云计算开源产业联盟

全球云计算开源峰会2017 中国·北京
聚合云计算新势力，拥抱全世界新开源
GLOBLE CLOUD COMPUTING OPEN SOURCE SUMMIT

Kubernetes and Google Container Engine

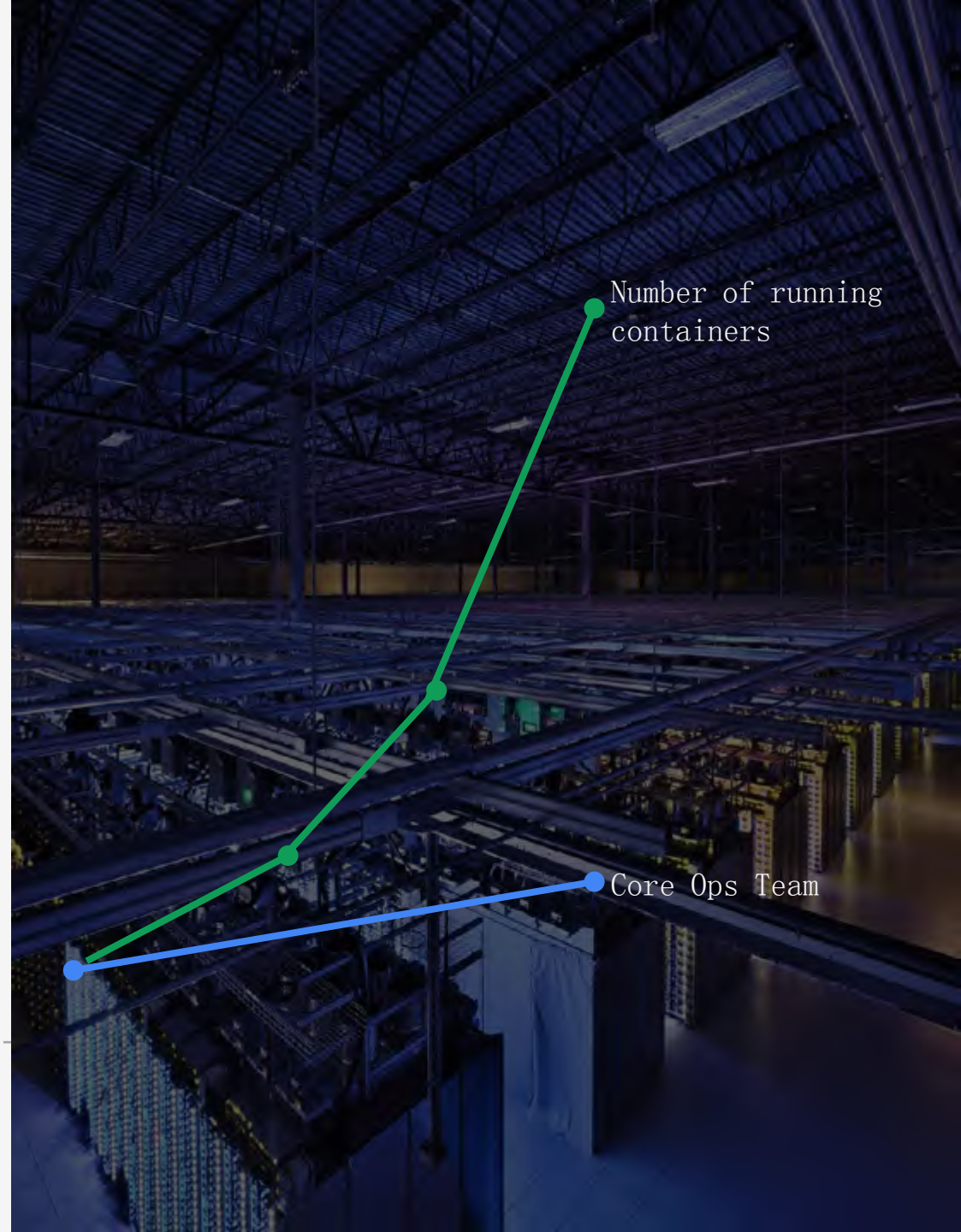
Harry Wang
Google Inc.

Google Cloud Platform


Google Cloud Platform

Containers make operations easier

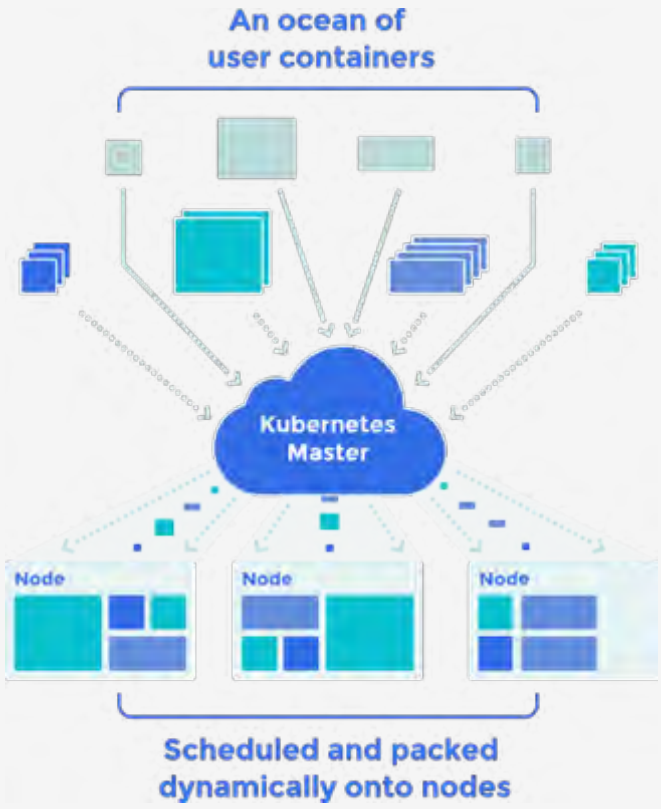
Enabled Google to grow our fleet over 10x faster than we grew our ops team



What is Kubernetes

Container orchestration

- Container centric infrastructure
- Inspired by Google’s internal systems and experience managing containers
- **Runs Anywhere**
- Open sourced in 2014
- Created CNCF to host Kubernetes and an ecosystem of cloud-native infrastructure

