CANONICAL

Telecom infrastructure the open source way

October 2021

The Service Provider technology landscape is complicated at the best of times, and like most technology-based industries, the only real constant is the state of continuous change forced by evolutions.

Many external factors contribute to the complexity of modern service providers, none more present than the major standards-based evolutions for the core and radio technologies in mobile service providers. Combine this with the myriad of companies offering solutions to overcome individual use-case demands and needs, and the complexity increases exponentially.

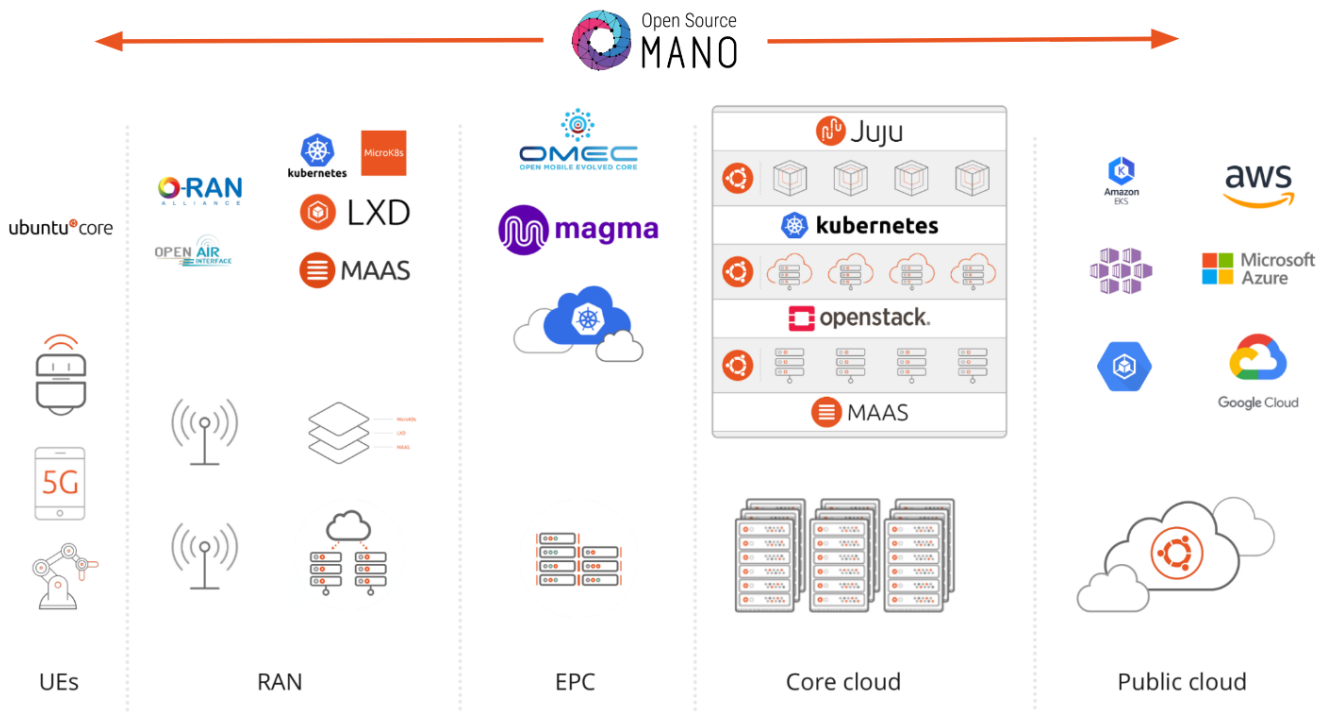
Your source of innovation

Scale and complexity brings similar challenges, big or small. Be it hybrid cloud, public cloud, on-prem large datacenter or highly distributed infrastructure— Canonical open source technology building blocks provide a unified approach to enabling service providers to meet any current or future use-cases, from OpenRAN, next generation Core (5G and beyond) or AI at the edge.

Open source software is becoming increasingly visible in the telecommunications space, starting with VNFs, which mostly run on OpenStack and CNFs living in the Kubernetes world. Thanks to that interest and connecting traditional telco ecosystems with open source innovators, we now see many excellent open source telco projects. Some of them matured enough to be running in Tier 1 mobile networks already - such as ETSI OSM, OpenRAN, Magma, OMEC amongst others.

This paper will walk you through different telecom infrastructure components step by step from an industry perspective, and discuss how open source software can greatly improve the price performance for you. Introducing open source solutions into highly regulated and secure environments is not easy. In this paper, you will also find solutions to the most common issues people face when they introduce open source to their networks.

Telecom infrastructure



There are many ways to categorise telecom infrastructure. In this paper, we will do it based on the compute power of the piece as follows:

1. We start with the smallest ones, user equipment (UE) which are typically single CPU devices, phones, drones, robots or routers all benefit from dedicated OS.

2. Higher up, we have the cell tower sites, where infra is needed to run radio access network (RAN) software and edge compute next to it in the form of MEC, hosting third party workloads that require low latency.

3. After this, we have Evolved Packet Core (EPC), typically for private mobile network use cases like IoT, CBRS or smart city.

4. Next, we enter the realm of big data centers, going to the core network and main operator clouds with huge OpenStack or Kubernetes based environments hosting the beating heart of the telecom network.

5. The last category, even though this is not an operator owned infrastructure, is public clouds. Many mobile network operators (MNOs) see benefits of using hybrid-cloud approach for scalability and opening up completely new business models.

Now let's go into the details of each category and see what type of infrastructure delivers best price performance.

Core network

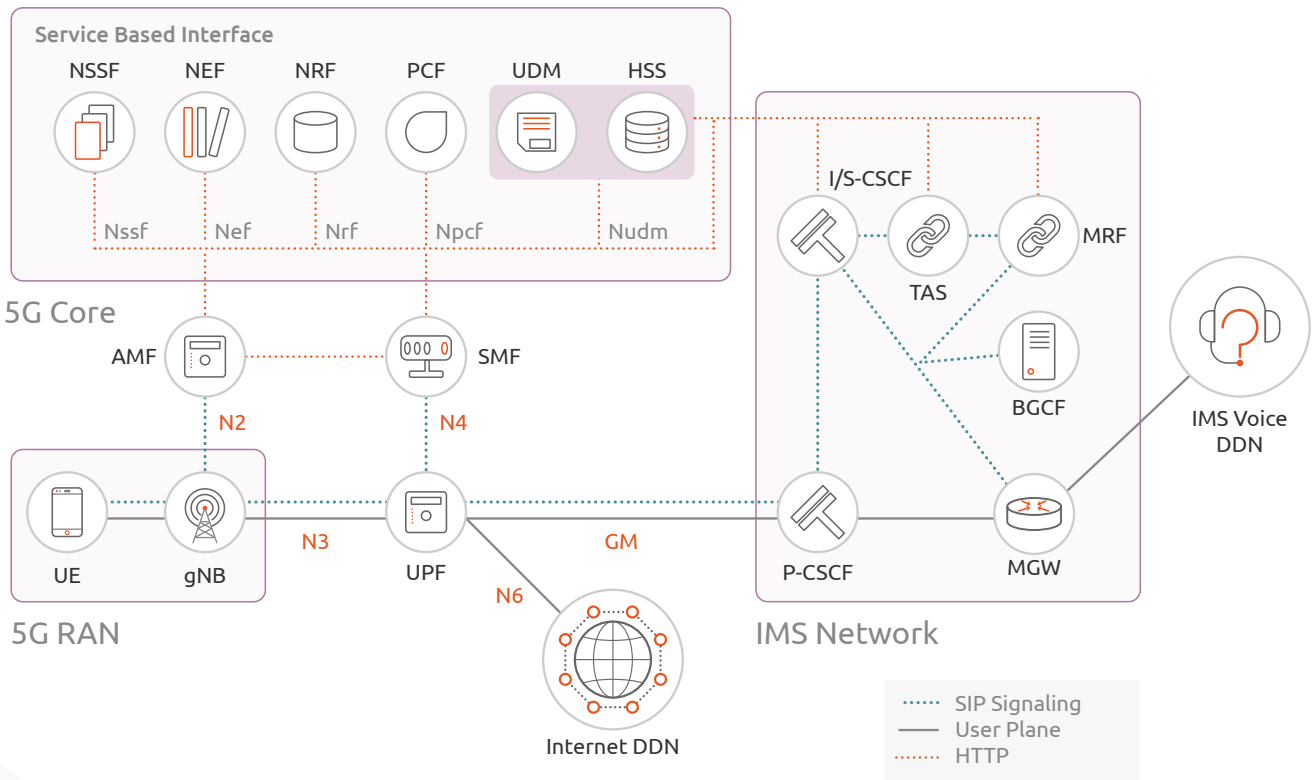
Traditionally, in telecommunication networks, the term “core” is used by service providers and refers to the high capacity communication facilities that connect primary nodes. A core/backbone network provides paths in exchange of information between different sub-networks.

