CONTAINERS AND MICROSERVICES
FLIGHT DYNAMICS IN A BOX

Scott Lowe

Architecture Lead

Email: scott.lowe@ai-solutions.com

Website: ai-solutions.com/meridian

CONTAINERS AND MICROSERVICES
FLIGHT DYNAMICS IN A BOX

25 years experience building flight dynamics ground systems

Flight dynamics and mission engineering for NASA, NOAA, USGS, USAF/USSF, MDA

We work with both international civilian space agencies and commercial space companies

CONTAINERS AND MICROSERVICES
FLIGHT DYNAMICS IN A BOX

Script based astrodynamics and mission design tool

Used in over 30 counties on hundreds of missions including TDRS, ISS, Landsat, and GPS, and at over 150 universities

Handles constellations from 10 to 1000s of spacecraft

Flight Dynamics Ground System

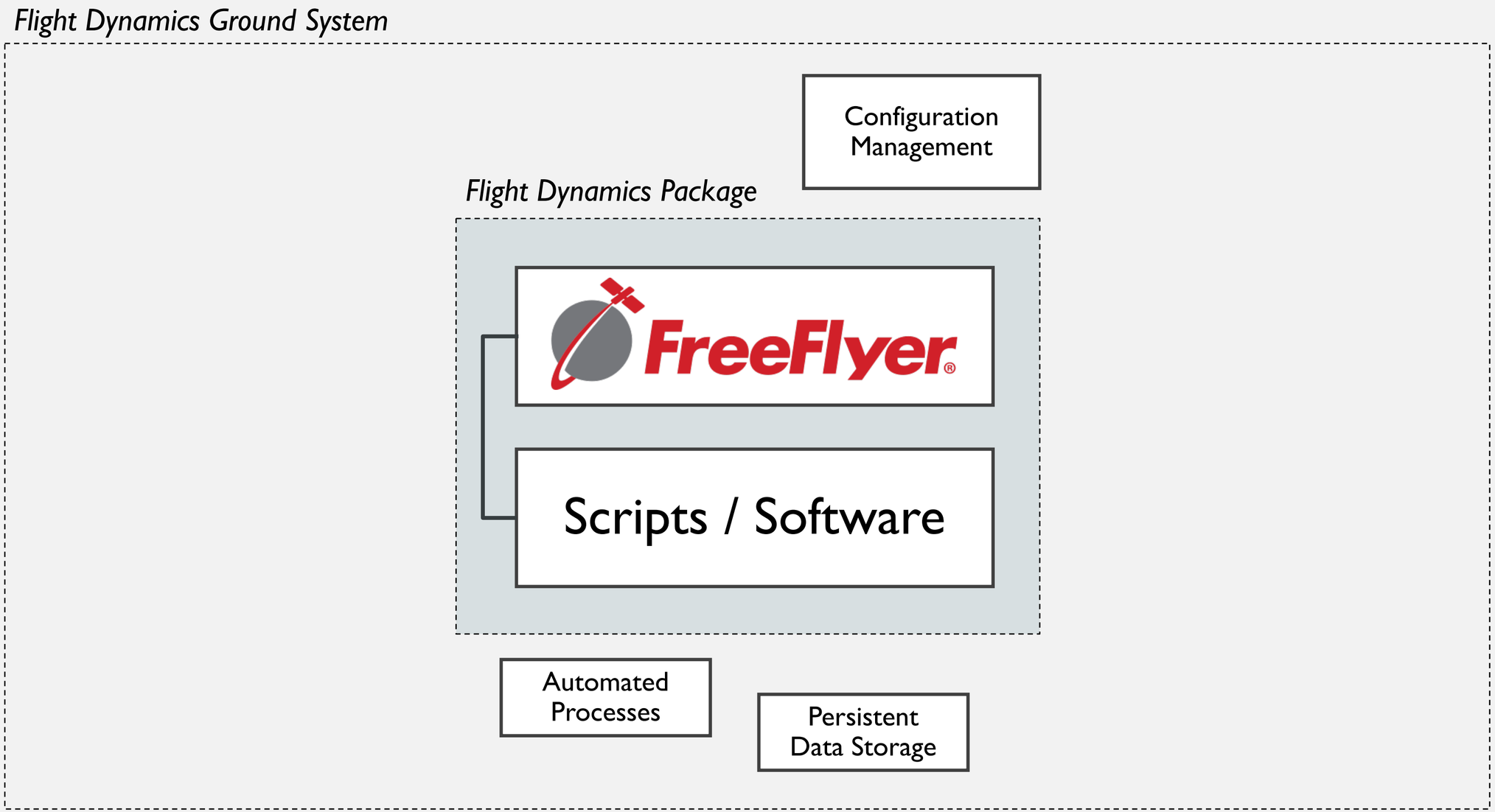
Flight Dynamics Engine

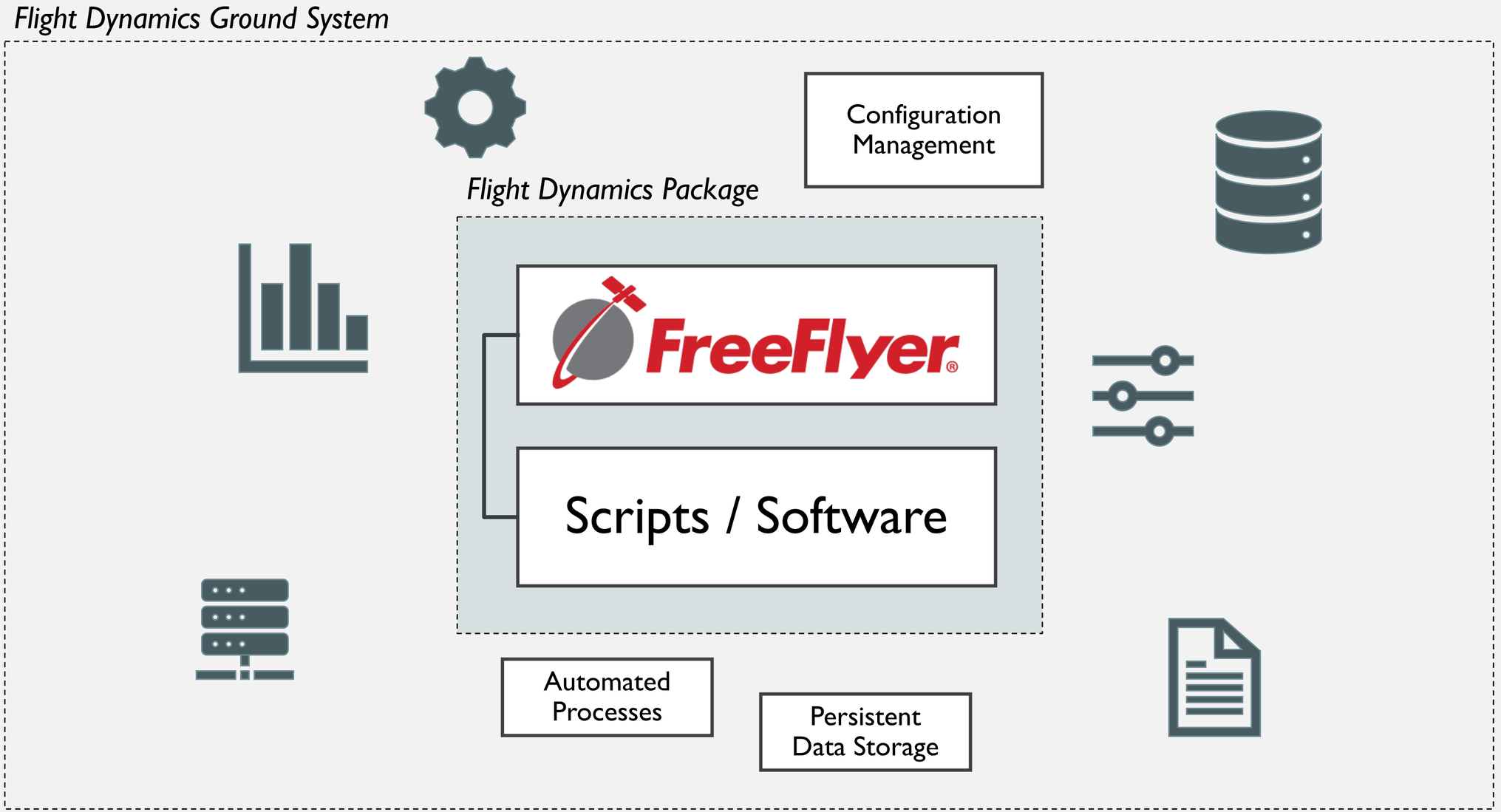
Flight Dynamics Ground System

Flight Dynamics Engine

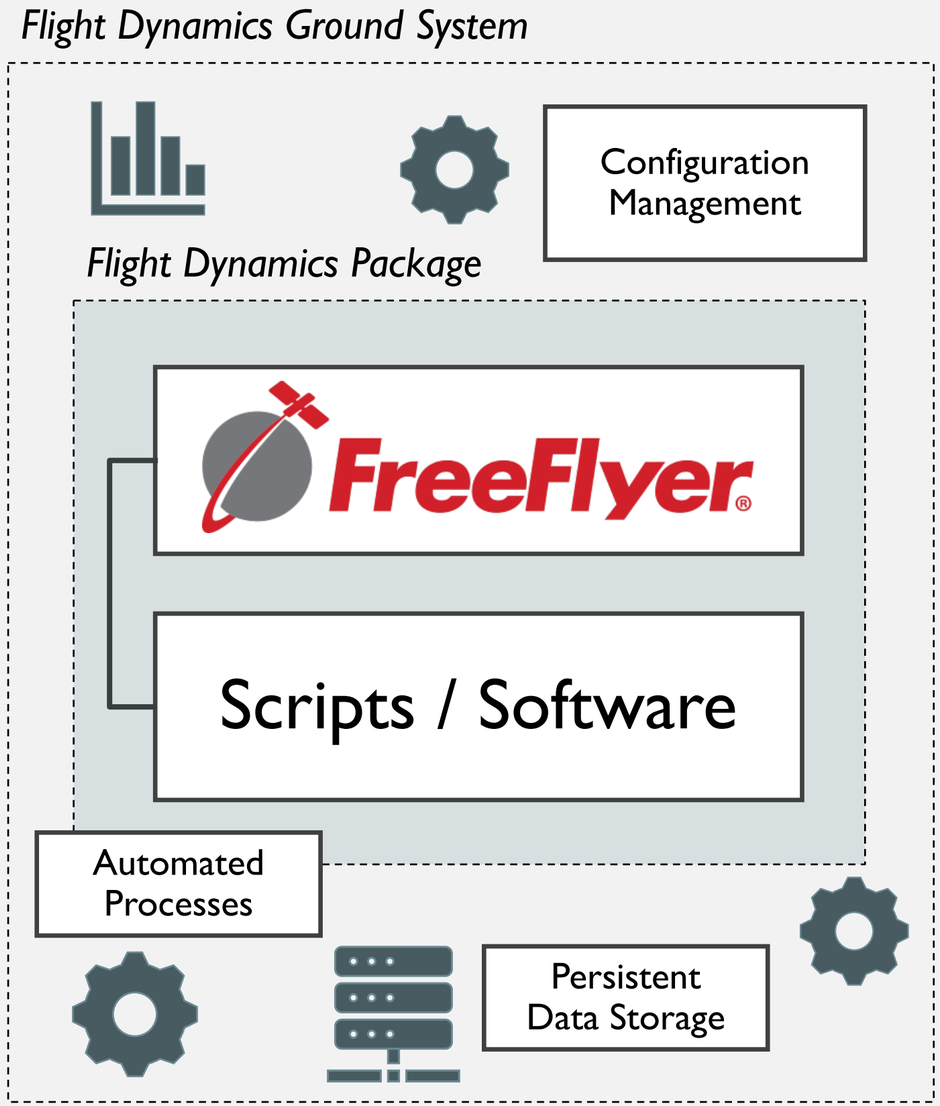
Flight Dynamics Ground System

Flight Dynamics Package

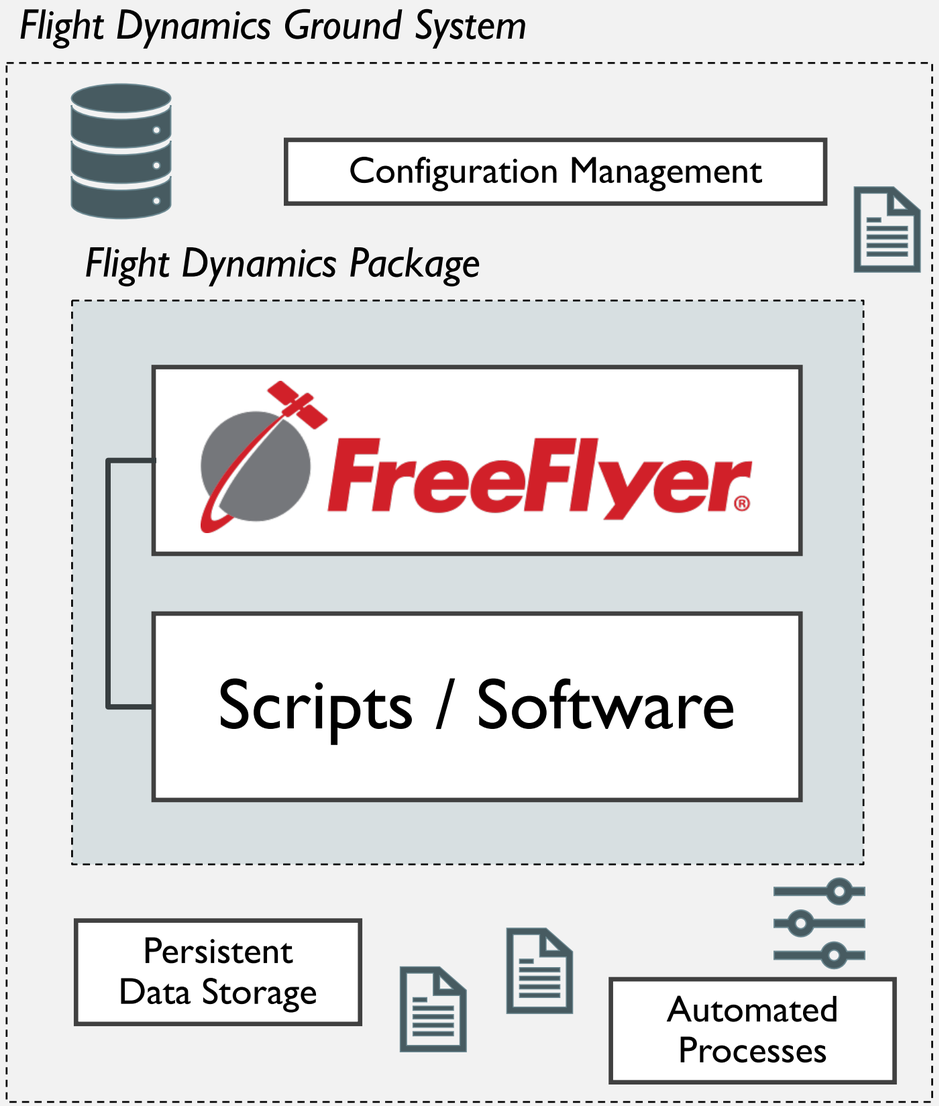

Flight Dynamics Ground System
Configuration Management
Flight Dynamics Package
FreeFlyer®
Scripts / Software
Automated Processes
Persistent Data Storage



