**Cybersecurity 401**

**Module 4 - Cloud Security**

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

## **Lab 19 - Cloud Detective Controls**

## 

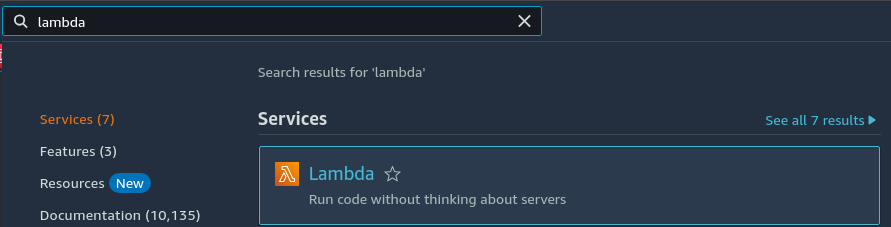
**| Rodrigo Brasil 12/2023 |**

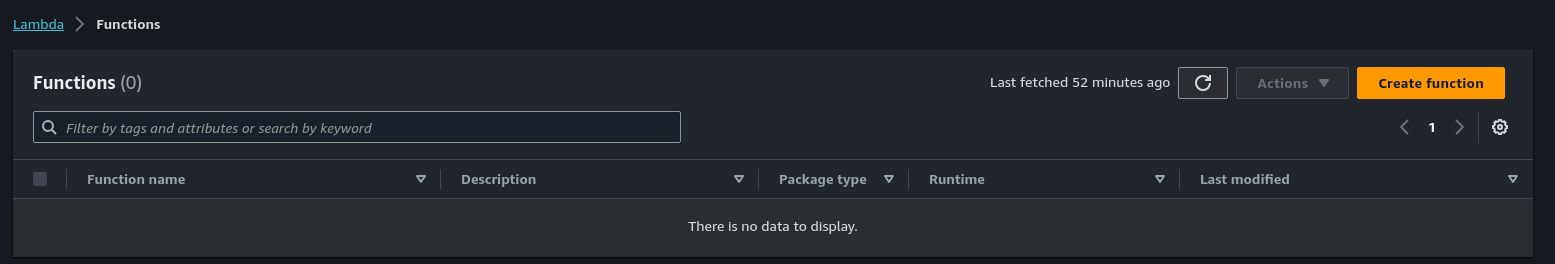
## **Tasks**

### **Part 1: Create an AWS Lambda Function**

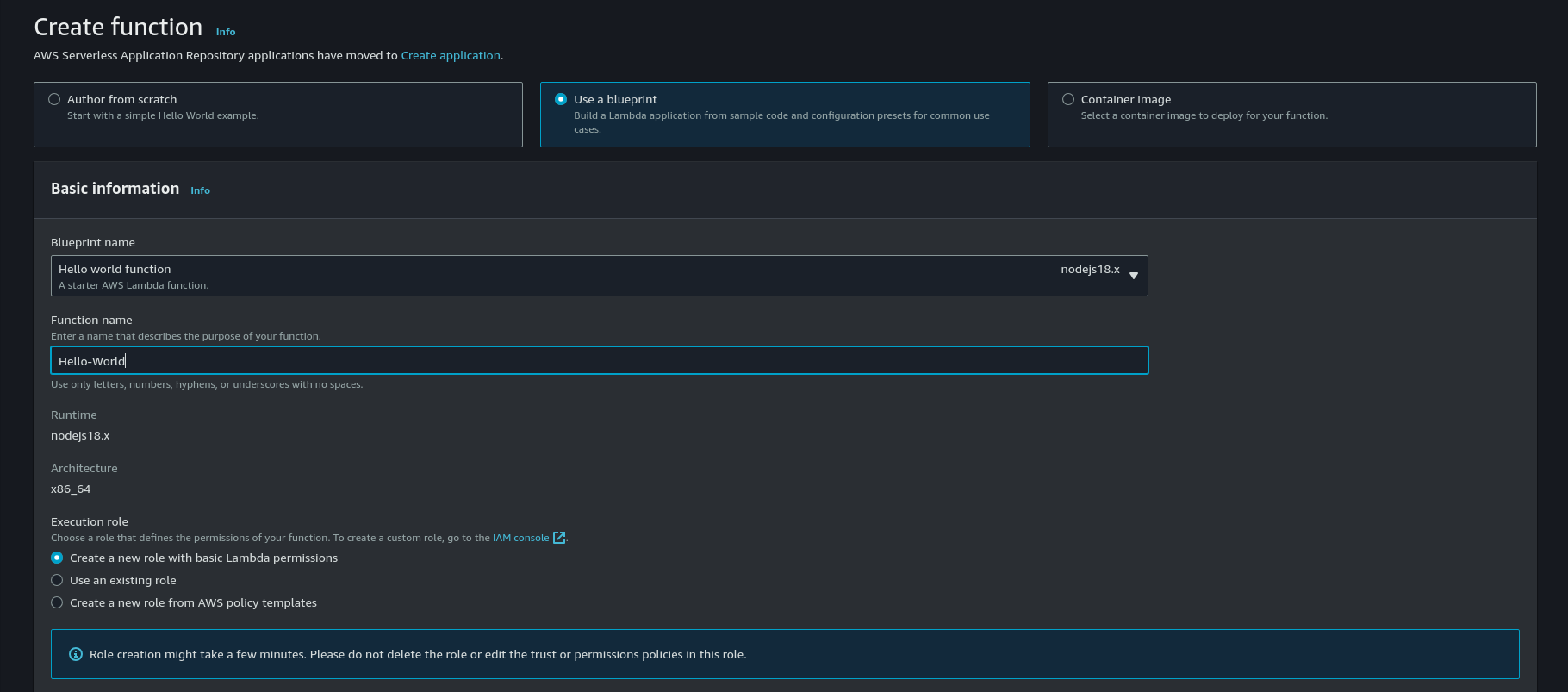
In the previous class we triggered Events with CloudWatch. **AWS Lambda** is an