Core networks usually offer the following features:

- **Aggregation:** The top degree of aggregation can be seen in a service provider network. Next, in the hierarchy within the core nodes, is the distribution networks, followed by the edge networks.
- **Authentication:** Determines whether the user demanding a service from a telecom network is permitted to complete the task within the network.
- **Call control or switching:** Determines the future span of a call depending on the processing of call signaling.
- **Charging:** Deals with the processing and collation of charging the data created by multiple network nodes.
- **Service invocation:** A core network executes the service invocation task for its customers. Service invocation may occur in line with certain activity (such as call forwarding) by the users or unconditionally (such as for call waiting).
- **Gateways:** Are used in core networks for accessing other networks. The functionality of gateways depends on the kind of network to which it is connected.

In 5G, which is the current network evolution standard, we have the 5G core network, which enables the advanced functionality of 5G networks. 5G core uses a cloud-aligned Service-Based Architecture (SBA) to support authentication, security, session management and aggregation of traffic from connected devices, all of which requires the complex interconnection of network functions.

The main components of the 5G architecture are:



3GPP defines them as follows:

User Equipment (UE) like 5G smartphones or 5G cellular devices connect over the 5G New Radio Access Network to the 5G core and further to Data Networks (DN), like the Internet.

- **The Access and Mobility Management Function (AMF)** acts as a single-entry point for the UE connection.

Based on the service requested by the UE, the AMF selects the respective **Session Management Function (SMF)** for managing the user session.

The **User Plane Function (UPF)** transports the IP data traffic (user plane) between the User Equipment (UE) and the external networks.

- The **Authentication Server Function (AUSF)** allows the AMF to authenticate the UE and access services of the 5G core.

Other functions like the Session Management Function (SMF), the Policy Control Function (PCF), the Application Function (AF) and the Unified Data Management (UDM) function provide the policy control framework, applying policy decisions and accessing subscription information, to govern the network behavior.

PNF, VNF, CNF and more *NF's

Now after looking into what a core network is and that it is composed of many network functions, it's time to explore different open source infrastructure options. Network functions come in various sizes and forms with the three main types being:

- **Physical network functions (PNF):** PNF refers to an actual piece of networking hardware such as a router, switch, firewall, etc. In a PNF, the hardware and software are tightly bound to each other. PNFs are widely deployed and are preferred when the consumer can afford vendor lock-in and does not want to get involved with the installation of the network.

- **Virtual network function (VNF):** VNF refers to the software implementation of a networking function/module. These allow the features of the PNF to be replicated on software. VNF runs on Virtual Machines (VMs) that allow it to be run on commodity hardware, essentially eliminating vendor lock-in. VNFs are relatively new as compared to PNFs and are not as widely deployed. These are ideal for consumers who want to have control over installation and deployment. They are also preferred when portability/migration is required.

- **Cloud-native network functions (CNF):** CNF refers to containerizing a PNF or VNF and running it on the cloud. This includes creating microservices that communicate with each other to form a network. These microservices run on docker containers and management of the containers is done using Kubernetes (k8s) as well as the CI/CD principles in a cloud architecture. CNF and KNF (Kubernetes-based Network Functions) are used interchangeably in the NFV domain as both are based on cloud-native concepts. CNFs are preferred for scenarios where the consumer requires added features to VNFs such as elasticity, scalability, availability, and ease of installation and deployment. All of these features are enabled by cloud and container technologies.

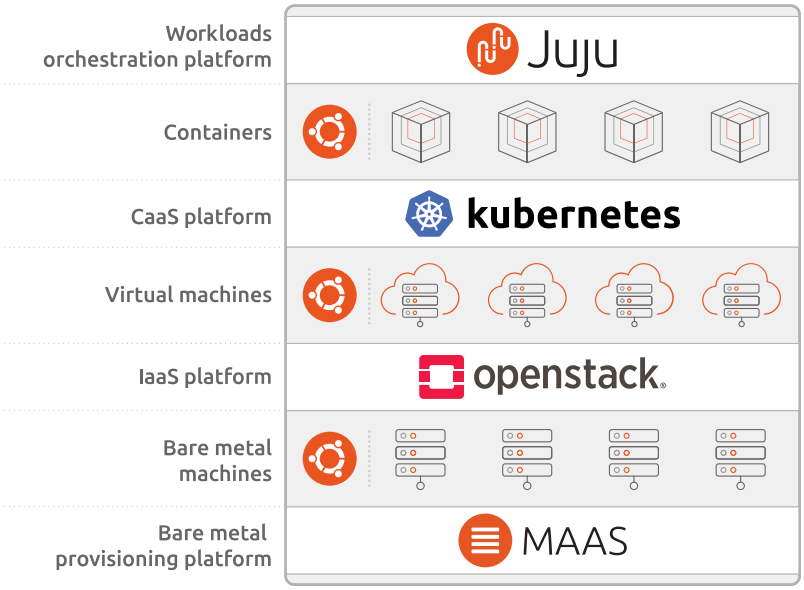
Most mobile networks are not greenfield deployments created last year, typically they have all technologies combined, with 2G, 3G, 4G and 5G coexisting in the same network. This means there are all of the above network function types there.

Another point to consider is growing complexity. New technologies are being introduced every year, with different architectures, networking and configuration parameters. Unfortunately, at the same time, there is a huge pressure on cost cutting and operations, and maintenance teams are not growing at a similar rate.

Best core network infrastructure will be able to:

- Recognize and instantiate bare metal
- Create and manage virtual machines
- Create and manage containers
- Provide the same automation and tools regardless of the substrate
- Not only deploy but also perform day1 and day2 operations

Open source ecosystem is all about choice and the freedom to use tools you like as per your business needs. Therefore, there are several ways to build such infrastructure, but the most popular way Canonical observes with their partners is:

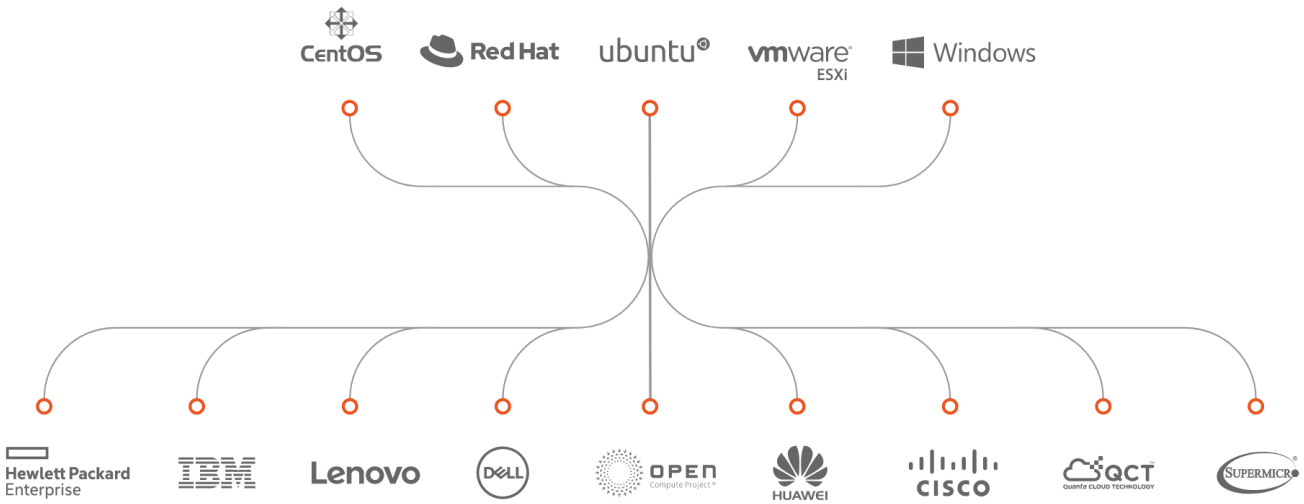


Open source components used in this infrastructure blueprint are:

MAAS

MAAS is Metal As A Service, a service that lets you treat physical servers like virtual machines – instances – in the cloud. No need for you to manage servers individually– MAAS turns your bare metal into an elastic, cloud-like resource. Main MAAS features include:

- Bare metal cloud with on-demand servers
- Remote edge cluster operations
- Infrastructure monitoring and discovery
- Ansible, Chef, Puppet, SALT, Juju integration
- Super fast install from scratch
- VMWare ESXi, Windows, CentOS, RHEL, Ubuntu
- Custom images with pre-installed apps
- Disk and network configuration including machine cloning
- API-driven DHCP, DNS, PXE, IPAM
- REST API for provisioning
- LDAP user authentication
- Role-based access control (RBAC)
- Hardware testing and commissioning

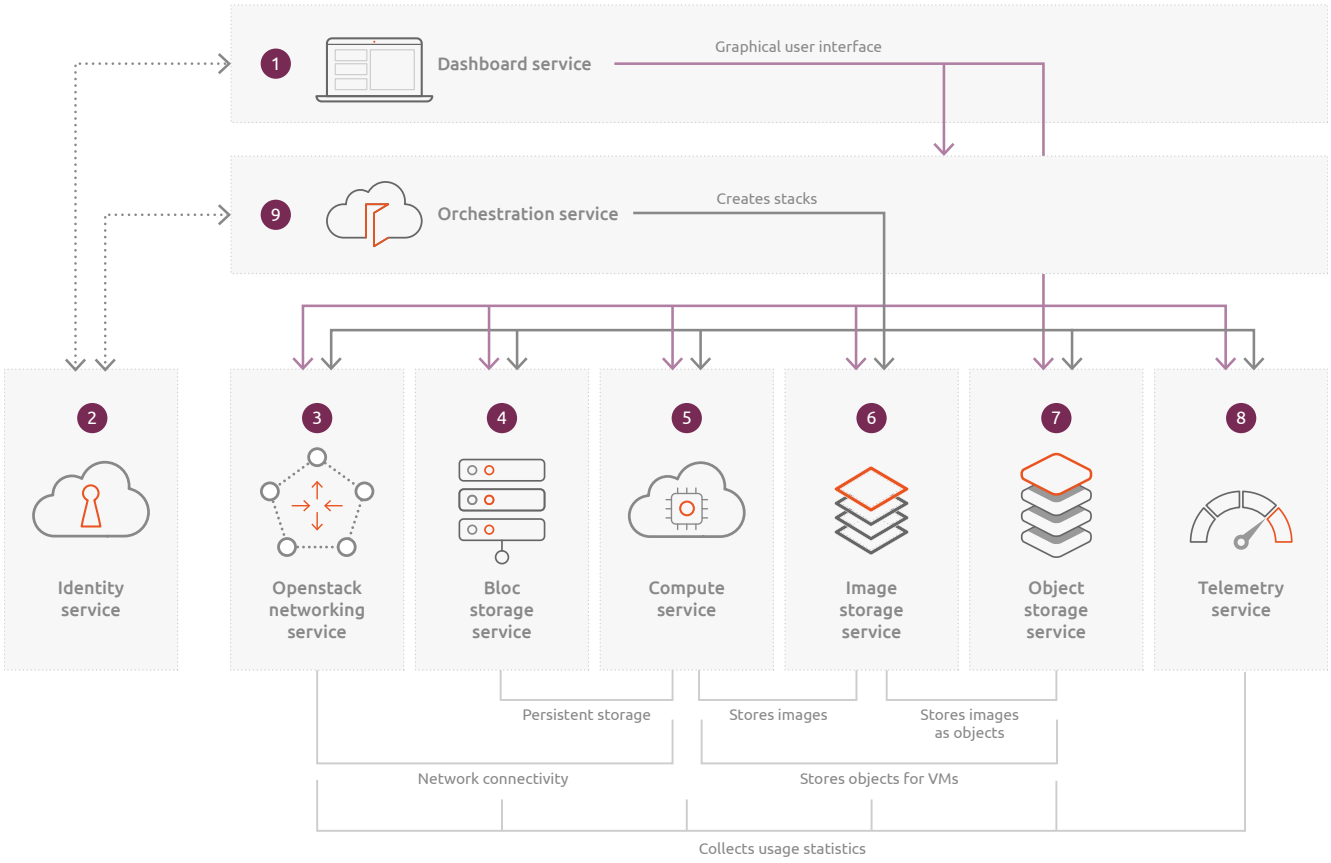


Charmed OpenStack

OpenStack is a carrier-grade VIM (Virtual Infrastructure Manager). In turn, Charmed OpenStack is an OpenStack distribution engineered for the best price-performance. Charmed OpenStack is deployable, maintainable and upgradable economically. This is achieved by putting full automation around its deployments and exposed deployment operations. Key features of Charmed Openstack from VNF perspective include:

- Model-driven deployments and operations
- Fully automated lifecycle management and day-2 operations
- Predictable release cadence and upgrade path
- Smooth upgrades to new OpenStack versions
- IaC approach and CICD integration possibilities
- OVS hardware acceleration
- SR-IOV, DPDK
- CPU pinning
- NUMA topology
- Hugepages
- GPU and PCI passthrough

On a picture below we can see an overview of key components that OpenStack provides and their functions:

