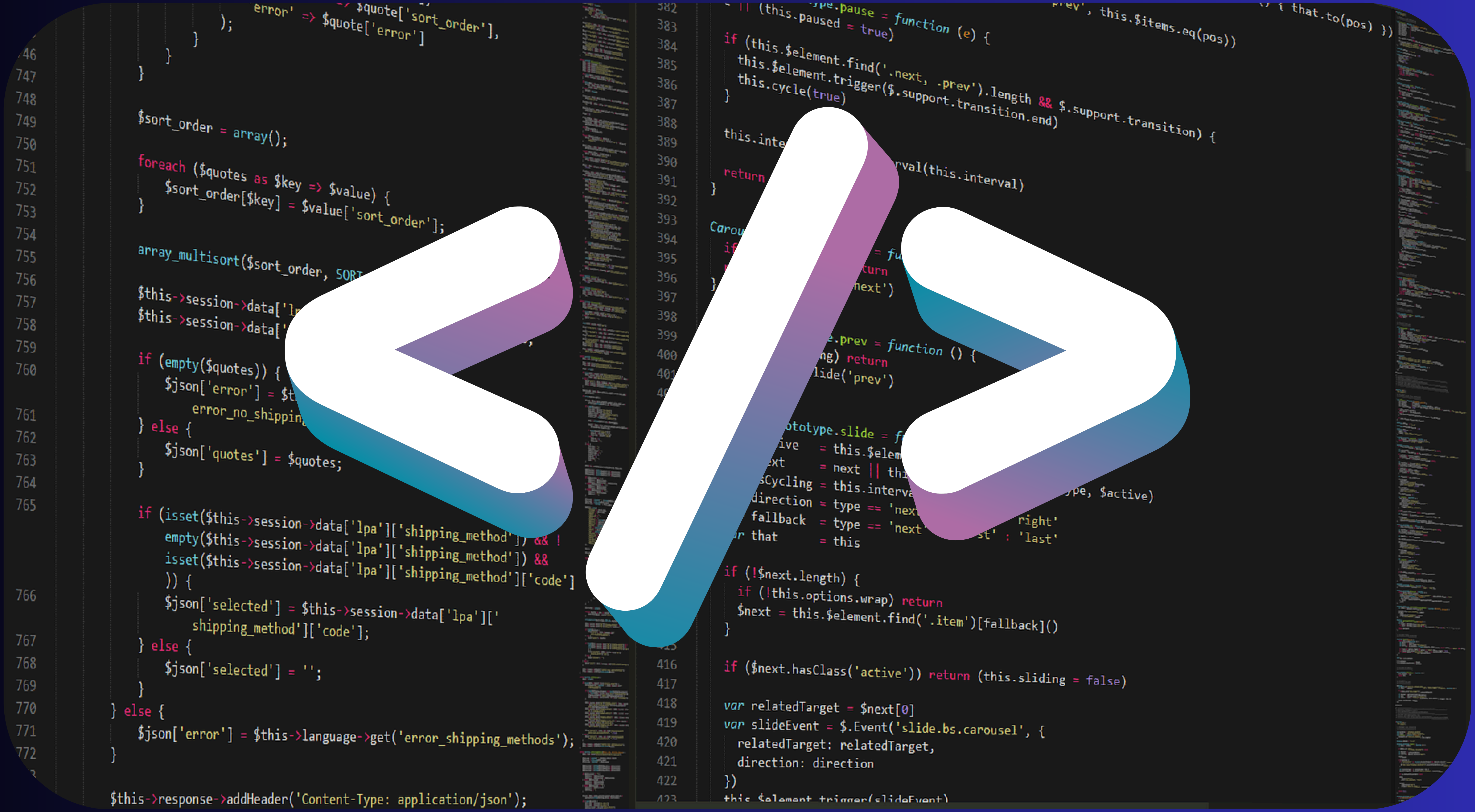
# AWS Lambda

lets you run code without thinking about servers.