Highly Customized Flight Dynamics System

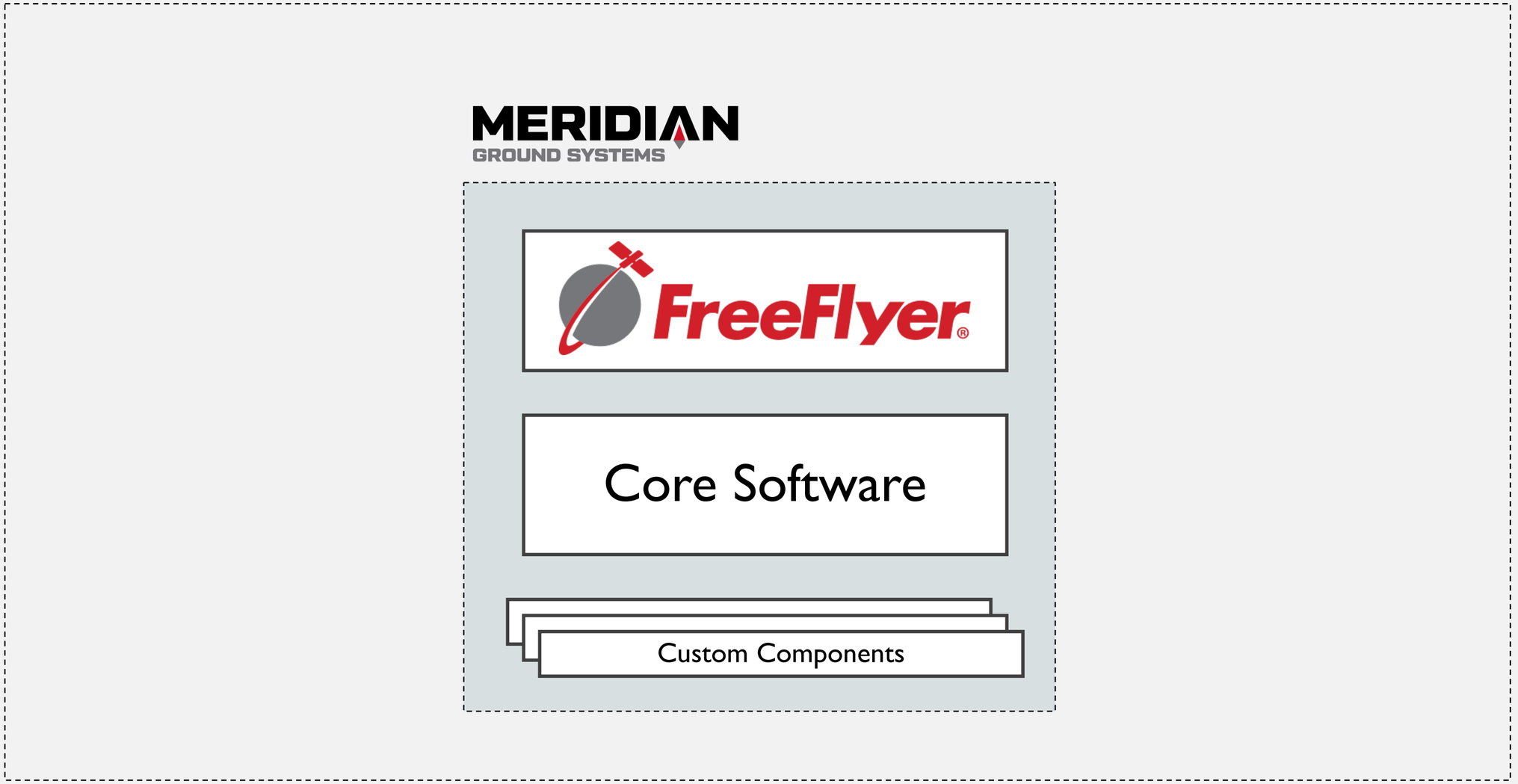


Mission A



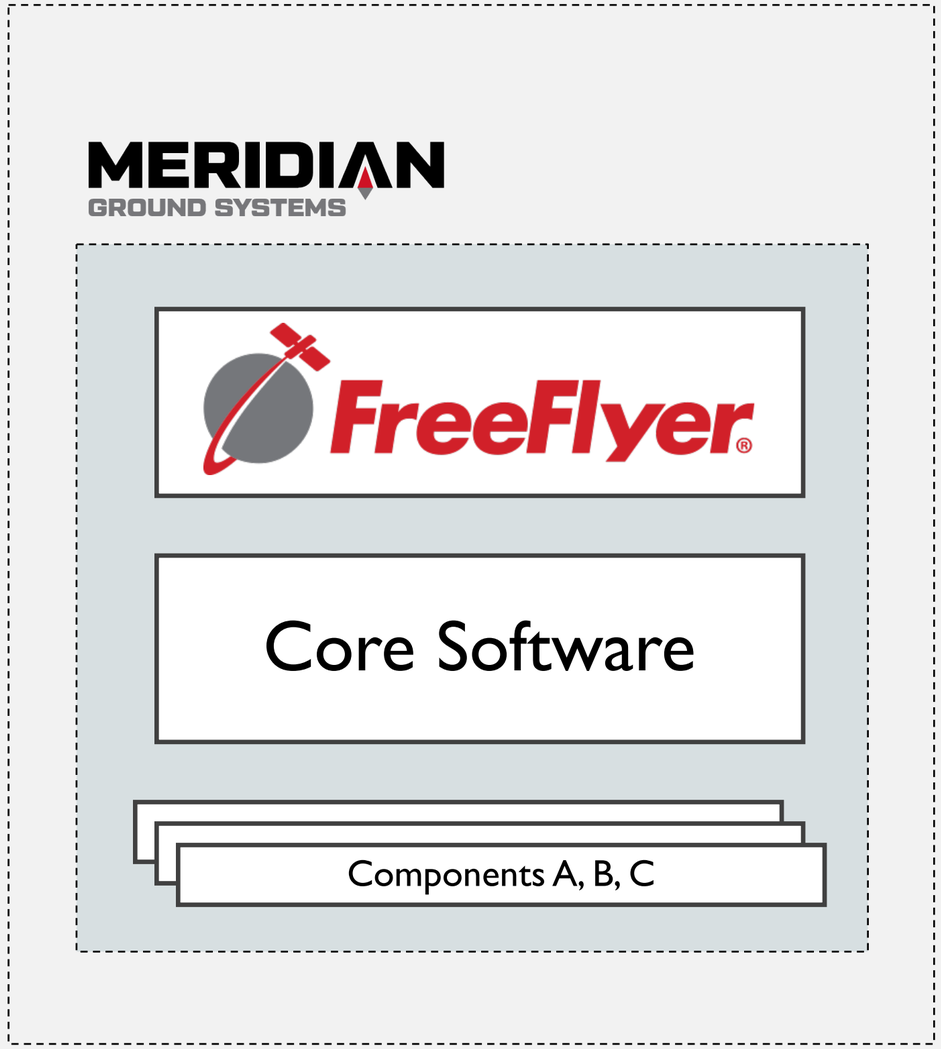
Mission B

Flight Dynamics Ground System



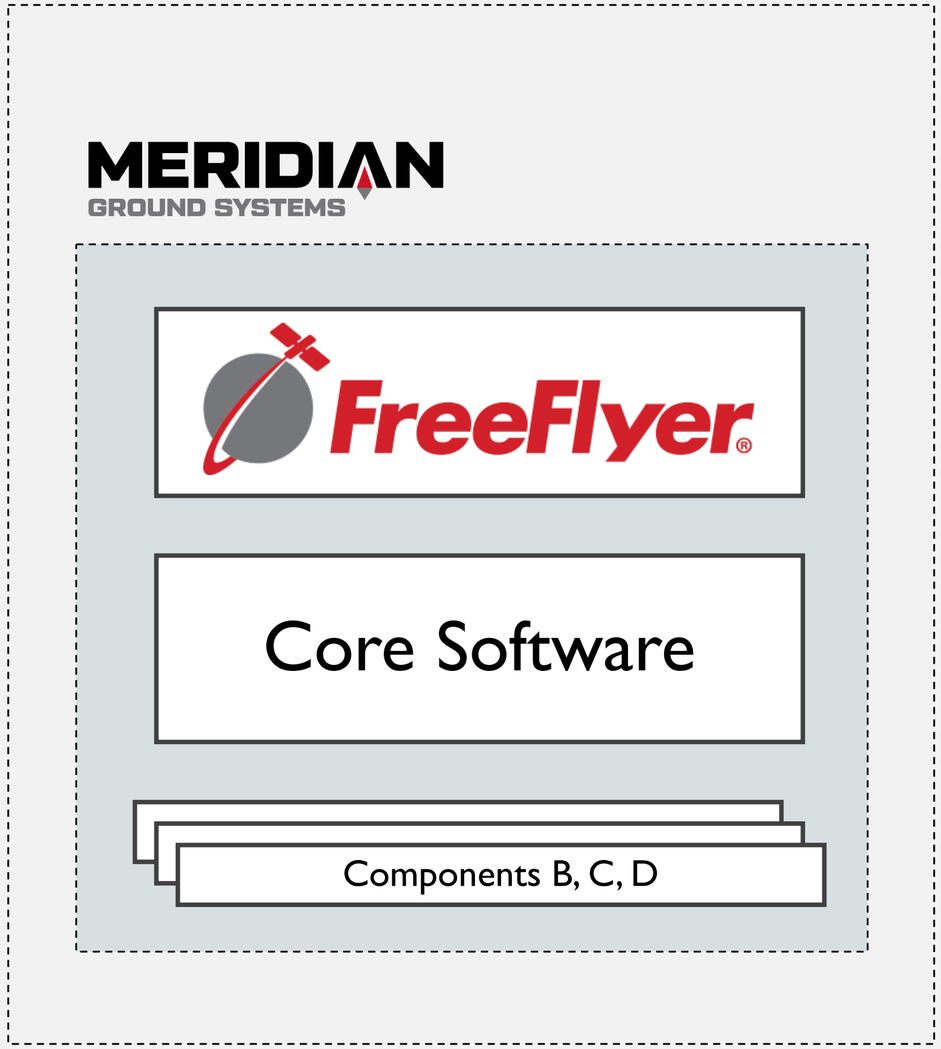
Flight Dynamics in a Box

Flight Dynamics Ground System

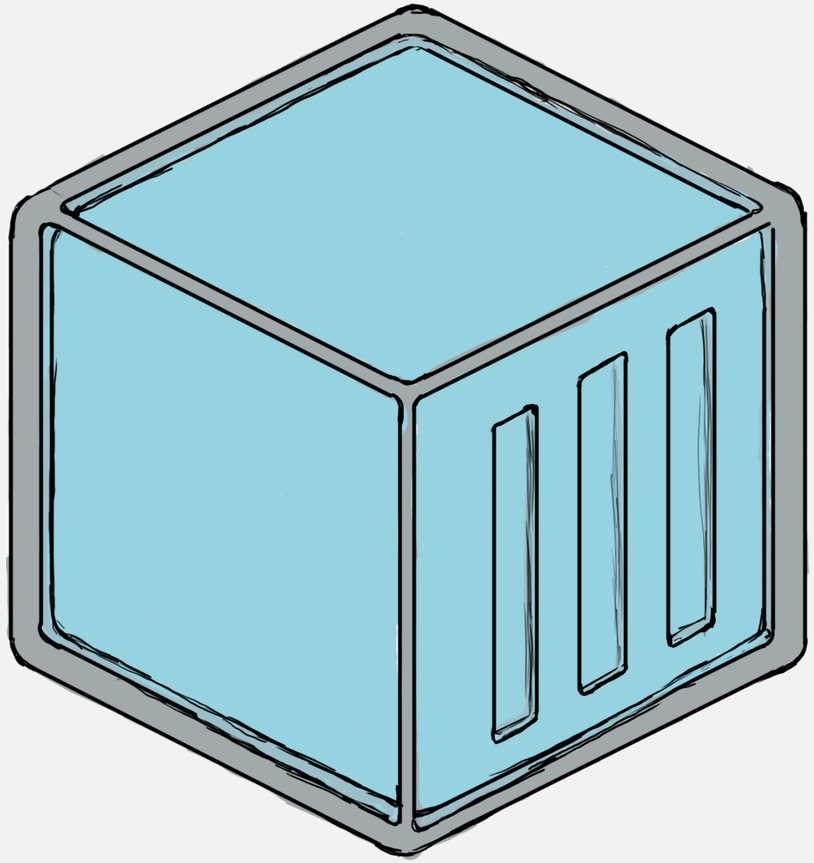


Packaged Flight Dynamics

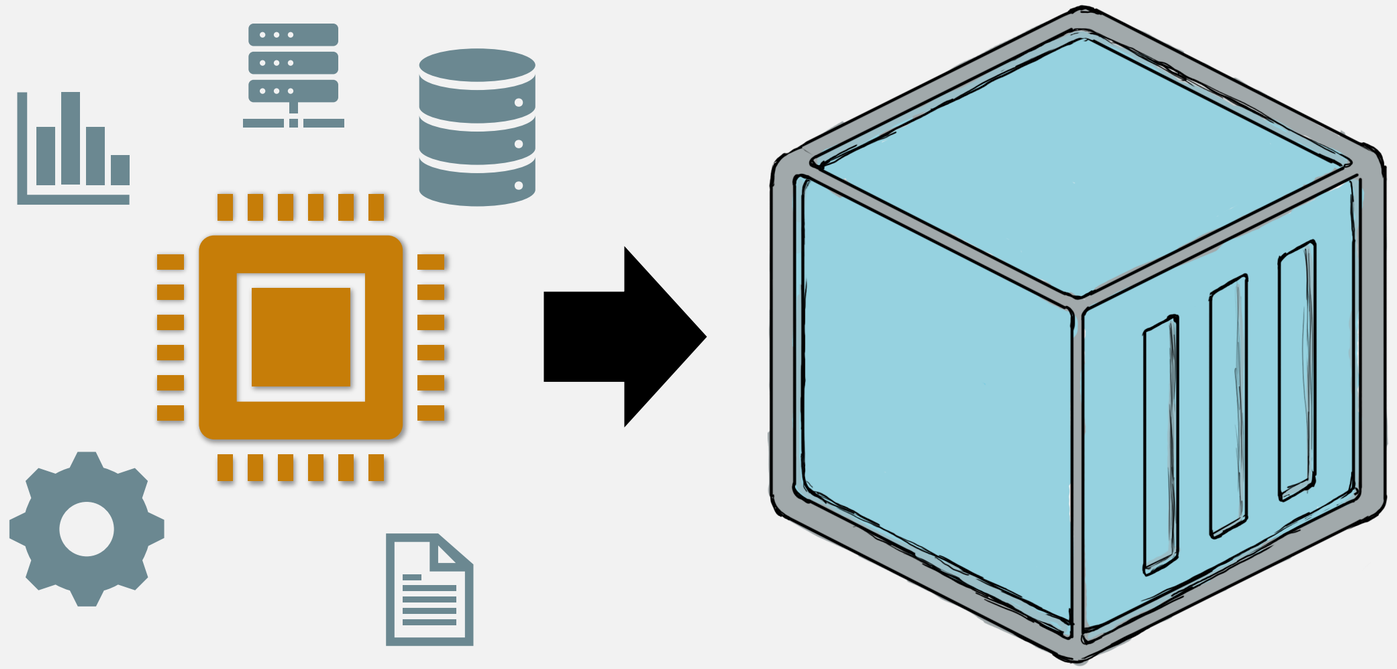
Flight Dynamics Ground System



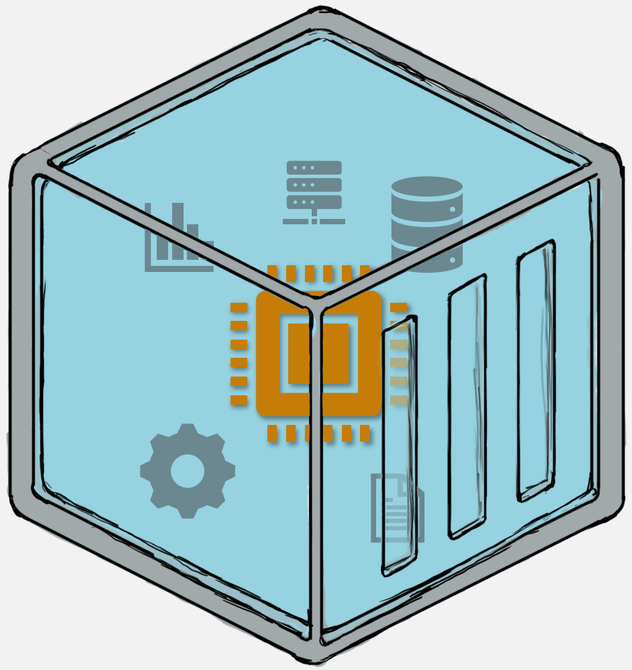
Packaged Flight Dynamics



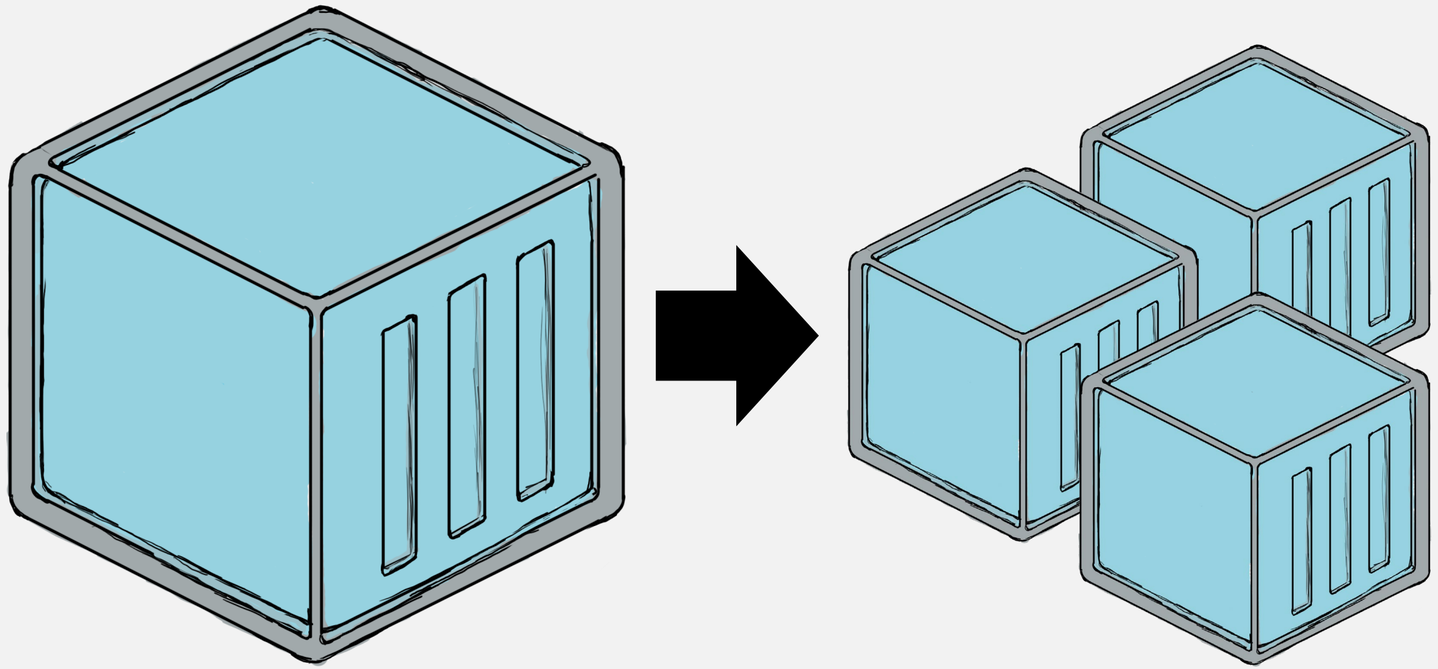
Containers



Encapsulation, isolation, portability, and control

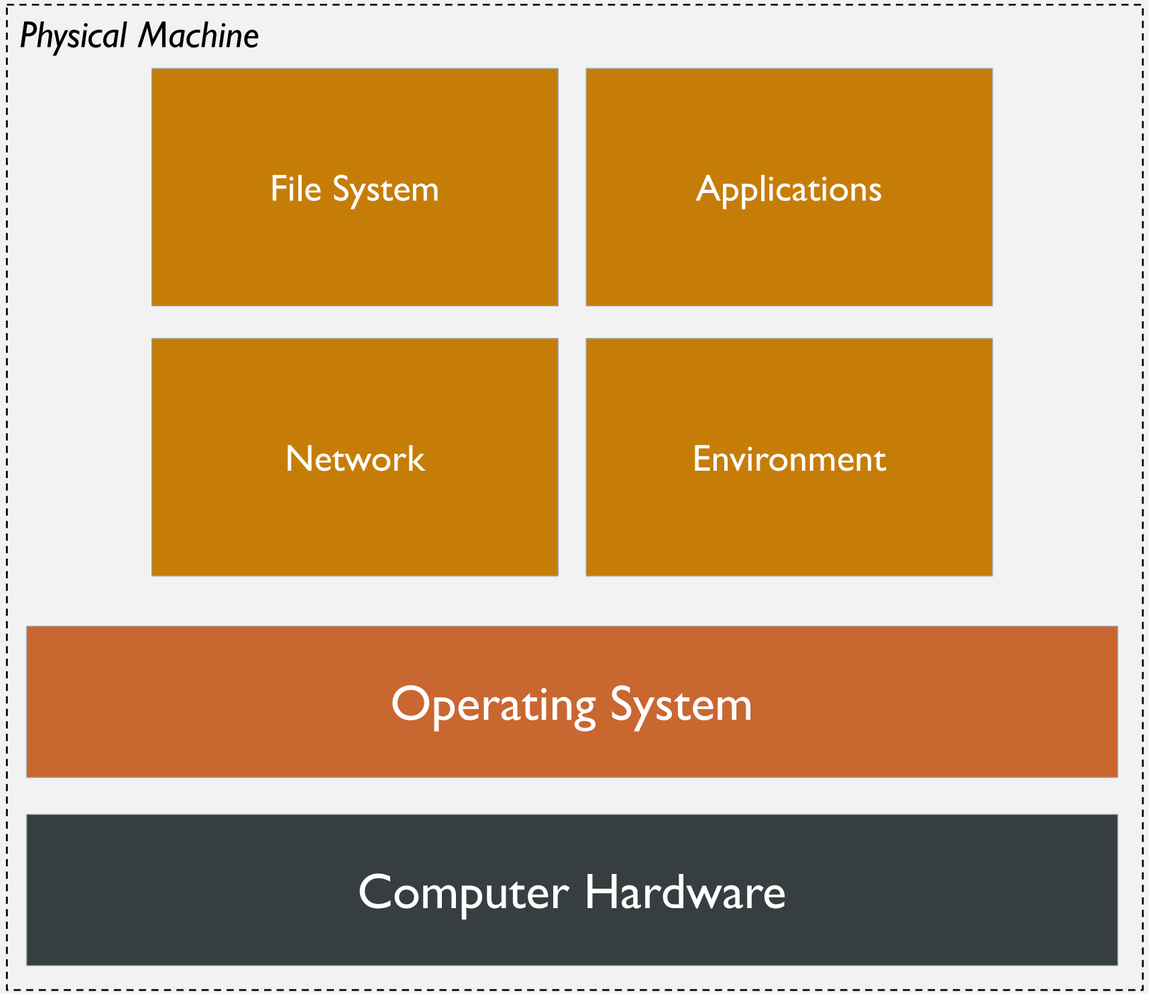


