

1 - Deploy Backend with AWS Lambda + API Gateway.

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

The screenshot shows the AWS Lambda homepage. At the top, there's a search bar and a navigation bar with links like 'Console Home', 'EC2', 'VPC', 'S3', 'Elastic Beanstalk', and 'Route 53'. On the left, under 'Compute', there's a large heading 'AWS Lambda' with the subtext 'lets you run code without thinking about servers.' Below this, a box titled 'Get started' contains the text 'Author a Lambda function from scratch, or choose from one of many preconfigured examples.' with a 'Create a function' button. On the right, there's a 'How it works' section with tabs for '.NET', 'Java', 'Node.js' (which is selected), 'Python', 'Ruby', and 'Custom runtime'. A code snippet in Node.js is shown:

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

At the bottom of the page, there are links for 'CloudShell' and 'Console Mobile App', along with copyright information and links for 'Privacy', 'Terms', and 'Cookie preferences'.

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 'Create function' wizard. At the top, there's a search bar and a navigation bar with links like 'Console Home', 'EC2', 'VPC', 'S3', 'Elastic Beanstalk', and 'Route 53'. The path 'Lambda > Functions > Create function' is visible. The main section is titled 'Create function' with a 'Info' link. It says 'Choose one of the following options to create your function.' There are three radio buttons: 'Author from scratch' (selected), 'Use a blueprint' (unselected), and 'Container image' (unselected). The 'Author from scratch' option has a sub-note 'Start with a simple Hello World example.' Below this, there's a 'Basic information' section. Under 'Function name', the value 'cloudFunFactsCreate' is entered. A note says 'Function name must be 1 to 64 characters, must be unique to the Region, and can't include spaces. Valid characters are a-z, A-Z, 0-9, hyphens (-), and underscores (_).' Under 'Runtime', the value 'Python 3.14' is selected. A note says 'Choose the language to use for your function. Note that the console code editor supports only Node.js, Python, and Ruby.' Under 'Architecture', the values 'arm64' and 'x86_64' are listed. A note says 'Choose the instruction set architecture you want for your function code.' At the bottom, there are links for 'CloudShell', 'Feedback', and 'Console Mobile App', along with copyright information and links for 'Privacy', 'Terms', and 'Cookie preferences'.

4. Permissions: Create a new role with basic Lambda permissions.

The screenshot shows the AWS Lambda 'Create function' wizard. The current step is 'Permissions'. It asks if you want to 'Create a new role with basic Lambda permissions'. 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 says '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 is a large orange 'Create function' button.

5. Click “Create Function”.

👉 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.

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 subscription",
            "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 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 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 2009",
            "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})
        }
    
```

Event Name: TestEvent
Invocation type: Synchronous
Event sharing settings: Private
Template - optional: Hello World
Event JSON: { "fact": "AWS S3 was launched in 2006 and still rules cloud storage"}

👉 What this does:

- 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.

```

    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 subscription",
            "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 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 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 2009",
            "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})
        }
    
