

FaaS is Not Only the Serverless Stream Processing with Serverless

Jun Makishi, Kensaku Komatsu
NTT Communications



KubeCon



CloudNativeCon

Europe 2019



Kensaku Komatsu

Technical Manager, NTT Communications



@komasshu



<https://github.com/kensakukomatsu>



Jun Makishi

Senior Architect, NTT Communications



@JunMakishi



<https://github.com/j-maxi>

Main topic of this talk

Practical study and our experiment of

“Serverless Real-time Media Processing Platform for WebRTC interface”

built with **Kubernetes** and **open ecosystems**.



- recording
- detection
- recognition
-



Today, we'll talking about ...



New type of **Serverless** - Real-time Media Processing



Kubernetes



cloudevents

and more



Motivation

“Serverless Real-time Media Processing Platform for
WebRTC interface”

Our business on



Innovation Through Real-time Communication.

ECLWebRTC is a platform that lets you add video conversation to applications, Web sites and IoT devices.

Announcements SkyWay iOS / Android SDK v1.1.0 released

[See all](#)

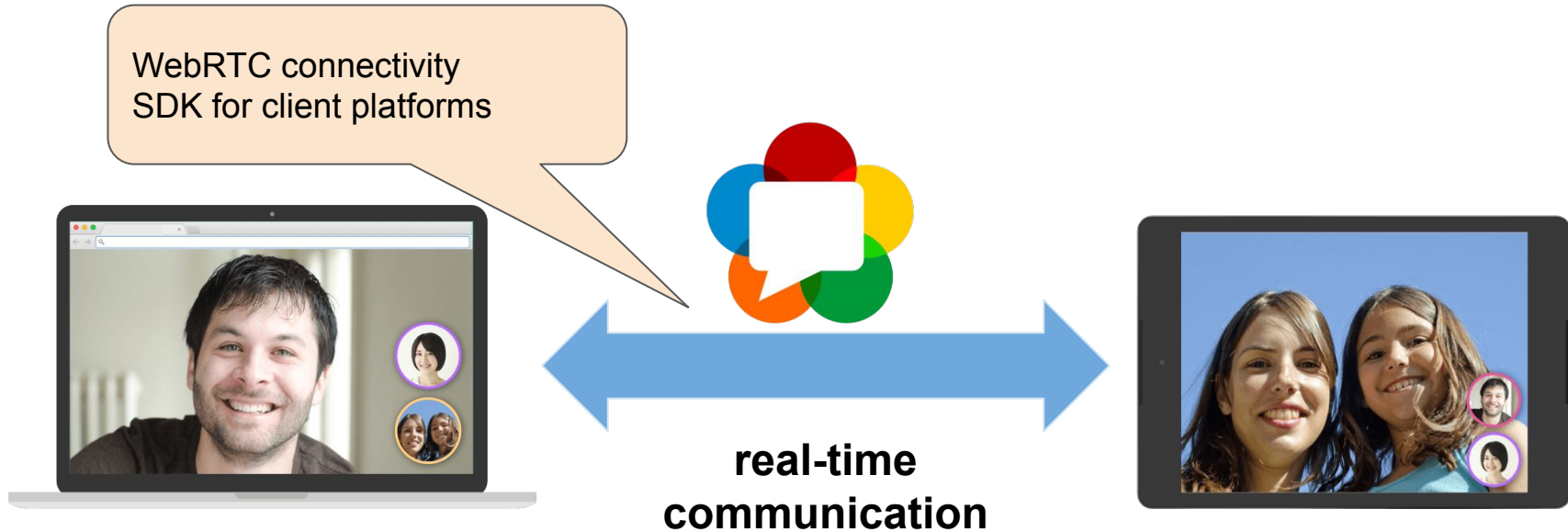
About ECLWebRTC

The need for online real-time communication such as video conferencing, contact centers, remote work support, online education and live distribution is continuously increasing. It has become easier to implement online real-time communication as WebRTC, a standard technology for real-time voice/video/data communications, emerged.

<https://webrtc.ecl.ntt.com/en/> 6

With ECLWebRTC, you can enjoy video/voice conversations and data communication easily without setting up and operating servers normally required for

Current model

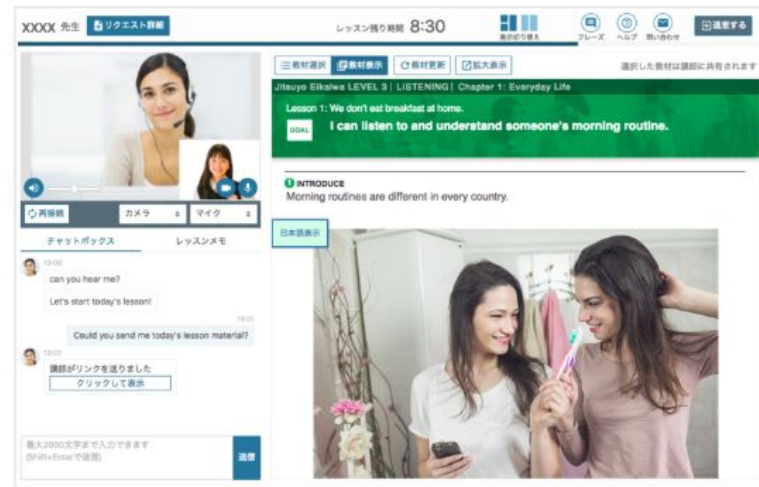


Use cases

- online education
- online healthcare
- video conference
- remote expert
- robot control
-

特徴2：一画面で映像やチャット、教材表示が完結するシームレスなレッスン体験

講師の映像やチャット、教材が一画面上に表示されることにより、Webブラウザと通信ソフトを行き来する複雑な操作が必要なく、集中してレッスンを進めることができます。「映像モード」と「教材モード」の表示切替機能により、教材を使ったレッスンや、講師の画面を大きくして口元を見ながら発音の練習も可能です。