Encapsulation, isolation, portability, and control

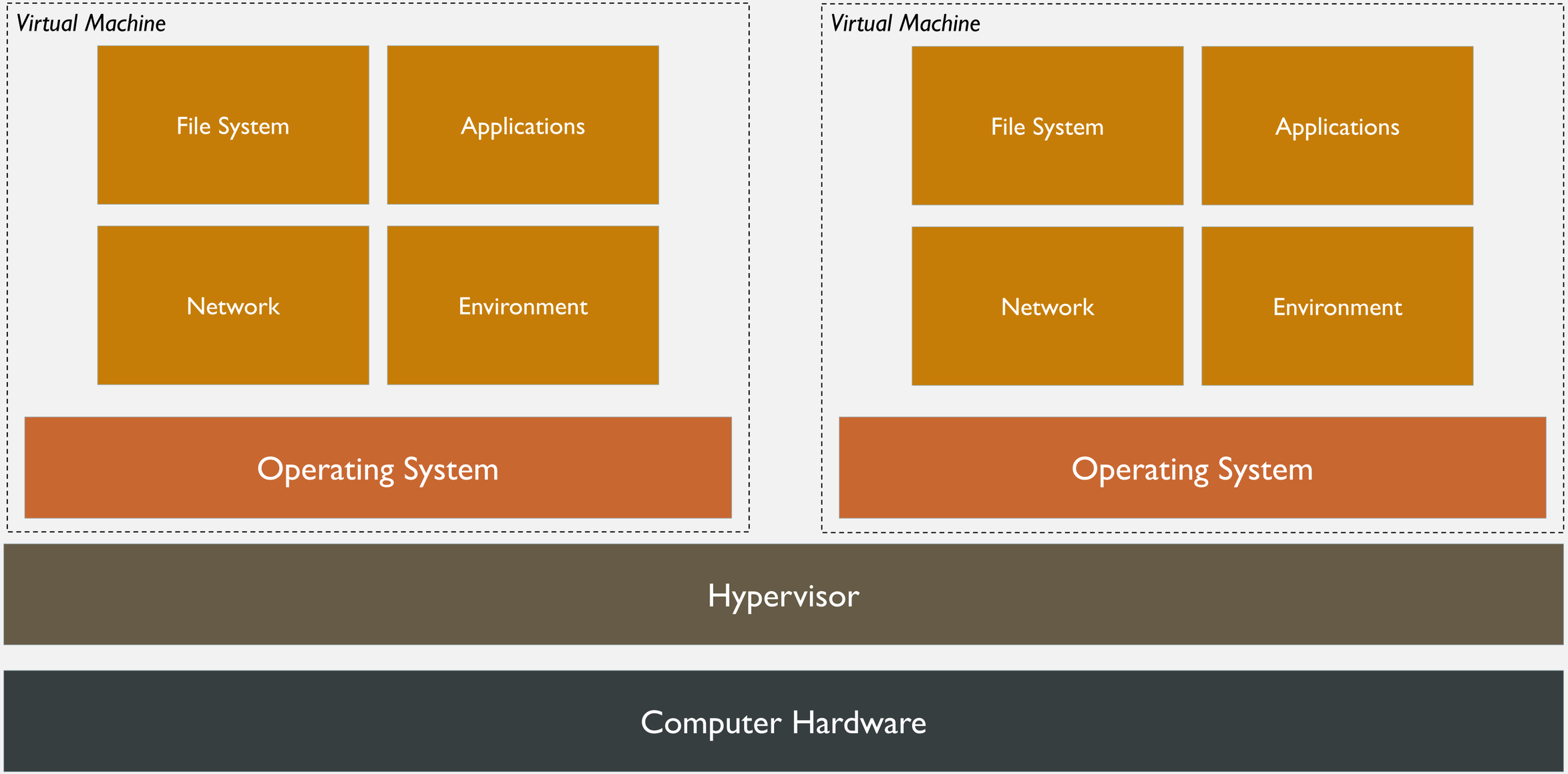


Encapsulation, isolation, portability, and control

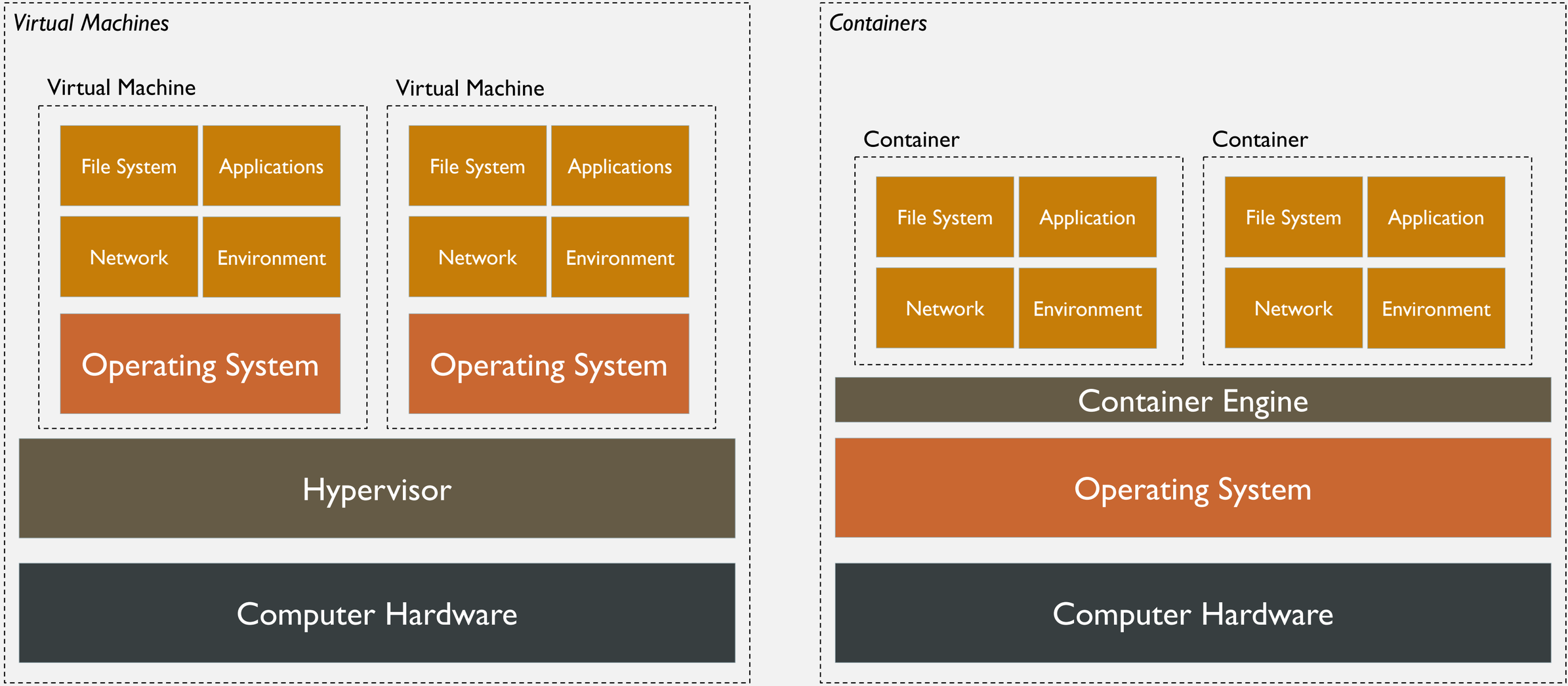
TRADITIONAL SERVER



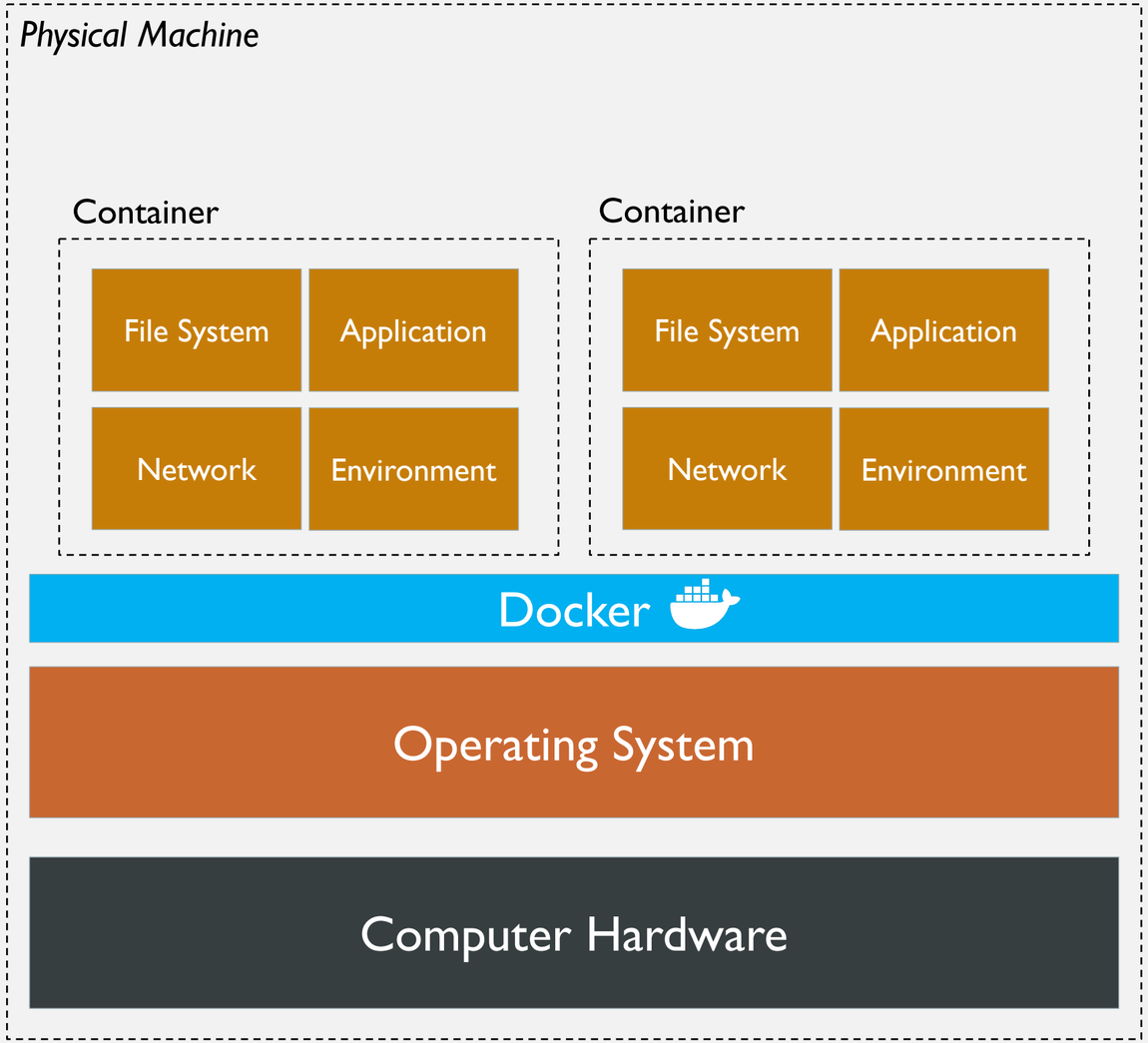
VIRTUALIZATION



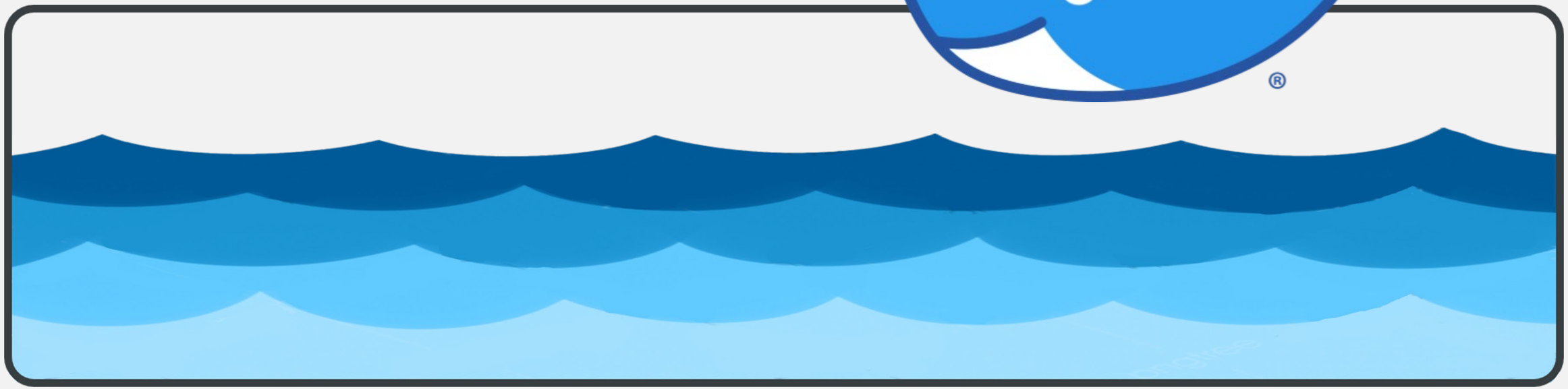
VIRTUAL vs CONTAINER



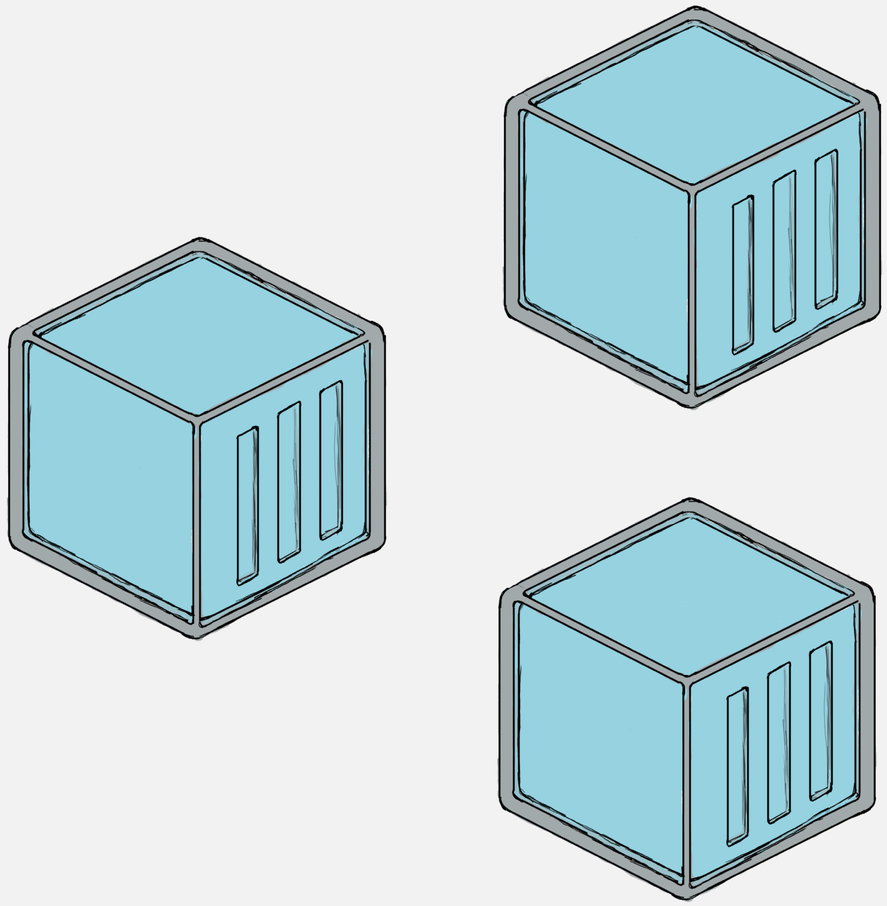
MERIDIAN

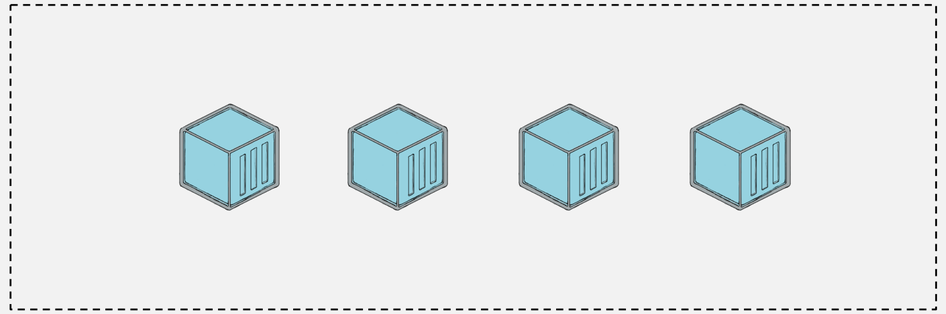

Physical Machine
Container
File System
Application
Network
Environment
Container
File System
Application
Network
Environment
Docker
Operating System
Computer Hardware

- Container engine
- Lightweight, easy to use frontend for Container management
- Cross platform

