# Pulumi Infrastructure as code

**START HERE**

`pulumi new <template>`
Create a project

`pulumi config set <key> <value>`
Set a config value

`pulumi preview`
Show planned changes

`pulumi stack select <stack>`
Switch stack

`pulumi up`
Deploy the stack

`pulumi refresh`
Sync state with the cloud

`pulumi stack output`
Read stack outputs

`pulumi destroy`
Tear everything down

**Program** code per project → `preview` → **Engine** diff vs state → `up` → **Stack** one environment → `provider API` → **Cloud resources** real infrastructure

## CONCEPTS

Your program declares the desired resources in a real language. Pulumi diffs that against the stack's stored state and applies only the difference.

A project is the code (Pulumi.yaml); a stack is one deployed instance of it (dev, prod) with its own config and state (Pulumi.<stack>.yaml).

Preview before up on shared stacks, and use pulumi refresh if someone changed resources by hand.

## PROJECTS & DEPLOYS

`pulumi new <template>`
Create a project from a template
*List templates with pulumi new --list-templates (shows locally cached ones).*

`pulumi new <template> --yes`
Accept all defaults

`pulumi new --generate-only`
Scaffold only, no stack or install

`pulumi preview`
Show what up would change

`pulumi preview --diff`
Preview as a rich diff

`pulumi preview --expect-no-changes`
Fail if drift or changes exist

`pulumi up`
Preview, confirm, then deploy

`pulumi up --yes`
Deploy without the prompt
*Use in CI; pair with --skip-preview only if a prior step already previewed.*

`pulumi up --skip-preview`
Deploy without previewing first

`pulumi up --refresh`
Refresh state before deploying

`pulumi up --diff`
Show changes as a rich diff

`pulumi up --parallel <n>`
Limit concurrent operations

`pulumi up --continue-on-error`
Keep going after failures

`pulumi destroy`
Delete all resources in the stack

`pulumi destroy --remove`
Also remove the empty stack

## PROJECTS & DEPLOYS (cont.)

`pulumi destroy --preview-only`
Show what would be deleted

`pulumi destroy --exclude-protected`
Skip protected resources

`pulumi refresh`
Update state from the real cloud

`pulumi refresh --yes`
Refresh without the prompt

`pulumi watch`
Redeploy on file changes

## TARGETING

`pulumi up --target <urn>`
Update only these resources
*URNs accept wildcards. Use for exceptions; the rest of the stack is left as is.*

`pulumi up --target <urn> --target-dependents`
Include dependents too

`pulumi up --replace <urn>`
Force recreating a resource

`pulumi up --exclude <urn>`
Leave these resources alone

`pulumi up --target-replace <urn>`
Replace one resource, touch nothing else

`pulumi up --ignore-protect`
Allow changing protected resources

`pulumi stack --show-urns`
List resources with their URNs

## STACKS

`pulumi stack init <name>`
Create a stack

`pulumi stack ls`
List stacks

`pulumi stack select <name>`
Switch the current stack

`pulumi stack`
Show current stack and resources

`pulumi stack output`
Show all outputs

`pulumi stack output <name>`
Print one output value

`pulumi stack output --json`
Outputs as JSON

`pulumi stack output --show-secrets`
Reveal secret outputs

## STACKS (cont.)

`pulumi stack rm <name>`
Delete an empty stack
*Add --force to drop a stack that still has resources; they are not destroyed.*

`pulumi stack rename <new>`
Rename the current stack

`pulumi stack history`
Show past updates

`pulumi stack export > <file>`
Dump the stack state as JSON

`pulumi stack import --file <file>`
Load a state dump

`pulumi up --stack <name>`
Act on a stack without selecting it

## CONFIG & SECRETS

`pulumi config`
List config values

`pulumi config set <key> <value>`
Set a config value

`pulumi config set --secret <key> <value>`
Store a value encrypted

`pulumi config get <key>`
Read one value

`pulumi config rm <key>`
Remove a value

`pulumi config set --type int <key> <value>`
Store as bool, int or float

`pulumi config get <key> --show-secrets`
Decrypt a secret

`PULUMI_CONFIG_PASSPHRASE`
Unlock a passphrase secrets provider

## BACKENDS & STATE

`pulumi login`
Log in to Pulumi Cloud
*Reads PULUMI_ACCESS_TOKEN if set, otherwise prompts.*

`pulumi login --local`
Keep state on this machine

`pulumi login file://<path>`
Use a local directory as backend

`pulumi login s3://<bucket>`
Use an S3 bucket as backend

`pulumi state delete <urn>`
Drop a resource from state only
*The real resource is left running; add --target-dependents if others depend on it.*

## BACKENDS & STATE (cont.)

`pulumi cancel`
Release a stuck update lock

`pulumi whoami`
Show the logged-in user and backend

## IMPORT & LOGS

`pulumi import <type> <name> <id>`
Adopt an existing resource
*Prints generated code to paste into your program; add the same resource there afterwards.*

`pulumi import --file <file>`
Import many from a JSON list

`pulumi logs`
Show recent resource logs

`pulumi logs --follow`
Stream logs live

## GLOBAL FLAGS

`--non-interactive`
Never prompt

`--json`
Machine-readable output
*Supported by up, preview, destroy, import and stack output; not by every command.*

## RECIPES

**First deployment**

```
pulumi new aws-typescript
pulumi config set aws:region
eu-central-1
pulumi preview
pulumi up
pulumi stack output
```

**Add a prod stack**

```
pulumi stack init prod
pulumi config set --secret
dbPassword <pw>
pulumi up --stack prod
```

**Self-managed state in S3**

```
export
PULUMI_CONFIG_PASSPHRASE=<pass>
pulumi login s3://<bucket>
pulumi stack init dev
pulumi up --yes
```

**Adopt an existing resource**

```
pulumi import aws:s3/
bucket:Bucket my-bucket
<bucket-name>
# paste the generated code into
the program
pulumi preview   # expect no
changes
```

**CI deploy**

```
pulumi login
pulumi stack select <org>/
<project>/prod
pulumi preview --non-
interactive
pulumi up --yes --non-
interactive
```