Cluster Management: Unlocking the Secrets of Cluster API and Its Providers

Intro

Subhasmita Swain

SRE/R&D @CIVO

LFX Mentee'22 @CAPG,
Outreachy'22 @Apache

Aniruddha Basak

SWE @Syself

LFX Mentee'22 @CAPG

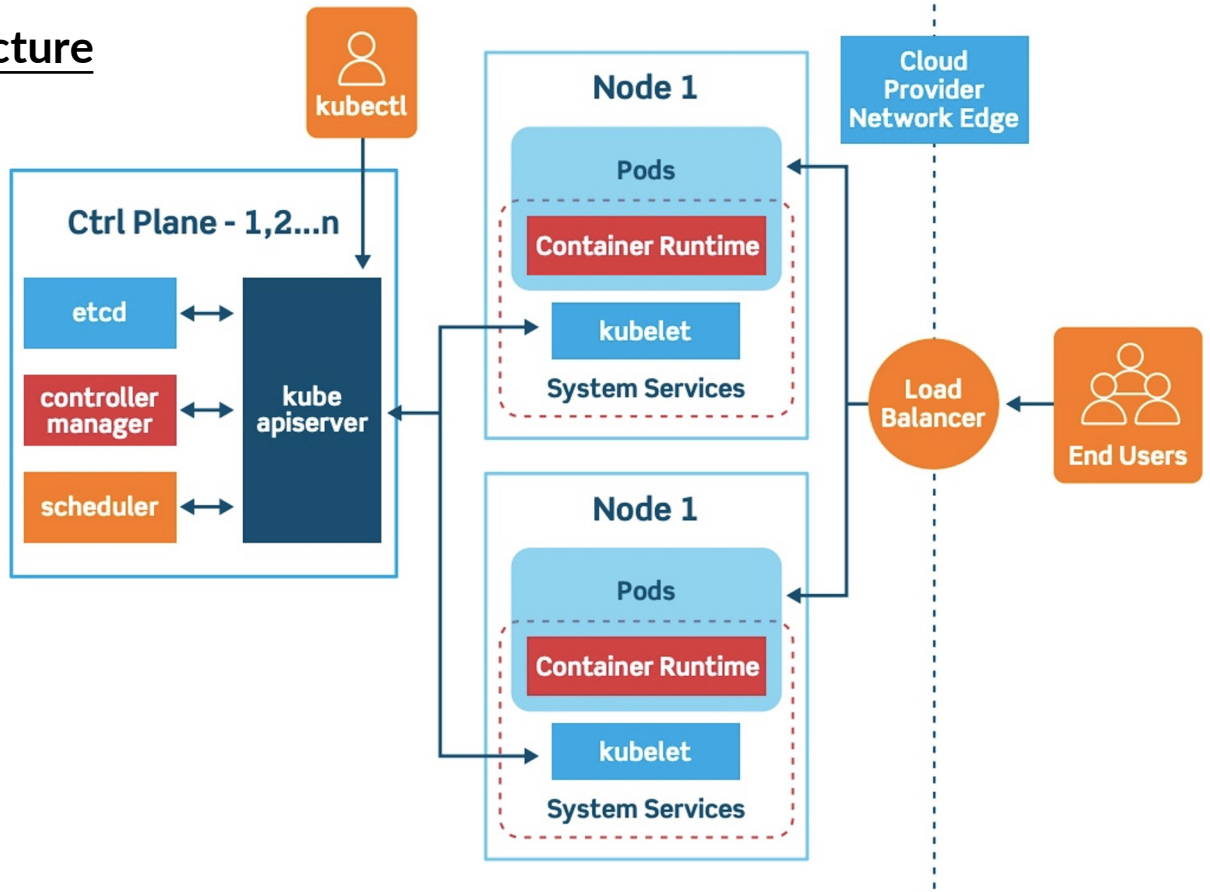
Agenda

- ➔ Cluster Lifecycle
- ➔ Cluster API Architecture
- ➔ Types of CAPI Providers
- ➔ Examples of Infrastructure Providers

All hail KUBERNETES!

