**Pre-Requisites:**

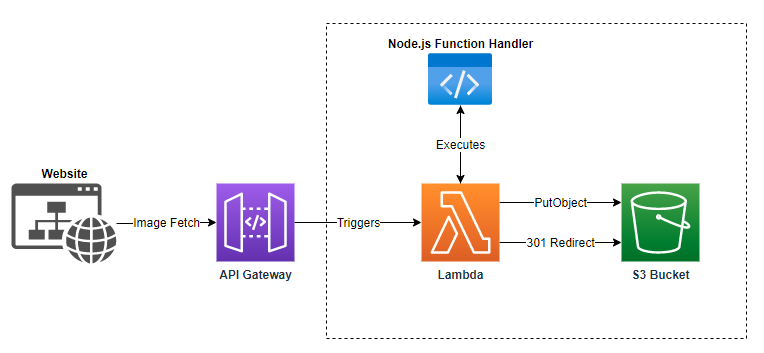
* Terraform
* An AWS account
* AWS CLI to be installed
* Docker (Optional)

**Tools we have used:**

* Terraform
* AWS cloud
* Docker

**Services used in AWS:**

* **S3**
* **Lambda**
* **IAM**
* **Api Gateway**

****

**In this project, we are going to deploy the Node js image resizing app. For that we are using terraform, to automate the deployment and the automation.**

**Overview:**

* For CI CD, we can use jenkins. When the code change happened in the nodejs app, it will trigger the deployment. So that, we can integrate terraform with Jenkins.
* It will deploy the latest version of the app into the AWS lambda.
* AWS lambda will be connected to the API GATEWAY. Api gateway will give us the public endpoints to access the api from outside.
* Whenever the API GW url is triggered, the lambda will convert the image into the particular size based on the request and it will resize the image and upload it back to the s3 for our reference.

