



Architecture as Code (AaC) with Python

Ruslan Kornichuk | 23.04.2025

ver. 1.0.0

TABLE OF CONTENTS

01

Introduction

02

Problem statement

03

Solution

04

Diagrams
Python lib

05

Closing slides

01

Introduction

Infrastructure as Code
Search term

Architecture as Code
Search term

+ Add comparison

United States

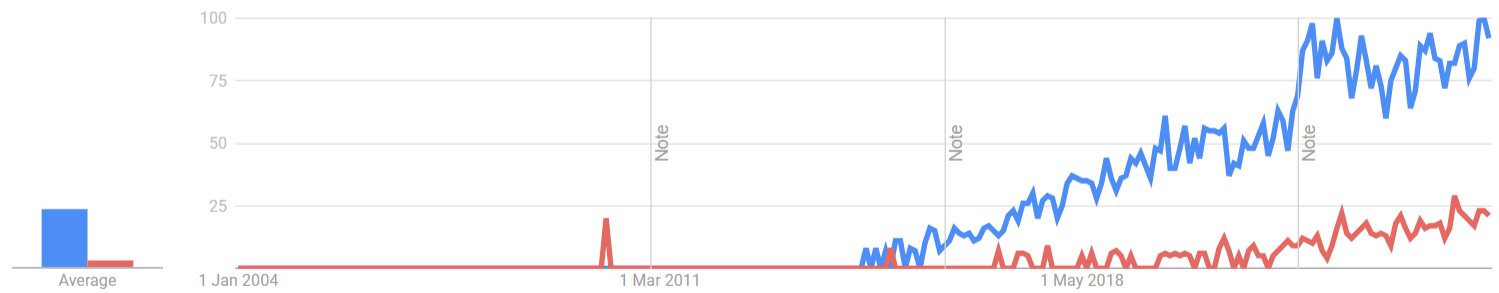
2004 – present

All categories

Web Search

Interest over time

Download Compare Share



Infrastructure as ...
Search term

Diagram as Code
Search term

Architecture as C...
Search term

+ Add comparison

United States

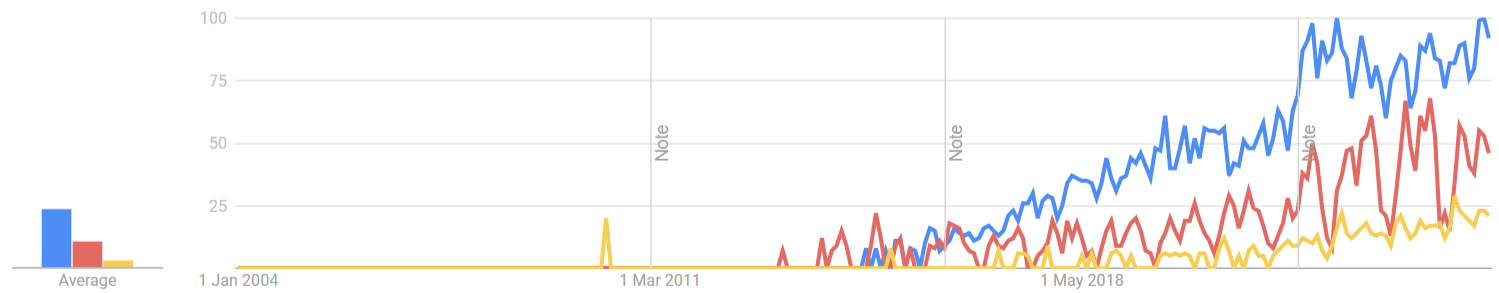
2004 – present

All categories

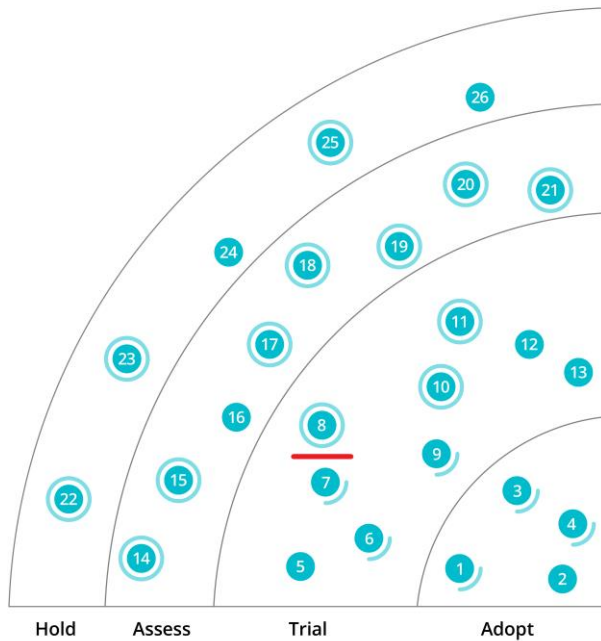
Web Search

Interest over time

Download Full Screen Share



Technology Radar by ThoughtWorks



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. Distrosless 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: tinyurl.com/2nrh4uw2

