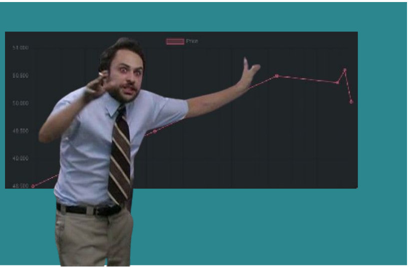
leCAG

Crypto in the Cloud

Agenda

1. Team
2. Project Idea
3. Motivation
4. Use-Cases & Features
5. Costs
6. Architectural Design
7. Pipeline Setup
8. Implementation Details & Examples
9. Live-Demo
10. Outlook

The Team

		
Ruf, Lukas	Tahiri, Arlind	Gigl, Sophie
● DevOps, CI Pipeline ● Cryptocurrency API Wrapper ● Infrastructure Setup	● Backend development ● Frontend development ● HealthCheck	● Heartbeat ● Notifications/Alerts

Project Idea

Basic Idea

Creating a Demo Crypto Trading and Tracking Platform

Cloud Aspect

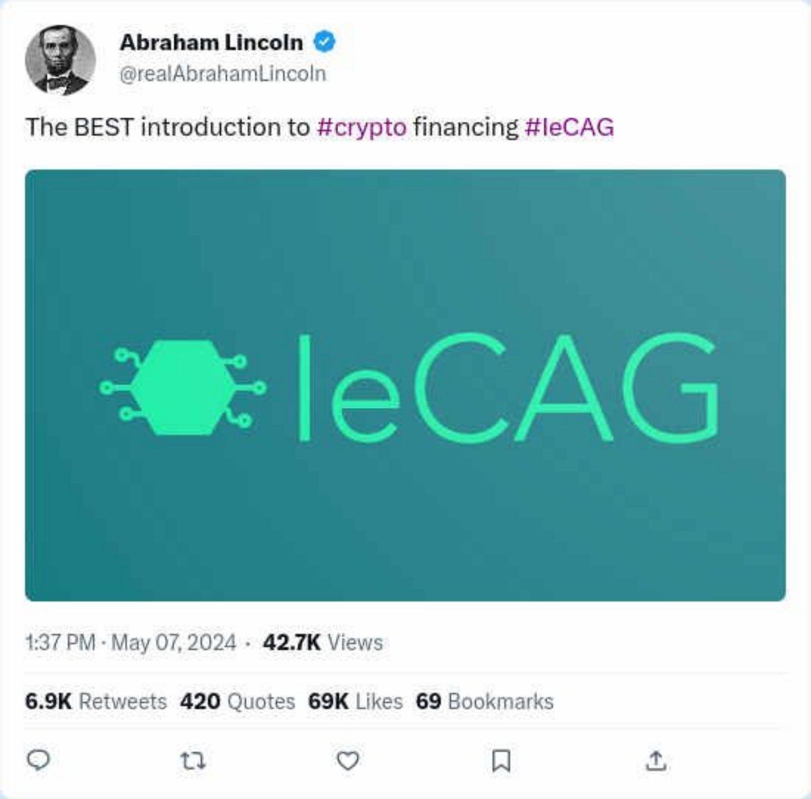
Storing Data like prices of cryptocurrencies, user logins, trades of users, … on Azure Data Tables

Hosting an Web-App on Azure

Running an Email NotificationService with a SaaS

Healthcheck over HTTP Trigger Function

Motivation

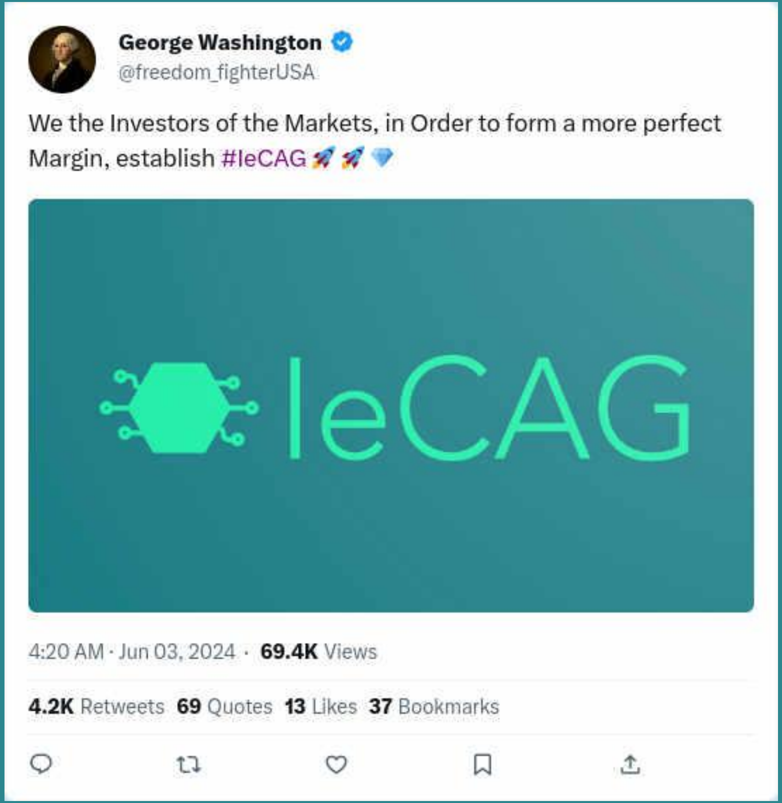


Abraham Lincoln
@realAbrahamLincoln

The BEST introduction to #crypto financing #leCAG

1:37 PM · May 07, 2024 · **42.7K** Views

6.9K Retweets **420** Quotes **69K** Likes **69** Bookmarks



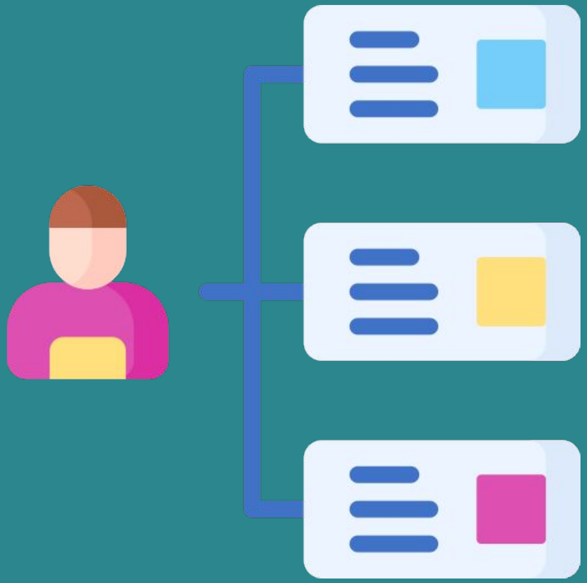
George Washington
@freedom_fighterUSA

We the Investors of the Markets, in Order to form a more perfect Margin, establish #leCAG 🚀 🚀 💎

4:20 AM · Jun 03, 2024 · **69.4K** Views

4.2K Retweets **69** Quotes **13** Likes **37** Bookmarks

Use-Cases
&
Features



User Creation

Collecting and managing user information to create demo accounts on the Platform.

User Crypto Wallets

Users can create crypto wallets for their account.

Trading

Users can buy or sell crypto with fiat currency.

Monitoring

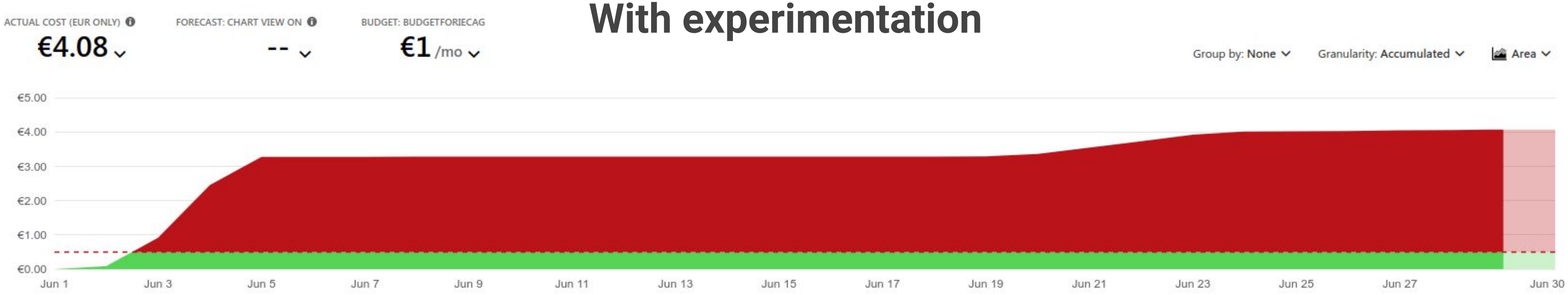
Users can see the current and historical prices of cryptocurrencies.

Notifications

Users can set Alarms or get push Notifications, when crypto prices rise or fall.

