JSON Patch Meets Terraform to Reveal the Sensitives

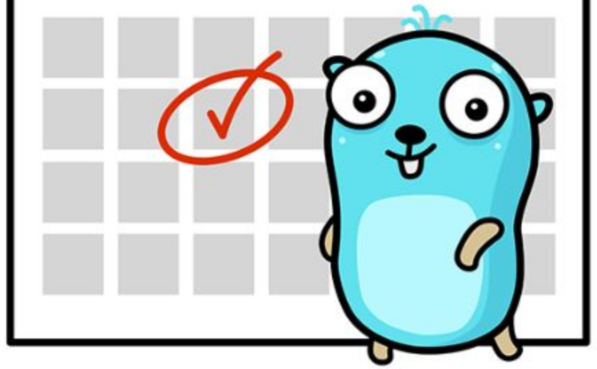
Bärner Go Meetup – 13.06.2024
Lucas Bremgartner

Hello!

I’m Lucas

lucas@bremis.ch | github.com/breml | linkedin.com/in/lucas-bremgartner/

1. The Problem™
2. Solution Exploration
3. JSON Patch
4. Go Nuggets
5. Wrap-up



The Problem™

1. Terraform for Infrastructure as Code
2. Cloud Infrastructure (Elastic Cloud)
3. Updates through central registry causes drift in Terraform
4. Drift causes changes in JSON values that may contain secrets
5. Sensitive values are hidden by Terraform
6. We are blind when applying changes

Bottom line: **reveal sensitive values in Terraform plan**.

Generalized Example

```
package main

import "fmt"

func main() {
  a := make([]int, 5)
  printSlice("a", a)
b := make([]int, 0, 5)
```

Solution Approaches

- Search Google ❌
- Use JSON plan from Terraform ❌
- Use Terraform `output` with `nonsensitive()` ❌
- Change Terraform ❌
- Change OpenTofu ❌
- Shell Script (`terraform plan` + `jq` + `diff`) ❌

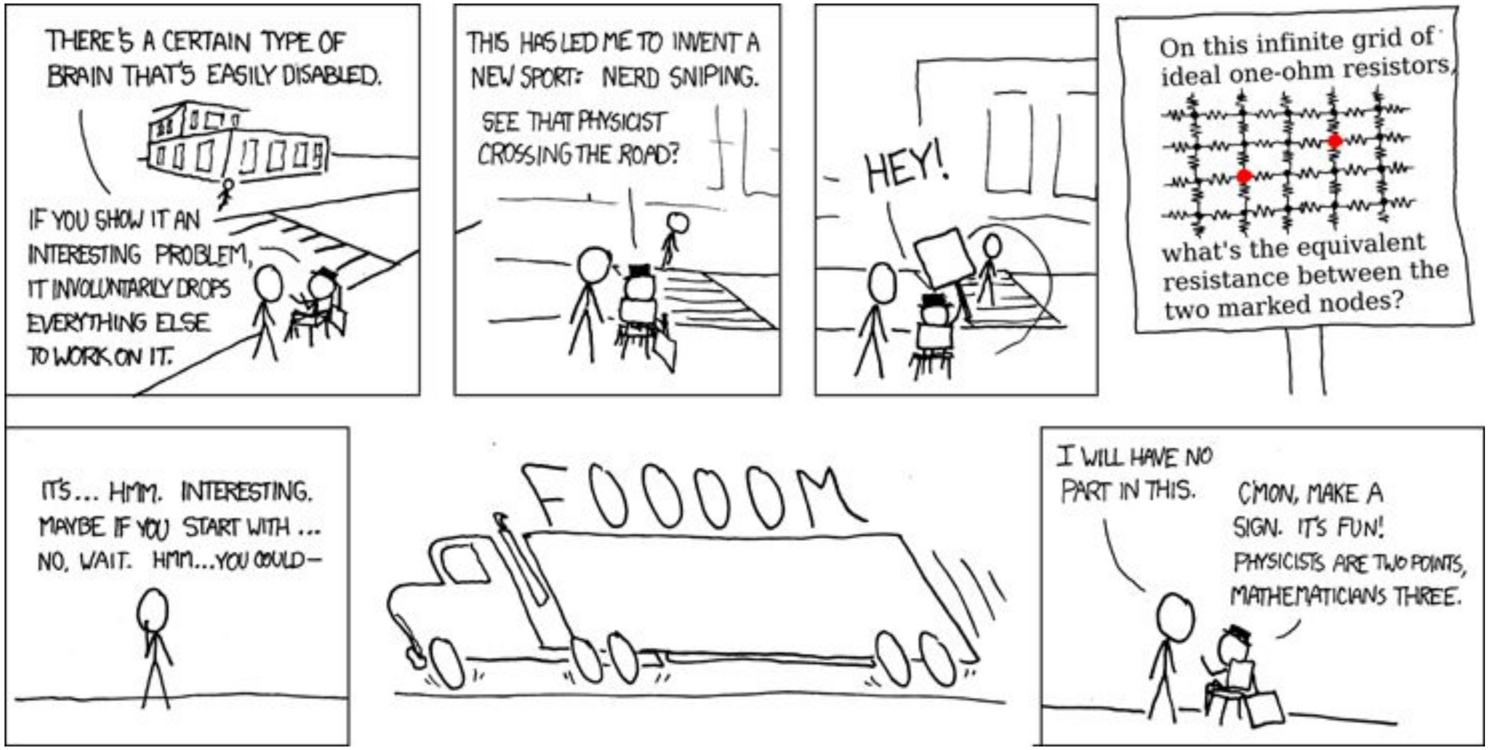
Generalized Example – Part 2

```
package main

import "fmt"

func main() {
  a := make([]int, 5)
  printSlice("a", a)
  b := make([]int, 0, 5)
```

