

# LEAP Life - Computer Literacy – January 6, 2026

## Day 2



Department of Electrical & Computer Engineering



# What does Object Oriented Programming Mean to You?



# What are the benefits of OOP?

- **Encapsulation**
  - Briefly talked about this best practice...
    - Prior to OOP, data could be encapsulated
    - OOP enables encapsulation of the data as well as its supported behaviors
  - This is particularly beneficial for larger code bases as it makes them more
    - modular
    - readable
    - maintainable
    - scalable

- **classes** are templates that define state (variables) and methods (functions) that are tightly coupled together.
- **instances** are specific objects created from the **class** template that have their own state based on the **class** definition.
  - Your program may involve very many instances of a particular class, just as it may involve many **integer** or **string** variables.
- A lot more core concepts to come but let's start on an example that demonstrates these two most core ones

```
class ExamScore:  
    student_id: str  
    exam_name: str  
    score: float
```



three pieces of information  
**encapsulated** together

```
class ExamScore:  
    student_id: str  
    exam_name: str  
    score: float
```

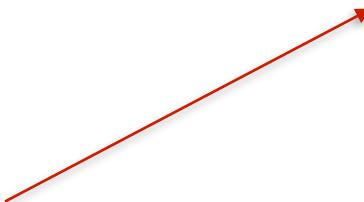
creation of an **instance** of the **ExamScore** class

```
expected = ExamScore(  
    student_id="U123",  
    exam_name="midterm",  
    score=75.4  
)
```

```
@dataclass
class ExamStats:
    average_final: float
    unique_students: int

    def to_dictionary(self):
        return {
            "average_final": self.average_final,
            "unique_students": self.unique_students
        }
```

**behavior** (converting to a dictionary)  
encapsulated together with state



```
@dataclass
class ExamStats:
    average_final: float
    unique_students: int

    def to_dictionary(self):
        return {
            "average_final": self.average_final,
            "unique_students": self.unique_students
        }
```

invocation of a method (behavior)  
on an instance of the **ExamStats**  
class

```
result = read_and_compute(
    LocalCSVDataFetcher(INPUT_FILENAME),
    ExamDataProcessor()
)

with open(OUTPUT_FILENAME, "w") as out:
    json.dump(result.to_dictionary(), out, indent=2)
```

# What does the term Interface mean to you?



```
from dataclasses import dataclass

@dataclass
class ExamScore:
    student_id: str
    exam_name: str
    score: float

@dataclass
class ExamStats:
    average_final: float
    unique_students: int

    def to_dictionary(self):
        return {
            "average_final": self.average_final,
            "unique_students": self.unique_students
        }

class DataFetcher:
    def fetch(self) -> [ExamScore]:
        raise NotImplementedError

class DataProcessor:
    def compute_average_final(self, scores: [ExamScore]) -> float:
        raise NotImplementedError

    def compute_number_of_unique_students(self, scores: [ExamScore]) -> int:
        raise NotImplementedError
```

contracts.py in Base repo

# Day 2 Hands-on Exercise 1

- Re-factor your code to use classes
- Re-factor your code to implement interfaces
- Update your tests to cover your updated structure
- Make sure your code still works
  - From base repo, copy the following files:
    - main.py
    - contracts.py
    - exam\_data\_processor.py
    - local\_csv\_data\_fetcher.py
    - test\_exam\_data\_processor.py
    - test\_local\_csv\_data\_fetcher.py
- **Too make everything work, you should only have to change:**
  - exam\_data\_processor.py
  - local\_csv\_data\_fetcher.py
- You can disable your old tests if needed



## Day 2 Hands-on Exercise 1 - What did you learn?



# What does the term Cloud Computing mean to you?



# What is Cloud Computing?

Cloud computing is the delivery of on-demand computing resources, including **storage**, **processing power**, and **applications**, over the internet, allowing users to access and manage data and services **remotely without owning the physical infrastructure**.

# Definition & History

1950 -1960

- mainframe  
computers were very  
expensive and  
resource intensive so  
engineers started  
thinking about time  
sharing techniques  
for them



# Definition & History

1970 -1990

- IBM introduced VM  
(virtual machine)  
operating system as  
a way to allow  
multiple, separate  
environments to be  
executed on the  
same physical  
machine



# Definition & History

1990 -2010

- Majority of companies rent/own data centers, or space in data centers, and manage their physical infrastructure there
- Deploying new services or scaling existing ones requires months of planning overhead
  - procure machines
  - install machines in data center
  - bring them on-line etc...



# Definition & History

1990 -2000

- Software as a Service starts to gain popularity.  
Companies like **Salesforce** pioneer large scale, enterprise software delivery with zero installed software.



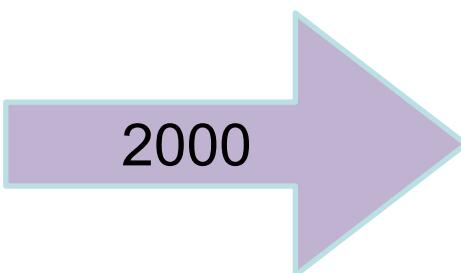
Software as a Service



Department of Electrical & Computer Engineering



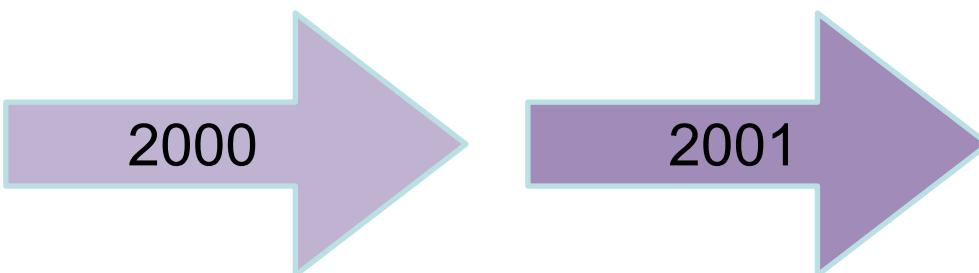
# Definition & History



2000

- Amazon dominates  
on-line sales of  
physical books

# Definition & History

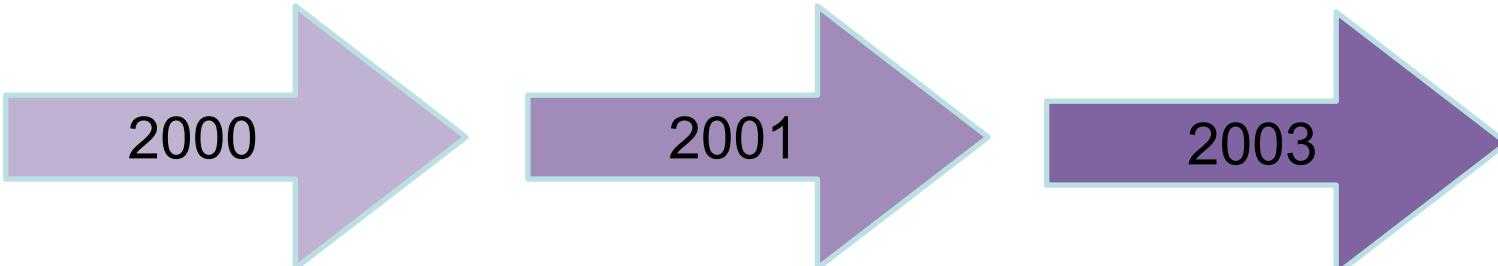


2000

2001

- Amazon dominates on-line sales of physical books
- Amazon opens up their virtual store platform to other retailers

## Definition & History



2000

2001

2003

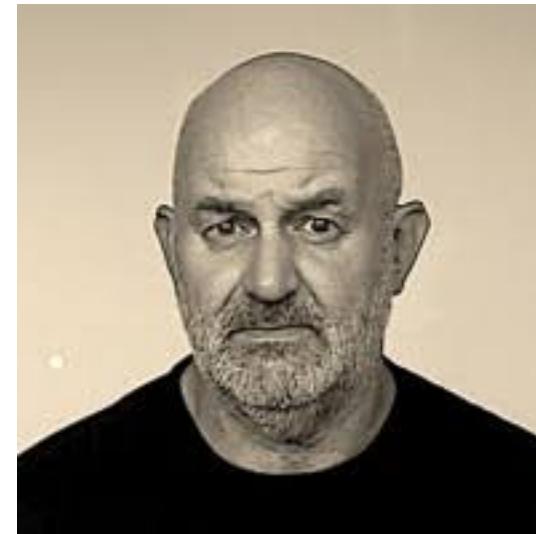
- Amazon dominates on-line sales of physical books
- Amazon opens up their virtual store platform to other retailers
- Group of Amazon engineers realize that Amazon purchases infrastructure to meet their spike loads, but during regular load only ~ 10% of their infrastructure is utilized.
- What day of the year would a much higher fraction of their infrastructure get utilized on?

