
AWS
re:Invent

STG310

Modernize your applications with AWS Lambda and Amazon EFS

Will Ochandarena
Principal Product Manager
AWS

Jeff Tougas
Director, Software Engineering
Asurion

AWS re:Invent

© 2020, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Agenda

Modernization basics

Getting started with Amazon EFS and AWS Lambda

Best practices – performance, cost, and ingest

Case study – how Asurion uses Lambda and Amazon EFS to analyze customer experiences in real time



© 2020, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Amazon EFS sessions

STG213 What's new with Amazon EFS

STG311 Optimizing Amazon EFS for cost and performance, using best practices

STG307 Deep dive on Amazon EFS

STG310 Modernize your applications with AWS Lambda and Amazon EFS

STG312 Modernize your applications with containers using Amazon EFS

STG306 Security best practices with Amazon EFS

STG313 Reduce TCO for your Linux file-based applications

STG314 Amazon EFS performance best practices for Linux workloads

Modernization basics



AWS re:Invent

© 2020, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Why modernize?



1. Increased agility and scalability



2. Faster time to market



3. Improved reliability, simpler operations, and lower cost

Modernizing with Containers, AWS Fargate, and Lambda

More

Opinionated

Less

	AWS manages	Customer manages
AWS Lambda Serverless functions	Data source integrations Physical hardware, software, networking, and facilities Provisioning	Application code
AWS Fargate Serverless containers	Container orchestration, provisioning Cluster scaling Physical hardware, host OS / kernel, networking, and facilities	Application code Data source integrations Security configuration and network configuration Management tasks
Amazon ECS/EKS Container-management -as-a-service	Container orchestration control plane Physical hardware software, networking, and facilities	Application code Data source integrations Work clusters Security configuration and updates, network configuration, firewall, management tasks
Amazon EC2 Infrastructure-as-a-Service	Physical hardware software, networking, and facilities	Application code Data source integrations Scaling Security configuration and updates Network configuration Provisioning, managing, scaling, and patching of servers

What is Amazon EFS?

Simple, scalable, fully managed, highly durable, and available shared file system for AWS compute

SIMPLE AND HIGHLY RELIABLE



Elastic

Pay only for capacity used

Performance built-in, scales with capacity



3-AZ durable and all-AZ available

Designed for 11 9's of durability

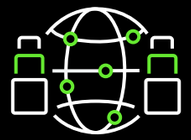
99.99% availability SLA

SERVERLESS SHARED STORAGE



Serverless and scalable

No provisioning, scale capacity, connections, and IOPs



Concurrent access for 10,000s of connections

EC2 instances, containers, and Lambda invocations

PERFORMANT AND COST OPTIMIZED



Performant

10s of GB/s of and 500,000+ IOPS



Two storage classes

Lifecycle-based cost optimization

Benefits of Amazon EFS for Lambda

Simple

Amazon EFS and Lambda are fully serverless, so no instances to manage

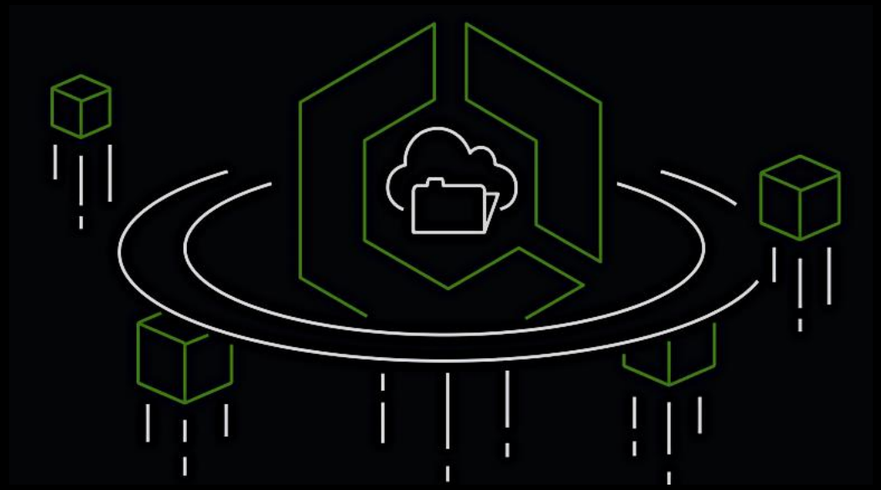
Amazon EFS configuration is inside Lambda function definition

Developers can focus on their applications, not infrastructure

Available and durable

Lambda and Amazon EFS are regional services

Customers can build applications that span multiple availability zones, with automatic failover



Elastic

Lambda and Amazon EFS are elastic and scale up and down rapidly based on demand

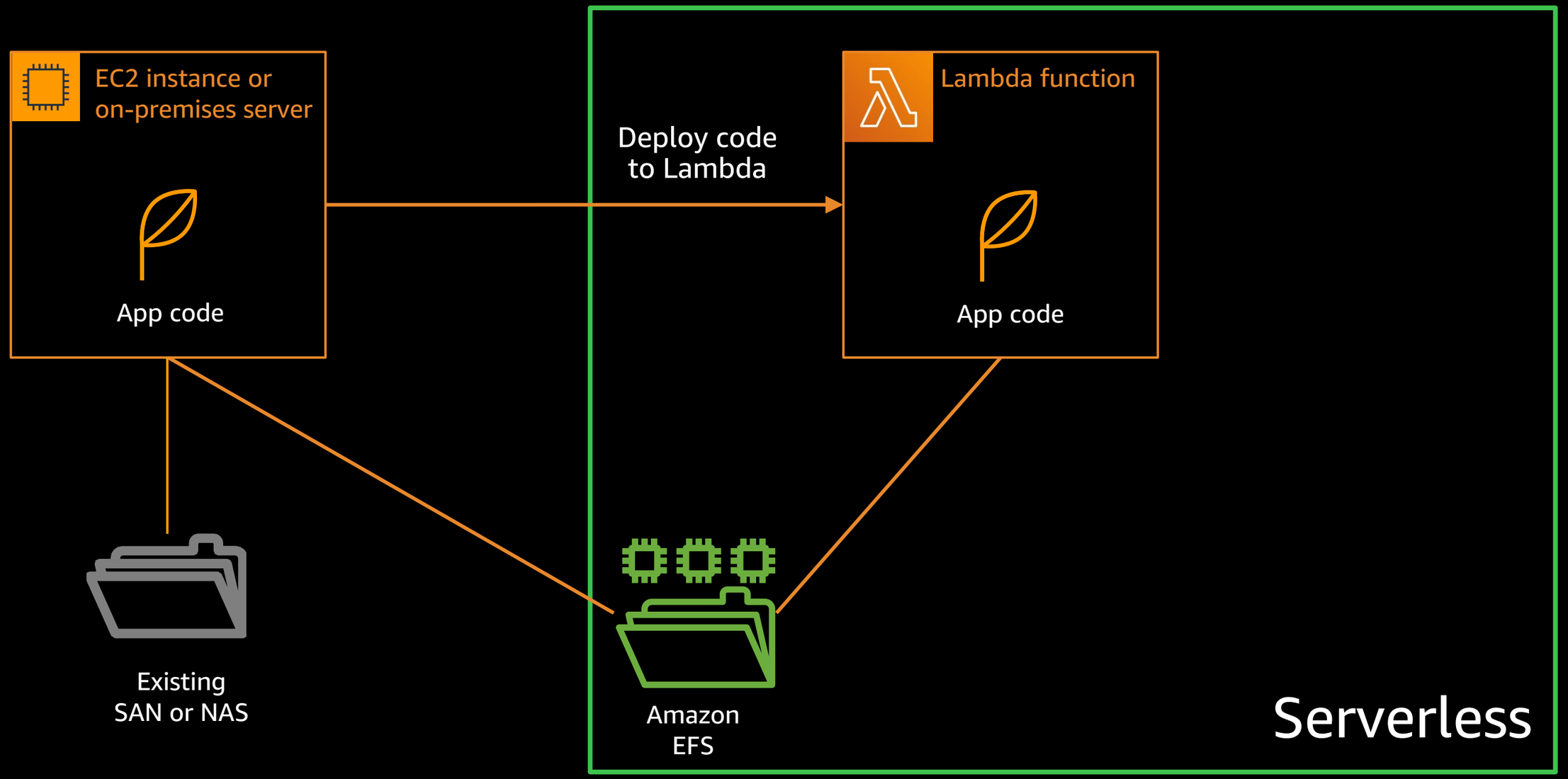
Customers pay only for what they use

Secure

Access to Amazon EFS can be restricted based on AWS Identity and Access Management (IAM) role of Lambda function

