Stock Replenishment App

**SPL-BE-200-DVSRPA-1 - Version 1.0.6**

© 2024 Amazon Web Services, Inc. or its affiliates. All rights reserved. This work may not be reproduced or redistributed, in whole or in part, without prior written permission from Amazon Web Services, Inc. Commercial copying, lending, or selling is prohibited. All trademarks are the property of their owners.

Note: Do not include any personal, identifying, or confidential information into the lab environment. Information entered may be visible to others.

Corrections, feedback, or other questions? Contact us at [*AWS Training and Certification*](https://support.aws.amazon.com/#/contacts/aws-training).

**Lab Overview**

This lab simulates and event-based application. An event-based application is a software architecture pattern where components of the system communicate through events. Events are discrete messages that represent a change in state or occurrence in the system. In event-driven-architecture, components are decoupled, which allows them to act independently when an event is generated, leading to improved scalability, flexibility, and fault tolerance.

The event-based application used in this lab is made up of the following AWS services. AWS EventBridge (formerly known as AWS EventBus), is the service responsible for managing the flow of events between different components of the application. AWS Event Rules, are used to filter and route events based on specific criteria. AWS Serverless HTTP API is an API Gateway service that manages the HTTP requests and responses. AWS Serverless Functions are Lambda functions that process events and perform specific actions in the response. DynamoDB is used to store and retrieve the application’s data.

The application is based on an inventory system. The *Get Stock Level* option is working when the lab starts. The *Create Purchase* option does not work. You are tasked with troubleshooting why this option fails and then make the necessary updates so it is operational.

OBJECTIVES

By the end of this lab, you will be able to do the following:

* Troubleshoot the event-based application.
* Review events written to the event bus
* Review the EventBridge bus rules.
* Update and redeploy the event-based application.

TECHNICAL KNOWLEDGE PREREQUISITES

To successfully complete this lab:

* Familiarity with the basic navigation of the AWS Management Console.
* Versed in editing and running scripts using an AWS Cloud9 code editor and terminal.
* A basic understanding and familiarity with Amazon API Gateway, AWS Serverless Application Model (SAM), AWS Lambda, and AWS CloudFormation.
* Prior experience with AWS services and serverless computing will be helpful but is not necessarily required.

DURATION

This lab requires *60* minutes to complete.

ICON KEY

Various icons are used throughout this lab to call attention to different types of instructions and notes. The following list explains the purpose for each icon:

* **Command:** A command that you must run.
* **Expected output:** A sample output that you can use to verify the output of a command or edited file.
* **Note:** A hint, tip, or important guidance.
* **Consider:** A moment to pause to consider how you might apply a concept in your own environment or to initiate a conversation about the topic at hand.
* **Task complete:** A conclusion or summary point in the lab.

**Start lab**

1. To launch the lab, at the top of the page, choose **Start lab**.

**Caution:** You must wait for the provisioned AWS services to be ready before you can continue.

1. To open the lab, choose **Open Console**.

You are automatically signed in to the AWS Management Console in a new web browser tab.

**WARNING:** **Do not change the Region unless instructed.**

COMMON SIGN-IN ERRORS

**Error: You must first sign out**



If you see the message, **You must first log out before logging into a different AWS account:**

* Choose the **click here** link.
* Close your **Amazon Web Services Sign In** web browser tab and return to your initial lab page.
* Choose **Open Console** again.

**Error: Choosing Start Lab has no effect**

In some cases, certain pop-up or script blocker web browser extensions might prevent the **Start Lab** button from working as intended. If you experience an issue starting the lab:

* Add the lab domain name to your pop-up or script blocker’s allow list or turn it off.
* Refresh the page and try again.

**Task 1: Review CloudWatch Logs**

In this task, you connect to the Cloud9 environment and review the CloudWatch Logs for the */application/StockApp* log group. You are looking for clues to help you determine what is causing the Create Purchase option to fail.

TASK 1.1: CONNECT TO CLOUD9

Connect to the Cloud9 environment provisioned as part of this lab.

1. Copy the **Cloud9Environment** URL link from the **Lab Information** section to the left of these instructions and paste it into a new browser tab. The browser takes you to the Cloud9 environment that you use during this lab.
2. You do not need the **Cloud9 Welcome screen** or any of the other default tabs that appear when you first launch **Cloud9**, so choose the **x** next to each tab to close them. This section is where you update various file throughout this lab.

**Consider:** You are working in another Cloud9 environment similar to other labs. The only difference is the application files that you see in the file tree. If you need a refresher, take a moment to familiarize yourself with the **AWS Cloud9** IDE interface by expanding the *Cloud9 review* section.

