TIBCO Cloud Events Workshop

Virtual Workshop via Zoom

TIBCO Cloud Events – Asynchronous API Workshop

21st May 2020

CONFIDENTIALITY & DISCLAIMER

This document (including, without limitation, any product roadmap or statement of direction data) illustrates the planned testing, release and availability dates for TIBCO products and services. This document is provided for informational purposes only and its contents are subject to change without notice. TIBCO makes no warranties, express or implied, in or relating to this document or any information in it, including, without limitation, that this document, or any information in it, is error-free or meets any conditions of merchantability or fitness for a particular purpose.

The material provided is for informational purposes only, and should not be relied on in making a purchasing decision. The information is not a commitment, promise or legal obligation to deliver any material, code, or functionality. The development, release, and timing of any features or functionality described for our products remains at our sole discretion.

During the course of this presentation, TIBCO or its representatives may make forward-looking statements regarding future events, TIBCO's future results or our future financial performance. These statements are based on management's current expectations. Although we believe that the expectations reflected in the forward-looking statements contained in this presentation are reasonable, these expectations or any such forward-looking statements could prove to be incorrect and actual results or financial performance could differ materially from those stated herein. TIBCO does not undertake to update any forward-looking statement that may be made from time to time or on its behalf.

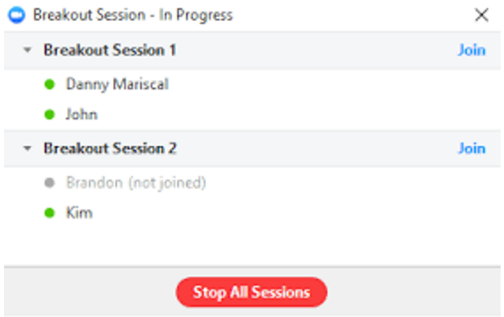
About TIBCO

TIBCO Software unlocks the potential of real-time data for making faster, smarter decisions. Our Connected Intelligence Platform seamlessly connects any application or data source; intelligently unifies data for greater access, trust, and control; and confidently predicts outcomes in real time and at scale. Learn how solutions to our customers' most critical business challenges are made possible by TIBCO at www.tibco.com.

Agenda & Timings

Time	Length	Subject	Presenter
13:00	15 Minutes	Welcome & Introduction to the team	
13:15	30 Minutes	Driving agile innovation across the Enterprise with the TIBCO Cloud Platform and our Event Based and API-led connectivity capabilities: Opportunities, challenges of connectivity and integration in today's reality	
14:00	30 mins	Product Roadmap	
14:30	2 Hours	Hands-on Labs	
15:30	10 Minutes	Break	
	30 Minutes	**Discussion & Wrap-up**	

Virtual Delivery / Digital Etiquette



- We'll be using Video - but don't feel pressured to do so yourself!
- We'll be using a single Zoom session for the majority of the time. Breakout rooms will be available for individual help during the Labs
- Do Mute yourself when not speaking as this helps with the audio experience
- Use the "Raise Hand" button if you want to ask a question, but want to wait for a convenient point
- Put a question in the Chat window
  - Send it to everyone if you think the other participants would benefit
- Or just Unmute yourself and ask!

TIBCO®

Mark Mussett

Senior Solutions Engineer

Veteran IT specialist with a career spanning 21 years in the Integration industry. Joined TIBCO in 2005, with 11 years in Professional Services as Technical Architect and 3 years in Pre-Sales organisations working with our Connected portfolio of products.

Expertise in API (Mashery), Integration (BusinessWorks, Flogo), Messaging (JMS, FTL, and RV), and Golang programming.

TIBCO®

Dave Winstone

Senior Solutions Engineer

I have had the benefit of working for TIBCO for the last 14 years, and that has allowed me to work with a wide variety of customers – from the largest Banks and Telco's in the world, to the smallest innovators and disruptors.

I specialise in our "Connect" portfolio, allowing me to work with our customers on ensuring the right solutions and architectures are employed to provide the base for Digital Transformation.

Bryn Garton

Senior Solutions Engineer

Except for a brief spell as an Electrical Engineer my entire career has been spent working for software companies. For the majority of that time I have worked in the field of middleware across its many generations, with roles in Consulting, Software Development, Product Management and most recently Presales.

Since joining TIBCO in 2005 I have worked with TIBCO's customers across many industry verticals including Finance, Energy, Logistics, Utilities and Government.

TIBCO®

Rishikesh Palve

Senior Product Manager

I am a Senior Product Manager at TIBCO. I have over 9 years of experience working in various roles across different groups such as P&T, Customer Engineering, PreSales and Product Management.

I take keen interest in APIs, Microservices, Event-driven architectures and Cloud-Native ecosystem. At TIBCO, I am responsible for building product roadmap and vision for our Cloud-Native, Event-driven product capabilities that are part of the TIBCO Connected Intelligence platform.

Customer Challenge

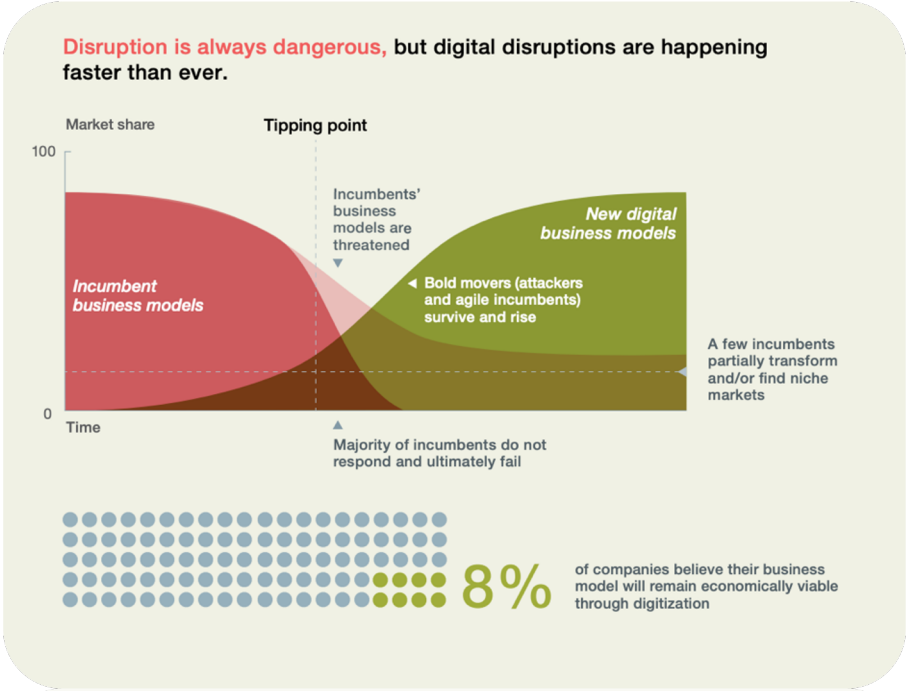
How Do You Create & Accelerate Value in the Digital World

TIBCO's Value To Our Customers

Delivering Connected Intelligence through our Digital Business Platform

Digital disruption is happening *faster than ever*

- Incumbent Models are under attack
- Agile Movers Survive & Rise
- New Models Created
- <8% believe “their biz model will remain eco viable through digitalization”



McKinsey Report available at https://www.mckinsey.com/~/media/McKinsey/Business%20Functions/McKinsey%20Digital/Our%20Insights/Why%20digital%20strategies%20fail/Why-digital-strategies-fail.ashx

Digital disrupts = Companies need to change how they work & play.. Ecosystems

- Platforms facilitate Ecosystem Business Models
- = Business Value
- = Agility
- Businesses must be Open, API(s) will create the new models assisting new models & opportunities

TIBCO

How Are TIBCO's Customers Innovating for the **Connected Intelligence** Era?

THE UNIVERSITY OF CHICAGO MEDICINE

$600,000
Anticipated OR cost savings PA

"In real time, we are able to alert our rapid response team to go to the bedside of a patient who is likely to go into cardiac arrest. So far, we've successfully reduced the number of cardiac arrests in the hospital by an estimated 15 to 20%."
Chief Enterprise Architect, Christine Watts

Use Case
Integrated systems and centralized data to more effectively measure services and predict risks

SEPA Scottish Environment Protection Agency

60%
Saving of time compared with previous technologies used

"The software's ability to cover the full data science development lifecycle: data retrieval, preparation, transformation, analysis, and final interactive dashboard presentation, meant it could easily meet the aspirations of our proposed agile informatics process," Informatics Head of Unit SEPA, Mark Hallard

Use Case
Evidence-based decisions and efficient and automated analytics

AMG PETRONAS MOTORSPORT

6 Years
Time to deliver own visualizations
vs. 2 months

The driver changes gears about 100 times a lap. Every time there's a gear change, we collect around 1,000 data points. Those gear changes can be categorized into how fast they were, and how much stress they put on the gears. When you visualize that data, you can actually make the gear box last longer, or more importantly, make harsher gear changes. You can then find that if you put the gear box into one particular mode, it's roughly 50 milliseconds faster per lap. Cars that were fourth and fifth in qualifying were one one-thousandth of a second different, so 50 milliseconds matters. So for us to be able to make those harsher gear changes is important. We spent six years creating our own visualizations. TIBCO turned up, and in a matter of two months, did better work." Head of IT, Matt Harris

Use Case
Real Time Car Dynamics Changing, Continuous Improvement

https://www.tibco.com/customers/scottish-environment-protection-agency
https://www.tibco.com/customers/university-chicago-medicine
https://www.tibco.com/formula-one-competitive-advantage

Only TIBCO can guarantee its customers digital sovereignty by providing:

Comprehensive Platform

Capabilities for Digital Execution

Innovate and collaborate without constraints

Accelerated Innovation

Simplification, Automation & Agility

Deploying anywhere and everywhere at pace

Freedom

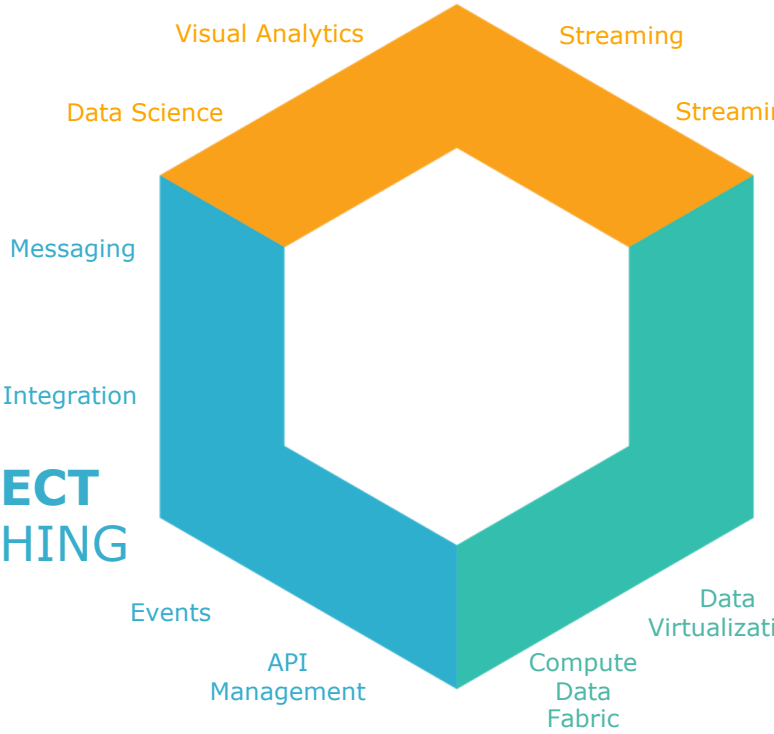
Choice & Scale

Operational Excellence From experimentation to mission critical

Digital is Built On This Foundation..

Best-In-Class, Open and Independent

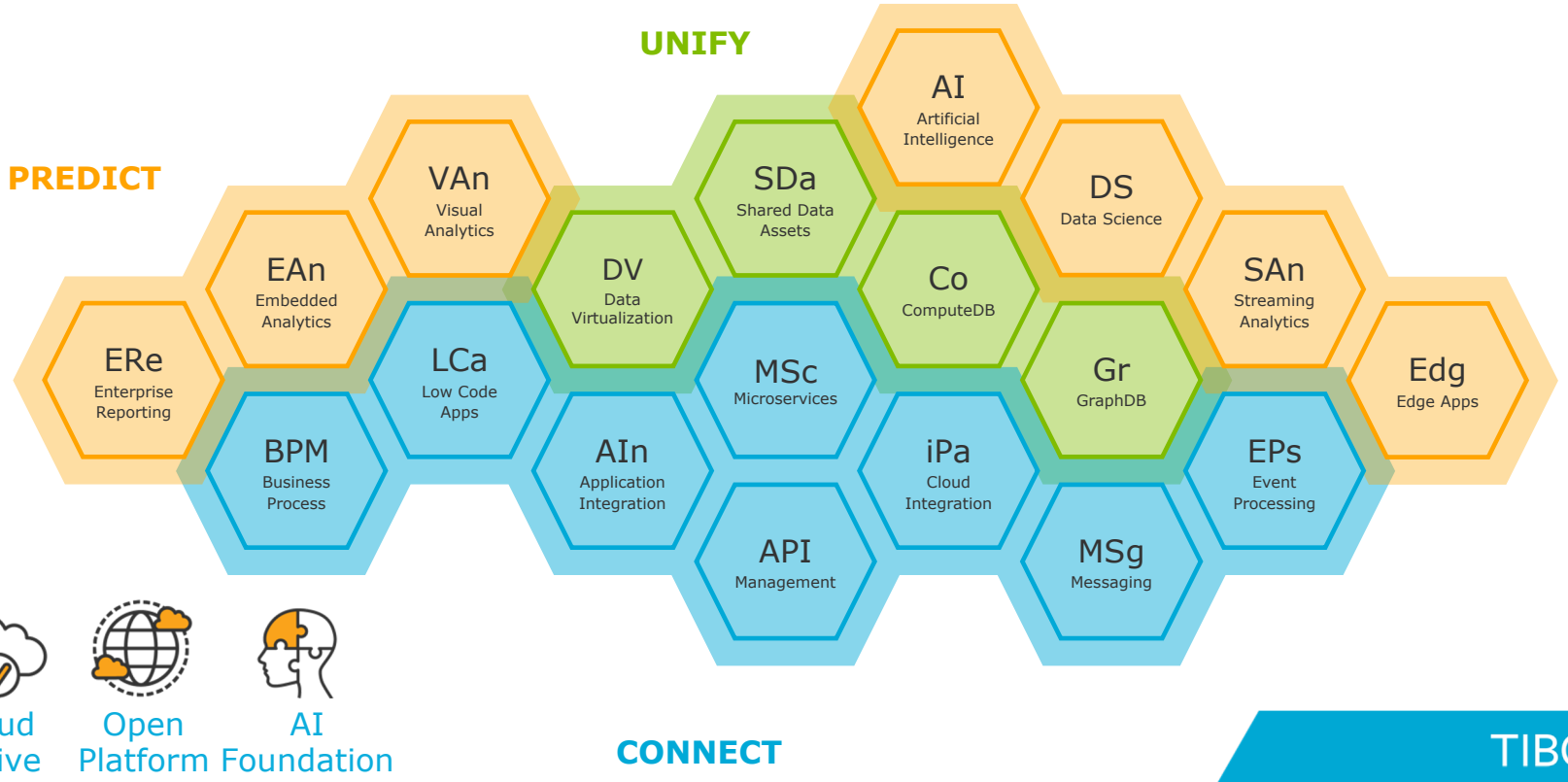
PREDICT



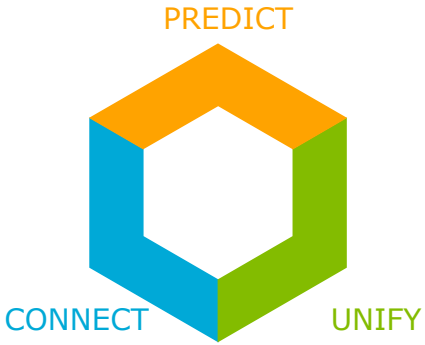
CONNECT EVERYTHING

UNIFY INFORMATION

How TIBCO delivers the Connected Intelligence Vision



Built on Strong Core Principles



Cloud Native

- Re-architected core capabilities as cloud-native ‘engines’
- Deliver the same core cloud-native components across workloads
- Significantly reduced or removed any proprietary dependencies
- Devops/Cloudops/Dataops support delivered as open source

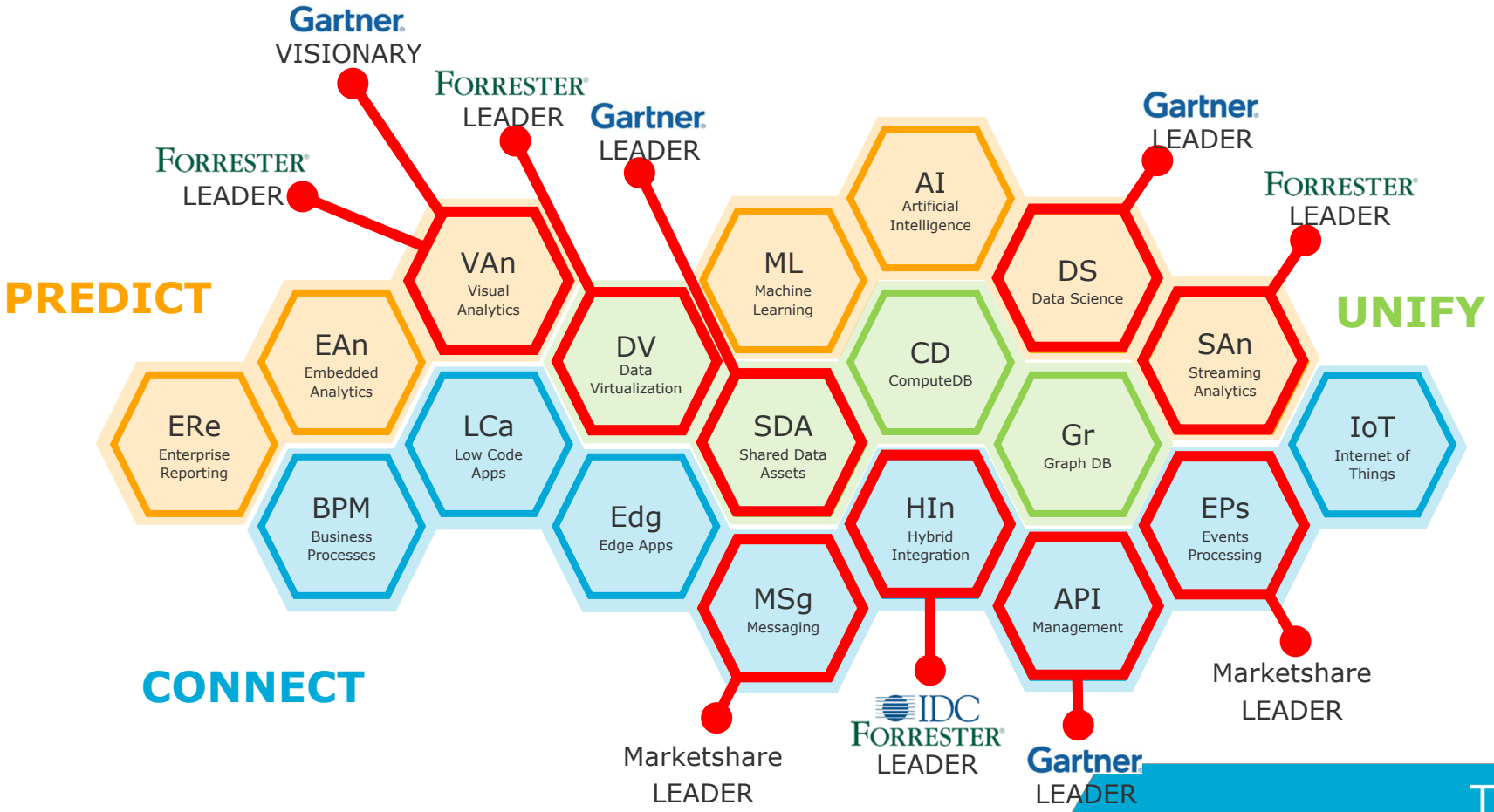
Open Platform

- Provide customers the same tools used internally to extend capabilities
- Open source delivery of new key capabilities like Project Flogo
- Continued adoption of enterprise and emerging open source (e.g. Kafka)
- Increasingly active engineering participation in larger scale OSS projects

AI Foundation

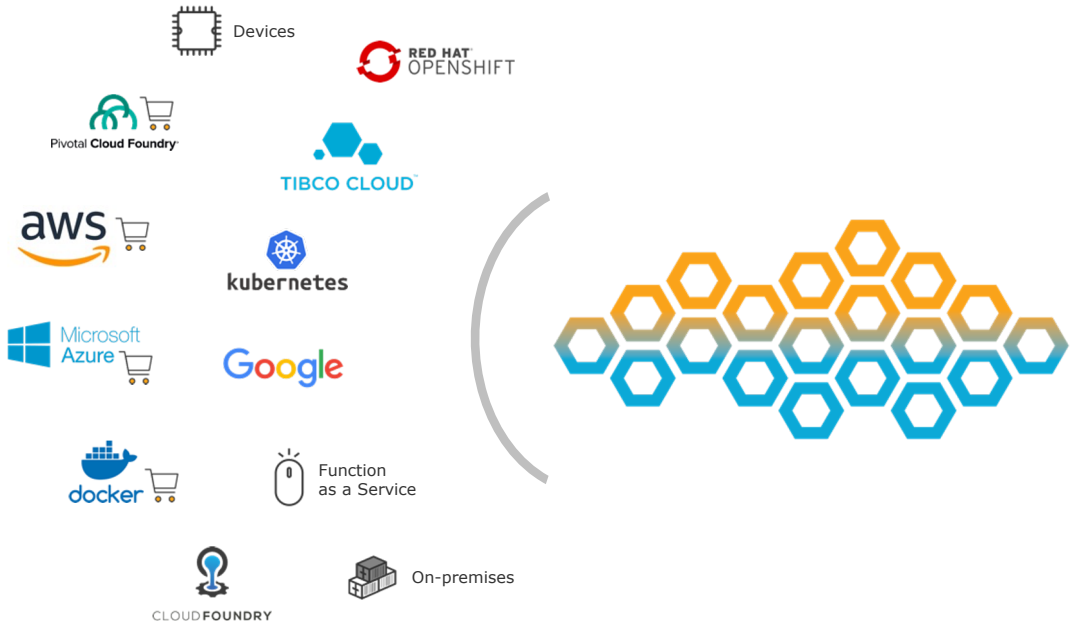
- Application of AI principles and technology across capabilities to enhance workflow and user productivity
- Integration to leading AI frameworks both for real-time projects (Flogo with TensorFlow) as well as Data Science modeling and design
- Increasing use of AI as a platform for developing new capabilities such as adaptive business logic and execution

Market Leadership Across The Platform



Available anywhere, everywhere

TIBCO Connected Intelligence



Open Source

Give developers easy access to core capabilities of the Connected Intelligence Cloud

SaaS

Quickly get up and running with a fully TIBCO hosted and managed service

Hybrid

With all options for self-host, multi-cloud, or edge deployments

Dynamic Cloud

Optimize cost by dynamically scaling on cloud infrastructure such as Amazon Web Services

API-Led Strategy Is the Connective Tissue of Digital Business

App Dev & Integration

Internal Agility
Standardize services

Cloud-native
Microservices & microgateways

Integration
SaaS and Hybrid Services

Digital Business Platform

Channel Expansion
New Distribution

B2B & B2C Innovation
Partner Collaboration

New Business Models
Becoming an ecosystem platform

API Platform

Enterprise Data, Services, and Applications

TIBCO

TIBCO Connected Intelligence Cloud

Capabilities

PREDICT

Data Management

Advanced Data Visualization

Streaming Analytics

Shared services

Billing

SSO

Account Management

Customer Servicing

Service Registry

Integration and API Management

Messaging and Events Processing

Digital Process Automation

INTERCONNECT EVERYTHING

TIBCO CLOUD™ Platform

TIBCO

TIBCO Cloud in a Nutshell

Capabilities available on TIBCO Cloud

- TIBCO Cloud Nimbus
- TIBCO Cloud Live Apps
- TIBCO Cloud Integration
- TIBCO Cloud Mashery
- TIBCO Cloud AuditSafe
- TIBCO Cloud Events
- TIBCO Cloud Messaging
- TIBCO Cloud Spotfire

TIBCO Cloud Services

At-a-Glance

High Availability

Disaster Recovery

Security, Privacy, & Compliance

PKI Management

Role-based Access

Auditability

Service Level Agreement

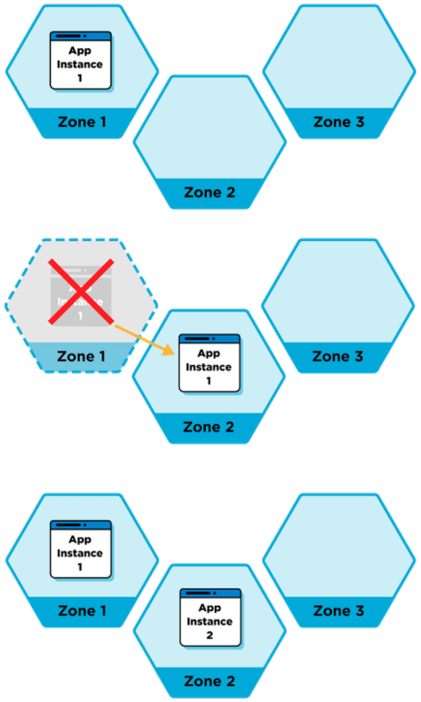
Reporting

Operational Dashboard

Transaction Tracking

Support

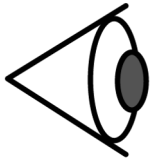
High Availability



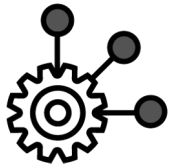
- Each application is executed in an availability zone automatically selected based on available resources.
- Automatic switch-over of apps to new zone if they go down or stop reporting status.
- Scale apps across multiple zones for high availability - load is distributed evenly by the system.

