LIFION by ADP

AVOID TRAPS IN YOUR STACK AND YOUR CULTURE

THE MICROLITH

Jason Melo - Chief Architect

MICROSERVICES AREN'T FREE. . . AND WE HAVE YET TO "SOLVE" IT.

Everyone

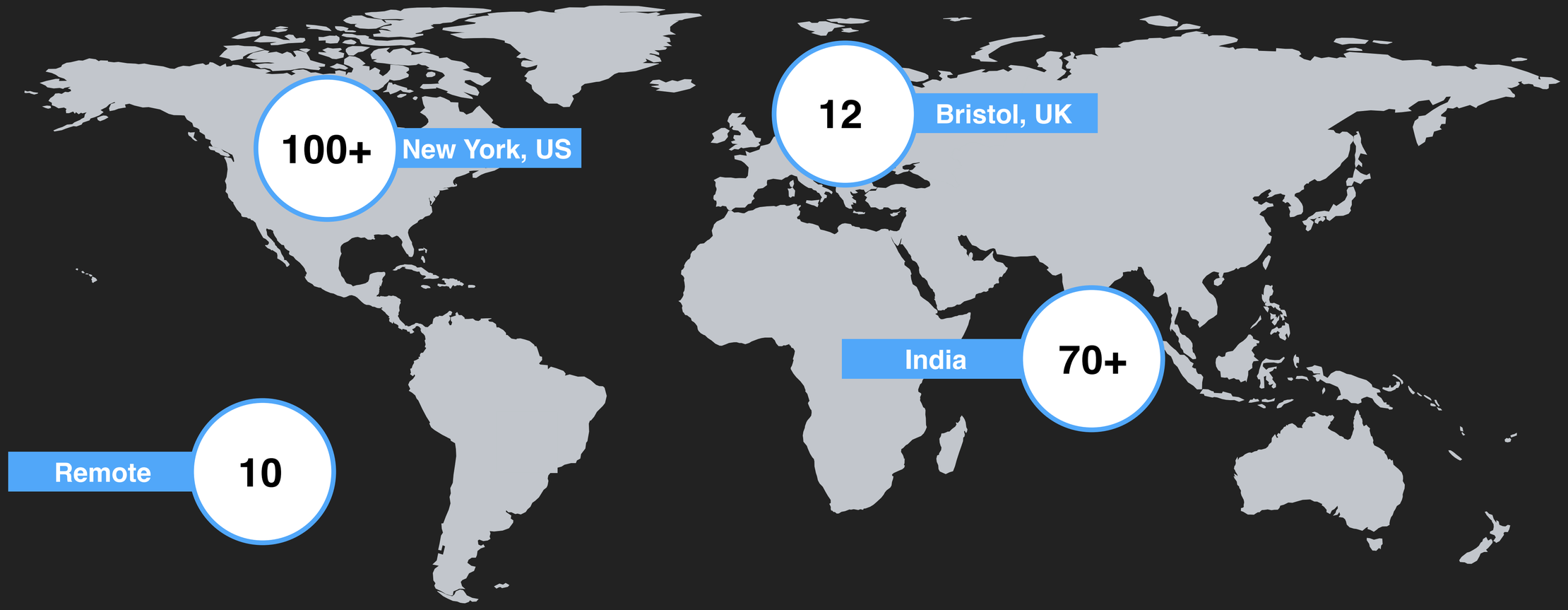
Microlith, defined as a Distributed Monolith

…a large number of “micro”services experiencing **coupled computing & data dependency** hell when executing top-level business transactions

IN CONCLUSION

- Realize your services as a fully **distributed architecture**
- **Microservices don’t come for free** - the investment in tech & culture is not trivial
- Go faster - your PaaS is a unit of computing, not a pattern to force your developers into

LIFION BY ADP – AROUND THE WORLD



SOME NUMBERS

- We deal with Humans, their professional & personal interactions from birth to grave on a global scale with localized precision - massive complexity

