# Airflow at high scale for Autonomous Driving

# The Speakers

Philipp Lang

- Solution Architect @ DXC Technology
- Head of Airflow Dev Team for > 2 years
- Background in Astrophysics

Anton Ivanov

- Senior DevOps engineer @ DXC Technology
- 10+ years experience
- Background in infrastructure and system administration

DXC TECHNOLOGY

**$17.7B**
FY21 revenue

**70+**
countries

**130,000+**
employees worldwide

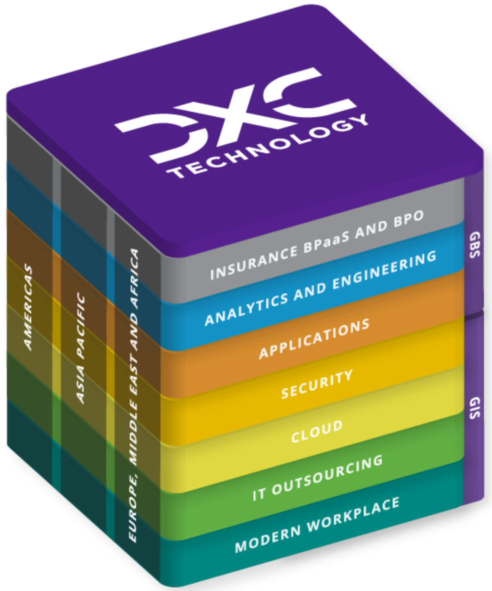
**240+**
240+ customers in the Fortune 500, and leading global companies

**200+**
partners

## Delivering eXcellence for our Customers and Colleagues

DXC is an IT services market leader delivering excellence for our customers and colleagues. We are delivering business impact and are an employer of choice where people want to work and stay.

## Transform your business across the Enterprise Technology Stack



GBS: Global Business Services
GIS: Global Infrastructure Services

We help customers across the globe create a rich workplace experience, simplify and optimize on-premises IT, and achieve a secure, high-performance cloud environment to realize positive business outcomes.

Our services weave cyber resilience throughout the enterprise, help customers reimagine business with transformative applications, and enable data-driven decisions, automation, and state-of-the-art engineering.

DXC business process outsourcing helps customers transform operations to a digital business model.

## Leadership and recognition

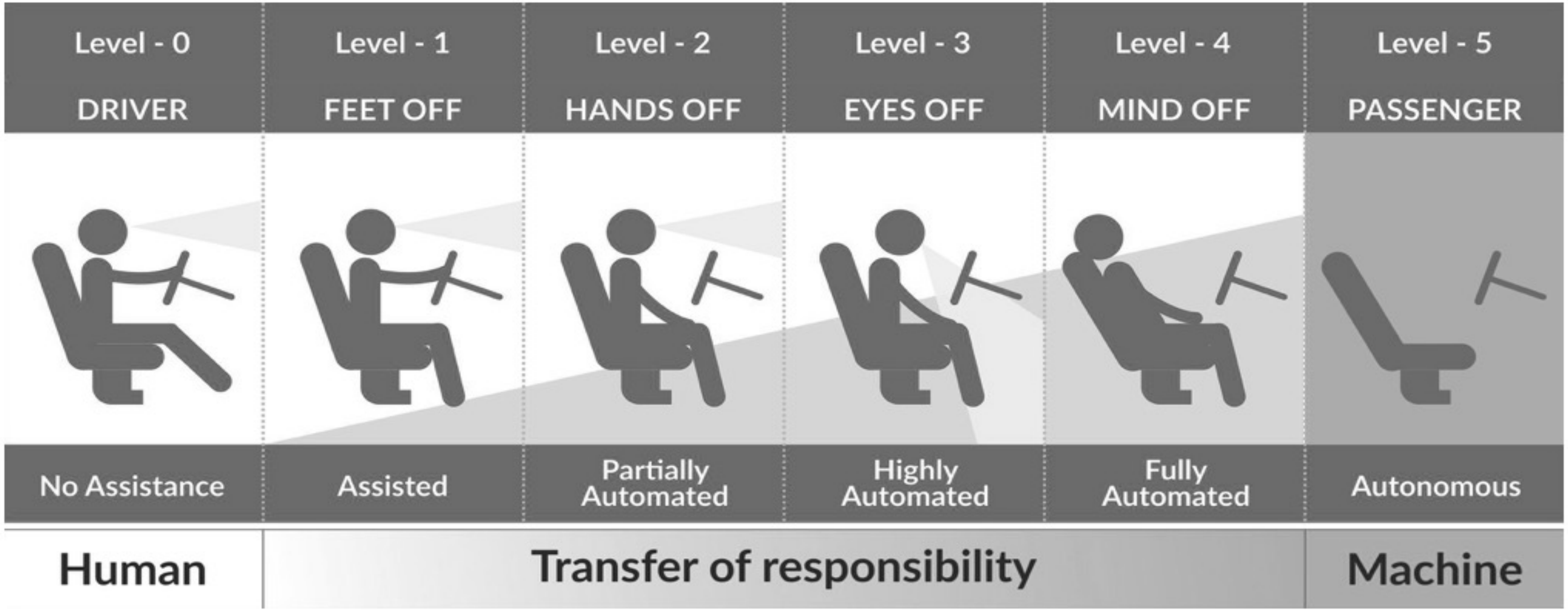
- A Fortune 500 company, No. 152 in 2021 ranking (NYSE: DXC)
- Leader: IDC Worldwide Managed Security Services MarketScape
- Leader: NelsonHall Advanced Digital Workplace Services 2020 (Overall, Run, Build)
- Leader: NelsonHall Infrastructure Brokerage and Orchestration 2020
- Forbes World's Best Employers 2021
- Newsweek's America's Most Responsible Companies 2022
- Barron's 100 Most Sustainable Companies

