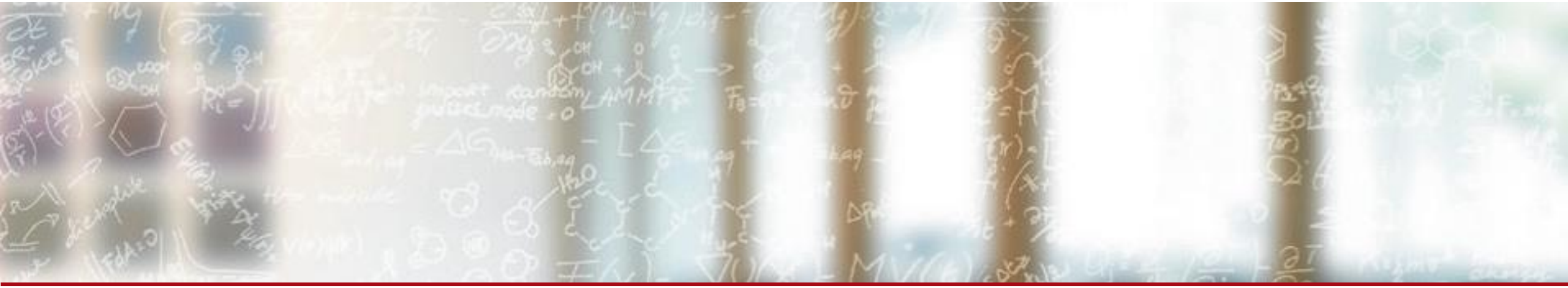
CSCS
Centro Svizzero di Calcolo Scientifico
Swiss National Supercomputing Centre

ETH zürich



Kubernetes Infrastructure at CSCS

Dino Conciatore – System Engineer
Elia Oggian – System Engineer

Tuesday, May 28th, 2024

CSCS – Lugano, Switzerland

Background

- Operating a **multi-tenant Kubernetes** cluster can be very complex
  - Based on past experience with fulen, a large multi-tenant cluster with multiple VLANs and users with varying requirements and workflows
- Improvements were necessary to ensure a **greater user experience**
- **Simplify the deployment** of CSCS internal services
- **Security and Isolation**:
  - network isolation should be implemented per cluster or purpose

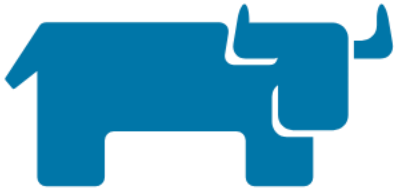
Introducing Key Components

Introduction to Harvester

Harvester is an open-source hyperconverged infrastructure solution

- **Kubernetes** as orchestration engine
- Based on **KVM** and **KubeVirt**, which enable virtualization and orchestration of VMs
- **Storage** management through Longhorn to provide a persistent storage
- **Networking** based on **Multus** enables isolation on multiple VLANs

Introduction to Rancher



Rancher is an open-source Kubernetes clusters manager.

- Centrally manage **multiple Kubernetes clusters**
- Simplified **provisioning and scaling** of clusters, node management, and upgrades.
- Strong **security features**, including role-based access control (RBAC), network policies and integration with identity providers to enhance security and compliance.
- Rancher has its own Kubernetes distribution called RKE2

Introduction to ArgoCD

Argo CD is an open-source continuous delivery tool specifically designed for Kubernetes that follows the GitOps methodology.

- **Application deployments** are declared using Kubernetes manifests or Helm charts stored in Git repositories.
  - Argo CD then ensures that the actual cluster state matches the desired configuration

- **Graphical UI** for visualizing and managing application deployments status. Additionally, it offers a command-line interface (CLI) for scripting and automation.

