SWIFT Developer Series

Containers and Orchestration

June 21, 2022

Introductions

Jeff Stein
jstein@mitre.org
The MITRE Corporation
Principal Software Engineer

Joey Menzenski
jmenzenski@mitre.org
The MITRE Corporation
Lead Software Engineer

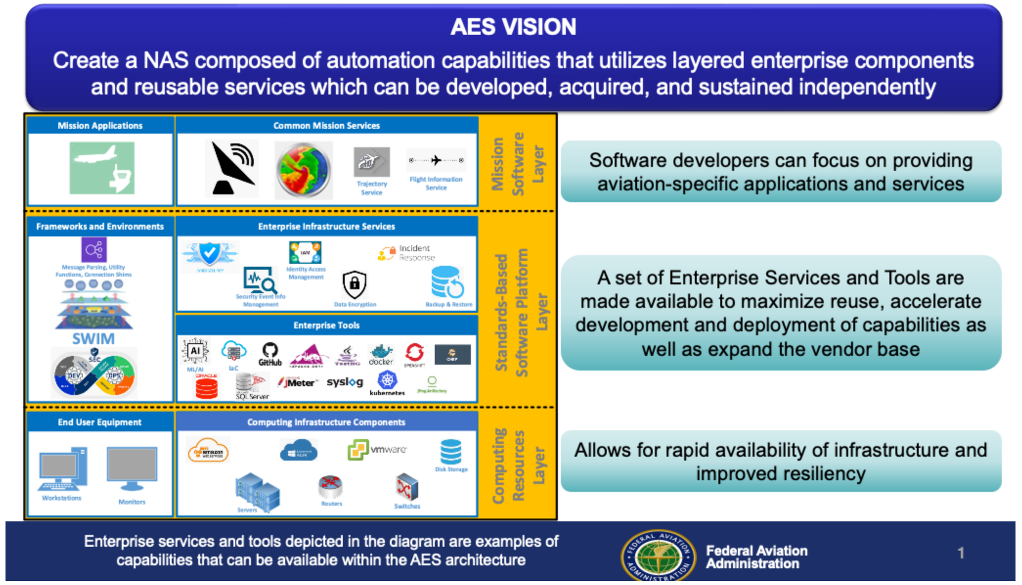
Kevin Long
klong@mitre.org
The MITRE Corporation
Principal Software Engineer

Recap of the SWIFT Developer Series

SWIFT Developer Series: Objectives

- **Review the basics of connecting and consuming SWIM data**

- **After the series, participants will:**
  - Have a deeper understanding of integrating SWIM data and be empowered to develop solutions to address a problem space
  - Understand how the Automation Evolution Strategy will enable iterative development and common services to meet the needs of the users (internal and external)
  - Appreciate how capabilities can be collaboratively built and evolve over time


AES VISION

Create a NAS composed of automation capabilities that utilizes layered enterprise components and reusable services which can be developed, acquired, and sustained independently

Software developers can focus on providing aviation-specific applications and services

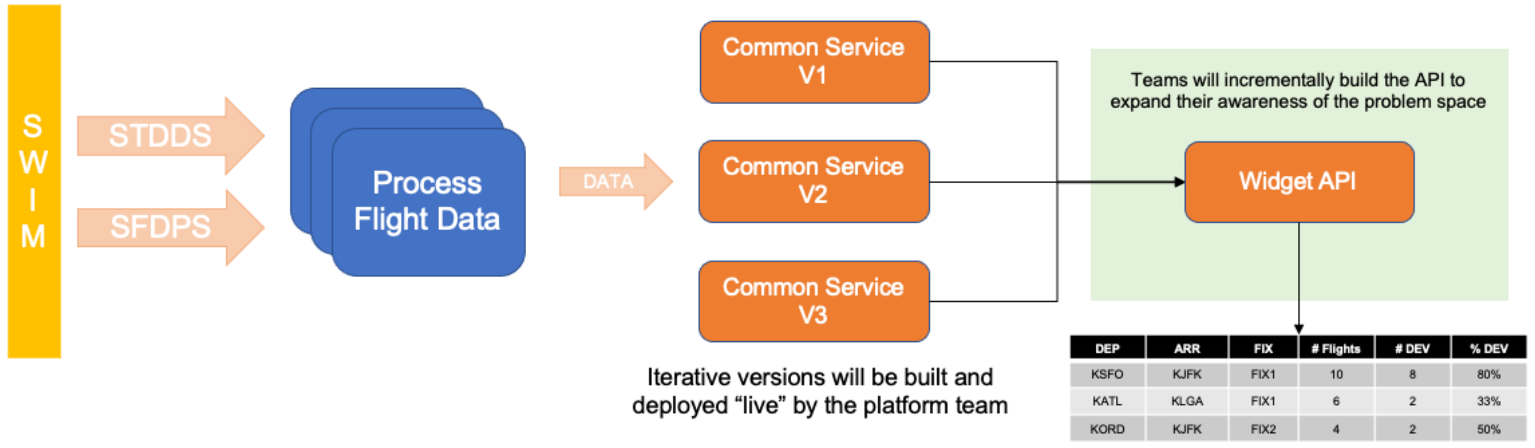
A set of Enterprise Services and Tools are made available to maximize reuse, accelerate development and deployment of capabilities as well as expand the vendor base

