Architecture as Code (AaC) in 2023

Presented by: Ruslan Korniichuk 16.02.2023

Introduction

Google Trends | Compare

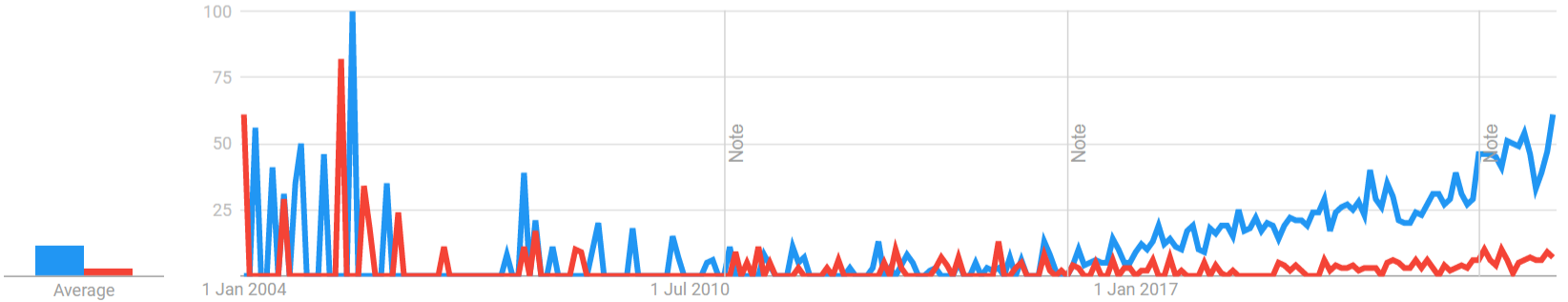
Infrastructure as Code
Search term

Architecture as Code
Search term

+ Add comparison

United States | 2004 – present | All categories | Web Search

Interest over time



Google Trends | Compare

Infrastructure as C...
Search term

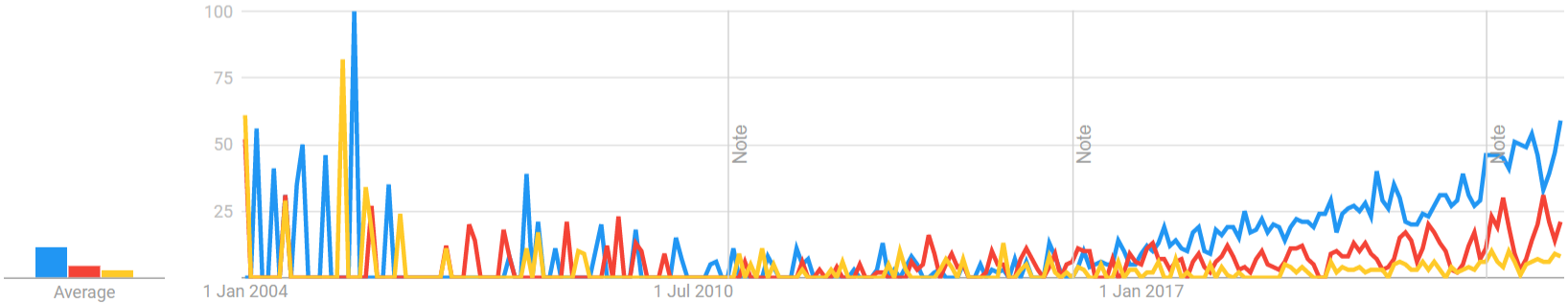
Diagram as Code
Search term

Architecture as Co...
Search term

+ Add comparison

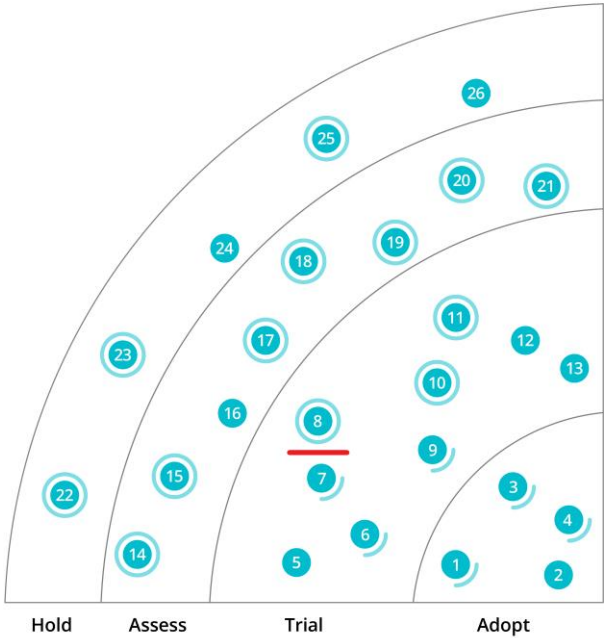
United States | 2004 – present | All categories | Web Search

Interest over time



Technology Radar by ThoughtWorks

October 2020



Adopt
1. Dependency drift fitness function
2. Run cost as architecture fitness function
3. Security policy as code
4. Tailored service templates

Trial
5. Continuous delivery for machine learning (CD4ML)
6. Data mesh
7. Declarative data pipeline definition
8. Diagrams as code
9. Distroless Docker images
10. Event interception
11. Parallel run with reconciliation
12. Use "remote native" processes and approaches
13. Zero trust architecture

Assess
14. Bounded low-code platforms
15. Browser-tailored polyfills
16. Decentralized identity
17. Kube-managed cloud services
18. Open Application Model (OAM)
19. Secure enclaves
20. Switchback experimentation
21. Verifiable credentials

Hold
22. Apollo Federation
23. ESBs in API Gateway's clothing
24. Log aggregation for business analytics
25. Micro frontend anarchy
26. Productionizing notebooks

Source: thght.works/3frtPKE

Infrastructure as Code vs Diagram as Code vs Architecture as Code

Infrastructure as Code (IaC) is the managing and provisioning of infrastructure through code instead of through manual processes.
Tools: Terraform, CloudFormation, Serverless Application Model (SAM), Cloud Development Kit (CDK).

Infrastructure as Code vs Diagram as Code vs Architecture as Code

Infrastructure as Code (IaC) is the managing and provisioning of infrastructure through code instead of through manual processes.
Tools: Terraform, CloudFormation, Serverless Application Model (SAM), Cloud Development Kit (CDK).

Diagram as Code is the creation of diagrams using code, and without any design tools.
Tools: Diagrams, Ilograph, Mermaid, Structurizr DSL.

