TWO SIGMA

www.twosigma.com

# Towards Trading on Kubernetes: Operating Multi-Tenant and Secure Clusters

- Javier Diaz-Montes and Andrzej (Andrew) Kochut

December 12, 2018

# Disclaimer

This document is being distributed for informational and educational purposes only and is not an offer to sell or the solicitation of an offer to buy any securities or other instruments. The information contained herein is not intended to provide, and should not be relied upon for, investment advice.   The views expressed herein are not necessarily the views of Two Sigma Investments, LP or any of its affiliates (collectively, “Two Sigma”).  Such views reflect the assumptions of the author(s) of the document and are subject to change without notice. The document may employ data derived from third-party sources. No representation is made by Two Sigma as to the accuracy of such information and the use of such information in no way implies an endorsement of the source of such information or its validity.

The copyrights and/or trademarks in some of the images, logos or other material used herein may be owned by entities other than Two Sigma. If so, such copyrights and/or trademarks are most likely owned by the entity that created the material and are used purely for identification and comment as fair use under international copyright and/or trademark laws. Use of such image, copyright or trademark does not imply any association with such organization (or endorsement of such organization) by Two Sigma, nor vice versa.

# Two Sigma at a Glance

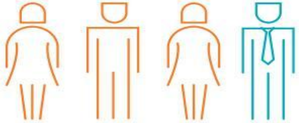
- Systematic investment manager  founded by a computer scientist and a mathematician

