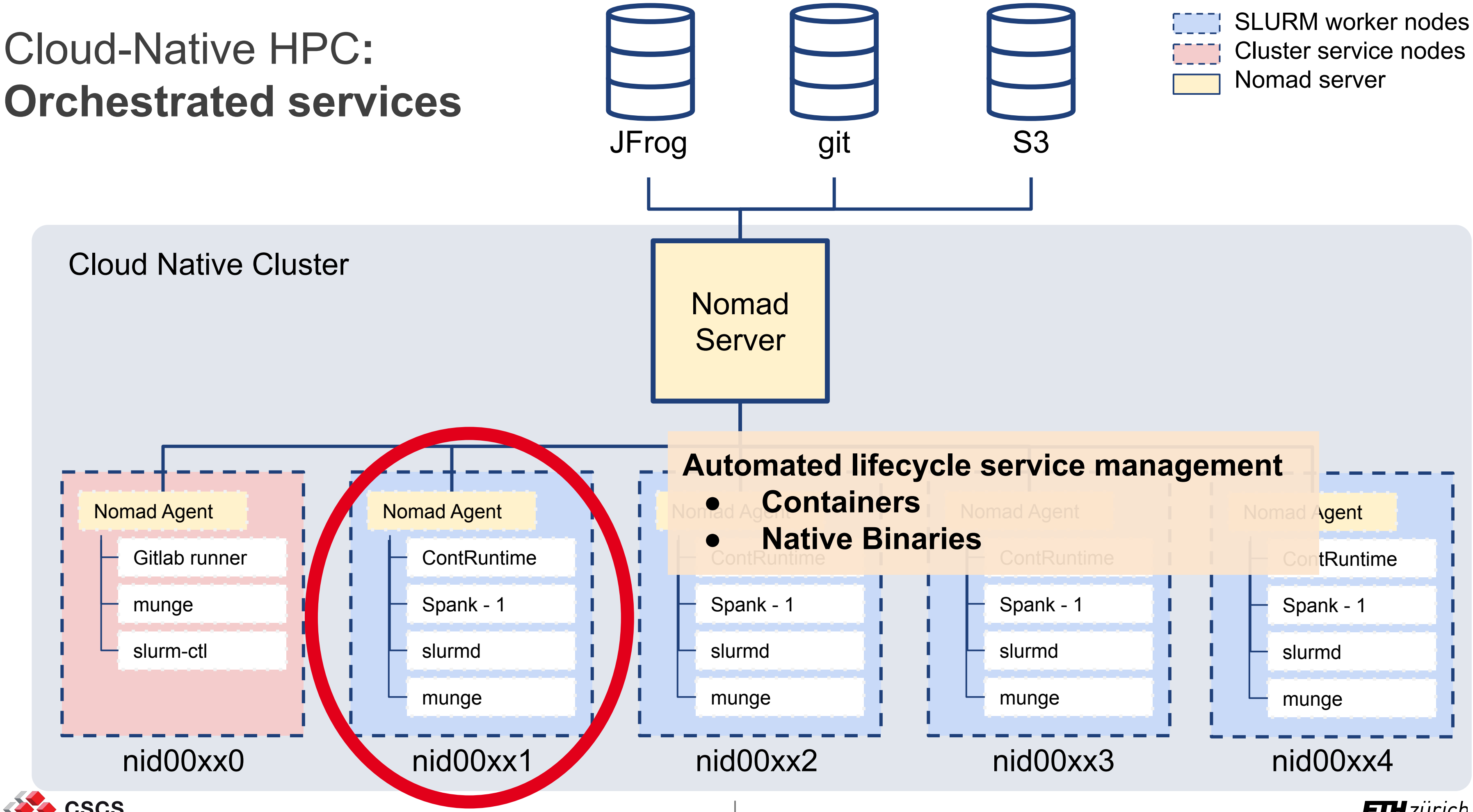
Cloud-Native HPC: Orchestrated Services

Artifact Repositories

- **Service applications**
  - **Containers**
  - **Binaries**
- **Configuration files**



Cloud-Native HPC:
Orchestrated services



- ▨ SLURM worker nodes
- ▨ Cluster service nodes
- ▨ Nomad server

JFrog

git

S3

Cloud Native Cluster

Nomad Server

nid00xx0: Nomad Agent – Gitlab runner, munge, slurm-ctl

nid00xx1: Nomad Agent – ContRuntime, Spank - 1, slurmd, munge

nid00xx2: Nomad Agent – ContRuntime, Spank - 1, slurmd, munge

nid00xx3: Nomad Agent – ContRuntime, Spank - 1, slurmd, munge

nid00xx4: Nomad Agent – ContRuntime, Spank - 1, slurmd, munge

Automated lifecycle service management
- **Containers**
- **Native Binaries**

On HPC microservices

Cloud-Native HPC:
On HPC microservices

- HCL file describes the service
- Specify resources to use
- Specify the artifacts it consumes
- Most services try containers

- Traditional HPC service
  - Not container friendly
  - Conflict of cgroups
  - Conflict of namespaces
  - Heavy with IPC

