Cloud Development Kit for Terraform (CDKTF)

Nuremberg AWS User Group
June 12, 2023

Team inovex
Karlsruhe · Köln · München · Hamburg
Berlin · Stuttgart · Pforzheim · Erlangen

inovex

Bernd Kaiser

in Bernd Kaiser

@meldron

Software Developer at inovex Erlangen

Focus:

- Web
- Security

JSCC23 Organizer

inovex

Thanks invoex!

inovex is an IT project center driven by innovation and quality, focusing its services on ‘Digital Transformation’.

Our current focus:

- Agile Transformation
- Product Development Workshops
- E-Health
- Recommender Systems
- Generative AI

inovex Meetup - Android Open Source Project & KAIZEN

June 14, 2023, 18:00, Design Offices in Erlangen

- **The principles of KAIZEN** 🇺🇸
  Bridging the gap in understanding and application to enhance problem-solving and enrich experiences in engineering life
- **Embedded System with the Android Open Source Project** 🇩🇪
  Advantages and challenges of embedded Android compared to Linux-based systems

https://www.meetup.com/inovex-meetup-erlangen/events/293171627/

Agenda

- Infrastructure as Code Overview
- What is Terraform?
- CDKTF
- Testing
- TF / CDKTF Interoperability

Infrastructure as Code

What is IaC and what tools are currently

Infrastructure as Code (IaC)

Managing and provisioning computing infrastructure through machine-readable definition files

IaC Advantages

- Automates IT infrastructure management
- Reduces errors, enhances replication speed
- Ensures consistent, predictable deployments
- Facilitates collaboration, increases efficiency
- Streamlines path from development to production

IaC Tools

- AWS CloudFormation / AWS CDK
- Serverless Stack Toolkit (SST)
- **Terraform / CDKTF**
- Pulumi
- Chef (Infra)
- Ansible

Terraform

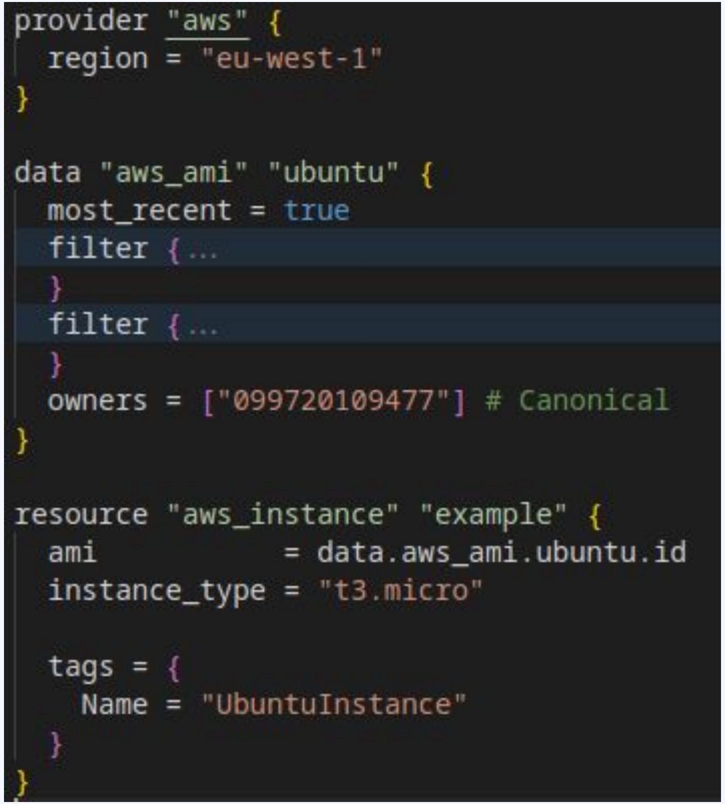
One to rule them all

Terraform

- Open-source Infrastructure as Code tool
- Developed by HashiCorp, provides declarative language
- **Uses HashiCorp Configuration Language (HCL)**
- Supports many cloud providers and services:
  AWS, Azure, GCP, Alibaba, Cloudflare, Hetzner Cloud, …

Terraform

- Infrastructure is defined in configuration files
- Terraform generates an execution plan describing actions
- Terraform manages resources with a (shared) state file



```
provider "aws" {
  region = "eu-west-1"
}

data "aws_ami" "ubuntu" {
  most_recent = true
  filter {...
  }
  filter {...
  }
  owners = ["099720109477"] # Canonical
}

resource "aws_instance" "example" {
  ami           = data.aws_ami.ubuntu.id
  instance_type = "t3.micro"

  tags = {
    Name = "UbuntuInstance"
  }
}
```

CDKTF

TypeScript time!

