INNOVATE

MODERN APPLICATIONS EDITION

27&28 October 2021

Persistent storage on containers using Amazon EFS

Manikandan Chandrasekaran

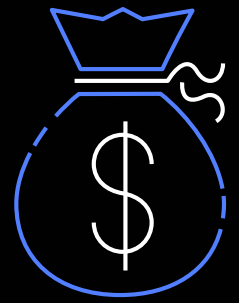
Principal Solutions Architect
AISPL

Why persistent storage for containers?

Motivation to modernize applications with containers

1. Increased agility and scalability

2. Faster time to market



3. Improved reliability, simpler operations, and lower cost

Applications that need persistent storage

Long-running stateful applications

Shared data sets

Developer tools

Jenkins
Jira
Git

Web and content management

WordPress
Drupal
nginx

Machine learning

MXNet
TensorFlow

Data science tools

JupyterHub
Airflow

Application modernization with containers & Amazon Elastic File Storage (Amazon EFS)

Modern compute spectrum

More

Opinionated

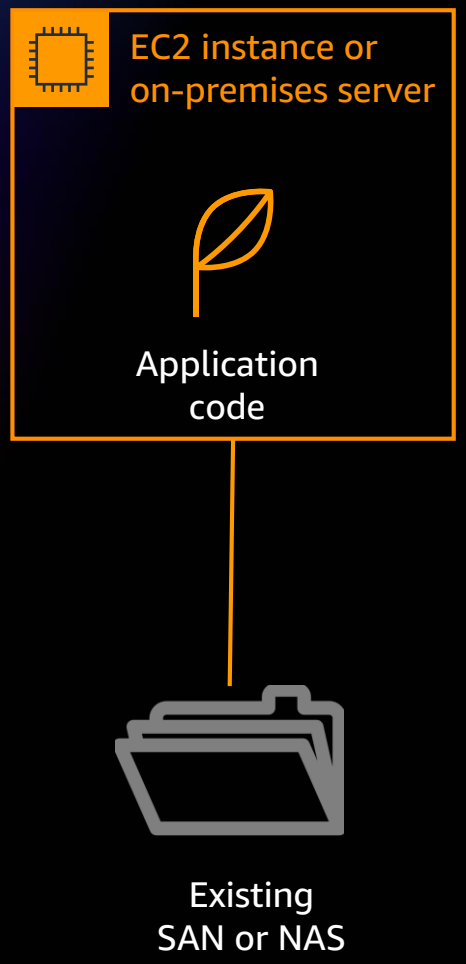
Less

	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 config and updates	Network config 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 config and updates, network config, firewall, and management tasks
Amazon Elastic Compute Cloud (Amazon EC2) Infrastructure as a Service	Physical hardware, software, networking, and facilities	Application code Data source integrations Scaling Security config and updates Network config	Management tasks Provisioning, managing, scaling, and patching of servers

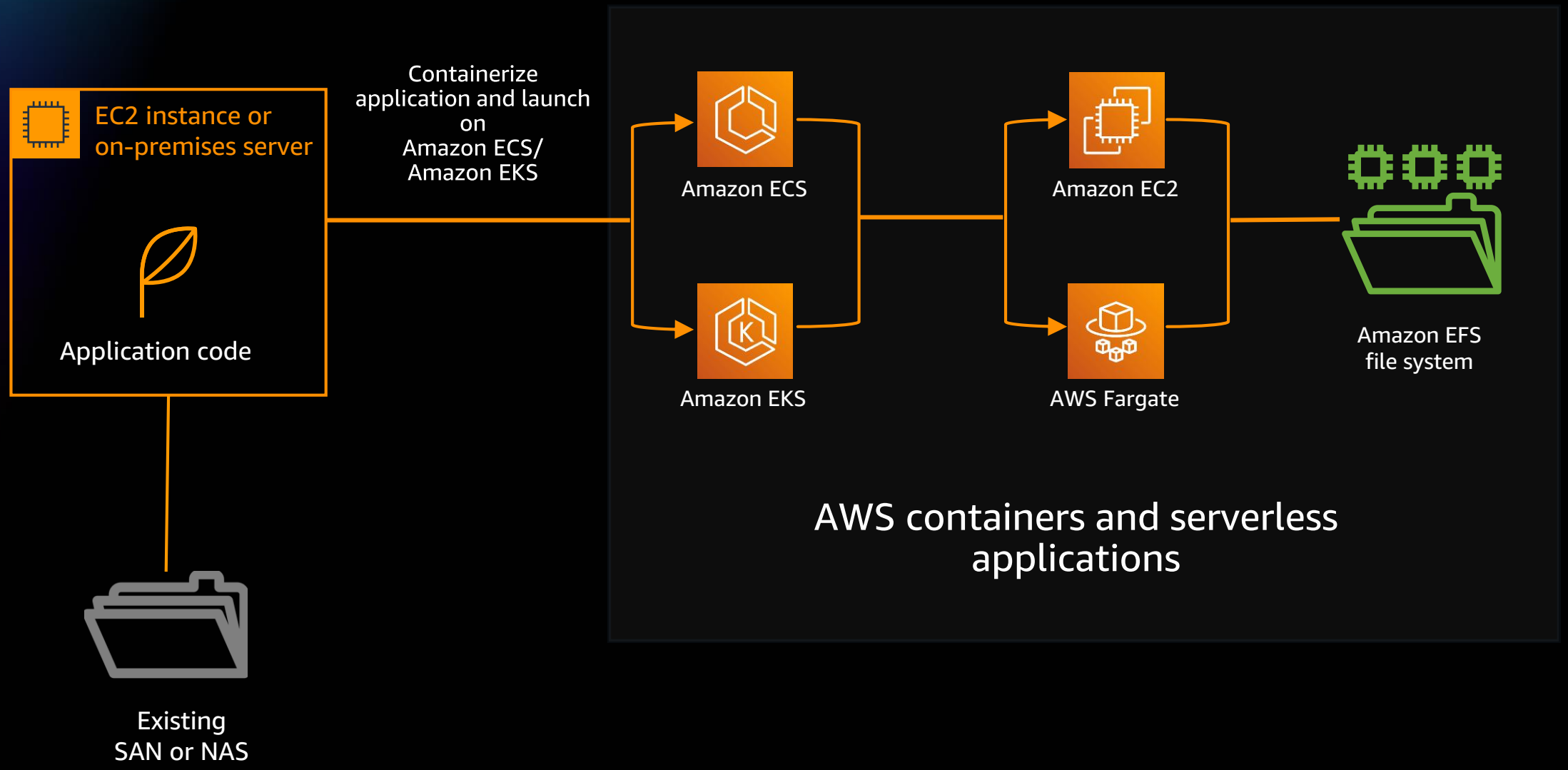
*Amazon Elastic Container Service (Amazon ECS)
*Amazon Elastic Kubernetes Service (Amazon EKS)

