

1 - Deploy Backend with AWS Lambda + API Gateway.

~ In this first stage, you will build a **Serverless API** that return cloud fun facts.

Step - 1: Create an AWS Lambda Function:

1. Go to AWS Management Console → Search for “**Lambda**”.
2. click “**Create Function**” in AWS Lambda.
3. Select **Author from scratch**.

Function name: CloudFunFacts

Runtime: Python 3

The screenshot shows the AWS Lambda home page. The main heading is "AWS Lambda" with the subtext "lets you run code without thinking about servers." Below this, a note states: "You pay only for the compute time that you consume — there is no charge when your code is not running. With Lambda, you can run code for virtually any type of application or backend service, all with zero administration." To the right, a "Get started" box contains the text "Author a Lambda function from scratch, or choose from one of many preconfigured examples." A yellow "Create a function" button is at the bottom. At the bottom of the page, a code editor shows a Node.js script:

```
1 exports.handler = async (event) => {
2   console.log(event);
3   return 'Hello from Lambda!';
}
```

The screenshot shows the "Create function" wizard. The first step is "Basic information". It includes fields for "Function name" (set to "cloudFunFactsCreate"), "Runtime" (set to "Python 3.14"), and "Architecture" (set to "x86_64"). There are three options available: "Author from scratch" (selected), "Use a blueprint" (unselected), and "Container image" (unselected). The "Author from scratch" option has a note: "Start with a simple Hello World example." The "Container image" option has a note: "Select a container image to deploy for your function." The bottom of the page includes standard AWS navigation links like CloudShell, Feedback, and Console Mobile App.

The screenshot shows the AWS Lambda 'Create function' wizard. The current step is 'Permissions'. It asks for a role definition, with the 'Create a new role with basic Lambda permissions' option selected. A note says 'Role creation might take a few minutes. Do not delete the role or edit the trust or permissions policies in this role.' Below this, it states that Lambda will create an execution role named 'cloudFunFactsCreate-role-z9075mpc' with permission to upload logs to Amazon CloudWatch Logs. There is also an 'Additional configurations' section. At the bottom right are 'Cancel' and 'Create function' buttons.

👉 What's happening here?

- Lambda is AWS's serverless compute service — you don't need to manage servers, scaling, or patching.
- By creating this function, you're telling AWS: "Whenever this function is triggered, run my Python code and return the output."
- The basic Lambda role gives it permission to write logs into CloudWatch so you can debug later.

4. Click "Create Function".

Step - 2: Add code in AWS Lambda Function:

