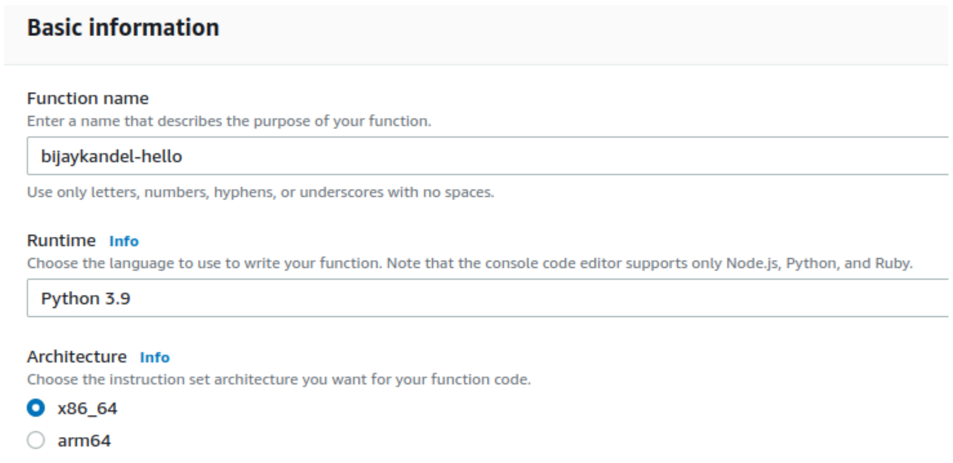
- Create two Lambda Functions
- First Lambda function returns 200 Response as {“Hello”: “Default”}
- Second Lambda function returns 200 Response as {“Hello”: “{Dynamic route name}”}
- Configure API Gateway with that hits first lambda function on / and the second lambda function on /*

Two lambda functions were created for this assignment:

First one is bijaykandel-hello



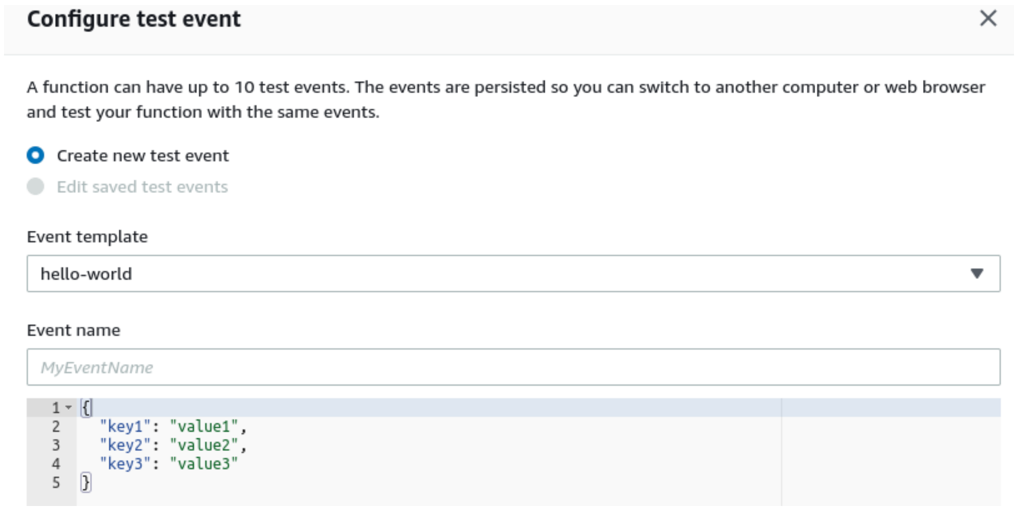
Basic information

Function name
Enter a name that describes the purpose of your function.
bijaykandel-hello
Use only letters, numbers, hyphens, or underscores with no spaces.

Runtime Info
Choose the language to use to write your function. Note that the console code editor supports only Node.js, Python, and Ruby.
Python 3.9

Architecture Info
Choose the instruction set architecture you want for your function code.
[✓ x86_64]
arm64

And its test event is configured as default.



Configure test event

A function can have up to 10 test events. The events are persisted so you can switch to another computer or web browser and test your function with the same events.

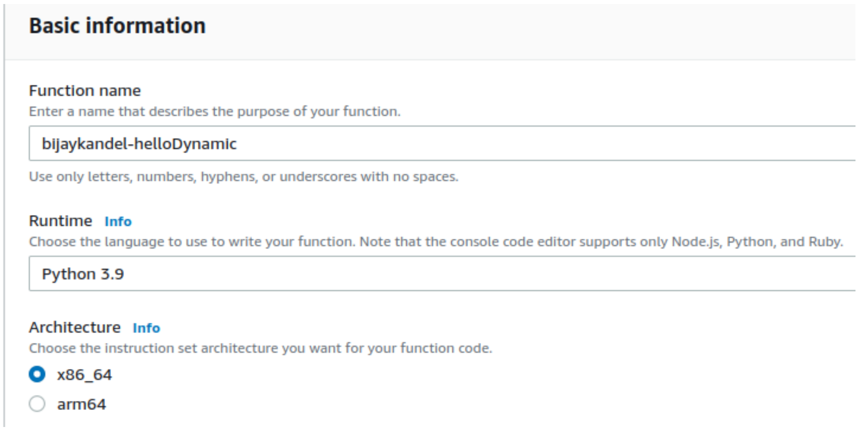
[✓ Create new test event]
Edit saved test events

Event template
hello-world

Event name
MyEventName

```
{
  "key1": "value1",
  "key2": "value2",
  "key3": "value3"
}
```

The second lambda function is named as bijaykandel-helloDynamic



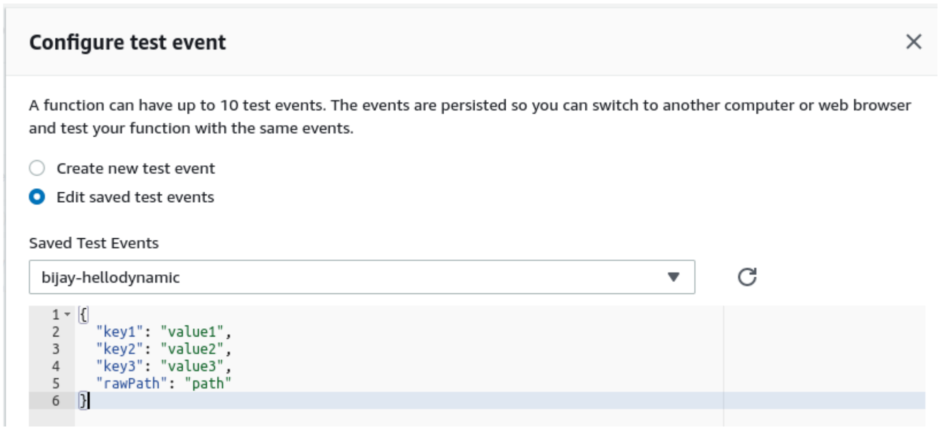
Basic information

Function name
Enter a name that describes the purpose of your function.
bijaykandel-helloDynamic
Use only letters, numbers, hyphens, or underscores with no spaces.

Runtime Info
Choose the language to use to write your function. Note that the console code editor supports only Node.js, Python, and Ruby.
Python 3.9

Architecture Info
Choose the instruction set architecture you want for your function code.
[✓ x86_64]
arm64

And its test event is configured as follows: ("rawPath" option is added in the existing default document)



Configure test event

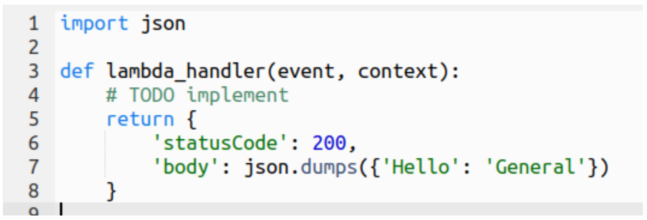
A function can have up to 10 test events. The events are persisted so you can switch to another computer or web browser and test your function with the same events.