important element in automating responses to security events in your cloud environment, with its support for serverless functions. For example, we can use Lambda functions to raise alerts in response to certain conditions detected by AWS. For now, let’s just deploy and test a simple Lambda for the first time.

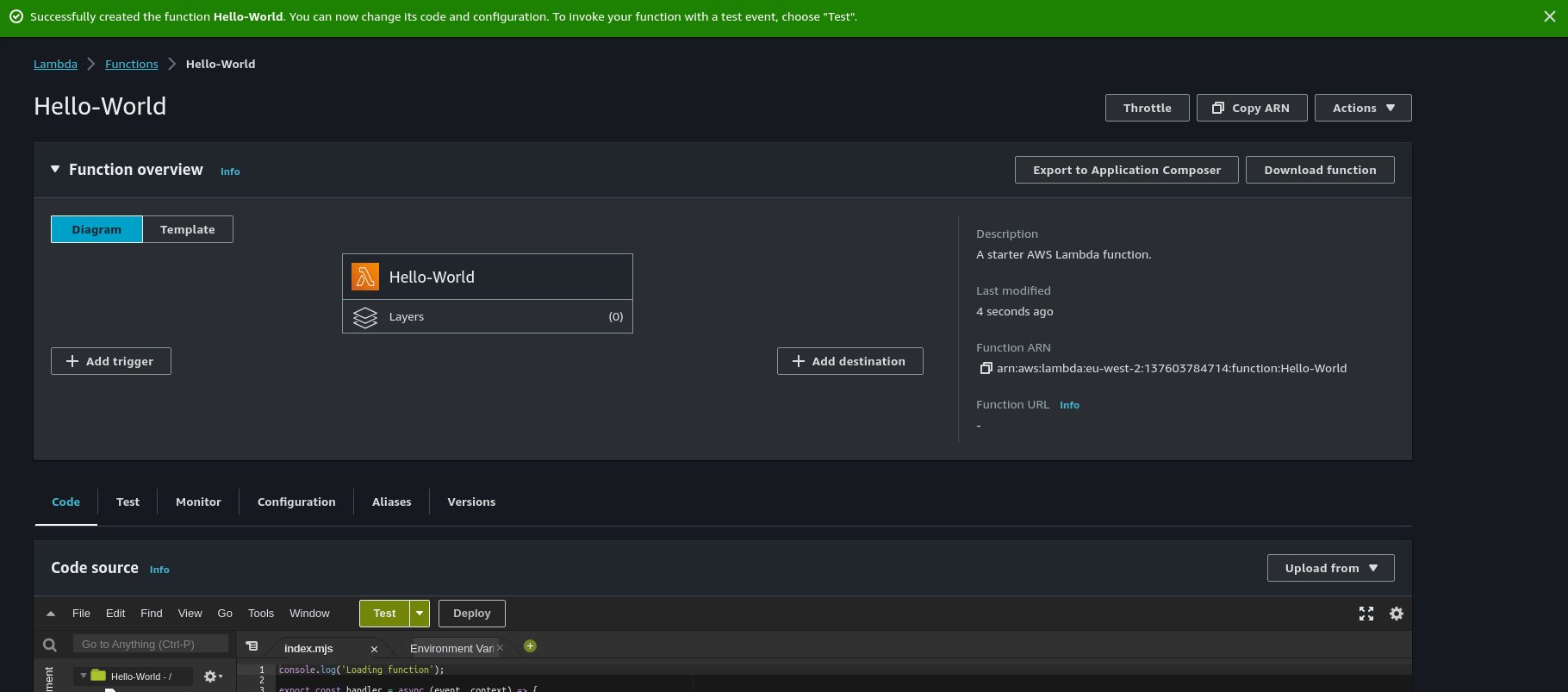
* In AWS Management Console, type “Lambda” into the search and access the Lambda service dashboard.





