Serverless deployments with Terraform

Michael Bright, Terraform Associate

for ArdanLabs, 2021-Sep-7

https://linkedin.com/in/mjbright @mjbright @mjbright

Agenda

Resources

Resources from the session will be added at

https://github.com/mjbright/webinars

- These slides
- Source code repo

Agenda

Serverless deployments with Terraform

Intro: Infra as Code / Terraform

Serverless concept

AWS “Serverless services”

Developing for AWS Lambda

Better together: Serverless & Servers
- Combined with EC2 – managing AWS resources

What is Infrastructure as Code ?

IaC

What is Infrastructure as Code ?

IaC

What is Infrastructure as Code (IaC) ?

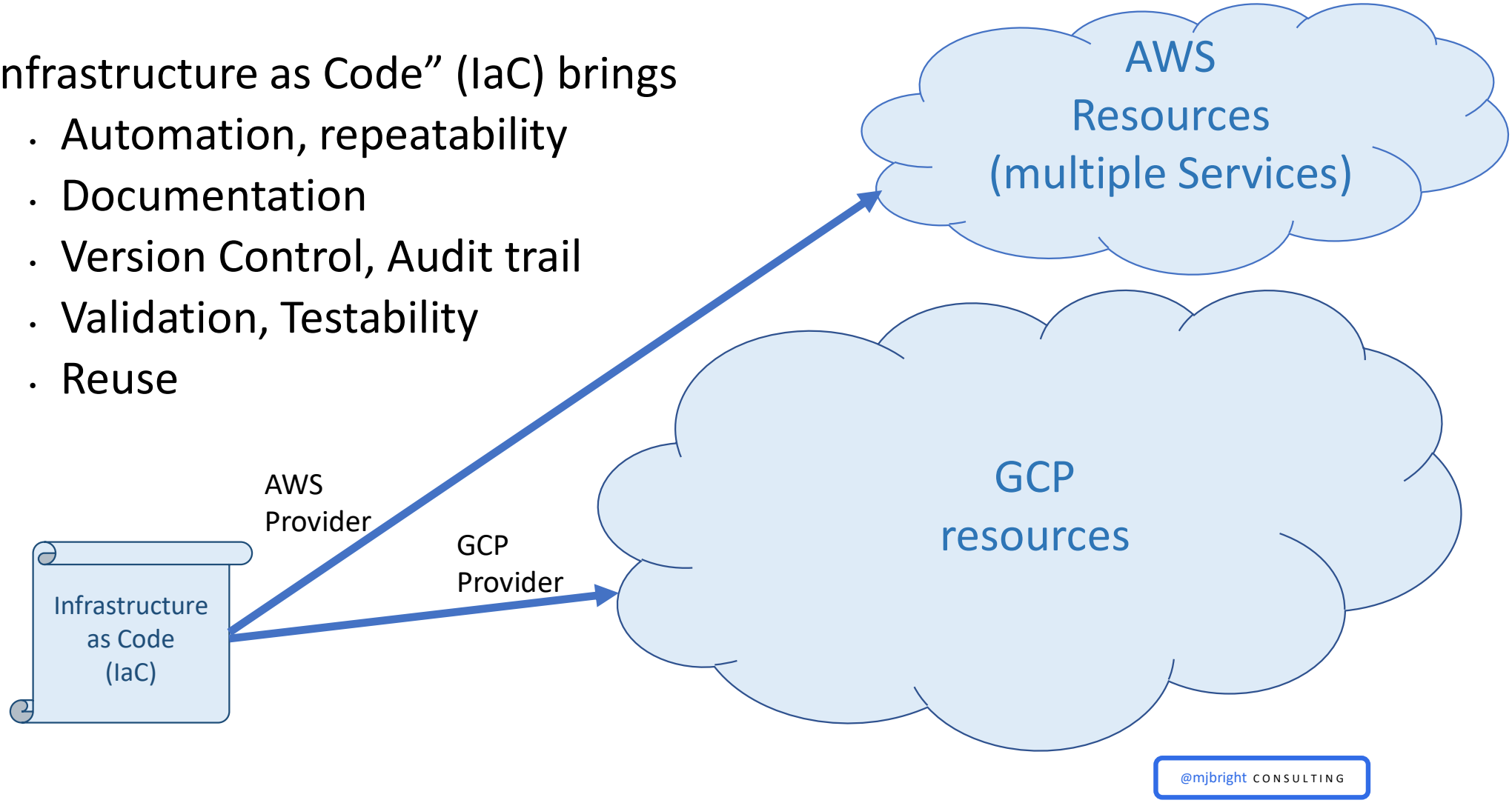
Why Infrastructure as Code ?

Why Terraform ?

Introduction

"Infrastructure as Code" (IaC) brings

- Automation, repeatability
- Documentation
- Version Control, Audit trail
- Validation, Testability
- Reuse



What is Serverless ?

Serverless = FaaS + BaaS

What is Serverless ?

