



# Running Projects in Application Containers, System Containers & VMs

Differences and Use Cases



# Agenda

- Virtualization and Container Types
- Effective Usage of Infrastructure
- Scaling VMs vs Containers
- Pay-as-you-Go vs Pay-per-Actual-Use
- Kubernetes Challenges & Solutions

# Speakers



Ihor Kolodyuk

Senior Cross-Platform Architect

 **Virtuozzo**



Simon Ekstrand

CTO

 **beebyte**



Tetiana Fydorenchyk

Director of Product Marketing

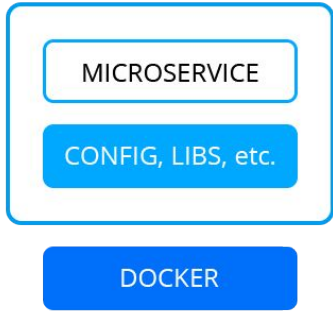
 **Virtuozzo**

# VIRTUALIZATION

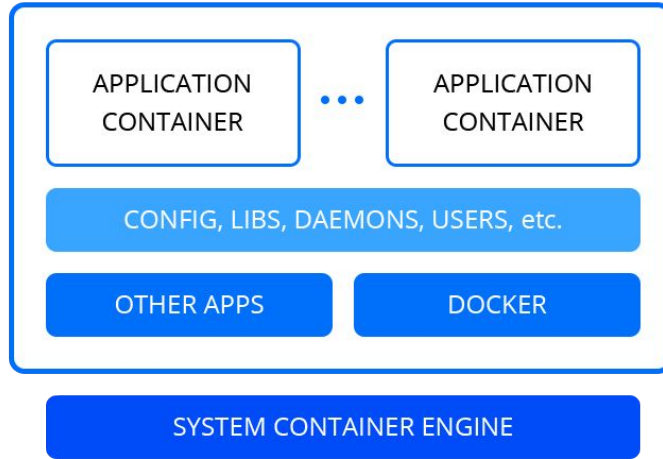
The background features several overlapping, rounded geometric shapes. A large red shape is in the upper right, a blue shape is in the lower right, and a light blue shape is in the lower center. A small yellow shape is visible in the bottom left corner.

# Virtualization Types

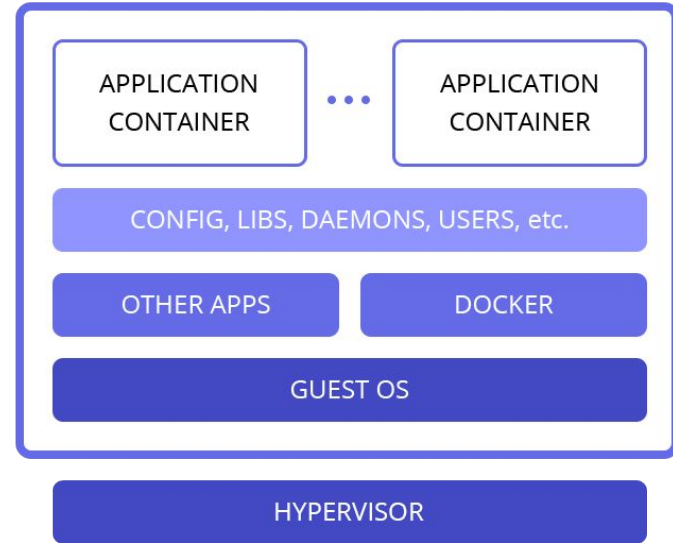
APPLICATION CONTAINER



SYSTEM CONTAINER

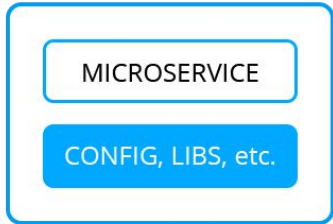


VIRTUAL MACHINE

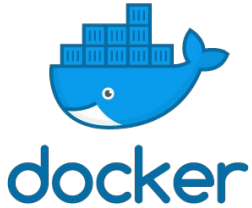


# Virtualization Types

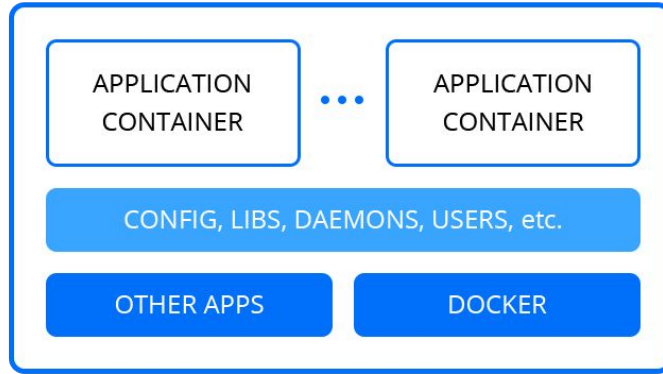
## APPLICATION CONTAINER



DOCKER

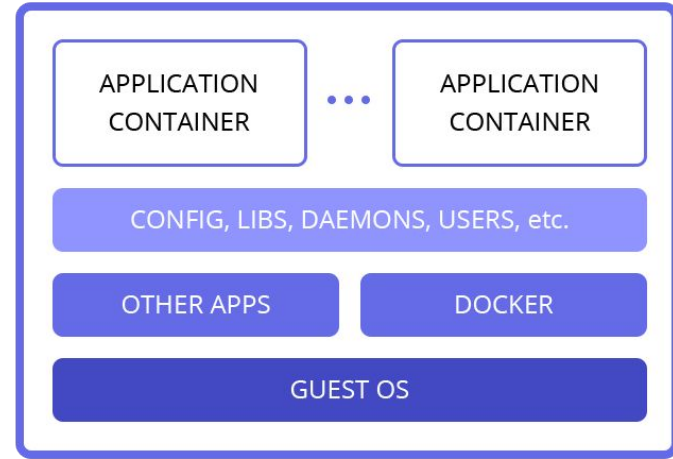


## SYSTEM CONTAINER



SYSTEM CONTAINER ENGINE

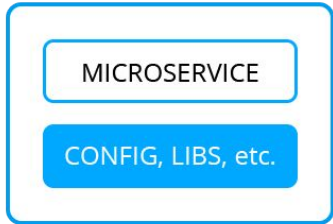
## VIRTUAL MACHINE



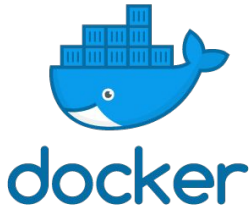
HYPERVISOR

# Virtualization Types

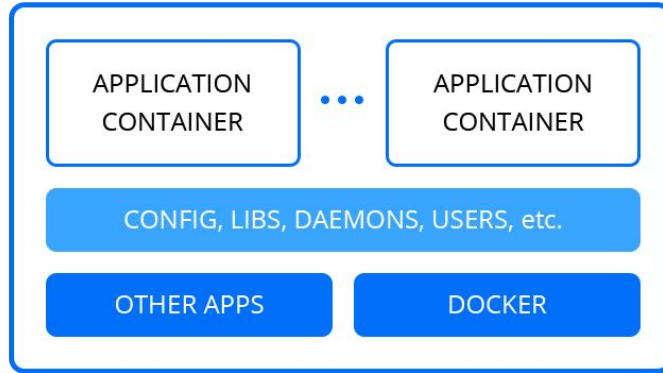
## APPLICATION CONTAINER



DOCKER



## SYSTEM CONTAINER

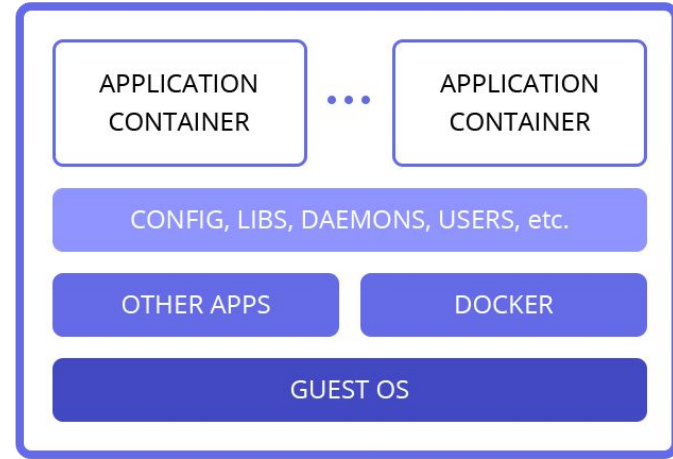


SYSTEM CONTAINER ENGINE

**Virtuozzo**



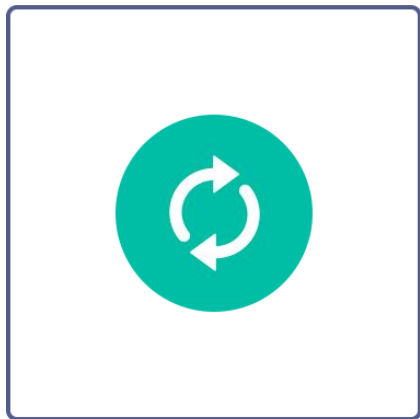
## VIRTUAL MACHINE



HYPERVERSOR

# Container Types

Application Container



System Container



Main Service



Other Main or Auxiliary Services

- Used to run for a single service
- Layered filesystems
- Examples: Docker, CRI-O

- Used as an OS to run multiple services
- No layered filesystems by default
- Stronger isolation
- Examples: Virtuozzo, LXC



# Certified Managed Containers

Java PHP Ruby .NET Node.js Python GO Lang Docker

SSL

**Application Servers** ON

Vertical Scaling per Node

Reserved 24 cloudlet(s) 3 GiB, 9.6 GHz

Scaling Limit up to 80 cloudlet(s) up to 10 GiB, 32 GHz

Horizontal Scaling

Tomcat 9.0.36

Tomcat 9.x.x

GlassFish 8.x.x

Java Engine 7.x.x

Jenkins

Jetty

Payara

Spring Boot

TomEE+

WildFly

PHP

Tomcat 9.0.36 Oracle OpenJD...

Disk Limit 10 GB

Sequential restart delay 30 sec

High-Availability OFF

Public IPv4 OFF

Variables Volumes Links More

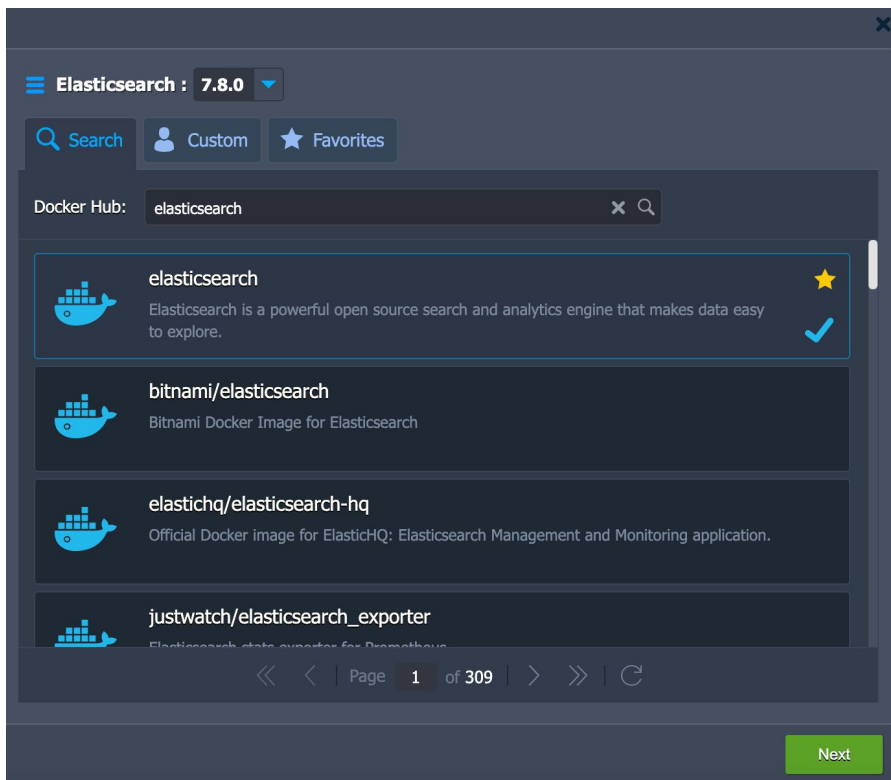


# Virtual Private Servers (Elastic VPS)

The screenshot displays the Elastic VPS configuration interface. On the left, a vertical stack of services is shown: Balancing, Application Servers, Cache, SQL, NoSQL, and Storage. The main configuration area is titled "Elastic VPS" and is currently "ON". It features a "Vertical Scaling per Node" section with a "Reserved" slider set to 24 cloudlet(s) (3 GiB, 9.6 GHz) and a "Scaling Limit" slider set to up to 80 cloudlet(s) (up to 10 GiB, 32 GHz). Below this is the "Horizontal Scaling" section, showing 2 "Stateful" instances. The operating system is set to "CentOS 7.7", with a dropdown menu open showing options: CentOS 7.7, Debian 7.6, Ubuntu, Virtuozzo 7, and Docker Image... Other settings include "Disk Limit" at 10 GB, "Sequential restart delay" at 30 sec, and "Public IPv4" set to OFF. At the bottom, there are buttons for "Variables", "Volumes", "Links", and "More".



