😊 Amazon EventBridge

Amazon EventBridge is a serverless event bus service provided by Amazon Web Services (AWS). It enables you to connect your applications with data from various sources, such as AWS services, SaaS applications, and custom applications.

Here's a brief overview of its key features:

1. **Event Bus:** EventBridge provides a central event bus where events from different sources are routed and can be processed by various targets.
2. **Event Sources:** It supports a wide range of event sources, including AWS services like AWS Lambda, Amazon S3, Amazon SQS, Amazon SNS, etc., as well as third-party SaaS applications and custom applications.
3. **Rules:** EventBridge allows you to define rules to route events from event sources to specific event targets based on certain criteria, such as event patterns or event content.
4. **Event Targets:** You can specify one or more event targets for each rule. Supported targets include AWS Lambda functions, SNS topics, SQS queues, Kinesis streams, Step Functions state machines, and more.
5. **Schema Registry:** EventBridge includes a schema registry that allows you to define the structure of your events using JSON Schema. This helps with event validation and provides a standard way to understand the data format of events.
6. **Integration with AWS Organizations:** EventBridge integrates with AWS Organizations, allowing you to easily share events across accounts within your organization.

😄 Use Cases of EventBridge:

AWS EventBridge can be utilized in a variety of use cases across different industries and scenarios. Here are some common examples:

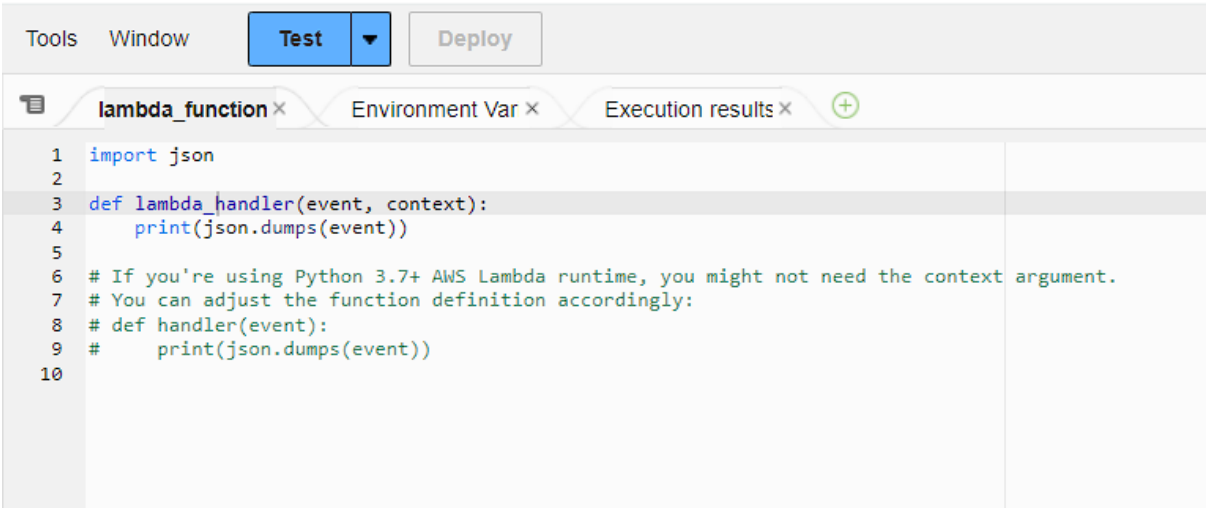
1. **Automated Workflows:** EventBridge can orchestrate automated workflows by triggering actions in response to events. For instance, you can set up a workflow where an event from an AWS service like S3 indicating the upload of a new file triggers a Lambda function to process the file and store the result in a database.
2. **Real-time Data Processing:** EventBridge can facilitate real-time data processing by ingesting events from various sources and processing them immediately. For example, you can use EventBridge to ingest streaming data from IoT devices, analyze it with Lambda functions, and take actions based on the analysis results.
3. **Alerting and Monitoring:** EventBridge can be used for alerting and monitoring purposes by reacting to specific events and triggering notifications or actions. For instance, you can create rules to monitor application logs for errors and trigger alerts to notify DevOps teams for immediate action.
4. **Scheduled Tasks:** EventBridge can schedule tasks by generating events on a predefined schedule. This can be useful for tasks like periodic data backups, system maintenance, or executing batch jobs at specific intervals.

