LF NETWORKING

LFN Developer & Testing Forum

Cloud Native OpenStack

OpenStack done the Kubernetes way...

About Me

@parthyadav3105

Ex-LFN Intern

Ex-Contributor @ Student Volunteer Project, OPNFV

Contributor at CIRV-SDV, Vineperf projects

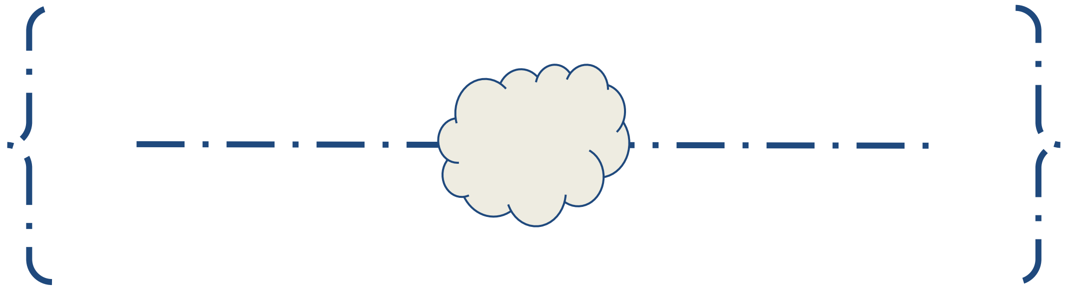
Research Associate @ University of Delhi

Interests: *Cloud-Native | NFVi | Hybrid-Cloud | Multi-Cloud | Edge | Container Networks*

Introduction

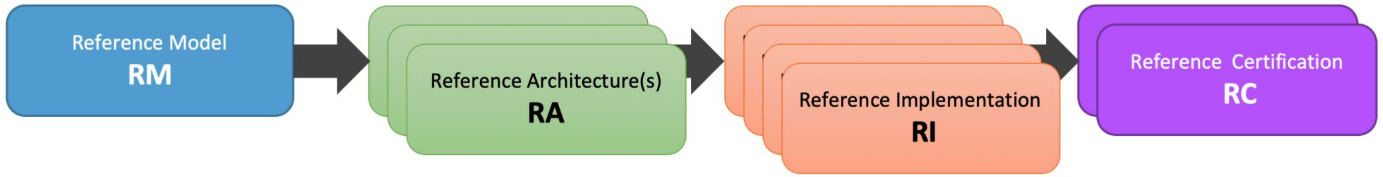
Let’s get to same page first!

Introduction



NFVi Implementations, Anuket

LFN Developer & Testing Forum



- RM principles require Cloud Infrastructure(NFVi) to be cloud-native
  - Both RA1(OpenStack), RA2(Kubernetes) must be cloud-native
    - req.gen.cnt.01 (RA1) [**stateless** design]
    - req.gen.cnt.02 (RA1, RA2) [**immutable**]
    - req.gen.cnt.03 (RA2) [conformant/certified]
    - req.gen.cnt.04 (RA2) [**abstraction**]
    - req.gen.cnt.05 (RA2) [**configurable**, **automated**, open-APIs]
- Managing Cloud Infrastructure is complex,
- Installers, Airship-RI approaches OOK(OpenStack on Kubernetes)
- Vanilla Kubernetes is not enough

Problem Statements

Currently, as a community, we all are trying to solve:

- Cloud-Native Infrastructure.
- Run CNFs and VNFs together.
- Multi/Hybrid cloud deployments.
- Edge to Data Centers.
- DevOps and Security Pipelines.

Without any lock-in stack...

