Building a Serverless API

Craig Godden-Payne
Oct 26, 2018 · 7 min read ★

When it comes to creating an API, it is possible to be completely serverless, by utilising technologies such as API Gateway, and Lambda.

What is serverless?

> Serverless architectures are application designs that run in managed, ephemeral containers on a "Functions as a Service" (FaaS) platform.
>
> Such architectures remove much of the need for a traditional always-on server component and may benefit from significantly reduced operational cost, complexity, and engineering lead time.

. . .

How to build this?

In this article, we will explain how to do this, giving examples using infrastructure as code via terraform.

Let's start by creating the API gateway.

The API Gateway will contain all the endpoints for the service.

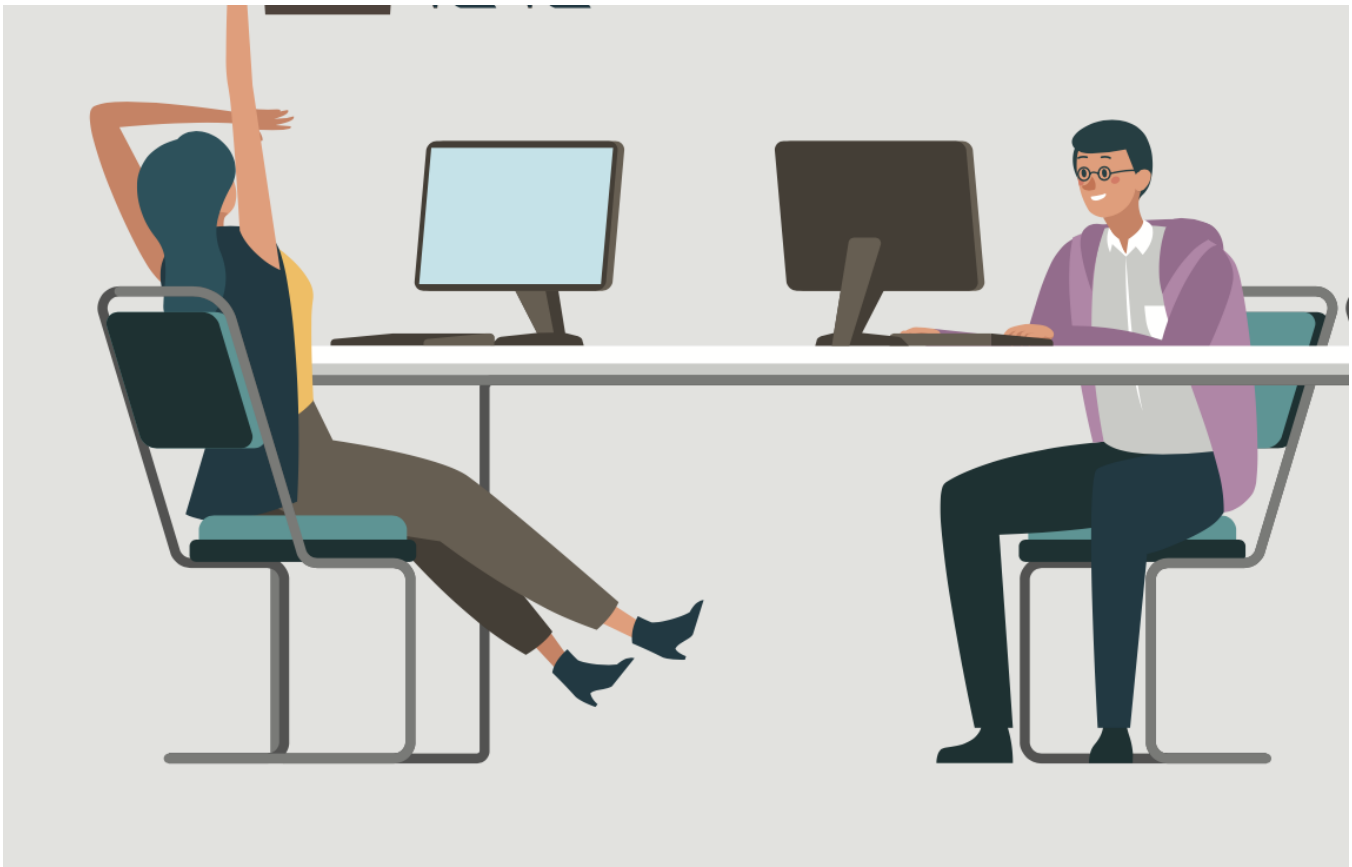
```
resource "aws_api_gateway_rest_api" "site_api" {
  name        = "site-api"
  description = "Website Api"
}
```