## Customer stories

- **BMW Group** harvests and manages a daily collection of more than 1,500TB of raw data from vehicle sensors, simplifying insights and reducing time to develop autonomous vehicles.
- **Sabre Corp.** uses outsourcing and IT modernization capabilities to transform technology to help the company unlock additional value across its businesses.
- **Campbell Soup Company** supports operations across infrastructure, applications and security with an IT foundation focused on value-added activities.
- **Lockheed Martin Aeronautics** is implementing its vision for smart factories using a next-generation digital manufacturing execution system and the digital thread framework.

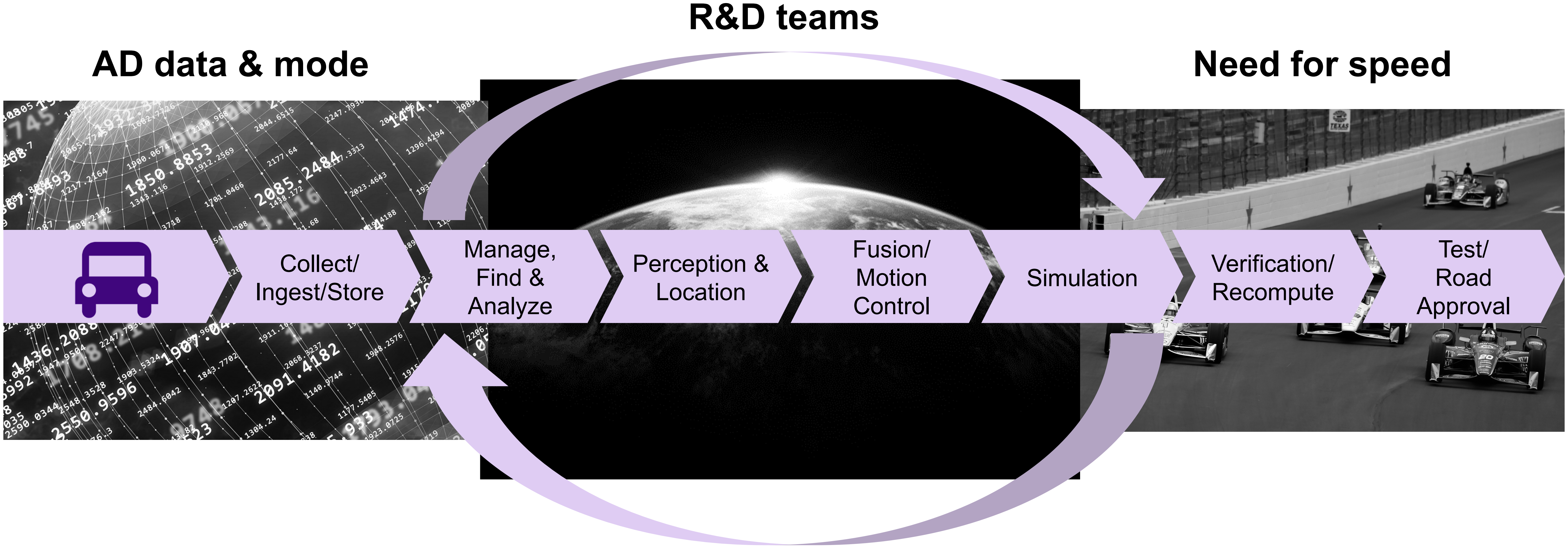
# Data Driven Development for Autonomous Driving