Create new test event
[✓ Edit saved test events]

Saved Test Events
bijay-hellodynamic

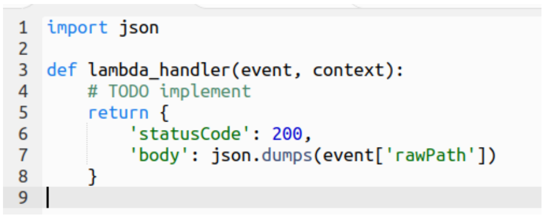
```
{
  "key1": "value1",
  "key2": "value2",
  "key3": "value3",
  "rawPath": "path"
}
```

First lambda function is written as:



```
import json

def lambda_handler(event, context):
    # TODO implement
    return {
        'statusCode': 200,
        'body': json.dumps({'Hello': 'General'})
    }
```

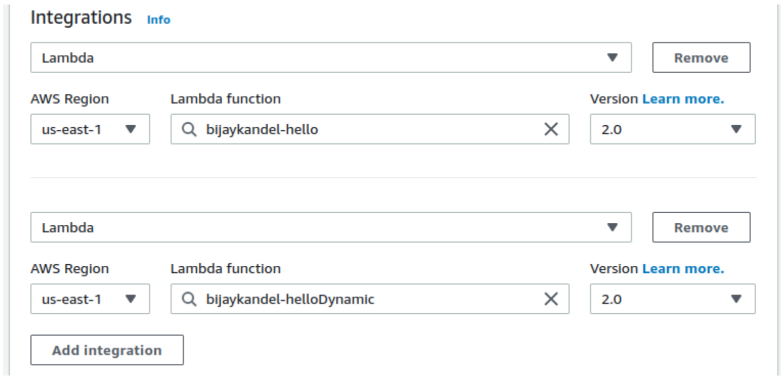
And the second lambda function is written as:



```
import json

def lambda_handler(event, context):
    # TODO implement
    return {
        'statusCode': 200,
        'body': json.dumps(event['rawPath'])
    }
```

Now API gateway is created: **API Gateway >> Create API gateway:**

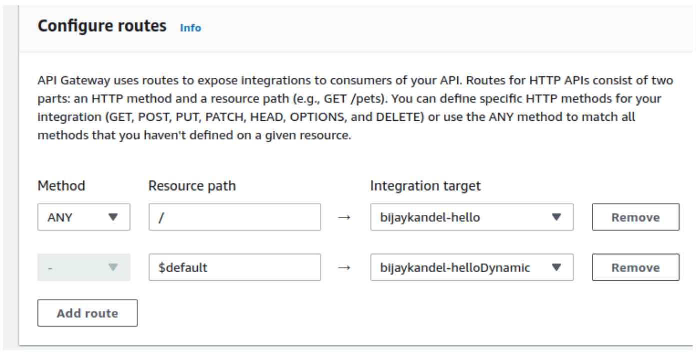
We added two integrations for two lambda functions as follows:



Integrations Info

Integration	AWS Region	Lambda function	Version
Lambda	us-east-1	bijaykandel-hello	2.0
Lambda	us-east-1	bijaykandel-helloDynamic	2.0

And Routes are configured so that accessing the URL triggers first lambda function and accessing URL/* triggers the second lambda function.

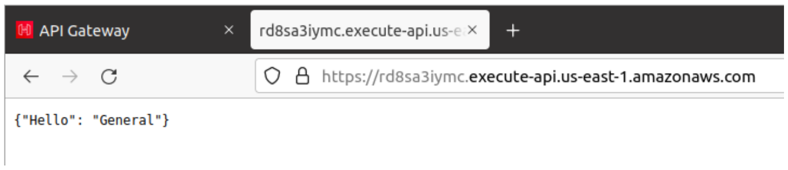


Configure routes Info

API Gateway uses routes to expose integrations to consumers of your API. Routes for HTTP APIs consist of two parts: an HTTP method and a resource path (e.g., GET /pets). You can define specific HTTP methods for your integration (GET, POST, PUT, PATCH, HEAD, OPTIONS, and DELETE) or use the ANY method to match all methods that you haven't defined on a given resource.

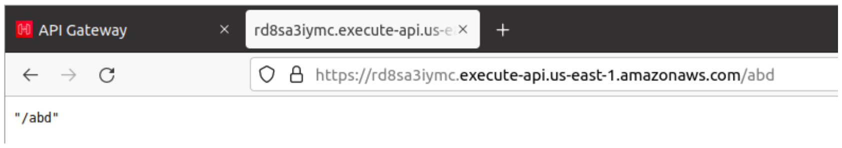
Method	Resource path	Integration target
ANY	/	bijaykandel-hello
-	$default	bijaykandel-helloDynamic

After creating the API gateway, we can click on the link inside our API gateway. With no path specified, the url returns the following value:



```
https://rd8sa3iymc.execute-api.us-east-1.amazonaws.com
{"Hello": "General"}
```

And if path is provided with url, it returns the given path as shown below:

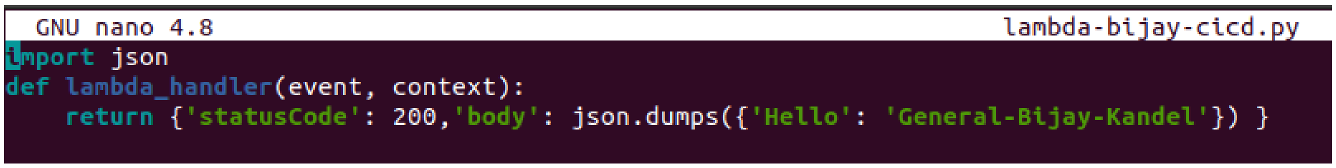


```
https://rd8sa3iymc.execute-api.us-east-1.amazonaws.com/abd
"/abd"
```

- <u>Create a bash script to deploy your lambda functions</u>

To deploy our lambda functions, we first created our lambda function in python:

Sudo nano lambda-bijay-cicd.py

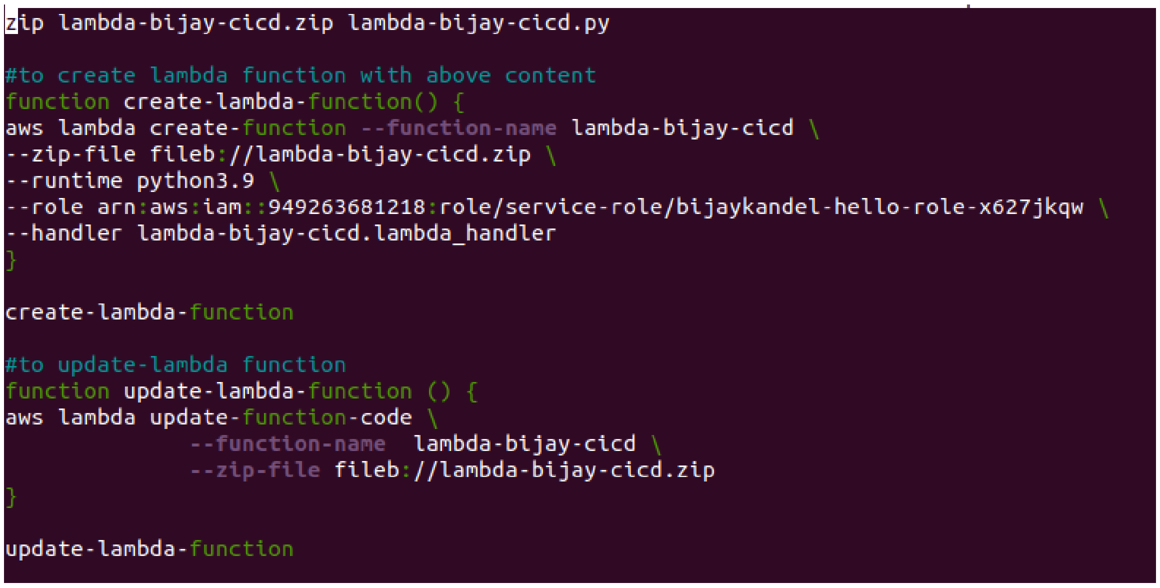


```
  GNU nano 4.8                                                          lambda-bijay-cicd.py
import json
def lambda_handler(event, context):
    return {'statusCode': 200,'body': json.dumps({'Hello': 'General-Bijay-Kandel'}) }
```