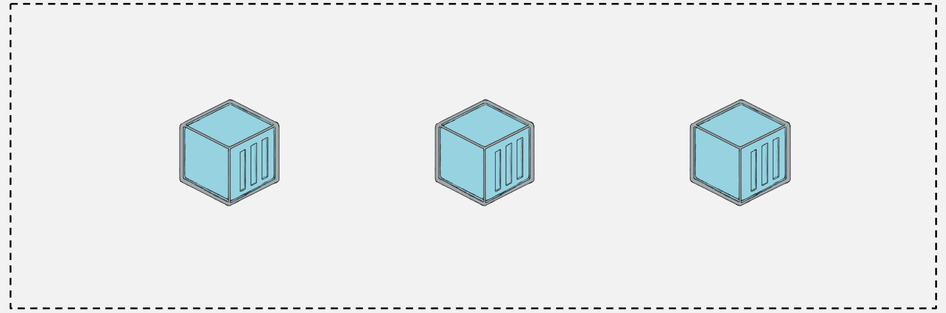


MERIDIAN
GROUND SYSTEMS



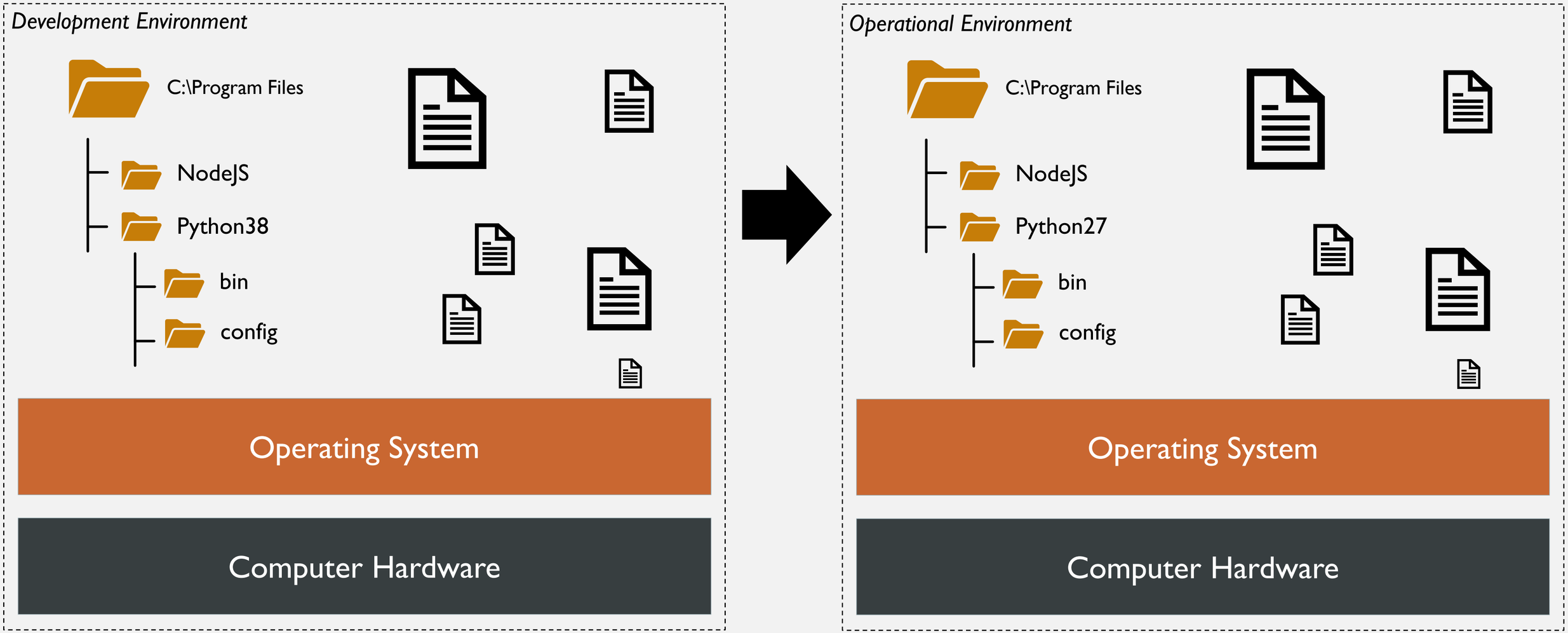


Mission A

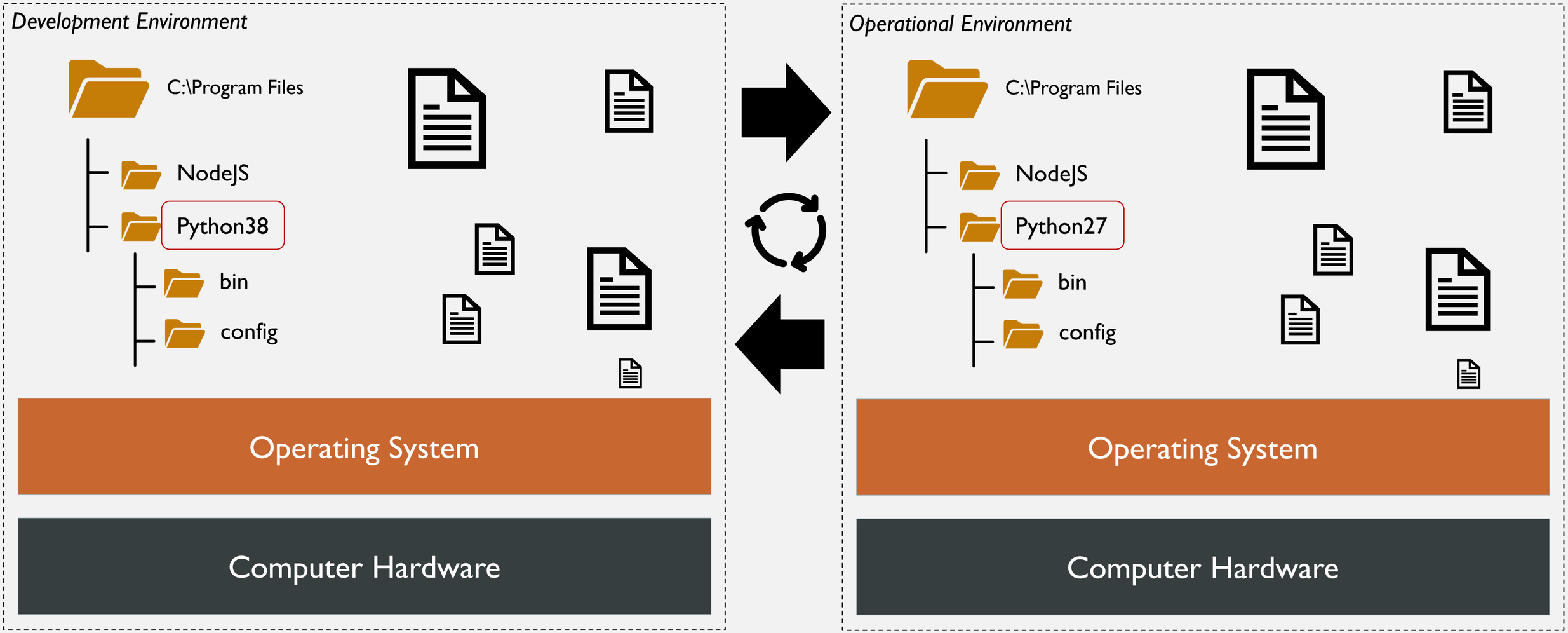


Mission B

TRADITIONAL DEPLOY

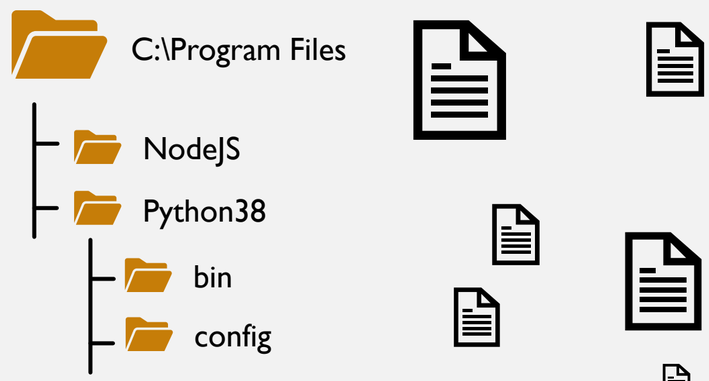


TRADITIONAL DEPLOY



DOCKER DEPLOY

Development Environment



Docker

Operating System

Computer Hardware

Operational Environment

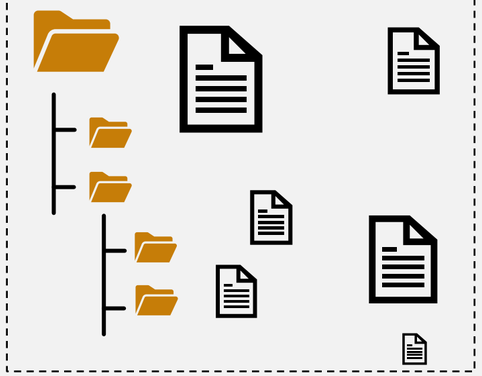
Operating System

Computer Hardware

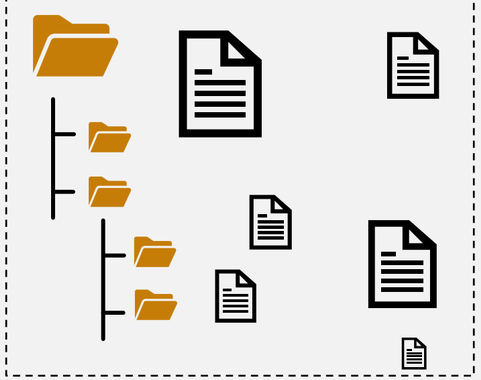
DOCKER DEPLOY

Development Environment

Container



Container



Docker

Operating System

Computer Hardware

Operational Environment

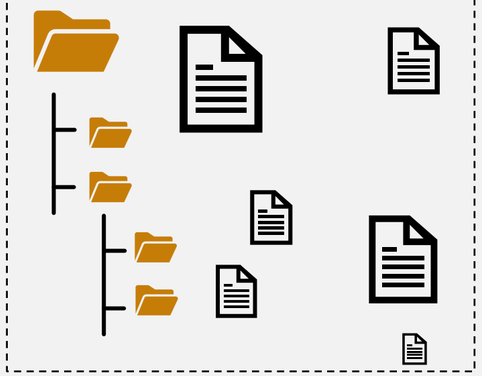
Operating System

Computer Hardware

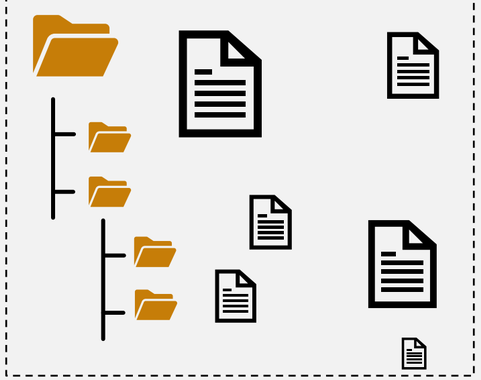
DOCKER DEPLOY

Development Environment

Container



Container



Docker

Windows

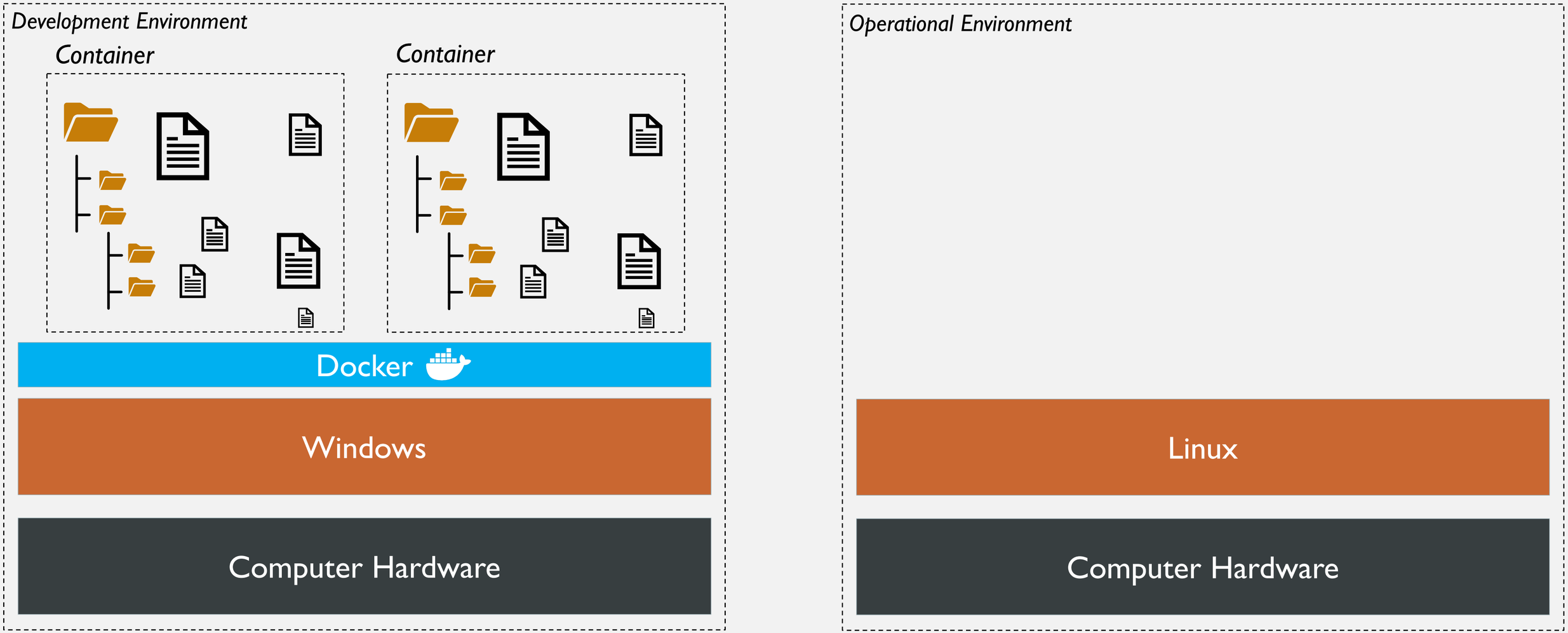
Computer Hardware

Operational Environment

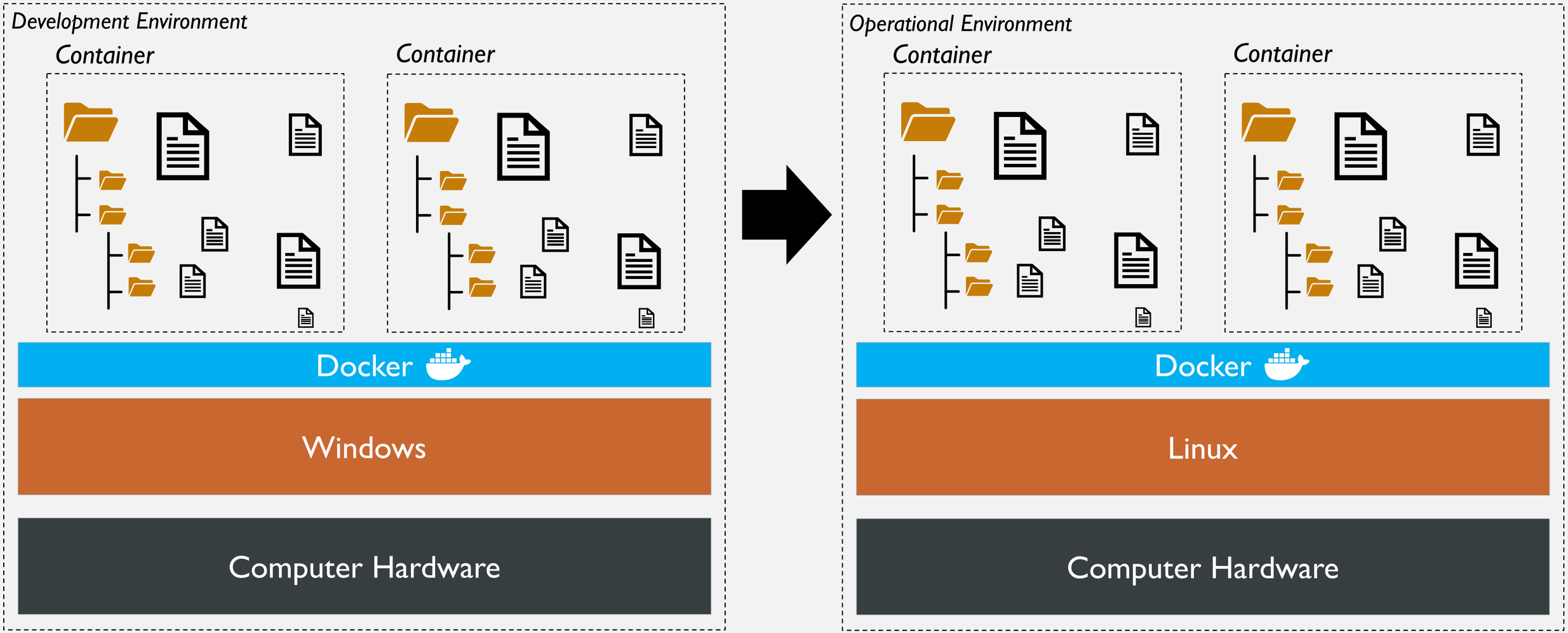
Operating System

Computer Hardware

DOCKER DEPLOY



DOCKER DEPLOY



Docker Tools

Docker Daemon	System runtime service that manages Docker objects
Docker CLI	Command-line client to interact with the Daemon
Registry	Server which stores, manages and hosts docker images

