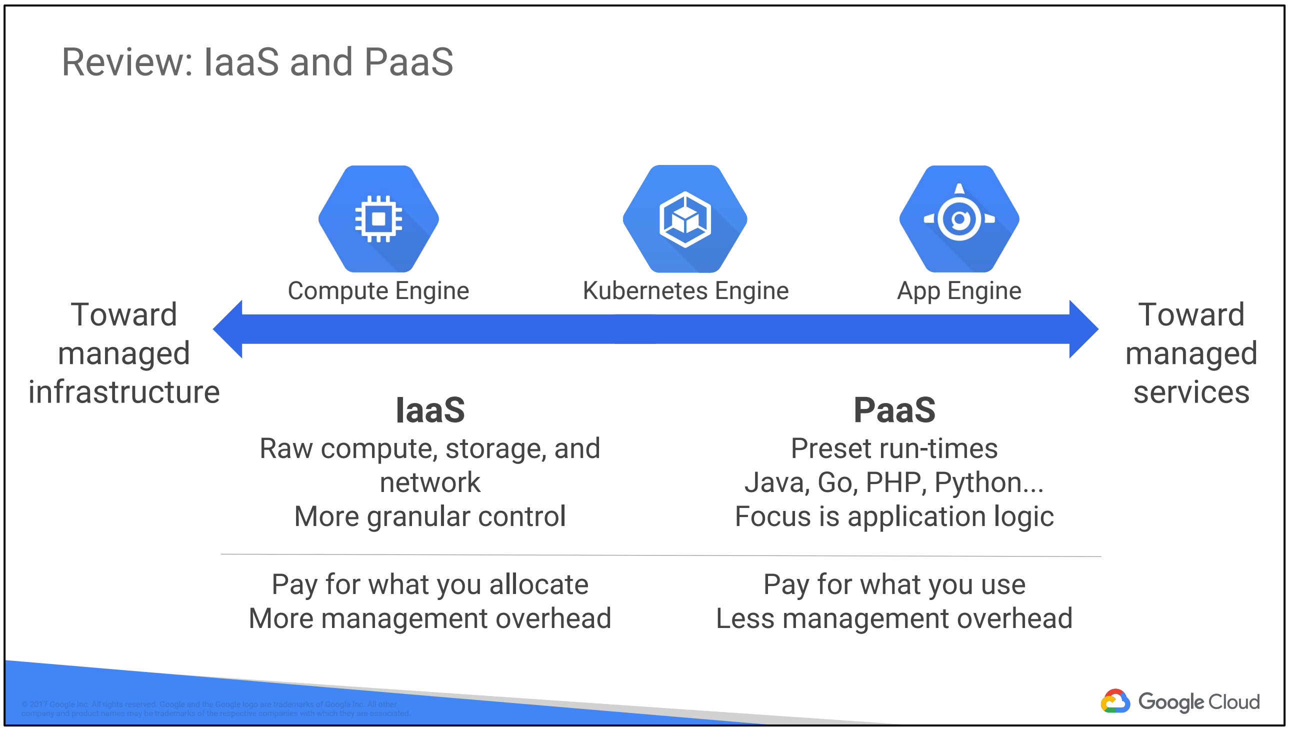
Containers in the Cloud

GCP Fundamentals: Core Infrastructure

QWIKLABS Getting Started with Kubernetes Engine

Google Cloud

Last modified 2018-08-13



Review: IaaS and PaaS

IaaS
Raw compute, storage, and network
More granular control

PaaS
Preset run-times
Java, Go, PHP, Python...
Focus is application logic

Pay for what you allocate
More management overhead

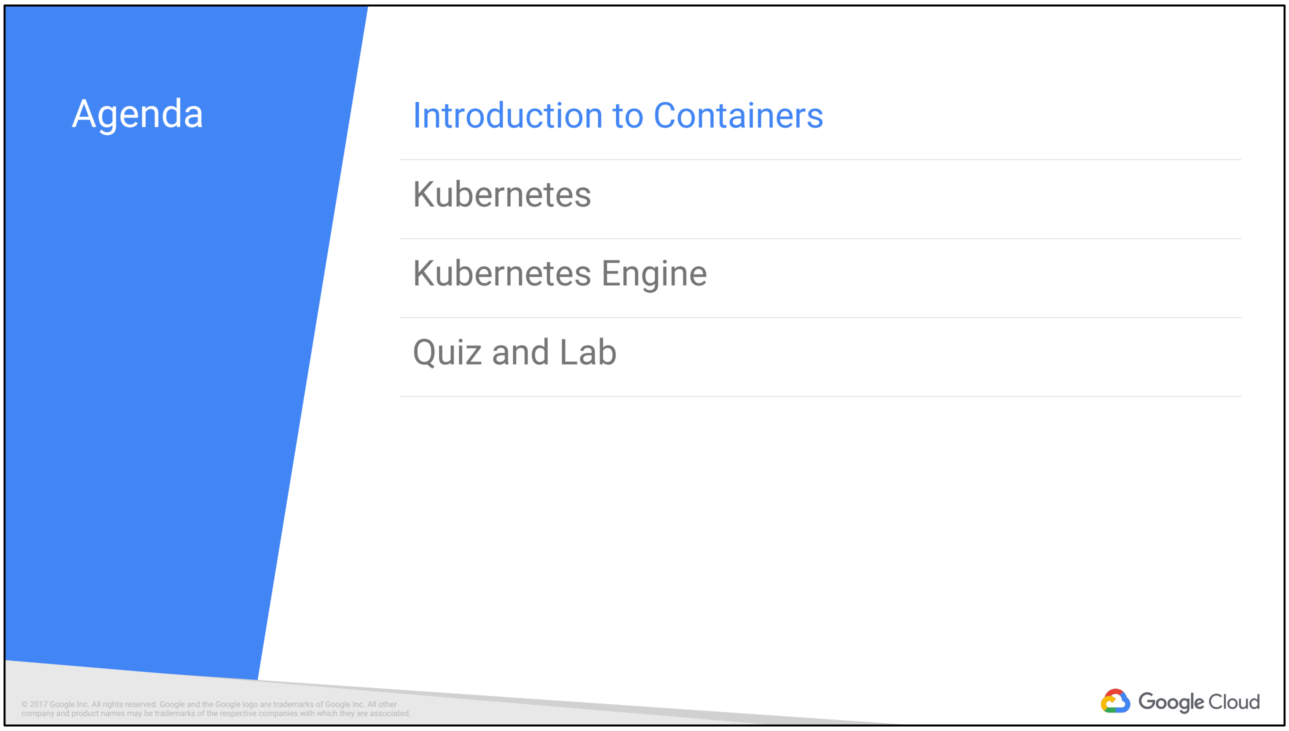
Pay for what you use
Less management overhead

Google Cloud

We've already discussed the spectrum between Infrastructure as a Service and Platform as a Service. And we've already discussed Compute Engine, which is GCP's pure Infrastructure-as-a-Service offering. In this module, we'll see how Kubernetes Engine is a kind of hybrid between IaaS and PaaS, offering the managed infrastructure of an IaaS offering with the developer orientation of a PaaS offering.

Another choice for organizing your compute is using containers rather than virtual machines, and using Kubernetes Engine to manage them. Containers are simple and interoperable, and they enable seamless, fine-grained scaling. Google launches billions of containers every week.

