Algorithmic trading w Python

Plan

- **Wprowadzenie**
- Opis innych projektów
- DCA
  - Opis strategii
  - Architektura
  - Implementacja
  - Uruchomienie
  - Terraform
- Podsumowanie

Jak się nazywał
ten koleś co umie
programować...

Siema Oskar jest
sprawa mam
pomysł tylko
musisz mi pomóc
z pythonem

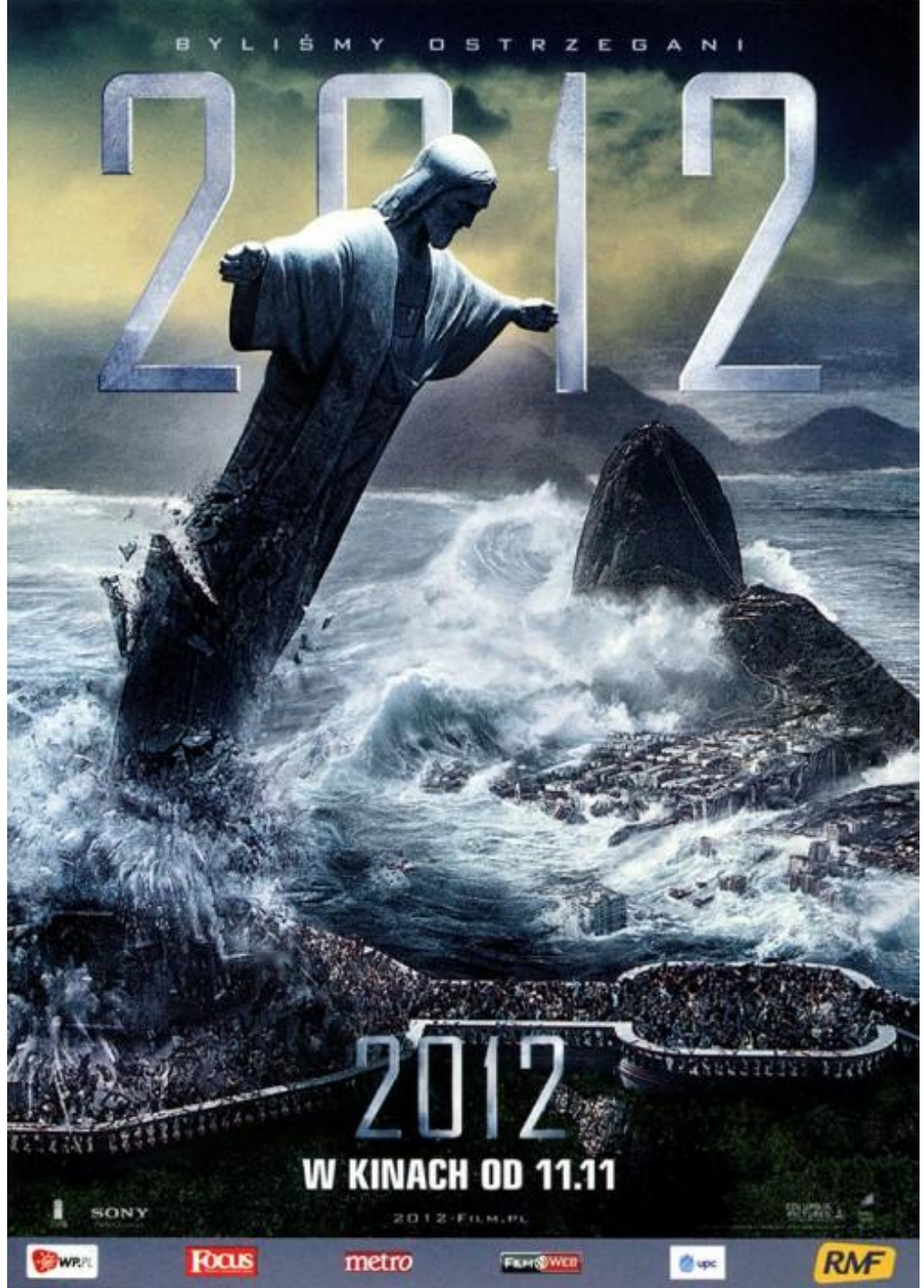
No jasne
co tam
potrzeba

Wchodzimy na rynek krypto

pamiętasz jak mówiłem o bitkoinie w liceum?