Docker Objects

Dockerfile	A text file used to generate a Docker image (think “recipe”)
Docker Image	A binary snapshot of a container
Container	A running, virtualized “instance” of the kernel

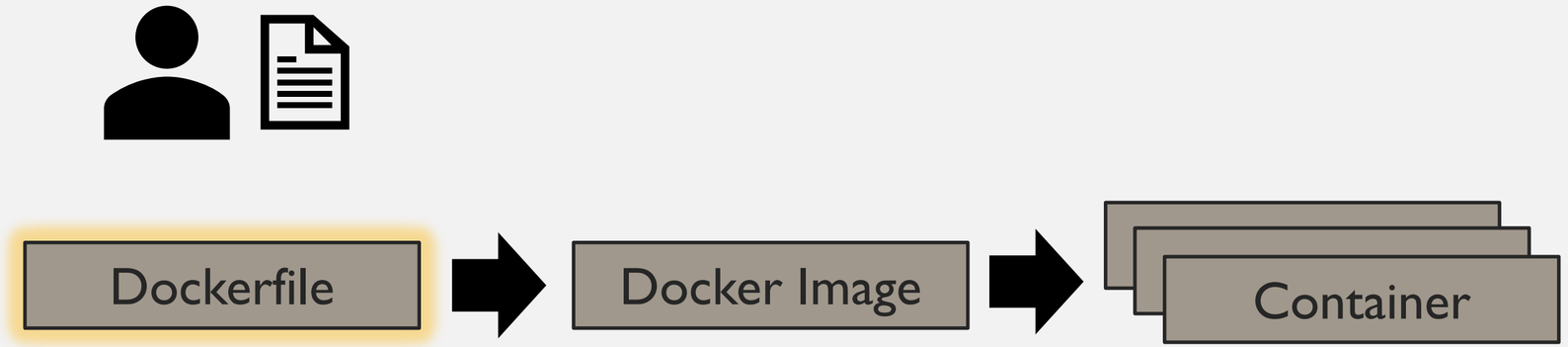
Docker Tools

Docker Daemon	System runtime service that manages Docker objects
Docker CLI	Command-line client to interact with the Daemon
Registry	Server which stores, manages and hosts docker images

Docker Objects

Dockerfile	A text file used to generate a Docker image (think “recipe”)
Docker Image	A binary snapshot of a container
Container	A running, virtualized “instance” of the kernel

1 A developer uses a Dockerfile along with the Docker CLI to generate a Docker Image

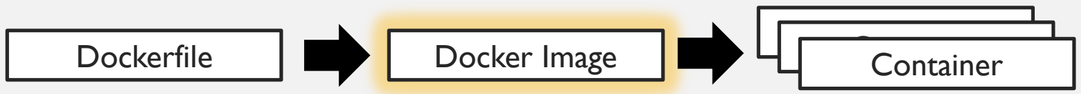


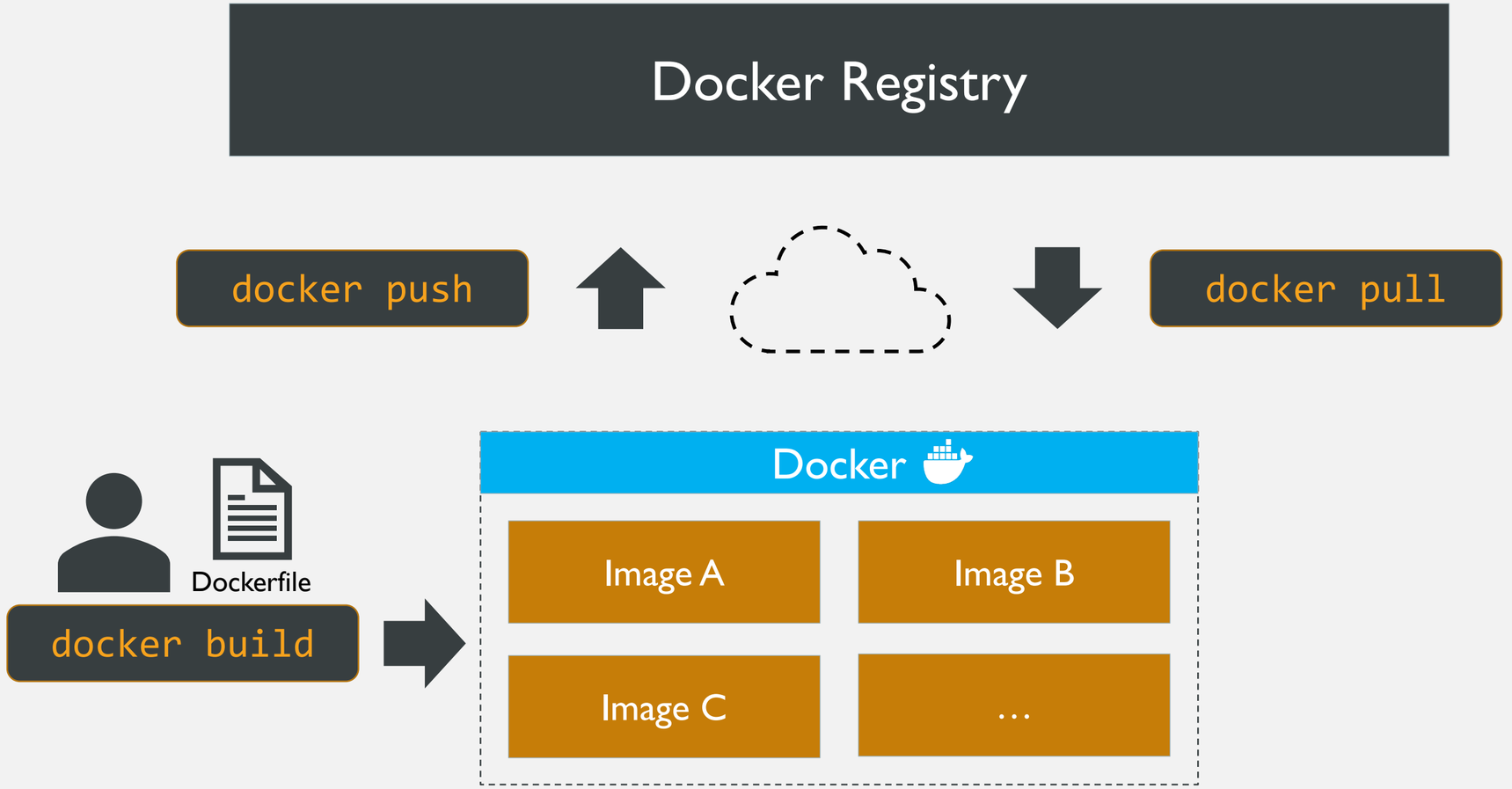

```
docker build . -t <image name>
```

2 A Docker Image is tagged with a specific version and hash which tracks each iterative generation

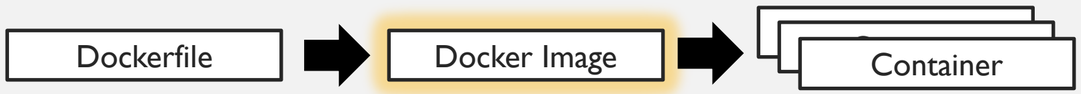

```
docker image ls
```

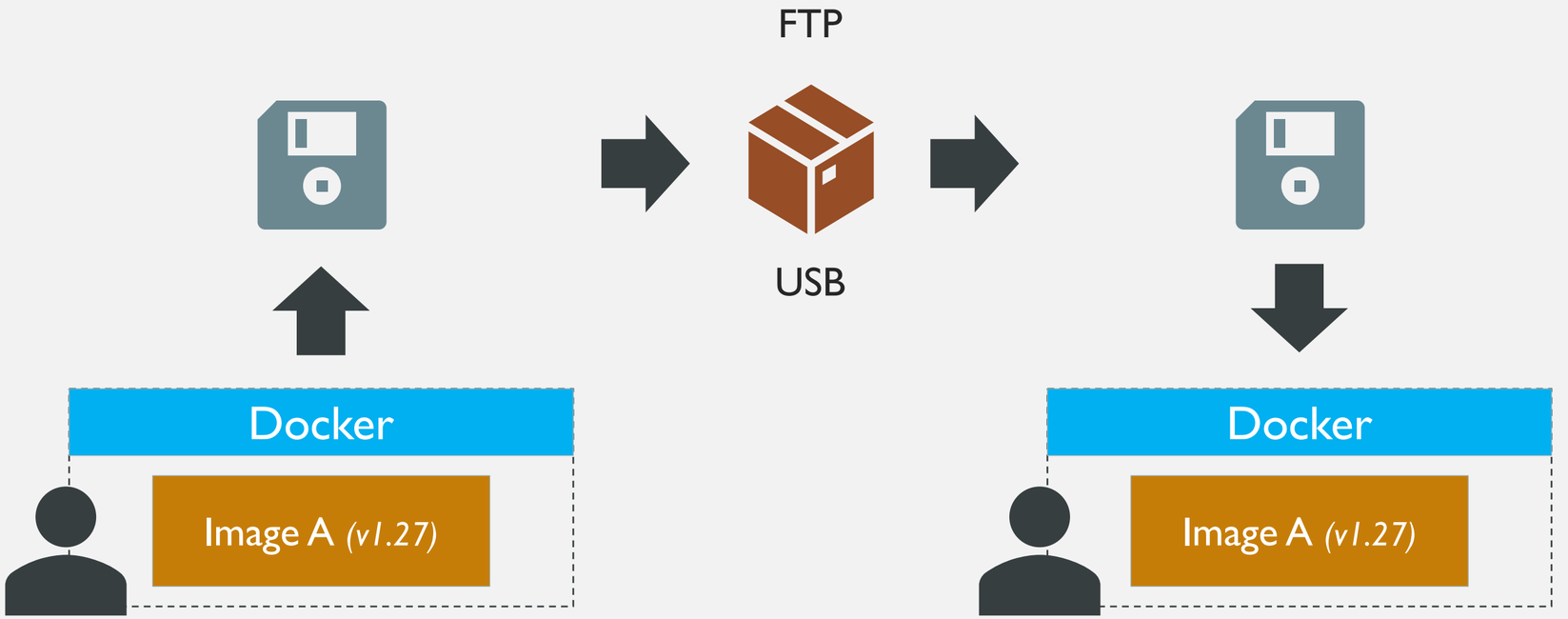
IMAGE MANAGEMENT





OFFLINE IMAGES





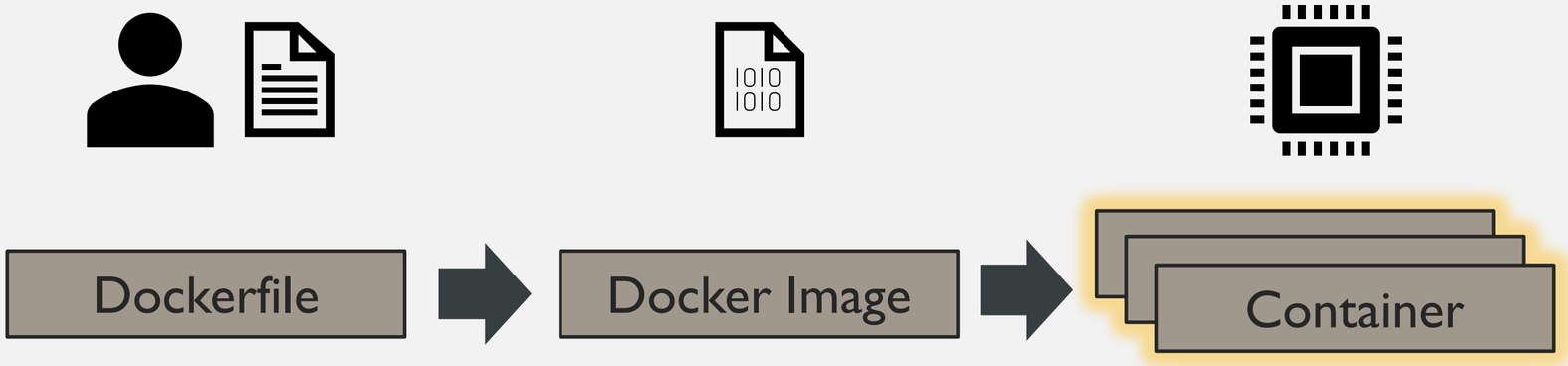

```
docker save –output <image file> <image name>
```

```
docker load -i <image file>
```