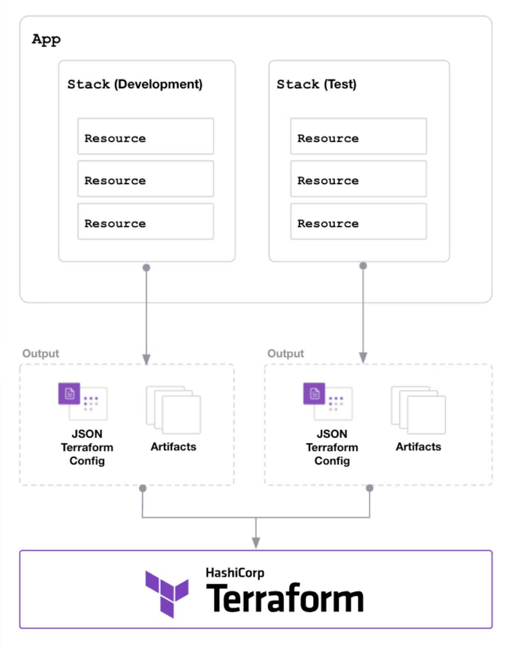
Terraform Cloud Development Kit

- Use familiar programming languages and Development Environments
- **No HCL knowledge needed**
- All Terraform providers available
- Supports **TypeScript**, Python, Java, C#, and Go

CDKTF Application Architecture

- App: container for the infrastructure configuration
- Stack: collection of resources with separate state
- Resource: definition of one or more infrastructure objects

Everything is implemented by extending `Constructs`



Source: https://developer.hashicorp.com/terraform/cdktf/concepts/cdktf-architecture

Constructs

- Constructs serve as the building blocks of applications
- Structured hierarchically
- Each construct symbolizes a "piece of system state"
- Composition of Constructs can be tested
- Aspects: Visitor pattern to apply an operation to all constructs within a given scope

Requirements

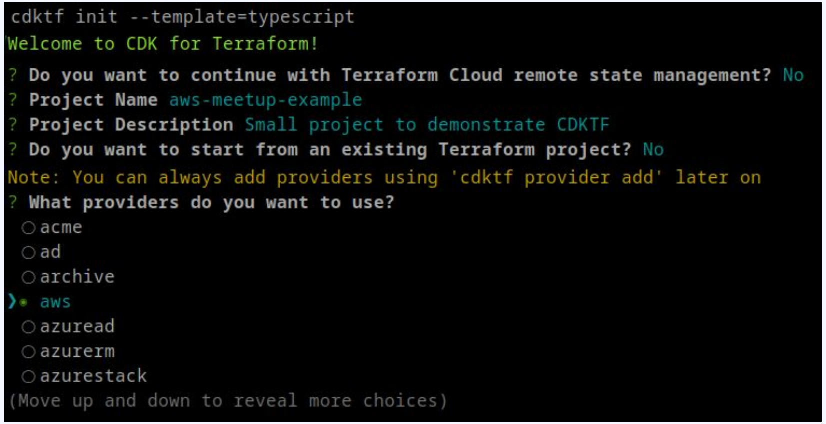
- Terraform CLI (1.2+)
- NodeJS (v16+)
- AWS CLI
- (Docker)
- (Visual Studio Code + devcontainers)

CDKTF AWS Example Project

https://github.com/meldron/aws-meetup-nuremberg-cdktf

Project Setup

- Use .devcontainer/devcontainer.json
- npm install --global cdktf-cli@latest
- cdktf init --template=typescript



```
cdktf init --template=typescript
Welcome to CDK for Terraform!
? Do you want to continue with Terraform Cloud remote state management? No
? Project Name aws-meetup-example
? Project Description Small project to demonstrate CDKTF
? Do you want to start from an existing Terraform project? No
Note: You can always add providers using 'cdktf provider add' later on
? What providers do you want to use?
 ○acme
 ○ad
 ○archive
❯◉ aws
 ○azuread
 ○azurerm
 ○azurestack
(Move up and down to reveal more choices)
```