CSCS
Centro Svizzero di Calcolo Scientifico
Swiss National Supercomputing Centre

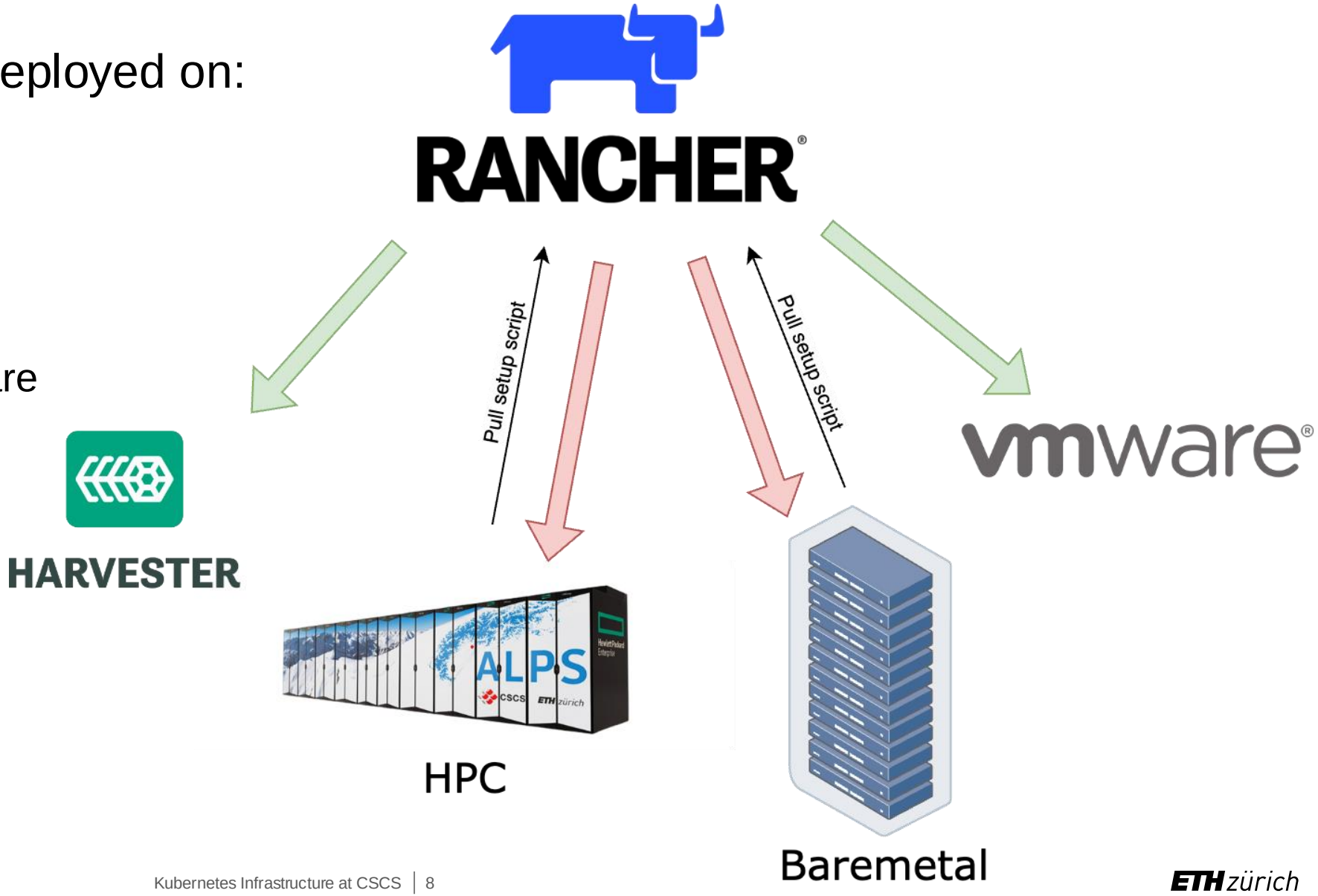
ETH zürich

Hyperconverged Cloud Infrastructure at CSCS

Where we run kubernetes

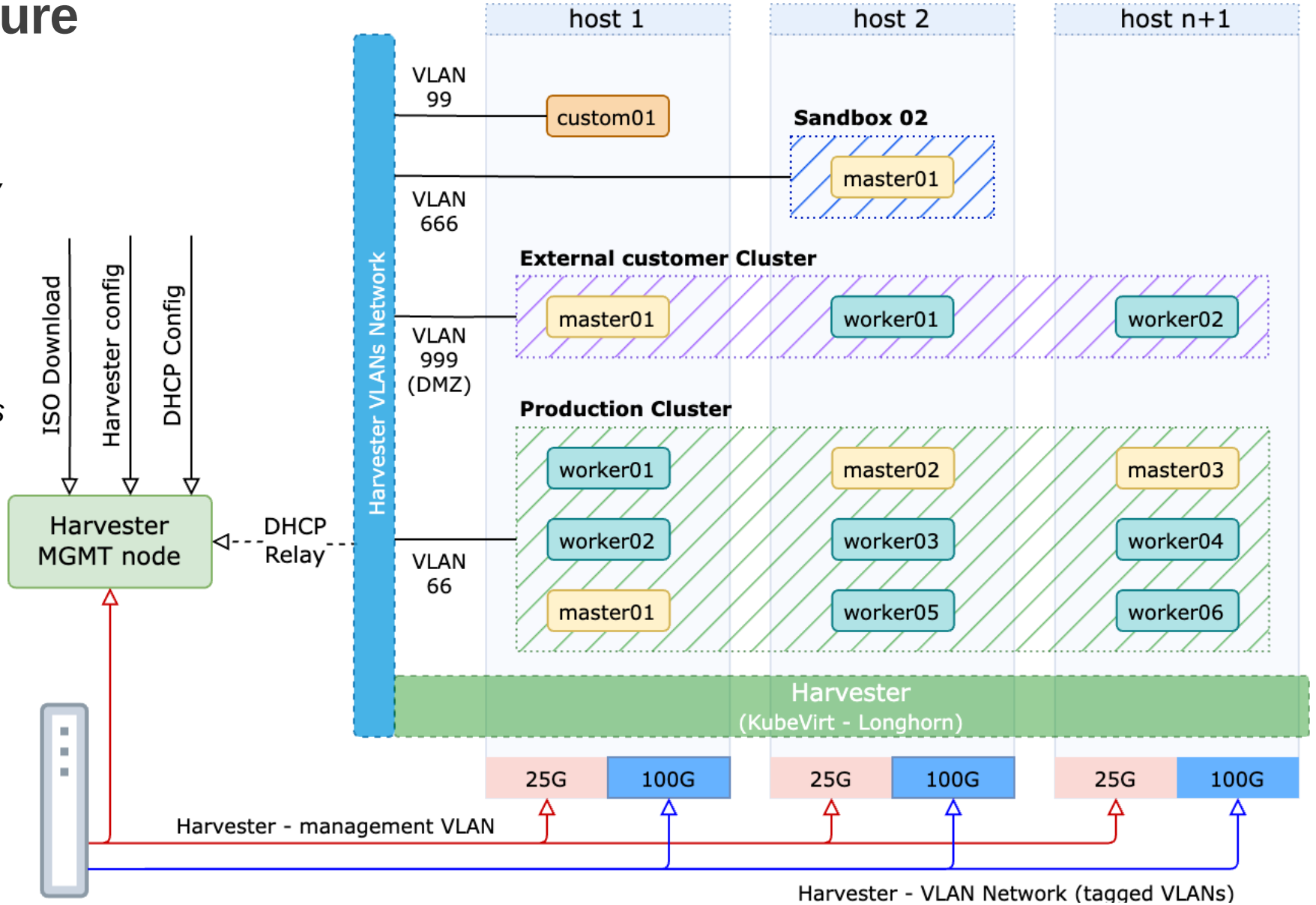
Kubernetes clusters deployed on:

- Harvester
- VMware
- Bare metal
  - Commodity Hardware
  - HPC (CSCS Alps)



Harvester architecture

- **Harvester host:**
  - Management (25G)
    - *Kubernetes overlay*
    - *VMs Live migration*
    - *Harvester StorageClass*
  - VLANs (100G)
    - *Downstream clusters*
  - Local NVMEs
  - 512G ram
  - 64 Cores
- **Management node:**
  - DHCP Relay for VMs
  - Hosts setup
    - iPXE Boot
    - ISO Repository
    - Hosts config