Teraz można na tym nieźle zarobić

Trzeba zautomatyzować
arbitraż


BYLIŚMY OSTRZEGANI
2012
2012
W KINACH OD 11.11
SONY
2012-FILM.PL
WP.PL
Focus
metro
FILMSWEB
upc
RMF

10 $


ZAPRASZAMY NA

Nie no stary muszę zakuwać do sesji

2015


TensorFlow

Zainteresowałem się ML

- NLP
- Sentiment Analysis
- Common Crawl

- Korelacja między sentymentem na temat krypto a jego ceną?


Za mało czasu
Za mało umiejętności
Za mało funduszy

2019


FINTECH
34%
12%

10000 $

Miałeś rację

Jest jeszcze opcja żeby napisać tego bota?

Robimy

Plan

- Wprowadzenie
- **Opis innych projektów**
- DCA
  - Opis strategii
  - Architektura
  - Implementacja
  - Uruchomienie
  - Terraform
- Podsumowanie

Powstało kilka projektów

- **Proof-of-concept**
  - Działał
  - Zarabiał
  - Stworzony w jedno popołudnie
  - Nieefektywny
  - Cała alfa była oparta o jeden dziwny serwis
- **Hiroshima**
  - Wszystko działało na Hetznerze na jednej wirtualce
  - Grafana
  - Dojrzały kod, nie rzucał błędów
  - Przestał zarabiać, gdy straciliśmy dziwny serwis
- **Nagasaki**
  - Większy zespół
  - Chcieliśmy odtworzyć zarobki z proof-of-concept
  - Stworzyliśmy fajny framework
  - Ergonomiczny interfejs do podłączania strategii

Dziwny serwis

- Kwotowanie USD/PLN to nietrywialny problem.
  - Mamy dobre serwisy które są drogie
  - Mamy kiepskie serwisy które są darmowe
- Zaciągaliśmy kurs USD/PLN z giełdy krypto
  - Był on podawany jako wynik crossowania BTC/USD oraz BTC/PLN
  - Potrafił się zmieniać o 10% w pół godziny
  - W jakiś sposób to działało

Strategia

- Wcześniejsze projekty miały jeden istotny problem: były **ambitne**.
- Postanowiłem postawić na prostotę

Plan

- Wprowadzenie
- Opis innych projektów
- **DCA**
  - **Opis strategii**
  - Architektura
  - Implementacja
  - Uruchomienie
  - Terraform
- Podsumowanie

Bitcoin World

Dollar Cost Averaging

Dollar cost averaging

- Kupuj aktywa o zadanej wartości w stałych interwałach czasowych

dca Public

Pin Unwatch 1 Fork 0 Star 1

main 1 Branch 0 Tags Go to file t Add file Code

dependabot[bot] and oskarissimus Bump pydantic-settings from 2.4.0 to 2.5.2 in /src ec56ee0 · 3 days ago 77 Commits

.github	Create dependabot.yml	last month
src	Bump pydantic-settings from 2.4.0 to 2.5.2 in /src	3 days ago
terraform	lets goo	3 weeks ago
.gitignore	ignore tf lock info file	2 years ago
LICENSE	Initial commit	2 years ago
README.md	test that auto deploy	3 weeks ago

README GPL-3.0 license

DCA

Dollar-cost averaging for xtb and zonda. Made with Python and Terraform on GCP.

