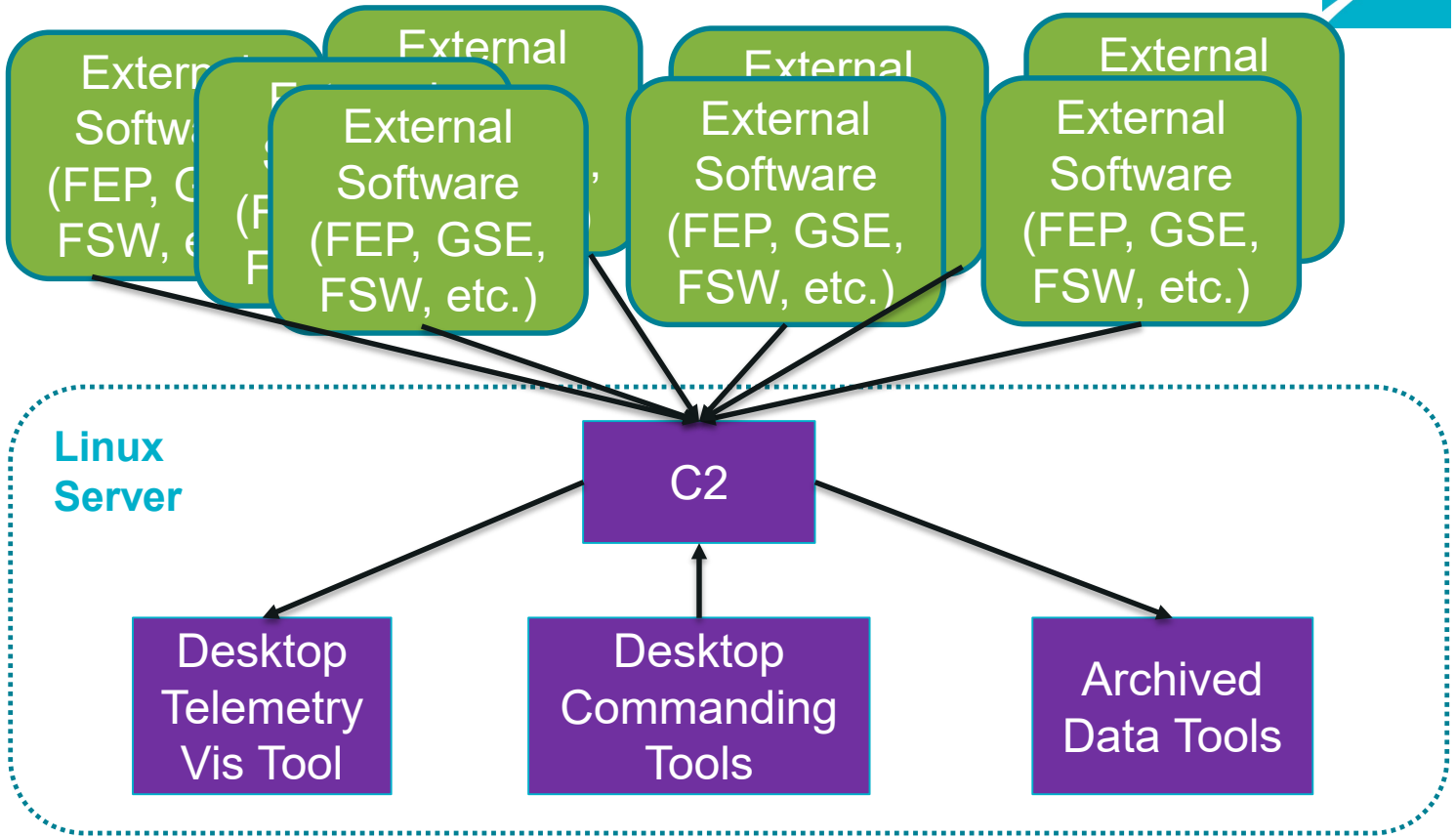
An Open Architecture for an Infinitely Scalable C2 system using Docker, Kubernetes, and Istio

