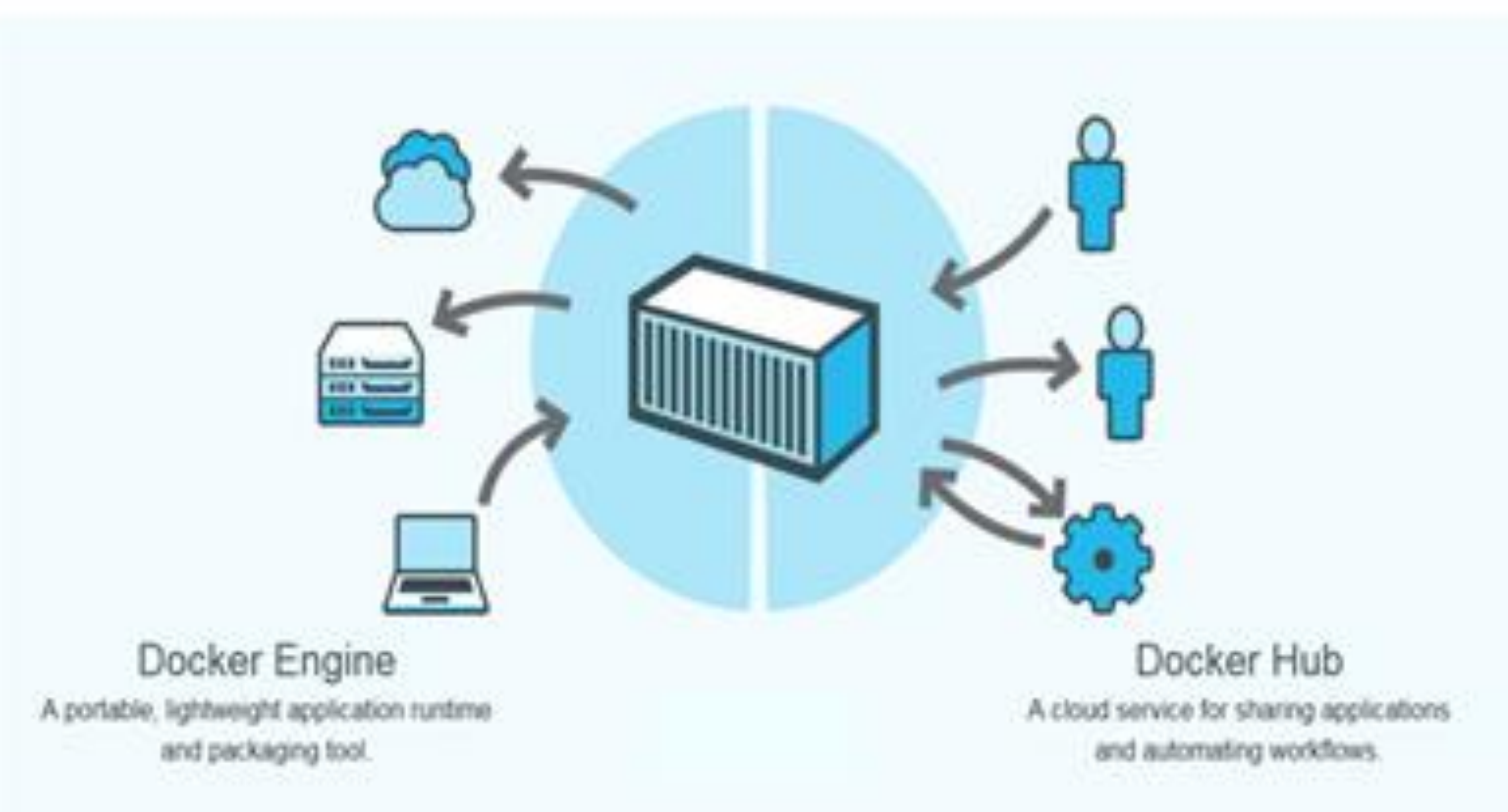
Docker Introduction

VISHWANATH M S

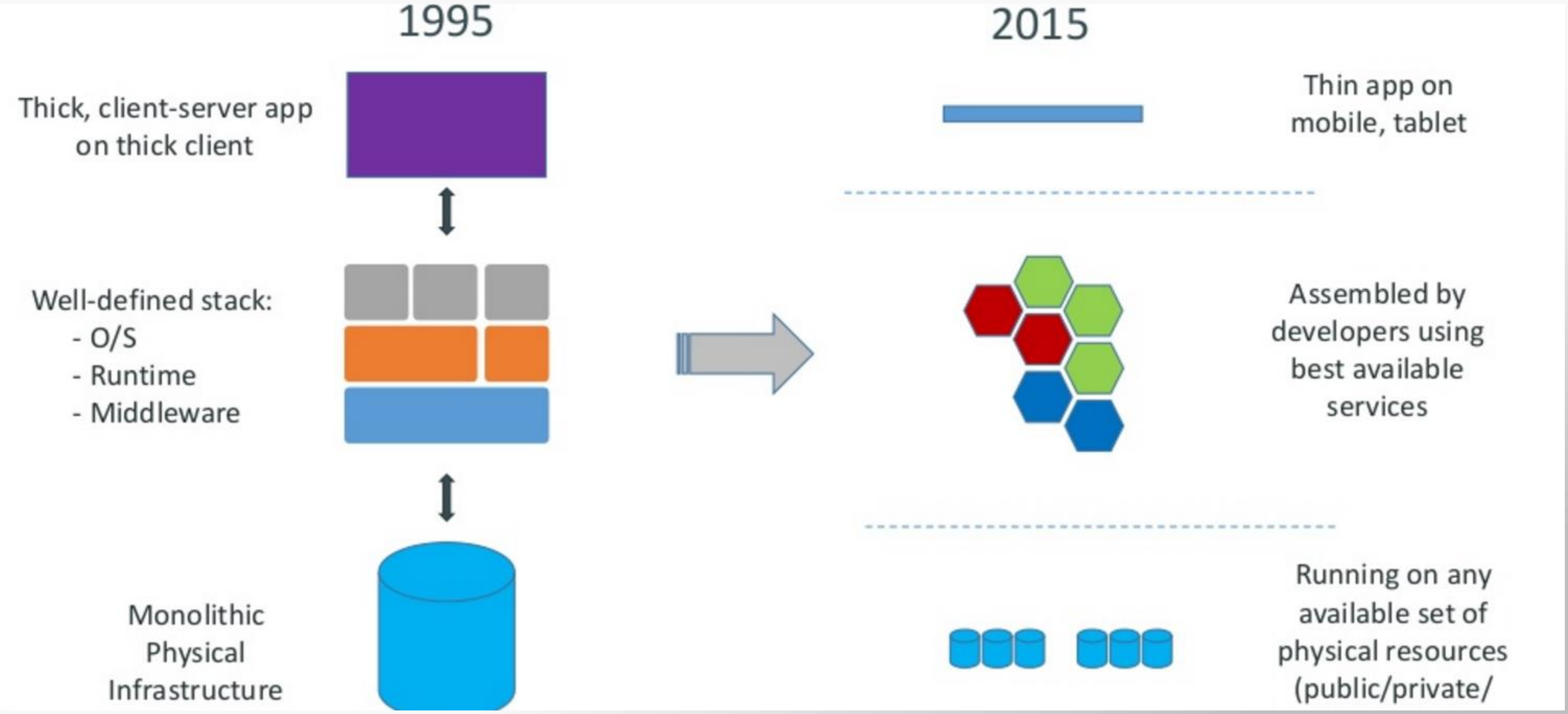
VISHWACLOUDLAB.COM

Docker

Open platform for developers and system administrators to build and test cloud applications