Need for Cloud-Native OpenStack

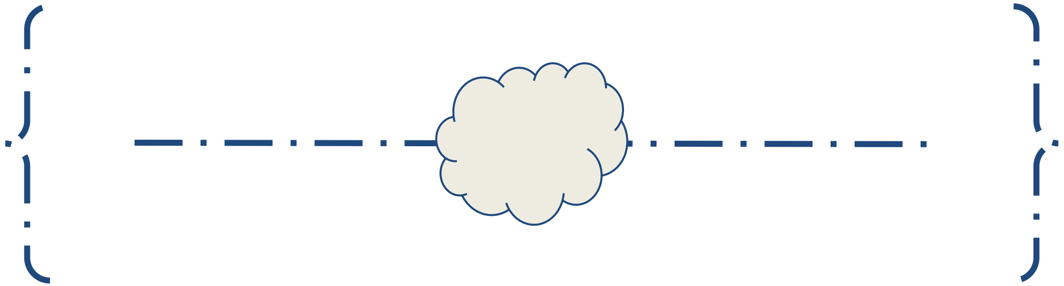
Needs for making OpenStack cloud-native?

- OpenStack is stable but complex to maintain, upgrade.
- Need for **abstractions**, **self-healing**, **zero-downtime**, etc.
- Small edge deployments, yet configurable and scalable.

Motivation:

- Hybrid Academic Cloud
  - Small, easy to maintain
  - Switch between public and on-prem
  - Self-managed