Run on Google Cloud

Cost

I chose Google Cloud Platform (GCP) because, contrary to what you might think, it is actually more cost-effective. The paid resources include:

- **Pub/Sub**: Inexpensive due to the small number of messages.

About

No description, website, or topics provided.

Readme
GPL-3.0 license
Activity
1 star
1 watching
0 forks

Releases

No releases published
Create a new release

Packages

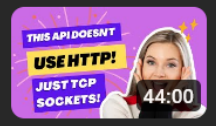
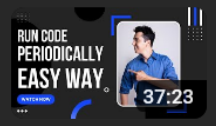
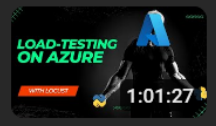
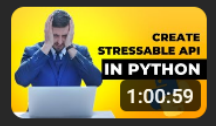
No packages published
Publish your first package

Contributors 3

oskarissimus
dependabot[bot]
kwikiel Kacper Wikieł

Languages

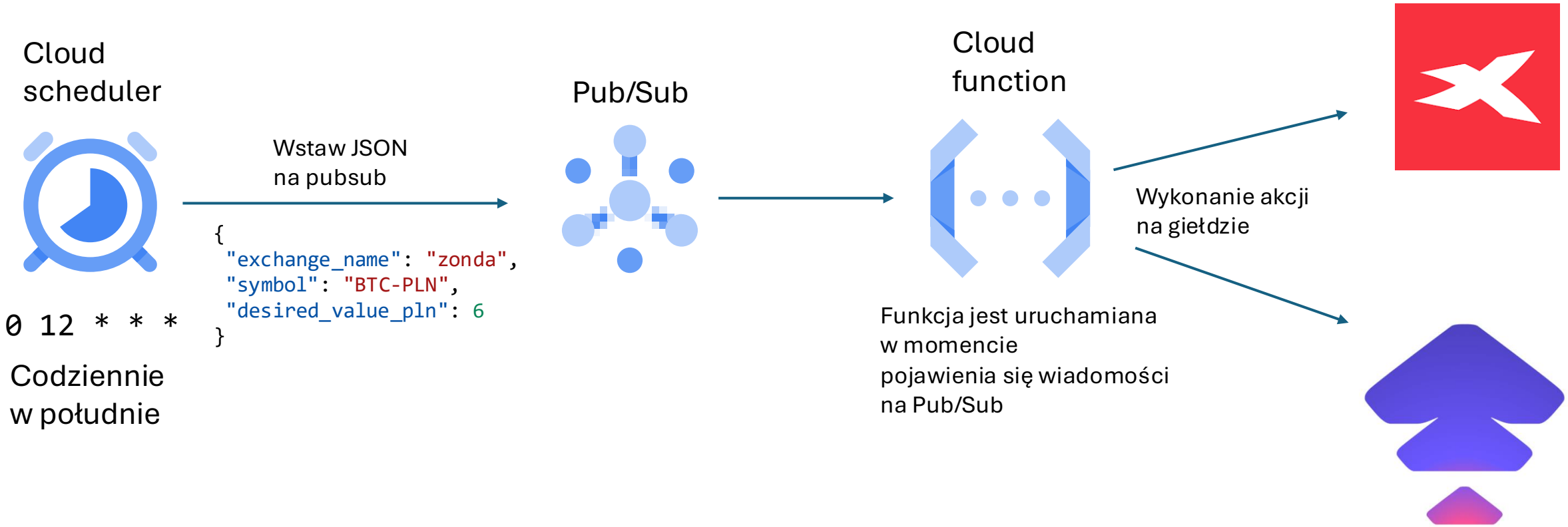
Python 60.8% HCL 30.2%
Jupyter Notebook 8.7% Makefile 0.3%