**Cloud9 review**

TASK 1.2: REVIEW THE APPLICATION CLOUDWATCH LOGS

There is a role writing logs to CloudWatch from the Even App. Start troubleshooting efforts by reviewing the CloudWatch logs in the log group specific to the application. One option to view the events of the log group as they occur is to run the *tail* command. A tail command outputs the most recent additions to a log file as they occur. This is useful in situations like this when you want to monitor a log file in real-time.

1. **Command:** Run the following **tail** command in the terminal session to review the most recent events added to the **/application/StockApp** log group:

aws logs tail /application/StockApp --follow --format json

**Expected output:**

*None, until the application is invoked to create a purchase or get a stock level.*

1. From the menu at the left of these instructions, copy the **WebsiteURL** value and open it in a new browser tab.

The browser opens the *Event App* API interface. You see following two buttons.

**Create Purchase** **Get Stock Level**

1. Choose **Get Stock Level** once to get the current inventory count for this item.

**Expected output:**

| **Location** | **SKU** | **Stock Level** |
| --- | --- | --- |
| 90210 | ItemX | 5 |

The *Stock Level* has been populated with a value of *5*.

1. Choose **Create Purchase** one time. This simulates making a purchase which should reduce the on-hand quantity by 1.
2. Now choose **Get Stock Level** to see the current stock level.

It still shows a quantity of *5*. Why?

1. Switch back to the browser tab opened to the Cloud9 environment.

The tail command is still running and should now be updated with a log entry.

**Expected output:**

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\*\* This is OUTPUT ONLY. \*\*\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

2023-03-23T22:01:49+00:00 ea8c41df-0856-3608-9e18-6e992f69fe20

{

"version": "0",

"id": "84641f7f-4c8d-e4c8-b6d9-037a28eb445c",

"detail-type": "i-am-broken",

"source": "StockApp",

"account": "111111111111",

"time": "2023-03-23T22:01:49Z",

"region": "us-west-2",

"resources": [],

"detail": {

"Location": "90210",

"SKU": "ItemX",

"PurchaseCount": "1"

}

}

When you look at the log information you see that the *detail-type* key has a value of *i-am-broken*. In the next task, you learn how this value is being set and if it is the correct value or if it should be updated.

TASK 1.3: REVIEW THE APPLICATION PERFORMING THE PUT EVENTS

The *purchase/app.py* file is responsible for the *put event* actions which is responsible for the log entry you looked at previously. Open this file to see what it is writing the event to for the *DetailType* value.

1. From the file-tree, open the file **purchase/app.py**.
2. Under the **eventbridge.put\_events()** find the **DetailType** and its value.

The value is set to *i-am-broken*. This is a good indicator that the value needs to be updated, but what should it be set to?

Given this is tied to an *event bus*, you should be able to look at the *rules* for the *event bus* to learn more about the values it is configured to use.

**Task complete:** You have successfully reviewed the CloudWatch logs to help determine what is causing the Purchase event to fail. In the next task, you implement the corrective steps to get it working.

**Task 2: Review EventBridge rules and update the application**

In this task, you review the EventBridge rules to see if they can help sort out what the *DetailType* key value should be set to. Once the value has been identified, you update the application and redeploy it to test the update.

TASK 2.1: REVIEW THE EVENTBRIDGE RULES

Open the EventBridge console to review the EventBridge rules.

1. At the top of the AWS Management Console, in the search bar, search for and choose

EventBridge

.

1. From the EventBridge navigation menu to the left, choose  **Buses** > **Rules**.
2. From the **Select event bus** section, choose **StockAppBus**.
3. From the **Rules** section of the **Rules** page, choose the link with **events-app-InventoryFunctionDeliveryEvents** in the name.

In the *Event pattern* section, you see the values for the *detail-type* being set as one of two options: [“Purchase”, “Replenish”]. Since this is a *purchase event*, the application needs to have the *DetailType* value set to *Purchase*.

TASK 2.2: UPDATE AND TEST THE APPLICATION

In this task, you update the application with the correct value for the purchase event *DetailType* with the value expected by the EventBridge rules. Once updated, you redeploy the application and then test it to see if it corrects the errors you were seeing.

1. Return to the browser tab opened to the Cloud9 environment.
2. The **purchase/app.py** file should already be opened for editing. If closed, go ahead and re-open it.
3. Update the **DetailType** key value to

Purchase

.