OpenStack Architecture

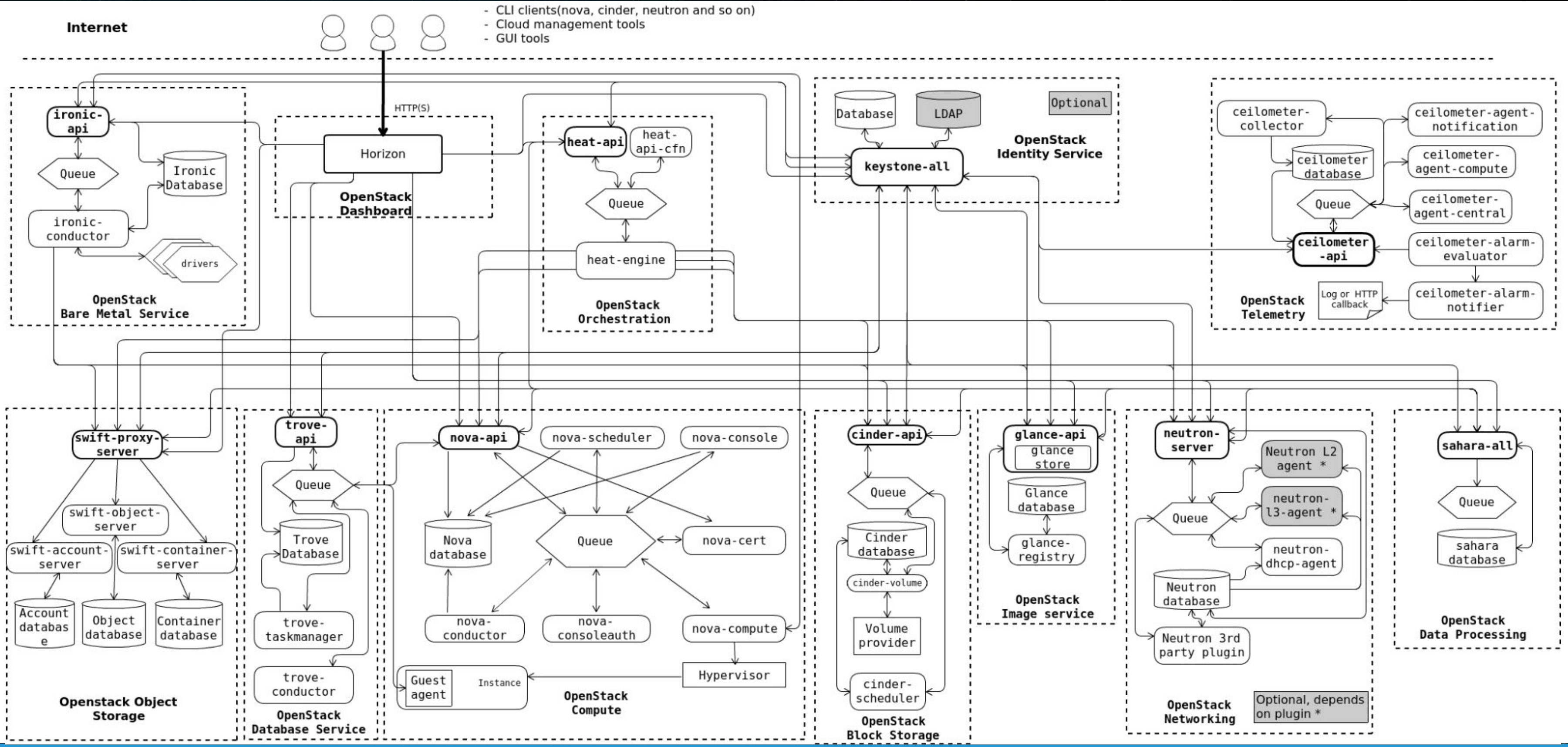


Key Observations

- loosely coupled, **distributed** architecture
- Many services
    - Nova(Compute)
    - Cinder(Block Storage)
    - Keystone(Identity)
    - ……..
- Every service internally has its own architecture.
    - Composed of several process.
    - Common design choices:
        - Public APIs for other services to integrate.
        - Communication b/w processes: AMQP message broker.
        - Database to store state.

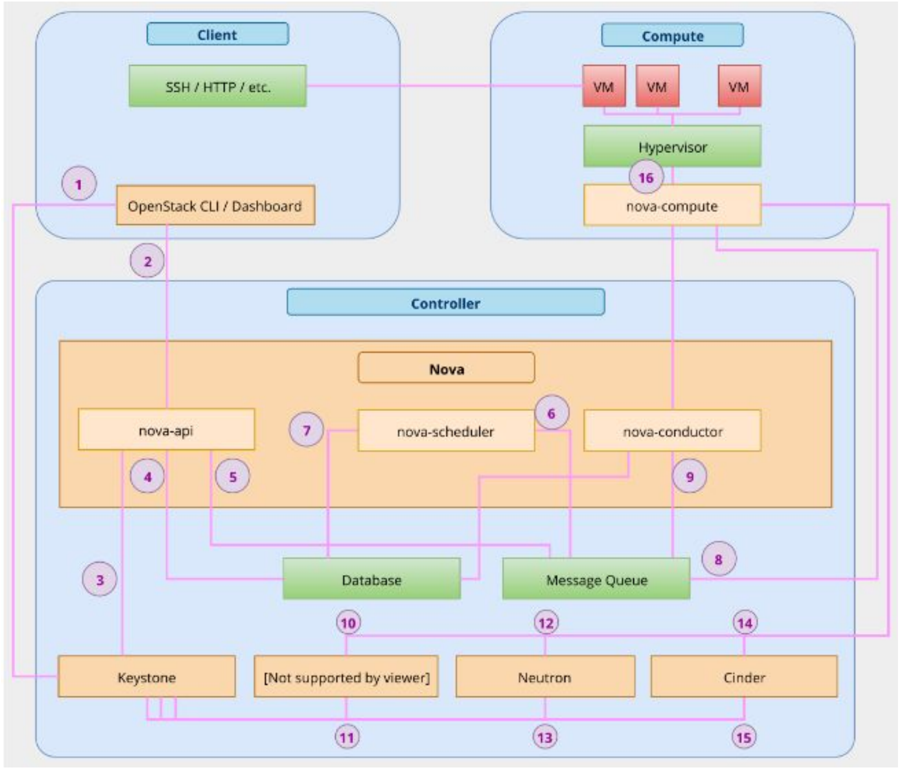
Example

LF NETWORKING
LFN Developer & Testing Forum



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

Example flow, Nova



Reference: 1. bit.ly/openstack-troubleshoot
2. https://www.linuxtechi.com/step-by-step-instance-creation-flow-in-openstack/

