Swiss Army Knife for SaaS Products Build with Go

•••

Huseyin BABAL
Cloud Engineer

Choosing Right Tools

It is important to choose right tools to build Go project for the stability of project

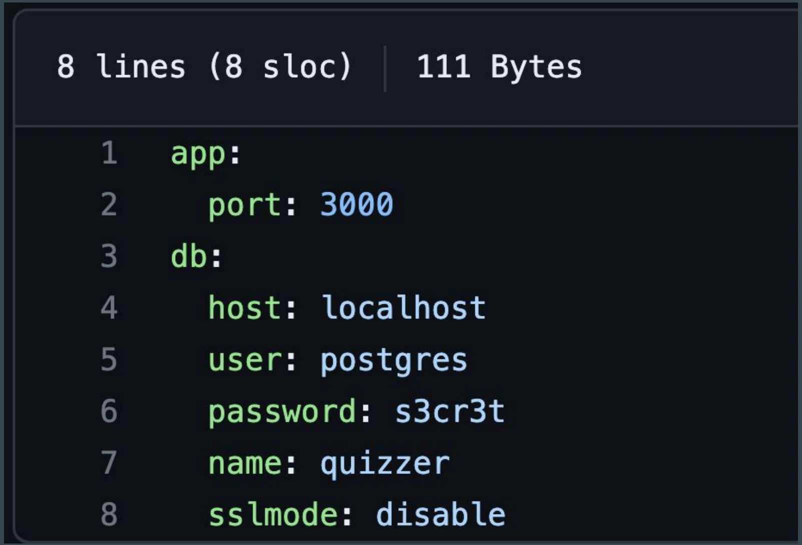
- Configuration
- Logging
- Artifact Generation
- Code Quality Checker
- Vulnerability Scanning
- CI/CD
- Infrastructure as Code
- Payment System

Golang Modular Projects

```
go mod init github.com/huseyinbabal/quizzer
```

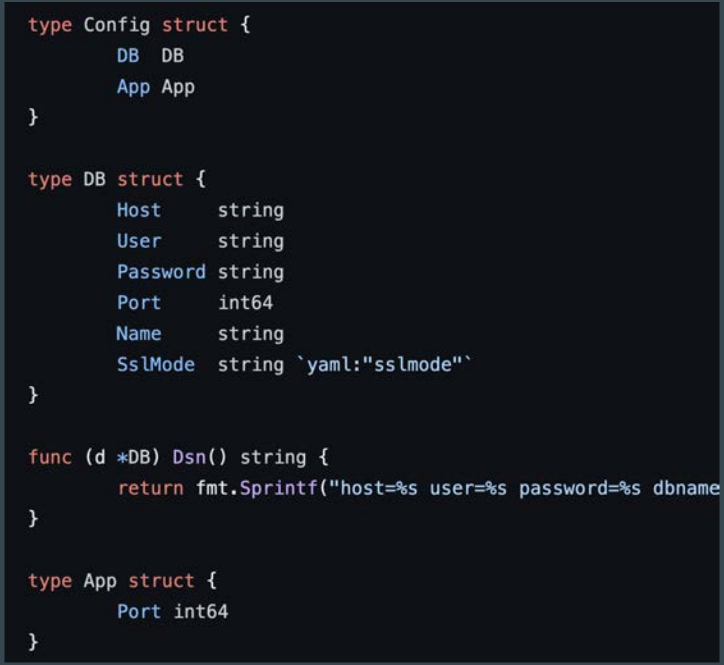
Configuration

Koanf (https://github.com/knadh/koanf)



```
8 lines (8 sloc) | 111 Bytes
```

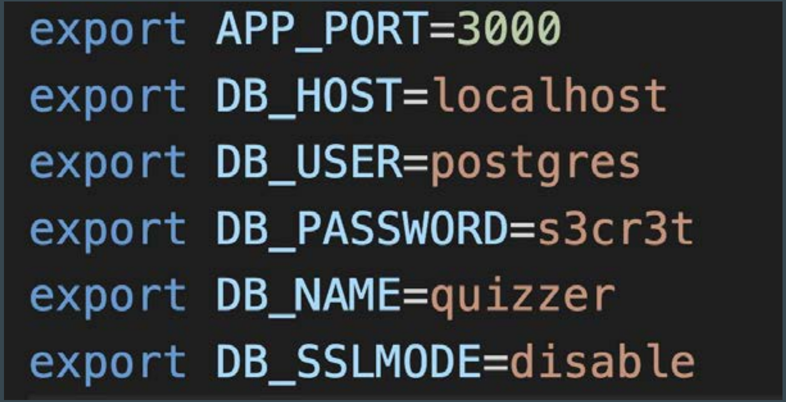
```yaml
app:
  port: 3000
db:
  host: localhost
  user: postgres
  password: s3cr3t
  name: quizzer
  sslmode: disable
```



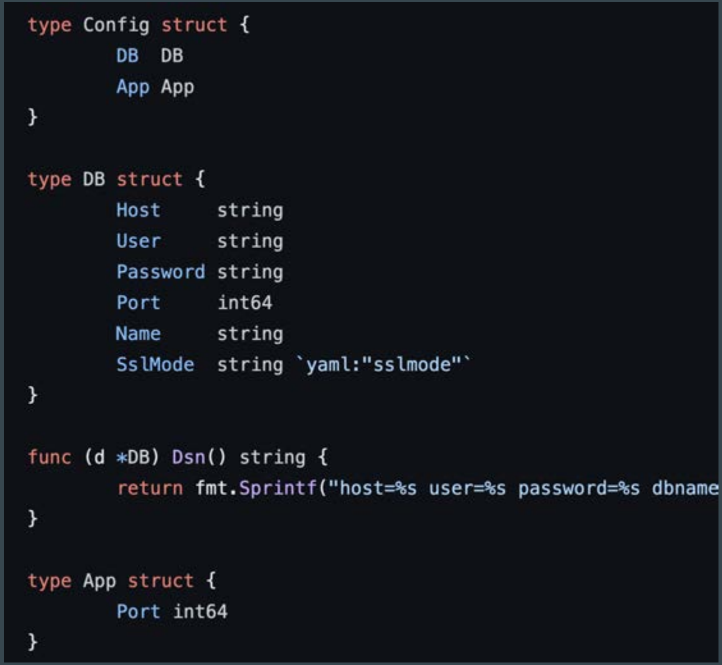
```go
type Config struct {
	DB  DB
	App App
}

type DB struct {
	Host     string
	User     string
	Password string
	Port     int64
	Name     string
	SslMode  string `yaml:"sslmode"`
}

func (d *DB) Dsn() string {
	return fmt.Sprintf("host=%s user=%s password=%s dbname
}

type App struct {
	Port int64
}
```

