AWS CDK Infrastructure in code

START HERE

`cdk init app --language typescript`
New project

`cdk bootstrap`
Prepare account/region

`cdk ls`
List stacks

`cdk synth`
Print template

`cdk diff`
Preview changes

`cdk deploy`
Create or update

`cdk watch`
Deploy on save

`cdk destroy`
Delete the stack

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
Run in an empty directory. Languages include typescript, python, java, go, csharp.

`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

`npx aws-cdk <command>`
Run the project-local CLI
Uses the version pinned in package.json instead of a global install.

STACKS

`cdk ls`
List stack IDs
Add -l for account and region, -d for dependencies.

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

`cdk ls --show-dependencies`
Show stack dependencies

`cdk deploy <stack>`
Deploy a single stack

`cdk deploy --all`
Deploy every stack

`cdk deploy "Prod/*"`
Deploy stacks matching a pattern
*? matches one character, * any number, ** a whole hierarchy.*

`cdk deploy -e <stack>`
Skip the stack's dependencies

SYNTHESIZE & DIFF

`cdk synth`
Print the CloudFormation template
Needs a stack name when the app has more than one.

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

`cdk synth -q`
Synthesize without printing

`cdk synth -o <dir>`
Write output to another folder

`cdk diff`
Compare code with the deployed stack

`cdk diff <stack> --template <file>`
Compare with a saved template

`cdk diff --security-only`
Only show security-relevant changes

`cdk diff --fail`
Exit 1 if anything differs
Useful as a CI gate; --fail-on broadening narrows it to security changes.

DEPLOY

`cdk deploy`
Deploy the only stack

`cdk deploy --require-approval never`
Skip the security prompt
Other values: any-change, broadening (default), destructive.

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

`cdk deploy --parameters <Name>=<value>`
Set a CloudFormation parameter
Repeat the flag; prefix Stack: to target one stack.

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

`cdk deploy --no-rollback`
Keep resources after a failed deploy
Handy in development; the next deploy resumes from the failure.

`cdk deploy --hotswap`
Update Lambda and ECS directly
Dev only. Without a fallback flag it fails on non-swappable changes; --hotswap-fallback falls back to CloudFormation.

DEPLOY (cont.)

`cdk deploy --method prepare-change-set`
Create the change set, do not run it

`cdk deploy --concurrency <n>`
Deploy independent stacks in parallel

`cdk deploy -t <key>=<value>`
Tag the stack

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

`cdk deploy -y`
Answer prompts with yes

WATCH & DESTROY

`cdk watch`
Redeploy on file changes
Uses hotswap by default; add --no-hotswap for full deploys. Not for production.

`cdk deploy --watch`
Same as cdk watch

`cdk watch --logs`
Stream logs while watching

`cdk destroy`
Delete the only stack

`cdk destroy --all`
Delete every stack

`cdk destroy <stack> -f`
Delete without confirming
Resources with a retain policy survive the delete.

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

`cdk bootstrap --trust <account>`
Allow another account to deploy

`cdk bootstrap --cloudformation-execution-policies <arn>`
Limit what deploys may create
Default grants broad rights; restrict it for shared accounts.

`cdk bootstrap --qualifier <name>`
Use a custom resource prefix

CONTEXT & CONFIG

`cdk context`
List cached context values

`cdk context --reset <key>`
Drop one cached value
Key may also be its index from the list.

`cdk context --clear`
Drop all cached values

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

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

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
Needs a clean cdk diff and constructs matching the live resource.

RECIPES

Safe deploy pipeline

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

First deploy in a new account

```
aws sso login --profile dev
cdk bootstrap aws://<account>/
<region> --profile dev
cdk deploy --all --profile dev
```

Fast Lambda iteration

```
cdk deploy <stack>
cdk watch <stack>
# edit code: changes hotswap on
save
```

Per-environment context

```
# cdk.json
{
  "app": "npx ts-node bin/
app.ts",
  "context": { "stage": "dev" }
}
# override: cdk deploy -c
stage=prod
```