Create A Highly Available Kubernetes Cluster

0 **December 6, 2016**

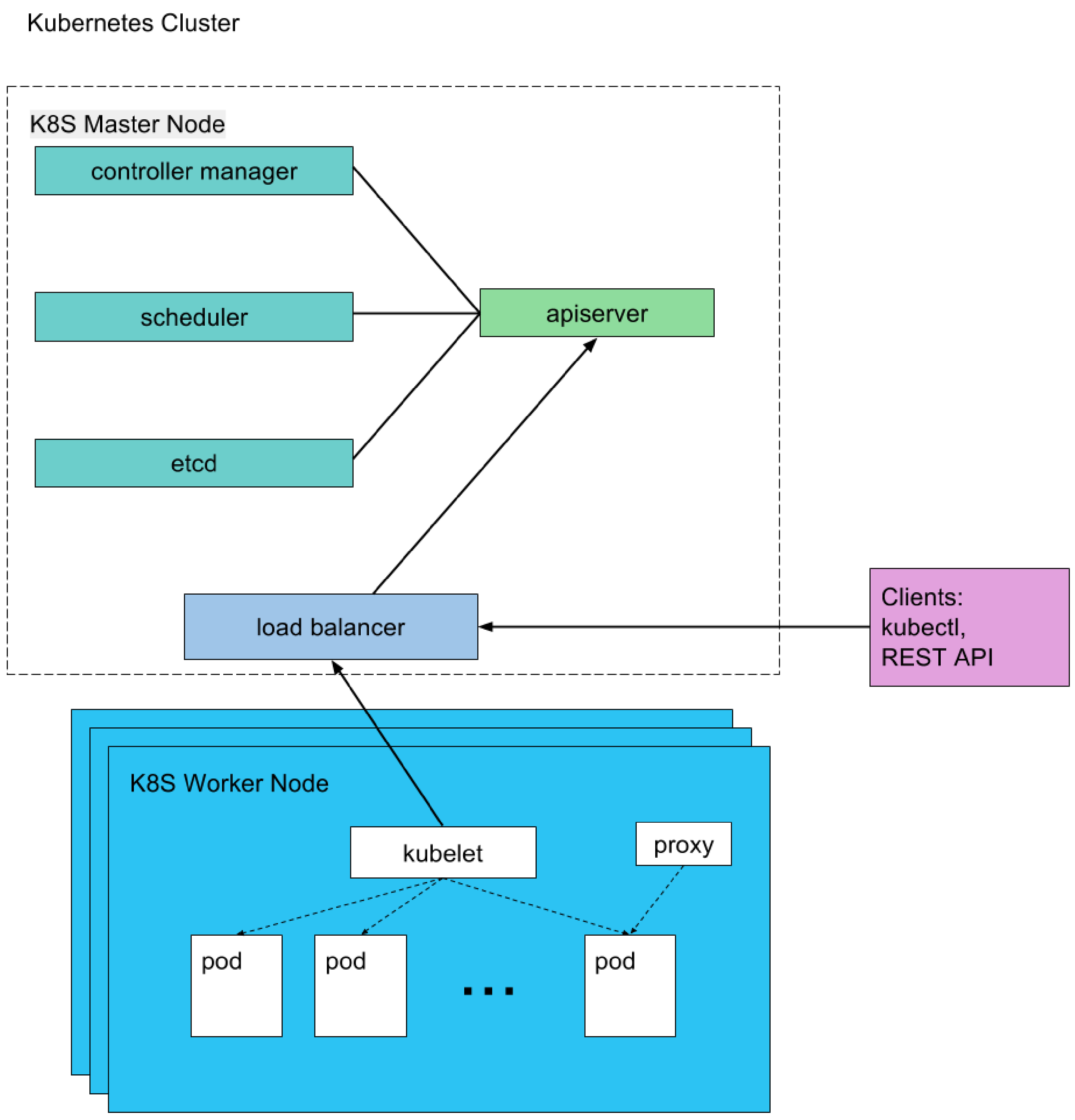
Tags: high availability kubernetes (https://platform9.com/tag/high-availability-kubernetes/), highly available Kubernetes (https://platform9.com/tag/highly-available-kubernetes/), kubernetes cluster (https://platform9.com/tag/kubernetes-cluster/), multi master kubernetes (https://platform9.com/tag/multi-master-kubernetes/)

