Open Source Summit Japan 2018

Thursday, June 21 ・ 12:00 - 12:40

FUJITSU

shaping tomorrow with you

FaaS Shell

Multi-Cloud Portable Serverless Function Workflow

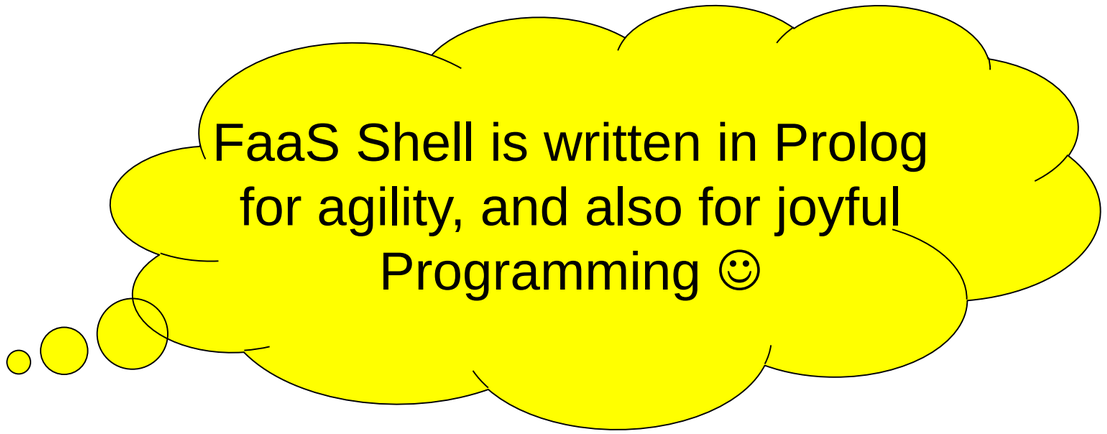
Naohiro Tamura

Professional Engineer
Fujitsu Limited

About me

FUJITSU

- Recent work
  - Apr. 2017 – Open Source PaaS, FaaS, Cloud Native Development (FaaS Shell)
  - Oct. 2014 – Mar. 2017 Open Source IaaS Development (OpenStack Ironic)
  - - Sep. 2014 Proprietary System and Resource Management Software Development
- Expertise
  - System and Resource Management Software Development
- Personal Interest
  - Functional and Logic Programming



TOC

FUJITSU

- What is Function Workflow?
- What is FaaS Shell and Why?
- Demo
- How does FaaS Shell work?
- Summary
- Q&A

What is Function Workflow?

FUJITSU

If we look at Serverless Function from FPL's point of view…

- Serverless Function
  - Small for doing single task
  - Stateless and Immutable
  - Event Driven and Reactive
  - Auto scale
  - Pay per use
- Functional Programming Language (FPL) Function
  - Small for doing single task
  - Stateless and Immutable
  - Good at Event Driven and Reactive
  - Good at Concurrent and Parallel

- FPL main theme is how to compose function
  - sequential, conditional, repetition, parallel, exceptional handling, retry

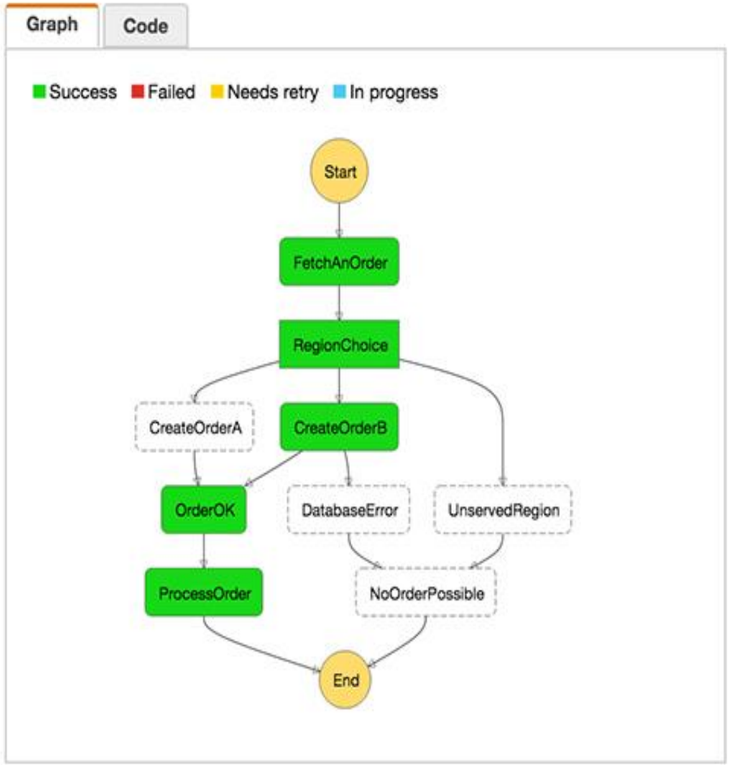
- Serverless Function requires Function Composition as well as FPL

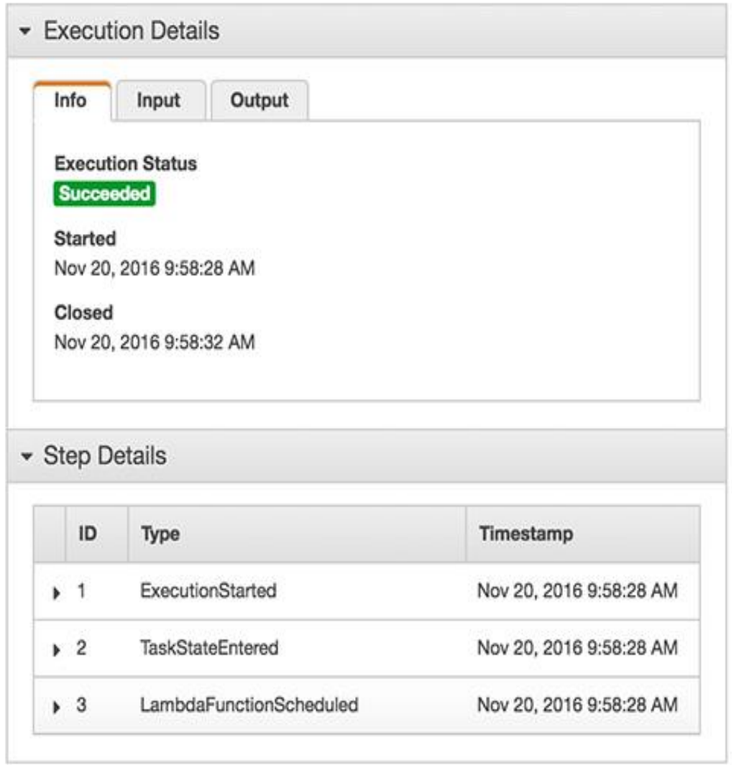
  ⇒ That is Serverless Function Workflow

