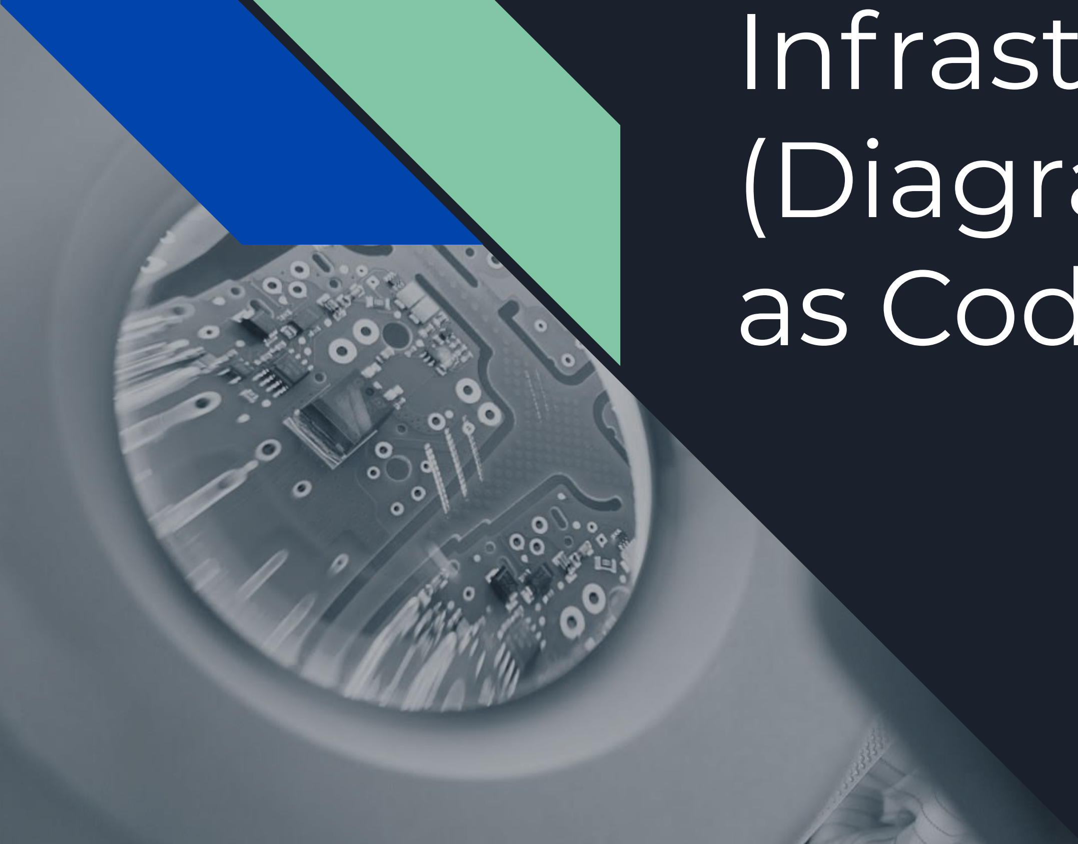


# Infrastructure (Diagrams!) as Code

CLEpy MotM talk - November 2020

Diagrams Module

# Intro / WHOAMI

David Egbert

Full Stack Developer (Python / React / AWS)
Work at REPAY (Remote - even before it was mandatory)

