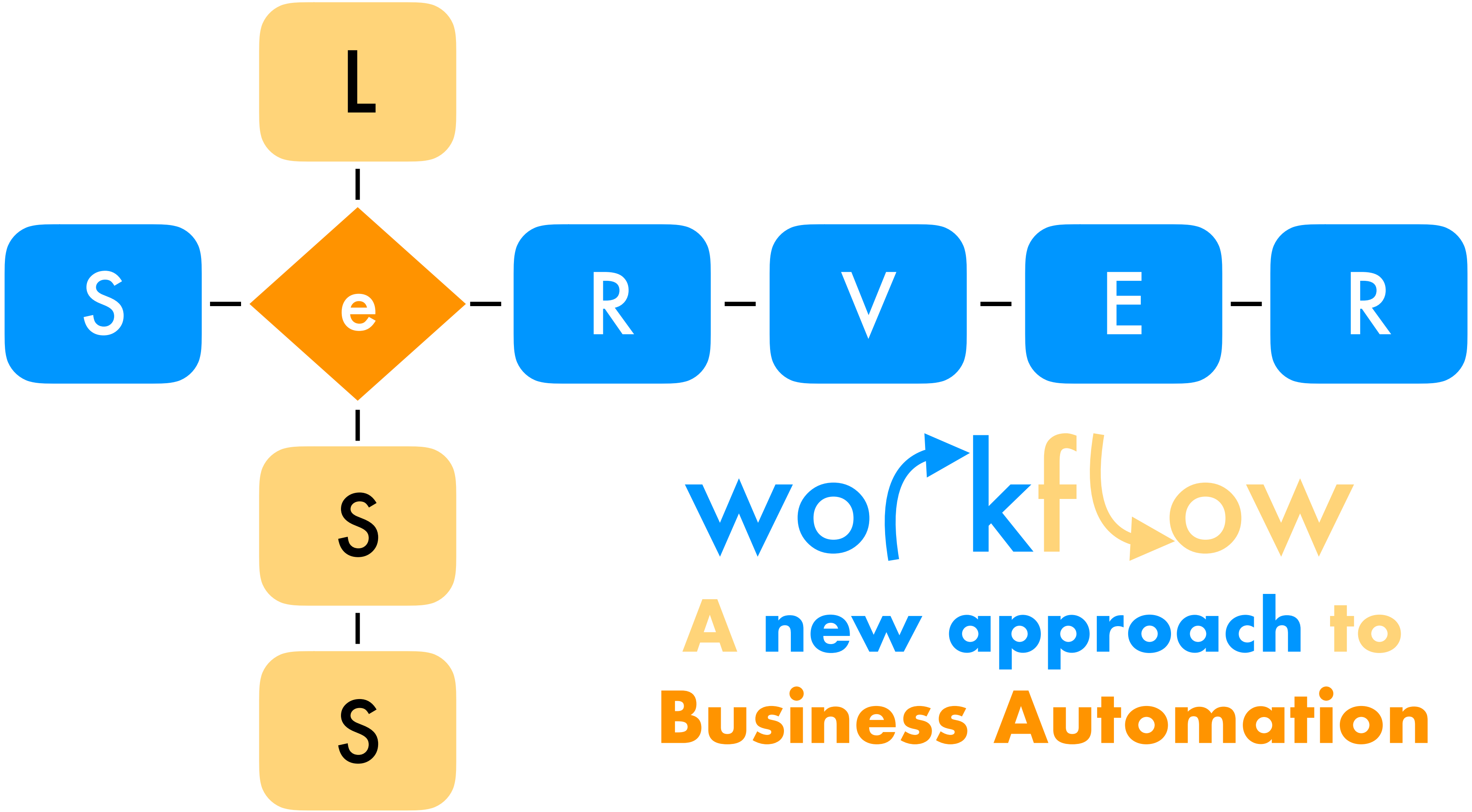


A new approach to Business Automation

Tihomir Surdilovic
Maciej Swiderski

Red Hat

Red Hat Microservices Day, Atlanta

Introduction - Serverless

- Managed platforms that run our code (event-triggered functions)
- Serverless platform characteristics:

Managed Servers

Automatic Scaling

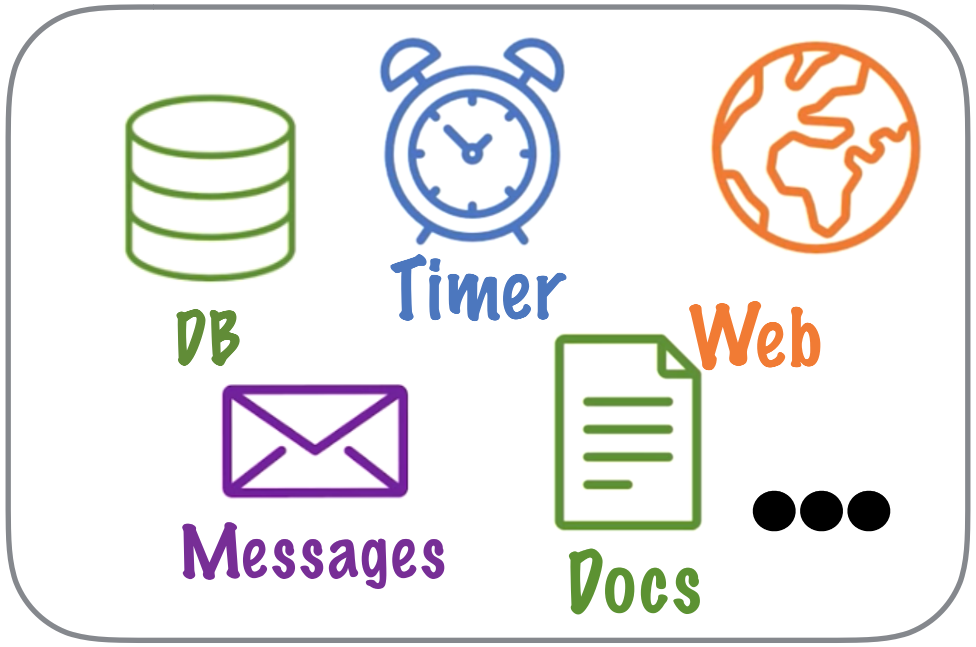
Pay for value

Basically, serverless programming allows devs to focus on code and application development, not the infrastructure.

Introduction - Serverless Functions

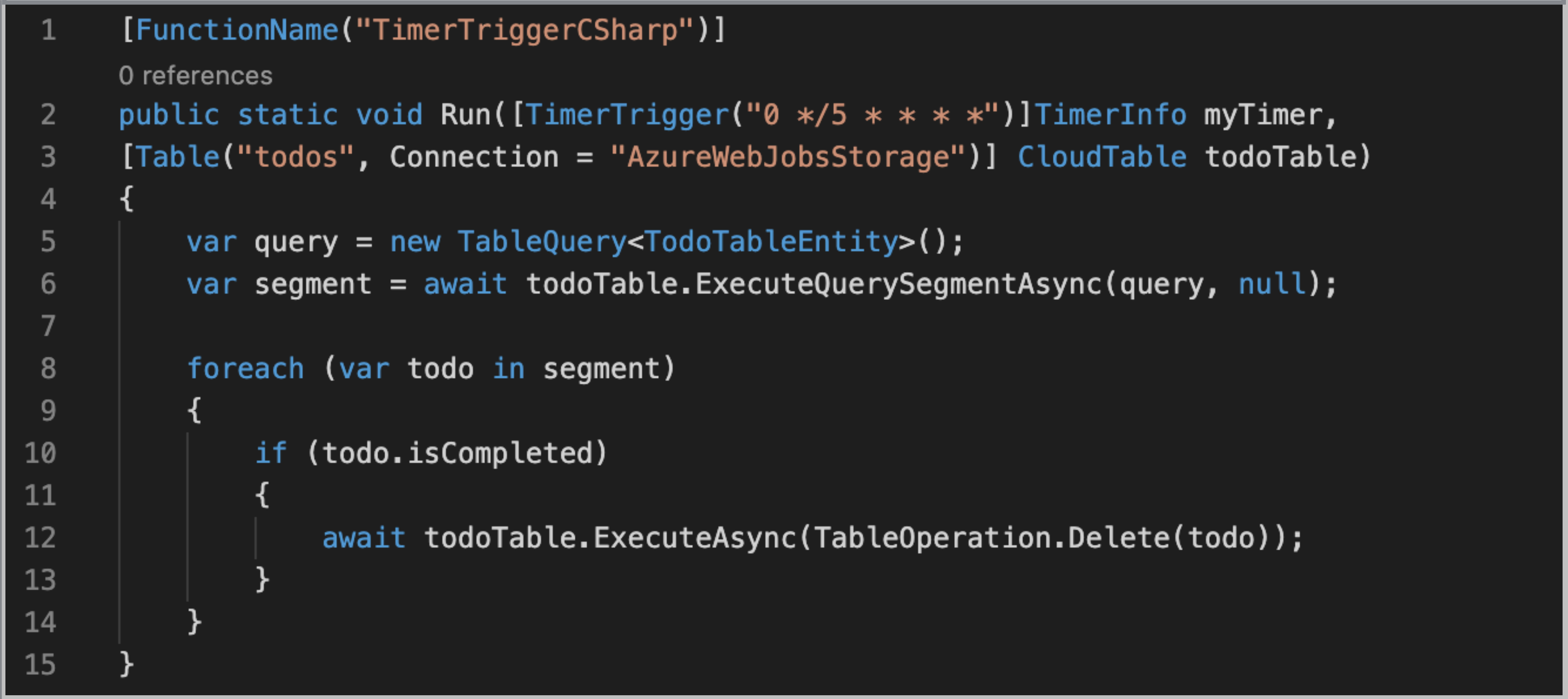
- Event-driven (consume / produce)
- Loosely-coupled (Microservices)
- Polyglot
- Main Focus:
  solve business requirements
  or be part/step of solution

Events



Introduction - Serverless Functions

* Are we actually solving business requirements?

business req	implementation
When a todo item is completed, remove it.	

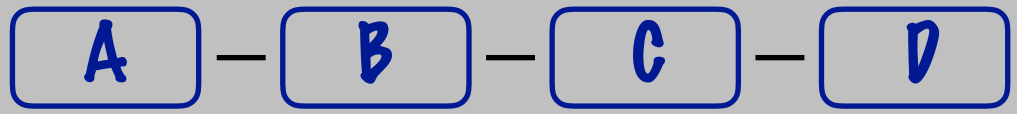
```
[FunctionName("TimerTriggerCSharp")]
0 references
public static void Run([TimerTrigger("0 */5 * * * *")]TimerInfo myTimer,
[Table("todos", Connection = "AzureWebJobsStorage")] CloudTable todoTable)
{
    var query = new TableQuery<TodoTableEntity>();
    var segment = await todoTable.ExecuteQuerySegmentAsync(query, null);

    foreach (var todo in segment)
    {
        if (todo.isCompleted)
        {
            await todoTable.ExecuteAsync(TableOperation.Delete(todo));
        }
    }
}
```

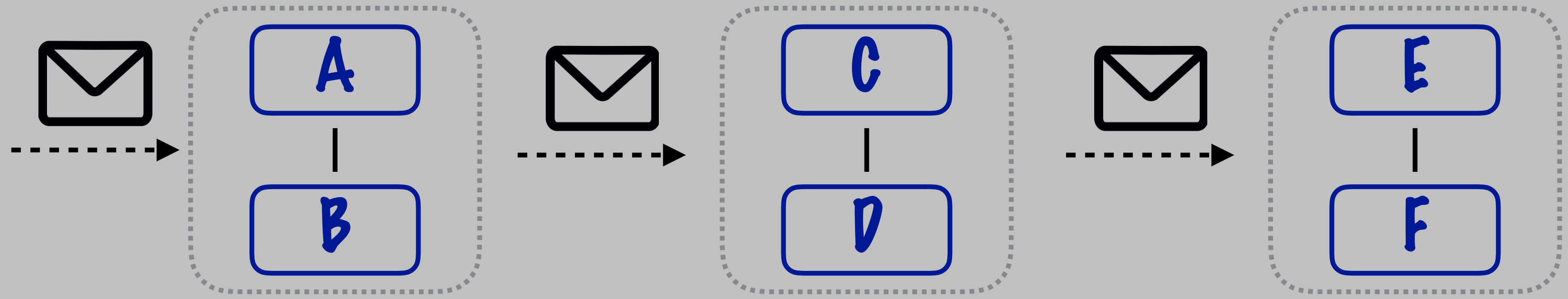
Focus on business requirements...
...Orchestrate everything else

Evolving towards serverless workflows

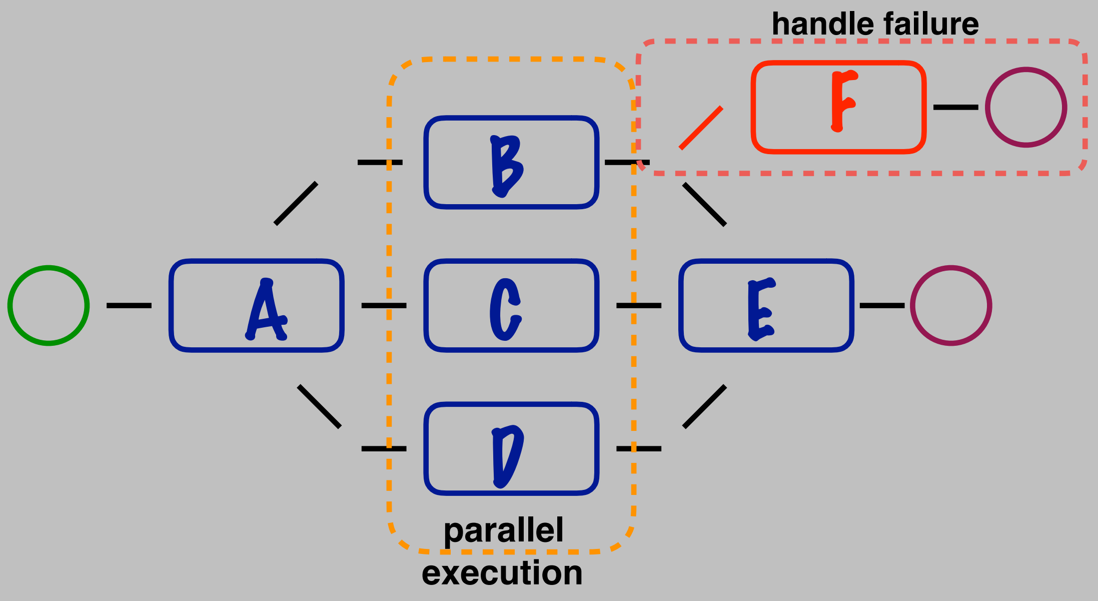
* Sequential Functions:



