Splitting your CDK stack into multiple parts to reduce blast radius

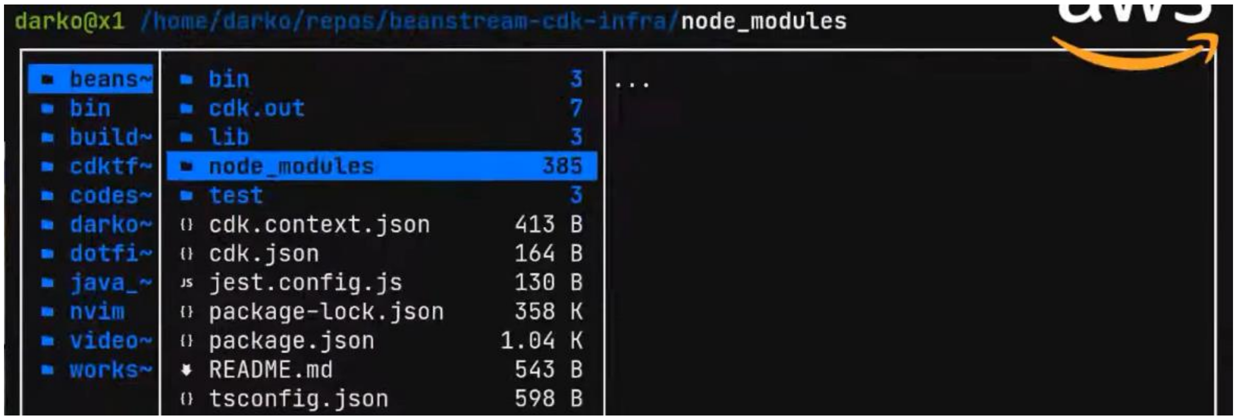
Not Cobus Not Darko
408 subscribers

Subscribe · 21 · Share · Download · Clip

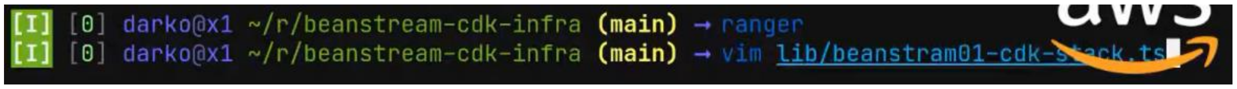
1,221 views Streamed live on Sep 4, 2020 Bean Streaming Sessions
#BeanStreaming #AWSCDK #DevOps

In this episode on 4 Sept 2020, we start to break our monolithic application into multiple smaller part by moving the Amazon Route53 creation to its own stack. The DNS kept changing each time we tore down the stack, so we decided it is time to split things up.

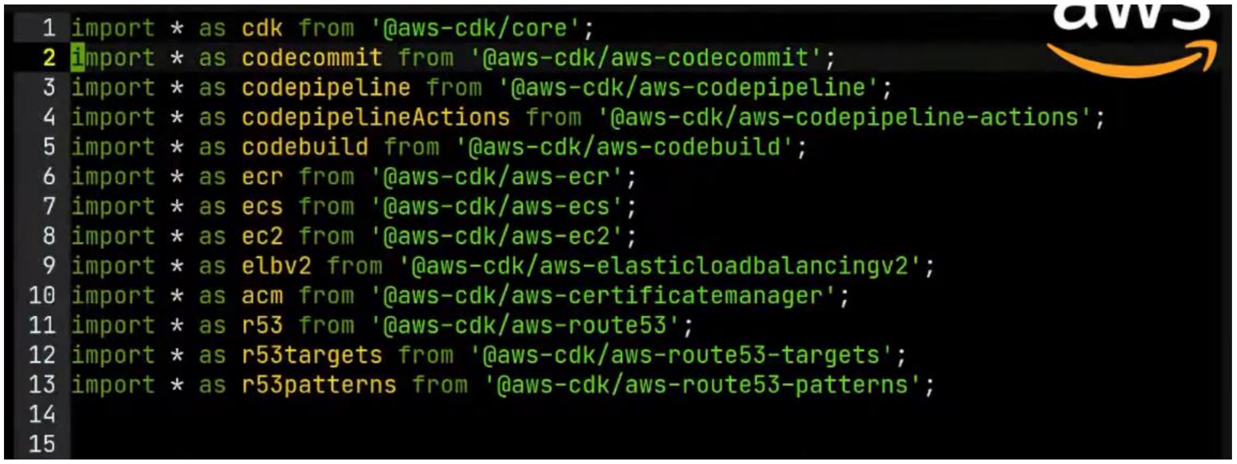
Today: We continue to build an application with AWS CDK.



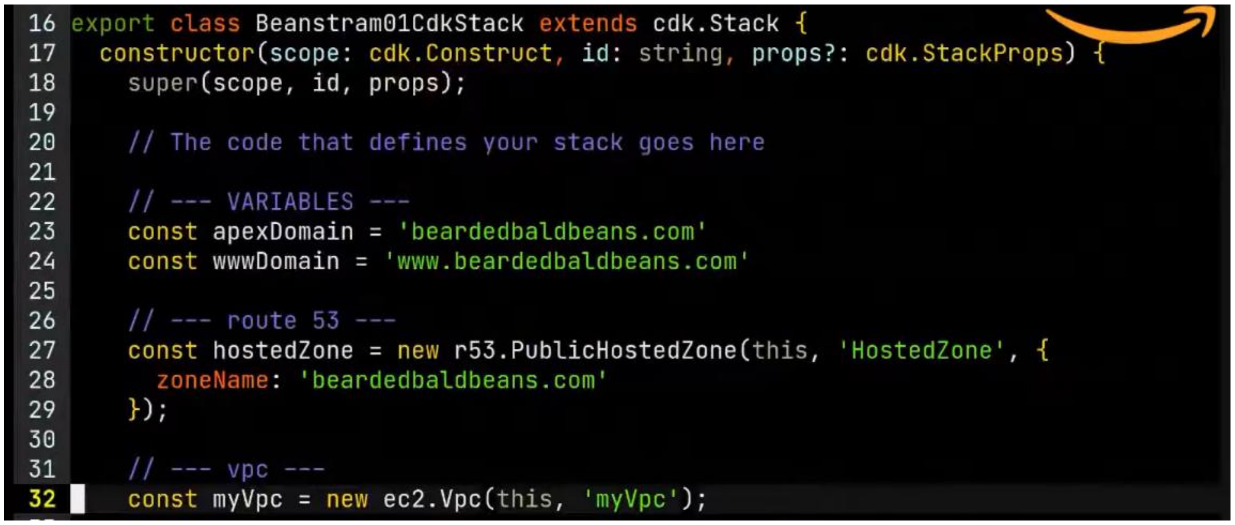
```
darko@x1 /home/darko/repos/beanstream-cdk-infra/node_modules

 beans~   bin                       3    ...
 bin      cdk.out                   7
 build~   lib                       3
 cdktf~   node_modules            385
 codes~   test                      3
 darko~   cdk.context.json      413 B
 dotfi~   cdk.json              164 B
 java_~   jest.config.js        130 B
 nvim     package-lock.json     358 K
 video~   package.json         1.04 K
 works~   README.md             543 B
          tsconfig.json         598 B
```



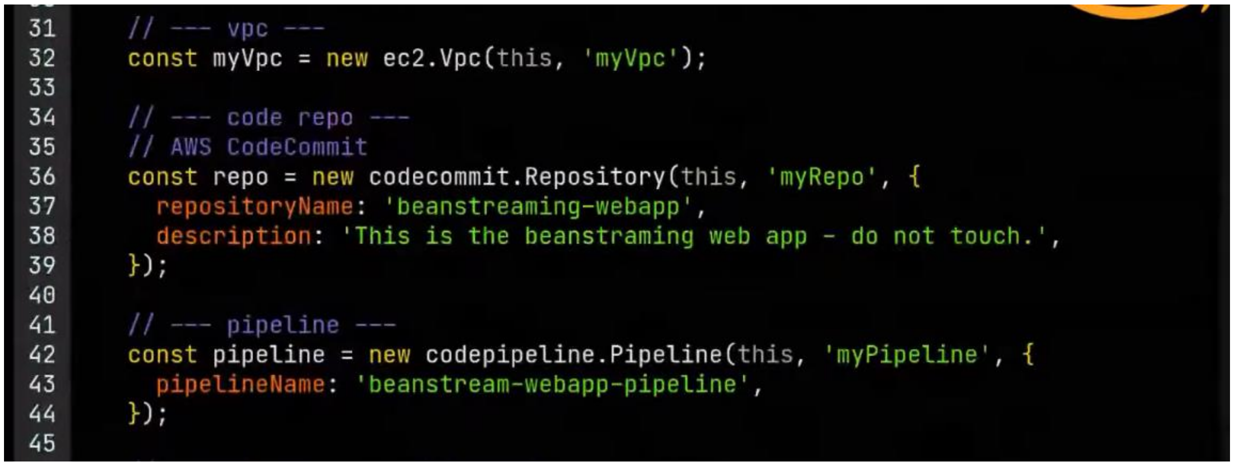
```
[I] [0] darko@x1 ~/r/beanstream-cdk-infra (main) → ranger
[I] [0] darko@x1 ~/r/beanstream-cdk-infra (main) → vim lib/beanstram01-cdk-stack.ts
```



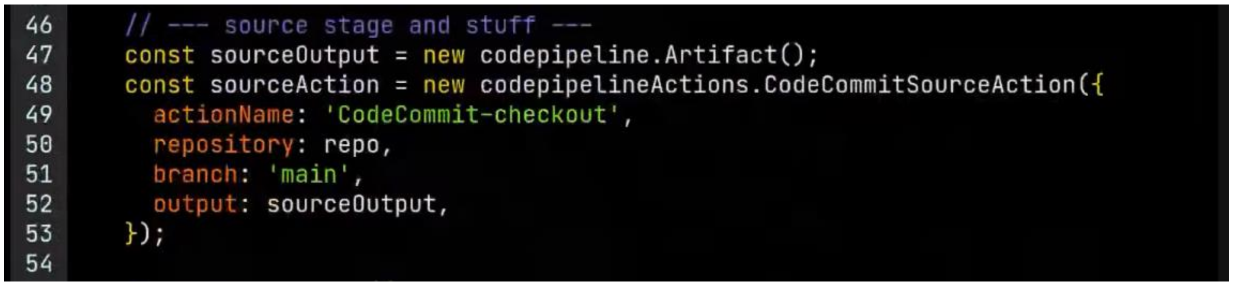
```
import * as cdk from '@aws-cdk/core';
import * as codecommit from '@aws-cdk/aws-codecommit';
import * as codepipeline from '@aws-cdk/aws-codepipeline';
import * as codepipelineActions from '@aws-cdk/aws-codepipeline-actions';
import * as codebuild from '@aws-cdk/aws-codebuild';
import * as ecr from '@aws-cdk/aws-ecr';
import * as ecs from '@aws-cdk/aws-ecs';
import * as ec2 from '@aws-cdk/aws-ec2';
import * as elbv2 from '@aws-cdk/aws-elasticloadbalancingv2';
import * as acm from '@aws-cdk/aws-certificatemanager';
import * as r53 from '@aws-cdk/aws-route53';
import * as r53targets from '@aws-cdk/aws-route53-targets';
import * as r53patterns from '@aws-cdk/aws-route53-patterns';

```



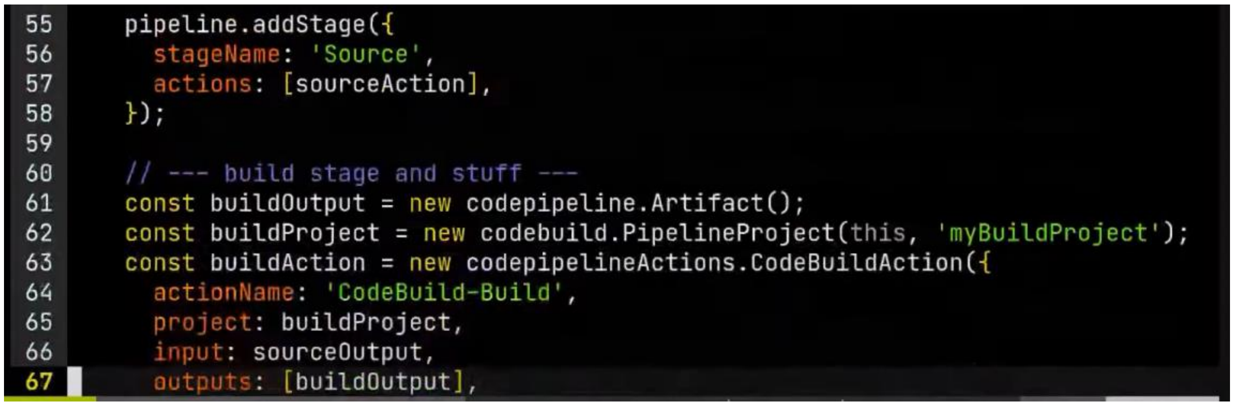
```
export class Beanstram01CdkStack extends cdk.Stack {
  constructor(scope: cdk.Construct, id: string, props?: cdk.StackProps) {
    super(scope, id, props);

    // The code that defines your stack goes here

    // --- VARIABLES ---
    const apexDomain = 'beardedbaldbeans.com'
    const wwwDomain = 'www.beardedbaldbeans.com'

    // --- route 53 ---
    const hostedZone = new r53.PublicHostedZone(this, 'HostedZone', {
      zoneName: 'beardedbaldbeans.com'
    });

    // --- vpc ---
    const myVpc = new ec2.Vpc(this, 'myVpc');
```