We are going to define the structure of what we think the API will start like. At the moment, we have decided it will have two endpoints, users and authentication, so it will look as follows:

```
site_api/users
site_api/authentication
```

The base path does not need to have any functionality, so I can create it as follows:

```
resource "aws_api_gateway_base_path_mapping" "site_api" {
  api_id      = "${aws_api_gateway_rest_api.site_api.id}"
  stage_name  = "${aws_api_gateway_deployment.site_api.stage_name}"
}
```



. . .

Within the terraform code, the stage name refers to a deployment, so I will need to create the deployment to be able to create the infra.

I know that I need to specify the dependencies in order to make sure they exist before this stage, so I specify them in the depends on section.

```
resource "aws_api_gateway_deployment" "site_api" {
  depends_on = [
    "aws_api_gateway_method.users_post",
    "aws_api_gateway_integration.users_post",
    "aws_api_gateway_method.authentication_post",
    "aws_api_gateway_integration.authentication_post",
  ]

  rest_api_id       = "${aws_api_gateway_rest_api.site_api.id}"
  stage_name        = "application"
  stage_description = "1.0"
  description       = "1.0"

  lifecycle {
    create_before_destroy = true
  }
}
```

Now is the time to specify the actual resource, for the value "users"

```
resource "aws_api_gateway_resource" "users" {
  rest_api_id = "${aws_api_gateway_rest_api.site_api.id}"
  parent_id   =
"${aws_api_gateway_rest_api.site_api.root_resource_id}"
  path_part   = "users"
}
```

I then need to specify the actual http method that will be called on the site_api.

```
resource "aws_api_gateway_method" "users_post" {
  rest_api_id   = "${aws_api_gateway_rest_api.site_api.id}"
  resource_id   = "${aws_api_gateway_resource.users.id}"
  http_method   = "POST"
  authorization = "NONE"
}
```

. . .

I then need to add an integration, to call the lambda from the resource.

```
resource "aws_api_gateway_integration" "users_post" {
  rest_api_id             =
"${aws_api_gateway_rest_api.site_api.id}"
  resource_id             = "${aws_api_gateway_resource.users.id}"
  http_method             =
"${aws_api_gateway_method.users_post.http_method}"
  type                    = "AWS_PROXY"
  uri                     = "arn:aws:apigateway:eu-west-
2:lambda:path/2015-03-
31/functions/${aws_lambda_function.users.arn}/invocations"
  integration_http_method = "POST"
}
```

I then need to make sure I add a method response, so the gateway knows is allowed to return.

```
resource "aws_api_gateway_method_response" "users_post_201" {
  rest_api_id = "${aws_api_gateway_rest_api.site_api.id}"
  resource_id = "${aws_api_gateway_resource.users.id}"
  http_method = "${aws_api_gateway_method.users_post.http_method}"
  status_code = "201"
}
```

I also want to specify for a bad request

```
resource "aws_api_gateway_method_response" "users_post_400" {
  rest_api_id = "${aws_api_gateway_rest_api.site_api.id}"
  resource_id = "${aws_api_gateway_resource.users.id}"
  http_method = "${aws_api_gateway_method.users_post.http_method}"
  status_code = "400"
}
```