Configuration

Twelve-Factor App (https://12factor.net/)



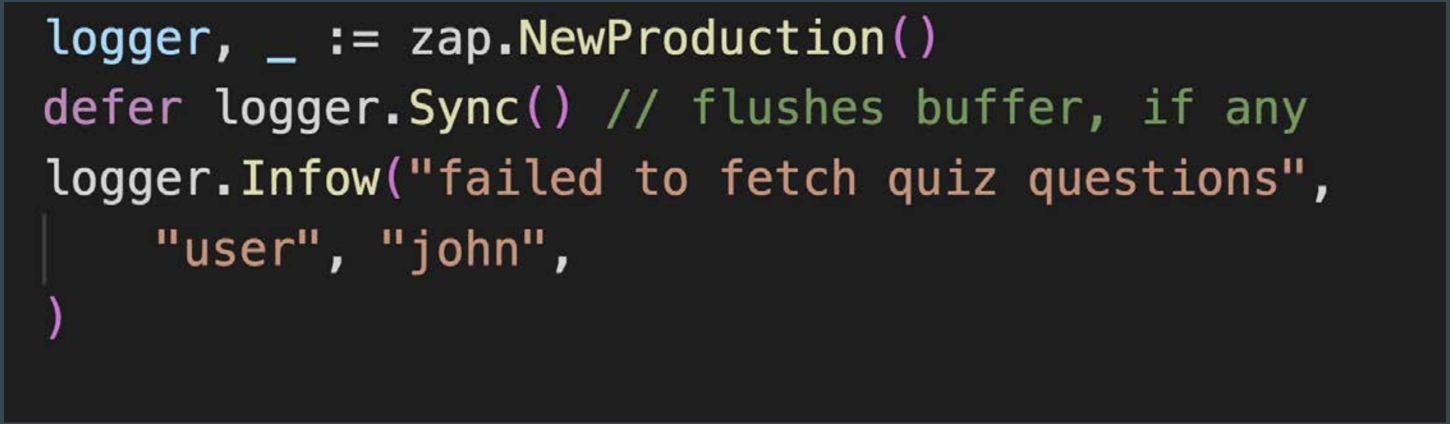
```
export APP_PORT=3000
export DB_HOST=localhost
export DB_USER=postgres
export DB_PASSWORD=s3cr3t
export DB_NAME=quizzer
export DB_SSLMODE=disable
```



```
type Config struct {
	DB  DB
	App App
}

type DB struct {
	Host     string
	User     string
	Password string
	Port     int64
	Name     string
	SslMode  string `yaml:"sslmode"`
}

func (d *DB) Dsn() string {
	return fmt.Sprintf("host=%s user=%s password=%s dbname
}

type App struct {
	Port int64
}
```

Logging

Zap (https://github.com/uber-go/zap)

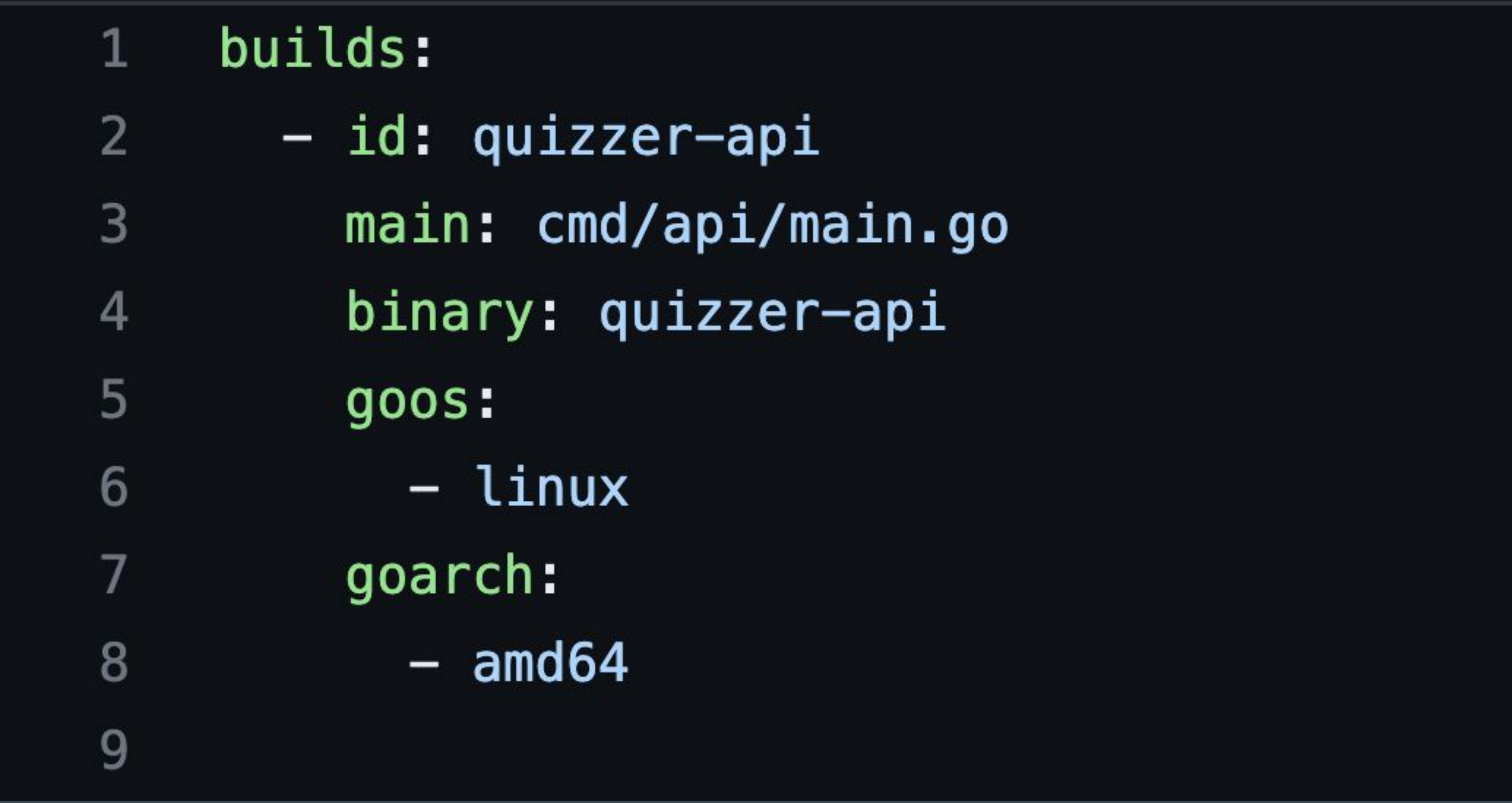


```go
logger, _ := zap.NewProduction()
defer logger.Sync() // flushes buffer, if any
logger.Infow("failed to fetch quiz questions",
    "user", "john",
)
```

Go Releaser

It helps you to build cross-platform artifacts and release them to various platforms.

