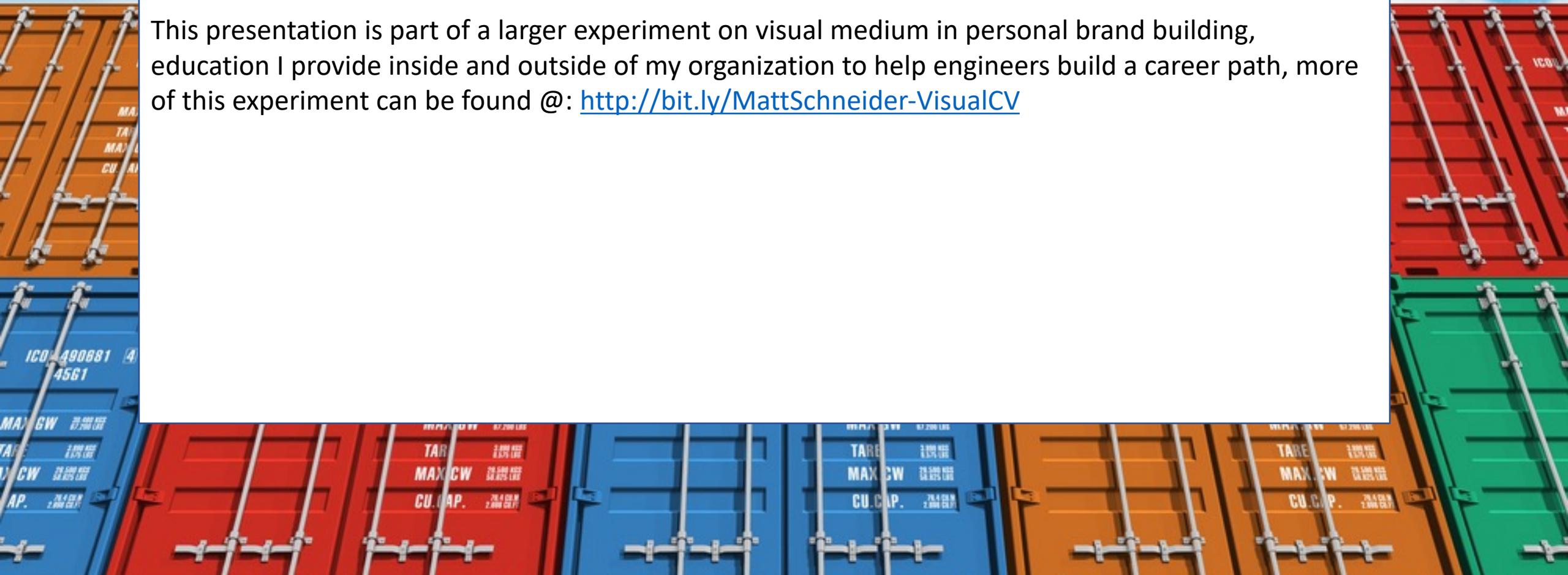


Explaining complex technology problems in a simplistic way is a skill I've been working to hone. Marketing approached me to explain to all of their organization "containers" in a simplistic and fun way, as part of a series they called "The Stark Hour", after Tony Stark.

I enjoyed making this deck so much, and the feedback was so positive, it's been used in countless other customer conversations as well as new hire, college graduate programs and more.

This presentation is part of a larger experiment on visual medium in personal brand building, education I provide inside and outside of my organization to help engineers build a career path, more of this experiment can be found @: <http://bit.ly/MattSchneider-VisualCV>





CONTAINERS

CONTAINERS





CONTAINERS



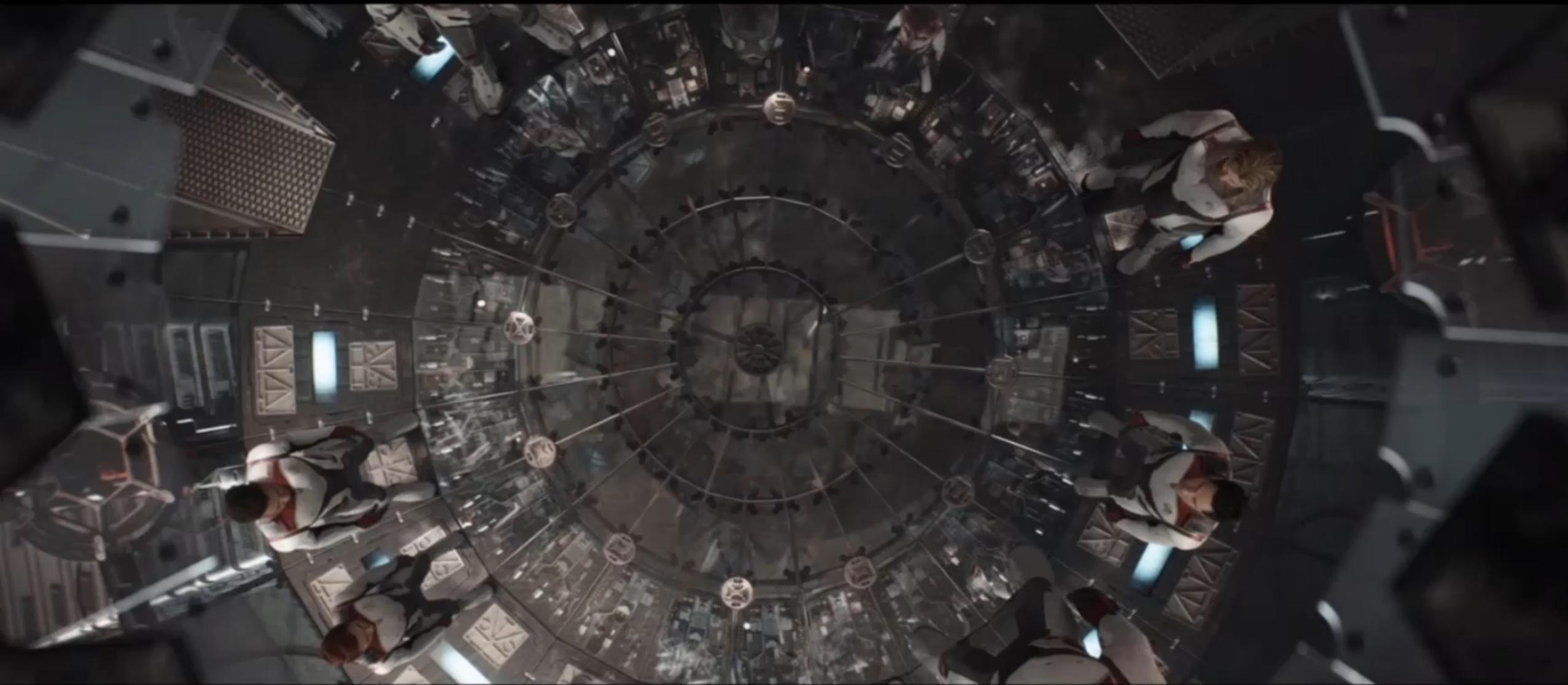


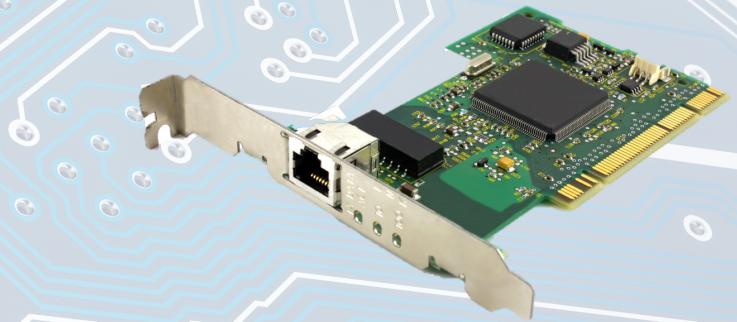
Matt Schneider
Field CTO & Principal Engineer
Enterprise Americas Presales

