



Go Transmorgify Yourself.

The magical way of automating cross-API communications with Twonicorn.

... Who Am I

I am Mike Mackintosh

Manager by Title

Engineer by Trade

Hacker by Heart



Verizon Wireless



Shutterstock



Signal Sciences



Snapchat

What Qualifies Me to Talk?

○ ○ ○

Absolutely nothing, and that shouldn't stop you either



What Is Transmorgification?

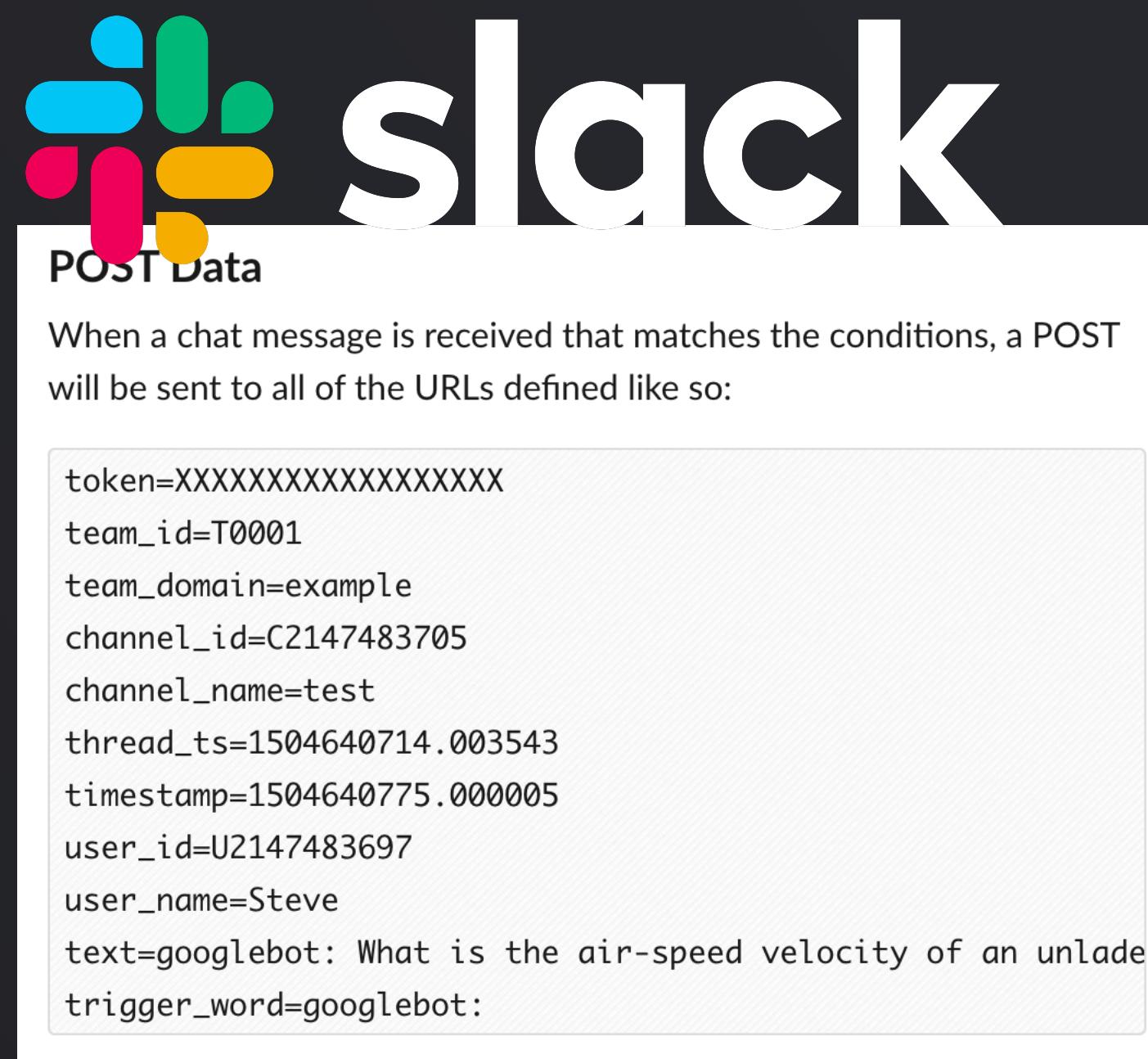
• • •



“ transform in a surprising or magical manner.

... Why Transmorgify?