Create .goreleaser.yml and run goreleaser build/release --clean



```yaml
builds:
  - id: quizzer-api
    main: cmd/api/main.go
    binary: quizzer-api
    goos:
      - linux
    goarch:
      - amd64

```

Docker Image Generation

```yaml
dockers:
  - id: quizzer-api
    goos: linux
    goarch: amd64
    ids:
      - quizzer-api
    image_templates:
      - "ghcr.io/huseyinbabal/quizzer-api:{{ .Tag }}"
    build_flag_templates:
      - "--build-arg=module=quizzer-api"
      - "--label=org.opencontainers.image.source=https://github.com/huseyinbabal/quizzer-api"
    extra_files:
      - "config.dist.yml"
    skip_push: false
```

Code Quality Check

Golangci-lint (https://golangci-lint.run/)

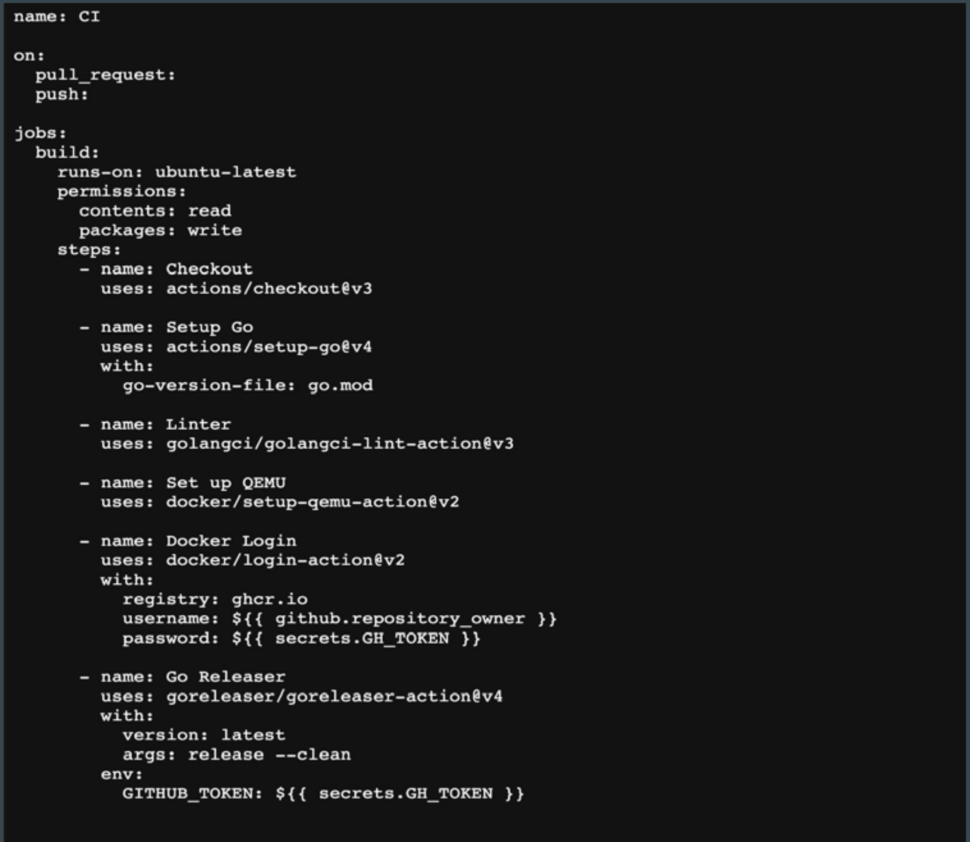
Create .golangci.yml and run golangci-lint run

Linters https://golangci-lint.run/usage/linters/

Continuous Integration

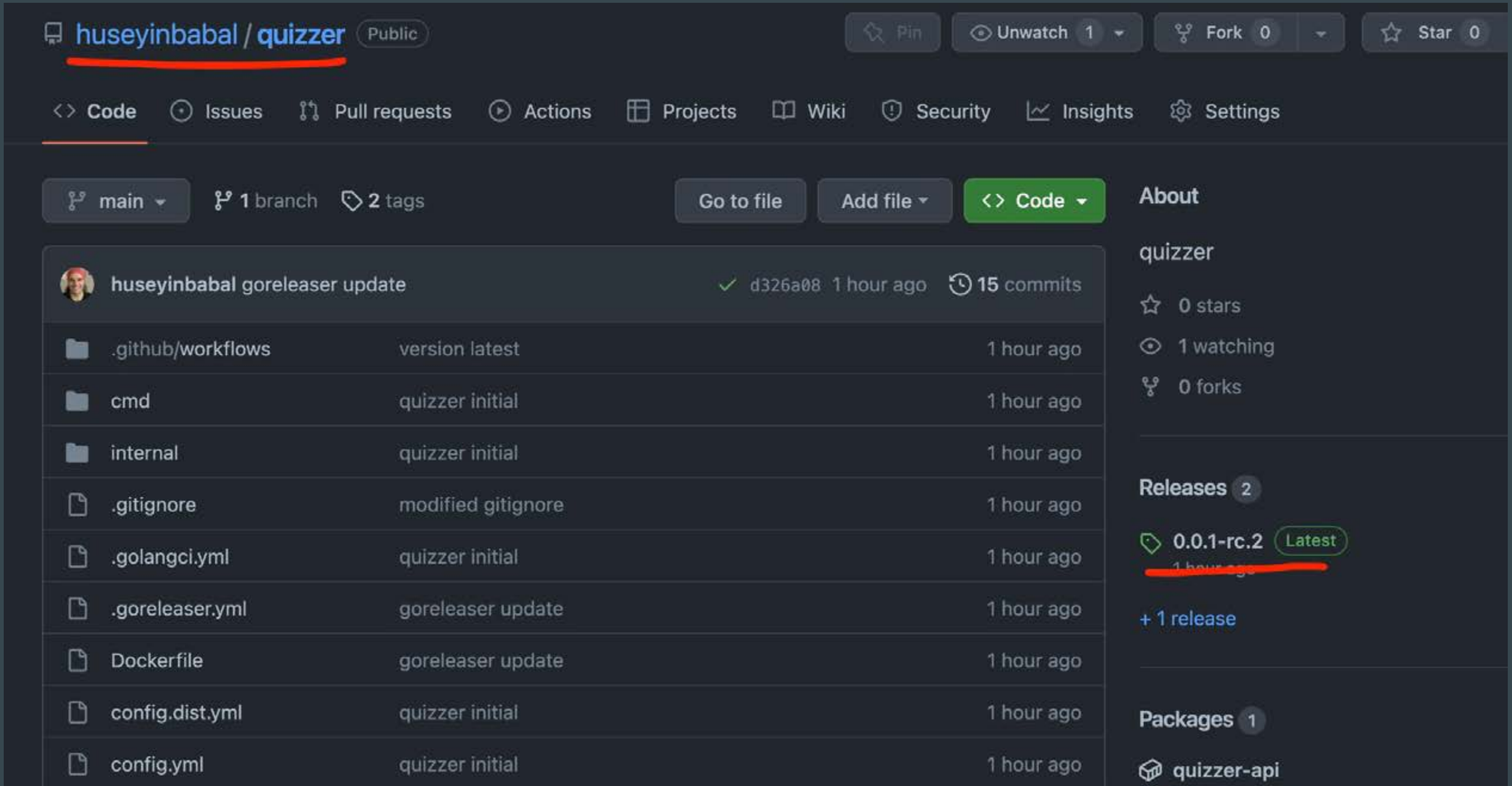
With the help of Github Actions, we can test, verify project also generate artifacts to use in production example

