[Mohamed Raafat Abdel Aziz]

THEORETICAL

1. What are the benefits of using Elastic BeanStalk over just EC2?

I finished my search

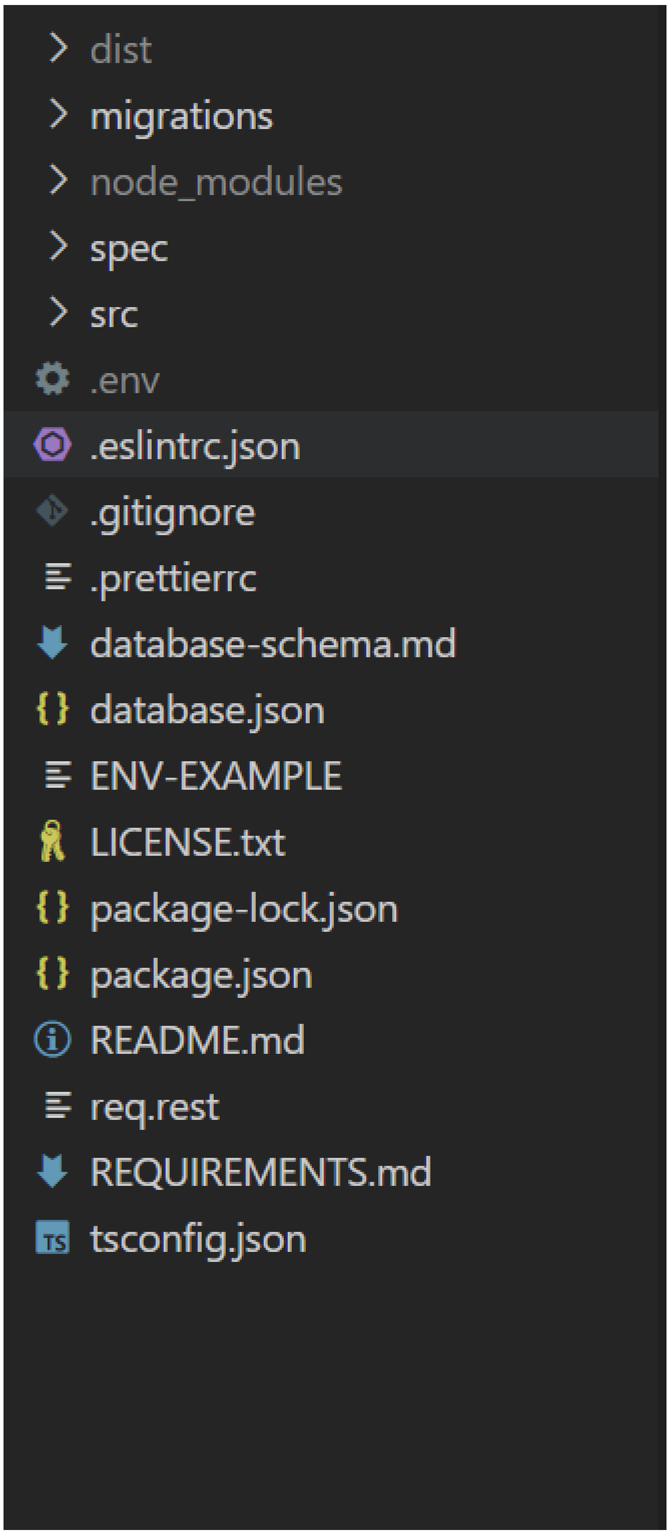
Elastic beanstalk gives you the ability to create an environment that contains a number of ec2 instances, an optional database as well as other aws services such as elastic load balancer, auto-scaling group, security group, eb will manage these things for you when you want to create an new instance, that why it is called an orchestiration service where helps you setup other services next to ec2 and comes at no cost.
where as ec2 is just a service that gives you the ability to create servers (aka. instances) in aws.

PRACTICAL

1. Configure a TypeScript app with the correct EB settings and build/start scripts

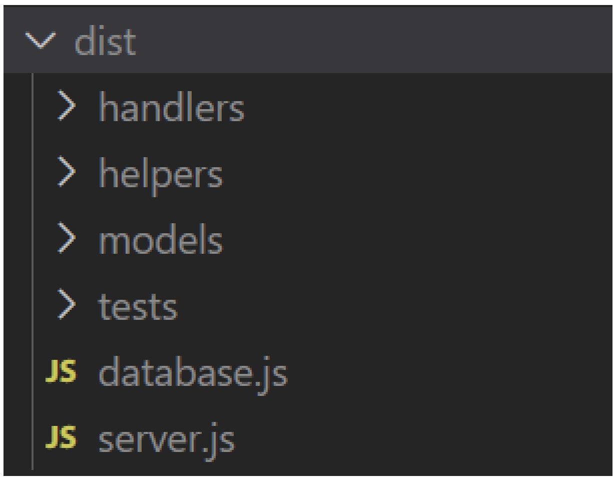
I finished my search

well, first things first i need to determine what will go on the server



```
> dist
> migrations
> node_modules
> spec
> src
.env
.eslintrc.json
.gitignore
.prettierrc
database-schema.md
database.json
ENV-EXAMPLE
LICENSE.txt
package-lock.json
package.json
README.md
req.rest
REQUIREMENTS.md
tsconfig.json
```

by looking at my file structure all what i need is the dist folder.

