

A robotic hand with a black and blue gripper, labeled 'ROBOTIQ', is reaching towards a human hand with red nail polish. The background is a blurred, colorful bokeh of light circles. The text 'HUN REN' is displayed in white, bold, sans-serif font in the top right corner.

HUN  
REN



SZTAKI

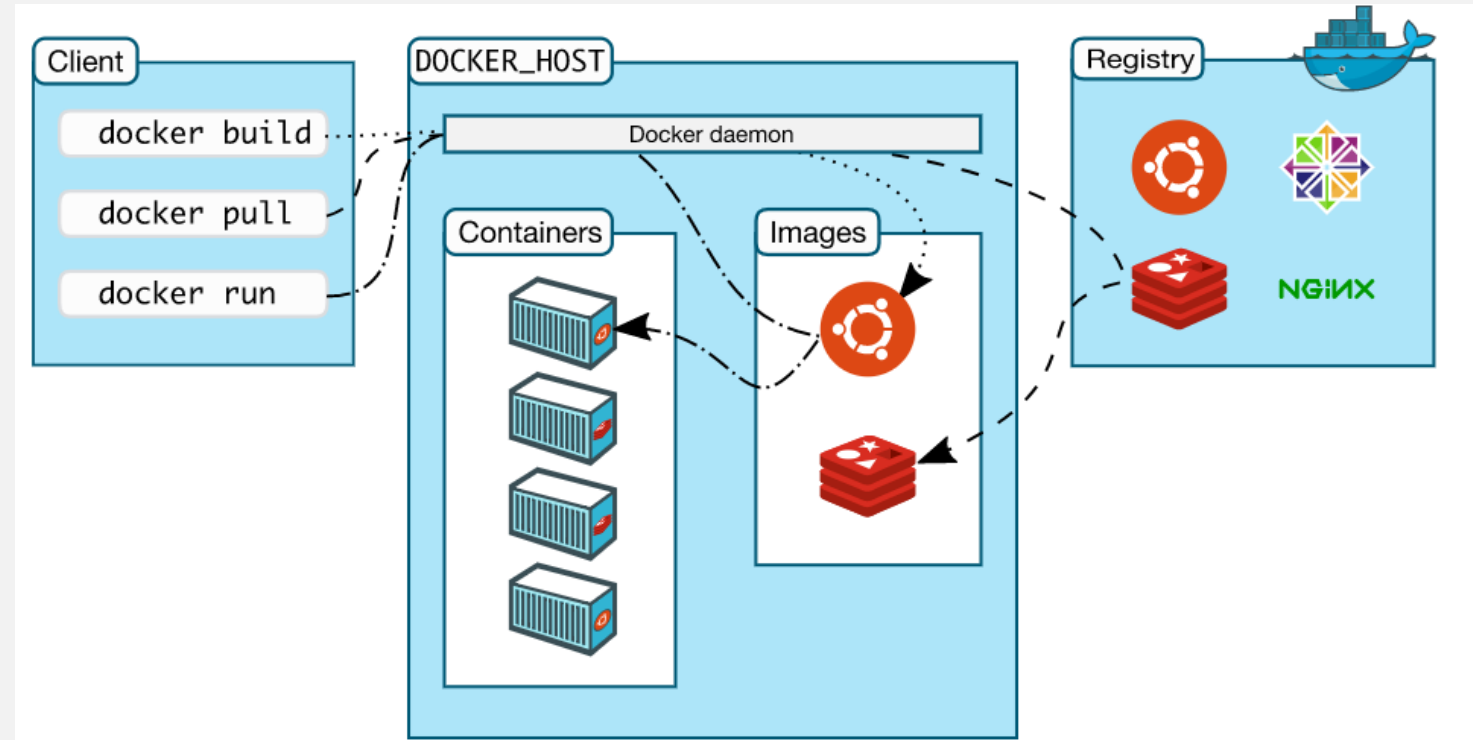
# Container orchestration with Kubernetes reference architecture

