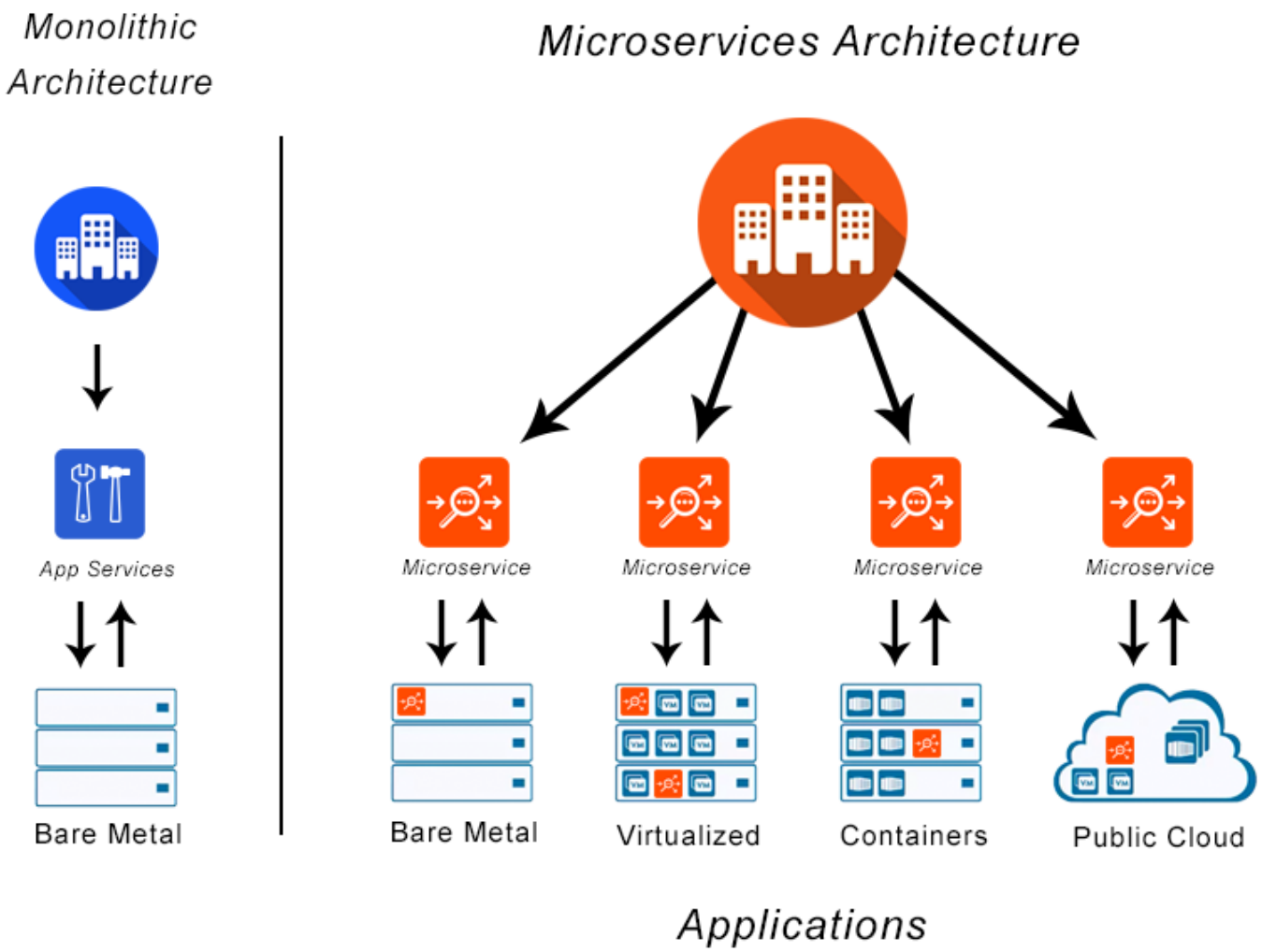
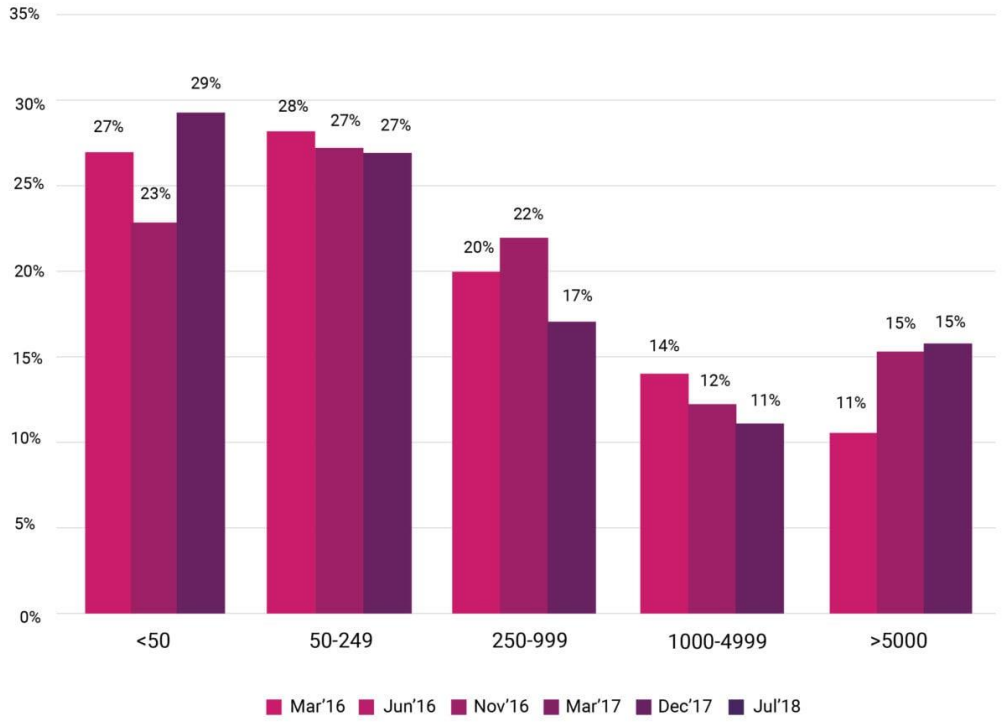
Auto Scaling in Kubernetes