And a script is created for deploying above lambda function:

Nano scriptlambda.sh

Here, the role is copied from our lambda role from the previous assignment.

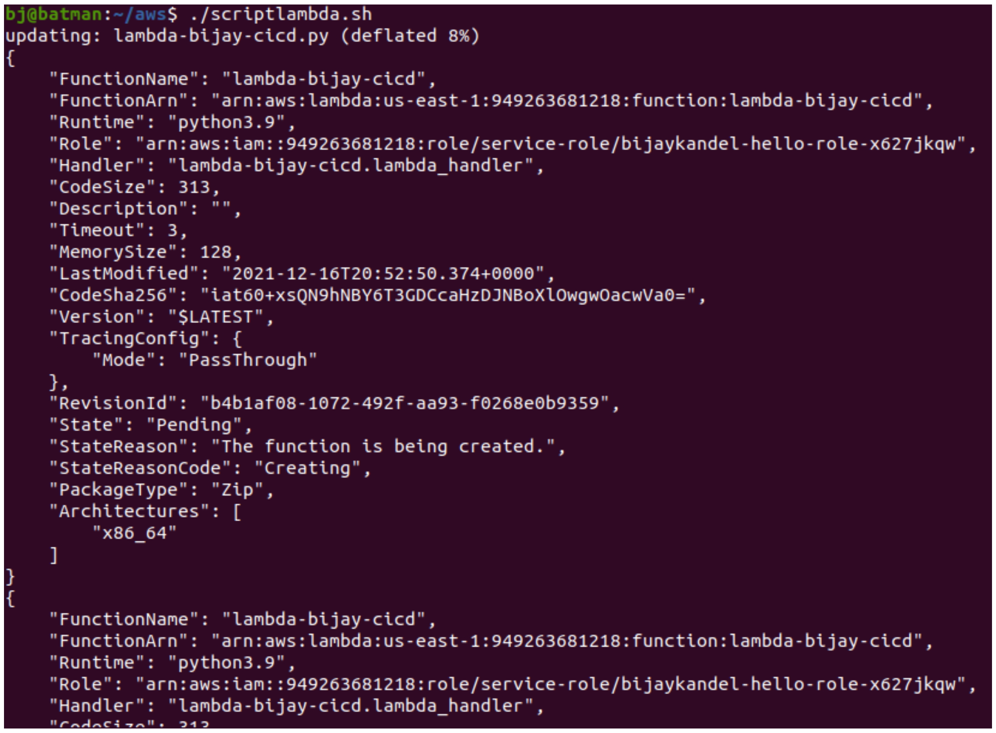


```
zip lambda-bijay-cicd.zip lambda-bijay-cicd.py

#to create lambda function with above content
function create-lambda-function() {
aws lambda create-function --function-name lambda-bijay-cicd \
--zip-file fileb://lambda-bijay-cicd.zip \
--runtime python3.9 \
--role arn:aws:iam::949263681218:role/service-role/bijaykandel-hello-role-x627jkqw \
--handler lambda-bijay-cicd.lambda_handler
}

create-lambda-function

#to update-lambda function
function update-lambda-function () {
aws lambda update-function-code \
              --function-name  lambda-bijay-cicd \
              --zip-file fileb://lambda-bijay-cicd.zip
}

update-lambda-function
```

The script is made executable with command:

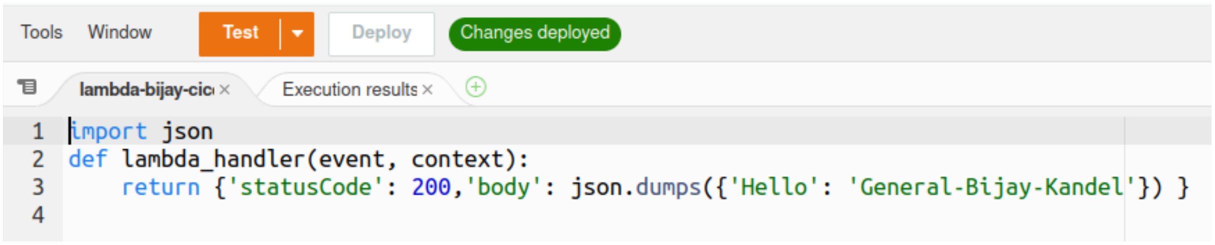
Sudo chmod +x scriptlambda.sh

Now when we run the script with command ***./scriptlambda.sh***



```
bj@batman:~/aws$ ./scriptlambda.sh
updating: lambda-bijay-cicd.py (deflated 8%)
{
    "FunctionName": "lambda-bijay-cicd",
    "FunctionArn": "arn:aws:lambda:us-east-1:949263681218:function:lambda-bijay-cicd",
    "Runtime": "python3.9",
    "Role": "arn:aws:iam::949263681218:role/service-role/bijaykandel-hello-role-x627jkqw",
    "Handler": "lambda-bijay-cicd.lambda_handler",
    "CodeSize": 313,
    "Description": "",
    "Timeout": 3,
    "MemorySize": 128,
    "LastModified": "2021-12-16T20:52:50.374+0000",
    "CodeSha256": "iat60+xsQN9hNBY6T3GDCcaHzDJNBoXlOwgwOacwVa0=",
    "Version": "$LATEST",
    "TracingConfig": {
        "Mode": "PassThrough"
    },
    "RevisionId": "b4b1af08-1072-492f-aa93-f0268e0b9359",
    "State": "Pending",
    "StateReason": "The function is being created.",
    "StateReasonCode": "Creating",
    "PackageType": "Zip",
    "Architectures": [
        "x86_64"
    ]
}
{
    "FunctionName": "lambda-bijay-cicd",
    "FunctionArn": "arn:aws:lambda:us-east-1:949263681218:function:lambda-bijay-cicd",
    "Runtime": "python3.9",
    "Role": "arn:aws:iam::949263681218:role/service-role/bijaykandel-hello-role-x627jkqw",
    "Handler": "lambda-bijay-cicd.lambda_handler",
```

On the console, we can see that our lambda function has been deployed.