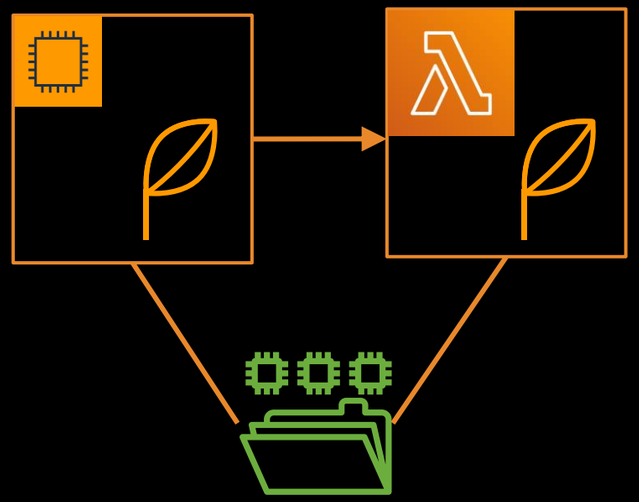
Amazon EFS Access Points can enforce file system permissions when multiple apps share a file system

App modernization with Lambda



Modernization approaches

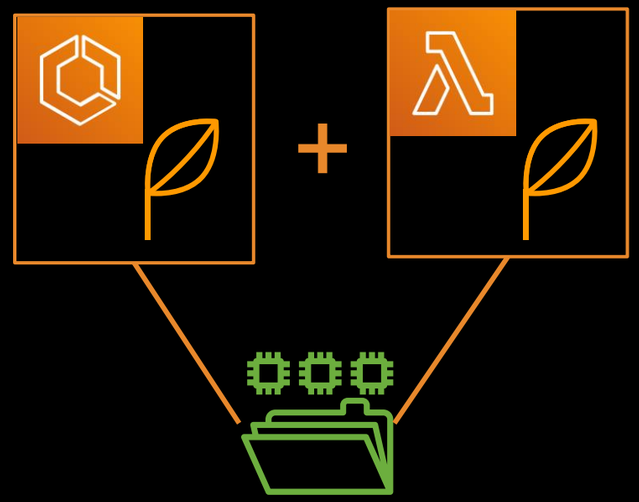
App migration



Web / API serving

Batch simulations

App augmentation



Real-time ML inference

Virus scanning

Migration use case: Web / API serving



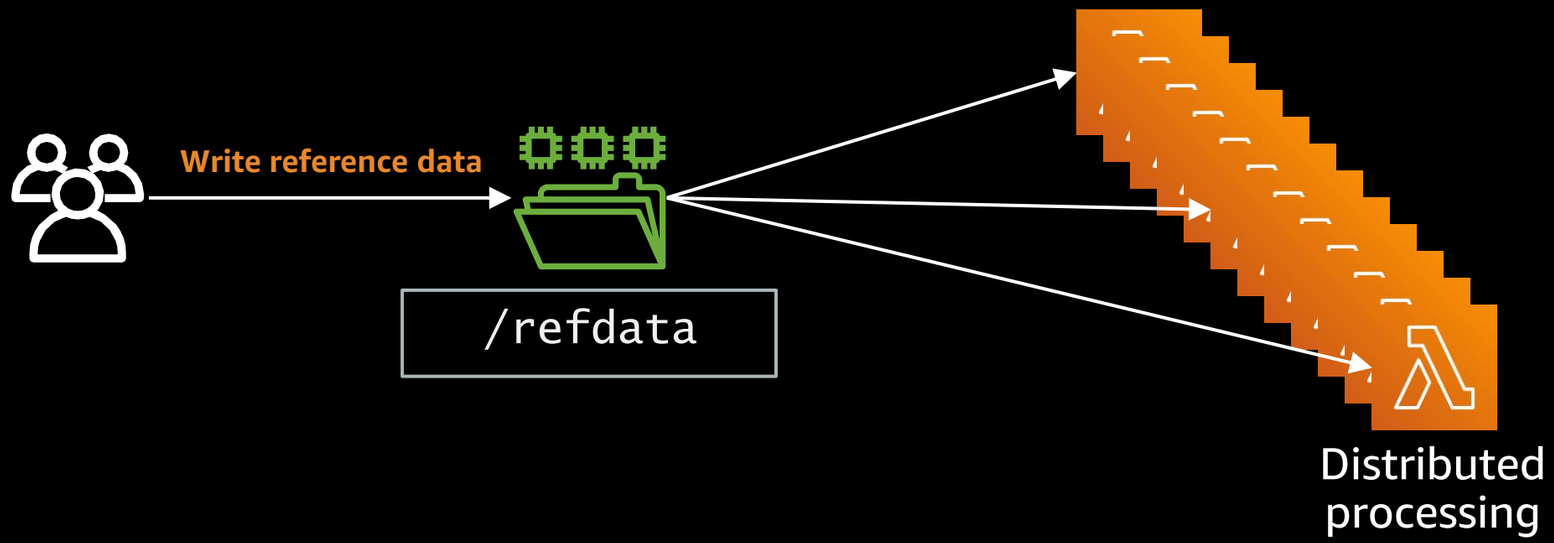
Instantly scale based on demand

Share web assets across all invocations

Content changes strongly consistent

No infrastructure to manage

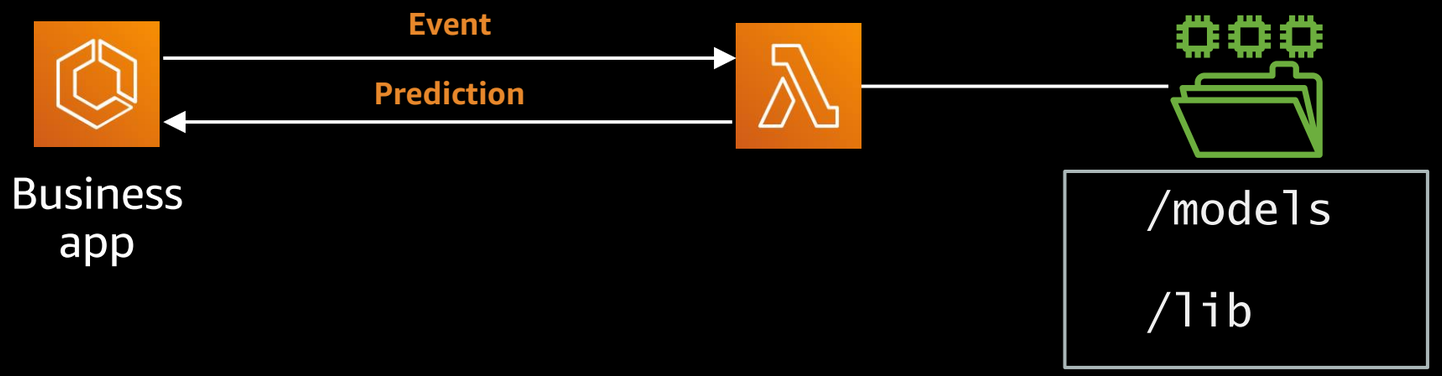
Migration use case: Batch simulations



Instantly scale-up parallel processing

Pay only for execution time and data stored

Augmentation use case: Real-time ML inference



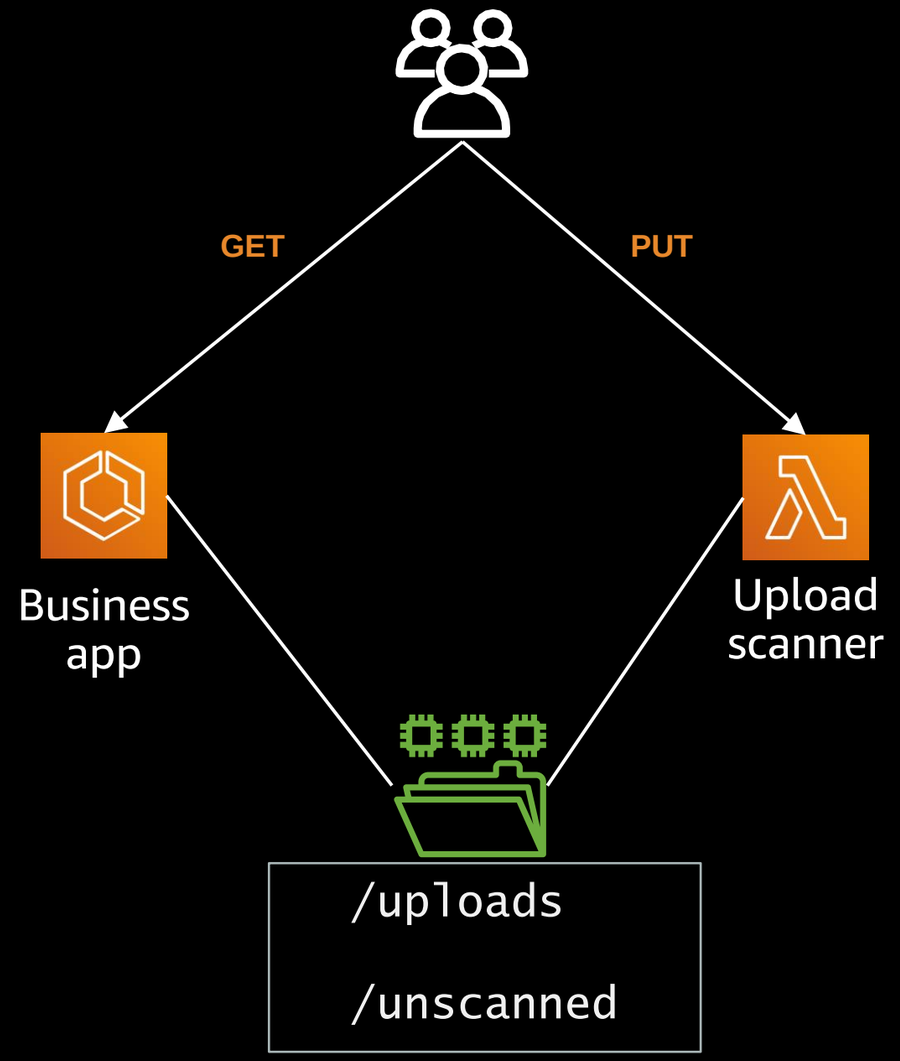
- TensorFlow
- PyTorch, NumPy, Keras
- Pre-trained AI models
- C++ libraries
- …

Import large AI models quickly

Load extra code libraries

Deploy modeling and AI solutions with Lambda