Neha Bhoi - Liam Nguyen

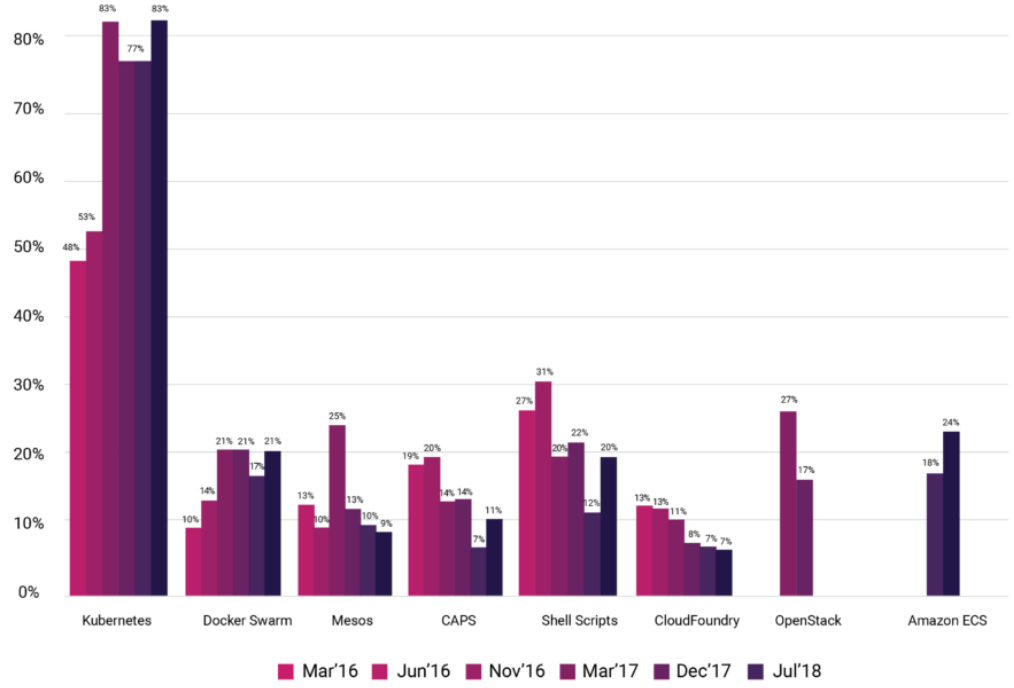
Monolithic vs Microservice



Usage Statistics



Number of containers that organizations are running

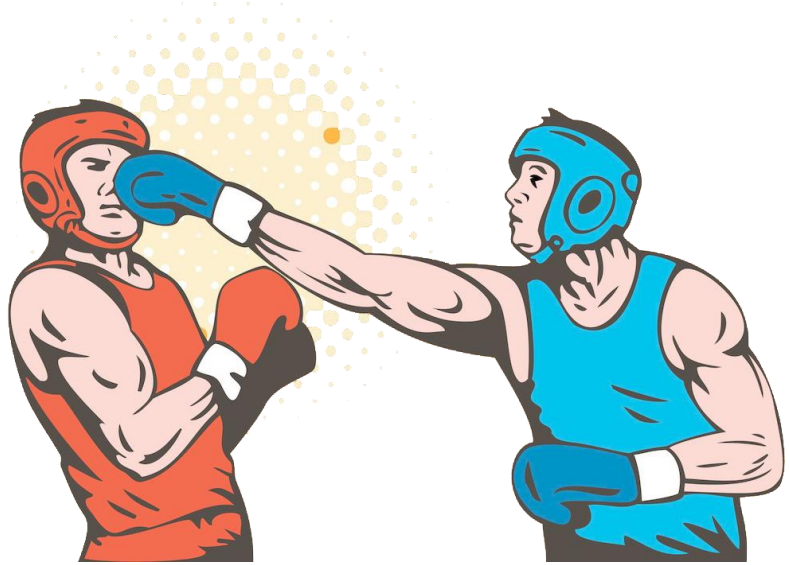


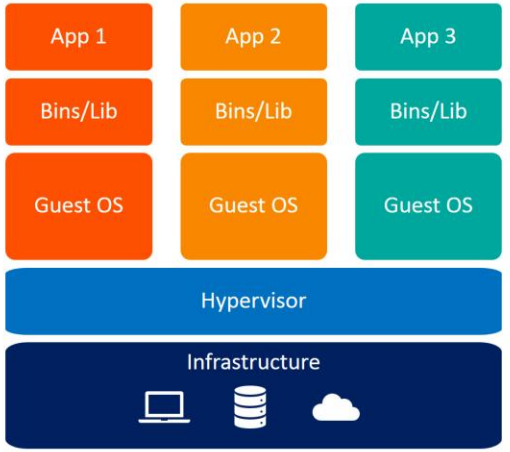
Container management tools usage

Container vs Virtual Machine

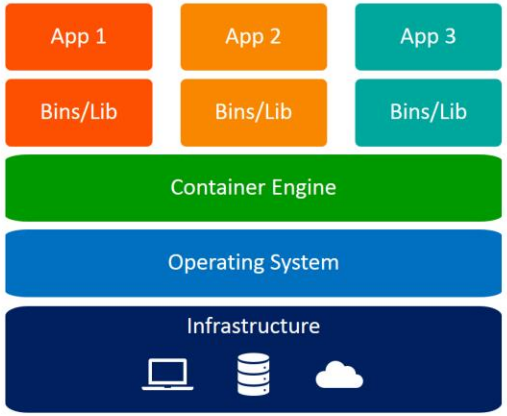
- Virtual Machine:
  - Pros:
    - OS resources available
    - Better security controls
  - Cons:
    - Consume lot of resources
    - Difficult to develop

- Containers
  - Pros:
    - Light-weight (few MB)
    - Quick react to changes (few seconds)
    - Quick to develop and deploy
  - Cons:
    - Persistent data is complicated
    - Don't run bare metal speed