## Definition & History

2006



- Amazon releases Simple Storage Service (S3). Consumers can access scalable, cheap file-like storage via API's.

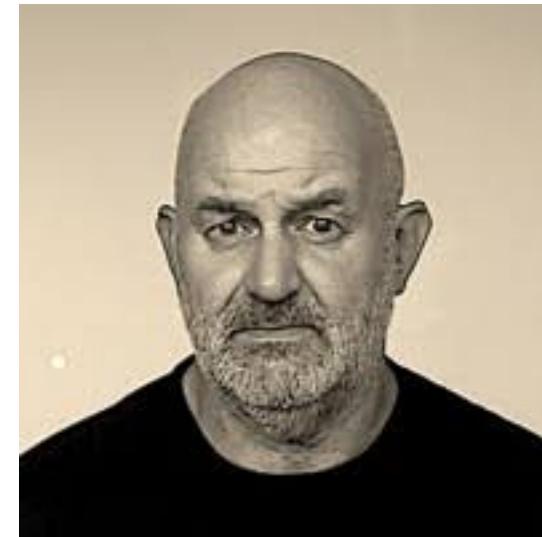


# Definition & History

2006



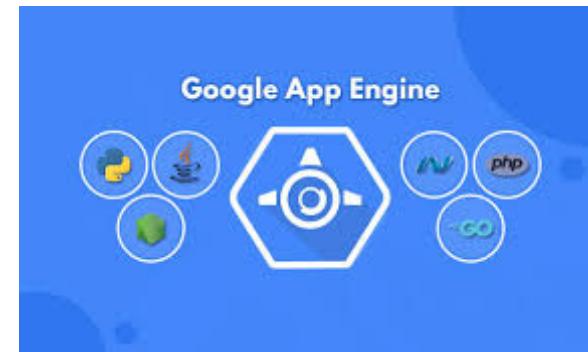
- Amazon releases Elastic Compute Cloud (EC2) - one of the first “cloud computing” offerings. Consumers could rent virtual machines in Amazon for \$0.10 / hour.



# Definition & History

2008-2010

- Amazon innovation catches on and first competitors to Amazon emerge



## Definition & History

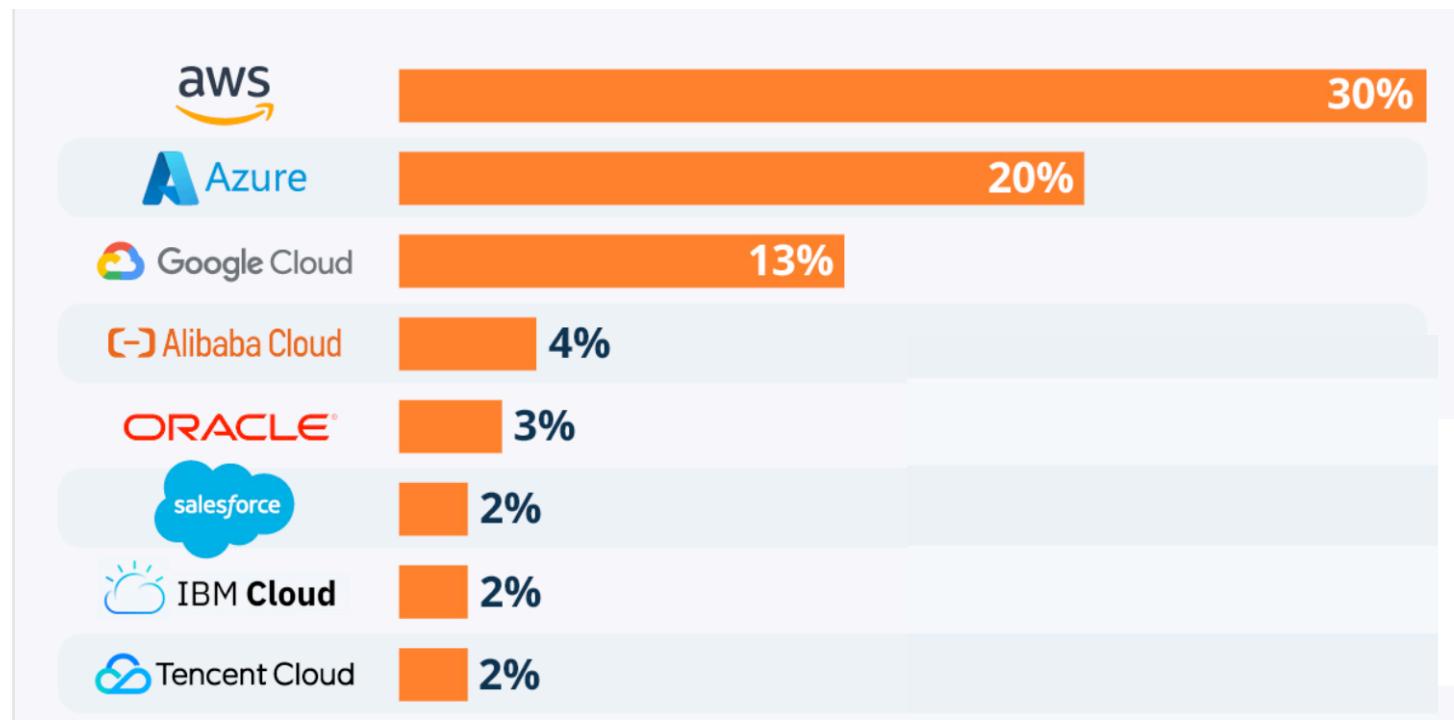


today

- In 2008, Amazon's competitors were in their infancy and Amazon had virtually 100% of the market share.
- **What do you think the market share is today?**

# Definition & History

today



# Definition & History



today

- In 2008, global cloud computing revenue was ~ **\$6 billion**.
- **What do you think global cloud computing revenue is today?**

# Definition & History



today

- Depends on how you measure it exactly but its somewhere in the \$700 - \$900 billion range. **100x from 2008**

# Lambda



- Serverless computing service
- You write code and define events that should trigger that code
- Code can be in one of ten or so supported languages
- Examples of triggers:
  - file uploaded
  - HTTP request made
  - Event appears in the queue
  - database row inserted

## Day 2 Hands-on Exercise 2

- Re-factor to prep for cloud
  - create three folders where all of your code should go:
    - common
    - local
    - aws
- for AWS Lambda, wrap exam score results in HTML
  - by getting **lambda\_app.py** from **base**
  - We will configure AWS Lambda to call **lambda\_handler** in **lambda\_app.py**
- can get S3 data reading by getting **s3\_data\_fetcher.py** from **base**



2

# Setting Up Your Lambda Functions

The screenshot shows the AWS Lambda Functions page. On the left, there's a sidebar with 'Lambda' selected. The main area has a blue header bar with the text 'Introducing the AWS Serverless generative AI webinar series' and a 'See more details' button. Below this, it says 'Functions (1)'. A search bar is present. The table lists one function:

Function name	Description	Package type	Runtime	Type	Last modified
[Function Name]	[Description]	[Package type]	[Runtime]	[Type]	[Last modified]

At the bottom right of the table, there are buttons for 'Actions' and 'Create function'.



