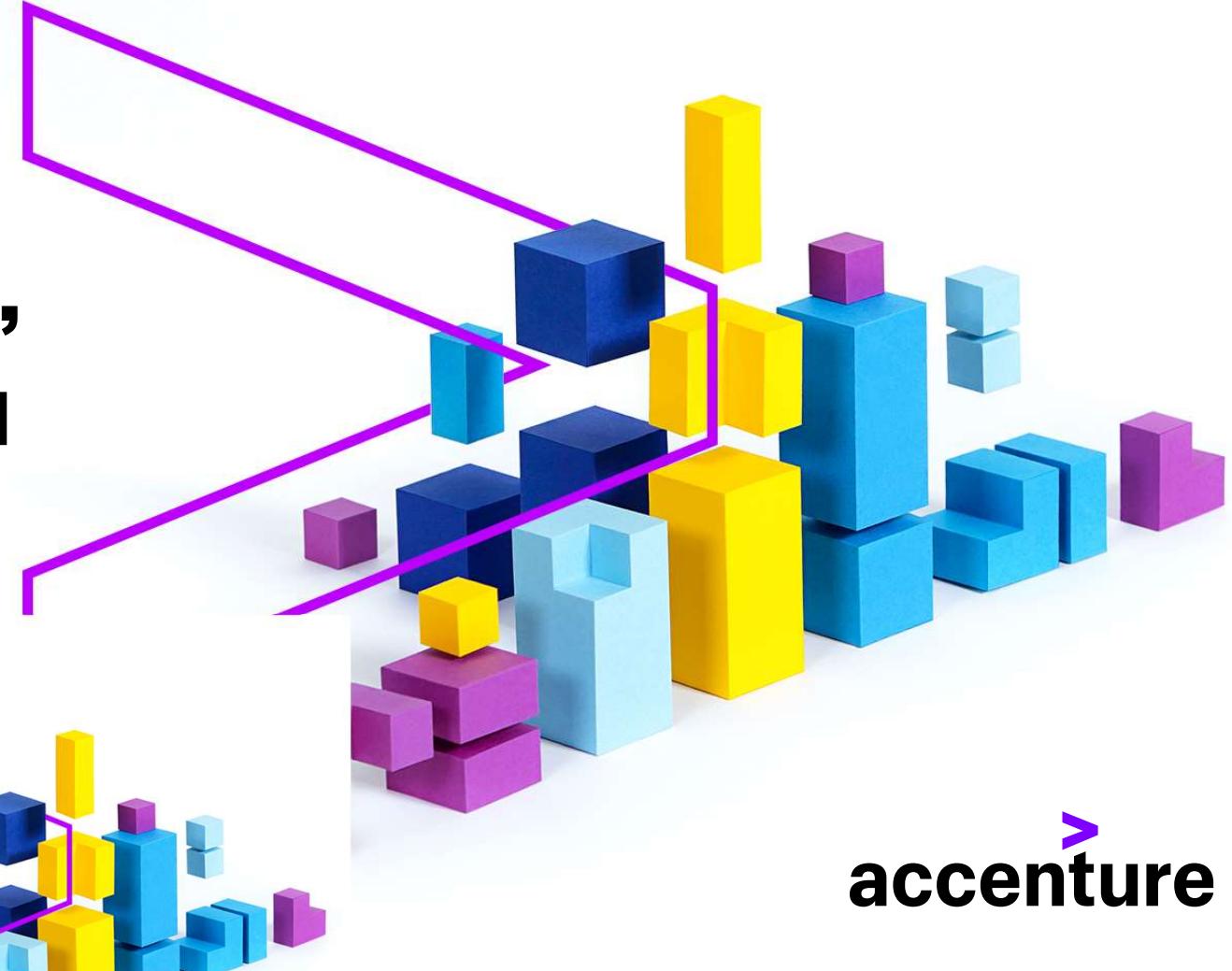


HIGH VOLUME, SECURE TRANSACTION SYSTEMS ON OPENSIFT

HIGH VOLUME, SECURE
TRANSACTION SYSTEMS ON
OPENSIFT
SOLUTIONS & CHALLENGES



accenture

TABLE OF CONTENTS



Program Overview



Lessons Learned



Akka



Platform Evolution

INTRODUCTION