1. Developer builds a docker image and saves image to a ".tar" binary file

2. System admin loads the docker image from the ".tar" binary file

3 Using a Docker Image as a template, new Docker Containers are created which exist until the container process exits




```
docker run <image name>
```

```
docker ps
```

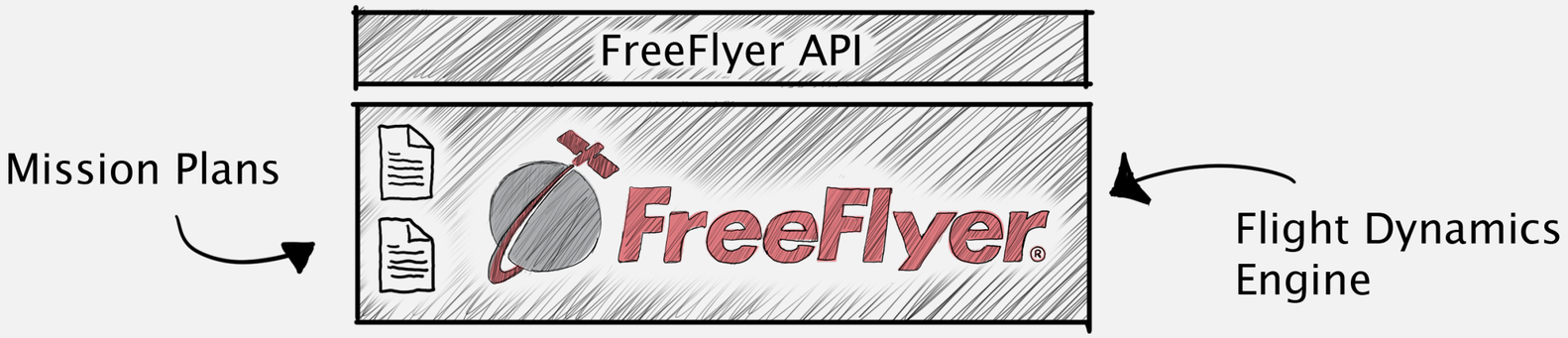
MERIDIAN

MERIDIAN

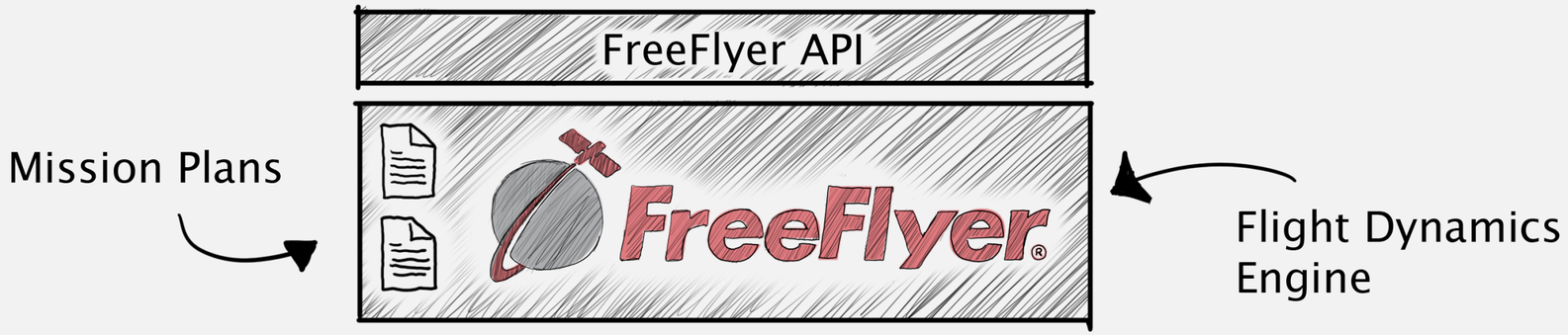
Mission Plans

Flight Dynamics Engine

MERIDIAN


FreeFlyer API
Mission Plans
FreeFlyer®
Flight Dynamics
Engine

MERIDIAN


FreeFlyer API
Mission Plans
FreeFlyer®
Flight Dynamics
Engine

MERIDIAN

Operator

Network
Access

Web
Interface

FreeFlyer API

Mission Plans

FreeFlyer®

Flight Dynamics
Engine

MERIDIAN

Operator

Network
Access

Web
Interface

Automation

REST API

Database

FreeFlyer API

Mission Plans

FreeFlyer®

Flight Dynamics
Engine

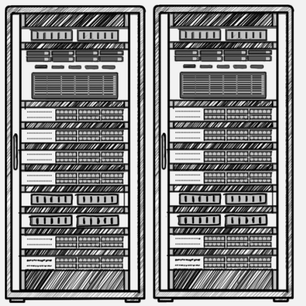
MERIDIAN

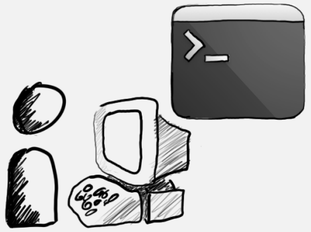
Operator

Network Access

Web Interface

External Systems





System Admin

Automation

REST API

Database

Network Access

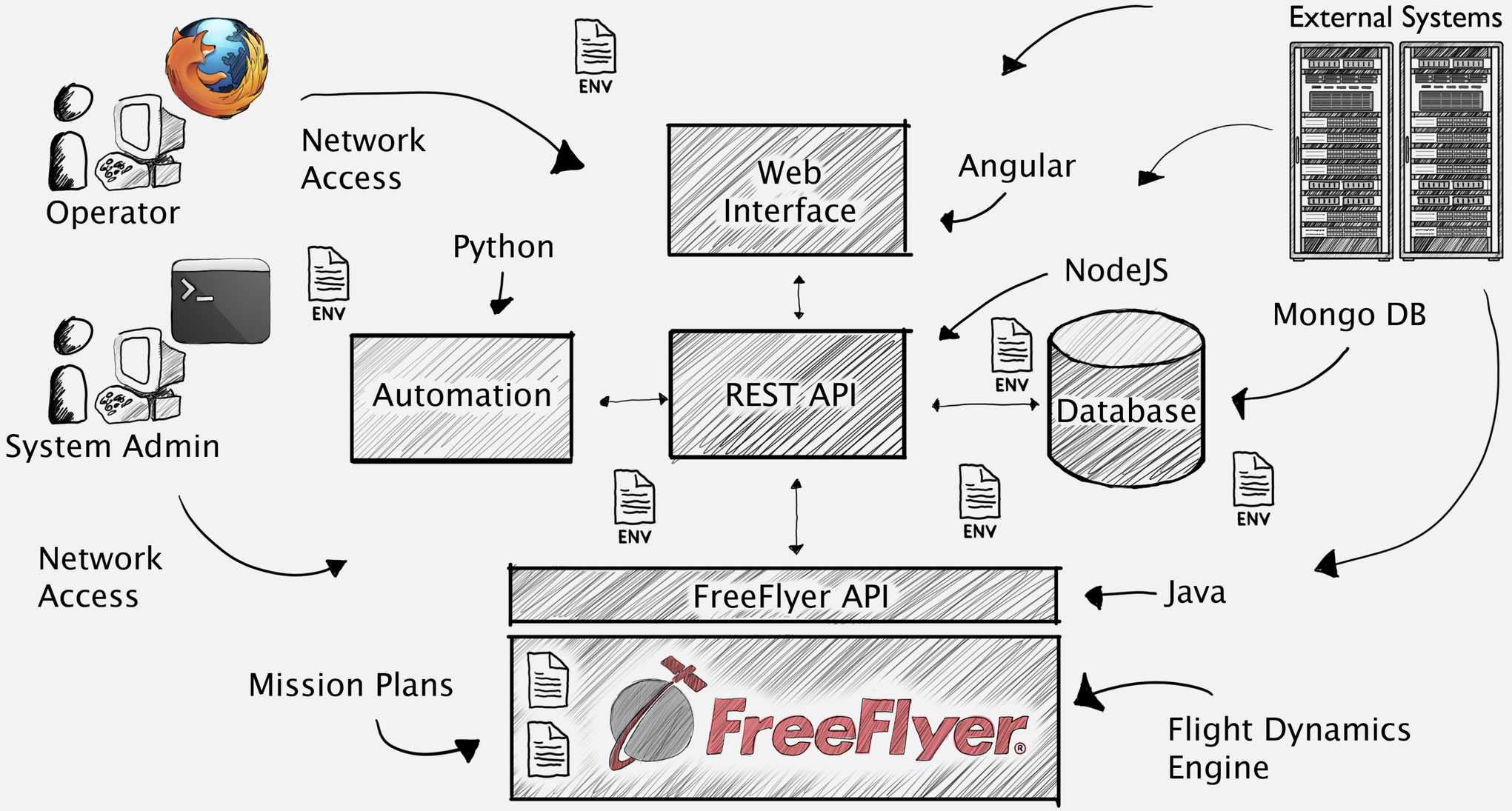
FreeFlyer API

Mission Plans

FreeFlyer®

Flight Dynamics Engine

MERIDIAN


External Systems
ENV
Network Access
Web Interface
Angular
Operator
Python
NodeJS
ENV
Automation
REST API
ENV
Database
Mongo DB
System Admin
ENV
ENV
ENV
Network Access
FreeFlyer API
Java
Mission Plans
FreeFlyer®
Flight Dynamics Engine

MERIDIAN

Operator

System Admin

Container

Web Interface
(Angular)

Container

Automation
(Python)

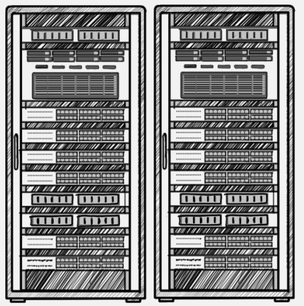
Container

REST API
(Node JS)

Container

Mongo
DB

External Systems

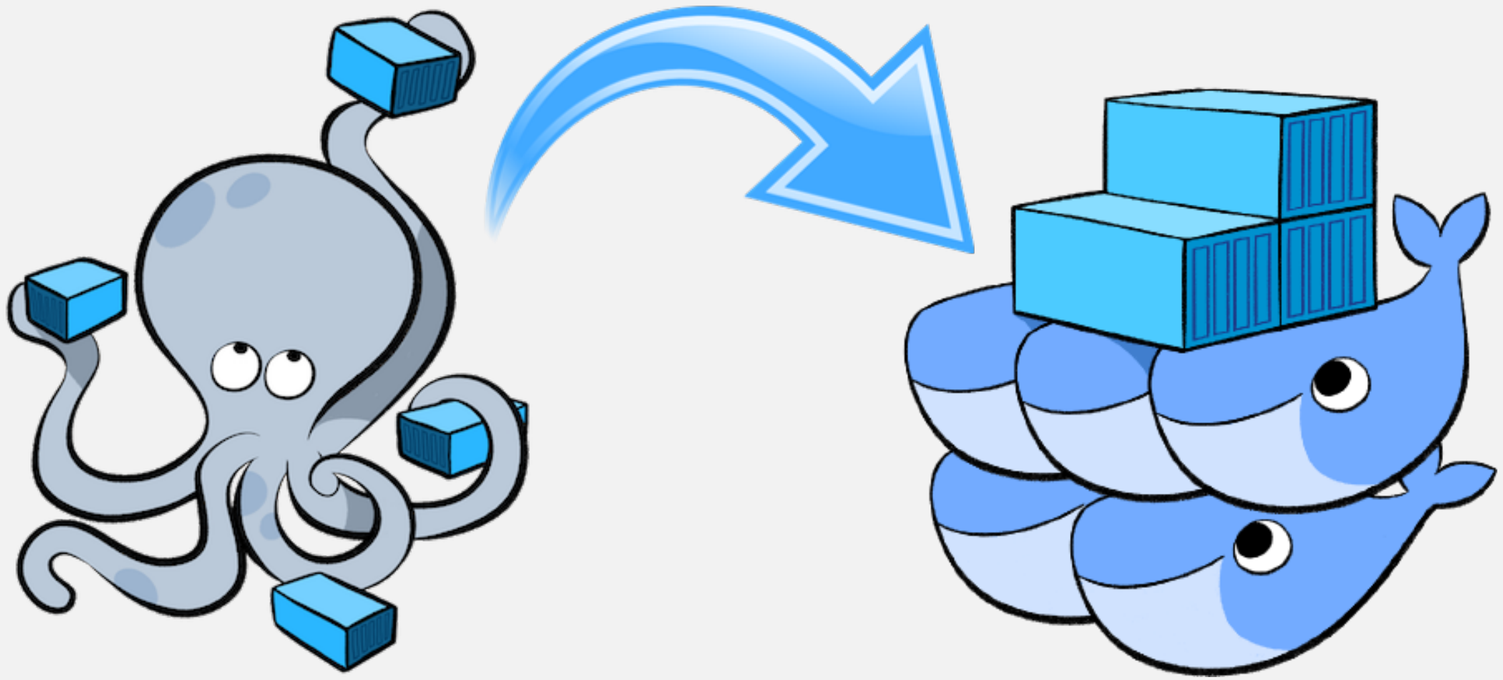


Container

FreeFlyer API (Java)

