Experiences VLAIO TETRA OpenCloudEdge project

TETRA OpenCloudEdge team
Vrije Universiteit Brussel

Steffen Thielemans, Luca Gattobigio, Priscilla Benedetti

steffen.thielemans@vub.be, luca.gattobigio@vub.be, priscilla.benedetti@vub.be

VRIJE UNIVERSITEIT BRUSSEL

UNIVERSITÀ DEGLI STUDI DI PERUGIA

VLAIO TETRA OpenCloudEdge partners

Itecon

MD Management

TETRA OpenCloudEdge project goals & results

Investigate open source cloud ease of use, deployment & maintenance

Consume public cloud: @

On-premise cloud: &

Primary workload: JupyterHub SaaS for educational purposes

TETRA OpenCloudEdge project goals & results

Extend the cloud to the edge

kubernetes orchestrating Raspberry Pi IoT edge devices

Serverless computing with OpenFaaS Fuction-as-a-Service

Multi / hybrid cloud interaction

Cloud-agnostic Infrastructure as Code (IaC) via HashiCorp Terraform

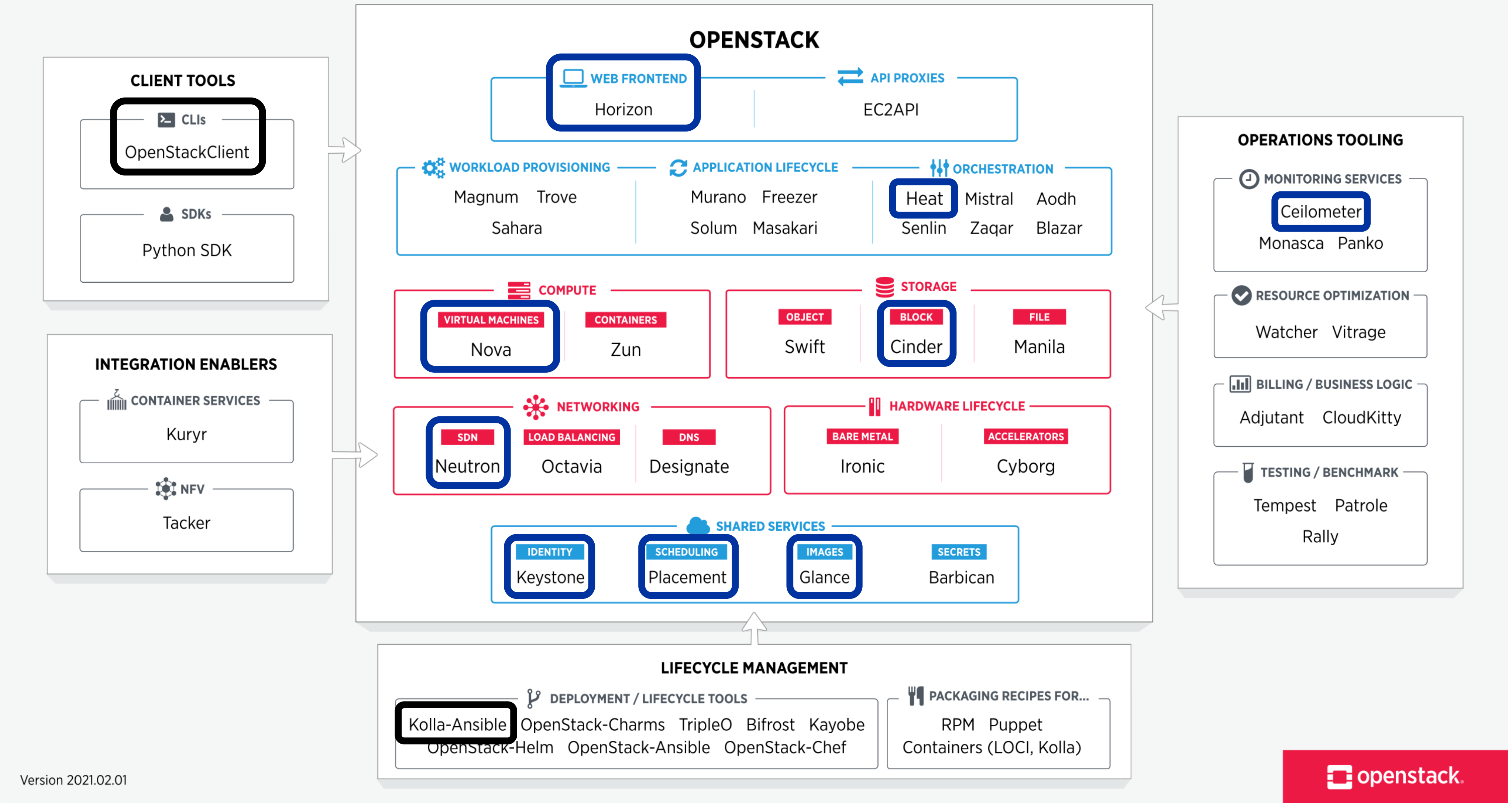
Inter-cloud communications via HashiCorp Consul

What is OpenStack?

openstack

- Open source project for cloud computing infrastructure
- First release in 2010
  - Initiative by NASA & rackspace the open cloud company
- Semi-annual release schedule
- A vast collection of different service components
  - **Compute**, **storage**, **networking**, authentication, orchestration, metering, etc.

OpenStack service components



Source: https://www.openstack.org/software/

On-premise OpenStack cloud setup

OpenStack using **Kolla-Ansible** deployment

- OpenStack **Kolla** project
  - OpenStack components in **containers**
  - Straightforward distribution, deployment & <u>versioning</u>

- **Ansible** = Open source automated deployment tool
  - Kolla-Ansible features the necessary playbooks

On-premise OpenStack cloud setup

- OpenStack service components operate on top of a **host OS**
- Kolla-Ansible supports ***CentOS***, *RHEL*, ***Ubuntu***, *Debian*
- Original choice CentOS **8**
- Migration to Ubuntu Server **20.04 LTS**
- On-premise OpenStack & other cloud deployments migrated

Upgrading on-premise OpenStack

OPENSTACK ROCKY → OPENSTACK XENA

Experiment: OpenStack *Rocky* (Aug 2018) → *Xena* (Oct 2021)

- **Kolla-Ansible** provides **version-by-version** upgrades
  - Include database changes, API changes, etc.

- Intermediate host OS & dependency upgrades

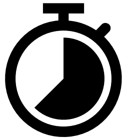
  → Consult upgrade notes & support matrix

OPENSTACK ROCKY → OPENSTACK STEIN → OPENSTACK TRAIN → OPENSTACK USSURI → OPENSTACK VICTORIA → OPENSTACK WALLABY → OPENSTACK XENA

OpenStack scheduled maintenance

Downtime upgrade OpenStack components

→ ≈ 1 min per service (Horizon, Keystone, Neutron, etc.)



→ Certain version upgrades require Docker / host OS upgrade
Extended maintenance: Upgrade nodes on individual basis

Cloud instances remain operational during upgrades ✓

- *Nova*'s back-end (*libvirt/KVM/QEMU*) not directly affected by the upgrades

Network connectivity briefly interrupted

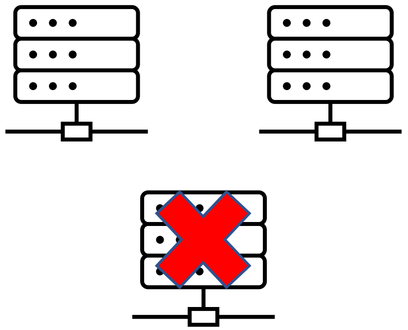
- *Neutron* & *OpenVSwitch* Kolla services upgraded

OpenStack (un)scheduled outages

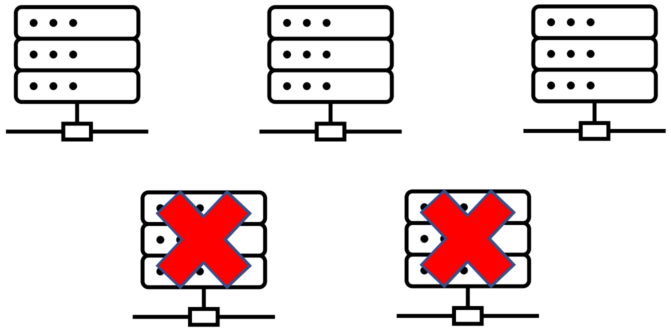
High-availability OpenStack APIs via *Haproxy & KeepAlived*