- Now 17 years old, 1200+ people

2/3

employees in research and development

72%



non-financial background

- Our mission: discover value in the world’s data

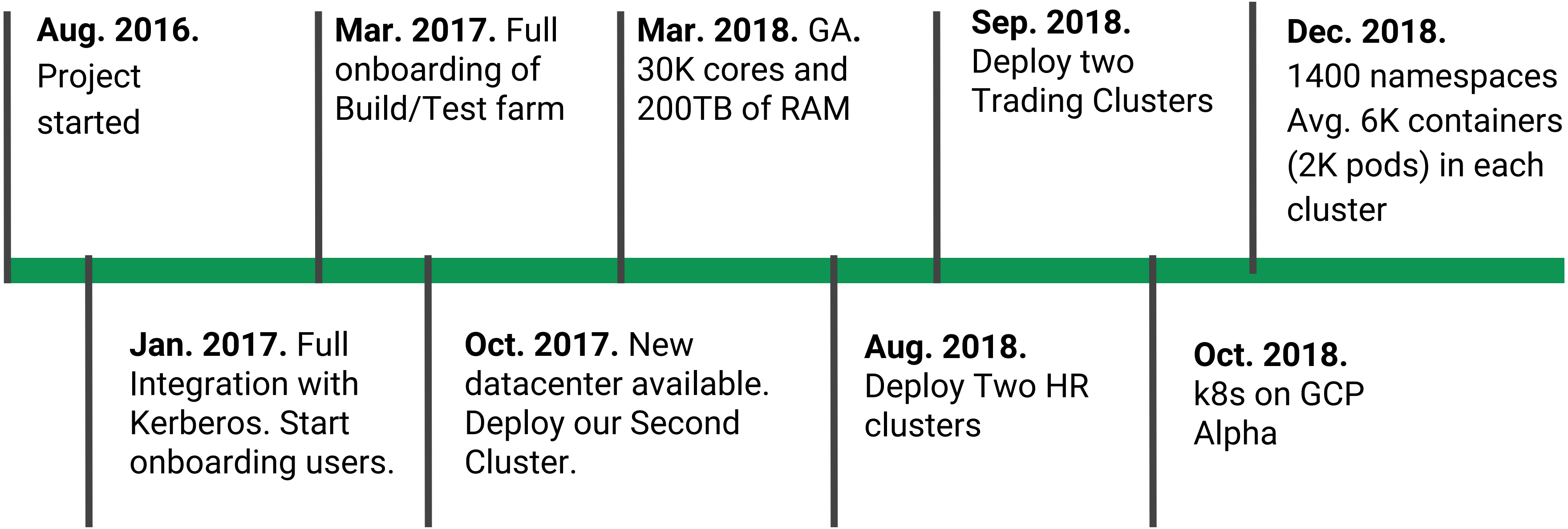
# Introduction

- Two Sigma, as a financial sector company, faces stringent security, resiliency, and isolation requirements
- Concerns such as data exfil by bad actors, as well as that we operate in highly regulated industry, are often a higher priority than for many other industries
- At the same time, we must support a diverse set of applications to perform large-scale data processing and complex mathematical modeling

# Why Kubernetes? Challenges of Private Cloud

- Few cloud-native applications
  - Lack of automation, test automation, and planning for failure
  - Hard to scale horizontally
- VM sprawl
  - Better than bare-metal, but still inefficient utilization
  - High management overhead
- Inconsistent environments
  - Hand-crafted, one-off machines (snowflaky)
  - Dev vs. QA vs. Prod