I then need to do the same, for the authentication resource

```
resource "aws_api_gateway_resource" "authentication" {
  rest_api_id = "${aws_api_gateway_rest_api.site_api.id}"
  parent_id   =
"${aws_api_gateway_rest_api.site_api.root_resource_id}"
  path_part   = "authentication"
}

resource "aws_api_gateway_method" "authentication_post" {
  rest_api_id   = "${aws_api_gateway_rest_api.site_api.id}"
  resource_id   = "${aws_api_gateway_resource.authentication.id}"
  http_method   = "POST"
  authorization = "NONE"
}

resource "aws_api_gateway_integration" "authentication_post" {
  rest_api_id             =
"${aws_api_gateway_rest_api.site_api.id}"
  resource_id             =
"${aws_api_gateway_resource.authentication.id}"
  http_method             =
"${aws_api_gateway_method.authentication_post.http_method}"
  type                    = "AWS_PROXY"
  uri                     = "arn:aws:apigateway:eu-west-
2:lambda:path/2015-03-
31/functions/${aws_lambda_function.authentication.arn}/invocations"
  integration_http_method = "POST"
}

resource "aws_api_gateway_method_response" "authentication_post_201"
{
  rest_api_id = "${aws_api_gateway_rest_api.site_api.id}"
  resource_id = "${aws_api_gateway_resource.authentication.id}"
  http_method =
"${aws_api_gateway_method.authentication_post.http_method}"
  status_code = "201"
}

resource "aws_api_gateway_method_response" "authentication_post_400"
{
  rest_api_id = "${aws_api_gateway_rest_api.site_api.id}"
  resource_id = "${aws_api_gateway_resource.authentication.id}"
  http_method =
"${aws_api_gateway_method.authentication_post.http_method}"
```

```
  status_code = "400"
}
```

Now we have the endpoint integrations setup, we can start looking at the lambdas.

. . .

First we need to create a lookup to our deployed function in S3

```
data "aws_s3_bucket_object" "s3_build_artifact_bucket" {
  bucket = "CraigGoddenPayneBuildArtifacts"
  key    = "site-api/1.0/site-api.zip"
}
```

And make sure that we have a role, that can be used

```
resource "aws_iam_role" "lambda_role" {
  name = "site-api-role"

  assume_role_policy = <<EOF
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Action": "sts:AssumeRole",
      "Principal": {
        "Service": "lambda.amazonaws.com"
      },
      "Effect": "Allow",
      "Sid": ""
    }
  ]
}
EOF
}
```

We can then create a lambda, and point to the Function handler.

```
resource "aws_lambda_function" "users" {
  function_name     = "site-api-users"
  role              = "${aws_iam_role.lambda_role.arn}"
  description       = "Users"
  handler           = "SiteApi::SiteApi.Function::Users"
  runtime           = "dotnetcore2.0"
  timeout           = 30
  s3_bucket         =
"${data.aws_s3_bucket_object.s3_build_artifact_bucket.bucket}"
  s3_key            =
"${data.aws_s3_bucket_object.s3_build_artifact_bucket.key}"
  s3_object_version =
"${data.aws_s3_bucket_object.s3_build_artifact_bucket.version_id}"

  environment {
    variables = {
      Environment = "${terraform.workspace}"
    }
  }

  vpc_config = {
    subnet_ids = [
      "${data.aws_subnet_ids.private.ids[0]}",
      "${data.aws_subnet_ids.private.ids[1]}",
    ]

    security_group_ids = ["${aws_security_group.site_api.id}"]
  }

  tags {
```

```
    Owner       = "Craig Godden-Payne"
    Environment = "${terraform.workspace}"
  }
}
```

. . .

If you want to setup security group rules, you need to add the lambda into a subnet with internet access.

```
data "aws_vpc" "vpc" {
  filter {
    name   = "tag:Name"
    values = ["vpc.craigs-vpc"]
  }
}

data "aws_subnet" "private" {
  count = "${length(data.aws_subnet_ids.private.ids)}"
  id    = "${data.aws_subnet_ids.private.ids[count.index]}"
}

data "aws_subnet_ids" "private" {
  vpc_id = "${data.aws_vpc.vpc.id}"
```

```
  tags {
    Name = "private.*"
  }
}

resource "aws_security_group" "site_api" {
  name        = "site-api"
  description = "site api"
  vpc_id      = "${data.aws_vpc.vpc.id}"

  tags {
    Name = "site-api security group"
  }
}

resource "aws_security_group_rule" "private_egress_all" {
  type              = "egress"
  to_port           = 65535
  protocol          = "tcp"
  from_port         = 1024
  security_group_id = "${aws_security_group.site_api.id}"
  description       = "Private access to all"
  cidr_blocks       = ["0.0.0.0/0"]
}
```

