Google Cloud Platform

Google Container Engine

Google Cloud Platform Fundamentals

V2.1

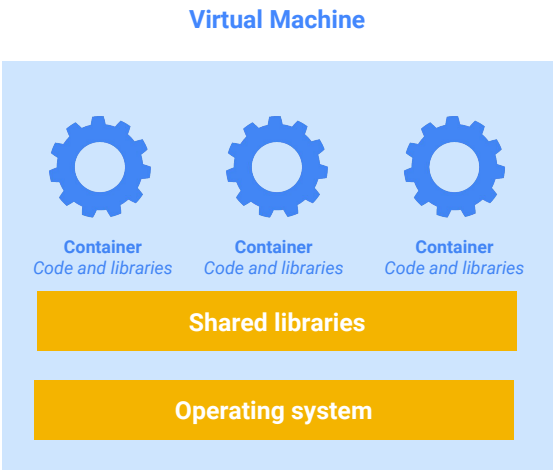
Timing: Approximately 30 minutes

Agenda

1. Introduction to Containers
2. Kubernetes
3. Google Container Engine
4. Quiz & Lab

What is a Container?

- Virtualization at the operating system layer
- Separates operating system from application code and dependencies
- Isolates individual processes
- Popular implementations include Docker and rkt



Why Use Containers?

- Support **consistency** across **development**, **testing**, and **production** environments
- **Loose coupling** between application and operating system layers
- Much simpler to **migrate workloads** between on-premises and cloud environments
- Support **agile** development and operations

Notes:
So what does a container provide that a VM does not?

- Simple deployment: By packaging your application as a singularly addressable, registry-stored, one-command-line deployable component, a container radically simplifies the deployment of your app no matter where you're deploying it.
- Rapid availability: By abstracting just the OS rather than the whole physical computer, this package can "boot" in ~1/20th of a second compared to a minute or so for a modern VM.
- Leverage microservices: Containers allow developers and operators to further subdivide compute resources. If a micro VM instance seems like overkill for your app, or if scaling an entire VM at a time seems like a big step function, containers will make a big, positive impact in your systems.

What are the implications of some of these advantages?

- An obvious one is that a developer has in their laptop plenty of compute power to run multiple containers, making for easier and faster development. While it is certainly possible to run several virtual machines on a laptop, it's far from fast, easy, or lightweight.
- Similarly, release administration is easier: pushing a new version of a container is a single command. Testing is cheaper as well: in a public

- cloud where a VM is billed for a minimum of 10 minutes of compute time (or, gasp, a whole hour?!), that means a single test might only cost you a small amount. But if you're running thousands of programmatically driven tests per day, this starts to add up. With a container, you could do thousands of simple tests at the same cost, amounting to large savings for your production applications.
- Another implication is the composability of application systems using this model, especially with applications using open source software. While it might be a daunting systems administration (not to mention pronunciation) task for a developer to install and configure MySQL, memcached, MongoDB, Hadoop, GlusterFS, RabbitMQ, node.js, nginx, and so on together on a single box to provide a platform for their application, it is much easier and vastly lower risk to start a few containers housing these applications with some very compact scripting. Consider the amount of error-prone, specialized, boilerplate work this model eliminates. Add to this the public registration of canonical implementations for these types of core building blocks, and you have the beginnings of a real ecosystem for quality components.

Agenda

1. Introduction to Containers
2. Kubernetes
3. Google Container Engine
4. Quiz & Lab

Kubernetes ('k8s')

- Open source container cluster orchestration system
  - Automates deployment, scaling, and operations for container clusters
- Based on Google's experience over 10+ years
- Built for a multi-cloud world:
  - Public, private, hybrid

Notes:
Google has built a cluster management, networking, and naming system — the first version of which was called Borg and its successor, called Omega — to allow container technology to operate at Google scale. We now start around seven thousand new containers per second, or over two billion per week.
We've recently applied our operational experience and technical depth around containers to create Kubernetes (sometimes shortened to "K8s" on public forums) to the world.

Kubernetes has created a layer of abstraction that allows the developer and administrator to work collectively on improving the behavior and performance of the desired service, rather than any of its individual component containers or infrastructure resources.

What does a Kubernetes cluster provide that a single container does not? By refocusing control at the service level rather than the container level, Kubernetes allows for intelligent, active management of the service as a whole. A service which scales, self-heals, and is easily updated is another of those "Good Ideas". For example, at Google we apply machine learning techniques to ensure that a given service is operating at top efficiency.