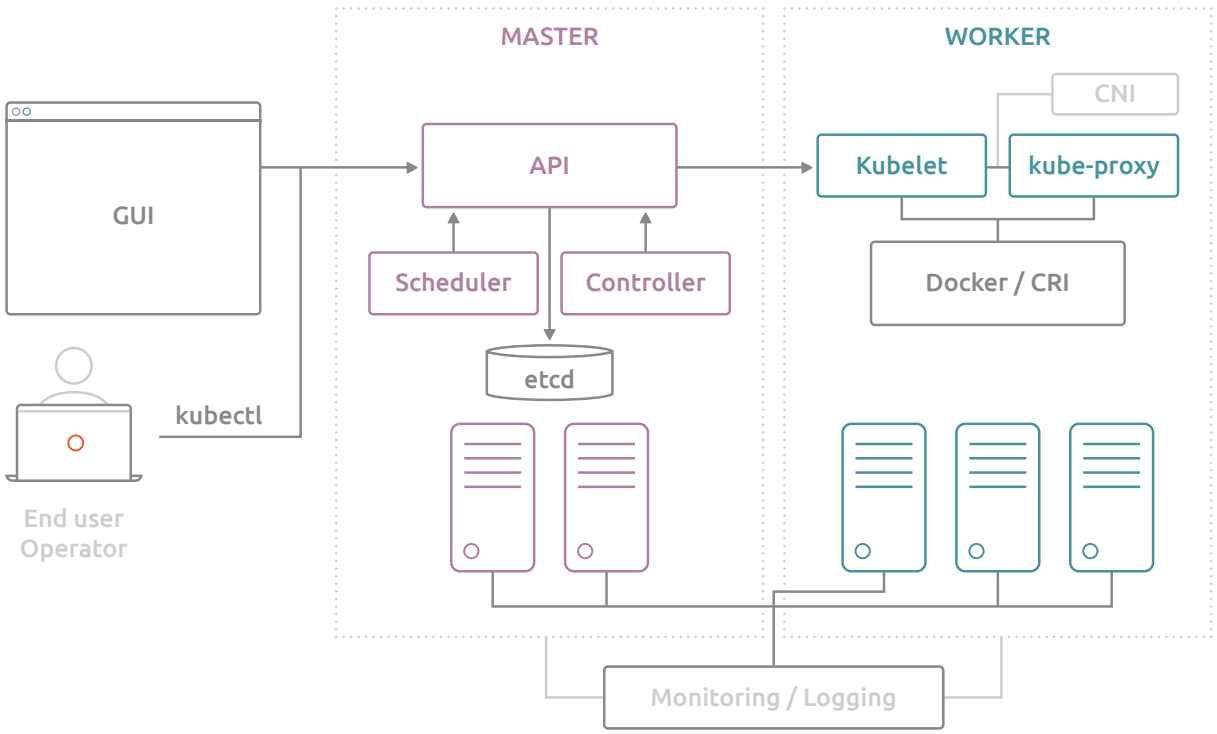


Charmed Kubernetes

Kubernetes, or K8s for short, is an open source platform, pioneered by Google, which started as a simple container orchestration tool but has grown into a cloud native platform. It's one of the most significant advancements in IT since the public cloud came to being in 2009 and has an unparalleled 5-year 30% year-on-year growth rate in both market revenue and overall adoption. Charmed Kubernetes gives you perfect portability of workloads across all infrastructures, from the core network to the public cloud. With a strong focus on CNF and AI/ML, Ubuntu is the platform of choice for K8s in telco. Key features of Charmed Kubernetes from CNF perspective include:

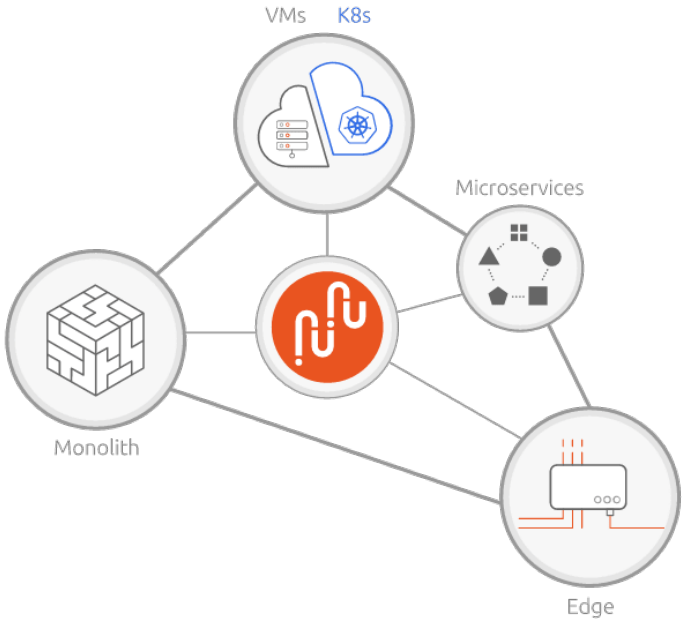
- Pure upstream - no vendor-specific APIs
- Model-driven, declarative cluster operations
- Multi-cloud and hybrid-cloud compatibility
- Highly available configuration
- Pluggable CNI, CSI, CRI and observability stack
- Support for DPDK, CPU pinning, SR-IOV, Hugepages
- Fine-grained service placement
- GPU acceleration
- Support for third-party components and services
- Single-subscription enterprise support

This is how Charmed Kubernetes deployment looks like:



Juju

Juju is a Charmed Operator Framework, composed of a Charmed Operator Lifecycle Manager and the Charmed Operator SDK. Deploy, integrate and manage Kubernetes, container and VM-native applications seamlessly across hybrid clouds. Juju drives Day 0 through Day 2 operations in your complex environment.



Juju can:

- Save your team endless hours of script management
- Reduce OPEX
- Ensure redundancy and resiliency
- Monitor all activity across substrates
- Maximise price performance of your hybrid cloud architecture

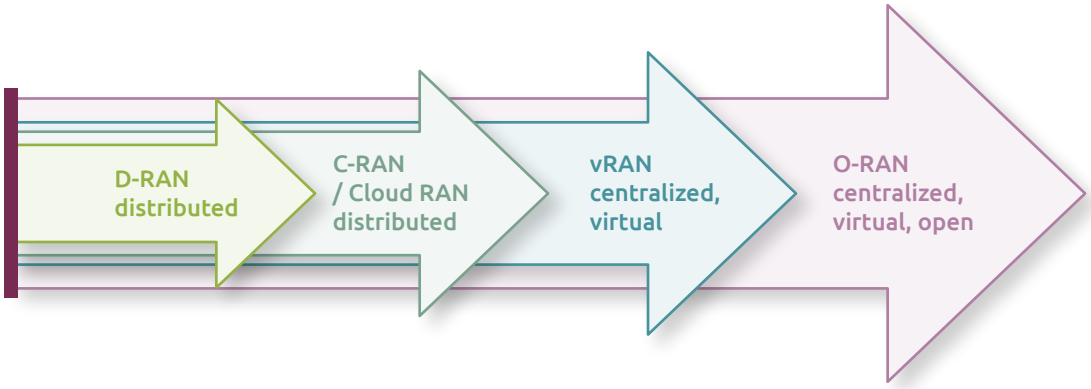
Juju is a powerful tool, and in the context of core network infrastructure, it is used to provide a high level of automation at scale. It's successfully used by many Tier 1 network operators and adopted by ETSI as part of the ETSI Open Source MANO (OSM). There is a separate section on automation later in this paper, where the full context and the power of Juju will be explained. Stay tuned.

Radio network

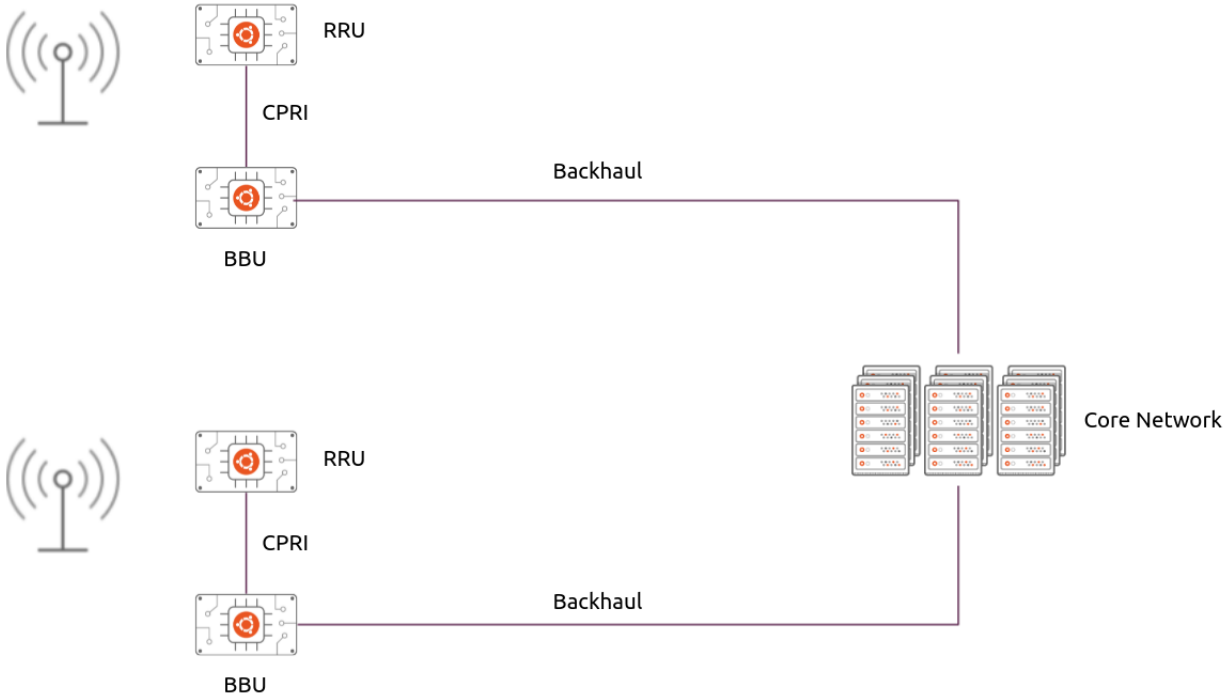
Infrastructure for the core network is similar to a typical IT data center, which has established best practices and way of working. On the other hand, we have a hastily growing open source radio network with projects like OpenRAN and great commercial success stories, like Rakuten. In order to deep dive into one of most exciting topics in telco right now, one needs to start with some basic terms and definitions.