Solution Approaches

- Search Google ❌
- Use JSON plan from Terraform ❌
- Use Terraform `output` with `nonsensitive()` ❌
- Change Terraform ❌
- Change OpenTofu ❌
- Shell Script (`terraform plan` + `jq` + `diff`) ❌
- DIY (Do it yourself)

How hard can it possibly be? – Getting “Nerd sniped”


THERE'S A CERTAIN TYPE OF BRAIN THAT'S EASILY DISABLED.
IF YOU SHOW IT AN INTERESTING PROBLEM, IT INVOLUNTARILY DROPS EVERYTHING ELSE TO WORK ON IT.
THIS HAS LED ME TO INVENT A NEW SPORT: NERD SNIPING.
SEE THAT PHYSICIST CROSSING THE ROAD?
HEY!
On this infinite grid of ideal one-ohm resistors, what's the equivalent resistance between the two marked nodes?
IT'S... HMM. INTERESTING. MAYBE IF YOU START WITH ... NO, WAIT. HMM...YOU COULD–
FOOOOM
I WILL HAVE NO PART IN THIS.
C'MON, MAKE A SIGN. IT'S FUN! PHYSICISTS ARE TWO POINTS, MATHEMATICIANS THREE.

Source: https://xkcd.com/356/

The Plan™

- **Run Terraform plan:**
  ```
  terraform plan -out plan.out
  ```

- **Export plan as JSON:**
  ```
  terraform show -json plan.out > plan.json
  ```

- **Use The Thing™ to show the Terraform plan including the sensitive values:**
  ```
  tfreveal plan.json
  ```

Idea for tfreveal & jsondiffprinter was born.

Semantic Difference of JSON Documents

Formatting – prettyfied vs. minified

```
{
  "baz": "qux",          {"baz":"qux","
  "foo": "bar"       =   foo":"bar"}
}
```

Key Order in Objects

```
{                          {
  "baz": "qux",              "foo": "bar",
  "foo": "bar"         =     "baz": "qux"
}                          }
```

Numbers

```
1 == 1.0 == 1.00 == 1e0 == 1e-0 == 1e+0 == 1e00 == 1E0
```

String Escaping

```
"a" == "\u0061"
"/" == "\/" == "\u002f" == "\u002F"
```

Diff JSON

Implement the Diff algorithm for JSON myself? 🤔

JSON Patch Libraries – Usage / Maintenance

Github User	JSON Patch	⭐	⑂	👁️	Last Commit	Imported by:	Coverage
yudai	No	> 500	~ 80	7	6 years ago	> 300	0.0%[1]
cameront	Yes	< 20	< 10	3	6 years ago	< 5	82.5%[1]
herkyl	Yes	< 20	< 10	3	5 years ago	< 5	89.1%[1]
mattbaird	Yes	> 100	~ 50	7	5 months ago	> 400	77.8%[1]
MianXiang	Yes	< 20	< 10	2	3 years ago	< 5	80.5%
snorwin	Yes	< 20	< 10	2	5 days ago	~ 15	92.0%[2]
VictorLowther	Yes	< 20	< 10	2	4 years ago	~ 35	84.9%
wI2L	Yes	> 400	~ 40	4	2 months ago	~ 80	97.3%