Augmentation use case: Virus scanning



Unlimited file storage for data-intensive microservices

Increased agility through decoupling of dev / release

Simplicity of data sharing between microservices

Getting started with Amazon EFS and AWS Lambda

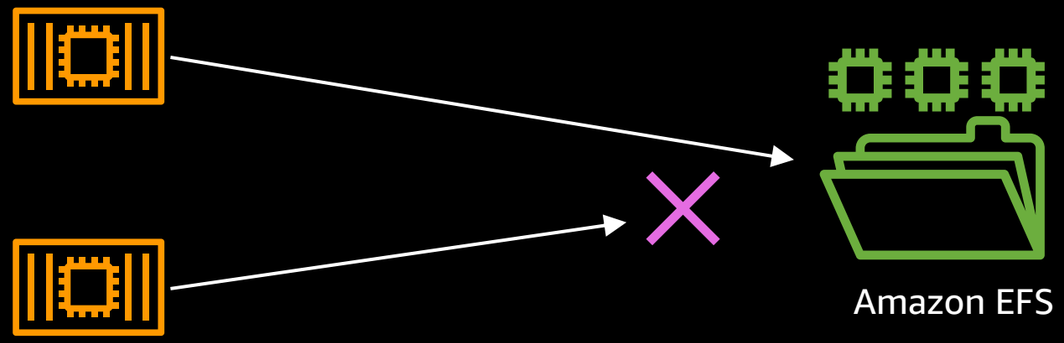


AWS re:Invent

© 2020, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Goals for security and identity

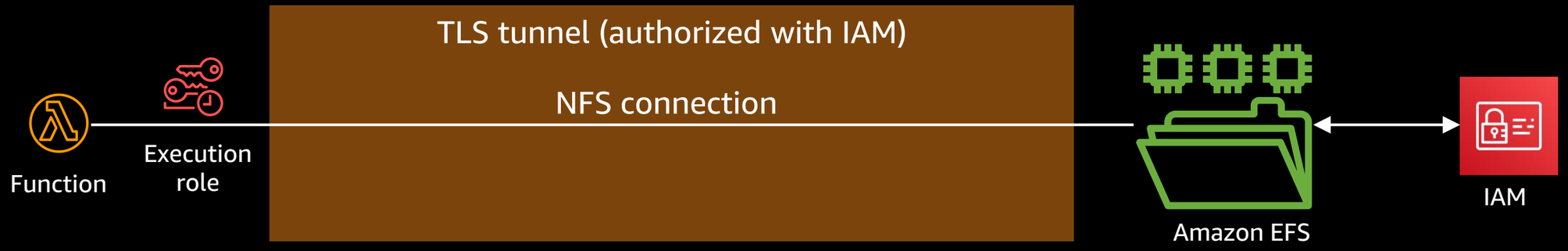
1. File systems should only be mountable by applications that need them
2. Apps that mount file systems should only have access to data they need




```
$ cat /my_app/data
### SUCCESS THIS IS MY FILE ###

$ cat /someone_elses_app/data
cat: /someone_elses_app/data : Permission denied
```

Using IAM for file system access



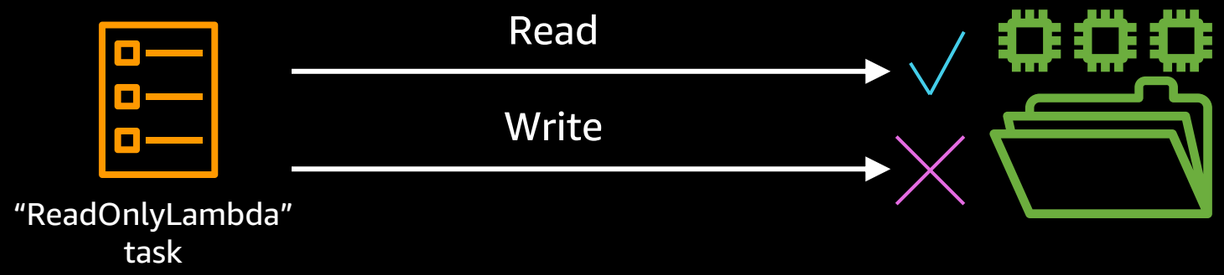
Identity policy attached to IAM role

```
{
  “Statement” : {
    “Effect” : “allow”,
    “Action” : “elasticfilesystem:Client*”,
    "Resource": ”fs-feedfeed"
  }
}
```

