Gmail - ⚡ Serverless Framework

mail.google.com/mail/u/0

AWS for the Real World
Not Just for Certifications

👀 *This is not properly displayed? Read all of our issues online!* 💡

Hi Chidvi 👋🏾

Let's start with the obvious: **We love AWS Lambda!** 💛

And we're pretty sure you too, regardless of whether you're a long-term AWS Fundamentals subscriber or just recently joined. 🙏

That's why we want to dedicate this newsletter to the fastest IaC to launch Serverless applications on AWS: **The Serverless Framework**

Citing **DataDog**'s The State of Serverless:

*[...] Serverless Framework is the **most popular IaC tool** for managing AWS Lambda functions among Datadog customers.*

Even more:

*It provides streamlined opinionated workflows that **make it straightforward to deploy applications and serverless infrastructure together**. This, along with the strong community support it has among developers, has made it a popular and inviting entry point for newcomers and serverless-focused teams.*

Let's get going! 👾

Why Bother with Serverless Framework?

Creating the necessary infrastructure for a **simple service powered by Lambda** and **exposed to the internet via API Gateway** can be a complex task.

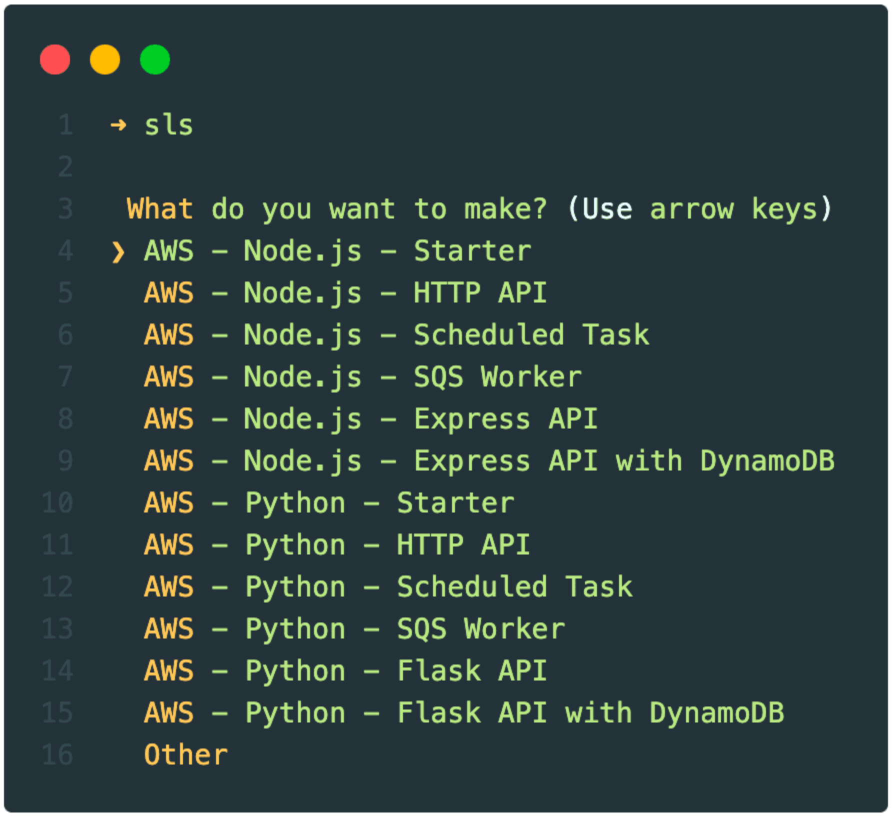
While it is possible to manually configure everything in the AWS console, this approach is not recommended for serious projects due to its lack of reproducibility and potential for errors.

Serverless Framework simplifies this tremendously. It **abstracts away much of the boilerplate configuration** needed for **Lambda**, **API Gateway**, and other native integrations with AWS services like **SQS** or **EventBridge**.

In addition, Serverless Framework handles the packaging of your code and offers a powerful CLI that allows you to easily deploy Lambda functions.