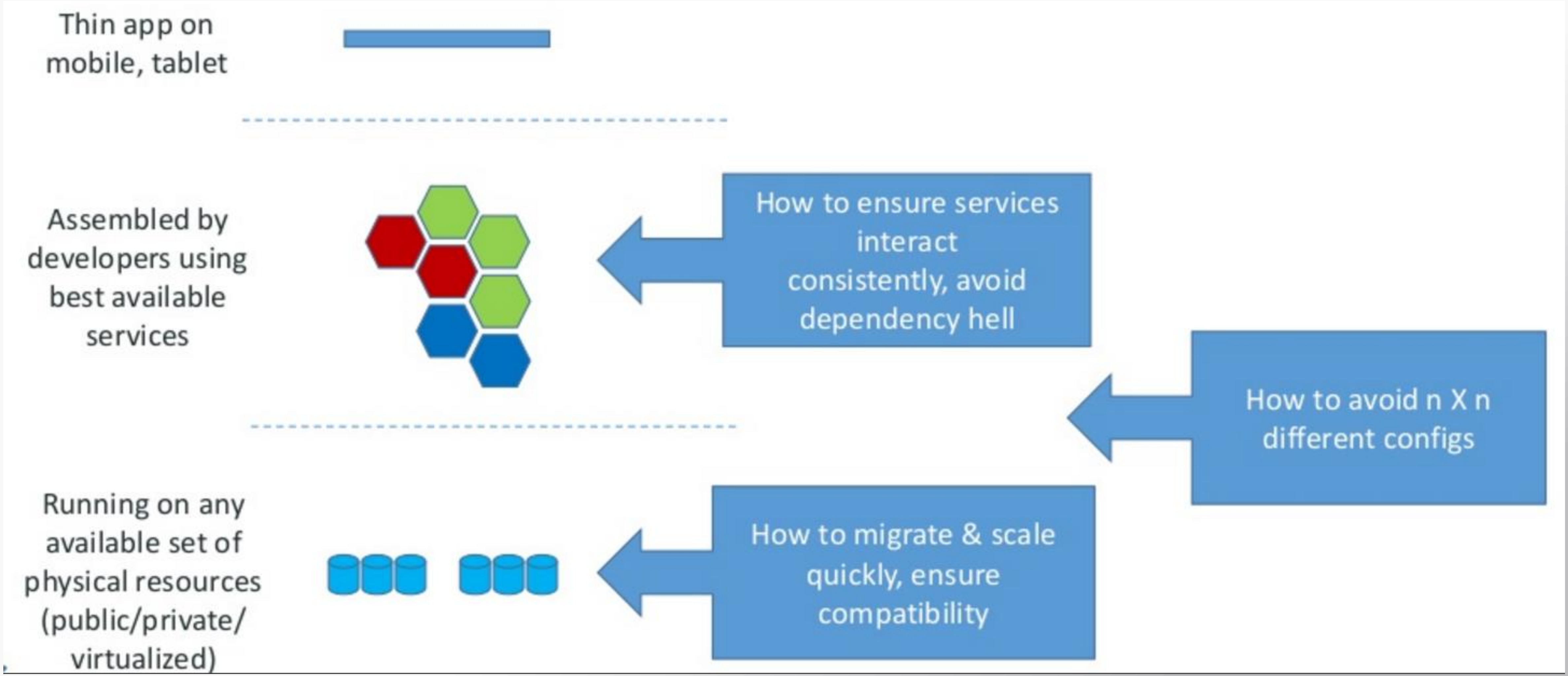

Docker Engine
A portable, lightweight application runtime and packaging tool.
Docker Hub
A cloud service for sharing applications and automating workflows.

Motivation for the Creation of Containers



1995
2015
Thick, client-server app on thick client
Well-defined stack:
- O/S
- Runtime
- Middleware
Monolithic Physical Infrastructure
Thin app on mobile, tablet
Assembled by developers using best available services
Running on any available set of physical resources (public/private/

The challenge



Thin app on mobile, tablet

Assembled by developers using best available services

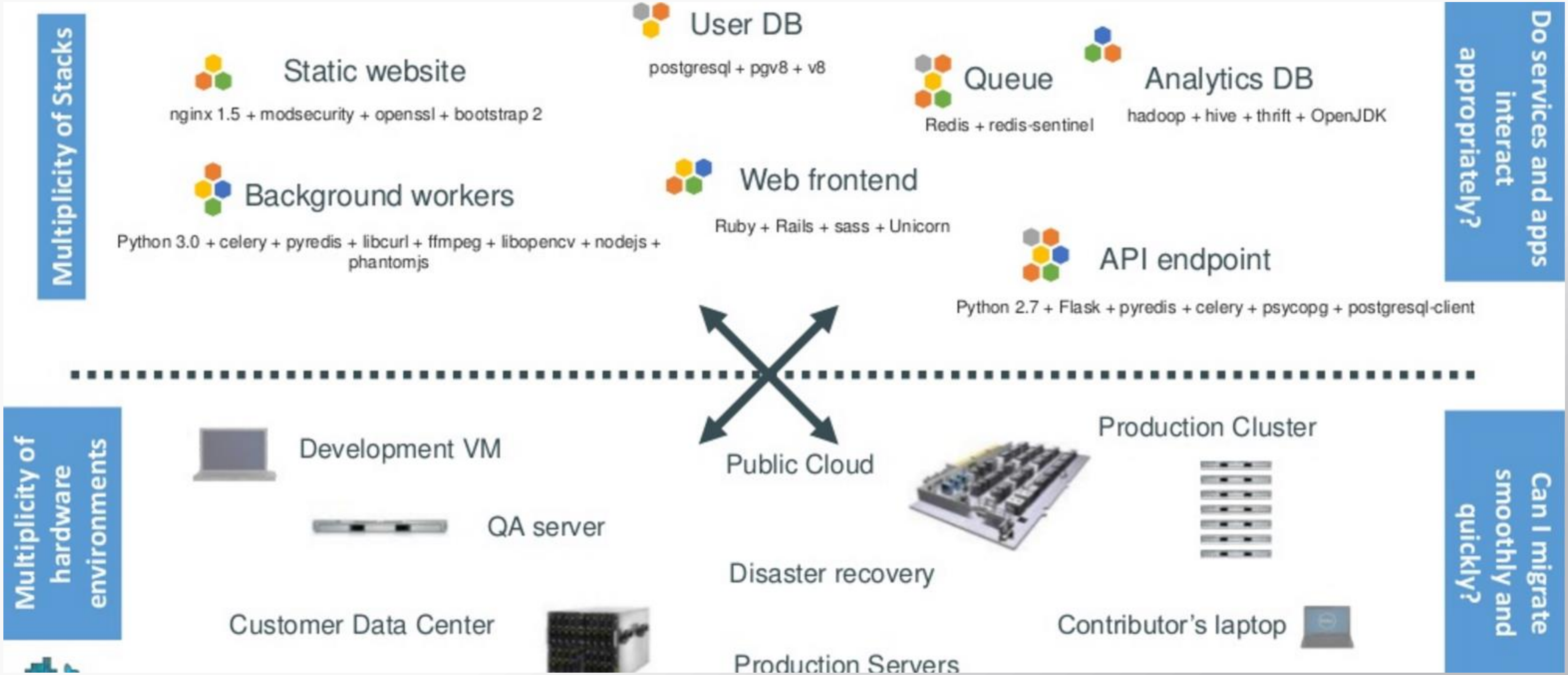
Running on any available set of physical resources (public/private/ virtualized)

How to ensure services interact consistently, avoid dependency hell

How to avoid n X n different configs

How to migrate & scale quickly, ensure compatibility

The challenge continued



Multiplicity of Stacks

Static website

nginx 1.5 + modsecurity + openssl + bootstrap 2

User DB

postgresql + pgv8 + v8

Queue

Redis + redis-sentinel

Analytics DB

hadoop + hive + thrift + OpenJDK

Background workers

Python 3.0 + celery + pyredis + libcurl + ffmpeg + libopencv + nodejs + phantomjs

Web frontend

Ruby + Rails + sass + Unicorn

API endpoint

Python 2.7 + Flask + pyredis + celery + psycopg + postgresql-client

Do services and apps interact appropriately?

