CDK Day May 2022 - Track 2

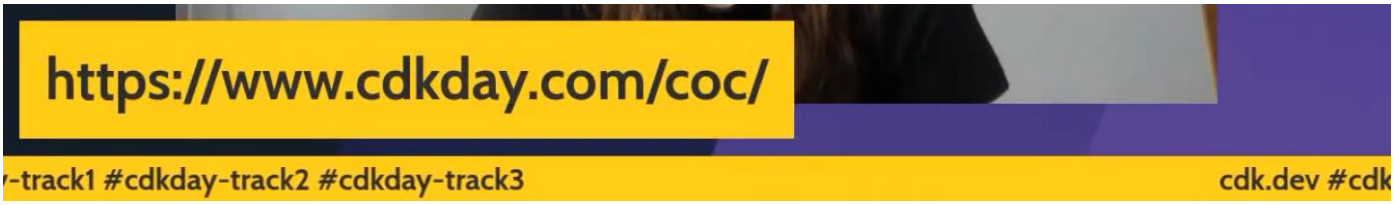
CDK Day
2.15K subscribers
Subscribe

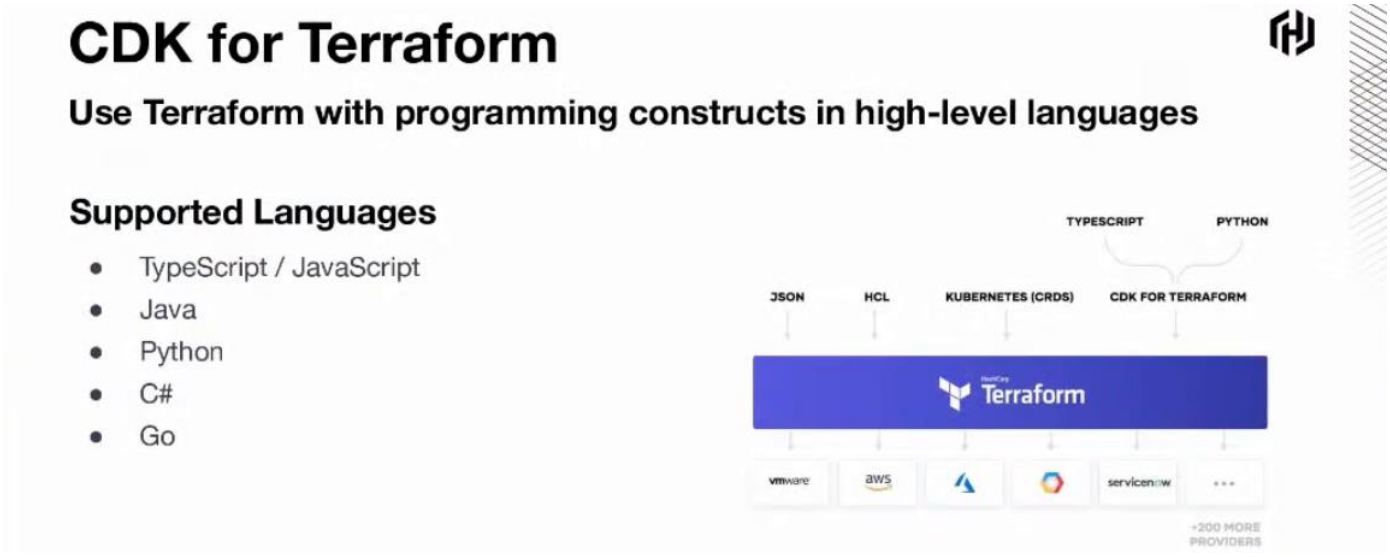
32 | Share | Download | Clip | Save

3,026 views Streamed live on May 26, 2022
Cloud Development Kit (CDK) is a developer tool built on the open source Constructs model. CDK Day is a one-day conference that attempts to showcase the brightest and best of CDK from AWS CDK, CDK for Terraform (CDKtf), CDK for Kubernetes (cdk8s), and projen. Let's talk serverless, Kubernetes and multi-cloud all on the same day.

This is the livestream for Track 2 of CDK Day, featuring talks from Ansgar Mertens, Jenna Krolick, Bill Penberthy, and more. For the full agenda, please see https://www.cdkday.com/.