# Custom Docker Containers



# Docker Engine CE (Docker Native)

## Docker Engine CE

This package provides standalone Docker Engine. You can use it for performing *docker build*, *docker run*, *docker-compose up* and *docker swarm join*. For using *docker swarm* and *docker stack* please review [dedicated package](#) of Docker Swarm cluster.

Docker Version: 20.10.7

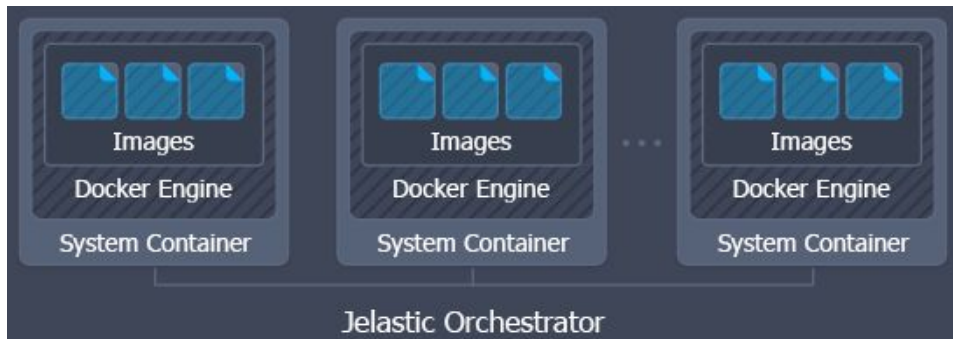
- Create a clean standalone engine
- Connect to an existing swarm cluster
- Deploy containers from compose.yml
- Install [Portainer UI](#) and Let's Encrypt SSL certificates

Environment: dockerengine ✓ .vip.jelastic.cloud

Display Name: Docker Engine CE


Region: Premium

Back Install



# Kubernetes Cluster

## Kubernetes Cluster

 Dedicated Kubernetes cluster with automated scaling and cost efficient pay-per-use pricing for running cloud-native microservice applications. The cluster can be used for development and production environments.

Version: v1.19.10 | K8s Dashboard: v2

Topology: Development | Ingress Controller: NGINX

Deployment:  Clean cluster  Custom

NFS Storage:

Modules:  Prometheus & Grafana  Jaeger Tracing Tools  Remote API Access

Environment: kubernetes | .vip.jelastic.cloud

Display Name: Kubernetes Cluster

Region: Premium





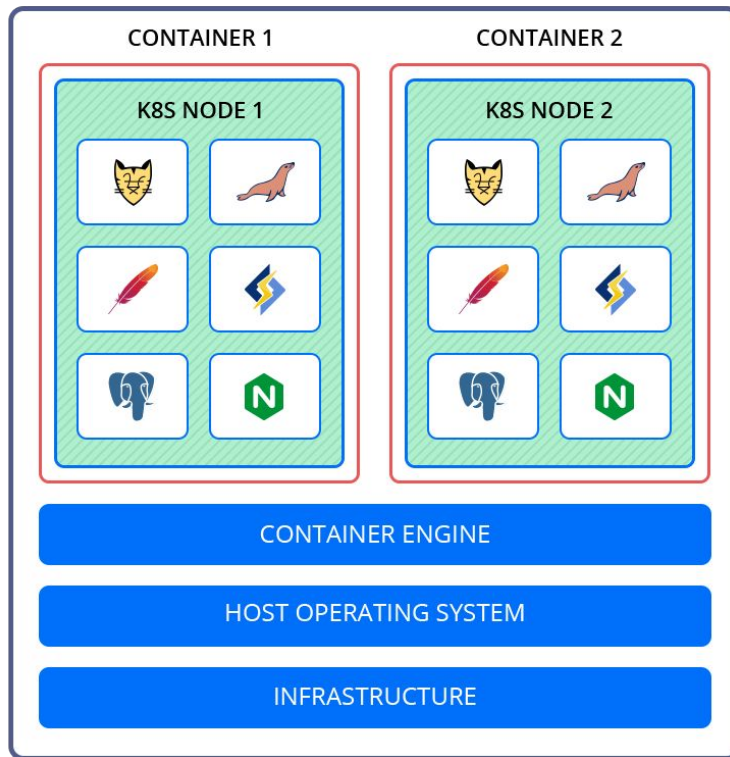
# EFFECTIVE USAGE OF INFRASTRUCTURE

# Running Kubernetes on VMs vs System Containers

VM-BASED HOST

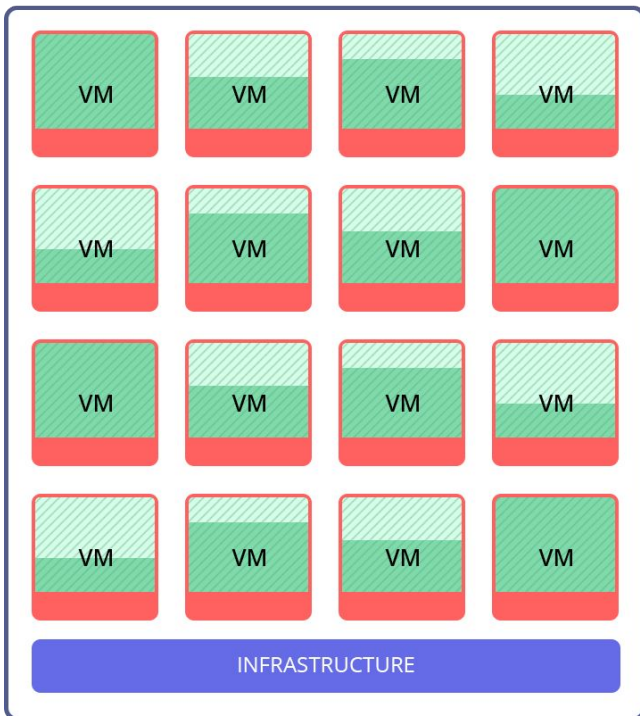


CONTAINER-BASED HOST

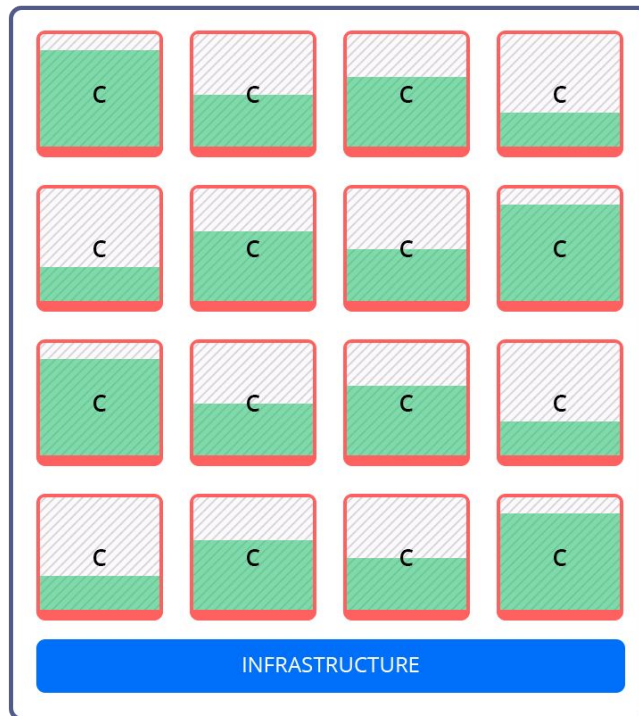


# Kubernetes on VMs vs System Containers

VM-BASED HOST



CONTAINER-BASED HOST

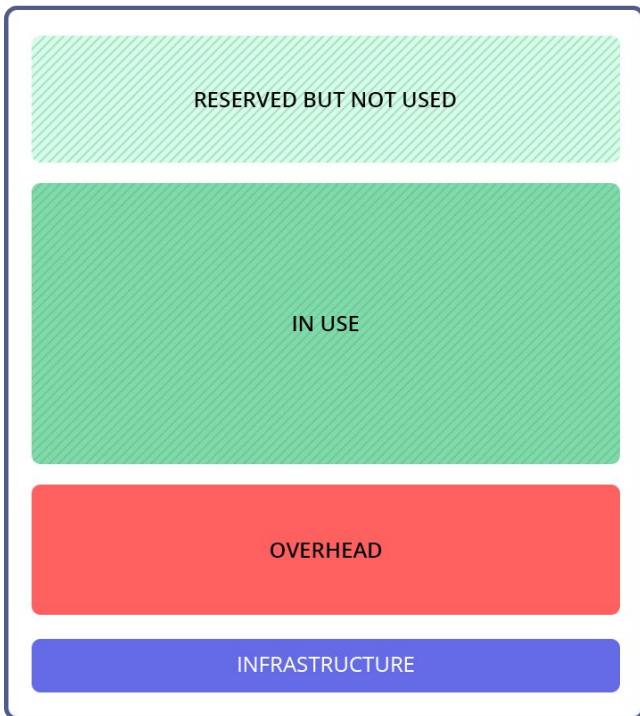


■ Guest OS Footprint   ■ In Use   ■ Reserved but not used   ▨ Available

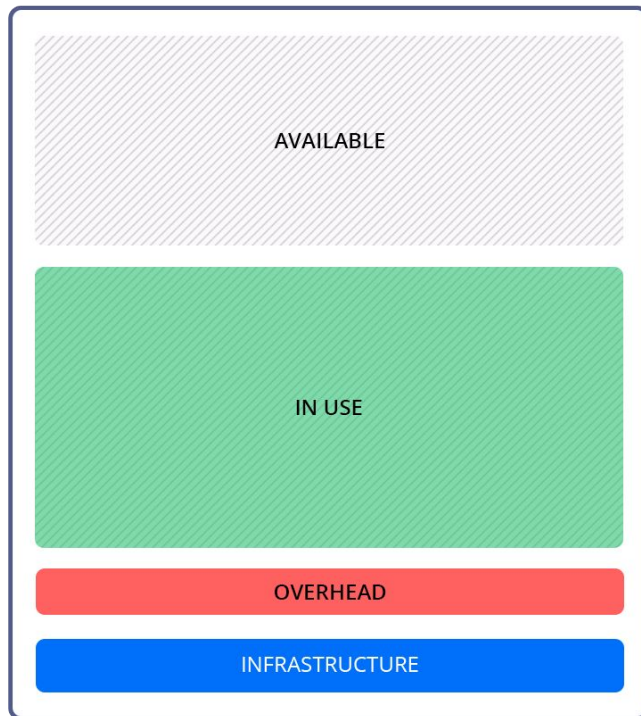


# Kubernetes on VMs vs System Containers

VM-BASED HOST

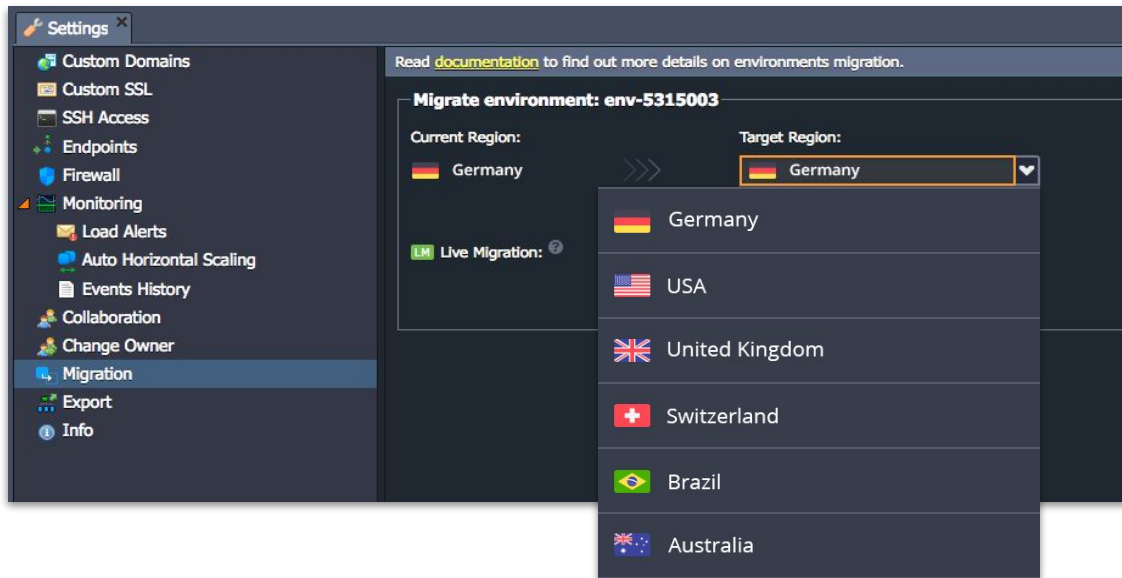


CONTAINER-BASED HOST

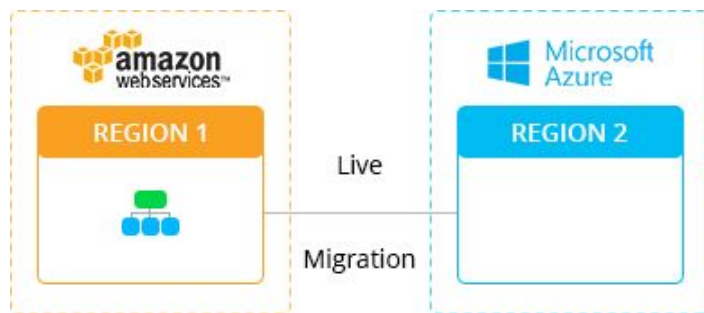
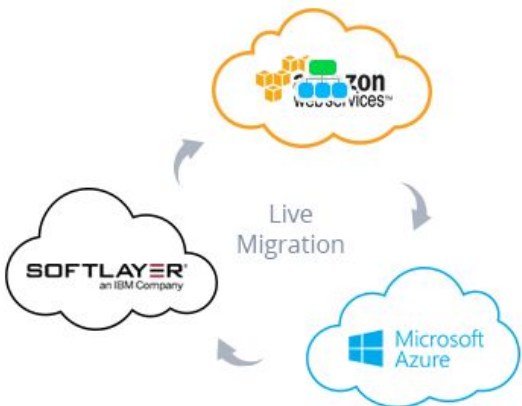
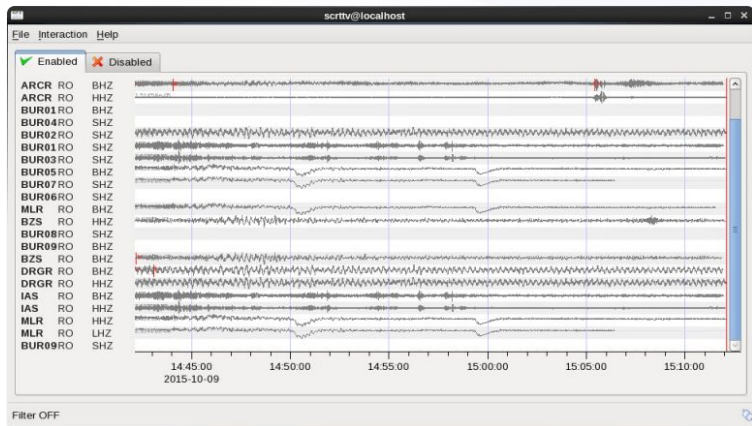


# Live Migration of Containers

- Zero downtime hardware maintenance
- Load rebalancing across host nodes
- High-availability across clouds



# Live Migration across Clouds without Downtime



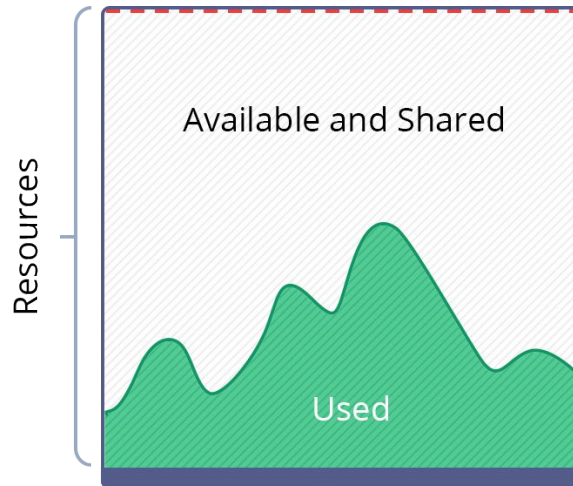
# SCALING JAVA IN VMs vs CONTAINERS

The background features several overlapping, rounded geometric shapes. A large red shape is in the upper right, a blue shape is in the lower right, and a light blue shape is in the lower middle. The overall aesthetic is clean and modern.

# Resource Limit vs Real Usage in VM and Container



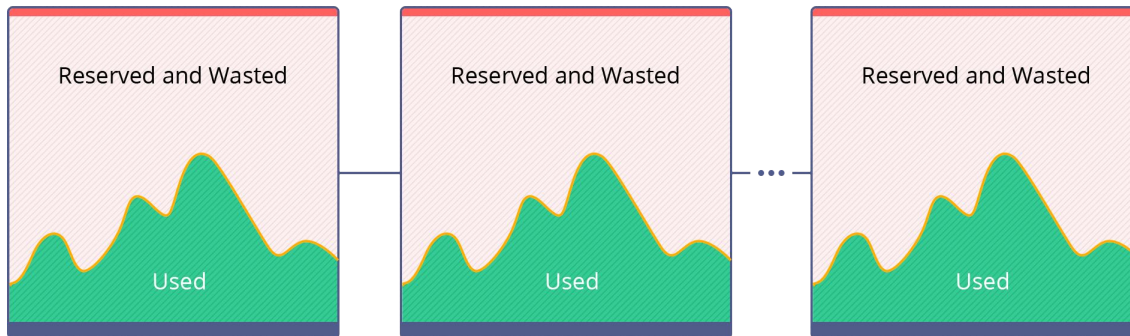
VM



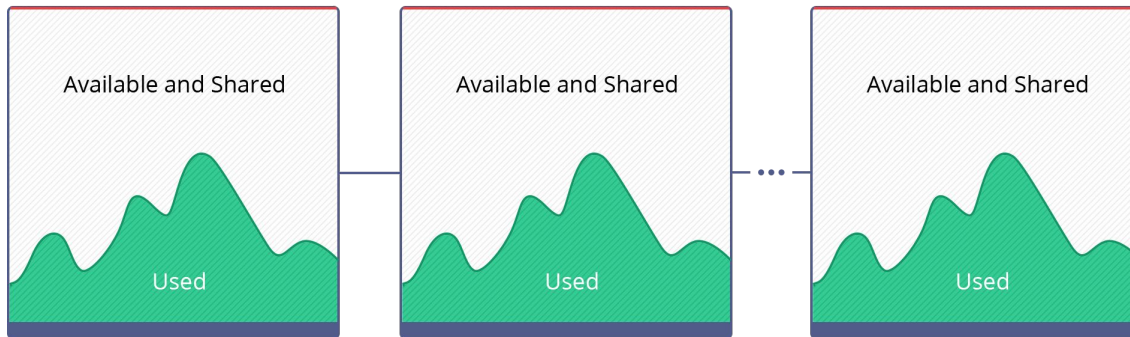
Container

# Horizontal Scaling: VMs vs Containers

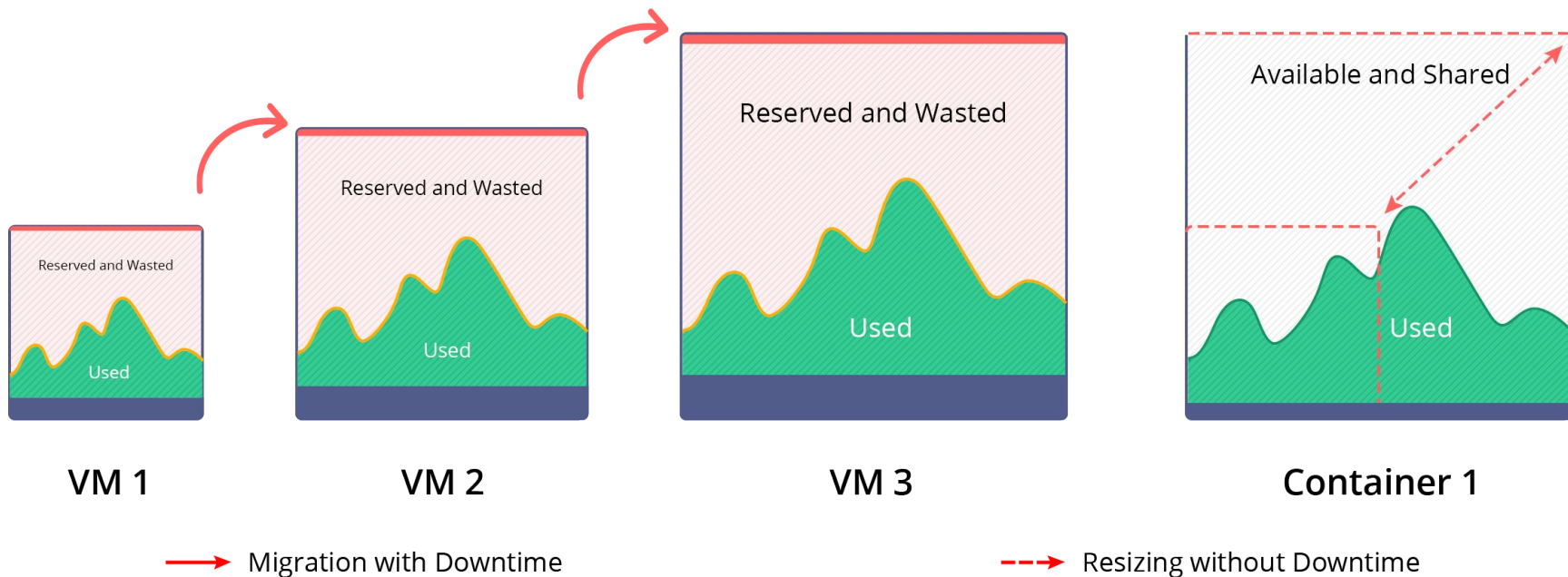
VMs



Containers



# Vertical Scaling: VMs vs Containers



Resizing of the same container on the fly is **easier, cheaper and faster** than moving to a larger VM.