You need a similar setup for the authentication lambda

```
resource "aws_lambda_permission" "authentication" {
  statement_id  = "AllowExecutionFromAPIGateway"
  action        = "lambda:InvokeFunction"
  function_name = "${aws_lambda_function.authentication.arn}"
  principal     = "apigateway.amazonaws.com"

  source_arn = "arn:aws:execute-api:eu-west-
2:000000000000:${aws_api_gateway_rest_api.site_api.id}/*/${aws_api_g
ateway_method.authentication_post.http_method}${aws_api_gateway_reso
urce.authentication.path}"
}

resource "aws_lambda_function" "authentication" {
  function_name     = "site-api-authentication"
  role              = "${aws_iam_role.lambda_role.arn}"
  description       = "Authentication"
  handler           = "SiteApi::SiteApi.Function::Authentication"
  runtime           = "dotnetcore2.0"
  timeout           = 30
  s3_bucket         =
"${data.aws_s3_bucket_object.s3_build_artifact_bucket.bucket}"
  s3_key            =
"${data.aws_s3_bucket_object.s3_build_artifact_bucket.key}"
  s3_object_version =
"${data.aws_s3_bucket_object.s3_build_artifact_bucket.version_id}"
```

```
  environment {
    variables = {
      Environment = "${terraform.workspace}"
    }
  }

  tags {
    Owner       = "Craig Godden-Payne"
    Environment = "${terraform.workspace}"
  }
}
```

If you want to see the full configuration, check out the below!

