CLOUDQUERY & KOMISER NOTES

Bennett Wiley

CloudQuery Notes:

Why CloudQuery?

CloudQuery extracts, transforms, and loads configuration from cloud APIs to a variety of supported destinations such as databases, data lakes, or streaming platforms for further analysis.

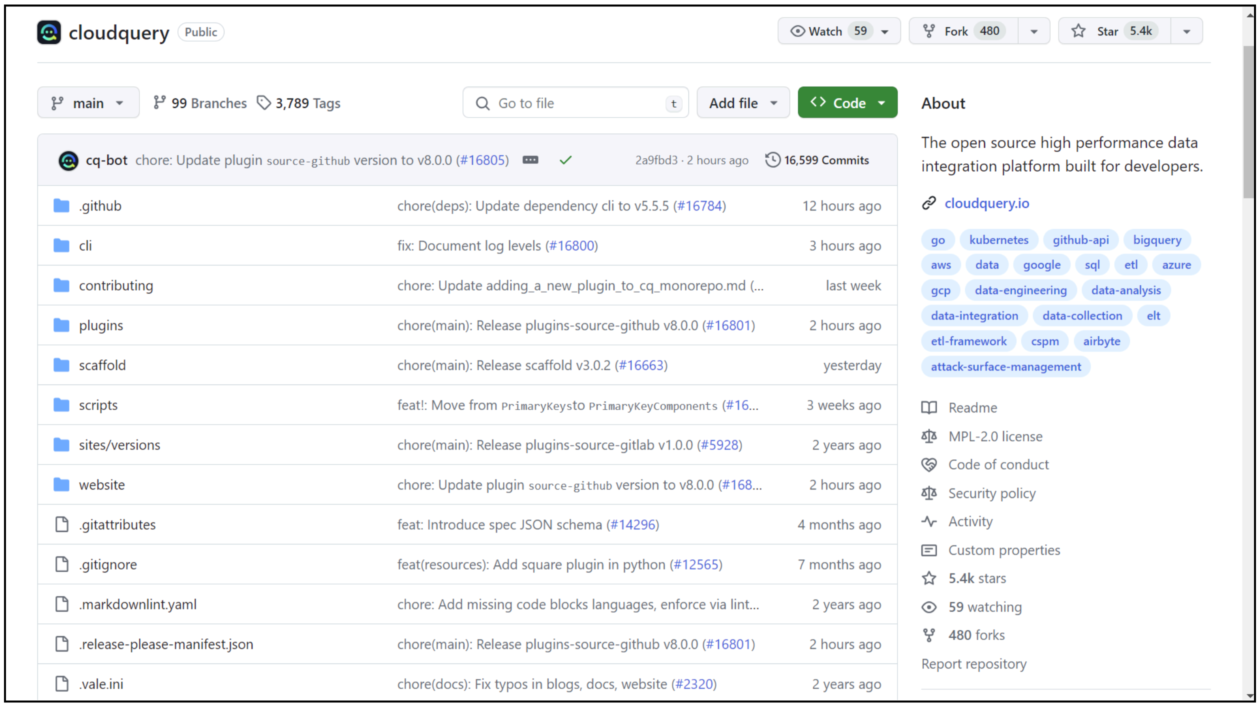
- ➔ Open source: Extensible plugin architecture. Contribute to our official plugins or develop your own with CloudQuery SDK.
- ➔ Blazing fast: CloudQuery is optimized for performance, utilizing the excellent Go concurrency model with light-weight goroutines.
- ➔ Deploy anywhere: CloudQuery plugins are single-binary executables and can be deployed and run anywhere.
- ➔ Pre-built queries: CloudQuery maintains a number of out-of-the-box security and compliance policies for cloud infrastructure.
- ➔ Eliminate data silos: Eliminate data silos across your organization, unifying data between security, infrastructure, marketing and finance teams.
- ➔ Unlimited scale: CloudQuery plugins are stateless and can be scaled horizontally on any platform, such as VMs, Kubernetes or batch jobs.

The CloudQuery Framework:

The CloudQuery framework consists of SDK and CLI are open source while plugins available under plugins are open core, hence not all contributions to plugins directory will be accepted if they are part of the commercial plugin offering.