Infrastructure as Code vs Diagram as Code vs Architecture as Code

Infrastructure as Code (IaC) is the managing and provisioning of infrastructure through code.

Tools: Terraform, Pulumi, AWS 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.

Tools: Terraform, Pulumi, AWS Cloud Development Kit (CDK)

Diagram as Code is the creation of diagrams using code.

Tools: Diagrams Python lib, CloudGram, Ilograph, Mermaid

Infrastructure as Code vs Diagram as Code vs Architecture as Code

Infrastructure as Code (IaC) is the managing and provisioning of infrastructure through code.

Tools: Terraform, Pulumi, AWS Cloud Development Kit (CDK)

Diagram as Code is the creation of diagrams using code.

Tools: Diagrams Python lib, CloudGram, Ilograph, Mermaid

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

Tools: Diagrams Python lib

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

```

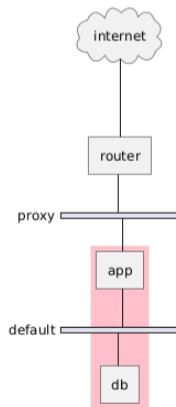
@startuml
nwdiag {
    internet [ shape = cloud];
    internet -- router;

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

```

https://www.plantuml.com/plantuml/png/LSrH3e8m30RWPtUAXdTEG4YuX_6XiEc4iJ3Pb3Hh3tPtArFNjN_VzxPQ84dNs9gnsn04U1jAC8Je9B18HbYk0wnPwJsFFJRckQn3I51hpNgItbMG25hhTNrtxv4yv8_CdR0MpxeBgumuFoFmZz1m1XjJ38tmCzUH9eeU8nJ5KscHTuCvqBLcV_10c
[Decode URL](#)

[Submit](#)
[Discover the future PlantUML Web Editor!](#)
[PNG](#)
[SVG](#)
[ASCII Art](#)





File ▾

Edit ▾

View ▾

Style ▾

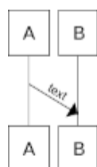
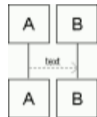
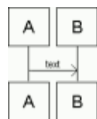
Help ▾



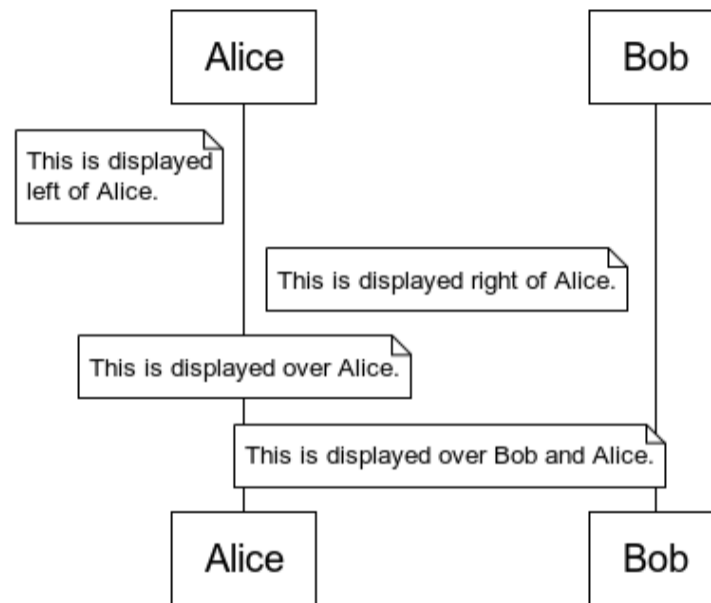
Year end sale



Sign in



```
1 participant Alice
2 participant Bob
3
4 note left of Alice
5 This is displayed
6 left of Alice.
7 end note
8 note right of Alice: This is displayed right of Alice.
9 note over Alice: This is displayed over Alice.
10 note over Alice, Bob: This is displayed over Bob and Alice.
```

www.websequencediagrams.com

Untitled ! (unsaved)



There are no errors in the syntax of the diagram.