# Automatic Vertical Scaling

Every container hosted with Jelastic PaaS is divided into granular units – cloudlets (128MiB of RAM and 400MHz of CPU)

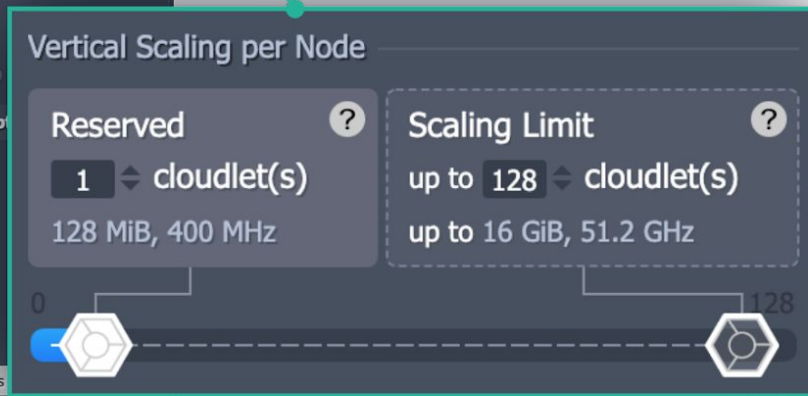
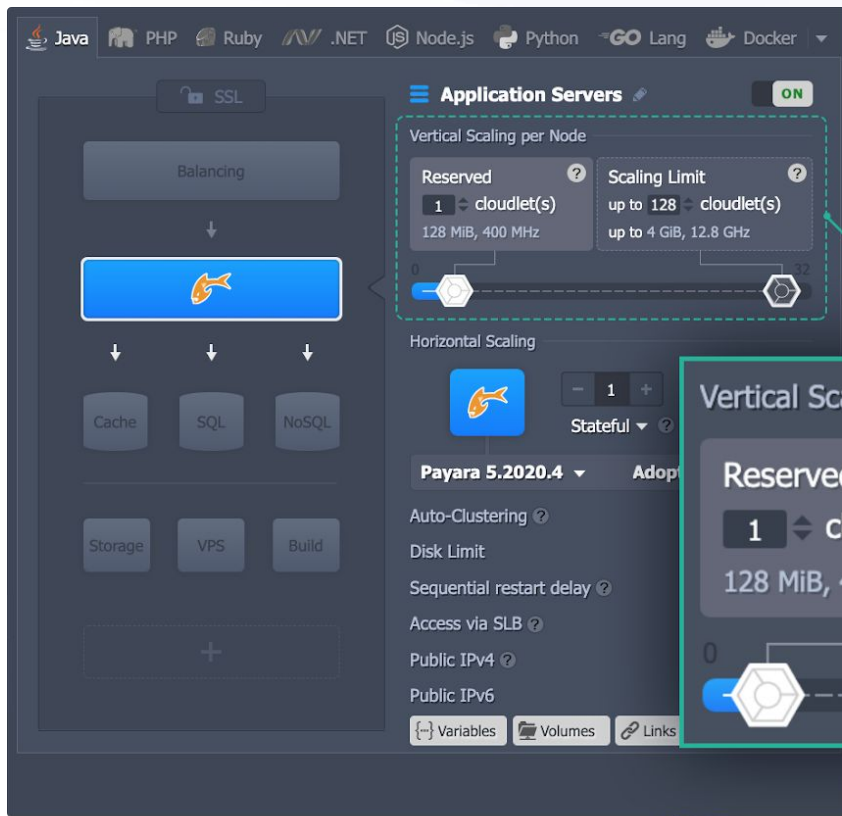
The screenshot displays the Jelastic PaaS control panel. At the top, there are language and technology icons for Java, PHP, Ruby, .NET, Node.js, Python, GO Lang, and Docker. Below this, the 'Application Servers' section is active, with a toggle switch set to 'ON'. The main area shows a configuration for a server with a blue fish icon. A dashed green box highlights the 'Vertical Scaling per Node' settings, which include a 'Reserved' field set to 1 cloudlet(s) (128 MiB, 400 MHz) and a 'Scaling Limit' field set to up to 32 cloudlet(s) (up to 4 GiB, 12.8 GHz). Below this, there are sections for 'Horizontal Scaling' (set to 1), 'Stateful' (set to Stateful), and 'Payara 5.2020.4'. Other settings like Auto-Clustering, Disk Limit, Sequential restart delay, Access via SLB, Public IPv4, and Public IPv6 are visible. At the bottom, there are buttons for 'Variables', 'Volumes', and 'Links'.

This is a zoomed-in view of the 'Vertical Scaling per Node' settings. It shows two main sections: 'Reserved' and 'Scaling Limit'. The 'Reserved' section is set to 1 cloudlet(s) with a sub-note of 128 MiB, 400 MHz. The 'Scaling Limit' section is set to up to 32 cloudlet(s) with a sub-note of up to 4 GiB, 12.8 GHz. Below these sections is a horizontal slider with a blue indicator at the left end (0) and a white indicator at the right end (128). The slider has two gear icons at its ends.



# Automatic Vertical Scaling

You can set up a maximum Scaling Limit for each container, so the resources will be always available in case of load spikes or other consumption changes



# Horizontal Scaling

The screenshot displays the Jelastic control panel interface for configuring application servers. The top navigation bar includes language options (Java, PHP, Ruby, .NET, Node.js, Python, Lang, Docker) and a 'Premium' status indicator.

**Application Servers Configuration:**

- Vertical Scaling per Node:** A slider shows 'Reserved' at 16 cloudlet(s) (2 GiB, 6.4 GHz) and 'Scaling Limit' at 80 cloudlet(s) (up to 10 GiB, 32 GHz).
- Horizontal Scaling:** A slider is set to 4. A dropdown menu is open, showing 'Stateful' selected, with 'Stateless' and another 'Stateful' option visible.
- Application:** Payara 5.2020.4, OpenJDK...
- Auto-Clustering:** OFF
- Disk Limit:** 10 GB
- Sequential restart delay:** 30 sec
- Access via SLB:** ON
- Public IPv4:** OFF
- Additional Options:** Variables, Volumes, Links, More

**Resources Summary:**

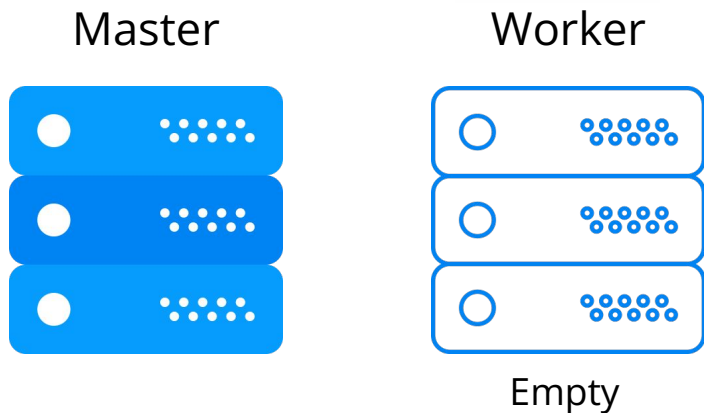
- 1 cloudlet = 128 MiB + 400 MHz
- From Reserved Cloudlets: 40 + 32 = 72
- To Scaling Limit: 40 + 160 = 200

**Deployment Overview:**

- payara** (payara.vip.jelastic.cloud)
- Load Balancer** (NGINX 1.18.0)
- Application Servers x 4** (Payara 5.2020.4)
- Node ID: 266896 (IP: 172.25.2.53)
- Public IPv4
- Node ID: 267716
- Node ID: 267717
- Node ID: 267718

# Stateless (Create New) vs Stateful (Clone)

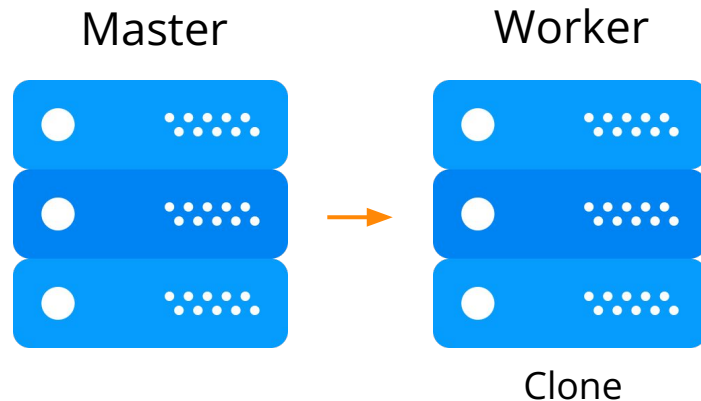
## Stateless



**Stateless** mode creates an empty node from a base container image template.

Works faster than stateful and easy to parallelize the scaling process.

## Stateful



**Stateful** mode creates a new node as a full copy (clone) from the master.

Usually takes longer than stateless, but data is replicated automatically.

# Automatic Horizontal Scaling

The screenshot displays the 'Settings' menu on the left, with 'Auto Horizontal Scaling' selected. The main panel shows the configuration for 'Application Servers' under the 'CPU' tab. The 'Add Nodes' and 'Remove Nodes' sections are highlighted with red boxes. A line graph on the right shows CPU usage over time, with a red horizontal line at 85% and a blue horizontal line at 35%.

**Settings**

- Custom Domains
- Custom SSL
- SSH Access
- Endpoints
- Firewall
- Load Alerts
- Auto Horizontal Scaling**
- Collaboration
- Change Owner
- Migration
- Export
- Info

**Application Servers** | CPU | Memory | Network | Disk I/O | Disk IOPS

**Add Nodes**

- When loading is more than: 85 %
- For at least: 5 minutes
- Scale out to: 8 nodes by 1 count

**Remove Nodes**

- When loading is less than: 35 %
- For at least: 10 minutes
- Scale in to: 2 nodes by 1 count

Send Email Notification:  ON

Close **Add**

**CPU (Hz)** 12 hours

85% 15 GHz

35% 5 GHz

02Oct 02:00 04:00 06:00 08:00 10:00

# Load Alerts

The screenshot shows the 'Settings' tab for 'Application Servers'. The left sidebar contains various configuration options, with 'Load Alerts' selected. The main area displays a table of alerts for the 'Load Balancer' (NGINX 1.18.0). The table has columns for 'Name' and 'Condition'. The alerts listed are:

Name	Condition
auto_alert_cpu	CPU > 70%
auto_alert_mem	Memory > 80%
auto_alert_disk	Storage > 90%
auto_alert_inodes	
auto_alert_net_ext_out	
auto_alert_oom_killer	

Below the table, there are sections for 'Application Servers' (Tomcat 10.0.0-M9 x 3) and 'SQL Databases' (MariaDB 10.4.16 x 2).

