Physical servers, virtual machines, containers, serverless

1 physical servers

2 virtual machines

3 containers

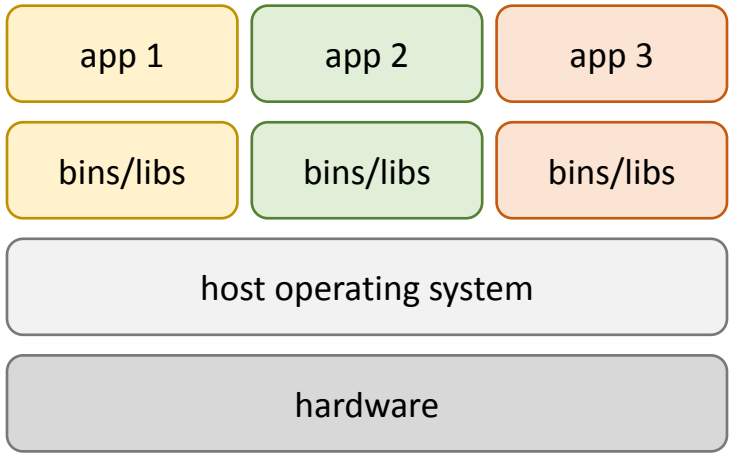
4 serverless

Physical servers, virtual machines, containers, serverless

pros

- Gives complete control of its software stack and hardware resources.
- Has a lot of processing power.
- Provides isolation between tenants (a bare-metal server).
- Eliminates the noisy neighbor phenomenon.
- Offers high security.

physical servers



cons

- Expensive.
- Hard to manage.
- Hard to scale.
- Hard to port.
- Slow to provision and boot.

good for

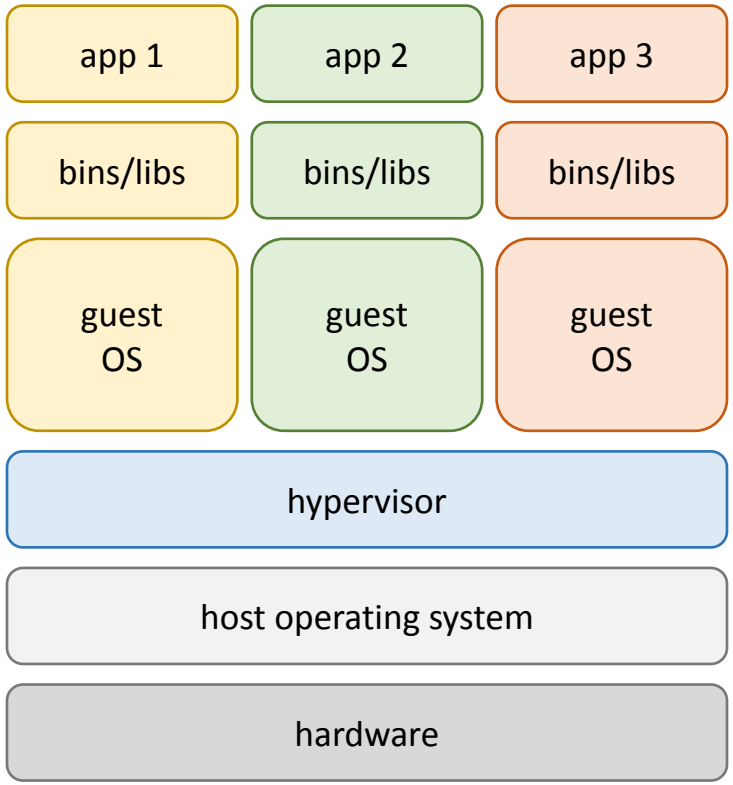
- When we need highly productive hardware to run our applications (e.g. data-intensive workloads).
- When we need to meet complex corporate security, compliance, and regulatory requirements.