# Our Kubernetes Journey



**Aug. 2016.** Project started

**Jan. 2017.** Full Integration with Kerberos. Start onboarding users.

**Mar. 2017.** Full onboarding of Build/Test farm

**Oct. 2017.** New datacenter available. Deploy our Second Cluster.

**Mar. 2018.** GA. 30K cores and 200TB of RAM

**Aug. 2018.** Deploy Two HR clusters

**Sep. 2018.** Deploy two Trading Clusters

**Oct. 2018.** k8s on GCP Alpha

**Dec. 2018.** 1400 namespaces Avg. 6K containers (2K pods) in each cluster

# Fitting in Two Sigma Environment

- Stringent security constraints
  - No general access to *root* -- even in containers
  - Users can only use specific role accounts

**Entity**

| | |
|---|---|
| Entity Type: | unix_role_account |
| Unix_login: | tsfoo |
| Unix_uid: | 1234 |
| Unix_gid: | 1111 |

**Relationships**

| | |
|---|---|
| • Can be logged into by: | andrew |
| • Can be logged into by: | javier |
| • Member of group: | tsbar |
| • Member of group: | tsbaz |

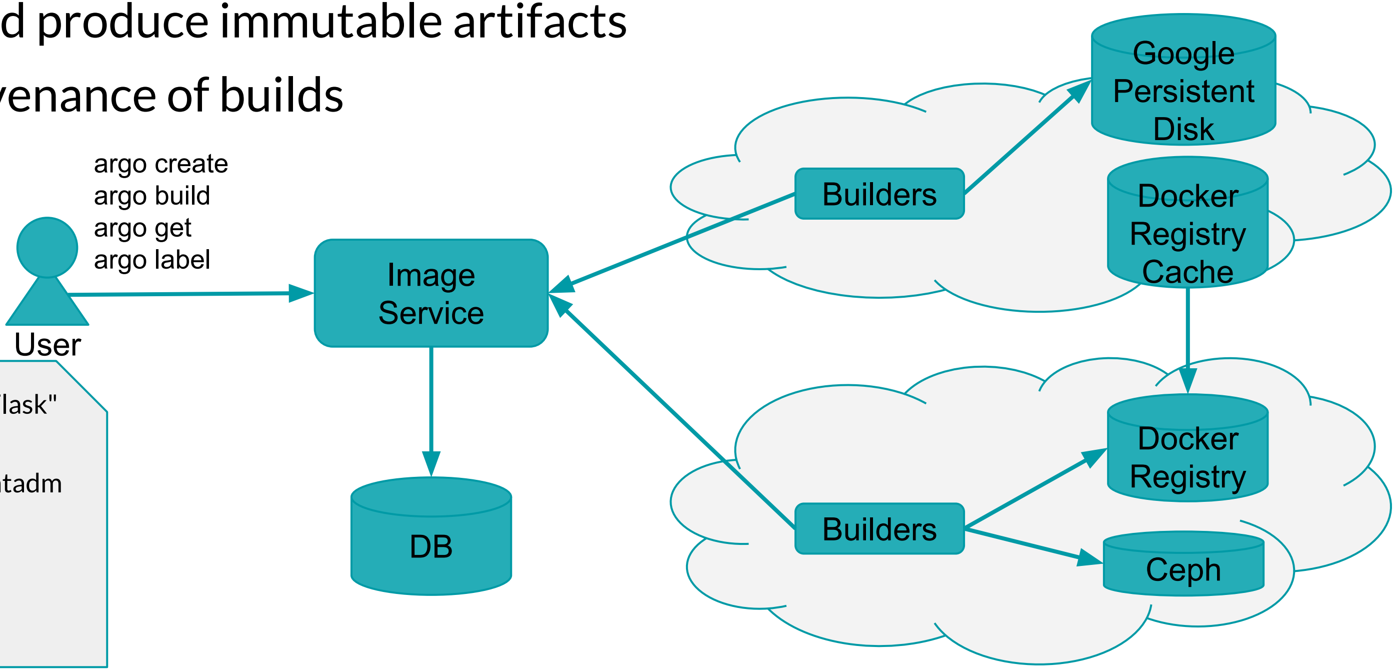
- Large number of heterogeneous applications of limited size

