ICN R6 Architecture Document

Introduction

The ICN blueprint family intends to address deployment of workloads in a large number of edges and also in public clouds using K8s as resource orchestrator in each site and Edge Multi-Cluster Orchestration (EMCO) as service level orchestrator (across sites). ICN also intends to integrate infrastructure orchestration which is needed to bring up a site using bare-metal servers. Infrastructure orchestration, which is the focus of this page, needs to ensure that the infrastructure software required on edge servers is installed on a per-site basis, but controlled from a central dashboard. Infrastructure orchestration is expected to do the following:

- Installation: First-time installation of all infrastructure software.
  - Keep monitoring for new servers and install the software based on the role of the server machine.
- Patching: Continue to install the patches (mainly security-related) if new patch release is made in any one of the infrastructure software packages.
  - May need to work with resource and service orchestrators to ensure that workload functionality does not get impacted.
- Software updates: Updating software due to new releases.

The user experience needs to be as simple as possible and even a novice user should be able to set up a site.

Use Cases

1. SDEWAN Controller with Open source based SDWAN CNF and SDEWAN HUB to establish IPSEC tunneling between Edge Distributions with Service Function Chaining (SFC)
2. Composite vFirewall (vFW) to show case telco and cable use cases using EMCO

Where on the Edge

Nowadays best efforts are put to keep the Cloud native control plane close to workload to reduce latency, increase performance, and fault tolerance. A single orchestration engine to be lightweight and maintain the resources in a cluster of compute node, Where the customer can deploy multiple Network Functions, such as VNF, CNF, Micro service, Function as a service (FaaS), and also scale the orchestration infrastructure depending upon the customer demand.

ICN target on-premises edge, 5G, IoT, SDWAN, Video streaming, Edge Gaming Cloud. A single deployment model to target multiple edge use case.

