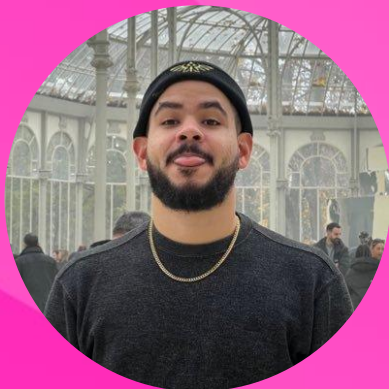


¿Cuál es mejor? Una comparativa entre laC imperativa vs declarativa

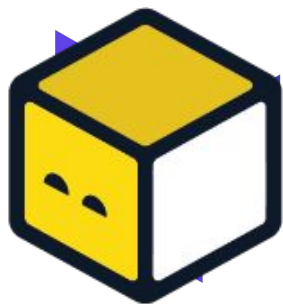


Christian Polanco
Adevinta

Esta presentación no está enfocada
en herramientas de configuración
como Ansible y Chef

laC tiene mucha oferta

Declarativa



Imperativa




Declarativo

Quiero
una
m2.xlarge



Imagen de fabrikasimf en Freepik

Declarativo



```
resource "aws_instance" "example" {  
  ami          = "ami-0c55b159cbfafa1f0"  
  instance_type = "m2.xlarge"  
  
  tags = {  
    Name = "example-instance"  
  }  
}
```

Declarativo



Imagen de fabrikasimf en Freepik

Imperativo

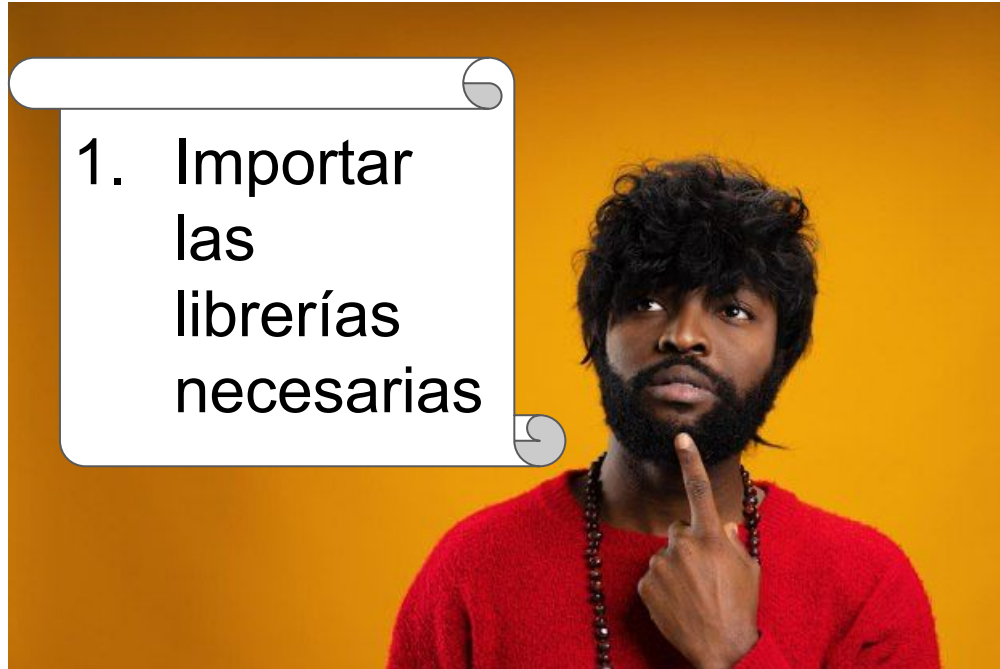



Imagen de fabrikasimf en Freepik


Imperativo



2. Crear
variable con la
configuración

Imagen de fabrikasimf en Freepik

Imperativo



```
import * as cdk from '@aws-cdk/core'
import * as ec2 from '@aws-cdk/aws-ec2'

export class MyCdkProjectStack extends cdk.Stack {
  constructor(scope: cdk.Construct, id: string, props?: cdk.StackProps) {
    super(scope, id, props)

    const instance = new ec2.Instance(this, 'MyInstance', {
      instanceType: ec2.InstanceType.of(
        ec2.InstanceClass.T2,
        ec2.InstanceSize.MICRO
      ),
      machineImage: ec2.MachineImage.latestAmazonLinux(),
    })
  }
}

const app = new cdk.App()
new MyCdkProjectStack(app, 'MyCdkProjectStack')
```

Imperativo



Imagen de fabrikasimf en Freepik

¿Y cual es mejor?



