



# Create Step Functions State Machine Using AWS Lambda

1. So, this lambda is well-suited for task states because lambda functions are serverless and easy to write. We can write code in the mission console or favourite editor a place, handle the details, or provide competing for our function and run it.
2. Now in your AWS Console go to Lambda and create two functions.
3. Your first function is **PurchaseFunction** you must choose Python 3.11 for this and just create your lambda function.

The screenshot shows the AWS Lambda Functions page with the 'PurchaseFunction' selected. The 'Function overview' tab is active. The main area displays the function name 'PurchaseFunction' and a 'Layers' section showing '(0)'. Below these are buttons for '+ Add trigger' and '+ Add destination'. On the right side, there are sections for 'Description' (empty), 'Last modified' (4 minutes ago), 'Function ARN' (arn:aws:lambda:ap-south-1:878893308172:function:PurchaseFunction), and 'Function URL' (info). Top navigation includes 'Throttle', 'Copy ARN', 'Actions', 'Export to Application Composer', and 'Download'.

4. And you must paste the code given below.

```
import json

def lambda_handler(event, context):
    print("event:", json.dumps(event, indent=2))
    print(event.get('OrderType'))

    response = {
        "statusCode": 200,
        "body": f'Hello from {PurchaseFunction} OrderType: {event.get("OrderType")}'
    }
    return response
```

5. Also, you can test the code using the below statement.

```
{
    "OrderType": "Purchase"
}
```

Code | Test | Monitor | Configuration | Aliases | Versions

**Code source** Info

File Edit Find View Go Tools Window Test Deploy

Go to Anything (Ctrl-P) lambda\_function Environment Var +

Environment PurchaseFunction lambda\_function.py

```
1 import json
2
3 def lambda_handler(event, context):
4     print("event:", json.dumps(event, indent=2))
5     print(event.get('OrderType'))
6
7     response = {
8         "statusCode": 200,
9         "body": f'Hello from PurchaseFunction. OrderType: "{event.get("OrderType")}"'
10    }
11
12    return response
```

Executing function: succeeded ([logs](#))

▼ Details

The area below shows the last 4 KB of the execution log.

```
{  
    "statusCode": 200,  
    "body": "Hello from PurchaseFunction. OrderType: \"Purchase\""  
}
```

6. After that you must create another function **RefundFunction** using the same Python 3.11 as your runtime and just create your function.
7. Then you must add the code given below.

```
import json

def lambda_handler(event, context):
    print("event:", json.dumps(event, indent=2))
    print(event.get('OrderType'))

    response = {
        "statusCode": 200,
        "body": f'Hello from RefundFunction. OrderType: {event.get("OrderType")}'
    }
    return response
```

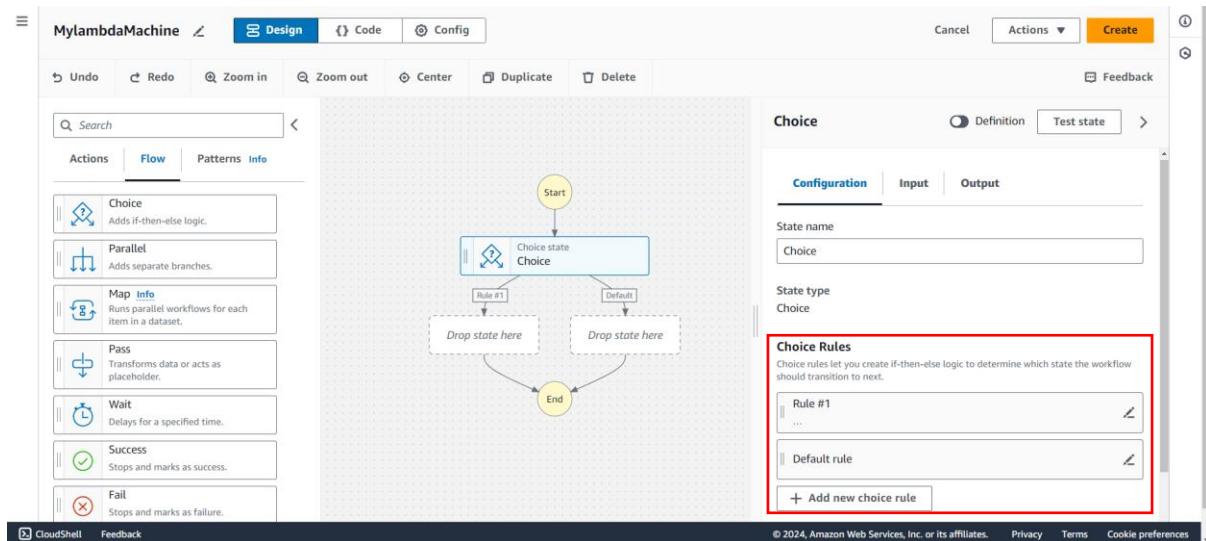
```

1 import json
2
3 def lambda_handler(event, context):
4     print("event:", json.dumps(event, indent=2))
5     print(event.get('OrderType'))
6
7     response = {
8         "statusCode": 200,
9         "body": f'Hello from RefundFunction. OrderType: "{event.get("OrderType")}"'
10    }
11
12    return response

