# Benefits of Using the Diagrams Package In Python

A sophisticated approach to creating diagrams

Bhavika Sewpal

August 15, 2022

# Motivation

- Faster than methods that involve manually inserting and editing images
- Automatic alignment of icons and arrows
- Consistency of icons
- Easy to make updates
- Easy to monitor changes in diagrams
- Can be included as part of a project in a github repository as python code

# Purpose

- Diagrams let you draw the cloud system architecture in Python code
- Diagrams currently supports main major providers including:
  - AWS
  - Azure
  - GCP
  - Kubernetes
  - Saas

# Requirements

- Diagrams require Python 3.6 or higher
- It uses Graphviz - an open source graph visualization software - to render the diagrams

# Concepts

- Diagrams has 4 concepts:
  - Diagrams
  - Nodes
  - Clusters
  - Edges

# Diagrams

- Diagram represents a global diagram context
- A diagram context is created with the Diagram class

```
from diagrams import Diagram
from diagrams.k8s.compute import Pod

with Diagram("Simple Diagram", show=False):
    Pod("pod instance")
```

# Diagrams (cont. )


pod instance

Simple Diagram

# Nodes

- A node represents a single system component
- A node consists of three parts:
  - a provider
  - a resource type
  - a name

```
from diagrams.k8s.compute import Pod
```

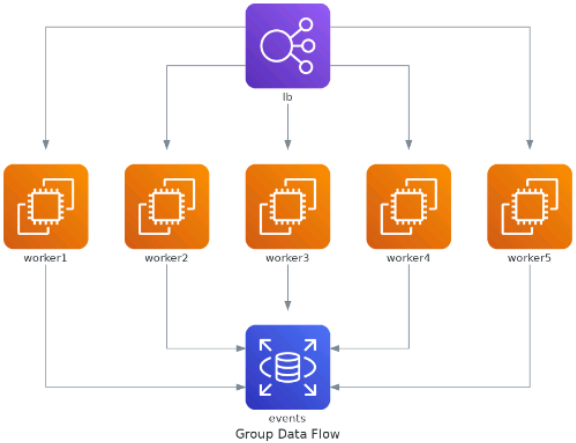
In the above example, k8s is the provider, compute is the resource type and Pod is the name

# Nodes - Data Flow

- `>>` : Connects node in left to right direction
- `<<` : Connects node in right to left direction
- `:` : Undirected

```
from diagrams import Diagram
from diagrams.aws.compute import EC2
from diagrams.aws.database import RDS
from diagrams.aws.network import ELB
with Diagram("Group Data Flow", show=False, direction="TB"):
    ELB("lb") >> [EC2("worker1"),
                  EC2("worker2"),
                  EC2("worker3"),
                  EC2("worker4"),
                  EC2("worker5")] >> RDS("events")
```

# Nodes - Data Flow (cont.)

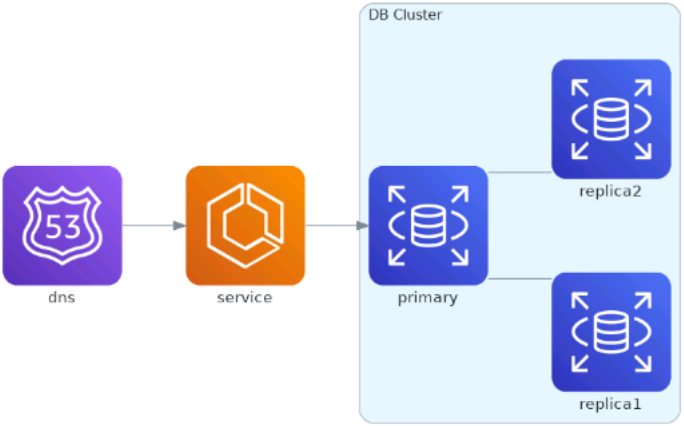


# Clusters

- Cluster allows you to group nodes in an isolated group
- Clusters can be nested as well

```
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("Simple Web Service with DB Cluster", 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
```

# Clusters (cont.)




Simple Web Service with DB Cluster

# Edges

- An edge represents a linkage between nodes with some additional properties
- An edge object contains three attributes:
  - label
  - color
  - style (example: dashed, dotted, bold)

# Simplified Diagram for BlobCSI System

- Diagram showing the links between Kubeflow Notebooks, PVCs, PVs and Azure Containers
- blobcsi_kubeflow_pvc_pv_azure.py
- This diagram illustrates that:
  - clusters can be nested
  - nodes can be joined across clusters
  - the edges can be labelled and formatted

# Simplified Diagram for BlobCSI System (cont.)

# Summary

- Diagrams as code in python demonstrated to be an easy, practical and useful tool
- Storing diagrams as code improves comprehension of complex CI/CD cloud deployments
- Diff comparisons of diagram as code makes change management easier
- Leverages python skills – just learn python markup methods but easier than mastering a separate model or GUI tool

# Questions?