Serverless = FaaS + BaaS

“Serverless is all about not being bothered about Servers …”

What is Serverless ?

Serverless = FaaS + BaaS

Serverless is both

- Functions as a Service
- Backend as a Service

Paying for what you use … when you use it …

What is Serverless ?

Serverless = FaaS + BaaS

Each major cloud provider has their own FaaS Service combined with backend services

- AWS Lambda
- Google cloud functions
- Azure functions

But also other “serverless” services

We’ll be looking at AWS Lambda and other services

AWS “Serverless services”

AWS “Serverless services”

https://aws.amazon.com/serverless/

The list of AWS serverless services is constantly updated.
So it's better to track the page to see currently available ones.

https://www.quora.com/Which-AWS-services-are-serverless

AWS “Serverless services”

AWS “Serverless services”

Compute
- Lambda
- Fargate

Data Storage
- S3
- Aurora Serverless
- DynamoDB
- RDS Proxy

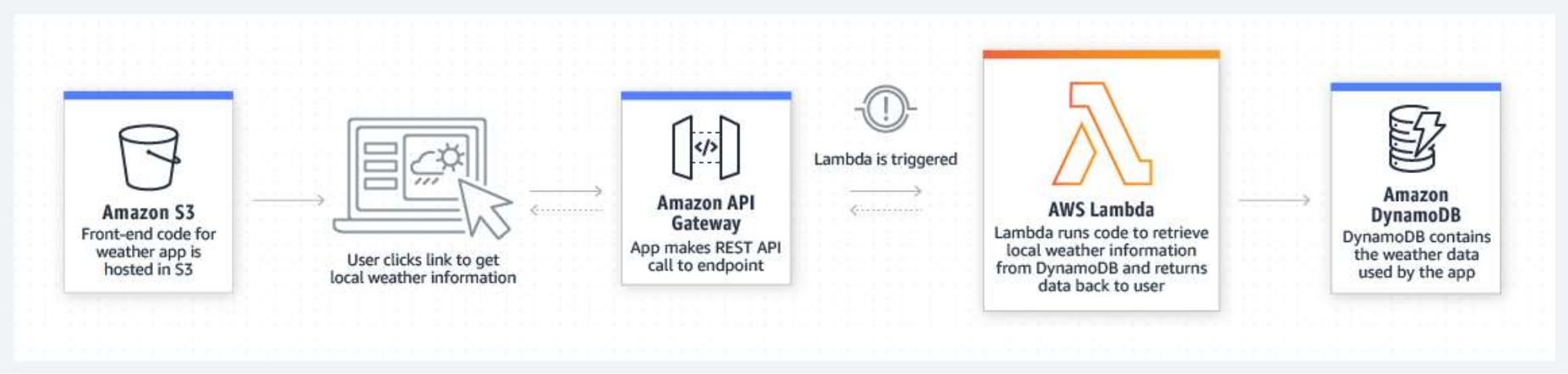
Integration
- SQS
- SNS
- API Gateway
- EventBridge
- AppSync
- StepFunctions

AWS “Serverless services”

Compute

AWS “Serverless services”

Compute: AWS Lambda



AWS Lambda is FaaS: Functions as a Service

- No servers to manage
- Continuous scaling
- Cost optimized with msec metering
- Consistent performance at any scale

https://aws.amazon.com/lambda/

AWS “Serverless services”

Compute: AWS Lambda

Requests	$0.20 per 1M requests
Duration	$0.0000166667 for every GB-second

Memory (MB)	Price per 1ms
128	$0.0000000021
512	$0.0000000083

1 million requests free
per month with the AWS Free Tier

https://aws.amazon.com/lambda/

AWS “Serverless services”

Compute: Fargate

Fargate provides serverless compute engines for containers on
Amazon Elastic Container Service (ECS) or Amazon Elastic Kubernetes Service (EKS)

Focus on application development, removing the need to provision & manage servers, paying for used resources per application.

Fargate allocates the right amount of compute, eliminating the need to choose instances & scale cluster capacity. Pay for the resources required to run your containers, so there is no over-provisioning and paying for additional servers.

Improves security through application isolation by design - runs each task or pod in its own isolated compute environment providing workload isolation & improved security.

https://aws.amazon.com/fargate

AWS “Serverless services”

Application Integration

AWS “Serverless services”

Integration: API Gateway

Create, publish, maintain, monitor, and secure REST & Websocket APIs at any scale.

Fully managed service allows to create, publish, maintain, monitor & secure APIs at any scale. API Gateway can create RESTful APIs & WebSocket APIs to enable real-time two-way communication applications. Supports containerized and serverless workloads, as well as web applications.

Handles tasks involved in accepting & processing up to hundreds of thousands of concurrent API calls, including traffic management, CORS support, authorization and access control, throttling, monitoring, and API version management.

No minimum fees or startup costs. Pay for the number of received API calls & amount of data transferred out.