Key Observations

With OpenStack we get scenarios:

- Multiple nodes.
- Multiple software dependency stack.
- Multiple configurations

Which results into problems:

- **Complex to manage**
  - Multiple operating system, software versions,
    - Different troubleshooting for each
      - **Hard to Automate**
- After long term maintenance & troubleshooting
  - State, configurations of dependencies can differ from node-to-node even for same software stack.
    - Infrastructure is not immutable
      - Again, **Hard to maintain**

Solution

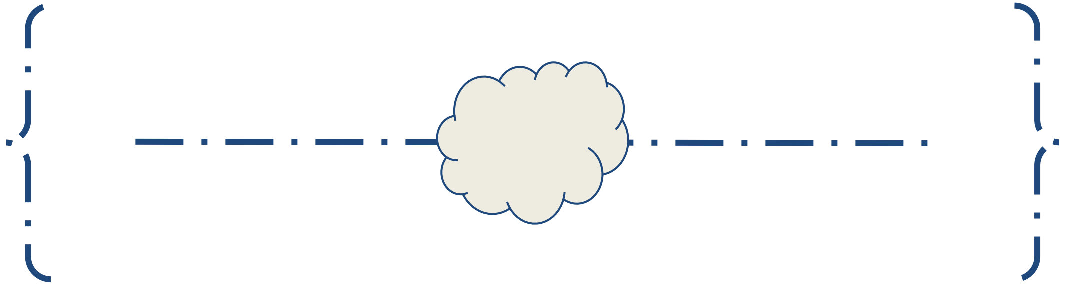
From these observations we understands:

- We need **immutable infrastructure** for OpenStack deployment and maintenance.
- We need **automation** for deploying OpenStack services.
- We need dependency management with proper **version** control.

We conclude that to making OpenStack cloud-native, we first need OpenStack deployments to have cloud-native values like **repeatable deployment, immutable infrastructure, automation, etc.**

As we know solution to this problem is already available: **Containerization**

Containerized OpenStack



Containerized OpenStack

With containerization we get high portability, repeatable, version-controlled deployments. Images follow Image Immutability Principle making it easy to automate.

At OpenStack community has been containerizing OpenStack for some time and we have multiple projects for container images:

- OpenStack-Ansible-LXC
- Kolla
- LOCI

Automation

With containerization, complexity decreases slightly

- We still have multiple nodes and multiple containers
  - Now **we need automation on Containerized OpenStack**

We also need life-cycle management operations on these deployments

We need configuration management for various OpenStack deployment.

We need easy-to-scale OpenStack deployments.

Some Deployment Studies

- OpenStack on OpenStack (OOO)
    - Example: TripleO
- OpenStack on Kubernetes (OOK)
    - Example: Airship

Key Observations:

- Undercloud-Overcloud pattern
- OOK deployment:
    - Kubernetes brings
        - Easy **scalability**, **LCM**, **resiliency**, **declarative** infrastructure, **immutable**

OpenStack-Helm

- OpenStack community project
  - For deployment of OpenStack on Kubernetes
  - OpenStack-Helm
  - Image agonist
  - LCM of OpenStack

Cloud-Native values from OOK deployments

With an OOK deployment and OpenStack-Helm, we build cloud-native paradigms in the process of deploying OpenStack like:

- **Declarative**, **immutable**, **scalable**, **repeatable**, **disposable**, **consistent**, **automated**, **resilient**, **versioned**, **packaged**, **configurable** deployment.
- We can built deep **observability** on this deployment.
- We can built secure **pipelines** for this deployment.

OOK deployment are not enough

- We have observed:
  - OpenStack is **distributed service** based by design.
  - OOK deployment builds other cloud-native values for installing OpenStack.

But **still, the operations done by OpenStack are not cloud-native**.