# Setting Up Your Lambda Functions

**Create function** [Info](#)

Choose one of the following options to create your function.

**Author from scratch**  
Start with a simple Hello World example.

**Use a blueprint**  
Build a Lambda application from sample code and configuration presets for common use cases.

**Container image**  
Select a container image to deploy for your function.

**Basic information**

**Function name**  
Enter a name that describes the purpose of your function.

Function name must be 1 to 64 characters, must be unique to the Region, and can't include spaces. Valid characters are a-z, A-Z, 0-9, hyphens (-), and underscores (\_).

**Runtime** | [Info](#)  
Choose the language to use to write your function. Note that the console code editor supports only Node.js, Python, and Ruby.  
 🕒 Last fetched 1/5/2026, 11:01:51 PM

**Durable execution - new** | [Info](#)  
Enable durable execution to simplify building resilient multi-step applications that checkpoint progress and resume after interruptions. Supports Python and Node.js runtimes. [View pricing](#).  
 Enable

**Architecture** | [Info](#)  
Choose the instruction set architecture you want for your function code.  
 arm64  
 x86\_64

**Permissions** [Info](#)  
By default, Lambda will create an execution role with permissions to upload logs to Amazon CloudWatch Logs. You can customize this default role later when adding triggers.

**▼ Change default execution role**

**Execution role**  
Choose a role that defines the permissions of your function. To create a custom role, go to the [IAM console](#).  
 Create a new role with basic Lambda permissions  
 Use an existing role  
 Create a new role from AWS policy templates

**Existing role**  
Choose an existing role that you've created to be used with this Lambda function. The role must have permission to upload logs to Amazon CloudWatch Logs.  
 🕒 Last fetched 1/5/2026, 11:01:51 PM  
[View the lambda-execution-role role](#) on the IAM console.



# To create zip file that you can upload to AWS Lambda

```
zip -r lambda.zip aws common -x "*/__pycache__/*" "*.pyc"  
".DS_Store"
```

can copy this from README of our **base** repo



# Upload your zip file to AWS Lambda ...

Successfully created the function **bustest-function2**. You can now change its code and configuration. To invoke your function with a test event, choose "Test".

## bustest-function2

**Function overview** [Info](#)

[Diagram](#) | [Template](#)

**bustest-function2**

**Layers** (0)

[+ Add trigger](#) [+ Add destination](#)

**Description**  
-

**Last modified**  
2 minutes ago

**Function ARN**  
 arn:aws:lambda:us-east-2:879381238148:function:bustest-function2

**Function URL** | [Info](#)  
-

[Code](#) | [Test](#) | [Monitor](#) | [Configuration](#) | [Aliases](#) | [Versions](#)

**Code source** [Info](#)

[Open in Visual Studio Code](#) [Upload from](#)

[.zip file](#) [Amazon S3 location](#)

EXPLORER

BUTEST-FUNCTION2

lambda\_function.py

```
lambda_function.py
1 import json
2
3 def lambda_handler(event, context):
4     # TODO implement
5     return {
6         'statusCode': 200,
7         'body': json.dumps('Hello from Lambda!')
8     }
9
```



Department of Electrical & Computer Engineering



# Edit Runtime Setting and set Handler to lambda\_app.lambda\_handler

The screenshot shows the 'Runtime settings' section of the AWS Lambda function configuration. It includes fields for 'Runtime' (Python 3.14), 'Handler' (lambda\_function.lambda\_handler), and 'Architecture' (x86\_64). There are 'Edit' and 'Edit runtime management configuration' buttons at the top right.

**Runtime settings** [Info](#)

**Runtime**  
Python 3.14

**Handler** [Info](#)  
lambda\_function.lambda\_handler

**Architecture** [Info](#)  
x86\_64

[Edit](#) [Edit runtime management configuration](#)

▶ Runtime management configuration

# Edit Configuration to Extend Timeout to 10 seconds

General configuration

- Triggers
- Permissions
- Destinations
- Function URL
- Environment variables
- Tags
- VPC

**General configuration** [Info](#) [Edit](#)

Description	Memory	Ephemeral storage
-	128 MB	512 MB
Timeout	SnapStart	
0 min 3 sec	Info	
	None	



Department of Electrical & Computer Engineering



# Test Your Function

**Code source** [Info](#)

AWS Lambda code editor based on Code-OSS  
(VS Code open source)

**BUTEST-FUNCTION2**

- contracts.py
- exam\_data\_processor.py
- lambda\_app.py
- main.py**
- s3\_data\_fetcher.py

**DEPLOY** Current

**Deploy (F5)**

**Test (Shift+F5)**

**TEST EVENTS [SELECTED: TESTONE]**

- + Create new test event
- Private saved events
- testone

**ENVIRONMENT VARIABLES**

✓ Lambda Deployed    0 ▲ 0    ▶ Amazon Q

**lambda\_app.py**

```
22 def lambda_handler(event, context):
```

**main.py**

```
1  from contracts import DataFetcher
2  from contracts import DataProcessor
3  from contracts import ExamStats
4  from local_csv_data_fetcher import LocalCSVDataFetcher
5  from exam_data_processor import ExamDataProcessor
6
7  import os
8  import json
9
10 def read_and_compute(data_fetcher: DataFetcher, data_processor: DataProcessor):
11     data = data_fetcher.fetch()
12     return ExamStats(
13         average_final= data_processor.compute_average_final(data),
14         unique_students= data_processor.compute_number_of_unique_students(data)
15     )
16
17 INPUT_FILENAME = "test_scores.csv"
18 OUTPUT_FILENAME = "output.json"
19
20 if os.path.exists(OUTPUT_FILENAME):
```

**PROBLEMS** **OUTPUT** **CODE REFERENCE LOG** **TERMINAL**

Function Logs:

```
START RequestId: 0921d979-324c-4996-a9c6-b10e36fe1f2c Version: $LATEST
Starting lambda!
Fetching from S3!
ExamStats(average_final=0, unique_students=0)
END RequestId: 0921d979-324c-4996-a9c6-b10e36fe1f2c
REPORT RequestId: 0921d979-324c-4996-a9c6-b10e36fe1f2c Duration: 689.93 ms Billed Duration: 690 ms Memory Size: 128 MB Max Memory Used: 97 MB
```

Request ID: 0921d979-324c-4996-a9c6-b10e36fe1f2c

Ln 1, Col 1   Spaces: 4   UTF-8   LF   Python    Lambda   Layout: U.S.  



# Create a Function URL for your Function with Auth = None

Code | Test | Monitor | **Configuration** | Aliases | Versions

General configuration  
Triggers  
Permissions  
Destinations  
**Function URL**  
Environment variables  
Tags  
VPC  
RDS databases  
Monitoring and operations tools

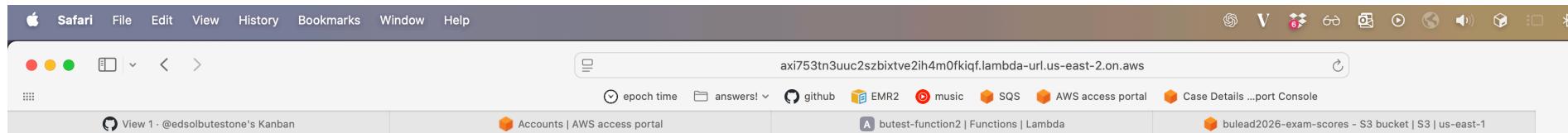
**Function URL** [Info](#) [Delete](#) [Edit](#)

ⓘ Your function URL is public. Anyone with the URL can access your function.

