rq: Datalog for your Shell Pipelines

Charles Daniels

September 14, 2022

About Me – Charles Daniels

- Computer Engineering B.S.E. – UofSC 2019
- Computer Science M.S. - UofSC 2021
- Backend Software Engineer – Styra Inc.

All views and opinions expressed are my own, and do not represent those of Styra. rq is not supported or endorsed by Styra.

Background: What is OPA

- Open Policy Agent (OPA) is a CNCF-graduated open-source runtime for policy-as-code
- Policies are written in a Datalog-based DSL: **Rego**
- Rego allows expressing arbitrary transforms over JSON documents
- Try it in your browser: https://play.openpolicyagent.org/

Open Policy Agent

Enter rq

- Problem: OPA has limited ability to use Rego in shell contexts
- rq (“Rego Query”) embeds OPA as a library, allows using Rego from pipelines and scripts
- Adds additional builtins to support this use case, as well as adapters for popular formats (CSV, JSON, YAML, TOML)
- https://git.sr.ht/~charles/rq

Sample Data (books.json)

```json
[
    {
        "title": "Writing An Interpreter In Go",
        "authors": ["Thorsten Ball"],
        "isbn": "978-3982016115",
        "year": 2018
    },
    {
        "title": "Writing A Compiler In Go",
        "authors": ["Thorsten Ball"],
        "isbn": "978-3982016108",
        "year": 2018
    },
    {
        "title": "The Go Programming Language",
        "authors": ["Alan A. A. Donovan", "Brian W. Kernighan"],
        "isbn": "978-0134190440",
        "year": 2015
    },
    {
        "title": "Hands-On GUI Application Development in Go",
        "authors": ["Andrew Williams"],
        "isbn": "978-1789138412",
        "year": 2019
    },
    {
        "title": "Building Cross-Platform GUI Applications with Fyne",
        "authors": ["Andrew Williams"],
        "isbn": "1800563167",
        "year": 2021
    }
]
```

Some Basic Interactions with rq (1/3)

```
$ rq 'input[0]' < books.json
{
    "authors": [
        "Thorsten Ball"
    ],
    "isbn": "978-3982016115",
    "title": "Writing An Interpreter In Go",
    "year": 2018
}
```

Some Basic Interactions with rq (2/3)

```
$ rq 'input[0].title' < books.json
"Writing An Interpreter In Go"
$ rq 'object.filter(input[0], ["title", "year"])' < books.json
{
    "title": "Writing An Interpreter In Go",
    "year": 2018
}
```

Some Basic Interactions with rq (3/3)

```
$ rq '{b.title | b := input[_]; b.year > 2018}' < books.json
[
    "Building Cross-Platform GUI Applications with Fyne",
    "Hands-On GUI Application Development in Go"
]
$ rq '{b.title | b := input[_]; count(b.authors) > 1}' < books.json
[
    "The Go Programming Language"
]
```

Scripting With rq

```
#!/usr/bin/env rq

# rq: data-paths ./books.json
# rq: query data.script.output
# rq: output-format csv
# rq: silent-query data.script.check_args

args := rq.args()

author_search := args[0]
```

```
check_args {
	count(args) != 1
	rq.error("usage: byauthor.rego AUTHOR_FRAGMENT")
}

output := {
	object.filter(b, ["title", "isbn", "year"])
	|
	b := data.books[_]
	contains(lower(b.authors[_]), lower(author_search))
}
```

```
$ ./byauthor.rego andrew
isbn,title,year
1800563167,Building Cross-Platform GUI Applications with Fyne,2021
978-1789138412,Hands-On GUI Application Development in Go,2019
```

… and so Much More!

- This is just a few samples of what rq can do
- More features not shown
  - Syntax highlighted output
  - rq.run()
  - rq.env()
  - --data
  - … and others

End.

- Questions?

Further Reading

- OPA – Policy Language
  - https://www.openpolicyagent.org/docs/latest/policy-language/
- OPA – Policy Reference
  - https://www.openpolicyagent.org/docs/latest/policy-reference/
- Styra Academy (free, requires signup)
  - https://academy.styra.com/
- rq Documentation
  - https://git.sr.ht/~charles/rq/tree/master/item/doc/README.md
- rq Repository
  - https://git.sr.ht/~charles/rq