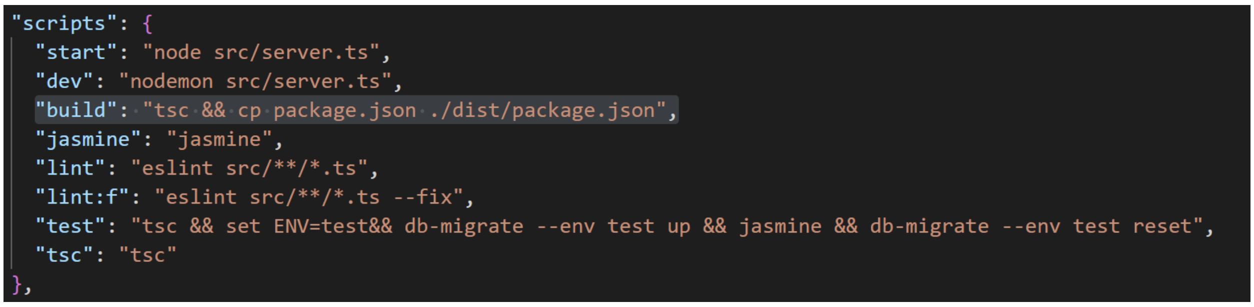


but that dist is just the transpiled src folder files, it doesn't have package.json

so the first thing i need to do is to move package.json inside the dist folder, this can be done using the command

`cp package.json ./dist/package.json`

So now my build script looks something like this

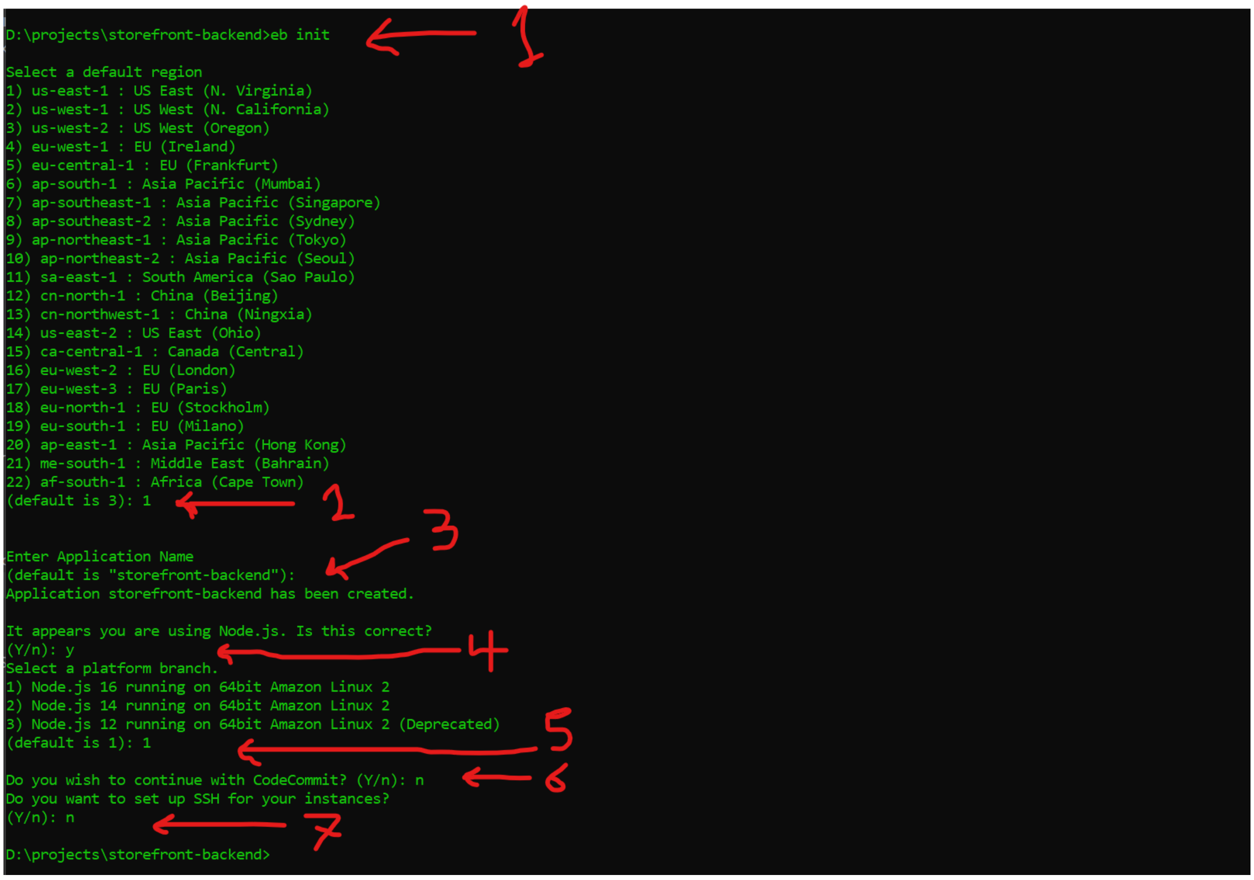


```
"scripts": {
  "start": "node src/server.ts",
  "dev": "nodemon src/server.ts",
  "build": "tsc && cp package.json ./dist/package.json",
  "jasmine": "jasmine",
  "lint": "eslint src/**/*.ts",
  "lint:f": "eslint src/**/*.ts --fix",
  "test": "tsc && set ENV=test&& db-migrate --env test up && jasmine && db-migrate --env test reset",
  "tsc": "tsc"
},
```