* Queues:

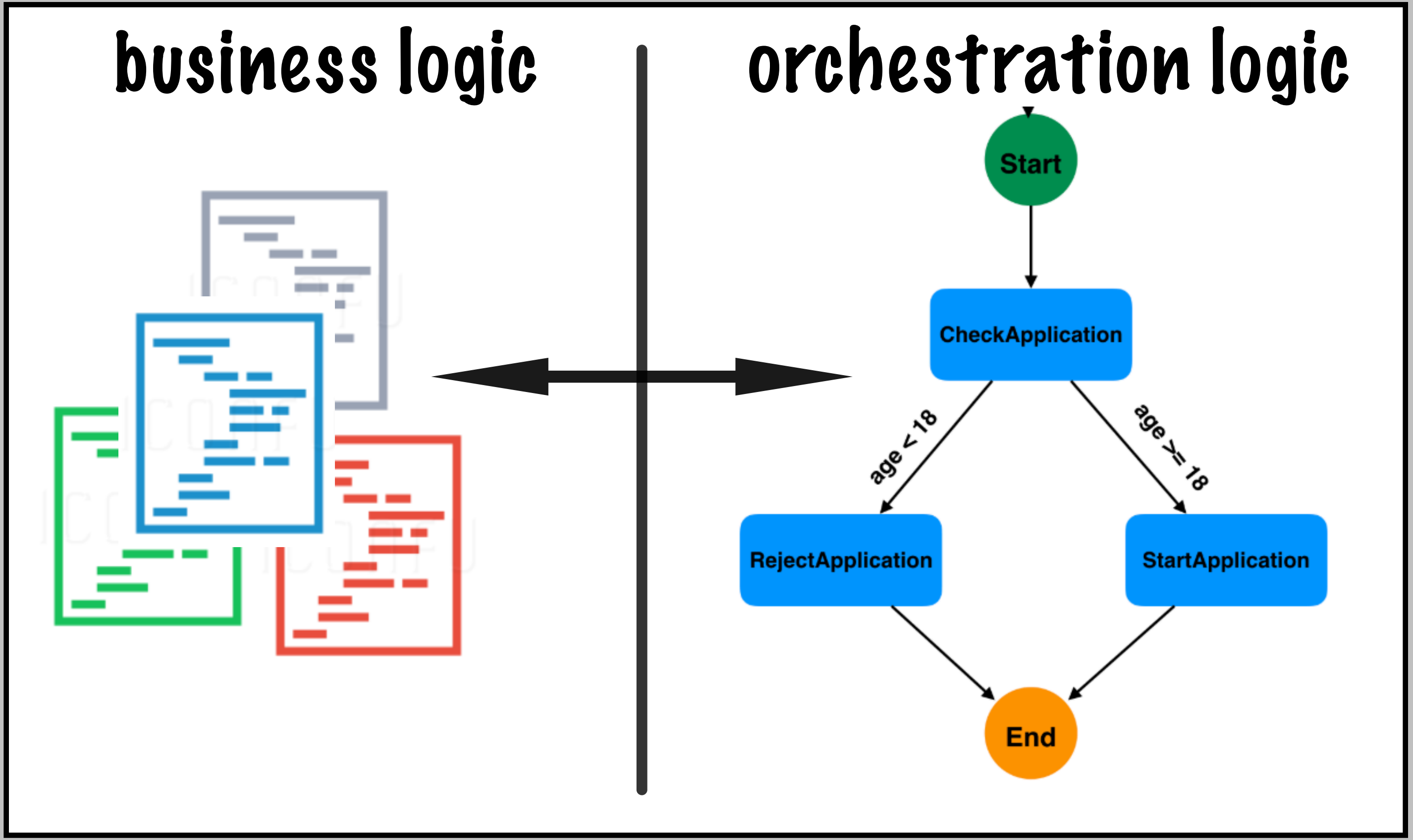


* Workflows:



Benefits of Serverless Workflows:

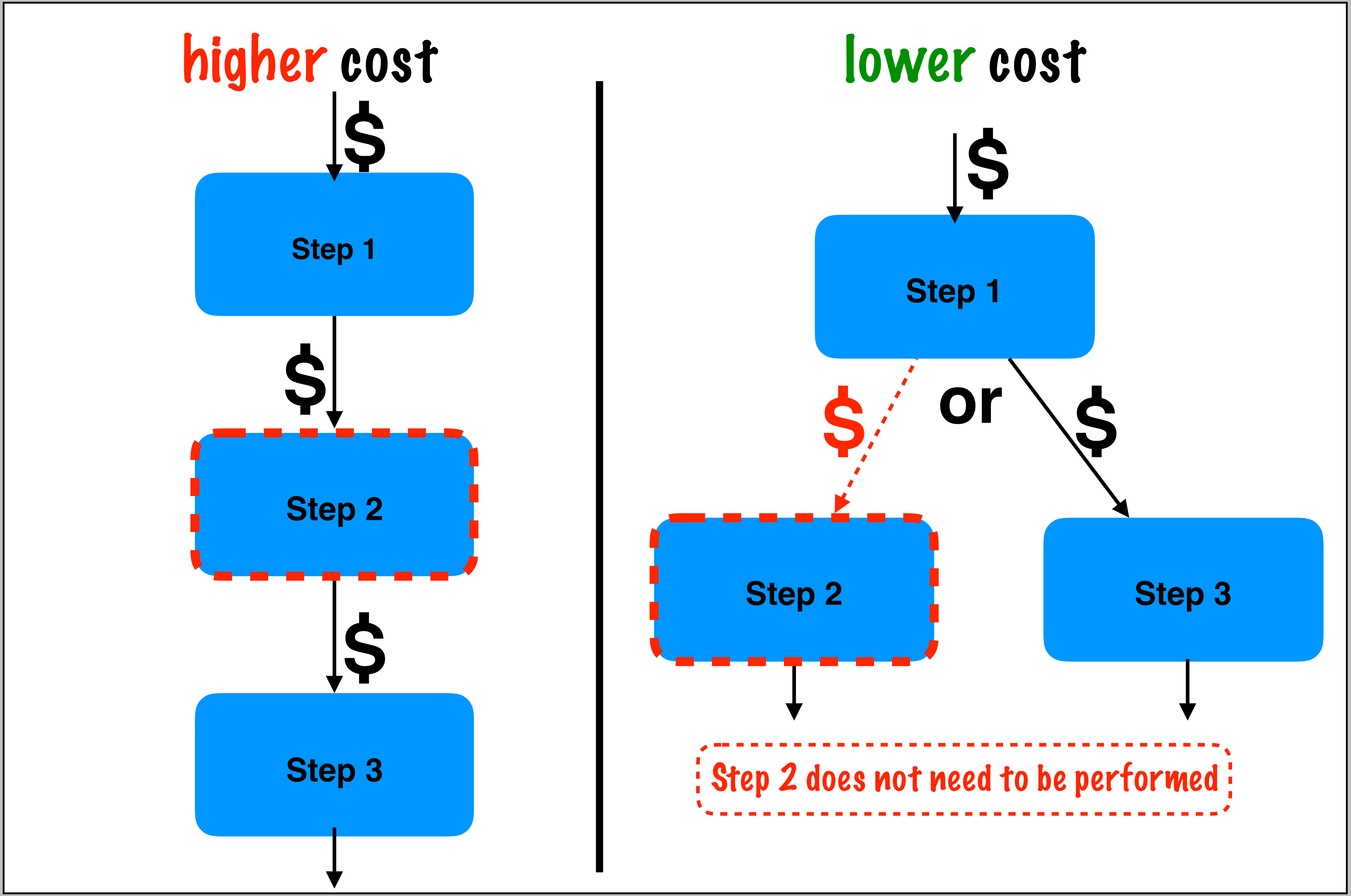
- Clear separation of concerns:



- Many more:
  - Visualization
  - Quick turnaround and maintenance
  - State/data management (between functions)
  - Flow control patterns!!
  - Human readable definition
  - Extensible and embeddable

Serverless Workflows: Cost

- Will not necessarily reduce cost!
- Have to think about workflow and function cost
- Workflow cost typically based on transitions



Step 2 does not need to be performed

Current state of Serverless Workflow Implementations

AWS

Microsoft Azure
Logic Apps

fission
Workflows

conductor
Netflix

Apache Airflow

StackStorm
Orquesta

fn
Fn Flow

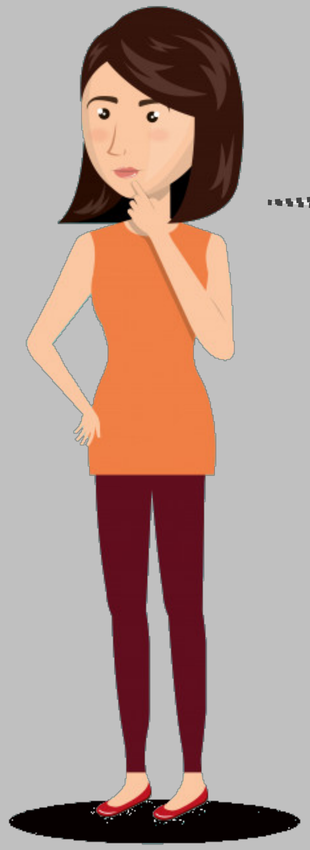
HUAWEI
FunctionGraph

Alibaba Cloud
Flow

Kogito

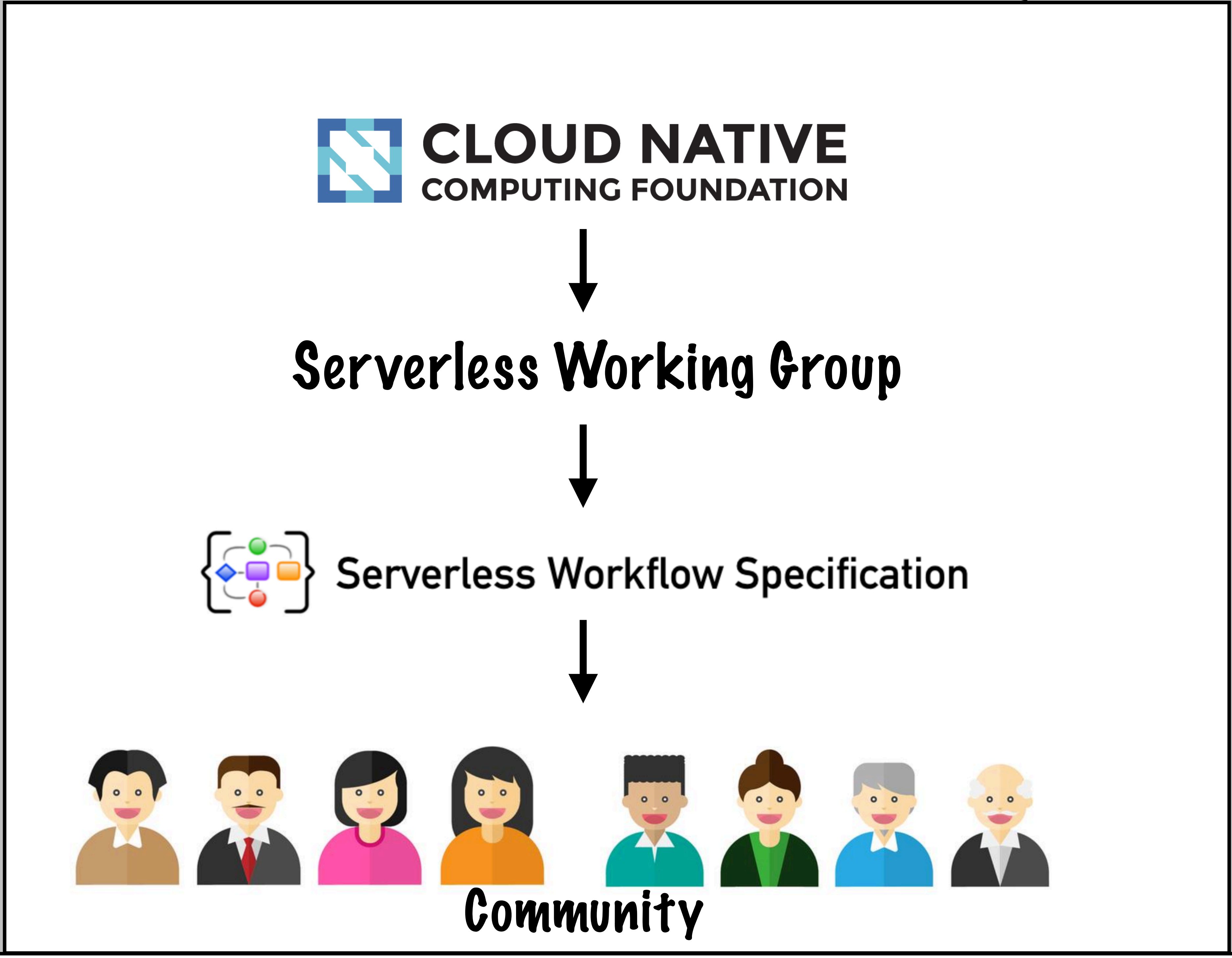
Current state of Serverless Workflow Implementations

- Proprietary and different workflow models and definitions
- Not portable / vendor neutral
- Definition not covering stateless and stateful orchestration
- Integration with proprietary services / events / etc
- Different workflow notations



... sounds to me like
we need a
portable and
vendor-neutral
specification for
Serverless Workflows!

CNCF - Serverless WG - Serverless Workflow Spec.



