:doctype: book

include::attributes.txt[]

// Attributes

[.topic] [#work-with-cdk-java] = Working with the {aws} CDK in Java :info\_titleabbrev: In Java

// Content start

Java is a fully-supported client language for the {aws} CDK and is considered stable. You can develop {aws} CDK applications in Java using familiar tools, including the JDK (Oracle’s, or an OpenJDK distribution such as Amazon Corretto) and Apache Maven.

The {aws} CDK supports Java 8 and later. We recommend using the latest version you can, however, because later versions of the language include improvements that are particularly convenient for developing {aws} CDK applications. For example, Java 9 introduces the Map.of() method (a convenient way to declare hashmaps that would be written as object literals in TypeScript). Java 10 introduces local type inference using the var keyword.

= [NOTE]

Most code examples in this Developer Guide work with Java 8. A few examples use Map.of(); these examples include comments noting that they require Java 9.

====

You can use any text editor, or a Java IDE that can read Maven projects, to work on your {aws} CDK apps. We provide https://www.eclipse.org/downloads/[Eclipse] hints in this Guide, but IntelliJ IDEA, NetBeans, and other IDEs can import Maven projects and can be used for developing {aws} CDK applications in Java.

It is possible to write {aws} CDK applications in JVM-hosted languages other than Java (for example, Kotlin, Groovy, Clojure, or Scala), but the experience may not be particularly idiomatic, and we are unable to provide any support for these languages.

[#java-prerequisites] == Get started with Java

To work with the {aws} CDK, you must have an {aws} account and credentials and have installed Node.js and the {aws} CDK Toolkit. See xref:getting-started[Getting started with the {aws} CDK].

Java {aws} CDK applications require Java 8 (v1.8) or later. We recommend https://aws.amazon.com/corretto/[Amazon Corretto], but you can use any OpenJDK distribution or https://www.oracle.com/technetwork/java/javase/downloads/jdk8-downloads-2133151.html[Oracle’s JDK]. You will also need https://maven.apache.org/download.cgi[Apache Maven] 3.5 or later. You can also use tools such as Gradle, but the application skeletons generated by the {aws} CDK Toolkit are Maven projects.

= [NOTE]

Third-party language deprecation: language version is only supported until its EOL (End Of Life) shared by the vendor or community and is subject to change with prior notice.

====

[#java-newproject] == Creating a project

You create a new {aws} CDK project by invoking cdk init in an empty directory. Use the --language option and specify java:

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

$ mkdir my-project $ cd my-project $ cdk init app –language java —

cdk init uses the name of the project folder to name various elements of the project, including classes, subfolders, and files. Hyphens in the folder name are converted to underscores. However, the name should otherwise follow the form of a Java identifier; for example, it should not start with a number or contain spaces.

The resulting project includes a reference to the software.amazon.awscdk Maven package. It and its dependencies are automatically installed by Maven.

If you are using an IDE, you can now open or import the project. In Eclipse, for example, choose *File* > *Import* > *Maven* > *Existing Maven Projects*. Make sure that the project settings are set to use Java 8 (1.8).

[#java-managemodules] == Managing {aws} Construct Library modules

Use Maven to install {aws} Construct Library packages, which are in the group software.amazon.awscdk. Most constructs are in the artifact aws-cdk-lib, which is added to new Java projects by default. Modules for services whose higher-level CDK support is still being developed are in separate “experimental” packages, named with a short version (no {aws} or Amazon prefix) of their service’s name. https://search.maven.org/search?q=software.amazon.awscdk[Search the Maven Central Repository] to find the names of all {aws} CDK and {aws} Construct Module libraries.

= [NOTE]

The https://docs.aws.amazon.com/cdk/api/v2/java/index.html[Java edition of the CDK API Reference] also shows the package names.

====

Some services’ {aws} Construct Library support is in more than one namespace. For example, Amazon Route 53 has its functionality divided into software.amazon.awscdk.route53, route53-patterns, route53resolver, and route53-targets.

The main {aws} CDK package is imported in Java code as software.amazon.awscdk. Modules for the various services in the {aws} Construct Library live under software.amazon.awscdk.services and are named similarly to their Maven package name. For example, the Amazon S3 module’s namespace is software.amazon.awscdk.services.s3.

We recommend writing a separate Java import statement for each {aws} Construct Library class you use in each of your Java source files, and avoiding wildcard imports. You can always use a type’s fully-qualified name (including its namespace) without an import statement.

If your application depends on an experimental package, edit your project’s pom.xml and add a new <dependency> element in the <dependencies> container. For example, the following <dependency> element specifies the CodeStar experimental construct library module:

== [source,xml,subs=“verbatim,attributes”]++++++++++++software.amazon.awscdk++++++ ++++++codestar-alpha++++++ ++++++2.0.0-alpha.10++++++++++++

’’’

= [TIP]

If you use a Java IDE, it probably has features for managing Maven dependencies. We recommend editing pom.xml directly, however, unless you are absolutely sure the IDE’s functionality matches what you’d do by hand.

====

[#work-with-cdk-java-dependencies] == Managing dependencies in Java

In Java, dependencies are specified in pom.xml and installed using Maven. The <dependencies> container includes a <dependency> element for each package. Following is a section of pom.xml for a typical CDK Java app.