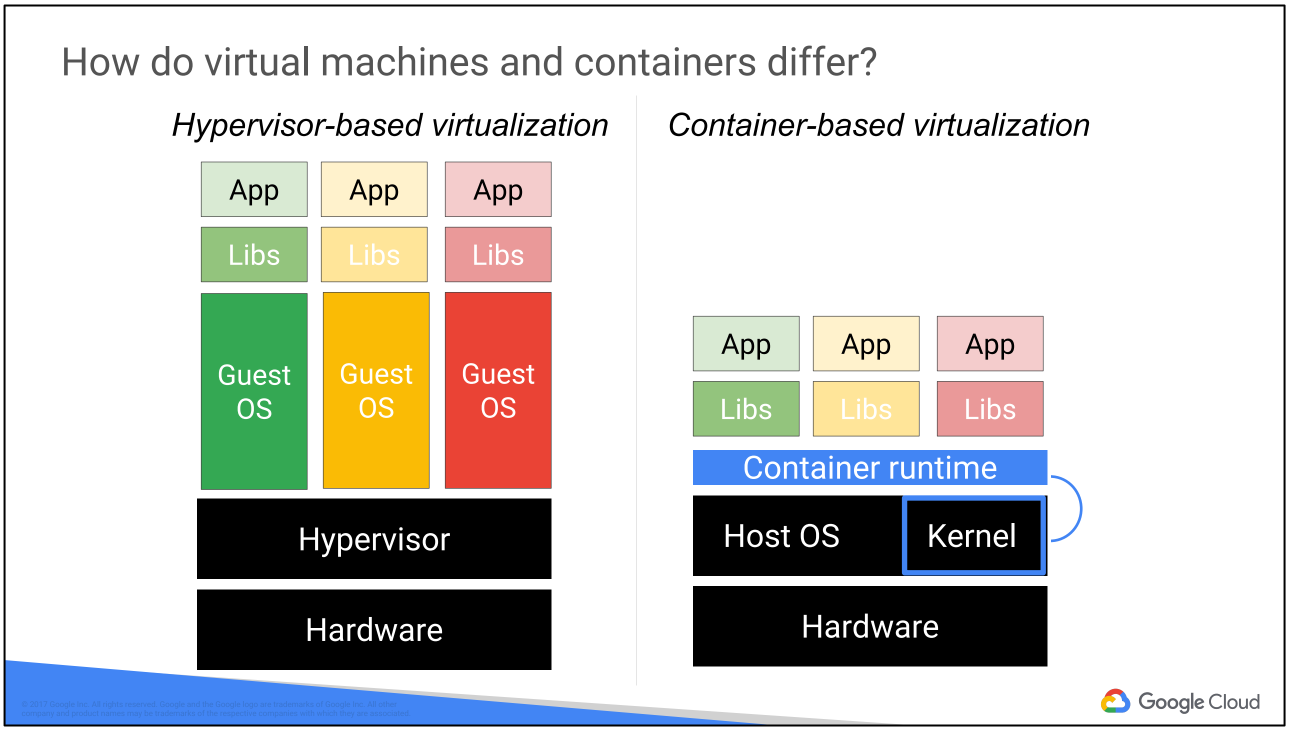
Let's learn about how Kubernetes Engine works and how it helps us deploy and manage applications in containers.



Agenda

- Introduction to Containers
- Kubernetes
- Kubernetes Engine
- Quiz and Lab

Google Cloud



How do virtual machines and containers differ?

Google Cloud

Container-based virtualization is an alternative to hardware virtualization, as in traditional virtual machines. Virtual machines are isolated from one another in part by having each virtual machine have its own instance of the operating system. But operating systems can be slow to boot and can be resource-heavy. Containers respond to these problems by using modern operating systems' built-in capabilities to isolate environments from one another.

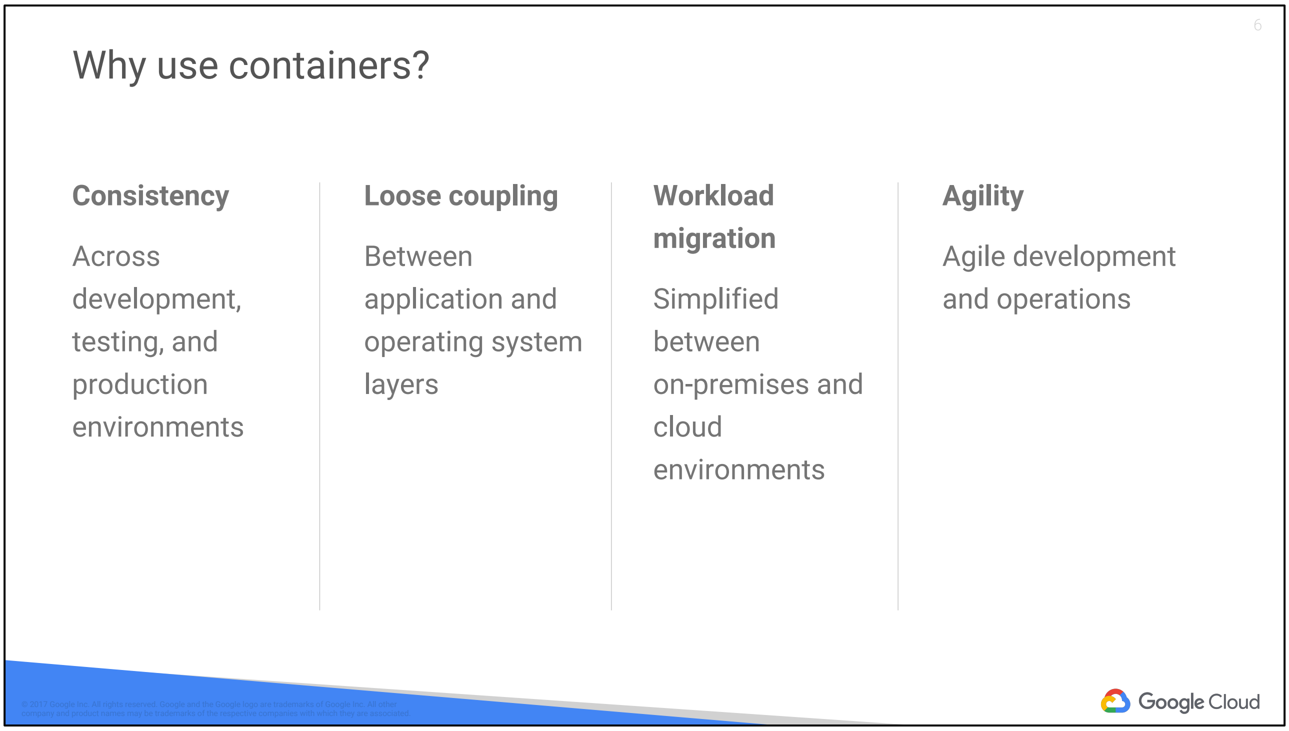
A process is a running program. In Linux and Windows, the memory address spaces of running processes have long been isolated from one another. Popular implementations of software containers build on this isolation. They take advantage of additional operating-system features that give processes the capability to have their own namespaces and that give a supervisor process the ability to limit other processes' access to resources.

Containers start much faster than virtual machines and use fewer resources, because each container does not have its own instance of the operating system. Instead, developers configure each container with a minimal set of software libraries to do the job. A lightweight container runtime does the plumbing jobs needed to allow that container to launch and run, calling into the kernel as necessary. The container runtime also determines the image format.

