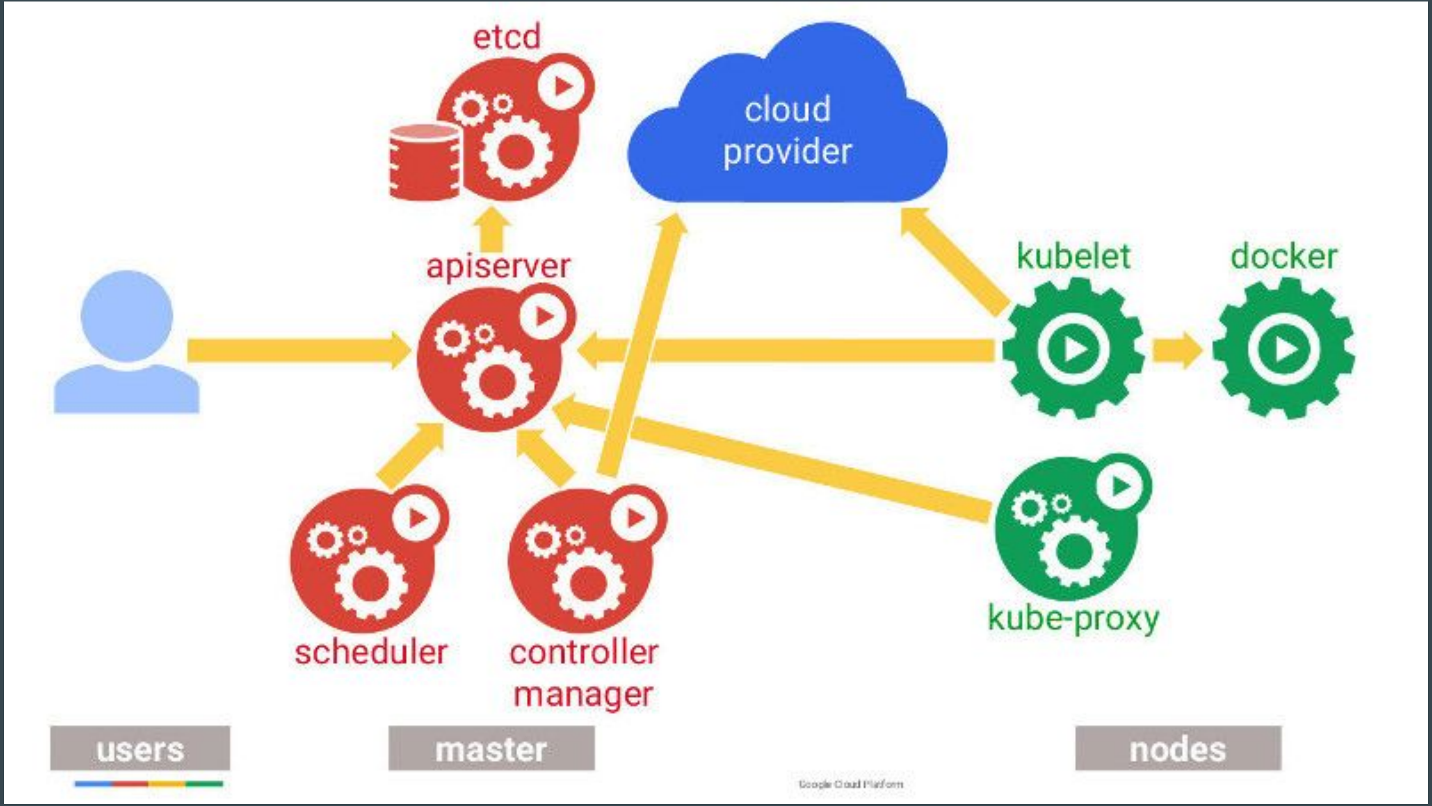
Kubernetes @ NCSA

•••

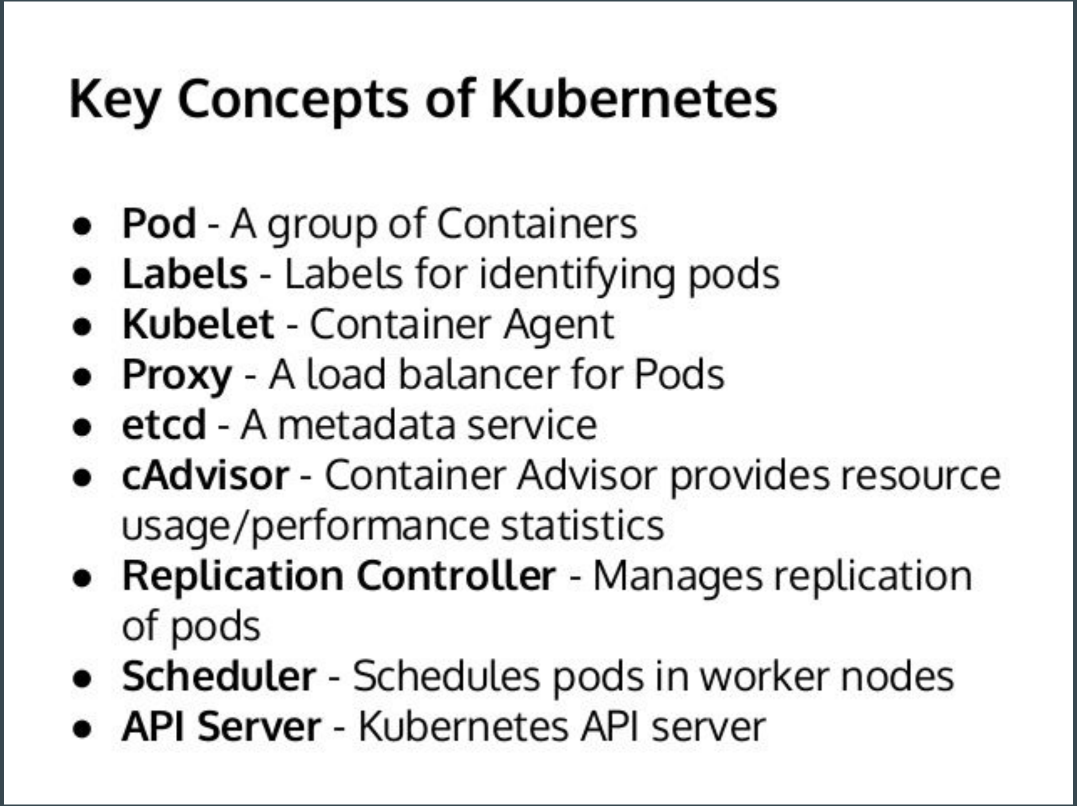
Matias Carrasco Kind • 09.14.2017

Outline

Introduction



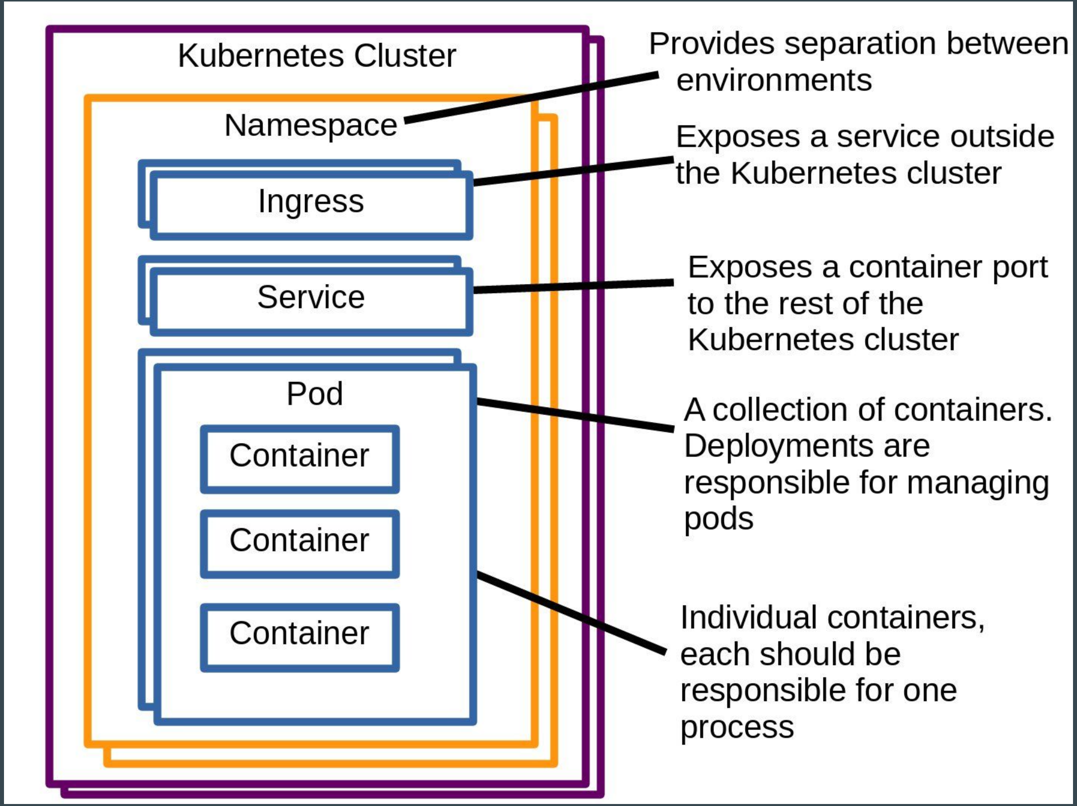
Introduction



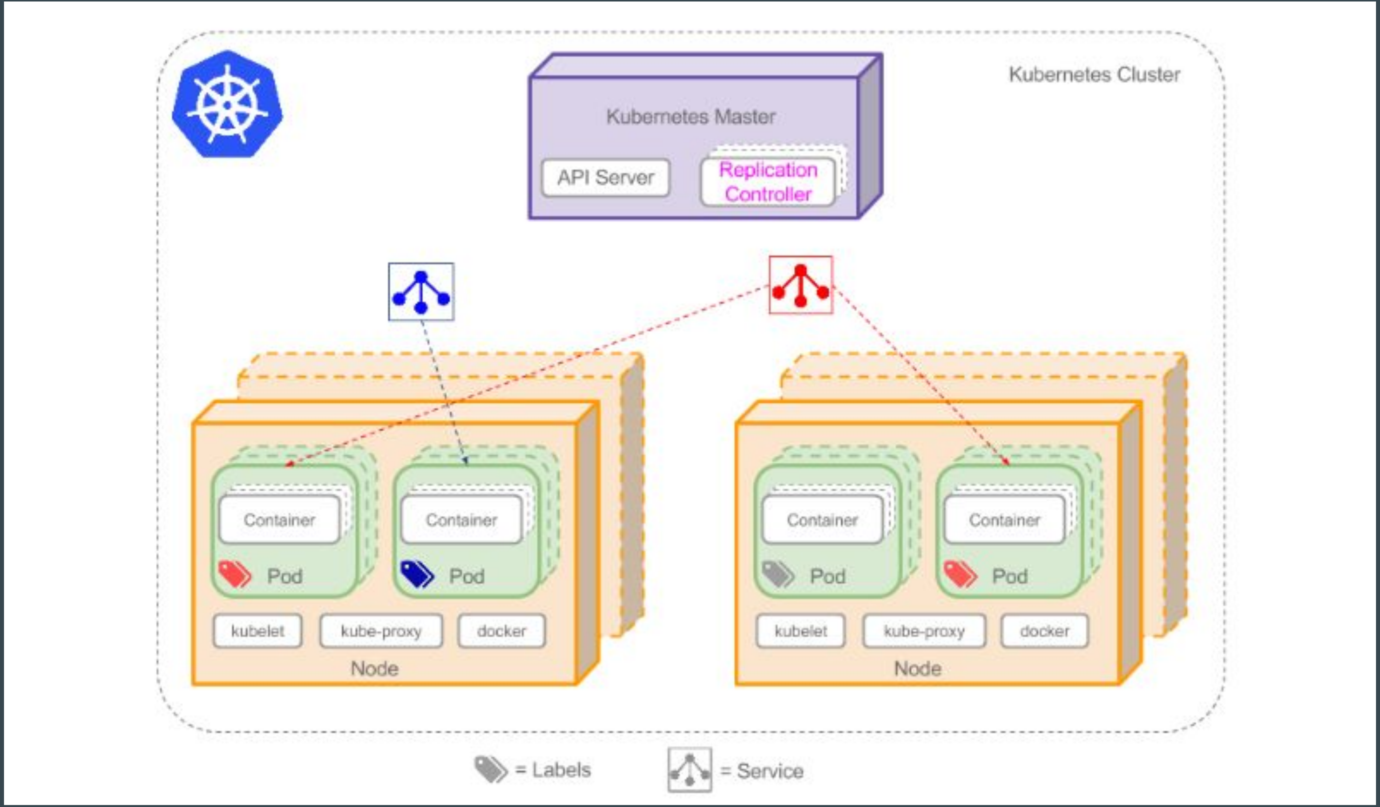
Key Concepts of Kubernetes

- **Pod** - A group of Containers
- **Labels** - Labels for identifying pods
- **Kubelet** - Container Agent
- **Proxy** - A load balancer for Pods
- **etcd** - A metadata service
- **cAdvisor** - Container Advisor provides resource usage/performance statistics
- **Replication Controller** - Manages replication of pods
