include::attributes.txt[]

// Attributes [.topic] [#core-concepts] = Learn {aws} CDK core concepts :info\_titleabbrev: CDK core concepts :keywords: {aws} Cloud Development Kit ({aws} CDK), {aws} CDK, IaC, Infrastructure as code, {aws} CloudFormation, Serverless applications, Modern applications, {aws}

== [abstract]

## Learn core concepts behind the {aws} Cloud Development Kit ({aws} CDK).

// Content start

Learn core concepts behind the {aws} Cloud Development Kit ({aws} CDK).

[#concepts-iac] == {aws} CDK and [.noloc]IaC

The {aws} CDK is an open-source framework that you can use to manage your {aws} infrastructure using code. This approach is known as *infrastructure as code ([.noloc]IaC)*. By managing and provisioning your infrastructure as code, you treat your infrastructure in the same way that developers treat code. This provides many benefits, such as version control and scalability. To learn more about IaC, see https://aws.amazon.com/what-is/iac/[What is Infrastructure as Code?]

[#concepts-cfn] == {aws} CDK and {aws} CloudFormation

The {aws} CDK is tightly integrated with {aws} CloudFormation. {aws} CloudFormation is a fully managed service that you can use to manage and provision your infrastructure on {aws}. With {aws} CloudFormation, you define your infrastructure in templates and deploy them to {aws} CloudFormation. The {aws} CloudFormation service then provisions your infrastructure according to the configuration defined on your templates.

{aws} CloudFormation templates are *declarative*, meaning they declare the desired state or outcome of your infrastructure. Using JSON or YAML, you declare your {aws} infrastructure by defining {aws} *resources* and *properties*. Resources represent the many services on {aws} and properties represent your desired configuration of those services. When you deploy your template to {aws} CloudFormation, your resources and their configured properties are provisioned as described on your template.

With the {aws} CDK, you can manage your infrastructure *imperatively*, using general-purpose programming languages. Instead of just defining a desired state declaratively, you can define the logic or sequence necessary to reach the desired state. For example, you can use if statements or conditional loops that determine how to reach a desired end state for your infrastructure.

Infrastructure created with the {aws} CDK is eventually translated, or *synthesized* into {aws} CloudFormation templates and deployed using the {aws} CloudFormation service. So while the {aws} CDK offers a different approach to creating your infrastructure, you still receive the benefits of {aws} CloudFormation, such as extensive {aws} resource configuration support and robust deployment processes.

To learn more about {aws} CloudFormation, see https://docs.aws.amazon.com/AWSCloudFormation/latest/UserGuide/Welcome.html[What is {aws} CloudFormation?] in the *{aws} CloudFormation User Guide*.

[#concepts-abstractions] == {aws} CDK and abstractions

With {aws} CloudFormation, you must define every detail of how your resources are configured. This provides the benefit of having complete control over your infrastructure. However, this requires you to learn, understand, and create robust templates that contain resource configuration details and relationships between resources, such as permissions and event-driven interactions.

With the {aws} CDK, you can have the same control over your resource configurations. However, the {aws} CDK also offers powerful abstractions, which can speed up and simplify the infrastructure development process. For example, the {aws} CDK includes constructs that provide sensible default configurations and helper methods that generate boilerplate code for you. The {aws} CDK also offers tools, such as the {aws} CDK Command Line Interface ({aws} CDK CLI), that perform infrastructure management actions for you.

[#concepts-learn] == Learn more about core {aws} CDK concepts

[#concepts-learn-interact] *Interacting with the {aws} CDK*:: + When using with the {aws} CDK, you will primarily interact with the {aws} Construct Library and the {aws} CDK CLI.

[#concepts-learn-develop] *Developing with the {aws} CDK*:: + The {aws} CDK can be written in any xref:languages[supported programming language]. You start with a xref:projects[CDK project], which contains a structure of folders and files, including xref:assets[assets]. Within the project, you create a xref:apps[CDK application]. Within the app, you define a xref:stacks[stack], which directly represents a CloudFormation stack. Within the stack, you define your {aws} resources and properties using xref:constructs[constructs].

[#concepts-learn-deploy] *Deploying with the {aws} CDK*:: + You deploy CDK apps into an {aws} xref:environments[environment]. Before deploying, you must perform a one-time xref:bootstrapping[bootstrapping] to prepare your environment.

[#concepts-learn-more] *Learn more*:: + To learn more about {aws} CDK core concepts, see the topics in this section.

include::languages.adoc[leveloffset=+1]

include::libraries.adoc[leveloffset=+1]

include::projects.adoc[leveloffset=+1]

include::apps.adoc[leveloffset=+1]

include::stacks.adoc[leveloffset=+1]

include::stages.adoc[leveloffset=+1]

include::constructs.adoc[leveloffset=+1]

include::environments.adoc[leveloffset=+1]

include::bootstrapping.adoc[leveloffset=+1]

include::resources.adoc[leveloffset=+1]

include::identifiers.adoc[leveloffset=+1]

include::tokens.adoc[leveloffset=+1]

include::parameters.adoc[leveloffset=+1]

include::tagging.adoc[leveloffset=+1]

include::assets.adoc[leveloffset=+1]

include::permissions.adoc[leveloffset=+1]

include::context.adoc[leveloffset=+1]

include::feature-flags.adoc[leveloffset=+1]

include::aspects.adoc[leveloffset=+1]