

Advanced Connectivity Workshop

Seamless Business System Integration via APIs



FEATURING

Tim Reblitz

Technical Account Manager, Tulip

AGENDA

01

Ideal Integration Architecture

Real-Time Transactions with
Systems of Record

02

Practical Limitations

Platform Constraints to Consider in
Integration Architecture

03

Other Integration Practices

- iPaaS
- Asynchronous Integration Methods

04

Don'ts from Past Experience

05

Upcoming Best Practices

...what's your Function?
Secret Agent Man

Let's make this a
discussion, *not a lecture.*

Scope of Discussion:

Integration with
Transactional Business Systems
(e.g. ERPs)

Ideal Integration Architecture

Real-Time Transactions with Systems of Record

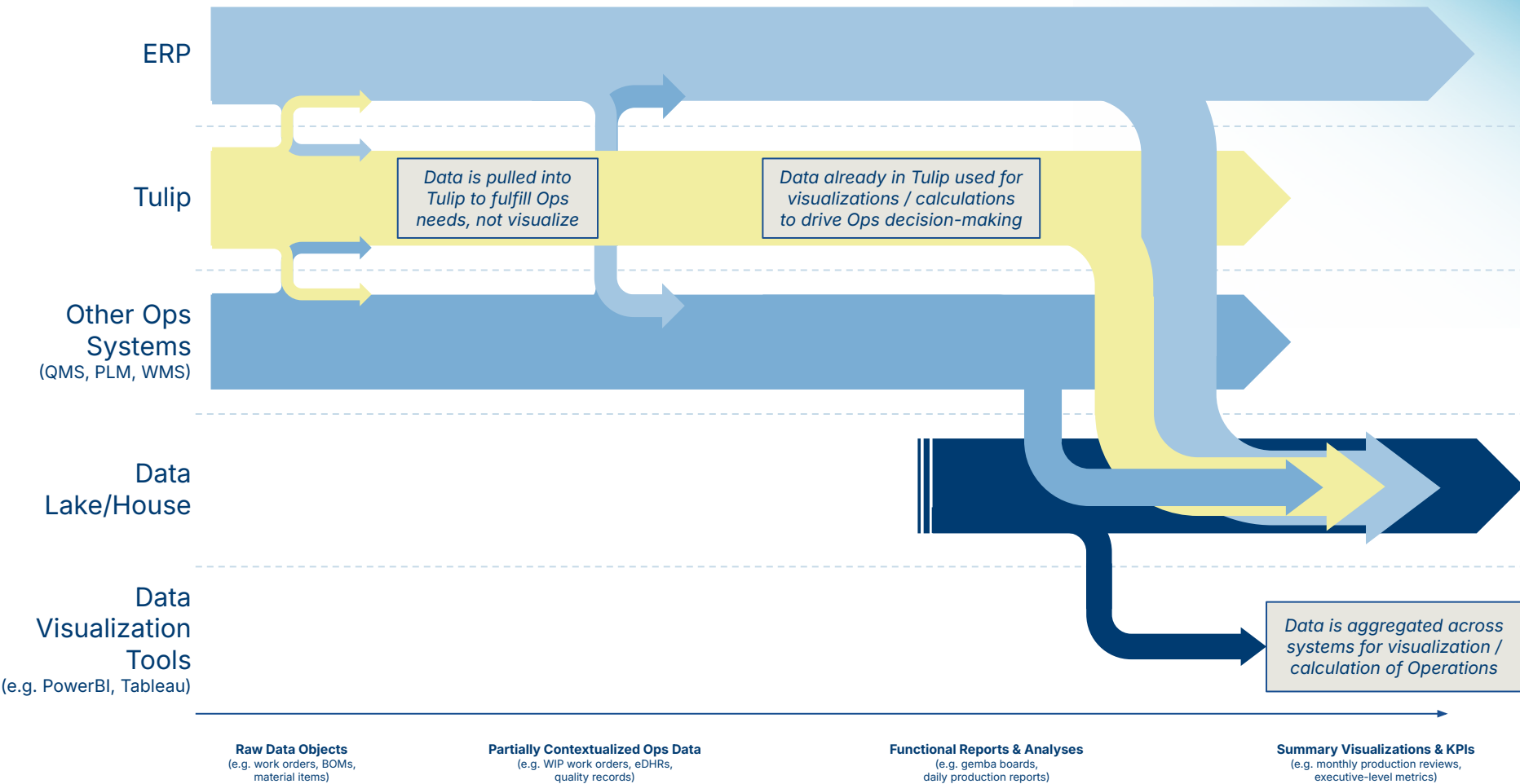


The 'Ideal' Best Practice

The 'Ideal' Best Practice

- **Maintain a single Source of Truth / System of Record.**
- **Assumptions**
 - The Source of Truth has the latest/greatest content.
 - The Source of Truth can be integrated via HTTP REST APIs
- **'Ideal' Best Practice**
 - **Integrate with transactional Sources of Truth in real-time via HTTP Connector Functions**

Tulip's position within an Enterprise Data Flow



Connectors

Preferred practice over SQL Connector.