Stints at OnShift and Explorys/IBM Watson Health

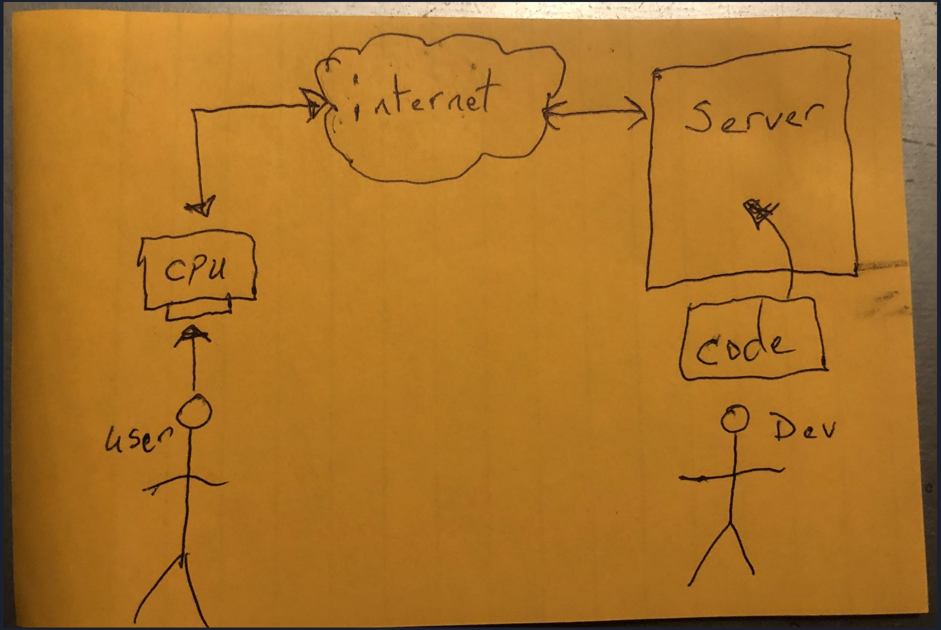
Bootcamp Grad - (RIP Software Guild - Akron)

# Can you point me to the documentation for… ?

- UX/UI - Wire Frames
- General Info - READMEs, Confluence, etc
- APIs - Swagger, etc.
- Application Design - Visio, LucidChart
- Database Entity-Relationship Diagrams
- **Architecture Diagrams**

# Architecture Diagrams (Before Cloud Days)

A slight simplification:



# One Way to Explain Cloud Architecture

“You're a f$@#$%@ rock star, OK? You just don't know cloud, this tiny, little, s*&%#y area, which is becoming super important, and in many ways is the future of computing… That sort of went south on me, but you understand what I'm saying.”