aws

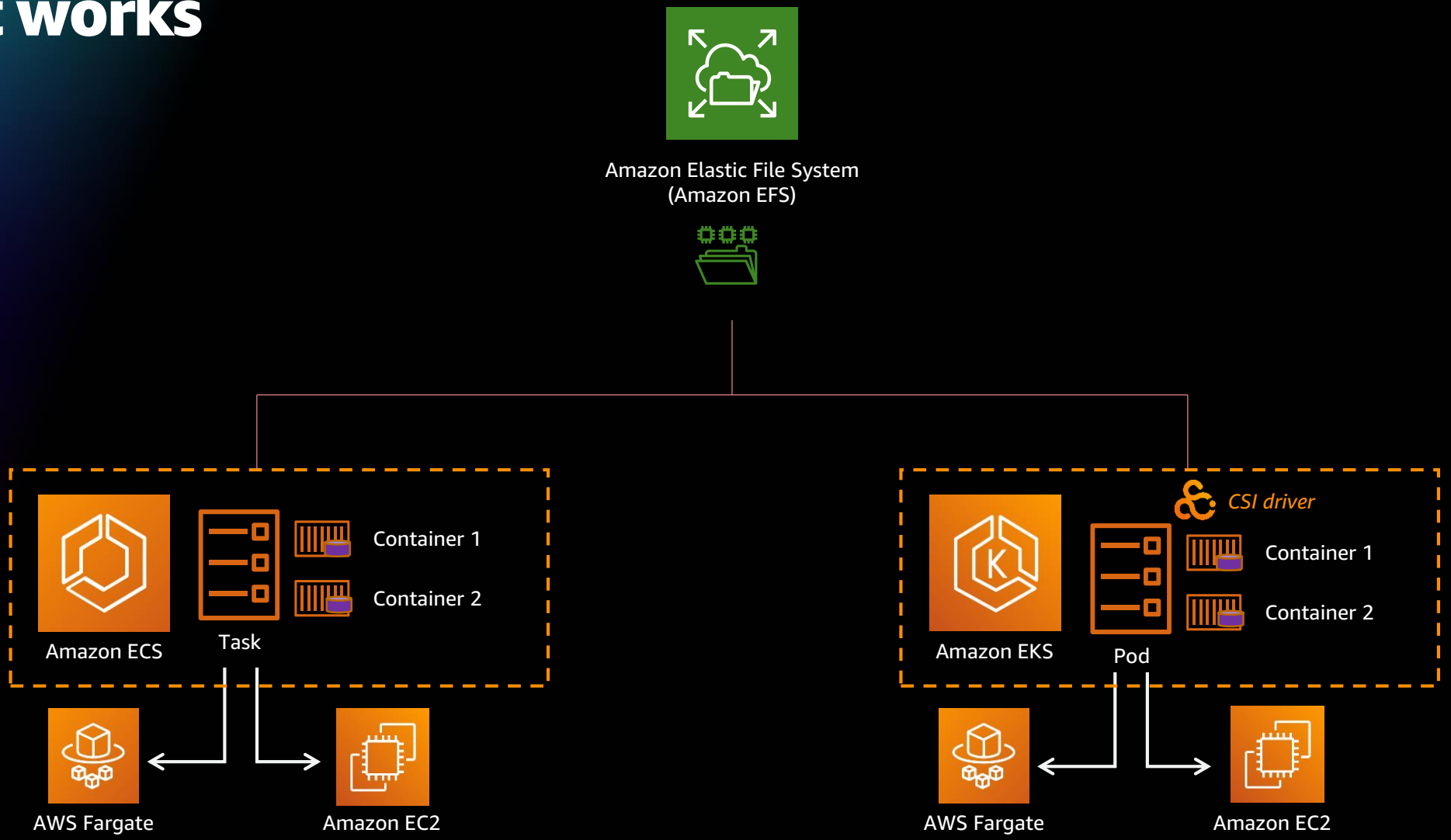
Application modernization with containers



Application modernization with containers



How it works



Amazon ECS volume definition example

```
"containerDefinitions": [
        {
. ..
"mountPoints": [
                {
                    "readOnly": null,
                    "containerPath": "/data",
                    "sourceVolume": "FargateDemoEFS"
                }
            ],
...
            "name": "FileBrowser"
        }
    ],
"taskRoleArn": "arn:aws:iam::..:role/FargateRole",
...
    "volumes": [
        {
            "efsVolumeConfiguration": {
                "transitEncryptionPort": null,
                "fileSystemId": "fs-41c7f3c1",
                "authorizationConfig": {
                    "iam": "ENABLED",
                    "accessPointId": "fsap-0f7741bf379626fc2"
                },
                "transitEncryption": "ENABLED",
                "rootDirectory": "/"
            },
            "name": "FargateDemoEFS",
```

Mount point definition

Amazon EFS volume definition

Amazon EKS volume definition example

Open source EFS CSI Driver

```yaml
apiVersion: storage.k8s.io/v1beta1
kind: CSIDriver
metadata:
  name: efs.csi.aws.com
spec:
  attachRequired: false
---
kind: StorageClass
apiVersion: storage.k8s.io/v1
metadata:
  name: efs-sc
provisioner: efs.csi.aws.com
---
apiVersion: v1
kind: PersistentVolume
metadata:
  name: wordpress-efs-pv
spec:
  capacity:
    storage: 100Gi
  volumeMode: Filesystem
  accessModes:
    - ReadWriteMany
  persistentVolumeReclaimPolicy: Retain
  storageClassName: efs-sc
  csi:
    driver: efs.csi.aws.com
    volumeHandle: $WOF_EFS_FS_ID::$WOF_EFS_AP
---
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: wordpress-efs-uploads-pvc
spec:
  accessModes:
    - ReadWriteMany
  storageClassName: efs-sc
  resources:
    requests:
      storage: 25Gi
```

Storage class

Persistent volume

Persistent volume claim

Amazon Elastic File Storage (Amazon EFS) key features

Amazon EFS highlights

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

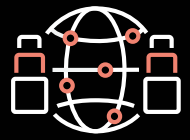
Designed for 11 9's of durability

99.99% availability SLA

Serverless shared storage

Serverless and scalable

No provisioning, scale capacity, connections, and IOPs



Concurrent access for 10,000s of connections

Amazon EC2 instances, containers, and AWS Lambda invocations

Performant and cost optimized

Performant

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

Two storage classes

Lifecycle-based cost optimization

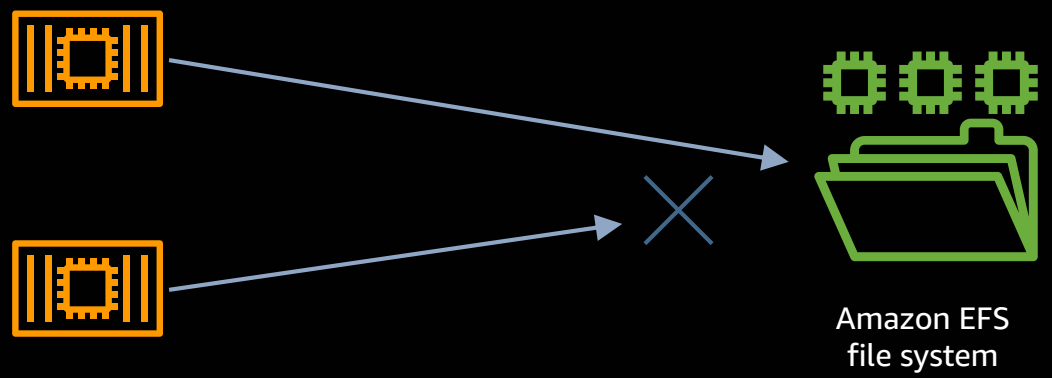
aws

Checklist

- Security and identity
- Performance optimization
- Cost optimization
- Data protection

