From User to Developer: A Journey of Open-source Cloud Infra Projects

2023-05-25

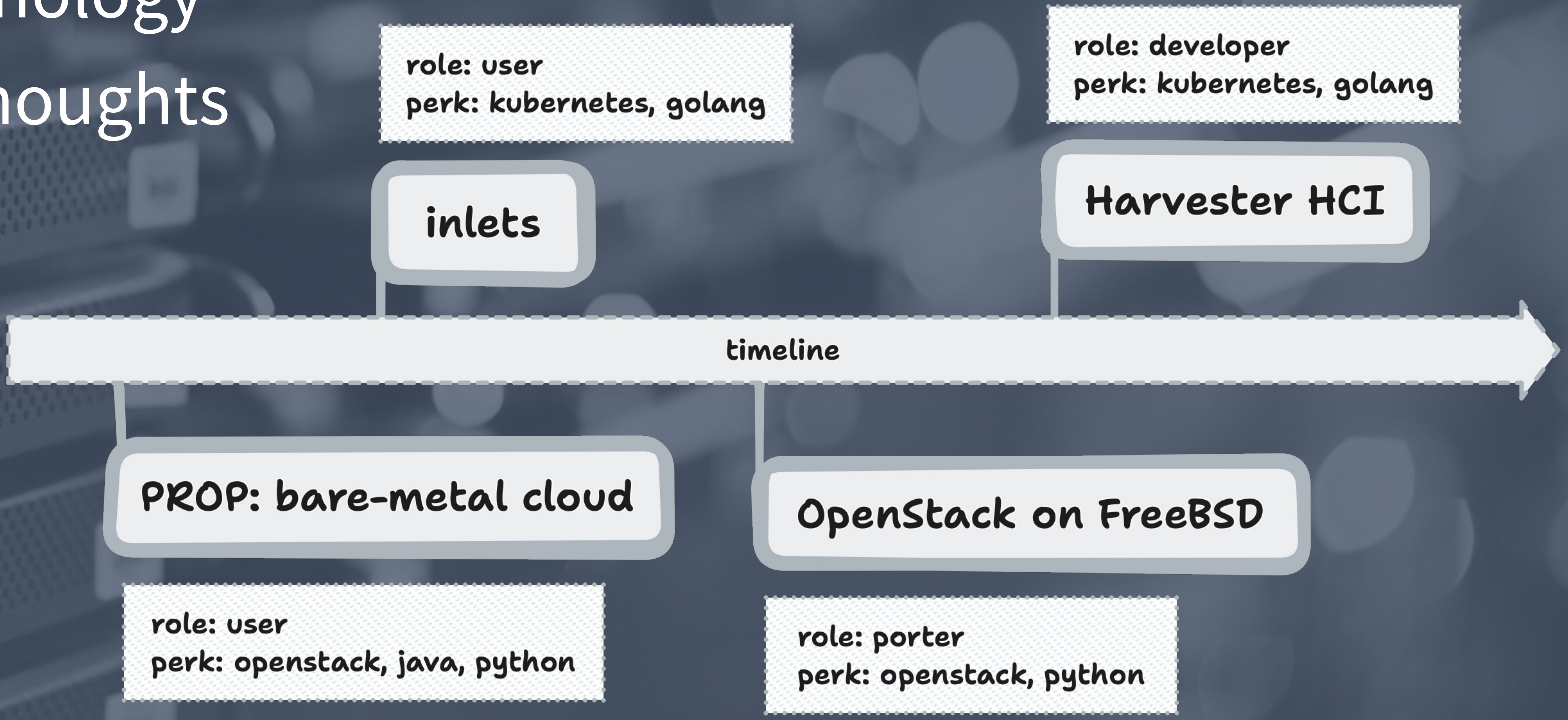
Chih-Hsin Chang

About Me

- Chih-Hsin Chang, aka Zespre（張至欣）
- Education
  - NCTU CS BS (Class of 2014)
  - NCTU CS MS (graduated in 2017)
  - NCTU CSCC member (2014 - 2017)
- Working experience
  - ITRI
  - FreeBSD Foundation (contractor)
  - SUSE Taiwan (current)
- Contact information
  - Blog https://blog.zespre.com
  - Twitter @starbops
  - Email chihhsin@cs.nctu.edu.tw

Prologue

- Role shifting in the open-source ecosystem
- Background technology
- Observations & thoughts



timeline

PROP: bare-metal cloud
role: user
perk: openstack, java, python

inlets
role: user
perk: kubernetes, golang

OpenStack on FreeBSD
role: porter
perk: openstack, python

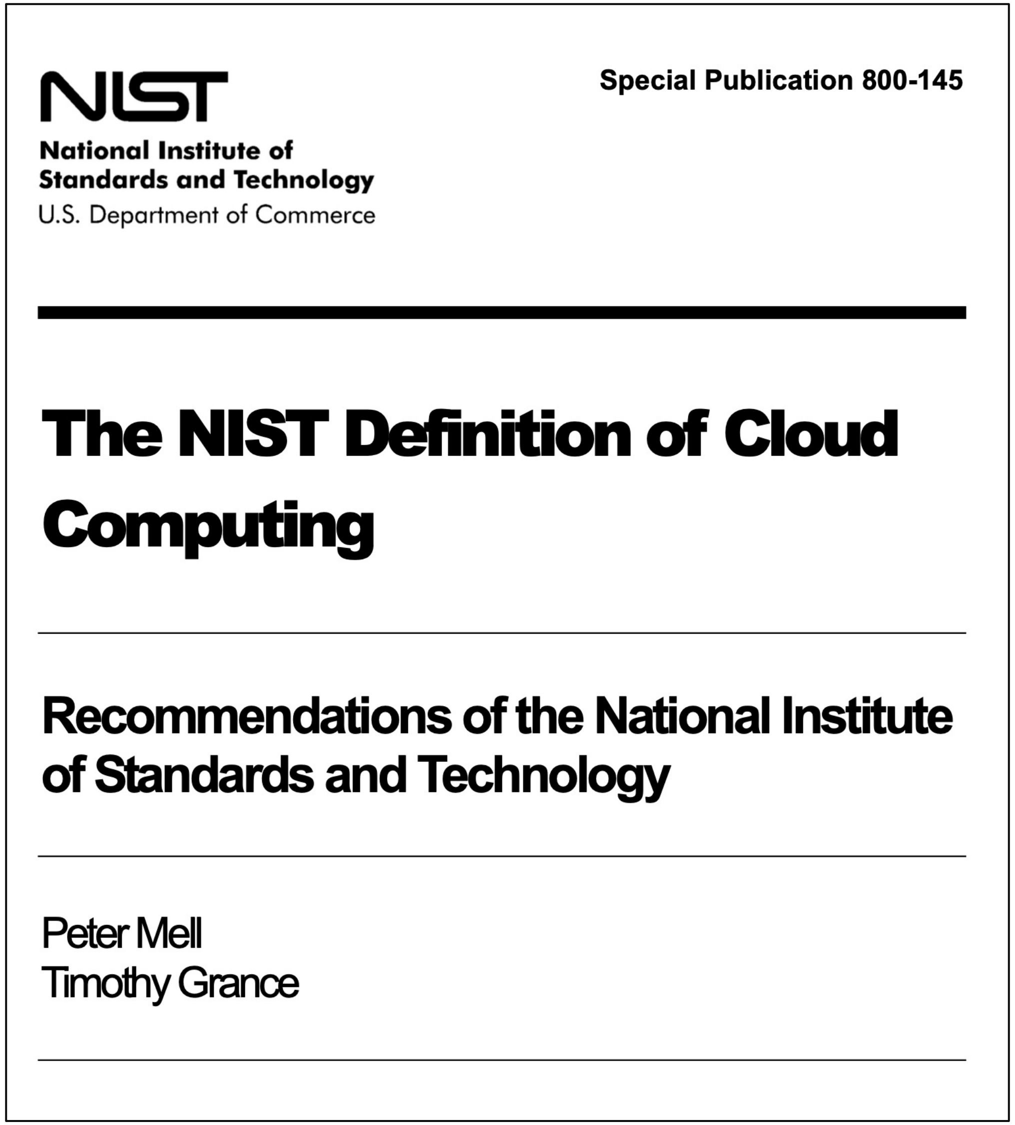
Harvester HCI
role: developer
perk: kubernetes, golang

Why Cloud?

- Buzzwords
  - Big Data
  - Machine Learning
  - Artificial Intelligence
  - Augmented/Virtual Reality
  - Internet of Things
  - Blockchain
  - …
- Cloud computing: the cornerstone of all of the above

The Baseline

- Essential characteristics
  - On-demand self-service
  - Broad network access
  - Resource pooling
  - Rapid elasticity
  - Measured service
- Service models
  - SaaS/PaaS/IaaS
- Deployment models
  - Private cloud
  - Public cloud



