Open Policy Agent Policy as code

CONCEPTS

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

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

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

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

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

REPL AND SERVER

`opa run`
Start the interactive REPL

REPL AND SERVER (cont.)

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

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

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

FORMAT AND CHECK

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

`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

BUNDLES

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

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

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

OTHER COMMANDS

`opa version`
Show version and build