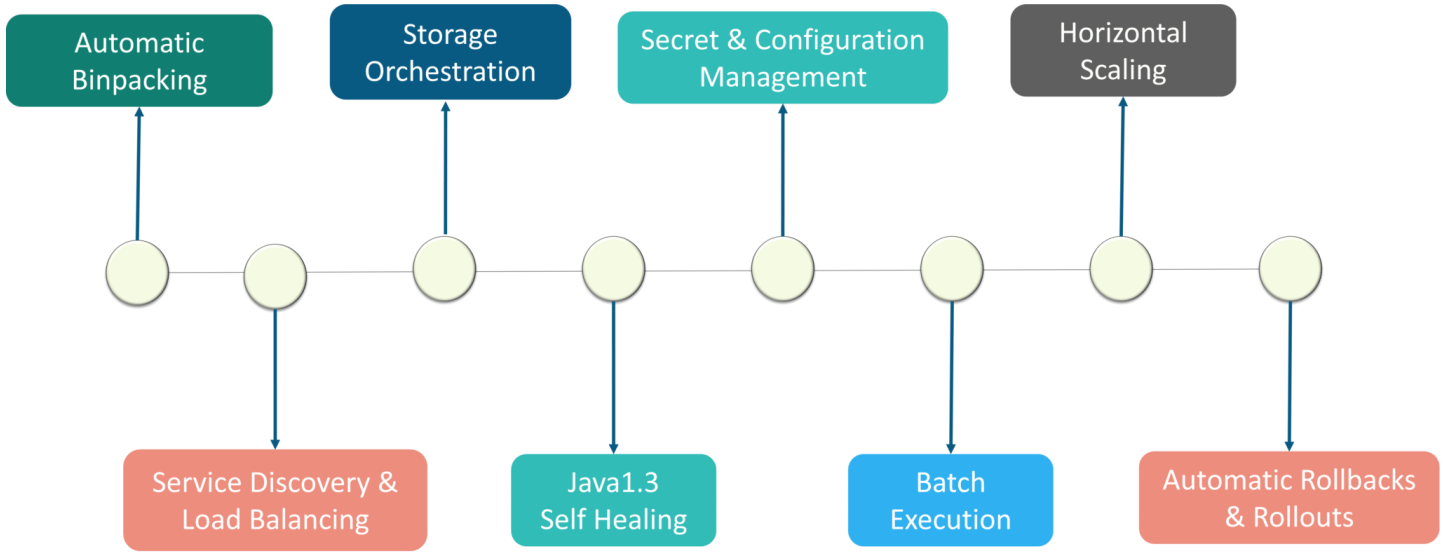
Virtual Machines



Containers

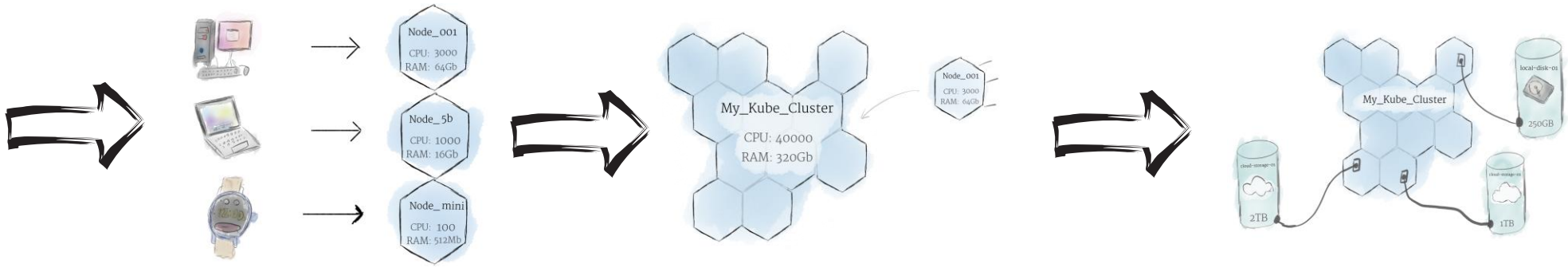
Kubernetes

Container Orchestration Platform



Fundamental Units

Hardware

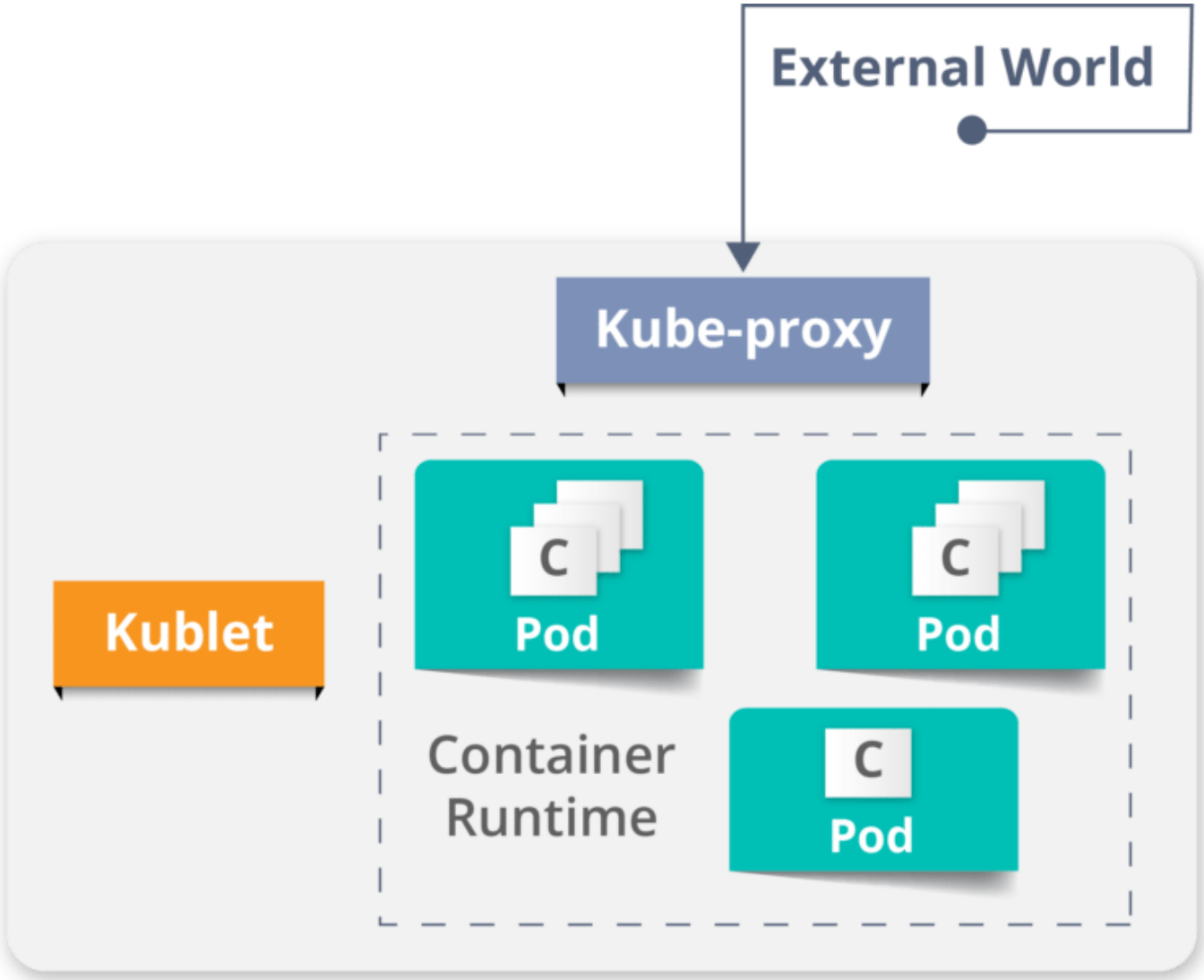


Software

Picture Source: https://medium.com/google-cloud/kubernetes-101-pods-nodes-containers-and-clusters-c1509e409e16