 @md_schneider
 /in/mdschneider

★★
CTO Ambassadors
• 2020 •







Hardware



Processor

CPU
GPU
FPGA
ASIC



Storage

PMEM
RAM
NVME/SSD
Hard Drive



Network

Ethernet
Fibre Channel
Infiniband
RDMA

Operating System



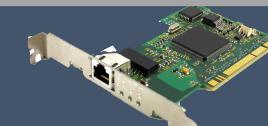
Hardware



Processor



Storage



Network

CPU
GPU
FPGA
ASIC

PMEM
RAM
NVME/SSD
Hard Drive

Ethernet
Fibre Channel
Infiniband
RDMA

Platforms



Operating System



Hardware



Processor



Storage

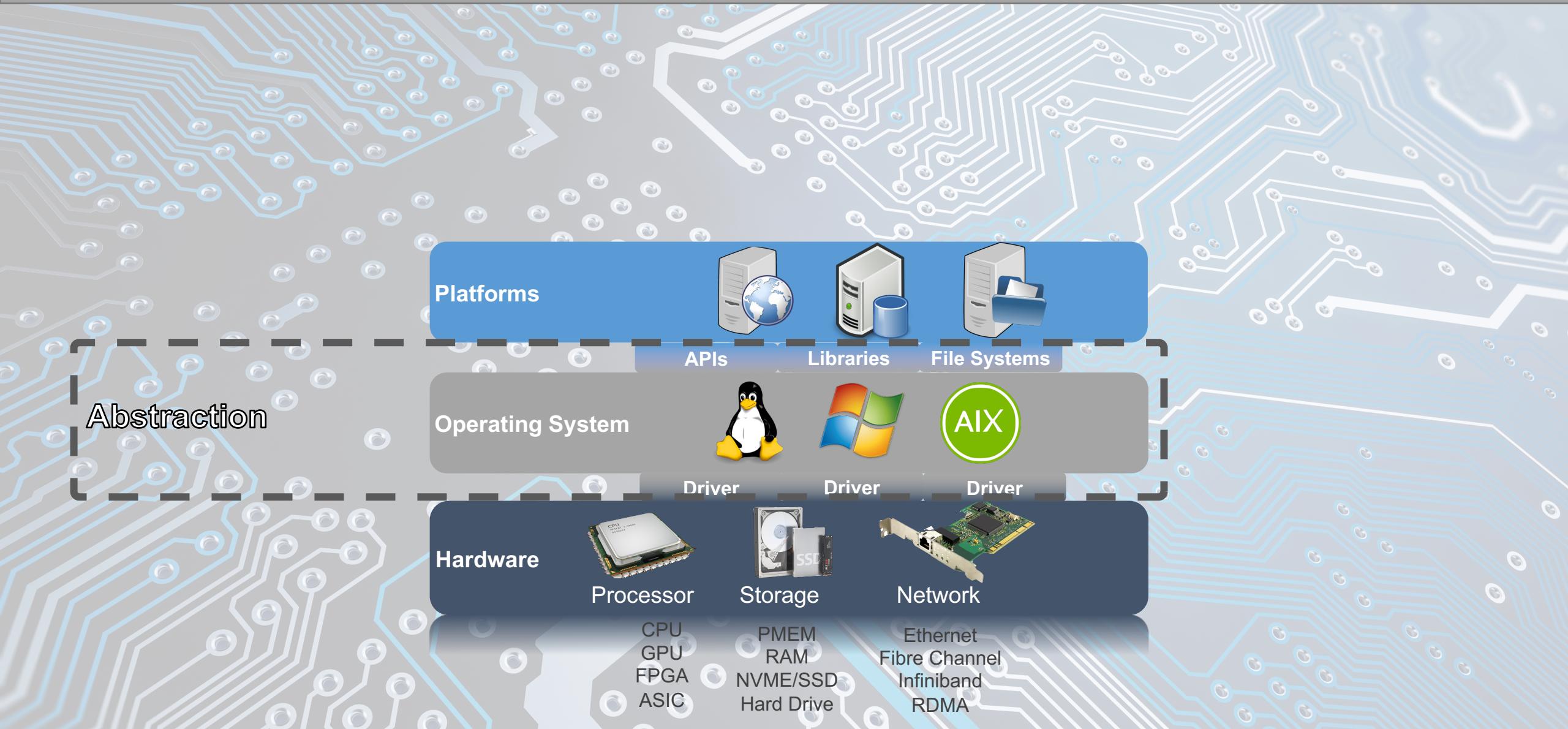


Network

CPU
GPU
FPGA
ASIC

PMEM
RAM
NVME/SSD
Hard Drive

Ethernet
Fibre Channel
Infiniband
RDMA





Platforms

APIs

Libraries

File Systems

Operating System

Driver

Driver

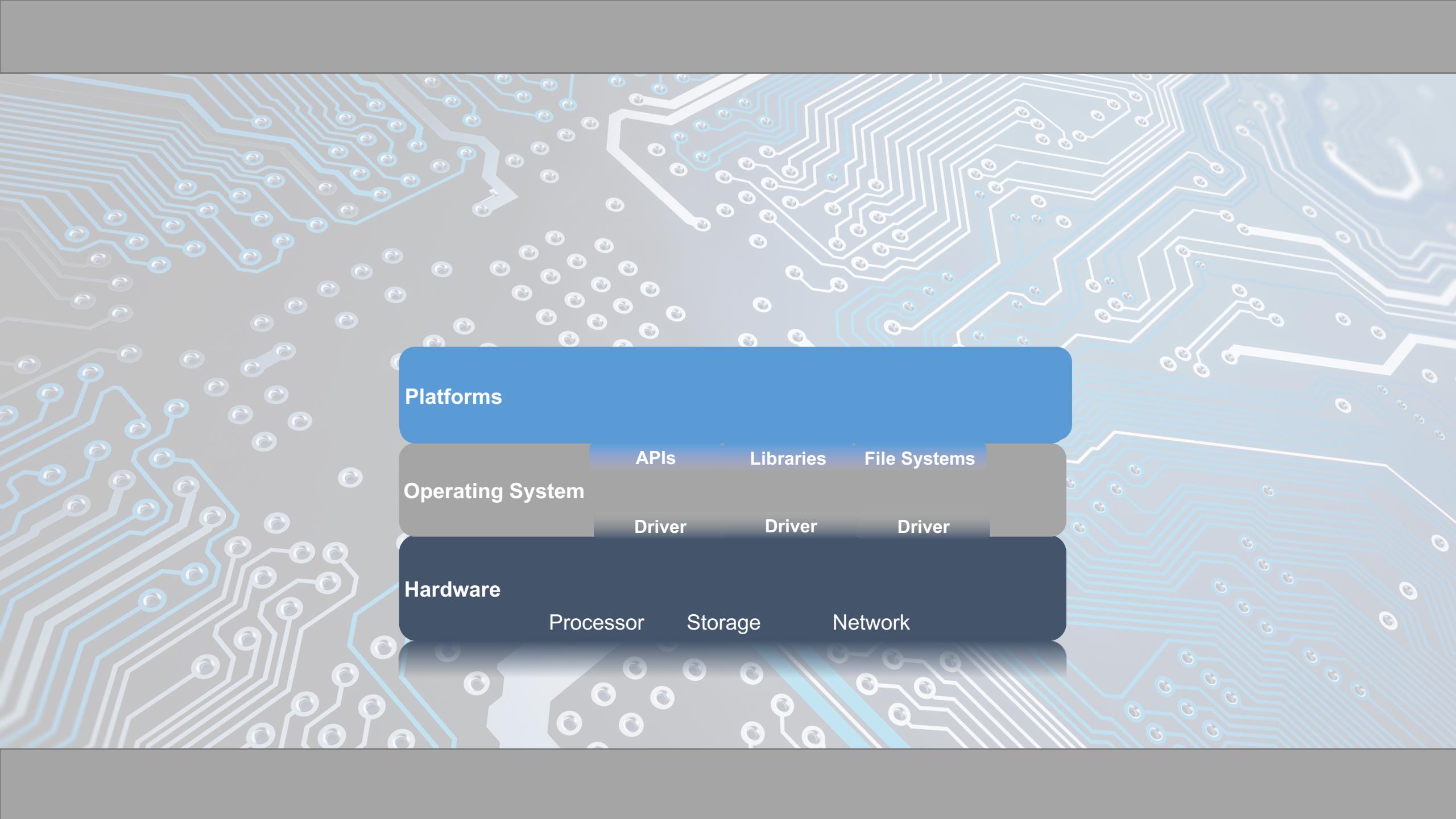
Driver

Hardware

Processor

Storage

Network