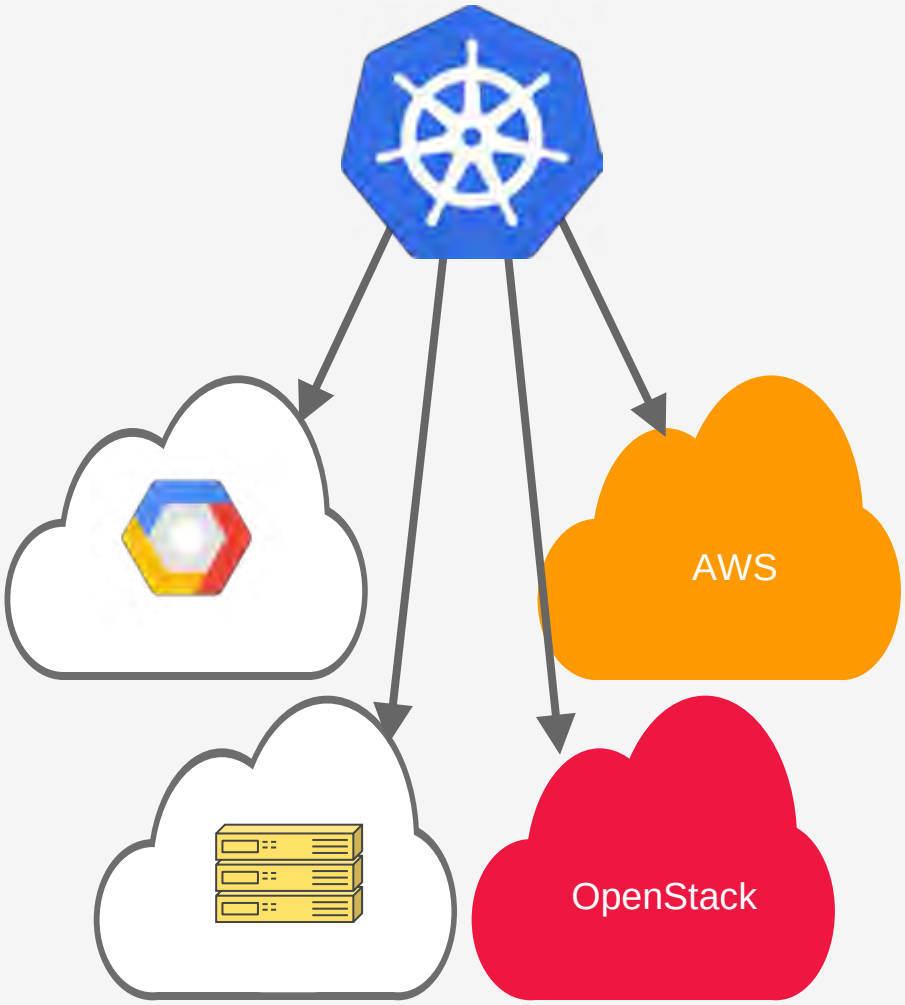


Platform flexibility

Runs in many environments, including “bare metal” and “your laptop”

The API and the implementation are 100% open

The whole system is modular and replaceable



Kubernetes community

1,500+
Contributors

43,000+
Commits

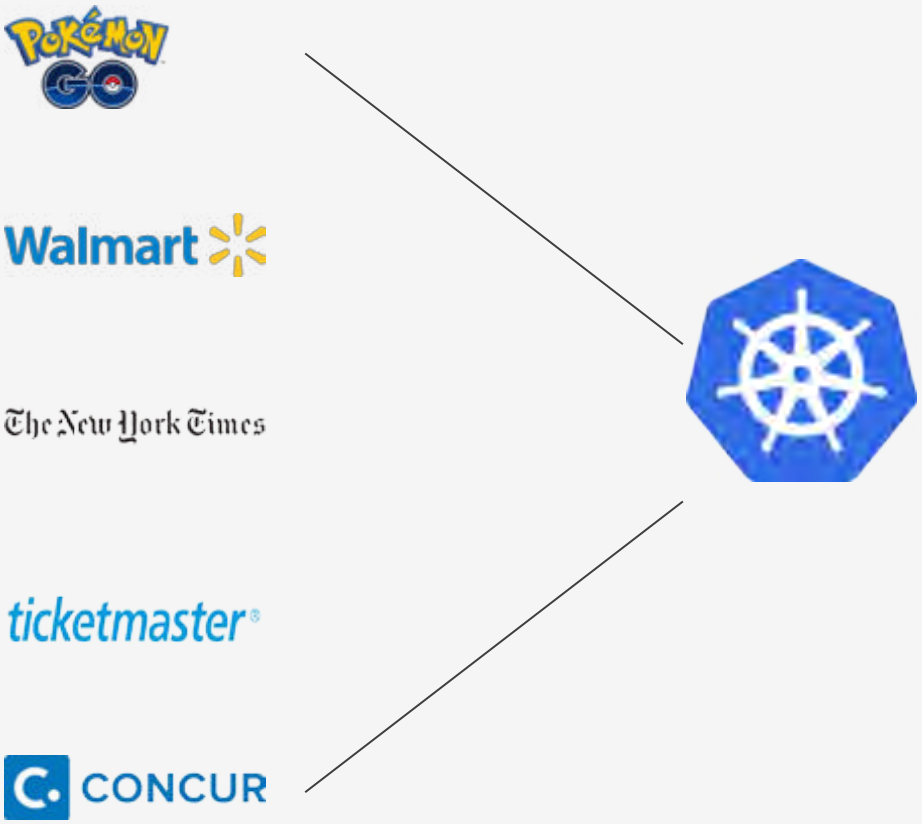
4,000+
External Projects Based on Kubernetes