== [source,xml,subs=“verbatim,attributes”]++++++++++++++++++software.amazon.awscdk++++++ ++++++aws-cdk-lib++++++ ++++++2.14.0++++++++++++ ++++++++++++software.amazon.awscdk++++++ ++++++appsync-alpha++++++ ++++++2.10.0-alpha.0++++++++++++++++++

’’’

= [TIP]

Many Java IDEs have integrated Maven support and visual pom.xml editors, which you may find convenient for managing dependencies.

====

Maven does not support dependency locking. Although it’s possible to specify version ranges in pom.xml, we recommend you always use exact versions to keep your builds repeatable.

Maven automatically installs transitive dependencies, but there can only be one installed copy of each package. The version that is specified highest in the POM tree is selected; applications always have the last word in what version of packages get installed.

Maven automatically installs or updates your dependencies whenever you build (mvn compile) or package (mvn package) your project. The CDK Toolkit does this automatically every time you run it, so generally there is no need to manually invoke Maven.

[#java-cdk-idioms] == {aws} CDK idioms in Java

[#java-props] === Props

All {aws} Construct Library classes are instantiated using three arguments: the *scope* in which the construct is being defined (its parent in the construct tree), an *id*, and *props*, a bundle of key/value pairs that the construct uses to configure the resources it creates. Other classes and methods also use the “bundle of attributes” pattern for arguments.

In Java, props are expressed using the link:https://en.wikipedia.org/wiki/Builder\_pattern[Builder pattern]. Each construct type has a corresponding props type; for example, the Bucket construct (which represents an Amazon S3 bucket) takes as its props an instance of BucketProps.

The BucketProps class (like every {aws} Construct Library props class) has an inner class called Builder. The BucketProps.Builder type offers methods to set the various properties of a BucketProps instance. Each method returns the Builder instance, so the method calls can be chained to set multiple properties. At the end of the chain, you call build() to actually produce the BucketProps object.

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

Bucket bucket = new Bucket(this, “amzn-s3-demo-bucket”, new BucketProps.Builder() .versioned(true) .encryption(BucketEncryption.KMS\_MANAGED) .build()); —

Constructs, and other classes that take a props-like object as their final argument, offer a shortcut. The class has a Builder of its own that instantiates it and its props object in one step. This way, you don’t need to explicitly instantiate (for example) both BucketProps and a Bucket–and you don’t need an import for the props type.

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

Bucket bucket = Bucket.Builder.create(this, “amzn-s3-demo-bucket”) .versioned(true) .encryption(BucketEncryption.KMS\_MANAGED) .build(); —

When deriving your own construct from an existing construct, you may want to accept additional properties. We recommend that you follow these builder patterns. However, this isn’t as simple as subclassing a construct class. You must provide the moving parts of the two new Builder classes yourself. You may prefer to simply have your construct accept one or more additional arguments. You should provide additional constructors when an argument is optional.

[#java-generic-structures] === Generic structures

In some APIs, the {aws} CDK uses JavaScript arrays or untyped objects as input to a method. (See, for example, {aws} CodeBuild’s link:https://docs.aws.amazon.com/cdk/api/v2/docs/aws-cdk-lib.aws\_codebuild.BuildSpec.html#static-fromwbrobjectvalue[BuildSpec.fromObject()] method.) In Java, these objects are represented as java.util.Map<String, Object>. In cases where the values are all strings, you can use Map<String, String>.

Java does not provide a way to write literals for such containers like some other languages do. In Java 9 and later, you can use link:https://docs.oracle.com/javase/9/docs/api/java/util/Map.html#ofEntries-java.util.Map.Entry…-[java.util.Map.of()] to conveniently define maps of up to ten entries inline with one of these calls.

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

java.util.Map.of( “base-directory”, “dist”, “files”, “LambdaStack.template.json” ) —

To create maps with more than ten entries, use link:https://docs.oracle.com/javase/9/docs/api/java/util/Map.html#ofEntries-java.util.Map.Entry…-[java.util.Map.ofEntries()].

If you are using Java 8, you could provide your own methods similar to to these.

JavaScript arrays are represented as List<Object> or List<String> in Java. The method java.util.Arrays.asList is convenient for defining short pass:[Lists].

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

List++++++cmds = Arrays.asList(“cd lambda”, “npm install”, “npm install typescript”) —-++++++

[#java-missing-values] === Missing values

In Java, missing values in {aws} CDK objects such as props are represented by null. You must explicitly test any value that could be null to make sure it contains a value before doing anything with it. Java does not have “syntactic sugar” to help handle null values as some other languages do. You may find Apache ObjectUtil’s link:https://commons.apache.org/proper/commons-lang/apidocs/org/apache/commons/lang3/ObjectUtils.html#defaultIfNull-T-T-[defaultIfNull] and link:https://commons.apache.org/proper/commons-lang/apidocs/org/apache/commons/lang3/ObjectUtils.html#firstNonNull-T…-[firstNonNull] useful in some situations. Alternatively, write your own static helper methods to make it easier to handle potentially null values and make your code more readable.

[#java-running] == Build and run CDK applications

The {aws} CDK automatically compiles your app before running it. However, it can be useful to build your app manually to check for errors and to run tests. You can do this in your IDE (for example, press Control-B in Eclipse) or by issuing mvn compile at a command prompt while in your project’s root directory.

Run any tests you’ve written by running mvn test at a command prompt.