<b>Function URL</b> <a href="https://axi753tn3uuc2szbixtve2ih4m0fkqf.lambda-url.us-east-2.on.aws/">https://axi753tn3uuc2szbixtve2ih4m0fkqf.lambda-url.us-east-2.on.aws/</a>	<b>Auth type</b> NONE	<b>Invoke mode</b> <a href="#">Info</a> BUFFERED
<b>Creation time</b> 49 seconds ago	<b>Last modified</b> 49 seconds ago	
CORS (Not enabled)		



# Go to your Function URL to see it in Action!



## Student Scores!

- Average score: 0
- Unique students: 0



Department of Electrical & Computer Engineering



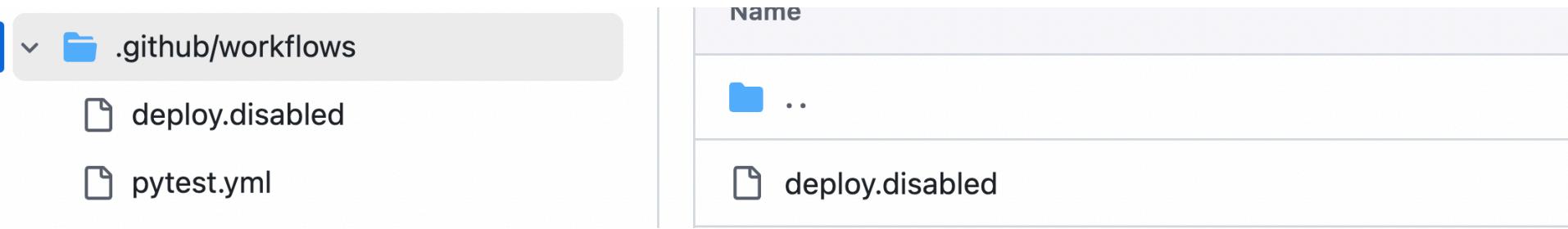
## Day 2 Hands-on Exercise 3

- Setting up your AWS accounts
- Set up your Lambda Function and manually alter its code
- Invoke your function via a web browser, confirm behavior, tweak the code and refresh to confirm changes reflected



3

# In your repos - rename deploy.disabled to deploy.yml



# deploy.yml

```

name: CD (Deploy to Lambda)

on:
  workflow_run:
    workflows: [ "Python tests (pytest)"]
    types: [completed]
    branches: [main]

jobs:
  deploy:
    if: ${{ github.event.workflow_run.conclusion == 'success' }}
    runs-on: ubuntu-latest

    permissions:
      contents: read
      id-token: write

    steps:
      - name: Check out code at the commit that passed CI
        uses: actions/checkout@v4
        with:
          ref: ${{ github.event.workflow_run.head_sha }}

      - name: Configure AWS credentials (OIDC)
        uses: aws-actions/configure-aws-credentials@v4
        with:
          role-to-assume: arn:aws:iam::879381238148:role/github-actions-lambda-deployer
          aws-region: us-east-2 # MAKE SURE REGION IS CORRECT HERE

      - name: Build Lambda zip
        run: |
          rm -f lambda.zip
          zip lambda.zip contracts.py lambda_app.py s3_data_fetcher.py exam_data_processor.py

      - name: Deploy to Lambda
        run: |
          aws lambda update-function-code \
            --function-name butest-function2 \ # UPDATE YOURN FUNCTION NAME HERE
            --zip-file fileb://lambda.zip

      - name: Wait for update to complete
        run: |
          aws lambda wait function-updated --function-name butest-function2 # UPDATE YOURN FUNCTION NAME HERE

```



## Day 2 Hands-on Exercise 4

- Set up Continuous Deployment of your Lambda function code via GitHub Actions
- Validate that you can make changes, have the code deployed automatically, and confirm behavior by invoking your function via a web browser



4

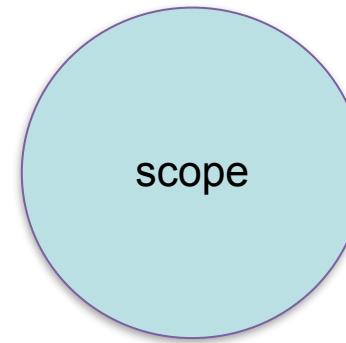
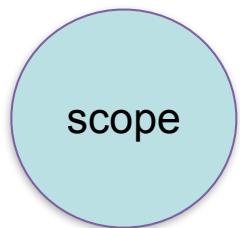
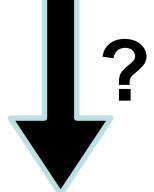
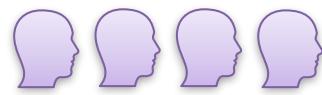
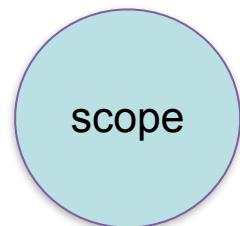
# Iron Triangle

The iron triangle of project management

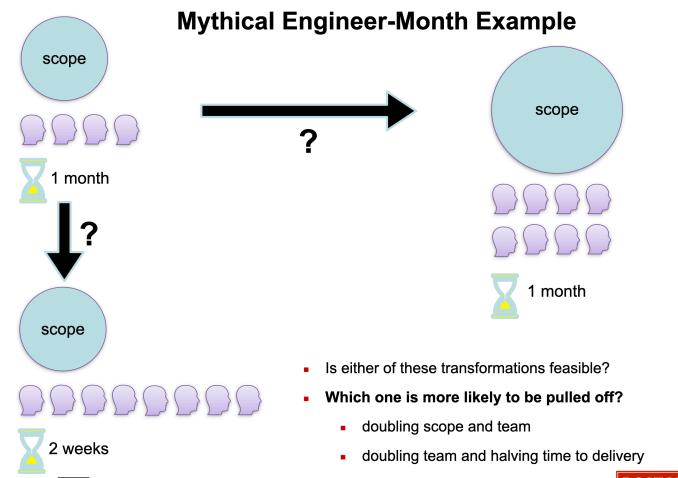


- No one wants to ship low quality software.
- We assume that the quality bar is very high and we are not willing to sacrifice on it.
- The other three dimensions play a game of tug of war:
  - Increasing Scope requires an increment of Time or Cost
  - Reducing Cost requires a reduction in Scope or increment of Time.
  - Reducing Time requires reducing Scope or an increment of Cost

# Mythical Engineer-Month Example



- Is either of these transformations feasible?
- **Which one is more likely to be pulled off?**
  - doubling scope and team
  - doubling team and halving time to delivery



- Fred Brook's classic 1975 book **The Mythical Man-Month: Essays on Software Engineering**.
- The central idea is that adding more people to a late software project often makes it later, rather than speeding it up.
- Software engineering should not be thought of as linear work.
- **Communication overhead is exponential with the the number of people involved on a project.**

# Roles in a Waterfall World

project planning



requirements & analysis



design



coding



testing



deployment



CEO



product manager



UX researcher



designer



developer



quality assurance



release engineer

# Agile

## Agile Development

is *adaptive* rather than predictive  
is *people-oriented* rather than process-oriented



Martin Fowler



Department of Electrical & Computer Engineering

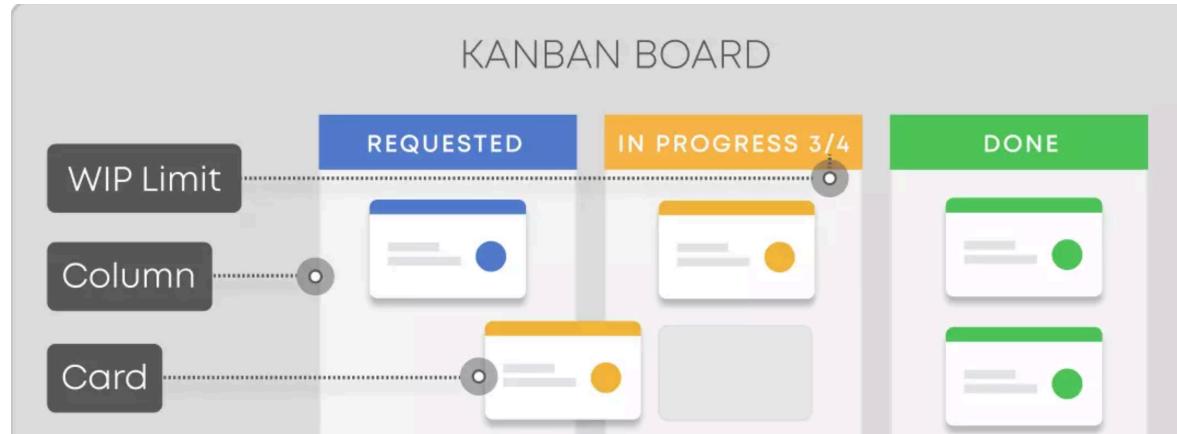


# Agile

- There are multiple methodologies and processes that are considered **Agile**, we will dig into a few of them, but the main distinction from its predecessor methodologies:
  - In **Plan-driven** processes, e.g. Waterfall, success is measured according to how well development follows the detailed, predictive plan.
  - In **Agile** processes, careful plans are still made, but are thought to be in constant revision to reflect learnings and a quickly evolving ecosystem.
  - In **Agile** processes, success is measured based on value delivered by the software.



