

## ICT & Infra S3 Automation & Orchestration, week 7

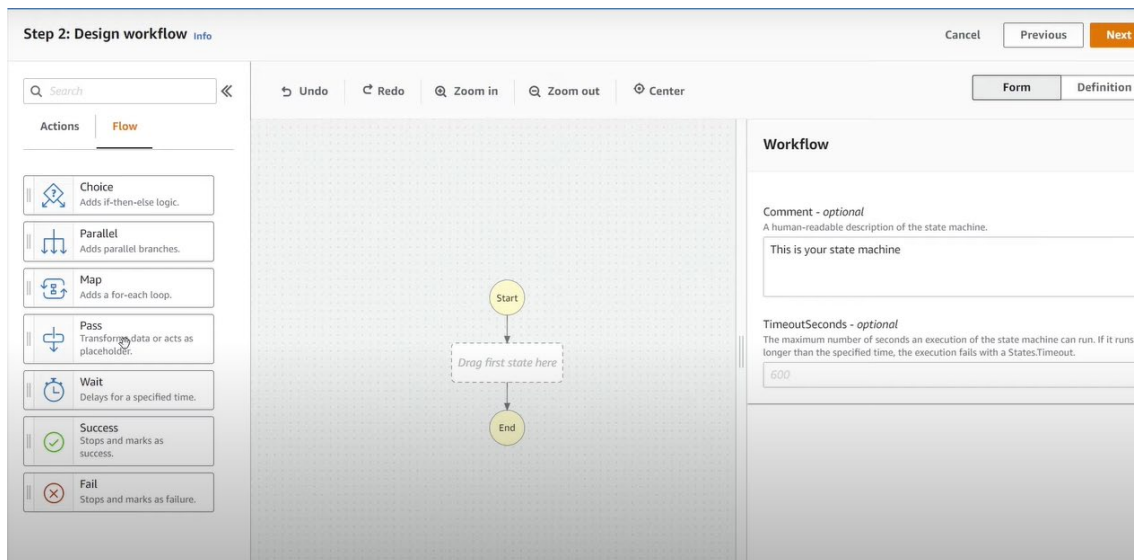
Class:	CB01
Student number:	4961854
Student name:	Heiko Morales

### Introduction

In this practice I will make a step function on my own in order to learn more about it. I will create a very basic function to learn more. The function will be something very simple, I will just replicate a credit card purchase.

### Assignment. Create a step function

I first entered the design workflow and started to investigate its functions.



as a first state, I created a choice and set the following conditions:

**Conditions for rule #1**

Choice rules contain conditional statements, which are used to evaluate one or more node values (called variables) in your state's JSON input. [Learn more](#)

Simple  
Evaluates a single conditional statement.

Not	Variable	Operator	Value
<input type="checkbox"/>	<input type="text" value="\$transactionType"/> <small>Must use JsonPath.</small>	<input type="text" value="is equal to"/>	<input type="text" value="PURCHASE"/>

Cancel Save conditions

**Conditions for rule #2**

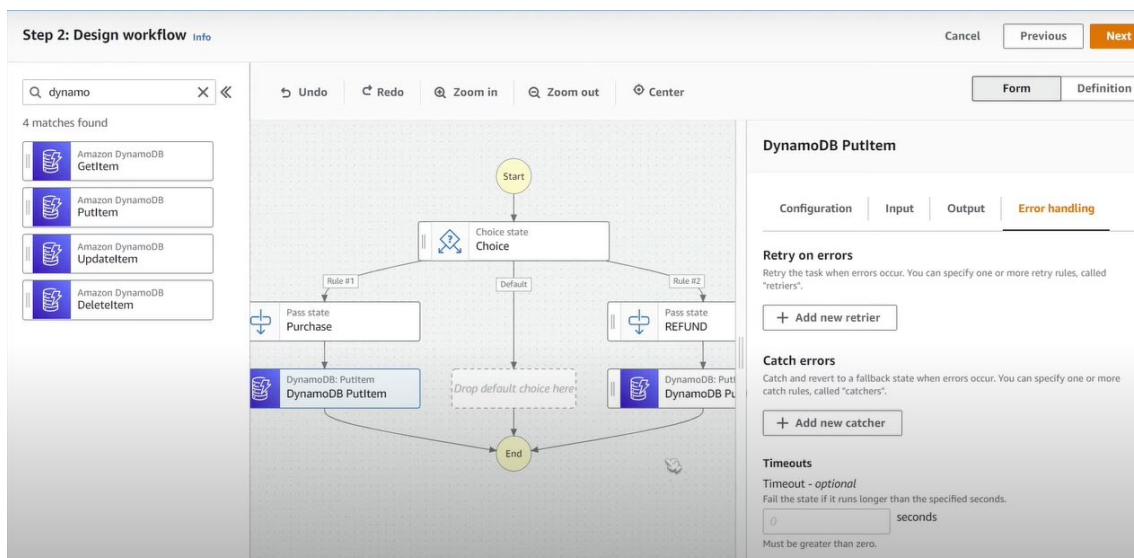
Choice rules contain conditional statements, which are used to evaluate one or more node values (called variables) in your state's JSON input. [Learn more](#)

Simple  
Evaluates a single conditional statement.

