# Introduction to Cloud COMPUTING

Giuliano Taffoni - I.N.A.F.

"Cloud Service Model"

**Lecture 4**

2023 @ Università di Trieste

# Outline

Intro to Cloud Service Models

IaaS & PaaS

# Cloud Definition: again

I. A model of computation and data storage based on “pay as you go”, access to “unlimited” remote data center capabilities

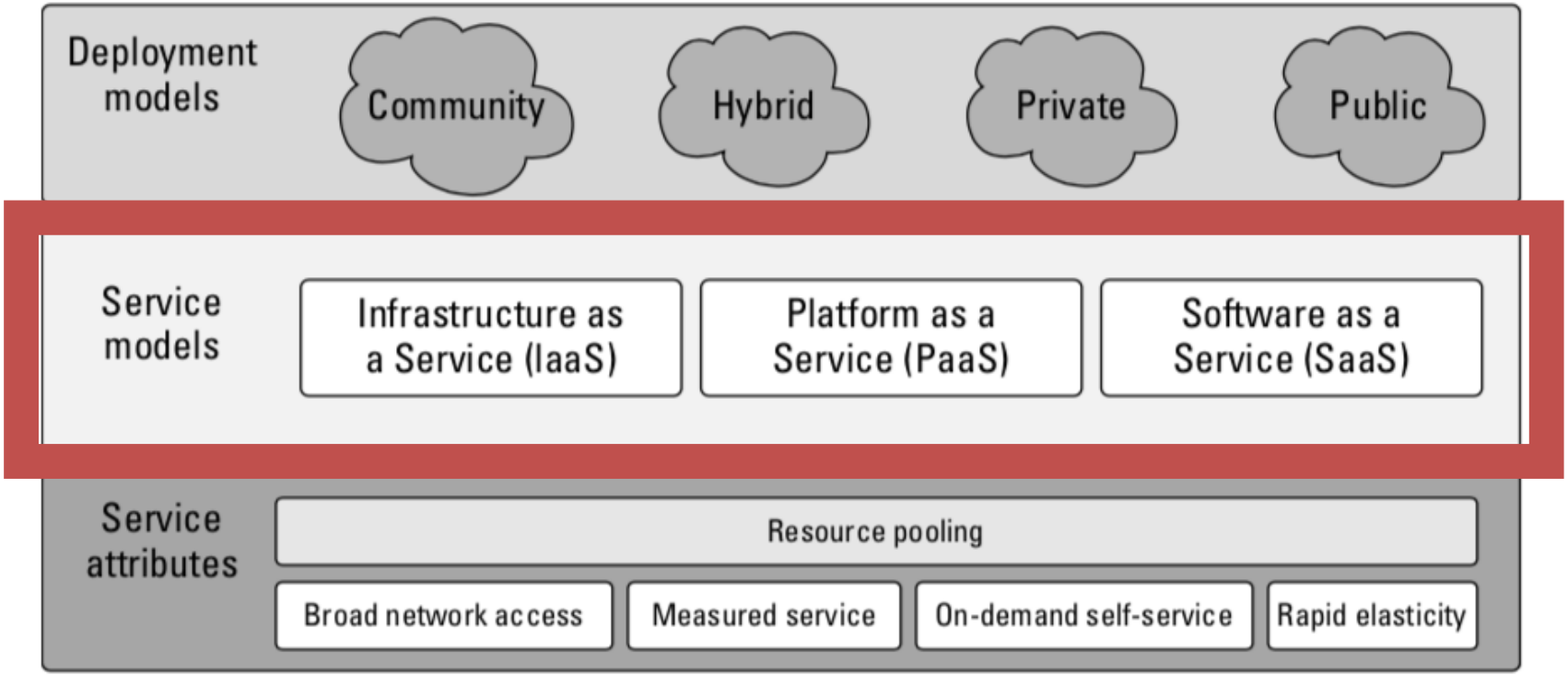
II. A cloud infrastructure provides a framework to manage scalable, reliable, on-demand access to applications

III. Cloud services provide the “invisible” backend to many of our mobile applications

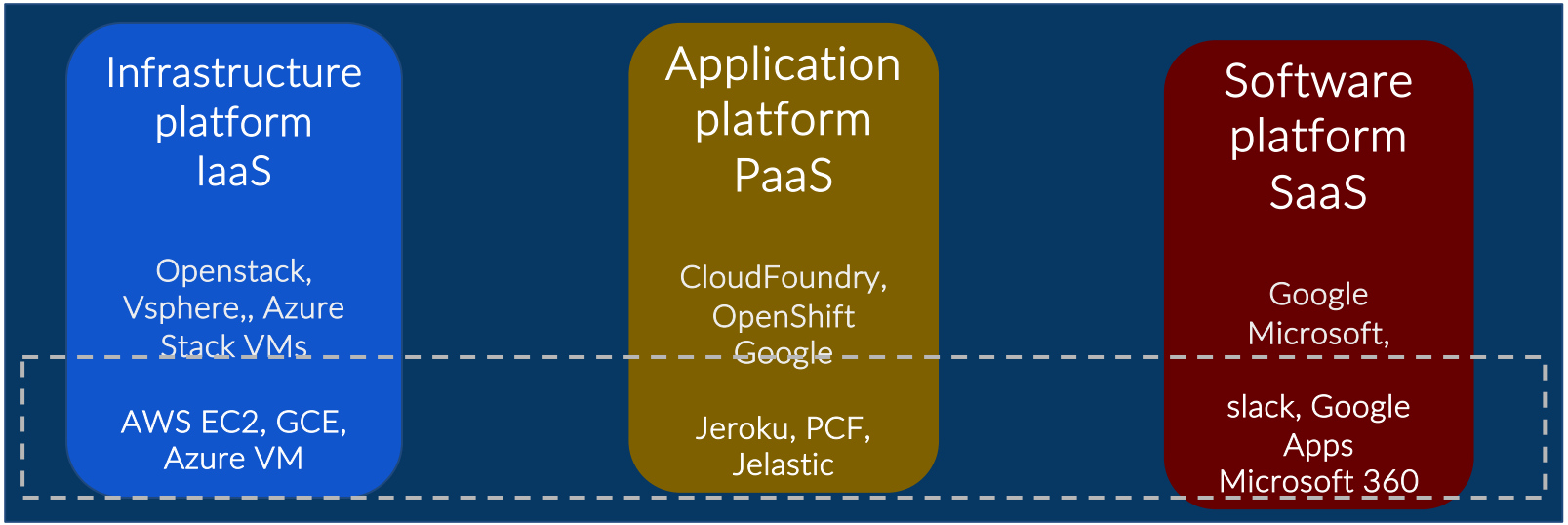
IV. High level of elasticity in consumption

V. Historical roots in today’s Internet apps
   - Search, email, social networks
   - File storage (Live Mesh, Mobile Me, Flicker, …)