```

Code properties: info

👉 What's happening here?

- Picks a random fun fact from our list and 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.

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.

The screenshot shows the AWS Lambda console interface. In the center, there's a modal window titled "The test event 'TestEvent' was successfully saved." It contains the following text:
AWS Regions have multiple Availability zones for redundancy.
Amazon originally created S3 to solve its own scaling needs.
More than 80% of Fortune 500 companies use AWS.
Cloud helps startups scale globally without huge upfront costs.
Below this, there's a code editor area with tabs for "TEST EVENTS (SELECTED: TESTEVENT1)" and "ENVIRONMENT VARIABLES". The "TEST EVENTS" tab shows a single entry named "TestEvent". The "ENVIRONMENT VARIABLES" tab is empty. To the right of the code editor, there's a "Response" pane displaying the JSON output of the test execution:

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

At the bottom of the modal, there are "Deploy" and "Test" buttons. The "Test" button is highlighted. Below the modal, the main Lambda function page shows the function name "cloudFunFactsCreate" and a "Code properties" section with details like package size (1,008 bytes), SHA256 hash (uGZyFShhg7A+rIBW4sOOiwWxpfa08owKCqZhu3KIL4E=), and last modified (2 minutes ago). A note indicates that the function is encrypted with an AWS KMS customer-managed KMS key.

👉 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.

The screenshot shows the AWS API Gateway landing page. At the top, there's a banner with the text "Networking & Content Delivery" and "API Gateway" followed by "Create and manage APIs at scale". Below the banner, a sub-banner states: "Amazon API Gateway is a fully managed service that makes it easy for developers to create, publish, maintain, monitor, and secure APIs." To the right of the sub-banner, there's a "Get started" section with a "Create an API" button. The main content area is divided into several sections:

- How it works:** A diagram illustrating the architecture. It shows various clients (Browser, Mobile, Webhooks, IoT devices) connecting to an "Amazon API Gateway" instance. This instance then routes requests to an "API Gateway cache" or directly to "AWS Lambda", "Amazon Kinesis", "Amazon ElastiCache", "Amazon DynamoDB", "Amazon Simple Queue Service", or "Amazon Data Pipeline".
- Pricing:** Text explaining that users only pay when their APIs are in use, with no minimum fees or upfront commitments. It specifies pricing for HTTP and REST APIs based on API calls received and data transferred, and for WebSocket APIs based on message count and duration.
- Resources:** A link to a page listing available resources.

At the bottom of the page, there are standard AWS navigation links: CloudShell, Feedback, and Console Mobile App on the left, and Copyright notice, Privacy, Terms, and Cookie preferences on the right.

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.

Step 3 - optional

- Define stages
- Step 4**
- Review and create

API name
An HTTP API must have a name. The name is a non-unique value you use to identify and organize your APIs. To programmatically refer to this API, use the API ID that API Gateway generates for you.

FunfactsAPI

IP address type [IPv4](#)
Includes only IPv4 addresses.

[Dualstack](#)
Includes IPv4 and IPv6 addresses.

Integrations (1) [Info](#)
Specify the backend services that your API will communicate with. These are called integrations. For a Lambda integration, API Gateway invokes the Lambda function and responds with the response from the function. For an HTTP integration, API Gateway sends the request to the URL that you specify and returns the response from the URL.

AWS Region	Lambda function	Version	Learn more
ap-south-1	awslambda:ap-south-1:820314685364:function:cloudFunf	2.0	Remove

[Add integration](#)

[Cancel](#) [Review and create](#) **Next**

3. Add a route:

Method: GET

Path: /funfact

Step 1 [Configure API](#)

Step 2 - optional **Configure routes**

Step 3 - optional [Define stages](#)

Step 4 [Review and create](#)

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](#)

[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

Review and create

API name and integrations

API name: FunfactsAPI IP address type: IPv4

Integrations:

- cloudFunFactsCreate (Lambda)

Routes

Routes:

- GET /funfact → cloudFunFactsCreate (Lambda)

Stages

Stages:

- \$default (Auto-deploy: enabled)

4. Click “Create” to create API GATEWAY.

API Gateway

APIs

- Custom domain names
- Domain name access associations
- VPC links

API: FunfactsAPI...({ljmj29fbz4})

FunfactsAPI

API details

API ID: ljmj29fbz4	Protocol: HTTP	Created: 2025-11-19
Description: No Description	IP address type: -	Default endpoint: Enabled https://ljmj29fbz4.execute-api.ap-south-1.amazonaws.com
ARN: arn:aws:apigateway:ap-south-1::apis/ljmj29fbz4		

Stages for FunfactsAPI (1)

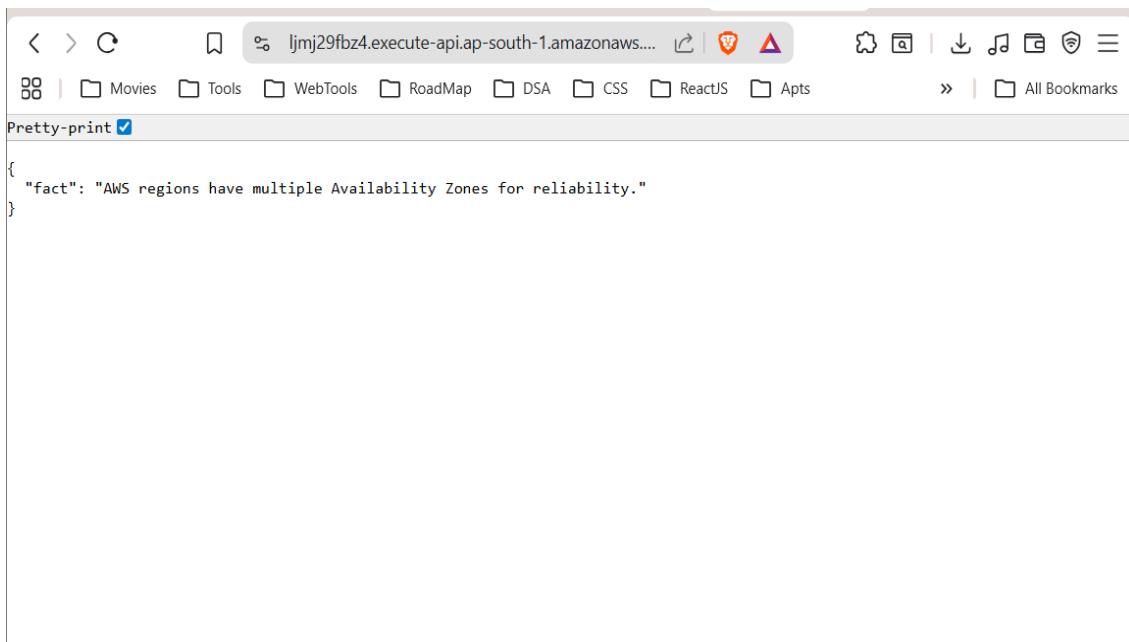
Stage name	Invoke URL	Attached deployment	Auto deploy	Last updated
\$default	https://ljmj29fbz4.execute-api.ap-south-1.amazonaws.com	hxwdmw	enabled	2025-11-19

👉 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>).



A screenshot of a browser window displaying a JSON response. The URL in the address bar is `ljjmj29fbz4.execute-api.ap-south-1.amazonaws...`. The response body shows a single object with a key "fact":