A radio access network is a technology that connects individual devices to other parts of a network through radio connections. It is a major part of modern telecommunications. LTE and 5G network connections for mobile phones are examples of radio access networks.

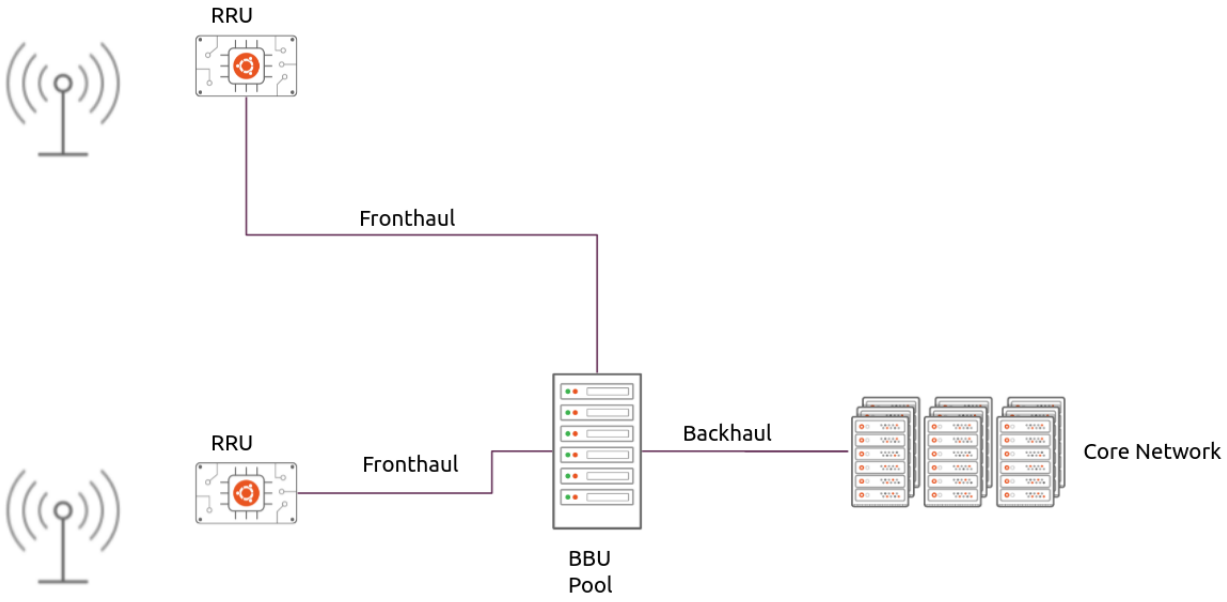
From an infrastructure perspective, we have a split for D-RAN, C-RAN, vRAN and O-RAN:



D-RAN stands for Distributed RAN. D-RAN is a classical setup. RRU (Remote Radio Unit) and BBU (BaseBand Unit) are co-located at every cell site. They run proprietary applications on specialized hardware. Each radio site with all of its functions are in a single location and connected back to the core network through backhaul as shown below:

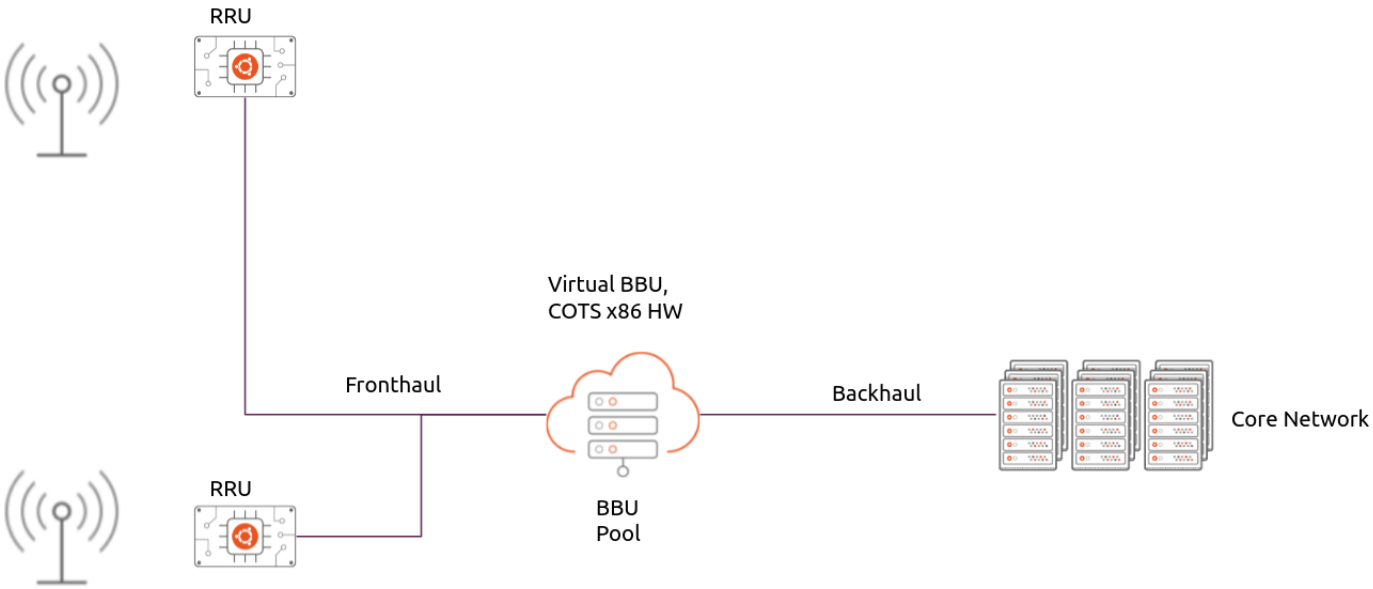


C-RAN (Also CRAN) stands for Centralized RAN or Cloud RAN. In C-RAN, the BBUs from several sites are pooled together and located in one location. This leads to more efficient use of computing resources. Still software and hardware is coming from a single vendor and runs in a bare metal setup.



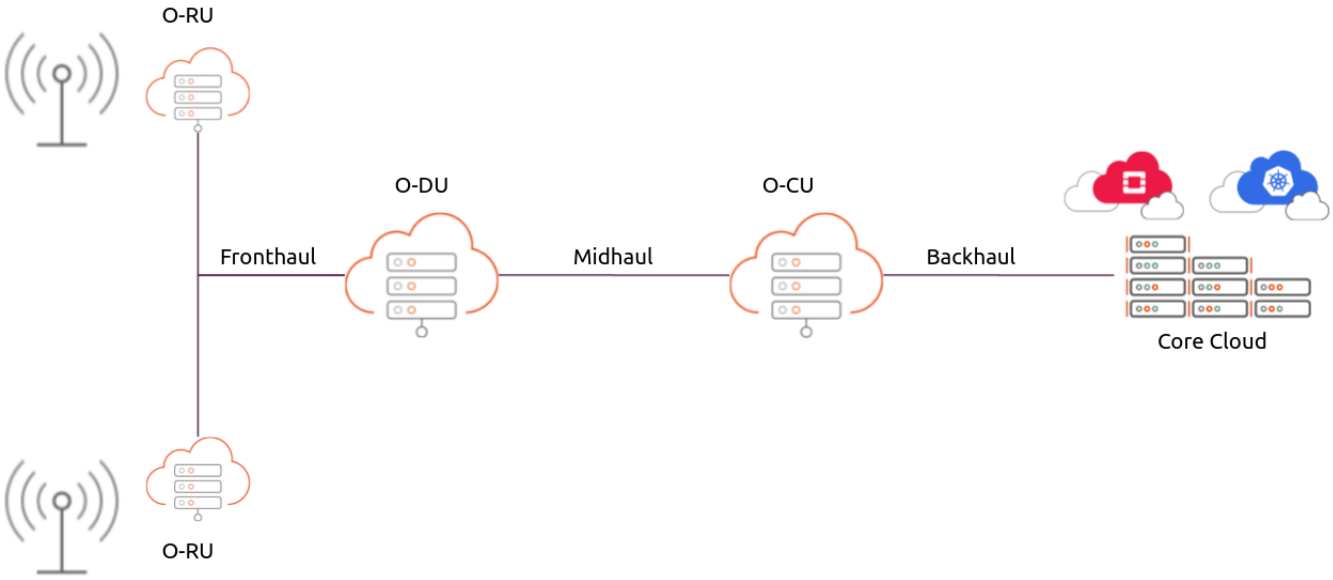
vRAN decouples the software from hardware by virtualizing Network Functions. It uses virtual machines (VNF approach) or containers (CNF approach) to deploy CU and DU on top of COTS servers.

vRAN and C-RAN are very similar. The only difference is that C-RAN uses proprietary hardware while vRAN deploys on top of COTS servers, the same as you can find in a typical IT datacenter.



