SERVERLESS BEST PRACTICES



The oldest deployment framework around today for serverless applications is known as the Serverless Framework. This deployment framework has its own set of best practices for securing your serverless deployment.

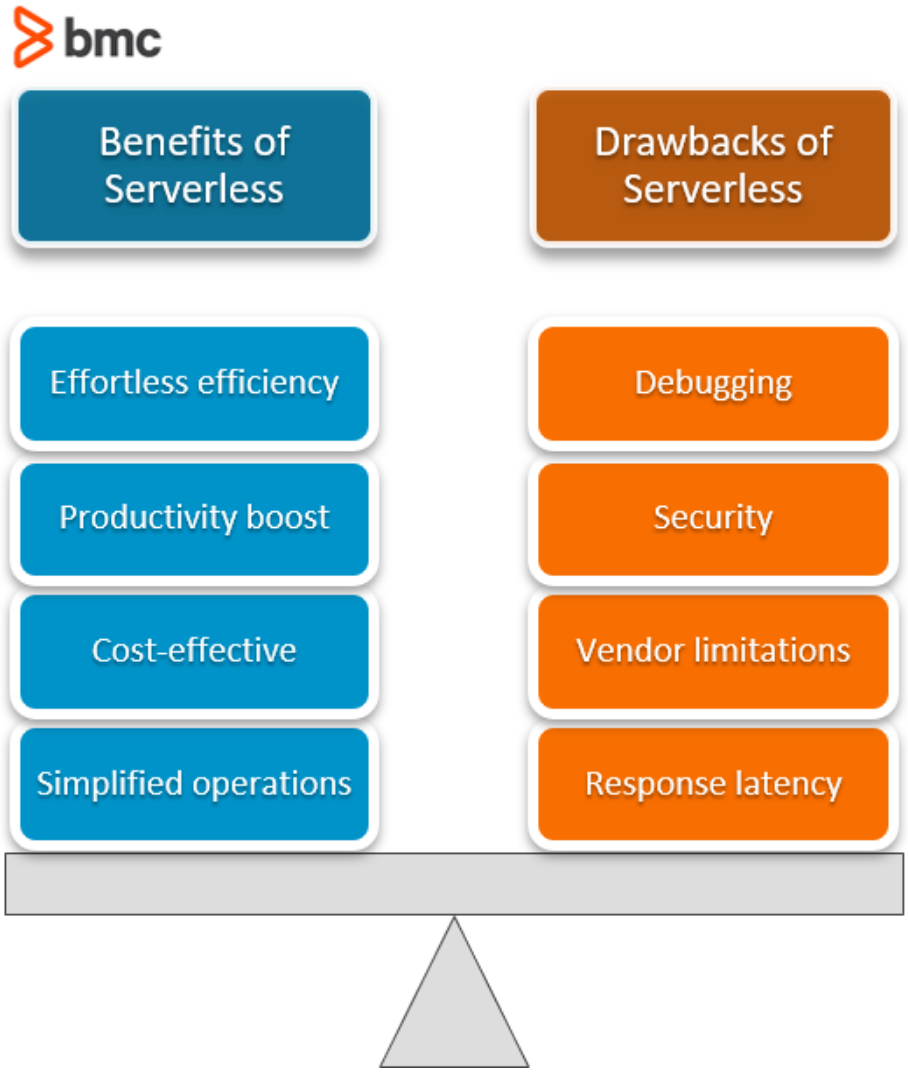
This article will review the Serverless Framework and outline the best practices for developing and deploying serverless apps.

Developing serverless apps

Serverless computing is a way for developers to build apps without the headache of managing infrastructure. More specially, it enables them to write in serverless code without having to:

- Provision a server
- Ensure its functionality
- Create test environments on a server
- Maintain server uptime

This frees up teams and resources to focus their attention on accelerating innovation in today's competitive digital economy.



What's the Serverless Framework?

One particular framework is known as the Serverless Framework. This open-source framework is free, of course, and written using Node.js. Serverless is the first framework developed for building applications on AWS Lambda, a serverless computing platform that's part of the Amazon Web Services suite.

A couple of lambda functions can accomplish some simple tasks through a serverless app or an entire back-end composed of hundreds of lambda functions. Serverless supports all runtimes offered within your chosen cloud provider.

Keep in mind the best practices explained below are not the only practices. These practices all rely on a set of underlying assumptions. We'll talk specifically about Lambda, simply because it's so popular. But remember that Lambda isn't the only serverless option.

Serverless best practices

When working with the Serverless Framework, here are the top best practices worth adopting to ensure your applications are secure and robust.

Start locally

When your Lambda starts to get complicated, app developers find themselves making numerous tweaks to configurations and functions code. Each change leads to waiting for the code to deploy and the rest of the stack to go live.

To save yourself time, start locally.

Instead of operating your dev cycle as "save, open the console, deploy, wait, refresh, wait, refresh," shift the cycle to "save, build, refresh." The AWS Serverless Application Model directs line interface can replicate lambdas and API endpoints and several sources—all in a local Docker container.

Use 1 function per route

When using HTTP, it is a best practice to avoid single function proxy. Why? This method does not:

- Scale well
- Isolate issues

If the function of a series of routes ties strictly to a single table, it is decoupled from the application. This best practice may add a level of complexity for management, but it helps isolate errors and issues when scaling your app.

Don’t rewrite your code

JavaScript is the higher-order language in a Serverless Framework. When writing JavaScript code, it will be interpreted and executed in the most performative way possible.

Rewriting code is not the way to solve a performance problem. Instead, there are many ways to improve your program:

- Combining multiple requests to other services
- Stopping loops when you have enough matches
- Returning helpful error information



Manage code, not configurations

The Serverless Framework lets you pass a dedicated deployer role to AWS CloudFormation to run deployments with the architecture. CloudFormation deployment role is the AWS path out of 'config that is only stored in the UI' their Serverless Application Model lets you create YAML that defines your stack in a file you can use to track changes.

You can use a different role for each project or team and apply the least privilege principle to the deployment pipeline.

Perform load testing

Load testing your lambda functions will help determine the amount of memory to allocate and the optimum timeout value.

There may be complex applications in a serverless environment, and you may not be aware of dependencies within your applications that could not perform a function on a large scale. Load testing allows you to catch potential issues that may be vital to maintaining a highly available application.

Deploy API gateways for security

Security should be a top priority for your applications, whether for a traditional architecture or serverless.

Implementing an API gateway as the event source for your lambda function is vital. It would be best if you secured API Gateway endpoints. This API Gateway provides several options for securing your API.

Serverless supports agile software design

Developing applications with a Serverless Framework is the modern approach to application development. There are other best practices to explain when using a Serverless Framework, and over time, you will find what practices best serve you and your project goals.

As your project progresses, your users can tell you what needs to be improved.

Related reading

- BMC DevOps Blog
- The AWS Well-Architected Framework: 5 Pillars & Best Practices, part of our AWS Guide
- Application Performance Management in DevOps
- Bring Kubernetes to the Serverless Party
- Docker Security: 14 Best Practices for Securing Docker Containers
- AWS Serverless Application Model