Tutorial Faas

Concrete example - AWS Lambda

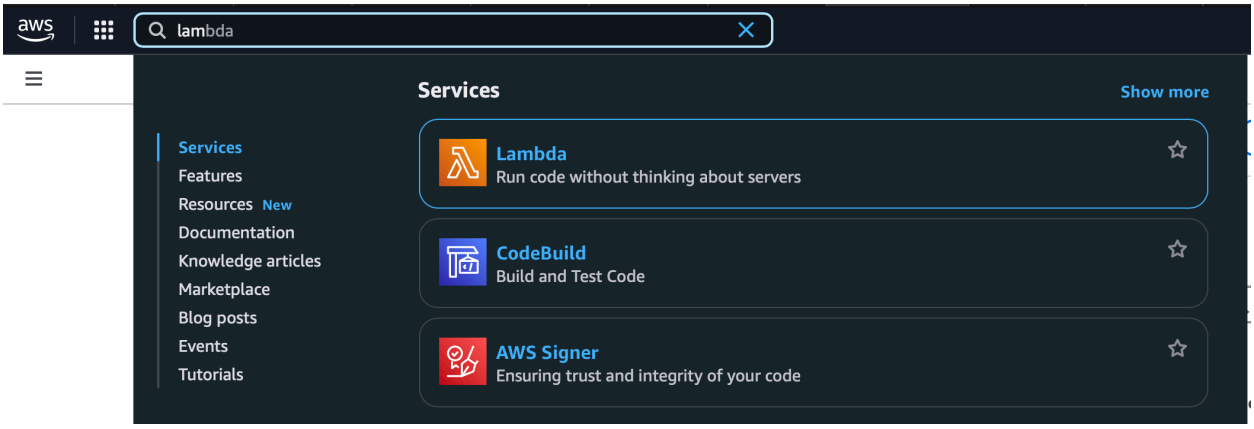
AWS Lambda is a **serverless compute service** that allows you to run code without managing servers. You simply upload your code, and Lambda automatically handles the infrastructure, scaling, and execution. You only pay for the compute time your code consumes, making it cost-effective for event-driven applications.

Step 1: Sign up for an AWS Account

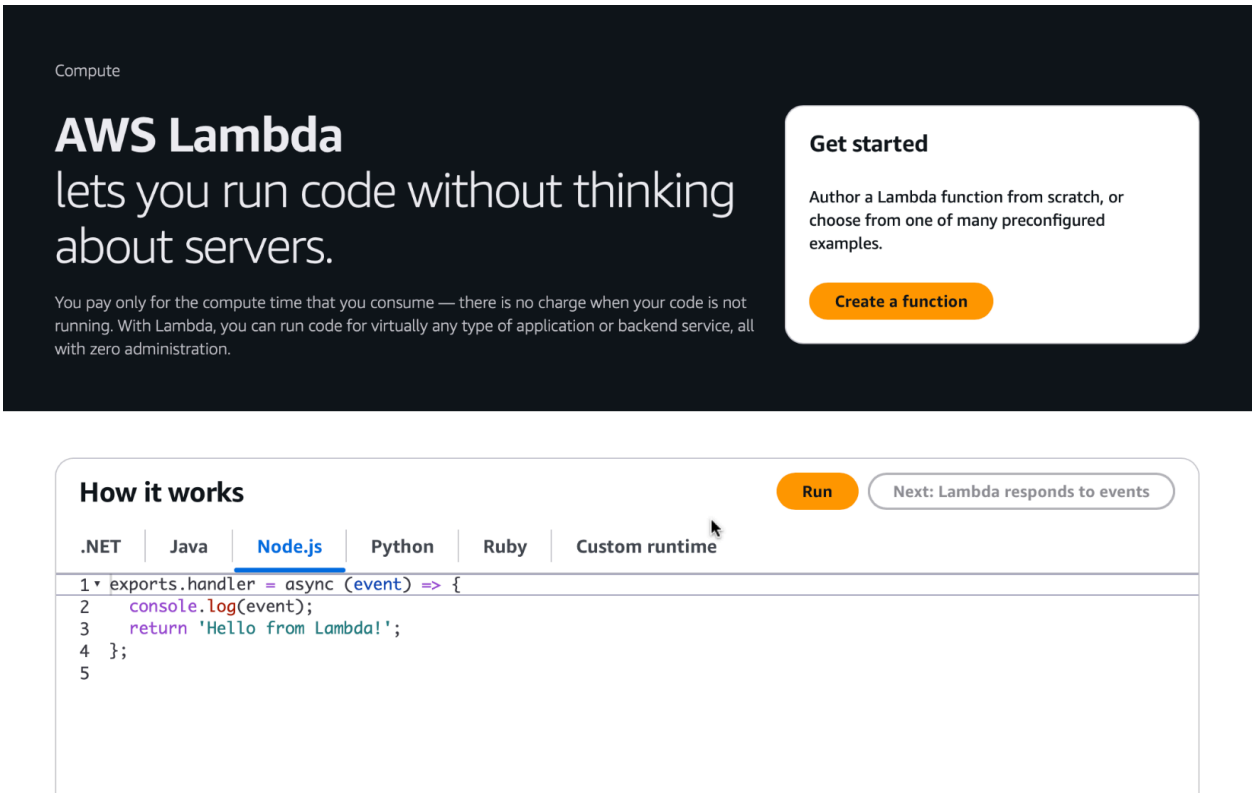
If you don't already have an AWS account, go to AWS and sign up. You'll need to provide credit card details for billing, but AWS offers a free tier for most services, including Lambda.

