Canonical Charmed Kubernetes on Supermicro A+ systems Reference Architecture

March 2020

Written by:
Ravi Chintala, Supermicro
Andrey Grebennikov, Canonical

Super Micro Computer, Inc.
980 Rock Avenue
San Jose, CA 95131 USA
www.supermicro.com

CHAPTER 1 OVERVIEW

This document provides a complete reference architecture guide for Ubuntu Kubernetes solution on Supermicro hardware delivered by Canonical, including Supermicro A+ servers for workloads, storage, and Supermicro networking.

This guide discusses the Supermicro hardware specifications and the tools and services to set up both the hardware and software, including the foundation cluster and the Kubernetes cluster. It also covers other tools used for the monitoring and management of the cluster with an overview of how these components work. The guide also provides the deployment steps and references to configuration and automation scripts developed by Supermicro and Canonical for the deployment process. Finally, examples, along with validation of the deployed solution with expected results provided.

Executive summary

A Kubernetes cluster is now a common need by many organizations. Supermicro and Canonical have worked together to build a jointly engineered and validated architecture that details software, hardware, and integration points of all solution components. The architecture provides authoritative guidance and recommendations for:

- Hardware design
  - Infrastructure nodes
  - Cloud nodes
  - Storage nodes
- Network hardware and design
- Software layout
- System configurations

Supermicro A+ overview

Supermicro's latest range of H12 Generation A+ Systems and Building Block Solutions® optimized for the AMD EPYC™ 7002 series processors offer new levels of application-optimized performance per watt and dollar. They deliver outstanding core density, superior memory bandwidth, and unparalleled I/O capacity. All nodes in the rack are A+ 2U servers handling compute, control, and storage functions, as assigned by the Metal as a Service (MAAS) management node that is represented by A+ AS-1123US-TR4 1U server.

For more information regarding the A+ hardware, refer to the Supermicro hardware specifications section.

Kubernetes

This architecture guide is based on upstream Kubernetes release 1.16. Ubuntu Kubernetes solution always includes the current upstream version of Kubernetes that is evolving at a very

rapid pace, and the focus is to have an easily upgradeable solution to the next version once it released.

Core components

Component	Codename
Persistent Storage	Ceph RBD
Compute	Kubernetes Worker (Docker-based)
Networking	Flannel or Canal (Calico/Flannel)
Logging	Graylog
Monitoring	Prometheus

The standards-based APIs are the same between all Kubernetes deployments, and they enable customer and vendor ecosystems to operate across multiple clouds. The site-specific infrastructure combines open and proprietary software, Supermicro hardware, and operational processes to deliver cloud resources as a service.

The implementation choices for each cloud infrastructure are highly specific to the requirements of each site. Many of these choices can be standardized and automated using the tools in this reference architecture. Conforming to the best practices help reduce operational risk by leveraging the accumulated experience of Supermicro and Canonical.

Canonical's Metal as a Service (MAAS) is used as a bare metal and VM provisioning tool. The foundation cluster is composed of MAAS and other services (running in highly available (HA) mode) that used to deploy, manage and update the Kubernetes cluster nodes.

Kubernetes and Canonical

This reference architecture based on Canonical's Charmed Kubernetes. Canonical commercially distributes and supports the pure upstream version of Kubernetes. Ubuntu is the reference operating system for Kubernetes deployments, making it an easy way to build Kubernetes clusters. In Ubuntu, Kubernetes delivered in the form of snaps - the universal Linux app packaging format - which dramatically simplifies the installation and upgrades of components.

Canonical's Discoverer family of services provides the service to design, deploy, manage, and support customer clouds in POC, development, pre-production, and production environments. Canonical reference architectures delivered on a converged infrastructure approach, where any of the servers can accommodate more than one specific Kubernetes role or service simultaneously. This converged approach has many benefits, including simplicity of operation and

management overhead. Canonical can also deploy Kubernetes in a more traditional manner, grouping servers per role - controllers, storage, and container pods.

MAAS physical cloud

MAAS is complete automation for the datacenter of physical servers operation efficiency on-premises. It is open source and supported by Canonical. MAAS treats physical servers like virtual machines or instances in the cloud. Rather than having to manage each server individually, MAAS turns bare metal into an elastic cloud-like resource.

MAAS provides the management of a large number of physical machines by creating a single resource pool out of them. Participating machines can be provisioned automatically and then used as normal. When those machines are no longer required, they are "released" back into the pool. MAAS integrates all the tools needed in one smooth experience. It includes:

- Web UI, optimized for mobile devices
- Ubuntu, CentOS, Windows, RHEL and VMWare ESXi installation support open source IP Address Management (IPAM)
- Full API/CLI support
- High availability
- IPv6 support
- Inventory of components
- DHCP and DNS for other devices on the network
- DHCP relay integration
- VLAN and fabric support
- NTP for the entire infrastructure
- Hardware testing
- Composable hardware support

MAAS works with any system configuration, and recommended by the teams behind both Chef and Juju as a physical provisioning system.

Key MAAS Features

Feature	Description
Automation	Automatic discovery and registration of every device on the network. BMC (IPMI, AMT and more) and PXE (IPv4and IPv6) automation.
Fast deployment	Zero-touch deployment of Ubuntu, CentOS, Windows, RHEL, SUSE and ESXi. Deploys Linux distributions in less than 5 minutes.
Machine configuration	Configures the machine's network interfaces with bridges, VLANs, bonds and more. Creates advanced file system layouts

Feature	Description
	with RAID, bcache, LVM and more.
DevOps integration	Integration with DevOps automation tools like conjure-up, Juju, Chef, Puppet, SALT, Ansible and more.
Pod management	Turns bare-metal servers into hypervisors, allowing automated creation of virtual machines, and presents them as new servers available for the deployment.
Network management	Observes and catalogs every IP address on the network (IPAM). Built-in highly available DHCP (active-passive) and DNS (active-active).
Service tracking	Monitors and tracks critical services to ensure proper operations.
Manage	Comes with a REST API, Web UI and CLI.

Juju modeling tool

Juju is an open-source application modeling tool. It can deploy, configure, scale, and operate cloud infrastructures quickly and efficiently on public clouds such as AWS, GCE, and Azure, along with private clouds such as MAAS, OpenStack, and VMware VSphere.

The Juju store allows access to a wide range of best practice solutions, which can be deployed with a single command that can be used from the command line or through its powerful graphical representation of the model in the GUI.

Why use Juju?

Whether it involves deep learning, container orchestration, real-time big data, or stream processing, significant software needs operations to be open source and automated. Juju is the best way to encapsulate all the ops knowledge required to automate the behavior of the application.

Software versions

The following versions of software are part of this reference architecture:

Software versions

Component	Version
Ubuntu	18.04.3 LTS (kernel 4.15)
Kubernetes	1.16

MAAS	2.6
Juju	2.7.0
Kubernetes charms	latest

CHAPTER 2 HARDWARE SPECIFICATIONS

The base validated reference architecture solution is on the combination of Supermicro A+ servers. The reference architecture uses the following rack and server specifications.

Supermicro rack specifications

Supermicro rack specifications

Component type	Component description	Quantity
Rack	Standard data center rack	1
Chassis	Supermicro A+ AS-1123US-TR4 (Infrastructure node)	1
	Supermicro A+ AS-2123BT-HNR ***(Four hot-pluggable systems (nodes) in a 2U form factor.)***	1
Data switches (25G)	SSE-F3548S	2
Provisioning switch (10G)	SSE-X3348T	1

Server components firmware versions

NOTE: The versions listed below are the versions that were available at the time this Reference Architecture was developed. Ensure that the firmware on all servers, storage devices, and switches are up to date.

