Infrastructure & Architecture

John Abbott (jabbott@us.ibm.com)
John Webb (john.webb@us.ibm.com)

Objectives

After completing this lecture, you will be able to:

- Explain the IBM Cloud Private included components
- Understand add-on components provided by IBM Cloud Private and their value
- Have a foundation for diving deeper into specific areas of the solution

Agenda

Installation Scenarios

IBM Cloud Private – logical components

Network architecture

Storage considerations

Security

Scaling and sizing

Configuration topologies

Simple

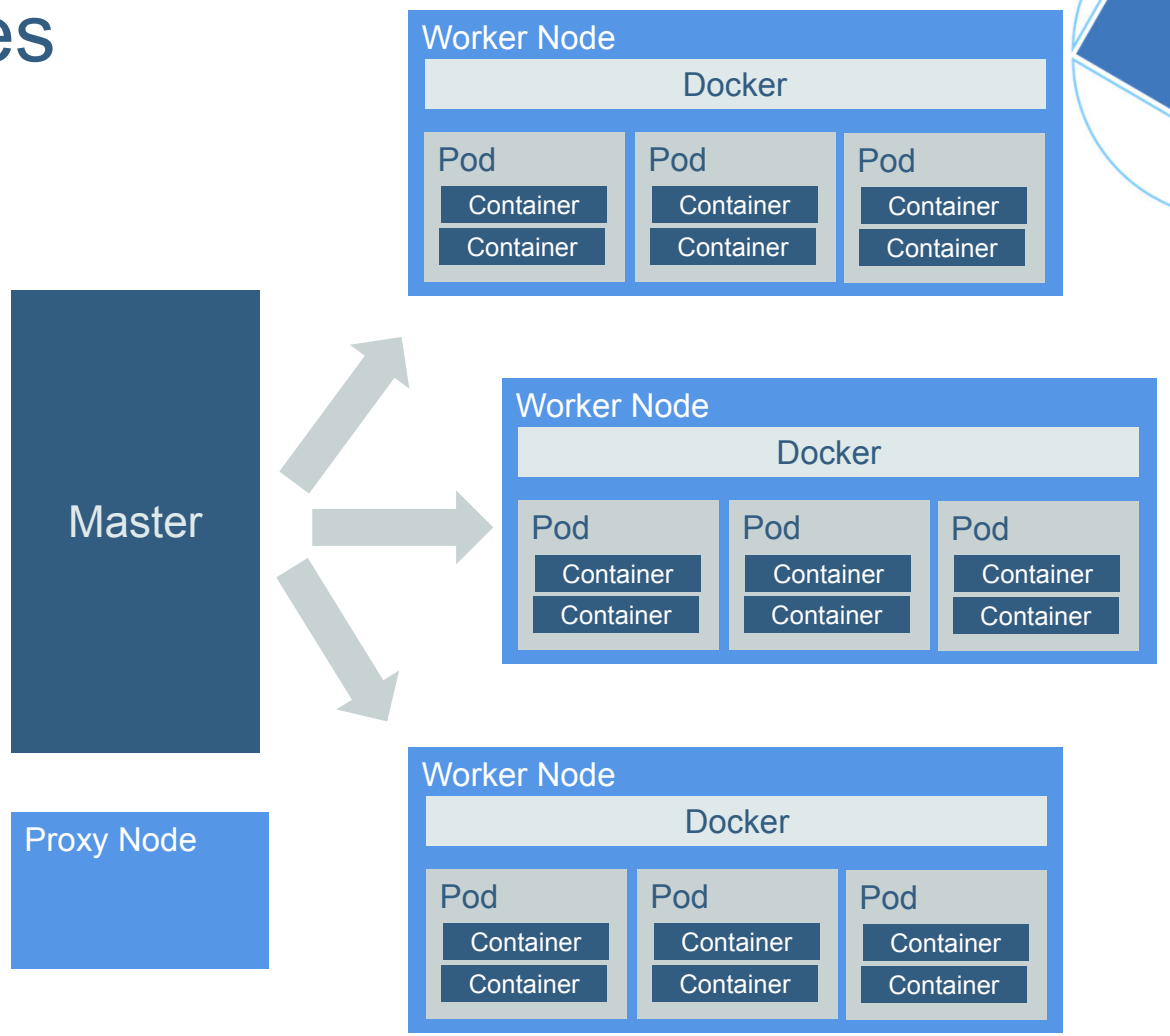
- Single machine install (master is a worker)
- Great for testing and learning about the platform

Standard

- Single master (single master, 3 workers, 1 proxy)
- Great for non-production testing environment

High Availability

- Multiple masters (3 masters, 3+ workers, 3 proxy)
- Production installation



ICP – types of nodes

Boot node: A boot or bootstrap node is used for running installation, configuration, node scaling, and cluster updates. Only one boot node is required for any cluster. You can use a single node for both master and boot.

Master node: A master node provides management services and controls the worker nodes in a cluster. Master nodes host processes that are responsible for resource allocation, state maintenance, scheduling, and monitoring. Multiple master nodes are in a high availability (HA) environment to allow for failover if the leading master host fails. Host that can act as the master are called master candidates.

Worker node: A worker node is a node that provides a containerized environment for running tasks. As demands increase, more worker nodes can easily be added to your cluster to improve performance and efficiency. A cluster can contain any number of worker nodes, but a minimum of one worker node is required.

Proxy Node: A proxy node is a node that transmits external request to the services created inside your cluster. Multiple proxy nodes are deployed in a high availability (HA) environment to allow for failover if the leading proxy host fails. While you can use a single node as both master and proxy, it is best to use dedicated proxy nodes to reduce the load on the master node. A cluster must contain at least one proxy node if load balancing is required inside the cluster.

Installation overview

Container-based installer:

- Download the installation container for CE or EE and execute the install
- The installer pulls down additional containers from Docker Hub for CE, local repo for EE

Supported for RHEL and Ubuntu on X, POWER and Z (workers)