Goals for security and identity

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

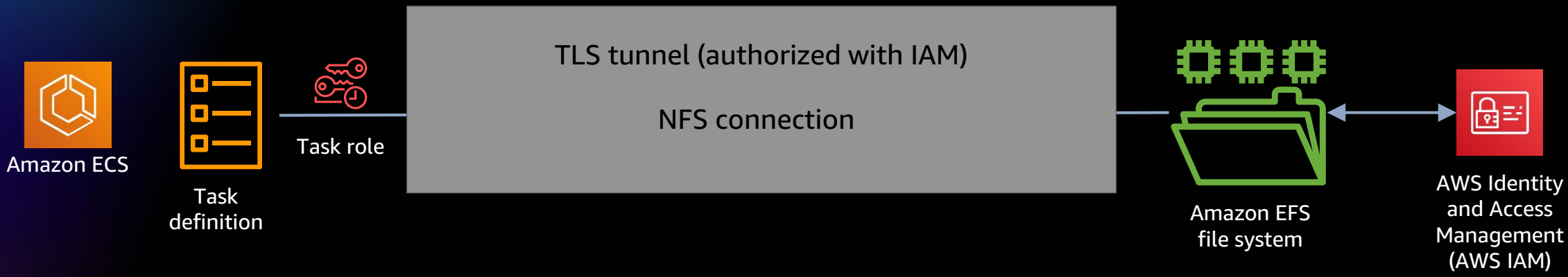



```
$ cat /my_app/data
### SUCCESS THIS IS MY FILE ###

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

Using AWS IAM for file system access

Amazon ECS ✓
Amazon EKS ✗

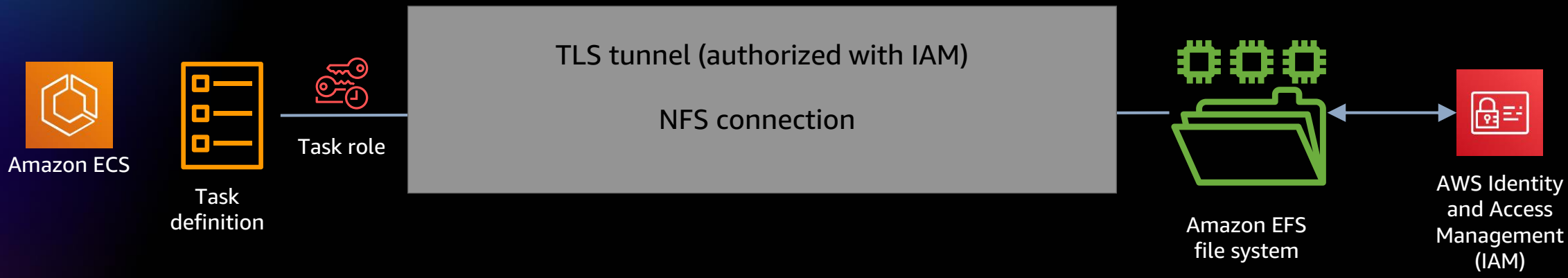


Identity policy attached to IAM role

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

Using AWS IAM for file system access

Amazon ECS ✓
Amazon EKS ✗



Identity policy attached to IAM role

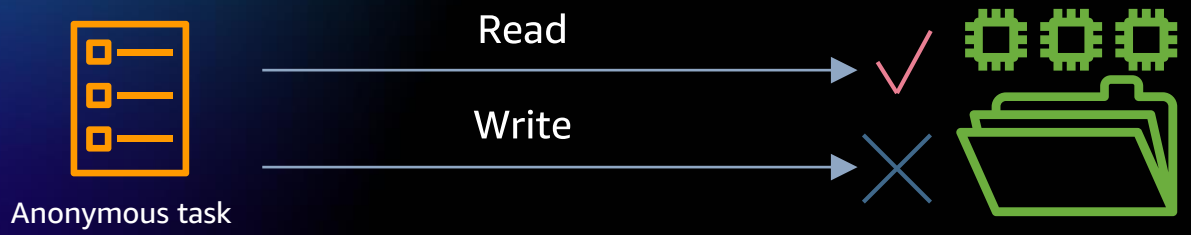
```
{
  “Statement” : {
    “Effect” : “allow”,
    “Action” : “elasticfilesystem:Client*”,
    "Resource": ”fs-feedfeed"
  }
}
```

File system resource policy

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

Handling authorization using AWS IAM

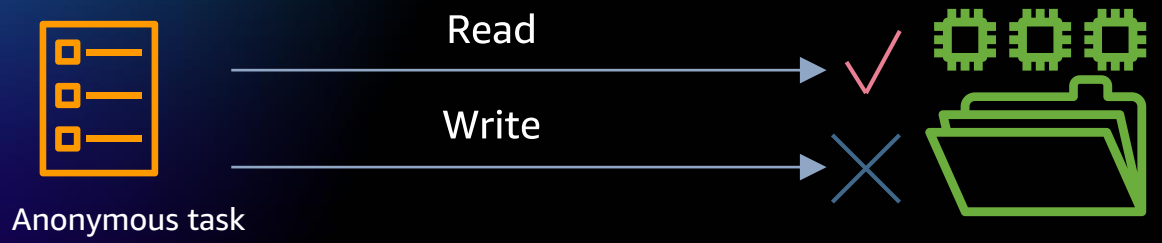
Amazon ECS ✓
Amazon EKS ✗



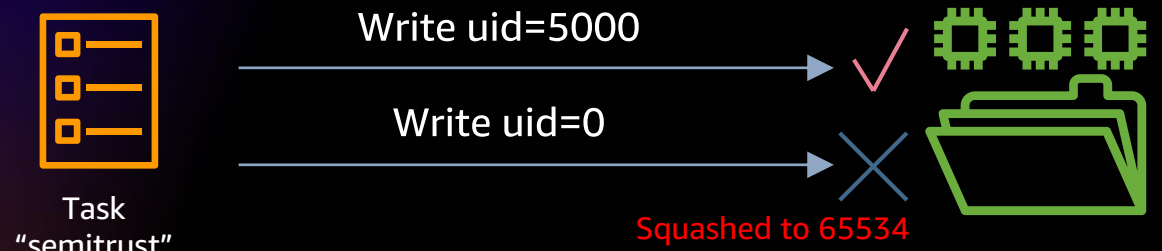