Kubernetes clusters enable a higher level of abstraction to deploy and manage a group of containers that comprise the micro-services in a cloud-native application. A Kubernetes cluster provides a single Kubernetes API entry point, a cluster-wide resource naming scheme, a placement engine and scheduler for pods, a service network routing domain and an authentication and authorization model. In this article, we'll discuss how to create a highly available Kubernetes cluster.

As DevOps teams increasingly deploy cloud-native applications in production environments, they have to consider requirements like uptime in various failure scenarios, bursting workloads to public clouds during traffic spikes, architecting high availability across multiple regions/multiple availability zones etc. This leads to being able to deploy Kubernetes clusters in these scenarios:

- Across multiple availability zones, but within a single cloud provider. Cloud provider services like storage, load balancers, network routing can be assumed to easily interoperate
  - In the same geographical region – network performance will be fast enough to act like a single data center.
  - Across multiple geographical regions – high network cost and poor network performance may be prohibitive.
- Across multiple cloud providers with incompatible services and limited interoperability. Inter-cluster networking will be complex and expensive to set up in a performant manner

Cluster Components and Failure Scenarios



(https://platform9.com/wp-content/uploads/2016/12/Blog-HA-Cluster-K8S-diagram.png)

From the diagram above, the various components in the cluster are:

Master Components

Master components provide the cluster's control plane and are responsible for global activities about the cluster such as scheduling, and detecting and responding to cluster events.

- **kube-apiserver** – exposes the Kubernetes API and is the front-end for the Kubernetes control plane.

- **etcd** – used as Kubernetes' backing store. All cluster data is stored here. *etcd* may also be, and often is, set up in a cluster separate from the master node cluster.
- **kube-controller-manager** – runs controllers that handle routine tasks in the cluster e.g. Replication Controller, Endpoints Controller etc.
- **kube-scheduler** – watches newly created pods that have no node assigned, and selects a node for them to run on.

Node Components

Node components run on every node, maintain running pods and provide them the Kubernetes runtime environment.

- **kubelet** – the primary node agent that watches for pods that have been assigned to its node and performs actions to keep it healthy and functioning e.g. mount pod volumes, download pod secrets, run containers, perform health checks etc
- **kube-proxy** – enables the Kubernetes service abstraction by maintaining network rules on the host and performing connection forwarding.
- **Docker** – used for actually running containers.

Failure Scenarios

Looking at the cluster components and the deployment use cases in the previous section, there are many potential failure scenarios –

- **Loss of Master node** – If not configured for HA, loss of the master node or its services will have a serious impact to the application. The cluster will not be able to respond to API commands or deploy nodes. Each service in the master node along with the storage layer is critical and must be configured for HA
- **Loss of etcd service** – Whether *etcd* is run on the master node itself or setup separately from the master node, losing etcd data will be catastrophic since it contains all cluster information. *etcd* must be setup in a HA cluster to prevent this.
- **Loss of Worker node(s)** – In most cases, Kubernetes will be able to automatically detect and failover pods. Based on how the services are load balanced, there may be no impact to the end users of the application. If any pods on a node are non-responsive, then kubelet will detect that and inform the master to spin up another pod.
- **Loss of an entire availability zone (AZ)** – If the cluster is not architected to be highly available across availability zones, then loss of a AZ will bring down the application.

Depending on recovery process, it may take a while to bring up the application again on the cloud provider.

- **Network failures** – The master and worker nodes in a Kubernetes cluster can become unreachable due to network failure and partitions. The will be treated as node failures in some cases. For example, an auto scaling group will spin up a new node to replace a node that becomes unreachable.