40m Active Users

500k Clients

30k Logins/sec

- ABOUT 11 MONTHS OLD
- 90+ SERVICES, MOSTLY DOCKER
- DISTRIBUTED MODEL
- EACH ENVIRONMENT ~1000 INSTANCES

LIF
ION
by ADP

SOME CONTEXT

- Essentially building a massive distributed JIT compiler & runtime in the cloud. Patterns that we've seen in the industry don’t quite fit, some unique challenges are presented
- At Lifion all services cannot be run in a fully independent way…Metadata is everything - our users create data structures as well as user interfaces via Metadata
- We now think of our **Microservices as a distributed architecture**, would you want your Cassandra cluster communicating & distributing data using synchronous http REST?…same rules should apply to your application

THE JOURNEY

PHASE 1 – CULTURAL MIND-SHIFT IS EVERYTHING

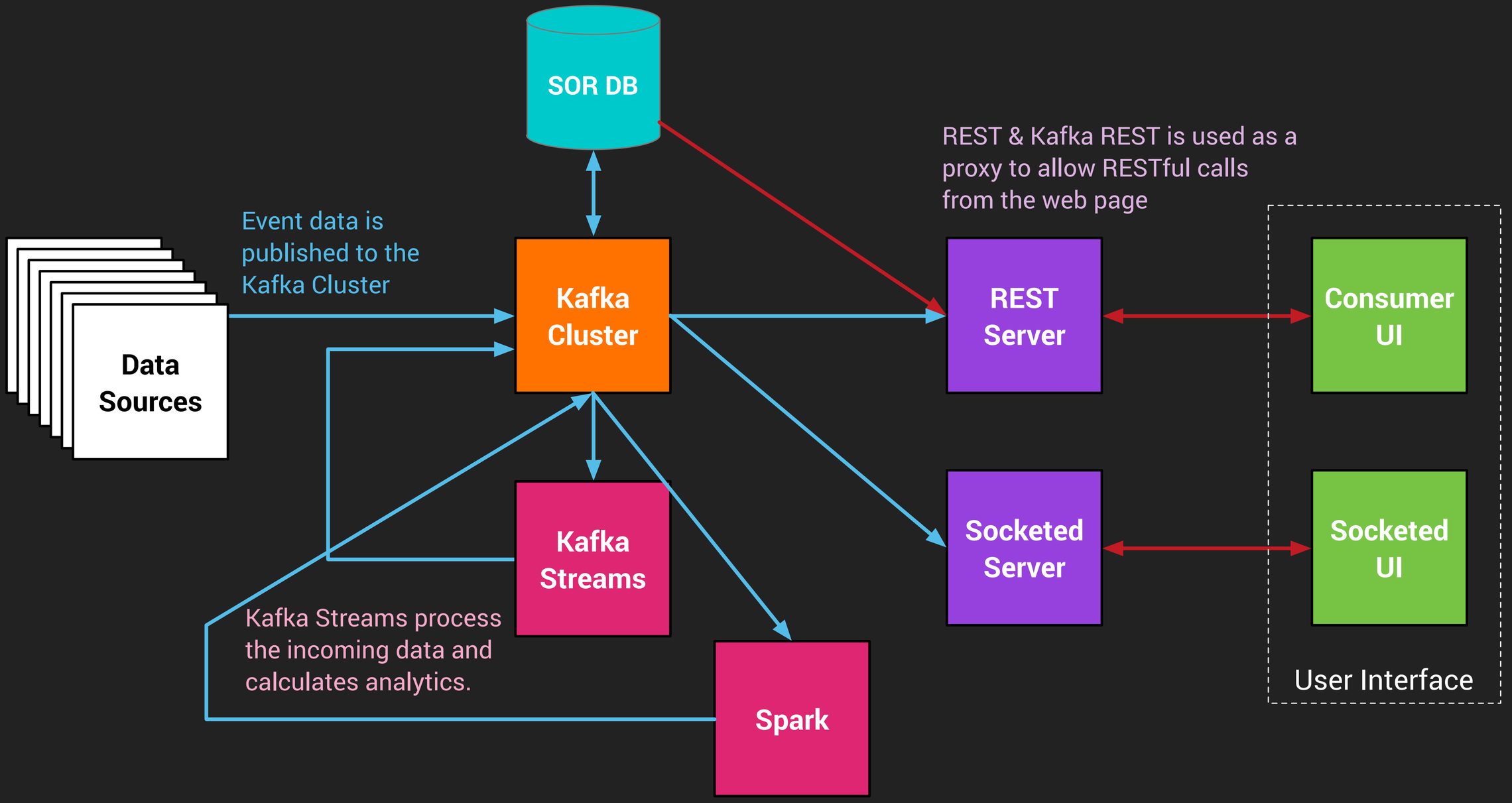
- Tooling & attitude necessary to go from 1 to 100's of services
- Gates of service entry
  - stateful vs stateless services
  - devops is not a team, its you!
  - service deltas are additive
  - well defined service ownership & accountability
- Embrace Chaos
  - tons of databases - Kafka, Couchbase, Cassandra, Graph, Time-series, MySQL, Memcache, Mongo
    - A service team defines their own Architecture & DB needs, we just don't care if they've satisfied our gates
  - multiple runtimes - Node (multi-ver), JVM (Java, Groovy & Scala) & Python