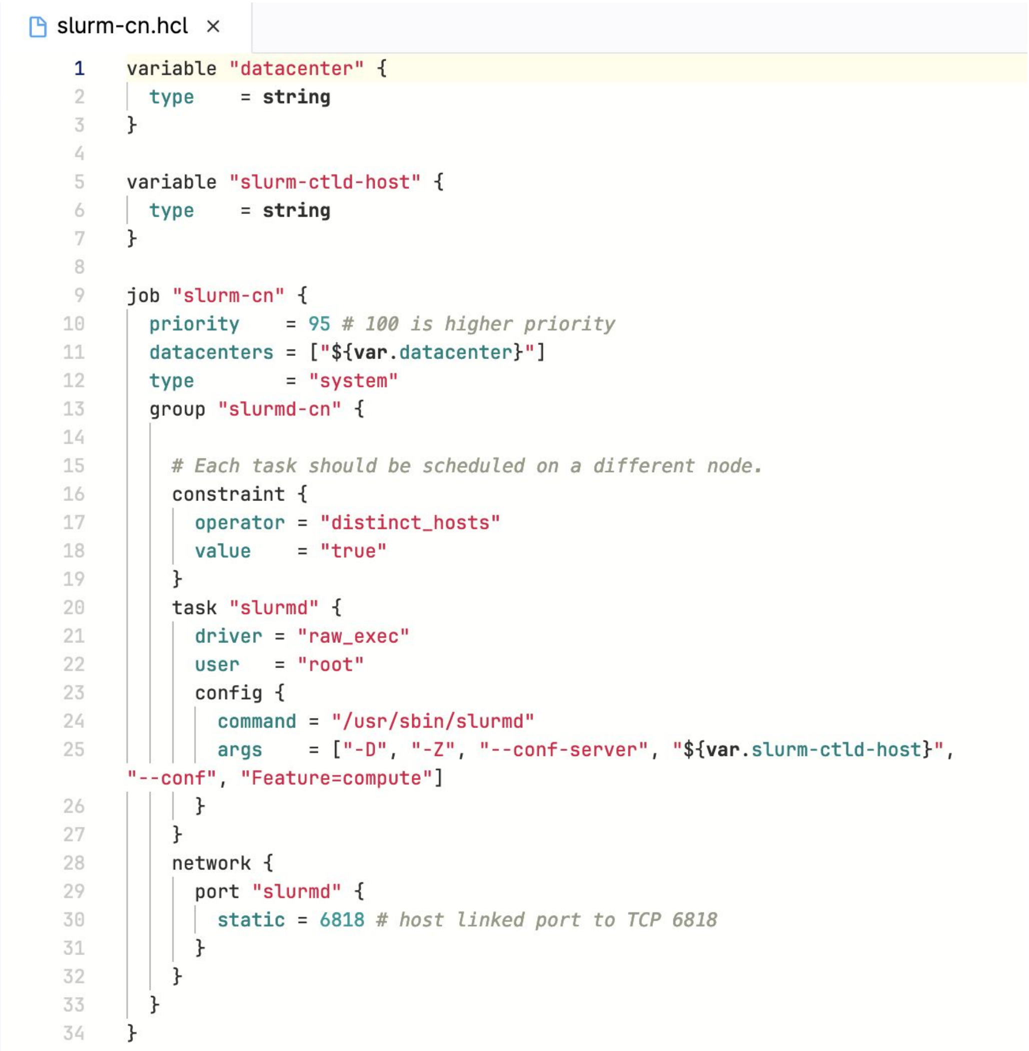
- Nomad’s flexibility
  - Use “raw_exec” driver
  - Native binary
  - With arguments
  - As root
  - Pass config artifacts

Cloud-Native HPC:
On HPC microservices

- HCL file describes the service
- Specify resources to use
- Specify the artifacts it consumes
- Most services try containers

- Traditional HPC service
  - Not container friendly
  - Conflict of cgroups
  - Conflict of namespaces
  - Heavy with IPC

- Nomad's flexibility
  - Use "raw_exec" driver
  - Native binary
  - With arguments
  - As root
  - Pass config artifacts



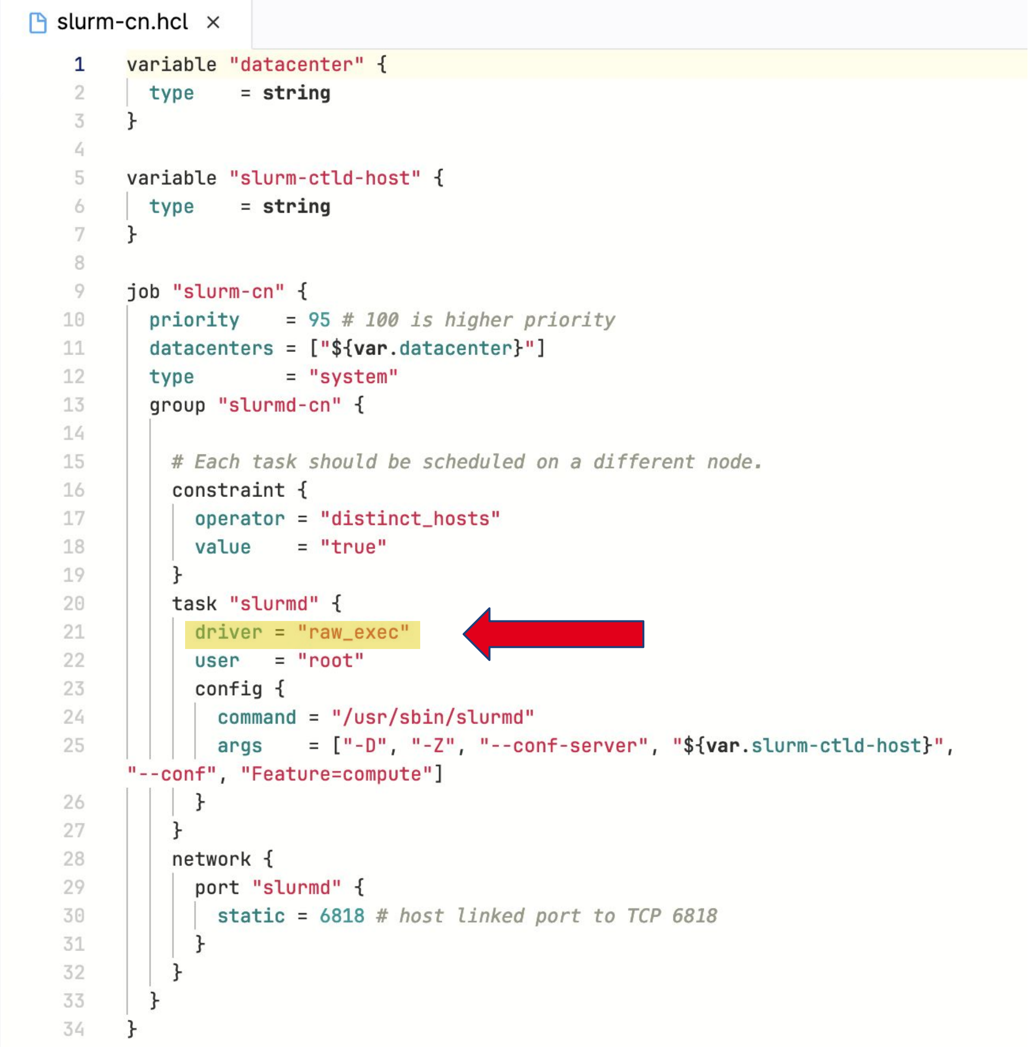
```
variable "datacenter" {
  type     = string
}

variable "slurm-ctld-host" {
  type     = string
}

job "slurm-cn" {
  priority    = 95 # 100 is higher priority
  datacenters = ["${var.datacenter}"]
  type        = "system"
  group "slurmd-cn" {

    # Each task should be scheduled on a different node.
    constraint {
      operator = "distinct_hosts"
      value    = "true"
    }
    task "slurmd" {
      driver = "raw_exec"
      user   = "root"
      config {
        command = "/usr/sbin/slurmd"
        args    = ["-D", "-Z", "--conf-server", "${var.slurm-ctld-host}", "--conf", "Feature=compute"]
      }
    }
    network {
      port "slurmd" {
        static = 6818 # host linked port to TCP 6818
      }
    }
  }
}
```