ABOUT ACCENTURE FEDERAL SERVICES

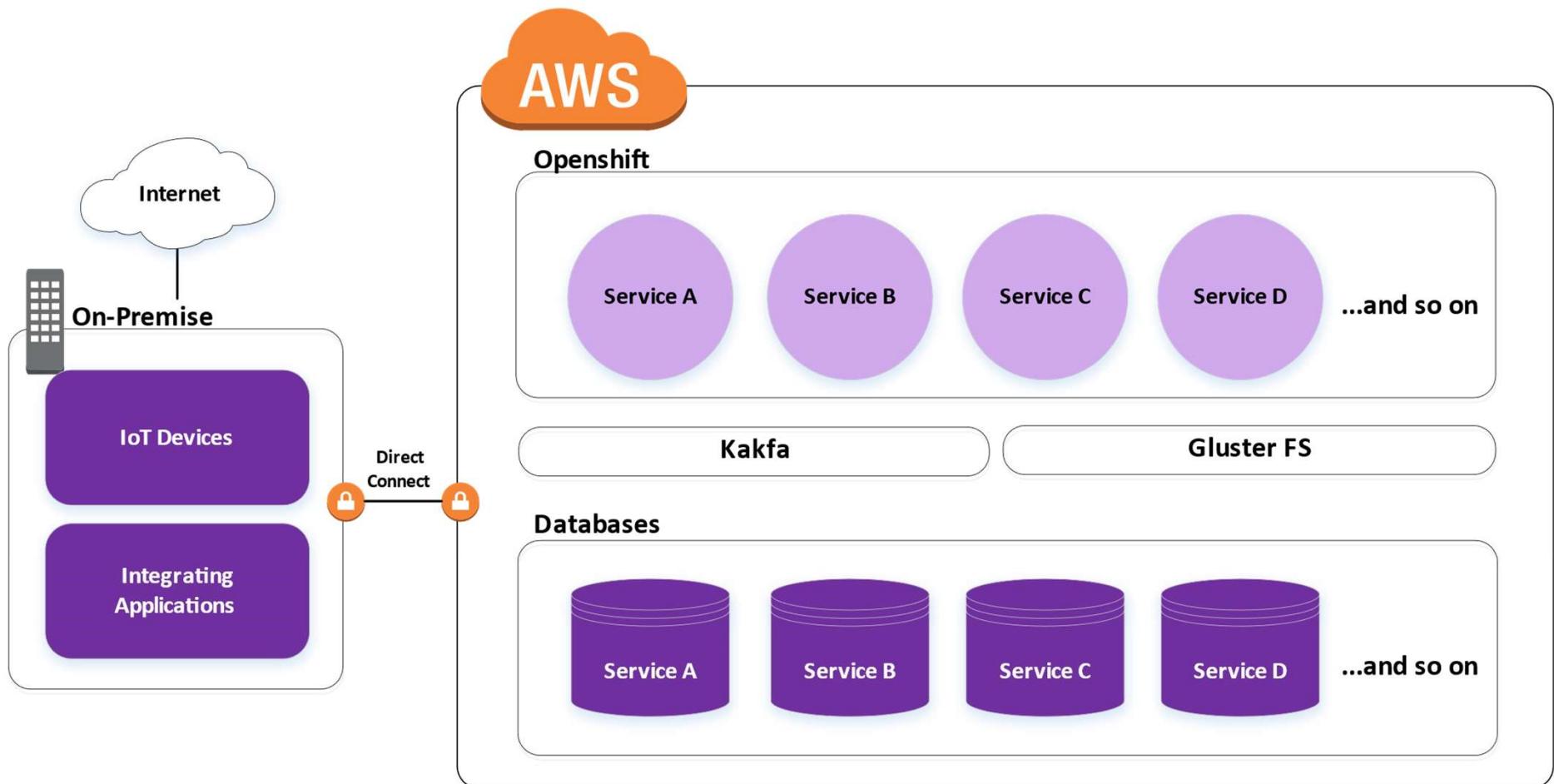
- Aligned to Public Sector, powered by commercial practices & experience
- Broad range of capabilities & partners – cloud, DevOps, AI, data science, etc.
- Culture of Innovation that is driven by outcomes