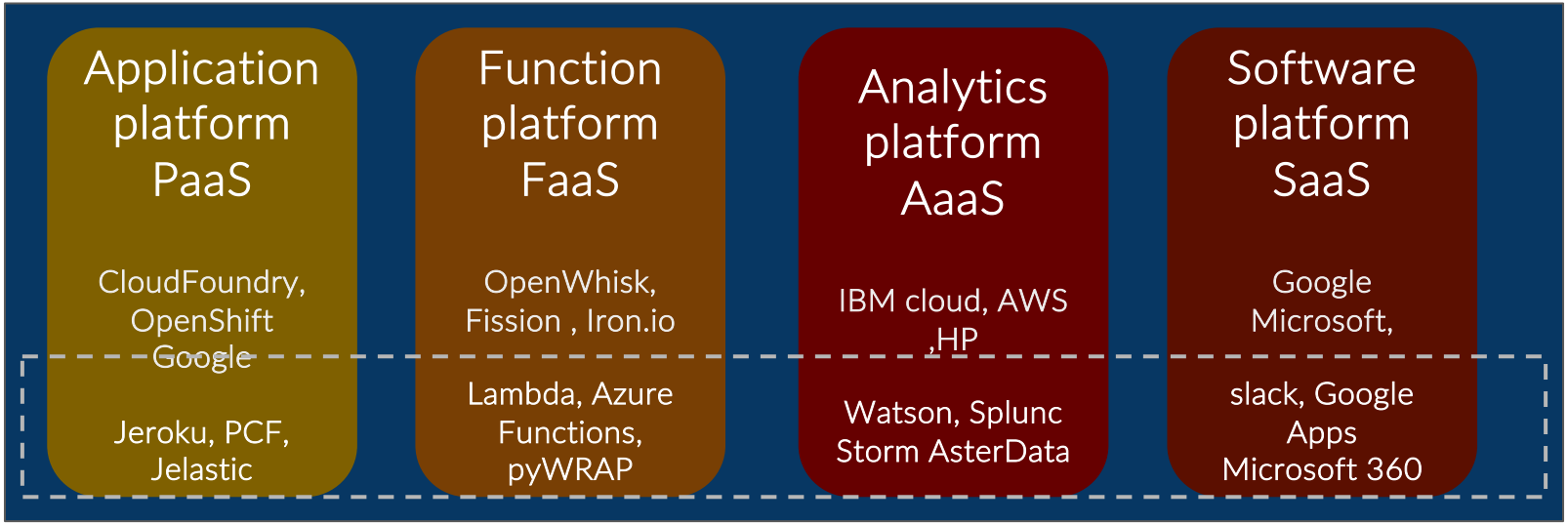
# Cloud Architecture



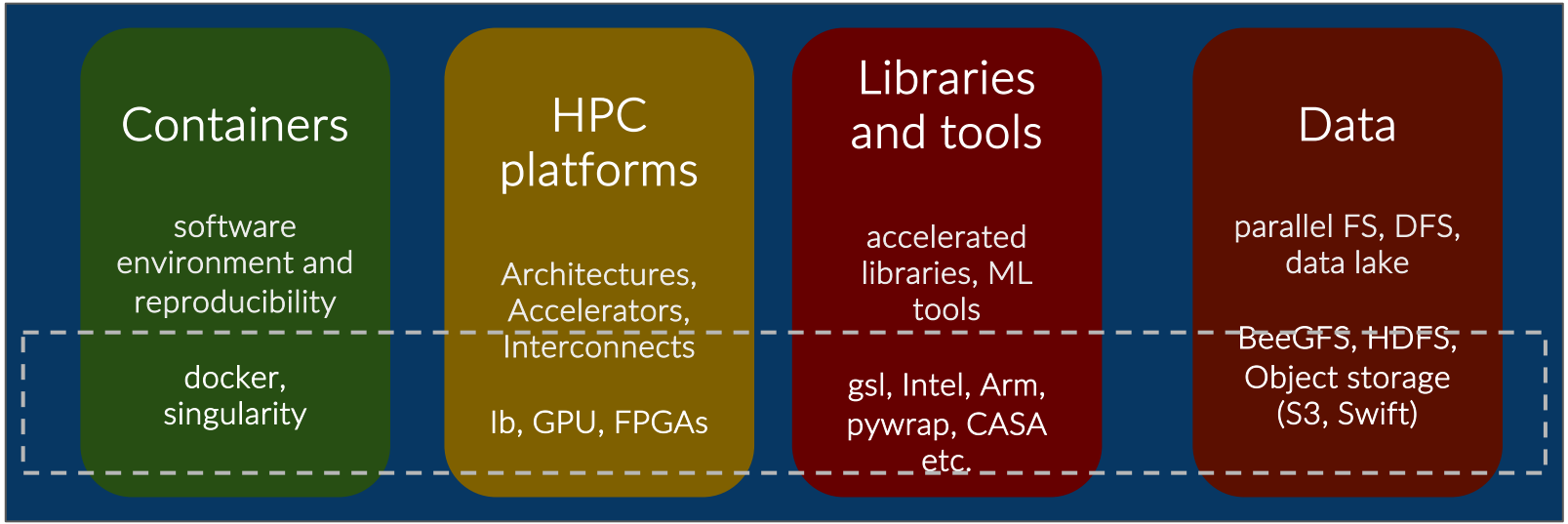
# Traditional service models



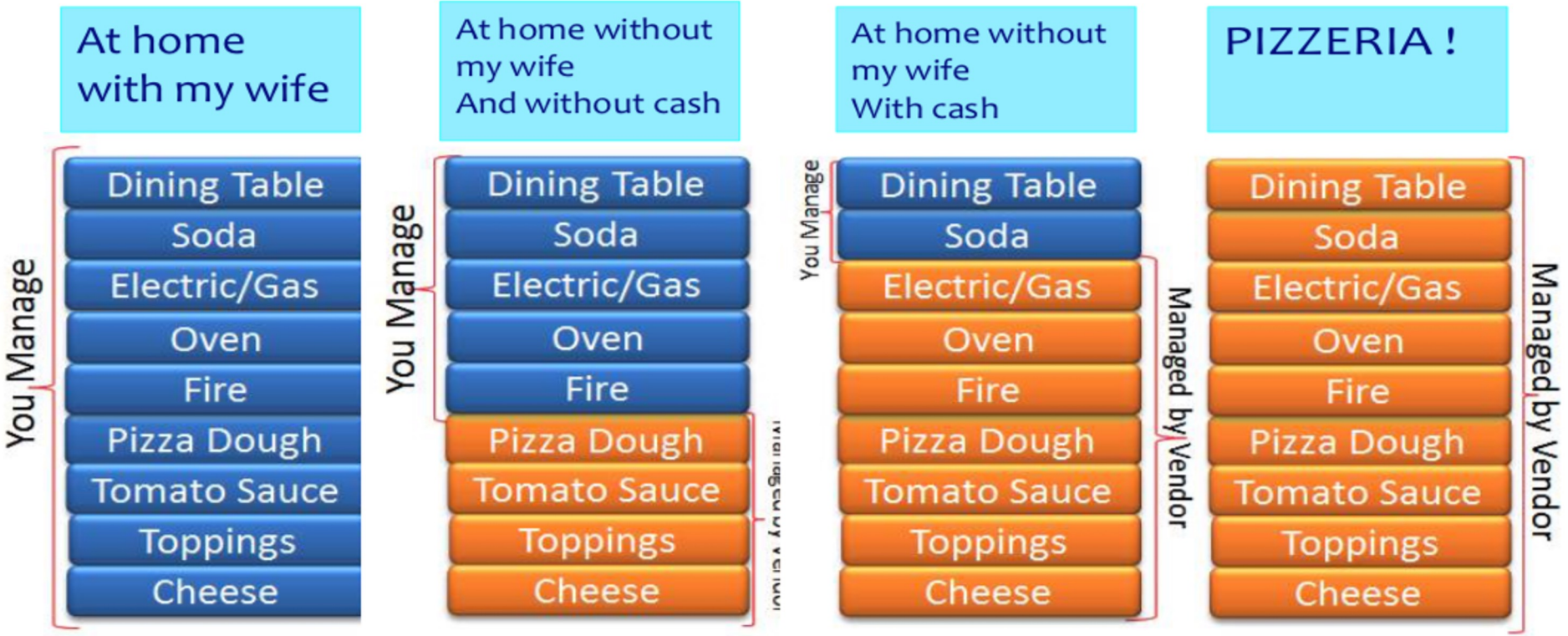
# Modern Models



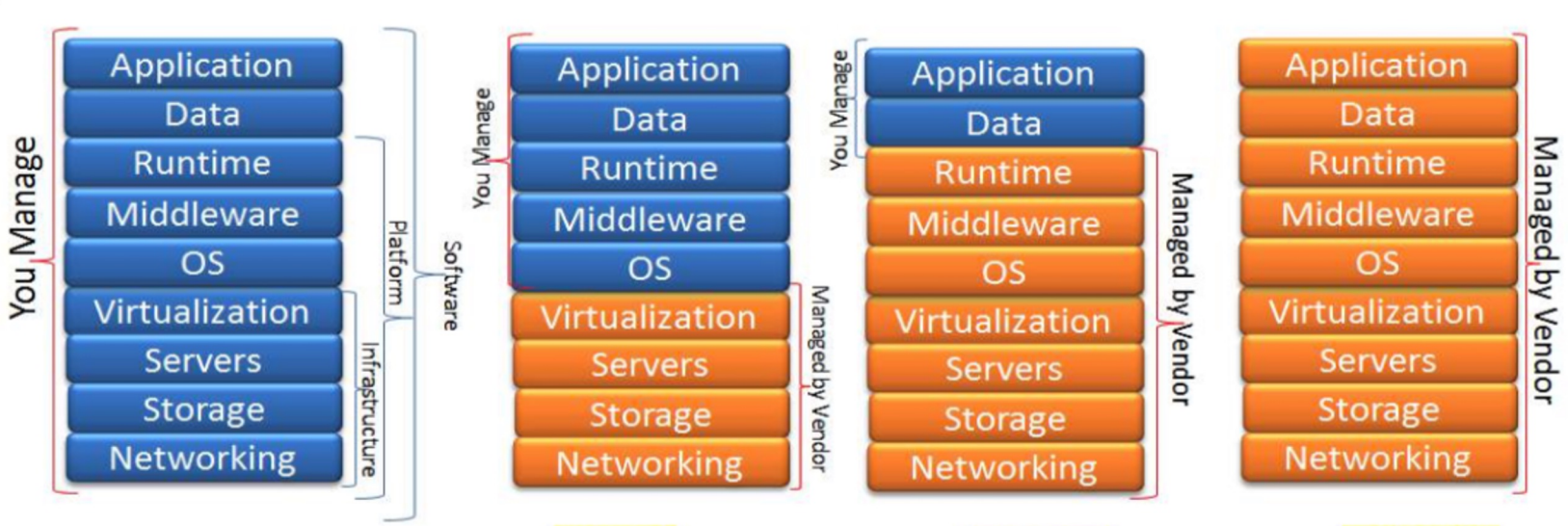
# Modern Models



# When I want a Pizza



# When I manage IT stuff...




Taken and Slightly changed from
https://cloudcelebrity.wordpress.com/2011/11/22/introduction-to-cloud-services-iaas-paas-saas/

# What is cloud?

A cloud infrastructure is:

*An infrastructure to provide users with the most **flexible** way to allocate computational power and storage space (and any other IT services)*

To quickly provide a customized IT infrastructure

…and you must integrate infrastructure provisioning in your data-analysis.

# For example in OpenStack….

```
>>> from keystoneauth1 import loading
>>> from keystoneauth1 import session
>>> from novaclient import client
>>> loader = loading.get_plugin_loader('password')
>>> auth =
loader.load_from_options(auth_url=AUTH_URL,
...
username=USERNAME,
...
password=PASSWORD,
...
project_id=PROJECT_ID)
>>> sess = session.Session(auth=auth)
>>> nova = client.Client(VERSION, session=sess)
```

# Infrastructure as a Service

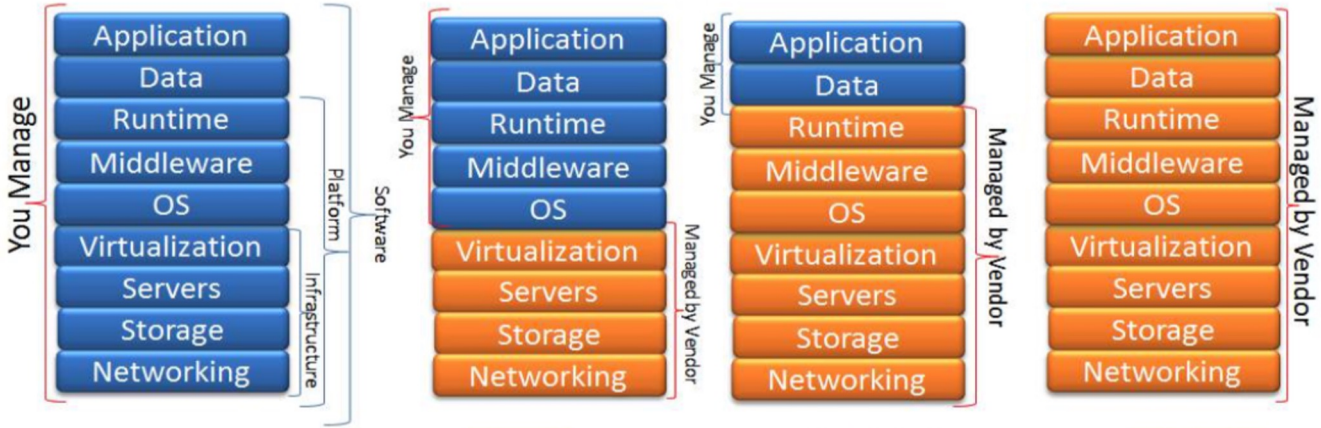
“Infrastructure as a service (IaaS) refers to online services that provide high-level APIs used to dereference various low-level details of underlying network infrastructure like physical computing resources, location, data partitioning, scaling, security, backup etc. A **hypervisor**, such as Xen, Oracle VirtualBox, Oracle VM, KVM, VMware ESX/ESXi, or Hyper-V, LXD, runs the virtual machines as guests. Pools of hypervisors within the **cloud operational system** can support large numbers of virtual machines and the ability to **scale services up and down** according to customers' varying requirements.”

Wikipedia

# IaaS

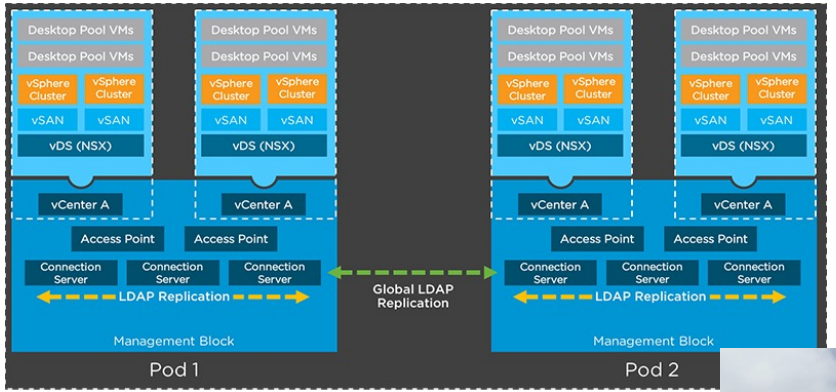
Infrastructure as a Service is a cloud computing service model in which hardware is virtualized in the cloud.

Separation of responsabilities



# IaaS architectural elements

- Workloads
- Point of Delivery
- Aggregation
- Silos

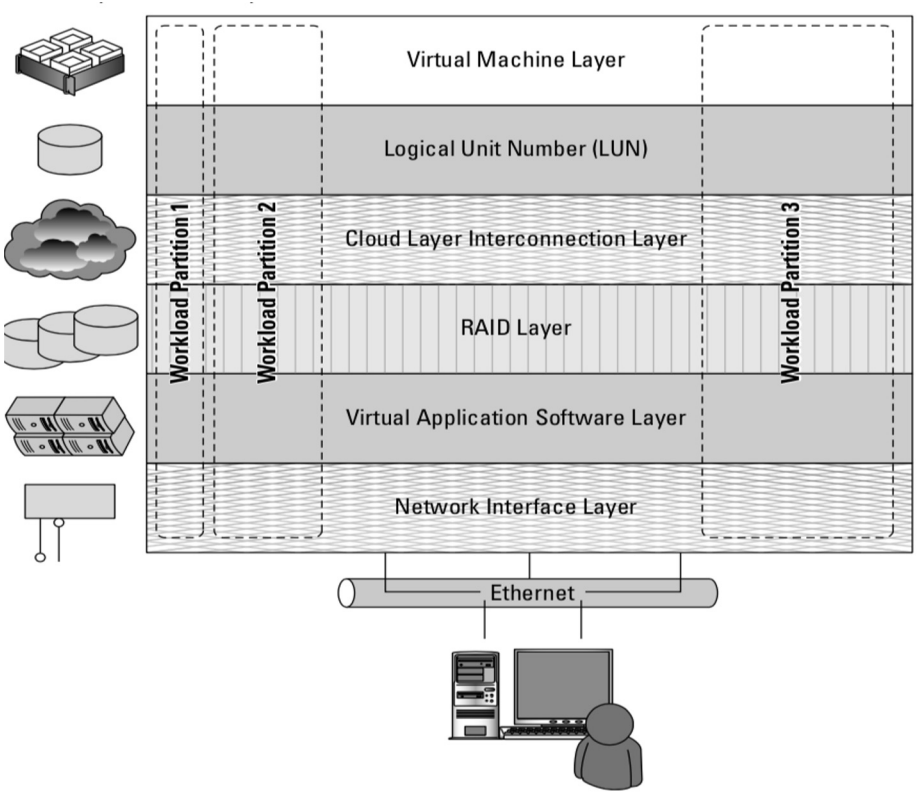




# IaaS Worloads

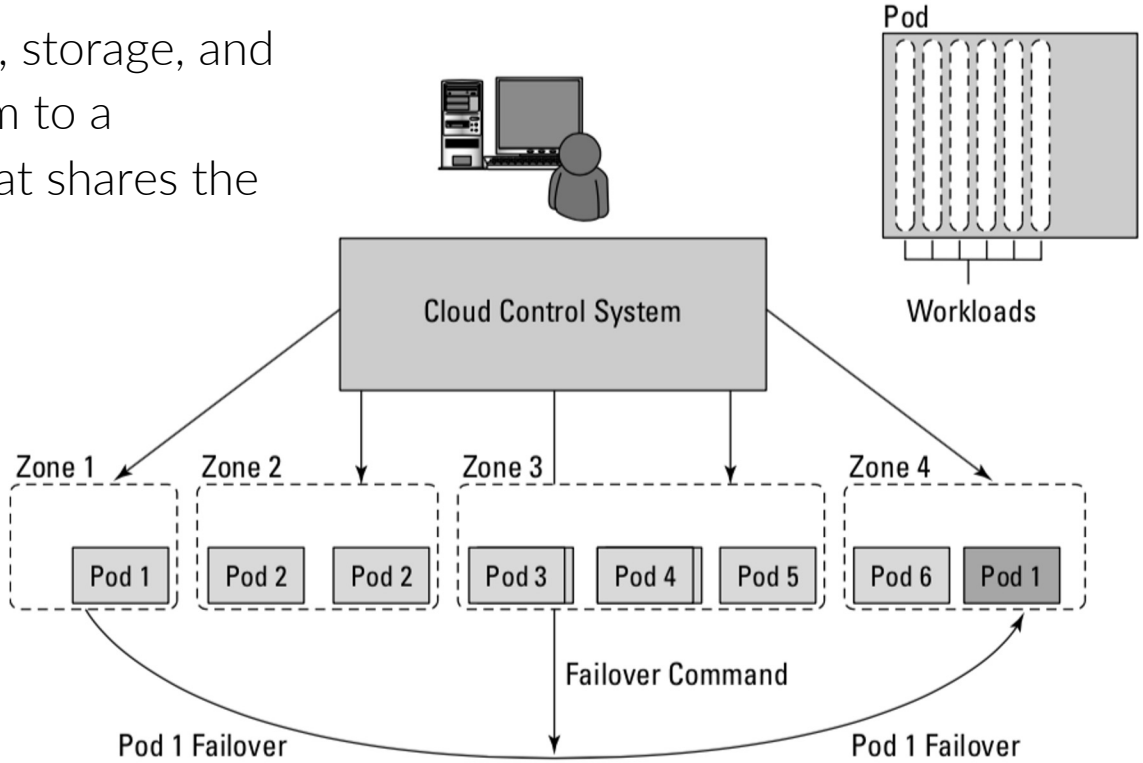
A cloud workload is a specific application, service, capability or a specific amount of work that can be run on a cloud resource. Virtual machines, databases, containers,

Hadoop nodes and applications are all considered cloud workloads.



# Point of Delivery

POD is a collection of compute, storage, and network resources that conform to a standard operating footprint that shares the same failure domain.



# POD vs SILO

POD are aggregated into pools within the same IaaS zone or area: the *availability zone*.