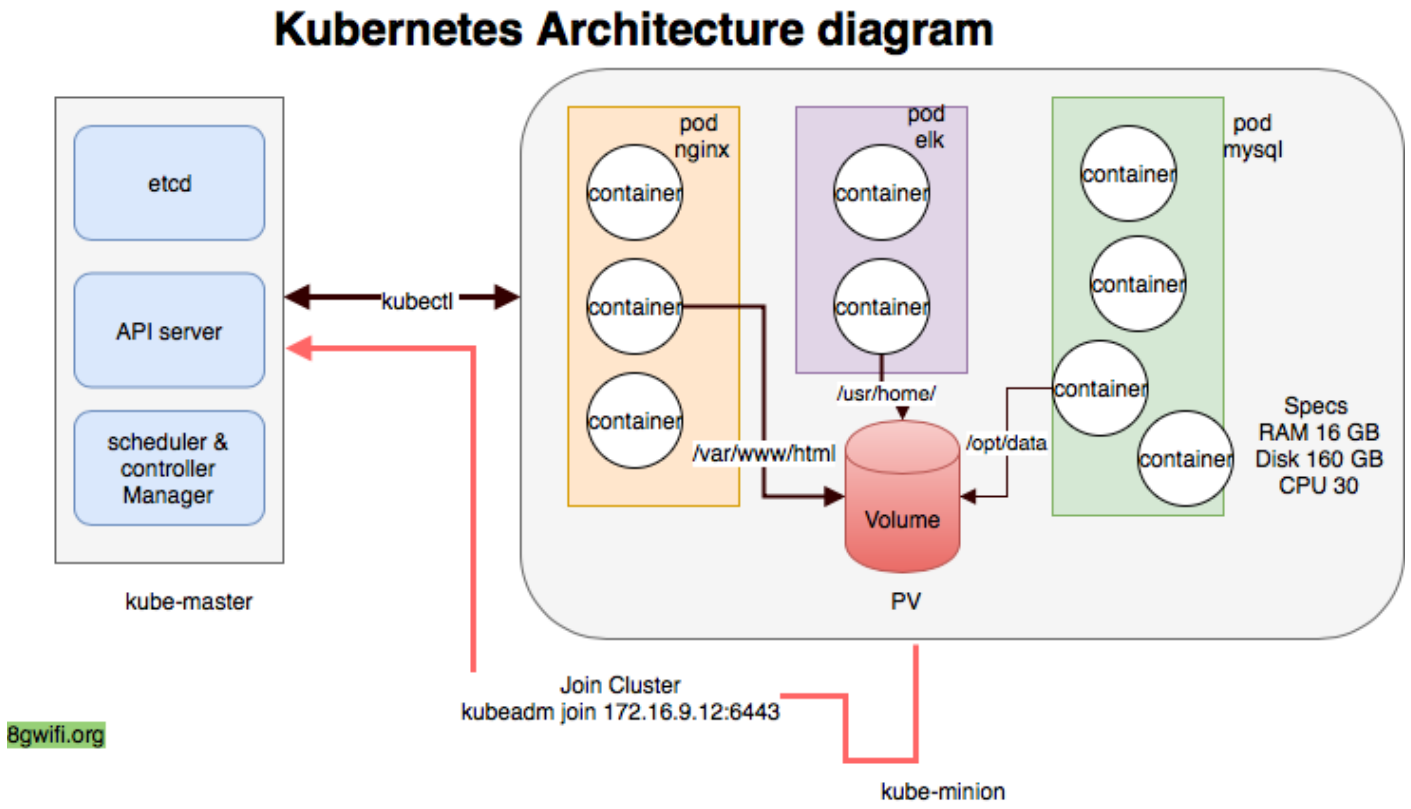
Kubernetes's Architecture

- Master – slave architecture
- Slave:
  - Contain nodes
  - **Kubelet** – a bridge to master
  - **CAdvisor** – report CPU, MEM, etc..



