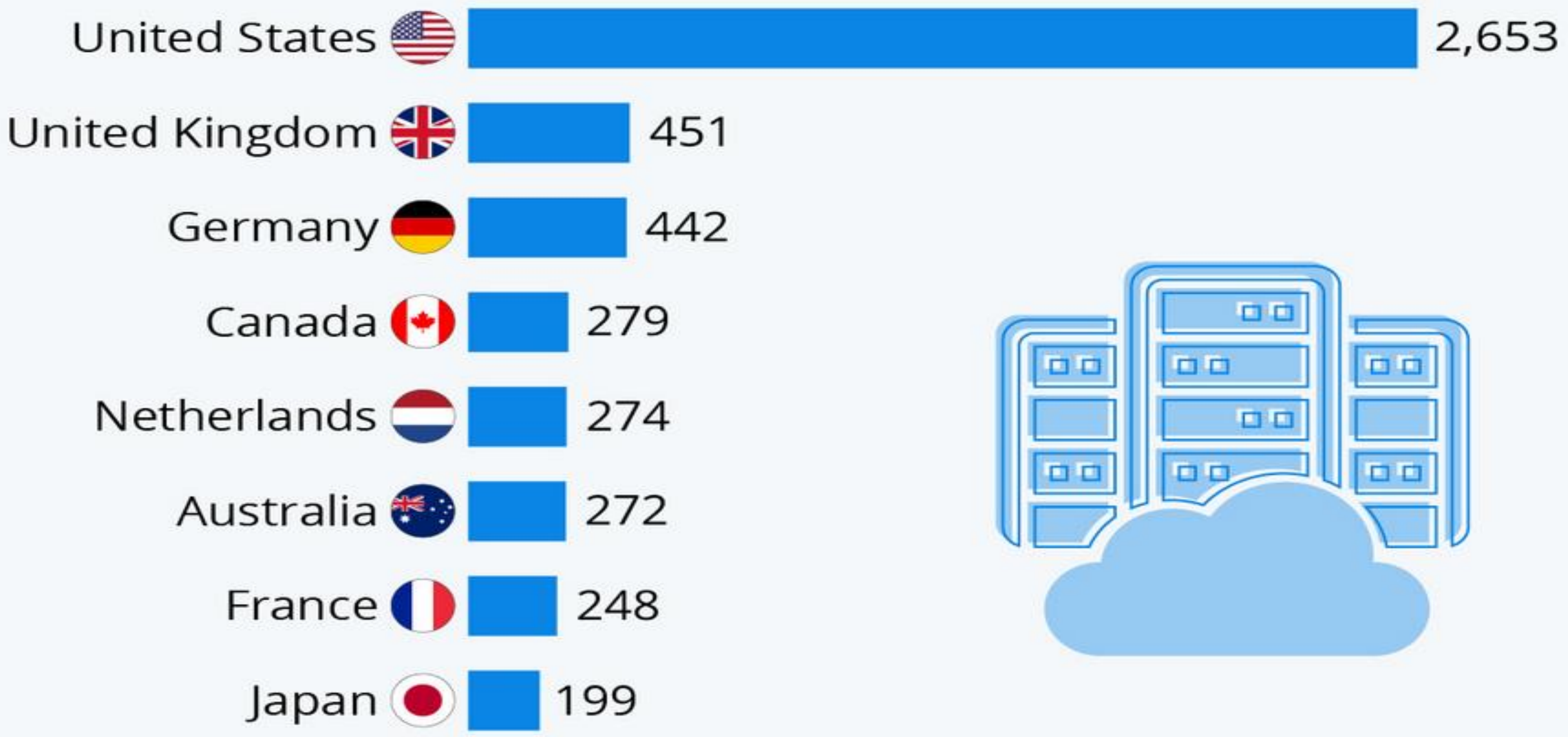
Amrita Vishwa Vidyapeetham

Amritapuri Campus

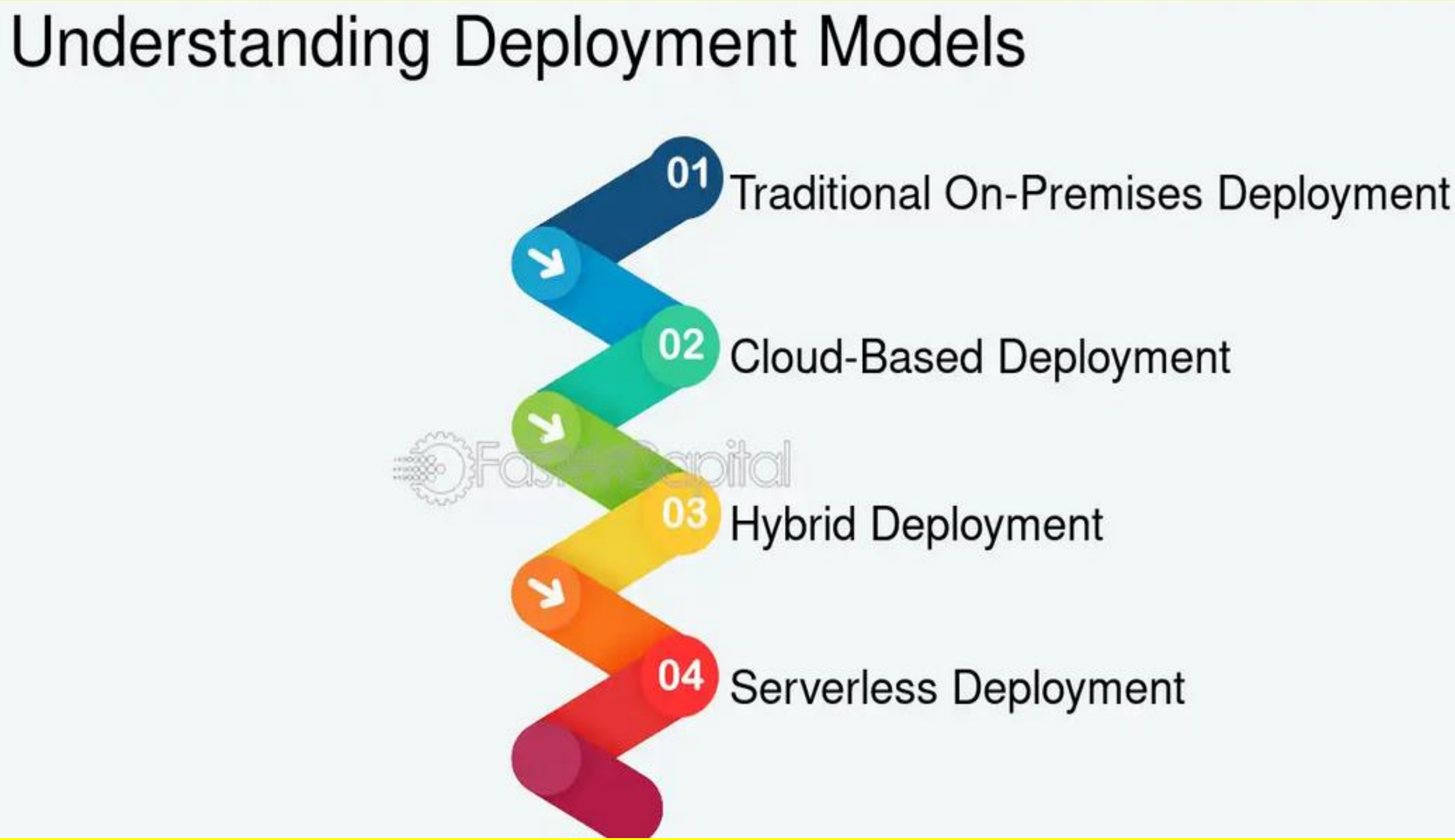
22AIE305: CLOUD COMPUTING

Which Countries Have The Most Data Centers?