Infrastructure as Code vs Diagram as Code vs Architecture as Code

Infrastructure as Code (IaC) is the managing and provisioning of infrastructure through code instead of through manual processes.
Tools: Terraform, CloudFormation, Serverless Application Model (SAM), Cloud Development Kit (CDK).

Diagram as Code is the creation of diagrams using code, and without any design tools.
Tools: Diagrams, Ilograph, Mermaid, Structurizr DSL.

Architecture as Code (AaC) is the prototyping and visualization of system architecture using code.
Tools: Diagrams (Python).

Diagram as Code tools

- AsciiDoctor Diagram: set of Asciidoctor extensions
- CloudGram: Domain-Specific Language (DSL)
- D2: DSL
- Graphviz: DOT language
- Ilograph: YAML language
- Mermaid: Mermaid's syntax (Markdown-inspired)
- PlantUML: PlantUML language
- Structurizr DSL: DSL
- WebSequenceDiagrams: DSL

Static vs interactive

Static diagrams (e.g., PNG, SVG):

- D2
- Graphviz
- Mermaid
- PlantUML
- WebSequenceDiagrams

Interactive diagrams (e.g., browser-based):

- CloudGram
- Ilograph
- Structurizr DSL

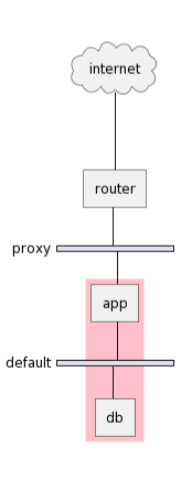
```
@startuml
nwdiag {
    internet [ shape = cloud];
    internet -- router;

    network proxy {
        router;
        app;
    }
    network default {
        app;
        db;
    }
    group {
      color = "pink";
      app;
      db;
    }
}
@enduml
```

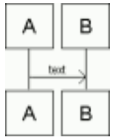
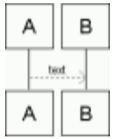
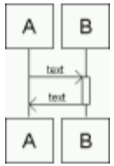
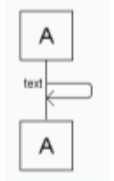
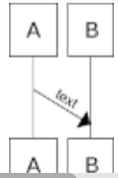
//www.plantuml.com/plantuml/png/LSrH3e8m30RWPtUAXdTEG4YuX_6Xiec4iJPb3Hh3tPtArFNjN_VzxPQ84dNs9gnsn04U1jAC8Je9Bi8HbYkoWnPwJsfFJRckQn3I51hpNgItbMG25hhTNrtxv4yv8_CdR0MpxeBgumuFoFmZz1mlXjJ8tmCzUH9eeU8nJ5KscHTuCvqBLcV_100

Decode URL

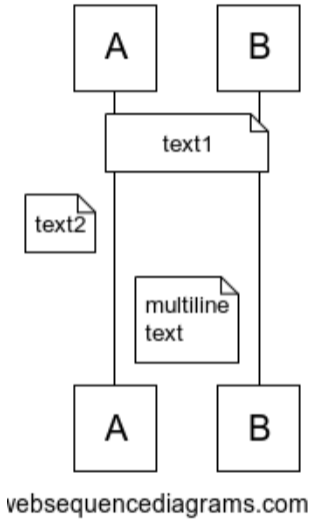
Submit

PNG SVG ASCII Art



```
note over A,B: text1
note left of A: text2
note right of A
    multiline
    text
end note
```



websequencediagrams.com

Untitled

Mermaid | Code | Config | Auto sync | DOCS

```
flowchart LR
    A[Hard] -->|Text| B(Round)
    B --> C{Decision}
    C -->|One| D[Result 1]
    C -->|Two| E[Result 2]
```

Sample Diagrams

History

Actions

COPY IMAGE TO CLIPBOARD

PNG | SVG | PNG | SVG | KROKI

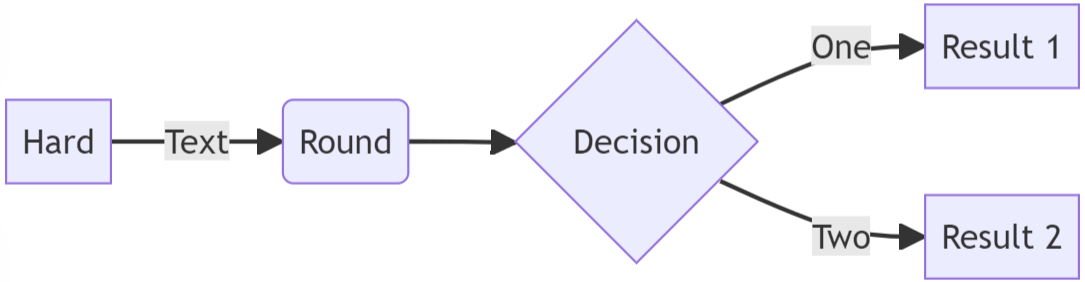
PNG size Auto Width Height

[![](https://mermaid.ink/img/pako:eNpVkE1PwzAMhv9 | COPY MARKDOWN

https://gist.github.com/sidharthv96/6268a23e673a533dcb198 | LOAD GIST

This site is powered by Netlify

Diagram | Pan & Zoom | FULL SCREEN



Mermaid | Code | Config | Auto sync | DOCS

```
C4Context
  title System Context diagram for Internet Banking System
  Enterprise_Boundary(b0, "BankBoundary0") {
    Person(customerA, "Banking Customer A", "A customer of the bar
    Person(customerB, "Banking Customer B")
    Person_Ext(customerC, "Banking Customer C", "desc")

    Person(customerD, "Banking Customer D", "A customer of the bar

    System(SystemAA, "Internet Banking System", "Allows customers

    Enterprise_Boundary(b1, "BankBoundary") {

      SystemDb_Ext(SystemE, "Mainframe Banking System", "Stores a

      System_Boundary(b2, "BankBoundary2") {
```

