Kubernetes is a powerful open-source system, initially developed by Google, for managing containerized applications in a clustered environment. It aims to provide better ways of managing related, distributed components and services across varied infrastructure.

we'll discuss some of Kubernetes' basic concepts. We will talk about the architecture of the system, the problems it solves, and the model that it uses to handle containerized deployments and scaling.

What is Kubernetes?

- Kubernetes, at its basic level, is a system for running and coordinating containerized applications across a cluster of machines. It is a platform designed to completely manage the life cycle of containerized applications and services using methods that provide predictability, scalability, and high availability.
- Kubernetes is a platform for hosting docker containers In a clustered environment with multiple docker hosts, Providing grouping, load balancing,auto-healing ,and scaling
- As a Kubernetes user, you can define how your applications should run and the ways they should be able to interact with other applications or the outside world. You can scale your services up or down, perform graceful rolling updates, and switch traffic between different versions of your applications to test features or rollback problematic deployments. Kubernetes provides interfaces and composable platform primitives that allow you to define and manage your applications with high degrees of flexibility, power, and reliability.

Introduction to Kubernetes

Kubernetes architecture is broadly divided into master/worker role and each of the role designation has different functions to play.

The master node is the control plane for the entire cluster and exposes the application program interface (API), container scheduling, controllers management and deploying and managing the entire cluster. Nodes are the workhorses of a Kubernetes cluster.

They expose compute, networking and storage resources to applications. Each node runs a container runtime (such as docker or rkt), an agent that communicates with the master, additional components for logging, monitoring, service discovery and optional add-ons.

Nodes can be virtual machines (VMs) running in a cloud or bare metal servers running within the data center.

Kubernetes is used for managing cloudnative micro-services applications and also for CI/CD. The most common features of Kubernetes are:

- Portability
- Service discovery
- Security
- Replication of services
- Auto-scaling
- Load balancing
- Rolling updates
- Logging across services
- Monitoring/ health checks

WHAT DOES IT SOLVE?

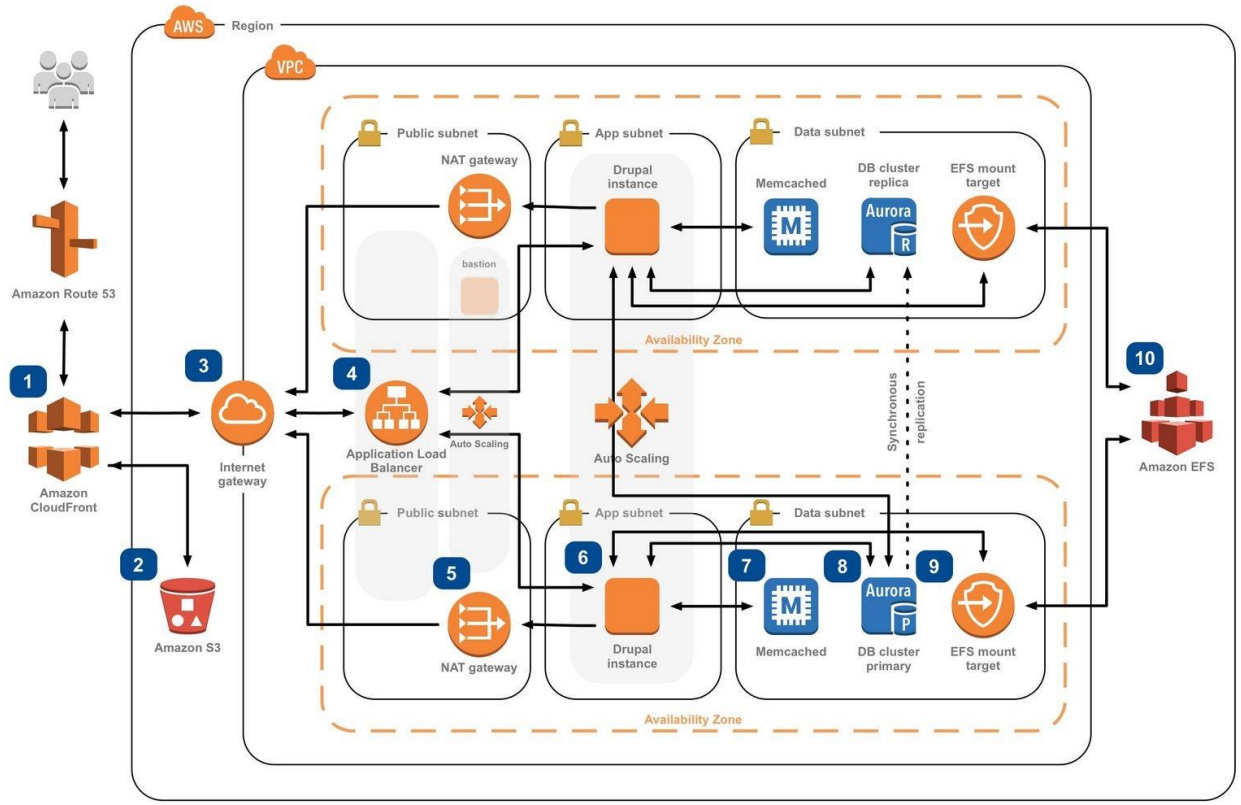
- Describe networked applications simply
  - Put app components into Docker containers
  - Containers see flat network
  - Simple relationships between containers
  - Make networked storage easy to use
- Build resiliency and redundancy into infra
  - Health checking, load balancing, failure detection
  - Run many workloads on the same machines

What we know:

- Very popular
- Based on containers

Questions:

- Why are containers great?
  - Why are they not enough?