Number of data centers per country as of February 09, 2021



Source: Cloudscene



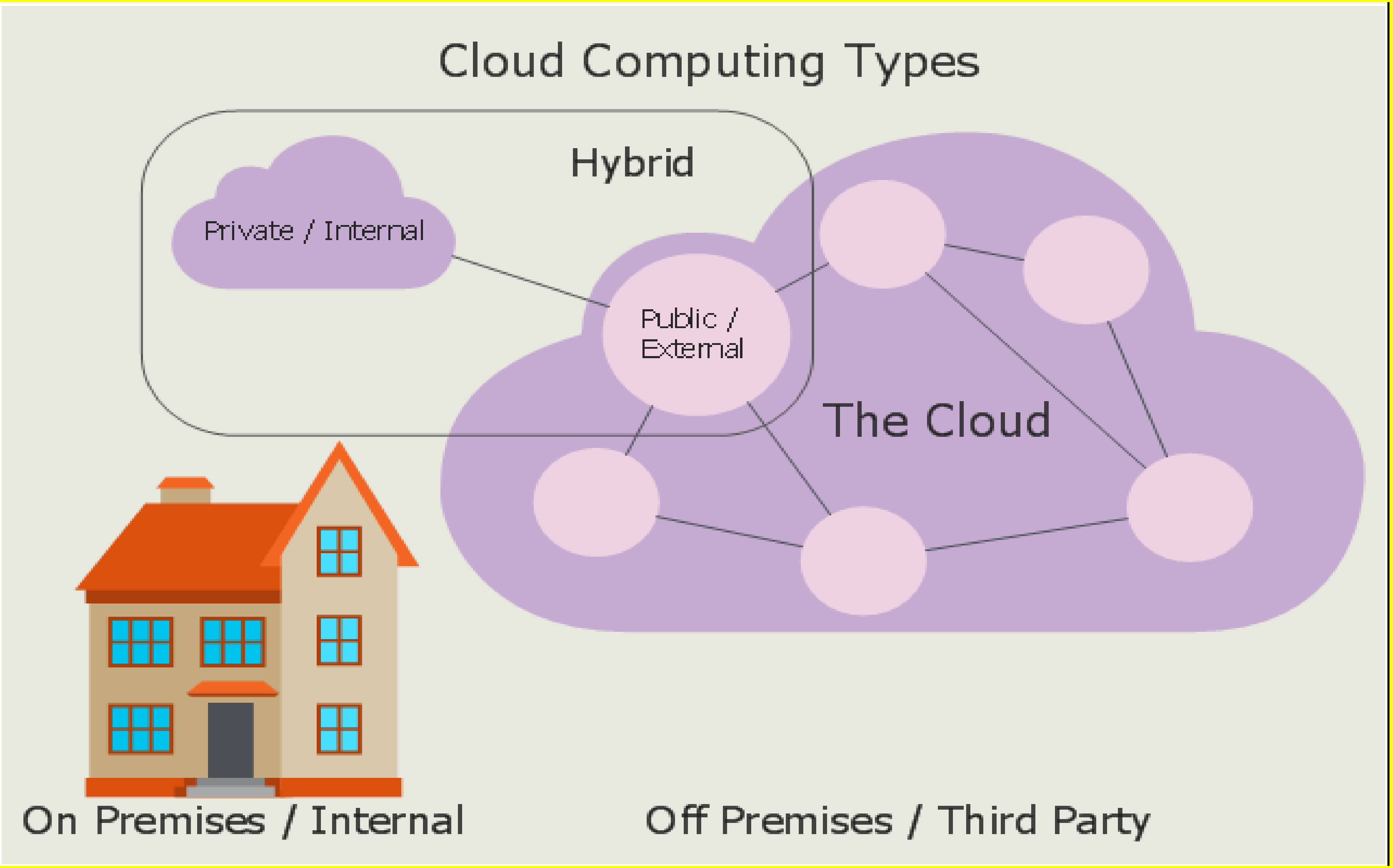
Understanding Deployment Models

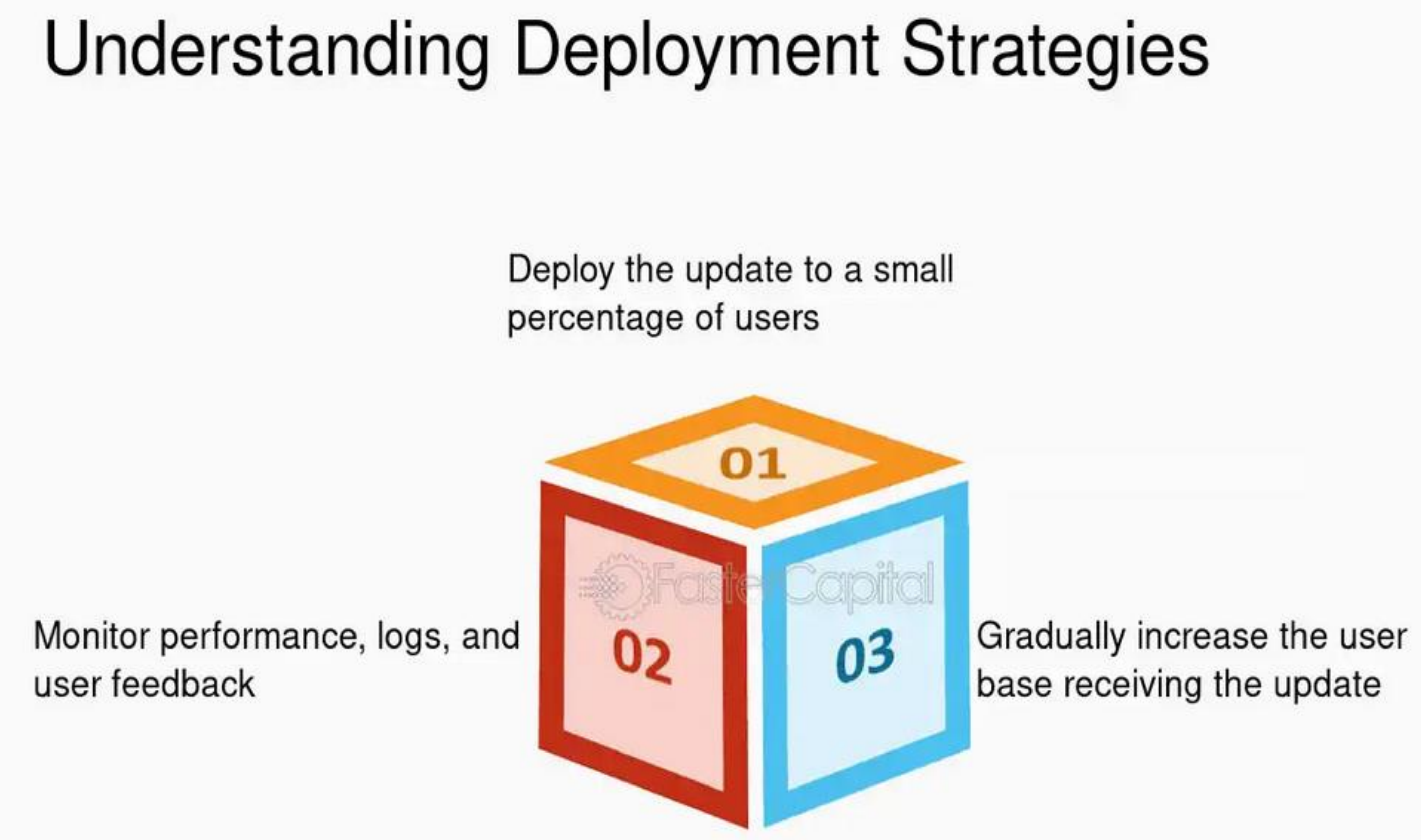
01 Traditional On-Premises Deployment

02 Cloud-Based Deployment

03 Hybrid Deployment

04 Serverless Deployment


Cloud Computing Types
Hybrid
Private / Internal
Public / External
The Cloud
On Premises / Internal
Off Premises / Third Party