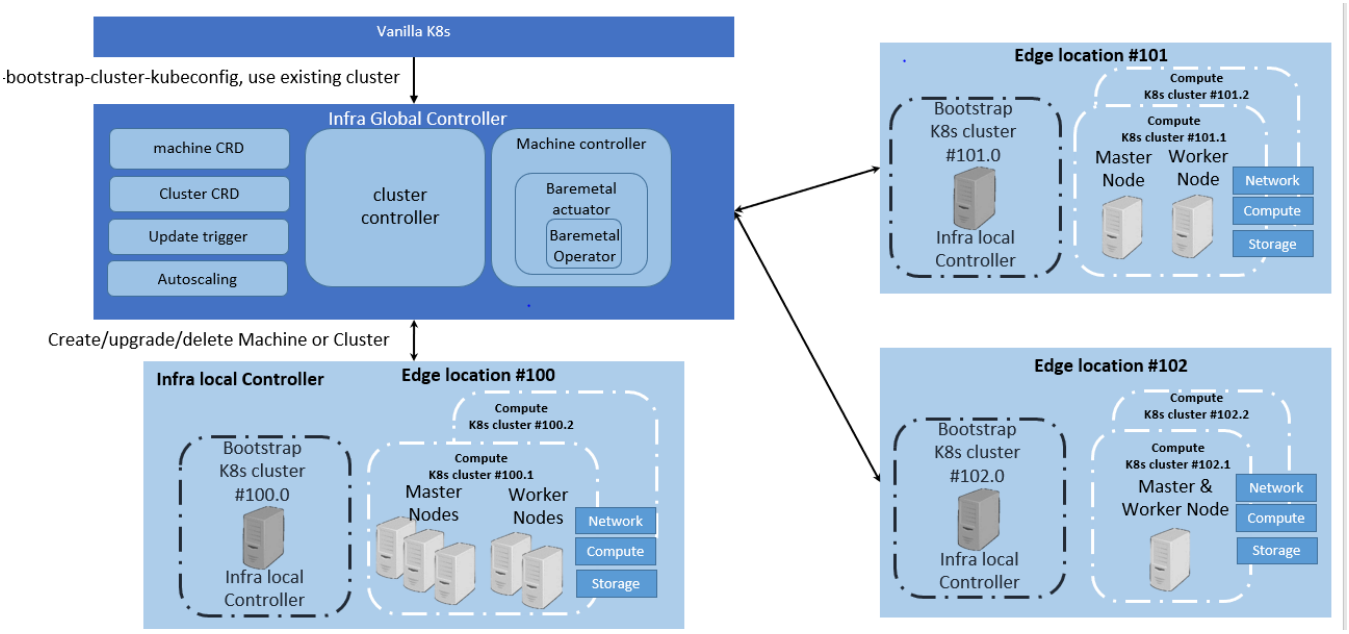
Overall Architecture

On an edge deployment, there may be multiple edges that need to be brought up. The Administrator going to each location, using the infra-local-controller to bring up application-K8S clusters in compute nodes of each location, is not scalable. Therefore, we have an "**infra-global-controller**" to control multiple "**infra-local-controllers**" which are controlling the worker nodes. The "infra-global-controller" is expected to provide a centralized software provisioning and configuration system. It provides one single-pane-of-glass for administrating the edge locations with respect to infrastructure. The worker nodes may be bare metal servers, or they may be virtual machines resident on the infra-local-controller. So the minimum platform configuration is one global controller and one local controller (although the local controller can be run without a global controller).

Since, there are a few K8s clusters, let us define them:

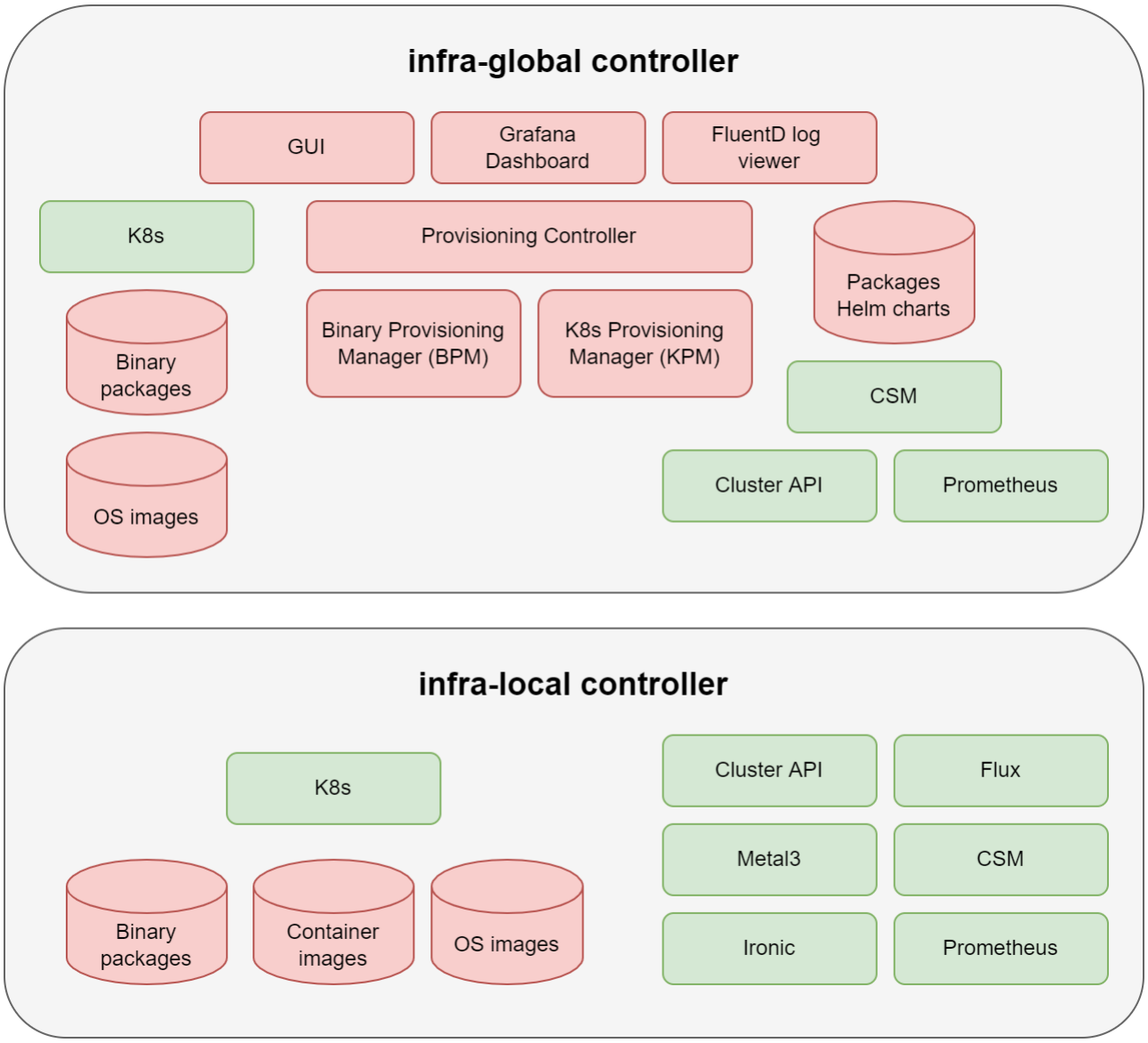
- infra-global-controller-k8s : This is the K8s cluster where infra-global-controller related containers are run.
- infra-local-controller-k8s: This is the K8s cluster where the infra-local-controller related containers are run, which bring up compute nodes.
- application-k8s: These are K8s clusters on compute nodes, where application workloads are run.