GH Example



```yaml
name: CI

on:
  pull_request:
  push:

jobs:
  build:
    runs-on: ubuntu-latest
    permissions:
      contents: read
      packages: write
    steps:
      - name: Checkout
        uses: actions/checkout@v3

      - name: Setup Go
        uses: actions/setup-go@v4
        with:
          go-version-file: go.mod

      - name: Linter
        uses: golangci/golangci-lint-action@v3

      - name: Set up QEMU
        uses: docker/setup-qemu-action@v2

      - name: Docker Login
        uses: docker/login-action@v2
        with:
          registry: ghcr.io
          username: ${{ github.repository_owner }}
          password: ${{ secrets.GH_TOKEN }}

      - name: Go Releaser
        uses: goreleaser/goreleaser-action@v4
        with:
          version: latest
          args: release --clean
        env:
          GITHUB_TOKEN: ${{ secrets.GH_TOKEN }}
```

Output



huseyinbabal / quizzer Public

Pin | Unwatch 1 | Fork 0 | Star 0

Code | Issues | Pull requests | Actions | Projects | Wiki | Security | Insights | Settings

main | 1 branch | 2 tags | Go to file | Add file | Code

huseyinbabal goreleaser update ✓ d326a08 1 hour ago 15 commits

.github/workflows	version latest	1 hour ago
cmd	quizzer initial	1 hour ago
internal	quizzer initial	1 hour ago
.gitignore	modified gitignore	1 hour ago
.golangci.yml	quizzer initial	1 hour ago
.goreleaser.yml	goreleaser update	1 hour ago
Dockerfile	goreleaser update	1 hour ago
config.dist.yml	quizzer initial	1 hour ago
config.yml	quizzer initial	1 hour ago

About

quizzer

0 stars

1 watching

0 forks

Releases 2

0.0.1-rc.2 Latest

1 hour ago

+ 1 release

Packages 1

quizzer-api

Where to ship those artifacts?

We can easily deploy those artifacts to Kubernetes. However, we need to handle IaC first to have up and running K8s environment

Terraform Cloud is a good candidate to connect your repo to Terraform Cloud and maintain the underlying infrastructure

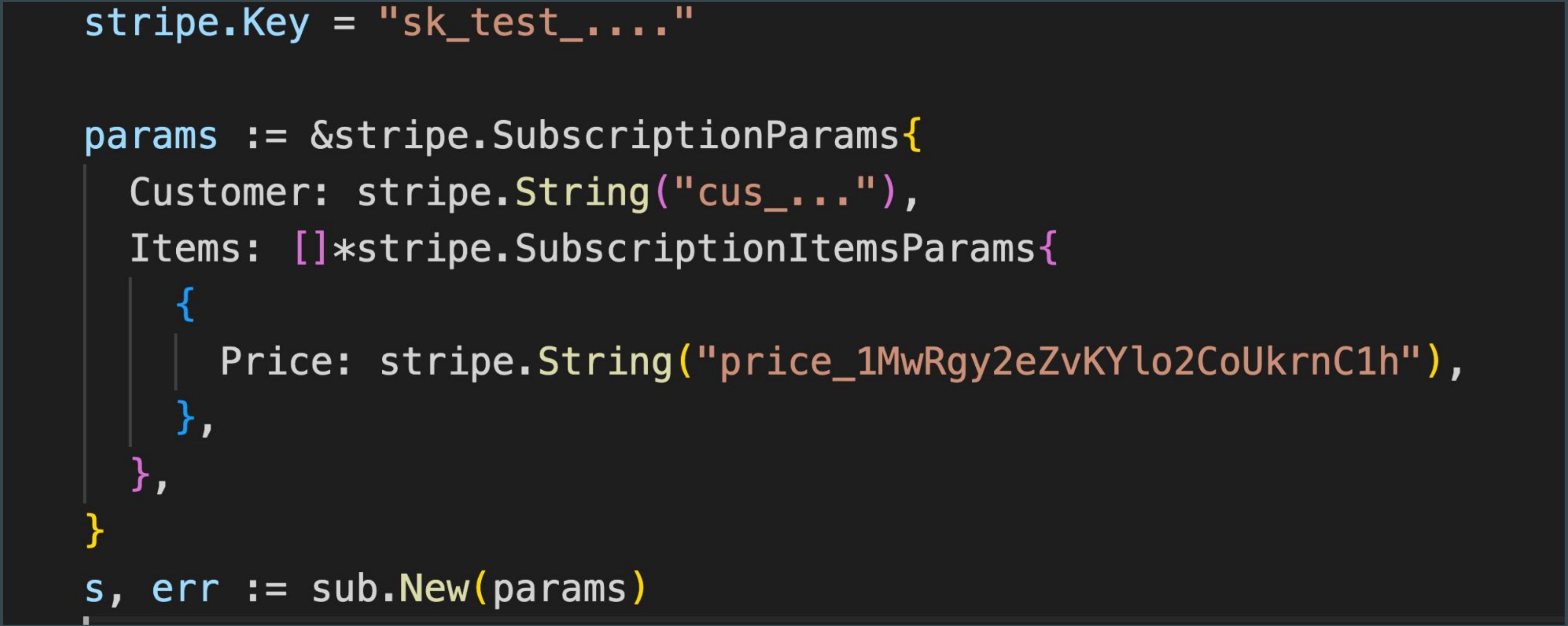
Payment System

Stripe

Subscription & Subscription Item (https://stripe.com/docs/api/subscriptions)

