

# Why and How We Went Serverless, and How You Can Too

CNI Spring 2021 Virtual Membership Meeting

Yinlin Chen and Bill Ingram  
Virginia Tech Libraries  
March 15-26, 2021

# The Presenters



**Yinlin Chen**

Digital Library Architect And  
Assistant Professor  
University Libraries,  
Virginia Tech



**William Ingram**

Assistant Dean  
University Libraries,  
Virginia Tech

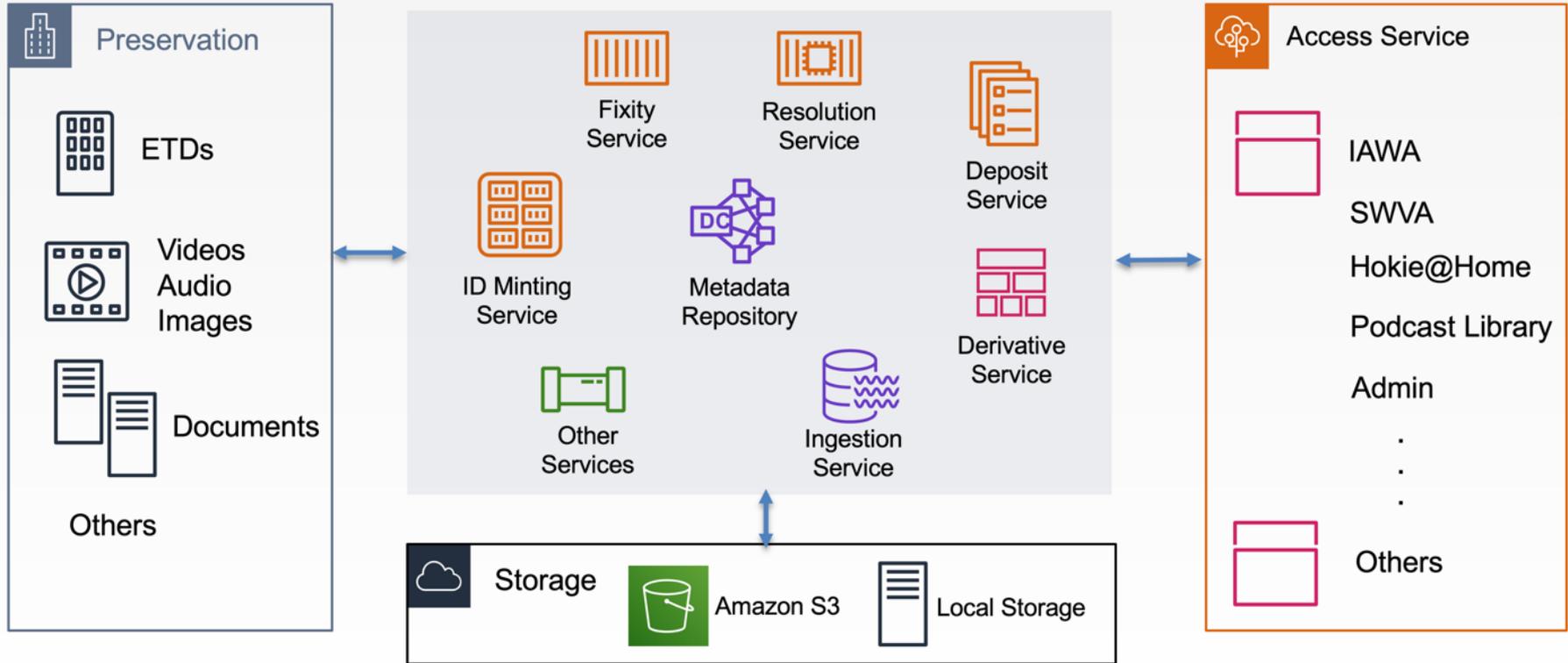
# Outline

- Challenges
- Virginia Tech Digital Library Platform
- Methodology
- What we did: Before vs Now
- Conclusion
- Future work

# Challenges

- Numerous, web applications with similar stacks stretching resources
- Limited in-house capacity to address performance, resilience, and scaling
- Library-specific software requires training or competing for few experienced library Software developers and Sysops
- The complexity nature to modify or add new feature in the monolithic applications