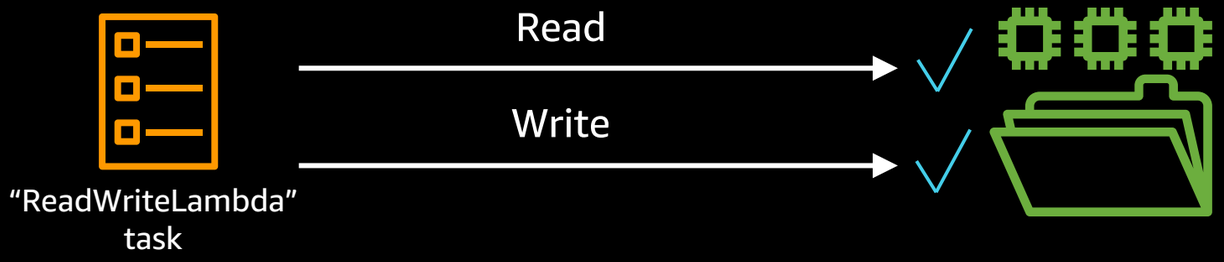
File system resource policy

```
{
  “Statement” : {
    “Effect” : “allow”,
    “Action” : “elasticfilesystem:Client*”,
    “Principal” : { “AWS”: “arn:..Lambda” }
  }
}
```

Handling Amazon EFS authorization using IAM

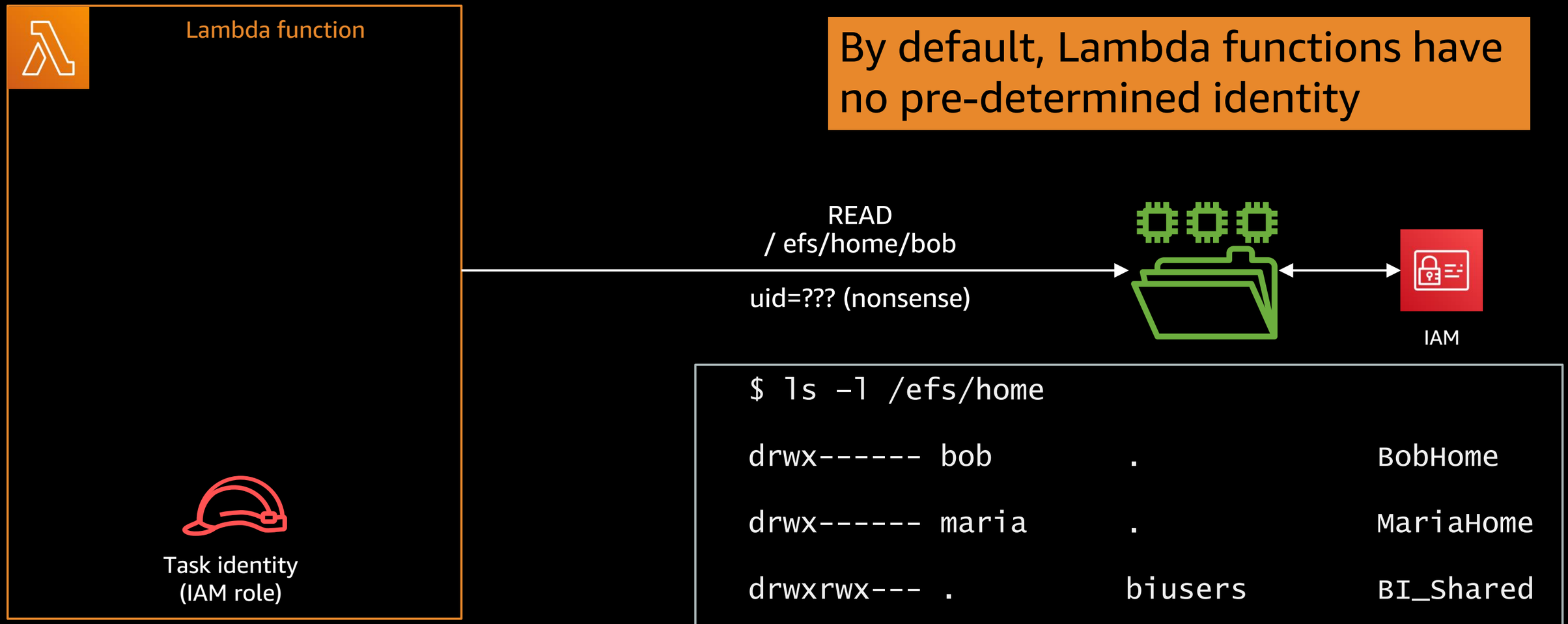



```
"Effect" : "allow",
"Action" : "elasticfilesystem:ClientMount",
"Principal" : { "AWS": "ReadOnlyLambda" }
```




```
"Effect" : "allow",
"Action" : ["elasticfilesystem:ClientMount",
            "elasticfilesystem:ClientWrite"],
"Principal" : { "AWS": "ReadWriteLambda" }
```

Understanding function identity



Amazon EFS access points

Amazon EFS access point

```
{
  “Name”: “MyApp”,
  "FileSystemId": ”fs-feedfeed",
  “PosixUser”: {
    “Uid”: 123
    “Gid”: 123,
    “SecondaryGids”: [100, 200, 300]
  },
  “RootDirectory”: {
    “Path”: “/apps/myapp”,
    “CreationInfo”: {
      “OwnerUid”: 123,
      “OwnerGid”: 123,
      “Permissions”: “0700”
    }
  }
}
```