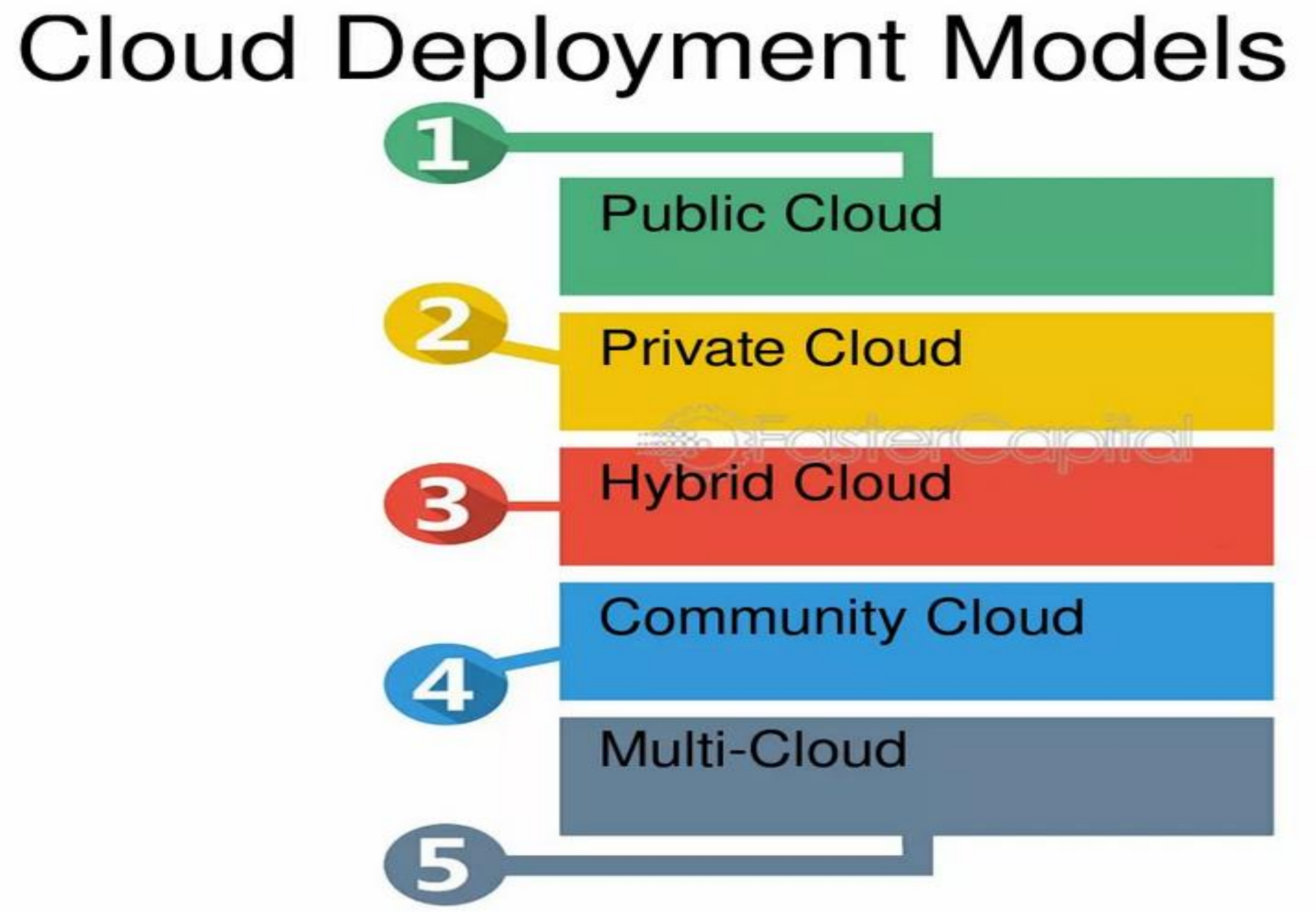


Understanding Deployment Strategies

01 Deploy the update to a small percentage of users

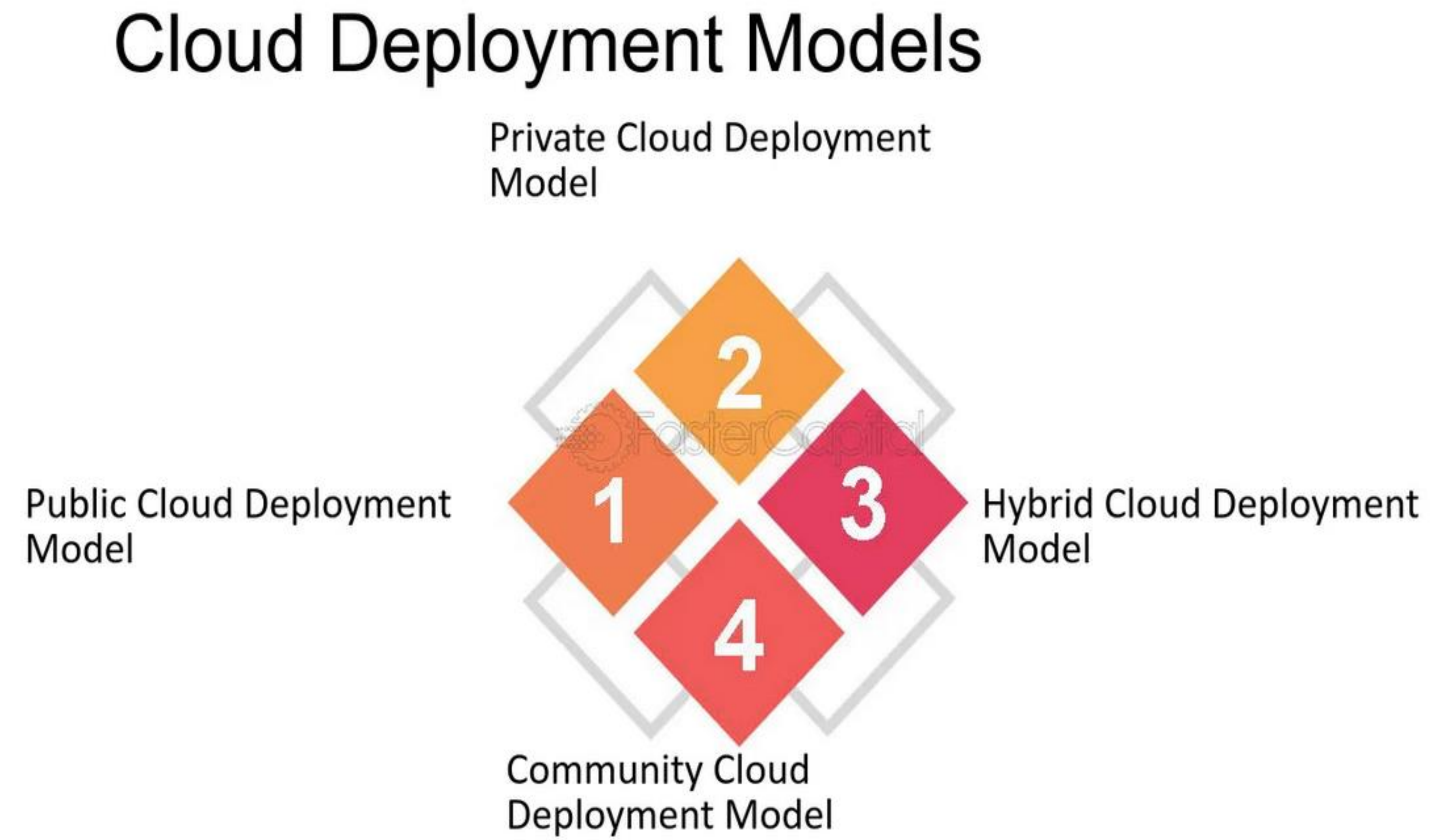
02 Monitor performance, logs, and user feedback