Flows & Sequence Diagrams



Each edge location has infra local controller, which has a bootstrap cluster, which has all the components required to boot up the compute cluster.

Platform Architecture



Infra-global-controller:

Administration involves

- First time bring up.
- Addition of new compute nodes in locations.
- Removal of compute nodes from locations
- Software patching
- Software upgrading

The infra-local-controller will be brought up in each location. The infra-local-controller kubeconfig will be made known to the infra-global-controller. Beyond that, everything else is taken care by the infra-global-controller. The infra-global-controller communicates with various infra-local-controllers to do the job of software installation and provisioning.

Infra-global-controller runs in its own K8s cluster. All the components of infra-global-controllers are containers. The following components are part of the infra-global-controller.

- Provisioning controller (PC) Micro Services
- Binary Provisioning Manager (BPM) Micro services
- K8s Provisioning Manager (KPM) Micro-services
- Certificate and Secret Management (CSM) related Micro-services
- MongoDB for storing packages and OS images.

Since we expect the infra-global-controller to be reachable from the Internet, we should be secured using

- Istio and Envoy (for internal communication as well as for external communication)
- Store Citadel private keys using CSM.
- Store secrets using SMS of CSM.

Infra-local-controller:

The "infra-local-controller" runs on the bootstrap machine in each location. The Bootstrap is the one which installs the required software in compute nodes used for future workloads. For example, say a location has 10 servers. 1 server can be used as the bootstrap machine and all other 9 servers can be used as compute nodes for running workloads. The Bootstrap machine not only installs all required software in the compute nodes, but is also expected to patch and update compute nodes with newer patched versions of the software.

As you see above in the picture, the bootstrap machine itself is based on K8s. Note that this K8s is different from the K8s that gets installed in compute nodes. That is, these are two different K8s clusters. In case of the bootstrap machine, it itself is a complete K8s cluster with one node that has both master and minion software combined. All the components of the infra-local-controller (such as Flux, Cluster API, Metal3 and Ironic) are containers.

Since we expect infra-local-controller is reachable from outside we expect it to be secured using

- Istio and Envoy (for internal communication as well as for external communication)

Infra-local-controller is expected to be brought up in two ways:

- As a USB bootable disk: One should be able to get any bare-metal server machine, insert USB and restart the server. This means that the USB bootable disk shall have basic Linux, K8s and all containers coming up without any user actions. It must also have packages and OS images that are required to provision actual compute nodes. As in above example, these binary, OS and packages are installed on 9 compute nodes.
- As individual entities: As developers, one shall be able to use any machine without inserting a USB disk. In this case, the developer can choose a machine as a bootstrap machine, install Linux OS, Install K8s using kubeadm and then bring up Flux, Cluster API, Metal3 and Ironic.
- As a KVM/QEMU Virtual machine image: One shall be able to use any VM as a bootstrap machine using this image.

Note that the infra-local-controller can be run without the infra-global-controller. In the interim release, we expect that only the infra-local-controller is supported. It is the goal that any operations done in the interim release on infra-local-controller manually are automated by infra-global-controller. And hence the interface provided by infra-local-controller is flexible enough to support both manual actions as well as automated actions.