5. **Cross-account Integration:** EventBridge supports cross-account event delivery, allowing you to share events across different AWS accounts within your organization. This can be beneficial for scenarios where multiple teams or applications need to react to the same set of events.
6. **Third-party Integration:** EventBridge can integrate with third-party SaaS applications and services, enabling you to build event-driven architectures across hybrid environments. For example, you can integrate events from third-party applications like Zendesk or Salesforce with your AWS environment to automate business processes.
7. **Event-driven Microservices:** EventBridge can facilitate the development of event-driven microservices architectures where services communicate asynchronously through events. This decouples components and allows for better scalability and flexibility in distributed systems.
8. **Audit and Compliance:** EventBridge can help with audit and compliance requirements by capturing and logging events related to system activities. You can use EventBridge to centralize event logs from various sources and integrate with auditing tools for compliance reporting and analysis.

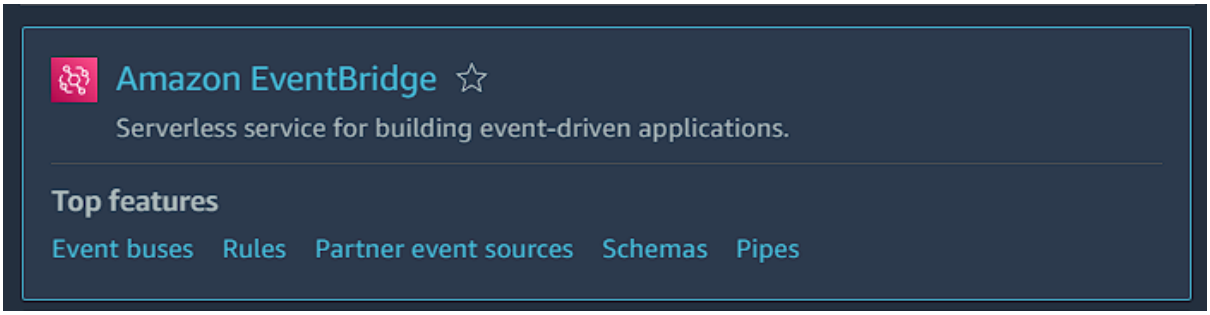
To begin with the Lab:

1. Login to AWS Console. Then Navigate to EC2 and create an instance based on anything.
2. Then navigate to Lambda and create a function with runtime environment as Python.
3. Once your function is created then you need to paste this code. After that deploy your code.

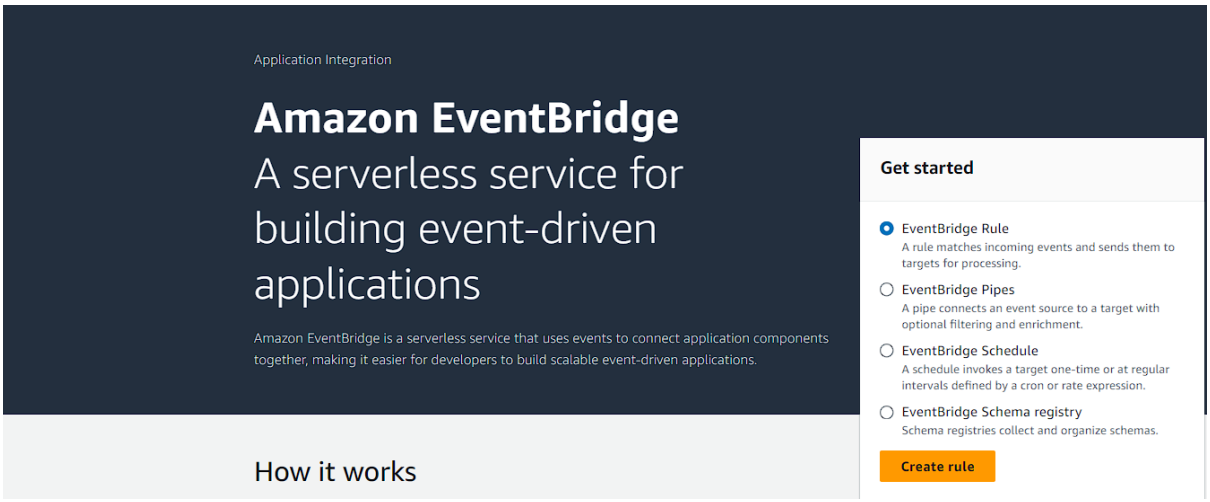
```
import json

def lambda_handler(event, context):
    print(json.dumps(event))

# If you're using Python 3.7+ AWS Lambda runtime, you might not need the context argument.
# You can adjust the function definition accordingly:
# def handler(event):
#     print(json.dumps(event))
```