Serverless Function Workflow

- Consists of 3 elements, for example AWS case
  - Serverless Function : AWS Lambda
  - Serverless Function Workflow : Step Functions
  - Workflow Language : Amazon States Language

- Workflow = Composition
  - Sequential
  - Conditional
  - Repetition
  - Parallel
  - Exceptional handling
  - Retry





Amazon States Language (ASL)

FUJITSU

- State Machine
  - Pass
  - Task
  - Choice
  - Wait
  - Succeed
  - Fail
  - Parallel

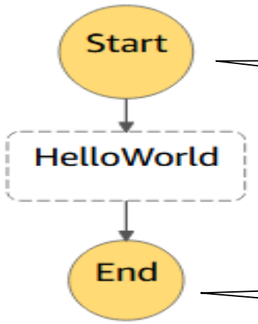
JSON format

```
{
  "Comment": "A Hello World example of the Amazon States Language using a Task state",
  "StartAt": "HelloWorld",
  "States": {
    "HelloWorld": {
      "Type": "Task",
      "Resource":"arn:aws:lambda:us-east-2:410388484666:function:hello",
      "End": true
    }
  }
}
```

Entry State

Task State

Specify AWS Lamda Function
in arn (Amazon Resource Name)



① Input {“name”: “**FaaS Shell**”}

② Output {“payload”: “Hello, **FaaS Shell**!”}

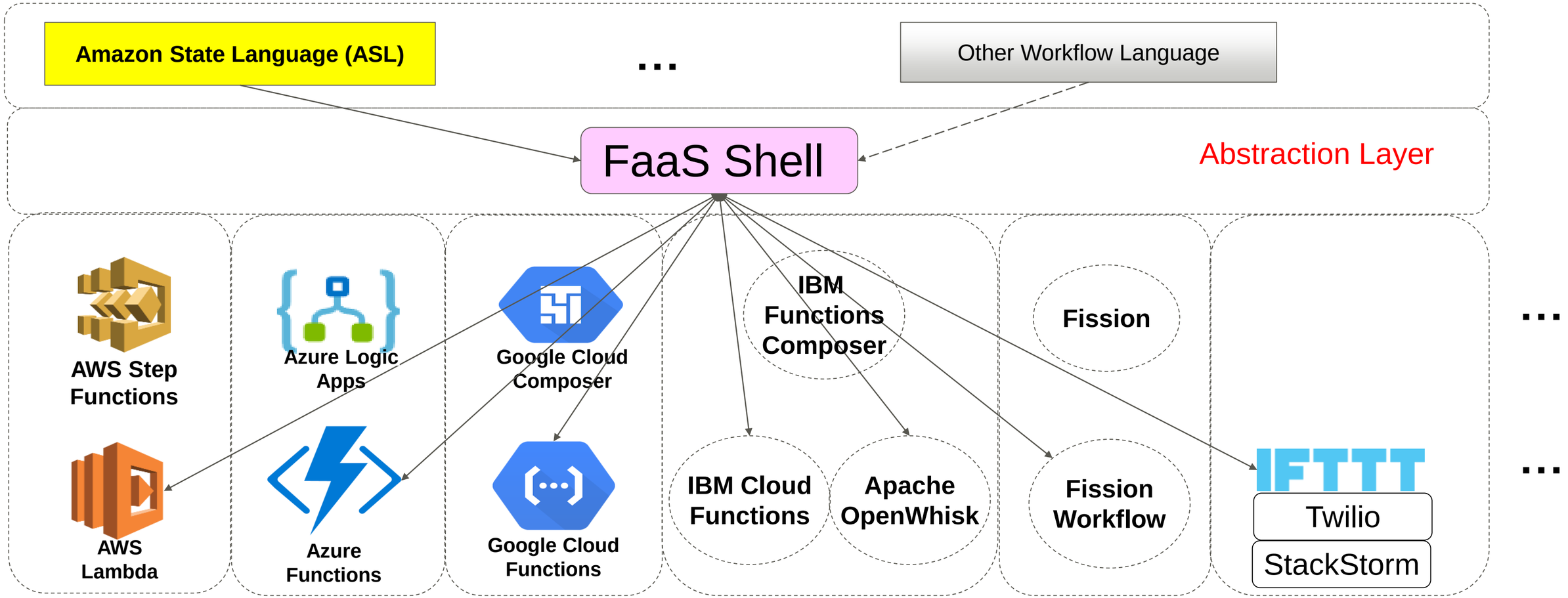
TOC

FUJITSU

- What is Function Workflow?
- What is FaaS Shell and Why?
- Demo
- How does FaaS Shell work?
- Summary
- Q&A

What is FaaS Shell?

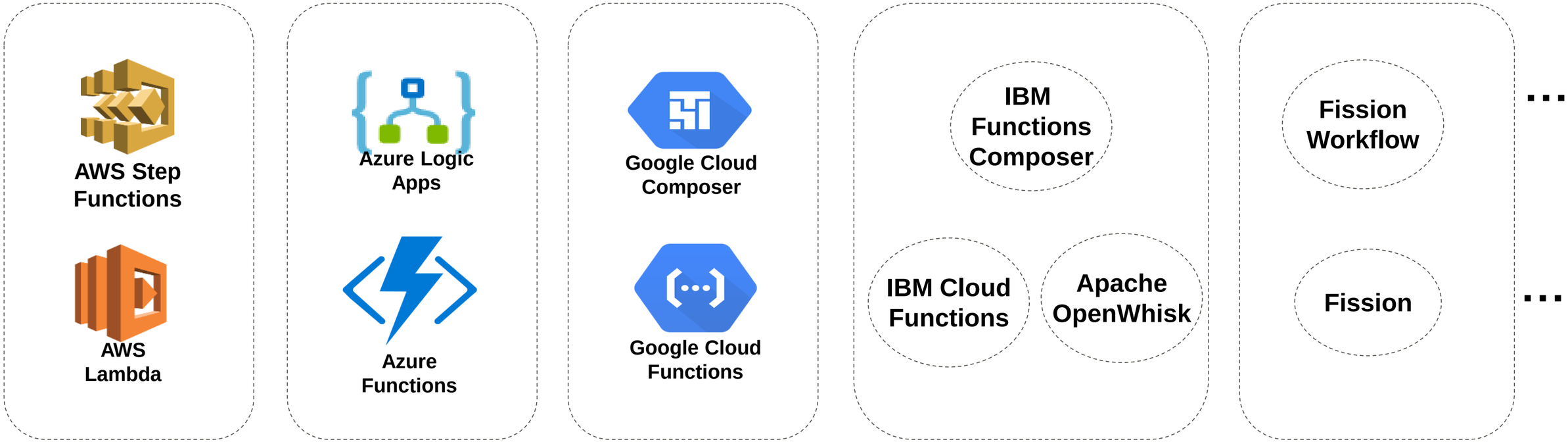
- It’s a shell which enables to exploit Serverless Functions across Multiple Clouds
  - Covers major 4 FaaS Providers as well as reusable Functions
  - Works as abstraction layer with keeping agility for future changes