Sample Diagrams

History

Actions

COPY IMAGE TO CLIPBOARD

PNG | SVG | PNG | SVG | KROKI

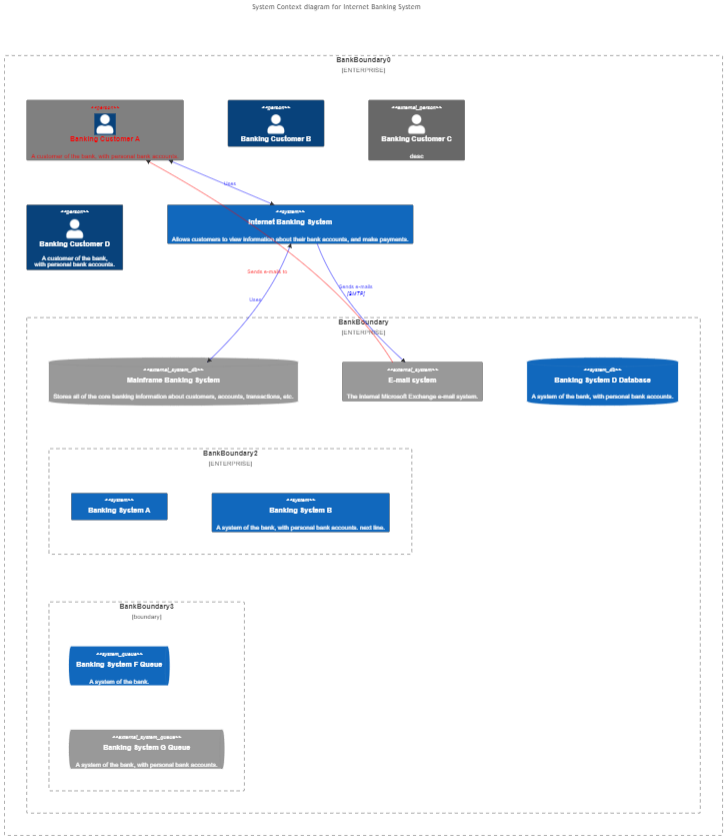
PNG size Auto Width Height

[![](https://mermaid.ink/img/pako:eNqdVm1v0zAQ_iu\ | COPY MARKDOWN

https://gist.github.com/sidharthv96/6268a23e673a533dcb198 | LOAD GIST

This site is powered by Netlify

Diagram | Pan & Zoom | FULL SCREEN



Structurizr DSL language reference

Upload | Render

```
workspace {

    model {
        u = person "User"
        s = softwareSystem "Software System" {
            webapp = container "Web Application" "" "Spring Boot"
            database = container "Database" "" "Relational database schema"
        }

        u -> webapp "Uses"
        webapp -> database "Reads from and writes to"

        live = deploymentEnvironment "Live" {
            deploymentNode "Amazon Web Services" {
                tags "Amazon Web Services - Cloud"

                deploymentNode "US-East-1" {
                    tags "Amazon Web Services - Region"

                    route53 = infrastructureNode "Route 53" {
                        tags "Amazon Web Services - Route 53"
                    }
                    elb = infrastructureNode "Elastic Load Balancer" {
                        tags "Amazon Web Services - Elastic Load Balancing"
                    }

                    deploymentNode "Amazon EC2" {
                        tags "Amazon Web Services - EC2"

                        deploymentNode "Ubuntu Server" {
                            webApplicationInstance = containerInstance webapp
                        }
                    }

                    deploymentNode "Amazon RDS" {
                        tags "Amazon Web Services - RDS"

                        deploymentNode "MySQL" {
                            tags "Amazon Web Services - RDS MySQL instance"

                            containerInstance database
                        }
                    }
                }
            }

            route53 -> elb "Forwards requests to" "HTTPS"
            elb -> webApplicationInstance "Forwards requests to" "HTTPS"
        }
    }

    views {
        deployment s live {
```

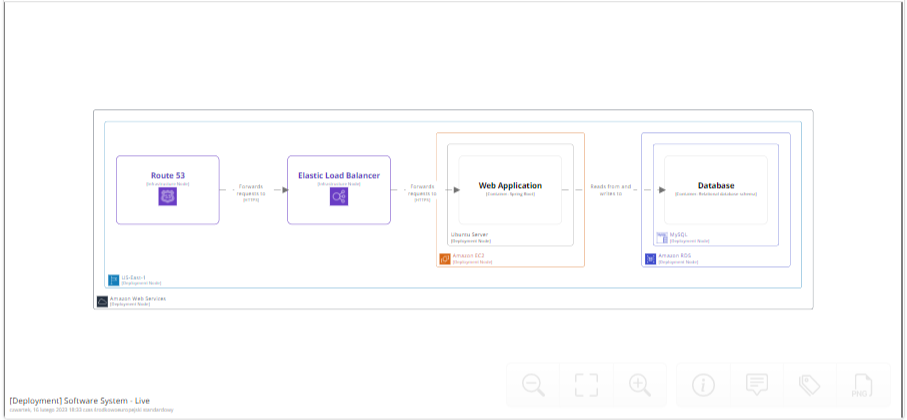
Some features (`!docs` , `!adrs` , `!script` , etc) are unavailable on this demo page - see Help - DSL for details.

DSL language reference | DSL cookbook | Examples: Getting started | Big Bank plc | Amazon Web Services

Structurizr Diagram | Structurizr Graph | Export PlantUML | Export C4-PlantUML | Export Mermaid | Export DOT | Export WebSequenceDiagrams | Export Ilograph

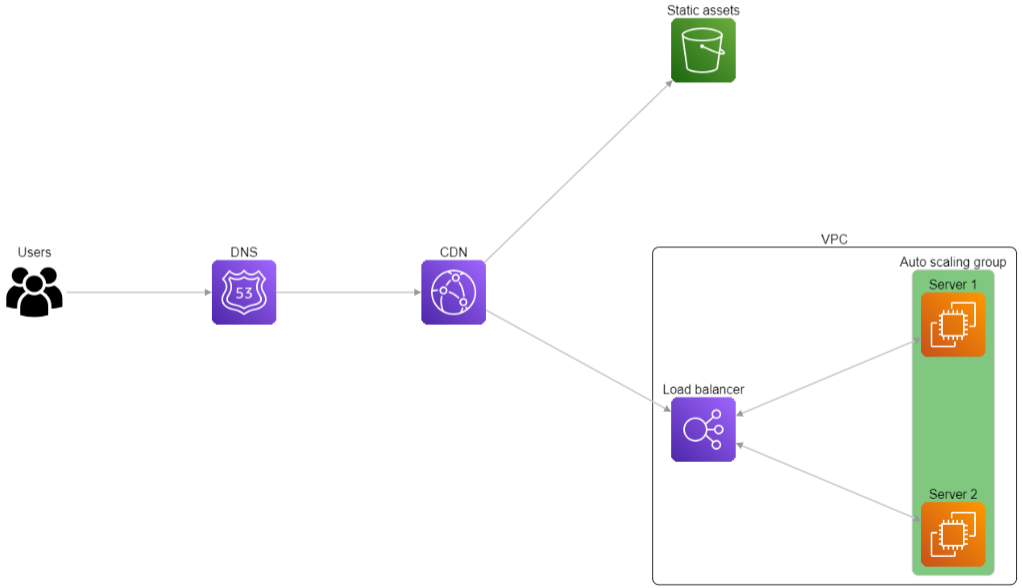
[Deployment] Software System - Live (#Deployment-1)

This diagram is not editable because automatic layout is enabled - see Help - Automatic layout for more details.