- **SDK:** Stands for Software Development Kit and its a combination of instruments, libraries and documentation. Developers use SDK to generate software applications for a precise platform. CloudQuery uses SDK to provide the developers with the instruments and libraries to interchange the framework to help apply it for the apps and expand on the functionality.
- **CLI:** Stand for Command-Line Interface and its an interface made up of text to work with software by applying commands in a terminal and or command prompt like PowerShell. CloudQuery would use CLI to make users carry out many tasks and operations that are similar to the framework from the command line. For example querying cloud resources, managing configurations, and executing commands.

This image below shows CloudQuery's entire github repository. Key things to take from this image is how they organize the work in the first few files. The .github folder only looks at github issues like styles and workflow. CLI folder focuses on the Command Line Interface. The Contributing folder looks at adding and developing environments, plug-ins and resources. Plugins folder focuses on the plug-in's destination and source. Scaffold folder helps the CLI scaffold (bootstrap) new source or destination plugins. Scripts folder focuses on missing tables, sentry alerts, table diff., and also workflows. Sites/versions folder hold the versions on the vercel json. Last the website folder only hold programs and info for their website to display



cloudquery Public

Watch 59 · Fork 480 · Star 5.4k

main · 99 Branches · 3,789 Tags · Go to file · Add file · Code

cq-bot chore: Update plugin `source-github` version to v8.0.0 (#16805) ✓ 2a9fbd3 · 2 hours ago · 16,599 Commits

.github	chore(deps): Update dependency cli to v5.5.5 (#16784)	12 hours ago
cli	fix: Document log levels (#16800)	3 hours ago
contributing	chore: Update adding_a_new_plugin_to_cq_monorepo.md (...	last week
plugins	chore(main): Release plugins-source-github v8.0.0 (#16801)	2 hours ago
scaffold	chore(main): Release scaffold v3.0.2 (#16663)	yesterday
scripts	feat!: Move from `PrimaryKeys`to `PrimaryKeyComponents` (#16...	3 weeks ago
sites/versions	chore(main): Release plugins-source-gitlab v1.0.0 (#5928)	2 years ago
website	chore: Update plugin `source-github` version to v8.0.0 (#168...	2 hours ago
.gitattributes	feat: Introduce spec JSON schema (#14296)	4 months ago
.gitignore	feat(resources): Add square plugin in python (#12565)	7 months ago
.markdownlint.yaml	chore: Add missing code blocks languages, enforce via lint...	2 years ago
.release-please-manifest.json	chore(main): Release plugins-source-github v8.0.0 (#16801)	2 hours ago
.vale.ini	chore(docs): Fix typos in blogs, docs, website (#2320)	2 years ago

About

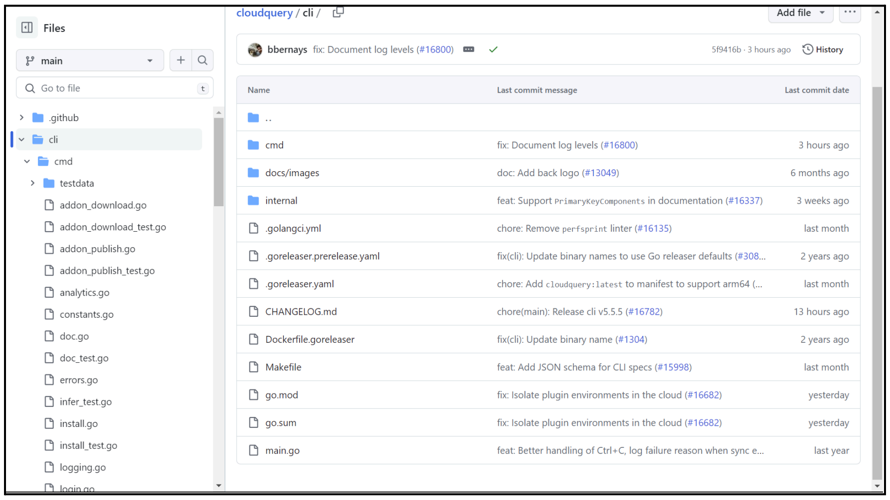
The open source high performance data integration platform built for developers.

cloudquery.io

go · kubernetes · github-api · bigquery · aws · data · google · sql · etl · azure · gcp · data-engineering · data-analysis · data-integration · data-collection · elt · etl-framework · cspm · airbyte · attack-surface-management

Readme
MPL-2.0 license
Code of conduct
Security policy
Activity
Custom properties
5.4k stars
59 watching
480 forks
Report repository

The image below also shows CloudQuery's repository, but notice that it focuses on the CLI folder. It's important to see how the CLI folder consists of a CMD folder, docs/images folder, and an internal folder.