Serverless Templates

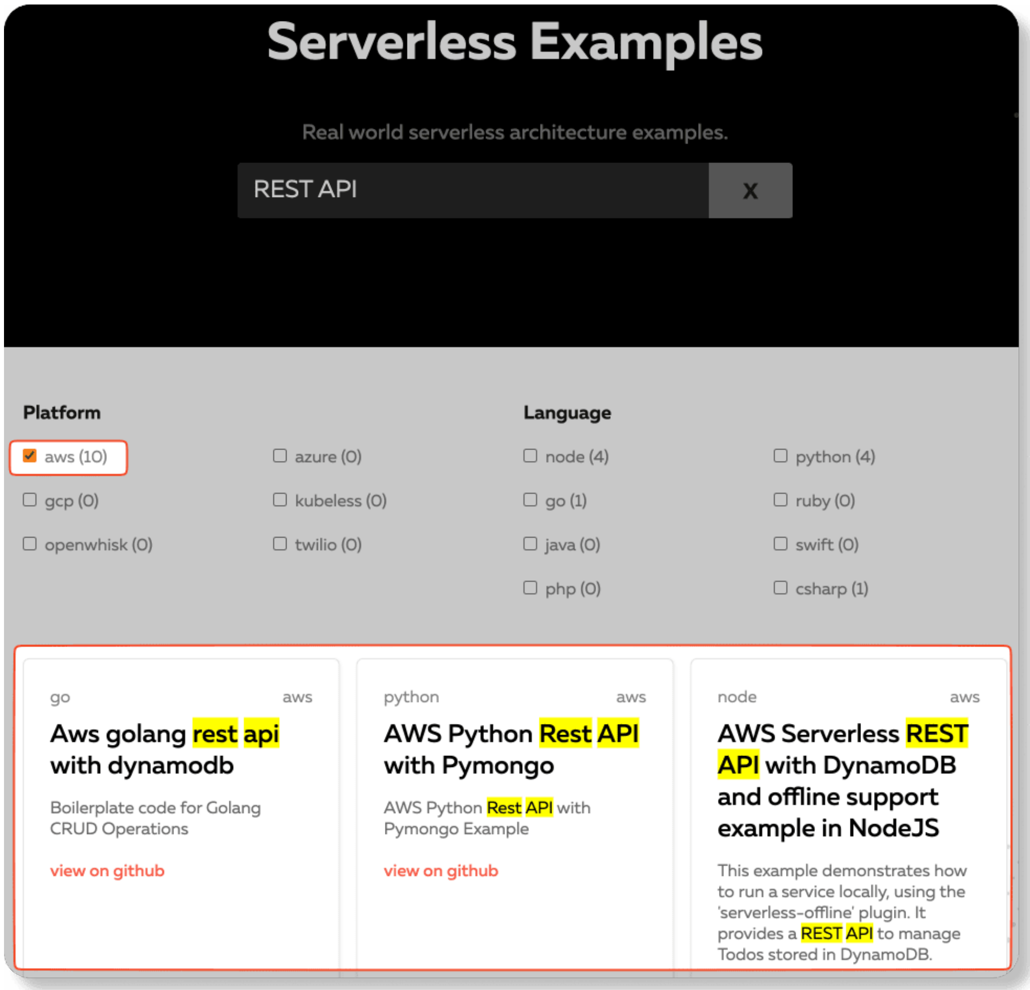
When running **sls** , you will be prompted to choose a **starter template**, so you can get things going very quickly.



```
➜ sls

 What do you want to make? (Use arrow keys)
❯ AWS - Node.js - Starter
  AWS - Node.js - HTTP API
  AWS - Node.js - Scheduled Task
  AWS - Node.js - SQS Worker
  AWS - Node.js - Express API
  AWS - Node.js - Express API with DynamoDB
  AWS - Python - Starter
  AWS - Python - HTTP API
  AWS - Python - Scheduled Task
  AWS - Python - SQS Worker
  AWS - Python - Flask API
  AWS - Python - Flask API with DynamoDB
  Other
```

But that's not all offered by Serverless Framework.


Serverless Examples

Real world serverless architecture examples.

REST API

Platform
- [✓ aws (10)]
- azure (0)
- gcp (0)
- kubeless (0)
- openwhisk (0)
- twilio (0)

Language
- node (4)
- python (4)
- go (1)
- ruby (0)
- java (0)
- swift (0)
- php (0)
- csharp (1)

go — aws
Aws golang rest api with dynamodb
Boilerplate code for Golang CRUD Operations
view on github

python — aws
AWS Python Rest API with Pymongo
AWS Python Rest API with Pymongo Example
view on github

node — aws
AWS Serverless REST API with DynamoDB and offline support example in NodeJS
This example demonstrates how to run a service locally, using the 'serverless-offline' plugin. It provides a REST API to manage Todos stored in DynamoDB.

