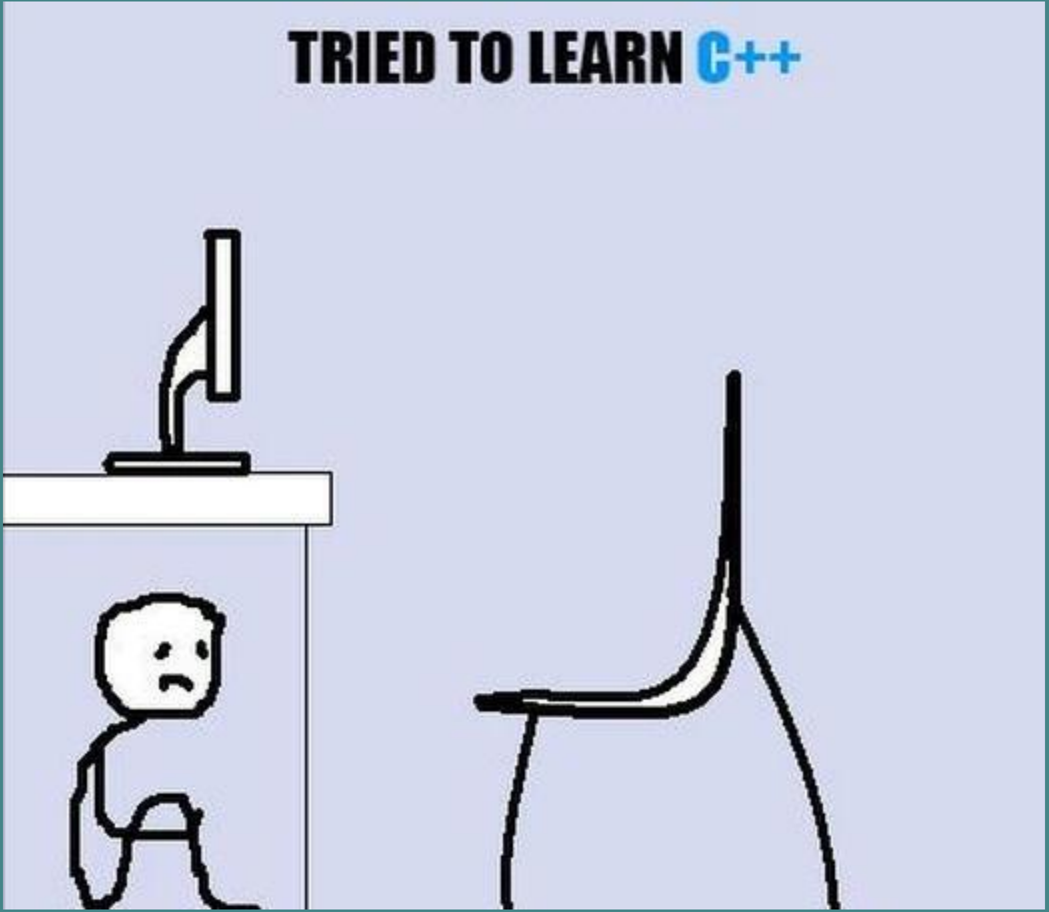
Extending Osquery

Why

229 Tables

curl
docker
prometheus
python_packages
ec2_instances
kinesis/kafka loggers


TRIED TO LEARN C++

How

Config

func NewPlugin

```
func NewPlugin(name string, fn GenerateConfigsFunc) *Plugin
```