03 Gradually increase the user base receiving the update



Cloud Deployment Models

1. Public Cloud
2. Private Cloud
3. Hybrid Cloud
4. Community Cloud
5. Multi-Cloud



Cloud Deployment Models

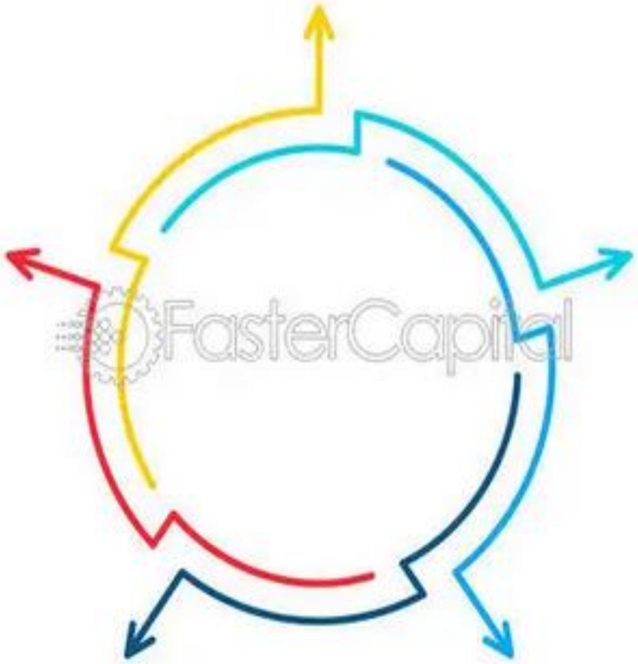
Private Cloud Deployment Model

Public Cloud Deployment Model

Hybrid Cloud Deployment Model

Community Cloud Deployment Model

Cloud Deployment Models



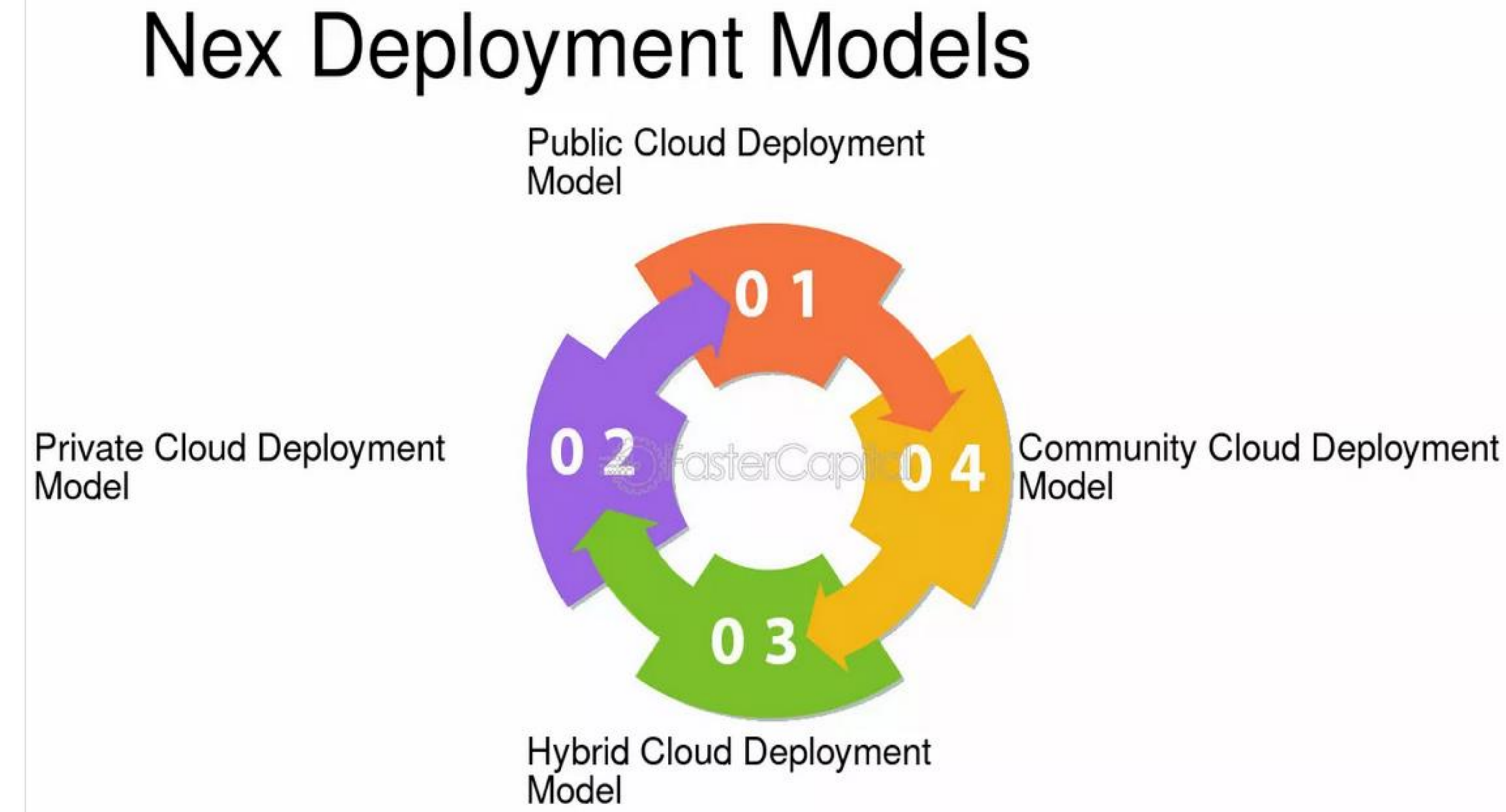
0 1 Public cloud - In this deployment model, a third-party cloud service provider delivers computing resources over the internet

0 2 Private cloud - A private cloud is a cloud environment that is dedicated to a single organization

0 3 Hybrid cloud - A hybrid cloud combines the benefits of both public and private clouds

0 4 Community cloud - This deployment model is designed for a group of organizations that have shared interests, such as regulatory compliance

0 5 Multi-cloud - A multi-cloud strategy involves using two or more cloud providers to meet an organization's needs