now whenever i run the build command, package.json now is copied to the dist folder.

now i need to initialize an new eb, and always remember the 3 sequences we will follow our golder rule.

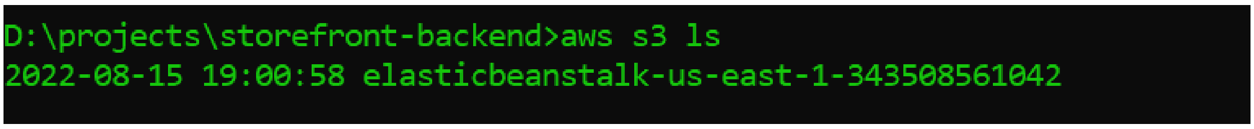
eb init -> eb create -> eb deploy



```
D:\projects\storefront-backend>eb init

Select a default region
1) us-east-1 : US East (N. Virginia)
2) us-west-1 : US West (N. California)
3) us-west-2 : US West (Oregon)
4) eu-west-1 : EU (Ireland)
5) eu-central-1 : EU (Frankfurt)
6) ap-south-1 : Asia Pacific (Mumbai)
7) ap-southeast-1 : Asia Pacific (Singapore)
8) ap-southeast-2 : Asia Pacific (Sydney)
9) ap-northeast-1 : Asia Pacific (Tokyo)
10) ap-northeast-2 : Asia Pacific (Seoul)
11) sa-east-1 : South America (Sao Paulo)
12) cn-north-1 : China (Beijing)
13) cn-northwest-1 : China (Ningxia)
14) us-east-2 : US East (Ohio)
15) ca-central-1 : Canada (Central)
16) eu-west-2 : EU (London)
17) eu-west-3 : EU (Paris)
18) eu-north-1 : EU (Stockholm)
19) eu-south-1 : EU (Milano)
20) ap-east-1 : Asia Pacific (Hong Kong)
21) me-south-1 : Middle East (Bahrain)
22) af-south-1 : Africa (Cape Town)
(default is 3): 1


Enter Application Name
(default is "storefront-backend"):
Application storefront-backend has been created.

It appears you are using Node.js. Is this correct?
(Y/n): y
Select a platform branch.
1) Node.js 16 running on 64bit Amazon Linux 2
2) Node.js 14 running on 64bit Amazon Linux 2
3) Node.js 12 running on 64bit Amazon Linux 2 (Deprecated)
(default is 1): 1

Do you wish to continue with CodeCommit? (Y/n): n
Do you want to set up SSH for your instances?
(Y/n): n

D:\projects\storefront-backend>
```

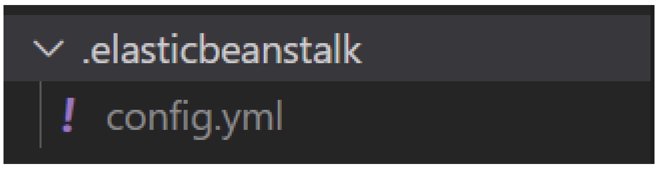
1)i first started with eb init command.
2) then choose north virginia as a region.
3)then accepted the default name it suggested
4)i confirmed that my project uses node
5)i choose my node version, if you don't know yours run the command `node -v`
6)i choose no for the code commit
   - code commit is a source control service that hosts private git repos in aws, imagine it as aws's github
7)i choose no for the ssh option
   - ssh is a communication protocol that allows you to remotely connect on a server

- here we can realize what is meant by orchestiration service, all what i did until now is initializing eb but if i run the command `aws s3 ls`, we will reallize that eb automatically created a bucket for us