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.



```
                    'error' => $quote['error']
                );
            }
        }
    }

    $sort_order = array();

    foreach ($quotes as $key => $value) {
        $sort_order[$key] = $value['sort_order'];
    }

    array_multisort($sort_order, SORT

    $this->session->data['l
    $this->session->data['

    if (empty($quotes)) {
        $json['error'] = $t
            error_no_shippin
    } else {
        $json['quotes'] = $quotes;
    }

    if (isset($this->session->data['lpa']['shipping_method']) && !
        empty($this->session->data['lpa']['shipping_method']) &&
        isset($this->session->data['lpa']['shipping_method']['code'])
        )) {
        $json['selected'] = $this->session->data['lpa']['
            shipping_method']['code'];
    } else {
        $json['selected'] = '';
    }
} else {
    $json['error'] = $this->language->get('error_shipping_methods');
}

$this->response->addHeader('Content-Type: application/json');
```

```
  || (this.paused = true)        ype.pause = function (e) {
  if (this.$element.find('.next, .prev').length && $.support.transition) {
    this.$element.trigger($.support.transition.end)
    this.cycle(true)
  }

  this.inte                rval(this.interval)

  return
}

Carou
  if                 = fu
                     turn
}                    next')

                     e.prev = function () {
                     ng) return
                     lide('prev')

                     ototype.slide = f
                     ive    = this.$elem
                     xt     = next || thi
                     sCycling = this.interva
                     direction = type == 'nex
                     fallback  = type == 'next'
  r that             = this

  if (!$next.length) {
    if (!this.options.wrap) return
    $next = this.$element.find('.item')[fallback]()
  }

  if ($next.hasClass('active')) return (this.sliding = false)

  var relatedTarget = $next[0]
  var slideEvent = $.Event('slide.bs.carousel', {
    relatedTarget: relatedTarget,
    direction: direction
  })
```

# What is Lambda?

AWS Lambda is a serverless computing service that lets you run code in response to events without managing servers.

You just upload your code, and AWS automatically handles the rest, scaling as needed and only charging for the time your code runs.

# When to use Lambda?

- "**Image Processing**: Let’s say users upload images to your app. You can set up Lambda to automatically resize, compress, or even apply filters to each image as it’s uploaded."

- "**Data Transformation**: If you need to clean up or process data before storing it in a database, a Lambda function can handle that transformation automatically."

- "**Real-Time Notifications**: If an event happens—like a new user signing up—you can use Lambda to trigger an email, SMS, or other notifications instantly."

# Event-Driven Execution:

- Lambda is an event-driven service, meaning that it runs your code in response to certain triggers or events.

- These events can come from many different AWS services like
  - S3 (file uploads),
  - DynamoDB (database changes),
  - API Gateway (HTTP requests),
  - CloudWatch (scheduled events), etc.