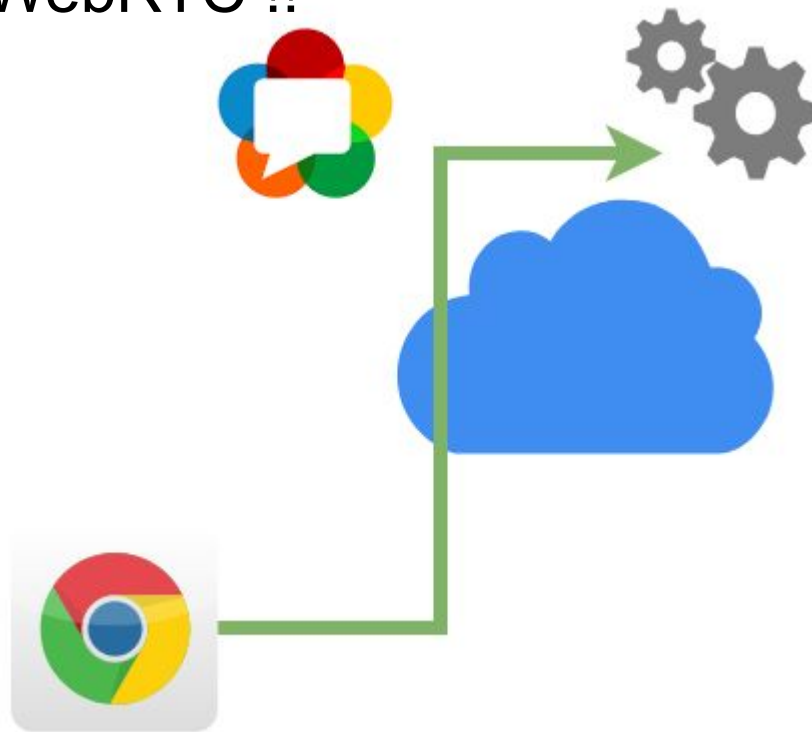
「教材モード」時のレッスンルームの画面イメージ（PCブラウザ版）

Voice from customers

- recording
- voice recognition
- object detection
- live splitting
- AR/MR
-



Need cloud computing power and full-managed platform for WebRTC !!

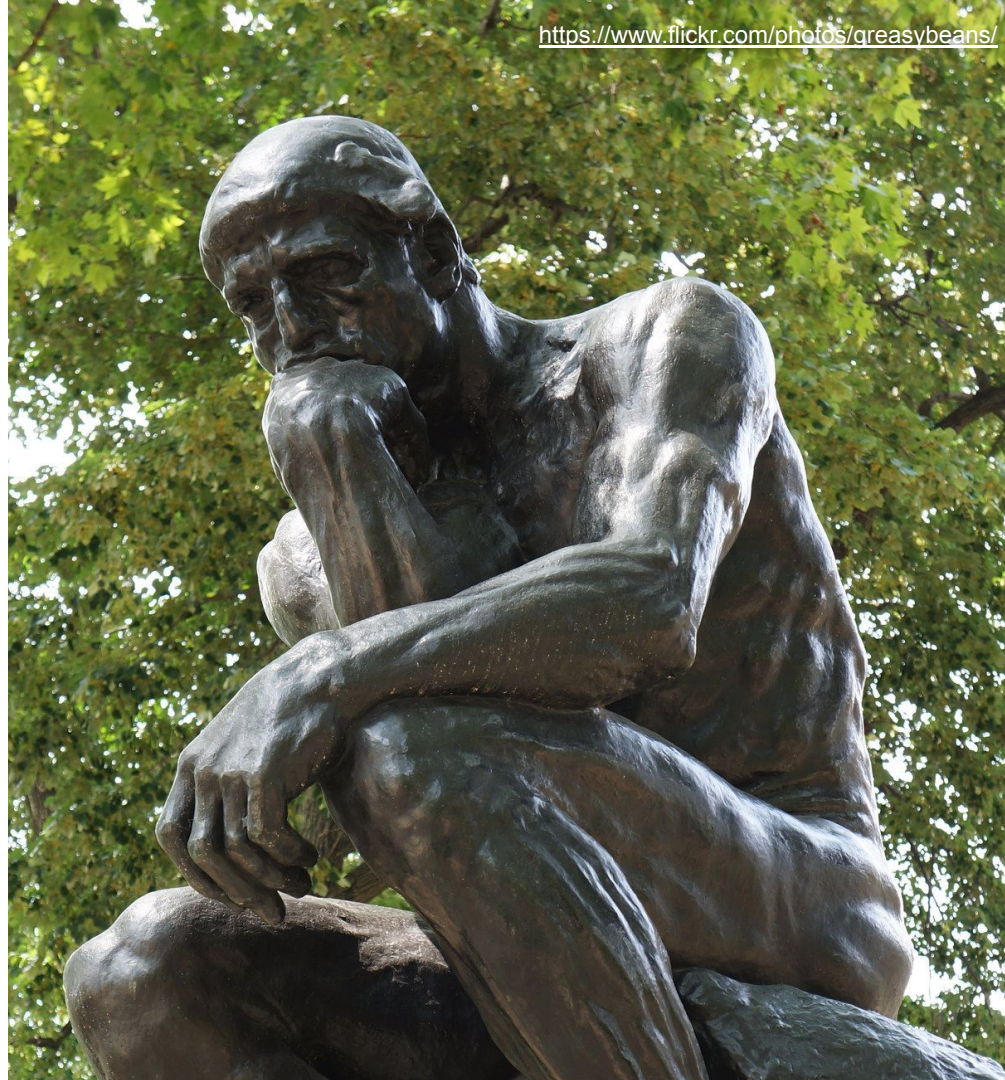


We thought ...

WebRTC IF PaaS ?

Serverless with
Media streaming ?

Long-term session
lifecycle ?