Kubernetes Engine uses the Docker container runtime, and Docker containers are what we'll focus on in this course.

A process is a running program. In Linux and Windows, the memory address spaces of running processes have long been isolated from one another. Popular implementations of software containers such as Docker build on this isolation, taking advantage of additional operating-system features that give processes the capability to have their own namespaces and that give a supervisor process the ability to limit other processes' access to resources.

Containers start much faster than virtual machines and use fewer resources, because each container does not have its own instance of the operating system.



Why use containers?

Consistency	Loose coupling	Workload migration	Agility
Across development, testing, and production environments	Between application and operating system layers	Simplified between on-premises and cloud environments	Agile development and operations

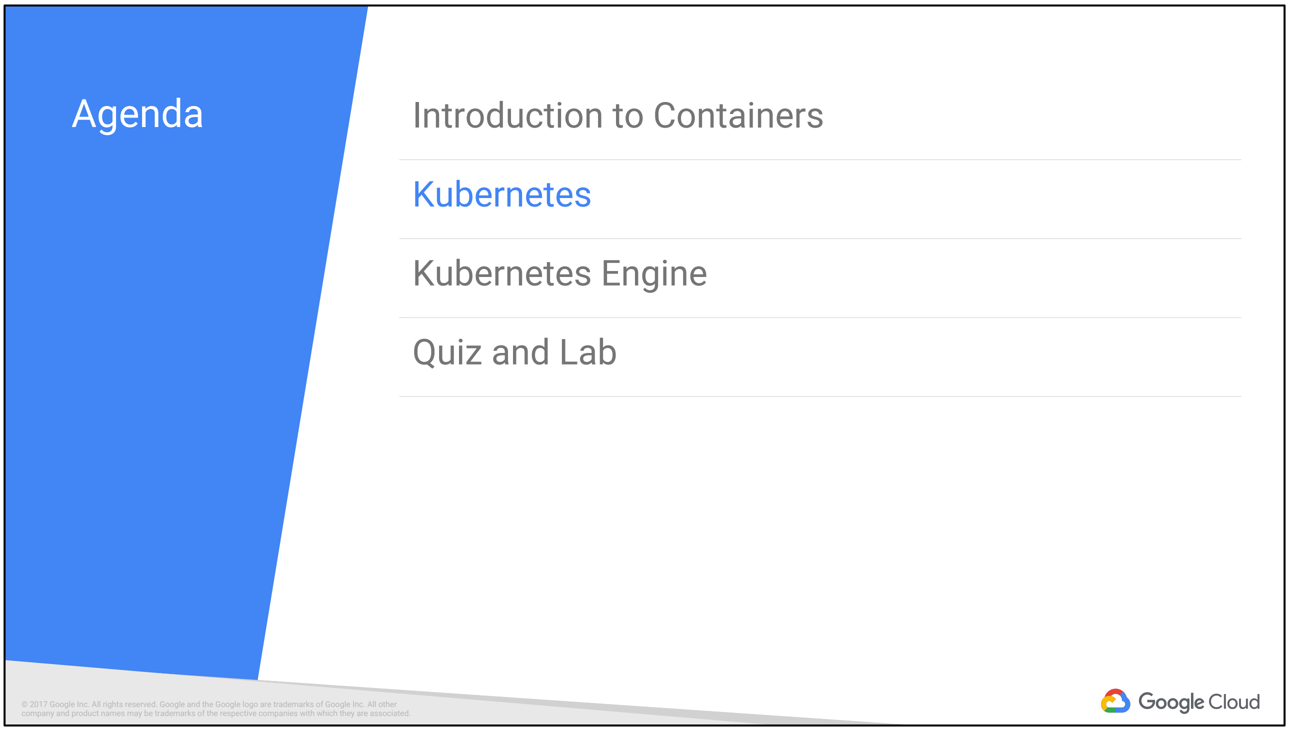
So what does a container provide that a virtual machine does not?

- 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 instead of the whole physical computer, this package can "boot" in ~1/20th of a second, compared to a minute or so for a modern virtual machine.
- 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?

- A developer has in their laptop plenty of compute power to run multiple containers, making for easier and faster development. Although 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

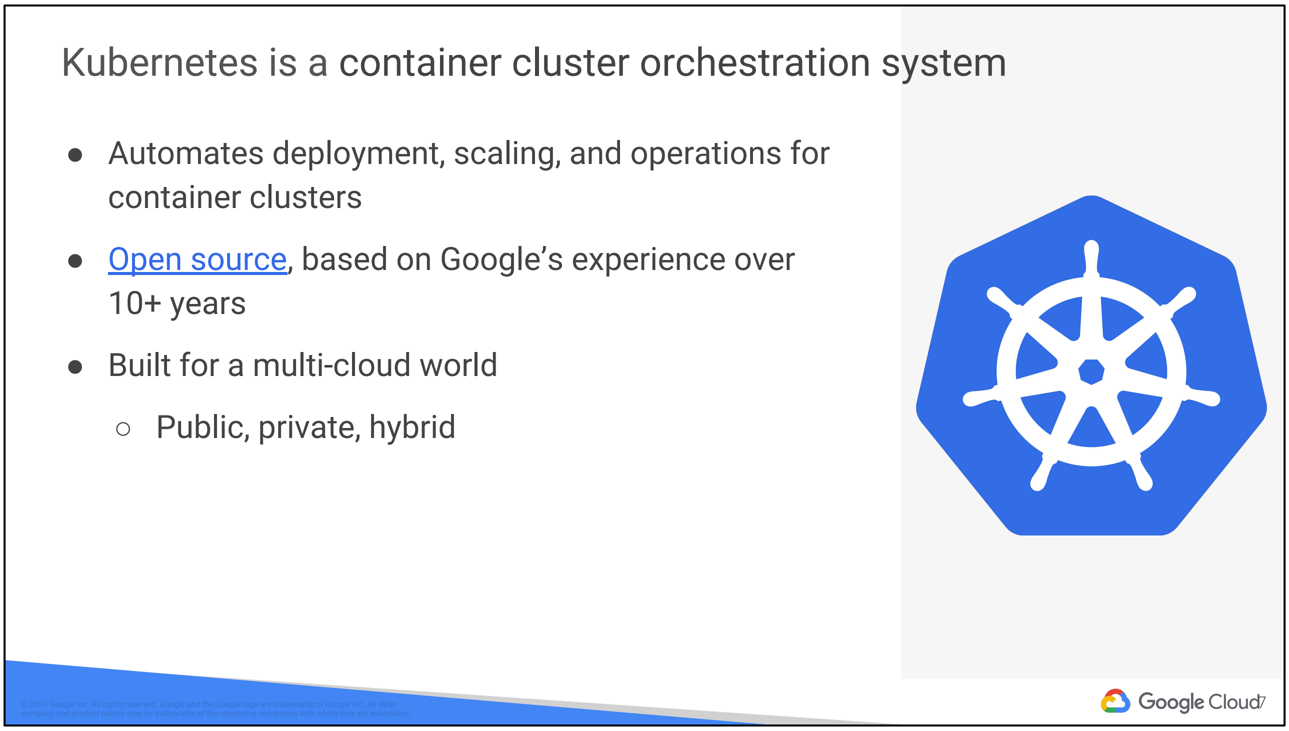
- container is a single command. Testing is also cheaper: in a public cloud where a VM is billed for a minimum of 10 minutes of compute time, a single test might only cost you a small amount. But if you’re running thousands of programmatically driven tests per day, this starts to add up. With a container, 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. Although it might be a daunting systems administration 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 a vastly lower risk to start a few containers housing these applications with some very compact scripting. This model eliminates a significant amount of error-prone, specialized, boilerplate work.