Multiplicity of hardware environments

Development VM

QA server

Customer Data Center

Public Cloud

Disaster recovery

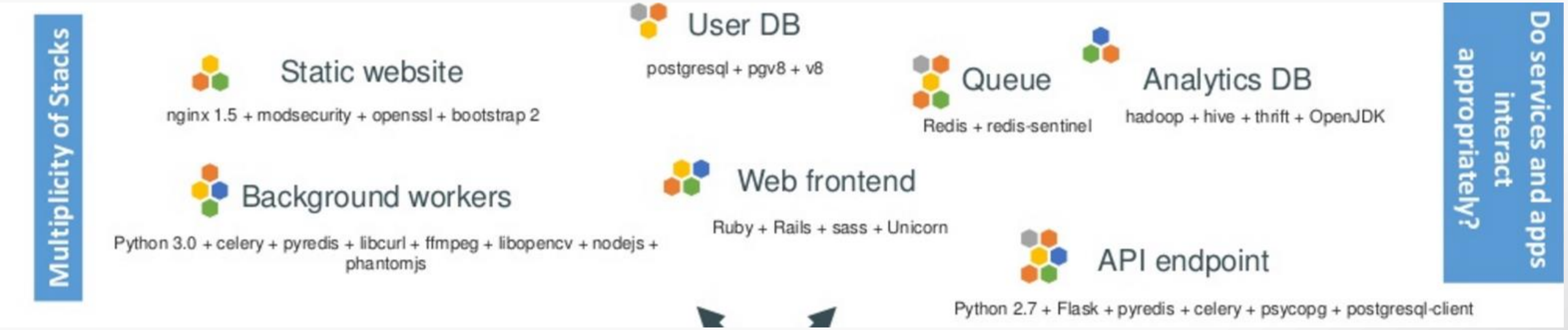
Production Servers

Production Cluster

Contributor's laptop

Can I migrate smoothly and quickly?

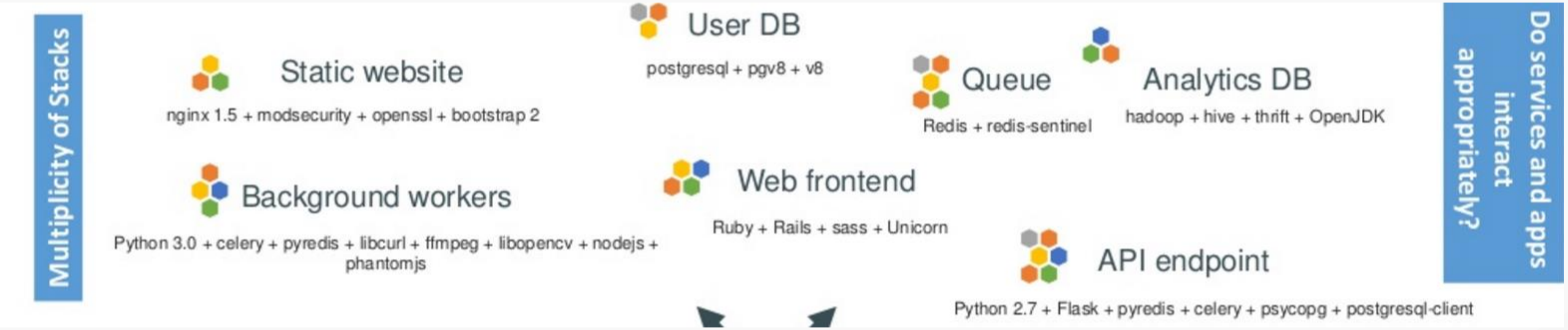
What were some of these technologies??


Multiplicity of Stacks
Static website
nginx 1.5 + modsecurity + openssl + bootstrap 2
User DB
postgresql + pgv8 + v8
Queue
Redis + redis-sentinel
Analytics DB
hadoop + hive + thrift + OpenJDK
Background workers
Python 3.0 + celery + pyredis + libcurl + ffmpeg + libopencv + nodejs + phantomjs
Web frontend
Ruby + Rails + sass + Unicorn
API endpoint
Python 2.7 + Flask + pyredis + celery + psycopg + postgresql-client
Do services and apps interact appropriately?

Static web: nginx is a fast webserver, generally faster than Apache

Queue: Redis = in-memory **data structure store**, used as a database, **cache** and message broker.

What were some of these technologies??



Multiplicity of Stacks

Static website
nginx 1.5 + modsecurity + openssl + bootstrap 2

User DB
postgresql + pgv8 + v8

Queue
Redis + redis-sentinel

Analytics DB
hadoop + hive + thrift + OpenJDK

Background workers
Python 3.0 + celery + pyredis + libcurl + ffmpeg + libopencv + nodejs + phantomjs

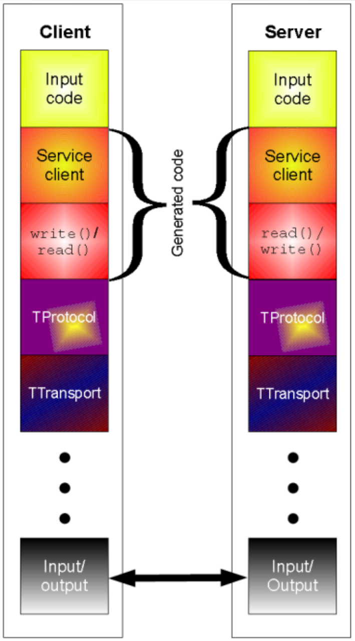
Web frontend
Ruby + Rails + sass + Unicorn

API endpoint
Python 2.7 + Flask + pyredis + celery + psycopg + postgresql-client

Do services and apps interact appropriately?