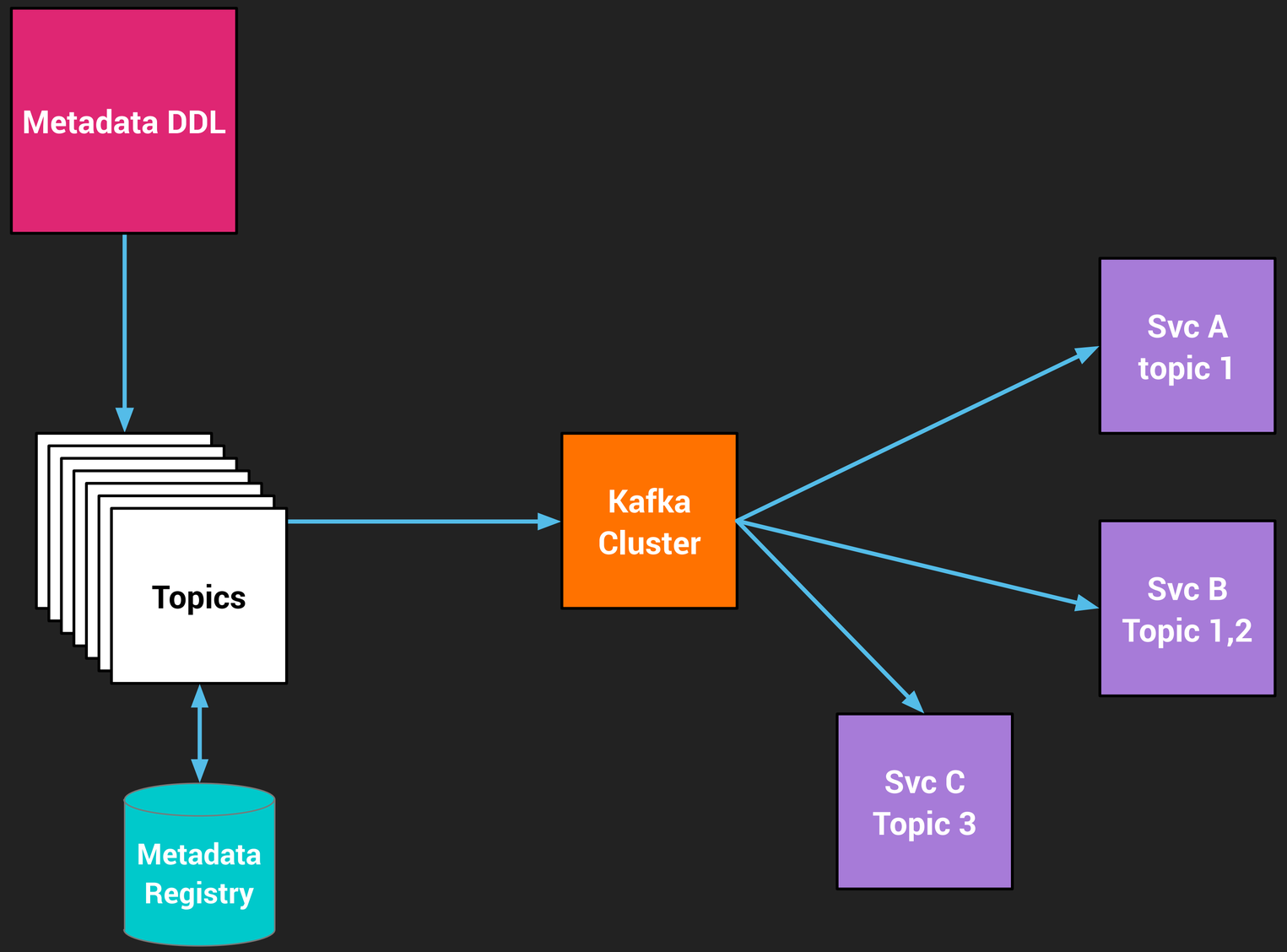
PHASE 1 – STRANGLING THE MONOLITH

- **Teams now control their services** with domain boundaries, guard rails & build pipelines
- Very **high network chattiness** dependent on synchronous REST
- Metadata makes mocks almost useless
  - **difficult to develop locally** with indicative use cases
  - a top-level **business transaction require synchronous calls** across a minimum of 8+ services
- Service registration & discovery is elegant **allowing us to solve the scaling problem & cloud challenges**

PHASE 2

With **Event messaging** patterns over REST, governance, aggregated logging, transaction tracers & architecture supporting streaming data we are **decoupling the Microlith - fun, not easy**


SOR DB
REST & Kafka REST is used as a proxy to allow RESTful calls from the web page
Event data is published to the Kafka Cluster
Data Sources
Kafka Cluster
REST Server
Consumer UI
Socketed Server
Socketed UI
Kafka Streams
Kafka Streams process the incoming data and calculates analytics.
Spark
User Interface



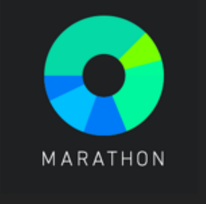
- Services register their intent to consume Metadata which are paired with our pipeline Topics
- As Metadata DDL occurs the distributed in-process cache in the SDK is invalidated (SDK's only for Metadata, not as clients)
- Significantly reduces synchronous chatter & eliminates the need to consume Metadata over the wire in realtime

LOGGING & ANALYTICS

- ELK, Kafka, Spark & S3 - aggregated logging solution
  - standardized data pipeline in use for compute, orchestration & logging
  - extremely high message throughput for realtime log ingestion & analysis
  - realtime security & performance analytics
- Standard logging libraries & formats governed across all service owners
- Realtime log analysis on deployment in blue/green scenarios using the same analytics platform