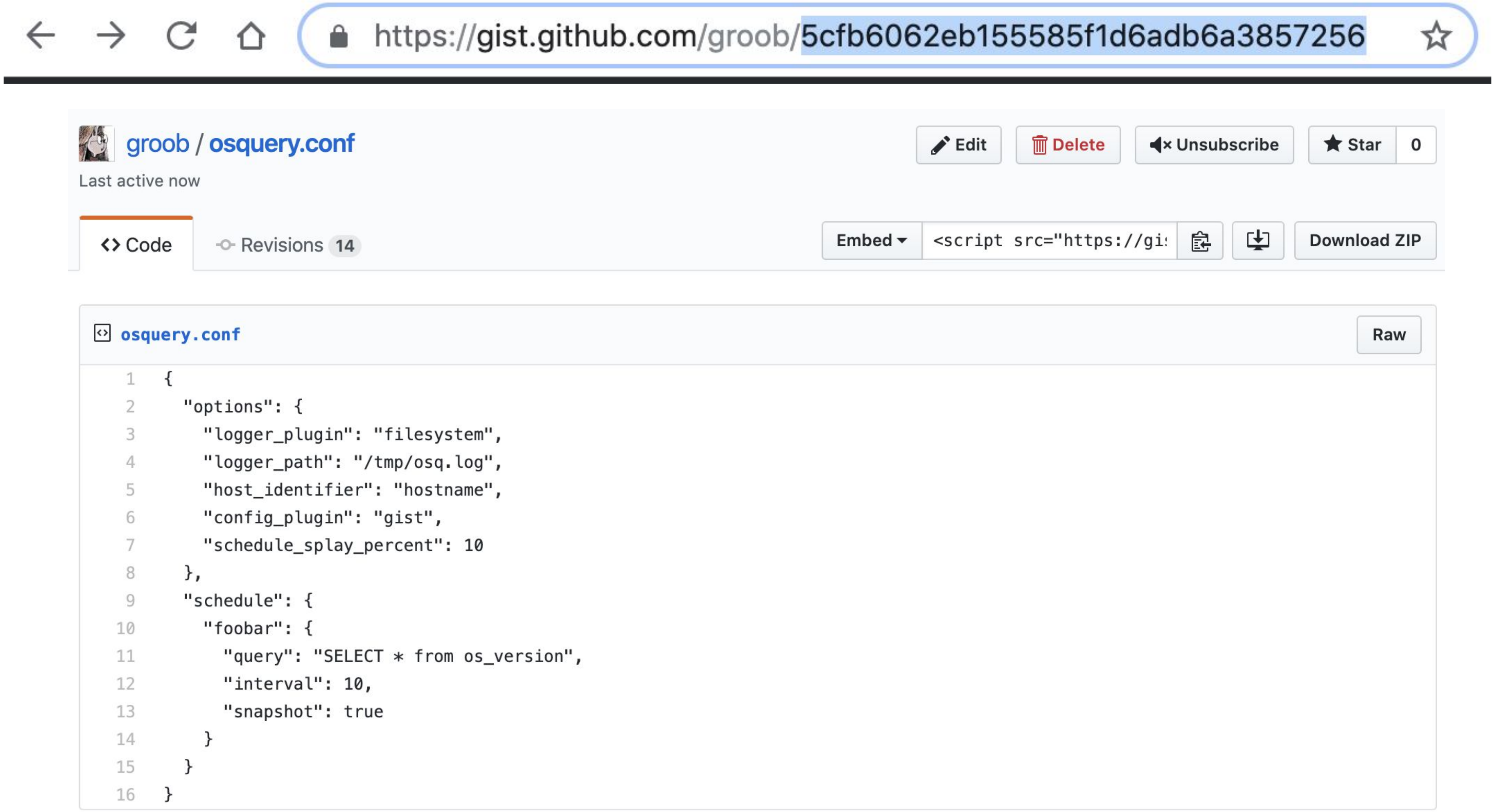
func NewPlugin

```
func NewPlugin(name string, fn GenerateConfigsFunc) *Plugin
```

type GenerateConfigsFunc

```
type GenerateConfigsFunc func(ctx context.Context) (map[string]string, error)
```

GenerateConfigsFunc returns the configurations generated by this plugin. The returned map should use the source name as key, and the config JSON as values. The context argument can optionally be used for cancellation in long-running operations.



```
{
  "options": {
    "logger_plugin": "filesystem",
    "logger_path": "/tmp/osq.log",
    "host_identifier": "hostname",
    "config_plugin": "gist",
    "schedule_splay_percent": 10
  },
  "schedule": {
    "foobar": {
      "query": "SELECT * from os_version",
      "interval": 10,
      "snapshot": true
    }
  }
}
```

```go
type Plugin struct { gistID string; client *github.Client }
type Plugin struct { gistID string; client *github.Client }

func New() *config.Plugin {
    plugin := &Plugin{
        client: github.NewClient(nil),
        gistID: "5cfb6062eb155585f1d6adb6a3857256",
    }
    return config.NewPlugin("gist", plugin.GenerateConfigs)
}
```

```go
func (p *Plugin) GenerateConfigs(ctx context.Context) (map[string]string, error) {
	gist, _, err := p.client.Gists.Get(ctx, p.gistID)
	if err != nil {
		return nil, fmt.Errorf("fetch gist %s", p.gistID)
	}

	if file, ok := gist.Files["osquery.conf"]; ok {
		return map[string]string{"gist": file.GetContent()}, nil
	}
	return nil, fmt.Errorf("no osquery.conf file in gist %s", p.gistID)
}
```

Loggers

func NewPlugin

```
func NewPlugin(name string, fn LogFunc) *Plugin
```

NewPlugin takes a value that implements LoggerPlugin and wraps it with the appropriate methods to satisfy the OsqueryPlugin interface. Use this to easily create plugins implementing osquery loggers.

func NewPlugin

```
func NewPlugin(name string, fn LogFunc) *Plugin
```

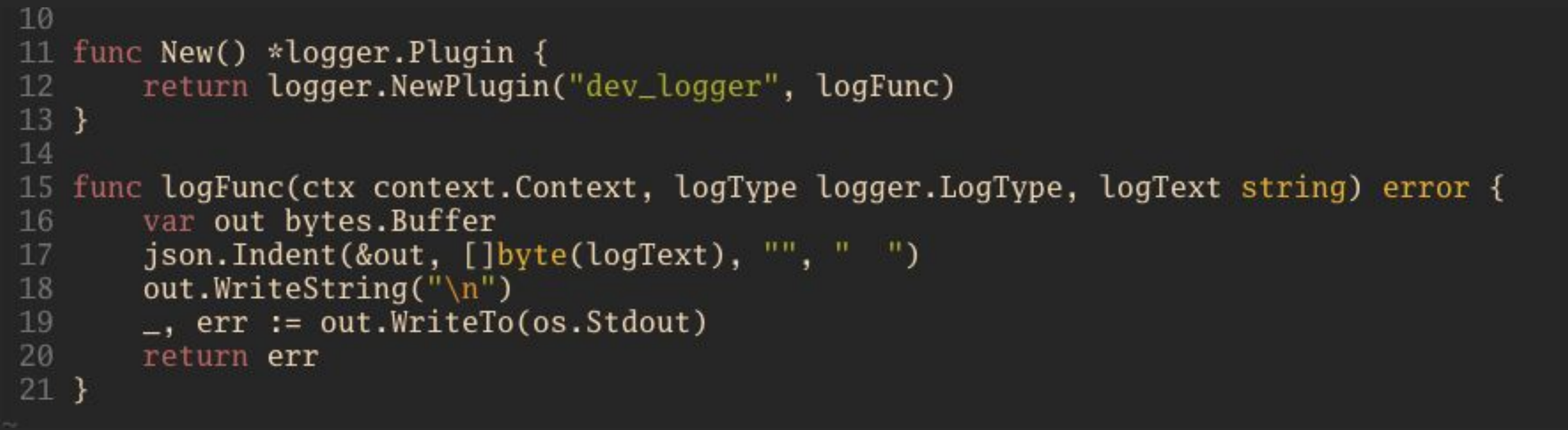
NewPlugin takes a value that implements LoggerPlugin and wraps it with the appropriate methods to satisfy the OsqueryPlugin interface. Use this to easily create plugins implementing osquery loggers.