Security, Privacy and Compliance

Customer traffic, secrets and encryption



Authentication and Authorization

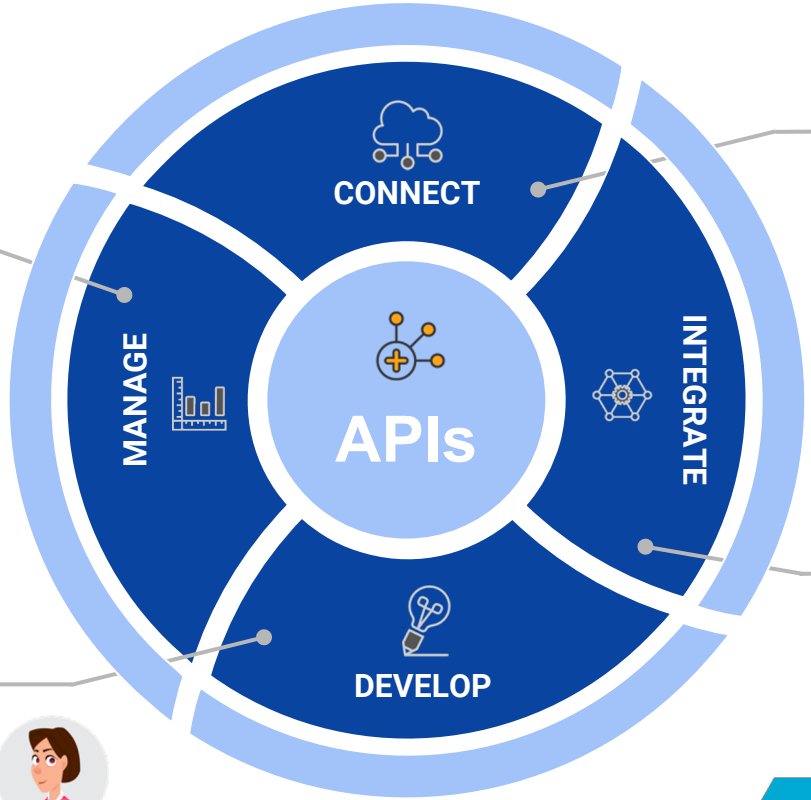


Secure API's for Mobile and Command line interface

- Multi-tenant platform with logical separation of customers.
- Built with privacy and security in mind.
- Encrypted network traffic and data storage.
- Hybrid Connectivity

Accelerate Integrations Across Your Business

Enable Collaborative Integration for Everyone through Tailored User Experiences



CONNECT SaaS business applications

Citizen Integrator / Business Analyst

MANAGE APIs for integration

API Product Manager

INTEGRATE enterprise apps across on-premises and cloud

Integration Specialist

DEVELOP event-driven microservices and functions

Application Developer

TIBCO

Desired state for enterprises

IT, LOB

Integrator, Developer

Citizen dev, Biz analyst

APIs - Data Virtualization

Integration

Algorithms – Data Science (AI,ML)

Events

ERP

Legacy Apps

Storage

Network

Cloud Provider 1

CRM

Cloud Provider 2

HR

Cloud Provider 3

Marketing

On-Premise

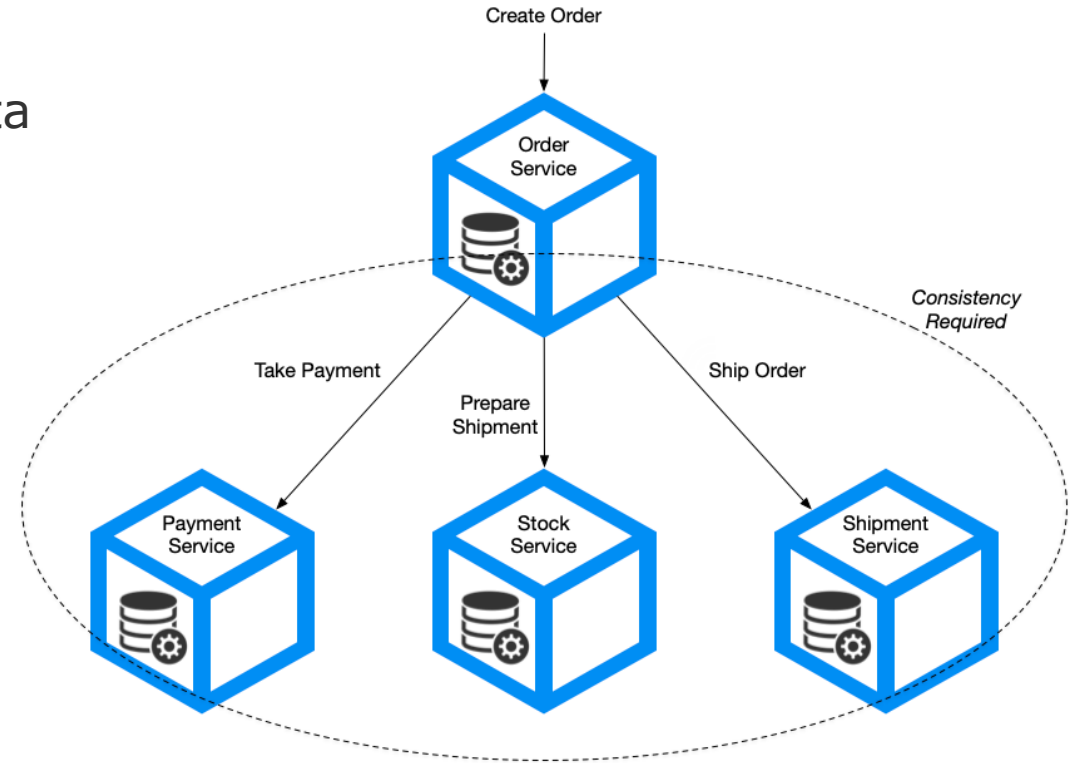
Private Cloud

Public Cloud

The Labs

Business Transactions in a Microservice Architecture

- Need to maintain data consistency across bounded context
- Saga Pattern solves the challenge of consistency through the use of Choreography or Orchestration of events across boundaries



Saga Pattern

- Pattern used to maintain data consistency in a Microservice architecture.
- Relies on the ability of every Microservice being able to maintain coherent and consistent state.
- Pattern utilises an approach of coordinating each step in the business transaction.
- By using messaging we can maintain transactional flow between participants.

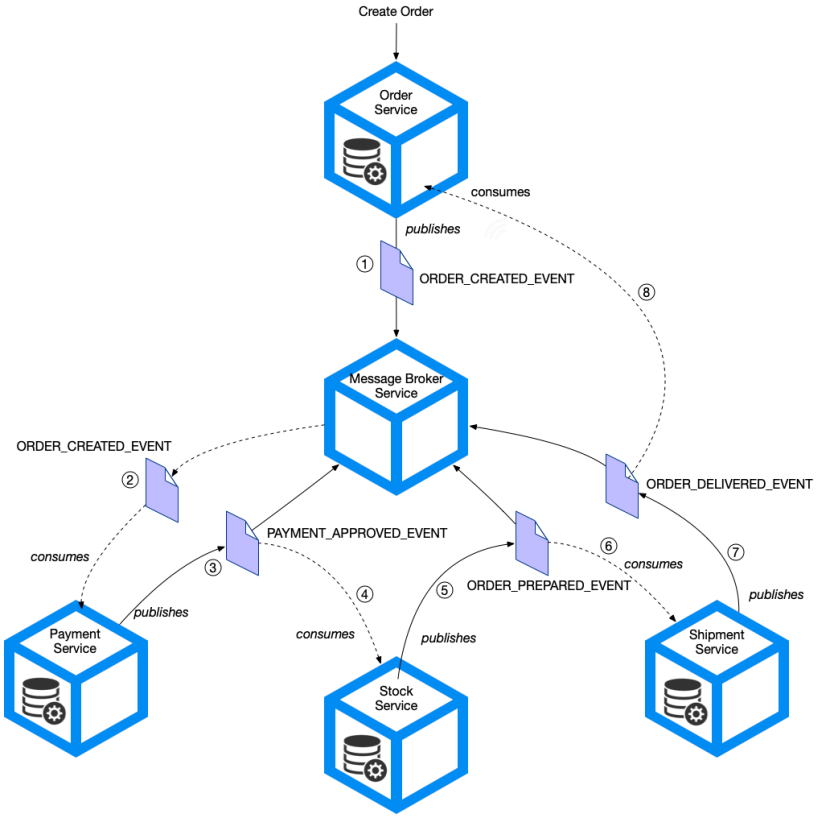
Use Case – Create Order

1. Order Service - Create an Order in PAYMENT_PENDING state.
2. Payment Service - Authorize payment for Order.
3. Order Service - Update Order to PAYMENT_APPROVED and AWAIT_PICKING state.
4. Stock Service - Prepare Order Items for dispatching.
5. Order Service - Update Order to ITEMS_PICKED state.
6. Shipment Service - Create Delivery of Order.
7. Order Service - Update Order to ORDER_SHIPPED state.

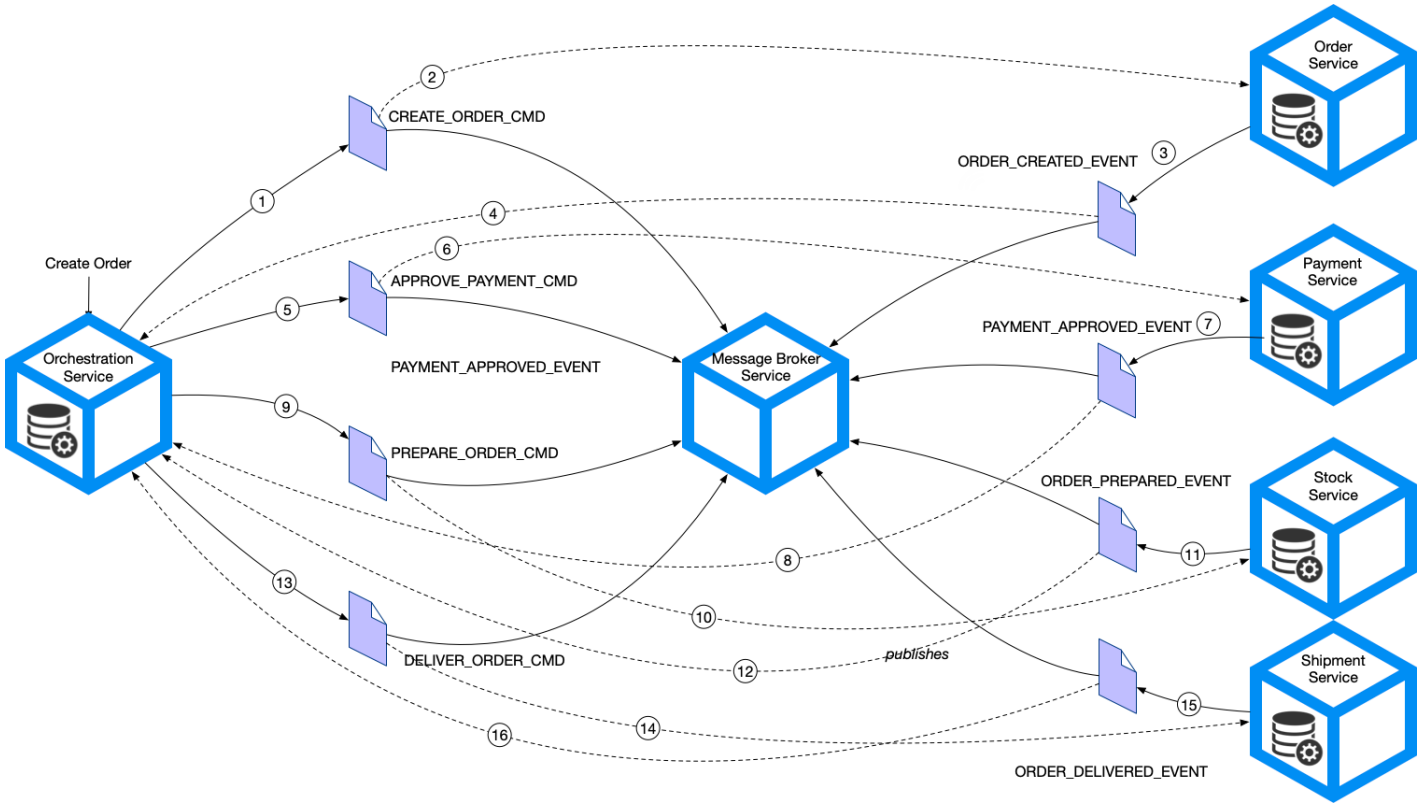
Coordinating Transactions

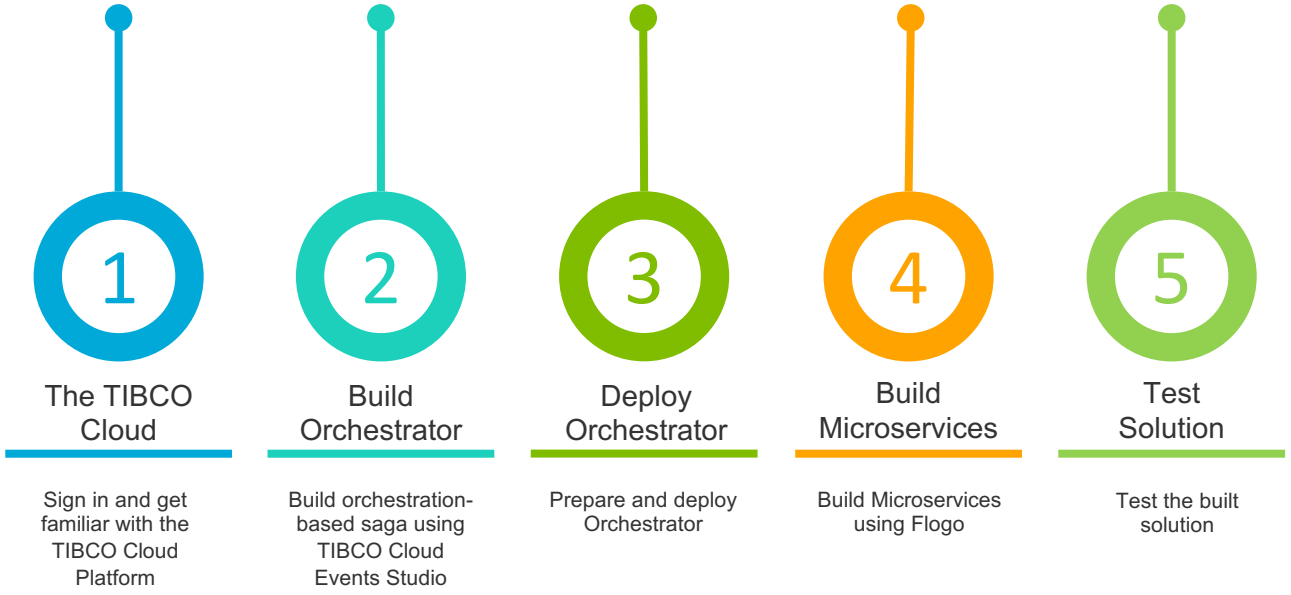
1. Choreography
2. Orchestration

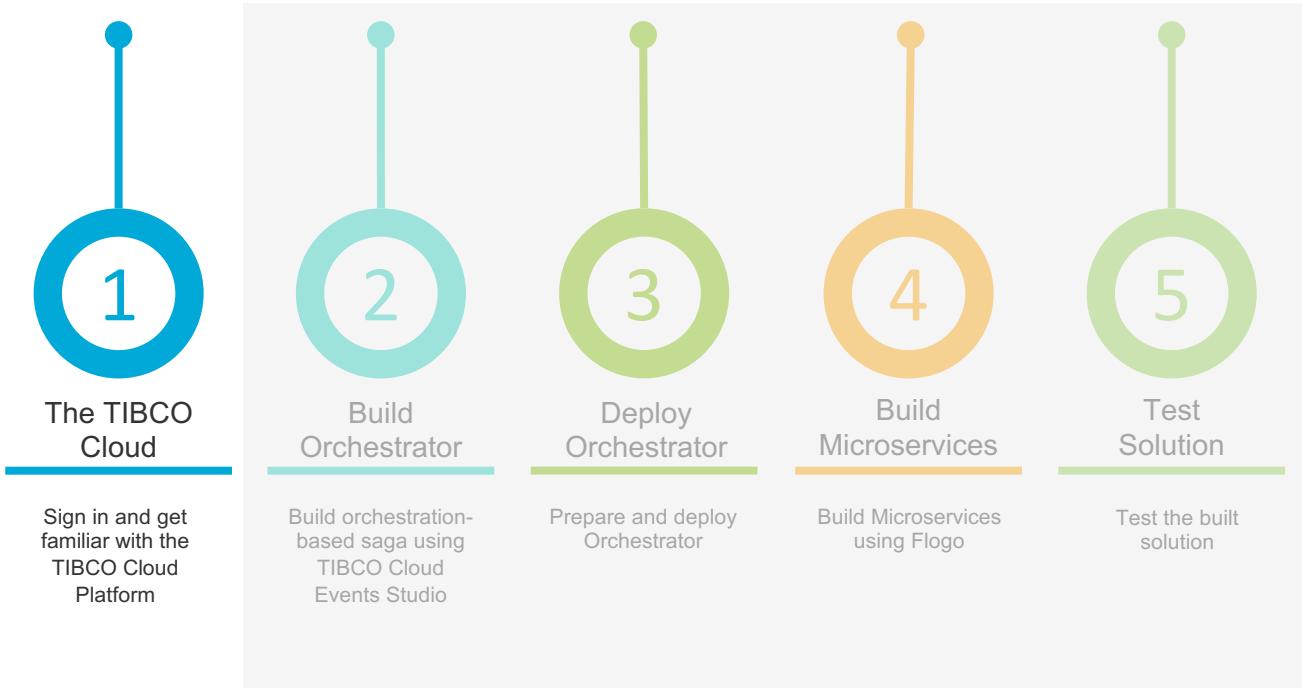
Choreography-based Saga



Orchestration-based Saga



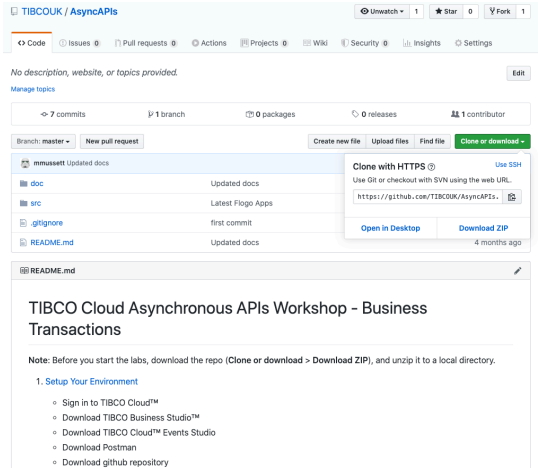

1
The TIBCO Cloud
Sign in and get familiar with the TIBCO Cloud Platform
2
Build Orchestrator
Build orchestration-based saga using TIBCO Cloud Events Studio
3
Deploy Orchestrator
Prepare and deploy Orchestrator
4
Build Microservices
Build Microservices using Flogo
5
Test Solution
Test the built solution


1
The TIBCO Cloud
Sign in and get familiar with the TIBCO Cloud Platform
2
Build Orchestrator
Build orchestration-based saga using TIBCO Cloud Events Studio
3
Deploy Orchestrator
Prepare and deploy Orchestrator
4
Build Microservices
Build Microservices using Flogo
5
Test Solution
Test the built solution

Key Resources

TIBCO CLOUD™ http://cloud.tibco.com

Download and extract the Hands On Lab : https://github.com/TIBCOUK/AsyncAPIs



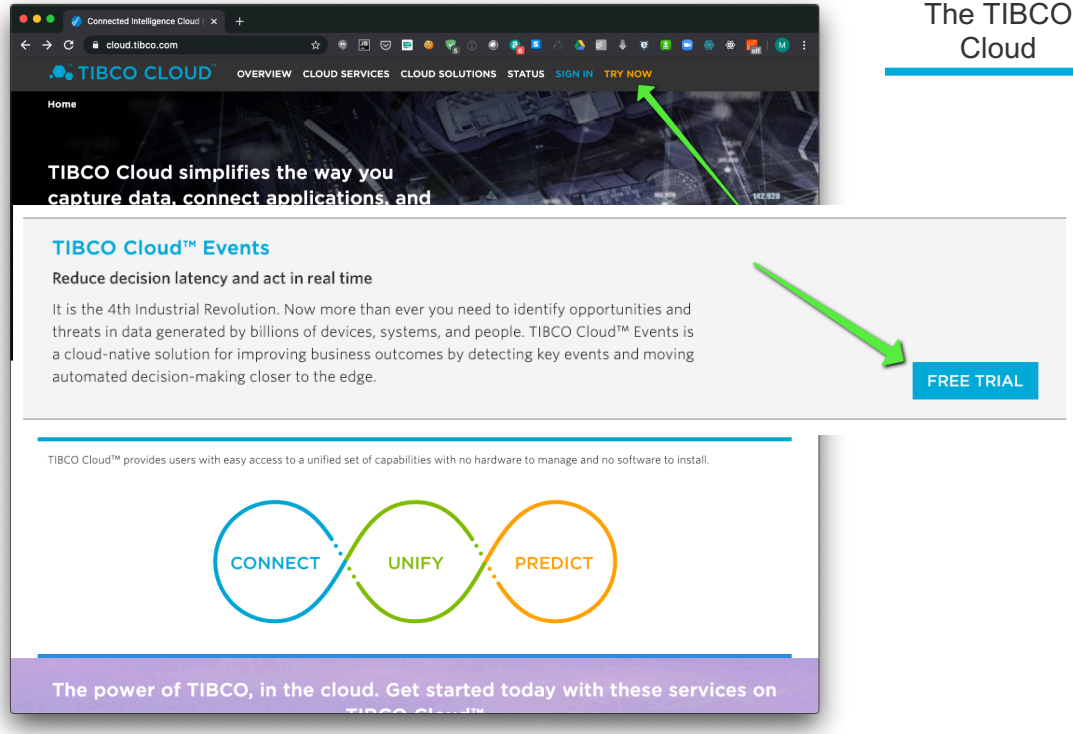
TIBCO Community :

- https://community.tibco.com/products/tibco-cloud-events
- https://community.tibco.com/products/tibco-cloud-integration
- https://community.tibco.com/products/tibco-activematrix-businessworks

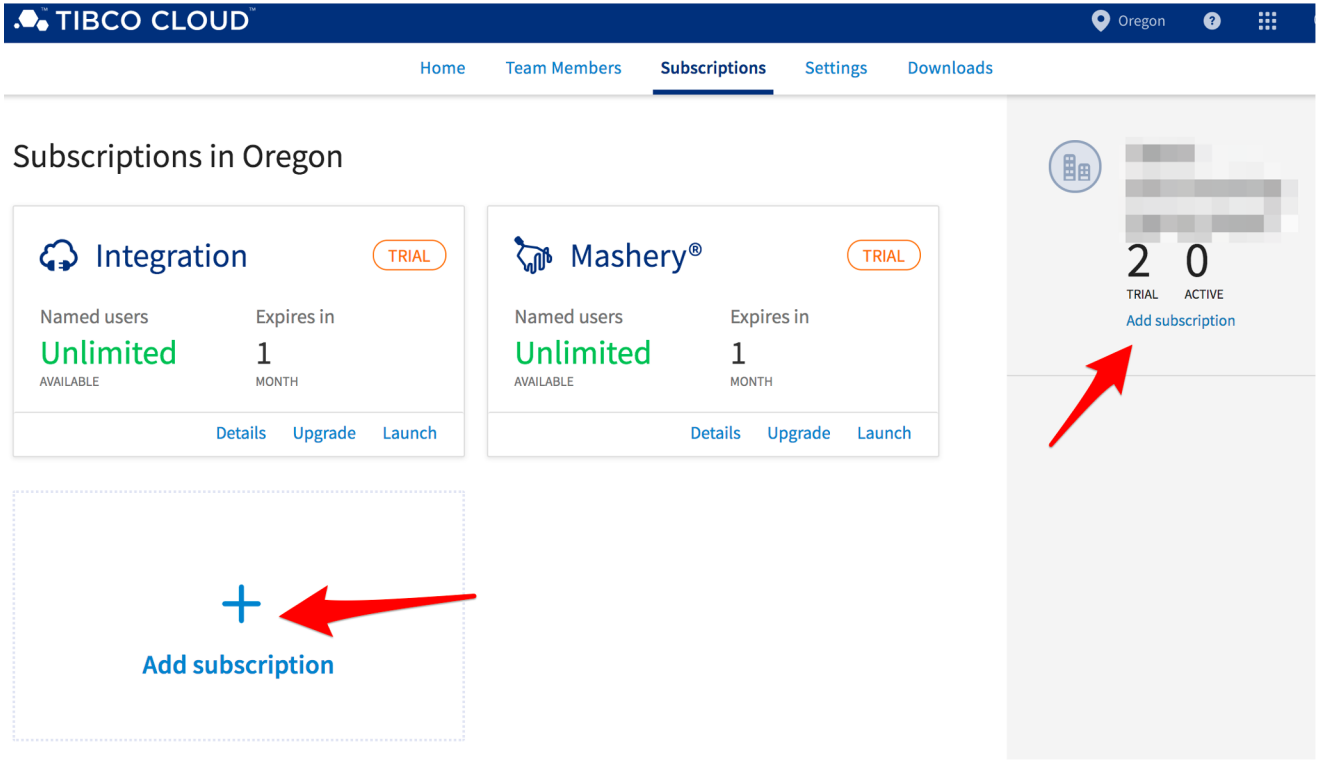
Docs TIBCO : https://docs.tibco.com/

Sign Up

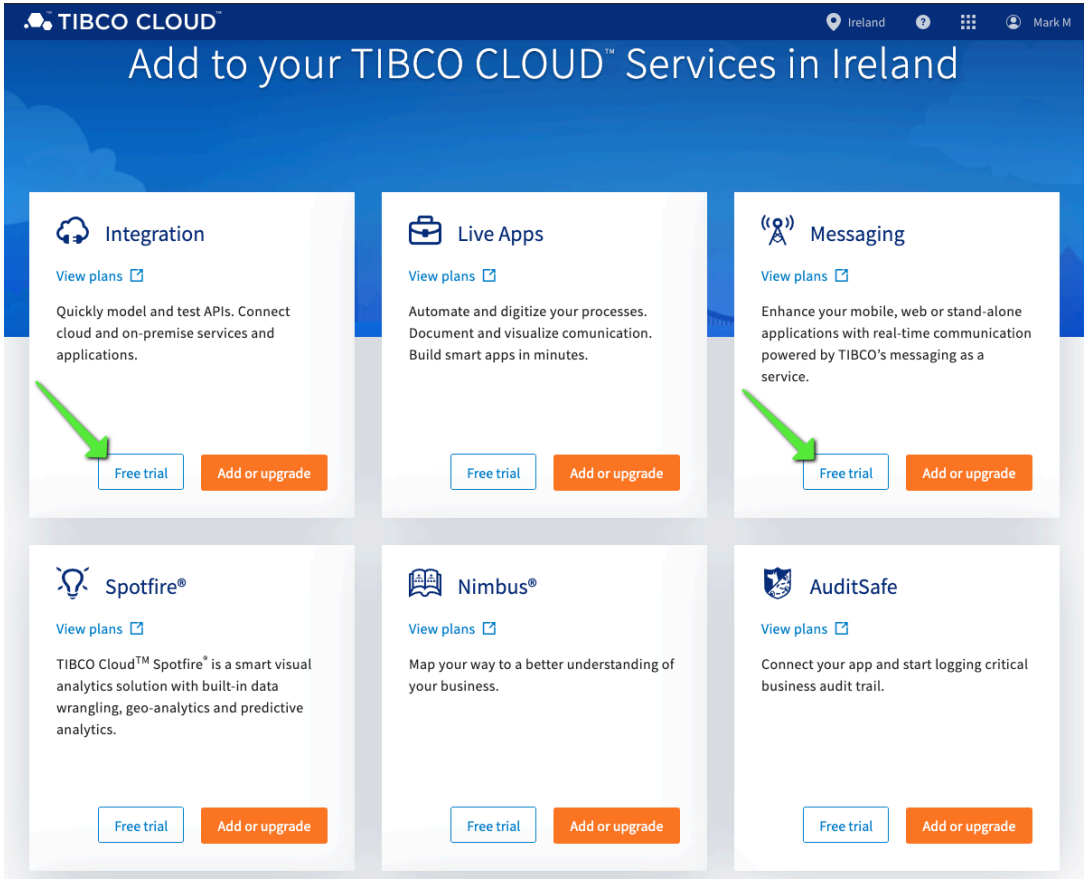
- Go to cloud.tibco.com
  - https://cloud.tibco.com
  - Register
    - Login: your email address
    - Password: your password
    - Region: **EU** (**Ireland)**
  - Check your email for activation email
    - And Junk folder!