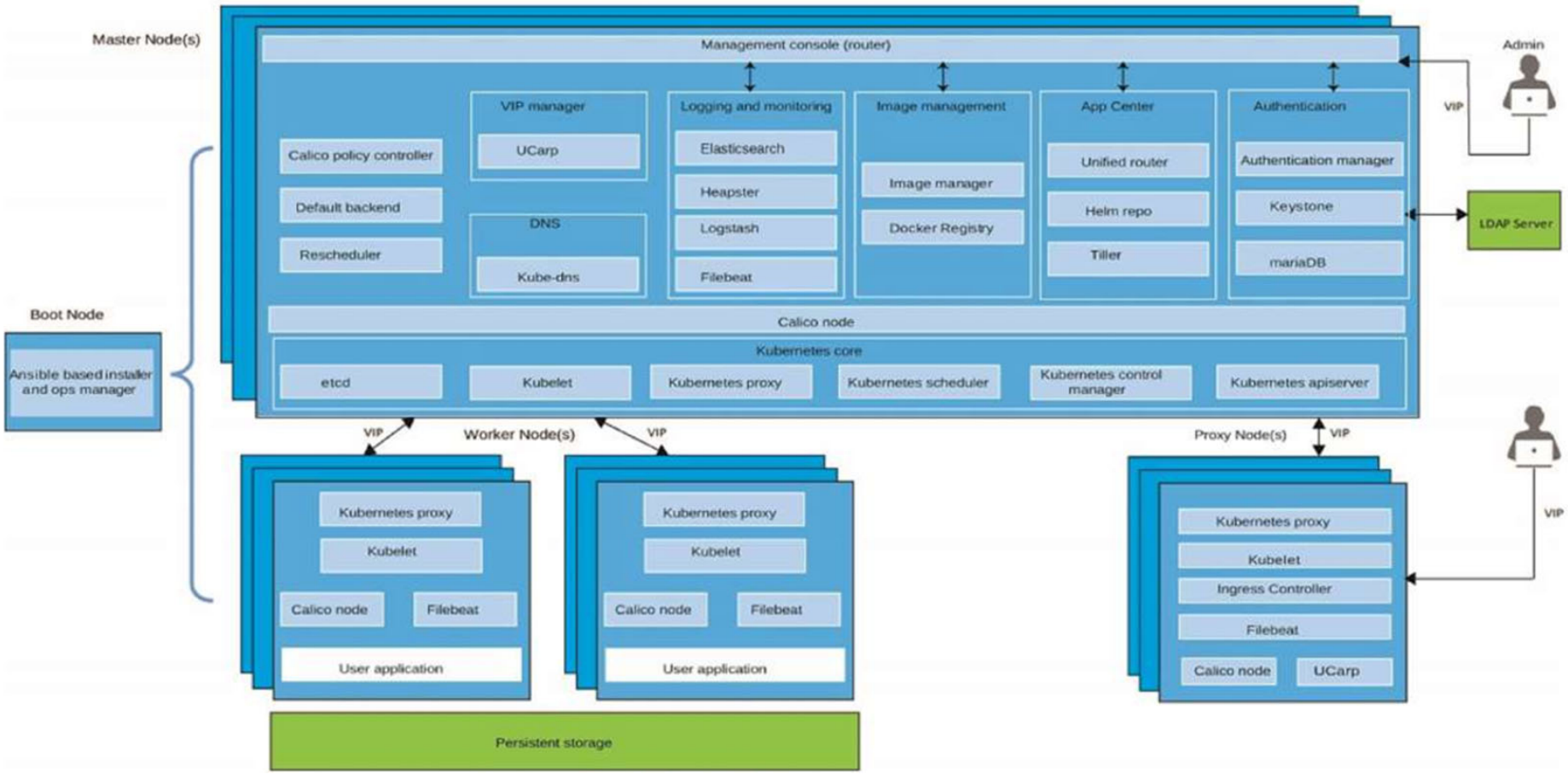
Basic installation steps:

- 1. Configure OS
- 2. Modify installation configuration files and run the installer

Overall installation should take < 4 hours depending on scenario

- (90% System Config, 10% Installation)

IBM Cloud Private – architecture components



K8s master components

Master components provide the cluster's control plane and global decisions about the cluster and detecting and responding to cluster events

Etcd: A strong, consistent, and highly-available key value store which Kubernetes uses for persistent storage of all of its API objects.

Kubelet: The primary "node agent" that runs on each node.

K8s Proxy: The Kubernetes network proxy runs on each node.

K8s Scheduler: A policy-rich, topology-aware, workload-specific function that significantly impacts availability, performance, and capacity.

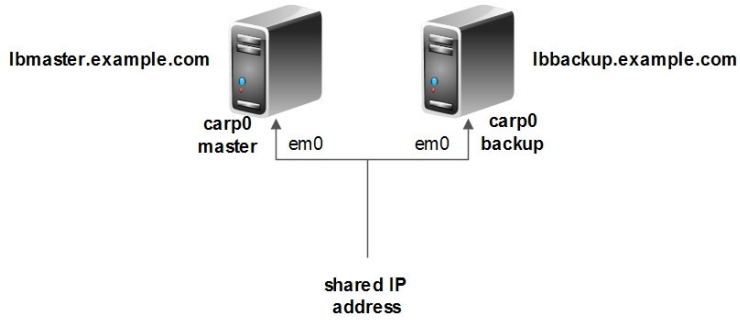
K8s Control Manager: A daemon that embeds the core control loops shipped with K8s. In K8s, a controller is a control loop that watches the shared state of the cluster through the apiserver and makes changes attempting to move the current state towards the desired state.

K8s apiserver: Validates and configures data for the api objects which include pods, services, replication controllers, and others. The API Server services REST operations and provides the frontend to the cluster's shared state through which all other components interact.

Network services components

DNS: (kube-dns, Cluster DNS) K8s DNS schedules a DNS pod and service on the cluster, and configures the kubelets to tell individual containers to use the DNS service's IP to resolve DNS names. Every service defined in the cluster (including the DNS server itself) is assigned a DNS name. By default, a client pod's DNS search list will include the pod's own namespace and the cluster's default domain.

VIP and UCarp: UCarp allows a couple of hosts to share common virtual IP (or floating IP) addresses in order to provide automatic failover.



Calico

A new approach to virtual networking and network security for containers, VMs, and bare metal services, that provides a rich set of security enforcement capabilities running on top of a highly scalable and efficient virtual network

- The calico/node Docker container runs on the Kubernetes master and each Kubernetes node in the cluster
- The calico-cni plug-in integrates directly with the Kubernetes kubelet process on each node to discover which pods have been created, and adds them to Calico networking
- The calico/kube-policy-controller container runs as a pod on top of Kubernetes and implements the NetworkPolicy API

PROJECT CALICO

Ingress resources

Ingress: Typically, services and pods have IPs only routable by the cluster network. All traffic that ends up at an edge router is either dropped or forwarded elsewhere.

- An Ingress is a collection of rules that allow inbound connections to reach the cluster services.
- It can be configured to give services externally-reachable URLs, load balance traffic, terminate SSL, offer name based virtual hosting etc.
- Users request ingress by POSTing the Ingress resource to the API server

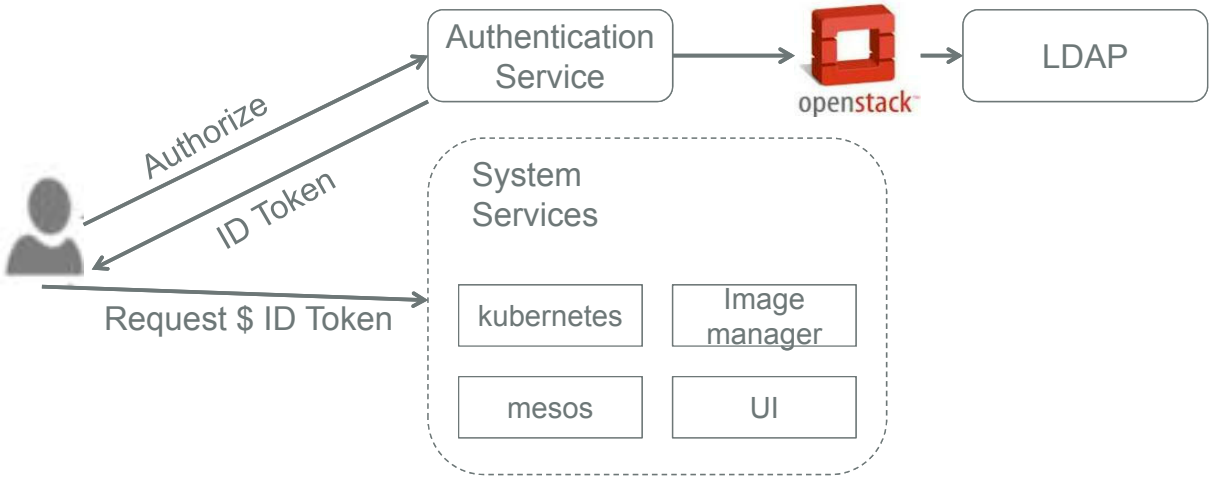
Ingress Controller: Responsible for fulfilling the Ingress, usually with a load balancer, though it may also configure your edge router or additional frontends to help handle the traffic in an HA manner.