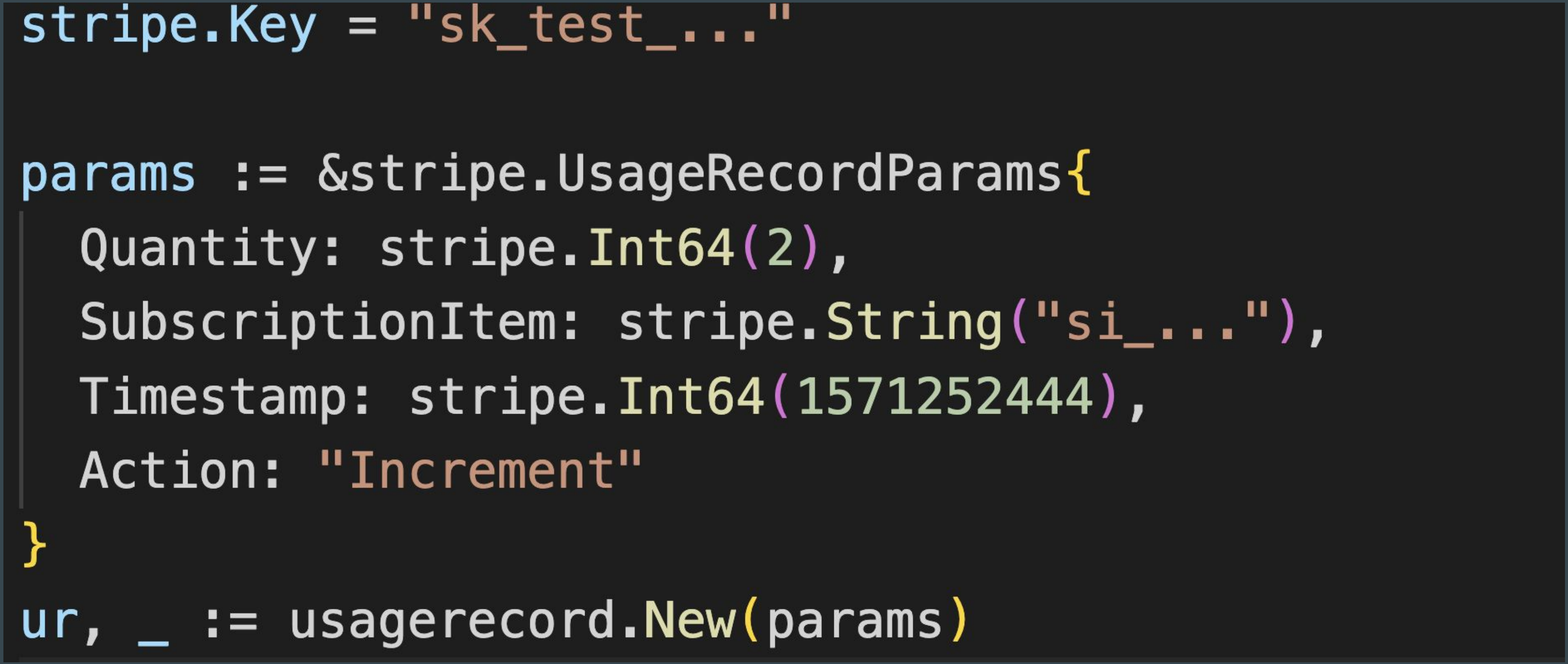
Metered Billing (https://stripe.com/docs/api/usage_records)

Subscribe Customer



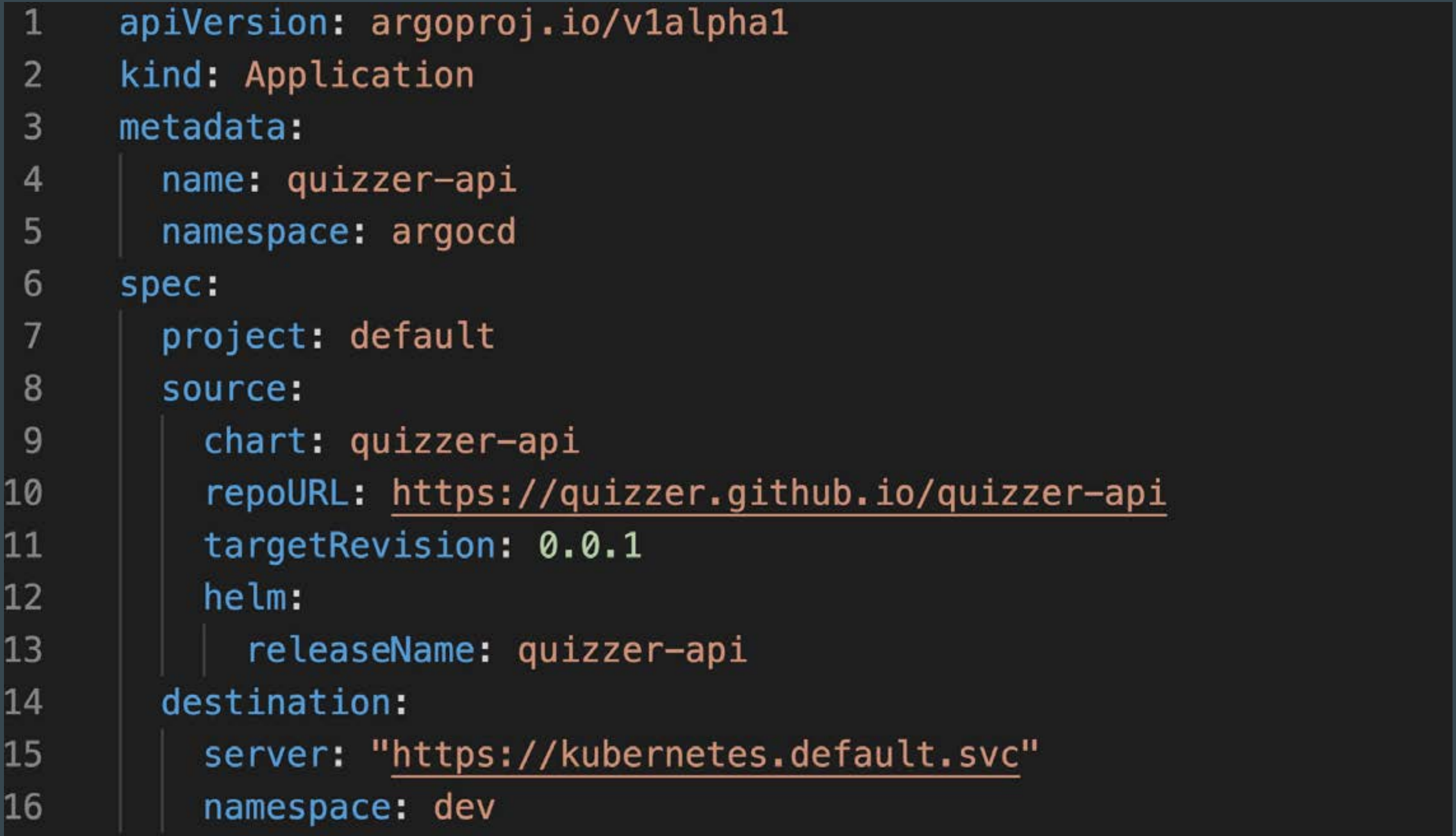
```go
stripe.Key = "sk_test_...."

params := &stripe.SubscriptionParams{
  Customer: stripe.String("cus_..."),
  Items: []*stripe.SubscriptionItemsParams{
    {
      Price: stripe.String("price_1MwRgy2eZvKYlo2CoUkrnC1h"),
    },
  },
}
s, err := sub.New(params)
```