Add Subscriptions to the Trial

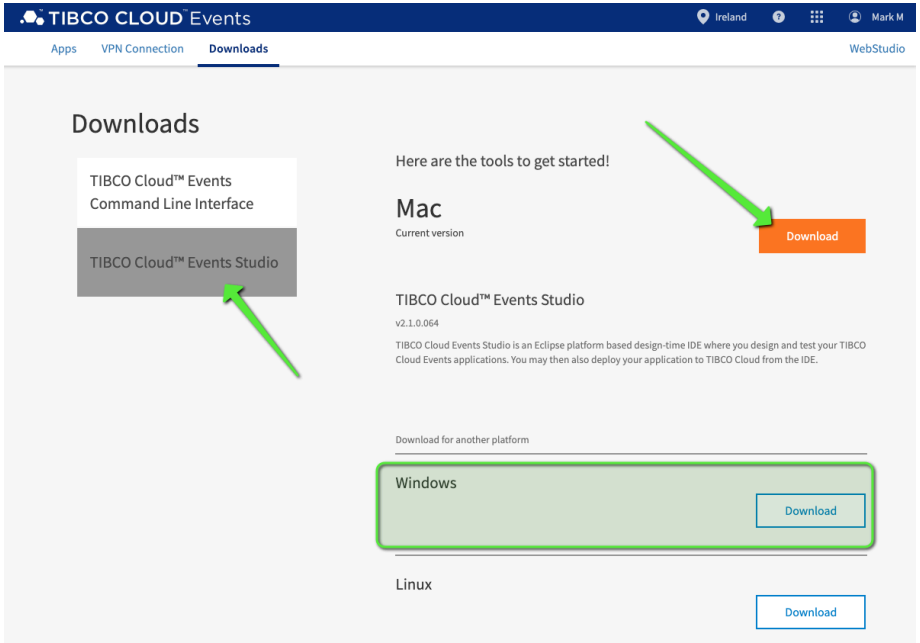


Add Integration & Messaging



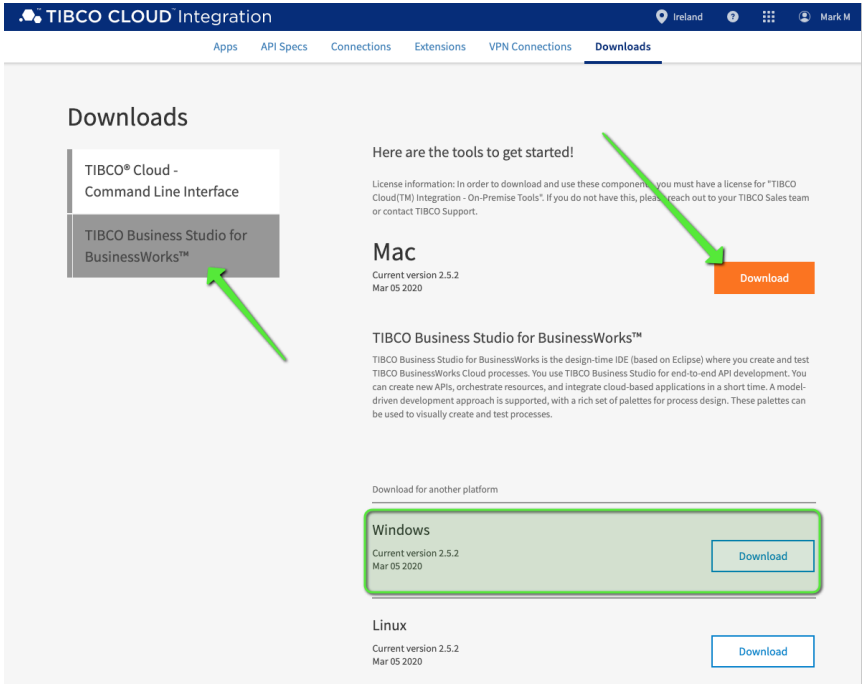
Download and Install TIBCO Cloud Events Studio

https://eu.events.cloud.tibco.com/#!/downloads



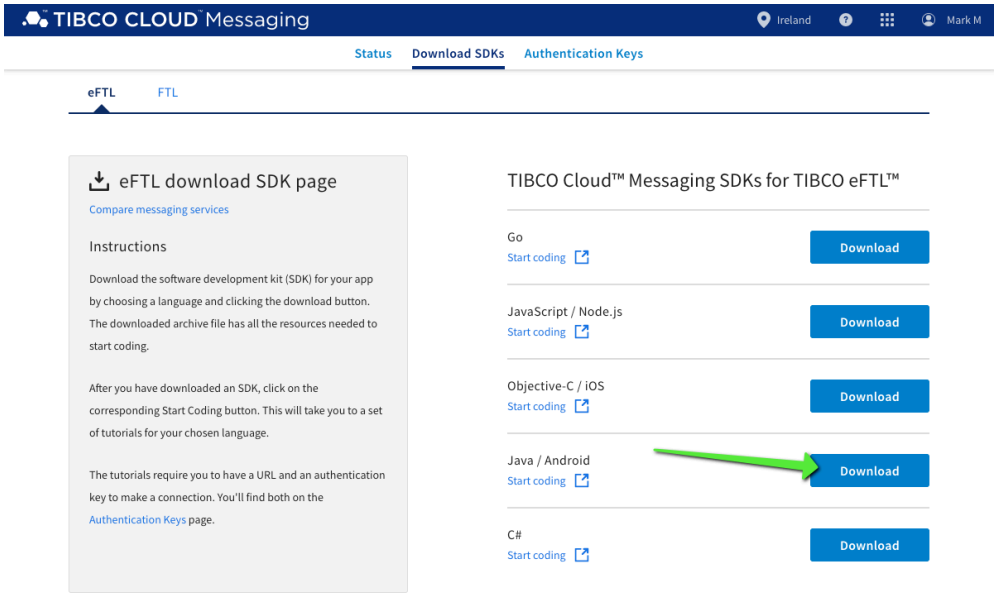
Download and Install TIBCO Business Studio for BusinessWorks

https://eu.integration.cloud.tibco.com/download



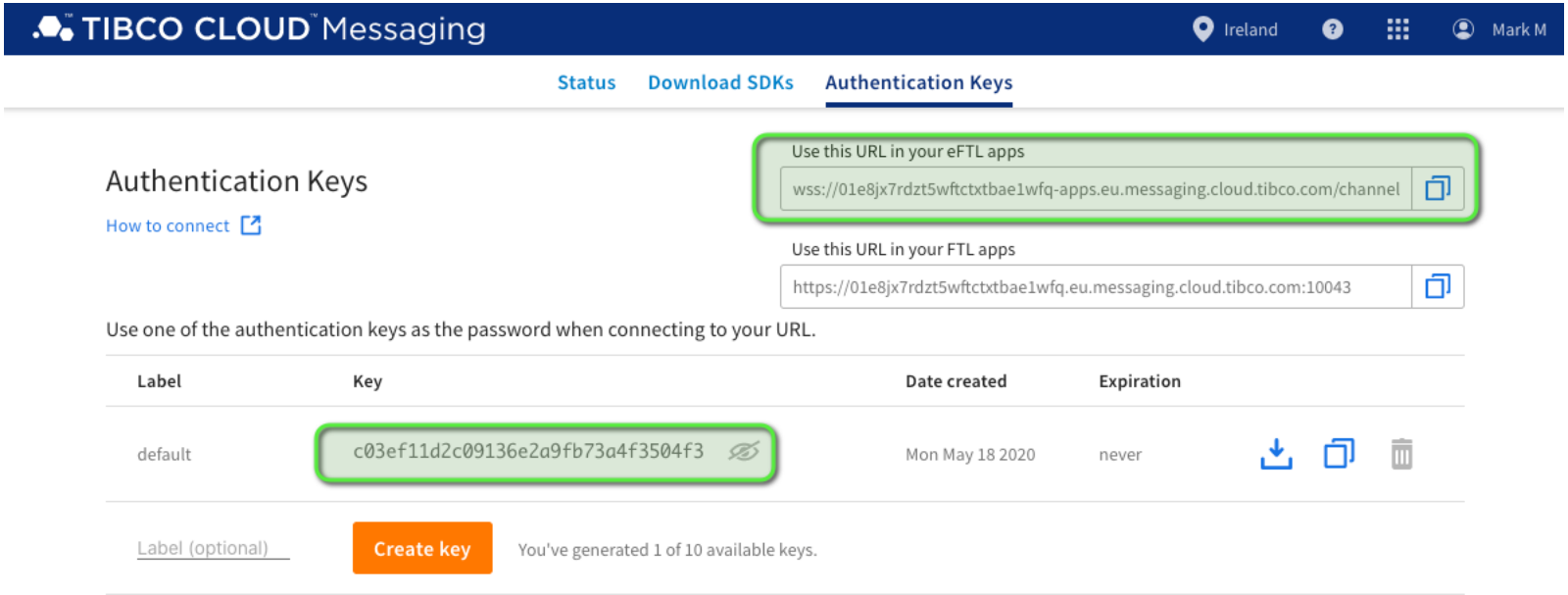
Download TIBCO Cloud Messaging Java SDK

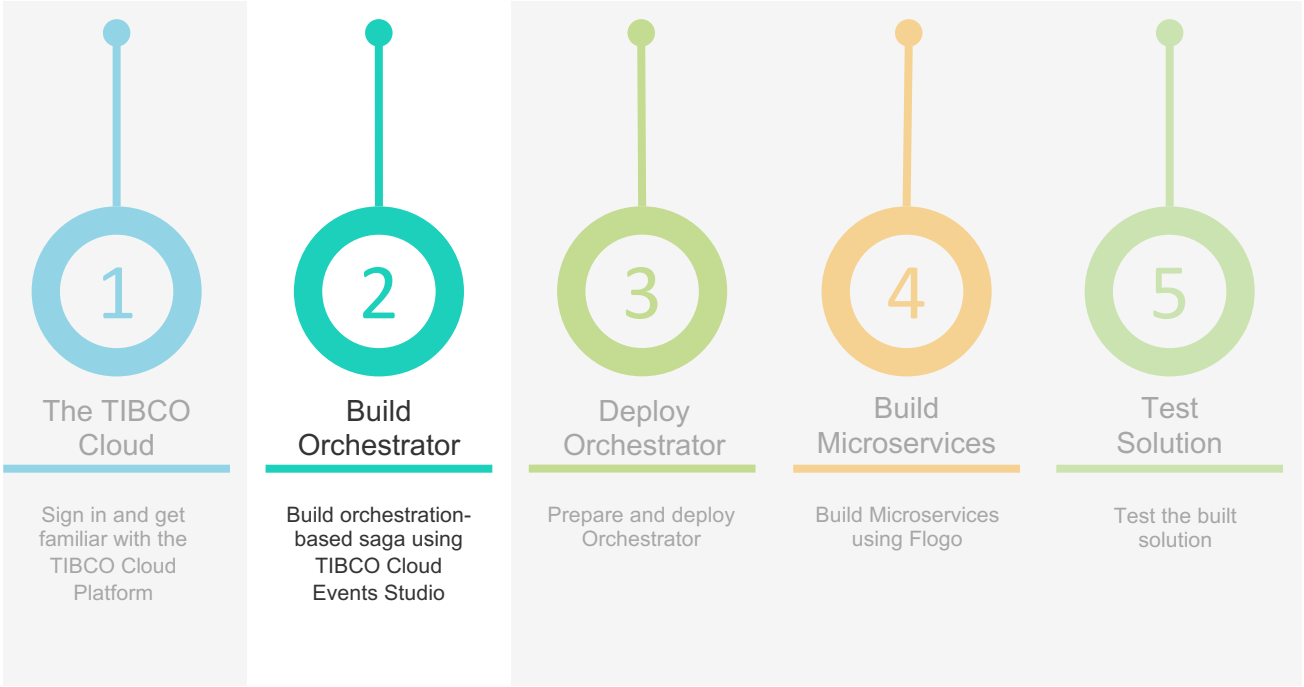
https://eu.messaging.cloud.tibco.com/tcm/ui/downloads



Record TCM Information

https://eu.messaging.cloud.tibco.com/tcm/ui/admin




1
The TIBCO Cloud
Sign in and get familiar with the TIBCO Cloud Platform
2
Build Orchestrator
Build orchestration-based saga using TIBCO Cloud Events Studio
3
Deploy Orchestrator
Prepare and deploy Orchestrator
4
Build Microservices
Build Microservices using Flogo
5
Test Solution
Test the built solution

Lab Objectives

In this lab, you will create and deploy a TIBCO Cloud™ Events solution that will implement an orchestration-based saga based on a very simple Order use-case:

1. Take Payment - implemented by Payment Microservice
2. Prepare Order - implemented by Stock Microservice
3. Ship Order - implemented by Delivery Microservice

Getting Started with TIBCO Cloud™ Events Studio - Create Project

2

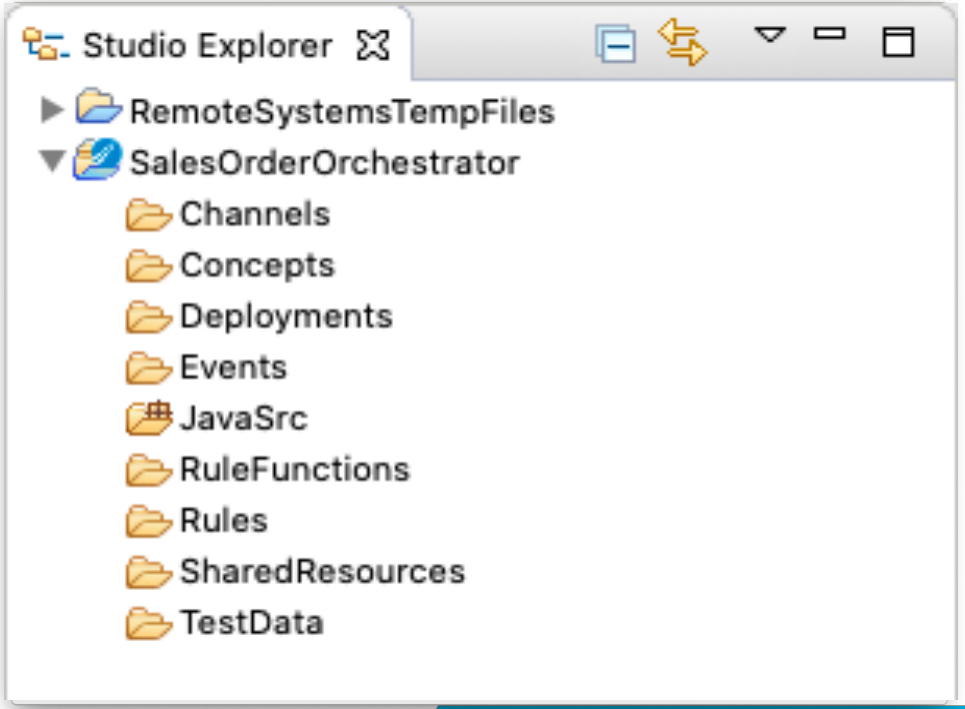
Build Orchestrator

Let's start, open TIBCO Cloud™ Events Studio and create a new Workspace, your screen should look like this:

Right click in the Studio Explorer view, select New->Studio Project...

Enter 'SalesOrderOrchestrator' in the Project name field and click 'Finish' button.

Expand the Project list within the Studio Explorer view to show your newly created project...



Create Concept Definitions

Build Orchestrator

Within TIBCO Cloud Events, a Concept definition is used to define objects and their properties. Concepts are similar to object-oriented concept of a class in that they represent class-level information.
At runtime, the instance of a concept are called objects and are persisted in memory or database.

For our use-case we are going to create 4 concepts:

1. **Order** - holds the Order properties (Order ID , Customer ID, email, and Order State)

2. **Order Item** - holds the Order Items (Item Unit Price, Quantity Ordered, Line Price)

3. **Shipping Address** - holds the Order Shipping properties (Address Lines, City, Country, and Postcode)

4. **InFlightOrders** - tracks Orders ID

Create Shipping Address Concept

2

Build Orchestrator

Right click in the Studio Explorer view, select New >Concept...

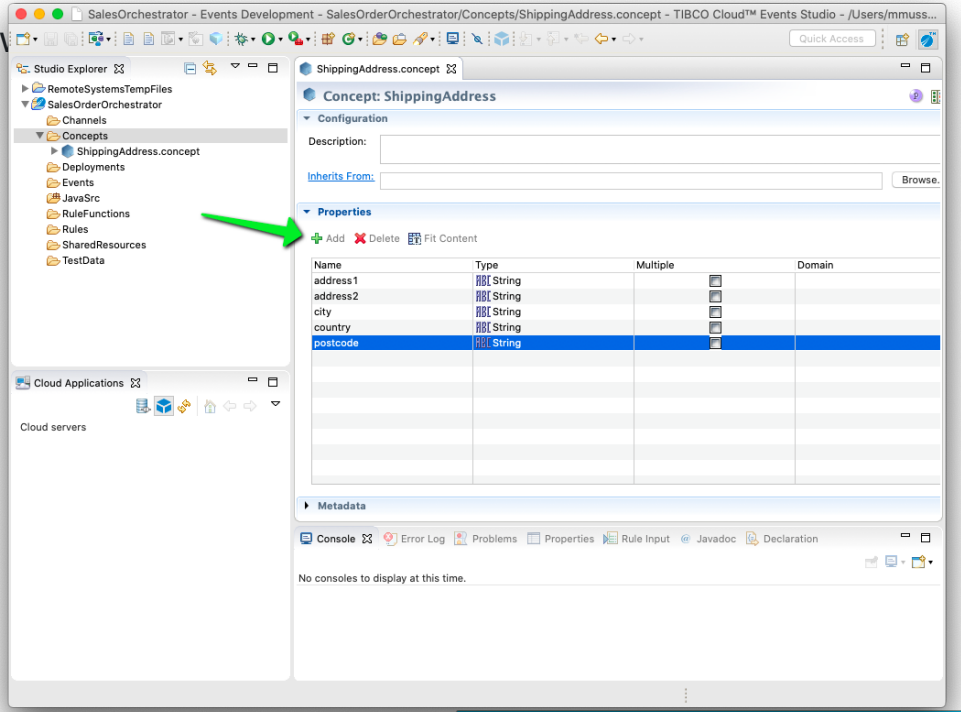
A dialog box will appear for you to create a new concept.

Enter 'ShippingAddress' in the concept name field. Click 'Finish' button.

Using the Concept Properties editor, click + and add the following 5 properties:

- address1 - Type=String
- address2 - Type=String
- city - Type=String
- country - Type=String
- postcode - Type=String

Click 'Save' icon on the toolbar.



Create Order Item Concept

2

Build Orchestrator

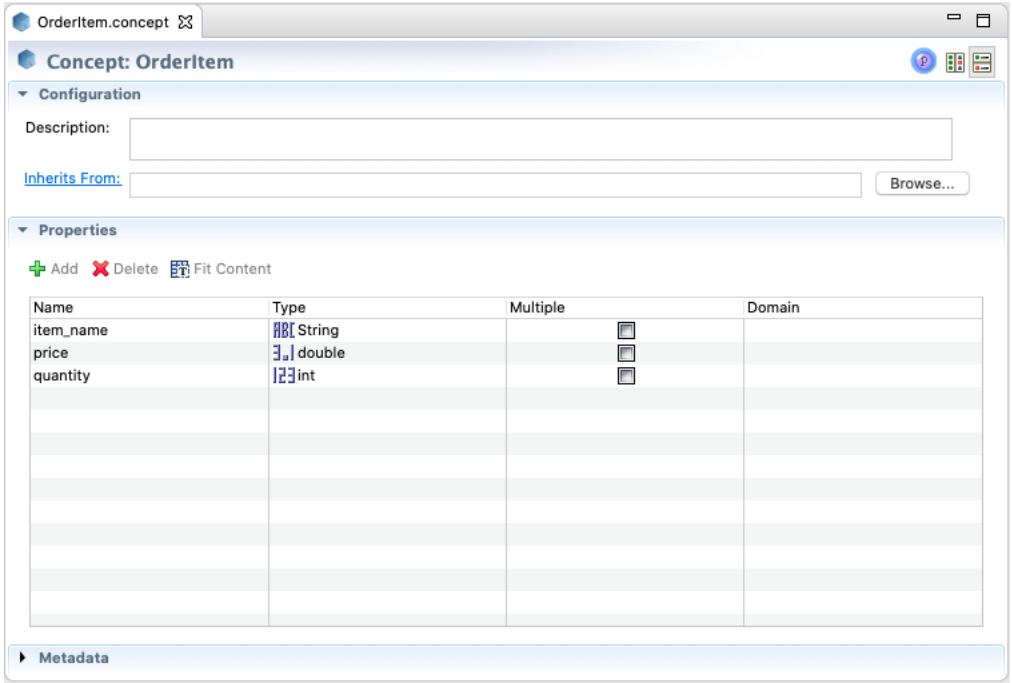
Now that you've created your first concept, go ahead and create Order Item concept.

This time use 'OrderItem' as the Concept name.

Add the following 3 properties to the Order Item concept:

- item_name - Type=String
- price - Type=double
- quantity - Type=int

Click 'Save' icon on the toolbar.



Create In-Flight Order Concept

2

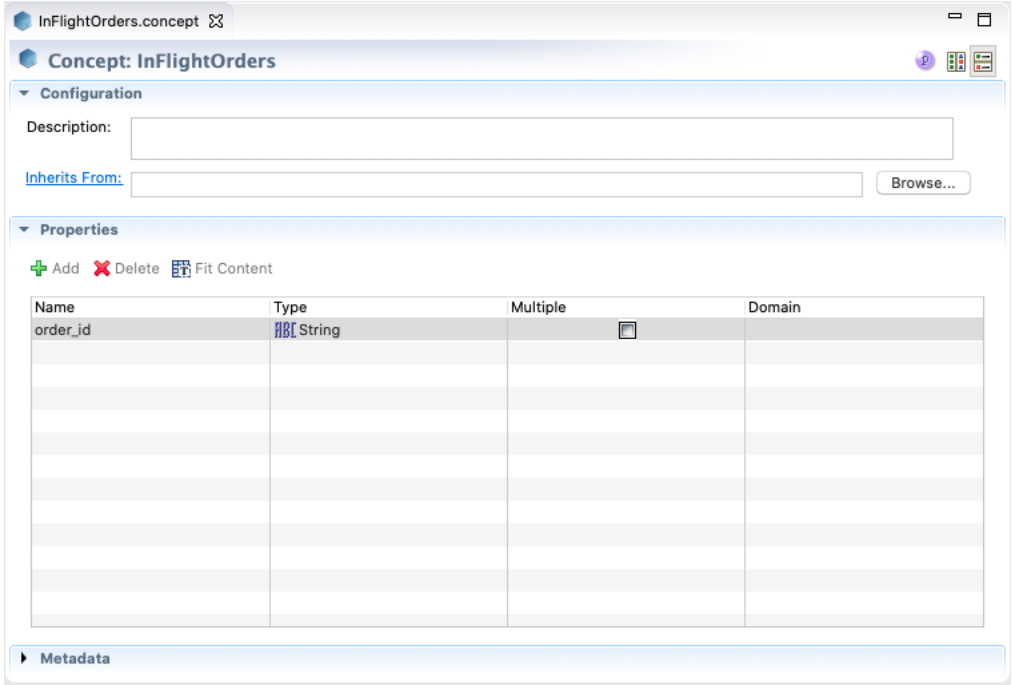
Build Orchestrator

You should be a 'dab-hand' now at creating concepts, go ahead and create In-flight Order Concept.
This time use 'InFlightOrders' as the Concept name.

Add the following property to the In-Flight Order concept:

- order_id - Type=String

Click 'Save' icon on the toolbar.



Create Order Concept

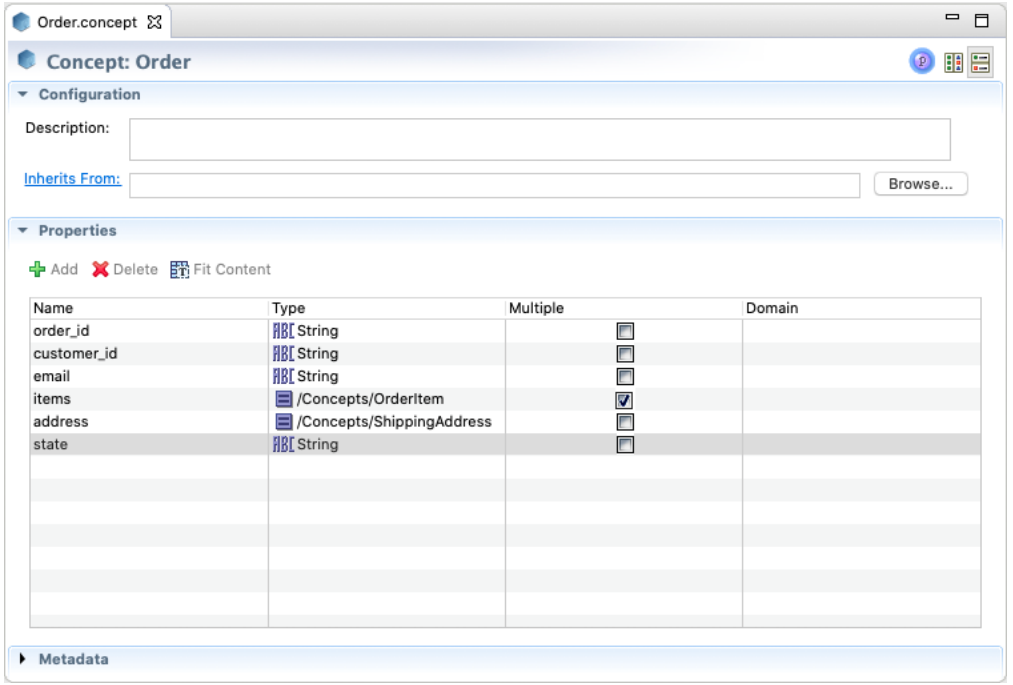
2

Build Orchestrator

Create Order Concept. This time use 'Orders' as the Concept name.

Add the following property to the In-Flight Order Concept:

- order_id - Type=String
- customer_id - Type=String
- email - Type=String
- items - Type=ContainedConcept, Resource=OrderItem, Multiple=enabled
- address - Type=ContainedConcept, Resource=ShippingAddress



Click 'Save' icon on the toolbar.

Create Connection Shared Resource Definitions

We need to get data in and out of TIBCO Cloud Events, to do this we need to create two connections:

- HTTP Connection - this will be used by TIBCO Cloud Events to receive the initial Order JSON request.
- TCM Connection - this will be used by TIBCO Cloud Events to send & receive asynchronous messages with.

Shared Resources can be created in TIBCO Cloud Events and used by components of the solution.

Shared Resources configure the connection and contain properties such as host address, URL, or passwords.

Create HTTP Connection

2

Build Orchestrator

Right click in the Studio Explorer view, select New->Other...

Enter 'HTTP' in the Wizards field, click 'Next >' button.

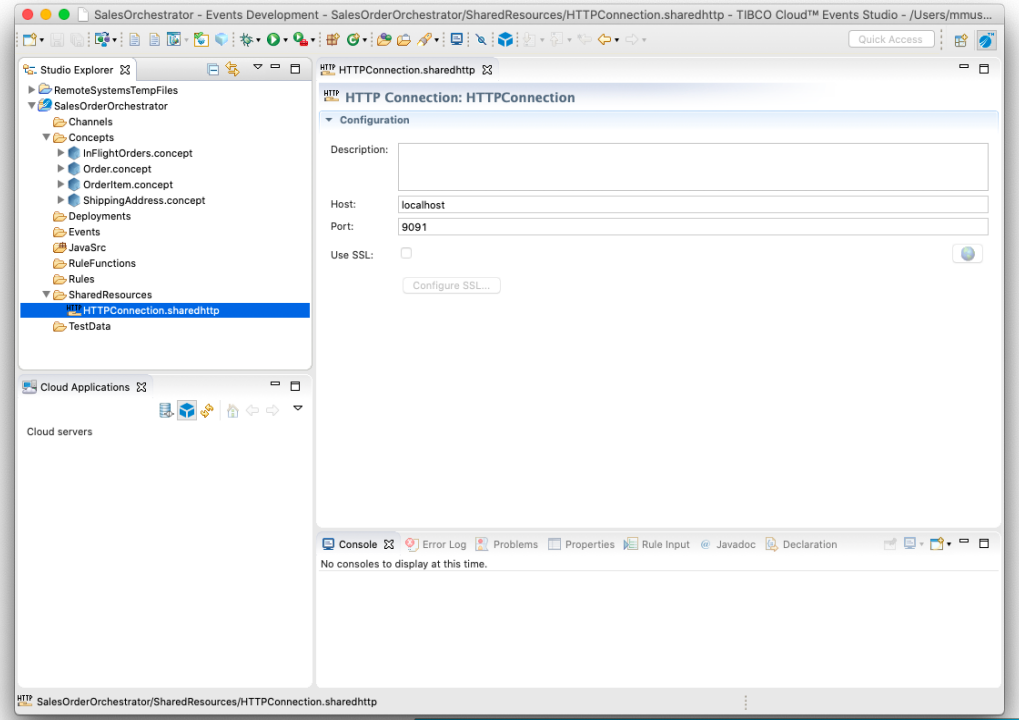
Select 'SharedResources' folder.

Enter 'HTTPConnection' in the Filename field, click 'Finish'.

Enter 'localhost' in the Host field in the HTTP Connection view.

Enter '9091' in the Port field in the HTTP Connection view.

Click 'Save' icon on the toolbar.



Create TCM Connection

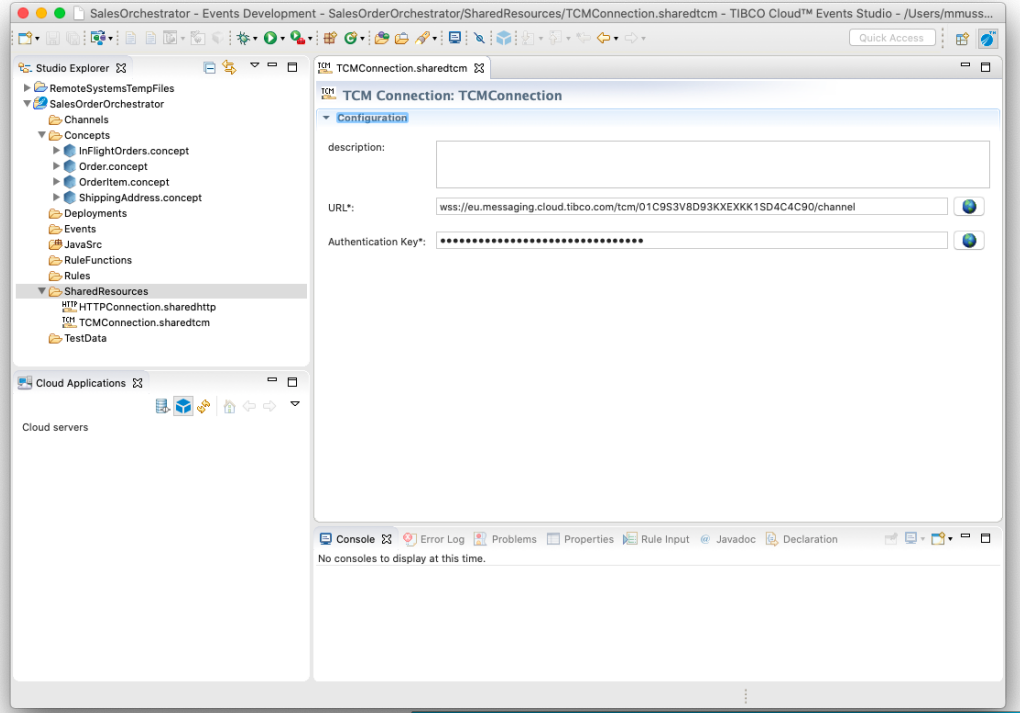
Right click in the Studio Explorer view, select New->Other...

Enter 'TCM' in the Wizards field, click 'Next >' button.

Select 'SharedResources' folder.

Enter 'TCMConnection' in the Filename field, click 'Finish'.

Paste in the TCM URL (make sure to over-write what's there) in the URL field in the TCM Connection view.
Paste in the TCM Authentication Key in the Authentication Key field in the TCM Connection view.
Click 'Save' icon on the toolbar.