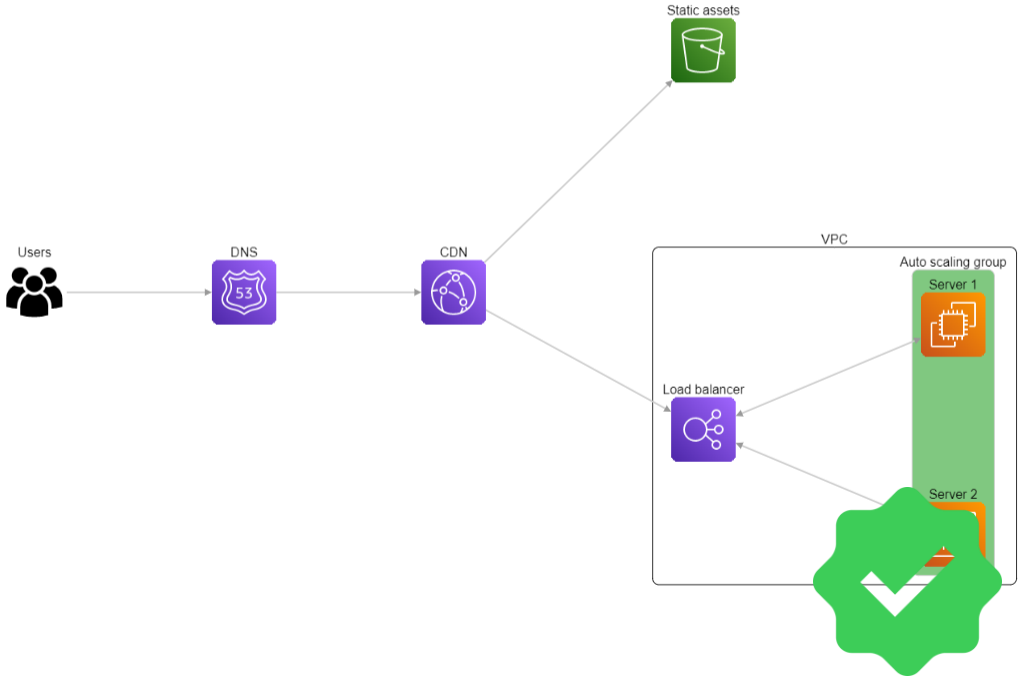
Embed Copy link png Save

```
diagram "example" [direction=lr] {
  // creating the nodes
  generic.users Users;
  aws.route53 DNS;
  aws.cloudfront cf [label="CDN"];
  aws.s3 s3 [label="Static assets"];

  group vpc [label="VPC",style=solid,stroke=black,opacity=0] {
      aws.elb load_balancer [label="Load balancer"];

      group asg [label="Auto scaling group",style=solid,fill="#80c880"] {
          aws.ec2 server1 [label="Server 1"];
          aws.ec2 server2 [label="Server 2"];
      }
  }

  // creating the edges
  Users -> DNS -> cf -> load_balancer <=> asg;
  cf -> s3;
}
```



Embed Copy link png Save

```
diagram "example" [direction=lr] {
  // creating the nodes
  generic.users Users;
  aws.route53 DNS;
  aws.cloudfront cf [label="CDN"];
  aws.s3 s3 [label="Static assets"];

  group vpc [label="VPC",style=solid,stroke=black,opacity=0] {
      aws.elb load_balancer [label="Load balancer"];

      group asg [label="Auto scaling group",style=solid,fill="#80c880"] {
          aws.ec2 server1 [label="Server 1"];
          aws.ec2 server2 [label="Server 2"];
      }
  }

  // creating the edges
  Users -> DNS -> cf -> load_balancer <=> asg;
  cf -> s3;
}
```