Cloud-Native HPC:
On HPC microservices

- HCL file describes the service
- Specify resources to use
- Specify the artifacts it consumes
- Most services try containers

- Traditional HPC service
  - Not container friendly
  - Conflict of cgroups
  - Conflict of namespaces
  - Heavy with IPC

- Nomad’s flexibility
  - **Use “raw_exec” driver**
  - Native binary
  - With arguments
  - As root
  - Pass config artifacts



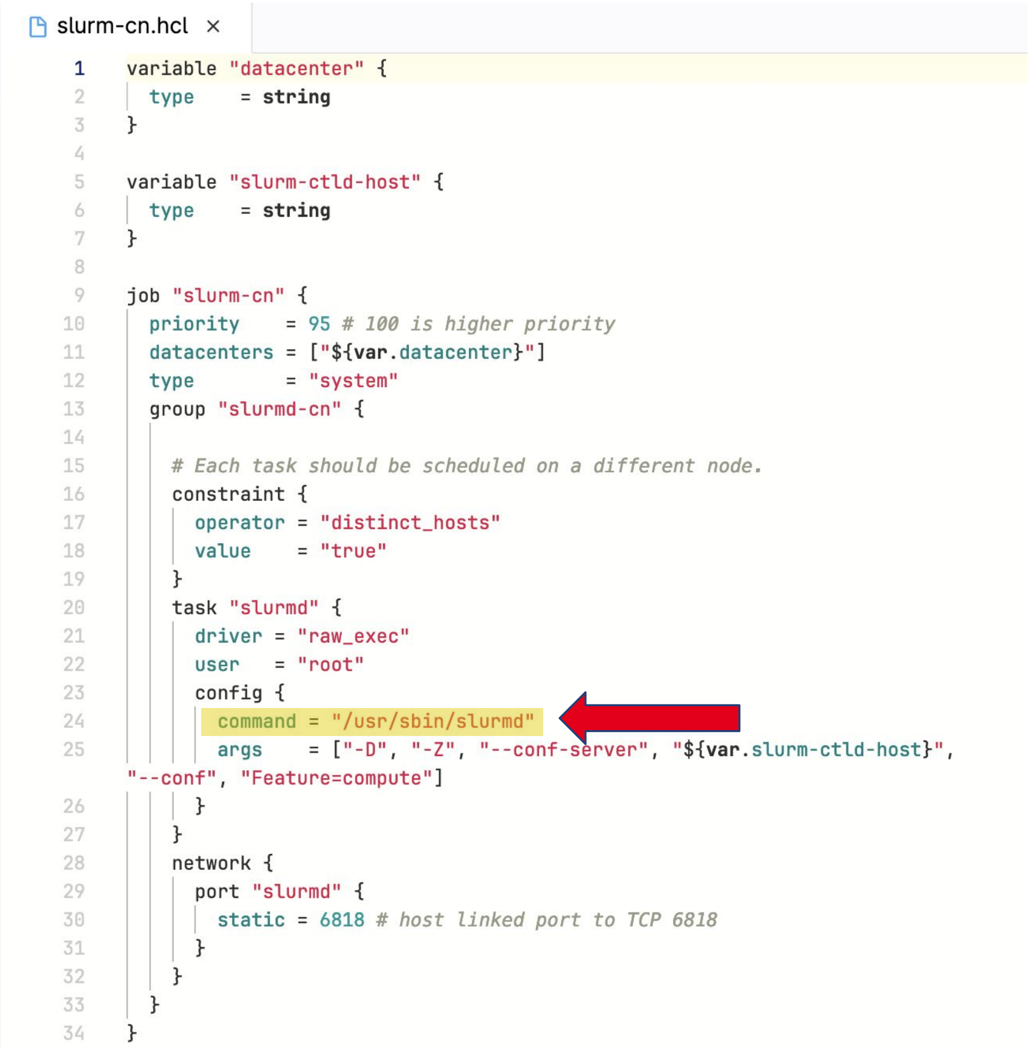
```
variable "datacenter" {
  type     = string
}

variable "slurm-ctld-host" {
  type     = string
}

job "slurm-cn" {
  priority    = 95 # 100 is higher priority
  datacenters = ["${var.datacenter}"]
  type        = "system"
  group "slurmd-cn" {

    # Each task should be scheduled on a different node.
    constraint {
      operator = "distinct_hosts"
      value    = "true"
    }
    task "slurmd" {
      driver = "raw_exec"
      user   = "root"
      config {
        command = "/usr/sbin/slurmd"
        args    = ["-D", "-Z", "--conf-server", "${var.slurm-ctld-host}", "--conf", "Feature=compute"]
      }
    }
    network {
      port "slurmd" {
        static = 6818 # host linked port to TCP 6818
      }
    }
  }
}
```

Cloud-Native HPC:
On HPC microservices

- HCL file describes the service
- Specify resources to use
- Specify the artifacts it consumes
- Most services try containers

- Traditional HPC service
  - Not container friendly
  - Conflict of cgroups
  - Conflict of namespaces
  - Heavy with IPC

- Nomad's flexibility
  - Use "raw_exec" driver
  - **Native binary**
  - With arguments
  - As root
  - Pass config artifacts