```

8. Now go to Step Function create a new machine choose the blank template to start with and click on Select.

9. Now here from the flow choose choice step and drag it to middle and now you are going to create two rules. Ignore the default rule.



10. Choose rule 1 and click on add condition. Then you must choose the same conditions as you can see in the snapshot.

## Choice Rules

Choice rules let you create if-then-else logic to determine which state the workflow should transition to next.

**Rule #1** X Close

---

If these conditions are true:

**Add conditions**

Then next state is:

**Add new state** ▼

## 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
▼	\$.OrderType	is equal to	String constant ▼ PURCHASE
Must use JsonPath.			

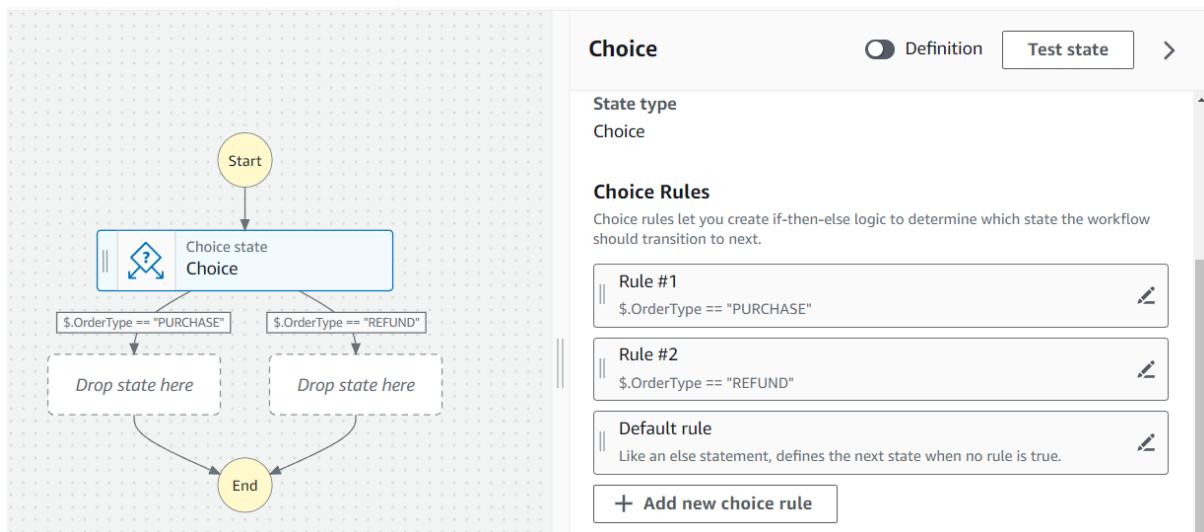
11. Then in a similar way you have to create rule 2 but for refund and save it. Also, you can see the diagram.