200+
Meetups Around the World

Kubernetes community

Contributors and users

Fast, scalable, open



Fast: Developer productivity

- Minutes from commit to prod
- Release 20—50X / day

Scalable: efficient scale out

- Fastest app to 1bn usd
- Black Friday demand

Open: use anywhere

- 200 Walmart warehouses on VMware
- Hybrid and multi cloud

Key Concepts in Kubernetes

Pods

A small group of tightly coupled containers & volumes, composed together

The atom of Kubernetes

Shared lifecycle and fate

Shared networking - a shared “real” IP, containers see each other as localhost

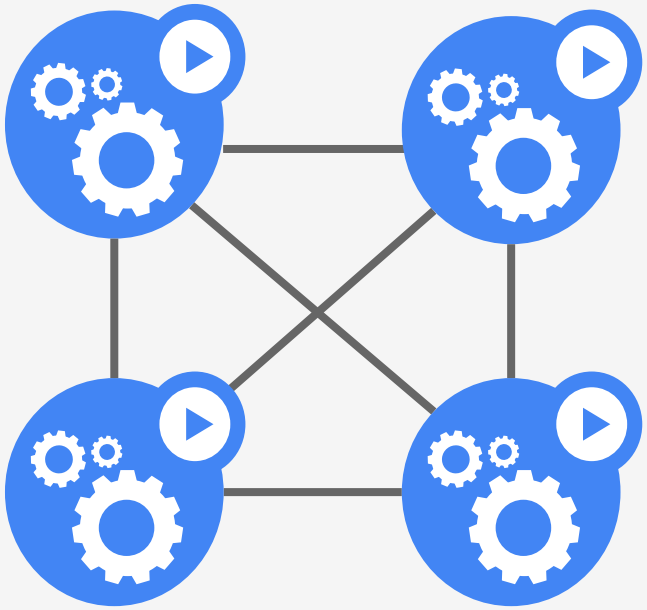
Every pod has a real IP address

This is different from the out of the box model Docker offers

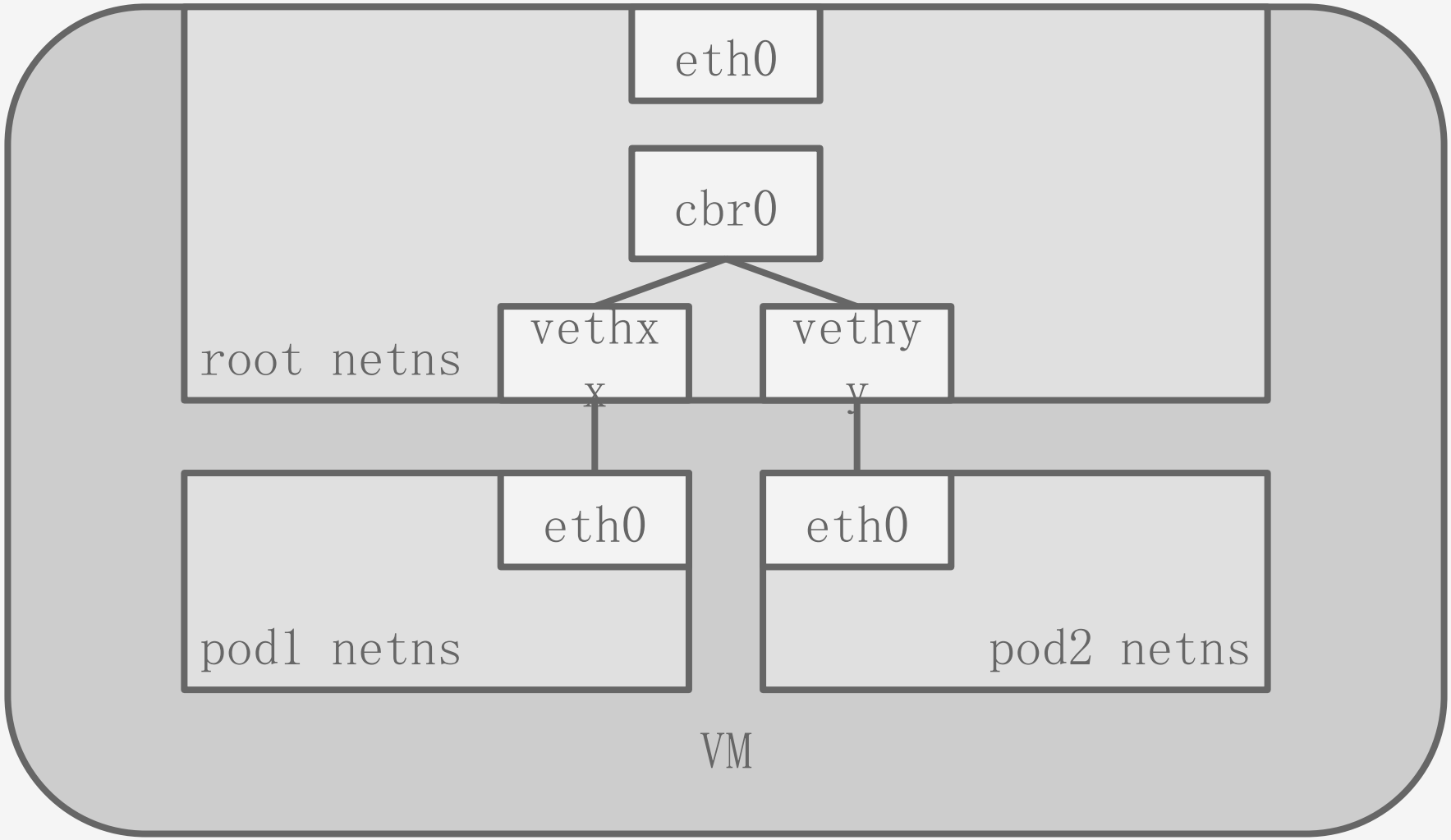
- No machine private IPs
- No port mapping

Pod IPs are accessible from other pods, regardless of which VM they are on

Linux “network namespaces” (aka “netns”) and virtual interfaces



Network namespaces

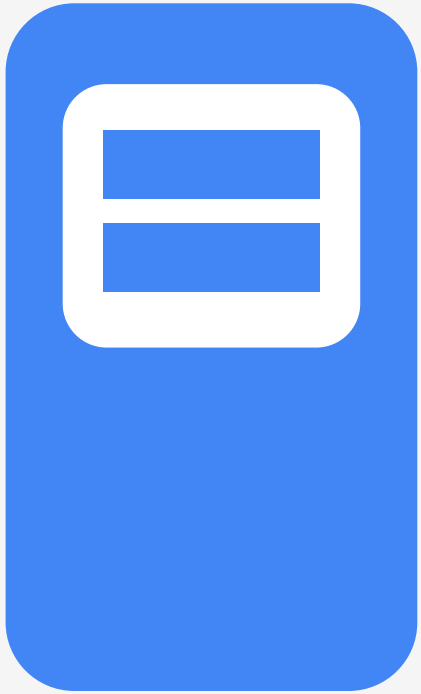


Node

Machine where containers run

Transparent for cluster users

On the radar of cluster **administrator**

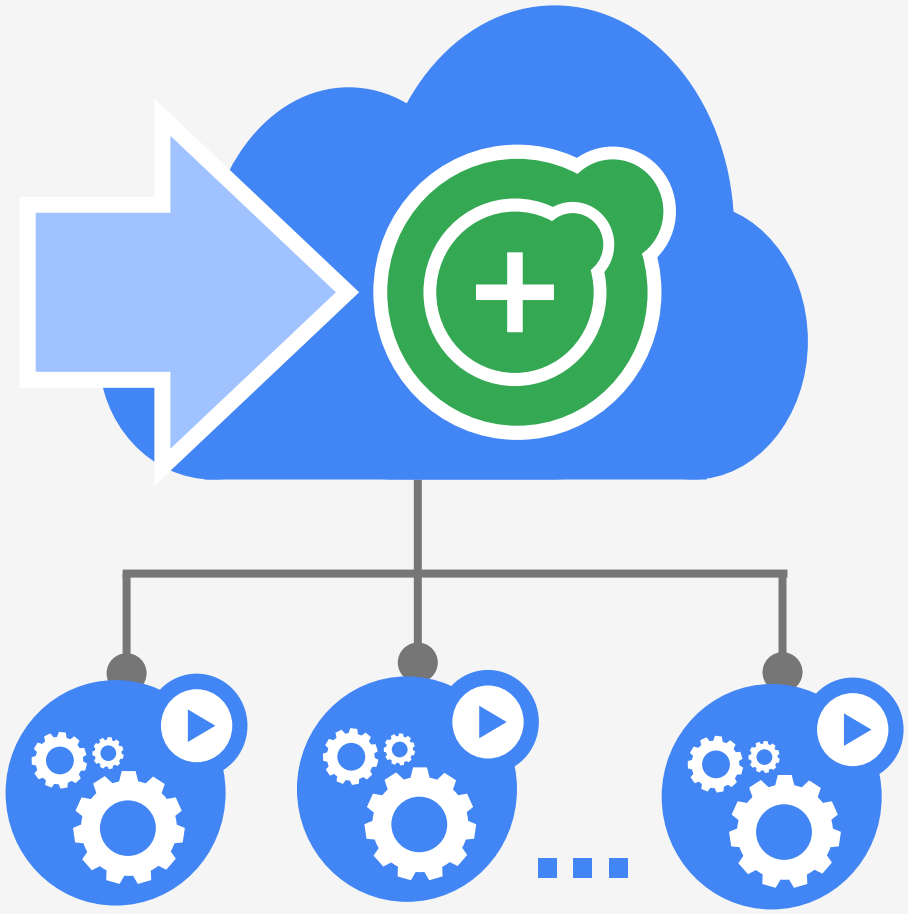


Deployment

The way to **deploy** an application

Creates and **updates** instance of the application

Self-healing mechanism



The service abstraction

A service is a group of endpoints (usually pods)

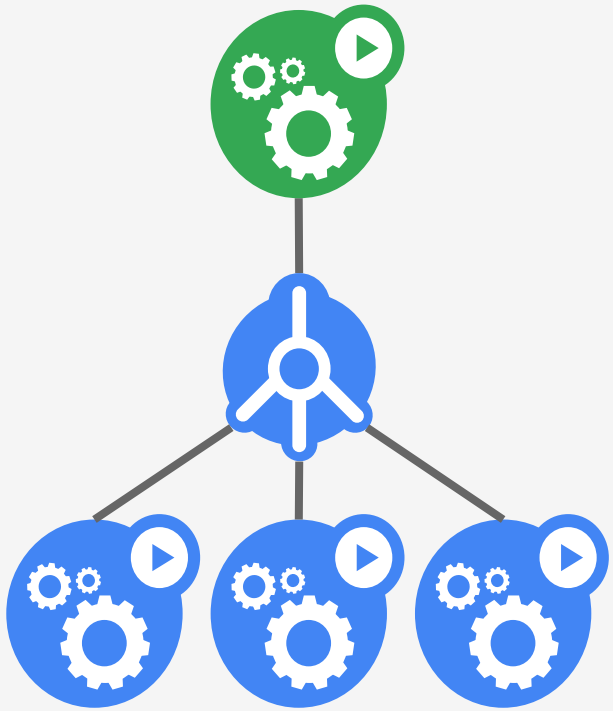
Services provide a stable VIP

VIP automatically routes to backend pods

- Implementations can vary
- We will examine the default implementation

The set of pods “behind” a service can change

Clients only need the VIP, which doesn’t change



Service

What you submit is simple

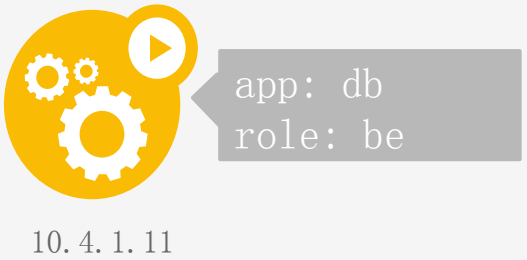
- Other fields will be defaulted or assigned