HTTP Connector

Connect to any HTTP service

Retrieve data as JSON or XML from your HTTP services

(Example Transactional Systems)

ORACLE
NETSUITE

 Dynamics 365

Logiware
NiceLabel

SAP S/4 HANA

 BarTender®
BY SEAGULL SCIENTIFIC

SQL Connector

Connect MS SQL Server, PostgreSQL, Oracle DB, MySQL

Use parameterized queries or stored procedures to:
Select, Insert, and Update

ORACLE®
DATABASE

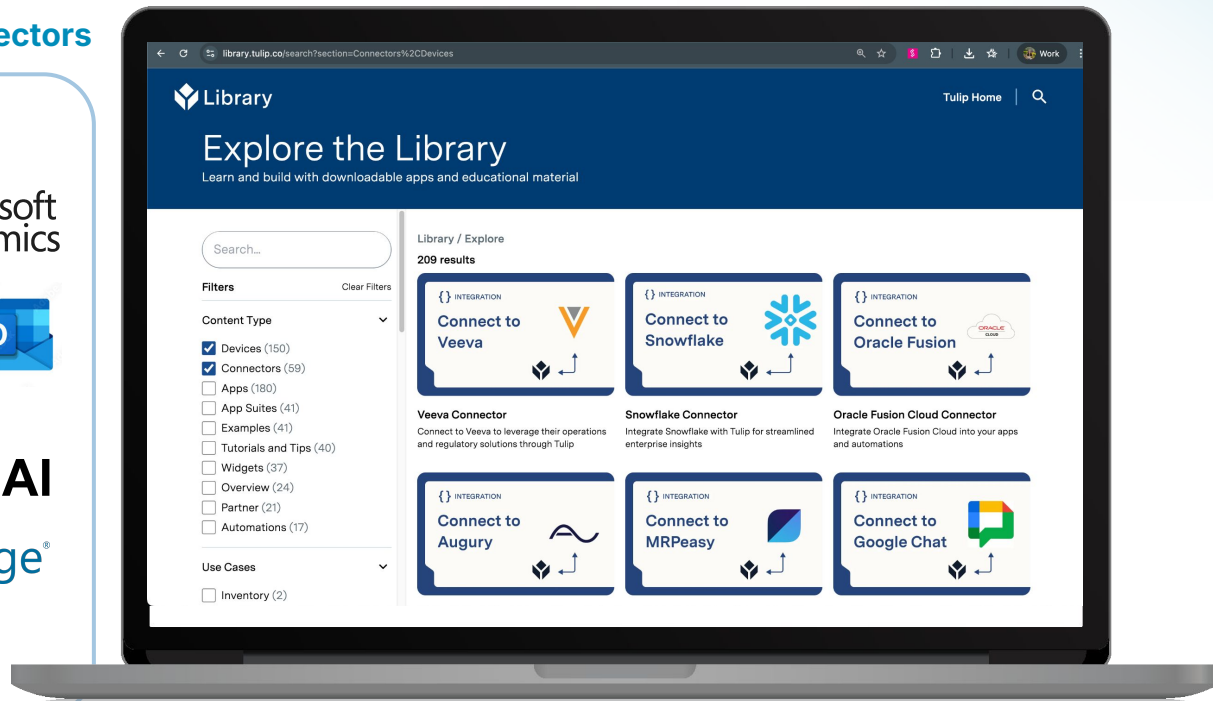
 Microsoft®
SQL Server®

 MySQL™

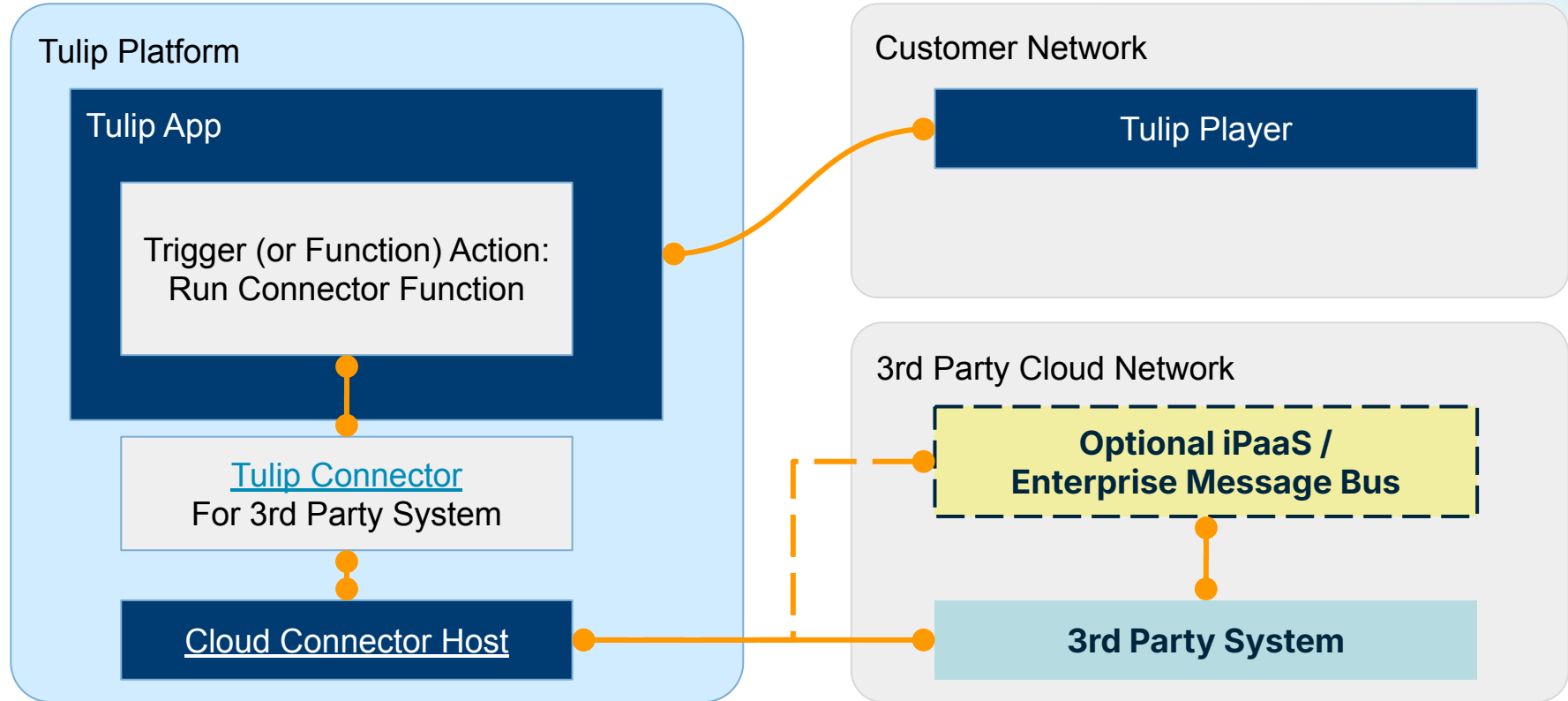
 PostgreSQL

Connectors in the Library

Connect the systems that power your