Visualising software architecture with the C4 model

1-day workshop | Tuesday 3rd June | Antwerp, Belgium

Structurizr DSL documentation

🏠 Upload </> 📄 Render

```
1 workspace "Name" "Description"
2
3 !identifiers hierarchical
4
5 model {
6   u = person "User"
7   ss = softwareSystem "Software System" {
8     wa = container "Web Application" {
9       db = container "Database Schema" {
10        tags "Database"
11      }
12    }
13  }
14  u -> ss.wa "Uses"
15  ss.wa -> ss.db "Reads from and writes to"
16 }
17
18 views {
19   systemContext ss "Diagram1" {
20     include *
21     autolayout lr
22   }
23
24   container ss "Diagram2" {
25     include *
26     autolayout lr
27   }
28
29   styles {
30     element "Element" {
31       color white
32     }
33     element "Person" {
34       background #116611
35       shape person
36     }
37     element "Software System" {
38       background #2D882D
39     }
40     element "Container" {
41       background #55aa55
42     }
43     element "Database" {
44       shape cylinder
45     }
46   }
47 }
48
49 }
```

Structurizr DSL v4.0.0 - some features (e.g. `!docs`, `!script`, etc) are unavailable via this demo page; see [DSL](#) for details.

Examples: [Big Bank plc](#) | [Microservices](#) | [Amazon Web Services](#)

Structurizr UI

Graph

PlantUML

C4-PlantUML

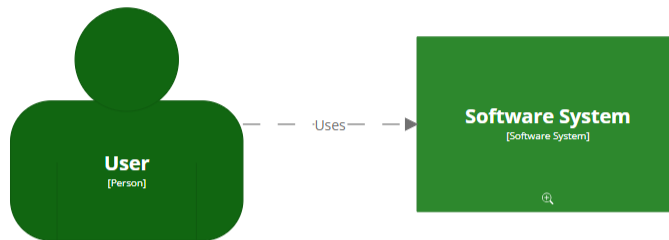
Mermaid

DOT

WebSequenceDiagrams

Illograph

[System Context] Software System (#Diagram1) ▾



[System Context] Software System

Tuesday, April 22, 2025 at 11:59 PM Central European Summer Time



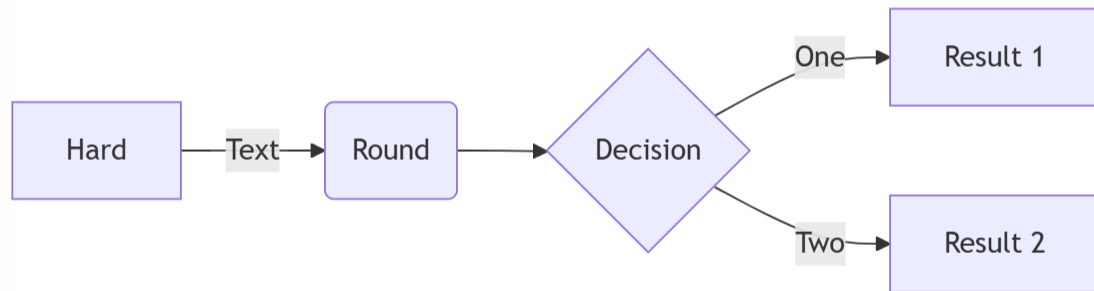
Mermaid ☒ Code Config Auto sync ☒ DOCS

```

1 flowchart LR
2   A[Hard] -->|Text| B(Round)
3   B --> C{Decision}
4   C -->|One| D[Result 1]
5   C -->|Two| E[Result 2]
6