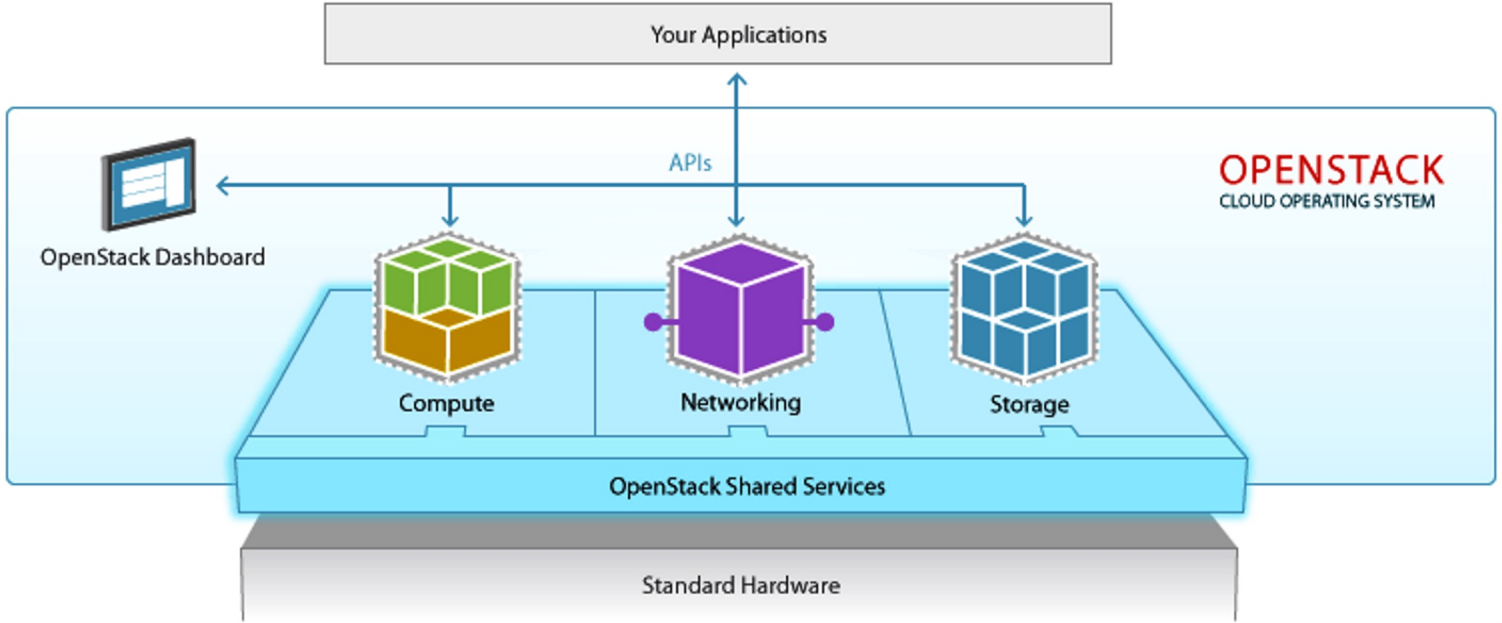
When cloud computing infrastructure isolates user clouds from each other so the management system is incapable of interoperating with other private clouds, it creates an information silo.

# IaaS examples

Amazon Web Services (AWS)
Google Compute Engine (GCE)
IBM Cloud
Microsoft Azure
Rackspace
Linode
Cisco Metacloud
Digital Ocean
Vultr
Oracle Cloud

European Open Science Cloud (OpenStack) (https://eosc-portal.eu/)

# OpenStack




OpenStack is a **cloud operating system** that controls large **pools** of compute, storage, and networking resources throughout a datacenter, all managed through a **dashboard** that gives administrators control while **empowering their users** to provision resources through a web Interface.

# OpenStack main concepts

- The OpenStack project is an open source cloud computing platform that supports all types of cloud environments.
- The project aims for simple implementation, massive scalability, and a rich set of features.
- Cloud computing experts from around the world contribute to the project.
- Based on services (composability), each service offers an application programming interface (API) that facilitates this integration

# OpenStack principles

A. **Open development model:** Code reviews and roadmaps are public.The code for OpenStack is freely available under the Apache 2.0 license.
B. **Open design process**: Every six months the development community holds a design summit to gather requirements and write specifications for the upcoming release. The summit is open to the public and attendees include users, operators, developers, and upstream project personnel.
C. **Open community:** OpenStack is dedicated to producing a healthy,vibrant, and active developer and user community. Most decisions are made by consensus. All processes are documented, open and transparent.

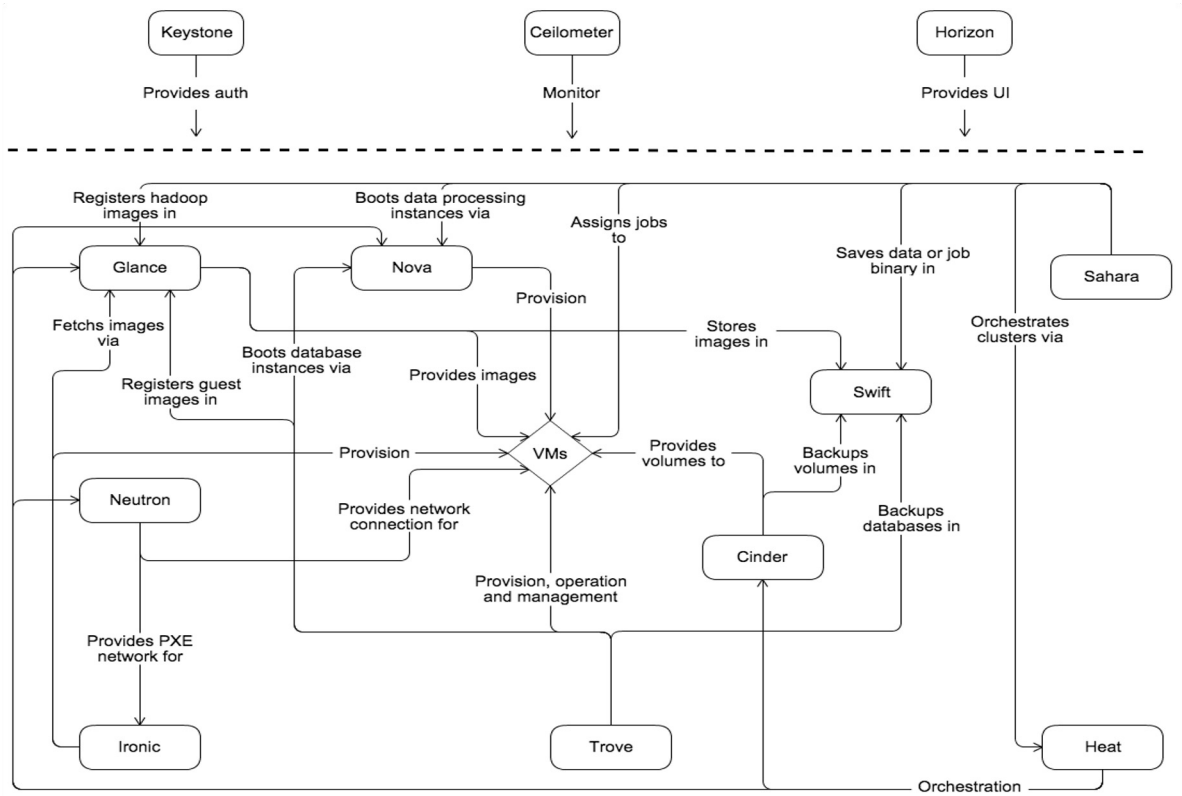
# OpenStack Capabilities

- Virtual machines (VMs) and Containers on demand
  - provisioning
  - snapshotting
- Networks
- Volumes
- Storage for VMs and arbitrary files
- Multi-tenancy
  - quotas for different projects, users
  - user can be associated with multiple projects

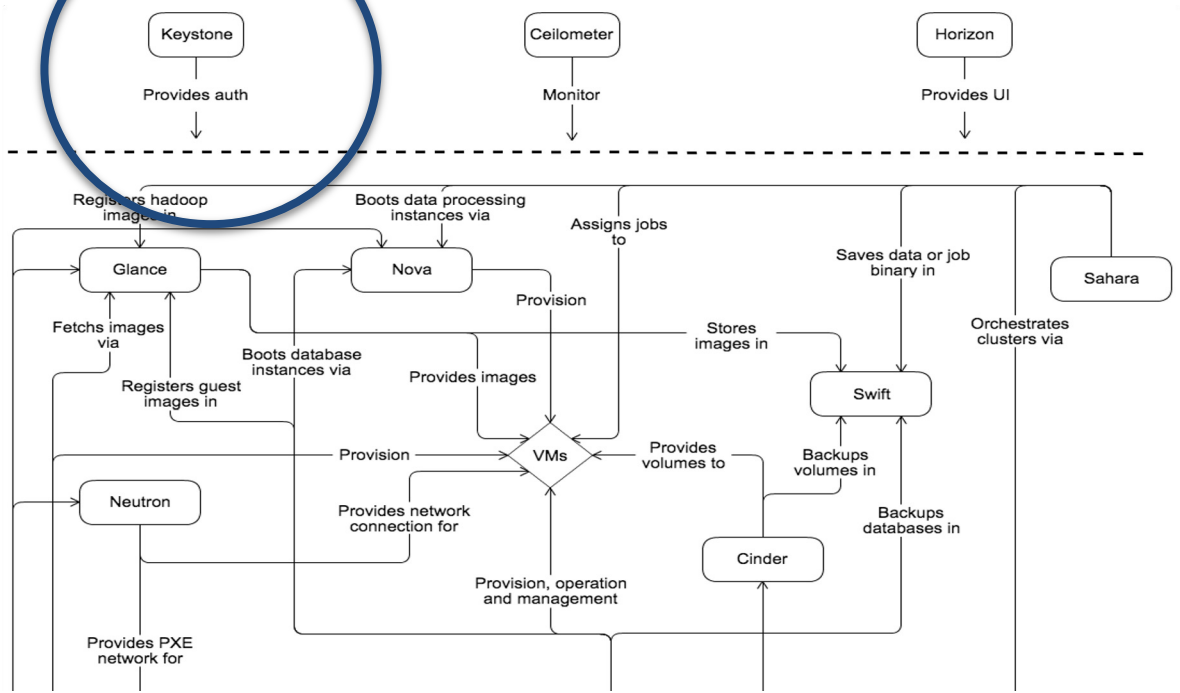
# OpenStack tech details

- Written in Python (plus auxiliary shell scripts)
- Built around independent components
- Highly distributed architecture
  - designed for very big installations
- Intrinsic HA of most OpenStack services
  - MySQL and RabbitMQ have to be properly configured
- *SQL database used to store persistent data
- RabbitMQ used for RPC and notification
- RESTful APIs for all the services

# OpenStack IaaS services view



# OpenStack IaaS services view

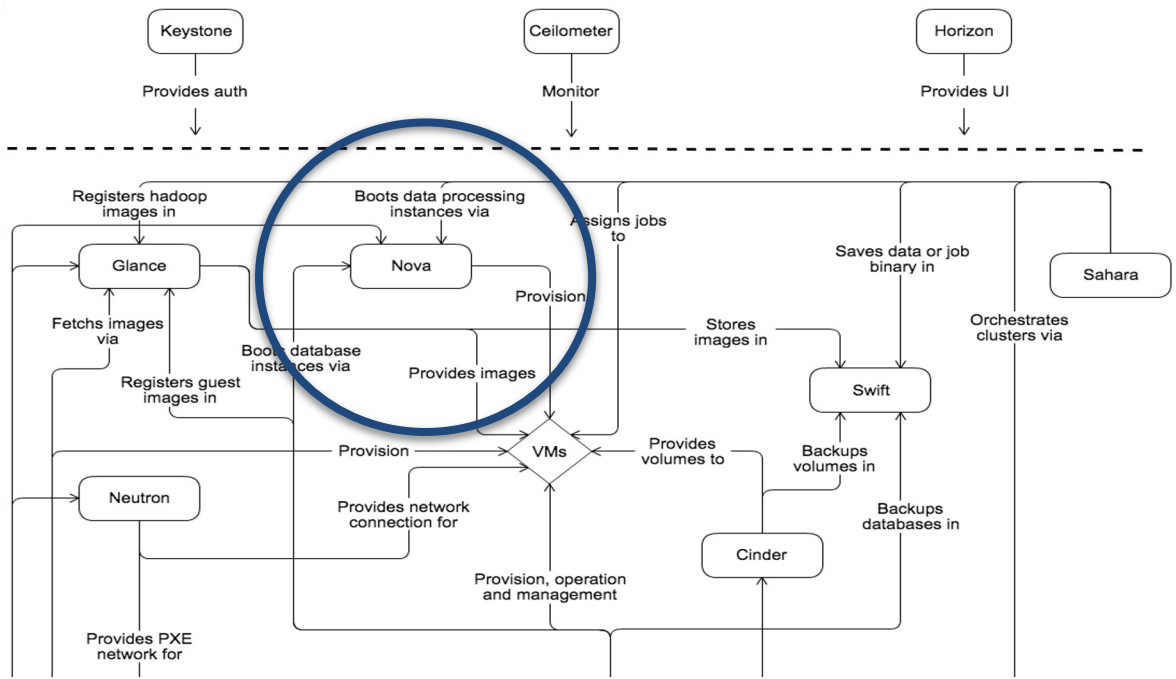



## Keystone:authentication service

# Keystone

```
>>> from keystoneauth1.identity import v3
>>> from keystoneauth1 import session
>>> from keystoneclient.v3 import client
>>> auth=v3.Password(auth_url='https://my.keystone.com:5000/v3',
...                  user_id='myuserid',
...                  password='mypassword',
...                  project_id='myprojectid')
>>> sess=session.Session(auth=auth)
>>> keystone=client.Client(session=sess)
```