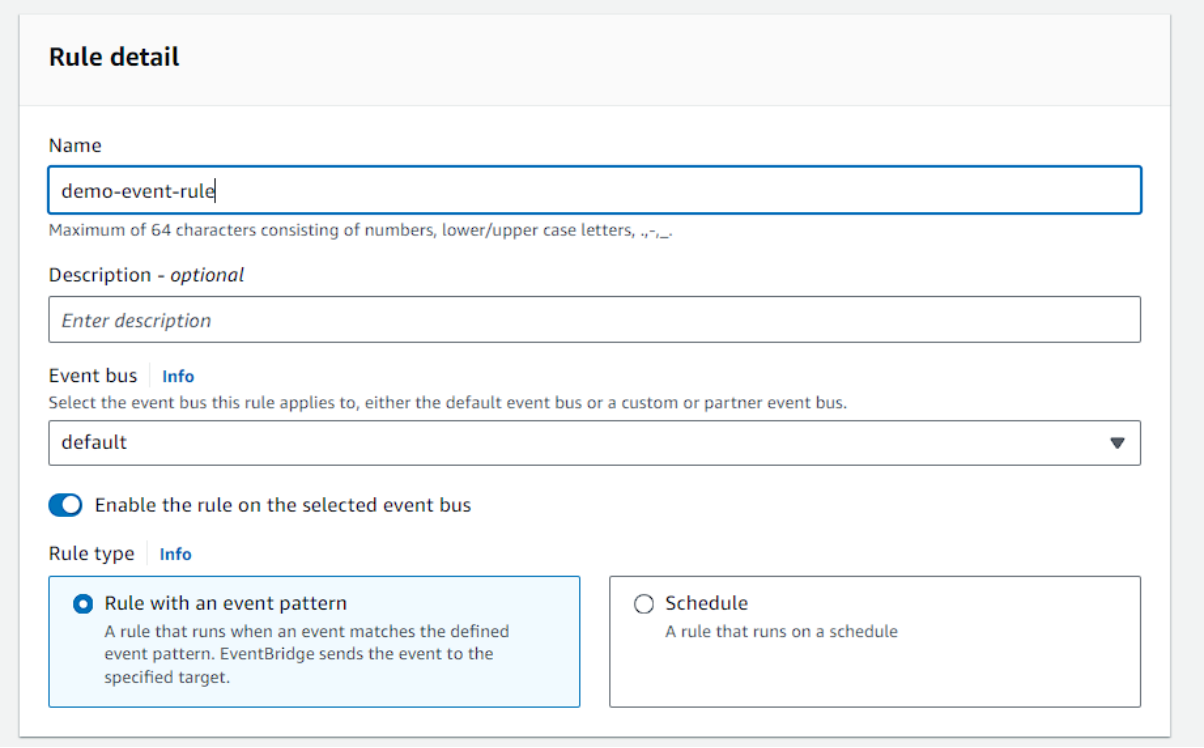
4. Now you need to navigate to EventBridge. Choose this service accordingly.