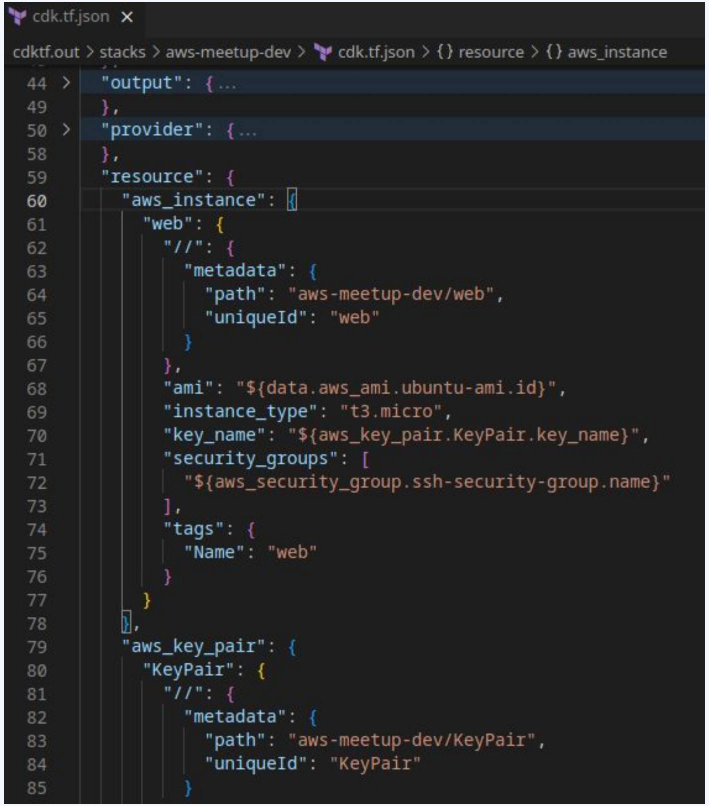
```typescript
import { Construct } from "constructs";
import { App, TerraformStack } from "cdktf";

class MyStack extends TerraformStack {
  constructor(scope: Construct, id: string) {
    super(scope, id);
    }
}

const app = new App();
new MyStack(app, "dev");
app.synth();
```

Create simple EC2 Resource

```
const ubuntuAmi = new DataAwsAmi(this, "ubuntu-ami", {
  filter: [{
    name: "name",
    values: ["ubuntu/images/hvm-ssd/ubuntu-focal-20.04-amd64-server-*"],
  }],
  owners: ["099720109477"],
});

const ec2 = new Instance(this, "web", {
  ami: ubuntuAmi.id,
  instanceType: "t3.micro",
  tags: {
    Name: "web",
  },
});
```

cdktf synth

- Translates CDKTF application to JSON
- JSON represents Terraform configuration
- Outputs into a designated directory per Stack


cdk.tf.json ×
cdktf.out > stacks > aws-meetup-dev > cdk.tf.json > {} resource > {} aws_instance

```
  "output": {...
  },
  "provider": {...
  },
  "resource": {
    "aws_instance": {
      "web": {
        "//": {
          "metadata": {
            "path": "aws-meetup-dev/web",
            "uniqueId": "web"
          }
        },
        "ami": "${data.aws_ami.ubuntu-ami.id}",
        "instance_type": "t3.micro",
        "key_name": "${aws_key_pair.KeyPair.key_name}",
        "security_groups": [
          "${aws_security_group.ssh-security-group.name}"
        ],
        "tags": {
          "Name": "web"
        }
      }
    },
    "aws_key_pair": {
      "KeyPair": {
        "//": {
          "metadata": {
            "path": "aws-meetup-dev/KeyPair",
            "uniqueId": "KeyPair"
          }
```

Synthesized Files

- Terraform commands can be used as usual:
  - plan, apply, destroy
  - import, state, ...
- JSON files could be checked in (are ignored per default)
- Contain a lock file
- Install provider dependencies (e.g, linux_amd64/terraform-provider-aws_v4.66.1_x5)