Firmware versions

Component	Version
BMC	01.46
BIOS	2.0
REDFISH	1.0.1

Supermicro A+ Servers Specifications

Description	Infra	Cloud
Base Server Model	AS-1123US-TR4	AS-2123BT-HNR (four hot-pluggable systems (nodes) in a 2U form factor.)
CPU	2 X PPSE-ROM7542-0075 (32C/64T 2.9G 128M)	2 X PPSE-ROM7542-0075 (32C/64T 2.9G 128M)
RAM	16 X MEM-DR416L-HL01-ER32 (256 GB)	16 X MEM-DR432L-SL02-ER32 (512GB)
2.5" OS SATA drive	2 X HDS-SUN0-MZQLB3T8HALS07 (3.8TB,NVMe PCIe3.0x4,2.5")	2 X HDS-SUN1-MZQLB1T9HAJR07 (1.92TB NVMe PCIe3x4, 2.5")
2.5" SSD or NVMe (if applicable)		4 x HDS-SUN1-MZQLB960HAJR07 (960GB NVMe PCIe3x4,2.5")
AOC (Network)	1 x AOC-MCX4121A-ACAT-MLN (25GbE dual-port SFP28, PCIe3.0 x8)	1 X AOC-MCX512A-ACAT (25GbE dual-port SFP28, PCIe3.0 x8)
RAID Card	1 X AOC-S3108L-H8iR	1 X AOC-S3108L-H8iR

Rack layout

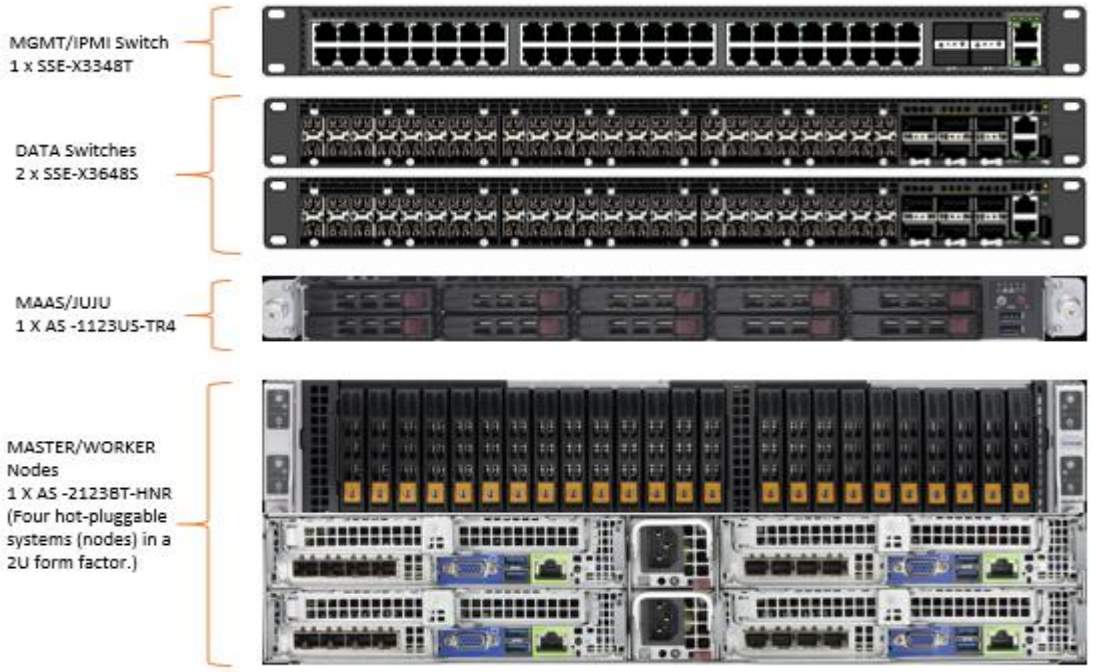
The reference deployment of Canonical Kubernetes on the Supermicro A+ servers utilizes the one node as an infrastructure node and four nodes for the Master/Worker. The reference deployment uses the following purpose:

Infrastructure nodes:

Node	Purpose
Rack1-MAAS1	Infra #1 (MAAS, LMA)

Cloud nodes:

Node	Purpose
Rack1-cloud1	Converged node handling Kubernetes components + Storage functions
Rack1-cloud2	
Rack1-cloud3	
Rack1-cloud4	



Hardware Configuration Notes

The Supermicro A+ configurations used with 10GbE networking require each node have two network cards, each offering 4 x 10GbE ports. The following configurations that need A+ server(s) for the Supermicro Charmed Kubernetes solution:

- BIOS
- IPMI
- RAID
- Network

Verify that the physical and virtual disks are in ready state, and that the virtual disks are auto-configured to RAID-0. In each A+ SR650 server, the IPMI over LAN option must be enabled through the BIOS.
For detailed hardware configurations of the A+ solution for the Charmed OpenStack platform, consult a Supermicro sales and services representative.
Caution: Please ensure that the firmware on hardware is up to date or match the versions from the table above.

CHAPTER 3 NETWORK ARCHITECTURE

A Supermicro A+ solution is agnostic to the top of rack (ToR) switch a customer may choose. This reference implementation uses the Supermicro A+ SSE-X3348T switch for the management network role. Also, two of the Supermicro SSE-F3548S switches used at the leaf-layer of the standard leaf-spine topology, to implement high availability on the data network. A pair of switches of similar or better capacity can replace at the spine-layer of the topology if desired.

Rack Data Switch SSE-F3548S 25GbE Switch

SSE-F3548S and its companion SSE-F3548SR Layer 3 Ethernet Switch both offer 25-Gigabit 48 Ethernet ports allowing data center friendly connectivity to 25GbE servers. These 48 ports can also run in 10/1Gigabit speed to connect to existing low speed network devices. SSE-F3548S/R also offers six ports running at 100Gbps for access to high-speed backbone networks or storage servers. The 100Gbps ports can also operate in 40Gbps speed or each can split into four different ports to run in 25/10Gbps speed.

25GbE Switch Specification

Variable	Description
SFP+ ports	48 x 25GbE SFP+ ports
QSFP+ ports	6 x 100GbE QSFP+ ports
RJ45 ports	1 x Console/Management Port
Operating System	Supermicro NOS

Refer to the Supermicro SSE-F3548S switch specification sheet for more information.

Rack Management Switch SSE-X3348T 10GbE Switch

SSE-X3348T and its companion SSE-X3348TR Layer 3 Ethernet Switch both offer 10-Gigabit 48 Ethernet ports using the popular new 10GBase-T connection option, allowing even more flexibility in providing data center-friendly connectivity to 10GE routers, servers, and backbones. They also offer four 40Gbps ports for access to high-speed backbone networks or storage servers

10GbE Switch Specification

Variable	Description
QSFP+ ports	6 x 100GbE QSFP+ ports
RJ45 ports	48 x 10 Gigabit Ethernet Ports
Operating System	Supermicro NOS

Refer to the SSE-X3348T switch specification sheet for more information.

Infrastructure layout

The network consists of the following major network infrastructure layouts:

- Data: The server NICs and the leaf switch pair. The leaf switches are, connected to the datacenter user networks and carry the primary service traffic in/out of the reference architecture.
- Management: The BMC management network, which consists of IPMI ports and the OOB management ports that are aggregated into a 1-rack unit (RU) SSE-X3348T switch. This 1-RU switch, in turn, can connect to the datacenter management network.
- MAAS Services: The MAAS Rack Controllers (see below) provide DHCP, IPMI, PXE, TFTP and other local services on the provisioning and IPMI network. Ensure that the MAAS DCHP server is isolated from the data center DHCP server.