SCHEDULING & ORCHESTRATION

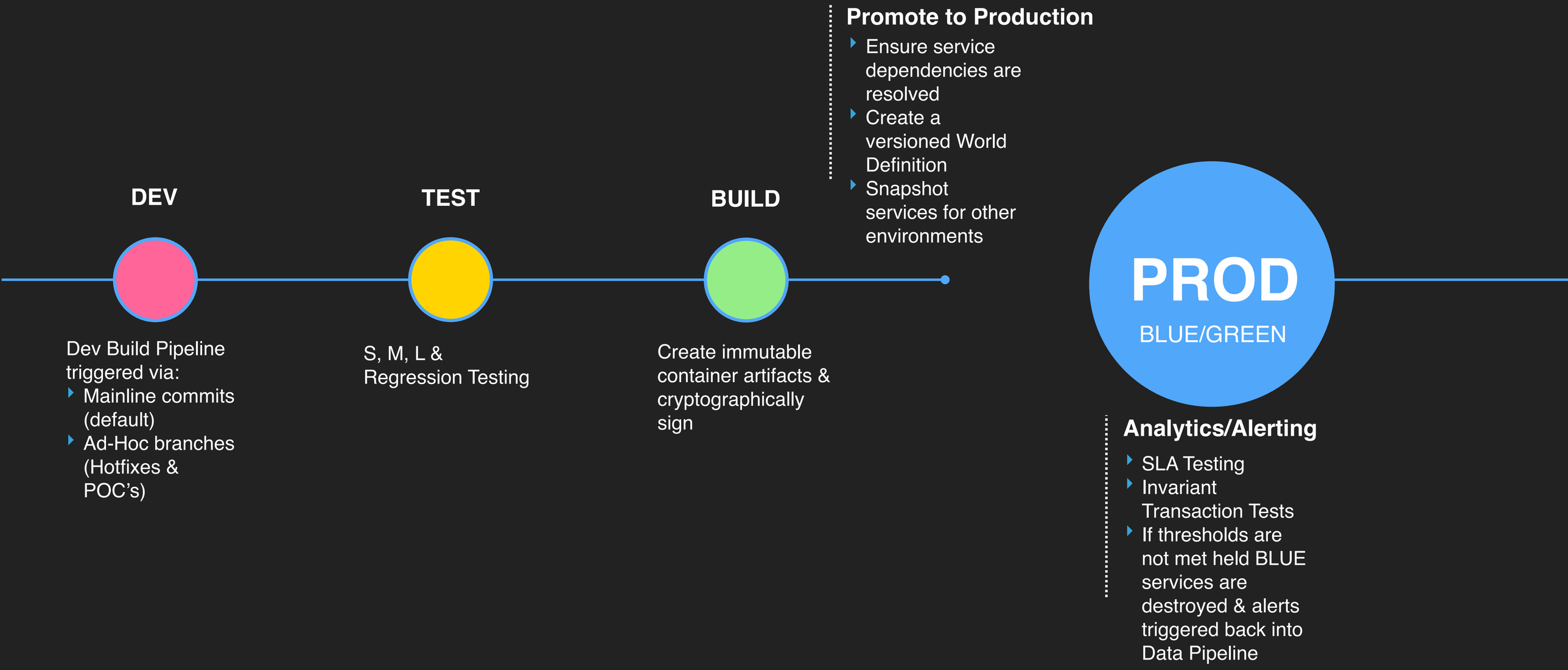
- Jenkins, Docker UCP/Swarm on AWS, Cloud Formation & Ansible
  - known, reliable and un-complicated
  - you need to work hard at getting persistence & scaling right
- Distributed data streaming, messaging & analytics pipeline runs on the above solidly, However:
  - Beta program with DC/OS, Triton & Kubernetes
    - known, a bit unstable (improving rapidly), higher operational complexity
    - local development challenging