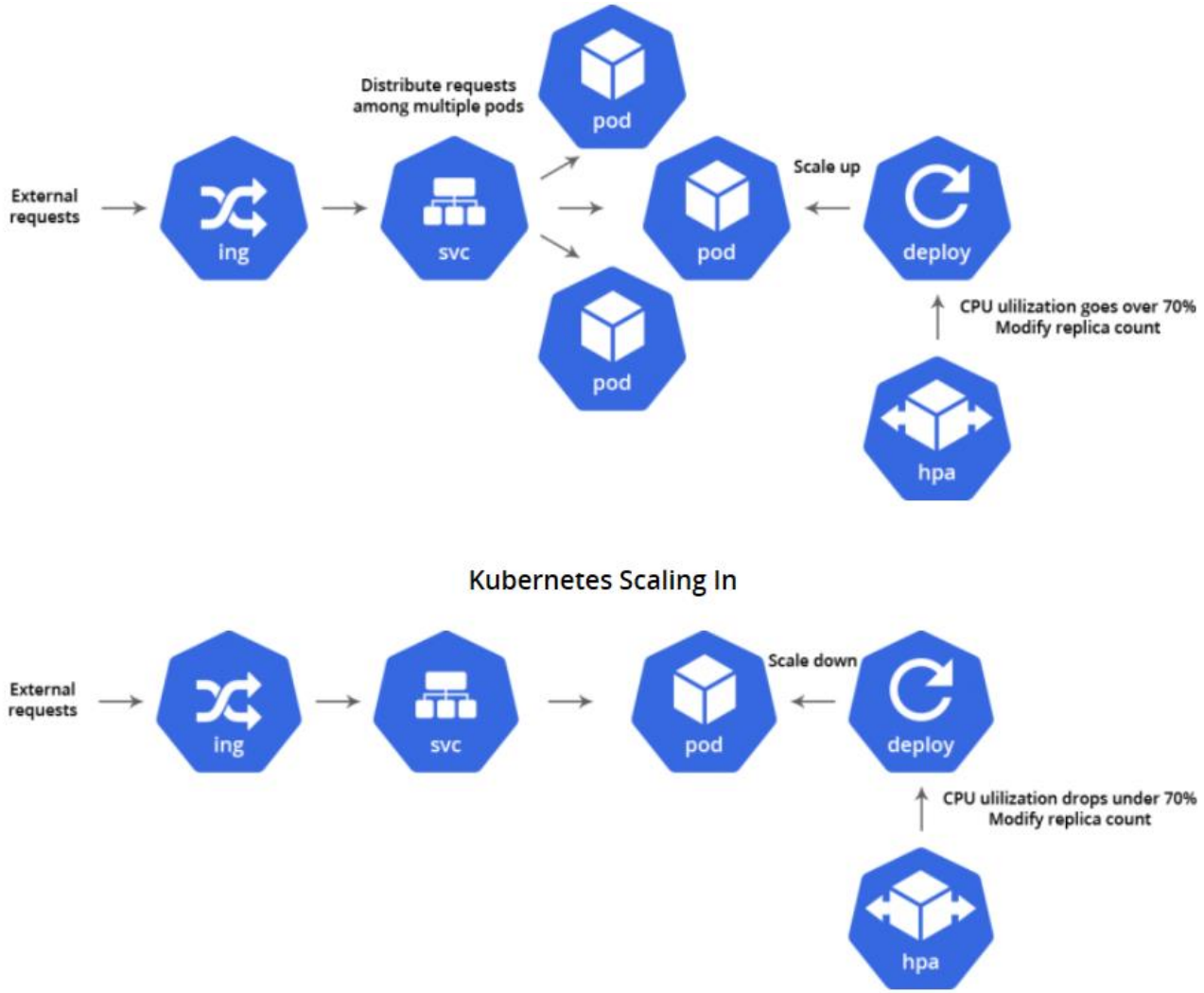
Kubernetes's Architecture

- Master:
  - **Etcd**: store cluster data
  - **Kube-apiserver**: front-end to the cluster, receives all request.
  - **Kube-controller-manager**: regulate clusters' state and perform tasks
  - **Kube-scheduler:** schedule pods

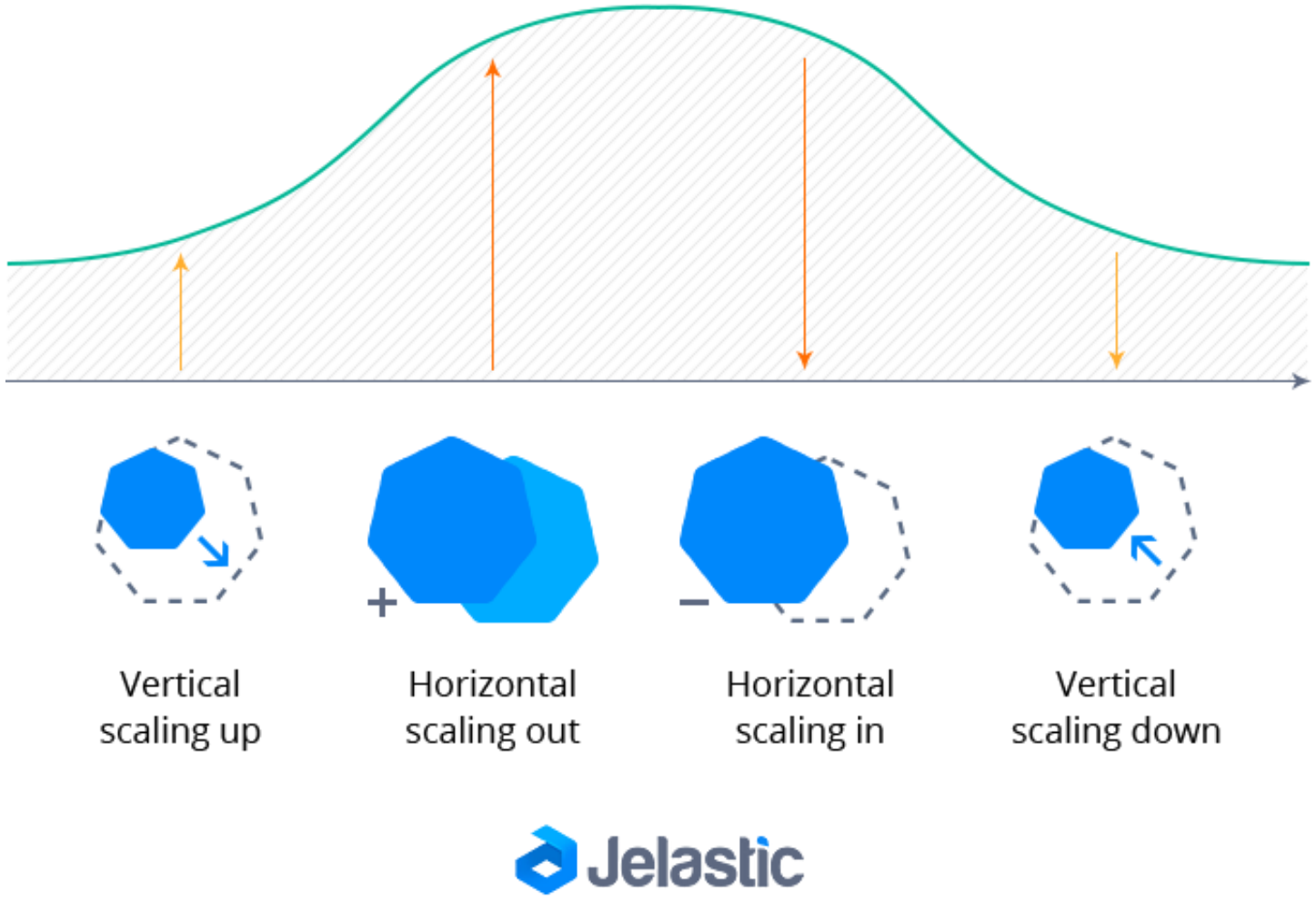


Horizontal Scaling

- Horizontal Pod Autoscaling (HPA) Changes shape of kubernete automatically in response to the workload's CPU or memory consumption, or in response to custom matrix.