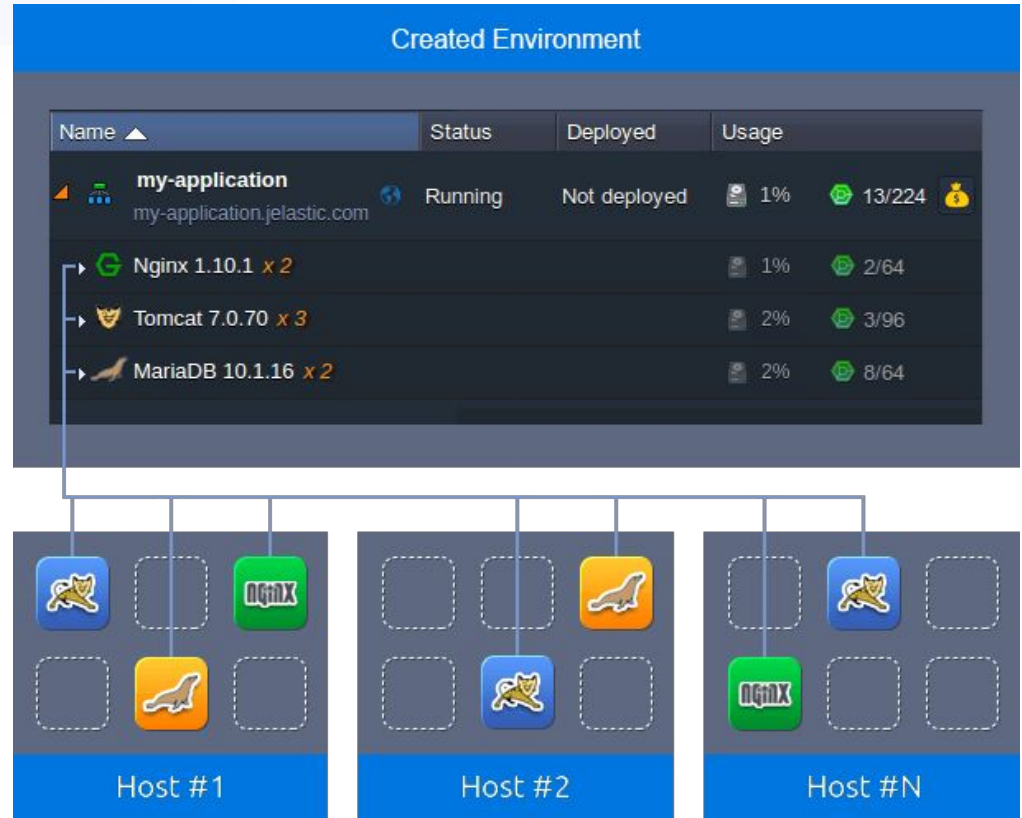
The 'Add Alert' dialog box is shown with the following configuration:

- Name: Alert 1
- Nodes: Application Servers
- Whenever: Cloudlets (Memory, CPU)
- Is: > 55 %
- For at least: 10 minutes
- Notification frequency: 1 hour(s)

Buttons: Cancel, Add

# Anti-Affinity Rules

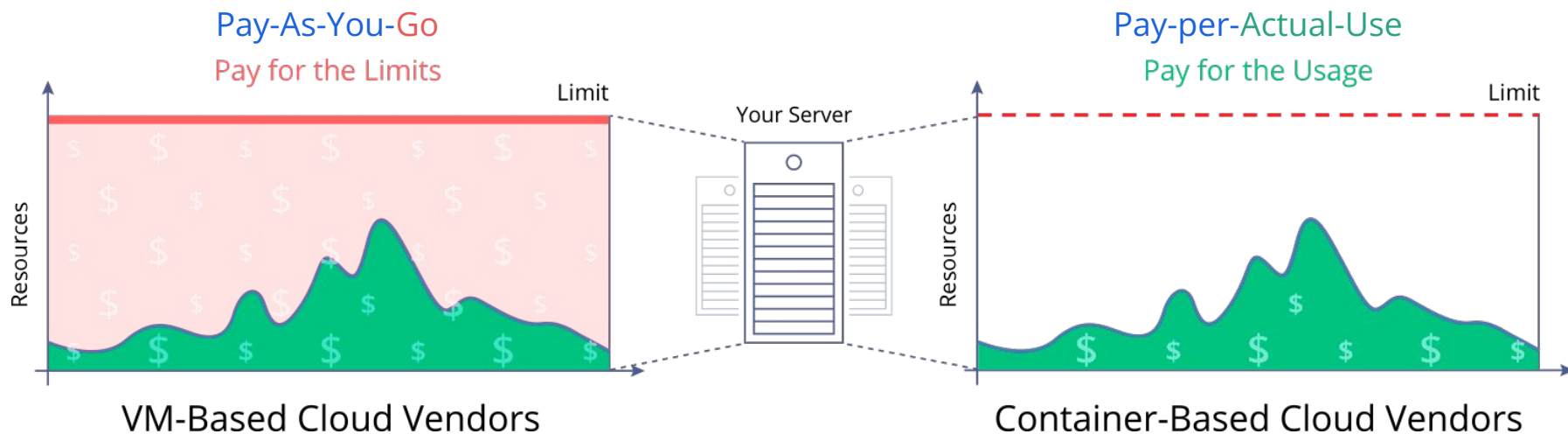
All newly added containers of the single layer are created at the different hosts, providing advanced high-availability and failover protection.





PAY-PER-USE  
VS  
PAY-AS-YOU-GO

# Pay-As-You-Go vs Pay-per-Actual-Use



Using automatic vertical scaling, cloud providers can offer economically advantageous pricing based on the actual resource consumption

**Forbes** - [Deceptive Cloud Efficiency: Do You Really Pay As You Use?](#)



# Linux Terminology for RAM Consumption

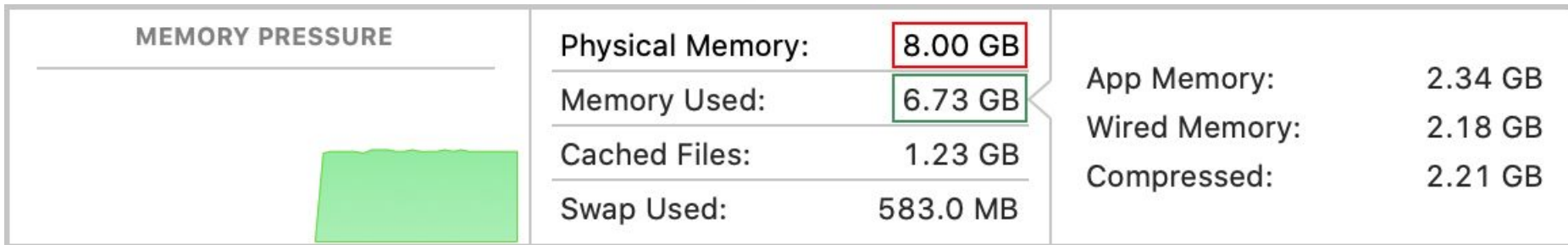
~\$ [free -m](#)

```
tomcat@node275130-0002-test-env ~ $ free -m
Mem:      total      used      free      shared  buff/cache   available
Swap:    4096         0      4096
```

- **total** - memory limit, amount of memory that can be used
- **used** - currently used memory, calculated as  $total - free - buffers - cache$
- **shared** - extra used memory and shared with other containers on the same host
- **buff/cache** - temporary used memory which can be reclaimed any time on demand

# MacOS Terminology for RAM Consumption

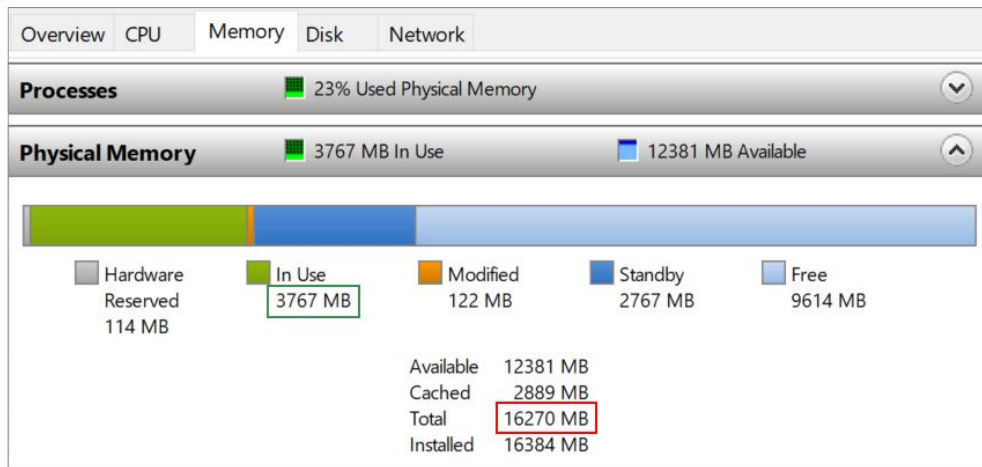
To check **RAM usage** on **Mac**, go to Activity Monitor (Applications > Utilities)



- **Physical Memory** - the amount of RAM installed
- **Memory Used** - the amount of RAM being used
  - **App Memory** - the amount of memory being used by apps
  - **Wired Memory** - memory required by the system to operate. This memory can't be cached and must stay in RAM, so it's not available to other apps
  - **Compressed** - the amount of memory that has been compressed to make more RAM available

# Windows Terminology for RAM Consumption

- **Total** - memory limit, amount of memory that can be used
- **In Use** - currently used memory, calculated as  $total - free - standby - modified$
- **Modified** - memory whose contents must be written to disk before it can be used for another purpose
- **Standby** - memory that contains cached data and code that is not actually in use
- **Free** - memory does not contain any valuable data and that will be used first and processors drivers or the operating system needs more memory



# Speculation with Pay-per-Use Term

**Pay-per-Use** **!=** {  
Pay-as-You-Go  
Pay-as-You-Allocate  
Pay-as-You-Reserve  
} **= Pay-for-Limits**

# AWS's Pay-per-Use = Pay-per-Allocated-Limits

## EC2 Launch Type Model

There is no additional charge for EC2 launch type. You pay for AWS resources (e.g. EC2 instances or EBS volumes) you create to store and run your application. **You only pay for what you use, as you use it**; there are no minimum fees and no upfront commitments.

See detailed pricing information on the [Amazon EC2 pricing page](#).

## Task size

Task size allows you to size at the task level and optionally set container-specific CPU and memory sizes. You are **billed for the task memory and task CPU allocated**.

Task memory\*

16GB (16384) ▼

Task CPU\*

4 vCPU (4096) ▼

0.25 vCPU (256)

0.5 vCPU (512)

1 vCPU (1024)

2 vCPU (2048)

4 vCPU (4096)

<https://aws.amazon.com/fargate/pricing/>

# Google's Pay-per-Use = Pay-per-Allocated-Limits

## Cloud Run pricing

Cloud Run charges you only for the resources you use, rounded up to the nearest 100 millisecond. Note that each of these resources have a free tier. Your total Cloud Run bill will be the sum of the resources in the pricing table.