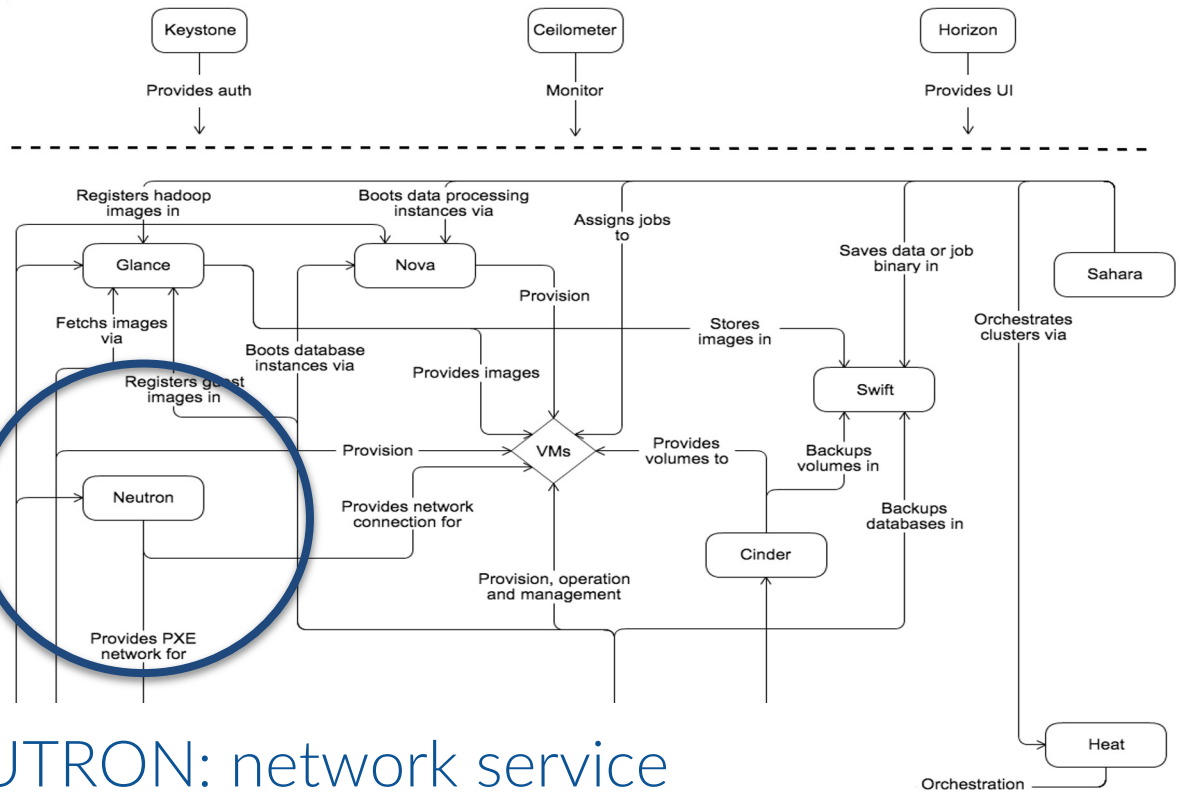
# OpenStack IaaS services view




## Nova: computational service

```
>>> nova = client.Client(VERSION, session=sess)

…

>>> nova.servers.list()

[<Server: buildslave-ubuntu-9.10>]


>>> nova.flavors.list()

[<Flavor: 256 server>,

<Flavor: 512 server>,

<Flavor: 1GiB server>,

<Flavor: 2GiB server>,

<Flavor: 4GiB server>,

<Flavor: 8GiB server>,

<Flavor: 15.5GiB server>]


>>> fl = nova.flavors.find(ram=512)

>>> nova.servers.create("my-server", flavor=fl)

<Server: my-server>
```

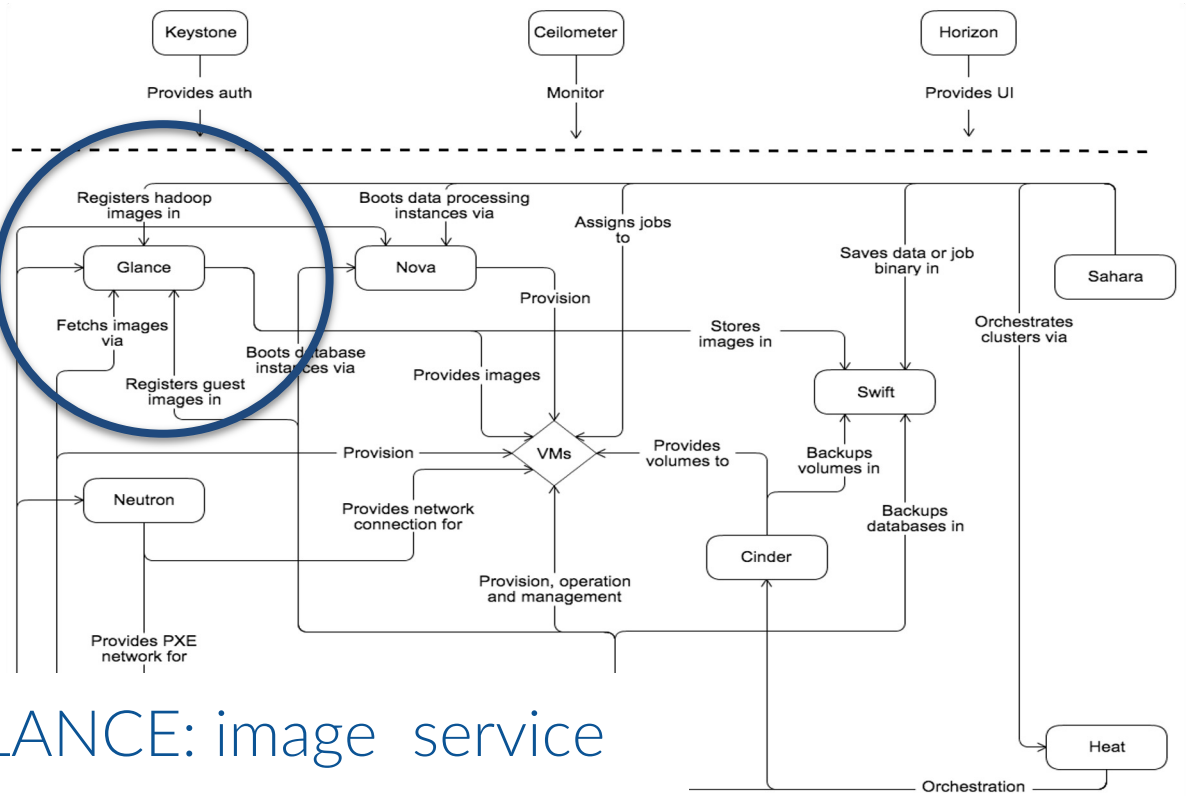
# OpenStack IaaS services view




NEUTRON: network service

```
>>> neutron = client.Client(session=sess)

...

...

>>> network = {'name': 'mynetwork', 'admin_state_up': True}

>>> neutron.create_network({'network':network})

>>> networks = neutron.list_networks(name='mynetwork')

>>> print networks

>>> network_id = networks['networks'][0]['id']

>>> neutron.delete_network(network_id)
```

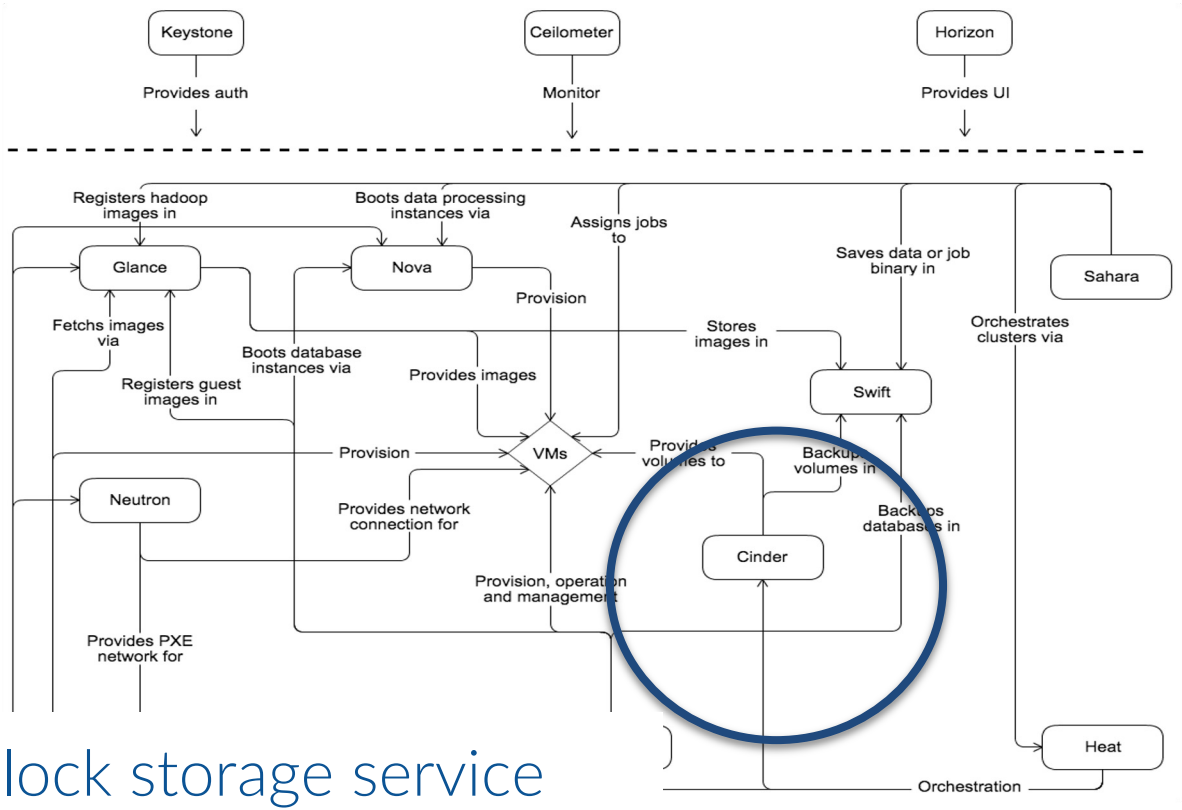
# OpenStack IaaS services view




GLANCE: image service

```
>>> from glanceclient import Client
>>> glance = Client('1', endpoint=OS_IMAGE_ENDPOINT, token=OS_AUTH_TOKEN)
>>> image = glance.images.create(name="My Test Image")
>>> print image.status
'queued'
>>> image.update(data=open('/tmp/myimage.iso', 'rb'))
>>> print image.status
'active'
>>> image.update(properties=dict(my_custom_property='value'))
>>> with open('/tmp/copyimage.iso', 'wb') as f:
        for chunk in image.data():
        f.write(chunk)
>>> image.delete()
```

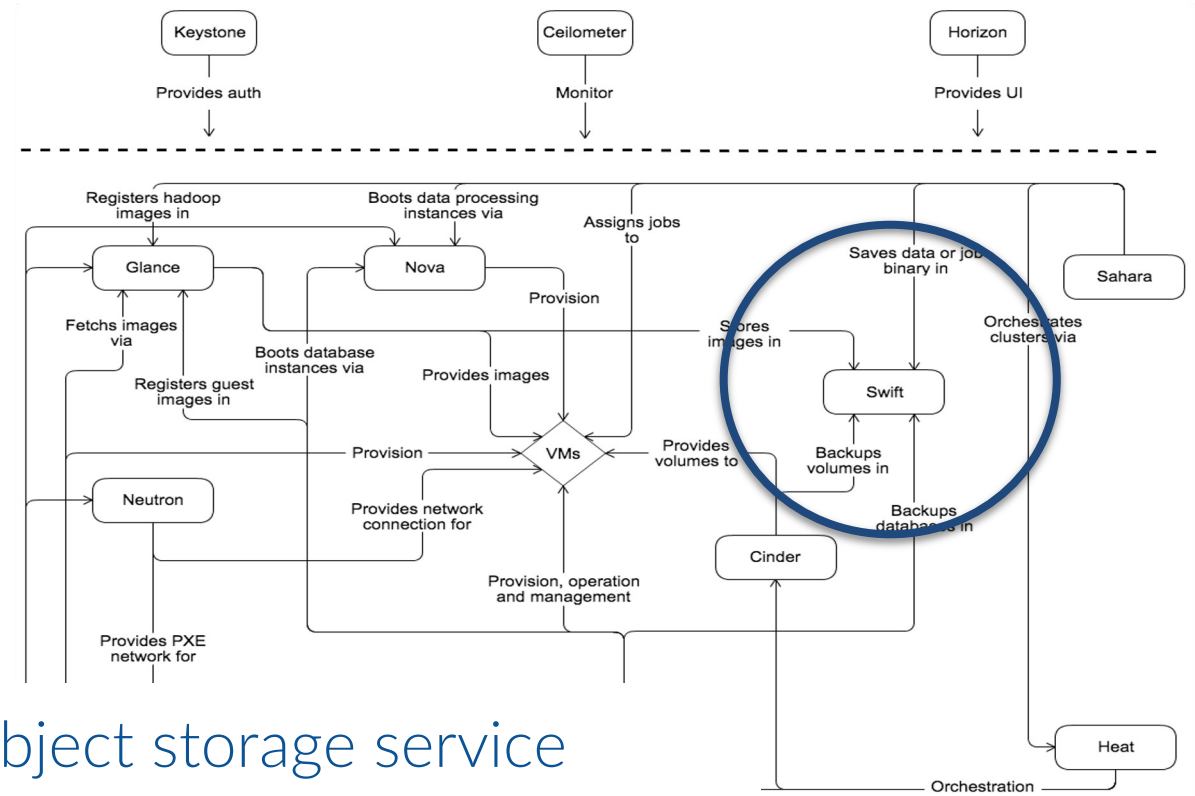
# OpenStack IaaS services view




CINDER: block storage service

```
>>> sess = session.Session(auth=auth)
>>> cinder = client.Client(VERSION, session=sess)
>>> cinder.volumes.list()
>>> myvol = cinder.volumes.create(display_name="test-vol", size=1)
>>> myvol.id
ce06d0a8-5c1b-4e2c-81d2-39eca6bbfb70
>>> cinder.volumes.list()
[<Volume: ce06d0a8-5c1b-4e2c-81d2-39eca6bbfb70>]
>>>myvol.delete
[]
```