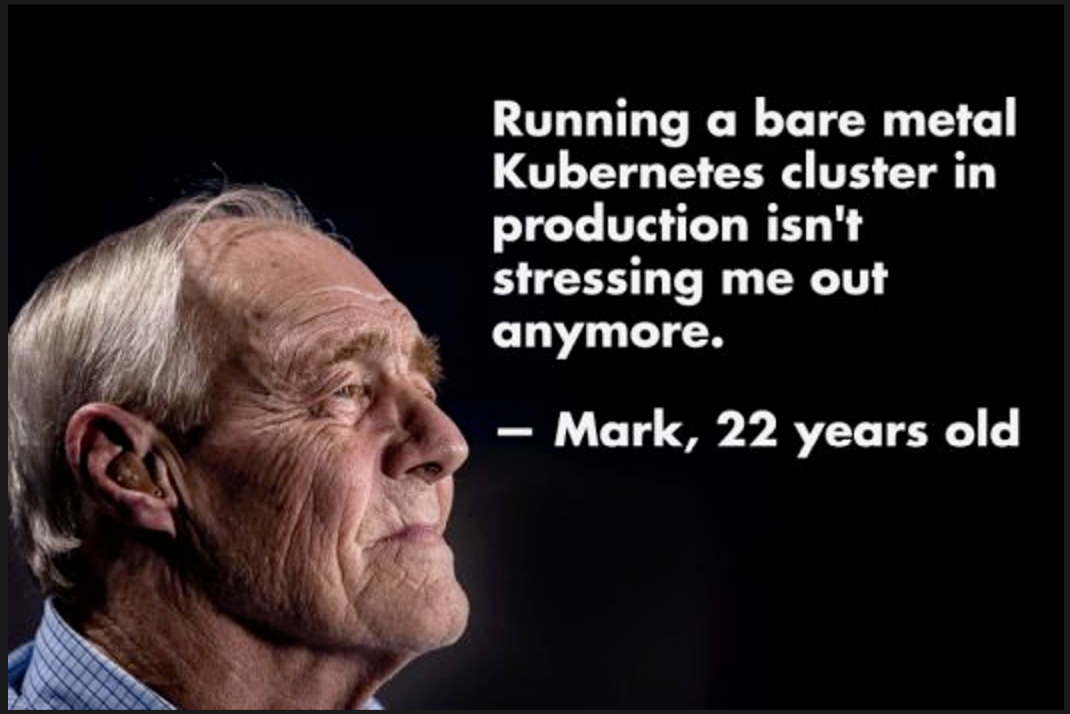
Kubernetes provides a framework to run distributed systems resiliently, scaling and managing the lifecycle of containerized applications.

K8s Cluster Architecture



Why manage clusters?

1. **Scaling and Load Balancing**
   - Channel traffic volume
   - Maintain uptime & performance
   - Distributed load balancing
2. **Automated Rollouts and Rollbacks**
   - Rollback changes if anything goes wrong
   - Progressively rollout changes on configs
3. **Self-Healing**
   - Can detect and replace instances that stop responding
   - Based on set params can kill/recreate unresponsive instances
4. Configuration management & Secrets
   - Manage sensitive information
   - Hierarchical secret masking


Running a bare metal Kubernetes cluster in production isn't stressing me out anymore.
— Mark, 22 years old

What is CAPI

The Cluster API is a Kubernetes bootstrapping project to bring declarative, Kubernetes-style APIs to multiple cluster creation, configuration, and management.