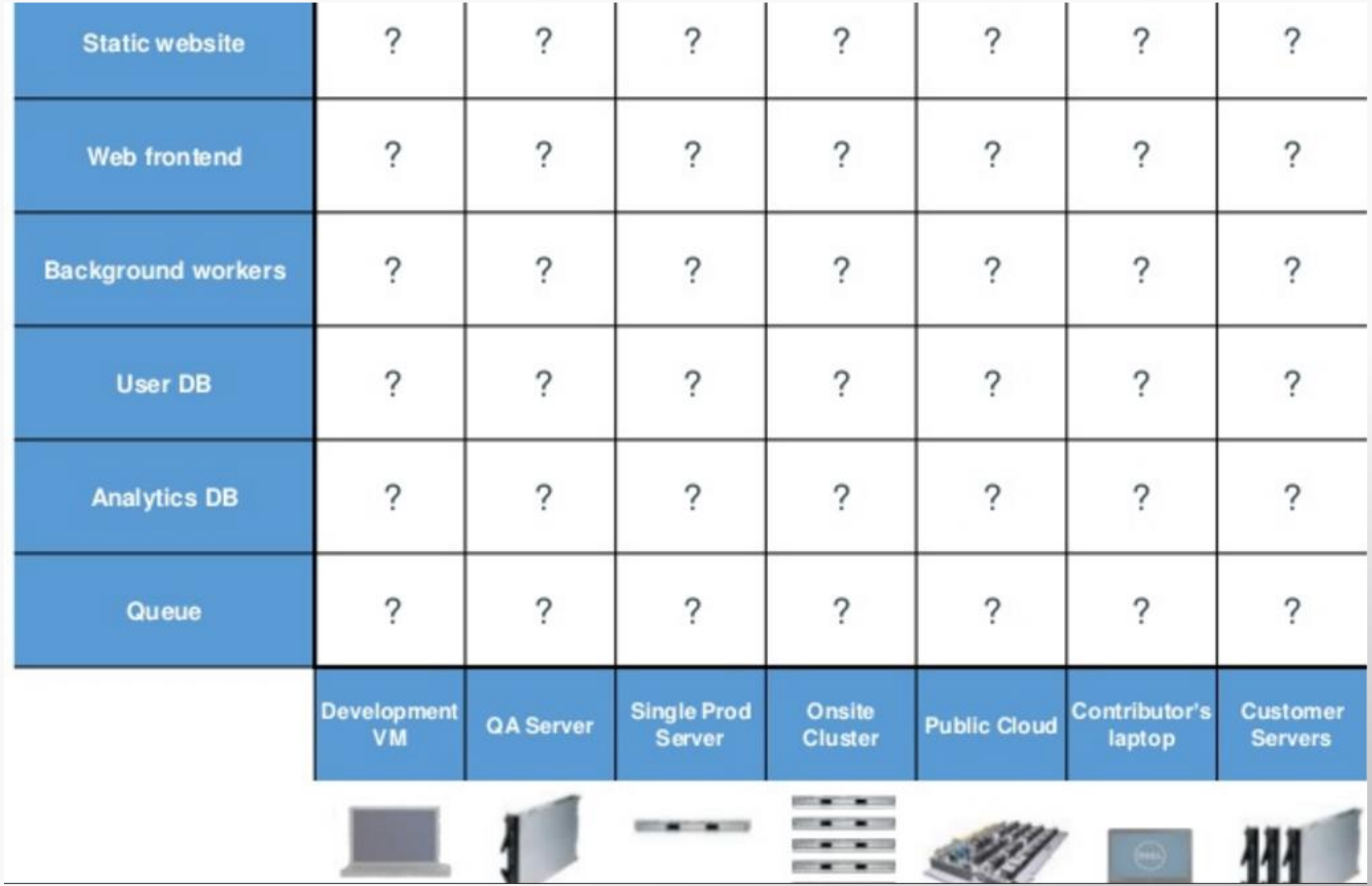
Analytics DB:

- Hadoop = distributed storage and processing of dataset of big data using MapReduce programming model
- Hive = Hadoop based Database, offer SQL-like interfaces with HDFS-based data
  - ---allows these database developers or data analysts to use Hadoop without knowing the Java programming language or MapReduce. Now, instead of challenging MapReduce code, you can design a star schema data warehouse or a normalized database.
- Thrift = used to define and create services for numerous languages.[2]
  - It is used as a remote procedure call (RPC) framework and was developed at Facebook for "scalable cross-language services development".



Looking for all kinds of solutions…

Too many to consider



Static website	?	?	?	?	?	?	?
Web frontend	?	?	?	?	?	?	?
Background workers	?	?	?	?	?	?	?
User DB	?	?	?	?	?	?	?
Analytics DB	?	?	?	?	?	?	?
Queue	?	?	?	?	?	?	?
	Development VM	QA Server	Single Prod Server	Onsite Cluster	Public Cloud	Contributor's laptop	Customer Servers

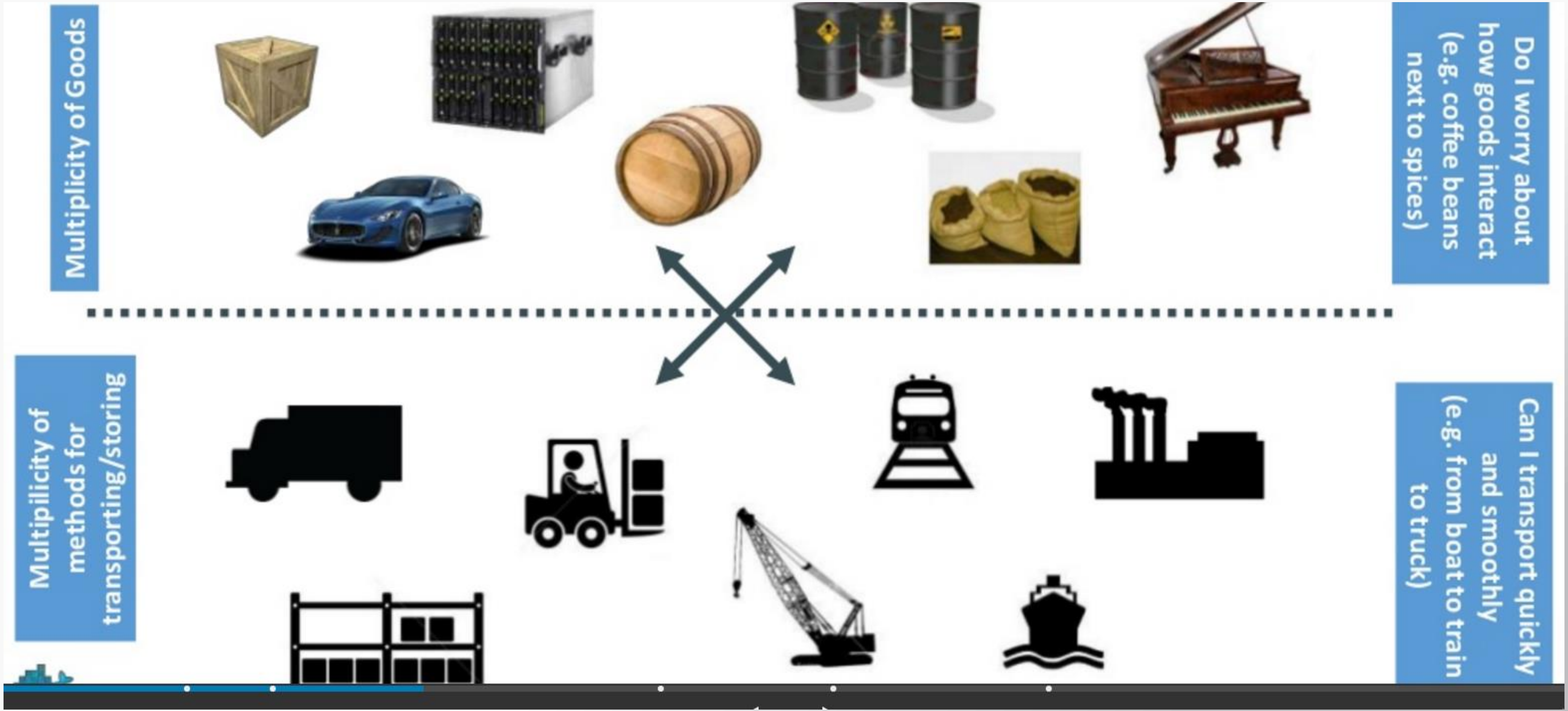
THE ANSWER

Something called a CONTAINER ----which is the business Docker has created.

Huh??? Container

Here is an analogy that Docker use’s to let you understand….