Network components

The following component blocks make up this network:

- Server nodes
- Leaf switches and networks
- VLANs
- Out-of-Band Management switch and network

Server nodes

For maximum availability, the network must be resilient to the loss of a single network switch, network interface card (NIC), or bad cable. Achieving this requires the network configuration to use channel bonding across the servers and switches. Among several types (or modes) of channel bonding, only 802.3ad or LACP (mode=4) is recommended and supported for this solution. The endpoints for all nodes are terminated to switch ports that have been configured for LACP bonding mode, across two Supermicro SSE-F3548SR's configured with LAG across them. For details regarding network configuration on the servers, please contact your Supermicro services and sales representative.

Recommended channel bonding modes

Node type	Channel Bonding type
Infrastructure nodes	802.3ad (LACP mode 4, channel fast)

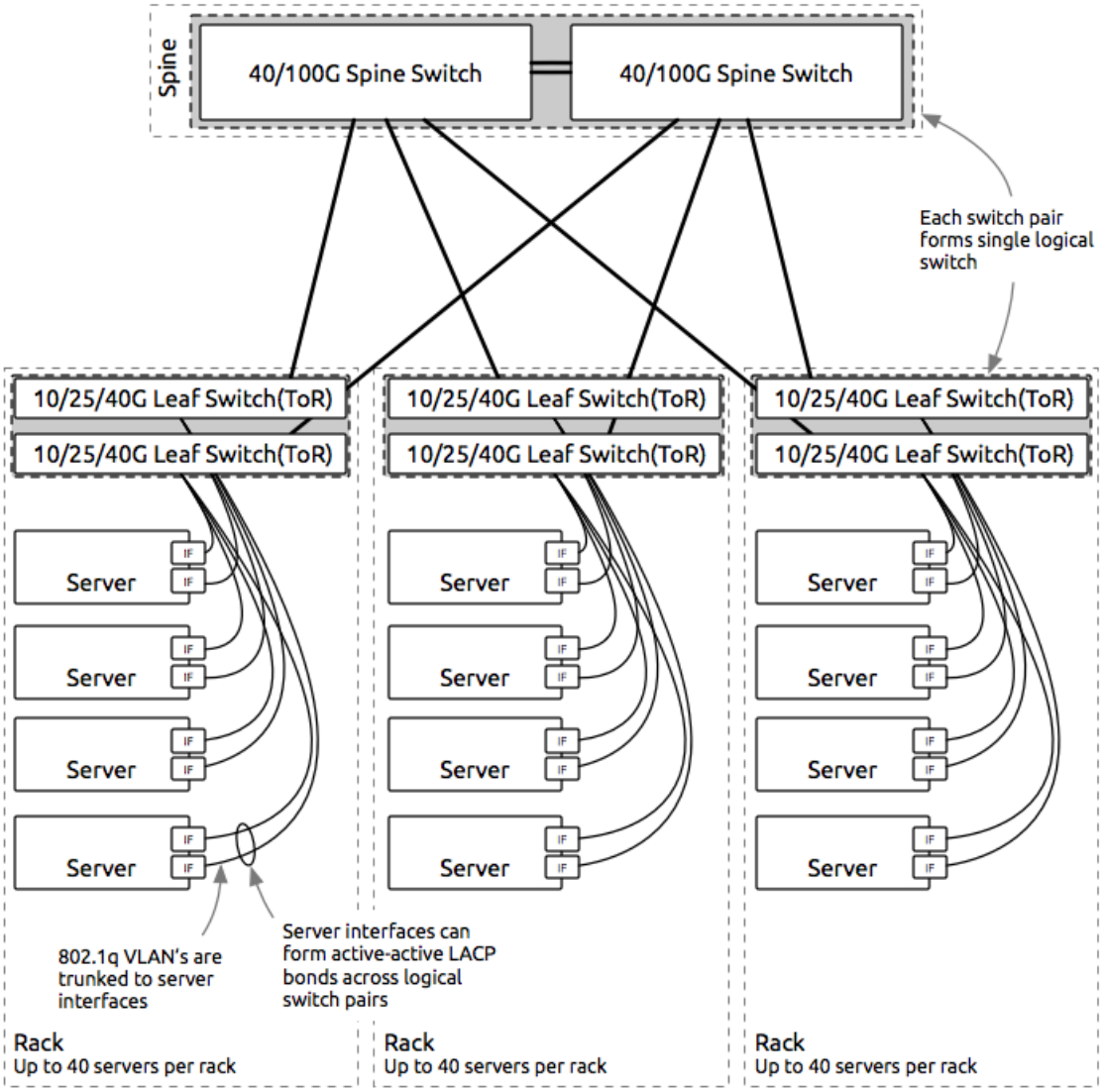
Cloud nodes	802.3ad (LACP mode 4, channel fast)

On the servers for separating critical types of traffic from each other multiple bonds can be created and allocating them on different physical interfaces. The actual layout depends on the particular cluster configuration and is out of scope of the Reference Architecture.

Leaf switches

This reference implementation uses two Supermicro SSE-F3548S switches. There is a redundant physical 2x 100GbE connection between the two switches. The recommended architecture uses LAG between the switches in the leaf pair.

Sample physical connections diagram, representing bonding setup of servers' interfaces and switches LAG setup:



VLANs

This reference architecture implements a minimum of four separate networks through Layer-2 VLANs. Multiple networks below can be combined into a single subnet based on end-user requirements.

VLAN	Description
OOB Management	Used for the BMC/IPMI network.
Internal	Used for cluster provisioning, monitoring and management

External	Used for communication between cluster components, as well as external access to the workloads, also for consuming persistent storage resources by the workloads.
Storage (cluster)	Used for replicating persistent storage data between units of Ceph.

Out-of-Band management network

The Management network of all the servers aggregated into the Supermicro SSE-X3348T switch in the reference architecture. One interface on the Out-of-Band (OOB) switch provides an uplink to a router/jumphost. The OOB management network used for several functions:

- The highly available software uses it to reboot and partition servers.
- When an uplink to a router is added, and the BMCs are configured to use it as a gateway, there are tools for monitoring the servers and gathering metrics.

A discussion of this topic is beyond the scope of this document—Contact Supermicro sales representative for additional information.

CHAPTER 4 CLUSTER INFRASTRUCTURE COMPONENTS

The infrastructure nodes are composed of the following services and tools:

- MAAS
- Juju
- Monitoring
- Log aggregation

This section provides details about how each of these components works.

How MAAS works

MAAS has a tiered architecture with a central Postgres database backing a region controller (regiond) that deals with operator requests. Distributed rack controllers, or (rackd), provide high-bandwidth services to multiple racks. The controller itself is stateless and horizontally scalable and only presents a REST API.

Rackd provides DHCP, IPMI, PXE, TFTP, and other local services. They cache large items like operating systems, install images at the rack level for performance, but maintain no exclusive state other than credentials to talk to the controller.