- **Scheduler** - Schedules pods in worker nodes
- **API Server** - Kubernetes API server

Introduction


Kubernetes Cluster
Namespace
Ingress
Service
Pod
Container
Container
Container
Provides separation between environments
Exposes a service outside the Kubernetes cluster
Exposes a container port to the rest of the Kubernetes cluster
A collection of containers. Deployments are responsible for managing pods
Individual containers, each should be responsible for one process

Introduction



Current Efforts @ NCSA

- DES Labs (deslabs.ncsa.illinois.edu), collection of services for DES, @Nebula
  - Frontpage
  - SQL Web client with front-end, monitor, Redis and job submission (pods = celery workers)
  - JupyterHub for internal DES
  - Cutout server
- DES Data Release interface, similar to DES Labs, using GPFS
- LSST K8s @ Nebula (lsstlabs.ncsa.illinois.edu) under development
  - 15 nodes (8 cores, 16 GB RAM and 160GB local disk per node)
  - 3 Cinder volumes attached to master (10TB, 1TB, 1TB) served using NFS
  - Nginx Ingress controller, RBAC,
- 2nd k8s lsst cluster @Nebula for testing
  - Spark, dns testing, namespaces cross services, celery workers, Jobs
- Kubernetes cluster in Vsphere machines, openshift → under R&D
- Other efforts @NCSA using kubernetes from Openstack/Nebula from other groups, monthly kubernetes meeting

Current Efforts @ NCSA : Limitations (to be addressed…)

- Openstack/Nebula ≠ AWS, GKE
- No LoadBalancer type for services (dynamic ip provider), only using Ingress Rules, or especial ports
  - Not to be confused with Load Balancing for the end points of the services
- Cinder Volumes can only be attached to individual nodes, need to be served via NFS for Persistent volumes
- No dynamic volume provisioning
- Nodes need to created manually, for scalability
- No read-write GPFS, read-only access using 2 NFS bridges (unless mounted to network)
- And other small technical issues…