The core design principles of Kubernetes are:

- Meet users where they are: rewriting apps is not an option
- Coupling is bad: independent parts last longer
- Declarative > imperative: self-healing is critical
- Open > closed: open source and open standards
- Modularity == longevity: interchangeable and replaceable components
- Portability: lock-in is bad for business

These principles allow you to focus on applications, not infrastructure.

Features of Kubernetes (1 of 2)

- *Workload portability*
  - Run in many environments, across cloud providers
  - Implementation is open and modular
- *Rolling updates*
  - Upgrade application with zero downtime
- *Autoscaling*
  - Automatically adapt to changes in workload

Notes:
To support the design goal of portability:

- Kubernetes 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
- You can build your apps on-premises, then “lift-and-shift” into cloud when you are ready
- The product is cloud vendor agnostic - Kubernetes doesn’t force apps to know about concepts that are cloud-provider-specific, such as:
  - Network model
  - Ingress/egress
  - Load balancers
  - Persistent volumes

For more information on rolling updates in Kubernetes, see: http://kubernetes.io/docs/user-guide/rolling-updates/.

For more information on autoscaling in Kubernetes, see: http://kubernetes.io/docs/user-guide/horizontal-pod-autoscaling/.

Features of Kubernetes (2 of 2)

- *Persistent storage*
  - Abstracts details of how storage is provided from how it is consumed
- *Multi-zone clusters*
  - Run a single cluster in multiple zones
- *Load balancing*
  - External IP address routes traffic to correct port

Notes:
For more information on persistent storage in Kubernetes, see: http://kubernetes.io/docs/user-guide/persistent-volumes/.

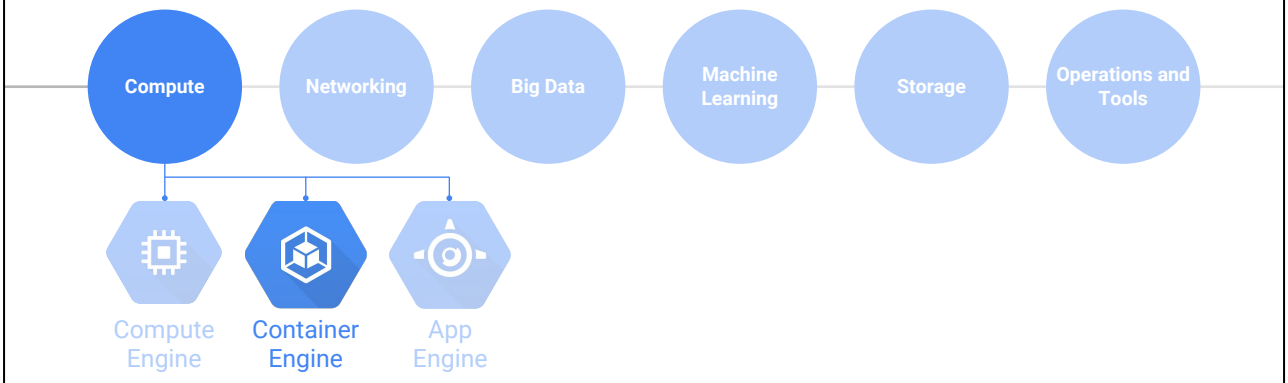
For more information on multi-zone clusters in Kubernetes, see: http://kubernetes.io/docs/admin/multiple-zones/.

For more information on load balancing in Kubernetes, see: http://kubernetes.io/docs/user-guide/load-balancer/.