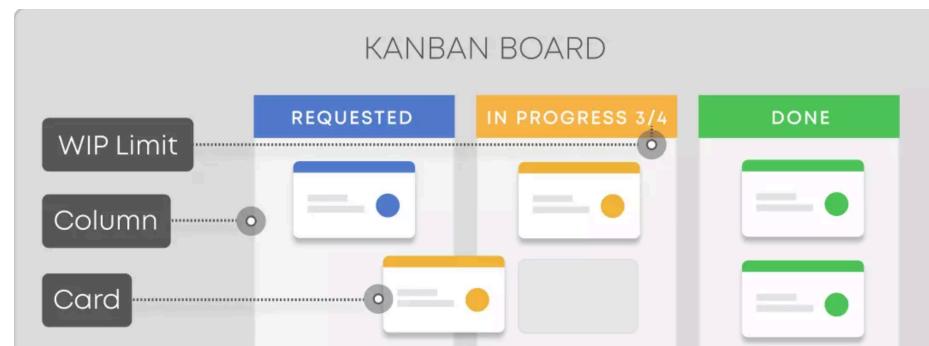
# Kanban

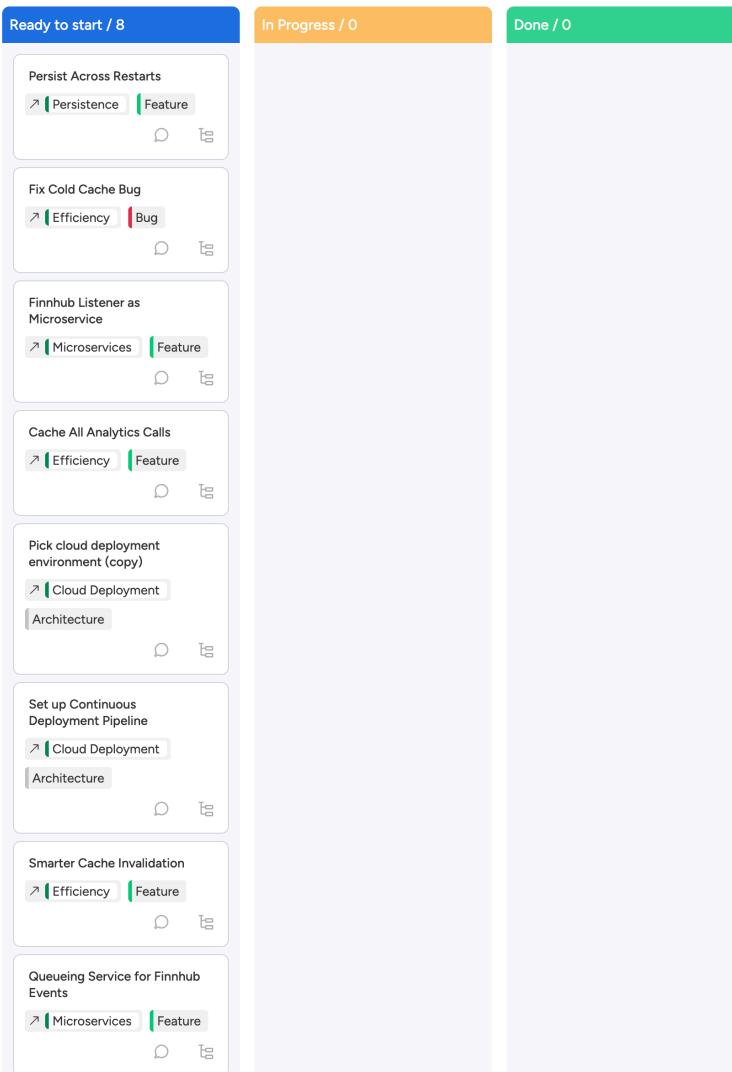


- Originated in manufacturing by Toyota in the 1940's
- Inspired by supermarket inventory practices of visualizing workflow and limiting Work In Progress (WIP)
- Physical Card System Core to the philosophy
- Popularity in software processes rose in the early 2000's
- Integrated into Agile philosophies

# Kanban Tenets

- Visualize all work
- Limit Work in Progress
- Keep Increments Small
- Manage flow, not individuals
- Create Feedback Loops
- Make Process Policies Explicit via Working Agreements
- Improve Collaboratively
- Celebrate Small Wins