Costs

With experimentation



ACTUAL COST (EUR ONLY) €4.08

FORECAST: CHART VIEW ON --

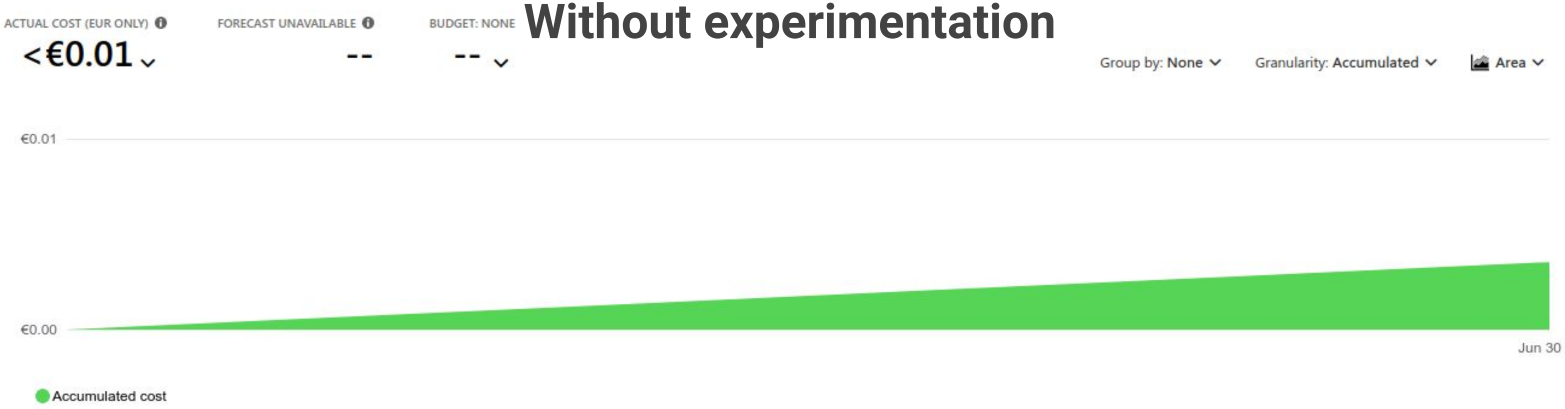
BUDGET: BUDGETFORIECAG €1 /mo

Group by: None

Granularity: Accumulated

Area

Without experimentation



ACTUAL COST (EUR ONLY) <€0.01

FORECAST UNAVAILABLE --

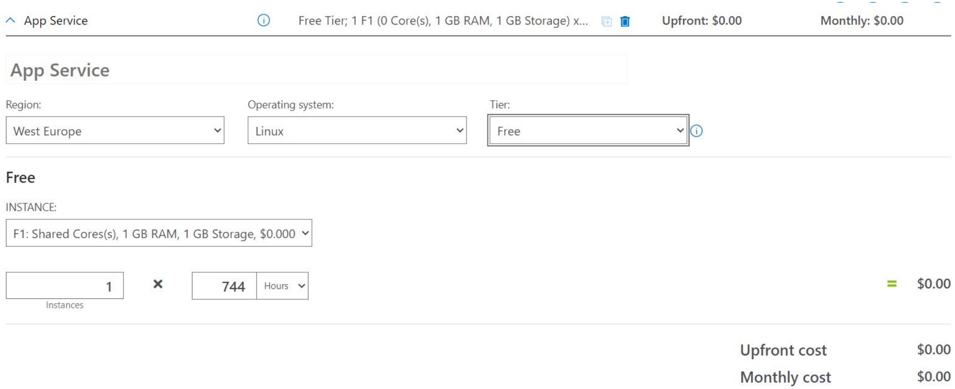
BUDGET: NONE --

Group by: None

Granularity: Accumulated

Area

Costs



App Service | Free Tier; 1 F1 (0 Core(s), 1 GB RAM, 1 GB Storage) x... | Upfront: $0.00 | Monthly: $0.00

App Service

Region: West Europe

Operating system: Linux

Tier: Free

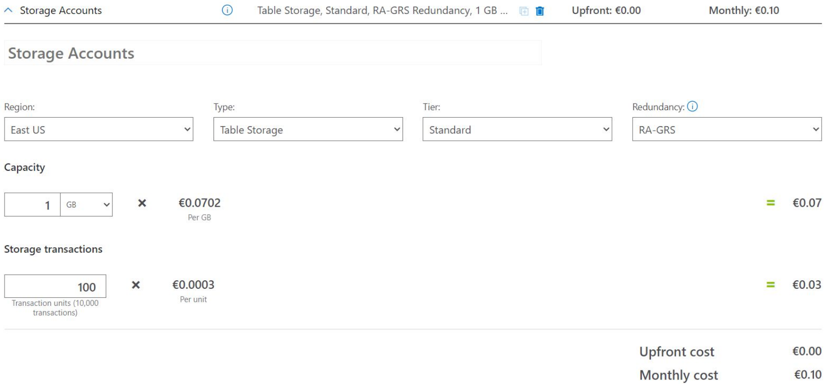
Free

INSTANCE: F1: Shared Cores(s), 1 GB RAM, 1 GB Storage, $0.000

1 Instances × 744 Hours = $0.00

Upfront cost $0.00

Monthly cost $0.00



Storage Accounts | Table Storage, Standard, RA-GRS Redundancy, 1 GB ... | Upfront: €0.00 | Monthly: €0.10

Storage Accounts

Region: East US

Type: Table Storage

Tier: Standard

Redundancy: RA-GRS

Capacity

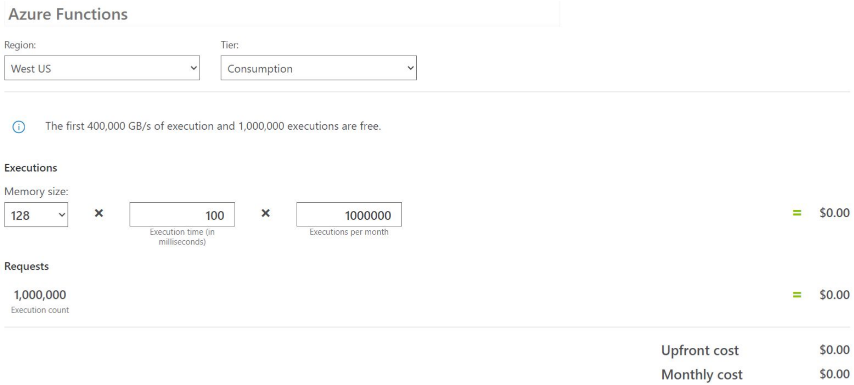
1 GB × €0.0702 Per GB = €0.07

Storage transactions

100 Transaction units (10,000 transactions) × €0.0003 Per unit = €0.03

Upfront cost €0.00

Monthly cost €0.10



Azure Functions

Region: West US

Tier: Consumption

The first 400,000 GB/s of execution and 1,000,000 executions are free.

Executions

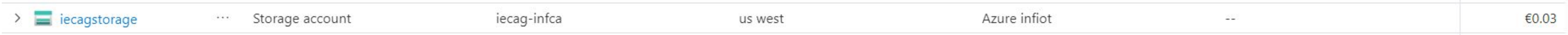
Memory size: 128 × 100 Execution time (in milliseconds) × 1000000 Executions per month = $0.00