- Erlich, Silicon Valley, Season 1, Episode 6

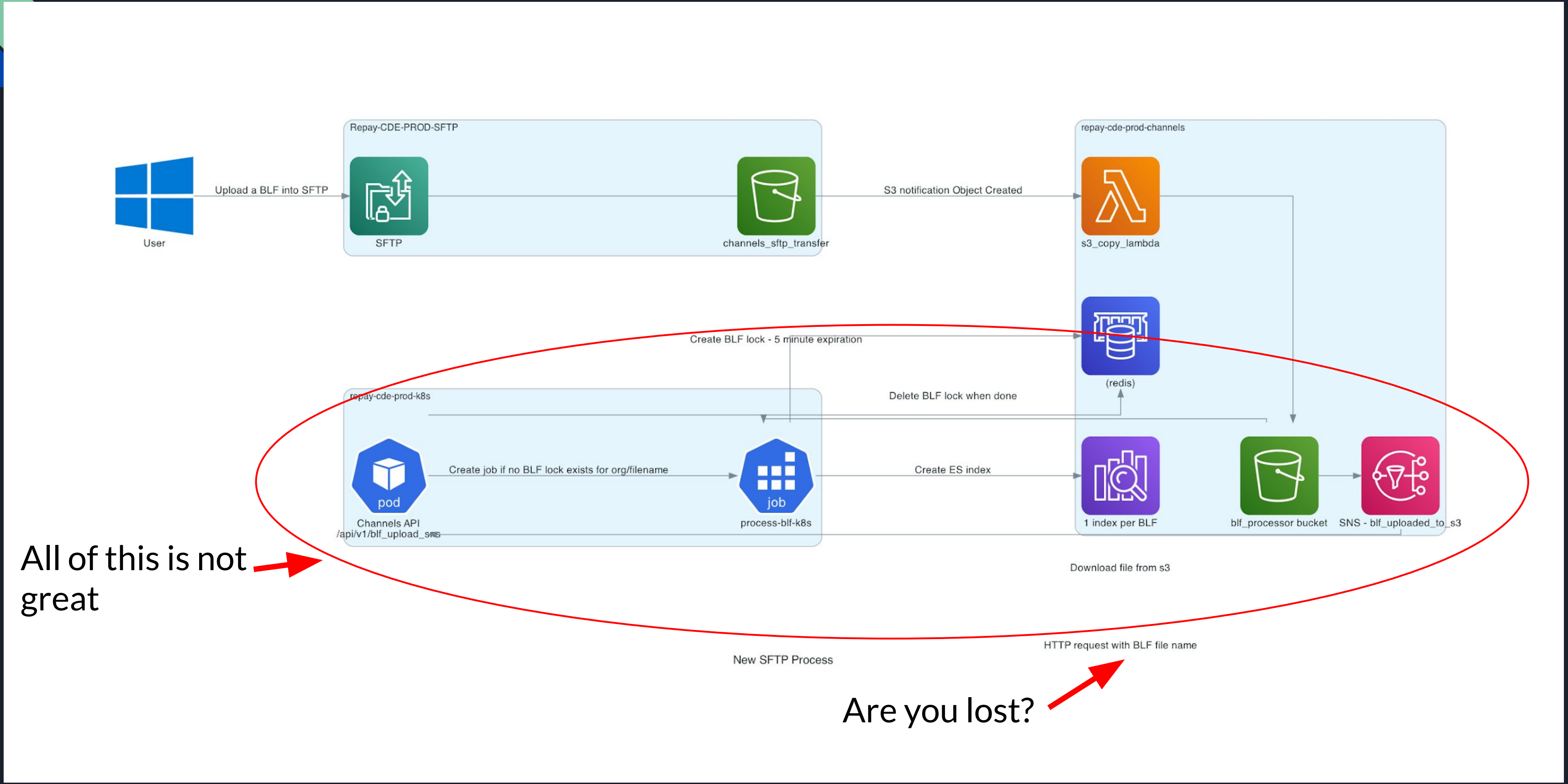
# Don’t be Erlich - Share the knowledge

01 Use GUI tools like Visio that are usually used for flow charts.