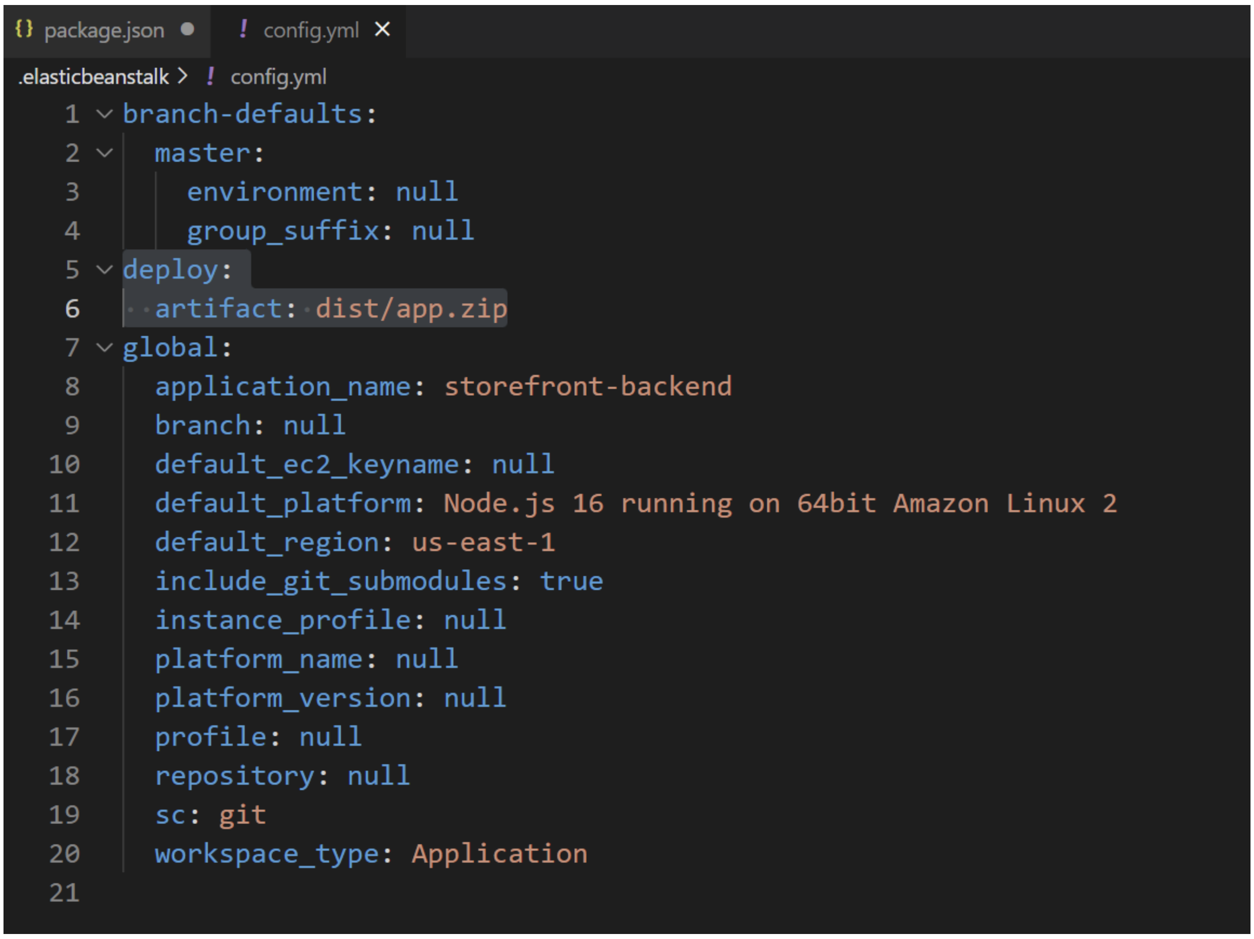


```
D:\projects\storefront-backend>aws s3 ls
2022-08-15 19:00:58 elasticbeanstalk-us-east-1-343508561042
```

now if looked at my file structure again i will find a new folder called .elasticbeanstalk