Line of Business Application

Web Server

Database

APIs

Libraries

File Systems

Operating System

Driver

Driver

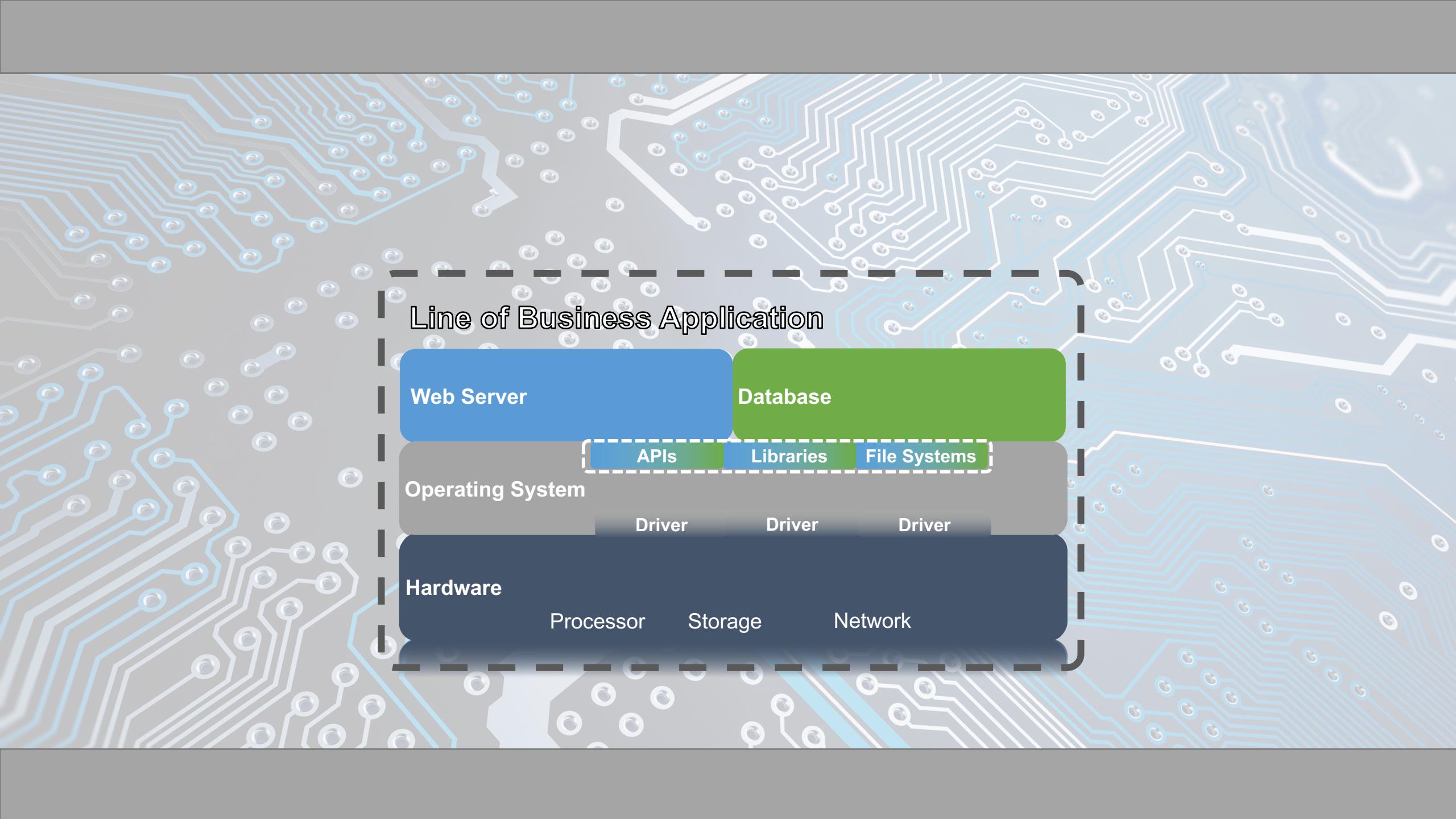
Driver

Hardware

Processor

Storage

Network



Line of Business Application

Web Server

Database

APIs

Libraries

File Systems

Operating System

Driver

Driver

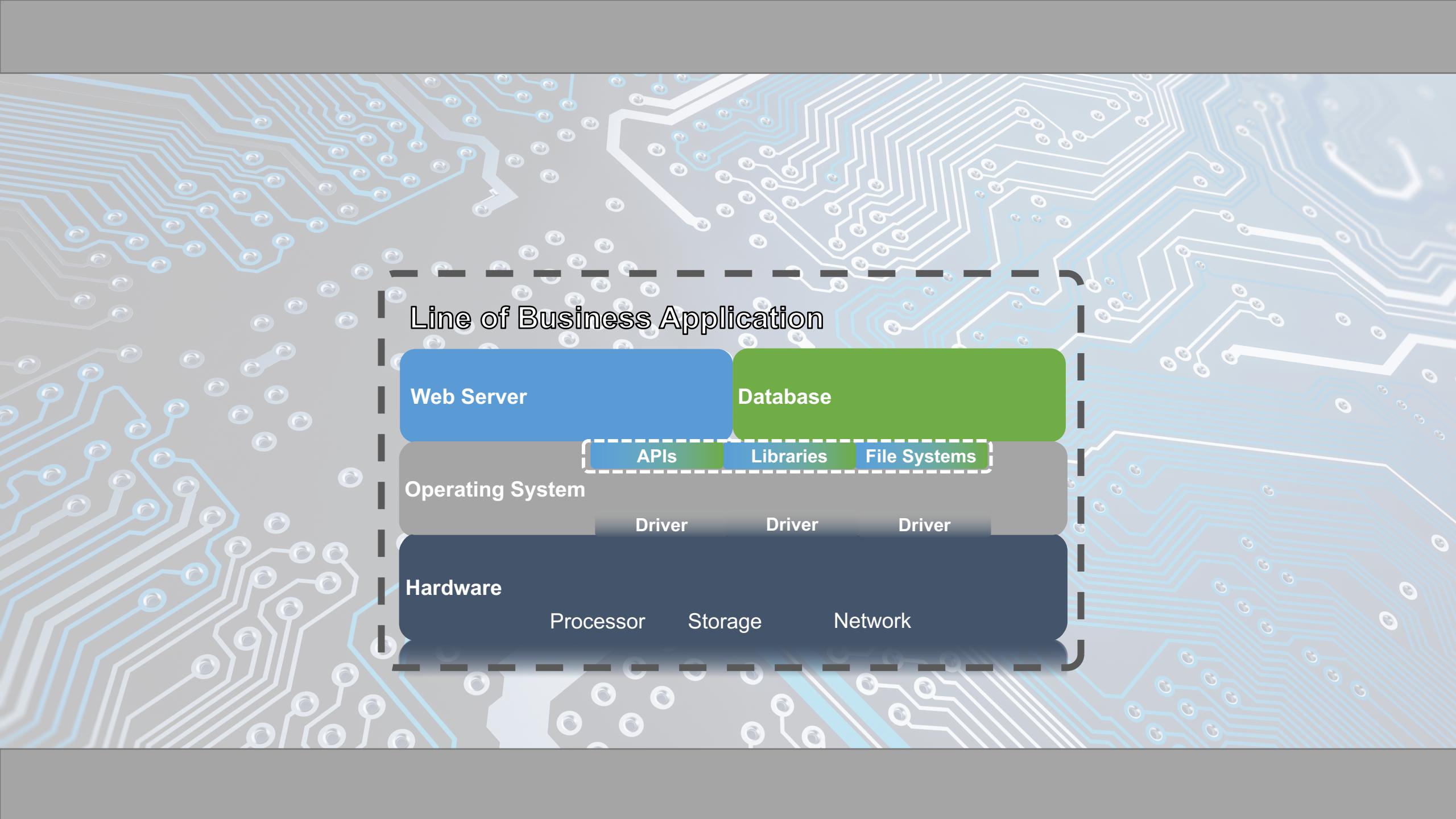
Driver

Hardware

Processor

Storage

Network



Line of Business Application

Web Server

Database

Operating System

APIs

Libraries

File Systems

Hardware

Driver

Driver

Driver

Processor

Storage

Network

Line of Business Application

Web Server

APIs

Libraries

File Systems

Operating System

Driver

Driver

Driver

Hardware

Processor

Storage

Network

Database

APIs

Libraries

File Systems

Operating System

