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


NOTHING IS BETTER
THAN A GOOD LAUGH!

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

slack

POST Data

When a chat message is received that matches the conditions, a POST will be sent to all of the URLs defined like so:

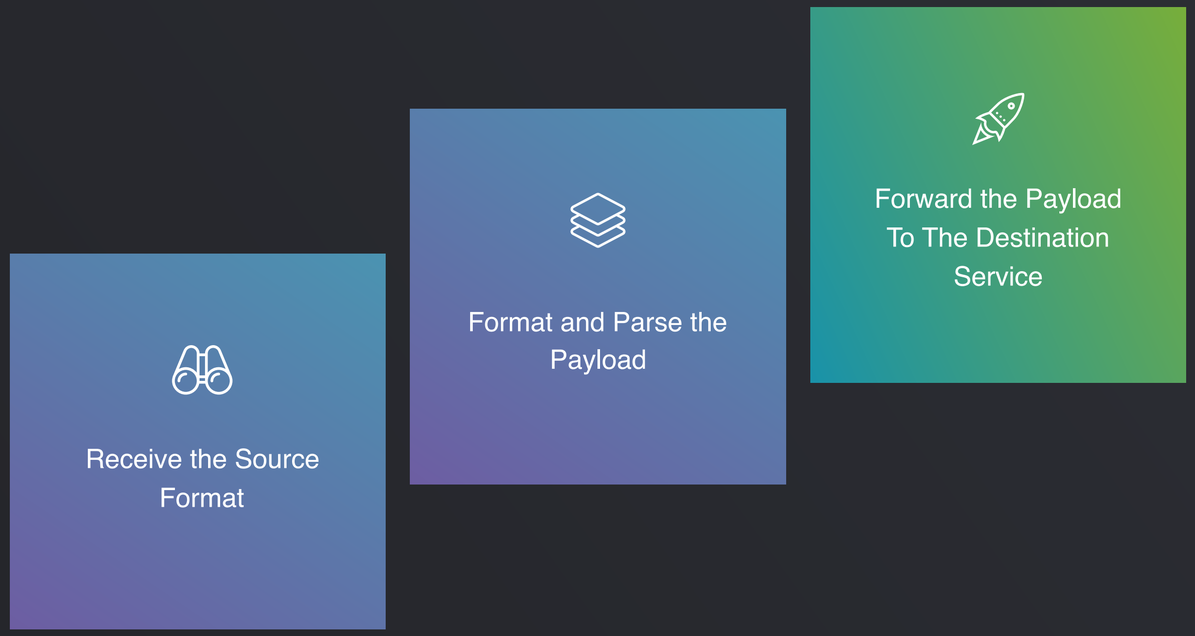
```
token=XXXXXXXXXXXXXXXXX
team_id=T0001
team_domain=example
channel_id=C2147483705
channel_name=test
thread_ts=1504640714.003543
timestamp=1504640775.000005
user_id=U2147483697
user_name=Steve
text=googlebot: What is the air-speed velocity of an unlade
trigger_word=googlebot:
```