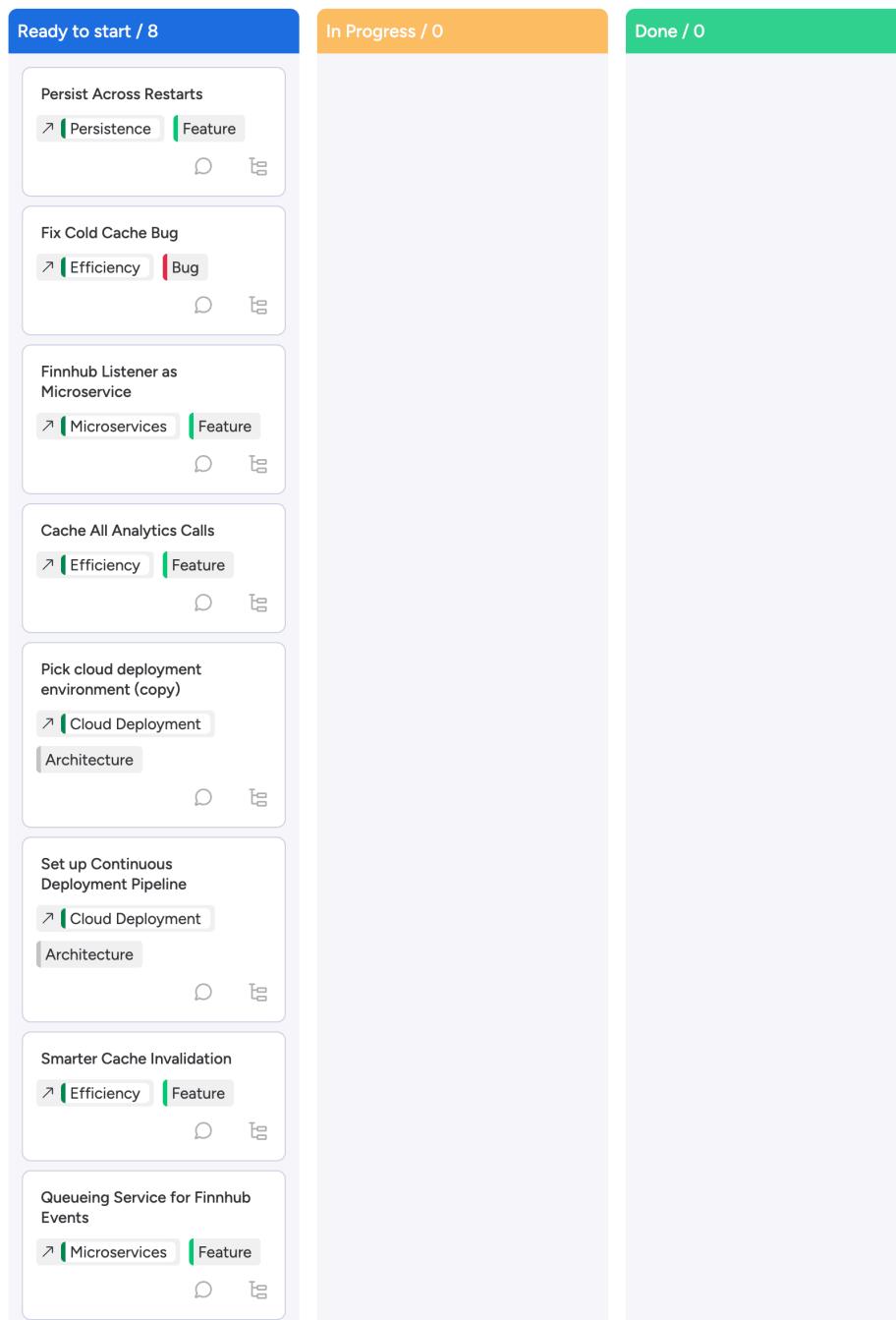




# Kanban Board

- Only three states that we care deeply about:
  - What work is prioritized and imminently ready for us?
  - What are we working on currently?
  - What did we recently finish?
- Distinguish
  - Bugs
  - Features





Shar



Larry



Jane



Jarred



Donna



Department

- team of 5 developers
- Mentioned WIP (work in progress) limit earlier.
- **What is a reasonable limit for a team of 5?**





Shar



Larry



Jane



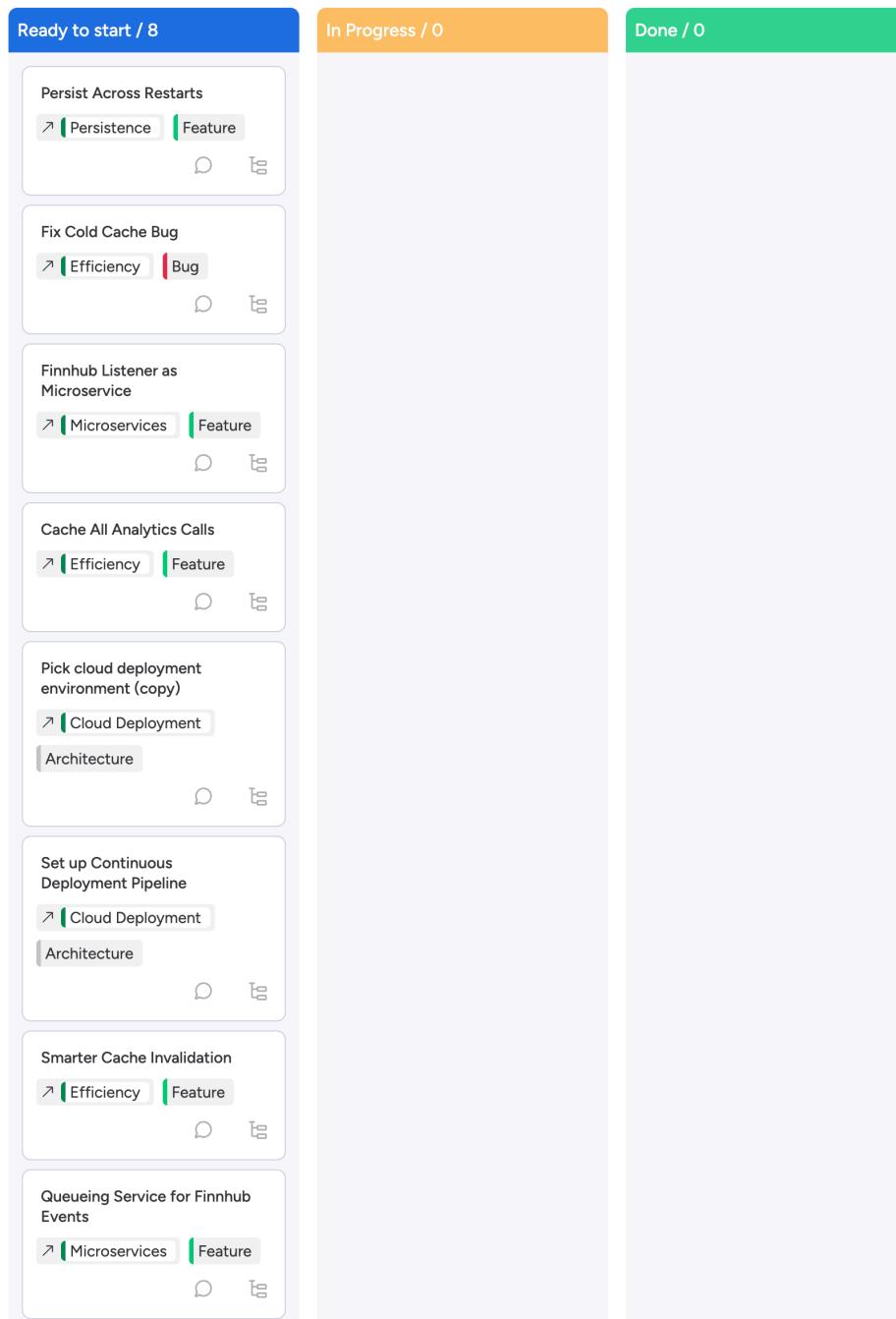
Jarred



Donna

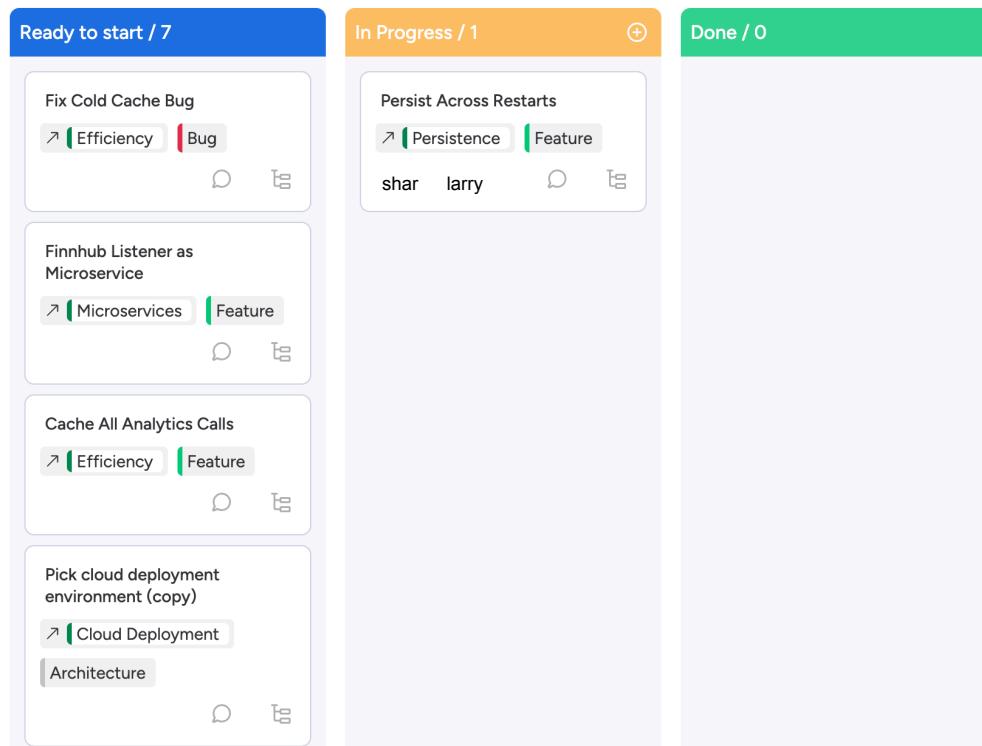


Department



- team of 5 developers
- starting with this prioritized roadmap
- question to ask:
  - who should start work on highest priority task?





