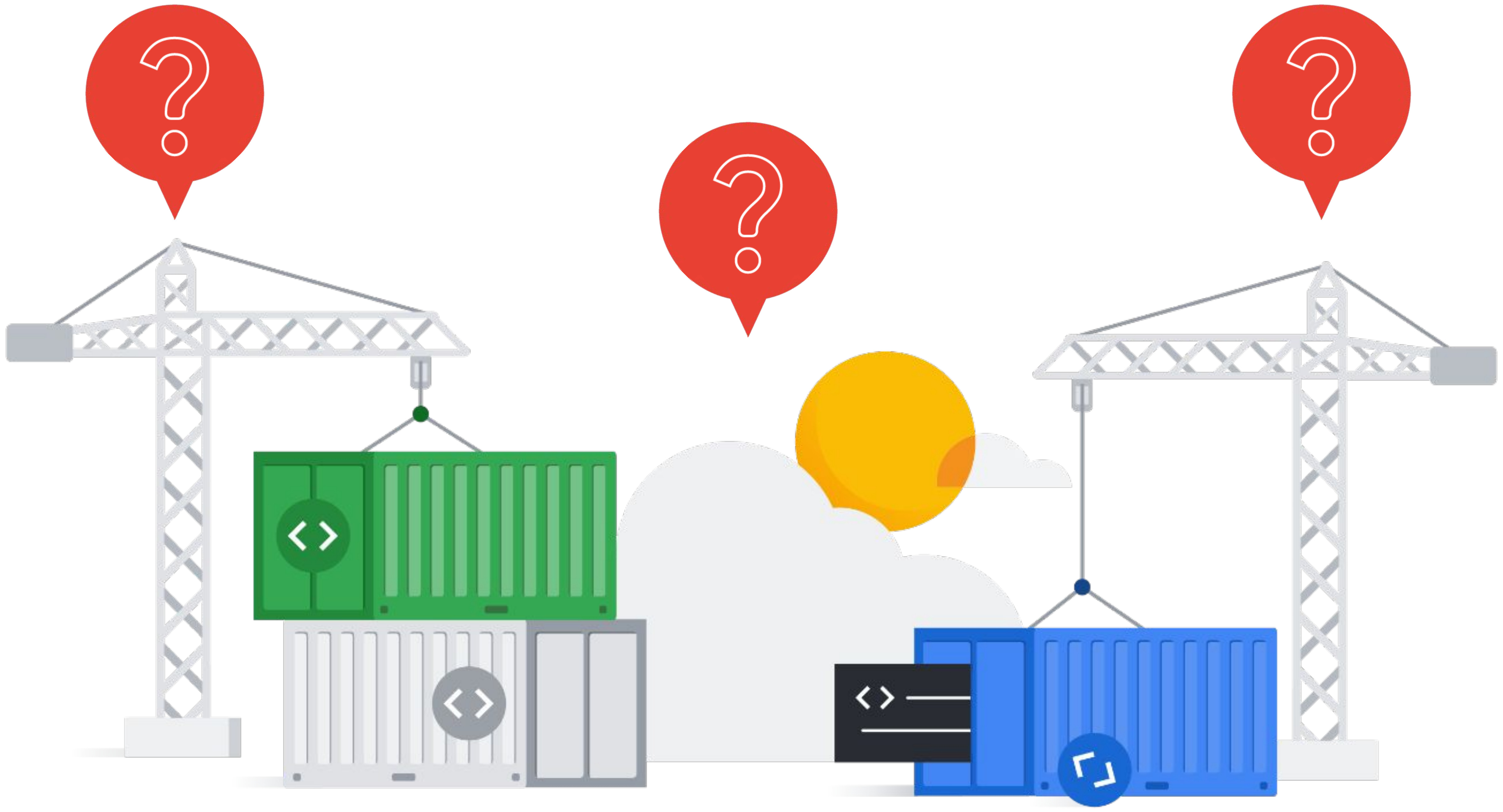
02

Introduction to Containers and Kubernetes

What are containers? How do they work?



Introduction to Containers and Kubernetes

01 Containers

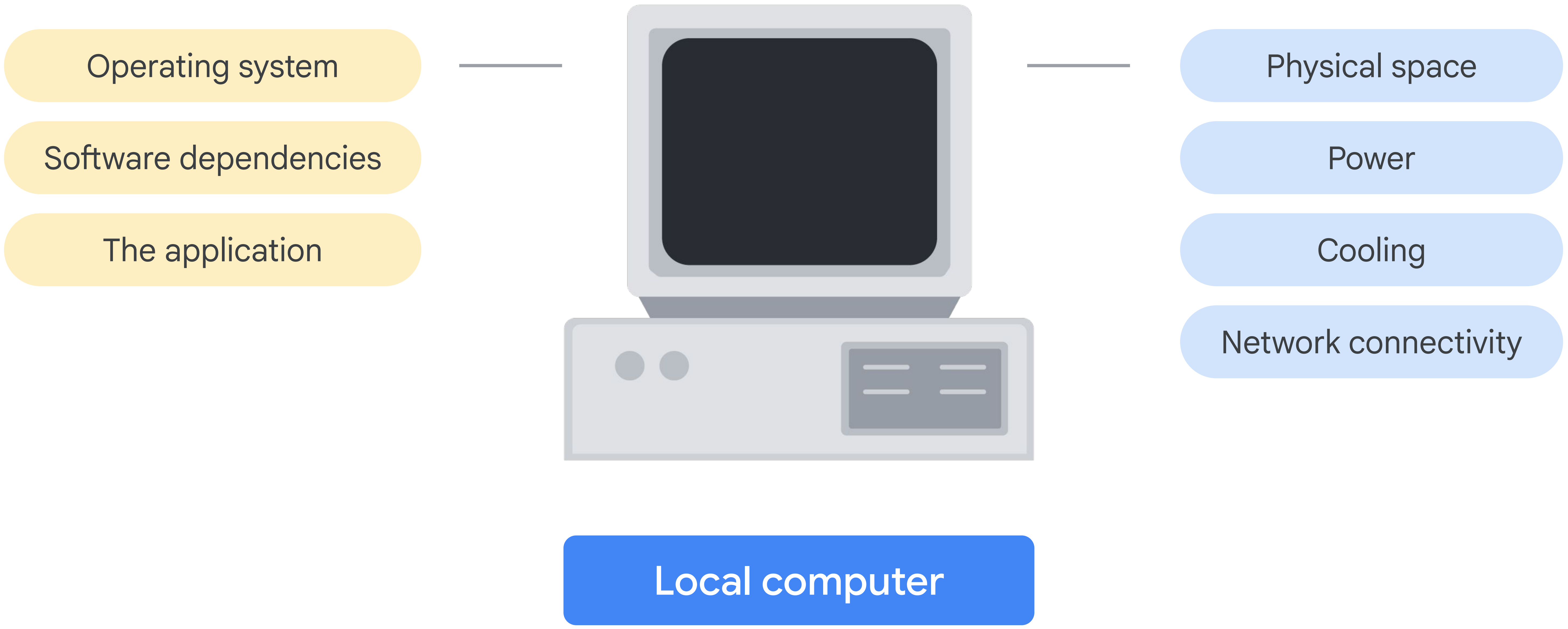
02 Container images

03 Kubernetes

04 Google Kubernetes Engine

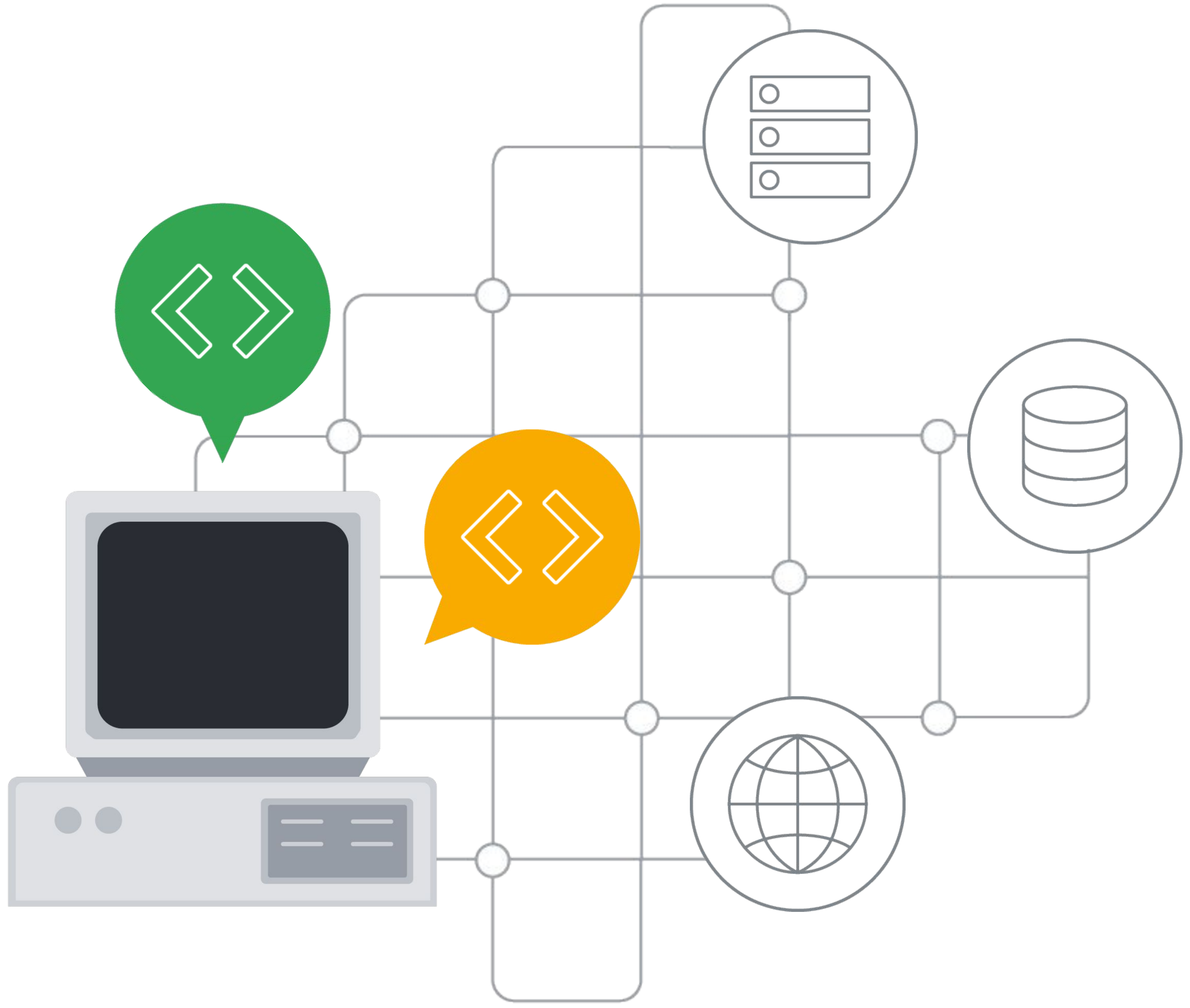
Introduction to Containers and Kubernetes