ABOUT THIS PRESENTATION

- Large Federal Agency – Openshift-centric program
- Focused on technology and lessons learned and not the business/functional aspects

PROGRAM OVERVIEW

PLATFORM ARCHITECTURE



PROCESSING GOALS SCALE & NON-FUNCTIONAL REQUIREMENTS

HIGH-VOLUME

20-50M raw scan events per day with spikes both daily and seasonally

NEAR REAL-TIME

Posting payments and generating customer data pushes with minute-level frequency

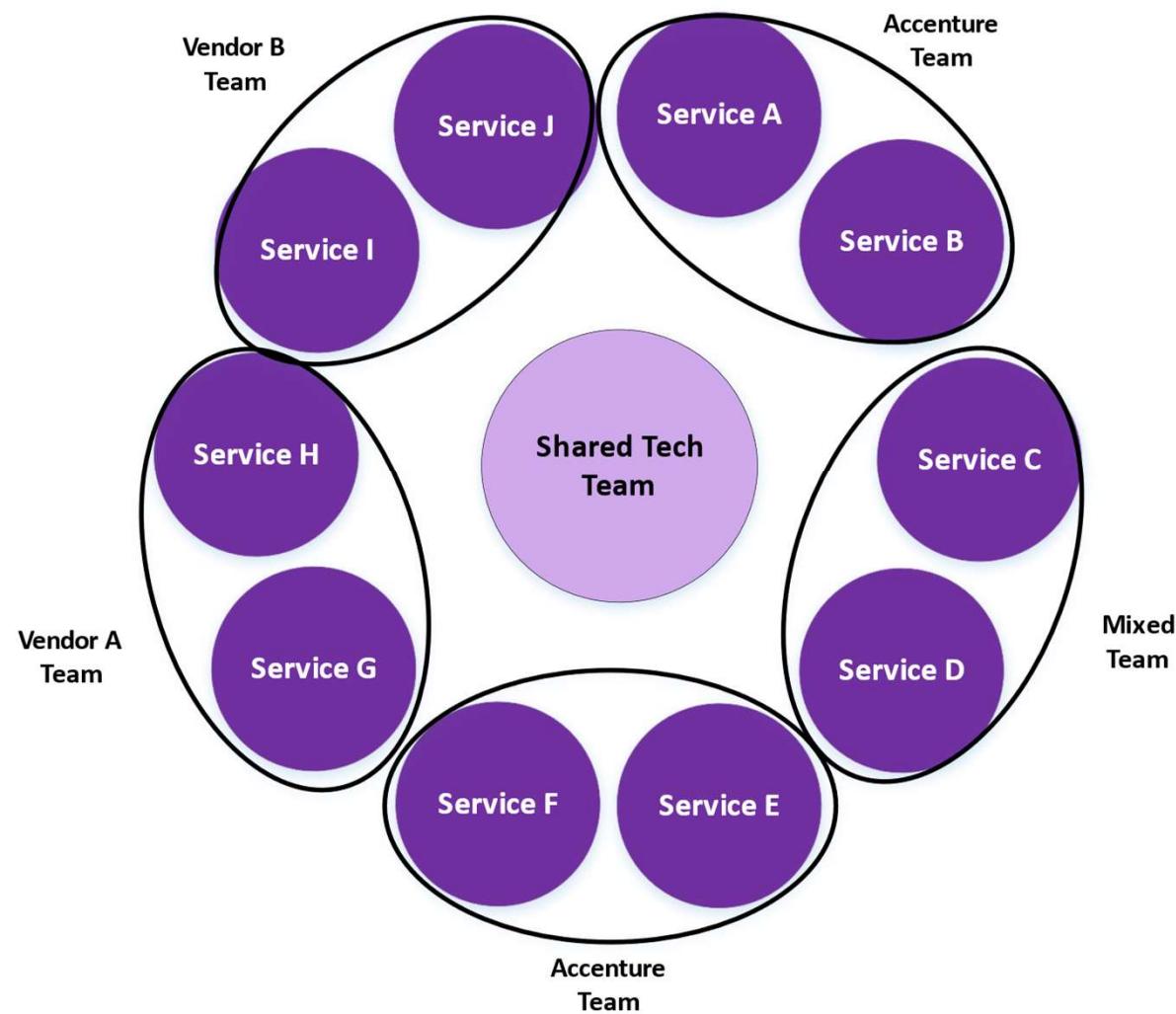
ENVIRONMENTS

8 environments each with 10 separate service domains each with dedicated databases and namespaces