Driver

Driver

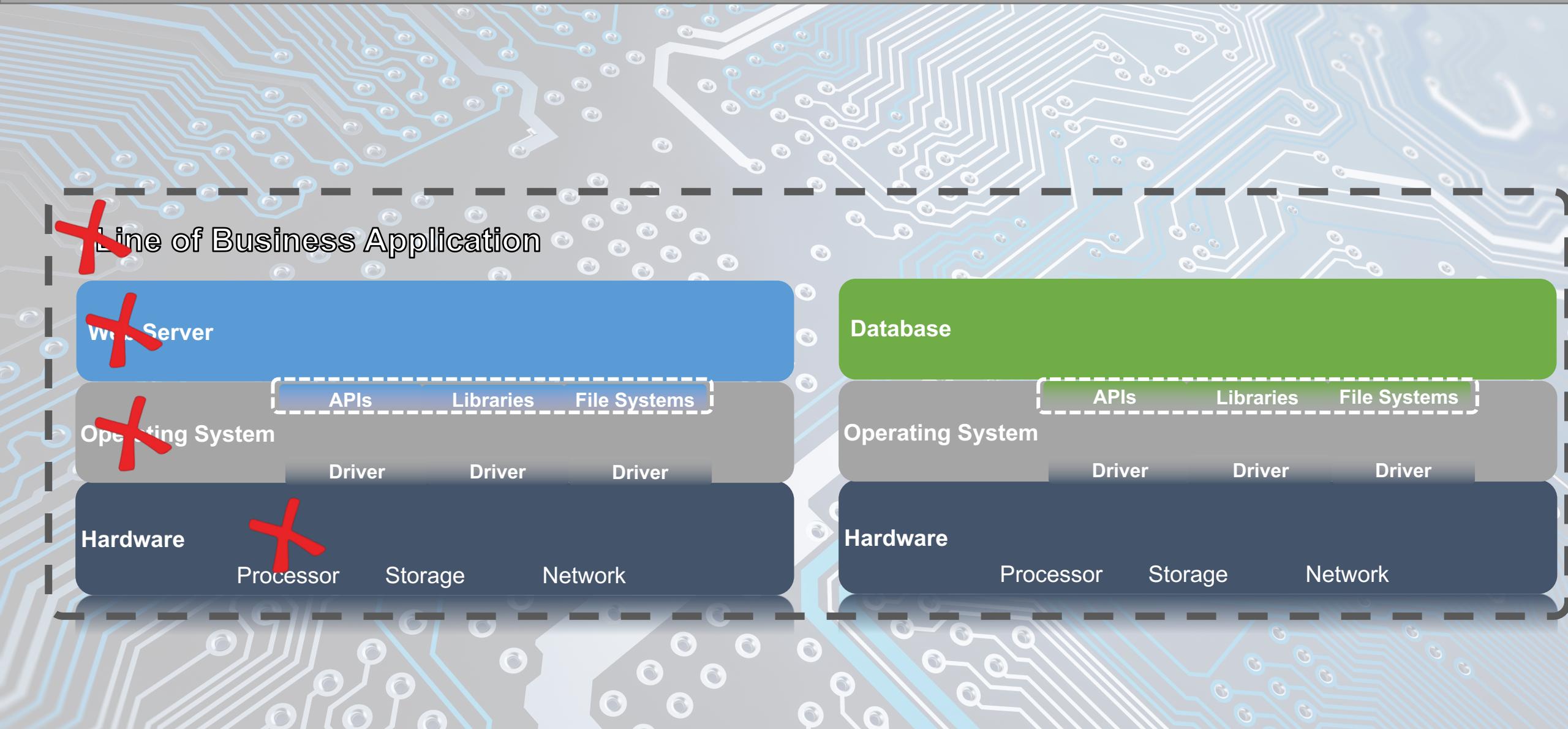
Driver

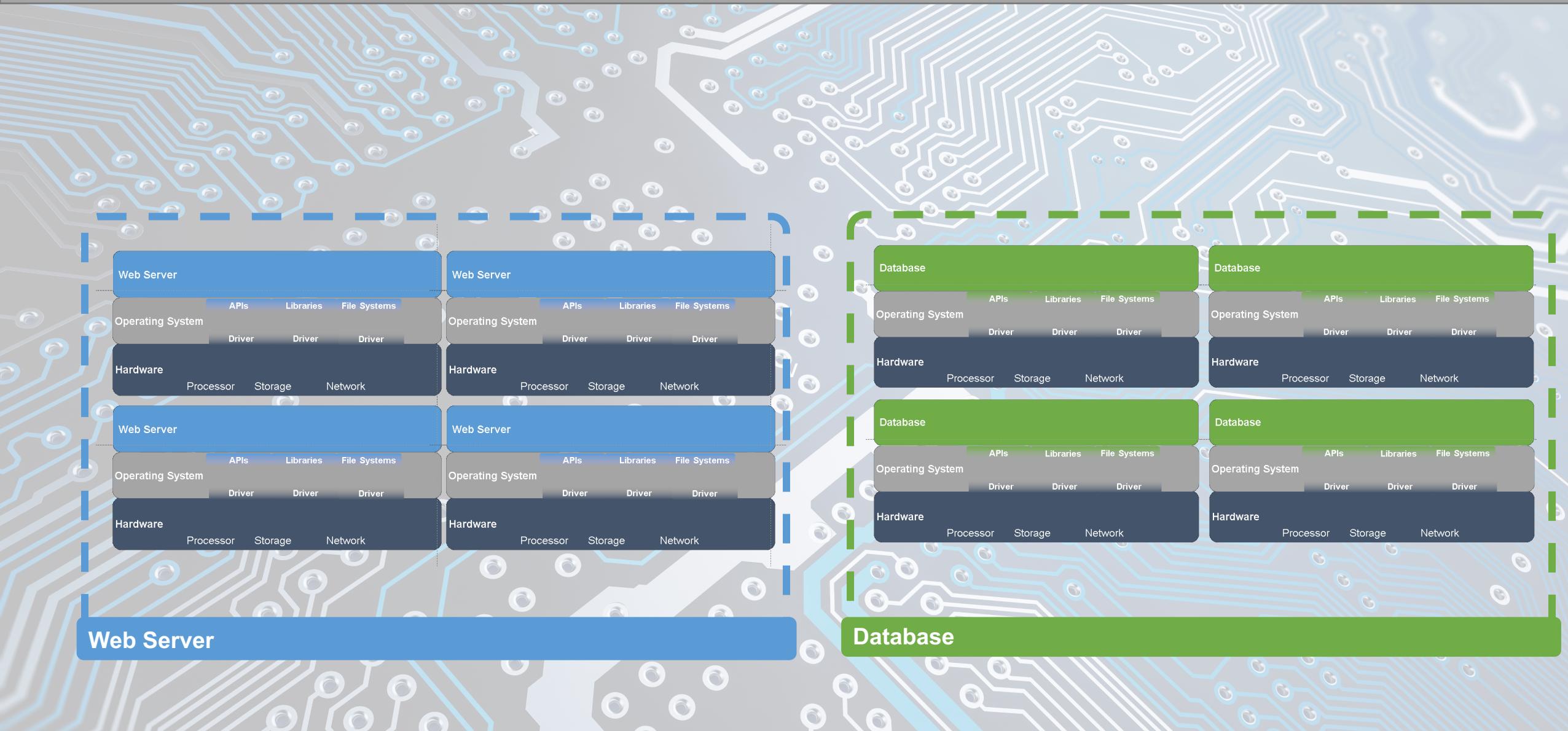
Hardware

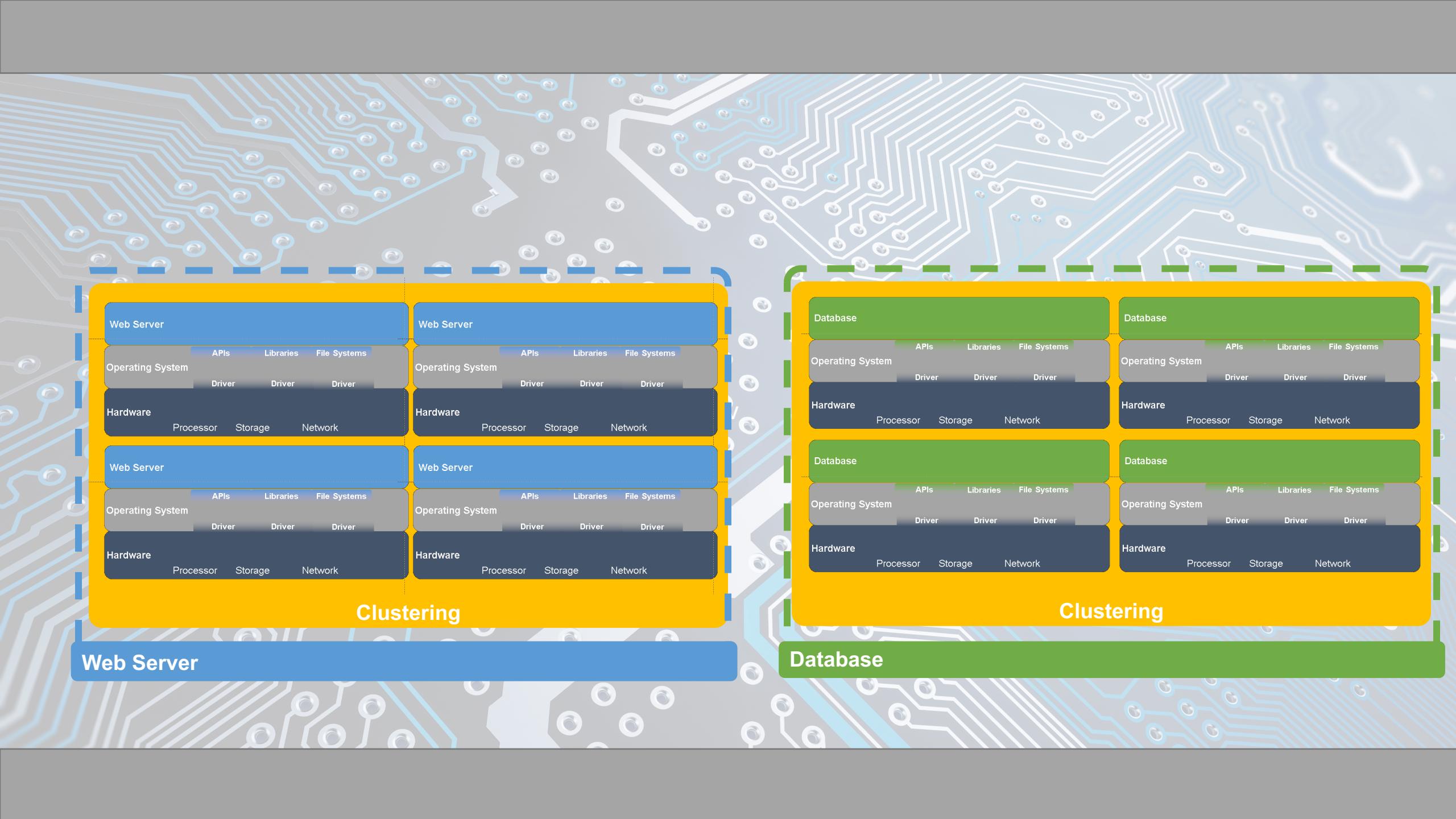
Processor

Storage

Network



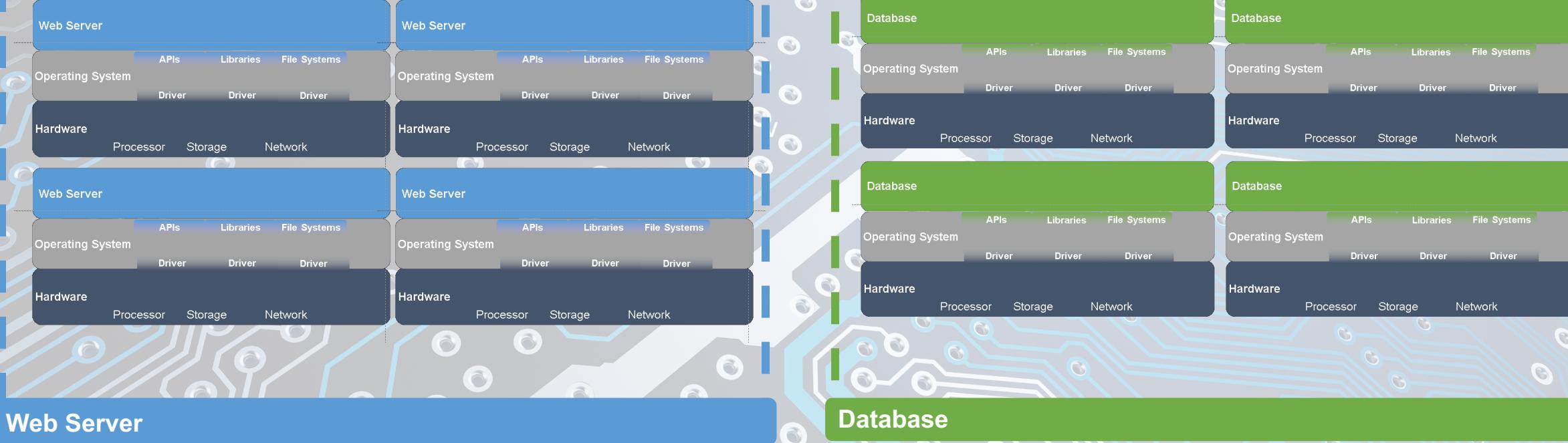




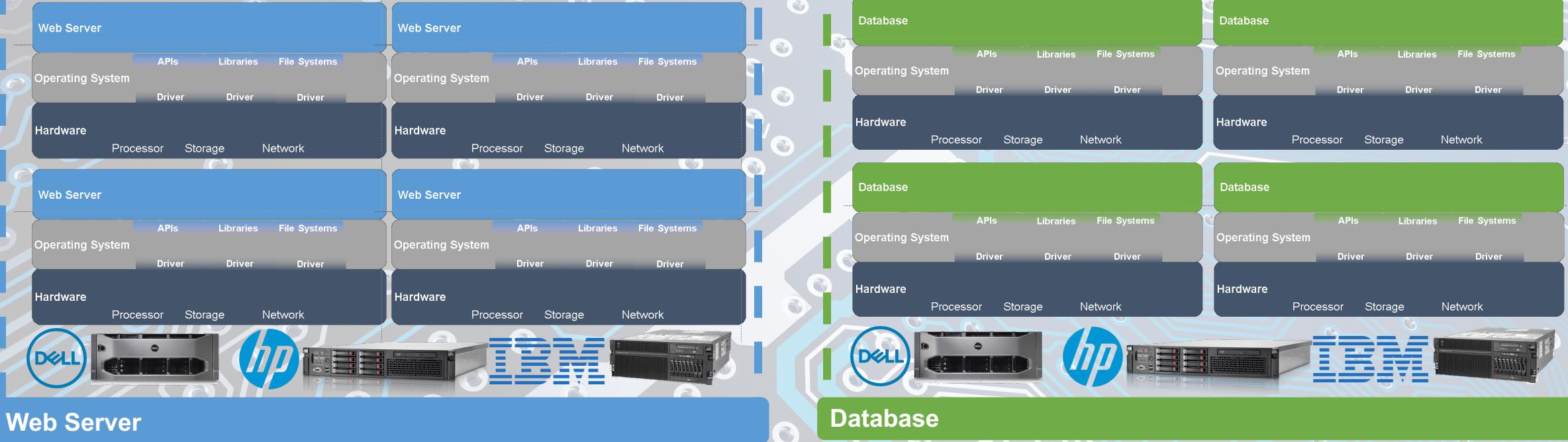


LOB
APP

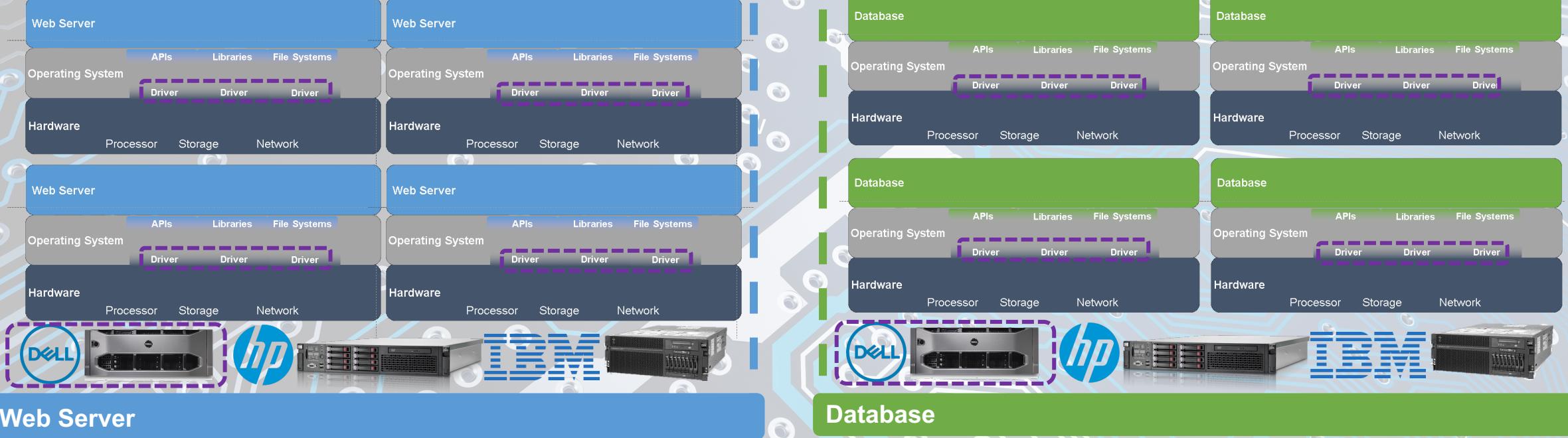
LOB APP



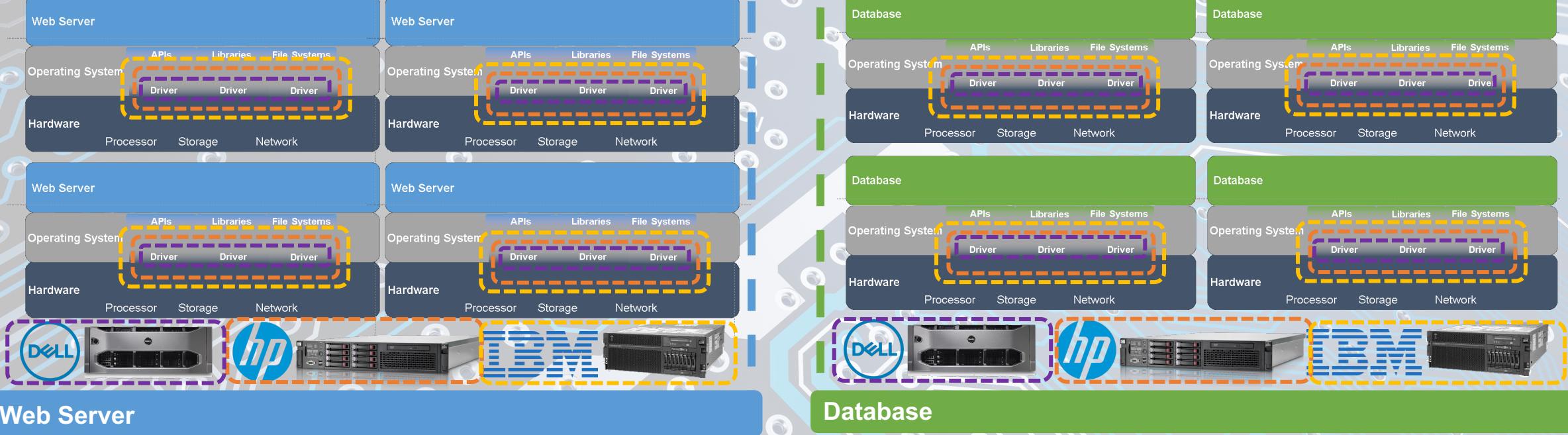
LOB APP

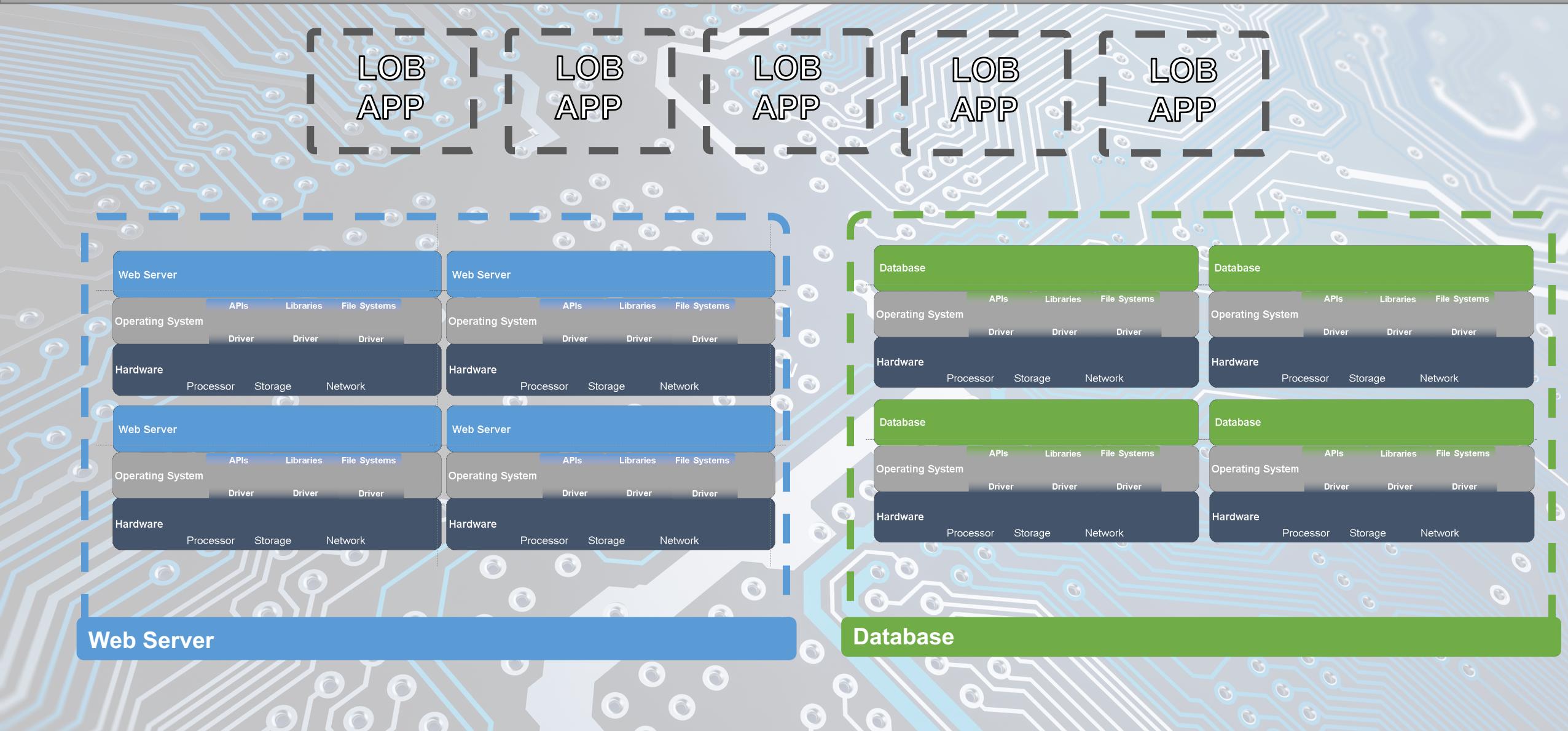


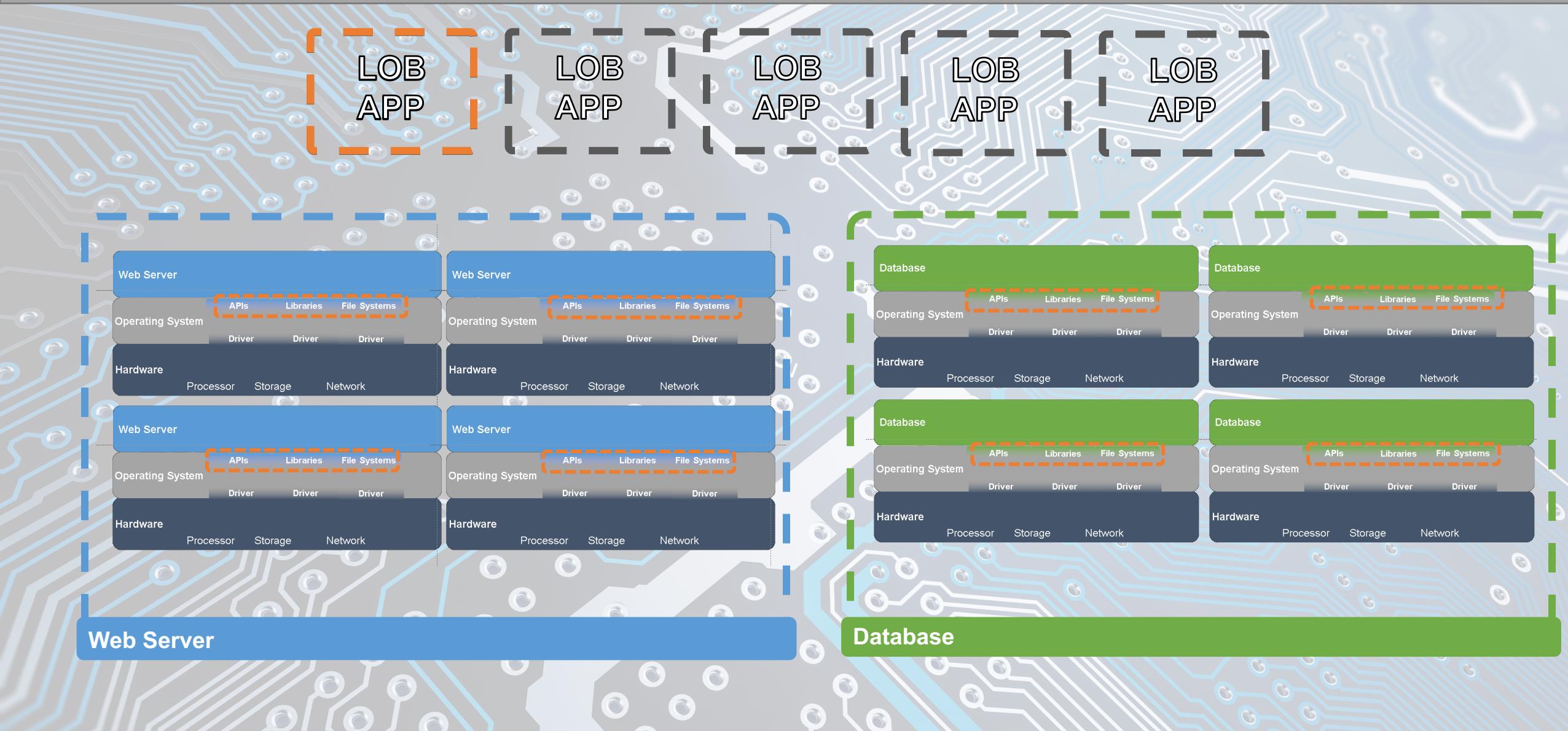
LOB APP

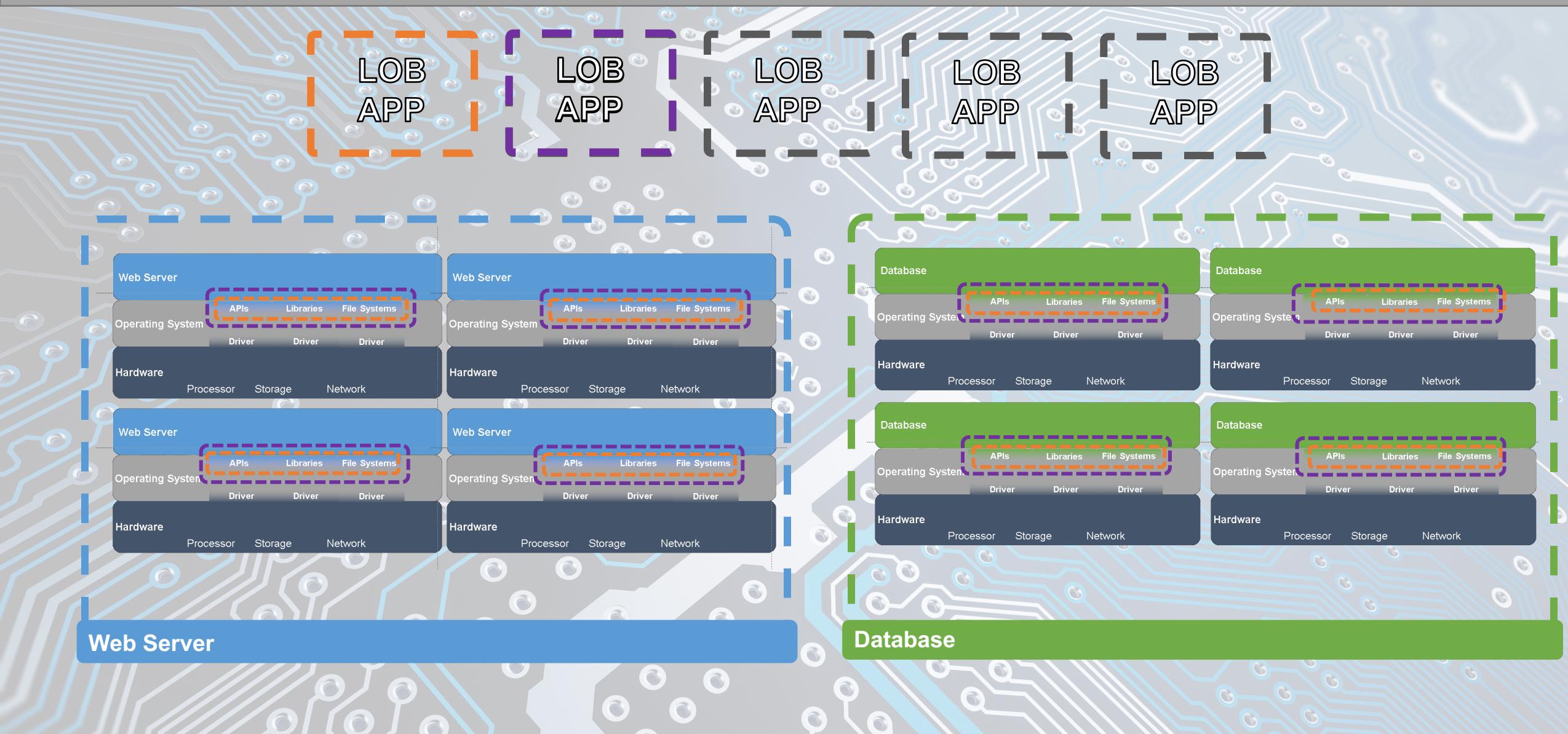


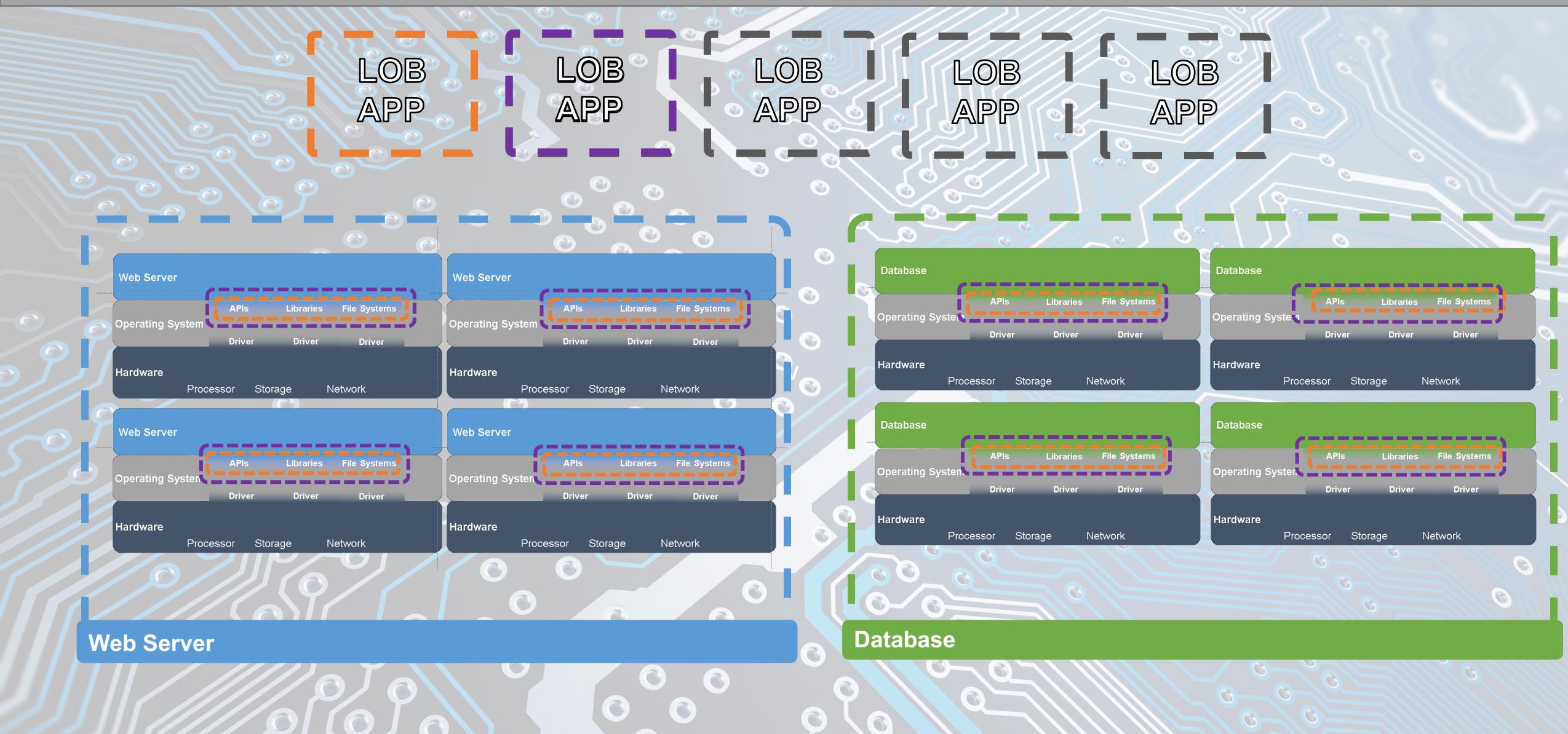
LOB APP

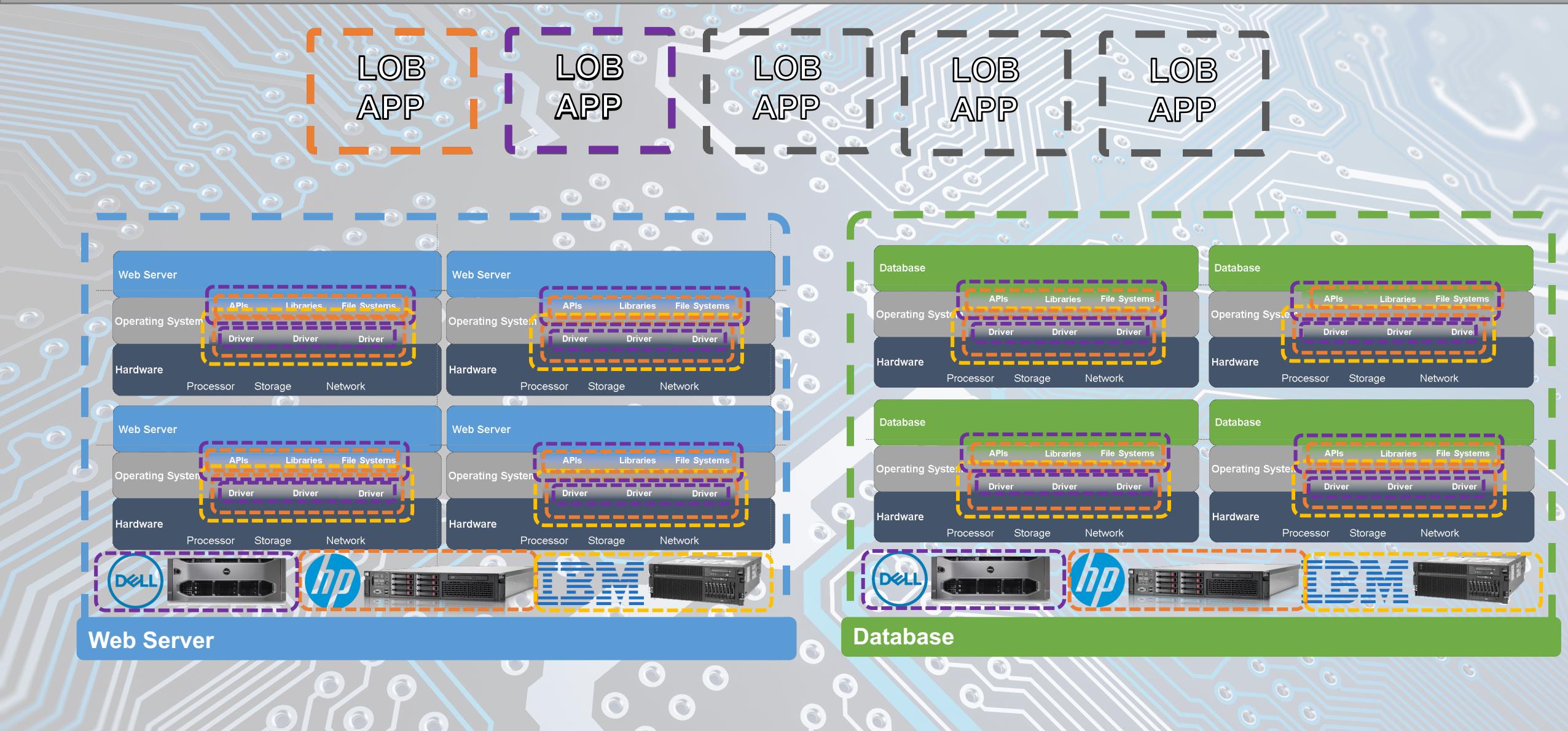














Web Server



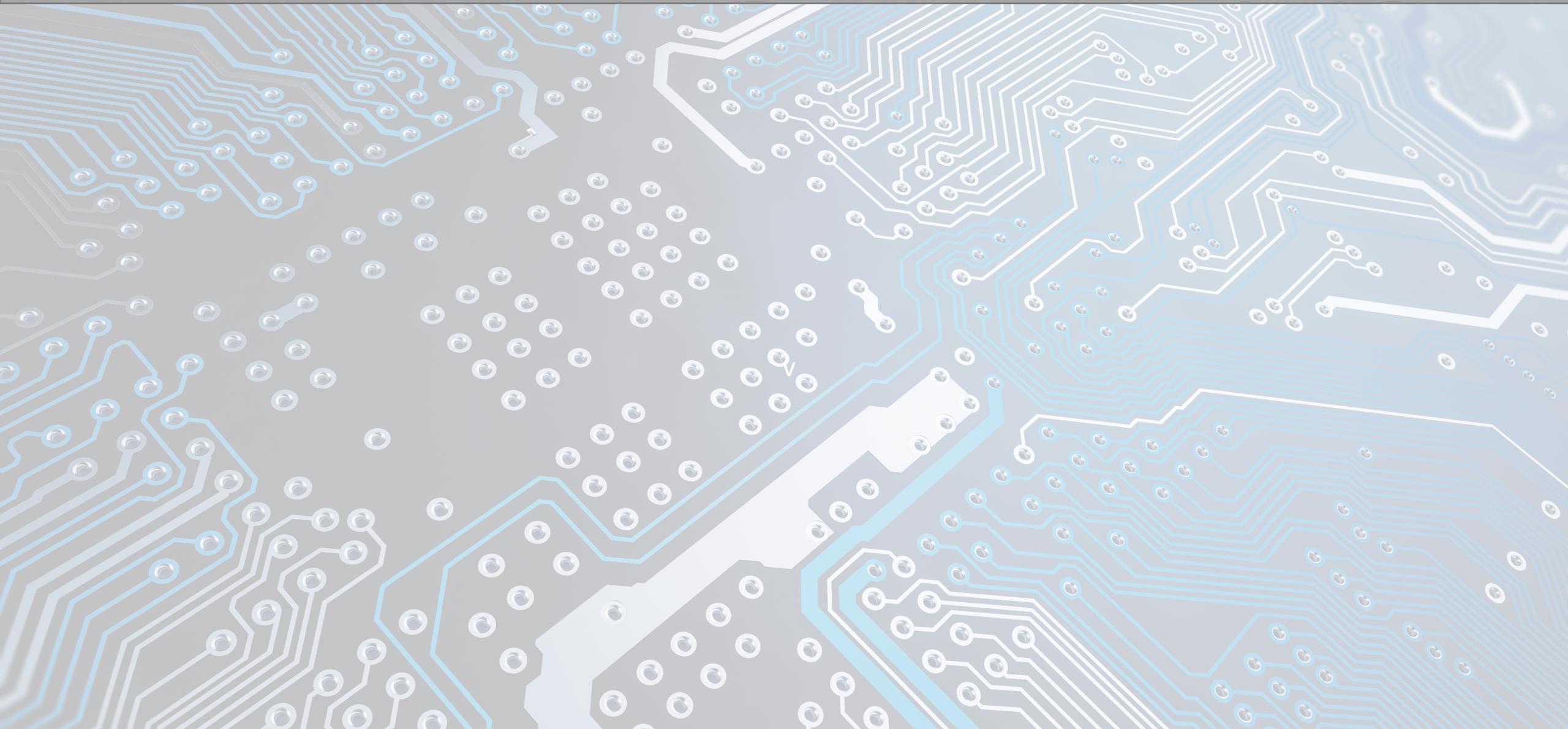
hardware



rage Network



A white server rack unit with multiple drive bays and a power supply unit.