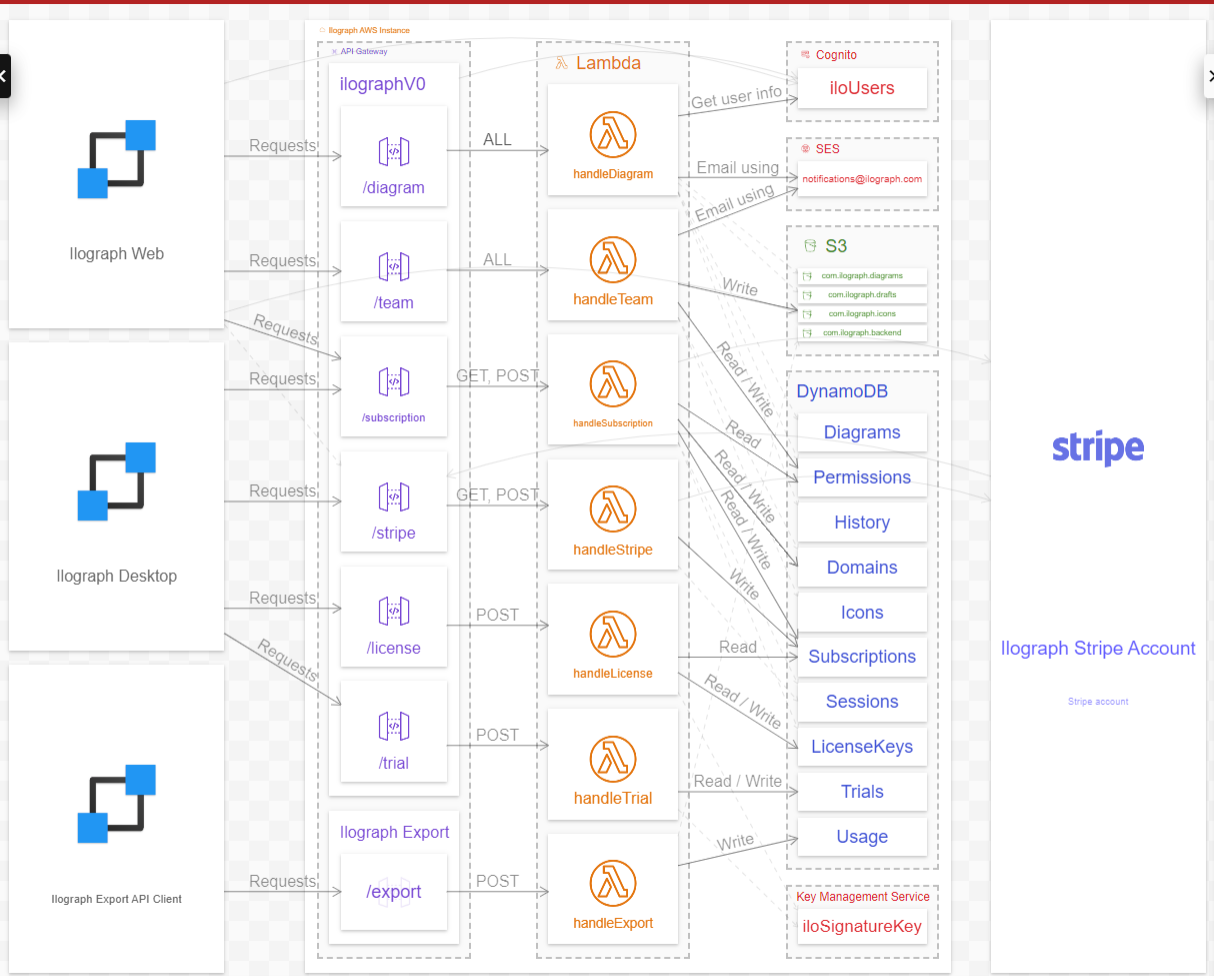


New diagram

This diagram is read-only. Click here to create a new diagram or log in now.

```
imports:
- from: ilograph/aws
  namespace: AWS

resources:
- name: Ilograph Web
  subtitle: Web Application
  description: Advanced diagramming web
    application for creating interactive,
    multi-perspective diagrams in the browser
  icon: Companies/ilograph.png

- name: Ilograph Desktop
  subtitle: Desktop Application
  description: Advanced diagramming [desktop
    application](https://www.ilograph.com
    /desktop/) for creating interactive, multi
    -perspective diagrams
  icon: Companies/ilograph.png

- name: Ilograph Export API Client
  subtitle: NPM Package and CLI Tool
  description: Package and CLI [tool](https
    ://www.ilograph.com/export-api/index.html)
    for exporting Ilograph diagrams to
    standalone HTML files

  icon: Companies/ilograph.png

- name: User
  subtitle: Application User
  description: End user of Ilograph
  icon: AWS/_General/User.svg

- name: Ilograph AWS Instance
  instanceOf: AWS::AWS Instance
  children:
  - name: CloudFront
    instanceOf: AWS::CloudFront
    description: A CDN which speeds up the
      distribution of content to end users
    children:
    - name: d3dxxxxx.cloudfront.net
      instanceOf: AWS::CloudFront
        ::Distribution

    - name: d2wxxxxx.cloudfront.net
      instanceOf: AWS::CloudFront
        ::Distribution

    - name: d3exxxxx.cloudfront.net
      instanceOf: AWS::CloudFront
        ::Distribution

  - name: API Gateway
    instanceOf: AWS::ApiGateway
    description: Service for creating REST
      APIs for Lambdas and other services
    children:
    - name: ilographV0
      instanceOf: AWS::ApiGateway::RestApi
      description: The Ilograph REST API
      children:
      - id: diagram
        name: /diagram
        instanceOf: AWS::ApiGateway::Resource
        description: Resource for Ilograph
          diagrams, their drafts, and their
          permissions
```

PROBLEMS 0 WARNINGS 0



Start Walkthrough

Perspective notes

This perspective shows the run-time dependencies between resources when fulfilling requests to the Ilograph back-end.

This is a header

The back-end uses a classic AWS Serverless architecture, with an *API Gateway* API calling *Lambda* functions backed by *DynamoDB* tables and *S3* buckets.

Requests related to purchasing and licenses also have an external dependency on Stripe.

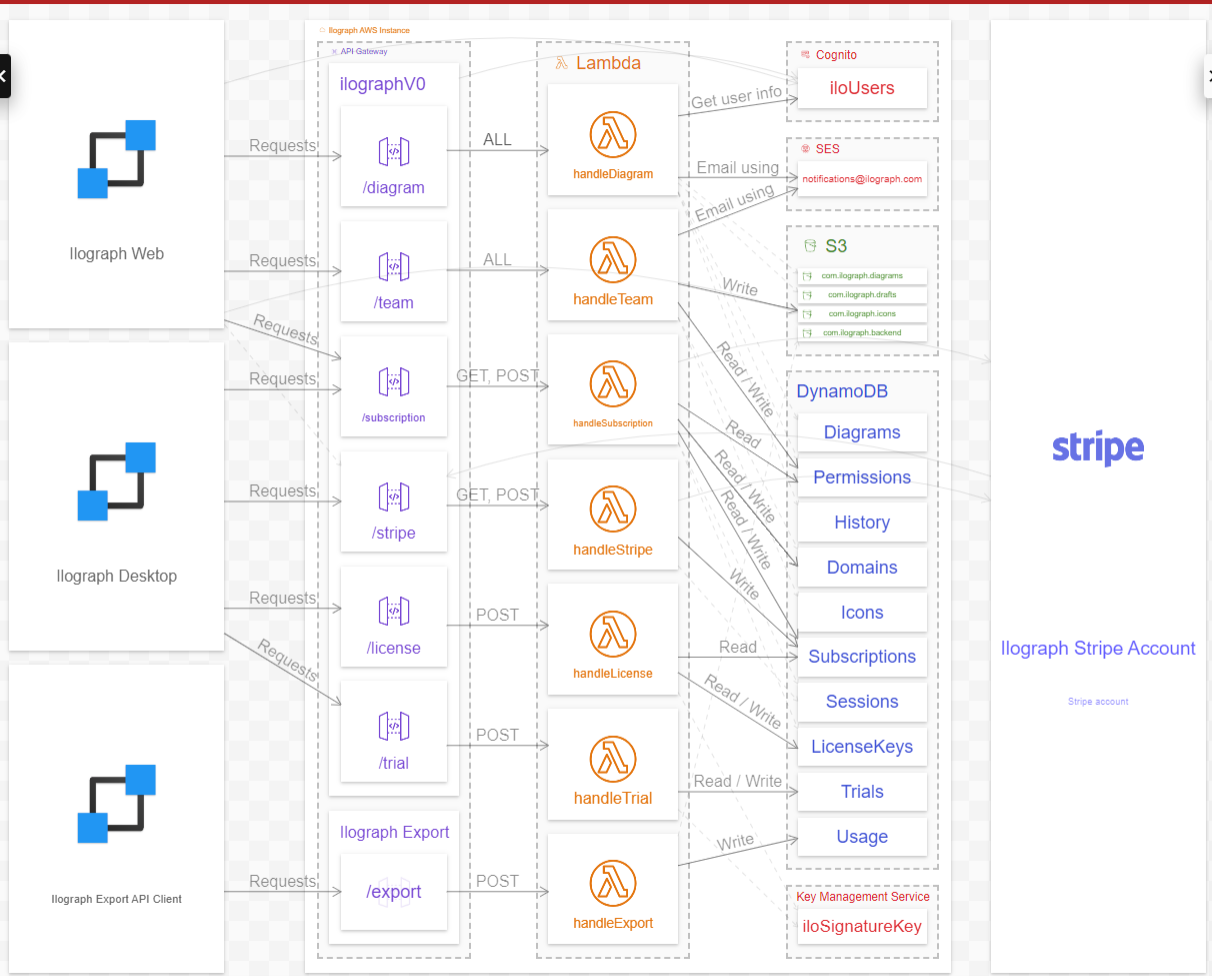
Request | Code | DNS + Origins | Get diagram | Update diagram | Invite teammate | Upload icon | Purchase subscription | Claim subscription | Activate licen