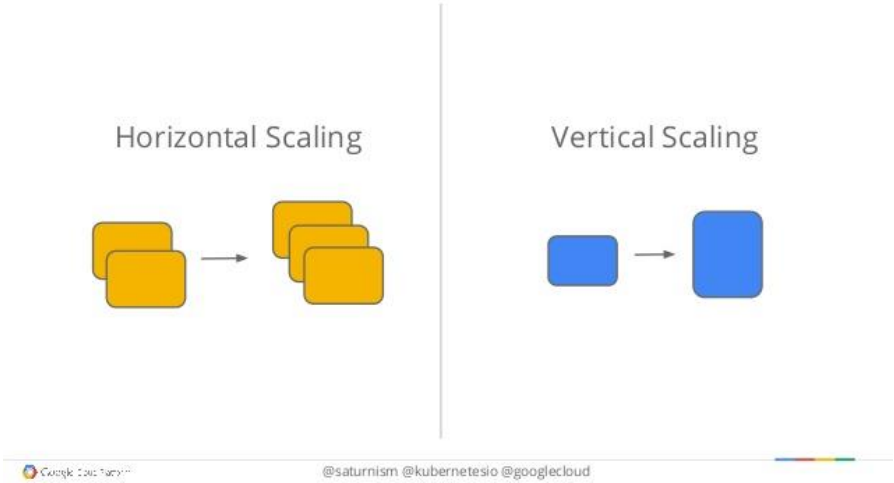


Vertical Scaling



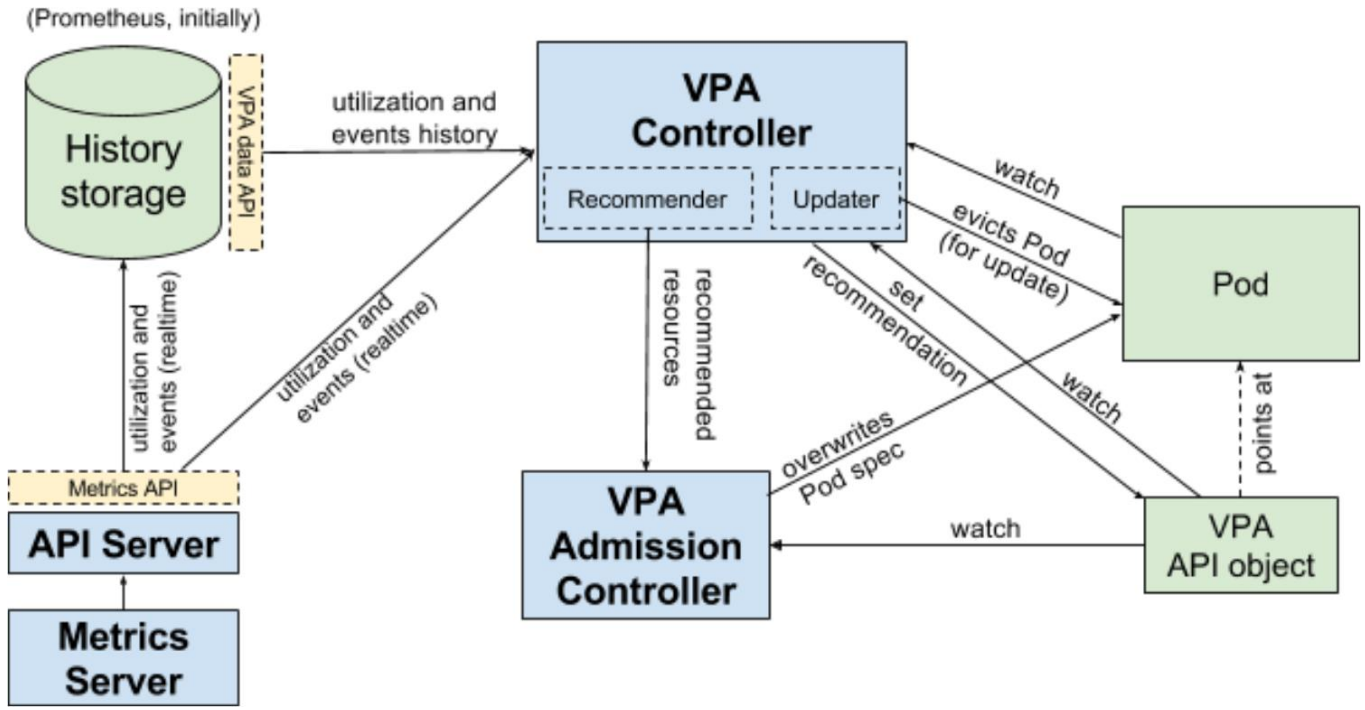
Vertical Pod Autoscaler (VPA)

- Allocate more/less CPU or memory to existing pods.
- It usually requires pod to be restarted
- It respects pods distribution budget (PDB)