Allows for rapid availability of infrastructure and improved resiliency

Enterprise services and tools depicted in the diagram are examples of capabilities that can be available within the AES architecture

Mission Applications
Common Mission Services
Trajectory Service
Flight Information Service
Mission Software Layer
Frameworks and Environments
Message Parsing, Utility Functions, Connection Shims
SWIM
Enterprise Infrastructure Services
Incident Response
Identity Access Management
Security Event Info Management
Data Encryption
Backup & Restore
Enterprise Tools
ML/AI
IaC
GitHub
docker
JMeter
syslog
kubernetes
SQL Server
Standards-Based Software Platform Layer
End User Equipment
Workstations
Monitors
Computing Infrastructure Components
vmware
Disk Storage
Servers
Routers
Switches
Computing Resources Layer
Federal Aviation Administration
1

Developer Workshop Overview


S W I M
STDDS
SFDPS
Process Flight Data
DATA
Common Service V1
Common Service V2
Common Service V3
Teams will incrementally build the API to expand their awareness of the problem space
Widget API
Iterative versions will be built and deployed "live" by the platform team

DEP	ARR	FIX	# Flights	# DEV	% DEV
KSFO	KJFK	FIX1	10	8	80%
KATL	KLGA	FIX1	6	2	33%
KORD	KJFK	FIX2	4	2	50%

- Participants will create an Application Programming Interface (API) that will drive an analytics chart
  - Consume data from a common data service
  - Process the data to make it available for table using a known schema
  - API will be deployed via pipeline
- As the exercise progresses – new versions of common service will become available with more extensive data.
  - Participants will update their applications accordingly
- Participants will have some level of language choice
  - Java, Python, JavaScript

Preparing for the In-Person Developer Workshop

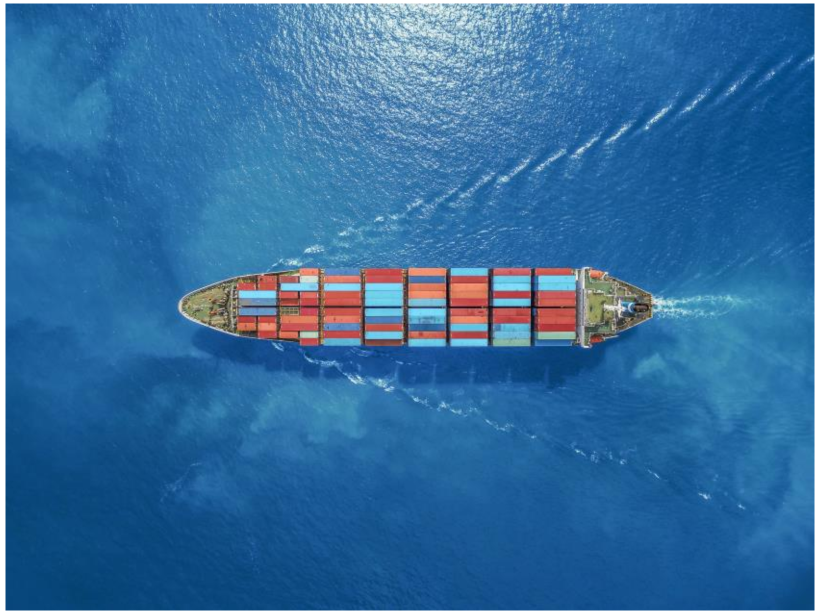
- **Webinar 1**
  - Experience building and running containerized software
  - Familiarity with deploying containerized software
- **Webinar 2**
  - Experience connecting to SWIM and consuming data
  - Some SWIM data knowledge
- **Webinar 3**
  - Background on the operational problem space (Trajectory Deviation Study)