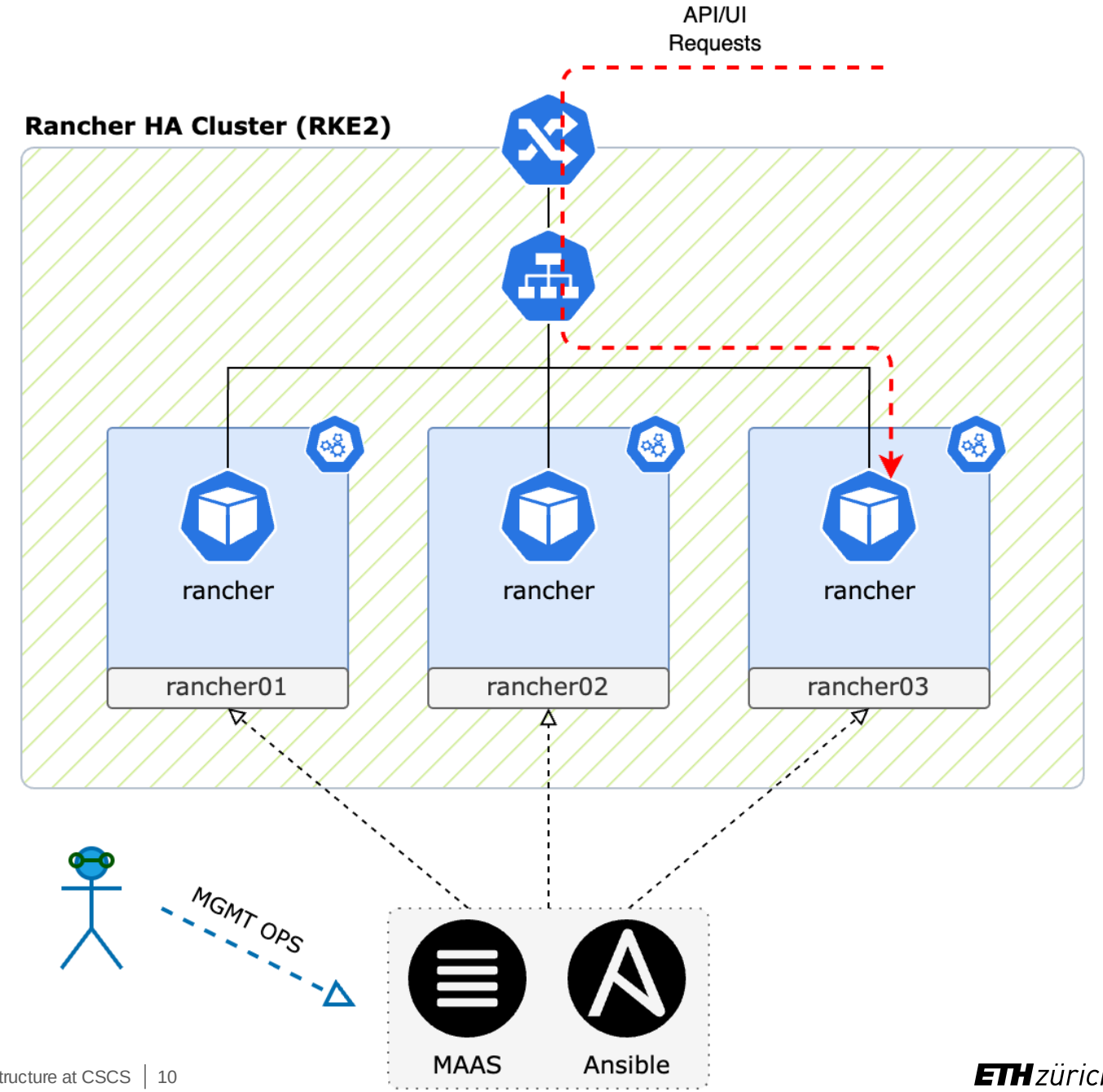


Rancher cluster architecture

- RKE2 Cluster
- 3 nodes both master and worker
- Each node is expected to have a Rancher pod
- kube-vip enables external access to the UI/API

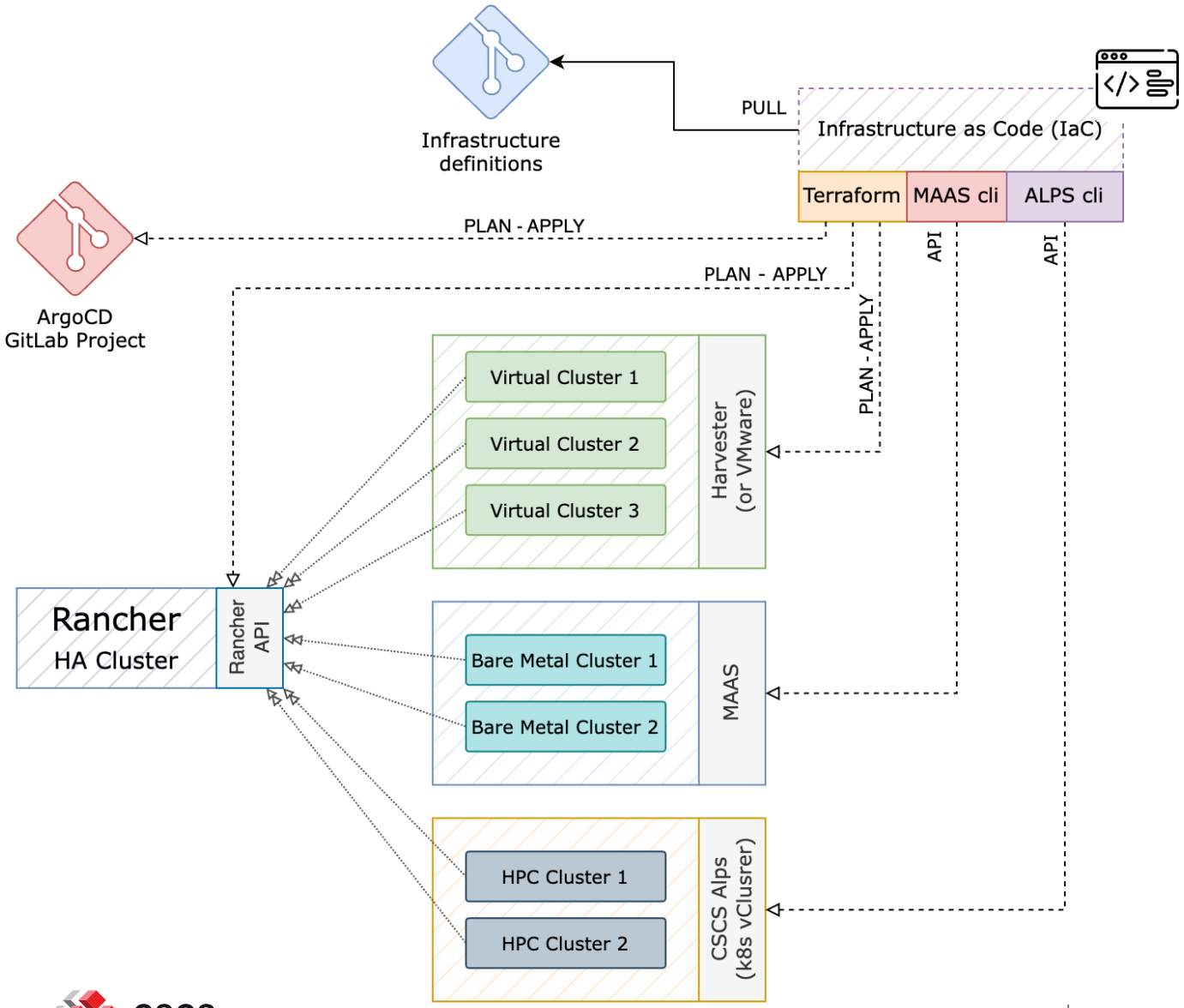
Node management:

- Node lifecycle through MAAS
- RKE2 installed and managed with Ansible
- Rancher deployment managed with Helm



Infrastructure as a code and GitOps

Infrastructure as a code



- **Harvester (or VMware)**
  - Define the cluster and git push:
    - CPUs, Ram, Nodes, VLAN

```
$ terraform apply
```

  - Triggers the VMs creation (masters/workes)
  - Rancher will run the cluster join command on the newly created VMs

- **MAAS (Bare Metal nodes)**
  - Define the cluster and git push:

```
$ terraform apply
```

```
$ ansible-playbook deploy-rke2.yml
```

  - Nodes will join the cluster

- **CSCS Alps (HPC)**

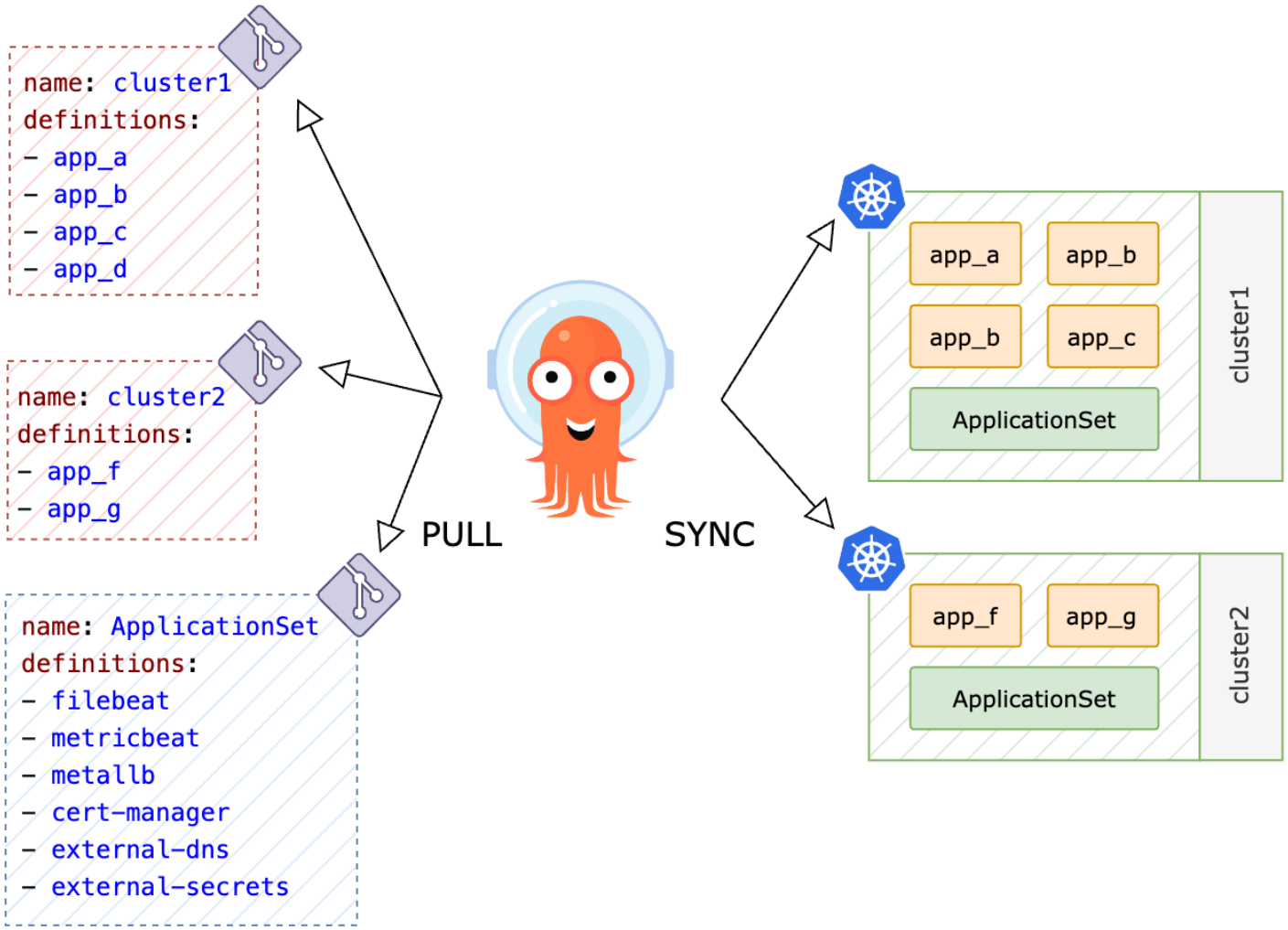
```
$ terraform apply
```

  - Master nodes creation on Harvester
    - And join

```
$ ./rancher-agent-install.sh
```