Files

main · Go to file

.github
cli
cmd
testdata
addon_download.go
addon_download_test.go
addon_publish.go
addon_publish_test.go
analytics.go
constants.go
doc.go
doc_test.go
errors.go
infer_test.go
install.go
install_test.go
logging.go
login.go

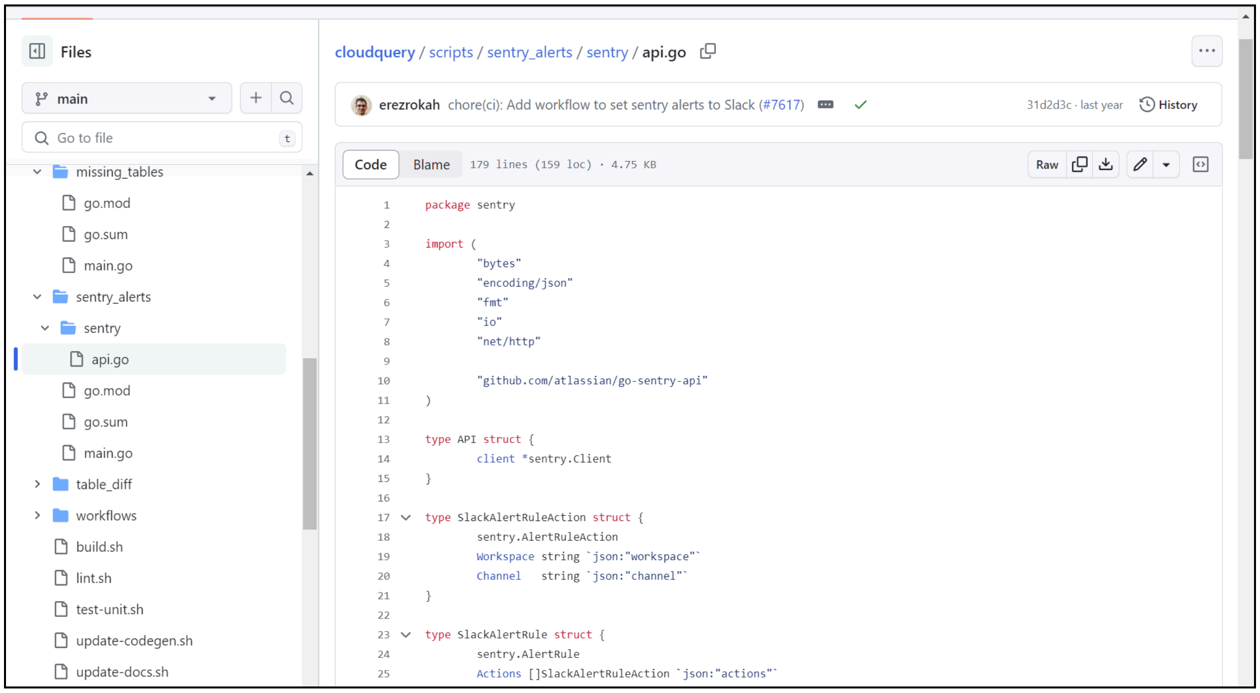
cloudquery / cli /

Add file

bbernays fix: Document log levels (#16800) ✓ 5f9416b · 3 hours ago · History

Name	Last commit message	Last commit date
..		
cmd	fix: Document log levels (#16800)	3 hours ago
docs/images	doc: Add back logo (#13049)	6 months ago
internal	feat: Support `PrimaryKeyComponents` in documentation (#16337)	3 weeks ago
.golangci.yml	chore: Remove `perfsprint` linter (#16135)	last month
.goreleaser.prerelease.yaml	fix(cli): Update binary names to use Go releaser defaults (#308...	2 years ago
.goreleaser.yaml	chore: Add `cloudquery:latest` to manifest to support arm64 (...	last month
CHANGELOG.md	chore(main): Release cli v5.5.5 (#16782)	13 hours ago
Dockerfile.goreleaser	fix(cli): Update binary name (#1304)	2 years ago
Makefile	feat: Add JSON schema for CLI specs (#15998)	last month
go.mod	fix: Isolate plugin environments in the cloud (#16682)	yesterday
go.sum	fix: Isolate plugin environments in the cloud (#16682)	yesterday
main.go	feat: Better handling of Ctrl+C, log failure reason when sync e...	last year