```

Diagram Rough ☐ Pan & Zoom ☐ FULL SCREEN SAVE TO MERMAID CHART



```

Mermaid </> Code ⚙️ Config Auto sync ☒ 📄 DOCS
1 C4Context
2 title System Context diagram for Internet Banking System
3 Enterprise_Boundary(b0, "BankBoundary0") {
4   Person(customerA, "Banking Customer A", "A customer of the bank, with personal bank accounts")
5   Person(customerB, "Banking Customer B")
6   Person_Ext(customerC, "Banking Customer C", "desc")
7
8   Person(customerD, "Banking Customer D", "A customer of the bank, <br/> with personal bank accounts")
9
10  System(SystemAA, "Internet Banking System", "Allows customers to view information about their bank accounts, and make payments")
11
12  Enterprise_Boundary(b1, "BankBoundary") {
13
14    SystemDb_Ext(SystemE, "Mainframe Banking System", "Stores all of the core banking information about customers, accounts, transactions, etc.")
15
16    System_Boundary(b2, "BankBoundary2") {
17      System(SystemA, "Banking System A")
18      System(SystemB, "Banking System B", "A system of the bank, with personal bank accounts")
19    }
20
21    System_Ext(SystemC, "E-mail system", "The internal Microsoft Exchange e-mail system")
22    SystemDb(SystemD, "Banking System D Database", "A system of the bank, with personal bank accounts")
23
24    Boundary(b3, "BankBoundary3", "boundary") {
25      SystemQueue(SystemF, "Banking System F Queue", "A system of the bank.")
26      SystemQueue_Ext(SystemG, "Banking System G Queue", "A system of the bank, with personal bank accounts")
27    }
28  }
29
30  BiRel(customerA, SystemAA, "Uses")
31  BiRel(SystemAA, SystemE, "Uses")
32  Rel(SystemAA, SystemC, "Sends e-mails", "SMTP")
33  Rel(SystemC, customerA, "Sends e-mails to")
34
35  UpdateElementStyle(customerA, $fontColor="red", $bgColor="grey", $borderColor="red")
36  UpdateRelStyle(customerA, SystemAA, $textColor="blue", $lineColor="blue", $offsetX=-50, $offsetY=-50)
37  UpdateRelStyle(SystemAA, SystemE, $textColor="blue", $lineColor="blue", $offsetY=-50)
38  UpdateRelStyle(SystemAA, SystemC, $textColor="blue", $lineColor="blue", $offsetY=-50)
39  UpdateRelStyle(SystemC, customerA, $textColor="red", $lineColor="red", $offsetX=-50, $offsetY=-50)
40
41
42  UpdateLayoutConfig($c4ShapeInRow="3", $c4BoundaryInRow="1")
43

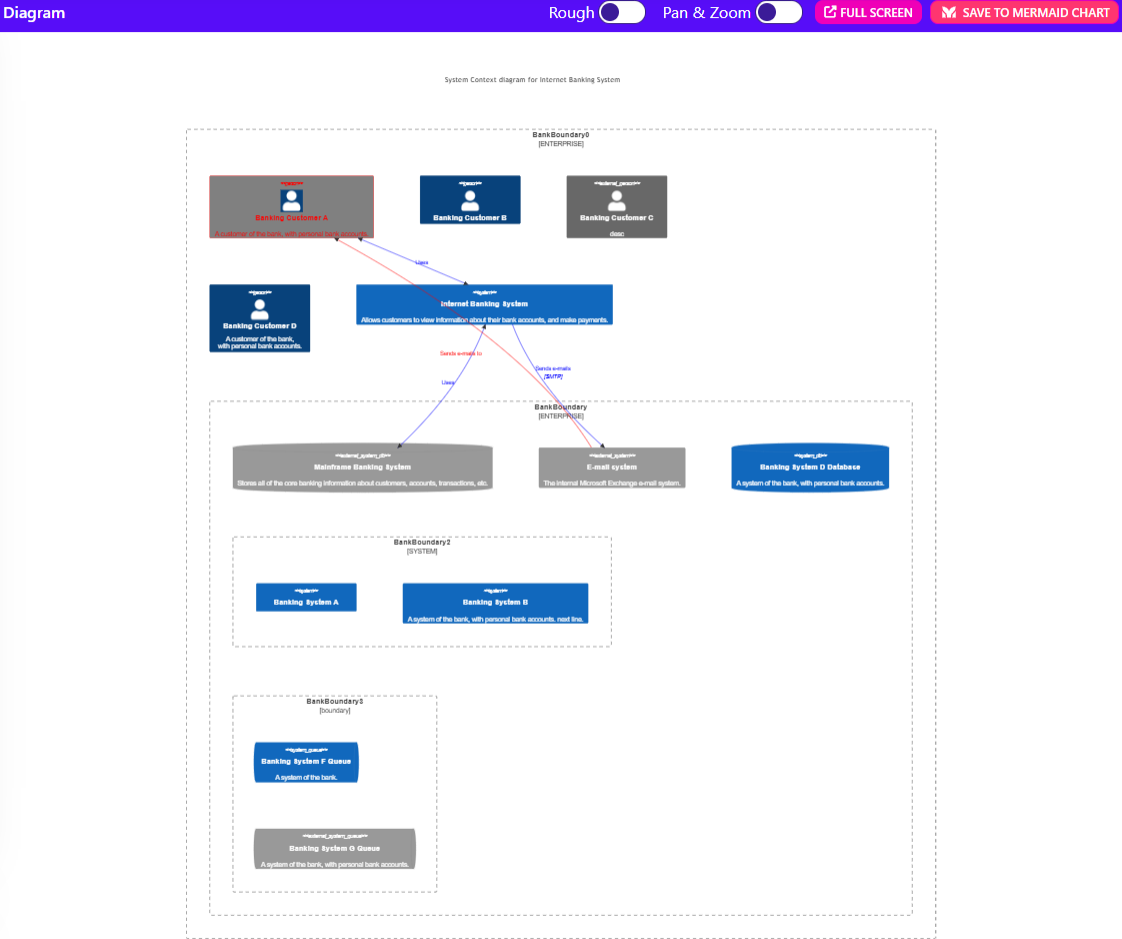
```