NIST
National Institute of
Standards and Technology
U.S. Department of Commerce

Special Publication 800-145

The NIST Definition of Cloud Computing

Recommendations of the National Institute of Standards and Technology

Peter Mell
Timothy Grance

https://nvlpubs.nist.gov/nistpubs/legacy/sp/nistspecialpublication800-145.pdf

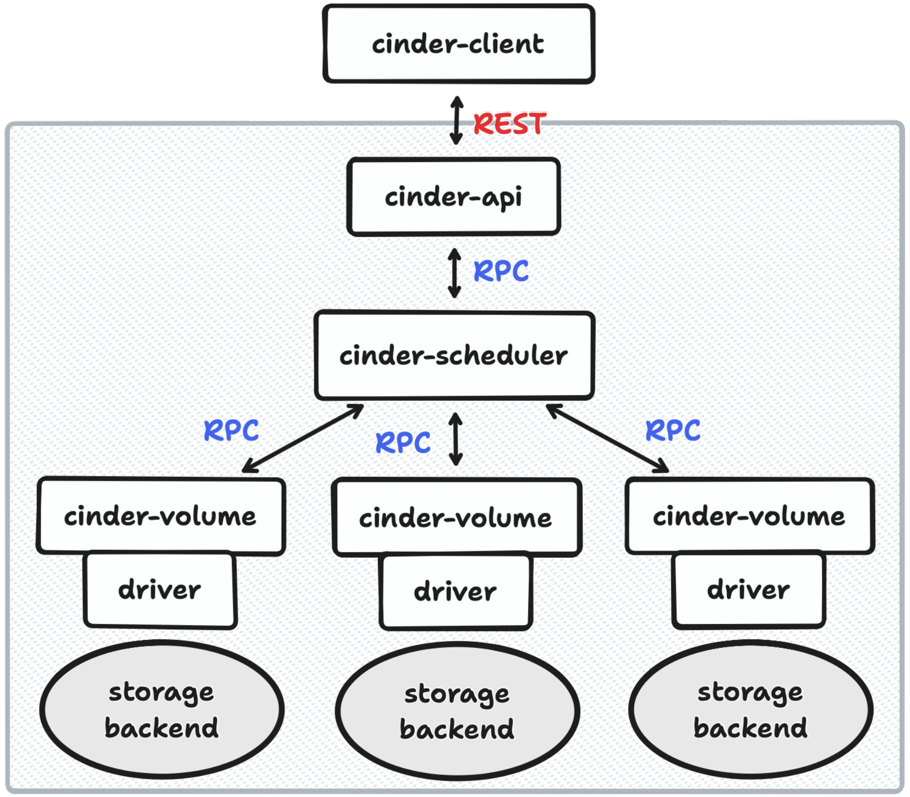
Glossary

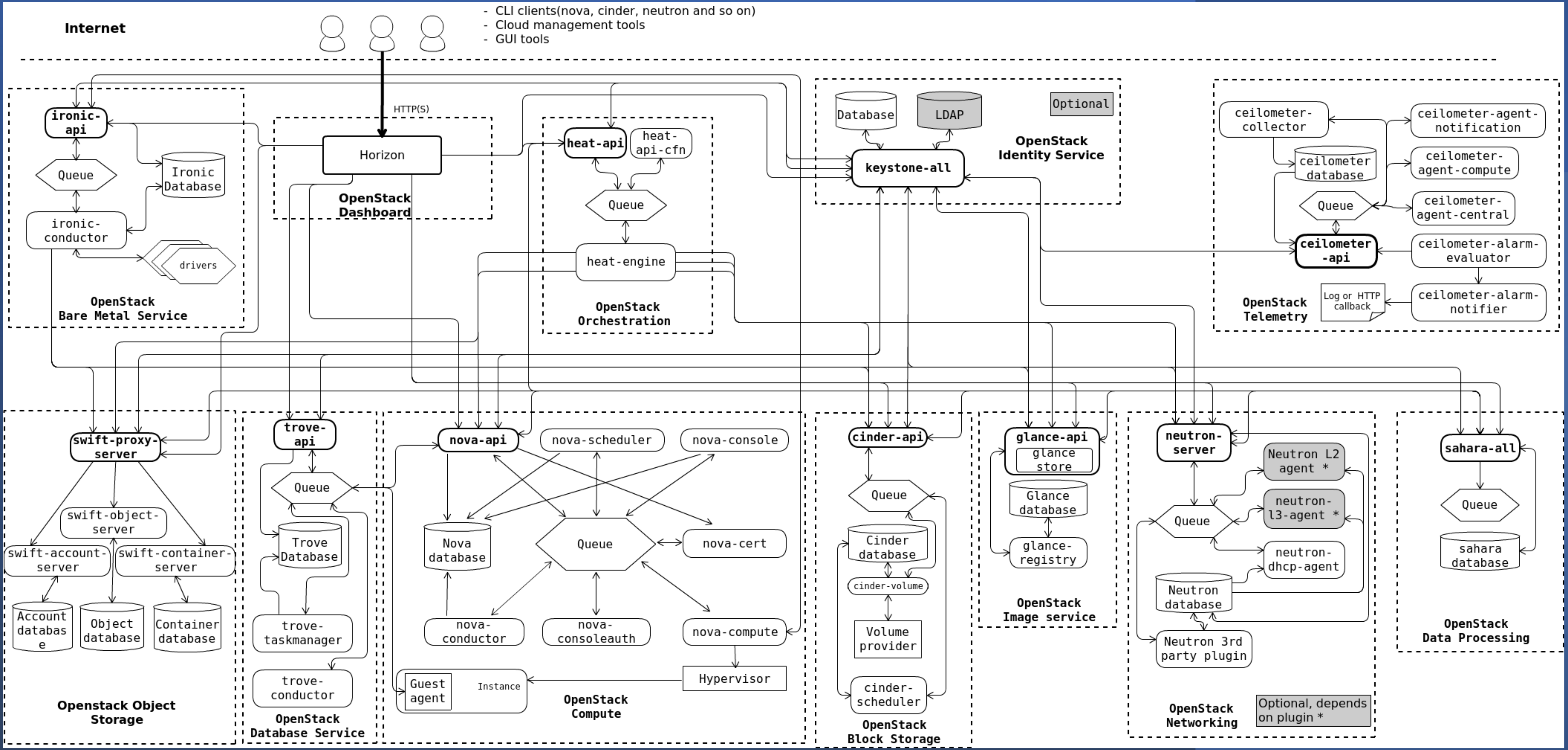
- Cloud computing – traditional, public clouds
  - Multi-tenant environment
  - Pay-as-you-go/subscribe model
  - Cost-efficient
  - Easy to scale up and down
- On-premises (on-prem) - private clouds
  - One-time investment
  - Full-control of infrastructure
  - Legal compliance
  - Ability to build with customized hardware

➢Debate https://world.hey.com/dhh/why-we-re-leaving-the-cloud-654b47e0

Introduction to OpenStack

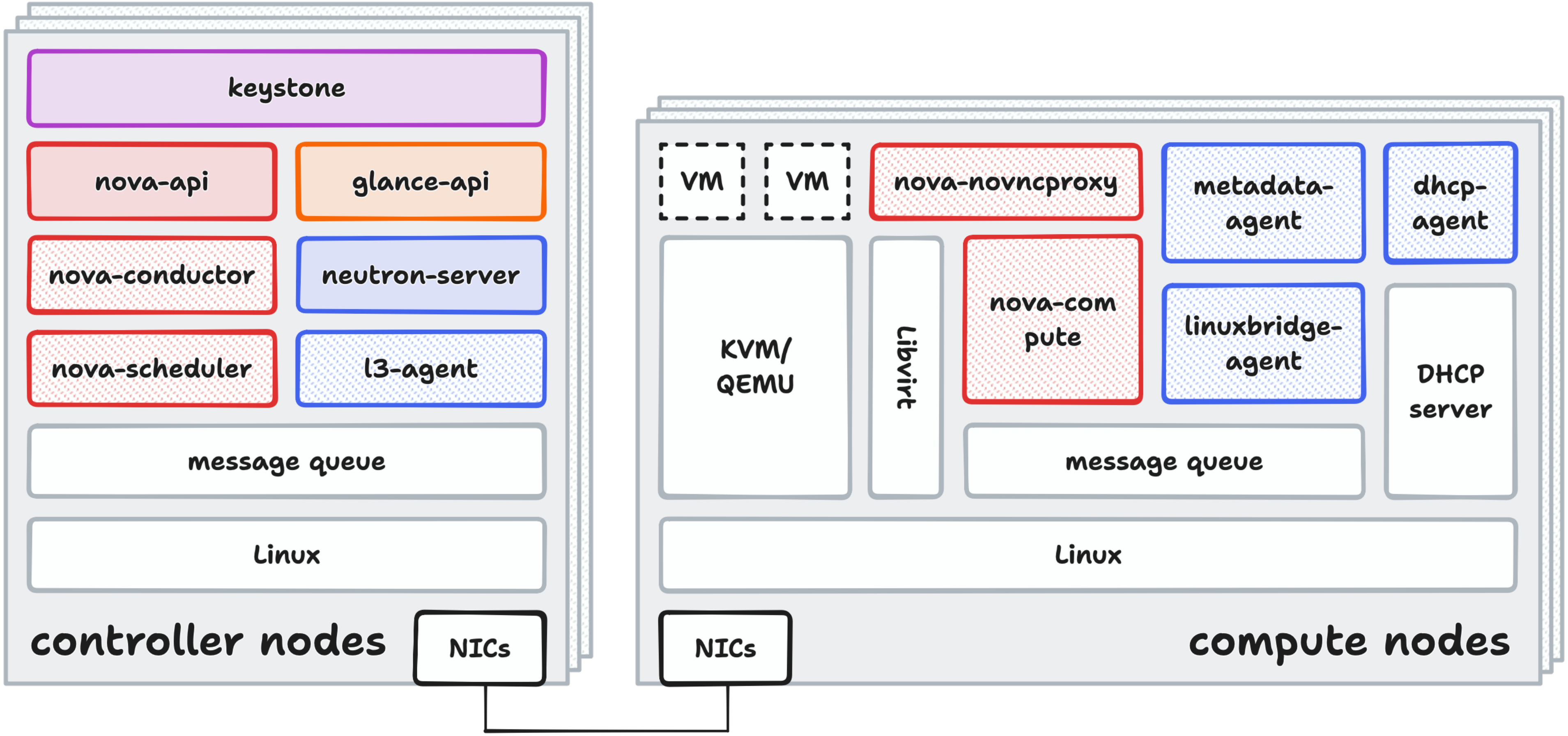
- An open-source cloud platform
- History
  - Launched by Rackspace & NASA in 2010
  - Managed by OpenStack Foundation
  - Versioning from A to Z
  - 2023.1.Antelope (latest)
- Communication
  - Inter-project: RESTful APIs
  - Intra-project: RPC APIs
- Common libraries: Oslo