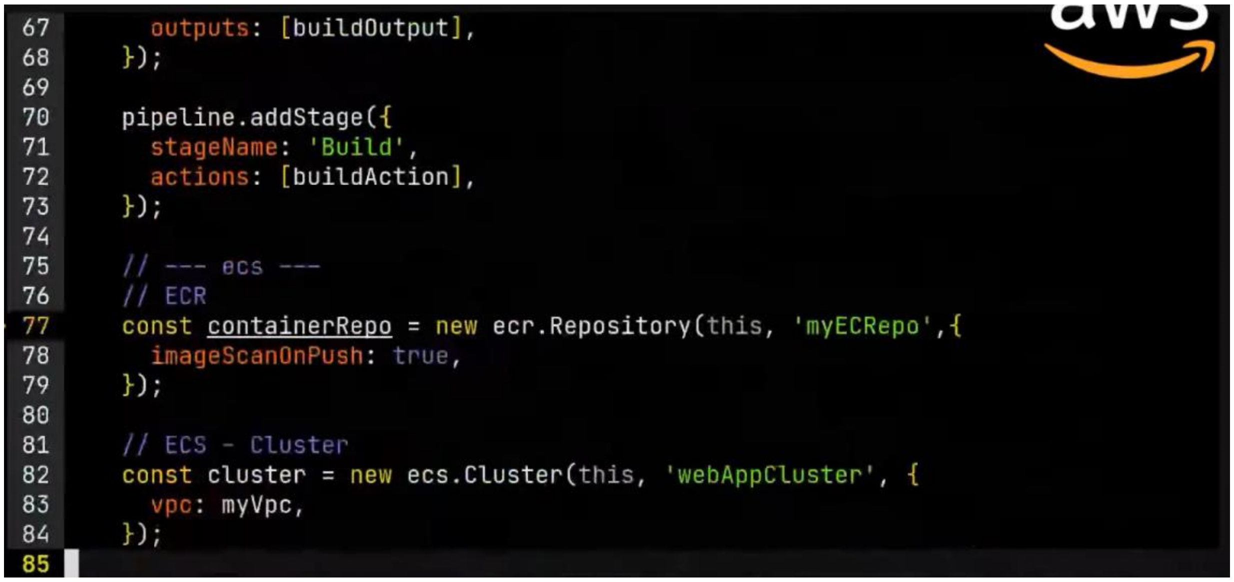
```typescript
    // --- vpc ---
    const myVpc = new ec2.Vpc(this, 'myVpc');

    // --- code repo ---
    // AWS CodeCommit
    const repo = new codecommit.Repository(this, 'myRepo', {
      repositoryName: 'beanstreaming-webapp',
      description: 'This is the beanstraming web app - do not touch.',
    });

    // --- pipeline ---
    const pipeline = new codepipeline.Pipeline(this, 'myPipeline', {
      pipelineName: 'beanstream-webapp-pipeline',
    });

```



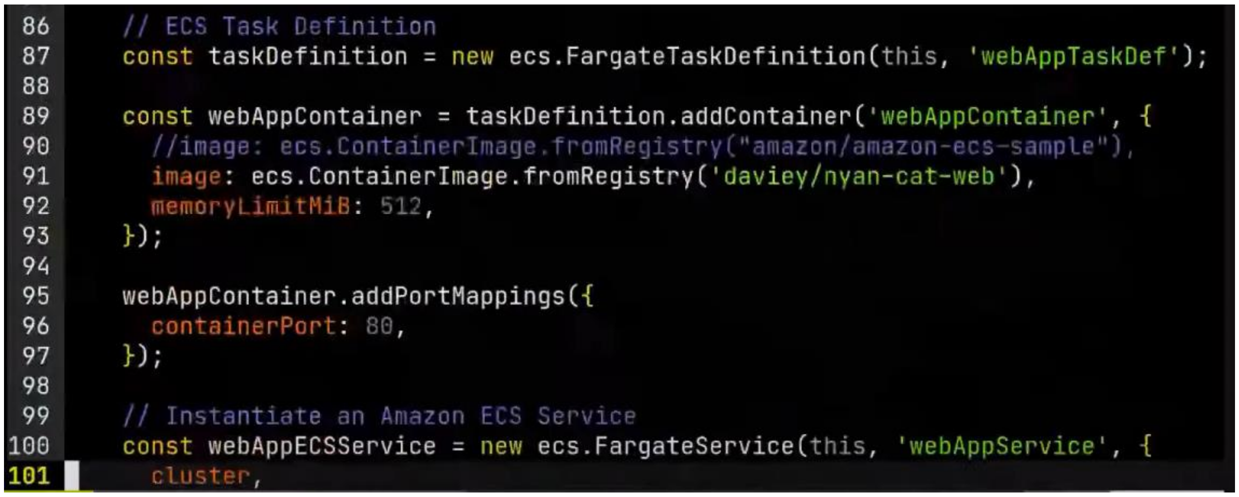
```typescript
    // --- source stage and stuff ---
    const sourceOutput = new codepipeline.Artifact();
    const sourceAction = new codepipelineActions.CodeCommitSourceAction({
      actionName: 'CodeCommit-checkout',
      repository: repo,
      branch: 'main',
      output: sourceOutput,
    });

```



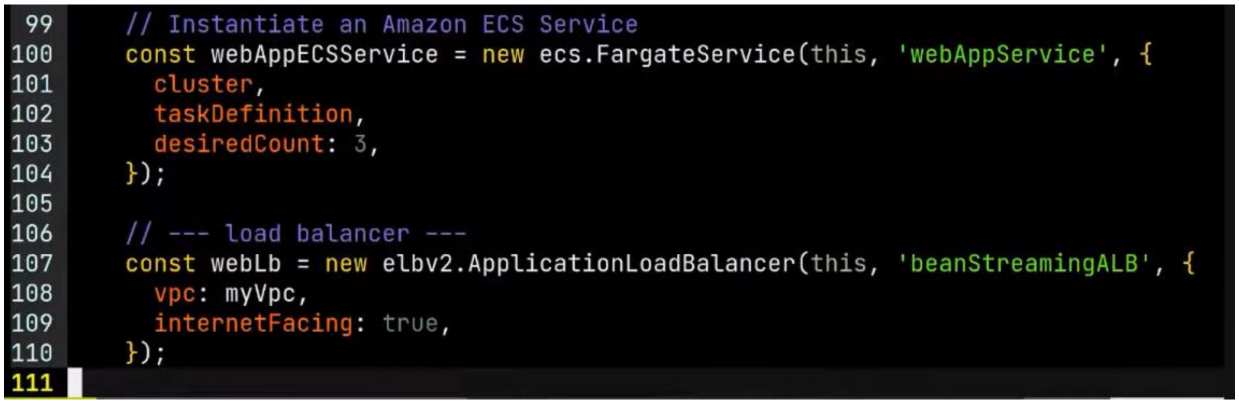
```typescript
    pipeline.addStage({
      stageName: 'Source',
      actions: [sourceAction],
    });

    // --- build stage and stuff ---
    const buildOutput = new codepipeline.Artifact();
    const buildProject = new codebuild.PipelineProject(this, 'myBuildProject');
    const buildAction = new codepipelineActions.CodeBuildAction({
      actionName: 'CodeBuild-Build',
      project: buildProject,
      input: sourceOutput,
      outputs: [buildOutput],
```



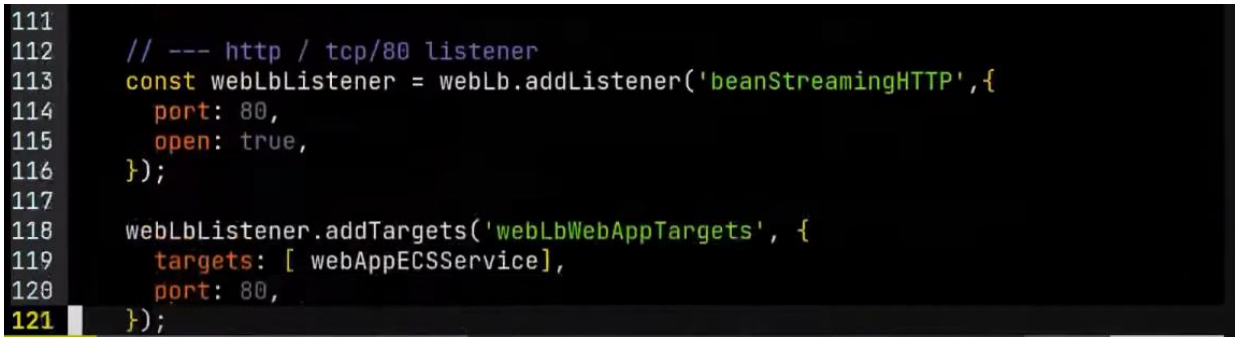
```typescript
      outputs: [buildOutput],
    });

    pipeline.addStage({
      stageName: 'Build',
      actions: [buildAction],
    });

    // --- ecs ---
    // ECR
    const containerRepo = new ecr.Repository(this, 'myECRepo',{
      imageScanOnPush: true,
    });

    // ECS - Cluster
    const cluster = new ecs.Cluster(this, 'webAppCluster', {
      vpc: myVpc,
    });

```



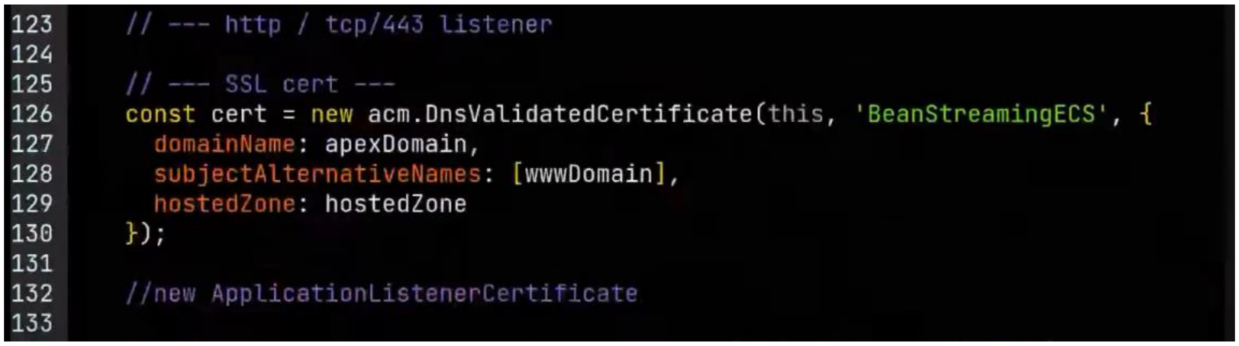
```
    // ECS Task Definition
    const taskDefinition = new ecs.FargateTaskDefinition(this, 'webAppTaskDef');

    const webAppContainer = taskDefinition.addContainer('webAppContainer', {
      //image: ecs.ContainerImage.fromRegistry("amazon/amazon-ecs-sample"),
      image: ecs.ContainerImage.fromRegistry('daviey/nyan-cat-web'),
      memoryLimitMiB: 512,
    });

    webAppContainer.addPortMappings({
      containerPort: 80,
    });

    // Instantiate an Amazon ECS Service
    const webAppECSService = new ecs.FargateService(this, 'webAppService', {
      cluster,
```



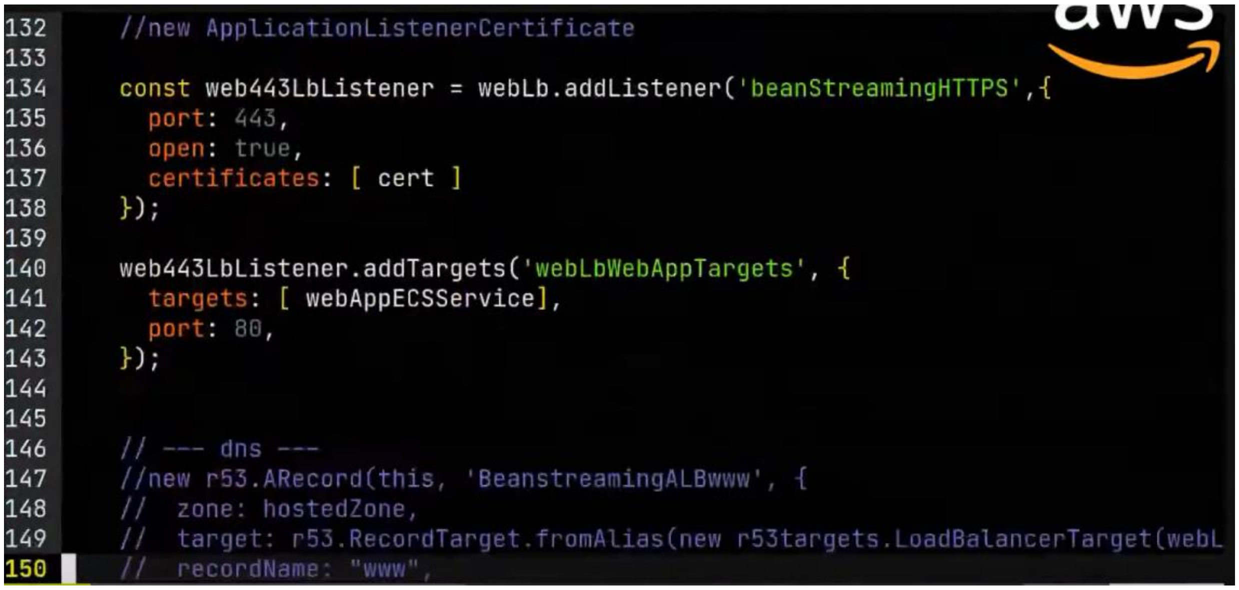
```
    // Instantiate an Amazon ECS Service
    const webAppECSService = new ecs.FargateService(this, 'webAppService', {
      cluster,
      taskDefinition,
      desiredCount: 3,
    });

    // --- load balancer ---
    const webLb = new elbv2.ApplicationLoadBalancer(this, 'beanStreamingALB', {
      vpc: myVpc,
      internetFacing: true,
    });

```



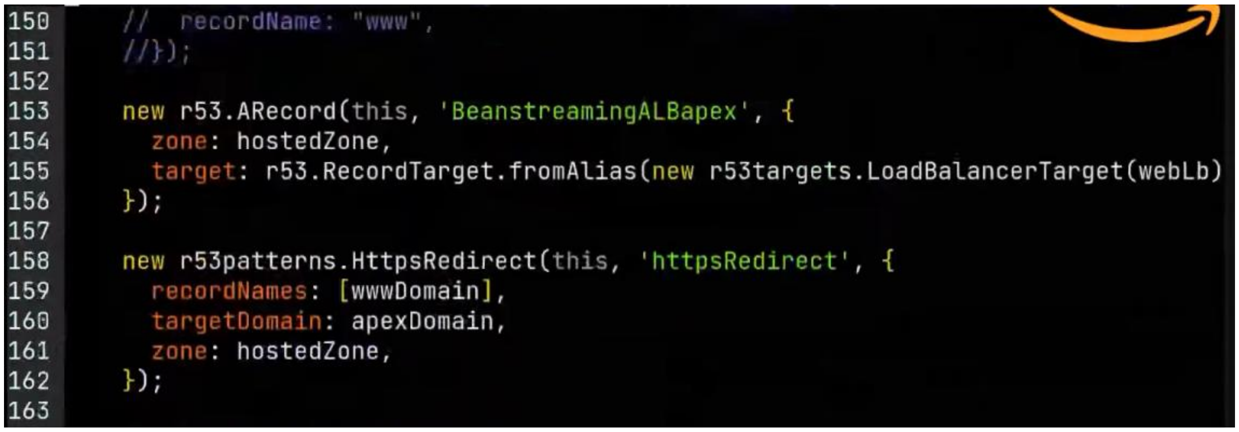
```

    // --- http / tcp/80 listener
    const webLbListener = webLb.addListener('beanStreamingHTTP',{
      port: 80,
      open: true,
    });

    webLbListener.addTargets('webLbWebAppTargets', {
      targets: [ webAppECSService],
      port: 80,
    });
```