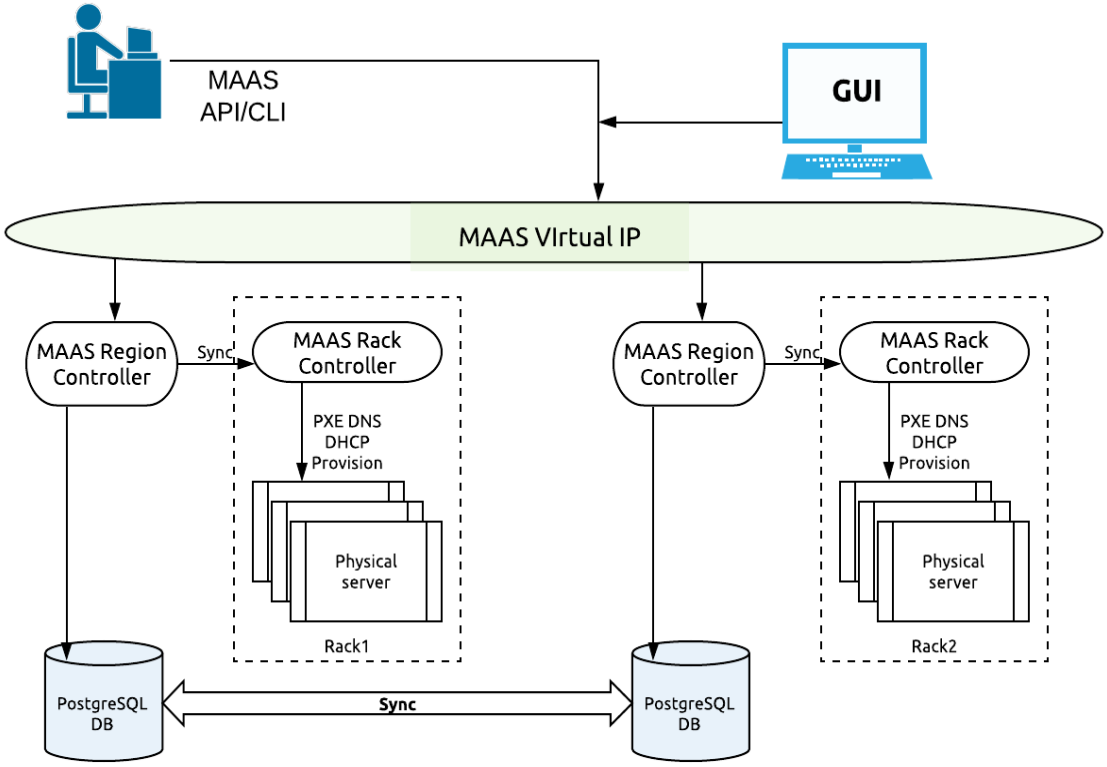
High availability in MAAS

MAAS is a mission-critical service that provides infrastructure coordination upon which HPC and cloud infrastructures depend. High availability in the region controller is achieved at the database level. The region controller will automatically switch gateways to ensure the high availability of services to network segments in the event of a rack failure.

MAAS can scale from a small set of servers to many racks of hardware in a datacenter. High-bandwidth activities (such as the initial operating system installation) are handled by the distributed gateways enabling massively parallel deployments.

The picture below represents the logical design of MAAS and the high availability of its components.

MAAS diagram



In the current setup, all components of MAAS are set up on a single server; however, for the production-grade cluster, it is highly recommended to follow the high-availability MAAS guide.

The node lifecycle

Each machine, or "node," managed by MAAS goes through a lifecycle — from new to enlistment, further to commissioned, and in the end to a ready state. There are also special statuses such as Broken and Testing.

NEW

New machines that PXE-boot on a MAAS network will be enlisted automatically if MAAS can detect their BMC parameters. During the enlistment phase, MAAS will ensure that it can control the power status of the machine through its BMC. Another option is to add devices through the API by supplying BMC credentials.

COMMISSIONING

In the Commissioning phase, MAAS collects all data about the machine, which includes detailed hardware inventory like CPU model, memory setup, disks, and chipsets. It also collects information about network connectivity. This information can be used later in deployments. In this phase, custom commissioning scripts can apply, which can update firmware, configure hardware RAID, etc.

READY

A machine that successfully commissioned is considered "Ready." A "Ready" machine has configured BMC credentials (on IPMI based BMCs) for ongoing power control. It ensures that MAAS can start or stop the machine and allocate or redeploy it with a fresh operating system.

ALLOCATED

Ready machines can be, Allocated to users, who can configure network interface bonding and addressing, and disks, such as LVM, RAID, bcache or partitioning.

DEPLOYING

Users can request that MAAS to turn the machine on and install a complete operating system from scratch without any manual intervention, configuring network interfaces, disk partitions, and more.

RELEASING

When a user has finished with the machine, they can release it back to the shared pool of capacity. It is possible to request MAAS to verify that there is a full disk-wipe of the machine when that happens.

Install MAAS

In the current setup, all components of MAAS are set up on a single server; however, for the production-grade cluster, it is highly recommended to follow the high-availability MAAS guide. For a detailed configuration procedure, please contact Canonical representatives.

Configuring Hardware

MAAS requires one small server and at least one server that can be managed with a BMC. Supermicro recommends configuring the MAAS server to provide DHCP and DNS on a network to which the managed machines are connected.

Install Ubuntu Server

Download Ubuntu Server 18.04 LTS, and follow the step-by-step installation instructions on the MAAS server.

MAAS Installation

This section describes the following MAAS installation topics:

- Prerequisites
- Infrastructure nodes requirements

Infrastructure nodes requirements

The infrastructure node pre-installed with the latest Ubuntu 18.04-LTS must be available to host multiple services intended to support building and operating the OpenStack solution, including:

- MAAS and its dependencies, including PostgreSQL
- Juju controller
- Monitoring and alerting systems
- Log aggregation and analysis systems

The node must have SSH access to the **ubuntu** user enabled through authorized_keys.

The Infrastructure node hosts these services either on the bare metal or in KVM virtual machines. Infrastructure node must have network access to:

- The PXE and BMC networks to commission and provision machines.
- The various APIs which must be monitored to monitor Kubernetes, the nodes must have access to the Kubernetes API.
- Internet, to the Ubuntu archives and other online services, to obtain images, packages, and other reference data.

To provide HA, infrastructure nodes must:

- Be placed in separate hardware availability zones
- MAAS has a concept zone where server hardware can be placed into different racks, and each rack can be placed in a single zone. Or within the same rack hardware can be divided based on the power redundancy or the slots within the rack. It would be helpful to place different services in different hardware zones.
- Have bonded network interfaces to provide resiliency from the switch or NIC failures.
- Have the MTU on the bonded interfaces set to 9000B (jumbo frames).
- Have a bridge (broam) interface active, which has the primary bond (typically `bond0`) as its only member. The bridge inherits the MTU of the underlying device, so there is no need to set its MTU explicitly.

MAAS initial configurations

This section describes the following MAAS initial configurations:

- MAAS credentials
- Enlist and commission servers

MAAS Credentials

For the initial installation of MAAS, follow the official procedure.

All-Region controllers should point to the Virtual IP of PostgreSQL database (or to the localhost address in non-HA mode). More info on MAAS HA configuration can be found in MAAS documentation

After the package installation it is required to create a set of credentials for admin users with **"maas init"** command.

Enlist and commission servers

Now MAAS is ready to enlist and commission machines. To perform that task:

1. Set all the servers to PXE boot from the first 10Gbe network interface.
2. Boot each machine once. The machines should appear in MAAS.
3. Select all of the machines and commission them by clicking on the Take action button.
4. When machines have a Ready status, the services can be deployed.

Set up MAAS KVM pods

Once MAAS is completely set up, all infrastructure nodes should be turned into KVM hosts managed by MAAS. Once done, MAAS will be able to dynamically provision Virtual Machines on the nodes and present them as available servers to the users. Follow the guide for turning Infrastructure nodes into KVM Pods and creating a set of VMs for setting up Juju controllers and LMA components.

Juju components

For an overview of Juju, refer to the Juju modeling tool. This section discusses the working of different components of Juju.