All Things Containers

Overview

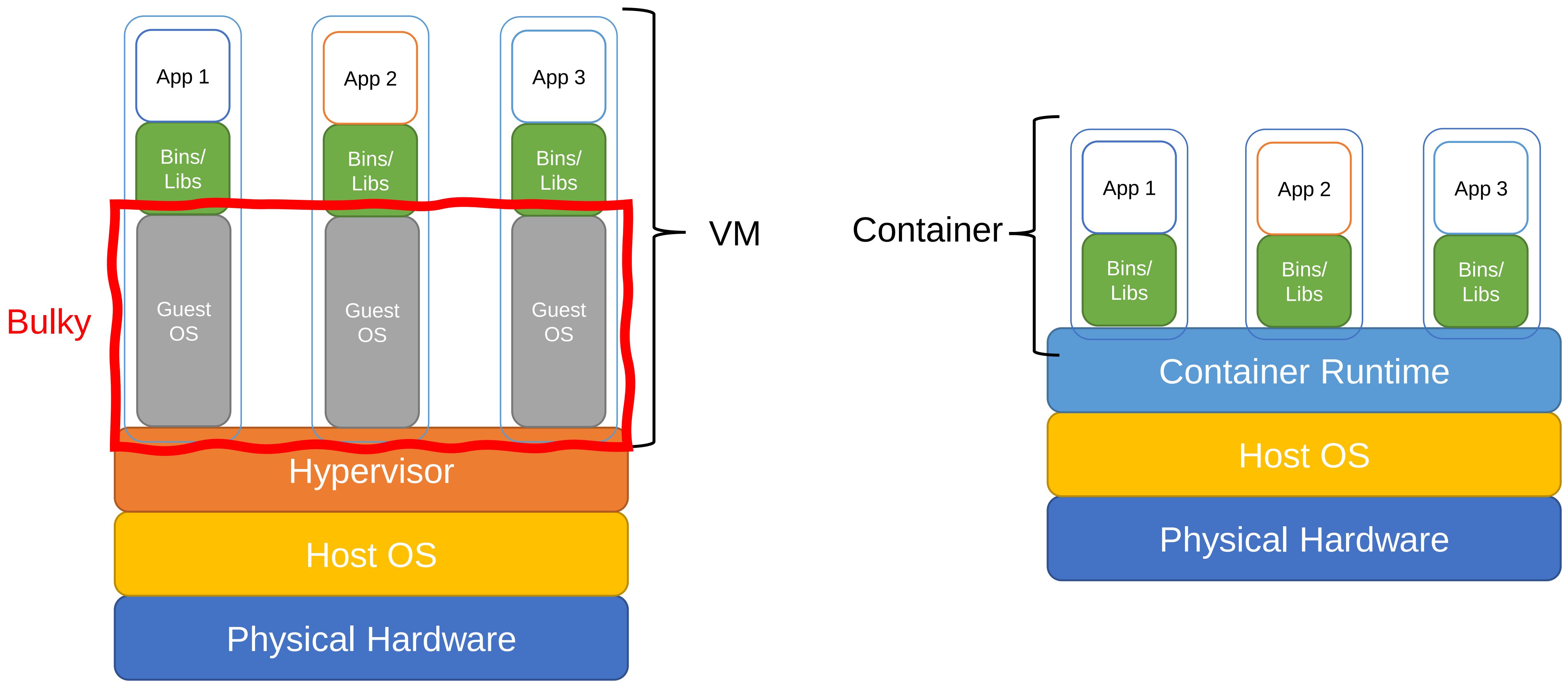
- What is a container?
- Comparison to virtual machines
- Containerizing an existing application
- Advantages / disadvantages of using containers
- Assignment: Running a containerized SCDS application

What is a Container?