- Multi-tenant cluster integrated with Two Sigma’s entitlements

- Large size of build artifacts not compatible with docker

# Kubernetes at Two Sigma: Building Containers

- Define container image specification (OS, packages, and software artifacts)
- Build it and produce immutable artifacts
- Keep provenance of builds



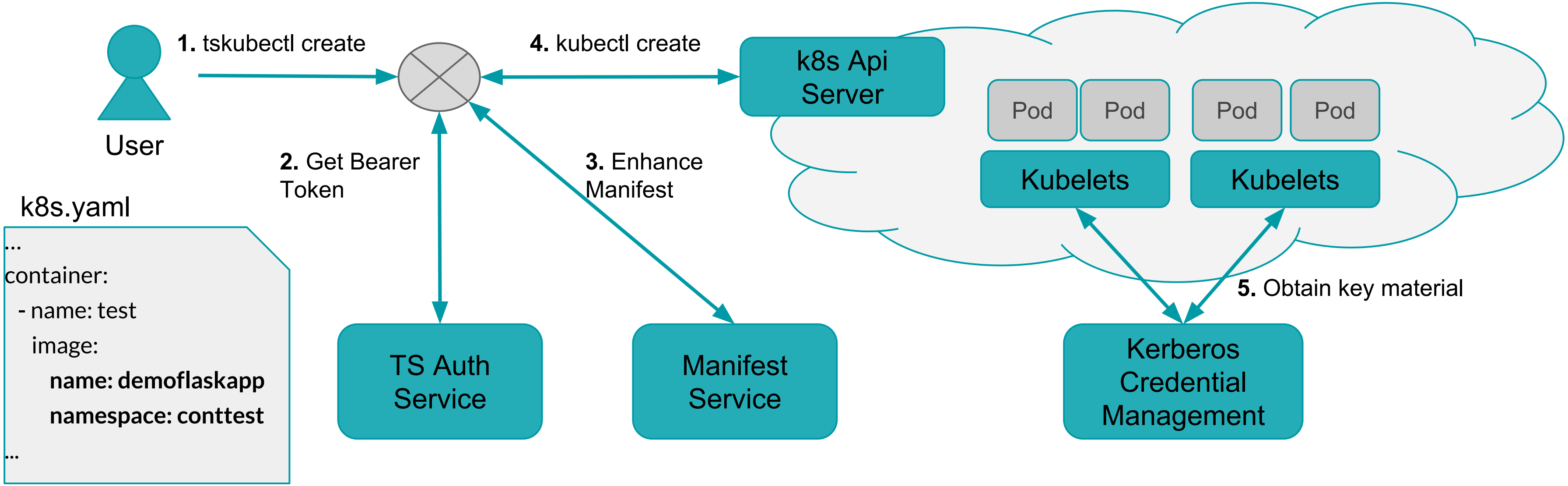
```
argo create
argo build
argo get
argo label
```

