Jelastic
A DIVISION OF Virtuozzo

beebyte

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

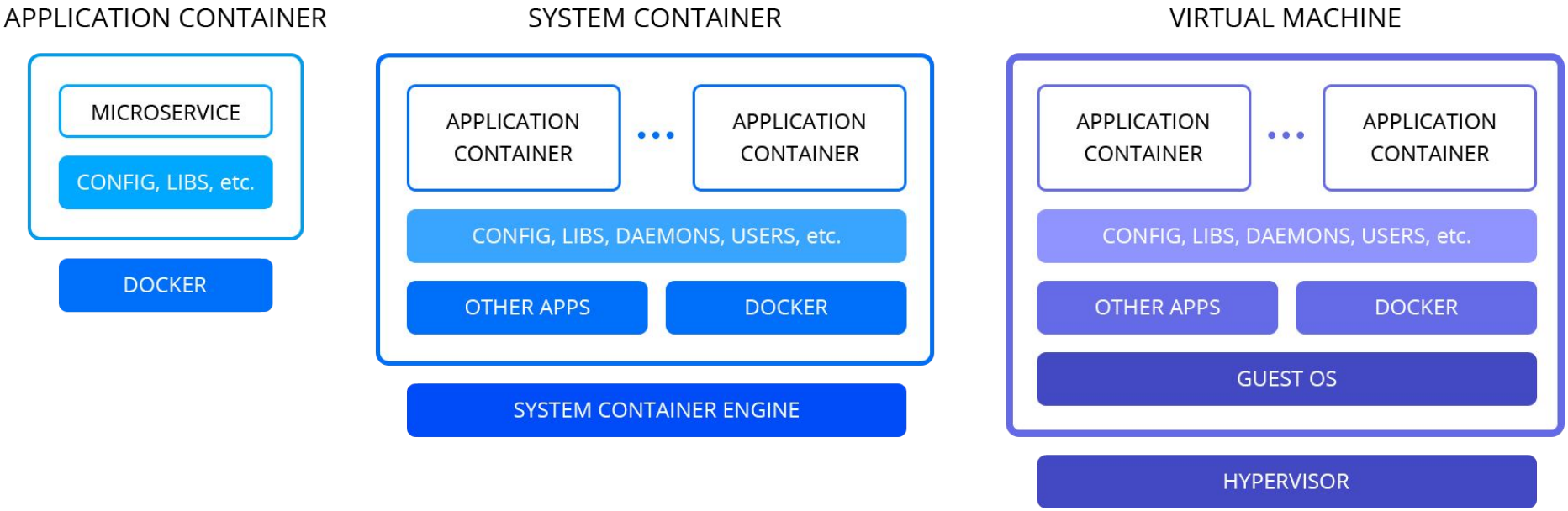
beebyte

Tetiana Fydorenchyk
Director of Product Marketing

Virtuozzo

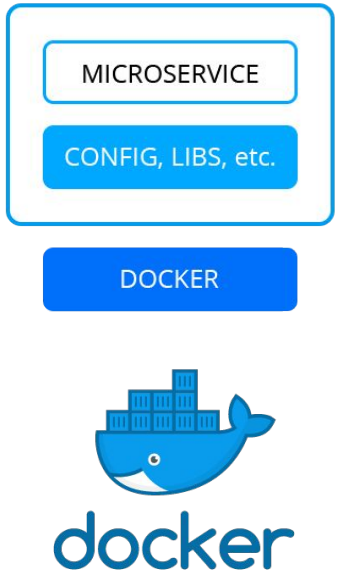
VIRTUALIZATION

Virtualization Types

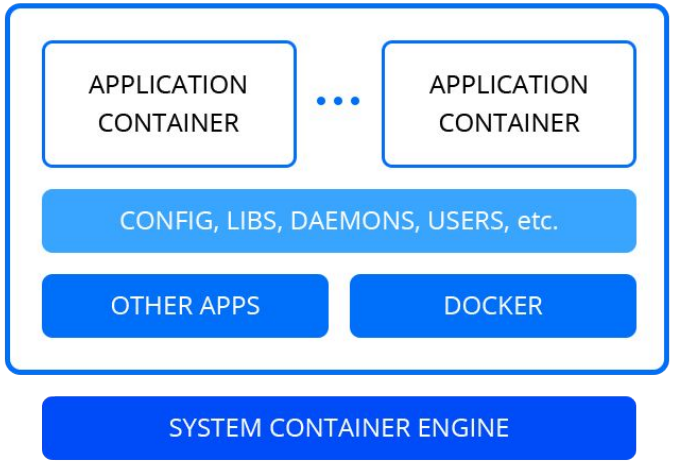


Virtualization Types

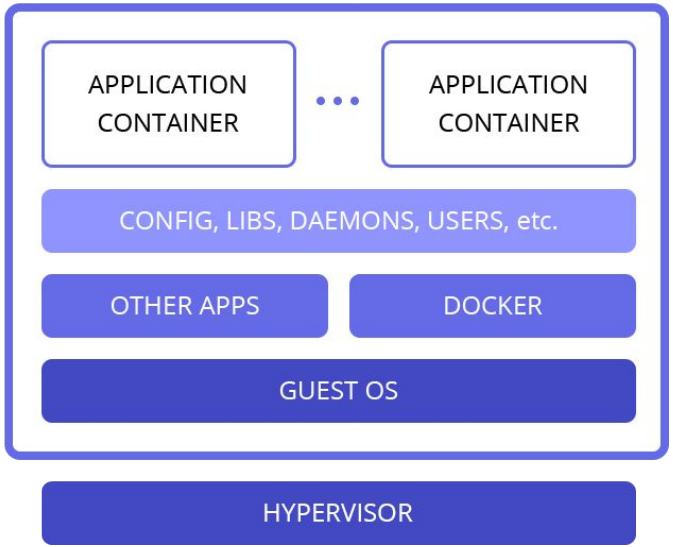
APPLICATION CONTAINER



SYSTEM CONTAINER

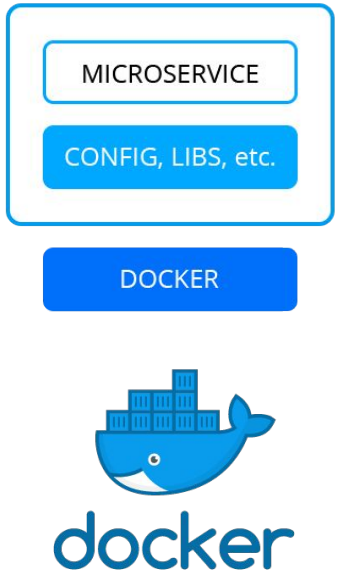


VIRTUAL MACHINE

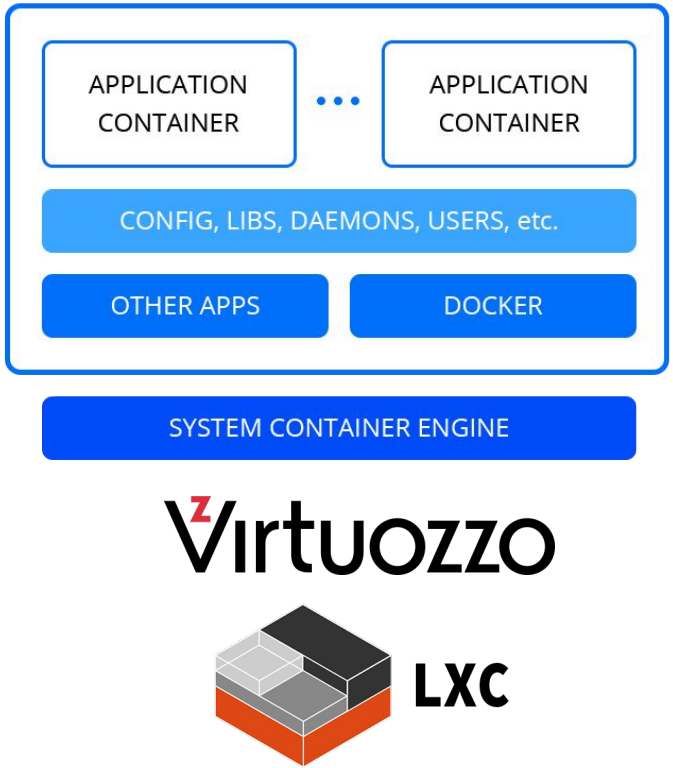


Virtualization Types

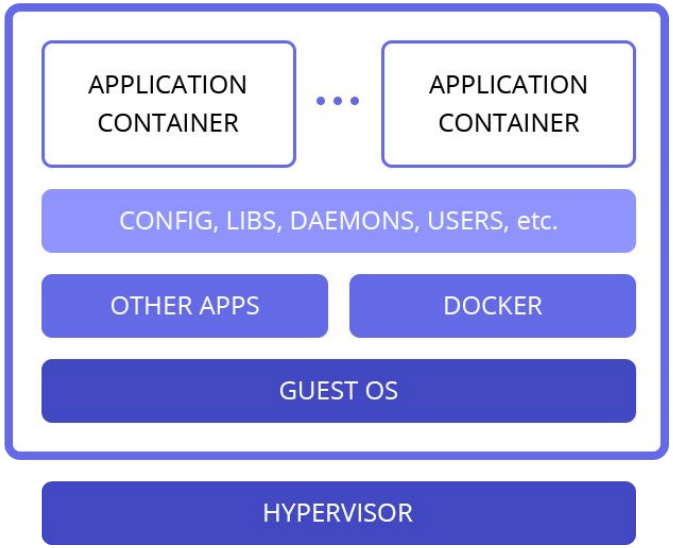
APPLICATION CONTAINER



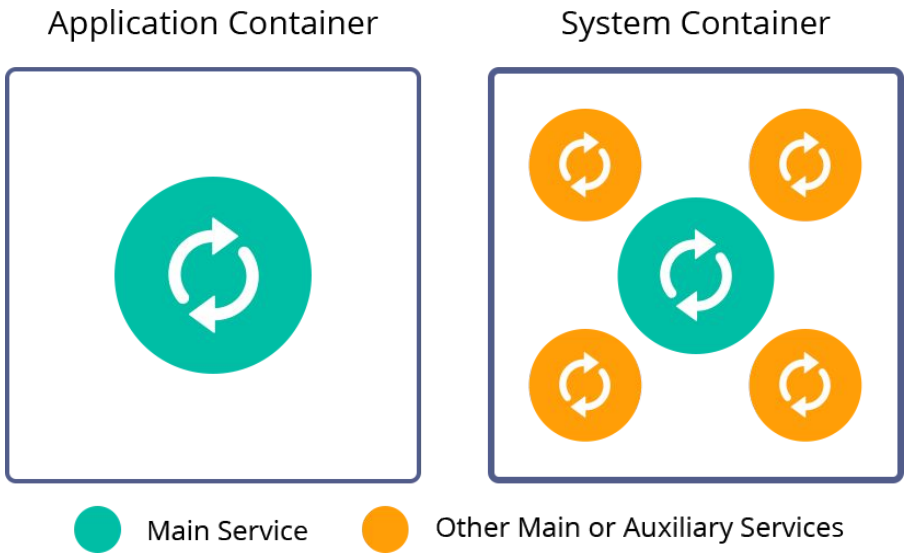
SYSTEM CONTAINER



VIRTUAL MACHINE



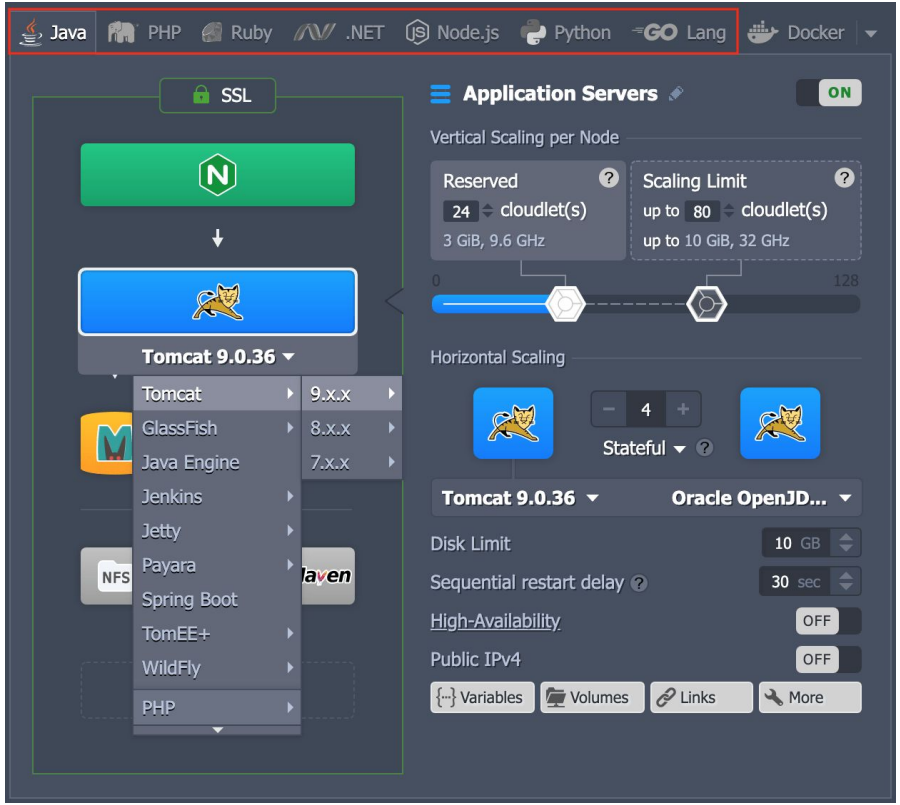
Container Types



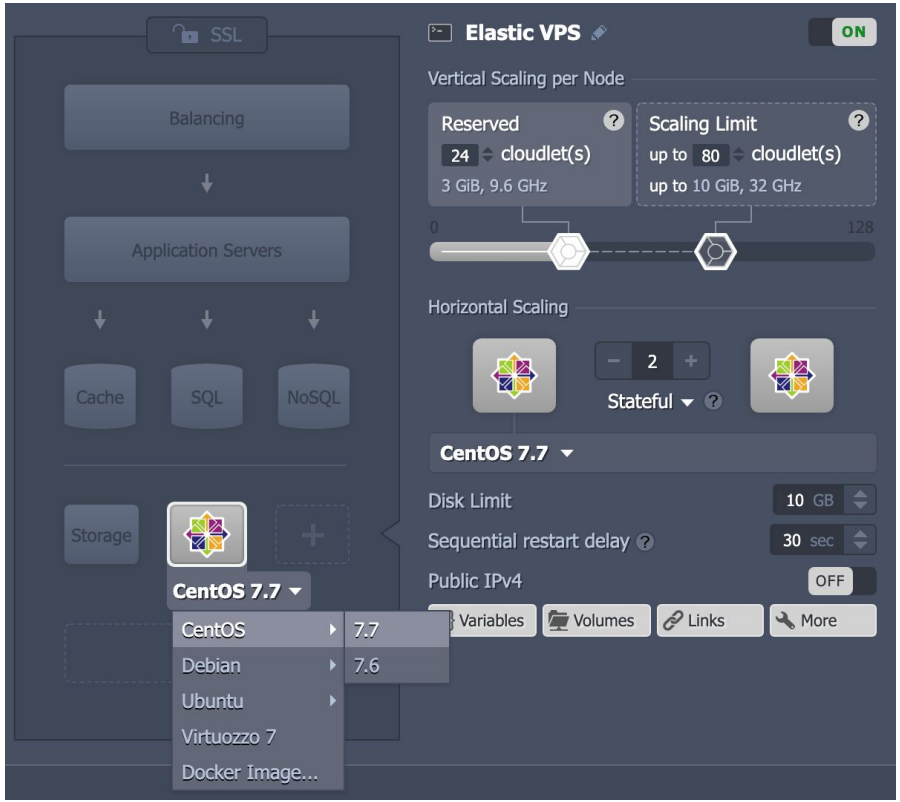
- Used to run for a single service
- Layered filesystems
- Examples: Docker, CRI-O