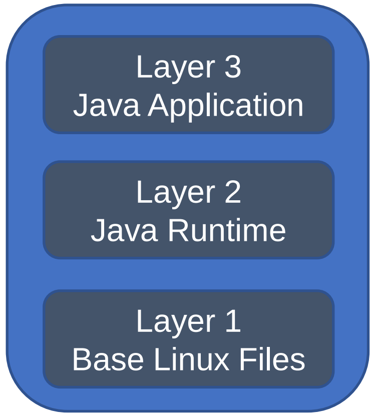


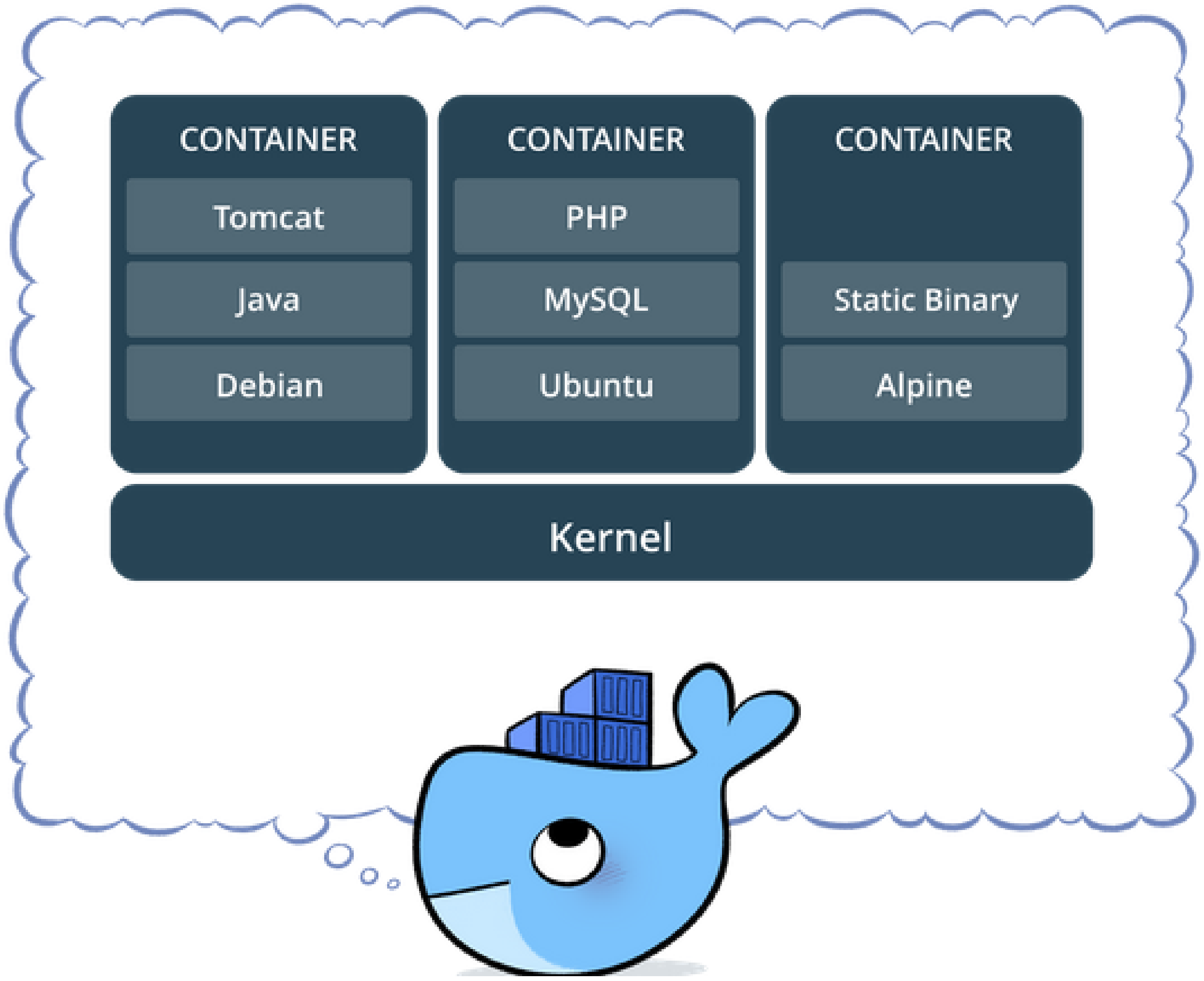
- “Containerization is an approach to software development in which an application or service, its dependencies, and its configuration (abstracted as deployment manifest files) are packaged together as a container image.”
- Analogy of a meal-delivery box vs. going to the store to buy for a recipe
  - Container gives you everything you need to do stuff without bloat
- Segmenting physical resources vs. virtualizing hardware
- Generally, containers use less resources than traditional virtual machines and improve portability of software