Create Channels & Destinations

Channels are the messaging transport where events enter or exit TIBCO Cloud Events.

A channel uses connection resource.

A channel may have one or more destination, which represents a source of messages from that connection resource as well as a sink of messages that are to be sent to the outside.

Whenever we wish to interact with participants either sending or receiving messages we do so through Channels and Destinations.

For our use-case we are going to create 4 channels:

- **Order** - HTTP Channel used for sending in an Order request.
- **Payment** - TCM Channel used to communicate with the Payment Microservice using async messaging.
- **Stock** - TCM Channel used to communicate with the Stock Microservice using async messaging.
- **OrderDelivery** - TCM Channel used to communicate with the Delivery Microservice using async messaging.

Create Order Channel

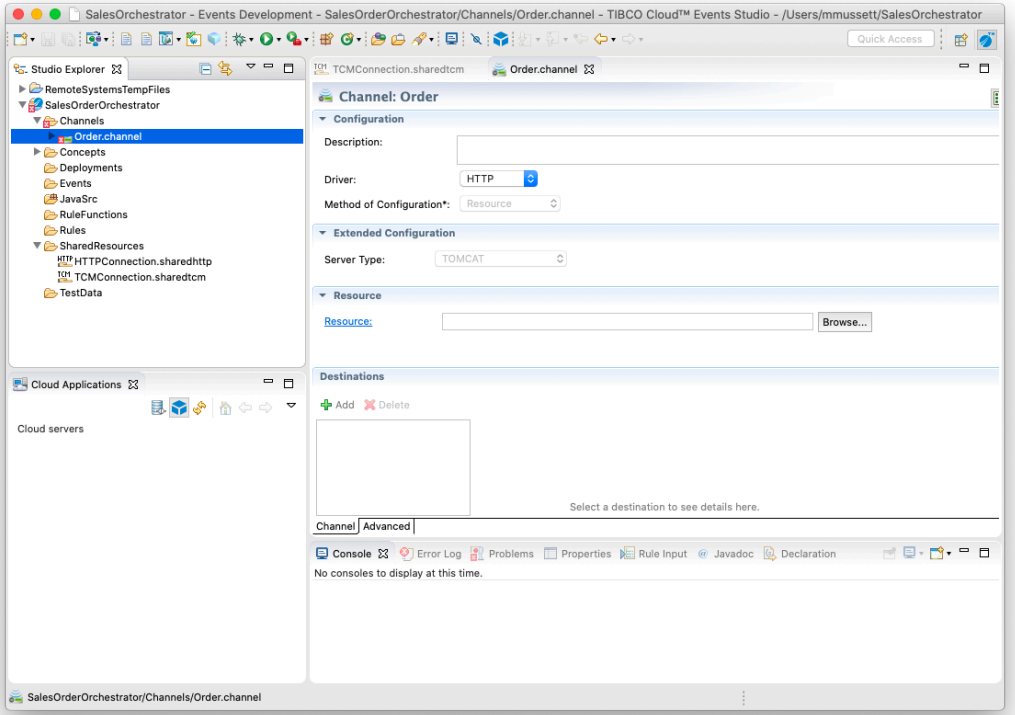
Right click in the Studio Explorer view, select New->Channel...

Enter '**Order**' in the Channel name field, select 'HTTP' as the Driver Type and click 'Finish' button.

Hit the Resource 'Browse...' button, a Select Resource dialog will be shown.

Select the HTTPConnection Shared Resource, then click 'OK'.

Click 'Save' icon on the toolbar.



Create Destinations on Order Channel

2

Build Orchestrator

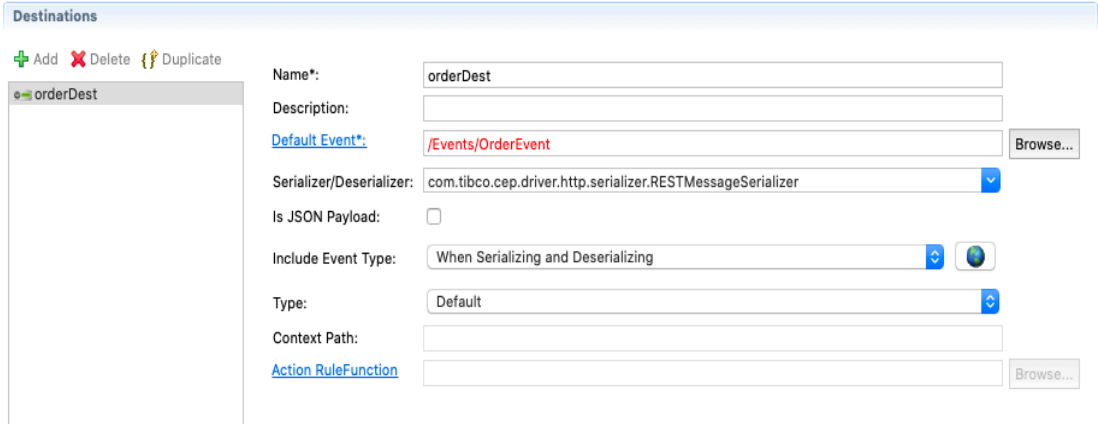
Click '+' in the Destinations panel...

A new destination will be created called 'NewDestination_0'.

Set the Name field to '**OrderDest**'.

Set the Default Event to '/Events/OrderEvent'.

Click 'Save' icon on the toolbar.



Create Payments Channel

Right click in the Studio Explorer view, select New->Channel...

A dialog box will appear for you to create a new channel. Enter **'Payment'** in the Channel name field, select 'TCM' as the Driver Type and click 'Finish' button.

Hit the Resource 'Browse...' button, a Select Resource dialog will be shown.

Select the TCMConnection Shared Resource, then click 'OK'.

Click 'Save' icon on the toolbar.

Create Destinations on Payments Channel

We need 3 destinations for the **Payment** Channel.

Each destination will serve to send or receive asynchronous messages to the Payment microservice over TIBCO Cloud Messaging:

- **ExecutePayment** - used to send payment request.
- **ExecuteRefund** - used to send refund of payment request.
- **PaymentExecuted** - used to receive acknowledgement of payment.

Name	Default Event	TCM Destination	Durable Name	Durable Subscription Type	Is JSON Payload
ExecutePayment	/Events/PaymentEvent	payment		Standard Durable	Yes
ExecuteRefund	/Events/RefundEvent	refund		Standard Durable	Yes
PaymentExecuted	/Events/PaymentExecutedEvent	paymentexecuted		Standard Durable	Yes

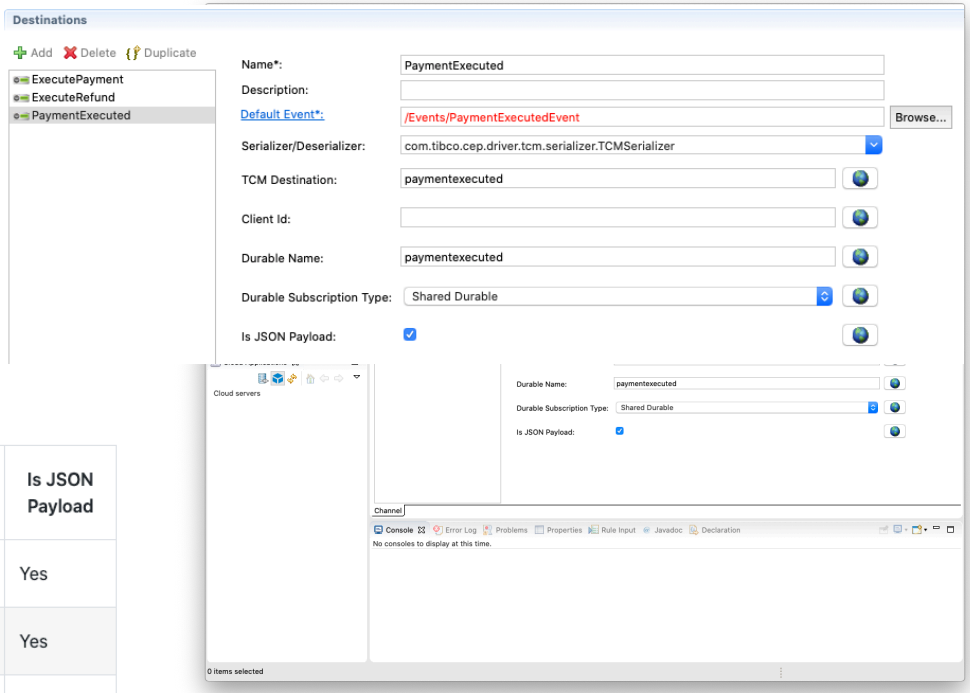
Create Destinations on Payments Channel

Click '+' in the Destinations panel

Configure each field with the correct row information.

Click 'Save' icon on the toolbar.

Repeat the above steps using the information provided in the table to create the 3 destinations.



Name	Default Event	TCM Destination	Durable Name	Durable Subscription Type	Is JSON Payload
ExecutePayment	/Events/PaymentEvent	payment		Standard Durable	Yes
ExecuteRefund	/Events/RefundEvent	refund		Standard Durable	Yes
PaymentExecuted	/Events/PaymentExecutedEvent	paymentexecuted		Standard Durable	Yes

Create Stock Channel

Right click in the Studio Explorer view, select New->Channel...

A dialog box will appear for you to create a new channel. Enter '**Stock**' in the Channel name field, select 'TCM' as the Driver Type and click 'Finish' button.

Hit the Resource 'Browse...' button, a Select Resource dialog will be shown.

Select the TCMConnection Shared Resource, then click 'OK'.

Click 'Save' icon on the toolbar.

Create Destinations on Stock Channel

We need 2 destinations for the **Stock** Channel.

Each destination will serve to send or receive asynchronous messages to the Stock microservice over TIBCO Cloud Messaging:

- **PrepareOrder** - used to send order preparation request.
- **OrderPrepared** - used to receive acknowledgement that the order has been prepared.

Name	Default Event	TCM Destination	Durable Name	Durable Subscription Type	Is JSON Payload
PrepareOrder	/Events/PrepareOrderEvent	prepare		Standard Durable	Yes
OrderPrepared	/Events/OrderPreparedEvent	prepared		Standard Durable	Yes

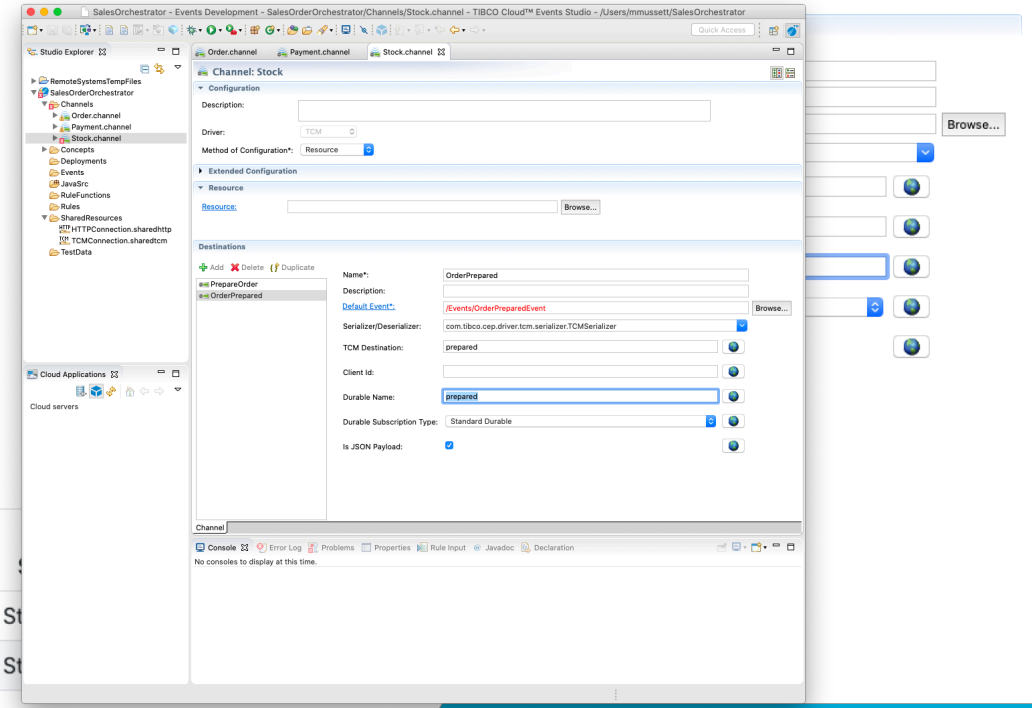
Create Destinations on Stock Channel

Click '+' in the Destinations panel

Configure each field with the correct row information.

Click 'Save' icon on the toolbar.

Repeat the above steps using the information provided in the table to create the 2 destinations.



Name	Default Event	TCM Destination	Durable Name	S
PrepareOrder	/Events/PrepareOrderEvent	prepare		St
OrderPrepared	/Events/OrderPreparedEvent	prepared		St

Create OrderDelivery Channel

Right click in the Studio Explorer view, select New->Channel...

A dialog box will appear for you to create a new channel. Enter '**OrderDelivery**' in the Channel name field, select 'TCM' as the Driver Type and click 'Finish' button.

Hit the Resource 'Browse...' button, a Select Resource dialog will be shown.

Select the TCMConnection Shared Resource, then click 'OK'.

Click 'Save' icon on the toolbar.

Create Destinations on OrderDelivery Channel

We need 2 destinations for the **OrderDelivery** Channel.

Each destination will serve to send or receive asynchronous messages to the Delivery microservice over TIBCO Cloud Messaging:

- **DeliverOrder** - used to send order delivery request.
- **OrderDelivered** - used to receive acknowledgement that the order has been delivered.

Name	Default Event	TCM Destination	Durable Name	Durable Subscription Type	Is JSON Payload
DeliverOrder	/Events/DeliverOrderEvent	deliver		Standard Durable	Yes
OrderDelivered	/Events/OrderDeliveredEvent	delivered		Standard Durable	Yes

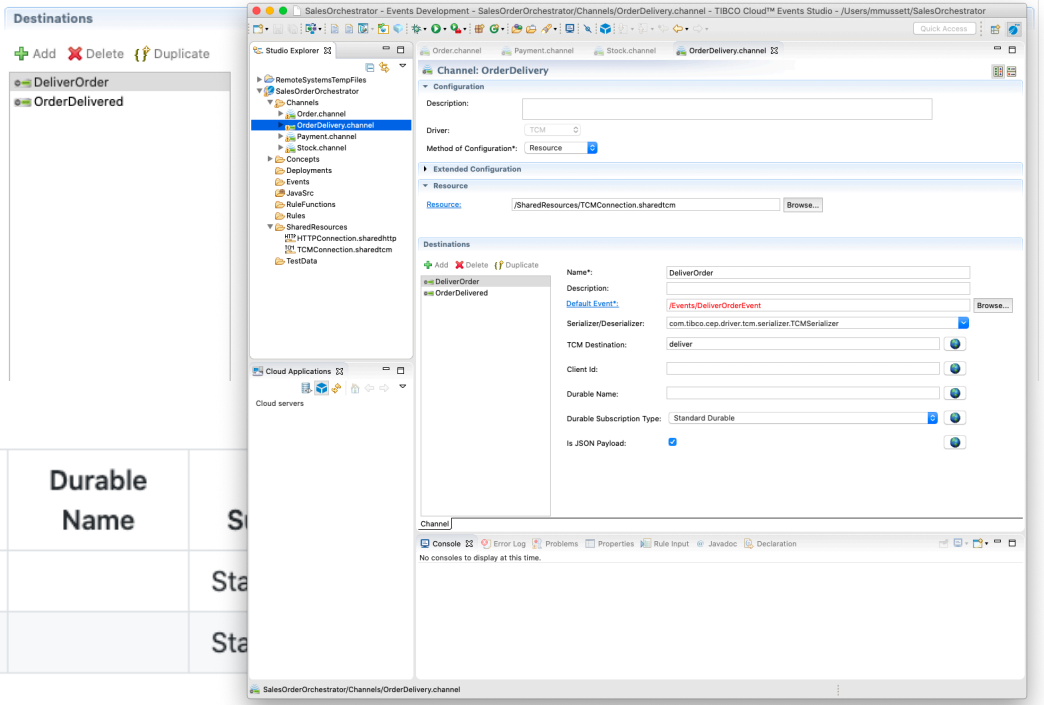
Create Destinations on OrderDelivery Channel

Click '+' in the Destinations panel

Configure each field with the correct row information.

Click 'Save' icon on the toolbar.

Repeat the above steps using the information provided in the table to create the 2 destinations.



Name	Default Event	TCM Destination	Durable Name	S
DeliverOrder	/Events/DeliverOrderEvent	deliver		Sta
OrderDelivered	/Events/OrderDeliveredEvent	delivered		Sta

Create Events

Build
Orchestrator

An event is an object representing some occurrence or point of time. In our use-case events represent business activities such as Take Order Payment or Deliver Order.

When messages arrive on channel destinations we assign these messages to events.

Channel destinations either emit or consume events from external participants.

Events model state similar to concept properties.

Create Events

For our use-case we are going to create 9 events which represent the following interactions:

- **OrderEvent** - A new Order arriving.
- **ReplyOrderEvent** - Acknowledgement for a new Order.
- **PaymentEvent** - A payment for an Order is needed.
- **RefundEvent** - A refund on an Order is needed.
- **PaymentExecuteEvent** - Payment confirmation for an Order.
- **PrepareOrderEvent** - Prepare Order for shipment.
- **OrderedPreparedEvent** - An Order is ready for shipment.
- **DeliverOrderEvent** - Dispatch Order to the customer.
- **OrderDeliveredEvent** - An Order has been delivered to the customer.

Create Order Event

Right click in the Studio Explorer view, select New->Simple Event...

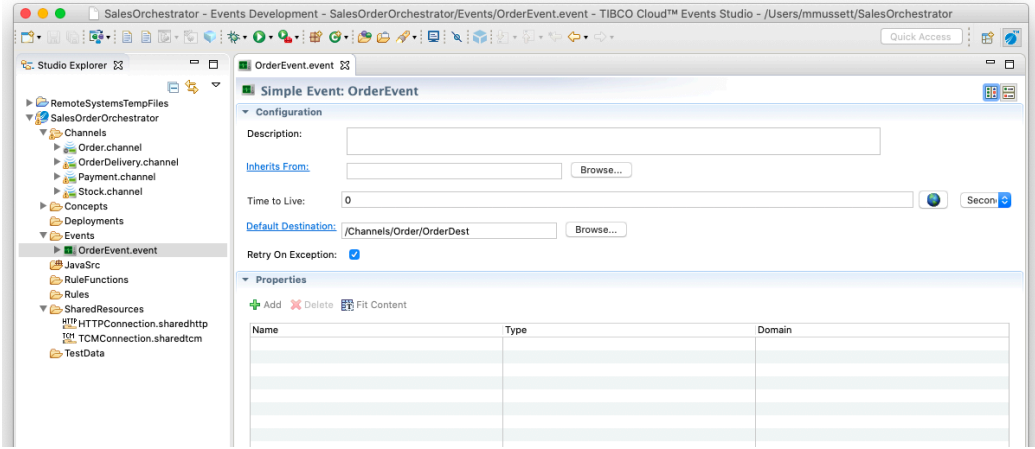
A dialog box will appear for you to create a new Simple Event.

Select 'Events' folder. Enter 'OrderEvent' in the Simple Event name field.

Click 'Finish' button.

Click 'Browse...' button for Default Destination.

A dialog box will appear for you to select a Channel Destination. Select the 'OrderDest' destination on the 'Order' channel and click 'OK' button.



Create Order Event

2

Build Orchestrator

Click the 'Advanced' tab on the Properties section.

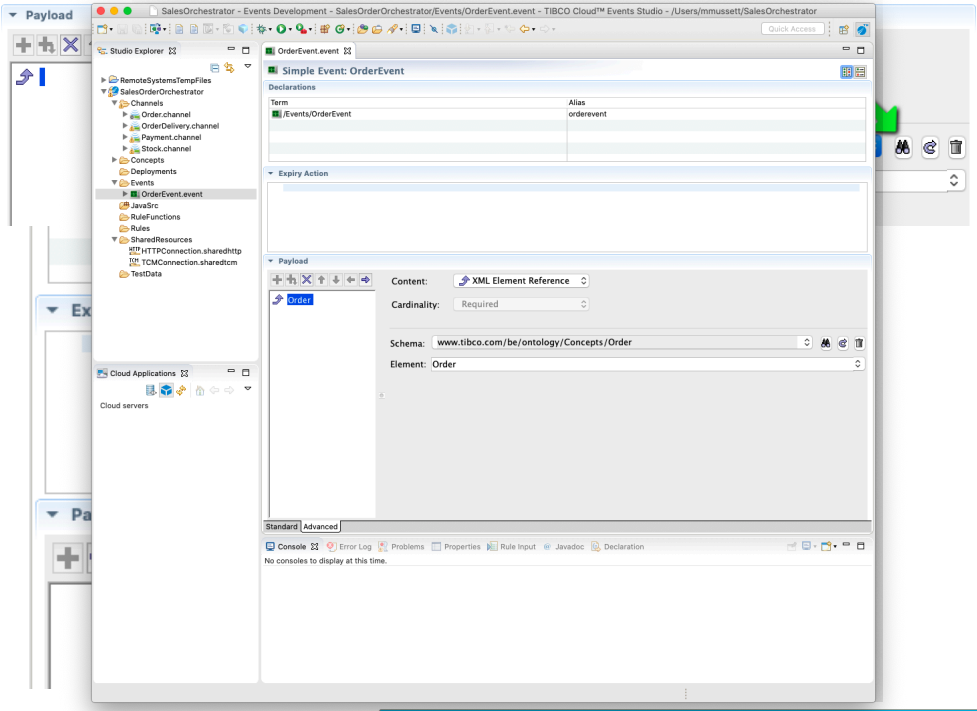
Click the '+' symbol in the 'Payload' section...

Click the 'Content' dropdown arrow and select 'XML Element Reference'...

Click the binocular icon next to the 'Schema' box...

'Select a Resource...' dialog box will be shown.
Navigate the Concepts folder and select the 'Order' concept.
Click 'OK'.

Click 'Save' icon on the toolbar.



Create ReplyOrder Event

Right click in the Studio Explorer view, select New->Simple Event...

A dialog box will appear for you to create a new Simple Event.

Select 'Events' folder. Enter '**ReplyOrderEvent**' in the Simple Event name field.

Click 'Finish' button.

Click 'Browse...' button for Default Destination.

A dialog box will appear for you to select a Channel Destination. Select the '**OrderDest**' destination on the '**Order**' channel and click 'OK' button.

Create ReplyOrder Event

Click the 'Advanced' tab on the Properties section.

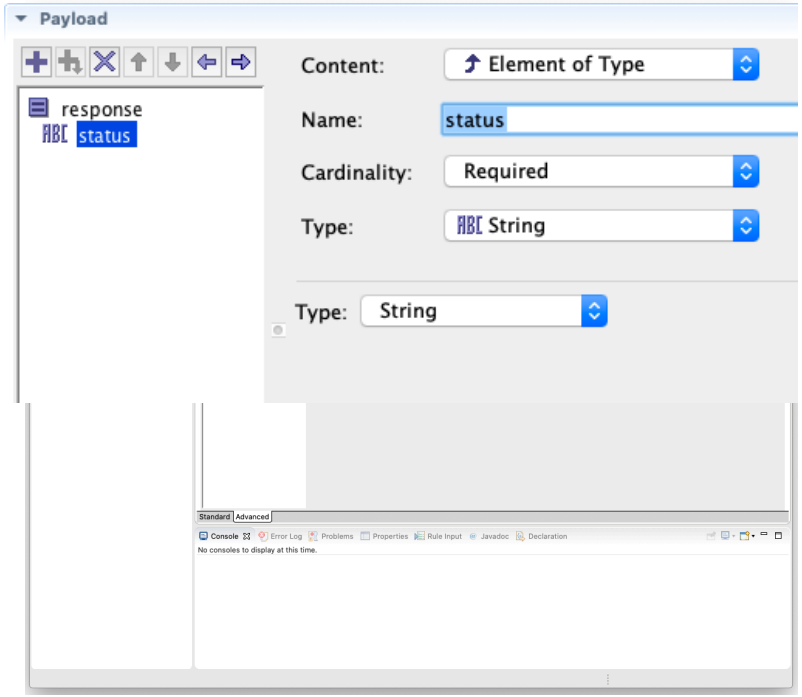
Click the '+' symbol in the 'Payload' section.

Click the '+' symbol again.

Select 'root' Complex Element Content and change the Name field from 'root' to 'response'.

Select 'param' Element and change the Name field from 'param' to 'status'.

Click 'Save' icon on the toolbar.



Create Payment Event

2

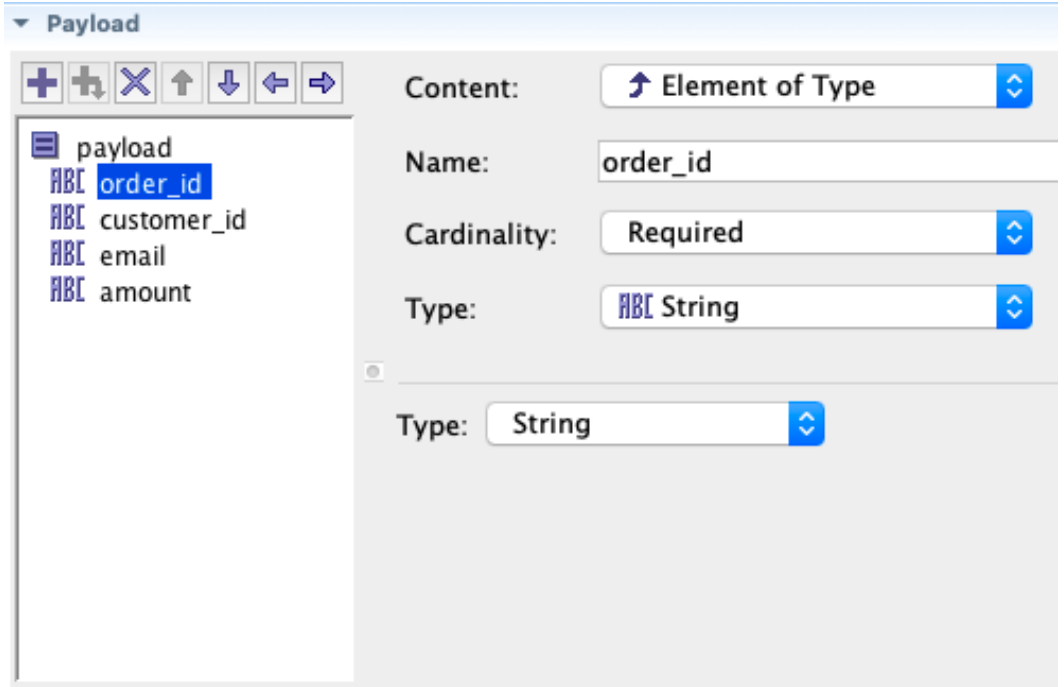
Build Orchestrator

Follow the same steps previously but create the Simple Event named '**PaymentEvent**'.

Set the Default Destination set to '**Payment**' channel and '**ExecutePayment**' destination.

On the 'Advanced' tab, create the Properties structure with a Complex element named 'payload' and 4 child elements named:

- order_id - Content=Element of Type String
- customer_id - Content=Element of Type String
- email - Content=Element of Type String
- amount - Content=Element of Type String



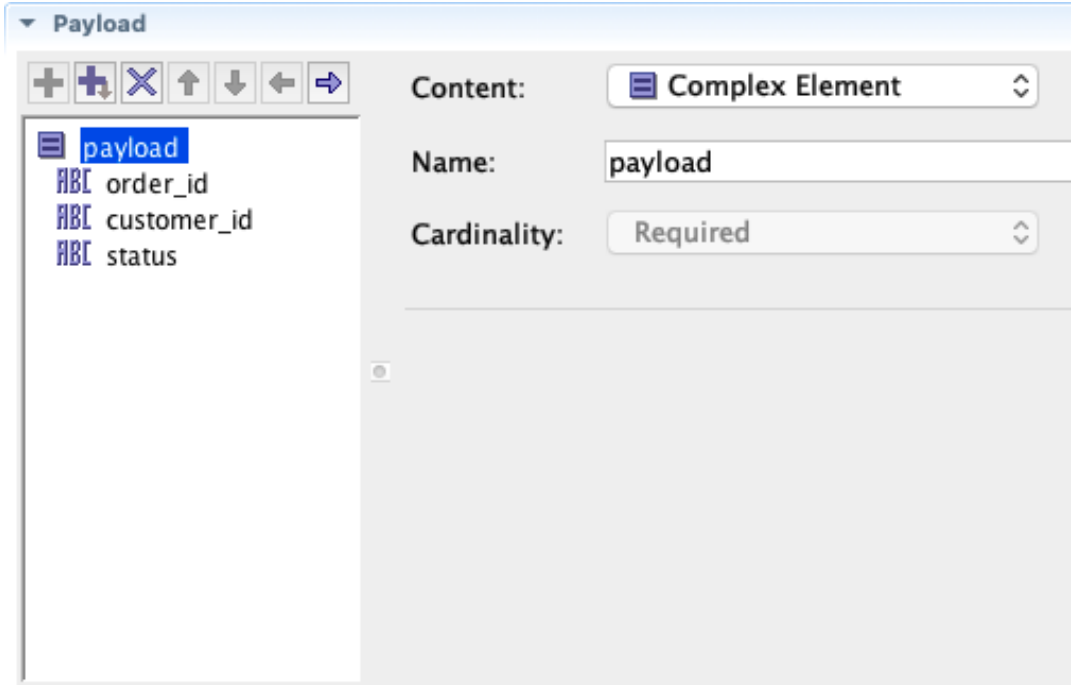
Create PaymentExecuted Event

Create the Simple Event named '**PaymentExecutedEvent**'.