The image below I wanted to show is just some code to show how CloudQuery organizes and stores its code. The code here is from the scripts folder and it's under the sentry_alerts titled api.go.



```go
package sentry

import (
	"bytes"
	"encoding/json"
	"fmt"
	"io"
	"net/http"

	"github.com/atlassian/go-sentry-api"
)

type API struct {
	client *sentry.Client
}

type SlackAlertRuleAction struct {
	sentry.AlertRuleAction
	Workspace string `json:"workspace"`
	Channel   string `json:"channel"`
}

type SlackAlertRule struct {
	sentry.AlertRule
	Actions []SlackAlertRuleAction `json:"actions"`
```

Komiser Notes:

Why Komiser?

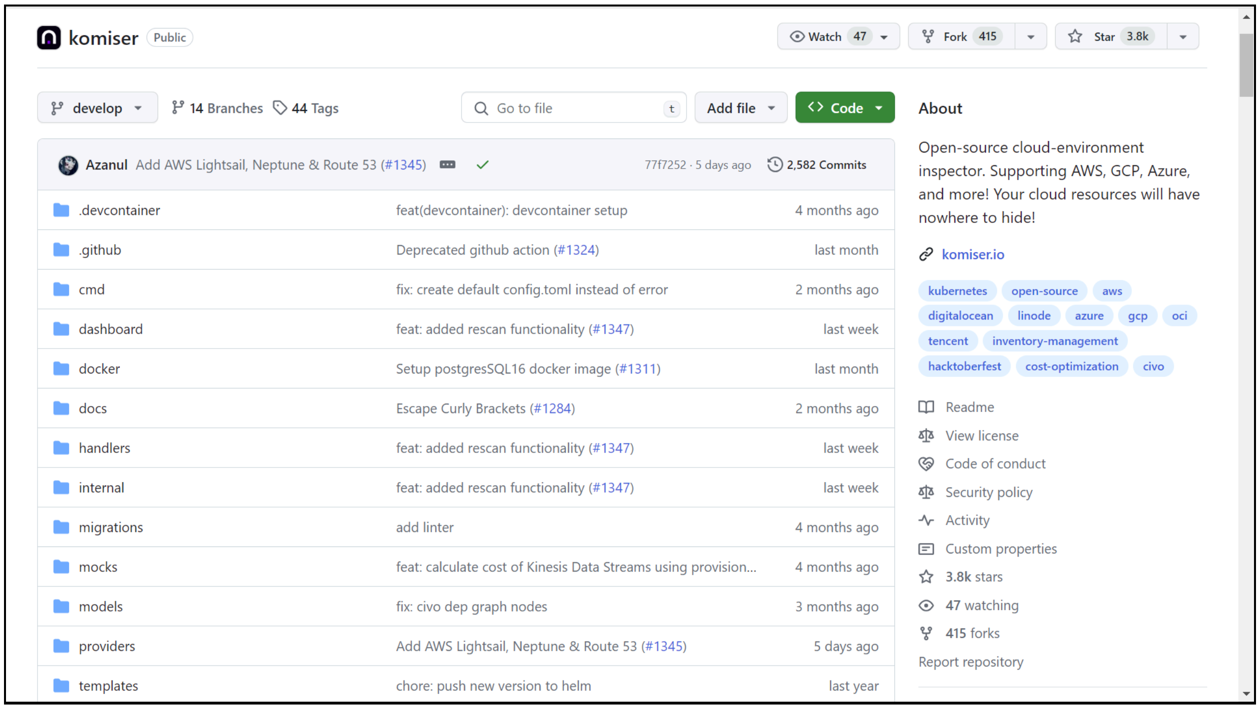
Komiser is an open source project created to analyze and manage cloud cost, usage, security and governance all in one place. With komiser you can also:

- ➔ Build an inventory of your cloud infrastructure assets.
- ➔ Control your resource usage and gain visibility across all used services to achieve maximum cost-effectiveness.
- ➔ Detect potential vulnerabilities that could put your cloud environment at risk.
- ➔ Get a deep understanding of how you spend on AWS, Azure, GCP, Civo, DigitalOcean and OCI.
- ➔ Uncover idle and untagged resources, ensuring that no resource goes unnoticed.