As indicated above, infra-local-controller will bring up K8s clusters on the compute nodes used for workloads. Bringing up a workload K8S cluster normally requires the following steps

1. Bring up a Linux operating system.
2. Provision the software with the right configuration
3. Bring up basic K8s components (such as kubelet, containerd, kubectl, kubeadm etc..)
4. Bring up additional components

Step 1 and 2 are performed by Metal3 and Ironic. Step 3 is performed by Cluster API and Step 4 is done by Flux.

Metal3 Bare Metal Operator & Ironic

The Bare Metal Operator provides provisioning of compute nodes (either bare metal or VM) by using the K8s API. The Bare Metal Operator defines a CRD BareMetalHost object representing a physical server; it represents several hardware inventories. Ironic is responsible for provisioning the physical servers, and the Bare Metal Operator is for responsible for wrapping the Ironic and represents them as CRD object.

Cluster API (CAPI)

The job of CAPI is provision the bare metal infrastructure with Metal3 and bootstrap K8s with kubeadm. CAPI provides CRs to accomplish the following:

- To upload site-specific information - compute nodes and their roles
- To instantiate the binary package installation.
- To get hold of application K8s kubeconfig file.
- Get status of the installation
- To upload a Linux Operating system that are needed in compute nodes.
- When a new compute node is added, once the administrator adds the new compute node in the site list, it shall take care of installing the packages.

Flux

Flux implements GitOps workflows to install K8s-based packages after K8s is bootstrapped:

- Upload binary images that are used to install the packages in compute nodes.
- Get status of installation of all packages as prescribed before.
- If a new binary package version is uploaded, it shall take care of figuring out the compute nodes that require this new version and update that compute node with the new version.

Infra-local-controller

Since compute nodes may not have Internet connectivity

- The infra-local-controller also acts as a local Docker Hub repository and ensures that all K8s container packages (that need to be installed on the application-K8s) are served locally here.
- The infra-local-controller also configures the container runtime to access packages from this local repository.

The infra-local-controller is expected to store any private key and secret information in CSM.

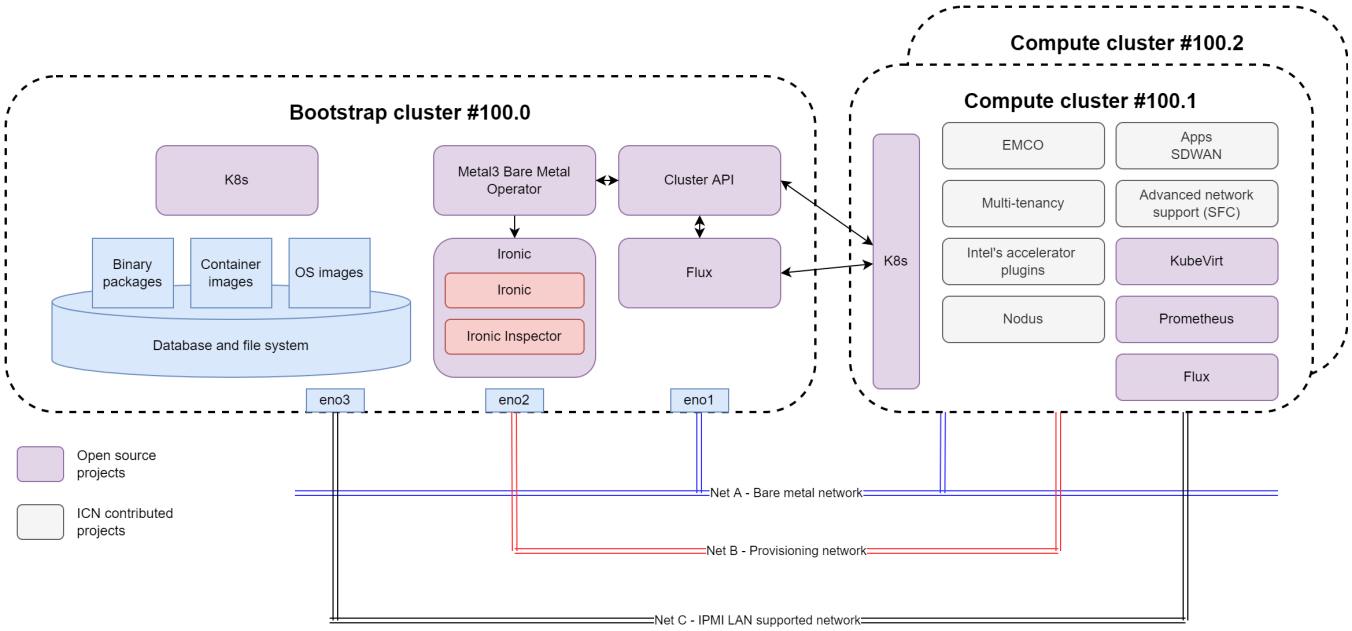
- SSH passwords used to authenticate with the compute nodes is expected to be stored in SMS of CSM
- kubeconfig used to authenticate with application-K8s.