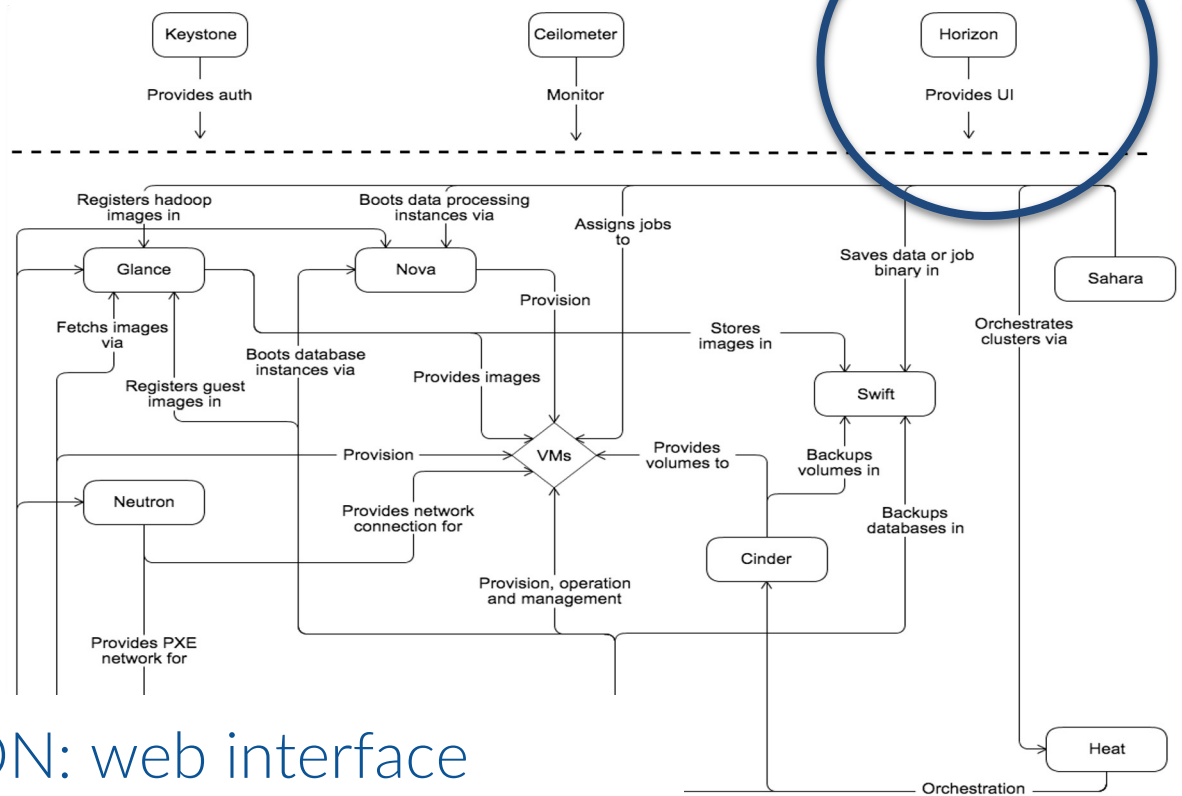
# OpenStack IaaS services view



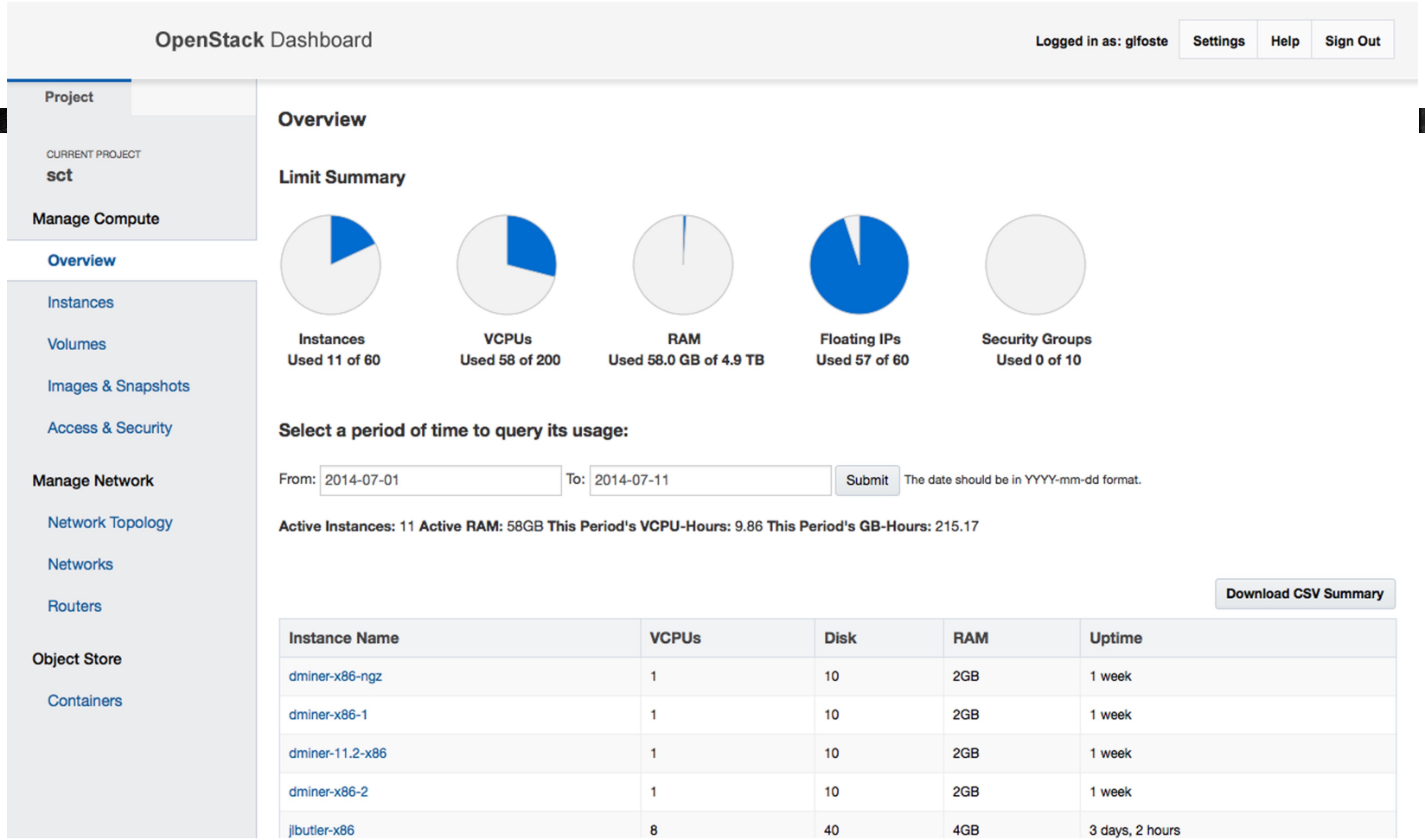

SWIFT: object storage service

```
container = 'new-container'
conn.put_container(container)
resp_headers, containers = conn.get_account()
if container in containers:
    print("The container was created")


container = 'new-container'
with open('local.txt', 'r') as local:
    conn.put_object(
          container,
          'local_object.txt',
          contents=local,
          content_type='text/plain'
    )
obj = 'local_object.txt'
container = 'new-container'
resp_headers, obj_contents = conn.get_object(container, obj)
with open('local_copy.txt', 'w') as local:
    local.write(obj_contents)
```

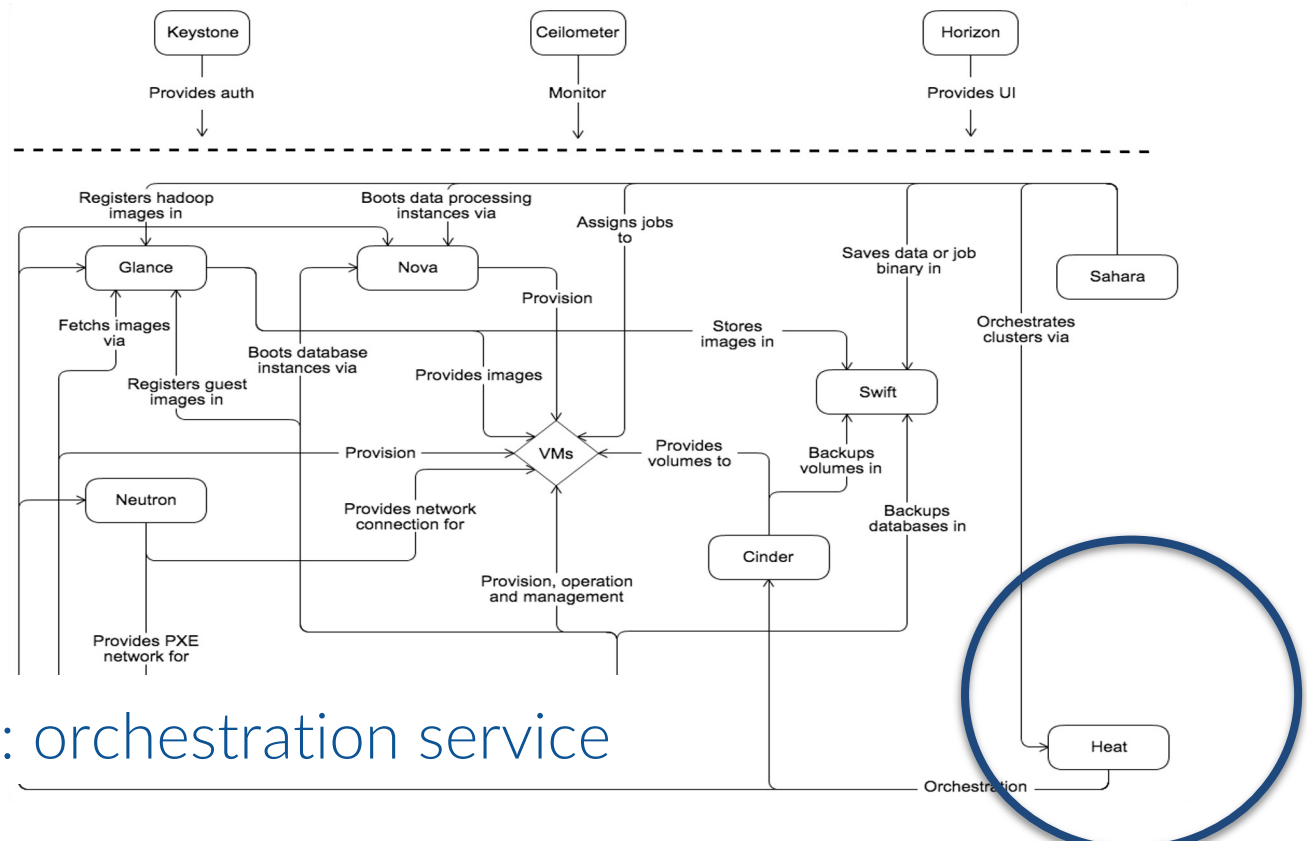
# OpenStack IaaS services view




HORIZON: web interface

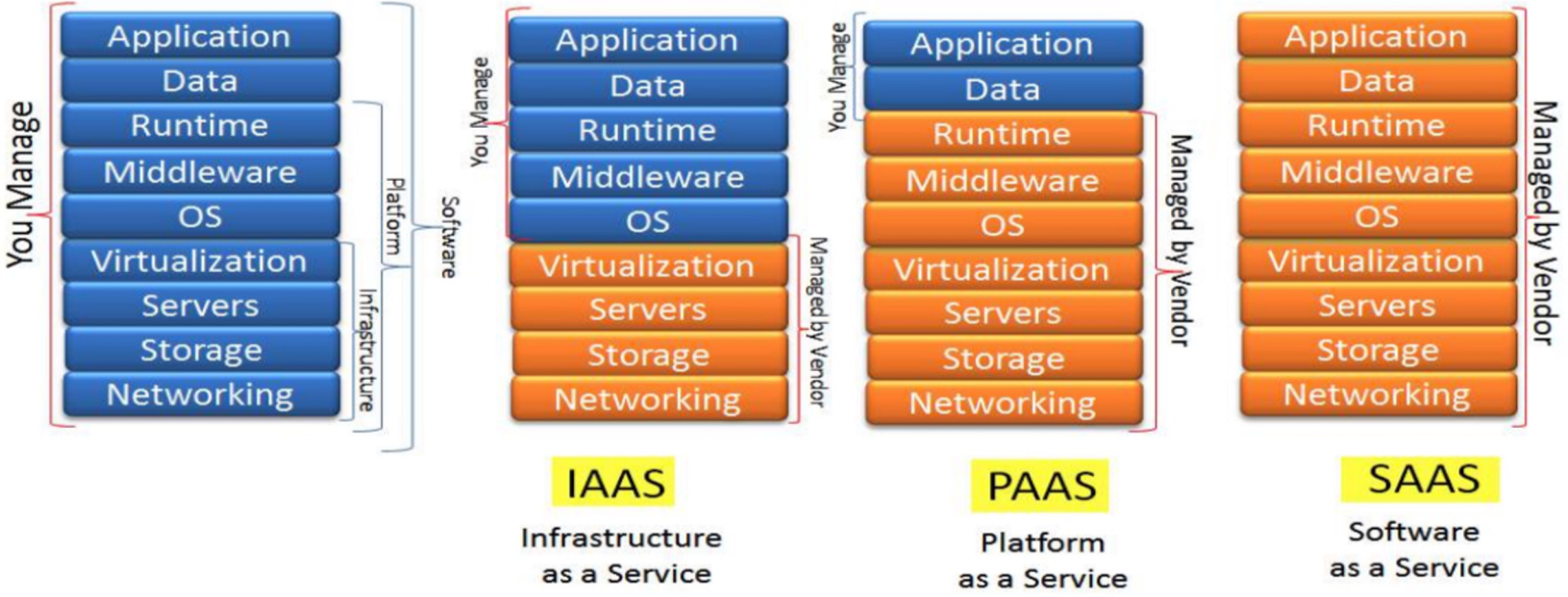

OpenStack Dashboard
Logged in as: glfoste
Settings
Help
Sign Out
Project
CURRENT PROJECT
sct
Manage Compute
Overview
Instances
Volumes
Images & Snapshots
Access & Security
Manage Network
Network Topology
Networks
Routers
Object Store
Containers
Overview
Limit Summary
Instances
Used 11 of 60
VCPUs
Used 58 of 200
RAM
Used 58.0 GB of 4.9 TB
Floating IPs
Used 57 of 60
Security Groups
Used 0 of 10
Select a period of time to query its usage:
From: 2014-07-01
To: 2014-07-11
Submit
The date should be in YYYY-mm-dd format.
Active Instances: 11 Active RAM: 58GB This Period's VCPU-Hours: 9.86 This Period's GB-Hours: 215.17
Download CSV Summary
Instance Name
VCPUs
Disk
RAM
Uptime
dminer-x86-ngz
1
10
2GB
1 week
dminer-x86-1
1
10
2GB
1 week
dminer-11.2-x86
1
10
2GB
1 week
dminer-x86-2
1
10
2GB
1 week
jlbutler-x86
8
40
4GB
3 days, 2 hours

# OpenStack IaaS services view




HEAT: orchestration service

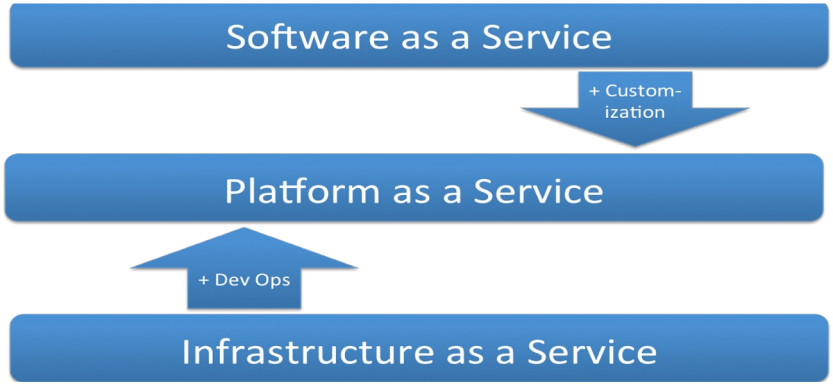
# Platform As A Service




Taken and Slightly changed from
https://cloudcelebrity.wordpress.com/2011/11/22/introduction-to-cloud-services-iaas-paas-saas/

# PaaS main concepts

- A platform is anything you can leverage to accomplish something in a simpler, faster, or otherwise better way than you could without.
- As a programmer, you leverage pre-existing code rather than starting from scratch and writing everything.
- The most well-known software platforms for desktop software are Windows and Mac OS



# Platform as a Service

PaaS provides a framework for:

- Application development: A PaaS platform either provides the means to use programs you create in a supported language or offers a visual development environment that writes the code for you.
- Collaboration: Many PaaS systems are set up to allow multiple individuals to work on the same projects.
- Data management: Tools are provided for accessing and using data in a data store. | Instrumentation, performance, and testing: Tools are available for measuring your applications and optimizing their performance.