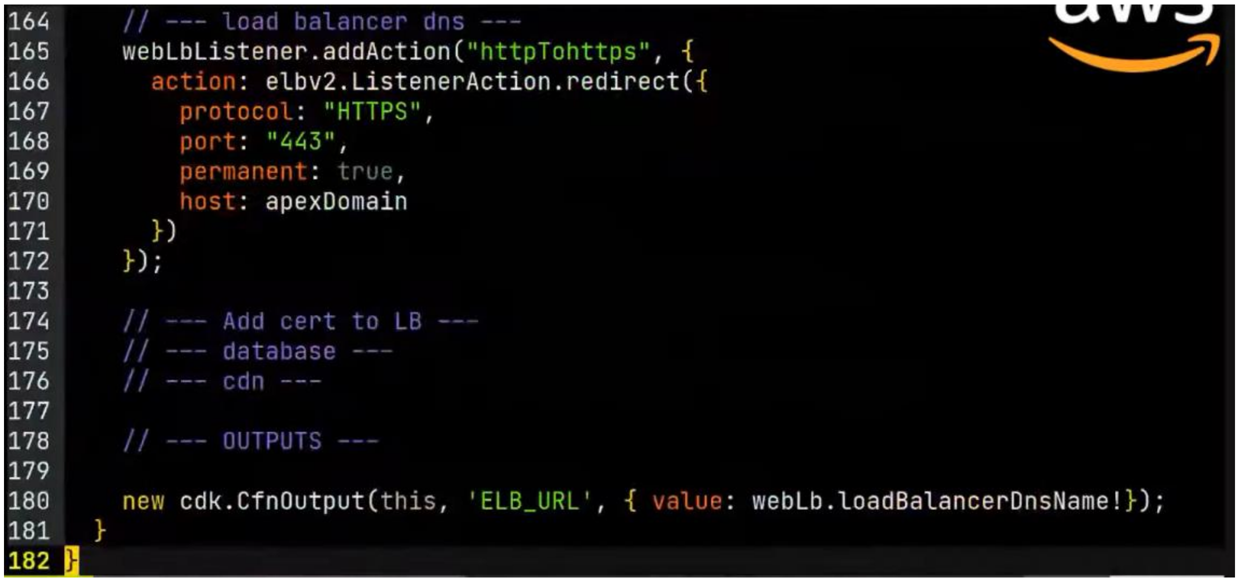
```
    // --- http / tcp/443 listener

    // --- SSL cert ---
    const cert = new acm.DnsValidatedCertificate(this, 'BeanStreamingECS', {
      domainName: apexDomain,
      subjectAlternativeNames: [wwwDomain],
      hostedZone: hostedZone
    });

    //new ApplicationListenerCertificate

```



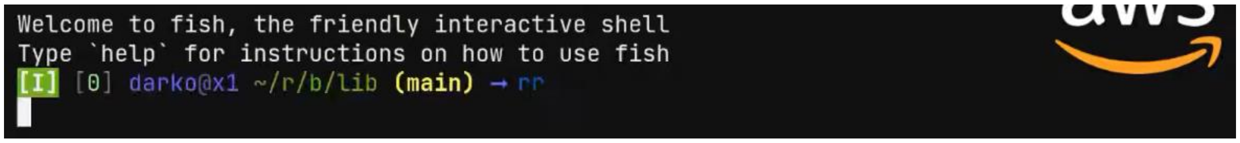
```
    //new ApplicationListenerCertificate

    const web443LbListener = webLb.addListener('beanStreamingHTTPS',{
      port: 443,
      open: true,
      certificates: [ cert ]
    });

    web443LbListener.addTargets('webLbWebAppTargets', {
      targets: [ webAppECSService],
      port: 80,
    });


    // --- dns ---
    //new r53.ARecord(this, 'BeanstreamingALBwww', {
    //  zone: hostedZone,
    //  target: r53.RecordTarget.fromAlias(new r53targets.LoadBalancerTarget(webL
    //  recordName: "www",
```



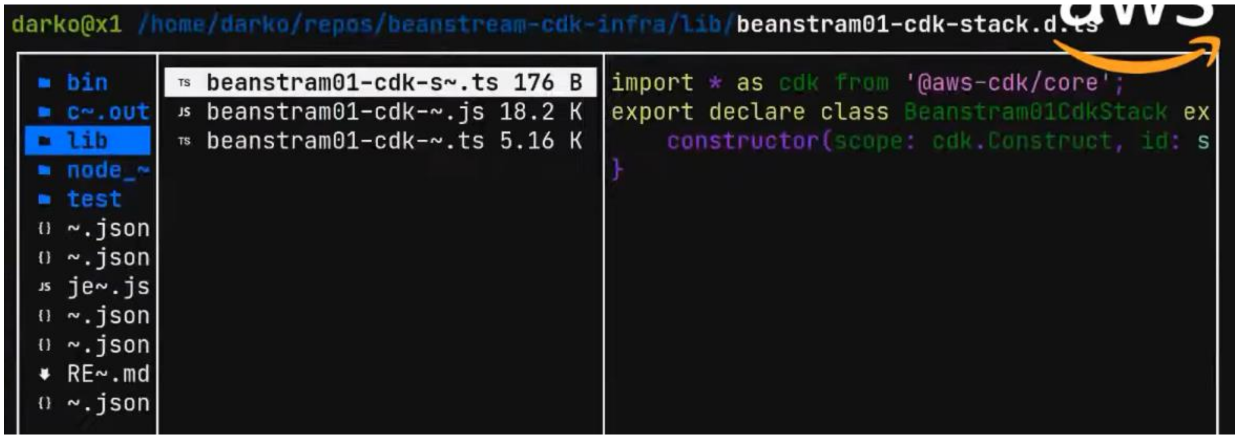
```
    //  recordName: "www",
    //});

    new r53.ARecord(this, 'BeanstreamingALBapex', {
      zone: hostedZone,
      target: r53.RecordTarget.fromAlias(new r53targets.LoadBalancerTarget(webLb)
    });

    new r53patterns.HttpsRedirect(this, 'httpsRedirect', {
      recordNames: [wwwDomain],
      targetDomain: apexDomain,
      zone: hostedZone,
    });

```



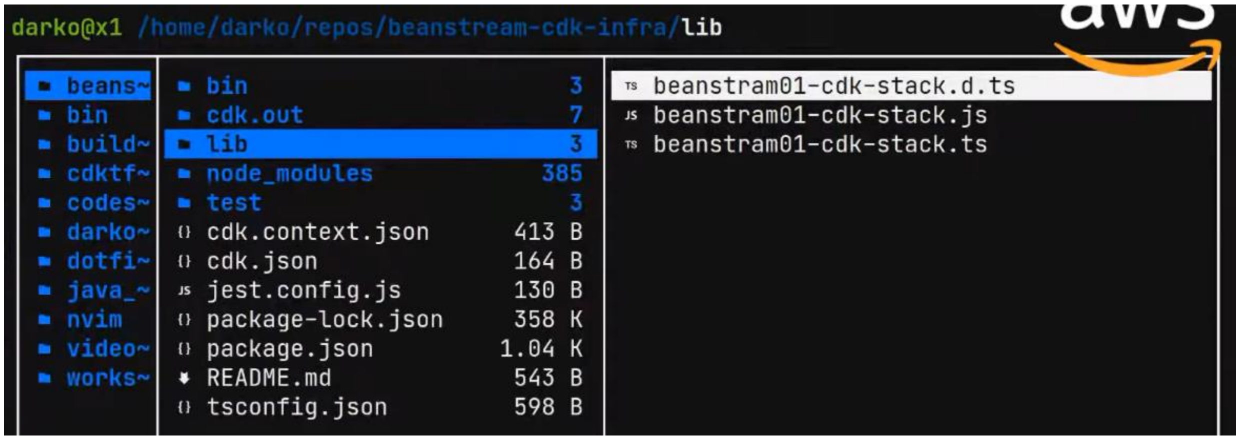
```
    // --- load balancer dns ---
    webLbListener.addAction("httpTohttps", {
      action: elbv2.ListenerAction.redirect({
        protocol: "HTTPS",
        port: "443",
        permanent: true,
        host: apexDomain
      })
    });

    // --- Add cert to LB ---
    // --- database ---
    // --- cdn ---

    // --- OUTPUTS ---

    new cdk.CfnOutput(this, 'ELB_URL', { value: webLb.loadBalancerDnsName!});
  }
}
```



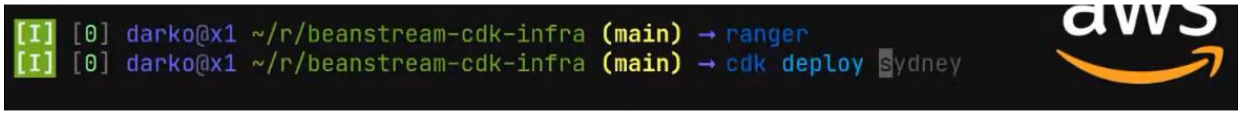
```
Welcome to fish, the friendly interactive shell
Type `help` for instructions on how to use fish
[I] [0] darko@x1 ~/r/b/lib (main) → rr
```



```
darko@x1 /home/darko/repos/beanstream-cdk-infra/lib/beanstram01-cdk-stack.d.ts

bin        beanstram01-cdk-s~.ts 176 B   import * as cdk from '@aws-cdk/core';
c~.out     beanstram01-cdk-~.js 18.2 K   export declare class Beanstram01CdkStack ex
lib        beanstram01-cdk-~.ts 5.16 K       constructor(scope: cdk.Construct, id: s
node_~                                   }
test
~.json
~.json
je~.js
~.json
~.json
RE~.md
~.json
```

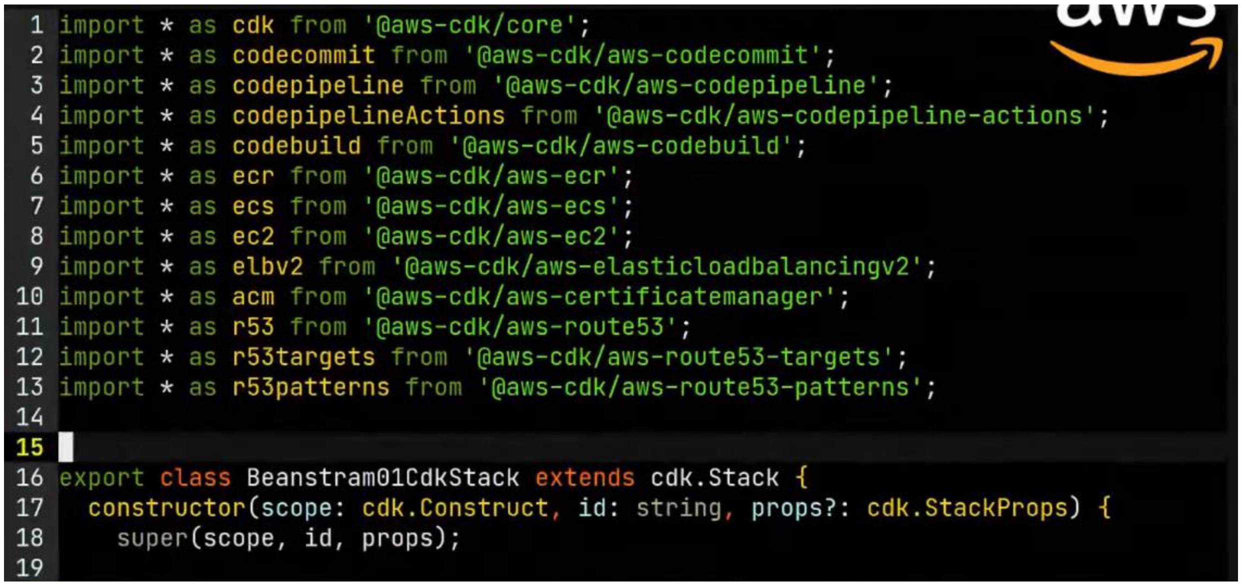


```
darko@x1 /home/darko/repos/beanstream-cdk-infra/lib

beans~    bin                      3   beanstram01-cdk-stack.d.ts
bin       cdk.out                  7   beanstram01-cdk-stack.js
build~    lib                      3   beanstram01-cdk-stack.ts
cdktf~    node_modules           385
codes~    test                     3
darko~    cdk.context.json     413 B
dotfi~    cdk.json             164 B
java_~    jest.config.js       130 B
nvim      package-lock.json    358 K
video~    package.json        1.04 K
works~    README.md            543 B
          tsconfig.json        598 B
```



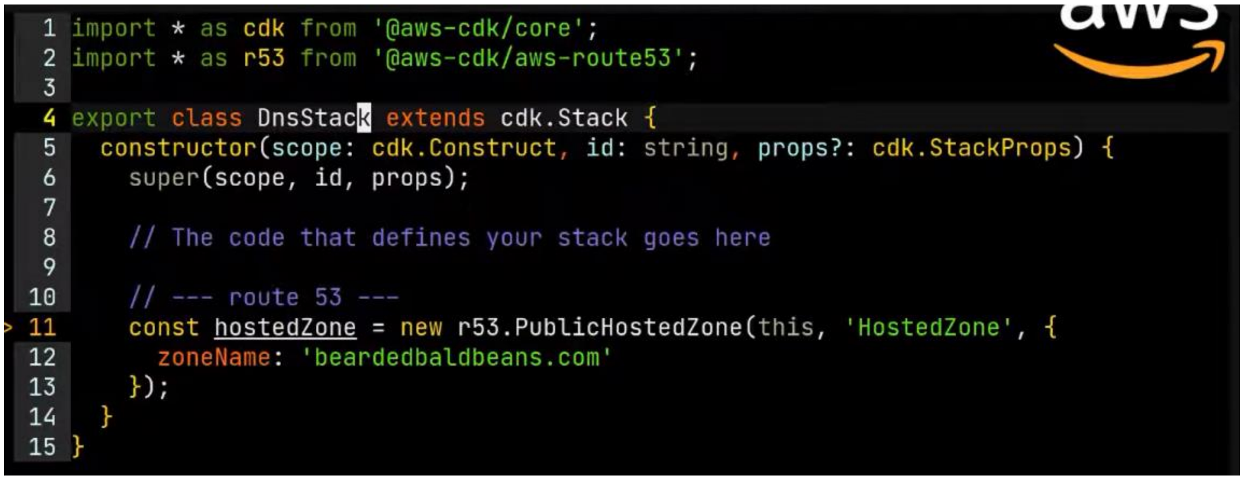
```
[I] [0] darko@x1 ~/r/beanstream-cdk-infra (main) → ranger
[I] [0] darko@x1 ~/r/beanstream-cdk-infra (main) → cdk deploy sydney
```

Doing a **cdk deploy** command will be deployed thru a single CF stack, we can split the DNS part of the Route53 hosted host into its own part in a different stack. To create a 2nd stack, create a 2nd file inside the lib directory above and move the relevant contents into it.