One of the most common needs is to solve for differences
in egress and ingress API schema's



The image shows a screenshot of the Slack POST Data interface. It features the Slack logo and the title "POST Data". Below this, a text area contains the following message: "When a chat message is received that matches the conditions, a POST will be sent to all of the URLs defined like so:". A code block below lists various parameters:

```
token=XXXXXXXXXXXXXXXXXXXX
team_id=T0001
team_domain=example
channel_id=C2147483705
channel_name=test
thread_ts=1504640714.003543
timestamp=1504640775.000005
user_id=U2147483697
user_name=Steve
text=googlebot: What is the air-speed velocity of an unladen
trigger_word=googlebot:
```

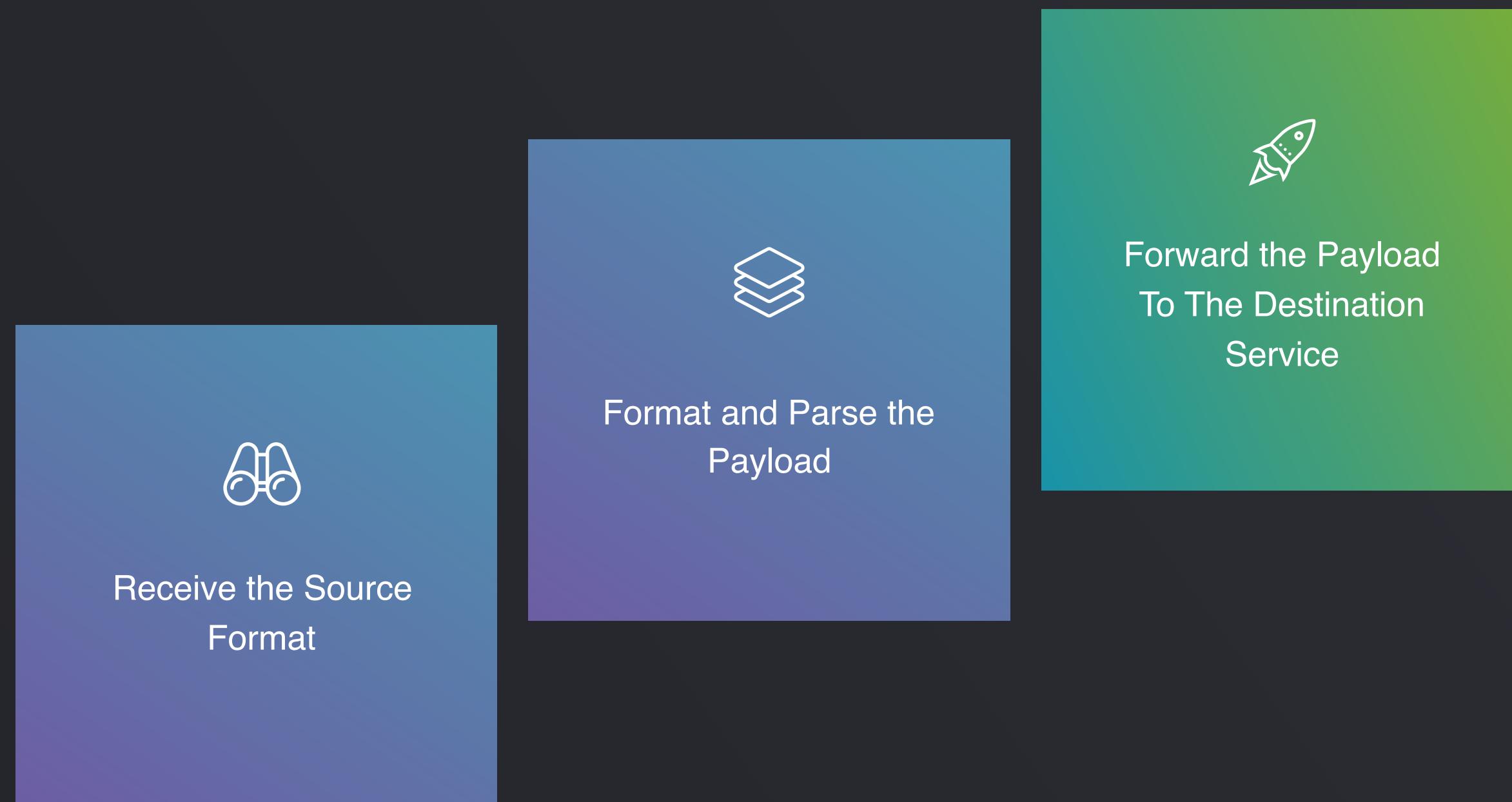


The image shows a screenshot of a JSON response for a Twitch API update. The JSON is as follows:

```
1  {
2    "update": {
3      "comment": [
4        {
5          "add": {
6            "body": "It is time to finish this task"
7          }
8        }
9      ]
10    }
11 }
```