type LogFunc

```
type LogFunc func(ctx context.Context, typ LogType, log string) error
```

LogFunc is the logger function used by an osquery Logger plugin.

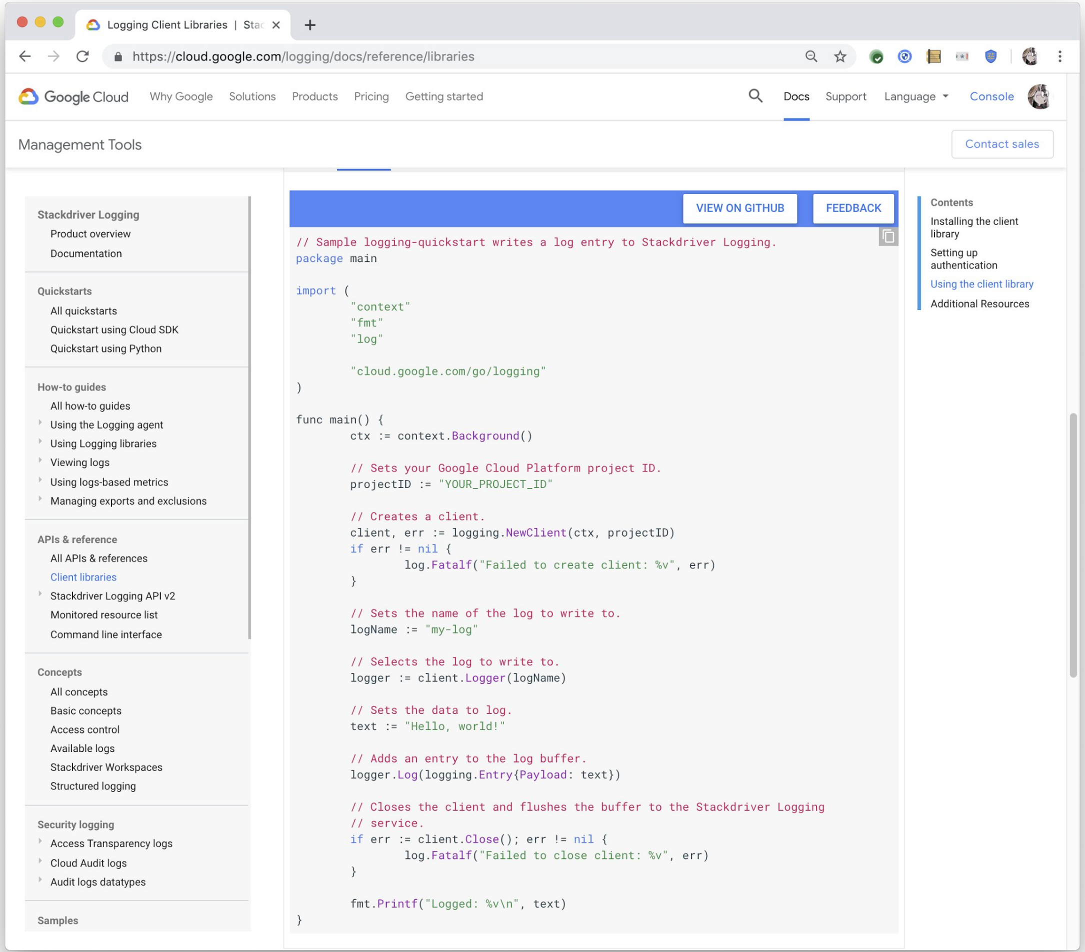
The LogFunc should log the provided result string. The LogType argument can be optionally used to log differently depending on the type of log received. The context argument can optionally be used for cancellation in long-running operations.



```go

func New() *logger.Plugin {
    return logger.NewPlugin("dev_logger", logFunc)
}

func logFunc(ctx context.Context, logType logger.LogType, logText string) error {
    var out bytes.Buffer
    json.Indent(&out, []byte(logText), "", "  ")
    out.WriteString("\n")
    _, err := out.WriteTo(os.Stdout)
    return err
}
```