operations with 50+ configurable connectors



Tulip to 3rd Party Cloud-Hosted System (Typical Best Practice Approach)



Focus: Integration with ERP

Tulip vs. ERP Systems - Assumed Systems of Record (i.e. Sources of Truth)

ERP

Order Management

- Planning & Scheduling
- Order Items
- Complete Quantity
- Scrapped Quantity

General Ledger

- Goods Receipt
- Inventory Moves
- All Costs (e.g. Material, Labor)

Inventory / Material Management

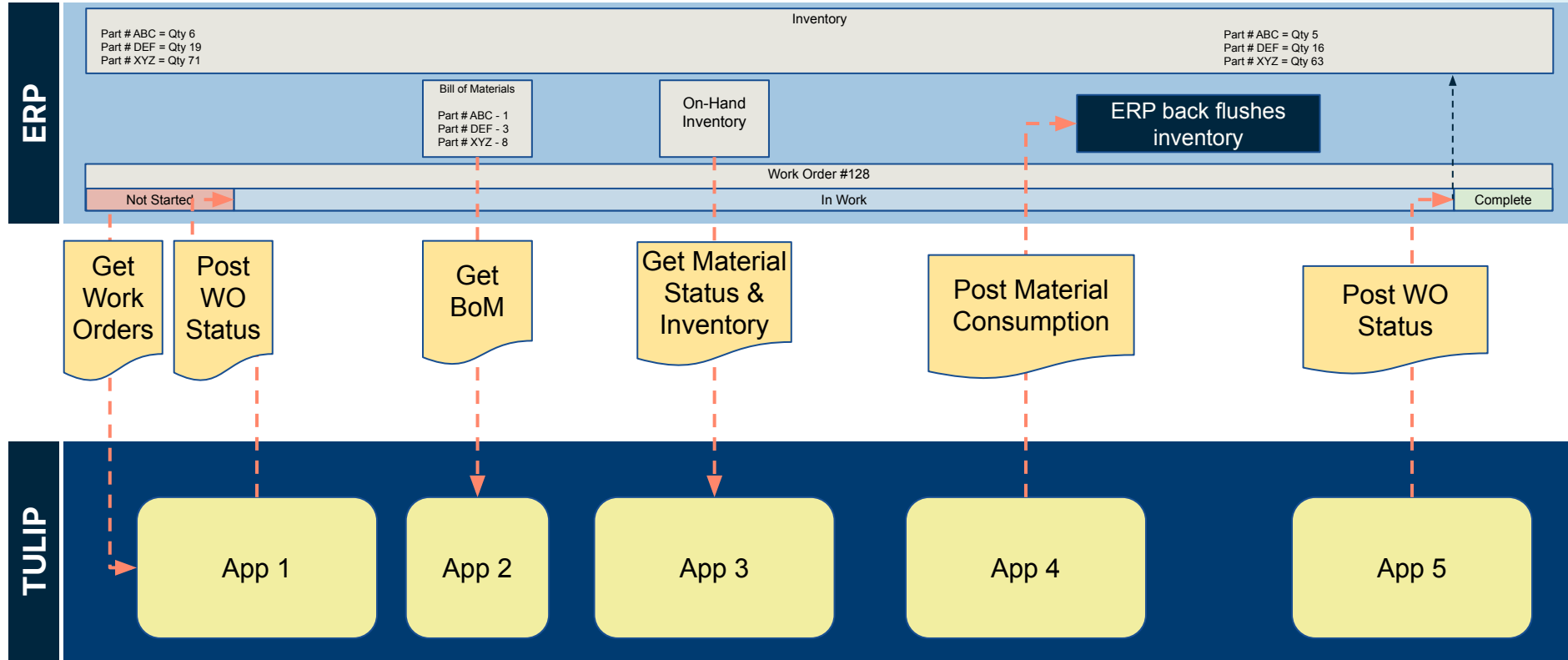
- Quantity On Hand
- Item Masters
 - Material Costs
 - Cross Reference Info