Containers vs Virtual Machines



Containers are layered





Advantages of Using Containers

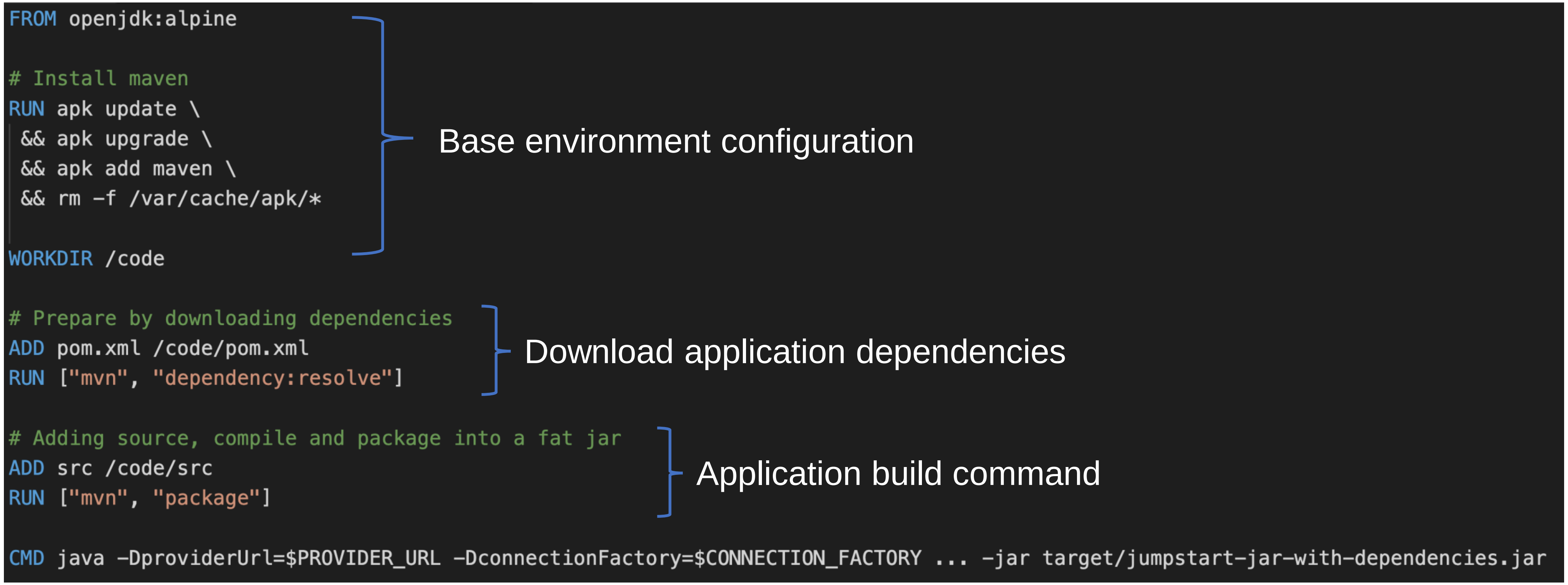
- Build once, run virtually anywhere
- Faster launch time than a Virtual Machine
  - Simplifies service scaling
  - Service scaling can be done more rapidly for periods of high traffic demand
- “It works on my machine”
  - Container images bring along everything that’s needed
  - More consistent development environment
  - Consistent building (ex Hadoop)
- Can build more generic applications to which you apply run-time configuration
- Micro services architecture

Disadvantages of Using Containers

- Harder to debug than native runs
- Some additional up-front effort to implement
- Can complicate CI/CD pipelines for building and testing
- Typical workflows rely on Image Registries – more infrastructure
- “Mega containers” (monoliths)
- Some applications are not appropriate for containerization
- Set up of runtime container platform

Question Break

Image Definition: The Dockerfile



```
FROM openjdk:alpine

# Install maven
RUN apk update \
 && apk upgrade \
 && apk add maven \
 && rm -f /var/cache/apk/*

WORKDIR /code

# Prepare by downloading dependencies
ADD pom.xml /code/pom.xml
RUN ["mvn", "dependency:resolve"]

# Adding source, compile and package into a fat jar
ADD src /code/src
RUN ["mvn", "package"]

CMD java -DproviderUrl=$PROVIDER_URL -DconnectionFactory=$CONNECTION_FACTORY ... -jar target/jumpstart-jar-with-dependencies.jar
```

Base environment configuration

Download application dependencies

Application build command

Default application run command

Orchestration

- Orchestration facilitates:
  - Running multiple different services in cohesive container environment (e.g., a web app with a backend API and database storage)
  - Running containers at scale (e.g., 5 versions of the same service) and load balancing

- Abstracts critical operational efforts
  - Service scaling, networking, load balancing, health monitoring

- Several tools/platforms out there for handling orchestration (e.g., Docker Swarm, OpenShift, Kubernetes, etc.)