Requests

1,000,000 Execution count = $0.00

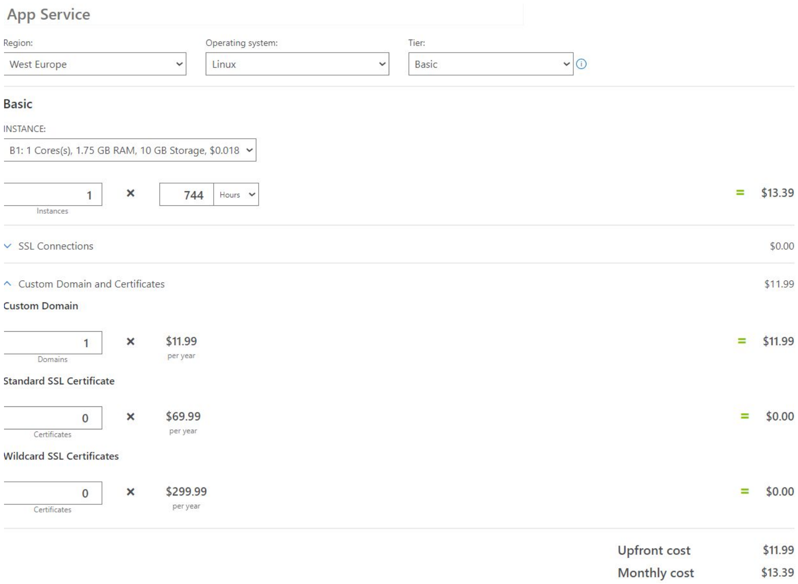
Upfront cost $0.00

Monthly cost $0.00



iecagstorage	Storage account	iecag-infca	us west	Azure infiot	--	€0.03

Scaling



App Service

Region: West Europe

Operating system: Linux

Tier: Basic

Basic

INSTANCE: B1: 1 Cores(s), 1.75 GB RAM, 10 GB Storage, $0.018

1 Instances	×	744 Hours	=	$13.39

SSL Connections $0.00

Custom Domain and Certificates $11.99

Custom Domain

1 Domains	×	$11.99 per year	=	$11.99

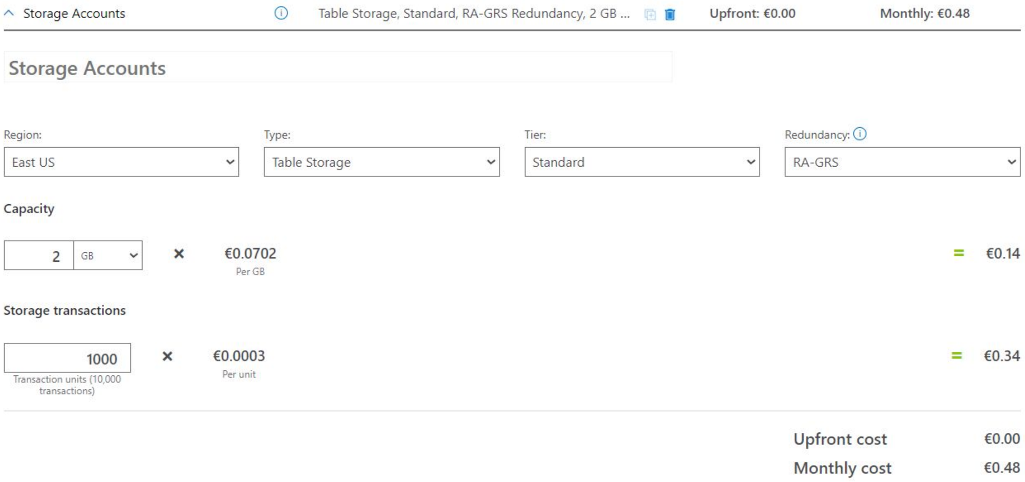
Standard SSL Certificate

0 Certificates	×	$69.99 per year	=	$0.00

Wildcard SSL Certificates

0 Certificates	×	$299.99 per year	=	$0.00

Upfront cost	$11.99
Monthly cost	$13.39



Storage Accounts — Table Storage, Standard, RA-GRS Redundancy, 2 GB ... — Upfront: €0.00 — Monthly: €0.48

Storage Accounts

Region: East US

Type: Table Storage

Tier: Standard

Redundancy: RA-GRS

Capacity

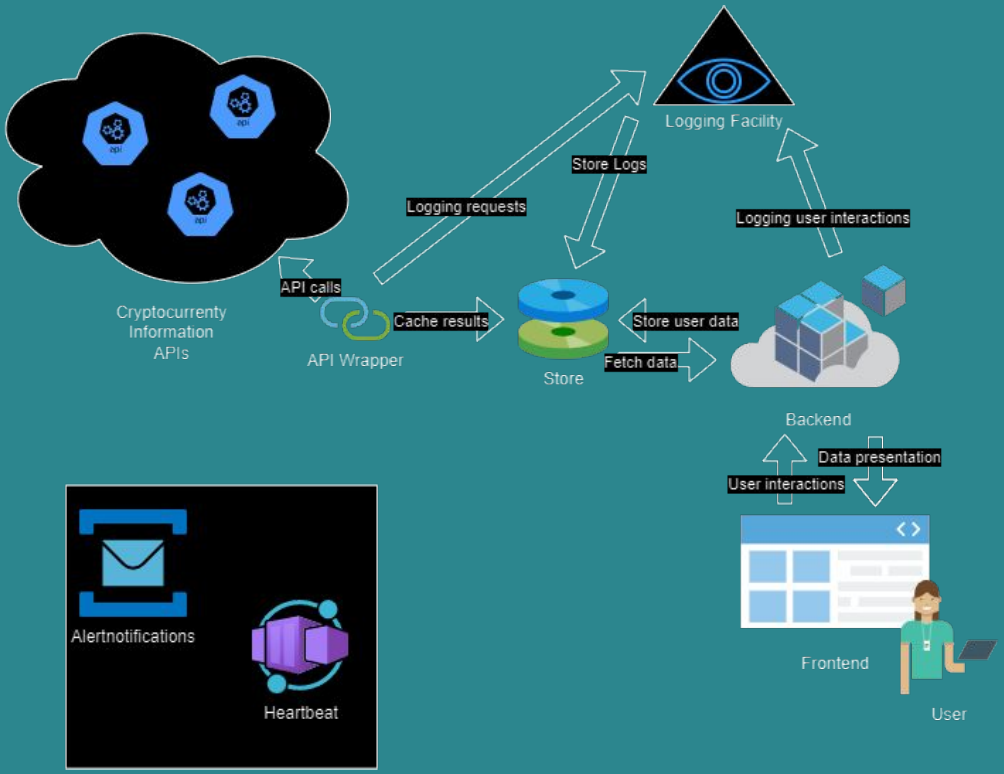
2 GB	×	€0.0702 Per GB	=	€0.14

Storage transactions

1000 Transaction units (10,000 transactions)	×	€0.0003 Per unit	=	€0.34

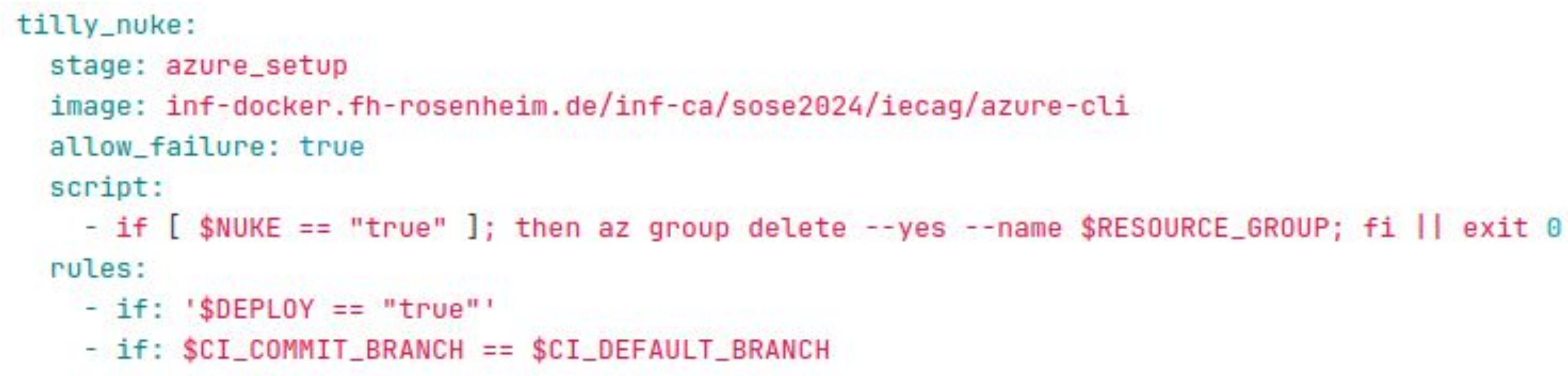
Upfront cost	€0.00
Monthly cost	€0.48

Architectural Design



Pipeline Setup

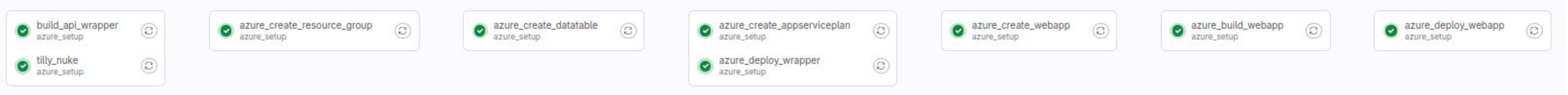
Fire drill



```
tilly_nuke:
  stage: azure_setup
  image: inf-docker.fh-rosenheim.de/inf-ca/sose2024/iecag/azure-cli
  allow_failure: true
  script:
    - if [ $NUKE == "true" ]; then az group delete --yes --name $RESOURCE_GROUP; fi || exit 0
  rules:
    - if: '$DEPLOY == "true"'
    - if: $CI_COMMIT_BRANCH == $CI_DEFAULT_BRANCH
```