spec.yaml

```
description: "HelloWorld Flask"
base_spec_name: stretch
base_spec_namespace: contadm
deployments:
 - k8s-quickstart
name: demoflaskapp
namespace: conttest
```

# Kubernetes at Two Sigma: Running Containers

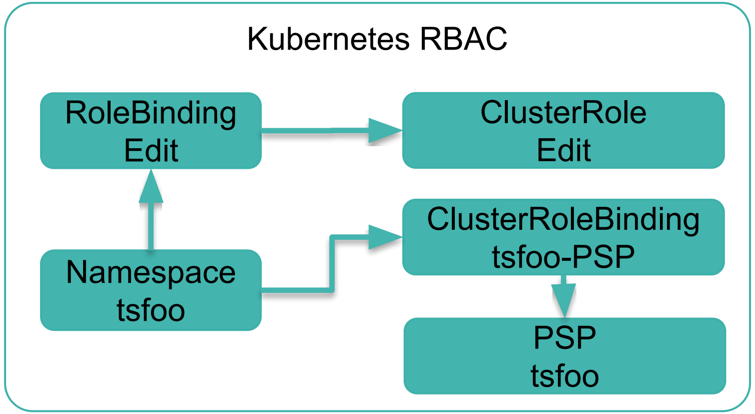
- Service to enhance Kubernetes users' manifests
- Enterprise integrations



```
...
container:
  - name: test
    image:
      name: demoflaskapp
      namespace: conttest
...
```

# Enforcing User Identities in Kubernetes

- Leverage RBAC rules to control user's identities in Kubernetes
  - Associate each user (UID) with a namespace
  - PSP to enforce each namespace can only use its assigned UID and specific supplemental groups
  - Admission controller to enforce runAsUser field in Security Context
- Automatically Synchronize Kubernetes state with our Corporate Identity System



Kubernetes RBAC

RoleBinding Edit

ClusterRole Edit

Namespace tsfoo

ClusterRoleBinding tsfoo-PSP

PSP tsfoo

**Entity**

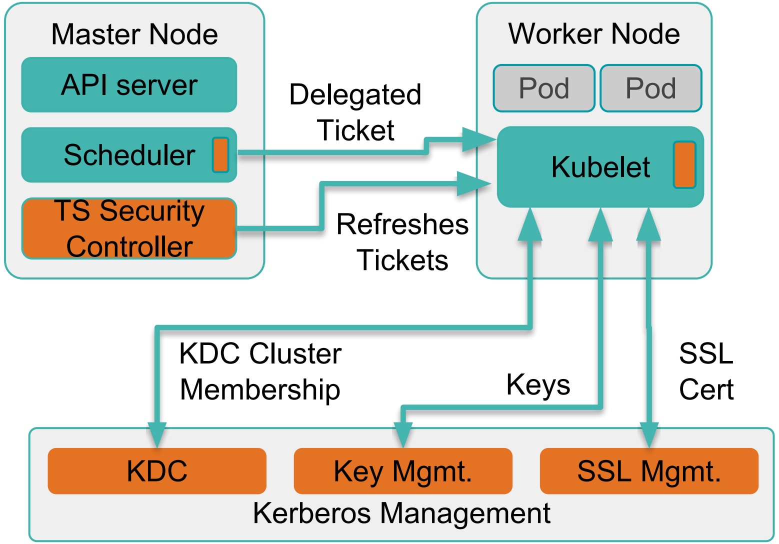
| | |
|---|---|
| Entity Type: | unix_role_account |
| Unix_login: | tsfoo |
| Unix_uid: | 1234 |
| Unix_gid: | 1111 |