# Levels of Autonomous Driving



| Level - 0 | Level - 1 | Level - 2 | Level - 3 | Level - 4 | Level - 5 |
|---|---|---|---|---|---|
| DRIVER | FEET OFF | HANDS OFF | EYES OFF | MIND OFF | PASSENGER |
| No Assistance | Assisted | Partially Automated | Highly Automated | Fully Automated | Autonomous |
| Human | Transfer of responsibility | | | | Machine |

# Robotic Drive: an End-to-End Data and AI Capability Ecosystem for AD developement



# Airflow on Robotic Drive

# Airflow on Robotic Drive

## Our Use Case

Airflow is used as orchestration layer for a large, multi -Tennant HPC-platform built on Robotic Drive

**Open Source**

**Scalable**

**Customizable**

**Active community**

# Requirements on orchestration I

## Scalability

- (Vertical) scalable to run 1000s of DAGs concurrently
- Volume of jobs on average: > 500,000 Dagruns / month

## Orchestration workloads

- Spark jobs
- K8s pod operator jobs
- Complex DAG dependencies, eg.
  - Trigger of workloads across DAGs and Airflow instances
  - Usage of sensors and TriggerDagRunOperators

# Requirements on orchestration II

## Flexibility

- Implement custom features in code-base and configuration
- Different Airflow instances with individual configuration

## Resiliency and stability

- Fault tolerance against container errors / restarts
- Regular updates with minimal business impact

## Security

- Authorization and Authentication
- Multi-Tenancy

# From POC to Production

## POC Q2/2019

| | |
|---|---|
| Apache Airflow | v1.10.2 |
| Celery | 32 tasks / instance |
| | PostgreSQL 9.4 |
| RabbitMQ | Single instance / queue |
| | Ansible automation |

## Current Production

custom v1.10.10

10,000 tasks / instance

pgbouncer + Crunchy PostgreSQL-HA

RabbitMQ-HA (v3.8.5) 50+ queues

Helm chart with rolling updates HELM

# Airflow – Setup I

## Deployed on OpenShift

- Deployment and updates via helm
- Celery – based scaling
- Several scalable Airflow instances + ad-hoc instances
- Integration of K8s pod Operator

## Integrated with MapR

- Location of deployed DAGs + airflow config
- Spark-job submission to YARN + customized Spark-submit hook

# Airflow – Setup II

## Logging/Monitoring

- **Metrics collection + Monitoring**
  StatsD → Prometheus → Grafana
- **Log collection:** ElasticSearch + Kibana
- **Extensive alerting framework**
- **Customized Airflow logger**

## Security + IAM

- **Authentication + Authorization** through ldap + centralized IAM
- **Token-based authentication** for Airflow's REST API

# Airflow customizations

# Airflow customizations – The Spark submit operator

Use case of Spark submit operator

- Jobs submitted to YARN in “cluster” mode
- Challenge for failed jobs diagnostics: Correlation between Yarn app logs and airflow tasks
- Challenge for scaling up: ”stuck” Spark jobs with higher loads

### 1. Improved logging

- YARN logs with ID and job status visible in Airflow
- Application logs of the Spark job imported as Airflow task log

### 2. Improved scalability and stability

- Solved issue of “stuck runs” through adaption of Spark submit hook
- Improved resiliency towards connection issues from airflow to YARN via timeouts/retries

### 3. Customized Airflow’s Spark Submit Operator

- Extend list of parameters where templating is supported
- Included „properties-file” in operator constructor