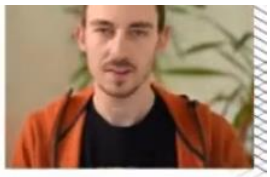
For the livestream of talks from Track 1, please head over to https://bit.ly/cdkday2022-track1
For the livestream of talks from Track 3, please head over to https://bit.ly/cdkday2022-track3





It allows you to use any high-level language with Terraform's HCL (Hashicorp Configuration Language)

Example



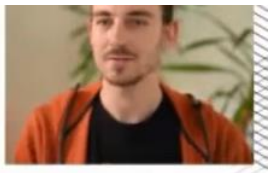
HCL

```
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'
    }
});
```

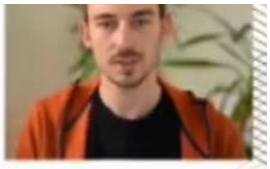
A year ago ...



April 2021 - Extending constructs of the CDK for Terraform, Ansgar Mertens

AWS CDK

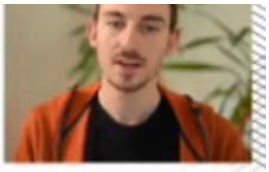
Layer 2 constructs



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

bucket.grantReadWrite(lambda);
```

AWS Adapter

Use AWS CDK constructs with CDKTF

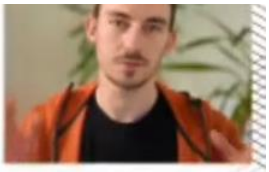


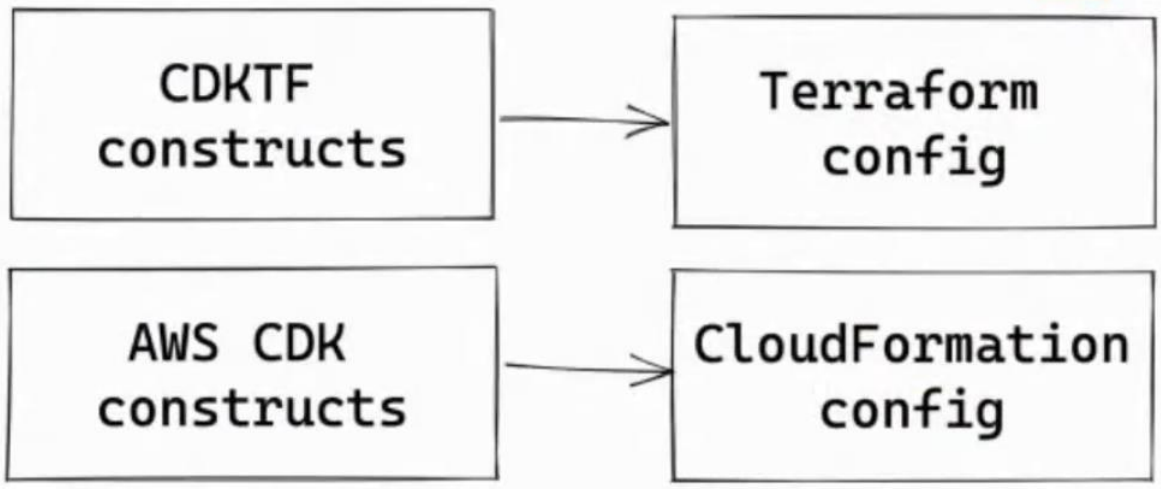
```
// import adapter and AWS CDK constructs
import { aws_lambda } from "aws-cdk-lib";
import { AwsTerraformAdapter } from "@cdktf/aws-cdk";

// instantiate adapter, passing a TerraformStack ("this")
const awsAdapter = new AwsTerraformAdapter(this, "adapter");
// pass the adapter to AWS CDK constructs as scope when using them
const lambdaFn = new aws_lambda.Function(awsAdapter, "lambda", { ... });
```

How does it work?

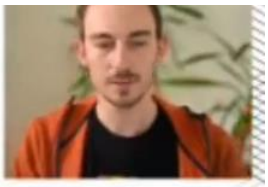
Construct Output

CDKTF vs. AWS CDK



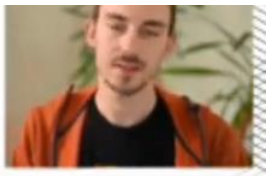


CfnElement



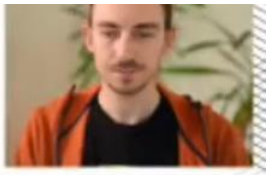
```
105   /**
106    * Returns the CloudFormation 'snippet' for this entity. The snippet will only be merged
107    * at the root level to ensure there are no identity conflicts.
108    *
109    * For example, a Resource class will return something like:
110    * {
111    *   Resources: {
112    *     [this.logicalId]: {
113    *       Type: this.resourceType,
114    *       Properties: this.props,
115    *       Condition: this.condition
116    *     }
117    *   }
118    * }
119    *
120    * @internal
121    */
122   public abstract _toCloudFormation(): object;
123
```

TerraformElement