**Step 1:**

**Create the zip file and upload it into the s3 bucket.**

**data "archive\_file" "lambda\_image\_resizer" {**

**type = "zip"**

**source\_dir = "${path.module}/image-resizer"**

**output\_path = "${path.module}/image-resizer.zip"**

**}**

**resource "aws\_s3\_bucket" "lambda\_image\_resizer" {**

**bucket = "${var.bucket\_name}"**

**acl = "${var.acl\_value}"**

**}**

**resource "aws\_s3\_object" "lambda\_image\_resizer" {**

**bucket = aws\_s3\_bucket.lambda\_image\_resizer.id**

**key = "image-resizer.zip"**

**source = data.archive\_file.lambda\_image\_resizer.output\_path**

**etag = filemd5(data.archive\_file.lambda\_image\_resizer.output\_path)**

**}**

**Step2:**

**Create the lambda function**

**resource "aws\_lambda\_function" "image\_resizer" {**

**function\_name = "image\_resizer"**

**s3\_bucket = aws\_s3\_bucket.lambda\_image\_resizer.id**

**s3\_key = aws\_s3\_object.lambda\_image\_resizer.key**

**runtime = "nodejs12.x"**

**handler = "hello.handler"**

**source\_code\_hash = data.archive\_file.lambda\_image\_resizer.output\_base64sha256**

**role = aws\_iam\_role.iam\_img\_resize\_role.arn**

**}**

**resource "aws\_cloudwatch\_log\_group" "image\_resizer" {**

**name = "/aws/lambda/${aws\_lambda\_function.image\_resizer.function\_name}"**

**retention\_in\_days = 30**

**}**

**Step3:**

**Create HTTP api with APIGATEWAY**

**resource "aws\_apigatewayv2\_api" "lambda" {**

**name = "serverless\_lambda\_gw"**

**protocol\_type = "HTTP"**

**}**

**resource "aws\_apigatewayv2\_stage" "lambda" {**

**api\_id = aws\_apigatewayv2\_api.lambda.id**

**name = "serverless\_lambda\_stage"**

**auto\_deploy = true**

**access\_log\_settings {**

**destination\_arn = aws\_cloudwatch\_log\_group.api\_gw.arn**

**format = jsonencode({**

**requestId = "$context.requestId"**

**sourceIp = "$context.identity.sourceIp"**

**requestTime = "$context.requestTime"**

**protocol = "$context.protocol"**

**httpMethod = "$context.httpMethod"**

**resourcePath = "$context.resourcePath"**

**routeKey = "$context.routeKey"**

**status = "$context.status"**

**responseLength = "$context.responseLength"**

**integrationErrorMessage = "$context.integrationErrorMessage"**

**}**

**)**

**}**

**}**

**resource "aws\_apigatewayv2\_integration" "image\_resizer" {**

**api\_id = aws\_apigatewayv2\_api.lambda.id**

**integration\_uri = aws\_lambda\_function.image\_resizer.invoke\_arn**

**integration\_type = "AWS\_PROXY"**

**integration\_method = "POST"**

**}**

**resource "aws\_apigatewayv2\_route" "image\_resizer" {**

**api\_id = aws\_apigatewayv2\_api.lambda.id**

**route\_key = "POST /image"**

**target = "integrations/${aws\_apigatewayv2\_integration.image\_resizer.id}"**

**}**

**resource "aws\_cloudwatch\_log\_group" "api\_gw" {**

**name = "/aws/api\_gw/${aws\_apigatewayv2\_api.lambda.name}"**

**retention\_in\_days = 30**

**}**

**resource "aws\_lambda\_permission" "api\_gw" {**

**statement\_id = "AllowExecutionFromAPIGateway"**

**action = "lambda:InvokeFunction"**

**function\_name = aws\_lambda\_function.image\_resizer.function\_name**

**principal = "apigateway.amazonaws.com"**

**source\_arn = "${aws\_apigatewayv2\_api.lambda.execution\_arn}/\*/\*"**

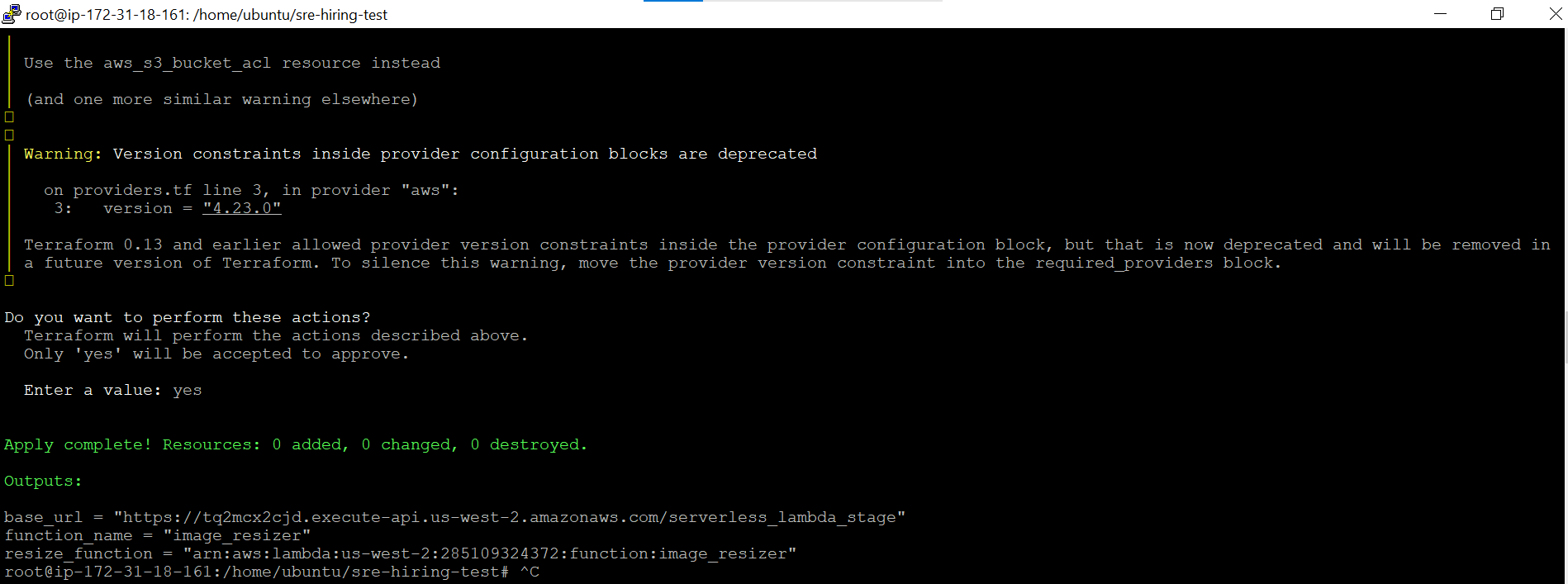
**}**

**Step 4:**

**Apply the code**

**Terraform apply**

**Sample Output:**

****