# Airflow customizations – Scalability and HA Components

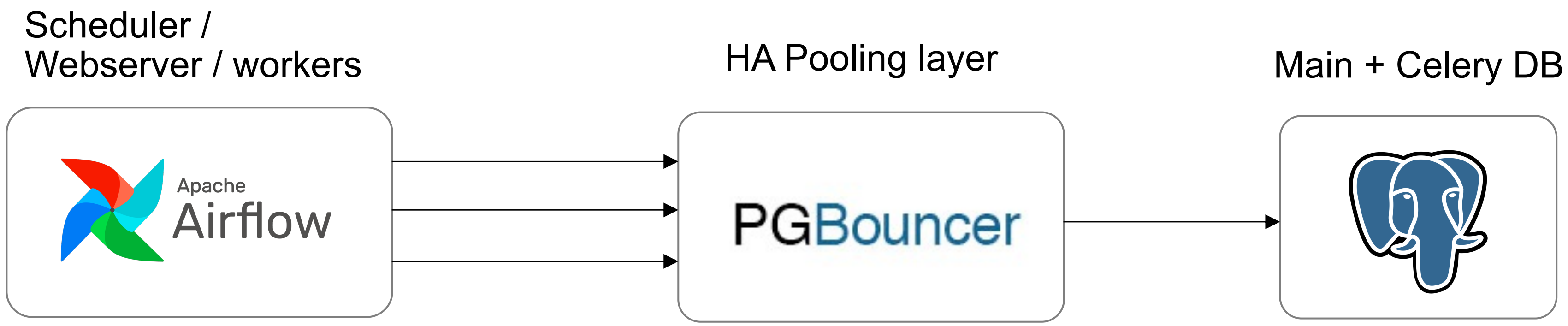
**Bottlenecks & Challenges for vertically scaling a single Airflow instance (10s’ → 10K tasks)**

- Connection from airflow to it’s Main Database + Celery Database
- Airflow scheduler performance
- RabbitMQ limitations for scaling celery nodes
- Complexity and architecture of DAGs

# Airflow customizations – Scalability and HA Components

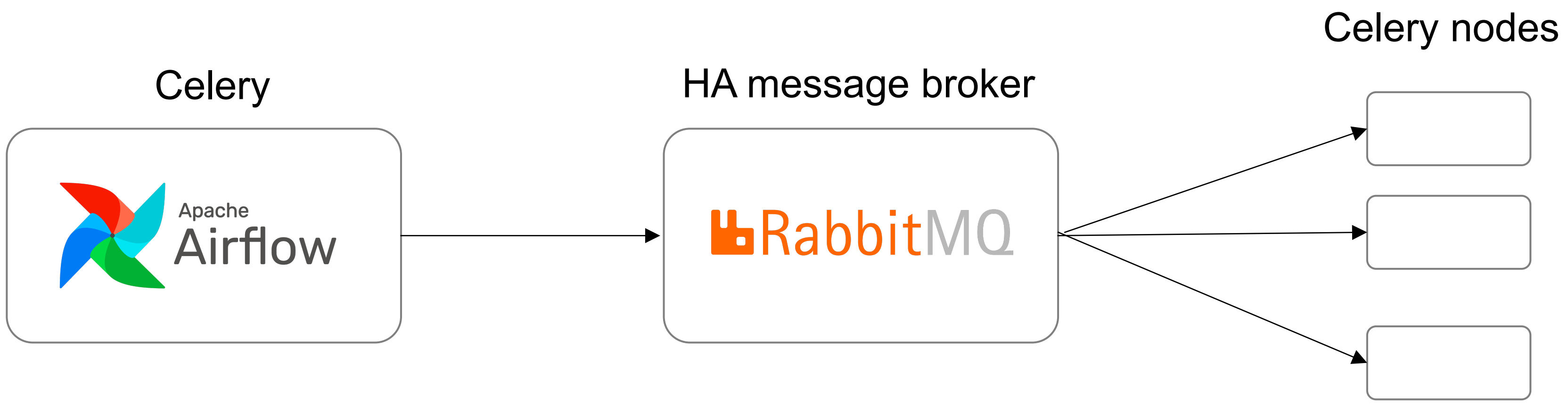
## Airflow – DB connection

- **Connection pooling** using Pgbouncer with high client/server ratio (>10)
- **Optimization of DB internals** & upgrading PostgreSQL (+ Crunchy)
- Optimization of Airflow's **SqlAlchemy**