Agenda

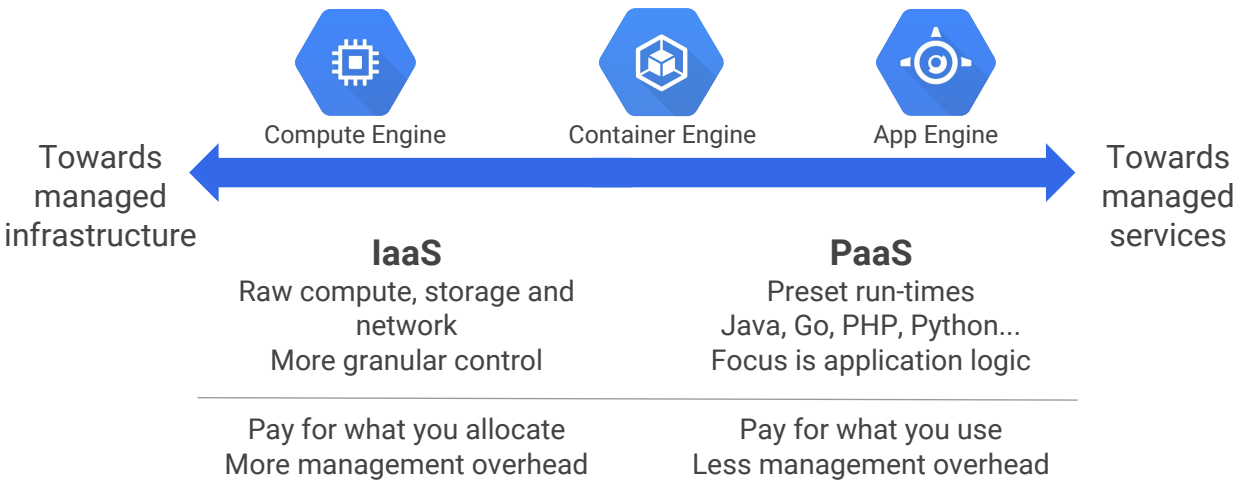
1. Introduction to Containers
2. Kubernetes
3. Google Container Engine
4. Quiz & Lab

Google Cloud Platform



Notes:
Container Engine is one of several Google Cloud Platform compute options for running your applications.

IaaS and PaaS



Notes:
This slide highlights the shift in cloud computing. Virtualized datacenters have made infrastructure as a service (IaaS) accessible to both large and small organisations. At Google, we aim to offer services that allow you the flexibility to incorporate existing practices that are currently popular, while supporting a transition to building what's next, by adopting agile working practices and technologies.

As cloud computing has evolved, the momentum has shifted toward increasingly automated operations, and closer collaboration between developers and operations professionals, as well as automating operations away entirely. Google Cloud Platform allows you to delegate management of various aspects of your computing resource needs by selectively adopting our managed services.

Google Container Engine is an example of the shift toward managed services. Container Engine is a powerful cluster manager and orchestration system for running your Docker containers. Container Engine schedules your containers into the cluster and manages them automatically based on requirements you define (such as CPU and memory). It's built on the open source Kubernetes system, giving you the flexibility to take advantage of on-premises, hybrid, or public cloud infrastructure.

Container Engine provides you with container management and orchestration while simultaneously managing Compute Engine infrastructure needed to run your applications. As such, Container Engine gives you the best of both worlds: a managed container service coupled with managed infrastructure. With Container Engine, you can create a managed cluster that’s ready for container deployment, in just a few clicks. Container Engine is fully managed by Google reliability engineers, so you don’t have to worry about cluster availability or software updates. Container Engine also makes application management easier. Your cluster is equipped with common capabilities, such as logging and container health checking, to give you insight into how your application is running. And, as your application’s needs change, resizing your cluster with more CPU or memory is easy.

Google Container Engine (1 of 2)

- Fully managed cluster management and orchestration system for running containers
  - Based on Kubernetes
  - Uses Compute Engine instances and resources
- Complimentary services:
  - *Google Cloud Container Builder* - Create Docker container images from app code in Google Cloud Storage
  - *Google Container Registry* - Secure, private Docker image storage

Notes:
Google Cloud Container Builder lets you create Docker container images from application source code located in Google Cloud Storage. Container images created by Container Builder are automatically stored in Google Container Registry. You can deploy the container images you create on Google Container Engine, Google Compute Engine, Google App Engine Managed VMs or other services where you can run applications from Docker containers. Container Builder uses Google Cloud Logging to store and manage build logs.

Google Container Registry provides secure, private Docker image storage on Google Cloud Platform. While Docker provides a central registry to store public images, you may not want your images to be accessible to the world. In this case, you must use a private registry. The Google Container Registry runs on Google Cloud Platform, so can be relied upon for consistent uptime and security. The registry can be accessed through an HTTPS endpoint, so you can pull images from any machine, whether it's a Google Compute Engine instance or your own hardware.

The Google Container Registry only charges for the Google Cloud Storage storage and network egress consumed by your Docker images.