Agenda

- Introduction to Containers
- Kubernetes
- Kubernetes Engine
- Quiz and Lab

Google Cloud

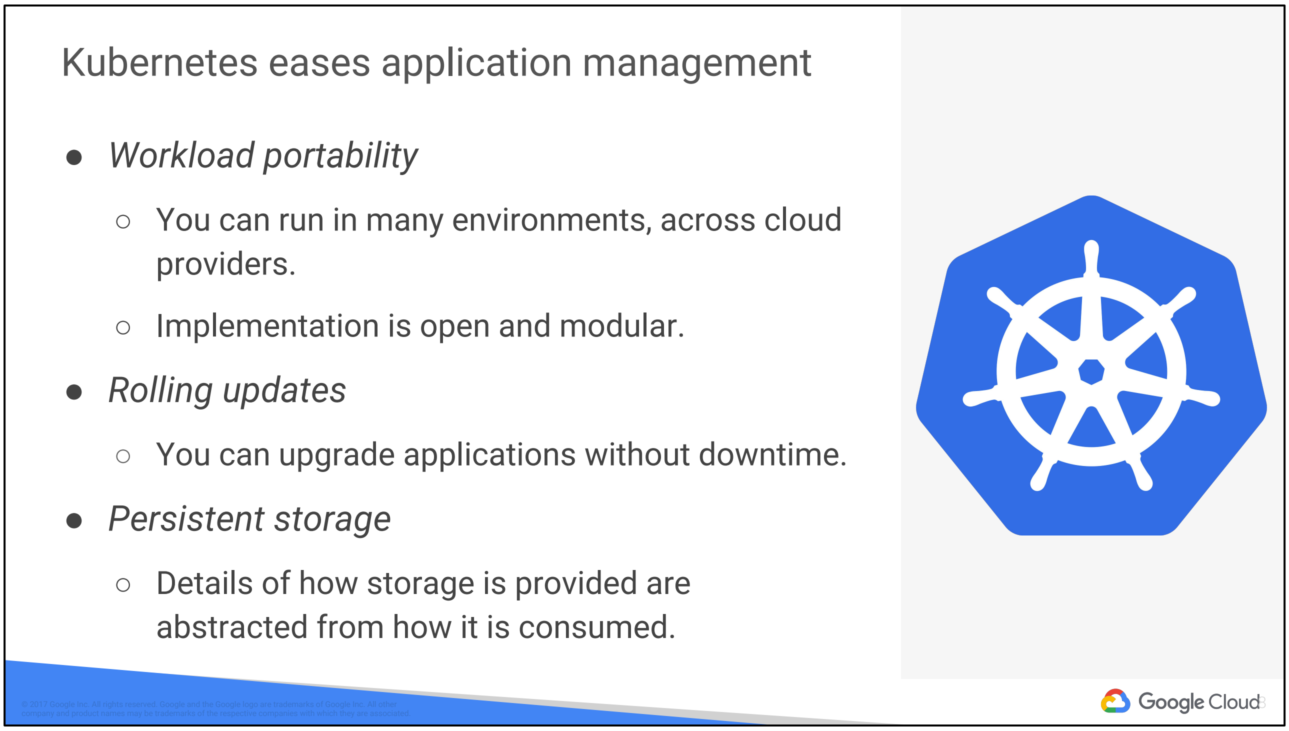


Kubernetes is a container cluster orchestration system

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

For its own internal operations, Google built a cluster management, networking, and naming system to allow container technology to operate at Google scale. Google now starts billions of containers per week. Google has applied its experience with containers to offer Kubernetes to the world. (The name is sometimes shortened to "k8s" on public forums.)

Kubernetes is an orchestration layer for containers, developed and released as open source by Google.



Kubernetes eases application management

- *Workload portability*
  - You can run in many environments, across cloud providers.
  - Implementation is open and modular.
- *Rolling updates*
  - You can upgrade applications without downtime.
- *Persistent storage*
  - Details of how storage is provided are abstracted from how it is consumed.

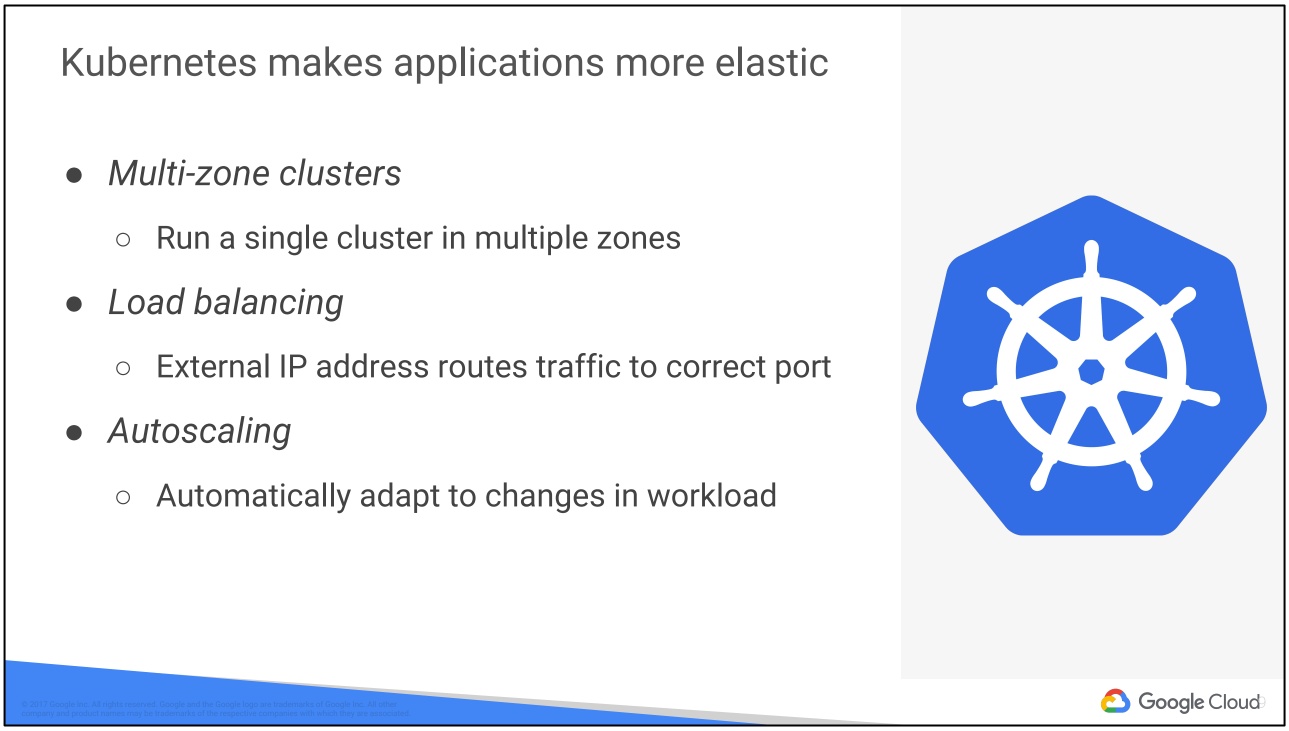
Google Cloud

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, and then “lift-and-shift” into the 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 persistent storage in Kubernetes, see: http://kubernetes.io/docs/user-guide/persistent-volumes/.