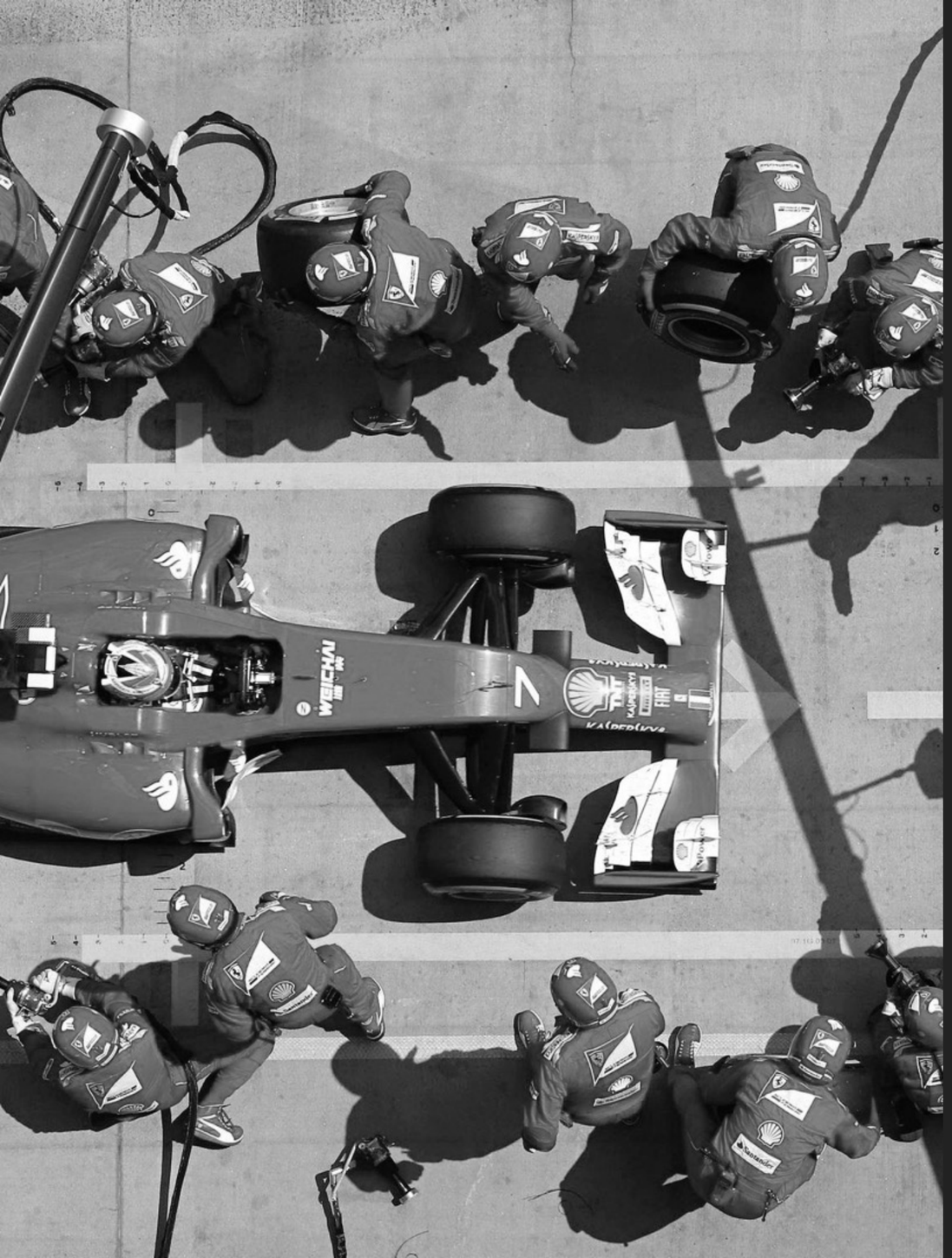


DESCRIBING OUR WORLD

- All service builds enter our “**World Definition**”
  - JSON definition similar to Marathon or Kubernetes catalog
  - each instance of **our world is versioned** for use on any environment, even local
- All orchestrated via an in-house tool we built named **Tailor**
  - we call it Tailor as it’s **our bespoke stitcher**
  - created down the road in lovely **Bristol, UK**
  - developers issue commands such as: **tlr world:up**

BUILD, SHIP, RUN





IN CONCLUSION

- APPLICATION SERVICES AS FULLY DISTRIBUTED ARCHITECTURE
- MICROSERVICES DON'T COME FOR FREE
- INFRASTRUCTURE IS A UNIT OF COMPUTE

www.lifion.com
jason.melo@adp.com

LIFION by ADP

thank you