Kubernetes and GKE - Best Practices from the Field

Allan Naim
Platform Lead - Kubernetes and Container Engine
Google, Inc
anaim@google.com | allannaim@
Agenda

• Kubernetes and Google Container Engine
• Best Practices
  – Namespaces
  – Projects, Networks and Clusters
  – Resource Allocation
  – Monitoring
• Summary
Deployments and Operations is hard

**Speed**
Unable to release new products and features fast enough due to slow IT and difficulty in configuring environments.

**Scalability**
Scaling applications to match customer demand
Scaling across regions for global reach

**Hybrid**
Designing, running, and maintaining your own resilient infrastructure
Running on more than one cloud

**Time**
Too much time spent on infrastructure, and not enough building new products

**Cost**
Poor efficiency
Costs skyrocketing
Data center leases
Containers are really hot these days

**Benefits**

<table>
<thead>
<tr>
<th><strong>Immutable infrastructure</strong></th>
<th><strong>Isolation</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Faster deployment times</strong></td>
<td><strong>Portability</strong></td>
</tr>
<tr>
<td><strong>Reusability</strong></td>
<td><strong>Introspection</strong></td>
</tr>
<tr>
<td><strong>Versioning</strong></td>
<td><strong>Ease of sharing</strong></td>
</tr>
</tbody>
</table>
New Set of Challenges

“Where should I run my containers?”

“If we run our containers on VMs, I don’t want to manage anything”

“We don’t want to be locked into one cloud provider”

“How do I get my containers to talk to one another?”

“How do we ensure our containers are running smoothly?”
It just so happens that Google knows a few things about running containers at scale!
Enabled Google to grow our fleet over 10x faster than we grew our ops team.
Kubernetes

Open Source, Run Anywhere, Container-centric Infrastructure, for managing containerized applications across multiple hosts.
- Auto-scaling, rolling upgrades, A/B testing ...
- Handles **container and node failures**
- Multiple **cloud and bare-metal environments**
- 100% Open Source, written in Go

Inspired by Google’s experiences and internal systems ([blog post, research paper](http://example.com))
What is Kubernetes, really?

- **nodes and node pools** — the machines hosting the cluster, each pool run on homogeneous hardware
  - **pods** — a small group of tightly coupled containers, sharing the same IP, disk, etc.

- **deployments** — provides declarative updates and implement changes at a controlled rate
  - **replica sets / stateful sets** — a group of stateless or stateful pods with a desired state
  - **services** — a load-balanced set of pods with a stable IP, allowing service discovery by DNS name

- **labels** — identifies and organizes pods, e.g. which pods belong to a particular service

```yaml
apiVersion: v1
kind: Pod
metadata:
  name: redis-django
labels:
  app: webapp
spec:
  containers:
    - name: key-value-store
      image: redis
      ports:
        - containerPort: 6379
    - name: frontend
      image: django
      ports:
        - containerPort: 8000
```
Google Container Engine

Pure Kubernetes. Easy, Powerful and Google Supported

Performance, Scale and Efficiency of Google Cloud Platform

Manage Applications - NOT Machines!
Benefits of Google Container Engine

- Built on industry-leading performance differentiators (30 sec boot times, global HTTP load balancer with instant >1M QPS)
- Runs in all our regions
- Sustained use discounts and per-minute billing
- Use with Preemptible VMs, custom machine types, GPUs (Alpha) and mix and match node shapes with NodePools.
- Scale pods within a cluster, or scale the number of instances in your cluster
- Reduces operational burden - Google manages and upgrades the masters, ensures cluster health, node auto-upgrade, node repair
Best Practices: Use Namespaces

Problem: I have too much stuff!
• name collisions in the API
• poor isolation between users
• don’t want to expose things like Secrets

Solution: Slice up the cluster
• create new Namespaces as needed
  • per-user, per-app, per-department, etc.
• Namespaces are just another API object
• One-step cleanup - delete the Namespace
• Obvious hook for policy enforcement (e.g. quota)
Problem Statement: We have many app teams that want to deploy containerized workloads. How do we structure our projects, networks and clusters?
Mapping Clusters to Teams

Stage: proof of concept

Start with a single cluster
Everyone has access to this cluster
No real access management
Additional teams want in on the action

Admin

Is a single cluster enough?