Ryan Melton

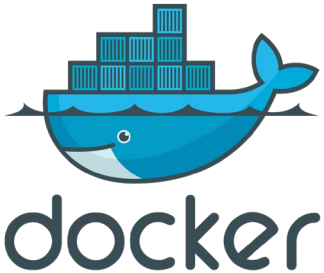
Ball Aerospace

The Problem


External Software (FEP, GSE, FSW, etc.)
External Software (FEP, GSE, FSW, etc.)
External Software (FEP, GSE, FSW, etc.)
Linux Server
C2
Desktop Telemetry Vis Tool
Desktop Commanding Tools
Archived Data Tools
Key:
Linux Process
External

The Solution: Infrastructure

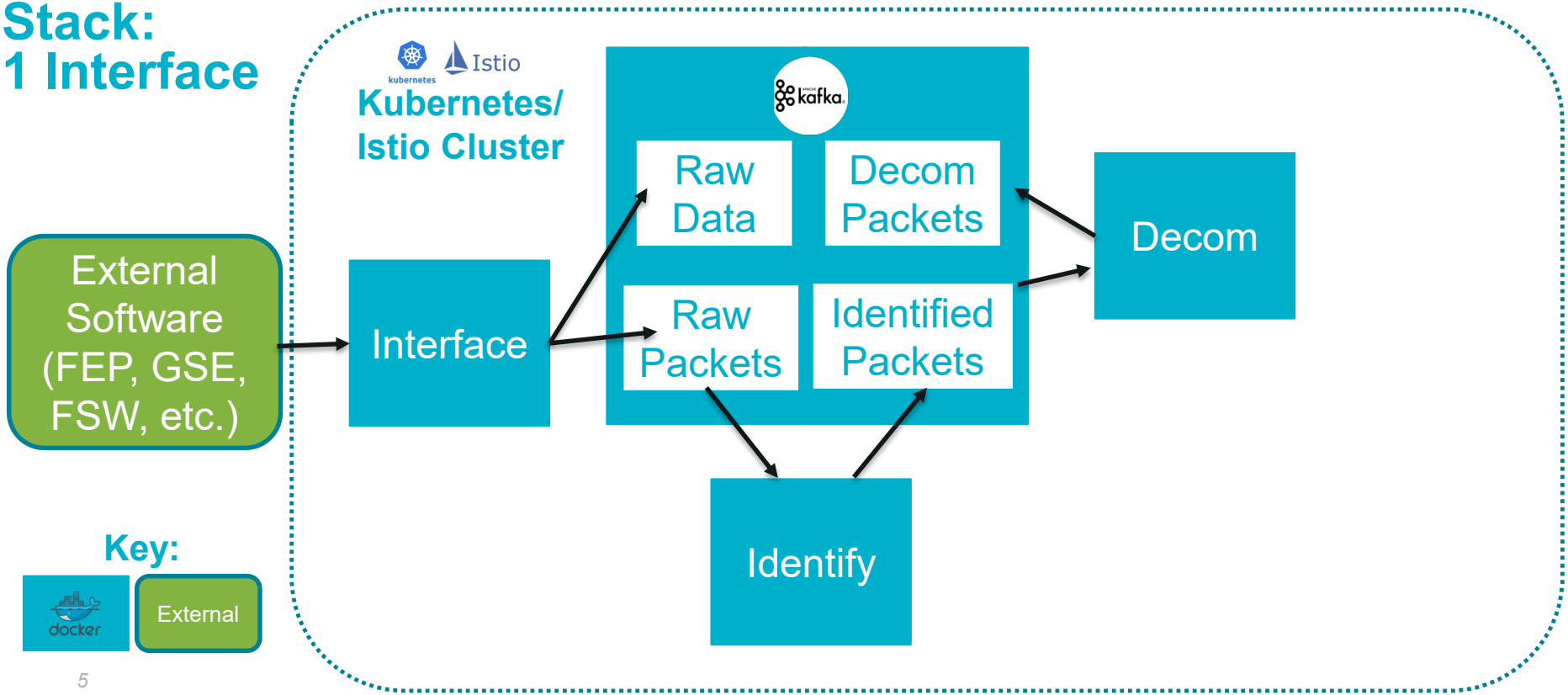
- **Docker** – Builds and runs containers – Ideally one process per container
- **Kubernetes** – Organizes and orchestrates containers in “pods” – Provides networking, process monitoring, DNS, replication, and autoscaling
- **Istio** – Adds a proxy server to every pod that all network traffic goes through – Allows mTLS without any application changes, detailed built in monitoring metrics, and strict security rule enforcement