1 million API calls received free
per month for 12 months with the AWS Free Tier

https://aws.amazon.com/api-gateway/

AWS “Serverless services”

Integration: SQS - Simple Queue Service

Fully managed message queue service enabling decoupling & scaling of microservices, distributed systems & serverless applications. Eliminates complexity & overhead of operating message-oriented middleware.

Send, store & receive messages between software components at volume without message loss or requiring other services to be available.

Offers 2 types of message queues:

- Standard - for maximum throughput, best-effort ordering & at-least-once delivery.
- SQS FIFO - guarantees messages are processed exactly once, in the exact order that they are sent.

1 million requests free per month with the AWS Free Tier

https://aws.amazon.com/sqs/

	Standard Queues (per Million requests)	FIFO Queues (per Million requests)
First 1 Million Requests/Month	Free	Free
From 1 Million to 100 Billion Requests/Month	$0.40	$0.50
From 100 Billion to 200 Billion Requests/Month	$0.30	$0.40
Over 200 Billion Requests/Month	$0.24	$0.35

AWS “Serverless services”

Integration: SNS - Simple Notification Service

Fully managed messaging service for app-to-app (A2A) & app-to-person (A2P) communication

- A2A pub/sub functionality provides topics for high-throughput, push-based, many-to-many messaging between distributed systems, microservices & event-driven serverless applications. Apps can fanout messages to a large number of subscriber systems including Amazon SQS queues, AWS Lambda functions and HTTPS endpoints, for parallel processing, and Amazon Kinesis Data Firehose

- A2P functionality enables sending messages to users at scale via SMS, mobile push, and email.

Standard topic requests include publishes, topic owner operations, and subscription owner operations

- First 1 million Amazon SNS requests per month are free, $0.50 per 1 million requests thereafter

Note: Each 64KB chunk of data is billed as 1 request. A publish with 256KB payload is billed as 4 requests.

1 million requests free
per month with the AWS Free Tier

https://aws.amazon.com/sns/

AWS “Serverless services”

Data Storage

AWS “Serverless services”

S3 – Simple Storage Service

Store any amount of data with scalability, availability, security & performance

S3 is an object storage service, which can be used to store & protect data for a range of use cases such as data lakes, websites, mobile applications, backup and restore, archive, enterprise applications, IoT devices & big data analytics.

S3 provides easy-to-use management features to organize data & configure finely-tuned access controls

S3 is designed for 99.999999999% (11 9's) of durability

https://aws.amazon.com/s3

AWS “Serverless services”

Serverless Databases: Aurora Serverless

Automatically scale capacity based on your application's need with Aurora Serverless

An *Aurora Serverless DB cluster* is an on-demand, auto-scaling configuration for Amazon Aurora - automatically starts, stops & scales capacity up/down based on your application needs.

Eliminates the need to manually provision & size Aurora DB clusters

Create a database endpoint, optionally specify the desired database capacity range, and connect your applications

Pay on a per-second basis for database capacity used when the database is active, migrate between standard and serverless configurations with a few clicks in the Amazon RDS Management Console.

https://aws.amazon.com/rds/aurora/serverless

AWS “Serverless services”

Serverless Databases: DynamoDB

Single-digit millisecond performance at scale with this key-value / document database

Key-value & document database delivering low-latency performance at scale.

Fully managed, multi-region, multi-active, durable database with built-in security, backup & restore, in-memory caching for internet-scale applications. DynamoDB can handle more than 10 trillion requests per day and can support peaks of more than 20 million requests per second.

Create a new table for your application and let DynamoDB handle the rest.

https://aws.amazon.com/dynamodb

AWS “Serverless services”

Serverless Databases: RDS Proxy

Increase scalability, resiliency & security with this proxy for Relational Database Service (RDS)

Fully managed, HA database proxy for Amazon Relational Database Service (RDS) that makes applications more scalable, more resilient to database failures, and more secure.

Applications, including serverless, can have many open connections to the database & open and close connections at a high rate

RDS Proxy allows applications to pool & share DB connections, improving efficiency & scalability. RDS Proxy reduces failover times for Aurora & RDS databases by up to 66%.

DB credentials, authentication & access can be managed through integration with AWS Secrets Manager and AWS Identity and Access Management (IAM).

RDS Proxy can be enabled with no code changes or provisioning of additional infrastructure.
Pricing is simple and predictable: pay per vCPU of the DB instance.
Generally available for Aurora MySQL, Aurora PostgreSQL, RDS MySQL and RDS PostgreSQL.