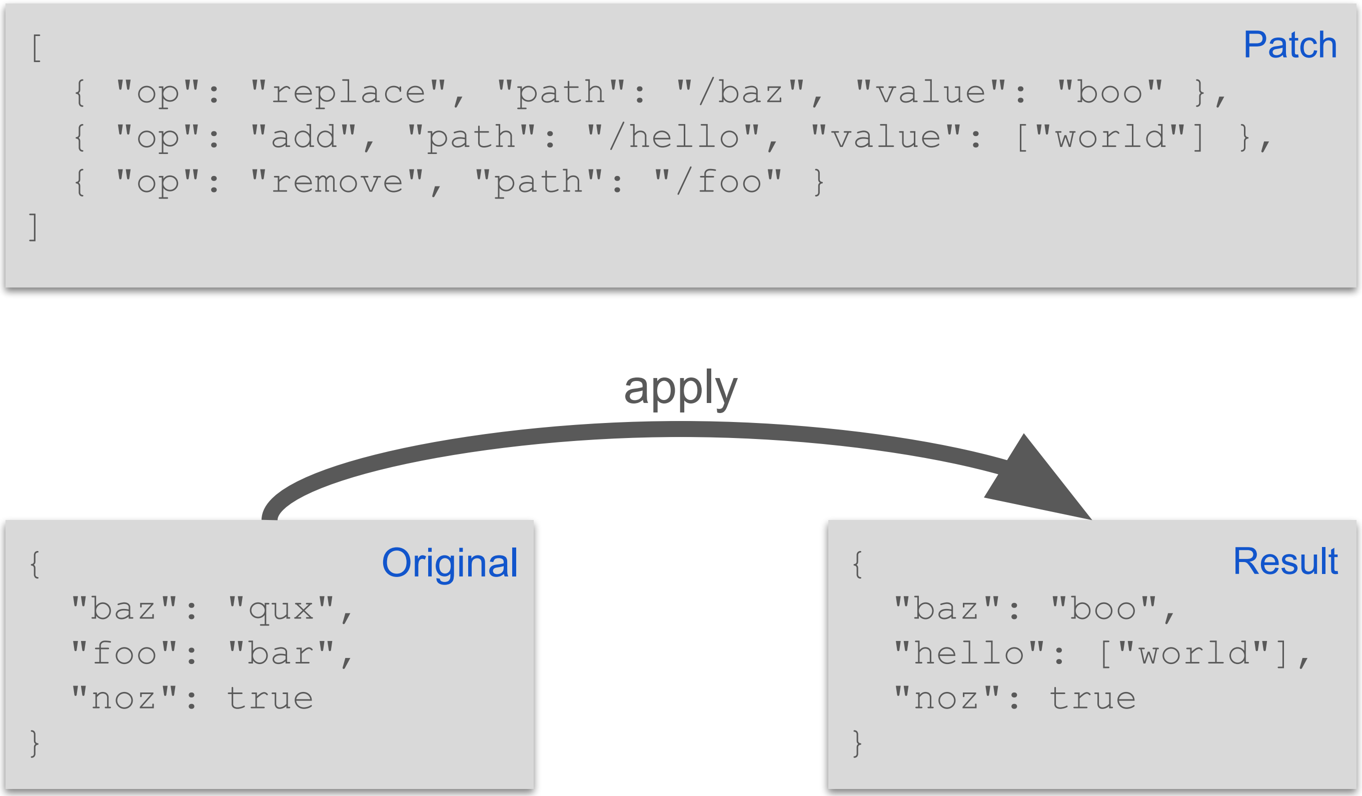
1) Not yet a Go Module
2) Tests extremely slow

JSON Patch Libraries – Capabilities

Github User	the basics	array (change + add)	array (remove)	array (items type change)	items type change	root ≠ object	very large number
yudai	partially OK	wrong	OK	wrong	OK	error	error
cameront	partially OK	OK (LCS)	OK	panic	panic	error	error
herkyl	OK	full replace	full replace	full replace	full replace	OK	error
mattbaird	OK	OK (LCS)	OK	panic	OK	OK	error
MianXiang	OK	OK (LCS)	wrong	remove+add	OK	OK	error
snorwin	OK	item by item	item by item	error	error	OK	error
VictorLowther	OK	full replace	full replace	full replace	OK	OK	error
wI2L	OK	item by item	item by item	OK	OK	OK	error
diff	partially OK	item by item	OK	OK	full replace	OK	partially OK

the basics: whitespaces (prettify vs. minify), unsorted keys in objects, unicode strings & escaping, numbers, nested arrays
diff: semantic free text compare with `diff -u` on preprocessed files with `jq .` (keys sorted, prettify-ed).