CLOUD NATIVE COMPUTING FOUNDATION

↓

Serverless Working Group

↓

Serverless Workflow Specification

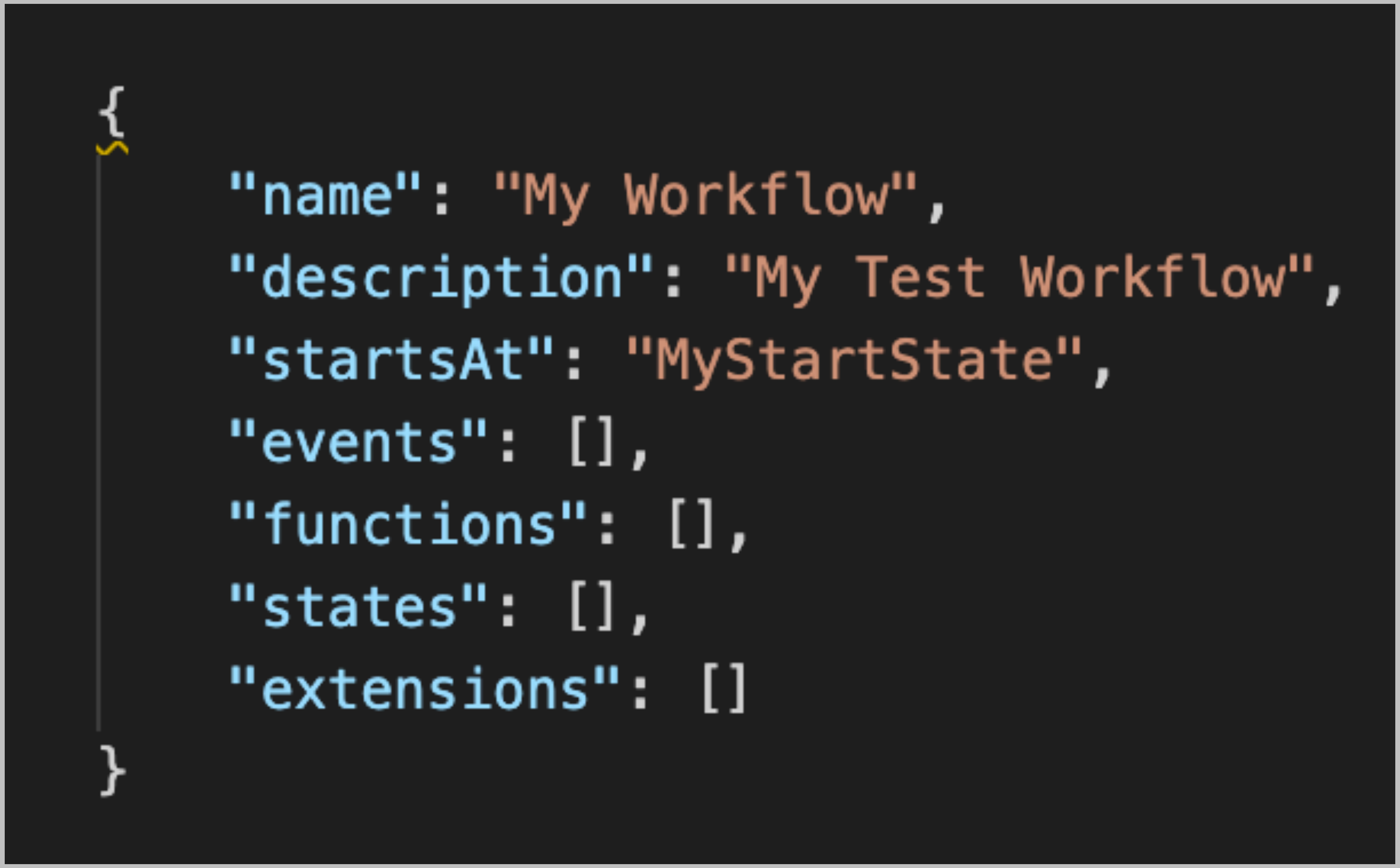
↓

Community

https://github.com/cncf/wg-serverless/tree/master/workflow/spec

Serverless Workflow Specification

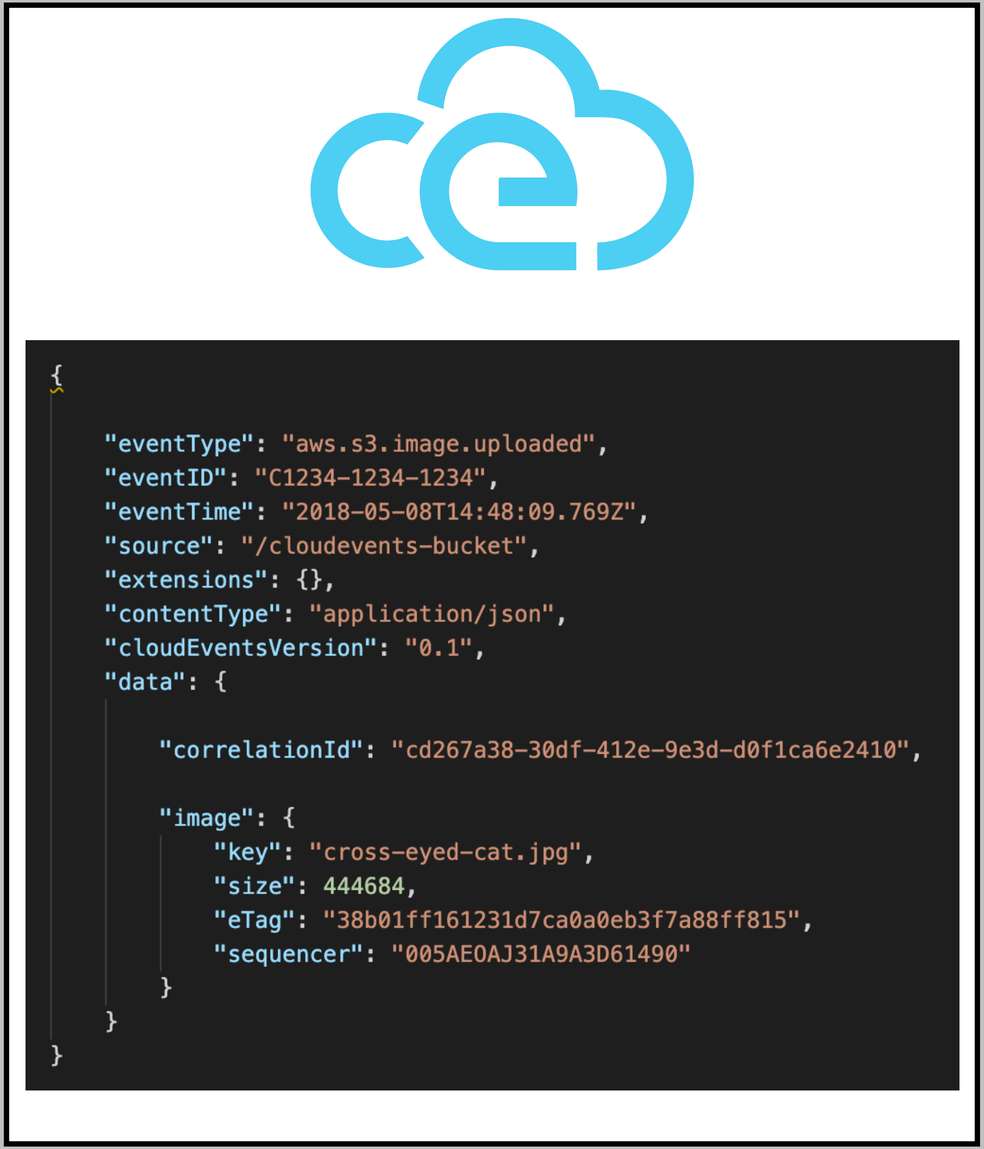
- Lightweight + Embeddable (JSON and YAML)
- Human-readable:



```json
{
    "name": "My Workflow",
    "description": "My Test Workflow",
    "startsAt": "MyStartState",
    "events": [],
    "functions": [],
    "states": [],
    "extensions": []
}
```

Serverless Workflow - Event Triggers

- Define **incoming events** that trigger **Actions**
- Events defined via **CloudEvents** specification
- Control flow **decisions** based on event data
- **Correlate** incoming events to same workflows instances

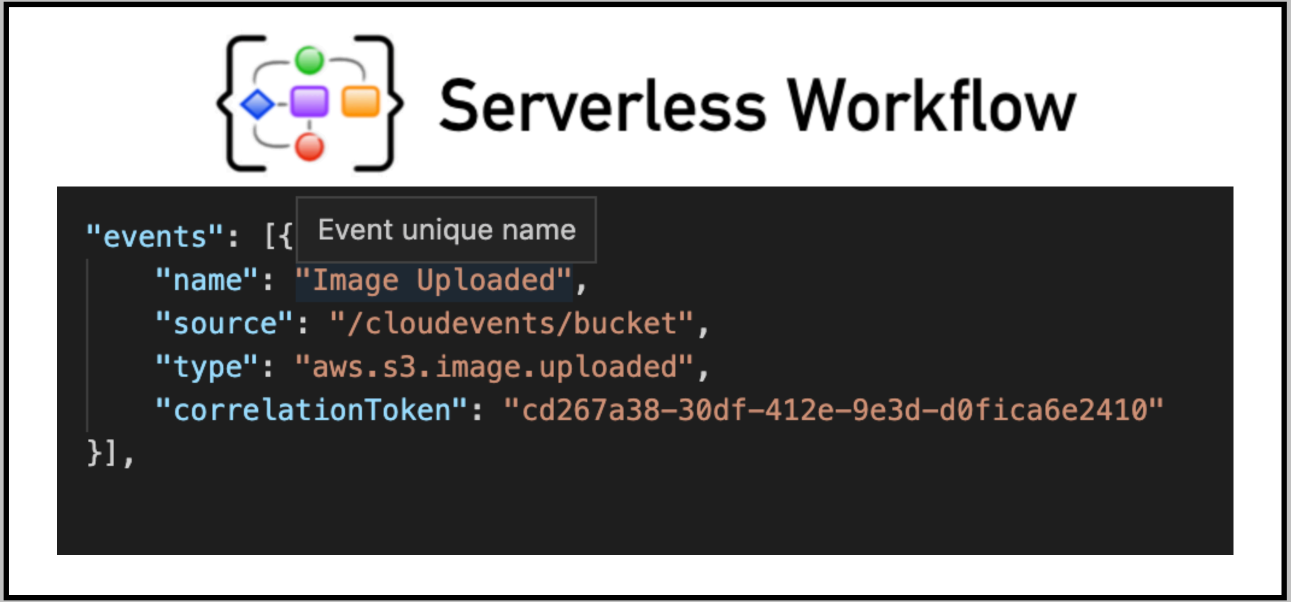


```
{

    "eventType": "aws.s3.image.uploaded",
    "eventID": "C1234-1234-1234",
    "eventTime": "2018-05-08T14:48:09.769Z",
    "source": "/cloudevents-bucket",
    "extensions": {},
    "contentType": "application/json",
    "cloudEventsVersion": "0.1",
    "data": {

        "correlationId": "cd267a38-30df-412e-9e3d-d0f1ca6e2410",

        "image": {
            "key": "cross-eyed-cat.jpg",
            "size": 444684,
            "eTag": "38b01ff161231d7ca0a0eb3f7a88ff815",
            "sequencer": "005AEOAJ31A9A3D61490"
        }
    }
}
```

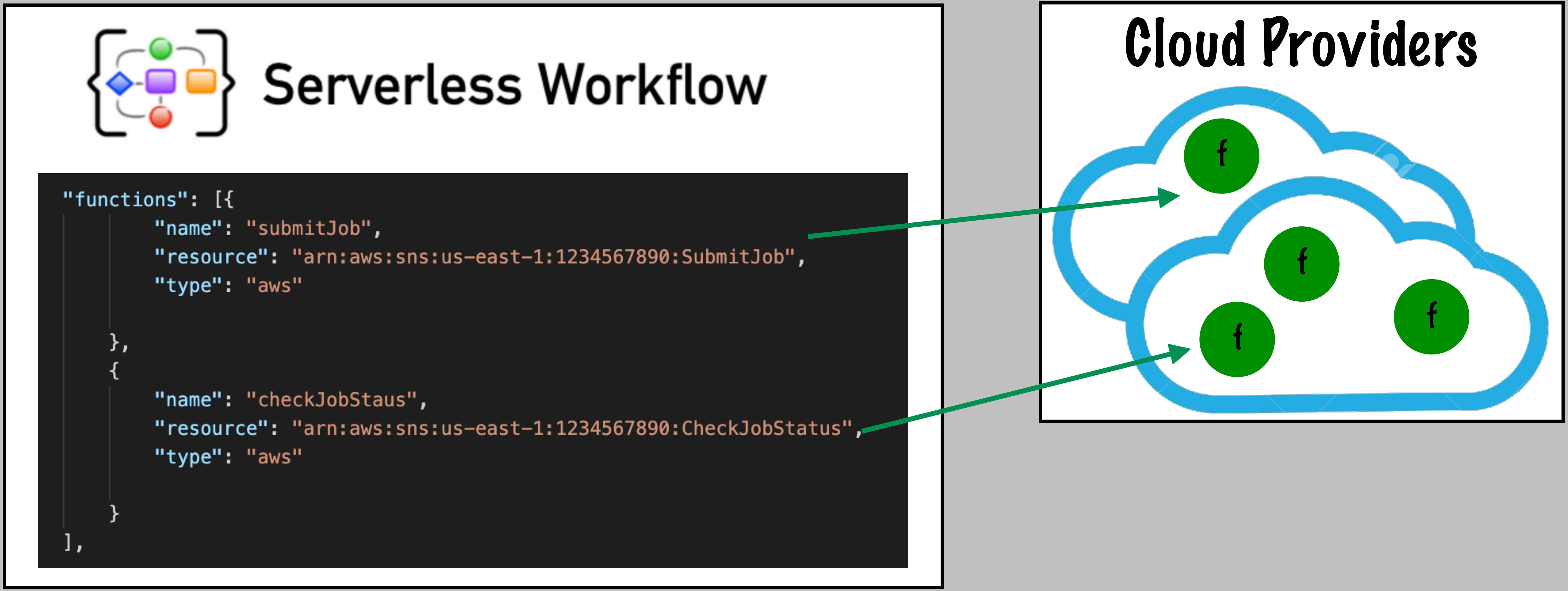


Event unique name

```
"events": [{
    "name": "Image Uploaded",
    "source": "/cloudevents/bucket",
    "type": "aws.s3.image.uploaded",
    "correlationToken": "cd267a38-30df-412e-9e3d-d0fica6e2410"
}],
```