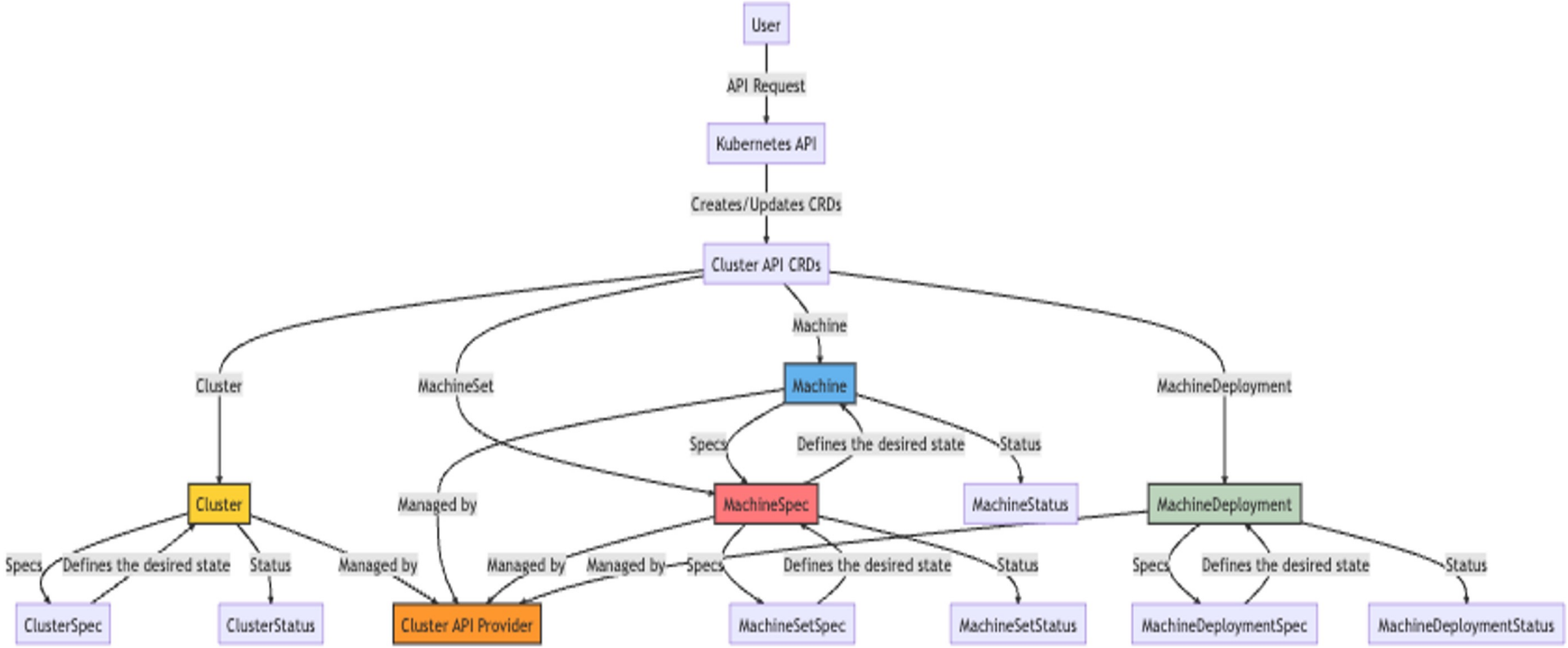
It provides optional, additive functionality on top of core Kubernetes to manage the lifecycle of a Kubernetes cluster.

CAPI, in its simplest form, is a K8s operator that implements **infrastructure-specific** functionality used in conjunction with the main Cluster API when controlling the lifetime of clusters.

Depending on the kind of provider, the operator implements a contract using its **CRDs (Custom Resources).**

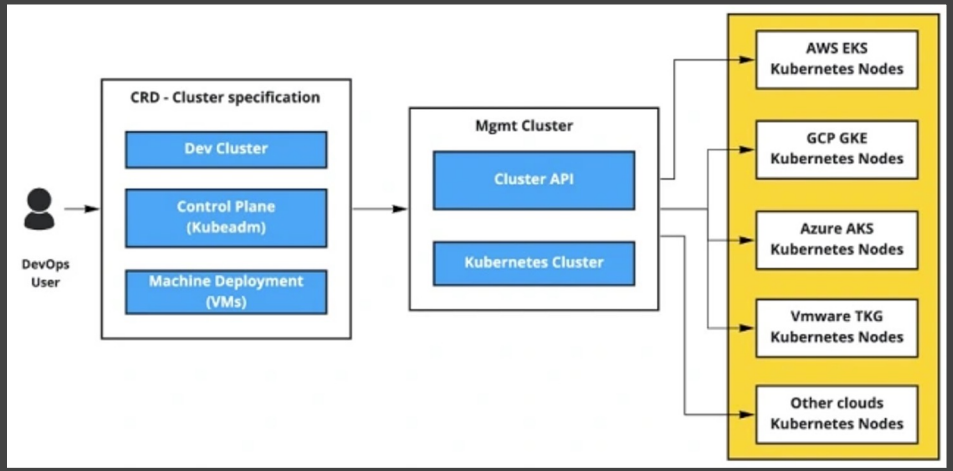
Today, over **100 Kubernetes distributions and installers** have been created, each with different default configurations for clusters and supported infrastructure providers.