Step 2: Create a Lambda Function

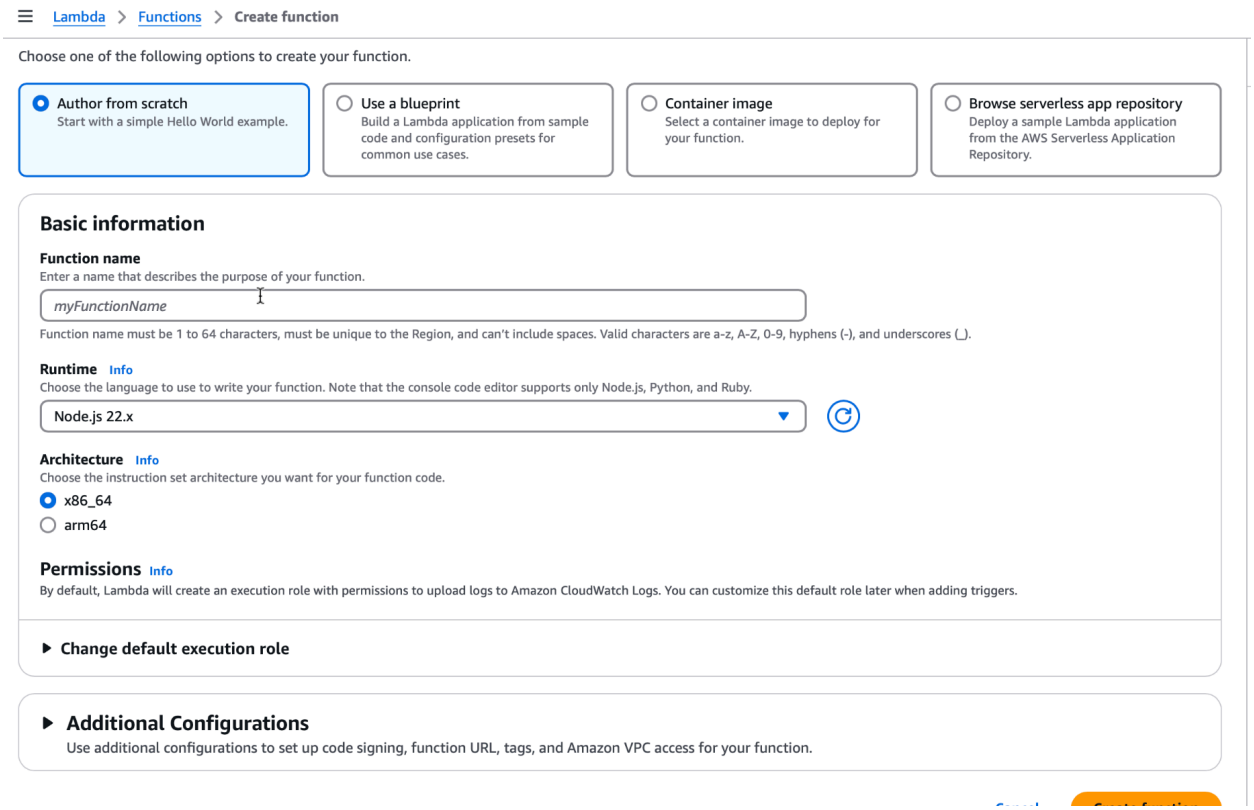
Sign in to your AWS Account and search for Lambda.