- We'll talk more about container orchestration at Webinar #2 and use orchestration during the in-person event in August

Hands-On Training Summary

- **Goals**
  - Practice working with containers
  - Get SCDS access squared away

- **What will you be doing?**
  - Gaining access to SCDS
  - Downloading the Jumpstart Kit provided on SWIFT portal
  - Building a Docker container image
  - Preparing Jumpstart Kit configuration
  - Running a Docker container image

- **Step-by-step slides and video will be posted to SWIFT portal after this webinar concludes**

Questions?

Upcoming Schedule

- **Webinar #2 - July 19, 2022 at 1:00PM EDT**
  - *Consuming SCDS Data and Container Orchestration*

- **Webinar #3 – August 16, 2022 at 1:00PM EDT**
  - *Recapping the Trajectory Deviation Study*

- ***Developer Workshop – August 29-30, 2022***
  - *In Person Event at MITRE McLean Campus*

SWIFT Developer Series

Webinar #1 | Hands-On Training

June 21, 2022

Hands-On Training

Some Upfront Assumptions

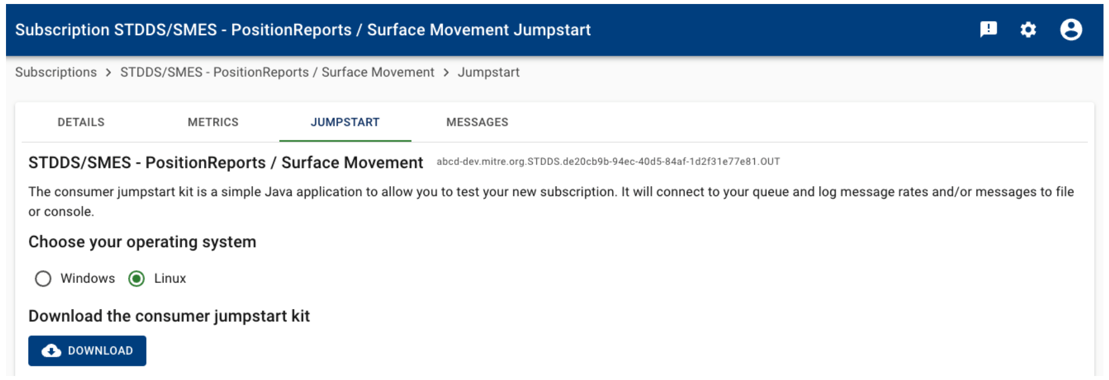
- Have an SCDS Account and associated data approvals
  - https://support.swim.faa.gov/hc/en-us/articles/360034136992-How-to-create-a-SWIFT-Portal-account

- Have Docker / Docker Desktop installed on the machine
  - https://docs.docker.com/get-docker/

Hands-On Training

Getting Set-Up

1. Create a new subscription in SCDS for use with this exercise
2. Create a folder for development
3. Download the Jumpstart Kit for Linux



Subscription STDDS/SMES - PositionReports / Surface Movement Jumpstart
Subscriptions > STDDS/SMES - PositionReports / Surface Movement > Jumpstart
DETAILS
METRICS
JUMPSTART
MESSAGES

STDDS/SMES - PositionReports / Surface Movement abcd-dev.mitre.org.STDDS.de20cb9b-94ec-40d5-84af-1d2f31e77e81.OUT

The consumer jumpstart kit is a simple Java application to allow you to test your new subscription. It will connect to your queue and log message rates and/or messages to file or console.

Choose your operating system

Windows [✓ Linux]