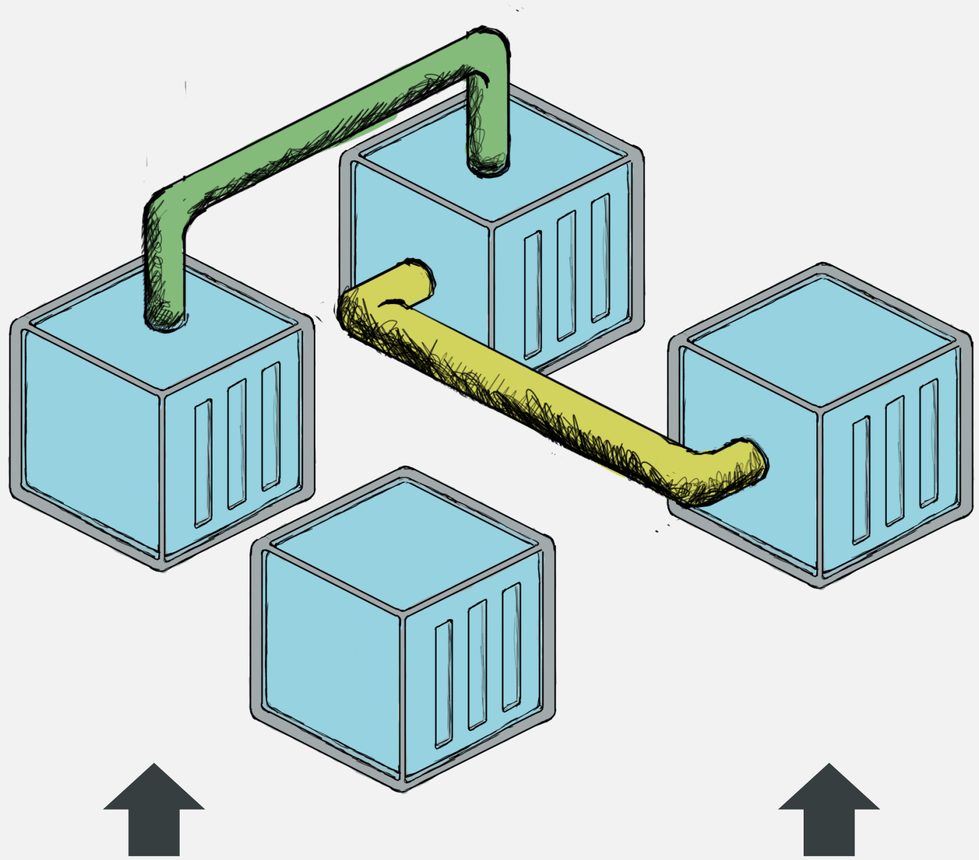
FreeFlyer

DOCKER COMPOSE



DOCKER COMPOSE

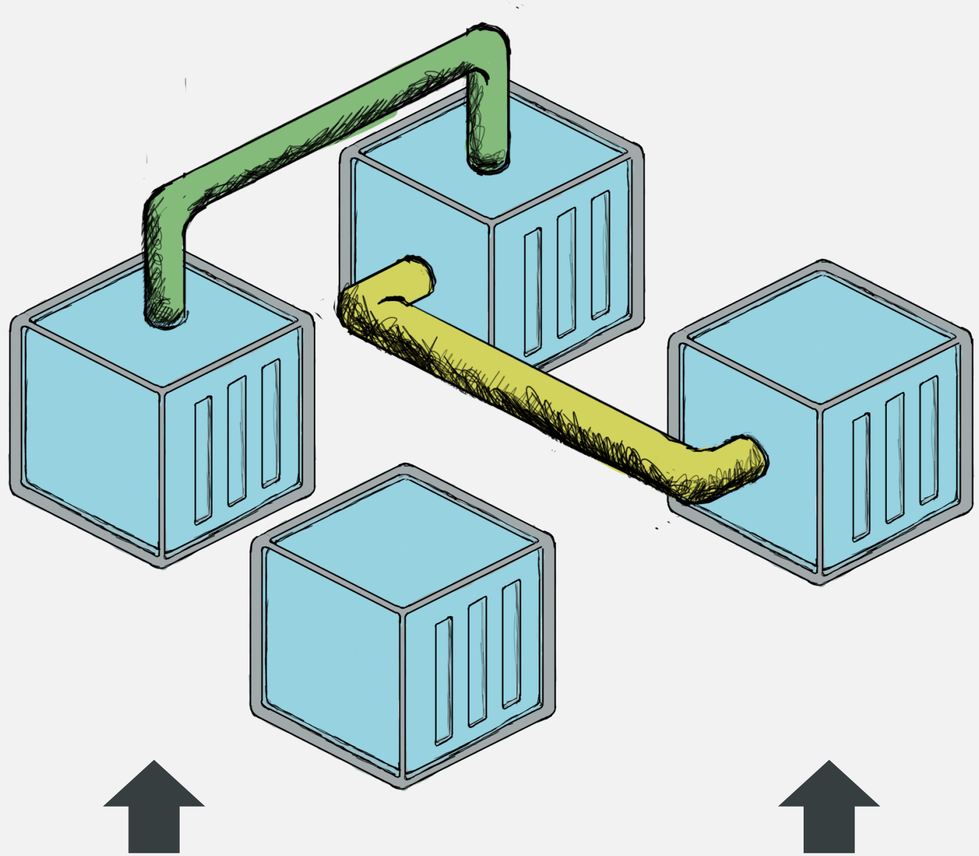
```
meridian-ui:
  image: meridian/web-interface:1.7
  ports:
    - 443:443
  volumes:
    - opt/meridian/config:/app/config
  depends_on:
    - rest-api

meridian-rest:
  image: meridian/rest-api:3.6
  environment:
    - NODE_ENV=production
  ports:
    -  8080:3000
  extra_hosts:
    - external-system:10.0.10.198
  links:
    - database
  depends_on:
    - database

meridian-db:
  image: meridian/database:2.1
  ports:
    - 27017:27107
  environment:
    - JAVA_HOME
```



```
docker-compose up –f <docker compose>.yml
```

DOCKER COMPOSE

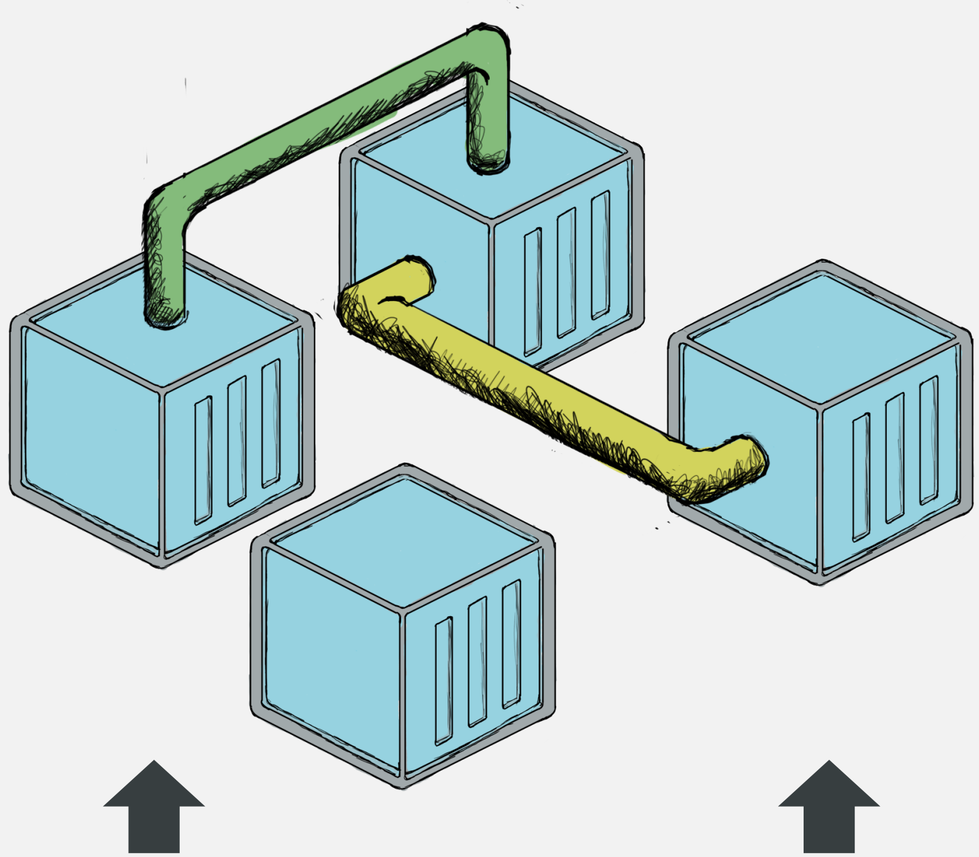
```
meridian-ui:
  image: meridian/web-interface:1.7
  ports:
    - 443:443
  volumes:
    - opt/meridian/config:/app/config
  depends_on:
    - rest-api

meridian-rest:
  image: meridian/rest-api:3.6
  environment:
    - NODE_ENV=production
  ports:
    -  8080:3000
  extra_hosts:
    - external-system:10.0.10.198
  links:
    - database
  depends_on:
    - database

meridian-db:
  image: meridian/database:2.1
  ports:
    - 27017:27107
  environment:
    - JAVA_HOME
```



```
docker-compose up –f <docker compose>.yml
```

DOCKER COMPOSE

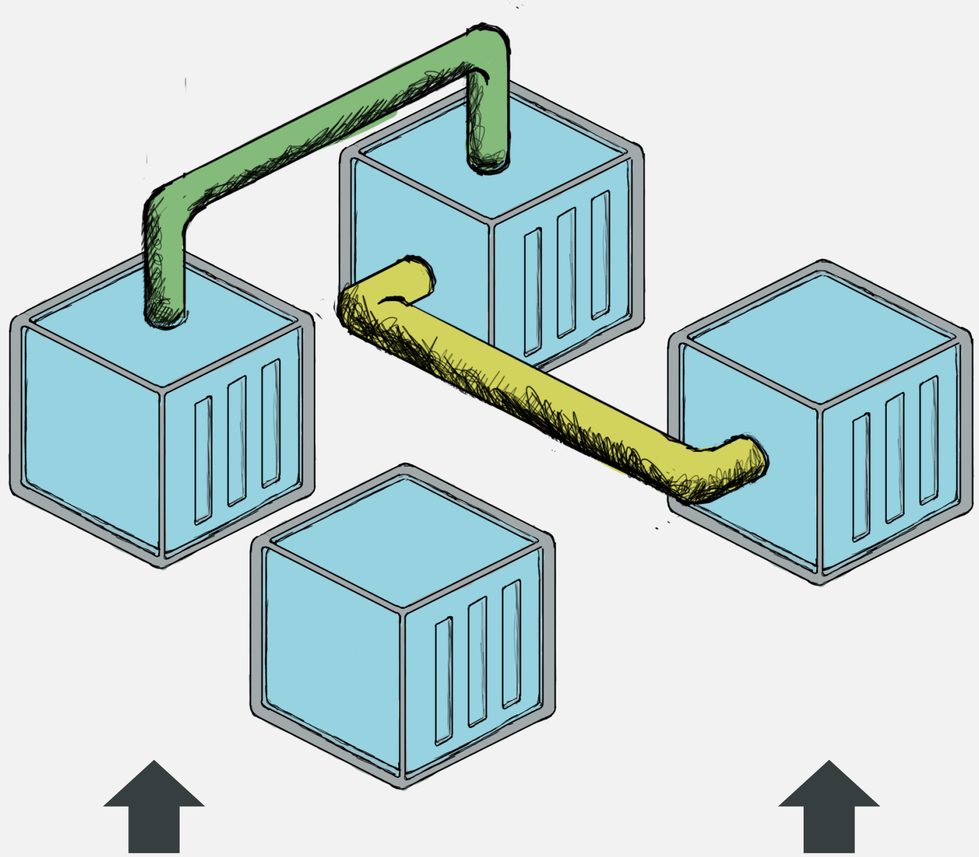
```
meridian-ui:
  image: meridian/web-interface:1.7
  ports:
    - 443:443
  volumes:
    - opt/meridian/config:/app/config
  depends_on:
    - rest-api

meridian-rest:
  image: meridian/rest-api:3.6
  environment:
    - NODE_ENV=production
  ports:
    -  8080:3000
  extra_hosts:
    - external-system:10.0.10.198
  links:
    - database
  depends_on:
    - database

meridian-db:
  image: meridian/database:2.1
  ports:
    - 27017:27107
  environment:
    - JAVA_HOME
```



```
docker-compose up -f <docker compose>.yml
```

DOCKER COMPOSE

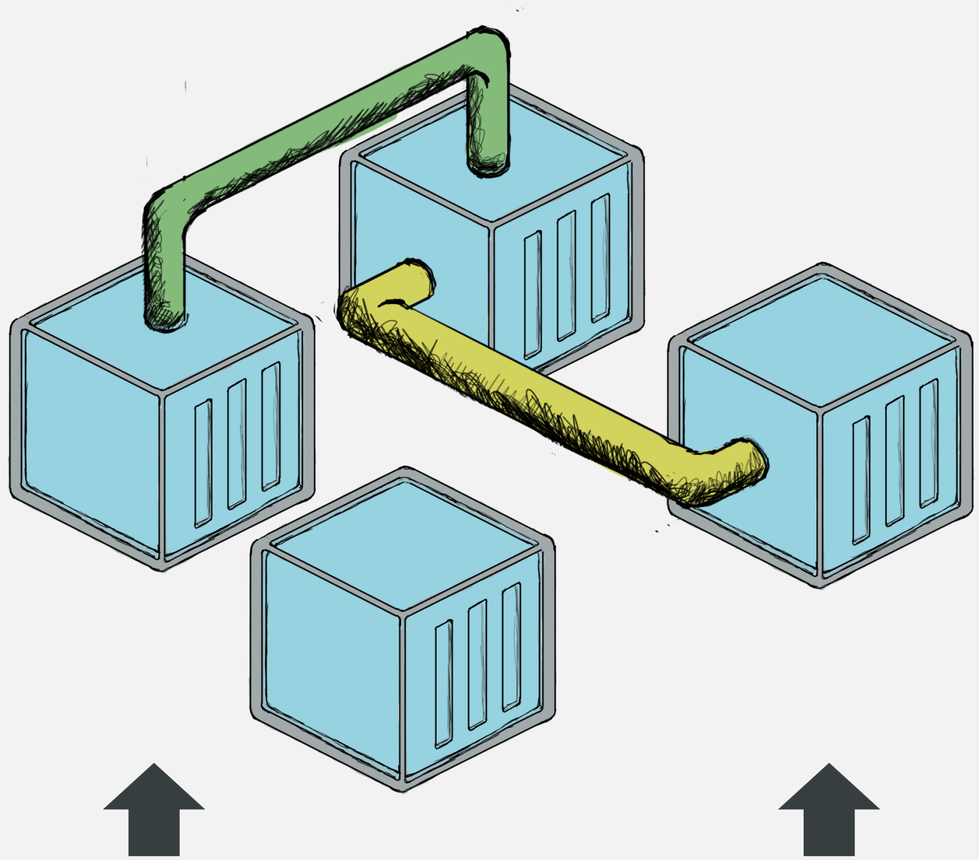
```
meridian-ui:
  image: meridian/web-interface:1.7
  ports:
    - 443:443
  volumes:
    - opt/meridian/config:/app/config
  depends_on:
    - rest-api

meridian-rest:
  image: meridian/rest-api:3.6
  environment:
    - NODE_ENV=production
  ports:
    -  8080:3000
  extra_hosts:
    - external-system:10.0.10.198
  links:
    - database
  depends_on:
    - database

meridian-db:
  image: meridian/database:2.1
  ports:
    - 27017:27107
  environment:
    - JAVA_HOME
```



```
docker-compose up -f <docker compose>.yml
```

DOCKER COMPOSE

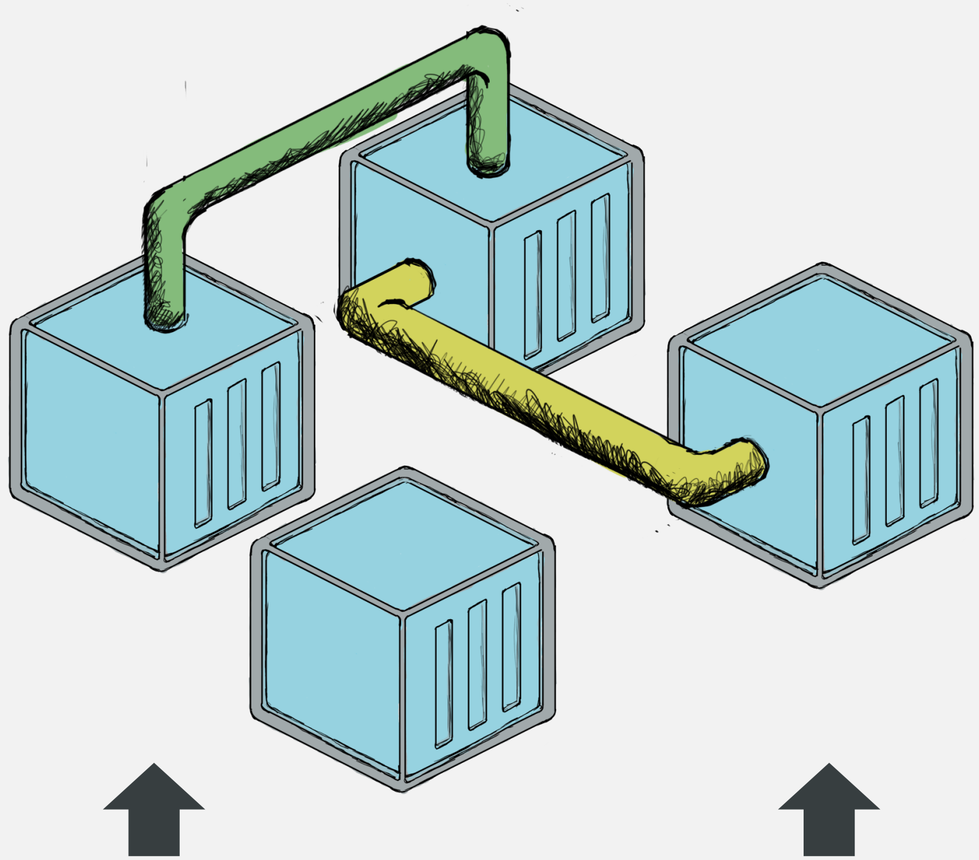
```
meridian-ui:
  image: meridian/web-interface:1.7
  ports:
    - 443:443
  volumes:
    - opt/meridian/config:/app/config
  depends_on:
    - rest-api

meridian-rest:
  image: meridian/rest-api:3.6
  environment:
    - NODE_ENV=production
  ports:
    -  8080:3000
  extra_hosts:
    - external-system:10.0.10.198
  links:
    - database
  depends_on:
    - database

meridian-db:
  image: meridian/database:2.1
  ports:
    - 27017:27107
  environment:
    - JAVA_HOME
```



```
docker-compose up -f <docker compose>.yml
```