**Relationships**

| | |
|---|---|
| • Can be logged into by: | Andrew |
| • Can be logged into by: | Javier |
| • Member of group: | tsbar |
| • Member of group: | tsbaz |

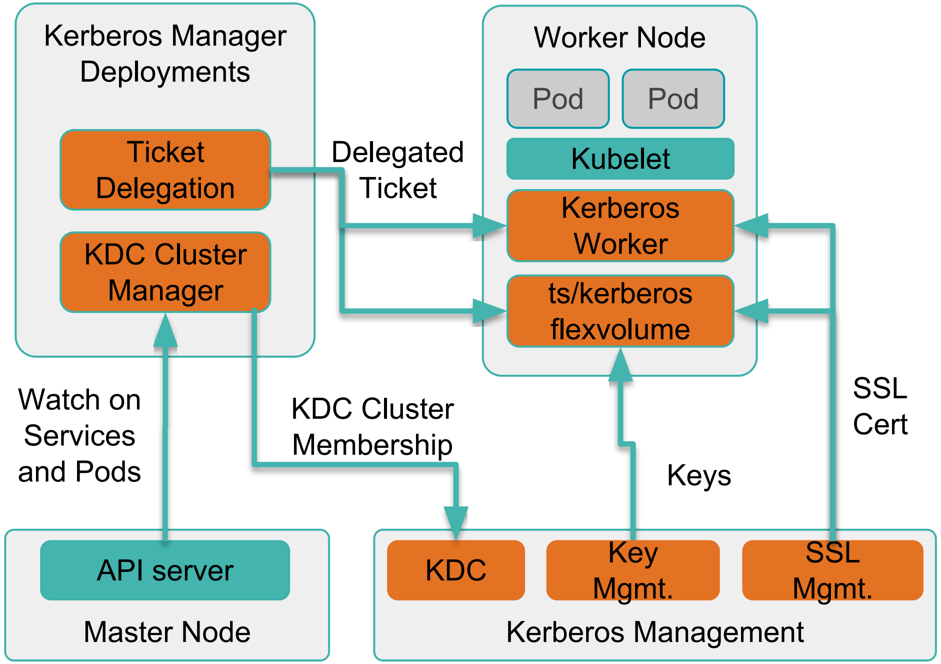
# Kerberos Credential Management - 1st attempt

- User request key material via custom annotations
  - ts/services, ts/prestashtxt, ts/certs
- Challenges:
  - Fork encumbers upgrading Kubernetes
  - Security of  workers managing KDC
  - Negotiating keys when worker nodes go down without deregistering themselves
  - Kerberos infrastructure struggles to keep up with request rate



# Kerberos Credential Management - 2nd attempt

- Create central controller outside of Kubernetes code
- Externalized using flexvolume
- Centralized key distribution mechanisms
- Considering Open Source Kerberos - Kubernetes integration framework



# State of the World

- Total Capacity: ~30k cores, 200TB memory
- Examples:

**Critical Infrastructure**

- Build and test farm
- Integration Testing
- Distributed Memory Cache

**Trading Related Services**

- Post trade data transformation processes
- Event System services

**Burstable Modelling Workloads**

- Interactive Spark Driver
- Modelling workflow
- News translation service

**Cloud Native Microservices**

- People Data Platform
- QA services for Back Office and trading

# Use Case - Build and Test Farm

Runs in containers on K8s for every build and pre-push test in the company

**Metrics:**

- 250k+ builds per day
- 3 million average, 9 million peak tests per day
- 11 TB of artifacts served per day

**Benefits:**

- Machine maintenance was a pain, no longer
- Used to run multiple threads, now use isolation to run one thread / container
- Able to easily run different architectures (Debian Wheezy and Stretch)
- No concerns about scaling during peak times