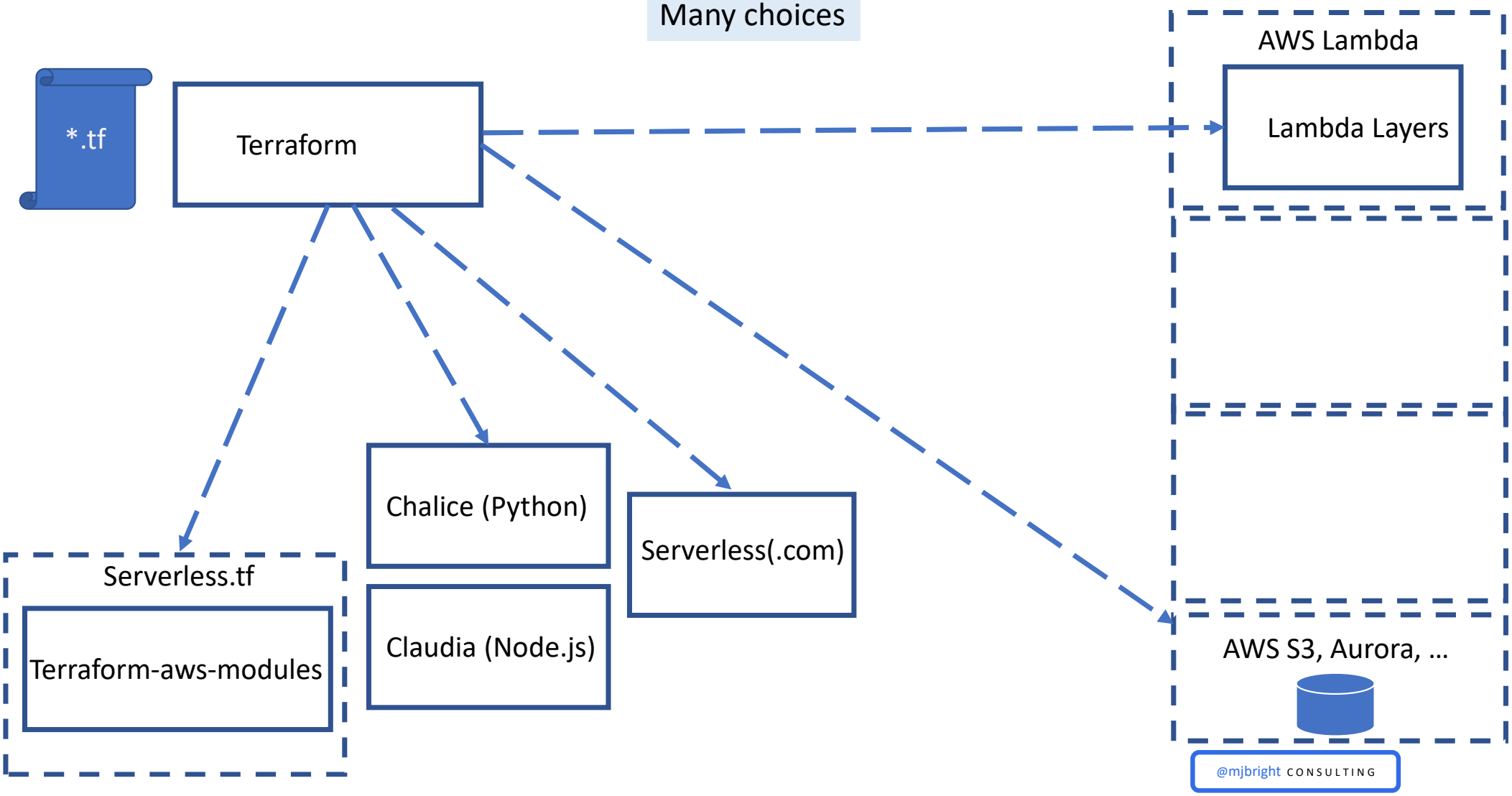
https://aws.amazon.com/dynamodb

Terraform + Serverless

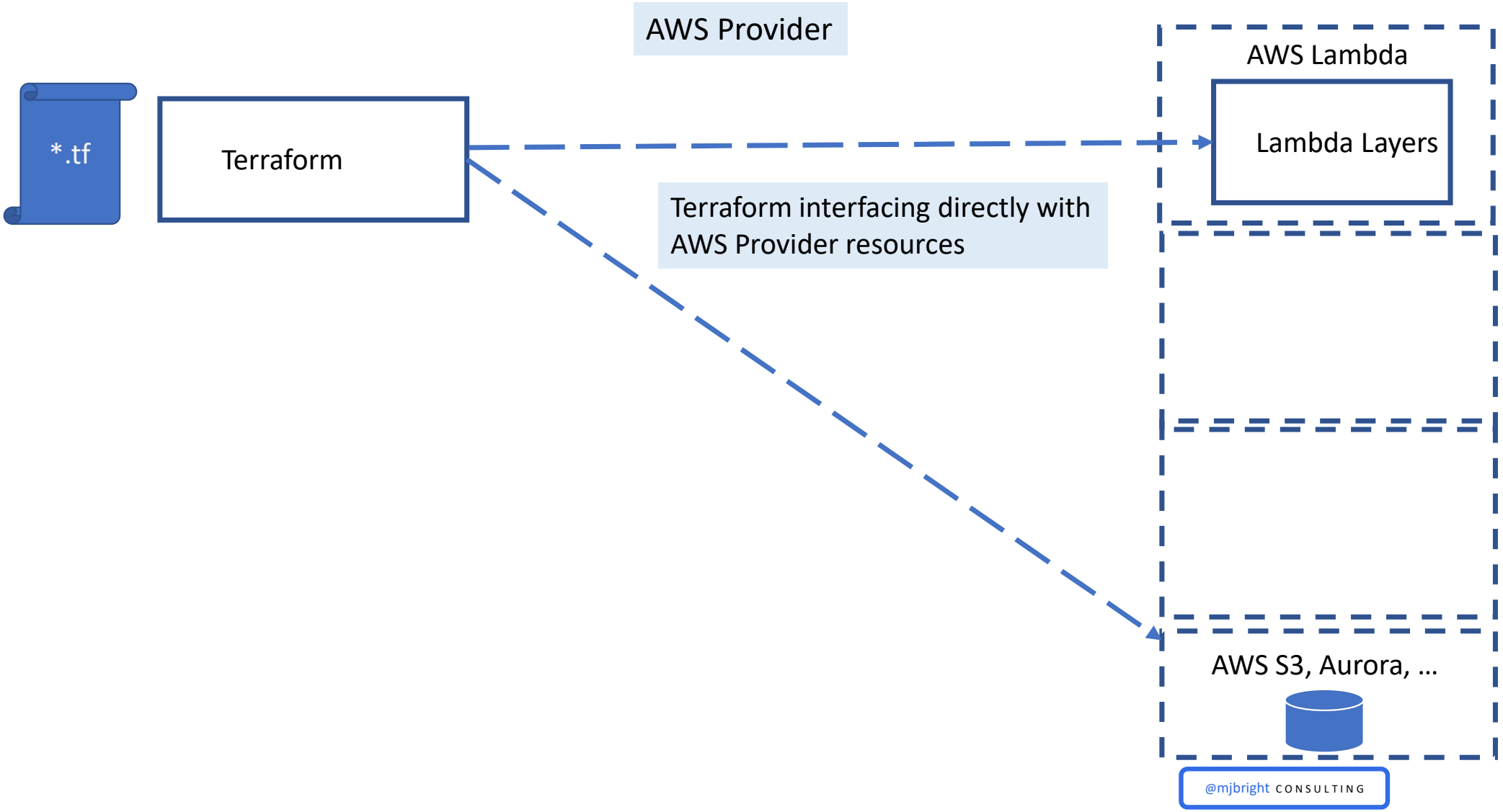
Many choices

Terraform + Serverless

Many choices

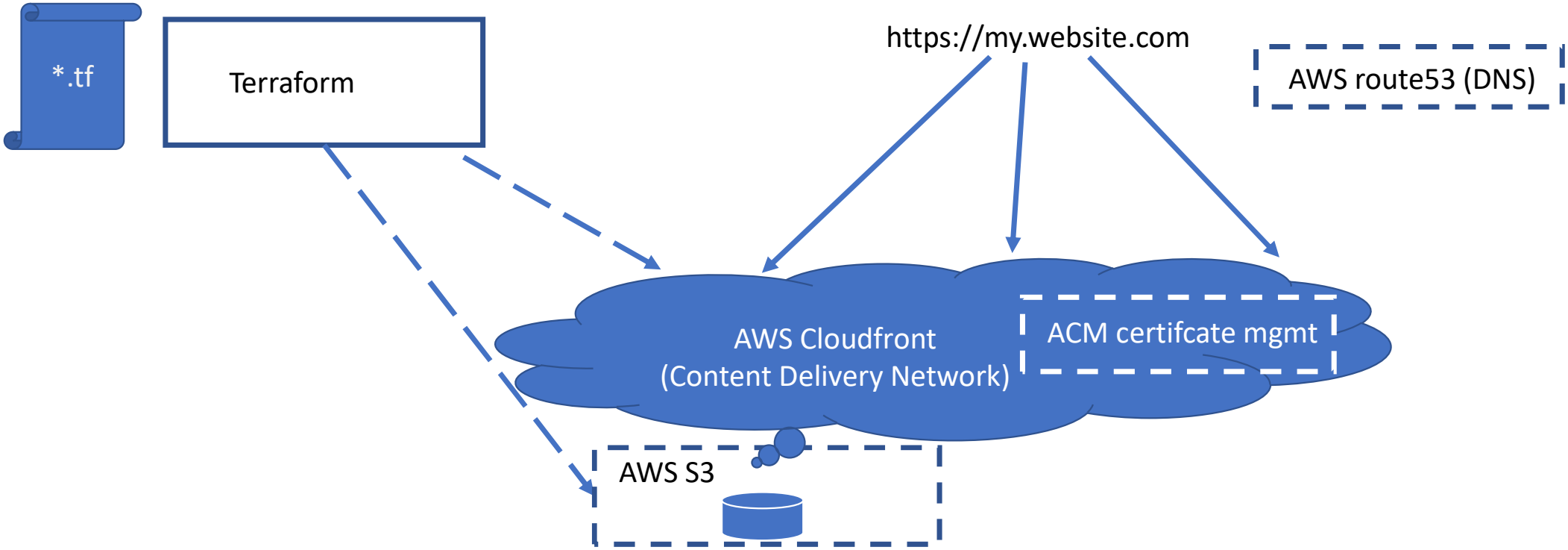


Terraform + Serverless



Demo scenarios

