include::attributes.txt[]

// Page attributes

[.topic] [#toolkit-library-examples] = Advanced CDK Toolkit Library examples :info\_titleabbrev: Advanced examples :keywords: CDK Toolkit Library, Examples, {aws} CDK, {aws} Cloud Development Kit ({aws} CDK), deploy, {aws} CloudFormation, Infrastructure as Code

== [abstract]

## Learn how to use advanced features of the {aws} CDK Toolkit Library through practical examples. This guide provides detailed code samples for error handling, deployment monitoring, and cloud assembly management that build upon the basic concepts covered in other sections.

// Content start

Learn how to use advanced features of the {aws} CDK Toolkit Library through practical examples. This guide provides detailed code samples for error handling, deployment monitoring, and cloud assembly management that build upon the basic concepts covered in other sections.

[#toolkit-library-examples-integration] == Integrating features

The following example demonstrates how to combine cloud assembly sources, custom io host implementation, and deployment options:

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

import { Toolkit, StackSelectionStrategy, IIoHost } from ‘@aws-cdk/toolkit-lib’;

async function deployApplication(appPath, environment, options = {}) { // Create toolkit with custom message handling const toolkit = new Toolkit({ ioHost: { notify: async (msg) => { // Add environment to all messages console.log(+[${environment}][${msg.time}] ${msg.level}: ${msg.message}+); }, requestResponse: async (msg) => { // In production environments, use default responses if (environment === ‘production’) { console.log(Auto-approving for production: ${msg.message}); return msg.defaultResponse; }

// For other environments, implement custom approval logic  
 return promptForApproval(msg);

} } as IIoHost });

try { // Create cloud assembly source from the CDK app console.log(+Creating cloud assembly source from ${appPath}+); const cloudAssemblySource = await toolkit.fromCdkApp(appPath);

…. // Synthesize the cloud assembly console.log(Synthesizing cloud assembly); const cloudAssembly = await toolkit.synth(cloudAssemblySource);

try { // Deploy with environment-specific options console.log(Deploying to ${environment} environment); return await toolkit.deploy(cloudAssembly, { stacks: options.stacks || { strategy: StackSelectionStrategy.ALL\_STACKS }, parameters: options.parameters || {}, tags: { Environment: environment, DeployedBy: ‘CDK-Toolkit-Library’, DeployTime: new Date().toISOString() } }); } finally { // Always dispose when done await cloudAssembly.dispose(); } } catch (error) { console.error(Deployment to ${environment} failed:, error); throw error; } } ….

// Example usage await deployApplication(‘ts-node app.ts’, ‘staging’, { parameters: { MyStack: { InstanceType: ‘t3.small’ } } }); —

[#toolkit-library-examples-progress] == Tracking deployment progress

Track deployment progress with detailed status updates:

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

import { Toolkit, StackSelectionStrategy, IIoHost } from ‘@aws-cdk/toolkit-lib’;

// Create a progress tracker class DeploymentTracker { private startTime: Date; private resources = new Map<string, string>();

constructor() { this.startTime = new Date(); }

onStackEvent(stackName: string, event: string, timestamp: string) { // Calculate elapsed time if needed, or use the provided timestamp const elapsed = (new Date().getTime() - this.startTime.getTime()) / 1000; console.log(+[${timestamp}] (${elapsed.toFixed(1)}s elapsed) Stack ${stackName}: ${event}+); }

onResourceEvent(resourceId: string, status: string) { this.resources.set(resourceId, status); this.printProgress(); }

private printProgress() { console.log(‘Status:’); for (const [id, status] of this.resources.entries()) { console.log(+- ${id}: ${status}+); } console.log(); } }

// Use the tracker with the toolkit const tracker = new DeploymentTracker(); const toolkit = new Toolkit({ ioHost: { notify: async (msg) => { if (msg.code.startsWith(‘CDK\_DEPLOY’)) { // Track deployment events if (msg.data && ‘stackName’ in msg.data) { tracker.onStackEvent(msg.data.stackName, msg.message, msg.time); } } else if (msg.code.startsWith(‘CDK\_RESOURCE’)) { // Track resource events if (msg.data && ‘resourceId’ in msg.data) { tracker.onResourceEvent(msg.data.resourceId, msg.message); } } } } as IIoHost });