Security

Namespaces/Labels

- Namespaces; can partition cluster in resources, users, etc. Different namespaces for different environments (prod vs devel)
- Labels: Used to select resources within the cluster or namespace, to select pods, nodes, deployments,

Users/Groups/Service Accounts

- User and Groups refer to humans running and using resources. Permissions/Roles are applied at these levels . Cluster scoped
- Service Accounts are for processes, permissions/Roles can be applied to allow a running pod to schedule another pod, etc... Namespace scoped

Security

Roles/Cluster Roles

- Roles are namespaced scoped
  - CRUD resources namespace
  - Pods, deployments, PVC, service
  - Roles are bind to users/groups/sa
- Cluster Roles are cluster scoped
  - CRUD resources at cluster level
  - Nodes, namespaces, secrets, policy

Policies

- Pod Policy: at Cluster level to control how the pods/containers are run
  - Disabled running as root or a particular group, allow certain volumes to be mounted, limit access to port in host machine
- Network policy: how groups of pods are allowed to communicate with each other and other network endpoints. Namespaced scope, traffic control, use labels

Resource Manager

Resource Quotas

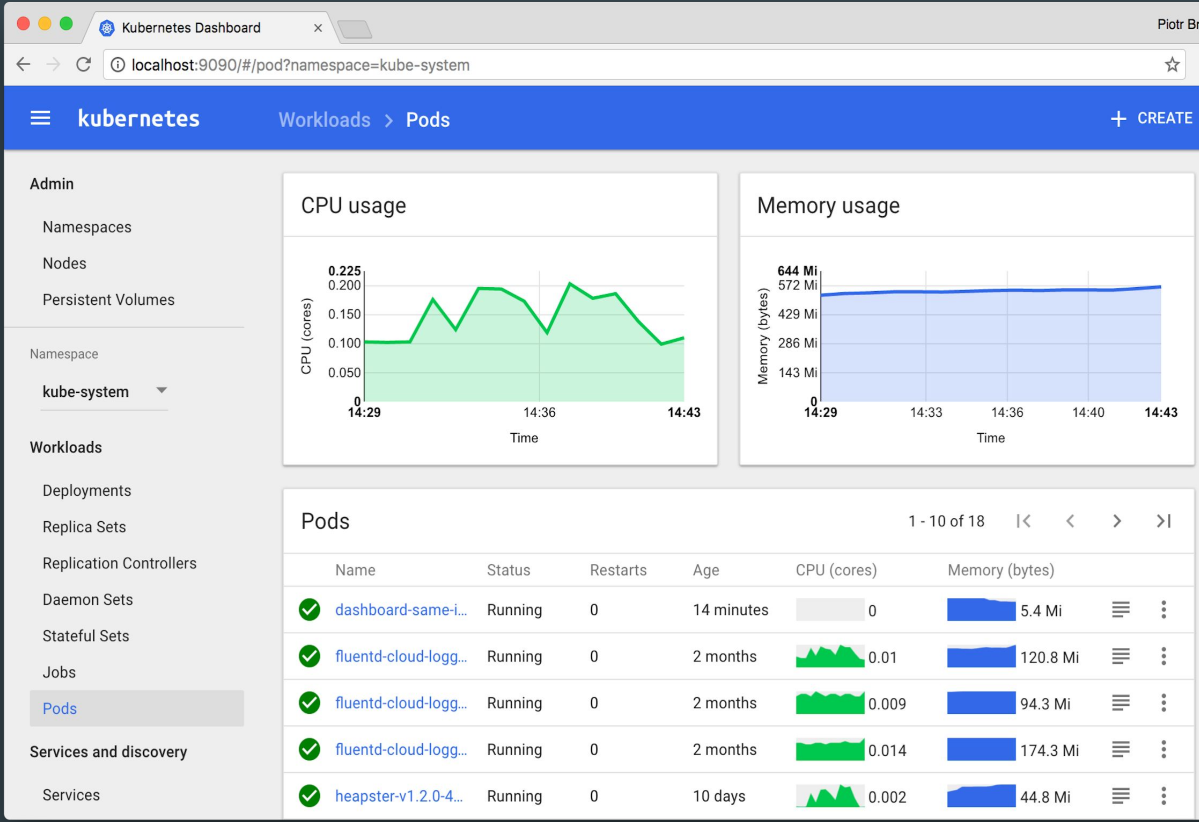
- Namespace scoped
- Limit cpu, memory, storage, etc
- Limit count of pods, deployments, claims, services, pods, etc.
- Doesn’t work on nodes
- Can be updated dynamically