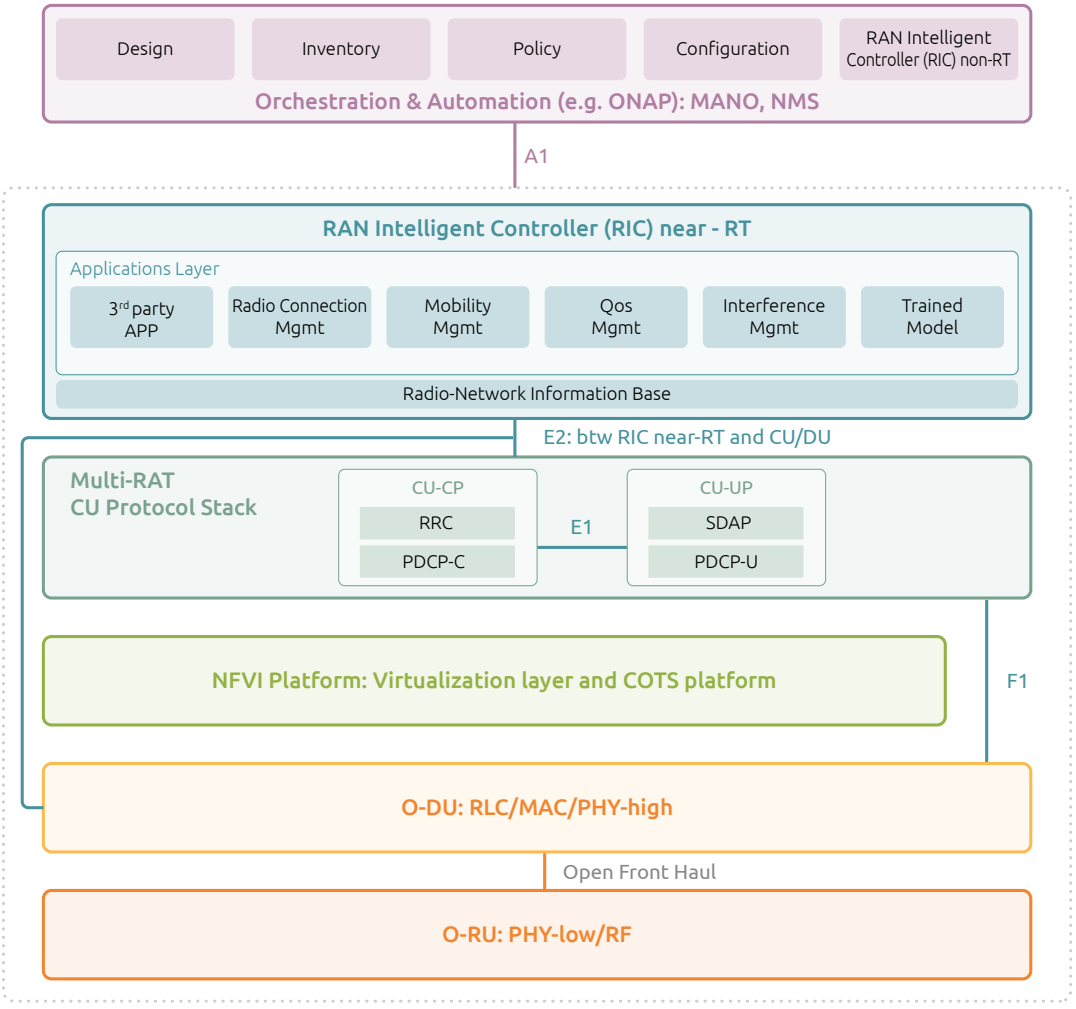
vRAN is a closed network, as RU, DU and CU are provided by a single vendor and use internal means of communication instead of open interfaces.

The O-RAN alliance did a great job creating a specification for open fronthaul, mid haul and backhaul interfaces. This means that finally you can mix and match different vendors and use different approaches depending on what is the use case.



The Open RAN infrastructure

As described above, OpenRAN is the most open and mobile-network-operator-friendly way of setting up the radio access network. In the picture below, you can see how O-RAN alliance defines its architecture:



OpenRAN solutions require low latency and high throughput. They are built based on custom L1/L2 implementation or using specialized platforms like Intel FlexRAN or NVidia Aerial. They require an infrastructure where:

- You can deploy containers without any performance or security compromise
- You have support for various accelerators, such as FPGA, SmartNICs, GPUs
- Operating system supports normal, low-latency and real-time kernel options
- 1000s of sites can be managed in fully automated fashion
- API-driven re-building of a site has completely different configurations
- You are able to provide tenant separation for MORAN and MOCN use-cases

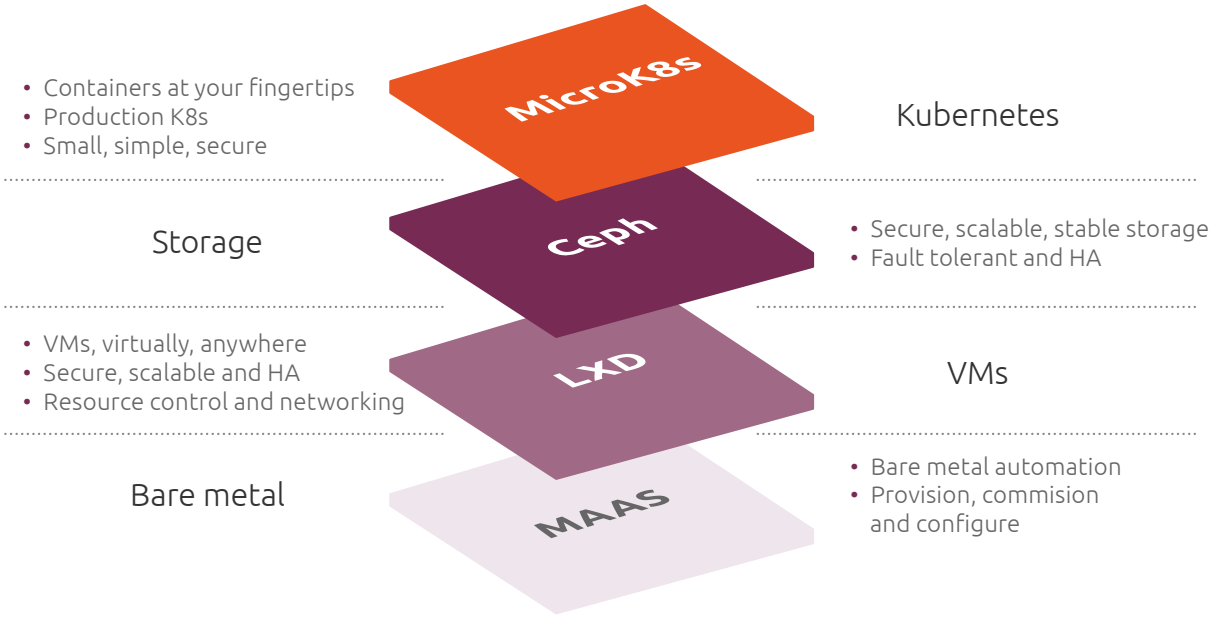
One of the best ways to achieve that is using micro clouds based on Ubuntu.