> Sample Diagrams

> History



> Actions

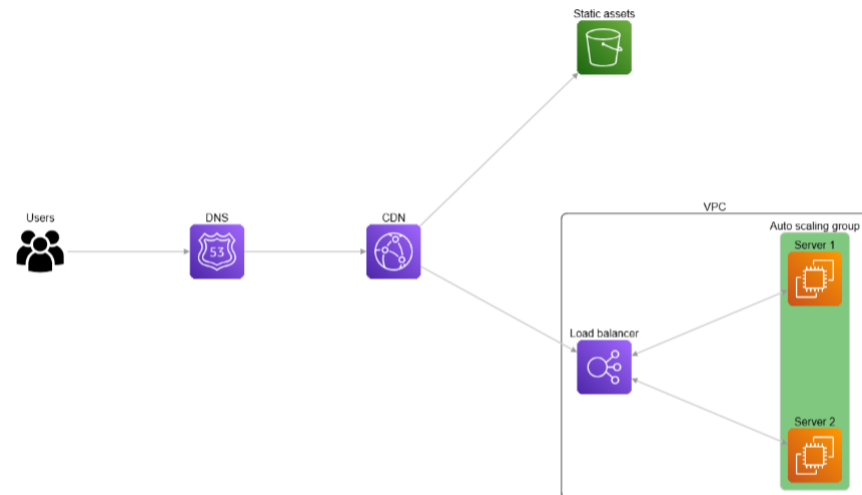




```

1 diagram "example" [direction=lr] {
2   // creating the nodes
3   generic.users Users;
4   aws.route53 DNS;
5   aws.cloudfront cf [label="CDN"];
6   aws.s3 s3 [label="Static assets"];
7
8   group vpc [label="VPC",style=solid,stroke=black,opacity=0] {
9     aws.elb load_balancer [label="Load balancer"];
10
11     group asg [label="Auto scaling group",style=solid,fill="#80c880"] {
12       aws.ec2 server1 [label="Server 1"];
13       aws.ec2 server2 [label="Server 2"];
14     }
15   }
16
17   // creating the edges
18   Users -> DNS -> cf -> load_balancer <=> asg;
19   cf -> s3;
20 }

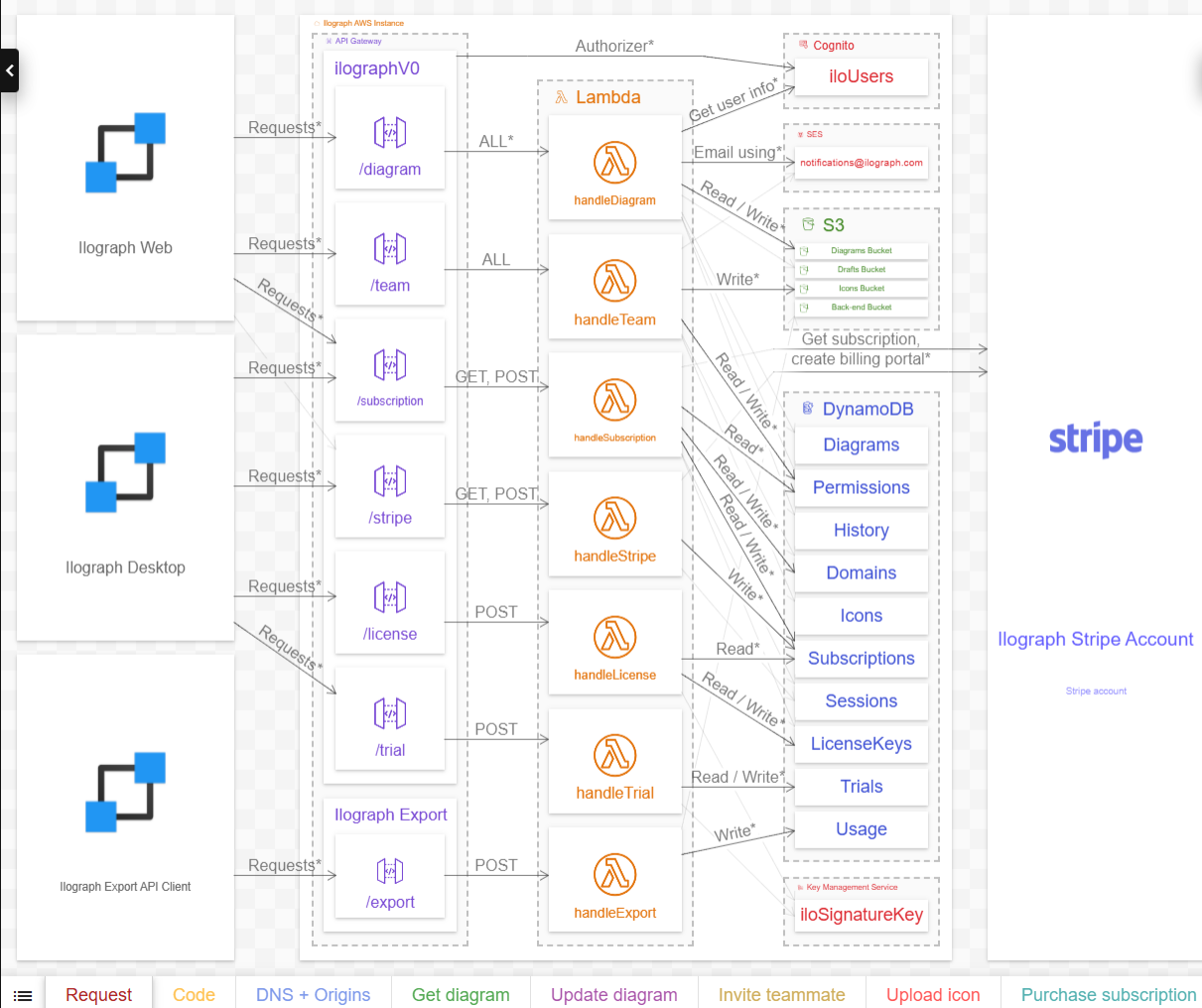
```