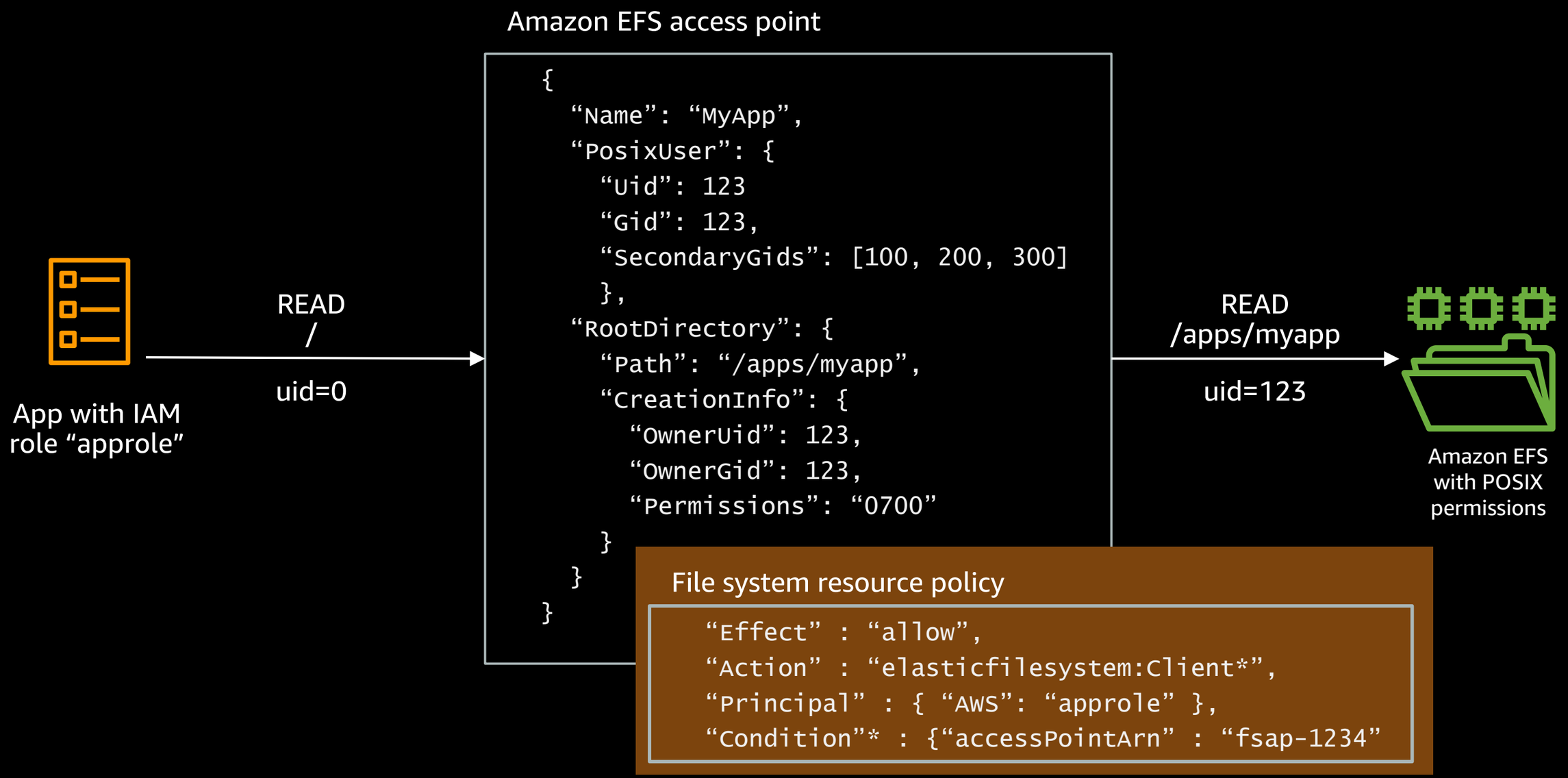
Enforces file system identity

- Root containers cannot escalate access
- Arbitrary users are not locked out

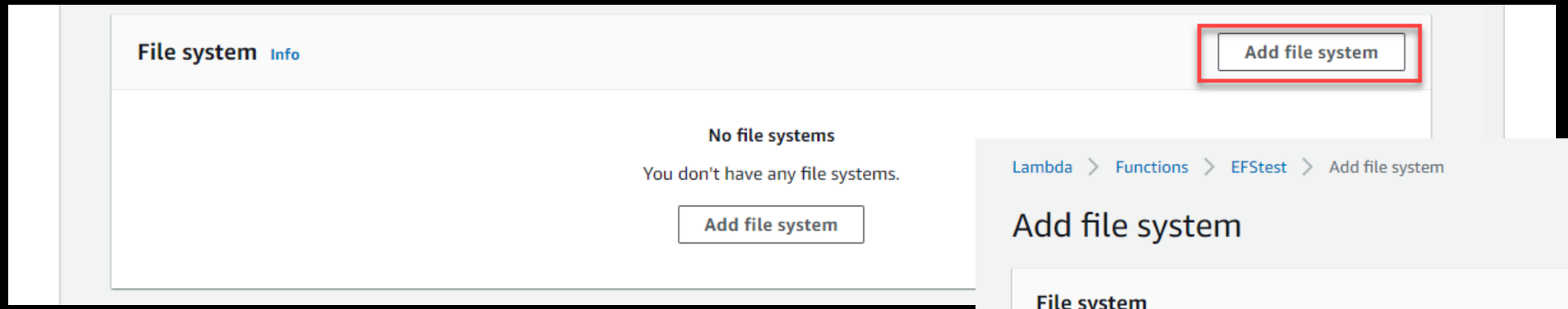
Creates app-specific directory and permissions

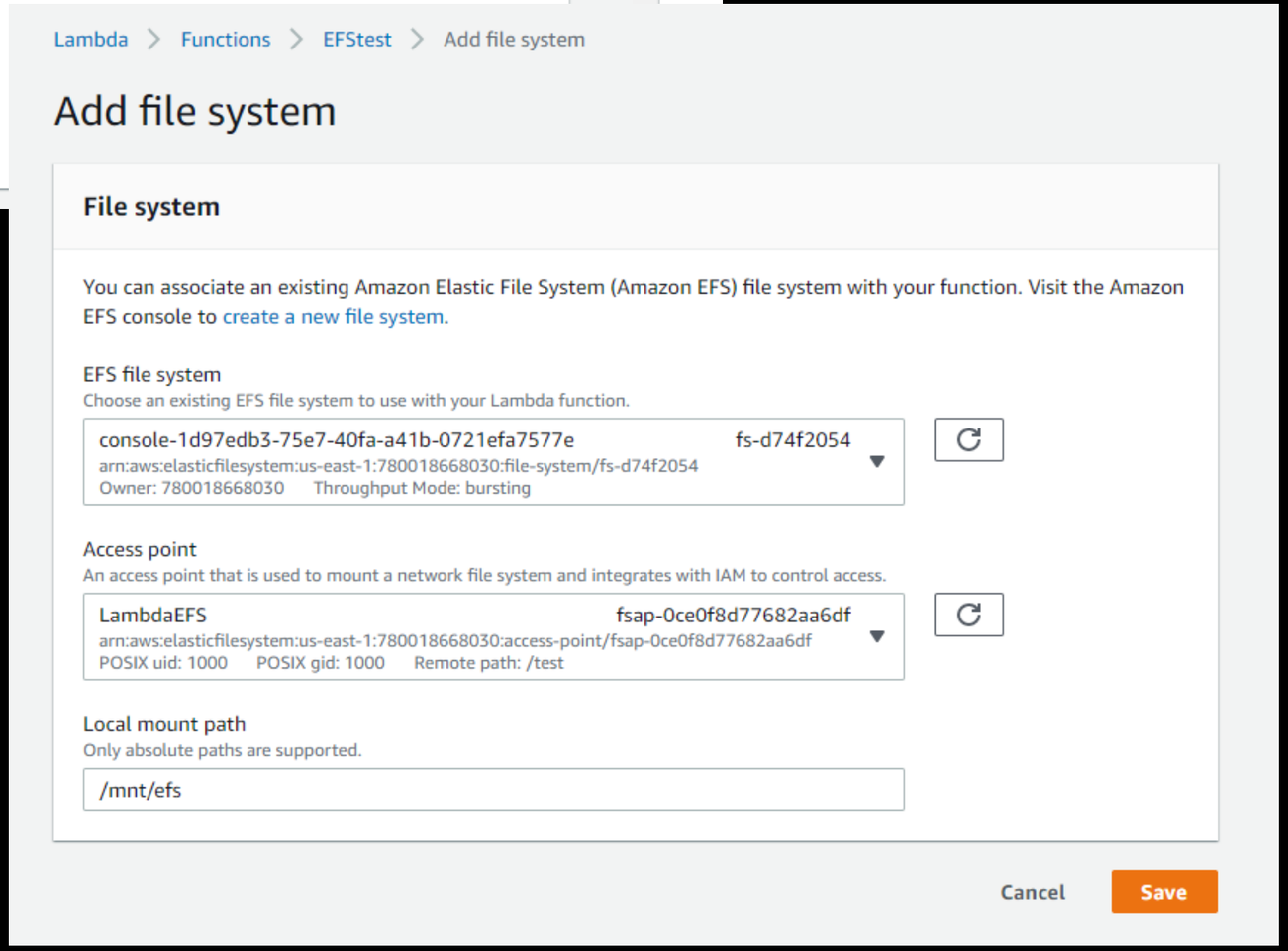
- No EC2 instance required
- Apps only see data they need