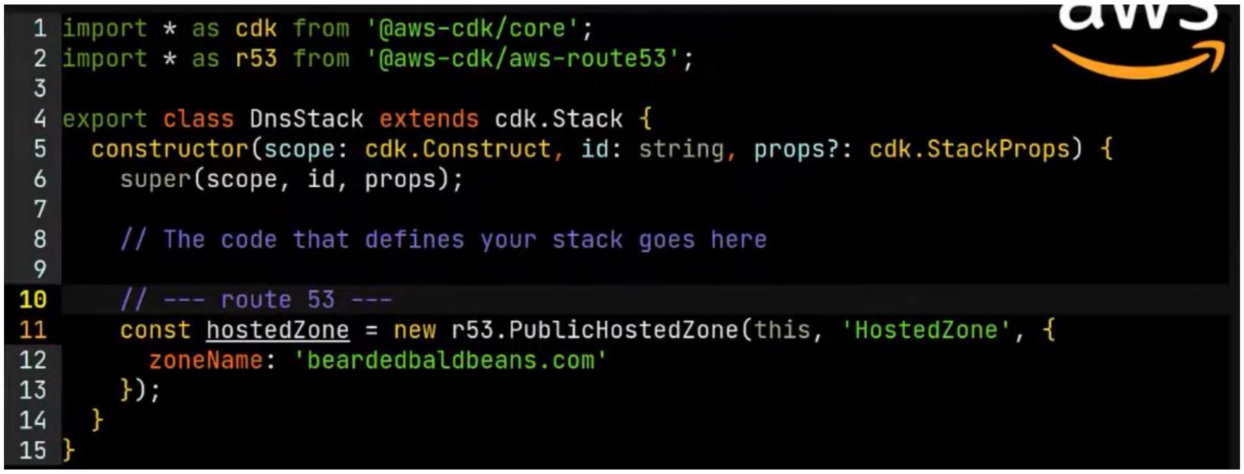
```
import * as cdk from '@aws-cdk/core';
import * as codecommit from '@aws-cdk/aws-codecommit';
import * as codepipeline from '@aws-cdk/aws-codepipeline';
import * as codepipelineActions from '@aws-cdk/aws-codepipeline-actions';
import * as codebuild from '@aws-cdk/aws-codebuild';
import * as ecr from '@aws-cdk/aws-ecr';
import * as ecs from '@aws-cdk/aws-ecs';
import * as ec2 from '@aws-cdk/aws-ec2';
import * as elbv2 from '@aws-cdk/aws-elasticloadbalancingv2';
import * as acm from '@aws-cdk/aws-certificatemanager';
import * as r53 from '@aws-cdk/aws-route53';
import * as r53targets from '@aws-cdk/aws-route53-targets';
import * as r53patterns from '@aws-cdk/aws-route53-patterns';


export class Beanstram01CdkStack extends cdk.Stack {
  constructor(scope: cdk.Construct, id: string, props?: cdk.StackProps) {
    super(scope, id, props);

```

Copy this content into the new file, rename the class name and delete unneeded parts to get the below file

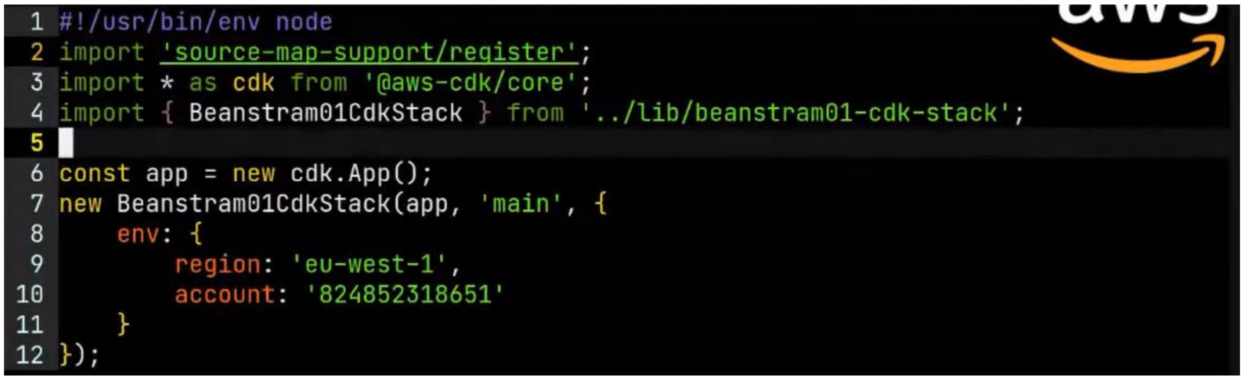


```
import * as cdk from '@aws-cdk/core';
import * as r53 from '@aws-cdk/aws-route53';

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

    // The code that defines your stack goes here

    // --- route 53 ---
    const hostedZone = new r53.PublicHostedZone(this, 'HostedZone', {
      zoneName: 'beardedbaldbeans.com'
    });
  }
}
```

We can share constructs between different stacks to share that resources BUT we cannot share stacks.

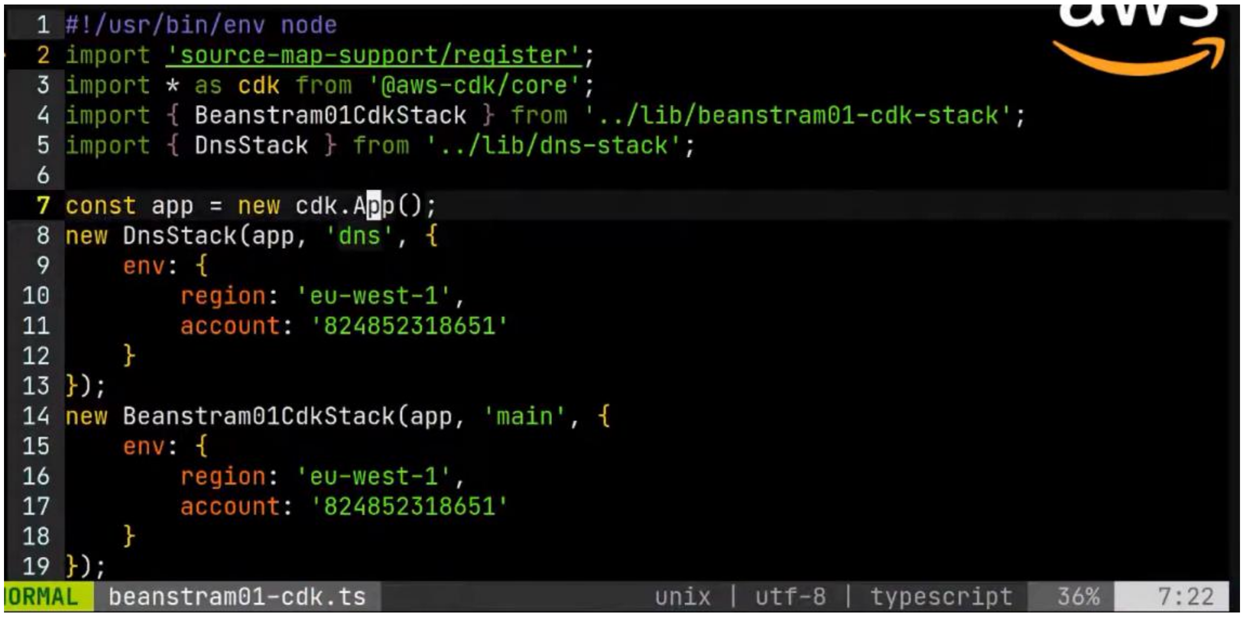


```
import * as cdk from '@aws-cdk/core';
import * as r53 from '@aws-cdk/aws-route53';

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

    // The code that defines your stack goes here

    // --- route 53 ---
    const hostedZone = new r53.PublicHostedZone(this, 'HostedZone', {
      zoneName: 'beardedbaldbeans.com'
    });
  }
}
```



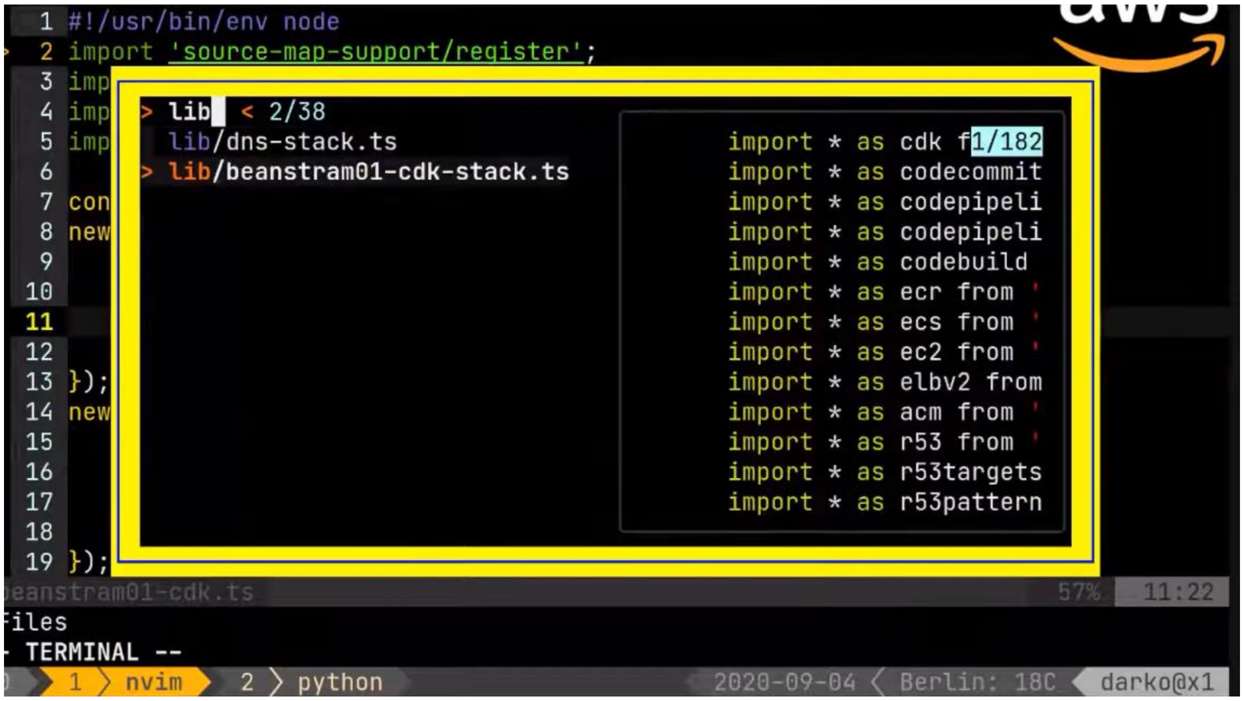
```
#!/usr/bin/env node
import 'source-map-support/register';
import * as cdk from '@aws-cdk/core';
import { Beanstram01CdkStack } from '../lib/beanstram01-cdk-stack';

const app = new cdk.App();
new Beanstram01CdkStack(app, 'main', {
    env: {
        region: 'eu-west-1',
        account: '824852318651'
    }
});
```

This is where we will define our stacks. We run **cdk deploy** command to deploy all our defined stacks here

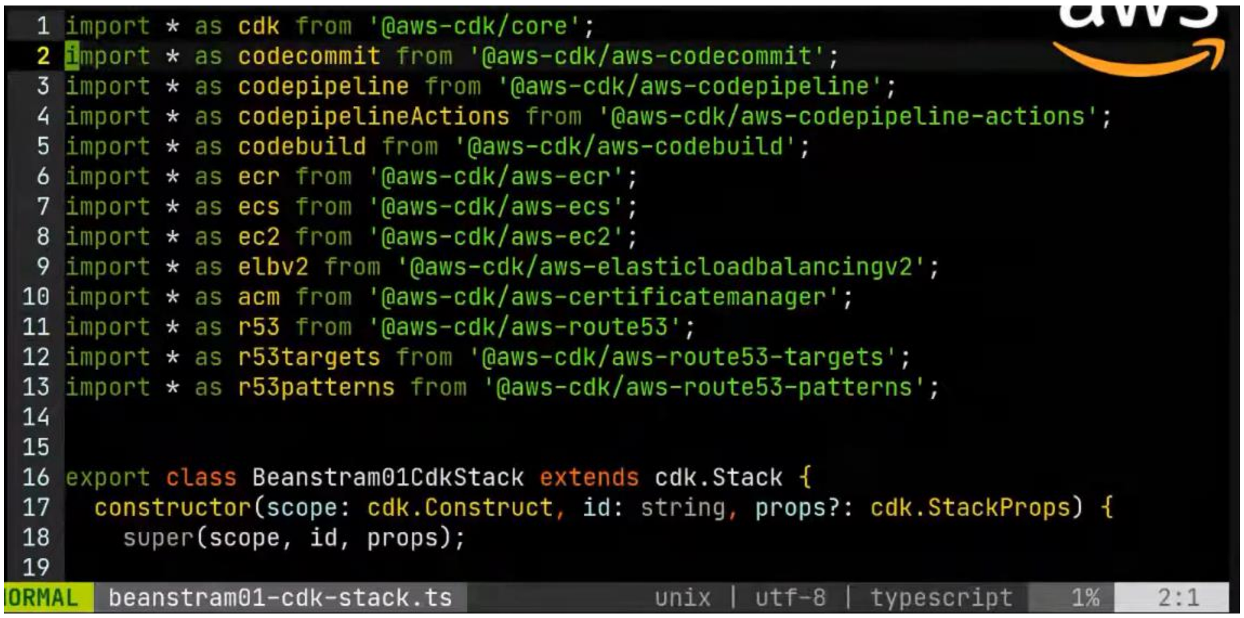


```
#!/usr/bin/env node
import 'source-map-support/register';
import * as cdk from '@aws-cdk/core';
import { Beanstram01CdkStack } from '../lib/beanstram01-cdk-stack';
import { DnsStack } from '../lib/dns-stack';

const app = new cdk.App();
new DnsStack(app, 'dns', {
    env: {
        region: 'eu-west-1',
        account: '824852318651'
    }
});
new Beanstram01CdkStack(app, 'main', {
    env: {
        region: 'eu-west-1',
        account: '824852318651'
    }
});
ORMAL  beanstram01-cdk.ts          unix | utf-8 | typescript  36%  7:22
```

We can now create 2 stacks called **dns** and **main** when we run the **cdk deploy** command.