Set the Default Destination set to '**Payment**' channel and '**PaymentExecuted**' destination.

On the 'Advanced' tab, create the Properties structure with a Complex element named 'payload' and 3 child elements named:

- order_id - Content=Element of Type String
- customer_id - Content=Element of Type String
- status - Content=Element of Type String



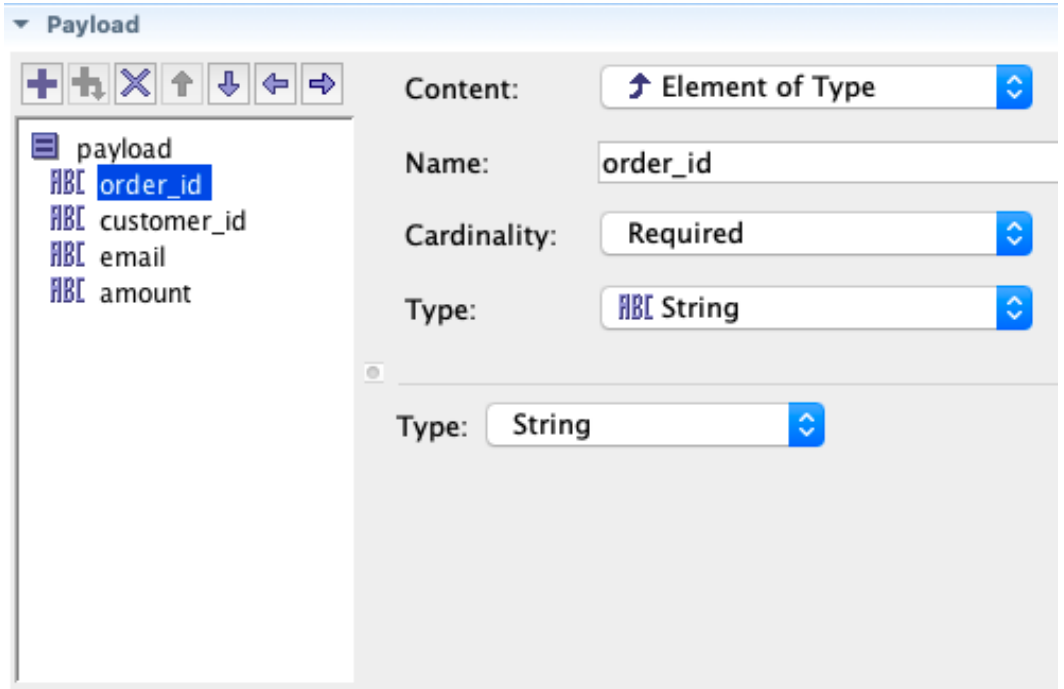
Create Refund Event

Follow the same steps previously but create the Simple Event named '**RefundEvent**'.

Set the Default Destination set to '**Payment**' channel and '**ExecuteRefund**' destination.

On the 'Advanced' tab, create the Properties structure with a Complex element named 'payload' and 4 child elements named:

- order_id - Content=Element of Type String
- customer_id - Content=Element of Type String
- email - Content=Element of Type String
- amount - Content=Element of Type String



Create Prepare Order Event

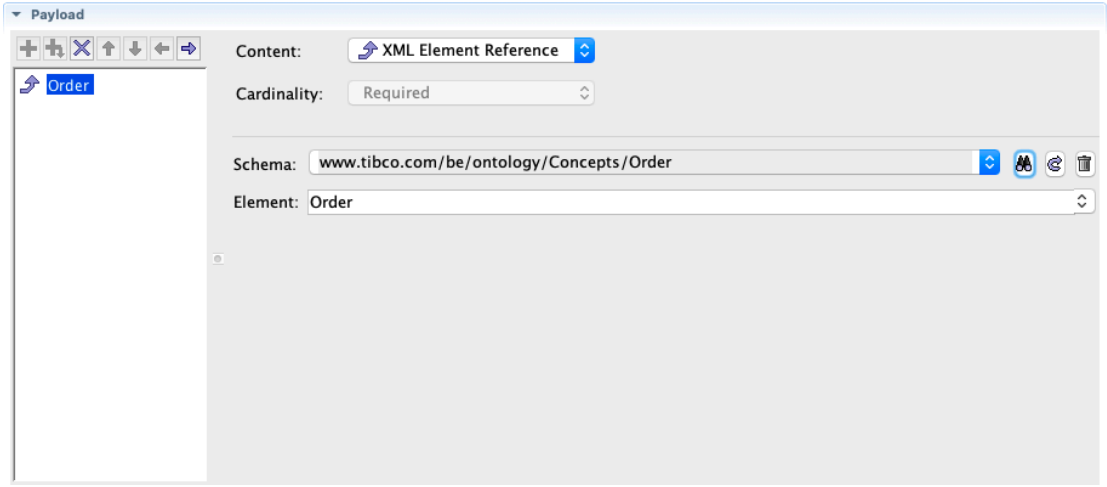
2

Build Orchestrator

Follow the same steps previously but create the Simple Event named '**PrepareOrderEvent**'.

Set the Default Destination set to '**Stock**' channel and '**PrepareOrder**' destination.

On the 'Advanced' tab, create the Properties structure with a XML Element Reference to the 'Order' Concept.



Create OrderPrepared Event

2

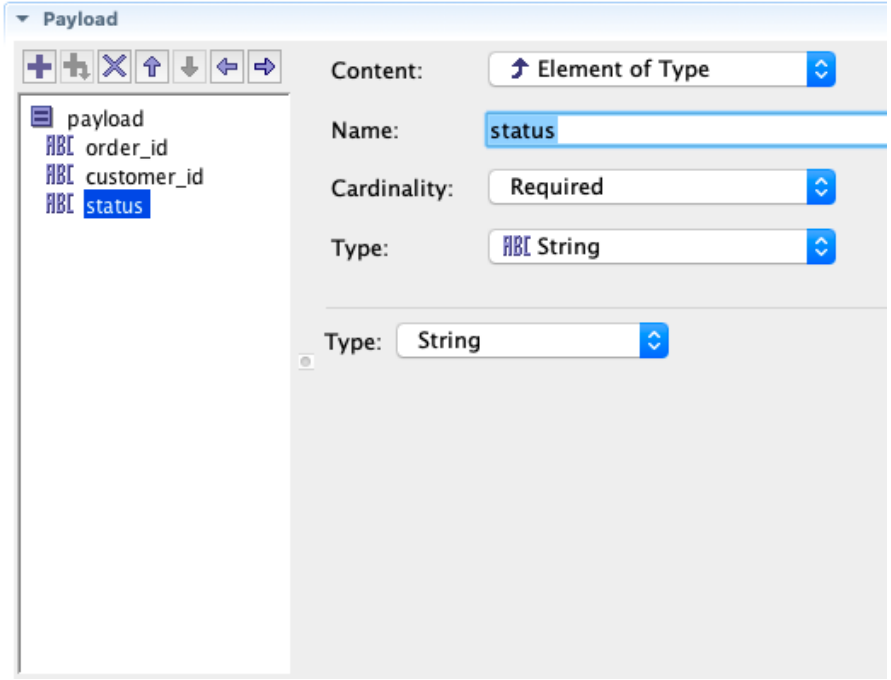
Build Orchestrator

Follow the same steps previously but create the Simple Event named '**OrderPreparedEvent**'.

Set the Default Destination set to '**Stock**' channel and '**OrderPrepared**' destination.

On the 'Advanced' tab, create the Properties structure with a Complex element named 'payload' and 3 child elements named:

- order_id - Content=Element of Type String
- customer_id - Content=Element of Type String
- status - Content=Element of Type String



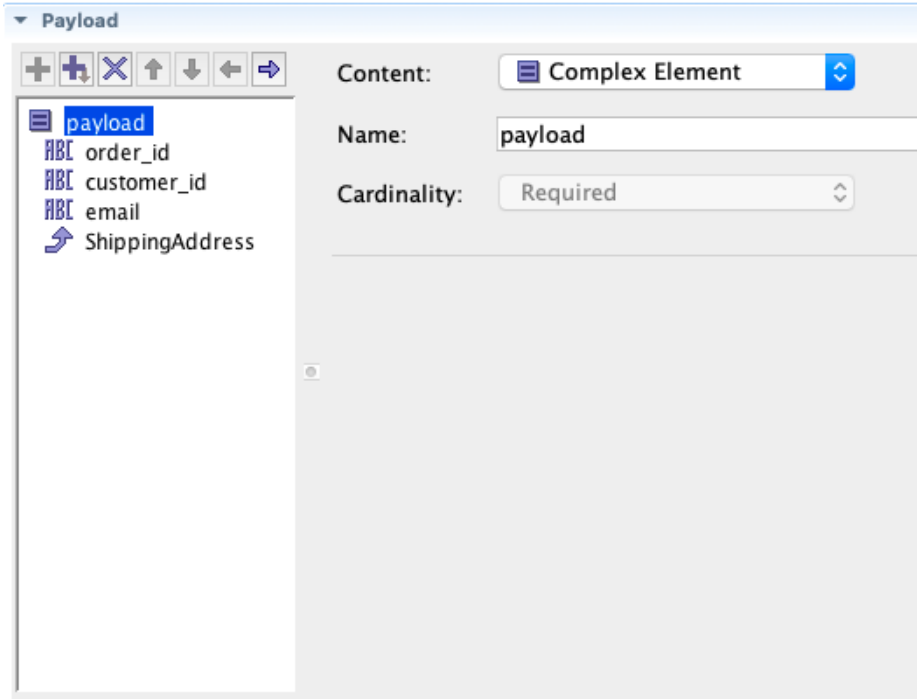
Create DeliverOrder Event

Follow the same steps previously but create the Simple Event named '**DeliverOrderEvent**'.

Set the Default Destination set to '**OrderDelivery**' channel and '**DeliverOrder**' destination.

On the 'Advanced' tab, create the Properties structure with a Complex element named 'payload' and 4 child elements named:

- order_id - Element of Type=String
- customer_id - Element of Type=String
- email - Element of Type=String
- ShippingAddress - XML Element Reference=Concepts/ShippingAddress



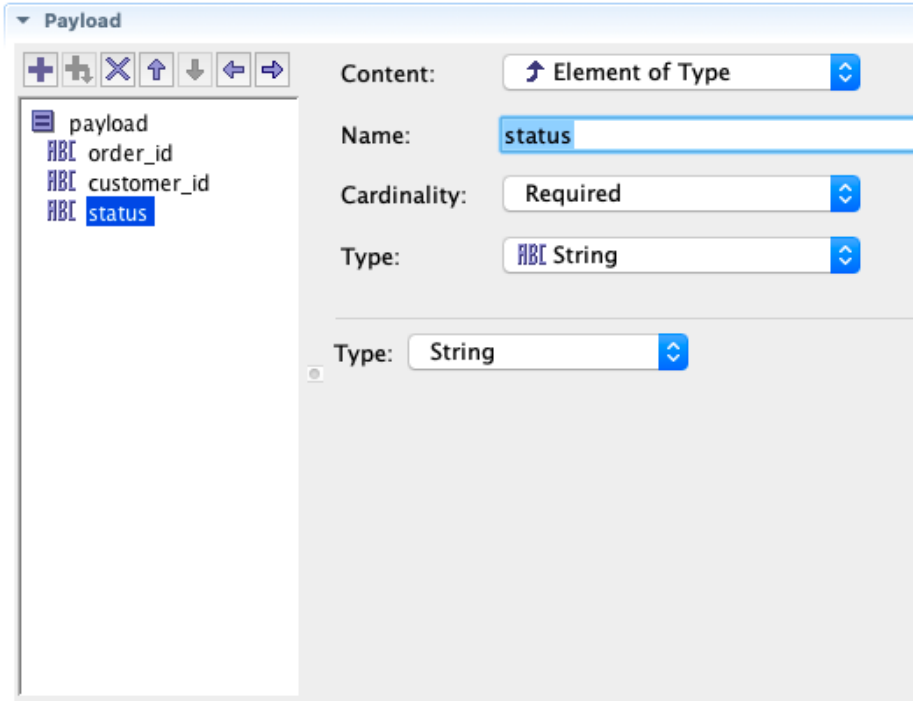
Create OrderedDelivered Event

Follow the same steps previously but create the Simple Event named '**OrderDeliveredEvent**'.

Set the Default Destination set to '**OrderDelivery**' channel and '**OrderDelivered**' destination.

On the 'Advanced' tab, create the Properties structure with a Complex element named 'payload' and 3 child elements named:

- order_id - Content=Element of Type String
- customer_id - Content=Element of Type String
- status - Content=Element of Type String



Create Rules & Rule Functions

Rules are the logic behind the events.

Rules are executed whenever an Event is created or modified within TIBCO Cloud Events.

Rules allow us to codify our business logic using Java.

We use Rules and Events to determine the next action to take.

The TIBCO Cloud Events engine will determine which rules to execute based on predicated logic

e.g. Payment=APPROVED AND Order Status=PREPARED

Create OrderPrepProcessor Rule Function

Right click in the Studio Explorer view, select New->Rule Function...

A dialog box will appear for you to create a new Rule Function.

Select 'RuleFunctions' folder.

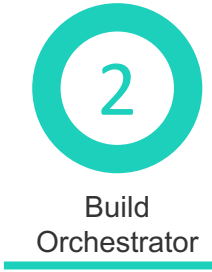
Enter '**OrderPreProcessor**' in the Rule Function name field.

Click 'Finish' button.

Paste the contents of the OrderPreProcessor.rulefunction text file located in src/TCE/SalesOrderOrchestrator/RuleFunctions github repository.

Click 'Save' icon on the toolbar.

Create CalculateOrderAmount Rule Function



Right click in the Studio Explorer view, select New->Rule Function...

A dialog box will appear for you to create a new Rule Function. Select 'RuleFunctions' folder.

Enter '**CalculateOrderAmount**' in the Rule Function name field.

Click 'Finish' button.

Paste the contents of the CalculateOrderAmount.rulefunction text file located in src/TCE/SalesOrderOrchestrator/RuleFunctions github repository.

Click 'Save' icon on the toolbar.

Create Order Rule

Right click in the Studio Explorer view, select New->Rule...

A dialog box will appear for you to create a new Rule. Select 'Rule' folder.

Enter '**OrderRule**' in the Rule Function name field.

Click 'Finish' button.

Paste the contents of the OrderRule.rule text file located in src/TCE/SalesOrderOrchestrator/Rule github repository.

Click 'Save' icon on the toolbar.

Create PaymentExecute Rule

Right click in the Studio Explorer view, select New->Rule...

A dialog box will appear for you to create a new Rule. Select 'Rule' folder.

Enter '**PaymentExecutedRule**' in the Rule Function name field.

Click 'Finish' button.

Paste the contents of the PaymentExecutedRule.rule text file located in src/TCE/SalesOrderOrchestrator/Rule github repository.

Click 'Save' icon on the toolbar.

Create OrderPrepared Rule

Right click in the Studio Explorer view, select New->Rule...

A dialog box will appear for you to create a new Rule. Select 'Rule' folder.

Enter '**PaymentExecutedRule**' in the Rule Function name field.

Click 'Finish' button.

Paste the contents of the OrderPreparedRule.rule text file located in src/TCE/SalesOrderOrchestrator/Rule github repository.

Click 'Save' icon on the toolbar.

Create OrderDelivered Rule

Right click in the Studio Explorer view, select New->Rule...

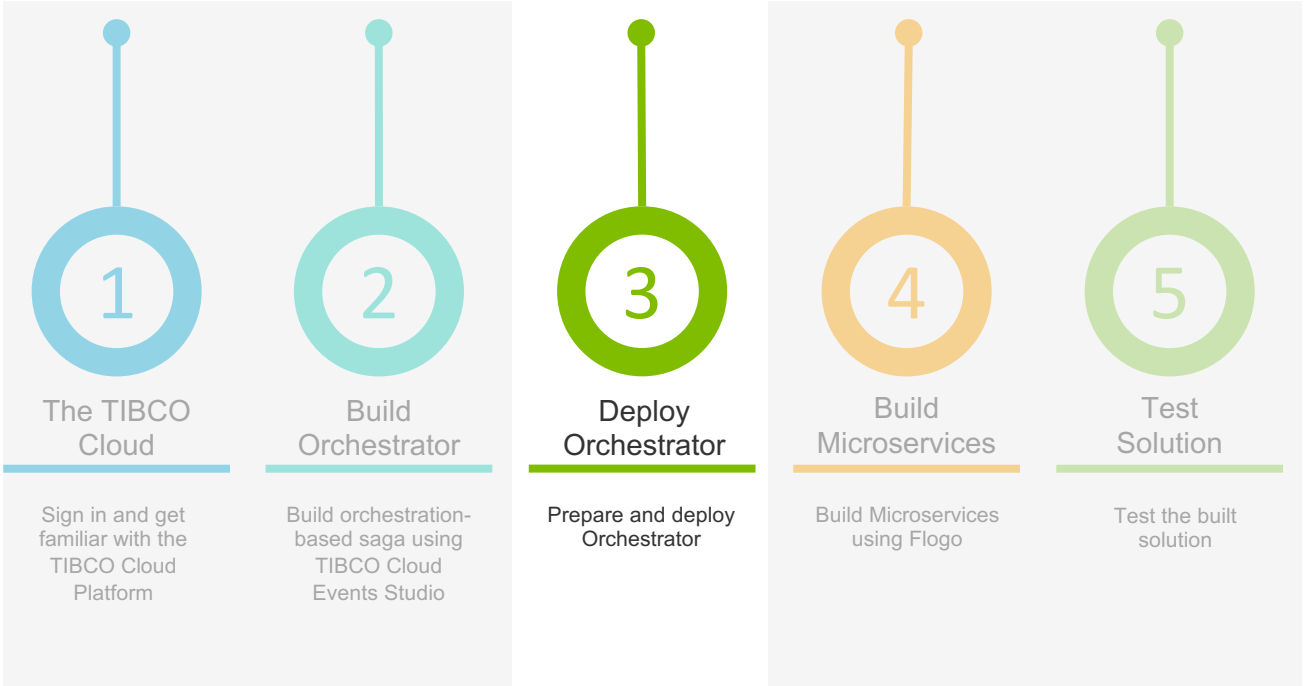
A dialog box will appear for you to create a new Rule. Select 'Rule' folder.

Enter '**OrderDeliveredRule**' in the Rule Function name field.

Click 'Finish' button.

Paste the contents of the OrderDeliveredRule.rule text file located in src/TCE/SalesOrderOrchestrator/Rule github repository.

Click 'Save' icon on the toolbar.


1
The TIBCO Cloud
Sign in and get familiar with the TIBCO Cloud Platform
2
Build Orchestrator
Build orchestration-based saga using TIBCO Cloud Events Studio
3
Deploy Orchestrator
Prepare and deploy Orchestrator
4
Build Microservices
Build Microservices using Flogo
5
Test Solution
Test the built solution

Import Cluster Deployment Descriptor

To run the TIBCO Cloud Events project we need to create a Cluster Deployment Descriptor file.

Import the file 'Default.cdd' located in src/TCE/SalesOrderOrchestrator folder of the github repository.

You can import directly in to the project by dragging and dropping the Default.cdd file using either your Mac Finder or Windows Explorer.

Eclipse will prompt you how it should import the files into the project, make sure to choose 'Copy Files'.

Import Swagger

To run the TIBCO Cloud Events project we need to create Swagger API file.

Import the file ‘swagger.json’ located in src/TCE/SalesOrderOrchestrator folder of the github repository.

You can import directly in to the project by dragging and dropping the swagger.json file using either your Mac Finder or Windows Explorer.

Eclipse will prompt you how it should import the files into the project, make sure to choose 'Copy Files'.

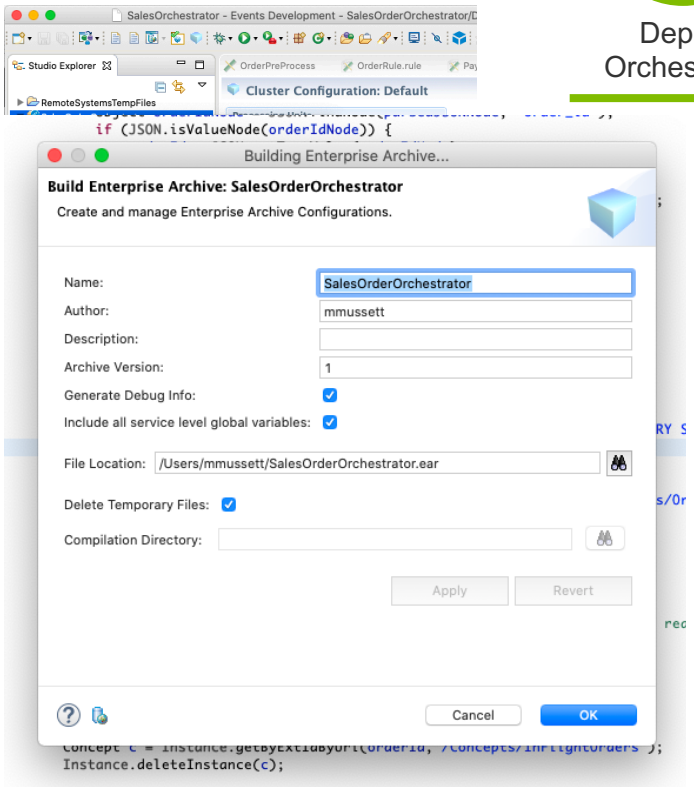
Build Enterprise Archive

Right click the SalesOrderOrchestrator project in the Studio Explorer view.

Click 'Build Enterprise Archive...' menu item.

Set the File Location to where you wish the Enterprise Archive file to be created.

Click ‘OK’



Deploy Enterprise Archive

3

Deploy Orchestrator

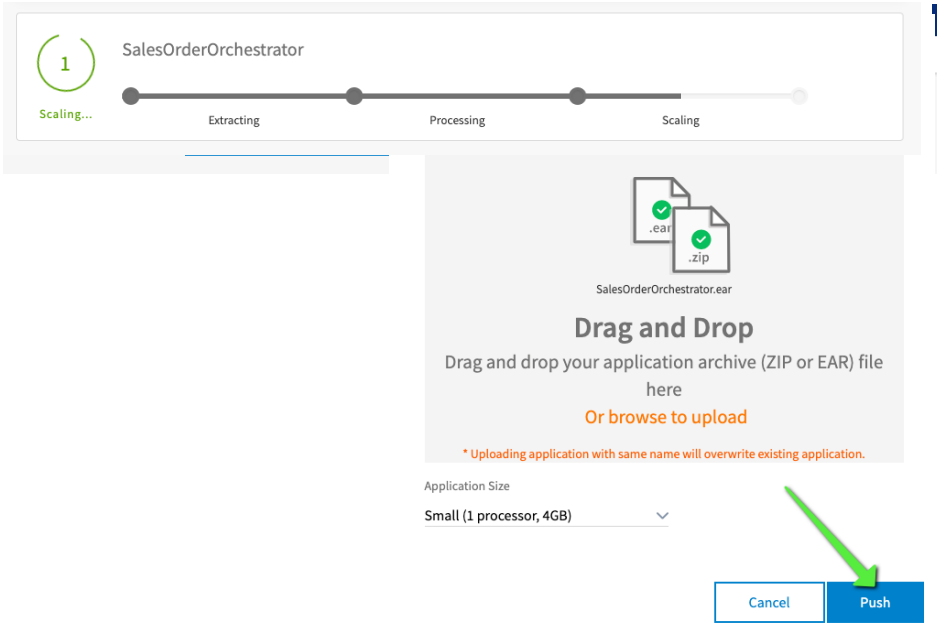
Open browser to https://eu.events.cloud.tibco.com

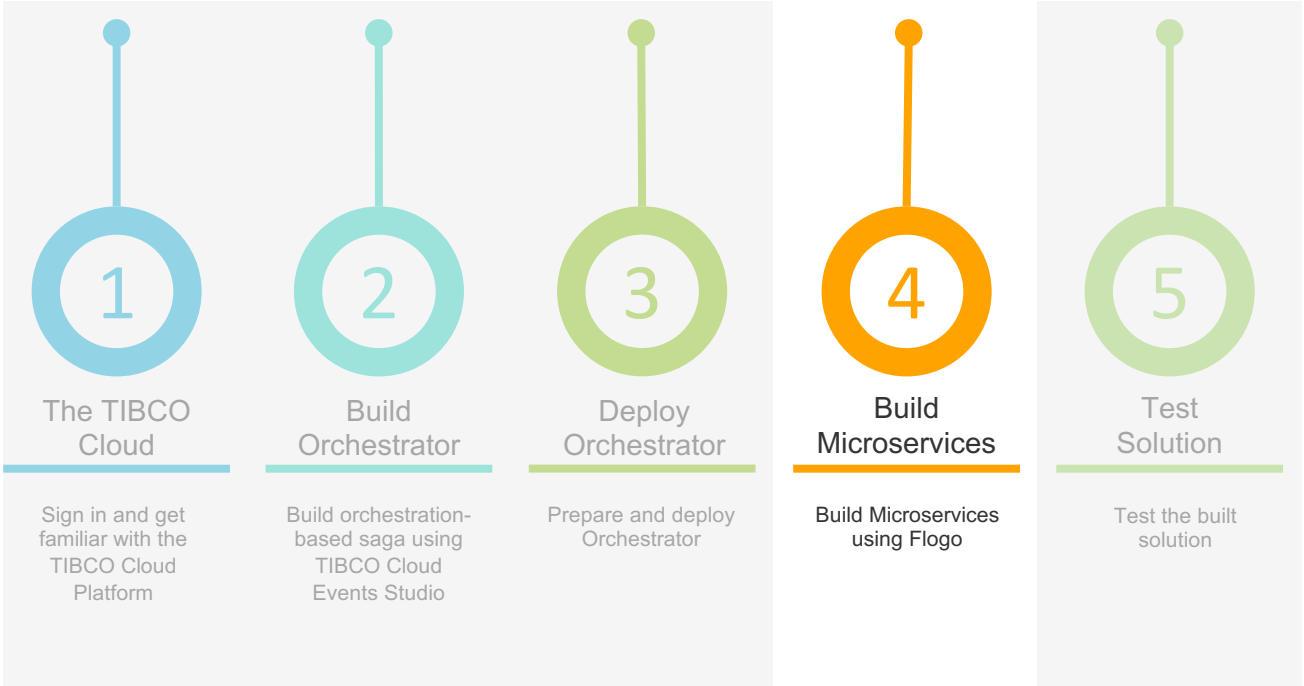
Click upload icon

Drag EAR file into dialog window

Click ‘Push’ button

The TCE Application will begin to deploy on to TIBCO Cloud




1
The TIBCO Cloud
Sign in and get familiar with the TIBCO Cloud Platform
2
Build Orchestrator
Build orchestration-based saga using TIBCO Cloud Events Studio
3
Deploy Orchestrator
Prepare and deploy Orchestrator
4
Build Microservices
Build Microservices using Flogo
5
Test Solution
Test the built solution

TIBCO Cloud Messaging

4

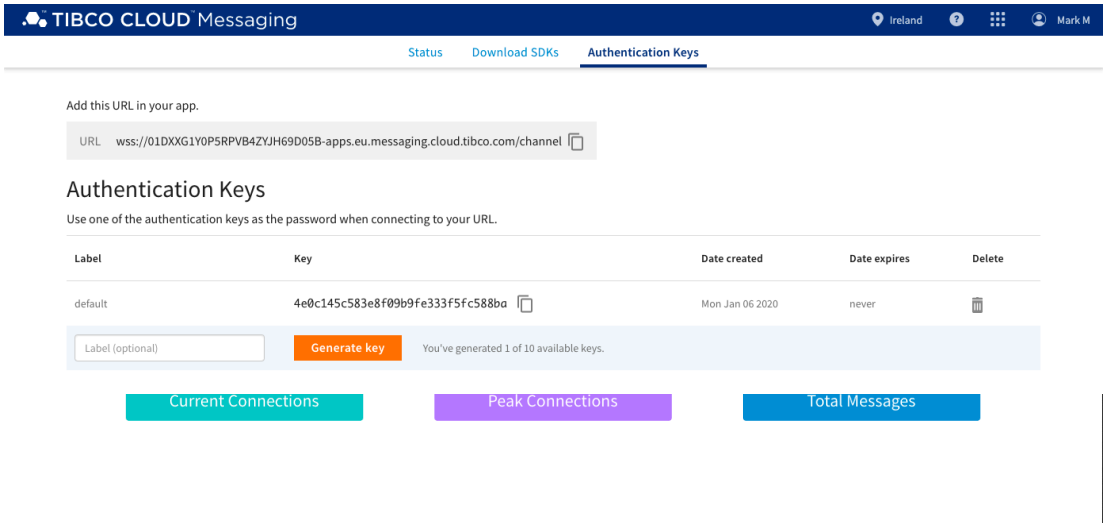
Build Microservices

Navigate to TIBCO Cloud Messaging Status Page here

Verify the service is running, if not start it.

