KubeCon | CloudNativeCon

North America 2018

Deep Dive: CNCF Serverless WG & CloudEvents

Cathy Zhang - Cathy.H.Zhang@huawei.com
Clemens Vasters - clemensv@microsoft.com

Agenda

- Serverless Workflow
- CloudEvents Deep Dive
- Q&A

KubeCon | CloudNativeCon

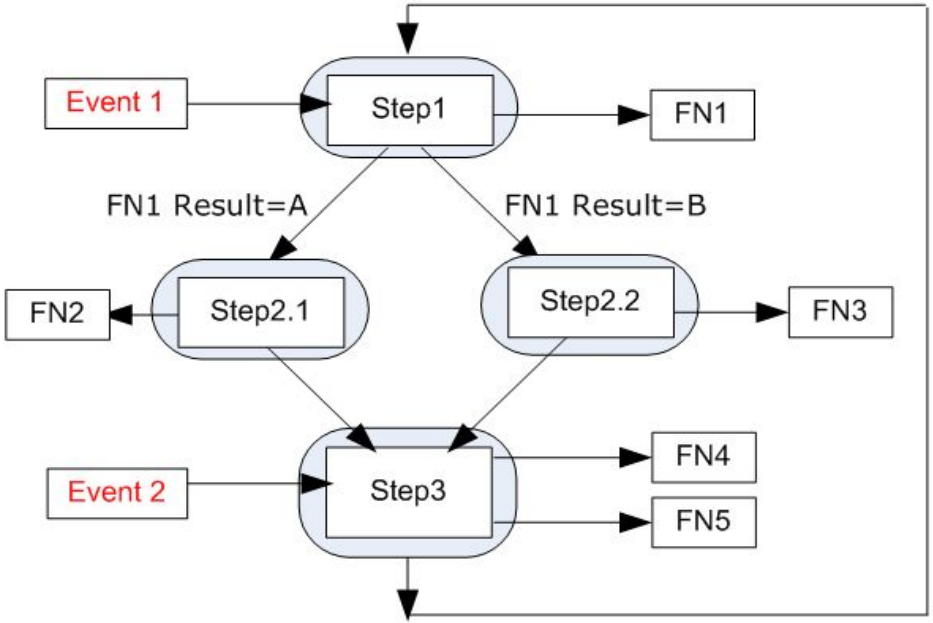
North America 2018

- What is Serverless Workflow
- Why Do We Need a Workflow Specification
- Workflow Specification Scope
- Overview of Workflow Primitives
- Example Use Cases

What is Serverless WorkFlow

Many serverless applications are composed of multiple steps of function execution.

Function Workflow = Function Execution Flow



Why is the WorkFlow Spec Needed

Work Decoupled:

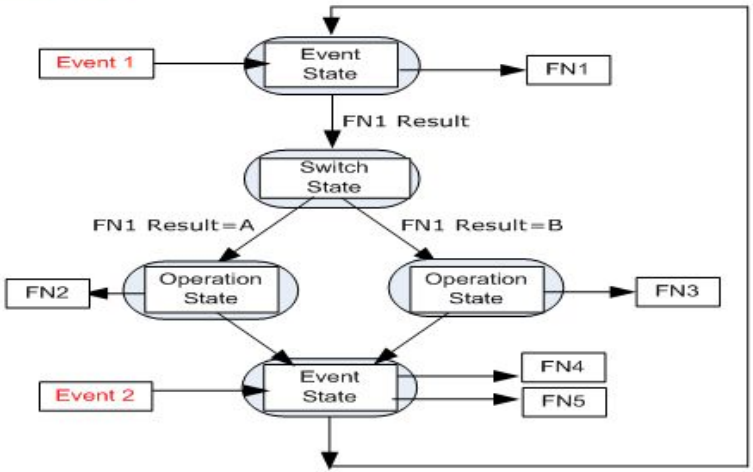
App Developer: just provides function logic