Operating System



Hardware



Processor



Storage



Network



Operating System

Driver

Driver

Driver

Virtualization Layer

Driver

Driver

Driver

Hardware



Operating System

Driver

Driver

Driver

Virtualization Layer

Driver

Driver

Driver

Hardware



Operating System

Driver

Driver

Driver

Virtualization Layer

Driver

Driver

Driver

Hardware

IBM





Operating System

Driver

Driver

Driver

Hypervisor

Driver

Driver

Driver

Hardware

Web Server

Database

Operating System

Hypervisor

Hardware



Web Server

Database

Operating System

Operating System

Hypervisor

Hardware



Line of Business Application

Web Server

Database

Operating System



Operating System



Hypervisor

Hardware

Line of Business Application

Web Server

Database

APIs Libraries File Systems

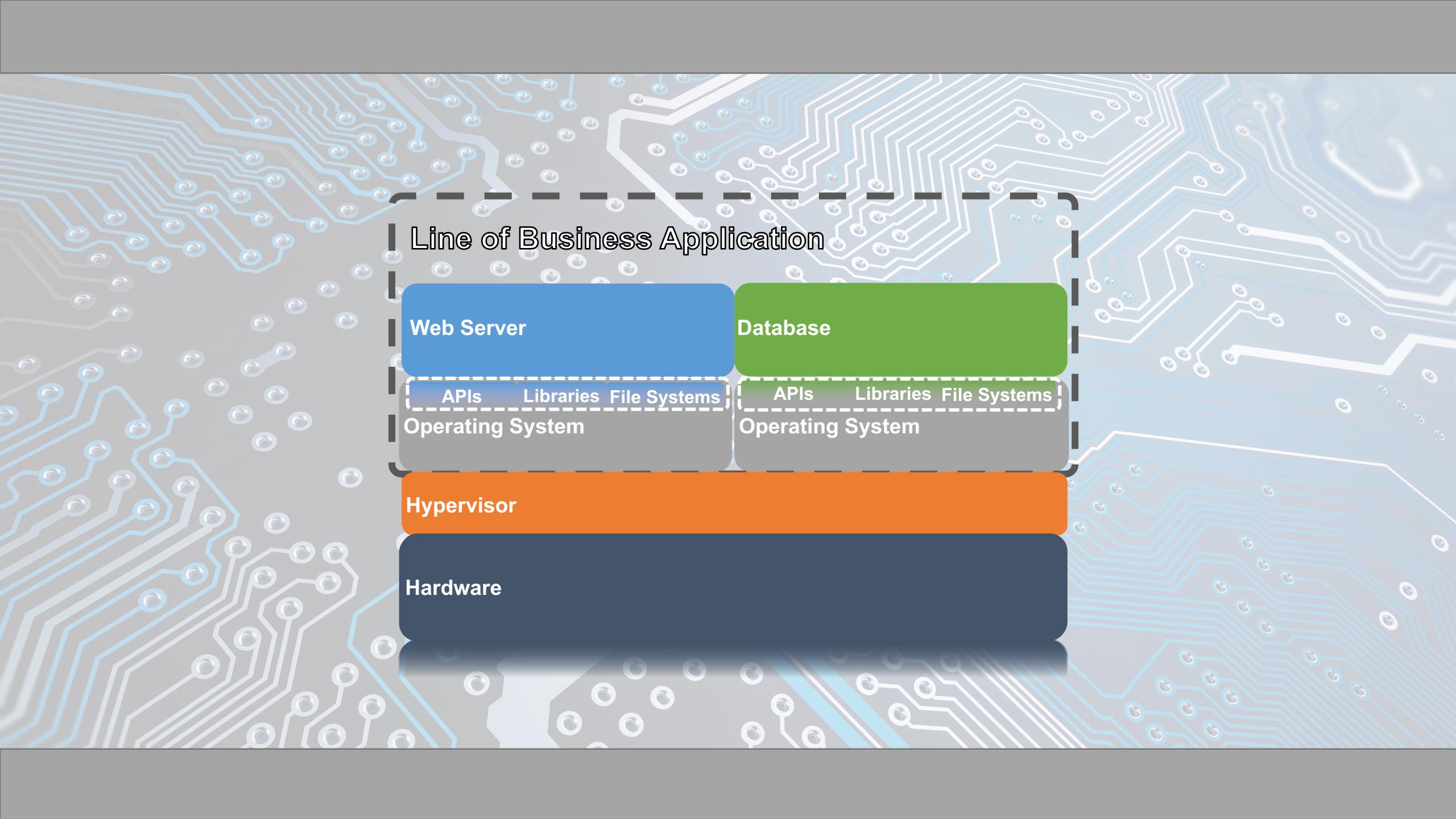
Operating System

APIs Libraries File Systems

Operating System

Hypervisor

Hardware



Line of Business Application

Web Server

Database

Operating System

Operating System

Hypervisor

Hypervisor

Hardware

Hardware

Application Development

Web Server

Database

Operating System

Operating System

Hypervisor

Hypervisor

Hardware

Hardware

Infrastructure Operations

Application Development

Web Server

Database

Operating System

Operating System

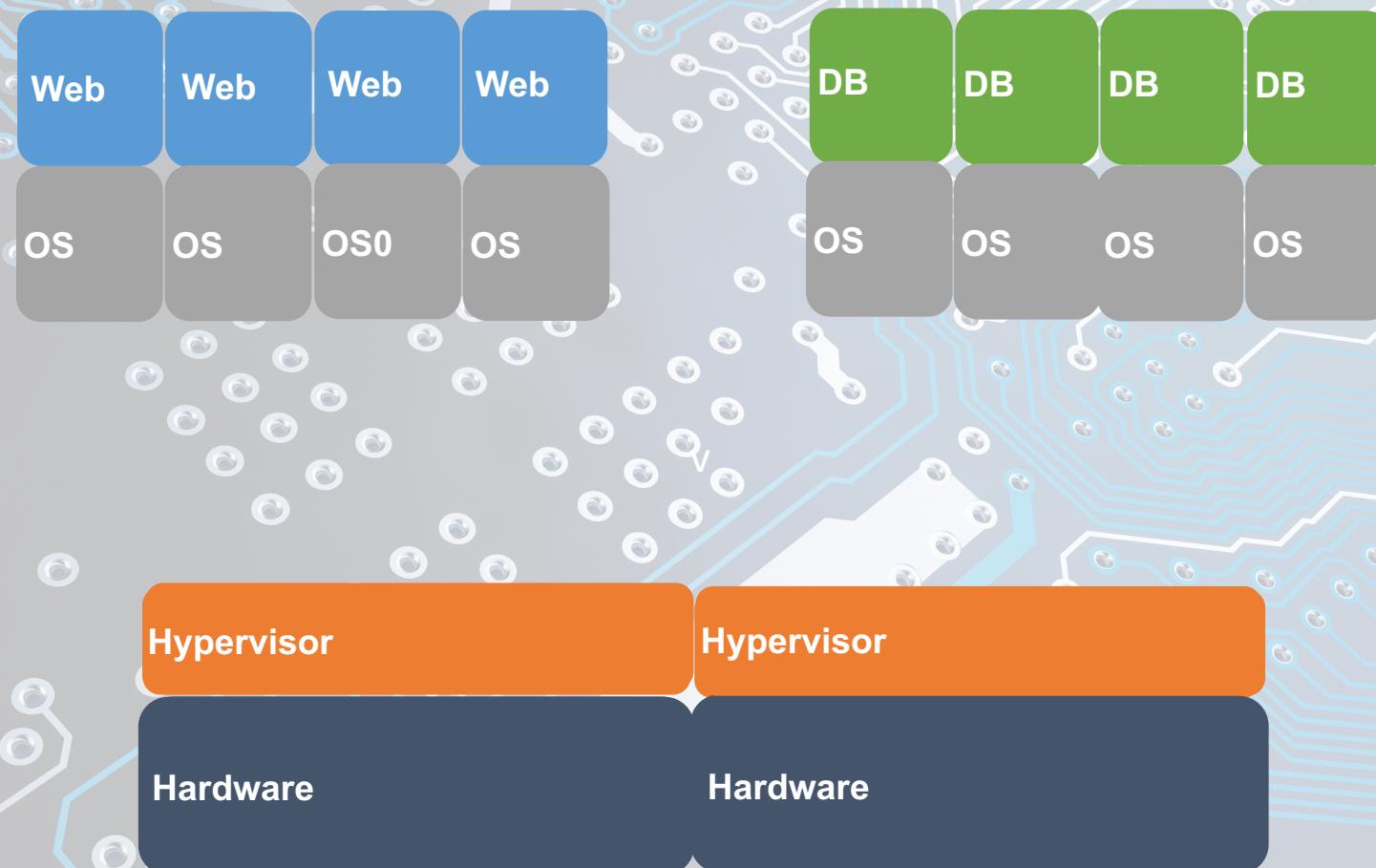
Hypervisor

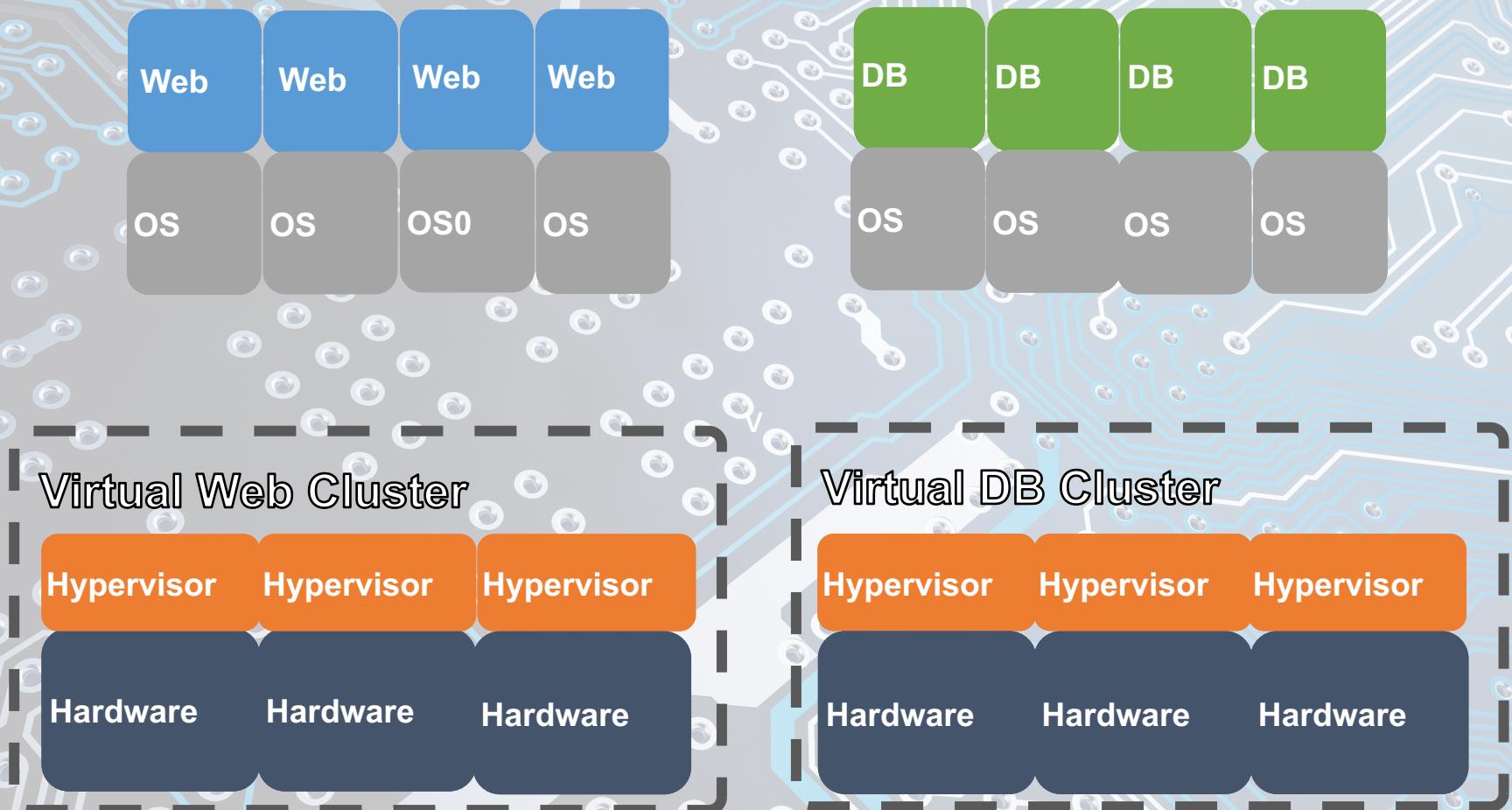
Hypervisor

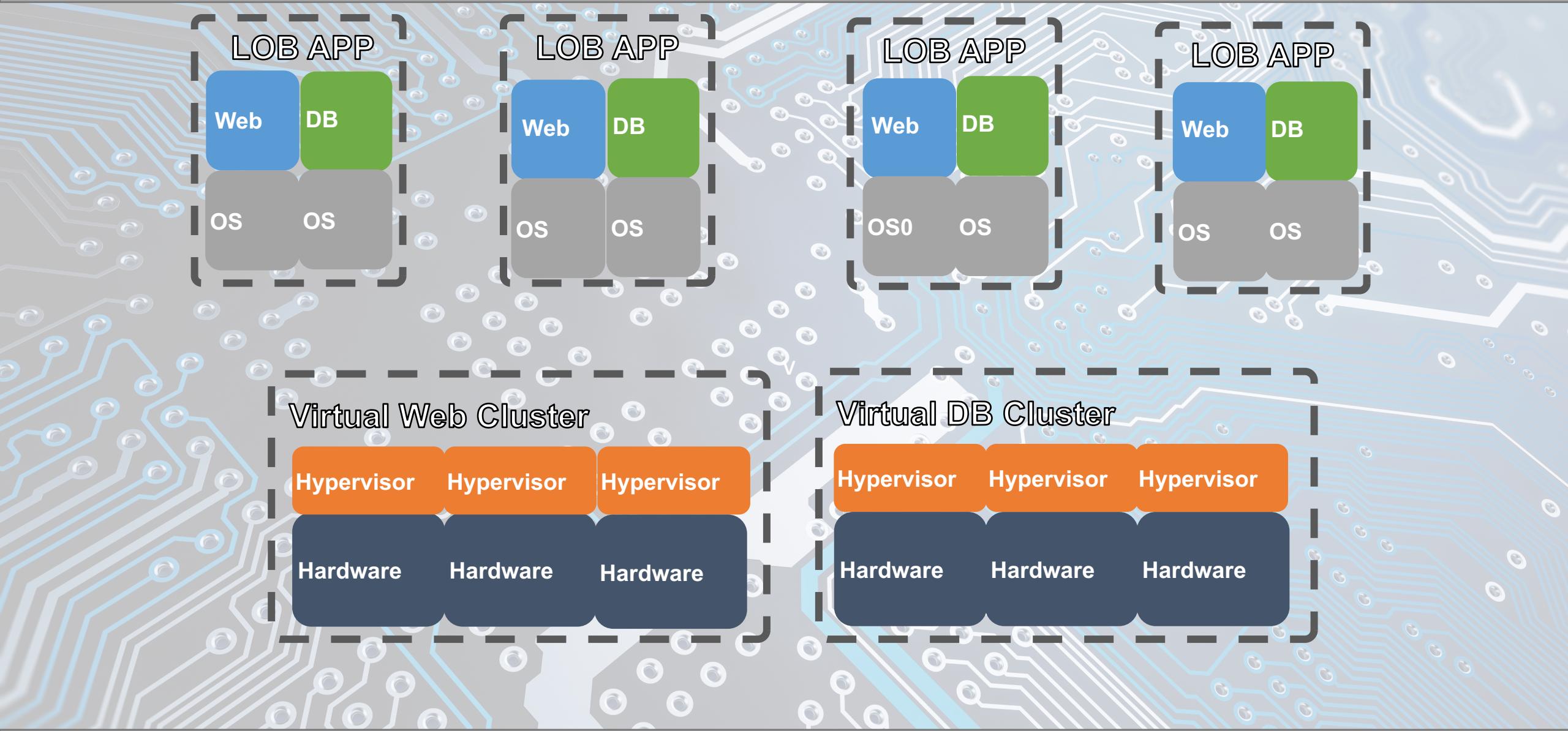
Hardware

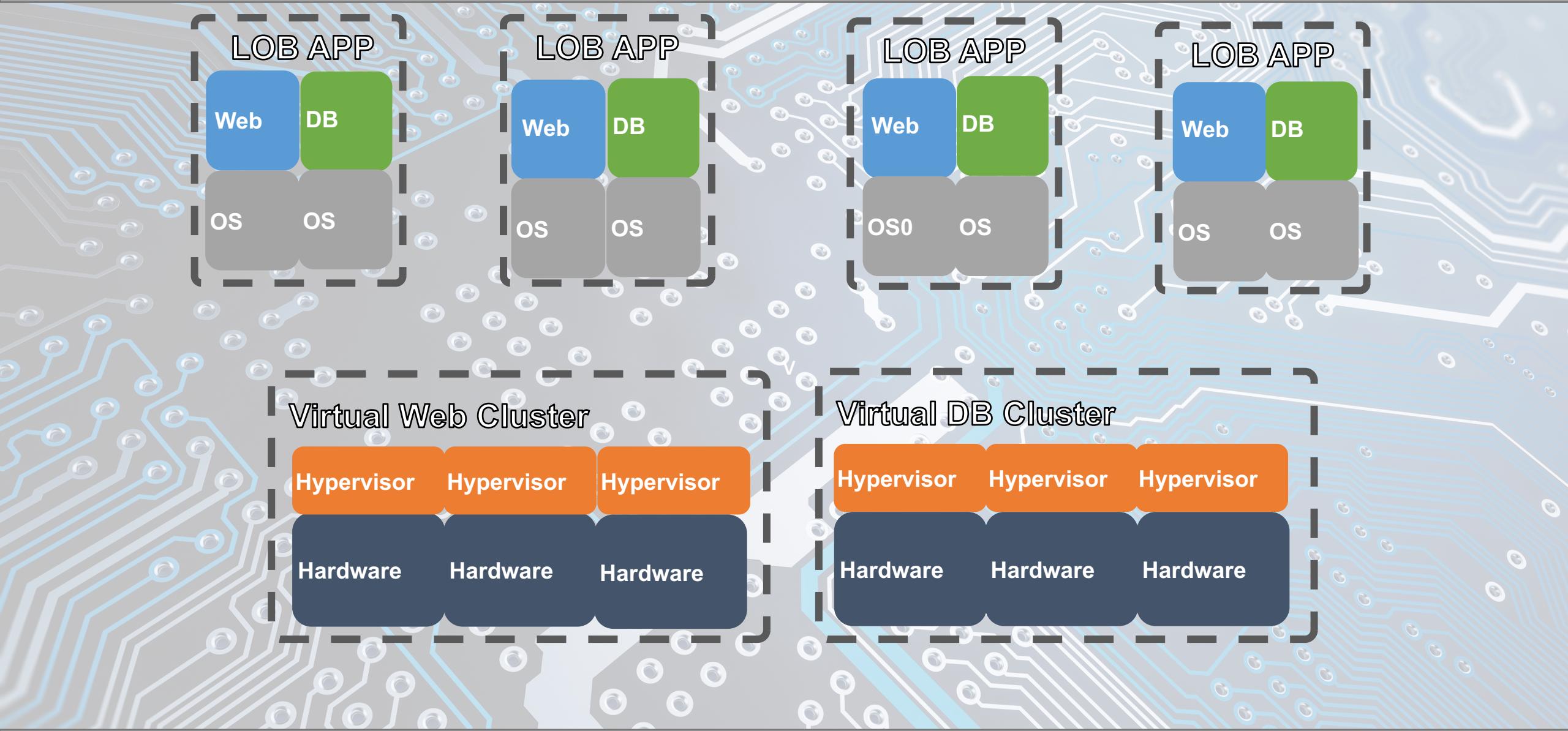
Hardware

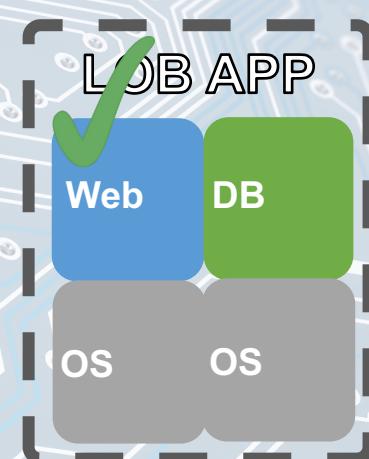
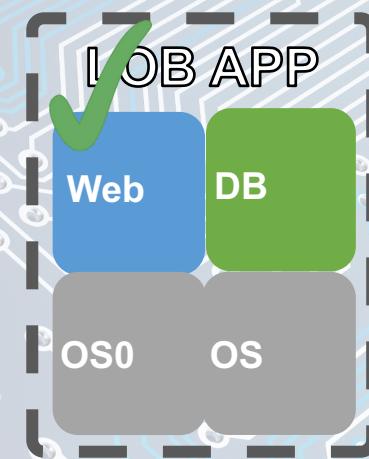
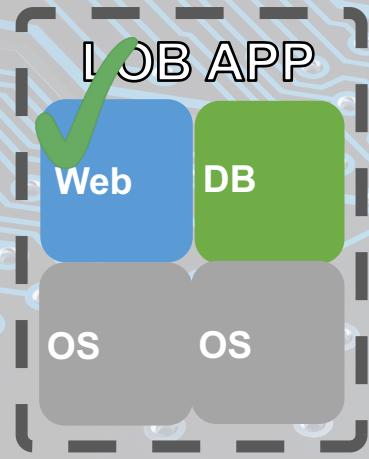
Infrastructure Operations