Understanding the architecture


User
API Request
Kubernetes API
Creates/Updates CRDs
Cluster API CRDs
Machine
Cluster
MachineSet
Machine
MachineDeployment
Specs
Defines the desired state
Status
Cluster
Managed by
MachineSpec
MachineStatus
MachineDeployment
Specs
Defines the desired state
Status
Managed by
Managed by
Managed by
Specs
Defines the desired state
Status
Specs
Defines the desired state
Status
ClusterSpec
ClusterStatus
Cluster API Provider
MachineSetSpec
MachineSetStatus
MachineDeploymentSpec
MachineDeploymentStatus

CAPI Provider Types

1. Bootstrap
2. Control Plane
3. Infrastructure
4. IP Address Management (IPAM)
5. API Adopters and Addons



Bootstrap

01

It is responsible for initializing a machine with the necessary software to join a Kubernetes cluster.

- **Kubernetes Configurations:** generate certificates, setup network configs, install soft.
- **Declarative API:** Defined desired state of cluster
- **Integration with Infrastructure Providers**

Eg., **Kubeadm** sets up CP and worker nodes, a bootstrap provider. Similarly, EKS, Talos, MicroK8s

```yaml
apiVersion: bootstrap.cluster.x-k8s.io/v1beta1
kind: KubeadmConfigTemplate
metadata:
  name: "test-cluster-md-0"
spec:
  template:
    spec:
      joinConfiguration:
        nodeRegistration:
          kubeletExtraArgs:
            cloud-provider: gce
```

Control Plane

02

It is used to control the creation & lifecycle of the kubernetes control plane. It can utilize the resources created by bootstrap and infrastructure providers.

- Kubeadm control plane (KCP) is core, others include - MicroK8s, Talos, Nested.
- Managed K8s (i.e., AKS, GKS) implementations
- Can manage multiple CP nodes to ensure high availability of k8s clusters.

```yaml
kind: KubeadmControlPlane
apiVersion: controlplane.cluster.x-k8s.io/v1beta1
metadata:
  name: "test-control-plane"
spec:
  replicas: 3
  machineTemplate:
    infrastructureRef:
      kind: GCPMachineTemplate
      apiVersion: infrastructure.cluster.x-k8s.io/v1beta1
      name: "test-control-plane"
  kubeadmConfigSpec:
    initConfiguration:
      nodeRegistration:
        kubeletExtraArgs:
          cloud-provider: gce
    clusterConfiguration:
      apiServer:
        extraArgs:
          cloud-provider: gce
      controllerManager:
        extraArgs:
          cloud-provider: gce
    joinConfiguration:
      nodeRegistration:
        kubeletExtraArgs:
          cloud-provider: gce
  version: "1.26"
```

Infrastructure

03

Responsible for creating, managing and deleting infra resources. Eg., networking, security groups, hosts etc.

- Provider specific implementations.
- Integration with other providers.
- Widely used infra providers include - AWS (CAPA), Azure (CAPZ), GCP (CAPG), etc.

```yaml
apiVersion: infrastructure.cluster.x-k8s.io/v1beta1
kind: GCPCluster
metadata:
  name: "test-cluster"
spec:
  project: "test-project"
  region: "us-west1-a"
  network:
    name: "test-network"
---
kind: GCPMachineTemplate
apiVersion: infrastructure.cluster.x-k8s.io/v1beta1
metadata:
  name: "test-cluster-control-plane"
spec:
  template:
    spec:
      instanceType: "n2-standard-4"
      image: "123456789"
---
apiVersion: infrastructure.cluster.x-k8s.io/v1beta1
kind: GCPMachineTemplate
metadata:
  name: "test-cluster-md-0"
spec:
  template:
    spec:
      instanceType: "n2-standard-4"
      image: "123456789"
```

IP Address Management (IPAM)

04

It is a method of tracking and modifying the information associated with a network's Internet Protocol address space.