Authentication components

Authentication Manager: Provides an HTTP API for managing users. Protocols are implemented in a RESTful manner. Keystone is used for authentication. Pass-through is used for external LDAP integration.

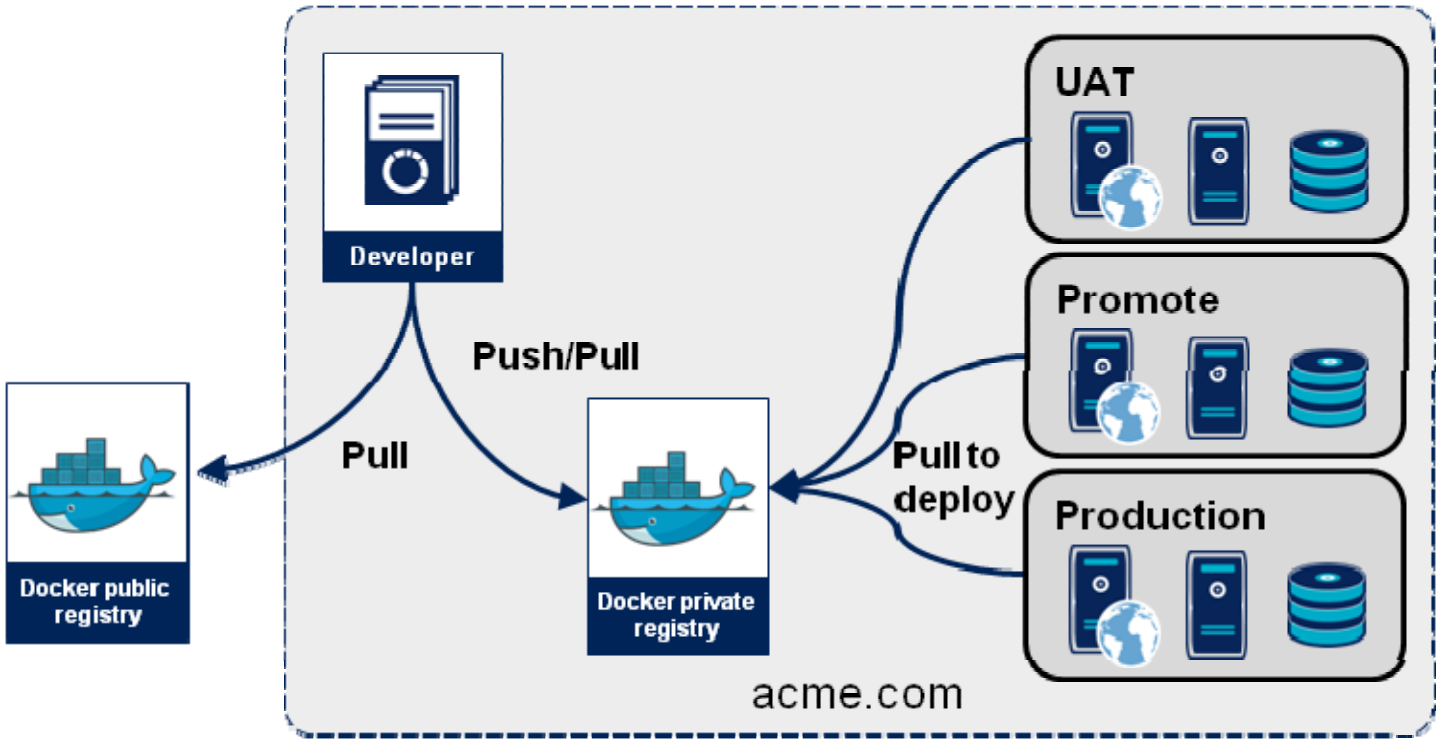
Keystone: The OpenStack provided identity service currently supporting token-based authN and user-service authorization.

MariaDB: An open source relational database made by the original developers of MySQL. In this case it is used to back-end Keystone.



Images and Registries

- You create a Docker image and push it to a registry before referring to it in a Kubernetes pod
- There will likely be many registries used in your deployment

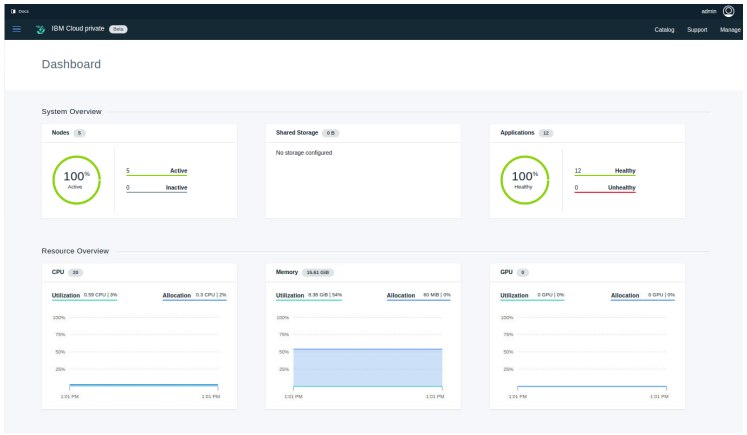


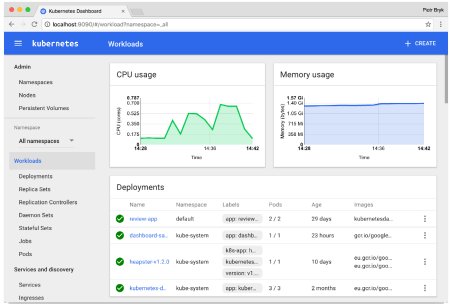
User Interfaces

Cluster Management Console: (ICP component) Use to manage, monitor, and troubleshoot your applications and cluster from a single, centralized, and secure management console.

K8s Web UI: Can use to deploy containerized applications to a Kubernetes cluster, troubleshoot your containerized application, and manage the cluster itself along with its attendant resources.