Metered Billing, Create Usage Record



```go
stripe.Key = "sk_test_..."

params := &stripe.UsageRecordParams{
  Quantity: stripe.Int64(2),
  SubscriptionItem: stripe.String("si_..."),
  Timestamp: stripe.Int64(1571252444),
  Action: "Increment"
}
ur, _ := usagerecord.New(params)
```

ArgoCD



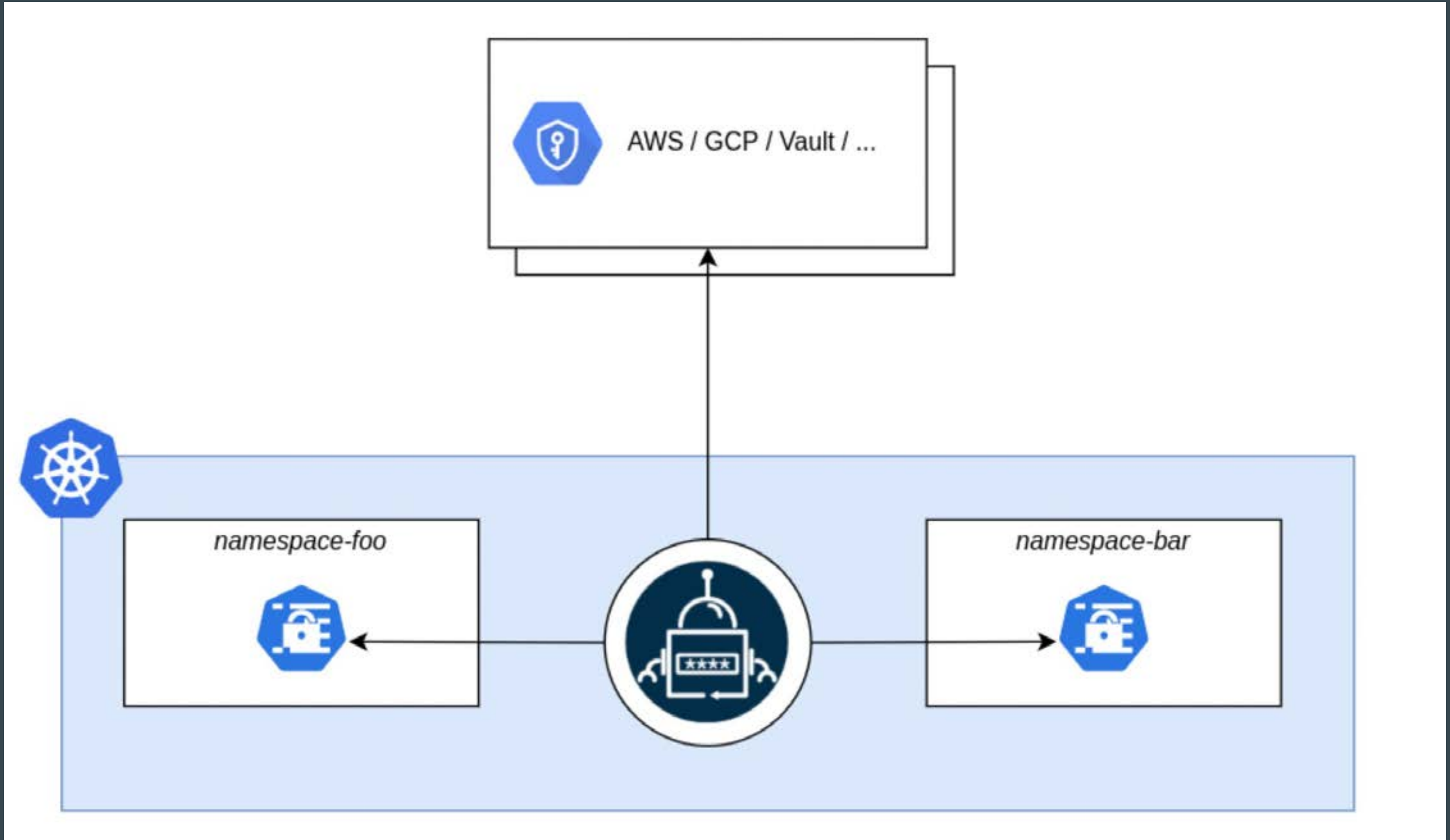
```yaml
apiVersion: argoproj.io/v1alpha1
kind: Application
metadata:
  name: quizzer-api
  namespace: argocd
spec:
  project: default
  source:
    chart: quizzer-api
    repoURL: https://quizzer.github.io/quizzer-api
    targetRevision: 0.0.1
    helm:
      releaseName: quizzer-api
  destination:
    server: "https://kubernetes.default.svc"
    namespace: dev
```

Confidential Data

Secrets can be passed to application via environment variables by syncing from secret resources.