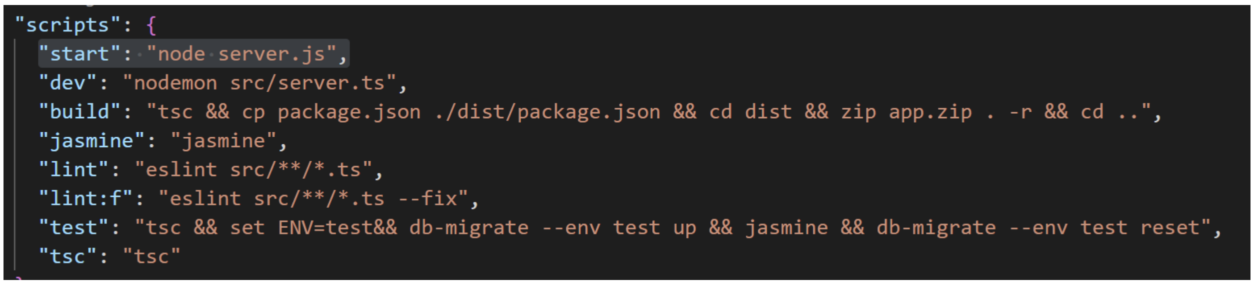
at this point we are ready to upload our code but we need to upload only the dist folder, here comes the use of our new .elasticbeanstalk folder using artifact



```yaml
branch-defaults:
  master:
    environment: null
    group_suffix: null
deploy:
  artifact: dist/app.zip
global:
  application_name: storefront-backend
  branch: null
  default_ec2_keyname: null
  default_platform: Node.js 16 running on 64bit Amazon Linux 2
  default_region: us-east-1
  include_git_submodules: true
  instance_profile: null
  platform_name: null
  platform_version: null
  profile: null
  repository: null
  sc: git
  workspace_type: Application

```

now eb will look only for app.zip file in dist folder to upload and will ignore the rest.

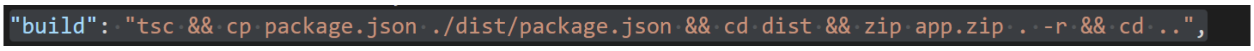
one important thing before we zip the dist folder is we need to modify the start command



```json
"scripts": {
  "start": "node server.js",
  "dev": "nodemon src/server.ts",
  "build": "tsc && cp package.json ./dist/package.json && cd dist && zip app.zip . -r && cd ..",
  "jasmine": "jasmine",
  "lint": "eslint src/**/*.ts",
  "lint:f": "eslint src/**/*.ts --fix",
  "test": "tsc && set ENV=test&& db-migrate --env test up && jasmine && db-migrate --env test reset",
  "tsc": "tsc"
```

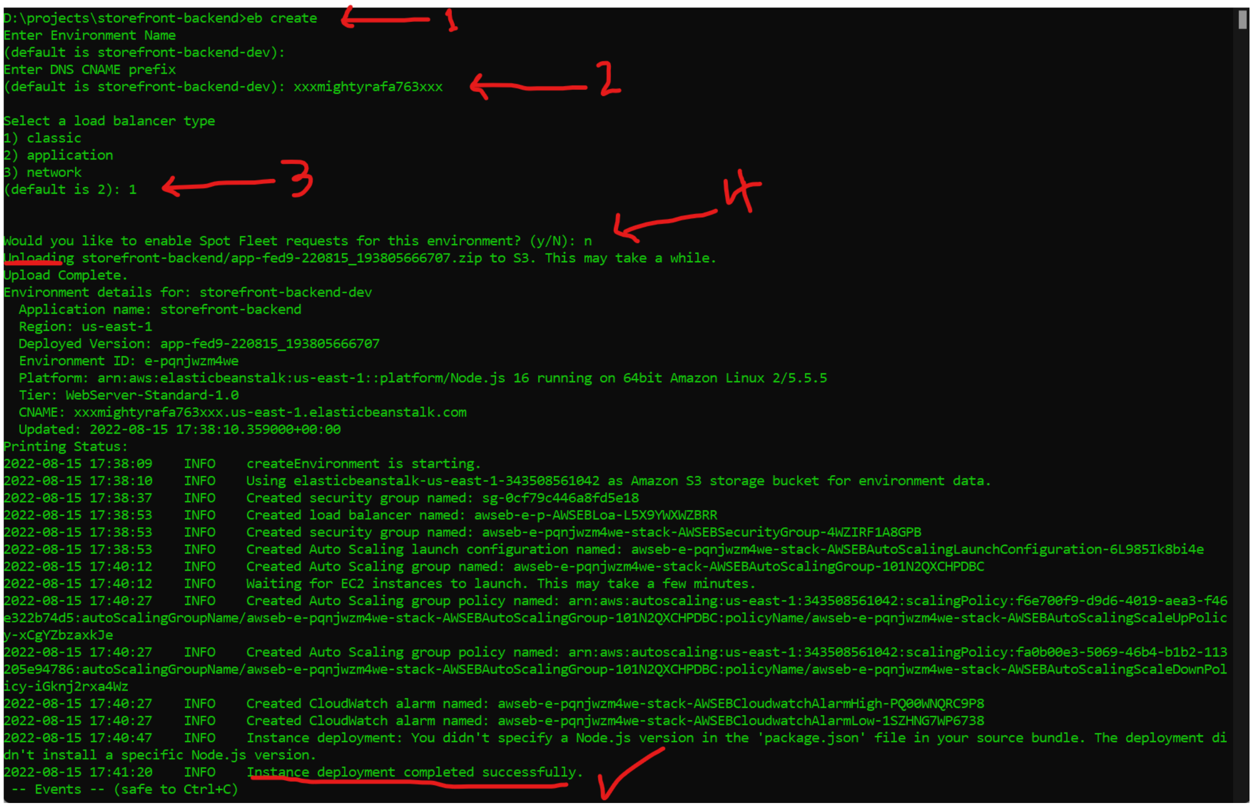
we now need to zip the dist folder contents, so we will go to our package.json and add the command `cd dist && zip app.zip . –r && cd ..` to the build script