The ‘selector’ field chooses which pods to balance across

```
kind: Service
apiVersion: v1
metadata:
  name: store be
spec:
 selector:
    app: store
    role: be
  ports:
  - name: http
    port: 80
```

Endpoints

```
selector:
  app: store
  role: be
```

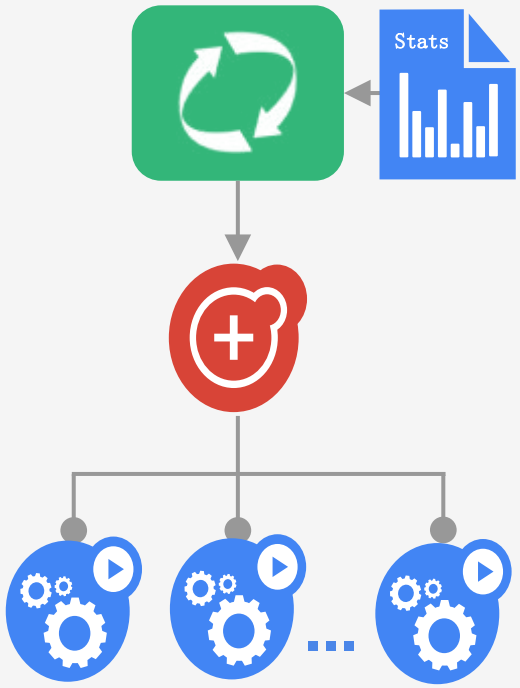


Horizontal pod autoscaling

Automatically add (or remove) pods as needed

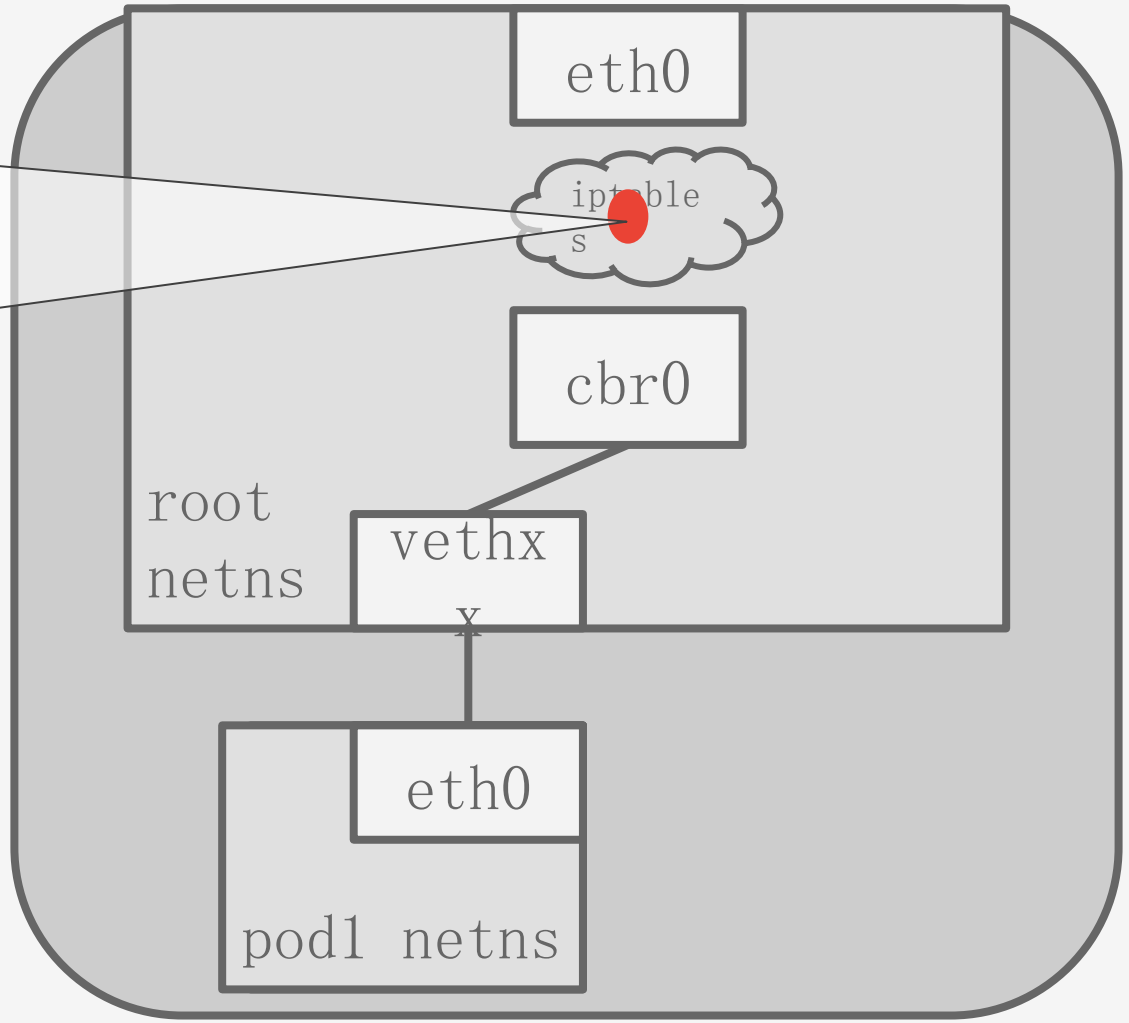
- Based on CPU utilization (for now)
- Custom metrics in Alpha

- Efficiency now, capacity when you need it
- Operates within user defined min/max bounds
- Set it and forget it