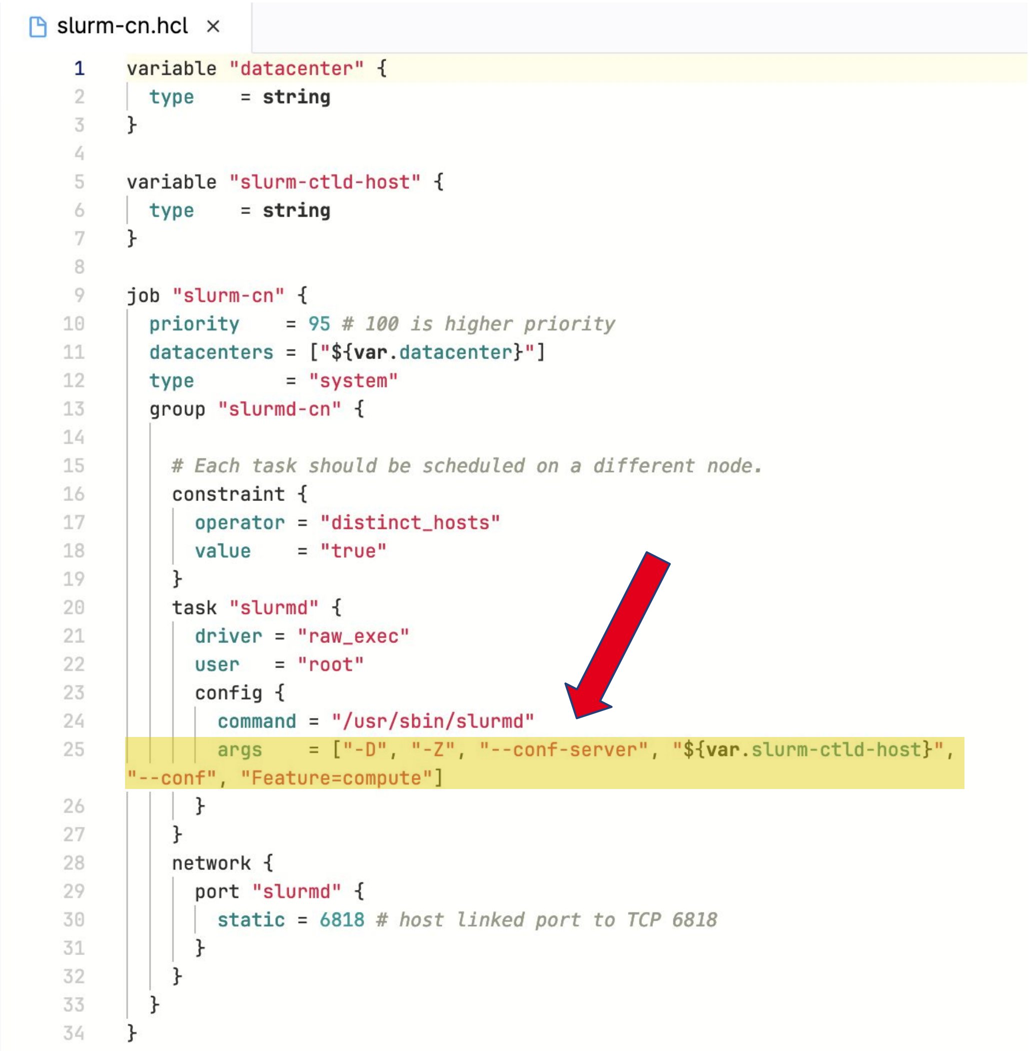
```hcl
variable "datacenter" {
  type     = string
}

variable "slurm-ctld-host" {
  type     = string
}

job "slurm-cn" {
  priority    = 95 # 100 is higher priority
  datacenters = ["${var.datacenter}"]
  type        = "system"
  group "slurmd-cn" {

    # Each task should be scheduled on a different node.
    constraint {
      operator = "distinct_hosts"
      value    = "true"
    }
    task "slurmd" {
      driver = "raw_exec"
      user   = "root"
      config {
        command = "/usr/sbin/slurmd"
        args    = ["-D", "-Z", "--conf-server", "${var.slurm-ctld-host}", "--conf", "Feature=compute"]
      }
    }
    network {
      port "slurmd" {
        static = 6818 # host linked port to TCP 6818
      }
    }
  }
}
```

Cloud-Native HPC:
On HPC microservices

- HCL file describes the service
- Specify resources to use
- Specify the artifacts it consumes
- Most services try containers

- Traditional HPC service
  - Not container friendly
  - Conflict of cgroups
  - Conflict of namespaces
  - Heavy with IPC

- Nomad’s flexibility
  - Use “raw_exec” driver
  - Native binary
  - **With arguments**
  - As root
  - Pass config artifacts



```hcl
variable "datacenter" {
  type     = string
}

variable "slurm-ctld-host" {
  type     = string
}

job "slurm-cn" {
  priority    = 95 # 100 is higher priority
  datacenters = ["${var.datacenter}"]
  type        = "system"
  group "slurmd-cn" {

    # Each task should be scheduled on a different node.
    constraint {
      operator = "distinct_hosts"
      value    = "true"
    }
    task "slurmd" {
      driver = "raw_exec"
      user   = "root"
      config {
        command = "/usr/sbin/slurmd"
        args    = ["-D", "-Z", "--conf-server", "${var.slurm-ctld-host}", "--conf", "Feature=compute"]
      }
    }
    network {
      port "slurmd" {
        static = 6818 # host linked port to TCP 6818
      }
    }
  }
}
```