  - Worker nodes will join the cluster

GitOps with ArgoCD



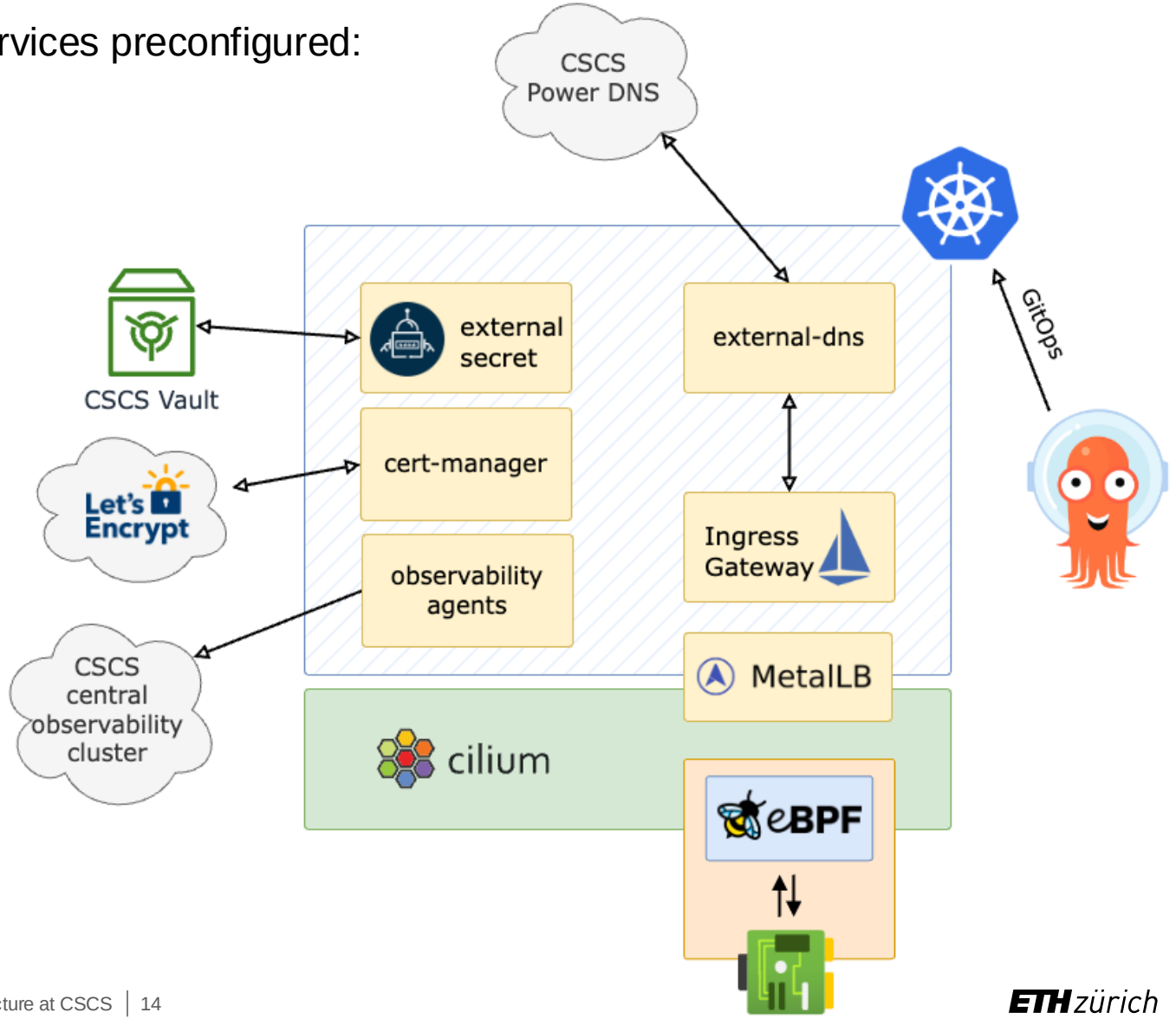
- **Application definitions**
  - Each cluster has its own Git repo with all application manifests (HELM or pure k8s manifests)
- **ApplicationSet**
  - Using ArgoCD Generators common applications can be deployed on all clusters or on clusters with specific labels

Argo CD ensures that the state of apps matches the desired configuration.

Kubernetes clusters common features

All clusters deployed have the following common services preconfigured:

- CNI: Cilium
  - Service mesh (eBPF)
  - Hubble (observability UI)
- CSI:
  - cephfs
  - ceph rbd
- Ingress controller
  - Istio (no sidecars)
    - Started migration to new API Gateway
  - Nginx
- MetalLB
  - Gratuitous ARP
- Automated DNS and Certs
- External Secret
  - For all Vault secrets
- Observability agents
  - Filebeat and Metricbeat



Observability

All clusters come with automatically deployed logs and metrics collectors (ArgoCD ApplicationSet), which send everything to our central Observability Infrastructure.

Currently we are using Elastic Beats

- Autodiscovery
  - Triggered on **`New`**, **`Modified`** and **`Deleted`** pod
- Hint based
  - Configure Beat modules via pod Annotations:

```
annotations:
        co.elastic.metrics/hosts: '"${data.host}:9404"'
        co.elastic.metrics/metrics_path: /metrics
        co.elastic.metrics/metricsets: collector
        co.elastic.metrics/module: prometheus
        co.elastic.metrics/processors.1.add_fields.fields.namespace: kafka
        co.elastic.metrics/processors.1.add_fields.target: data_stream
```

Operations

Infrastructure upgrade

- **Harvester**
  - Upgrade triggered automatically via the user interface.
    - RKE2 upgrade
    - Draining one host at a time
      - VMs are live migrated
    - OS Image replacement
- **Rancher**
  - Ansible managed nodes:
    - Drain Kubernetes nodes one by one.
    - Reinstall or upgrade the OS
  - Rancher
    - Upgrade done via Ansible (same playbook used for the installation)
      - `rke2_version: v1.26.9+rke2r1`
      - **NOTE:** Don't skip intermediate minor versions when upgrading, eg. : *v1.28.x → 1.29.x*
- **ArgoCD**
  - via Helm