Understanding….an analogy
…cargo transport pre-1960



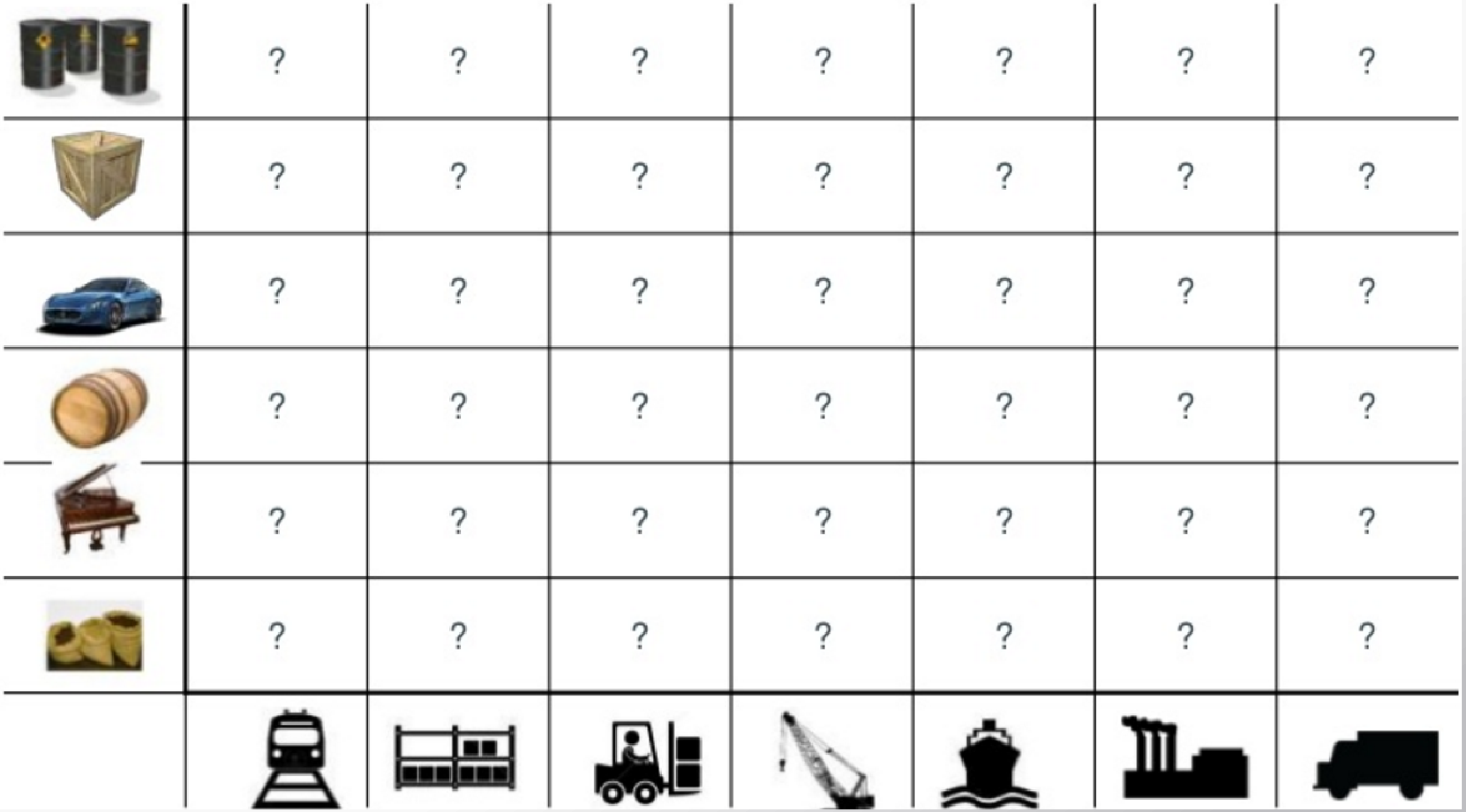
Multiplicity of Goods

Do I worry about how goods interact (e.g. coffee beans next to spices)

Multipilicity of methods for transporting/storing

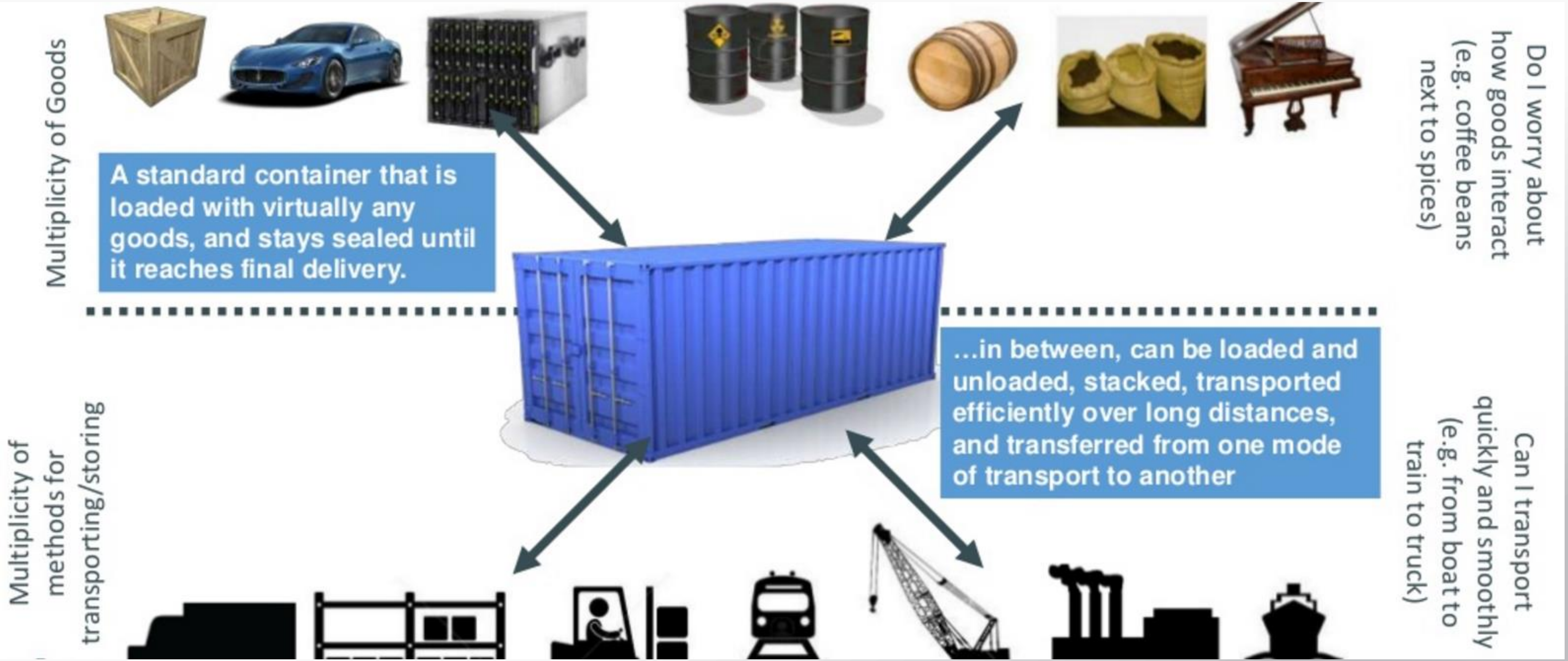
Can I transport quickly and smoothly (e.g. from boat to train to truck)

What are the possibilities



	?	?	?	?	?	?	?
	?	?	?	?	?	?	?
	?	?	?	?	?	?	?
	?	?	?	?	?	?	?
	?	?	?	?	?	?	?
	?	?	?	?	?	?	?

SOLUTION—shipping containers



Multiplicity of Goods

A standard container that is loaded with virtually any goods, and stays sealed until it reaches final delivery.