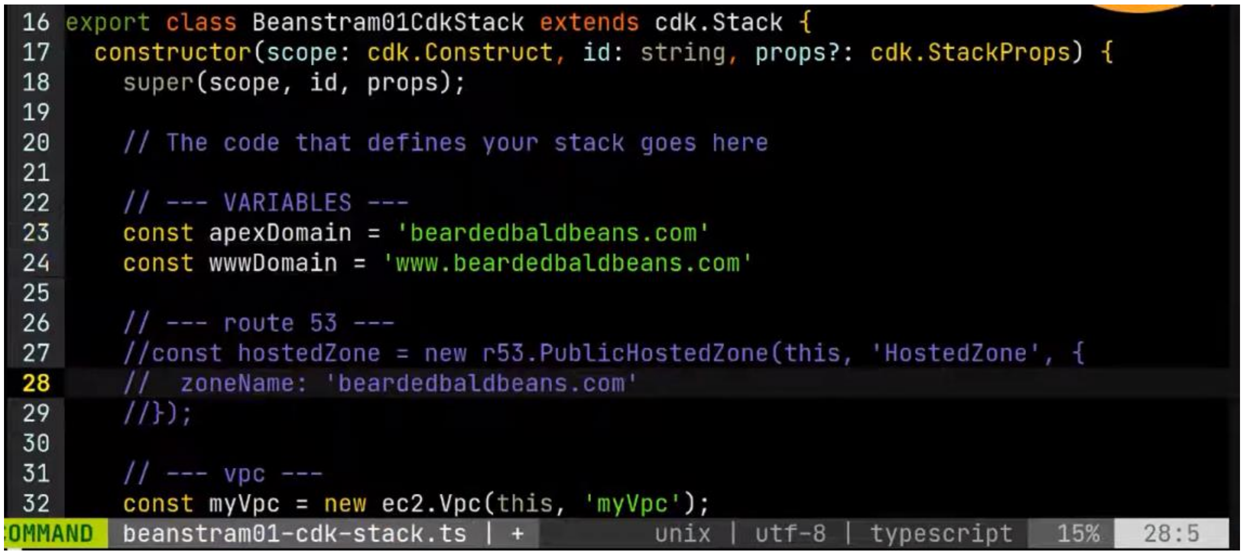


```
> lib  < 2/38
  lib/dns-stack.ts
> lib/beanstram01-cdk-stack.ts

import * as cdk f1/182
import * as codecommit
import * as codepipeli
import * as codepipeli
import * as codebuild
import * as ecr from '
import * as ecs from '
import * as ec2 from '
import * as elbv2 from
import * as acm from '
import * as r53 from '
import * as r53targets
import * as r53pattern
```

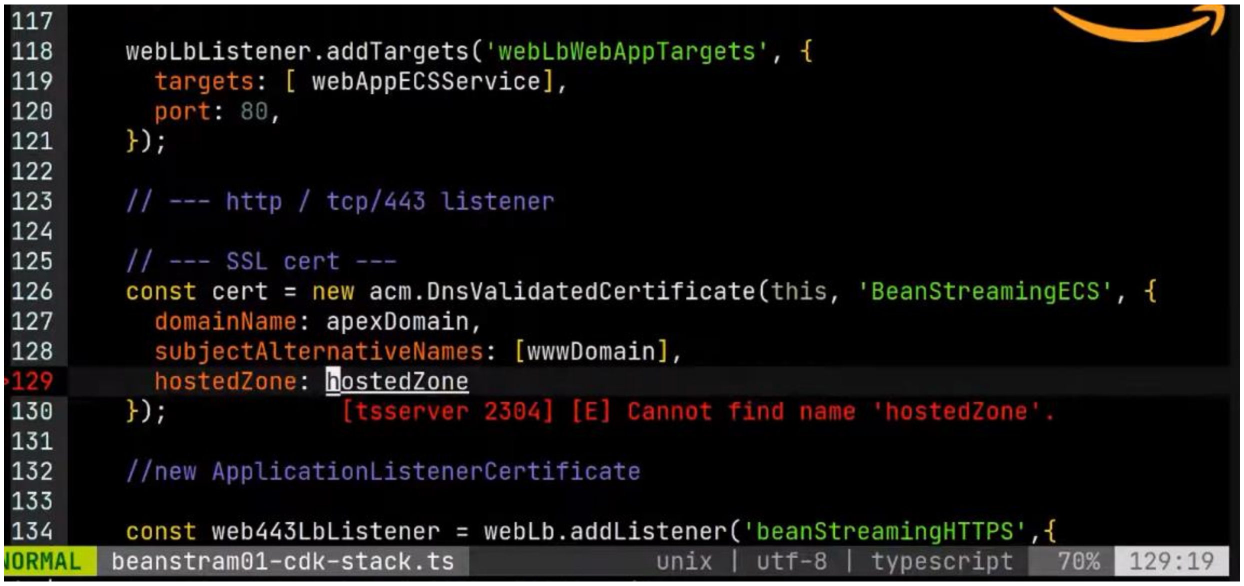


```
import * as cdk from '@aws-cdk/core';
import * as codecommit from '@aws-cdk/aws-codecommit';
import * as codepipeline from '@aws-cdk/aws-codepipeline';
import * as codepipelineActions from '@aws-cdk/aws-codepipeline-actions';
import * as codebuild from '@aws-cdk/aws-codebuild';
import * as ecr from '@aws-cdk/aws-ecr';
import * as ecs from '@aws-cdk/aws-ecs';
import * as ec2 from '@aws-cdk/aws-ec2';
import * as elbv2 from '@aws-cdk/aws-elasticloadbalancingv2';
import * as acm from '@aws-cdk/aws-certificatemanager';
import * as r53 from '@aws-cdk/aws-route53';
import * as r53targets from '@aws-cdk/aws-route53-targets';
import * as r53patterns from '@aws-cdk/aws-route53-patterns';


export class Beanstram01CdkStack extends cdk.Stack {
  constructor(scope: cdk.Construct, id: string, props?: cdk.StackProps) {
    super(scope, id, props);

ORMAL  beanstram01-cdk-stack.ts          unix | utf-8 | typescript   1%   2:1
```

In our **main** stack file above, we need to remove the **dns** part since we are separating it out into its own stack.

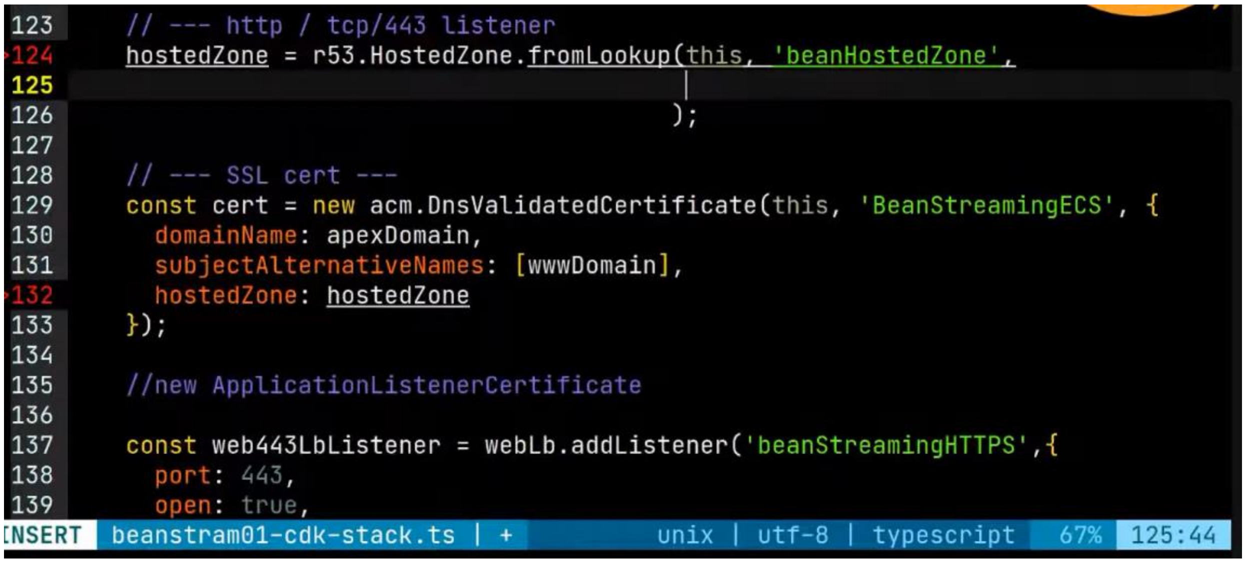


```
export class Beanstram01CdkStack extends cdk.Stack {
  constructor(scope: cdk.Construct, id: string, props?: cdk.StackProps) {
    super(scope, id, props);

    // The code that defines your stack goes here

    // --- VARIABLES ---
    const apexDomain = 'beardedbaldbeans.com'
    const wwwDomain = 'www.beardedbaldbeans.com'

    // --- route 53 ---
    //const hostedZone = new r53.PublicHostedZone(this, 'HostedZone', {
    //  zoneName: 'beardedbaldbeans.com'
    //});

    // --- vpc ---
    const myVpc = new ec2.Vpc(this, 'myVpc');
OMMAND  beanstram01-cdk-stack.ts | +      unix | utf-8 | typescript   15%   28:5
```

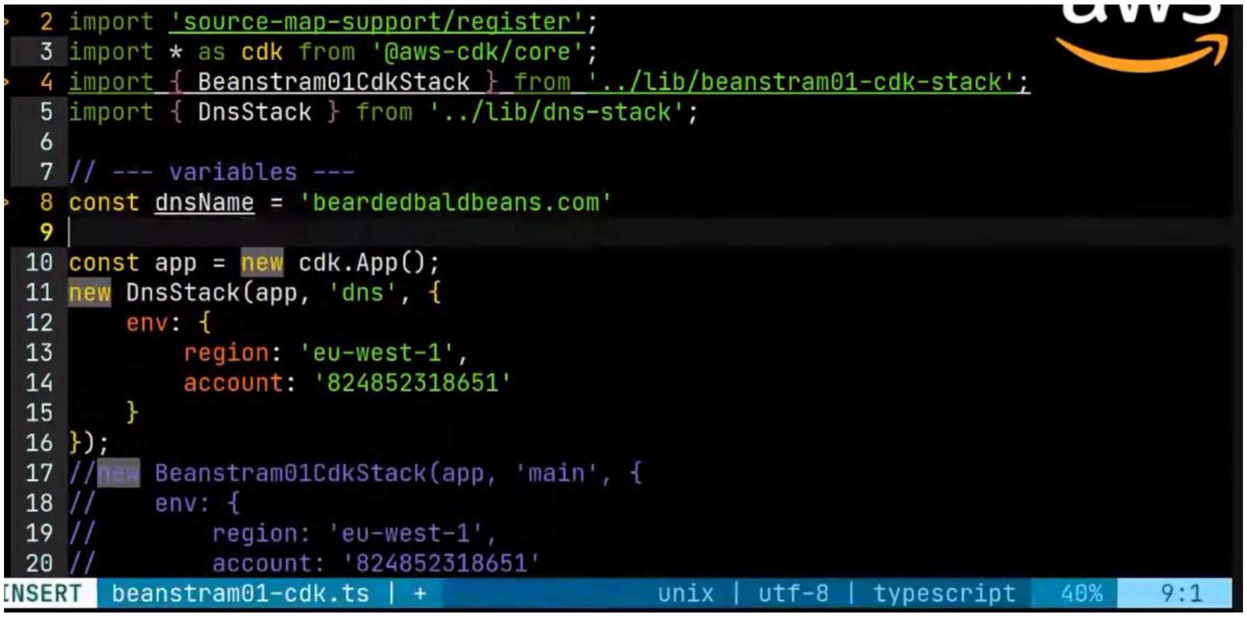


```

    webLbListener.addTargets('webLbWebAppTargets', {
      targets: [ webAppECSService],
      port: 80,
    });

    // --- http / tcp/443 listener

    // --- SSL cert ---
    const cert = new acm.DnsValidatedCertificate(this, 'BeanStreamingECS', {
      domainName: apexDomain,
      subjectAlternativeNames: [wwwDomain],
      hostedZone: hostedZone
    });            [tsserver 2304] [E] Cannot find name 'hostedZone'.

    //new ApplicationListenerCertificate

    const web443LbListener = webLb.addListener('beanStreamingHTTPS',{
ORMAL  beanstram01-cdk-stack.ts          unix | utf-8 | typescript   70%   129:19
```