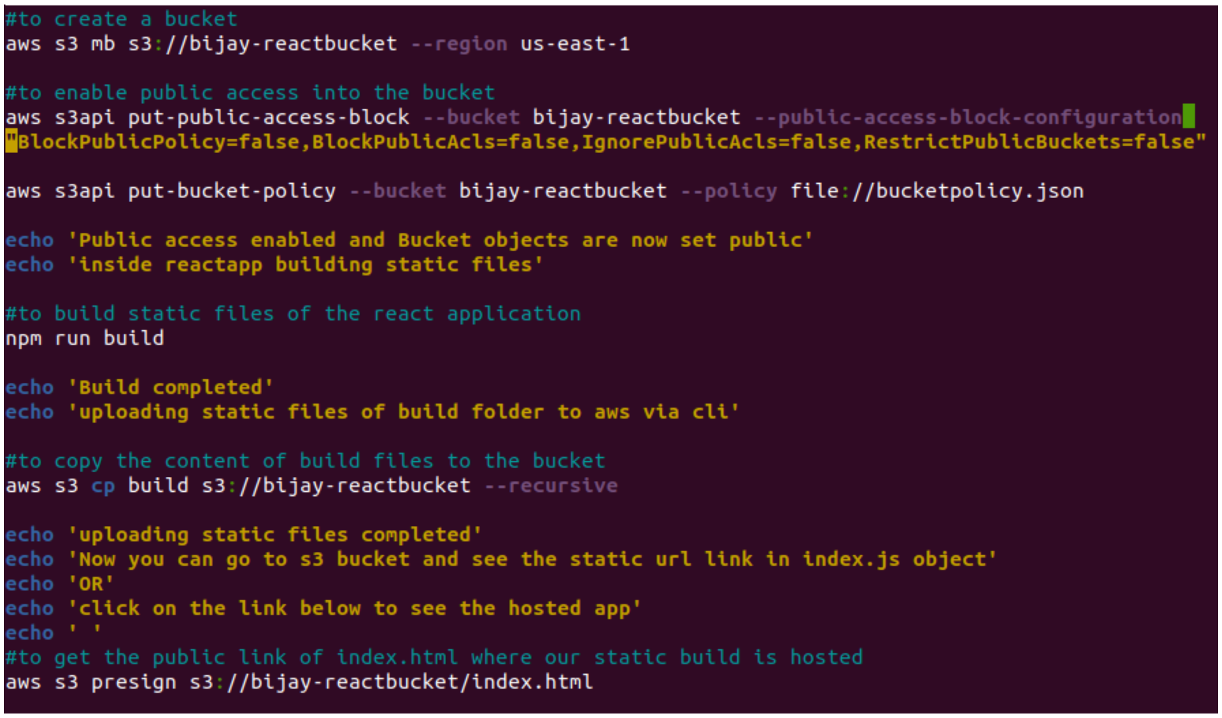


```python
import json
def lambda_handler(event, context):
    return {'statusCode': 200,'body': json.dumps({'Hello': 'General-Bijay-Kandel'}) }
```

- <u>Create a bash script to deploy your react app to S3</u>

To deploy react app to s3, following bash script file is created:

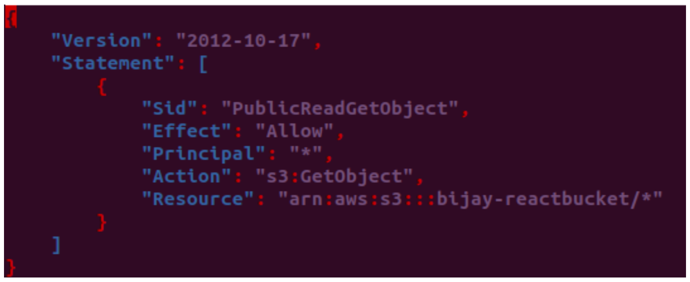
Sudo nano reactcicd.sh



```
#to create a bucket
aws s3 mb s3://bijay-reactbucket --region us-east-1

#to enable public access into the bucket
aws s3api put-public-access-block --bucket bijay-reactbucket --public-access-block-configuration
"BlockPublicPolicy=false,BlockPublicAcls=false,IgnorePublicAcls=false,RestrictPublicBuckets=false"

aws s3api put-bucket-policy --bucket bijay-reactbucket --policy file://bucketpolicy.json

echo 'Public access enabled and Bucket objects are now set public'
echo 'inside reactapp building static files'

#to build static files of the react application
npm run build

echo 'Build completed'
echo 'uploading static files of build folder to aws via cli'

#to copy the content of build files to the bucket
aws s3 cp build s3://bijay-reactbucket --recursive

echo 'uploading static files completed'
echo 'Now you can go to s3 bucket and see the static url link in index.js object'
echo 'OR'
echo 'click on the link below to see the hosted app'
echo ' '
#to get the public link of index.html where our static build is hosted
aws s3 presign s3://bijay-reactbucket/index.html
```

And bucket policy should be updated from a json file. So we created a json file and following part is saved to the file:

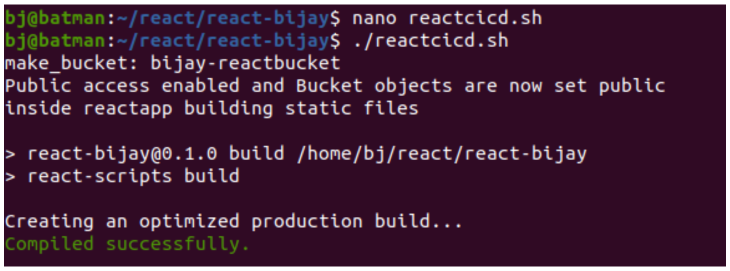
Sudo nano bucketpolicy.json



```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "PublicReadGetObject",
            "Effect": "Allow",
            "Principal": "*",
            "Action": "s3:GetObject",
            "Resource": "arn:aws:s3:::bijay-reactbucket/*"
        }
    ]
}
```

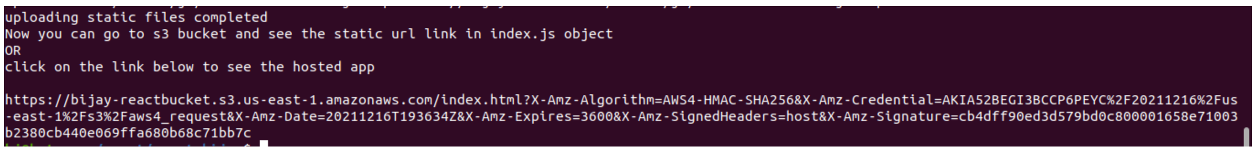
Making the script executable: ***sudo chmod +x reactcicd.sh***

Running the script file with command: ***./reactcicd.sh***



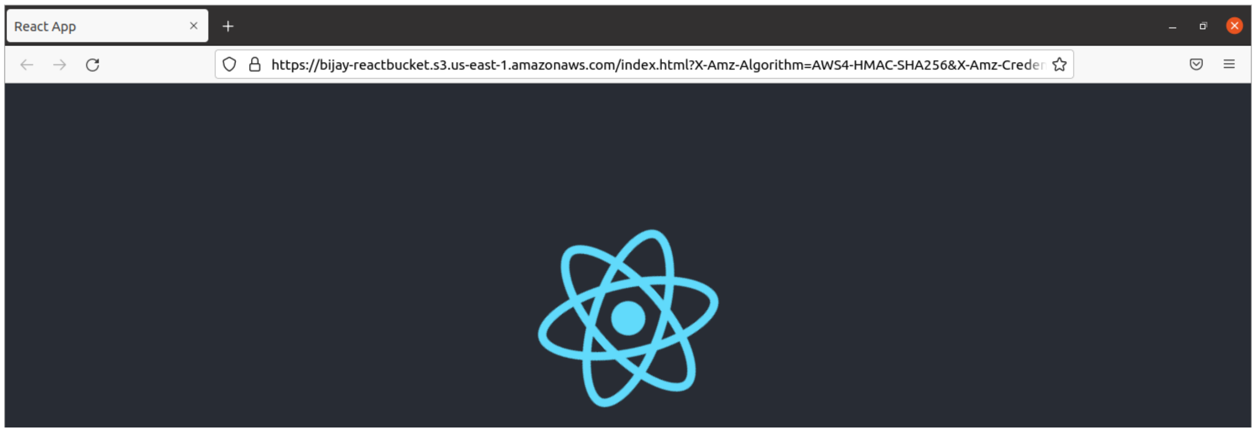
```
bj@batman:~/react/react-bijay$ nano reactcicd.sh
bj@batman:~/react/react-bijay$ ./reactcicd.sh
make_bucket: bijay-reactbucket
Public access enabled and Bucket objects are now set public
inside reactapp building static files

> react-bijay@0.1.0 build /home/bj/react/react-bijay
> react-scripts build

Creating an optimized production build...
Compiled successfully.
```