```
43
44   public toTerraform(): any {
45     return {};
46   }
47
```

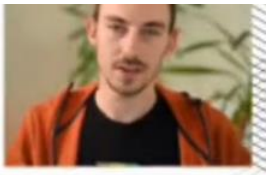
Construct



```
462    */
463   constructor(scope: Construct, id: string) {
464     this.node = new Node(this, scope, id);
465
```

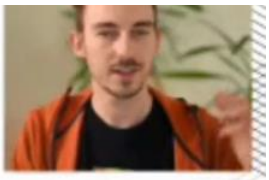
Highlight

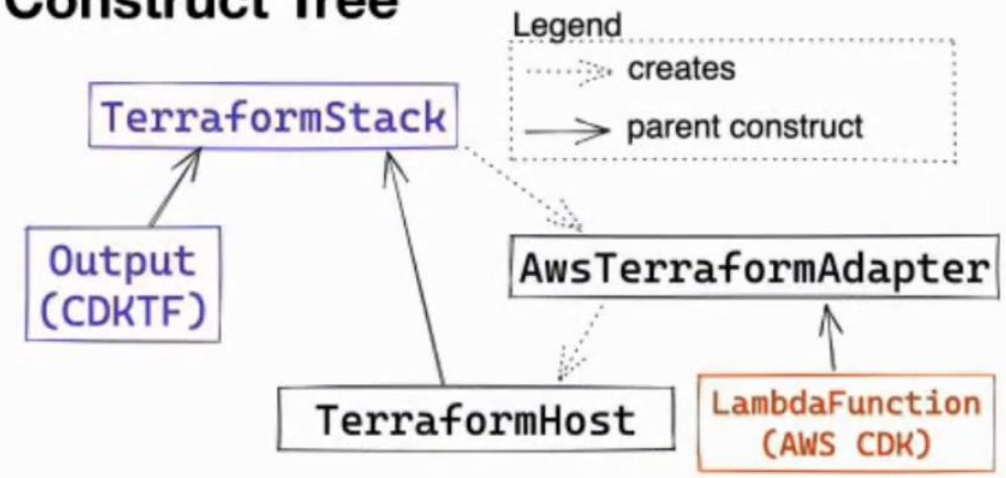
Passing a different scope to AWS CDK constructs



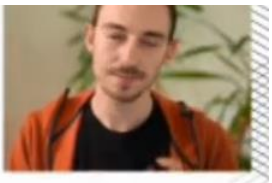
```
// import adapter and AWS CDK constructs
import { aws_lambda } from "aws-cdk-lib";
import { AwsTerraformAdapter } from "@cdktf/aws-cdk";