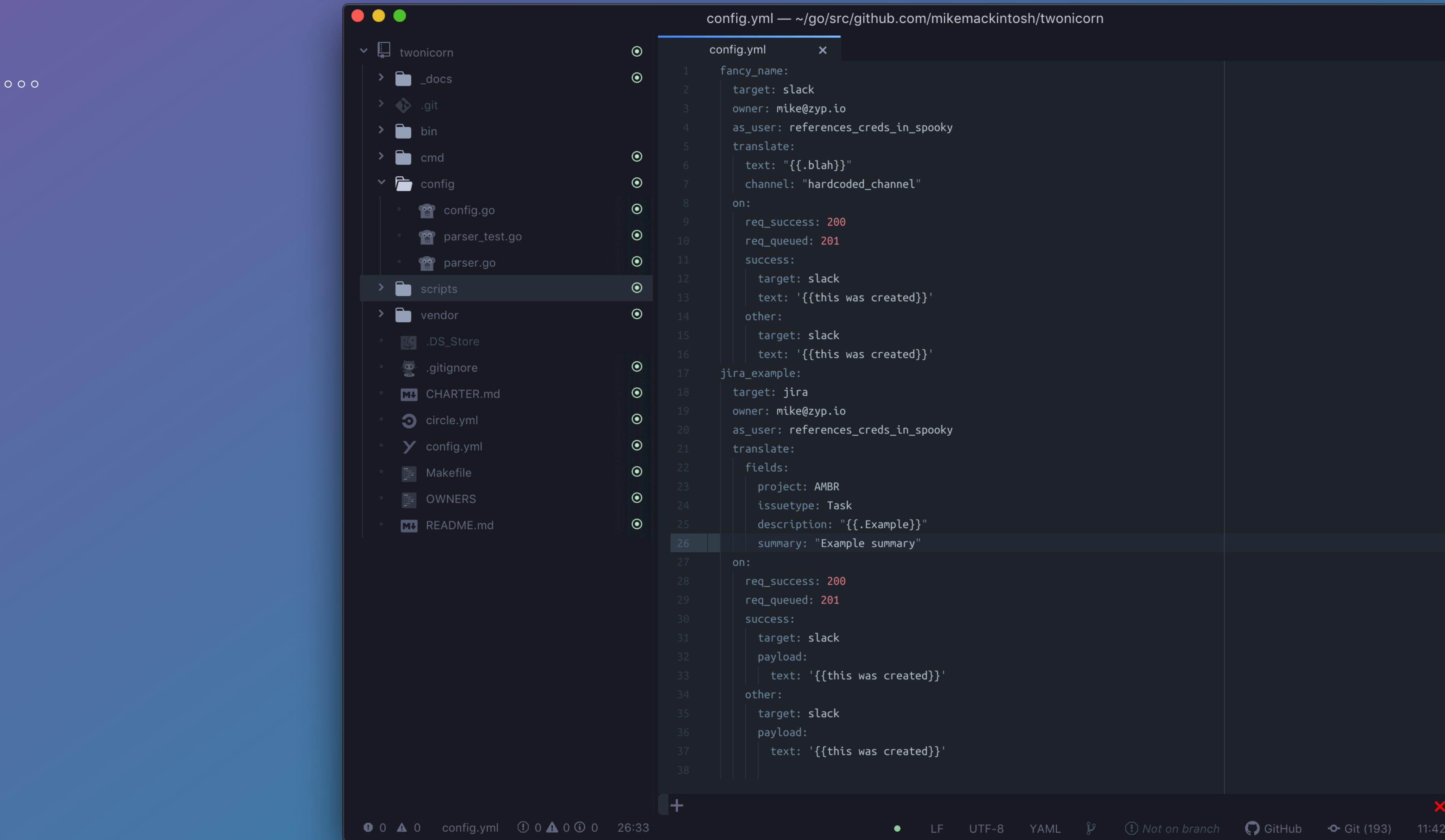
... How Can We Accomplish This?