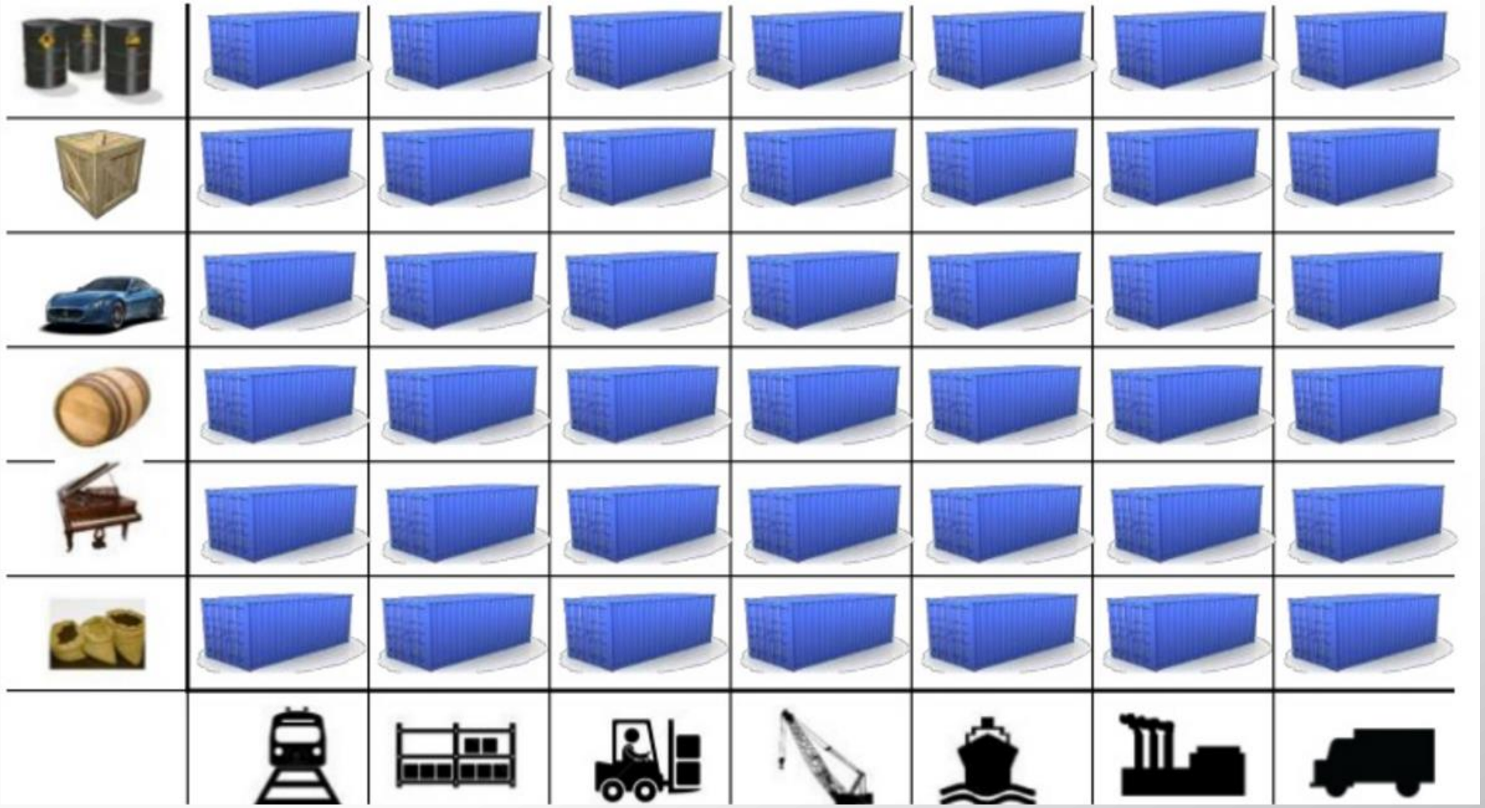
Do I worry about how goods interact (e.g. coffee beans next to spices)

Multiplicity of methods for transporting/storing

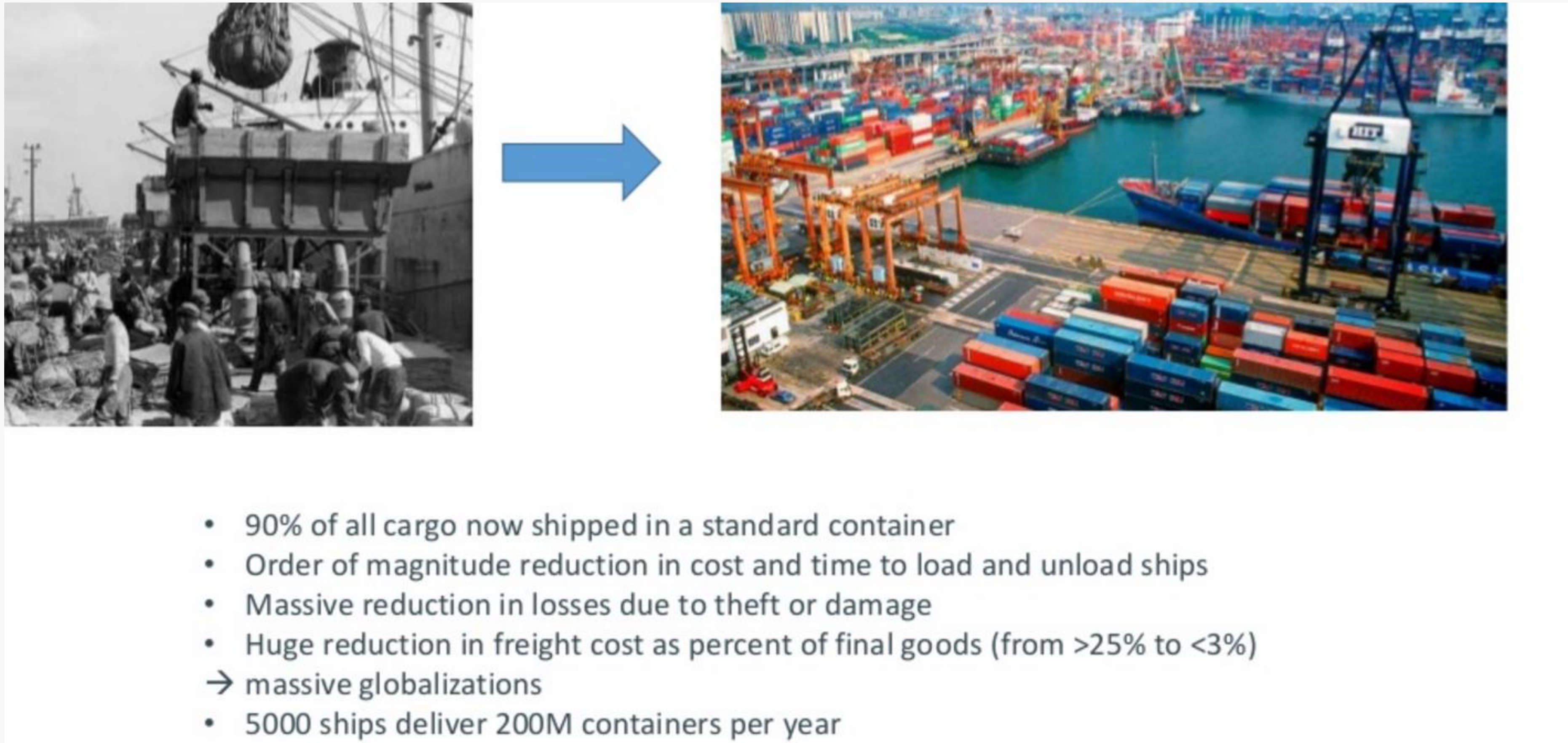
…in between, can be loaded and unloaded, stacked, transported efficiently over long distances, and transferred from one mode of transport to another

Can I transport quickly and smoothly (e.g. from boat to train to truck)