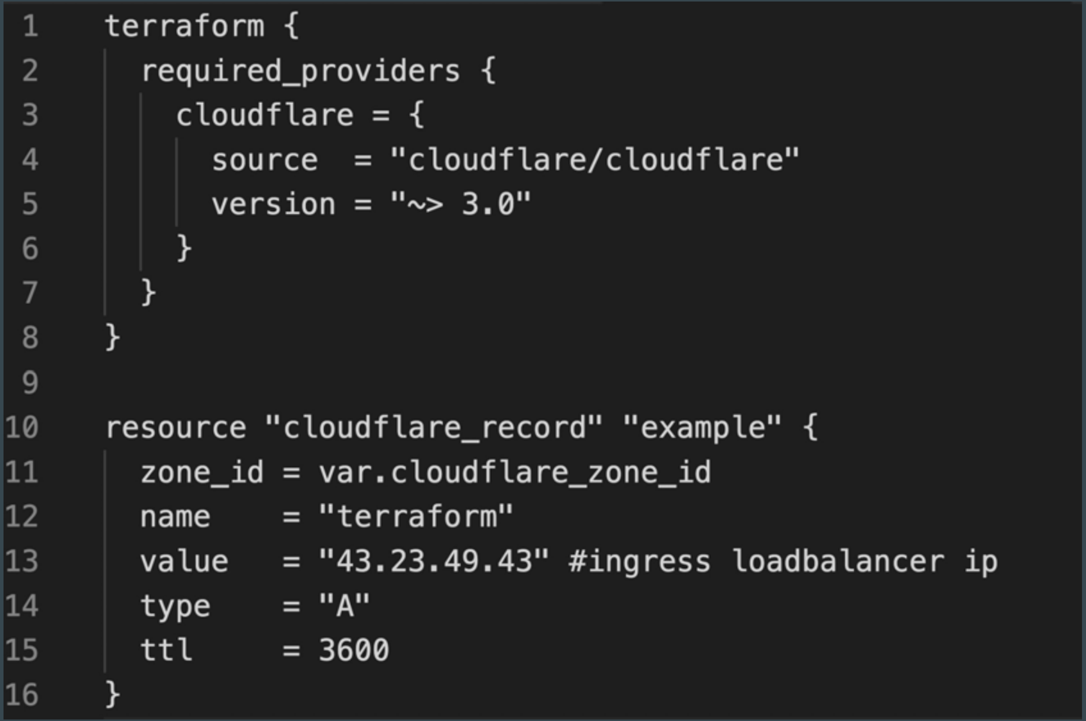
Those secret resource can be managed by External Secrets project

https://external-secrets.io/v0.8.1/


AWS / GCP / Vault / ...
namespace-foo
namespace-bar

Public Access

It can be handled by Cloudflare, hopefully they have TF Provider



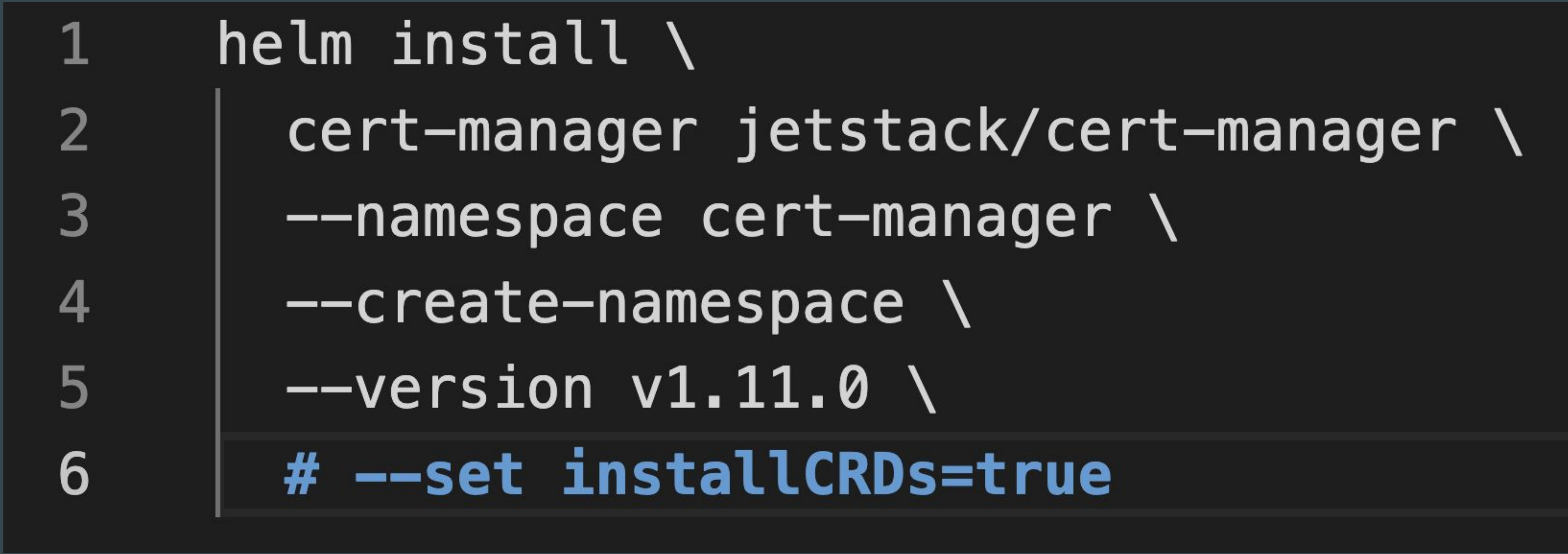
```
terraform {
  required_providers {
    cloudflare = {
      source  = "cloudflare/cloudflare"
      version = "~> 3.0"
    }
  }
}

resource "cloudflare_record" "example" {
  zone_id = var.cloudflare_zone_id
  name    = "terraform"
  value   = "43.23.49.43" #ingress loadbalancer ip
  type    = "A"
  ttl     = 3600
}
```

Certificate Management

Cert-Manager helps you to manage your certificates in k8s, it has good integration with letsencrypt