But complex.

Kubernetes' strength is abstraction paired with powerful, automated, infrastructure and app management.

1. Automation of many difficult tasks for ops & API access.
2. An even playing field to devs, regardless of cloud.
3. With abstracted principles.

Key Feature

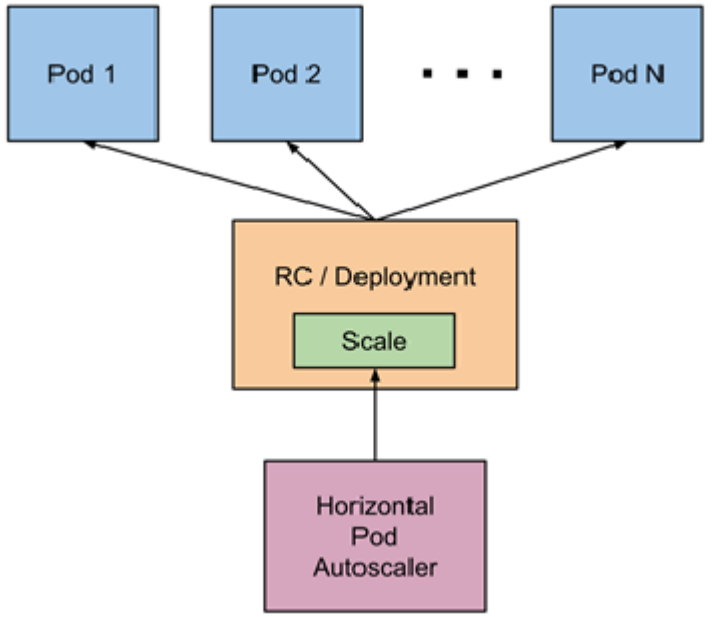
- Automatic binpacking
- Horizontal Pod Autoscaling(HPA)
- Automated rollouts and rollbacks
- Storage orchestration
- Self-healing
- Service discovery and load balancing
- Secret and configuration management
- Batch execution

Automatic binpacking

- CPU/Memory's utilization can define on Pods (Smallest Unit of Kubernetes)
- Schedule will select the node by ensure all resource is enough for running Pods as required
- If reach memory limit
- Current Pods will be terminate (Kill)
- If restart flag was set. Kubenetes will try to restart Pods on other node
- If reach cpu limit • Schedule will not kill Pods and waiting for it back to normal state
- Check available node resource by command: kubectl describe node

Horizon Pods Autoscaling (hpa)

- CPU/Memory's utilization can define on Pods (Smallest Unit of Kubernetes)
- Schedule will select the node by ensure all resource is enough for running Pods as required
- If reach memory limit
- Current Pods will be terminate (Kill)
- If restart flag was set. Kubenetes will try to restart Pods on other node
- If reach cpu limit • Schedule will not kill Pods and waiting for it back to normal state
- Check available node resource by command: kubectl describe node



Storage Orchestrator

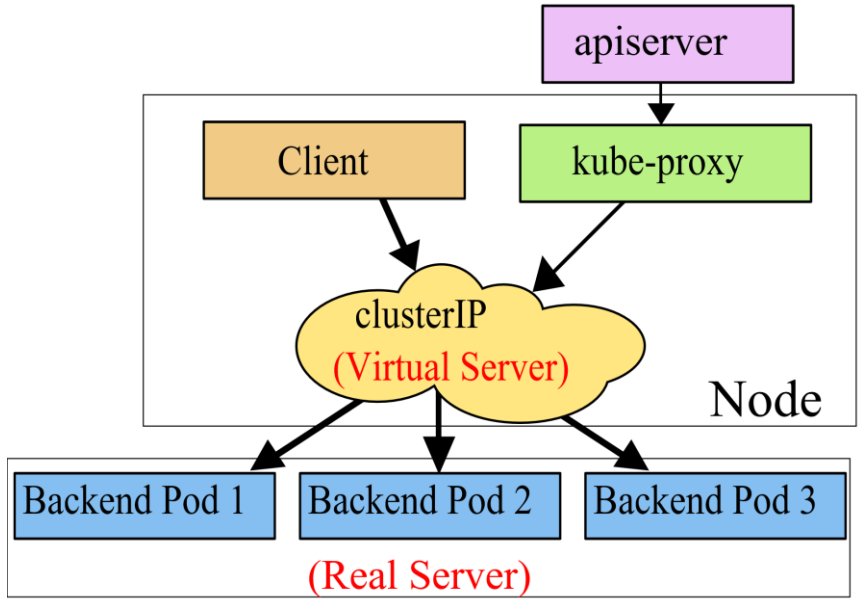
- Support several storage type:
- Local Storage • Network Storage (NFS, iScsi, Gluster, Ceph, Cinder, Flocker)
- Cloud Storage (AWS, GCE, AzueDisk etc)