This is the next step to address, to take all previous learnings and build a Cloud-Native OpenStack.

What’s Next?

Cloud Native OpenStack

Considerations for making OpenStack Cloud Native

- A lot has been done and a lot has to be done.
  - OpenStack has 20M+ lines of code and ~60K commits in a year and over ~60 projects.
  - Widely used.
  - Very complex.
  - Solves many problem statements, is a complete IaaS.

Not possible to rewrite all this code to make it cloud-native.

But we need cloud-native operations.

How Kubernetes is Cloud-Native

Various design choices….

- Declarative API
- **Controller patterns**
- And many more…

Controller patterns:

- Operation for the platform are coded using **loops**
  - Loops reconcile **current state** to the **desired state**.
  - Desired state is obtained from a **highly available** key/value **store**.
  - This store is updated only through a **declarative api**.

Kubernetes allow extending itself using controllers and custom resources.

KupenStack

Kubernetes + OpenStack = KupenStack

Principles:

- Should not change anything in OpenStack *(i.e., Compatibility with any certified OpenStack)*
- Should not change anything in Kubernetes *(i.e., Compatibility with any certified Kubernetes)*

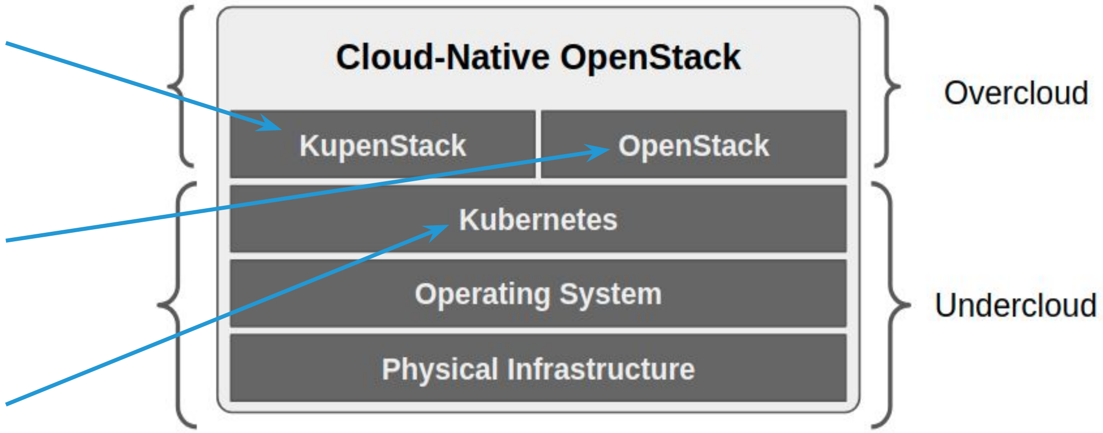
Cloud-Native OpenStack Architecture

LF NETWORKING
LFN Developer & Testing Forum

Kupenstack
A OOK controller with intelligence to build cloud-native operations on top of OpenStack. Operations like provisioning, scaling, self-healing, lcm, zero-downtime, upgrades of OpenStack infrastructure as well as resources(like VM, Subnet, Routers, etc.) and provide them as declarative APIs to its users.