Scenario1 : s3 static website

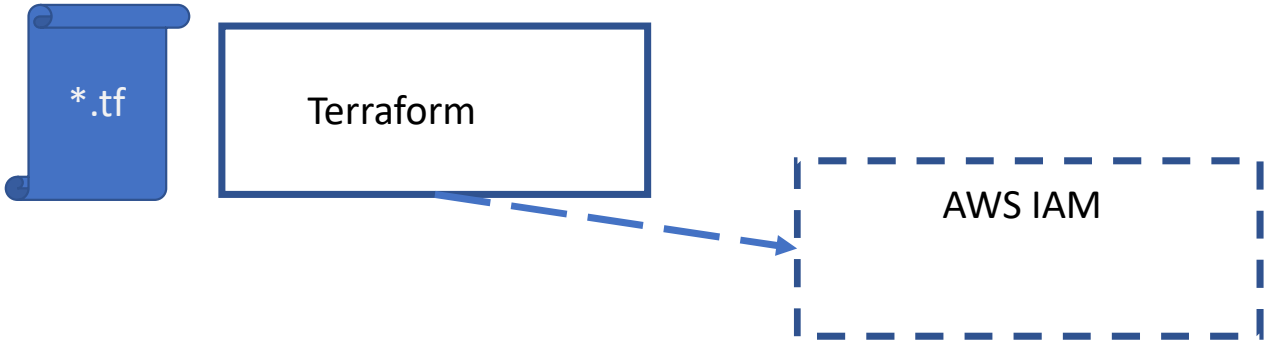


Note: this scenario is using S3 (serverless storage service) along with other backend services (CloudFront, Route53)
CloudFront provides SSL termination and content caching