There's also a searchable set of Real-world (😏) serverless architecture examples. 🏗️

Abstraction in Perfection

Serverless Framework provides a wide range of tools and features that not only simplify the process of setting up intricate infrastructure configurations but also incorporate industry best practices seamlessly.

Let's explore its concepts. 📚

Services

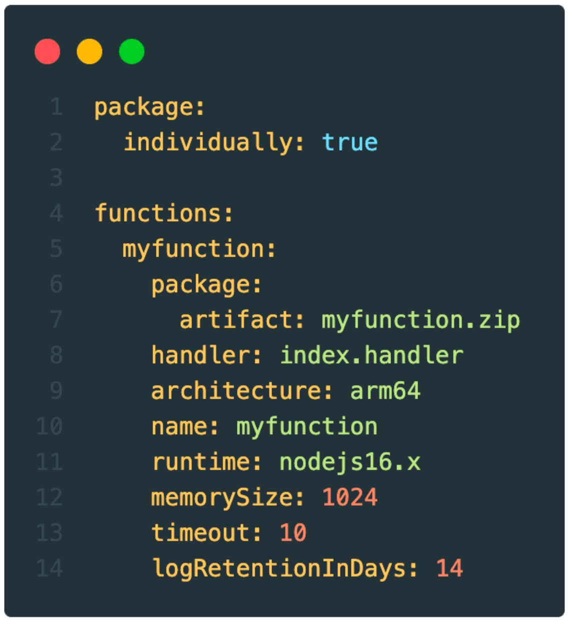
In the Serverless Framework, a service is a fundamental unit of organization that is defined by the **serverless.yml** configuration file (or **serverless.ts** if you want to use **TypeScript**!).

This configuration file is then transformed into a CloudFormation template, which is used to create the necessary resources.

To create a new service configuration, you can simply run the **sls** command without any parameters. 🏗️

Functions

Functions serve as the compute layer in your serverless architecture, running on **AWS Lambda**.



```yaml
package:
  individually: true

functions:
  myfunction:
    package:
      artifact: myfunction.zip
    handler: index.handler
    architecture: arm64
    name: myfunction
    runtime: nodejs16.x
    memorySize: 1024
    timeout: 10
    logRetentionInDays: 14
```

They operate independently in terms of execution and deployment. While they typically serve a single purpose (**small, dedicated functions**), they are not limited to it and can perform multiple tasks if needed (**fat functions**).

Serverless Framework also offers easy configuration for:

- VPCs for private networks
- Lambda Layers for externalizing dependencies
- Destinations
- Dead Letter Queues for failed invocations
- Versioning of functions

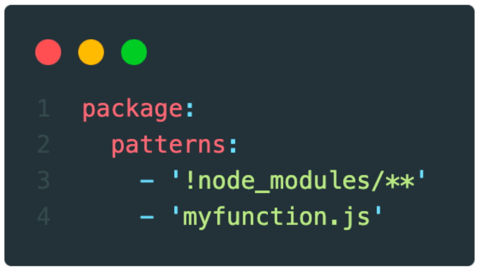
- X-Ray & tracing

… and much more.

Packaging

In the previous instance, we discussed an externally packaged deployment unit as an example.

However, Serverless also offers the capability to **package your function's code internally**.



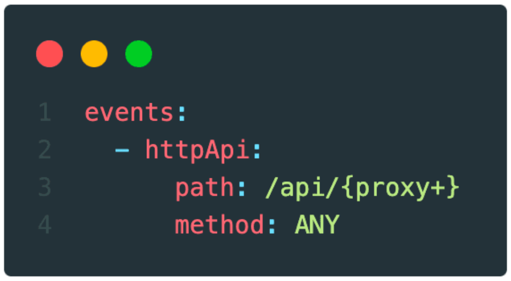
```yaml
package:
  patterns:
    - '!node_modules/**'
    - 'myfunction.js'
```

Rather than relying on an artifact, you can utilize patterns to specify the files that should be included in your deployment unit ZIP file.

Triggers

Serverless simplifies the process of creating triggers for your functions through events such as HTTP calls or messages.

It automatically creates the required infrastructure to seamlessly connect these **events** to your **functions**. 🤝



```yaml
events:
  - httpApi:
      path: /api/{proxy+}
      method: ANY
```

💡 **Example**: An HTTP event is received through the Gateway, specifically for the `/api/*` path.

Community Plugins

Serverless is an incredibly robust tool that comes equipped with a wide range of built-in features.

