Tooling in the age of serverless computing

Donna Malayeri
Product and Community Manager, Pulumi
@lindydonna
In 2008, AWS had 4 services

- **Amazon Elastic Compute Cloud (Amazon EC2)**
  Amazon Elastic Compute Cloud delivers scalable, pay-as-you-go compute capacity in the cloud.

- **Amazon SimpleDB**
  Amazon SimpleDB works in conjunction with Amazon S3 and Amazon EC2 to run queries on structured data in real time.

- **Amazon Simple Storage Service (Amazon S3)**
  Amazon Simple Storage Service provides a fully redundant data storage infrastructure for storing and retrieving any amount of data, at any time, from anywhere on the Web.

- **Amazon Simple Queue Service (Amazon SQS)**
  Amazon Simple Queue Service provides a hosted queue for storing messages as they travel between computers, making it easy to build automated workflow between Web services.
Today, each vendor has over a hundred services
#serverless? Or #servicefull?

Patrick Debois
@patrickdebois

I think #serverless is just a manifestation of a bigger trend that is #servicefull

6:57 AM - May 24, 2016

14 likes 16 people are talking about this
Now, we have a new problem

- Relying on cloud vendors for managed services == net good
- Having to manage hundreds of resources...
Option A: use vendor’s technology

AWS CLOUDFORMATION

AZURE RESOURCE MANAGER

GOOGLE CLOUD DEPLOYMENT MANAGER
Option B: use a deployment tool
When should you use a tool?

• Your team does not have expertise in the vendor technology
  • Or, some team members don’t have this expertise
• You’re comfortable taking a new dependency, if it buys productivity
• You have a small team and can’t afford to specialize
A framework for evaluating tools

- How do I describe infrastructure?
- How do I write my app?
- How do I deploy?
- How do I learn?
Serverless deployment tools

- serverless
- Chalice
- Claudia.js
- SPARTA
- Zappa
- Terraform
- APEX
- pulumi
Option B: use a deployment tool
Option B: use a deployment tool

More at github.com/lindydonna/velocity-examples
Count the number of times a route has been hit

$\texttt{curl}\ \$\text{URL}/foo
\{"id":"foo","count":1\}$

$\texttt{curl}\ \$\text{URL}/foo
\{"id":"foo","count":2\}$

$\texttt{curl}\ \$\text{URL}/bar
\{"id":"bar","count":1\}$
Serverless Framework

- Serverless is your toolkit for deploying and operating serverless architectures. Focus on your application, not your infrastructure.

- First class experience for serverless functions on AWS, Azure, GCP, and more
- Manages code and infrastructure
- Can author functions in JavaScript, Python, Java, and more
- Rich ecosystem of community plugins
Serverless Framework example

Serverless Framework example

```
const AWS = require('aws-sdk');

const dynamoDb = new AWS.DynamoDB.DocumentClient();

module.exports.handler = (event, context, callback) => {
  ...
}
```

45 LOC for config

30 LOC for Lambda
Deploying with Serverless Framework

```javascript
sls deploy - Uses vendor’s deployment service
  • Such as CloudFormation on AWS

After deployment
  • sls logs - View logs for a specific function
  • CloudFormation console
```
Serverless Framework

- **How do I describe infrastructure?**
  - Lambda and APIG are first-class
  - For everything else, use CloudFormation

- **How do I write my app?**
  - No change to how Lambda is authored

- **How do I deploy?**
  - `sls deploy`, uses CloudFormation

- **How do I learn?**
  - An active community
  - Lots of plugins and tutorials
Terraform

- HashiCorp Terraform enables you to safely and predictably create, change, and improve infrastructure. Define infrastructure as code to increase operator productivity and transparency.

- Defines a custom language for describing deployments to AWS, Azure, etc
- No special support for serverless scenarios
resource "aws_iam_role" "lambda_exec" {
  assume_role_policy = <<EOF
    {
      "Version": "2012-10-17",
      "Statement": [
        {
          "Action": "sts:AssumeRole",
          "Principal": {
            "Service": "lambda.amazonaws.com",
          },
          "Effect": "Allow",
          "Sid": ""
        }
      ]
    }
EOF
}

resource "aws_api_gateway_rest_api" "example" {
}

resource "aws_api_gateway_resource" "proxy" {
  rest_api_id = "${aws_api_gateway_rest_api.example.id}
  parent_id = "${aws_api_gateway_rest_api.example.root_resource_id}
  path_part = "proxy+"
}

resource "aws_api_gateway_method" "proxy" {
  rest_api_id = "${aws_api_gateway_rest_api.example.id}
  resource_id = "${aws_api_gateway_resource.proxy.id}
  http_method = "ANY"
  authorization = "NONE"
}

resource "aws_api_gateway_integration" "lambda" {
  rest_api_id = "${aws_api_gateway_rest_api.example.id}
  resource_id = "${aws_api_gateway_method.proxy.resource_id}
  http_method = "${aws_api_gateway_method.proxy.http_method}
  integration_http_method = "POST"
  type = "AWS_PROXY"
  uri = "${aws_lambda_function.example.invoke_arn}
}

resource "aws_lambda_function" "example" {
  filename = "${data.archive_file.lambda.output_path}
  function_name = "ServerlessExample"
  handler = "app.handler"
  runtime = "nodejs8.10"
  role = "${aws_iam_role.lambda_exec.arn}
  source_code_hash = "${base64sha256(file("${data.archive_file.lambda.output_path}"))}
  publish = true
  environment { variables = {
              DYNAMODB_TABLE = "${aws_dynamodb_table.dynamo-table.id}"
            }}
}

data "archive_file" "lambda" {
  type = "zip"
  source_file = "app.js"
  output_path = "lambda.zip"
}