JSON Patch



Patch

```
[
  { "op": "replace", "path": "/baz", "value": "boo" },
  { "op": "add", "path": "/hello", "value": ["world"] },
  { "op": "remove", "path": "/foo" }
]
```

Original

```
{
  "baz": "qux",
  "foo": "bar",
  "noz": true
}
```

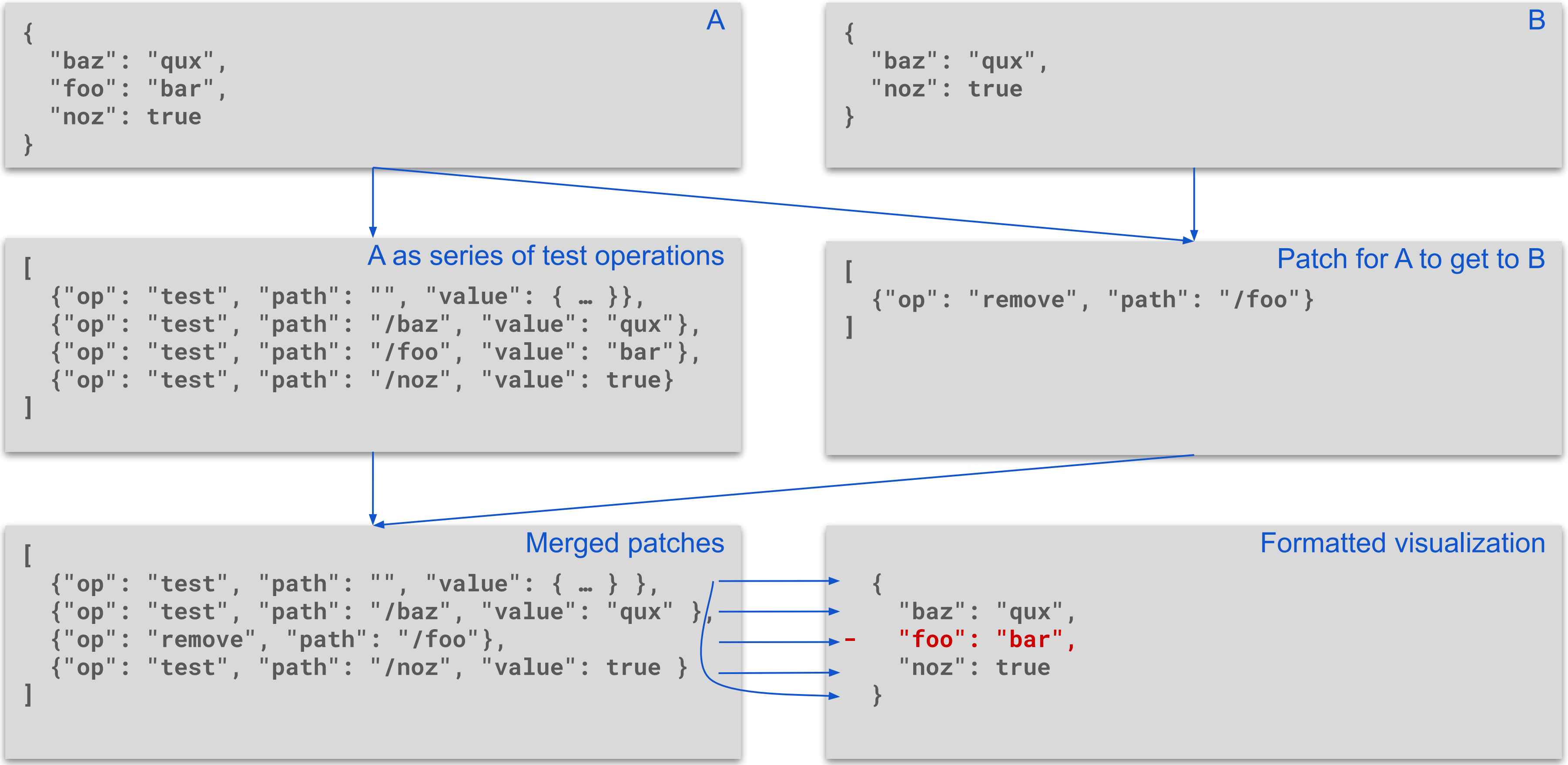
Result

```
{
  "baz": "boo",
  "hello": ["world"],
  "noz": true
}
```