Dev

I need a prod environment

Dev

What about our app?

Dev

Shared Cluster?
How do I set up my clusters to support multiple environments and multiple teams?

- Single project, single cluster, multiple namespaces
- Single project, multiple clusters, multiple namespaces
- Multiple projects, multiple clusters, multiple namespaces
Option 1

Shared project + shared cluster + different namespace

Pros:
- Easier management
- Potential cost savings
- Resource isolation via quota
- RBAC based object isolation

Cons:
- Managing quotas for each namespace
- Resource isolation limited to CPU and memory

![Diagram showing applications in different namespaces across projects and clusters.](image-url)
Option 2

Shared project + different cluster + different namespace

Pros:
- Improved resource isolation (no sharing of nodes/VMs/DNS)
- Separate networks (isolation)
- Limit scope of DoS on control plane
- Assign different availability characteristics for your environments:
  - cluster spread across zones

Cons:
- Lose management benefits
Option 3

Different project + different cluster + different namespace

Project Staging
- Cluster Blue Team
  - Namespace Blue App-1
  - Namespace Blue App-2

Project Prod
- Cluster Green Team
  - Namespace Green App-1
  - Namespace Green App-2

Pros:
- Greater isolation
- Separate billing domains
- Greater control over IAM domains

Cons:
- Lose management benefits
Suggested Best Practice across your Environments

Different project for Prod + shared cluster + different namespace
Suggested Best Practice for Production

- **Project Prod**
- **Network Prod**
  - HTTP(S) Load Balancer
  - Region: us-west
    - Subnetwork: prod-west
      - us-west1-a
      - us-west1-b
      - GKE Cluster West Node Pool
  - Region: us-east
    - Subnetwork: prod-east
      - us-east1-a
      - us-east1-b
      - GKE Cluster East Node Pool
Problem Statement: *How do I achieve high density while maintaining my quality of service?*
Each namespace can be assigned a resource quota (maximum request and limit across all pods)

- applies to each type of resource (CPU, mem, # of pods)
- ensure no user/app/department abuses the cluster
- applied at admission time
**Best Practice: Monitoring to Auto Everything**

**Initial resources** - sets Request based on observed historical CPU/memory usage of the container

**Horizontal Pod autoscaling** - change # of pod replicas based on CPU usage & app metrics (alpha)

**Cluster autoscaling**
- Add nodes when needed  
  e.g. CPU usage too high
- Remove nodes when not needed  
  e.g. CPU usage too low
Putting it all together

- Could use projects to encapsulate your various operating environments
  - Separate Dev/Test/Staging from Prod in their own project
- Could run Dev/Test/Staging in their own respective cluster
- Kube namespaces as team boundary within cluster
- Leverage Groups for the application of roles to specific project environments
- Kube namespace resource quotas to avoid user/app/department abuses the cluster
- Set appropriate Pod (requests, limits) per QoS requirements (Guaranteed, Burstable, Best Effort)
Useful Resources

- Learn more about Kubernetes
  - [http://kubernetes.io](http://kubernetes.io)
  - [https://github.com/kubernetes/kubernetes](https://github.com/kubernetes/kubernetes)
  - [https://codelabs.developers.google.com/](https://codelabs.developers.google.com/), search ‘kubernetes’
  - [https://kubernetesbootcamp.github.io/kubernetes-bootcamp/](https://kubernetesbootcamp.github.io/kubernetes-bootcamp/)

- Learn more about Container Engine
  - [https://cloud.google.com/container-engine/](https://cloud.google.com/container-engine/)
  - [Preparing Container Engine for Production](https://cloud.google.com/container-engine/)

- Get help
  - ‘kubernetes’ and ‘google-container-engine’ tags on [http://stackoverflow.com](http://stackoverflow.com)
  - [http://slack.k8s.io/](http://slack.k8s.io/)
Become Kubernetes Certified

Certified Kubernetes Administrator (CKA)