<https://cloud.google.com/run/pricing#cloudrun-pricing>

Cloud Run | Create service

**Capacity**

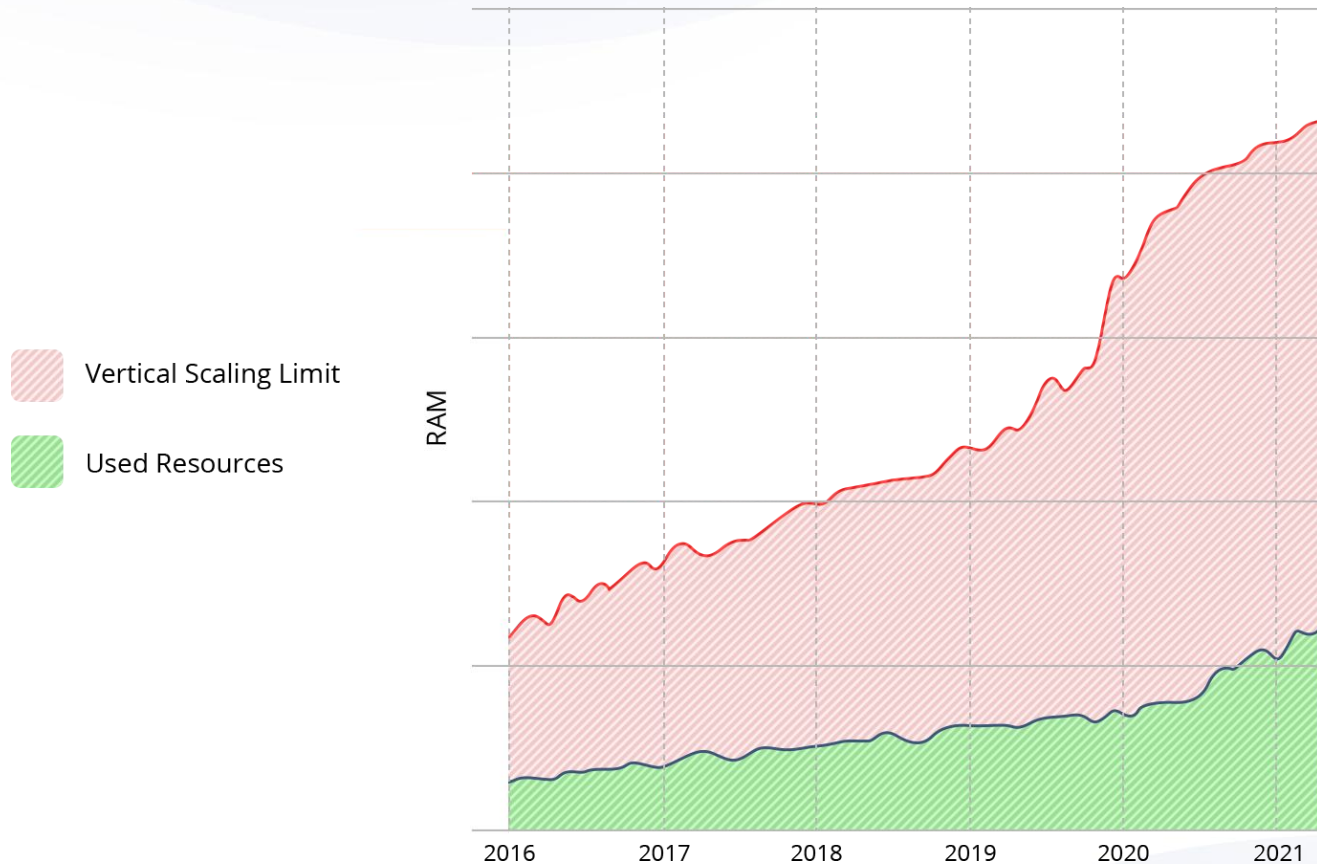
Memory allocated: 4 GiB  
Memory to allocate to each container instance.

CPU allocated: 4  
Number of vCPUs allocated to each container instance.

Request timeout: 300 seconds  
Time within which a response must be returned (maximum 3600 seconds).

Maximum requests per container: 80  
The maximum number of concurrent requests that can reach each container instance. [What is concurrency?](#)

# Real Statistics of Resource Consumption with Containers



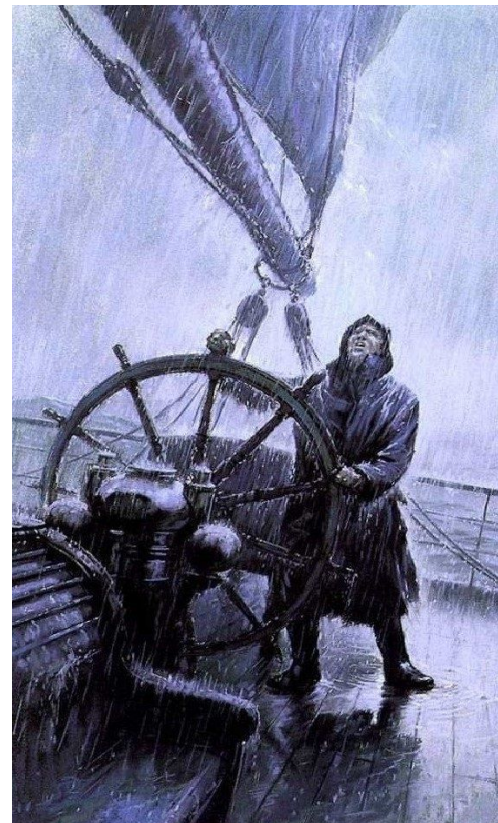
# KUBERNETES HOSTING

The background features several overlapping, rounded geometric shapes. A large red shape is in the upper right, a blue shape is in the lower right, and a light blue shape is in the lower center. A small yellow shape is visible in the bottom left corner.




# Technical Struggles with Kubernetes Services

- Too many components to manage (pod, node, service, ingress and ingress controller, namespace, deployment, statefulset, RBAC, nodeport, load balancer, physical volume, physical volume claim, networks, resource limits, and so on)
- High entry barrier for beginners, most of features are API-managed only, default Kubernetes Dashboard UI provides limited functionality
- Migration complexity of traditional and legacy applications
- K8s was designed for large scale cloud-native apps and microservices, so it's not suitable for all workloads
- Upgrade to next Kubernetes version requires proper automation and may be a challenge



# Automated Kubernetes Cluster Installation

## Kubernetes Cluster ★

 Dedicated Kubernetes cluster with automated scaling and cost efficient pay-per-use pricing for running cloud-native microservice applications. The cluster can be used for development and production environments.

Version:  K8s Dashboard:

Topology:  Ingress Controller:

Deployment:  Clean cluster  Custom

NFS Storage

Modules:  Prometheus & Grafana  Jaeger Tracing Tools  
 Remote API Access

Environment:   .vip.jelastic.cloud

Display Name:

Region:

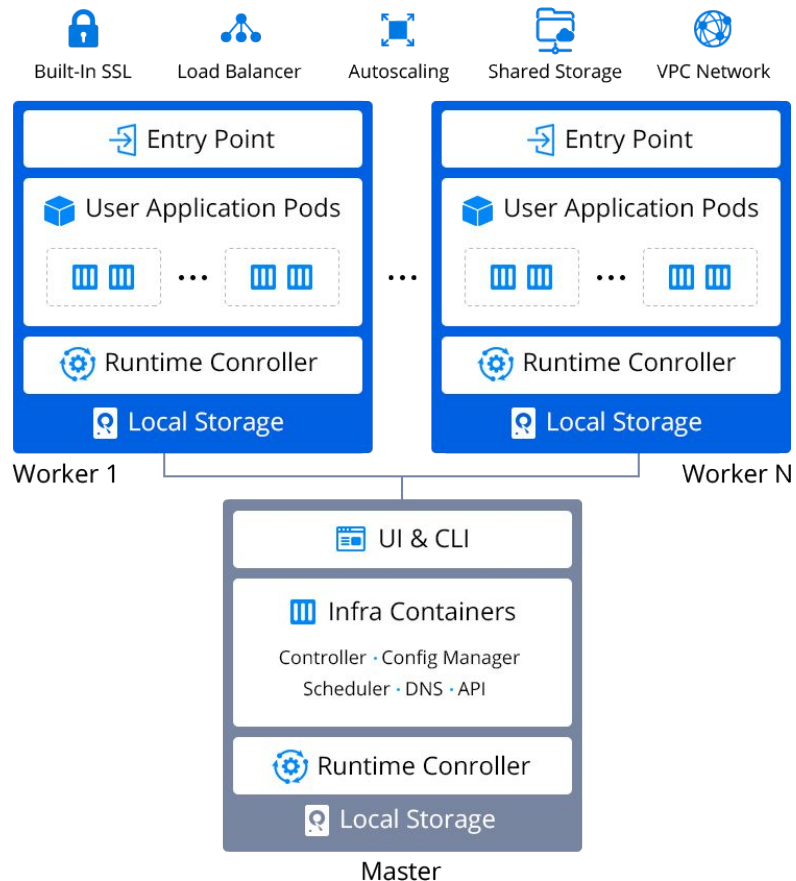
## Cluster Configuration

Configure remote API access and install complementary tools

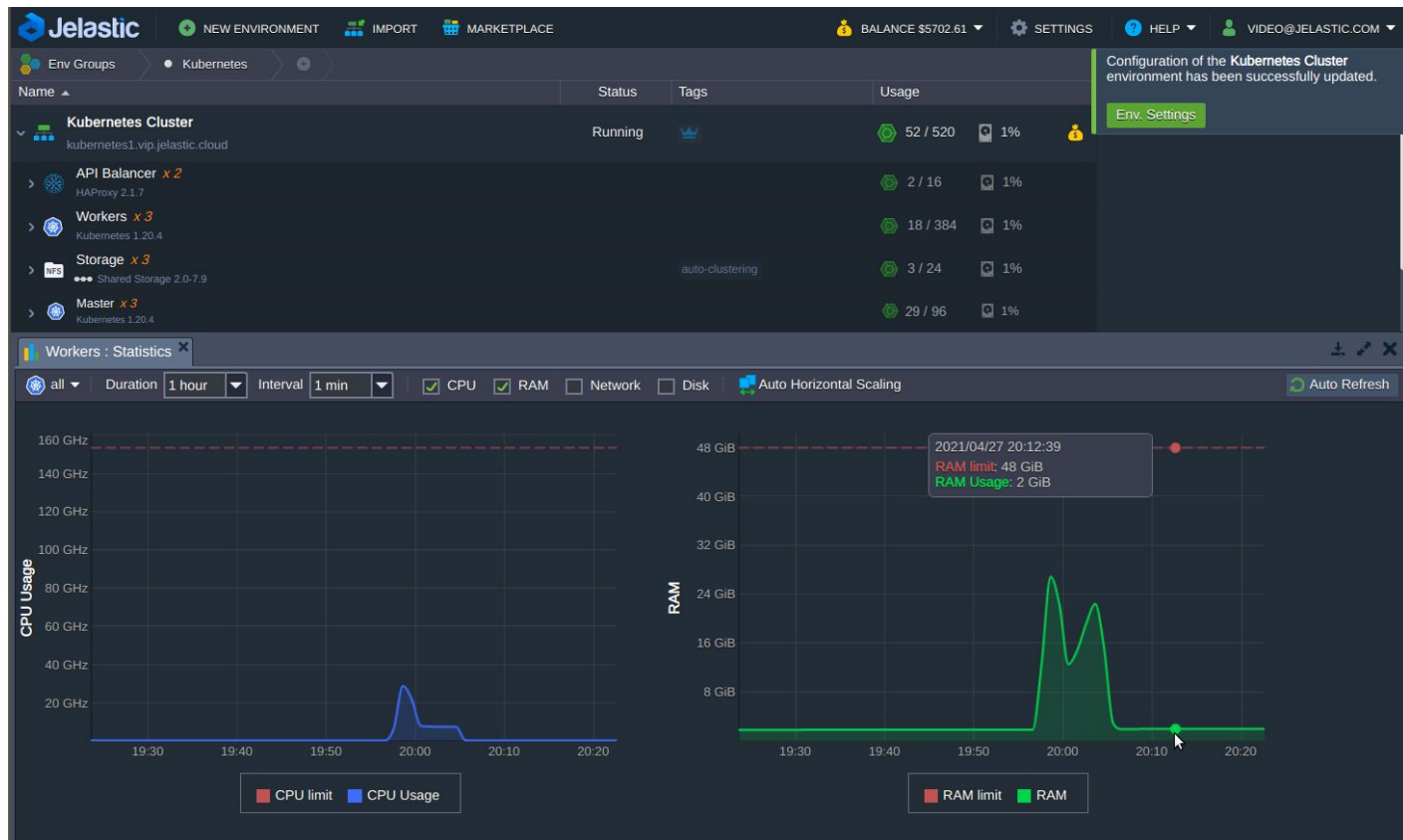
<https://www.youtube.com/watch?v=I9H28icAIUg>