Why FaaS Shell?

FUJITSU

- Serverless Function Workflow Landscape - Serveless Silo
  - Each provider or platform has its own workflow service
  - There is no interoperability among providers
  - This situation will continue along with competing each other



Benefit of using FaaS Shell

FUJITSU

FaaS Shell helps us shift Single Strategy to Multi Cloud Strategy

- 1st Stage: Single Cloud Strategy
  - Commit to one cloud vendor after thorough investigation
  - Staying vendor lock-in state is nothing wrong if you and your customer are happy

Demo Next

- 2nd Stage: Multi Cloud Strategy
  - Serverless Function doesn't cost at all unless called
  - Each Cloud has strengths and weaknesses, own characteristics, especially in AI area
  - Reusable FaaS such as IFTTT (if-this-then-that), Twilio, StackStorm, Node-RED, etc are available

- Why not exploit the best part of each cloud and integrate them?
  - In fact, we actually have the fortune of selecting the most attractive features from each provider, to enable a multi-cloud strategy.

TOC

FUJITSU

- What is Function Workflow?
- What is FaaS Shell and Why?
- Demo
- How does FaaS Shell work?
- Summary
- Q&A

AI Services Evaluation Demo Background

FUJITSU

Which provider's service is suitable for a specific app? We need evaluation!

Natural Language Processing

- Amazon Comprehend
- Amazon Translate
- Amazon Transcribe
- Amazon Lex
- Natural Language API
- Translation API
- Watson Language Translator
- Watson Language Classifier
- Language understanding intelligent service
- translator
- textanalytics
- emotion
- web-language-model
- recommendations
- Watson Tone Analyzer
- Watson Personal Insights

Dataset and Algorism are different each other

Speech Recognition

- Amazon Polly
- Speech
- Speech API
- Watson Speech to Text Text to Speech

Image Recognition

- Amazon Rekognition
- Computer Vision
- Vision API
- IBM Visual Recognition

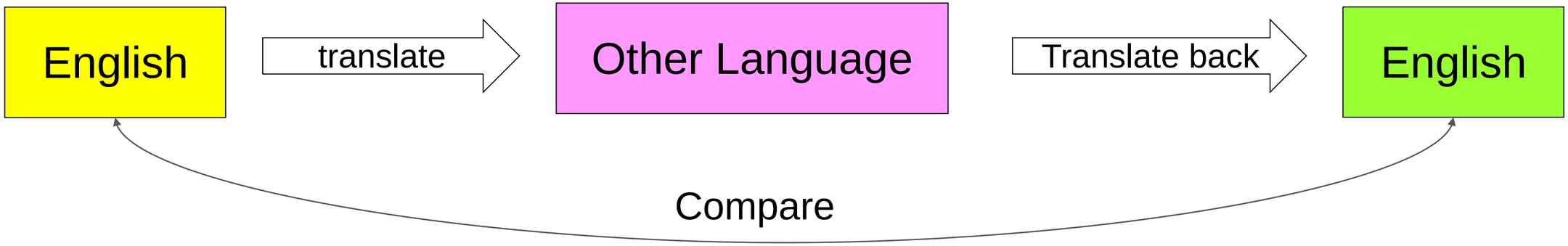
Translation Service Eval. Demo across Multiple Clouds

FUJITSU

- Evaluate Translation using an Ambiguous Sentence
  - very basic Natural Language Processing example
  - it may be that the professor is lecturing with the cat, or that the student has the cat.

the professor lectures to the student with the cat. [1]

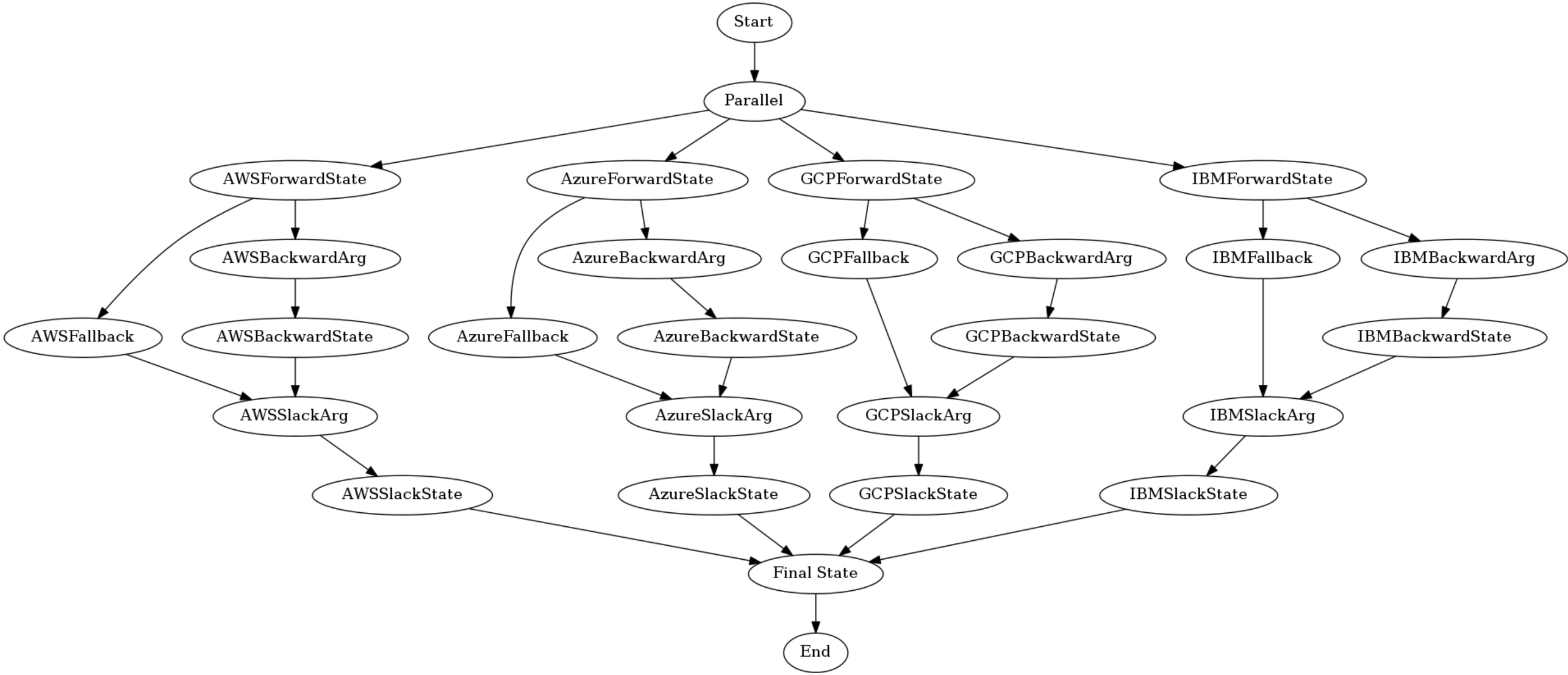
- See how each service translates it into other language and back to English