# Use Case: Continuous Integration with Jenkins

A large central Jenkins for general integration tests of our monorepo, with per team dedicated instances for custom pipelines

***Metrics:***

- Around 5K pipelines and an average of 30K tests per day

**Benefits:**

- Easy to deploy and manage

**Challenges**

- Jenkins-Kubernetes driver generally works, but tries to DoS API Server
- Jenkins manages state using tiny files in disk, does not work well on NFS
- Persistence maintained in Ceph images, provisioned using Kubernetes PVC
    - Mounting persistent volumes using RBD driver is unreliable and has timeouts

# Use Case - news translation

The news team uses Kubernetes to translate foreign language news and documents into English, to be sourced for research

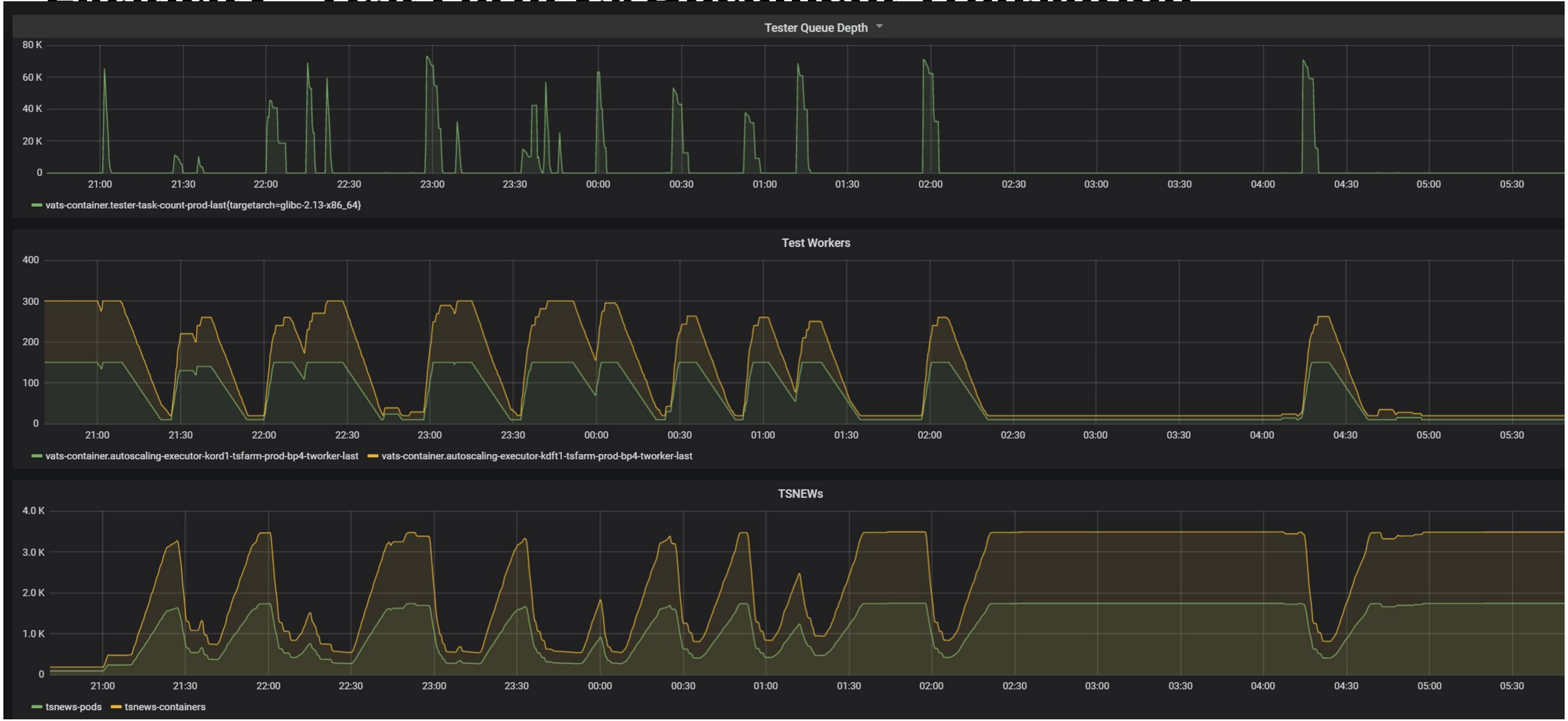
**Metrics:**

- 15 years of data
- 300M documents in 5 different languages

**Benefits:**

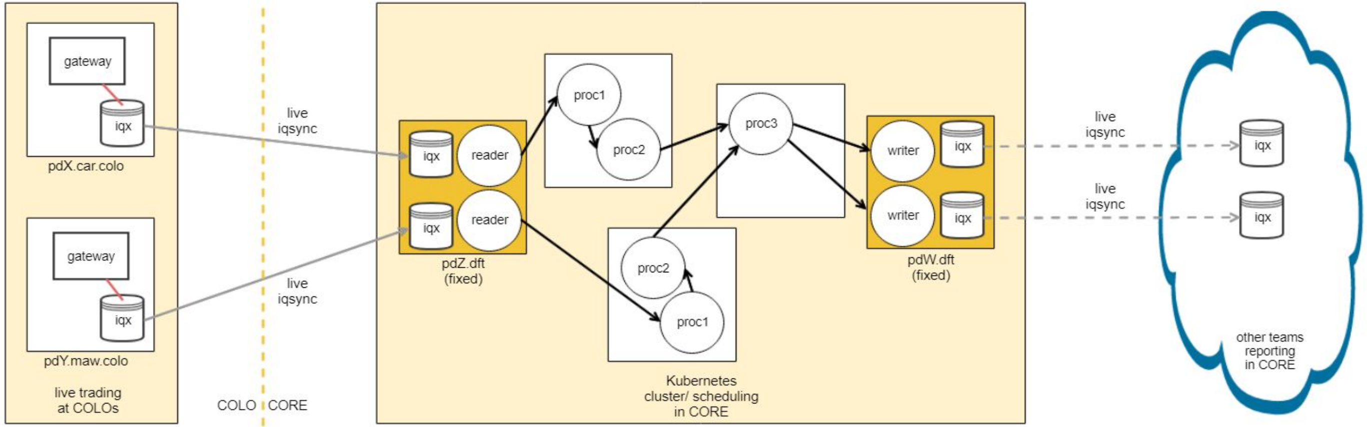
- Running in a container with vendor image by the translation service
- Leverages elasticity of the Kubernetes platform

# Elasticity - Test Farm vs. Document Translation



# Use Case: Trading

- Process post trading data using custom workflow stream engine
- Kubernetes allows easy resource management
- Performance comparable to baremetal
  - Jobs with latency of 100ms from event landed to all relevant reports completed
  - Throughput of 300k x 1kb msgs/s



# Cultural Change: Towards Cloud Native

- Introduced Chaos engineering by default in our general purpose clusters

- Introduced process to triage non-cloud native applications

- Very successful initiative with very few exceptions (e.g., jenkins, Postgres)

# Lessons Learned: Operations

- Building “well-behaved” applications using Kubernetes is not trivial
- Kubernetes has proven to be very resilient and scalable with little effort
- Runtime issues: Enterprise integrations, Docker, OS level
- Kubernetes Scheduler is deterministic and a bad worker can cause issues
- Monitoring for early warning
- Active probing
- Upgrading Kubernetes in place

# Ongoing Work

- Self-healing infrastructure
- Moving towards Kubernetes on Public Cloud -- exploring GCP and  GKE
- Extending scheduler to improve quota management and fairness
- Expanding use cases and adopting more mission critical applications

Questions?