Not long ago, the common way to deploy an application was on a local computer

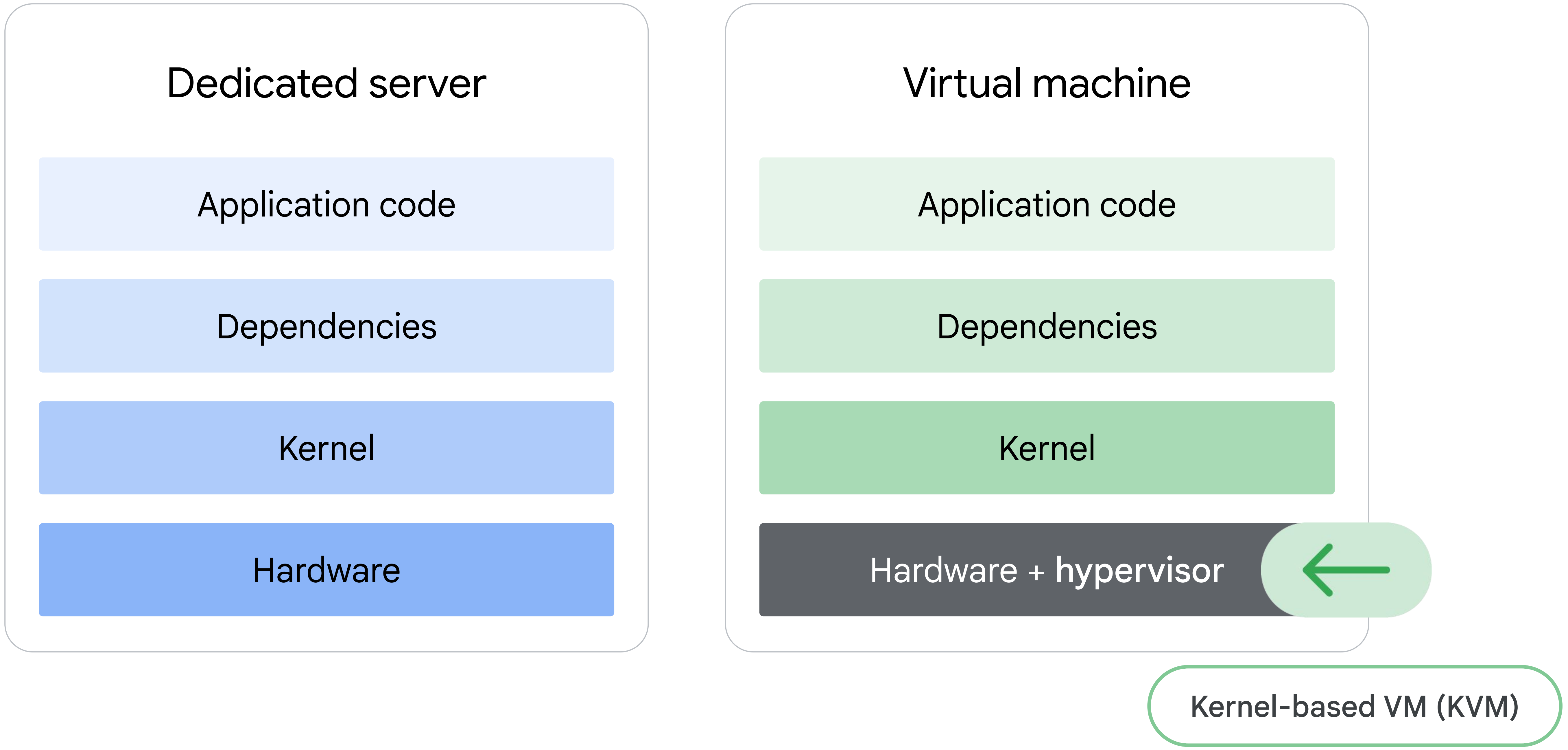


Then came virtualization

Virtualization is the process of creating a virtual version of a physical resource, such as a server, storage device, or network.



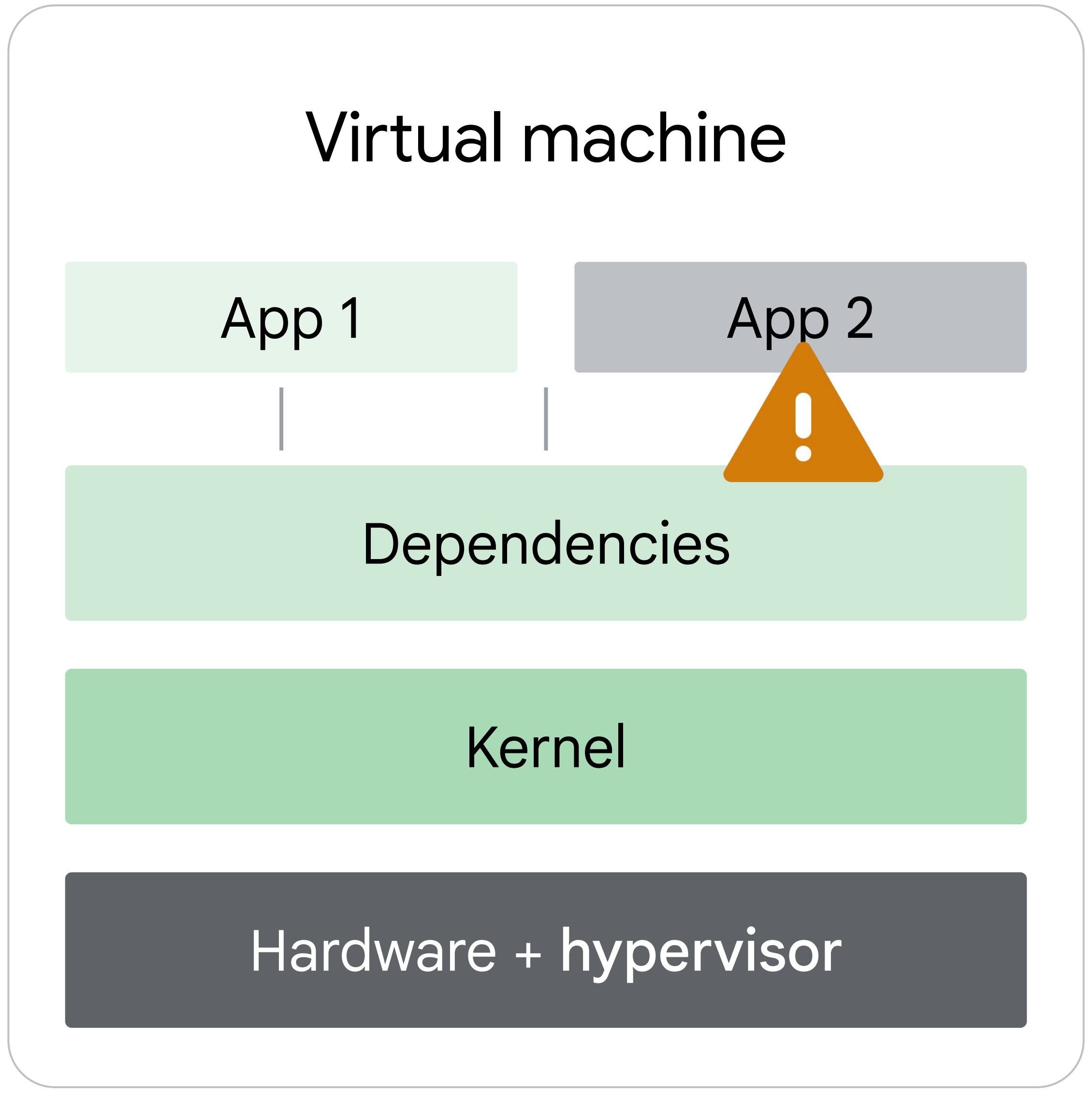
Hypervisor


Dedicated server
Application code
Dependencies
Kernel
Hardware
Virtual machine
Application code
Dependencies
Kernel
Hardware + hypervisor
Kernel-based VM (KVM)

Virtualization

- ✓ It takes less time to deploy new solutions.
- ✓ Fewer resources are wasted.
- ✓ Portability is improved.
- ! It isn't easy to move a VM from one hypervisor.
- ! VM operating systems take time to boot up.