Jane



Jarred



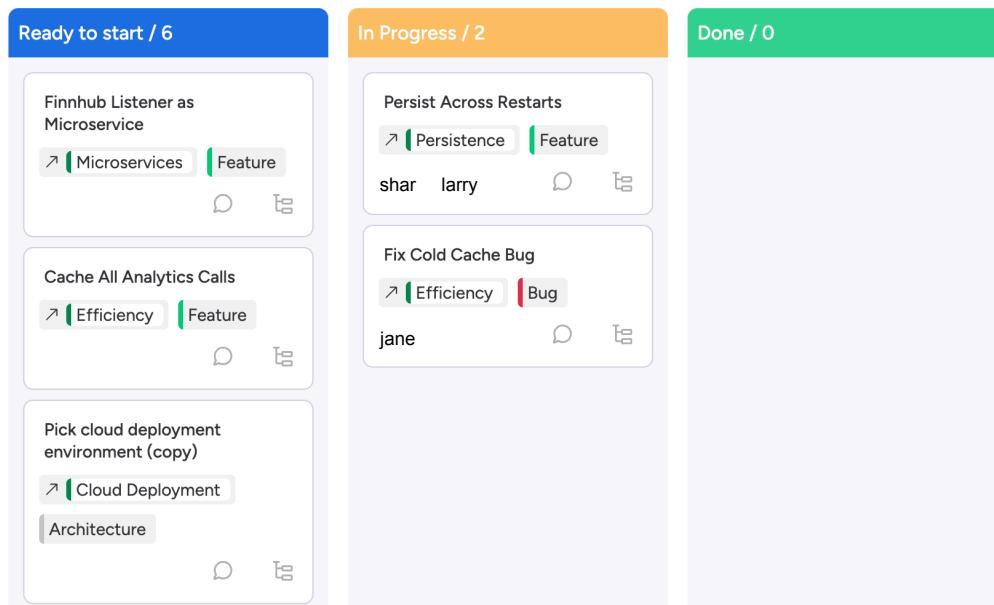
Donna

- Shar & Larry start work on persisting state across StockApp restarts.
- Still have 3 free engineers, what should they do?



Department of Electrical &amp; Computer Engineering





- Jane has some background on caching and volunteers to take a look at the bug
- We avoid anyone working alone on a feature, but for smaller bugs its ok for Jane to look at this on her own
- What do Jarred and Donna do?



Jarred

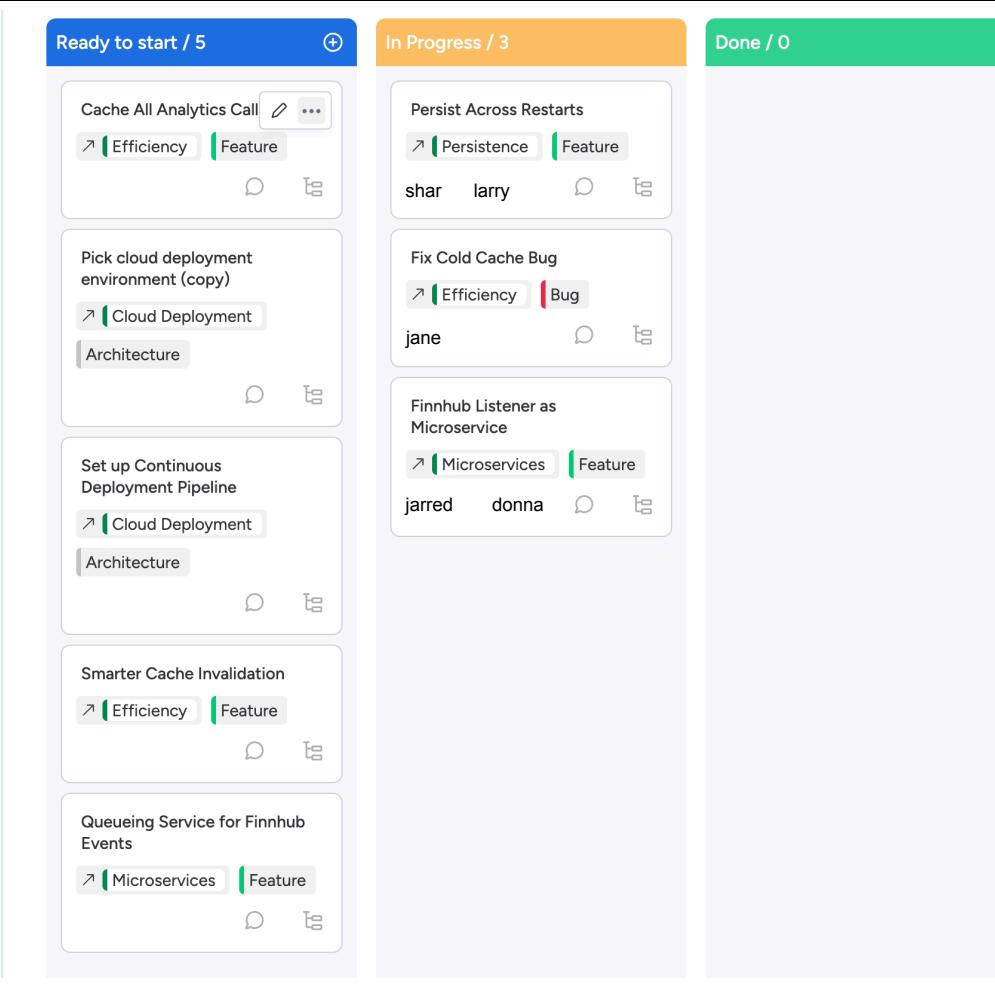


Donna



Department of Electrical &amp; Computer Engineering



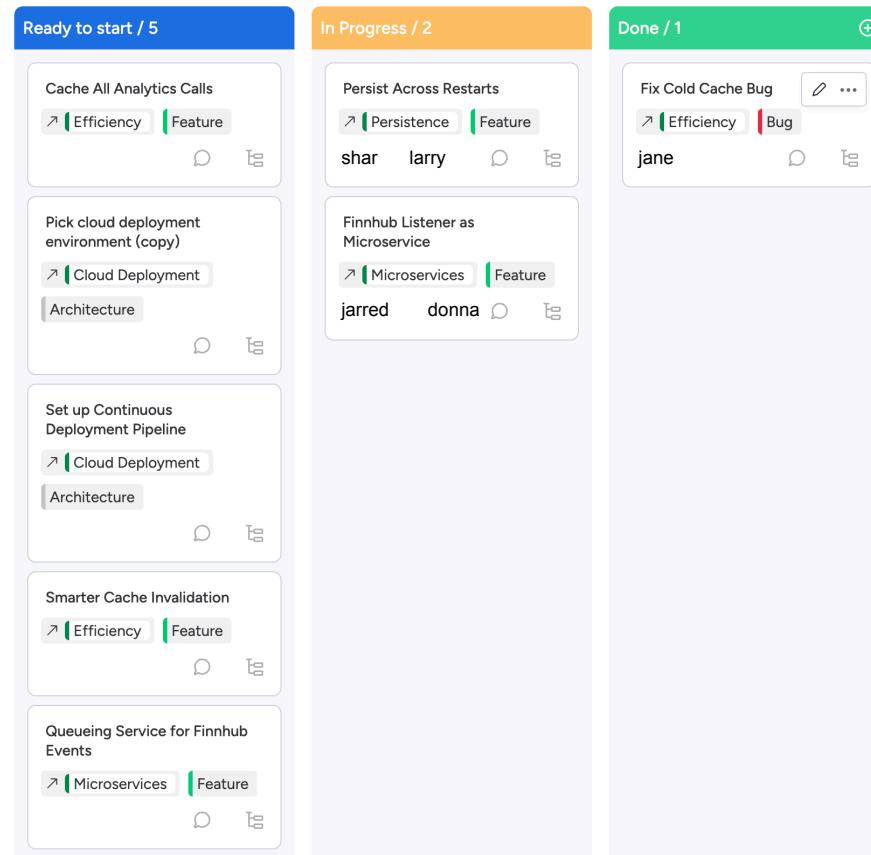


- Jarred and Donna pick up the Finnhub Listener as Microservice work and now our team are all making progress towards our highest priority tasks...