Live stream	Type	Visibility	Restrictions	Date	Views ↓	Live viewers	Comments	Likes (vs. dislikes)
Live replay								
 DCA for S&P 500 ETF on XTB with Python live stream of creating python script to periodically buy ETF units using XTB platform	Streaming software	Public	None	Sep 11, 2022 Streamed	1,589	–	2	76.5% 26 likes
 DCA for S&P 500 ETF on XTB with Python par... Live stream of creating python script to periodically buy ETF nits using XTB platform. It...	Streaming software	Public	None	Sep 12, 2022 Streamed	669	–	3	100.0% 9 likes
 DCA for S&P 500 ETF on XTB with Python par... Live stream of creating python script to periodically buy ETF nits using XTB platform....	Streaming software	Public	None	Sep 13, 2022 Streamed	322	–	2	87.5% 7 likes
 Load-testing with Python and Locust on Azure Deploy stressable app and test it on Azure	Streaming software	Public	None	Jan 17, 2023 Streamed	145	–	0	100.0% 3 likes
 Creating stressable API with FastAPI in Python In this stream I will show how to create API that will be later used as target for service load testin...	Streaming software	Public	None	Jan 3, 2023 Streamed	75	–	0	100.0% 3 likes

Plan

- Wprowadzenie
- Opis innych projektów
- **DCA**
  - Opis strategii
  - **Architektura**
  - Implementacja
  - Uruchomienie
  - Terraform
- Podsumowanie

Architektura aplikacji



Architektura aplikacji

- Trochę skomplikowane jak na use case?
  - Nie da się odpalać funkcji bezpośrednio przez cron, trzeba używać pubsub
  - Gdybym wiedział że e2-micro jest w free tier to bym tego nie tworzył :/
  - Można by mieć całe rozwiązanie w python bez Terraform itp.

- Dobre ćwiczenie na terraform a także wygodne że jest GCP UI do wszystkich usług i można zawsze podejrzeć co się dzieje

Cloud Scheduler

Jobs

CREATE JOB | REFRESH | FORCE RUN | EDIT | COPY | PAUSE | RESUME | DELETE

SCHEDULER JOBS | APP ENGINE CRON JOBS

Filter | Filter jobs

Name	Status of last execution	Region	State	Description	Frequency	Target	Last run	Next run	Last updated	Actions
buy_btc_twice_a_day	Success	europe-central2	Enabled		0 10,22 * * * (Europe/Warsaw)	Topic : projects/dca-2136/topics/buy_market	Aug 27, 2024, 10:00:00 AM	Aug 27, 2024, 10:00:00 PM	Aug 25, 2024, 9:21:01 PM	

Cloud Run functions

Functions

CREATE FUNCTION REFRESH LEARN RELEASE NOTES

We will be integrating Cloud Functions into Cloud Run UI in the upcoming months. GO TO CLOUD RUN

Filter Filter functions

Environment	Name	Last deployed	Region	Recommendation	Trigger	Runtime	Memory allocated	Executed function	Ac
Cloud Run function (1st gen)	buy_market	Aug 27, 2024, 7:41:42 PM	europe-central2		Topic: buy_market	Python 3.12	256 MB	buy_market	

zondacrypto

- Dashboard

MARKET
- Giełda
- Moje oferty

KONTO
- Portfele
- Historia

ZND
- Earn
- ZND Token

NARZĘDZIA
- Kantor FIAT
- Tryb PRO

INNE
- Afiliacje

Operacje | Transakcje | Wpłaty/Wypłaty | Szczegółowa

RYNKI: Wybrano walut: 2 | CZAS: Ostatni tydzień | TYP: Transakcje

DATA OPERACJI	RODZAJ	WARTOŚĆ	SALDO CAŁKOWITE PO OPERACJI
27.08.2024, 20:18:21	Pobranie środków z transakcji z rachunku: PLN	−5.99 PLN	980.74 PLN
27.08.2024, 20:18:21	Otrzymanie środków z transakcji na rachunek: BTC	0.00002514 BTC	0.00009235 BTC
27.08.2024, 20:16:34	Pobranie środków z transakcji z rachunku: PLN	−6.00 PLN	986.73 PLN
27.08.2024, 20:16:34	Otrzymanie środków z transakcji na rachunek: BTC	0.00002517 BTC	0.00006727 BTC
27.08.2024, 18:57:29	Pobranie środków z transakcji z rachunku: PLN	−10.00 PLN	992.73 PLN
27.08.2024, 18:57:29	Otrzymanie środków z transakcji na rachunek: BTC	0.00004219 BTC	0.00004219 BTC

