Devops
The cloud
Documentation

Admin

- HW4 due today
- HW5 out today: group homework, a bit more involved than prior HWs

Things I worried about in school

Operational semantics*

Implementing virtual memory

Proving things about regular languages

Median of medians algorithm (very cool)

Building parsers

Things I worried about in school

Operational semantics*

Implementing virtual memory

Proving things about regular languages

Median of medians algorithm (very cool)

Building parsers

Things I worried about in school

Things I worried about shipping our first product

Operational semantics*

Implementing virtual memory

Proving things about regular languages

Median of medians algorithm (very cool)

Building parsers

Things I worried about in school

Why my AWS CDK was generating malformed CloudFormation

Resource limits

How to set up deployment pipeline

Locking down outbound traffic from VPC

Permissions in the AWS accounts

Things I worried about shipping our first product

You have started a company!
You are an online welding supply store

You have started a company!
You are an online welding supply store

It is the year 2000
What do you do

You have started a company!
You are an online welding supply store

- You put a server in your garage
- You use the LAMP stack:
  - Linux
  - Apache web server (serves directories and files)
  - MySQL database (holds your welding supply info)
  - All the code is Perl (e.g., for updating inventory)

What is nice about this stack?

- You put a server in your garage
- You use the LAMP stack:
  - Linux
  - Apache web server (serves directories and files)
  - MySQL database (holds your welding supply info)
  - All the code is Perl (e.g., for updating inventory)

What is nice about this stack?

- Run everything yourself
- Relatively simple — you probably understand what's going on
- …?

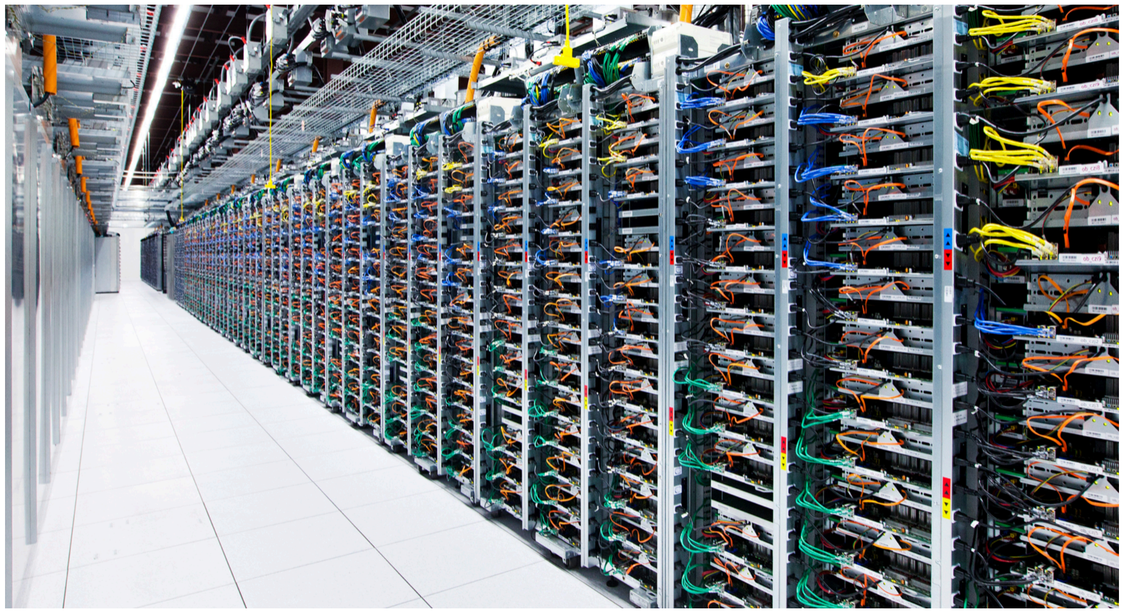
What is tough about this stack?

- You put a server in your garage
- You use the LAMP stack:
  - Linux
  - Apache web server (serves directories and files)
  - MySQL database (holds your welding supply info)
  - All the code is Perl (e.g., for updating inventory)

What is tough about this stack?

- Run everything yourself
- Scaling —what if you need more e.g., storage space?
- Where is configuration information stored? What happens if you need to re-start the machine? Re-deploy the server?