Recommendation distributed decision making **quorum**: uneven # control nodes

→ Over half of the nodes available and agree with decisions



3 nodes → can lose 1 node

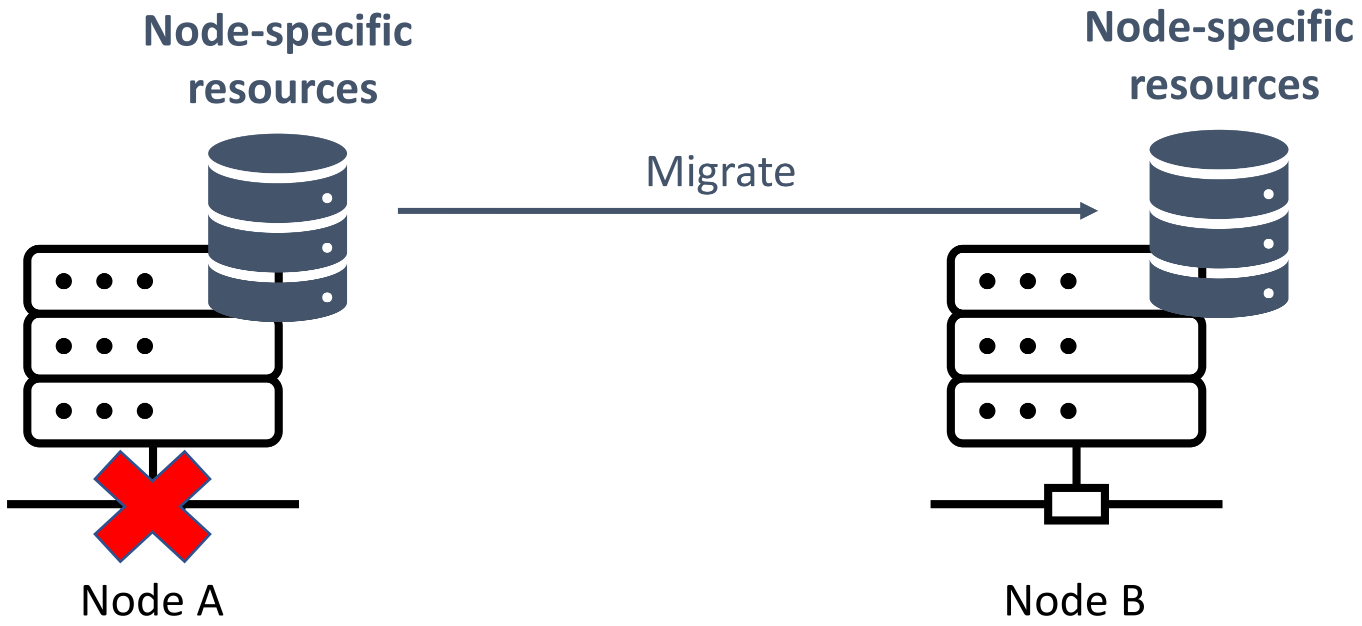


5 nodes → can lose 2 nodes

OpenStack (un)scheduled outages

Node-specific resources can become unavailable

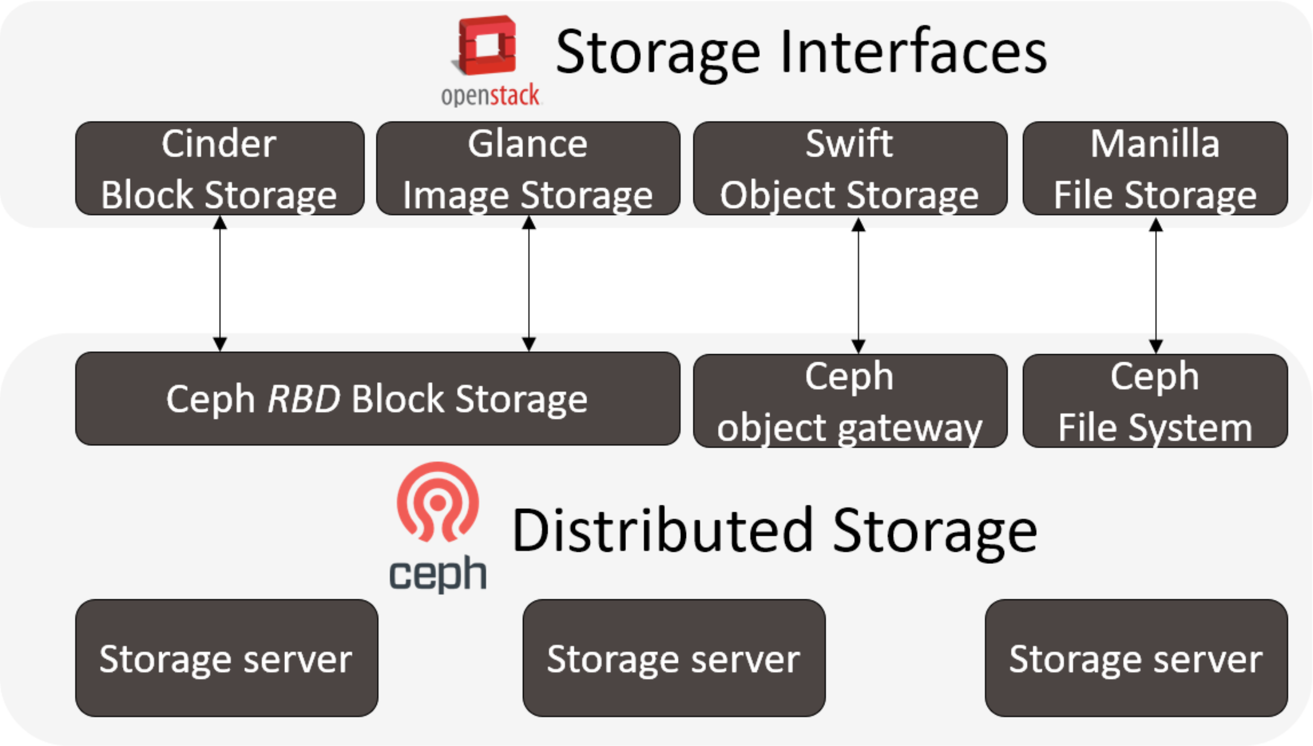
- Cloud instances, Network resources (Neutron routers & DHCP), Storage
- → Scheduled outage: first **migrate resources** to different nodes
- → Unscheduled outage: downtime or redeployment required



Ceph on-premise distributed storage

Network storage essential in cloud infrastructure

Ceph: open source distributed **block** / **object** / **file** storage



Ceph on-premise distributed storage

Initial budget-friendly Ceph solution

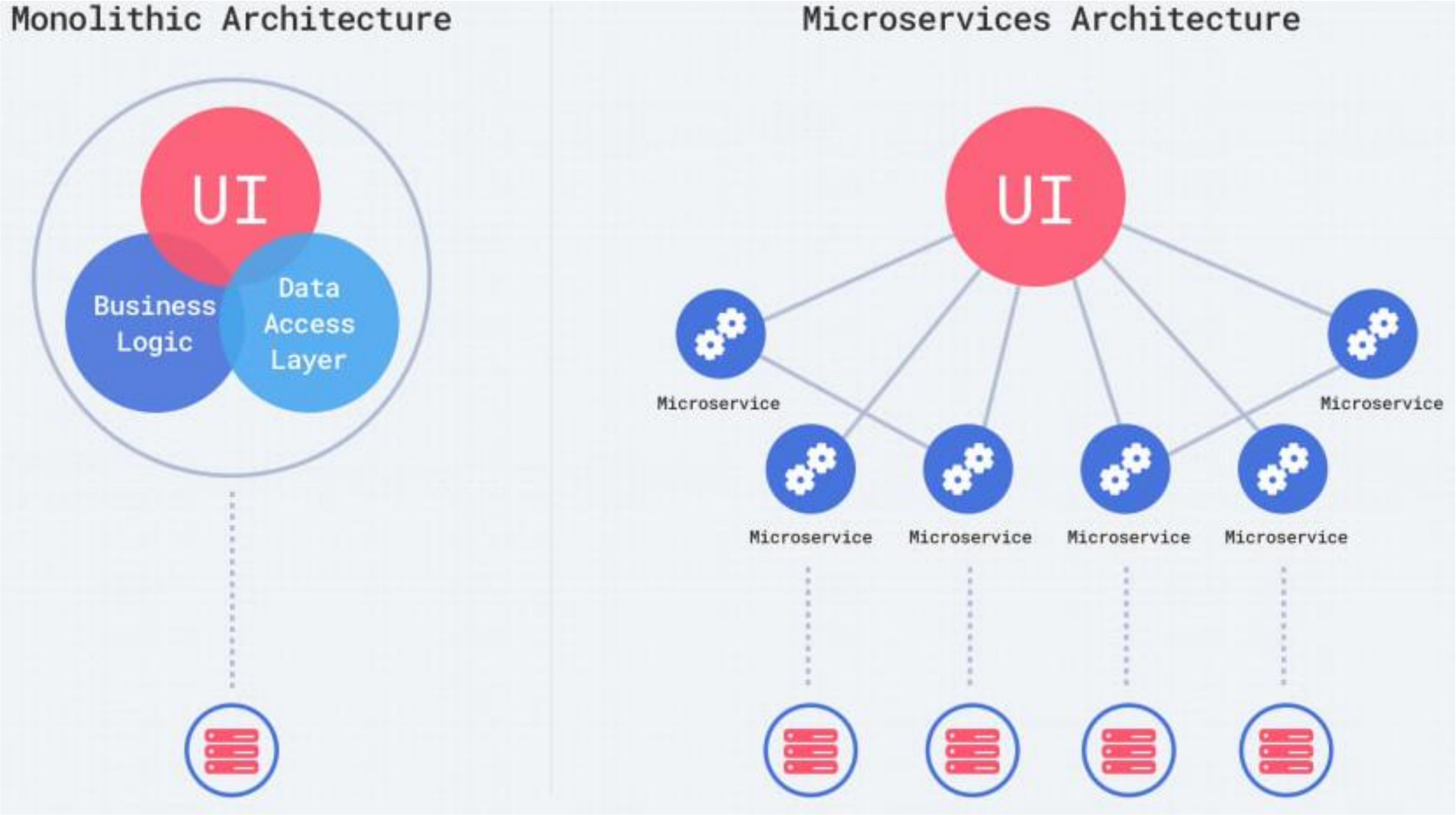
- Consumer grade 2,5” HDDs & dual 1 Gbps networks

