# Open Policy Agent

Policy as code

**Policy** .rego files → `opa check / fmt` → **Tested** test_ rules → `opa test` → **Bundle** .tar.gz → `opa build` → **Server** opa run -s → `query` → **Decision** JSON result

## CONCEPTS

OPA evaluates a query against data (policies plus JSON) and an input document; the answer is JSON.

Rules are queried as data.<package>.<rule>; an undefined result prints an empty object, not false.

## EVAL

**opa eval -d** *<path>* **'***<query>***'**
Evaluate a query against policy and data

**opa eval -d** *<path>* **-i** *<input.json>* **'***<query>***'**
Supply an input document

**opa eval -d** *<path>* **-I '***<query>***'**
Read input from stdin

**opa eval -b** *<bundle>* **'***<query>***'**
Load a bundle instead of paths

**opa eval -f pretty '***<query>***'**
Human-readable output

**opa eval -f raw '***<query>***'**
Script-friendly value output

## EVAL (cont.)

**opa eval --fail '***<query>***'**
Exit non-zero if undefined or empty

**opa eval --fail-defined '***<query>***'**
Exit non-zero if the result is defined

**opa eval --explain=full '***<query>***'**
Trace how the query evaluated

## REPL AND SERVER

**opa run**
Start the interactive REPL

**opa run** *<path>*
REPL with policies and data loaded

**opa run -s** *<path>*
Start the HTTP server

**opa run -s -a** *<addr>*
Listen on another address

**opa run -s -b** *<bundle>*
Serve a bundle

**opa run -w** *<path>*
Reload on file changes

**opa run -s -c** *<config.yaml>*
Use a config file

## REPL AND SERVER (cont.)

The server answers POST /v1/data/<path> with a JSON body of the form {"input": {...}}.

## TESTING

**opa test** *<path>*
Run all test_ rules

**opa test -v** *<path>*
Verbose, one line per test

**opa test -r '***<regex>***'** *<path>*
Run matching tests only

**opa test -c** *<path>*
Report coverage

**opa test --threshold** *<pct>* **-c** *<path>*
Fail below a coverage percentage

**opa test -w** *<path>*
Rerun tests on file changes

**opa test -f json** *<path>*
Machine-readable results

## FORMAT AND CHECK

**opa fmt** *<path>*
Print formatted source

**opa fmt -w** *<path>*
Rewrite files in place

## FORMAT AND CHECK (cont.)

**opa fmt -d** *<path>*
Show a diff of changes

**opa fmt -l** *<path>*
List files that would change

**opa fmt --fail** *<path>*
Exit non-zero if files need formatting

**opa check** *<path>*
Parse and compile check

**opa check -S** *<path>*
Strict mode checks

## BUNDLES

**opa build** *<path>*
Build a bundle (bundle.tar.gz)

**opa build -b** *<dir>* **-o** *<file>*
Name the output file

**opa build -r** *<rev>* *<path>*
Set the bundle revision

**opa build -e** *<entrypoint>* *<path>*
Declare an entrypoint

**opa build -O=1 -e** *<entrypoint>* *<path>*
Optimize (needs an entrypoint)

## BUNDLES (cont.)

**opa build -t wasm -e** *<entrypoint>* *<path>*
Compile to WebAssembly

**opa inspect** *<bundle>*
Summarize a bundle

## OTHER COMMANDS

**opa exec -b** *<bundle>* *<input.json>*
Run the default decision on input files

**opa version**
Show version and build

Most commands take -d for data/policy paths and -b for bundles; both read directories recursively.

## RECIPES

**Gate a Kubernetes manifest in CI**

```
# deny rule must come back empty
opa eval --fail-defined -d policy/ -i deploy.json 'data.k8s.deny'
```

**Check, format and test before commit**

```
opa fmt --fail policy/
opa check -S policy/
opa test -c --threshold 80 policy/
```