Pod to service

src: pod1
~~dst: svc1~~
dst: pod99

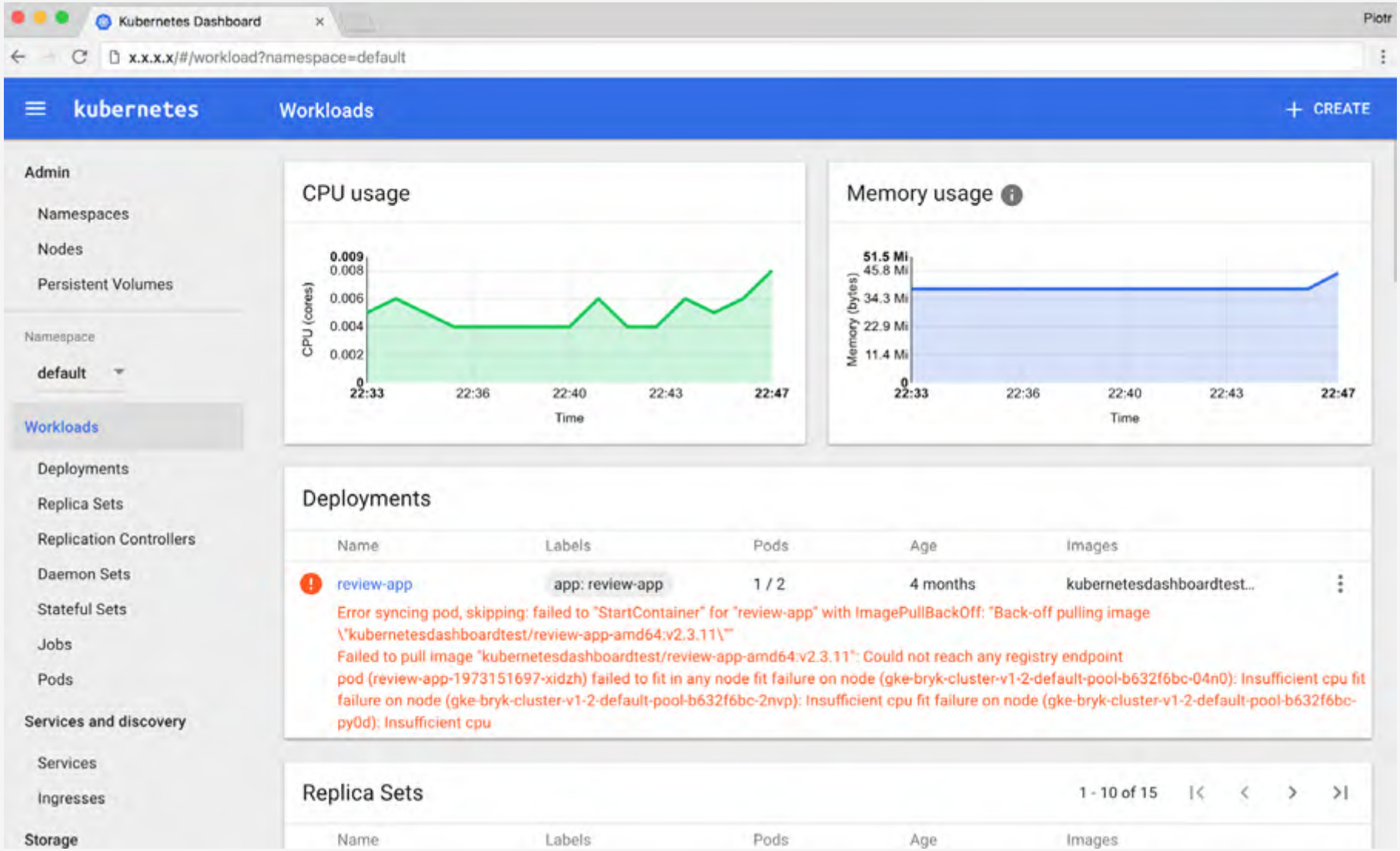


kubectl

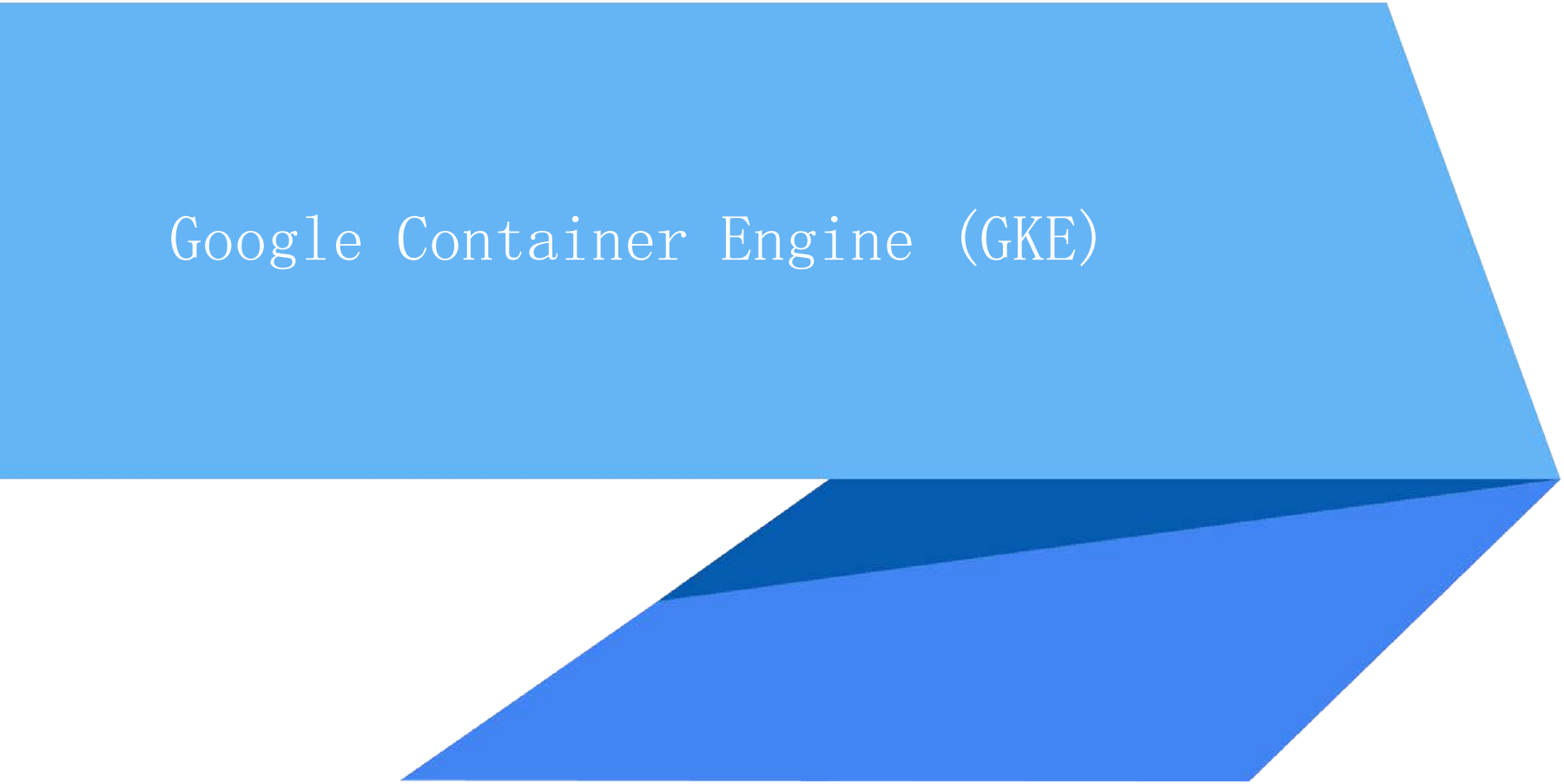
Kubernetes **command line** client tool

Controls Kubernetes cluster manager

```
$kubectl
kubectl controls the Kubernetes cluster manager.

Find more information at https://github.com/kubernetes/kubernetes.

Basic Commands (Beginner):
  create         Create a resource by filename or stdin
  expose         Take a replication controller, ...
  run            Run a particular image on the cluster
  set            Set specific features on objects

Basic Commands (Intermediate):
  get            Display one or many resources
  explain        Documentation of resources
  edit           Edit a resource on the server
  delete         Delete resources by filenames, ...
```


Kubernetes Dashboard
Piotr
x.x.x.x/#/workload?namespace=default
kubernetes
Workloads
CREATE
Admin
Namespaces
Nodes
Persistent Volumes
Namespace
default
Workloads
Deployments
Replica Sets
Replication Controllers
Daemon Sets
Stateful Sets
Jobs
Pods
Services and discovery
Services
Ingresses
Storage
CPU usage
CPU (cores)
0.009
0.008
0.006
0.004
0.002
0
22:33
22:36
22:40
22:43
22:47
Time
Memory usage
Memory (bytes)
51.5 Mi
45.8 Mi
34.3 Mi
22.9 Mi
11.4 Mi
0
22:33
22:36
22:40
22:43
22:47
Time
Deployments
Name
Labels
Pods
Age
Images
review-app
app: review-app
1 / 2
4 months
kubernetesdashboardtest...
Error syncing pod, skipping: failed to "StartContainer" for "review-app" with ImagePullBackOff: "Back-off pulling image \"kubernetesdashboardtest/review-app-amd64:v2.3.11\""
Failed to pull image "kubernetesdashboardtest/review-app-amd64:v2.3.11": Could not reach any registry endpoint
pod (review-app-1973151697-xidzh) failed to fit in any node fit failure on node (gke-bryk-cluster-v1-2-default-pool-b632f6bc-04n0): Insufficient cpu fit failure on node (gke-bryk-cluster-v1-2-default-pool-b632f6bc-2nvp): Insufficient cpu fit failure on node (gke-bryk-cluster-v1-2-default-pool-b632f6bc-py0d): Insufficient cpu
Replica Sets
1 - 10 of 15
Name
Labels
Pods
Age
Images