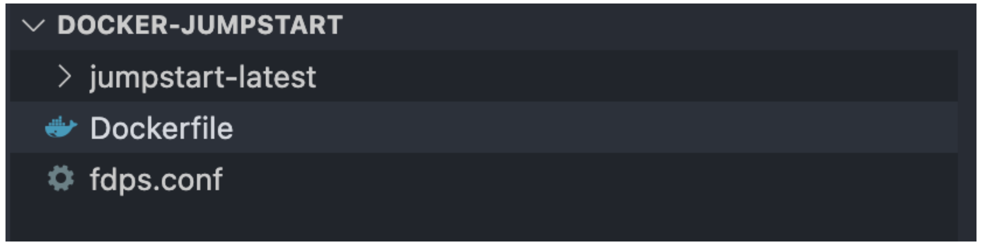
Download the consumer jumpstart kit

4. Extract the downloaded archive to the development folder

Hands-On Training

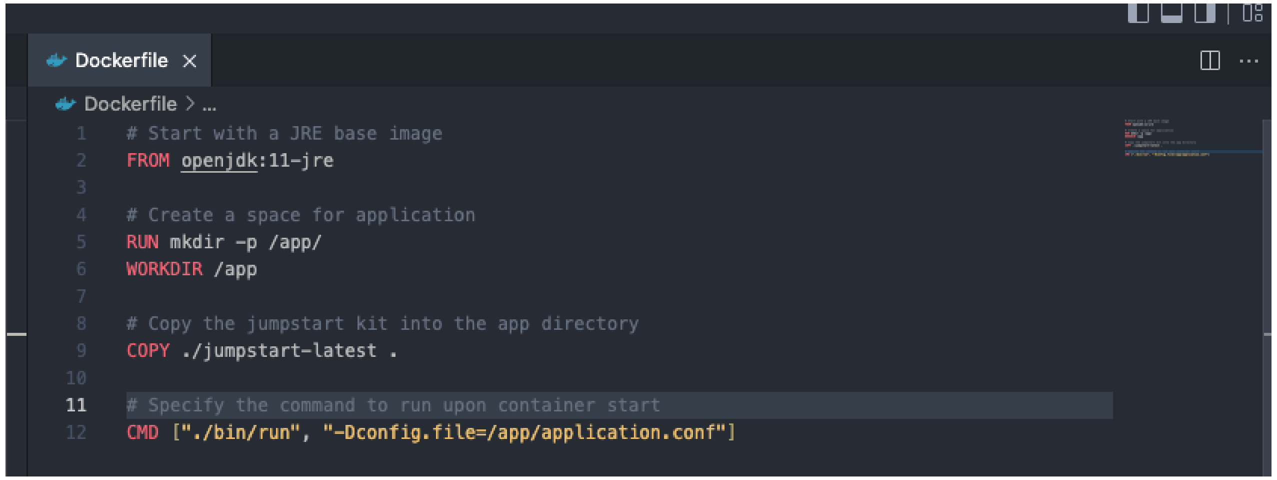
Building a Docker Container Image

1. Create a Dockerfile in the development directory (sibling to jumpstart-latest directory



```
DOCKER-JUMPSTART
  > jumpstart-latest
  Dockerfile
  fdps.conf
```

2. Write the Dockerfile



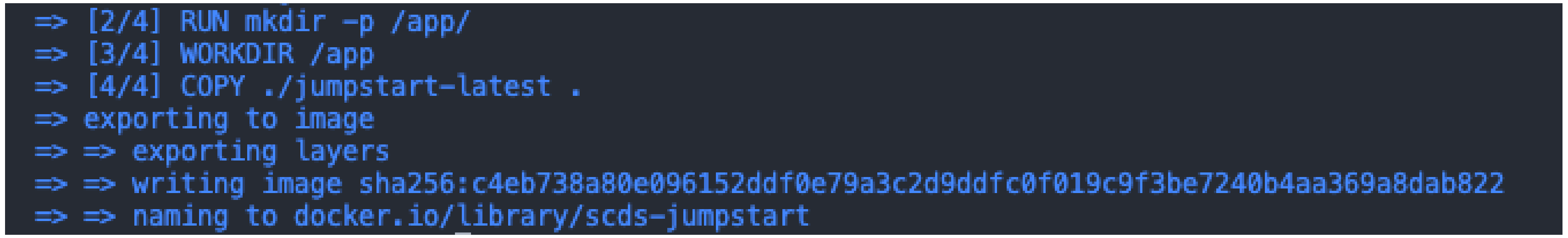
```
# Start with a JRE base image
FROM openjdk:11-jre

# Create a space for application
RUN mkdir -p /app/
WORKDIR /app

# Copy the jumpstart kit into the app directory
COPY ./jumpstart-latest .

# Specify the command to run upon container start
CMD ["./bin/run", "-Dconfig.file=/app/application.conf"]
```

Hands-On Training

Build the Container Image

- From the development directory run the following docker command:
  - `docker build -t scds-jumpstart .`