🔗

This diagram is read-only. [Click here](#) to create a new diagram or [log in](#) now.

```
1872 *confirmUser* Lambda function. This
1873 function is passed the confirmed
1874 user's email address and username.
1875
1876 - highlight: Permissions
1877 text: |-
1878     First, *confirmUser* checks if there
1879     are any pending permissions that
1880     need to be created for the new user
1881     . This occurs if the new user has
1882     been invited to view existing
1883     diagram(s) and/or join an existing
1884     team.
1885
1886     If any are found, the appropriate
1887     permission(s) are written to the
1888     *Permissions* table.
1889
1890 - highlight: notifications@ilograph.com
1891 text: |-
1892     Next, a welcome email is sent to the
1893     new user. This email contains links
1894     to Ilograph documentation and
1895     sample diagrams.
1896
1897 - highlight: Ilograph Users
1898 text: |-
1899     Lastly, if the user has signed up for
1900     Ilograph updates, their email
1901     address will be added to the
1902     *Ilograph Users* contact list on
1903     MailChimp.
1904
1905 - select: iloUsers
1906 highlight: Pre Token Generation
1907 text: |-
1908     Next, we'll look at the log-in
1909     trigger.
1910
1911 - select: generateToken
1912 text: |-
1913     *Pre Token Generation* calls the
1914     *generateToken* Lambda function.
1915     This function is passed the logging
1916     -in user's username.
1917
1918     This function generates the user's
1919     permissions by querying the
1920     *Permissions* and *Domains* tables.
1921
1922     Those permissions are converted to
1923     claims, signed by Cognito, and
1924     returned to the user for.
1925
1926 description: |-
1927     This Ilograph diagram documents the
1928     Ilograph Serverless back-end from eleven
1929     perspectives. Select a perspective below
1930     to get started.
1931
1932 *©2022 Ilograph LLC. Stripe, Namecheap,
1933 Mailchimp, AWS, and their icons are
1934 trademarks of their respective trademark
1935 holders.*
```



Start Walkthrough

Perspective notes

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

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 licer

DSL to Python trend

Infrastructure as Code (IaC):

Terraform --> Pulumi, AWS Cloud Development Kit (CDK)

Workflow as Code:

Oozie, Azkaban, Mistral --> Airflow, Luigi, Temporal

Architecture as Code (AaC):

Draw.io (diagrams.net) --> Diagrams Python lib

02

Problem statement

Do you use UML?