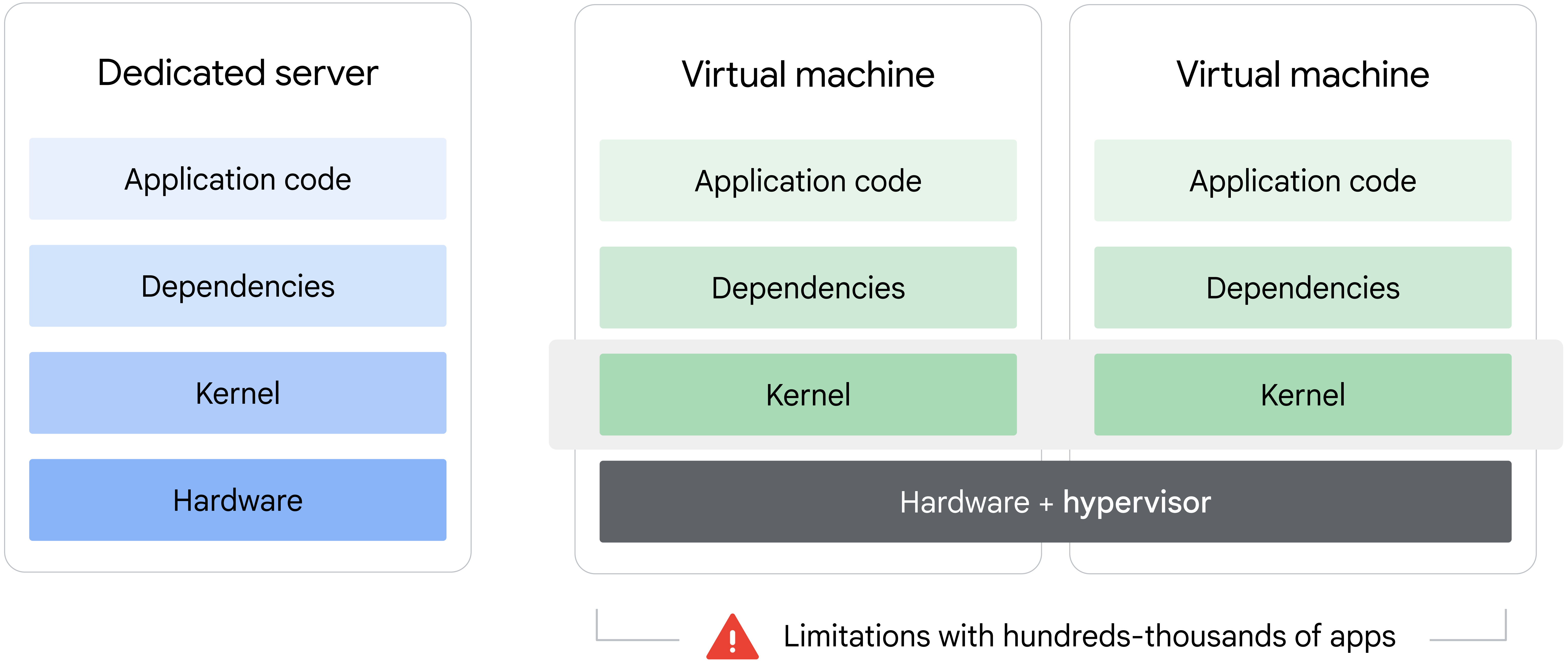
Applications that share dependencies are not isolated from each other



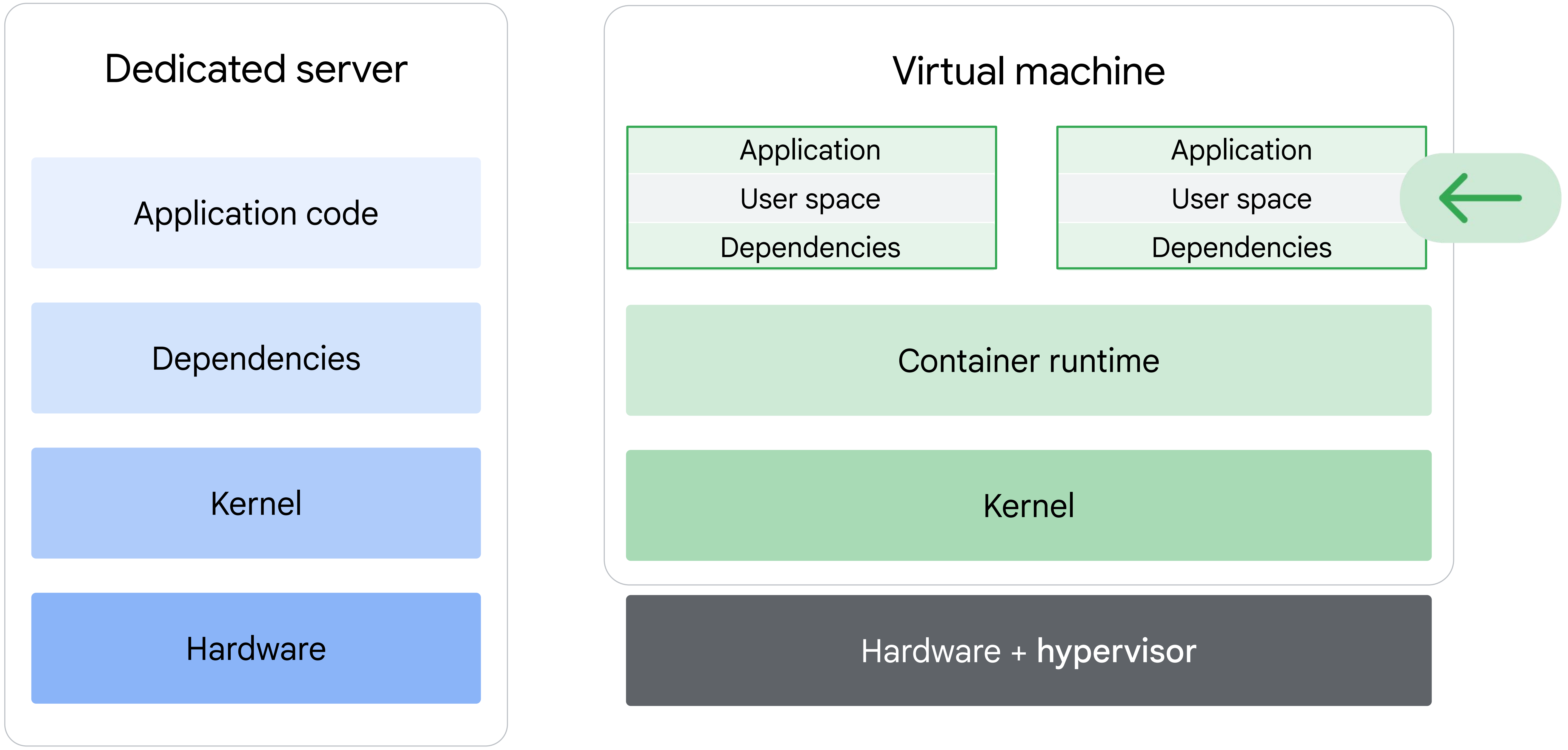
Try and solve this problem with:

- ✓ Software engineering policies
  - Need to be updated occasionally.
- ✓ Integration tests
  - Can cause novel failure modes that are hard to troubleshoot.
  - Slow down development.

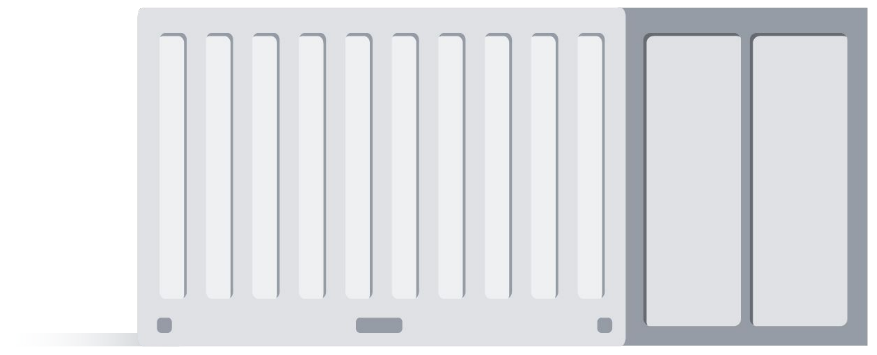
VM-centric solution: Run a dedicated VM for each application



Implement abstraction at the level of the application and its dependencies



Containers are isolated user spaces for running application code



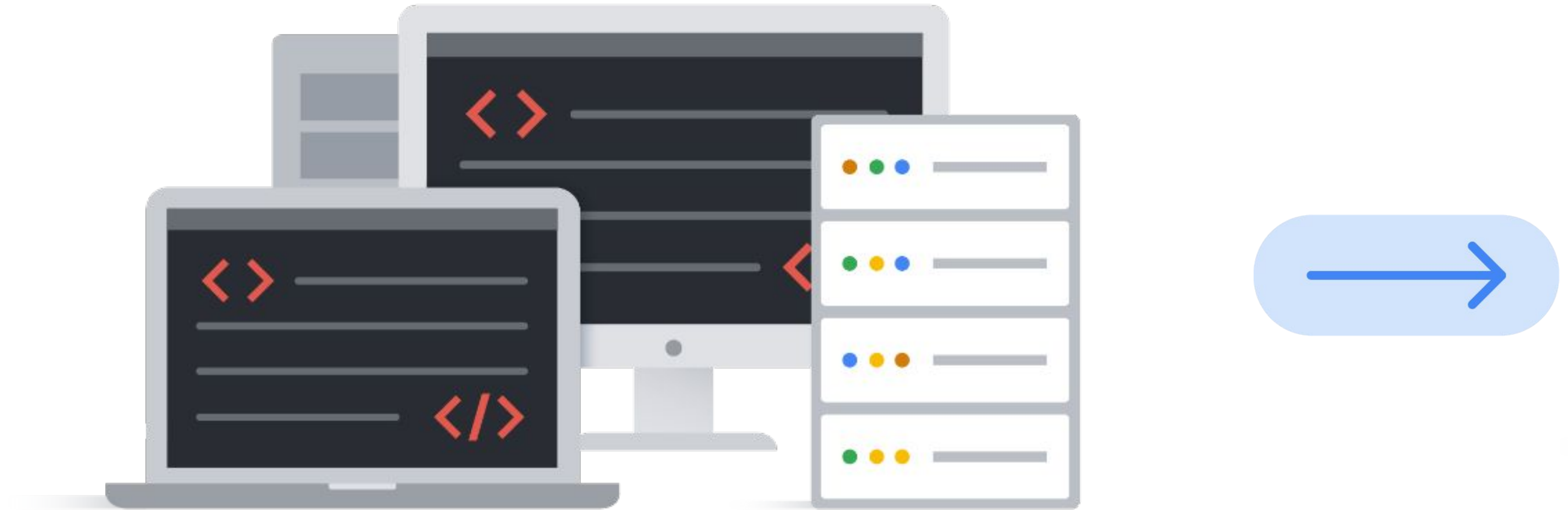
Containers are lightweight because they don't carry a full operating system.

Containers can be scheduled or integrated tightly with the underlying system.

Containers can be created and shut down quickly.

Containers do not boot an entire VM or initialize an operating system for each application.