# Pre-Installed Kubernetes Components

- **Nginx, HAProxy** or **Traefik**, ingress controllers for transferring HTTP/HTTPS requests to services
- CNI plugin (powered by **Weave**) for overlay network support
- **CoreDNS** for internal names resolution
- Dynamic provisioner of persistent volumes
- **Metrics Server** for gathering stats
- Jelastic SSL for protecting ingress network
- Kubernetes **Dashboard**
- **HELM** package manager to auto-install pre-packed solutions from repositories
- Jaeger, Prometheus and Grafana



# Automatic Vertical Scaling with Pay-Per-Actual-Use Pricing



An example of Workers vertical scaling: available capacity 48 GiB vs actually used and billed 2 GiB

# Changing Worker Node Resource Limits on the Fly

The screenshot shows the configuration interface for worker nodes. On the left, there are various service icons like Balancing, Cache, SQL, NoSQL, NFS, VPS, and Build. The main area is titled 'Workers' and includes a 'Vertical Scaling per Node' section with 'Reserved' (1 cloudlet) and 'Scaling Limit' (up to 64 cloudlets) settings. Below that is 'Horizontal Scaling' with a 'Stateless' dropdown and a count of 2. The 'Kubernetes 1.16.6' section includes 'Disk Limit' (502.4 GB), 'Sequential restart delay' (0 sec), and 'Public IPv4/6' (both OFF). A 'Resources' panel on the right shows '1 cloudlet = 128 MiB + 400 MHz' and calculates 'From Reserved Cloudlets: 2 + 2 + 4 = 8' and 'To Scaling Limit: 16 + 128 + 104 = 248'. The 'Estimated Cost' is shown as 'FROM \$0.09' to 'TO \$2.727'.

The monitoring dashboards show 'CPU Usage' and 'Memory Usage' over time. The CPU usage graph shows a fluctuating green area between 0 and 0.2 cores. The memory usage graph shows a constant blue bar at 5 Gi. Below the graphs is a table of nodes.

Name	Labels	Ready	CPU requests (cores)	CPU limits (cores)	Memory requests (bytes)	Memory limits (bytes)	Age
node242526-kubernetes.vip.jelastic.cloud	beta.kubernetes.io/arch: amd64 beta.kubernetes.io/os: linux	True	220.00m (2.75%)	100.00m (1.25%)	190.00Mi (2.32%)	100.00Mi (1.22%)	a.day
node242527-kubernetes.vip.jelastic.cloud	beta.kubernetes.io/arch: amd64 beta.kubernetes.io/os: linux	True	320.00m (4.00%)	200.00m (2.50%)	290.00Mi (3.54%)	200.00Mi (2.44%)	a.day
node242529-kubernetes.vip.jelastic.cloud	beta.kubernetes.io/arch: amd64 beta.kubernetes.io/os: linux	True	870.00m (21.75%)	100.00m (2.50%)	240.00Mi (5.86%)	440.00Mi (10.74%)	a.day
node242528-kubernetes.vip.jelastic.cloud	beta.kubernetes.io/arch: amd64 beta.kubernetes.io/os: linux	True	670.00m (16.75%)	100.00m (2.50%)	100.00Mi (2.44%)	100.00Mi (2.44%)	a.day
node242525-kubernetes.vip.jelastic.cloud	beta.kubernetes.io/arch: amd64 beta.kubernetes.io/os: linux	True	670.00m (16.75%)	100.00m (2.50%)	100.00Mi (2.44%)	100.00Mi (2.44%)	a.day

# Access to Worker and Master Nodes Via Web SSH

The screenshot displays a cloud management interface for a Kubernetes cluster. The cluster is named 'Kubernetes Cluster' and is in a 'Running' state. It consists of several components: API Balancer (x2), Workers (x3), Storage (x3), and Master (x3). The cluster is auto-clustering. The interface shows resource usage for the cluster and its components, including CPU and RAM usage. A 'Web SSH' button is highlighted, leading to a terminal window for a master node.

**Kubernetes Cluster** Running

- API Balancer x2 (HAProxy 2.1.7) 2 / 16 CPU 1%
- Workers x3 (Kubernetes 1.20.4) 16 / 384 CPU 1%
- Storage x3 (Shared Storage 2.0-7.9) 3 / 24 CPU 1%
- Master x3 (Kubernetes 1.20.4) RAM 29%, CPU 2%, 1%

auto-clustering

Web SSH

```
root@node296590-kubernetes1 ~ $ kubectl get nodes -o wide
NAME                                STATUS    ROLES    AGE   VERSION   INTERNAL-IP   EXTERNAL-IP   OS-IMAGE             KERNEL-VERSION
node296586-kubernetes.vip.jelastic.cloud Ready    control-plane,master 46m   v1.20.4   172.25.2.84   <none>        CentOS Linux 7 (Core) 3.10.0
node296587-kubernetes.vip.jelastic.cloud Ready    <none>    44m   v1.20.4   172.25.2.83   <none>        CentOS Linux 7 (Core) 3.10.0
node296588-kubernetes.vip.jelastic.cloud Ready    control-plane,master 45m   v1.20.4   172.25.2.85   <none>        CentOS Linux 7 (Core) 3.10.0
node296589-kubernetes.vip.jelastic.cloud Ready    <none>    44m   v1.20.4   172.25.2.91   <none>        CentOS Linux 7 (Core) 3.10.0
node296590-kubernetes.vip.jelastic.cloud Ready    control-plane,master 46m   v1.20.4   172.25.2.87   <none>        CentOS Linux 7 (Core) 3.10.0
node296610-kubernetes.vip.jelastic.cloud Ready    <none>    27m   v1.20.4   172.25.2.99   <none>        CentOS Linux 7 (Core) 3.10.0
root@node296590-kubernetes1 ~ $
```

# Upgrade Procedure

The screenshot displays the management console for a Kubernetes Cluster. The cluster is identified as 'kubernetes.vip.jelastic.cloud' and is currently in a 'Redeploying...' state. The main configuration area lists several components: 'API Balancer x 2' (HAProxy 2.1.7), 'Workers x 3' (Kubernetes 1.20.4), 'Storage x 3' (Shared Storage 2.0-7.9), and 'Master x 3' (Kubernetes 1.20.4). An 'Add-Ons' tab is active, showing two primary options: 'Cluster Upgrade' and 'Cluster Configuration'. The 'Cluster Upgrade' option is highlighted with a red border and includes a 'Start Cluster Upgrade' button. The 'Cluster Configuration' option includes buttons for 'Remote API', 'Storage', and 'Jaeger'.

**Kubernetes Cluster** Redeploying...

kubernetes.vip.jelastic.cloud

- API Balancer x 2  
HAProxy 2.1.7
- Workers x 3  
Kubernetes 1.20.4
- Storage x 3  
Shared Storage 2.0-7.9
- Master x 3  
Kubernetes 1.20.4

auto-clustering

Add-Ons

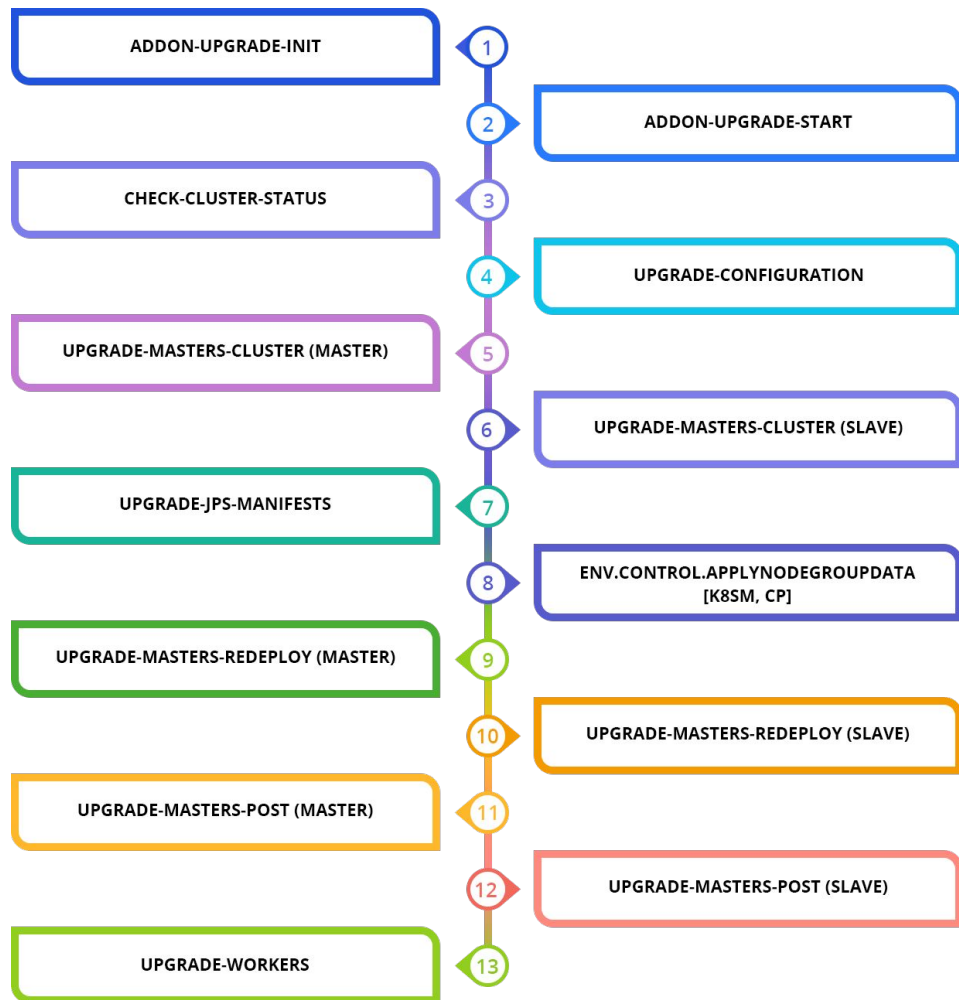
Master : Add-Ons

**Cluster Upgrade**  
Upgrade Kubernetes cluster to a newer version  
[Start Cluster Upgrade](#)

**Cluster Configuration**  
Configure remote API access and install complementary tools  
[Remote API](#) [Storage](#) [Jaeger](#)


















# Upgrade Procedure

- Check if the cluster is eligible to upgrade, and availability of final version(s)
- Upgrade installed cluster components (Weave, ingresses, dashboards, hello-world, metrics-server, Helm, Prometheus+Grafana, etc.)
- Check if deprecated components are present in the cluster
- Upgrade master instances one-by-one via redeploy
- Evict PODs, upgrade worker instances one-by-one via redeploy