Jane



- One day later...
- Jane ships the cache bug fix (jarred took an interrupt to review her Pull Request)
- Jane has now freed up, what should she do?

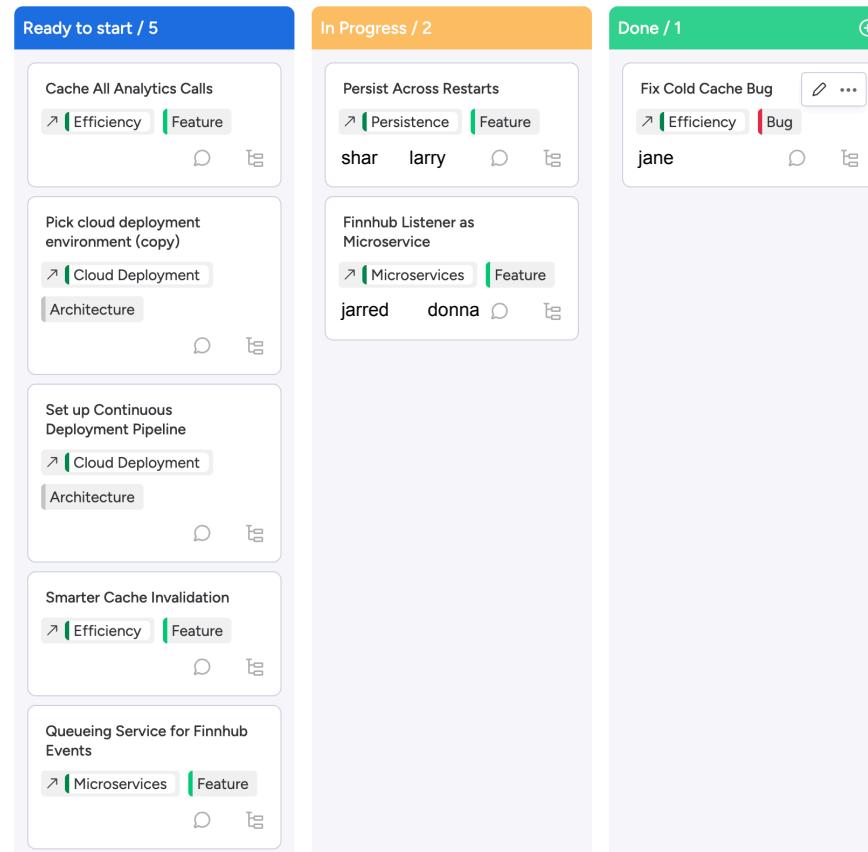


Department of Electrical &amp; Computer Engineering





Jane



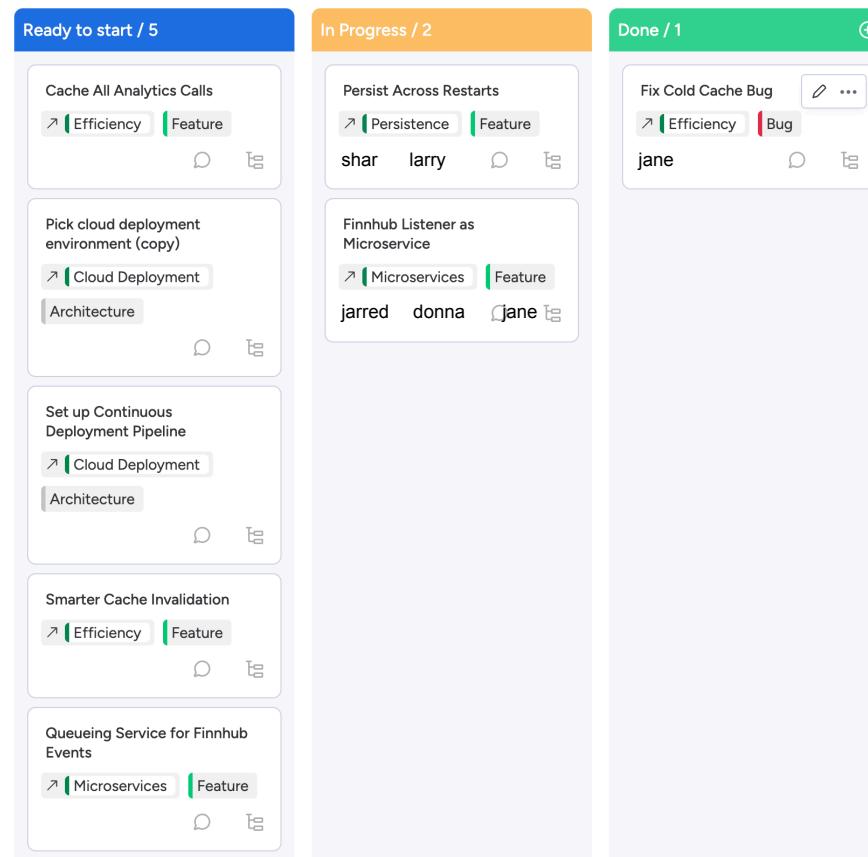
- First, everyone gives Jane a high five for shipping the fix!

- Pulling a new feature would push our WIP too high - 3 features across 5 engineers is too much.
- Jane should ask Shar & Larry and Jarred & Donna which team could use more help and join one of the two in progress features based on their response and her skill set.



Department of Electrical &amp; Computer Engineering





- Jane has some experience with WebSockets and Jarred and Donna tell her that there is plenty of work left to do on their project, so she joins their team.



# Kanban Tenets

- Visualize all work
- Limit Work In Progress
- Keep Increments Small





developer

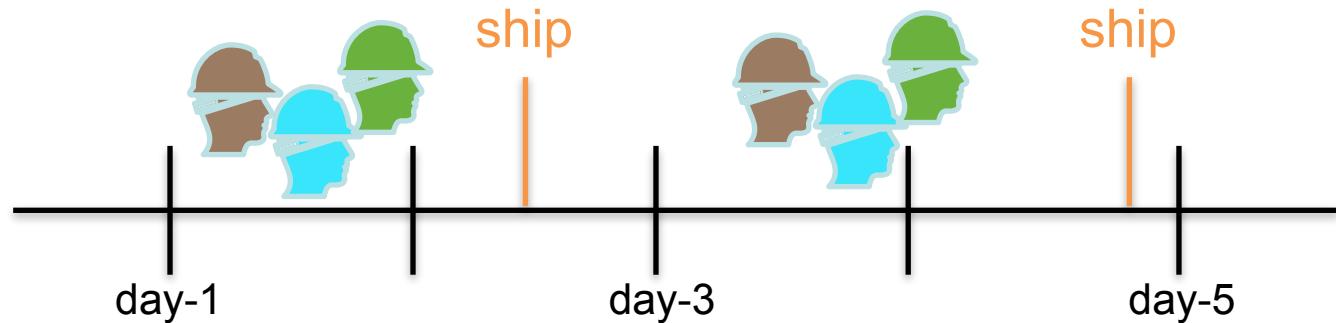


quality  
assurance



release  
engineer

## Merging of Roles



- two factors pushing for the merging of these 3 roles:
  - automation of testing and integration + deployment
  - compressed time frames & iterations
- resulting in ...
  - DevOps!



## Day 2 Hands-on Exercise 5

- Work in Teams of 4 or 5 - people at your table are a great group to work with!
- Pick one of your repos as the one to use for your final project and:
  - Set up a Kanban board in your GitHub Repo
  - Invite others from your team to that repo
- Given that you have access to Lambda functions and S3, think about what kind of interesting / useful function - API based application you could build in one day
  - Add tasks to your Kanban board to reflect this and plan the implementation.