A complete scenario could combine this approach whilst calling out to Lambda functions from the website pages

Demo scenarios

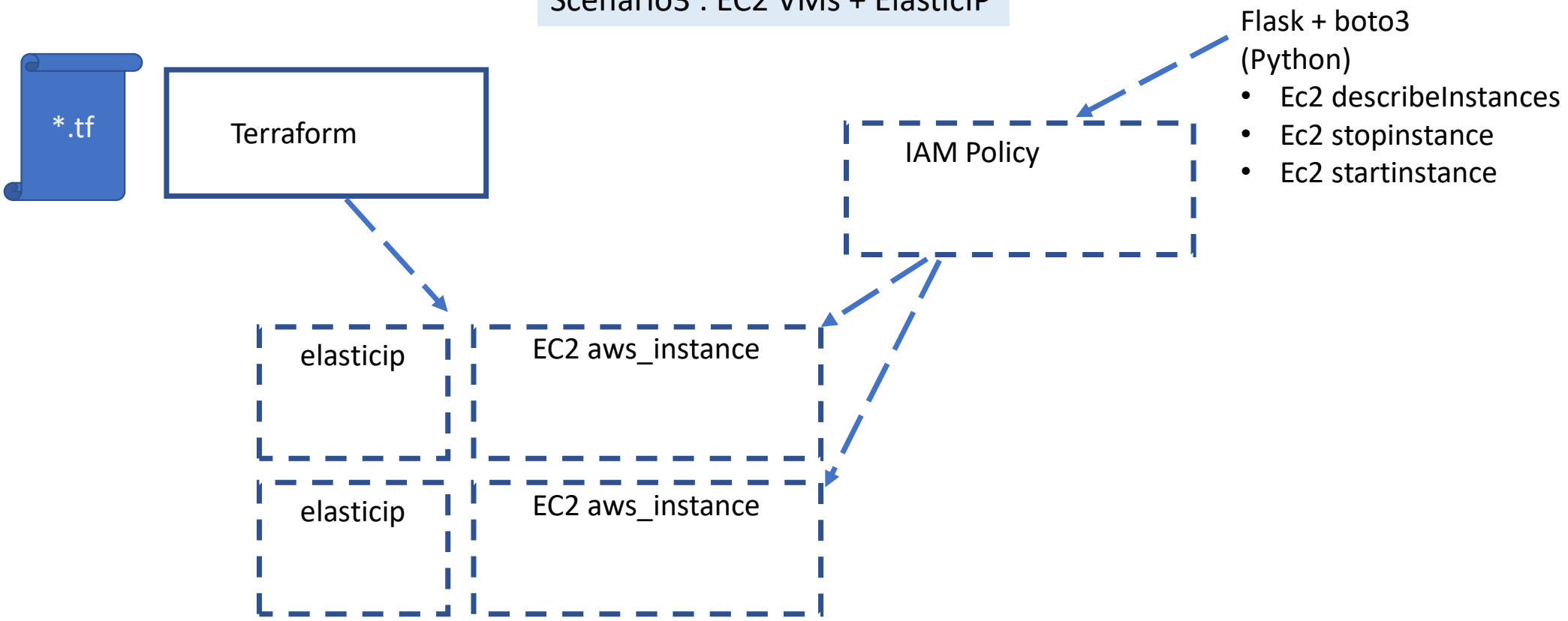
Scenario2 : IAM Policy Creation



Note: this scenario is creating an IAM profile – this can be used later in Scenario 3 or 4 (lambda)