Kubernetes makes applications more elastic

- *Multi-zone clusters*
  - Run a single cluster in multiple zones
- *Load balancing*
  - External IP address routes traffic to correct port
- *Autoscaling*
  - Automatically adapt to changes in workload

Google Cloud

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

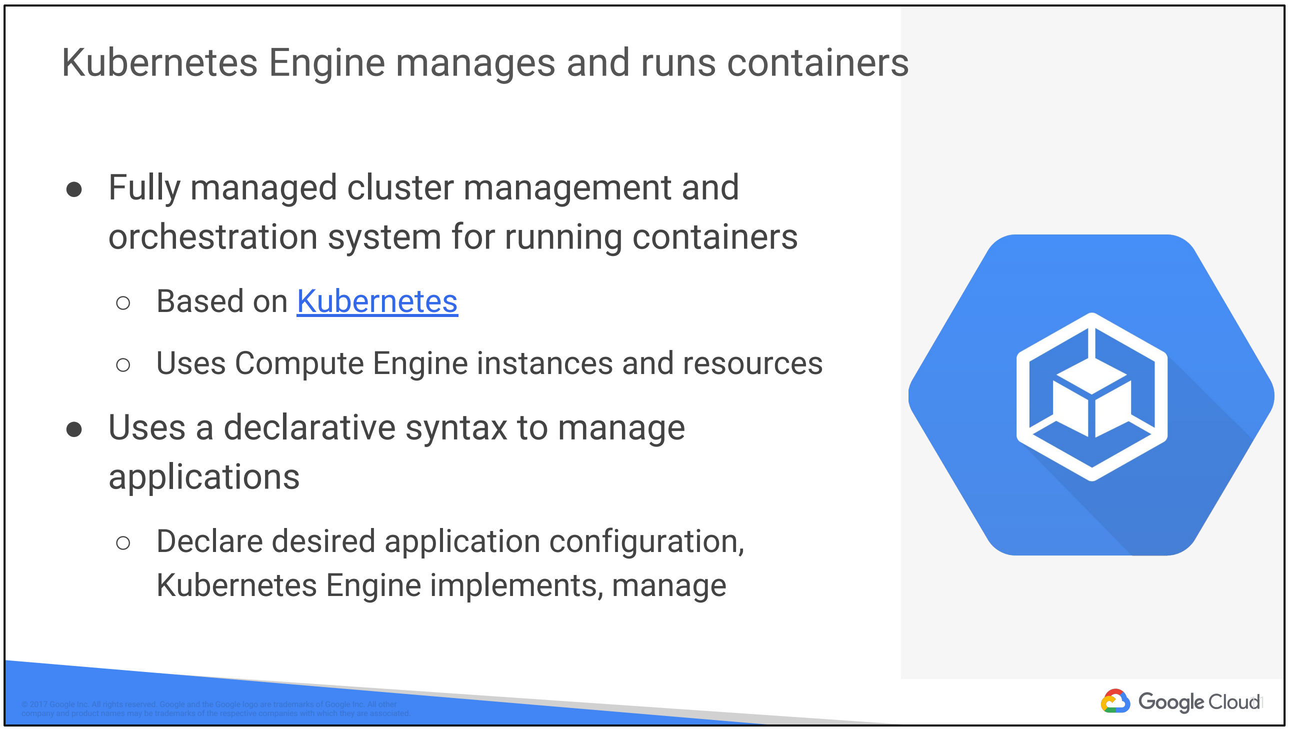
For more information on load balancing in Kubernetes, see: http://kubernetes.io/docs/user-guide/load-balancer/.

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

Agenda

- Introduction to Containers
- Kubernetes
- Kubernetes Engine
- Quiz and Lab

Google Cloud

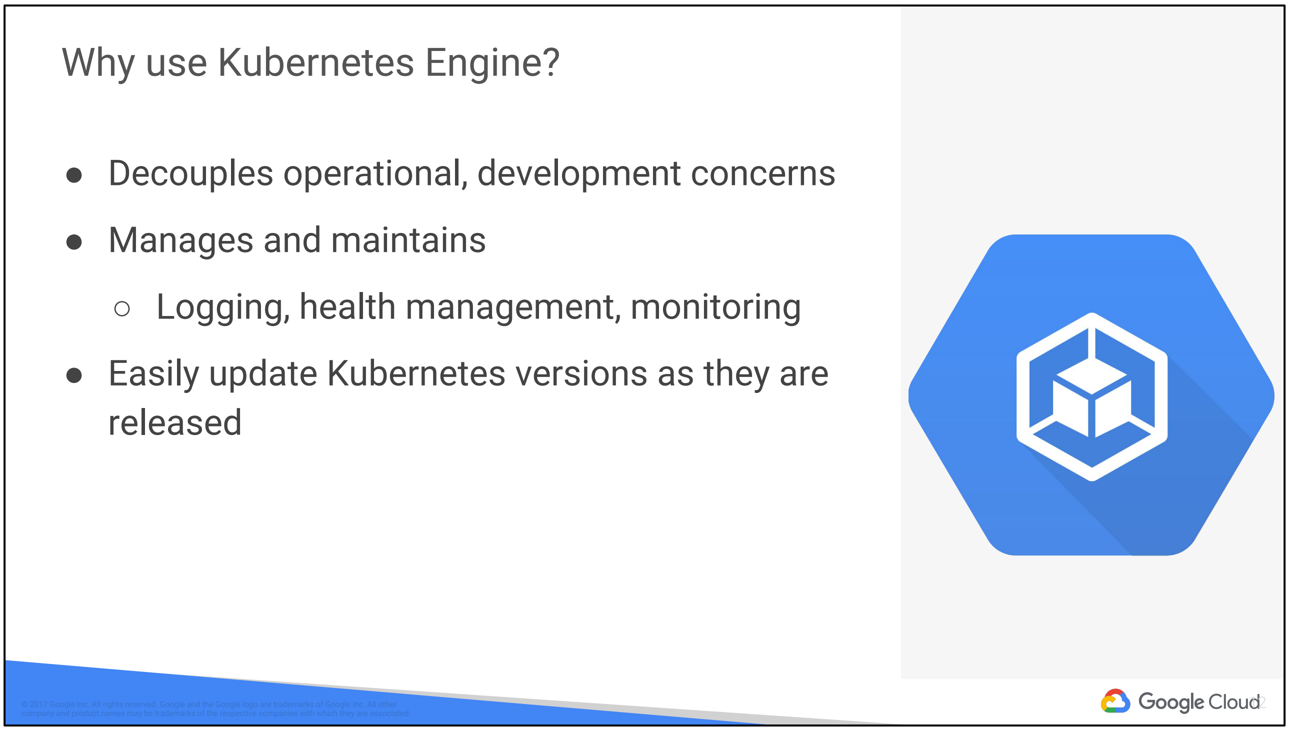


Kubernetes Engine manages and runs containers

- Fully managed cluster management and orchestration system for running containers
  - Based on Kubernetes
  - Uses Compute Engine instances and resources
- Uses a declarative syntax to manage applications
  - Declare desired application configuration, Kubernetes Engine implements, manage