Google Container Engine (2 of 2)

- Uses a declarative syntax to manage applications
  - Declare desired application configuration, Container Engine implements, manages
- Decouples operational, development concerns
- Manages and maintains
  - Logging, health management, monitoring
- Easily update Kubernetes versions as they are released

"Our platform sometimes has to be deployed on a cluster. How do we enable containers to communicate from different hosts? Google has the answer: Kubernetes. This awesome tool helps us manage our clusters of containers as if they were a single system."

scale

Docker containers automate scalability

speed

REST APIs speed provisioning of new instances; JAVA applications can be deployed in minutes

-30%

Administrative costs reduced by 30%

Notes:
Read more about the Treeptik story here:
https://cloud.google.com/customers/treeptik/.

Deploying Apps: Container Engine vs App Engine

	Container Engine	App Engine Standard	App Engine Flexible
Language support	Any	Java, Python, Go & PHP	Any
Service model	Hybrid	PaaS	PaaS
Primary use case	Container-based workloads	Web and mobile applications	Web and mobile applications, container-based workloads

Agenda

1. Introduction to Containers
2. Kubernetes
3. Google Container Engine
4. Quiz & Lab

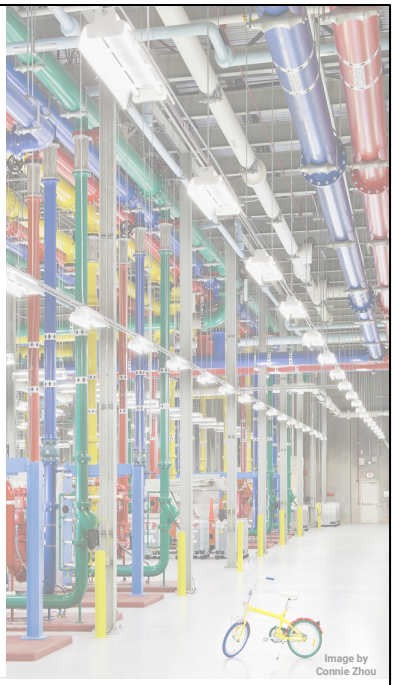
Quiz

1. Name two reasons for deploying applications using containers.
2. *True or False*: Kubernetes allows you to manage container clusters in multiple cloud providers.
3. *True or False*: Google Cloud Platform provides a secure, high-speed container image storage service for use with Container Engine.

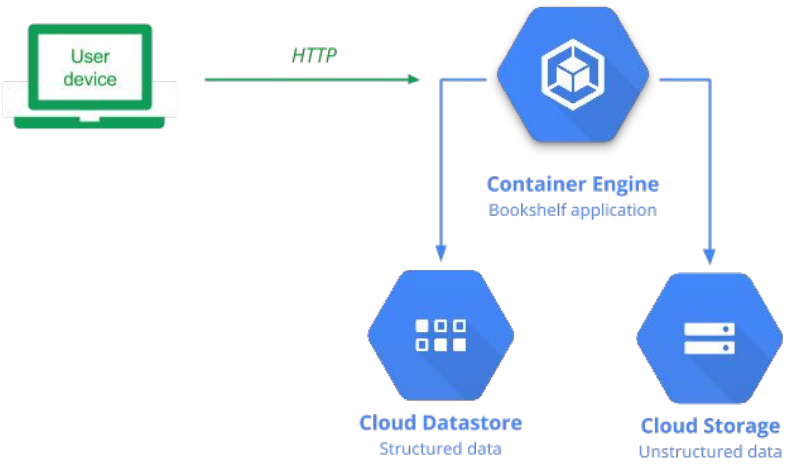
Lab (1 of 2)

Deploy the Bookshelf application to Container Engine.

1. Create a Container Engine cluster
2. Build and push a Bookshelf image to Google Container Registry
3. Use the kubectl command utility to deploy the Bookshelf container
4. Test the Bookshelf application in your browser



Lab (2 of 2)



Notes:
In this lab, you use create a Google Container Engine cluster and deploy the Bookshelf application to it using the Kubernetes command-line tool, kubectl. You continue to use existing storage services you used earlier in the course, including Google Cloud Datastore, and Google Cloud Storage.

Resources

- Container Engine Overview
  https://cloud.google.com/container-engine/
- Container Engine tutorials
  https://cloud.google.com/container-engine/docs/tutorials
- Kubernetes
  http://kubernetes.io/

Quiz Answers

1. Name two reasons for deploying applications using containers.

   Answer: Consistency across development, testing, production environments; Simpler to migrate workloads; Loose coupling; Agility

2. *True*: Kubernetes allows you to manage container clusters in multiple cloud providers.

3. *True*: Google Cloud Platform provides a secure, high-speed container image storage service for use with Container Engine.


cloud.google.com