Micro clouds

Micro clouds trade the exponential scalability of public and private clouds for security, privacy, governance, and low-latency of decentralised environments. They reproduce the APIs and primitives of the big clouds at the scale of the edge — tiny clusters replicated over thousands of edge sites. A micro cloud strategy helps build an ecosystem of many interconnected systems over a complex network.

Micro clouds are mostly targeted for enterprises and mobile service providers to easily deploy and lifecycle manage distributed micro clouds - bare metal compute clusters of between 3-100. Commonly uses off-the-shelf servers - to meet not only initial use-case requirements, but also be able to easily cope with future use-cases, without having to retool or redeploy.

Micro cloud typically consists of following layers:



Depending on the particular site purpose, there might be different combinations used, i.e., Bare metal Kubernetes site with MAAS and MicroK8s and ephemeral storage only for O-DU or even plain MAAS + Ubuntu with low latency kernel for O-RU. Using open source based infrastructure gives you a freedom of choice and ability to craft your architecture to exactly match the requirements.

The newly mentioned open source projects are:

MicroK8s

MicroK8s is a low-ops, minimal production Kubernetes, for devs, cloud, clusters, workstations, Edge and IoT. It's highly opinionated, with most of the choices made by Canonical to provide:

Best of breed

Enterprise	Platforms	Networking	Usability	Built-in
Clustering	Windows	CoreDNS	Dashboard	Registry
Auto-updating	macOS	Ingress	Prometheus	Knative
Confinement	Intel	Istio	Fluentd	Kubeflow
Storage	ARM	Linkerd	Jaeger	GPU support

Main benefits of MicroK8s are:

- Pure upstream - no vendor-specific APIs
- Edge, IoT and appliance compatibility
- Zero-ops user experience, minimal intervention needed
- Highly available configuration
- A single package and 60-second setup for a full Kubernetes
- Single-node and multi-node cluster support
- Opinionated distribution with “sensible defaults”
- Automatic updates and security patching

LXD

LXD is a next generation system container and virtual machine manager. It offers a unified user experience around full Linux systems running inside containers or virtual machines.

It’s image based with pre-made images available for a wide number of Linux distributions and is built around a very powerful, yet pretty simple, REST API.

MEC/Edge

MEC, as ETSI defines it, stands for Multi-access Edge Computing and is sometimes referred to as Mobile Edge Computing. MEC is a solution that gives content providers and software developers cloud-computing capabilities that are close to the end users. This micro cloud deployed on the edge of mobile operators’ networks has ultra low latency and high bandwidth, which enables new types of applications and business use cases.

Roll-out of new 5G sites costs a lot of money, and MEC is the best way to monetize this investment. In order to do it effectively and securely, mobile network operators can choose one or several of the below options:

MEC capacity in B2B model

The simplest approach would be to allow enterprises to put their workloads on MEC infrastructure and charge per capacity consumed. There are many businesses that require wide coverage, even across the entire country. Self-driving cars, logistics, smart cities, utilities companies - all of them would benefit from putting apps closer to the users. With projects like Anbox, even gaming companies might be interested in widening their customer base by offering high-end games to low-end device users. This is also quite easy to secure, as there will be a small number of well-known and trusted tennants operating on the site. The main challenge would be to manage workload deployment. However, this can be done easily with open source tools, such as ETSI OSM, where they can be on-boarded just like any other network function.

MEC capacity in B2C model

Some operators are also trying to become cloud providers and open up the edge infrastructure to the public, where anyone can register and deploy their workloads. This could bring the biggest revenues but it is the most risky model. In order to do it properly, the infrastructure needs to be:

- Always up to date and monitored
- Providing strong tennant separation
- Easy to use, with simple way of deploying workloads to attract developers
- Very well documented, with tutorials, videos and some developer relations efforts
- Providing very good sandboxing and separation (no bare metal K8s, there needs to be a VM layer to prevent container sandbox escape attacks)

Become a service provider

Another route for an MNO is to provide services themselves. This can be lower level services, such as IoT devices on-boarding and monitoring or even business-specific services like smart home services. This obviously requires significant software engineering capabilities and marketing/sales; therefore, it's an option only for the big players. The good thing from a security point of view is that all the software is created in-house, and MEC sites are not exposed to third parties at all.

Integrate MEC infrastructure with hyperscalers

The last approach worth mentioning, which is happening in the North American market now, is a tight integration with hyperscalers, and providing services to consumers as partners. This approach has a lot of benefits:

- Hyperscalers already have customer base, self-service portals and billing machine
- They have huge software engineering capabilities
- MEC site can be treated as just another Kubernetes cluster in the account
- Least amount of efforts required on the MNO site

The key to cloud integration is to negotiate a good deal, and make the integration as smooth as possible. The latter can be addressed by using Ubuntu, as it's the most popular Linux distribution on the public cloud, and the tooling and automation based on Juju, which is already used in core networks natively supports all major public cloud vendors making the integration effortless. Public clouds, however, can be also used in other ways in the telco context.

Public cloud

There is no one size fits all cloud architecture. Developing the optimum cloud strategy requires evaluating your business needs and aligning them with the different solutions available. In a public cloud environment, compute, storage and other infrastructure resources are provided as a service by an external provider. Public clouds provide the ability to scale resource consumption virtually without limits. But in telco, another big advantage of public clouds is quick time to market when introducing new services thanks to immediate access to the resources. Another huge benefit of public clouds is access to advanced services , such as chatbots, speech to text, image recognition etc., which you don't need to implement yourself, just consume with an API call.

Obviously, MNO cannot move their entire compute pool to the public cloud. So, the only model we are looking at is a hybrid cloud. A hybrid cloud architecture combines the usage of a private cloud and one or more public cloud services with a workload orchestration engine between the platforms. A hybrid strategy provides maximum OpEx efficiency when executed in the right way. Hybrid clouds are ideal for workloads with diverse requirements. In telco context, this could be a way to address seasonality and traffic spikes, where you need a huge capacity only for a few days per year like New Years Eve.

Here is where a consistent approach to automation pays off. Juju natively supports all major hyperscalers API's and is a de-facto standard tool for micro clouds. As a result, for Charmed Kubernetes and Charmed OpenStack, you don't need separate code or configuration to migrate a workload from one environment to another. Moreover, you can scale a workload running locally to burst some Kubernetes pods to the cloud and instantly increase your service capacity.

Same data center primitives and automation

All of those open source projects are strictly adhering to the principles of the same primitives and automation approach. In the world of growing complexity, 5G rollout into existing networks and 6G specifications being worked on already, simplicity of tools and automation is the only way to cope with the changes. Imagine you have a simple PostgreSQL database in RIC in your radio network. You also have a database in IMS and several other core network elements. On top of that, you have a customer facing portal on AWS that also uses such a database. With Juju, you can deploy all of them from a single charm, using the same automation, just relating it to different surrounding components. Additionally, if you wish to migrate from AWS to Azure, nothing changes. You can use the same model of your application, including the database charm, and just deploy it on a different substrate. All of those actions are performed with a single command in CLI or click in the JAAS UI. This is the level of power and flexibility available to you with model-driven operations.

Future of telco infrastructure

We believe that the future of telco infrastructure would be highly distributed compute pools with even more disaggregation and distribution. This would require a powerful, AI-driven observability stack to be able to react to issues instantly. It also means there will be no centralized control point.