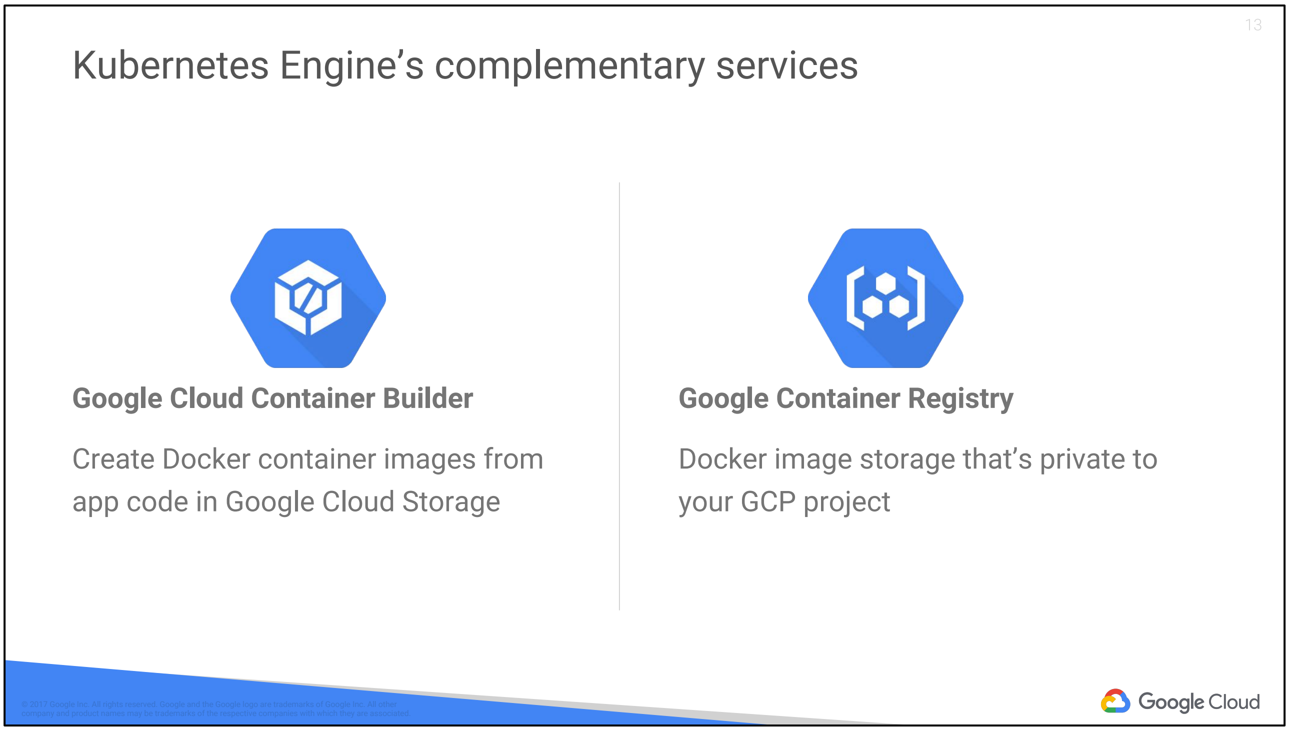
Google Cloud

Kubernetes Engine is Kubernetes as a service. You can run Kubernetes elsewhere too, both on your own premises and in other providers' clouds. But Kubernetes Engine is a scalable managed offering that runs on Google's infrastructure. You direct the creation of a cluster, and Kubernetes Engine schedules your containers into the cluster and manages them automatically, based on requirements you define.



Why use Kubernetes Engine?

- Decouples operational, development concerns
- Manages and maintains
  - Logging, health management, monitoring
- Easily update Kubernetes versions as they are released



Kubernetes Engine’s complementary services

Google Cloud Container Builder

Create Docker container images from app code in Google Cloud Storage

Google Container Registry

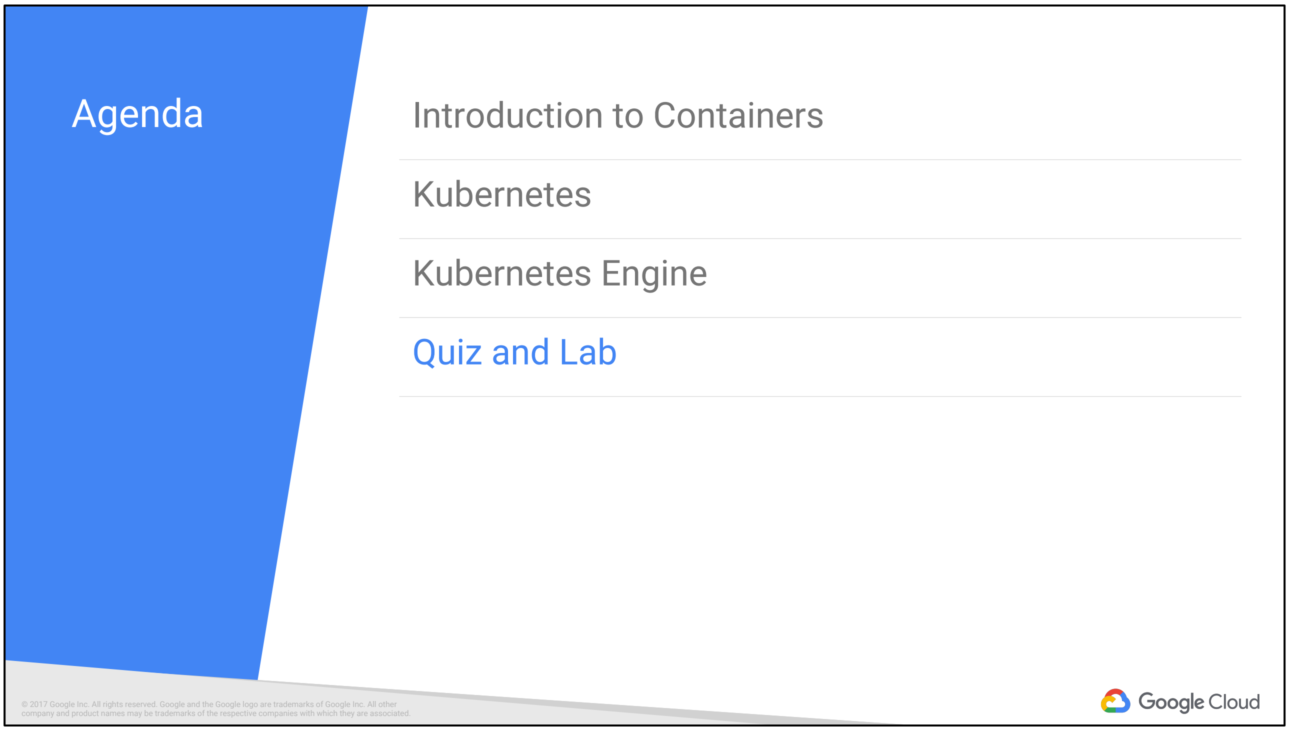
Docker image storage that’s private to your GCP project

Google Cloud

Container Builder lets you create Docker container images from application source code located in Cloud Storage. Container images created by Container Builder are automatically stored in Container Registry. You can deploy the container images you create on Kubernetes Engine, Compute Engine, App Engine Flexible Environment, or other services where you can run applications from Docker containers.

Container Registry provides private Docker image storage on Google Cloud Platform. Although Docker provides a central registry to store public images, you might not want your images to be accessible to the world. In this case, you must use a private registry. Container Registry runs on Google Cloud Platform, so it 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 Compute Engine instance or your own hardware.