kubectl: A command-line interface for running commands against Kubernetes clusters.






```
[root@ip-172-31-56-194 kubernetes]# export PATH=/home/ec2-user/kubernetes/platforms/linux/amd64:$PATH
[root@ip-172-31-56-194 kubernetes]# kubectl get nodes
NAME                                         LABELS                                                    STATUS    AGE
ip-172-20-0-138.ec2.internal   kubernetes.io/hostname=ip-172-20-0-138.ec2.internal   Ready     3m
ip-172-20-0-25.ec2.internal    kubernetes.io/hostname=ip-172-20-0-25.ec2.internal    Ready     20m
[root@ip-172-31-56-194 kubernetes]# kubectl run ttnd-nginx --image=nginx
replicationcontroller "ttnd-nginx" created
[root@ip-172-31-56-194 kubernetes]# kubectl get pods
NAME              READY     STATUS    RESTARTS   AGE
ttnd-nginx-wp2y9   0/1       Pending   0          8s
[root@ip-172-31-56-194 kubernetes]# kubectl get pods --namespace=kube-system
NAME                                                    READY     STATUS    RESTARTS   AGE
elasticsearch-logging-v1-mcd2q                          1/1       Running   0          23m
elasticsearch-logging-v1-mnqqm                          1/1       Running   0          23m
fluentd-elasticsearch-ip-172-20-0-138.ec2.internal      1/1       Running   0          3m
fluentd-elasticsearch-ip-172-20-0-25.ec2.internal       1/1       Running   0          20m
heapster-v10-eco91                                      1/1       Running   0          23m
kibana-logging-v1-slamn                                 1/1       Running   0          23m
kube-dns-v9-7pe5g                                       4/4       Running   0          23m
kube-ui-v2-0573n                                        1/1       Running   0          23m
monitoring-influxdb-grafana-v2-mgzvo                    2/2       Running   0          23m
[root@ip-172-31-56-194 kubernetes]#
[root@ip-172-31-56-194 kubernetes]#
[root@ip-172-31-56-194 kubernetes]# kubectl get pods
NAME              READY     STATUS    RESTARTS   AGE
ttnd-nginx-wp2y9   1/1       Running   0          36s
```

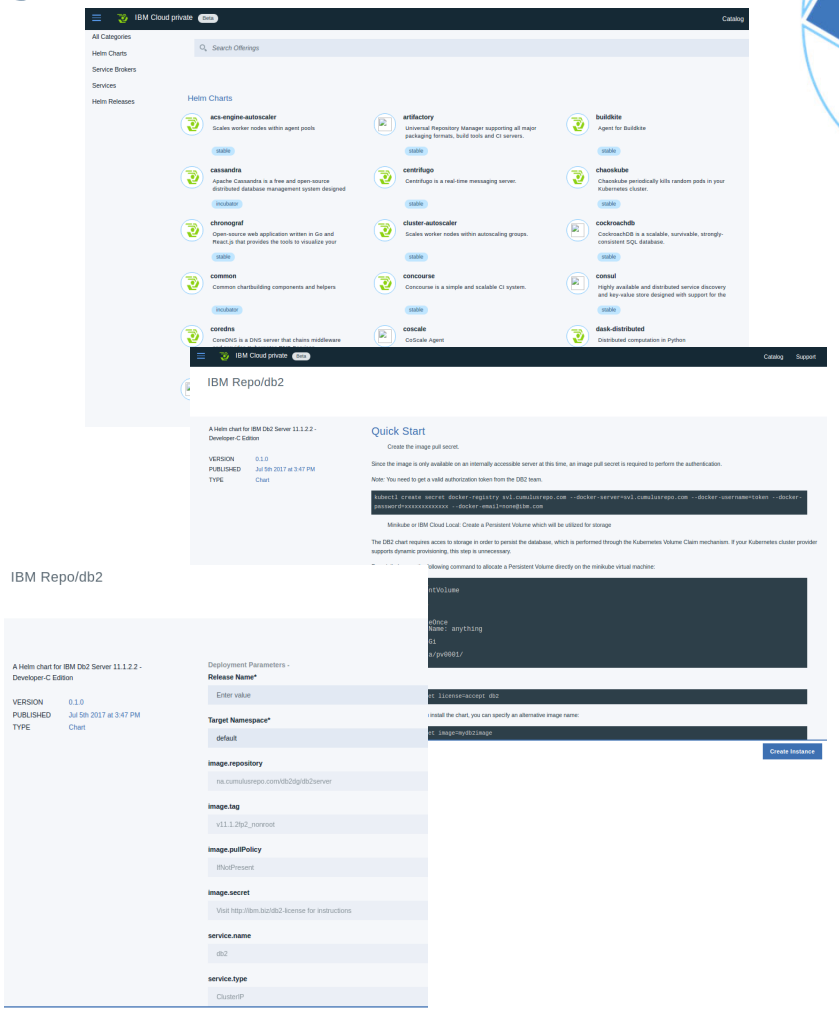
Application Center components

Application center provides a centralized location from which you can browse, and install packages in your cluster.

Helm: A tool for managing Kubernetes charts. Charts are packages of pre-configured Kubernetes resources.

Helm Repository: A Helm chart repository is a location where packaged charts can be stored and shared.

Tiller: Runs inside of the cluster, and manages releases (installations) of your charts.



Logging components

The easiest and most embraced logging method for containerized applications is to write to standard out and standard error

Filebeat: A log data shipper for local files. Filebeat monitors the log directories or specific log files, tails the files, and forwards them either to Elasticsearch and/or Logstash for indexing.

Elasticsearch: An open source full-text search engine based on Lucene. It provides HTTP web interface and schema-free JSON documents.

Logstash: A open source tool for collecting, parsing, and storing logs for future use.

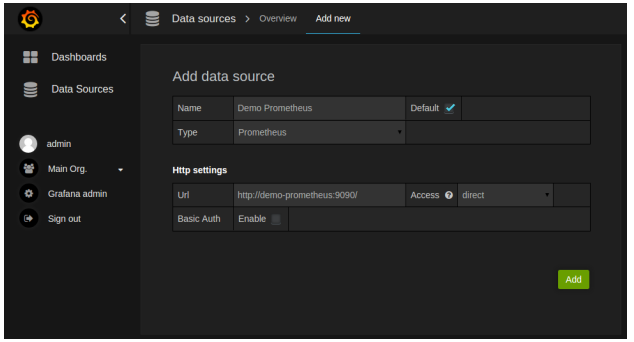
Heapster: The Kubernetes network proxy runs on each node.

Kibana: An open source data visualization plugin for Elasticsearch. Users can create bar, line and scatter plots, or pie charts and maps on top of large volumes of data.

Monitoring: Prometheus and Grafana

Prometheus: An open-source systems monitoring and alerting toolkit originally built at SoundCloud. Since its inception in 2012, many companies and organizations have adopted Prometheus, and the project has a very active developer and user community. It is now a standalone open source project and maintained independently of any company.

Grifana: An open-source, general purpose dashboard and graph composer, which runs as a web application.



Persistent storage components (1 of 2)

Traditionally Containers: stateless, ephemeral in nature

- Storage exists within the container
- The container goes away and so goes the storage

Some applications desire state and thus persistent storage:

- Specific aspects of configuration
- Database (structured and unstructured)
- Application data (website definitions, etc.)