## Conditions for rule #2

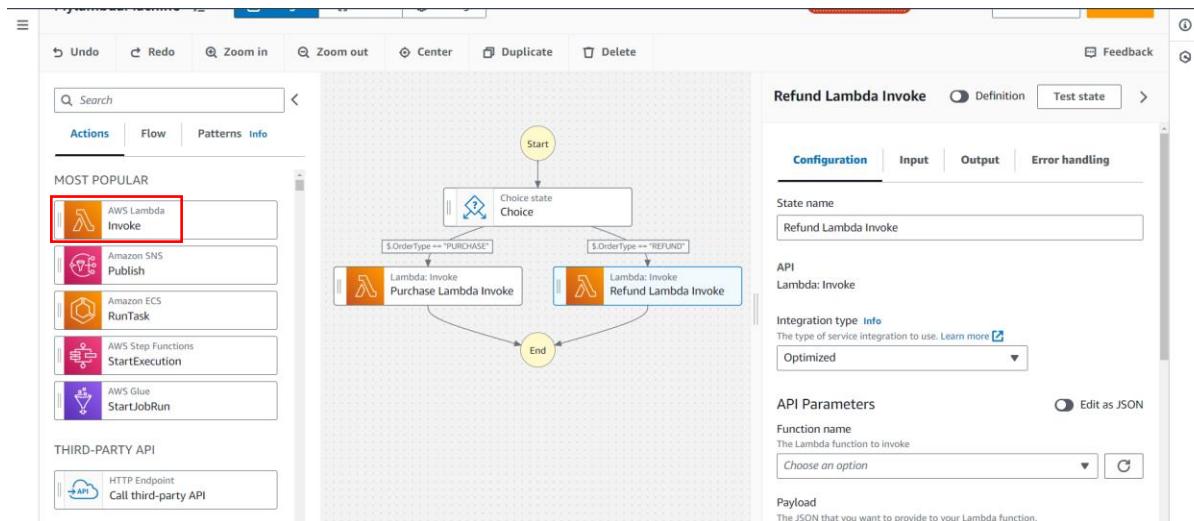
Choice rules contain conditional statements, which are used to evaluate one or more node values (called variables) in your state's JSON input. [Learn more](#)

Simple  
Evaluates a single conditional statement.

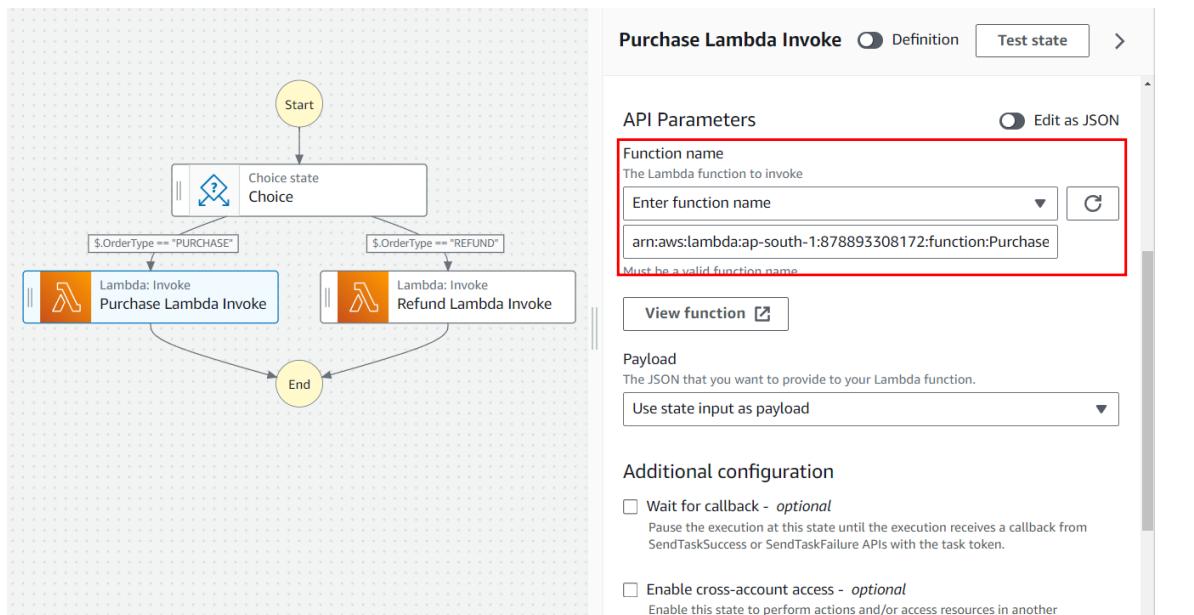
Not	Variable	Operator	Value
▼	\$.OrderType	is equal to	String constant ▼ REFUND
Must use JsonPath.			