Serverless Workflow - Serverless Functions

- Reusable serverless functions definitions
- Consumed by workflow Actions
- Actions can pass parameters to functions



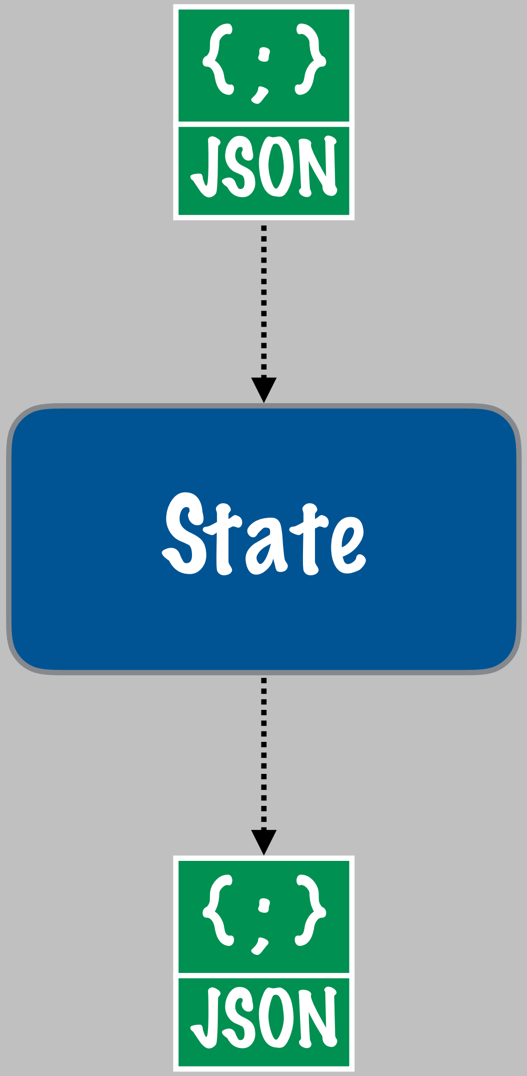
```
"functions": [{
        "name": "submitJob",
        "resource": "arn:aws:sns:us-east-1:1234567890:SubmitJob",
        "type": "aws"

    },
    {
        "name": "checkJobStaus",
        "resource": "arn:aws:sns:us-east-1:1234567890:CheckJobStatus",
        "type": "aws"

    }
],
```

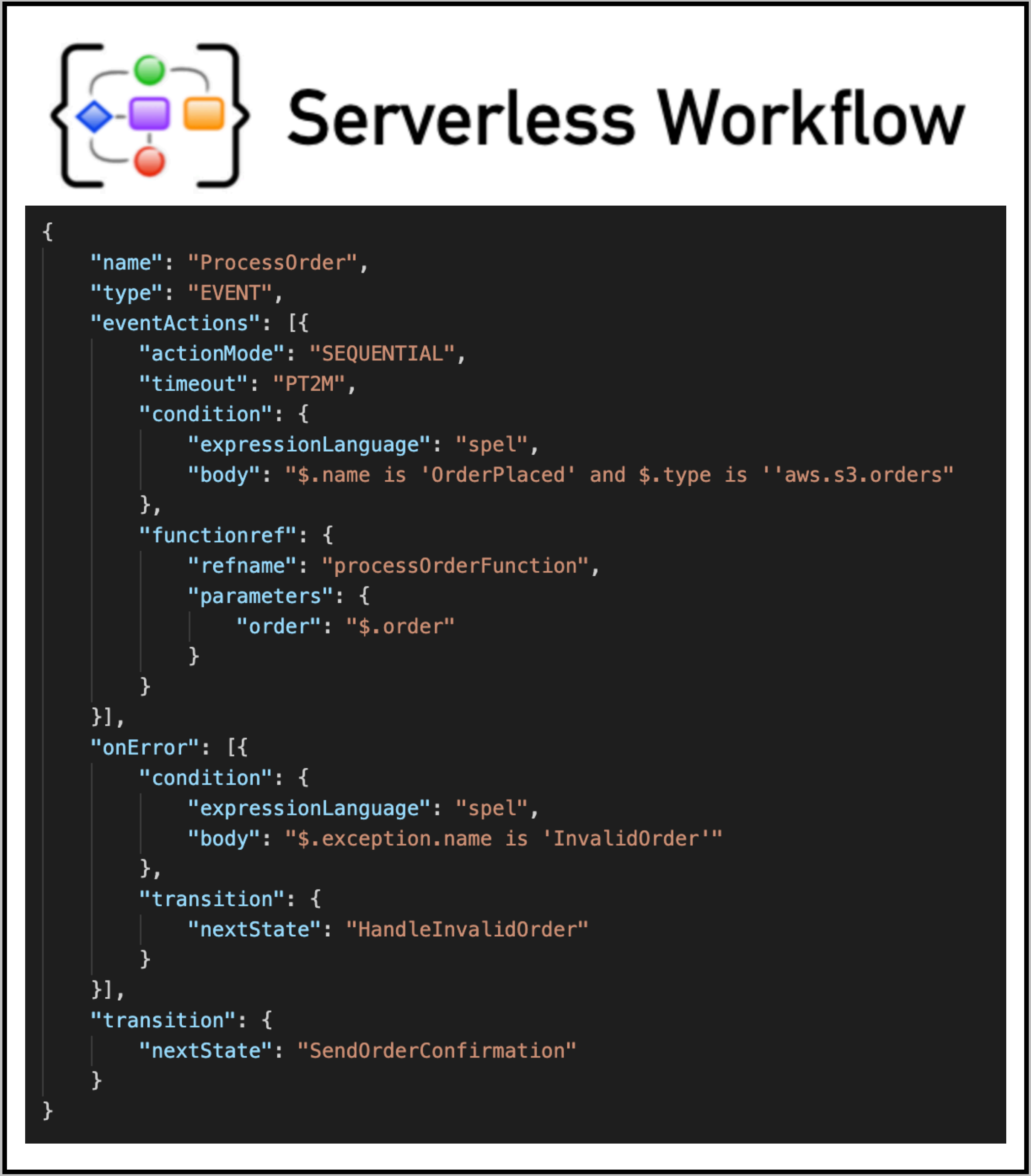
Serverless Workflow - States

- Building blocks in workflows
- Define application control flow
- Can perform Actions
- Make decisions based on data (JSON)
- Manage cross-cutting concerns (error handling, retries, timeouts, etc)
- Filters data (JSONPath)



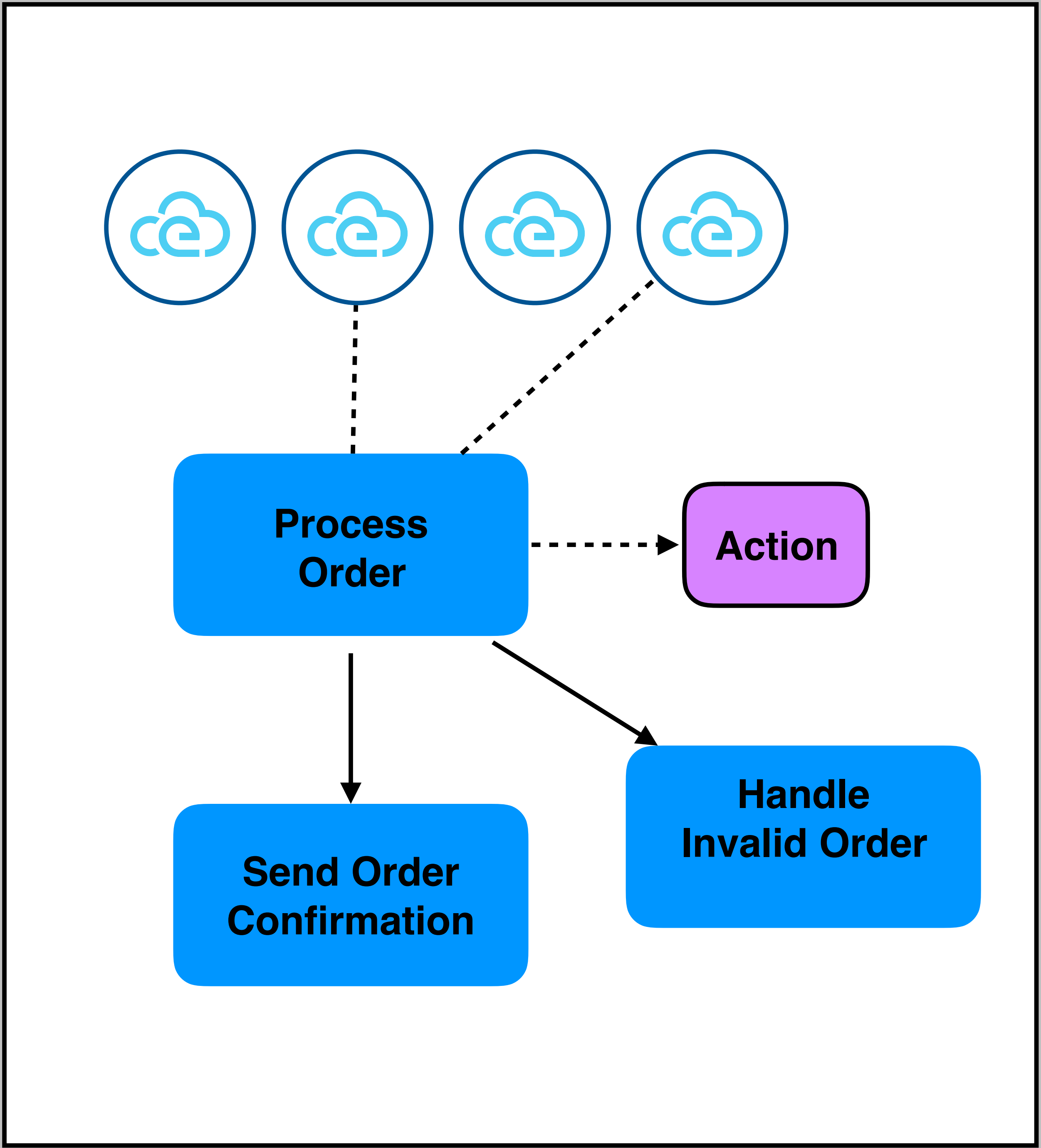
Workflow - Event State

- **Triggered** by events from event sources
- Perform one or more **Actions** (if defined events are exist)



Serverless Workflow

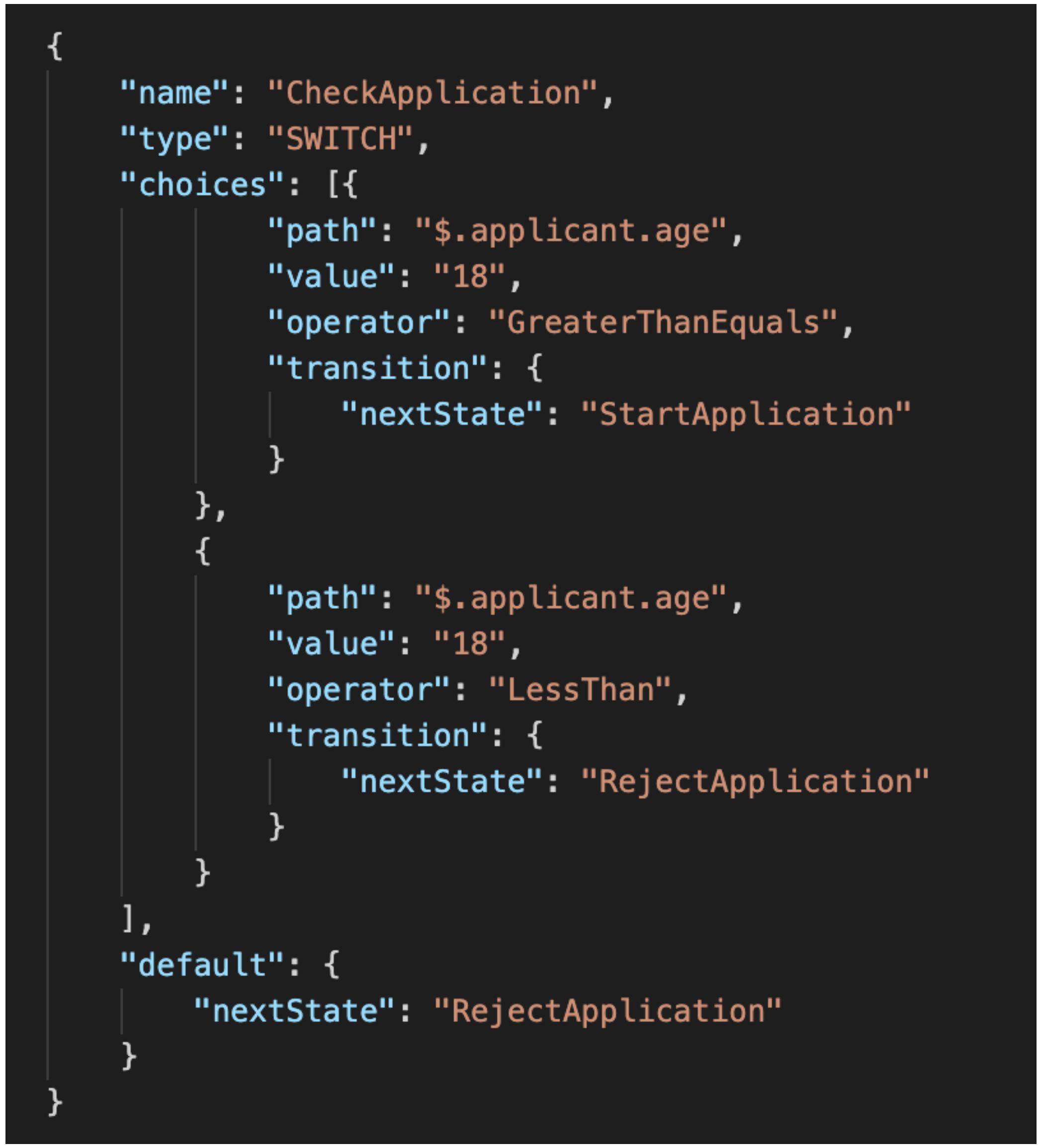
```
{
    "name": "ProcessOrder",
    "type": "EVENT",
    "eventActions": [{
        "actionMode": "SEQUENTIAL",
        "timeout": "PT2M",
        "condition": {
            "expressionLanguage": "spel",
            "body": "$.name is 'OrderPlaced' and $.type is ''aws.s3.orders"
        },
        "functionref": {
            "refname": "processOrderFunction",
            "parameters": {
                "order": "$.order"
            }
        }
    }],
    "onError": [{
        "condition": {
            "expressionLanguage": "spel",
            "body": "$.exception.name is 'InvalidOrder'"
        },
        "transition": {
            "nextState": "HandleInvalidOrder"
        }
    }],
    "transition": {
        "nextState": "SendOrderConfirmation"
    }
}
```