Click Create function.



Choose your settings (function name, runtime - e.g. Python 3.10 -, architecture) and click ‘create function’.

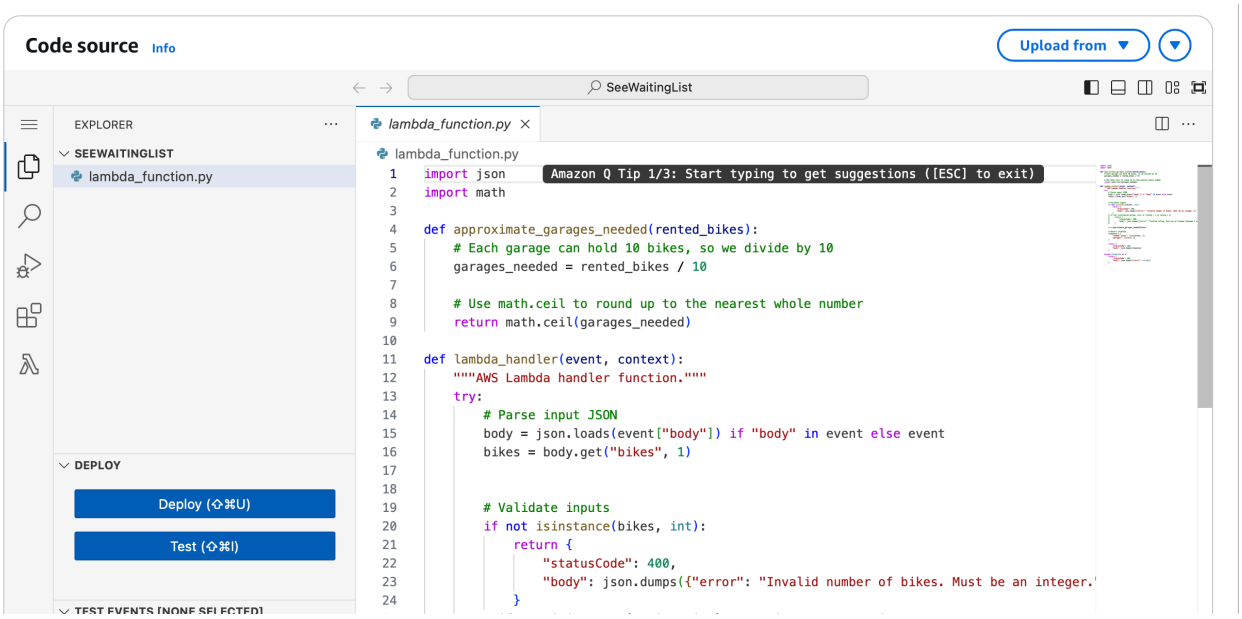


Step 3: Write Your Lambda Function Code

Once your function is created, you will be taken to the Lambda function's configuration page. Here, you can write or upload your function code.