Plan

- Wprowadzenie
- Opis innych projektów
- **DCA**
  - Opis strategii
  - Architektura
  - **Implementacja**
  - Uruchomienie
  - Terraform
- Podsumowanie

```python
import base64

import functions_framework

from dca.clients.exchange_clients import build_exchange_client_by_name
from dca.models.message import Message
from dca.settings import Settings


@functions_framework.cloud_event
def buy_market(cloud_event):
    raw_message = base64.b64decode(cloud_event.data["message"]["data"]).decode()
    message = Message.model_validate_json(raw_message)
    settings = Settings()
    client = build_exchange_client_by_name(message.exchange_name, settings=settings)
    symbol = client.parse_symbol(message.symbol)
    client.buy_market(symbol, message.desired_value_pln)
```

```python
import base64

import functions_framework

from dca.clients.exchange_clients import build_exchange_client_by_name
from dca.models.message import Message
from dca.settings import Settings


@functions_framework.cloud_event
def buy_market(cloud_event):
    raw_message = base64.b64decode(cloud_event.data["message"]["data"]).decode()
    message = Message.model_validate_json(raw_message)
    settings = Settings()
    client = build_exchange_client_by_name(message.exchange_name, settings=settings)
    symbol = client.parse_symbol(message.symbol)
    client.buy_market(symbol, message.desired_value_pln)
```

```python
def build_xtb_client(settings: Settings) -> XTBClientWrapper:
    return XTBClientWrapper(
        user_id=settings.xtb_user_id, password=settings.xtb_password, port=settings.xtb_api_port
    )


def build_zonda_client(settings: Settings) -> ZondaClient:
    return ZondaClient(api_key=settings.zonda_api_key, api_secret=settings.zonda_api_secret)


exchange_name_to_builder_mapping = {"xtb": build_xtb_client, "zonda": build_zonda_client}


def build_exchange_client_by_name(
    exchange_name: str, settings: Settings
) -> AbstractExchangeClient:
    if exchange_name not in exchange_name_to_builder_mapping:
        raise ExchangeClientDoesNotExistError(
            f"Exchange client for {exchange_name} does not exist"
        )
    return exchange_name_to_builder_mapping[exchange_name](settings)
```

```python
def build_xtb_client(settings: Settings) -> XTBClientWrapper:
    return XTBClientWrapper(
        user_id=settings.xtb_user_id, password=settings.xtb_password, port=settings.xtb_api_port
    )


def build_zonda_client(settings: Settings) -> ZondaClient:
    return ZondaClient(api_key=settings.zonda_api_key, api_secret=settings.zonda_api_secret)


exchange_name_to_builder_mapping = {"xtb": build_xtb_client, "zonda": build_zonda_client}


def build_exchange_client_by_name(
    exchange_name: str, settings: Settings
) -> AbstractExchangeClient:
    if exchange_name not in exchange_name_to_builder_mapping:
        raise ExchangeClientDoesNotExistError(
            f"Exchange client for {exchange_name} does not exist"
        )
    return exchange_name_to_builder_mapping[exchange_name](settings)
```

Jestem dumny z mojej małej fabryki

Ciekawostki o komunikacjach z giełdami

- Mogło by się wydawać że w dzisiejszych czasach dostaniemy dobrze udokumentowane API RESTowe
- Nic bardziej mylnego