1. In the Lambda “**Code Source**” section.
2. copy and paste the below code.

Lambda Code:

```
import random
import json

def lambda_handler(event, context):

    facts = [
        "AWS S3 was launched in 2006 and still rules cloud storage.",
        "Cloud computing can save companies up to 30% on IT costs.",
        "EC2 was one of the first AWS services to change IT forever.",
        "AWS offers more than 200 services — that's more than Starbucks drinks!",
        "Cloud lets you pay-as-you-go, just like your Netflix subscription.",
        "The name 'Amazon Web Services' was first used back in 2002.",
        "AWS data centers are so secure they require palm scanners.",
        "Netflix runs most of its infrastructure on AWS.",
        "Amazon DynamoDB can handle more than 10 trillion requests per day.",
        "AWS Lambda was launched in 2014 and started the serverless trend.",
        "Cloud reduces CO2 emissions by optimizing energy usage.",
        "AWS regions have multiple Availability Zones for reliability.",
        "Amazon originally created S3 to solve its own scaling issues.",
        "More than 80% of Fortune 500 companies use AWS.",
        "Cloud helps startups scale globally without huge upfront costs.",
        "Amazon's first region outside the US was launched in Ireland (2007).",
        "AWS provides free tiers so students can build projects affordably.",
        "AWS CloudFront is one of the largest CDNs in the world.",
        "Serverless means you never patch servers — AWS does it for you!",
        "AWS is the market leader in cloud with ~32% share (as of 2025)."

    ]
    fact = random.choice(facts)

    return {
        "statusCode": 200,
        "headers": {"Content-Type": "application/json"},
        "body": json.dumps({"fact": fact})
    }
```

```

import random
import json

def lambda_handler(event, context):
    facts = [
        "AWS S3 was launched in 2006 and still rules cloud storage",
        "Cloud computing can save companies up to 30% on IT costs",
        "EC2 was one of the first AWS services to change IT forever",
        "AWS offers more than 200 services - that's more than any other provider",
        "Cloud lets you pay-as-you-go, just like your Netflix bill",
        "The name 'Amazon Web Services' was first used back in 2006",
        "AWS data centers are so secure they require palm scanning",
        "Netflix runs most of its infrastructure on AWS",
        "Amazon DynamoDB can handle more than 10 trillion requests per second",
        "AWS Lambda was launched in 2014 and started the serverless trend",
        "Cloud reduces CO2 emissions by optimizing energy usage",
        "AWS regions have multiple Availability Zones for redundancy",
        "Amazon originally created S3 to solve its own scalability issues",
        "More than 80% of Fortune 500 companies use AWS",
        "Cloud helps startups scale globally without huge upfront investment",
        "Amazon's first region outside the US was launched in 2011",
        "AWS provides free tiers so students can build projects",
        "AWS CloudFront is one of the largest CDNs in the world",
        "Serverless means you never patch servers - AWS does it for you",
        "AWS is the market leader in cloud with ~32% share (as of 2025)"
    ]
    fact = random.choice(facts)
    return {
        "statusCode": 200,
        "headers": {"Content-Type": "application/json"},
        "body": json.dumps({"fact": fact})
    }

```

👉 What's happening here?

- Picks a random fun fact from our list.
- Returns it as a JSON response with status code 200 (success).
- The event and context arguments let Lambda handle inputs (like API calls) — even if we're not using them yet.

3. Click Deploy to save the changes.

Successfully updated the function cloudFunFactsCreate.

```

24     "Serverless means you never patch servers - AWS does it for you!",
25     "AWS is the market leader in cloud with ~32% share (as of 2025)."
]
]

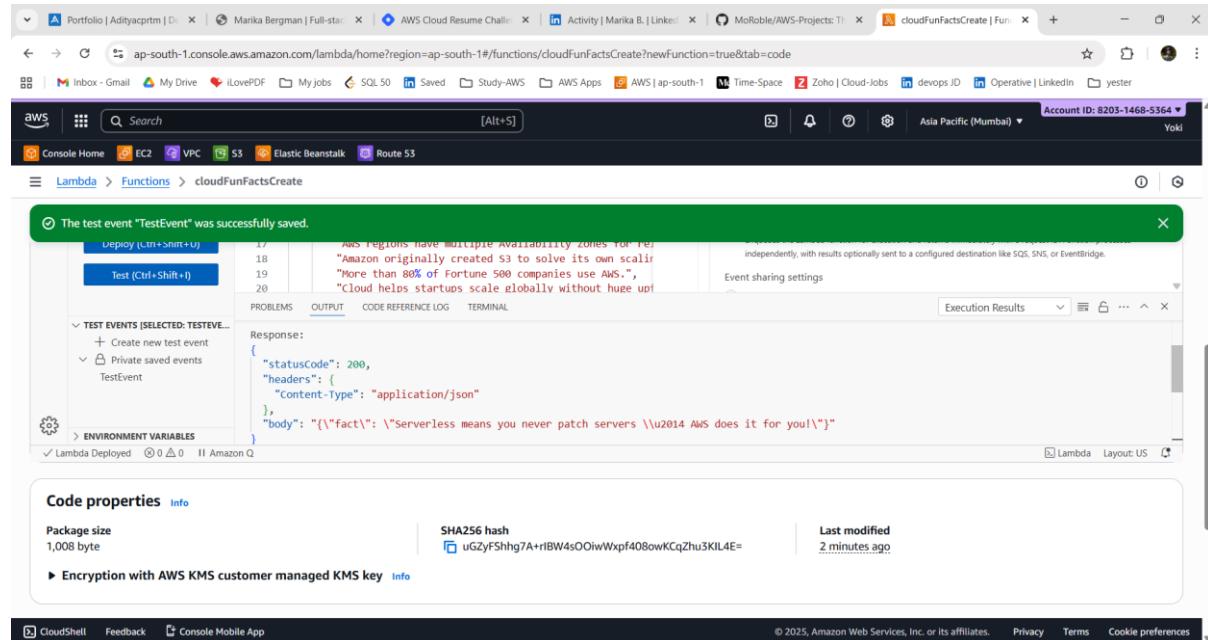
fact = random.choice(facts)
return {
    "statusCode": 200,
    "headers": {"Content-Type": "application/json"},
    "body": json.dumps({"fact": fact})
}