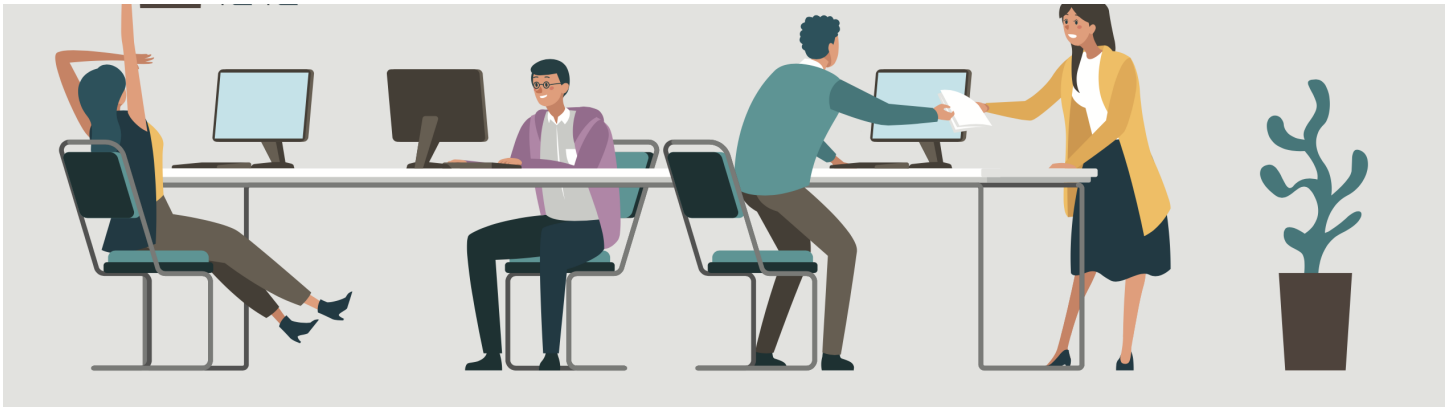
```
resource "aws_api_gateway_rest_api" "site_api" {
  name        = "site-api"
  description = "Website Api"
}

resource "aws_api_gateway_base_path_mapping" "site_api" {
  api_id      = "${aws_api_gateway_rest_api.site_api.id}"
  stage_name  = "${aws_api_gateway_deployment.site_api.stage_name}"
}

resource "aws_api_gateway_deployment" "site_api" {
  depends_on = [
    "aws_api_gateway_method.users_post",
    "aws_api_gateway_integration.users_post",
    "aws_api_gateway_method.authentication_post",
    "aws_api_gateway_integration.authentication_post",
  ]

  rest_api_id       = "${aws_api_gateway_rest_api.site_api.id}"
  stage_name        = "application"
  stage_description = "1.0"
  description       = "1.0"

  lifecycle {
    create_before_destroy = true
  }
}

resource "aws_api_gateway_resource" "users" {
  rest_api_id = "${aws_api_gateway_rest_api.site_api.id}"
  parent_id   =
"${aws_api_gateway_rest_api.site_api.root_resource_id}"
  path_part   = "users"
}

resource "aws_api_gateway_method" "users_post" {
  rest_api_id   = "${aws_api_gateway_rest_api.site_api.id}"
  resource_id   = "${aws_api_gateway_resource.users.id}"
  http_method   = "POST"
```

```
  authorization = "NONE"
}

resource "aws_api_gateway_integration" "users_post" {
  rest_api_id             =
"${aws_api_gateway_rest_api.site_api.id}"
  resource_id             = "${aws_api_gateway_resource.users.id}"
  http_method             =
"${aws_api_gateway_method.users_post.http_method}"
  type                    = "AWS_PROXY"
  uri                     = "arn:aws:apigateway:eu-west-
2:lambda:path/2015-03-
31/functions/${aws_lambda_function.users.arn}/invocations"
  integration_http_method = "POST"
}

resource "aws_api_gateway_method_response" "users_post_201" {
  rest_api_id = "${aws_api_gateway_rest_api.site_api.id}"
  resource_id = "${aws_api_gateway_resource.users.id}"
  http_method = "${aws_api_gateway_method.users_post.http_method}"
  status_code = "201"
}

resource "aws_api_gateway_method_response" "users_post_400" {
  rest_api_id = "${aws_api_gateway_rest_api.site_api.id}"
  resource_id = "${aws_api_gateway_resource.users.id}"
  http_method = "${aws_api_gateway_method.users_post.http_method}"
  status_code = "400"
}

resource "aws_api_gateway_resource" "authentication" {
  rest_api_id = "${aws_api_gateway_rest_api.site_api.id}"
  parent_id   =
"${aws_api_gateway_rest_api.site_api.root_resource_id}"
  path_part   = "authentication"
}

resource "aws_api_gateway_method" "authentication_post" {
  rest_api_id   = "${aws_api_gateway_rest_api.site_api.id}"
  resource_id   = "${aws_api_gateway_resource.authentication.id}"
  http_method   = "POST"
  authorization = "NONE"
}

resource "aws_api_gateway_integration" "authentication_post" {
  rest_api_id             =
"${aws_api_gateway_rest_api.site_api.id}"
  resource_id             =
"${aws_api_gateway_resource.authentication.id}"
  http_method             =
"${aws_api_gateway_method.authentication_post.http_method}"
  type                    = "AWS_PROXY"
  uri                     = "arn:aws:apigateway:eu-west-
2:lambda:path/2015-03-
31/functions/${aws_lambda_function.authentication.arn}/invocations"
  integration_http_method = "POST"
}
```

```
resource "aws_api_gateway_method_response" "authentication_post_201"
{
  rest_api_id = "${aws_api_gateway_rest_api.site_api.id}"
  resource_id = "${aws_api_gateway_resource.authentication.id}"
  http_method =
"${aws_api_gateway_method.authentication_post.http_method}"
  status_code = "201"
}

resource "aws_api_gateway_method_response" "authentication_post_400"
{
  rest_api_id = "${aws_api_gateway_rest_api.site_api.id}"
  resource_id = "${aws_api_gateway_resource.authentication.id}"