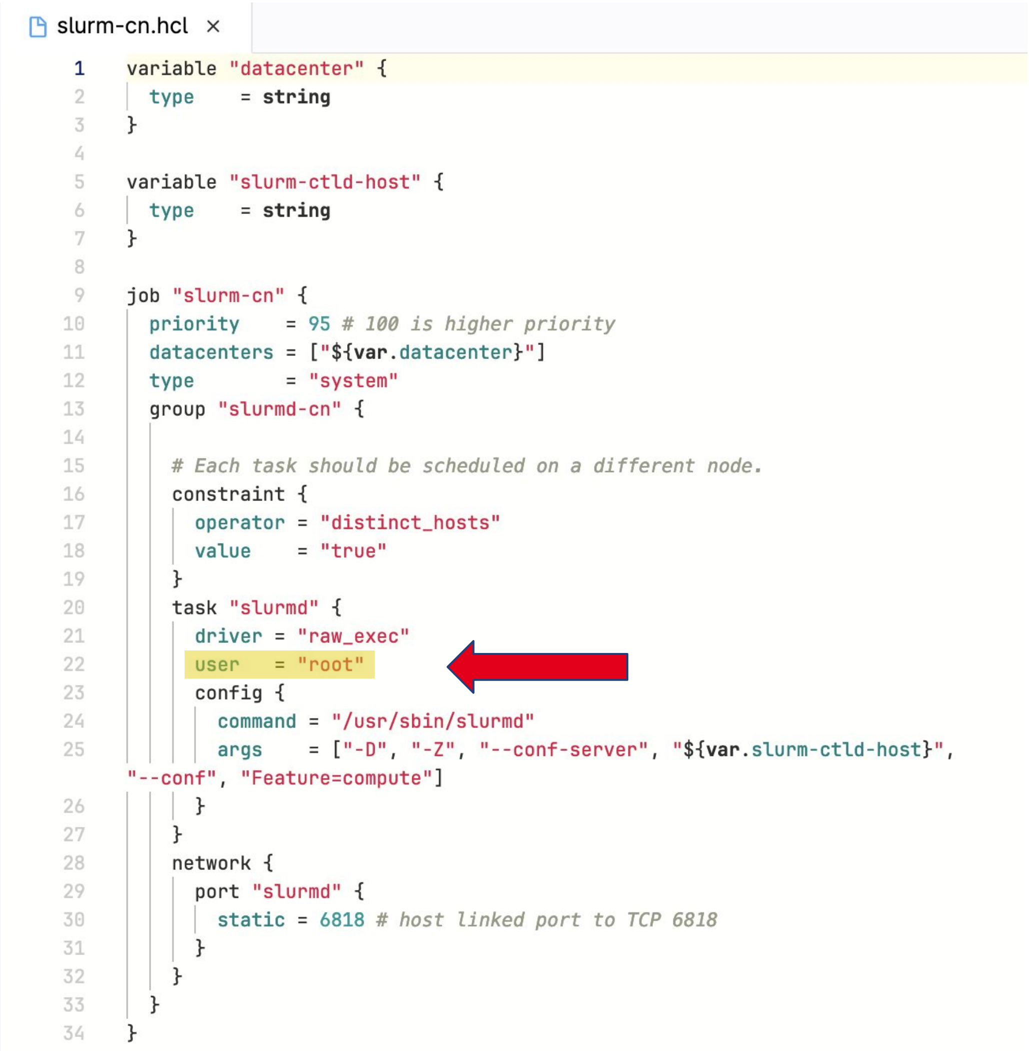
Cloud-Native HPC:
On HPC microservices

- HCL file describes the service
- Specify resources to use
- Specify the artifacts it consumes
- Most services try containers

- Traditional HPC service
  - Not container friendly
  - Conflict of cgroups
  - Conflict of namespaces
  - Heavy with IPC

- Nomad’s flexibility
  - Use “raw_exec” driver
  - Native binary
  - With arguments
  - **As root**
  - Pass config artifacts

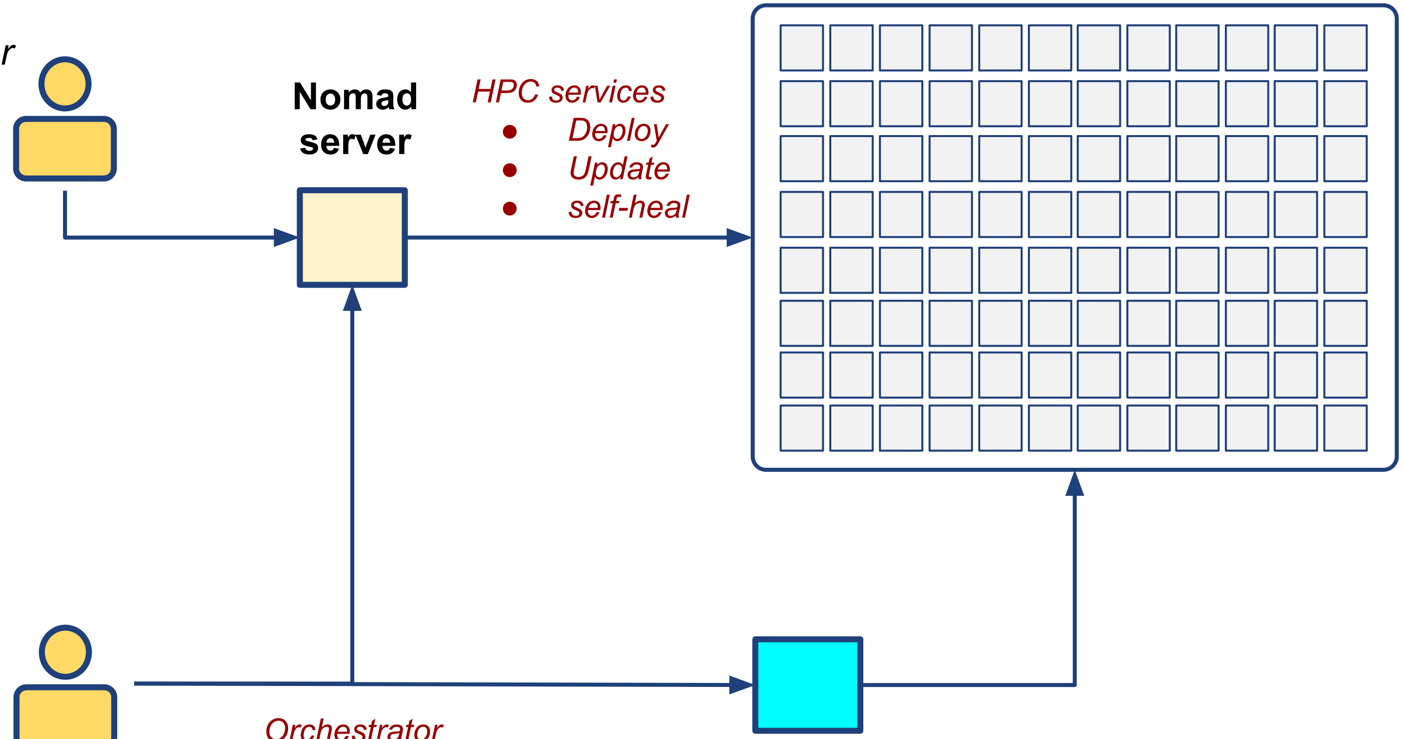
cscs



```
variable "datacenter" {
  type     = string
}

variable "slurm-ctld-host" {
  type     = string
}

job "slurm-cn" {
  priority    = 95 # 100 is higher priority
  datacenters = ["${var.datacenter}"]
  type        = "system"
  group "slurmd-cn" {

    # Each task should be scheduled on a different node.
    constraint {
      operator = "distinct_hosts"
      value    = "true"
    }
    task "slurmd" {
      driver = "raw_exec"
      user   = "root"
      config {
        command = "/usr/sbin/slurmd"
        args    = ["-D", "-Z", "--conf-server", "${var.slurm-ctld-host}", "--conf", "Feature=compute"]
      }
    }
    network {
      port "slurmd" {
        static = 6818 # host linked port to TCP 6818
      }
    }
  }
}
```