Copy

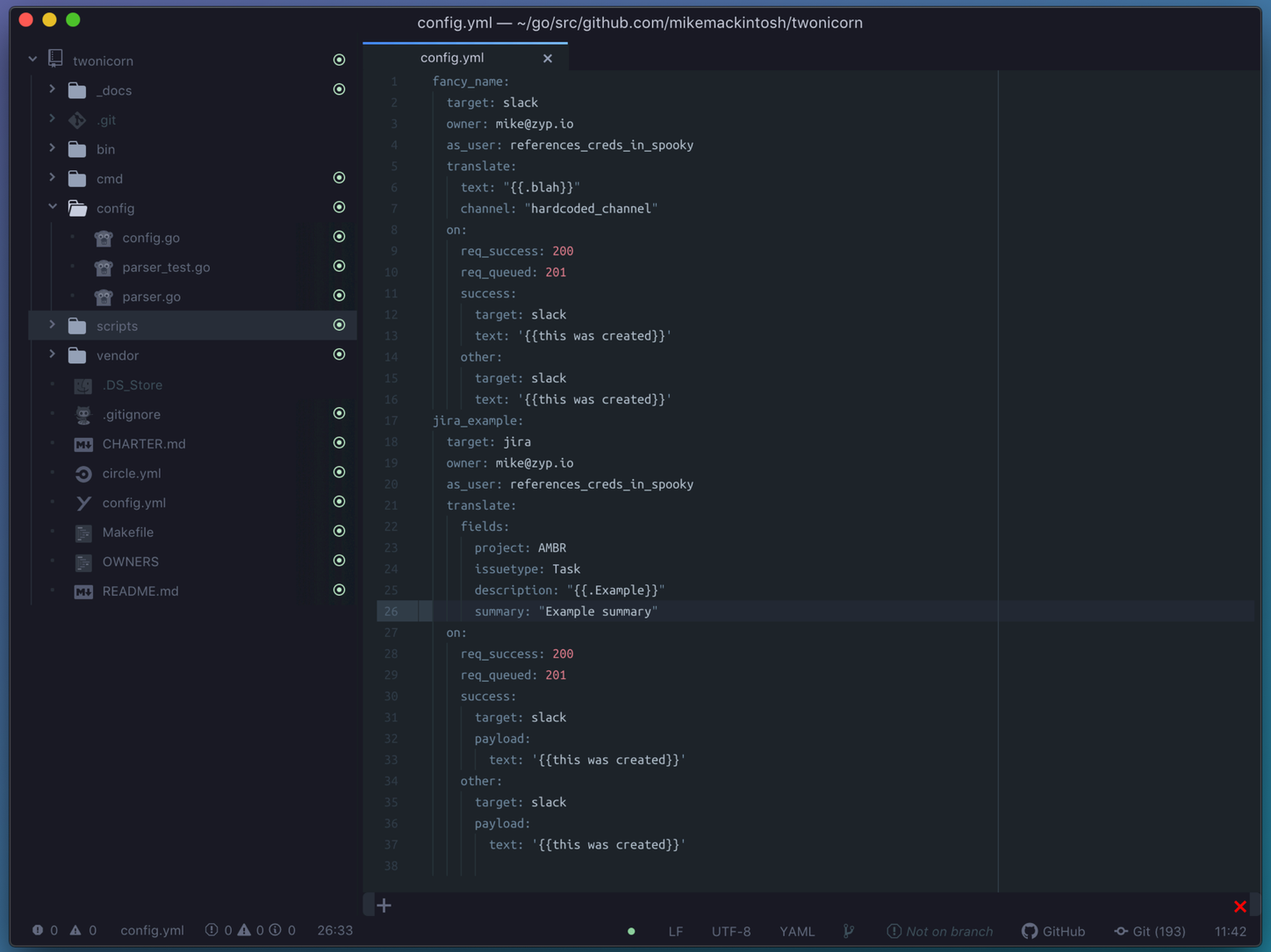
```
{
   "update": {
      "comment": [
         {
            "add": {
               "body": "It is time to finish this task"
            }
         }
      ]
   }
}
```

How Can We Accomplish This?