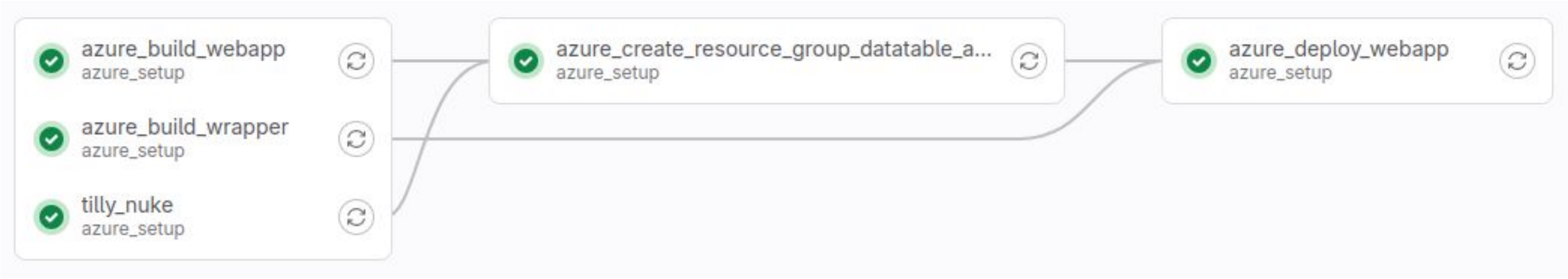
Bigger isn’t always faster

The chain



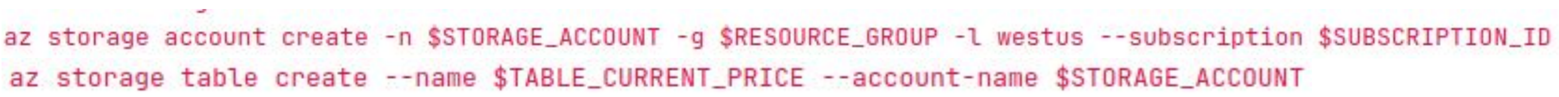
7 - 21 minutes. Max 44 minutes (it also crashed at the end)

The chonk



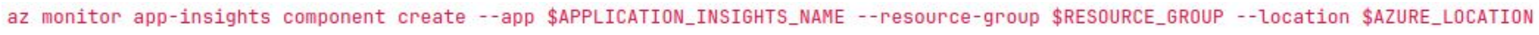
4 - 15 minutes.

Deployment with azure-cli



```
az storage account create -n $STORAGE_ACCOUNT -g $RESOURCE_GROUP -l westus --subscription $SUBSCRIPTION_ID
az storage table create --name $TABLE_CURRENT_PRICE --account-name $STORAGE_ACCOUNT
```

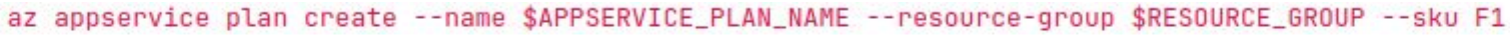
- Data storage



```
az monitor app-insights component create --app $APPLICATION_INSIGHTS_NAME --resource-group $RESOURCE_GROUP --location $AZURE_LOCATION
```

- Query logs
- Overview of service health

App Services plans



```
az appservice plan create --name $APPSERVICE_PLAN_NAME --resource-group $RESOURCE_GROUP --sku F1
```

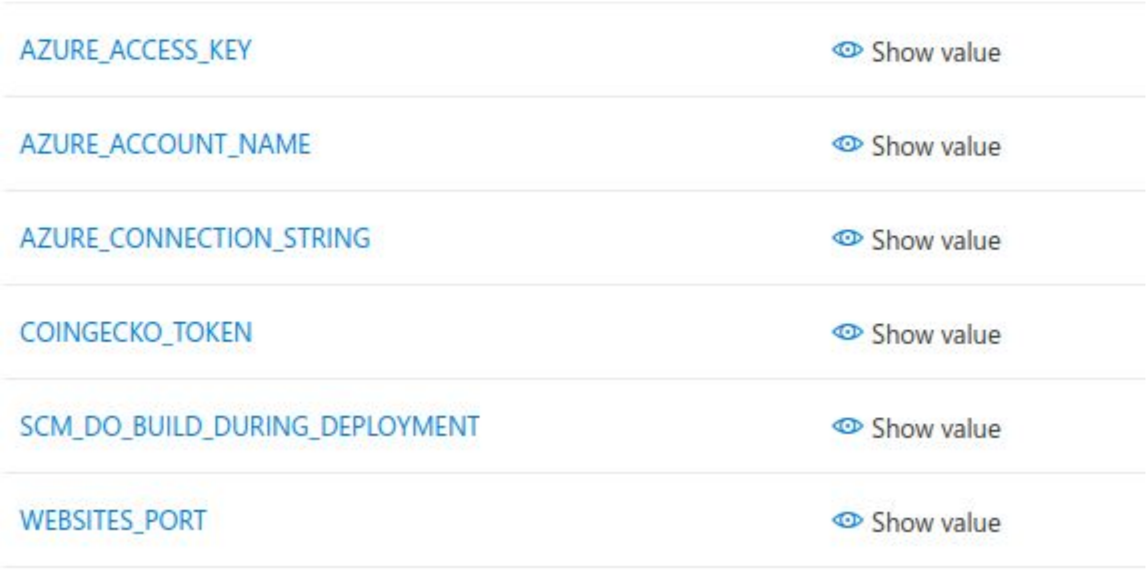
- On the cheapest SKU due to low demand
- Upgradable at any time
- Two separate plans due to platform
  - Python cannot run on Windows, apparently

Why WebApp?

Why WebApp?

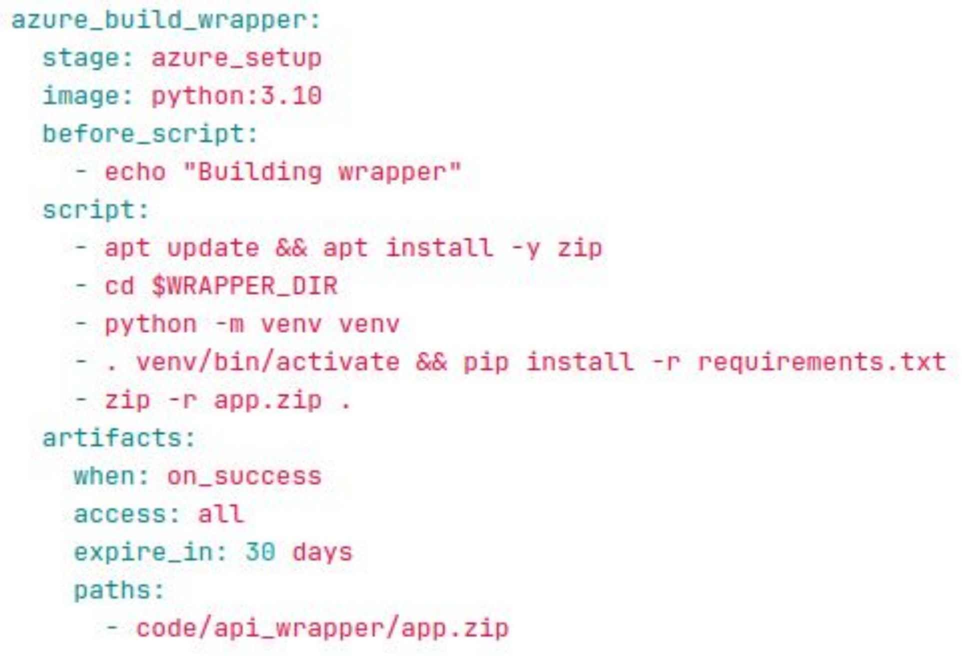
Webapp Deployment

```
az webapp create --resource-group $RESOURCE_GROUP --plan $APPSERVICE_PLAN_NAME --name $WEBAPP_NAME --runtime "dotnet:8"
```

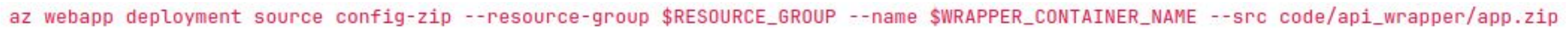


AZURE_ACCESS_KEY	👁 Show value
AZURE_ACCOUNT_NAME	👁 Show value
AZURE_CONNECTION_STRING	👁 Show value
COINGECKO_TOKEN	👁 Show value
SCM_DO_BUILD_DURING_DEPLOYMENT	👁 Show value
WEBSITES_PORT	👁 Show value