XTB

- API nie używa HTTP, zamiast tego jest własny protokół po TCP
- Może to normalne? Każdy kto trochę integrował wie że pod nazwą API potrafi się kryć wiele różności
- Całe szczęście jest dostępny oficjalny wrapper do pythona, więc jest trochę łatwiej. Aczkolwiek nie jest to pythoniczny kod, więc go ulepszyłem

XTB API

```
class JsonSocket(object):
    def __init__(self, address, port, encrypt = False):
        self._ssl = encrypt
        if self._ssl != True:
            self.socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
        else:
            sock = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
            self.socket = ssl.wrap_socket(sock)
        self.conn = self.socket
        self._timeout = None
        self._address = address
        self._port = port
        self._decoder = json.JSONDecoder()
        self._receivedData = ''
```

XTB API

```
def connect(self):
    for i in range(API_MAX_CONN_TRIES):
        try:
            self.socket.connect( (self.address, self.port) )
        except socket.error as msg:
            logger.error("SockThread Error: %s" % msg)
            time.sleep(0.25);
            continue
        logger.info("Socket connected")
        return True
    return False
```

XTB API

```
def _sendObj(self, obj):
    msg = json.dumps(obj)
    self._waitingSend(msg)

def _waitingSend(self, msg):
    if self.socket:
        sent = 0
        msg = msg.encode('utf-8')
        while sent < len(msg):
            sent += self.conn.send(msg[sent:])
            logger.info('Sent: ' + str(msg))
            time.sleep(API_SEND_TIMEOUT/1000)
```

XTB API

```
def _read(self, bytesSize=4096):
    if not self.socket:
        raise RuntimeError("socket connection broken")
    while True:
        char = self.conn.recv(bytesSize).decode()
        self._receivedData += char
        try:
            (resp, size) = self._decoder.raw_decode(self._receivedData)
            if size == len(self._receivedData):
                self._receivedData = ''
                break
            elif size < len(self._receivedData):
                self._receivedData = self._receivedData[size:].strip()
                break
        except ValueError as e:
            continue
    logger.info('Received: ' + str(resp))
    return resp
```

Zonda

- Tutaj wielkich zaskoczeń nie ma, jedynie podpisywanie requestów, co może też jest standardem?
- Być może ma to sens ponieważ nigdy się nie wysyła private key

```python
def _make_request(self, method: str, url: str, payload: str = "") -> requests.Response:
    current_timestamp = int(dt.datetime.now().timestamp())
    api_hash = self._make_signature(current_timestamp, payload)
    headers = {
        "Content-Type": "application/json",
        "API-Hash": api_hash,
        "API-Key": self._api_key,
        "Request-Timestamp": str(current_timestamp),
        "operation-id": str(uuid.uuid4()),
    }
    response = requests.request(method, url, data=payload, headers=headers, timeout=10)
    return response

def _make_signature(self, timestamp: int, payload: str) -> str:
    key = self._api_secret.encode("utf-8")
    msg = f"{self._api_key}{timestamp}{payload}".encode("utf-8")
    return hmac.new(key, msg, "SHA512").hexdigest()
```

Plan

- Wprowadzenie
- Opis innych projektów
- **DCA**
  - Opis strategii
  - Architektura
  - Implementacja
  - **Uruchomienie**
  - Terraform
- Podsumowanie

Instalacja

- terraform init
- terraform apply

- To nigdy nie jest takie proste

Konfiguracja `terraform.tfvars`

```
xtb_user_id       = 1
xtb_password      = "a"
xtb_api_port      = 5124
zonda_api_key     = "123"
zonda_api_secret = "123"
schedules_paused = false

schedules = {
  buy_btc_twice_a_day = {
    exchange_name     = "zonda"
    symbol            = "BTC-PLN"
    desired_value_pln = 6
    schedule          = "0 10,22 * * *"
  }
}
```

Plan

- Wprowadzenie
- Opis innych projektów
- **DCA**
  - Opis strategii
  - Architektura
  - Implementacja
  - Uruchomienie
  - **Terraform**
- Podsumowanie

Terraform – tworzone zasoby

- Cloud function
- Cloud schedule
- Pub/sub
- Bucket and objects (zip with function)
- Secrets and secret versions
- Service accounts (!)