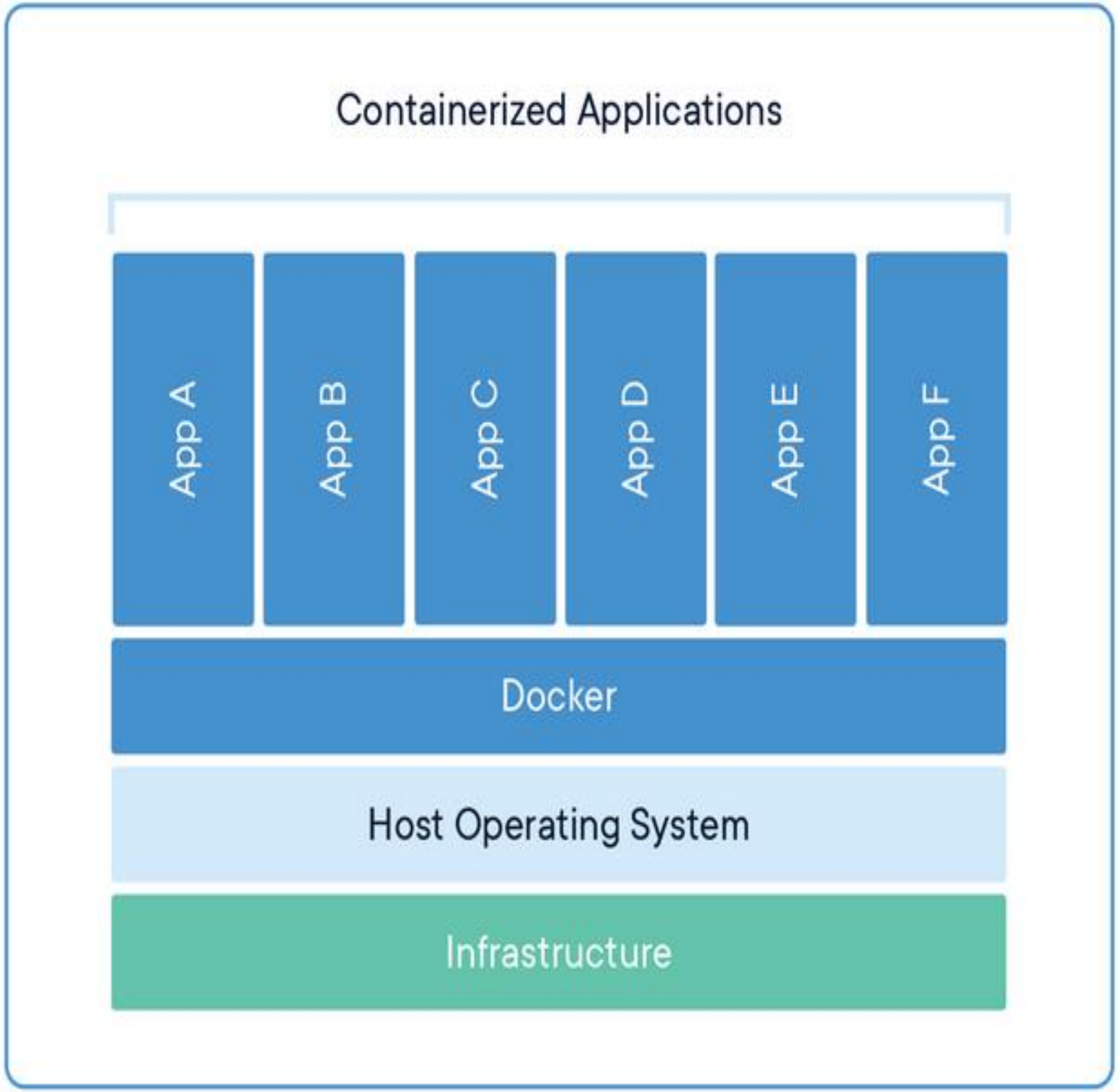
Nex Deployment Models

Public Cloud Deployment Model

Private Cloud Deployment Model

Community Cloud Deployment Model

Hybrid Cloud Deployment Model



Containerized Applications

App A

App B

App C

App D

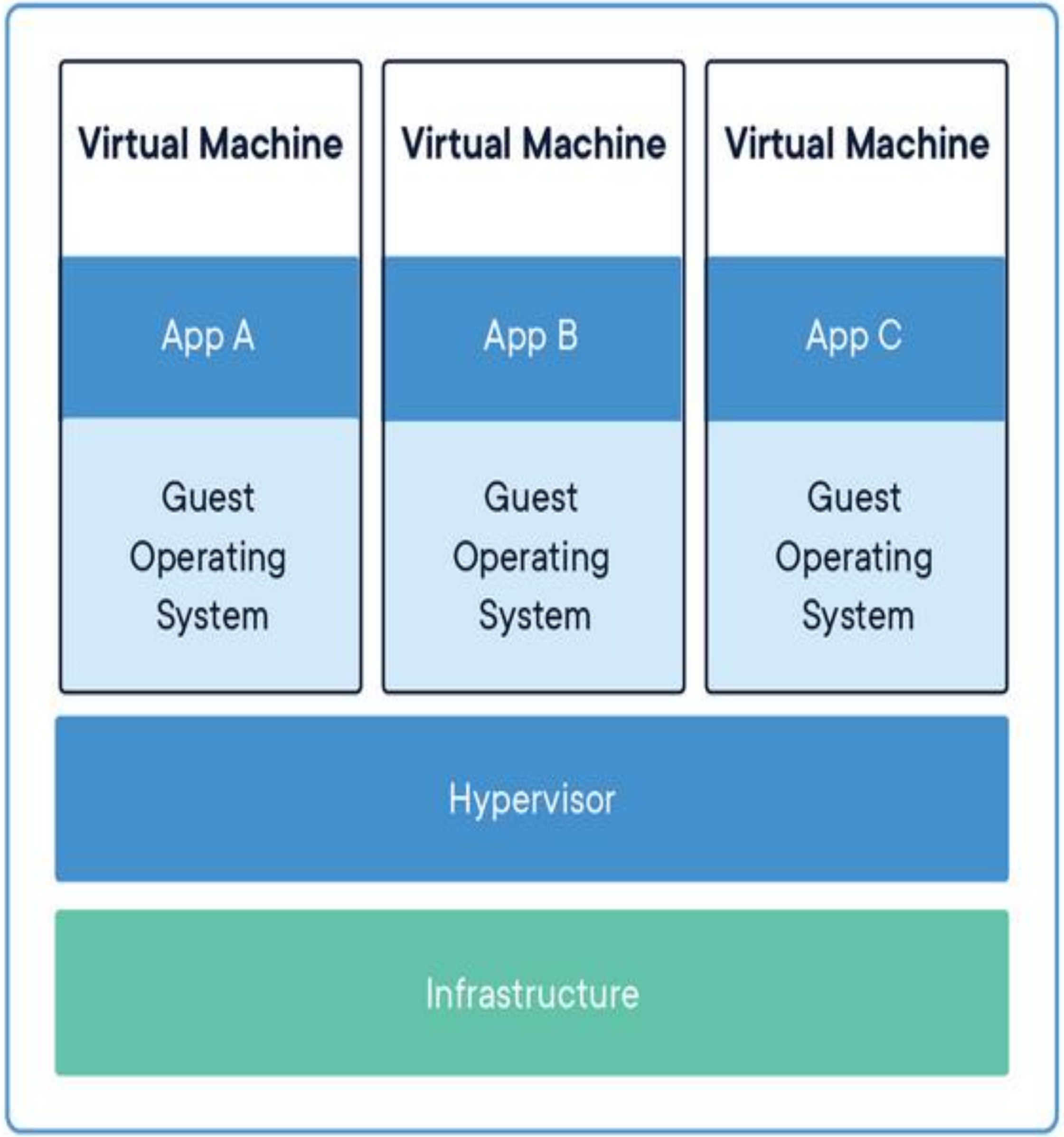
App E

App F

Docker

Host Operating System

Infrastructure



Virtual Machine

App A

Guest Operating System

Virtual Machine

App B

Guest Operating System

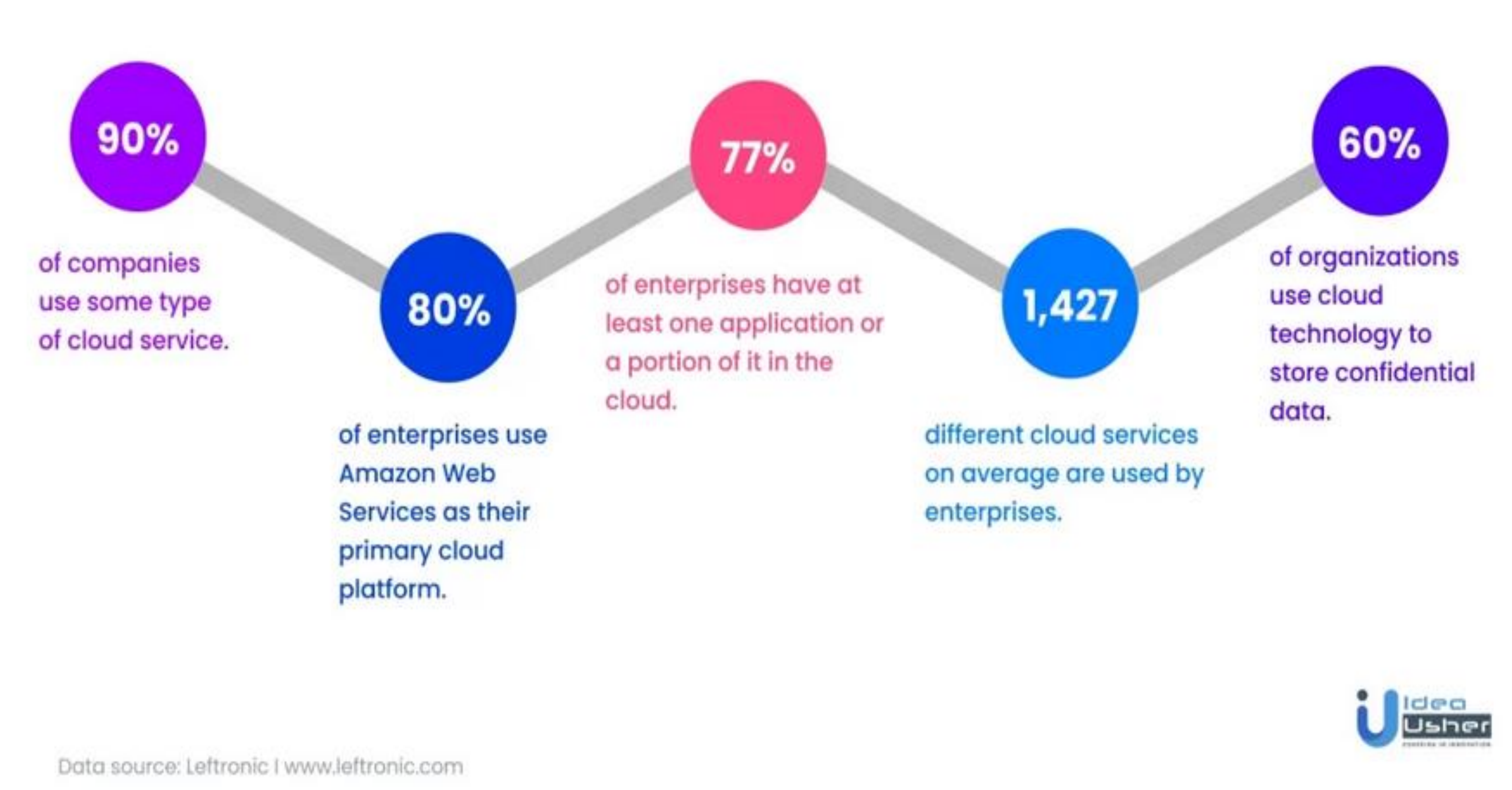
Virtual Machine

App C

Guest Operating System

Hypervisor

Infrastructure



90%

of companies use some type of cloud service.