WebApp Deployment - Source Zip

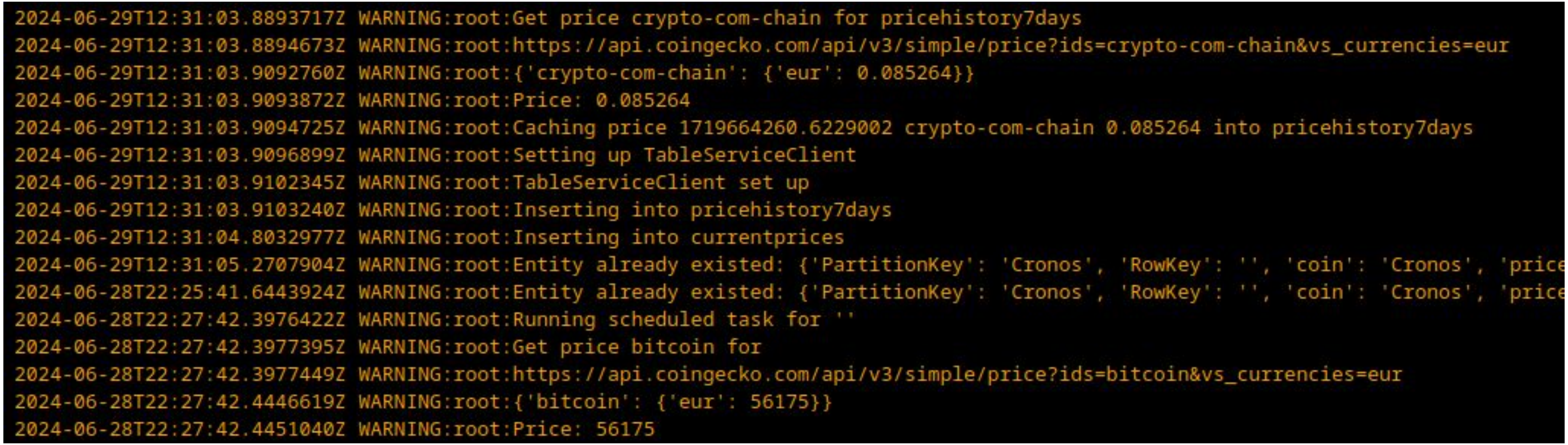


```
azure_build_wrapper:
  stage: azure_setup
  image: python:3.10
  before_script:
    - echo "Building wrapper"
  script:
    - apt update && apt install -y zip
    - cd $WRAPPER_DIR
    - python -m venv venv
    - . venv/bin/activate && pip install -r requirements.txt
    - zip -r app.zip .
  artifacts:
    when: on_success
    access: all
    expire_in: 30 days
    paths:
      - code/api_wrapper/app.zip
```



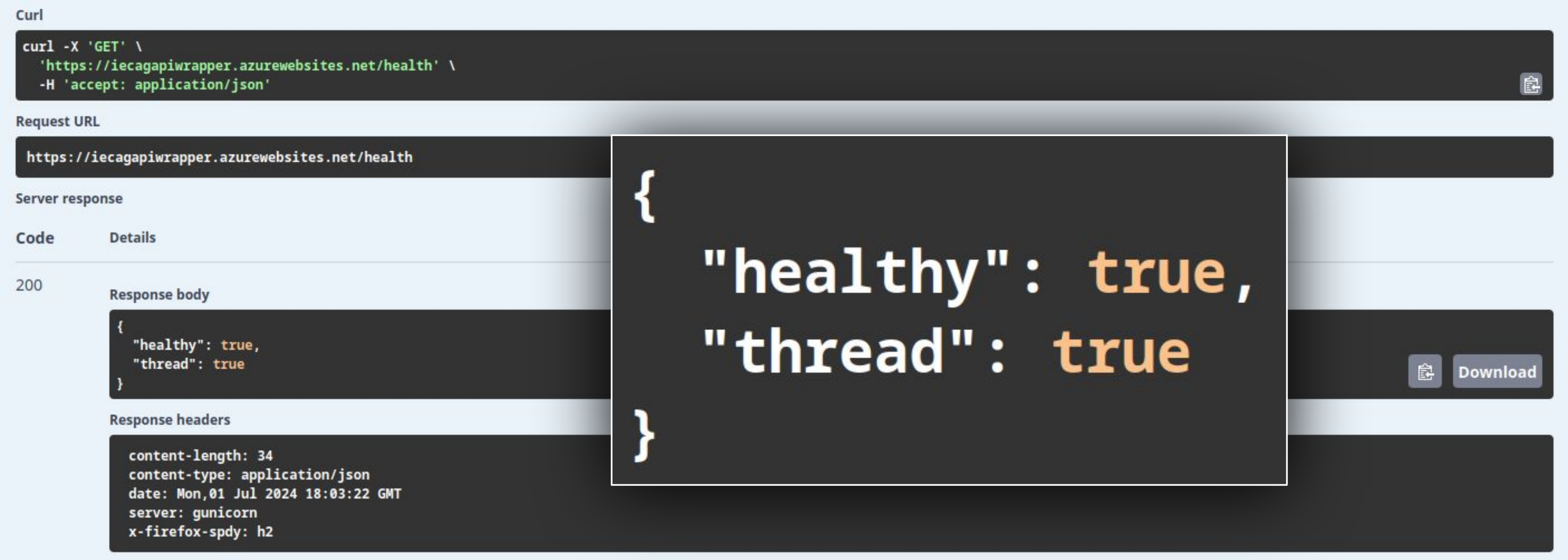
```
az webapp deployment source config-zip --resource-group $RESOURCE_GROUP --name $WRAPPER_CONTAINER_NAME --src code/api_wrapper/app.zip
```

WebApp Logging



```
2024-06-29T12:31:03.8893717Z WARNING:root:Get price crypto-com-chain for pricehistory7days
2024-06-29T12:31:03.8894673Z WARNING:root:https://api.coingecko.com/api/v3/simple/price?ids=crypto-com-chain&vs_currencies=eur
2024-06-29T12:31:03.9092760Z WARNING:root:{'crypto-com-chain': {'eur': 0.085264}}
2024-06-29T12:31:03.9093872Z WARNING:root:Price: 0.085264
2024-06-29T12:31:03.9094725Z WARNING:root:Caching price 1719664260.6229002 crypto-com-chain 0.085264 into pricehistory7days
2024-06-29T12:31:03.9096899Z WARNING:root:Setting up TableServiceClient
2024-06-29T12:31:03.9102345Z WARNING:root:TableServiceClient set up
2024-06-29T12:31:03.9103240Z WARNING:root:Inserting into pricehistory7days
2024-06-29T12:31:04.8032977Z WARNING:root:Inserting into currentprices
2024-06-29T12:31:05.2707904Z WARNING:root:Entity already existed: {'PartitionKey': 'Cronos', 'RowKey': '', 'coin': 'Cronos', 'price
2024-06-28T22:25:41.6443924Z WARNING:root:Entity already existed: {'PartitionKey': 'Cronos', 'RowKey': '', 'coin': 'Cronos', 'price
2024-06-28T22:27:42.3976422Z WARNING:root:Running scheduled task for ''
2024-06-28T22:27:42.3977395Z WARNING:root:Get price bitcoin for
2024-06-28T22:27:42.3977449Z WARNING:root:https://api.coingecko.com/api/v3/simple/price?ids=bitcoin&vs_currencies=eur
2024-06-28T22:27:42.4446619Z WARNING:root:{'bitcoin': {'eur': 56175}}
2024-06-28T22:27:42.4451040Z WARNING:root:Price: 56175
```

Health check/report of API Wrapper



Curl

```
curl -X 'GET' \
  'https://iecagapiwrapper.azurewebsites.net/health' \
  -H 'accept: application/json'
```

Request URL

```
https://iecagapiwrapper.azurewebsites.net/health
```

Server response

Code	Details
200	Response body

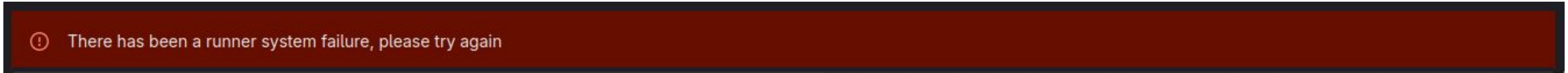
```
{
  "healthy": true,
  "thread": true
}
```

Response headers

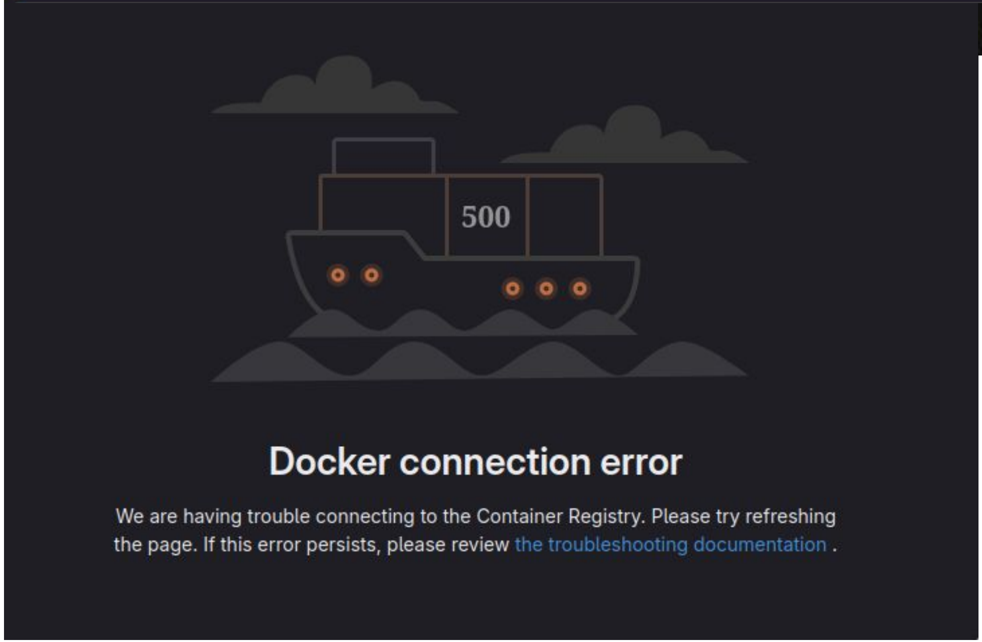
```
 content-length: 34
 content-type: application/json
 date: Mon,01 Jul 2024 18:03:22 GMT
 server: gunicorn
 x-firefox-spdy: h2
```

```
{
  "healthy": true,
  "thread": true
}
```