How Amazon EFS access points work



Attaching Amazon EFS to a function





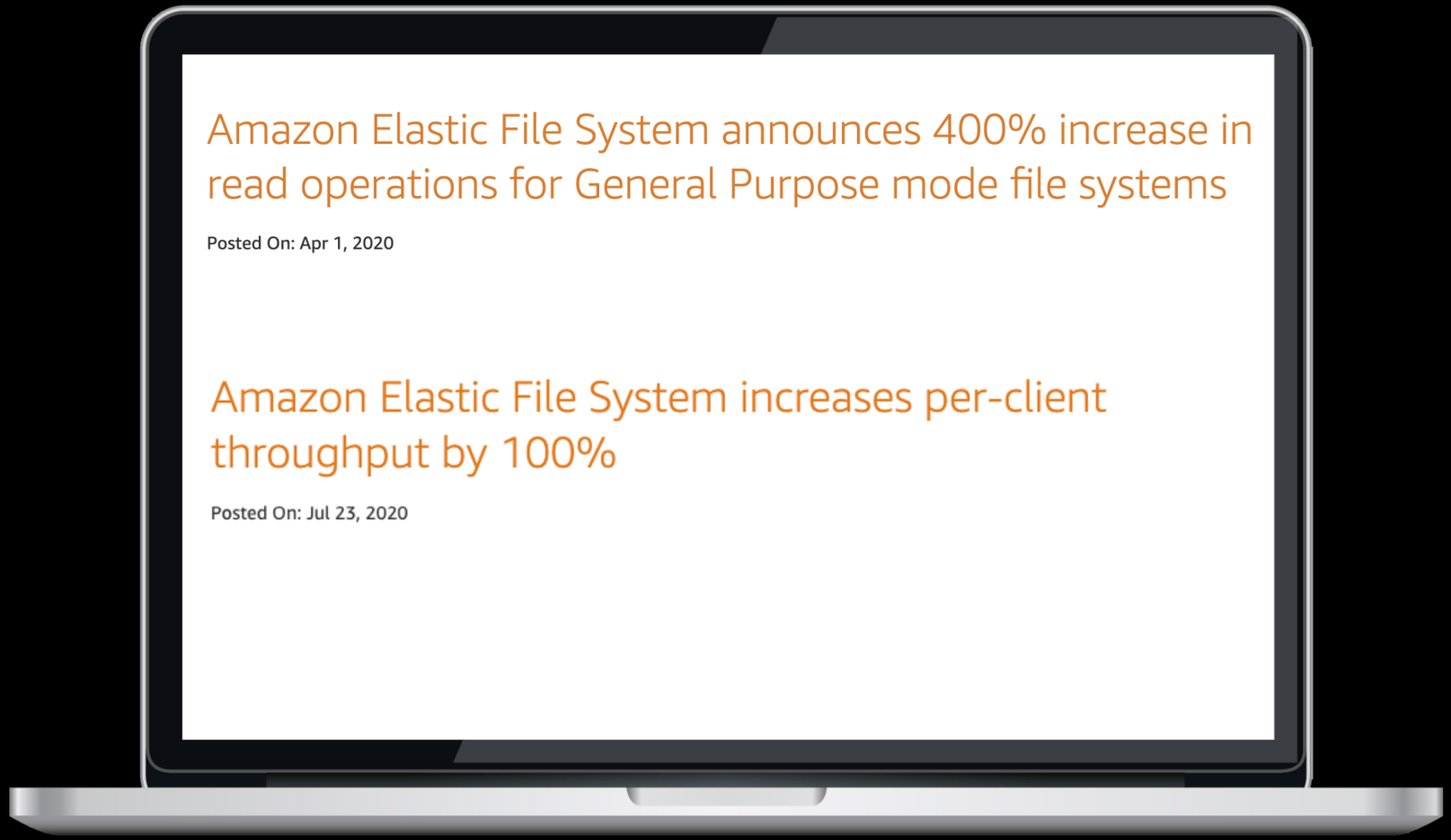
1. Create
    1. File system
    2. Mount targets
    3. Access point
2. Configure function for VPC of file system
3. Add file system
    1. Select file system
    2. Select access point

Best practices for security

- Enable encryption at rest – simple setup, no performance penalty
- Use Amazon EFS access points (default in Lambda) – do not leave UID and GID blank!
- Enable encryption in motion (default in Lambda)
- Use IAM authorization
  - Use resource policies to restrict IAM roles to Amazon EFS access points
  - Use identity policies to give single role “admin” access to file systems
  - Simple setup, no performance penalty

Best practices – Performance

- Use General Purpose (GP) for most applications
  - GP lower latency, now supports up to 35K read IOPS
  - Max I/O for scale-out analytics / ML that need 100K+ IOPS
- Configure provisioned throughput for initial need – as file system grows, you will eventually be given higher throughput
- Set up Amazon CloudWatch; monitor throughput, IOPS, and burst credits*



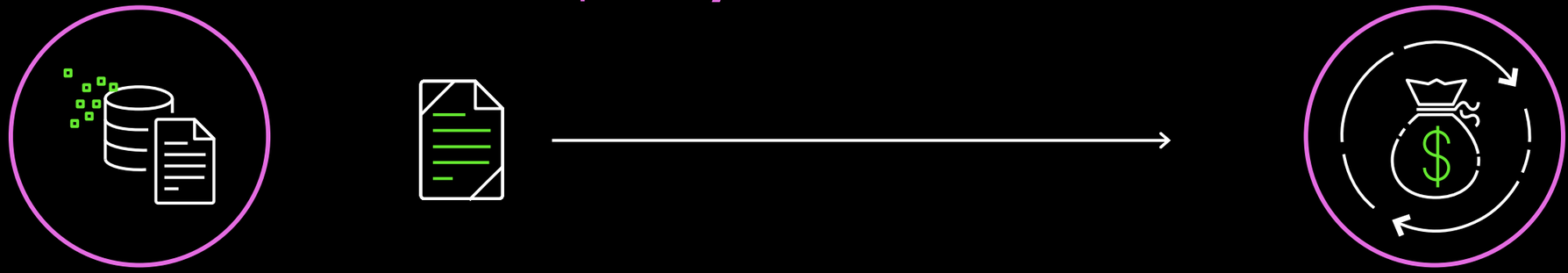
*github.com/aws-samples/amazon-efs-tutorial/tree/master/monitoring

Best practices – Performance and concurrency

- Be mindful of Lambda concurrency – each concurrent invocation gets 1/N of file system throughput and IOPS
- Cold starts and warm starts
  - Cold start – request comes in and no available containers, so new container created
  - Warm start – request can be fulfilled by existing container
- Client-side caching is your friend
  - Warm starts fetch already-accessed data out of container cache (great for ML models and other reference data)
  - Cold starts always require a new download, so be mindful of concurrent cold-starts
- For batch jobs, it is sometimes better to have lower concurrency to ensure container reuse for caching, rather than having 100% cold start

Optimize cost with Amazon EFS Infrequent Access (IA)