Software Platform Architecture

Local Controller: kubeadm, Flux, Cluster API, Metal3, Bare Metal Operator, Ironic, EMCO

Global Controller: kubeadm, KUD, K8s Provisioning Manager, Binary Provisioning Manager, CSM

R6 Release cover only Infra local controller:



Metal3

One of the major challenges to cloud admin managing multiple clusters in different edge location is coordinate control plane of each cluster configuration remotely, managing patches and updates/upgrades across multiple machines. In ICN family stack, Bare Metal Operator from Metal3 project is used as bare metal provider. It is used as a machine actuator that uses Ironic to provide K8s API to manage the physical servers that also run K8s clusters on bare-metal host.

Cluster API & Flux

Cluster API is a project that uses infrastructure and bootstrap providers (Metal3 and kubeadm in this case) to bring up a K8s deployment and some add-ons on a provisioned machine. Flux uses a GitOps workflow to bring up additional add-ons in a K8s deployment. One of the K8s clusters with high availability, which is provisioned and configured by Cluster API and Flux, will be used to deploy EMCO on K8s. ICN family uses Edge Multi-Cluster Orchestration for service orchestration. EMCO provides a set of helm chart to be used to run the workloads on a multi-cluster.

EMCO Block and Modules:

EMCO will be the Service Orchestration Engine in ICN family and is responsible for the VNF life cycle management, tenant management and tenant resource quota allocation and managing Resource Orchestration Engine (ROE) to schedule VNF workloads with multi-site scheduler awareness and Hardware Platform Abstraction (HPA). It can be used to deploy the K8s App components (as shown in fig. II), NFV Specific components and NFVi SDN controller in the edge cluster. Required an Akraino dashboard that sits on the top of EMCO to deploy the VNFs.

K8s Block and Modules:

K8s will be the Resource Orchestration Engine in ICN family to manage Network, Storage and Compute resource for the VNF application. ICN family will be using containerd as a de-facto container runtime. Each release supports different container runtimes that are focused on use cases.

K8s module is divided into 3 groups - K8s App components, NFV specific components and NFVi SDN controller components, all these components will be installed using EMCO

K8s App components: This block has K8s storage plugins, container runtime, OVN for networking, Service proxy, and responsible application management

NFV Specific components: This block is responsible for K8s compute management to support both software and hardware acceleration (including network acceleration) with CPU pinning and Device plugins such as SRIOV

SDN Controller components: This block is responsible for managing SDN controller and to provide additional features such as Service Function chaining (SFC) and Network Route manager.

Modules Design & Architecture:

Metal3:

ICN uses Metal3 project for provisioning server in the edge locations, ICN project uses Redfish with virtual media (preferred) or the IPMI protocol to identify the servers in the edge locations, and use Ironic & Ironic - Inspector to provision the OS in the edge location. For R6 release, ICN project provision Ubuntu 20.04 in each server, and uses the distinguished provisioning and bare-metal networks for inspection and Redfish/IPMI provisioning.

ICN project injects the user data in each server regarding network configuration, grub update to enable IOMMU, remote command execution using ssh and maintain a common secure mechanism for all provisioning the servers. Each local controller maintains IP address management for that edge location. For more information refer - Metal3 Bare Metal Operator in ICN stack

Cluster API:

ICN uses the Cluster API to provision the infrastructure and bootstrap K8s clusters.

Flux:

ICN uses the Flux deploy additional K8s packages after the cluster is bootstrapped.