TULIP

Shop Floor Context

- Work Instructions
- Real-Time Manufacturing Operation Status
- Scrap/Defect Context
- Material Traceability
- Shop Floor Events (e.g. Digital Andons)
- NCRs / Quality Events
- Process Results
- User Certifications

Typical ERP - Tulip Integration Use Cases

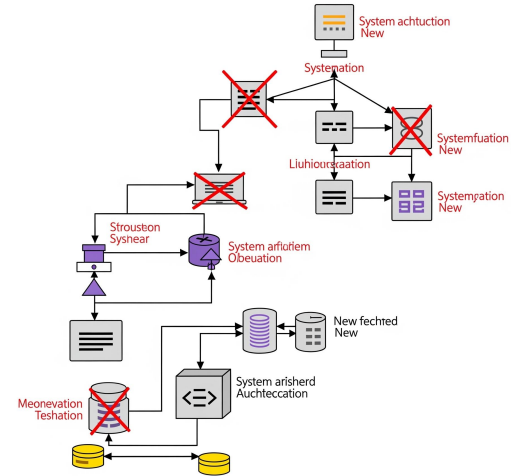


Practical Limitations

Platform Constraints to Consider in Integration Architecture



The Ideal vs. The Practical



The Ideal vs. The Practical

The 'Ideal' Best Practice

- **Maintain a single Source of Truth / System of Record.**
- **Assumptions**
 - The Source of Truth has the latest/greatest content.
 - The Source of Truth can be integrated via HTTP REST APIs
- **'Ideal' Best Practice**
 - Integrate with transactional Sources of Truth in real-time via HTTP Connector Functions

The 'Practical' Practice

- **Various factors may make the 'Ideal' Impractical**
 - The Source of Truth has regular long maintenance downtimes.
 - Process Orders with VERY LARGE Bills-of-Material → may be impractical to fetch via REST API.
 - Hard business requirement to ensure Manufacturing platform is minimally dependent on uptime of Sources of Truth
 - IT / System Owner won't let us integrate in 'ideal' fashion.
- **'Practical' Practice**
 - Tulip may 'integrate' with Sources of Truth asynchronously via content stored in Tulip Tables or elsewhere.
 - Ideally the Source of Truth (and iPaaS) synchronizes data as near-time as possible.

Platform Limits & Best Practices



Tulip Table API Limits

High-Frequency Table Writing Using Table APIs

Using Table APIs to write multiple times per second can impact your instance's performance and cause delay in storing data. The recommendations below apply to all scripts that run on a Tulip instance:

- Limit Table writes to 10 calls per second per table
- Avoid frequent writes to the same Record in a Table (more than 1/sec)
- When thresholds are exceeded, Tulip may rate limit and return error code 429. Scripts should be made resilient to 429 error codes to try again after a delay.

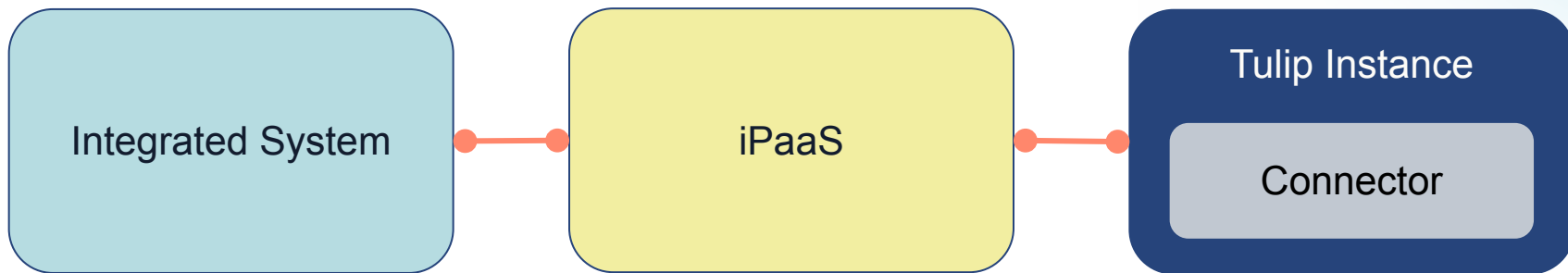
Other Integration Practices

- Using an iPaaS
- Asynchronous Integration Methods



Focus: Integration Platform as a Service (iPaaS)

Why Use an Integration Platform as a Service (iPaaS)?