* Create an AWS Lambda function from a blueprint named “hello-world-python”





* Before you can test your new Lambda function, you’ll need to create a test event if you don’t have an existing event to work with. Name it “testevent1” and save it.
* Run a test of your Lambda function, hello-world-python, using **testevent1**.
* Copy and paste the log output into your submission.
* Locate the corresponding CloudWatch log group and paste the ARN of this log group into your submission.
* In CloudWatch, click the “Log Stream” within this log group. Keeping this open, execute your Lambda function again with a test. Refresh the log stream. What do you see?

This should give you a basic sense of how Lambda functions are triggered by events, and how CloudWatch keeps a close eye on said log stream. As a stretch goal, consider taking the next leap into [event source mapping](https://docs.aws.amazon.com/lambda/latest/dg/invocation-eventsourcemapping.html). A cloud security professional will want certain events to trigger Lambda-based alerts.

Now that we’ve tinkered with automated responses to events in the cloud, let’s take a look at threat detection with Amazon GuardDuty.

### **Part 2: Enable Amazon GuardDuty**

* Open the Amazon GuardDuty Management Console.
* Enable GuardDuty.
* In the Setting section, generate sample findings.
* Review the findings and find one that looks interesting to you. Select a high priority (red triangle) finding to analyze.
  + Identify the Finding ID value. In GuardDuty Findings, filter your view to that specific Finding ID and grab a screenshot.
  + Download the raw JSON of this finding and paste it into a shareable [GitHub gist](https://gist.github.com/). Link to the gist in your submission. Take a moment to analyze the JSON for useful details as to what may have occurred.
  + View the “Full description” and “Remediation recommendations” for this event and explain in your own words what took place, and how the company should remediate.
  + What resource is being targeted (being sample data it won’t necessarily exist on your cloud)?
  + What is the action type and source IP address?
  + Review the networking information on this finding; what does this tell you about what happened?
  + Is there a MITRE ATT&CK® TTP we could map this to?

### **Part 3: Log Analysis with AWS Security Hub**

* Open the AWS Security Hub Management Console.
* Enable Security Hub. Select *CIS* and *AWS Foundational* security standards.
* In the Security Standards menu, check account compliance.
* Confirm in Findings that the sample findings from GuardDuty were also reported in AWS Security Hub.
  + What details were provided for these findings?
* Navigate the Insights page.
  + What instance had the most findings?
  + What security issues are there?
  + What should the company address first and why?

### **Part 4: Wrap Up**

* GuardDuty can be disabled from GuardDuty Console > Settings > Disable GuardDuty.