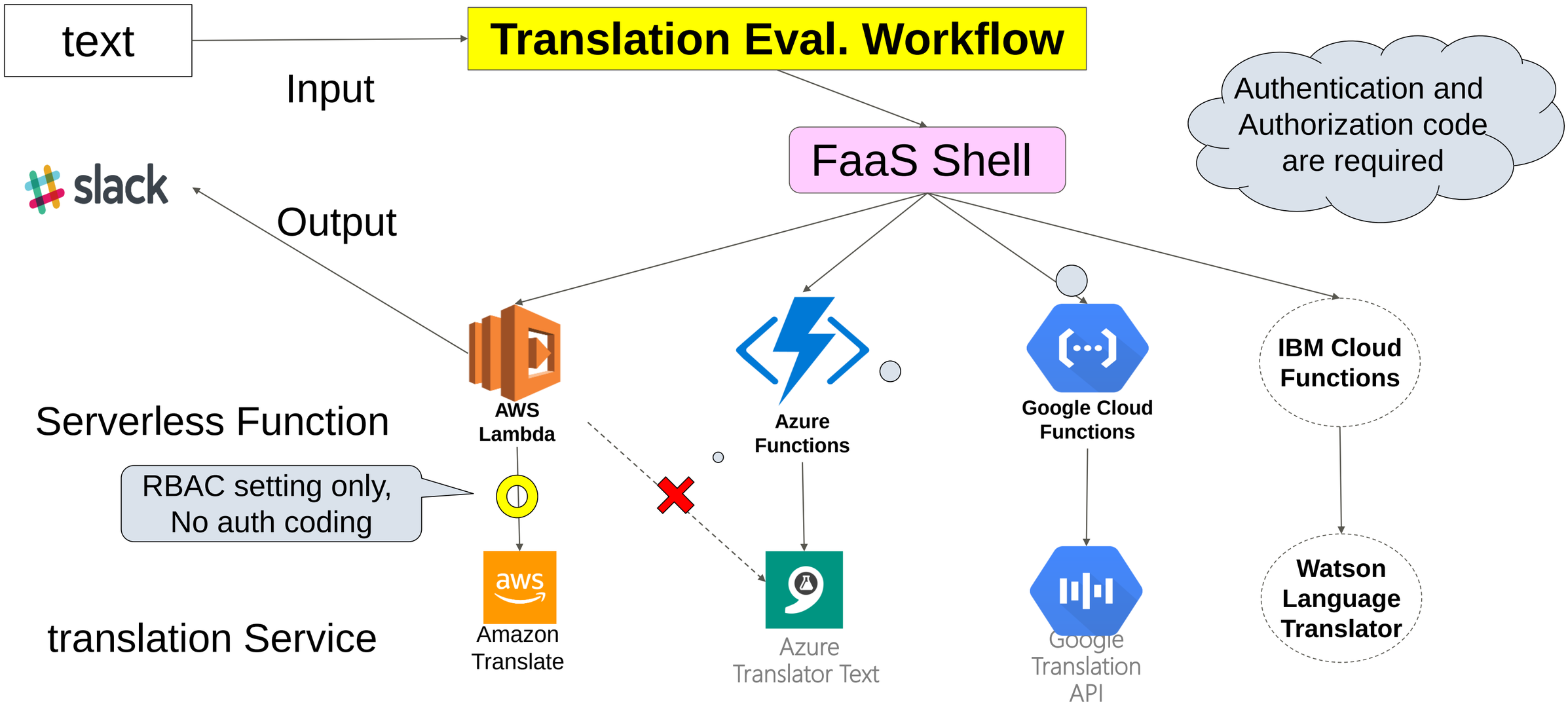
[1] “Structure and Interpretation of Computer Programs”

Translation Evaluation Demo Workflow Graph

- Total 26 States and 4 Parallel Transitions



Translation Evaluation Demo Deployment



Demo Steps through REST Interface

FUJITSU

- Register Statemachine
- Execute Statemachine
  - English to Arabic (ar), and back to English
  - English to Chinese (zh), and back to English
  - English to Japanese (ja), and back to English
- Generate Statemachine Graph (Graphviz DOT)