so now our build script looks something like this



```json
"build": "tsc && cp package.json ./dist/package.json && cd dist && zip app.zip . -r && cd ..",
```

now if we run the `eb create` command we should see something like this

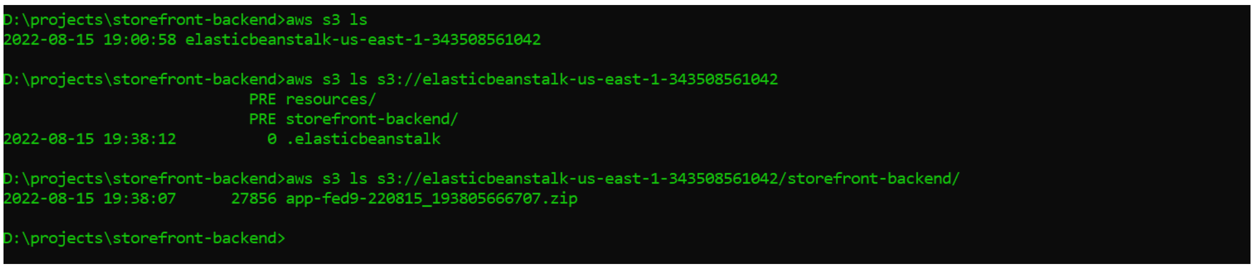


```
D:\projects\storefront-backend>eb create
Enter Environment Name
(default is storefront-backend-dev):
Enter DNS CNAME prefix
(default is storefront-backend-dev): xxxmightyrafa763xxx

Select a load balancer type
1) classic
2) application
3) network
(default is 2): 1


Would you like to enable Spot Fleet requests for this environment? (y/N): n
Uploading storefront-backend/app-fed9-220815_193805666707.zip to S3. This may take a while.
Upload Complete.
Environment details for: storefront-backend-dev
  Application name: storefront-backend
  Region: us-east-1
  Deployed Version: app-fed9-220815_193805666707
  Environment ID: e-pqnjwzm4we
  Platform: arn:aws:elasticbeanstalk:us-east-1::platform/Node.js 16 running on 64bit Amazon Linux 2/5.5.5
  Tier: WebServer-Standard-1.0
  CNAME: xxxmightyrafa763xxx.us-east-1.elasticbeanstalk.com
  Updated: 2022-08-15 17:38:10.359000+00:00
Printing Status:
2022-08-15 17:38:09    INFO    createEnvironment is starting.
2022-08-15 17:38:10    INFO    Using elasticbeanstalk-us-east-1-343508561042 as Amazon S3 storage bucket for environment data.
2022-08-15 17:38:37    INFO    Created security group named: sg-0cf79c446a8fd5e18
2022-08-15 17:38:53    INFO    Created load balancer named: awseb-e-p-AWSEBLoa-L5X9YWXWZBRR
2022-08-15 17:38:53    INFO    Created security group named: awseb-e-pqnjwzm4we-stack-AWSEBSecurityGroup-4WZIRF1A8GPB
2022-08-15 17:38:53    INFO    Created Auto Scaling launch configuration named: awseb-e-pqnjwzm4we-stack-AWSEBAutoScalingLaunchConfiguration-6L985Ik8bi4e
2022-08-15 17:40:12    INFO    Created Auto Scaling group named: awseb-e-pqnjwzm4we-stack-AWSEBAutoScalingGroup-101N2QXCHPDBC
2022-08-15 17:40:12    INFO    Waiting for EC2 instances to launch. This may take a few minutes.
2022-08-15 17:40:27    INFO    Created Auto Scaling group policy named: arn:aws:autoscaling:us-east-1:343508561042:scalingPolicy:f6e700f9-d9d6-4019-aea3-f46
e322b74d5:autoScalingGroupName/awseb-e-pqnjwzm4we-stack-AWSEBAutoScalingGroup-101N2QXCHPDBC:policyName/awseb-e-pqnjwzm4we-stack-AWSEBAutoScalingScaleUpPolic
y-xCgYZbzaxkJe
2022-08-15 17:40:27    INFO    Created Auto Scaling group policy named: arn:aws:autoscaling:us-east-1:343508561042:scalingPolicy:fa0b00e3-5069-46b4-b1b2-113
205e94786:autoScalingGroupName/awseb-e-pqnjwzm4we-stack-AWSEBAutoScalingGroup-101N2QXCHPDBC:policyName/awseb-e-pqnjwzm4we-stack-AWSEBAutoScalingScaleDownPol
icy-iGknj2rxa4Wz
2022-08-15 17:40:27    INFO    Created CloudWatch alarm named: awseb-e-pqnjwzm4we-stack-AWSEBCloudwatchAlarmHigh-PQ00WNQRC9P8
2022-08-15 17:40:27    INFO    Created CloudWatch alarm named: awseb-e-pqnjwzm4we-stack-AWSEBCloudwatchAlarmLow-1SZHNG7WP6738
2022-08-15 17:40:47    INFO    Instance deployment: You didn't specify a Node.js version in the 'package.json' file in your source bundle. The deployment di
dn't install a specific Node.js version.
2022-08-15 17:41:20    INFO    Instance deployment completed successfully.
 -- Events -- (safe to Ctrl+C)
```