```
"Effect" : "allow",
"Action" : "elasticfilesystem:ClientMount",
"Principal" : "*"
```

Handling authorization using AWS IAM

Amazon ECS ✓
Amazon EKS ✗



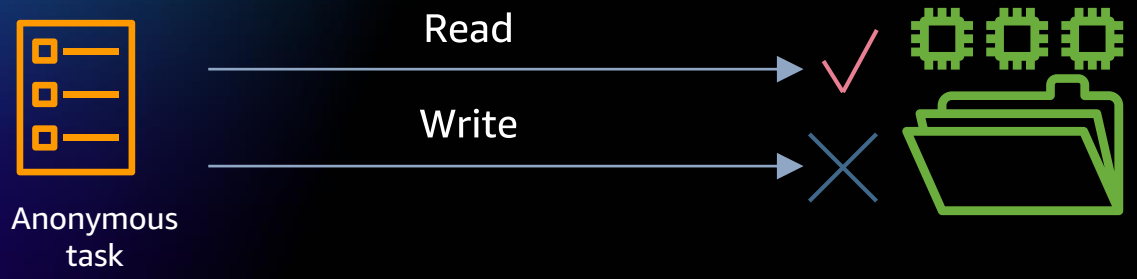

```
"Effect" : "allow",
"Action" : "elasticfilesystem:ClientMount",
"Principal" : "*"
```



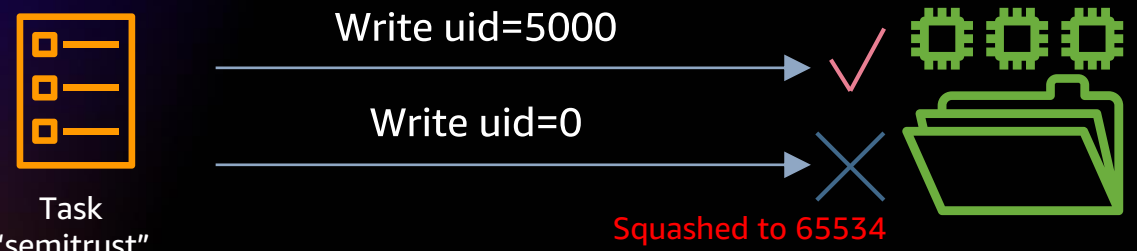

```
"Effect" : "allow",
"Action" : ["elasticfilesystem:ClientMount",
            "elasticfilesystem:ClientWrite"],
"Principal" : { "AWS": "semitrust" }
```

Handling authorization using AWS IAM

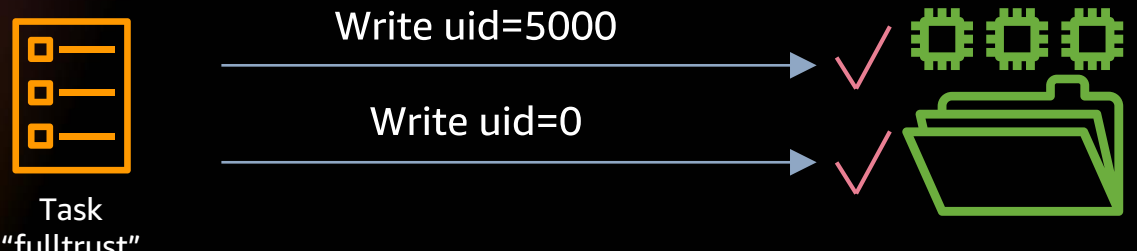
Amazon ECS	✓
Amazon EKS	✗




```
"Effect" : "allow",
"Action" : "elasticfilesystem:ClientMount",
"Principal" : "*"
```

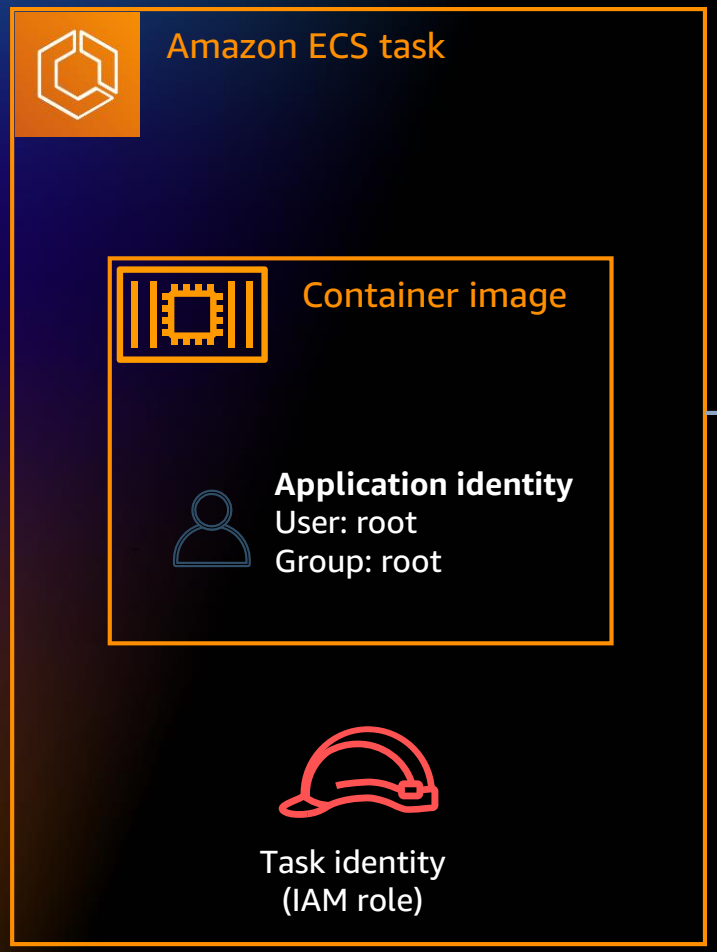