Our Descision

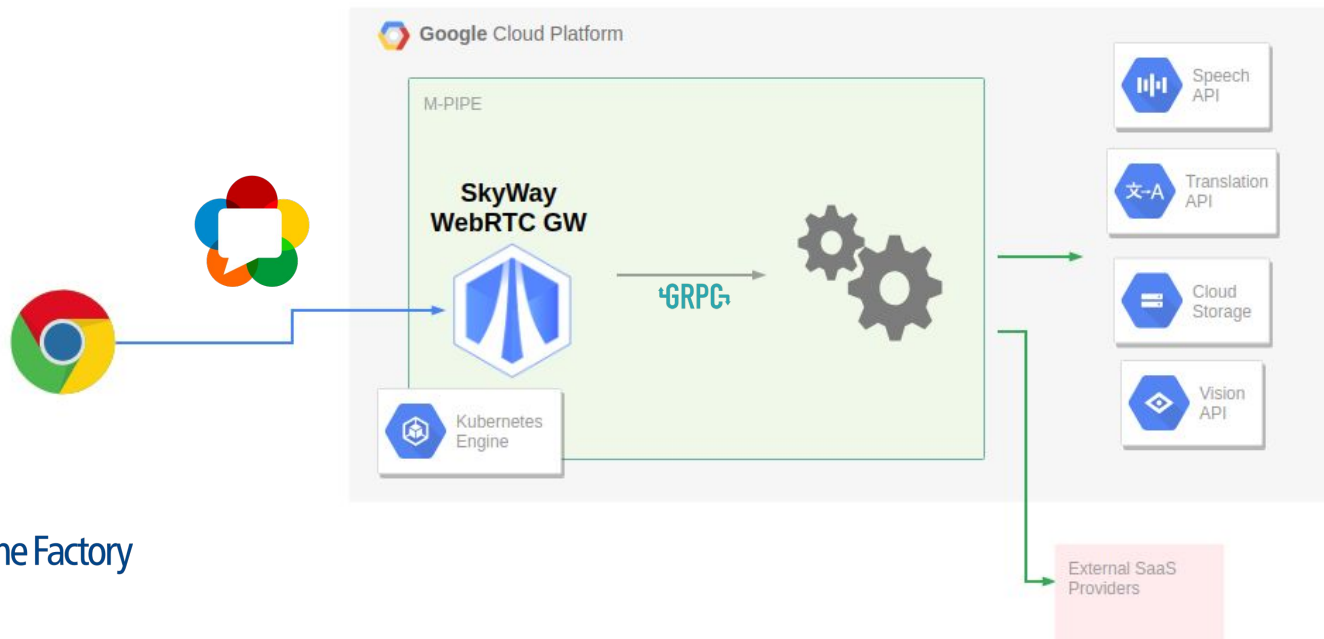


Media Pipeline Factory

Evolve your business with real-time data enriched with Cloud APIs.

Challenge

Built our own “Serverless Real-time media processing platform”
using our **WebRTC Gateway**



Media Pipeline Gateway x +

https://dashboard.m-pipe.net/projects/mpipe-demo/build/11d999dd-97cc-4492-99c2-d89a98647cd6

Media Pipeline Factory Build Deploy Insights

mpipe-demo

Pipeline Templates +

- hoge (1324f429)
- multi language translator scenario (0)
- recognition-translation demo (11d999dd-97cc-4492-99c2-d89a98647cd6)
- test (7c7463ab)

Build / recognition-translation demo v4

Properties Delete

```

    graph LR
      A[WebRTC gateway] --> B[File Writer]
      A --> C[Voice Recognizer]
      C --> D[Translator]
      D --> E[AWS DynamoDB]
  
```

ID	Name	Node Type	Service Type	Wires To
translator-001		translator	On demand	aws-dynamodb-001
voice-recognizer-001		voice-recognizer	On demand	translator-001
webrtc-gateway	Skyway WebRTC Gateway	webrtc-gateway	On demand	file-writer-001,voice-recognizer-001
aws-dynamodb-001		aws-dynamodb	On demand	
file-writer-001		file-writer	On demand	

Demo

Code snippet



custom function

- Input Stream

```
const { InputStream } = require('skyway-m-pipe-sdk/connector');

const inputStream = new InputStream();

// you need to set hostname and port number of previous component
// please make sure that same token with previous as well
inputStream.start({ host: inHost, port: inPort, token });

inputStream.on( 'data', data => {
  // #=> data.type - arbitrary type data in string format
  //      data.meta - arbitrary meta data in string format
  //      data.payload - arbitrary payload data in binary format
})
```

- Output Stream

```
const { OutputStream } = require('skyway-m-pipe-sdk/connector');

const outputStream = new OutputStream();

outputStream.start({ port: outPort, token })

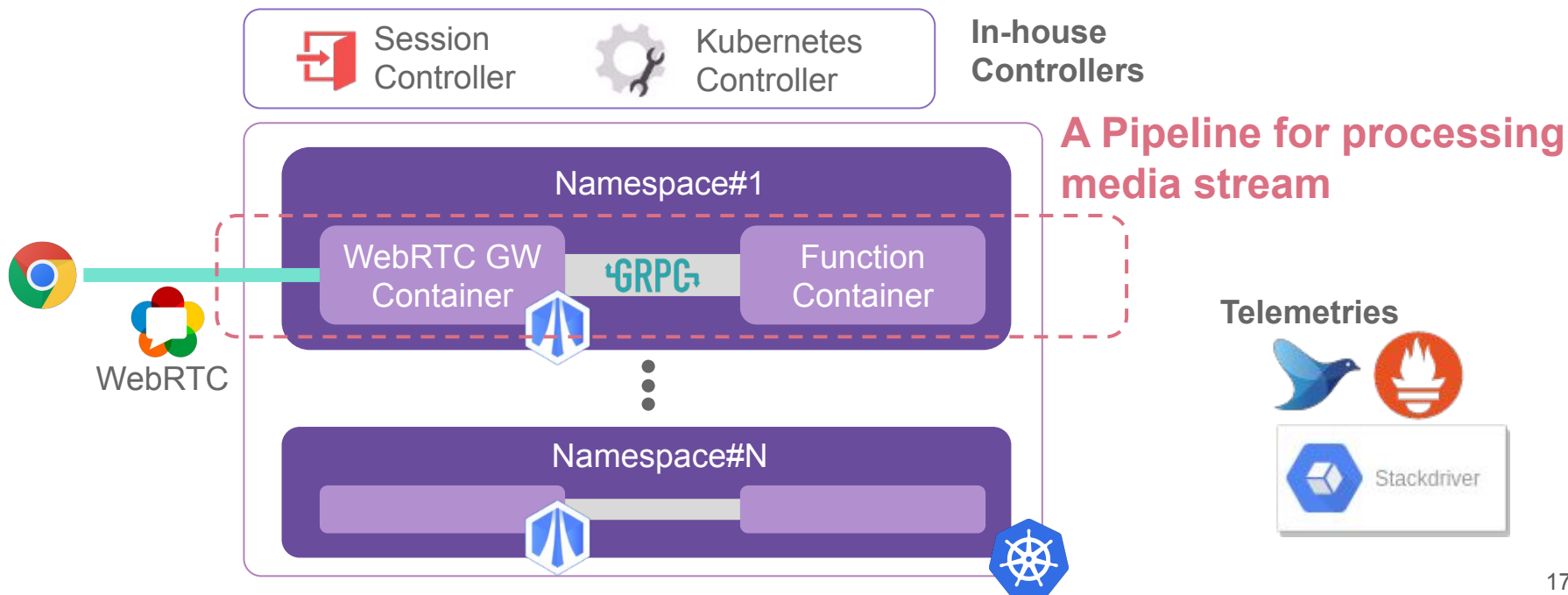
outputStream.write({
  type: 'test-stream',
  meta: JSON.stringify({ name: test, ts: Date.now() }),
  payload: Buffer.from( 'Hello world' )
})
```