TOC

FUJITSU

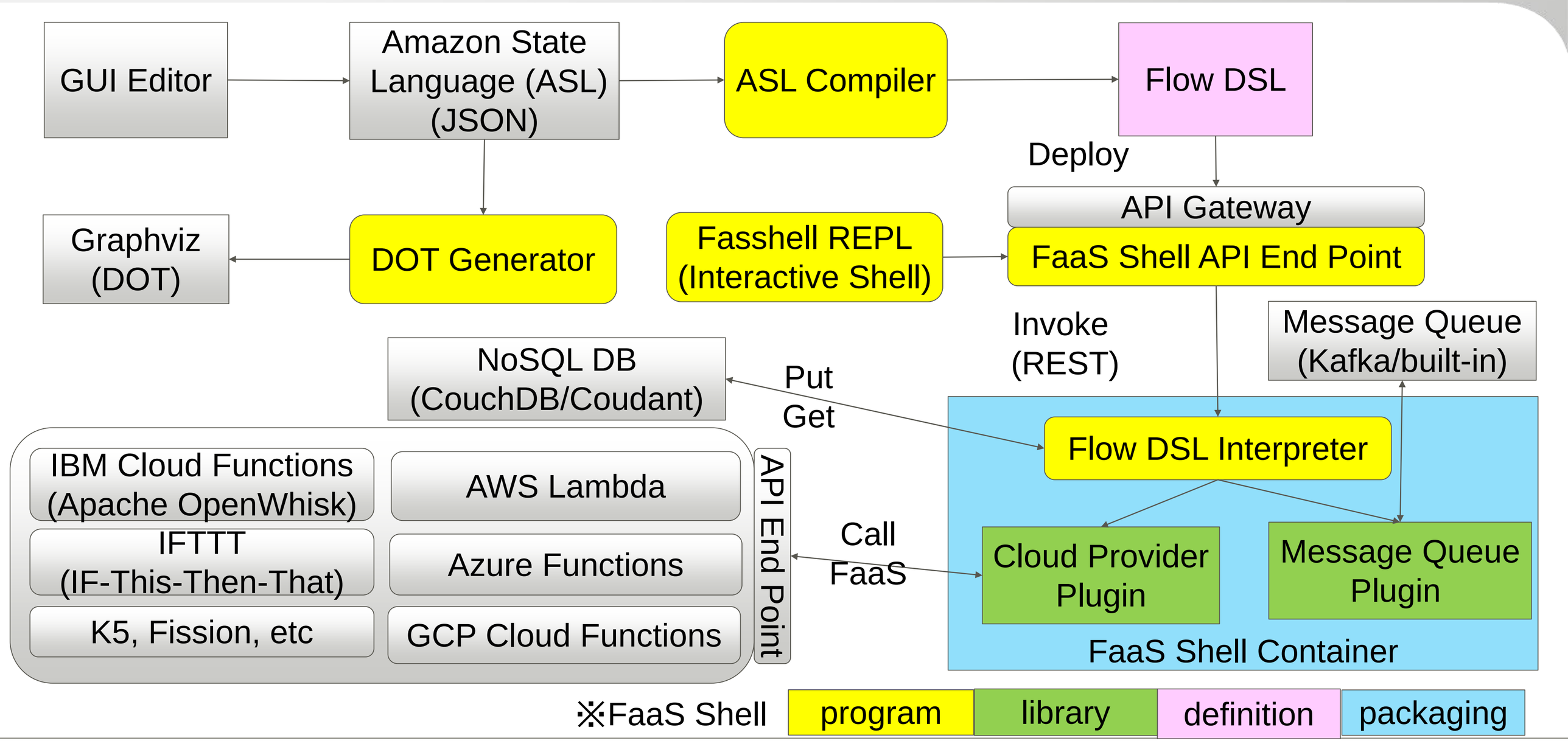
- What is Function Workflow?
- What is FaaS Shell and Why?
- Demo
- How does FaaS Shell work?
- Summary
- Q&A

REST API

FUJITSU

REST API	Function
/	Get version number
/executions/{execution_id}	Get background statemachine information
/activity/{activity_task}	Get/Put/Heartbeat activity
/trigger/{event_state}	Send Event
/faas/{function}	List callable function information in each FaaS
/statemachine/{statemachine}	Register/Execute/Delete/Graph statemachine
/shell/{dsl}	Register/Execute/Delete Workflow DSL

FaaS Shell Components



TOC

FUJITSU

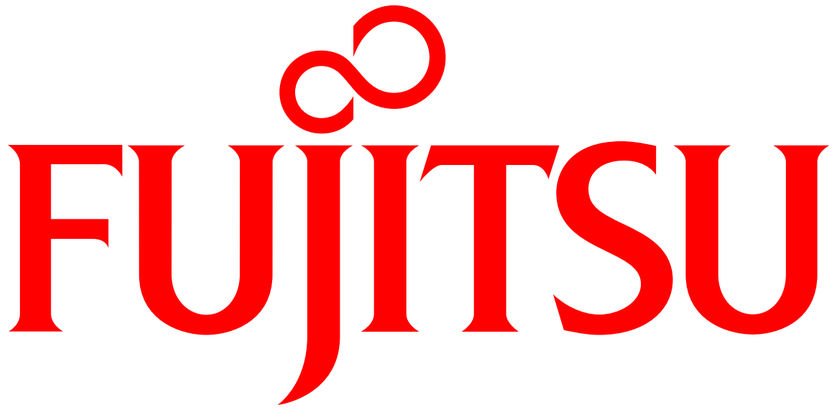
- What is Function Workflow?
- What is FaaS Shell and Why?
- Demo
- How does FaaS Shell?
- Summary
- Q&A

Summary

FUJITSU

- What is FaaS Shell?
  - FaaS Shell is a shell which enables to exploit Serverless Functions across Multiple Clouds
  - FaaS Shell helps us shift Single Strategy to Multi Cloud Strategy
- Future Plan
  - Azure LogicApp Workflow Definition Language
- Visit github “https://github.com/NaohiroTamura/faasshell” and Try demos
  - Especially if you are interested in Functional and Logic Programming
- Welcome your feedback!
  - naohirot@jp.fujitsu.com

Q & A



shaping tomorrow with you

APPENDIX

Translation Service Catalog Spec among Multiple Clouds

FUJITSU

- Surprisingly, AWS is not the leader in this area at this moment

Cloud	Service Name	Current Supported Languages (more than 6,000 languages on the planet)
AWS	Amazon Translate	6 languages to and from English for Preview: (Arabic, Simplified Chinese, French, German, Spanish, and Portuguese) 6 additional languages will be supported soon, (Japanese, Russian, Italian, Traditional Chinese, Turkish, and Czech)
Azure	Translator Text	supports more than 60 languages
GCP	Translation API	supports more than 100 different languages
IBM	Language Translator	Support 14 languages Arabic, Chinese (Simplified & Traditional), Dutch*, English, French, German, Italian, Japanese, Korean, Polish*, Portuguese (Brazil), Russian*, Spanish, and Turkish* *These languages are supported with the early access preview