EMCO:

EMCO is used as Service orchestration in ICN BP. EMCO is installed as a plugin in any cluster provisioned with Cluster API and Flux. EMCO installed Composite vFW application to install in any edge location.

SDEWAN:

SDEWAN CNF module is worked as a software-defined router located in each edge location and central hub K8s cluster to manage central-edge and edge-edge communication. It's functionality is realized via CNF (Containerized Network Function) and deployed by K8s, it is based on OpenWRT (an open-source project based on Linux, and used on embedded devices to route network traffic) and leverages Linux kernel functionality for packet processing to support network functionalities such as multiple wan link support (mwan3), firewall/SNAT/DNAT (fw3) , IPSec (strongswan) etc. It exposes Restful APIs for configuration, detail information can be found at: SDEWAN CNF

SDEWAN Configure Agent (also named SDEWAN Controller) module is worked as K8s controller located in each edge location and central hub K8s cluster to support configuration of SDEWAN CNF functionalities (e.g. mwan3, firwall, SNAT, DNAT, IPSec etc.) and monitor SDEWAN CNF status. It exposes CRDs to support configuration via K8s API server for unified authentication and authorization, detail information can be found at: SDEWAN CRD Controller

Cloud Storage:

Cloud Storage (Cloud Storage Design) act as storage service and plugins, currently can divide into two parts:

1. Storage Service for Local controller: which used by BPA Rest Agent to provide storage service for image objects with binary, container and operating system. There are 2 solutions, MinIO and GridFS, with the consideration of Cloud native and Data reliability, we propose to use MinIO, which is CNCF project for object storage and compatible with Amazon S3 API, and provide language plugins for client application, it is also easy to deploy in K8s and flexible scale-out. MinIO also provide storage service for HTTP Server. Since MinIO need export volume in bootstrap, local-storage is a simple solution but lack of reliability for the data safety, we will switch to reliability volume provided by Ceph CSI RBD in next release.
2. Optane Persistent Memory plugin in KUD, which can provide LVM and direct volumes on Optane PM namespaces, since the Optane PM has high performance and low latency compared with normal SSD storage device, it can be used as cache, metadata volume or other high throughput and low latency scenarios.

Software components:

Please refer to list of software components in the ICN R6 Release Notes

Hardware Management

Hostname	CPU Model	Memory	Storage	1GbE: NIC#, VLAN, (Connected extreme 480 switch)	10GbE: NIC# VLAN, Network (Connected with IZ1 switch)
Jump	2xE5-2699	64GB	3TB (Sata) 180 (SSD)	IF0: VLAN 110 (DMZ) IF1: VLAN 111 (Admin)	IF2: VLAN 112 (Private) **VLAN 114** (Management) IF3: VLAN 113 (Storage) **VLAN 1115** (Public)

node1	2xE5-2699	64GB	3TB (Sata) 180 (SSD)	IF0: VLAN 110 (DMZ) IF1: VLAN 111 (Admin)	IF2: VLAN 112 (Private) **VLAN 114** (Management) IF3: VLAN 113 (Storage) **VLAN 1115** (Public)
node2	2xE5-2699	64GB	3TB (Sata) 180 (SSD)	IF0: VLAN 110 (DMZ) IF1: VLAN 111 (Admin)	IF2: VLAN 112 (Private) **VLAN 114** (Management) IF3: VLAN 113 (Storage) **VLAN 1115** (Public)
node3	2xE5-2699	64GB	3TB (Sata) 180 (SSD)	IF0: VLAN 110 (DMZ) IF1: VLAN 111 (Admin)	IF2: VLAN 112 (Private) **VLAN 114** (Management) IF3: VLAN 113 (Storage) **VLAN 1115** (Public)
node4	2xE5-2699	64GB	3TB (Sata) 180 (SSD)	IF0: VLAN 110 (DMZ) IF1: VLAN 111 (Admin)	IF2: VLAN 112 (Private) **VLAN 114** (Management) IF3: VLAN 113 (Storage) **VLAN 1115** (Public)
node5	2xE5-2699	64GB	3TB (Sata) 180 (SSD)	IF0: VLAN 110 (DMZ) IF1: VLAN 111 (Admin)	IF2: VLAN 112 (Private) **VLAN 114** (Management) IF3: VLAN 113 (Storage) **VLAN 1115** (Public)

Licensing

Refer Software Components list