Virtual Web Cluster



Virtual DB Cluster



Platforms

Operating System

APIs

Libraries

File Systems

Hypervisor

Driver

Driver

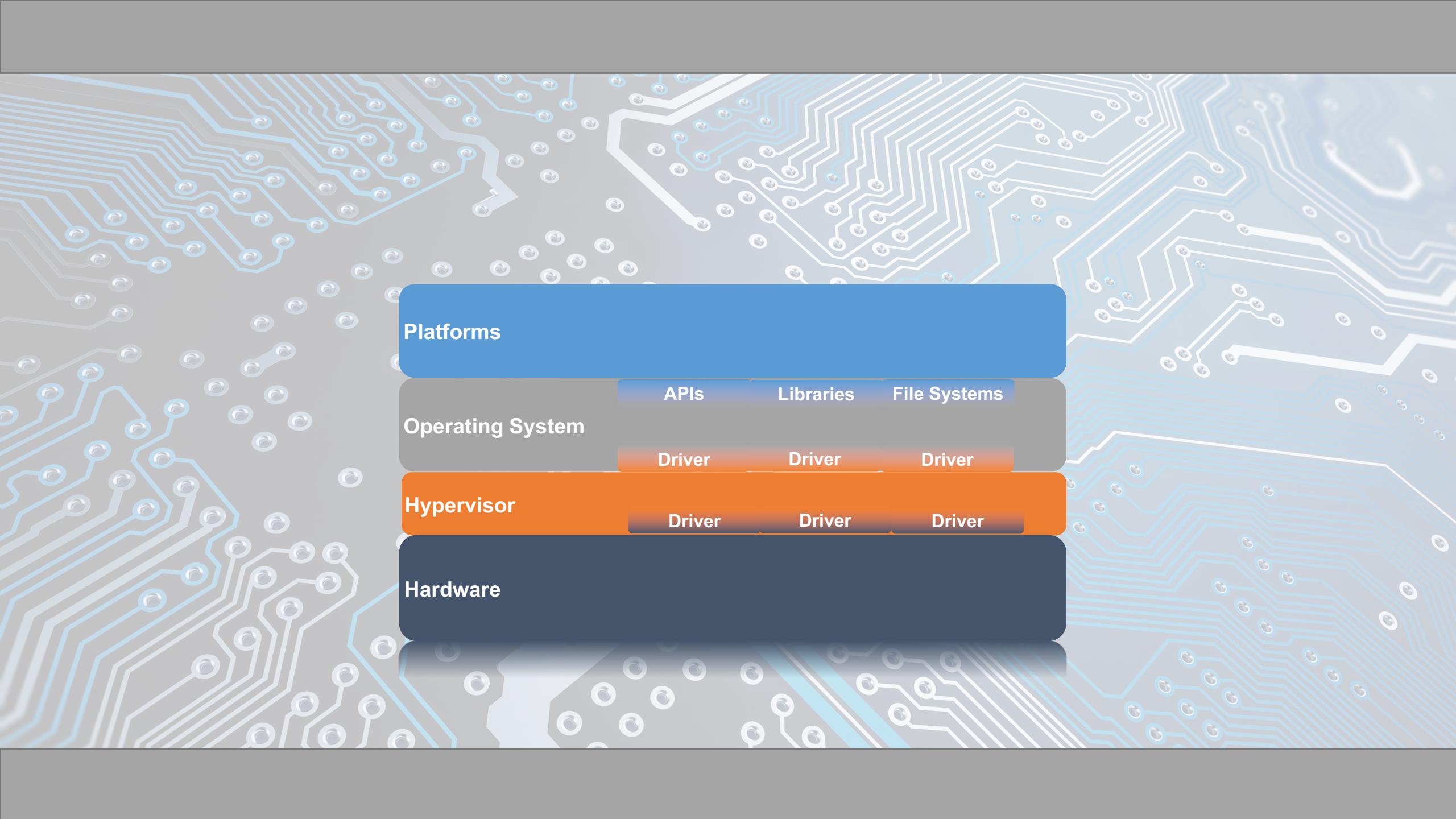
Driver

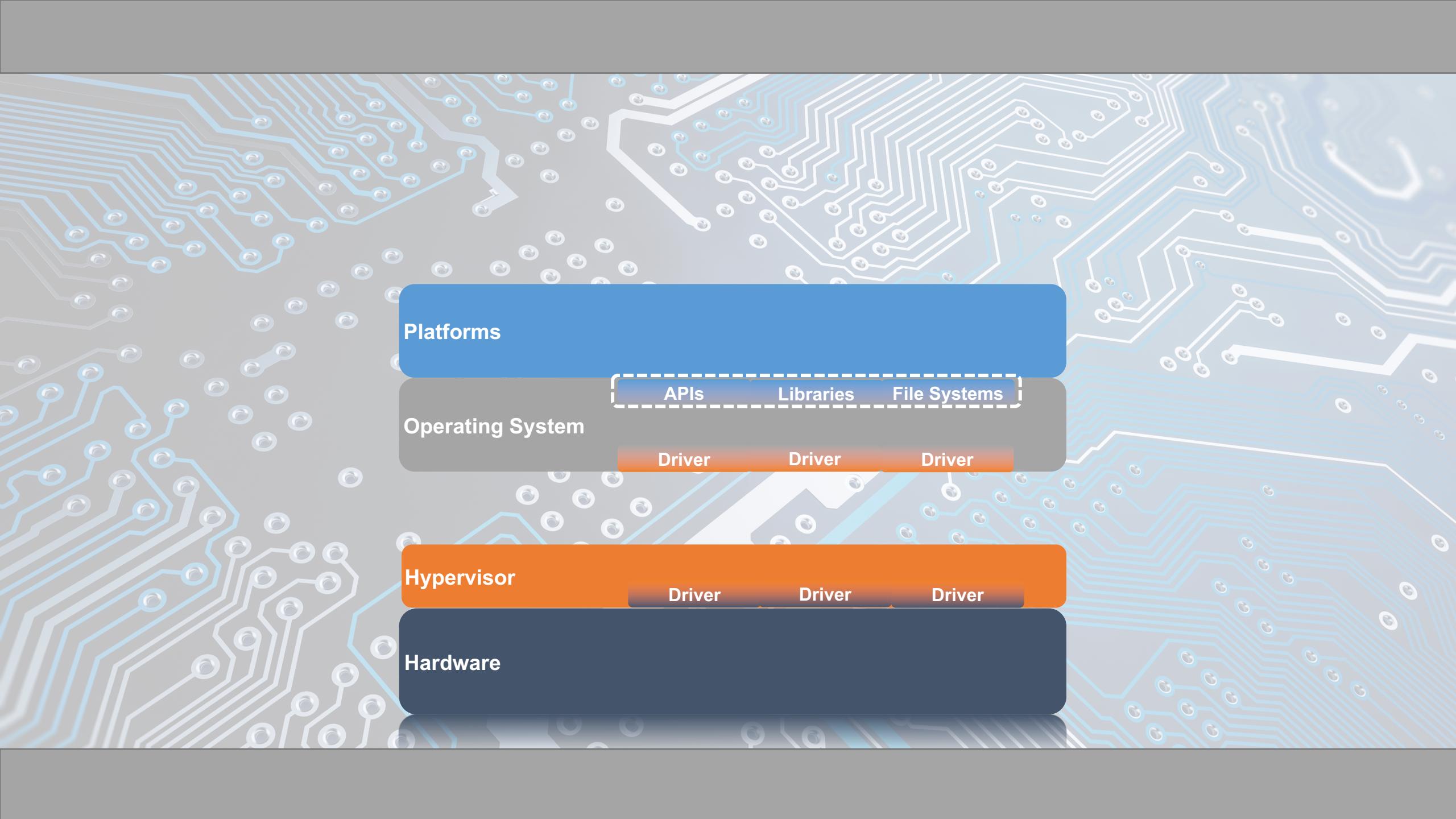
Hardware

Driver

Driver

Driver





A diagram illustrating the layered architecture of a system, set against a background of a printed circuit board (PCB) with blue traces and grey components. The layers are represented by rounded rectangles of increasing size from bottom to top.

- Hardware**: The base layer, colored dark grey.
- Hypervisor**: A middle layer, colored orange. It contains three "Driver" labels, each aligned under a corresponding "APIs", "Libraries", and "File Systems" label from the operating system layer above it.
- Operating System**: A grey layer containing three "Driver" labels, each aligned under a corresponding "APIs", "Libraries", and "File Systems" label from the platforms layer above it.
- Platforms**: The top layer, colored blue. It contains three "Driver" labels, each aligned under a corresponding "APIs", "Libraries", and "File Systems" label from the operating system layer below it.