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

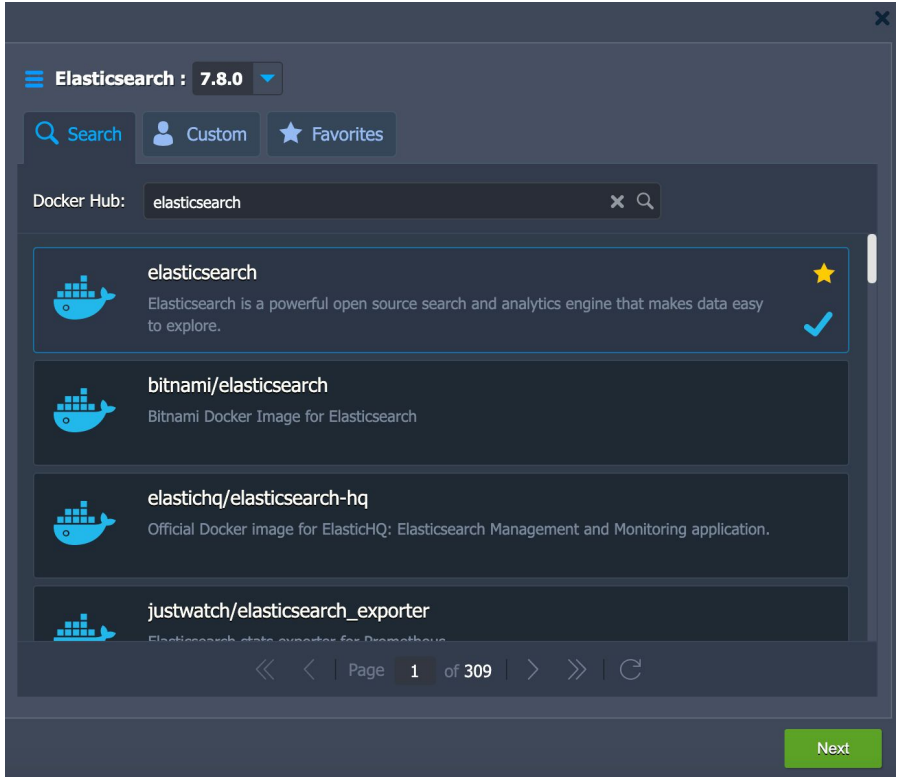
Certified Managed Containers



Virtual Private Servers (Elastic VPS)



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

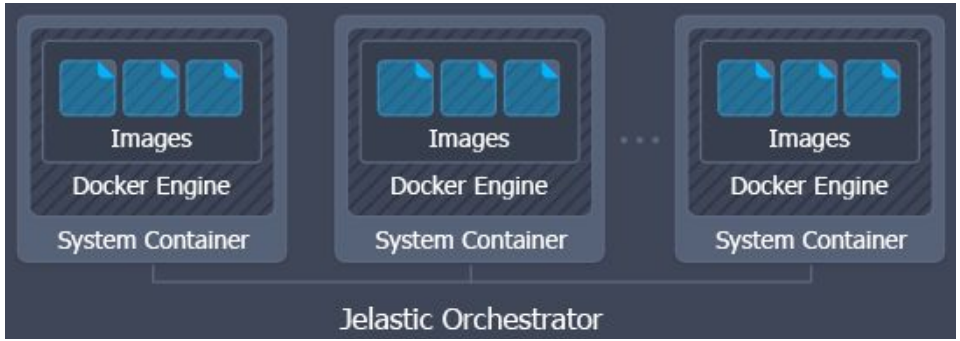
☑ Install Portainer UI and Let's Encrypt SSL certificates

Environment dockerengine .vip.jelastic.cloud

Display Name Docker Engine CE

Region Premium

Back Install



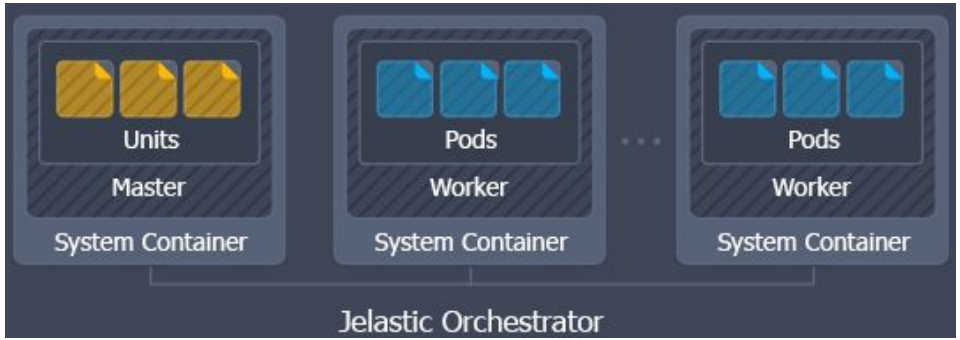
Kubernetes Cluster

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.

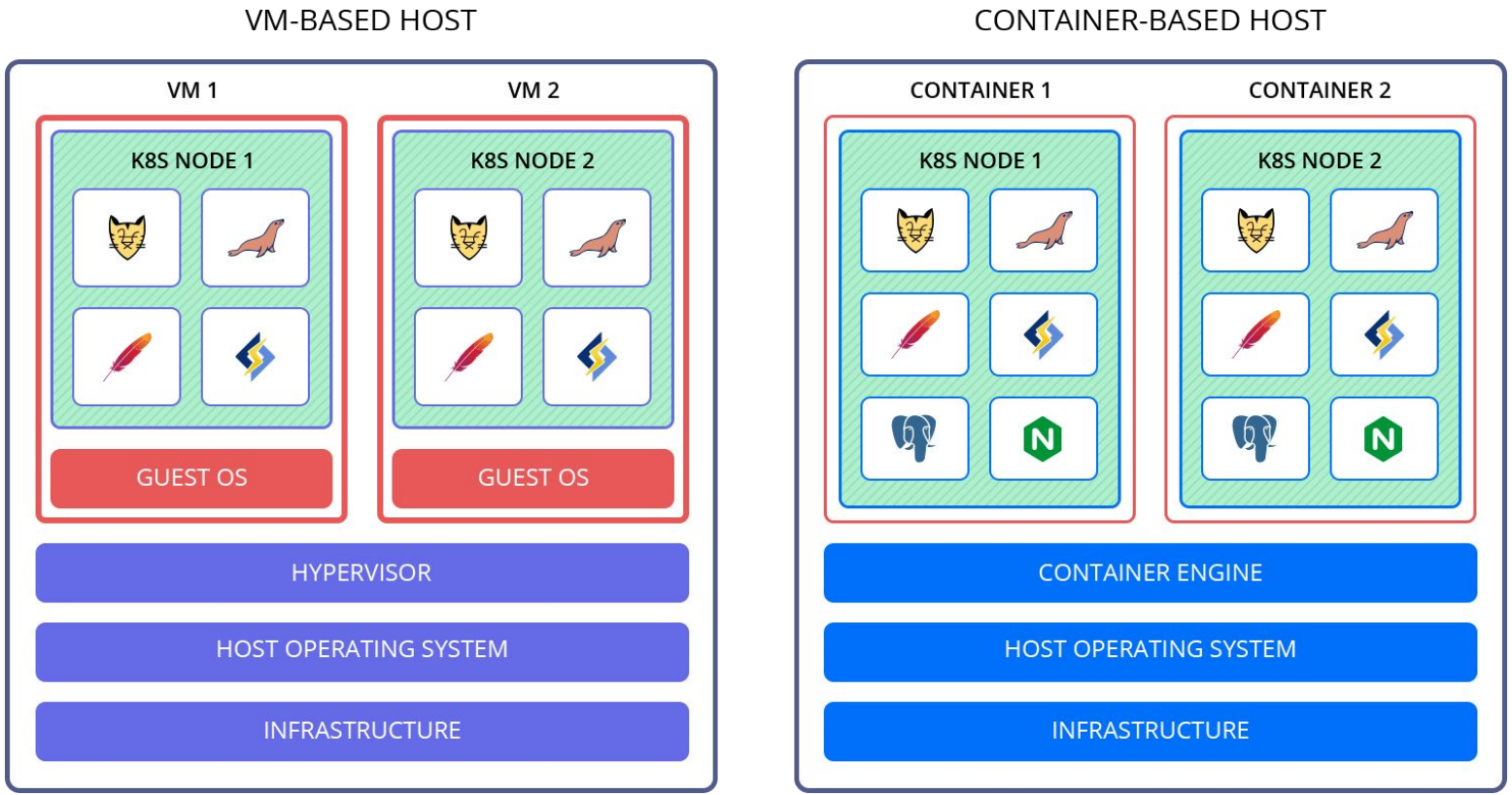
Version	v1.19.10	K8s Dashboard	v2
Topology	Development	Ingress Controller	NGINX
Deployment	Clean cluster	Custom	
NFS Storage			
Modules	Prometheus & Grafana	Jaeger Tracing Tools	
	Remote API Access		
Environment	kubernetes	.vip.jelastic.cloud	
Display Name	Kubernetes Cluster		
Region	Premium		