Our solutions

“**Serverless** Real-time Media Processing Platform for **WebRTC** interface”

Platform Overview

- Run a chain of containers for media streaming with Kubernetes



Serverless

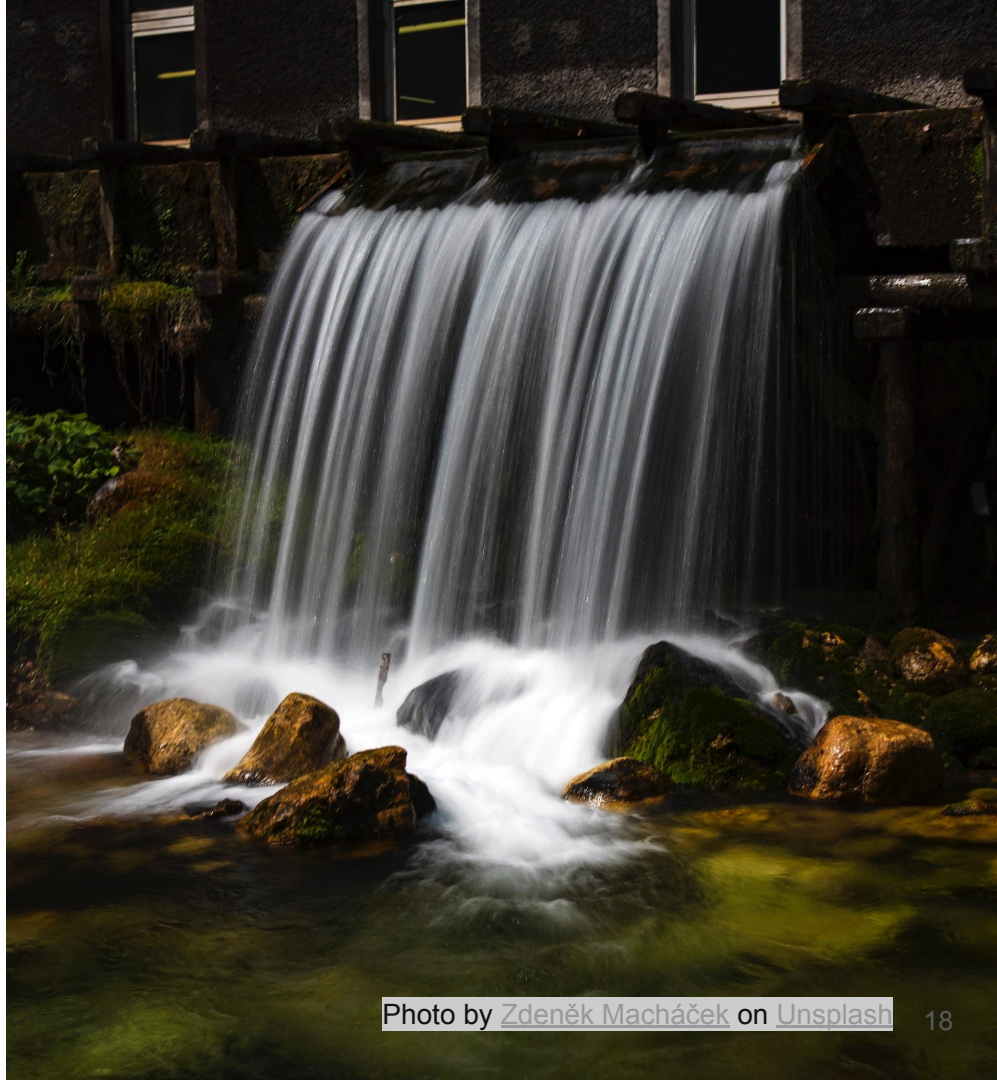
Run a function per event

Event

= Media Streaming

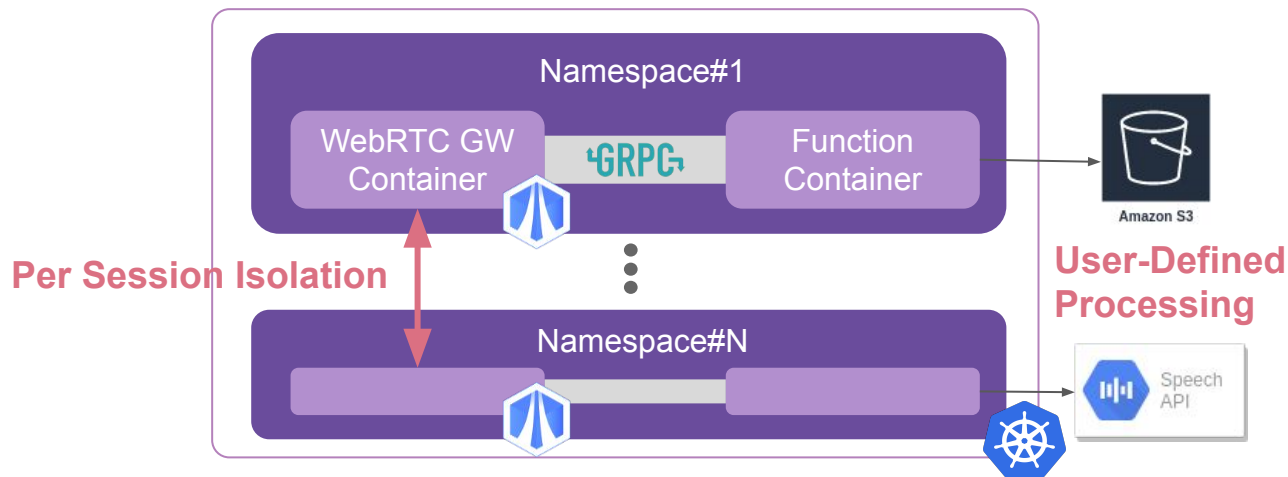
Function

= Real-time Processing



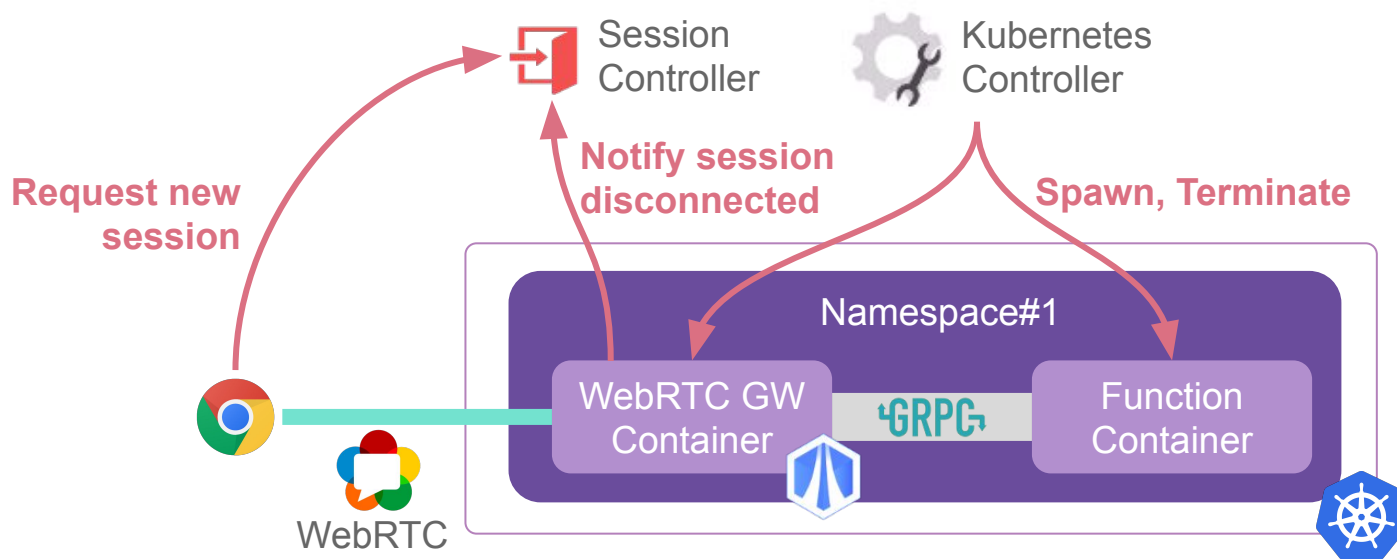
Serverless - Cascaded Functions per Session