Serverless Platform: takes over all other work

Control of Workflow

App's System ⟹ Cloud Serverless Platform

Workflow Specification



WorkFlow Specification Scope

1. Goal: Provide a standard way for the users to specify the function workflow for their serverless application so as to facilitate portability of the application across different vendors' platforms.
2. Non-Goal: Not a specification for function signature. Not a specification for event format or metadata definition
3. Specification Scope includes:
   - Steps/states involved in a serverless application workflow.
   - Event triggers for each function in the workflow (reference to events, event detail is defined in CloudEvents Specification)
   - Functions involved in each step (reference to functions)
   - How information is passed and filtered

Workflow Overview (Key Primitives)

A Serverless Function Workflow can be naturally modeled as a state machine. The Workflow may be invoked from a CLI command or triggered dynamically upon the arrival of an event. There are three key parts in a Workflow Model

- List of Events (trigger the function execution):

  For example: Storage event, Web Request event,
  Video Streaming event, DB Access event,
  Email event, etc.

- List of States (build the workflow structure):
  - event-state
  - operation-state
  - switch-state
  - delay-state
  - parallel-state
  - end-state with success or failure status

- Actions/Functions associated with a state (executes the business logic):
  - Directives for parallel/sequential execution of functions
  - Directives for Retry
  - Directives for information filtering/passing

Stock Trade Application Use Case

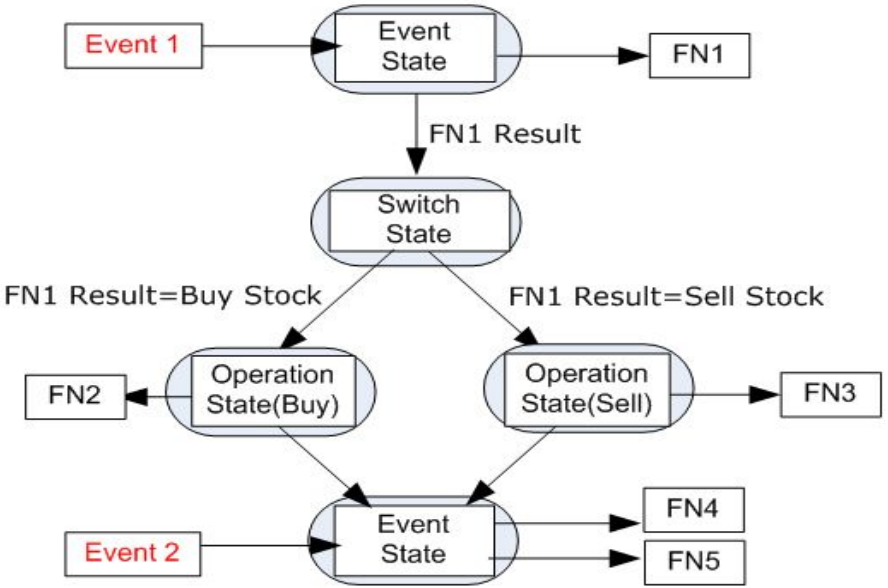
Stock Trade System

Every stock trade process begins with a customer submitting a stock trade request