Google Container Engine (GKE)



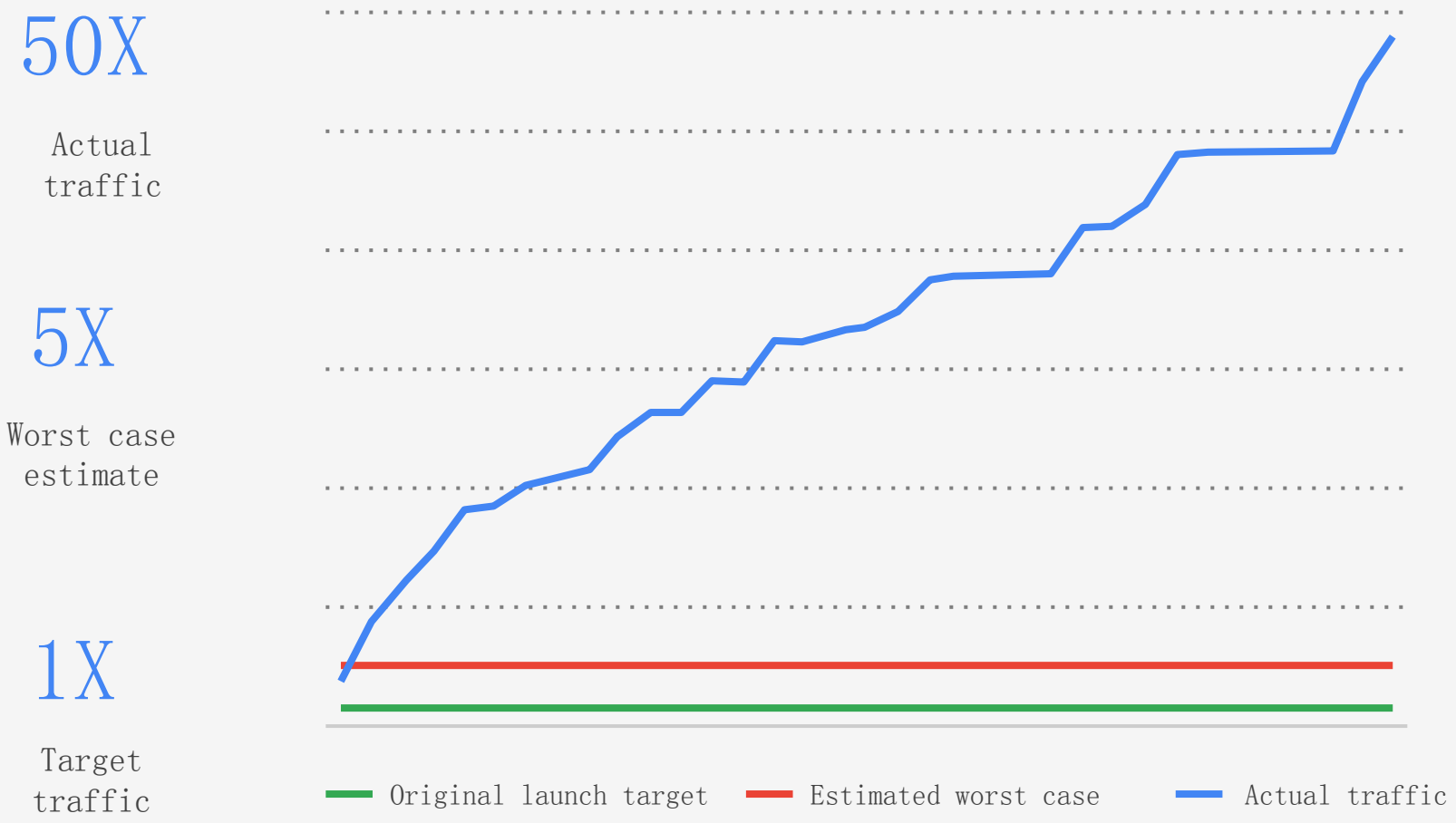
Google Container Engine:

- Pure upstream Kubernetes
- Operational excellence
- Monitoring, logging, IAM
- Network and load balancer integration

Images by Connie Zhou

Google Cloud Platform

Cloud Datastore Transactions Per Second



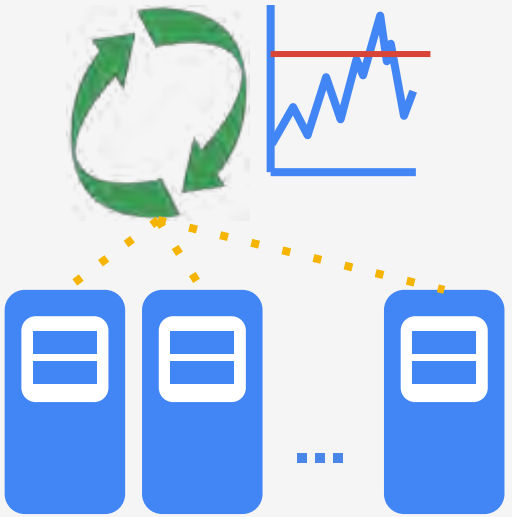
Cluster autoscaling

Add VMs when needed

- Based on unschedulable pods
- New VMs self-register with API server

Remove VMs when not needed

- e.g. CPU usage too low



BETA

GKE networking

Pods must be reachable across VMs.

Kubernetes doesn’t care HOW, but this is a requirement

- L2, L3, or overlay

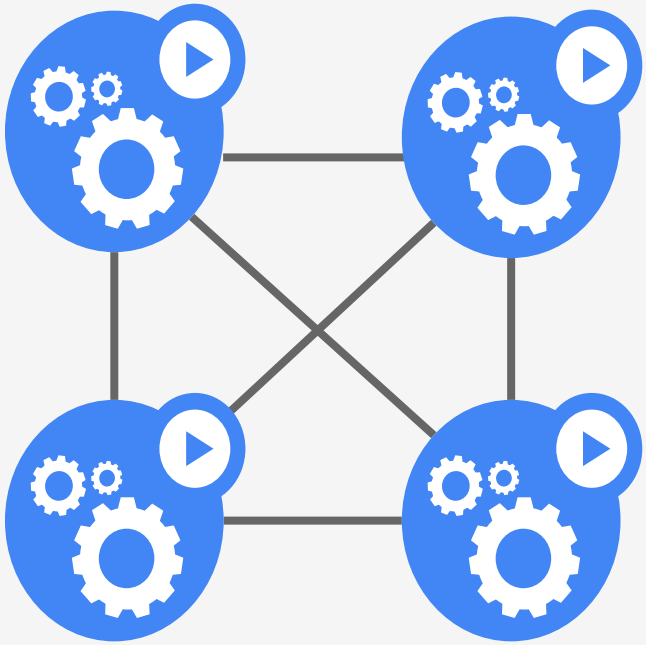
GKE VMs are created as “routers”

- --can-ip-forward
- Disable anti-spoof protection for this VM

Add one GCP static route for each VM

- gcloud compute routes create vm2 --destination-range=x.y.z.0/24 --next-hop-instance=vm2

The GCP network does the rest.