This solved the problem



Today shipping is done with containers

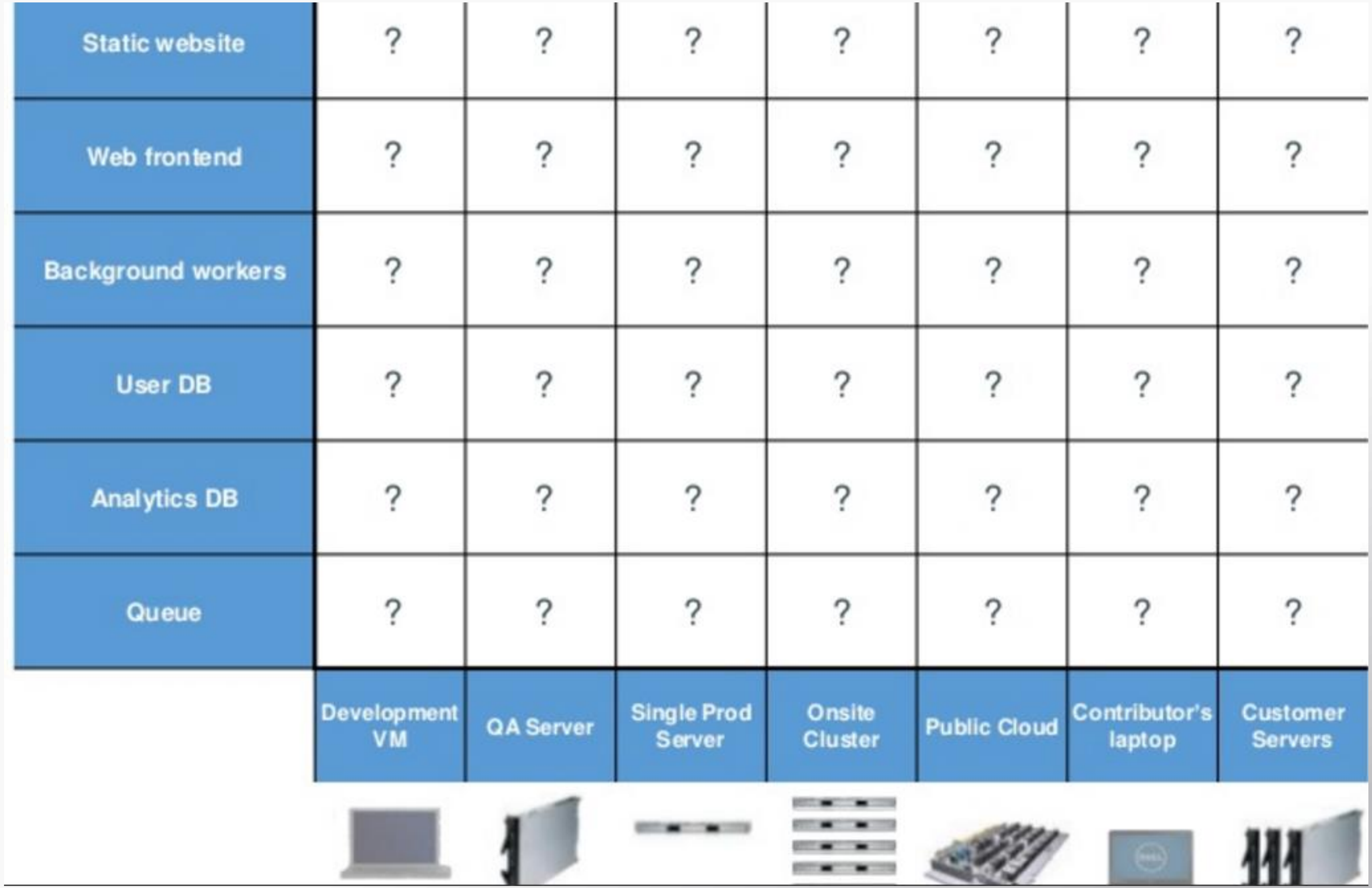


- 90% of all cargo now shipped in a standard container
- Order of magnitude reduction in cost and time to load and unload ships
- Massive reduction in losses due to theft or damage
- Huge reduction in freight cost as percent of final goods (from >25% to <3%)

→ massive globalizations

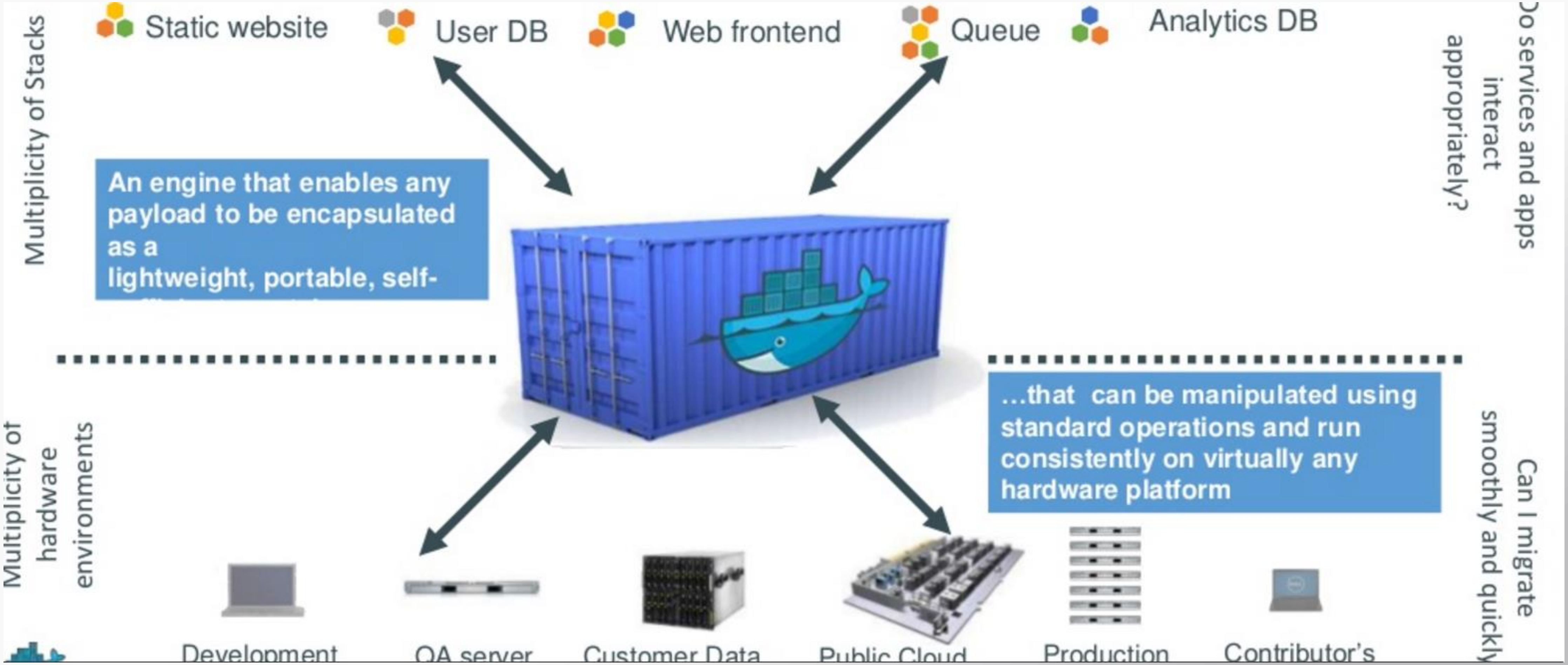
- 5000 ships deliver 200M containers per year

How does this container idea translate to our problem



	Development VM	QA Server	Single Prod Server	Onsite Cluster	Public Cloud	Contributor's laptop	Customer Servers
Static website	?	?	?	?	?	?	?
Web frontend	?	?	?	?	?	?	?
Background workers	?	?	?	?	?	?	?
User DB	?	?	?	?	?	?	?
Analytics DB	?	?	?	?	?	?	?
Queue	?	?	?	?	?	?	?