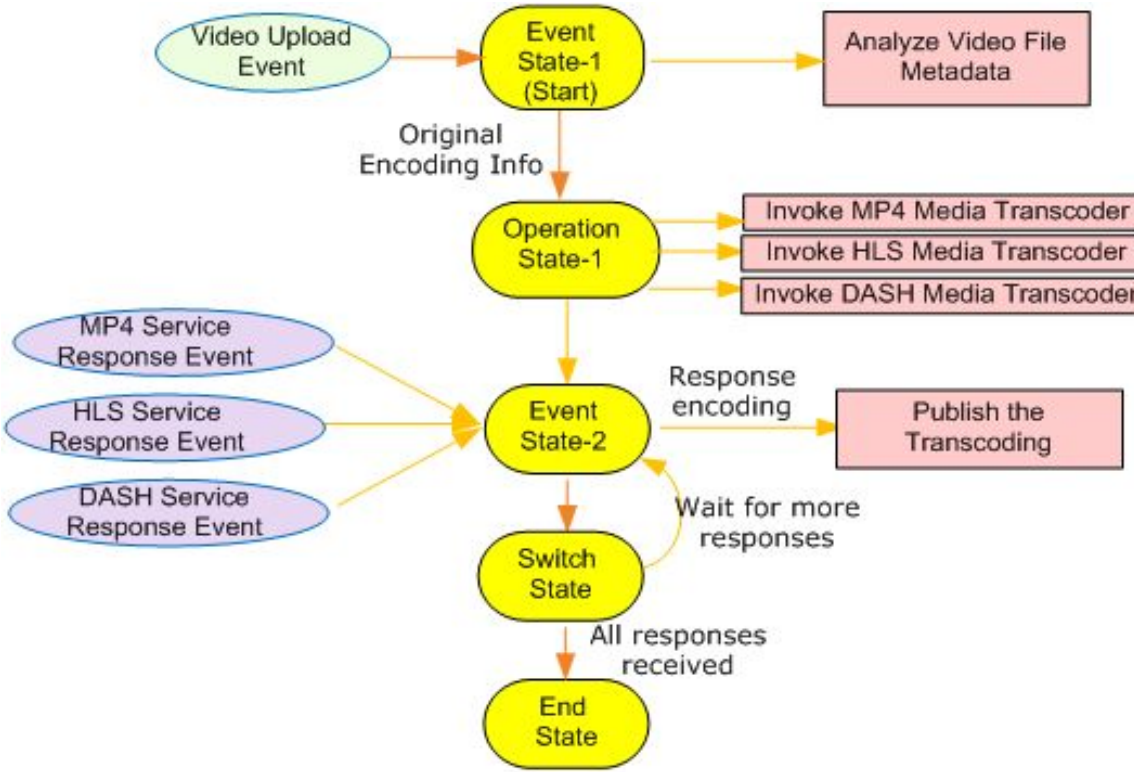
- When receiving a stock trade request, authenticate the customer, interpret the trade request (buy or sell), validate the stock ticker, check the transaction against security/trading rules, etc.
- A "buy stock" request---verify enough money to buy the stock, debit the stock cost from the account, and send a response for the customer to confirm the buy operation。
- A "sell stock" request--- verify enough shares for the sell transaction, send response for the customer to confirm the sell operation。
- Then the system will wait for the customer' s confirmation of the stock transaction
- When receiving the confirmation, execute the stock transaction, update the stock account with new number of shares, update the customer' s checking/saving account with new balance, set the transaction status to completion, and send out notification to customer about the transaction
- In parallel, the system will update the central stock transaction statistics DB with new buy/sell information

Stock Trade Application Use Case

- **Event1**: stock trade request from the customer
- **FN1**: authenticate the customer, interpret the trade request, validate the stock ticker, check the transaction against security/trading rules, process the trade request, return the processing result—buy or sell
- **FN2**: execute the buy stock operation. Verify enough money to buy the stock, debit the stock cost from the account, send response for the customer to confirm the buy operation。
- **FN3**: execute the sell stock operation. Verify enough shares for the sell transaction, send response for the customer to confirm the sell operation。
- **Event2**: customer confirms the stock transaction
- **FN4**: execute the transaction, update the stock account with new number of shares, update the checking/saving account with new balance, set the transaction status to completion, send out notification to customer
- **FN5**: Update the central stock transaction statistics DB with new buy/sell information



Video Streaming Application Use Case



Benefit of Breaking an Application into a Workflow

- *Modify/replace/add a function without impacting other functions or other steps.*
- *Flexibility to add a new step to provide new service, such as creating thumbnails, inserting ads in the video stream, inserting a video transcription, etc.*

CloudEvents Deep Dive

Eventing vs Messaging

- Events and messages are both mailing envelopes for data, decorated by metadata – but they are different.
- Events carry facts. They report things that have happened.
  - State transitions, observed conditions, objects having been created, …
- Messages carry intents. The sender expects something to happen.
  - Command execution, job handling, workflow progress, …
- Events are published as an information option for interested subscribers. The audience size may be zero or many.
- Messages are sent to handlers. There may be delivery and handling status feedback, replies, conversations, or complex control flows like Workflows and Sagas. The audience size may be one or many.