Navigate to TIBCO Cloud Messaging Authentication Keys Page here

Make a note of the URL and pre-created authentication Key labelled 'default'



Create TIBCO Cloud Messaging Connector

4

Build Microservices

Navigate to TIBCO Cloud Integration Connections page here

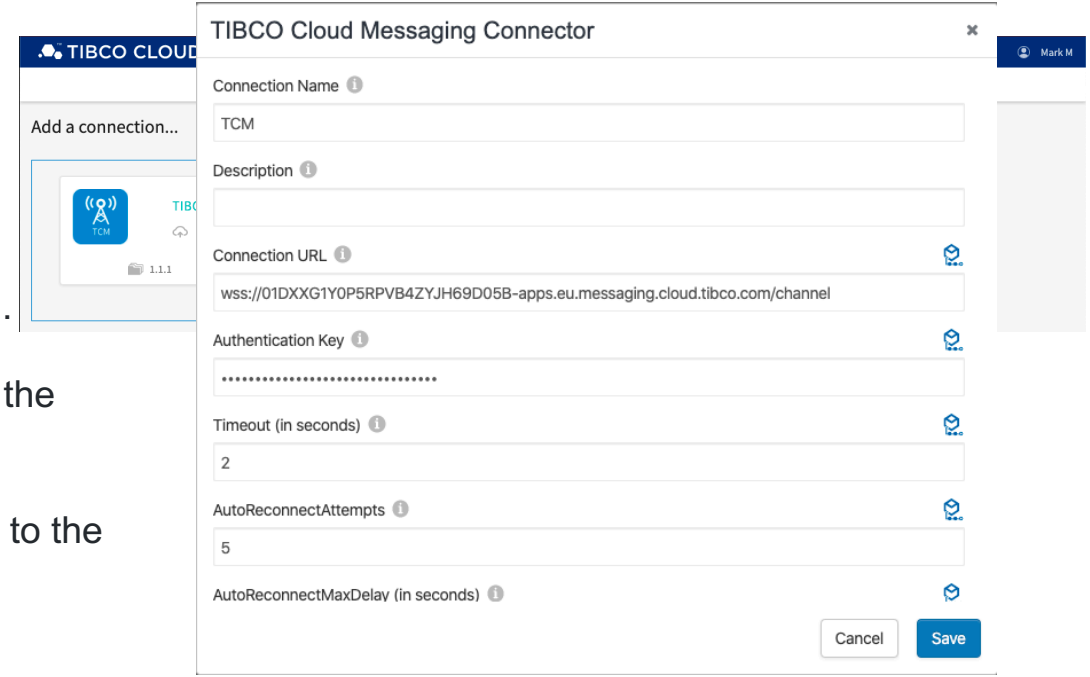
Enter 'TIBCO' in the search box and hit the magnifying glass symbol.

Click the 'TIBCO Cloud Messaging Connector'.

Paste the TIBCO Cloud Connection URL in to the Connection URL field.

Paste the TIBCO Cloud Authentication URL in to the Authentication Key field.

Click 'Save'



Create Payments Microservice

4

Build Microservices

Navigate to TCI Flogo Apps Page here

Click 'Create' button.

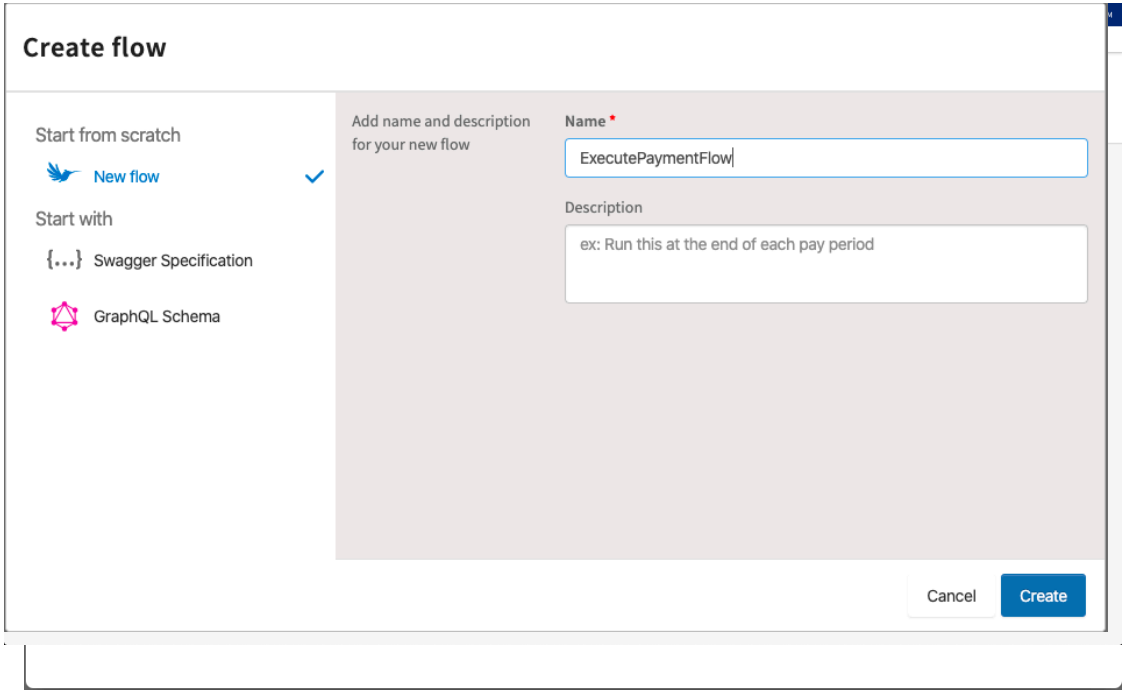
Enter 'PaymentService' for the app name.

Click 'Create' button.

Click 'Create a TIBCO Flogo App'.

Click '+ Create' to create Flogo Flow.

Enter 'ExecutePaymentFlow' for the Flow Name and click 'Create' button.

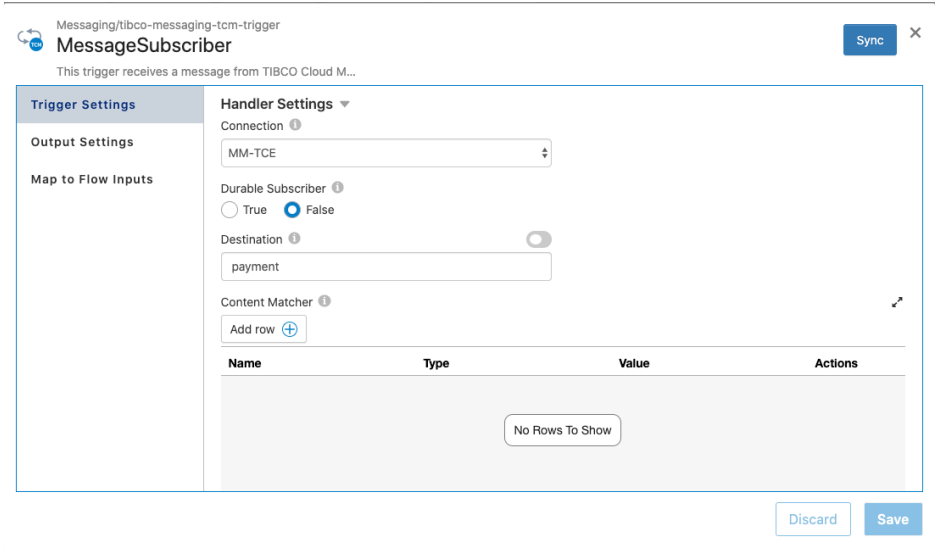


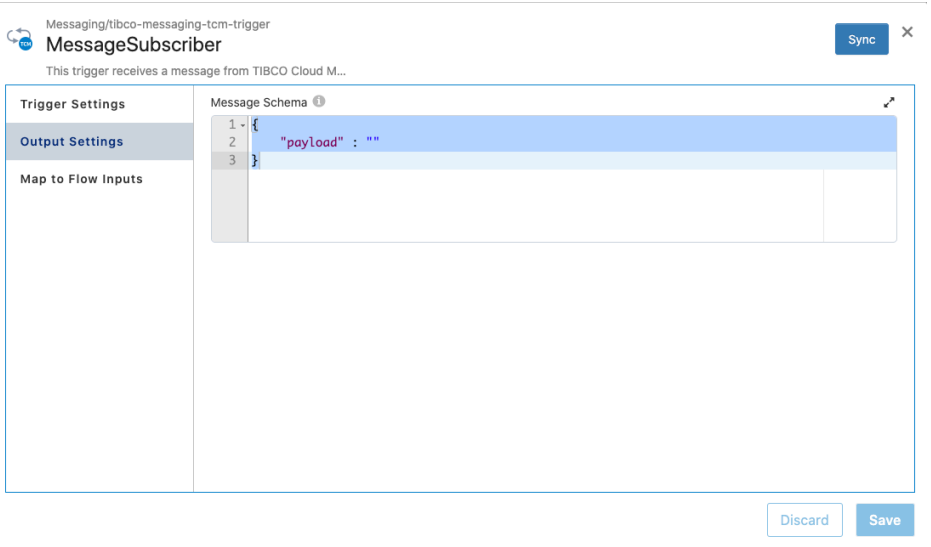
Create Payments Microservice

4

Build Microservices

We're going to create a trigger that looks like this:



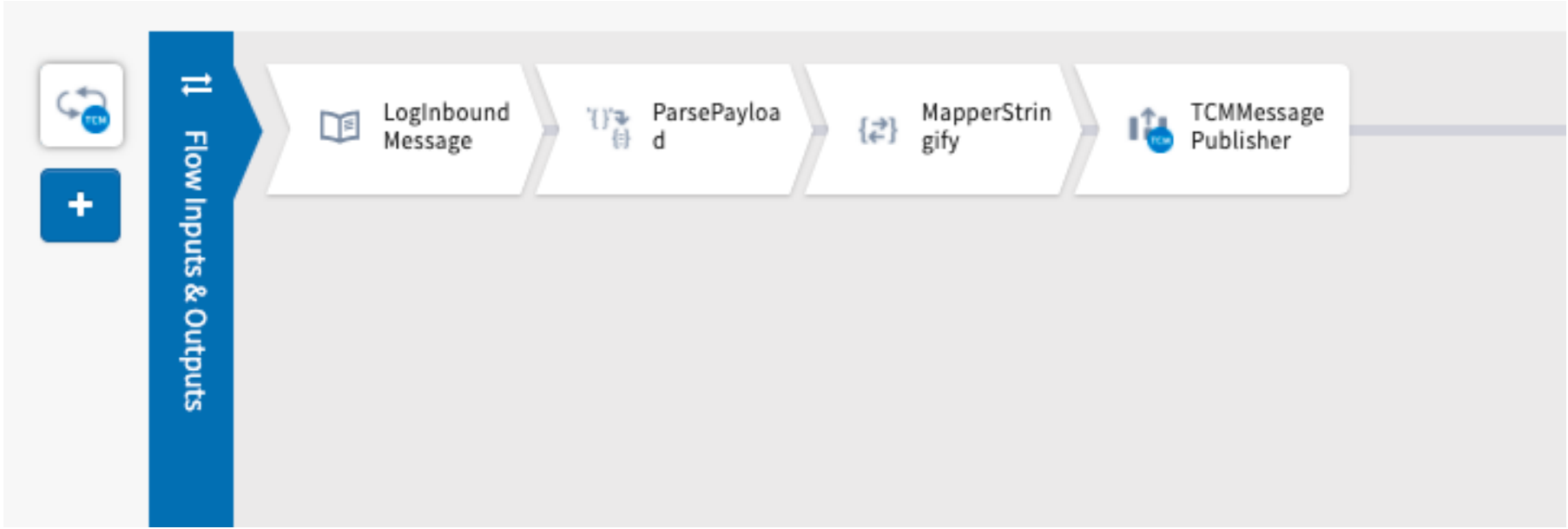


Create Payments Microservice

4

Build Microservices

We're going to create a flow that looks like this:



Create Payments Microservice - Trigger

4

Build Microservices

Click 'Start with a trigger'.

Click 'Message Subscriber' trigger.

Use the Connection dropdown box and choose the TCM Connector you created earlier and click 'Continue' button.

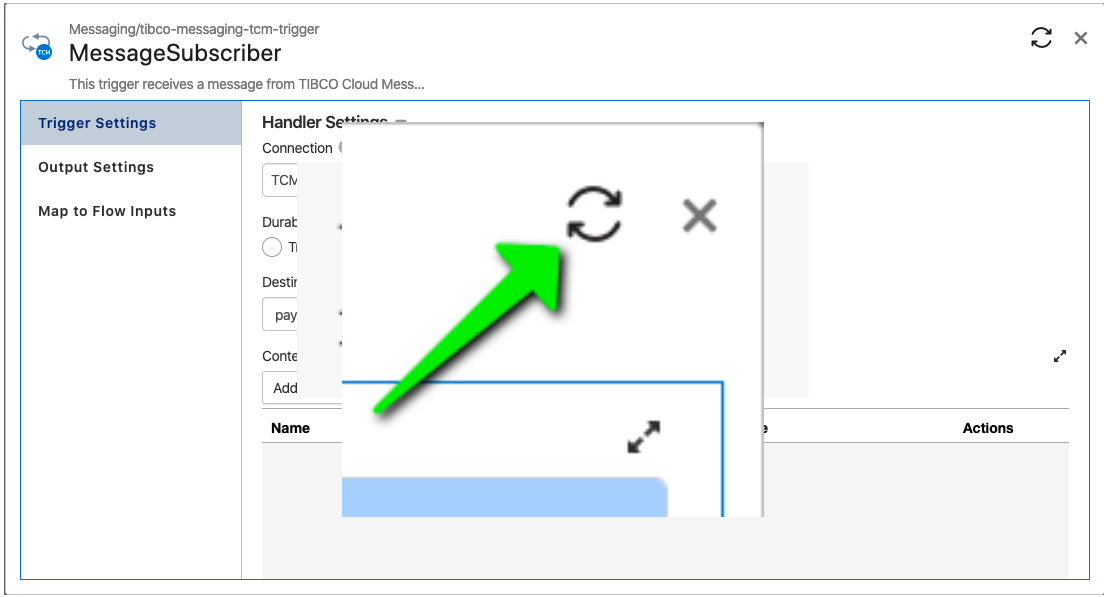
Click 'Just add the trigger' button

Click the TCM trigger symbol

Enter 'payment' into the Destination field.

Click 'Output Settings' in LHS panel and enter the following JSON in to the Message Schema

Click the sync flow input and output symbol in the top right corner of the dialog box



Create Payments Microservice - Trigger

Click 'Map to Flow Inputs' in LHS panel.

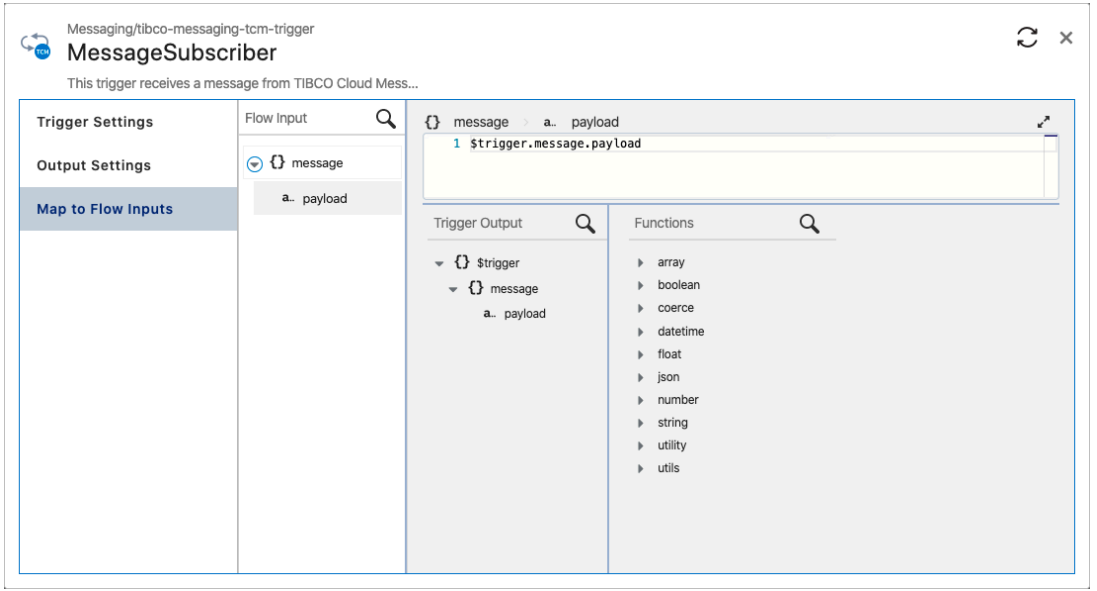
We need to map the Flow Input to the Trigger Output.

In the Flow Input pane navigate using the arrows to the payload field (message->payload).

In The Trigger Output panel navigate using the arrows to the payload field ($trigger->message->payload).

Click the 'payload' field.

Close the Message Subscriber dialog box.



Create Payments Microservice - Flow

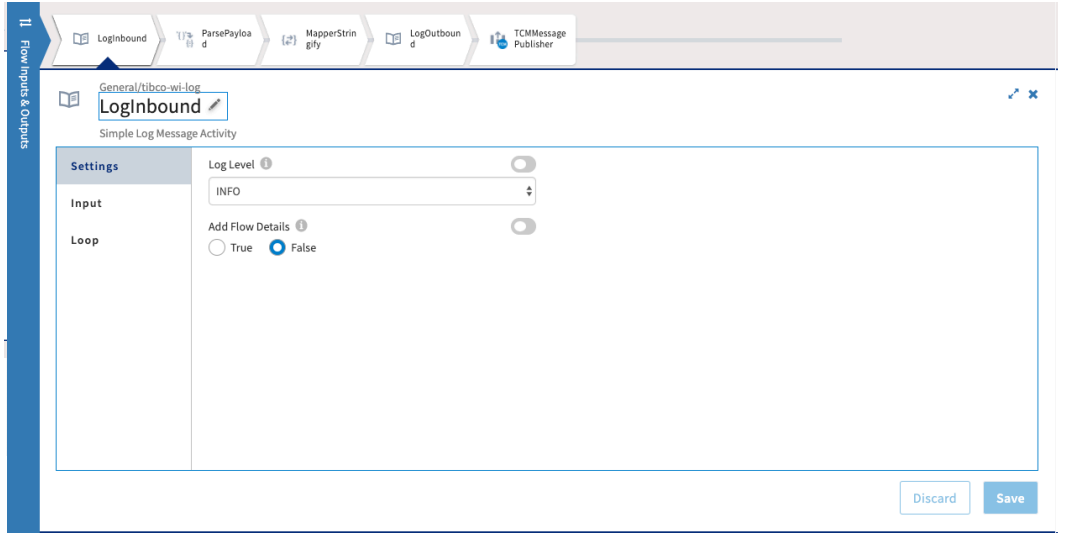
Hover over the Activity Line, and click the first free box to add activity. The Add Activity dialog will be show.

In the LHS panel, scroll to 'General' and click. The activities panel will show you the general activities available to place at this step.
Go ahead and click 'Log Message'

Click the 'LogMessage' activity title field and rename to 'LogInbound'

In the LHS panel of the LogMessage activity click 'Input' then click the 'a... * message' in the Activity input panel and enter the following in the message field:

Close the activity.



Create Payments Microservice - Flow

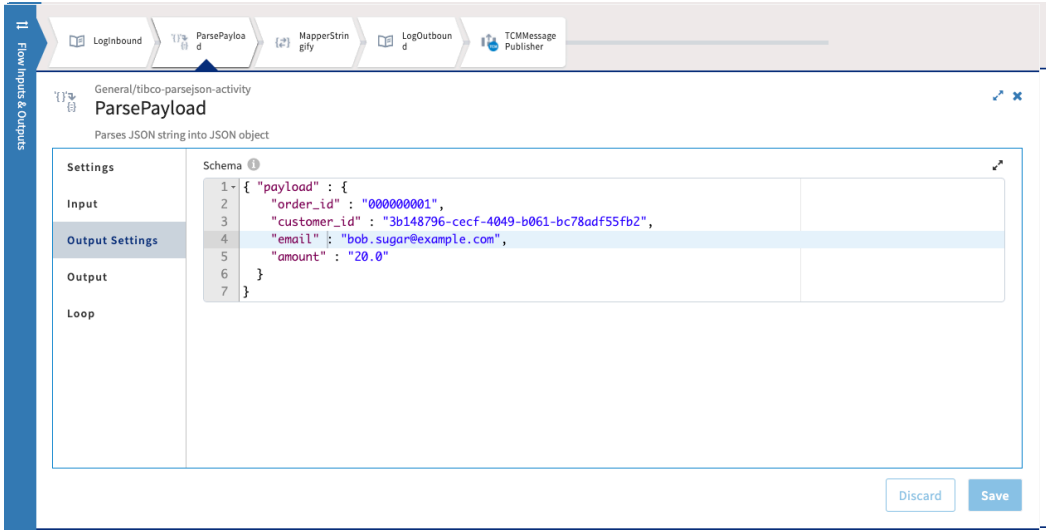
Add a Parse JSON Activity after the previous Log activity.

Click the 'ParseJSONActivity' activity title field and rename to 'ParsePayload'.

In the Input panel, map the jsonString of the Activity Input to '$flow.message.payload'.

Click 'Output Settings' in LHS panel and enter the following JSON in to the Schema:

Close the activity.



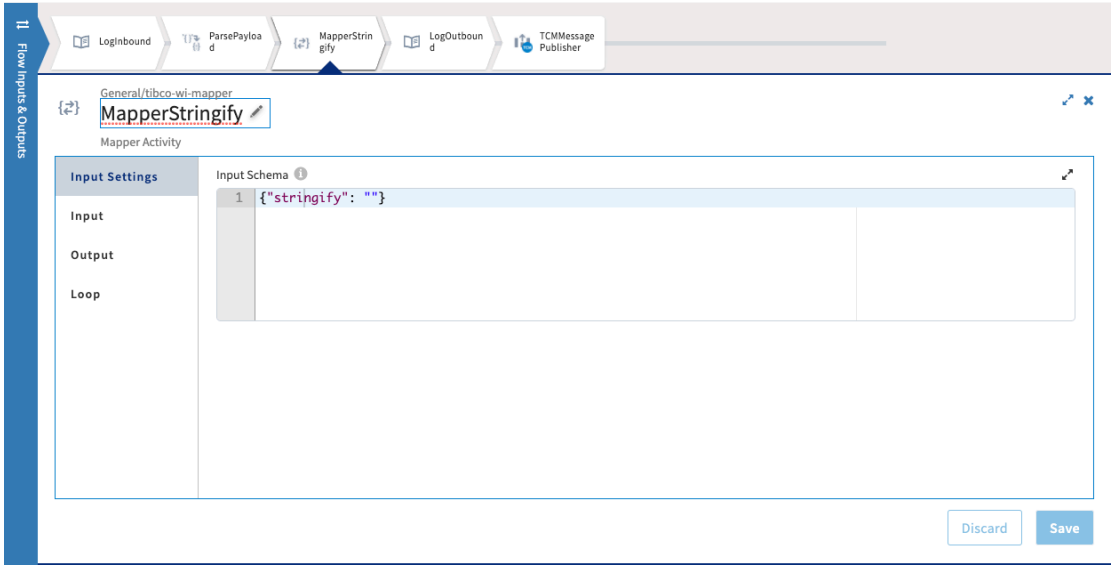
Create Payments Microservice - Flow

Add a Mapper Activity after the previous Parse JSON activity.

Click the 'Mapper' activity title field and rename to 'MapperStringify'.

In the 'Input Settings' enter the following JSON in to the Input Schema:

{“stringify”:””}

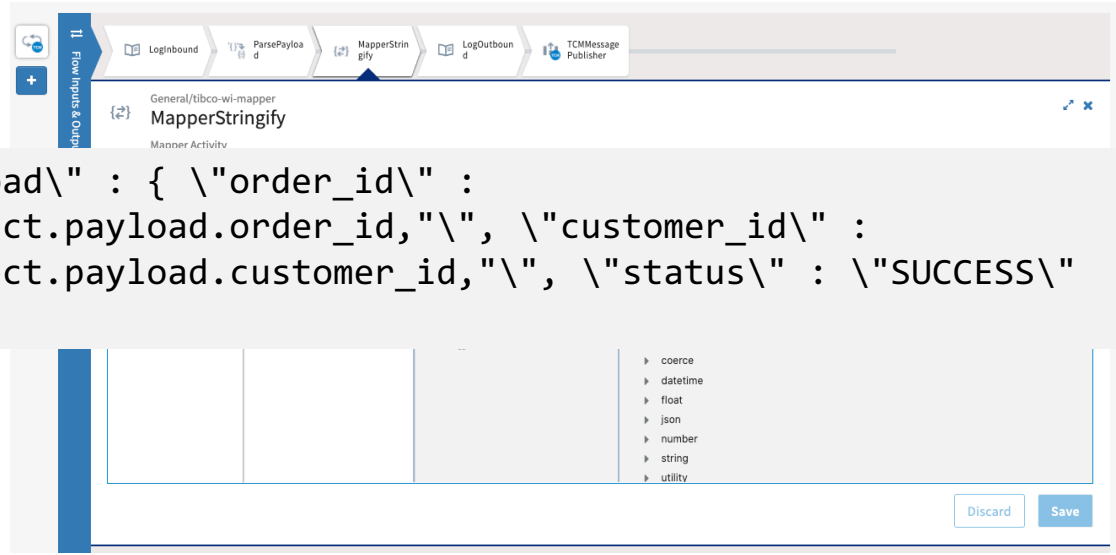


Create Payments Microservice - Flow

4

Build Microservices

In the Input panel, enter the follow expression in to the stringify field:




```
string.trim(string.concat("{ \"payload\" : { \"order_id\" :
\"",$activity[ParsePayload].jsonObject.payload.order_id,"\", \"customer_id\" :
\"",$activity[ParsePayload].jsonObject.payload.customer_id,"\", \"status\" : \"SUCCESS\"
}}"))
```

Close the activity.

Create Payments Microservice - Flow

4

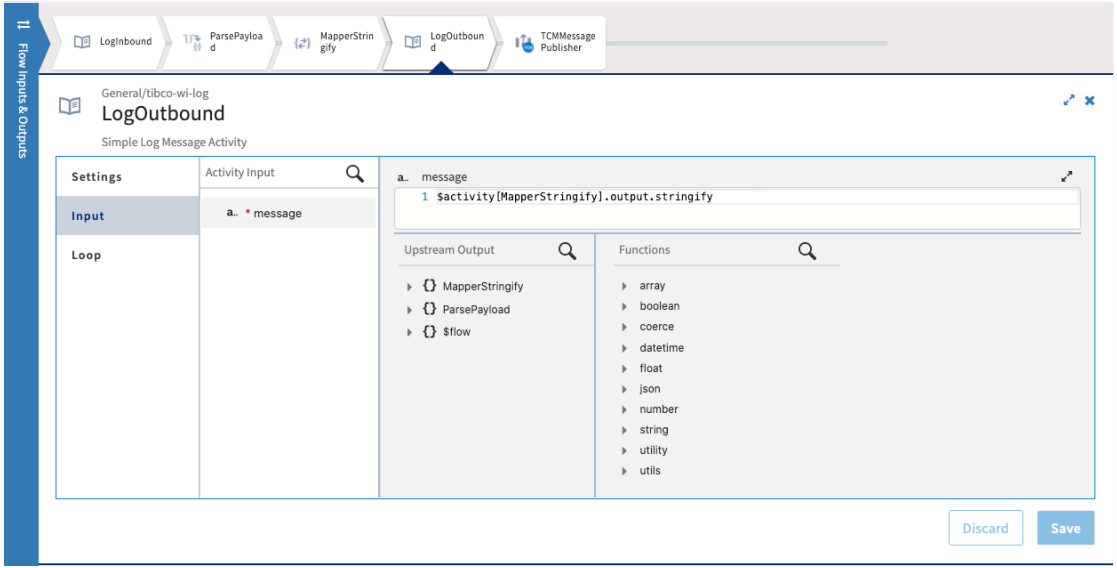
Build Microservices

Add a Logger Activity after the previous Mapper activity.

Click the ‘Logger' activity title field and rename to ‘LoggerOutbound’.

In the 'Input' enter the following

Close the Logger activity.



Create Payments Microservice - Flow

4

Build Microservices

Add a TCM Message Publisher Activity after the previous Mapper activity.