```
"Effect" : "allow",
"Action" : ["elasticfilesystem:ClientMount",
            "elasticfilesystem:ClientWrite"],
"Principal" : { "AWS": "semitrust" }
```

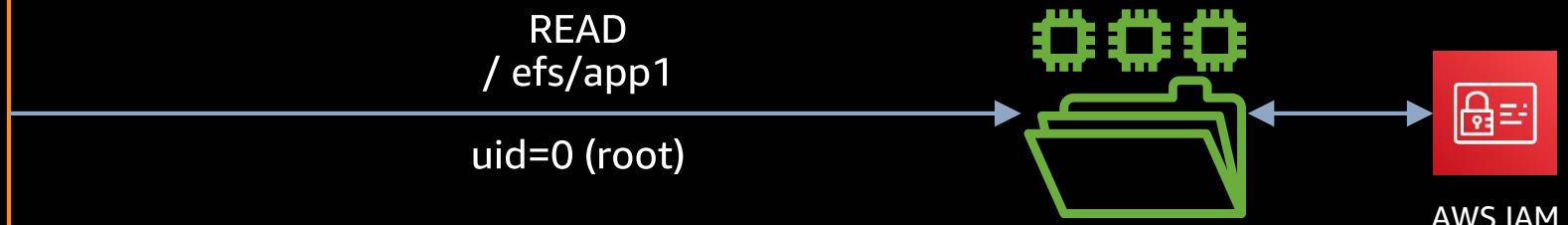



```
"Effect" : "allow",
"Action" : ["elasticfilesystem:ClientMount",
            "elasticfilesystem:ClientWrite",
            "elasticfilesystem:ClientRootAccess],
"Principal" : { "AWS": "fulltrust" }
```

Understanding container identity



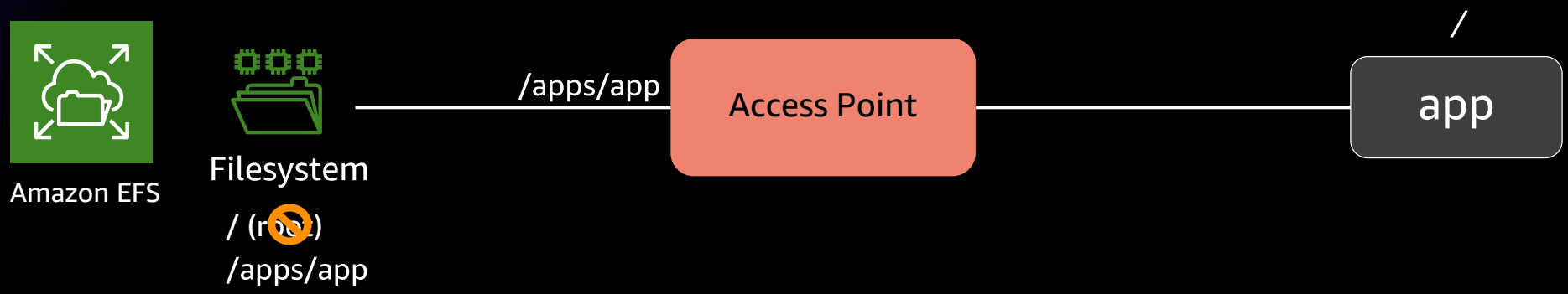
By default, POSIX identity comes from the container image, not the task/pod runtime