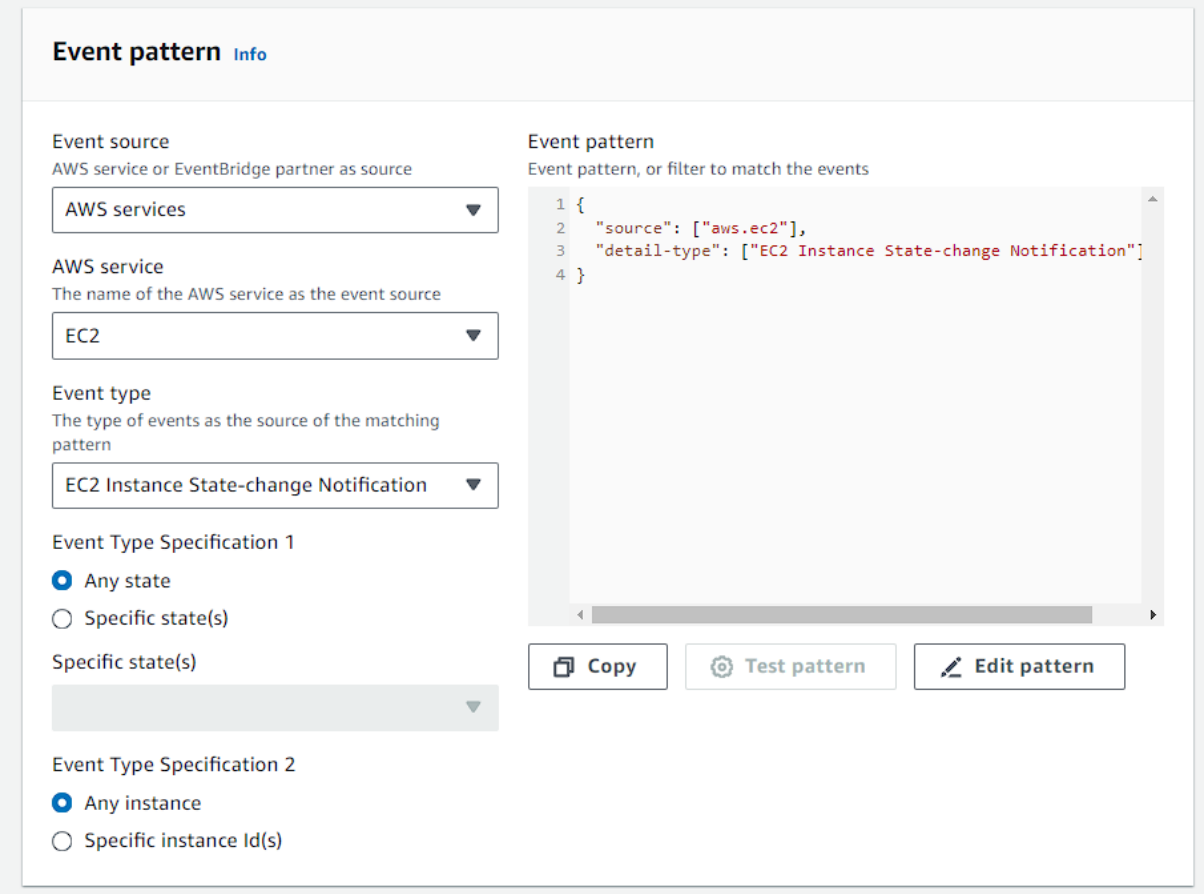
5. On the dashboard page of EventBridge select event bridge rule and click on create rule.