Back Install

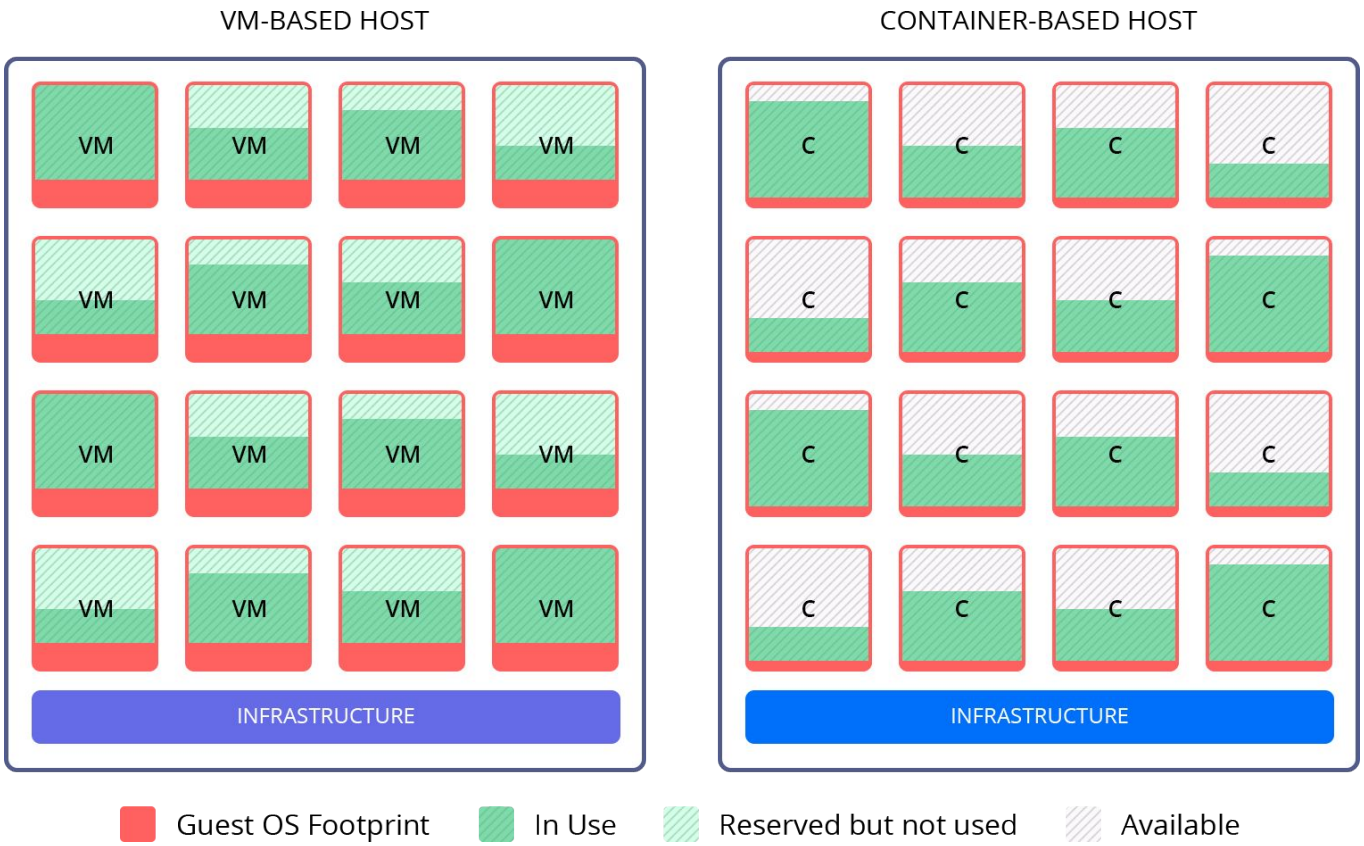


EFFECTIVE USAGE OF INFRASTRUCTURE

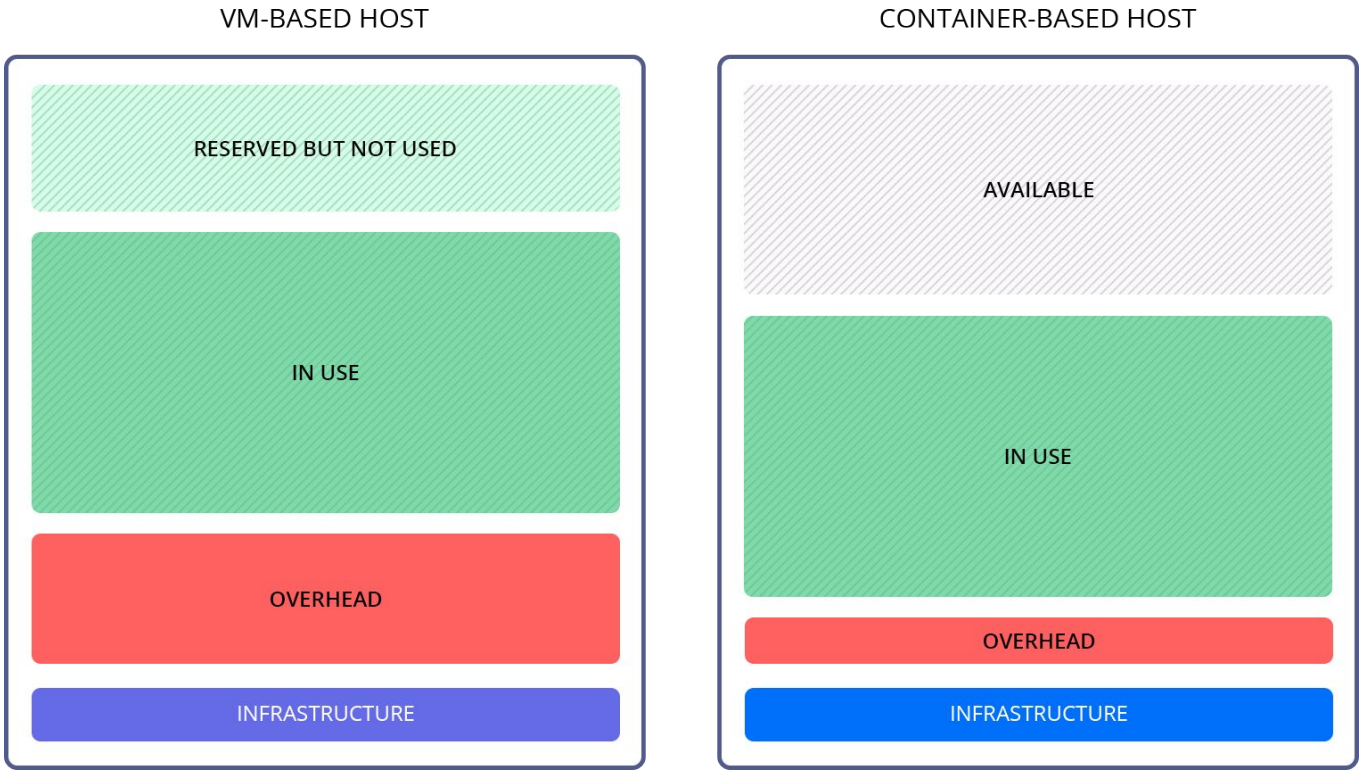
Running Kubernetes on VMs vs System Containers



Kubernetes on VMs vs System Containers

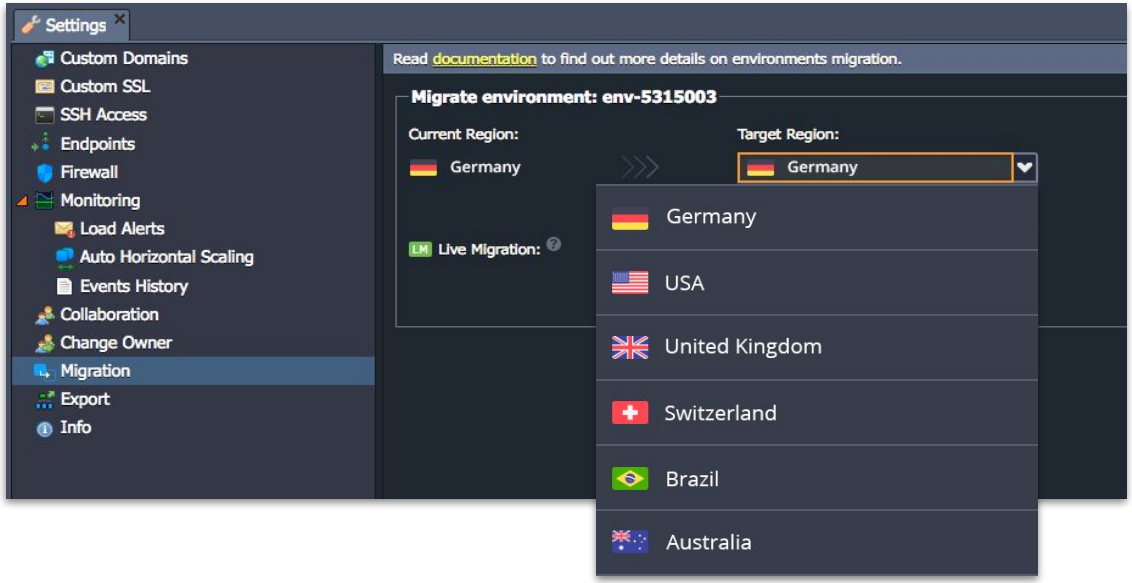


Kubernetes on VMs vs System Containers

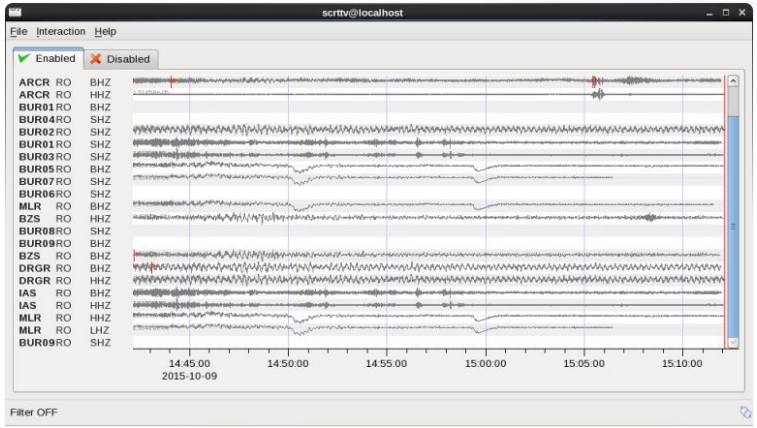


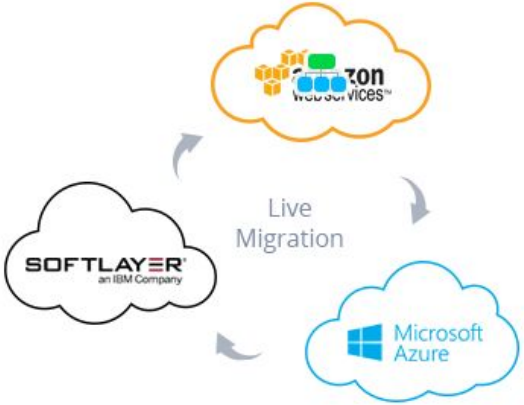
Live Migration of Containers

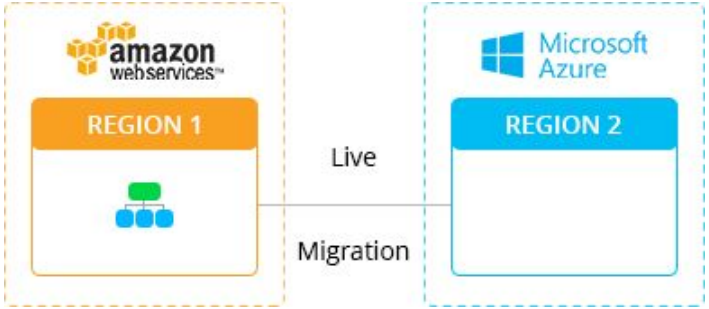
- Zero downtime hardware maintenance
- Load rebalancing across host nodes
- High-availability across clouds