→ **PERFORMANCE ISSUES**

<u>Ceph distributed storage recommendations:</u>

- Enterprise grade SSDs *(or HDDs)*
- 10 Gbps+ network(s) → <u>Distributed</u> storage

Microservices: Scalable cloud-agnostic apps



Source: Alex Barashkov, Microservices vs. Monolith Architecture

Cloud computing with Kubernetes

Market leading **container orchestrator** platform

Open source, first release in 2014 as Google project

Provides:

- Automated rollouts & self-healing
- Load balancing & horizontal scaling
- …

Cloud-agnostic *

Self-hosted (in cloud VMs) or as PaaS managed by cloud providers
→ Avoid vendor-specific tools and extensions

On-premise Kubernetes setup

Production environment deployed natively @ on-premise servers

Test environments deployed @ OpenStack

High-availability Kubernetes cluster via KEEPALIVED

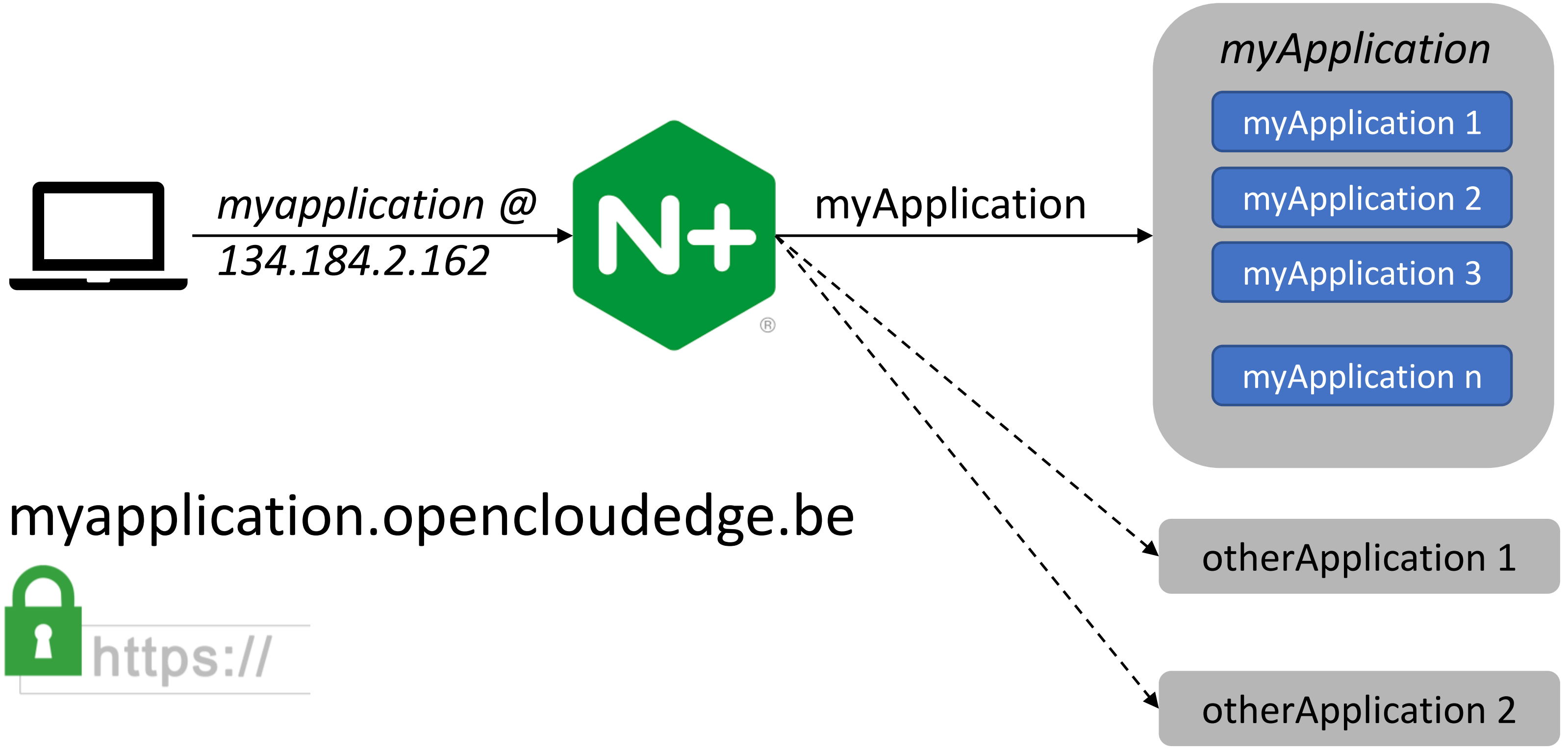
PROJECT CALICO as IPv4 & IPv6 Container Network Interface (CNI)

On-premise cached & internal Docker **image repositories**

On-premise Kubernetes setup

***Nginx* ingress** controller with **ExternalDNS** and **cert-manager**

Multiple apps on shared IP →



Automated DNS entries → myapplication.opencloudedge.be

Certificate management → https://

JupyterHub as cloud-based workload

JupyterHub SaaS for educational purposes

- Web-based **individual** *Jupyter Notebook* programming environments
  → Sandboxed multi-user compute & storage via Kubernetes

- Teaching & evaluation of various ir. / ing. programming courses
  → Sporadically exceeds 150 simultaneous sessions

Kubernetes for edge nodes

Centralized management of edge devices via Kubernetes API

Orchestration & **self-healing** of containerized applications

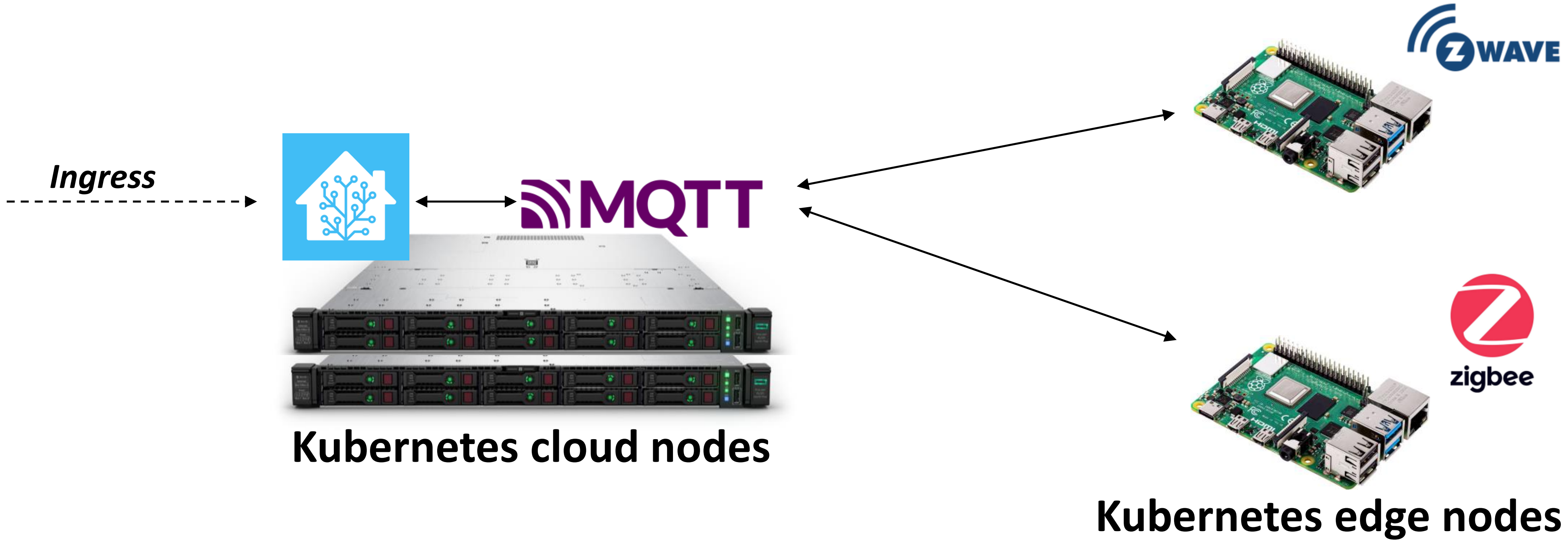
Managed **Network connectivity** & service discovery via integrated DNS