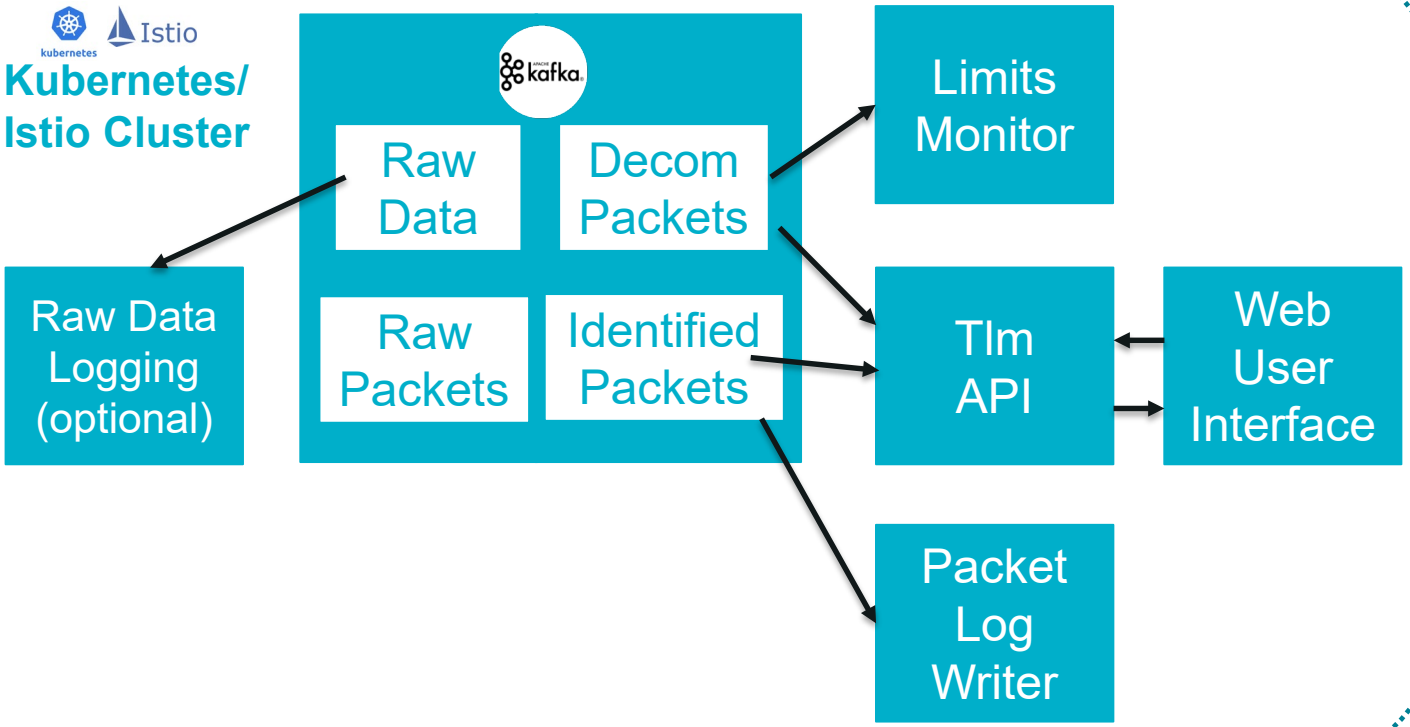
Event Driven Asynchronous Comm

- Streaming Message Broker
  - Provides a common asynchronous interface between the containers that make up the C2 system
  - Scales easily to new satellites by simply adding more containers
  - High performance from a simple internal architecture – streams are basically just files
  - Primarily used to receive streaming telemetry

Realtime Telemetry Microservices Stack: 1 Interface



Realtime Telemetry Microservices Stack: 1 Interface



Key:

docker External

Architecture Break #1: Docker

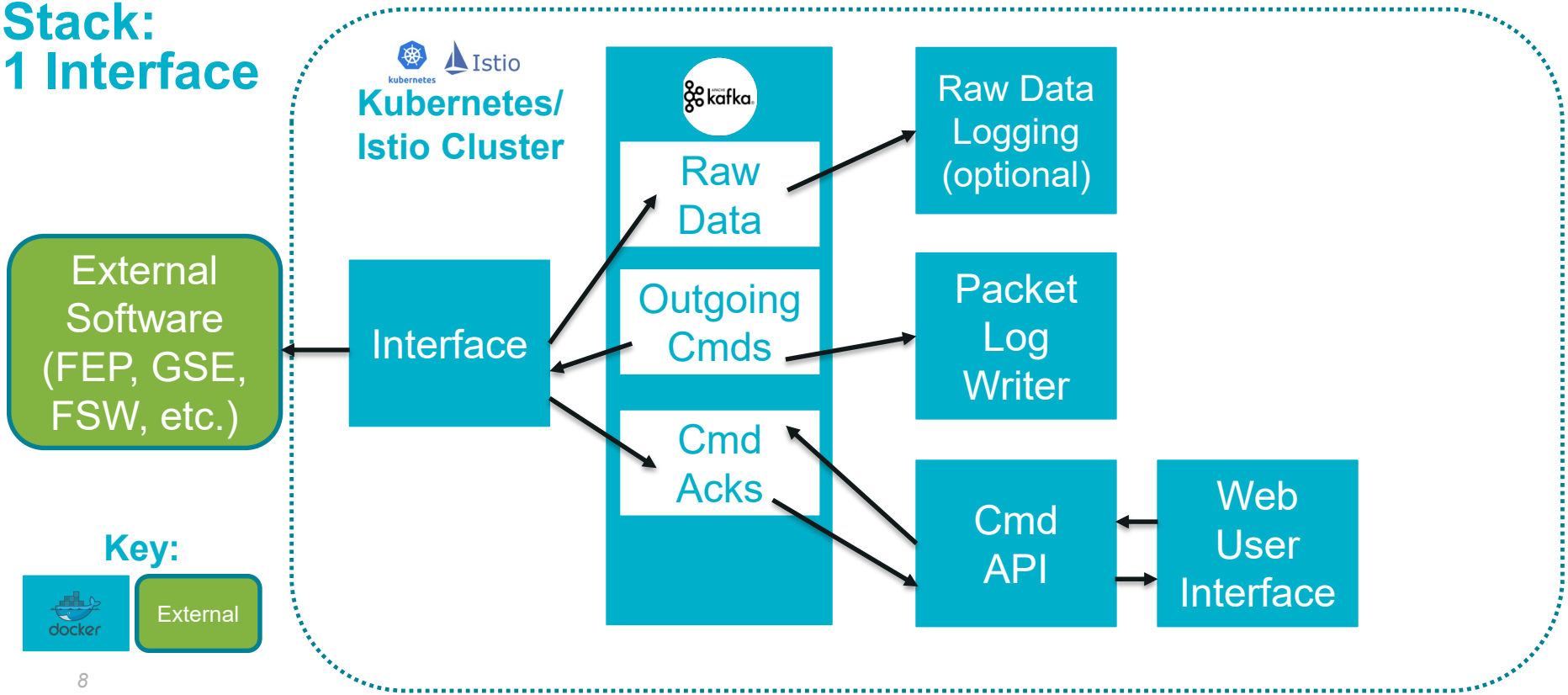
- Processes inside a docker container (1):

```
root@243ab21658d6: /
root@243ab21658d6:/# ps -e
  PID TTY          TIME CMD
    1 pts/0    00:00:00 bash
   10 pts/0    00:00:00 ps
root@243ab21658d6:/#
```