INTEGRATIONS

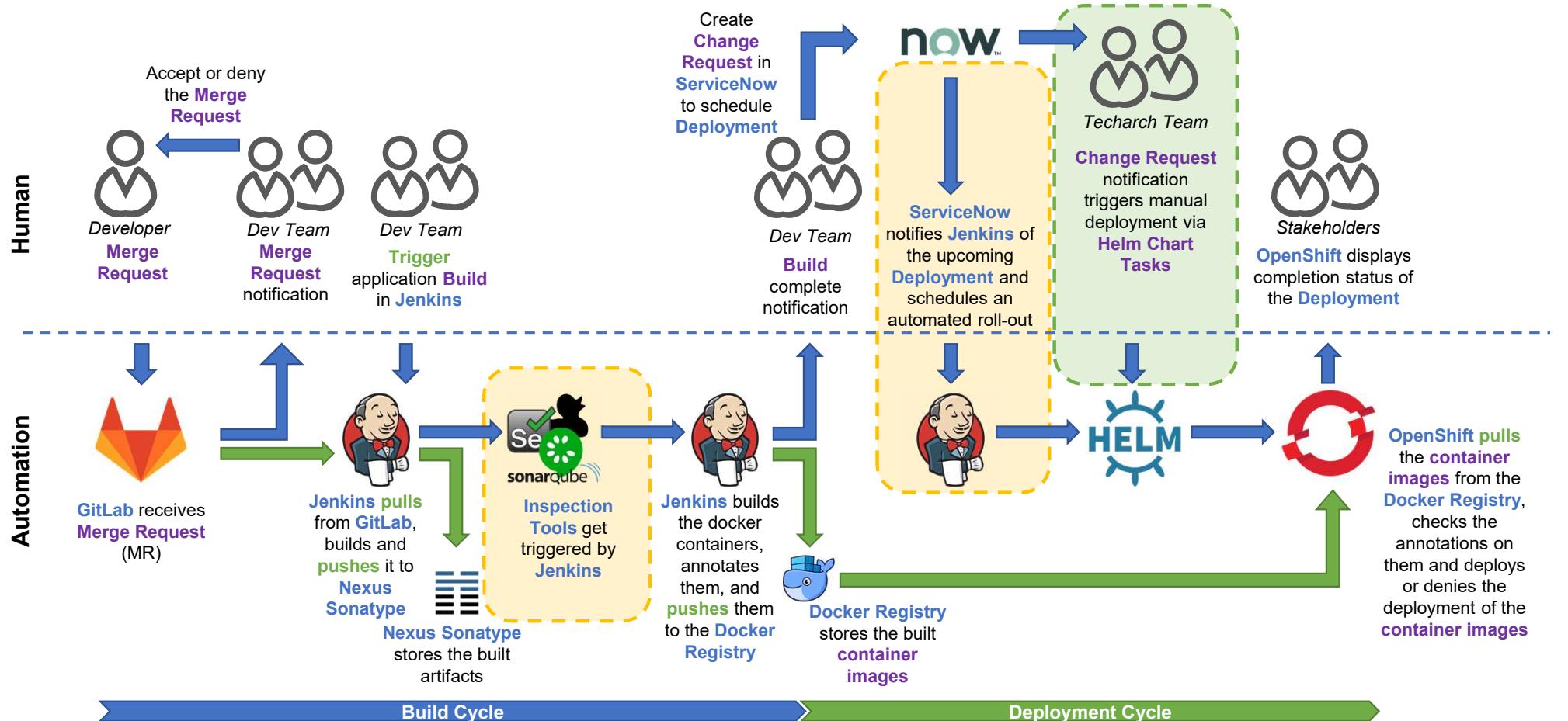
Many integrations with on-premise applications, singular internet connection, central logging and access controls