Stackdriver Logging
Product overview
Documentation
Quickstarts
All quickstarts
Quickstart using Cloud SDK
Quickstart using Python
How-to guides
All how-to guides
Using the Logging agent
Using Logging libraries
Viewing logs
Using logs-based metrics
Managing exports and exclusions
APIs & reference
All APIs & references
Client libraries
Stackdriver Logging API v2
Monitored resource list
Command line interface
Concepts
All concepts
Basic concepts
Access control
Available logs
Stackdriver Workspaces
Structured logging
Security logging
Access Transparency logs
Cloud Audit logs
Audit logs datatypes
Samples

```go
// Sample logging-quickstart writes a log entry to Stackdriver Logging.
package main

import (
	"context"
	"fmt"
	"log"

	"cloud.google.com/go/logging"
)

func main() {
	ctx := context.Background()

	// Sets your Google Cloud Platform project ID.
	projectID := "YOUR_PROJECT_ID"

	// Creates a client.
	client, err := logging.NewClient(ctx, projectID)
	if err != nil {
		log.Fatalf("Failed to create client: %v", err)
	}

	// Sets the name of the log to write to.
	logName := "my-log"

	// Selects the log to write to.
	logger := client.Logger(logName)

	// Sets the data to log.
	text := "Hello, world!"

	// Adds an entry to the log buffer.
	logger.Log(logging.Entry{Payload: text})

	// Closes the client and flushes the buffer to the Stackdriver Logging
	// service.
	if err := client.Close(); err != nil {
		log.Fatalf("Failed to close client: %v", err)
	}

	fmt.Printf("Logged: %v\n", text)
}
```

Contents
Installing the client library
Setting up authentication
Using the client library
Additional Resources

```go
type Plugin struct {
    logger *logging.Logger
}

func New() *logger.Plugin {
    ctx := context.Background()
    projectID := "querycon2019"

    client, err := logging.NewClient(ctx, projectID)
    if err != nil {
        panic(err)
    }

    plugin := &Plugin{
        logger: client.Logger("osquery-result"),
    }
    return logger.NewPlugin("gcplog", plugin.Log)
}

func (p *Plugin) Log(ctx context.Context, logType logger.LogType, logText string) error {
    return p.logger.LogSync(ctx, logging.Entry{Payload: logText})
}
```

```go

func New() *logger.Plugin {
	return logger.NewPlugin("dev_logger", logFunc)
}

func logFunc(ctx context.Context, logType logger.LogType, logText string) error {
	var out bytes.Buffer
	json.Indent(&out, []byte(logText), "", "  ")
	out.WriteString("\n")
	_, err := out.WriteTo(os.Stdout)
	return err
}
```

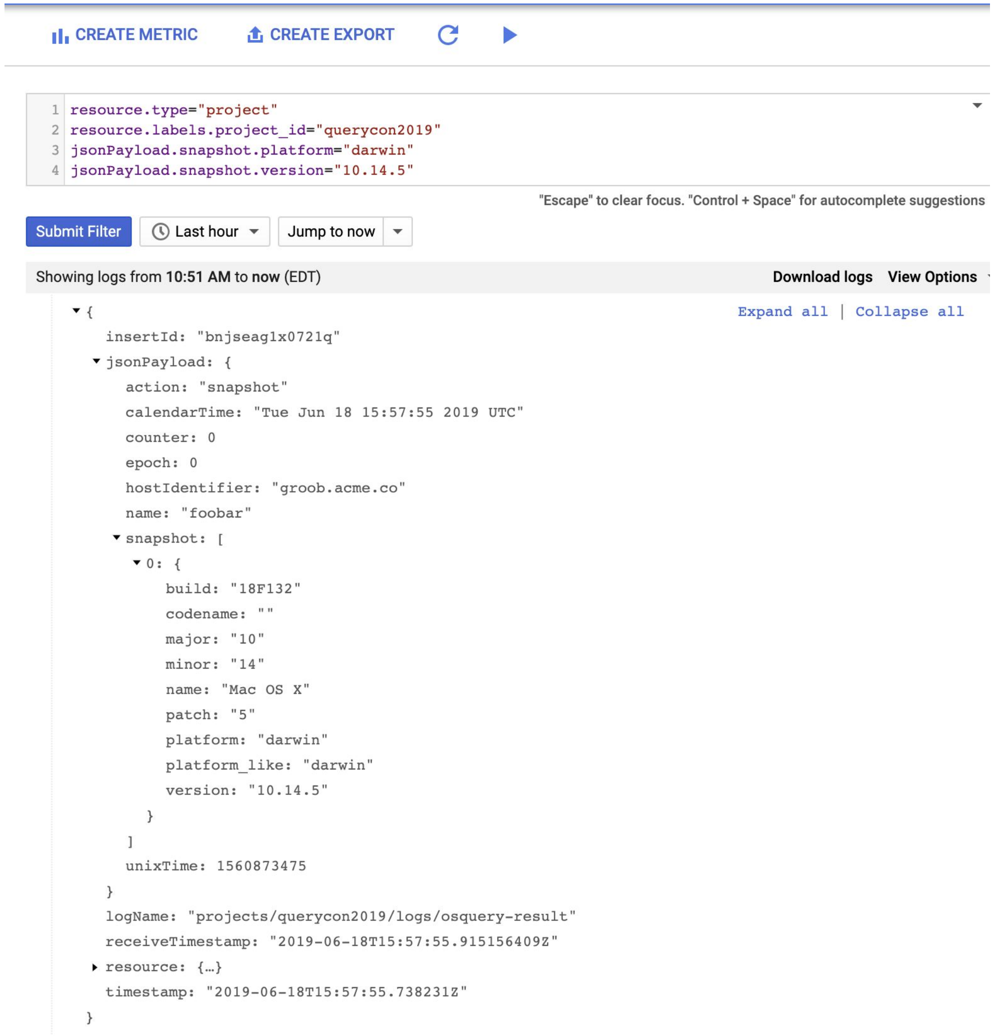
```go
type Plugin struct {
    logger *logging.Logger
}

func New() *logger.Plugin {
    ctx := context.Background()
    projectID := "querycon2019"

    client, err := logging.NewClient(ctx, projectID)
    if err != nil {
        panic(err)
    }

    plugin := &Plugin{
        logger: client.Logger("osquery-result"),
    }
    return logger.NewPlugin("gcplog", plugin.Log)
}

func (p *Plugin) Log(ctx context.Context, logType logger.LogType, logText string) error {
    return p.logger.LogSync(ctx, logging.Entry{Payload: logText})
}
```