Storage must be universally accessible across the K8s environment

Persistent storage components (2 of 2)

ICP Persistent Storage Support: HostPath, NFS, GlusterFS, vSphereVolume

Note: Reclaim policy and access modes and behaviors can vary

Access Modes:

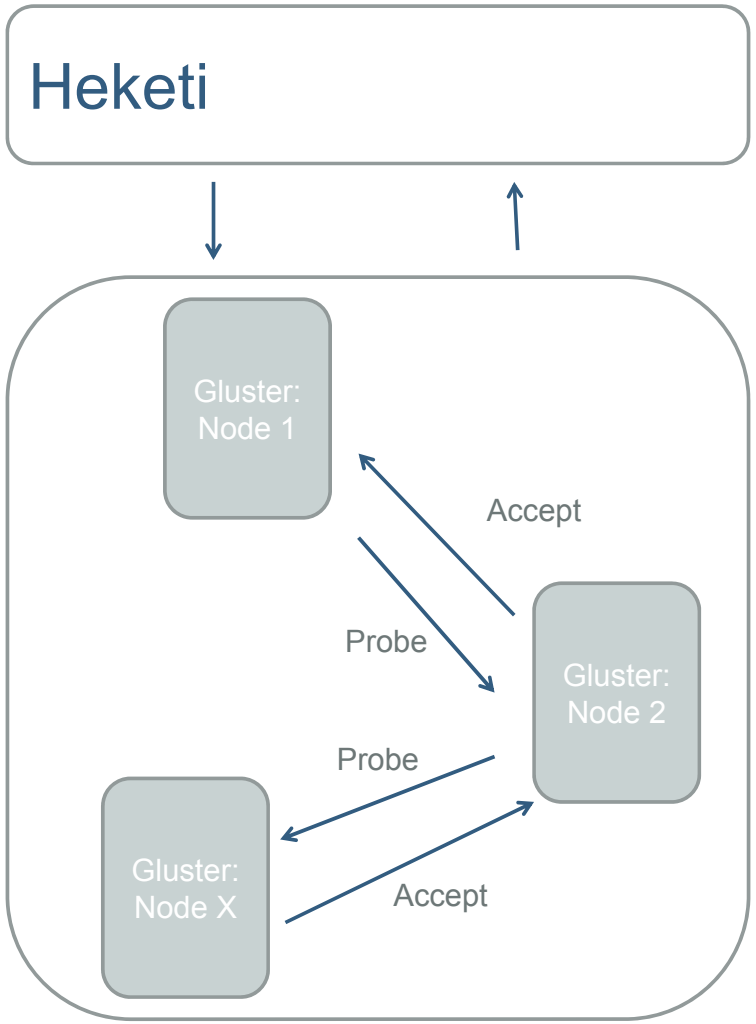
- ReadWriteOnce – the volume can be mounted as read-write by a single node
- ReadOnlyMany – the volume can be mounted read-only by many nodes
- ReadWriteMany – the volume can be mounted as read-write by many nodes

GlusterFS

- Network Attached Storage
- POSIX-Compliant distributed file-system
- Differentiator -> No central metadata server
- Aggregator of storage and metadata
- Flexible scaling for capacity and performance
- NOTE: Red Hat Gluster Storage (formerly Red Hat Storage Server) is built upon the open source GlusterFS at gluster.org

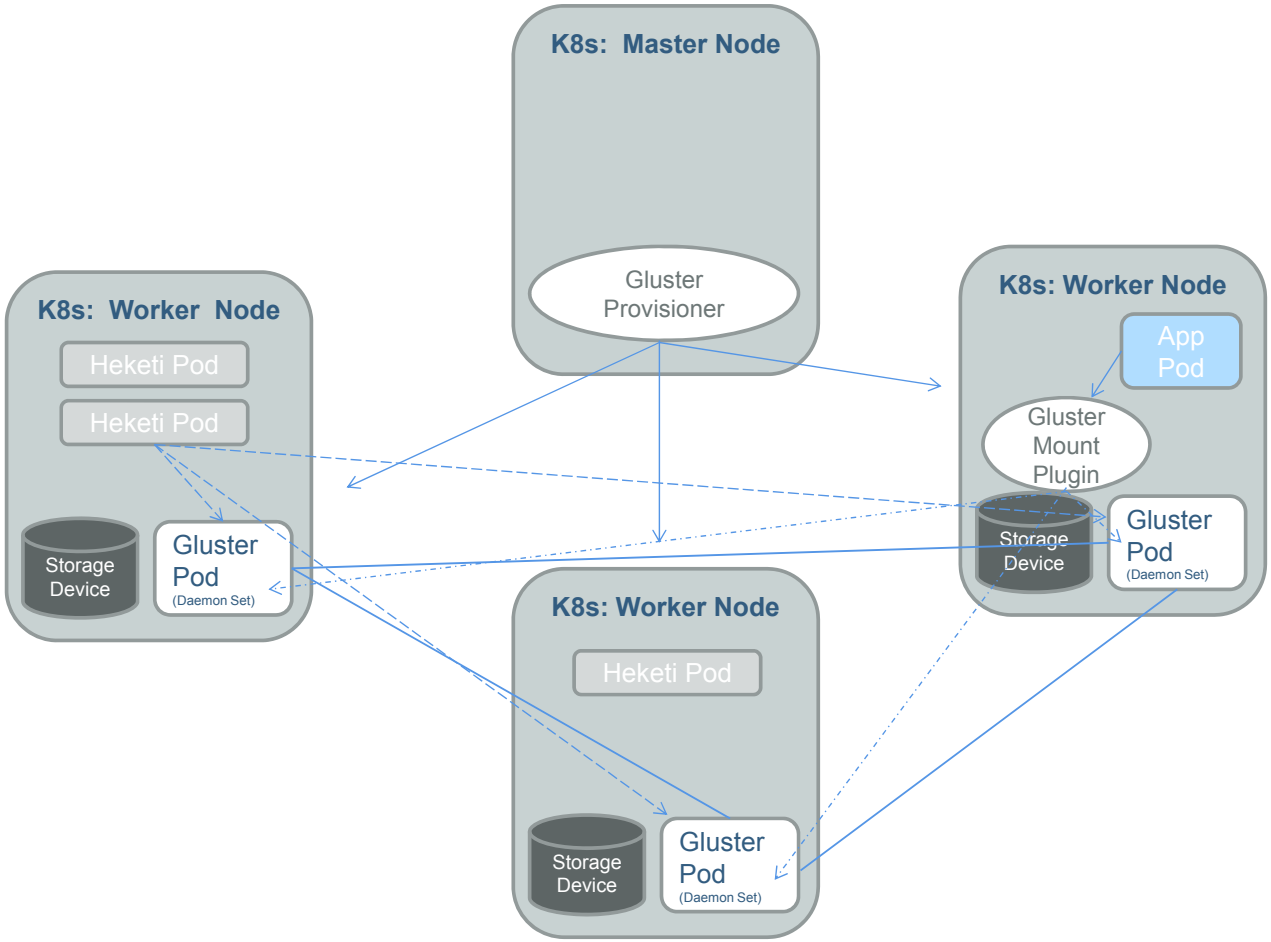
GlusterFS and Heketi

- Restful management interface for the management of GlusterFS volumes
- For Kubernetes, it provides dynamic provisioning of volumes
  - You provide nodes with storage devices
  - Heketi assembles the topology
  - Manages life-cycle of bricks, volumes, etc
  - ICP + K8s Talk to Heketi
- https://github.com/heketi/heketi