New diagram

This diagram is read-only. Click here to create a new diagram or log in now.

```
imports:
- from: ilograph/aws
  namespace: AWS

resources:
- name: Ilograph Web
  subtitle: Web Application
  description: Advanced diagramming web
    application for creating interactive,
    multi-perspective diagrams in the browser
  icon: Companies/ilograph.png

- name: Ilograph Desktop
  subtitle: Desktop Application
  description: Advanced diagramming [desktop
    application](https://www.ilograph.com
    /desktop/) for creating interactive, multi
    -perspective diagrams
  icon: Companies/ilograph.png

- name: Ilograph Export API Client
  subtitle: NPM Package and CLI Tool
  description: Package and CLI [tool](https
    ://www.ilograph.com/export-api/index.html)
    for exporting Ilograph diagrams to
    standalone HTML files

  icon: Companies/ilograph.png

- name: User
  subtitle: Application User
  description: End user of Ilograph
  icon: AWS/_General/User.svg

- name: Ilograph AWS Instance
  instanceOf: AWS::AWS Instance
  children:
  - name: CloudFront
    instanceOf: AWS::CloudFront
    description: A CDN which speeds up the
      distribution of content to end users
    children:
    - name: d3dxxxxx.cloudfront.net
      instanceOf: AWS::CloudFront
        ::Distribution

    - name: d2wxxxxx.cloudfront.net
      instanceOf: AWS::CloudFront
        ::Distribution

    - name: d3exxxxx.cloudfront.net
      instanceOf: AWS::CloudFront
        ::Distribution

  - name: API Gateway
    instanceOf: AWS::ApiGateway
    description: Service for creating REST
      APIs for lambdas and other services
    children:
    - name: ilographV0
      instanceOf: AWS::ApiGateway::RestApi
      description: The Ilograph REST API
      children:
      - id: diagram
        name: /diagram
        instanceOf: AWS::ApiGateway::Resource
        description: Resource for Ilograph
          diagrams, their drafts, and their
          permissions
```

PROBLEMS 0 WARNINGS 0



Start Walkthrough

Perspective notes

This perspective shows the run-time dependencies between resources when fulfilling requests to the Ilograph back-end.

This is a header

The back-end uses a classic AWS Serverless architecture, with an *API Gateway* API calling *Lambda* functions backed by *DynamoDB* tables and *S3* buckets.

Requests related to purchasing and licenses also have an external dependency on Stripe.

Request | Code | DNS + Origins | Get diagram | Update diagram | Invite teammate | Upload icon | Purchase subscription | Claim subscription | Activate licen

"Terraform to CDK" trend

Migration from Terraform and CloudFormation to Cloud Development Kit (CDK).

JSON/YAML → Python

Problem statement

Do you use UML?

Do you use Microsoft Visio?

Do you use Draw.io?

Do you use whiteboard?

UML, PowerPoint, Visio, and even Draw.io are not good enough for engineers to create system architecture in a fun way.

Solution

MAKE A DONATION TO OUR CHARITY FUND

The Charity Foundation "Come Back Alive" regularly and timely reports on its activities to all benefactors, state bodies and Ukrainian society.

READ THE REPORTS

Select the destination of your assistance

ARMY ASSISTANCE | SUPPORT THE FUND

All funds raised to help the army will be spent exclusively on purchases for the Armed Forces of Ukraine and other special forces in order to increase Ukraine's defense capabilities.

Payment method

PAYMENT BY CARD | SWIFT TRANSFERS | CRYPTOCURRENCY

Regularity

SUBSCRIPTION | ONE TIME

Payment currency

UAH, ₴ | USD, $ | EUR, €

Amount of donation*

€ 0

Email*

example@example.com

PAY

FAQ

HOW DO I KNOW WHAT'S REALLY GOING ON IN UKRAINE? +

I DON'T WANT TO DONATE MONEY TO THE WAR. HOW CAN I HELP? +

MAKE A DONATION TO OUR CHARITY FUND

The Charity Foundation "Come Back Alive" regularly and timely reports on its activities to all benefactors, state bodies and Ukrainian society.

READ THE REPORTS

Select the destination of your assistance

ARMY ASSISTANCE | SUPPORT THE FUND

All funds raised to help the army will be spent exclusively on purchases for the Armed Forces of Ukraine and other special forces in order to increase Ukraine's defense capabilities.

Payment method

PAYMENT BY CARD | SWIFT TRANSFERS | CRYPTOCURRENCY

Regularity

SUBSCRIPTION | ONE TIME

Payment currency

UAH, ₴ | USD, $ | EUR, €

Amount of donation*

€ 0

Email*

example@example.com

PAY

FAQ

HOW DO I KNOW WHAT'S REALLY GOING ON IN UKRAINE? +

I DON'T WANT TO DONATE MONEY TO THE WAR. HOW CAN I HELP? +

Architecture as Code (AaC) is easy to create, diff, version control, collaborate on, integrate into CI/CD.

A slide with no useful information at all

- Just filling in the gap between the last slide and the next one (which will be along in just a moment).
- No need to write this down, unless you feel compelled to do so.
- Nothing on this slide is examinable.
- In fact, I am not really sure why I bothered with it.

A slide with no useful information at all

- Just filling in the gap between the last slide and the next one (which will be along in just a moment).
- No need to write this down, unless you feel compelled to do so.
- Nothing on this slide is examinable.
- In fact, I am not really sure why I bothered with it.