Farkas Attila  
HUN-REN SZTAKI LPDS, research associate

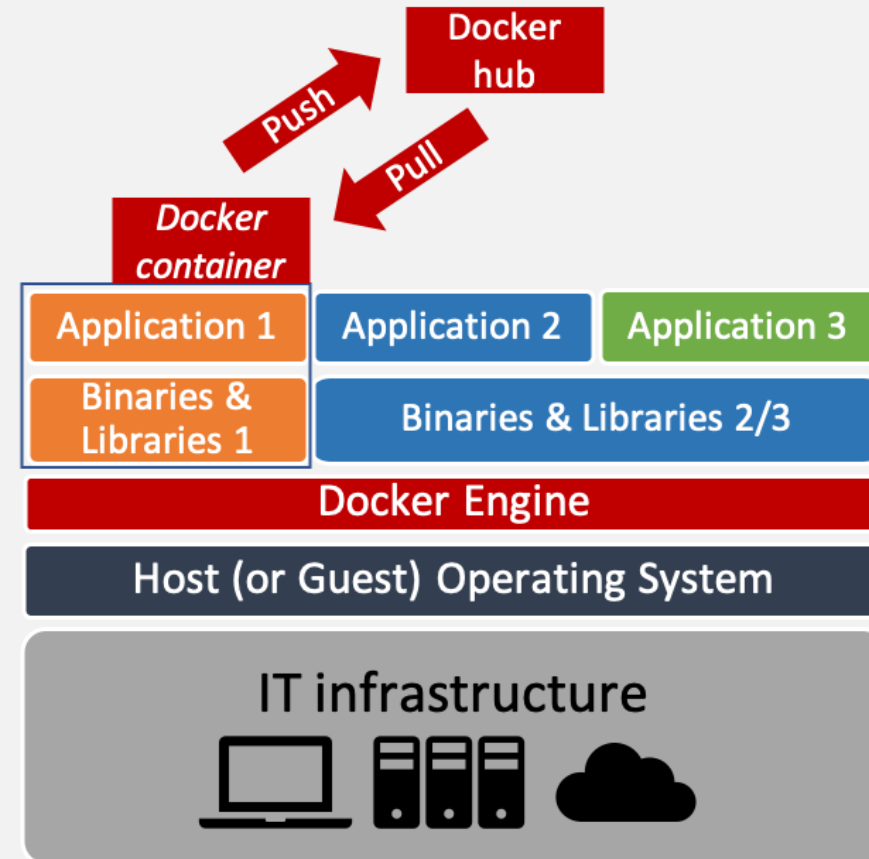
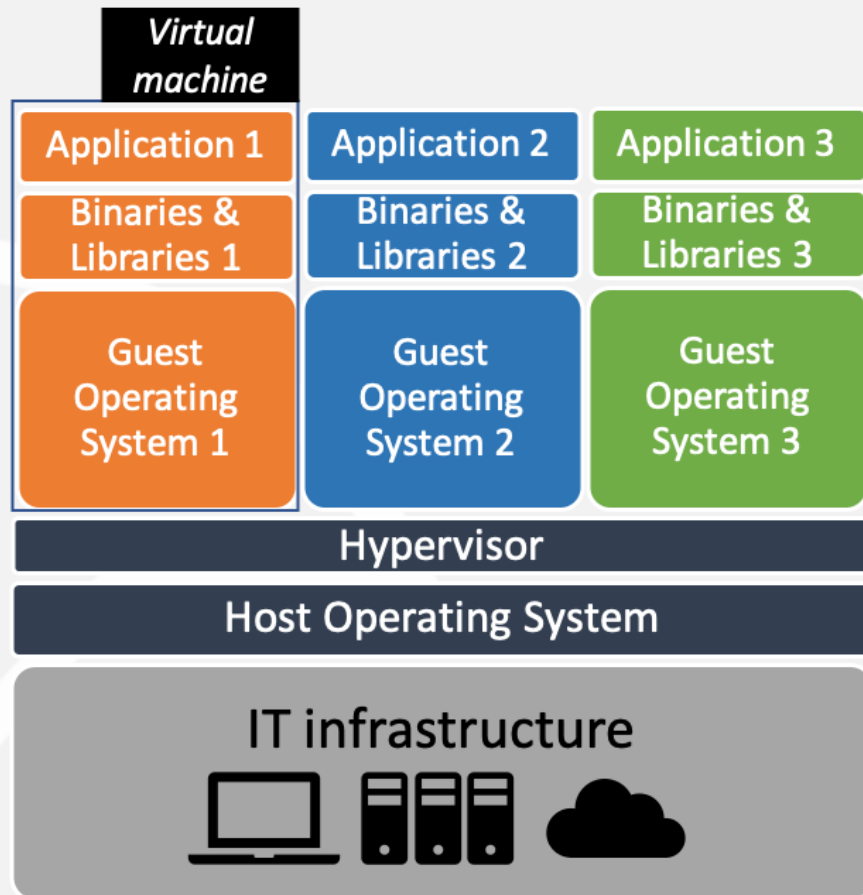
**HUN-REN**  
Hungarian Research Network

# Docker

- Open-source container platform
- Instead of virtualization
- Separated applications
- Based on Linux container technology
- Layered containers
- Centralized image repository
- Portability



# Virtual machine vs. Container

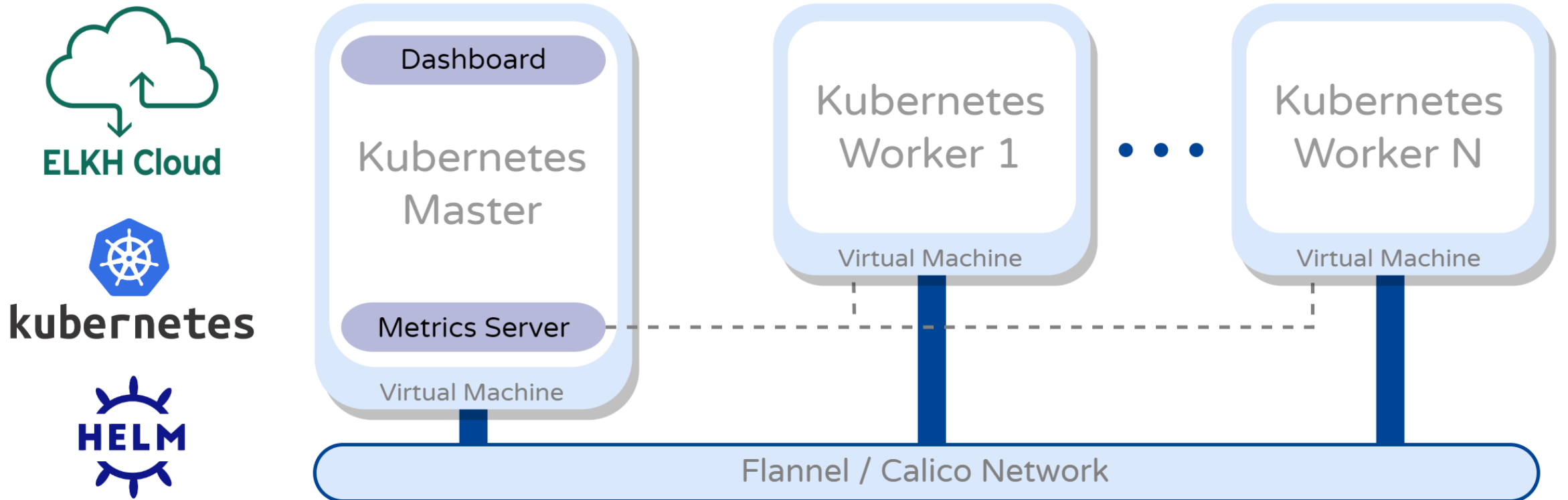


# Kubernetes

- Also known as K8s
- Open-source container orchestrator
- Automated rollouts and rollbacks
- Service discovery and load balancing
- Storage orchestration
- Self-healing
- Batch execution
- Horizontal scaling
- Designed for extensibility