// instantiate adapter, passing a TerraformStack ("this")
const awsAdapter = new AwsTerraformAdapter(this, "adapter");
// pass the adapter to AWS CDK constructs as scope when using them
const lambdaFn = new aws_lambda.Function(awsAdapter, "lambda", { ... });
```

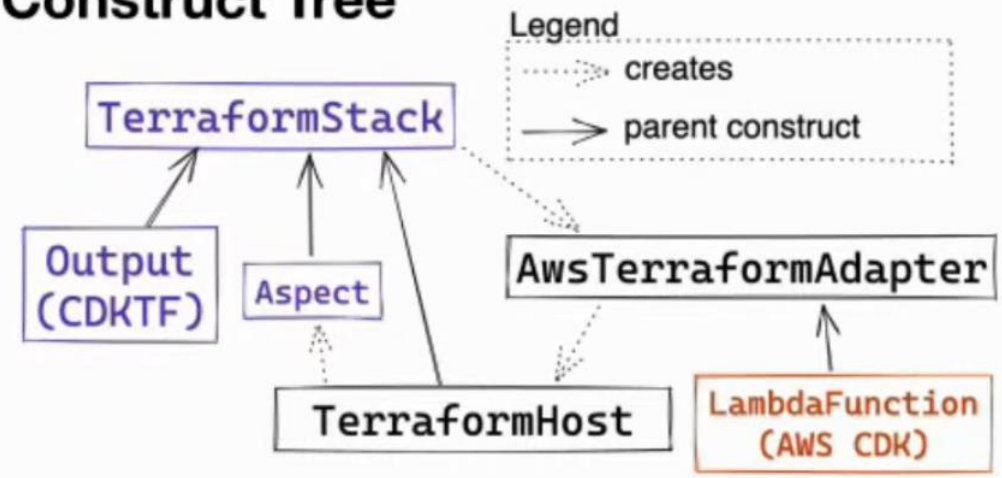
Construct Tree



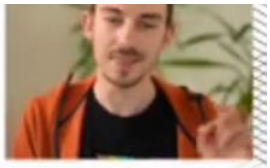


Construct Tree



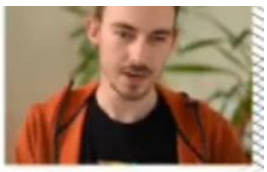


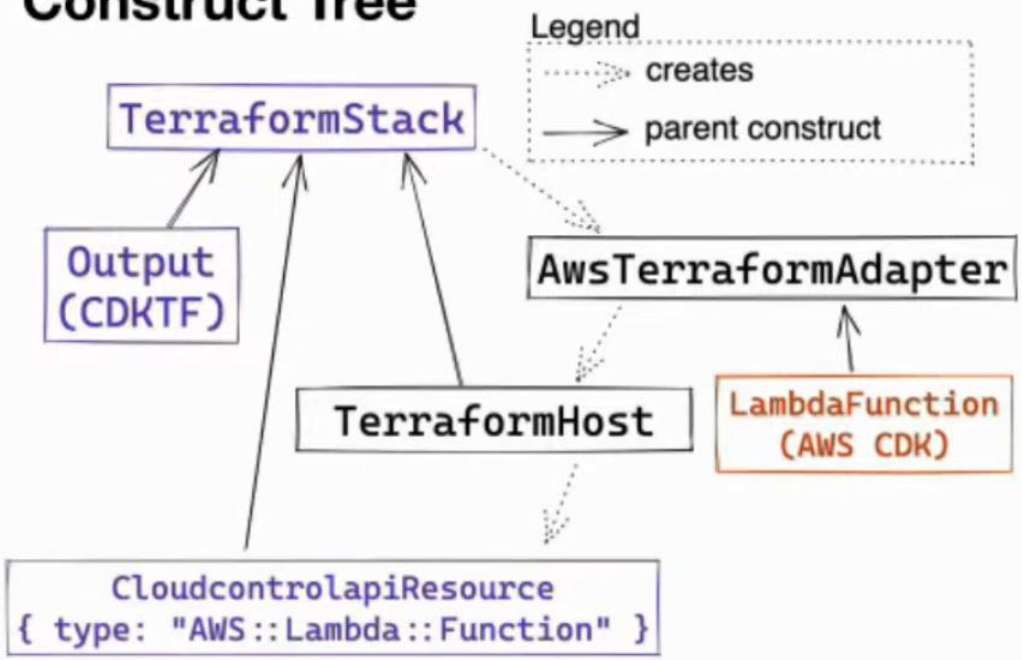
Aspects?



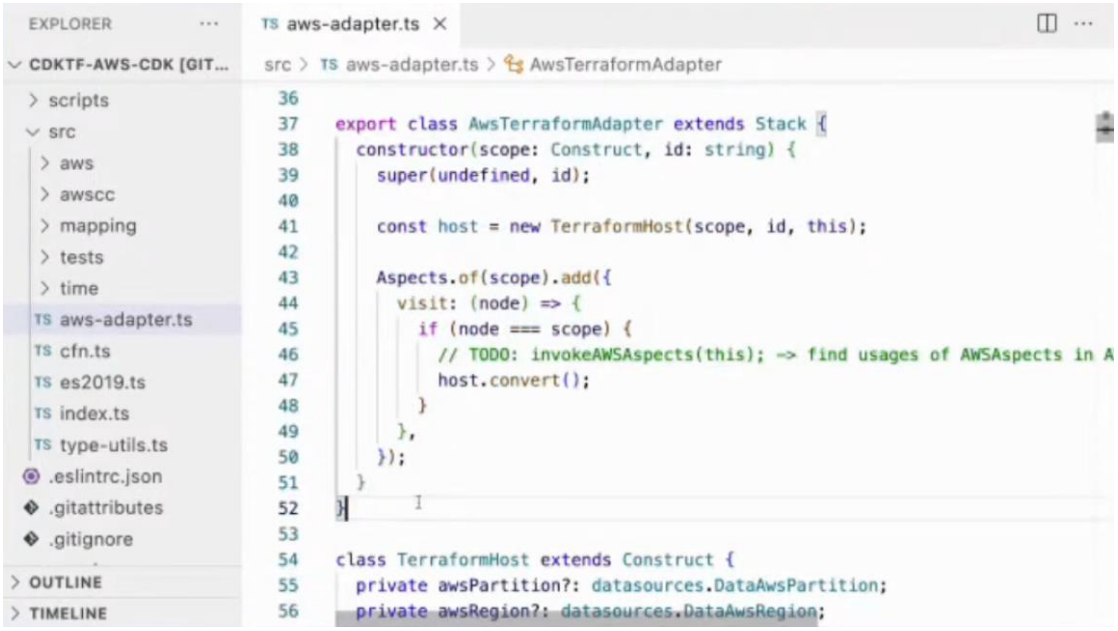
```
export class AwsTerraformAdapter extends Stack {
  constructor(scope: Construct, id: string) {
    super(undefined, id);

    const host = new TerraformHost(scope, id, this);