Everything has a solution if you're a little creative.



Go Transmorgify Yourself

8



```
fancy_name:
target: slack
owner: mike@zyp.io
as_user: references_creds_in_spooky
translate:
text: "{{.blah}}"
channel: hardcoded_channel
on:
req_success: 200
req_queued: 201
success:
target: slack
text: '{{this was created}}'
other:
target: slack
text: '{{this was created}}'
jira_example:
target: jira
owner: mike@zyp.io
as_user: references_creds_in_spooky
translate:
fields:
project: AMBR
issuetype: Task
description: "{{.Example}}"
summary: "Example summary"
on:
req_success: 200
req_queued: 201
success:
target: slack
payload:
text: '{{this was created}}'
other:
target: slack
payload:
text: '{{this was created}}'
```

0 0 ▲ 0 config.yml ① 0 ▲ 0 ① 0 26:33 ● LF UTF-8 YAML ⚡ ① Not on branch GitHub Git (193) 11:42 X

... One Magical Design



01.

Request is received by Twonicorn

02.

Twonicorn matches the incoming configuration path to a configuration entry

03.

Twonicorn parses the incoming body, URL Form Encoded post body or JSON

04.

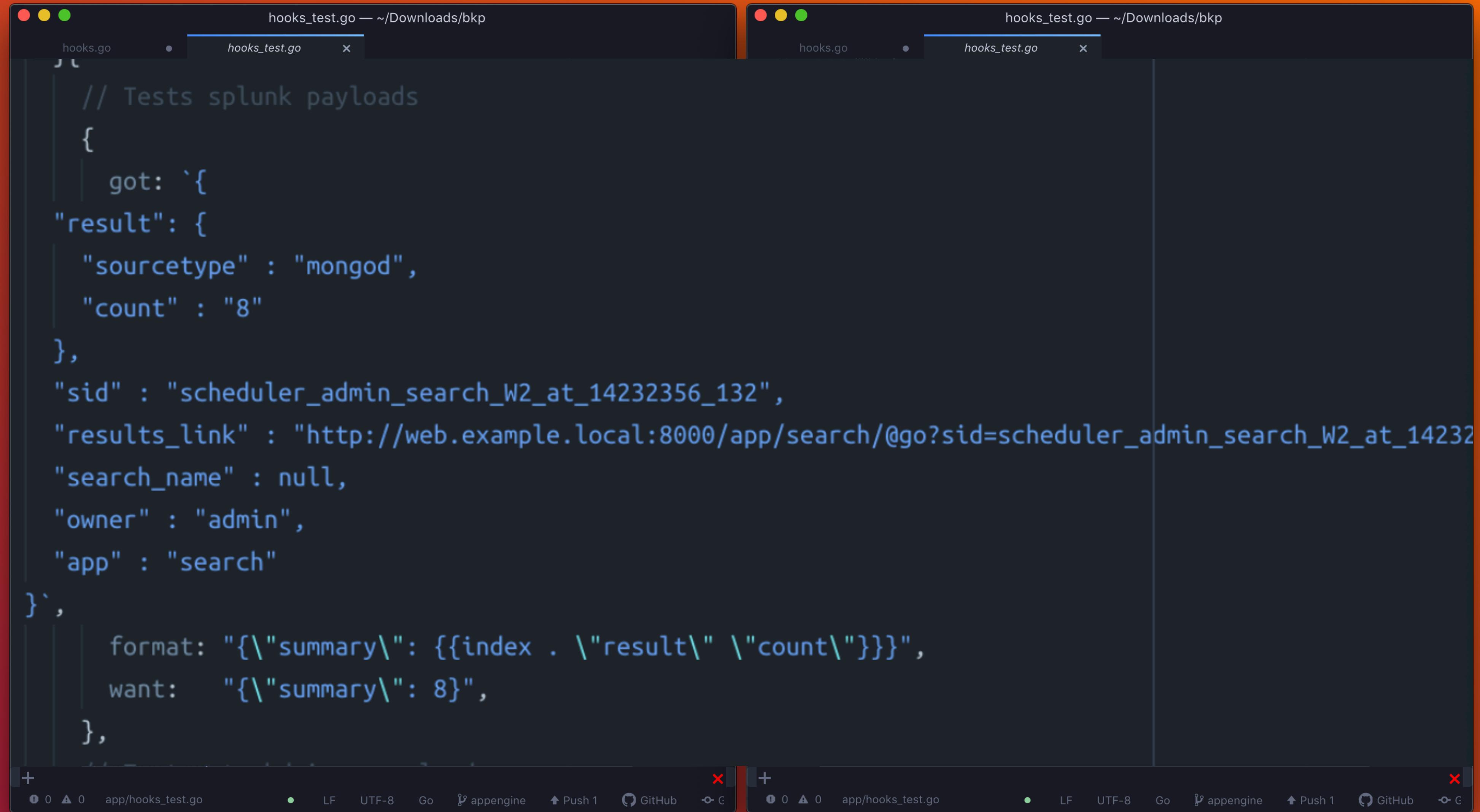
Using a Go template syntax to create outbound request body