https://docs.openstack.org/install-guide/get-started-logical-architecture.html

Bird's-eye View of OpenStack



keystone

nova-api

glance-api

nova-conductor

neutron-server

nova-scheduler

l3-agent

message queue

Linux

controller nodes

NICs

VM

VM

nova-novncproxy

metadata-agent

dhcp-agent

KVM/ QEMU

Libvirt

nova-com pute

linuxbridge-agent

DHCP server

message queue

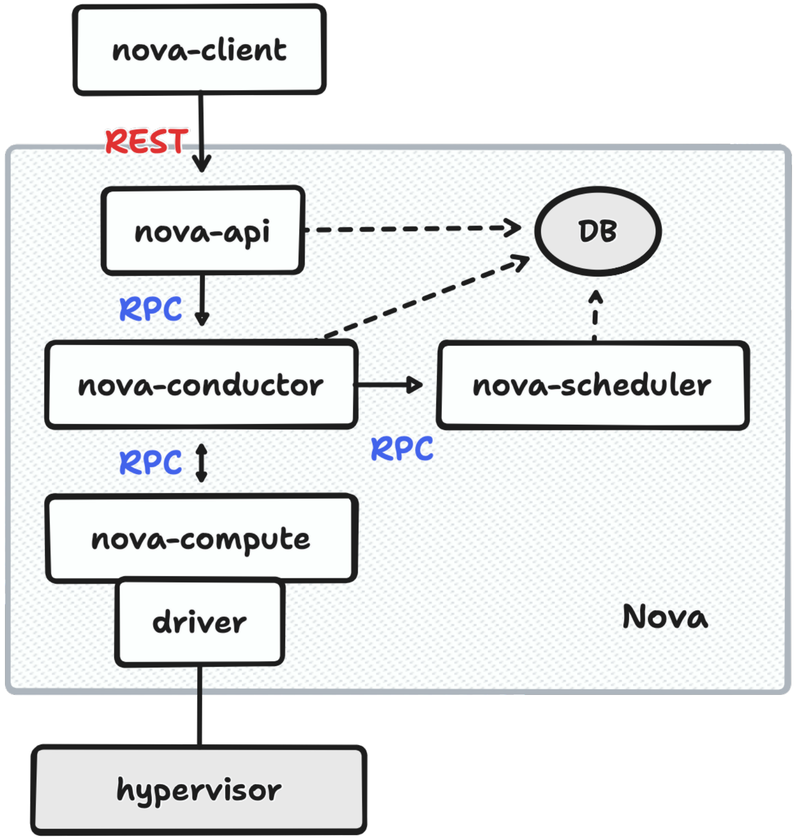
Linux

NICs

compute nodes

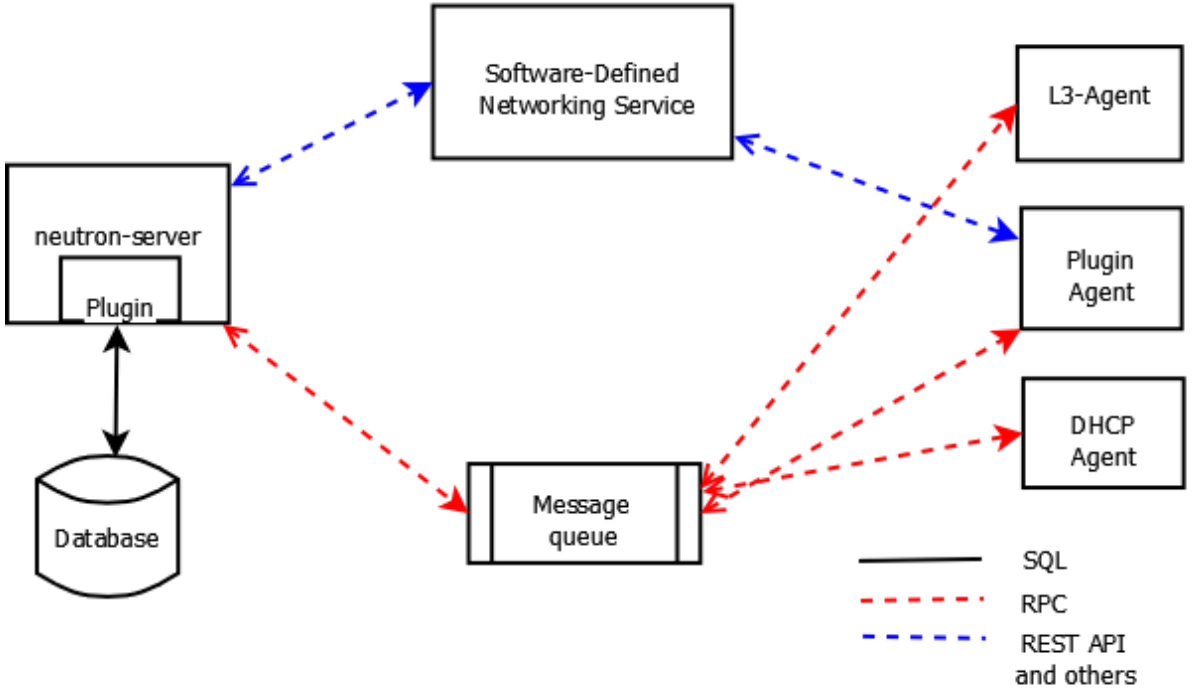
Compute – Nova

- Provisioning/managing compute instances
  - Virtual machines
  - Bare-metal servers
  - System containers
- Virtualization driver
  - HyperV
  - Libvirt*
    - QEMU/KVM
  - Vmware
  - XenServer
  - Fake*
  - Bare-metal (Ironic)

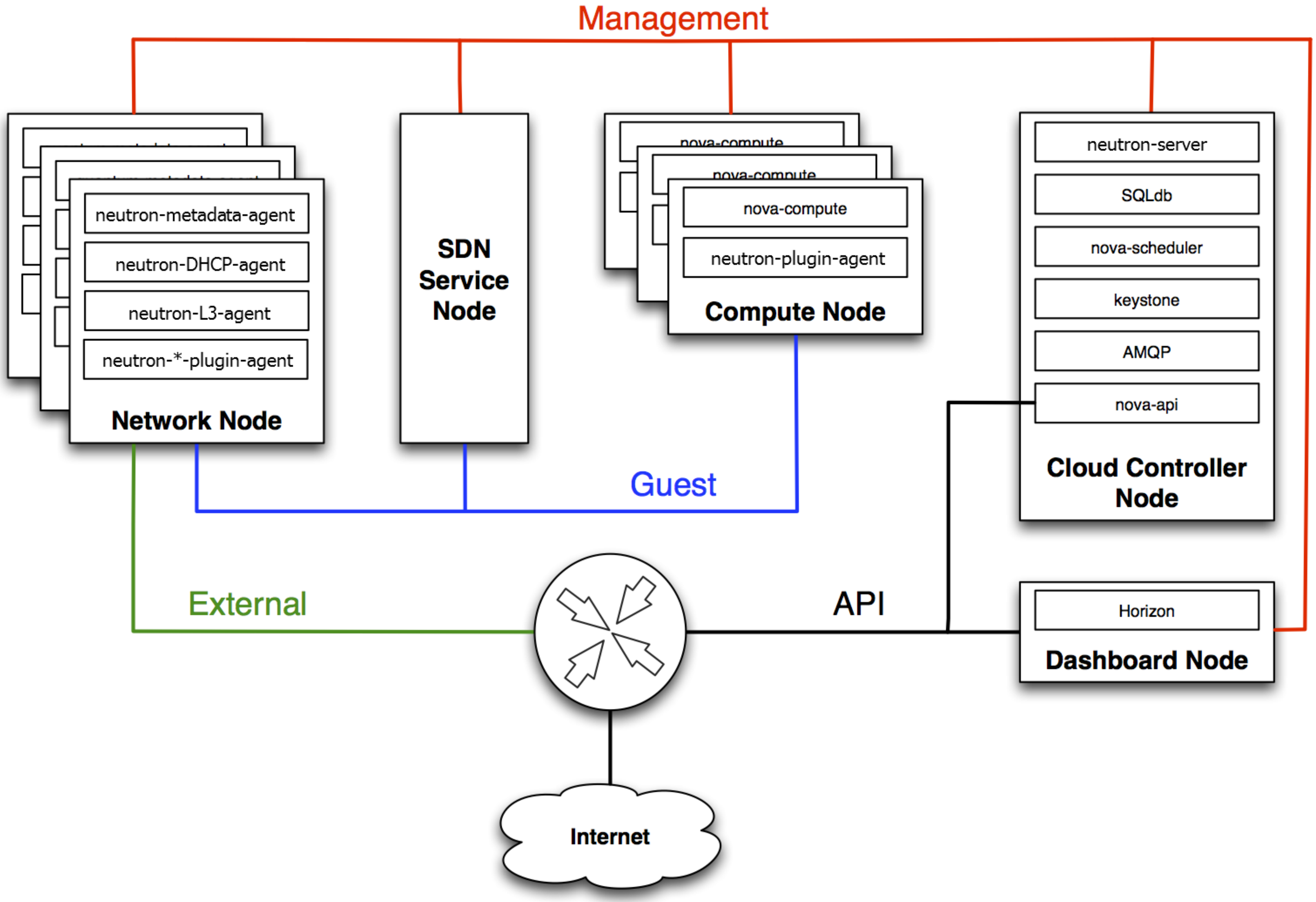


Networking – Neutron

- Network connectivity as a service
- Modular L2 (ML2) framework
  - Type driver
  - Mechanism driver
- L3
  - Routing
  - NAT
- Other networking services
  - Security groups (firewalling)
  - DHCP
  - Metadata