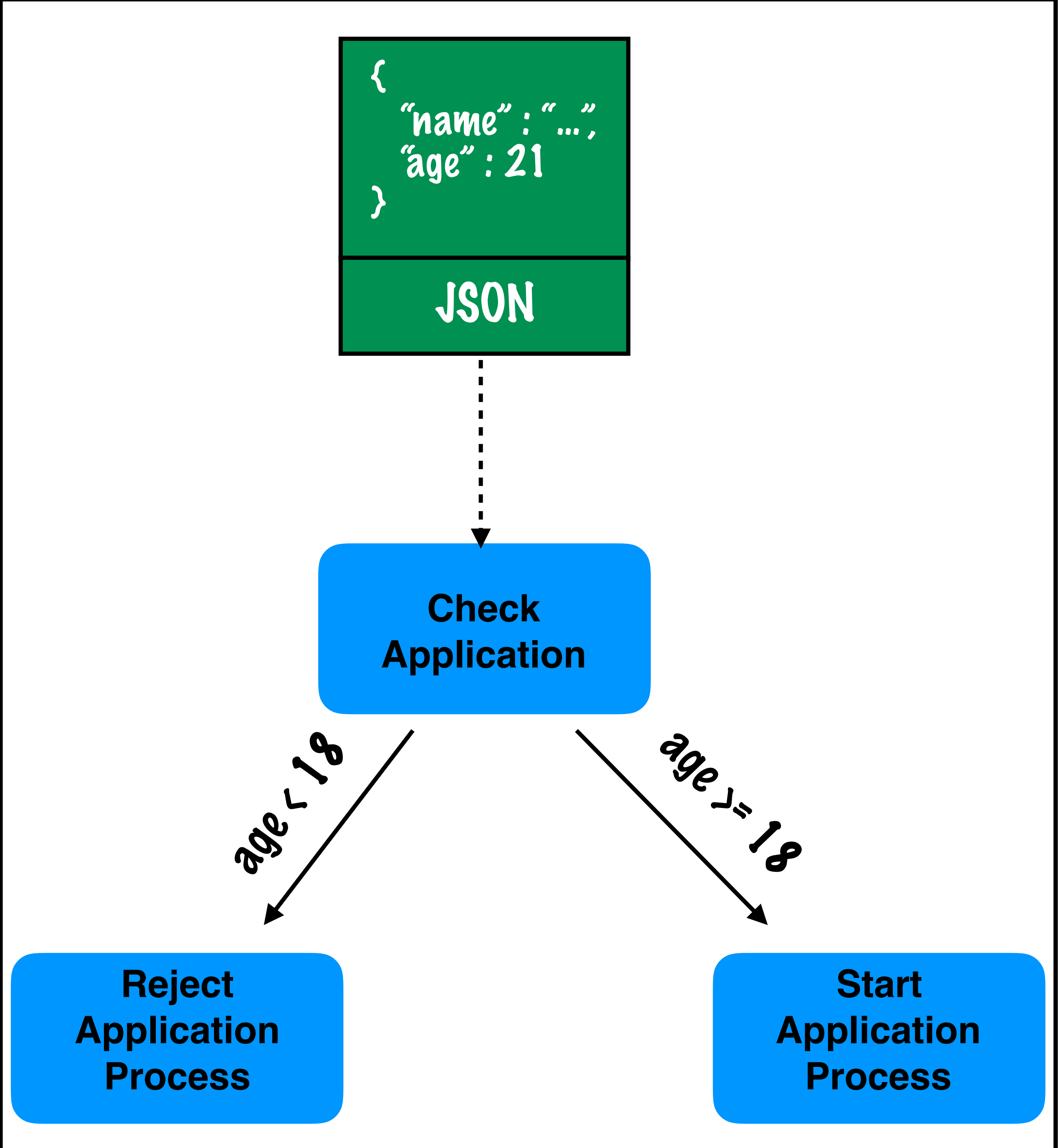
Workflow - Switch State

- Workflow gateways
- Define transitions based on data input

Serverless Workflow



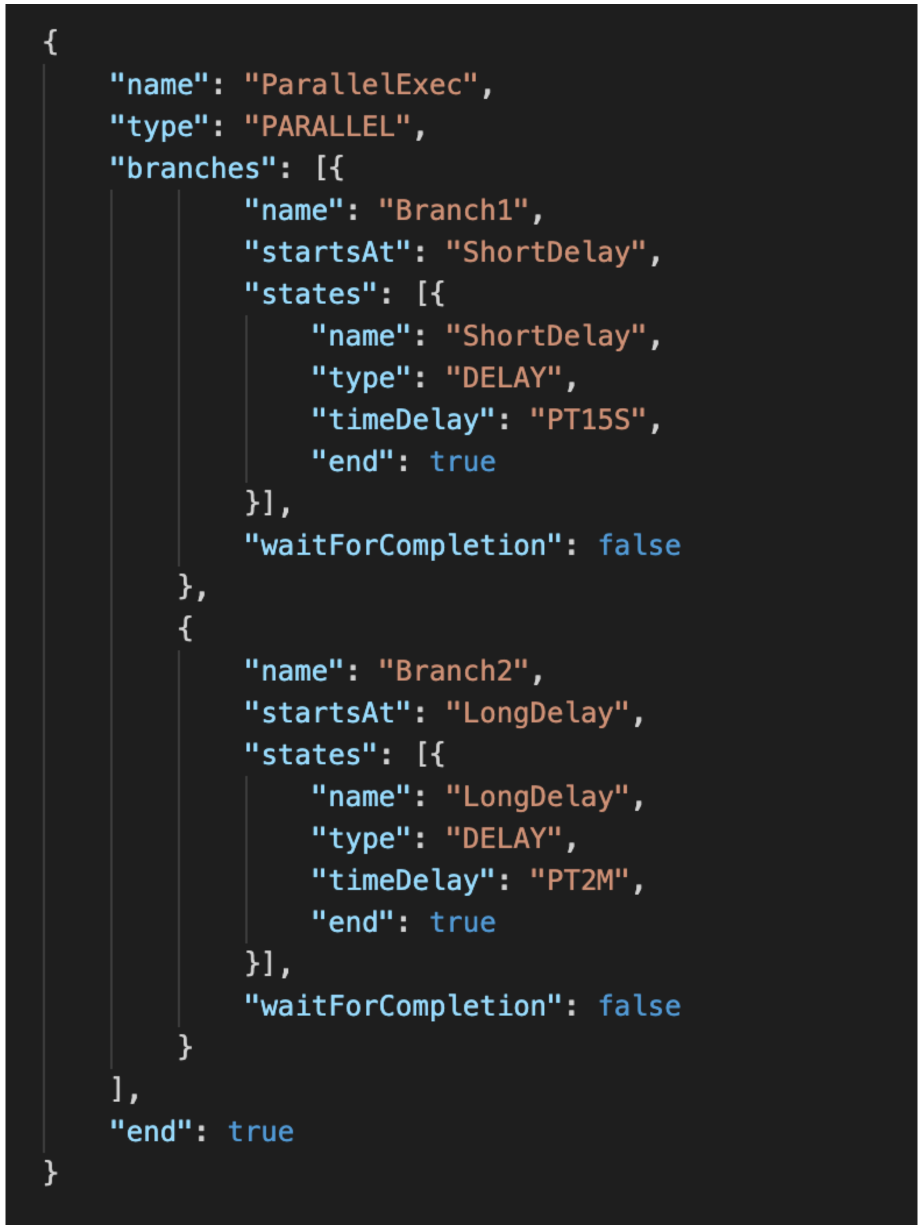
```
{
    "name": "CheckApplication",
    "type": "SWITCH",
    "choices": [{
            "path": "$.applicant.age",
            "value": "18",
            "operator": "GreaterThanEquals",
            "transition": {
                "nextState": "StartApplication"
            }
        },
        {
            "path": "$.applicant.age",
            "value": "18",
            "operator": "LessThan",
            "transition": {
                "nextState": "RejectApplication"
            }
        }
    ],
    "default": {
        "nextState": "RejectApplication"
    }
}
```



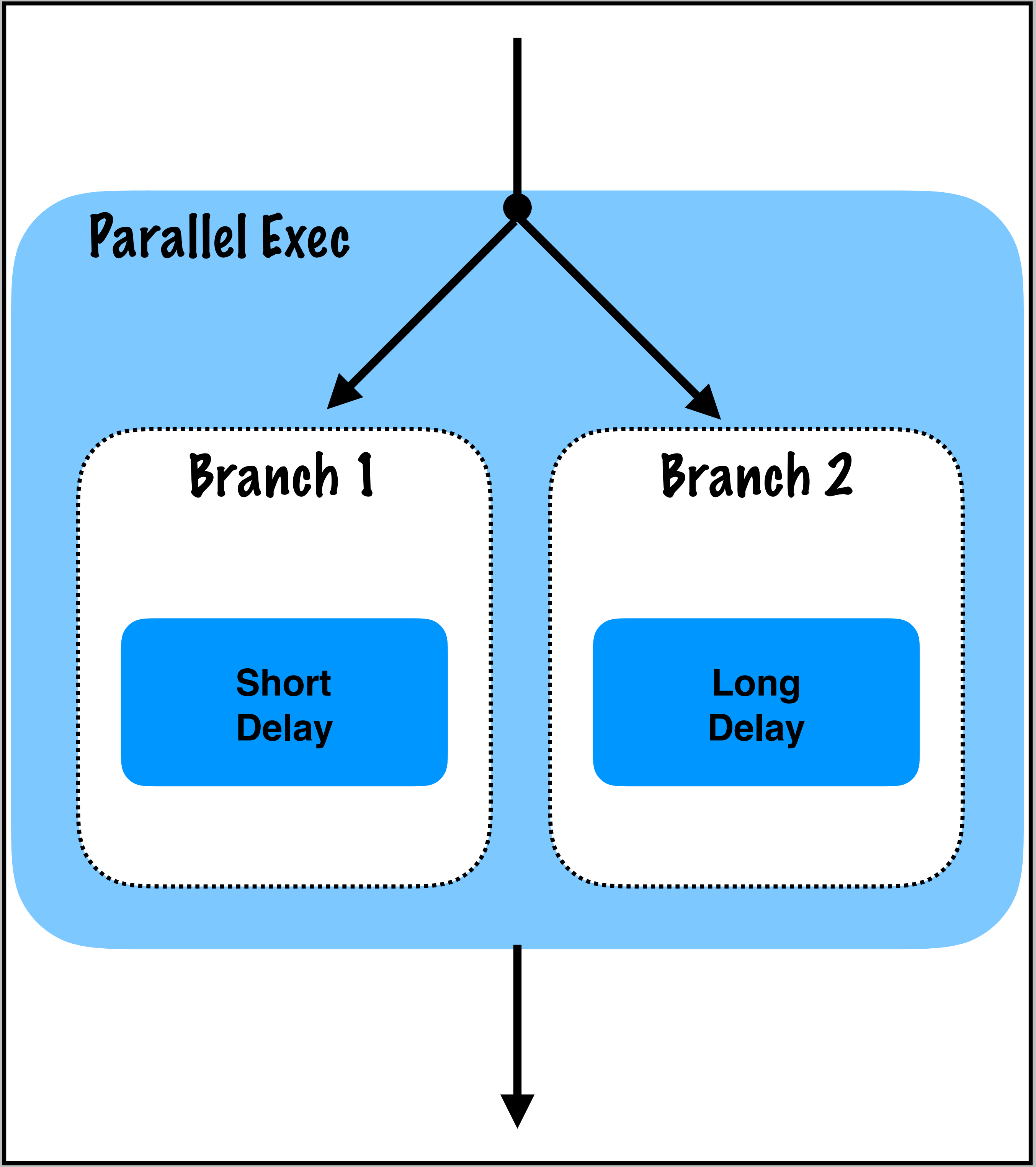
Workflow - Parallel State

- Define collection of branches to execute in parallel
- Branches contain one or more states

