HashiCorp

Terraform

Extending CDK for Terraform constructs

Making our lives easier with L2 constructs

About

Ansgar Mertens

Software Engineer, CDK for Terraform

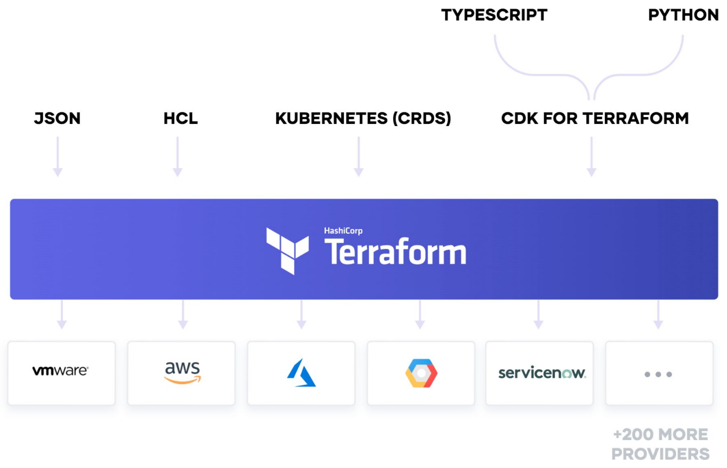
CDK for Terraform

Use Terraform with programming constructs in high-level languages

Supported Languages

- TypeScript / JavaScript
- Java
- Python
- C#

Soon: Golang



Example

HCL

```
resource "aws_instance" "web" {
  ami           = "ami-0848da720bb07de35"
  instance_type = "t3.micro"

  tags = {
    Name = "HelloWorld"
  }
}
```

TypeScript

```
new Instance(this, 'web', {
    ami: 'ami-0848da720bb07de35',
    instanceType: 't3.micro',
    tags: {
        Name: 'HelloWorld'
    }
});
```

Goals

Why should we extend constructs?

- Show intent instead of implementation details
- Move verbosity to a separate file
- Build reusable building blocks
- Keep things DRY

AWS CDK

Layer 2 constructs

```
const lambda = new lambda.Function(this, 'Lambda', { /* ... */ });
const bucket = new Bucket(this, 'MyBucket');

bucket.grantReadWrite(lambda);
```

Example from AWS CDK docs

Granting Access

Combining arbitrary Terraform resources of different providers

- Providers used in demo
  - AWS
  - DigitalOcean

Implementation

Extending a CDK for Terraform construct

```
import { Grantor } from "./lib/grants";
import { DatabaseCluster as BaseDatabaseCluster, Droplet } from "./.gen/providers/digitalocean";

const DatabaseCluster = Grantor(BaseDatabaseCluster);

const db = new DatabaseCluster(this, 'db', {...});
const droplet = new Droplet(this, 'droplet', {...});

db.grantAccess(droplet);
```

Show us the real code!

Recap

Extending a CDK for Terraform construct

```
import { Grantor } from "./lib/grants";
import { Droplet } from "./.gen/providers/digitalocean";
import { TerraformAwsModulesRdsAws as BaseTerraformAwsModulesRdsAws } from
'./.gen/modules/terraform-aws-modules/rds/aws'
import { DropletToRdsViaSecurityGroupGrantStrategy } from 'somewhere'

const TerraformAwsModulesRdsAws = Grantor(BaseTerraformAwsModulesRdsAws);

const securityGroup = new SecurityGroup(this, 'sg-rds-cdkday-test', {...});
const db = new TerraformAwsModulesRdsAws(this, 'db', {...});
const droplet = new Droplet(this, 'droplet', {...});

db.grantAccess(droplet, new DropletToRdsViaSecurityGroupGrantStrategy(securityGroup));
```

What can we do now?

Endless possibilities await

- Build abstractions for combinations of resources
- Use generics in TypeScript to catch invalid configuration very early on
- Use all the existing Terraform modules and providers

– 982 providers, 5681 modules & counting (registry.terraform.io)

  – Modules implementing lots of best practices already

- Use our own modules

What’s next?

From prototype to usable experiments

- Make this available in more languages via JSII
  - JSII currently does not support generics
- A lot more experimentation needed
  - Across many different resource types
  - What syntax is idiomatic in other languages

Links

Get started with the CDK for Terraform

https://cdk.tf

https://github.com/ansgarm/talk-cdkday-2021

https://learn.hashicorp.com/tutorials/terraform/cdktf

https://discuss.hashicorp.com/c/terraform-core/cdk-for-terraform/47

Thank You

ansgar@hashicorp.com
www.hashicorp.com