GlusterFS - common deployment architecture

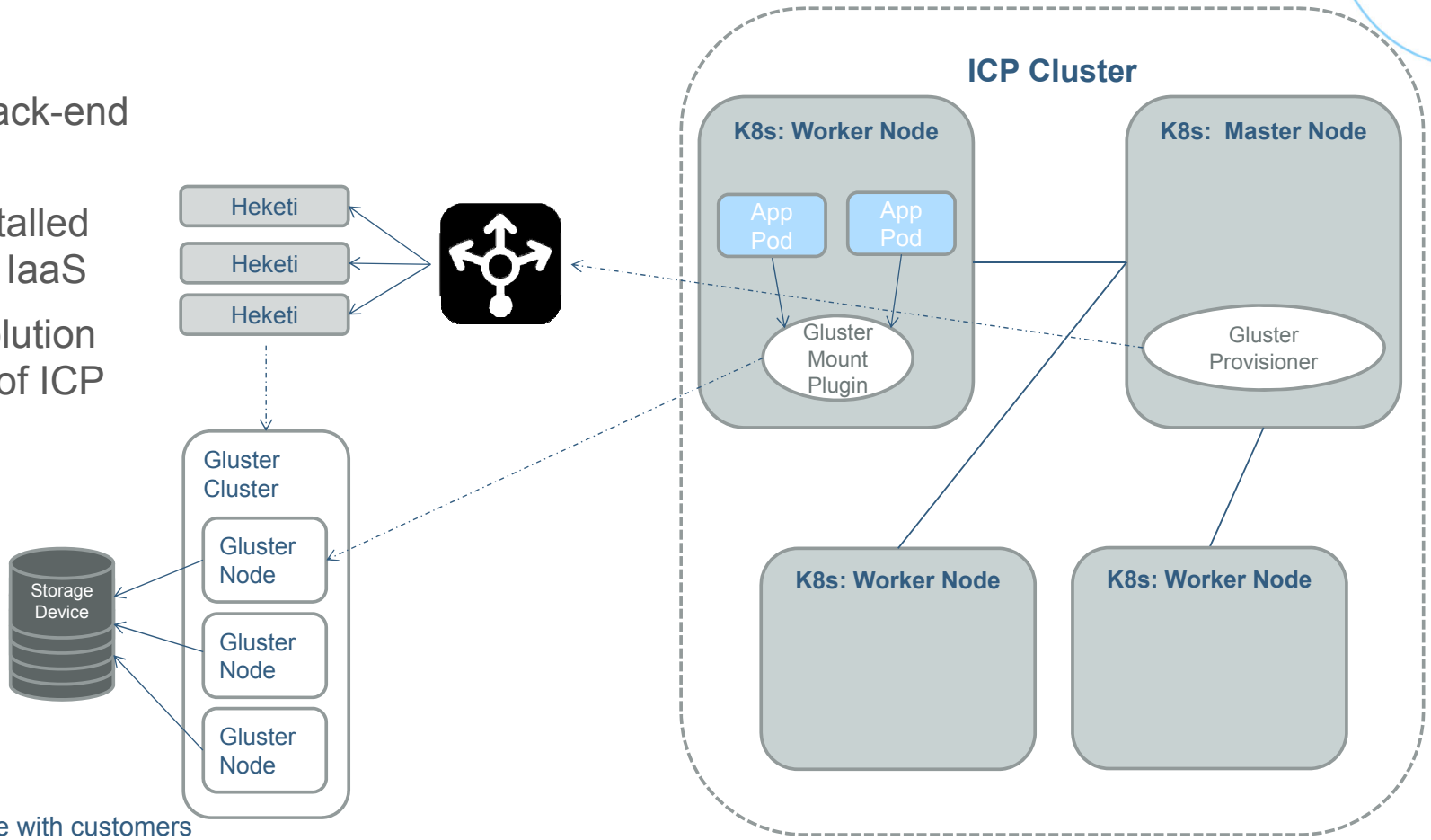
- Pod-ased Components
- Gluster pods Deployed to each K8s node
- Heketi pod or pods deployed to K8s



Enterprise architecture

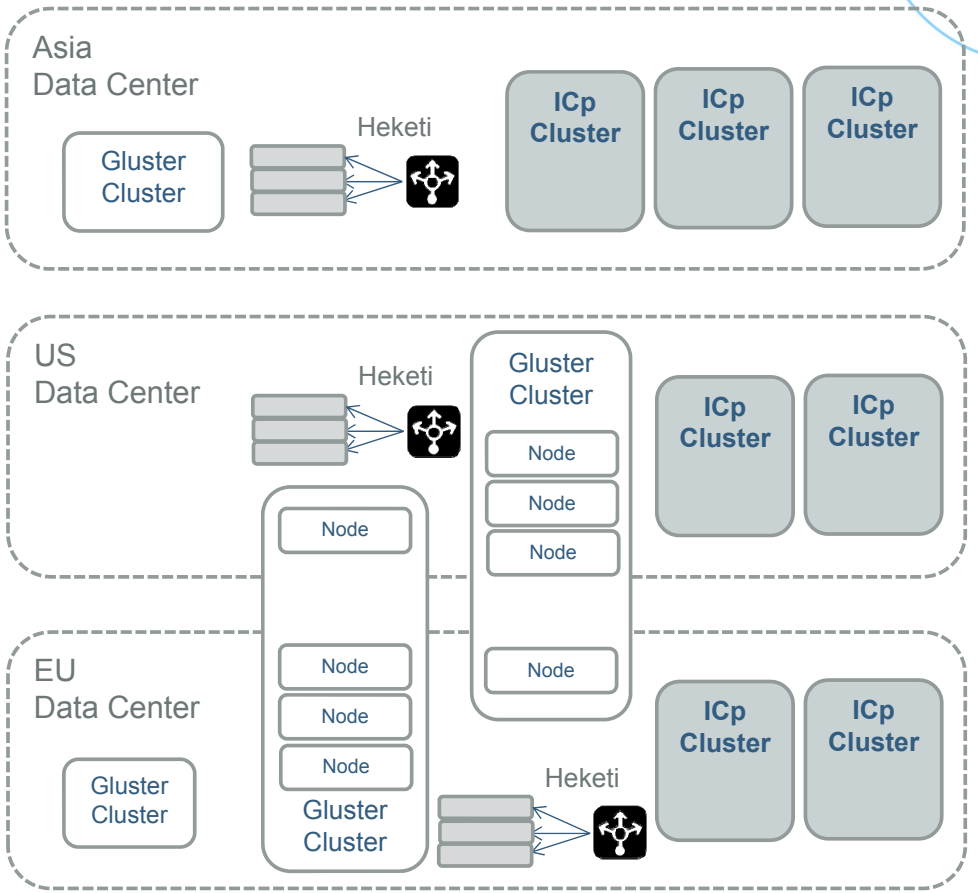
Principle: Remove management and back-end from the platform

- Heketi can be installed as pods or on HA IaaS
- Allows storage solution life-cycle outside of ICP