https://docs.openstack.org/security-guide/networking/architecture.html



https://docs.openstack.org/security-guide/networking/architecture.html

Some Thoughts

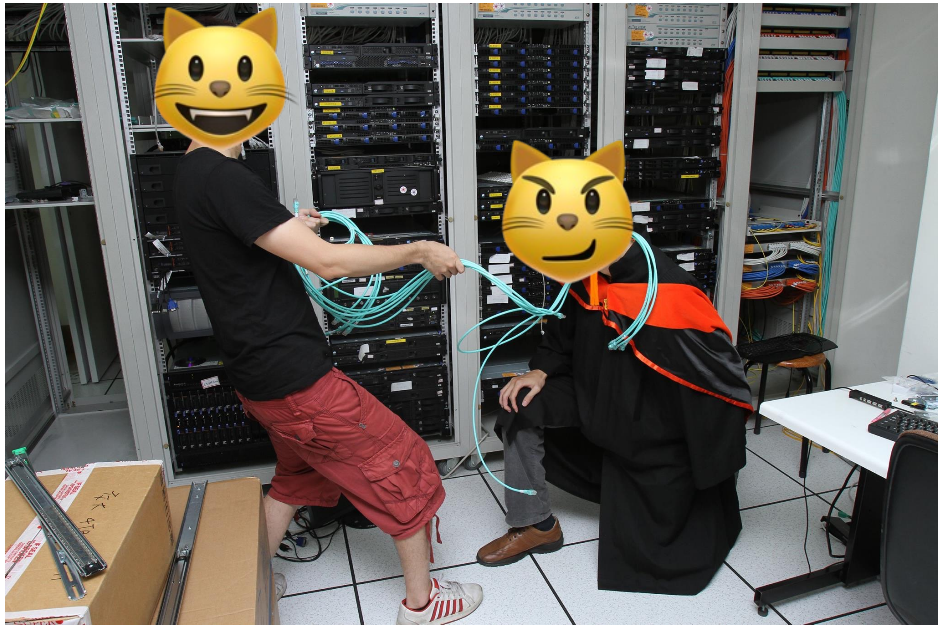
- Versatility - Big Tent
  - Cinder/Swift/Trove/Ironic/Magnum…
- The confusion caused by the complexity
  - Required/optional components
  - Deployment methods
    - Vanilla https://docs.openstack.org/install-guide/
    - DevStack – OpenStack quick scaffolding for dev environments
    - OpenStack-Ansible – Ansible playbooks for OpenStack deployment
    - Kolla – Containerized deployment of OpenStack
    - TripleO – OpenStack on OpenStack
- Mature workflows for contributor
  - https://wiki.openstack.org/wiki/How_To_Contribute

Pain Points

Pain Points

- Lots of bare-metal servers to operate
- Lots of applications/services to maintain

➢Lots of documents and urban myths



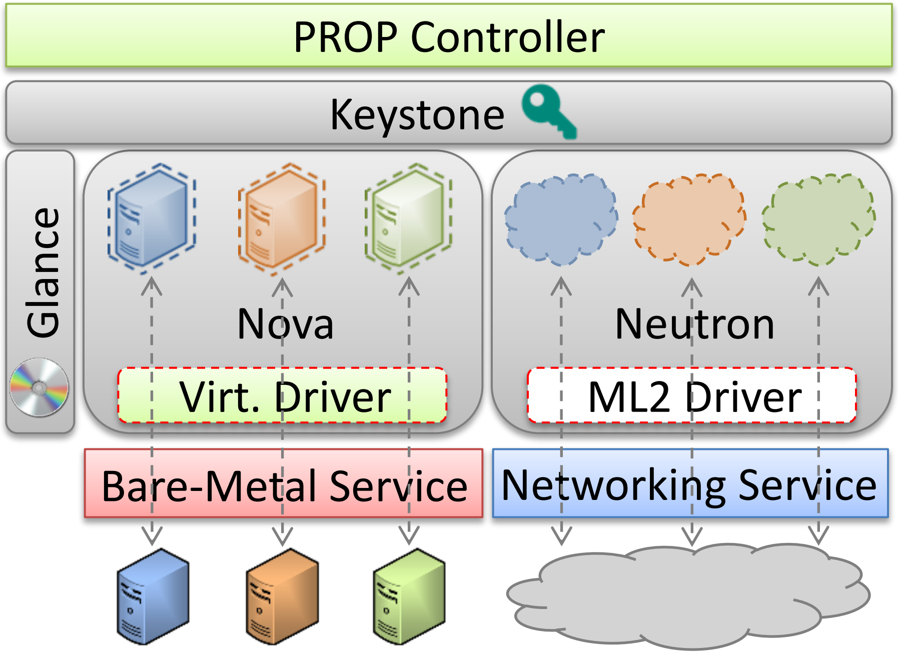
Bare-metal Cloud

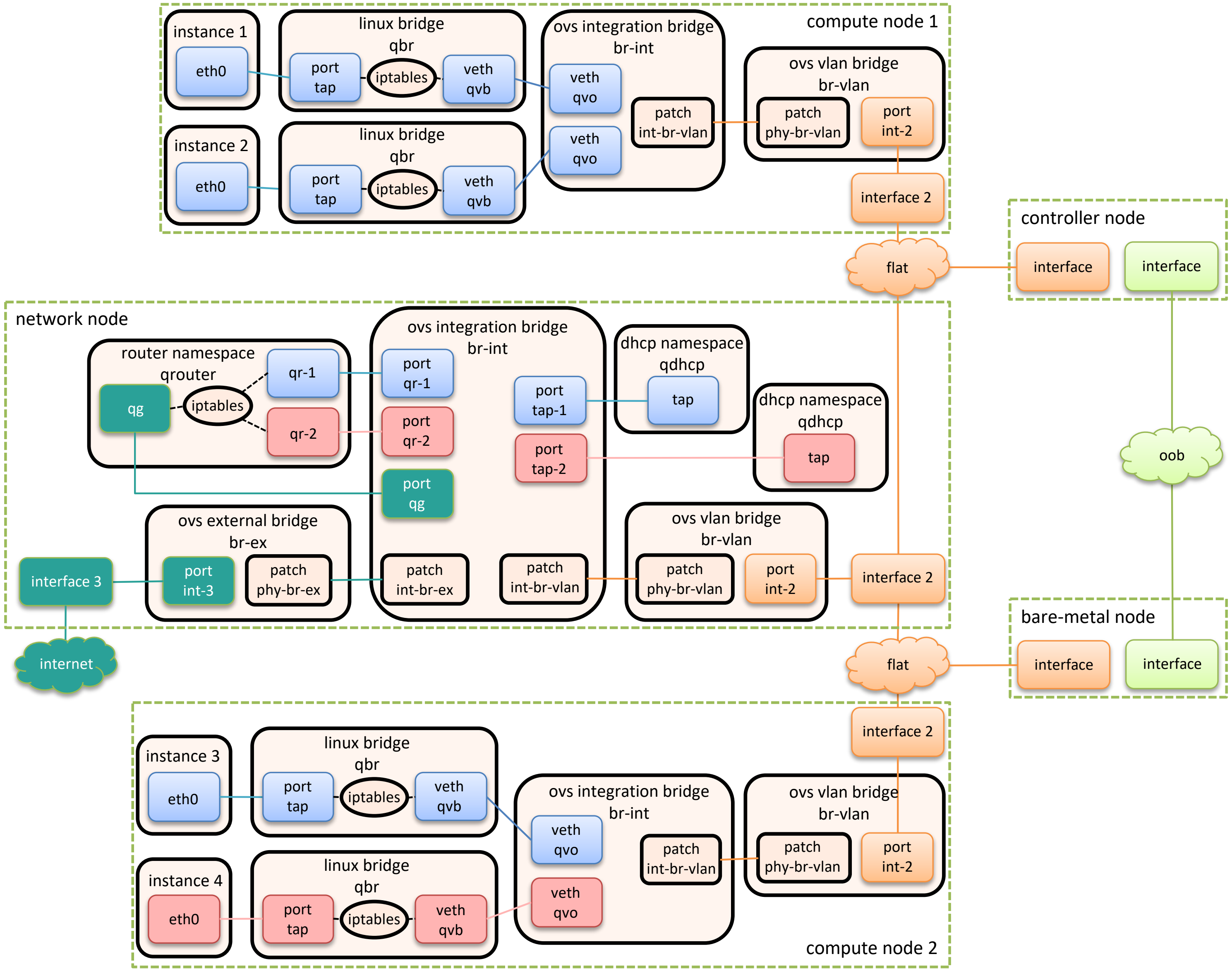
- Automation automation automation
- A cloud-like experience of a bunch of bare-metal machines

The Art of Integration (1)

- **OpenStack** as core, plus
  - Proprietory **bare-metal provisioning** software
  - Proprietory **SDN** controller (based on OpenDaylight)
  - Proprietory **distributed storage** software (based on Hadoop)
  - Proprietory **monitoring** software (based on Zenoss)

The Art of Integration (2)




compute node 1
instance 1
eth0
linux bridge
qbr
port
tap
iptables
veth
qvb
instance 2
eth0
linux bridge
qbr
port
tap
iptables
veth
qvb
ovs integration bridge
br-int
veth
qvo
veth
qvo
patch
int-br-vlan
ovs vlan bridge
br-vlan
patch
phy-br-vlan
port
int-2
interface 2
flat
controller node
interface
interface
network node
router namespace
qrouter
qg
iptables
qr-1
qr-2
ovs integration bridge
br-int
port
qr-1
port
qr-2
port
qg
port
tap-1
port
tap-2
dhcp namespace
qdhcp
tap
dhcp namespace
qdhcp
tap
oob
ovs external bridge
br-ex
interface 3
port
int-3
patch
phy-br-ex
patch
int-br-ex
patch
int-br-vlan
ovs vlan bridge
br-vlan
patch
phy-br-vlan
port
int-2
interface 2
internet
flat
bare-metal node
interface
interface
interface 2
instance 3
eth0
linux bridge
qbr
port
tap
iptables
veth
qvb
ovs integration bridge
br-int
veth
qvo
patch
int-br-vlan
ovs vlan bridge
br-vlan
patch
phy-br-vlan
port
int-2
instance 4
eth0
linux bridge
qbr
port
tap
iptables
veth
qvb
veth
qvo
compute node 2