  http_method =
"${aws_api_gateway_method.authentication_post.http_method}"
  status_code = "400"
}

data "aws_s3_bucket_object" "s3_build_artifact_bucket" {
  bucket = "CraigGoddenPayneBuildArtifacts"
  key    = "site-api/1.0/site-api.zip"
}

resource "aws_iam_role" "lambda_role" {
  name = "site-api-role"

  assume_role_policy = <<EOF
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Action": "sts:AssumeRole",
      "Principal": {
        "Service": "lambda.amazonaws.com"
      },
      "Effect": "Allow",
      "Sid": ""
    }
  ]
}
EOF
}

resource "aws_lambda_function" "users" {
  function_name     = "site-api-users"
  role              = "${aws_iam_role.lambda_role.arn}"
  description       = "Users"
  handler           = "SiteApi::SiteApi.Function::Users"
  runtime           = "dotnetcore2.0"
  timeout           = 30
  s3_bucket         =
"${data.aws_s3_bucket_object.s3_build_artifact_bucket.bucket}"
  s3_key            =
"${data.aws_s3_bucket_object.s3_build_artifact_bucket.key}"
  s3_object_version =
"${data.aws_s3_bucket_object.s3_build_artifact_bucket.version_id}"
```

```
  environment {
    variables = {
      Environment = "${terraform.workspace}"
    }
  }

  vpc_config = {
    subnet_ids = [
      "${data.aws_subnet_ids.private.ids[0]}",
      "${data.aws_subnet_ids.private.ids[1]}",
    ]

    security_group_ids = ["${aws_security_group.site_api.id}"]
  }

  tags {
    Owner       = "Craig Godden-Payne"
    Environment = "${terraform.workspace}"
  }
}

data "aws_vpc" "vpc" {
  filter {
    name   = "tag:Name"
    values = ["vpc.craigs-vpc"]
  }
}

data "aws_subnet" "private" {
  count = "${length(data.aws_subnet_ids.private.ids)}"
  id    = "${data.aws_subnet_ids.private.ids[count.index]}"
}

data "aws_subnet_ids" "private" {
  vpc_id = "${data.aws_vpc.vpc.id}"

  tags {
    Name = "private.*"
  }
}

resource "aws_security_group" "site_api" {
  name        = "site-api"
  description = "site api"
  vpc_id      = "${data.aws_vpc.vpc.id}"

  tags {
    Name = "site-api security group"
  }
}

resource "aws_security_group_rule" "private_egress_all" {
  type              = "egress"
  to_port           = 65535
  protocol          = "tcp"
  from_port         = 1024
```

```
  security_group_id = "${aws_security_group.site_api.id}"
  description       = "Private access to all"
  cidr_blocks       = ["0.0.0.0/0"]
}

resource "aws_lambda_permission" "authentication" {
  statement_id  = "AllowExecutionFromAPIGateway"
  action        = "lambda:InvokeFunction"
  function_name = "${aws_lambda_function.authentication.arn}"
  principal     = "apigateway.amazonaws.com"

  source_arn = "arn:aws:execute-api:eu-west-
2:000000000000:${aws_api_gateway_rest_api.site_api.id}/*/${aws_api_g
ateway_method.authentication_post.http_method}${aws_api_gateway_reso
urce.authentication.path}"
}

resource "aws_lambda_function" "authentication" {
  function_name     = "site-api-authentication"
  role              = "${aws_iam_role.lambda_role.arn}"
  description       = "Authentication"
  handler           = "SiteApi::SiteApi.Function::Authentication"
  runtime           = "dotnetcore2.0"
  timeout           = 30
  s3_bucket         =
"${data.aws_s3_bucket_object.s3_build_artifact_bucket.bucket}"
  s3_key            =
"${data.aws_s3_bucket_object.s3_build_artifact_bucket.key}"
  s3_object_version =
"${data.aws_s3_bucket_object.s3_build_artifact_bucket.version_id}"

  environment {
    variables = {
      Environment = "${terraform.workspace}"
    }
  }

  tags {
    Owner       = "Craig Godden-Payne"
    Environment = "${terraform.workspace}"
  }
}
```



Graphic Attributions:
https://www.freepik.com/free-photos-vectors/people
pikisuperstar

AWS Lambda Api Gateway