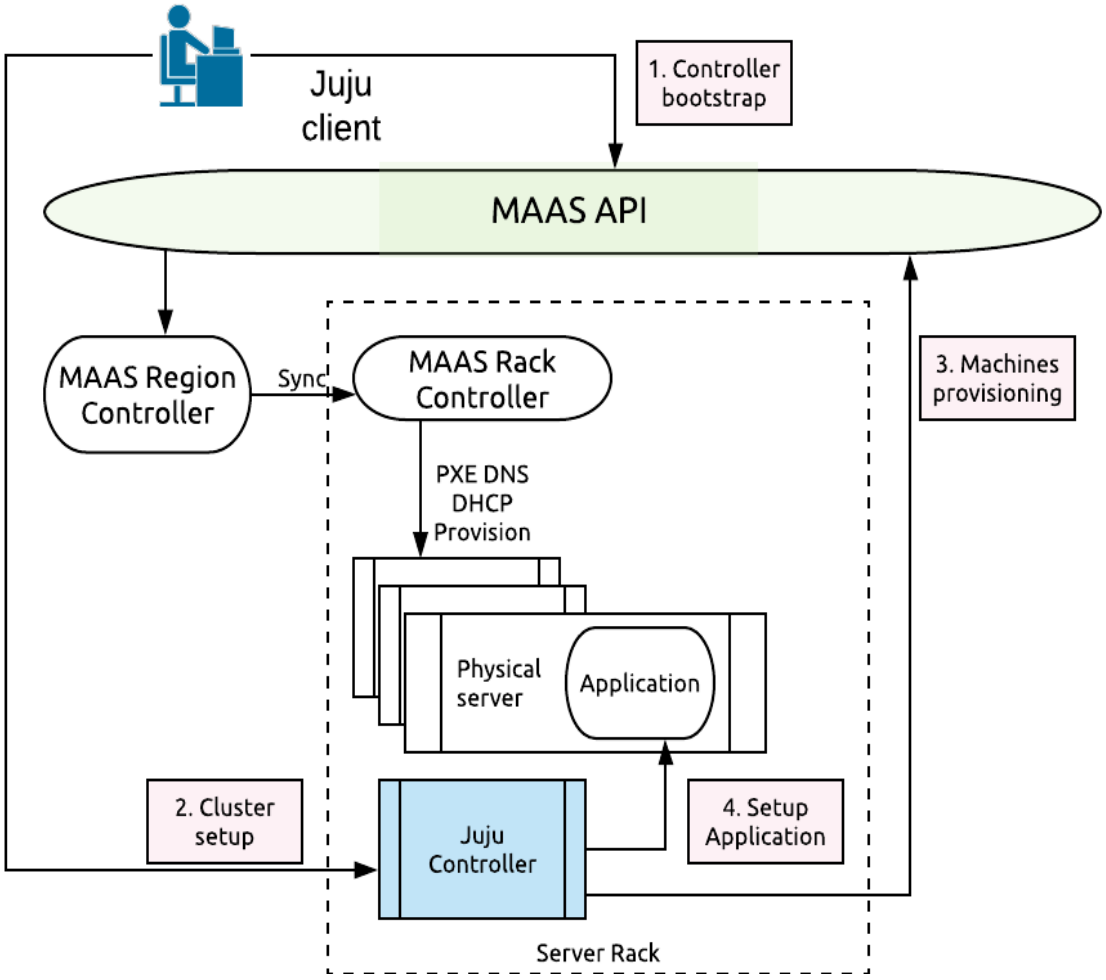
Juju controller - the heart of Juju

The Juju controller manages all the machines in the running models and responds to the events that are triggered throughout the system. It also manages scale-out, configuration, and placement of all models and applications.

Juju controller has to be located in the same physical segment of the network as the Kubernetes cluster and be able to execute calls to MAAS API and connect to Kubernetes Cluster nodes.

Juju controller is supposed to create using a set of KVM Virtual machines mentioned in the previous steps.

Below is the diagram of the standard workflow of bootstrapping the Juju controller in MAAS, provisioning the machines, and setting up the application with Juju.



Charms

Charms are a collection of scripts that contain all of the operations necessary to deploy, configure, scale, and maintain cloud applications with Juju. Charms encapsulate a single application and all the code and know-how that it takes to operate it, such as how to combine and work with other related applications, or how to upgrade it.

Charms also allow a hierarchy, with subordinate charms, to complement a main service.

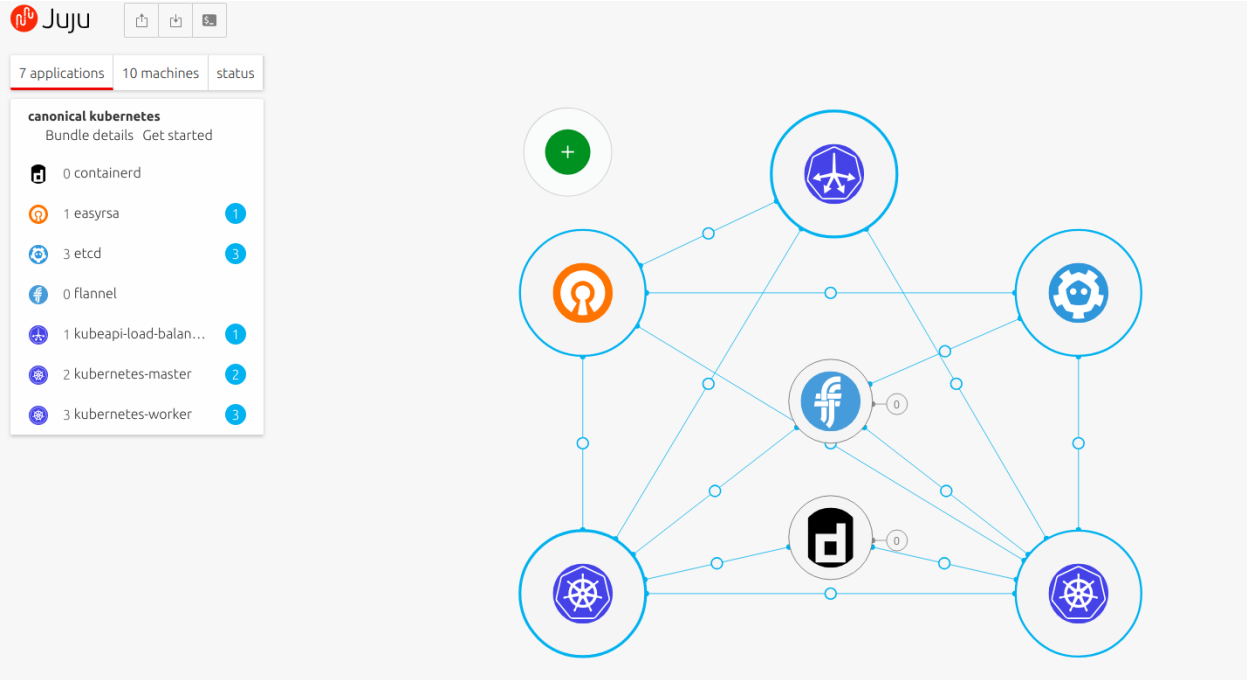
Bundles

Bundles are ready-to-run collections of applications that are modeled to work together and can include particular configurations and relations between the software to be deployed.
Bundles may also be optimized for different deployment scenarios of the same software—for example, a scale-out, production-ready version like the Canonical Distribution of Kubernetes. Also, consider a development-friendly test version like Kubernetes Core.

Bundles perform the following functions:

- Install
- Configure
- Connect
- Upgrade and update
- Scale-out and scale-back
- Perform health checks
- Undertake operational actions
- Benchmark

Juju supports UI representation of deployed bundles and allows it dynamically manipulate the cluster's configuration options and layout before the bundle deployment and during the lifetime.