Terraform – rzeczy łatwe i trudne

- Jest część zasobów które można stworzyć po pojedynczym przeczytaniu dokumentacji
- Inne trzeba debugować pół dnia bo nie wiadomo co się robi źle (na ogół IAM)

Rzeczy łatwe

- Cloud schedule
- Pub/sub
- Bucket and objects (zip with function)
- Secrets and secret versions

Rzeczy trudne

- Cloud function
- Service accounts (!)

Rzeczy denerwujące

- Włączanie serwisów w projekcie
- Jest możliwość aktywowania serwisów przez terraform, ale dwa z nich są kluczowe do działania samego terraform **`serviceusage`** i **`cloudresourcemanager`**
- Serwisy są *ostatecznie* spójne (eventually consistent)
- Nie ma możliwości sprawdzenia czy dany serwis jest już aktywowany czy nie – inaczej – czy aktywacja została rozpropagowana
- Komunikaty błędów terraform często nie są pomocne

Aktywacja serwisów

```
module "project-services" {
  source  = "terraform-google-modules/project-factory/google//modules/project_services"
  version = "~> 15.0"

  project_id                  = data.google_project.default.project_id
  disable_services_on_destroy = false

  activate_apis = [
    "secretmanager.googleapis.com",
    "cloudfunctions.googleapis.com",
    "cloudbuild.googleapis.com",
    "iam.googleapis.com",
    "containerregistry.googleapis.com",
    "cloudscheduler.googleapis.com",
    "pubsub.googleapis.com",
    "logging.googleapis.com",
    "cloudapis.googleapis.com",
    "compute.googleapis.com",
  ]
}
```

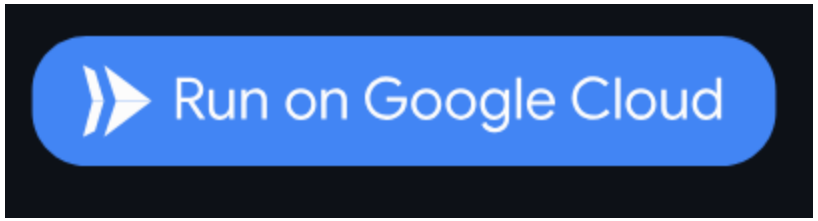
```
gcloud services enable serviceusage.googleapis.com cloudresourcemanager.googleapis.com && sleep 30
terraform init
terraform apply -target=module.project-services -auto-approve && sleep 60
terraform apply -auto-approve
```

Dlaczego cloud function w terraform jest trudne?