Imagen de fabrikasimf en Freepik

¿Y cual es mejor?



Declarativo

Imagen de fabrikasimf en Freepik

¡Muchas Gracias!

Okay okay... Tal vez no es tan simple.

Imperativo tiene una gran desventaja

```
import * as cdk from '@aws-cdk/core'
import * as ec2 from '@aws-cdk/aws-ec2'

export class MyCdkProjectStack extends cdk.Stack {
  constructor(scope: cdk.Construct, id: string, props?: cdk.StackProps) {
    super(scope, id, props)

    const instance = new ec2.Instance(this, 'MyInstance', {
      instanceType: ec2.InstanceType.of(
        ec2.InstanceClass.T2,
        ec2.InstanceSize.MICRO
      ),
      machineImage: ec2.MachineImage.latestAmazonLinux(),
    })
  }
}

const app = new cdk.App()
new MyCdkProjectStack(app, 'MyCdkProjectStack')
```

Imperativo tiene una gran desventaja

```
import * as cdk from '@aws-cdk/core'
import * as ec2 from '@aws-cdk/aws-ec2'

export class MyCdkProjectStack extends cdk.Stack {
  constructor(scope: cdk.Construct, id: string, props?: cdk.StackProps) {
    super(scope, id, props)

    const instance = new ec2.Instance(this, 'MyInstance', {
      instanceType: ec2.InstanceType.of(
        ec2.InstanceClass.T2,
        ec2.InstanceSize.MICRO
      ),
      machineImage: ec2.MachineImage.latestAmazonLinux(),
    })
  }
}

const app = new cdk.App()
new MyCdkProjectStack(app, 'MyCdkProjectStack')
```



Imperativo tiene una gran desventaja

```
import * as cdk from '@aws-cdk/core'
import * as ec2 from '@aws-cdk/aws-ec2'


function createEC2Instance(): ec2.Instance {
  return new ec2.Instance(this, 'MyInstance', {
    instanceType: ec2.InstanceType.of(
      ec2.InstanceClass.T2,
      ec2.InstanceSize.MICRO
    ),
    machineImage: ec2.MachineImage.latestAmazonLinux(),
  })
}

export class MyCdkProjectStack extends cdk.Stack {
  constructor(scope: cdk.Construct, id: string, props?: cdk.StackProps) {
    super(scope, id, props)
    const instance = createEC2Instance()
  }
}
```

Hagamos zoom




```
function createEC2Instance(): ec2.Instance {  
    return new ec2.Instance(this, 'MyInstance', {  
        instanceType: ec2.InstanceType.of(  
            ec2.InstanceClass.T2,  
            ec2.InstanceSize.MICRO  
        ),  
        machineImage: ec2.MachineImage.latestAmazonLinux(),  
    })  
}
```

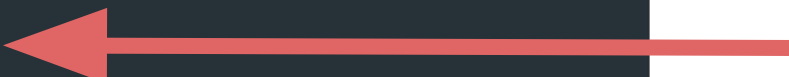


```
const env = 'dev'
```


```
function createEC2Instance(): ec2.Instance {  
  let instanceType: ec2.InstanceType  
  if (env === 'dev') {  
    instanceType = ec2.InstanceType.of(  
      ec2.InstanceClass.T2,  
      ec2.InstanceSize.MICRO  
    )  
  } else {  
    instanceType = ec2.InstanceType.of(  
      ec2.InstanceClass.M3,  
      ec2.InstanceSize.XLARGE  
    )  
  }  
  
  return new ec2.Instance(this, 'MyInstance', {  
    instanceType: instanceType,  
    machineImage: ec2.MachineImage.latestAmazonLinux(),  
  })  
}
```




```
const env = 'dev'
```



```
function createEC2Instance(): ec2.Instance {  
  let instanceType: ec2.InstanceType  
  if (env = 'dev') {  
    instanceType = ec2.InstanceType.of(  
      ec2.InstanceClass.T2,  
      ec2.InstanceSize.MICRO  
    )  
  } else {  
    instanceType = ec2.InstanceType.of(  
      ec2.InstanceClass.M3,  
      ec2.InstanceSize.XLARGE  
    )  
  }  
  
  return new ec2.Instance(this, 'MyInstance', {  
    instanceType: instanceType,  
    machineImage: ec2.MachineImage.latestAmazonLinux(),  
  })  
}
```



```
const env = getEnvironmentFromConfig()
```



```
function createEC2Instance(): ec2.Instance {
  let instanceType: ec2.InstanceType
  if (env === 'dev') {
    instanceType = ec2.InstanceType.of(
      ec2.InstanceClass.T2,
      ec2.InstanceSize.MICRO
    )
  } else {
    instanceType = ec2.InstanceType.of(
      ec2.InstanceClass.M3,
      ec2.InstanceSize.XLARGE
    )
  }

  return new ec2.Instance(this, 'MyInstance', {
    instanceType: instanceType,
    machineImage: ec2.MachineImage.latestAmazonLinux(),
  })
}
```

```
const env = getEnvironmentFromConfig()

function createEC2Instance(): ec2.Instance {
  let instanceType: ec2.InstanceType
  if (env === 'dev') {
    instanceType = ec2.InstanceType.of(
      ec2.InstanceClass.T2,
      ec2.InstanceSize.MTGD0
    )
  } else {
    instanceType = ec2.InstanceType.of(
      ec2.InstanceClass.T2,
      ec2.InstanceSize.MTGD0
    )
  }

  return new ec2.Instance({
    instanceType,
    // ...other properties
  })
}
```