Serverless Workflow



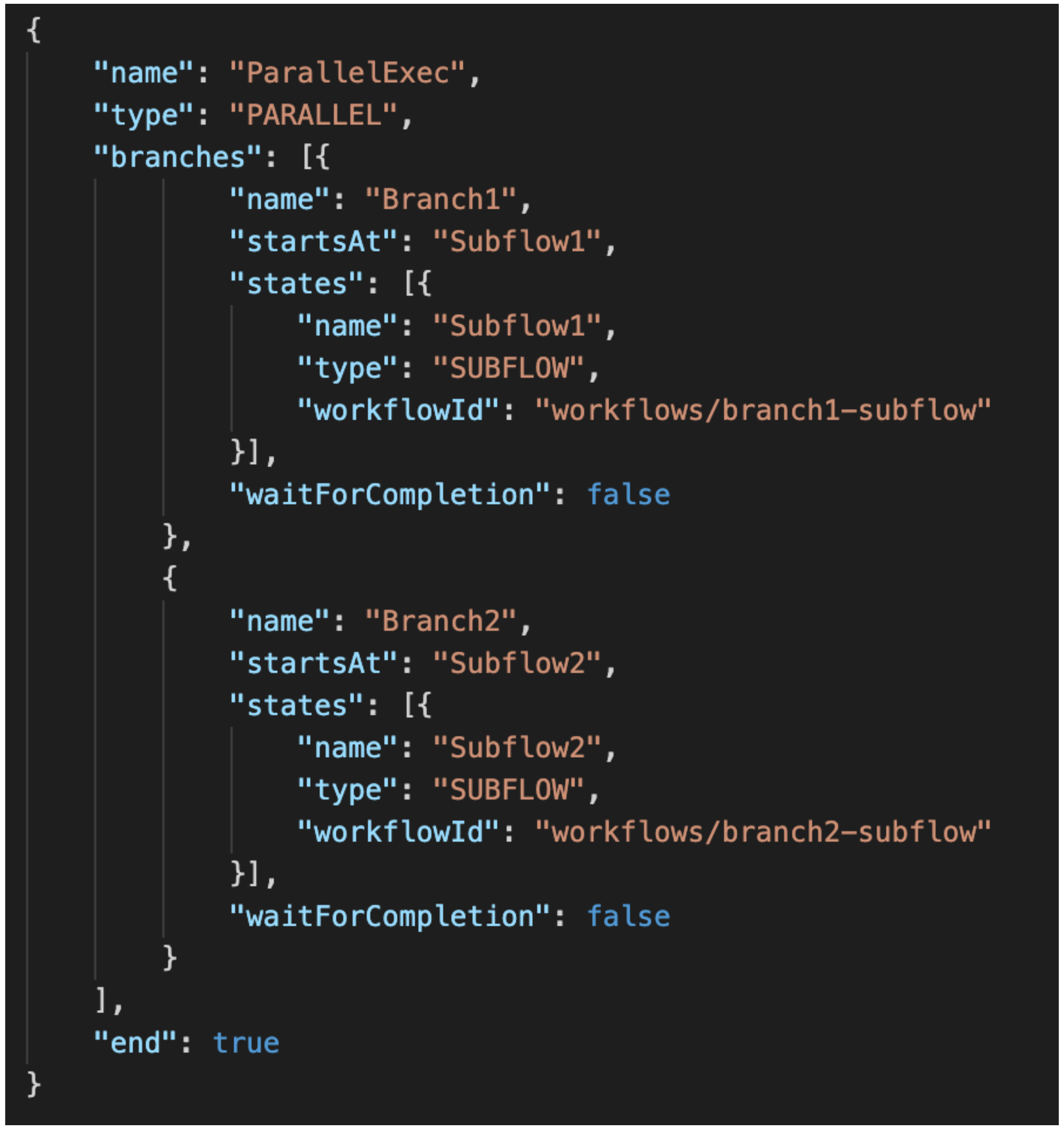
```
{
    "name": "ParallelExec",
    "type": "PARALLEL",
    "branches": [{
            "name": "Branch1",
            "startsAt": "ShortDelay",
            "states": [{
                "name": "ShortDelay",
                "type": "DELAY",
                "timeDelay": "PT15S",
                "end": true
            }],
            "waitForCompletion": false
        },
        {
            "name": "Branch2",
            "startsAt": "LongDelay",
            "states": [{
                "name": "LongDelay",
                "type": "DELAY",
                "timeDelay": "PT2M",
                "end": true
            }],
            "waitForCompletion": false
        }
    ],
    "end": true
}
```



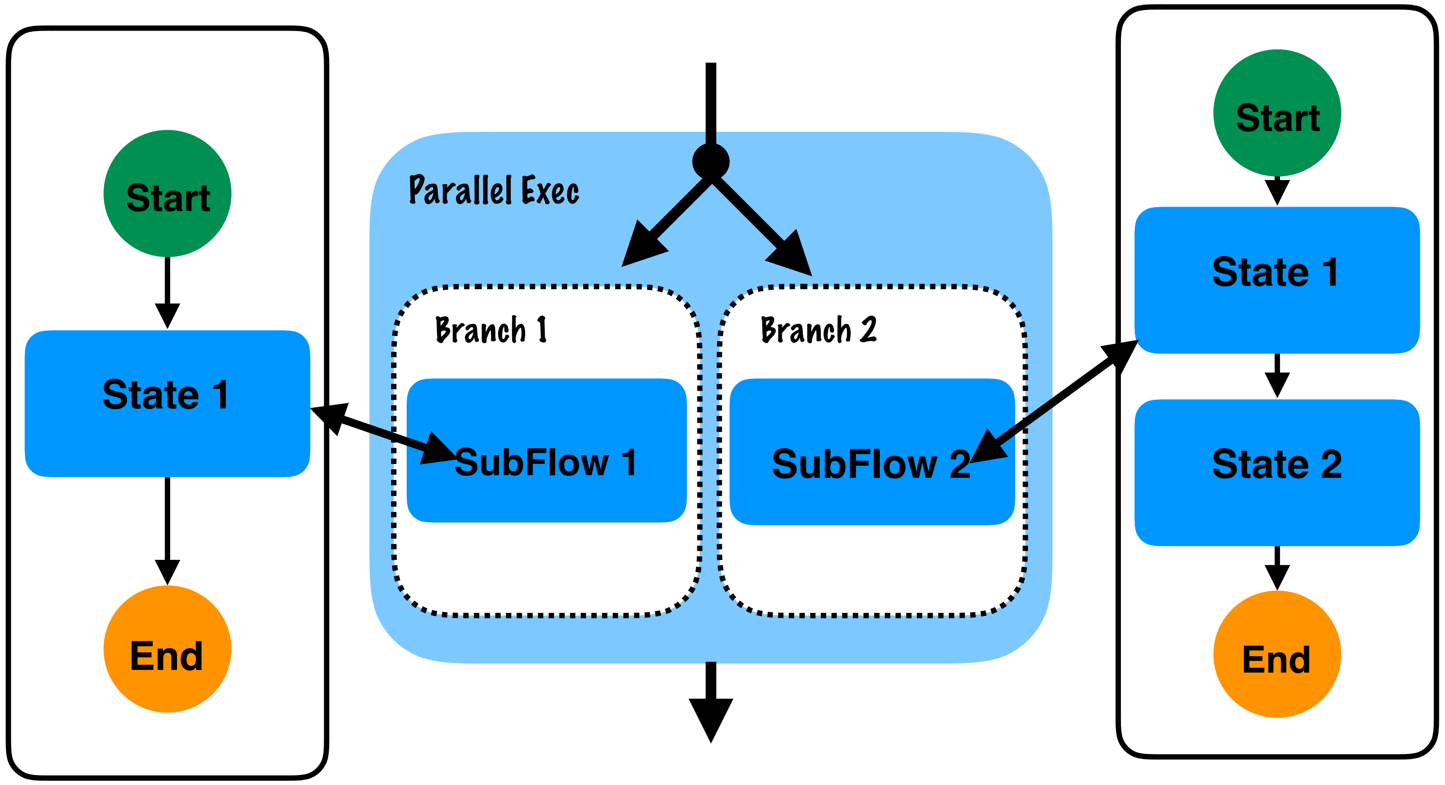
Workflow - SubFlow State

- Embed a sub-workflow and start its execution
- Promotes reusability and logical grouping

Serverless Workflow



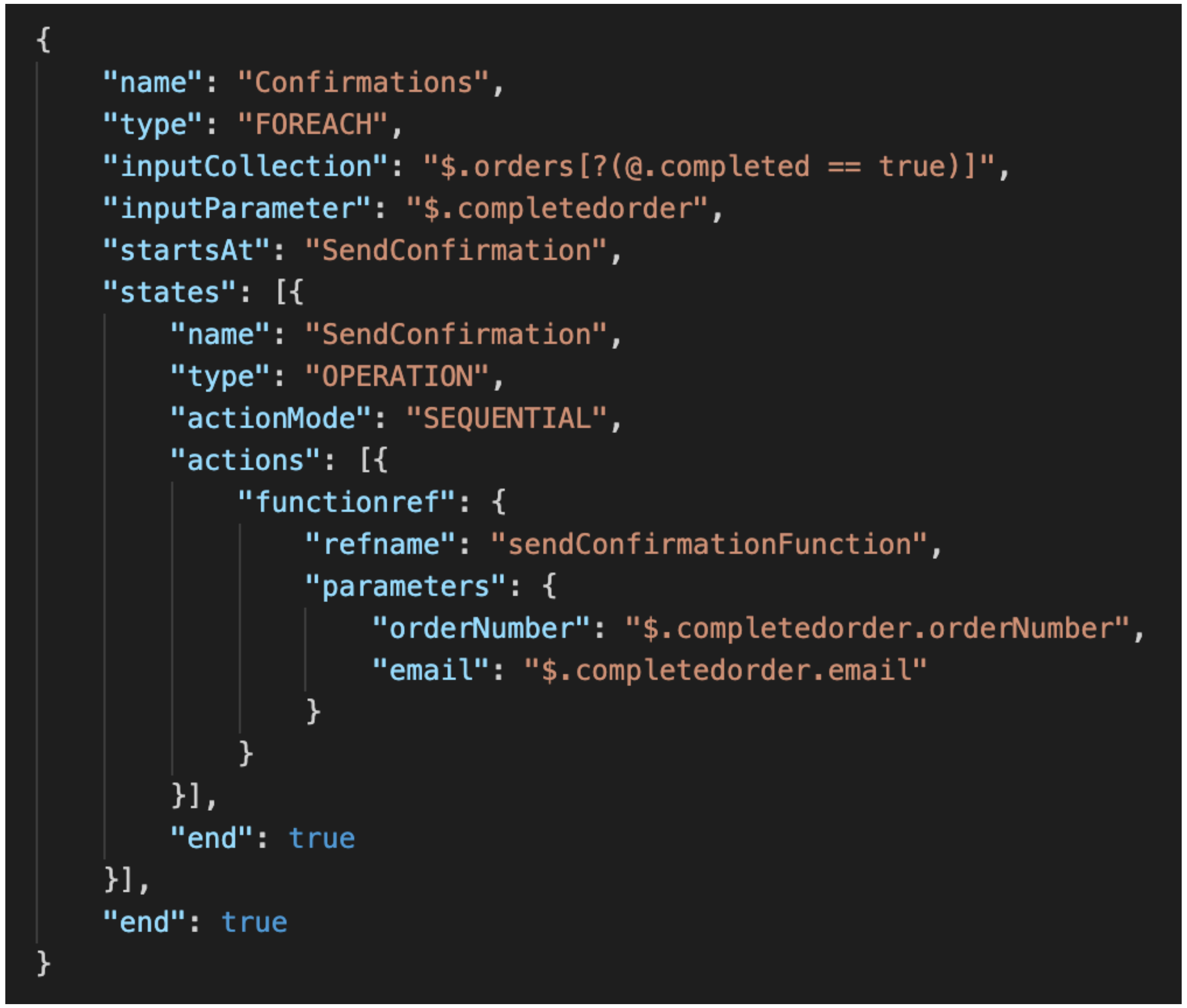
```
{
    "name": "ParallelExec",
    "type": "PARALLEL",
    "branches": [{
            "name": "Branch1",
            "startsAt": "Subflow1",
            "states": [{
                "name": "Subflow1",
                "type": "SUBFLOW",
                "workflowId": "workflows/branch1-subflow"
            }],
            "waitForCompletion": false
        },
        {
            "name": "Branch2",
            "startsAt": "Subflow2",
            "states": [{
                "name": "Subflow2",
                "type": "SUBFLOW",
                "workflowId": "workflows/branch2-subflow"
            }],
            "waitForCompletion": false
        }
    ],
    "end": true
}
```