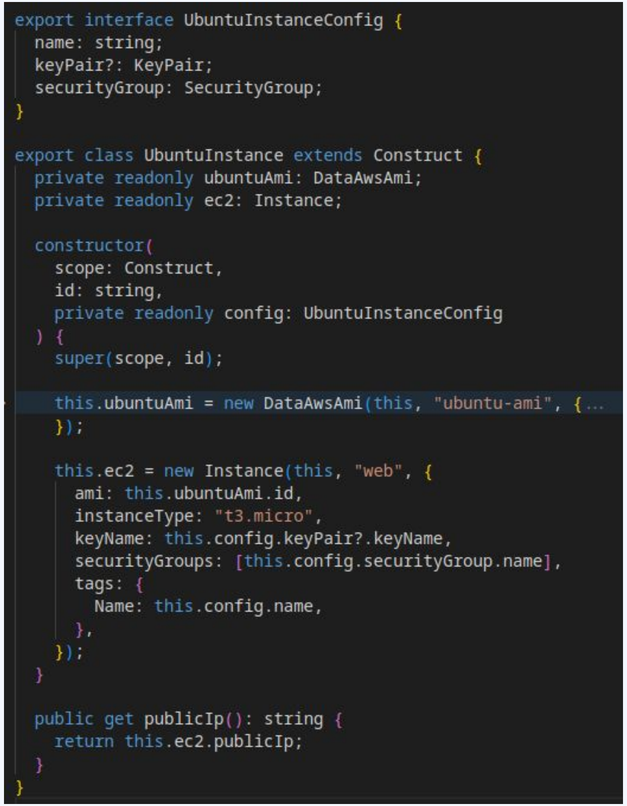
Directory **cdktf.out**:

```
manifest.json
stacks/aws-meetup-dev/.terraform/
stacks/aws-meetup-dev/.terraform.lock.hcl
stacks/aws-meetup-dev/cdk.tf.json
```

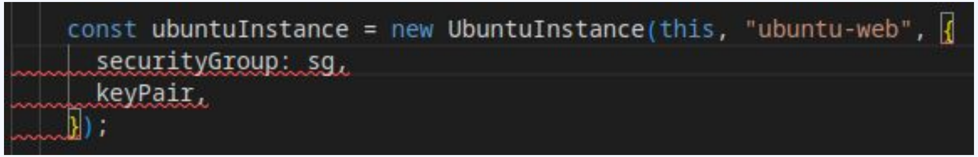
cdktf diff / deploy

- diff is the equivalent of tf plan
- deploy is the equivalent of tf plan & tf apply
- Both commands automatically synthesize

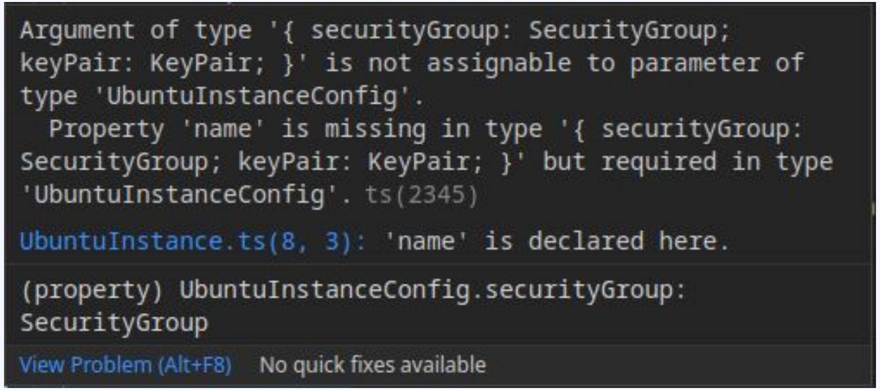
Custom Constructs



```typescript
export interface UbuntuInstanceConfig {
  name: string;
  keyPair?: KeyPair;
  securityGroup: SecurityGroup;
}

export class UbuntuInstance extends Construct {
  private readonly ubuntuAmi: DataAwsAmi;
  private readonly ec2: Instance;

  constructor(
    scope: Construct,
    id: string,
    private readonly config: UbuntuInstanceConfig
  ) {
    super(scope, id);

    this.ubuntuAmi = new DataAwsAmi(this, "ubuntu-ami", {…
    });

    this.ec2 = new Instance(this, "web", {
      ami: this.ubuntuAmi.id,
      instanceType: "t3.micro",
      keyName: this.config.keyPair?.keyName,
      securityGroups: [this.config.securityGroup.name],
      tags: {
        Name: this.config.name,
      },
    });
  }

  public get publicIp(): string {
    return this.ec2.publicIp;
  }
}
```



```typescript
const ubuntuInstance = new UbuntuInstance(this, "ubuntu-web", {
  securityGroup: sg,
  keyPair,
});
```



```
Argument of type '{ securityGroup: SecurityGroup;
keyPair: KeyPair; }' is not assignable to parameter of
type 'UbuntuInstanceConfig'.
  Property 'name' is missing in type '{ securityGroup:
SecurityGroup; keyPair: KeyPair; }' but required in type
'UbuntuInstanceConfig'. ts(2345)
UbuntuInstance.ts(8, 3): 'name' is declared here.
(property) UbuntuInstanceConfig.securityGroup:
SecurityGroup
```