In the context of Cluster API (CAPI), an IPAM provider is responsible for managing IP address allocation for clusters, conflict avoidance.

```yaml
apiVersion: ipam.example.com/v1
kind: IPAMProvider
metadata:
  name: gcp-ipam-provider
  namespace: kube-system
spec:
  projectID: your-gcp-project-id
  region: us-central1
  network: your-gcp-network
  subnetwork: your-gcp-subnetwork
  ipRange: 10.0.0.0/16
```

API Adopters and Addons

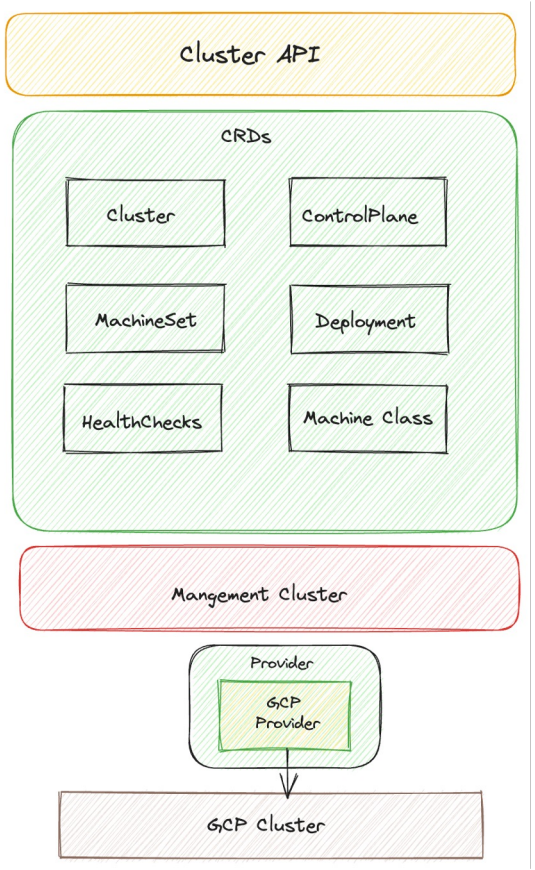
05

API Adopters are implementations managed by 3rd-parties according to the standard CAPI. Eg., Gardener Machine Controller manager, Kubermatic machine controller.

Addons are used to manage the lifecycle of workloads on the cluster after the initial provisioning. Eg., Helm

Cluster API Provider GCP (CAPG)

Enables to manage both GKE and Compute Engine-based Kubernetes clusters using the same API, providing a consistent experience across different types of GCP-based Kubernetes clusters.

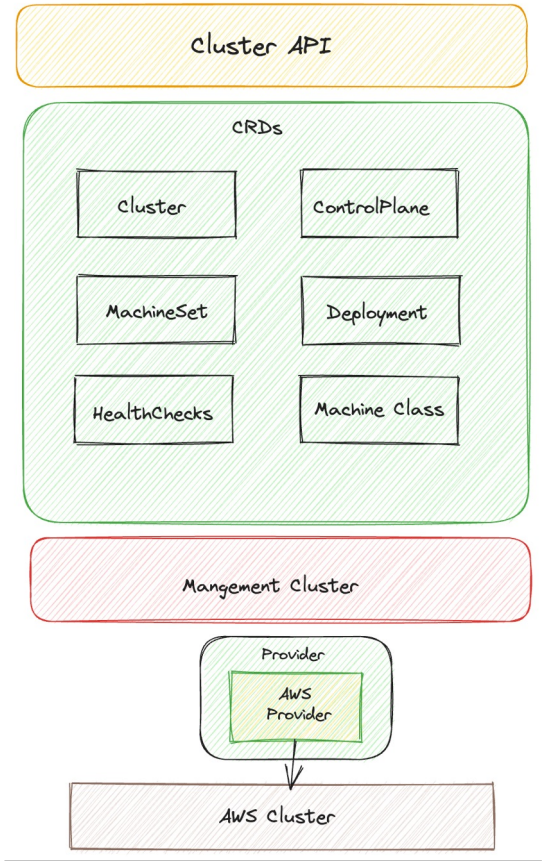


Cluster API Provider AWS (CAPA)

CAPA offers a Bastion Host feature, enabling secure SSH access to a private network from an external network.