Operations:

- add
- remove
- replace
- move
- copy
- test

Idea behind jsondiffprinter



```
{
  "baz": "qux",
  "foo": "bar",
  "noz": true
}
```

```
{
  "baz": "qux",
  "noz": true
}
```

```
[
  {"op": "test", "path": "", "value": { … }},
  {"op": "test", "path": "/baz", "value": "qux"},
  {"op": "test", "path": "/foo", "value": "bar"},
  {"op": "test", "path": "/noz", "value": true}
]
```

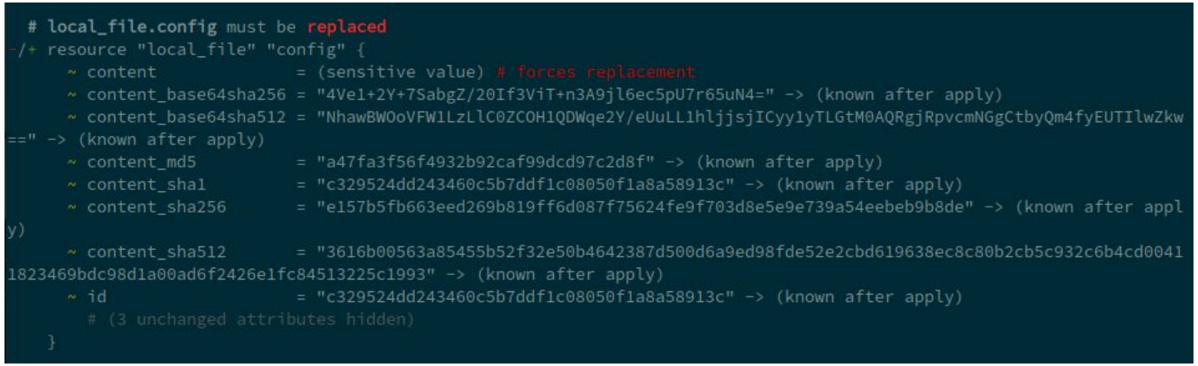
```
[
  {"op": "remove", "path": "/foo"}
]
```

```
[
  {"op": "test", "path": "", "value": { … } },
  {"op": "test", "path": "/baz", "value": "qux" },
  {"op": "remove", "path": "/foo"},
  {"op": "test", "path": "/noz", "value": true }
]
```

```
   {
     "baz": "qux",
-    "foo": "bar",
     "noz": true
   }
```