Time passes, your welding supply store becomes enormously successful



It’s always awesome to have thousands of machines:

It’s always awesome to have thousands of machines:

- Scaling is easy: buy more machines
- You can control redundancy
- You can control residency (i.e., what runs where)

It’s always awesome to have thousands of machines. But what are the cons of this situation?

It’s always awesome to have thousands of machines. But what are the cons of this situation?

- Massive cost (machines, cooling infrastructure, networking infrastructure)
- Likely low average utilization. You bought these machines to handle peak traffic
- You have to distribute applications across the machines
- You spend a lot of time physically maintaining infrastructure, installing infrastructure, configuring the network, etc.

Maybe you branch out beyond welding

responses to two problems. First: By the early 2000s, Amazon—still known mainly as an online bookseller—had built from scratch one of the world's biggest websites, but adding new features had become frustratingly slow. Software engineering teams were spending 70% of their time building the basic elements any project would require—most important, a storage system and an appropriate computing infrastructure. Building those elements for projects at Amazon scale was hard, and all that work merely produced a foundation on which to build the cool new customer-pleasing features Amazon was seeking. Every project team was performing the same drudgery. Bezos and other Amazon managers started calling it "undifferentiated heavy lifting" and complaining that it produced "muck."

https://fortune.com/longform/amazon-web-services-ceo-adam-selipsky-cloud-computing/

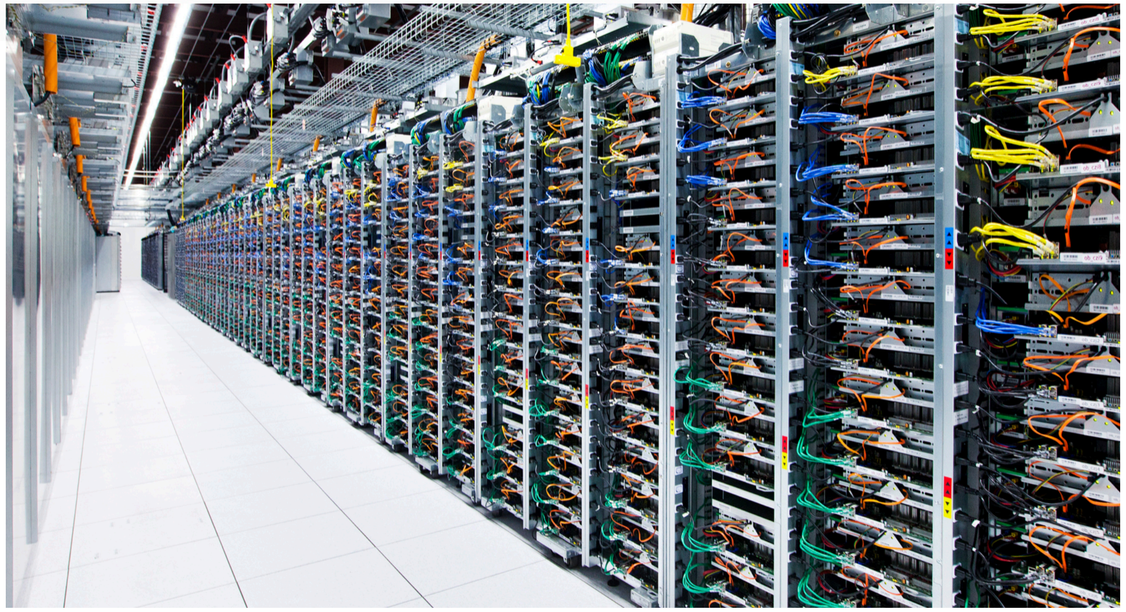
Maybe you branch out beyond welding

responses to two problems. First: By the early 2000s, Amazon—still known mainly as an online bookseller—had built from scratch one of the world's biggest websites, but adding new features had become frustratingly slow. Software engineering teams were spending 70% of their time building the basic elements any project would require—most important, a storage system and an appropriate computing infrastructure. Building those elements for projects at Amazon scale was hard, and all that work merely produced a foundation on which to build the cool new customer-pleasing features Amazon was seeking. Every project team was performing the same drudgery. Bezos and other Amazon managers started calling it "undifferentiated heavy lifting" and complaining that it produced "muck."

For every new app you keep building core infrastructure from scratch

It took three years before AWS went live. In 2005 Jassy hired Selipsky from a software firm to run marketing, sales, and support, Selipsky recalls: “Amazon called and told me there was this initiative for something about turning the guts of Amazon inside out, but other companies could use it.” AWS’s first service, for data storage, “was such a novel concept that it was even hard to explain and hard for me to understand,” he says.