The Komiser Framework:

Komiser is written in Golang and is Elv2 licensed. Komiser can build a detailed cloud asset inventory, no matter the location or platform. This includes AWS, GCP, Azure, ect. Komiser also facilitates a centralized inventory view. Komiser configuration holds credentials to cloud providers through a config.toml file.

The image below shows the Komiser github repository. Looking at it we can see a lot of similarities with the CloudQuery repository, but it seems Komiser has more files. This may seem tedious, but I noticed it was cleaner and more straightforward to find what you are looking for. It's simpler because you make a file for each purpose, rather than trying to make a broad file that holds more information. Similar files that we've seen before from CloudQuery include .github file and cmd file. New folders that I found interesting were docs, which held even more folders, but all relating to documents. The Internal folder holds an api/v1 and config folder. The Model folder is also very interesting and could be useful for our project.



Azanul Add AWS Lightsail, Neptune & Route 53 (#1345) ✓ 77f7252 · 5 days ago 2,582 Commits

.devcontainer	feat(devcontainer): devcontainer setup	4 months ago
.github	Deprecated github action (#1324)	last month
cmd	fix: create default config.toml instead of error	2 months ago
dashboard	feat: added rescan functionality (#1347)	last week
docker	Setup postgresSQL16 docker image (#1311)	last month
docs	Escape Curly Brackets (#1284)	2 months ago
handlers	feat: added rescan functionality (#1347)	last week
internal	feat: added rescan functionality (#1347)	last week
migrations	add linter	4 months ago
mocks	feat: calculate cost of Kinesis Data Streams using provision...	4 months ago
models	fix: civo dep graph nodes	3 months ago
providers	Add AWS Lightsail, Neptune & Route 53 (#1345)	5 days ago
templates	chore: push new version to helm	last year

About

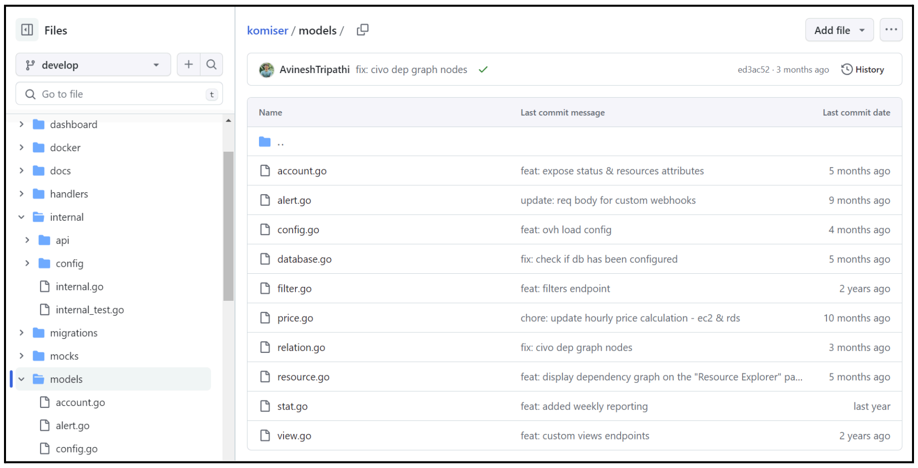
Open-source cloud-environment inspector. Supporting AWS, GCP, Azure, and more! Your cloud resources will have nowhere to hide!

komiser.io

kubernetes open-source aws digitalocean linode azure gcp oci tencent inventory-management hacktoberfest cost-optimization civo

Readme
View license
Code of conduct
Security policy
Activity
Custom properties
3.8k stars
47 watching
415 forks
Report repository