Live Migration across Clouds without Downtime



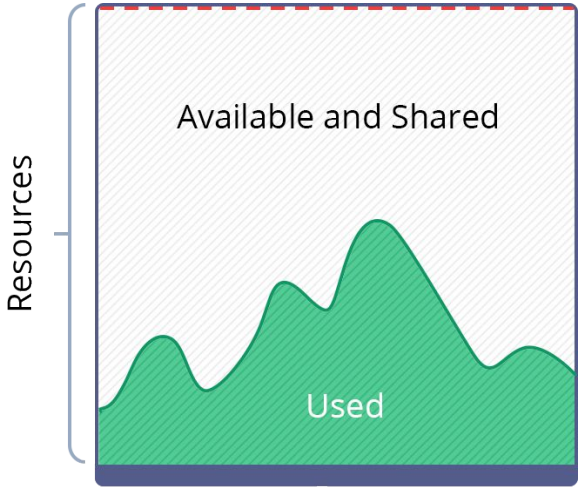




SCALING JAVA IN VMs vs CONTAINERS

Resource Limit vs Real Usage in VM and Container

VM

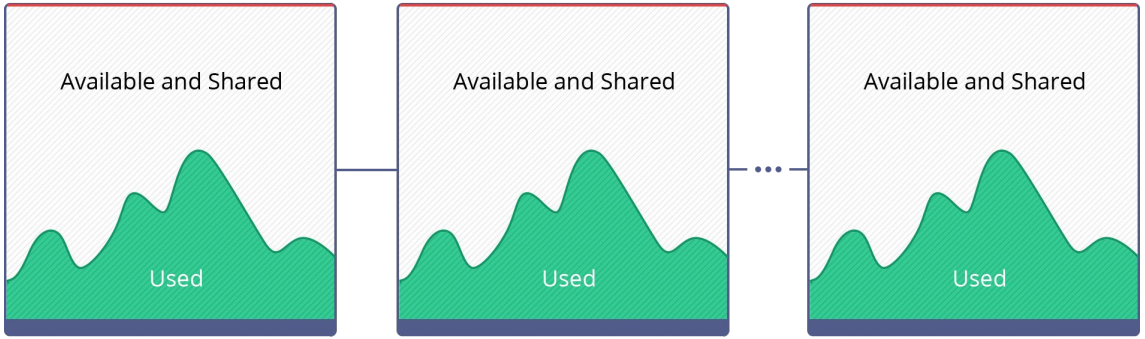


Container

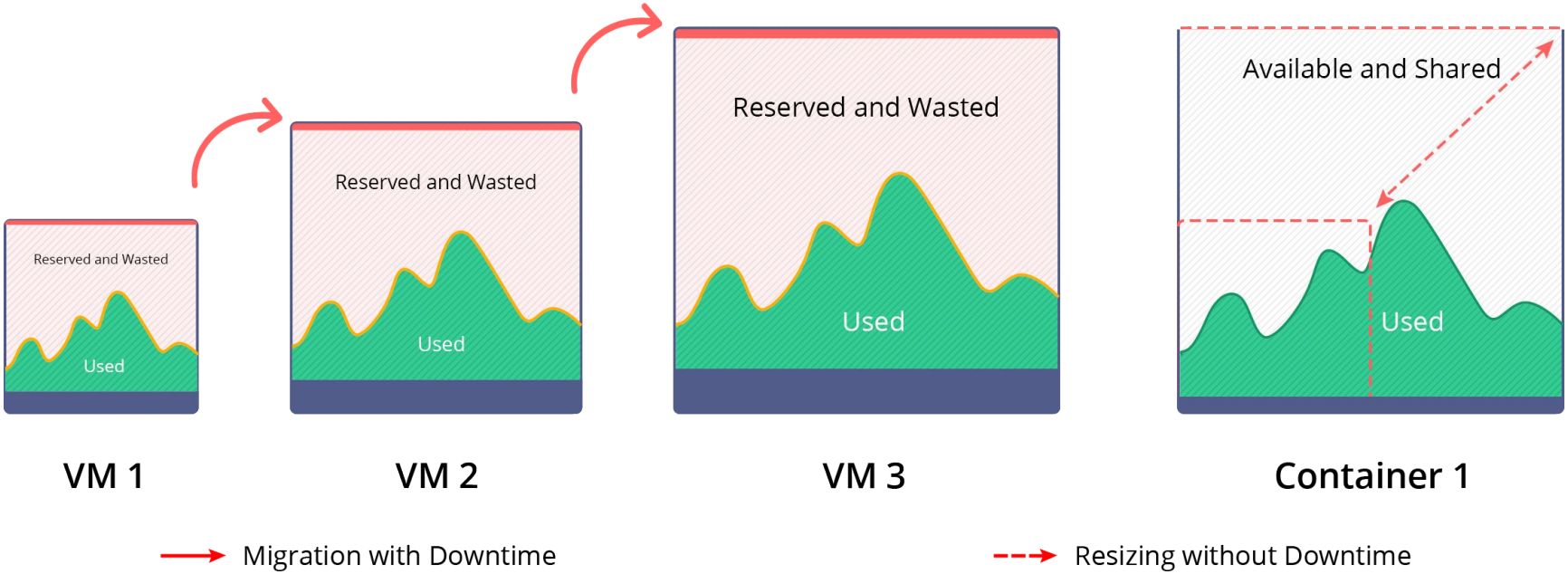
Horizontal Scaling: VMs vs Containers

VMs

Containers

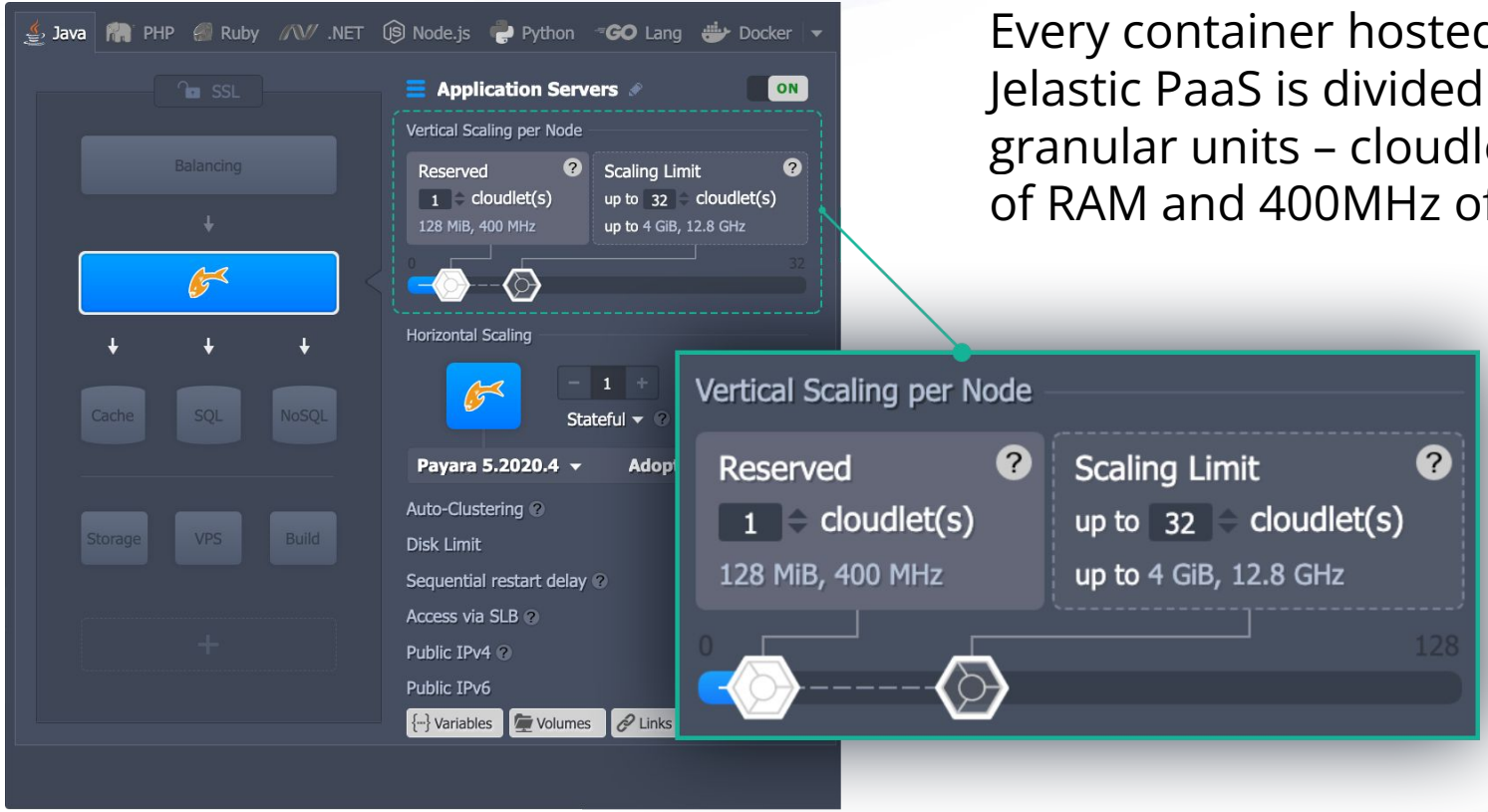


Vertical Scaling: VMs vs Containers



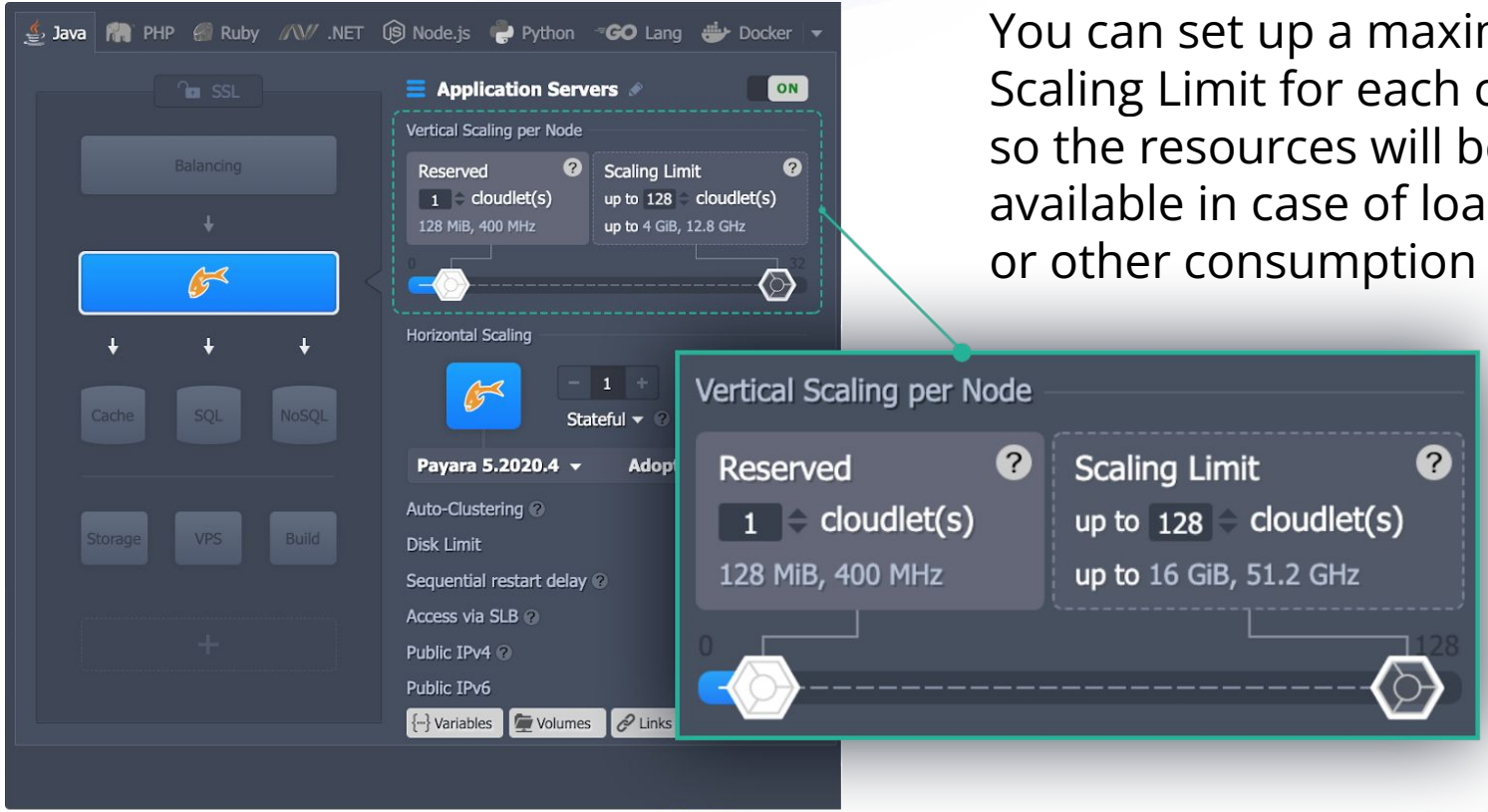
Resizing of the same container on the fly is easier, cheaper and faster than moving to a larger VM.

Automatic Vertical Scaling



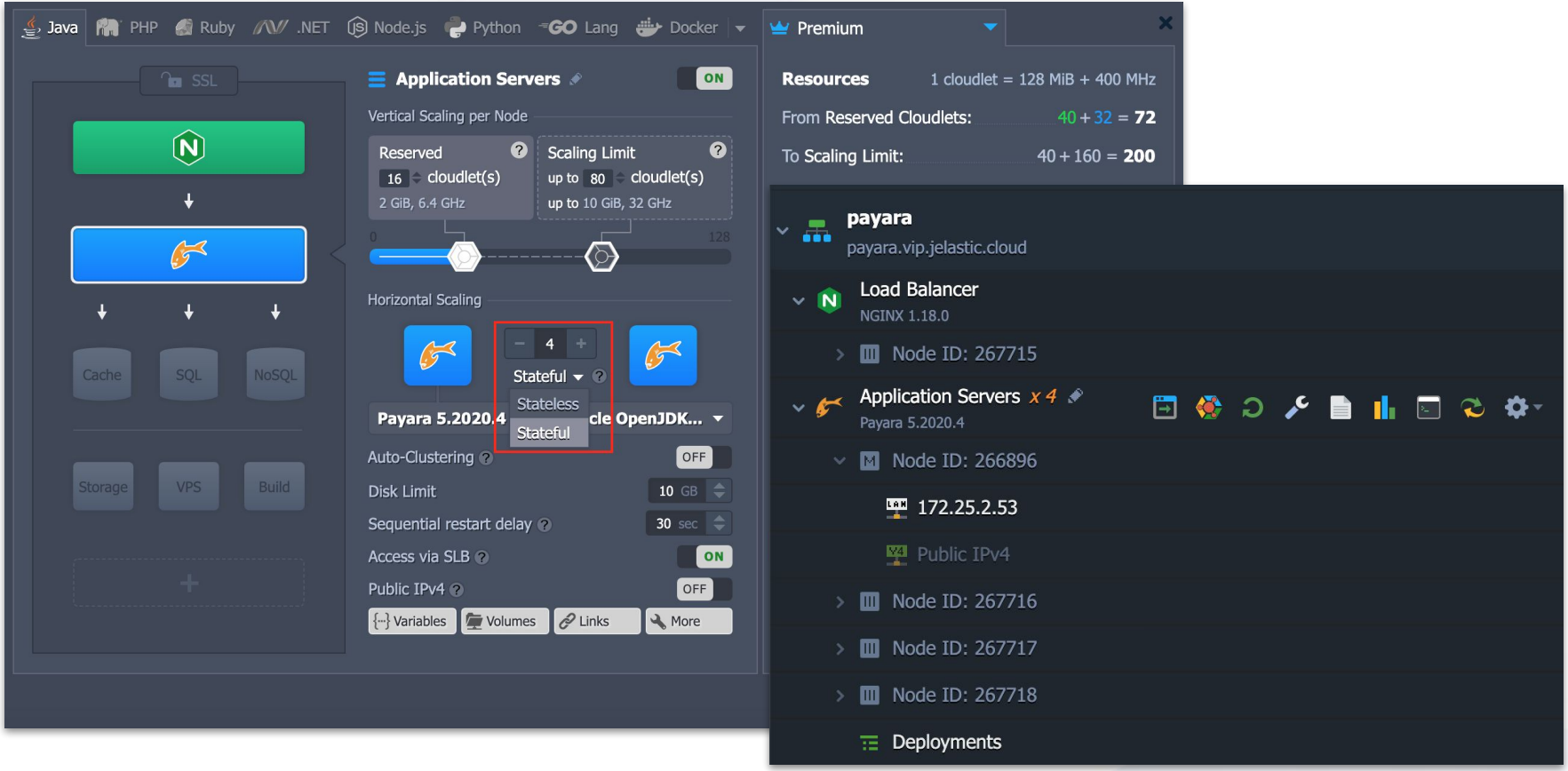
Every container hosted with Jelastic PaaS is divided into granular units – cloudlets (128MiB of RAM and 400MHz of CPU)