- Processes inside a virtual machine (280!):

```
27413 ?        00:00:00 unity-fallback-
27417 ?        00:00:00 polkit-gnome-au
27452 ?        00:00:42 compiz
27475 ?        00:00:00 evolution-calen
27486 ?        00:00:00 gvfs-udisks2-vo
27505 ?        00:00:00 evolution-calen
27516 ?        00:00:00 gvfs-gphoto2-vo
27517 ?        00:00:00 evolution-addre
27525 ?        00:00:00 gvfs-mtp-volume
27533 ?        00:00:00 gvfs-afc-volume
27536 ?        00:00:00 evolution-addre
27549 ?        00:00:00 gvfs-goa-volume
27588 ?        00:00:00 gvfsd-metadata
27621 ?        00:00:00 kworker/u2:1
27625 ?        00:00:00 gvfsd-trash
27634 ?        00:00:00 dbus-launch
27635 ?        00:00:00 dbus-daemon
27637 ?        00:00:00 notify-osd
27641 ?        00:00:00 gvfsd
27651 ?        00:00:00 gnome-screensav
27662 ?        00:00:00 sh
27666 ?        00:00:00 zeitgeist-daemo
27674 ?        00:00:00 zeitgeist-fts
27675 ?        00:00:00 zeitgeist-datah
27689 ?        00:00:01 gnome-terminal-
27696 ?        00:00:00 gconfd-2
27708 pts/36   00:00:00 bash
27929 ?        00:00:00 update-notifier
27940 pts/36   00:00:04 ruby
27982 ?        00:00:00 dhclient
28025 ?        00:00:00 kworker/0:3
28027 ?        00:00:00 kworker/0:4
28034 ?        00:00:00 deja-dup-monito
28057 pts/36   00:00:00 ps
29144 ?        00:00:00 systemd-udevd
31920 ?        00:00:00 kworker/0:1
31956 ?        00:00:00 apt.systemd.dai
32234 ?        00:34:18 unattended-upgr
32244 ?        00:00:02 gnome-terminal-
32251 pts/4    00:00:00 bash
32439 ?        00:00:02 postgres
32441 ?        00:00:00 postgres
32442 ?        00:00:02 postgres
32443 ?        00:00:02 postgres
32444 ?        00:00:01 postgres
32445 ?        00:00:01 postgres
32574 pts/4    00:00:00 su
32575 pts/4    00:00:00 bash
32599 pts/17   00:00:00 bash
rubytest@rmelton-VirtualBox:~$
```