- Nie wiem, wydaje się że powinno być łatwe
- Nie lubię kopiować kodu bez zrozumienia
- Terraform jest verbose, generalnie to dobrze, ale w przypadku secretów wydaje się to nadmiarowo skomplikowane, chyba że robię to źle
- Osobne konta serwisowe builder i executor – ok, ale dokumentacja wprowadza w błąd jeśli chodzi o domyślne wartości

```
data "archive_file" "source" {
  type        = "zip"
  source_dir  = "../src"
  output_path = "/tmp/function.zip"
}

resource "google_storage_bucket_object" "zip" {
  source       = data.archive_file.source.output_path
  content_type = "application/zip"

  name   = "src-${data.archive_file.source.output_md5}.zip"
  bucket = google_storage_bucket.function_bucket.name
}

resource "google_cloudfunctions_function" "function" {
  name                  = "buy_market"
  runtime               = "python312"
  entry_point           = "buy_market"
  source_archive_bucket = google_storage_bucket.function_bucket.name
  source_archive_object = google_storage_bucket_object.zip.name

  event_trigger {
    event_type = "providers/cloud.pubsub/eventTypes/topic.publish"
    resource   = google_pubsub_topic.buy_market.id
  }

  timeout = 120
  timeouts {
    create = "2m"
    update = "2m"
    delete = "2m"
  }

  service_account_email = google_service_account.function_runner.email
  build_service_account = google_service_account.function_builder.id

  secret_environment_variables {
    key     = "XTB_USER_ID"
    secret  = google_secret_manager_secret.xtb_user_id.secret_id
    version = "latest"
  }

  secret_environment_variables {
    key     = "XTB_PASSWORD"
    secret  = google_secret_manager_secret.xtb_password.secret_id
    version = "latest"
  }

  secret_environment_variables {
    key     = "ZONDA_API_KEY"
    secret  = google_secret_manager_secret.zonda_api_key.secret_id
    version = "latest"
  }

  secret_environment_variables {
    key     = "ZONDA_API_SECRET"
    secret  = google_secret_manager_secret.zonda_api_secret.secret_id
    version = "latest"
  }

  environment_variables = {
    XTB_API_PORT = var.xtb_api_port
  }
  depends_on = [
    google_secret_manager_secret_version.xtb_user_id,
    google_secret_manager_secret_version.xtb_password,
    google_secret_manager_secret_version.zonda_api_key,
    google_secret_manager_secret_version.zonda_api_secret,
    google_secret_manager_secret_iam_binding.xtb_user_id,
    google_secret_manager_secret_iam_binding.xtb_password,
    google_secret_manager_secret_iam_binding.zonda_api_key,
    google_secret_manager_secret_iam_binding.zonda_api_secret,
    google_project_iam_member.builder_artifact_registry_writer,
    google_project_iam_member.builder_storage_object_admin,
    google_project_iam_member.builder_logging_writer,
    google_storage_bucket_iam_member.builder
  ]
}
```

BONUS




```
cloudshell_open --repo_url "https://github.com/oskarissimus/dca.git"
--dir "terraform" --page "shell" --force_new_clone
```

```
{
    "name": "foo-app",
    "env": {
        "TF_VAR_xtb_user_id": {
            "required": false,
            "value": "1234"
        },
        "TF_VAR_xtb_password": {
            "required": false,
            "value": "password"
        },
        "TF_VAR_zonda_api_key": {
            "required": true,
            "value": "1234"
        },
        "TF_VAR_zonda_api_secret": {
            "required": true,
            "value": "password"
        }
    },
    "build": {
        "skip": true
    },
    "hooks": {
        "prebuild": {
            "commands": [
                "gcloud config set project $GOOGLE_CLOUD_PROJECT",
                "gcloud services enable serviceusage.googleapis.com cloudres
                "terraform init",
                "terraform apply -target=module.project-services -auto-appro
                "terraform apply -auto-approve"
            ]
        }
    }
}
```

Plan

- Wprowadzenie
- Opis innych projektów
- DCA
  - Opis strategii
  - Architektura
  - Implementacja
  - Uruchomienie
  - Terraform
- **Podsumowanie**

Co dalej?

- Wizualizacja profitu
- Podłączenie innych giełd

Na koniec

- Wszyscy znamy kogoś kto się dorobił na krypto, a może niektórzy z tej sali nadal hodlują?
- Może i zrobiłem ten projekt o wiele za późno i trzeba było słuchać Kacpra w 2012?
- Chciałem wam pokazać moją drogę – uczyłem się tradingu i uczyłem się programować
- Teraz mam fajny projekt na github i prezentację którą mogę się podzielić ze społecznością
- Dokończ swój projekt i się pochwal ☺

Dziękuję za uwagę

korczak.xyz

www.linkedin.com/in/oskar-korczak

github.com/oskarissimus