Automatic Vertical Scaling



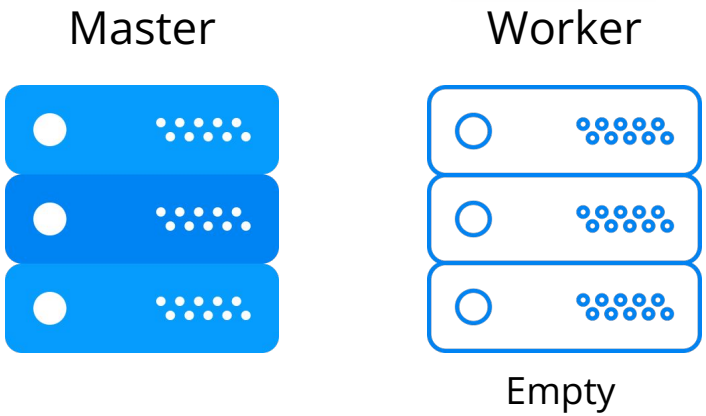
You can set up a maximum Scaling Limit for each container, so the resources will be always available in case of load spikes or other consumption changes

Horizontal Scaling



Stateless (Create New) vs Stateful (Clone)

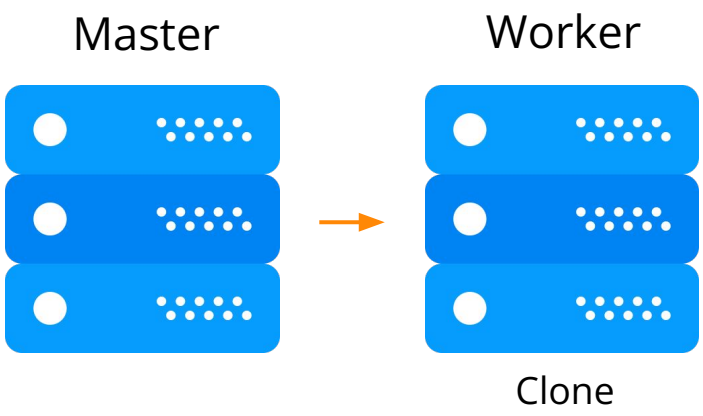
Stateless



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

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

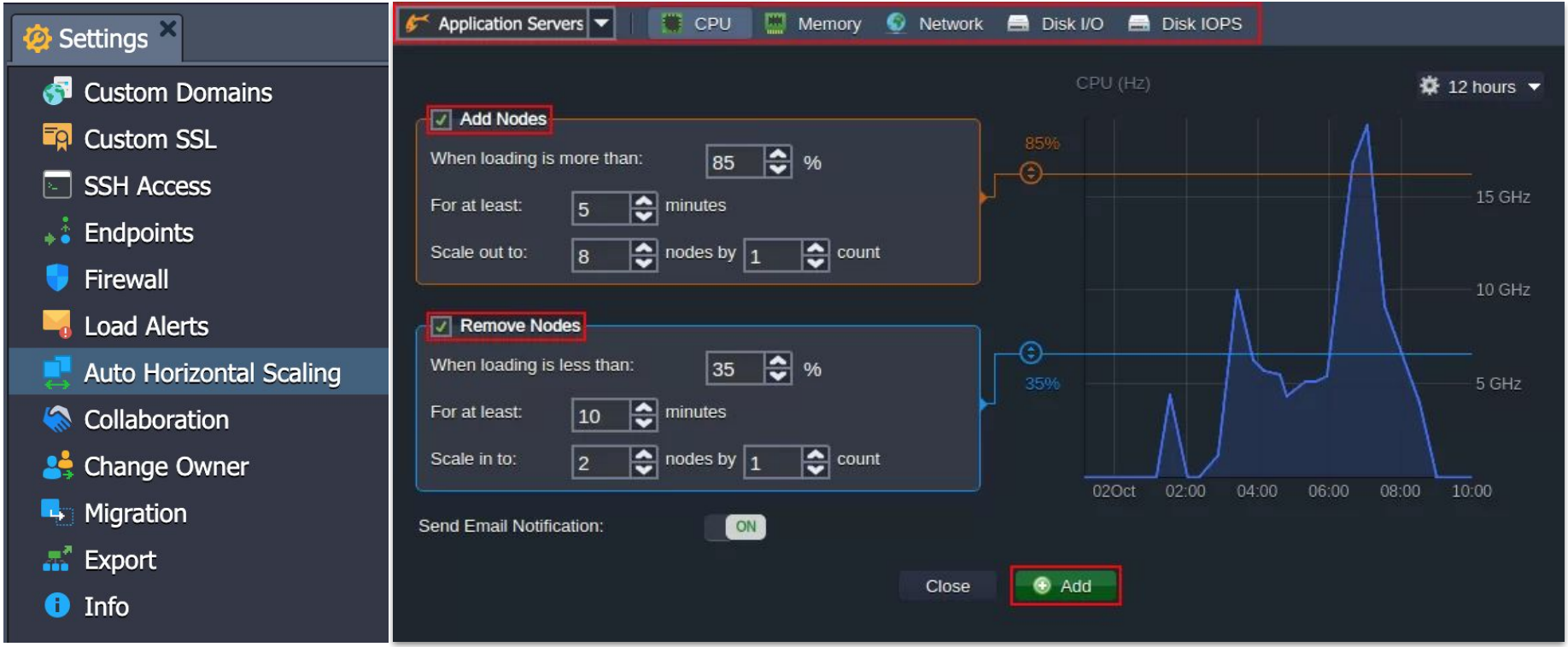
Stateful



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



Load Alerts

Application Servers : Statistics | Settings

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

Settings | History

Add | Edit | Remove | Enable | Refresh

Name	Condition
Load Balancer NGINX 1.18.0	
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	
Application Servers Tomcat 10.0.0-M9 x 3	
SQL Databases MariaDB 10.4.16 x 2	

Add Alert

Name: Alert 1

Nodes: Application Servers

Whenever: Cloudlets (Memory, CPU)

Is: > 55 %

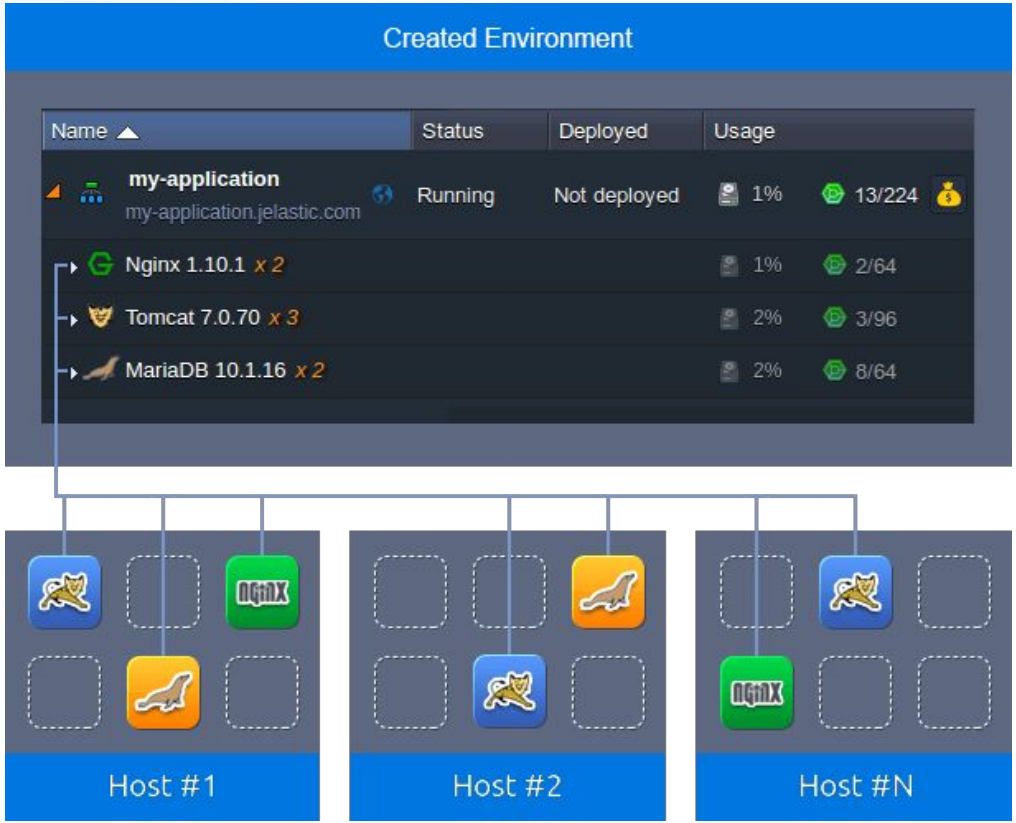
For at least: 10 minutes

Notification frequency: 1 hour(s)

Cancel | Add

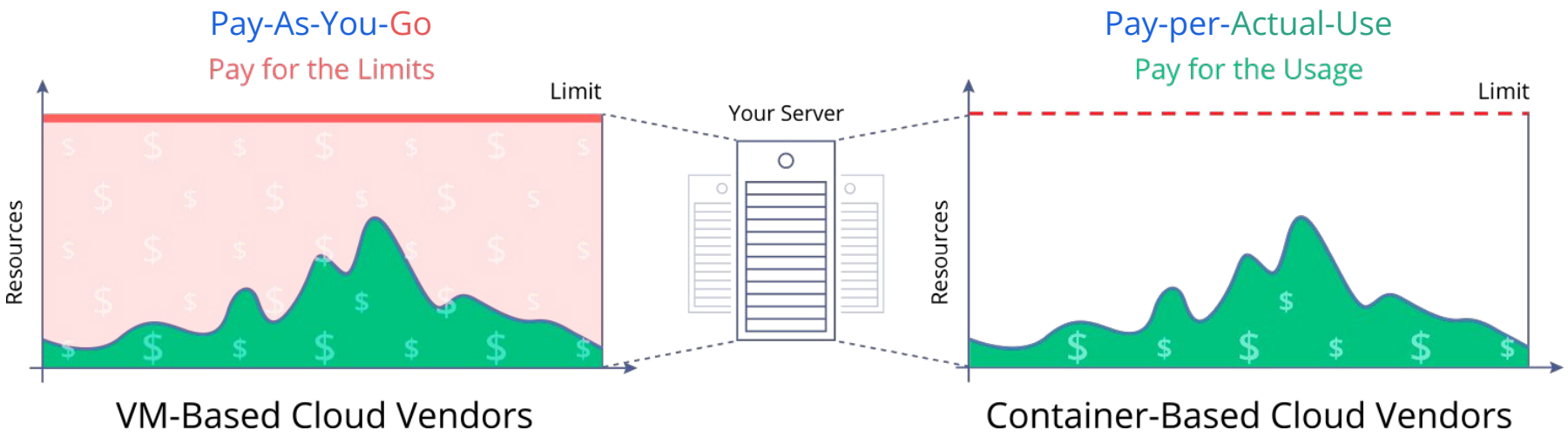
Anti-Affinity Rules

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 provides 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