Self-healing

- Replication Controller (RC) will maintain unit of Pods as design (not to much (kill) and not to few (create)
- Full-fill Pods on every failure case with automatic by system

Service Discovery and Load Balancing

- Service will act like “connector” for client need to connect with Pods
  - o Discovery will use for service to look “Pods” by environment variable or dns service
- Support load balancing between multiple Pods (replica)



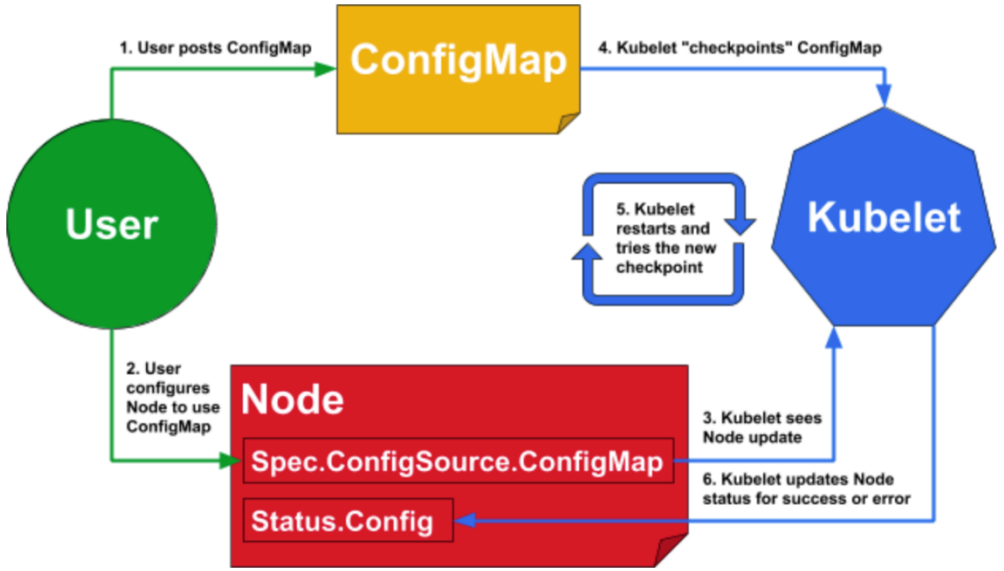
Secret and Configuration Management

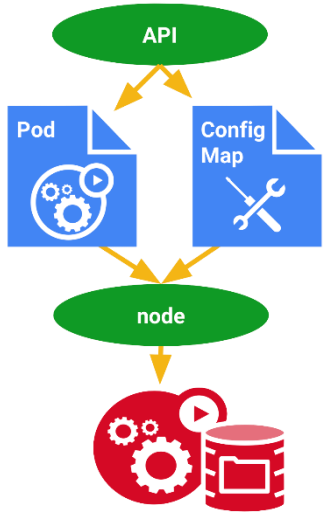
- Kubernetes can keep confidential data (such as username/password) for running application on encrypt format.
- Can reference on Pods instead plan text configuration.

```
# Create files needed for rest of example.
$ echo -n 'admin' > ./username.txt
$ echo -n '1f2d1e2e67df' > ./password.txt
The kubectl create secret command packages these files into a Secret and creates the object on the Apiserver.
$ kubectl create secret generic db-user-pass --from-file=./username.txt --from-file=./password.txt
secret "db-user-pass" created
You can check that the secret was created like this:
$ kubectl get secrets
NAME                  TYPE                                  DATA      AGE
db-user-pass          Opaque                                2         51s
$ kubectl describe secrets/db-user-pass
Name:           db-user-pass
Namespace:      default
Labels:         <none>
Annotations:    <none>
Type:           Opaque
Data
====
password.txt:   12 bytes
username.txt:   5 bytes
```

Secret and Configuration Management

- Sometime we have a many configuration that need to specify on each application, But it should be change every time need.
- Idea is create configuration file (ConfigMap) for define all configuration that reference on Pods instead





Batch Execution

- A job will response some kind of batch execute by create special Pods (Terminate when batch complete)
- Normally kubernetes will use job for maintain many background process for cluster system such as Replication Controller (RC) will submit job for start new Pods when existing is fail or delete. o Type of Job
- Non-parallel jobs
- Parallel jobs with fix-completion
- Parallel jobs with work queue

single-job

Details Events YAML

Message	Reason	First Seen	Last Seen	Count
Created pod: single-job-v7zbm	SuccessfulCreate	Jul 28, 2018, 3:38:03 PM	Jul 28, 2018, 3:38:03 PM	1