CloudEvents - Base Specification

- CloudEvents is a lightweight common convention for events.
- It's *intentionally* not a messaging model to keep complexity low.
  - No reply-path indicators, no message-to-message correlation, no target address indicators, no command verbs/methods.
- Metadata for handling of events by generic middleware and/or dispatchers
  - What kind of event is it? **eventtype**
  - When was it sent? **eventtime**
  - What context was it sent out of? **source**
  - What is this event's unique identifier? **eventid**
  - What's the shape of the carried event data? **contenttype**, **schemaurl**
- Event data may be text-based (esp. JSON) or binary

CloudEvents - Event Formats

- Event formats bind the abstract CloudEvents information model to specific wire encodings.
- All implementation must support JSON. JSON is the default encoding for where metadata text must be rendered, e.g. HTTP header values
- AMQP type system encoding defined for metadata mapping to AMQP properties and annotations
- Further compact binary event format candidates might be CBOR, or Protobuf.

```
{
  "specversion" : "0.1",
  "type" : "myevent",
  "source" : "uri:example-com:mydevice",
  "id" : "A234-1234-1234",
  "time" : "2018-04-05T17:31:00Z",
  "type" : "text/plain",
  "data" : "Hello"
}
```

JSON Representation

HTTP Transport Binding

- Transport bindings bind the CloudEvent event metadata and data to the transport frame of an existing application or transfer protocol.
- HTTP Transport Binding:
  - Binds a CloudEvent event to the HTTP message. Works for both requests and replies. Does not constrain usage of methods or status codes; can be used for all cases where HTTP carries entity bodies.
  - Structured mode: Complete event including metadata rendered carried in entity body. Upside: Easier to handle/forward
  - Binary mode: Only event data carried in entity body, metadata mapped to headers. Upside: More compact

HTTP Structured Binding Mode


```
HTTP/1.1 POST /myresource
...
content-type: application/cloudevents+json

{
  "specversion" : "0.1",
  "type" : "myevent",
  "source" : "uri:example-com:mydevice",
  "id" : "A234-1234-1234",
  "time" : "2018-04-05T17:31:00Z",
  "contenttype" : "text/plain",
  "data" : "Hello"
}
```

Complete event including metadata rendered carried in entity body. Upside: Easier to handle/forward

HTTP Binary Binding Mode


```
HTTP/1.1 POST /myresource
ce-specversion: 0.1
ce-type: myevent
ce-source: uri:example-com:mydevice
ce-id: A234-1234-1234
ce-time: 2018-04-05T17:31:00Z
content-type: text/plain

Hello
```

Only event data carried in entity body, metadata mapped to headers. Upside: More compact

Other Transport Bindings

- AMQP: ISO/IEC messaging protocol used for a variety of message brokers and event buses; defined in OASIS
  - Binds event to the AMQP message
  - Binary and structured modes
- MQTT: ISO/IEC lightweight pub/sub protocol for device telemetry propagation; defines in OASIS
  - Binds event to MQTT PUBLISH frame.
  - Binary and Structured for MQTT v5
  - Structured mode only for MQTT v3.1.1 (lacks custom frame headers)
- NATS: Text-based lightweight pub/sub protocol
  - Binds event to the NATS message.
  - Structured mode only (lacks custom frame headers)

Thank You!

- Serverless WG : https://github.com/cncf/wg-serverless
  - Workflow: https://github.com/cncf/wg-serverless/tree/master/workflow/spec
- CloudEvents : https://cloudevents.io/
  - Org : https://github.com/cloudevents
  - Spec repo : https://github.com/cloudevents/spec
  - SDKs : https://github.com/cloudevents/sdk-...

- Questions?


KubeCon
CloudNativeCon
North America 2018