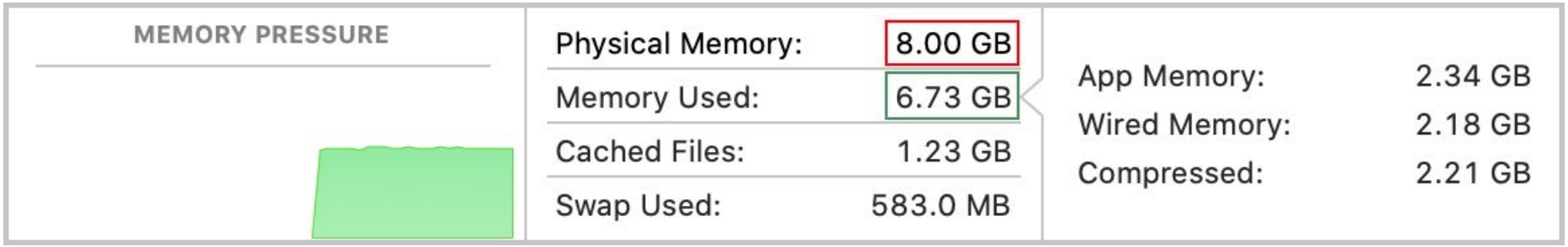
~$ *free -m*

```
tomcat@node275130-0002-test-env ~ $ free -m
              total        used        free      shared  buff/cache   available
Mem:           8192         102        8049           0          39        8088
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

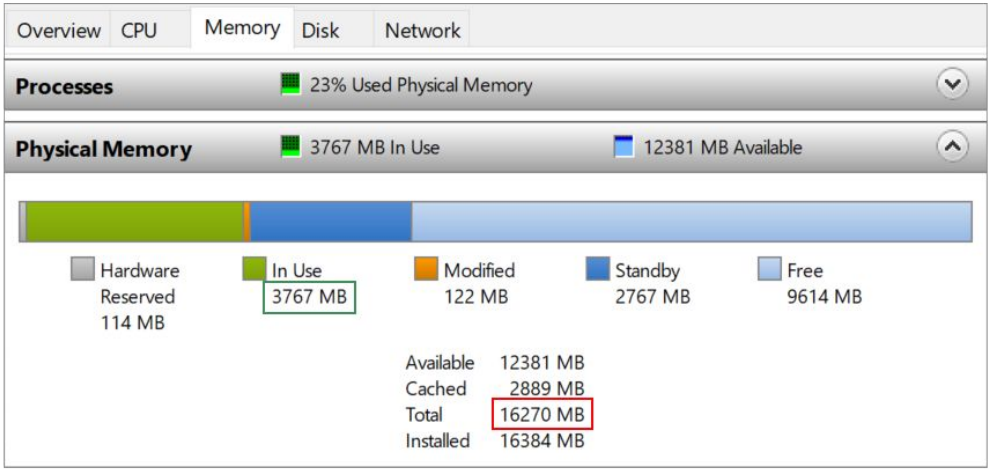
To check **RAM usage** on **Mac**, go to Activity Monitor (Applications > Utilities)



- ***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)

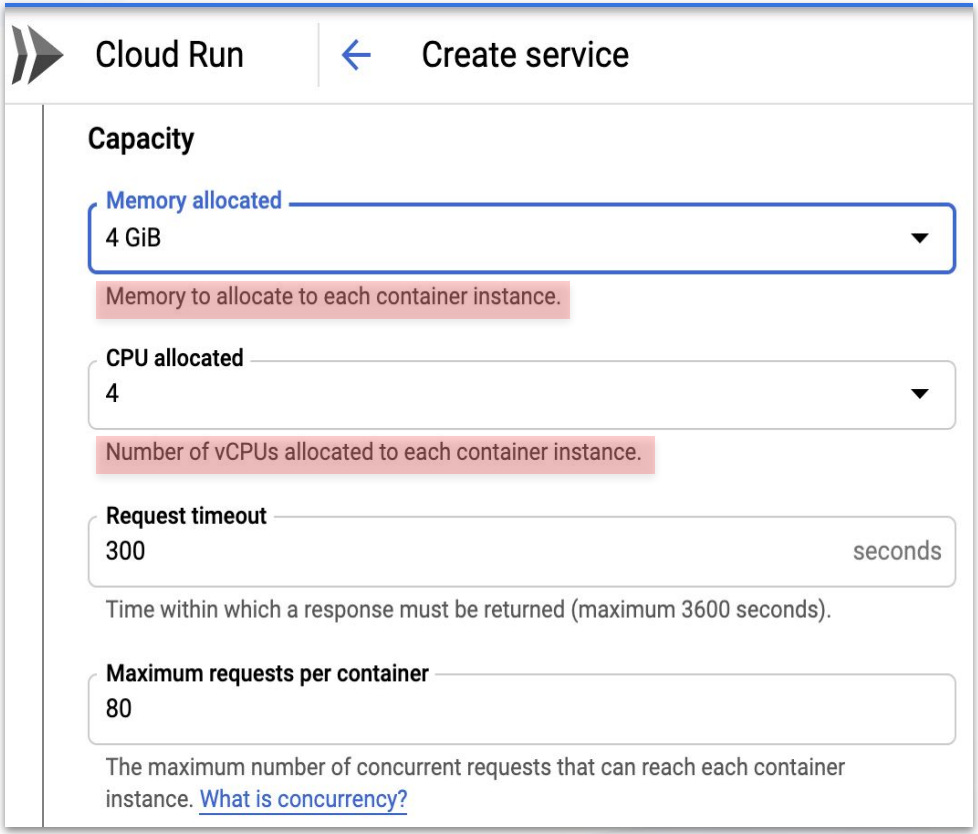
https://aws.amazon.com/fargate/pricing/

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

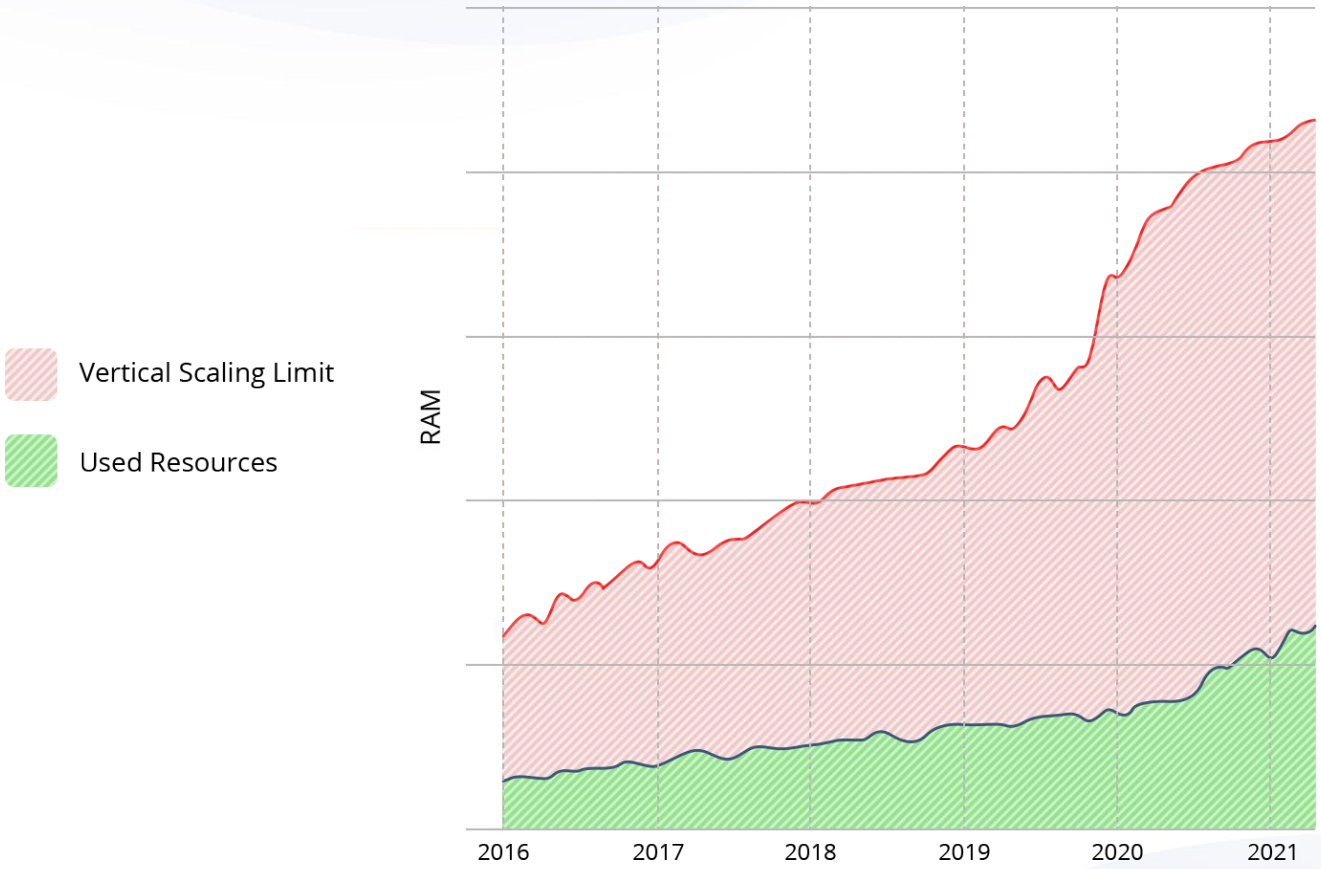
Cloud Run pricing

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



Real Statistics of Resource Consumption with Containers



KUBERNETES HOSTING

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	v1.20.4	K8s Dashboard	v2