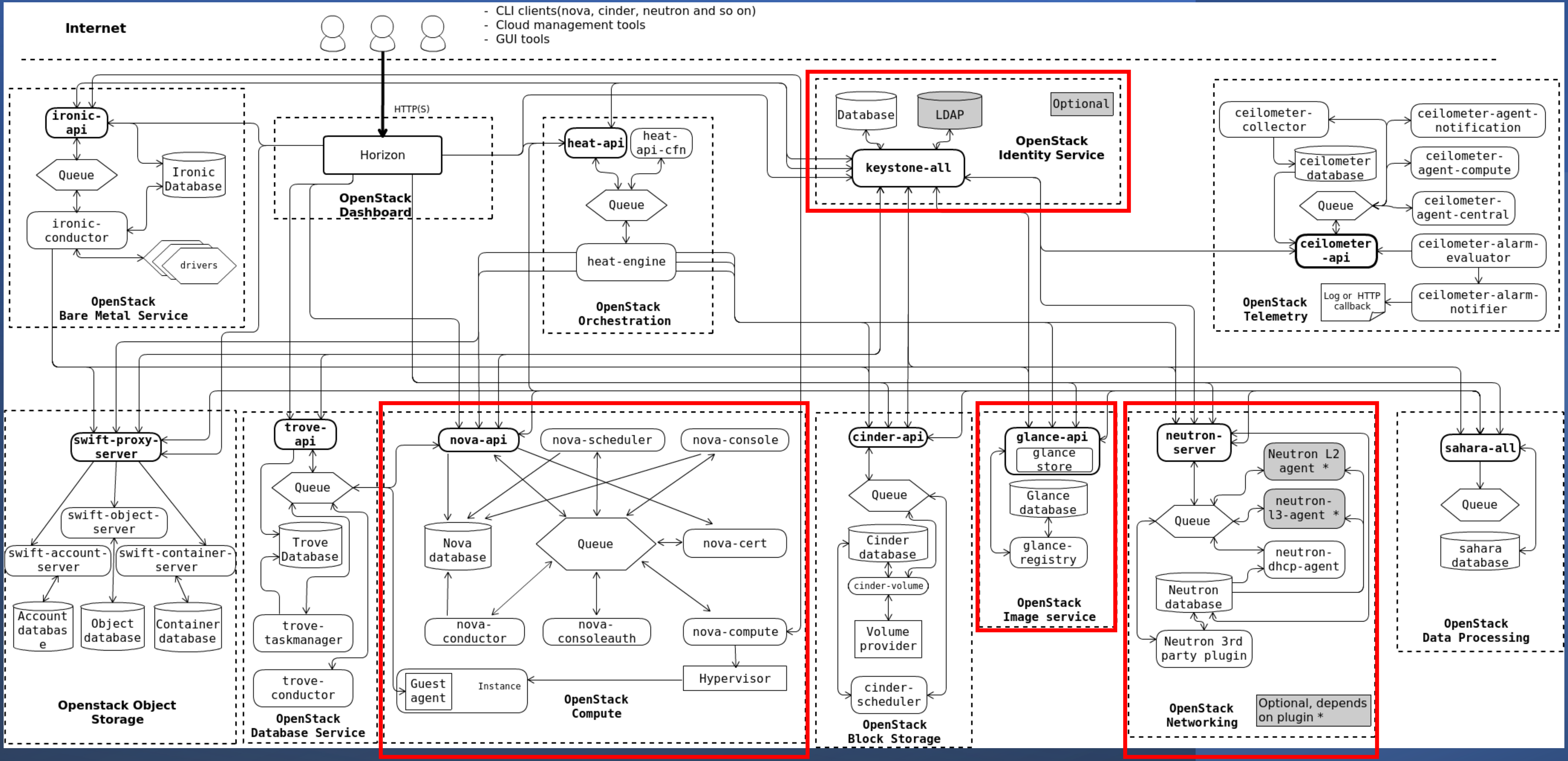
Some Thoughts

- Closed source VS open source
  - Bad code quality
  - No solid development workflow defined
  - Lack of instant & tangible advantages
  - Gray areas of open-source licenses

Meanwhile, on FreeBSD

- FreeBSD is only supported as a **guest OS** on OpenStack
- CHERI (Capability Hardware Enhanced RISC Instructions) project
  - Run OpenStack on FreeBSD machines to manage ARM boards

- The "OpenStack on FreeBSD" project
  - **Porting Linux-based OpenStack key components onto FreeBSD OS**
  - Started as a side project in Jan. 2022
  - Sponsored by the FreeBSD Foundation since Jul. 2022
  - Work in progress sharing at DevSummit 2023 in Tokyo



https://docs.openstack.org/install-guide/get-started-logical-architecture.html

Dev Environment

- Hardware
  - CPU: Intel Xeon E5-2680 v4*2
  - Motherboard: Supermicro X10DRL-i
  - RAM: 64 GB
  - Disk: 2 TB SSD
- Software
  - FreeBSD 13.1-RELEASE
  - OpenStack Xena
  - Python 3.8

Project Current Status

- Able to run key components on FreeBSD OS
  - Keystone
  - Glance
  - Placement API
  - Neutron
  - Nova
- Able to create instances (VMs) via OpenStack command line tool
  - Need to access compute node and connect the console with `cu(1)`
  - Need to set up static IP address for the VMs

Coming up

- VNC console integration (libvirt + noVNC)
- DHCP integration (jail + vnet)
- bhyve virtualization driver (libvirt)
- FreeBSD bridge plugin/agent (bridge + epair)
- Privilege separation adaptation (capability framework)
- Functional testing with tools like Rally
- Tidying up hackish code patches and converting to FreeBSD Ports

Some Thoughts (So Far)

- Working on open-source projects with a small group of people
  - Solid domain knowledge is crucial
  - Be systematic and methodical
  - Try to build the community
  - Grow with communities

How about Quit The Job?

- inlets – A cloud-native tunneling solution
  - Created by CNCF ambassador Alex Ellis

Some Observations

- Strategies – leverage on open source
  - Build personal brands
  - inlets itself is originally open-sourced, now turned into inlets-pro
  - Building an ecosystem – inletsctl, inlets-operator
  - Promotion – blog, Twitter, Reddit, Hacker News, LinkedIn, … etc.
  - Engage with your users
- Various types of source income
  - Product/personal sponsorship
  - E-books
  - coaching sessions
  - Consulting

➢You need to work very hard to make a living

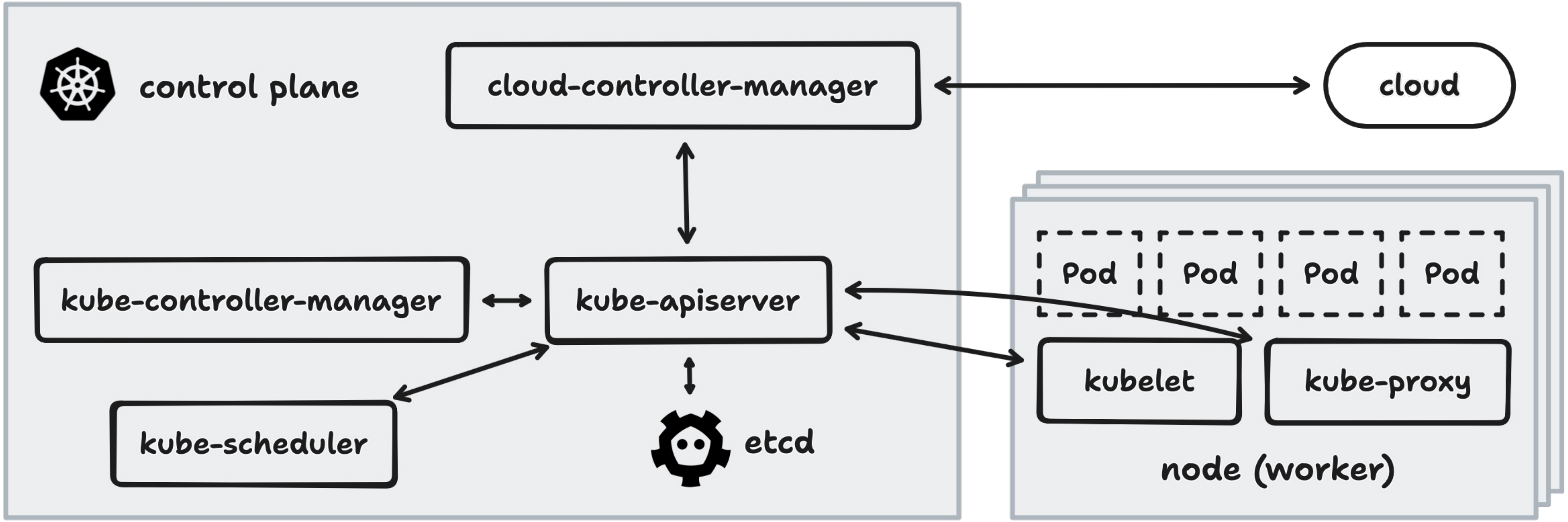
The World of Containers

- Building blocks of Linux containers
  - Visibility - Linux namespace
  - Isolation - Cgroups (Control Groups)
- LXC
  - System containers
  - Unprivileged containers
- Docker
  - Motto: build, ship, run
  - Filesystem
  - Images