Persistent network storage

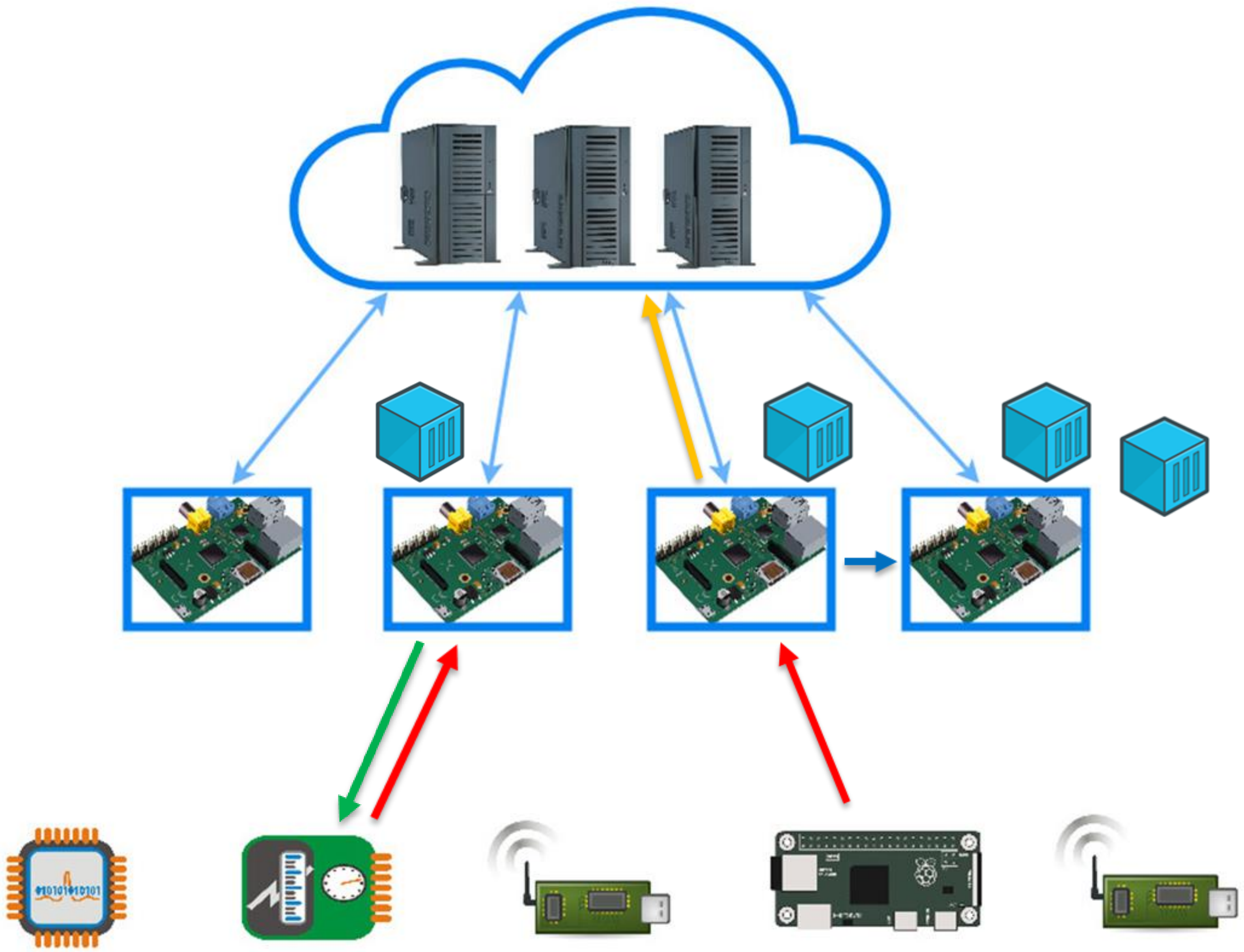
Kubernetes edge-based workload

Raspberry Pi 4 devices as ***edge* nodes** for IoT/building automation

→ Edge nodes *tainted* to only schedule specific workloads

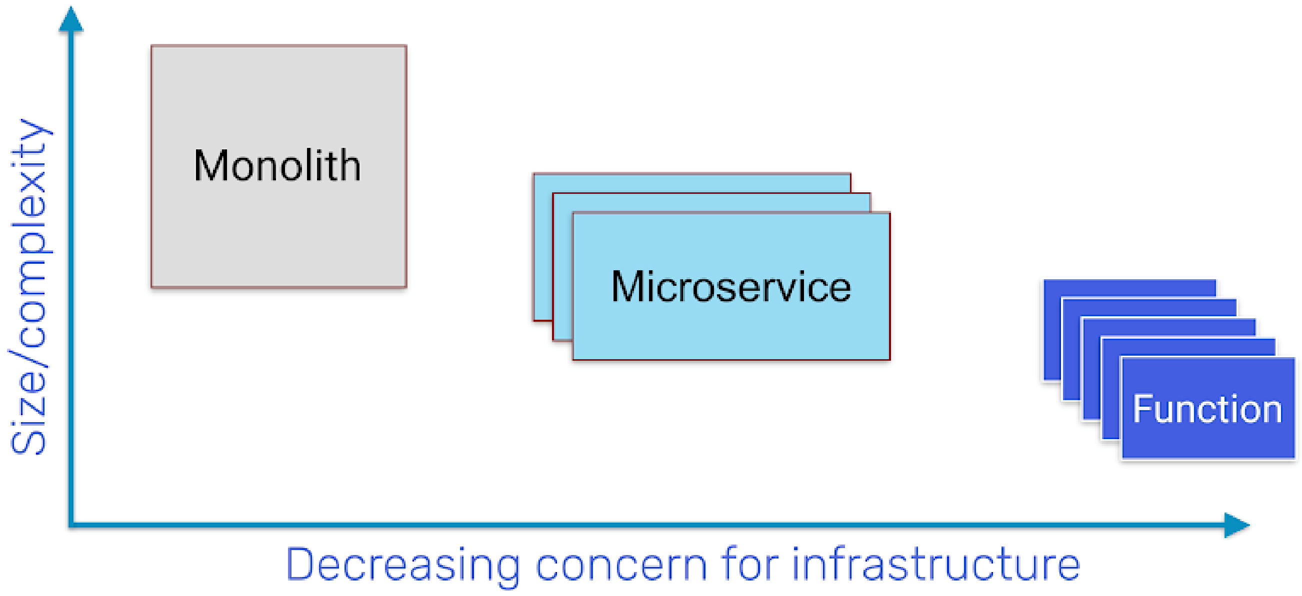


Use Serverless Computing



Functions-as-a-Service (FaaS)

- Focus on code
- Modular
- Event driven / REST endpoints
- Stateless
- Isomorphic
- Automatic scalability



From LinuxFoundationX: LFS157x,Introduction to Serverless on Kubernetes

Serverless Benefits vs Drawbacks

Benefits:

- **Automated scaling**
- **Smart resources usage**
- Strongly **reduced** infrastructure maintenance and costs
- **Simplified development**

Drawbacks:

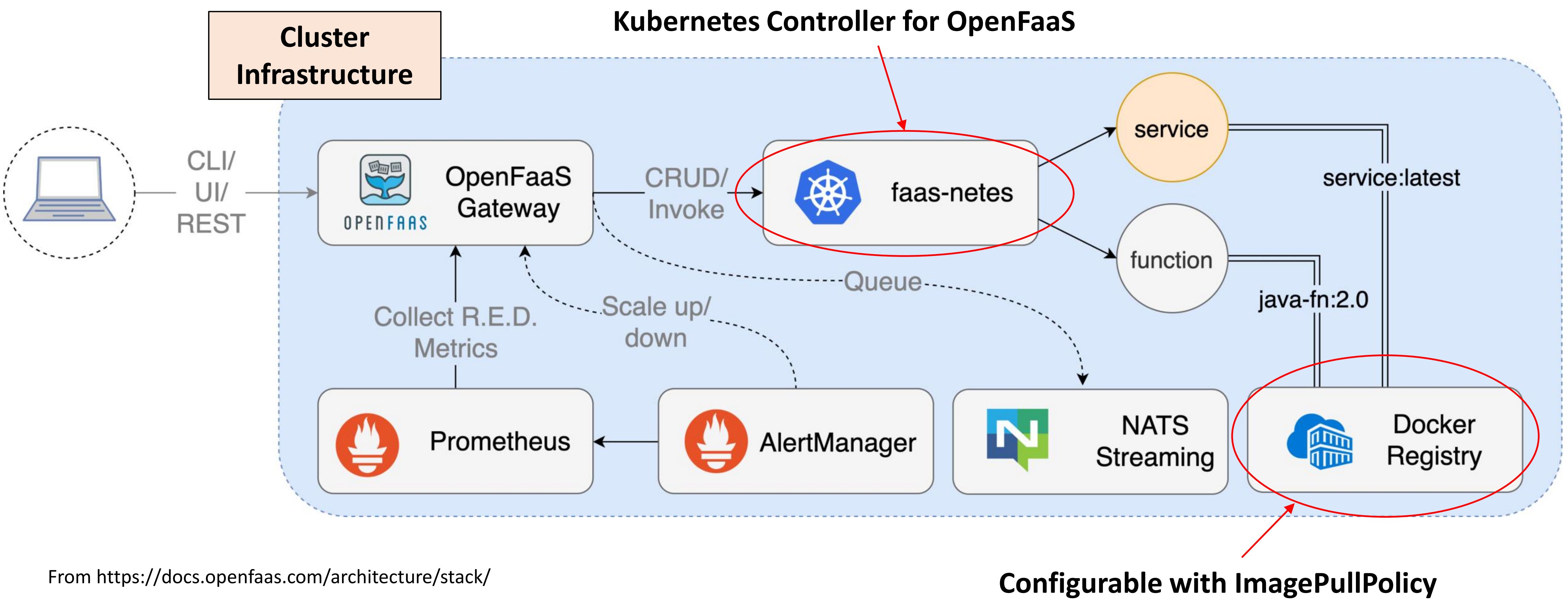
- **Lack of standardization** and ecosystem maturity
- Occasionally, more functions needed
- **Cold start latency issues**

General Scope

Serverless approach is particularly suitable for workloads:

- Asynchronous, concurrent and easy to parallelize into units of work
- Stateless, ephemeral and/or not too latency sensitive (cold start issues)
- With unpredictable variance in scaling requirements
- Highly dynamic in terms of changing business requirements

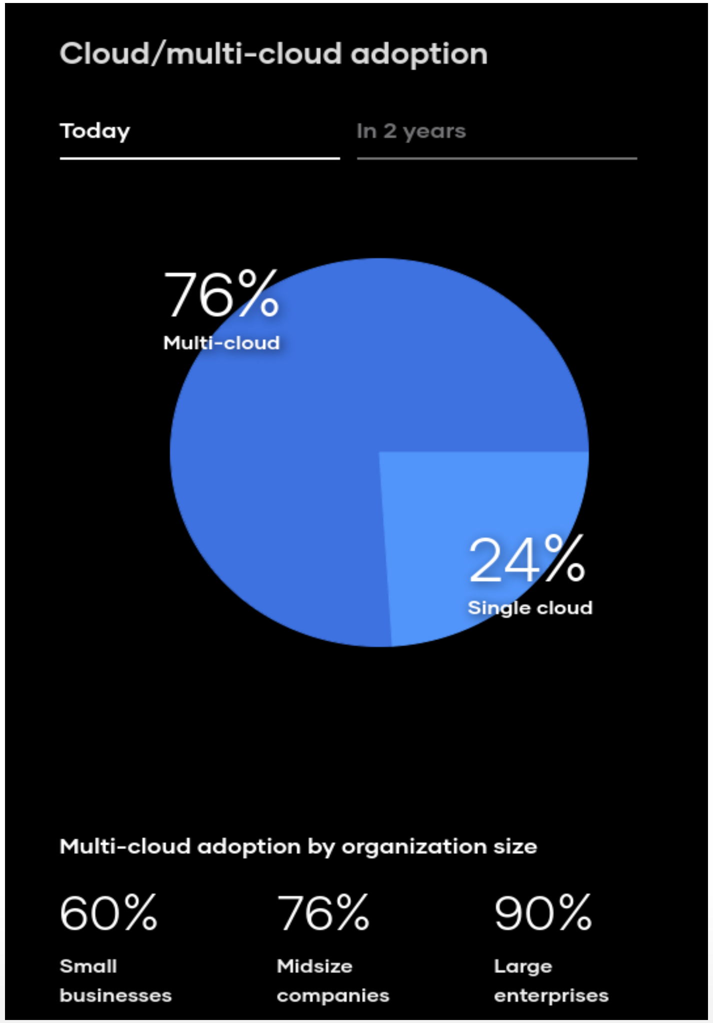
OpenFaaS



From https://docs.openfaas.com/architecture/stack/

MULTI-CLOUD ADOPTION

2021 Reports



https://www.hashicorp.com/state-of-the-cloud

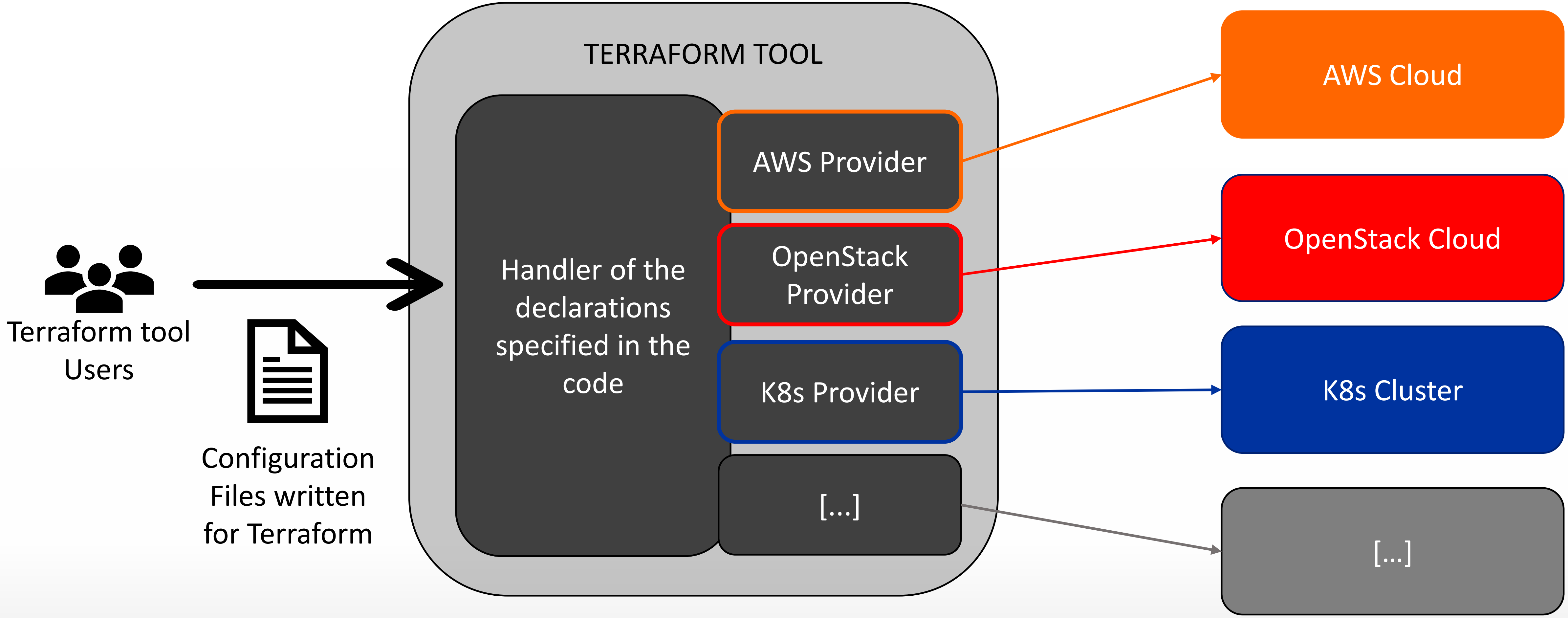
TERRAFORM

Multi-cloud environment orchestration

- Write, Plan, and Create Infrastructure-as-Code
- Code can be (re)used to create a given infrastructure on any (cloud) platform
- A single tool to manage any resource, regardless of its location

