AWS CDK Infrastructure in code

App code constructs → `cdk synth` → **cdk.out** cloud assembly → `cdk deploy` → **CloudFormation** change set → creates → **AWS resources** per stack

CONCEPTS

The CLI runs your app (the command in cdk.json) to produce CloudFormation templates; deploy then hands them to CloudFormation.

Bootstrap each account/region pair once before the first deploy that uses assets.

Quote stack wildcards so the shell does not expand them: cdk deploy "Prod/*".

PROJECT

`cdk init app --language <lang>`
Create an empty app here

`cdk init --list`
List available templates

`cdk init sample-app --language <lang>`
App with an example queue and topic

`cdk doctor`
Check the local setup

`cdk --version`
Show the CLI version

PROJECT (cont.)

`npx aws-cdk <command>`
Run the project-local CLI

STACKS

`cdk ls`
List stack IDs

`cdk ls --long`
Show stack names and environments

`cdk deploy <stack>`
Deploy a single stack

`cdk deploy --all`
Deploy every stack

`cdk deploy "Prod/*"`
Deploy stacks matching a pattern

SYNTHESIZE & DIFF

`cdk synth`
Print the CloudFormation template

`cdk synth <stack> --json`
Template as JSON

`cdk synth -q`
Synthesize without printing

`cdk diff`
Compare code with the deployed stack

`cdk diff --fail`
Exit 1 if anything differs

DEPLOY

`cdk deploy`
Deploy the only stack

`cdk deploy --require-approval never`
Skip the security prompt

`cdk deploy --outputs-file <file>`
Save stack outputs as JSON

`cdk deploy --parameters <Name>=<value>`
Set a CloudFormation parameter

`cdk deploy -c <key>=<value>`
Pass a context value

`cdk deploy --no-rollback`
Keep resources after a failed deploy

`cdk deploy --hotswap`
Update Lambda and ECS directly

`cdk deploy --profile <name>`
Use a named AWS profile

`cdk deploy -y`
Answer prompts with yes

WATCH & DESTROY

`cdk watch`
Redeploy on file changes

WATCH & DESTROY (cont.)

`cdk destroy`
Delete the only stack

`cdk destroy --all`
Delete every stack

BOOTSTRAP

`cdk bootstrap`
Bootstrap the app's environments

`cdk bootstrap aws://<account>/<region>`
Bootstrap a specific environment

`cdk bootstrap --profile <name>`
Bootstrap using a profile

CONTEXT & CONFIG

`cdk context`
List cached context values

`cdk context --reset <key>`
Drop one cached value

`cdk --context <key>=<value>`
Set context for one run

`cdk --profile <name> --region <region>`
Pick credentials and region

CONTEXT & CONFIG (cont.)

cdk.json keys

Key	Purpose
`app`	Command that runs your app
`build`	Pre-synth build command
`watch`	include / exclude globs for cdk watch
`context`	Fixed context values and feature flags
`requireApproval`	Default approval level
`output`	Synth folder (default cdk.out)
`profile`	Default AWS profile
`outputsFile`	Default for --outputs-file

ADOPT & INSPECT

`cdk import <stack>`
Bring existing resources into a stack

RECIPES

Safe deploy pipeline

```
cdk synth -q
cdk diff --fail
cdk deploy --require-approval never --outputs-file out.json
```