## RabbitMQ + Celery

- Upgrade to **RabbitMQ – HA**
- **Internal optimizations** (buffer sizes, etc) to improve max. number of celery nodes
- Vertical scaling of celery nodes



# Airflow customizations – Scalability and HA Components

## Scheduler performance

- Optimization of internal **scheduler settings**
- Leverage **multi-processing**
- **Load-balancing** / splitting of complex DAGs

## Other adaptions

- Spark Submit Operator
- Performance of underlying persistent storage
- Fine-grained resource allocation of Airflow queues
- Liveness/Readiness probes

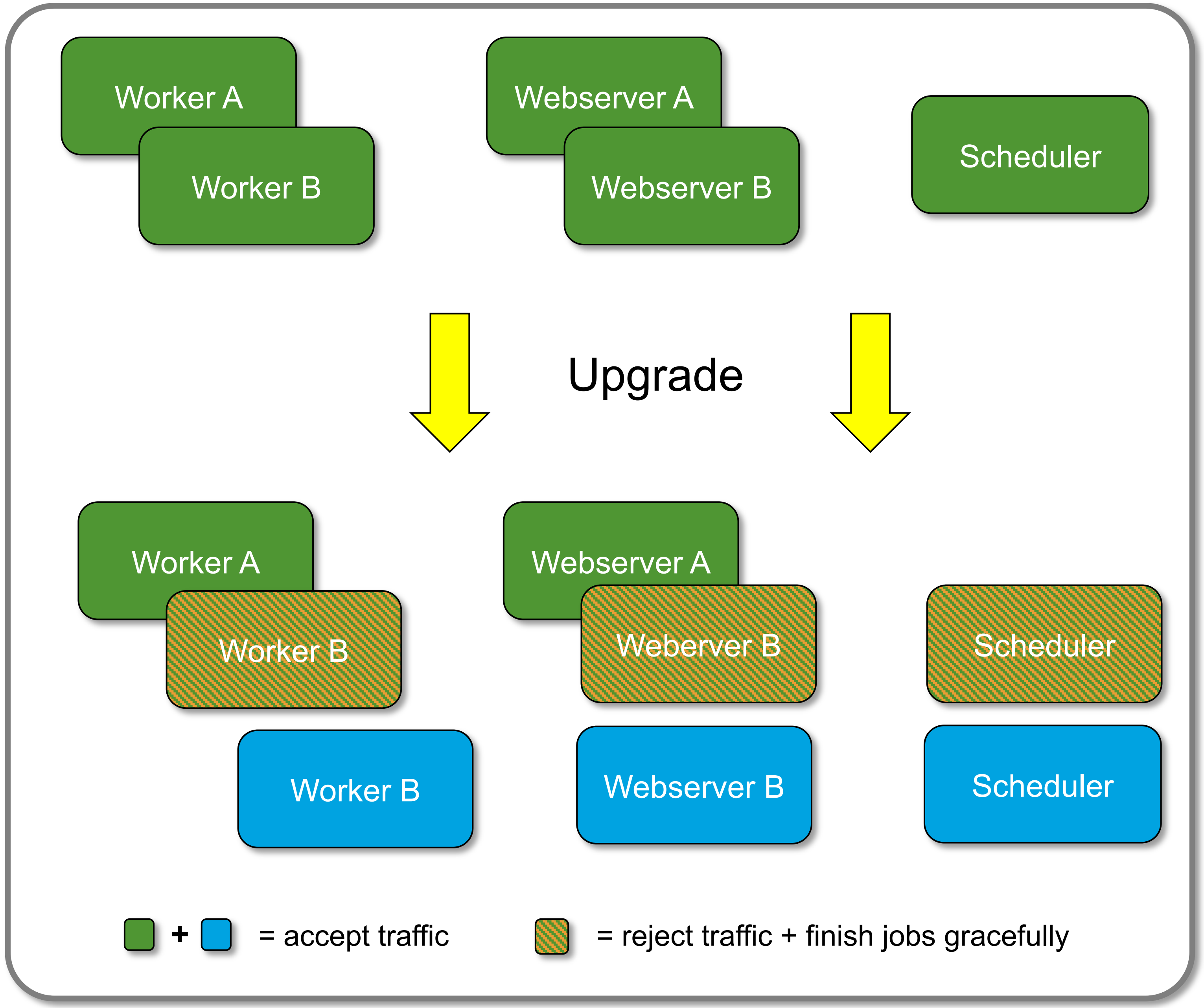
# Airflow customizations – Rolling upgrades