Enabling this feature allows for debugging and maintenance purposes, providing a unique key to the cluster.

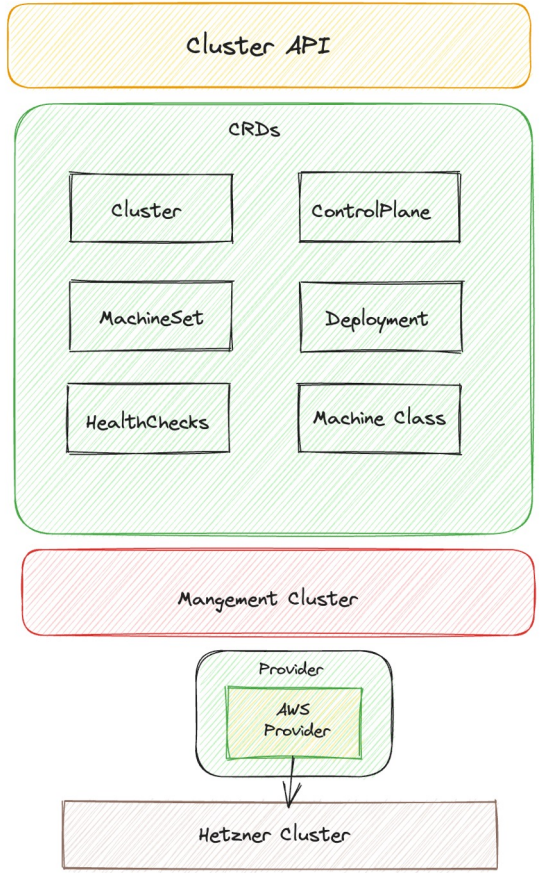
It's like having a special key to your castle!



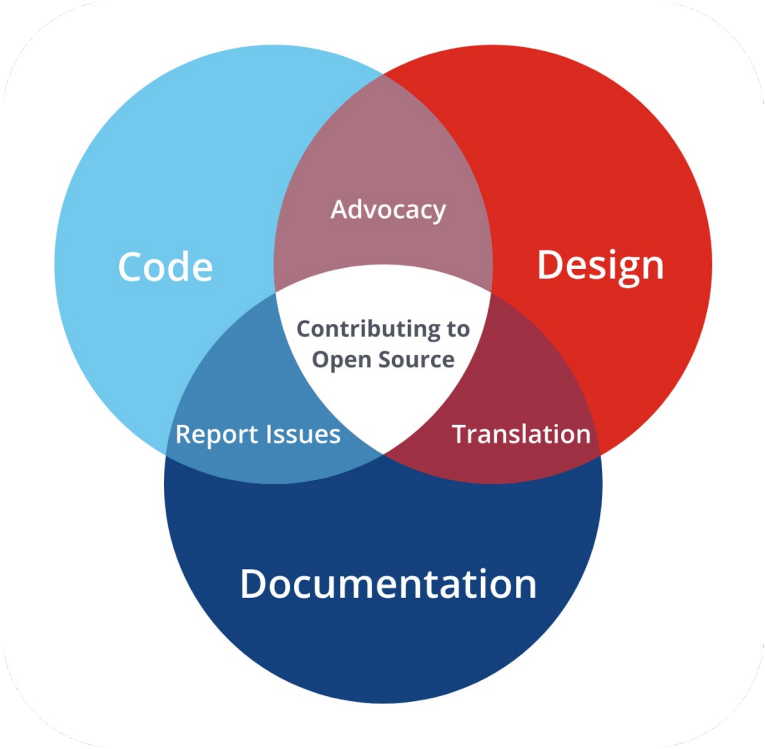
Cluster API Provider Hetzner (CAPH)

(CAPH) is not only used to manage Kubernetes clusters on Hetzner but it also plays a key role in the testing of Kubernetes itself.

critical component in maintaining the reliability of Kubernetes on Hetzner bare metal infrastructure.



Contribute Please!



Resources

Kubernetes Slack

http://slack.k8s.io/

Github

https://github.com/kubernetes-sigs/cluster-api

CAPI Cookbook

https://cluster-api.sigs.k8s.io/


That's all Folks!
Any Question ?

Thank you.