Do you use **MS Visio**?

Do you use **Draw.io**?

Do you use **whiteboard**?

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

03

Solution

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

Architecture as Code

pros:

- track architecture diagram changes with version control
- cloud agnostic (AWS, Azure, GCP, etc.)
- reuse of source code and collaboration
- better for tax deduction (e.g., in Poland)
- expect coding 🚀

cons:

- better for up-front design than long-lived documentation
- engineering background required
- whiteboard is better for prototyping

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



04

Diagrams
Python lib

master ▼

Public

3 Branches 46 Tags

Go to file

Add file ▾

<> Code ▾

- Diagram as Code for prototyping cloud system architectures

diagrams.mingrammer.com

graphviz

diagram

architecture

[diagram-as-code](#) [Readme](#)

 MIT license

 Activity

☆ 40.6k stars

👁 406 watching

2.6k forks

Report repository

Releases 44

v0.24.4 (Latest)

on Mar 10

+ 43 releases

Used by 1.9k



Contributors 166

shivamstaq Issue-1134: Renamed "AzureSpeedToText" to "AzureSpeechService" (#... 2d48685 · 2 days ago 623 Commits		
 .devcontainer	Update path to font file because existing path was giving an ...	2 months ago
 .github	add pre-commit to verify commit before create PR (#1066)	5 months ago
 assets/img	fix: resize logo	5 years ago
 diagrams	Issue-1134: Renamed "AzureSpeedToText" to "AzureSpeechS...	2 days ago
 docker/dev	chore(deps): bump python in /docker/dev (#1137)	2 days ago
 docs	Issue-1134: Renamed "AzureSpeedToText" to "AzureSpeechS...	2 days ago
 resources	Issue-1134: Renamed "AzureSpeedToText" to "AzureSpeechS...	2 days ago
 scripts	fix: Apply pre-commit hook changes (#1104)	2 months ago
 templates	Improve wording of documentation (#990)	last year
 tests	add pre-commit to verify commit before create PR (#1066)	5 months ago
 website	Issue-1134: Renamed "AzureSpeedToText" to "AzureSpeechS...	2 days ago
 .gitignore	Add programming languages and frameworks (#112)	5 years ago
 .isort.cfg	add pre-commit to verify commit before create PR (#1066)	5 months ago
 .pre-commit-config.yaml	add pre-commit to verify commit before create PR (#1066)	5 months ago
 CHANGELOG.md	add pre-commit to verify commit before create PR (#1066)	5 months ago
 CONTRIBUTING.md	Add new provider GIS (#847)	2 months ago

Diagrams Python lib

```
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")
```

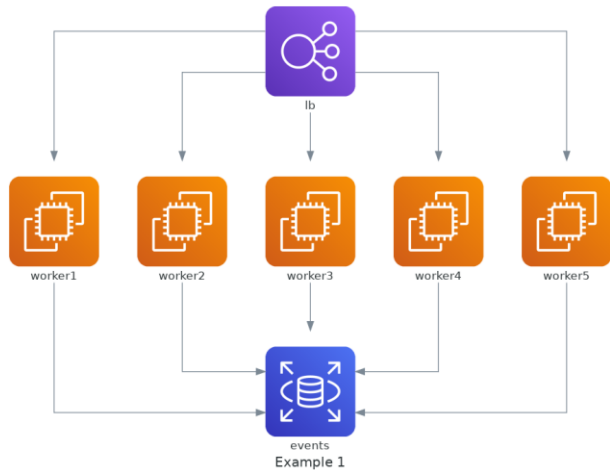


Diagram class

```
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")
```



web
Example 2

Nodes

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

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



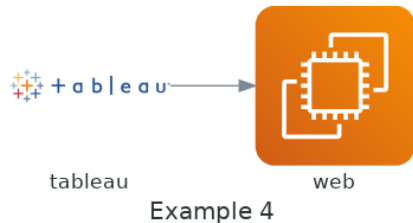
web
Example 3

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")
```

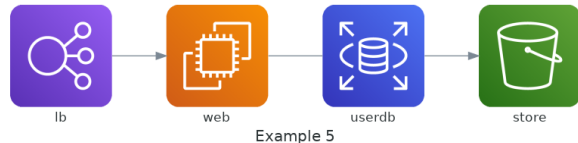


Data flow