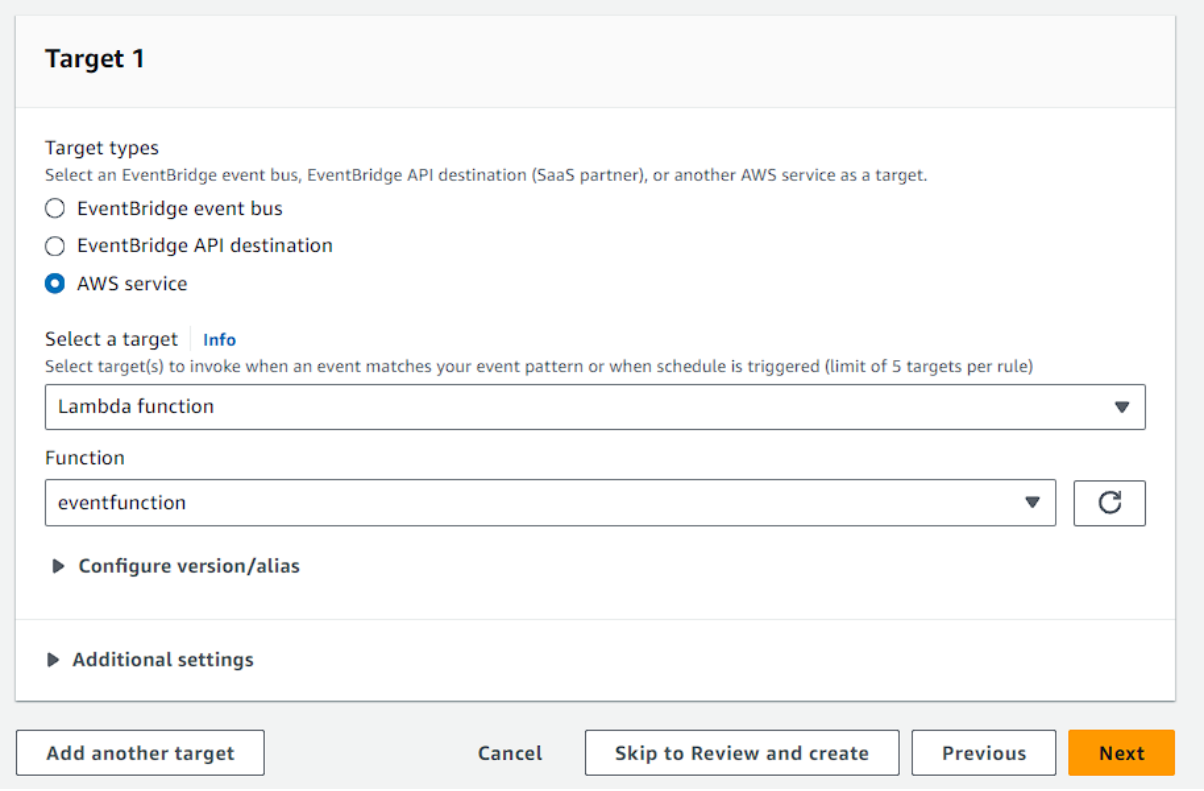
6. After that give it a name then go to next page.

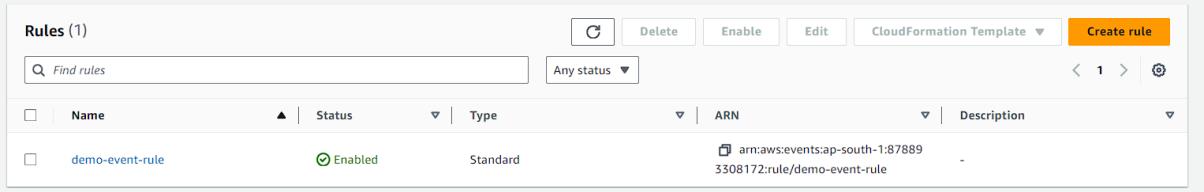


7. On the next page keep everything to default and scroll down to bottom. There in the event pattern, select event source as AWS Service.
8. In AWS Service choose EC2. Then in the event type choose EC2 state change notification.
9. After that click on next.

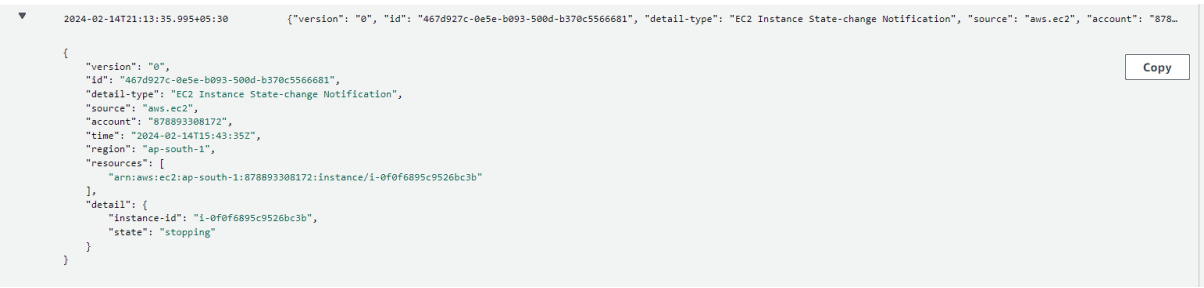


10. Now in the target you need to choose one.
11. For that select your target as Lambda function then chooses your lambda function.
12. Afterwards go to review page and create you event rule.





13. Once all it is done now go to EC2 and change the state of your instance. Set it to stop.
14. And once the instance is stopped then go to CloudWatch, there you need to open the log groups for your lambda function.
15. There you can see the notification in the form of logs.
16. In this log it is saying that instant state is changed to stopping.



17. And if you will wait for some time, you will see that another log has been generated for instance stopped.