After uploading the static files, we can get the link of index.html file as shown:

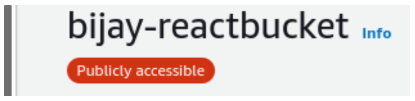


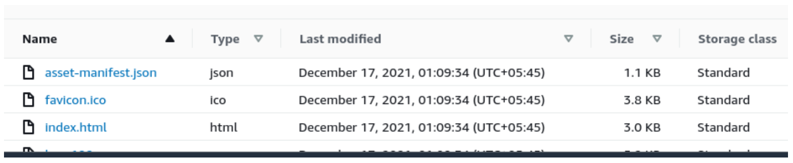
```
uploading static files completed
Now you can go to s3 bucket and see the static url link in index.js object
OR
click on the link below to see the hosted app

https://bijay-reactbucket.s3.us-east-1.amazonaws.com/index.html?X-Amz-Algorithm=AWS4-HMAC-SHA256&X-Amz-Credential=AKIA52BEGI3BCCP6PEYC%2F20211216%2Fus
-east-1%2Fs3%2Faws4_request&X-Amz-Date=20211216T193634Z&X-Amz-Expires=3600&X-Amz-SignedHeaders=host&X-Amz-Signature=cb4dff90ed3d579bd0c800001658e71003
b2380cb440e069ffa680b68c71bb7c
```

When we go to the link, we can see our page being hosted statically from s3.


React App
https://bijay-reactbucket.s3.us-east-1.amazonaws.com/index.html?X-Amz-Algorithm=AWS4-HMAC-SHA256&X-Amz-Creden

We can also see the bucket being publicly accessible and also the files uploaded in our s3 console.





Name	Type	Last modified	Size	Storage class
asset-manifest.json	json	December 17, 2021, 01:09:34 (UTC+05:45)	1.1 KB	Standard
favicon.ico	ico	December 17, 2021, 01:09:34 (UTC+05:45)	3.8 KB	Standard
index.html	html	December 17, 2021, 01:09:34 (UTC+05:45)	3.0 KB	Standard

- <u>Integrate both these scripts with one of Jenkins, Github Actions, CircleCI or TravisCI</u>

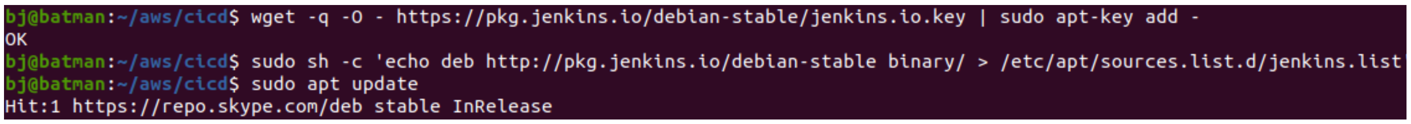
I used Jenkins for Continuous integration. First I installed jenkins in my local machine using following commands:

wget -q -O - https://pkg.jenkins.io/debian-stable/jenkins.io.key | sudo apt-key add -

sudo sh -c 'echo deb http://pkg.jenkins.io/debian-stable binary/ > /etc/apt/sources.list.d/jenkins.list'

sudo apt update

sudo apt install jenkins

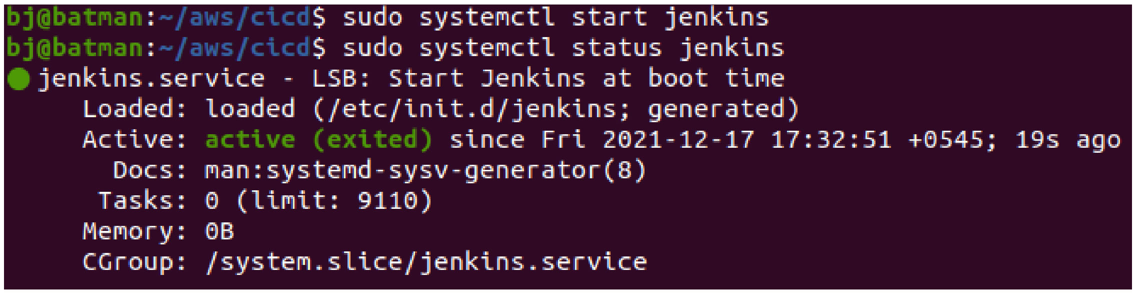


```
bj@batman:~/aws/cicd$ wget -q -O - https://pkg.jenkins.io/debian-stable/jenkins.io.key | sudo apt-key add -
OK
bj@batman:~/aws/cicd$ sudo sh -c 'echo deb http://pkg.jenkins.io/debian-stable binary/ > /etc/apt/sources.list.d/jenkins.list
bj@batman:~/aws/cicd$ sudo apt update
Hit:1 https://repo.skype.com/deb stable InRelease
```

Sudo systemctl start jenkins

sudo systemctl enable jenkins

Sudo systemctl status jenkins



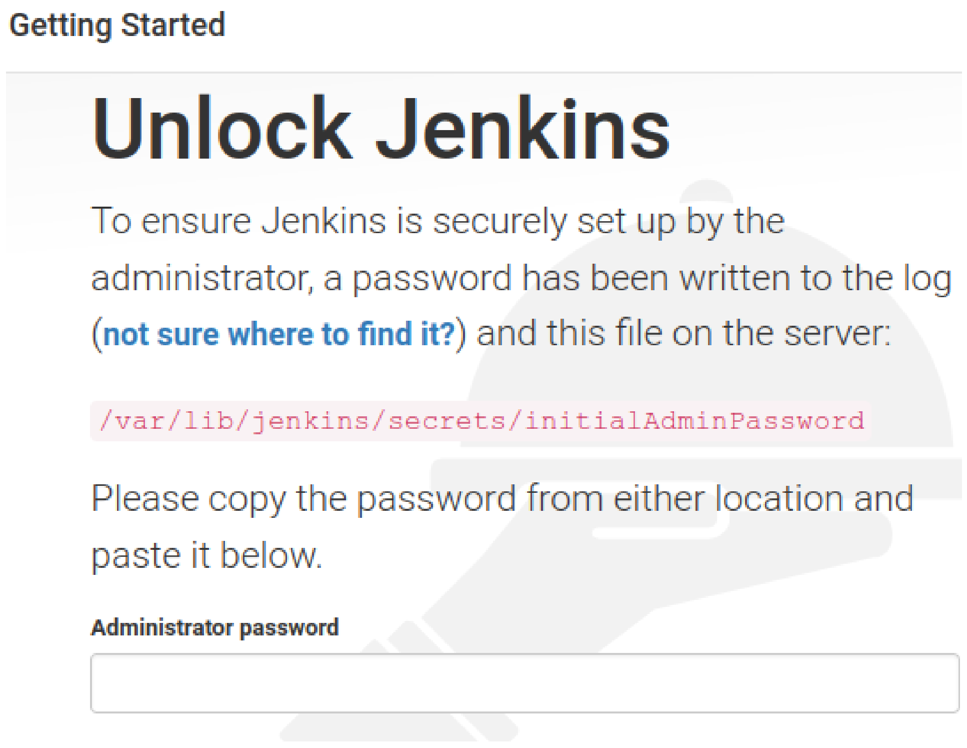
```
bj@batman:~/aws/cicd$ sudo systemctl start jenkins
bj@batman:~/aws/cicd$ sudo systemctl status jenkins
● jenkins.service - LSB: Start Jenkins at boot time
     Loaded: loaded (/etc/init.d/jenkins; generated)
     Active: active (exited) since Fri 2021-12-17 17:32:51 +0545; 19s ago
       Docs: man:systemd-sysv-generator(8)
      Tasks: 0 (limit: 9110)
     Memory: 0B
     CGroup: /system.slice/jenkins.service
```