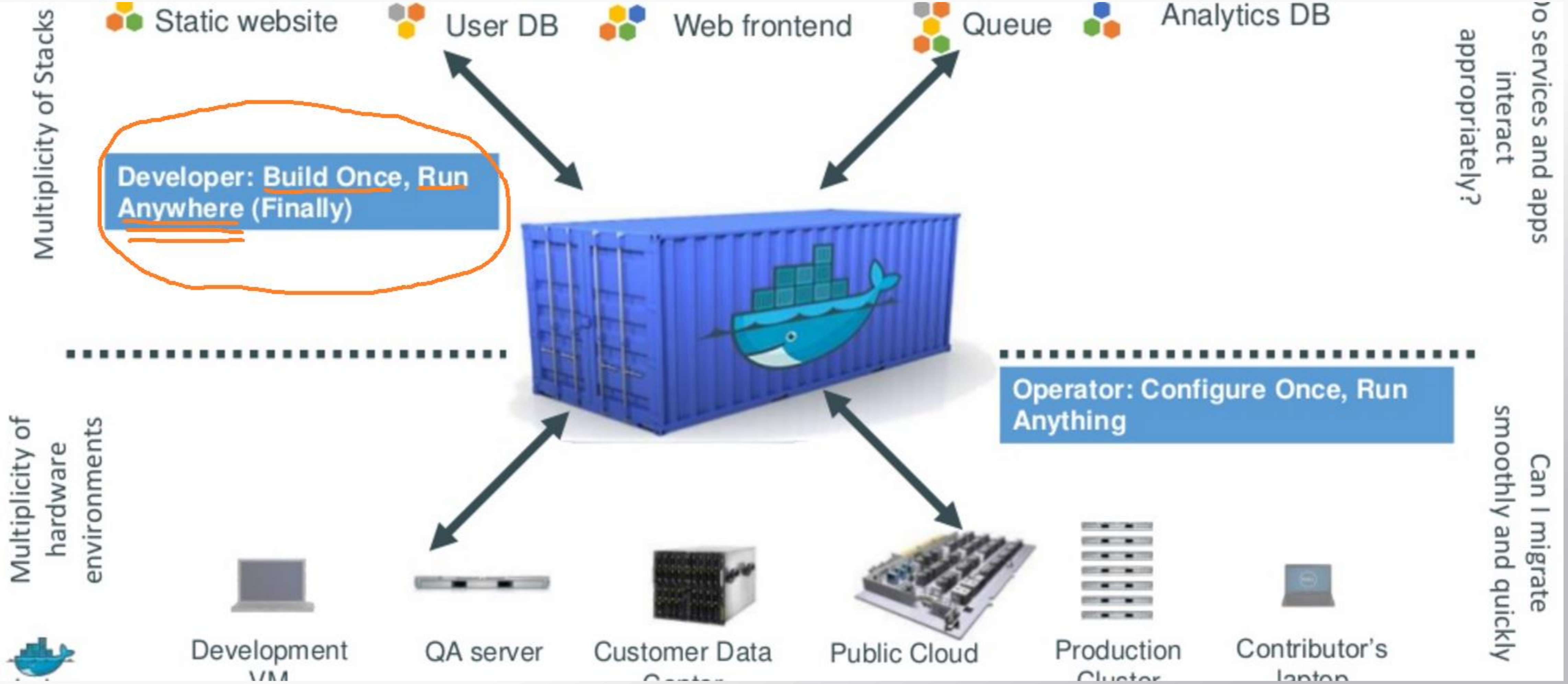
How does this container idea translate to our problem—container for code????



An engine that enables any payload to be encapsulated as a lightweight, portable, self-

…that can be manipulated using standard operations and run consistently on virtually any hardware platform

Do once run anywhere

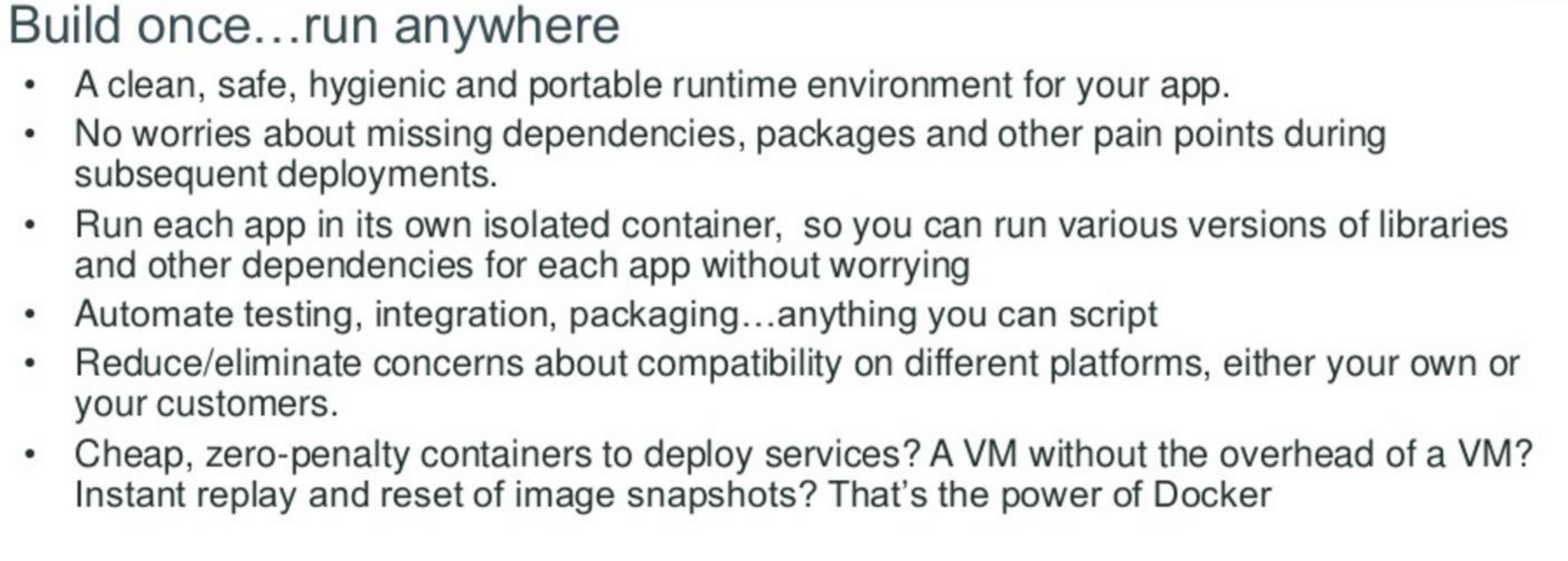

Multiplicity of Stacks
Static website
User DB
Web frontend
Queue
Analytics DB
Do services and apps interact appropriately?
Developer: Build Once, Run Anywhere (Finally)
Operator: Configure Once, Run Anything
Multiplicity of hardware environments
Can I migrate smoothly and quickly
Development VM
QA server
Customer Data Center
Public Cloud
Production Cluster
Contributor's laptop

Docker supported in many Cloud platforms

amazon web services™
DigitalOcean
docker
Google
Microsoft

Docker container—developer viewpoint

Build once … run anywhere



Build once…run anywhere

- A clean, safe, hygienic and portable runtime environment for your app.
- No worries about missing dependencies, packages and other pain points during subsequent deployments.
- Run each app in its own isolated container, so you can run various versions of libraries and other dependencies for each app without worrying
- Automate testing, integration, packaging…anything you can script
- Reduce/eliminate concerns about compatibility on different platforms, either your own or your customers.
- Cheap, zero-penalty containers to deploy services? A VM without the overhead of a VM? Instant replay and reset of image snapshots? That's the power of Docker