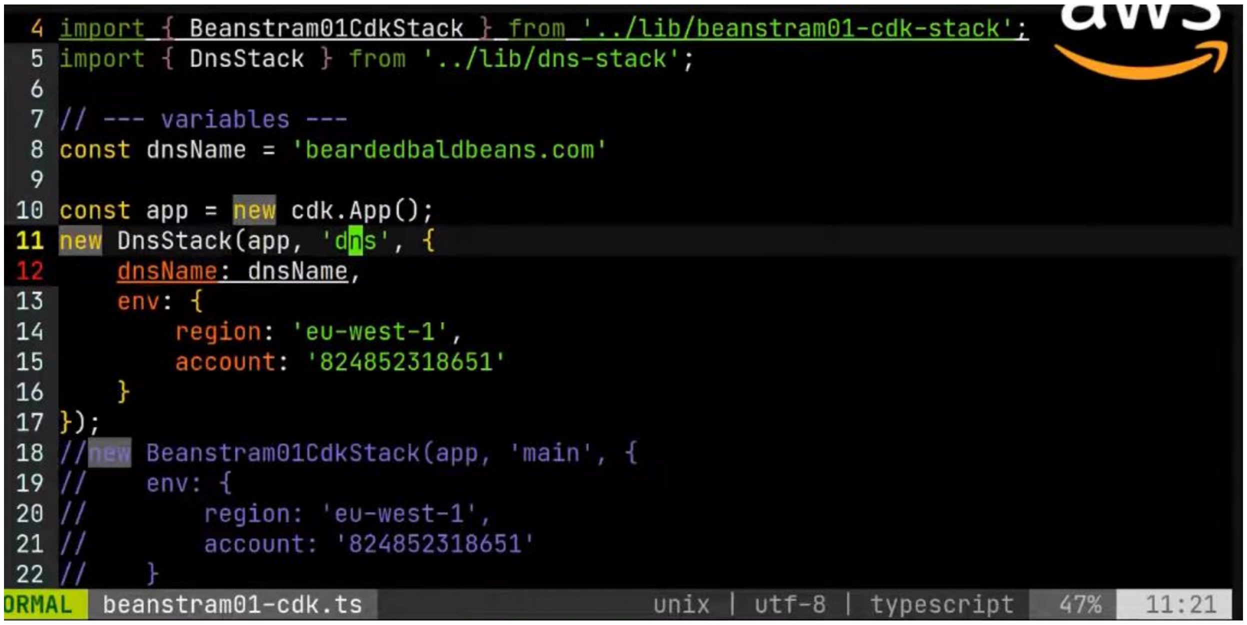
We now get an error because the cert is referencing the **hostedZone dns** object. We need to define a hostedZone



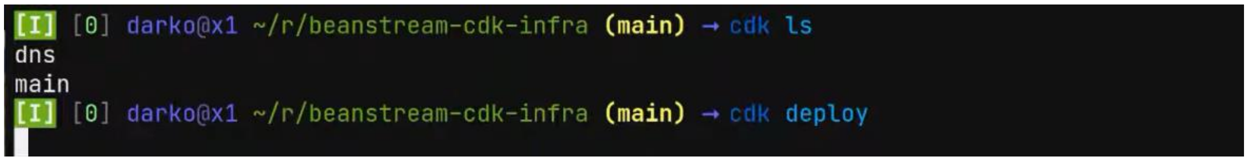
```
    // --- http / tcp/443 listener
    hostedZone = r53.HostedZone.fromLookup(this, 'beanHostedZone',

                                        );

    // --- SSL cert ---
    const cert = new acm.DnsValidatedCertificate(this, 'BeanStreamingECS', {
      domainName: apexDomain,
      subjectAlternativeNames: [wwwDomain],
      hostedZone: hostedZone
    });

    //new ApplicationListenerCertificate

    const web443LbListener = webLb.addListener('beanStreamingHTTPS',{
      port: 443,
      open: true,
INSERT  beanstram01-cdk-stack.ts | +          unix | utf-8 | typescript   67%  125:44
```



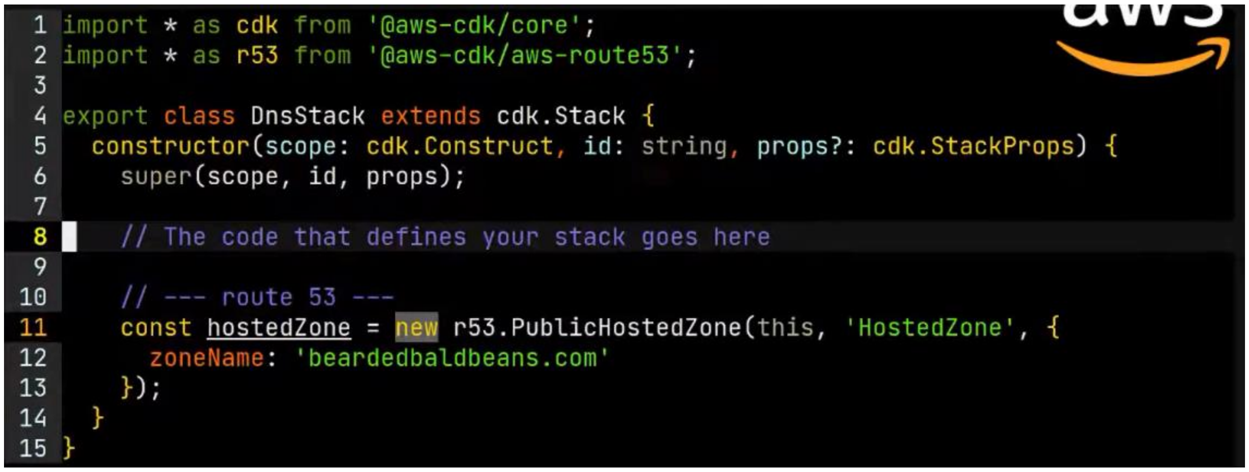
```
import 'source-map-support/register';
import * as cdk from '@aws-cdk/core';
import { Beanstram01CdkStack } from '../lib/beanstram01-cdk-stack';
import { DnsStack } from '../lib/dns-stack';

// --- variables ---
const dnsName = 'beardedbaldbeans.com'

const app = new cdk.App();
new DnsStack(app, 'dns', {
    env: {
        region: 'eu-west-1',
        account: '824852318651'
    }
});
//new Beanstram01CdkStack(app, 'main', {
//    env: {
//        region: 'eu-west-1',
//        account: '824852318651'
INSERT  beanstram01-cdk.ts | +          unix | utf-8 | typescript   40%   9:1
```



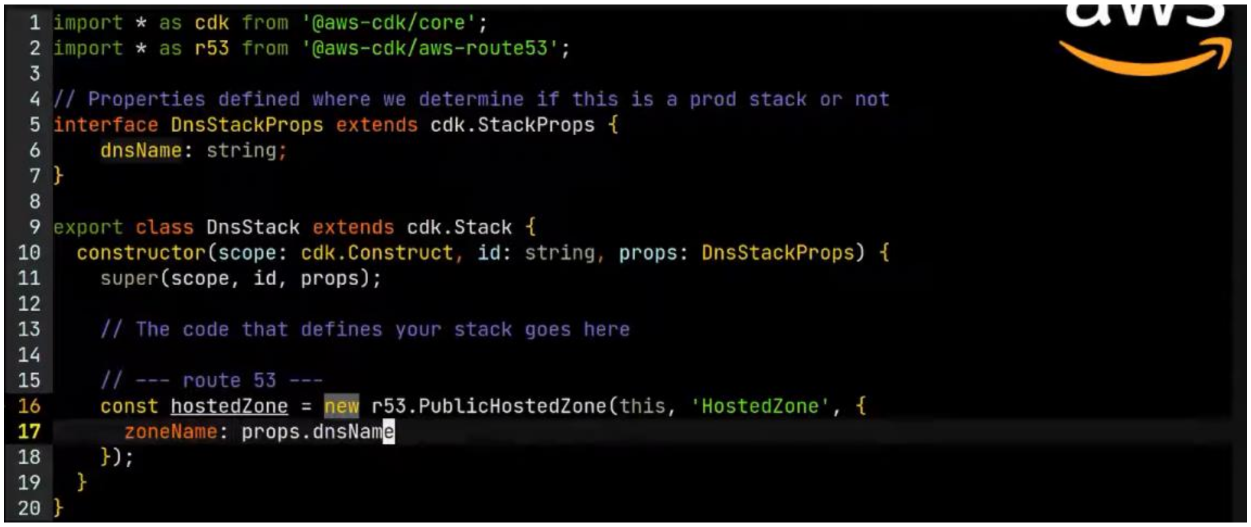
```
import { Beanstram01CdkStack } from '../lib/beanstram01-cdk-stack';
import { DnsStack } from '../lib/dns-stack';

// --- variables ---
const dnsName = 'beardedbaldbeans.com'

const app = new cdk.App();
new DnsStack(app, 'dns', {
    dnsName: dnsName,
    env: {
        region: 'eu-west-1',
        account: '824852318651'
    }
});
//new Beanstram01CdkStack(app, 'main', {
//    env: {
//        region: 'eu-west-1',
//        account: '824852318651'
//    }
NORMAL  beanstram01-cdk.ts          unix | utf-8 | typescript   47%  11:21
```



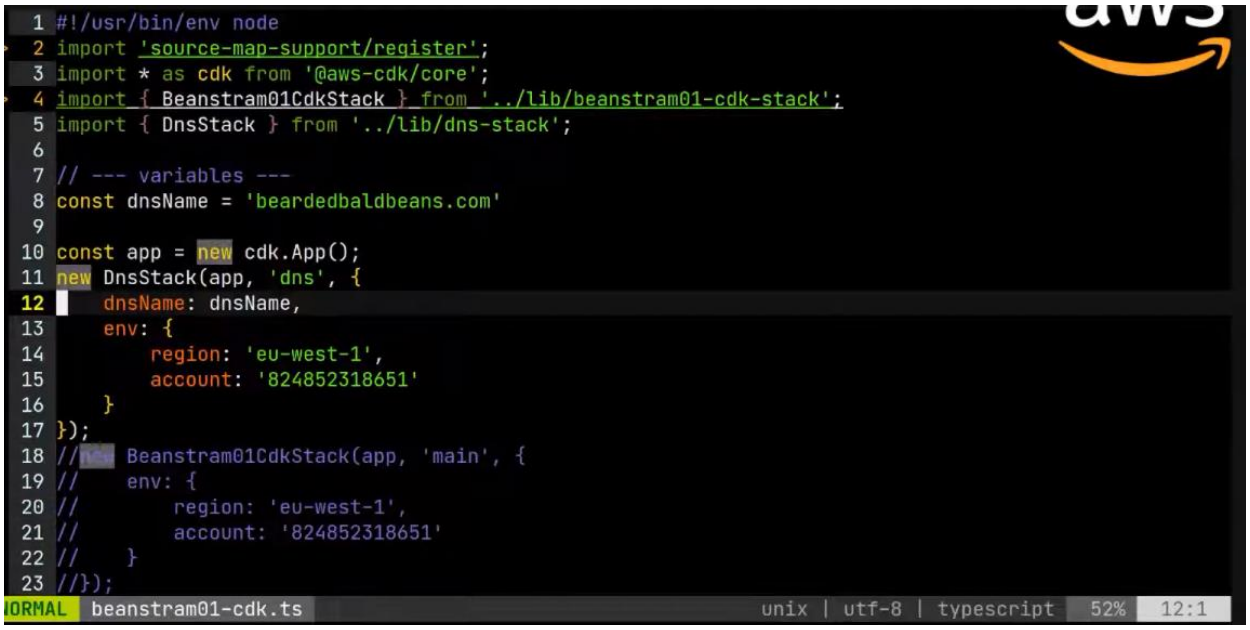
```
[I] [0] darko@x1 ~/r/beanstream-cdk-infra (main) → cdk ls
dns
main
[I] [0] darko@x1 ~/r/beanstream-cdk-infra (main) → cdk deploy
```



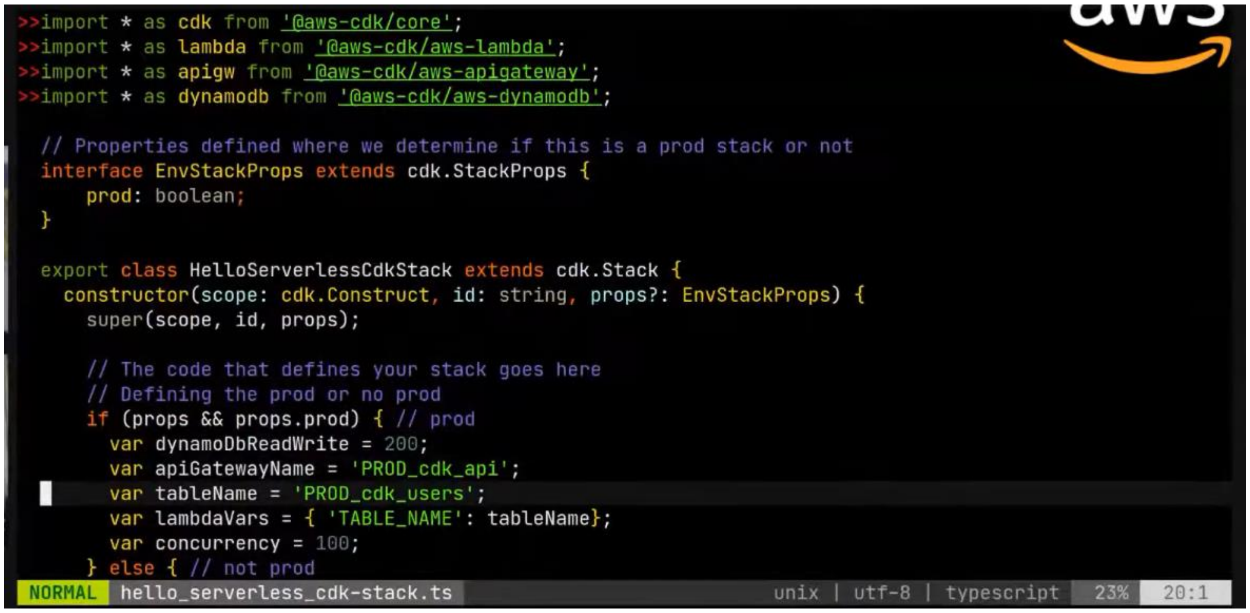
```typescript
import * as cdk from '@aws-cdk/core';
import * as r53 from '@aws-cdk/aws-route53';

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

    // The code that defines your stack goes here

    // --- route 53 ---
    const hostedZone = new r53.PublicHostedZone(this, 'HostedZone', {
      zoneName: 'beardedbaldbeans.com'
    });
  }
}
```