Topology	Production	Ingress Controller	NGINX
Deployment	Clean cluster	Custom	
NFS Storage			
Modules	Prometheus & Grafana	Jaeger Tracing Tools	
	Remote API Access		
Environment	kubernetes-cluster	.vip.jelastic.cloud	
Display Name	Kubernetes Cluster		
Region	Premium		

Cancel Install

Cluster Configuration

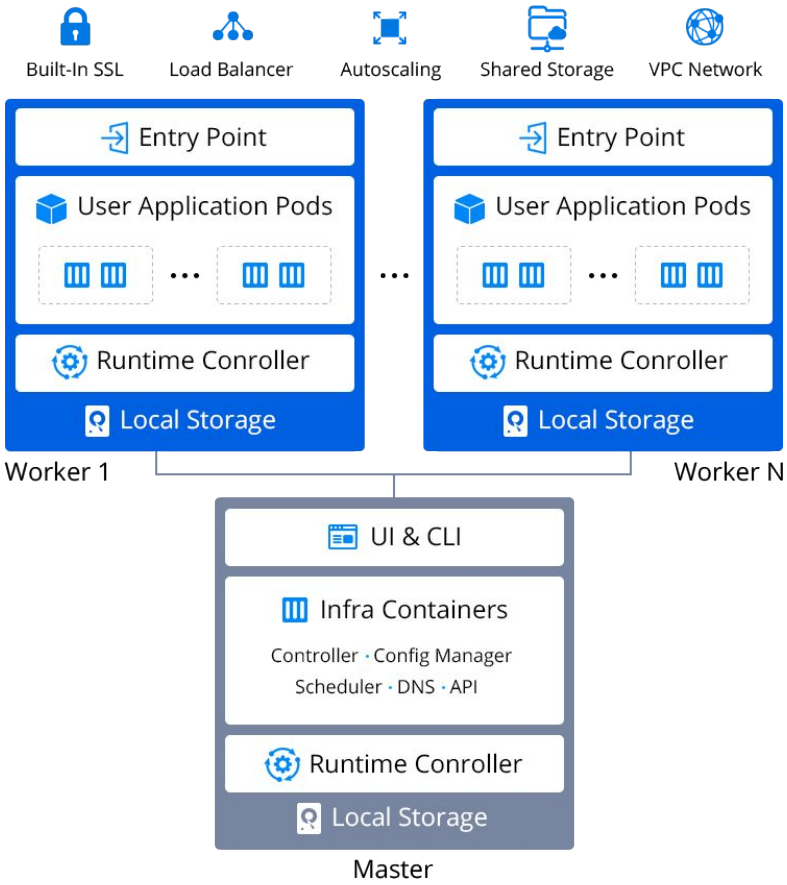
Configure remote API access and install complementary tools

Remote API Storage Jaeger

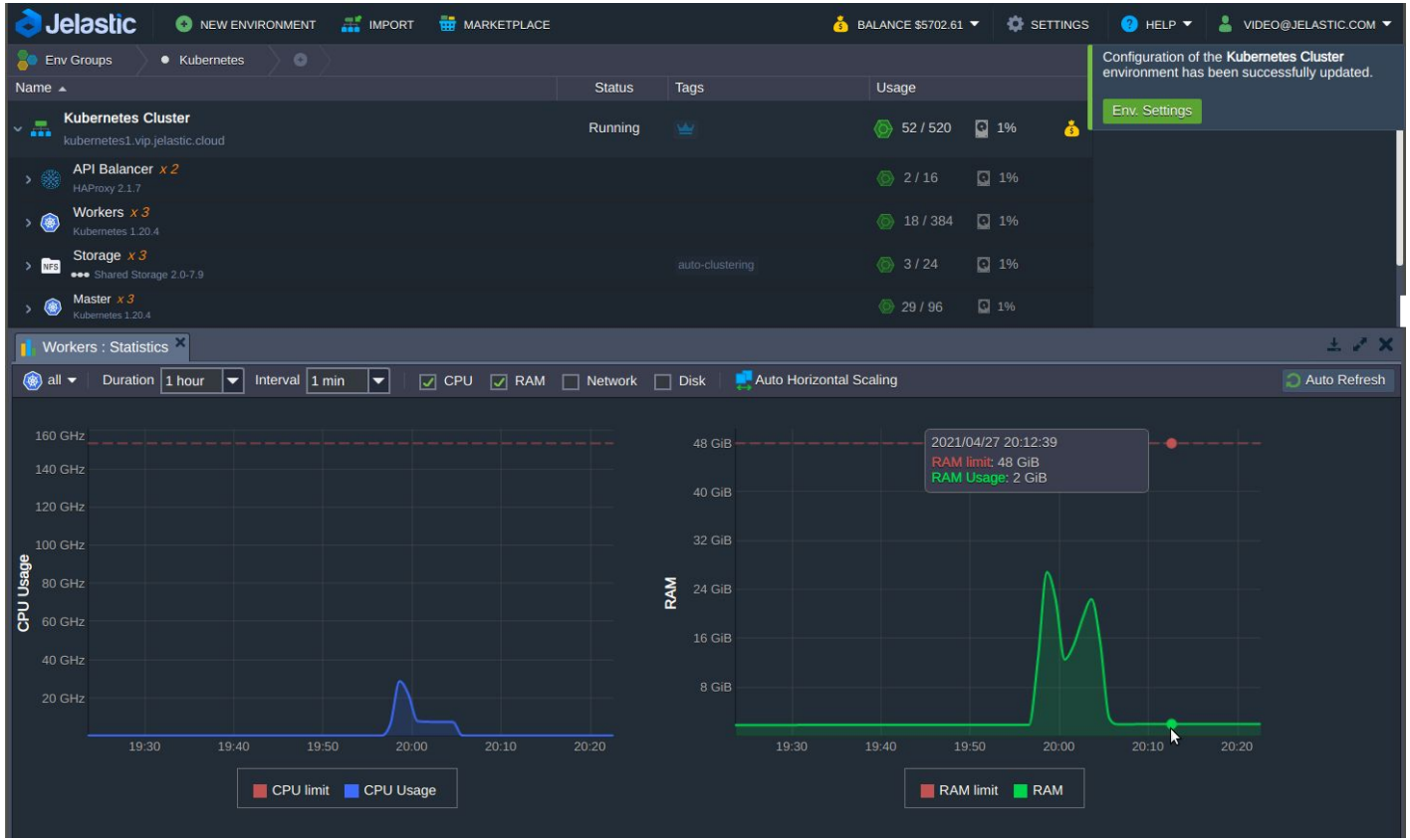
https://www.youtube.com/watch?v=l9H28icAlUg

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

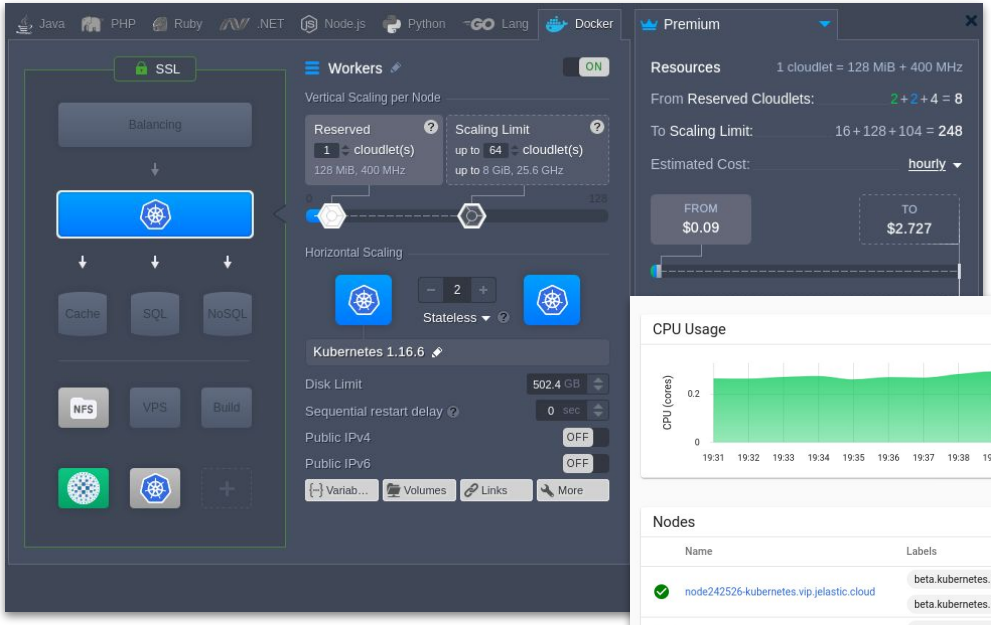


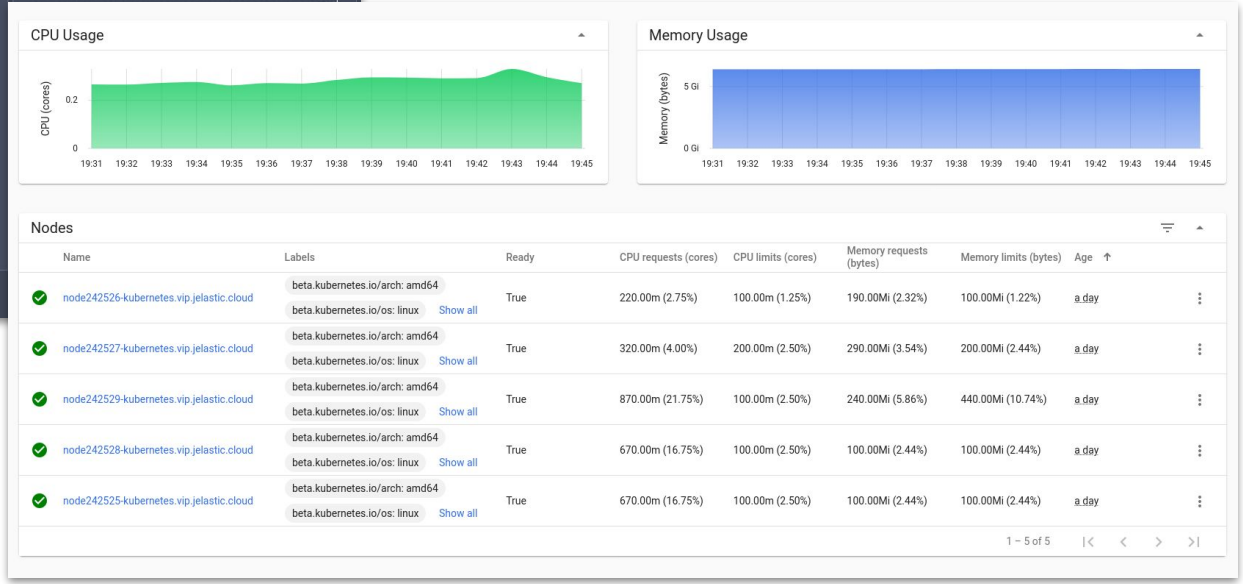
Automatic Vertical Scaling with Pay-Per-Actual-Use Pricing



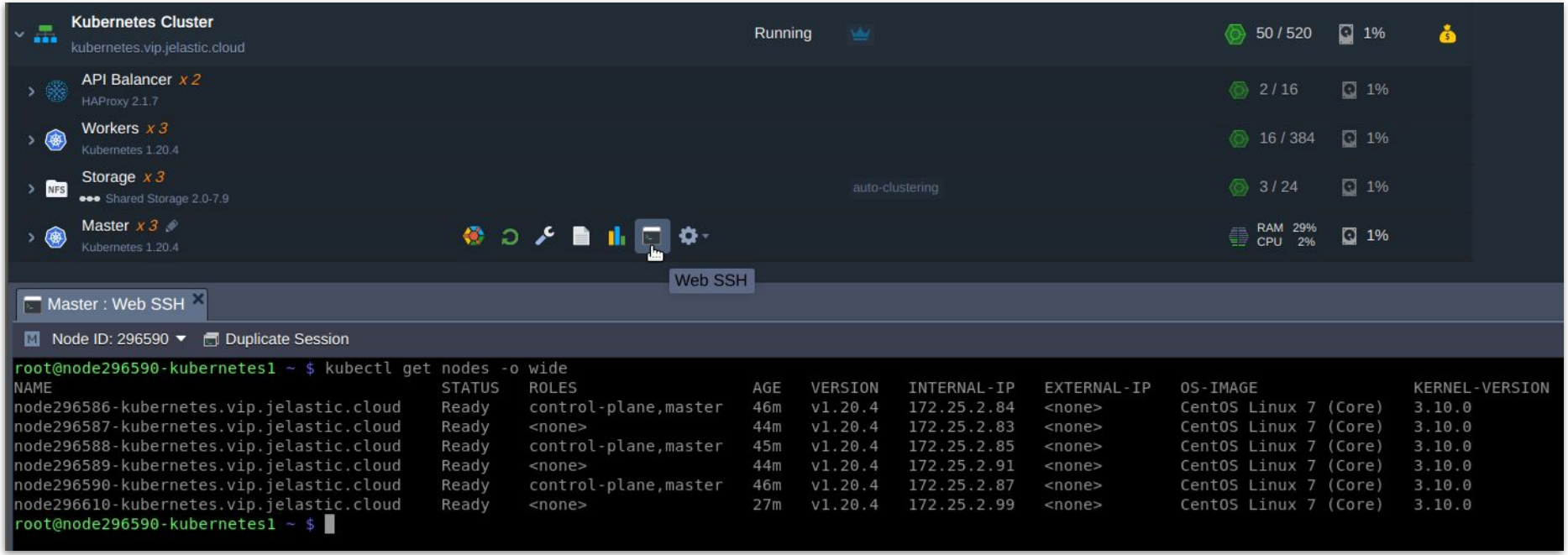
An example of Workers vertical scaling: available capacity 48 GiB vs actually used and billed 2 GiB