05.

Returns the response of the request to the requesting system

Go Transmorgify Yourself

10



The image shows two identical terminal windows side-by-side, both displaying the same Go code. The code is a test function named `TestSplunkPayload` located in `hooks_test.go`. The code uses the `assert` package to compare the output of a function call against expected results.

```
// Tests splunk payloads
{
    got: `{
        "result": {
            "sourcetype" : "mongod",
            "count" : "8"
        },
        "sid" : "scheduler_admin_search_W2_at_14232356_132",
        "results_link" : "http://web.example.local:8000/app/search/@go?sid=scheduler_admin_search_W2_at_14232356_132&search_name=null&owner=admin&app=search"
    }`,
    format: "{\"summary\": {{index . \"result\" \"count\"}}}",
    want:   "{\"summary\": 8}",
},

```

The terminal window includes standard macOS-style window controls (red, yellow, green buttons) and a tab bar at the bottom with tabs for `hooks.go`, `hooks_test.go`, and `app/hooks_test.go`. The status bar at the bottom of each terminal window shows file statistics (0 lines, 0 changes), encoding (UTF-8), and other system information like battery level and network status.

Go Transmorgify Yourself

11

```
1. bash
▶ plug ~/go/src/github.com/mikemackintosh/twunicorn ± HEAD
spitfire.local $ go test -v ./app/...
==== RUN TestHookReceive
---> Received payload:
{
    "result": {
        "sourcetype" : "mongod",
        "count" : "8"
    },
    "sid" : "scheduler_admin_search_W2_at_14232356_132",
    "results_link" : "http://web.example.local:8000/app/search/@go?sid=scheduler_admin_search_W2_at_14232356_132",
    "search_name" : null,
    "owner" : "admin",
    "app" : "search"
}
---> Wanted Format:
{"summary": [{"index . "result" "count"}]}
---> Parsed payload:
map[string]interface {}{"sid":"scheduler_admin_search_W2_at_14232356_132", "results_link":"http://web.example.local:8000/app/search/@go?sid=scheduler_admin_search_W2_at_14232356_132", "search_name":interface {}(nil), "owner":"admin", "app":"search", "result":map[string]interface {}{"sourcetype":"mongod", "count":"8"}}
---> Parsed output:
"{"\\"summary\\": 8}"
```

```
1. bash
---> Received payload:
{
    "incident": {
        "incident_id": "f2e08c333dc64cb09f75eaab355393bz",
        "resource_id": "i-4a266a2d",
        "resource_name": "webserver-85",
        "state": "open",
        "started_at": 1385085727,
        "ended_at": null,
        "policy_name": "Webserver Health",
        "condition_name": "CPU usage",
        "url": "https://app.google.stackdriver.com/incidents/f333dc64z",
        "summary": "CPU for webserver-85 is above the threshold of 1% with a value of 28.5%"
    },
    "version": 1.1
}
---> Wanted Format:
{"summary": "{{{index . "incident" "state}}} incident of {{{index . "incident" "summary"}}}"}
---> Parsed payload:
map[string]interface {}{"incident":map[string]interface {}{"incident_id":"f2e08c333dc64cb09f75eaab355393bz", "resource_name":"webserver-85", "state":"open", "url":"https://app.google.stackdriver.com/incidents/f333dc64z", "summary":"CPU for webserver-85 is above the threshold of 1% with a value of 28.5%", "resource_id":"i-4a266a2d", "started_at":1.385085727e+09, "ended_at":interface {}(nil), "policy_name":"Webserver Health", "condition_name":"CPU usage"}, "version":1.1}
---> Parsed output:
"{"\\"summary\\": \"open incident of CPU for webserver-85 is above the threshold of 1% with a value of 28.5%\"}"
```

```
hooks.go — ~/go/src/github.com/mikemackintosh/twonicorn
hooks.go x
1 package main
2
3 import (
4     "bytes"
5     "encoding/json"
6     "fmt"
7     "html/template"
8     "io"