Node Selectors/Admission Control

- By labeling nodes, nodes can be tainted, reserved or specifically selected for scheduling
- Can enforce a set of dedicated nodes for a namespace using Admission Control
- Need to restart api server (not cluster)

Monitor

- Daemons sets can monitor nodes health, volumes
- Etcd @ master monitor use of resources, status of resources
- Controller manager monitors the status of the deployments and other resources
- Dashboard



Name	Status	Restarts	Age	CPU (cores)	Memory (bytes)
✓ dashboard-same-i...	Running	0	14 minutes	0	5.4 Mi
✓ fluentd-cloud-logg...	Running	0	2 months	0.01	120.8 Mi
✓ fluentd-cloud-logg...	Running	0	2 months	0.009	94.3 Mi
✓ fluentd-cloud-logg...	Running	0	2 months	0.014	174.3 Mi
✓ heapster-v1.2.0-4...	Running	0	10 days	0.002	44.8 Mi

Other features

Federation

- Allows to controls and manage multiple clusters

Scalability

- Up to 5,000 nodes and 150,000 pods
- Auto horizontal scaling can be enabled and scripted

Cluster

Roles (examples)
create/delete pods (R1)
Read pods (R2)
Create deployments (R3)
System:admin (R4)

Cluster Roles
List nodes (C1)
List namespaces (C2)
List secrets (C3)
System: admin (C4)

Network Policy
No traffic allow (N1)
Only pod-pod allow (N2)

Namespace 1 (sqre)

Resources
- RQ: cpu, memory, volume
- Limit selector to labeled nodes

Network Policy (N1)

Users:

⬠ R1,R2,R3,C1,C2

♡

Service Account: SA_1 (R1)

Namespace 2 (devel)

Resources
- RQ: cpu, memory, volume
- Limit selector to labeled nodes

Network Policy (N1,N2)

Users:

♡ R1,R2,C1

Service Account: SA_1 (R1), SA_2(C1,R2)

kube-system (admin)

☺ R4,C4

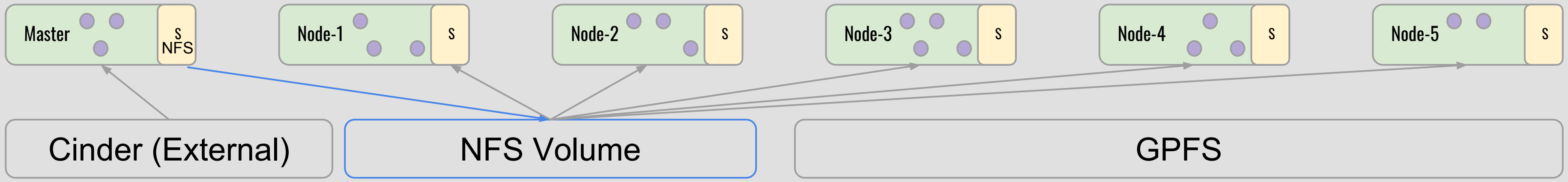
Users
Admin ☺
UserA ⬠
UserB ♡

Services Accounts
SA_1
SA_B

Pod Security Policy

-No root containers
-Limit volume access

Resources



Master
S
NFS
Node-1
S
Node-2
S
Node-3
S
Node-4
S
Node-5
S
Cinder (External)
NFS Volume
GPFS