However, there's even more potential via the community plugins available! 👥

Naming just a few prominent ones:

- **Resource Tagging** (`serverless-plugin-resource-tagging`) - easily add default tags for all of your template's resources.
- **Domain Manager** (`serverless-domain-manager`) - add custom domain names for your API Gateways without any effort.
- **If/Else** (`serverless-plugin-ifelse`) - create resources based on conditions, e.g. if you don't want to have certain resources only in specific regions or stages.
- **AWS Alerts** (`serverless-plugin-aws-alerts`) - create CloudWatch alerts in just a few lines of code.
- **Step Functions** (`serverless-step-functions`) - create Step Functions event workflows.

The plugin ecosystem is vast, offering a wide range of plugins to cater to almost any use case.

Much More to Explore

In this brief overview, we have only scratched the surface of the **Serverless Framework** and its capabilities. There is so much more to explore and learn about this powerful tool.

For a more comprehensive understanding, we recommend diving into our dedicated section on the Serverless Framework in our **AWS Fundamentals book**. 📙

So, if you're hungry for more information and eager to expand your understanding, be sure to check it out! As a newsletter subscriber, you're always in for a 15% discount on all of our products 😊

Beyond AWS Fundamentals

We wanted to give a shoutout to **Alex DeBrie**, an AWS Hero with a knack for **DynamoDB**, serverless applications, and cloud-native technologies. Whether you're a newbie or a seasoned AWS user, Alex's expertise can help you and your team make the most of AWS.

One thing that sets Alex apart is his ability to explain complex concepts in a simple and approachable way. He's also a fantastic writer, with articles like How you should think about DynamoDB costs and Event-Driven Architectures vs. Event-Based Compute in Serverless Applications showcasing his deep understanding of AWS services and his practical solutions to common challenges.

If you are looking to work with **DynamoDB** in the future, we highly recommend buying and reading “The DynamoDB Book”. This comprehensive resource provides valuable insights and guidance on effectively utilizing DynamoDB. It covers various topics, including best practices, advanced techniques, and practical examples, making it the perfect reference.

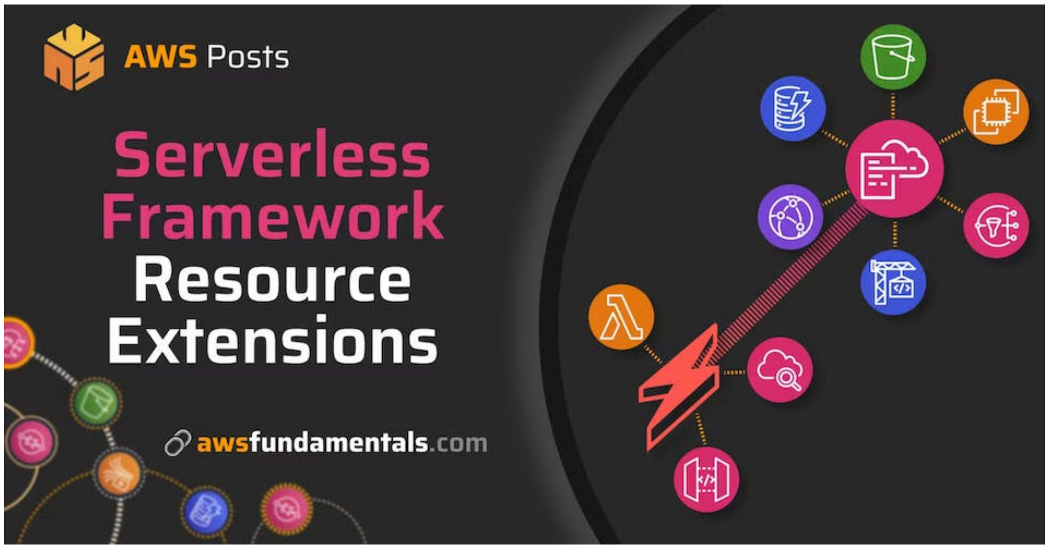
Whether you are a beginner or an experienced developer, it will greatly enhance your understanding and proficiency in working with DynamoDB.

We'll also finish this issue with Alex by passing you one of his articles about **Serverless Framework** vs. **AWS CDK**.

Wishing you a **great week** & **happy learning**! 📚

Sandro & Tobi 👋

Still hungry for AWS content? Have a look at our blog! 📚 ↓



AWS Posts
AWS
Lambda
Pricing
awsfundamentals.com
AWS Lanbda