## Requirements

- No service downtime during regular upgrades
- All running jobs must finish gracefully
- Upgrade under heavy load possible

## Implementation

- Helm chart to update airflow’s main components
  - scheduler / webserver / workers / RabbitMQ
- ‘Rolling’ restarts for containers using lifecycle hooks and smart Celery-queue assignments



# Airflow customizations – IAM solution

## IAM Authentication and Authorization for Airflow’s WebUI

- ➢ Log-in + assignment of roles through ldap + centralized IAM
- ➢ Integration with OAuth for token-based authentication

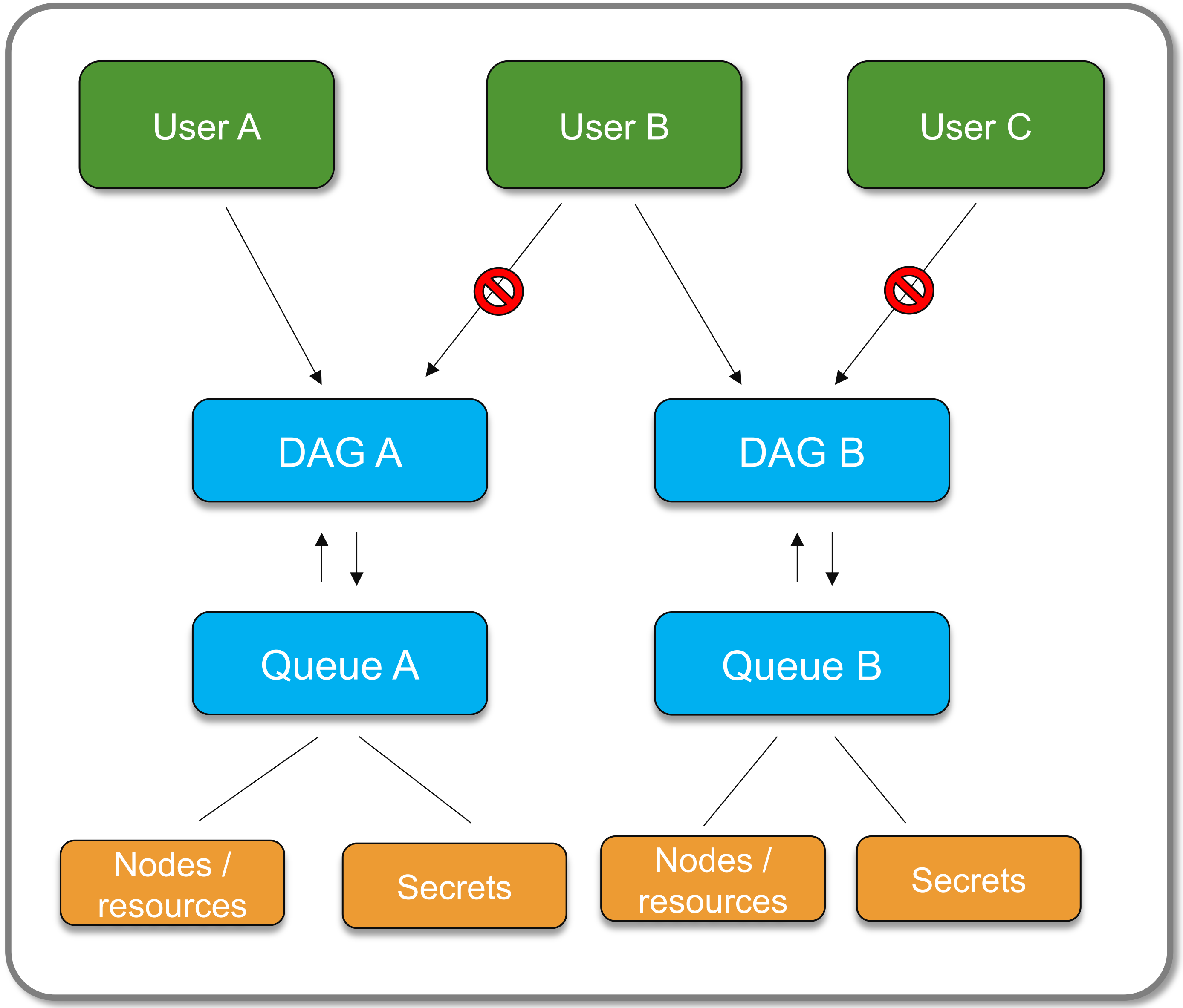
## ... and for Airflow’s REST-API

- ➢ Authorization layer to allow a mapping of users → DAG permissions
- ➢ Token-based authentication for airflow’s REST API