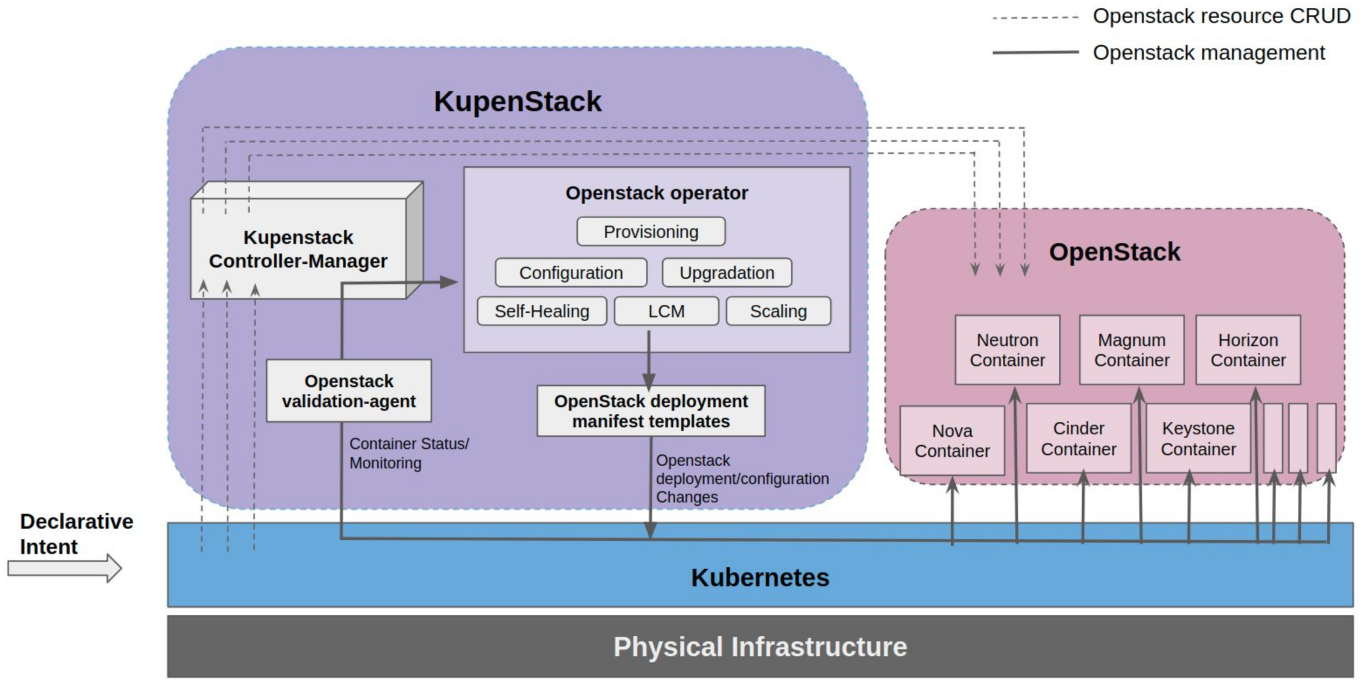
OpenStack
Containerized OpenStack(exactly same as before).

Kubernetes
Undercloud, k8s on bare metal, VM, public cloud, etc.(exactly same as before).