    Aspects.of(scope).add({
      visit: (node) => {
        if (node === scope) {
          // TODO: invokeAWSAspects(this); -> find usages of AWSAspects in AWS constructs
          host.convert();
        }
      },
    });
  }
}
```

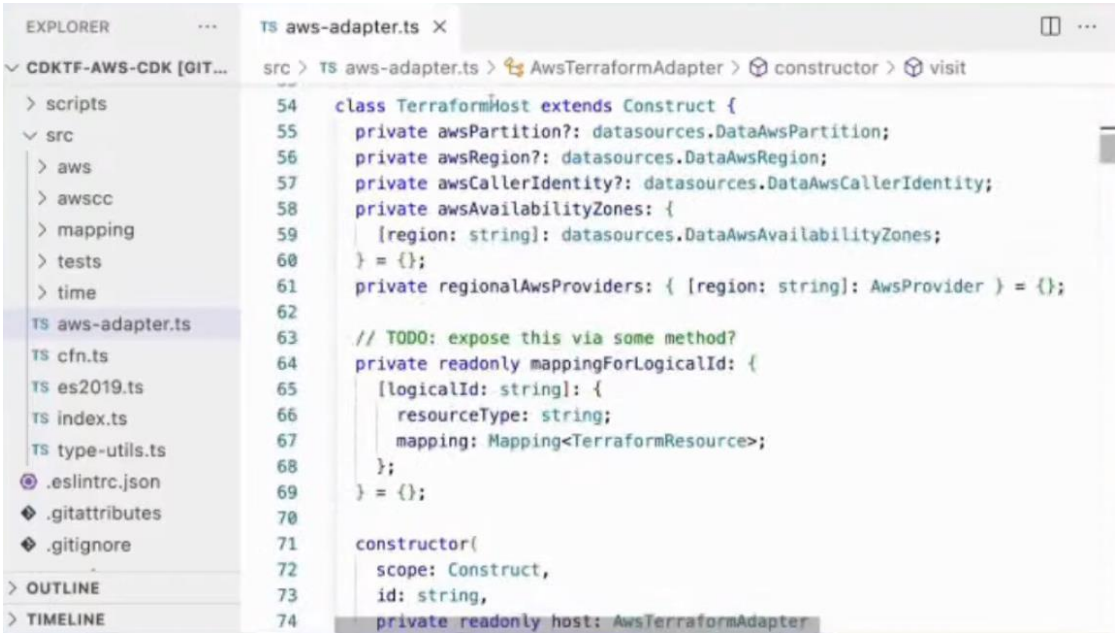
Construct Tree

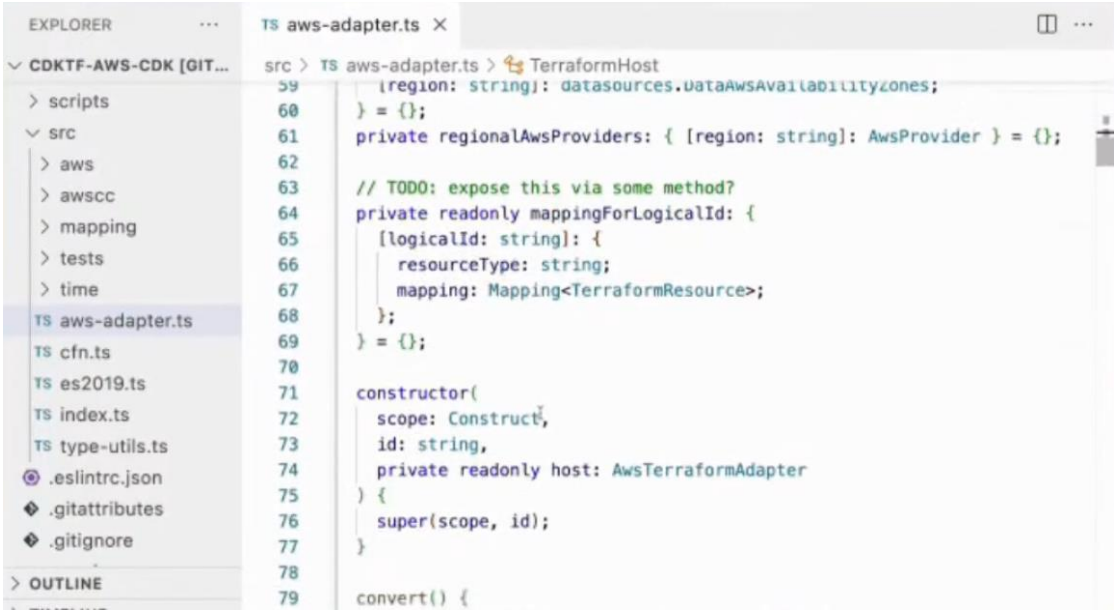


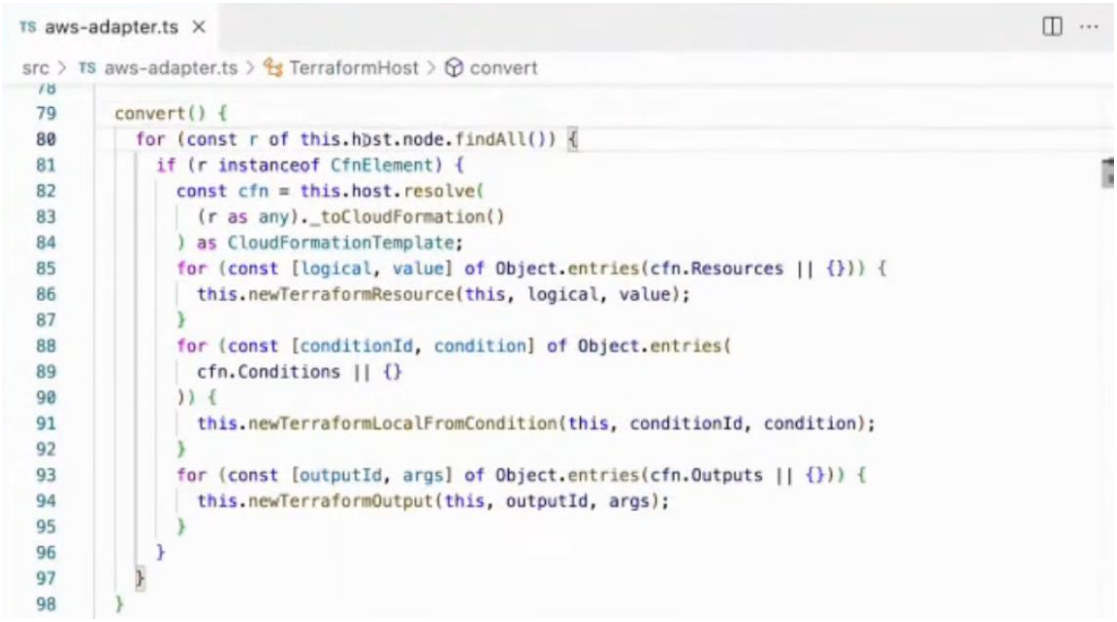


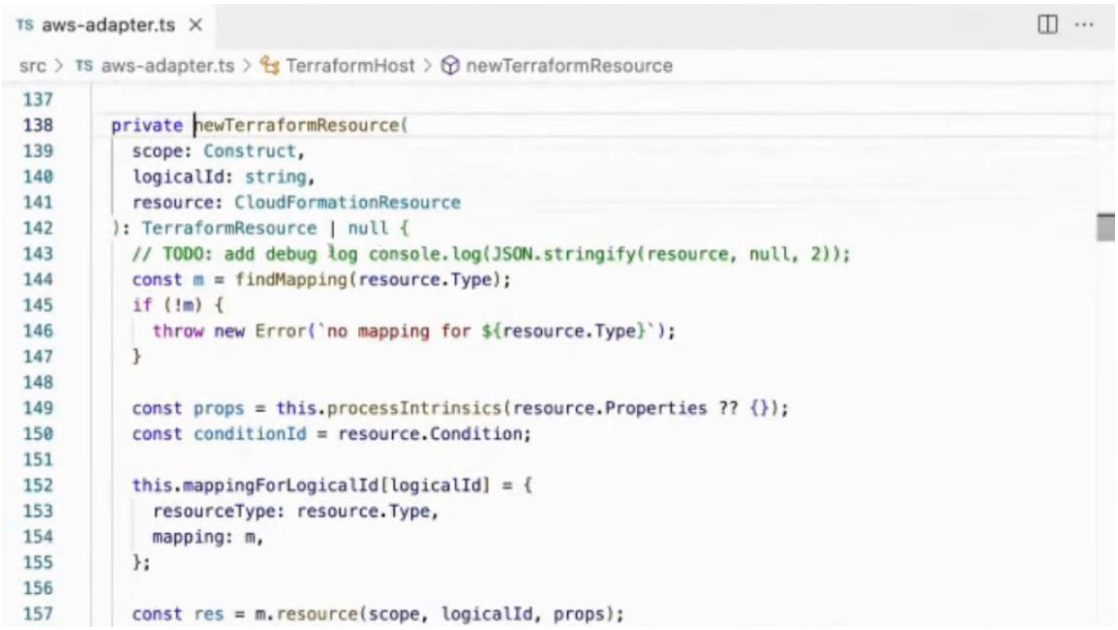
Let's look at the code!

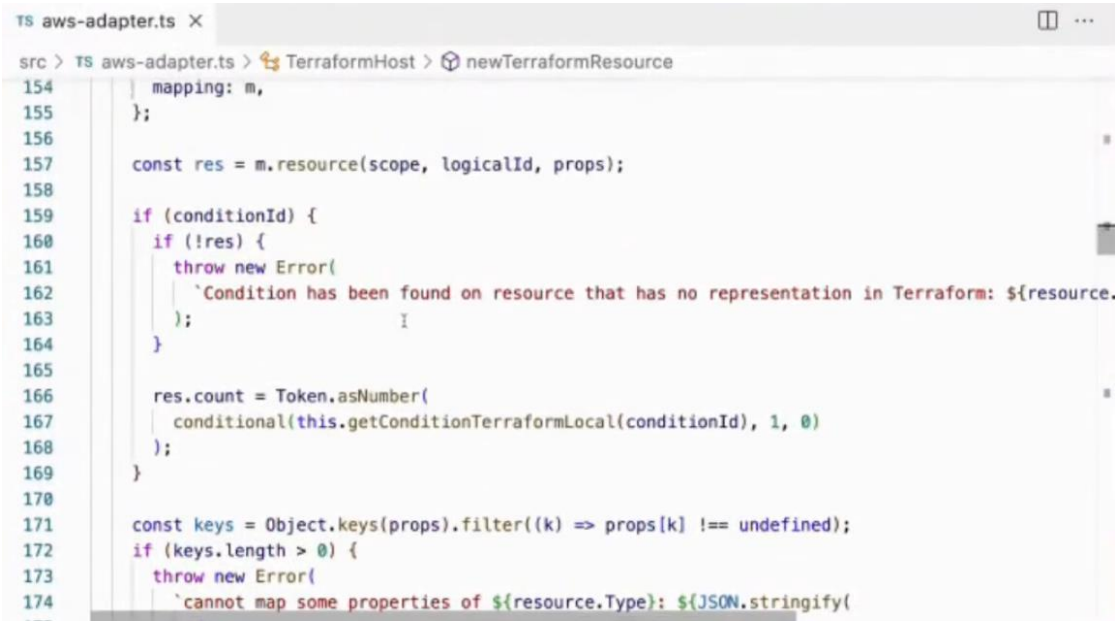

EXPLORER
aws-adapter.ts
CDKTF-AWS-CDK [GIT...
src > aws-adapter.ts > AwsTerraformAdapter
scripts
src
aws
awscc
mapping
tests
time
aws-adapter.ts
cfn.ts
es2019.ts
index.ts
type-utils.ts
.eslintrc.json
.gitattributes
.gitignore
OUTLINE
TIMELINE
export class AwsTerraformAdapter extends Stack {
constructor(scope: Construct, id: string) {
super(undefined, id);
const host = new TerraformHost(scope, id, this);
Aspects.of(scope).add({
visit: (node) => {
if (node === scope) {
// TODO: invokeAWSAspects(this); -> find usages of AWSAspects in A
host.convert();
}
},
});
}
}
class TerraformHost extends Construct {
private awsPartition?: datasources.DataAwsPartition;
private awsRegion?: datasources.DataAwsRegion;


EXPLORER
aws-adapter.ts
CDKTF-AWS-CDK [GIT...
src > aws-adapter.ts > AwsTerraformAdapter > constructor > visit
scripts
src
aws
awscc
mapping
tests
time
aws-adapter.ts
cfn.ts
es2019.ts
index.ts
type-utils.ts
.eslintrc.json
.gitattributes