Demo scenarios

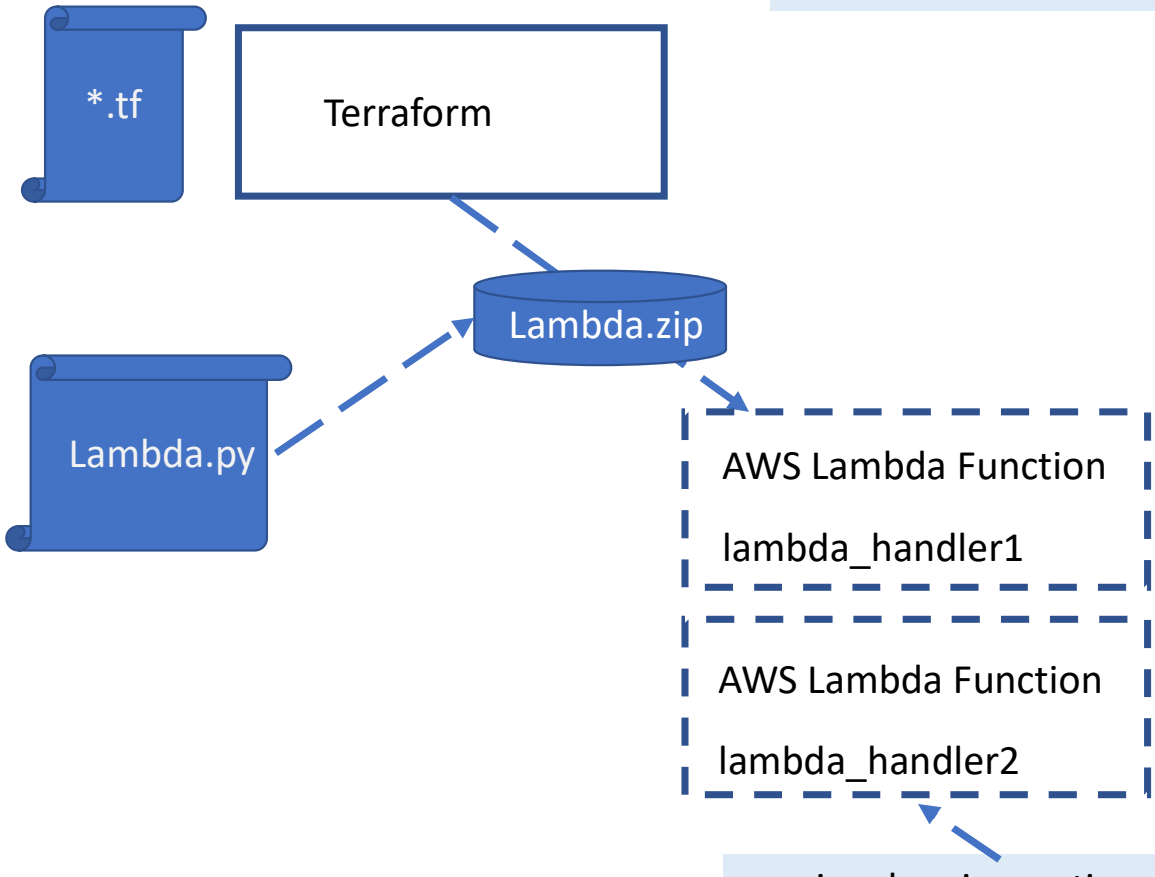
Scenario3 : EC2 VMs + ElasticIP



Note: this scenario is using the AWS SDK/api from an arbitrary machine – doesn't involve AWS Lambda