12. Now you need go to actions, choose AWS Lambda Invoke step, drag and drop it under choice, then just change their name. To change their name, you have to click on lambda invoke and from the configuration you can change the name.



13. Now go to one of the lambda functions choose it and then in the function name choose Purchase lambda. Similarly, go to another function and choose the Refund function.



The screenshot shows the AWS Step Functions console. On the left, a state machine diagram is displayed. It starts with a yellow 'Start' node, followed by a 'Choice state Choice' node. This leads to two parallel execution paths: one for 'Purchase' (Condition: '\$.OrderType == "PURCHASE"') leading to a 'Purchase Lambda Invoke' state, and another for 'Refund' (Condition: '\$.OrderType == "REFUND"') leading to a 'Refund Lambda Invoke' state. Both paths converge back to a yellow 'End' node. On the right, the configuration for the 'Refund Lambda Invoke' step is shown. It is set to 'Optimized' and uses the 'RefundFunction:\$LATEST' Lambda function. The 'Function name' field is highlighted with a red box. Other settings include 'Use state input as payload' for 'Payload' and various optional configurations like 'Wait for callback' and 'Enable cross-account access'.

14. Now click on create and your step machine will be created. Just confirm the roles.

The screenshot shows the 'Confirm role creation' dialog. It contains a message box stating: 'An execution role will be created with full permissions. A new execution role named StepFunctions-MylambdaMachine-role-9vvlpy3h3 will be created. All required permissions for the actions specified in your state machine will be auto-generated.' Below this, a table titled 'Role permissions' lists the services and actions that will be granted permissions. For AWS Lambda, it grants 'lambda:Invoke'. For AWS X-Ray, it grants 'xray:PutTraceSegments', 'xray:PutTelemetryRecords', 'xray:GetSamplingRules', and 'xray:GetSamplingTargets'. At the bottom, there are 'Cancel', 'View role configuration', and 'Confirm' buttons.

Service	Action(s)	Status	Documentation links
AWS Lambda	lambda:Invoke	Policy will be generated to perform the action for specified Lambda resources only	<a href="#">Call Lambda with Step Functions</a> <a href="#">Lambda policies for Step Functions</a>
AWS X-Ray	xray:PutTraceSegments xray:PutTelemetryRecords xray:GetSamplingRules xray:GetSamplingTargets	Policies will be generated for X-Ray tracing	<a href="#">X-Ray policies for Step Functions</a>

15. Then click to start your execution.

The screenshot shows the 'MylambdaMachine' state machine details page. It includes sections for 'Details' (with Arn, Type, Status, Creation date, and X-Ray tracing information), 'Executions' (0), 'Monitoring', 'Logging', 'Definition', 'Aliases', 'Versions', and 'Tags'. The 'Executions' tab is selected, showing a table with columns for Name, Status, Start Time (local), End Time (local), Duration, Version, and Alias. A search bar at the top of the executions table allows filtering by property or value, and a button to 'Start execution' is visible.