PROVISION

Specify the desired number of machines and the constraints, or let Juju do it automatically.

DEPLOY

Deploy the services, or (re)deploy the entire application infrastructure to another cloud, with a few clicks of the mouse.

MONITOR AND MANAGE

The Juju controller manages:

- Multiple models
- All VMs in all running models
- Scale-out, configure and placement
- User accounts and identification
- Sharing and access

Comparing Juju to any configuration management tool

Juju provides a higher level of abstraction and supplies the tools needed to manage the full scope of operations beyond deployment and configuration management, regardless of the machine on which it runs. One of the main advantages of Juju is the dynamic configuration ability, which enables:

- Reconfigure services on the fly.
- Add, remove, or change relationships between services.
- Scale in or out with ease, sharing the operational knowledge and making the most of the wider community.

Monitoring

Ubuntu Kubernetes solution includes a monitoring suite based on the best open-source tools available.

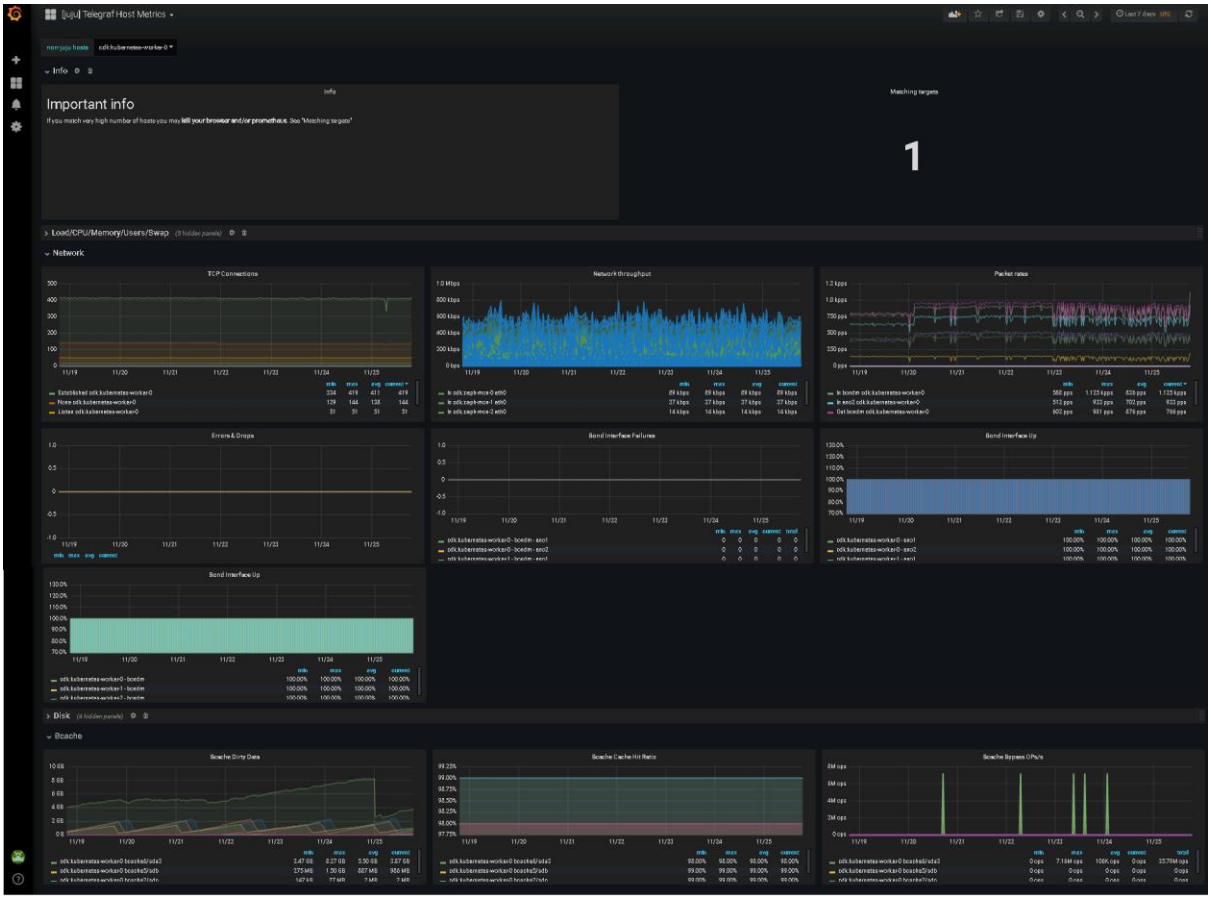
Observability Tools

The Canonical monitoring suite retrieves information from the Kubernetes components and infrastructure monitors, and combines it in a configurable portal, giving the customer visibility to all the different metrics.

The portal aggregates the relevant information from an operational perspective and differentiates various components, such as compute, network, and storage.

The Canonical observability tool allows both customers and operators to zoom in on the details of any of the higher-level graphs to obtain further information. The portal also includes an efficient time-series database that allows tracking of the evolution of the cloud metrics and health status over time.

Below is a display of the monitoring tool dashboard.



Log Aggregation

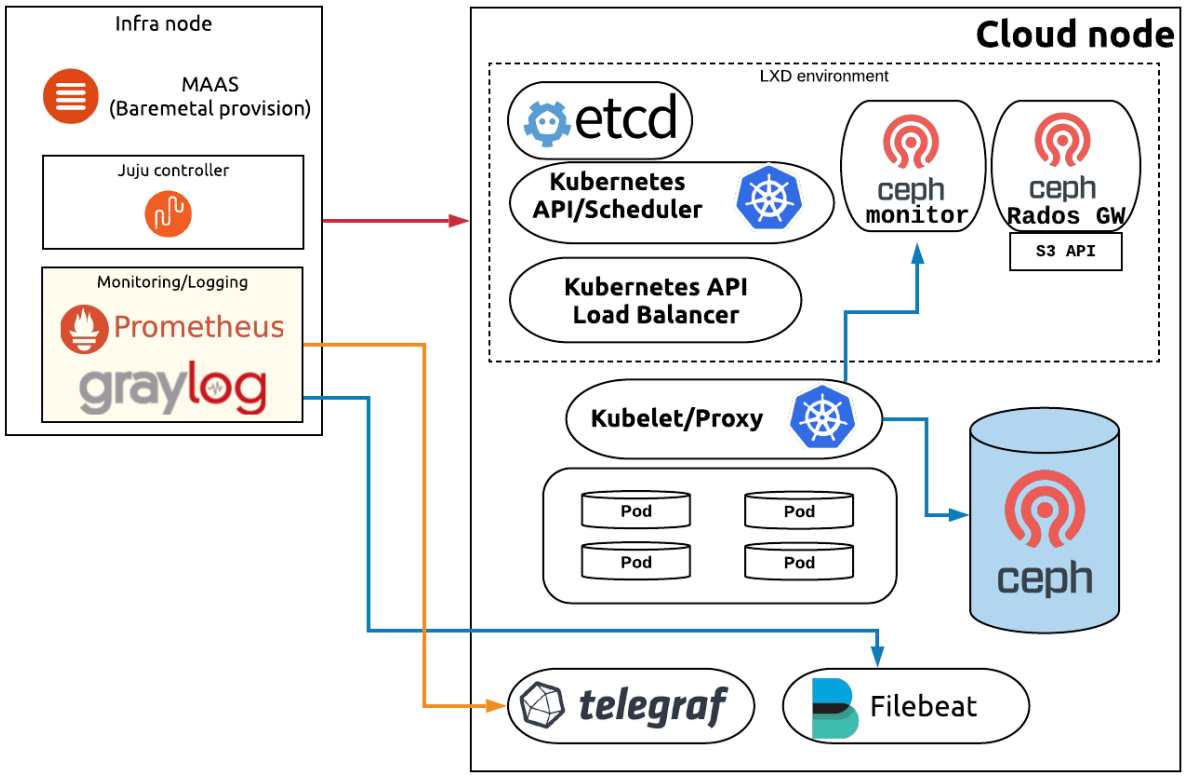
The solution also implements the Graylog suite for log aggregation, which makes it easy for customers to have visibility on the different logs from their cloud services without accessing them directly.