Great way to use those machines, too



“Cloud computing” is born

Five months later AWS launched its other foundational service, EC2, for Elastic Compute Cloud, which was also instantly popular. The revolution had begun. Instead of raising millions of dollars to buy servers and build data centers, startups could now get online with a credit card, and pay a monthly bill for just the computing power and storage they used. If their new app was a hit, they could immediately engage all the cloud services that they needed. If it bombed, they weren’t stuck with rooms of junk equipment. As a Silicon Valley entrepreneur and early AWS customer told *Wired* in 2008: “Infrastructure is the big guys’ most powerful asset. This levels the field.”

Disadvantages of cloud computing?

Disadvantages of cloud computing?

- What if cloud service goes down??????!
- What if other customers/apps affect your app?
- …less control over the machines/residency/the network/etc.

Disadvantages of cloud computing?

- What if cloud service goes down??????!
- What if other customers/apps affect your app*?
- …less control over the machines/residency/the network/etc.

- Hot take: probably not that big of a problem
- * Spectre?!?!

Advantages of cloud computing?

Advantages of cloud computing?

- Easy to provision a new instance!
- Never think about hardware:
  - Automatic backup and replication
  - Can set up complex virtual machines and virtual networks

Advantages of cloud computing?

- Easy to provision a new instance!
- Never think about hardware:
  - Automatic backup and replication
  - Can set up complex **virtual machines** and virtual networks

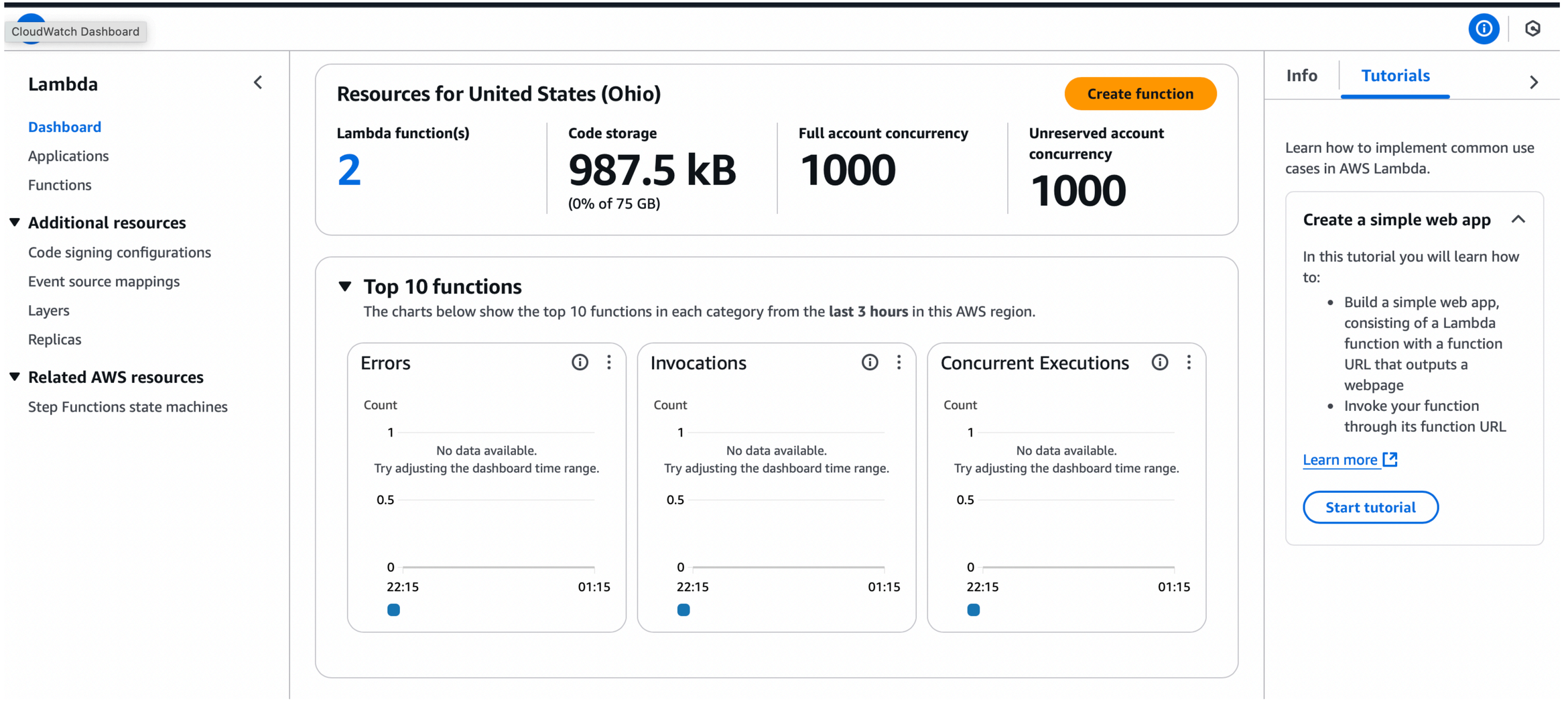
Advantages of cloud computing?