Container Orchestration

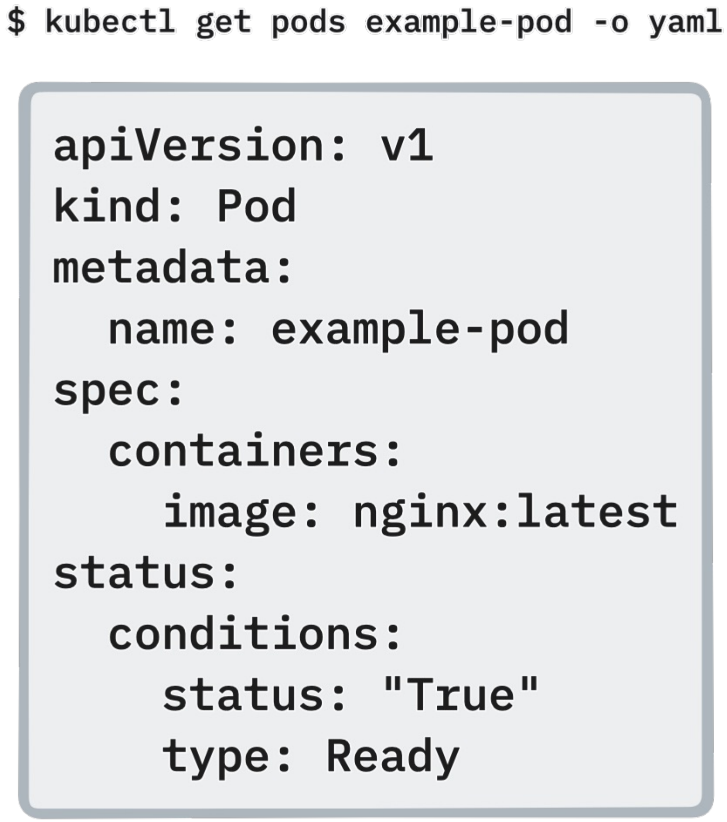
- So many containers…
  - Manageability (labeling system, health probes, …)
  - Autonomy (life-cycle, self-healing, …)
  - Orchestration (app deploy/upgrade strategies, affinity, …)
  - Observability (logs, metrics, …)
- Clustering solutions
  - Docker swarm
  - Nomad (by HashiCorp)
  - Kubernetes (formerly “Borg” from Google, donated to CNCF)

Bird's-eye View of Kubernetes



Core Concepts of K8s – API & KV Store (1)

- API schemas
  - Built-in resources
    - Node
    - Pod
    - Service
    - Deployment
    - Job
    - …
- Extending APIs
  - Custom resource definitions (CRDs): YAML only
  - API Aggregation: requires programming

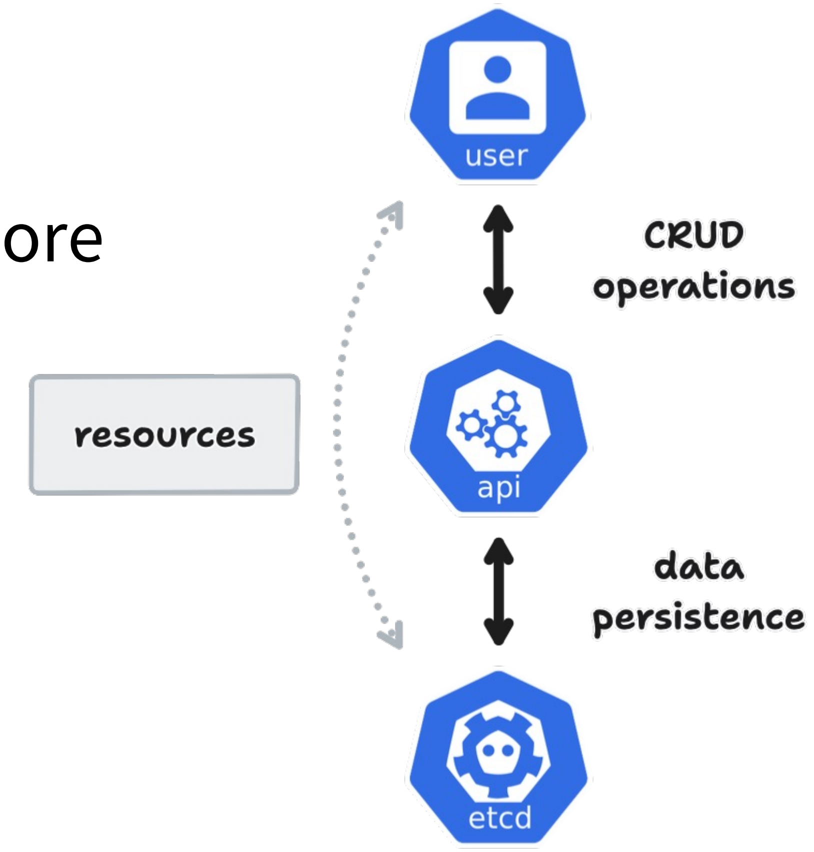


```
$ kubectl get pods example-pod -o yaml
```

```
apiVersion: v1
kind: Pod
metadata:
  name: example-pod
spec:
  containers:
    image: nginx:latest
status:
  conditions:
    status: "True"
    type: Ready
```

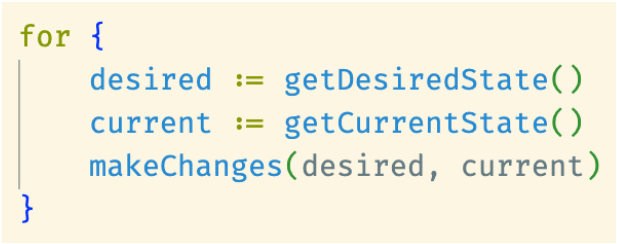
Core Concepts of K8s – API & KV Store (2)

- kube-apiserver: **Declarative** API server
  - Communicates to etcd
- etcd: Distributed, consistent key-value store
  - Raft consensus algorithm (**CA**P)
  - Act as backing database of kube-apiserver

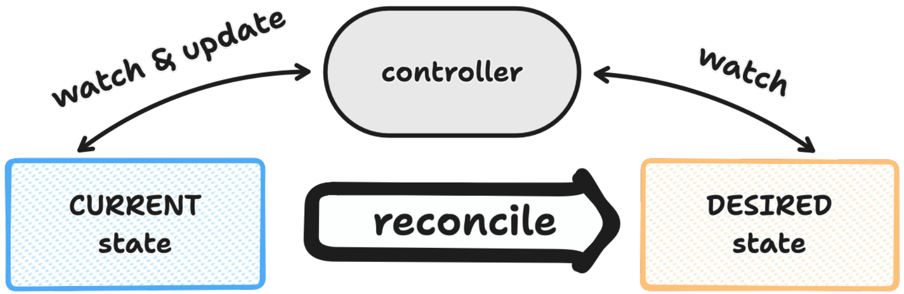


Core Concepts of K8s – Reconciliation (1)

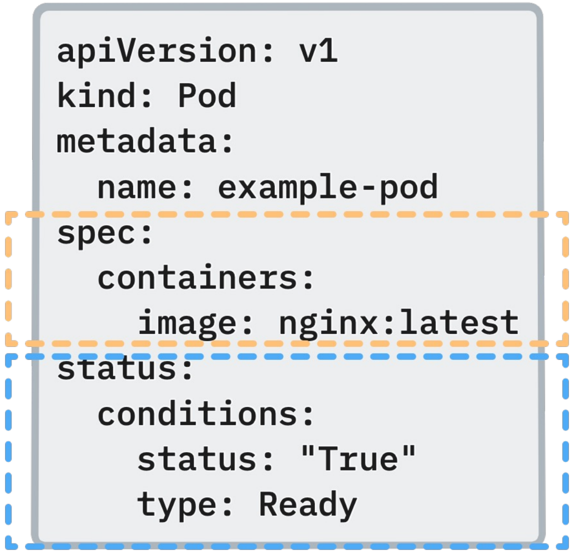
- Control loop (reconciliation)
  - A **non-terminating loop** that regulates the state of a system
  - Moving **current** state closer to **desired** state



```
for {
    desired := getDesiredState()
    current := getCurrentState()
    makeChanges(desired, current)
}
```



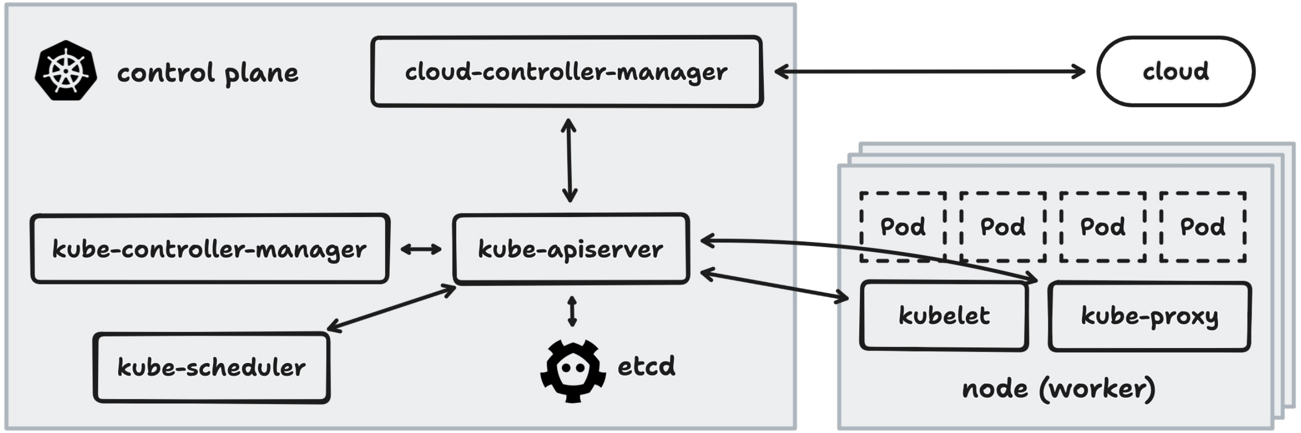
```
$ kubectl get pods example-pod -o yaml
```



```
apiVersion: v1
kind: Pod
metadata:
  name: example-pod
spec:
  containers:
    image: nginx:latest
status:
  conditions:
    status: "True"
    type: Ready
```

Core Concepts of K8s – Reconciliation (2)