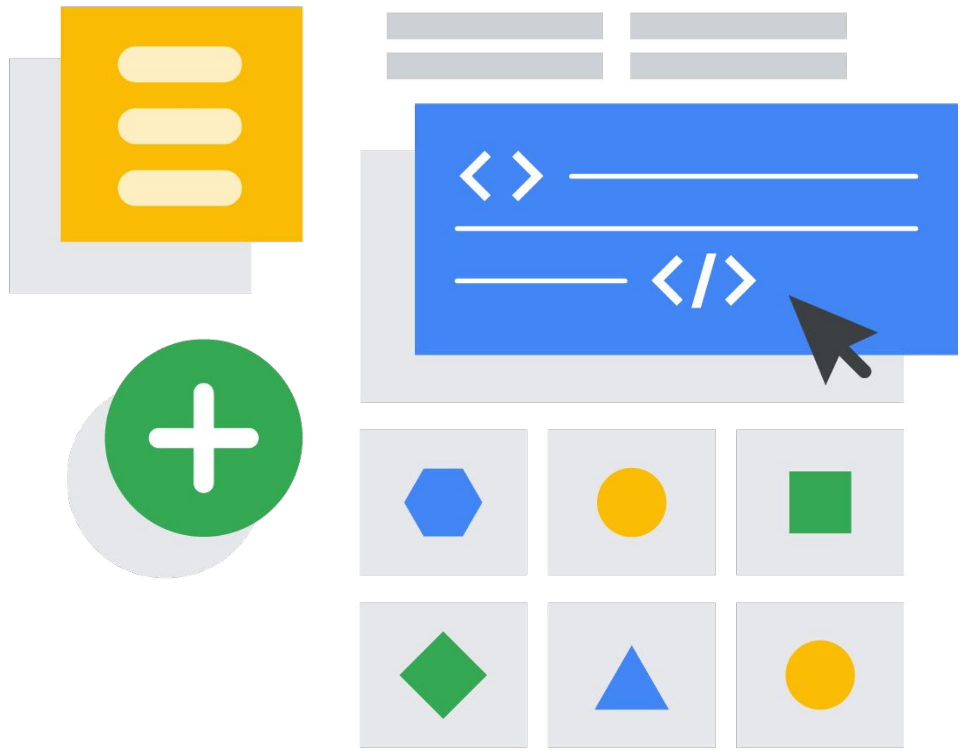
Develop containers in the usual ways, execute on VMs



Develop application code on desktops, laptops, and servers

The container can execute final code on VMs

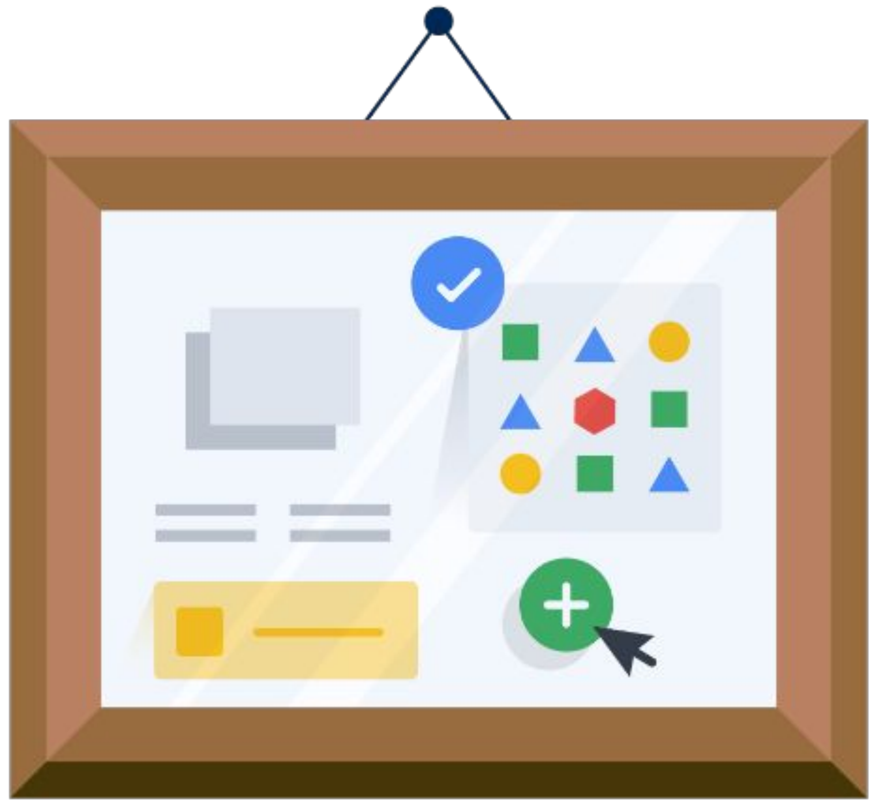
What makes containers so appealing to developers?



- ✓ They're a code-centric way to deliver high-performing, scalable applications.
- ✓ They provide access to reliable underlying hardware and software.
- ✓ Code will run successfully regardless if it is on a local machine or in production.
- ✓ They make it easier to build applications that use the microservices design pattern.

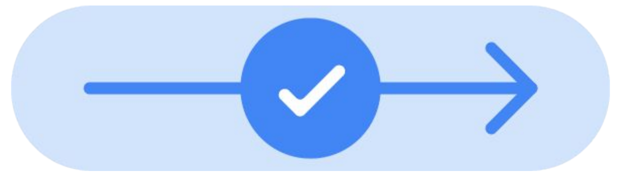
Introduction to Containers and Kubernetes

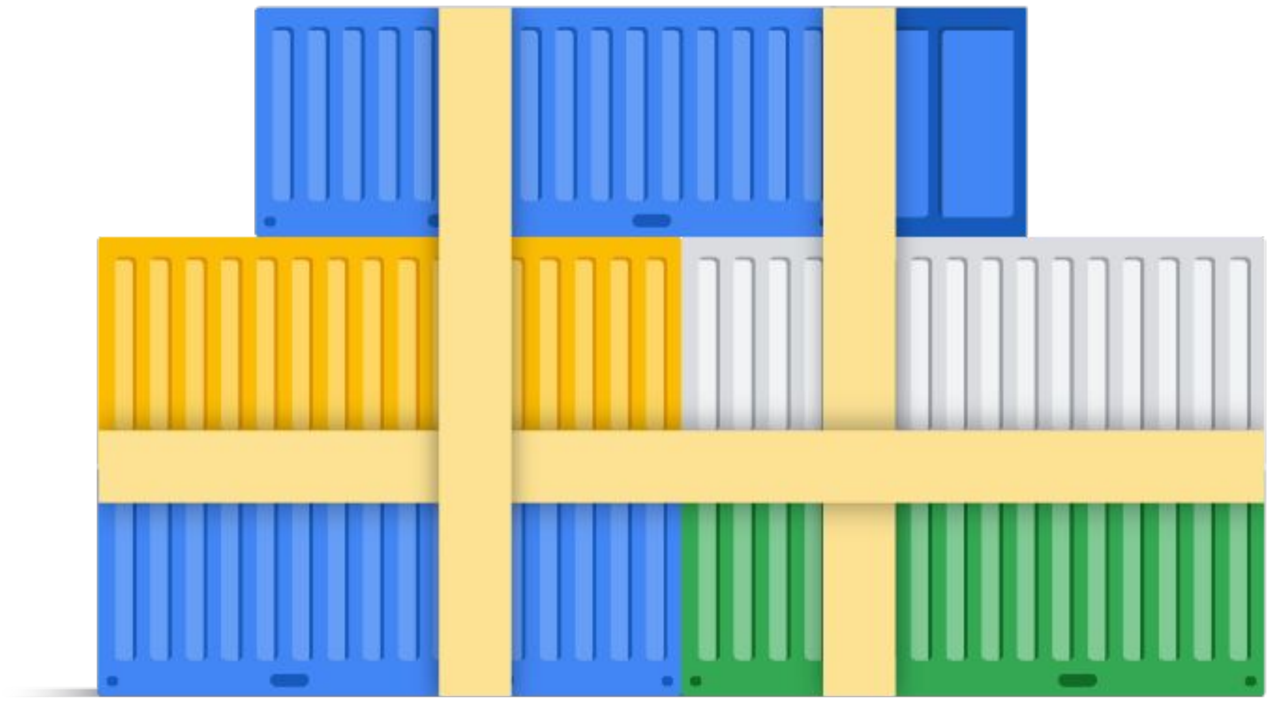
Container images



Application + dependencies =

Container image





Developers can package and ship an application without worrying about the system that it will run on.

You need software to build and run container images

Docker

✓ Docker is an open-source technology.

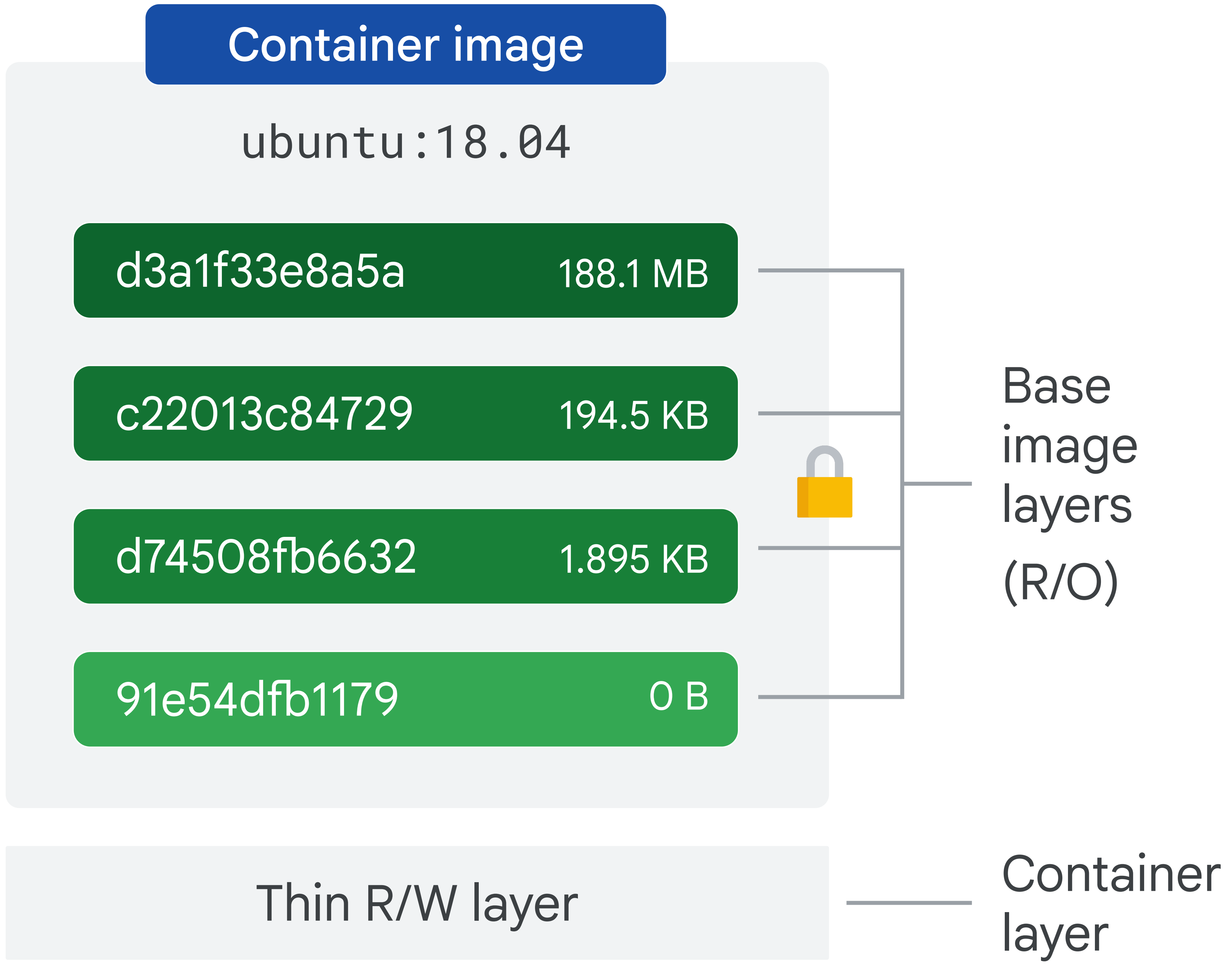
✓ It can be used to create and run applications in containers.