Changing Worker Node Resource Limits on the Fly

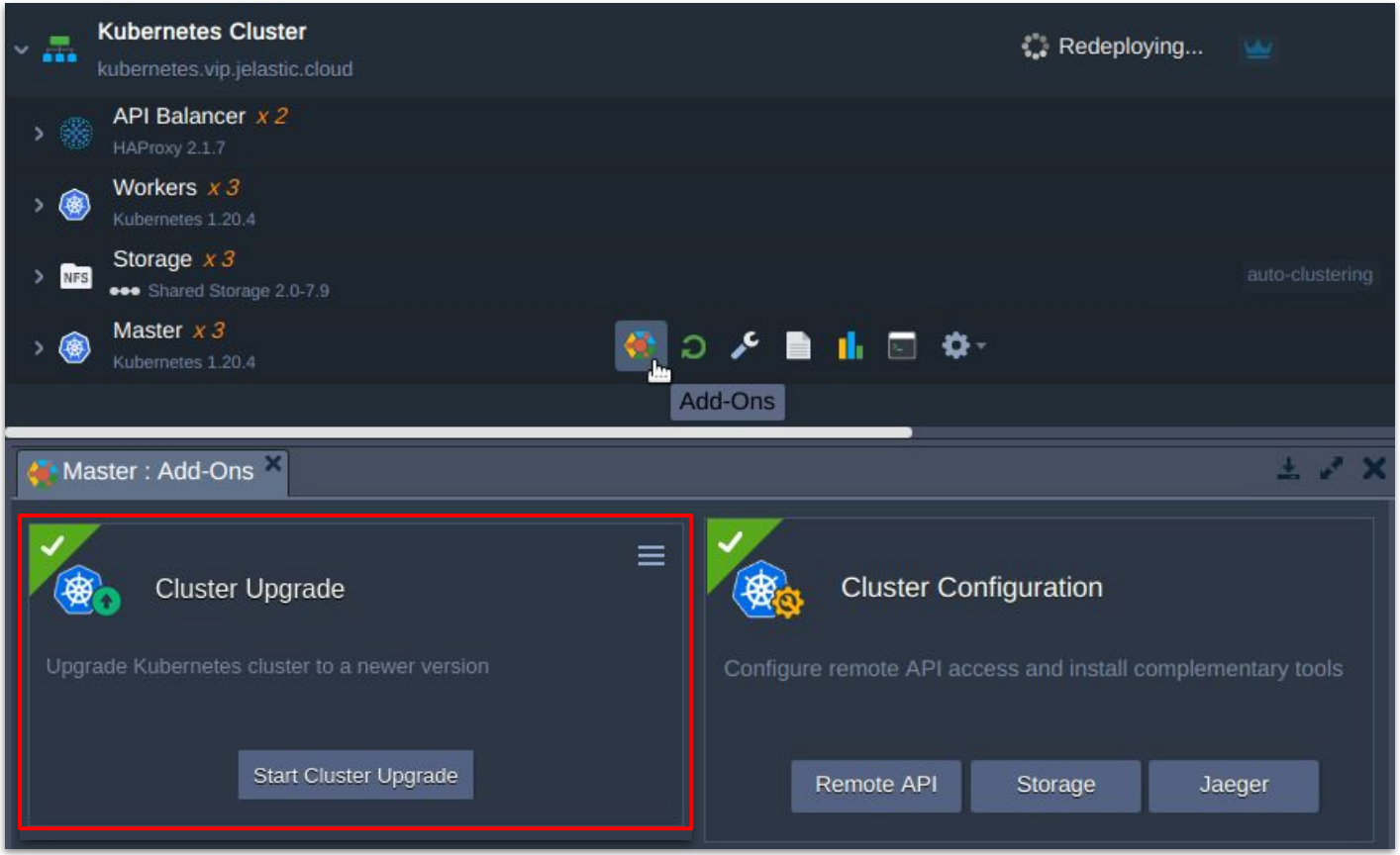




Access to Worker and Master Nodes Via Web SSH

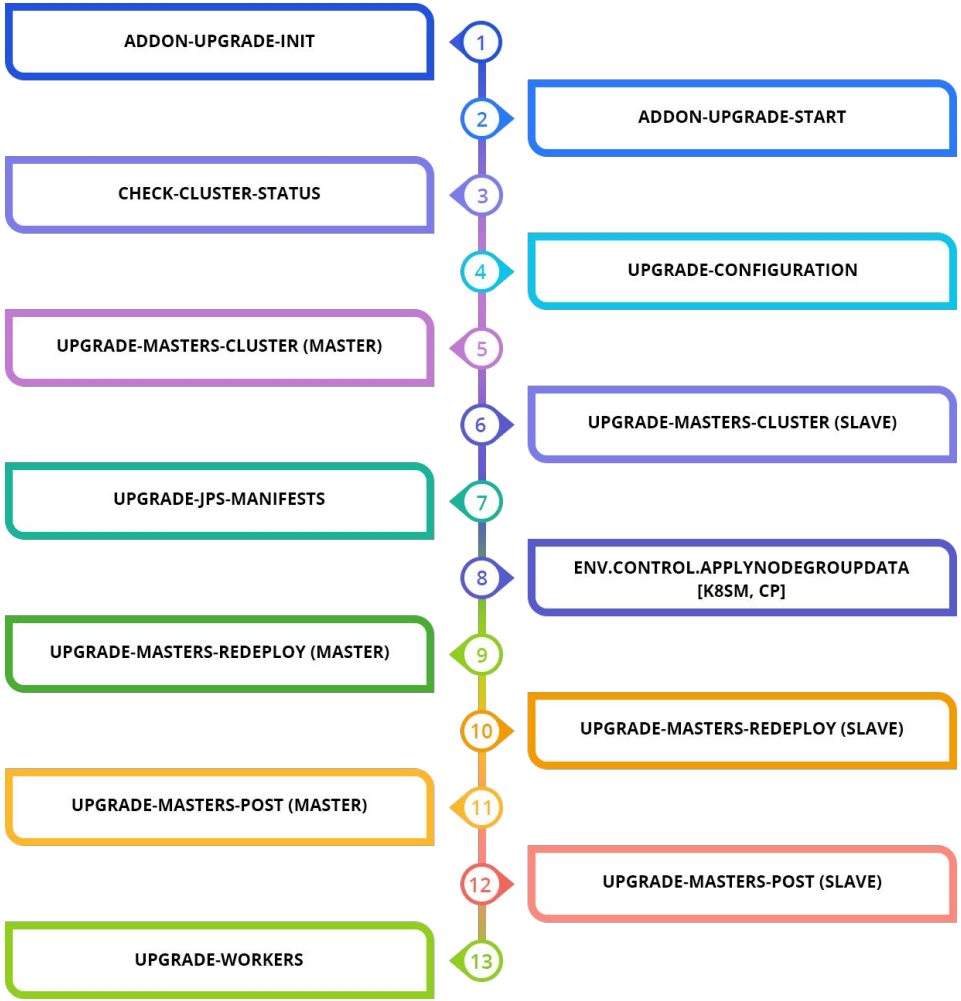


Upgrade Procedure

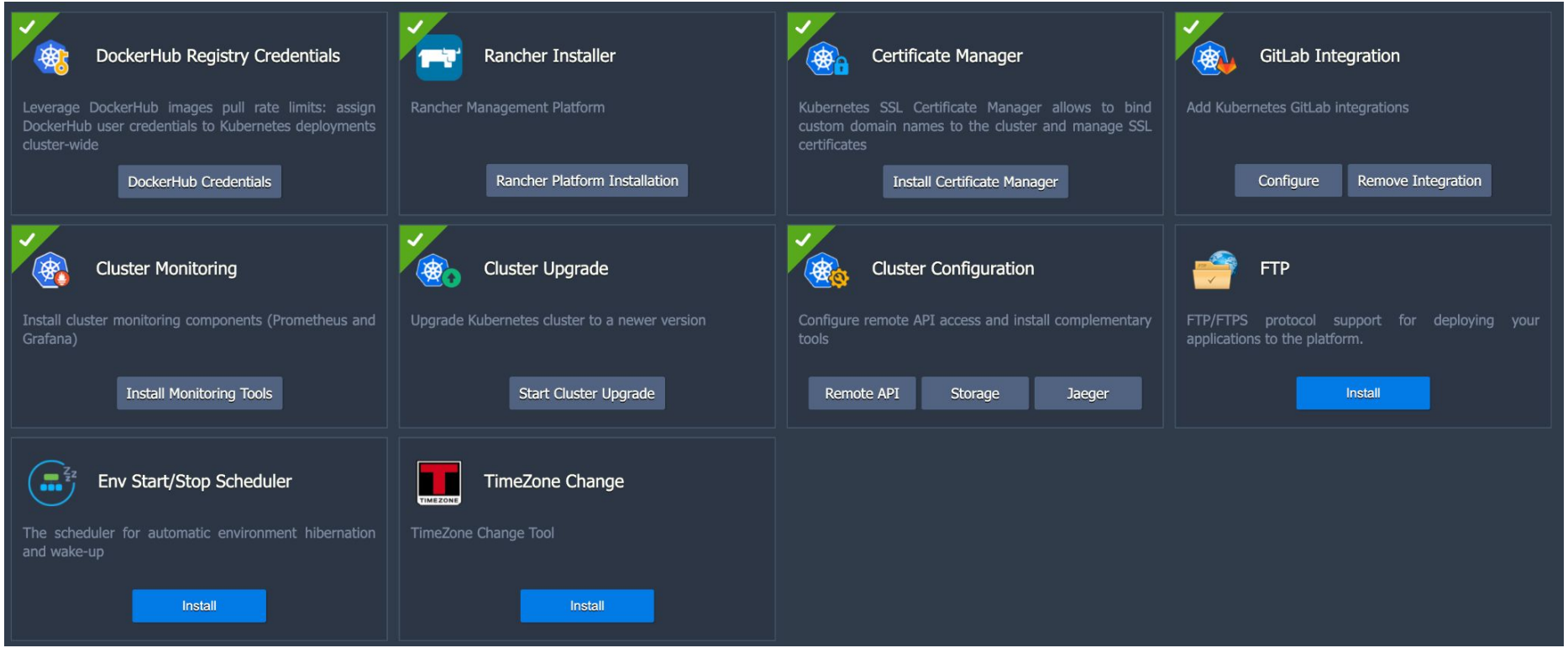


Upgrade Procedure

- 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

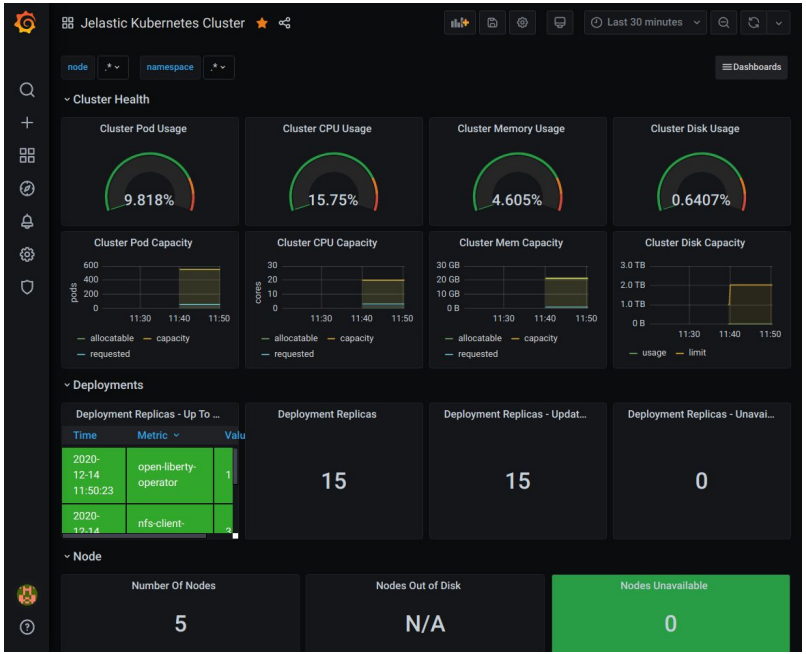


Built-In Add-Ons

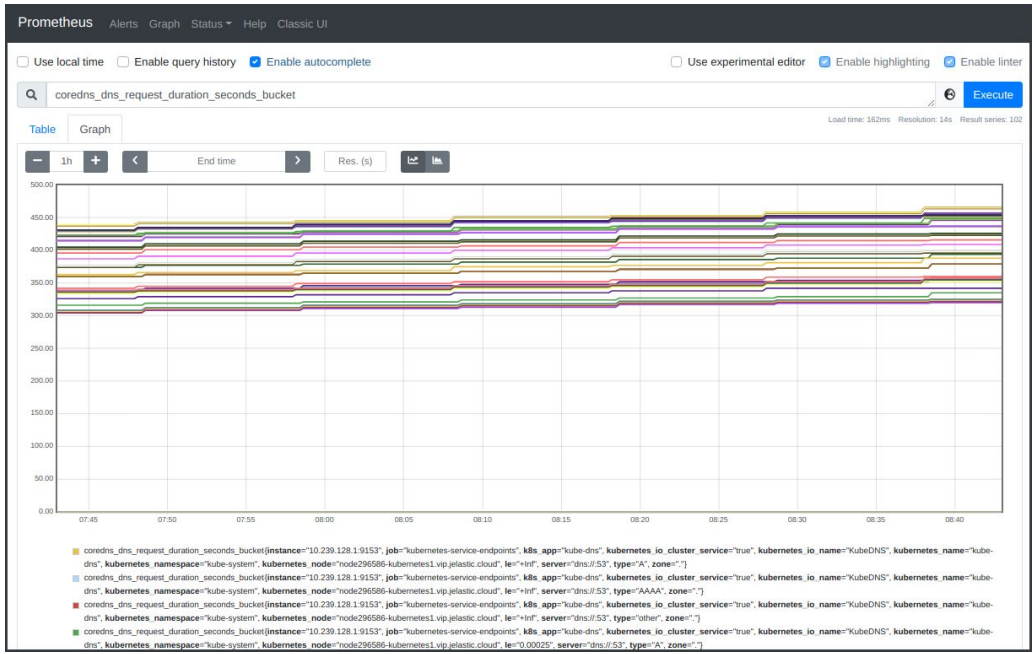


Embedded Cluster Monitoring

Grafana

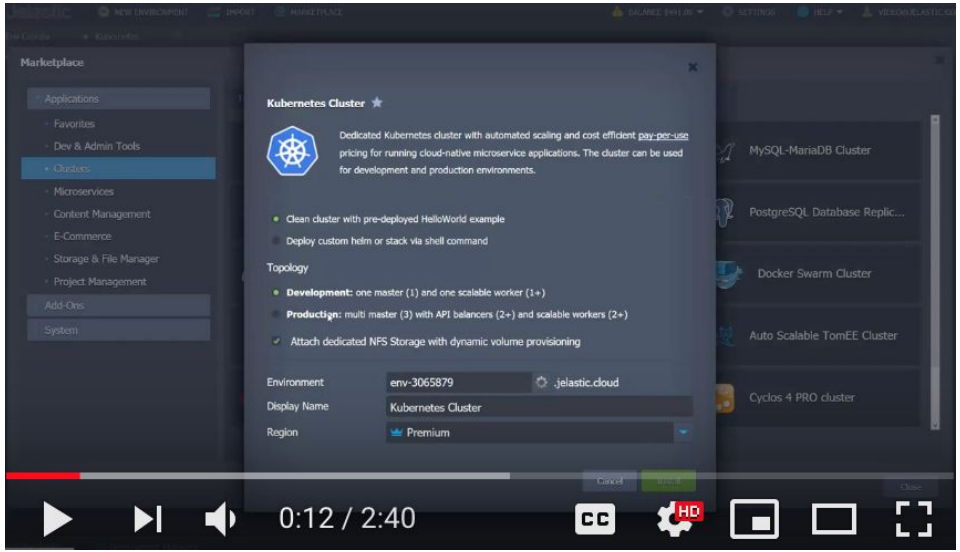


Prometheus



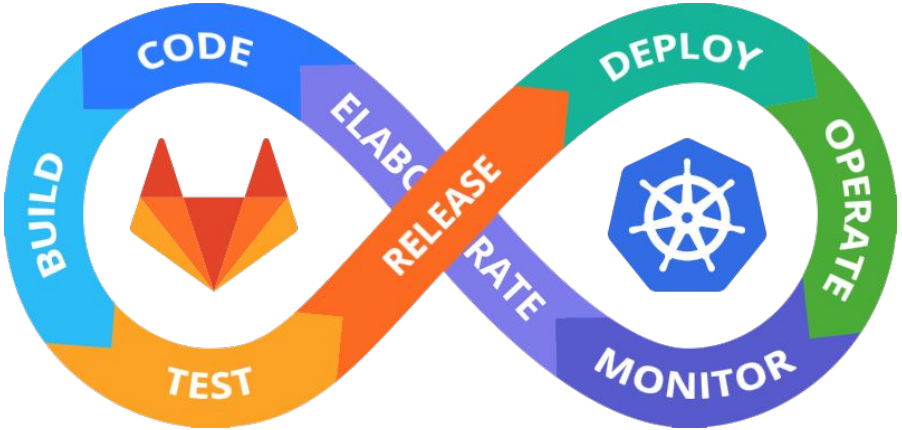
Kubernetes Issues Solved by Jelastic

- 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



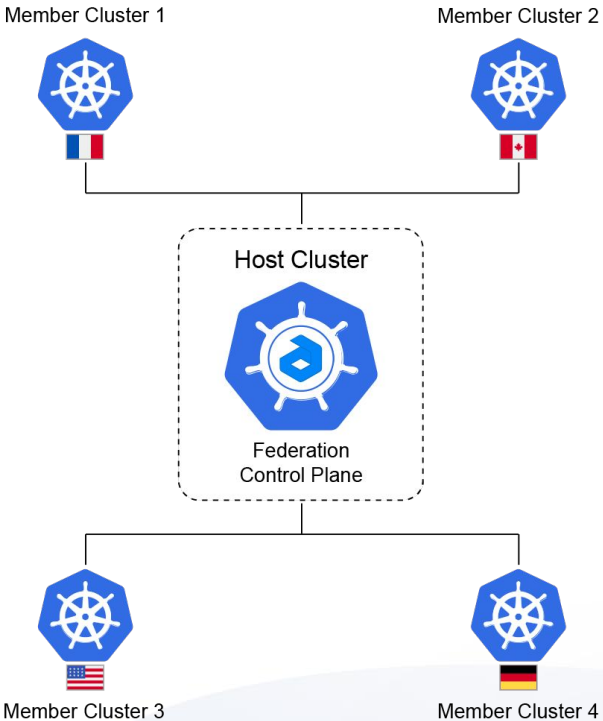
https://jelastic.com/blog/kubernetes-gitlab-ci-cd-integration/

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]



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

Jelastic
A DIVISION OF Virtuozzo

beebyte