- Allocate containers **per streaming session**.
- **Cascade** Gateway to user defined functions.
- Isolate sessions by container.
 - Horizontally scalable.
 - Split failure domain.



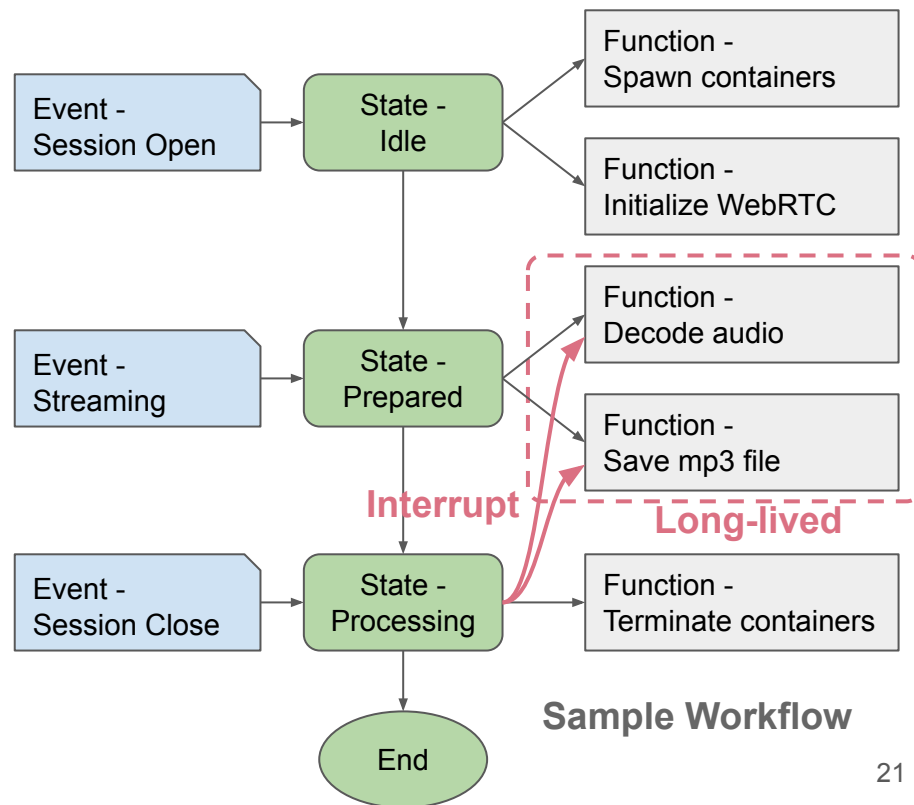
Serverless - Long-lived functions

- Run **long-lived containers** to follow streaming lifecycle.
 - Spawn containers for a new session.
 - Terminate containers on the session disconnected.