These services integrated with the Ubuntu Kubernetes solution as part of the charms, fulfilling the same requirements around upgradeability and operation.

CHAPTER 5 CHARMED KUBERNETES COMPONENTS

This chapter presents detailed information about the Kubernetes components included as charms in Charmed Kubernetes.

Below is the picture of all components that are parts of the cluster, including Kubernetes platform components, Ceph, and Logging/Monitoring solution.



Even though Juju allows flexible placement of the applications, it is considered to be optimal to run Control plane components in the LXD containers. This is because it provides further flexibility of their co-location, upgrades, and scaling, while Kubernetes worker (Kubelet) and Ceph OSD daemons are running on the level of the node Host OS. This allows direct access to the host's drives for data placement and direct access to GPUs in case they are present.

Logging and Monitoring components are running inside the Virtual Machines on top of the infrastructure nodes.

Storage charms

Ceph is a distributed storage and network file system designed to provide excellent performance, reliability, and scalability. Canonical uses Ceph by default for storage; however, this can be replaced by, or complemented with, another storage solution. The Ceph cluster provides the capacity for the container's persistent volumes.

Ceph-monitor

The charm deploys the core components of the Ceph cluster called Monitor. The set of monitors stores the information about the current cluster layout, placement groups, pools, etc.

Ceph-osd

This charm provides the Ceph OSD functionality for expanding storage capacity within a Ceph deployment.

Ceph-radoswg

RADOS Gateway is the component of the Ceph ecosystem for providing Swift and S3 API that allows the application to store and consume objects from the remote store (which is backed by the Ceph cluster).

Kubernetes charms

EASYRSA

EasyRSA is the PKI used by Charmed Kubernetes to build and deploy x509 certificates used to secure the communication between the various layers of Kubernetes: ETCD cluster, Kubernetes API, and others.

A simple CLI Tool that Juju leverages via SSH implemented over the charm relation protocol. Charms consuming certificates can query the relation to get access to the CA certificate or ask for the creation of separate server or client certificates.

KUBERNETES-MASTER

The Kubernetes Master is in charge of managing the Kubernetes cluster. It is made of three underlying components:

- The API Server is in charge of being the interface between the administrators and users of the cluster and the cluster itself, but also provides services within the cluster
- The Scheduler is in charge of allocating pods to their nodes
- The Controller Manager is in charge of maintaining the cluster in a desired state

All of these applications are stateless and can scale in and out very easily. The state of the cluster saved in the etcd database.

Kubernetes-worker

The Worker is the component that runs the compute tasks in Kubernetes, and based on two core elements:

- The kubelet is the Kubernetes Agent that executes the plan and drives container runtime and manipulates the core objects of the Kubernetes API
- The kube-proxy is a service that drives iptables to make sure that the translation between services and pods efficiently rendered.

Kubernetes-worker charm automatically recognizes GPU-based servers and installs all necessary packages and drivers to allow the containers to benefit from GPU-based acceleration.

Etcd

Etcd is the database holding the state of the cluster at any point in time. It is a critical element in Kubernetes, since damaging the DB (or worse, losing it) will irreparably impact the state of the cluster. Also, anyone with write access on etcd can potentially modify the state of the cluster.

Flannel (Container networking)

Flannel is a virtual network that gives a subnet to each host for use with container runtimes.

This charm will deploy flannel as a background service, and configure CNI for use with flannel, on any principal charm that implements the Kubernetes-cni interface.
Optionally Calico can be used as the overlay networking for the workloads, which is recommended for production clusters.

Container runtime

In the current release of Ubuntu Kubernetes, container runtime is moved out of the responsibility of Kubernetes Worker, and it's been decided to use a separate charm for configuring the runtime. Currently, Docker and Containerd are the only supported runtimes.

Optionally, Kata extension can be deployed in conjunction with the container runtime, allowing to run lightweight virtual machines instead of classic containers as the workloads.

Resource charms

This topic describes the resource charms used by Charmed Kubernetes.

API Load Balancer

The API load balancer is a simple nginx service that exposes a proxied endpoint to the Kubernetes API. It converts the API default port 6443 to run on the more classic 443.

Hacluster

There is an option available for defining the Virtual IP to provide access to the Kubernetes API load balancer.

To use virtual IP address(es), the clustered nodes must be on the same subnet, such that:

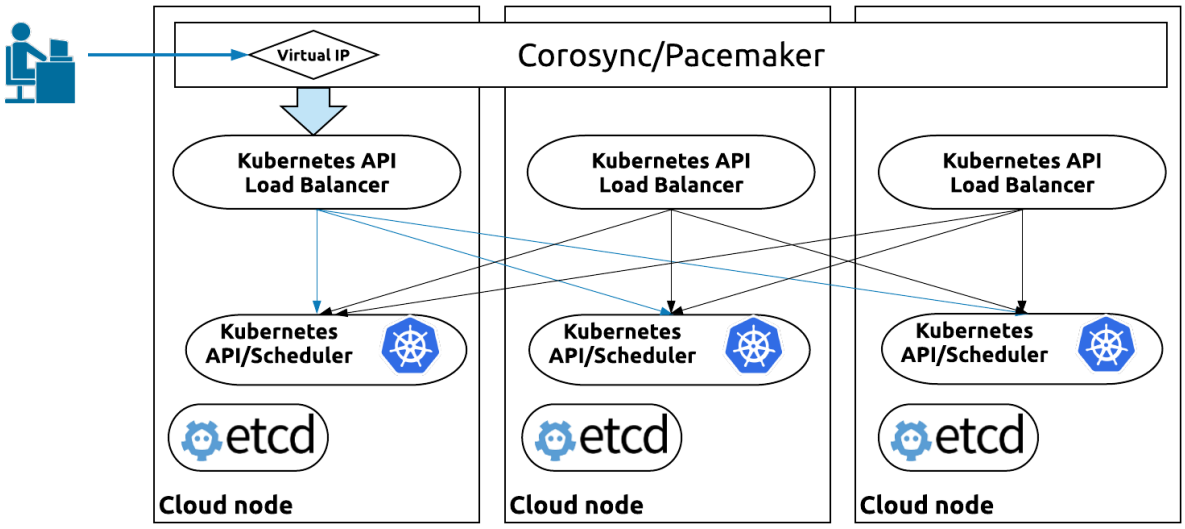
- The VIP is a valid IP address on the subnet for one of the node's interfaces;
- Each node has an interface in said subnet.
- The VIP becomes a highly-available API endpoint.

At a minimum, an option `ha-cluster-ip` must be set in Kubernetes-master charm to use virtual IP HA.

In this architecture, the Kubernetes-master component deployed as three units onto different physical hosts with their IP addresses.

Kubernetes-API load balancer also deployed as three units. One of them will have a Virtual IP associated with it, providing the endpoint to Kubernetes API and balancing the traffic between available units of Kubernetes-master.