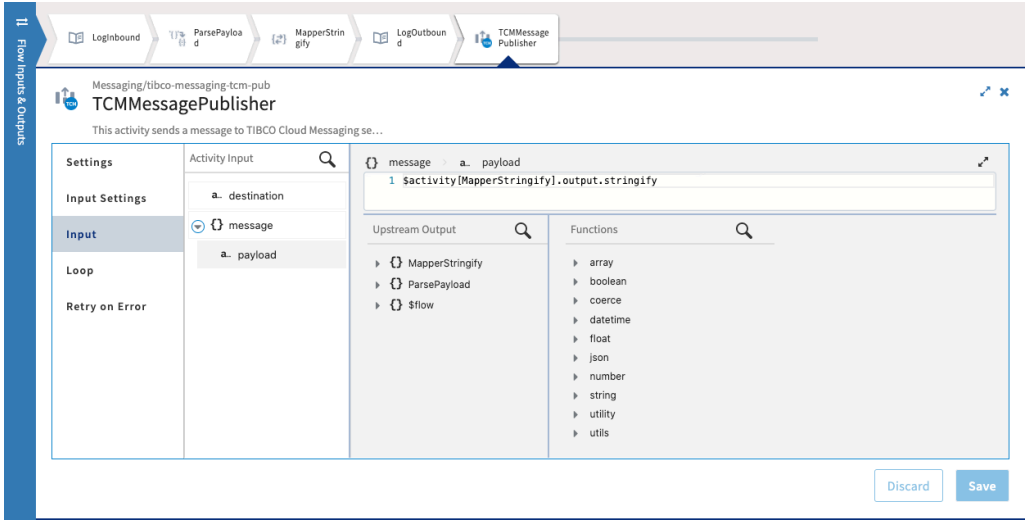
In Settings panel, select the TCM Connection.

In the 'Input Settings' enter the following JSON in to the Input Schema

In the ‘Input’ enter the following in to the ‘destination’ field.

In the ‘Input’ enter the following in to the ‘message payload’ field.

Close the activity.

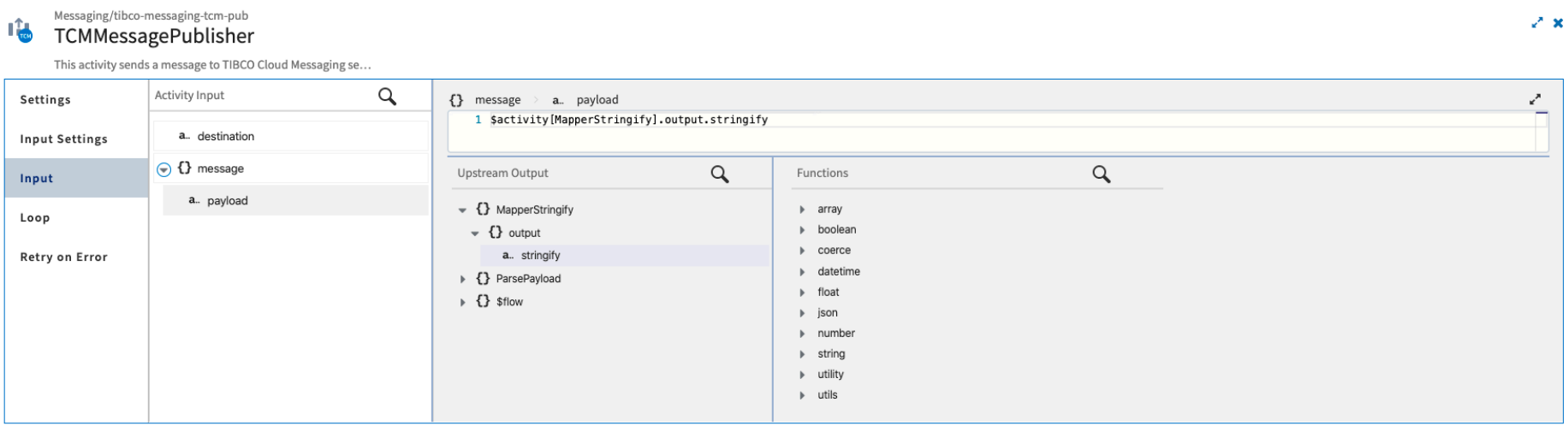


Create Payments Microservice

4

Build Microservices

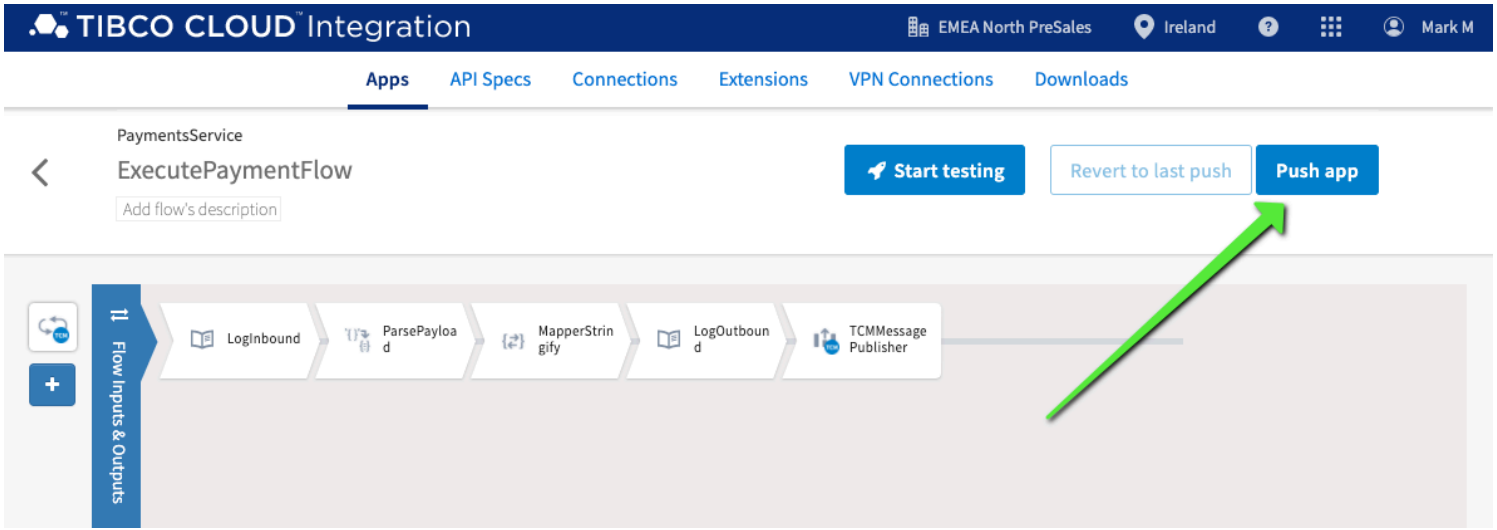
Map the 'stringify' field from the MapperStringify activity to the Activity Input of the message payload



Create Payments Microservice

Close the Message Publisher activity.

Click 'Push App'.

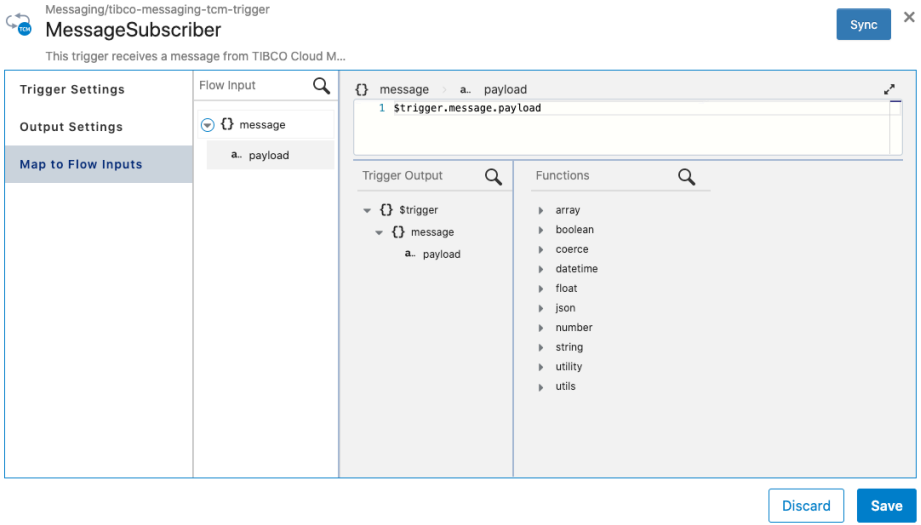


Create Stock Microservice

4

Build Microservices

We're going to create a trigger that looks like this:

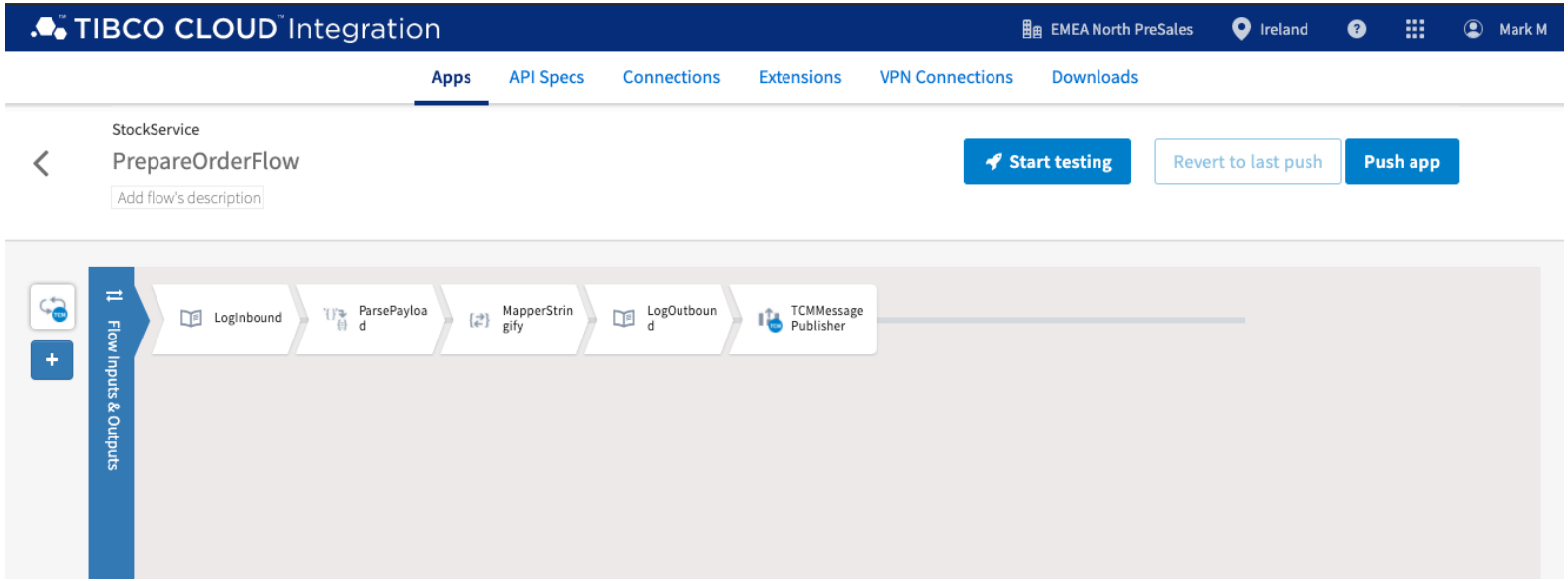


Create Stock Microservice

4

Build Microservices

We're going to create a flow that looks like this:



Create Stock Microservice - Trigger

Click 'Start with a trigger'.

Click 'Message Subscriber' trigger.

Use the Connection dropdown box and choose the TCM Connector you created earlier and click 'Continue' button.

Click 'Just add the trigger' button

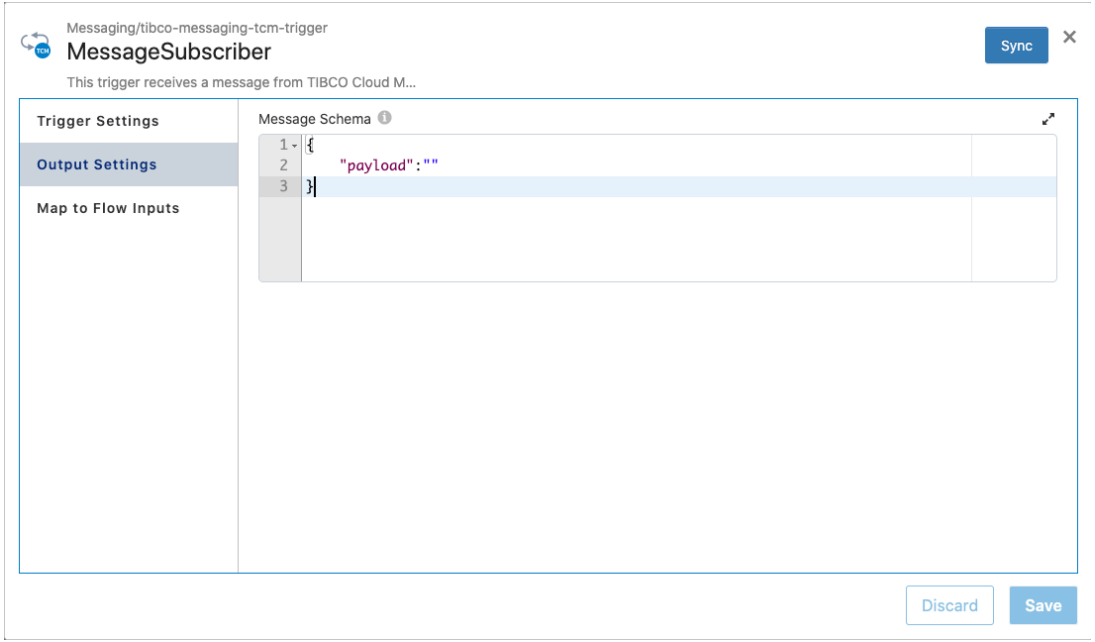
Click the TCM trigger symbol

Enter ‘prepare’ into the Destination field.

Click 'Output Settings' in LHS panel and enter the following JSON into the Message Schema

Click the sync flow input and output symbol in the top right corner of the dialog box

Close the Trigger

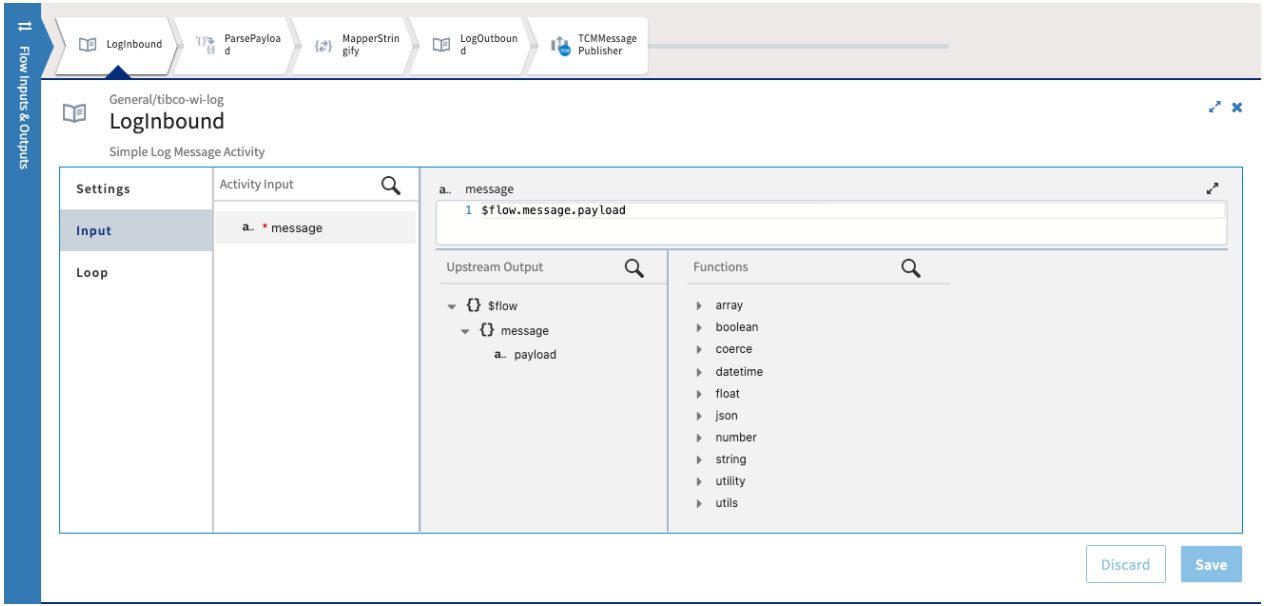


Create Stock Microservice - Flow

4

Build Microservices

Add Log Activity and configure

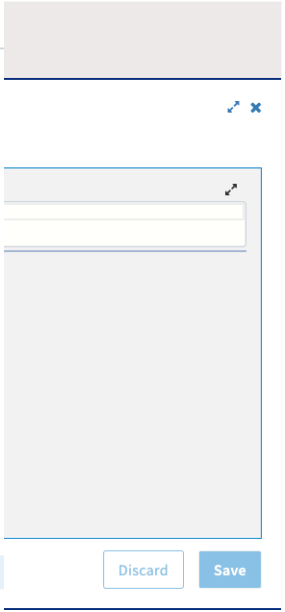


Create Stock Microservice - Flow

Add Parse Activity and configure

Schema

```
{
  "Order" : {
    "order_id" : "000000107",
    "customer_id" : "abc123",
    "email" : "bob.sugar@example.com",
    "items" : [ {
      "attributes" : {
        "Id" : "75",
        "type" : "www.tibco.com/be/ontology/Concepts/OrderItem"
      },
      "item_name" : "Gummy Bears",
      "price" : "2.0E0",
      "quantity" : "10"
    } ]
  }
}
```

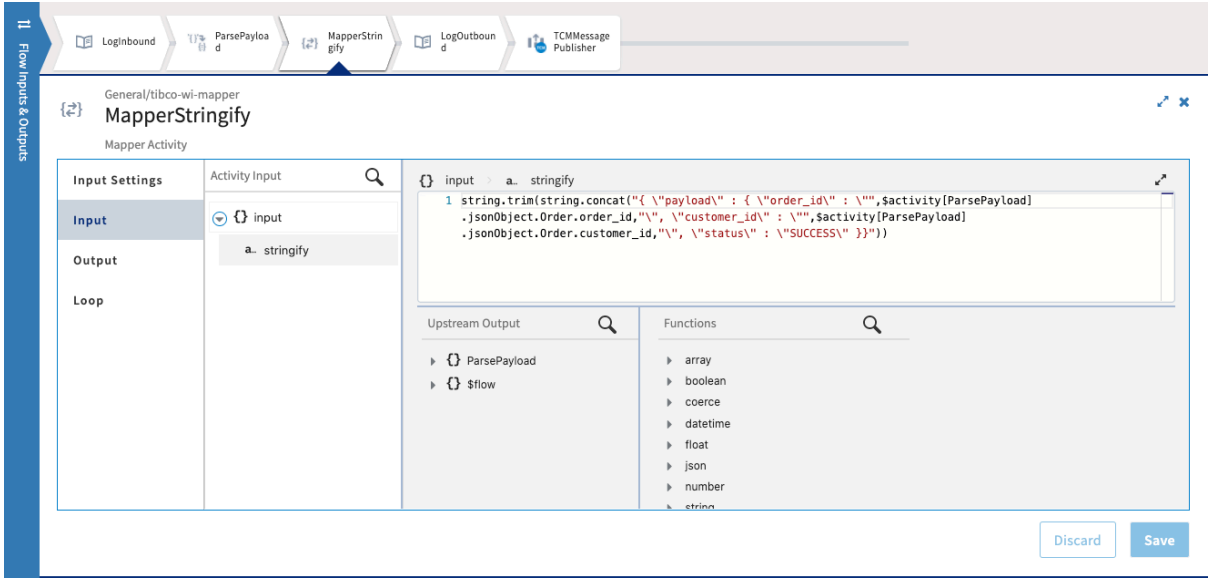


Create Stock Microservice - Flow

4

Build Microservices

Add Mapper Activity and configure

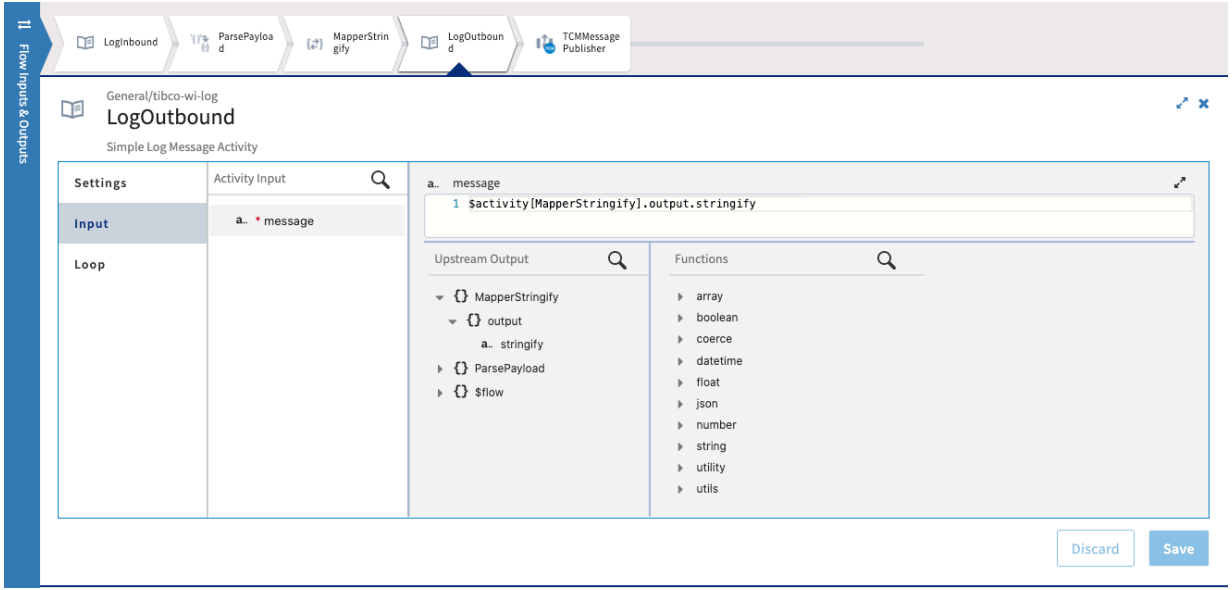


Create Stock Microservice - Flow

4

Build Microservices

Add Log Activity and configure



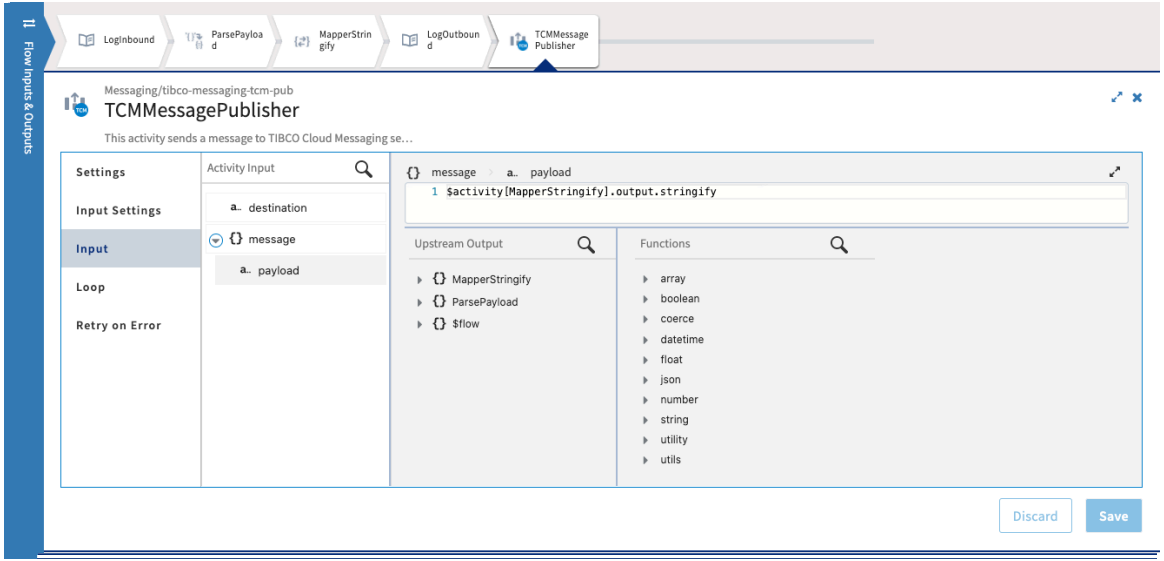
Create Stock Microservice - Flow

4

Build Microservices

Add a TCM Message Publisher Activity.

Save and Push Application.

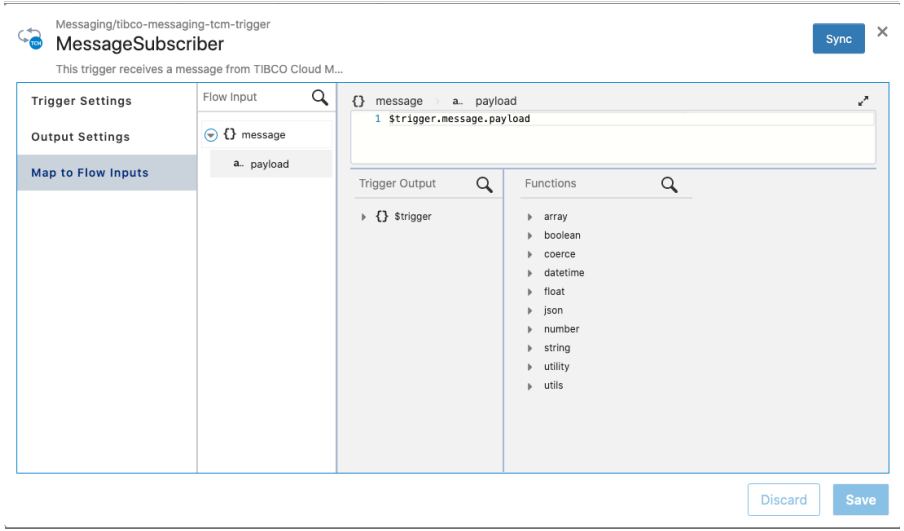


Create Delivery Microservice

4

Build Microservices

We're going to create a trigger that looks like this:

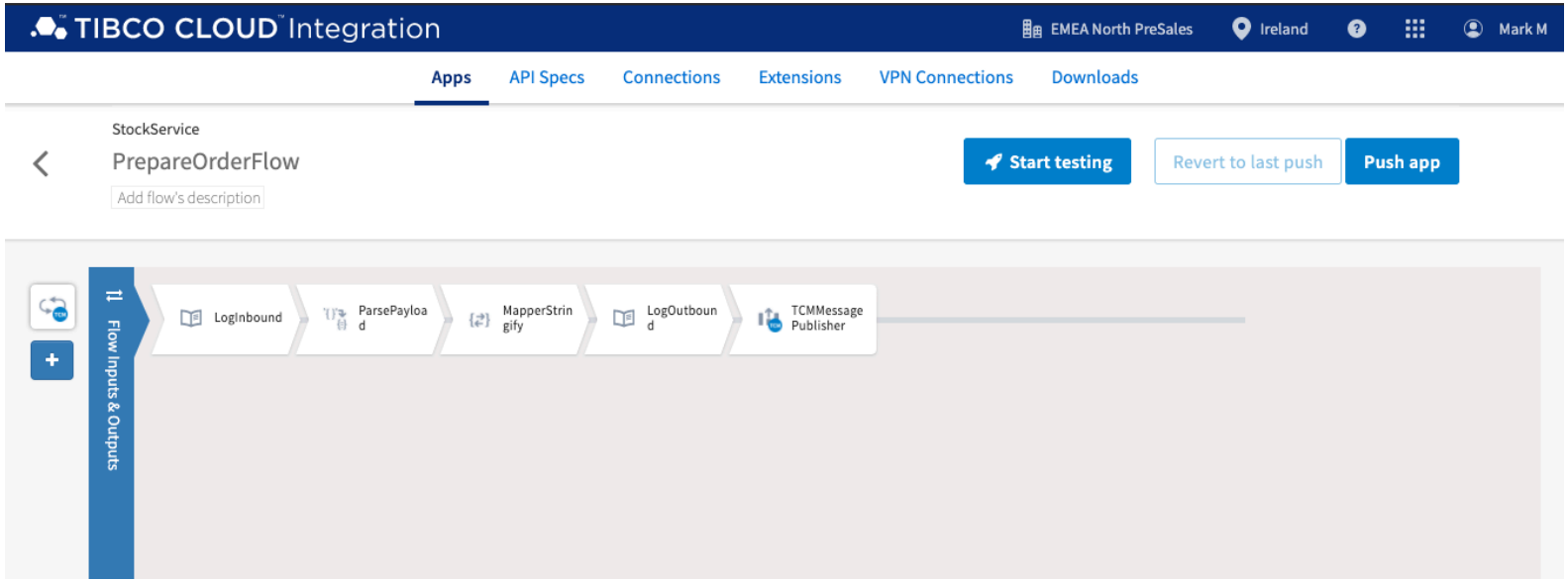


Create Delivery Microservice

4

Build Microservices

We're going to create a flow that looks like this:



Create Delivery Microservice - Trigger

4

Build Microservices

Click 'Start with a trigger'.