CREATE METRIC
CREATE EXPORT

```
resource.type="project"
resource.labels.project_id="querycon2019"
jsonPayload.snapshot.platform="darwin"
jsonPayload.snapshot.version="10.14.5"
```

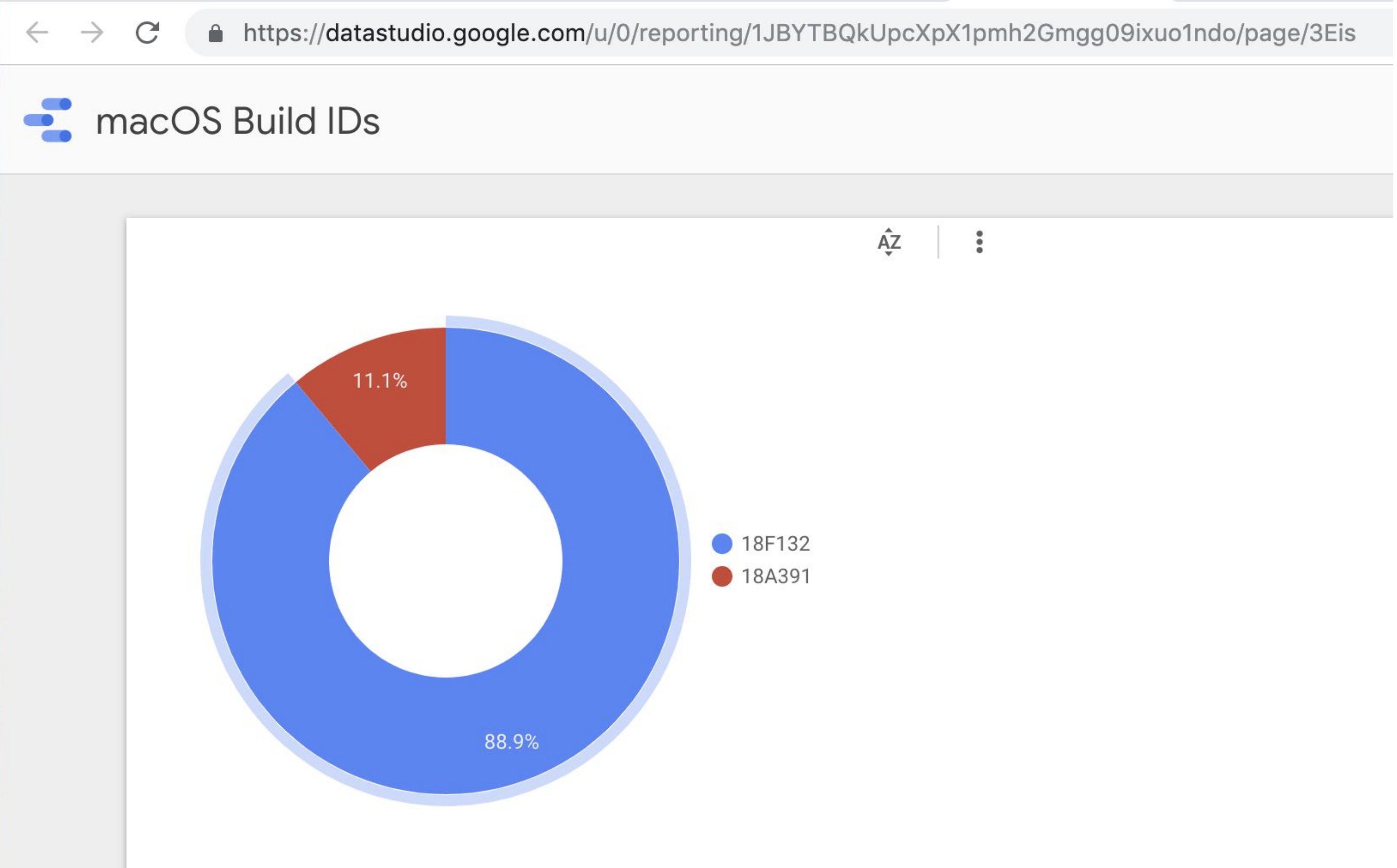
"Escape" to clear focus. "Control + Space" for autocomplete suggestions
Submit Filter
Last hour
Jump to now