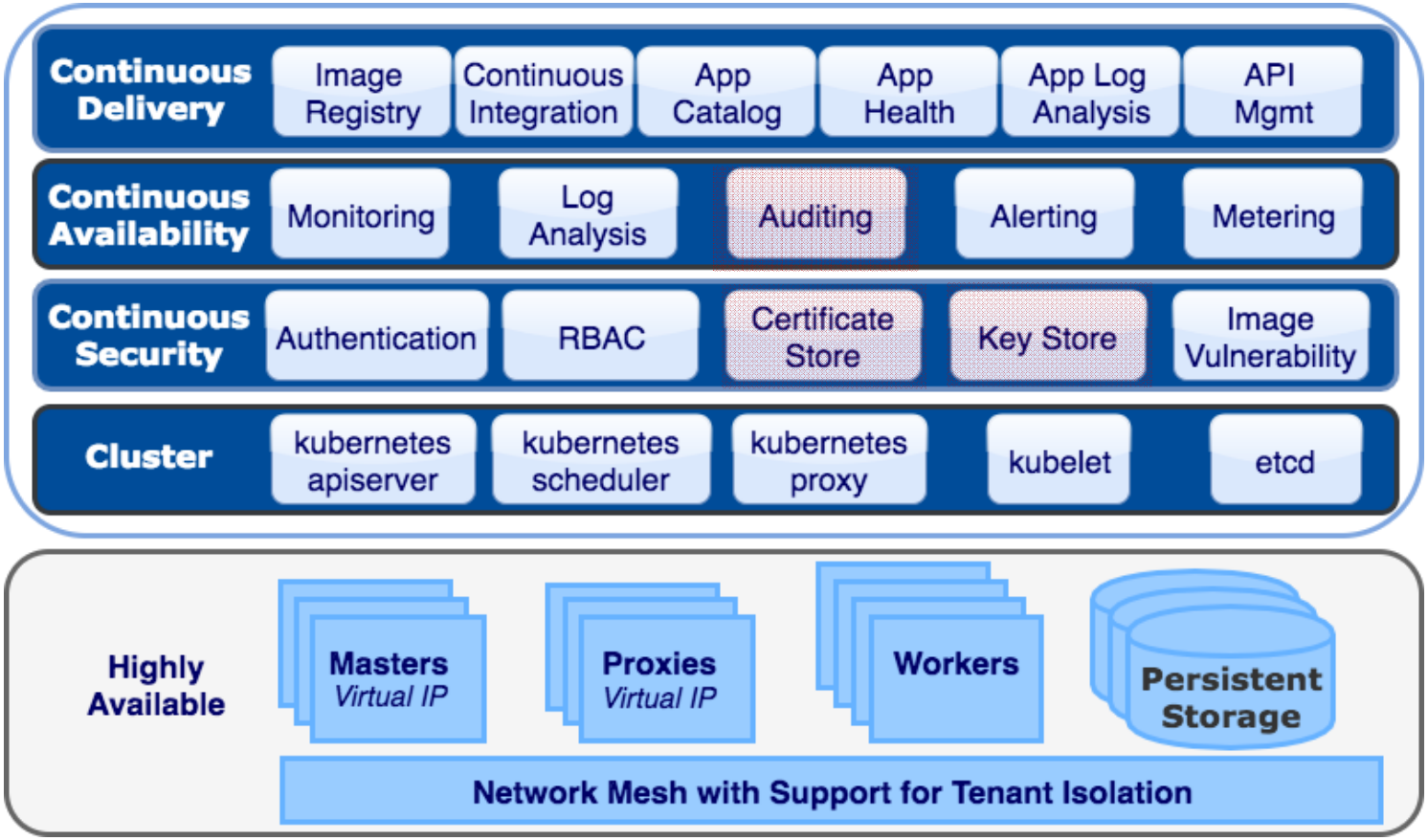


GluserFS architecture extended (example)

- Gluster cluster (trusted storage pool) serving one or more ICP cluster in the same data center
- Geo replication replicates data to remote GlusterFS node (one way)
- One ICP may access one or more GlusterFS trusted storage pools



ICP Functional Architecture



Services marked in red could not be contained for October.

Supported system configurations (October Release)

Looking into RHEL 7.4 support

OS/Browsers: Target to test latest 1-2 stable, supported versions as of 2 months prior to GA

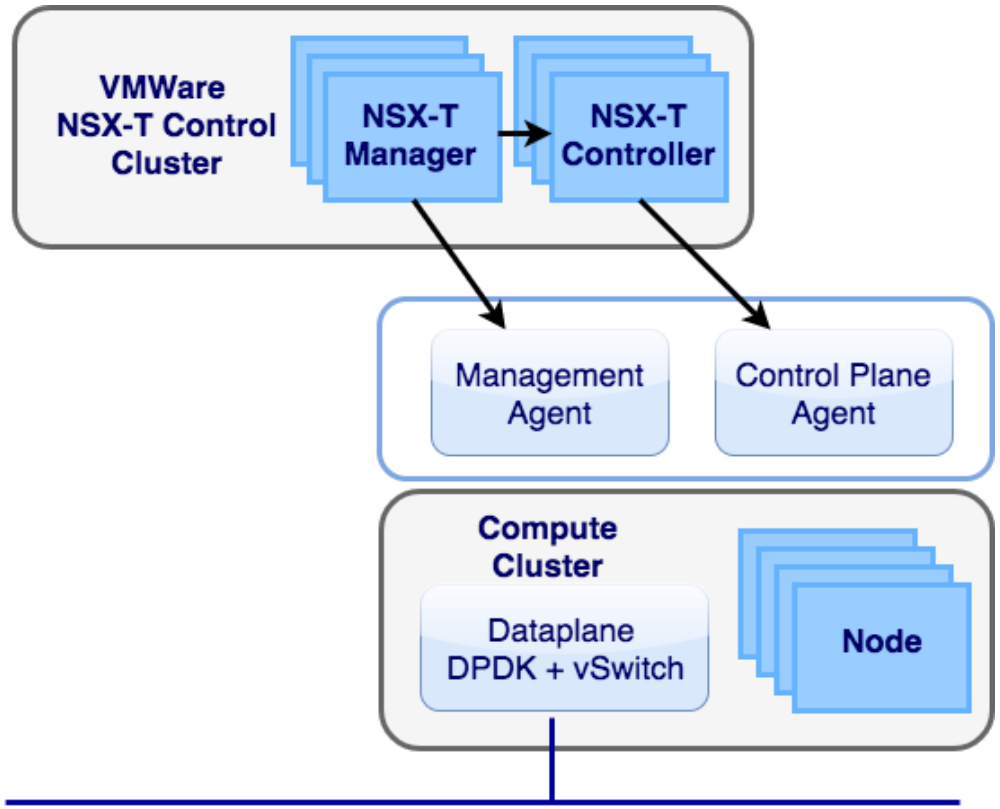
Specs		Support Statement	
OS	x86	RHEL ~~7.1~~, 7.2, 7.3, Ubuntu 16.04 LTS	
	Power	RHEL ~~7.1~~, 7.2, 7.3, Ubuntu 16.04 LTS	
	Z (workers)	RHEL 7.1, 7.2, 7.3, Ubuntu 16.04 LTS; deployed as zVM, zKVM or LPAR	
Browsers	Windows	IE 11, Firefox ~~52.x,~~ 53, ESR 52.x, Chrome 59.0.3071.115	
	Linux	Firefox ~~44, 45, 52~~, 53, ~~ESR 38, ESR45~~, ESR 52.x	
	MacOS	Safari ~~9.0.1,~~ 10.x, Firefox 53, ESR 52.x, Chrome 59.0.3071.115	
Docker	x86	17.03-ce (included in ICp)	Or customer installed: 1.12, 1.13, CE & EE ~~17.03-ce,~~ 17.06
	Power	17.03-ce (included in ICp)	Or customer installed: 1.12, 1.13, CE & EE ~~17.03-ce,~~ 17.06
	Z (workers)	17.03-ce (included in ICp)	Or customer installed: 1.12
Storage		Built-in storage options: GlusterFS + Heketi, vSphere vVol External data stores: NFS 4, Gluster FS 3.5.9, Spectrum Scale (via NFS or Hostpath) + all Kubernetes supported storage types	
Networking		Calico (default), NSX-T 2.0 (optional)	

Size & scale (October Release)

Specs		Support Statement
Node specs (minimum)	Master	2 cores @2.4 GHz, ~~4~~ 8 GB RAM, ~~6~~ 40 GB Disk
	Worker	1 core @2.4 GHz, 4 GB RAM, ~~6~~ 40 GB Disk
# of Nodes (PM or VM)	Minimum	1 (single node install supported)
	Maximum	~~60~~ 300 Power nodes, 300 x86 nodes (tested limits, not hard limits)
# of Pods		~~2000~~ 9000 (tested limit)
# of Users		10000 (tested limit, not hard limit)