ORG STRUCTURE



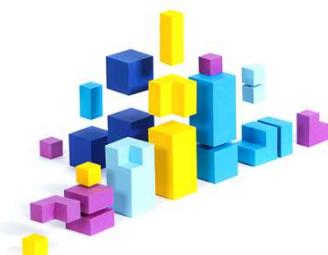
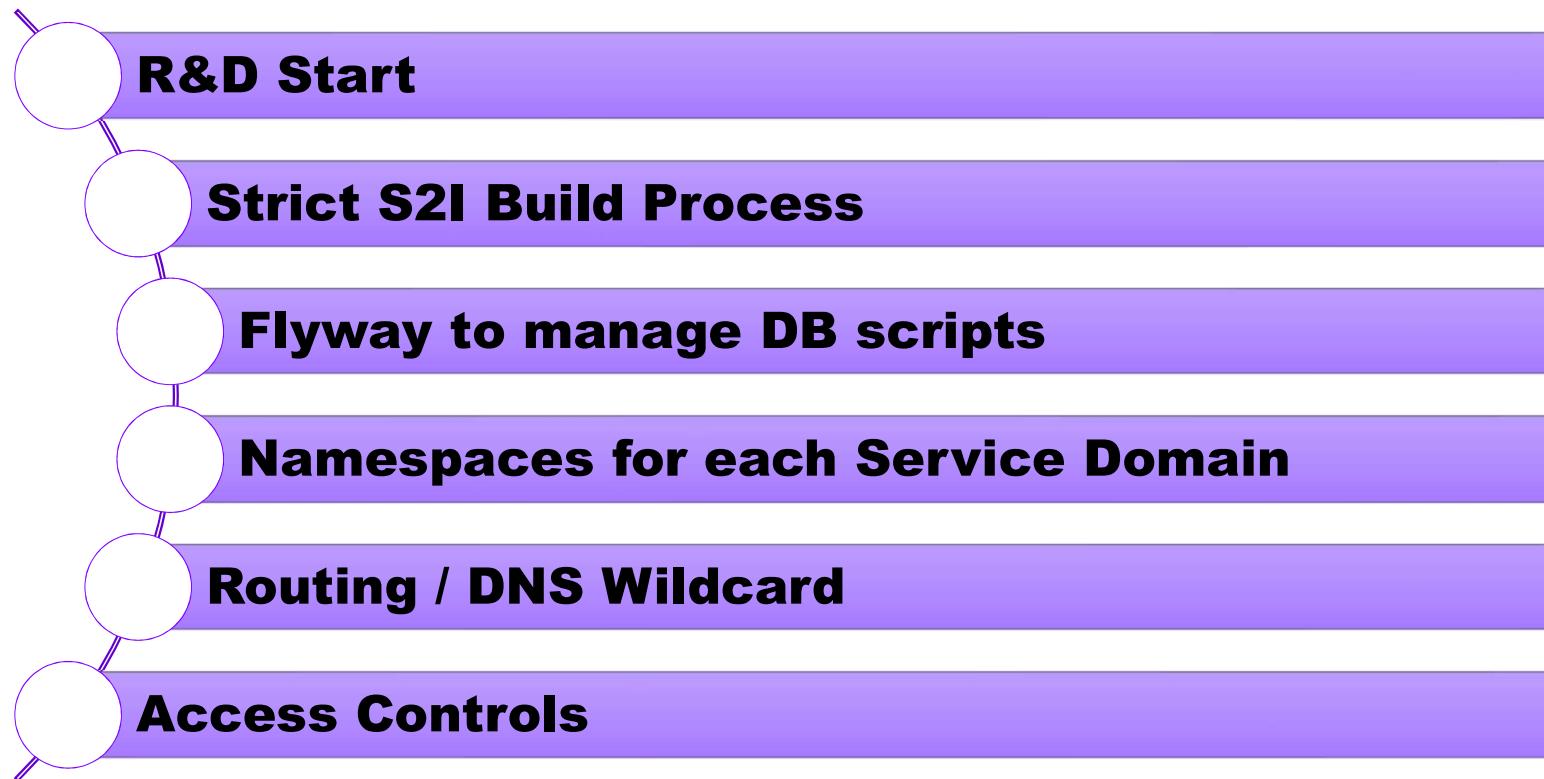
CI/CD PIPELINE