Downstream clusters upgrade

- **Kubernetes**
  - Required RKE2 version can be changed using Terraform

    ```
    kubernetes_version = "v1.29.2+rke2r1"
    ```

    ```
    $ terraform apply
    ```

    - **NOTE:** Don't skip intermediate minor versions when upgrading, eg. : *v1.28.x → 1.29.x*
  - Rancher will handle draining nodes and upgrading
- **OS Upgrades**
  - Harvester nodes (VMs):
    - Required OS can be changed using Terraform
      - `image = "default/ubuntu-jammy”`
      - `$ terraform apply`
  - Ansible managed nodes:
    - Drain Kubernetes nodes one by one.
    - Reinstall or upgrade the OS based on your requirements

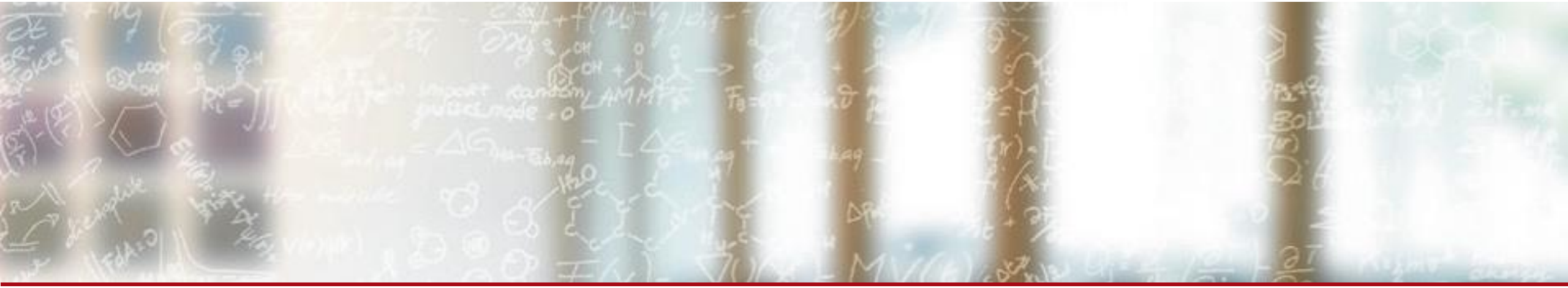
Use cases

Conclusion and Usecases

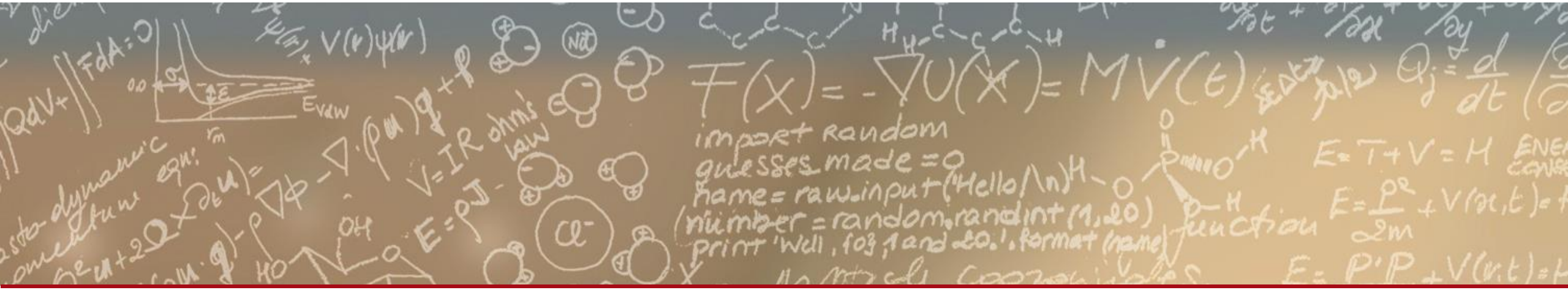
We are running a total of ~35 downstream clusters, with ~200 VMs and ~100 bare metal nodes

Here are some of the Kubernetes clusters:

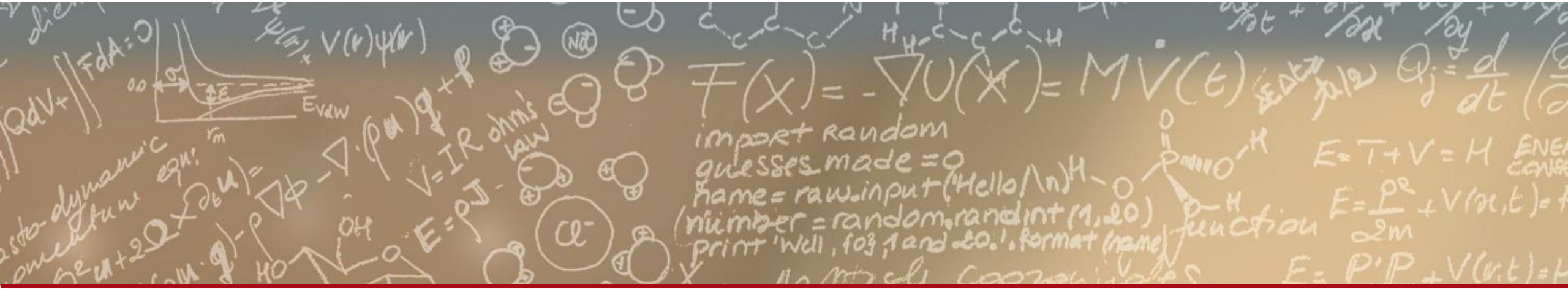
- dCache
  - 16 bare metal nodes
  - ~250 pools
  - 48TiB of PVCs via Ceph RBD (Total 12PB WLCG + CTA)
- WLCG Services
  - Frontier Squid
    - With multus and whereabouts (IPAM)
  - Nordugrid ARC CEs
  - site-bdii
- Observability cluster
  - 92 bare metal nodes
  - Local NVMe storage
    - Local Path for Elasticsearch data nodes
    - Longhorn for all other persistent services
- Central CSCS services
  - Vault
  - Jfrog
  - Gitlab-runners
- HPC Test (Rosa)
  - 3 master on Harvester
  - 4 worker on ALPS (1024 cores)