Pay-as-you-go, effective storage cost of
$0.08/GB-MONTH*



Standard storage class

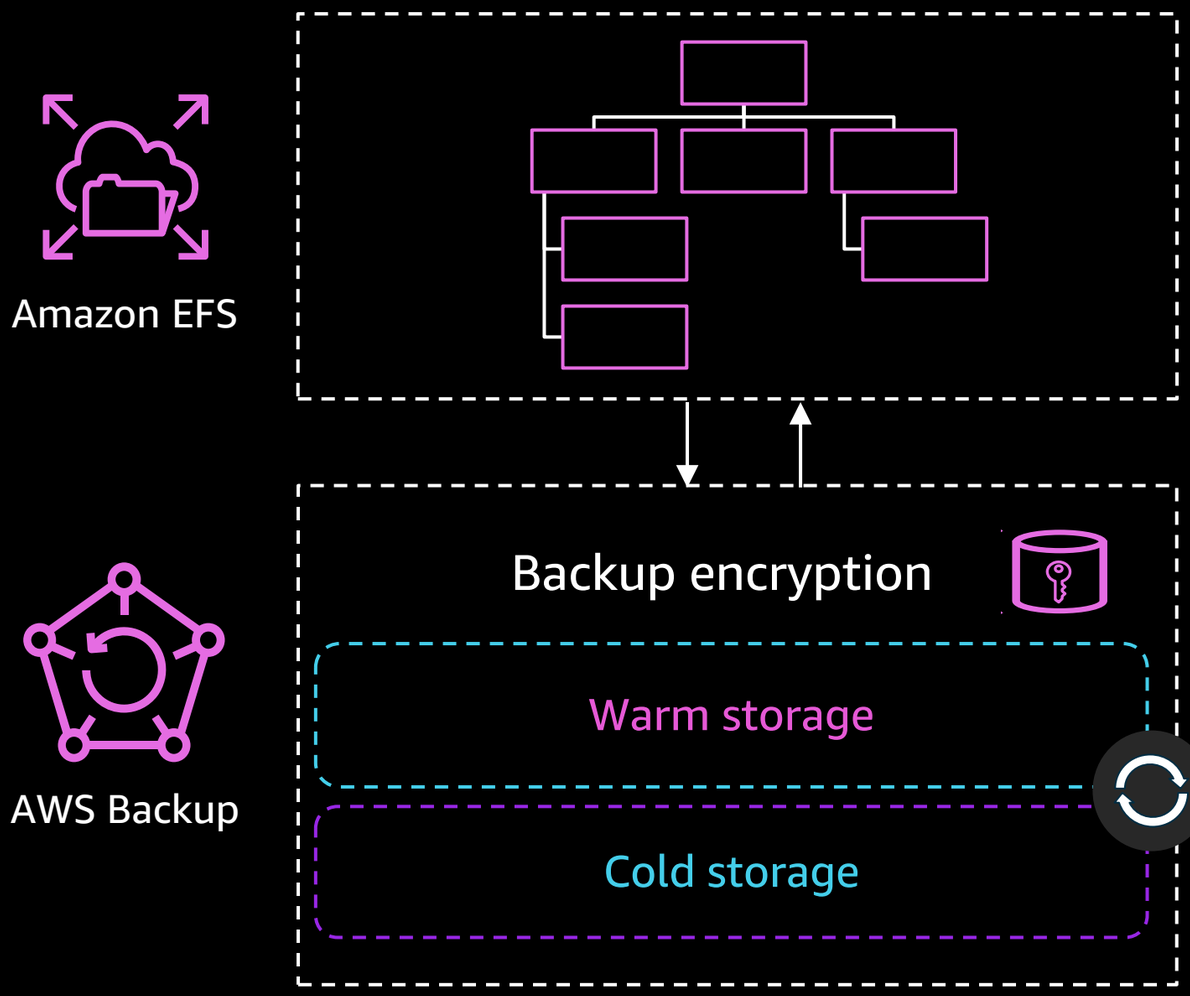
General-purpose file storage
$0.30/GB-month*

Infrequent Access storage class

Cost-optimized for files not accessed every day
$0.025/GB-month* for storage
$0.01/GB* for access

*Pricing in the US East (N. Virginia) region. Assumes 80% of the files are infrequently accessed

AWS Backup for Amazon EFS



- Amazon EFS file systems can be backed up and restored using AWS Backup
- AWS Backup provides automated backup scheduling and retention per user-defined policy
- AWS Backup offers two classes of service backup storage with ability to lifecycle to cold storage
- Restore individual files and directories

Case study – Asurion



© 2020, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Who is Asurion?

We help people protect, connect, and enjoy all their tech. Examples:

- Fixing cracked screens
- Replacing lost phones
- **Resolving** video streaming **issues**



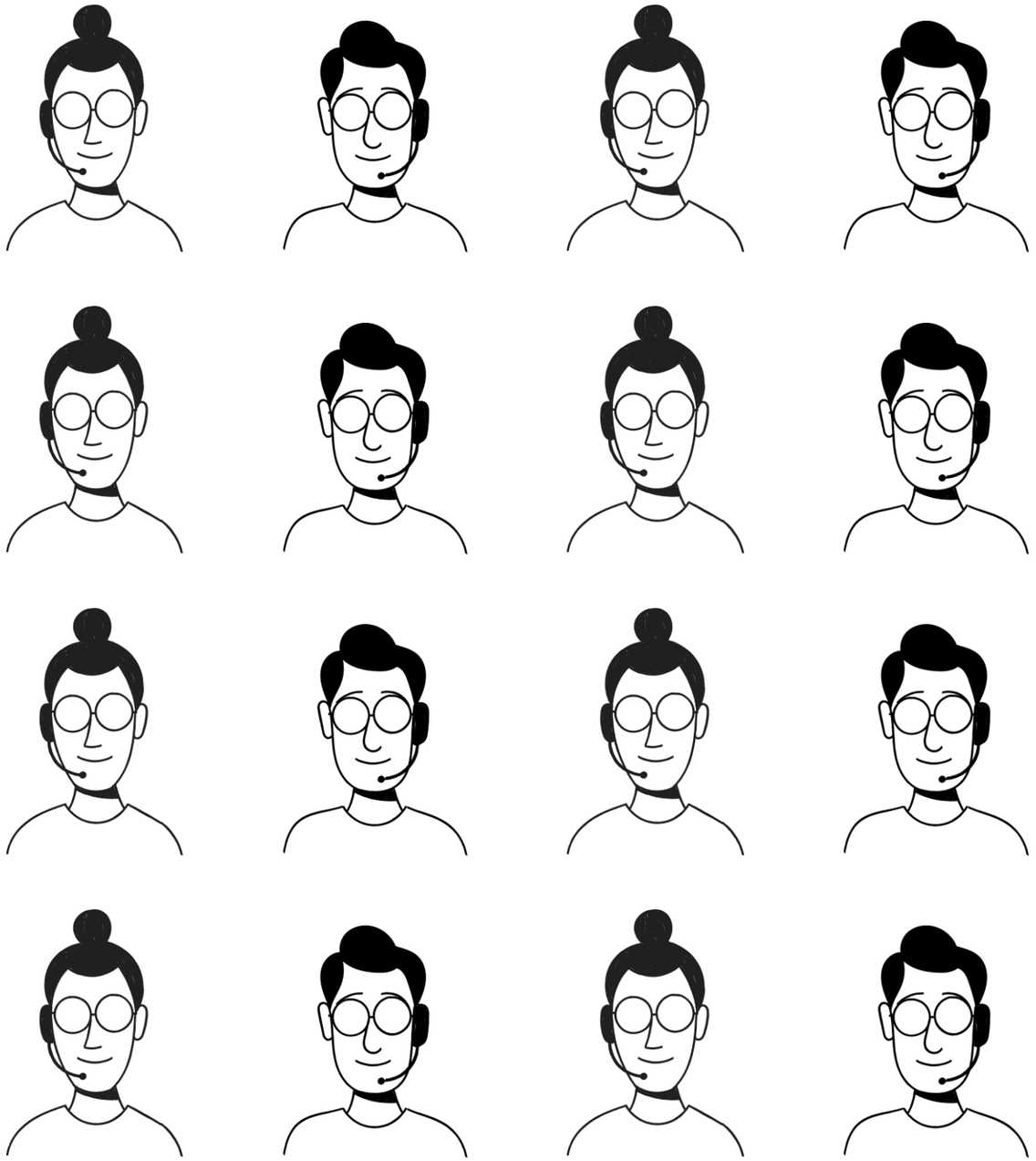
asurion

Jeff Tougas, Director, Software Engineering

“Your call may be recorded for training purposes . . .”