02 Use diagrams module - Create architecture drawings through python code.

03 Use mermaid - Create diagrams through markdown*ish* syntax.

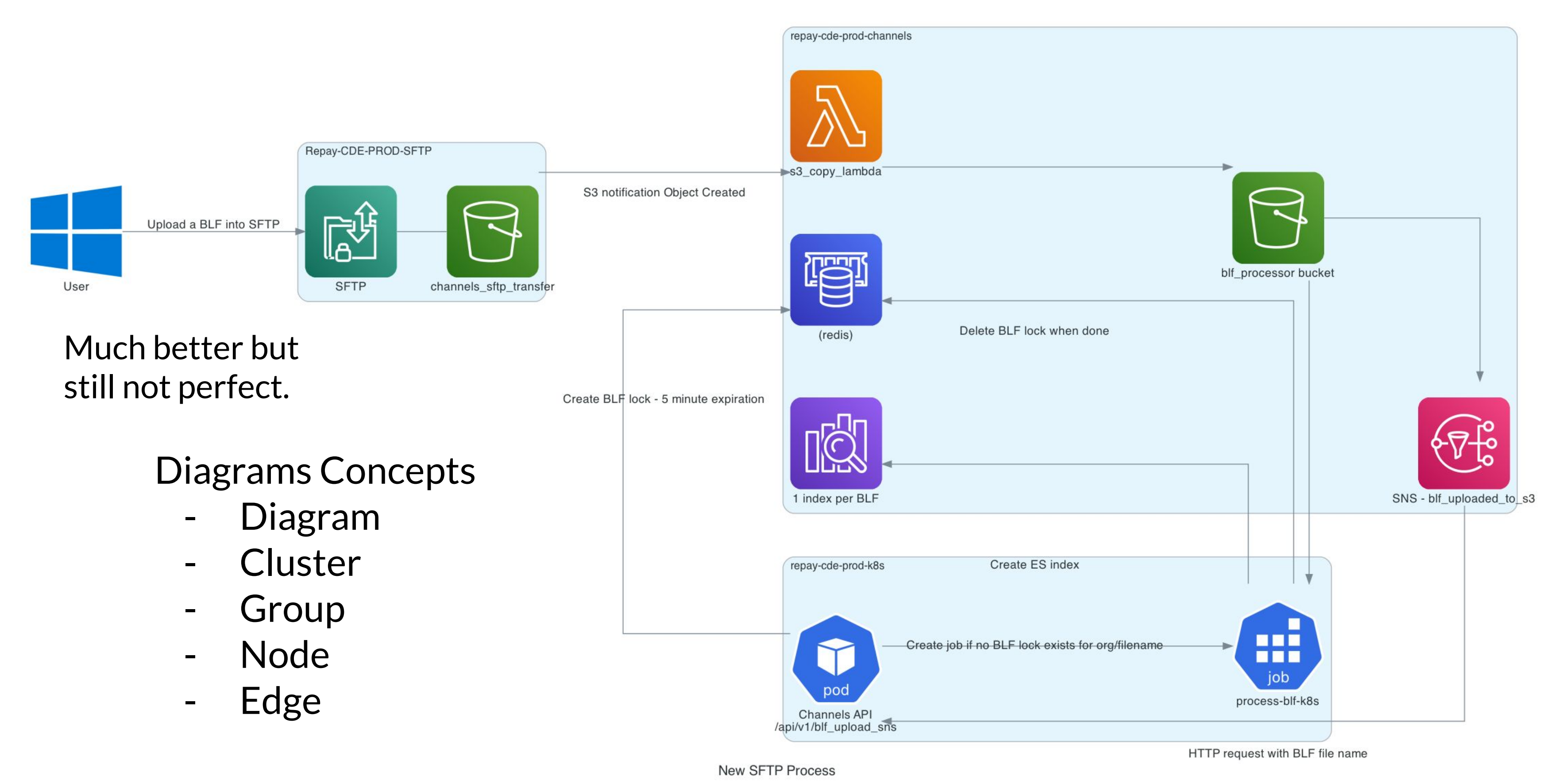
# Diagrams Example 1 - Take One



All of this is not great

Are you lost?