Since Jenkins runs on port 8080, we have to allow this port on the firewall.

sudo ufw allow 8080

Ufw reload

Now, when we visit our server IP on port 8080, we can see the jenkins server running for the first time:


Getting Started

Unlock Jenkins

To ensure Jenkins is securely set up by the administrator, a password has been written to the log (not sure where to find it?) and this file on the server:

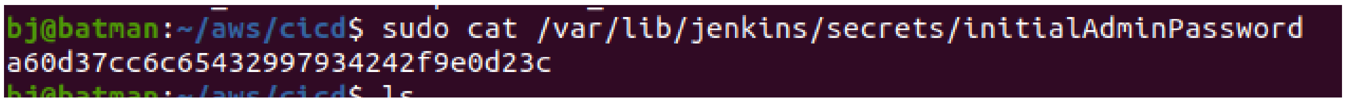
`/var/lib/jenkins/secrets/initialAdminPassword`

Please copy the password from either location and paste it below.

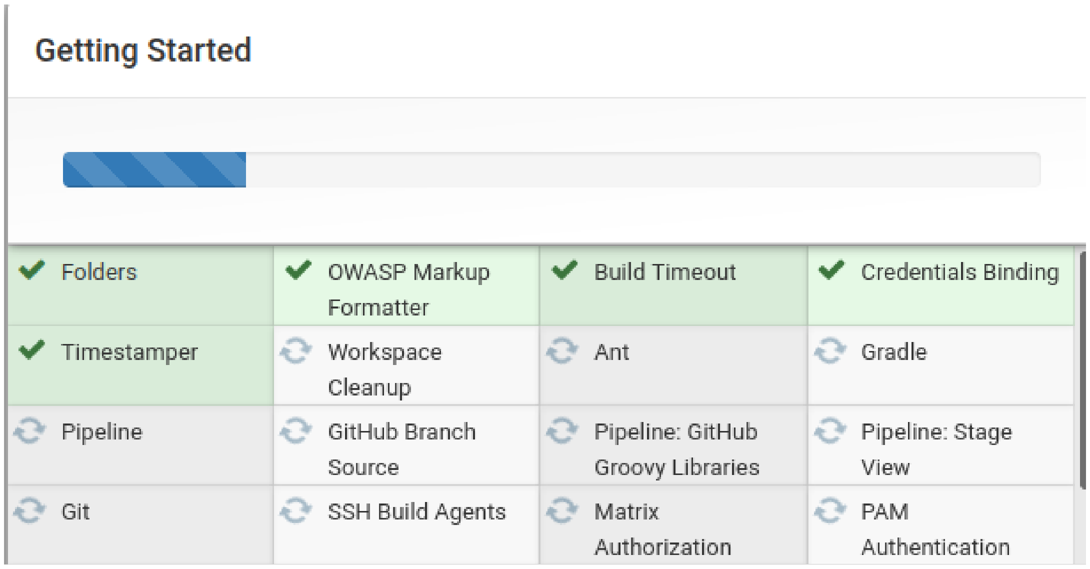
Administrator password

It asks for the Administrator password. For the initial password, we use this command and copy the key as password for initial setup.

sudo cat /var/lib/jenkins/secrets/initialAdminPassword


```
bj@batman:~/aws/cicd$ sudo cat /var/lib/jenkins/secrets/initialAdminPassword
a60d37cc6c654329979342 42f9e0d23c
```

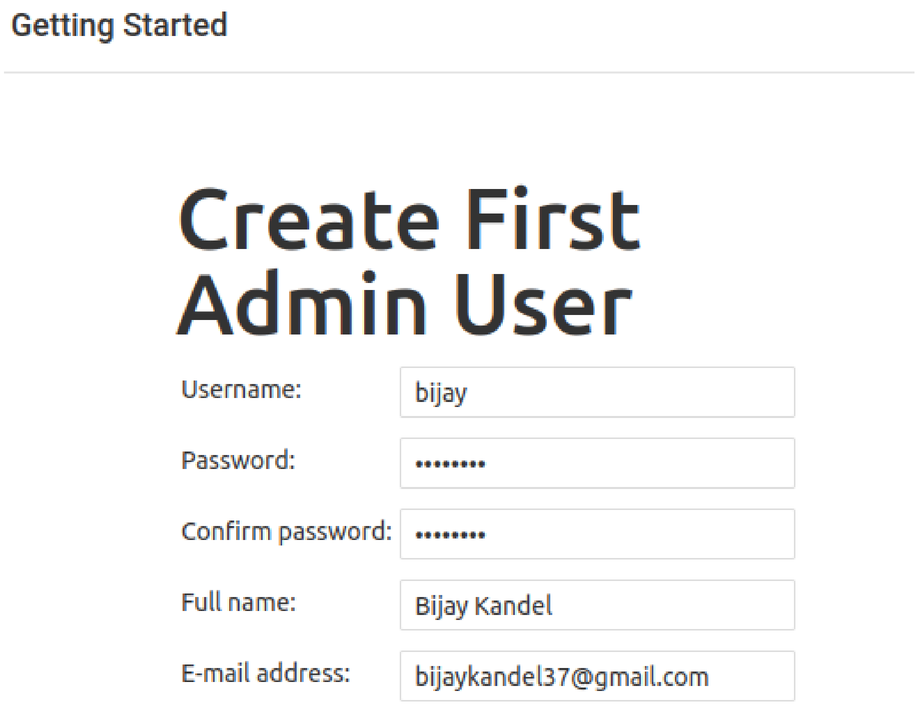
This key is used as password and then, Install suggested plugin option will install the required plugins for jenkins.



Getting Started

✓ Folders	✓ OWASP Markup Formatter	✓ Build Timeout	✓ Credentials Binding
✓ Timestamper	Workspace Cleanup	Ant	Gradle
Pipeline	GitHub Branch Source	Pipeline: GitHub Groovy Libraries	Pipeline: Stage View
Git	SSH Build Agents	Matrix Authorization	PAM Authentication

And we can now create Admin user only for the first time, after then, this data is used to login to the jenkins server.