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

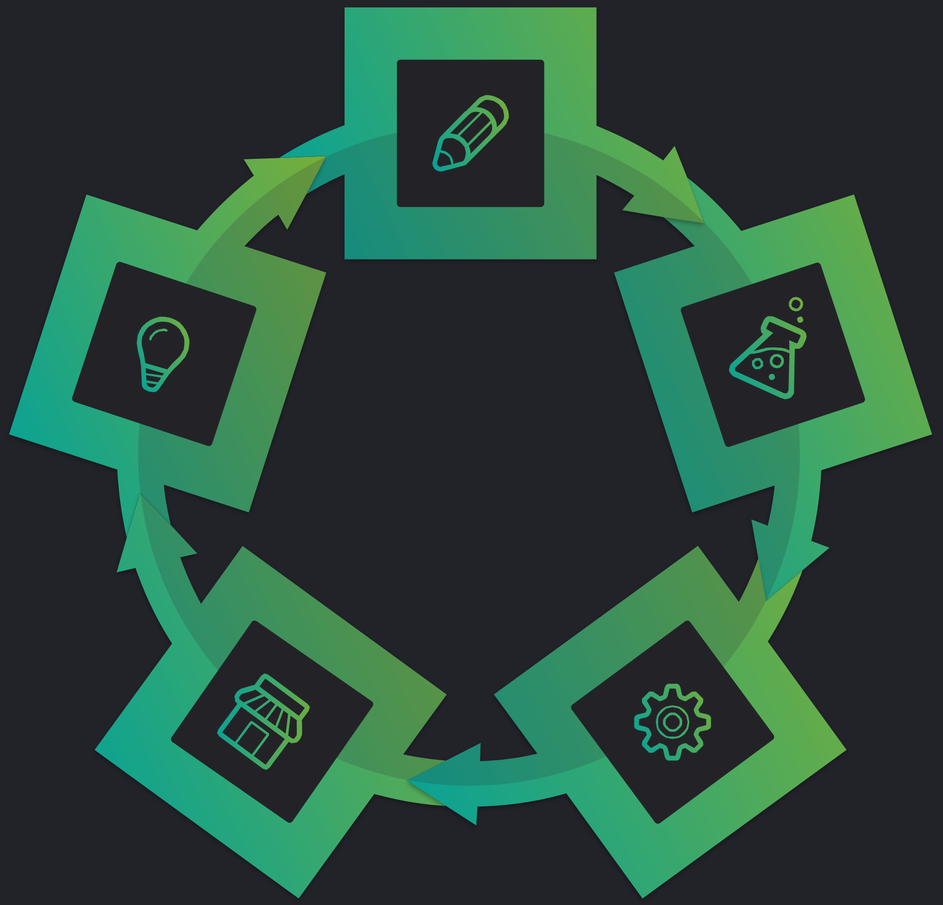


Go Transmorgify Yourself



@mikemackintosh

One Magical Design



01.

Request is received by Twonicorn

02.

Twonicorn matches the incoming configuration path to a configuration entry

03.

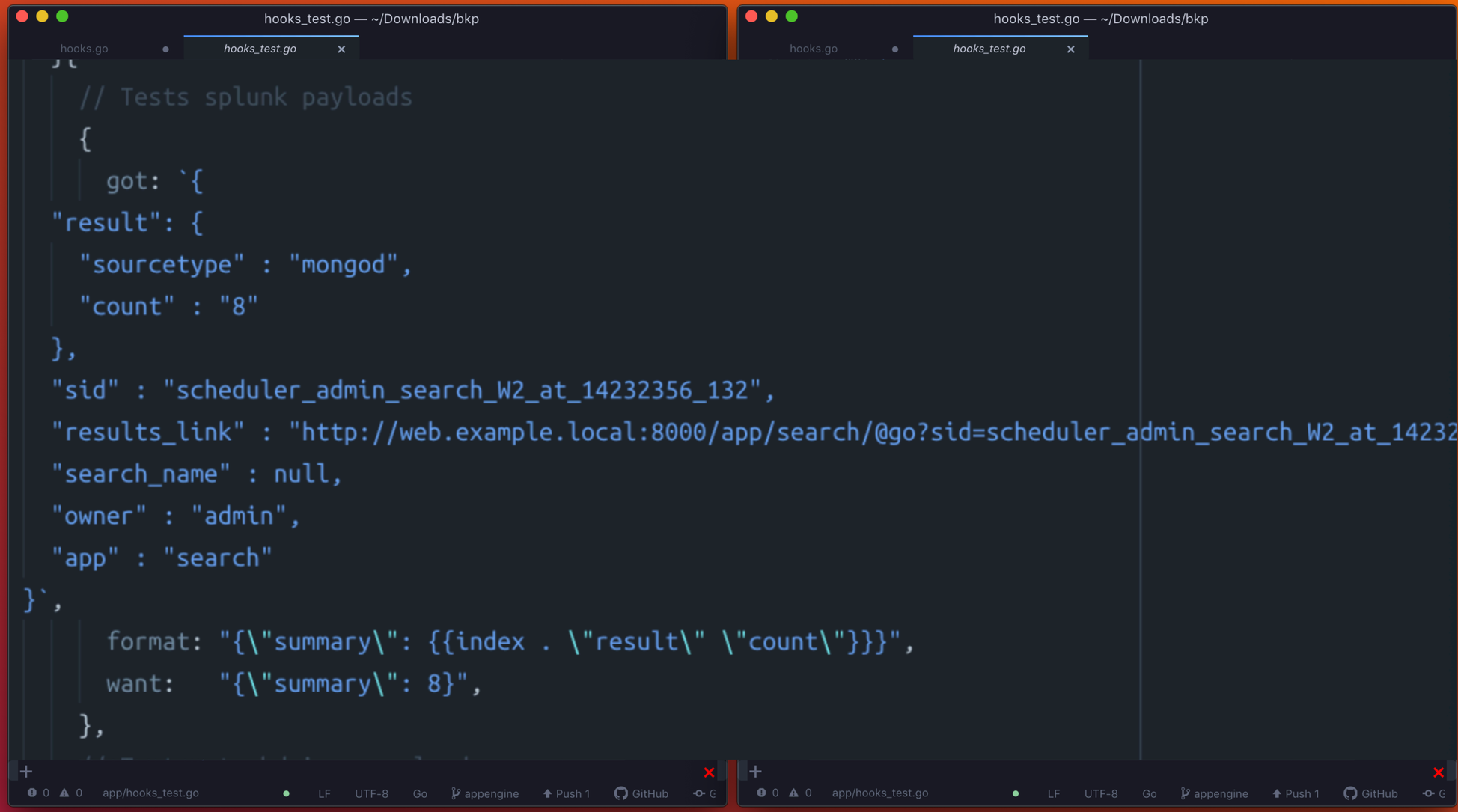
Twonicorn parses the incoming body, URL Form Encoded post body or JSON

04.