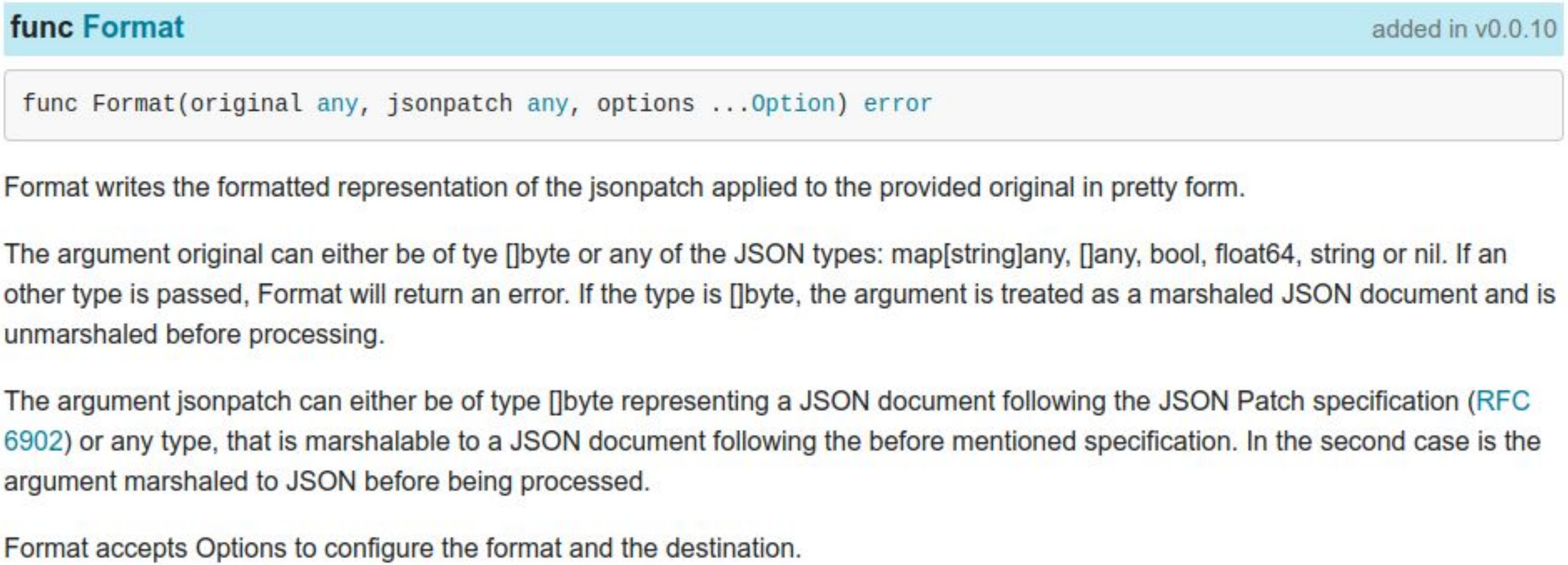
jsondiffprinter – Design Goals

- Focus: JSON diff pretty printing
- API based on a standard (JSON Patch, RFC 6902)
- Minimal external dependencies for Library
- Support Terraform formatting
  - “(known after apply)”
  - “# forces replacement” comments
  - Single line replace (instead of add + remove)
  - Indented diff markers
  - Collapse unchanged attributes



```
  # local_file.config must be replaced
-/+ resource "local_file" "config" {
      ~ content              = (sensitive value) # forces replacement
      ~ content_base64sha256 = "4Ve1+2Y+7SabgZ/20If3ViT+n3A9jl6ec5pU7r65uN4=" -> (known after apply)
      ~ content_base64sha512 = "NhawBWOoVFW1LzLlC0ZCOH1QDWqe2Y/eUuLL1hljjsjICyy1yTLGtM0AQRgjRpvcmNGgCtbyQm4fyEUTIlwZkw
==" -> (known after apply)
      ~ content_md5          = "a47fa3f56f4932b92caf99dcd97c2d8f" -> (known after apply)
      ~ content_sha1         = "c329524dd243460c5b7ddf1c08050f1a8a58913c" -> (known after apply)
      ~ content_sha256       = "e157b5fb663eed269b819ff6d087f75624fe9f703d8e5e9e739a54eebeb9b8de" -> (known after appl
y)
      ~ content_sha512       = "3616b00563a85455b52f32e50b4642387d500d6a9ed98fde52e2cbd619638ec8c80b2cb5c932c6b4cd0041
1823469bdc98d1a00ad6f2426e1fc84513225c1993" -> (known after apply)
      ~ id                   = "c329524dd243460c5b7ddf1c08050f1a8a58913c" -> (known after apply)
        # (3 unchanged attributes hidden)
    }
```

API



func Format added in v0.0.10

```
func Format(original any, jsonpatch any, options ...Option) error
```

Format writes the formatted representation of the jsonpatch applied to the provided original in pretty form.

The argument original can either be of tye []byte or any of the JSON types: map[string]any, []any, bool, float64, string or nil. If an other type is passed, Format will return an error. If the type is []byte, the argument is treated as a marshaled JSON document and is unmarshaled before processing.

The argument jsonpatch can either be of type []byte representing a JSON document following the JSON Patch specification (RFC 6902) or any type, that is marshalable to a JSON document following the before mentioned specification. In the second case is the argument marshaled to JSON before being processed.

Format accepts Options to configure the format and the destination.