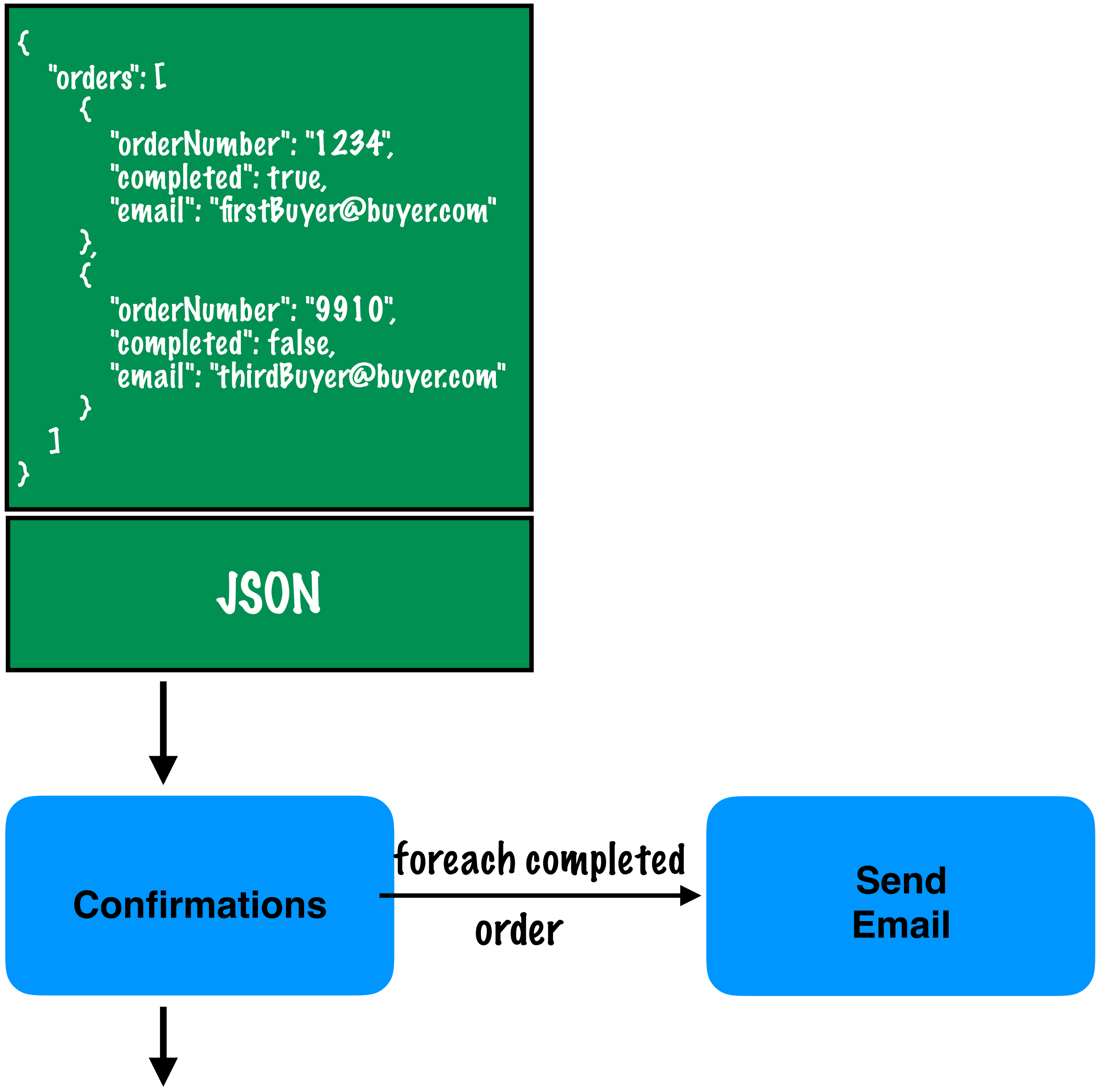
Workflow - ForEach State

- Define looping over data array

Serverless Workflow



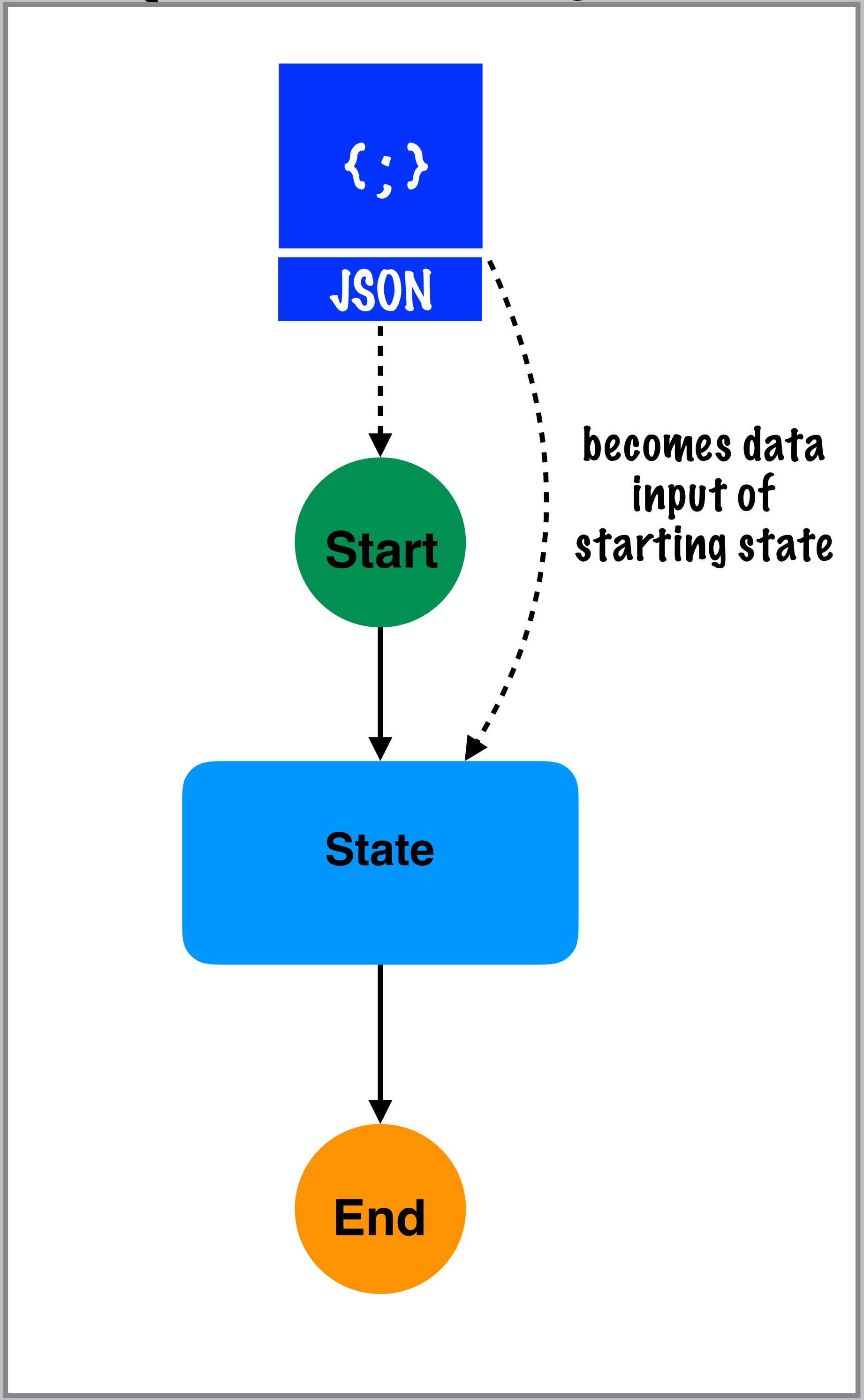
```
{
    "name": "Confirmations",
    "type": "FOREACH",
    "inputCollection": "$.orders[?(@.completed == true)]",
    "inputParameter": "$.completedorder",
    "startsAt": "SendConfirmation",
    "states": [{
        "name": "SendConfirmation",
        "type": "OPERATION",
        "actionMode": "SEQUENTIAL",
        "actions": [{
            "functionref": {
                "refname": "sendConfirmationFunction",
                "parameters": {
                    "orderNumber": "$.completedorder.orderNumber",
                    "email": "$.completedorder.email"
                }
            }
        }],
        "end": true
    }],
    "end": true
}
```



```
{
  "orders": [
     {
        "orderNumber": "1234",
        "completed": true,
        "email": "firstBuyer@buyer.com"
     },
     {
        "orderNumber": "9910",
        "completed": false,
        "email": "thirdBuyer@buyer.com"
     }
  ]
}
```

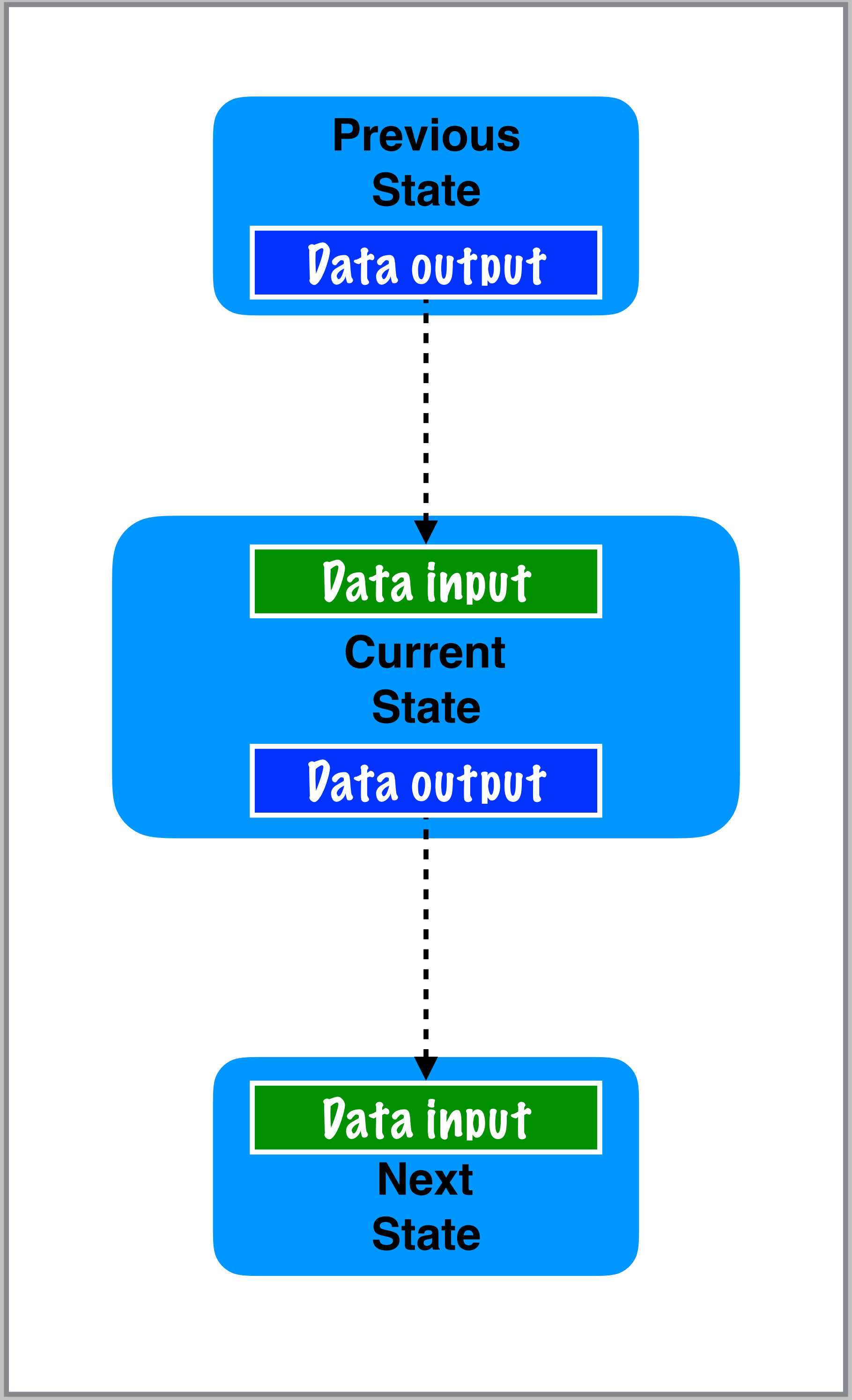
Workflow Data - Initial input

- All workflow data in JSON format
- Workflow data passed as input to starting state



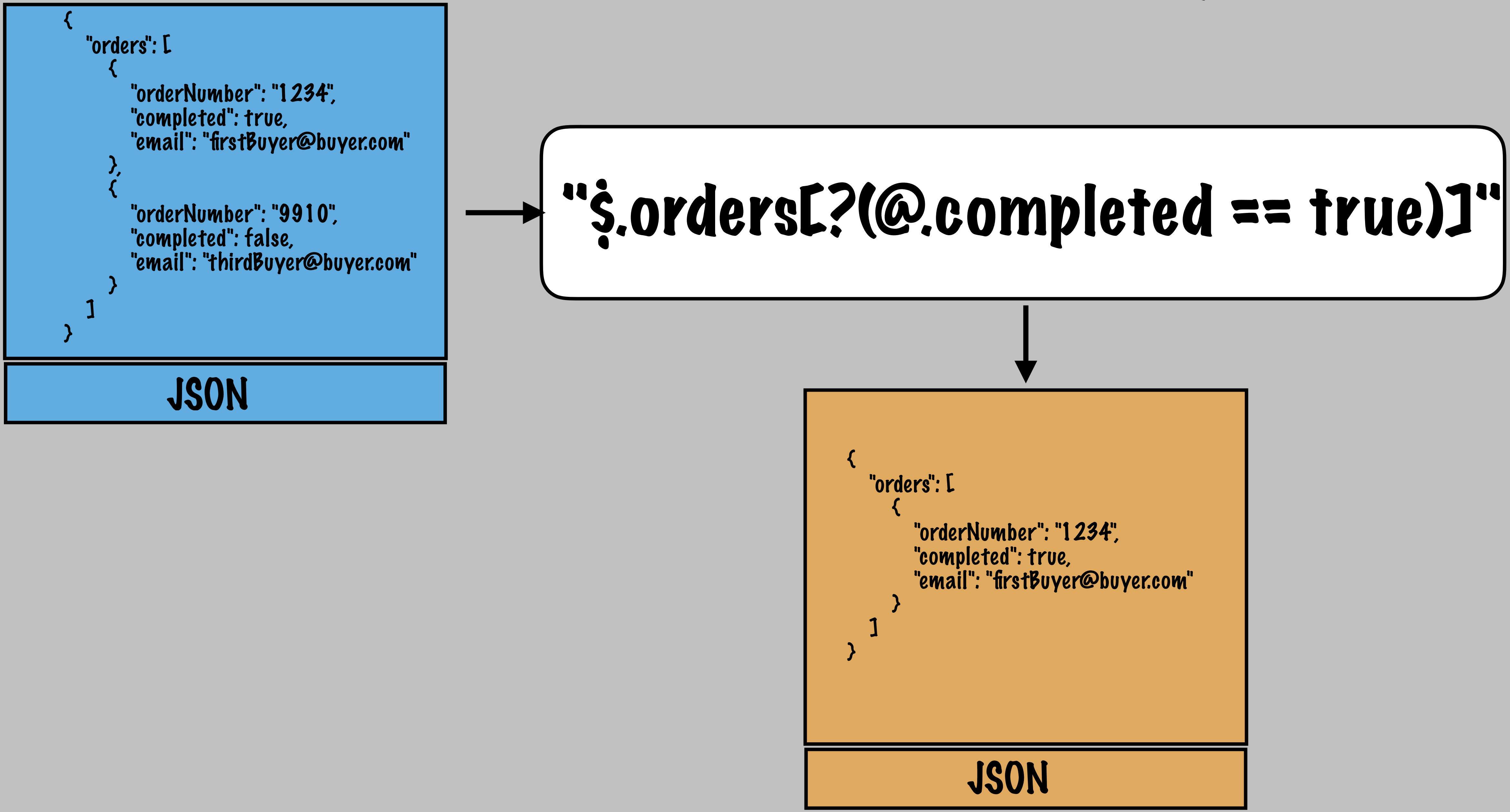
State Data - Information passing

- States receive data (data input)
- States produce data (data output)



State Data - Data filtering

- States access and manipulate data via Filters
- Filters are defined with JSONPath
- Filters can have three Paths
    - InputPath: select portion of states data input
    - OutputPath: select portion of states data output
    - ResultPath: select action results and combine with state data output



```
{
  "orders": [
    {
      "orderNumber": "1234",
      "completed": true,
      "email": "firstBuyer@buyer.com"
    },
    {
      "orderNumber": "9910",
      "completed": false,
      "email": "thirdBuyer@buyer.com"
    }
  ]
}
```

JSON

"$.orders[?(@.completed == true)]"