DOCKER COMPOSE

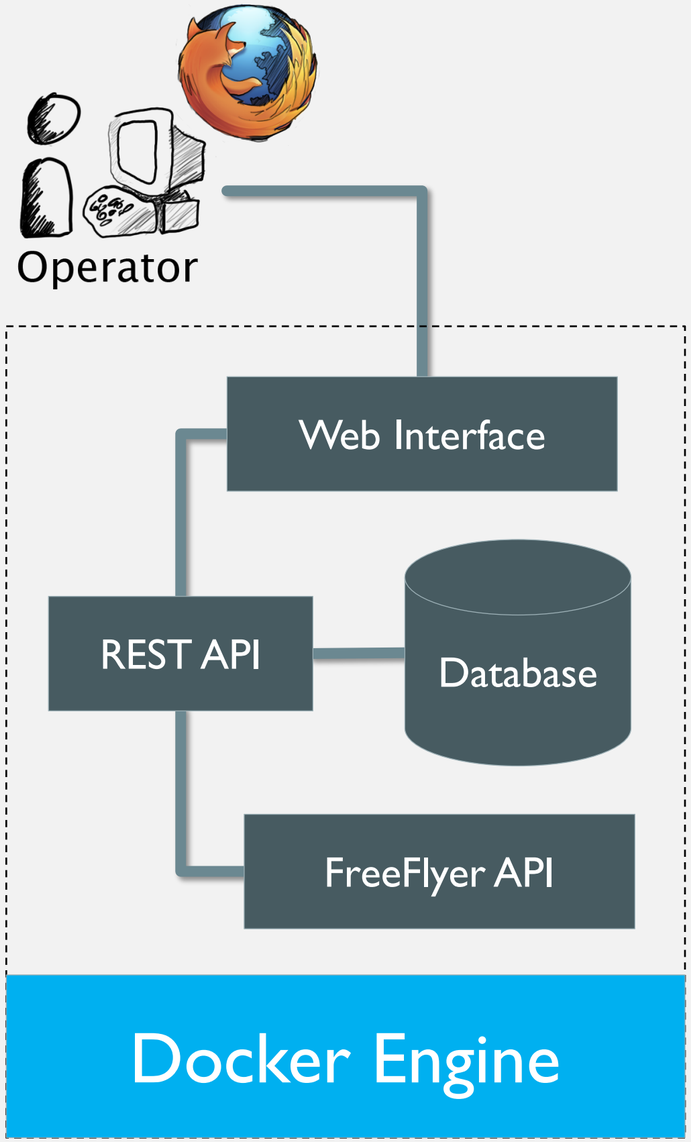
```
meridian-ui:
  image: meridian/web-interface:1.7
  ports:
    - 443:443
  volumes:
    - opt/meridian/config:/app/config
  depends_on:
    - rest-api

meridian-rest:
  image: meridian/rest-api:3.6
  environment:
    - NODE_ENV=production
  ports:
    -  8080:3000
  extra_hosts:
    - external-system:10.0.10.198
  links:
    - database
  depends_on:
    - database

meridian-db:
  image: meridian/database:2.1
  ports:
    - 27017:27107
  environment:
    - JAVA_HOME
```



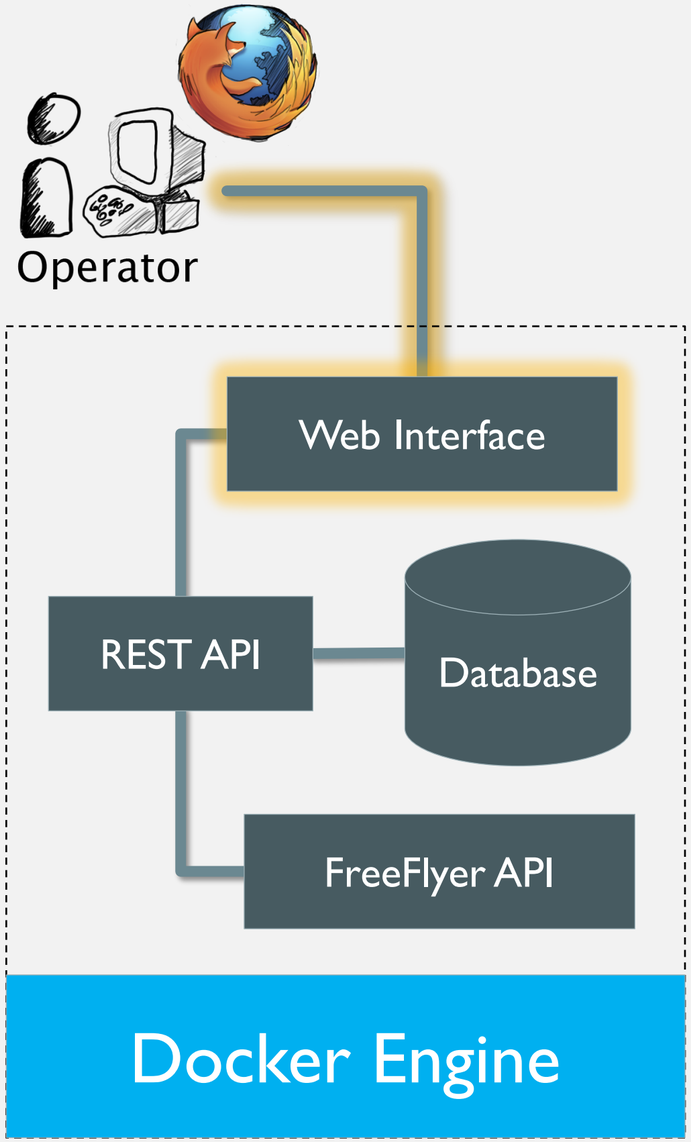
```
docker-compose up –f <docker compose>.yml
```

DOCKER COMPOSE

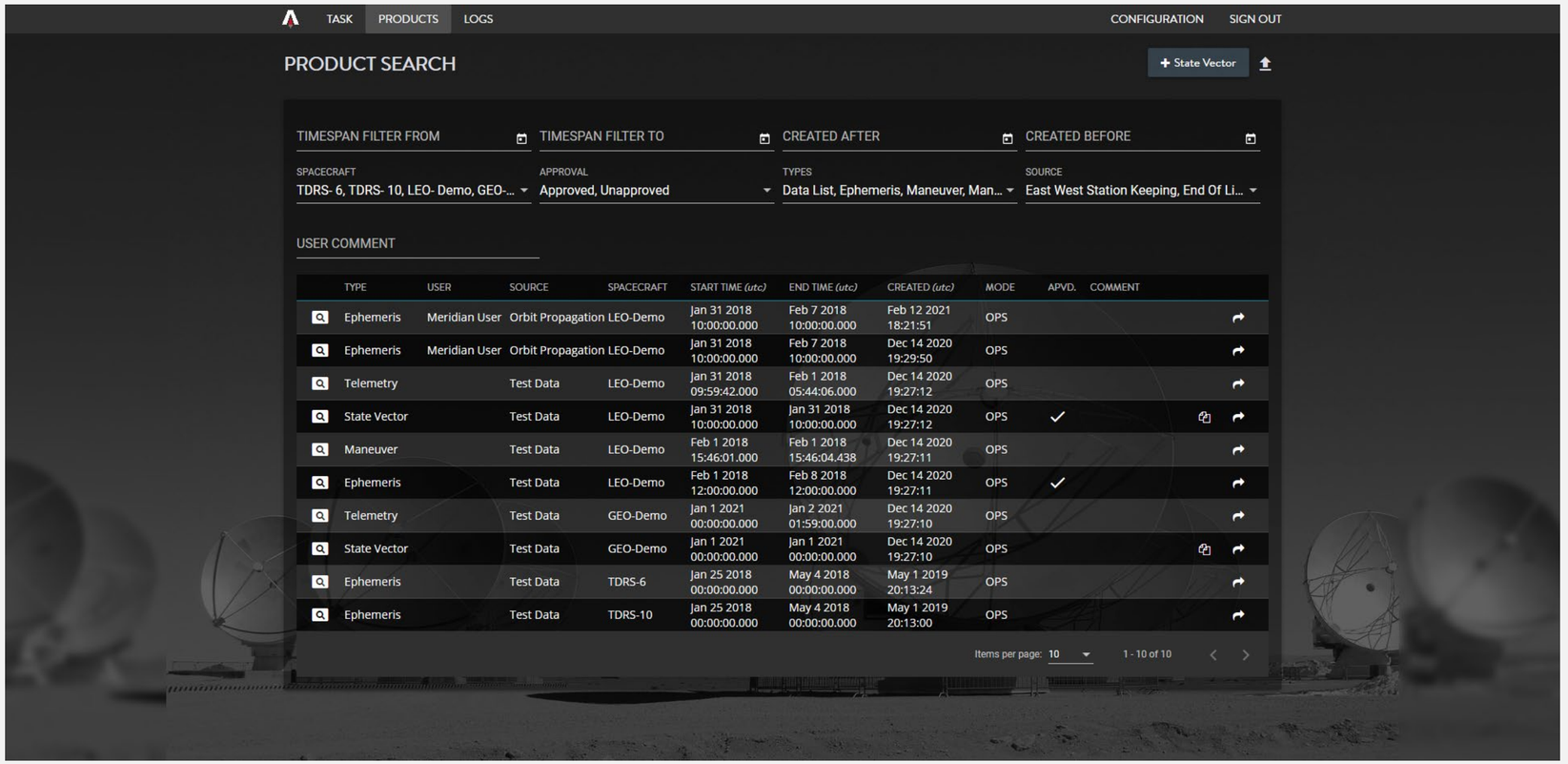
```
meridian@mdemo-sales-linux:/opt/meridian/config/docker
[operator@meridian-server]$ ls
docker-compose.yml
[operator@meridian-server]$ docker-compose -f docker-compose.yml up -d
Creating network "docker_default" with the default driver
Creating meridian-db ... done
Creating meridian-rest ... done
Creating meridian-ui   ... done
[operator@meridian-server]$
```



DOCKER COMPOSE



DOCKER COMPOSE



Operator

Web Interface

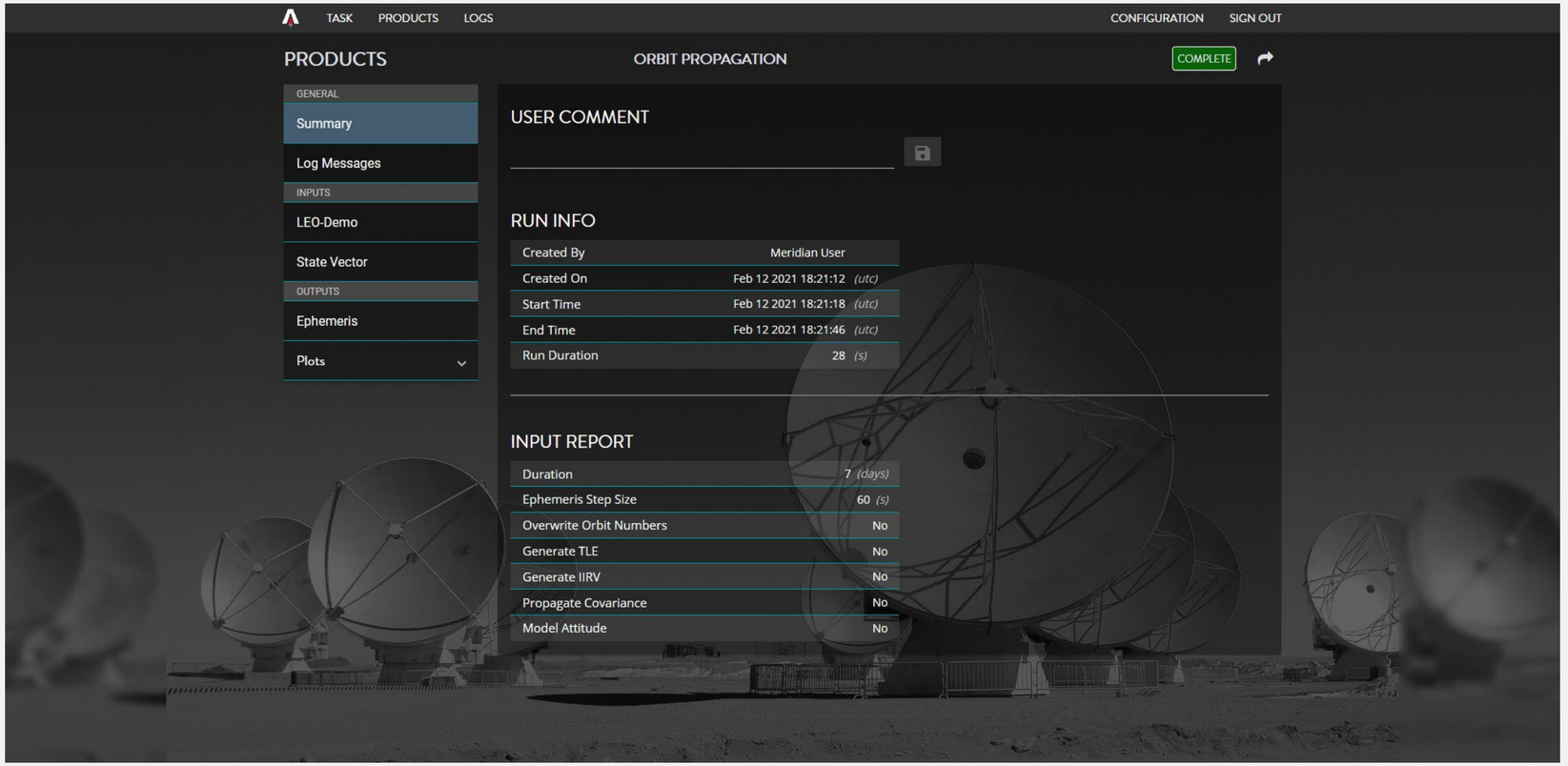
REST API

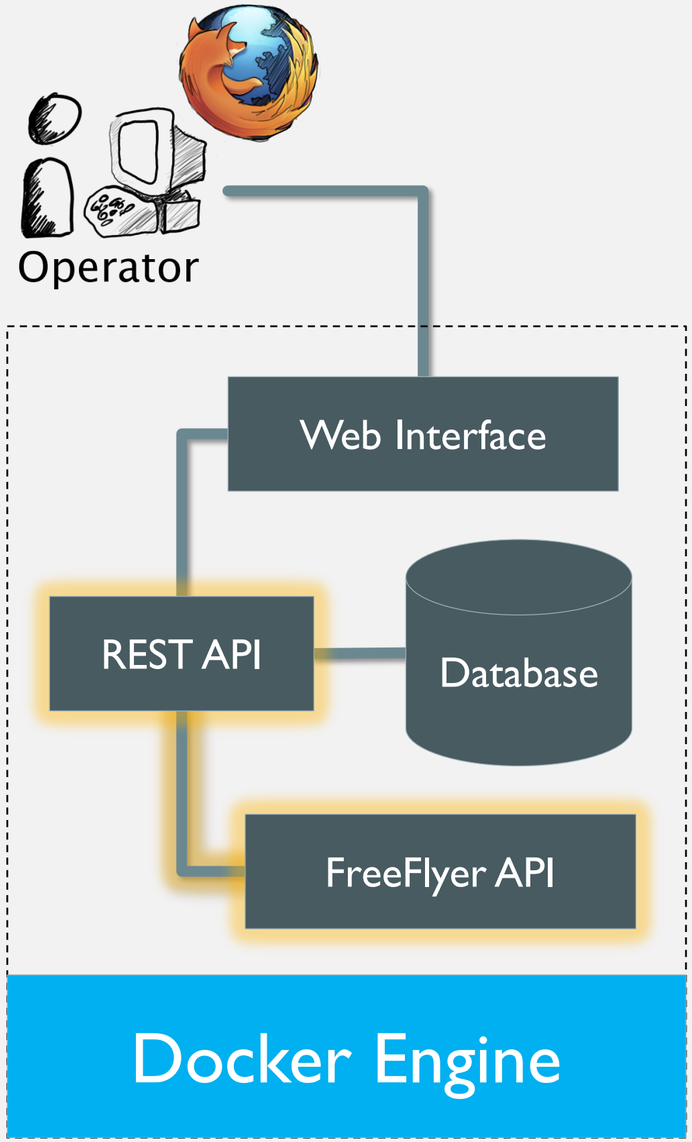
Database

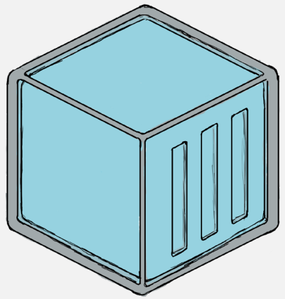
FreeFlyer API

Docker Engine

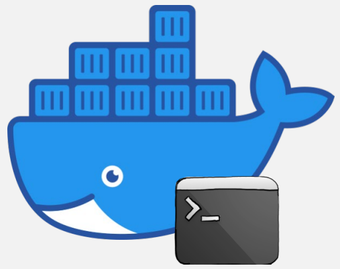
DOCKER COMPOSE



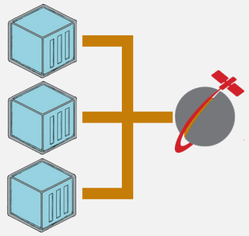




Containers



Docker



Scott Lowe

Architecture Lead

Email: scott.lowe@ai-solutions.com

Website: ai-solutions.com/meridian

a.i. solutions