.gitignore
OUTLINE
TIMELINE
class TerraformHost extends Construct {
private awsPartition?: datasources.DataAwsPartition;
private awsRegion?: datasources.DataAwsRegion;
private awsCallerIdentity?: datasources.DataAwsCallerIdentity;
private awsAvailabilityZones: {
[region: string]: datasources.DataAwsAvailabilityZones;
} = {};
private regionalAwsProviders: { [region: string]: AwsProvider } = {};
// TODO: expose this via some method?
private readonly mappingForLogicalId: {
[logicalId: string]: {
resourceType: string;
mapping: Mapping<TerraformResource>;
};
} = {};
constructor(
scope: Construct,
id: string,
private readonly host: AwsTerraformAdapter


EXPLORER
aws-adapter.ts
CDKTF-AWS-CDK [GIT...
src > aws-adapter.ts > TerraformHost
scripts
src
aws
awscc
mapping
tests
time
aws-adapter.ts
cfn.ts
es2019.ts
index.ts
type-utils.ts
.eslintrc.json
.gitattributes
.gitignore
OUTLINE
} = {};
private regionalAwsProviders: { [region: string]: AwsProvider } = {};
// TODO: expose this via some method?
private readonly mappingForLogicalId: {
[logicalId: string]: {
resourceType: string;
mapping: Mapping<TerraformResource>;
};
} = {};
constructor(
scope: Construct,
id: string,
private readonly host: AwsTerraformAdapter
) {
super(scope, id);
}
convert() {


aws-adapter.ts
src > aws-adapter.ts > TerraformHost > convert
convert() {
for (const r of this.host.node.findAll()) {
if (r instanceof CfnElement) {
const cfn = this.host.resolve(