Architecture as Code

pros

- track architecture diagram changes with version control
- visualize existing system architecture (e.g., Cloudiscovery tool)
- cloud agnostic (AWS, Azure, GCP, Alibaba, Oracle, etc.)
- reuse of source code and collaboration
- better for tax deduction
- expect coding 🚀

cons

- better for up-front design than long-lived documentation
- more for engineers
- whiteboard is better for prototyping

Diagrams Python lib

mingrammer / diagrams

Public

Watch 346 | Fork 1.7k | Starred 27.8k

Code | Issues 238 | Pull requests 72 | Discussions | Actions | Projects | Security | Insights

master | 8 branches | 40 tags | Go to file | Add file | Code

mingrammer bump: up to version 0.23.3 — a71e447 on Jan 13 — 511 commits

.github	ci(test): use ubuntu 20.04 version to run tests	last month
assets/img	fix: resize logo	4 years ago
diagrams	refactor(pvd/elastic): add an upper word and some aliases	last month
docker/dev	chore: fixing go get deprecation (#713)	6 months ago
docs	fix: rename frog to jfrog	last month
resources	fix: rename frog to jfrog	last month
scripts	docs: Add icons to docs (#499)	3 months ago
templates	docs: Add icons to docs (#499)	3 months ago
tests	fix(tests): set lower for case sensitive system (#805)	last month
website	docs(website): generate missing website images (#806)	last month
.gitignore	Add programming languages and frameworks (#112)	3 years ago
CHANGELOG.md	docs(changelog): update PR links	3 years ago
CONTRIBUTING.md	docs: fix simple typo, clearning -> cleaning (#724)	3 months ago
DEVELOPMENT.md	docs: fix typo (#730)	7 months ago
LICENSE	license: create LICENSE.md	4 years ago
README.md	docs: update README.md (#757)	3 months ago
autogen.sh	style: split the provider list by newline	3 months ago
config.py	refactor(pvd/elastic): add an upper word and some aliases	last month
poetry.lock	chore(deps-dev): bump pytest from 7.0.1 to 7.2.0 (#786)	last month
pyproject.toml	bump: up to version 0.23.3	last month

About

Diagram as Code for prototyping cloud system architectures

diagrams.mingrammer.com

graphviz | diagram | architecture | diagram-as-code

Readme
MIT license
27.8k stars
346 watching
1.7k forks

Releases 39

v0.23.3 Latest
on Jan 13

+ 38 releases

Used by 832

+ 824

Contributors 128

+ 117 contributors

Diagrams Python lib

```python
from diagrams import Diagram
from diagrams.aws.compute import EC2
from diagrams.aws.database import RDS
from diagrams.aws.network import ELB

with Diagram("Example 1", direction="TB", show=False):
    ELB("lb") >> [EC2("worker1"),
                  EC2("worker2"),
                  EC2("worker3"),
                  EC2("worker4"),
                  EC2("worker5")] >> RDS("events")
```

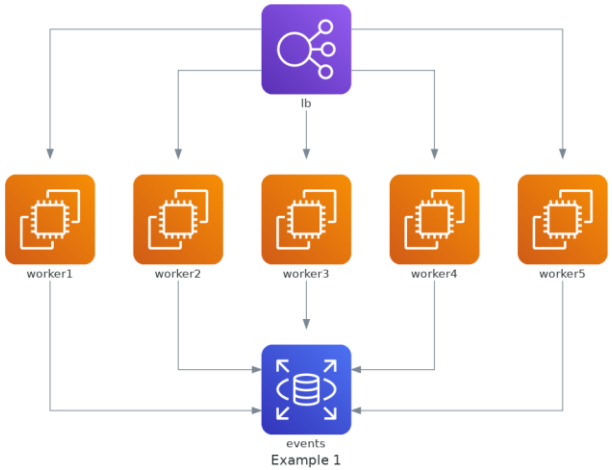


Diagram class

```
from diagrams import Diagram
from diagrams.aws.compute import EC2

graph_attr = {
    "bgcolor": "white"
}

with Diagram(
        name="Example 2",
        filename="example2",
        outformat=["png", "dot"],
        direction="LR",
        graph_attr=graph_attr,
        show=False):
    EC2("web")
```

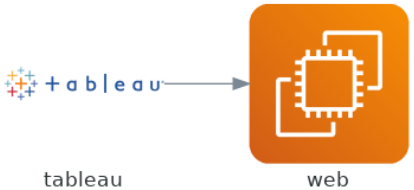
Nodes

```python
from diagrams import Diagram
from diagrams.aws.compute import EC2

with Diagram("Example 3", filename="example3", show=False):
    EC2("web")
```

Example 3

Custom nodes

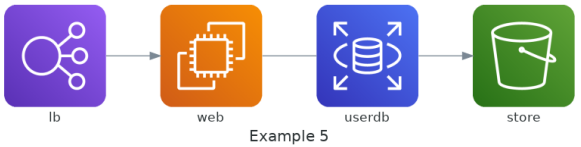
```
from diagrams import Diagram
from diagrams.aws.compute import EC2
from diagrams.custom import Custom

with Diagram("Example 4", filename="example4", show=False):
    tableau = Custom("tableau", "tableau_logo.png")

    tableau >> EC2("web")
```



Example 4

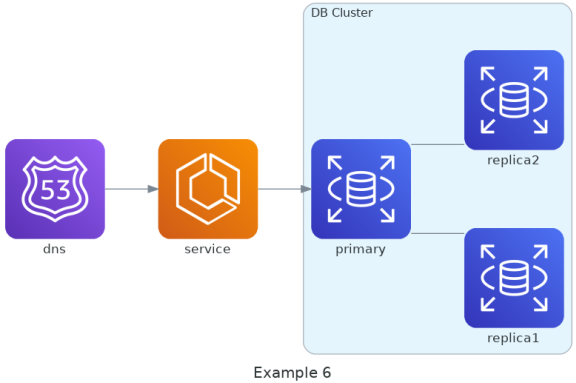
Data flow

```
from diagrams import Diagram
from diagrams.aws.compute import EC2
from diagrams.aws.database import RDS
from diagrams.aws.network import ELB
from diagrams.aws.storage import S3

with Diagram("Example 5", filename="example5", show=False):
    elb = ELB("lb")
    ec2 = EC2("web")
    rds = RDS("userdb")
    s3 = S3("store")

    (elb >> ec2) - rds >> s3
```