# Platform as a Service

PaaS provides a framework for:

- Storage: Data can be stored in either the PaaS vendor's service or accessed from a third- party storage service.
- Transaction management: Many PaaS systems provide services such as transaction managers or brokerage service for maintaining transaction integrity.

PaaS systems exist to allow you to create software that can be hosted as SaaS systems

# Platform as a Service

PaaS cloud are designed to allow in developing robust, scalable, and hopefully portable applications.

On this list would be the following attributes:

- Separate of data management from the user interface I Reliance on cloud computing standards
- An integrated development environment (IDE)
- Lifecycle management tools
- Multi-tenant architecture support, security, and scalability
- Performance monitoring, testing, and optimization tools

A PaaS provides the tools needed to construct different types of applications that can work together in the same environment (e.g. mashup of multiple data sources).

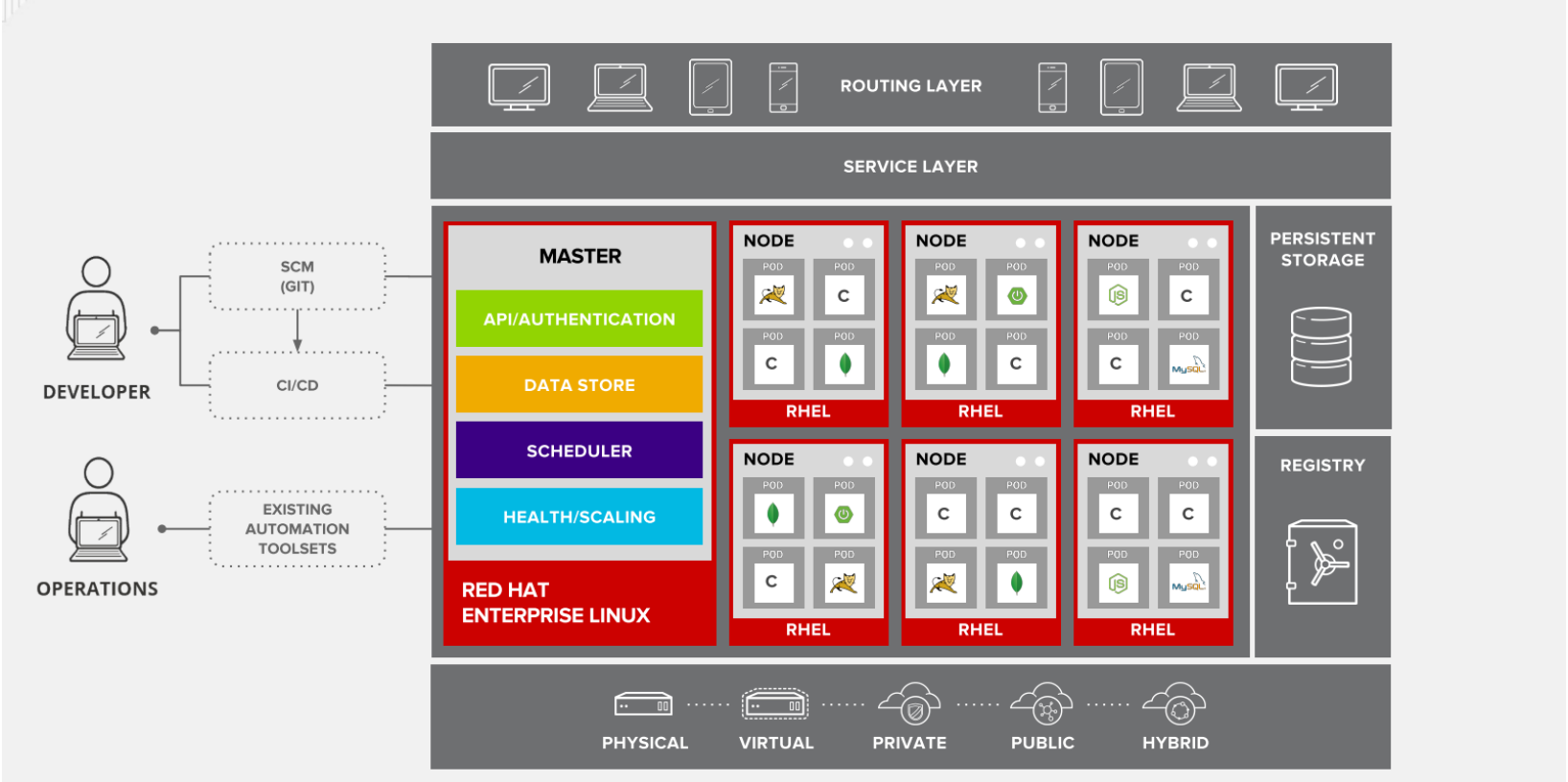
# In practice…

- To make it easier for you to run your website or web application no matter how much traffic it gets.
- You just deploy your application and the service figures out what to do with it.
- A platform as a service should handle scaling seamlessly for you so you can just focus on your website and the code running it.

# PaaS examples

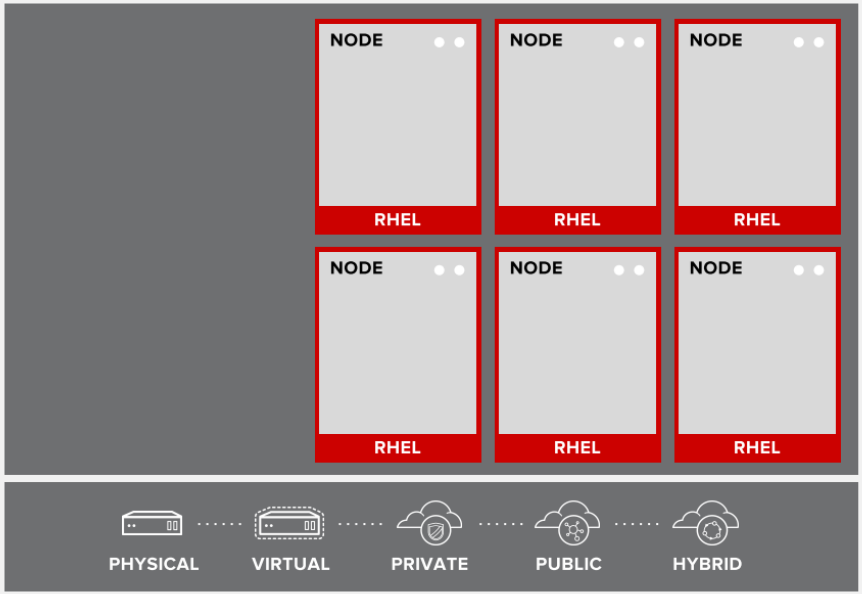
Google App Engine
OpenShift
Heroku
Force.com
Windows Azure
AWS Elastic Beanstalk
Apache Stratos
Magento Commerce
AWS Lamda
SAP Cloud