Demo scenarios

Scenario4a : AWS Lambda



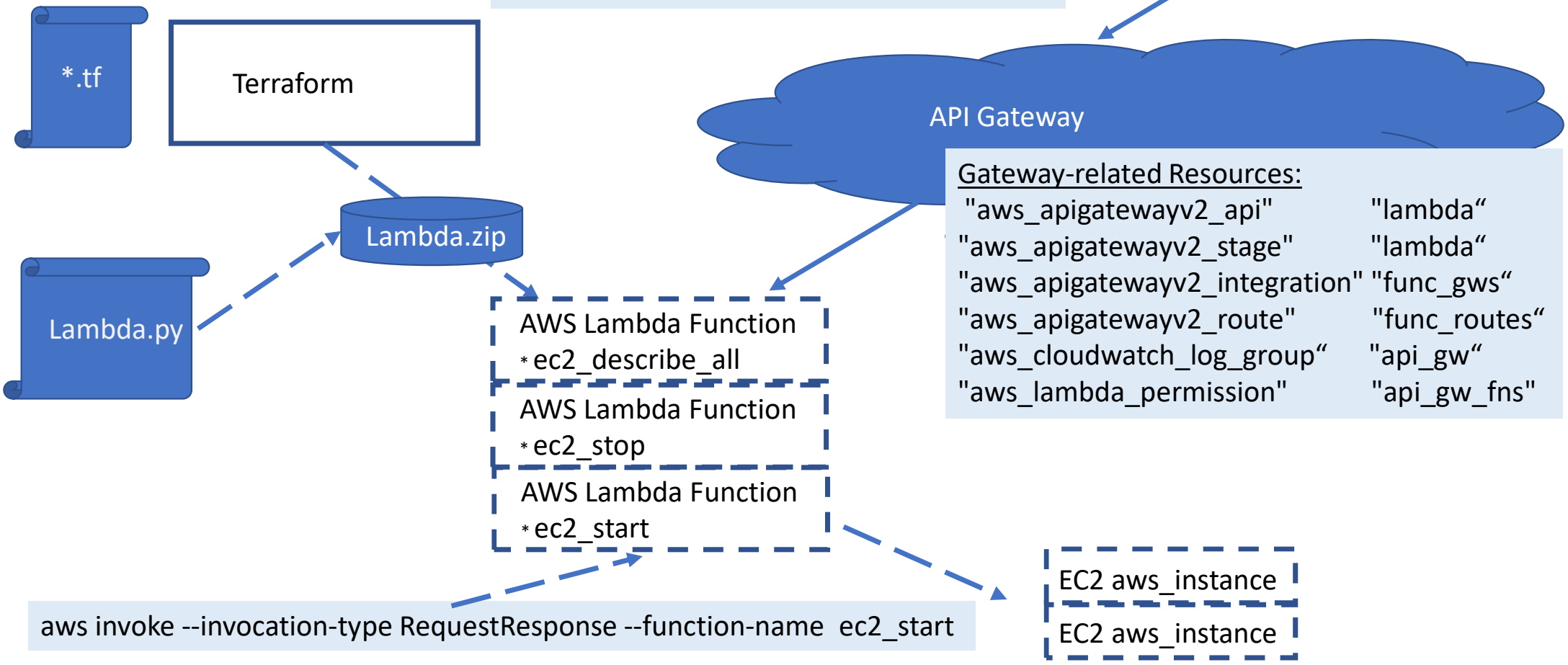
aws invoke --invocation-type RequestResponse --function-name lambda_handler2

Note: this is a basic AWS Lambda scenario to show packaging/deploying of lambda functions using Terraform and invocation of the function using the “aws cli” but without web access

Demo scenarios

https://<gway>/<stage>/<function>

Scenario4b : AWS Lambda – EC2 control



Note: this more realistic AWS Lambda scenario implements real functions, able to interface with EC2 via the AWS API (using the Python boto3 package). The functions can be invoked via the API Gateway

Terraform + Serverless

Summary: Scenarios

In these examples we have looked at

- Use of S3 as a serverless storage service, supplemented by the use of backend services
- Use of Terraform for packaging/deploying AWS Lambda functions
  - Basic function examples with invocation via AWS cli
  - Ability to call out to the AWS api allowing to interact with other AWS services (EC2)
  - Ability to invoke lambda functions via the API gateway

Terraform + Serverless

Summary: Conclusions

The AWS Lambda examples have used Terraform directly with the AWS Provider to demonstrate what is possible

Nevertheless it is clear that this does not provide abstractions to simplify the process – especially when used with API Gateway.

Whilst this is a viable approach – what should you do ?

Other solutions exist which it was not possible to cover in this short webinar

We will examine those options

Terraform + Serverless

Summary: Conclusions

If you are already developing with AWS Lambda but are new to Terraform then you would be best served by choosing an appropriate framework (Chalice or Claudia, SAM, “Serverless framework”) and sticking with that.

Note: If you wish, it is also possible to use Terraform with those frameworks

Terraform + Serverless

Summary: Conclusions

If you are a “Terraform shop” using Terraform for your deployments then it is more likely that combining Terraform with AWS Lambda is a good solution for you.

You might consider

- Using Chalice (Python) or Claudia (Node.js) with Terraform using the command
  - “chalice package --pkg-format terraform”
  - which generates json which can be used as a Terraform template
  - See https://aws.github.io/chalice/topics/tf.html#example

- Using the “Serverless framework” with Terraform

- If you prefer a pure Terraform approach then serverless.tf (based on the terraform-aws-modules) will provide better abstractions to use AWS Lambda and other Serverless services from Terraform

Thank you !

 https://linkedin.com/in/mjbright  @mjbright  @mjbright