```

Step - 3: Test Lambda:

1. Click “Test” → Create a new Test event → Name: **TestEvent1**
2. Leave the default JSON ({})
3. Run the test > Save and Click Invoke.

→ You should see a random fun fact returned in the response.



👉 What's happening here?

- You're manually invoking the Lambda in the console
- AWS runs your code in a temporary environment, then destroys it (that's the beauty of serverless).

Step - 4: Create API Gateway:

→ Now we'll make this Lambda accessible through a public API endpoint.

1. Go to API Gateway → Click Create API.

2. Select HTTP API (simpler and cheaper than REST API).

API Name → FunfactsAPI

Integration → Select your Lambda function CloudFunFacts > Click Next.

The screenshot shows the AWS API Gateway landing page. The top navigation bar includes links for Console Home, EC2, VPC, S3, Elastic Beanstalk, and Route 53. The main content area features a dark header with the text "Networking & Content Delivery" and "API Gateway". Below it, a sub-header says "Create and manage APIs at scale". A paragraph explains that Amazon API Gateway is a fully managed service for creating, publishing, maintaining, monitoring, and securing APIs. To the right, a "Get started" box contains instructions to create a new API or import an external definition file, with a prominent "Create an API" button. Another section, "How it works", contains a diagram showing the flow from various clients (mobile devices, web and mobile applications) through the API Gateway to AWS Lambda, Amazon API Gateway, and other services like Amazon VPC and Amazon DynamoDB. To the right of this, a "Pricing" section states that users only pay when APIs are in use, with no minimum fees or upfront commitments. It provides a link to learn more about pricing. At the bottom, there are links for CloudShell, Feedback, and Console Mobile App, along with copyright information for 2025 and links for Privacy, Terms, and Cookie preferences.

The screenshot shows the "Create HTTP API" step in the AWS API Gateway wizard. The top navigation bar is identical to the previous screenshot. The main form is titled "API name" and contains a field with the value "FunfactsAPI". Below this, the "IP address type" section is set to "IPv4". The "Integrations (1)" section lists a single integration with the Lambda function "arn:aws:lambda:ap-south-1:820314685364:function:cloudFun". At the bottom, there are "Cancel", "Review and create", and "Next" buttons, along with links for CloudShell, Feedback, and Console Mobile App, and standard footer links for Privacy, Terms, and Cookie preferences.

3. Add a route:

Method: GET

Path: /funfact

Configure routes - optional

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
GET	/funfact	cloudFunFactsCreate

Add route Remove Cancel Review and create Previous Next

~ Stages – Default.

Define stages - optional

Configure stages Info

Stages are independently configurable environments that your API can be deployed to. You must deploy to a stage for API configuration changes to take effect, unless that stage is configured to autodeploy. By default, all HTTP APIs created through the console have a default stage named \$default. All changes that you make to your API are autodeployed to that stage. You can add stages that represent environments such as development or production.