Recap: Bootstrapping a vCluster

Cloud-Native HPC: **bootstrapping a vCluster**



Cloud-Native HPC cluster

Cluster administrator

Nomad server

HPC services

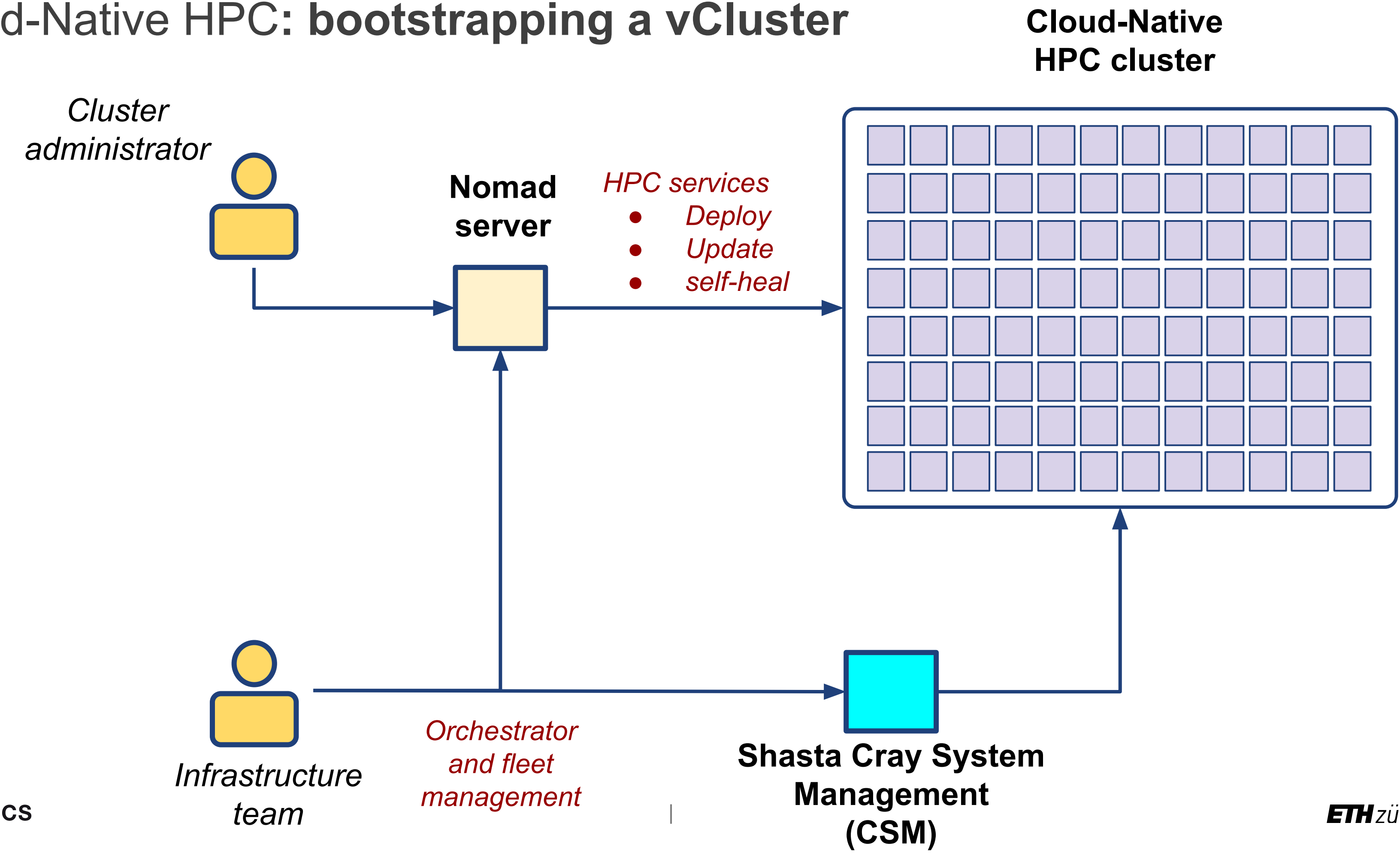
- *Deploy*
- *Update*
- *self-heal*