Using a Go template syntax to create outbound request body

05.

Returns the response of the request to the requesting system


hooks_test.go — ~/Downloads/bkp
hooks.go
hooks_test.go
// 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_14232
"search_name" : null,
"owner" : "admin",
"app" : "search"
}`,
format: "{\"summary\": {{index . \"result\" \"count\"}}}",
want: "{\"summary\": 8}",
},
0 0 app/hooks_test.go LF UTF-8 Go appengine Push 1 GitHub
hooks_test.go — ~/Downloads/bkp
hooks.go
hooks_test.go
0 0 app/hooks_test.go LF UTF-8 Go appengine Push 1 GitHub

Go Transmorgify Yourself

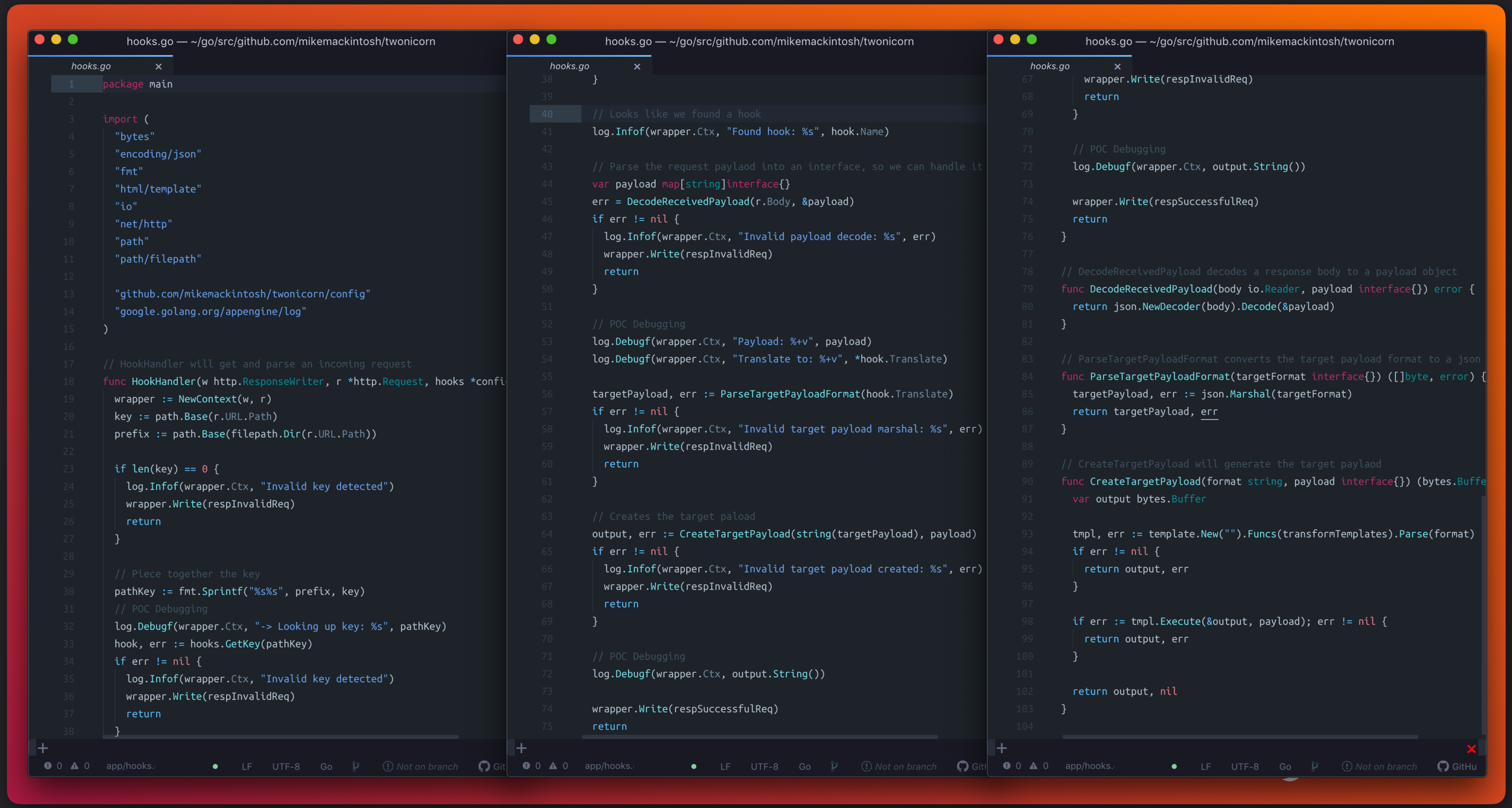
1. bash

```
▸ splug ~/go/src/github.com/mikemackintosh/twonicorn ± HEAD                                                    [56|12:13]
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%\"}"
```