✗ It doesn't orchestrate applications at scale.

Linux technologies enable containers to isolate workloads

01	Linux process	Each Linux process has its own virtual memory address space, separate from all others, and Linux processes can be rapidly created and destroyed.
02	Linux namespaces	Containers use Linux namespaces to control what an application can see, such as process ID numbers, directory trees, and IP addresses.
03	Linux cgroups	Linux cgroups control what an application can use, such as its maximum consumption of CPU time, memory, I/O bandwidth, and other resources.
04	Union file systems	Containers use union file systems to bundle everything needed into a neat package.

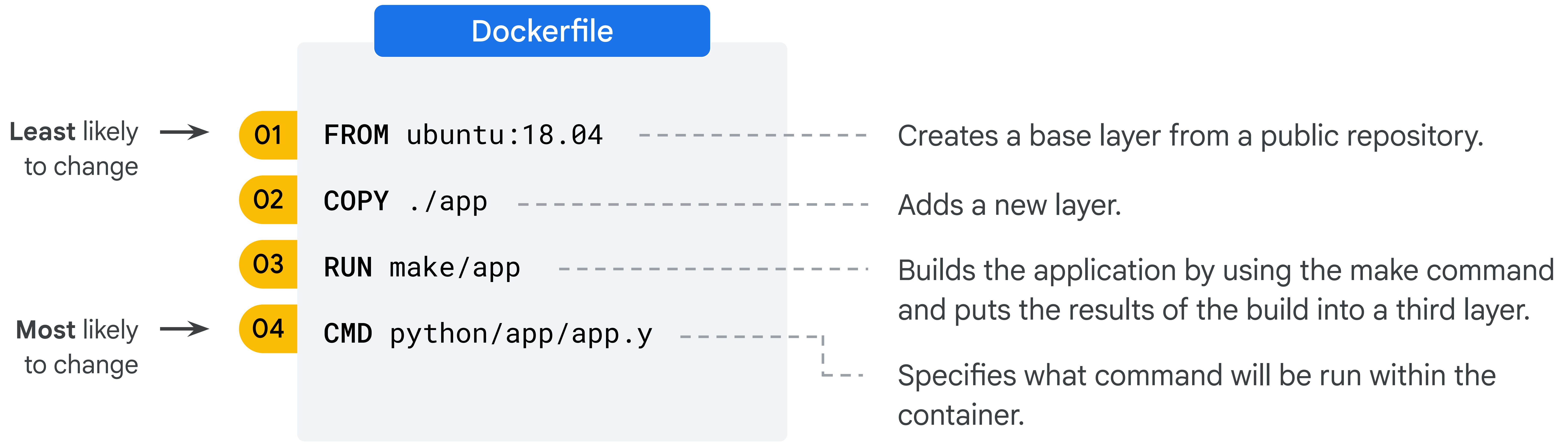
The container manifest and Dockerfile



Dockerfile

```
FROM ubuntu:18.04

COPY ./app

RUN make/app

CMD python/app/app.y
```

A Dockerfile contains four commands, each of which creates a layer



Dockerfile

```
01  FROM ubuntu:18.04
02  COPY ./app
03  RUN make/app
04  CMD python/app/app.y
```

Least likely to change	01
Most likely to change	04

Creates a base layer from a public repository.

Adds a new layer.

Builds the application by using the make command and puts the results of the build into a third layer.

Specifies what command will be run within the container.

Dockerfiles dos and don'ts

Dockerfile

```
FROM ubuntu:18.04

COPY ./app

RUN make/app

CMD python/app/app.y
```

It's not a best practice to build your application in the same container where you ship and run it.

✓ Application packaging relies on a multi-stage build process.

One container builds the final executable image and a separate container receives only what is needed to run the application.

Dockerfiles dos and don’ts

Dockerfile

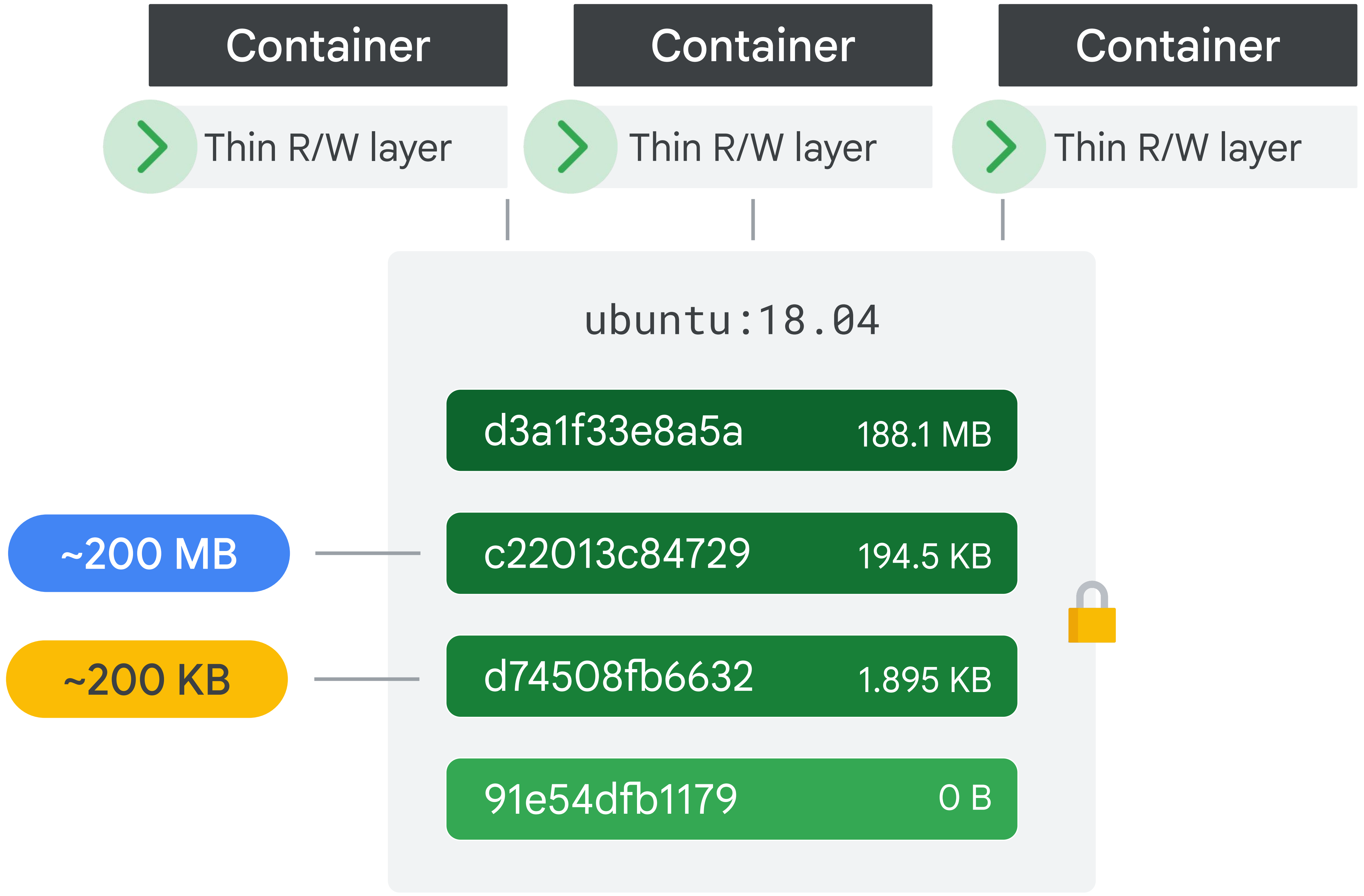
```
FROM ubuntu:18.04

COPY ./app

RUN make/app

CMD python/app/app.y
```

✓ The container runtime adds a new writable layer on top of the underlying layers called the **container layer**.

✓ All changes made to the running container are written to this thin writable container layer.

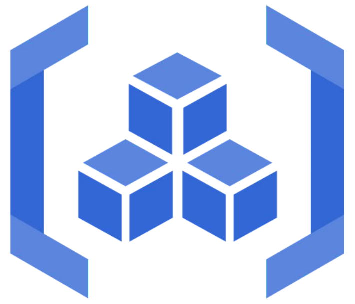
✓ The container layers is ephemeral.

Each container has its own writable container layer



How can you get or create containers?

Use publicly available open-source container images as the base for your own images.



Use the Google maintained **Artifact Registry** at pkg.dev, which contains public, open source images.