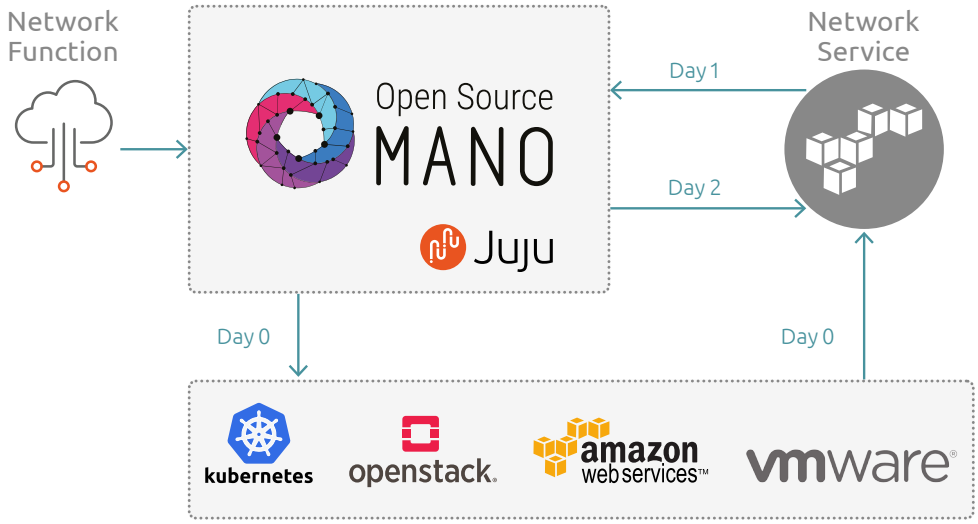
We imagine having a directory of computer pools, both bigger ones and micro clouds, their API endpoint addresses and geographical location, like a little extended DNS. We are currently working on such an observability stack, as well as extending all of our open source projects to be able to efficiently operate in such environments.

Consumable open source projects

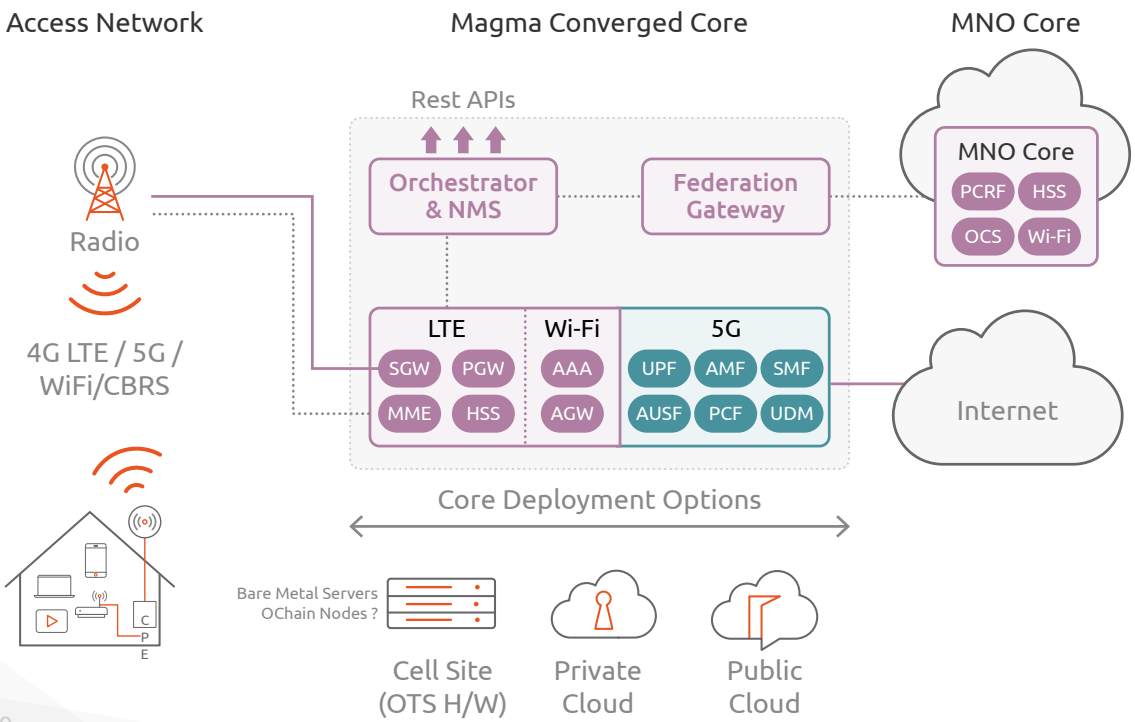
Besides the open source infrastructure, Canonical is committed to invest time and resources into most popular telecom related open source projects in order to increase open source adoption in the application layer. We are contributing to these projects and adding charms, so that you can deploy them with Juju, and get all of its benefits. Additionally, we are fixing CVEs by providing telco-grade support and directly contributing to projects.

Most notable mentions are:

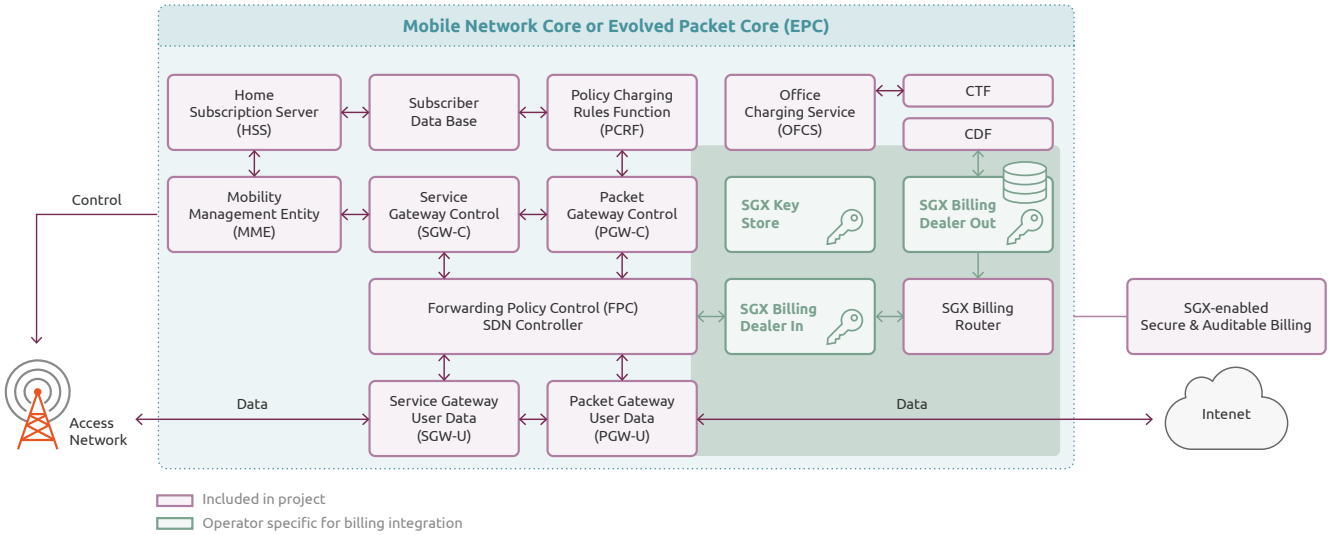
- ETSI OSM automates deployment and operations of network functions, and reduces OPEX, by using an open source implementation of ETSI NFV MANO.



- Magma is an open-source software platform that gives network operators an open, flexible and extendable mobile core network solution. Magma is designed to be 3GPP generation and access network (cellular or WiFi) agnostic. Running on Ubuntu, deployed with Juju Charms and on-boarded with OSM is how we like it.



- OMEC is a full-featured, scalable, high performance open source EPC. OMEC has been optimized to handle the ensuing onslaught of devices coming online as part of the move to 5G and IoT. It is designed to be used as a stand-alone EPC, and is also an upstream project for the COMAC platform that (among other things) is integrating mobile and fixed subscriber management functions. It provides 3GPP Release 13 compatibility and complete connectivity, billing and charging capabilities.



Summary

Open source projects are already more than ready to be used in the telecom industry. With proper automation and right architecture, they can be huge
OPEX savers, bring you out of the vendor lock-in trap and provide competitive advantage, thanks to the rapid innovation. We are committed to constantly improving what we do and would be happy to get your feedback on what is available now and what open source projects you would like to see in your network in the future.

Trusted by leading service providers, globally:

For more information about Canonical's solutions for telecommunications visit ubuntu.com/telco or call directly: (EMEA) +44 203 656 5291; (US) +1 737 204 0291.

Reach out to authors directly:
Arno Van Huyusteen - Telco Field CTO
Maciej Mazur - Product Strategy Manager

ubuntu
Delivered by Canonical