## Automatic Scaling:

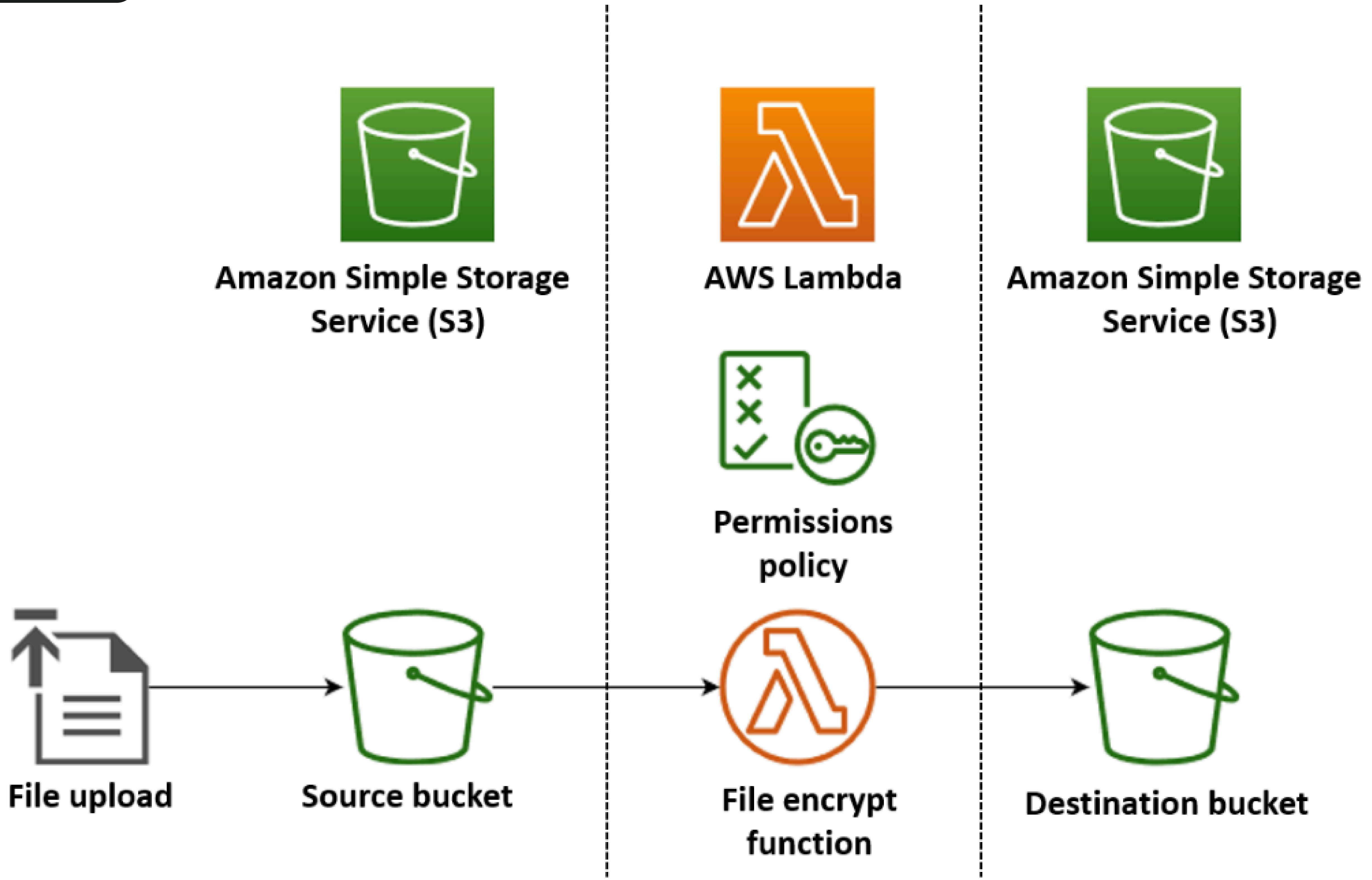
- AWS Lambda automatically scales the execution of functions in response to the number of incoming requests.
- If a thousand requests come in at the same time, AWS Lambda will handle them in parallel, making it highly scalable without any configuration.

# Pay-as-You-Go:

- Lambda uses a pay-as-you-go pricing model. You are billed based on the number of function executions and the duration of each function's runtime (measured in milliseconds).
- This is cost-effective as you only pay for the compute time that you use, and there's no charge for idle time.