The following diagram explains the availability of Kubernetes-API in Charmed Kubernetes:



Network space support

Kubernetes charms support the use of Juju Network Spaces, allowing the charm to be bound to network space configurations managed directly by Juju. API endpoints can be bound to distinct network spaces supporting the network separation of all existing endpoints. Network spaces mapped to different VLANs accordingly to manage by MAAS, making networking management transparent and flexible.

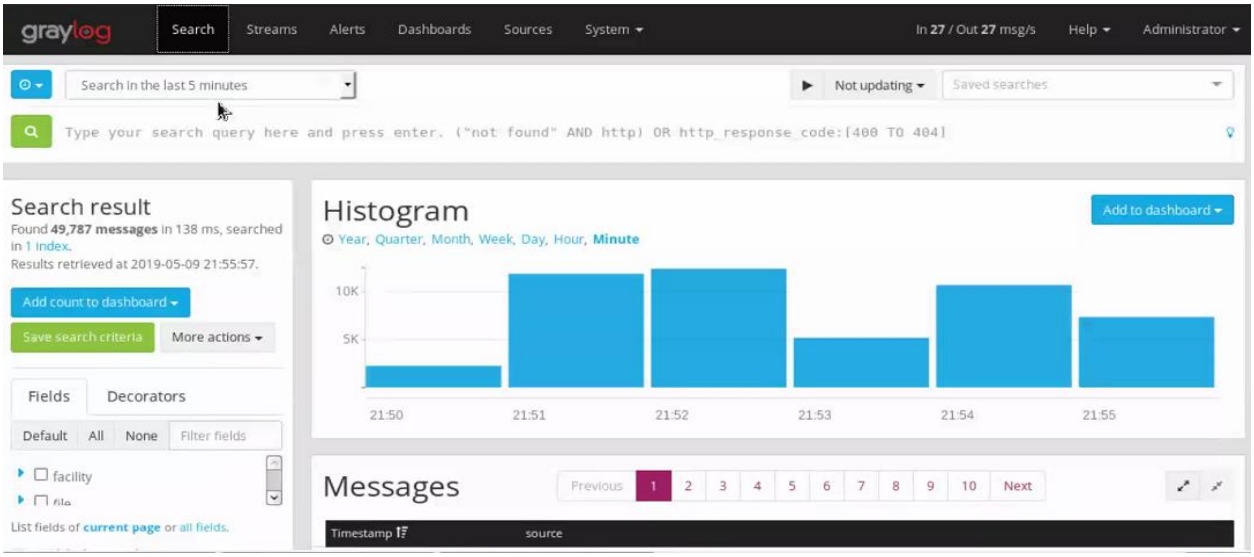
CHAPTER 6 MONITORING AND LOGGING TOOLS

This chapter describes the services, which were deployed to manage and monitor the Kubernetes cluster. Canonical has a specially designed architecture for its customers to manage and monitor Kubernetes clusters. Those services are an optional part of Canonical Kubernetes Discoverer services. If customers have different requirements as part of Canonical Kubernetes Discoverer, we design the architecture for customers and do the deployments.

Logging the cluster

GRAYLOG

Graylog is the solution for aggregating and managing the logs from various components of the cluster, as well as for providing visualization of the logs.
Below is the sample of the log aggregation dashboard.



ELASTICSEARCH

Elasticsearch is a distributed database used for storing indexed logs and acts as the backend for Graylog.

FILEBEAT

As a log forwarder, Filebeat tails various log files on the client-side and quickly sends this information to Graylog for further parsing and enrichment, or Elasticsearch for centralized storage and analysis.

Monitoring the cluster

From an architectural standpoint, the monitoring suite consists of Telegraf (host metric collection), Prometheus (monitoring server), Grafana (monitoring dashboard), and additional services that are gathering metrics from the API endpoints of Kubernetes and Ceph.

Prometheus

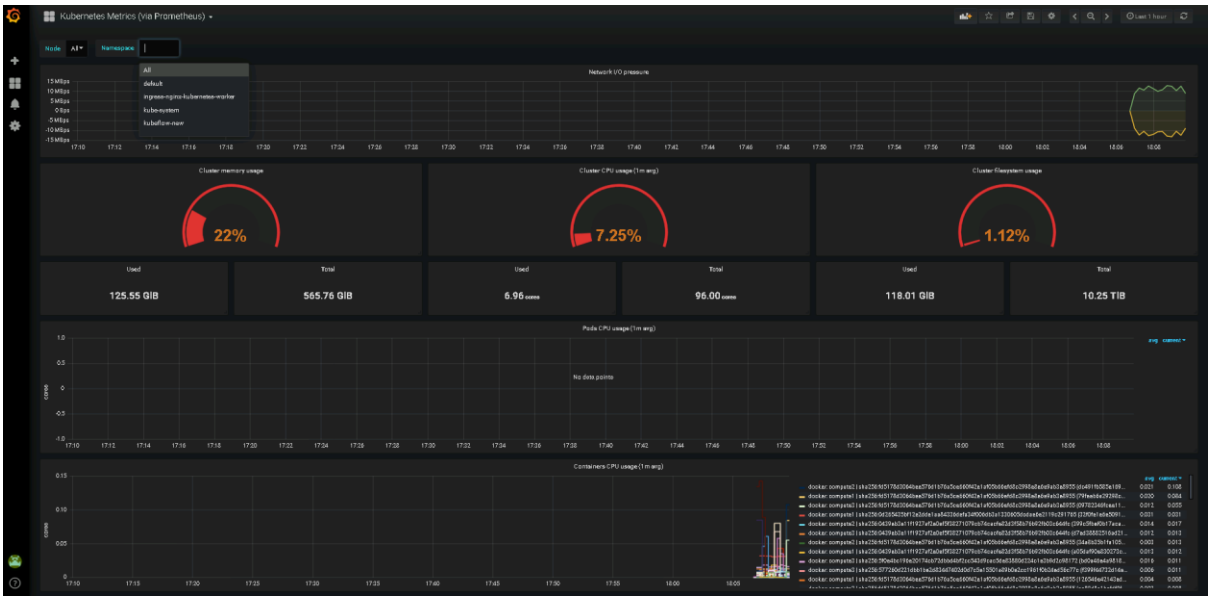
Prometheus is a systems and services monitoring system. It collects metrics from configured targets at given intervals, evaluates rule expressions, displays the results, and can trigger alerts if some condition is observed to be true.

Grafana

Grafana is the leading graph and dashboard builder for visualizing time series metrics.
The graphic below displays the monitoring tool dashboard.

Ceph-related dashboard

Kubernetes-related dashboard



Telegraf

Telegraf is a client-side system that collects information about the status of the host services and makes them available for pulling by any monitoring solutions (in this architecture Prometheus).

Prometheus-ceph-exporter

This application collects the metrics of the Ceph cluster via its API, such as current capacity and utilization, status of main components, capacity of the pools etc, and makes them available for Prometheus.

Appendix A References

Please see the following resources for more information.

Supermicro documentation

- https://www.supermicro.com/en/Aplus/system/1U/1123/AS-1123US-TR4.cfm
- https://www.supermicro.com/en/Aplus/system/2U/2123/AS-2123BT-HNR.cfm

Canonical documentation

- Charmed Kubernetes
- https://maas.io/
- https://wiki.ubuntu.com/
- https://www.ubuntu.com/
- BootStack: Fully managed operations of Kubernetes
- http://www.ubuntu.com/download/server
- Canonical Kubernetes Services
- Ubuntu Advantage Support

Kubernetes Documentation

- Kubernetes Community documentation
- Repository of Ubuntu Kubernetes components

To Learn More

If additional services or implementation help is required, please contact Supermicro sales representative or Canonical directly.