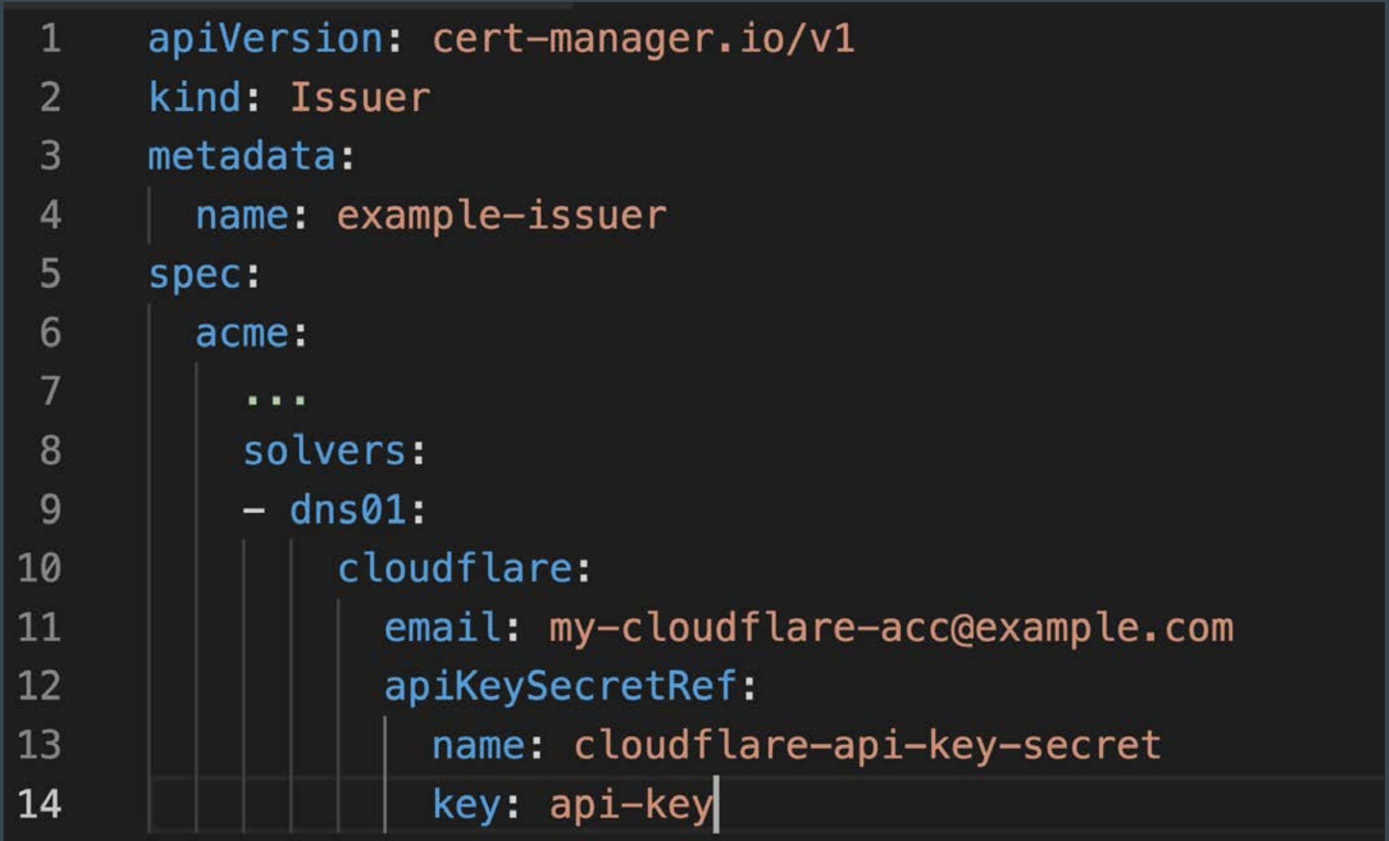
https://cert-manager.io/docs/installation/helm/

Helm install



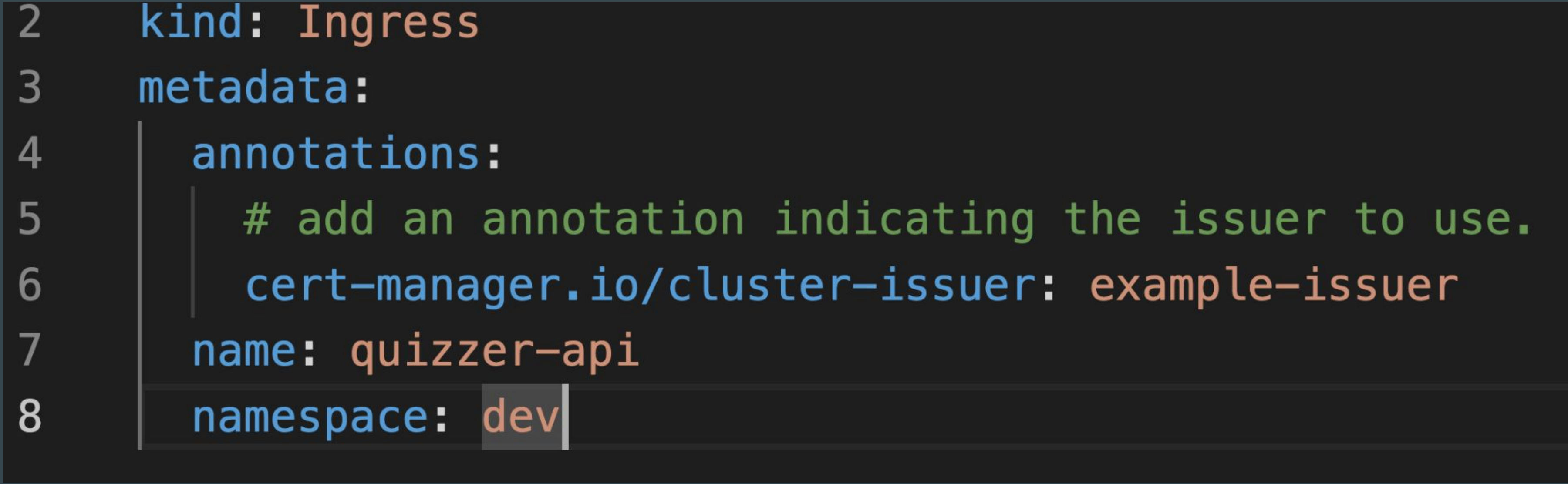
```
helm install \
  cert-manager jetstack/cert-manager \
  --namespace cert-manager \
  --create-namespace \
  --version v1.11.0 \
  # --set installCRDs=true
```

Cert Configuration



```yaml
apiVersion: cert-manager.io/v1
kind: Issuer
metadata:
  name: example-issuer
spec:
  acme:
    ...
    solvers:
    - dns01:
        cloudflare:
          email: my-cloudflare-acc@example.com
          apiKeySecretRef:
            name: cloudflare-api-key-secret
            key: api-key
```

Ingress integration



```yaml
kind: Ingress
metadata:
  annotations:
    # add an annotation indicating the issuer to use.
    cert-manager.io/cluster-issuer: example-issuer
  name: quizzer-api
  namespace: dev
```

Showcase