```
{  
  "fact": "AWS regions have multiple Availability Zones for reliability."  
}
```

Open your browser/Postman and test:

<https://abcdedf.execute-api.ap-south-1.amazonaws.com>

→ You should see something like:

```
{  
  "fact": "The name 'Amazon Web Services' was first used back in 2002."  
}
```

The screenshot shows the Postman application interface. At the top, there is a header bar with a search icon, a dropdown menu set to 'GET', a URL field containing 'https://ljmj29fbz4.execute-api.ap-south-1.amazonaws.com/funfact', and a blue 'Send' button. Below the header, a navigation bar has 'Query' selected, with other tabs like 'Headers 2', 'Auth', 'Body', 'Tests', and 'Pre Run'. To the left of the main area is a sidebar with icons for search, query parameters, headers, body, tests, pre-run, and AWS integration. The main content area displays a successful API call result. It shows 'Status: 200 OK', 'Size: 73 Bytes', and 'Time: 368 ms'. Below this, a 'Response' tab is selected, showing the JSON response body:

```
1  {
2     "fact": "Amazon originally created S3 to solve its own scaling
      issues."
3 }
```

Outcome of Stage - 1:

At this point, you've built a fully functional serverless API:

- ~ A **Python Lambda** that returns random cloud facts.
- ~ An **API Gateway endpoint** that makes it accessible over the internet.