Infrastructure team

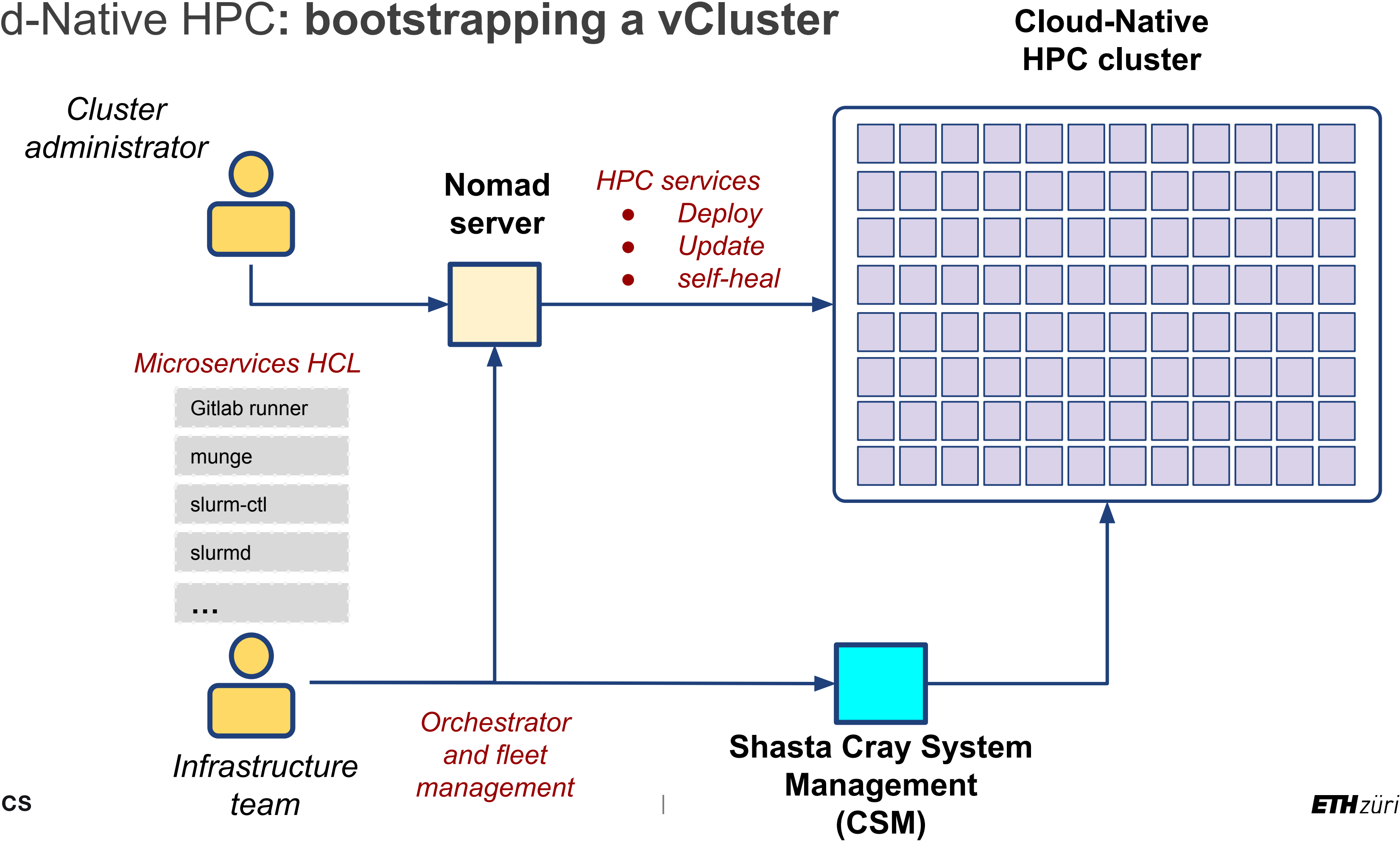
Orchestrator and fleet management

Shasta Cray System Management (CSM)

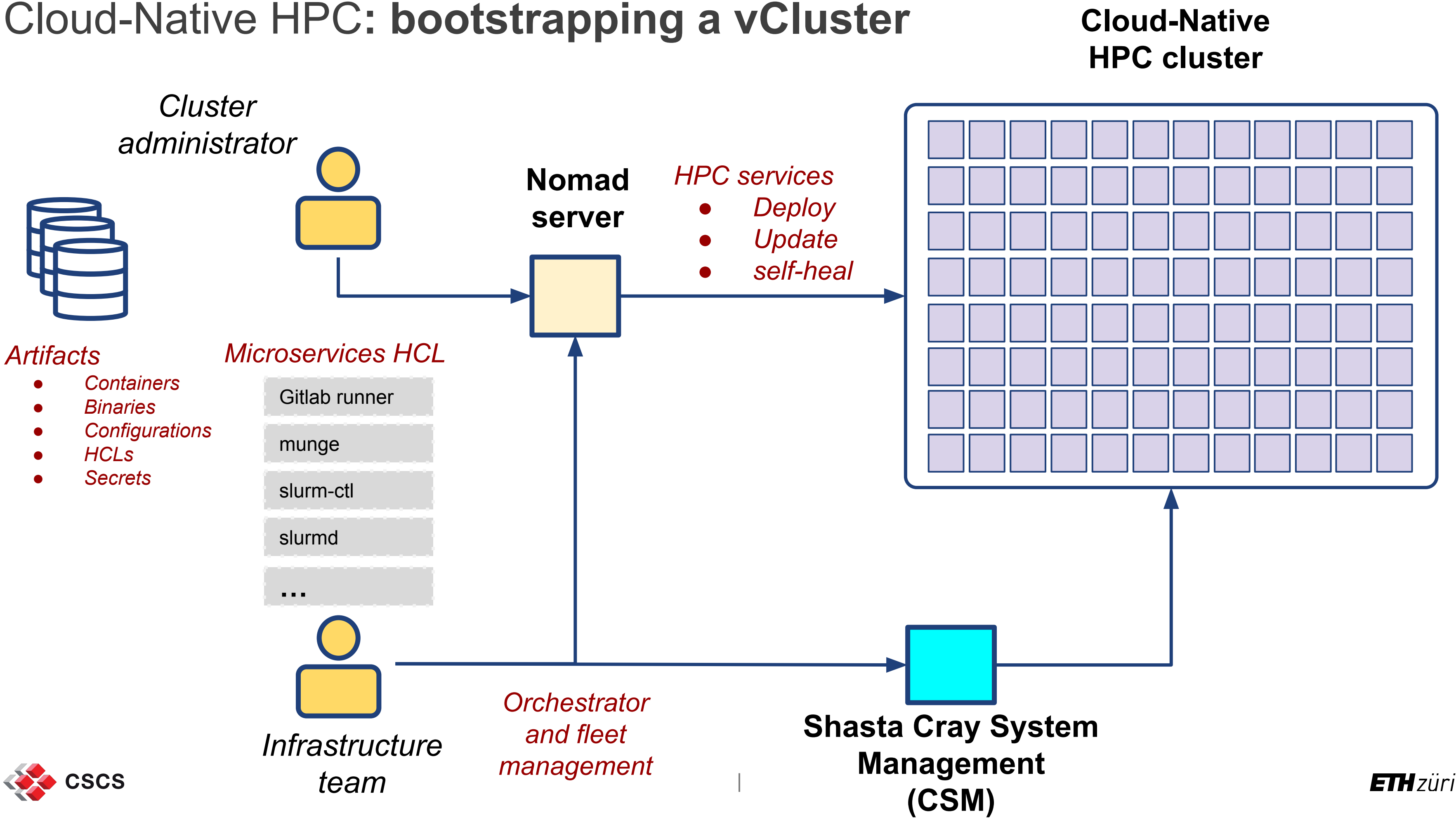
Cloud-Native HPC: **bootstrapping a vCluster**