- kube-controller-manager: A collection of built-in controllers
    - Service controller
    - Job controller
    - …
- Custom controllers
    - Custom resources
    - Aggregated API

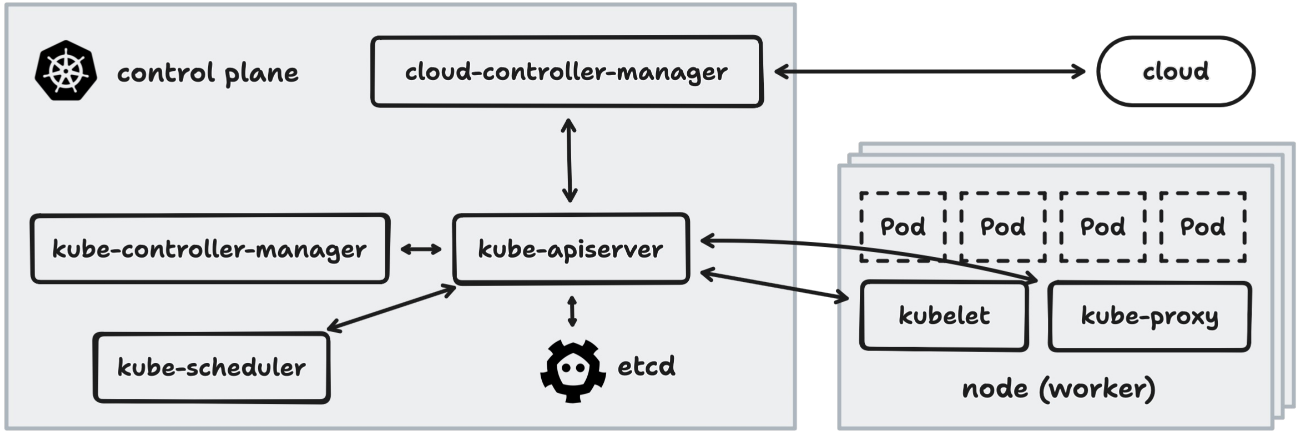


Core Concepts of K8s – Concurrency Control

- Race condition
- **Optimistic** concurrency control
  - resourceVersion

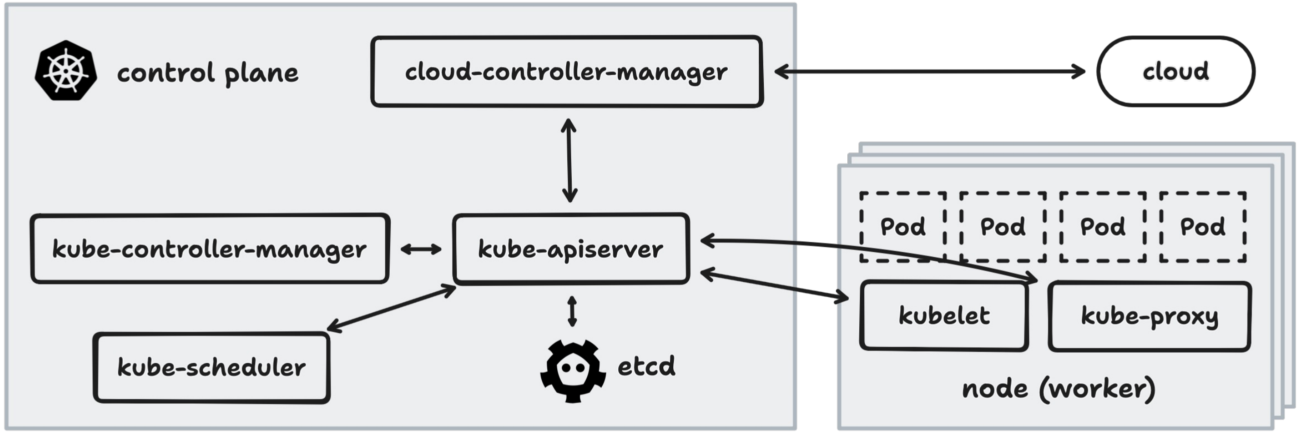
Core Concepts of K8s – Scheduling

- kube-scheduler
  - Watch for Pods
  - Assign Pods to Nodes according to constraints



Core Concepts of K8s – Runtime

- kubelet
  - Controller for Pod resources

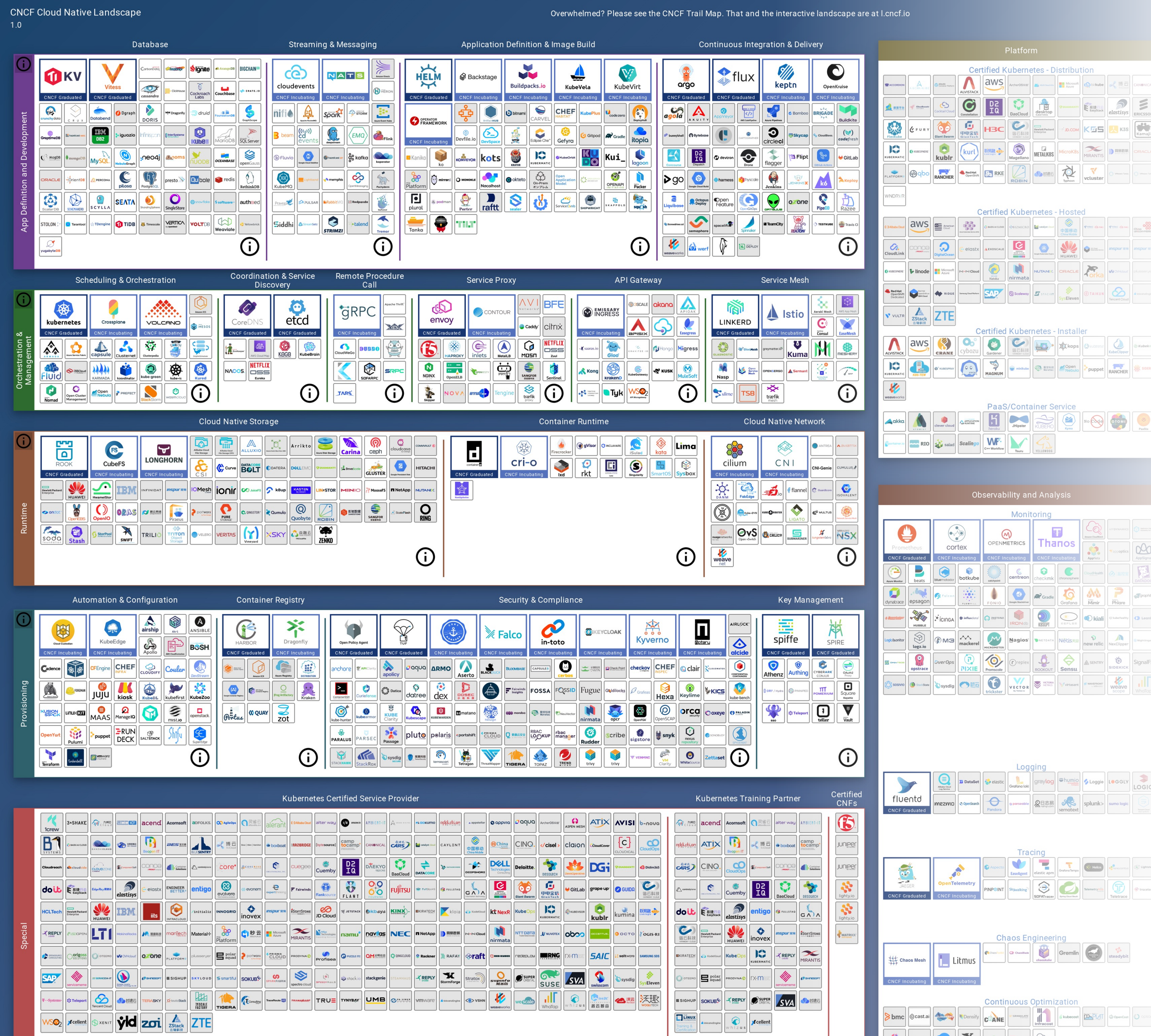


Some Observations

- We’re trying to move everything in the good old world to Kubernetes

Different Levels of Adoption

- Running applications on the cloud
  - Manifest files
  - Helm charts
- Writing operators for deployment of existing applications
  - Own business logic
  - inlets-operator
  - ECK (Elastic Cloud on Kubernetes)
- Cloud-native application/service
  - Longhorn
  - KubeVirt