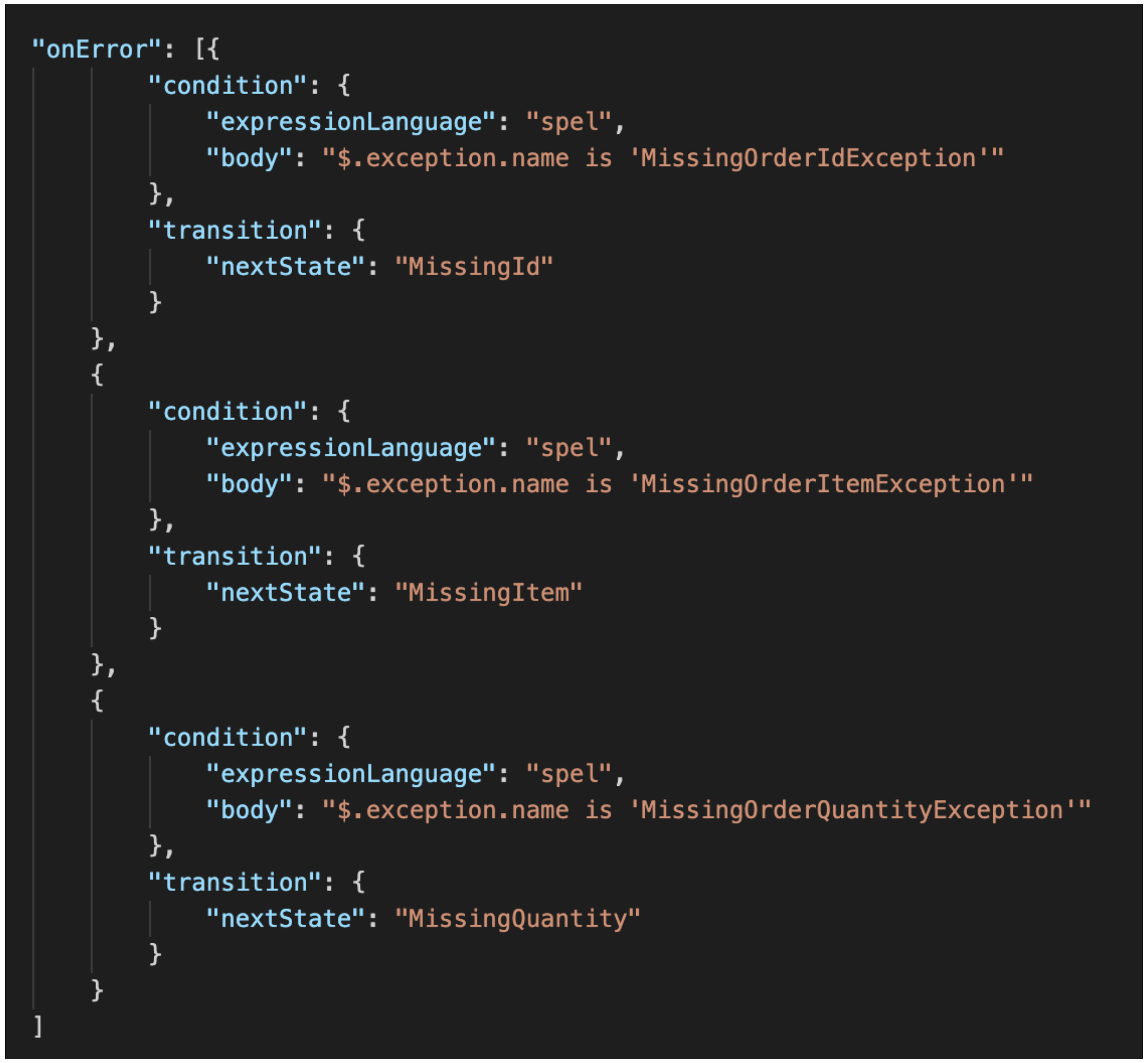
```
{
  "orders": [
    {
      "orderNumber": "1234",
      "completed": true,
      "email": "firstBuyer@buyer.com"
    }
  ]
}
```

JSON

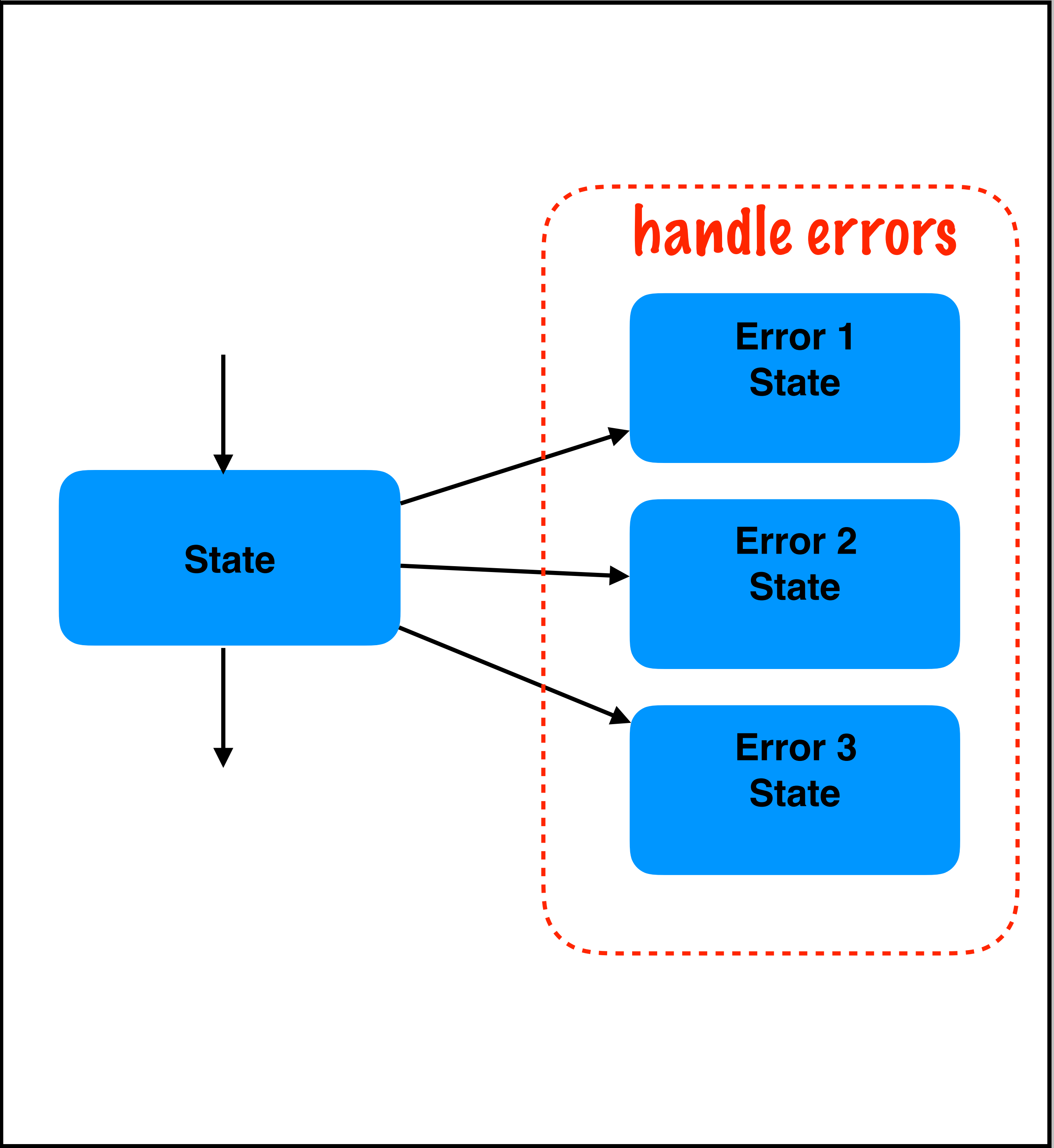
State - Error Handling

- **Explicitly** model logic in case of runtime errors
- **Transitions** based on error info

Serverless Workflow



```
"onError": [{
        "condition": {
            "expressionLanguage": "spel",
            "body": "$.exception.name is 'MissingOrderIdException'"
        },
        "transition": {
            "nextState": "MissingId"
        }
    },
    {
        "condition": {
            "expressionLanguage": "spel",
            "body": "$.exception.name is 'MissingOrderItemException'"
        },
        "transition": {
            "nextState": "MissingItem"
        }
    },
    {
        "condition": {
            "expressionLanguage": "spel",
            "body": "$.exception.name is 'MissingOrderQuantityException'"
        },
        "transition": {
            "nextState": "MissingQuantity"
        }
    }
]
```



Serverless Workflow Specification - Current Status

- Still in "Sandbox" state
- Looking for contributions
- Community sightings:
  - JSON Linter - https://github.com/serverless-workflow/workflow-linter-cli
  - VSCode plugin (JSON Schema) - https://marketplace.visualstudio.com/items?itemName=tsurdilovic.workflow-schema-vscode
  - JAVA API/SPI - https://github.com/serverless-workflow

Contributions Welcome!!

https://github.com/cncf/wg-serverless/tree/master/workflow/spec

Kogito - Introduction

Cloud-Native Business Automation for building intelligent applications, backed by battle-tested capabilities

Kogito - Introduction

Next generation business automation toolkit based on:

* Drools - the open source rule engine to provide decision and rule management capabilities

* jBPM - the open source process engine to provide automation and orchestration capabilities

It takes advantage of years of battle-tested features, while at the same time modernizing usage to fit the cloud-native ecosystem

Kogito - Business Domain

Kogito ergo domain

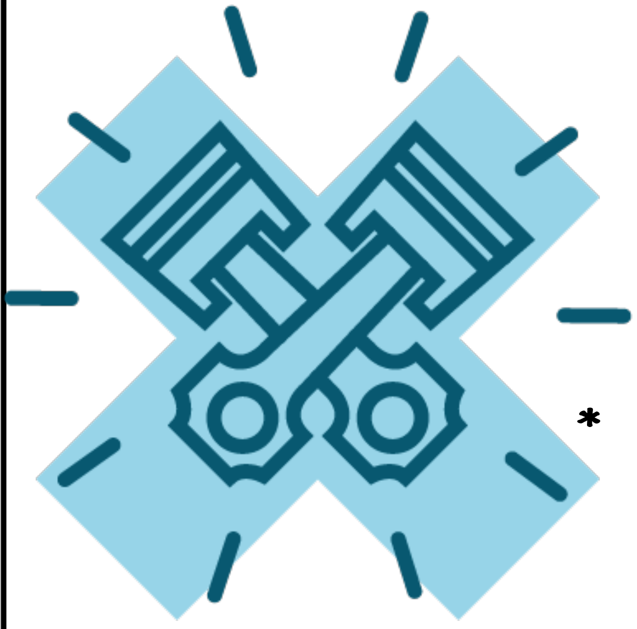
- Adapts to your business domain instead of the other way around
- No more leaking of abstractions of the platform into your client applications
- Stay focused on business requirements instead of implementation technology

Kogito - Business Domain

Not a little enhancement but a game changer

- Described and discoverable
  - swagger/openapi documentation
  - labels
- Backbone for complete and holistic business context
  - Follow up lists
  - Dashboards
  - Task lists

Kogito - Developer experience



Kogito ergo power

Offers a powerful developer experience

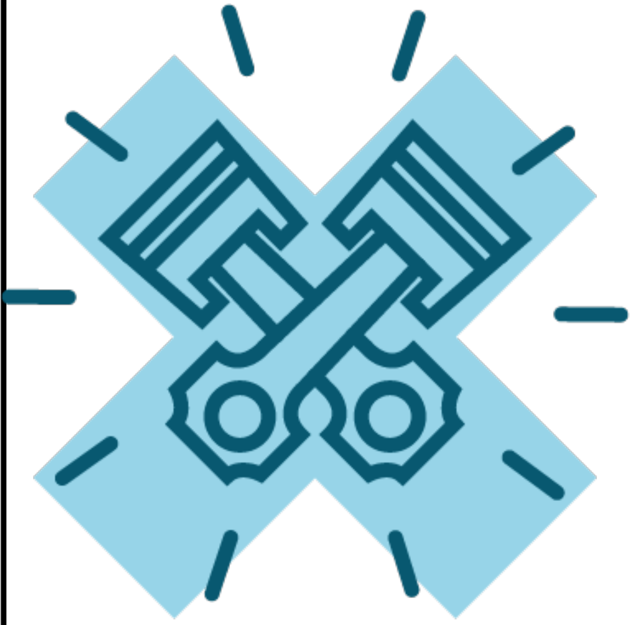
- Achieve instant developer efficiency by having
  - Tooling embeddable wherever you need it
  - Code generation taking care of 80% of the work
  - Flexibility to customize, only use what you need
  - Simplified local development with live reload

Kogito - Developer experience

Developer Focused

- Provides unified developer experience in an IDE that allows to work effectively with various asset types
- VSCode as the main target to allow
  - Model processes and decisions
  - Develop data model and services
  - Build and deploy
  - Iterative development experience with live reload

Kogito - Developer experience



Business Friendly

- Out of the box task inbox that can be
  - Customized
  - Embedded
- Business context based
  - Searches
  - Dashboards
- It's based on domain specific aspect of services that emit events that are index to provide business view instead of raw process or decision details

Kogito - Developer experience

Use existing and well-known tools

- Optionally equipped services with monitoring and tracing capabilities
  - Monitoring
    - Prometheus
  - Tracing
    - Open Tracing/Jeager
  - Visualization
    - Grafana

Kogito - Cloud

Kogito ergo cloud

Designed from ground up to run at scale

- If you think about business automation think about the cloud as this is where your business logic lives these days
- Achieves amazingly fast boot time and low memory footprint by leveraging newest technology
  - Quarkus
  - Kubernetes/OpenShift/KNative

Kogito - Cloud

Operation centric

- Responsible for building with selected runtime
  - Quarkus
  - Spring Boot
- Provision services on demand and remove them when no longer needed
- Manage deployed services including their dependencies
- Service discovery based on labels
- Guarding services of losing their dependants
- Security provisioning and propagation

Kogito - Get started

http://kogito.kie.org

Serverless Workflow

Kogito

Quarkus

Demo

Questions?