# Kubernetes reference architecture



<https://science-cloud.hu/en/reference-architectures/kubernetes-cluster>

# Usage of reference architectures

## Necessary steps from the user:

**Step 0 - Preparation** (HUN-REN Cloud project, creation of empty VM if necessary)

**Step 1 - Terraform and Ansible installation**

**Step 2 - Download the descriptor files**

HUN-REN Cloud webpage

**Step 3 - Create firewall rules (optional)**

HUN-REN Cloud OpenStack dashboard or descriptor file

**Step 4 - Personalizing descriptor files**

**Step 5 - Terraform initialization**

```
$ terraform init
```

**Step 6 - Infrastructure deployment**

```
$ terraform apply (--auto-approve)
```

**Step 7 - Infrastructure usage**

**Step 8 - Infrastructure destroy**

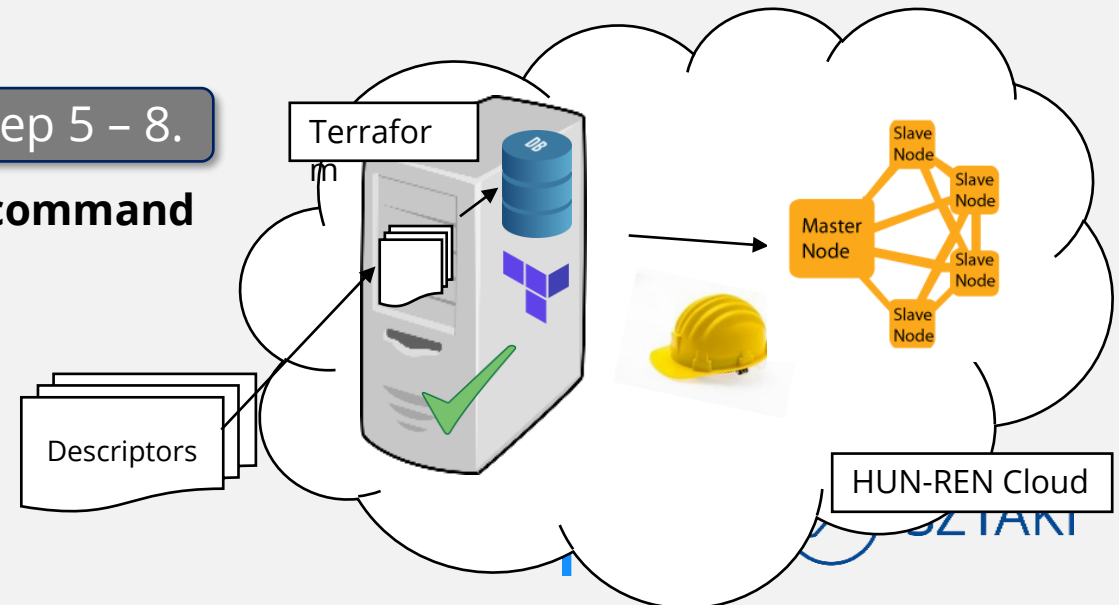
```
$ terraform destroy (--auto-approve)
```

Step 0 - 1. Only for the first time.

Step 2 - 4. Only 1 time per reference architecture

Step 5 - 8.

1 command



# Step 1 - Terraform and Ansible installation

- Terraform and Ansible installation

```
sudo apt-get update && sudo apt-get install -y gnupg software-properties-common curl
curl -fsSL https://apt.releases.hashicorp.com/gpg | sudo apt-key add -
sudo apt-add-repository "deb [arch=amd64] https://apt.releases.hashicorp.com $(lsb_release -cs) main,"
sudo apt-get update && sudo apt-get install terraform=1.3.6

sudo apt install python3-pip
sudo python3 -m pip install ansible==6.7.0
```

- (Optional) RefArch Toolset Docker image

```
docker run -it -v PATH_TO_WORKDIR:/home/refarch -v PATH_TO_PRIVATE_KEY:/root/.ssh/id_rsa:ro
git.sztaki.hu:5050/science-cloud/reference-architectures/refarch-toolset bash
```

# Step 2 - Download the descriptor files



HashiCorp

## Terraform

- **Provision infrastructure**
  - Virtual machines
  - Network settings
  - Firewall rules
  - Execute tasks
  - Invoke Ansible



## ANSIBLE

- **Configure nodes**
  - Install packages
  - Start services
  - Run Docker containers

- I. Set authentication information
- II. Set resource parameters
- III. Customize through variables
- IV. Deploy


```
horovod_master_node = ({  
  name = "horovod-master"  
  flavor_name = "SET_YOUR_FLAVOR_NAME"  
  image_id = "SET_YOUR_IMAGE_ID"  
  key_pair = "SET_YOUR_KEY_PAIR_NAME"  
  floating_ip = "SET_YOUR_  
  volume_size = 32  
})
```

```
user_config = ({  
  jupyter_password = "elkhcloud"  
  enable_gpu = true  
  monitoring = false  
  monitoring_password = "elkhcloud"  
  nvidia_driver_install = false  
  elkh_cloud_dedicated_network = false  
})
```



# Step 2 - Download the descriptor files

1




### MariaDB cluster

MariaDB is one of the most popular open-source relational database management systems (RDBMS), provided with multi-master replication capabilities by...

General Clusters Data oriented

→




### Slurm cluster

Slurm is an open source, fault-tolerant, and highly scalable cluster management and job scheduling system for large and small Linux clusters. Slurm...

General Clusters

→




### Kafka cluster

The Kafka reference architecture can be used to support IoT and BigData applications to create a Kafka cluster architecture that includes a Zookeeper...

Clusters Big data

→




### Horovod cluster

Horovod is a distributed deep learning framework for TensorFlow, Keras, Pytorch and Apache MXNet. It was originally developed by Uber, for the...

Artificial intelligence

→




### JupyterLab development environment

JupyterLab is a next-generation web-based user interface for Project Jupyter, an interactive development environment for handling Jupyter Notebooks...

Data oriented Big data Artificial intelligence

→

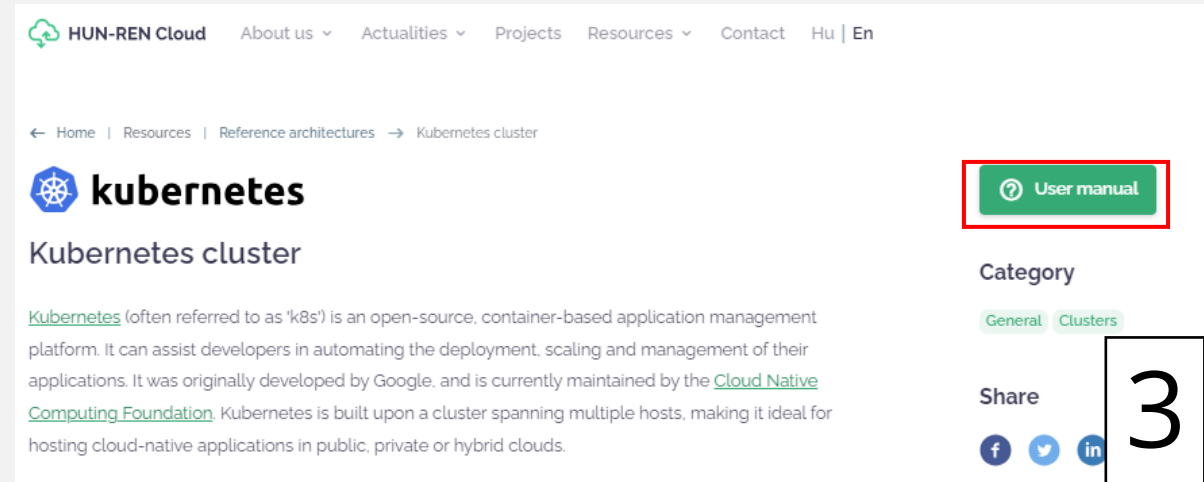


### Kubernetes cluster

Kubernetes (often referred to as 'k8s') is an open-source, container-based application management platform. It can assist developers in automating the...