# Airflow customizations – Queue isolation

## Queue isolation → 1 Airflow queue per DAG

- ➢ Each user have access only to own security tokens
- ➢ Users can trigger only their owned DAGs
- ➢ Queue limitation and fine-grained resource allocation

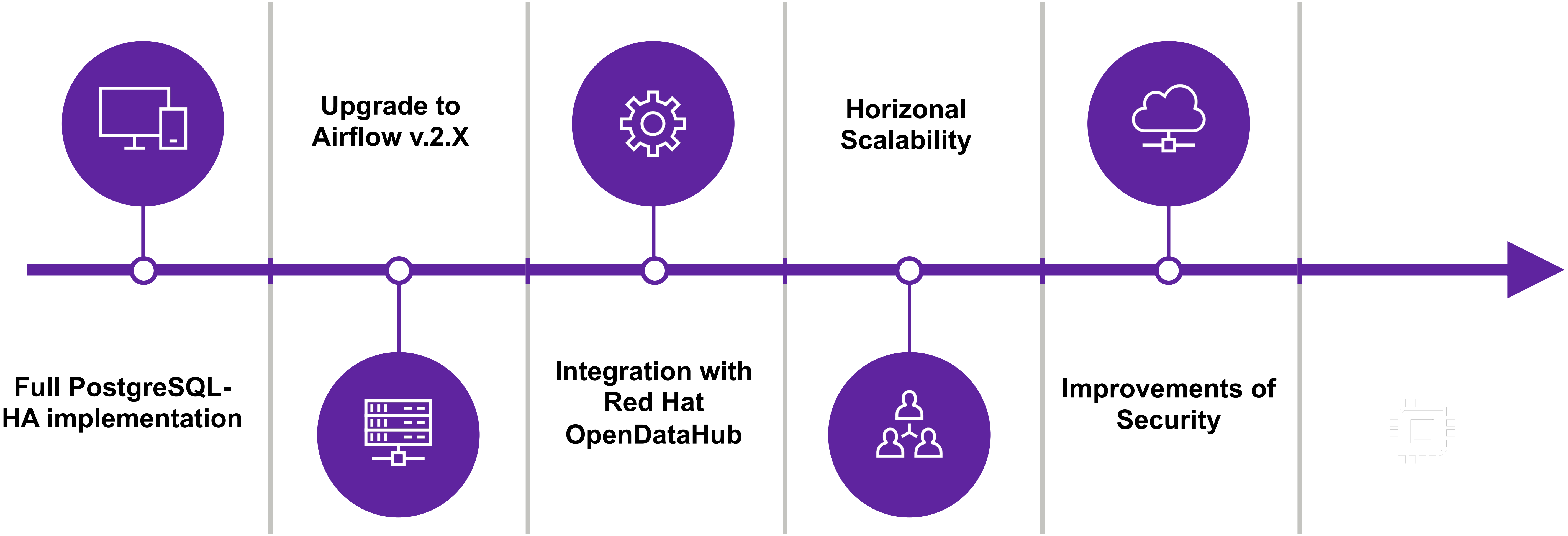


# Airflow customizations – Logging framework

## Adaptions of Airflow’s internal logging mechanism:

- Fully JSON-compatible logs → scrapeable by Elastic
  - Work in progress: Keeping default log format in parallel
- Additional logging fields via templates
- Yarn application logging

# What's next



Full PostgreSQL-HA implementation

Upgrade to Airflow v.2.X

Integration with Red Hat OpenDataHub

Horizonal Scalability

Improvements of Security

# Questions and answers

Philipp Lang – plang20@dxc.com
Anton Ivanov – anton.ivanov@dxc.com