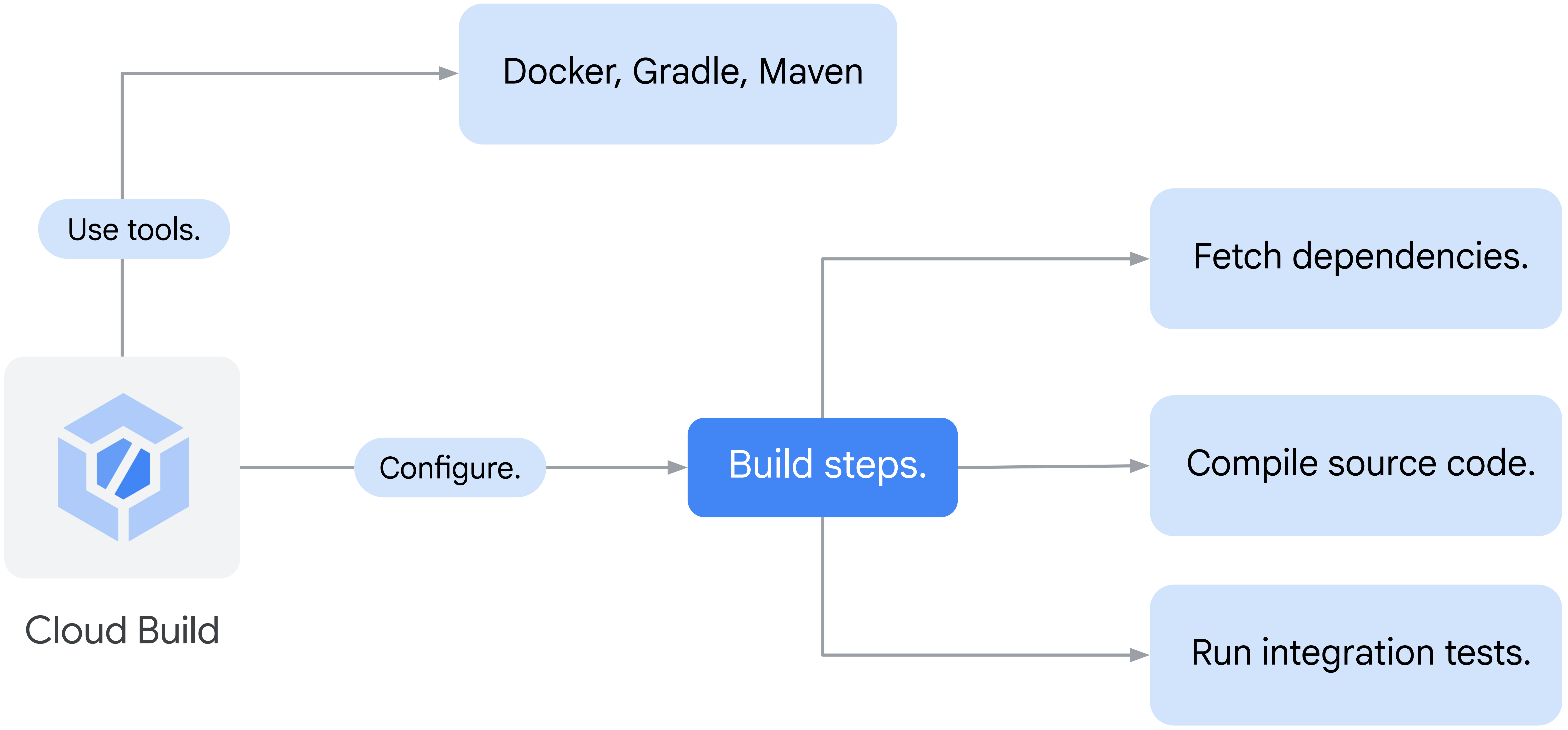
Use container images available in other public repositories, like the **Docker Hub Registry** and **GitLab.**

How can you get or create containers?

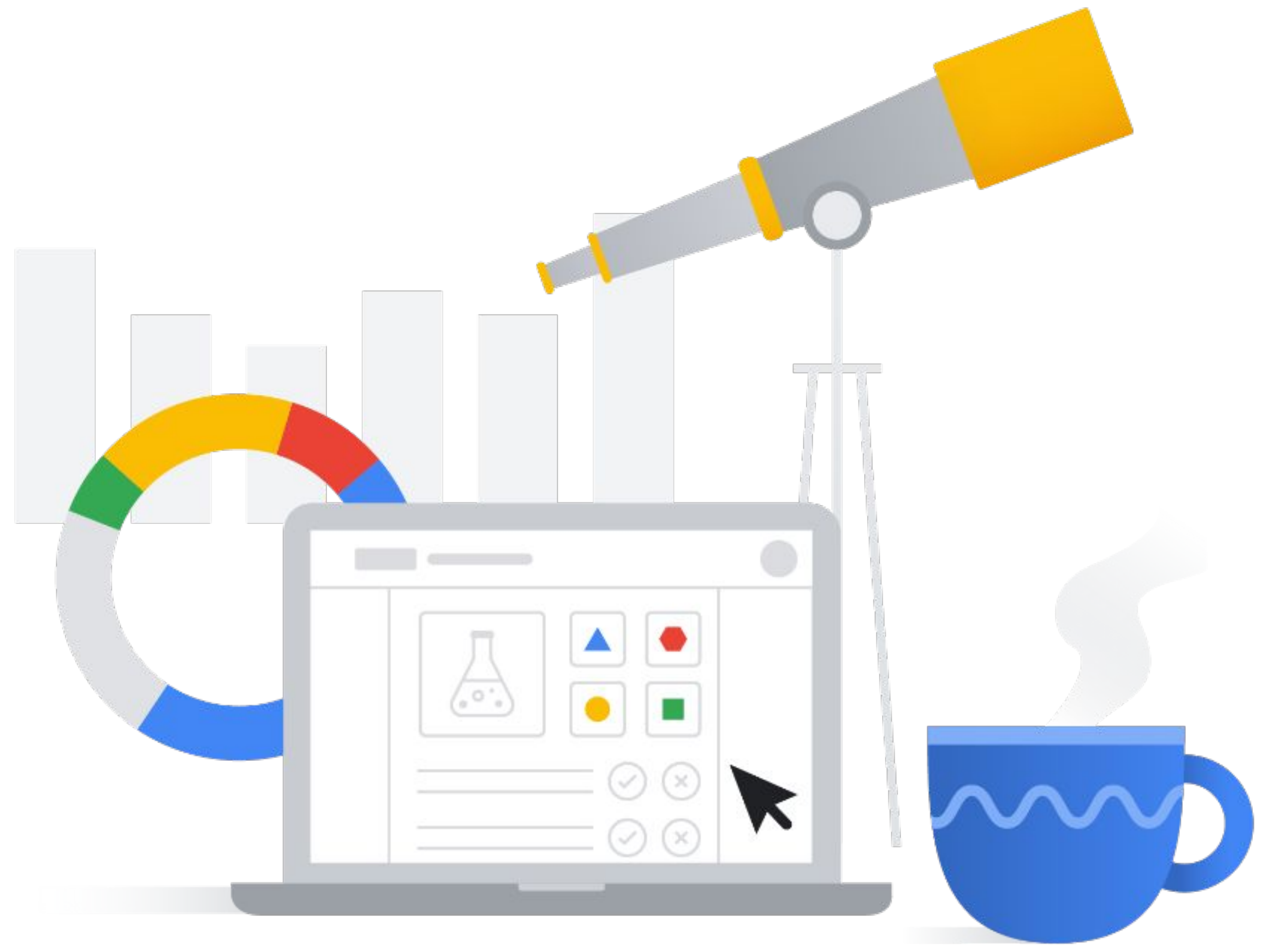
Use **Cloud Build**, a Google-provided managed service for building containers.

- ✓ Is integrated with Cloud IAM.
- ✓ Is designed to retrieve the source code builds from different code repositories:
  - Cloud Source Repositories
  - GitHub
  - Bitbucket

How to generate a build with Cloud Build



Lab: Working with Cloud Build



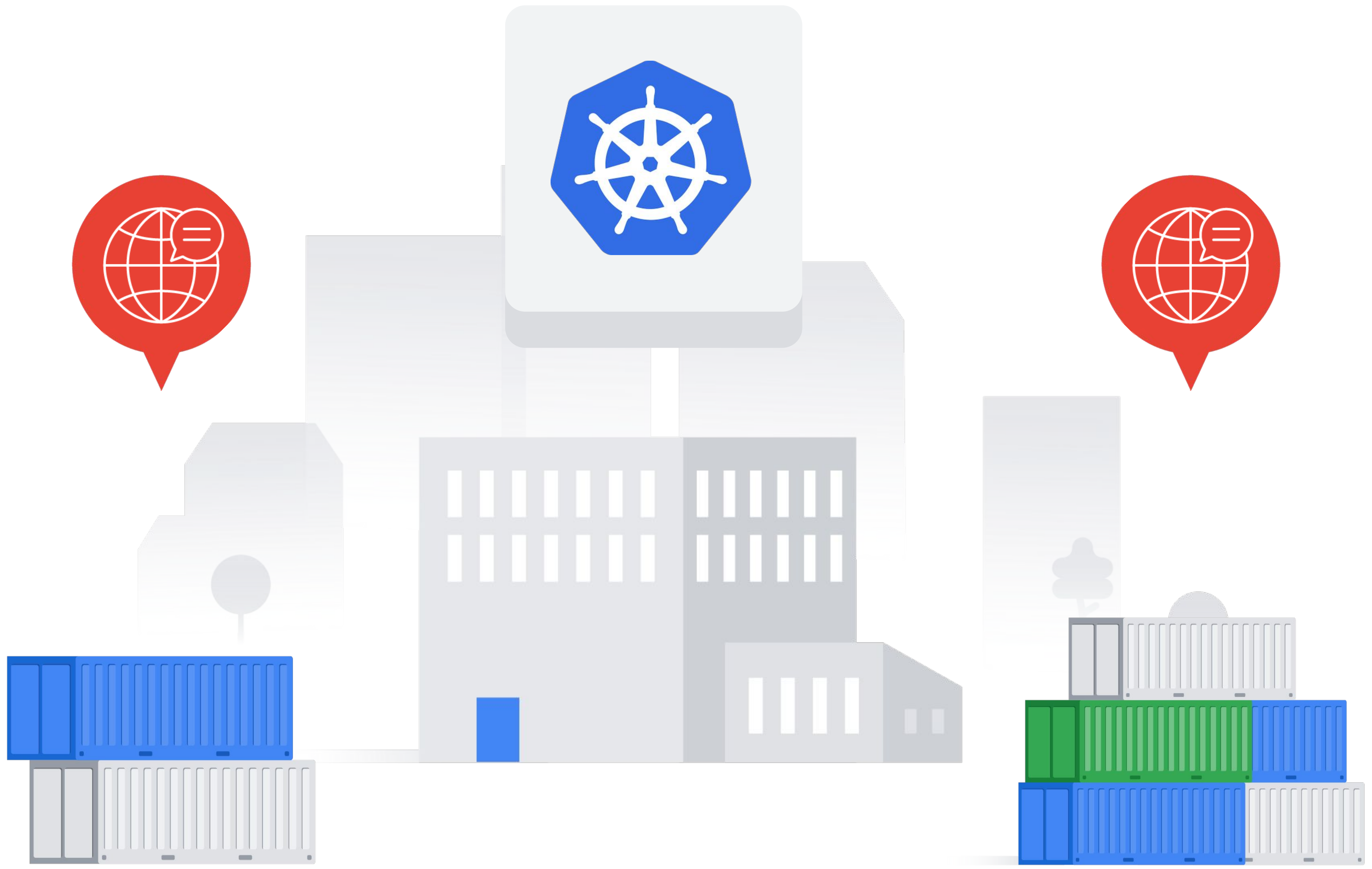
01 Use provided code to build a Docker container image and a Dockerfile.