Current
Future

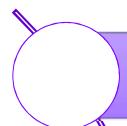


LESSONS LEARNED

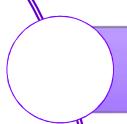
GOT IT RIGHT UP-FRONT



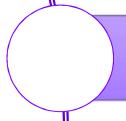
LEARNED QUICKLY



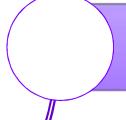
“Hello World” – quotas are important



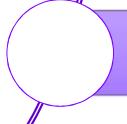
Helm Charts for configurations



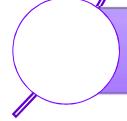
Metrics Plug-in



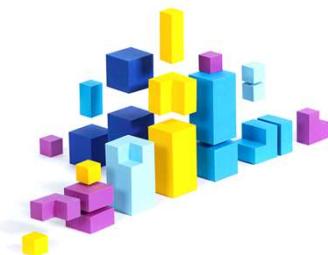
Event-Driven vs Polling



Prod Beta / Staging



Developer Learning Curve – Keep it Lean



AKKA

AKKA IS THE IMPLEMENTATION OF THE ACTOR MODEL ON THE JVM

CONCURRENT & DISTRIBUTED SYSTEMS

Actors & streams to scale
across a cluster

RESILIENT BY DESIGN

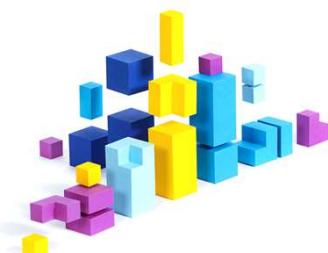
Self-healing systems that
stay responsive in failure
scenarios

HIGHLY PERFORMANT

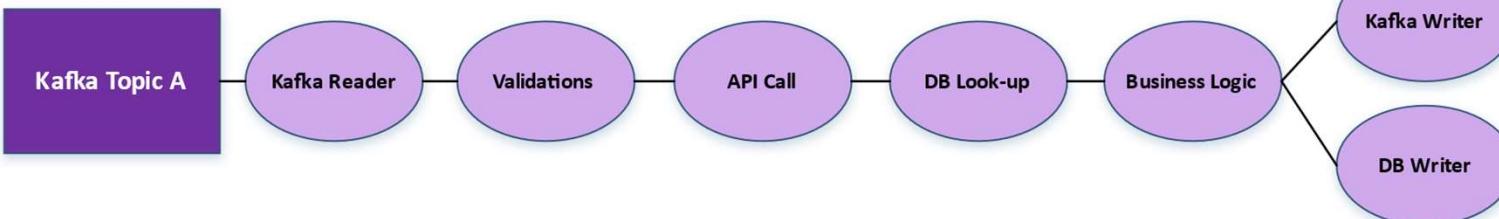
Many small actors with
low memory footprints