General Clusters

→



HUN-REN Cloud About us Actualities Projects Resources Contact Hu | En

← Home | Resources | Reference architectures → Kubernetes cluster

## kubernetes

### Kubernetes cluster

**User manual**

Category

General Clusters

Share

f t in

3



2

<https://science-cloud.hu/en/reference-architectures>

# Step 2 - Download the descriptor files

Kubernetes cluster user manual:

<https://git.sztaki.hu/science-cloud/reference-architectures/kubernetes>

## Prerequisites

- Configuring an SSH key on ELKH Cloud
- Terraform and Ansible are required
  - You can install them by running the commands below, or omit the installation and use the [RefArch Toolset Docker image](#), which includes these tools.

Installation of Terraform in accordance with the [Official guide](#):

```
sudo apt-get update && sudo apt-get install -y gnupg software-properties-common curl
curl -fsSL https://apt.releases.hashicorp.com/gpg | sudo apt-key add -
sudo apt-add-repository "deb [arch=amd64] https://apt.releases.hashicorp.com $(lsb_release -cs) main"
sudo apt-get update && sudo apt-get install terraform=1.3.6
```

Installation of Ansible in accordance with the [Official guide](#):

```
sudo apt install python3-pip
sudo python3 -m pip install ansible==6.7.0
```

## Deployment

1. Download and extract descriptor files:

```
wget https://git.sztaki.hu/science-cloud/reference-architectures/kubernetes/-/archive/master/kubernetes-master.tar.gz -O raf-kubernetes.tar
tar -zxvf raf-kubernetes.tar.gz
```

# Step 4 – Personalizing descriptor files

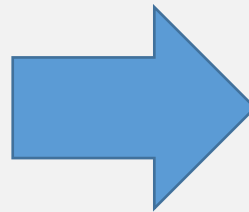
- Auth definition file (**terraform\_openstack/auth\_data.auto.tfvars**):

```
auth_data = ({
  auth_url           = "SET_YOUR_AUTH_URL"
  application_credential_id = "SET_YOUR_APPLICATION_CREDENTIAL_ID"
  application_credential_secret = "SET_YOUR_APPLICATION_CREDENTIAL_SECRET"
})
```

- Node definition file (**terraform\_openstack/resources.auto.tfvars**):

```
kubernetes_master_node = ({
  name           = „kubernetes-master“
  flavor_name    = "SET_YOUR_FLAVOR_NAME"
  floating_ip    = "SET_YOUR_FLOATING_IP"
  image_id       = "SET_YOUR_IMAGE_ID"
  key_pair       = "SET_YOUR_KEY_PAIR_NAME"
  volume_size    = 32
})

kubernetes_worker_node = ({
  name           = „kubernetes-worker“
  count          = 1
  flavor_name    = "SET_YOUR_FLAVOR_NAME"
  image_id       = "SET_YOUR_IMAGE_ID"
  key_pair       = "SET_YOUR_KEY_PAIR_NAME"
  volume_size    = 32
})
```



```
kubernetes_master_node = ({
  name           = „kubernetes-master“
  flavor_name    = “m2.large“
  floating_ip    = “193.225.250.4“
  image_id       = “11bf8b51-63d3-4277-b3dc-387f4c4bfd0a“
  key_pair       = „farkas“
  volume_size    = 32
})

kubernetes_worker_node = ({
  name           = „kubernetes-worker“
  count          = 2
  flavor_name    = “m2.large“
  image_id       = “11bf8b51-63d3-4277-b3dc-387f4c4bfd0a“
  key_pair       = „farkas“
  volume_size    = 32
})
```

# Step 4 – Personalizing descriptor files

- Set Kubernetes network

```
kubernetes_network = ({  
  name = „SET_YOUR_NETWORK_NAME“  
  network_subnet_range = „SET_YOUR_NETWORK_RANGE“  
})
```

- Set Kubernetes software versions

```
software_version = ({  
  kubernetes_version = „1.26“  
  containerd_version = „1.6“  
})
```

- Set user configuration

```
user_config = ({  
  container_network_interface = „flannel“  
  allow_pods_on_master = true  
  enable_gpu = false  
  install_prometheus_stack = true  
})
```

# Step 5 - Terraform initialization

Adding the private SSH key to the SSH agent:

```
# eval $(ssh-agent -s) && echo "$(cat PATH_TO_YOUR_KEY)" | tr -d '\r' | ssh-add -
```

Terraform initialization:

```
# terraform init
```

```
Terraform has been successfully initialized!

You may now begin working with Terraform. Try running "terraform plan" to see
any changes that are required for your infrastructure. All Terraform commands
should now work.

If you ever set or change modules or backend configuration for Terraform,
rerun this command to reinitialize your working directory. If you forget, other
commands will detect it and remind you to do so if necessary.
kporas@Krisz-PC:~/kubernetes/terraform_openstack$ terraform apply --auto-approve
```