02 Upload the container to the Google Cloud Artifact Registry.

15:00

Introduction to Containers and Kubernetes

Kubernetes



What is Kubernetes?

It's an open source platform for managing containerized workloads and services.

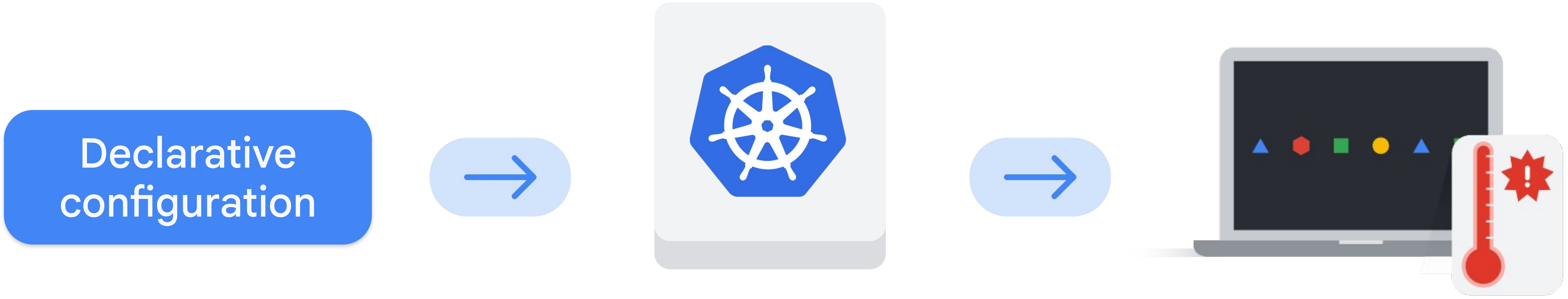
It makes it easy to orchestrate many containers on many hosts, scale them as microservices, and deploy rollouts and rollbacks.

It's a set of APIs to deploy containers on a set of nodes called a cluster.

It's divided into a set of primary components that run as the control plane and a set of nodes that run containers.

You can describe a set of applications and how they should interact with each other, and Kubernetes figures how to make that happen.

Kubernetes supports declarative configurations



Describe the state of the infrastructure you want to achieve.

Kubernetes's job is to make the deployed system conform to your desired state.

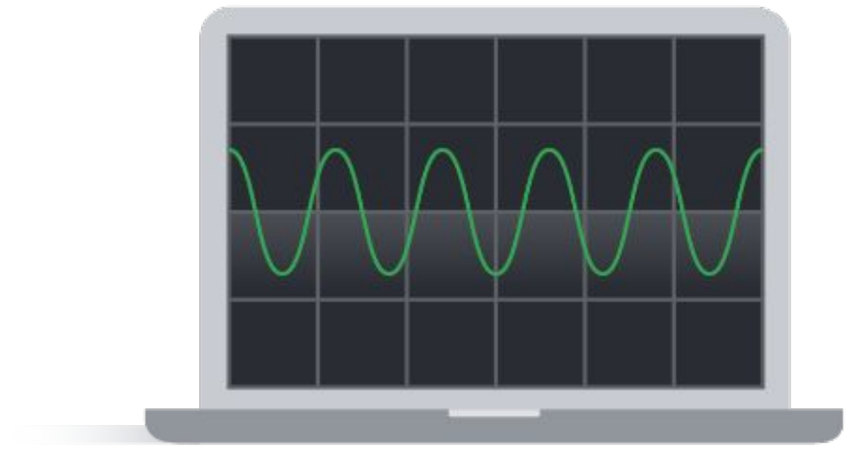
Reduces the risk of error.

Kubernetes also allows imperative configurations

Imperative configuration

Issue commands to change the system's state.

Kubernetes automatically keeps a system in a state you declare.



Experienced admins use them for quick, temporary fixes.

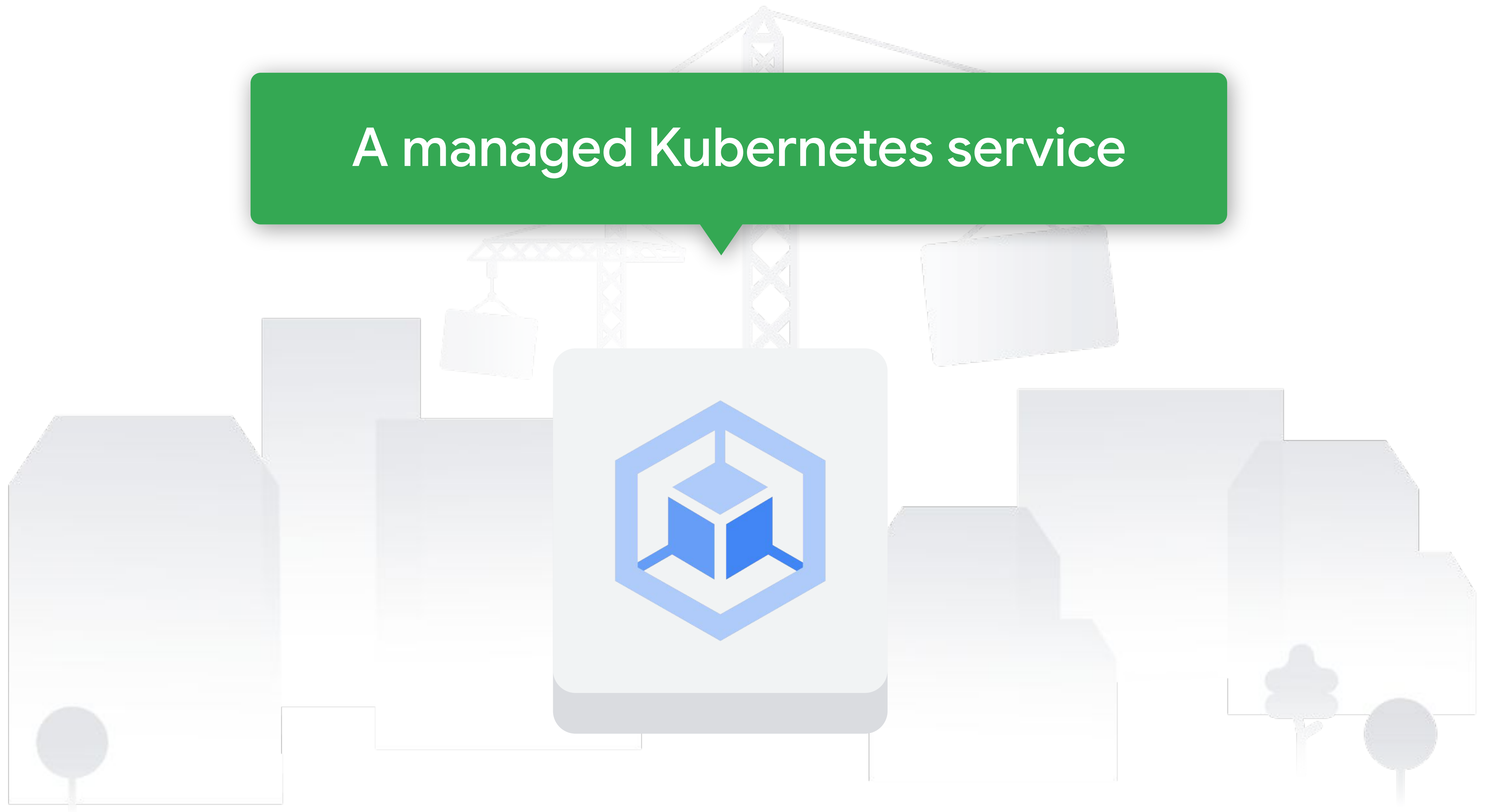
Kubernetes features

Kubernetes

- ✓ Supports **stateless** apps
- ✓ Supports **stateful** apps
- ✓ Autoscales containerized apps
- ✓ Allows resource request levels
- ✓ Allows resource limits
- ✓ Is extensible
- ✓ Is open source and portable

Introduction to Containers and Kubernetes

Google Kubernetes Engine



A managed Kubernetes service

Deploy Kubernetes

Manage Kubernetes

Scale Kubernetes

Google Kubernetes Engine features

GKE is fully managed.

Operating systems are optimized to scale quickly.

GKE Autopilot manages cluster configuration.

GKE auto upgrade ensures clusters have the latest stable version of Kubernetes.

GKE auto repair fixes unhealthy nodes.

GKE scales the cluster itself.

GKE is integrated with Cloud Build.

GKE is integrated with Identity and Access Management.

GKE is integrated with Operations Suite.

GKE is integrated with Virtual Private Clouds.

The Google Cloud console provides insights into GKE clusters and their resources.

Google Kubernetes Engine features

GKE is fully managed.

Operating systems are optimized to scale quickly.

GKE Autopilot manages cluster configuration.

GKE auto upgrade ensures clusters have the latest stable version of Kubernetes.

GKE auto repair fixes unhealthy nodes.

GKE scales the cluster itself.

GKE is integrated with Cloud Build.

GKE is integrated with Identity and Access Management.

GKE is integrated with Operations Suite.

GKE is integrated with Virtual Private Clouds.

The Google Cloud console provides insights into GKE clusters and their resources.