Challenges and Pitfalls

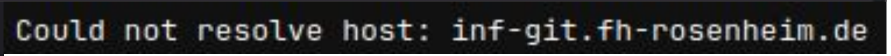


There has been a runner system failure, please try again



Docker connection error

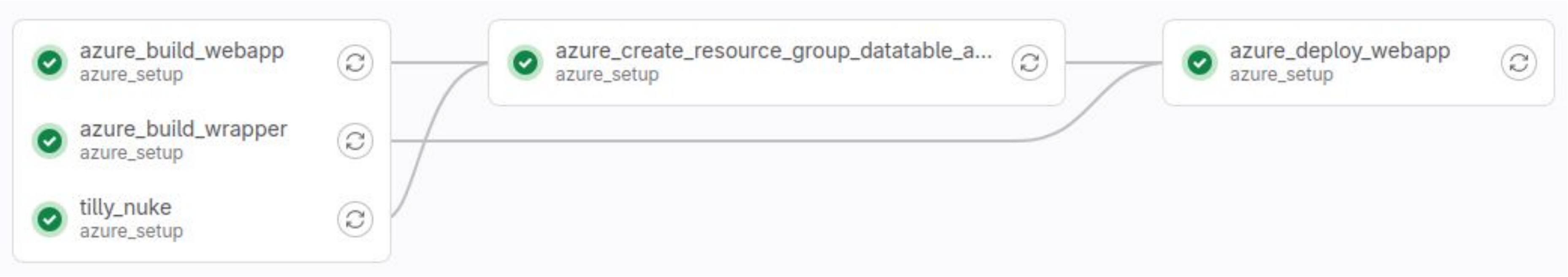
We are having trouble connecting to the Container Registry. Please try refreshing the page. If this error persists, please review the troubleshooting documentation .



```
Could not resolve host: inf-git.fh-rosenheim.de
```

Don’t have a screenshot of it, but once we couldn’t deploy because the Azure Location *westeurope* was over capacity

Slow job execution & parallelization



azure_build_webapp
azure_setup

azure_build_wrapper
azure_setup

tilly_nuke
azure_setup

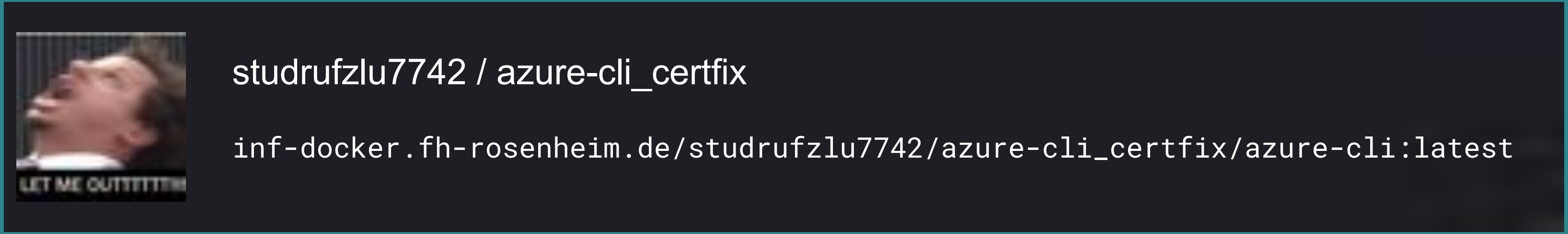
azure_create_resource_group_datatable_a...
azure_setup

azure_deploy_webapp
azure_setup

Parallel

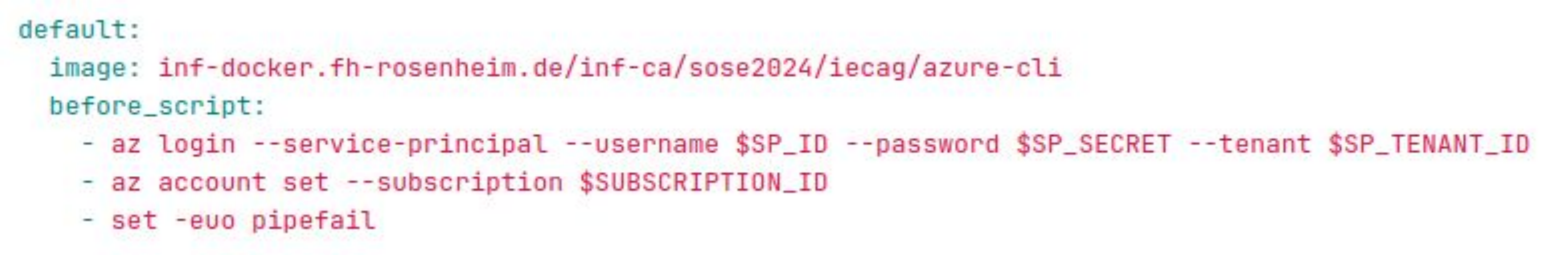
Absolute Unit

Certificate issues with Azure CLI



studrufzlu7742 / azure-cli_certfix

```
inf-docker.fh-rosenheim.de/studrufzlu7742/azure-cli_certfix/azure-cli:latest
```



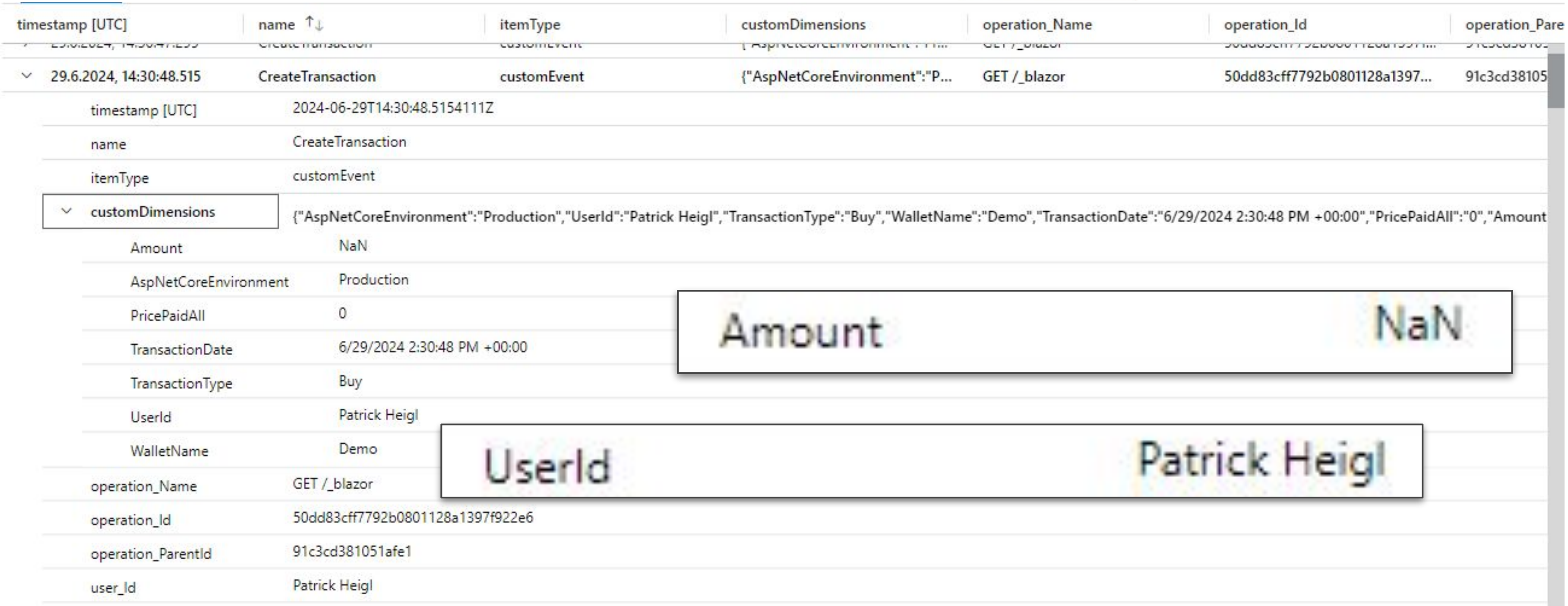
```
default:
  image: inf-docker.fh-rosenheim.de/inf-ca/sose2024/iecag/azure-cli
  before_script:
    - az login --service-principal --username $SP_ID --password $SP_SECRET --tenant $SP_TENANT_ID
    - az account set --subscription $SUBSCRIPTION_ID
    - set -euo pipefail
```

Why we used an App Service

for the Blazor WebApp

- **Costs:** Currently, there is a free App Service plan on Azure
- **Knowledge:** I have already created an App Service during my internship
- **Scalable:** With current configuration the website can handle over 2000 users. If needed the plan can be switched to a better one in 1 minute on azure portal

Intruder detected!!11!!!1!