Click 'Message Subscriber' trigger.

Use the Connection dropdown box and choose the TCM Connector you created earlier and click 'Continue' button.

Click 'Just add the trigger' button

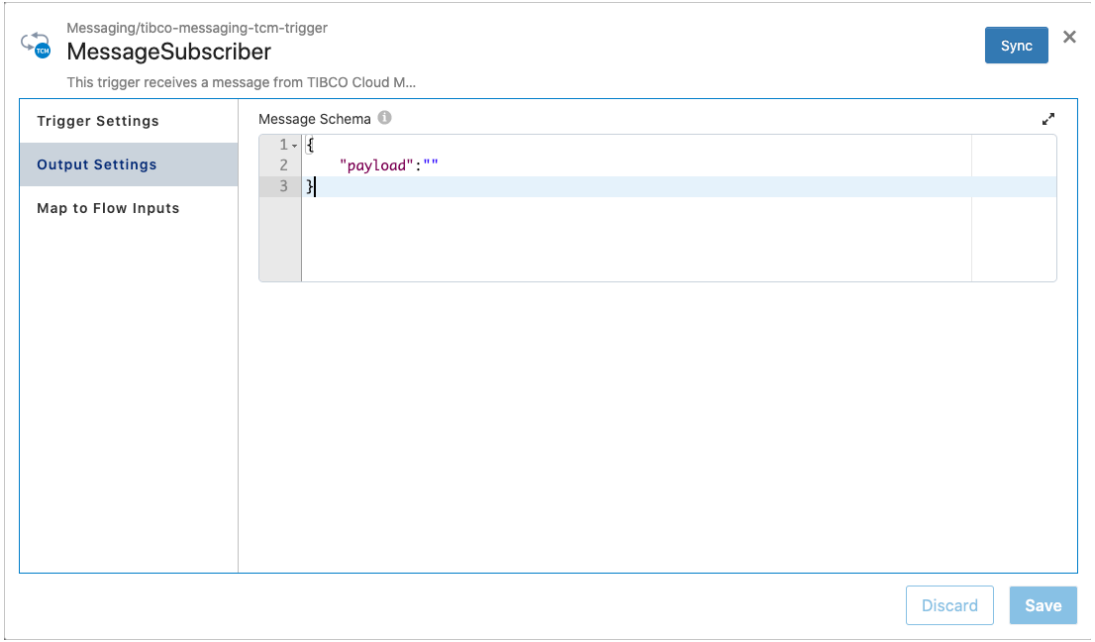
Click the TCM trigger symbol

Enter ‘prepare’ into the Destination field.

Click 'Output Settings' in LHS panel and enter the following JSON into the Message Schema

Click the sync flow input and output symbol in the top right corner of the dialog box

Close the Trigger

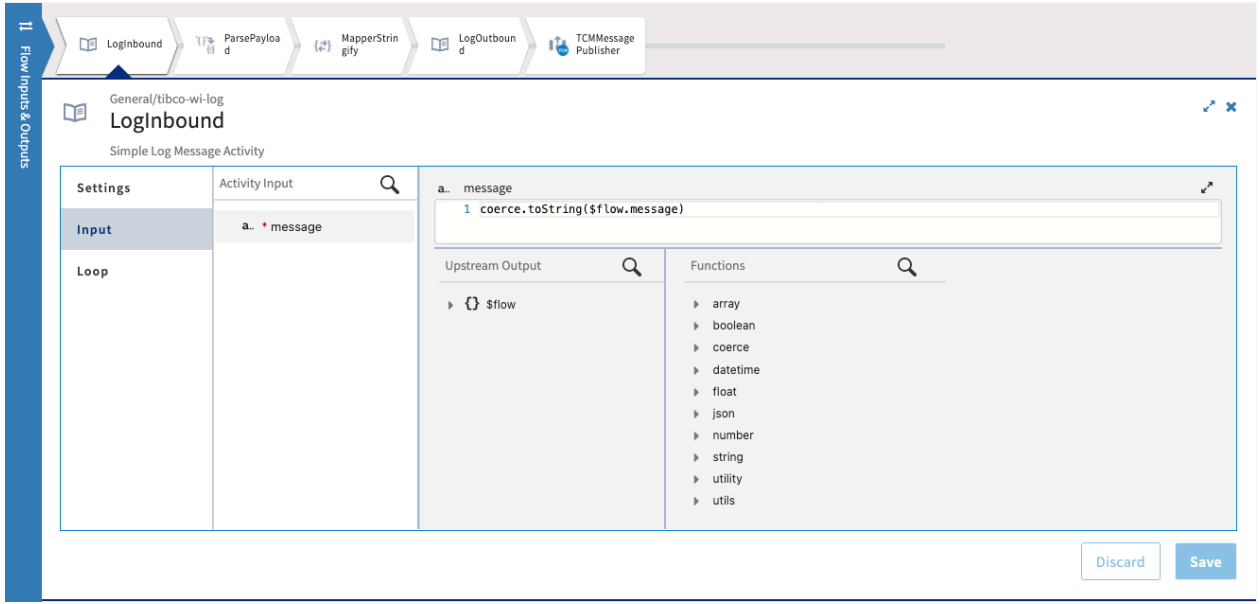


Create Delivery Microservice - Flow

4

Build Microservices

Add Log Activity and configure



Create Delivery Microservice - Flow

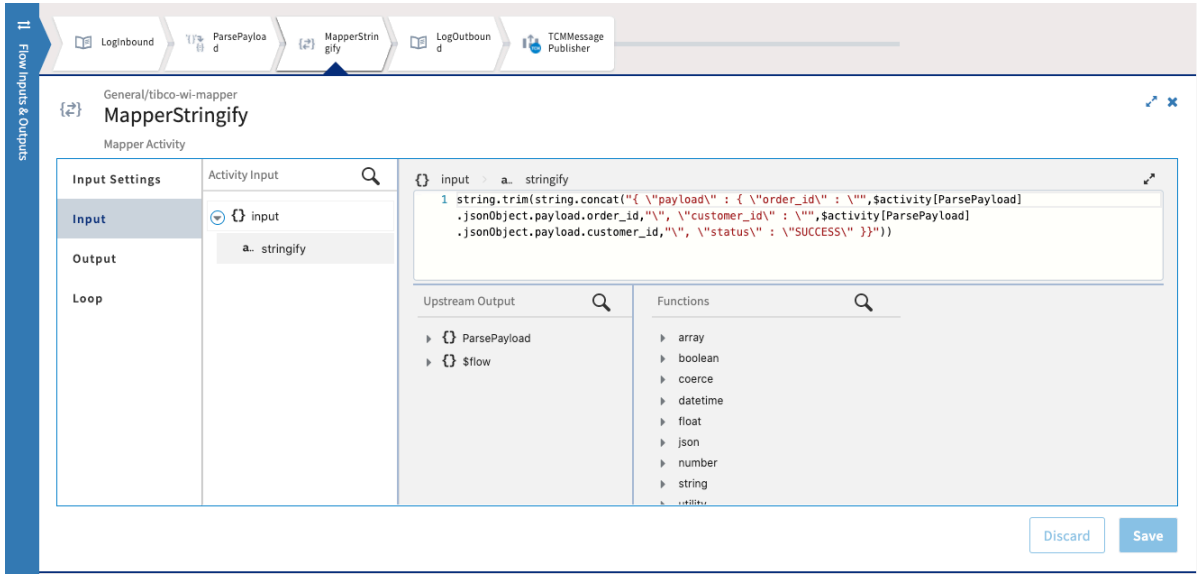
4

Build Microservices

Add Parse Activity and configure

Create Delivery Microservice - Flow

Add Mapper Activity and configure

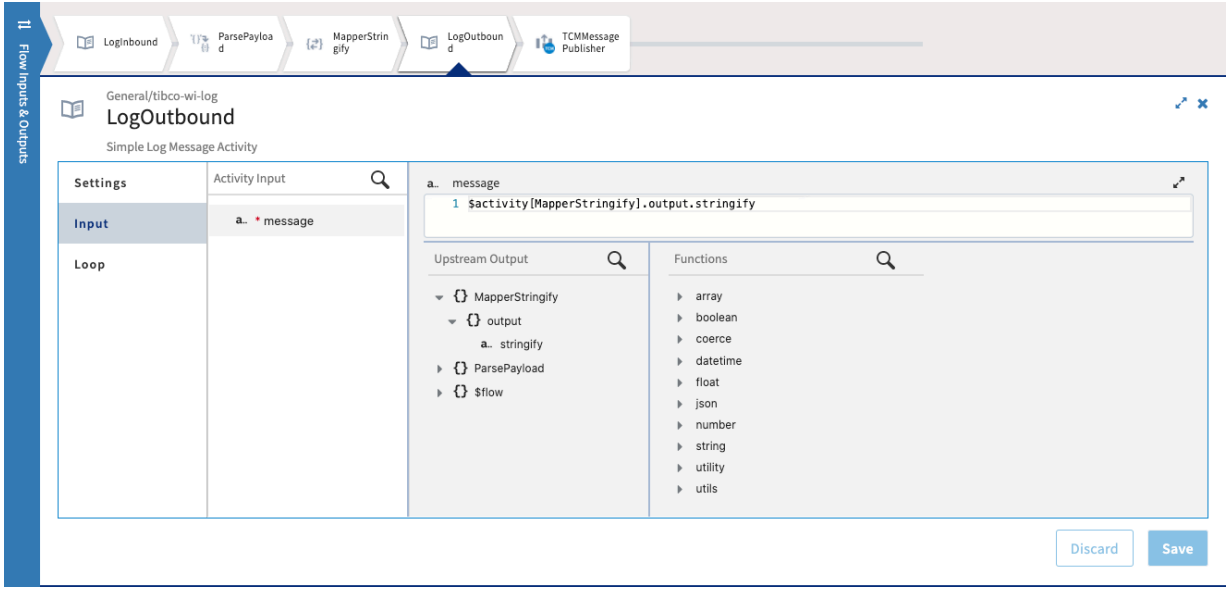


Create Delivery Microservice - Flow

4

Build Microservices

Add Log Activity and configure



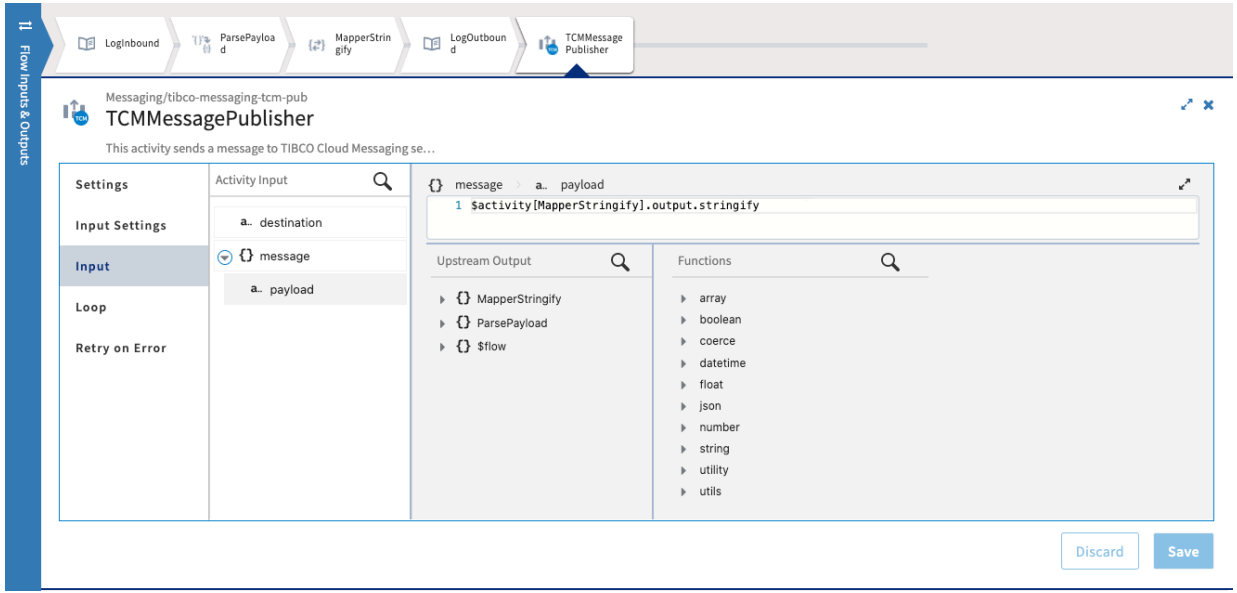
Create Delivery Microservice - Flow

4

Build Microservices

Add a TCM Message Publisher Activity.

Save and Push Application.

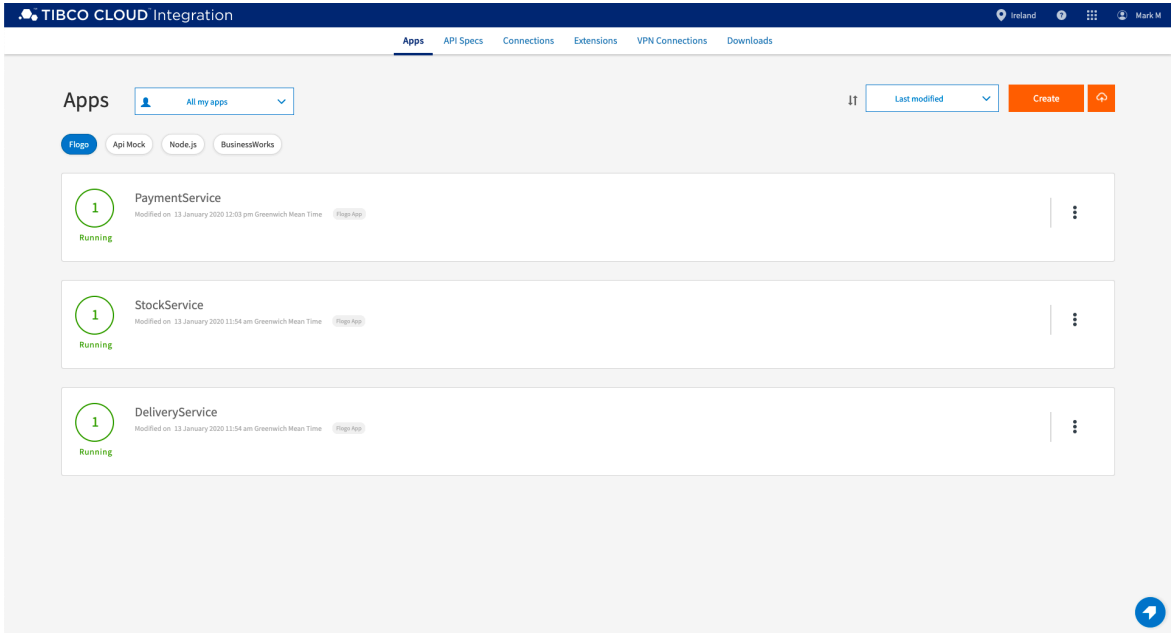


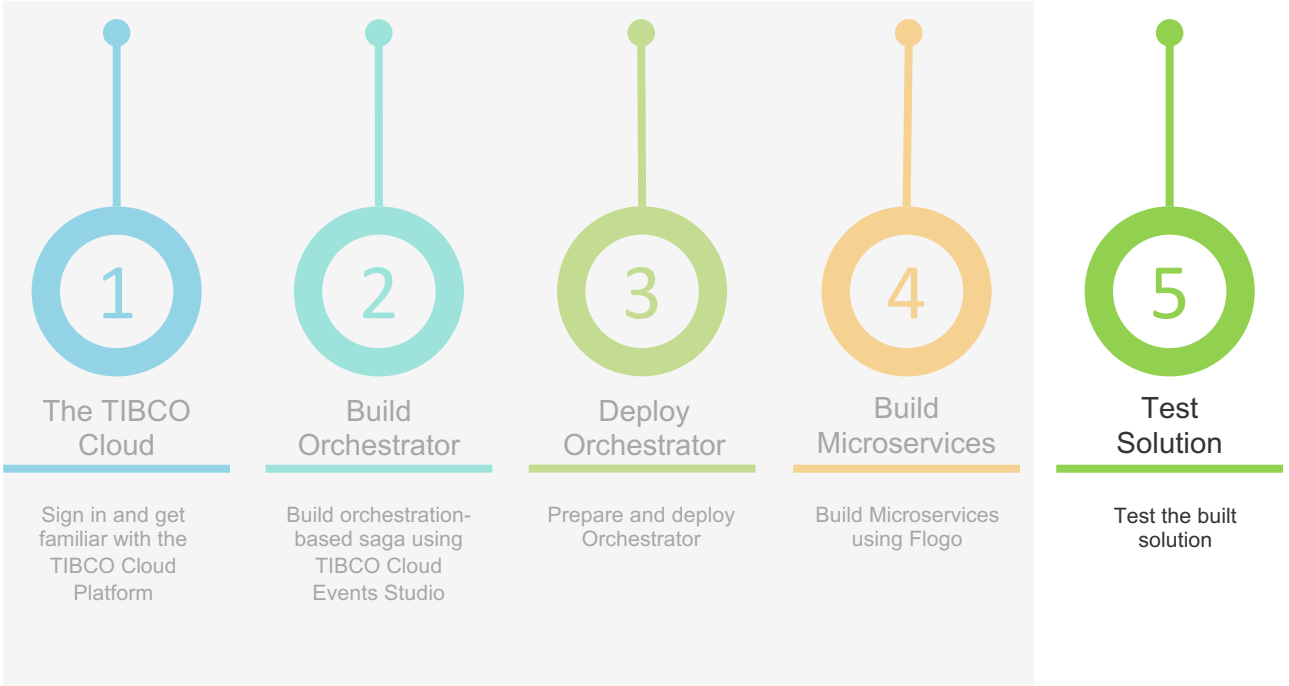
Start Flogo Apps

4

Build Microservices

Verify the 3 Flogo Applications are started




1
The TIBCO Cloud
Sign in and get familiar with the TIBCO Cloud Platform
2
Build Orchestrator
Build orchestration-based saga using TIBCO Cloud Events Studio
3
Deploy Orchestrator
Prepare and deploy Orchestrator
4
Build Microservices
Build Microservices using Flogo
5
Test Solution
Test the built solution

Use Swagger to Test Solution

5

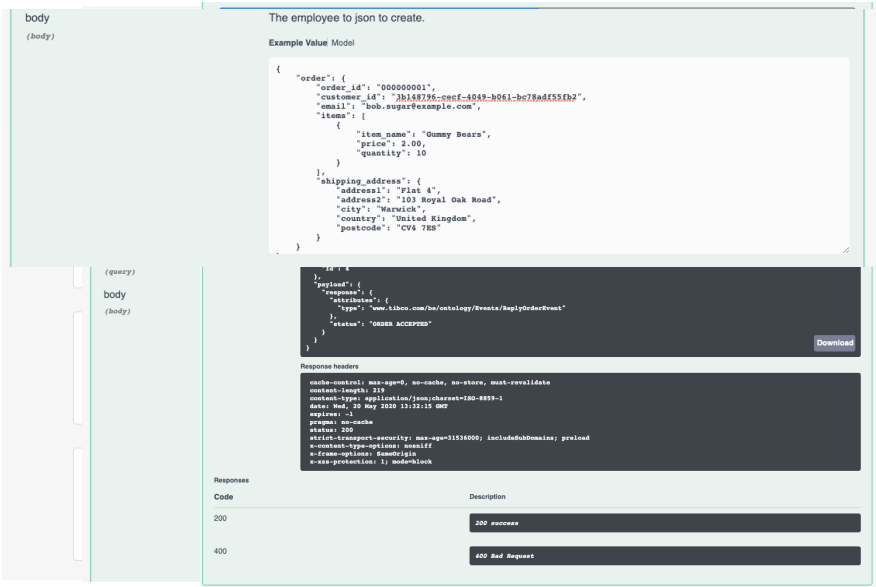
Test Solution

Click ’View and Test’ endpoints and select the swagger link

Click ‘Try it out’

Paste the content of ‘SalesOrderOrchestrator.json’in to the body (the file is in the root directory of the git repo)

Click ‘Execute’

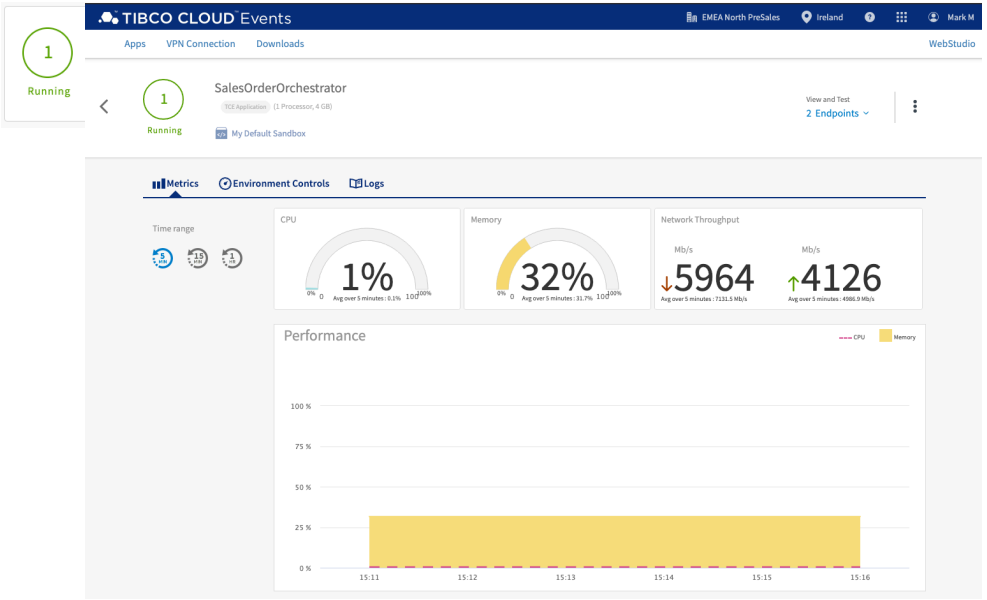


View Application Metrics

5

Test Solution

Click SalesOrderOrchestrator application to show Metrics page

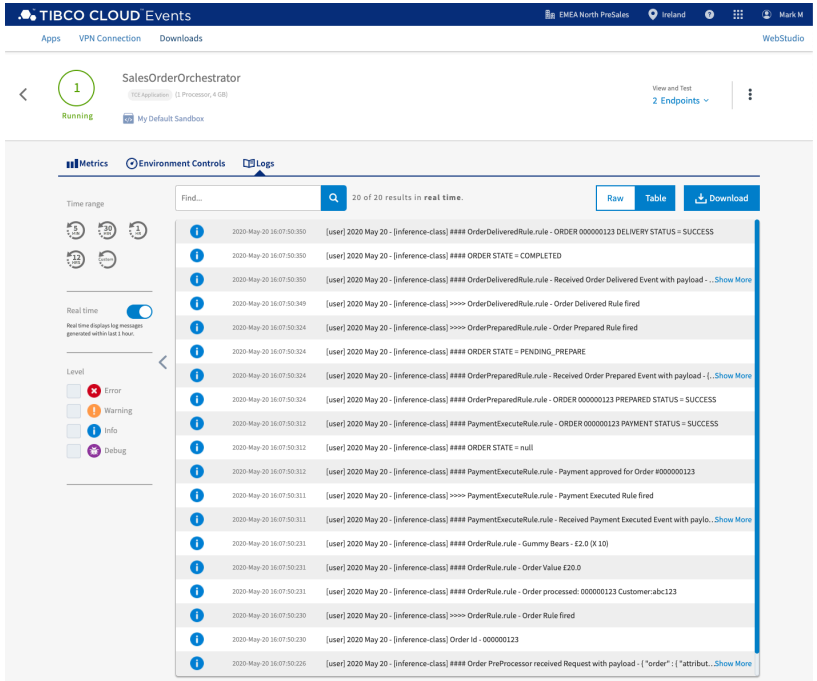


View Application Logs

5

Test Solution

Click ‘Logs’ Link to view the application logs.



View Application Logs

5

Test Solution

TIBCO Cloud Events Receives Order request over HTTP

Send Order

Customer Order

Order PreProcessor ->

Order Rule ->

TIBCO Cloud Events sends Payment Event to Payment Service

Payment Microservice

Take Payment

All Hands International Presales Call - Google Slides

	2020-May-20 16:07:50:226	[user] 2020 May 20 - [inference-class] #### Order PreProcessor received Request with payload - { "order" : { "attributes" : { "type" : "www.tibco.com/be/ontology/Concepts/Order" }, "order_id" : "000000123", "customer_id" : "abc123", "email" : "bob.sugar@example.com", "items" : [{ "item_name" : "Gummy Bears", "price" : "2.0", "quantity" : "10" }], "address" : { "address1" : "Flat 4", "address2" : "103 Royal Oak Road", "city" : "Warwick", "country" : "United Kingdom", "postcode" : "CV4 7ES" } } } Show Less
	2020-May-20 16:07:50:225	[user] 2020 May 20 - [inference-class] >>>> Order PreProcessor fired
	2020-May-20 16:07:50:231	[user] 2020 May 20 - [inference-class] #### OrderRule.rule - Gummy Bears - £2.0 (X 10)
	2020-May-20 16:07:50:231	[user] 2020 May 20 - [inference-class] #### OrderRule.rule - Order Value £20.0
	2020-May-20 16:07:50:231	[user] 2020 May 20 - [inference-class] #### OrderRule.rule - Order processed: 000000123 Customer:abc123
	2020-May-20 16:07:50:230	[user] 2020 May 20 - [inference-class] >>>> OrderRule.rule - Order Rule fired
	2020-May-20 16:07:50:230	[user] 2020 May 20 - [inference-class] Order Id - 000000123

TIBCO

View Application Logs

5

Test Solution

TIBCO Cloud Events Receives Payment Received Event from Payments Service

Payment Microservice

Payment Taken

Payment Executed Rule ->

2020-May-20 16:07:50:312	[user] 2020 May 20 - [inference-class] #### PaymentExecuteRule.rule - Payment approved for Order #000000123
2020-May-20 16:07:50:311	[user] 2020 May 20 - [inference-class] >>>> PaymentExecuteRule.rule - Payment Executed Rule fired
2020-May-20 16:07:50:311	[user] 2020 May 20 - [inference-class] #### PaymentExecuteRule.rule - Received Payment Executed Event with payload - { "payload" : { "attributes" : { "type" : "www.tibco.com/be/ontology/Events/PaymentExecutedEvent" }, "order_id" : "000000123", "customer_id" : "abc123", "status" : "SUCCESS" } } Show Less

TIBCO Cloud Events sends Prepare Order Event to Stock Service

Stock Microservice

Fulfill Order

TIBCO

View Application Logs

5

Test Solution

TIBCO Cloud Events Receives Order Prepared Event from Stock Service

Stock Microservice

Order Fulfilled

Order Prepared Executed Rule ->

2020-May-20 16:07:50:324	[user] 2020 May 20 - [inference-class] >>>> OrderPreparedRule.rule - Order Prepared Rule fired
2020-May-20 16:07:50:324	[user] 2020 May 20 - [inference-class] #### ORDER STATE = PENDING_PREPARE
2020-May-20 16:07:50:324	[user] 2020 May 20 - [inference-class] #### OrderPreparedRule.rule - Received Order Prepared Event with payload - { "payload" : { "attributes" : { "type" : "www.tibco.com/be/ontology/Events/OrderPreparedEvent" }, "order_id" : "000000123", "customer_id" : "abc123", "status" : "SUCCESS" } } Show Less
2020-May-20 16:07:50:324	[user] 2020 May 20 - [inference-class] #### OrderPreparedRule.rule - ORDER 000000123 PREPARED STATUS = SUCCESS

TIBCO Cloud Events sends Prepare Delivery Event to Delivery Service

Delivery Microservice

Dispatch Order

TIBCO

View Application Logs

5

Test Solution

TIBCO Cloud Events Receives Order Delivered Event from Delivery Service

Delivery Microservice

Order Delivered

Order Delivered Executed Rule ->

2020-May-20 16:07:50:350	[user] 2020 May 20 - [inference-class] #### OrderDeliveredRule.rule - ORDER 000000123 DELIVERY STATUS = SUCCESS
2020-May-20 16:07:50:350	[user] 2020 May 20 - [inference-class] #### ORDER STATE = COMPLETED
2020-May-20 16:07:50:350	[user] 2020 May 20 - [inference-class] #### OrderDeliveredRule.rule - Received Order Delivered Event with payload - .. Show More
2020-May-20 16:07:50:349	[user] 2020 May 20 - [inference-class] >>>> OrderDeliveredRule.rule - Order Delivered Rule fired

TIBCO Cloud Events sends Prepare Delivery Event to Delivery Service

Your journey of innovation does not stop here...

TIBCO provides the most optimal path to digital innovation

A single platform takes you from **connected** to **intelligent**.

Thank you