Physical servers, virtual machines, containers, serverless

virtual machines

pros (compared to physical servers)

- Cheaper
- Easier to maintain.
- Easier to scale.
- Easier to port.
- Faster to provision and boot.



cons

- Exposed to the noisy neighbor problem.
- Less secure due to potential vulnerabilities in a hypervisor.

good for

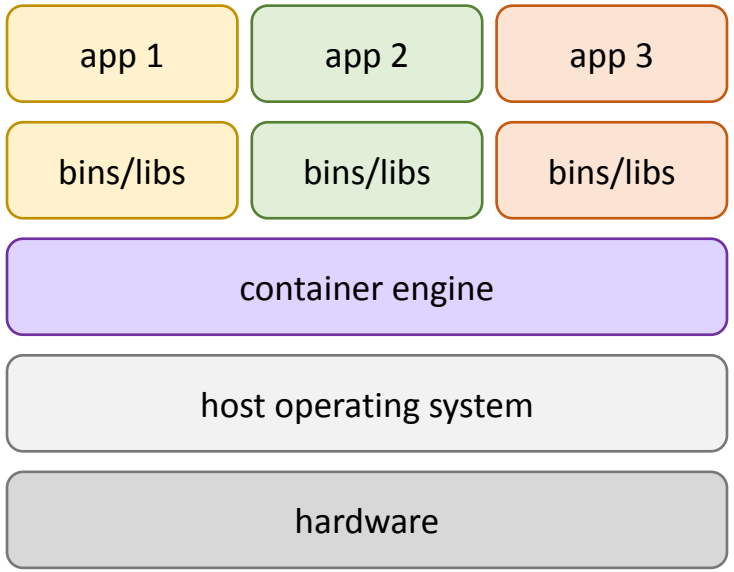
- Any workload. Configurations and prices vary a lot.

Physical servers, virtual machines, containers, serverless

containers

pros (compared to virtual machines)

- Lightweight.
- More scalable and portable.
- Easier to deploy and maintain at scale.
- Faster to start.



cons

- Possibly less secure.
- Less flexible in terms of dealing with multiple operating systems.

good for

- Any workload.

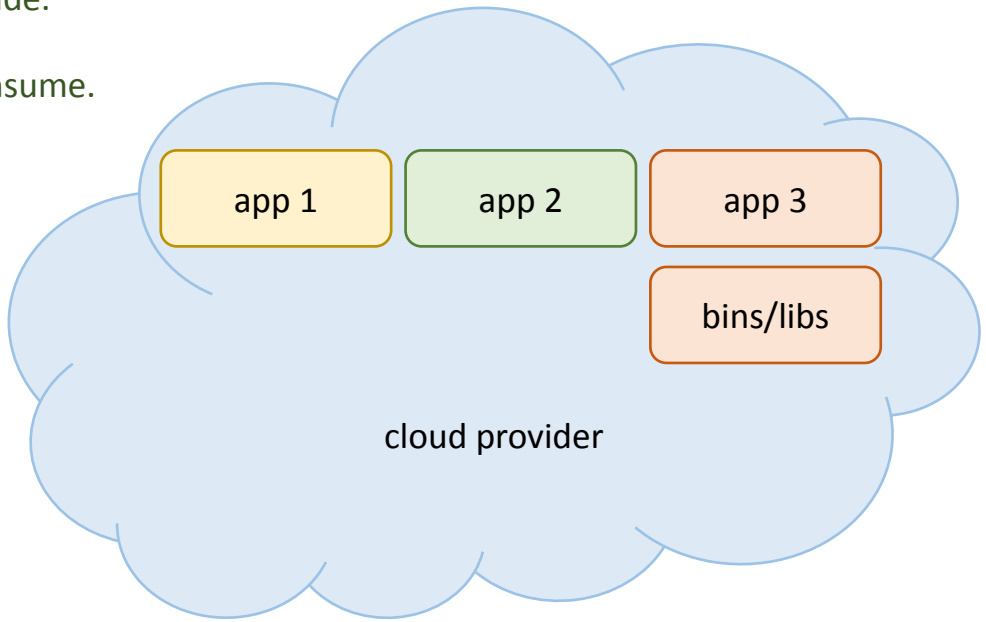
A container is a lightweight, standalone, executable package of software that includes everything needed to run an app: code, runtime, system tools, system libraries, and settings.
(from GCP docs)

Physical servers, virtual machines, containers, serverless

serverless

pros (compared to containers)

- No servers to provision, manage, or upgrade.
- We only pay for the compute time we consume.
- Automated scaling based on the load.
- Increased delivery speed.



cons

- Technical limitations (cold starts, invocation duration, memory size).
- Can be expensive when operating at scale.

good for

- Relatively small tasks that are performed in response to an event.