Stage name: \$default Auto-deploy: Remove Add stage

Cancel Previous Next

API name and integrations

API name: Funfacts API IP address type: IPv4

Integrations: cloudFunFactsCreate (Lambda)

Routes

Routes: GET /funfact → cloudFunFactsCreate (Lambda)

Stages

Stages: \$default (Auto-deploy: enabled)

Edit Create Cancel Previous Next

4. Click “Create” to create API GATEWAY.

The screenshot shows the AWS API Gateway console for the 'FunfactsAPI' (ljmj29fbz4). The main view displays the 'API details' section, which includes the API ID (ljmj29fbz4), Protocol (HTTP), and a 'Default endpoint' (https://ljmj29fbz4.execute-api.ap-south-1.amazonaws.com). The 'Stages for FunfactsAPI (1)' section shows a single stage named '\$default' with an attached deployment labeled 'hxwdmw'. The ARN listed is arn:aws:apigateway:ap-south-1::apis/ljmj29fbz4. The left sidebar provides navigation links for APIs, Develop, and Deploy.

👉 What's happening here?

- API Gateway acts as the front door to your Lambda.
- When someone hits /funfact with a GET request, API Gateway forwards it to Lambda, then returns the response.

Step - 5: Copy Endpoint:

1. In your API → go to Stages → Select **default**
2. Copy the Invoke URL (e.g., https://abcdedf.execute-api.ap-south-1.amazonaws.com).

The screenshot shows a browser window with the address bar containing the URL 'ljmj29fbz4.execute-api.ap-south-1.amazonaws.com'. The page content area displays a JSON response from the Lambda function, indicating that AWS regions have multiple Availability Zones for reliability. The browser interface includes a tab bar with various video and search results, and a bookmark bar at the bottom.

The screenshot shows the Postman application interface. At the top, there is a header bar with a magnifying glass icon, a dropdown menu set to "GET", the URL "https://ljmj29fbz4.execute-api.ap-south-1.amazonaws.com/funfact", and a blue "Send" button. Below the header, a sidebar on the left contains icons for Query (highlighted with a blue border), Headers, Auth, Body, Tests, and Pre Run, along with an "aws" icon. The main area is titled "Query Parameters" and shows a table with one row: a checkbox next to "parameter" and an empty "value" field. Below this, the status of the request is displayed as "Status: 200 OK", "Size: 73 Bytes", and "Time: 368 ms". The "Response" tab is selected, showing the JSON response: { "fact": "Amazon originally created S3 to solve its own scaling issues." }. Other tabs like "Headers", "Cookies", "Results", and "Docs" are also visible.

3. Run the test > Save and Click Invoke.

→ You should see a random fun fact returned in the response.

The screenshot shows the AWS Lambda console interface. At the top, there's a navigation bar with various tabs like 'Portfolio | Adityacprtm | D...', 'Marika Bergman | Full-stack...', 'AWS Cloud Resume Challenge...', 'Activity | Marika B. | LinkedIn...', 'MoRoble/AWS-Projects...', and 'cloudFunFactsCreate | Fun...'. Below the navigation bar, the main content area shows a Lambda function named 'cloudFunFactsCreate'. A success message box at the top left says 'The test event "TestEvent" was successfully saved.' The code editor shows a snippet of Python code:

```
 17  AWS Regions have multiple availability zones for redundancy.
 18  "Amazon originally created S3 to solve its own scaling challenges, and now more than 80% of Fortune 500 companies use AWS."
 19  "Cloud helps startups scale globally without huge upfront costs."
 20  "Cloud helps startups scale globally without huge upf
```

The 'TEST EVENTS' section shows a single entry: 'TestEvent'. The 'Response' pane displays the JSON output of the test event:

```
{  "statusCode": 200,  "headers": {    "Content-Type": "application/json"  },  "body": "{\"fact\": \"Serverless means you never patch servers \u2014 AWS does it for you!\"}"}
```

Below the code editor, there are sections for 'Code properties' (Info), 'Encryption with AWS KMS customer managed KMS key' (Info), and environment variables. The bottom of the page includes standard AWS footer links: CloudShell, Feedback, Console Mobile App, © 2025, Amazon Web Services, Inc. or its affiliates., Privacy, Terms, and Cookie preferences.

👉 What's happening here?

- You're manually invoking the Lambda in the console
- AWS runs your code in a temporary environment, then destroys it (that's the beauty of serverless).