16. Now it depends on which function you want to execute, so you need to pass the statement for that. As you can see below is the Refund statement so the refund lambda function will be called.

```
{  
  "OrderType": "Refund"  
}
```

## Start execution

Name

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

**Input - optional**

Enter input values for this execution in JSON format

```
1 {  
2   "OrderType": "REFUND"  
3 }
```

 [Start execution with latest revision](#)

[Open in a new browser tab](#)

17. Below you can see that the execution succeeded.

Step Functions > State machines > MylambdaMachine > Execution: 552f5ed0-4010-4bc0-b51e-1c62f5a1af9e

Execution: 552f5ed0-4010-4bc0-b51e-1c62f5a1af9e

Details | Execution input and output | Definition

Execution status: Succeeded

Execution type: Standard

Execution ARN: arn:aws:states:ap-south-1:878893308172:execution:MylambdaMachine:552f5ed0-4010-4bc0-b51e-1c62f5a1af9e

IAM role ARN: arn:aws:iam::878893308172:role/service-role/StepFunctions-MylambdaMachine-role-9vvlyp3h3

State transitions: Learn more 4

Start time: Aug 23, 2024, 19:33:21.025 (UTC+05:30)

End time: Aug 23, 2024, 19:33:21.448 (UTC+05:30)

Duration: 00:00:00.423

Alias: -

Version: -

**Graph view**

```

graph TD
    Start((Start)) --> Choice{Choice}
    Choice -- Purchase Lambda Invoke --> End((End))
    Choice -- Refund Lambda Invoke --> End
  
```

**Refund Lambda Invoke**

Logs | Lambda | Log group

Test state

Input | Output | Details | Definition | Events

Advanced view

```

1 {
2   "OrderType": "REFUND"
3 }
  
```

Formatted ↗

Actions ▾

Graph view | Refund Lambda Invoke

1 in progress 2 Failed 3 Caught error 4 Canceled 5 Succeeded

18. Similarly, for Purchase. Click on new Execution and give the purchase statement.

## Start execution

Name

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

*Input - optional*

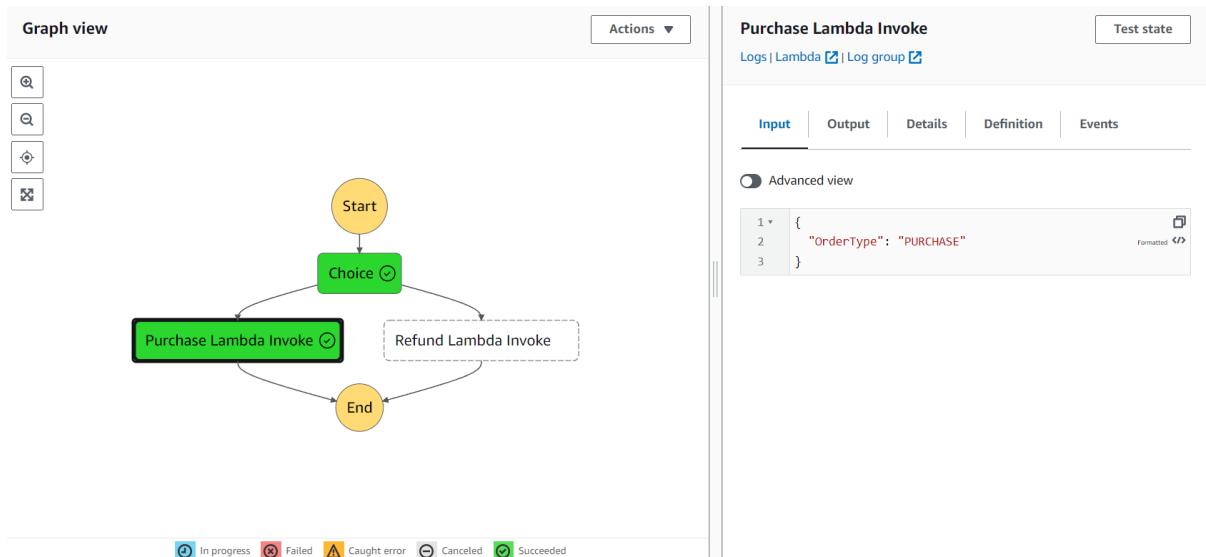
Enter input values for this execution in JSON format

Format JSONExportImport

```
1 ▾ {  
2   "OrderType": "PURCHASE"  
3 }
```

 [Start execution with latest revision](#)

[Open in a new browser tab](#)



19. Once you are done then delete your lambda function and the step functions.