Showing logs from **10:51 AM** to now (EDT)

Download logs
View Options
Expand all | Collapse all

```
{
  insertId: "bnjseag1x0721q"
  jsonPayload: {
    action: "snapshot"
    calendarTime: "Tue Jun 18 15:57:55 2019 UTC"
    counter: 0
    epoch: 0
    hostIdentifier: "groob.acme.co"
    name: "foobar"
    snapshot: [
      0: {
        build: "18F132"
        codename: ""
        major: "10"
        minor: "14"
        name: "Mac OS X"
        patch: "5"
        platform: "darwin"
        platform_like: "darwin"
        version: "10.14.5"
      }
    ]
    unixTime: 1560873475
  }
  logName: "projects/querycon2019/logs/osquery-result"
  receiveTimestamp: "2019-06-18T15:57:55.915156409Z"
  resource: {…}
  timestamp: "2019-06-18T15:57:55.738231Z"
}
```

Osquery
> BigQuery
> DataStudio


https://datastudio.google.com/u/0/reporting/1JBYTBQkUpcXpX1pmh2Gmgg09ixuo1ndo/page/3Eis

macOS Build IDs

Register **Plugins** in an **Extension**

```go
func main() {
    var (
        flSocketPath = flag.String("socket", "", "")
        flTimeout    = flag.Int("timeout", 0, "")
        _            = flag.Int("interval", 0, "")
        _            = flag.Bool("verbose", false, "")
    )
    flag.Parse()

    // create an extension server
    server, err := osquery.NewExtensionManagerServer(
        "co.acme.extension",
        *flSocketPath,
        osquery.ServerTimeout(time.Duration(*flTimeout)*time.Second),
    )
    if err != nil {
        log.Fatalf("creating extension: %s\n", err)
    }

    gistConfig := config.New()
    devLogger := devlogger.New()
    gcpLogger := gcplog.New()
    server.RegisterPlugin(
        gistConfig,
        devLogger,
        gcpLogger,
    )

    log.Fatal(server.Run())
}
```

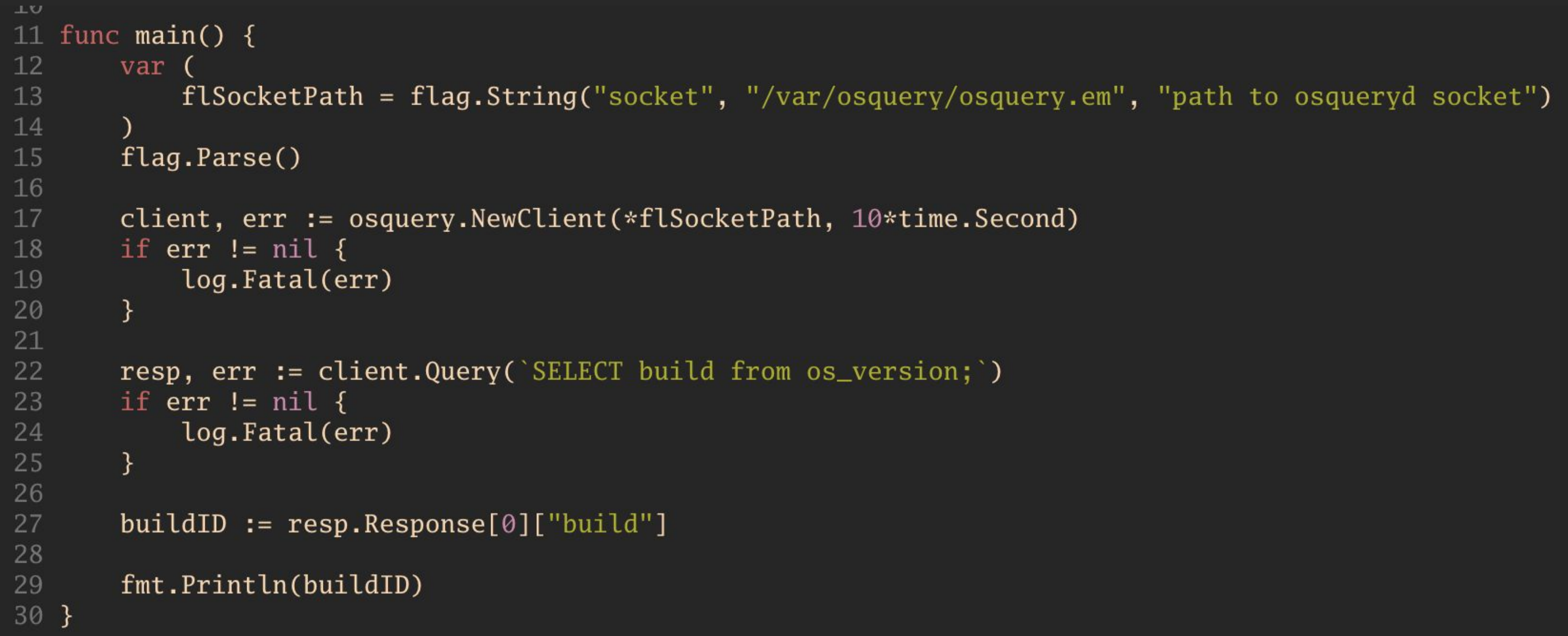
```
build:
  echo "$(shell pwd)/build/tutorial-extension.ext" > /tmp/extensions.load
  go build -o build/tutorial-extension.ext ./cmd/extension

osqueryd: build
  osqueryd \
    --extensions_autoload=/tmp/extensions.load \
    --pidfile=/tmp/osquery.pid \
    --database_path=/tmp/osquery.db \
    --extensions_socket=/tmp/osquery.sock \
    --config_refresh=60 \
    --config_plugin=gist