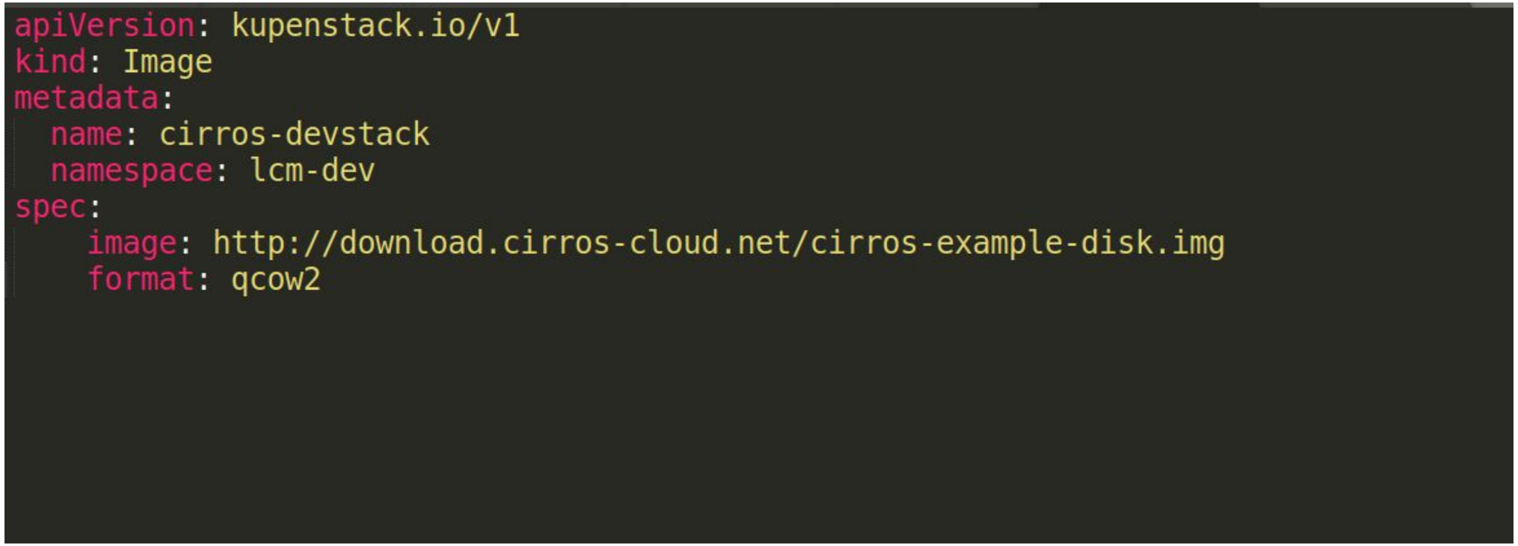
"KupenStack is the cloud-native layer between OpenStack and Kubernetes"

Logical Flow



OpenStack as-code

Example



```yaml
apiVersion: kupenstack.io/v1
kind: Image
metadata:
  name: cirros-devstack
  namespace: lcm-dev
spec:
    image: http://download.cirros-cloud.net/cirros-example-disk.img
    format: qcow2
```

Design choices

Important:

- Mapping OpenStack to Kubernetes
  - Namespace vs Projects
  - Authentication, Authorization(RBAC)
  - …..

- Future and integrations with other stack
  - Airship
  - Crossplane
  - KubeFed
  - Multi/Hybrid Cloud
  - ……

Checkout Paper: <link here>

Demo

LF NETWORKING
LFN Developer & Testing Forum

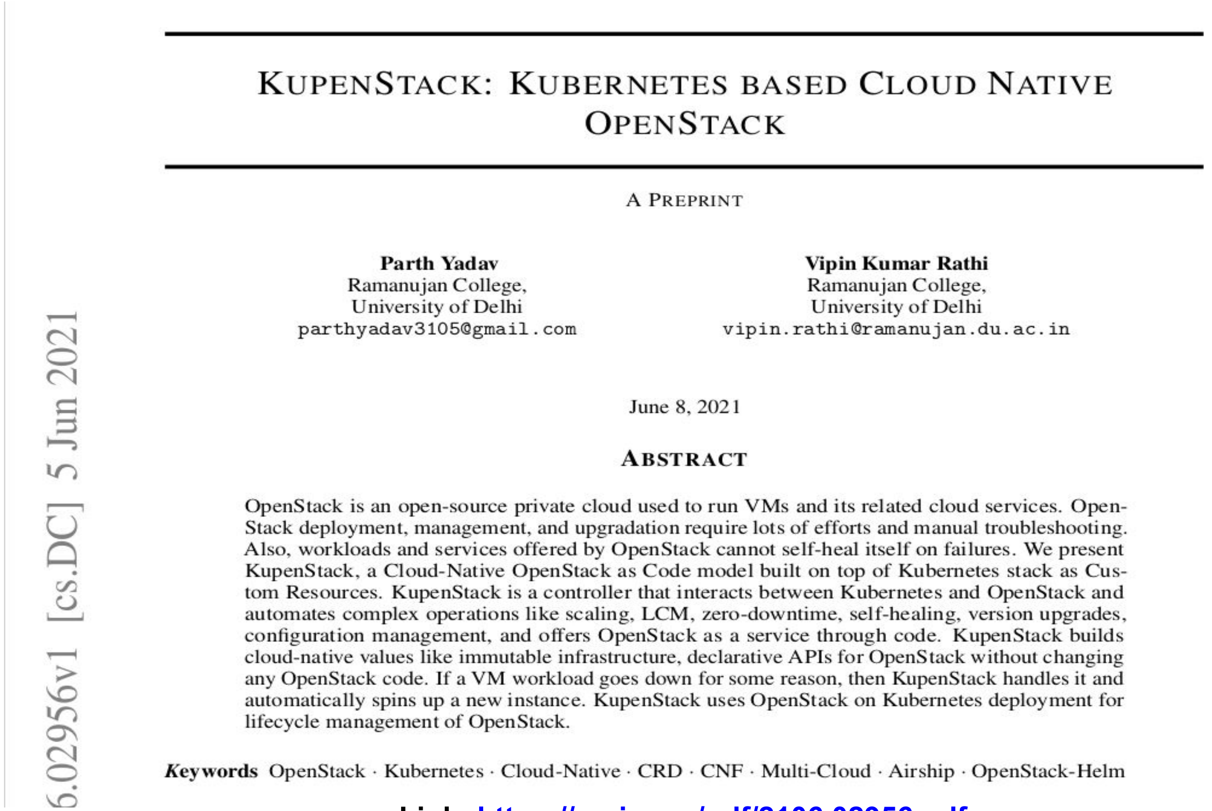
All code is available at: github.com/kupenstack