**Note:** That line of coded should read to match the example below:

* *‘DetailType’: ‘Purchase’,*

1. Save your changes.
2. Open a new terminal by choosing the green plus icon  and select **New Terminal**.
3. **Command:** Run the following grouped set of commands to change to the **~/environment** directory, build the deployment files, and redeploy the application with the latest code updates using the SAM CLI:

cd ~/environment; sam build; sam deploy

**Expected output:** Output has been truncated.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\*\* This is OUTPUT ONLY. \*\*\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Build Succeeded

Deploying with following values

===============================

Stack name : events-app

Region : us-west-2

Confirm changeset : False

Disable rollback : False

Deployment s3 bucket : aws-sam-cli-managed-default-samclisourcebucket-r6pi84czdjdp

Capabilities : ["CAPABILITY\_IAM"]

Parameter overrides : {}

Signing Profiles : {}

CloudFormation outputs from deployed stack

---------------------------------------------------------------------------

Outputs

---------------------------------------------------------------------------

Key TableName

Description Dynamo table name

Value Inventory

Key ApiGatewayEndpoint

Description API Gateway endpoint URL

Value https://949dybb2t1.execute-api.us-west-2.amazonaws.com/Prod/

---------------------------------------------------------------------------

Successfully created/updated stack - events-app in us-west-2

1. With the application updated, switch back to the browser tab opened to the website and choose **Create Purchase** .
2. Now choose **Get Stock Level** .

**Expected output:**

| **Location** | **SKU** | **Stock Level** |
| --- | --- | --- |
| 90210 | ItemX | 4 |

Notice that the stock level is reduced by one and now shows *4*.

1. Switch back to the Cloud9 browser tab and review the logs once again to see the newest log events.

**Expected output:**

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\*\* This is OUTPUT ONLY. \*\*\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

2023-03-24T05:31:42+00:00 63c631b1-327e-301b-b7cf-b0ed84278401

{

"version": "0",

"id": "3f8ae0c4-e8dc-675a-7cf9-edc7ffd44a4e",

"detail-type": "Purchase",

"source": "StockApp",

"account": "716640456232",

"time": "2023-03-24T05:31:42Z",

"region": "us-west-2",

"resources": [],

"detail": {

"Location": "90210",

"SKU": "ItemX",

"PurchaseCount": "1"

}

}

2023-03-24T05:31:43+00:00 2f84fe72-c9fe-3780-a0b2-59741e8052ca

{

"version": "0",

"id": "23d39b97-f2aa-7f78-80c5-0ee8a92571bb",

"detail-type": "StockLevel",

"source": "StockApp",

"account": "716640456232",

"time": "2023-03-24T05:31:43Z",

"region": "us-west-2",

"resources": [],

"detail": {

"Location": "90210",

"SKU": "ItemX",

"StockLevel": "4"

}

}

**Note:** If you encounter the following message instead of seeing the latest log entries:

* An error occurred (ExpiredTokenException) when calling the FilterLogEvents operation: The security token included in the request is expired
* **Command:** Run the the following command to refresh your tail of the logs:

aws logs tail /application/StockApp --follow --format json

You see the logs update with two events. One where the detail-type is *Purchase* and another where it is *StockLevel*, which now shows a value of *4*.

The application has written the event onto the bus, the Lambda function reacted to this event and reduced the stock level by 1. Then the function reported back the new stock level.

**Note:** **Note:** There is another Lambda function listening to the event bus. When stock levels fall below *2*, it will fire off a *Replenish* event. If you make additional stock purchases and then check the stock levels, you can see when they have been increased by a quantity of *10* for a new total of *11*.

**Task complete:** You have successfully troubleshooted the issue with the application. You reviewed the EventBridge *StockAppBus* and found that the *events-app-InventoryFunctionDeliveryEvents* rule was set to look for an event pattern for the detail-type being set as one of two options: [“Purchase”, “Replenish”]. You made the update to the application and redeployed it using the SAM CLI. Then the purchase feature started working with allowed the inventory to be reduced essentially fixing the broken application.

**Conclusion**

**Task complete:** You now have successfully:

* Troubleshooted the event-based application.
* Reviewed events written to the event bus
* Reviewed the EventBridge bus rules.
* Updated and redeployed the event-based application.

**End lab**

Follow these steps to close the console and end your lab.

1. Return to the **AWS Management Console**.
2. At the upper-right corner of the page, choose **AWSLabsUser**, and then choose **Sign out**.
3. Choose **End lab** and then confirm that you want to end your lab.

**Additional Resources**

* For more information about how to use AWS Lambda, see [AWS Lambda Documentation](https://docs.aws.amazon.com/lambda/index.html).

For more information about AWS Training and Certification, see [*https://aws.amazon.com/training/*](https://aws.amazon.com/training/).

*Your feedback is welcome and appreciated.*  
If you would like to share any feedback, suggestions, or corrections, please provide the details in our [*AWS Training and Certification Contact Form*](https://support.aws.amazon.com/#/contacts/aws-training).