# Virginia Tech Digital Library Platform



# Methodology

## The twelve-factor app

### 1. Codebase

One codebase, many deploys

### 2. Dependencies

Explicitly declare and isolate dependencies

### 3. Config

Store config in the environment

### 4. Backing services

Treat backing services as attached resources

### 5. Build, release, run

Strictly separate build and run stages

### 6. Processes

Execute the app as one or more stateless processes

### 7. Port binding

Export services via port binding

### 8. Concurrency

Scale out via the process model

### 9. Disposability

Maximize robustness with fast startup and graceful shutdown

### 10. Dev/prod parity

Keep development, staging, and production as similar as possible

### 11. Logs

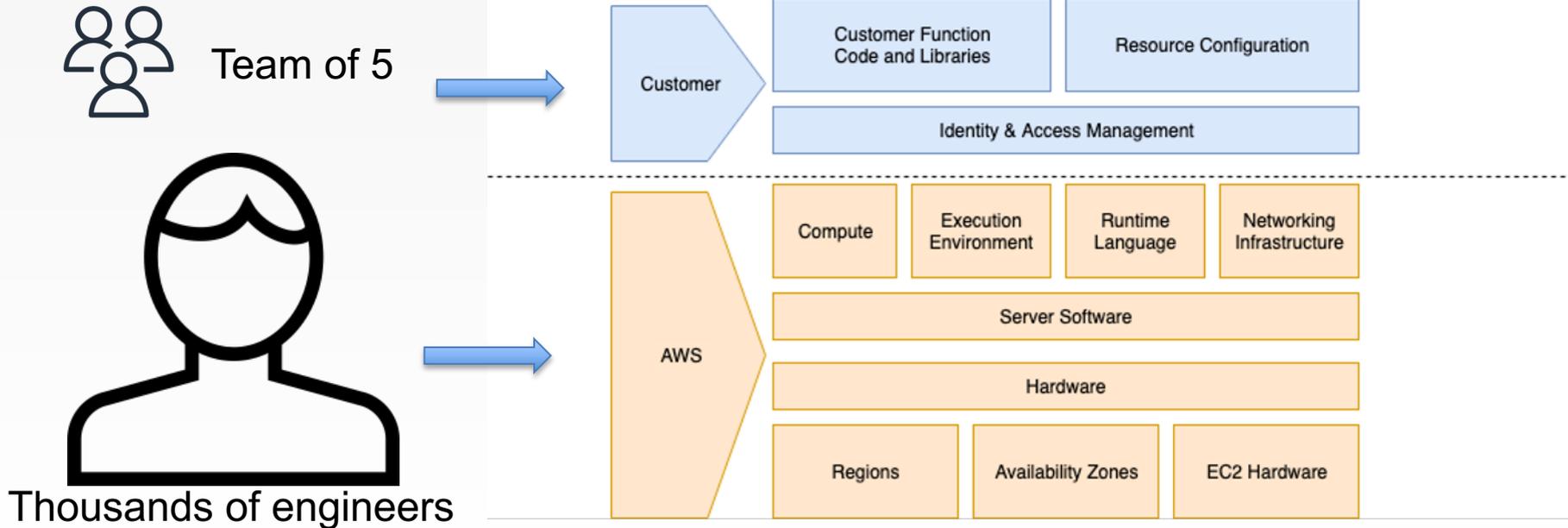
Treat logs as event streams

### 12. Admin processes

Run admin/management tasks as one-off processes

*If I have seen further, it is by  
standing on the shoulders of Giants*

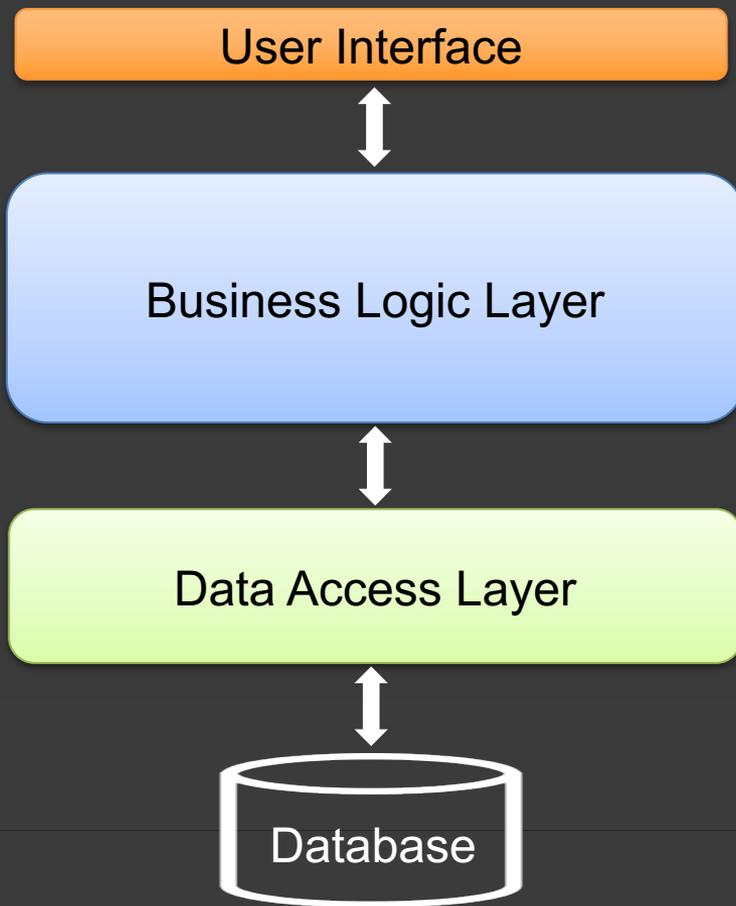
- Isaac Newton



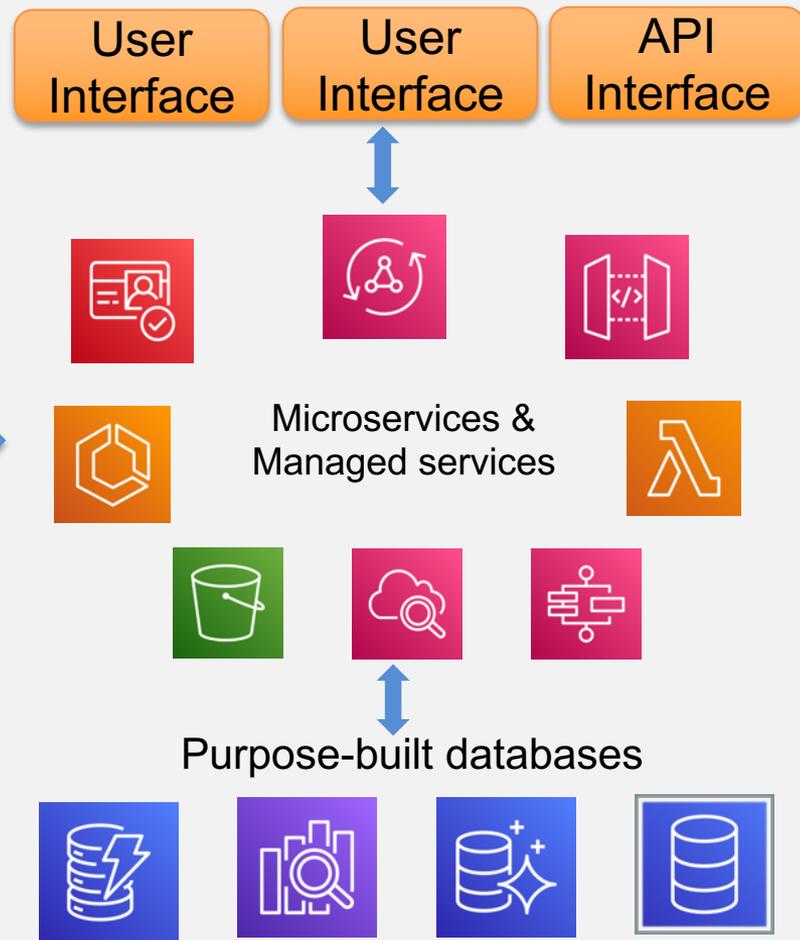
# Principles of Serverless Design

- ✓ Design push-based, even-driven pipelines
- ✓ Write single-purpose, stateless functions
- ✓ Execute functions on demand and use the resource that needs exactly
- ✓ Create thicker and more powerful front ends
- ✓ Three Pillars of Observability: Logs, Metrics, & Traces