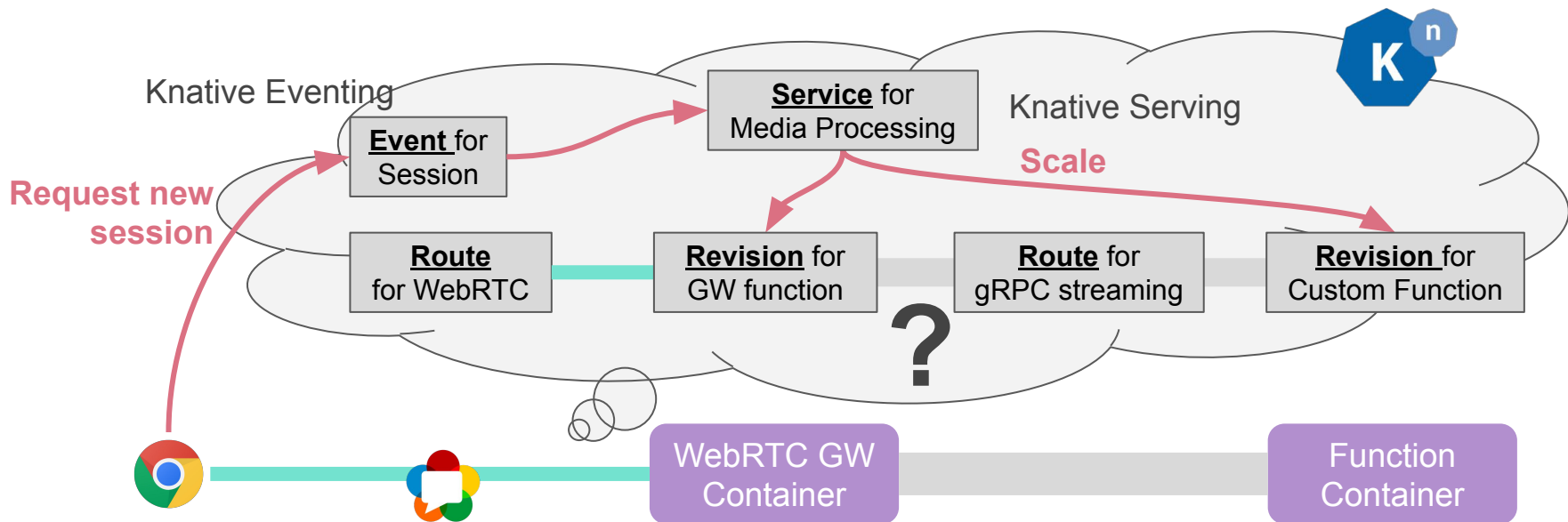
Serverless - Workflow for Long-lived Functions

- Our specifications
 - **Interrupt action** to stop function.
 - **Sub-workflow** for long-lived function.
- Two types of in-house controllers.
 - Kubernetes controller.
 - Session controller.



Serverless - Possible integration with Knative

- New event for WebRTC session lifecycle.
- Route function output to another function for streaming.



Internals - Kubernetes

Replicate and distribute Pods.

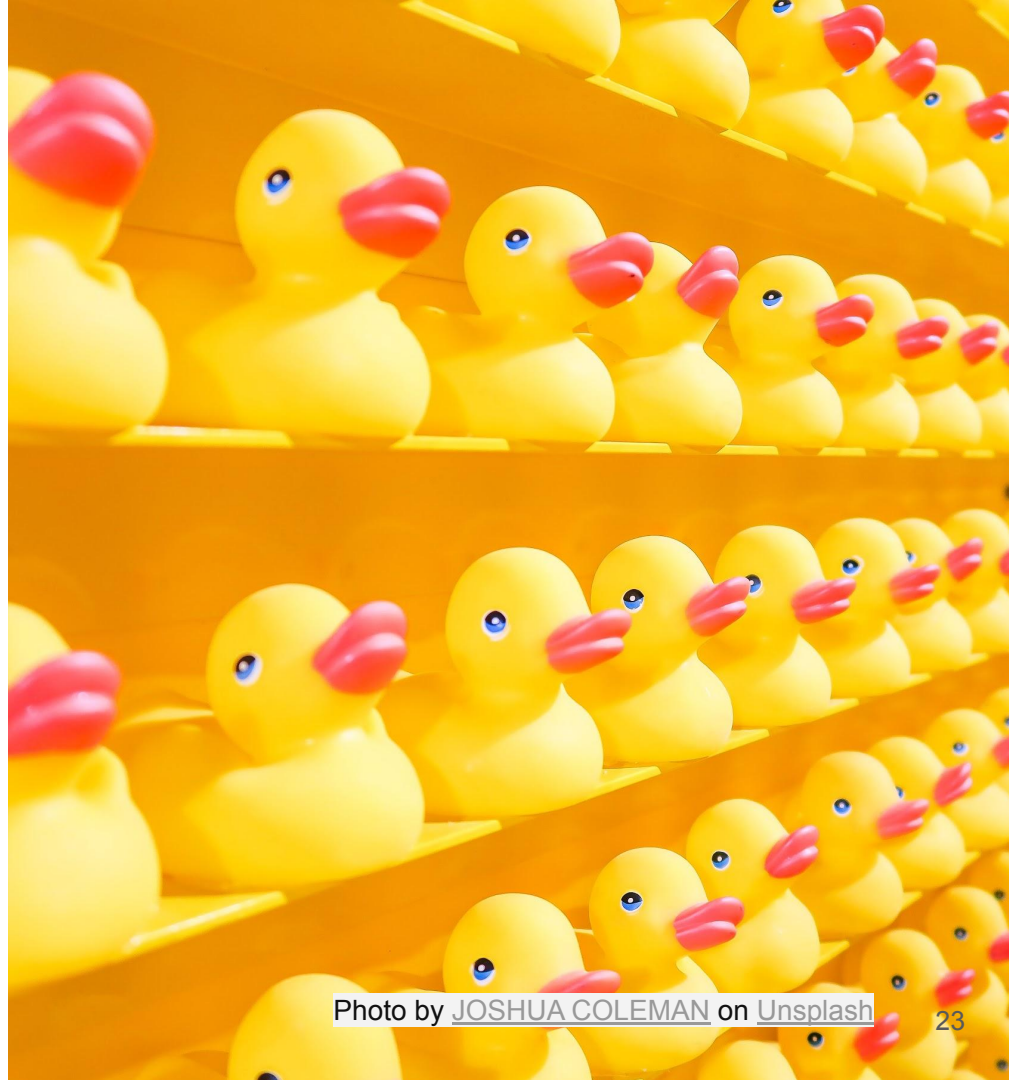
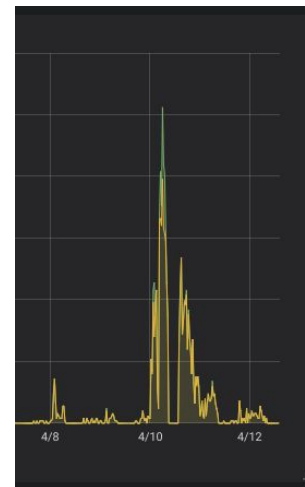
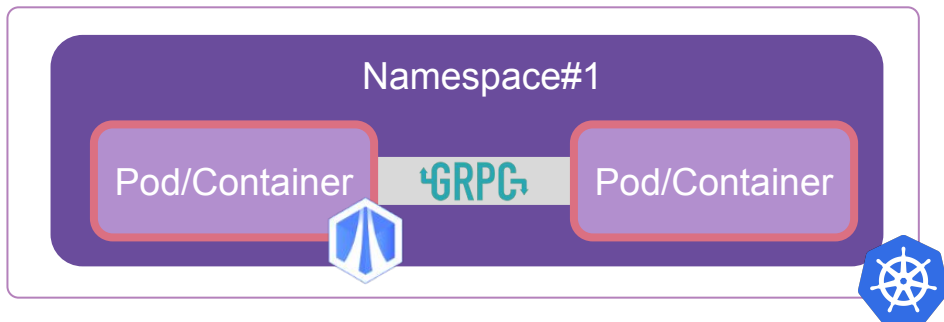