- **Why use an iPaaS?**
 - a. To provide modern API endpoints (REST/JSON preferred by Tulip) if the Integrated System cannot.
 - b. Automated retries on API timeouts.
 - c. API logging (enables better debugging when needed).
- **Which iPaaS?**
 - a. Tulip does not have any specific recommendations regarding which iPaaS (but Boomi & Mulesoft are common choices used by our customers).

Other Non-Ideal Integration Architectures

Customer Infrastructure

Tulip

1

Ideal

Business System

iPaaS

HTTP Connector

2

Business System

Business System's DB

SQL Connector

3

Business System

iPaaS

POSTs & PUTs

Tulip Table APIs

GETs

Tulip Tables

Tulip Solution

(Apps, Functions, Automations, AI Agents)

4

Business System

iPaaS

Database
(Oracle DB, MS SQL, MySQL, PostgreSQL)

SQL Connector

5

Business System

iPaaS

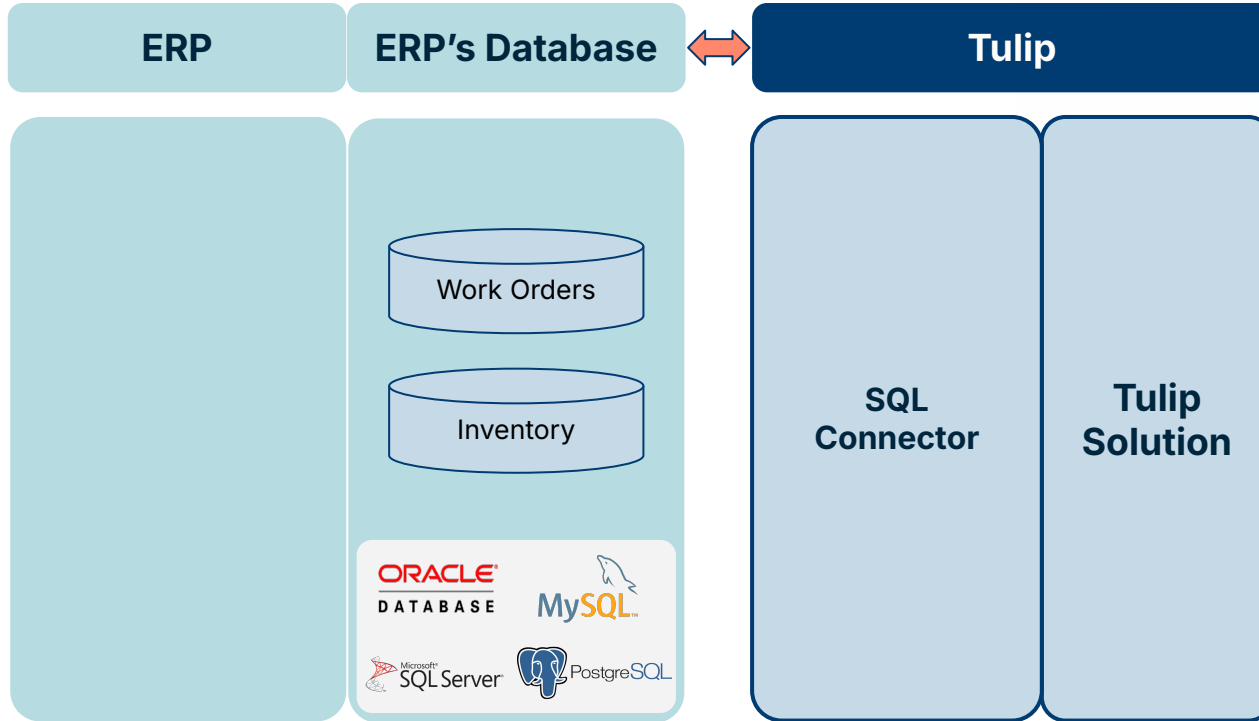
Data Lake/House

HTTP Connector

Focus:

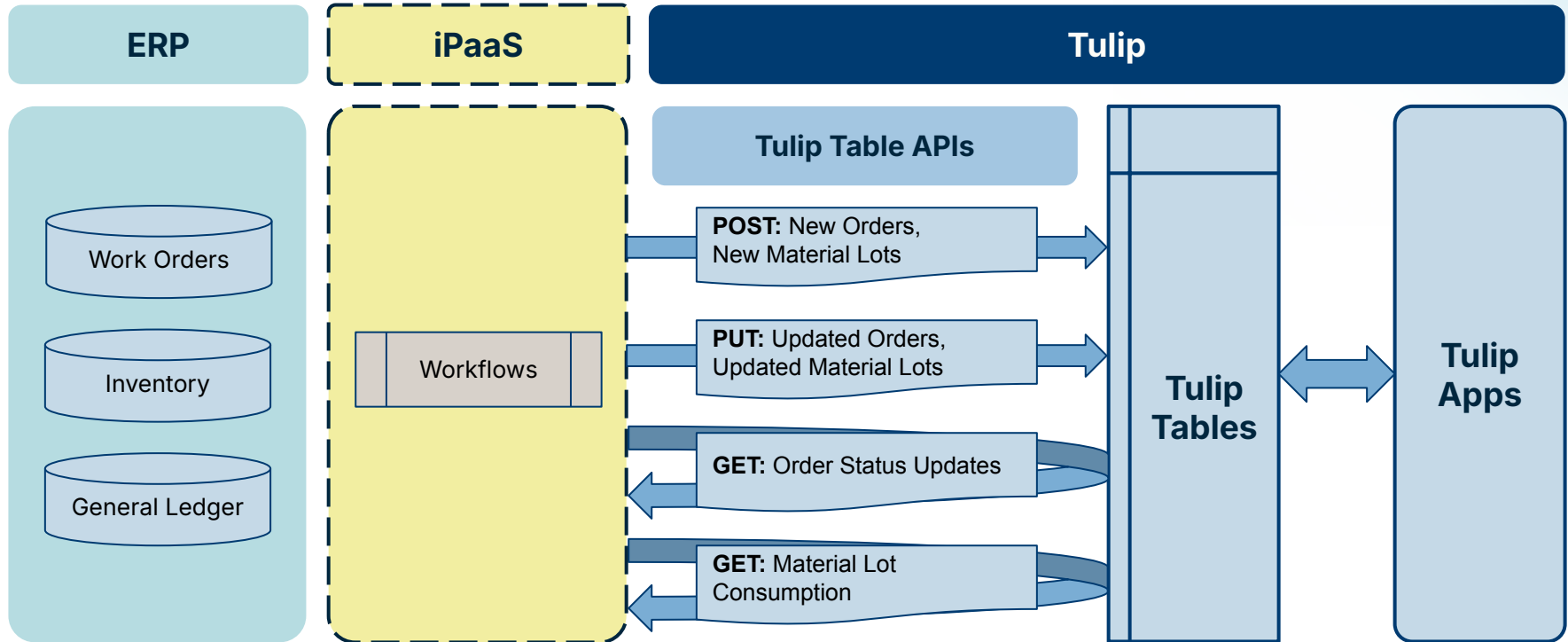
Integration via SQL Connector

Synchronous Integration via SQL Connector



Focus:
Asynchronous Integration via
Tulip Table Content &
Tulip Table APIs

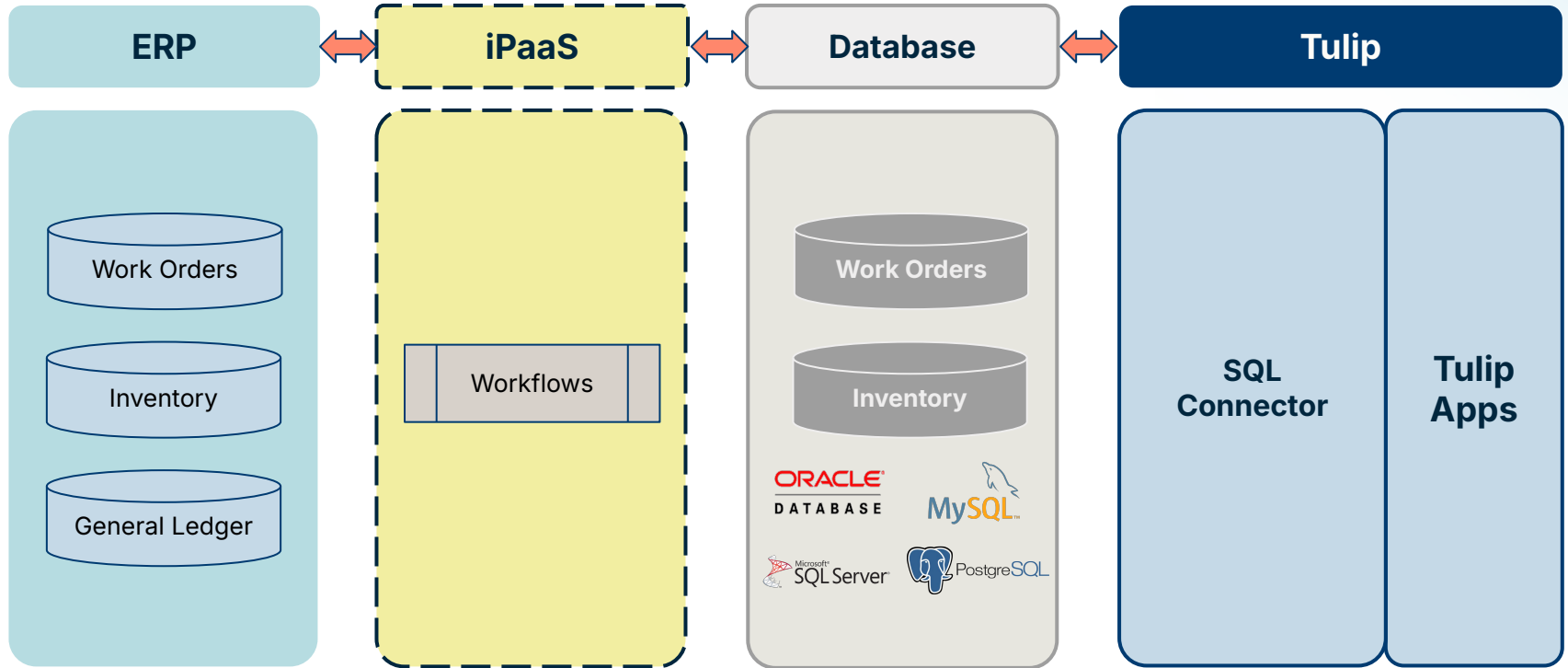
(Simplified) Asynchronous Integration via Tulip Tables & Tulip Table APIs