Comparación con ~~Terr~~ OpenTofu



```
resource "aws_instance" "example" {  
  ami          = "ami-0c55b159cbfaffe1f0"  
  instance_type = local.env == dev ? "t2.micro" : "m3.xlarge"  
  
  tags = {  
    Name = "example-instance"  
  }  
}
```

Comparación con ~~Terr~~ OpenTofu

```
resource "aws_instance" "example" {  
  ami          = "ami-0c55b159cbfaffe1f0"  
  instance_type = local.env == dev ? "t2.micro" : "m3.xlarge"  
  
  tags = {  
    Name = "example-instance"  
  }  
}
```



El problema tiende a ser las **funciones**

Las funciones definen ***procesos***,
pero con laC lo que queremos es
definir ***estados***.

Las funciones pueden de **manera no intencional** abstraer el estado final deseado.



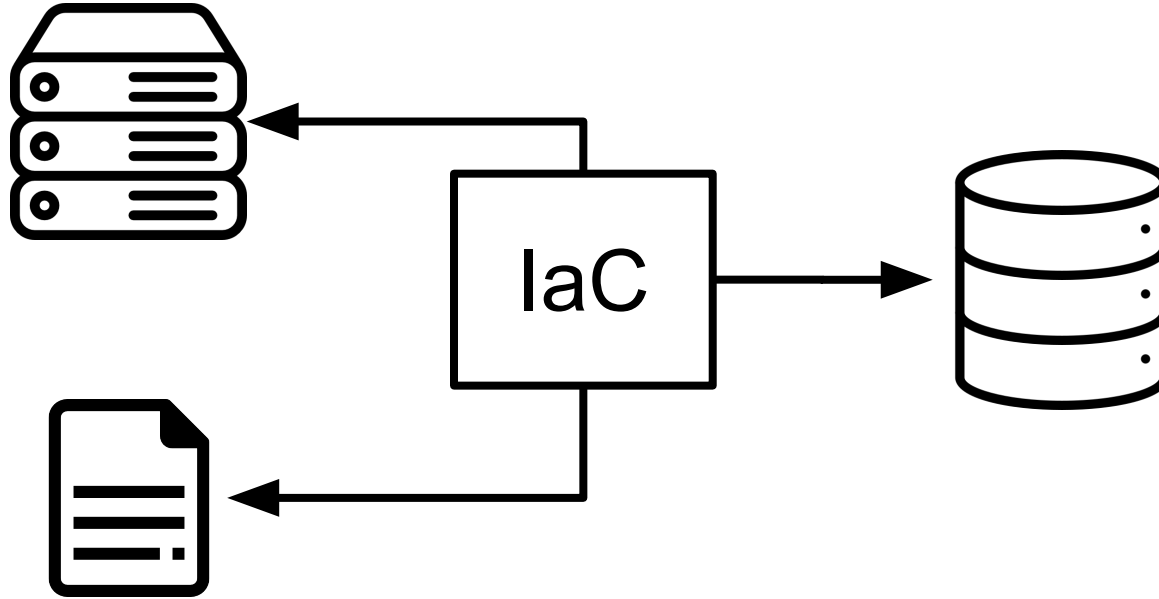
Yo
laC imperativo

ustedes


Francis Borgia Helping a
Dying Impenitent by Goya

Las herramientas imperativas **tienen**
sus casos de uso


Si manejas muchas fuentes de configuración



Icónos de Bartama Graphic, Pixel Perfect,
Smartline en Freepik



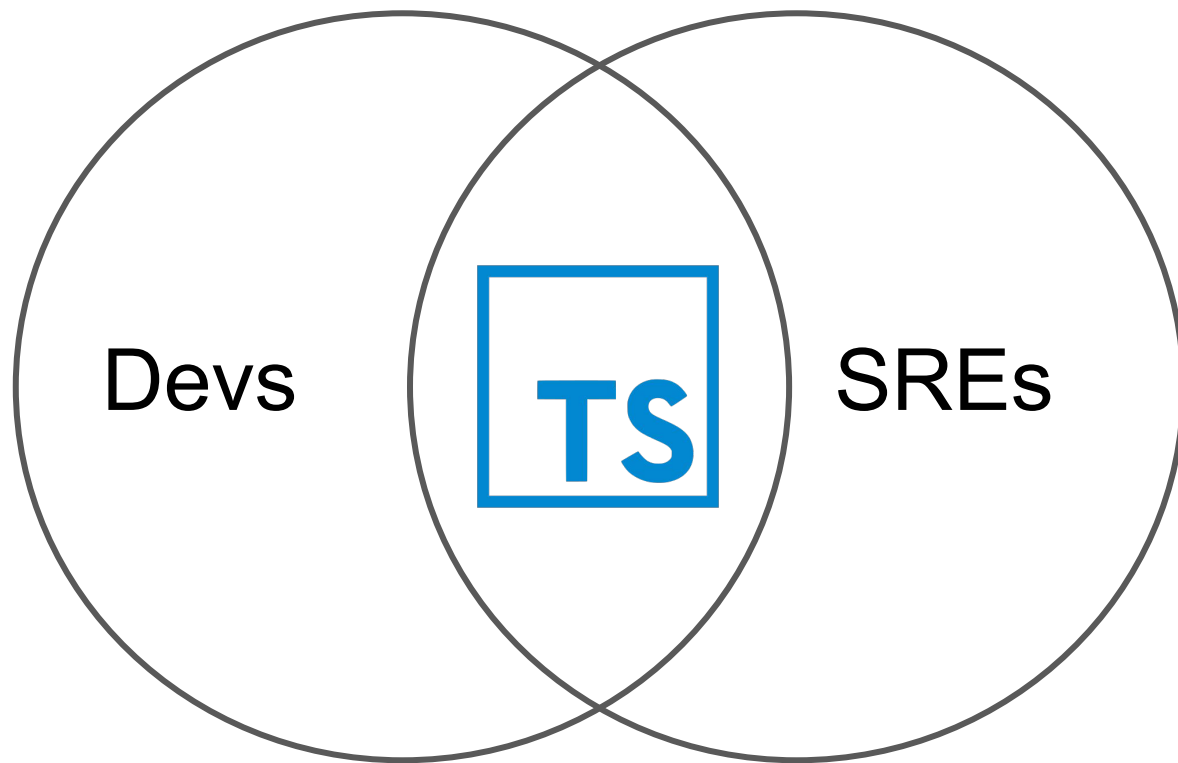
```
const env = getEnvironmentFromConfig()
```



```
function createEC2Instance(): ec2.Instance {
  let instanceType: ec2.InstanceType
  if (env === 'dev') {
    instanceType = ec2.InstanceType.of(