# OpenShift Architecture

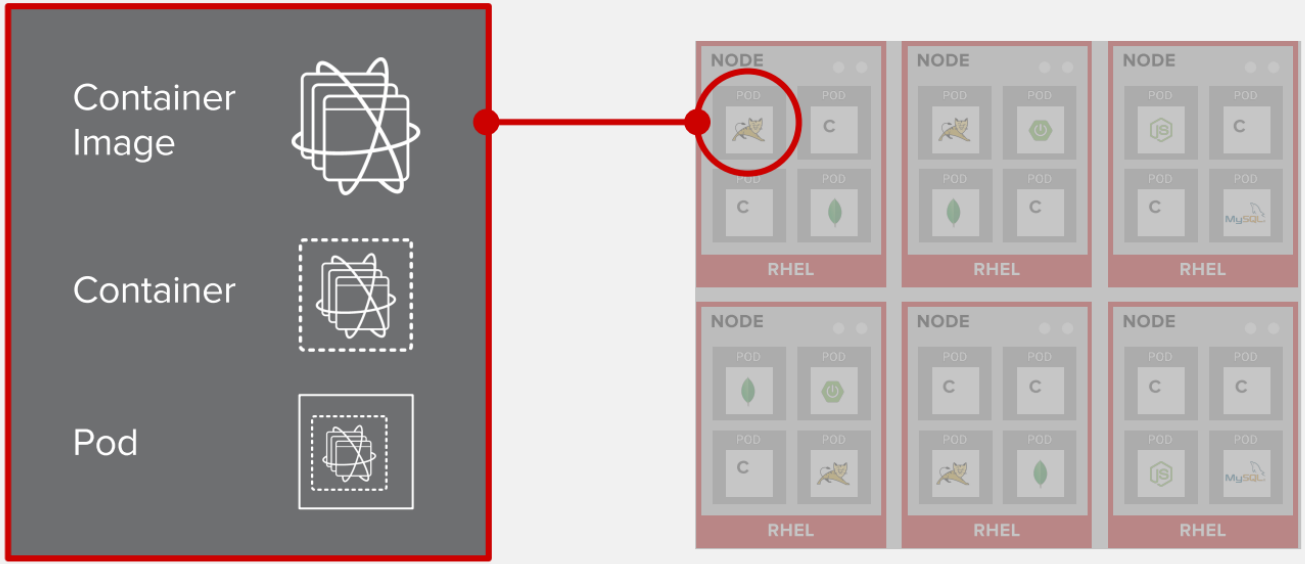


# You can choose an infrastructure

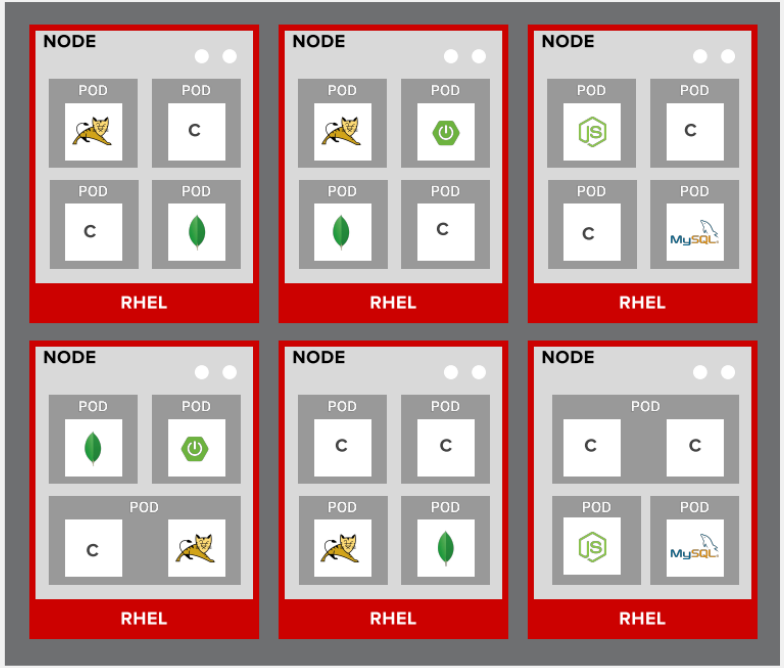
# Nodes| RHEL instance where apps run



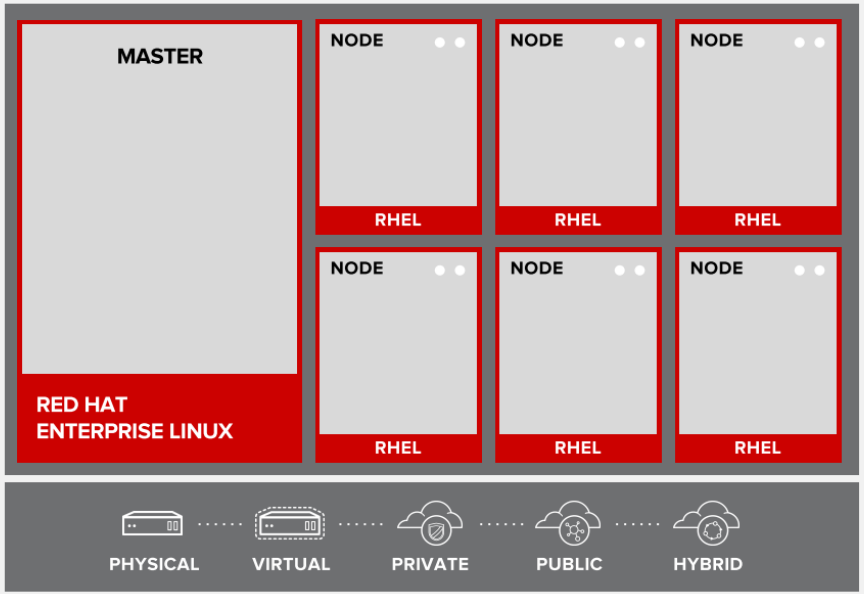
# Apps run in containers



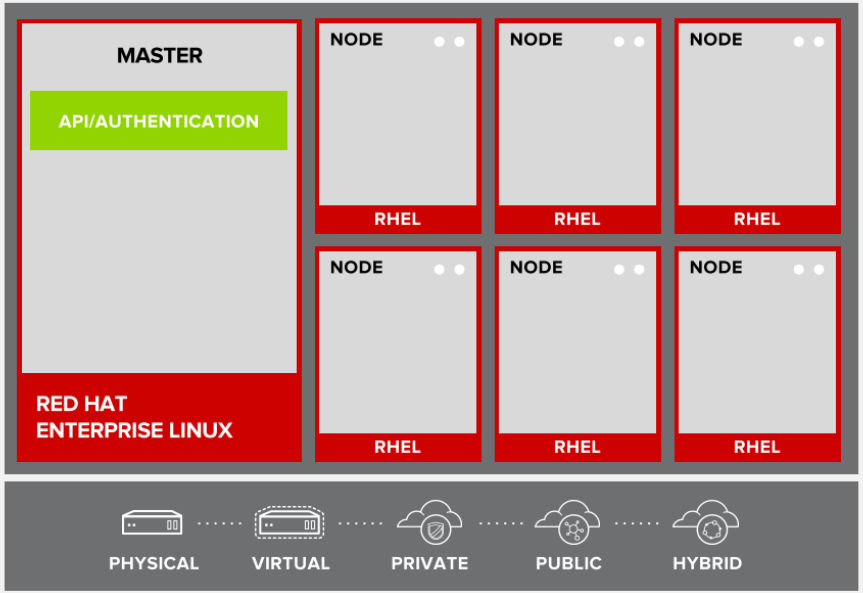
# PODs are units of orchstration



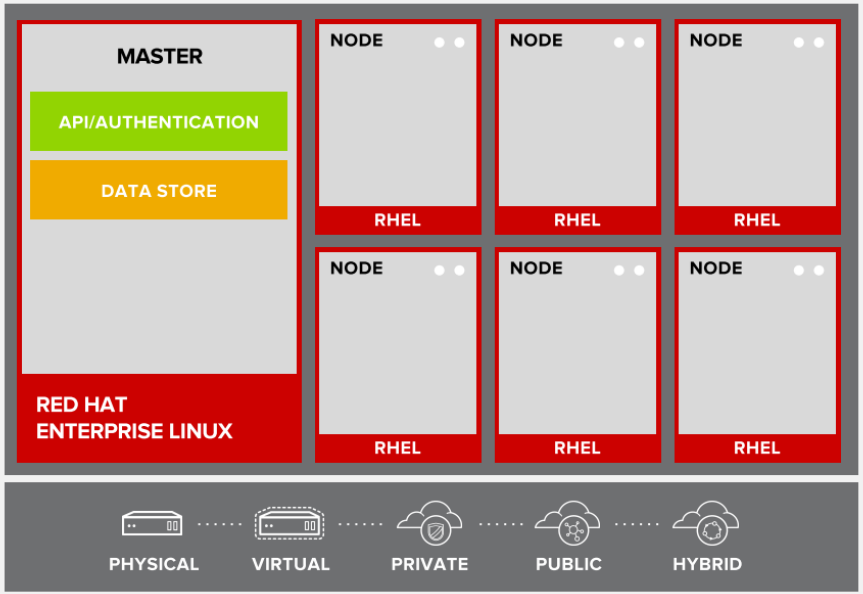
# Masters are control plane



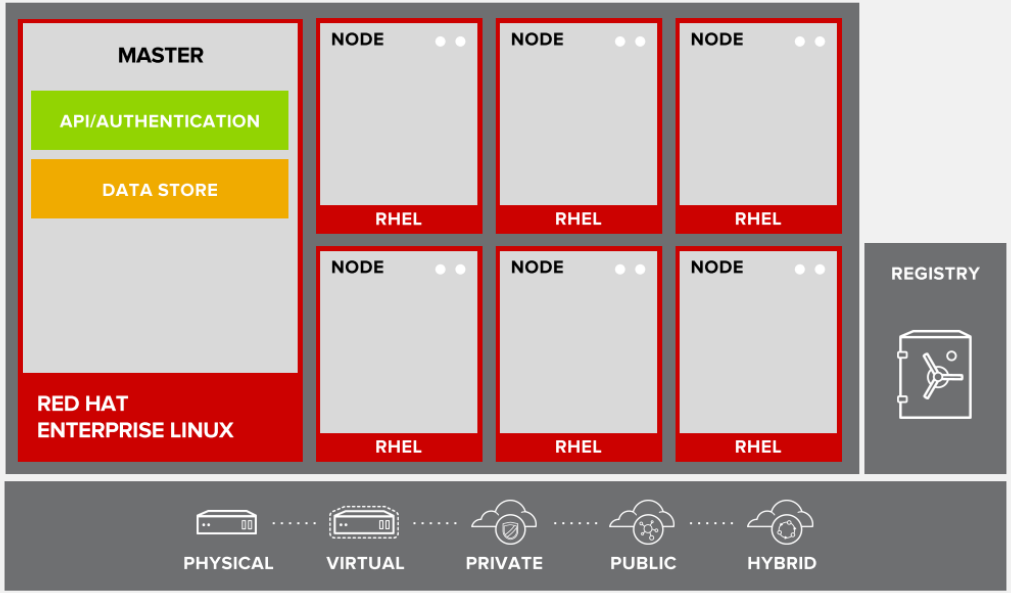
# API AND AUTHENTICATION



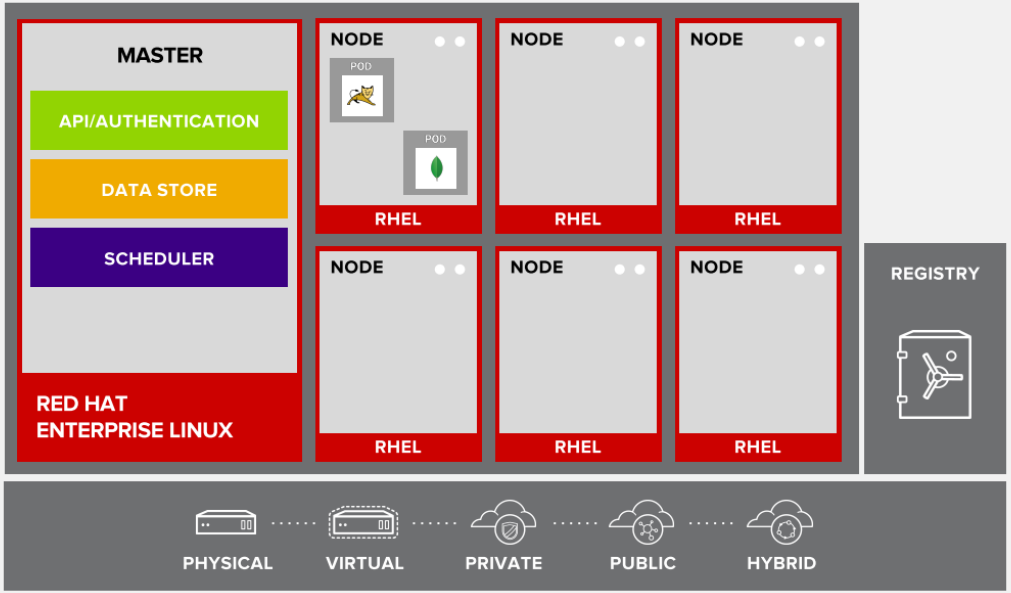
# DESIRED AND CURRENT STATE



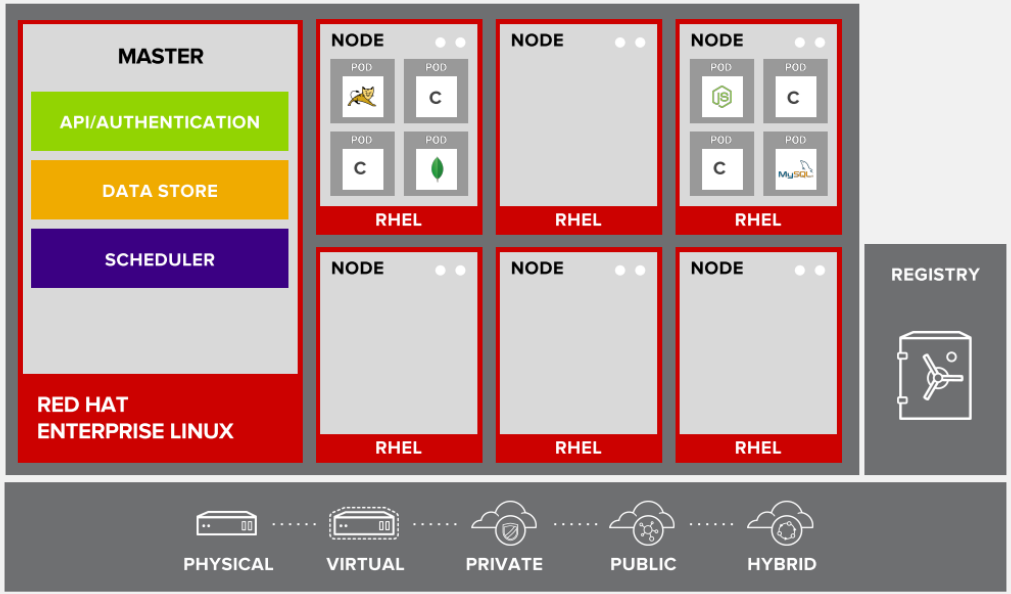
# INTEGRATED CONTAINER REGISTRY



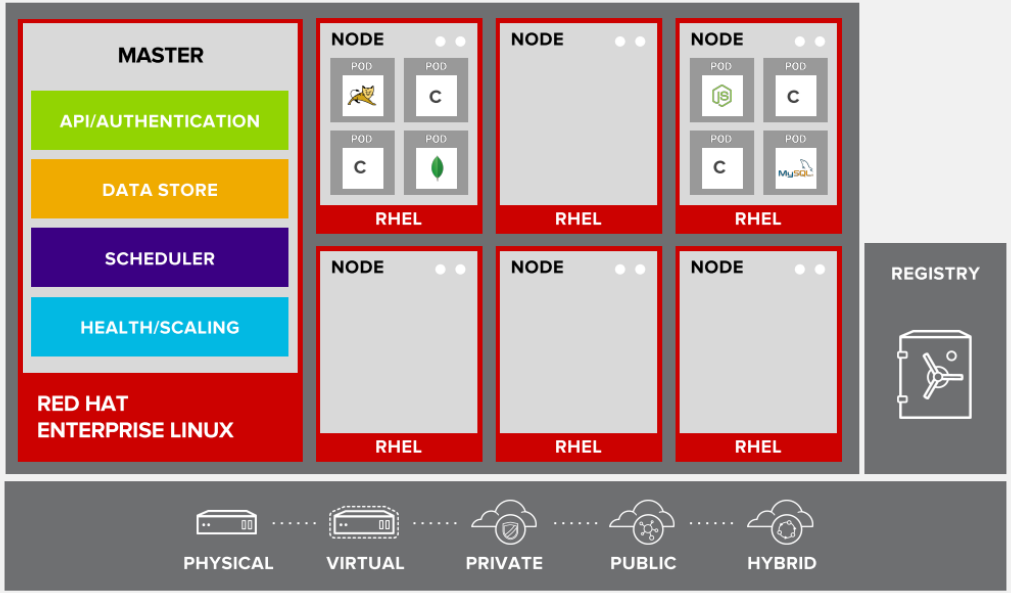
# ORCHESTRATION AND SCHEDULING



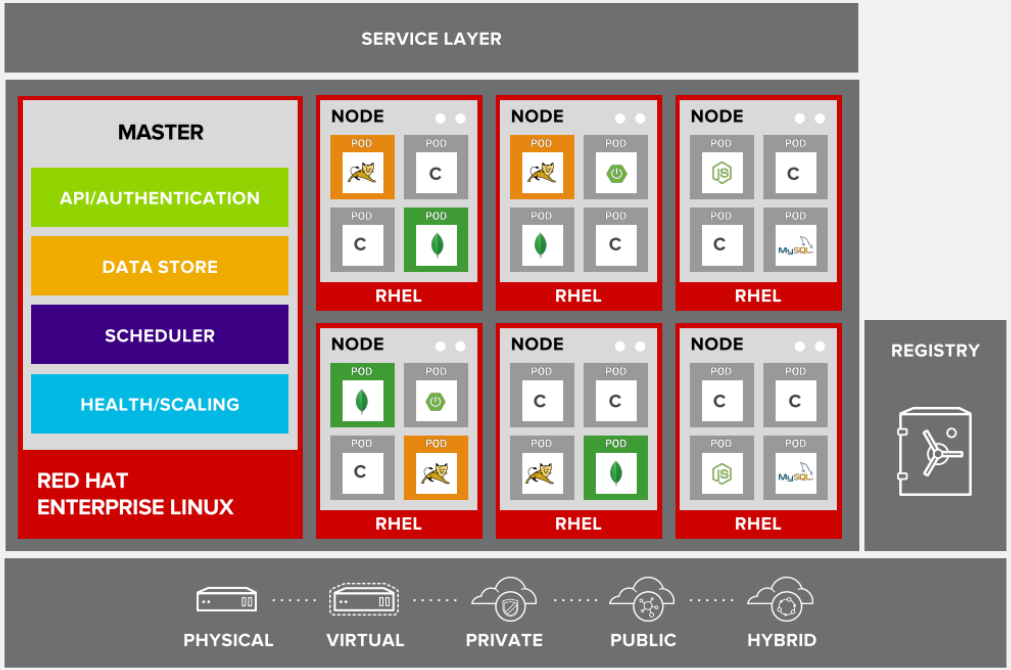
# PLACEMENT BY POLICY



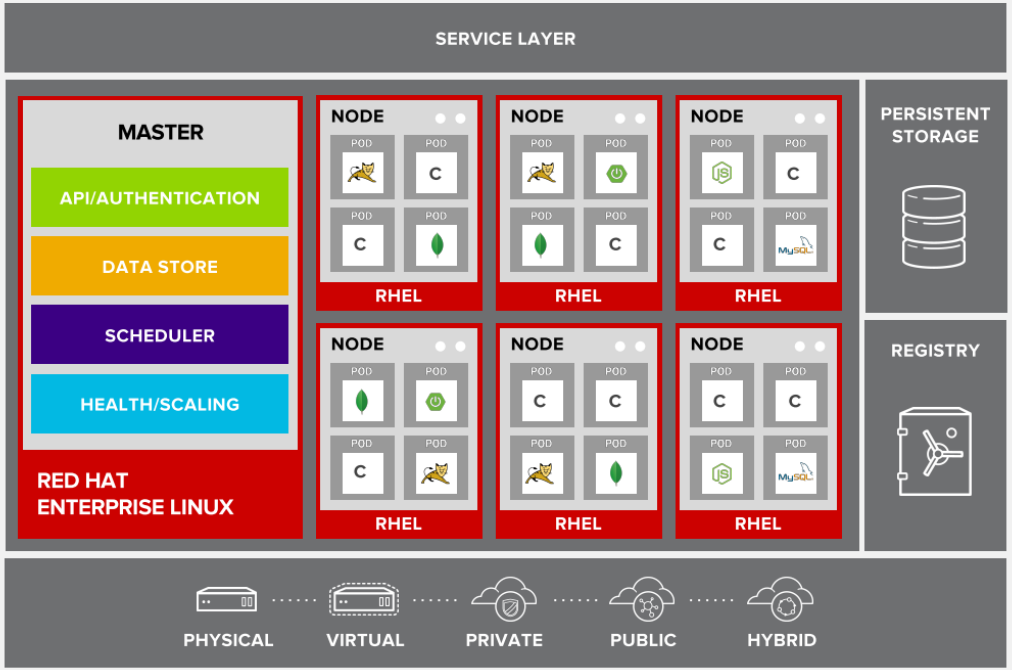
# AUTOSCALING PODS



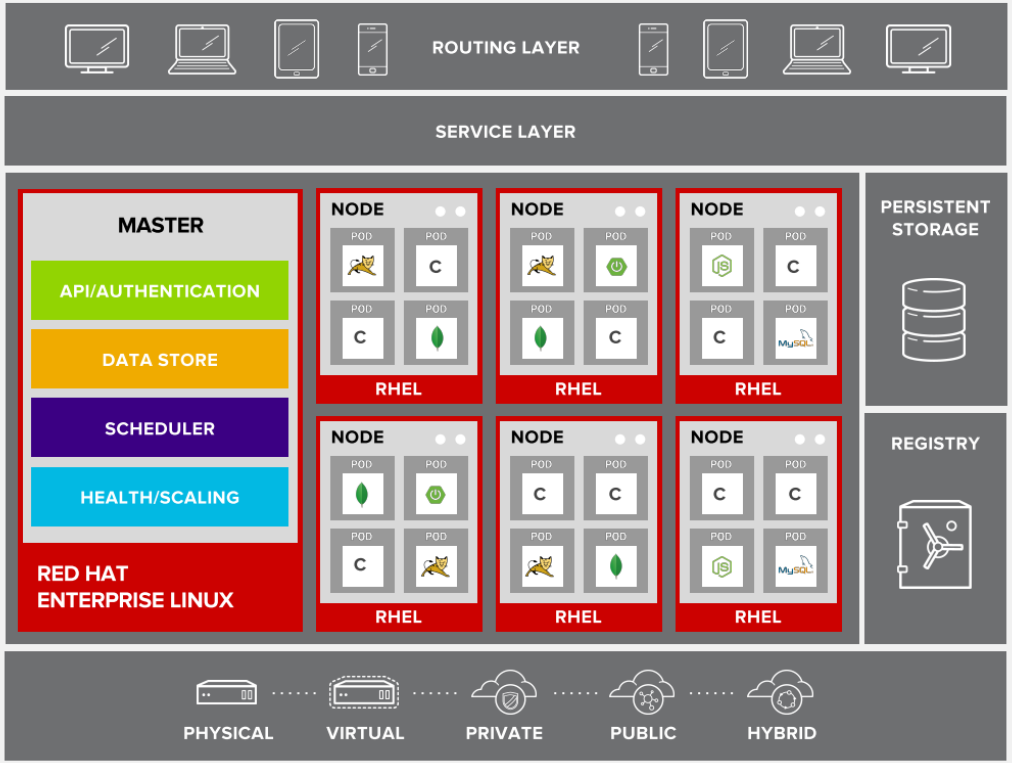
# SERVICE DISCOVERY



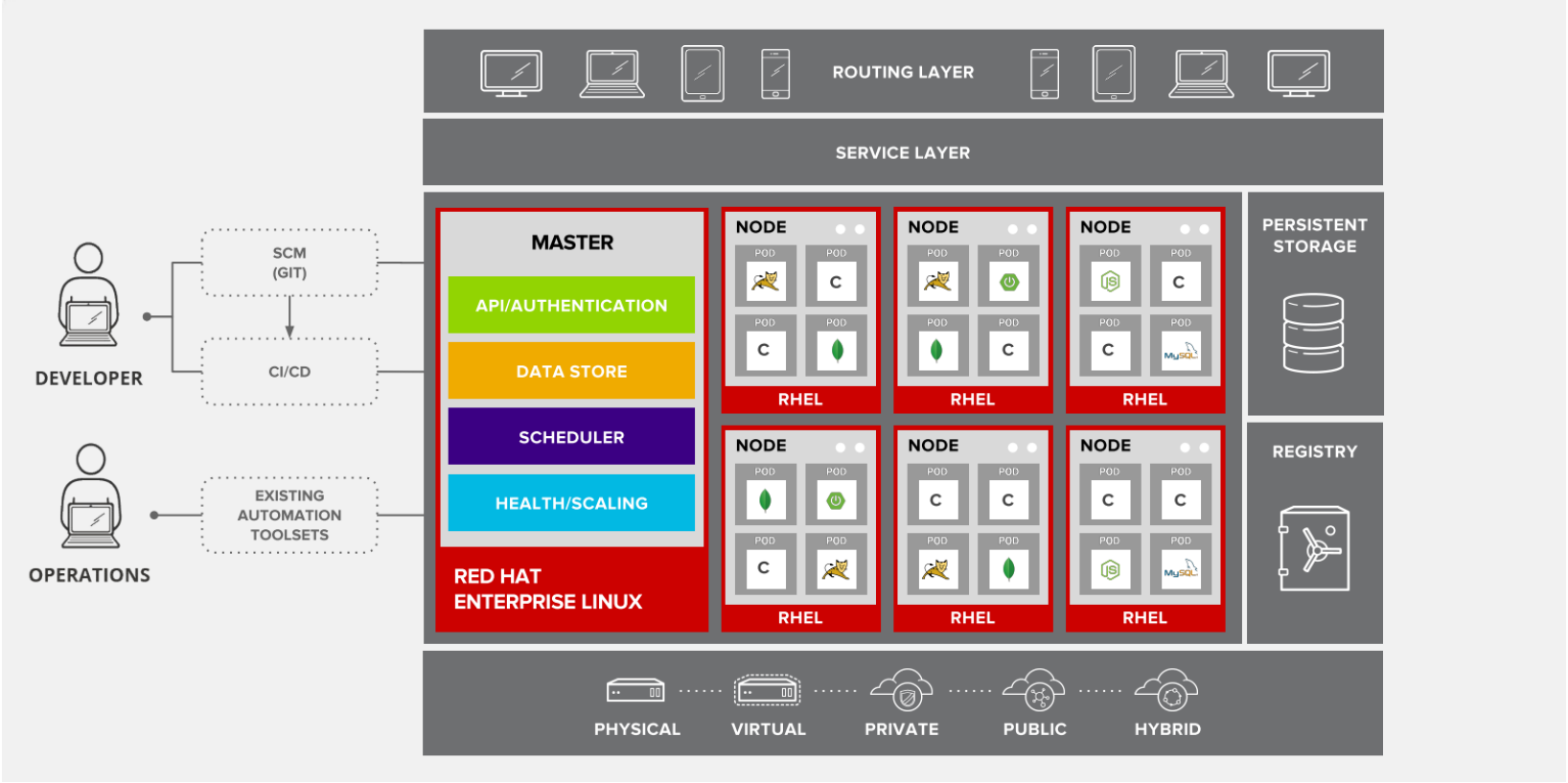
# PERSISTENT DATA IN CONTAINERS



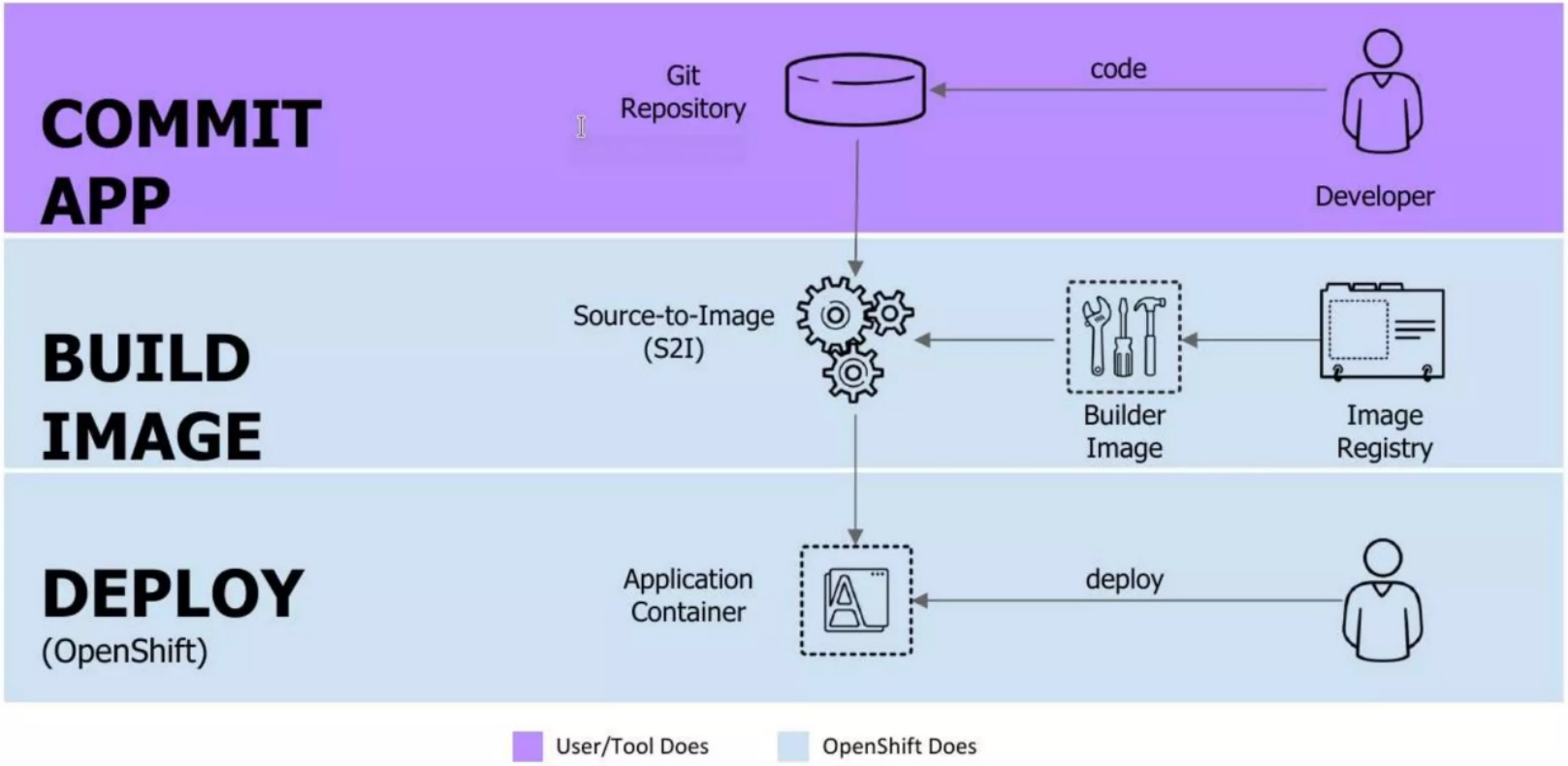
# ROUTING AND LOAD-BALANCING



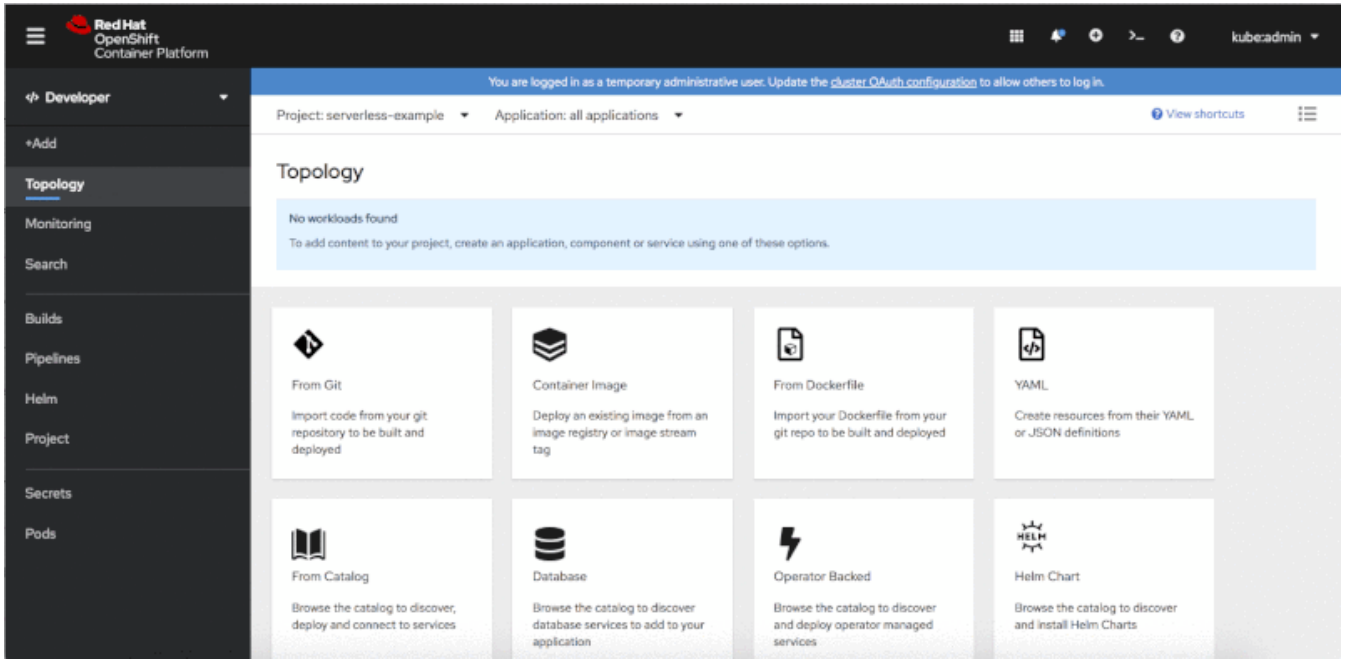
# ACCESS VIA WEB, CLI, IDE AND API



# How it works


COMMIT APP
Git Repository
code
Developer
BUILD IMAGE
Source-to-Image (S2I)
Builder Image
Image Registry
DEPLOY
(OpenShift)
Application Container
deploy
User/Tool Does
OpenShift Does

# console


Red Hat OpenShift Container Platform
kube:admin
Developer
You are logged in as a temporary administrative user. Update the cluster OAuth configuration to allow others to log in.
Project: serverless-example
Application: all applications
View shortcuts
+Add
Topology
Monitoring
Search
Builds
Pipelines
Helm
Project
Secrets
Pods
Topology
No workloads found
To add content to your project, create an application, component or service using one of these options.
From Git
Import code from your git repository to be built and deployed
Container Image
Deploy an existing image from an image registry or image stream tag
From Dockerfile
Import your Dockerfile from your git repo to be built and deployed
YAML
Create resources from their YAML or JSON definitions
From Catalog
Browse the catalog to discover, deploy and connect to services
Database
Browse the catalog to discover database services to add to your application
Operator Backed
Browse the catalog to discover and deploy operator managed services
Helm Chart
Browse the catalog to discover and install Helm Charts

# that’s all, have fun