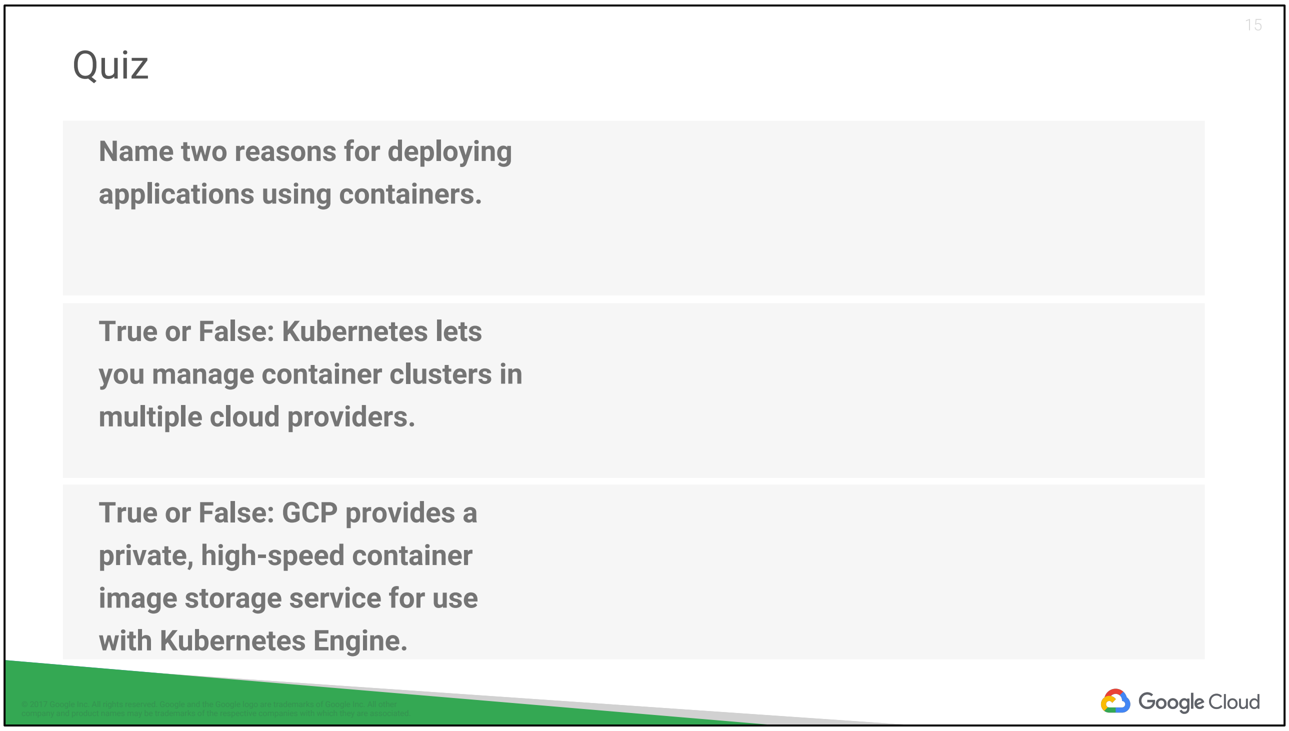
Container Registry only charges for the Cloud Storage storage and network egress consumed by your Docker images.



Agenda

- Introduction to Containers
- Kubernetes
- Kubernetes Engine
- Quiz and Lab

Google Cloud

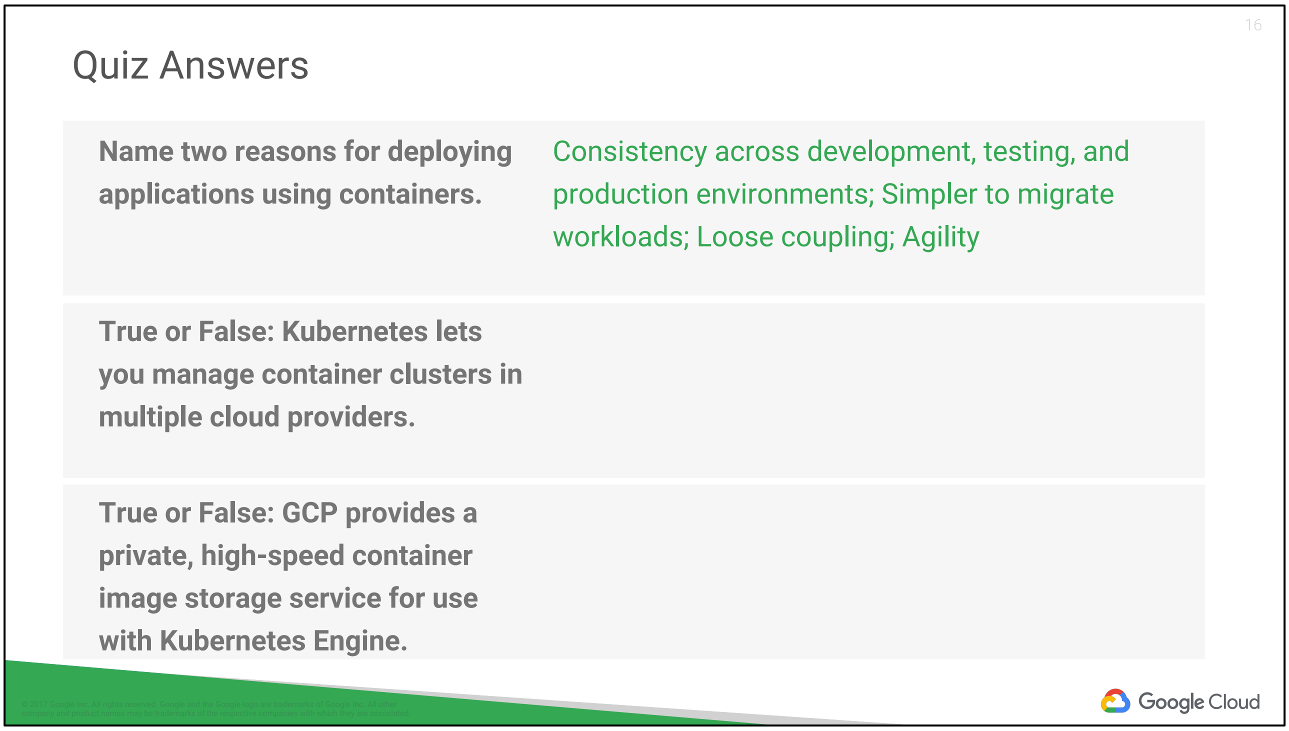


Quiz

Name two reasons for deploying applications using containers.

True or False: Kubernetes lets you manage container clusters in multiple cloud providers.