Conclusion

- KupenStack is the cloud-native layer b/w OpenStack and Kubernetes.
- Admins declare desired OpenStack infrastructure using Custom Resources.
- Users declare desired OpenStack resources using Custom Resources.
- KupenStack abstracts away all the complexities of OpenStack for end-users.
- Self-healing, zero-downtime, automated upgrades, scaling as Code for OpenStack.

- Use-cases:
  - Hybrid Academic Cloud.
  - Edge
  - Cloud Native NFVi for CNF, VNF, 5G
  - …..

Read More



KupenStack: Kubernetes based Cloud Native OpenStack

A Preprint

Parth Yadav
Ramanujan College,
University of Delhi
parthyadav3105@gmail.com

Vipin Kumar Rathi
Ramanujan College,
University of Delhi
vipin.rathi@ramanujan.du.ac.in

June 8, 2021

Abstract

OpenStack is an open-source private cloud used to run VMs and its related cloud services. OpenStack deployment, management, and upgradation require lots of efforts and manual troubleshooting. Also, workloads and services offered by OpenStack cannot self-heal itself on failures. We present KupenStack, a Cloud-Native OpenStack as Code model built on top of Kubernetes stack as Custom Resources. KupenStack is a controller that interacts between Kubernetes and OpenStack and automates complex operations like scaling, LCM, zero-downtime, self-healing, version upgrades, configuration management, and offers OpenStack as a service through code. KupenStack builds cloud-native values like immutable infrastructure, declarative APIs for OpenStack without changing any OpenStack code. If a VM workload goes down for some reason, then KupenStack handles it and automatically spins up a new instance. KupenStack uses OpenStack on Kubernetes deployment for lifecycle management of OpenStack.

Keywords OpenStack · Kubernetes · Cloud-Native · CRD · CNF · Multi-Cloud · Airship · OpenStack-Helm

Link: https://arxiv.org/pdf/2106.02956.pdf

Queries/Suggestions

Parth Yadav

Research Associate,

University of Delhi

parthyadav3105@gmail.com

Vipin Rathi

Assistant Professor,

University of Delhi

vipin.rathi@ramanujan.du.ac.in

Paper Link: https://arxiv.org/pdf/2106.02956.pdf

LF NETWORKING

LFN Developer & Testing Forum

Thank You