LIVE DEMO

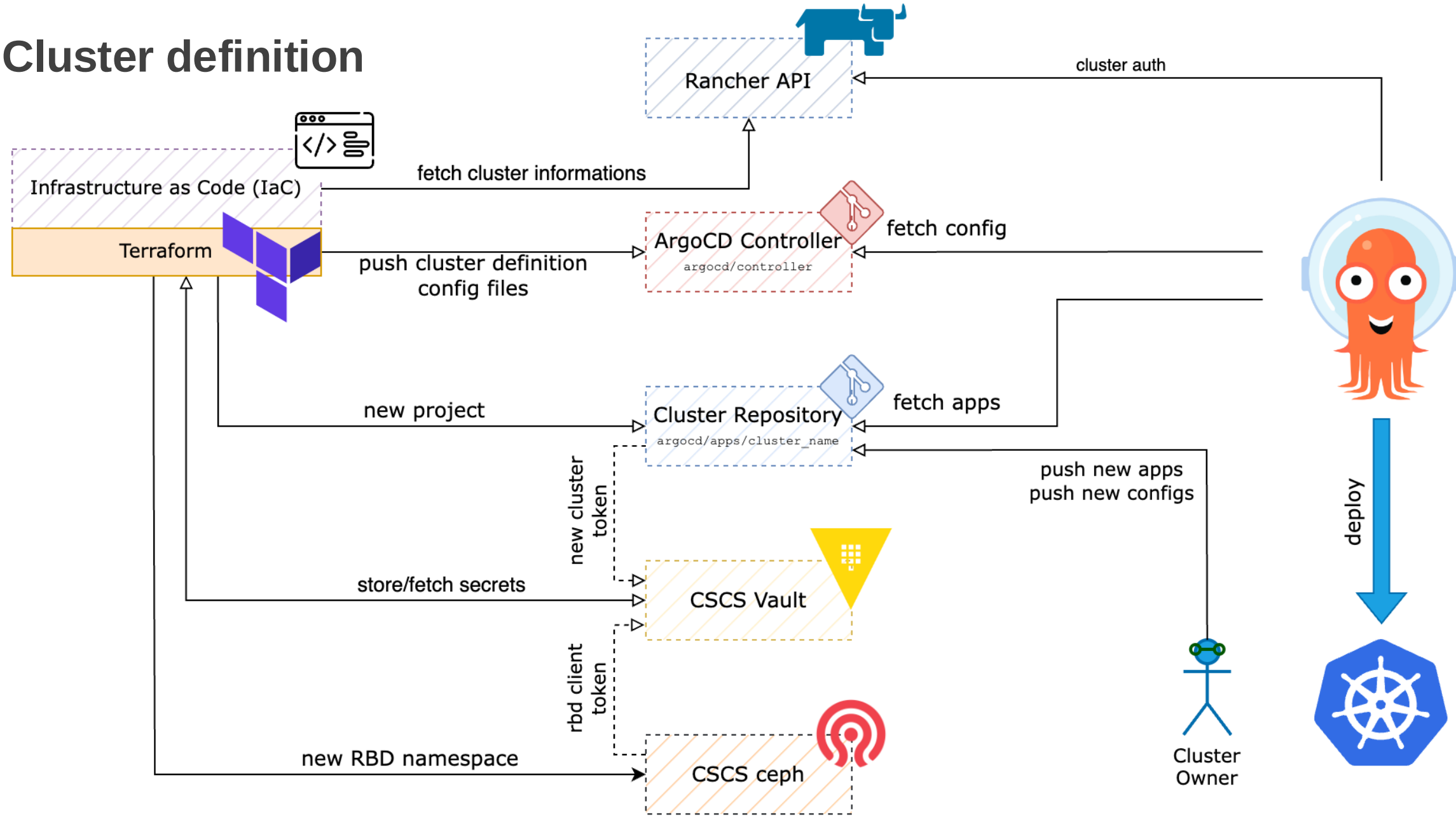

import random
guesses_made = 0
name = raw_input('Hello! What is your name?\n')
number = random.randint(1, 20)
print 'Well, ' + name + ', I am thinking of a number between 1 and 20.'

Questions?



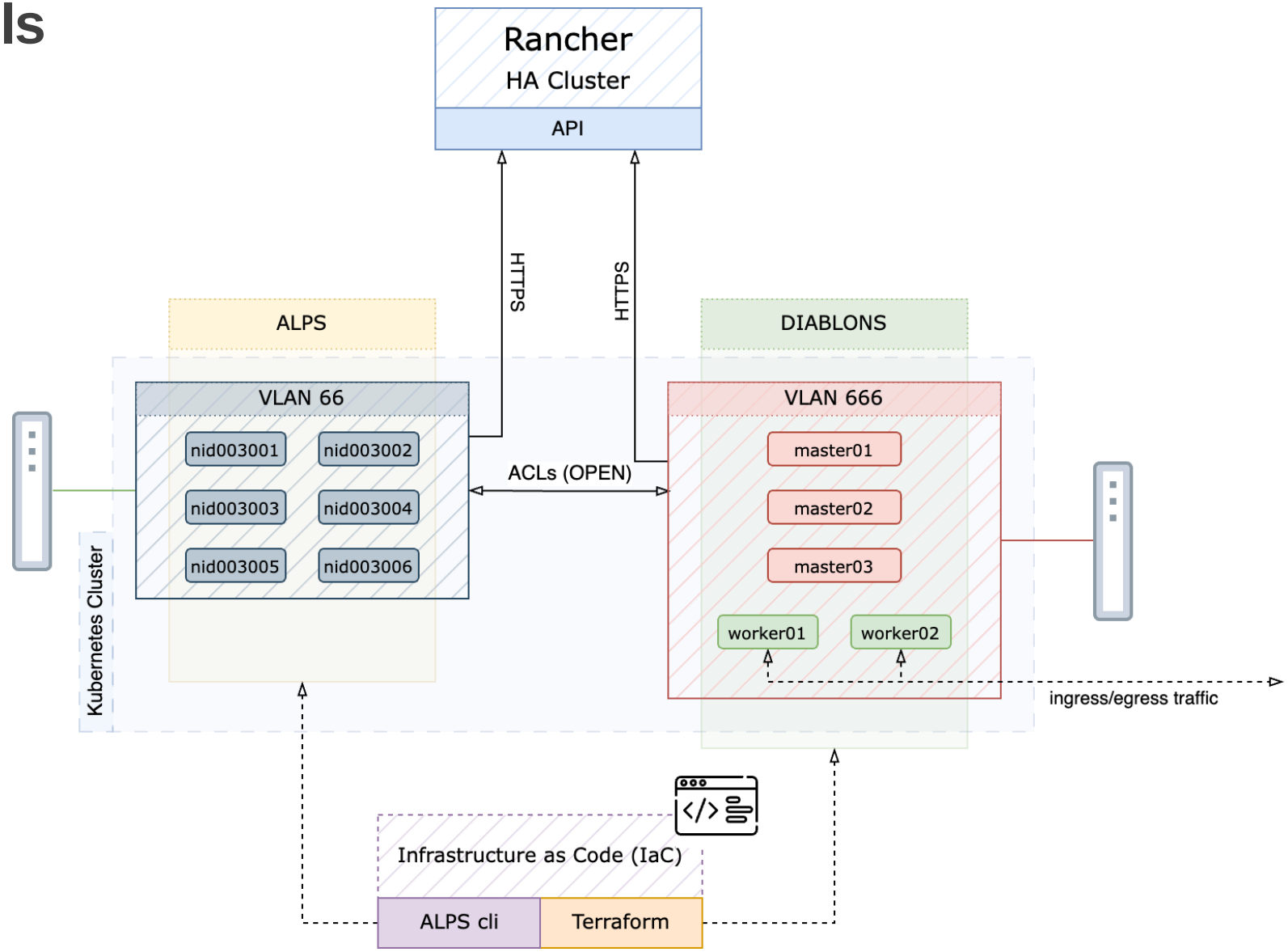
Thank you for your attention.