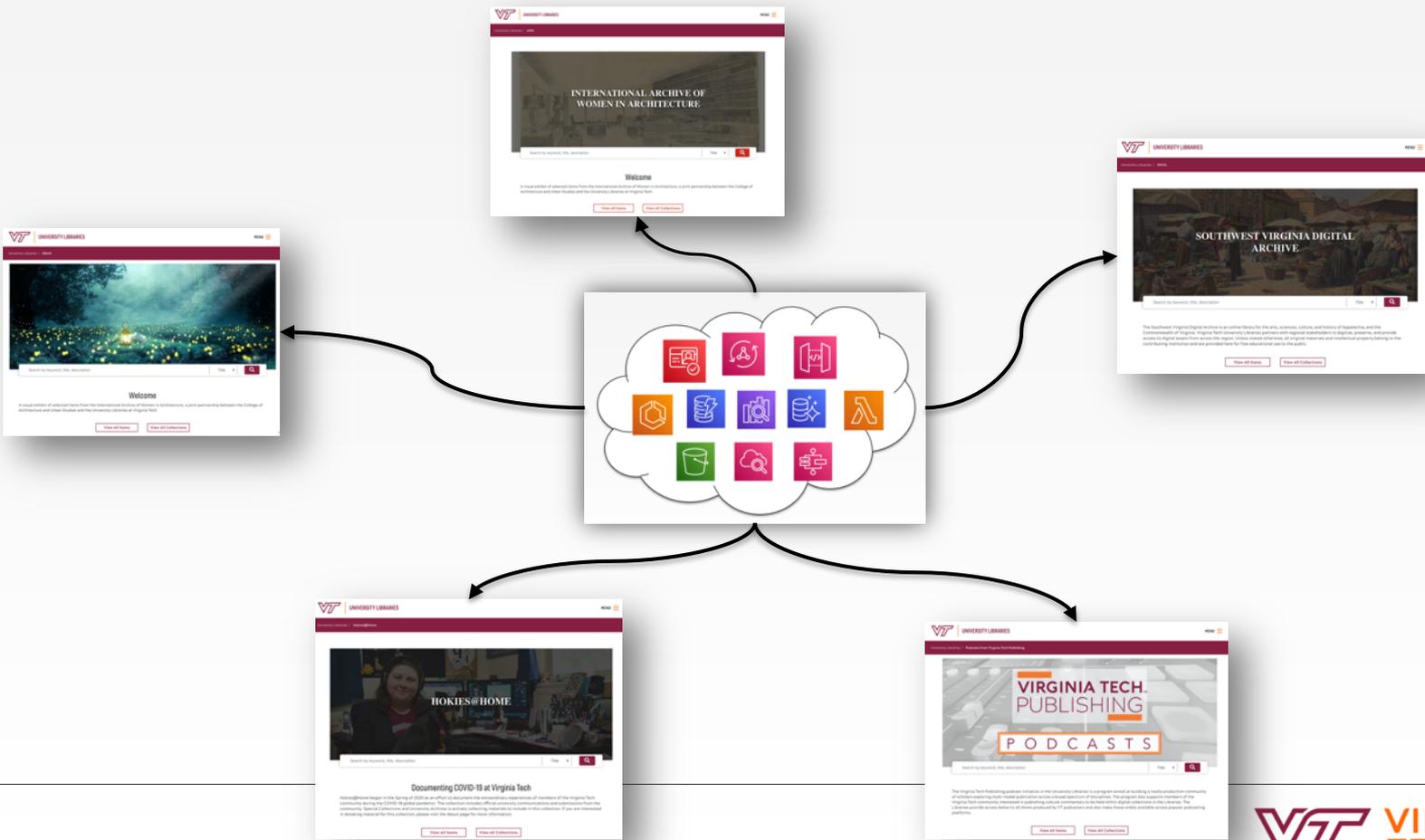
# Monolithic Architecture



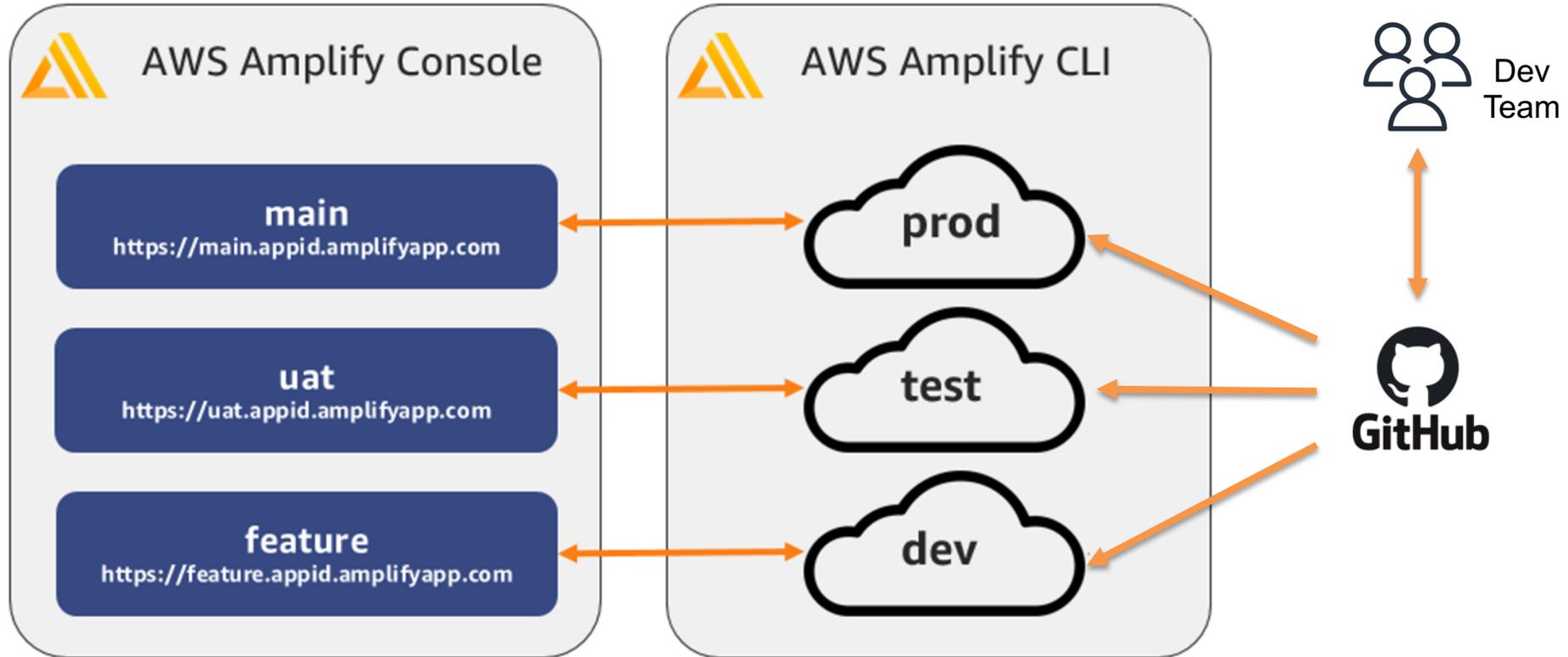
# Serverless Architecture



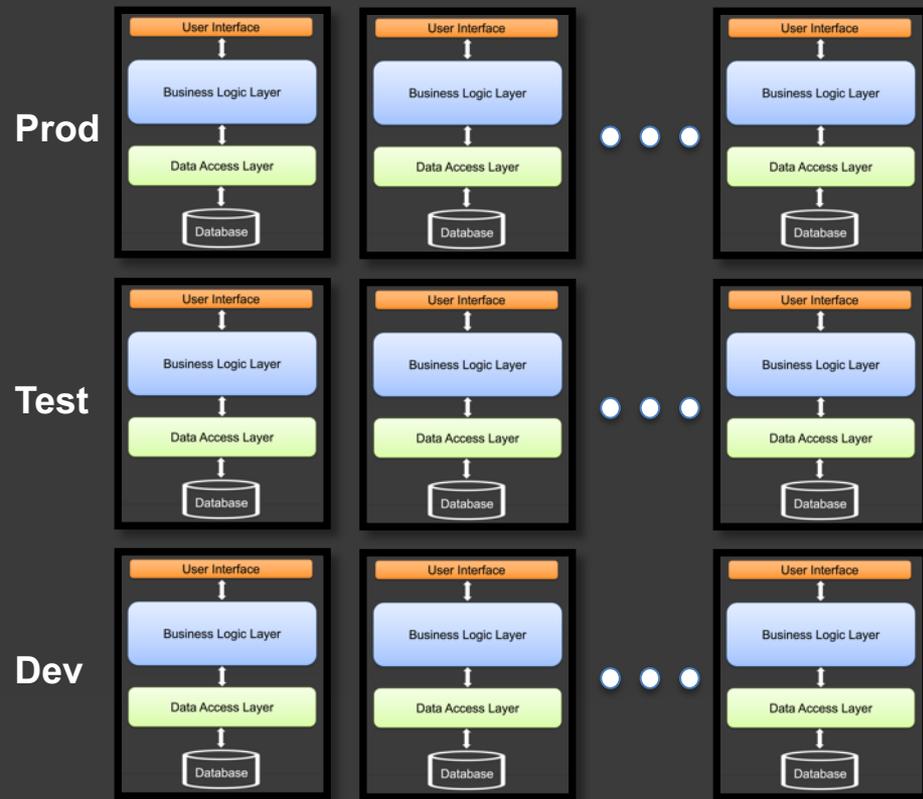
# Multi-Tenant Web Applications



# Multi-stage deployments through Dev/Test/Prod Environments

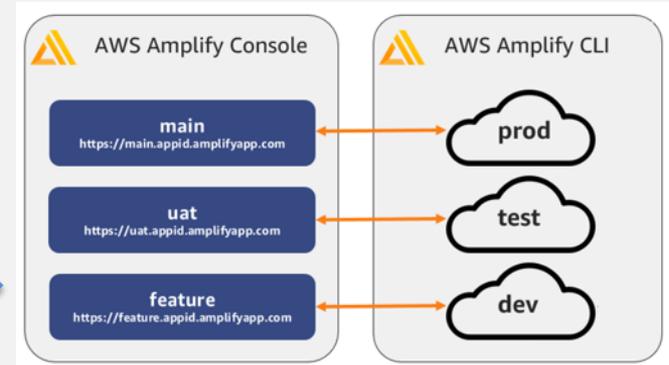


# Multi-site deployment : Before



Ansible Playbooks, Shell scripts, etc.