80%

of enterprises use Amazon Web Services as their primary cloud platform.

77%

of enterprises have at least one application or a portion of it in the cloud.

1,427

different cloud services on average are used by enterprises.

60%

of organizations use cloud technology to store confidential data.

Data source: Leftronic I www.leftronic.com

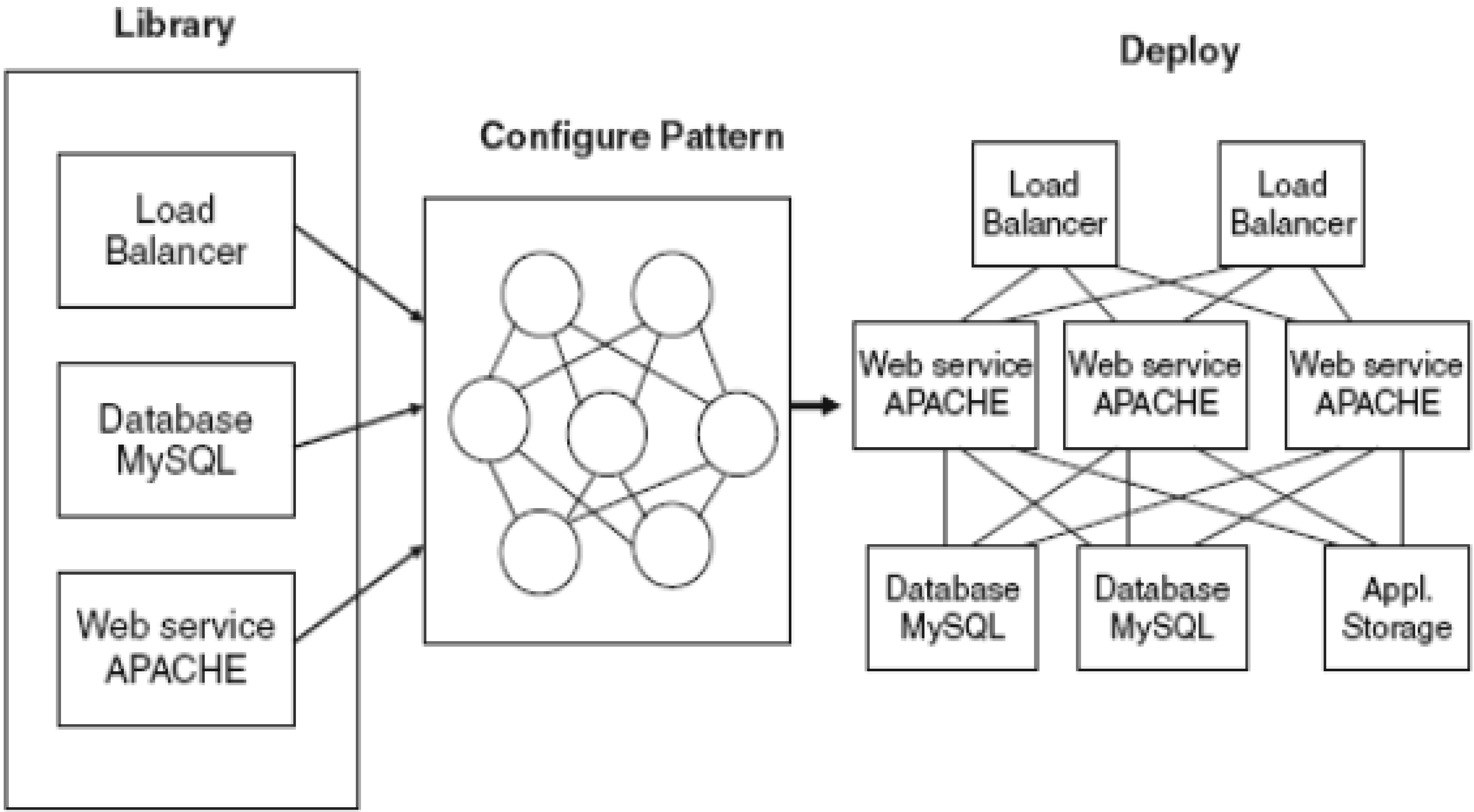
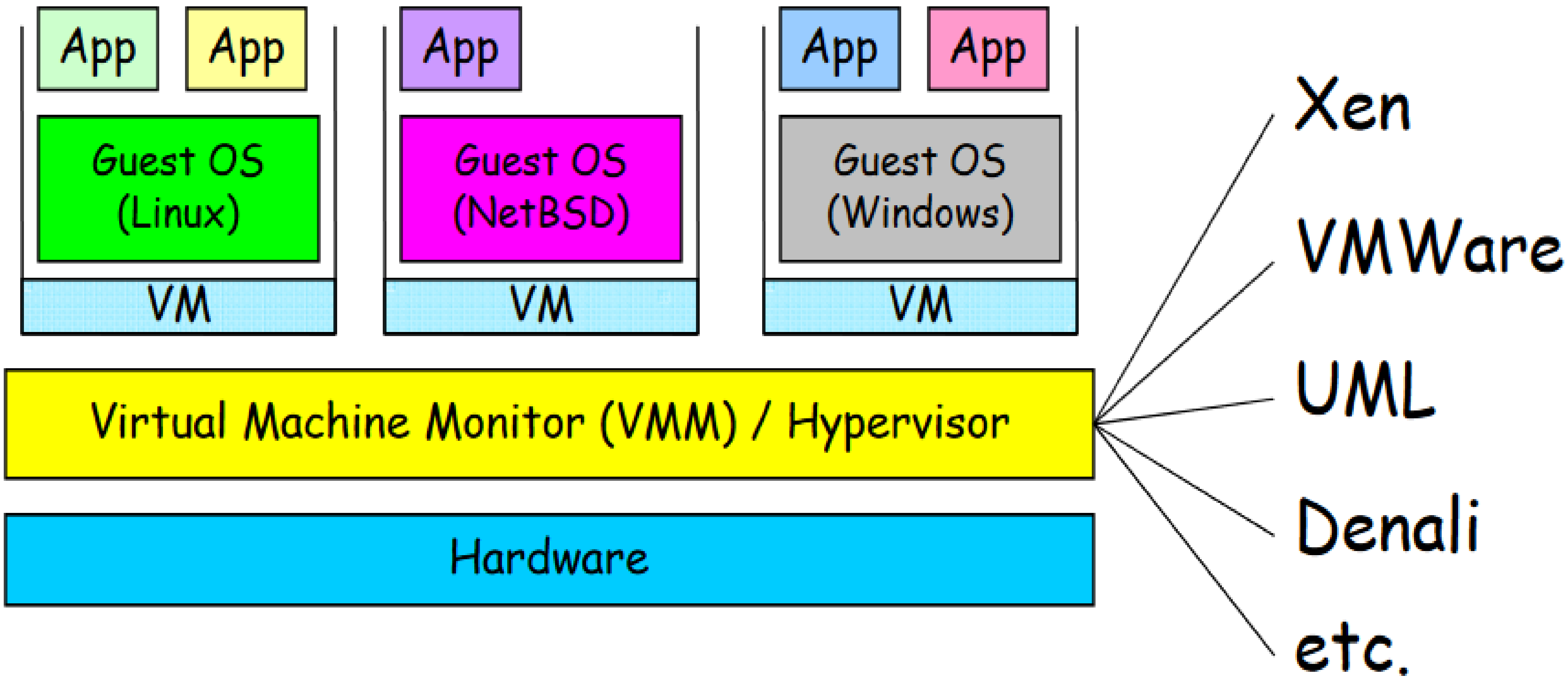


Figure 2. Deployment Strategy on Cloud for two tier architecture

VM technology allows multiple virtual machines to run on a single physical machine.



WHAT IS A CONTAINER?

Containers are a method of building, packaging and deploying software. A container includes all the code, runtime, libraries and everything else the containerized workload needs to run.

Container deployment is the act of pushing (ie. deploying) containers to their target environment, such as a cloud or on-premises server.

While a container might hold an entire application, in reality most container deployments are really multi-container deployments, meaning you are pushing multiple containers to a target environment.