# Diagrams Ex 1 - Take Two - INVISIBLE EDGES!

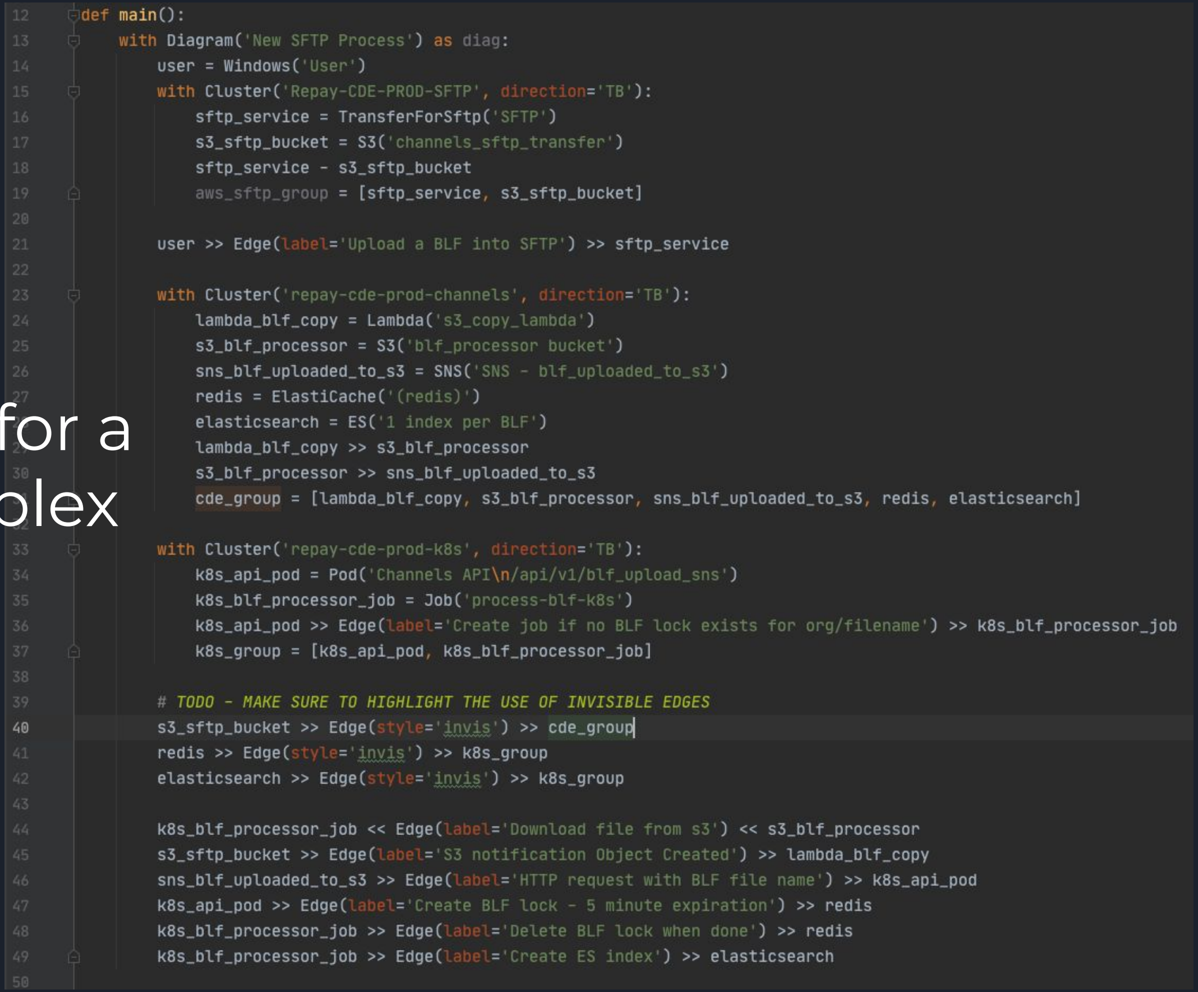


Much better but
still not perfect.

Diagrams Concepts

- Diagram
- Cluster
- Group
- Node
- Edge

# Code!
# ~ 40 lines for a fairly complex system



```
def main():
    with Diagram('New SFTP Process') as diag:
        user = Windows('User')
        with Cluster('Repay-CDE-PROD-SFTP', direction='TB'):
            sftp_service = TransferForSftp('SFTP')
            s3_sftp_bucket = S3('channels_sftp_transfer')
            sftp_service - s3_sftp_bucket
            aws_sftp_group = [sftp_service, s3_sftp_bucket]

        user >> Edge(label='Upload a BLF into SFTP') >> sftp_service

        with Cluster('repay-cde-prod-channels', direction='TB'):
            lambda_blf_copy = Lambda('s3_copy_lambda')
            s3_blf_processor = S3('blf_processor bucket')
            sns_blf_uploaded_to_s3 = SNS('SNS - blf_uploaded_to_s3')
            redis = ElastiCache('(redis)')
            elasticsearch = ES('1 index per BLF')
            lambda_blf_copy >> s3_blf_processor
            s3_blf_processor >> sns_blf_uploaded_to_s3
            cde_group = [lambda_blf_copy, s3_blf_processor, sns_blf_uploaded_to_s3, redis, elasticsearch]

        with Cluster('repay-cde-prod-k8s', direction='TB'):
            k8s_api_pod = Pod('Channels API\n/api/v1/blf_upload_sns')
            k8s_blf_processor_job = Job('process-blf-k8s')
            k8s_api_pod >> Edge(label='Create job if no BLF lock exists for org/filename') >> k8s_blf_processor_job
            k8s_group = [k8s_api_pod, k8s_blf_processor_job]

        # TODO - MAKE SURE TO HIGHLIGHT THE USE OF INVISIBLE EDGES
        s3_sftp_bucket >> Edge(style='invis') >> cde_group
        redis >> Edge(style='invis') >> k8s_group
        elasticsearch >> Edge(style='invis') >> k8s_group

        k8s_blf_processor_job << Edge(label='Download file from s3') << s3_blf_processor
        s3_sftp_bucket >> Edge(label='S3 notification Object Created') >> lambda_blf_copy
        sns_blf_uploaded_to_s3 >> Edge(label='HTTP request with BLF file name') >> k8s_api_pod
        k8s_api_pod >> Edge(label='Create BLF lock - 5 minute expiration') >> redis
        k8s_blf_processor_job >> Edge(label='Delete BLF lock when done') >> redis
        k8s_blf_processor_job >> Edge(label='Create ES index') >> elasticsearch
```

# Diagrams module - The Good

- PREREQ - MUST HAVE GRAPHVIZ INSTALLED
- Diagrams as code - version control, update as things change, etc.
- Good integration with Jupyter Notebooks for your data sciency friends.
- Graphics look great / Lots of providers
- Quick learning curve (if you know python)

# Diagrams module - The Bad and Ugly

- Trial and Error Needed to get things to look right

- Rendering is kinda magic - hard to predict where nodes end up. LTR and TB didn’t work as expected (YMMV)

- Python isn’t a universal language (sadly)

- Graphviz and Python 3.6 could be barriers.