For example, in Python:

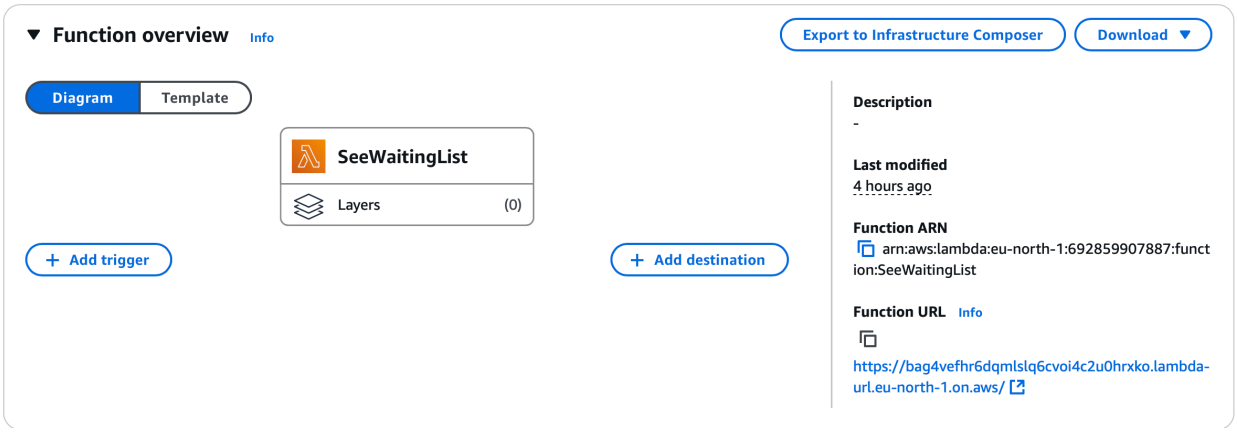
```
def lambda_handler(event, context):
    print("Hello from Lambda!")
    return {
        'statusCode': 200,
        'body': 'Hello from Lambda!'
    }
```



Click Deploy to save the function.

Step 4: Test the Lambda Function

Test the function by executing a ‘curl’, from your local machine, to the *Function URL*.




```
curl -X POST https://bag4vefhr6dqmlslq6cvoi4c2u0hrxko.lambda-url.eu-north-1
    -H "Content-Type: application/json" \
    -d '{
```

```
    "bikes": 28
}'
```

Step 5: Add an Event Source to Trigger the Lambda Function

Lambda functions can be triggered by various AWS services like S3, DynamoDB, API Gateway, etc. Let's use API Gateway to invoke the function via an HTTP request.

1. In your Lambda function's configuration page, click on **Add trigger**.
2. Select **API Gateway**.
3. Choose **Create an API** and set the following:
    - **API type**: Choose REST API or HTTP API (HTTP is simpler and cheaper for most use cases).
    - **Security**: Select **Open** if you want to allow anyone to invoke the function, or configure authentication methods.
4. Click **Add**.

API Gateway will create an HTTP endpoint for your Lambda function. After this, you'll see the endpoint URL, which you can use to make HTTP requests to invoke your function.

Step 6 (Optional): Integrate it in your python server

Add the LAMBDA_URL to your .env and then just send a request to AWS Lambda.

```python
@bike_bp.route('/get-garages', methods=['GET'])
@role_required('admin', 'renter')
def get_garages():
    bikes_count = Bike.query.count()
    response = requests.post(os.getenv('LAMBDA_URL'), json={"bikes": bikes_count})

    if response.status_code == 200:
        response_data = response.json()
        return jsonify(response_data)
    else:
        return jsonify({"error": "Error calling Lambda function"}), 500
```

Step 7 (Optional): Integrate it in your NGINX configuration

```
location /user/get-garages/ {
    proxy_pass http://lambda_faas:3000/dev/user/garages;  # Forward to the correct path

    add_header Access-Control-Allow-Origin *;
    add_header Access-Control-Allow-Methods "GET, POST, OPTIONS";
    add_header Access-Control-Allow-Headers "Authorization, Content-Type";
    add_header Access-Control-Allow-Credentials true;

    # Set headers for proper forwarding
    proxy_set_header Host $host;
    proxy_set_header X-Real-IP $remote_addr;
    proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
    proxy_set_header X-Forwarded-Proto $scheme;
    proxy_set_header X-Amzn-Trace-Id $http_x_amzn_trace_id;

    # Optional: Handle timeouts (adjust as needed)
    proxy_connect_timeout 60s;
    proxy_send_timeout 60s;
    proxy_read_timeout 60s;

    proxy_intercept_errors off;
}
```