# Step 6 - Infrastructure deployment

```
# terraform apply --auto-approve
```

```
null_resource.kubernetes_worker_config[0] (local-exec): TASK [kubernetes_worker : Join the node to cluster] *****
null_resource.kubernetes_worker_config[0]: Still creating... [2m0s elapsed]
null_resource.kubernetes_worker_config[2]: Still creating... [2m0s elapsed]
null_resource.kubernetes_worker_config[1]: Still creating... [2m0s elapsed]
null_resource.kubernetes_worker_config[2] (local-exec): changed: [192.168.0.173]

null_resource.kubernetes_worker_config[2] (local-exec): PLAY RECAP *****
null_resource.kubernetes_worker_config[2] (local-exec): 192.168.0.173      : ok=29  changed=17  unreachable=0  failed=0  skipped=8  rescued=0  ignored=2

null_resource.kubernetes_worker_config[2]: Creation complete after 2m3s [id=4442725935499717348]
null_resource.kubernetes_worker_config[1] (local-exec): changed: [192.168.0.171]

null_resource.kubernetes_worker_config[1] (local-exec): PLAY RECAP *****
null_resource.kubernetes_worker_config[1] (local-exec): 192.168.0.171      : ok=29  changed=17  unreachable=0  failed=0  skipped=8  rescued=0  ignored=2

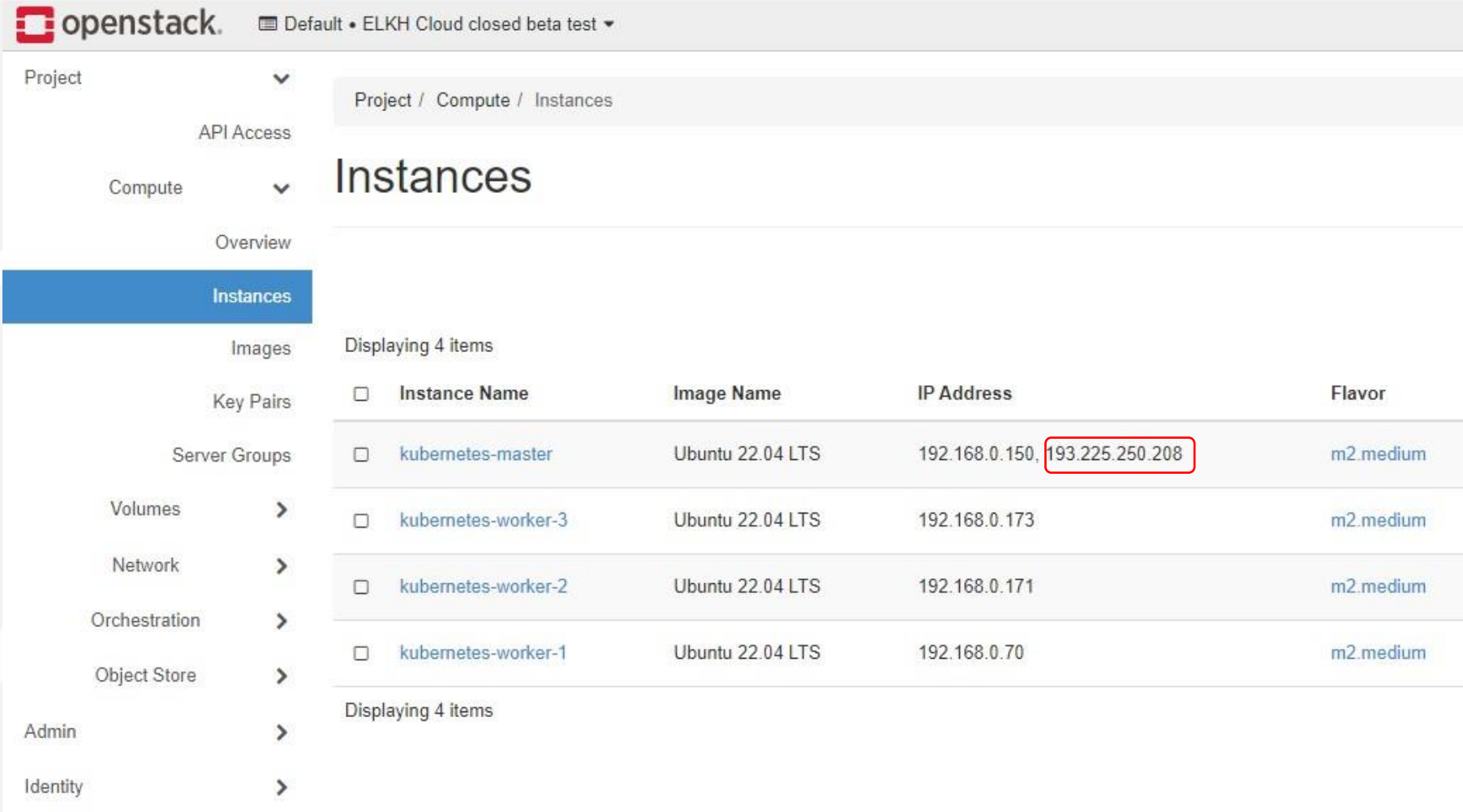
null_resource.kubernetes_worker_config[1]: Creation complete after 2m7s [id=1245252158394522665]
null_resource.kubernetes_worker_config[0]: Still creating... [2m10s elapsed]
null_resource.kubernetes_worker_config[0] (local-exec): changed: [192.168.0.70]

null_resource.kubernetes_worker_config[0] (local-exec): PLAY RECAP *****
null_resource.kubernetes_worker_config[0] (local-exec): 192.168.0.70      : ok=29  changed=17  unreachable=0  failed=0  skipped=8  rescued=0  ignored=2

null_resource.kubernetes_worker_config[0]: Creation complete after 2m11s [id=7442378902838305076]

Apply complete! Resources: 18 added, 0 changed, 0 destroyed.
kporas@Krisz-PC:~/kubernetes/terraform_openstack$
```

# Step 6 - Infrastructure deployment



openstack. Default • ELKH Cloud closed beta test

Project / Compute / Instances

## Instances

Displaying 4 items

<input type="checkbox"/>	Instance Name	Image Name	IP Address	Flavor
<input type="checkbox"/>	kubernetes-master	Ubuntu 22.04 LTS	192.168.0.150.193.225.250.208	m2.medium
<input type="checkbox"/>	kubernetes-worker-3	Ubuntu 22.04 LTS	192.168.0.173	m2.medium
<input type="checkbox"/>	kubernetes-worker-2	Ubuntu 22.04 LTS	192.168.0.171	m2.medium
<input type="checkbox"/>	kubernetes-worker-1	Ubuntu 22.04 LTS	192.168.0.70	m2.medium