Constructs

- Strict type checking
- Properties can be validated
- **Conditional Behavior**
- Unique name for each instance (Construct#Id)

```
class CustomS3Bucket extends S3Bucket {
  constructor(scope: Construct, name: string) {
    super(scope, name);
  }

  public giveAccess(
    item: LambdaFunction | CloudfrontDistribution
  ) {
    if (item instanceof LambdaFunction) {
      // Lambda IAM Policy for to access S3
    }

    if (item instanceof CloudfrontDistribution) {
      // CloudFront IAM Policy to access S3
    }
  }
}
```

Testing

it should work

Unit Testing

- Stack (Testing.synth) or Scope (Testing.synthScope) based
- Write Assertions
  - toHaveResource / toHaveResourceWithProperties
  - toHaveDataSource / toHaveDataSourceWithProperties
  - toHaveProvider / toHaveProviderWithProperties
- Snapshot testing (TypeScript only)

  (compares it to a reference snapshot file stored alongside the test)
- Terraform Integration
  - toBeValidTerraform
  - toPlanSuccessfully

Construct Unit Testing

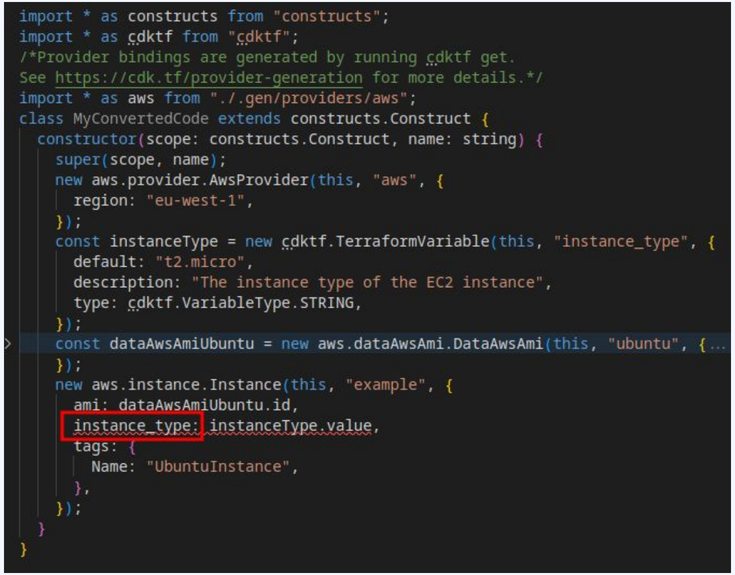
```
describe("UbuntuInstance", () => {
  it("should set tags.Name with supplied name", () => {
    expect(
      Testing.synthScope((scope) => {
        new UbuntuInstance(scope, "ubuntu", {
          name: "test-123",
          securityGroup: new SecurityGroup(scope, "sg"),
        });
      })
    ).toHaveResourceWithProperties(Instance, { tags: { Name: "test-123" } });
  });
});
```

Interoperability

TF 🤝 CDKTF

cdktf convert

- Converts HCL to language of choice
- cat main.tf | cdktf convert > imported.ts
- Bit buggy:

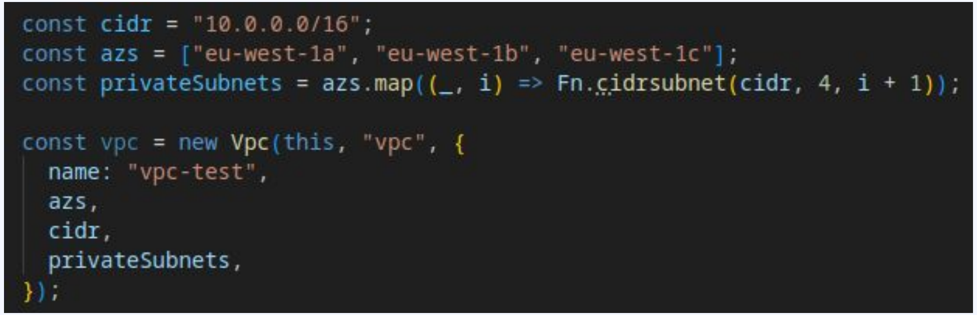


```typescript
import * as constructs from "constructs";
import * as cdktf from "cdktf";
/*Provider bindings are generated by running cdktf get.
See https://cdk.tf/provider-generation for more details.*/
import * as aws from "./.gen/providers/aws";
class MyConvertedCode extends constructs.Construct {
  constructor(scope: constructs.Construct, name: string) {
    super(scope, name);
    new aws.provider.AwsProvider(this, "aws", {
      region: "eu-west-1",
    });
    const instanceType = new cdktf.TerraformVariable(this, "instance_type", {
      default: "t2.micro",
      description: "The instance type of the EC2 instance",
      type: cdktf.VariableType.STRING,
    });
    const dataAwsAmiUbuntu = new aws.dataAwsAmi.DataAwsAmi(this, "ubuntu", {...
    });
    new aws.instance.Instance(this, "example", {
      ami: dataAwsAmiUbuntu.id,
      instance_type: instanceType.value,
      tags: {
        Name: "UbuntuInstance",
      },
    });
  }
}
```

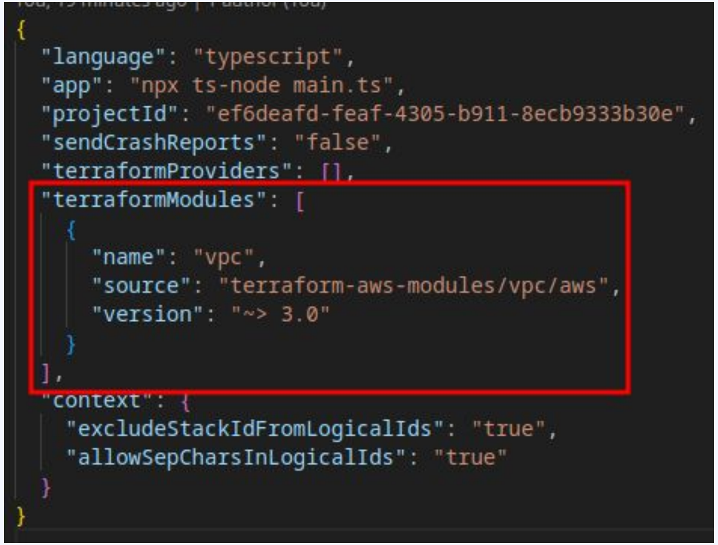
TF Modules from CDKTF

Any public or private module can be used:

- Add module to cdktf.json
- Generate module bindings (cdktf get)
- Configure module
  (map inputs must be specified as strings)