Photo by [JOSHUA COLEMAN](#) on [Unsplash](#)

Kubernetes - Function Pods

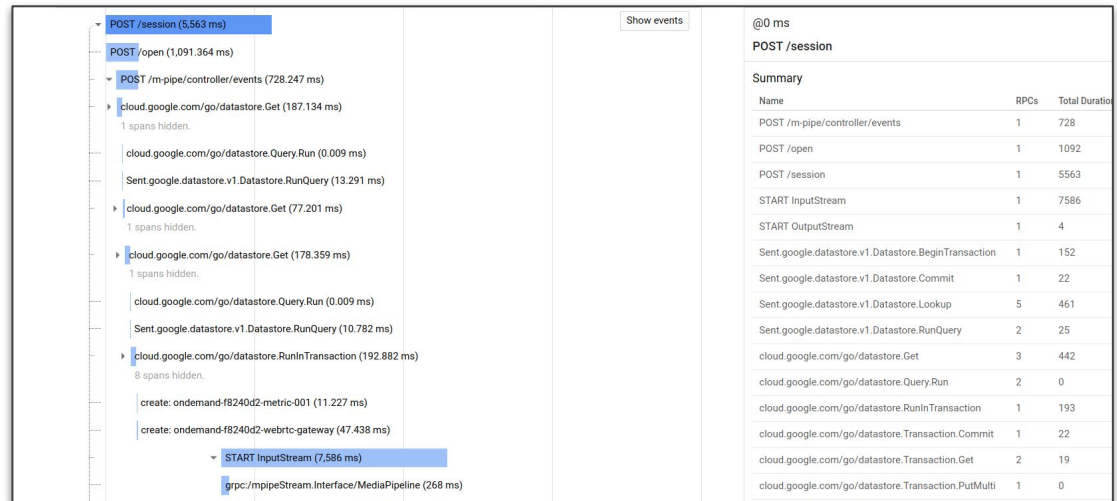
- Directly call Pod API to execute function.
- Challenges:
 - Sync multiple containers for a session.
 - API performance - Deal with **spike**. Spawn in FIFO.



Pod API Spike

Kubernetes - Observability

- Distributed Tracing for container orchestration.
 - **Correlate** each function's start-up latency to an end-end workflow.
- Challenges:
 - Bind trace context to container lifecycle, per-function initialization.



Kubernetes - Multi Tenancy

- Isolate session and pipeline per customer.
 - Special inter-function authentication mechanism.
- Challenges:
 - Credential Management, Security

Internals - Open Ecosystem

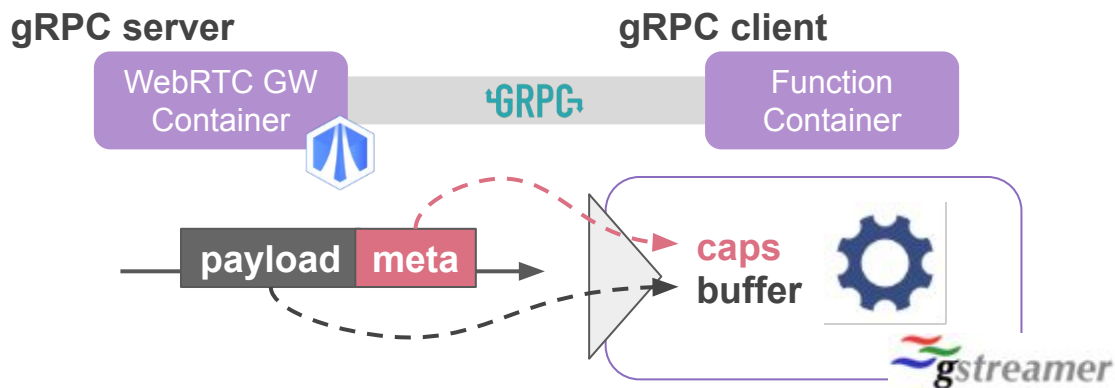
Integrate building blocks of open ecosystem.



Photo by [Ryan Fields](#) on [Unsplash](#)

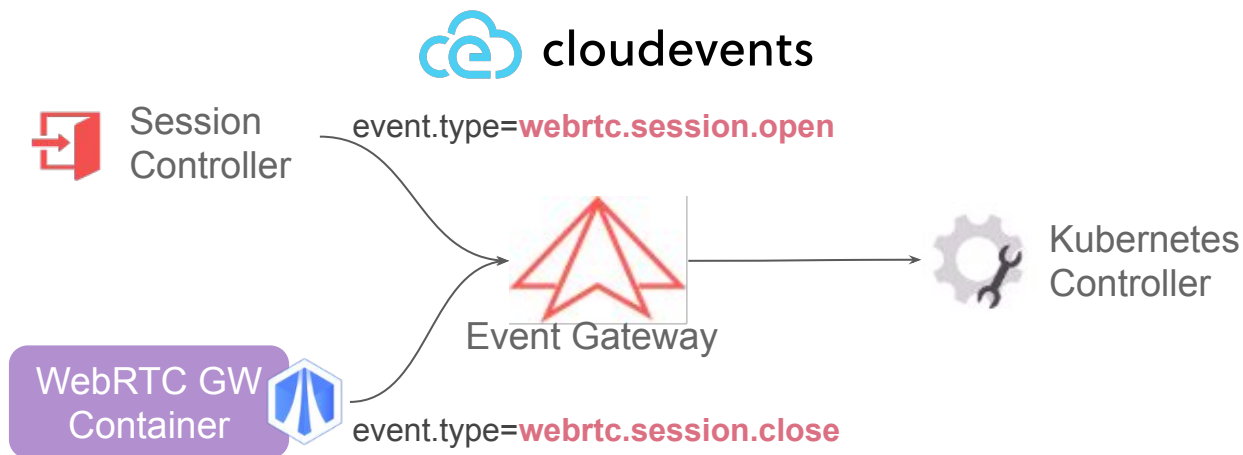
gRPC

- Server streaming RPC
- .proto message for media metadata and payload.
 - Inter-function operability, Gstreamer ready.
- Challenges:
 - Transport choice - UDP.
 - Authentication - Server validates token from client.