```
$ ls -l /efs
drwx------ user1      .          app1
drwx------ user2      .          app2
drwxrwx--- .          biusers    bi_shared
```

Amazon EFS access points

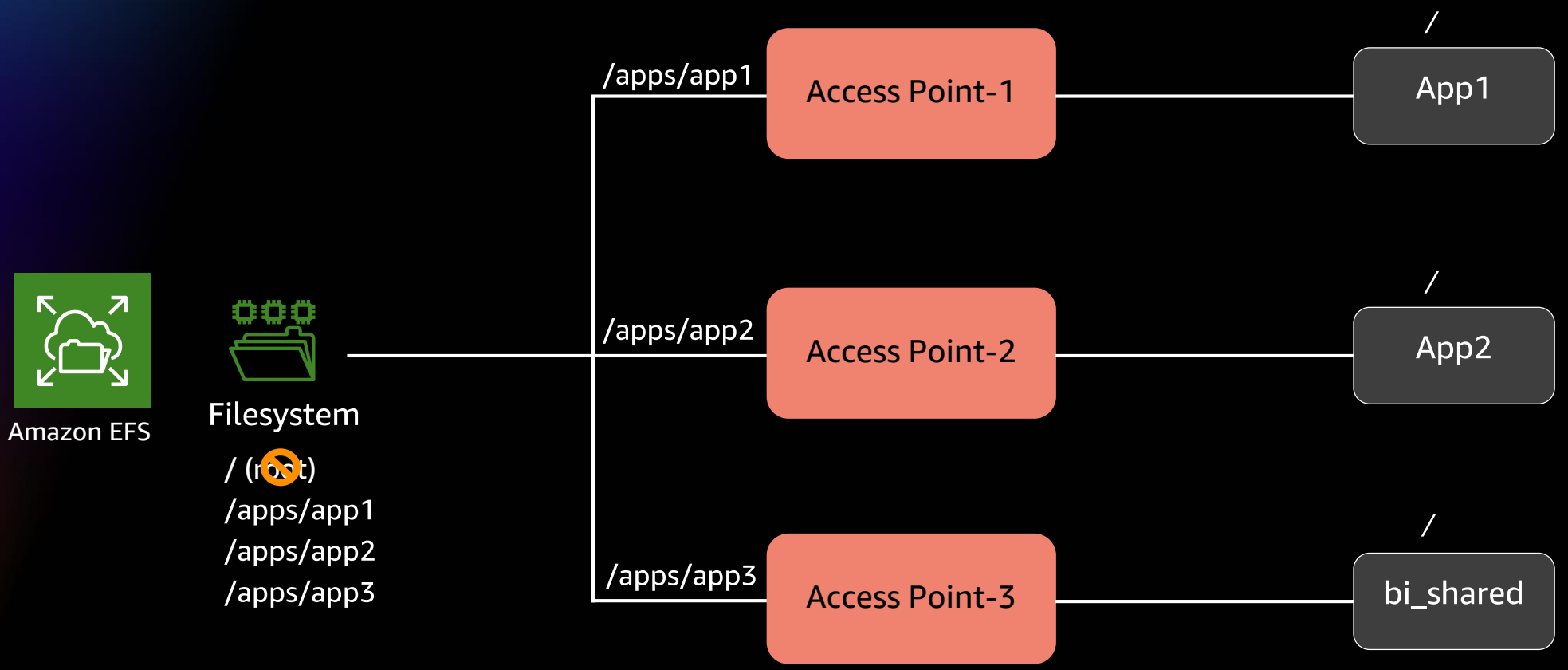
Amazon ECS ✓
Amazon EKS ✓



1. **Can enforce identity:** No root (UID=0) access please
2. **Can enforce different root directory:** You can access your directory /App

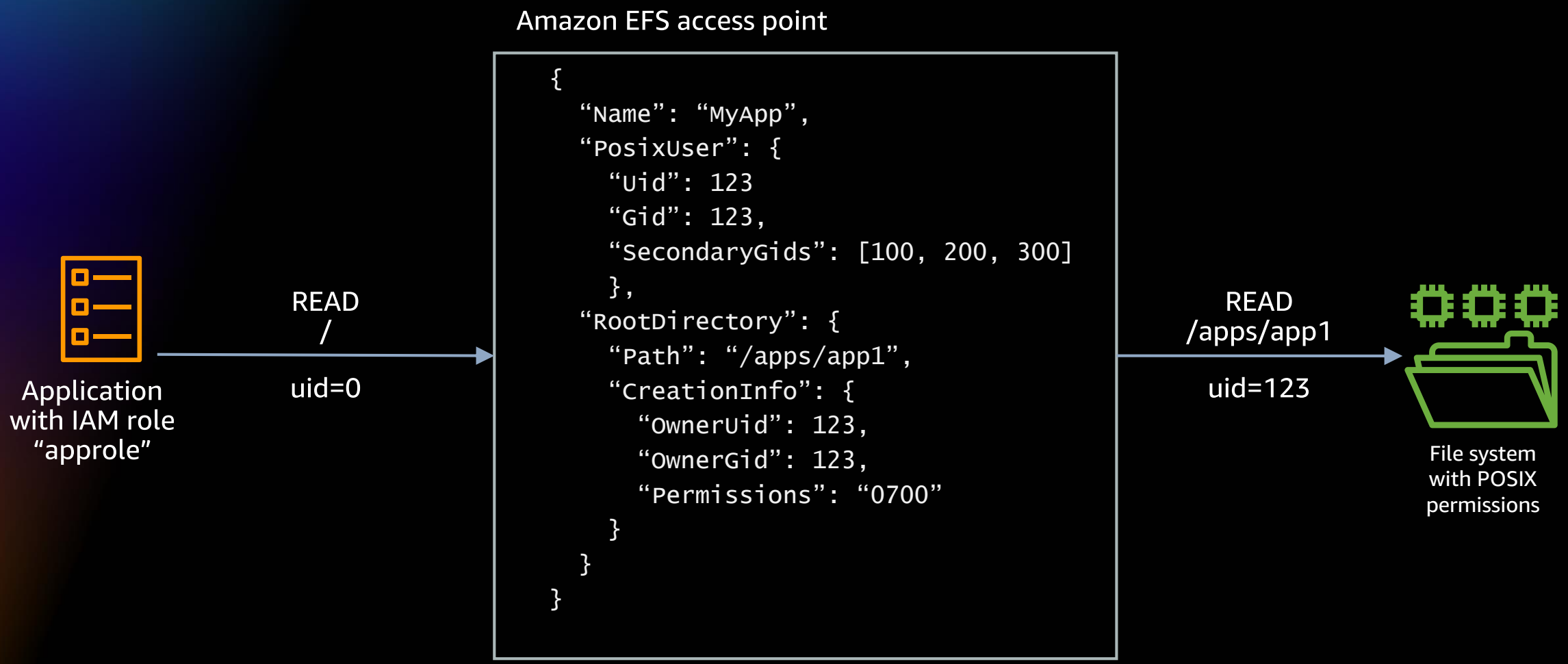
Amazon EFS access points

Amazon ECS ✓
Amazon EKS ✓



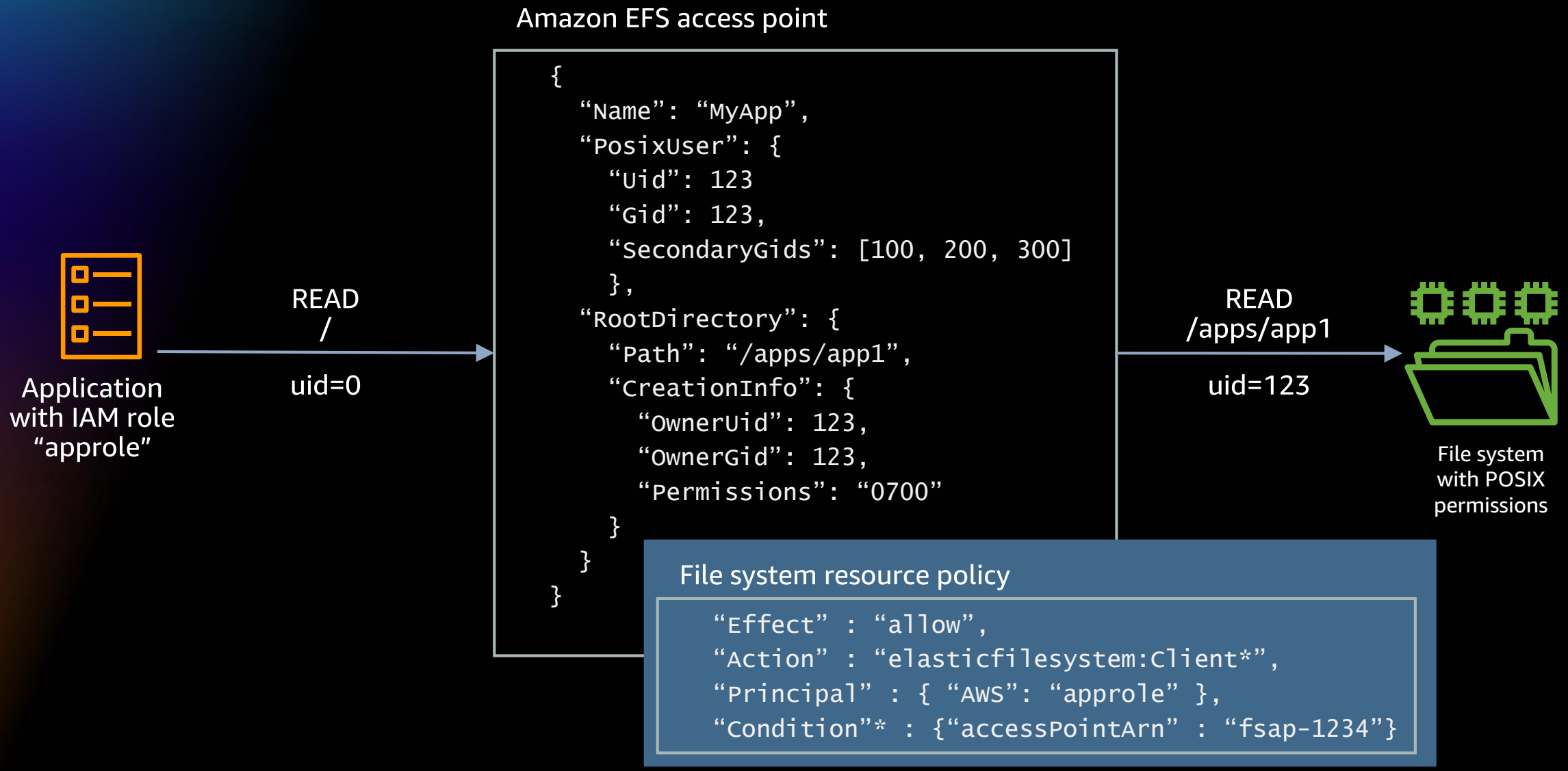
How Amazon EFS access points work

Amazon ECS ✓
Amazon EKS ✓



How Amazon EFS access points work

Amazon ECS ✓