TERRAFORM

From declaration to deployment

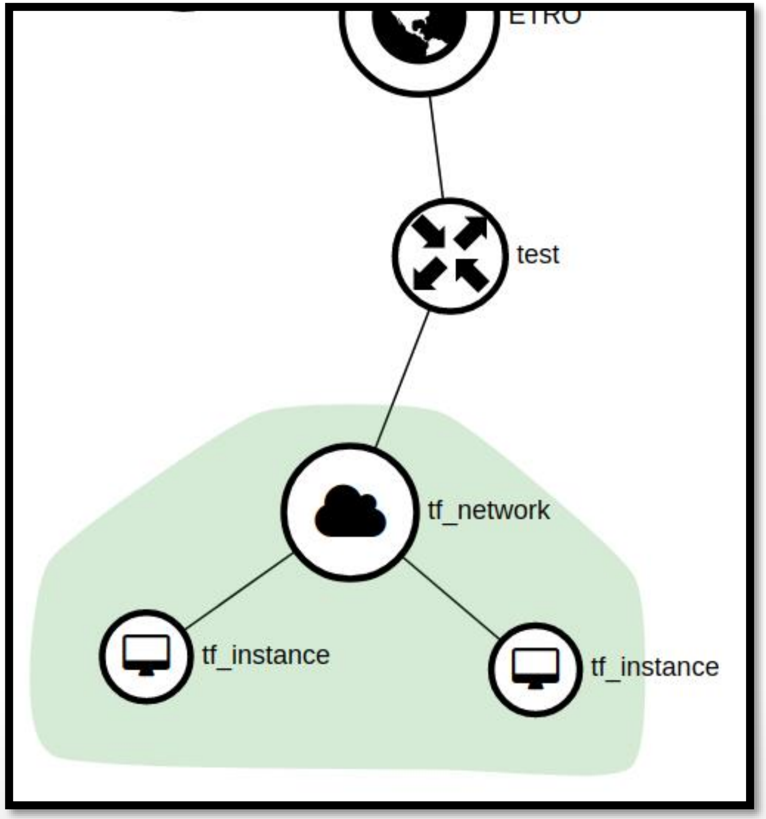


Infrastructure-as-Code (IaC)

```
provider "openstack" {}

resource "openstack_compute_instance_v2" "tf_instance" {
  count = 2
  name = "tf_instance"
  image_name = "cirros"
  flavor_name = "m1.tiny"
  network {
    name = openstack_networking_network_v2.tf_network.name
  }
}

resource "openstack_networking_network_v2" "tf_network" {
  name = "tf_network"
}

resource "openstack_networking_subnet_v2" "tf_subnet" {
  name = "tf_subnet"
  network_id = openstack_networking_network_v2.tf_network.id
  cidr = "192.168.150.0/24"
}

resource "openstack_networking_router_v2" "router" {
  name = "test"
  admin_state_up = true
  external_network_id = "0d91136f-b550-4c00-bf65-9542b8e3bb1d"
}
```

Configuration file example

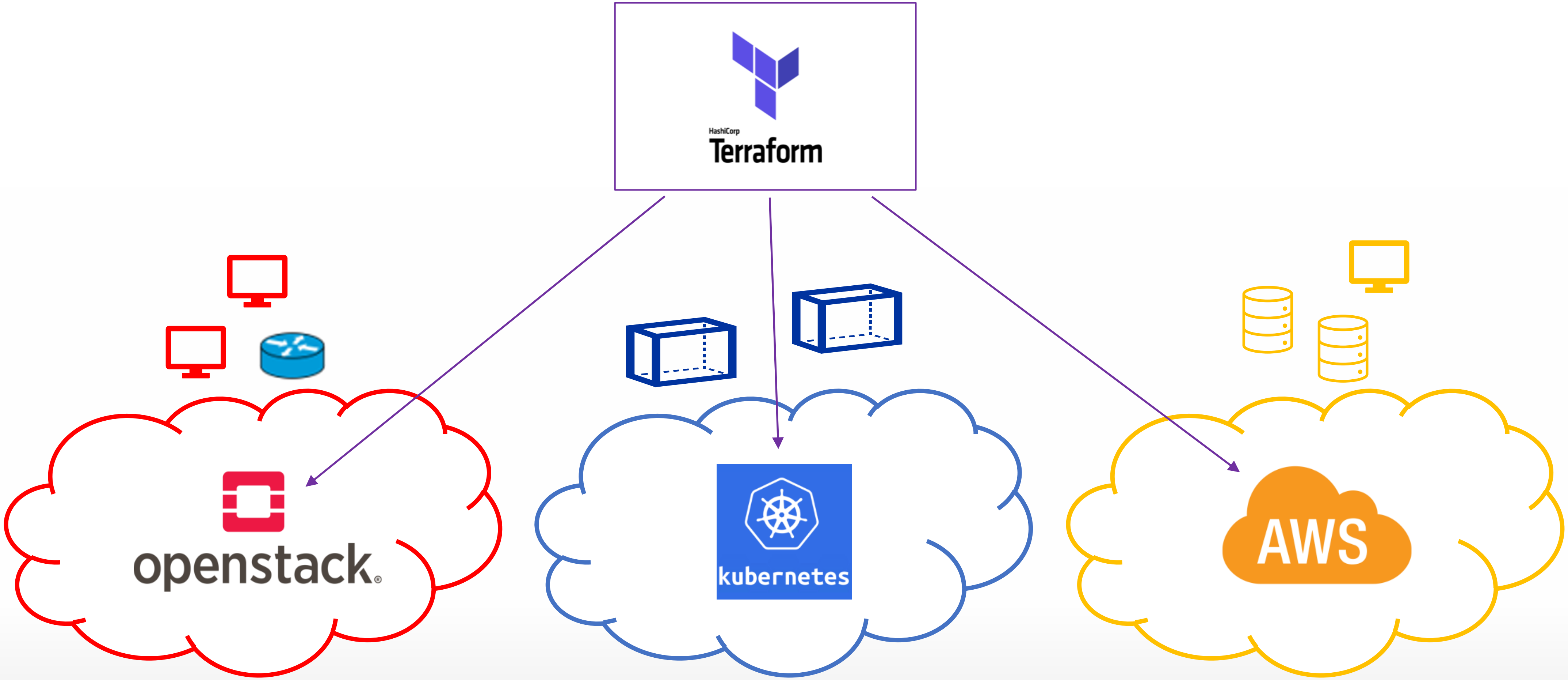
Instance Name	Image Name	IP Address	Flavor	Key Pair	Status
tf_instance	cirros	192.168.150.178	m1.tiny	-	Active
tf_instance	cirros	192.168.150.55	m1.tiny	-	Active



Infrastructure example

MULTI-CLOUD ORCHESTRATION VIA TERRAFORM

Multi environment deployment



TERRAFORM

OpenStack and AWS code comparison

```
provider "openstack" {}

resource "openstack_compute_instance_v2" "instance" {
  name = var.instance_name
  image_name = var.image_name
  flavor_name = var.flavor_name
  network { name = openstack_networking_network_v2.net.name }
}

resource "openstack_networking_network_v2" "net" {
  name = var.network_name
}

resource "openstack_networking_subnet_v2" "subnet" {
  name = var.subnet_name
  network_id = openstack_networking_network_v2.net.id
  cidr = var.cidr_value
}

resource "openstack_blockstorage_volume_v2" "vol" {
  name = var.volume_name
  size = var.volume_size
}
```

OpenStack

```
provider "aws" {}

resource "aws_instance" "instance" {
  tags = { Name = var.instance_name }
  ami = var.image_name
  instance_type = var.flavor_name
  subnet_id = aws_subent.subnet.id
}

resource "aws_vpc" "net" {
  name = var.network_name
}

resource "aws_subnet" "subnet" {
  name = var.subnet_name
  network_id = openstack_networking_network_v2.net.id
  cidr = var.cidr_value
}

resource "aws_ebs_volume" "vol" {
  name = var.volume_name
  size = var.volume_size
}
```

AWS

TERRAFORM WORKSHOP

Basics and advanced topics

- **INIT**
  Initialize Terraform and look for providers

- **PLAN**
  Specifies what to execute according to the configuration files

- **APPLY**
  Perform the execution

- **DESTROY**
  Destroy all the resources

```
  Enter a value: yes

openstack_networking_secgroup_v2.secgroup: Creating...
openstack_networking_floatingip_v2.ws_floating_ip: Creating...
openstack_networking_network_v2.workshop_network: Creating...
openstack_networking_router_v2.ws_router: Creating...
openstack_networking_secgroup_v2.secgroup: Creation complete after 0s [id=4c0d3741-1e58-4a4a-9e92-d2450ec2d1f9]
openstack_networking_secgroup_rule_v2.secgroup_rule: Creating...
openstack_networking_secgroup_rule_v2.secgroup_rule: Creation complete after 0s [id=401cec09-360d-477a-aac1-b948b4d910d3]
openstack_networking_network_v2.workshop_network: Creation complete after 5s [id=c1fb2002-a930-48b3-9476-08ac18708f06]
openstack_networking_subnet_v2.workshop_subnet: Creating...
openstack_compute_instance_v2.workshop_instance: Creating...
openstack_networking_floatingip_v2.ws_floating_ip: Creation complete after 6s [id=47ae9b31-423c-49ca-bc7d-b15f94e41e16]
openstack_networking_router_v2.ws_router: Creation complete after 6s [id=ed6c0809-6ac5-48a4-b07e-b1457811a628]
openstack_networking_subnet_v2.workshop_subnet: Creation complete after 5s [id=49522f7d-0815-43de-b219-74b3b0d6ab2a]
openstack_networking_router_interface_v2.router_interface: Creating...
openstack_compute_instance_v2.workshop_instance: Still creating... [10s elapsed]
openstack_networking_router_interface_v2.router_interface: Creation complete after 7s [id=a9044234-ac3a-4f49-a03c-bc5af5f1d840]
openstack_compute_instance_v2.workshop_instance: Creation complete after 12s [id=ff18f067-4ad9-4383-97e1-67a1a88e750e]
openstack_compute_floatingip_associate_v2.fip_associate: Creating...
openstack_compute_floatingip_associate_v2.fip_associate: Creation complete after 2s [id=10.20.29.149/ff18f067-4ad9-4383-97e1-67a1a88e750e/]

Apply complete! Resources: 9 added, 0 changed, 0 destroyed.

Outputs:

floating_ip = "10.20.29.149"
```

Welcome to nginx!