# AWS Lambda language support

- Node.js (JavaScript)
- Python
- Java
- C# (.NET Core)
- Ruby
- Custom Runtime API

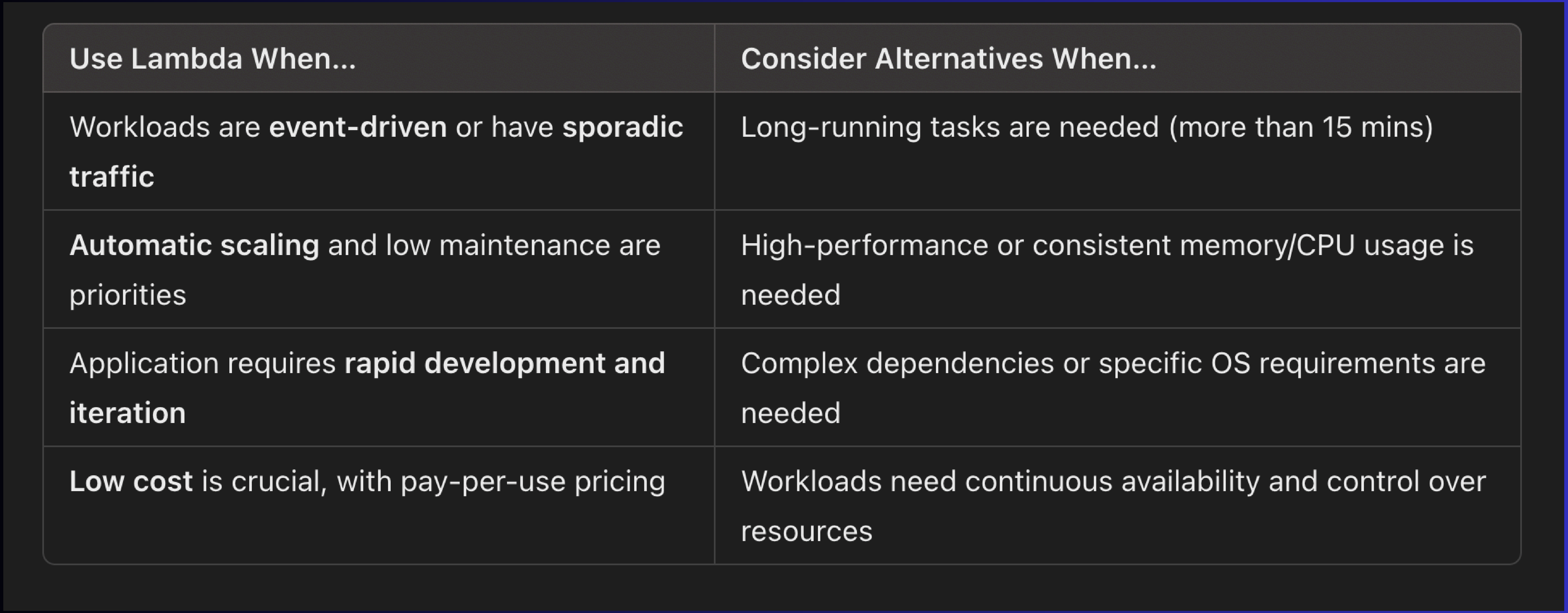

Amazon Simple Storage Service (S3)
AWS Lambda
Amazon Simple Storage Service (S3)
Permissions policy
File upload
Source bucket
File encrypt function
Destination bucket

# AWS Lamda Limitations

- **Execution Time Limit**: Lambda functions can only run for a maximum of 15 minutes. If you need longer-running tasks, Lambda might not be the best choice.

- **Stateless**: Lambda functions don’t keep state between invocations, so they’re best for tasks that don’t require long-term memory.

- **Cold Start Delays**: If a Lambda function hasn’t run in a while, there’s sometimes a slight delay—called a 'cold start'—when it starts up. This can add a little latency, but AWS provides ways to mitigate it for critical functions.



| Use Lambda When... | Consider Alternatives When... |
|---|---|
| Workloads are **event-driven** or have **sporadic traffic** | Long-running tasks are needed (more than 15 mins) |
| **Automatic scaling** and low maintenance are priorities | High-performance or consistent memory/CPU usage is needed |
| Application requires **rapid development and iteration** | Complex dependencies or specific OS requirements are needed |
| **Low cost** is crucial, with pay-per-use pricing | Workloads need continuous availability and control over resources |