Amazon EKS ✓

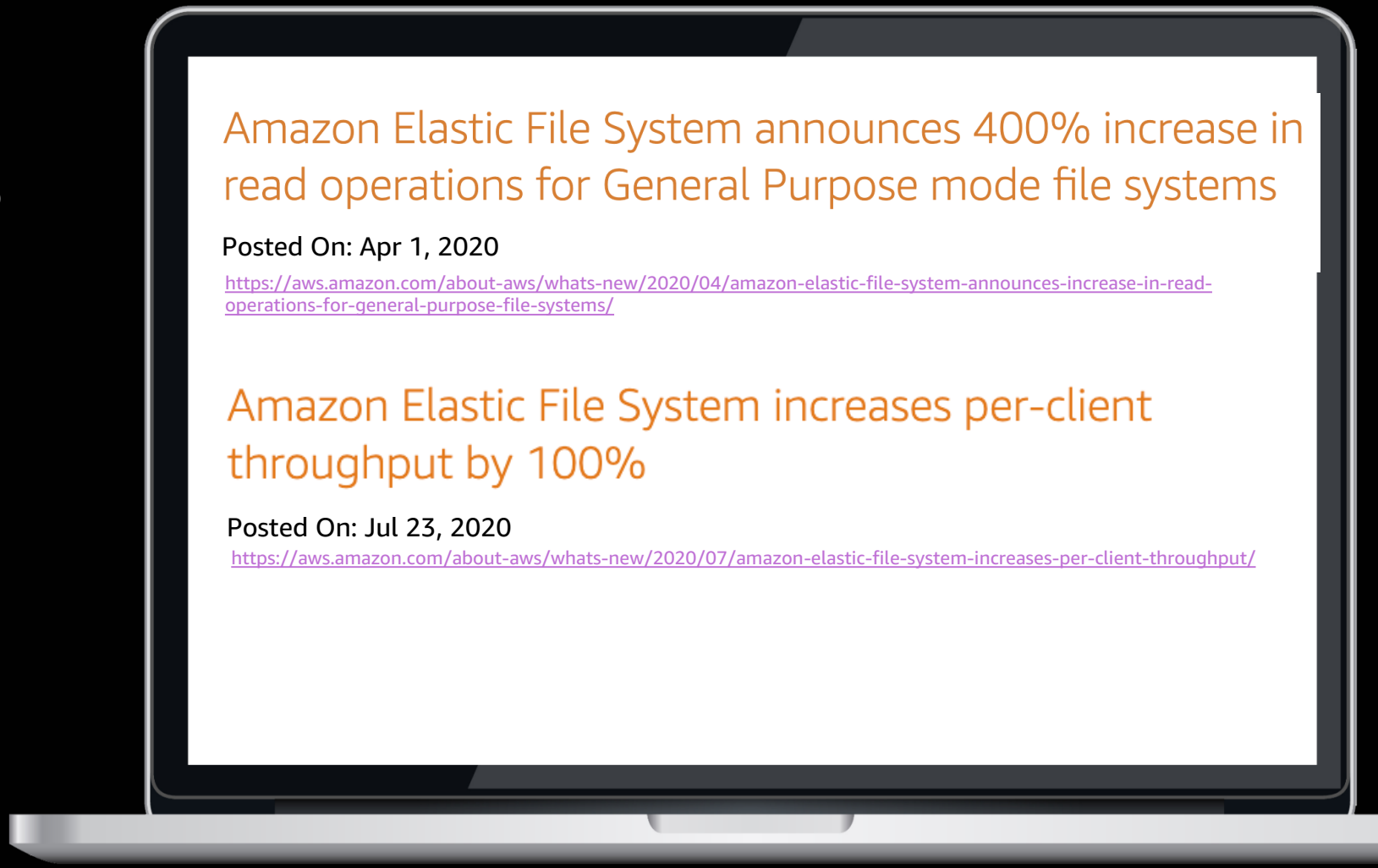


Best practices for security

- Use access points, even if single application per file system
  - Don’t leave UID / GID / RootDir blank!
- 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
- Enable encryption at rest and encryption in motion

Best practices for performance

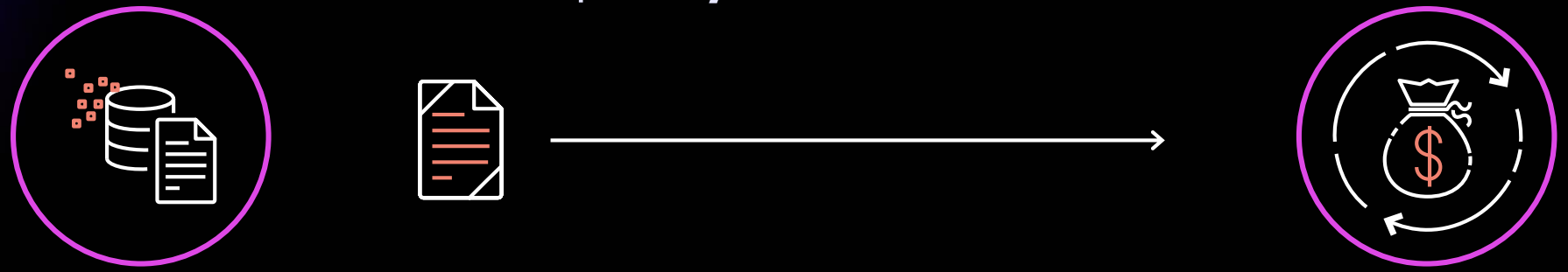
- Use General Purpose (GP) for most applications
  - GP – lower latency; now supports up to 35K read IOPS
  - Max I/O – for scale-out analytics / ML that needs 100K+ IOPS
- Configure provisioned throughput for initial need –as file system grows, you'll eventually be given higher throughput