Google Kubernetes Engine features

GKE is fully managed.

Operating systems are optimized to scale quickly.

GKE Autopilot manages cluster configuration.

GKE auto upgrade ensures clusters have the latest stable version of Kubernetes.

GKE auto repair fixes unhealthy nodes.

GKE scales the cluster itself.

GKE is integrated with Cloud Build.

GKE is integrated with Identity and Access Management.

GKE is integrated with Operations Suite.

GKE is integrated with Virtual Private Clouds.

The Google Cloud console provides insights into GKE clusters and their resources.

Google Kubernetes Engine features

GKE is fully managed.

Operating systems are optimized to scale quickly.

GKE Autopilot manages cluster configuration.

GKE auto upgrade ensures clusters have the latest stable version of Kubernetes.

GKE auto repair fixes unhealthy nodes.

GKE scales the cluster itself.

GKE is integrated with Cloud Build.

GKE is integrated with Identity and Access Management.

GKE is integrated with Operations Suite.

GKE is integrated with Virtual Private Clouds.

The Google Cloud console provides insights into GKE clusters and their resources.

Google Kubernetes Engine features

GKE is fully managed.

Operating systems are optimized to scale quickly.

GKE Autopilot manages cluster configuration.

GKE auto upgrade ensures clusters have the latest stable version of Kubernetes.

GKE auto repair fixes unhealthy nodes.

GKE scales the cluster itself.

GKE is integrated with Cloud Build.

GKE is integrated with Identity and Access Management.

GKE is integrated with Operations Suite.

GKE is integrated with Virtual Private Clouds.

The Google Cloud console provides insights into GKE clusters and their resources.

Google Kubernetes Engine features

GKE is fully managed.

Operating systems are optimized to scale quickly.

GKE Autopilot manages cluster configuration.

GKE auto upgrade ensures clusters have the latest stable version of Kubernetes.

GKE auto repair fixes unhealthy nodes.

GKE scales the cluster itself.

GKE is integrated with Cloud Build.

GKE is integrated with Identity and Access Management.

GKE is integrated with Operations Suite.

GKE is integrated with Virtual Private Clouds.

The Google Cloud console provides insights into GKE clusters and their resources.

Google Kubernetes Engine features

GKE is fully managed.

Operating systems are optimized to scale quickly.

GKE Autopilot manages cluster configuration.

GKE auto upgrade ensures clusters have the latest stable version of Kubernetes.

GKE auto repair fixes unhealthy nodes.

GKE scales the cluster itself.

GKE is integrated with Cloud Build.

GKE is integrated with Identity and Access Management.

GKE is integrated with Operations Suite.

GKE is integrated with Virtual Private Clouds.

The Google Cloud console provides insights into GKE clusters and their resources.

Google Kubernetes Engine features

GKE is fully managed.

Operating systems are optimized to scale quickly.

GKE Autopilot manages cluster configuration.

GKE auto upgrade ensures clusters have the latest stable version of Kubernetes.

GKE auto repair fixes unhealthy nodes.

GKE scales the cluster itself.

GKE is integrated with Cloud Build.

GKE is integrated with Identity and Access Management.

GKE is integrated with Operations Suite.

GKE is integrated with Virtual Private Clouds.

The Google Cloud console provides insights into GKE clusters and their resources.

Google Kubernetes Engine features

GKE is fully managed.

Operating systems are optimized to scale quickly.

GKE Autopilot manages cluster configuration.

GKE auto upgrade ensures clusters have the latest stable version of Kubernetes.

GKE auto repair fixes unhealthy nodes.

GKE scales the cluster itself.

GKE is integrated with Cloud Build.

GKE is integrated with Identity and Access Management.

GKE is integrated with Operations Suite.

GKE is integrated with Virtual Private Clouds.

The Google Cloud console provides insights into GKE clusters and their resources.

Google Kubernetes Engine features

GKE is fully managed.

Operating systems are optimized to scale quickly.

GKE Autopilot manages cluster configuration.

GKE auto upgrade ensures clusters have the latest stable version of Kubernetes.

GKE auto repair fixes unhealthy nodes.

GKE scales the cluster itself.

GKE is integrated with Cloud Build.

GKE is integrated with Identity and Access Management.

GKE is integrated with Operations Suite.

GKE is integrated with Virtual Private Clouds.

The Google Cloud console provides insights into GKE clusters and their resources.

Google Kubernetes Engine features

GKE is fully managed.

Operating systems are optimized to scale quickly.

GKE Autopilot manages cluster configuration.

GKE auto upgrade ensures clusters have the latest stable version of Kubernetes.

GKE auto repair fixes unhealthy nodes.

GKE scales the cluster itself.

GKE is integrated with Cloud Build.

GKE is integrated with Identity and Access Management.

GKE is integrated with Operations Suite.

GKE is integrated with Virtual Private Clouds.

The Google Cloud console provides insights into GKE clusters and their resources.

Quiz | Question 1 of 3

Question

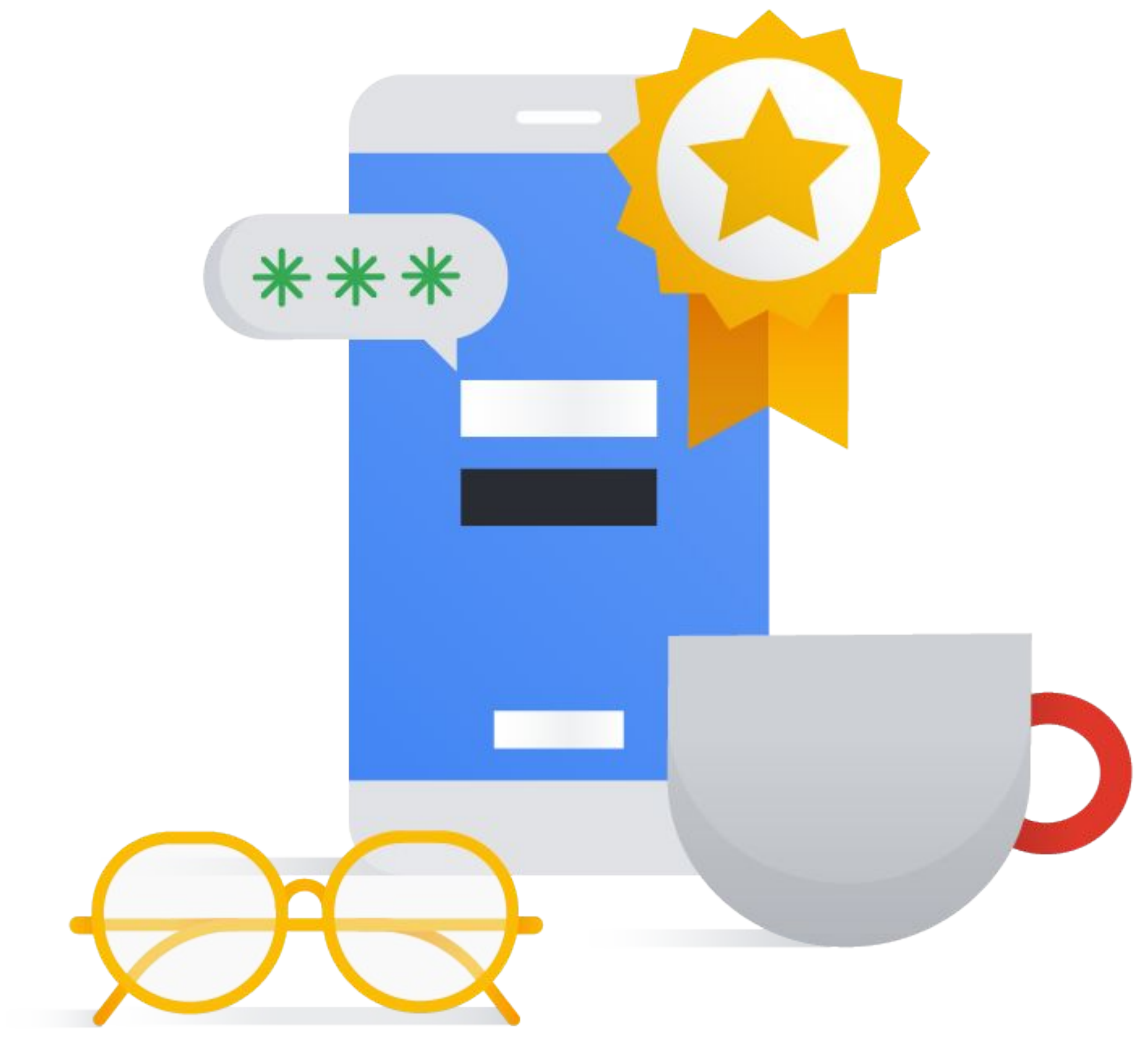
What is significant about the topmost layer in a container? Choose two options.

A. Reading from or writing to the topmost layer requires special privileges.

B. The topmost layer's contents are ephemeral. When the container is deleted, the contents are lost.

C. An application running in a container can only modify the topmost layer.

D. Reading from or writing to the topmost layer requires special software libraries.



Time for a short quiz

Quiz | Question 1 of 3

Answer

What is significant about the topmost layer in a container? Choose two options.

A. Reading from or writing to the topmost layer requires special privileges.

B. The topmost layer's contents are ephemeral. When the container is deleted, the contents are lost. ✓

C. An application running in a container can only modify the topmost layer. ✓

D. Reading from or writing to the topmost layer requires special software libraries.

Quiz | Question 2 of 3

Question

When using Kubernetes, you must describe the desired state you want, and Kubernetes's job is to make the deployed system conform to that desired state and keep it there despite failures. What is the name of this management approach?

A. Imperative configuration

B. Virtualization

C. Declarative configuration

D. Containerization

Quiz | Question 2 of 3

Answer

When using Kubernetes, you must describe the desired state you want, and Kubernetes's job is to make the deployed system conform to that desired state and keep it there despite failures. What is the name of this management approach?

A. Imperative configuration

B. Virtualization

C. Declarative configuration ✓

D. Containerization

Quiz | Question 3 of 3

Question

What is the name for the computers in a Google Kubernetes Engine cluster that run workloads?

A. Nodes

B. Control planes

C. Containers

D. Container images

Quiz | Question 3 of 3

Answer

What is the name for the computers in a Google Kubernetes Engine cluster that run workloads?

A. Nodes ✓

B. Control planes

C. Containers

D. Container images