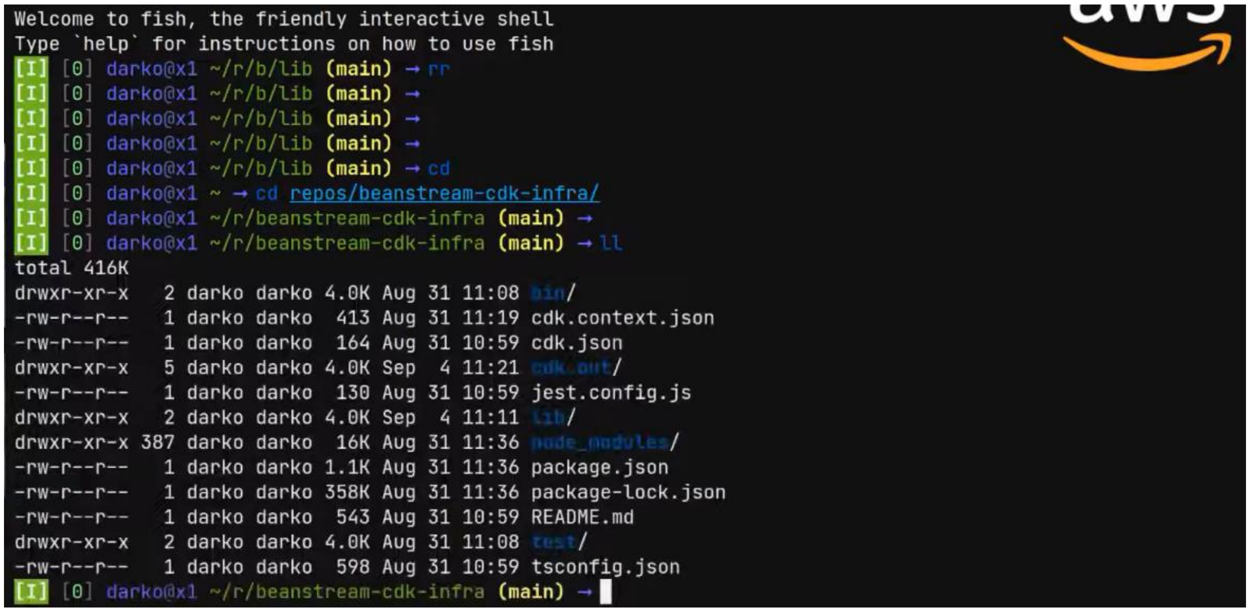
```typescript
import * as cdk from '@aws-cdk/core';
import * as r53 from '@aws-cdk/aws-route53';

// Properties defined where we determine if this is a prod stack or not
interface DnsStackProps extends cdk.StackProps {
    dnsName: string;
}

export class DnsStack extends cdk.Stack {
  constructor(scope: cdk.Construct, id: string, props: DnsStackProps) {
    super(scope, id, props);

    // The code that defines your stack goes here

    // --- route 53 ---
    const hostedZone = new r53.PublicHostedZone(this, 'HostedZone', {
      zoneName: props.dnsName
    });
  }
}
```



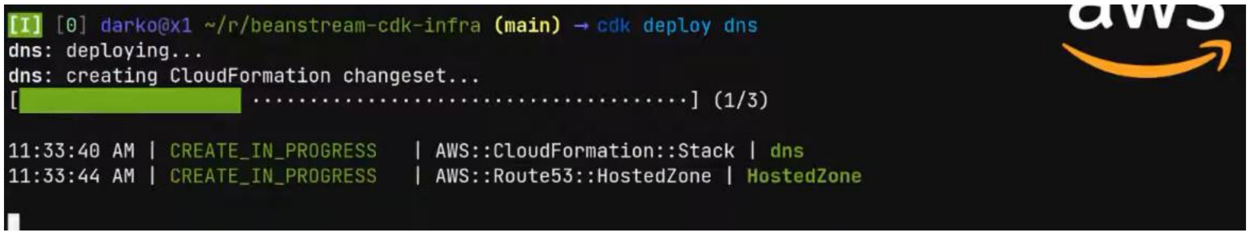
```typescript
#!/usr/bin/env node
import 'source-map-support/register';
import * as cdk from '@aws-cdk/core';
import { Beanstram01CdkStack } from '../lib/beanstram01-cdk-stack';
import { DnsStack } from '../lib/dns-stack';

// --- variables ---
const dnsName = 'beardedbaldbeans.com'

const app = new cdk.App();
new DnsStack(app, 'dns', {
    dnsName: dnsName,
    env: {
        region: 'eu-west-1',
        account: '824852318651'
    }
});
//new Beanstram01CdkStack(app, 'main', {
//    env: {
//        region: 'eu-west-1',
//        account: '824852318651'
//    }
//});
ORMAL  beanstram01-cdk.ts                    unix | utf-8 | typescript  52%   12:1
```



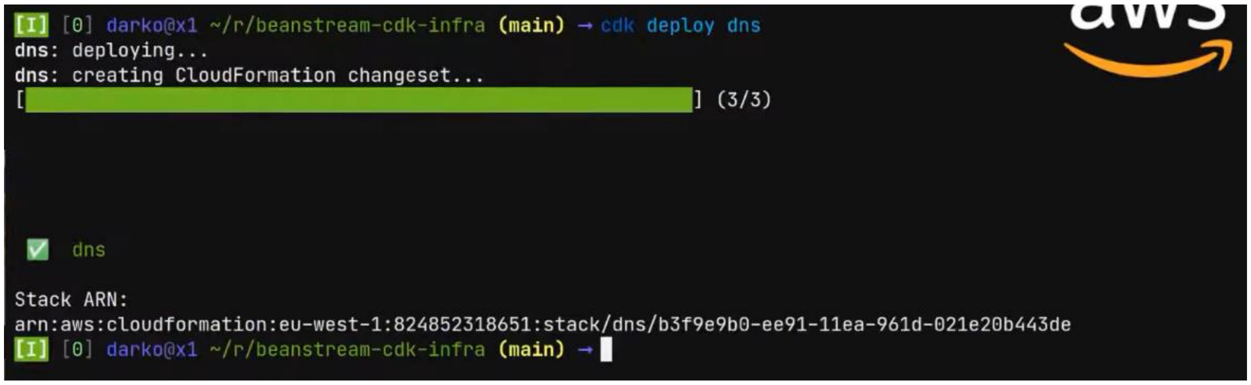
```
>>import * as cdk from '@aws-cdk/core';
>>import * as lambda from '@aws-cdk/aws-lambda';
>>import * as apigw from '@aws-cdk/aws-apigateway';
>>import * as dynamodb from '@aws-cdk/aws-dynamodb';

  // Properties defined where we determine if this is a prod stack or not
  interface EnvStackProps extends cdk.StackProps {
      prod: boolean;
  }

  export class HelloServerlessCdkStack extends cdk.Stack {
    constructor(scope: cdk.Construct, id: string, props?: EnvStackProps) {
      super(scope, id, props);

      // The code that defines your stack goes here
      // Defining the prod or no prod
      if (props && props.prod) { // prod
        var dynamoDbReadWrite = 200;
        var apiGatewayName = 'PROD_cdk_api';
        var tableName = 'PROD_cdk_users';
        var lambdaVars = { 'TABLE_NAME': tableName};
        var concurrency = 100;
      } else { // not prod
 NORMAL  hello_serverless_cdk-stack.ts                         unix | utf-8 | typescript  23%   20:1
```



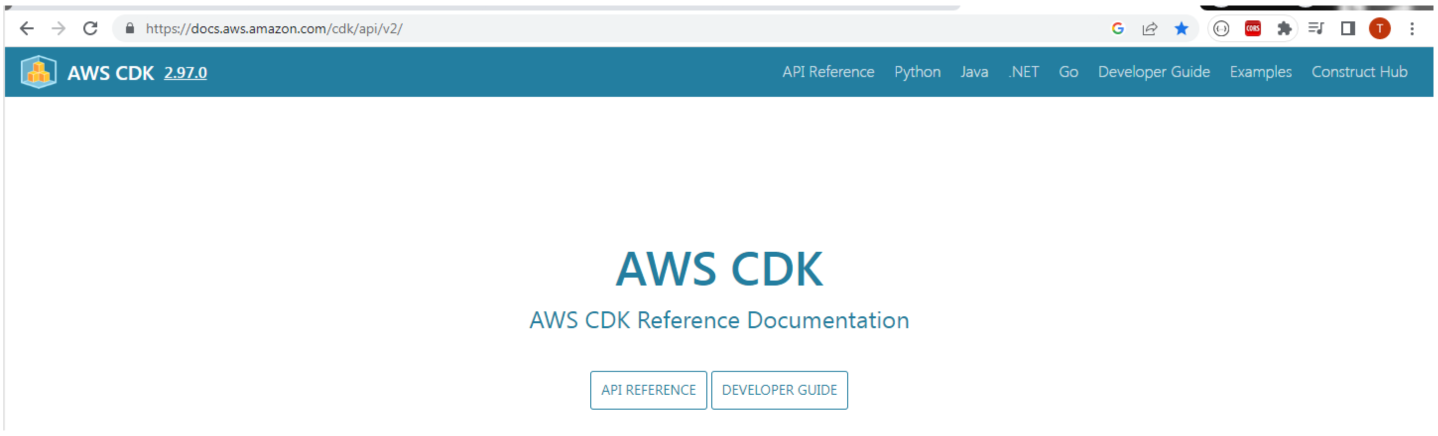
```
Welcome to fish, the friendly interactive shell
Type `help` for instructions on how to use fish
[I] [0] darko@x1 ~/r/b/lib (main) → rr
[I] [0] darko@x1 ~/r/b/lib (main) →
[I] [0] darko@x1 ~/r/b/lib (main) →
[I] [0] darko@x1 ~/r/b/lib (main) →
[I] [0] darko@x1 ~/r/b/lib (main) → cd
[I] [0] darko@x1 ~ → cd repos/beanstream-cdk-infra/
[I] [0] darko@x1 ~/r/beanstream-cdk-infra (main) →
[I] [0] darko@x1 ~/r/beanstream-cdk-infra (main) → ll
total 416K
drwxr-xr-x   2 darko darko 4.0K Aug 31 11:08 bin/
-rw-r--r--   1 darko darko  413 Aug 31 11:19 cdk.context.json
-rw-r--r--   1 darko darko  164 Aug 31 10:59 cdk.json
drwxr-xr-x   5 darko darko 4.0K Sep  4 11:21 cdk.out/
-rw-r--r--   1 darko darko  130 Aug 31 10:59 jest.config.js
drwxr-xr-x   2 darko darko 4.0K Sep  4 11:11 lib/
drwxr-xr-x 387 darko darko  16K Aug 31 11:36 node_modules/
-rw-r--r--   1 darko darko 1.1K Aug 31 11:36 package.json
-rw-r--r--   1 darko darko 358K Aug 31 11:36 package-lock.json
-rw-r--r--   1 darko darko  543 Aug 31 10:59 README.md
drwxr-xr-x   2 darko darko 4.0K Aug 31 11:08 test/
-rw-r--r--   1 darko darko  598 Aug 31 10:59 tsconfig.json
[I] [0] darko@x1 ~/r/beanstream-cdk-infra (main) →
```



```
[I] [0] darko@x1 ~/r/beanstream-cdk-infra (main) → cdk deploy dns
dns: deploying...
dns: creating CloudFormation changeset...
[███████████████████·····································] (1/3)

11:33:40 AM | CREATE_IN_PROGRESS   | AWS::CloudFormation::Stack | dns
11:33:44 AM | CREATE_IN_PROGRESS   | AWS::Route53::HostedZone | HostedZone
```



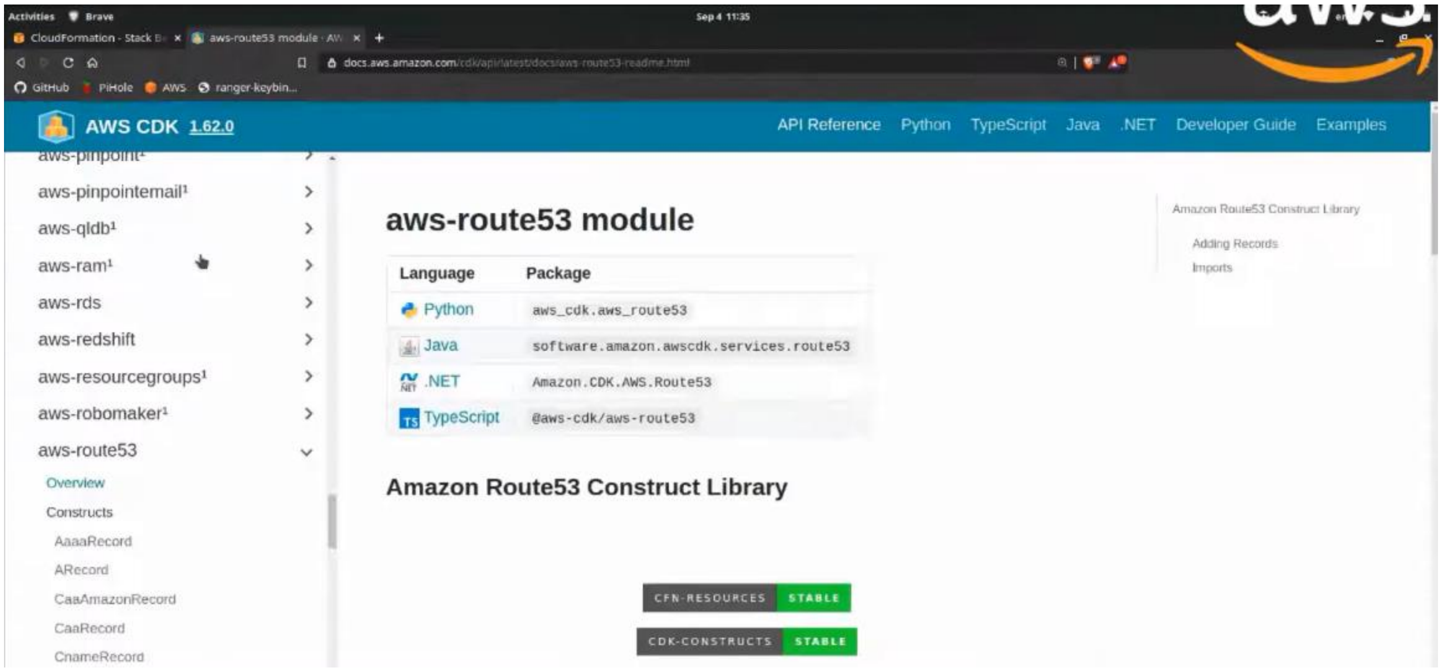
```
[I] [0] darko@x1 ~/r/beanstream-cdk-infra (main) → cdk deploy dns
dns: deploying...
dns: creating CloudFormation changeset...
[████████████████████████████████████████████████████████] (3/3)



 ✅  dns

Stack ARN:
arn:aws:cloudformation:eu-west-1:824852318651:stack/dns/b3f9e9b0-ee91-11ea-961d-021e20b443de
[I] [0] darko@x1 ~/r/beanstream-cdk-infra (main) →
```