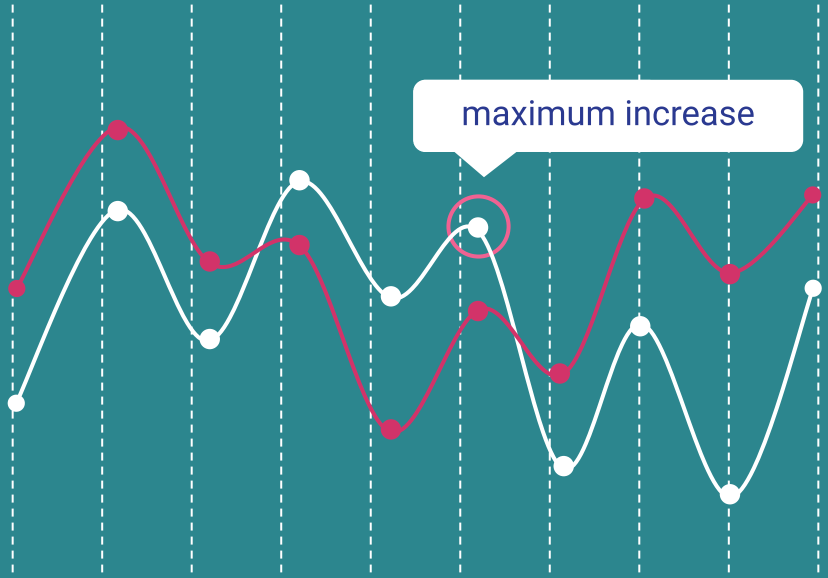
timestamp [UTC]	name	itemType	customDimensions	operation_Name	operation_Id	operation_Pare
29.6.2024, 14:30:48.515	CreateTransaction	customEvent	{"AspNetCoreEnvironment":"P...	GET /_blazor	50dd83cff7792b0801128a1397...	91c3cd38105

timestamp [UTC]	2024-06-29T14:30:48.5154111Z
name	CreateTransaction
itemType	customEvent
customDimensions	{"AspNetCoreEnvironment":"Production","UserId":"Patrick Heigl","TransactionType":"Buy","WalletName":"Demo","TransactionDate":"6/29/2024 2:30:48 PM +00:00","PricePaidAll":"0","Amount
Amount	NaN
AspNetCoreEnvironment	Production
PricePaidAll	0
TransactionDate	6/29/2024 2:30:48 PM +00:00
TransactionType	Buy
UserId	Patrick Heigl
WalletName	Demo
operation_Name	GET /_blazor
operation_Id	50dd83cff7792b0801128a1397f922e6
operation_ParentId	91c3cd381051afe1
user_Id	Patrick Heigl

Amount NaN

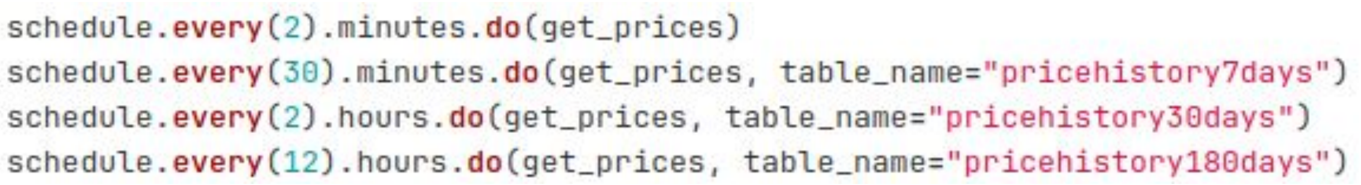
UserId Patrick Heigl

Implementation Examples



API Wrapper Logic

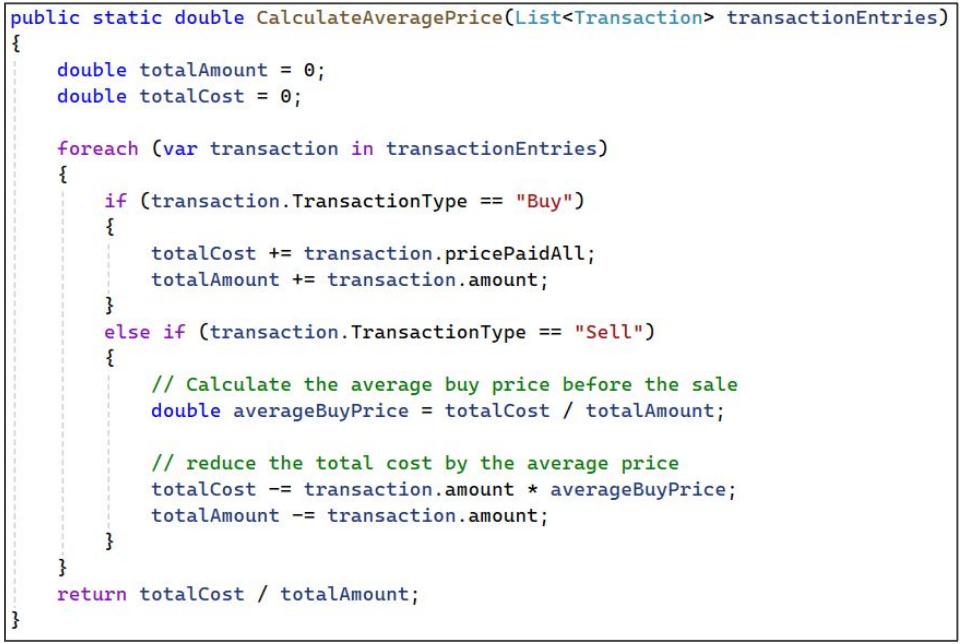
CoinGecko



```
schedule.every(2).minutes.do(get_prices)
schedule.every(30).minutes.do(get_prices, table_name="pricehistory7days")
schedule.every(2).hours.do(get_prices, table_name="pricehistory30days")
schedule.every(12).hours.do(get_prices, table_name="pricehistory180days")
```

- *current_prices* updated on every call

Wallet Calculations - CalculateAveragePrice



```
public static double CalculateAveragePrice(List<Transaction> transactionEntries)
{
    double totalAmount = 0;
    double totalCost = 0;

    foreach (var transaction in transactionEntries)
    {
        if (transaction.TransactionType == "Buy")
        {
            totalCost += transaction.pricePaidAll;
            totalAmount += transaction.amount;
        }
        else if (transaction.TransactionType == "Sell")
        {
            // Calculate the average buy price before the sale
            double averageBuyPrice = totalCost / totalAmount;

            // reduce the total cost by the average price
            totalCost -= transaction.amount * averageBuyPrice;
            totalAmount -= transaction.amount;
        }
    }
    return totalCost / totalAmount;
}
```

Returns the averagePrice as a double for the transactions.

Watch out:
When the transactionType is ‘Sell’, decrease the totalCost by the averagePrice that was calculated before the sale

Performance improvement:

- Sort the transactions by date
- Process only the transactions from the point where the amount was last zero

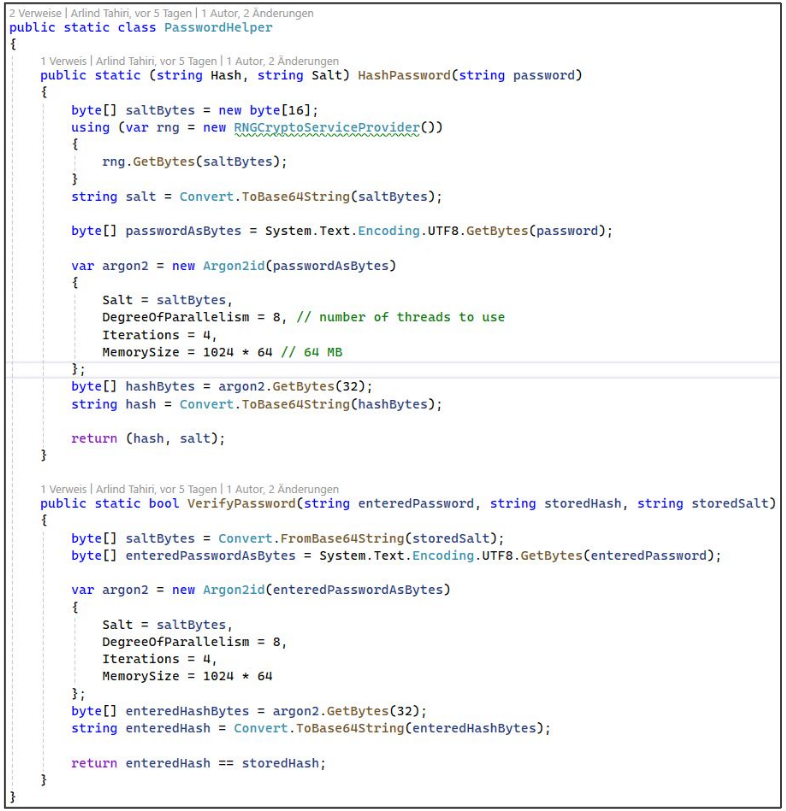
Wallet Calculations - CalculatePortfolioValueForTheLastXDays