# Mermaid

Very Flexible

Markdown Syntax

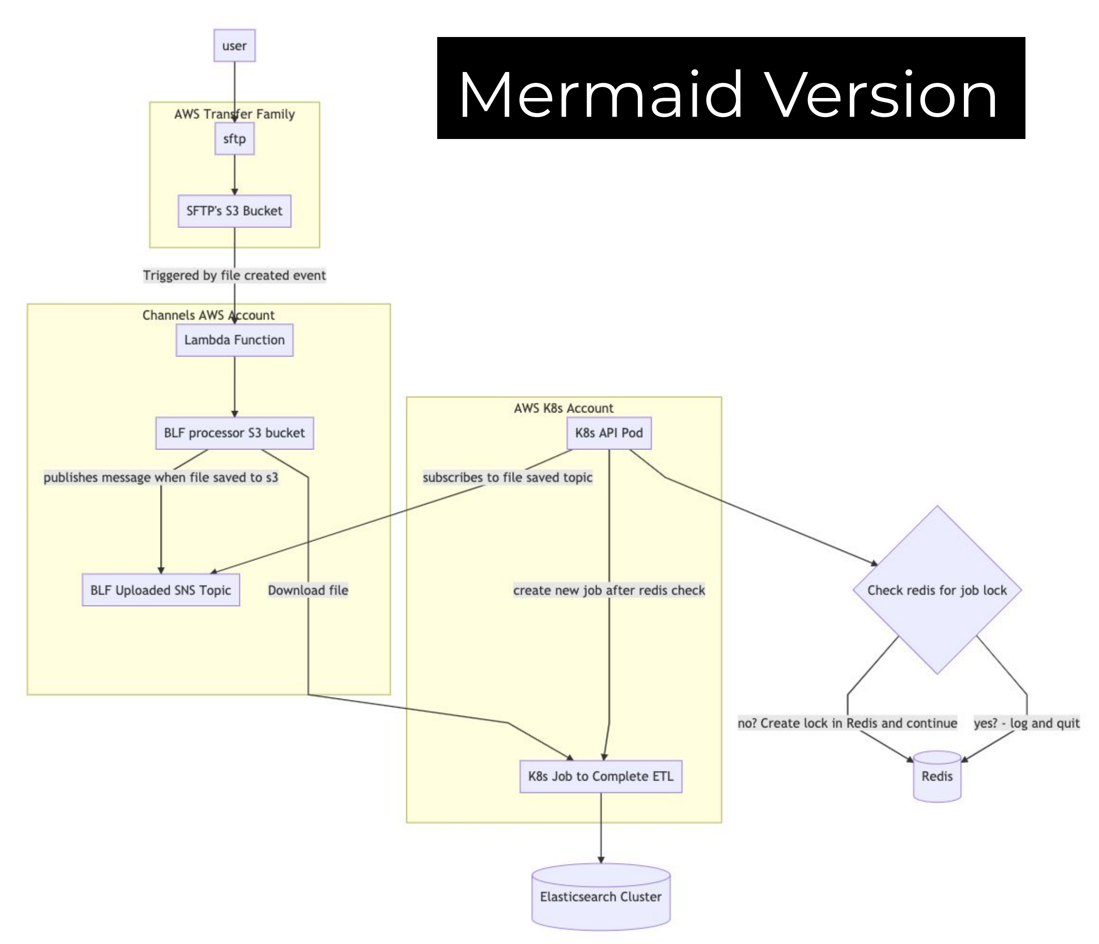
Javascript-based (It’s inescapable - even at a python meetup, sorry)