Realtime Command Microservices Stack: 1 Interface



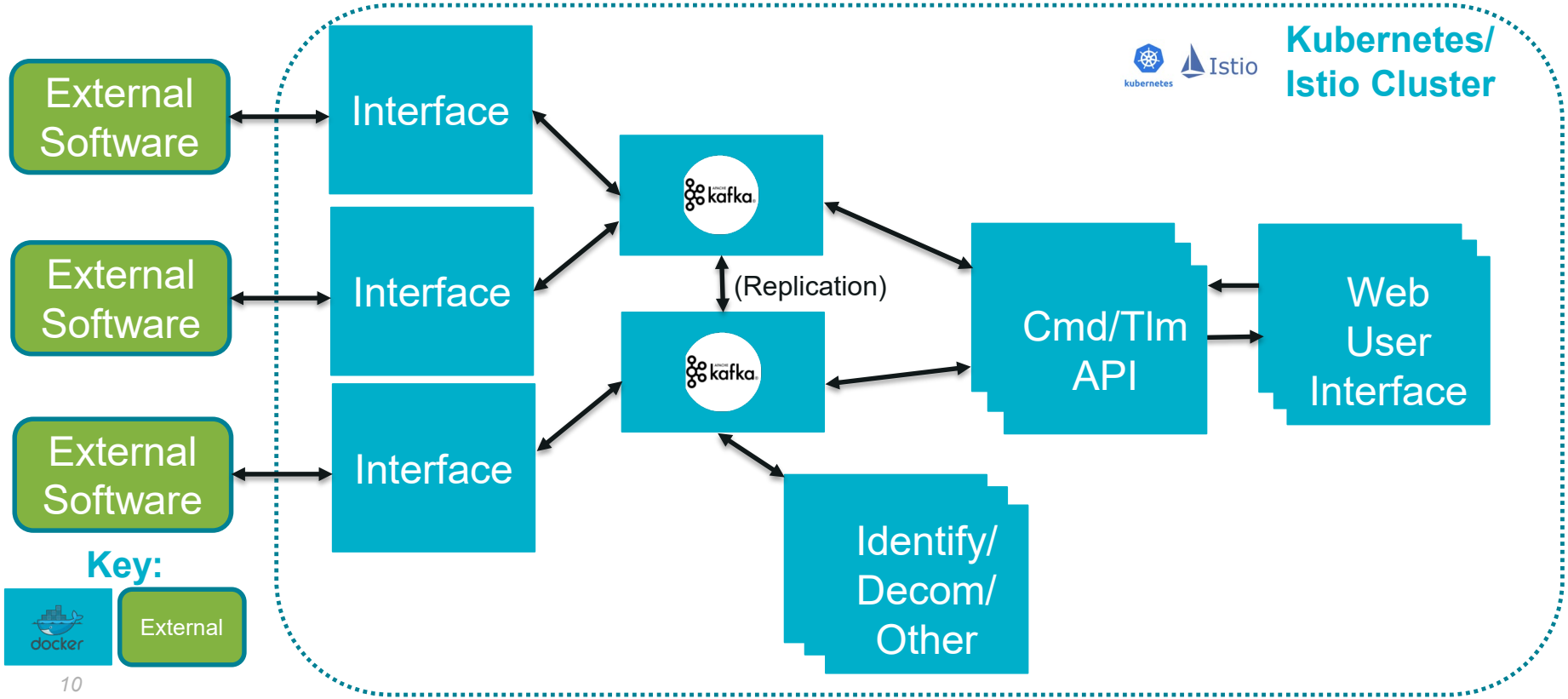
Architecture Break #2: Kubernetes

- Config file to deploy a replicated app in Kubernetes:

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: nginx-deployment
  labels:
    app: nginx
spec:
  replicas: 3
  selector:
    matchLabels:
      app: nginx
template:
    metadata:
      labels:
        app: nginx
    spec:
      containers:
      - name: nginx
        image: nginx:1.7.9
        ports:
        - containerPort: 80
---
apiVersion: v1
kind: Service
metadata:
  name: nginx-service
spec:
  selector:
    app: nginx
  ports:
    - protocol: TCP
      port: 80
      targetPort: 80
```

kubernetes

Cmd/Tlm Microservices Stack: Infinite



Architecture Break #3: Istio

- Installing Istio with mTLS in auto mode:

```
istioctl manifest apply --set profile=demo \
  --set values.global.mtls.auto=true \
  --set values.global.mtls.enabled=false
```

Summary

- Docker containers hold each microservice
- Kubernetes manages orchestrating and scaling microservices
- Istio provides encrypted communications between all microservices, and useful metrics
- Kafka provides an easily scalable message bus platform
- C2 architecture can be scaled to any number of interfaces, by adding more microservices and Kafka nodes when necessary

Questions

- Keep up with C2 at Ball at: cosmosrb.com/news

Backup Slides

Scaling

- As additional satellites and ground support hardware are added, new Kafka nodes can be added to continually scale
- Microservices performing identification, decommutation, and packet logging can be created and assigned to one or more pieces of hardware as bandwidth dictates
- The overall Kubernetes cluster can be grown by adding additional hardware nodes at anytime. In cloud environments, this can be done on demand.
- Additional PostgresSQL databases for archiving can be added to support unique new sets of hardware. The API will handle querying the correct backend database.

Reliability

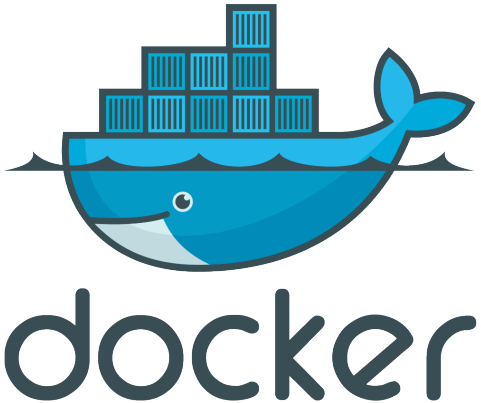
- Kafka data is replicated and any node can fail with automatic failover and no downtime
- Kubernetes will automatically respawn any pod that dies
- Postgres Database can be setup with master-master replication to support automatic failover

Security

- All internal C2 cluster data is encrypted and verified using mTLS as supported by Istio – This requires no changes to application code
- Istio policy ensures pods can only talk to other pods as required
- Ingress/Egress from the C2 cluster is secured by policy with only necessary access granted
- Kubernetes access is controlled by keys that are only available to admin users

Docker and Containers

- Like Virtual Machines But Better…
- Dockerfile
  - Text file captures the steps to build the container repeatably
- Lightweight
  - Starts in seconds
- Ideally only one process running in each container
- Contains only what is needed to run the single process



Kubernetes

- Orchestrates containers into groups called pods – starts them up and keeps them alive
- Provides an isolated network environment for the cluster and assigns them IP addresses / DNS names
- Provides load balancing for groups of containers
- Supports auto-scaling

kubernetes

Istio

- Adds a controlled proxy container to every pod
  - Enables Mutual TLS between pods with no changes to the application
  - Allows setting security constraints ie. Pod X can only talk to pod Y
  - Provides detailed built-in monitoring metrics, such as bandwidth in and out of each pod