```

Query Plugin

Query Plugin



```go
func main() {
	var (
		flSocketPath = flag.String("socket", "/var/osquery/osquery.em", "path to osqueryd socket")
	)
	flag.Parse()

	client, err := osquery.NewClient(*flSocketPath, 10*time.Second)
	if err != nil {
		log.Fatal(err)
	}

	resp, err := client.Query(`SELECT build from os_version;`)
	if err != nil {
		log.Fatal(err)
	}

	buildID := resp.Response[0]["build"]

	fmt.Println(buildID)
}
```

Table
Config
Logger
Distributed
Query

GopherAcademy

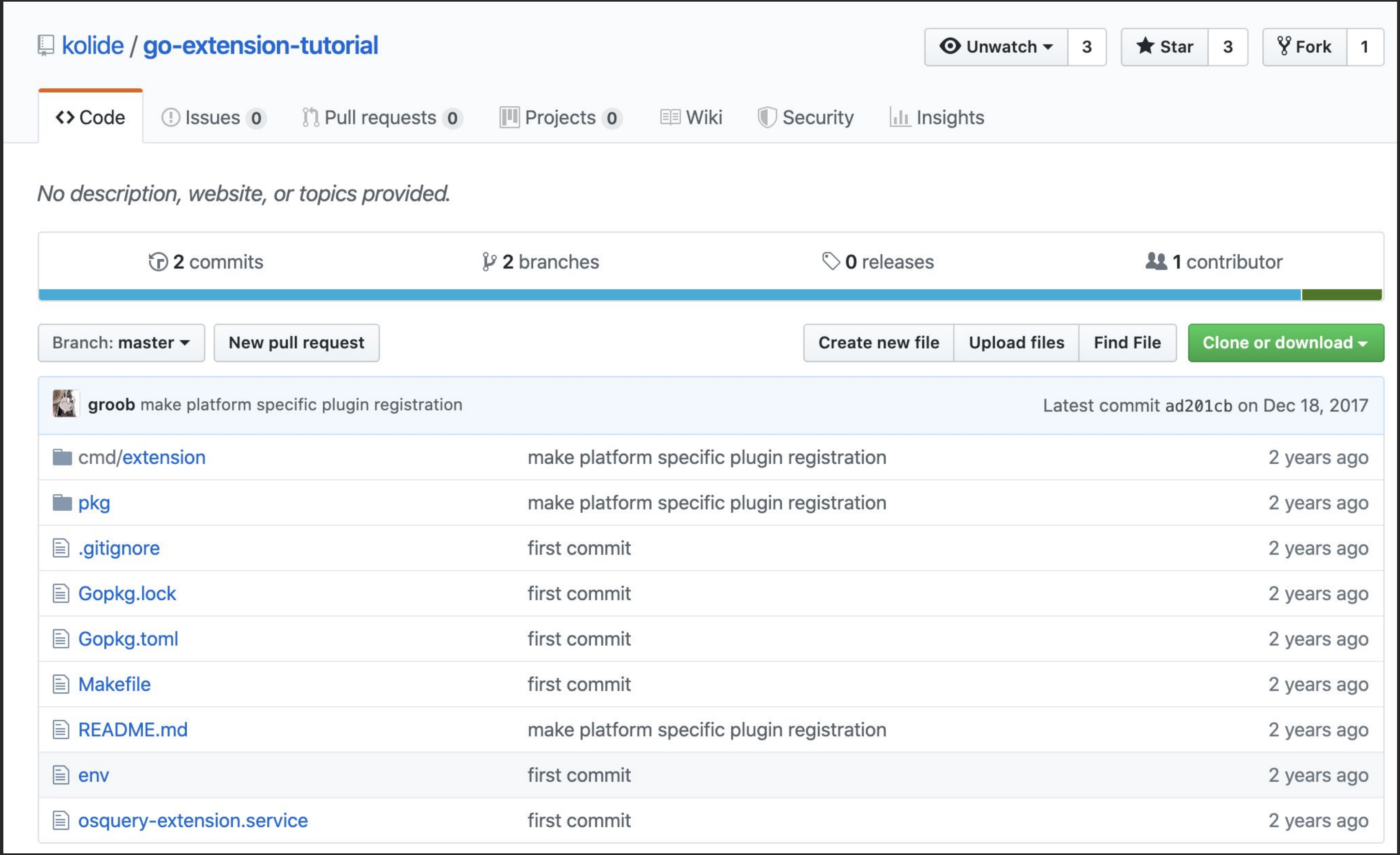
Gopher Academy Blog

Community Contributed Go Articles and Tutorials

Extending Osquery with Go

December 21, 2017
Contributed by Victor Vrantchan

What if you could use SQL to query any aspect of your infrastructure? Osquery, an open source instrumentation tool released by the Facebook security team allows you to do just that.