150 LOC for config
30 LOC for Lambda
Deploying with Terraform

terraform apply
  • Uses vendor APIs directly, does not use services like CloudFormation
    • Has a checkpoint file to store state of last deployment
    • In free version, you manage this yourself

After deployment
  • No specific tooling for viewing resources or getting logs
**Terraform**

How do I describe infrastructure?
- Use HCL to configure any cloud resource on any cloud
- Can author your own provider

How do I write my app?
- No change to how Lambda is authored

How do I deploy?
- Uses vendor APIs directly
- You have to manage a state file

How do I learn?
- Lots of tutorials and examples for infrastructure
- Less common for serverless
Claudia.js

- Claudia makes it easy to deploy Node.js projects to AWS Lambda and API Gateway. It automates all the error-prone deployment and configuration tasks, and sets everything up the way JavaScript developers expect out of the box.

- Supports JavaScript on AWS only
- Automatically sets up CORS, request/response format, etc
- With API builder, use API Gateway with regular JavaScript web server syntax
- Manages only API Gateway and Lambda resources
Create Dynamo table manually:

```bash
clara create	--region	us-east-1	--api-module	index	--policies	policy	--set-env-from-json	env.json
```

Claudia.js example:

```javascript
const ApiBuilder = require('claudia-api-builder');
var api = new ApiBuilder();

api.get('/{proxy+}', function (request) {
...
});
```

Policy JSON:

```json
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Action": [
```

AWS CLI:

```bash
aws dynamodb create-table --table-name RouteCounts ...
```

Create Dynamo table manually:

```bash
clara create	--region	us-east-1	--api-module	index	--policies	policy	--set-env-from-json	env.json
```
Deploying with Claudia.js

• Deploy new app with claudia create
  • Uses AWS APIs directly, creates a JSON file

• Deploy code with claudia update

After deployment

• No specific tooling for viewing resources or getting logs
Claudia.js

How do I describe infrastructure?
• JSON for APIG and Lambda
• Anything else, use an additional tool

How do I write my app?
• Convenient syntax for defining APIs

How do I deploy?
• Uses AWS APIs directly
• You have to manage a json state file

How do I learn?
• Lots of tutorials and examples
• Upcoming book later this year
Create containers, serverless functions, and cloud infrastructure, with all the benefits of immutable infrastructure, and real programming languages.

• For cloud configuration, use JavaScript, Python or Go
• For functions and containers, use any supported language
• Manage any resource in AWS, Azure, Kubernetes, or GCP
• Use low-level resources or higher-level framework
Pulumi example: low level

```javascript
const aws = require('@pulumi/aws');

const policy = {
  ...,
};

let deployment = new aws.apigateway.Deployment('api-deployment', {
  restApi: restApi,
  stageName: '',
});

let stage = new aws.apigateway.Stage('api-stage', {
  restApi: restApi,
  deployment: deployment,
  stageName: stageName
});

let invokePermission = new aws.lambda.Permission('api-lambda-permission', {
  action: 'lambda:invokeFunction',
  function: lambda,
  principal: 'apigateway.amazonaws.com',
  sourceArn: deployment.executionArn
});
```
Pulumi example: using `@pulumi/aws-serverless`

```javascript
const aws = require("@pulumi/aws");
const policy = { ... };
const role = new aws.iam.Role(...);
const fullAccess = new aws.iam.RolePolicyAttachment(...);

let lambda = new aws.lambda.Function("myfunction", {
  code: new pulumi.asset.FileArchive("./app"),
  role: role.arn,
  handler: "app.handler",
  runtime: aws.lambda.NodeJS8d10Runtime,
  environment: {
    variables: {
      "DYNAMODB_TABLE": table.name
    }
  }
});

const api = new serverless.apigateway.API("myapi", {
  routes: [
    { method: "GET", path: "/{proxy+}" , handler: lambda },
  ],
});
```
Pulumi example: define Lambda inline

```javascript
// The handler for GET /{route+}.
const getHandler = async (event) => {
    const awssdk = require("aws-sdk");
    const dynamo = new awssdk.DynamoDB.DocumentClient();

    const tableName = table.name.get();
    ...
}

const api = new serverless.apigateway.API("api", {
    routes: [
        { method: "GET", path: "/{route+}", handler: getHandler }
    ]
});
```

Reference table object directly

52 LOC for config AND Lambda
Deploying with Pulumi

pulumi update
• Uses vendor APIs directly
• Deployment state is stored on pulumi.com

After deployment
pulumi logs

```bash
$ pulumi logs -f
2018-05-25T12:58:34.960-07:00[ ] Starting ffmpeg task...
```
Pulumi

How do I describe infrastructure?
- Use JavaScript, Python or Go
- First class support for containers and serverless
- Target AWS, Azure, GCP, k8s

How do I write my app?
- Can author as regular Lambda
- Or inline with infra code

How do I deploy?
- Uses vendor APIs directly
- State file is saved on pulumi.com

How do I learn?
- Still in beta, but lots of examples and tutorials
Summary

• For a more productive cloud experience, consider a deployment tool
• Answer these questions
  • How do I describe infrastructure?
  • How do I write my app?
  • How do I deploy?
  • How do I learn?

• All examples available at github.com/lindydonna/velocity-examples
• Try out Pulumi at www.pulumi.com