Cluster definition

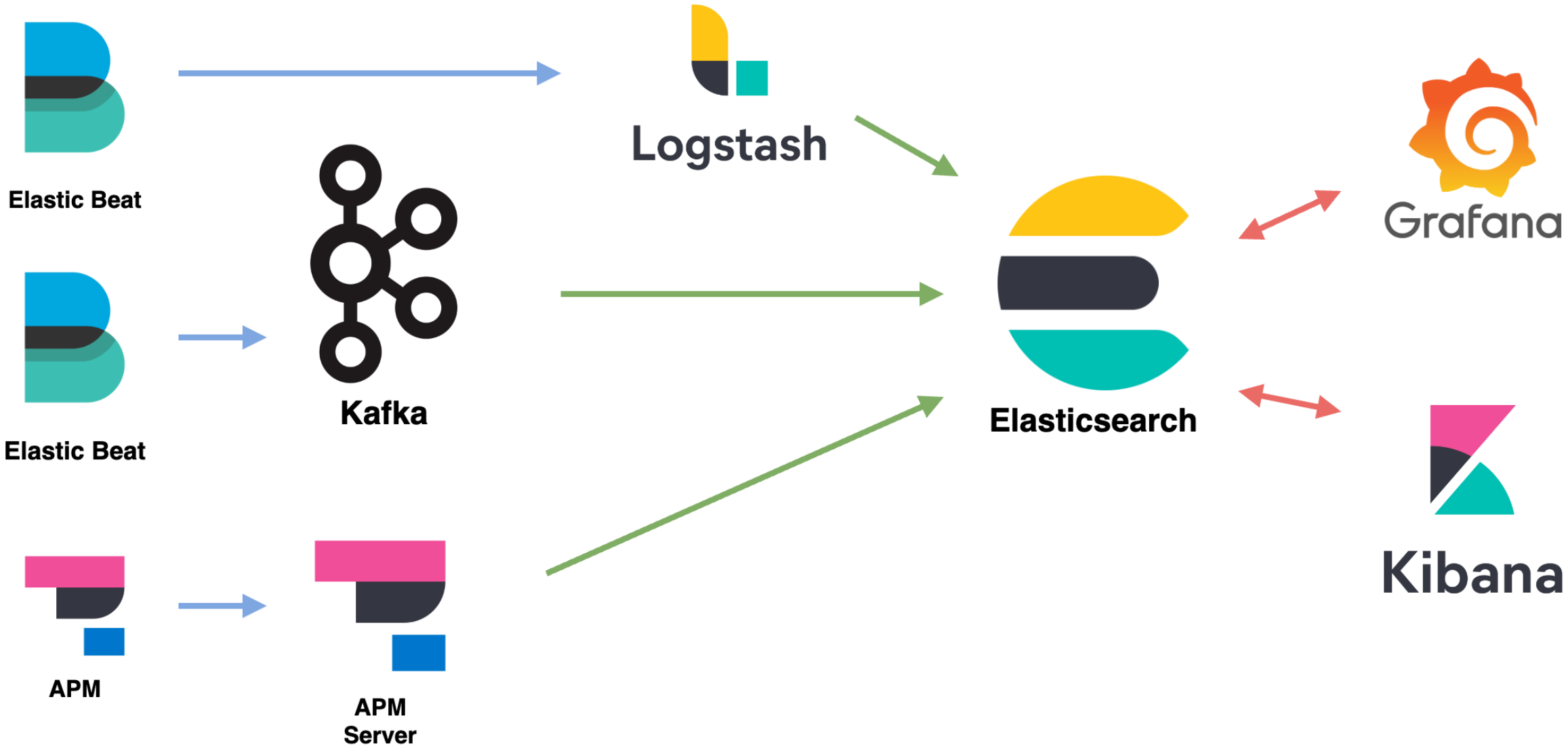


Kubernetes on alps details

- Harvester (DIABLONS)
  - Master nodes
  - Specific workload workers
    - Ingress Controller
    - Egress Controller
    - Low performance SVC
    - Etc…
- ALPS (HPC)
  - Stateless workernodes



Backup slide: Elastic Stack flow



Cilium tuning

```
rke2-cilium:
  bpf:
    masquerade: true
  egressGateway:
    enabled: true
  hubble:
    enabled: true
    relay:
      enabled: true
    ui:
      enabled: true
  ingressController:
    enabled: true
  k8sServiceHost: 127.0.0.1
  k8sServicePort: 6443
  kubeProxyReplacement: strict
```