Create Highly Available Kubernetes Clusters

Setting up reliable and highly available Kubernetes clusters will require a number of steps

- Make the master node services reliable
- Set up a redundant storage layer for etcd
- Use a highly available load balancer for the Kubernetes API services
- Setup multiple master nodes and configure a master election strategy

Make the master node services reliable

The cluster master node runs a number of critical processes that implement the Kubernetes API. Each of these must be automatically restarted if it fails. The kubelet service that runs on each worker node, and monitors the pods, is a convenient option to monitor these services on the master node. The kubelet service itself can be monitored by the OS specific tools such as monit on Debian.

Set up a redundant storage layer for etcd

In a Kubernetes cluster, the etcd service stores all the cluster data. If the etcd data is stored on a redundant and reliable storage layer, then the cluster state can be reconstructed during recovery from failure. In addition, multiple replicas of the *etcd* services are run to make the service itself highly available. Typically 3 to 5 instances of *etcd* are deployed to be able to form a quorum.The *etcd* services will replicate the storage to all master instances in the cluster.

Use a highly available apiserver and load balancer

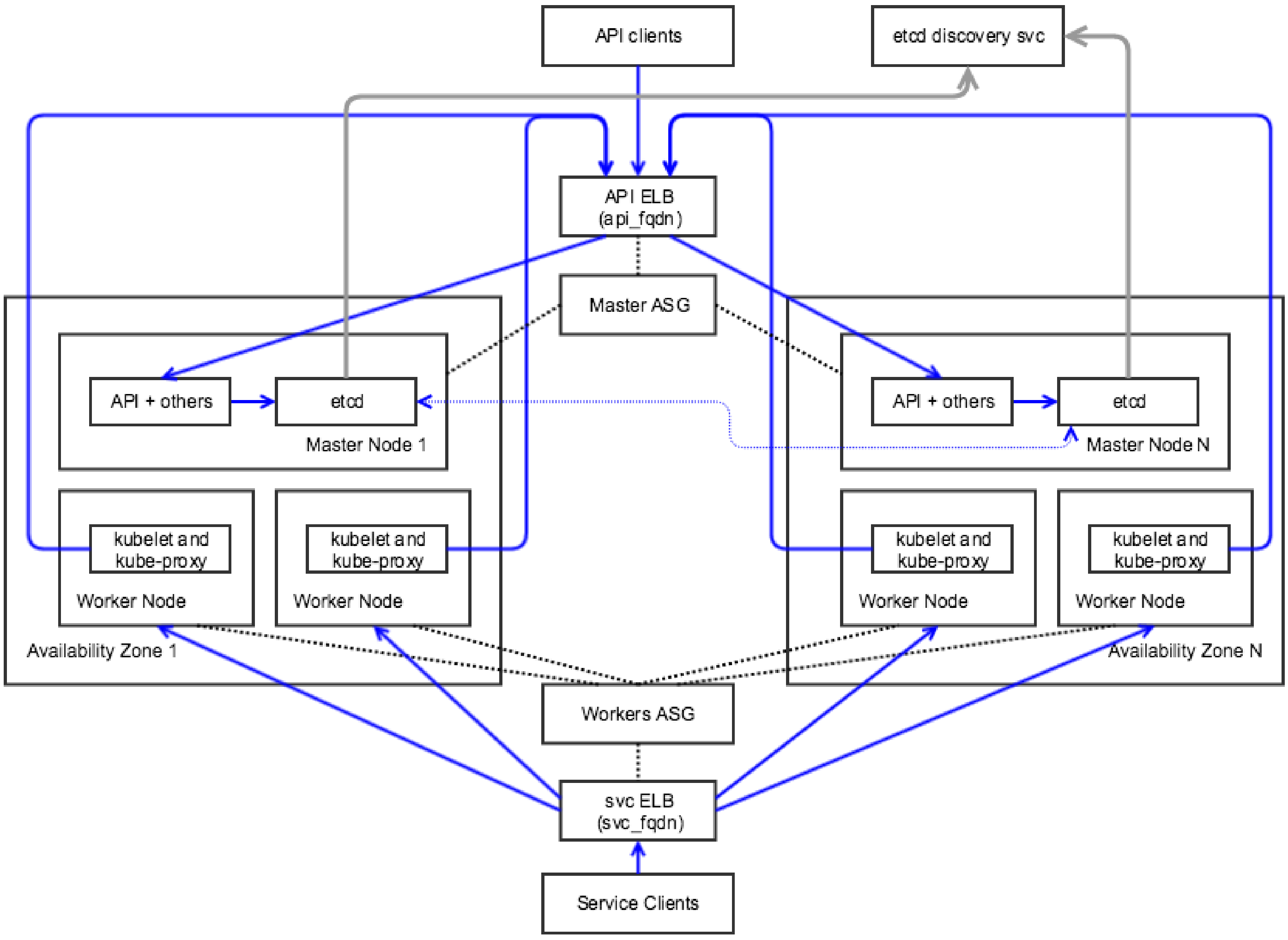
Similar to the replicated etcd service, the apiserver is also replicated in the cluster. Since the apiserver is the entry point to the cluster, the replicated apiserver is hosted behind a load balancer such as AWS ELB. If an instance of the apiserver goes down, the load balancer will automatically route the traffic to other running instances. In this case, the load balancer can also be a single point of failure. Therefore, multiple instances of the load balancer are run to make it highly available. For example, AWS ELB automatically creates and manages multiple instances of each load balancer, and typically uses Round Robin DNS to distribute requests.