(r as any)._toCloudFormation()
) as CloudFormationTemplate;
for (const [logical, value] of Object.entries(cfn.Resources || {})) {
this.newTerraformResource(this, logical, value);
}
for (const [conditionId, condition] of Object.entries(
cfn.Conditions || {}
)) {
this.newTerraformLocalFromCondition(this, conditionId, condition);
}
for (const [outputId, args] of Object.entries(cfn.Outputs || {})) {
this.newTerraformOutput(this, outputId, args);
}
}
}
}


aws-adapter.ts
src > aws-adapter.ts > TerraformHost > newTerraformResource
private newTerraformResource(
scope: Construct,
logicalId: string,
resource: CloudFormationResource
): TerraformResource | null {
// TODO: add debug log console.log(JSON.stringify(resource, null, 2));
const m = findMapping(resource.Type);
if (!m) {
throw new Error(`no mapping for ${resource.Type}`);
}
const props = this.processIntrinsics(resource.Properties ?? {});
const conditionId = resource.Condition;
this.mappingForLogicalId[logicalId] = {
resourceType: resource.Type,
mapping: m,
};
const res = m.resource(scope, logicalId, props);

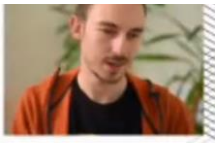

aws-adapter.ts
src > aws-adapter.ts > TerraformHost > newTerraformResource
mapping: m,
};
const res = m.resource(scope, logicalId, props);
if (conditionId) {
if (!res) {
throw new Error(
`Condition has been found on resource that has no representation in Terraform: ${resource.
);
}
res.count = Token.asNumber(
