:doctype: book

include::attributes.txt[]

// Attributes

[.topic] [#testing-locally-with-sam-cli] = Local testing {aws} CDK applications with {aws} SAM :info\_titleabbrev: Local testing :keywords: sam local, test, {aws} CDK, {aws} SAM

== [abstract]

## Use the sam local command to locally test {aws} CDK applications.

// Content start

You can use the {aws} SAM CLI to locally test your {aws} CDK applications by running the following commands from the project root directory of your {aws} CDK application:

* +link:https://docs.aws.amazon.com/serverless-application-model/latest/developerguide/sam-cli-command-reference-sam-local-invoke.html[sam local invoke]+
* +link:https://docs.aws.amazon.com/serverless-application-model/latest/developerguide/sam-cli-command-reference-sam-local-start-api.html[sam local start-api]+
* +link:https://docs.aws.amazon.com/serverless-application-model/latest/developerguide/sam-cli-command-reference-sam-local-start-lambda.html[sam local start-lambda]+

Before you run any of the sam local commands with a {aws} CDK application, you must run cdk synth.

When running sam local invoke you need the function construct identifier that you want to invoke, and the path to your synthesized {aws} CloudFormation template. If your application uses nested stacks, to resolve naming conflicts, you also need the stack name where the function is defined.

*Usage*:: + [source,none,subs=“verbatim,attributes”] —

= Invoke the function FUNCTION\_IDENTIFIER declared in the stack STACK\_NAME

$ sam local invoke ++++++++++++

= Start all APIs declared in the {aws} CDK application

$ sam local start-api -t <./cdk.out/CdkSamExampleStack.template.json> ++++++++++++

= Start a local endpoint that emulates {aws} Lambda

$ sam local start-lambda -t <./cdk.out/CdkSamExampleStack.template.json> ++++++—-++++++

[#testing-cdk-applications-examples] == Example

Consider stacks and functions that are declared with the following sample:

== [source,javascript,subs=“verbatim,attributes”]

app = new HelloCdkStack(app, “HelloCdkStack”, … ) class HelloCdkStack extends cdk.Stack { constructor(scope: Construct, id: string, props?: cdk.StackProps) { … new lambda.Function(this, ‘MyFunction’, { … });

new HelloCdkNestedStack(this, ‘HelloNestedStack’ ,{ … }); } }

class HelloCdkNestedStack extends cdk.NestedStack { constructor(scope: Construct, id: string, props?: cdk.NestedStackProps) { … new lambda.Function(this, ‘MyFunction’, { … }); new lambda.Function(this, ‘MyNestedFunction’, { … }); } } —

The following commands locally invokes the Lambda functions defined in example presented above:

== [source,none,subs=“verbatim,attributes”]

= Invoke MyFunction from the HelloCdkStack

$ sam local invoke -t <./cdk.out/HelloCdkStack.template.json> ++++++—-++++++

== [source,none,subs=“verbatim,attributes”]

= Invoke MyNestedFunction from the HelloCdkNestedStack

$ sam local invoke -t <./cdk.out/HelloCdkStack.template.json> ++++++—-++++++

== [source,none,subs=“verbatim,attributes”]

= Invoke MyFunction from the HelloCdkNestedStack

## $ sam local invoke -t <./cdk.out/HelloCdkStack.template.json>