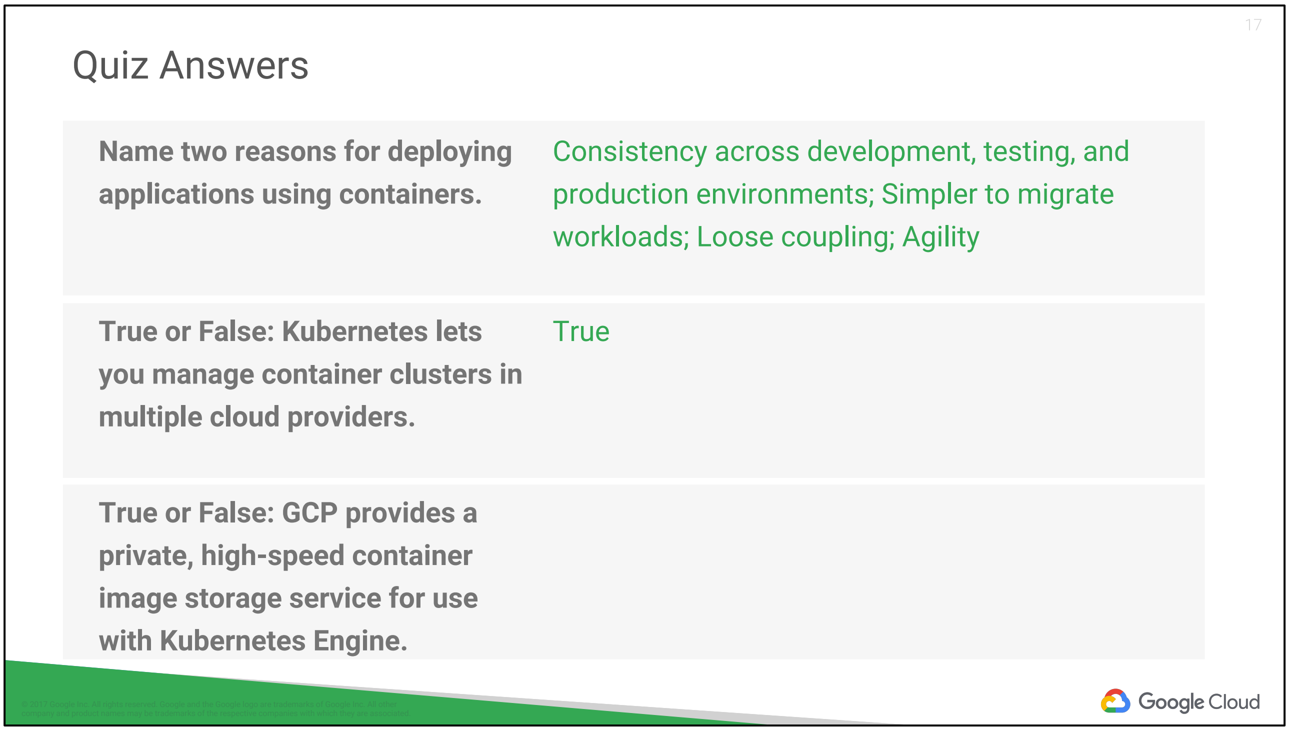
True or False: GCP provides a private, high-speed container image storage service for use with Kubernetes Engine.



Quiz Answers

Name two reasons for deploying applications using containers.	Consistency across development, testing, and production environments; Simpler to migrate workloads; Loose coupling; Agility
True or False: Kubernetes lets you manage container clusters in multiple cloud providers.	
True or False: GCP provides a private, high-speed container image storage service for use with Kubernetes Engine.	

Google Cloud



Quiz Answers

Name two reasons for deploying applications using containers.	Consistency across development, testing, and production environments; Simpler to migrate workloads; Loose coupling; Agility
True or False: Kubernetes lets you manage container clusters in multiple cloud providers.	True
True or False: GCP provides a private, high-speed container image storage service for use with Kubernetes Engine.	

Google Cloud

Quiz Answers

Name two reasons for deploying applications using containers.	Consistency across development, testing, and production environments; Simpler to migrate workloads; Loose coupling; Agility
True or False: Kubernetes lets you manage container clusters in multiple cloud providers.	True
True or False: GCP provides a private, high-speed container image storage service for use with Kubernetes Engine.	True

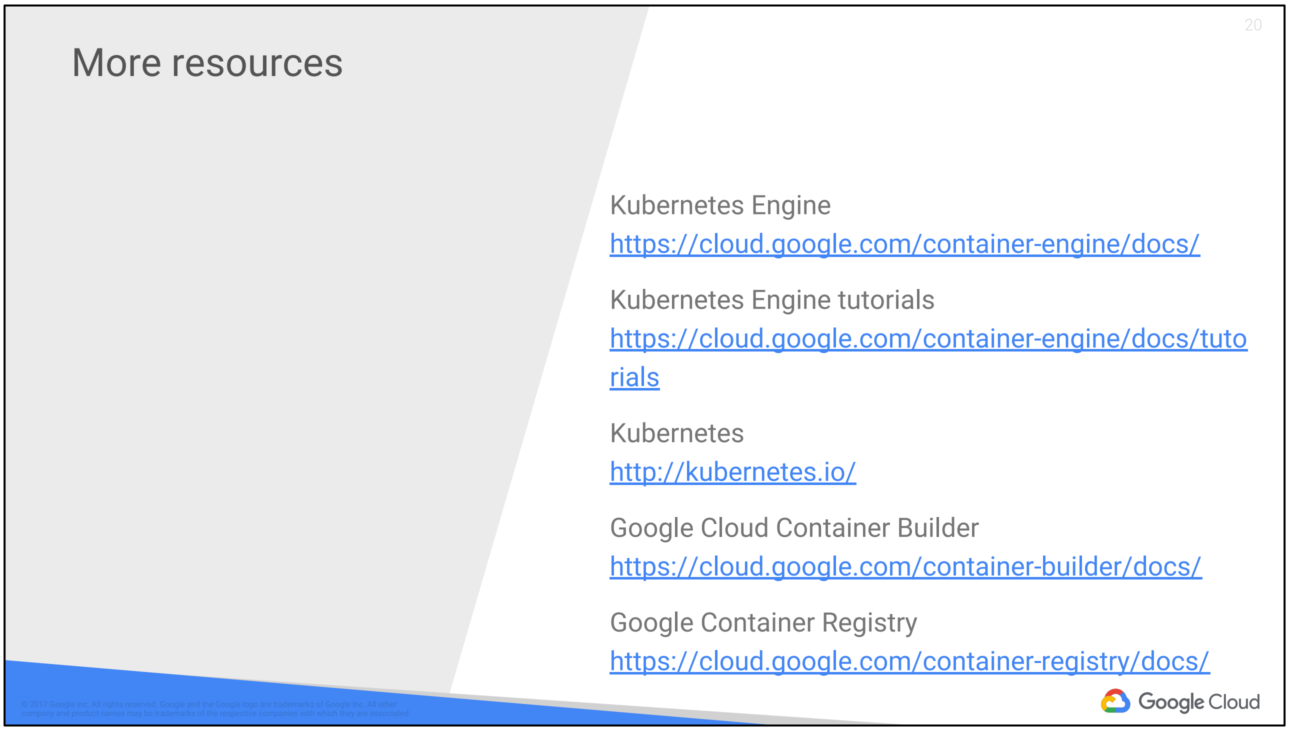
Google Cloud

Lab instructions

In this lab you will create a Kubernetes Engine cluster containing several containers, each containing a web server. You'll place a load balancer in front of the cluster and view its contents.

- Provision a Kubernetes cluster using Kubernetes Engine
- Deploy and manage Docker containers using kubectl

Google Cloud



More resources

Kubernetes Engine
https://cloud.google.com/container-engine/docs/

Kubernetes Engine tutorials
https://cloud.google.com/container-engine/docs/tutorials

Kubernetes
http://kubernetes.io/

Google Cloud Container Builder
https://cloud.google.com/container-builder/docs/

Google Container Registry
https://cloud.google.com/container-registry/docs/

Google Cloud