For more dynamic, large-scale systems, you might deploy hundreds or even thousands of containers a day.

CONTAINERS ARE ELASTIC

Containers are designed to be scaled up and down quickly depending on the application needs.

This is because containers are often used as a method of building, packaging, and deploying microservices.

Microservices describe a software architecture that breaks up a large solution -- sometimes called a monolith or monolithic application -- into smaller logical units.

Each of these microservices runs independently in its own container.

There are myriad advantages to this software practice, including the ability to speed up deployments and subsequent code changes.

COMPILING CONTAINER IMAGE

Add the following actions into the Dockerfile

Use the command to ‘pull/use’ base container image from repository

Define the run environment, this will add OS related binaries as available in the repository which are needed for the final container to communicate through the orchestration layer

Define additional environmental requirements to be pulled from the repository or copy from the developer’s libraries

Set the default environment definitions that the container must use, including working directories, commands that will run in the container, communication ports to be opened

Finally copy the microservice/ app source code to the Cloud

AUTOMATE CONTAINER DEPLOYMENTS

Various configuration management or infrastructure as code tools offer the means to create scripts that automate or partially automate container deployments, often working in tandem with a container platform like Docker.

Each of these tools has their own particular methods -- and technical instructions for automating a container deployment or the application's configuration.

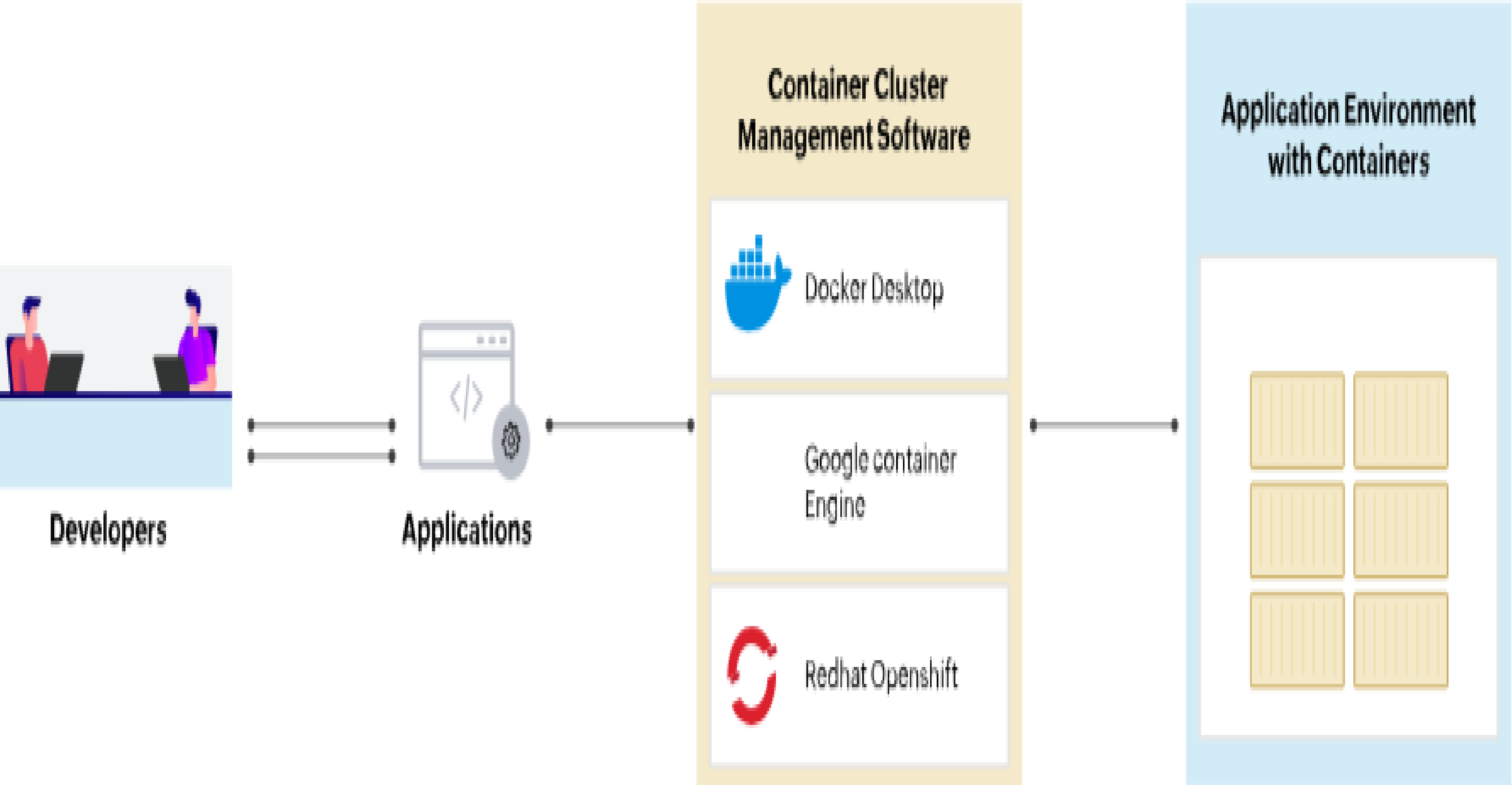
You can use configuration management or infrastructure as code tools to write scripts -- which go by different names on different platforms -- to automate certain tasks in your container deployment and management based on configuration best practices.

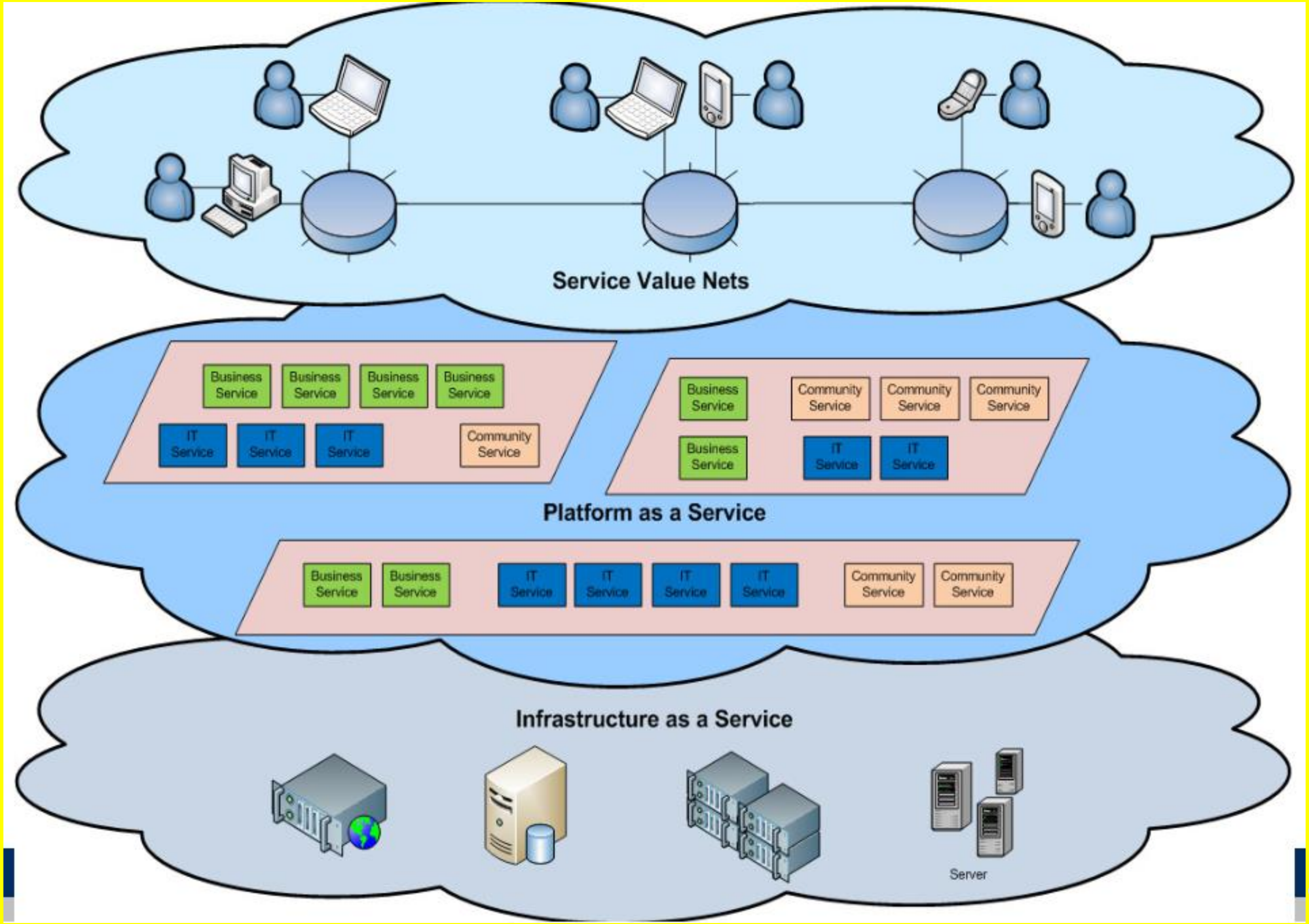
WHY USE CONTAINER DEPLOYMENT?