Driver

Driver

Driver

APIs

Libraries

File Systems

Operating System

Hypervisor

Hardware

Platforms

Platforms

Virtualization Layer

Operating System

APIs

Libraries

File Systems

Driver

Driver

Driver

Hypervisor

Driver

Driver

Driver

Hardware

Platforms

Container

Operating System

APIs

Libraries

File Systems

Platforms

Container

Operating System

APIs

Libraries

File Systems

Platforms

APIs

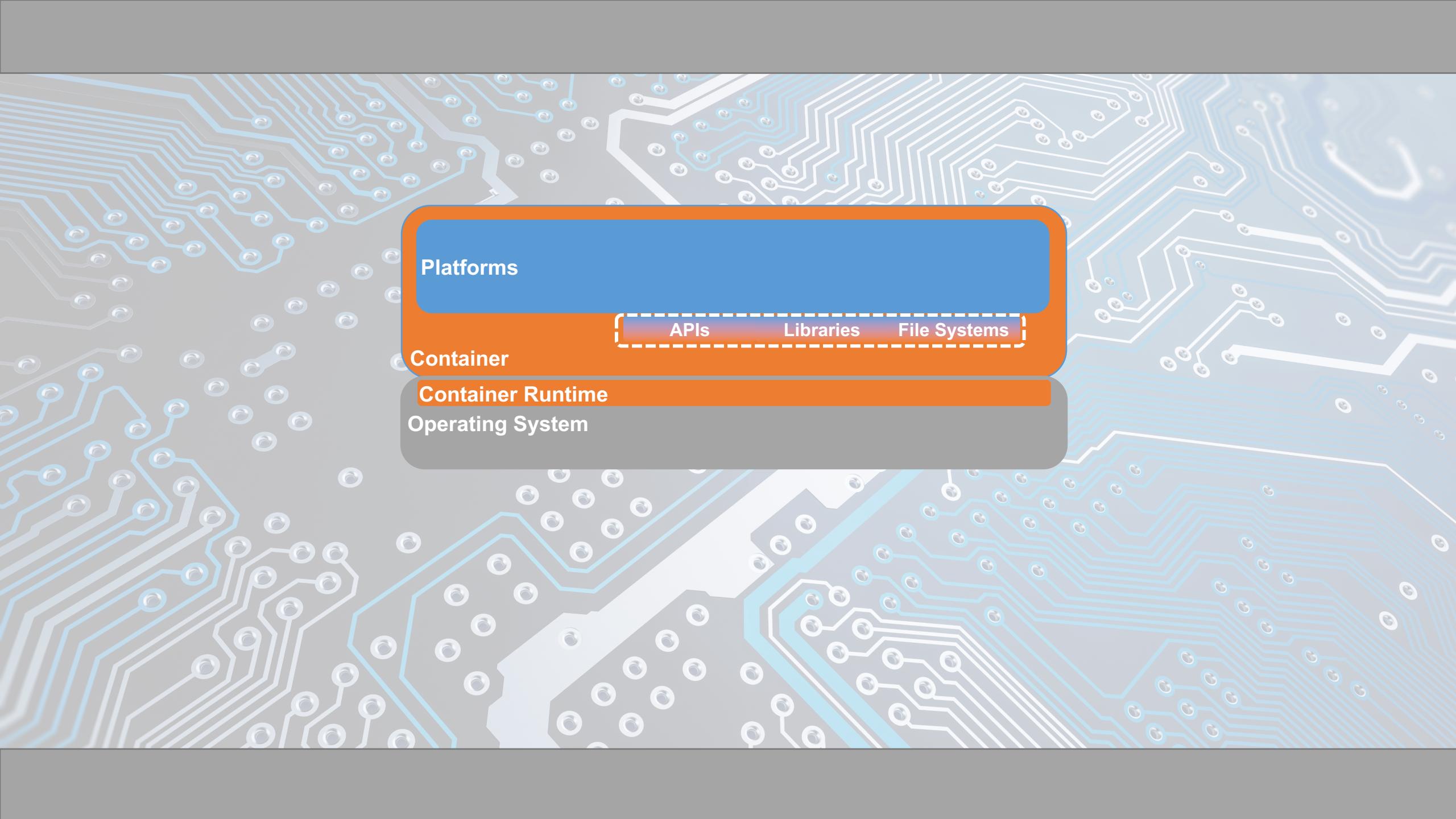
Libraries

File Systems

Container

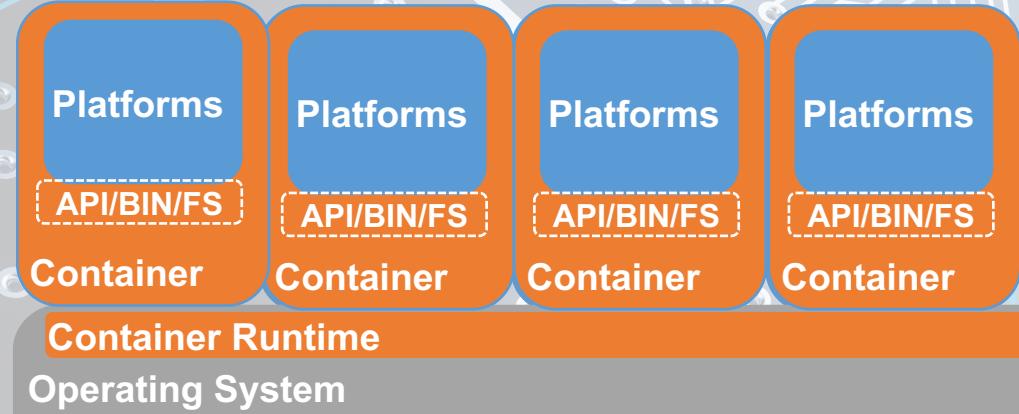
Container Runtime

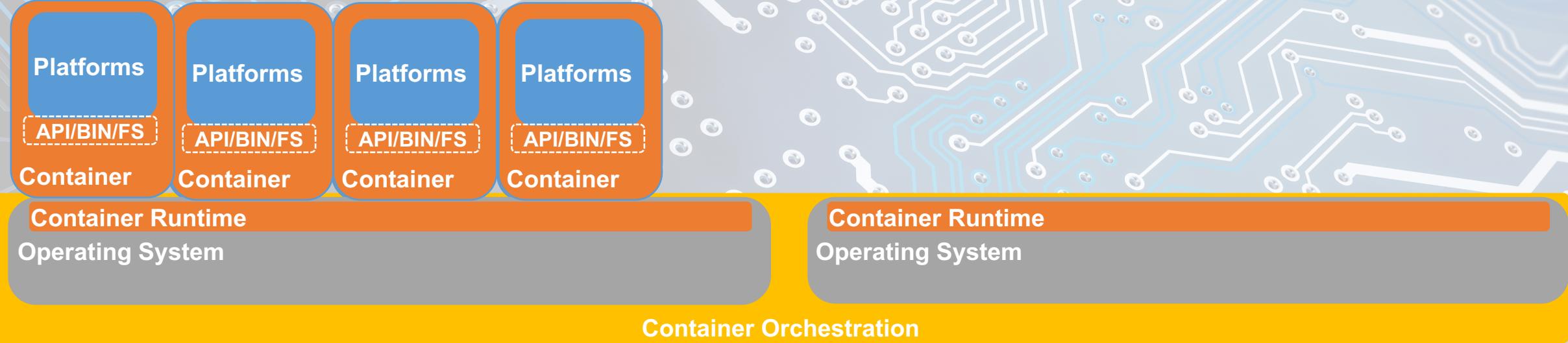
Operating System

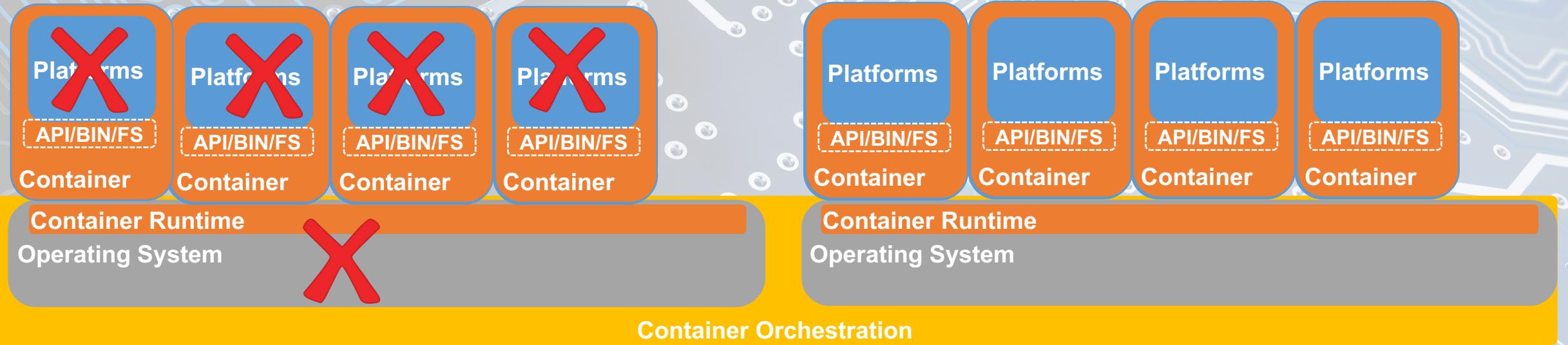


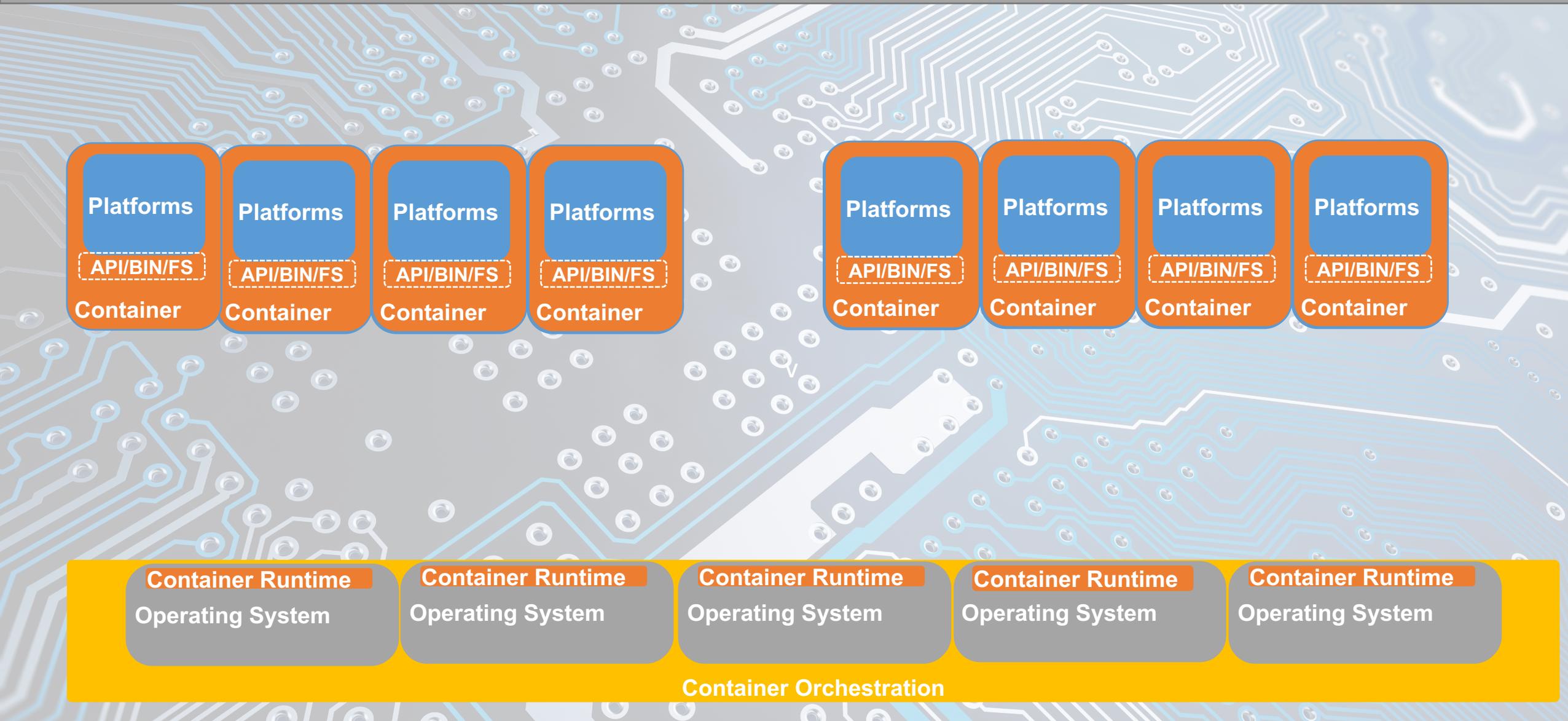


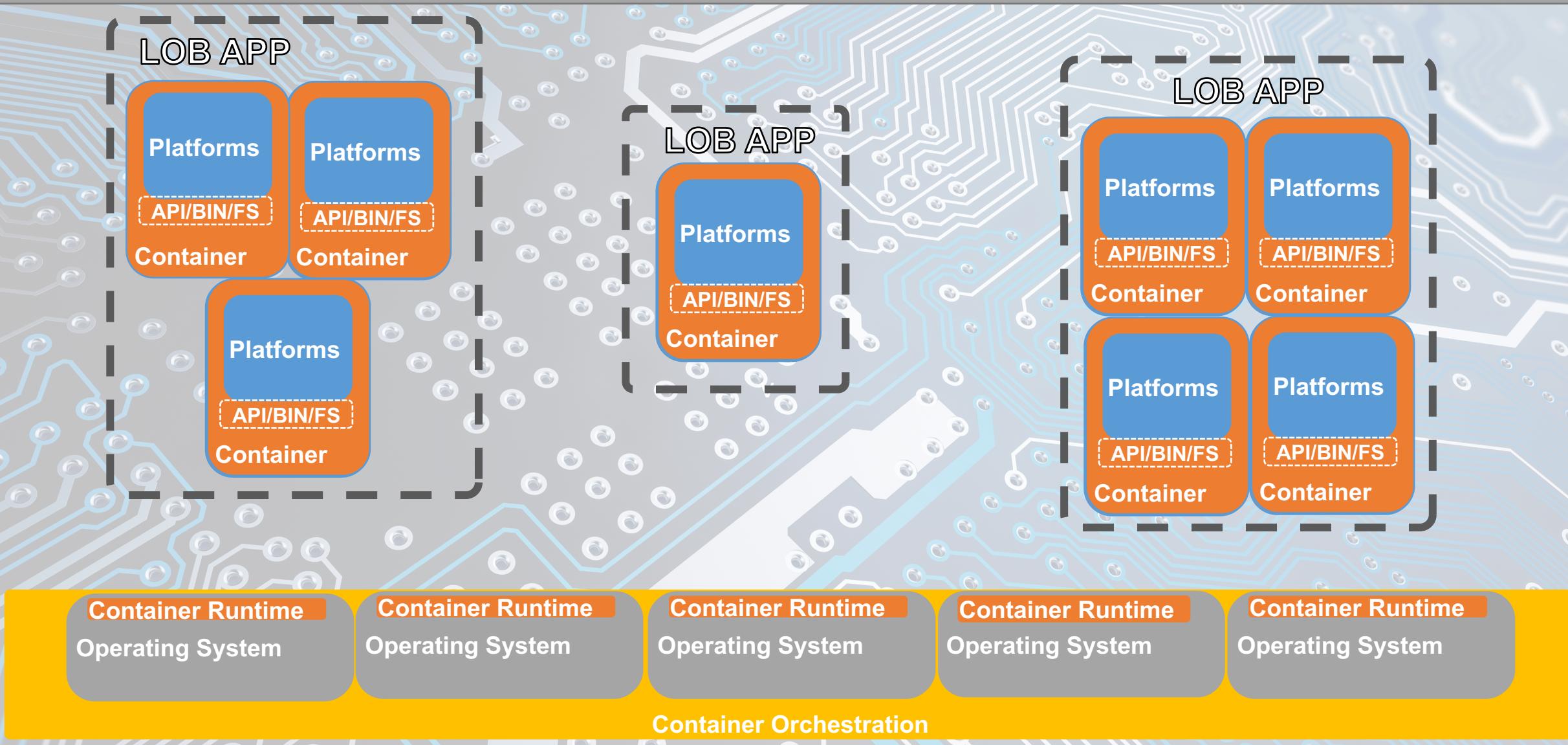
Container Runtime
Operating System

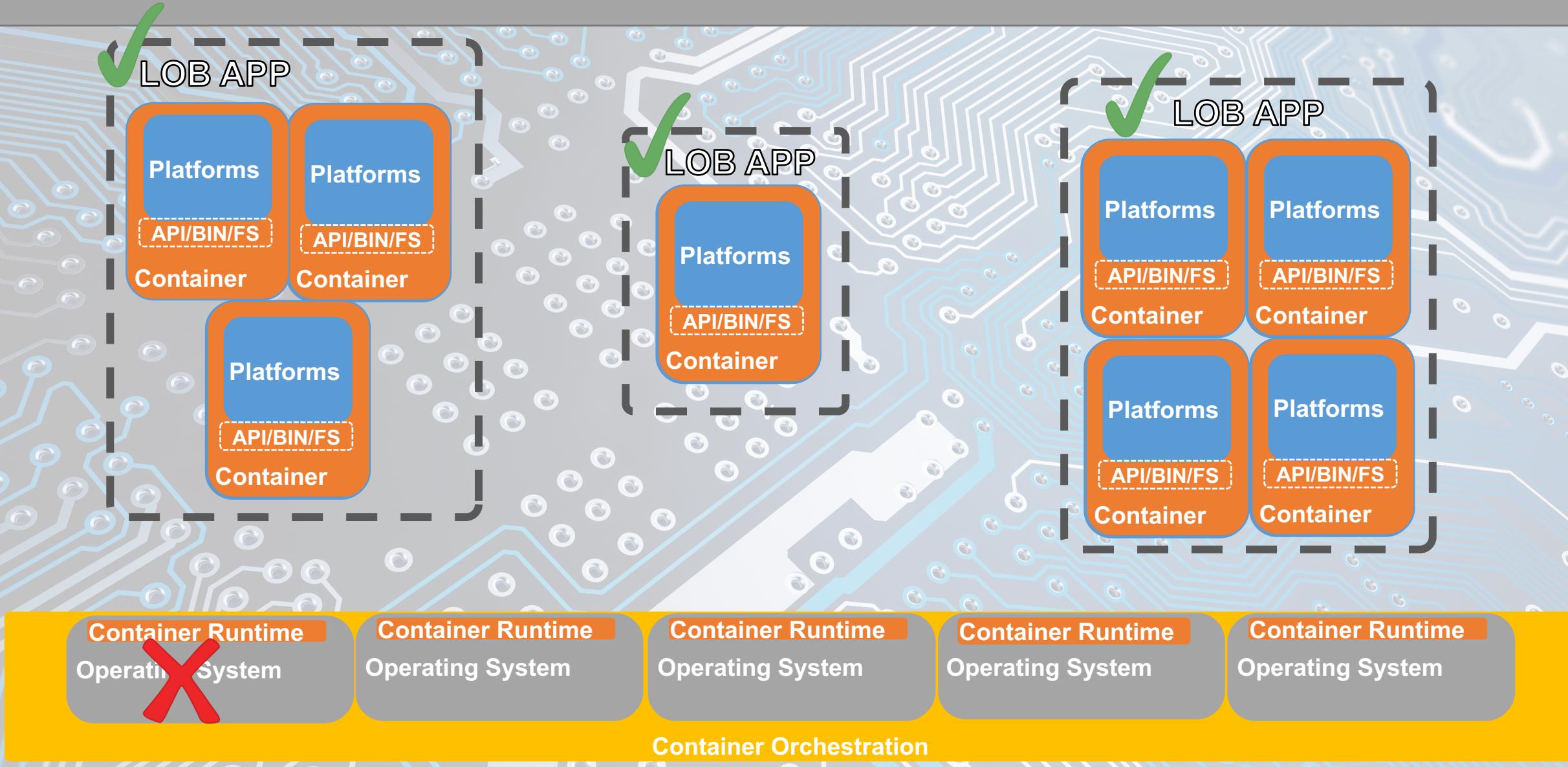


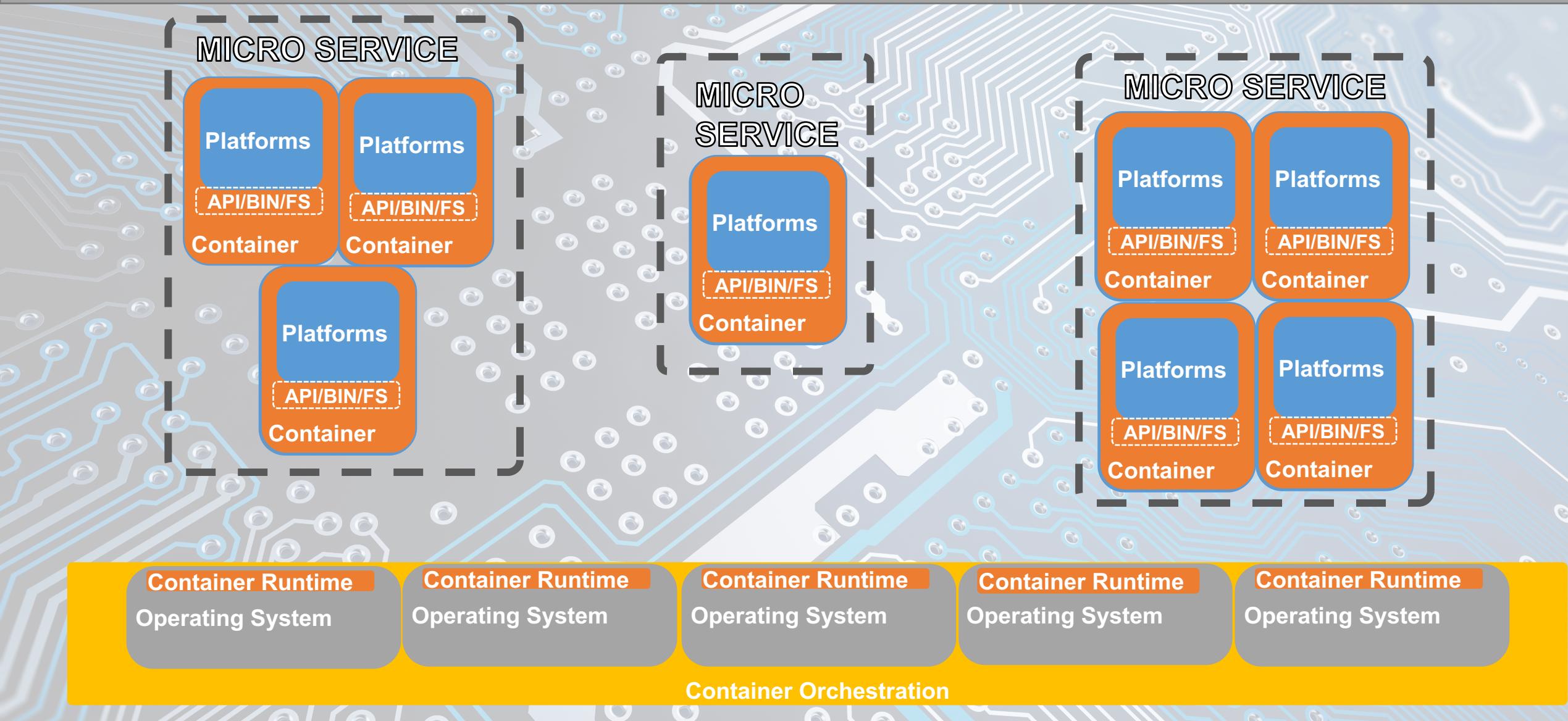


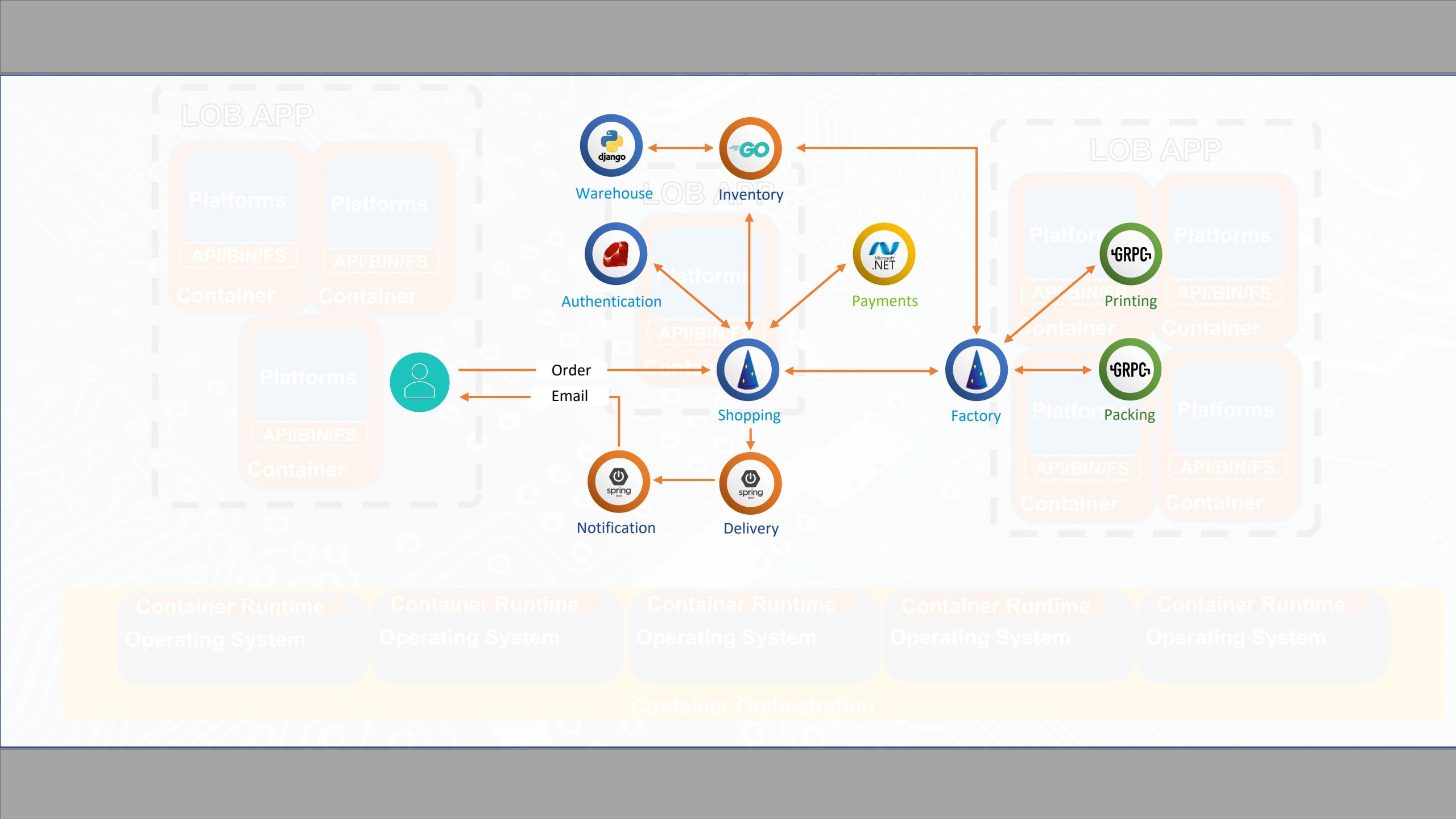


















1979

Development of “chroot”, allowed
storage level process isolation

1979 Development of “chroot”, allowed storage level process isolation

2000

“jail” command release, extending
isolation capability to users,
network & more.

1979 Development of “chroot”, allowed storage level process isolation
2000 “jail” command release, extending isolation capability to users, network & more.



2003

Google begins work on BORG

1979 Development of “chroot”, allowed storage level process isolation

2000 “jail” command release, extending isolation capability to users, network & more.

2003 Google begins work on BORG

2004

Solaris Zones released, providing user, process and filesystem isolation.

1979 Development of “chroot”, allowed storage level process isolation

2000 “jail” command release, extending isolation capability to users, network & more.

2003 Google begins work on BORG

2004 Solaris Zones released, providing user, process and filesystem isolation.

2007

AIX workload partitions release.
Providing containers for Power.

network & more.

2003 Google begins work on BORG

2004 Solaris Zones released, providing user, process and filesystem isolation.

2007 AIX workload partitions release. Providing containers for Power.

2013

Docker announced at PyCon

network & more.

2003 Google begins work on RQBC

2004 Solaris
isolation

2007 AIX wo

IN TARGET
0 HUG

PYCON US 2013

The future of Linux Containers

presented by

Solomon Hykes



DIAMOND SPONSOR



heroku

Google



filesystem

s for Power.

on

network & more.

2003 Google begins work on BORG

2004 Solaris Zones released, providing user, process and filesystem isolation.

2007 AIX workload partitions release. Providing containers for Power.

2013

Docker announced at PyCon

network & more.