- Easy to provision a new instance!
- Never think about hardware:
  - Automatic backup and replication
  - Can set up complex **virtual machines** and virtual networks
- Write infrastructural configurations as code

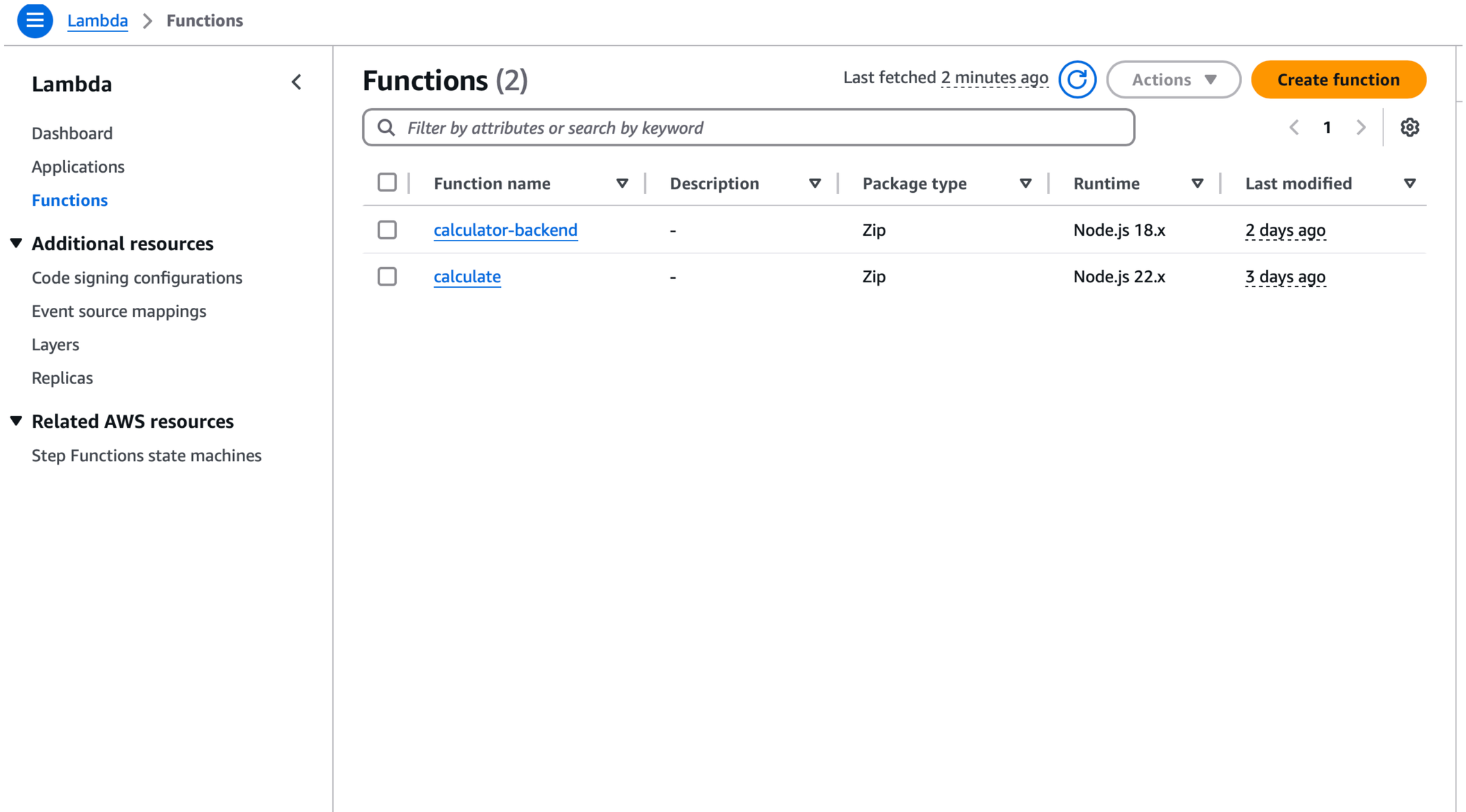
Advantages of cloud computing?