ML-powered quality control

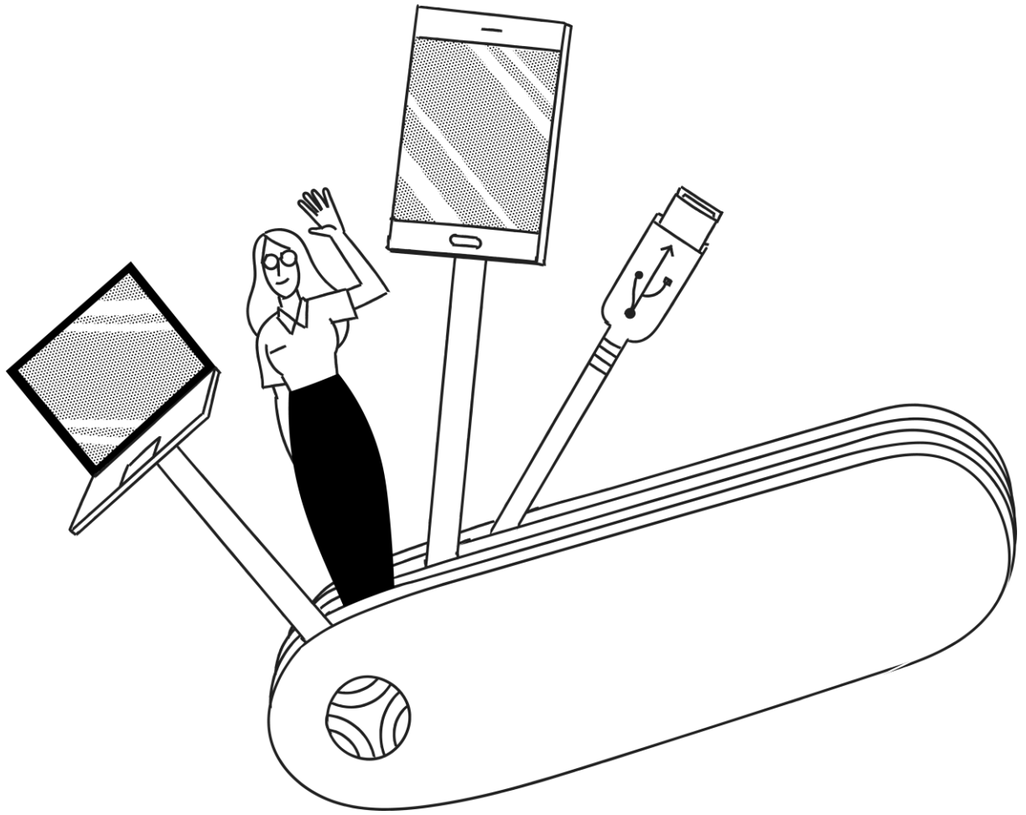
Achieving real-time analysis of customer experience during support calls using Lambda + Amazon EFS



Machine learning pipeline

Event-driven ML model execution platform

- **Serverless**
- **Config based for onboarding models**
- **Real-time, highly scalable, and performant**
- **Dozens of models (so far)**



Challenge



Analyze Asurion expert's enthusiasm from call recording

- WAV downloaded from Amazon S3
- Transcoded using ffmpeg
- Extract expert's portions of the WAV
- Phonetic speech analysis run on expert's WAV files
- Use ML to rate expert's enthusiasm based on scores compared to model

Custom, large dependencies, plus big WAV files, plus long runtime – viable for Lambda? Lets find out!

Challenge



Large Lambda layer size (375MB total)

- **ffmpeg (70MB)**
- **Python libs (180MB)**
- **R runtime + libs (40MB)**
- **Speech analysis libs (80MB)**

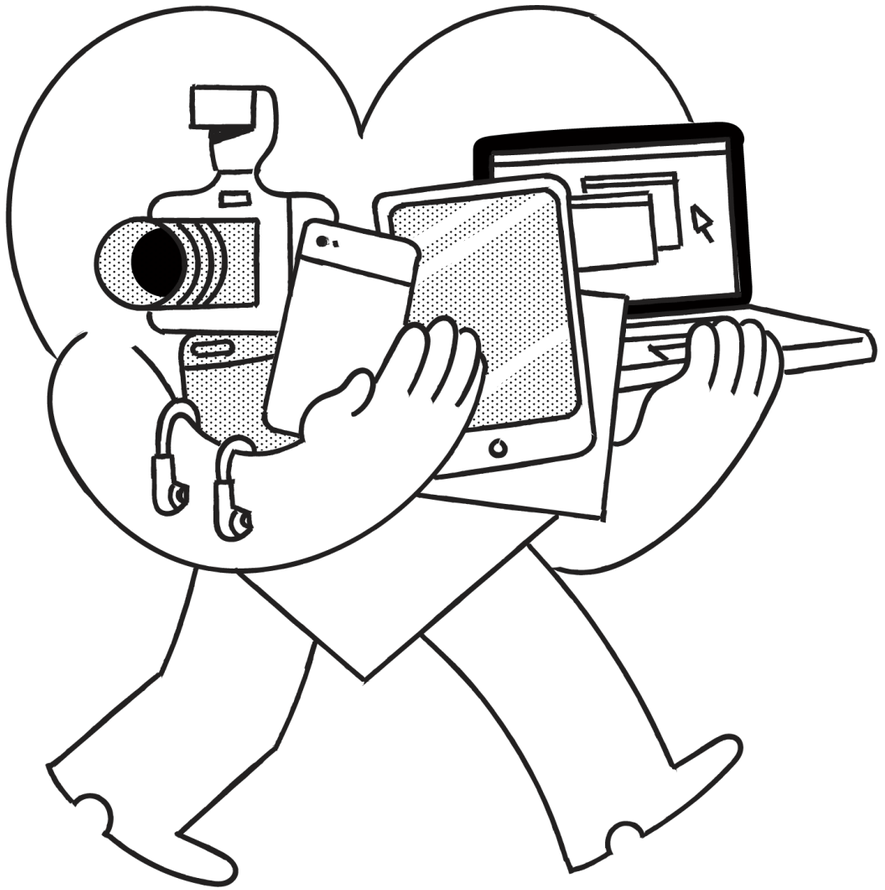
Large WAV files being processed

- **Upwards of 100MB for source WAV**
- **ffmpeg transcoding writes new 2x size WAV**
- **R writes additional WAV files for expert's speech segments**

Solution: Lambda + Amazon EFS!

Use Amazon EFS to provide scalable storage

- Amazon EFS serving temporary file processing needs
- Next up: Use Amazon EFS for package dependency and model loading also



Amazon EFS sessions

STG213 What's new with Amazon EFS

STG311 Optimizing Amazon EFS for cost and performance, using best practices

STG307 Deep dive on Amazon EFS

STG310 Modernize your applications with AWS Lambda and Amazon EFS

STG312 Modernize your applications with containers using Amazon EFS

STG306 Security best practices with Amazon EFS

STG313 Reduce TCO for your Linux file-based applications

STG314 Amazon EFS performance best practices for Linux workloads

Thank you!



AWS re:Invent

© 2020, Amazon Web Services, Inc. or its affiliates. All rights reserved.



Please complete the session survey

AWS re:Invent

© 2020, Amazon Web Services, Inc. or its affiliates. All rights reserved.