Focus:

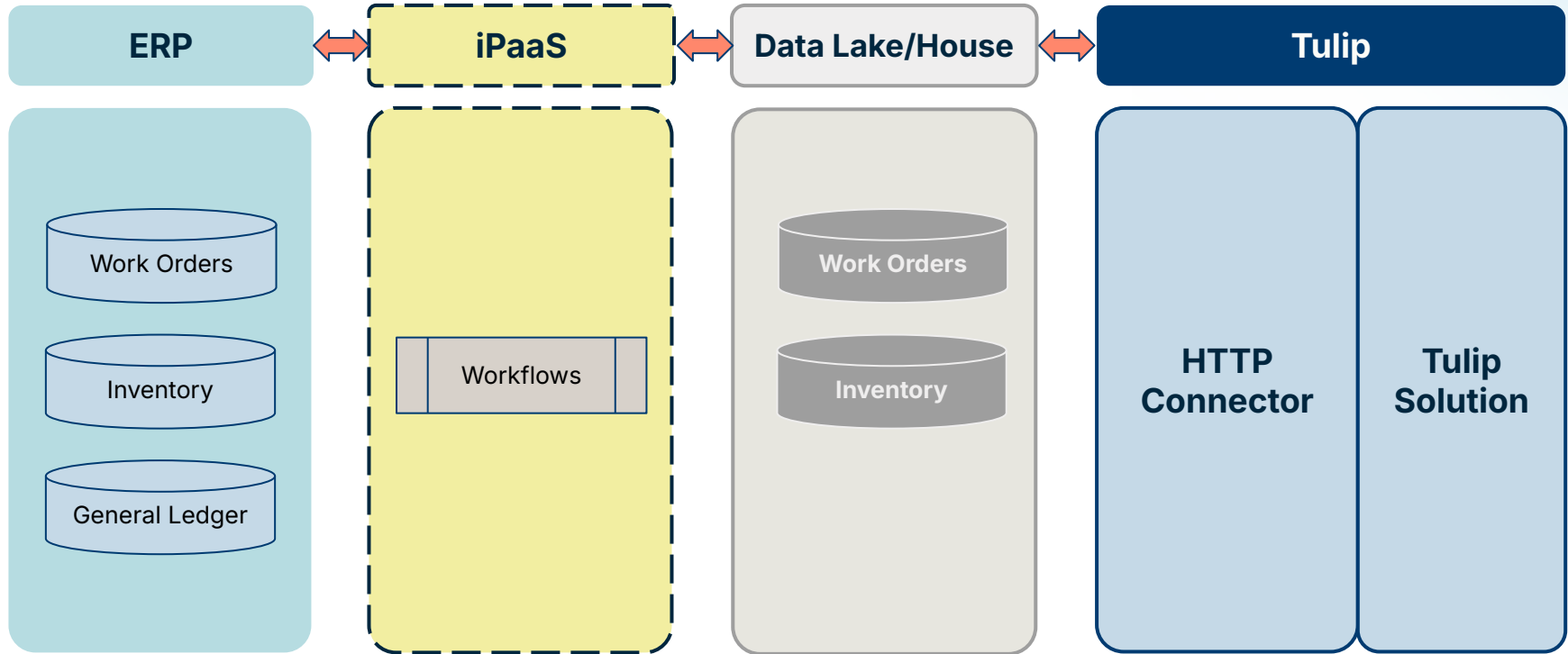
Asynchronous Integration via 3rd Party Database

(Simplified) Asynchronous Integration via 3rd Party Database



Focus:
Asynchronous Integration via
Date Lake/House

(Simplified) Asynchronous Integration via Data Lake/House



Discussion:

What other integration practices are you using? Lessons learned?