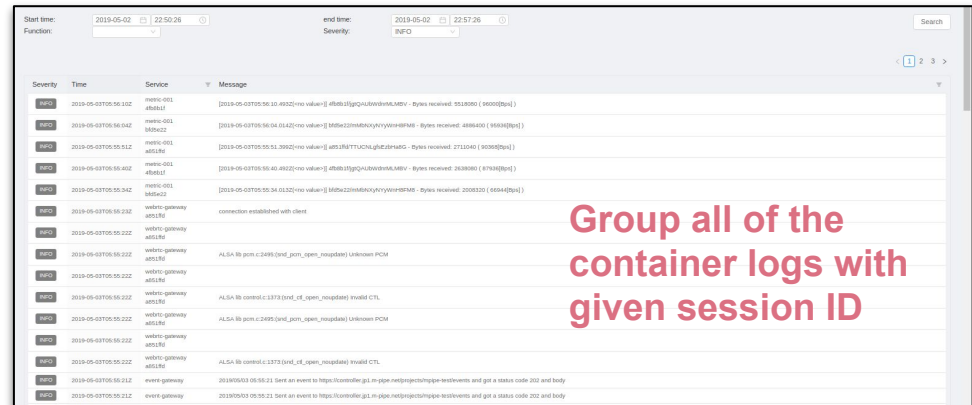
Cloudevents

- Defined streaming session events with Cloudevents v0.1.
 - **Loosely coupled** controllers and components.
- Challenges:
 - Event Tracing



Telemetry

- Group logs, metrics, and tracing with session ID.
 - Correlation based on Kubernetes metadata.
- Challenges:
 - Custom metadata correlation - Metadata Agent.
 - **Actionable metrics** - Drill down. Jump to other metrics with given metadata.



Start time: 2019-05-02 22:50:28 end time: 2019-05-02 22:57:26
Function: Severity: INFO

Severity	Time	Service	Message
INFO	2019-05-03T05:56:10Z	metrics-001-atsfcl1	[2019-05-03T05:56:10.493Z][no value] [4b8b113gq4u6w6v6m6A.M6V] - Bytes received: 5518000 (9030Bps)
INFO	2019-05-03T05:56:04Z	metrics-001-atsfcl1	[2019-05-03T05:56:04.014Z][no value] [M6b6220m6b6k6yryy6e@PM] - Bytes received: 4868400 (9684Bps)
INFO	2019-05-03T05:55:51Z	metrics-001-atsfcl1	[2019-05-03T05:55:51.399Z][no value] [ats12677U2Chg4h6c6r6w6S] - Bytes received: 3711040 (9030Bps)
INFO	2019-05-03T05:56:40Z	metrics-001-atsfcl1	[2019-05-03T05:56:40.482Z][no value] [4b8b113gq4u6w6v6m6A.M6V] - Bytes received: 263000 (8794Bps)
INFO	2019-05-03T05:55:34Z	metrics-001-atsfcl1	[2019-05-03T05:55:34.013Z][no value] [M6b6220m6b6k6yryy6e@PM] - Bytes received: 2008320 (6664Bps)
INFO	2019-05-03T05:56:23Z	wttrnc-gateway-atsfcl1	connection established with client
INFO	2019-05-03T05:56:22Z	wttrnc-gateway-atsfcl1	
INFO	2019-05-03T05:56:22Z	wttrnc-gateway-atsfcl1	ALSA lib pcm.c:2495:(snd_pcm_open_noupdate) Unknown PCM
INFO	2019-05-03T05:56:22Z	wttrnc-gateway-atsfcl1	
INFO	2019-05-03T05:56:22Z	wttrnc-gateway-atsfcl1	ALSA lib control.c:1373:(snd_ctl_open_noupdate) Invalid CTL
INFO	2019-05-03T05:56:22Z	wttrnc-gateway-atsfcl1	ALSA lib pcm.c:2495:(snd_pcm_open_noupdate) Unknown PCM
INFO	2019-05-03T05:56:22Z	wttrnc-gateway-atsfcl1	
INFO	2019-05-03T05:56:22Z	wttrnc-gateway-atsfcl1	ALSA lib control.c:1373:(snd_ctl_open_noupdate) Invalid CTL
INFO	2019-05-03T05:56:21Z	everx-gateway	20190503 05:56:21: Save an event to https://controller.gsl.m-pipe.net/projects/tripwire-testbeds and get a status code 202 and body
INFO	2019-05-03T05:56:21Z	everx-gateway	20190503 05:56:21: Save an event to https://controller.gsl.m-pipe.net/projects/tripwire-testbeds and get a status code 202 and body

Group all of the container logs with given session ID

Recap

Motivation:

Server-side (Cloud) real-time media processing for **WebRTC**



Solution:

Serverless Real-time Media Processing Platform

→ Empowered by Kubernetes, and other open ecosystem



Challenges:

- Introducing a **new serverless workflow and lifecycle**
- Hardening the Kubernetes integration to achieve production ready

Thank you



Media Pipeline Factory

Evolve your business with real-time data enriched with Cloud APIs.

<https://webrtc.ecl.ntt.com/m-pipe/en>

SDK of Media Pipeline Factory : <https://github.com/nttcom/skyway-m-pipe-sdk>

Sample codes of function container : <https://github.com/nttcom/skyway-m-pipe-components>

SkyWay WebRTC Gateway : <https://github.com/skyway/skyway-webrtc-gateway>