Cloud-Native HPC: **bootstrapping a vCluster**



Cloud-Native HPC: **bootstrapping a vCluster**



Cloud-Native HPC cluster

Cluster administrator

Nomad server

HPC services
- *Deploy*
- *Update*
- *self-heal*

Artifacts
- *Containers*
- *Binaries*
- *Configurations*
- *HCLs*
- *Secrets*

Microservices HCL

Gitlab runner

munge

slurm-ctl

slurmd

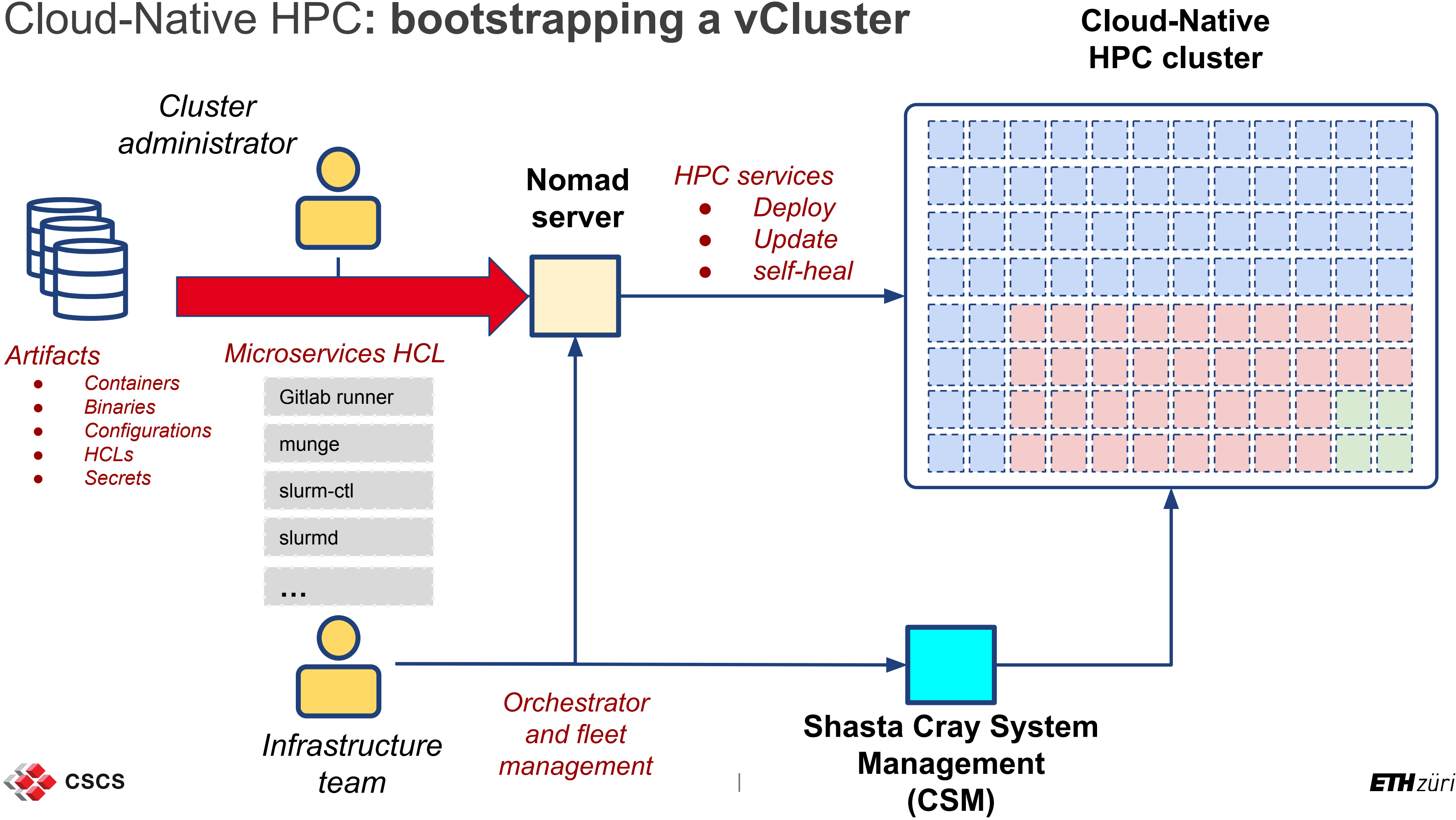
…

Infrastructure team

Orchestrator and fleet management

Shasta Cray System Management (CSM)

Cloud-Native HPC: **bootstrapping a vCluster**

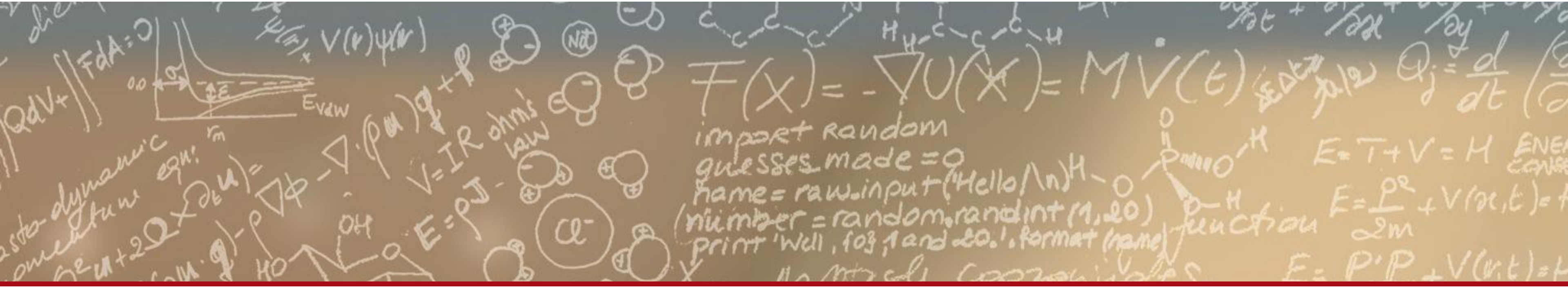


Cloud Native capabilities

- Microservice deployments
  - **Quick** vCluster deployments
  - **Reduced need for sysadmin interventions** (developer self-service)
  - Towards **independently deployable** services
    - Developed and operated by multiple teams
  - On **configuration** changes
    - Configuration artifact update + service restart
  - On **version** updates
    - New version artifact + update HCL + service restart
  - Dynamic resource utilization via orchestrator
    - **SW-defined** resource use: testing, staging, development, and production

Ongoing development

- Improve **microservice coverage** of traditional HPC services
- Implement more **CICD** pipelines for microservices
- DevOps **automation** of key microservices
  - Advanced self-healing
  - Node management
  - Enhance API capabilities for services management
- Work on a more granular **IAM** layer
- Improve **software delivery** options
- Build dashboards for improved **observability** of cluster
- Full HPC-containerized user environments (**containers-first for HPC**)


F(X) = -∇U(X) = MV(t)
import random
guesses_made = 0
name = raw_input('Hello\n')
number = random.randint(1, 20)
print 'Well, {0}, 1 and 20.'.format(name)
V=IR ohm's law
E=ρJ
E=T+V=H
Evdw
function

Thank you!