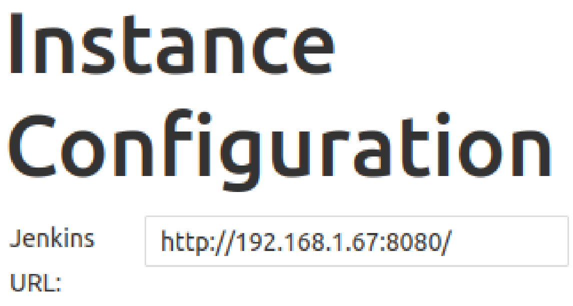


Getting Started

Create First Admin User

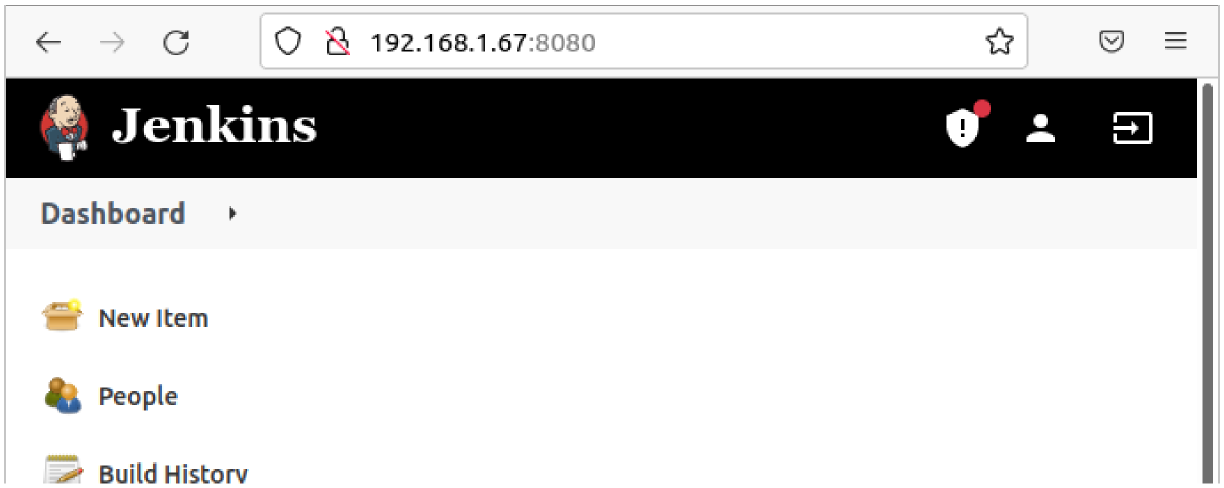
Username:	bijay
Password:	••••••••
Confirm password:	••••••••
Full name:	Bijay Kandel
E-mail address:	bijaykandel37@gmail.com

Now, we configure the instance of jenkins server on our server IP with port 8080.

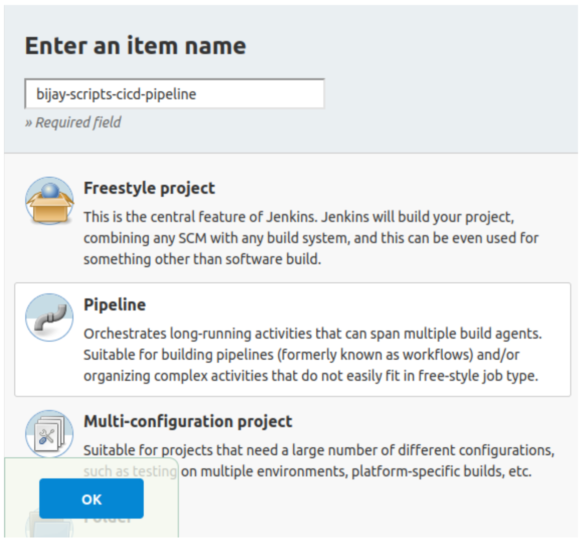

Instance Configuration

Jenkins URL: http://192.168.1.67:8080/

Now we can view the dashboard of jenkins as:



Now, to configure a new pipeline, we click on New item and give a name and select Pipeline:


Enter an item name

bijay-scripts-cicd-pipeline

» Required field

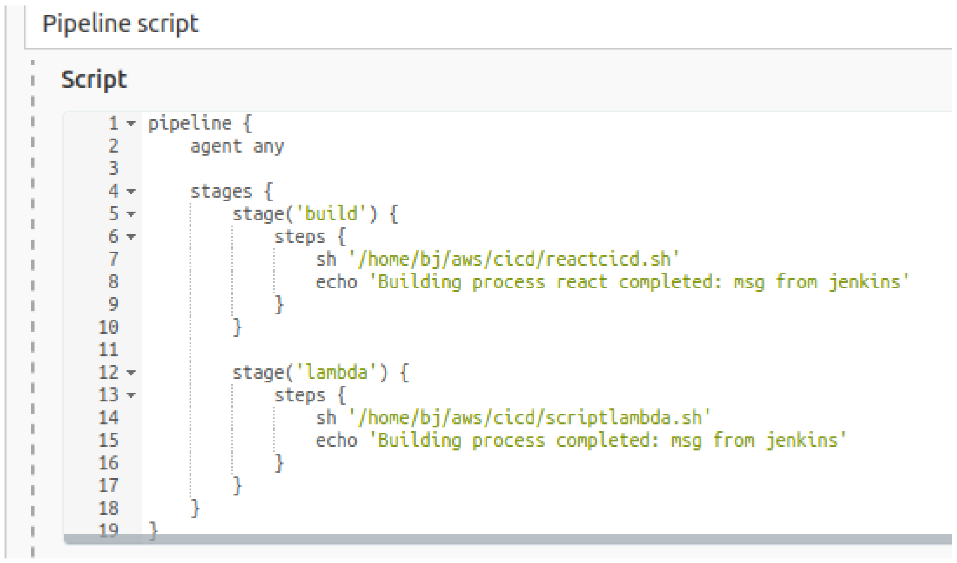
Freestyle project
This is the central feature of Jenkins. Jenkins will build your project, combining any SCM with any build system, and this can be even used for something other than software build.

Pipeline
Orchestrates long-running activities that can span multiple build agents. Suitable for building pipelines (formerly known as workflows) and/or organizing complex activities that do not easily fit in free-style job type.

Multi-configuration project
Suitable for projects that need a large number of different configurations, such as testing on multiple environments, platform-specific builds, etc.

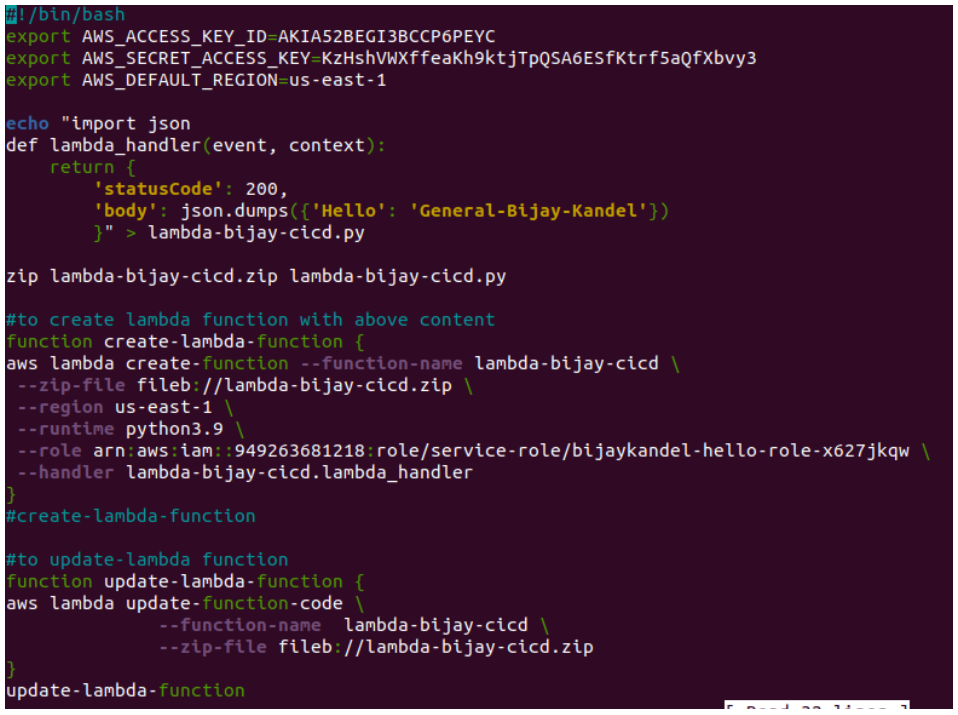
OK

Now, Pipeline is created for two scripts which are used in above questions.