      ec2.InstanceClass.T2,
      ec2.InstanceSize.MICRO
    )
  } else {
    instanceType = ec2.InstanceType.of(
      ec2.InstanceClass.M3,
      ec2.InstanceSize.XLARGE
    )
  }

  return new ec2.Instance(this, 'MyInstance', {
    instanceType: instanceType,
    machineImage: ec2.MachineImage.latestAmazonLinux(),
  })
}
```

Si los developers pueden definir parte de la
laC



En varios casos depende de la herramienta






Si vas a usar una herramienta imperativa:

K. I. S. S.

Keep It Simple, SRE!

Si vas a usar una herramienta imperativa:

1. Limita el uso de funciones. 
2. Utiliza las clases por defecto de la librería. 
3. Abstrae lo menos posible. 

Bottom line

1. Declarativo probablemente cubre tus necesidades.
2. En el software **no existen** las balas plateadas.

Referencias:

- **Why we use Terraform and not Chef, Puppet, Ansible, Pulumi, or CloudFormation**

<https://blog.gruntwork.io/why-we-use-terraform-and-not-chef-puppet-ansible-saltstack-or-cloudformation-7989dad2865c>

- **Stop Using AWS CDK** <https://blog.devspecops.com/stop-using-aws-cdk-b2052abb4cb5>

- **The benefits of moving from Terraform to Pulumi**

<https://medium.com/@christopherlenard/the-benefits-of-moving-from-terraform-to-pulumi-7e01a3ab8f43>