1) ran the eb create command and accepted the default environment name
2) choose a custom subdomain (not mandatory)
3) selected the classic load balancer
4) choose no for the spot fleet
   - spot fleet is what scales our application up/down based on the incoming traffic

*now it is saying uploading so we know that it is uploading the zip file we made

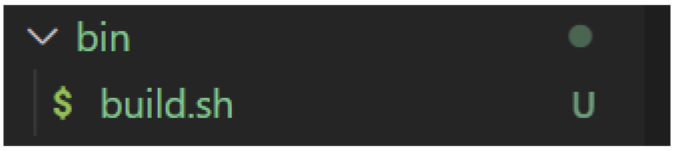
we can verify that by checking the s3 bucket content

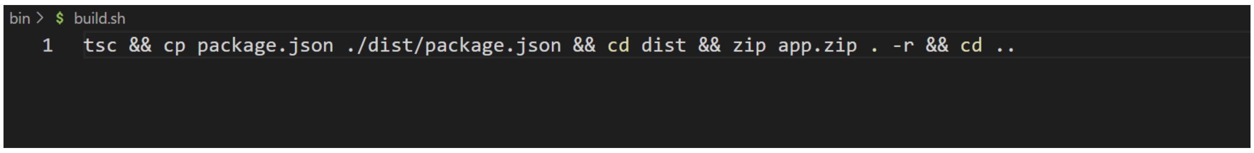


```
D:\projects\storefront-backend>aws s3 ls
2022-08-15 19:00:58 elasticbeanstalk-us-east-1-343508561042

D:\projects\storefront-backend>aws s3 ls s3://elasticbeanstalk-us-east-1-343508561042
                           PRE resources/
                           PRE storefront-backend/
2022-08-15 19:38:12          0 .elasticbeanstalk

D:\projects\storefront-backend>aws s3 ls s3://elasticbeanstalk-us-east-1-343508561042/storefront-backend/
2022-08-15 19:38:07      27856 app-fed9-220815_193805666707.zip

D:\projects\storefront-backend>
```