9     "net/http"
10    "path"
11    "path/filepath"
12
13    "github.com/mikemackintosh/twonicorn/config"
14    "google.golang.org/appengine/log"
15 )
16
17 // HookHandler will get and parse an incoming request
18 func HookHandler(w http.ResponseWriter, r *http.Request, hooks *config.Hooks) {
19     wrapper := NewContext(w, r)
20     key := path.Base(r.URL.Path)
21     prefix := path.Base(filepath.Dir(r.URL.Path))
22
23     if len(key) == 0 {
24         log.Infof(wrapper.Ctx, "Invalid key detected")
25         wrapper.Write(respInvalidReq)
26         return
27     }
28
29     // Piece together the key
30     pathKey := fmt.Sprintf("%s%s", prefix, key)
31     // POC Debugging
32     log.Debugf(wrapper.Ctx, "-> Looking up key: %s", pathKey)
33     hook, err := hooks.GetKey(pathKey)
34     if err != nil {
35         log.Infof(wrapper.Ctx, "Invalid key detected")
36         wrapper.Write(respInvalidReq)
37         return
38 }
```

```
hooks.go — ~/go/src/github.com/mikemackintosh/twonicorn
hooks.go x
38 }
39
40 // Looks like we found a hook
41 log.Infof(wrapper.Ctx, "Found hook: %s", hook.Name)
42
43 // Parse the request payload into an interface, so we can handle it
44 var payload map[string]interface{}
45 err = DecodeReceivedPayload(r.Body, &payload)
46 if err != nil {
47     log.Infof(wrapper.Ctx, "Invalid payload decode: %s", err)
48     wrapper.Write(respInvalidReq)
49     return
50 }
51
52 // POC Debugging
53 log.Debugf(wrapper.Ctx, "Payload: %+v", payload)
54 log.Debugf(wrapper.Ctx, "Translate to: %+v", *hook.Translate)
55 targetPayload, err := ParseTargetPayloadFormat(hook.Translate)
56 if err != nil {
57     log.Infof(wrapper.Ctx, "Invalid target payload marshal: %s", err)
58     wrapper.Write(respInvalidReq)
59     return
60 }
61
62 // Creates the target payload
63 output, err := CreateTargetPayload(string(targetPayload), payload)
64 if err != nil {
65     log.Infof(wrapper.Ctx, "Invalid target payload created: %s", err)
66     wrapper.Write(respInvalidReq)
67     return
68 }
69
70 // POC Debugging
71 log.Debugf(wrapper.Ctx, output.String())
72
73 wrapper.Write(respSuccessfulReq)
74
75 return
```

```
hooks.go — ~/go/src/github.com/mikemackintosh/twonicorn
hooks.go x
67 wrapper.Write(respInvalidReq)
68 return
69 }
70
71 // POC Debugging
72 log.Debugf(wrapper.Ctx, output.String())
73
74 wrapper.Write(respSuccessfulReq)
75 return
76 }
77
78 // DecodeReceivedPayload decodes a response body to a payload object
79 func DecodeReceivedPayload(body io.Reader, payload interface{}) error {
80     return json.NewDecoder(body).Decode(&payload)
81 }
82
83 // ParseTargetPayloadFormat converts the target payload format to a json
84 func ParseTargetPayloadFormat(targetFormat interface{}) ([]byte, error) {
85     targetPayload, err := json.Marshal(targetFormat)
86     return targetPayload, err
87 }
88
89 // CreateTargetPayload will generate the target payload
90 func CreateTargetPayload(format string, payload interface{}) (bytes.Buffer, error) {
91     var output bytes.Buffer
92
93     tmpl, err := template.New("").Funcs(transformTemplates).Parse(format)
94     if err != nil {
95         return output, err
96     }
97
98     if err := tmpl.Execute(&output, payload); err != nil {
99         return output, err
100    }
101
102    return output, nil
103 }
104
```



Future of Twonicorn

Pub/Sub

Add Pub/Sub-queueing support for request retry.

Metrics.

Add statsd/prometheus/etc metric monitoring.

Client-less Approach.

Removes the need for custom API clients by using a headers config field.

KMS support.

Securely store your secrets for as_user blocks in KMS.

Questions?

Special Thanks

Unicorns Exist