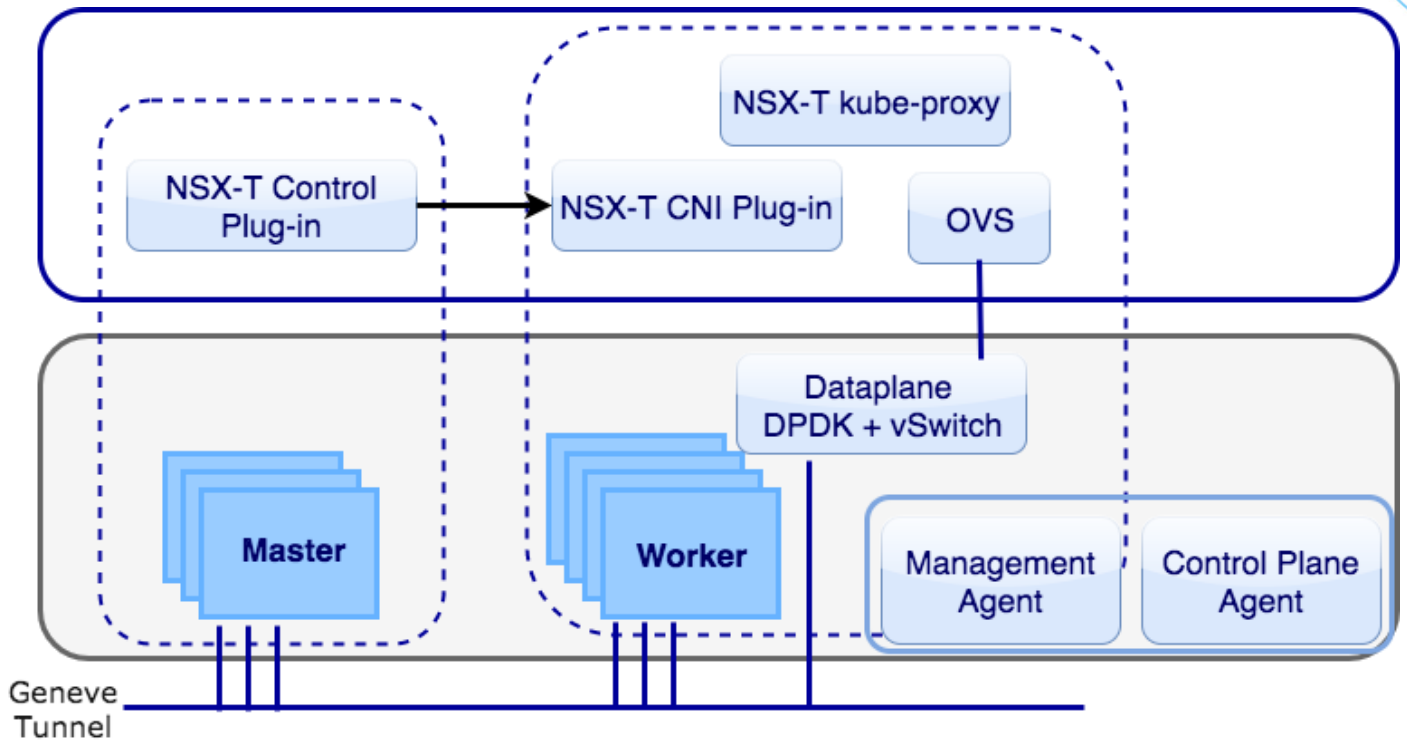
Supporting Material

NSX-T offers a new way to define networking on VMWare environments that offers flat networking between containers and virtual machines



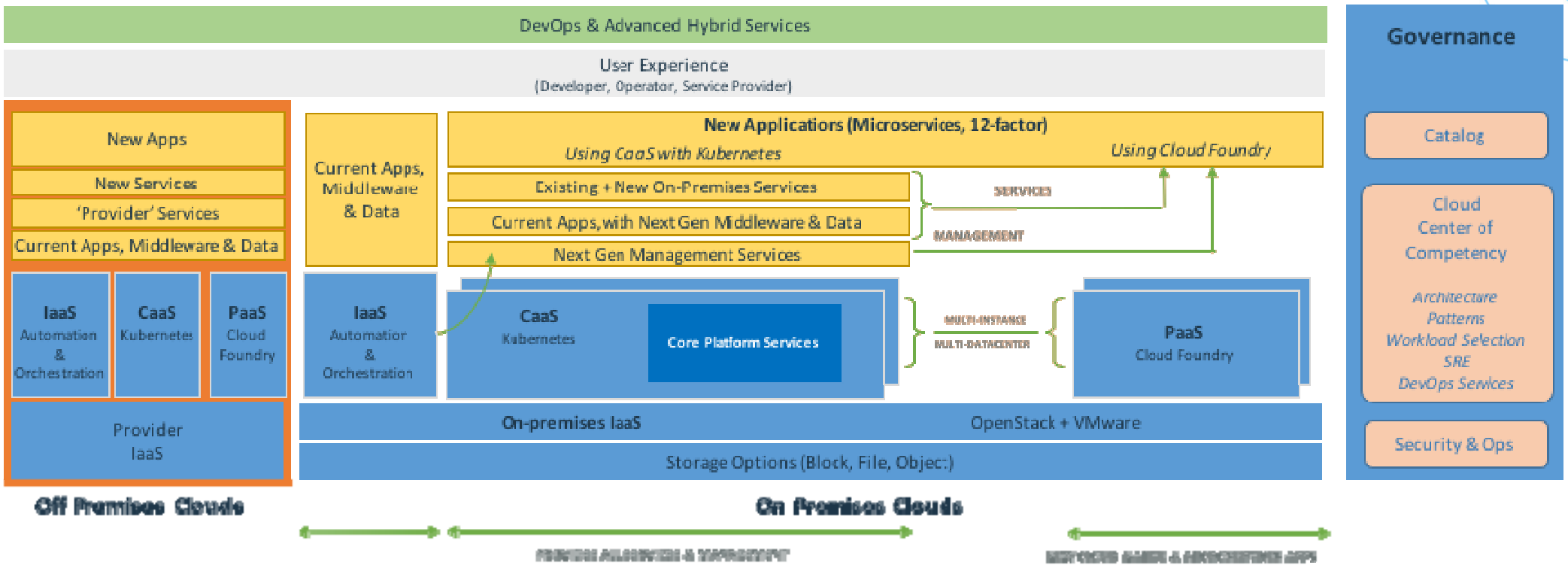
IBM Cloud private will offer NSX-T as the networking mesh between pods on VMWare ESXi 6.5

Calico will continue to be used in OpenStack and lower versions of VMWare (down to ESXi 5.5)



System z Support

- ICp 2.1 will support running worker nodes on z hardware (including z14) running Ubuntu 16.04 LTS and RHEL VMs.
- The management components (master, proxy, boot nodes) are not yet supported on z. These can be run on x86_64 or Power Linux based hardware.
- Does not include the Cloud Foundry option, only the Kubernetes features are supported on z workers
- Does not support zOS, only Linux on z running on zVM, zKVM or z LPAR.



This presentation is intended for an IBM internal audience only.