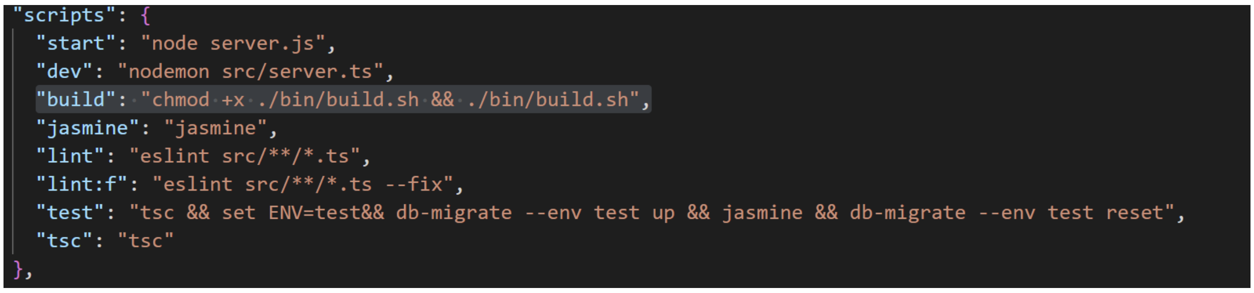
****

one thing i failed doing is moving the build script in and external .sh folder

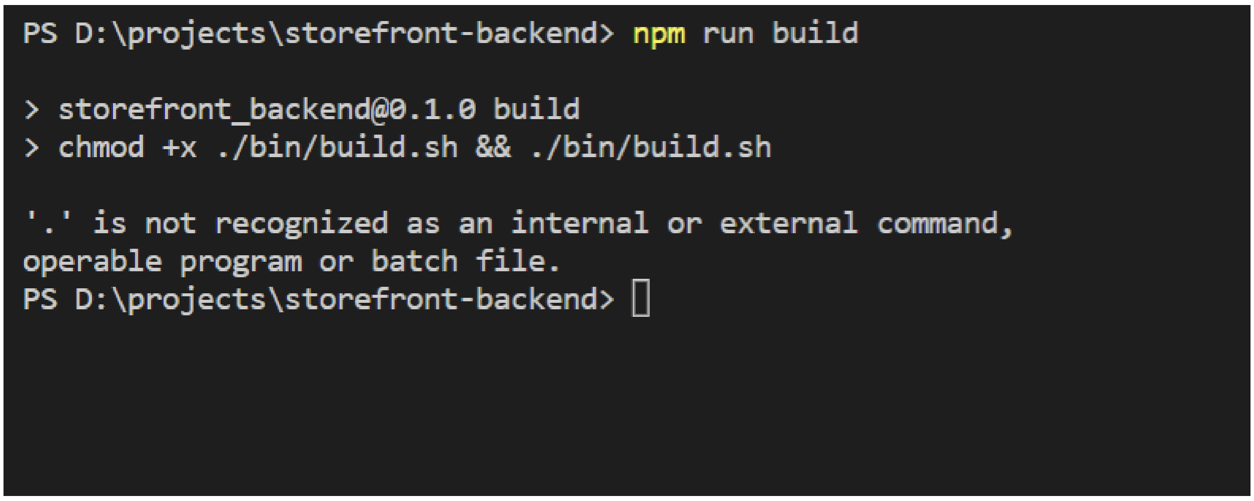




```
tsc && cp package.json ./dist/package.json && cd dist && zip app.zip . -r && cd ..
```



```
"scripts": {
  "start": "node server.js",
  "dev": "nodemon src/server.ts",
  "build": "chmod +x ./bin/build.sh && ./bin/build.sh",
  "jasmine": "jasmine",
  "lint": "eslint src/**/*.ts",
  "lint:f": "eslint src/**/*.ts --fix",
  "test": "tsc && set ENV=test&& db-migrate --env test up && jasmine && db-migrate --env test reset",
  "tsc": "tsc"
},
```



```
PS D:\projects\storefront-backend> npm run build

> storefront_backend@0.1.0 build
> chmod +x ./bin/build.sh && ./bin/build.sh

'.' is not recognized as an internal or external command,
operable program or batch file.
PS D:\projects\storefront-backend>
```

[Mohamed Aboarab]

THEORETICAL

1. What are the benefits of using Elastic BeanStalk over just EC2?

out of scope

add your search here …

PRACTICAL

1. Configure a TypeScript app with the correct EB settings and build/start scripts

I finished my search

A- in the aws eb console create a new env. with the intended env. name

for instance : ‘udagramapi-env’

B- use the `eb init` to start the elastic beanstalk cli and to create the configuration file:

assure the following configurations inside the file ‘.elasticbeanstalk/config.yml’

```
 branch-defaults:
 default:
   environment: Udagramapi-env
   group_suffix: null
environment-defaults:
 Udagramapi-env:
   branch: null
   repository: null
 udagram-app-env:
   branch: null
   repository: null
global:
 application_name: udagram-api
 branch: null
 default_ec2_keyname: null
 default_platform: Node.js 16
 default_region: us-east-1
 include_git_submodules: true
 instance_profile: null
 platform_name: null
 platform_version: null
 profile: null
 repository: null
 sc: null
 workspace_type: Application
```

C- create a ‘build’ script to execute the following :

1- ensure the installation of all the required packages via ‘npm install .’

2- build your project via ‘tsc’

3- and where the built version shall not contain the elasticbeanstalk , .npmrc neither the package.json we will add a copy command to the build folder ‘www’

4. to facilitate the deployment we will compress the built folder with the ‘zip’ command

```
"build": "npm install .&& tsc && cp -R .elasticbeanstalk
www/.elasticbeanstalk && cp .npmrc www/.npmrc && cp package.json
www/package.json && cd www && zip -r Archive.zip . && cd ..",
```

D- edit the .elasticbeanstalk/config.yml to consider the compressed built folder only for deployment by adding:

```
deploy:
  artifact: www/Archive.zip
```

E- start the deployment process with ‘deploy’ script to execute:

1- rum the build script

2- start the eb cli

3- use the established eb env.

4- set the necessary environmental variables via eb ‘setenv’

5- finally deploy the project via ‘eb deploy’

```
"deploy": "npm run build && eb init && eb use Udagramapi-env &&
eb setenv POSTGRES_USERNAME=$POSTGRES_USERNAME
POSTGRES_PASSWORD=$POSTGRES_PASSWORD POSTGRES_DB=$POSTGRES_DB
PORT=$PORT DB_PORT=$DB_PORT POSTGRES_HOST=$POSTGRES_HOST
AWS_REGION=$AWS_REGION AWS_ACCESS_KEY_ID=$AWS_ACCESS_KEY_ID
AWS_SECRET_ACCESS_KEY=$AWS_SECRET_ACCESS_KEY
JWT_SECRET=$JWT_SECRET && eb deploy",
```