Don'ts from Past Experiences



Don't use Tulip Automations for
high-frequency work that an
iPaaS is best suited for.

(e.g. managing data synchronization between ERP
and Tulip via Tulip Table content)

Don't integrate Tulip with a transactional business system via UNS/MQTT.

(MQTT in the Tulip platform is built largely with integration to machines/sensors in mind.)

Don't use a SQL Connector if
HTTP APIs are available.

Discussion:

Any lessons learned from the audience?

Upcoming Best Practices...



Write business logic once and reuse everywhere with Functions

Functions Help You Build Composable Solutions Faster

✓ Standardize logic and reuse integrations

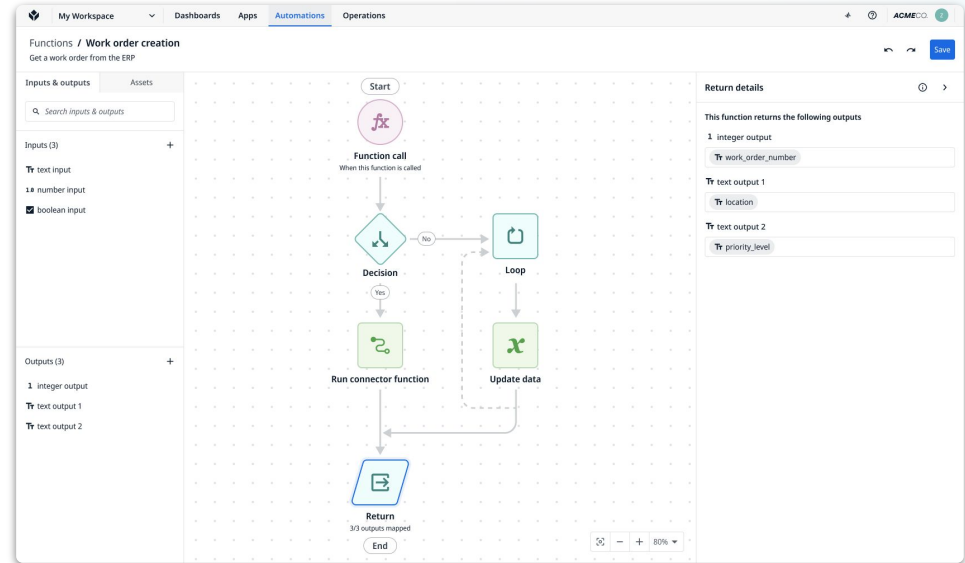
Enhance governance and ensure compliance with validated, centralized components

✓ Accelerate app development

Empower citizen developers to combine centrally-developed logic with user-specific requirements

Examples

- Bundle multiple connector calls for easy ERP data integrations
- Consolidate core Work Order processing logic across hundreds of apps
- Reuse logic for barcode scans and manual barcode entry



Compose AI Agents to Solve Real Problems

What is an Agent? Agents are a new, composable component of the Tulip system. Combine a set of building blocks to design an Agent that solves a specific problem.

Goal → Instructions + Tools

- **Goal:** Specific objective of the Agent
- **Instructions:** How to achieve that goal
- **Tools:** Capabilities to achieve that goal

What Makes Agents Different?

- **Autonomous:** Agents take action and make decisions
- **Collaborative:** Agents can work with each other
- **Context-Aware:** Agents understand your deployment

The screenshot shows the 'Build' tab of the 'Quality Review Assistant' agent configuration. It is divided into three main sections: Goal, Instructions, and Capabilities. To the right of each section is a light blue callout box with a label.

- Goal:** The goal text is 'Pete's Quality Reviewer aims to ensure that all parts produced meet the highest quality standards by systematically their specifications and performance. The agent will assist users by identifying defects or inconsistencies, allowing corrections and minimizing the risk of defective parts reaching customers.' A callout box labeled 'Set Goal' is positioned to the right.
- Instructions:** The instructions are a numbered list: 1. Receive part specification details and production output from the user. for this step use @connect.ERP.getOrder. 2. Conduct a thorough inspection by comparing each part against the provided specifications and quality criteria. 3. Identify any deviations, defects, or non-compliance issues in the parts and log these findings. 4. Generate a comprehensive report highlighting the quality status and any necessary corrective actions. 5. Advise the user on corrective measures to address identified quality issues and recommend follow-up inspection. 6. Maintain a record of the quality evaluations for future reference and continuous improvement initiatives. A callout box labeled 'Define Instructions' is positioned to the right.
- Capabilities:** The capabilities section shows a list of available tools with selection counts: 'Tulip Capabilities' (61 available, 7 selected), 'External MCP' (Coming Soon), 'Connector Functions' (10 available, 2 selected), and 'Tulip Functions' (11 available). A callout box labeled 'Configure Tools' is positioned to the right.

At the bottom of the interface is an 'Advanced Settings' dropdown menu.

QUESTIONS?

Tulip Knowledge Base:

Architecture Fundamentals of Integrating Tulip to a Transactional Business System



TULIP

THANK YOU