Not	Variable	Operator	Value
<input type="checkbox"/>	<input type="text" value="\$transactionType"/> <small>Must use JsonPath.</small>	<input type="text" value="is equal to"/>	<input type="text" value="REFUND"/>

Cancel Save conditions

Now I add DynamoDB from amazon to be able to store those transactions.



Then configure DynamoDB input and error handling.

### DynamoDB PutItem

Configuration | **Input 1** | Output | Error handling

During workflow execution, a Task state's input comes from the previous state's output. [Info](#)

☒ Filter input with InputPath - *optional* [Info](#)  
Use the InputPath filter to select a portion of the state input to use.

**Value cannot be empty.**  
Must use valid JSONPath syntax, and point to an existing key-value pair in the state input.

Form | Definition

Configuration | Input | Output | **Error handling**

#### Retry on errors

Retry the task when errors occur. You can specify one or more retry rules, called "retriers".

+ Add new retrier

#### Catch errors

Catch and revert to a fallback state when errors occur. You can specify one or more catch rules, called "catchers".

+ Add new catcher

#### Timeouts

**Timeout - optional**  
Fail the state if it runs longer than the specified seconds.

seconds

Must be greater than zero.

**Heartbeat - optional**  
Fail the state if more time than the specified seconds elapses between heartbeats.

seconds

Must be greater than zero.

Finally, I click next and aws generates the necessary code to implement what I specify.

### Review generated code - optional

Review your state machine's Amazon States Language (ASL) definition, which was generated by your actions in the previous step. Edit the definition if necessary.

Definition Export ▼ Layout ▼

Generate code snippet ▼ Format JSON

```
3  "startAt": "Choice",
4  "states": {
5    "Choice": {
6      "type": "Choice",
7      "choices": [
8        {
9          "variable": "$.transactionType",
10         "stringEquals": "PURCHASE",
11         "next": "Purchase"
12       },
13       {
14         "variable": "$.transactionType",
15         "stringEquals": "REFUND",
16         "next": "REFUND"
17       }
18     ],
19     "Purchase": {
20       "type": "Pass",
21       "next": "DynamoDB PutItem"
22     },
23     "DynamoDB PutItem": {
24
```

```
graph TD
    Start((Start)) --> Choice[Choice]
    Choice --> Purchase[Purchase]
    Choice --> REFUND[REFUND]
    Purchase --> PutItem1[DynamoDB PutItem]
    REFUND --> PutItem2[DynamoDB PutItem (1)]
    PutItem1 --> End((End))
    PutItem2 --> End
```

Finally, you will be asked for the configuration of your state machine.

## Specify state machine settings

### Name

State machine name

MyStateMachine

Must be 1-80 characters. Can use alphanumeric characters, dashes, or underscores.

### Permissions

#### Execution role

The IAM role that defines which resources your state machine has permission to access during execution. To create a custom role, go to the [IAM console](#).

☒ Create new role

Let Step Functions create a new role for you based on your state machine's definition and configuration details.

☐ Choose an existing role

☐ Enter a role ARN

### Logging

You can log your state machine's execution history to CloudWatch Logs. For Express state machines, you must enable logging to inspect and debug executions. CloudWatch Logs charges apply. [Learn more](#)

#### Log level

Indicates which execution history events to log

OFF

### Tracing

You can enable AWS X-Ray tracing on your state machine for end-to-end application debugging, performance profiling, and error analysis. Standard X-Ray charges apply. [Learn more](#)

☐ Enable X-Ray tracing

Step Functions will send traces to AWS X-Ray for state machine executions, even when a trace ID is not passed by an upstream service.

### Tags - optional

A tag is a label that you assign to an AWS resource. Each tag consists of a key and an optional value. You can use tags to search and filter your resources or track your AWS costs.

And that's it. I have a state machine created and ready to go.

State machine successfully created

NEW! AWS Step Functions now allows you more control over your event-driven applications.

You can now publish custom events from an AWS Step Functions state machine to an Amazon Eventbridge event bus, making it easier to build event-driven applications.

Learn more

Step Functions > State machines > MyStateMachine

MyStateMachine

EditStart executionDeleteActions

Details

ARN

IAM role ARN

Type

Standard

Creation date

Jun 19, 2021 02:09:04.890 PM

Executions

Logging

Definition

Tags

Executions (0)

View details

Stop execution

Start execution

Search for executions

Filter by status

< 1 >

Name

Status

Started

End Time