- Easy to provision a new instance!
- Never think about hardware:
  - Automatic backup and replication
  - Can set up complex virtual machines and virtual networks
- **Write infrastructural configurations as code <- huge in practice**

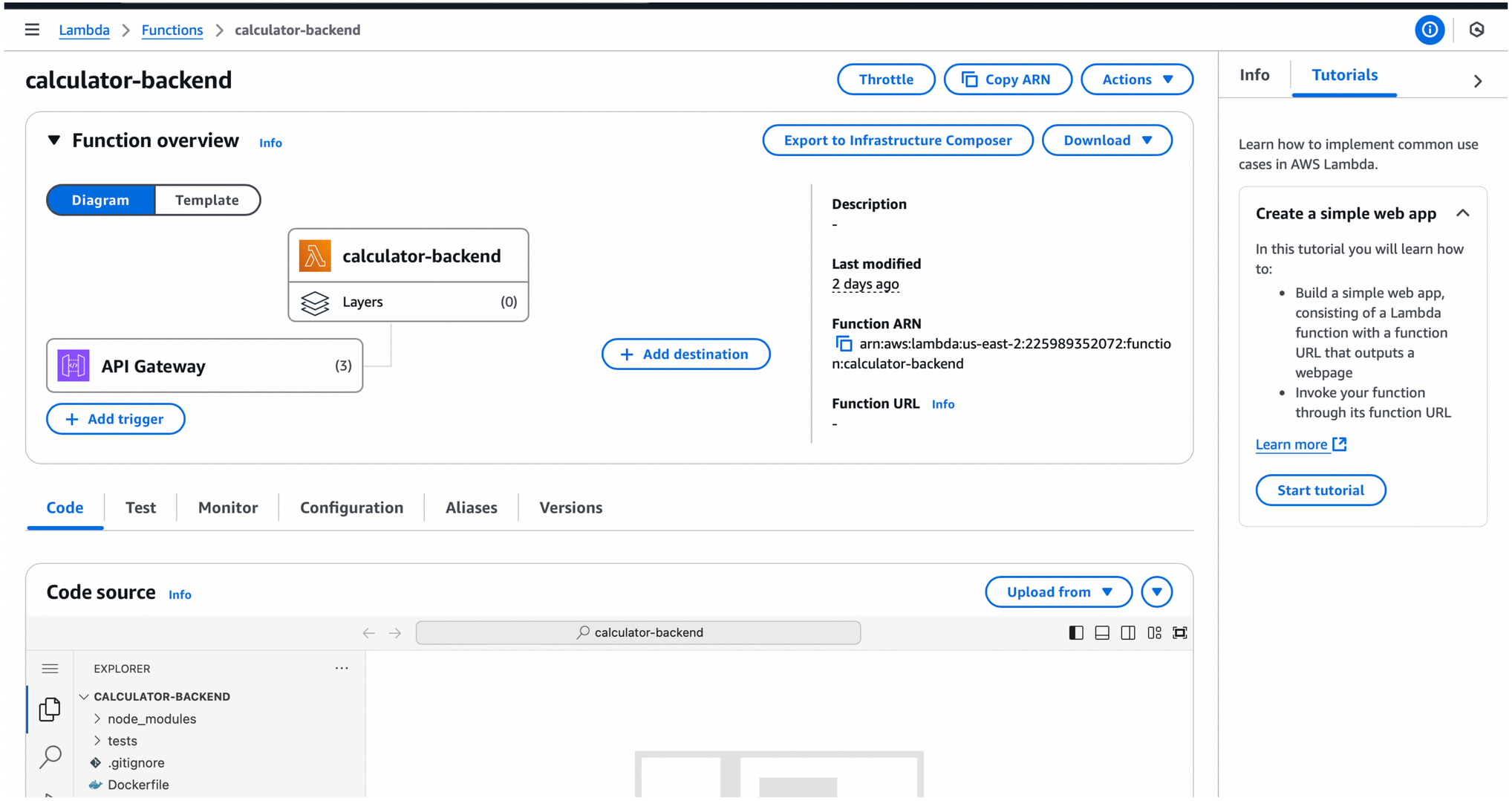
Using cloud infrastructure: Lambda



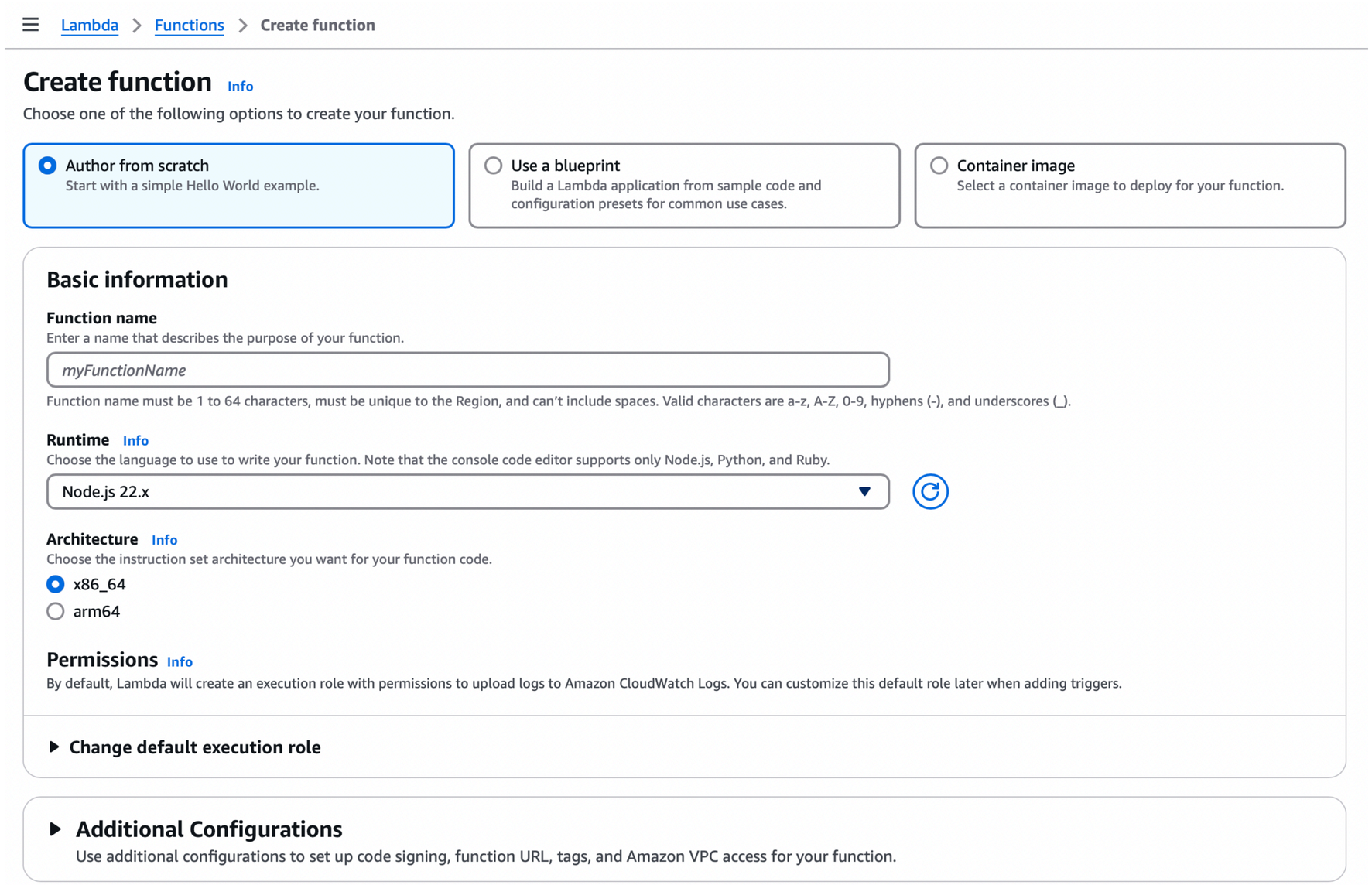
Lambda functions



Calculator lambda function (you'll do this!)