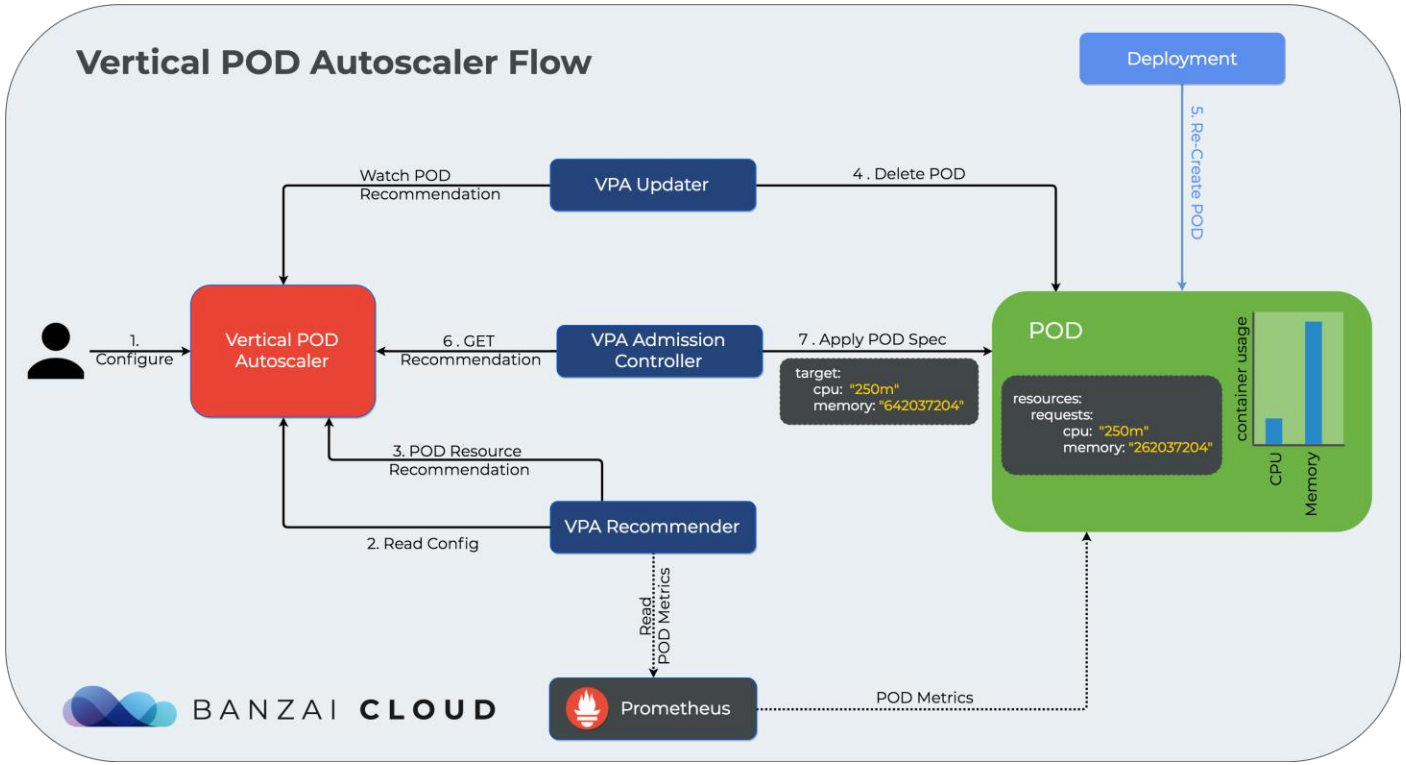
Internal Architecture

- 3 Components:
  - Recommender: monitor current and past consumption
  - Updater: check if the managed pods have the corrected resources set
  - Admission Controller: set the correct resource requests on new pods
- 3 Configurations:
  - Initial: only assign resources on creation, no change afterwards
  - Auto: (default) assign resources on creation and can update during lifetime
  - Off: never change.



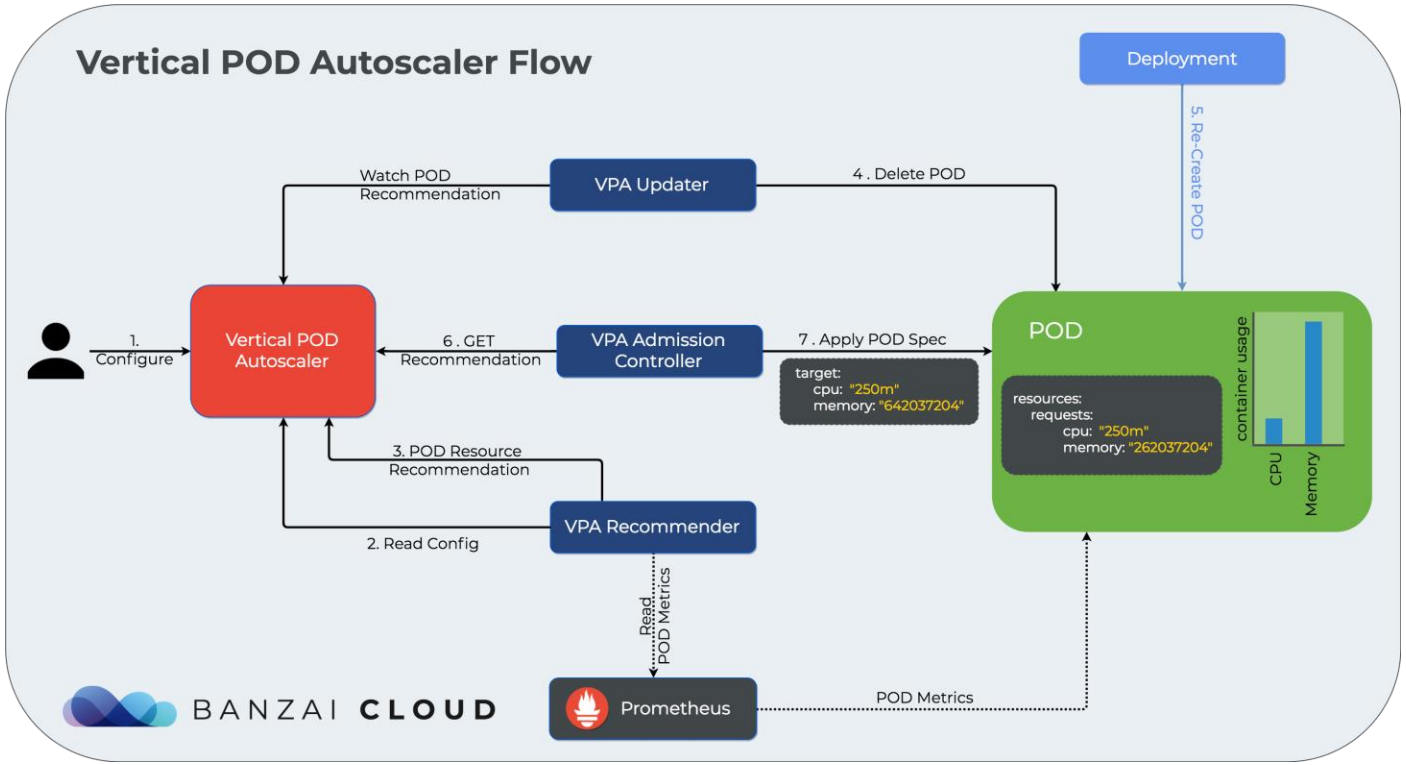
How it works

- Recommender
  - fetches historical report
  - fetches real time metric
  - computes new recommended resources and set recommendation in VPA object.
- Updater
  - fetches recommendation for pods
  - if needed, evicts pod(s)
  - disruptive. To avoid:
    - CPU starvation
    - Risk of correlated OOM across multiple pods at random time
    - Save resources



How it works

- Admission Controller
  - intercepts pod creation requests
  - It fetch VPA configuration and if pod is matched and mode is not "off", it rewrite requests.
  - Otherwise, it will leave pod unchanged