Not yet

Doesn't work

Comparison

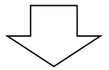
- Difference between Serverless Framework and FaaS Shell
  - Serverless Framework : Portable Solution for Serverless Function
  - FaaS Shell : Portable Solution for Serverless Function Workflow

- Related work
  - Serverless Inc. OSS Event Gateway : Similar but different focus and approach

FaaS Shell Complier and Workflow DSL

- Compiler supported Amazon State Language at first
  - Generate Workflow DSL
- fsm()
  - pass()
  - task()
  - choice()
  - wait()
  - succeed()
  - fail()
  - parallel()

```
{
  "Comment": "A Hello World example using a Task state",
  "StartAt": "HelloWorld",
  "States": {
    "HelloWorld": {
      "Type": "Task",
      "Resource":"arn:aws:lambda:us-east-2:410388484666:function:hello",
      "End": true
    }
  }
}
```



```
fsm([task('HelloWorld',"arn:aws:lambda:us-east-2:410388484666:function:hello",[])])
```

FaaS Shell Runtime

FUJITSU

- Execute Workflow DSL FSM (Finite State Machine)

- Implemented as Meta Interpreter
  - Very concise code in language such as Lisp or Prolog
  - This is important to keep agility for future changes

```
%%
%% reduce(+Dsl, +In, -Out, +EnvIn, -EnvOut)
%%
reduce(fsm(Dsl), I, O, EI, EO) :-          Entry
    !,
    reduce(Dsl, I, O, EI, EO).
reduce([], O, O, E, E) :-                  Exit
    !.
reduce([A|B], I, O, EI, EO) :-             Recursive
    !,
    reduce(A, I, M, EI, EM),
    reduce(B, M, O, EM, EO),
reduce(A, I, O, EI, EO) :-                 Call State
    call(A, I, O, EI, EO).
```

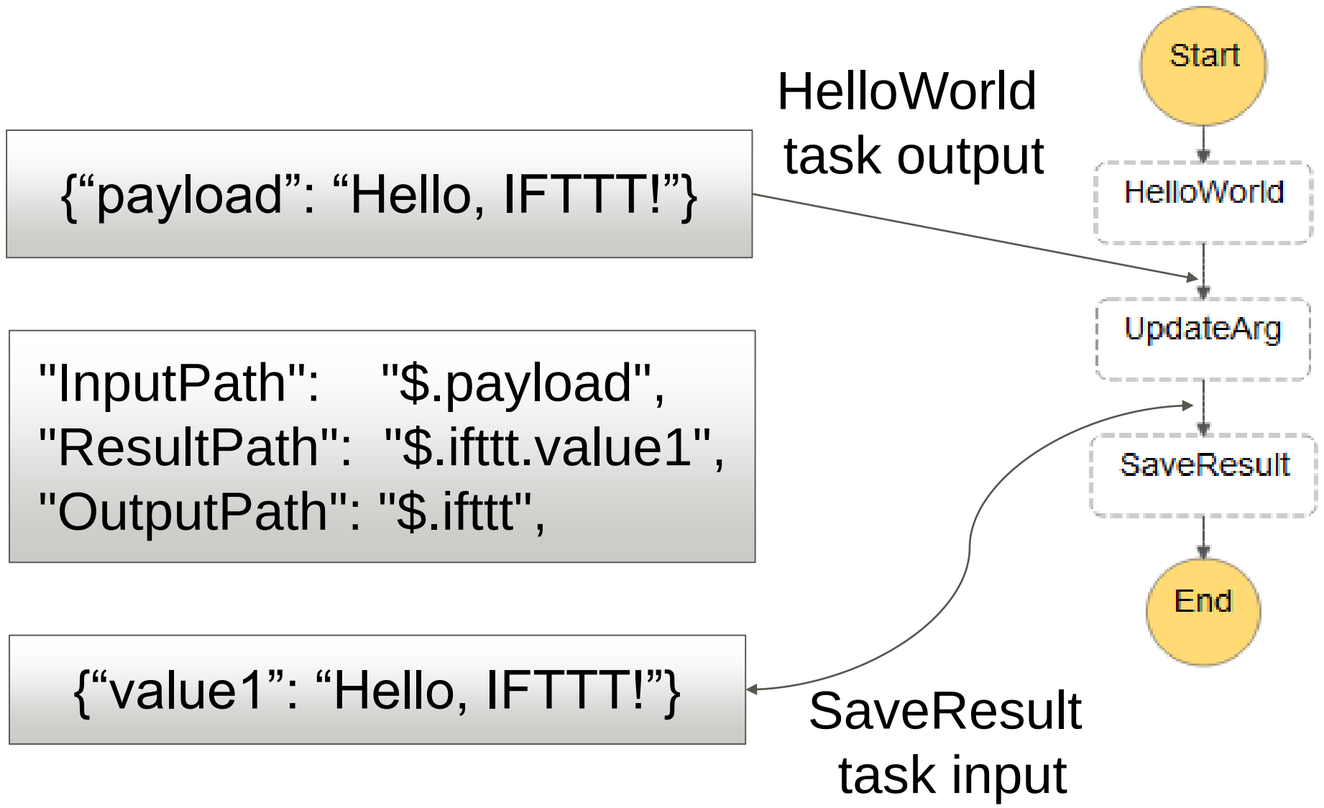
FaaS Shell Runtime Executes Workflow in Batch Mode

FUJITSU

```
$ curl -ksX POST ${FAASSHELL_APIHOST}/statemachine/hello_world_task.json?blocking=true ¥
  -H 'Content-Type: application/json' -d '{"input": {"name": "Curl"}}' -u $DEMO
 {
  "asl": {
    "Comment":"A Hello World example of the Amazon States Language using a Task state",
    "StartAt":"HelloWorld",
    "States": {
      "HelloWorld": {
        "End":true,
        "Resource":"arn:aws:lambda:us-east-2:410388484666:function:hello",
        "Type":"Task"
      }
    }
  },
  "dsl":"fsm([task('HelloWorld',¥"arn:aws:lambda:us-east-2:410388484666:function:hello¥",[])])",
  "input": {"name":"Curl"},
  "name":"hello_world_task.json",
  "namespace":"demo",
  "output": {"payload":"Hello, Curl!"}
}
```

How to resolve Function Input and Output Mismatch?