Sequence:

1. Retrieve all the coins the user invested in
2. Fetch the current prices of the coins
3. **Parallel:** Group the transactions by coin name: set current price and average price
4. **Parallel:** Group the transactions by coin name: fetch the price of the currency and calculate portfolio value for each time segment (day/hour/...)
5. Return combined data

→ By running these operations in parallel, the code executes very quickly

Security - Password Hashing and Salting with Argon2



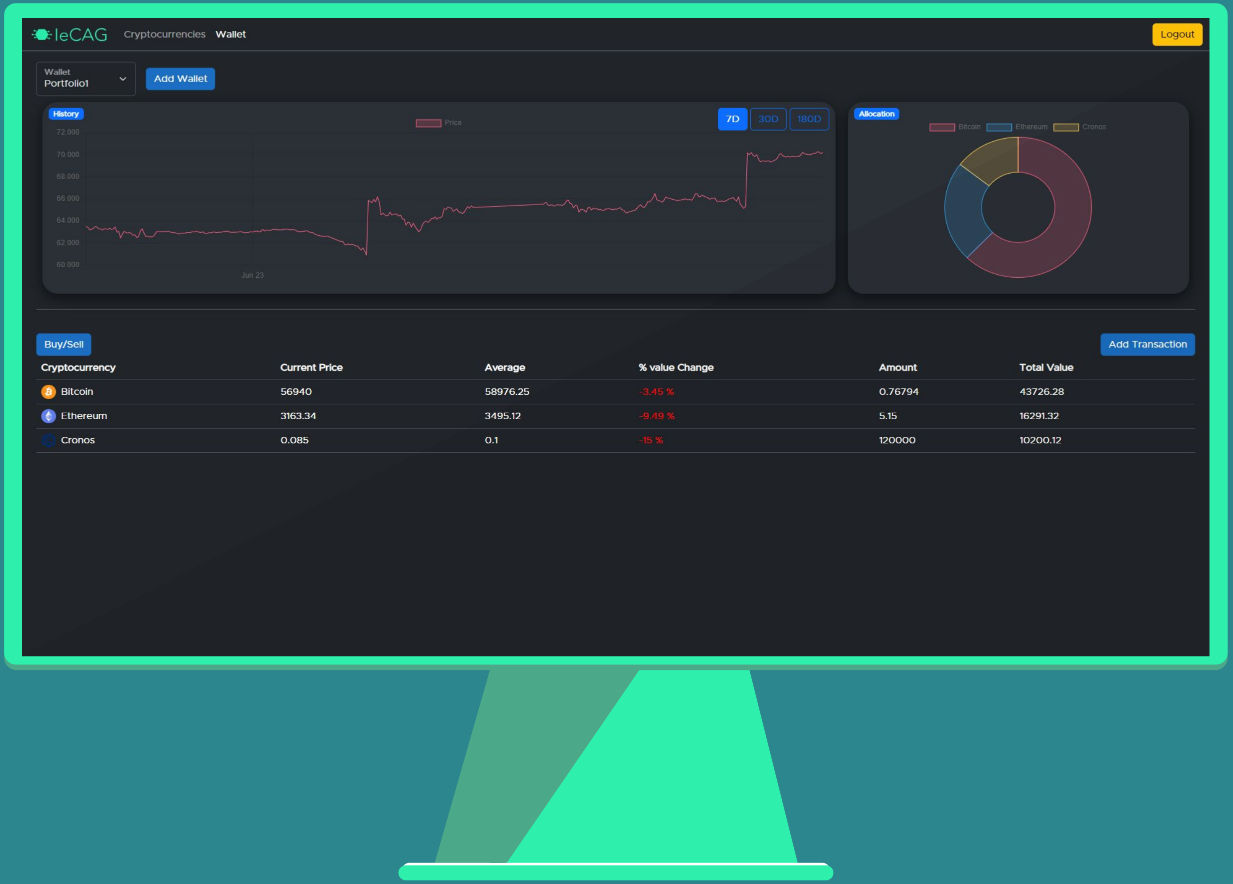
```
2 Verweise | Arlind Tahiri, vor 5 Tagen | 1 Autor, 2 Änderungen
public static class PasswordHelper
{
    1 Verweis | Arlind Tahiri, vor 5 Tagen | 1 Autor, 2 Änderungen
    public static (string Hash, string Salt) HashPassword(string password)
    {
        byte[] saltBytes = new byte[16];
        using (var rng = new RNGCryptoServiceProvider())
        {
            rng.GetBytes(saltBytes);
        }
        string salt = Convert.ToBase64String(saltBytes);

        byte[] passwordAsBytes = System.Text.Encoding.UTF8.GetBytes(password);

        var argon2 = new Argon2id(passwordAsBytes)
        {
            Salt = saltBytes,
            DegreeOfParallelism = 8, // number of threads to use
            Iterations = 4,
            MemorySize = 1024 * 64 // 64 MB
        };
        byte[] hashBytes = argon2.GetBytes(32);
        string hash = Convert.ToBase64String(hashBytes);

        return (hash, salt);
    }

    1 Verweis | Arlind Tahiri, vor 5 Tagen | 1 Autor, 2 Änderungen
    public static bool VerifyPassword(string enteredPassword, string storedHash, string storedSalt)
    {
        byte[] saltBytes = Convert.FromBase64String(storedSalt);
        byte[] enteredPasswordAsBytes = System.Text.Encoding.UTF8.GetBytes(enteredPassword);

        var argon2 = new Argon2id(enteredPasswordAsBytes)
        {
            Salt = saltBytes,
            DegreeOfParallelism = 8,
            Iterations = 4,
            MemorySize = 1024 * 64
        };
        byte[] enteredHashBytes = argon2.GetBytes(32);
        string enteredHash = Convert.ToBase64String(enteredHashBytes);

        return enteredHash == storedHash;
    }
}
```

Hashing with Argon2

Passwords are being stored hashed and salted on the Azure data table.

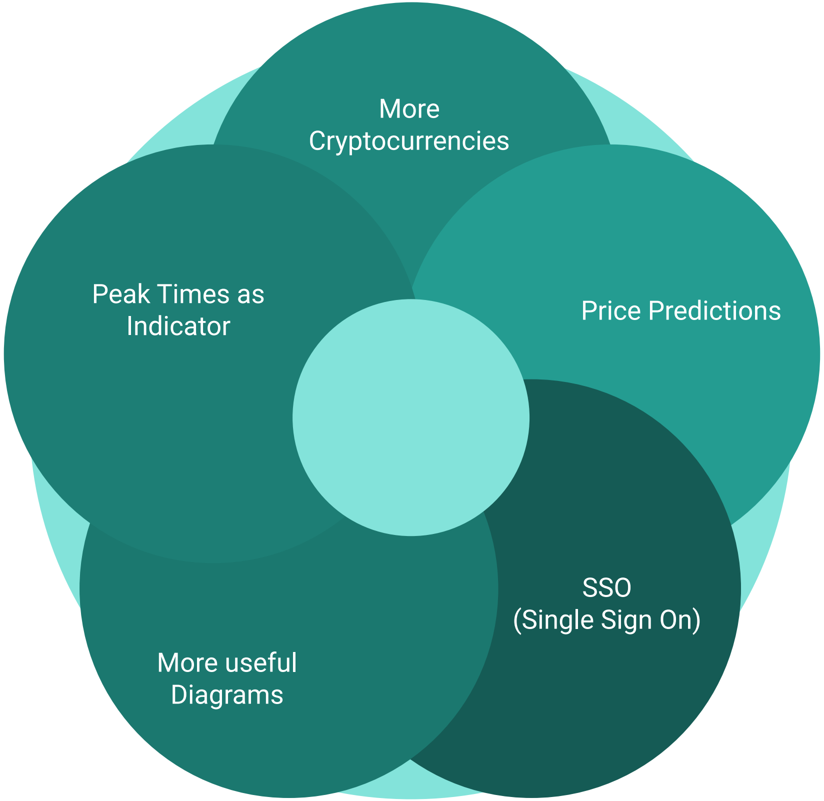
- Password hashing is a critical step in securing user passwords
- We use Argon2, a modern and secure hashing algorithm, to hash passwords
- Salting ensures that identical passwords have different hashes, providing an additional layer of security

Live - Demo



Cryptocurrency	Current Price	Average	% value Change	Amount	Total Value
Bitcoin	56940	58976.25	-3.45 %	0.76794	43726.28
Ethereum	3163.34	3495.12	-9.49 %	5.15	16291.32
Cronos	0.085	0.1	-15 %	120000	10200.12

Outlook



More Cryptocurrencies

Peak Times as Indicator

Price Predictions

SSO (Single Sign On)

More useful Diagrams

Link to website: