1. api gateway /
2. lambda
3. dynamodb
4. iam to secure your lambda and api gateway
5. s3 static web hosting
6. cloudwatch for debugging
7. docker/ecs/fargate

and aws amplify / aws cloud front

<https://www.linkedin.com/pulse/getting-started-aws-frontend-engineers-ricardo-guillen-i-/> aws topics for FE

<https://www.youtube.com/watch?v=RnFowJ130pc> for install npm aws by upload

<https://www.youtube.com/watch?v=flJ_cfQ53vQ> using layers

1 AWS Amplify (gen 1 and gen2)

Used to build and host website.

1. AWS Amplify :

--- go to console search bar and search for amplify  
A screenshot of a computer

Description automatically generated

---- click on create App  
A screenshot of a computer

Description automatically generated

----click on without GitHub. For deploy any code from local machine  
A screenshot of a computer

Description automatically generated

----- put your amplify app name and branch then upload your source code in zip format🡪save and deploy  
A screenshot of a computer

Description automatically generated

After save and deploy code will be deployed and share a link where we can check the deployed code.

--- how to delete the existing amplify app  
A screenshot of a computer

Description automatically generated

1. **Lamda Function**

A screenshot of a computer

Description automatically generated

----- then create a function----

A screenshot of a computer

Description automatically generated

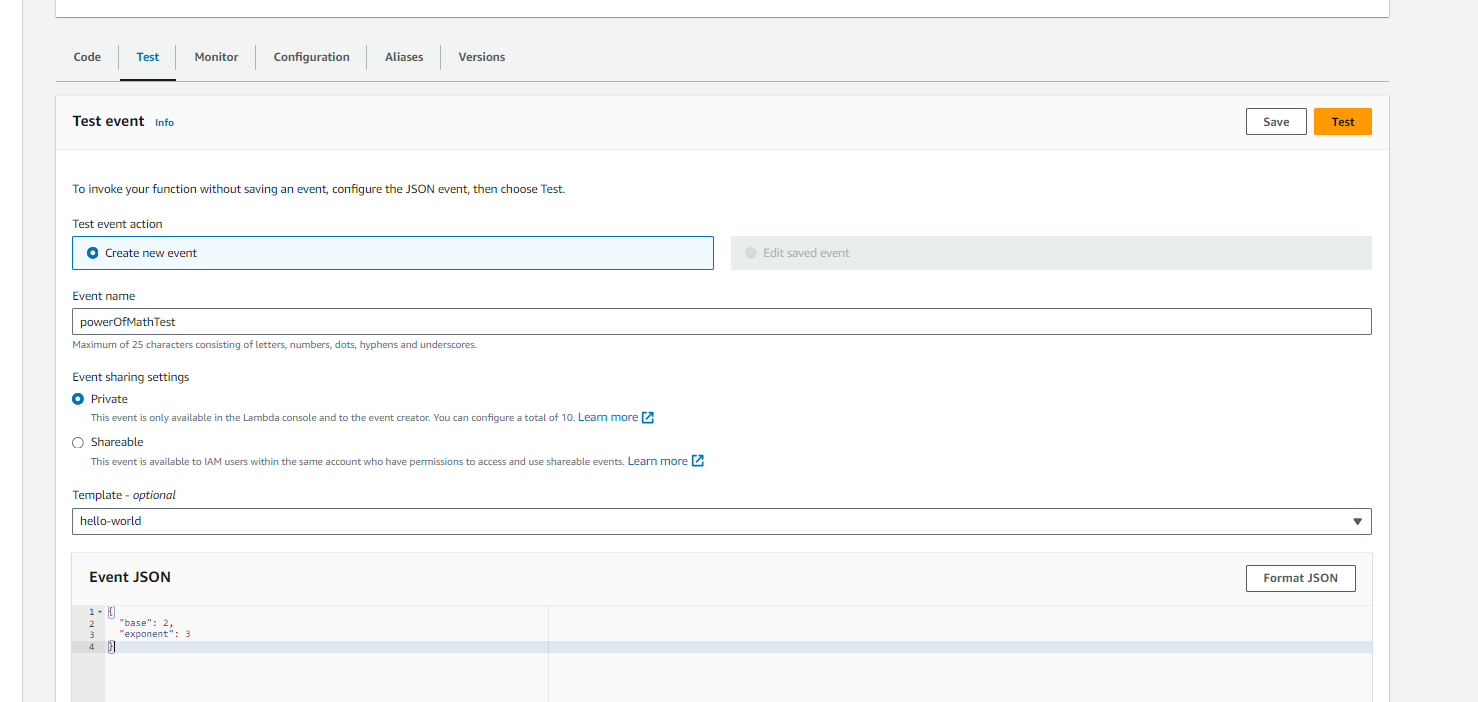
---- we will get a text editor where we can write a node js code

* Handler will be a default function name
* We can change as per our requirement
* Then we can deploy our changes

--- we can also upload a zip file from local or from S3 bucket

A screenshot of a computer

Description automatically generated

--- we can create a test environment for testing of our Lamda function. We can test the function output  


------ **how to give the permission to communicate lamda function with the DynamoDB**

A screenshot of a computer

Description automatically generated

Go to the configuration tab in lamda function -> lamdaFunctionName-role

------- it will open in new Tab window where we need to mapped the policy to connect with DynamoDB

A screenshot of a computer

Description automatically generated

---- it will open a page

A screenshot of a computer

Description automatically generated

{

"Version": "2012-10-17",

"Statement": [

{

"Sid": "VisualEditor0",

"Effect": "Allow",

"Action": [

"dynamodb:PutItem",

"dynamodb:DeleteItem",

"dynamodb:GetItem",

"dynamodb:Scan",

"dynamodb:Query",

"dynamodb:UpdateItem"

],

"Resource": "YOUR-TABLE-ARN" **update here your DynamoDB ARN number**

}

]

}

----- now we can write the DB queries in our lamda function

Example

<https://github.com/vinaycoder/aws/blob/main/index.mjs>

A screen shot of a computer program

Description automatically generated

1. **API gateway**

----- search **API gateway** in search Box

A screenshot of a computer

Description automatically generated

Then click on create new api -> rest api build

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

---- select method and click on create method

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

**---- after creatin method we need to enable CORS**

A
 screenshot of a computer

A screenshot of a computer

Description automatically generated

---- deploy the api

A screenshot of a computer

Description automatically generated

A screenshot of a stage page

Description automatically generated

----- we will the api link

A screenshot of a computer

Description automatically generated

---- test our api end point

A screenshot of a computer

Description automatically generated

1. **DynamoDB**

---- search DynamoDB in search Box

A screenshot of a computer

Description automatically generated

----- create table A screenshot of a computer

Description automatically generated

After creating the table ---- we need ARN number for using this table

A screenshot of a computer

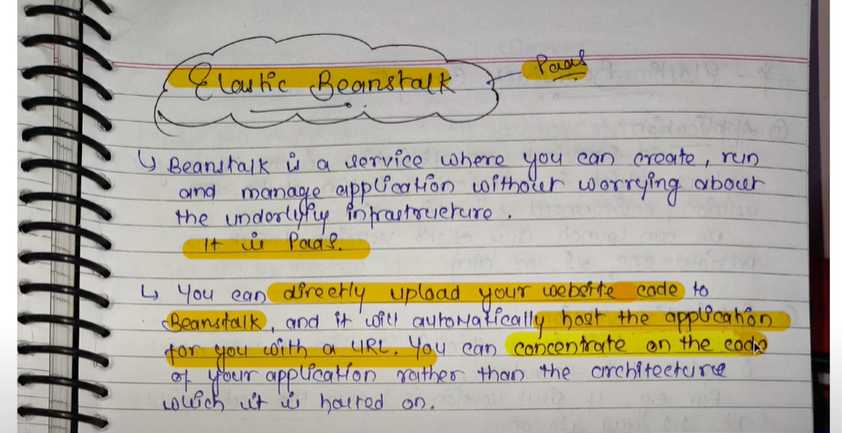
Description automatically generated

1. **How to install npm package in lamda function**

There are two ways for this

1. Create a sample node project and zip and upload  
   --- mkdir my-lambda-function   
   --- cd my-lambda-function  
   --- npm init -y  
   --- npm install aws-sdk  
   --- then create a index.js or .mjs file for writing the lamda function on local  
   --- zip that folder. And upload in lamda.
2. Using the layers

**Elastic Beanstalk**



* [Workers](https://www.sitepoint.com/node-js-multithreading/#nodejsworkerthreads)
* [Child processes](https://www.sitepoint.com/node-js-multithreading/#nodejschildprocesses)
* [Clustering](https://www.sitepoint.com/node-js-multithreading/#nodejsclustering)
* [Process Managers](https://www.sitepoint.com/node-js-multithreading/#processmanagers)
* [Containers](https://www.sitepoint.com/node-js-multithreading/#containerorchestration)

Node.js has two types of threads  
. The one **Event Loop** thread (aka the main thread).

. The **Worker Pool** (aka threadpool) threads.

Node js workers threads

[Worker threads](https://nodejs.org/dist/latest/docs/api/worker_threads.html) are the Node.js equivalent of [web workers](https://developer.mozilla.org/docs/Web/API/Web_Workers_API/Using_web_workers). The main thread passes data to another script which (asynchronously) processes it on a separate thread. The main thread continues to run and runs a callback event when the worker has completed its work.