// Example usage with progress tracking async function deployWithTracking(cloudAssemblySource: any) { try { // Synthesize the cloud assembly const cloudAssembly = await toolkit.synth(cloudAssemblySource);

try { // Deploy using the cloud assembly await toolkit.deploy(cloudAssembly, { stacks: { strategy: StackSelectionStrategy.ALL\_STACKS } }); } finally { // Always dispose when done await cloudAssembly.dispose(); } } catch (error) { // Display the error message console.error(“Operation failed:”, error.message); throw error; } } —-

[#toolkit-library-examples-error] == Handling errors with recovery

Implement robust error handling with recovery strategies:

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

import { Toolkit, ToolkitError, StackSelectionStrategy } from ‘@aws-cdk/toolkit-lib’;

async function deployWithRetry(toolkit: Toolkit, cloudAssemblySource: any) { try { // Synthesize the cloud assembly const cloudAssembly = await toolkit.synth(cloudAssemblySource);

try { // Deploy using the cloud assembly await toolkit.deploy(cloudAssembly, { stacks: { strategy: StackSelectionStrategy.ALL\_STACKS } }); } finally { // Always dispose when done await cloudAssembly.dispose(); } } catch (error) { // Simply show the error to the user console.error(“Operation failed:”, error.message); throw error; } }

// Example usage try { await deployWithRetry(toolkit, cloudAssemblySource); } catch (error) { console.error(“Operation failed:”, error.message); process.exit(1); } —

[#toolkit-library-examples-cicd] == Integrating with CI/CD pipelines

Integrate the CDK Toolkit Library into a CI/CD pipeline:

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

import { Toolkit, StackSelectionStrategy, IIoHost } from ‘@aws-cdk/toolkit-lib’; import \* as fs from ‘fs’; import \* as path from ‘path’;

async function cicdDeploy() { // Create a non-interactive toolkit for CI/CD environments const toolkit = new Toolkit({ ioHost: { notify: async (msg) => { // Write to both console and log file const logMessage = ${msg.time} [${msg.level}] ${msg.message}; console.log(logMessage);

// Append to deployment log file  
 fs.appendFileSync('deployment.log', logMessage + '\n');

}, requestResponse: async (msg) => { // Always use default responses in CI/CD console.log(Auto-responding to: ${msg.message} with: ${msg.defaultResponse}); return msg.defaultResponse; } } as IIoHost });

// Determine environment from CI/CD variables const environment = process.env.DEPLOYMENT\_ENV || ‘development’;

// Load environment-specific parameters const paramsPath = path.join(process.cwd(), +params.${environment}.json+); const parameters = fs.existsSync(paramsPath) ? JSON.parse(fs.readFileSync(paramsPath, ‘utf8’)) : {};

try { // Use pre-synthesized cloud assembly from build step const cloudAssemblySource = await toolkit.fromAssemblyDirectory(‘cdk.out’);

…. // Synthesize the cloud assembly const cloudAssembly = await toolkit.synth(cloudAssemblySource);

try { // Deploy with CI/CD specific options const result = await toolkit.deploy(cloudAssembly, { stacks: { strategy: StackSelectionStrategy.ALL\_STACKS }, parameters, tags: { Environment: environment, BuildId: process.env.BUILD\_ID || ‘unknown’, CommitHash: process.env.COMMIT\_HASH || ‘unknown’ } });

// Write outputs to a file for subsequent pipeline steps fs.writeFileSync( ‘stack-outputs.json’, JSON.stringify(result.outputs, null, 2) );

return result; } finally { // Always dispose when done await cloudAssembly.dispose(); } } catch (error) { // Display the error message console.error(“Operation failed:”, error.message); process.exit(1); } } ….

// Run the CI/CD deployment cicdDeploy().then(() => { console.log(‘CI/CD deployment completed successfully’); }); —

[#toolkit-library-examples-resources] == Additional resources

For more detailed information on specific components used in these examples, refer to:

* xref:toolkit-library-configure-ca[Managing cloud assembly sources] - Learn how to create and manage cloud assembly sources.
* xref:toolkit-library-configure-messages[Configuring messages and interactions] - Detailed guide on customizing the IIoHost interface.