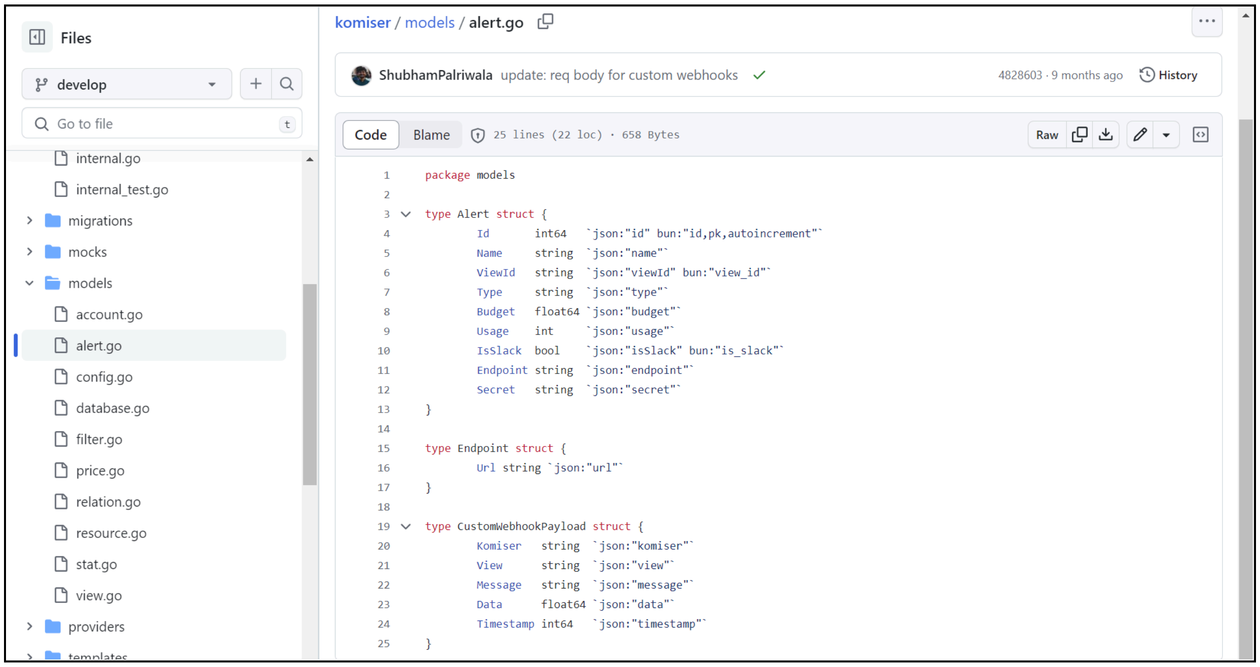
The image below is a closer look at the Model folder and it holds a ton of *".go"* files. These types of files are a simple UTF-8 text file with a *".go"* extension. You can see each one has its purpose like account, alert, and price.



AvineshTripathi fix: civo dep graph nodes ✓ ed3ac52 · 3 months ago History

Name	Last commit message	Last commit date
..		
account.go	feat: expose status & resources attributes	5 months ago
alert.go	update: req body for custom webhooks	9 months ago
config.go	feat: ovh load config	4 months ago
database.go	fix: check if db has been configured	5 months ago
filter.go	feat: filters endpoint	2 years ago
price.go	chore: update hourly price calculation - ec2 & rds	10 months ago
relation.go	fix: civo dep graph nodes	3 months ago
resource.go	feat: display dependency graph on the "Resource Explorer" pa...	5 months ago
stat.go	feat: added weekly reporting	last year
view.go	feat: custom views endpoints	2 years ago

The last image is another one that shows some code, but for the Komiser repository. This code is under the models file and is labeled alert.go. You can see what the program does and how it's organized like a list starting with Id and ending with secret.



```go
package models

type Alert struct {
	Id       int64   `json:"id" bun:"id,pk,autoincrement"`
	Name     string  `json:"name"`
	ViewId   string  `json:"viewId" bun:"view_id"`
	Type     string  `json:"type"`
	Budget   float64 `json:"budget"`
	Usage    int     `json:"usage"`
	IsSlack  bool    `json:"isSlack" bun:"is_slack"`
	Endpoint string  `json:"endpoint"`
	Secret   string  `json:"secret"`
}

type Endpoint struct {
	Url string `json:"url"`
}

type CustomWebhookPayload struct {
	Komiser   string  `json:"komiser"`
	View      string  `json:"view"`
	Message   string  `json:"message"`
	Data      float64 `json:"data"`
	Timestamp int64   `json:"timestamp"`
}
```