Clusters

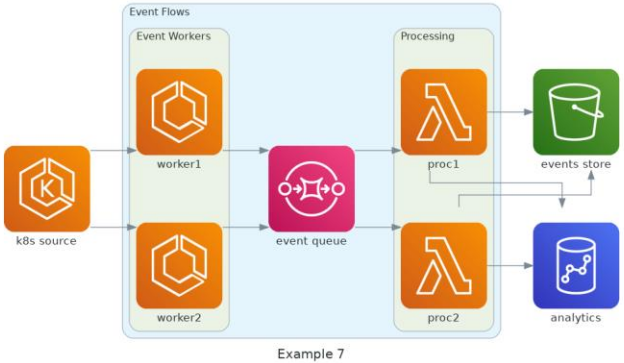
```
from diagrams import Cluster, Diagram
from diagrams.aws.compute import ECS
from diagrams.aws.database import RDS
from diagrams.aws.network import Route53

with Diagram("Example 6", filename="example6", show=False):
    dns = Route53("dns")
    web = ECS("service")

    with Cluster("DB Cluster"):
        db_primary = RDS("primary")
        db_primary - [RDS("replica1"),
                      RDS("replica2")]

    dns >> web >> db_primary
```



Example 6

Nested clusters

```
from diagrams import Cluster, Diagram
from diagrams.aws.compute import ECS, EKS, Lambda
from diagrams.aws.database import Redshift
from diagrams.aws.integration import SQS
from diagrams.aws.storage import S3

with Diagram("Example 7", filename="example7", show=False):
    source = EKS("k8s source")

    with Cluster("Event Flows"):
        with Cluster("Event Workers"):
            workers = [ECS("worker1"),
                       ECS("worker2")]

        queue = SQS("event queue")

        with Cluster("Processing"):
            handlers = [Lambda("proc1"),
                        Lambda("proc2")]

    store = S3("events store")
    dw = Redshift("analytics")

    source >> workers >> queue >> handlers
    handlers >> store
    handlers >> dw
```



Example 7

Cloud-Architects / cloudiscovery

Public

Watch 20 | Fork 70 | Star 604

Code | Issues 24 | Pull requests 2 | Security | Insights

develop | 4 branches | 23 tags | Go to file | Add file | Code

leandrodamascena Merge pull request #175 from Cloud-Architects/python_39 ... fad132e on Jan 4, 2021 | 510 commits

.circleci	fixed deployment pipeline	3 years ago
bin	renamed tool, 2.1.0 release	3 years ago
cloudiscovery	v2.4.4	3 years ago
docs	Moved out AWS specific code to separate files	3 years ago
.codacy.yml	renamed tool, 2.1.0 release	3 years ago
.coveragerc	excluded tests from code coverage	3 years ago
.gitignore	restored readme assets	3 years ago
.pre-commit-config.yaml	added prospector to precommit, resolved all issues	3 years ago
.prospector.yaml	#97 added public and private subnet VPC case	3 years ago
.pylintrc	Added check if service is available in region #98	3 years ago
Dockerfile	#97 implemented generation of empty draw.io file	3 years ago
LICENSE	README issues and version bump	3 years ago
README.md	Version bump and readme adjusts	3 years ago
dependabot.yml	Added dependabot	3 years ago
pytest.ini	renamed tool, 2.1.0 release	3 years ago
requirements-dev.txt	trying making automated release - 2.1.0; added twine	3 years ago
requirements.txt	#127 code review	3 years ago
setup.cfg	#127 code review	3 years ago
setup.py	Version bump, python version updated and readme issues	3 years ago

README.md

About

The tool to help you discover resources in the cloud environment

python | iot | aws | aws-cli | vpc | diagrams | aws-iot | devops-tools | aws-monitoring | iam-policy | aws-services

Readme

Apache-2.0 license

604 stars

20 watching

70 forks

Releases 19

cloud-discovery 2.4.4 Latest

on Jan 4, 2021

+ 18 releases

Contributors 2

meshuga Patryk Orwat

leandrodamascena Leandro Damascena

Languages

Python 97.9% | HTML 1.6% | Other 0.5%

Closing slides

Life Is On Schneider Electric

korniichuk / architecture-as-code-pycon-pl-2022 Public

Pin Unwatch 2 Fork 0 Star 1

Code Issues Pull requests Actions Projects Wiki Security Insights Settings

main 1 branch 0 tags Go to file Add file Code

korniichuk Improve readbility 4d14417 on Nov 3, 2022 4 commits

.gitignore	Initial commit	3 months ago
LICENSE	Initial commit	3 months ago
README.md	Initial commit	3 months ago
d1_example.py	Improve readbility	3 months ago
d2_diagram_class.py	Fix diagram 2	3 months ago
d3_nodes.py	Initial commit	3 months ago
d4_custom_nodes.py	Initial commit	3 months ago
d5_data_flow.py	Initial commit	3 months ago
d6_clusters.py	Initial commit	3 months ago
d7_nested_clusters.py	Initial commit	3 months ago
example1.png	Initial commit	3 months ago
example2.dot	Fix diagram 2	3 months ago
example2.png	Initial commit	3 months ago
example3.png	Initial commit	3 months ago
example4.png	Initial commit	3 months ago
example5.png	Initial commit	3 months ago
example6.png	Initial commit	3 months ago
example7.png	Initial commit	3 months ago
tableau_logo.png	Initial commit	3 months ago

README.md

About

Architecture as Code (AaC) with Python or way to become your own boss

Readme
Unlicense license
1 star
2 watching
0 forks

Releases

No releases published
Create a new release

Packages

No packages published
Publish your first package

Languages

Python 100.0%

Ruslan Korniichuk

Software Engineer (We're Hiring!)

Katowice, Śląskie, Poland · Contact info

500+ connections

Schneider Electric

Open to | Add profile section | More

Analytics

Private to you

217 profile views
Discover who's viewed your profile.

1,348 post impressions
Check out who's engaging with your posts.

105 search appearances
See how often you appear in search results.

Resources

Private to you

Creator mode Off
Get discovered, showcase content on your profile, and get access to creator tools

My network
See and manage your connections and interests.

Show all 5 resources →

About

Software and Artificial Intelligence Engineer based in Poland.

Solid experience of 7+ years in IT. Graduated as Information Systems and Intelligent Systems Engineer. Artificial Intelligence Architect at Schneider Electric. Former Python Developer and Data Engineer at Fortune 500 companies.

Edit public profile & URL

Add profile in another language

People also viewed

Sinem Ayyildiz • 1st
Backend Developer Intern @Schneider Electric
Message

Michal Dulemba • 1st
Machine learning engineer at Egnyte, podcaster at nieliniowy.pl
Message

Jakub Wawrzyniak • 2nd
Chief Technology Officer | Researcher at TIDK
+ Follow

Dawid Bienias • 1st
Python Software Engineer
Message

Laskowska Karolina • 2nd
Talent Acquisition Partner @ Schneider Electric
+ Follow

Show more

People you may know

Alexandre Clos Soler
Senior Data Engineer at Schneider Electric

Questions?