- State Optional Parameter
  InputPath, ResultPath, OutputPath




```
{
   "Comment": “IFTTT as FaaS Demo",
   "StartAt": "HelloWorld",
   "States": {
      "HelloWorld": {
         "Type": "Task",
         "Resource":"frn:wsk:functions:::function:hello",
         "Next": "UpdateArg"
      },
      "UpdateArg": {
         "Type": "Pass",
         "InputPath": "$.payload",
         "ResultPath": "$.ifttt.value1",
         "OutputPath": "$.ifttt",
         "Next": "SaveResult"
      },
      "SaveResult": {
         "Type": "Task",
         "Resource":"frn:ifttt:webhooks:::function:save_result",
         "End": true
      }
   }
}
```

FaaS Shell REPL Executes Workflow Interactively

FUJITSU

- Haskell REPL

```
ghic> hello = ¥e -> return (e+1)
ghic> update_arg= ¥e -> return (e*2)
ghic> save_result = ¥e -> return (e^2)
ghic> pure 1 >>= hello >>= update_arg >>= save_result
16
```

FaaS Shell **startsm** corresponds Haskell pure

FaaS shell comma operator '**,**' corresponds to Haskell bind operator '**>> =**'.

- FaaS Shell REPL

```
faasshell> Hello = task('HelloWorld',"frn:wsk:functions:::function:hello",[]),
| UpdateArg = pass('UpdateArg',[result_path('$.ifttt.value1'),input_path('$.payload'),out_path('$.ifttt')]),
| SaveResult = task('SaveResult',"frn:ifttt:webhooks:::function:save_result",[]),
| startsm(_{name:"Repl"}), Hello, Pass, SaveResult.
Output=Congratulations! You've fired the save_result event
```

Devises for supporting Multiple Clouds

FUJITSU

- Function Name Resolution
  - Resource Naming (Direct Invocation)
  - Function Discovery (Indirect Invocation)

- Event State
  - Supported Event State in Workflow DSL
  - CNCF WG-Serverless Whitepaper v1.0 [1] (P.32) pointed out:

    > AWS provides “step function” primitives (state machine based primitives) for the user to
    > specify its workflow, but step function does not allow specification of what event/events
    > triggering what functions in the workflow.
  - Amazon States Language can handle Event through Activity Task, but it’s not straightforward.

[1] https://github.com/cncf/wg-serverless/blob/master/whitepaper/cncf_serverless_whitepaper_v1.0.pdf

Resource Naming - Function Name Resolution (1)

- Resource Naming in Amazon States Language Specification [1]
  - A Task State MUST include a “Resource” field, whose value MUST be a URI that uniquely identifies the specific task to execute.

Either URN or URL

- AWS Step Functions Implementation
  - ARN (Amazon Resource Name)

    "Resource":"arn:aws:lambda:us-east-2:410388484666:function:hello“

- FaaS Shell Implementation
  - FRN (FaaS Resource Name)

AWS	“ Resource”:“frn:aws: …“ or “arn:aws: …“
Microsoft	“ Resource”:“frn:azure: …“
Google	“ Resource”:“frn:gcp: …“
IBM/OpenWhisk	“Resource”:“frn:wsk: … “
IFTTT	“ Resource”:“frn:ifttt: …“
Fujitsu	“ Resource”:“frn:k5: …“

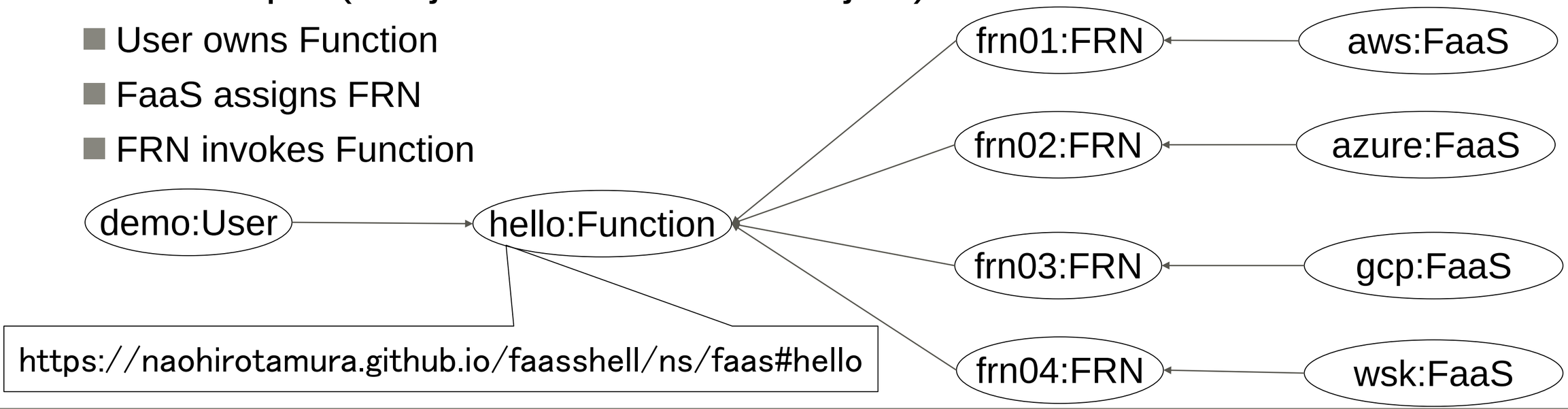
[1] https://states-language.net/spec.html

Function Discovery - Function Name Resolution (2)

- Discover Function Semantically
  - Make use of Open Linked Data technology, OWL (Web Ontology Language)
- FaaS Shell Implementation
  - "Resource": "https://naohirotamura.github.io/faasshell/ns/faas#hello"

URI: Unique Identifier

- Define Triple (Subject – Predicate – Object)
  - User owns Function
  - FaaS assigns FRN
  - FRN invokes Function

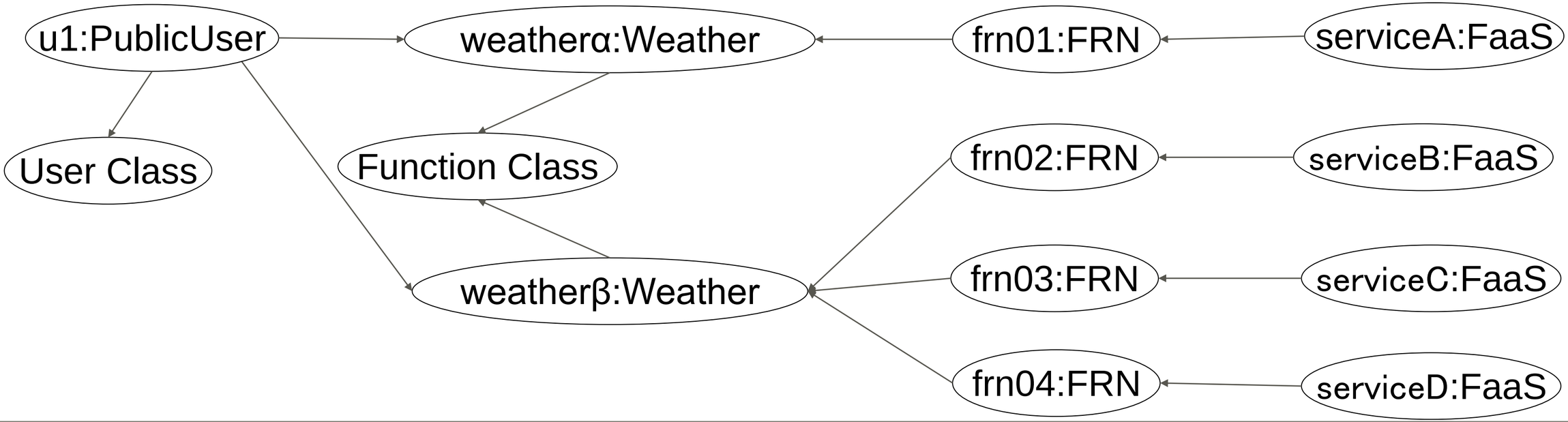


Function Discovery Cont. - Function Name Resolution (2)

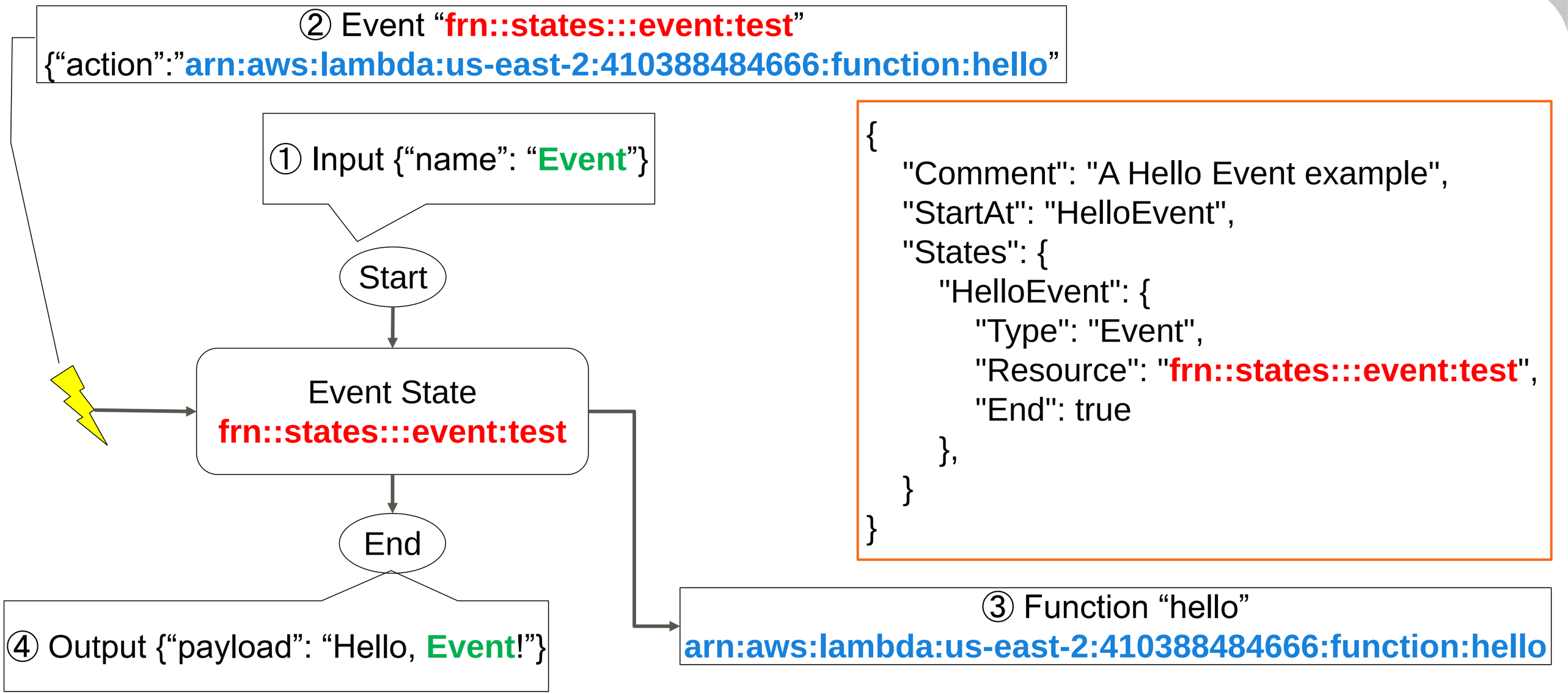
- There are several weather service on Internet
  - Assume reusable function “weather” open linked RDF data or SPARQL end point is available
  - "Resource": "https://weather.org/ns/faas#weatherβ"

URI is just Identifier

- Querying the linked data, we will figured out that “frn02”,”frn03”,”frn04” are callable using PublicUser “u1” account.



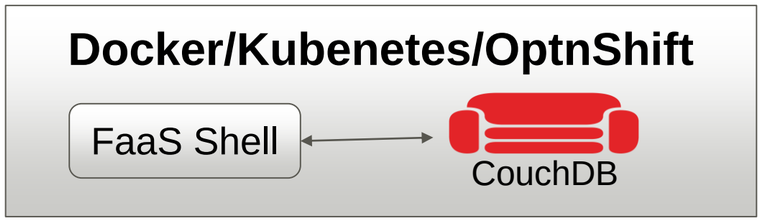
Event State




```
{
    "Comment": "A Hello Event example",
    "StartAt": "HelloEvent",
    "States": {
        "HelloEvent": {
            "Type": "Event",
            "Resource": "frn::states:::event:test",
            "End": true
        },
    }
}
```

FaaS Shell Deployment

① Development



② Horizontal Scale Out