2003 Google begins work on BORG

2004 Solaris Zones released, providing user, process and filesystem isolation.

2007 AIX workload partitions release. Providing containers for Power.

2013 Docker announced at PyCon

2015

Google replaces “LMCTFY” with
Docker engine (libcontainer)

2007 AIX workload partitions release. Providing containers for Power.
2013 Docker announced at PyCon

2015

Google replaces “LMCTFY” with
Docker engine (libcontainer)

Kubernetes 1.0 Released

2007 AIX workload partitions release. Providing containers for Power.

2013 Docker announced at PyCon

2015 Google replaces “LMCTFY” with Docker engine (libcontainer)

Kubernetes 1.0 Released

2017

Kubernetes becomes ‘defacto’
orchestration engine & widespread
adoption in enterprises

2007 AIX workload partitions release. Providing containers for Power.

2013 Docker announced at PyCon

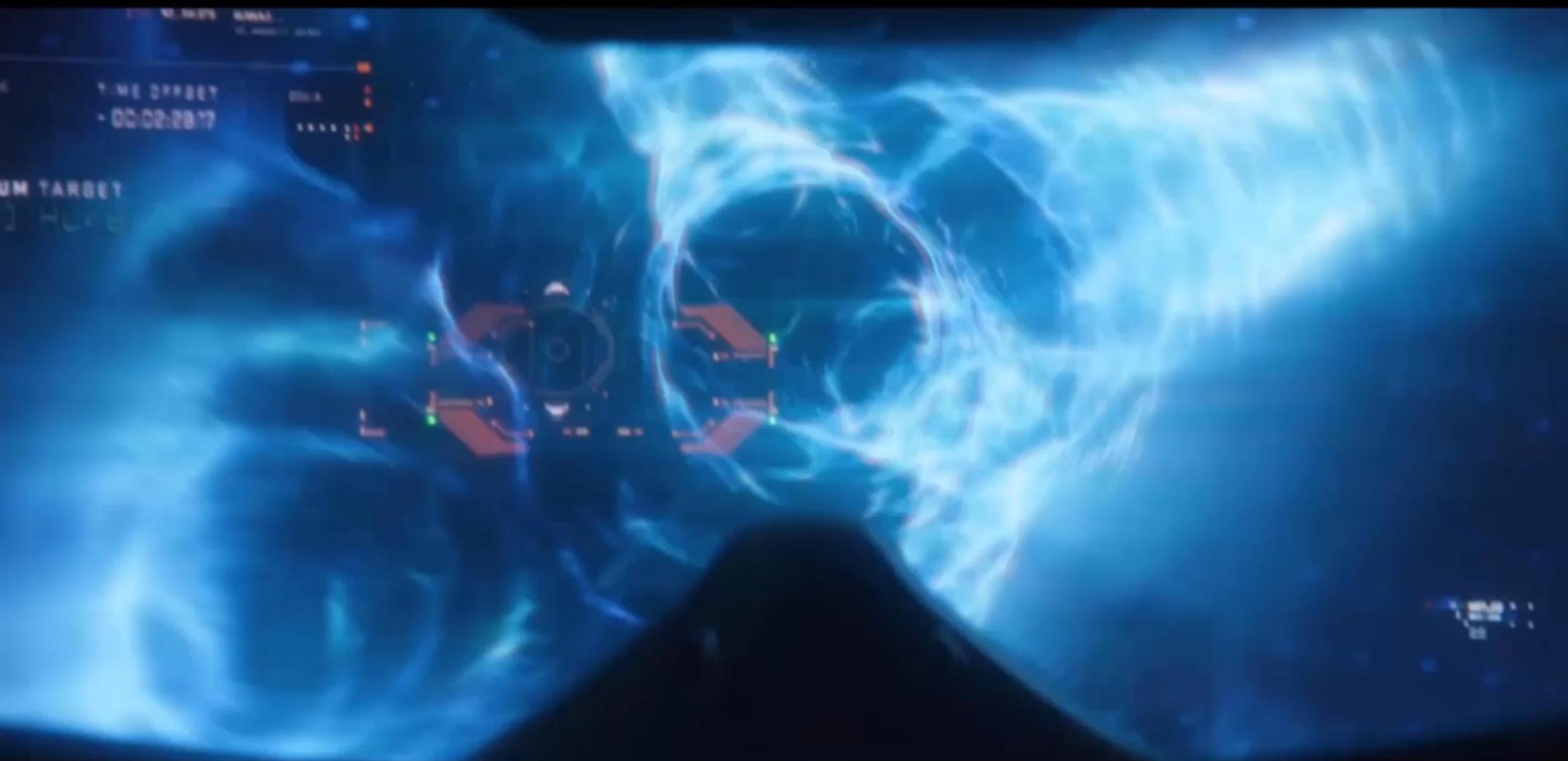
2015 Google replaces “LMCTFY” with Docker engine (libcontainer)

Kubernetes 1.0 Released

2017 Kubernetes becomes ‘defacto’ orchestration engine & widespread adoption in enterprises

2019

Vmware announced
Tanzu & Project Pacific





Containers vs VMs

Containers and virtual machines have similar resource isolation and allocation benefits, but function differently because containers virtualize the operating system instead of hardware. Containers are more portable and efficient.

CONTAINERS

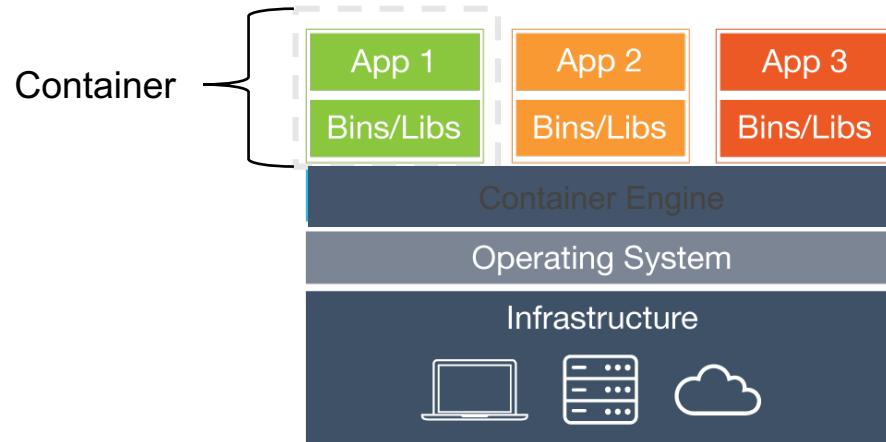
Containers are an abstraction at the app layer that packages code and dependencies together. Multiple containers can run on the same machine and share the OS kernel with other containers, each running as isolated processes in user space. Containers take up less space than VMs (container images are typically tens of MBs in size), can handle more applications and require fewer VMs and Operating systems.

VIRTUAL MACHINES

Virtual machines (VMs) are an abstraction of physical hardware turning one server into many servers. The hypervisor allows multiple VMs to run on a single machine. Each VM includes a full copy of an operating system, the application, necessary binaries and libraries - taking up tens of GBs. VMs can also be slow to boot.

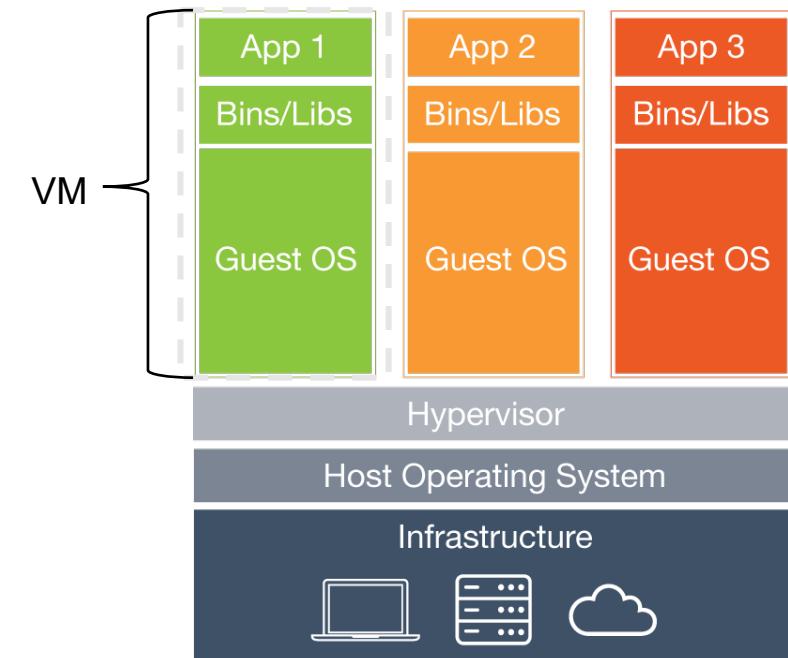
CONTAINERS

- Shared Kernel
- Nascent technology created complexity
- Super fast to create a new container
- Containers are very portable



VIRTUAL MACHINES

- Requires a full operating system
- Mature technology and management eco-system
- Slow to create a new VM
- VMs are not very portable



DECLARATIVE

“I would like some tea”

...Define “tea”

Tea is an infusion of tea¹ leaves in a cup

¹An infusion is obtained by letting the object steep a few minutes in hot water²

²Hot water is obtained by pouring it in a container and setting it on stove

IMPERATIVE

Boil some water.

Pour it in a teapot.

Add tea leaves.

Steep for a while.

Serve in a cup

DECLARATIVE

“I would like some tea”

- More complex
- Requires defining before hand
- Run automatically
- Knows current state of all components
- “Orchestration”
- YAML file

IMPERATIVE

Boil some water.
Pour it in a teapot.
Add tea leaves.
Steep for a while.
Serve in a cup

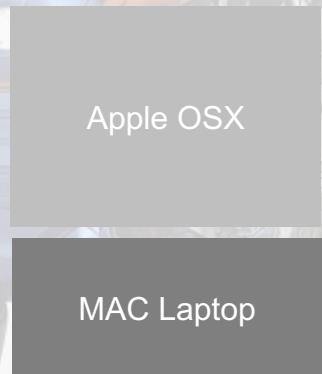
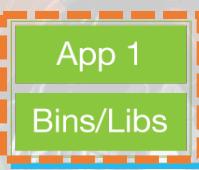
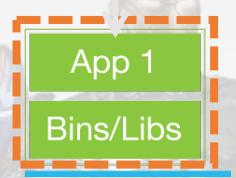
- Much simpler
- Requires scheduled action or human
- Run ad-hoc
- Lacks understanding of current state
- “Automation”
- Script

Consistency and Decoupling Throughout the Process

Define the App Environment



Developers



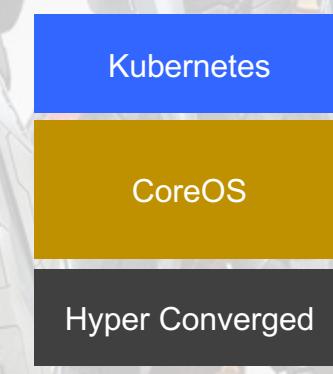
Development



Unit Test



Integration



Prod

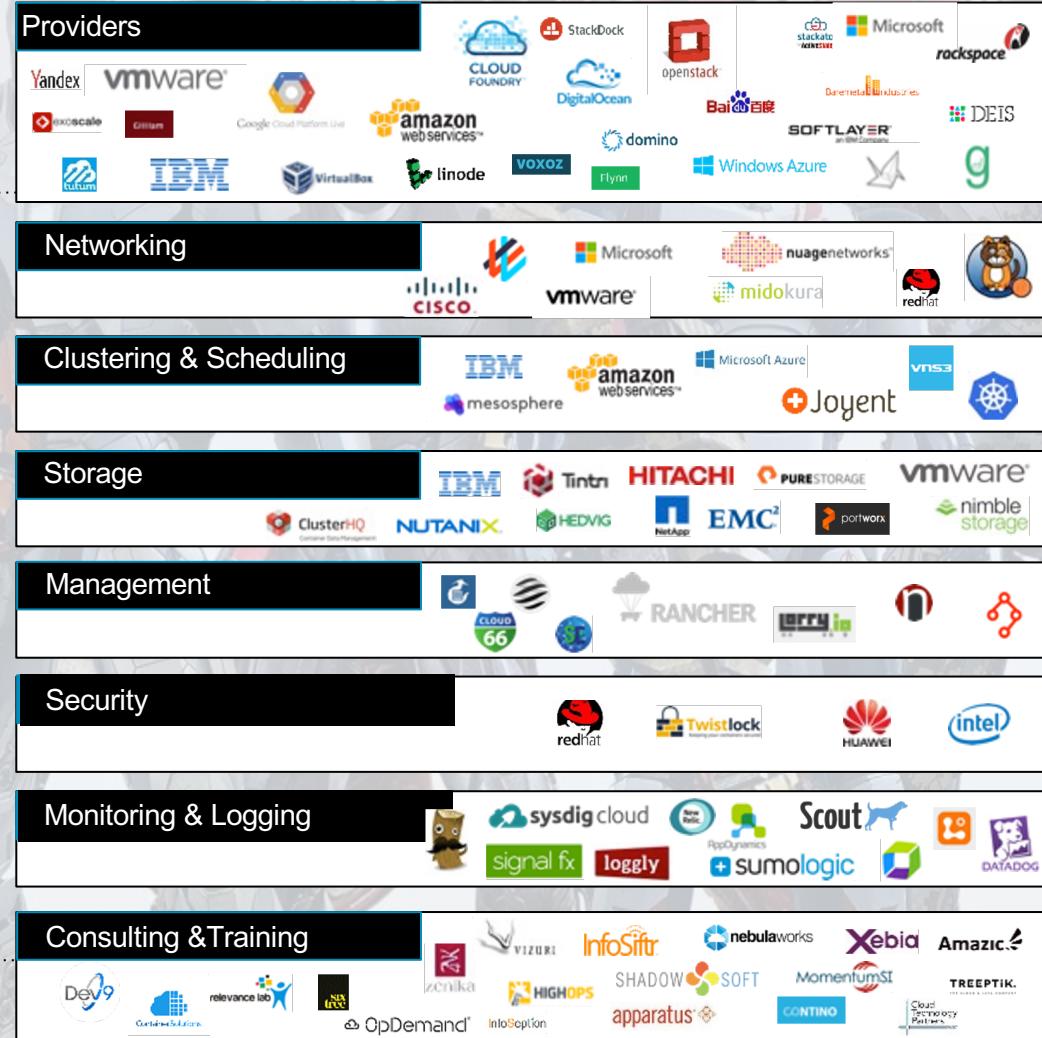
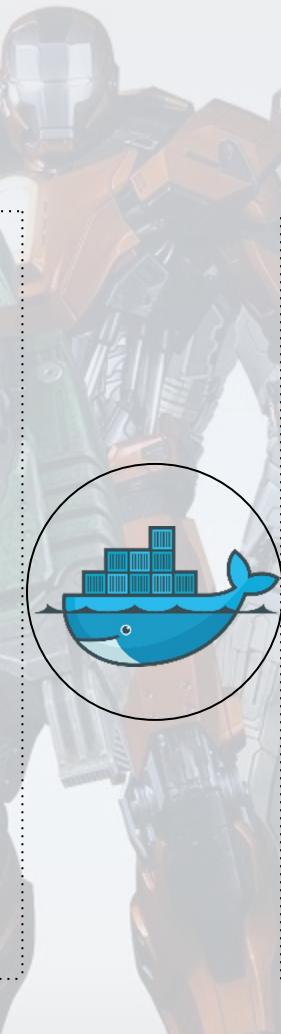
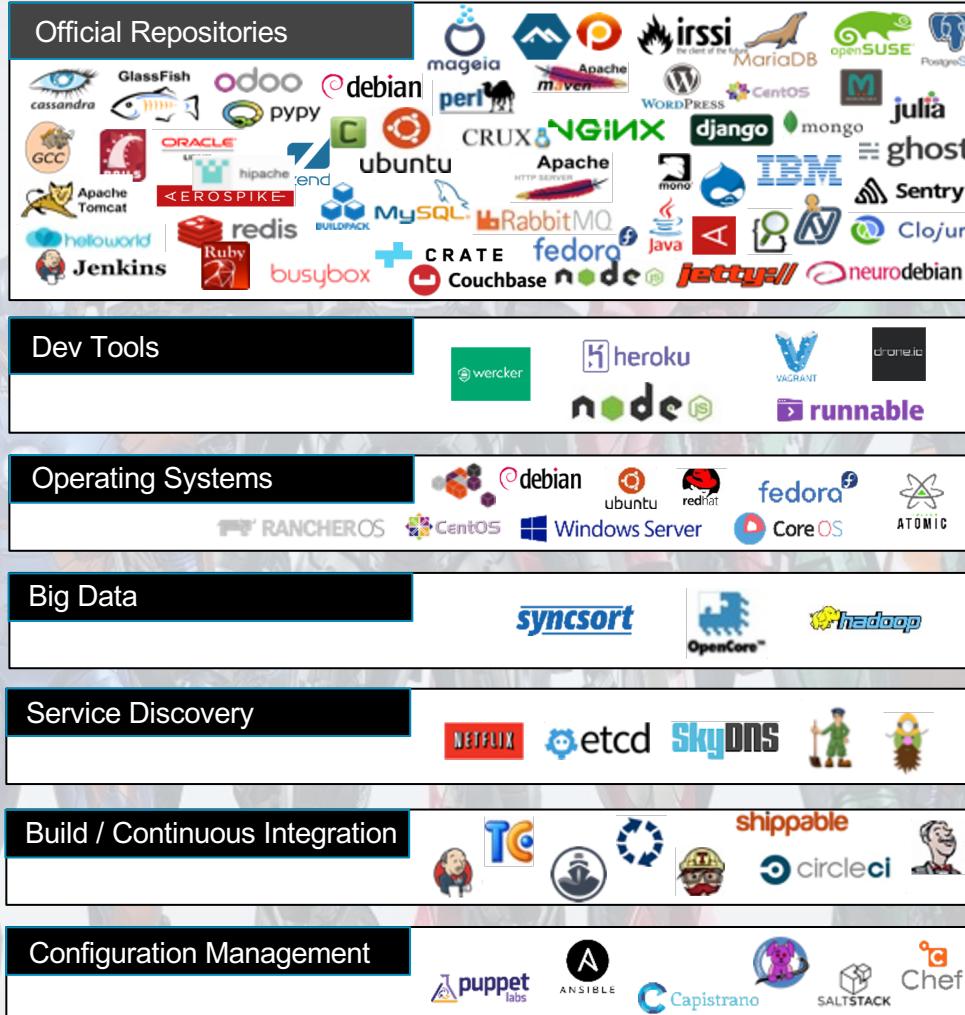


Operations

Container maintains consistent environment for application/code throughout the development and deployment process,

Container can be deployed and run virtually anywhere

THE DOCKER ECOSYSTEM





Introducing
VMware Tanzu

Build Modern Apps

Run Enterprise Kubernetes

Manage Kubernetes for Developers



Introducing VMware Tanzu

Build Modern Apps
Run Enterprise Kubernetes
Manage Kubernetes for Developers



Project Pacific: Rearchitecting vSphere with Native Kubernetes

Transforming vSphere into the App Platform of the Future