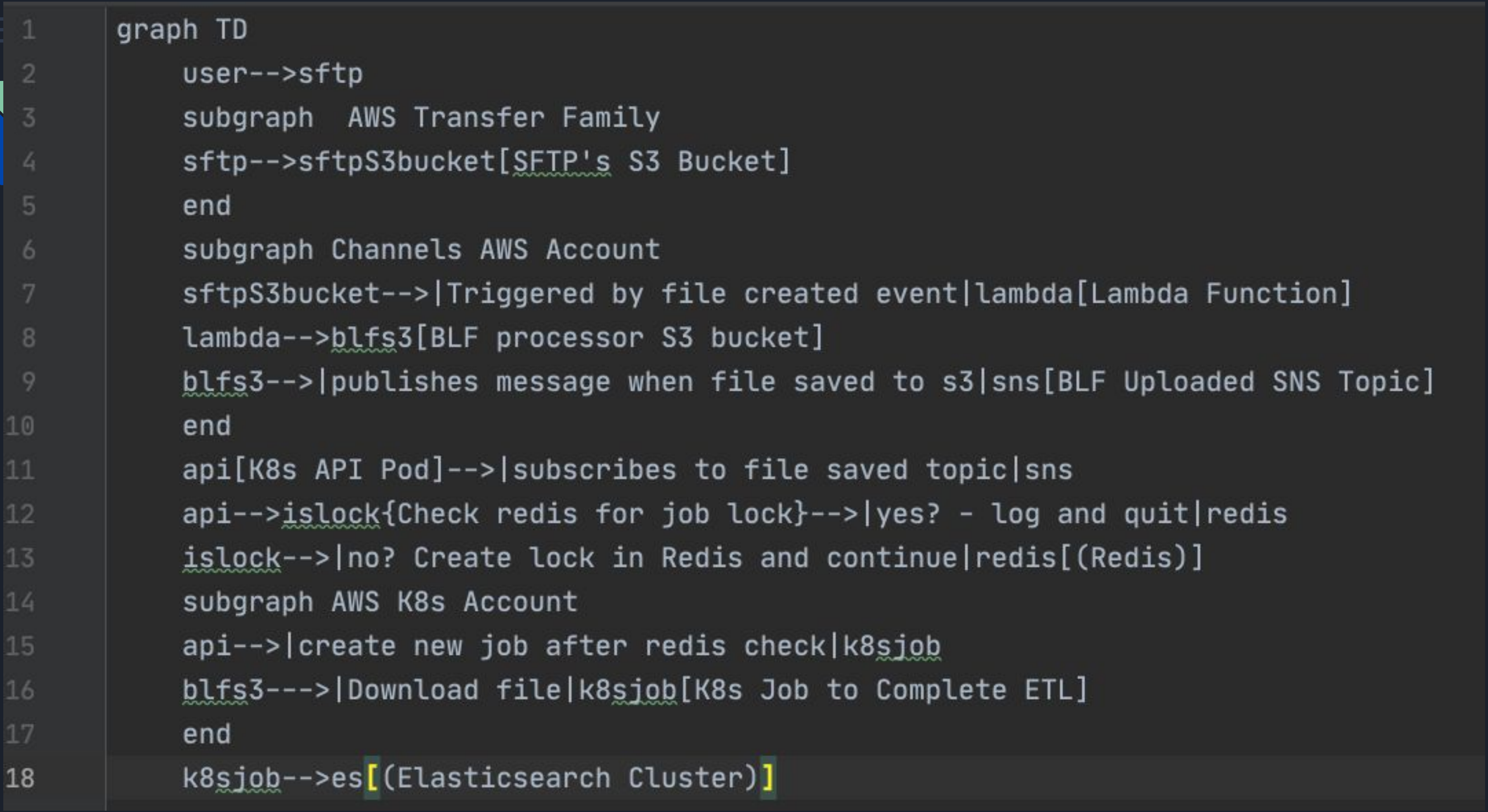
Many installation options and a live editor:

https://mermaid-js.github.io/mermaid-live-editor

https://mermaid-js.github.io/mermaid/

# Mermaid Version





```
graph TD
    user-->sftp
    subgraph  AWS Transfer Family
    sftp-->sftpS3bucket[SFTP's S3 Bucket]
    end
    subgraph Channels AWS Account
    sftpS3bucket-->|Triggered by file created event|lambda[Lambda Function]
    lambda-->blfs3[BLF processor S3 bucket]
    blfs3-->|publishes message when file saved to s3|sns[BLF Uploaded SNS Topic]
    end
    api[K8s API Pod]-->|subscribes to file saved topic|sns
    api-->islock{Check redis for job lock}-->|yes? - log and quit|redis
    islock-->|no? Create lock in Redis and continue|redis[(Redis)]
    subgraph AWS K8s Account
    api-->|create new job after redis check|k8sjob
    blfs3--->|Download file|k8sjob[K8s Job to Complete ETL]
    end
    k8sjob-->es[(Elasticsearch Cluster)]
```

# Mermaid - Good, Bad, and the ugly

- Very easy to learn / universal / non-devs
- Few lines of code / less magical than diagrams
- Lots of integrations
- Fairly generic looking compared to diagram
- More general than just architecture
- Can still be difficult to get lines where you want them

# Parting Words and Q and A

Use what works for you and your team(s)!

Questions???
Thanks for having me!!!

Code and Slides available at:
github.com/CLEpy/CLEpy-MotM

Me:
DMEgbert@gmail.com
@david_egbert on cleveland-tech slack