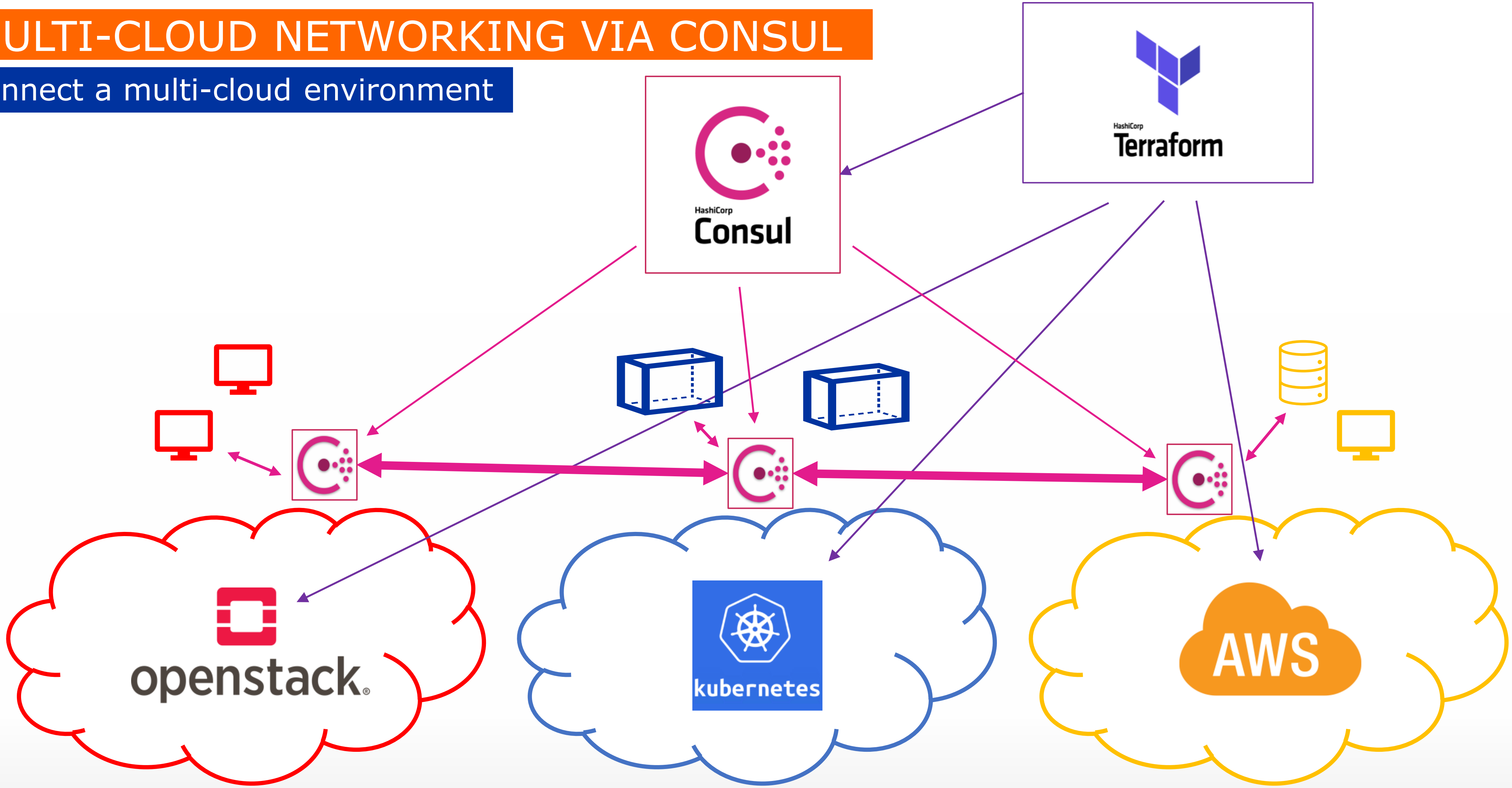
If you see this page, the nginx web server is successfully installed and working. Further configuration is required.

For online documentation and support please refer to nginx.org.
Commercial support is available at nginx.com.

Thank you for using nginx.

MULTI-CLOUD NETWORKING VIA CONSUL

Connect a multi-cloud environment



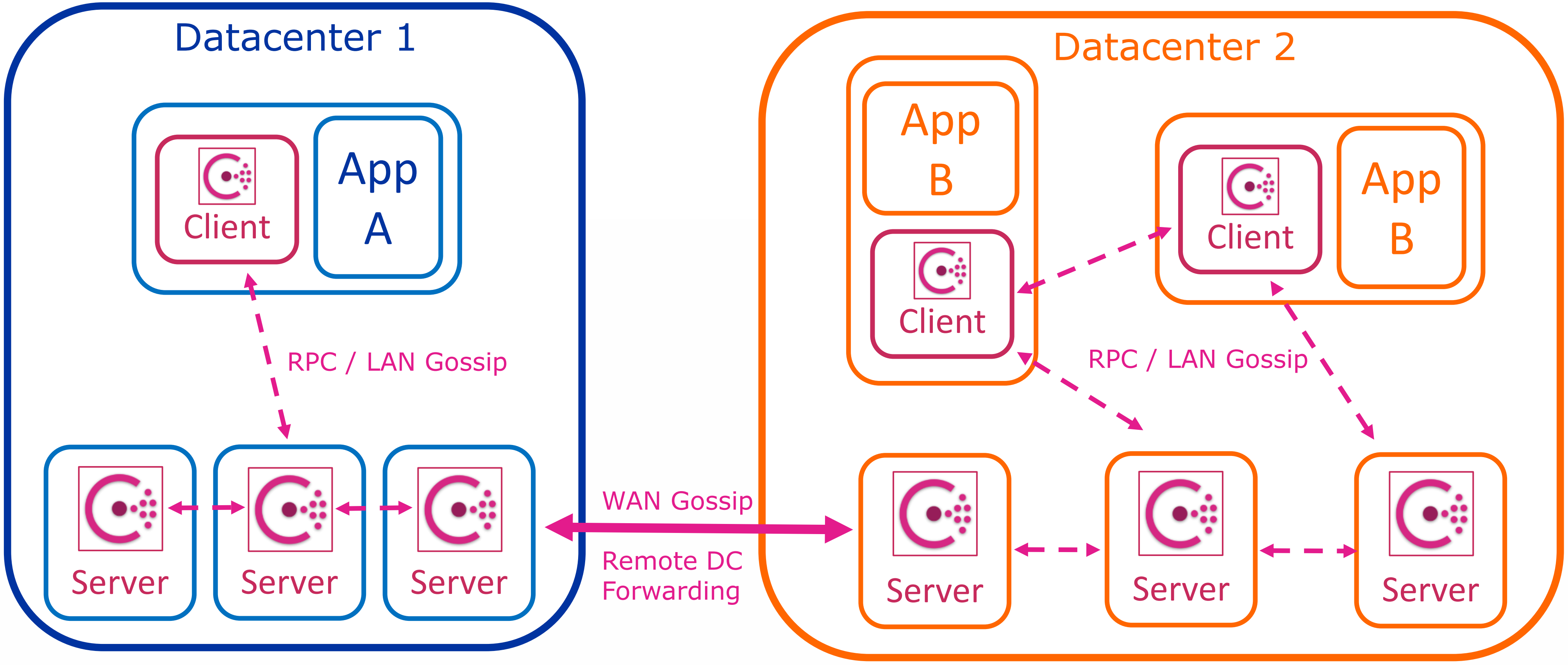
CONSUL

Multi-cloud service mesh

- Open-source tool for deploying a service mesh (Terraform integration)
- It offers centralized registry, service discovery, health checks, zero trust network, load-balancer, Key-Value store...
- Can be deployed in virtual machines and containers