AWS CDK

AWS CDK Reference Documentation

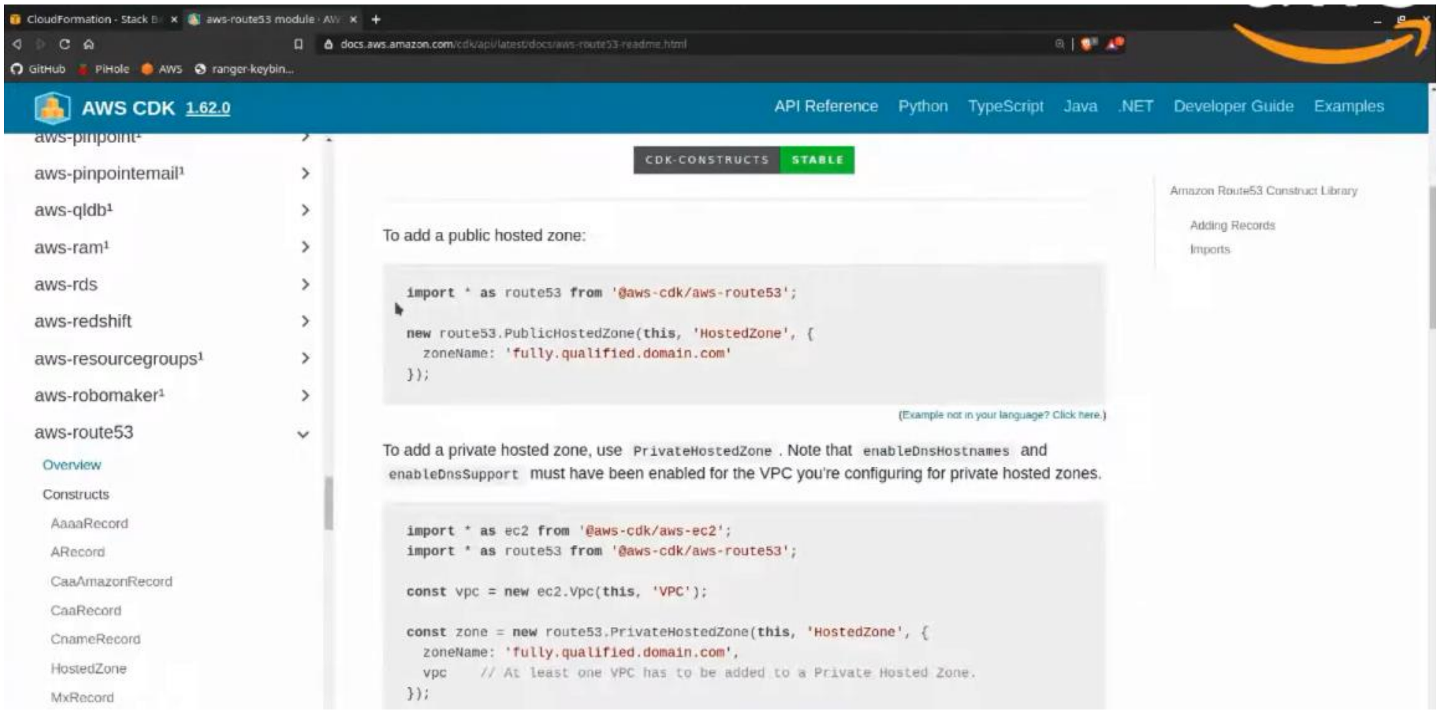
https://docs.aws.amazon.com/cdk/api/v2/



aws-route53 module

Language	Package
Python	`aws_cdk.aws_route53`
Java	`software.amazon.awscdk.services.route53`
.NET	`Amazon.CDK.AWS.Route53`
TypeScript	`@aws-cdk/aws-route53`

Amazon Route53 Construct Library



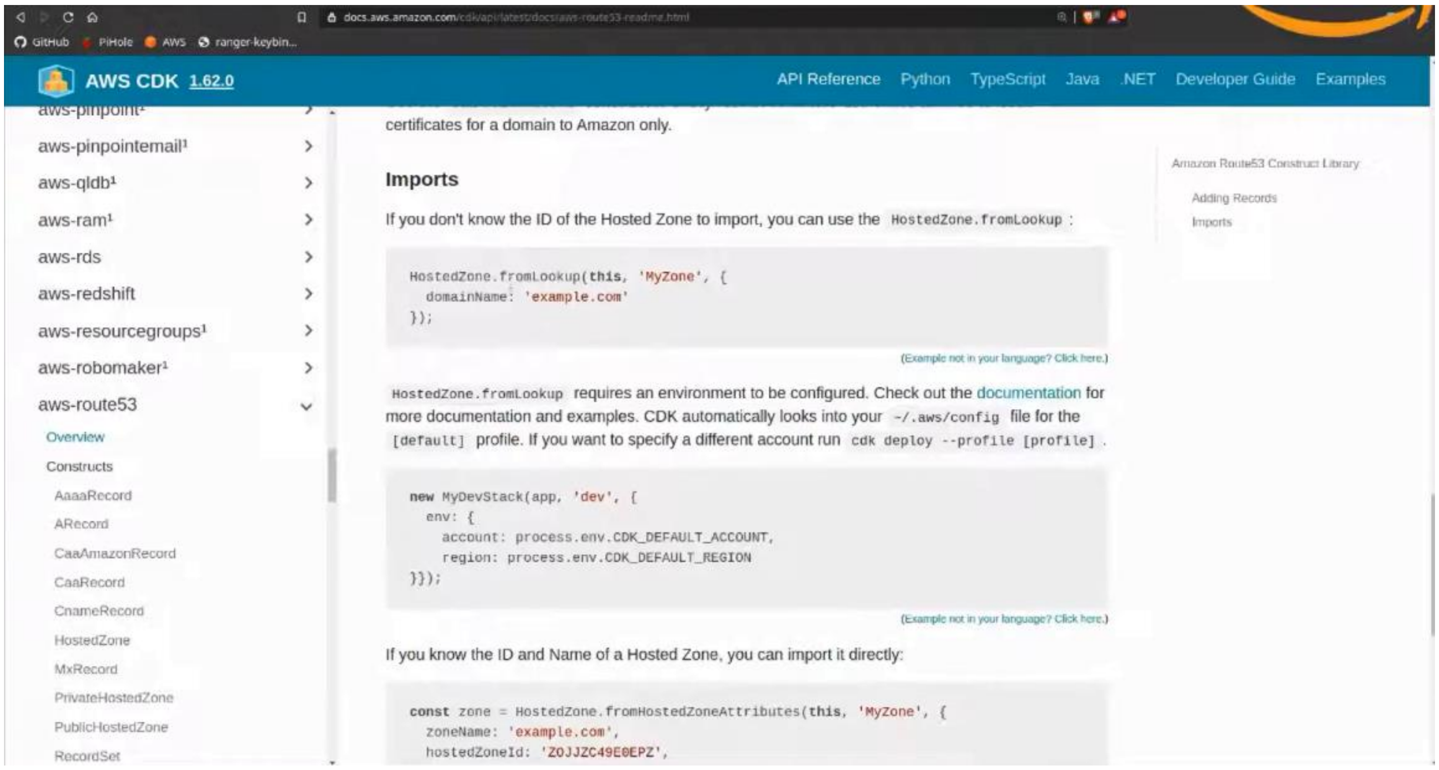
To add a public hosted zone:

```
import * as route53 from '@aws-cdk/aws-route53';

new route53.PublicHostedZone(this, 'HostedZone', {
  zoneName: 'fully.qualified.domain.com'
});
```

(Example not in your language? Click here.)

To add a private hosted zone, use `PrivateHostedZone`. Note that `enableDnsHostnames` and `enableDnsSupport` must have been enabled for the VPC you're configuring for private hosted zones.

```
import * as ec2 from '@aws-cdk/aws-ec2';
import * as route53 from '@aws-cdk/aws-route53';

const vpc = new ec2.Vpc(this, 'VPC');

const zone = new route53.PrivateHostedZone(this, 'HostedZone', {
  zoneName: 'fully.qualified.domain.com',
  vpc    // At least one VPC has to be added to a Private Hosted Zone.
});
```



certificates for a domain to Amazon only.

Imports

If you don't know the ID of the Hosted Zone to import, you can use the `HostedZone.fromLookup` :

```
HostedZone.fromLookup(this, 'MyZone', {
  domainName: 'example.com'
});
```

(Example not in your language? Click here.)

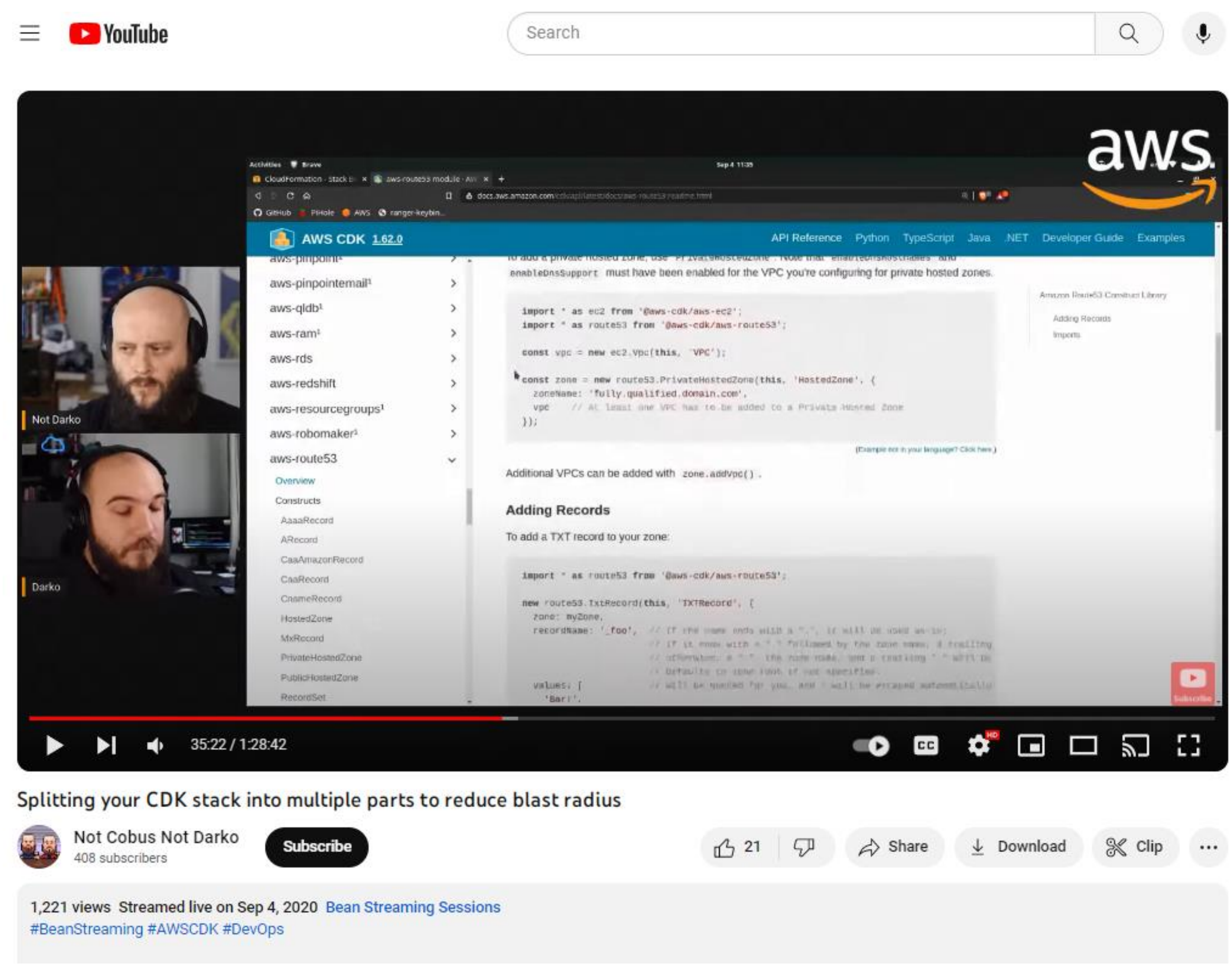
`HostedZone.fromLookup` requires an environment to be configured. Check out the documentation for more documentation and examples. CDK automatically looks into your `~/.aws/config` file for the `[default]` profile. If you want to specify a different account run `cdk deploy --profile [profile]` .

```
new MyDevStack(app, 'dev', {
  env: {
    account: process.env.CDK_DEFAULT_ACCOUNT,
    region: process.env.CDK_DEFAULT_REGION
}});
```

(Example not in your language? Click here.)

If you know the ID and Name of a Hosted Zone, you can import it directly:

```
const zone = HostedZone.fromHostedZoneAttributes(this, 'MyZone', {
  zoneName: 'example.com',
  hostedZoneId: 'ZOJJZC49E0EPZ',
```



`enableDnsSupport` must have been enabled for the VPC you're configuring for private hosted zones.

```
import * as ec2 from '@aws-cdk/aws-ec2';
import * as route53 from '@aws-cdk/aws-route53';

const vpc = new ec2.Vpc(this, 'VPC');

const zone = new route53.PrivateHostedZone(this, 'HostedZone', {
  zoneName: 'fully.qualified.domain.com',
  vpc    // At least one VPC has to be added to a Private Hosted Zone
});
```

(Example not in your language? Click here.)

Additional VPCs can be added with `zone.addVpc()` .

Adding Records

To add a TXT record to your zone:

```
import * as route53 from '@aws-cdk/aws-route53';

new route53.TxtRecord(this, 'TXTRecord', {
  zone: myZone,
  recordName: '_foo',  // If the name ends with a ".", it will be used as-is;
                       // if it ends with a "." followed by the zone name, a trailing
                       // otherwise, a ".", the zone name, and a trailing "." will be
                       // Defaults to zone root if not specified.
  values: [            // Will be quoted for you, and " will be escaped automatically
    'Bar!',
```

Splitting your CDK stack into multiple parts to reduce blast radius

Not Cobus Not Darko
408 subscribers

Subscribe

21 | Share | Download | Clip | ...

1,221 views Streamed live on Sep 4, 2020 Bean Streaming Sessions
#BeanStreaming #AWSCDK #DevOps