Displaying 4 items

# Step 7 - Infrastructure usage - Lens

### Add Clusters from Kubeconfig

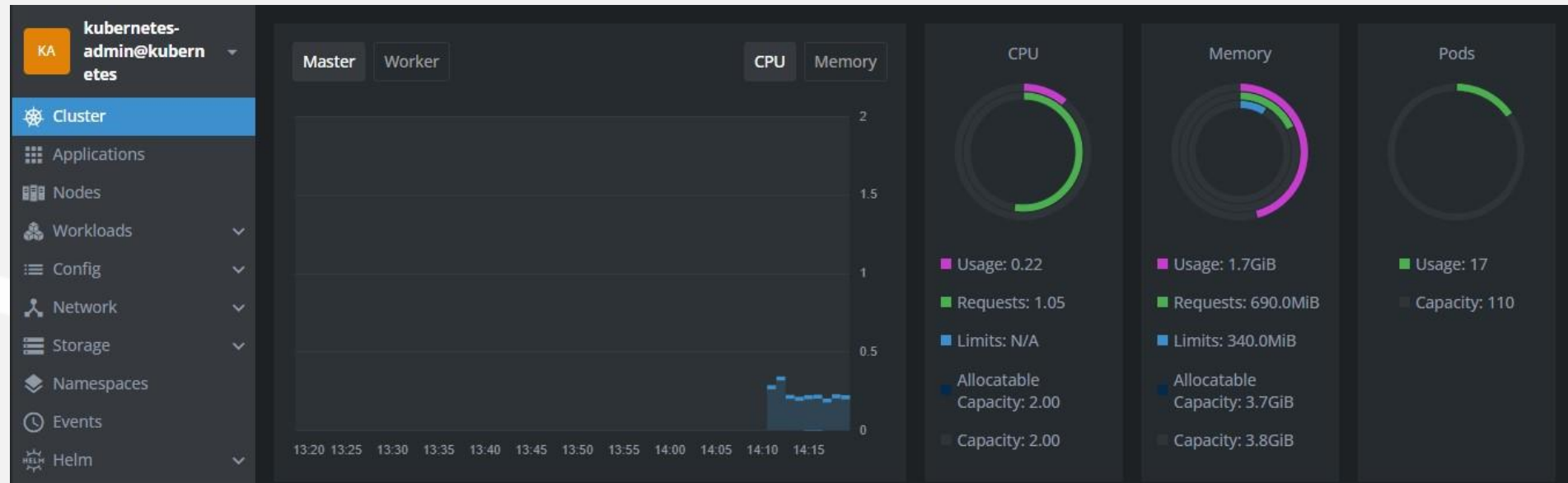
Clusters added here are **not** merged into the `~/.kube/config` file. [Read more about adding clusters.](#)

```
1  apiVersion: v1
2  clusters:
3  - cluster:
4    certificate-authority-data: LS0tLS1CRUdJTiBDRVJUSUZJQ0FURS0tLS0tCk
5    server: https://193.225.250.208:6443
6    name: kubernetes
7  contexts:
8  - context:
9    cluster: kubernetes
10   user: kubernetes-admin
11   name: kubernetes-admin@kubernetes
12  current-context: kubernetes-admin@kubernetes
13  kind: Config
14  preferences: {}
15  users:
16  - name: kubernetes-admin
17    user:
18      client-certificate-data: LS0tLS1CRUdJTiBDRVJUSUZJQ0FURS0tLS0tCk1JS
19      client-key-data: LS0tLS1CRUdJTiBSU0EgUFJJKFURSBLRVktLS0tLQpNSU1Fc
```

[Add cluster](#)



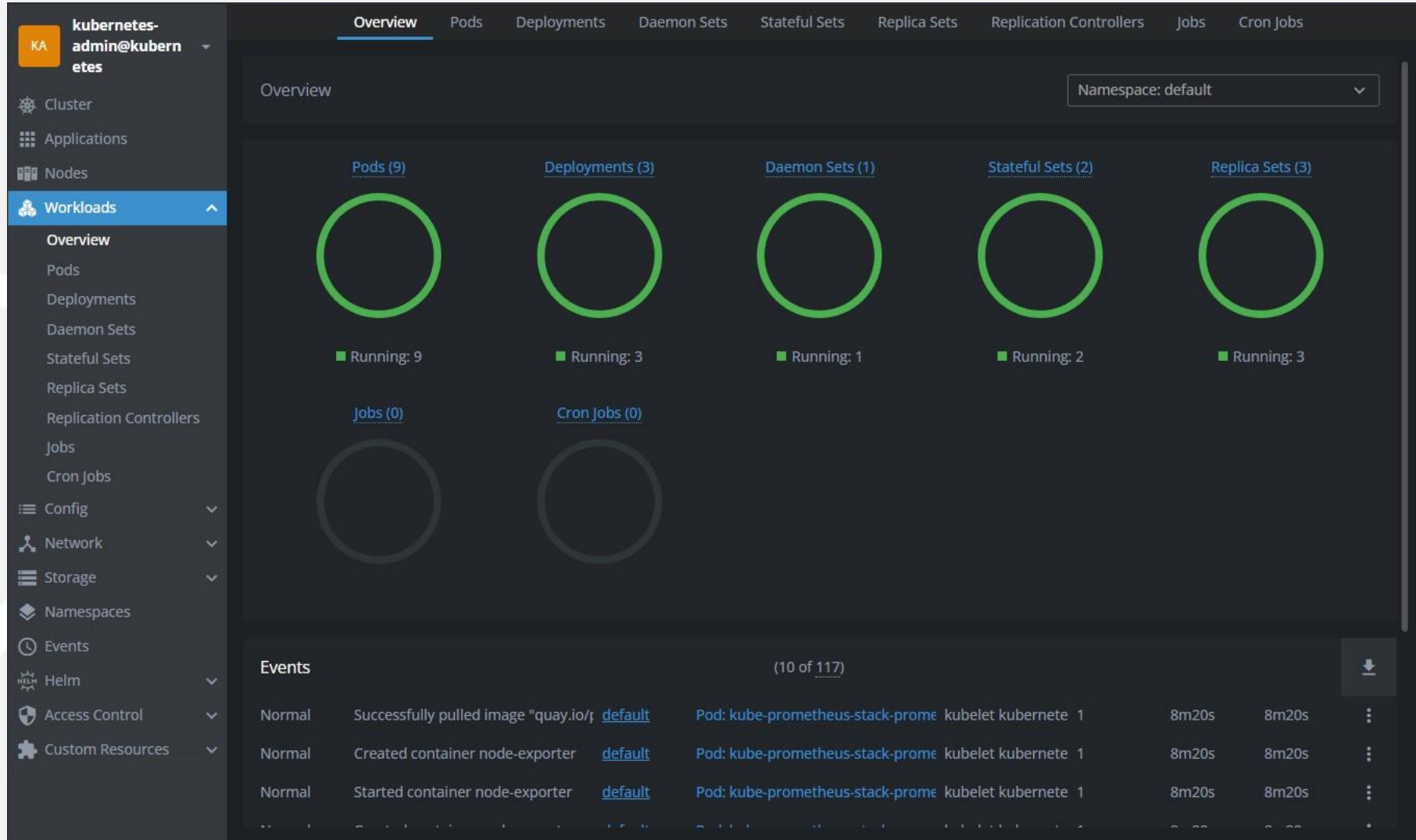
# Step 7 - Infrastructure usage - Lens



The screenshot displays the Kubernetes Lens interface for a cluster named 'kubernetes-admin@kubernetes'. The left sidebar shows navigation options: Cluster, Applications, Nodes, Workloads, Config, Network, Storage, Namespaces, Events, and Helm. The main area shows a list of nodes with columns: Name, CPU, Memory, Disk, Taints, Roles, Version, Age, and Conditions. There are 4 items listed.