Create a new function



There must be a better way

There must be a better way

Please save us from the clicky

Infrastructure as code to the rescue!

Infrastructure as code to the rescue!

Deeply, deeply frustrating, but so much better than the alternative

Infrastructure as code (IaC)

Automate infrastructure on any cloud with Terraform

Kubernetes, also known as K8s, is an open source system for automating deployment, scaling, and management of containerized applications.

Infrastructure as code (IaC)

Automate infrastructure on any cloud with Terraform

Kubernetes, also known as K8s, is an open source system for automating deployment, scaling, and management of containerized applications.

AWS CloudFormation

Speed up cloud provisioning with infrastructure as code

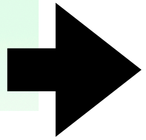
Azure Resource Manager

Simplify how you manage your app resources

Infrastructure as code (IaC)

AWS Cloud Development Kit

Define your cloud application resources using familiar programming languages



AWS CloudFormation

Speed up cloud provisioning with infrastructure as code

Example: Lambda config as code

```
44  /**
45   * Represents a lambda function and alias pairing, where the function is replaced while the alias is constant,
46   * but updated to point to the new function on each update
47   */
48  export class VersionedLambda extends Construct implements IGrantable, IDependable {
49    /** Required by IGrantable, which allows other constructs to grant permissions to this one */
50    get grantPrincipal(): IPrincipal {
51      return this.#alias.grantPrincipal;
52    }
53
54    /** The log group associated with the most current version of the function */
55    get logGroup(): ILogGroup {
56      return this.#fn.logGroup;
57    }
58
59    /** Role that is used when executing the function*/
60    get role(): IRole | undefined {
61      return this.#alias.role;
62    }
63
64    /** The ARN for the funciton. This ARN points the to alias and so is stable across version updates */
65    get functionArn(): string {
66      return this.#alias.functionArn;
67    }
68
69    /** Returns the DLQ associated with this function */
70    get deadLetterQueue(): IQueue | undefined {
71      return this.#fn.deadLetterQueue;
72    }
73
74    /** Allows adding permissions to the versioned function */
75    get addPermission() {
76      return this.#alias.addPermission.bind(this.#alias);
77    }
78
```

Example: Lambda constructor as code

```
102     /** Construct the versioned lambda
103      *
104      * @param { Construct } scope The scope in which to construct this lambda
105      * @param { string } id The name of the lambda
106      * @param { VersionedLambdaProps } props The properties for this lambda
107      */
108     constructor(scope: Construct, id: string, props: VersionedLambdaProps) {
109       super(scope, id);
110
111       // For each Lambda, we do the following:
112       // (1) Deploy the lambda (`new lambda.Function(...)`)
113       //
114       // (2) Publish a numbered version and make an alias to that version.
115       //     This lets us provision concurrency (and potentially other things).
116       //     We follow the CDK documentation here:
117       //        https://docs.aws.amazon.com/cdk/api/v2/docs/aws-cdk-lib.aws_lambda.Version.html
118
119       // (1) The underlying function that gets replaced every time there's an update to the code
120       this.#fn = new LambdaFunction(this, "Function", {
121         ...DEFAULT_ARGS,
122         code: Code.fromAsset(`${__dirname}/../../build/${props.assetName}.zip`),
123         ...props,
124       });
125
126       // (2) The underlying function that gets replaced every time there's an update to the code
127       this.#alias = new Alias(this, "Alias", {
128         aliasName: `${id}Alias`,
129         version: this.#fn.currentVersion,
130         ...props,
131       });
132     }
133   }
```

Whole SDK of AWS stuff to interact with

```
1   import { Duration } from "aws-cdk-lib/core";
2   import {
3     IGrantable,
4     IManagedPolicy,
5     IPrincipal,
6     IRole,
7     PolicyStatement,
8   } from "aws-cdk-lib/aws-iam";
9   import {
10    FunctionProps,
11    Runtime,
12    Tracing,
13    Function as LambdaFunction,
14    Code,
15    Alias,
16    IFunction,
17    AliasProps,
18  } from "aws-cdk-lib/aws-lambda";
19  import { ILogGroup } from "aws-cdk-lib/aws-logs";
20  import { Construct, IDependable } from "constructs";
21  import { IQueue } from "aws-cdk-lib/aws-sqs";
```

Pros/cons. Discuss.