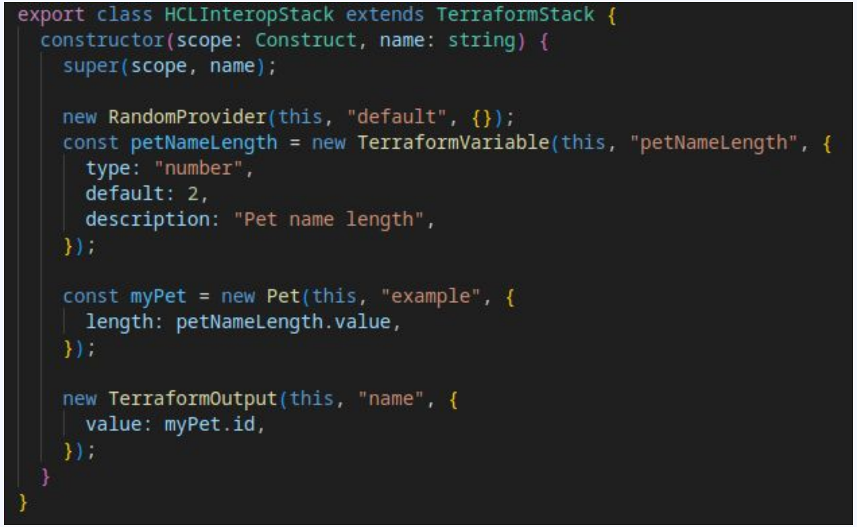
```typescript
const cidr = "10.0.0.0/16";
const azs = ["eu-west-1a", "eu-west-1b", "eu-west-1c"];
const privateSubnets = azs.map((_, i) => Fn.cidrsubnet(cidr, 4, i + 1));

const vpc = new Vpc(this, "vpc", {
  name: "vpc-test",
  azs,
  cidr,
  privateSubnets,
});
```



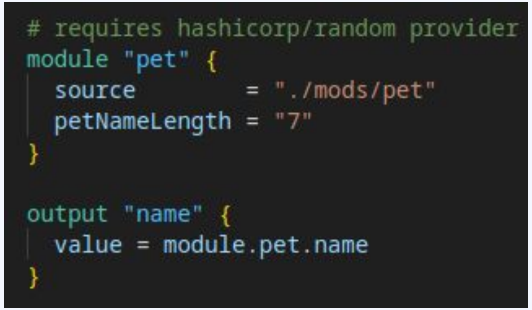
```json
You, 19 minutes ago | 1 author (You)
{
  "language": "typescript",
  "app": "npx ts-node main.ts",
  "projectId": "ef6deafd-feaf-4305-b911-8ecb9333b30e",
  "sendCrashReports": "false",
  "terraformProviders": [],
  "terraformModules": [
    {
      "name": "vpc",
      "source": "terraform-aws-modules/vpc/aws",
      "version": "~> 3.0"
    }
  ],
  "context": {
    "excludeStackIdFromLogicalIds": "true",
    "allowSepCharsInLogicalIds": "true"
  }
}
```

CDKTF Modules from TF

- Create a Class which extends TerraformStack
- Use TerraformVariable for inputs & TerraformOutput for outputs
- cdktf synth to create the cdktf.json file
- Copy file into a module directory inside your TF project
- Reference module like any other TF module



```
export class HCLInteropStack extends TerraformStack {
  constructor(scope: Construct, name: string) {
    super(scope, name);

    new RandomProvider(this, "default", {});
    const petNameLength = new TerraformVariable(this, "petNameLength", {
      type: "number",
      default: 2,
      description: "Pet name length",
    });

    const myPet = new Pet(this, "example", {
      length: petNameLength.value,
    });

    new TerraformOutput(this, "name", {
      value: myPet.id,
    });
  }
}
```



```
# requires hashicorp/random provider
module "pet" {
  source        = "./mods/pet"
  petNameLength = "7"
}

output "name" {
  value = module.pet.name
}
```

Conclusion

- 🌤️ Use the Tools you know
- 🌤️ No new DSL
- 🌤️ All of TF can still be used
- 🌤️ Good Quality of Generated Types & Co
- 🌤️ Good Community

- 🌧️ Pre v1.0
- 🌧️ Documentation sometimes outdated
- 🌧️ Rough edges here & there
- 🌧️ Possible Segregation across Teams

Resources

- https://developer.hashicorp.com/terraform/cdktf
- CDK for Terraform Improves HCL Conversion and Terraform Cloud Interactions
- 🎥 When, Why, and How to Use the CDK for Terraform

Thank you!

inovex is an IT project center driven by innovation and quality, focusing its services on ‘Digital Transformation’.

- founded in 1999
- 500+ employees
- 8 offices across Germany

www.inovex.de

Bernd Kaiser

Software Developer

bernd.kaiser@inovex.de

inovex