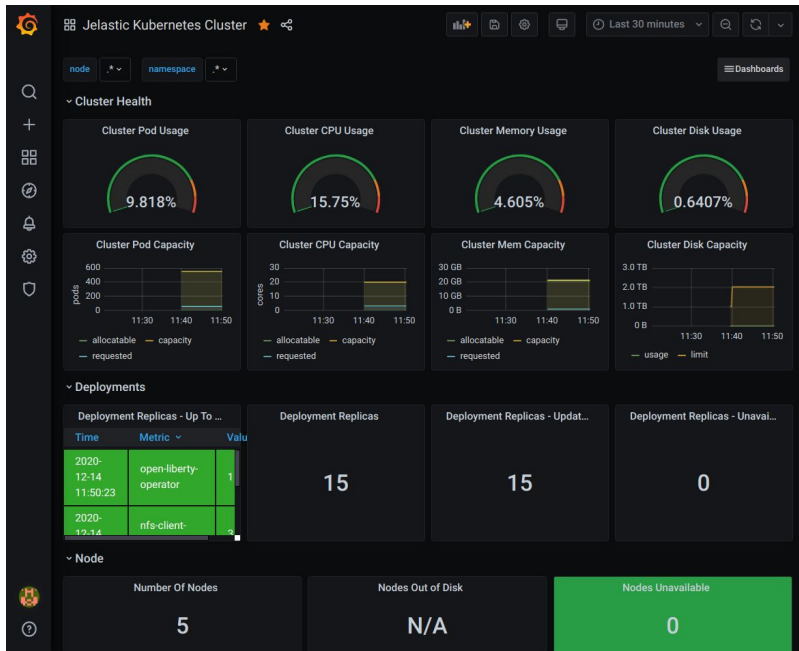


# Built-In Add-Ons

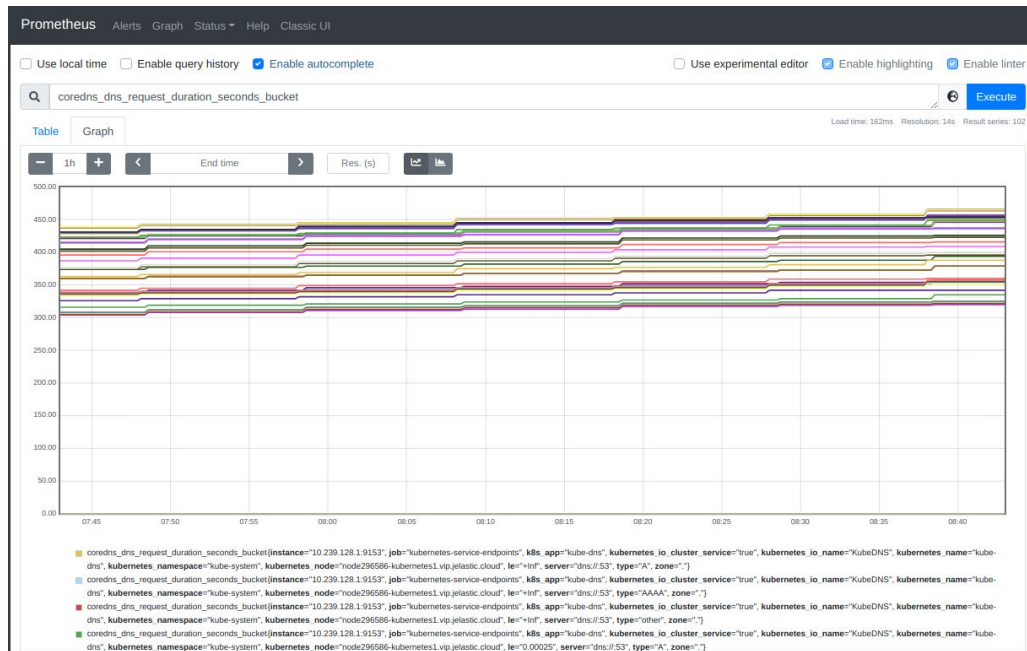
  <b>DockerHub Registry Credentials</b> <p>Leverage DockerHub images pull rate limits; assign DockerHub user credentials to Kubernetes deployments cluster-wide</p> <p><a href="#">DockerHub Credentials</a></p>	  <b>Rancher Installer</b> <p>Rancher Management Platform</p> <p><a href="#">Rancher Platform Installation</a></p>	  <b>Certificate Manager</b> <p>Kubernetes SSL Certificate Manager allows to bind custom domain names to the cluster and manage SSL certificates</p> <p><a href="#">Install Certificate Manager</a></p>	  <b>GitLab Integration</b> <p>Add Kubernetes GitLab integrations</p> <p><a href="#">Configure</a> <a href="#">Remove Integration</a></p>
  <b>Cluster Monitoring</b> <p>Install cluster monitoring components (Prometheus and Grafana)</p> <p><a href="#">Install Monitoring Tools</a></p>	  <b>Cluster Upgrade</b> <p>Upgrade Kubernetes cluster to a newer version</p> <p><a href="#">Start Cluster Upgrade</a></p>	  <b>Cluster Configuration</b> <p>Configure remote API access and install complementary tools</p> <p><a href="#">Remote API</a> <a href="#">Storage</a> <a href="#">Jaeger</a></p>	 <b>FTP</b> <p>FTP/FTPS protocol support for deploying your applications to the platform.</p> <p><a href="#">Install</a></p>
 <b>Env Start/Stop Scheduler</b> <p>The scheduler for automatic environment hibernation and wake-up</p> <p><a href="#">Install</a></p>	 <b>TimeZone Change</b> <p>TimeZone Change Tool</p> <p><a href="#">Install</a></p>		

# Embedded Cluster Monitoring

## Grafana

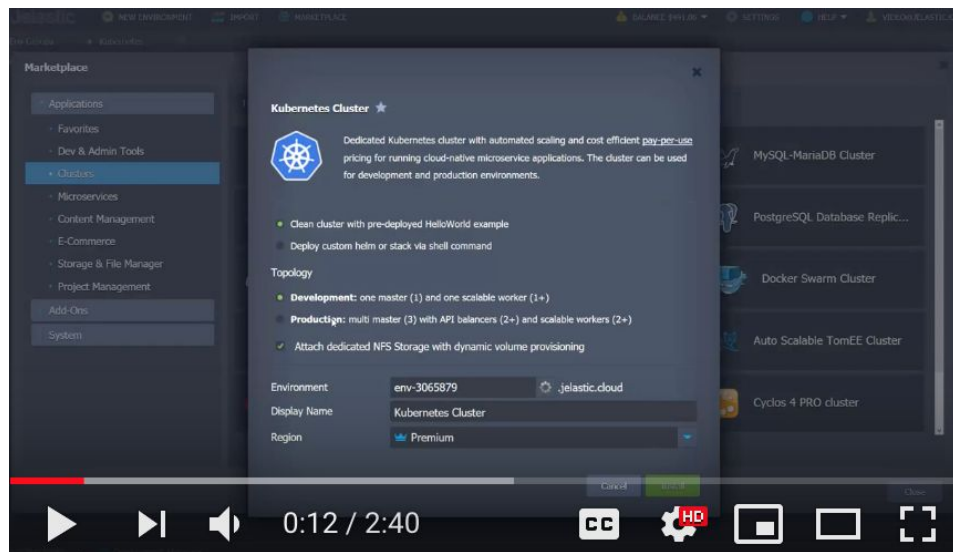


## Prometheus



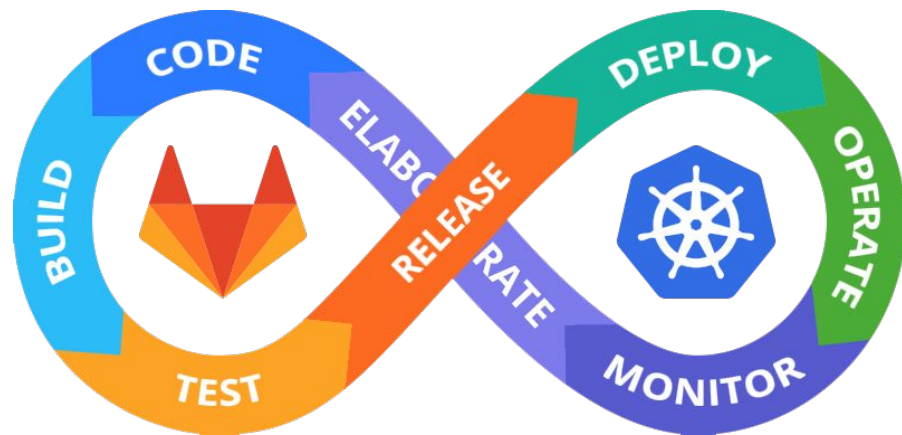
# Kubernetes Issues Solved by Jelastic

- Challenging setup is converted to “one click”
- Manual nodes configuration is fully automated
- Out of the box LB and SSL support
- K8s metrics and monitoring solutions pre installed
- Replacing VMs with system containers
  - “Pay-Per-Actual-Use” pricing
  - Fast scaling of K8s nodes
- Turnkey solution for Public Hosting Business
- Multi-cluster and multi-cloud management
- Built-in billing and monitoring tools
- Product and security updates automation



# GitLab CI/CD Pipeline Integrated with Kubernetes

- Built-in CI/CD for creating a pipeline and controlling the application delivery lifecycle, from uploading the code to the repository, up to deployment to production
- AutoDevOps helps to establish a CI/CD pipeline that automatically detects, builds, tests, and deploys the projects



<https://jelastic.com/blog/kubernetes-gitlab-ci-cd-integration/>

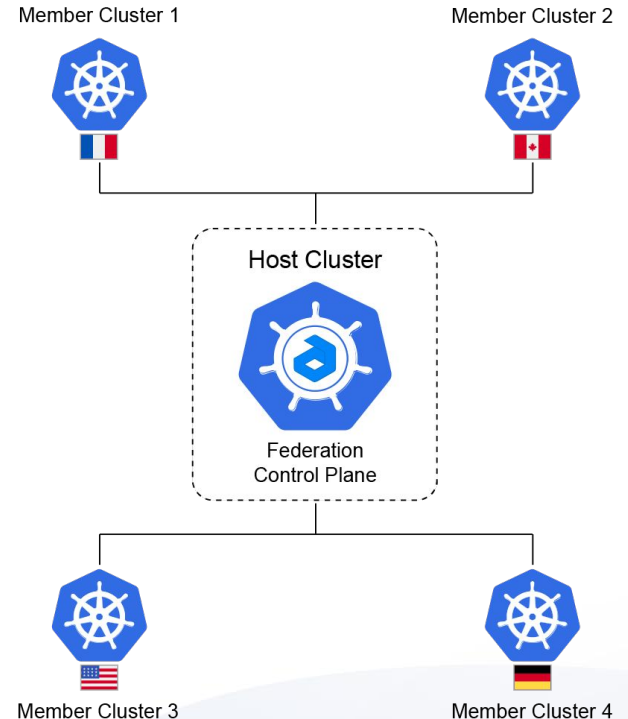
# Multi-Region Kubernetes Cluster Federation

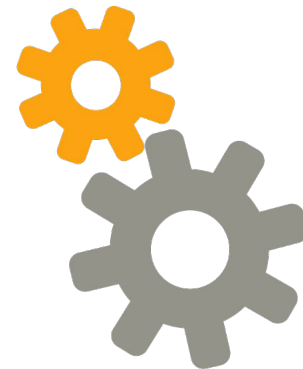
Kubernetes Federation is a multi-cloud or multi-region implementation for centralized deployment and management of applications and services across multiple Kubernetes clusters.

[Multi-Region Kubernetes Cluster Federation in Jelastic PaaS](#)

[How to Federate Resources across Kubernetes Clusters for Unified Deployment](#)

The screenshot displays the Jelastic dashboard for two federated Kubernetes clusters. The top cluster, 'Kubernetes Cluster' (fedhost.vip.jelastic.cloud), is in a 'Running' state with 14/72 resources and 1% usage. It contains three sub-resources: 'Workers' (5/32), 'Storage' (1/8), and 'Master' (8/32). The bottom cluster, 'Kubernetes Cluster' (member1.demo.jelastic.com), is also in a 'Running' state with 11/72 resources and 1% usage, containing 'Workers' (4/32), 'Storage' (1/8), and 'Master' (6/32).





GET STARTED WITH **30 DAYS** FOR FREE

<https://www.beebyte.se/platform-as-a-service-paas/>

# Give a Try Yourself

<https://jelastic.com/kubernetes-hosting/>

Contact for Partnership  
and Assistance

[info@jelastic.com](mailto:info@jelastic.com)