@mikemackintosh


hooks.go — ~/go/src/github.com/mikemackintosh/twonicorn
hooks.go
package main
import (
"bytes"
"encoding/json"
"fmt"
"html/template"
"io"
"net/http"
"path"
"path/filepath"
"github.com/mikemackintosh/twonicorn/config"
"google.golang.org/appengine/log"
)
// HookHandler will get and parse an incoming request
func HookHandler(w http.ResponseWriter, r *http.Request, hooks *confi
wrapper := NewContext(w, r)
key := path.Base(r.URL.Path)
prefix := path.Base(filepath.Dir(r.URL.Path))
if len(key) == 0 {
log.Infof(wrapper.Ctx, "Invalid key detected")
wrapper.Write(respInvalidReq)
return
}
// Piece together the key
pathKey := fmt.Sprintf("%s%s", prefix, key)
// POC Debugging
log.Debugf(wrapper.Ctx, "-> Looking up key: %s", pathKey)
hook, err := hooks.GetKey(pathKey)
if err != nil {
log.Infof(wrapper.Ctx, "Invalid key detected")
wrapper.Write(respInvalidReq)
return
}
app/hooks.
LF
UTF-8
Go
Not on branch
hooks.go — ~/go/src/github.com/mikemackintosh/twonicorn
hooks.go
}
// Looks like we found a hook
log.Infof(wrapper.Ctx, "Found hook: %s", hook.Name)
// Parse the request paylaod into an interface, so we can handle it
var payload map[string]interface{}
err = DecodeReceivedPayload(r.Body, &payload)
if err != nil {
log.Infof(wrapper.Ctx, "Invalid payload decode: %s", err)
wrapper.Write(respInvalidReq)
return
}
// POC Debugging
log.Debugf(wrapper.Ctx, "Payload: %+v", payload)
log.Debugf(wrapper.Ctx, "Translate to: %+v", *hook.Translate)
targetPayload, err := ParseTargetPayloadFormat(hook.Translate)
if err != nil {
log.Infof(wrapper.Ctx, "Invalid target payload marshal: %s", err)
wrapper.Write(respInvalidReq)
return
}
// Creates the target paload
output, err := CreateTargetPayload(string(targetPayload), payload)
if err != nil {
log.Infof(wrapper.Ctx, "Invalid target payload created: %s", err)
wrapper.Write(respInvalidReq)
return
}
// POC Debugging
log.Debugf(wrapper.Ctx, output.String())
wrapper.Write(respSuccessfulReq)
return
app/hooks.
LF
UTF-8
Go
Not on branch
hooks.go — ~/go/src/github.com/mikemackintosh/twonicorn
hooks.go
wrapper.Write(respInvalidReq)
return
}
// POC Debugging
log.Debugf(wrapper.Ctx, output.String())
wrapper.Write(respSuccessfulReq)
return
}
// DecodeReceivedPayload decodes a response body to a payload object
func DecodeReceivedPayload(body io.Reader, payload interface{}) error {
return json.NewDecoder(body).Decode(&payload)
}
// ParseTargetPayloadFormat converts the target payload format to a json
func ParseTargetPayloadFormat(targetFormat interface{}) ([]byte, error) {
targetPayload, err := json.Marshal(targetFormat)
return targetPayload, err
}
// CreateTargetPayload will generate the target paylaod
func CreateTargetPayload(format string, payload interface{}) (bytes.Buffe
var output bytes.Buffer
tmpl, err := template.New("").Funcs(transformTemplates).Parse(format)
if err != nil {
return output, err
}
if err := tmpl.Execute(&output, payload); err != nil {
return output, err
}
return output, nil
}
app/hooks.
LF
UTF-8
Go
Not on branch
GitHu

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 you secrets for as_user blocks in KMS.

Questions?

@mikemackintosh

Special Thanks

Unicorns Exist