# Multi-site deployment: Now



- No Servers need us to perform maintenance
- Unlimited ephemeral Test & Dev deployments
- Step up deployment and let AWS do the rest
- Initially a small instance is created and let AWS to scale it up and down for us
- ... and many many more

# Fully Automated CI/CD Pipeline

All apps > dlp-access > demo

demo

View latest build

View build history

Build 60

Redeploy this version



Domain

<https://vtdlp-demo.cloud.lib.vt.edu>

Source repository

<https://github.com/VTUL/dlp-access/tree/demo>

Started at

3/2/2021, 2:18:21 PM

Last commit message

[add archive permission](#)

Build duration

18 minutes 32 seconds

Provision

Build

Test

Deploy

Verify

All Cypress specs completed! 27 spec(s) passed

View log

Download artifacts

Spec name	Number of tests	Total duration	Video
About link	2 passed	00:05	Download artifacts to see this video.
Displays and updates browse collections page configurations	8 passed	00:35	Download artifacts to see this video.
Update attribute and change it back	5 passed	00:17	Download artifacts to see this video.

# Every Single Feature (Change) has a Preview

VTUL / dlp-access

<> Code Issues Pull requests 2

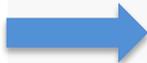
Labels Milestones New

Filters is:pr is:closed is:merged is:comments < 3 is:checked

Clear current search query, filters, and sorts

15 Total

Author	Label	Assignee	Sort
Updated to Mirador 3 ✓	Ready for review		2
#217 by andreaWaldren was merged on Nov 23, 2020			
upgrade cypress to 5.6 ✓			1
#212 by yinlinchen was merged on Nov 12, 2020			
refactor collection hierarchy sort ✓	Ready for review		2
#185 by whunter was merged on Sep 18, 2020			
LIBTD-2119: Change data model to make rights, rights_holder and citat... ✓			1
#85 by tingtingjh was merged on Apr 6, 2020			



All apps > dlp-access-2

## dlp-access-2

The app homepage lists all deployed frontend and backend environments.

Learn how to get the most out of Amplify Console 3 of 5 steps complete

Frontend environments Backend environments

This tab lists all connected branches, select a branch to view build details. Connect branch

### federated

Continuous deploys set up with newdev backend (Edit)

<https://digital-dev.cloud.lib...>

Provision Build Test Deploy Verify

Last deployment	Last commit	Previews
2/17/2021, 1:52:35 PM	Web component in footer (#243)   4bca558   GitHub - Federated	Disabled

### podcast

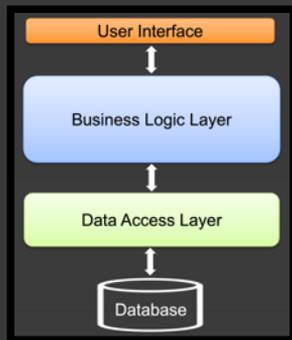
Continuous deploys set up with newdev backend (Edit)

<https://podcasts-dev.cloud.lib...>

Provision Build Test Deploy Verify

Last deployment	Last commit	Previews
1/29/2021, 5:35:19 PM	Updated one theme header (#239)   af3563a   GitHub - podcast	Disabled

# Before



## Resource Usage



# Now



## Resource Usage



## Website performance

Page	IAWAv1	IAWAv2	Percent Change
Index Page <a href="#">V1</a> <a href="#">V2</a>	796 ms	766 ms	4% faster
Item Browse Page <a href="#">V1</a> <a href="#">V2</a>	1.35 s	882 ms	35% faster
Collection Browse Page <a href="#">V1</a> <a href="#">V2</a>	926 ms	754 ms	20% faster
Item Page <a href="#">V1</a> <a href="#">V2</a>	1.01 s	861 ms	16% faster

There are other things that we can not compare the performance, because we can only do those in AWS

# Image Processing: Before

## Batch Scripting Jobs in Server

We only have 1 spare server for this, we will do 1 collection a time

It looks like we need 7 days to process one set of images, is it OK?