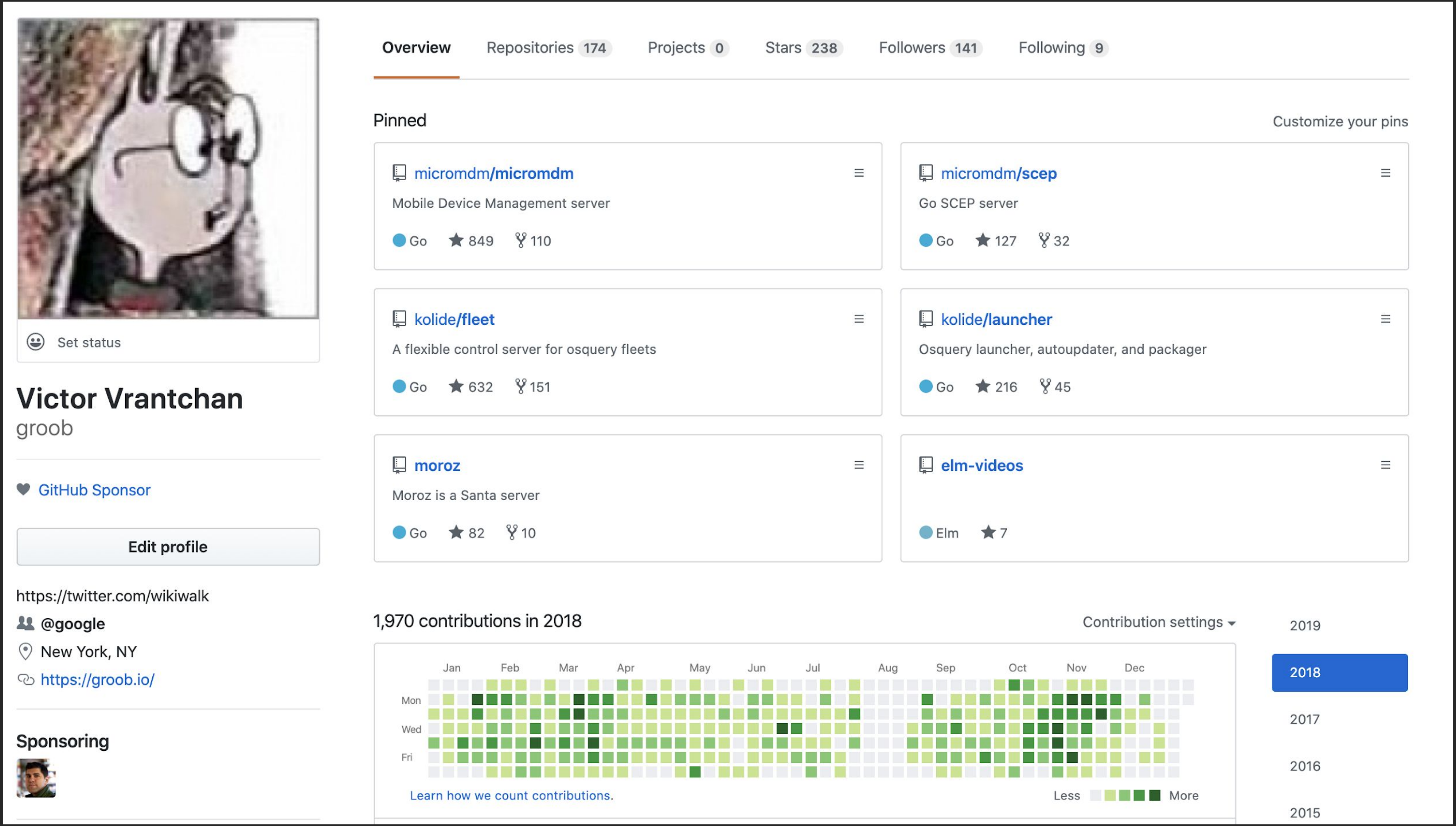

kolide / go-extension-tutorial

No description, website, or topics provided.

2 commits **2** branches **0** releases **1** contributor

groob make platform specific plugin registration — Latest commit `ad201cb` on Dec 18, 2017

cmd/extension	make platform specific plugin registration	2 years ago
pkg	make platform specific plugin registration	2 years ago
.gitignore	first commit	2 years ago
Gopkg.lock	first commit	2 years ago
Gopkg.toml	first commit	2 years ago
Makefile	first commit	2 years ago
README.md	make platform specific plugin registration	2 years ago
env	first commit	2 years ago
osquery-extension.service	first commit	2 years ago


Overview Repositories 174 Projects 0 Stars 238 Followers 141 Following 9

Victor Vrantchan
groob

Set status

GitHub Sponsor

Edit profile

https://twitter.com/wikiwalk
@google
New York, NY
https://groob.io/

Sponsoring

Pinned Customize your pins

micromdm/micromdm
Mobile Device Management server
Go ★ 849 110

micromdm/scep
Go SCEP server
Go ★ 127 32

kolide/fleet
A flexible control server for osquery fleets
Go ★ 632 151

kolide/launcher
Osquery launcher, autoupdater, and packager
Go ★ 216 45

moroz
Moroz is a Santa server
Go ★ 82 10

elm-videos
Elm ★ 7

1,970 contributions in 2018 Contribution settings

Learn how we count contributions.

2019
2018
2017
2016
2015