MULTI-CLOUD NETWORKING VIA CONSUL

Architecture

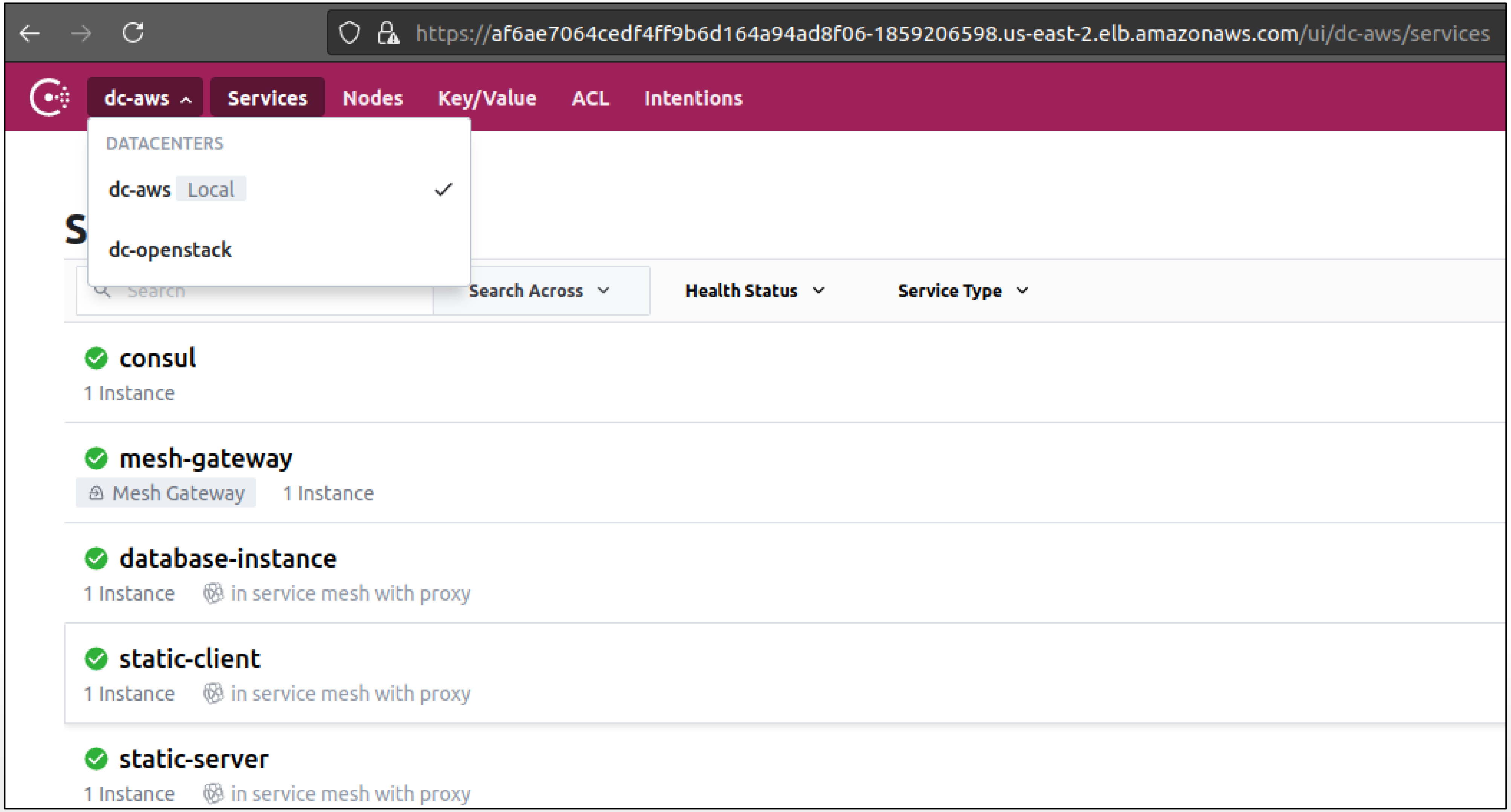


Name	Datacenter	IP	Status
App A	1	192.168.1.45	Alive
App B	2	10.0.0.54, 10.0.0.98	Alive, Alive

Name	Datacenter	IP	Status
App A	1	192.168.1.45	Alive
App B	2	10.0.0.54, 10.0.0.98	Alive, Alive

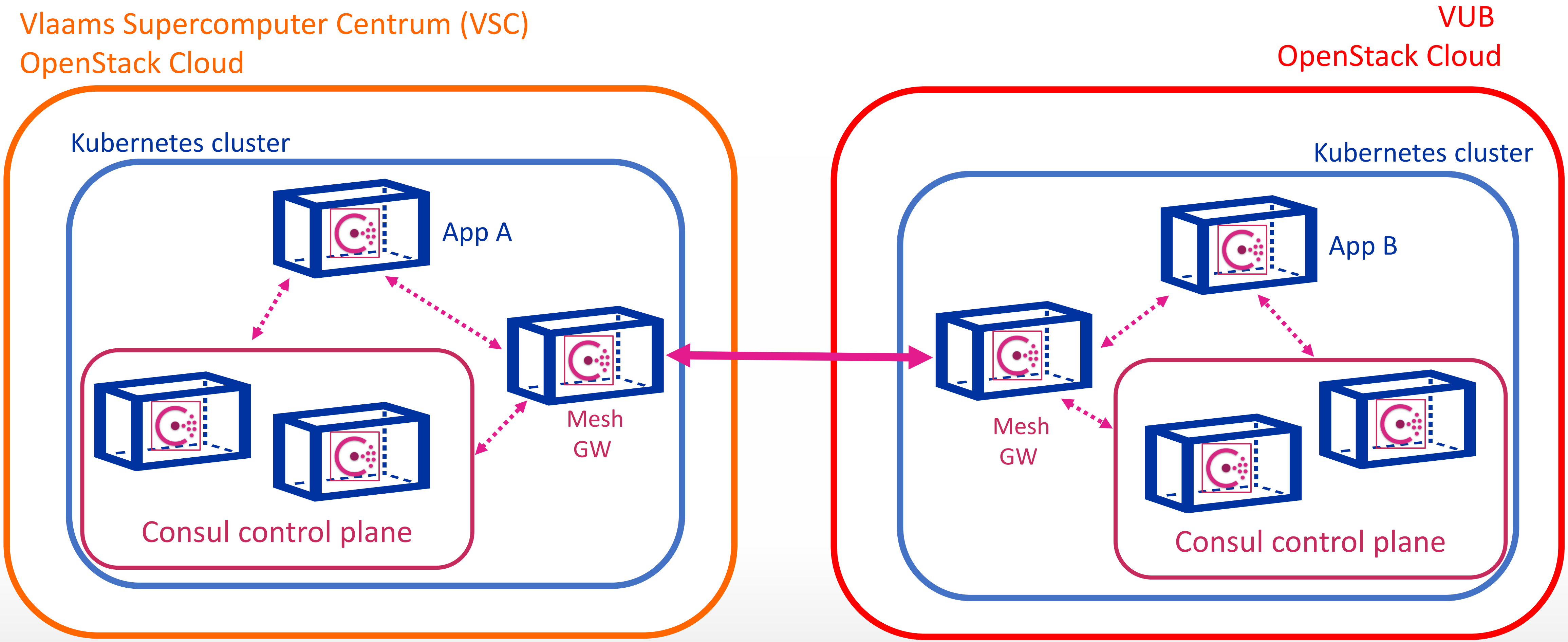
MULTI-CLOUD NETWORKING VIA CONSUL

Datacenter federation: Consul UI



MULTI-CLOUD CONNECTION

K8s cluster federation



WORKSHOPS AVAILABLE

Willingness to schedule additional workshops and share material

Contact us: **Steffen.Thielemans@vub.be**; **kris.steenhaut@vub.be**

Thank you! Questions?

Workshop

HashiCorp Terraform

- Software installation: Terraform CLI
- Hashicorp Configuration Language: usage and explanation
- Basic commands: init, plan, apply, destroy
- Infrastructure orchestration: deploy, change, delete
- Use of variables and functions: input, output, count
- Multiple providers: docker, OpenStack, authentications, providers comparison (example with AWS)
- Examples: modules, infrastructure import, deployment of a webserver

Workshop

- Set up & **consume** elementary OpenStack project
  - Tenant network, SSH key, Deploy cloud instances from image
  - Volume snapshots of modified cloud instances
  - Deploy additional instances from this volume snapshot

- Set up and **deploy** multi-node OpenStack infrastructure using **Kolla-Ansible**
  - *OpenStack-inside-OpenStack*: Deployed inside OpenStack VMs
  - Set up host machines, Ansible & Kolla-Ansible configuration
  - Deploy OpenStack with Kolla containers
  - Consume the OpenStack-inside-OpenStack

Workshop

- Brief introduction to **Docker** containers & docker-compose

- Deployment of multi-node **Kubernetes** environment
  - Simplified testing/development with **MicroK8s**

- Kubernetes interaction and deployments
  - *Kubectl* and Kubernetes Dashboard
  - *JupyterHub* from **Helm** package manager
  - Porting containerized webapp to autonomous **horizontally scalable** Kubernetes deployment