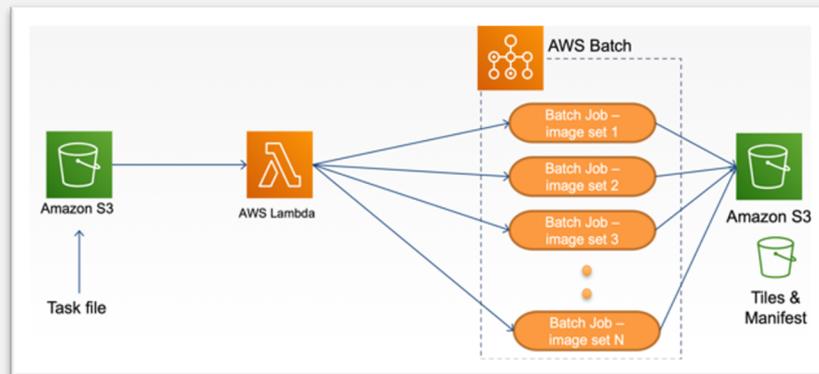
Run 11 servers for a week for this... seems expensive and hard to manage them.

We have 11 collections of images need to be processed

That take too long, can you provision servers in AWS and do all at once?

# Image Processing: Now

## Serverless Batch Jobs



- Processed 11 collections of images in 2 days
- Expect to process 20, 30, ... collections of images will still in 2 days
- There are still space to optimize microservices and make it even faster

# Infrastructure as Code (DevOps)

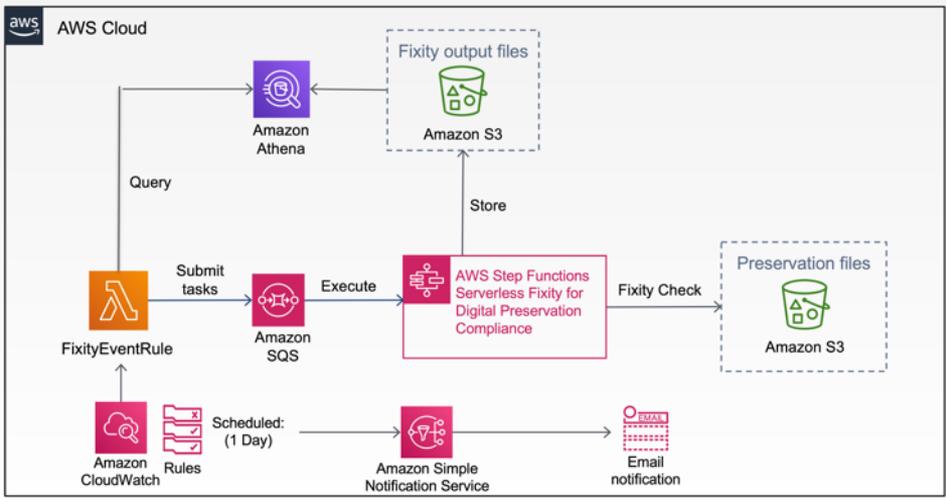
```

SubmitSteps:
  Type: AWS::Serverless::Function
  Properties:
    CodeUri: fixitycheck/
    Handler: steps.lambda_handler
    Runtime: python3.8
    Timeout: 30
    Layers:
      - !Ref SharedUtils
    Policies:
      - StepFunctionsExecutionPolicy:
        StateMachineName: !Ref StateMachineName
  Events:
    FixitySQSEvent:
      Type: SQS
      Properties:
        Queue: !GetAtt FixitySqsQueue.Arn
        BatchSize: 10

Environment:
  Variables:
    Region: !Ref Region
    StateMachineArn: !Sub "arn:aws:states:${Region}

FixitySqsQueue:
  Type: AWS::SQS::Queue

FixitySNS:
  Type: AWS::Serverless::Function
  Properties:
    CodeUri: fixitycheck/
    Handler: sns.lambda_handler
    Runtime: python3.8
    MemorySize: 2048
    Timeout: 120
    Layers:
  
```



**Launch Stack**

One click deploy

```

CloudFormation stack changeset
-----
Operation                               LogicalResourceId                       ResourceType
-----
* Modify                                  ConvertFileFunction                       AWS::Lambda::Function
* Modify                                  UploadBucket                              AWS::S3::Bucket

Changeset created successfully. arn:aws:cloudformation:us-east-1:909117335741:changeSet/samcli-deploy1614752401/f3165ad7-122e-4644-bd52-f4b4c8b1c8b1

2021-03-03 01:20:17 - Waiting for stack create/update to complete

CloudFormation events from changeset
-----
ResourceStatus                           ResourceType                               LogicalResourceId
-----
UPDATE_IN_PROGRESS                        AWS::Lambda::Function                     ConvertFileFunction
UPDATE_COMPLETE                           AWS::Lambda::Function                     ConvertFileFunction
*UPDATE_COMPLETE_CLEANUP_IN_PROGRESS      AWS::CloudFormation::Stack                testhello
UPDATE_COMPLETE                           AWS::CloudFormation::Stack                testhello
  