Name	CPU	Memory	Disk	Taints	Roles	Version	Age	Conditions
kubernetes-master	0	0	0	0	control-plane	v1.26.9	10m	Ready
kubernetes-worker-1	0	0	0	0		v1.26.9	7m55s	Ready
kubernetes-worker-2	0	0	0	0		v1.26.9	7m59s	Ready
kubernetes-worker-3	0	0	0	0		v1.26.9	8m3s	Ready

# Step 7 - Infrastructure usage - Lens



# Step 7 - Infrastructure usage - Helm

```
ubuntu@kubernetes-master:~$ helm repo add bitnami https://charts.bitnami.com/bitnami
"bitnami" has been added to your repositories
ubuntu@kubernetes-master:~$ helm install my-release --set service.type=NodePort bitnami/nginx
NAME: my-release
LAST DEPLOYED: Wed Oct 25 12:25:13 2023
NAMESPACE: default
STATUS: deployed
REVISION: 1
TEST SUITE: None
NOTES:
CHART NAME: nginx
CHART VERSION: 15.3.5
APP VERSION: 1.25.3

** Please be patient while the chart is being deployed **
NGINX can be accessed through the following DNS name from within your cluster:

    my-release-nginx.default.svc.cluster.local (port 80)

To access NGINX from outside the cluster, follow the steps below:

1. Get the NGINX URL by running these commands:

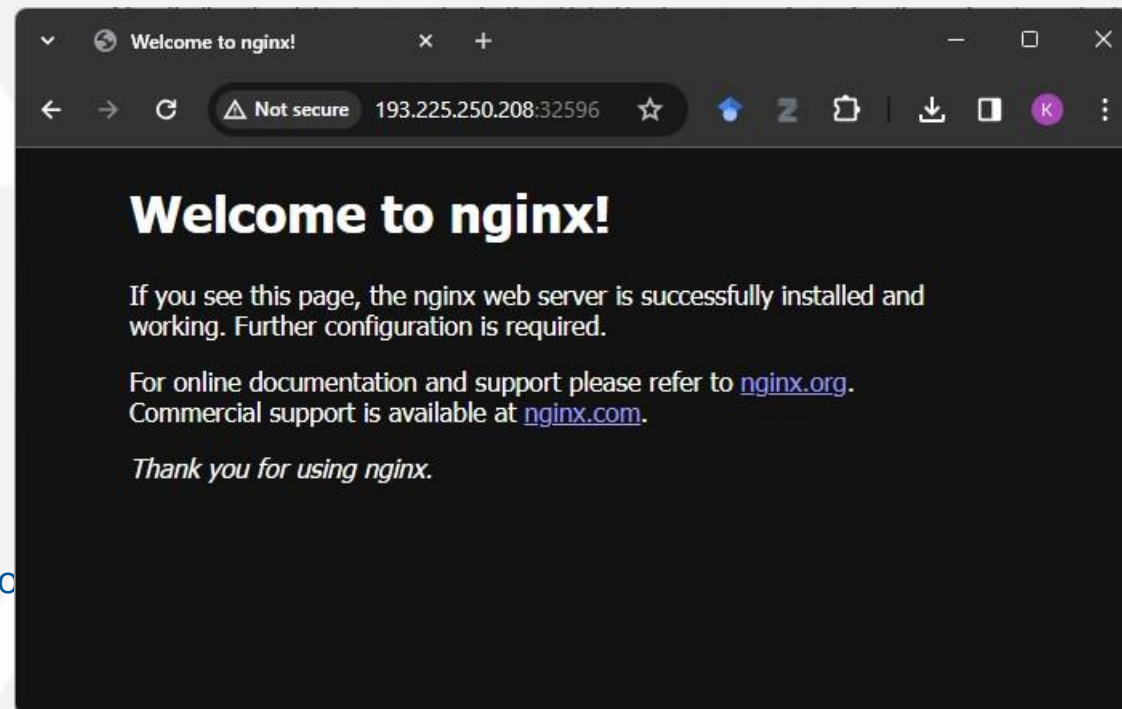
    export NODE_PORT=$(kubectl get --namespace default -o jsonpath="{.spec.ports[0].nodePort}" services my-release-nginx)
    export NODE_IP=$(kubectl get nodes --namespace default -o jsonpath="{.items[0].status.addresses[0].address}")
    echo "http://${NODE_IP}:${NODE_PORT}"
ubuntu@kubernetes-master:~$ kubectl get --namespace default -o jsonpath="{.spec.ports[0].nodePort}" services my-release-nginx
32596ubuntu@kubernetes-master:~$ |
```

# Step 7 - Infrastructure usage - Helm



The screenshot shows the Kubernetes dashboard interface. The left sidebar contains navigation options: Cluster, Applications, Nodes, Workloads (selected), Overview, and Pods. The main area displays the 'Pods' tab for the 'default' namespace, showing 10 items. A table lists the pods with columns for Name, Namespace, Container, CPU, Memory, Restarts, Controlled By, Node, QoS, Age, and Status. The first pod, 'my-release-nginx-7449cfd5b5-zzk6q', is highlighted with a red box and is in a 'Running' state.