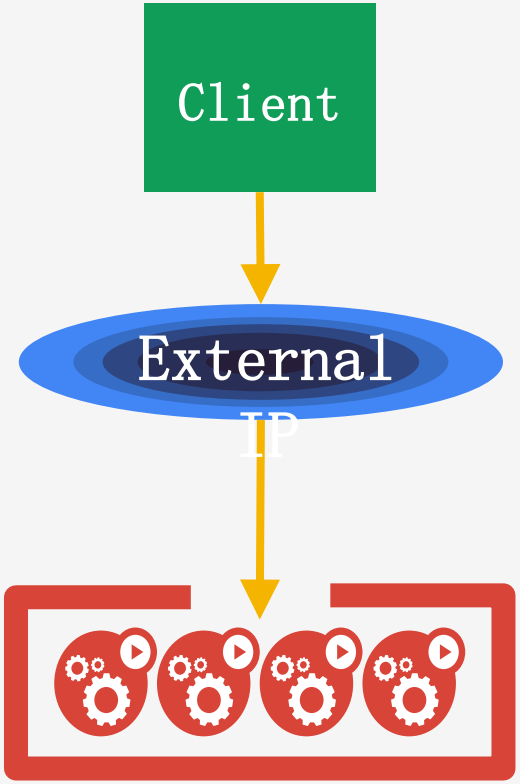
Load balancing (L4)

Services are a group of endpoints (usually pods) that provide a stable virtual IP (VIP)

The set of pods behind the VIP can change but clients only need the VIP, which doesn't change

External services are exposed as an IP and port

On Google Container Engine this is done using a Google Cloud Network Load Balancer



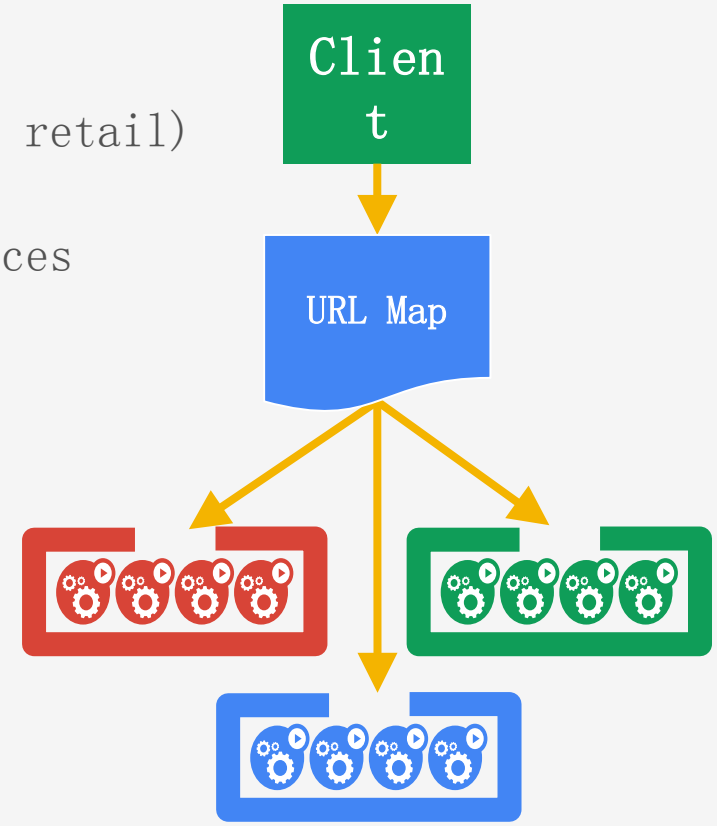
Load balancing (L7)

Many apps use HTTP/HTTPS (e.g., games, social, retail)

Ingress maps incoming traffic to backend services

- by HTTP host headers
- by HTTP URL paths

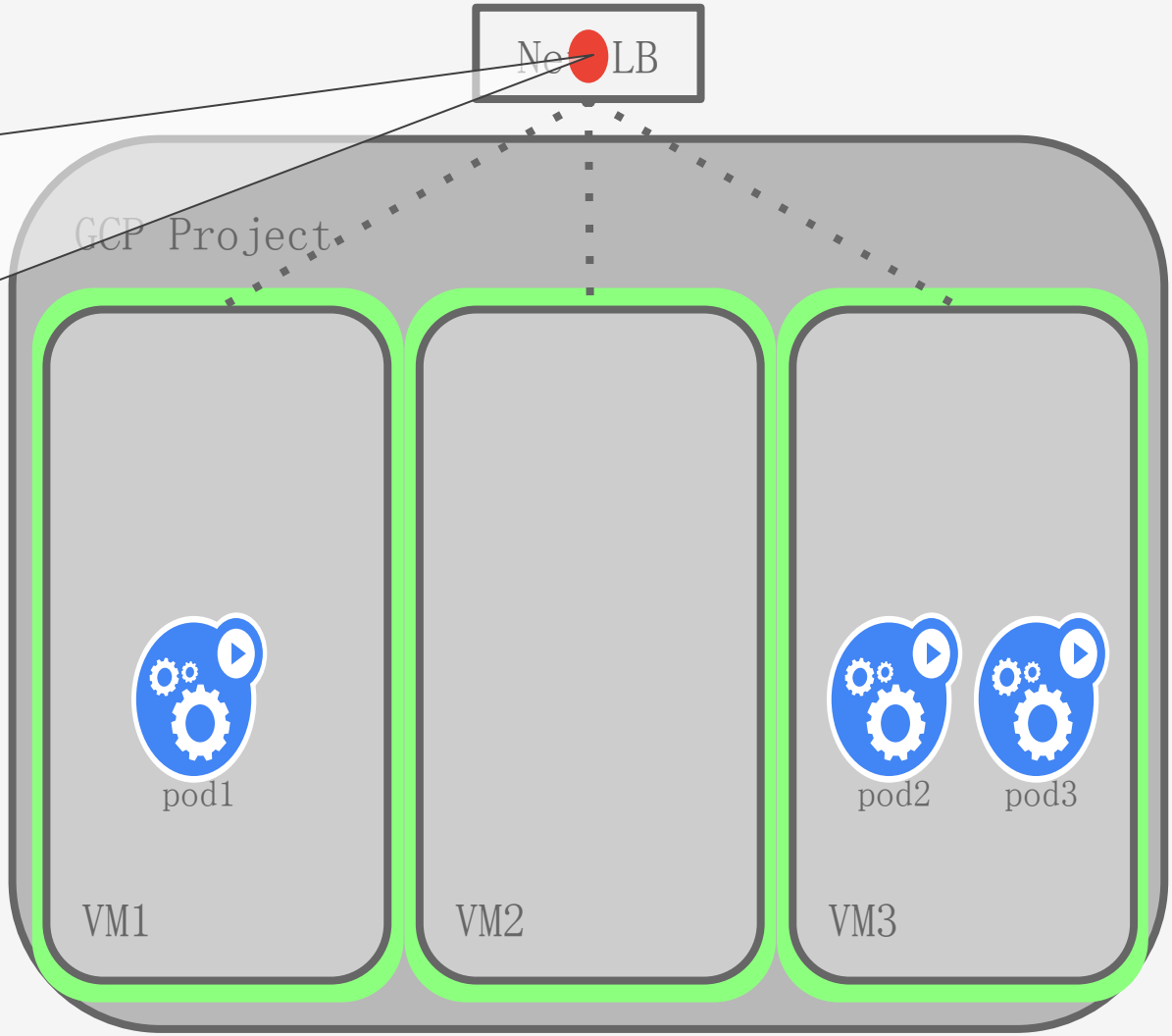
On Google Container Engine this is done using a Google Cloud HTTP(S) Load Balancer