It can speed up application development and reduce the budget on IT operations teams, because they're abstracted away from the environments they run in.

As a result, containerized applications have become a popular choice among DevOps teams and other organizations that have moved away from traditional monolithic (or "legacy") approaches to software development.

Container deployments work well with continuous integration (CI) and continuous delivery (CD) processes and tools. (The related but distinct field of continuous deployment, another CD" acronym, takes continuous delivery a step further and fully automates deployment of code to production, without requiring manual approval.)


Developers
Applications
Container Cluster Management Software
Docker Desktop
Google container Engine
Redhat Openshift
Application Environment with Containers


Service Value Nets
Business Service
Business Service
Business Service
Business Service
IT Service
IT Service
IT Service
Community Service
Business Service
Community Service
Community Service
Community Service
Business Service
IT Service
IT Service
Platform as a Service
Business Service
Business Service
IT Service
IT Service
IT Service
IT Service
Community Service
Community Service
Infrastructure as a Service
Server

WHAT IS A RESOURCE?

Resources are deployable runtime apps and databases.

Each app resource contains a list of its components, which can be web services, workers, jobs, or static sites. Web services, workers, and jobs are built from source code repositories or container images and are hosted in containers.

Static sites are built from a directory of static files and hosted on the Cloud

WHAT IS CLOUD DEPLOYMENT?

Cloud deployment is the process of deploying applications, software, or services on remote servers that can be accessed through the internet. This process involves deploying code to a cloud platform, such as Amazon AWS, Google Cloud Platform (GCP), or Microsoft Azure.

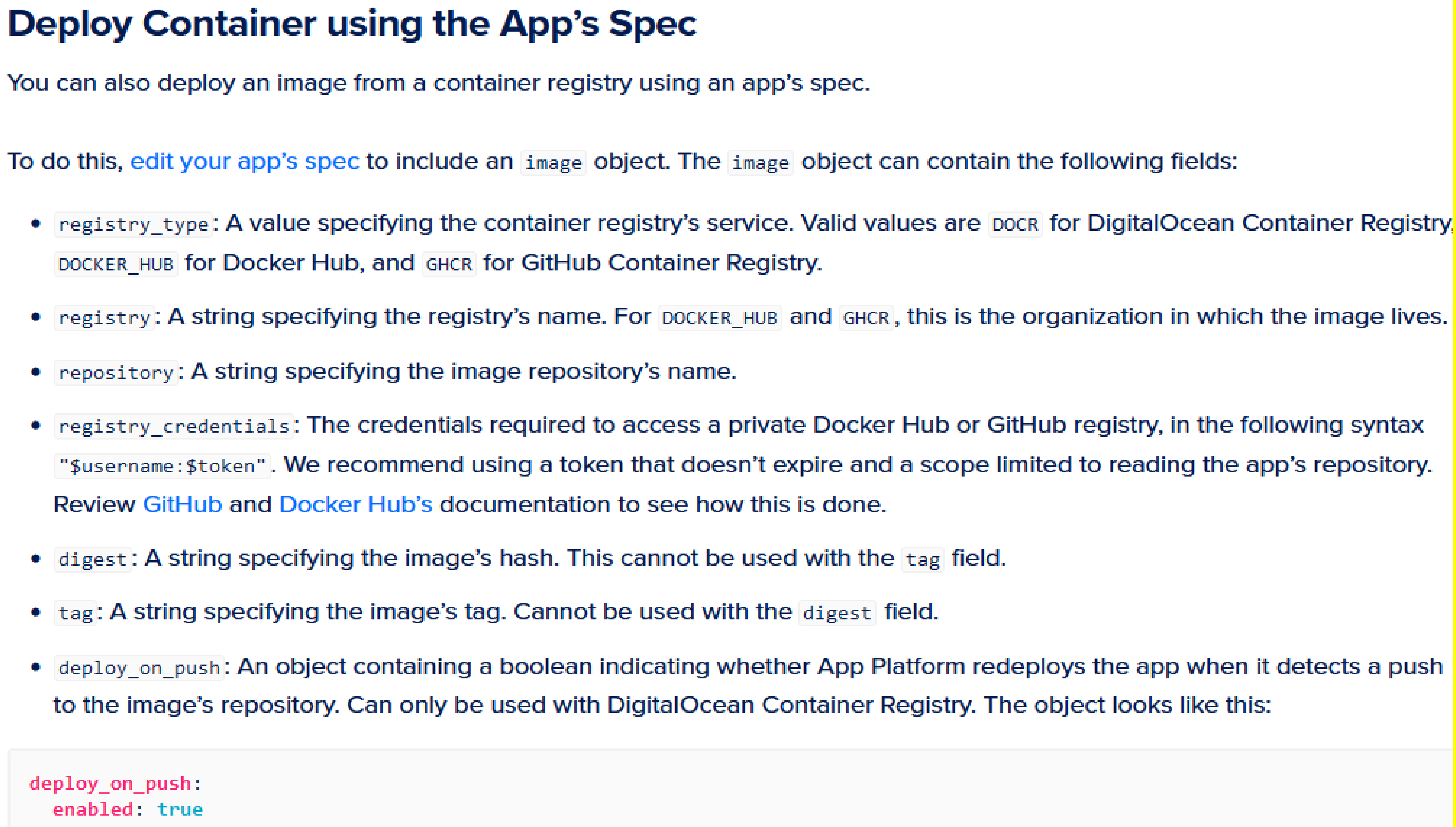
You can use the Command Line Interface (CLI) to push an app with a new or updated Docker image. Cloud provider then uses the Docker image to create containers for the app.

COMPILE MICROSERVICES

Once a microservice is successfully assessed for the varying scenarios (or use cases) that it must deliver to, it is ready to be compiled with requisite dependencies and binaries.

The list of dependencies can be built based on testing scenarios and errors addressed.

Tools from container platforms could be used to increase speed and reduce errors, for example, Cloud Build from Google can be used on Google Cloud Platform’s Google Container Engine GKE orchestrator.



Deploy Container using the App’s Spec

You can also deploy an image from a container registry using an app’s spec.

To do this, edit your app’s spec to include an `image` object. The `image` object can contain the following fields:

- `registry_type`: A value specifying the container registry’s service. Valid values are `DOCR` for DigitalOcean Container Registry, `DOCKER_HUB` for Docker Hub, and `GHCR` for GitHub Container Registry.
- `registry`: A string specifying the registry’s name. For `DOCKER_HUB` and `GHCR`, this is the organization in which the image lives.
- `repository`: A string specifying the image repository’s name.
- `registry_credentials`: The credentials required to access a private Docker Hub or GitHub registry, in the following syntax `"$username:$token"`. We recommend using a token that doesn’t expire and a scope limited to reading the app’s repository. Review GitHub and Docker Hub’s documentation to see how this is done.
- `digest`: A string specifying the image’s hash. This cannot be used with the `tag` field.
- `tag`: A string specifying the image’s tag. Cannot be used with the `digest` field.
- `deploy_on_push`: An object containing a boolean indicating whether App Platform redeploys the app when it detects a push to the image’s repository. Can only be used with DigitalOcean Container Registry. The object looks like this:

```
deploy_on_push:
  enabled: true
```

```
name: front-end
services:
- name: web
  image:
    registry_type: DOCKER_HUB
    registry: example-registry
    repository: example-repo
    registry_credentials:
"your_username:2YotnFZFEjr1zCsicMWpAA"
    digest:
sha256:1234567890abcdef1234567890abcdef123456789
0abcdef1234567890abcdef
```