```
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")
```

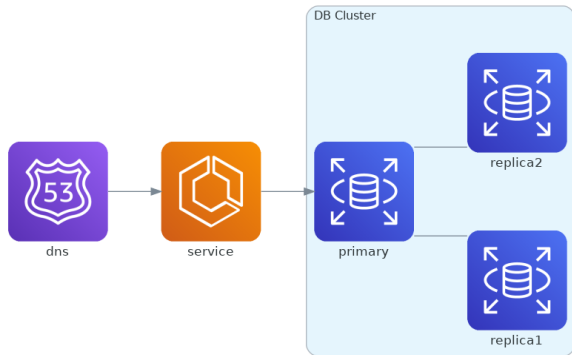
```
    web = ECS("service")
```

```
    with Cluster("DB Cluster"):
```

```
        db_primary = RDS("primary")
```

```
        db_primary - [RDS("replica1"),
                      RDS("replica2")]
```

```
    dns >> web >> db_primary
```



Example 6

```

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")

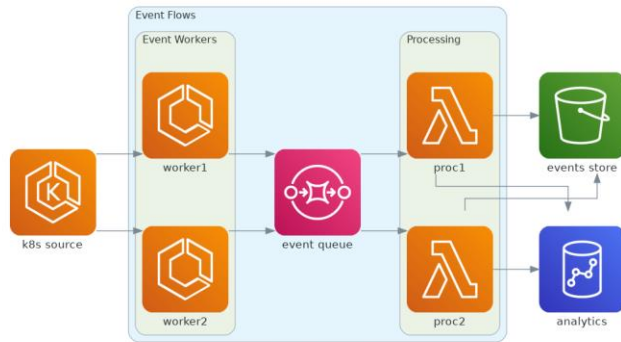
        with Cluster("Processing"):
            handlers = [Lambda("proc1"),
                      Lambda("proc2")]

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

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

```

Nested clusters



Example 7

05

Closing slides

 architecture-as-code-pycon-lithuania-2025 Public

[📌 Pin](#) [👁 Unwatch](#) 1 [🍴 Fork](#) 0 [★ Star](#) 0

[🔗 main](#) [🔗 1 Branch](#) [🏷 0 Tags](#)

🔍 Go to file

[Add file](#)

[<> Code](#)

About

Architecture as Code (AaC) with Python

- [📖 Readme](#)
- [📄 Unlicense license](#)
- [📈 Activity](#)
- [★ 0 stars](#)
- [👁 1 watching](#)
- [🍴 0 forks](#)

Releases

No releases published
[Create a new release](#)

Packages

No packages published
[Publish your first package](#)

Languages

 korniichuk Initial commit	54523c5 · 2 minutes ago	🕒 2 Commits
 .gitignore	Initial commit	11 minutes ago
 LICENSE	Initial commit	11 minutes ago
 README.md	Initial commit	2 minutes ago
 d1_example.py	Initial commit	2 minutes ago
 d2_diagram_class.py	Initial commit	2 minutes ago
 d3_nodes.py	Initial commit	2 minutes ago
 d4_custom_nodes.py	Initial commit	2 minutes ago
 d5_data_flow.py	Initial commit	2 minutes ago
 d6_clusters.py	Initial commit	2 minutes ago
 d7_nested_clusters.py	Initial commit	2 minutes ago
 example2.dot	Initial commit	2 minutes ago



RUSLAN KORNIICHUK

Software and Platform Engineer in Poland



Hire me

Solid experience of 10+ years in IT. Former Software Engineer and Architect at Fortune 500 companies. Core areas: Cloud Computing, and Software Engineering. Especially interested in Platform Engineering. Click the button above to hire me.



DONATE TO UKRAINE ARMY FUNDRAISER

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 components of the Defense Forces.

Payment method

PAYMENT BY CARD

SWIFT TRANSFERS

CRYPTOCURRENCY

Regularity

ONE TIME

SUBSCRIPTION EVEN BETTER

Payment currency

UAH, ₴

USD, \$

EUR, €

Amount of donation*

€ 10

Email*

example@example.com

☐ I consent to the processing of my email address to receive email newsletters (twice a month), invitations and receipts.

PAY

 Payment secured

tinyurl.com/PYCONLT1