Multiple Modules in one Repository – Module Workspaces

Provide jsondiffprinter as library and jd as command from the same repo while keeping the dependencies for the library minimal.

go.mod

```
module github.com/breml/jsondiffprinter

go 1.22.3

require golang.org/x/tools v0.21.0
```

go.work

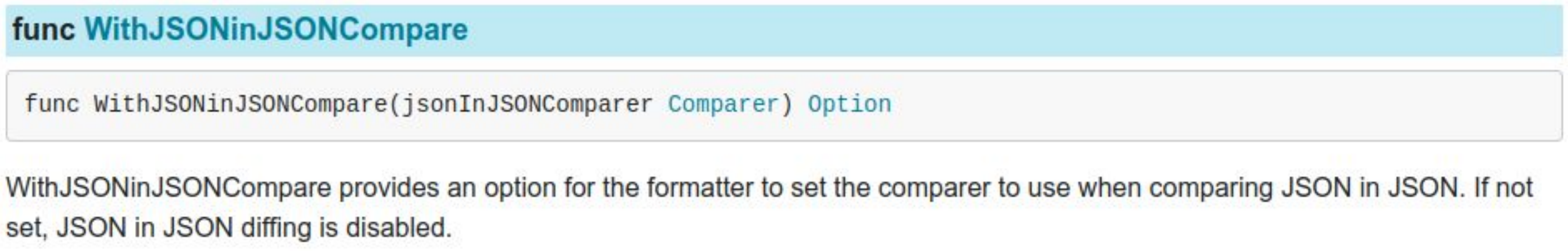
```
go 1.22.3

use (
    .
    ./cmd
)
```

```
.
├── cmd
│   ├── go.mod
│   ├── go.sum
│   └── jd
│       └── main.go
├── formatter.go
├── go.mod
├── go.sum
├── go.work
└── go.work.sum
```

cmd/go.mod

```
module github.com/breml/jsondiffprinter/cmd

go 1.22.3

require (
  github.com/breml/jsondiffprinter v0.0.8
  …
)

replace github.com/breml/jsondiffprinter => ../
```

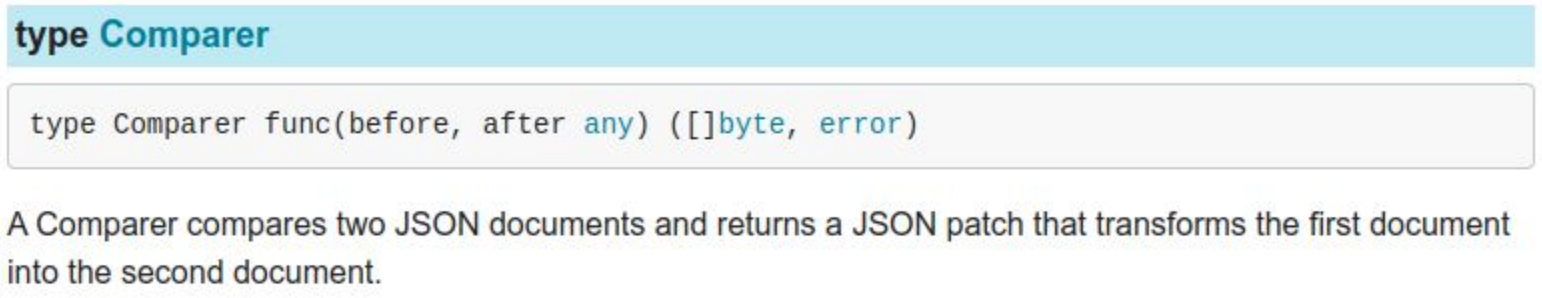
API – Part 2



func WithJSONinJSONCompare

```
func WithJSONinJSONCompare(jsonInJSONComparer Comparer) Option
```

WithJSONinJSONCompare provides an option for the formatter to set the comparer to use when comparing JSON in JSON. If not set, JSON in JSON diffing is disabled.



type Comparer

```
type Comparer func(before, after any) ([]byte, error)
```

A Comparer compares two JSON documents and returns a JSON patch that transforms the first document into the second document.

Testing: golang.org/x/tools/txtar

```go
func TestTxtar(t *testing.T) {
 files, _ :=
filepath.Glob("testdata/*.txtar")

 for _, filename := range files {
   t.Run(filename, func(t *testing.T) {
     ta, _ := txtar.ParseFile(filename)

     t.Log(ta.Comment)

     for _, file := range ta.Files {
       t.Log(file.Name, string(file.Data))
     }
   })
 }
}
```

```
This ist the comment section,
-- before.json --
"<null>"
-- after.json --
"<foobar>"
-- diff.json --
- "<null>"
+ "<foobar>"
-- diff.tf --
  "<null>" -> "<foobar>"
```

Pro Tip: txtar is sensitive to line endings, add txtar files to `.gitattributes` **to make sure they are valid on Windows.**