Name	Namespace	Contai...	CPU	Memory	Restarts	Controlled By	Node	QoS	Age	Status
my-release-nginx-7449cfd5b5-zzk6q	default	■	0.000	2.7MiB	0	ReplicaSet	kubernetes	BestEffort	2m23s	Running
kube-prometheus-stack-prometheus-r	default	■	0.001	7.8MiB	0	DaemonSet	kubernetes	BestEffort	15m	Running
kube-prometheus-stack-prometheus-r	default	■	0.001	7.5MiB	0	DaemonSet	kubernetes	BestEffort	15m	Running



# Step 7 - Infrastructure usage – Scaling

```
kubernetes_worker_node = ({  
  name = "kubernetes-worker"  
  count = 5  
  flavor_name = "m2.medium"  
  image_id = "7cf55b87-bd6e-4243-a708-b255bc33df2d"  
  key_pair = "kpora"  
  volume_size = 30  
})
```

<input type="checkbox"/>	Instance Name ▲	Image Name	IP Address	Flavor
<input type="checkbox"/>	kubernetes-master	Ubuntu 22.04 LTS	192.168.0.150, 193.225.250.208	m2.medium
<input type="checkbox"/>	kubernetes-worker-1	Ubuntu 22.04 LTS	192.168.0.70	m2.medium
<input type="checkbox"/>	kubernetes-worker-2	Ubuntu 22.04 LTS	192.168.0.171	m2.medium
<input type="checkbox"/>	kubernetes-worker-3	Ubuntu 22.04 LTS	192.168.0.173	m2.medium
<input type="checkbox"/>	kubernetes-worker-4	Ubuntu 22.04 LTS	192.168.0.59	m2.medium
<input type="checkbox"/>	kubernetes-worker-5	Ubuntu 22.04 LTS	192.168.0.148	m2.medium

KA kubernetes-admin@kubernetes

- Cluster
- Applications
- Nodes**
- Workloads
  - Overview
  - Pods
  - Deployments
  - Daemon Sets
  - Stateful Sets

### Nodes

6 items

Name	CPU	Memory	Disk	Taints	Roles	Version
kubernetes-master	<div style="width: 100%;"></div>	<div style="width: 100%;"></div>	<div style="width: 100%;"></div>	0	control-plane	v1.26.9
kubernetes-worker-1	<div style="width: 100%;"></div>	<div style="width: 100%;"></div>	<div style="width: 100%;"></div>	0		v1.26.9
kubernetes-worker-2	<div style="width: 100%;"></div>	<div style="width: 100%;"></div>	<div style="width: 100%;"></div>	0		v1.26.9
kubernetes-worker-3	<div style="width: 100%;"></div>	<div style="width: 100%;"></div>	<div style="width: 100%;"></div>	0		v1.26.9
kubernetes-worker-4	<div style="width: 100%;"></div>	<div style="width: 100%;"></div>	<div style="width: 100%;"></div>	0		v1.26.9
kubernetes-worker-5	<div style="width: 100%;"></div>	<div style="width: 100%;"></div>	<div style="width: 100%;"></div>	0		v1.26.9

# Difference between the SZTAKI and Wigner branches

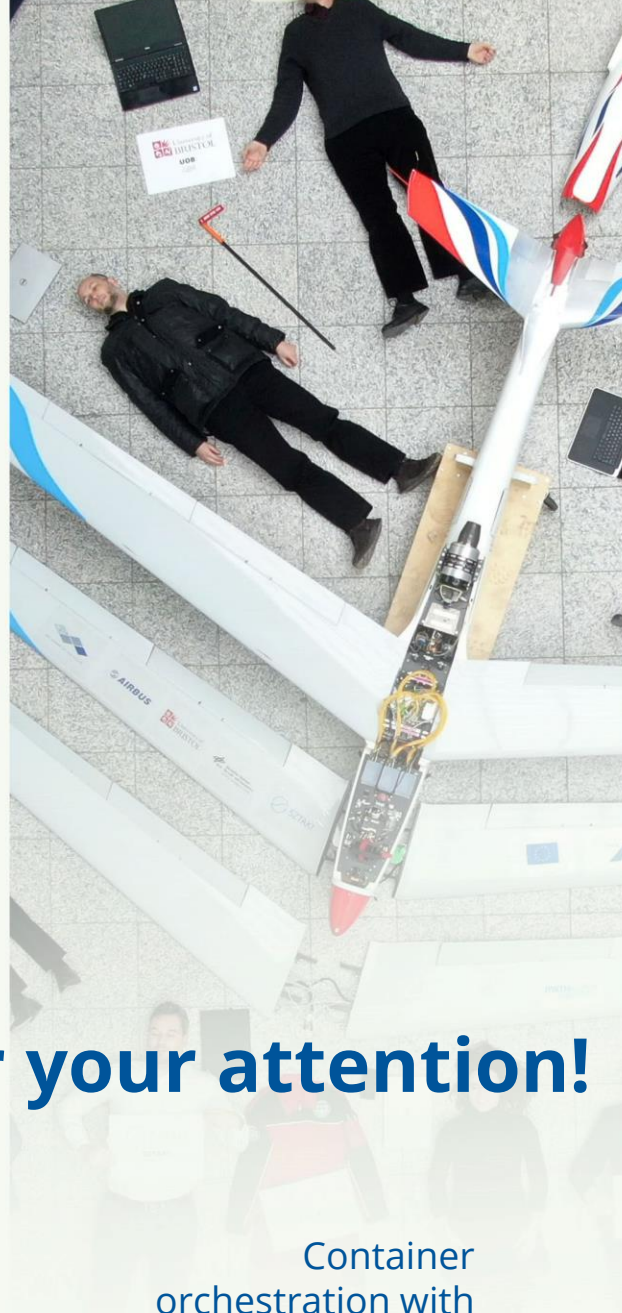
- A VPN connection is required
  - Cloud API
  - Accessing virtual machines
- Specifying a floating IP (external IP address)
- Adding an insecure parameter to the Provider section
- Connection to the interface remains **protected!**
  - VPN, Self-signed certificate

```
provider "openstack" {  
  auth_url           = var.auth_data.auth_url  
  application_credential_id = var.auth_data.application_credential_id  
  application_credential_secret = var.auth_data.application_credential_secret  
  # If your cloud provider uses self-signed cert, use the insecure flag  
  # insecure = true  
}
```

# Summary

- Operation of the Kubernetes reference architecture
- Process of building the reference architecture
- Using Kubernetes
- The creation of different functional Big Data/ML/container/workload manager environments is built automatically using the Terraform tool
- The installation does not require deep IT, network, or software installation and configuration knowledge
- The environments compiled so far (Hadoop, Spark, Tensorflow, Slurm, Horovod, Kafka, Kubernetes, MongoDB, MariaDB, OpenVPN, etc.) are available in the form of a reference architecture and can be tested on the HUN-REN Cloud
- HUN-REN Cloud technical support:

[info@science-cloud.hu](mailto:info@science-cloud.hu)



**Thank you for your attention!**