conditional(this.getConditionTerraformLocal(conditionId), 1, 0)
);
}
const keys = Object.keys(props).filter((k) => props[k] !== undefined);
if (keys.length > 0) {
throw new Error(
`cannot map some properties of ${resource.Type}: ${JSON.stringify(

aws-adapter.ts

src > aws-adapter.ts > TerraformHost > newTerraformResource

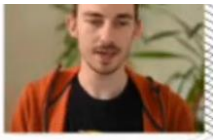
```
167         conditional(this.getConditionTerraformLocal(conditionId), 1, 0)
168       );
169     }
170
171     const keys = Object.keys(props).filter((k) => props[k] !== undefined);
172     if (keys.length > 0) {
173       throw new Error(
174         `cannot map some properties of ${resource.Type}: ${JSON.stringify(
175           props,
176           undefined,
177           2
178         )}`
179       );
180     }
181
182     return res;
183   }
184
185   private newTerraformOutput(scope: Construct, outputId: string, args: any) {
186     return new TerraformOutput(scope, outputId, {
187       value: this.processIntrinsics(args.Value),
```

Recap



1. Second, disjunct construct tree
2. Aspect for conversion step
   - Traverse second construct tree
   - Convert constructs
   - Add results to original construct tree
3. Original construct tree gets synthesized

Links



Get started with the CDK for Terraform

https://cdk.tf

https://cdk.tf/adapter

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

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