Receiving external-to-service traffic

src: client
dst: LB

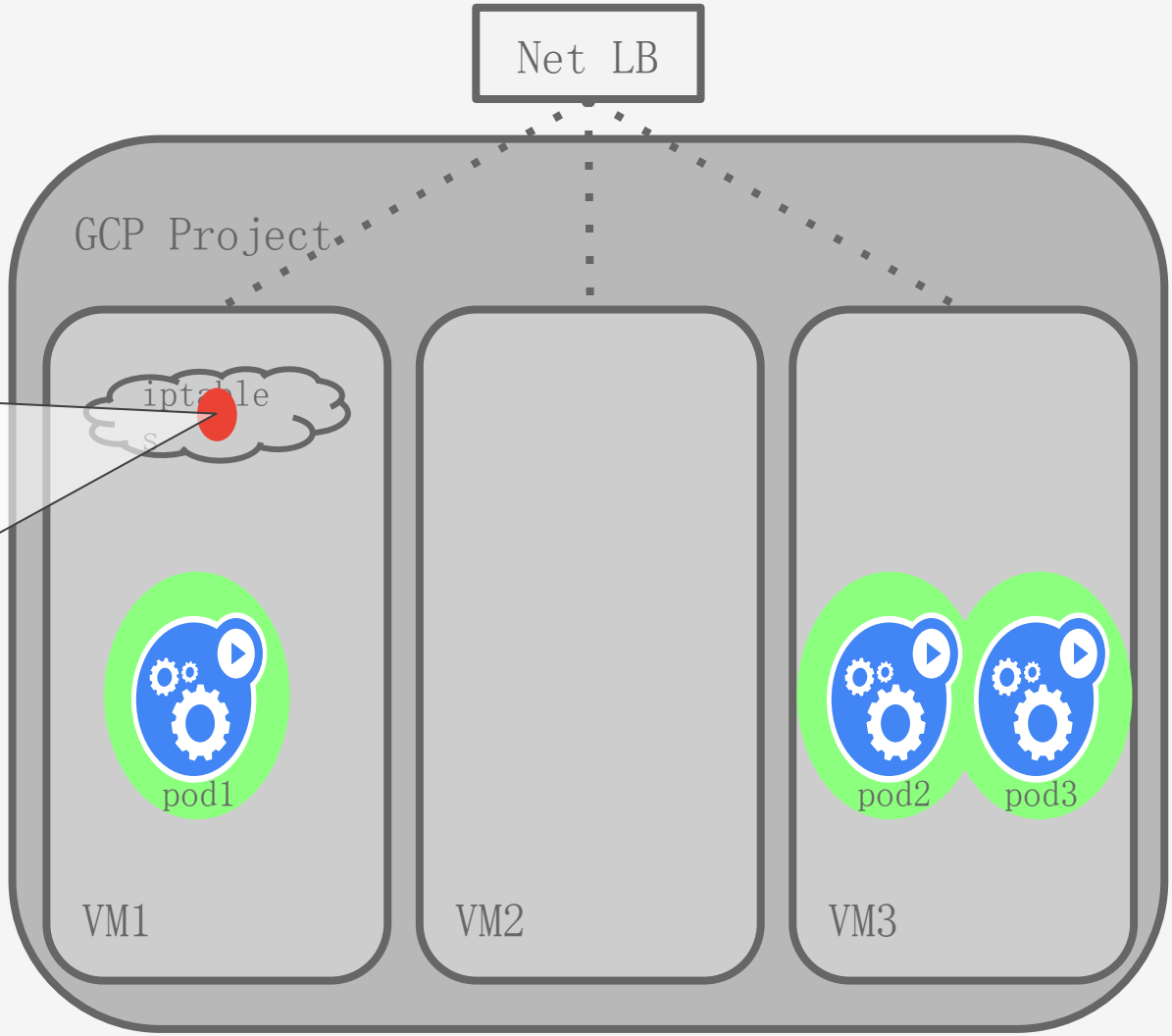
Choose a VM



Receiving external-to-service traffic

src: client
dst: LB

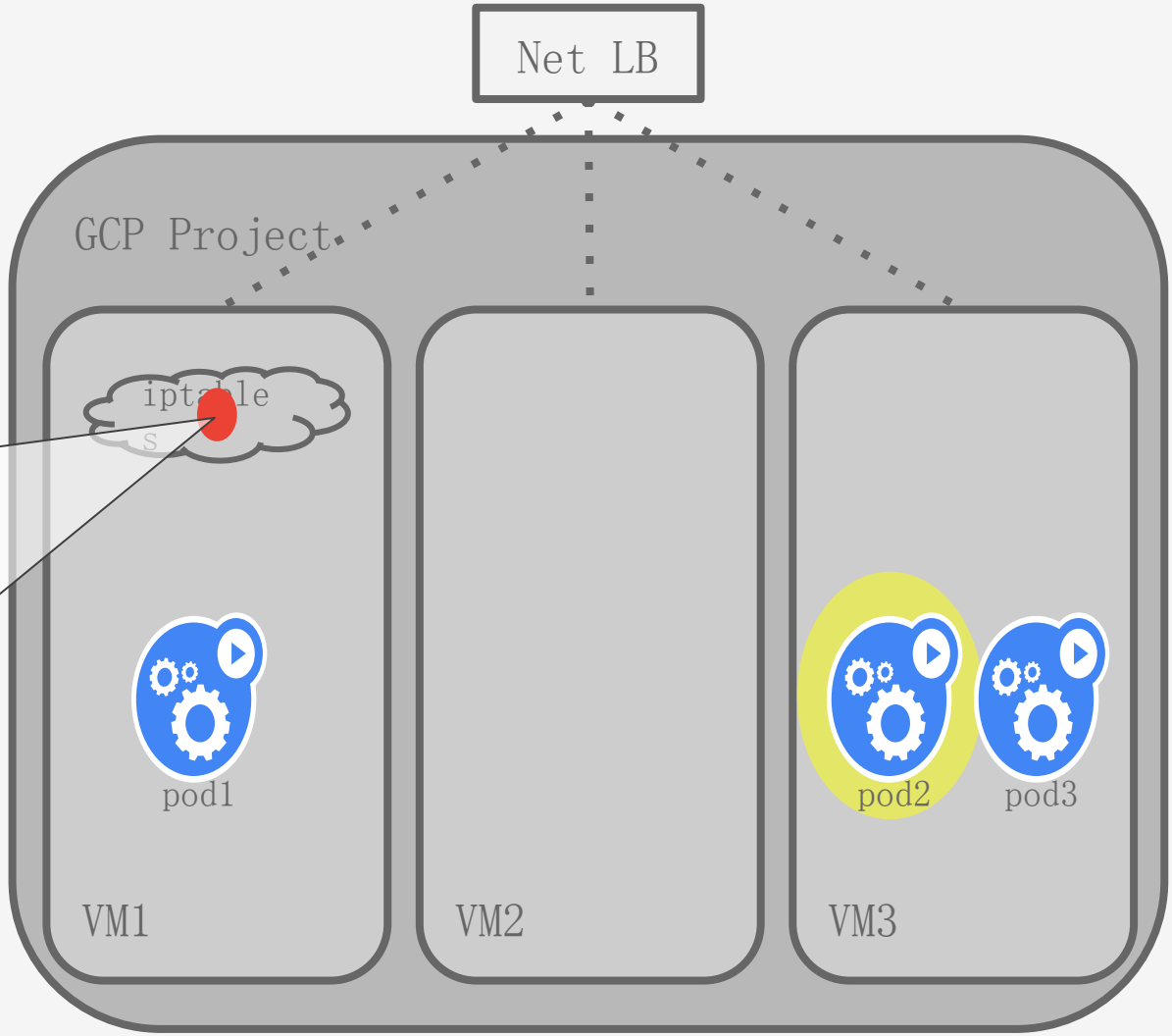
Choose a pod



Receiving external-to-service traffic

src: client
~~dst: LB~~
dst: pod2

NAT



Thank you!