HIGH VOLUME, SECURE TRANSACTION SYSTEMS ON OPENS SHIFT

SOLUTIONS & CHALLENGES
# Table of Contents

<table>
<thead>
<tr>
<th>Section</th>
</tr>
</thead>
<tbody>
<tr>
<td>Program Overview</td>
</tr>
<tr>
<td>Lessons Learned</td>
</tr>
<tr>
<td>Akka</td>
</tr>
<tr>
<td>Platform Evolution</td>
</tr>
</tbody>
</table>
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
PROCESSING GOALS
SCALE & NON-FUNCTIONAL REQUIREMENTS

<table>
<thead>
<tr>
<th>HIGH-VOLUME</th>
<th>NEAR REAL-TIME</th>
<th>ENVIRONMENTS</th>
<th>INTEGRATIONS</th>
</tr>
</thead>
<tbody>
<tr>
<td>20-50M raw scan events per day with spikes both daily and seasonally</td>
<td>Posting payments and generating customer data pushes with minute-level frequency</td>
<td>8 environments each with 10 separate service domains each with dedicated databases and namespaces</td>
<td>Many integrations with on-premise applications, singular internet connection, central logging and access controls</td>
</tr>
</tbody>
</table>
CI/CD PIPELINE

**Human**
- Developer
  - Merge Request
- Dev Team
  - Merge Request notification
  - Trigger application Build in Jenkins
- Dev Team
  - Accept or deny the Merge Request

**Automation**
- GitLab receives Merge Request (MR)
- Jenkins pulls from GitLab, builds and pushes it to Nexus Sonatype
- Nexus Sonatype stores the built artifacts
- Inspection Tools get triggered by Jenkins
- Jenkins builds the docker containers, annotates them, and pushes them to the Docker Registry
- Docker Registry stores the built container images
- OpenShift pulls the container images from the Docker Registry, checks the annotations on them and deploys or denies the deployment of the container images

**Build Cycle**
- ServiceNow notifies Jenkins of the upcoming Deployment and schedules an automated roll-out
- OpenShift displays completion status of the Deployment

**Deployment Cycle**
- Change Request notification triggers manual deployment via Helm Chart Tasks
- Stakeholders
  - OpenShift
- Techarch Team
  - ServiceNow
  - Jenkins
  - Dev Team
  - Jenkins completes notification

**Current**
- Accept or deny the Merge Request

**Future**
- Create Change Request in ServiceNow to schedule Deployment
- Stakeholders
  - Change Request notification
  - ServiceNow notifies Jenkins of the upcoming Deployment and schedules an automated roll-out
- Techarch Team
  - Change Request
  - Jenkins builds the docker containers, annotates them, and pushes them to the Docker Registry
- Nexus Sonatype
  - stores the built artifacts
- Inspection Tools get triggered by Jenkins
- Jenkins pulls from GitLab, builds and pushes it to Nexus Sonatype
- GitLab receives Merge Request (MR)

**Dev Team**
- Merge Request notification
- Trigger application Build in Jenkins
- Accept or deny the Merge Request

**Stakeholders**
- Merge Request notification
- Accept or deny the Merge Request
- Trigger application Build in Jenkins

**Inspection Tools** get triggered by Jenkins

**OpenShift**
- displays completion status of the Deployment

**ServiceNow**
- Change Request notification
- triggers manual deployment via Helm Chart Tasks

**Jenkins**
- builds the docker containers, annotates them, and pushes them to the Docker Registry
- stores the built container images

**Docker Registry**
- stores the built container images

**Nexus Sonatype**
- stores the built artifacts
LESSONS LEARNED
GOT IT RIGHT UP-FRONT

- R&D Start
- Strict S2I Build Process
- Flyway to manage DB scripts
- Namespaces for each Service Domain
- Routing / DNS Wildcard
- Access Controls
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 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/Writer, 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

- Management Cluster
- Security Tagging / Signage
- Federated Cluster
- Additional Logging (EFK)
- ServiceNow / DevOps
- Explore Strimzi
KEY REMINDERS & CLOSING

- Architecture Pattern
- Some Do's & Don'ts
- Continue to Evolve
QUESTIONS?
THANK YOU

plus.google.com/+RedHat
linkedin.com/company/red-hat
youtube.com/user/RedHatVideos
facebook.com/redhatinc
twitter.com/redhat