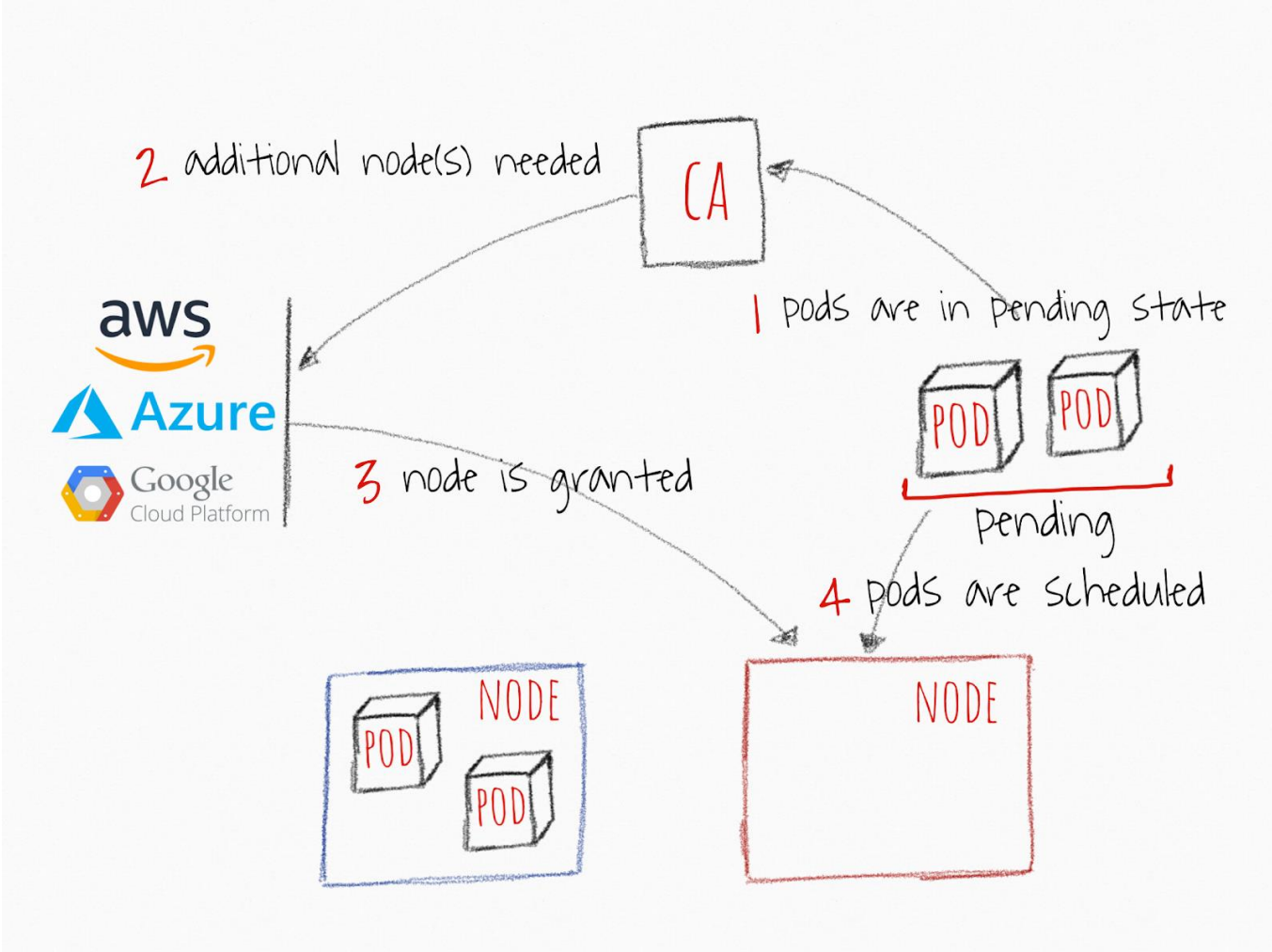


Recommended Model

- Memory and CPU consumption are
  - **independent random variables** with distribution equal to the one observed in the last N days
  - recommended value is N=8 to capture weekly peaks.
- For CPU the objective:
  - **is to keep the fraction of time when the container usage exceeds a high percentage (e.g. 95%) of request below a certain threshold (e.g. 1% of time).**
  - "CPU usage" = mean usage measured over a short interval.
  - the shorter the measurement interval = the better the quality of recommendations for spiky, latency sensitive workloads.
  - Minimum reasonable resolution is 1/min, recommended is 1/sec.
- For Memory the objective
  - **is to keep the probability of the container usage exceeding the request in a specific time window below a certain threshold** (e.g. below 1% in 24h).

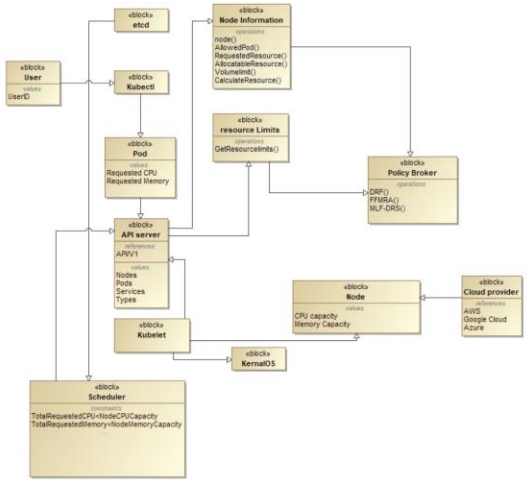
Cluster Autoscaler

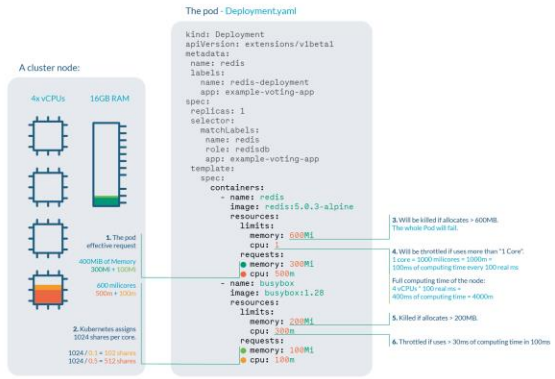
- Officially shipped in k8s 1.6 (current: 1.8)
- Run on master node
- Default deployed strategy on GCP, but support for other cloud providers.
- Automatically adjusts the size of the Kubernetes cluster based on:
  - Due to insufficient resources,
  - Underutilized nodes in the cluster that have been for an extended period of time and their pods can be placed on other existing nodes.



Future sections

- In details of pod – deployment yaml
- Autoscalers are mostly configured based on CPU and Memory and performs by threshold parameters.
- Issues:
  - Unfair scaling between pods
  - Disruptive Vertical Scaling
  - Difficult for application developers to anticipate needs and set configuration ahead of time
- Solutions:
  - Fairer metrics
  - External program to anticipate vertical scaling
  - Machine Learning





input : T samples of the set $\mathcal{M}$
output: Sorted set of eigenvectors $\mathcal{E}$ and eigenvalues Λ.

1 Construct matrix of samples X of size $T \times |\mathcal{M}|$
2 Compute covariance matrix $cov(X) = C$
3 Calculate $\mathcal{E}(C)$ and $\Lambda(C)$
4 **return** $sort(\Lambda(C))$, $\mathcal{E}(C)$