Setup multiple master nodes

With the above steps, the master node service are made reliable, along with the *etcd* storage layer and apiserver. This still does not address the scenario where an entire master node goes down e.g. due to hardware failures. The previous steps will no longer be sufficient to keep the cluster accessible. The workaround is to run multiple instances of the master node.

However, when multiple instances of the master node are running, the controller manager and scheduler on only one of them should actually be performing actions on the cluster e.g. deploying new pods. API servers can run in parallel across multiple master nodes. To enable this, an election must be carried out among the master nodes when a cluster is brought up or when the current master dies. Tools such as *etcd*, Consul and Zookeepers can be used to elect a new master.

Platform9 Architecture for Highly Available Kubernetes Clusters



(https://platform9.com/wp-content/uploads/2016/12/K8S-Deployment-Diagram.png)

Legend: An arrow indicates "makes API calls to" or "forwards to". Grey lines describe etcd communications. Dotted line indicates "manages" or "is related to"

Platform9's Managed Kubernetes deploys highly available Kubernetes clusters, across Availability Zones in a region, using the above architecture:

- Input: one region, N availability zones
  - Can withstand a zone failure only if N >= 3
  - One master node per zone, containing 1 *etcd* instance and k8s master components (API, cluster manager controller, scheduler)
- One API ELB associated with one API FQDN for serving API requests.
- One Service ELB associated with one Service FQDN for serving service requests
- One Master Autoscaling Group.
  - Number of Master nodes = M = number of availability zones (+1 if # AZ is even)
  - Associated with all AZes
  - ASG registers its nodes with:
    - the API ELB

    - the Service ELB as well if masters are also schedulable workers (optional flag)
- One Worker ASG, associated with all zones
- Worker ASG associates itself with the API ELB to register its worker nodes for load distribution. When autoscaling workers, the ELB is automatically notified of new worker nodes.

Platform9 Managed Kubernetes supports creation of highly available, multi-master Kubernetes clusters that can tolerate failure of one or more master nodes as well as an entire availability zone going down. This is critical for running production workloads in Kubernetes. A highly available cluster is composed of at least 3 master nodes. We choose an odd number of master nodes so that it's possible to establish quorum. At least one instance of a master node is deployed per availability zone. The worker nodes are then evenly spread across given availability zones.

If one availability zone goes down, an election may be required depending on which services were running on the lost nodes. For example, if the nodes in the AZ were hosting an *etcd* leader, a "follower" scheduler, and a leader controller manager, then re-election occurs in cluster for *etcd* and controller manager, but not for the scheduler.

Platform9 Managed Kubernetes also creates a highly available *etcd* deployment behind the scene, with each master node having a copy of *etcd*.

To see how easy it can be to set up highly available Kubernetes clusters, please check out a free trial (https://platform9.com/free-trial/) of Platform9 Managed Kubernetes.

(/resource/the-ultimate-guide-to-deploy-kubernetes/)