```

# Monitor Metrics for Applications in Near Real-time

## Monitoring

Monitor your site traffic, set up alarms, or download access logs. [Learn more](#)

**Metrics** | Alarms | Access logs

► Metrics definitions

Add to dashboard 1h 3h 12h 1d 3d 1w custom - ↻ ▾

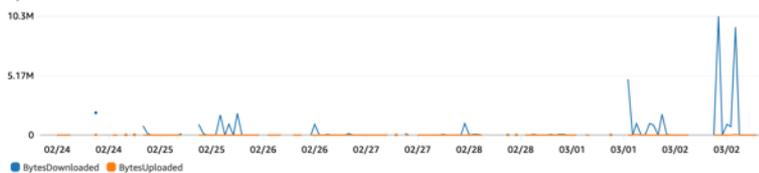
### Requests (sum)

No unit



### Data transfer (sum)

Bytes



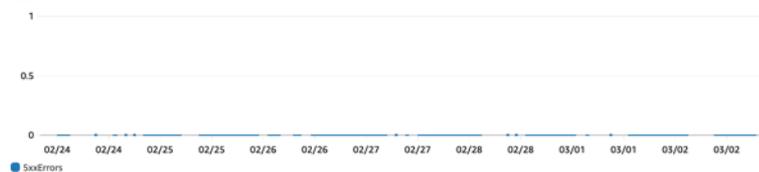
### 4xx errors (sum)

No unit



### 5xx errors (sum)

No unit



### Time to first byte (average)

Seconds



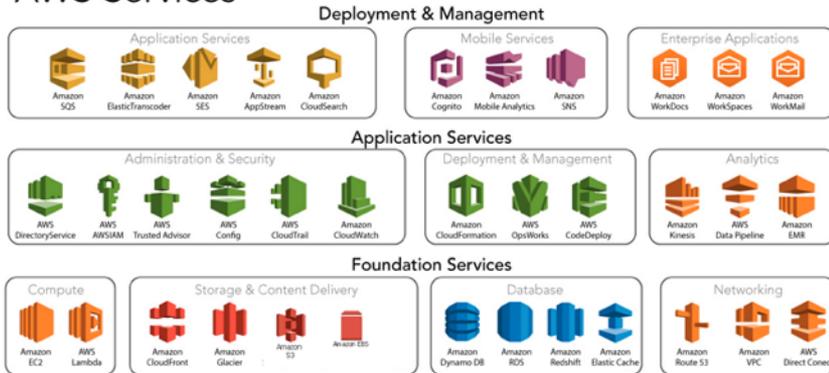
# Cost: Before



- Add one server, plus the cost
- Plan server capacity each year
  - Overestimation: shortage is even worse, so would rather purchase more than we need (Wasting resource is fine, we already paid money, but really?)

# Cost: Now

## AWS Services



- Pay what we use
  - Use it well than pay less
  - Pay 0 when services are idle
- Some (invisible) cost are vanished
  - Server maintenance cost
  - Labor, time, electric, etc.

# Conclusion

- AWS services just like Lego blocks, we pick and choose blocks and build powerful, scalable applications that suit our business
- With serverless, we write less code, fasten development speed, and enable creativities and innovations
- Applications we built are more resilient, secure, scalable, and cost-efficient than in a traditional, server-based environment
- AWS communities are larger than any software communities in the libraries



Image source: Lego.com

# Future Work

- New features and services
  - AI/ML/DL as a service
  - Large-scale text and data mining
  - Digital preservation and the management
- Continue performance tuning and cost optimization of cloud-native services
- Continue refactoring of legacy applications to AWS
- Collaborate with other institutions on the Serverless Cloud

# Thank you!



## Virginia Tech Digital Libraries

Cloud-native, multi-tenancy digital library software

<https://github.com/vt-digital-libraries-platform>



### aws-batch-iiif-generator



A project that demonstrates the use of AWS Batch to create IIIF tiles and manifests. It is used for VTDLP Derivative Service

● Python



### DLPservices



VTDLP Services



### resolution-service



VTDLP Resolution Service

● Python



### dlp-access



DLP Access Website: A Multi-Tenancy Serverless web application

● JavaScript



### NOID-mint



Mint Nice Opaque Identifiers

● Python



### FixityService



VTDLP Fixity Service: Regularly scheduled fixity checks and comparisons to the files stored in the AWS S3

● JavaScript