Testing: text/tabwriter

```go
func withTabwriter(want, got string) string {
 want = strings.ReplaceAll(want, " ", "·")
 got = strings.ReplaceAll(got, " ", "·")

 buf := bytes.NewBufferString("\n")
 w := tabwriter.NewWriter(buf, 0, 0, 3, ' ', 0)
 fmt.Fprintln(w, "want:\tgot:\n=====\t====\n\t")

 wantLines := strings.Split(want, "\n")
 gotLines := strings.Split(got, "\n")

 // Make slices the same length, omitted for brevity
 …

 for i := 0; i < len(wantLines); i++ {
   fmt.Fprintf(w, "%s\t%s\n", wantLines[i], gotLines[i])
 }
 w.Flush()
 return buf.String()
}
```

```
want:                 got:
=====                 ====

··{                   ··{
····"baz": "qux",     ····"baz": "qux",
-···"foo": "bar",     -···"foo": "bar",
····"noz": false      ····"noz": true
··}                   ··}
```

Pro Tip: make spaces visible with UTF-8 middle dot.

jsondiffprinter & tfreveal

Current state

- Prettyprint Terraform Plan ✅
- Compatibility with several 3rd Party JSON Patch packages ✅
- Support for embedded JSON (JSON in JSON) ✅
- Documentation 😓
- Code quality 😓

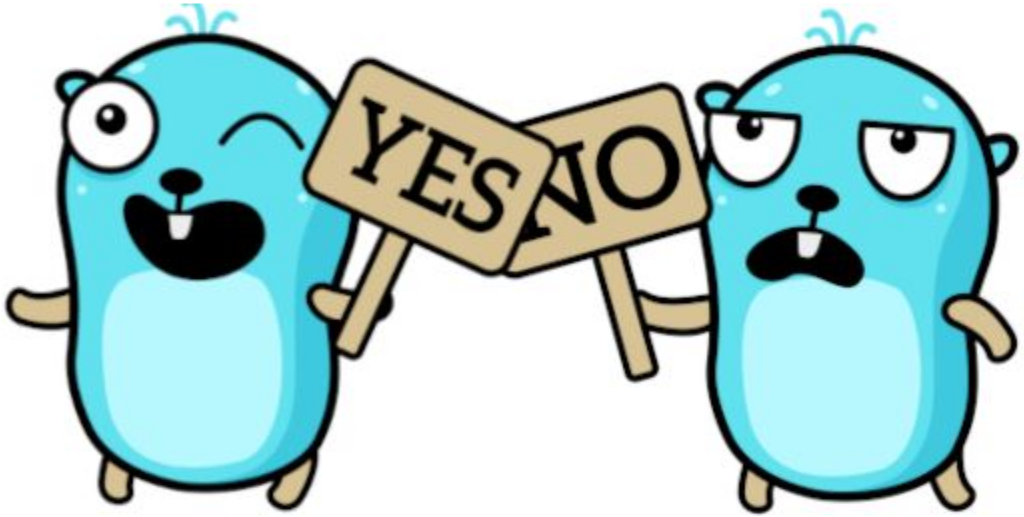
Future Ideas:

- Support detailed diff for long (multi-line) string
- Use tabwriter to print diffs side-by-side
- Go template based formatter (e.g. for HTML output)

My Takeaways

- A little copying is better than a little dependency. (Go Proverb)
- Embrace standards for flexible API
- (JSON Patch) makes the API flexible
- 1 Repo, multiple Go Modules
- Multi-Module Workspaces
- golang.org/x/tools/txtar for tests
  - Windows file ending!
- text/tabwriter: side-by-side output, make spaces visible

Questions



Thank you

Check out github.com/breml/jsondiffprinter and github.com/breml/tfreveal and give a ⭐.

Links

JSON Diff Printer: https://github.com/breml/jsondiffprinter

tfreveal: https://github.com/breml/tfreveal

JSON Patch: https://jsonpatch.com/, https://datatracker.ietf.org/doc/html/rfc6902/

JSON Pointer: https://datatracker.ietf.org/doc/html/rfc6901/

Multi-module workspaces: https://go.dev/doc/tutorial/workspaces

Go Packages:

- https://pkg.go.dev/text/tabwriter
- https://pkg.go.dev/golang.org/x/tools/txtar
- https://pkg.go.dev/github.com/hashicorp/terraform-json