ELASTIC & DECENTRALIZED

No single point of failure –
balanced and adaptive
across nodes

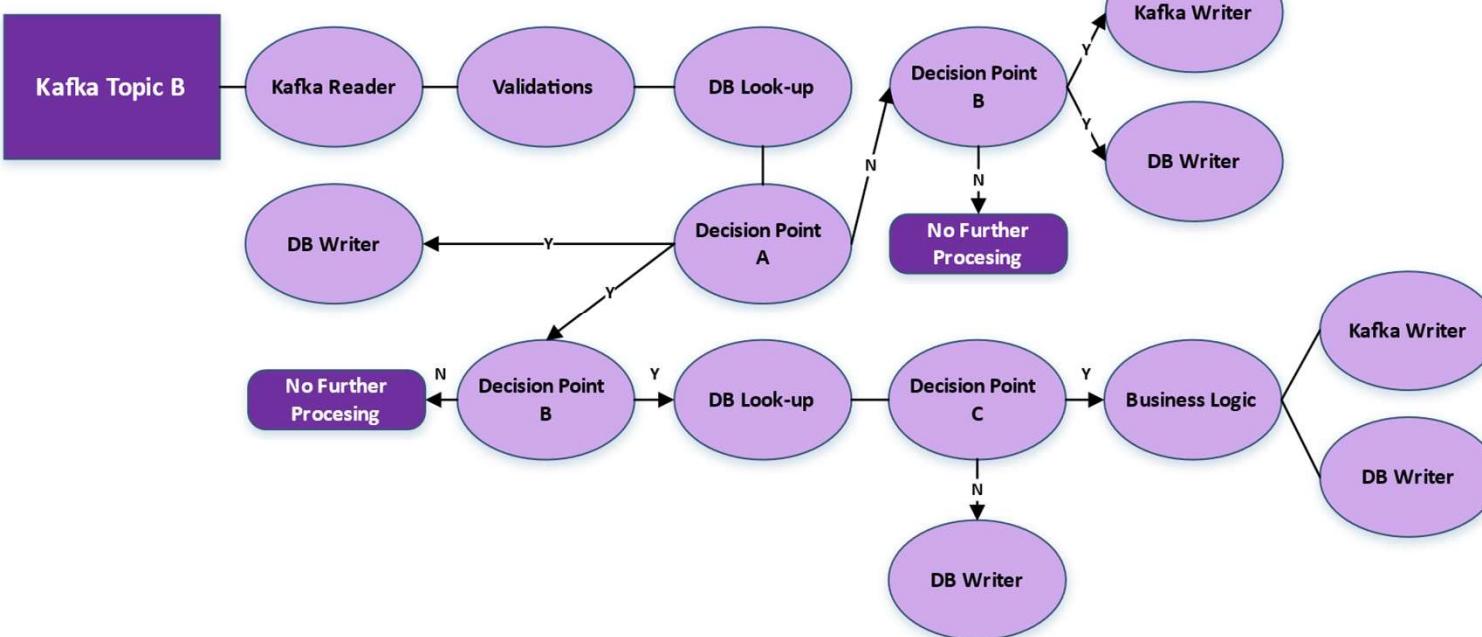


WHY AKKA?



REUSABLE COMPONENTS

Standard actors can be reused across teams – Kafka Reader/Writers, DB Writers, etc.



INDEPENDENTLY SCALABLE

Each of the actors can be scaled independently to handle any bottlenecks within the stream

CIRCUIT BREAKERS

Allows the stream to pause and resume processing as external resources availability changes

LIGHTWEIGHT

Many, small, purposeful components that focus on singular functions

PLATFORM EVOLUTION

MANAGEMENT INSIGHTS

A FEW THINGS YOU HAVE TO DO RIGHT

HAVE A STRATEGY

The lack of a container strategy can be fatal for its introduction. Centralize container architecture and provide solutions (e.g. Source-to-Image)

EDUCATE

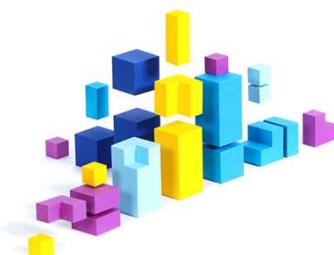
New technologies require new ways of thinking. Educate your stakeholders about both the tech and the benefits early on

PLAN AHEAD

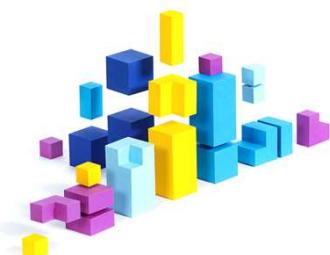
Layout an infrastructure and automation plan to know what challenges are still ahead. Continue to educate

HAVE A VISION

Technologies change and advance in a fast pace. Have a plan for future enhancements and the "bells & whistles"



FUTURE ENHANCEMENTS



KEY REMINDERS & CLOSING



Architecture Pattern



Some Do's & Don'ts



Continue to Evolve

QUESTIONS?



THANK YOU



plus.google.com/+RedHat



facebook.com/redhatinc



linkedin.com/company/red-hat



twitter.com/redhat



youtube.com/user/RedHatVideos