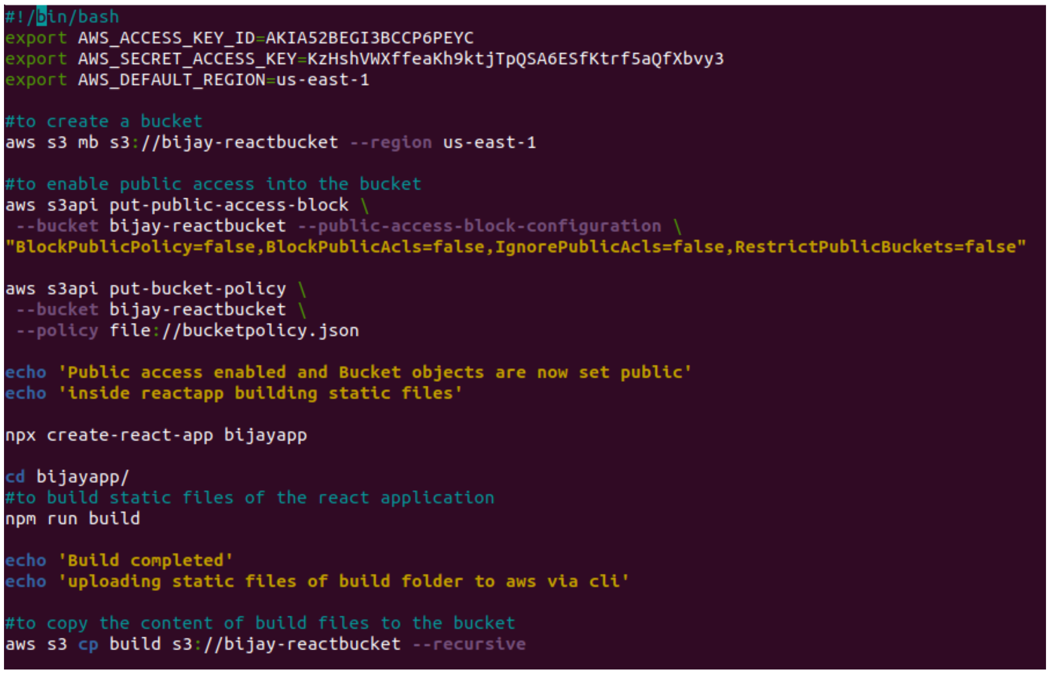
```
pipeline {
    agent any

    stages {
        stage('build') {
            steps {
                sh '/home/bj/aws/cicd/reactcicd.sh'
                echo 'Building process react completed: msg from jenkins'
            }
        }

        stage('lambda') {
            steps {
                sh '/home/bj/aws/cicd/scriptlambda.sh'
                echo 'Building process completed: msg from jenkins'
            }
        }
    }
}
```

But we have to provide credentials for AWS service. So with minor modification on above scripts, we set the scripts as follows: Lambda Script:



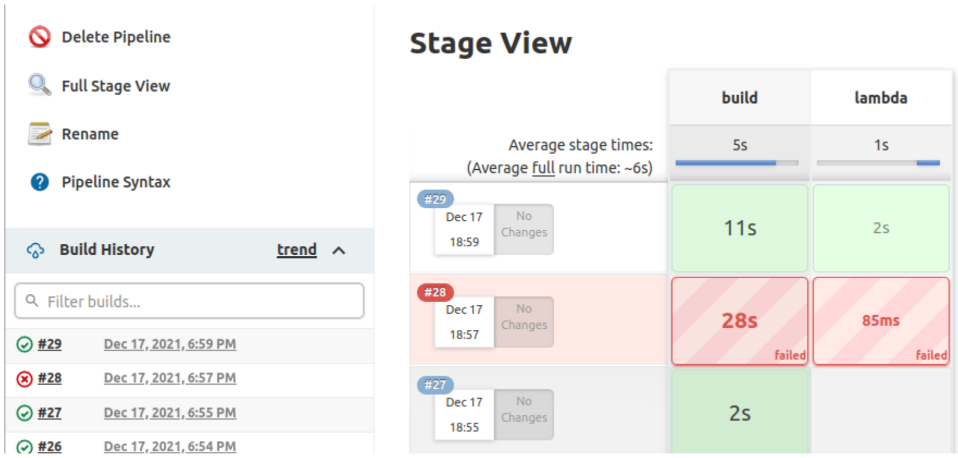
```
#!/bin/bash
export AWS_ACCESS_KEY_ID=AKIA52BEGI3BCCP6PEYC
export AWS_SECRET_ACCESS_KEY=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
export AWS_DEFAULT_REGION=us-east-1

echo "import json
def lambda_handler(event, context):
    return {
        'statusCode': 200,
        'body': json.dumps({'Hello': 'General-Bijay-Kandel'})
        }" > lambda-bijay-cicd.py

zip lambda-bijay-cicd.zip lambda-bijay-cicd.py

#to create lambda function with above content
function create-lambda-function {
aws lambda create-function --function-name lambda-bijay-cicd \
 --zip-file fileb://lambda-bijay-cicd.zip \
 --region us-east-1 \
 --runtime python3.9 \
 --role arn:aws:iam::949263681218:role/service-role/bijaykandel-hello-role-x627jkqw \
 --handler lambda-bijay-cicd.lambda_handler
}
#create-lambda-function

#to update-lambda function
function update-lambda-function {
aws lambda update-function-code \
              --function-name  lambda-bijay-cicd \
              --zip-file fileb://lambda-bijay-cicd.zip
}
update-lambda-function
```

React Script:



```
#!/bin/bash
export AWS_ACCESS_KEY_ID=AKIA52BEGI3BCCP6PEYC
export AWS_SECRET_ACCESS_KEY=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
export AWS_DEFAULT_REGION=us-east-1

#to create a bucket
aws s3 mb s3://bijay-reactbucket --region us-east-1

#to enable public access into the bucket
aws s3api put-public-access-block \
 --bucket bijay-reactbucket --public-access-block-configuration \
"BlockPublicPolicy=false,BlockPublicAcls=false,IgnorePublicAcls=false,RestrictPublicBuckets=false"

aws s3api put-bucket-policy \
 --bucket bijay-reactbucket \
 --policy file://bucketpolicy.json

echo 'Public access enabled and Bucket objects are now set public'
echo 'inside reactapp building static files'

npx create-react-app bijayapp

cd bijayapp/
#to build static files of the react application
npm run build

echo 'Build completed'
echo 'uploading static files of build folder to aws via cli'

#to copy the content of build files to the bucket
aws s3 cp build s3://bijay-reactbucket --recursive
```

Now, saving the pipeline and Clicking on Build Now builds the pipeline and returns errors if any. If no errors are found then build is success as shown below :



✓	#29	Dec 17, 2021, 6:59 PM
✗	#28	Dec 17, 2021, 6:57 PM
✓	#27	Dec 17, 2021, 6:55 PM
✓	#26	Dec 17, 2021, 6:54 PM

	build	lambda
Average stage times: (Average full run time: ~6s)	5s	1s
#29 Dec 17 18:59 No Changes	11s	2s
#28 Dec 17 18:57 No Changes	28s failed	85ms failed
#27 Dec 17 18:55 No Changes	2s	