- The previous command should end similar to this:



```
=> [2/4] RUN mkdir -p /app/
=> [3/4] WORKDIR /app
=> [4/4] COPY ./jumpstart-latest .
=> exporting to image
=> => exporting layers
=> => writing image sha256:c4eb738a80e096152ddf0e79a3c2d9ddfc0f019c9f3be7240b4aa369a8dab822
=> => naming to docker.io/library/scds-jumpstart
```

Hands-On Training

Configuration Files

- Create a configuration file in the development directory for the subscription you created earlier (e.g., fdps.conf)
- The configuration file should have the following lines using the values from the subscription “Details” page:

```
providerUrl:”<JMS Connection URL>"
queue:<Queue Name>
connectionFactory:<Connection Factory>
username:<Connection Username>
password:<Connection Password>
vpn:<Message VPN>
metrics:false
output:com.harris.cinnato.outputs.StdoutOutput
json:true
```

Make sure this is in quotes!!!

Hands-On Training

Run the Container

- From the development directory run the following docker command

  -
    ```
    docker run –it --rm --name scds-jumpstart-fdps -v
    ${PWD}/fdps.conf:/app/application.conf scds-jumpstart
    ```

- After running the above command, the container should launch and you should begin to see SCDS messages being received and displayed on the console:

```
{"ns5:MessageCollection":{"xmlns:ns2":"http://www.fixm.aero/base/3.0","xmlns:ns5":"http://www.faa.aero/nas/3.0","xmlns:ns3":"http://www.fixm.aero/flight/3.0","xmlns:ns4":"http://ww
w.fixm.aero/foundation/3.0","message":{"flight":{"gufi":{"codeSpace":"urn:uuid","content":"65617b75-7c99-4832-bd03-
972e775d360e"},"enRoute":{"xsi:type":"ns5:NasEnRouteType","boundaryCrossings":{"xsi:type":"ns5:NasUnitBoundaryType","handoff":{"xsi:type":"ns5:NasHa ndoffType","receivingUnit":{"xsi
:type":"ns2:IdentifiedUnitReferenceType","sectorIdentifier":35,"unitIdentifier":"ZDC"},"event":"INITIATION","transferringUnit":{"xsi:type":"ns2:IdentifiedUnitReferenceType","sector
Identifier":50,"unitIdentifier":"ZDC"}}}},"flightIdentification":{"computerId":113,"aircraftIdentification":"AAL688","siteSpecificPlanId":634,"xsi:type":"ns5:NasFlightIdentificatio
nType"},"arrival":{"xsi:type":"ns5:NasArrivalType","runwayPositionAndTime":{"runwayTime":{"estimated":{"time":"2022-06-
07T17:51:00Z"}}},"arrivalPoint":"KMIA"},"flightPlan":{"identifier":"KN50460300"},"xsi:type":"ns5:NasFlightType","centre":"ZDC","flightStatus":{"xsi:type":"ns5:NasFlightStatusType",
"fdpsFlightStatus":"ACTIVE"},"supplementalData":{"xsi:type":"ns5:NasSupplementalDataType","additionalFlightInformation":{"nameValue":[{"name":"MSG_SEQ_NO","value":24252322},{"name"
:"FDPS_GUFI","value":"us.fdps.2022-06-
07T14:01:00Z.000/13/300"},{"name":"FLIGHT_PLAN_SEQ_NO","value":8},{"name":"SOURCE_TIME_AND_SEQ","value":1608307321},{"name":" SOURCE_TIME","value":"16_08_30"}]}}},"source":"OH","oper
ator":{"operatingOrganization":{"organization":{"name":"AAL"}}},"system":"SLC","departure":{"departurePoint":"KJFK","xsi:type":"ns5:NasDepartureType","runwayPositionAndTime":{" runw
ayTime":{"actual":{"time":"2022-06-07T15:29:00Z"}}}},"timestamp":"2022-06-07T16:08:30.240Z"},"xsi:type":"ns5:FlightMessageType","xmlns:xsi":"http://www.w3.org/2001/XMLSchema-
instance"}}}
```

- To exit the container use Ctrl-C to stop the running jumpstart app

Hands-On Training

Some Things to Try

- Use difference configuration files to configure the container to consume different SCDS subscriptions:
  - `docker run –it --rm --name scds-jumpstart-stdds -v ${PWD}/stdds.conf:/app/application.conf scds-jumpstart`
- Change the output of the jumpstart application
  - View options in "`${PWD}/jumpstart-latest/README.md`"
  - Try making a volume mount between the host machine and the container's log directory to save files on the host machine
    - Hint: Add `-v ${PWD}/log:/app/log` to `docker run` command