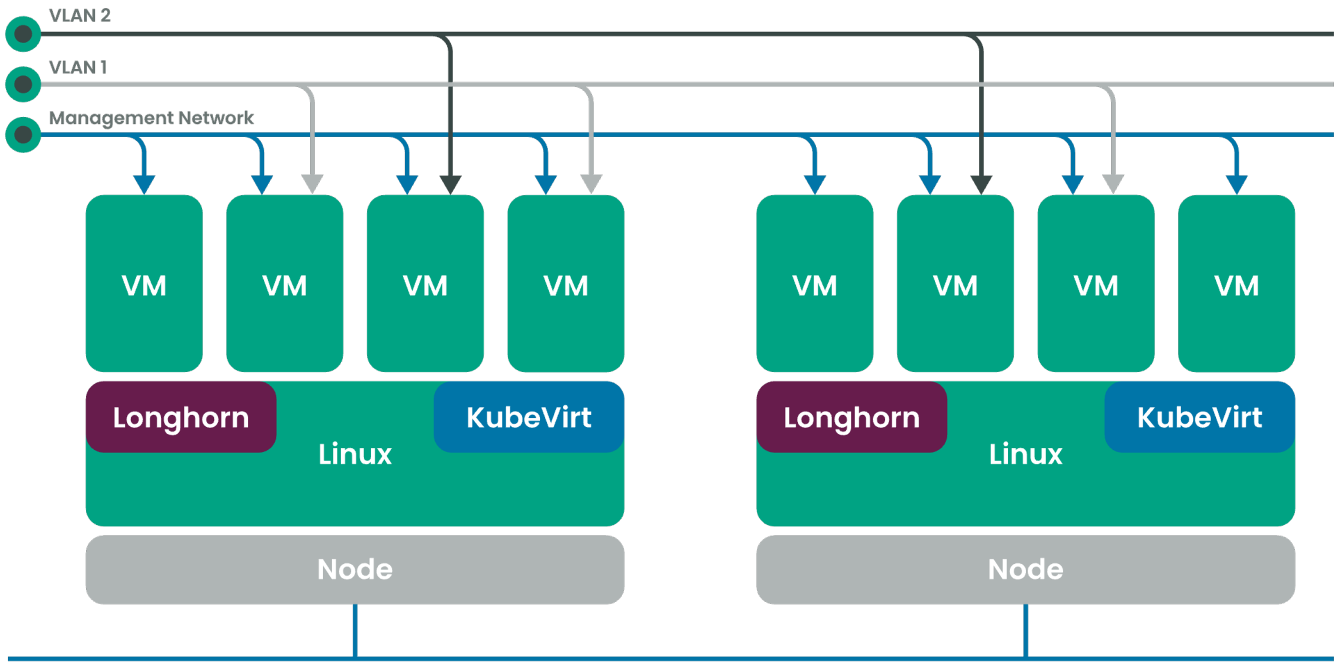

CNCF Cloud Native Landscape
1.0
Overwhelmed? Please see the CNCF Trail Map. That and the interactive landscape are at l.cncf.io
App Definition and Development
Database
Streaming & Messaging
Application Definition & Image Build
Continuous Integration & Delivery
Orchestration & Management
Scheduling & Orchestration
Coordination & Service Discovery
Remote Procedure Call
Service Proxy
API Gateway
Service Mesh
Runtime
Cloud Native Storage
Container Runtime
Cloud Native Network
Provisioning
Automation & Configuration
Container Registry
Security & Compliance
Key Management
Special
Kubernetes Certified Service Provider
Kubernetes Training Partner
Certified CNFs
Platform
Certified Kubernetes - Distribution
Certified Kubernetes - Hosted
Certified Kubernetes - Installer
PaaS/Container Service
Observability and Analysis
Monitoring
Logging
Tracing
Chaos Engineering
Continuous Optimization

CNCF Cloud Native Landscape

- https://landscape.cncf.io

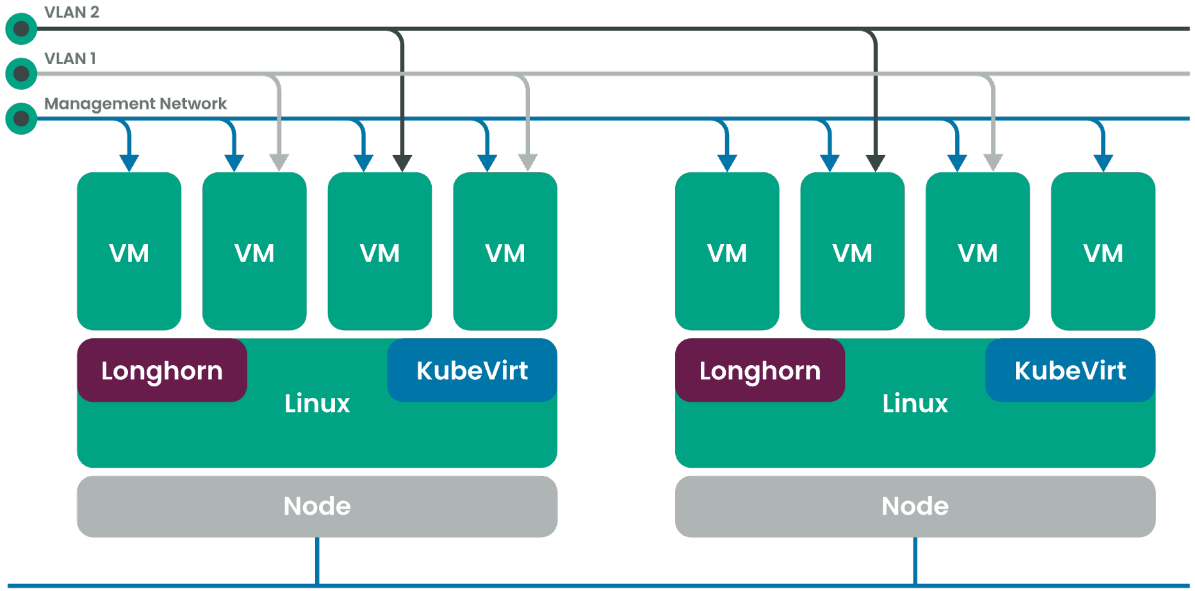
Introduce Harvester HCI

- An open source HCI solution https://github.com/harvester/harvester
- Building blocks
    - RKE2 (Rancher Kubernetes Engine)
    - KubeVirt
    - Longhorn
- Auxiliary services
    - Rancher
    - Prometheus
    - Seeder



Some Background about HCI

- Traditional DC/server farm deployment model
- Hyperconverged Infrastructure (HCI)
- Recent trends
  - Edge computing
  - dHCI (disaggregated HCI)

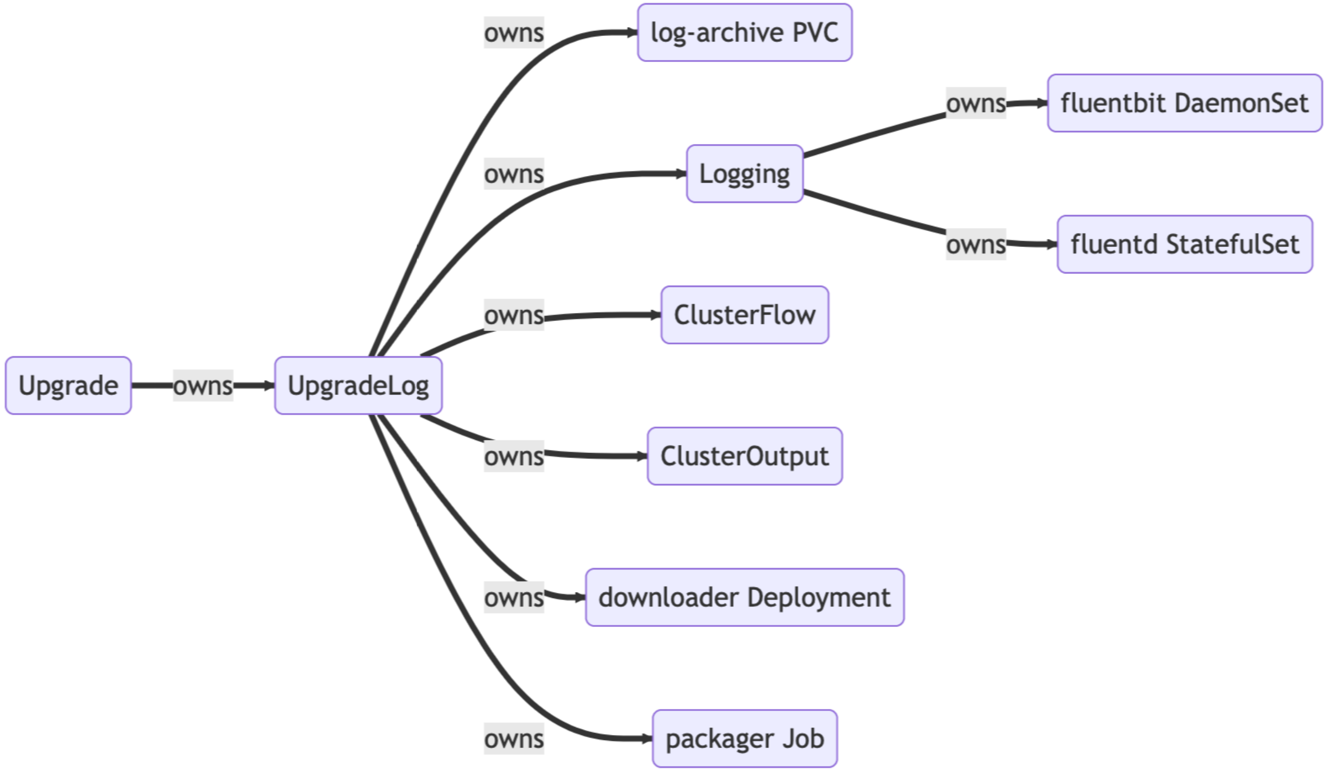


What’s Inside?

- Installer – harvester/harvester-installer
  - Golang program + lots of shell scripts
  - For installing Harvester
- Controllers – harvester/harvester
  - Golang program
  - Controllers for various CRDs
- More controllers under the Harvester organization

Writing A Controller

- Object handling
  - Retrieve from cache
  - Retrieve from API
- Control loop
- State machine



Some Thoughts (So Far)

- Working on open-source projects at a company
  - Well-defined rules for developing (open-sourced) software
  - Almost all your works are open to the public
  - Need to handle issues not just from paying customers but also community users

Building up Domain Knowledge

- Things you learned in school
  - Operating system
  - Computer networking
  - Virtualization
  - Filesystem
  - …
- Cloud computing
  - Bare-metals
  - Virtual machines
  - Containers

What are the Benefits of Working on Open-source Projects?

- ~~Make the world a better place~~
- Make things better by contributing to the upstream
- Building personal reputation and credits publicly

Write/Host Your Own Tech Blog

- Retrospection
- Sharing your thoughts
- Getting feedback
- Public records

- Don’t be afraid

Promotions

- FreeBSD Foundation
  - OpenStack on FreeBSD project
- OCF (Open Culture Foundation)
  - FreeBSD Taiwan Internship
  - https://blog.ocf.tw/2023/05/freebsd-intern.html
- Cambridge University & ARM
  - CHERI-related projects
  - lwshu

Thank You

References

- Open Source Guides https://opensource.guide