- Set up Amazon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



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

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

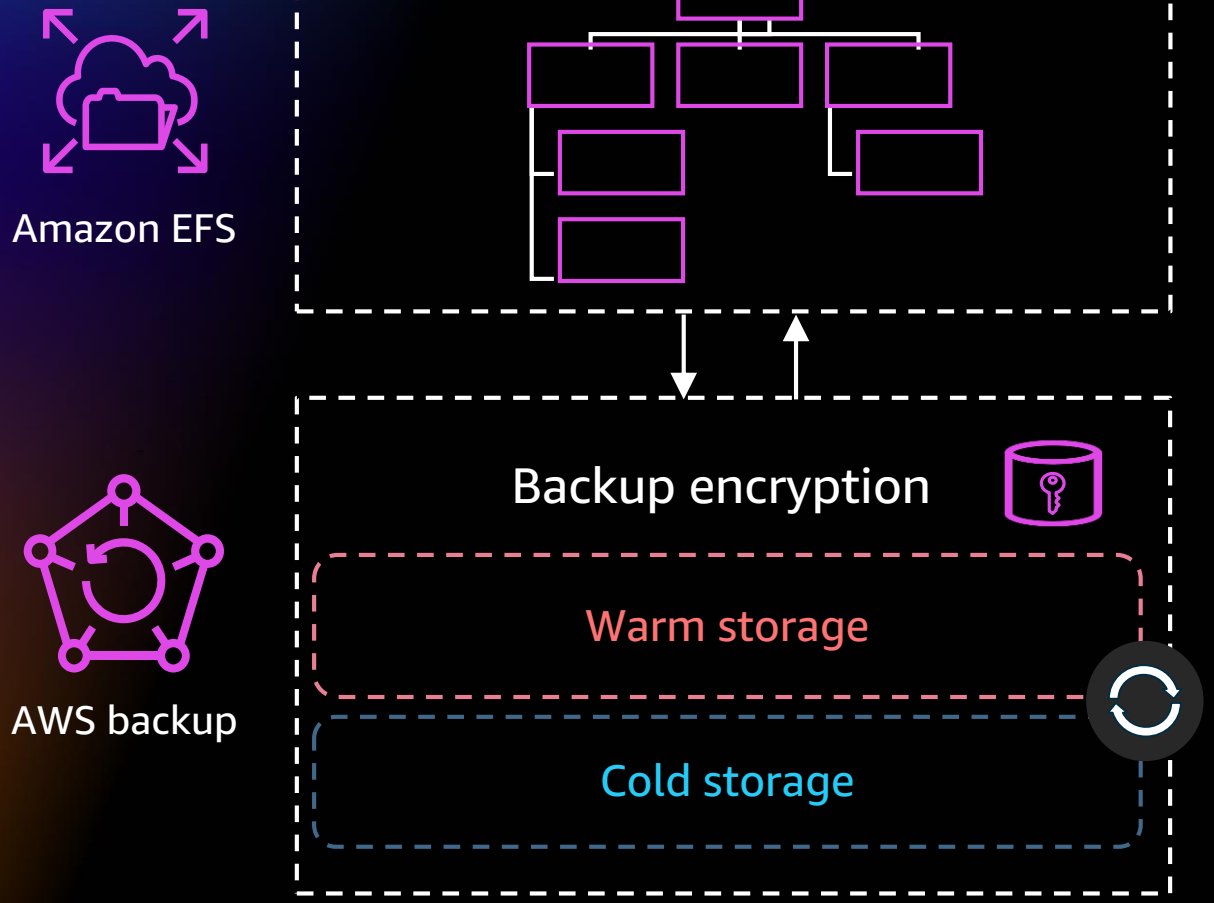
$0.025/GB-month* for storage

$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 the ability to lifecycle to cold storage
- AWS Backup restores individual files and directories

When should I use Amazon EFS vs. Amazon EBS?

Amazon EFS

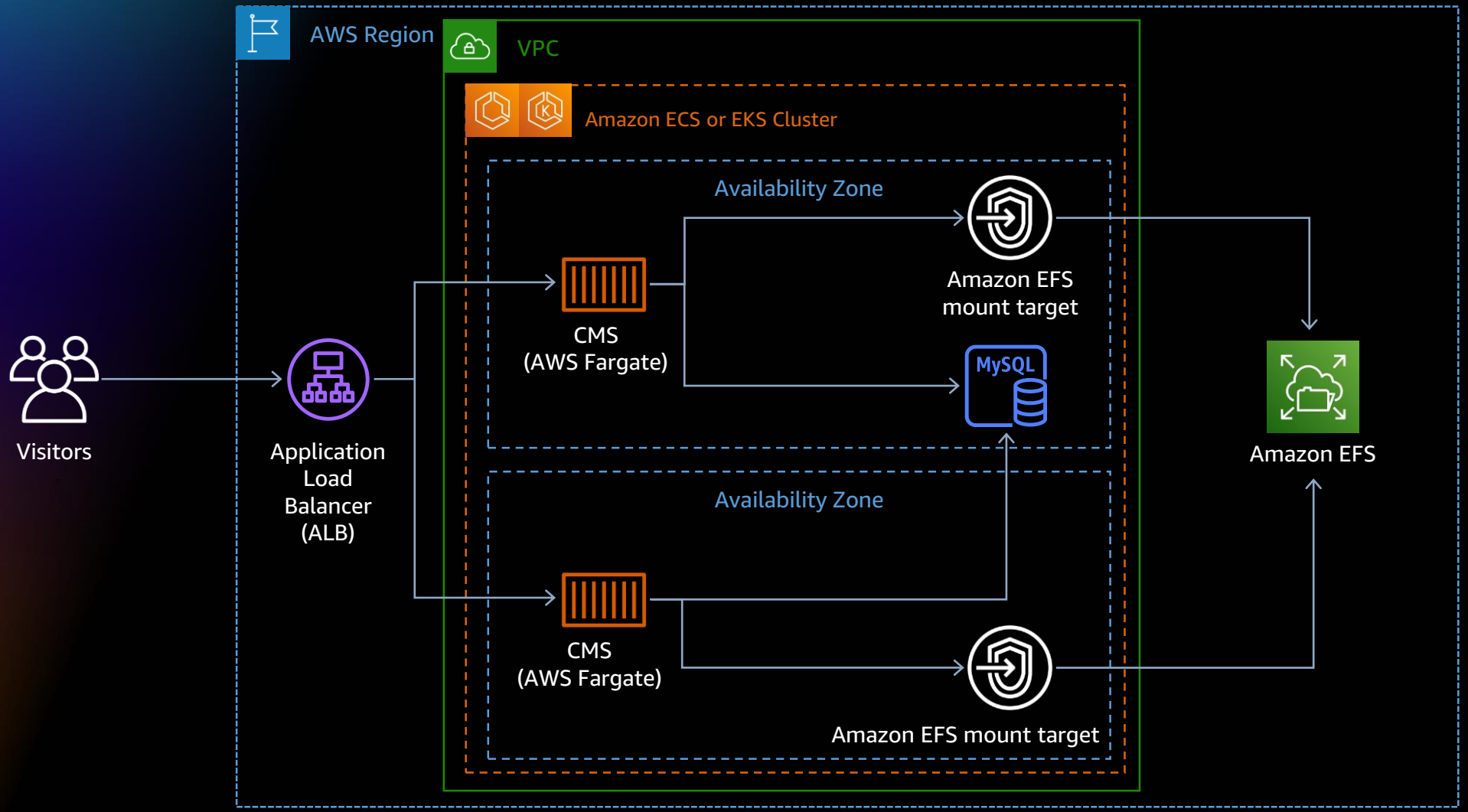
- I need to share data between containers
- I'd like to run across instances or AZs

Amazon EBS

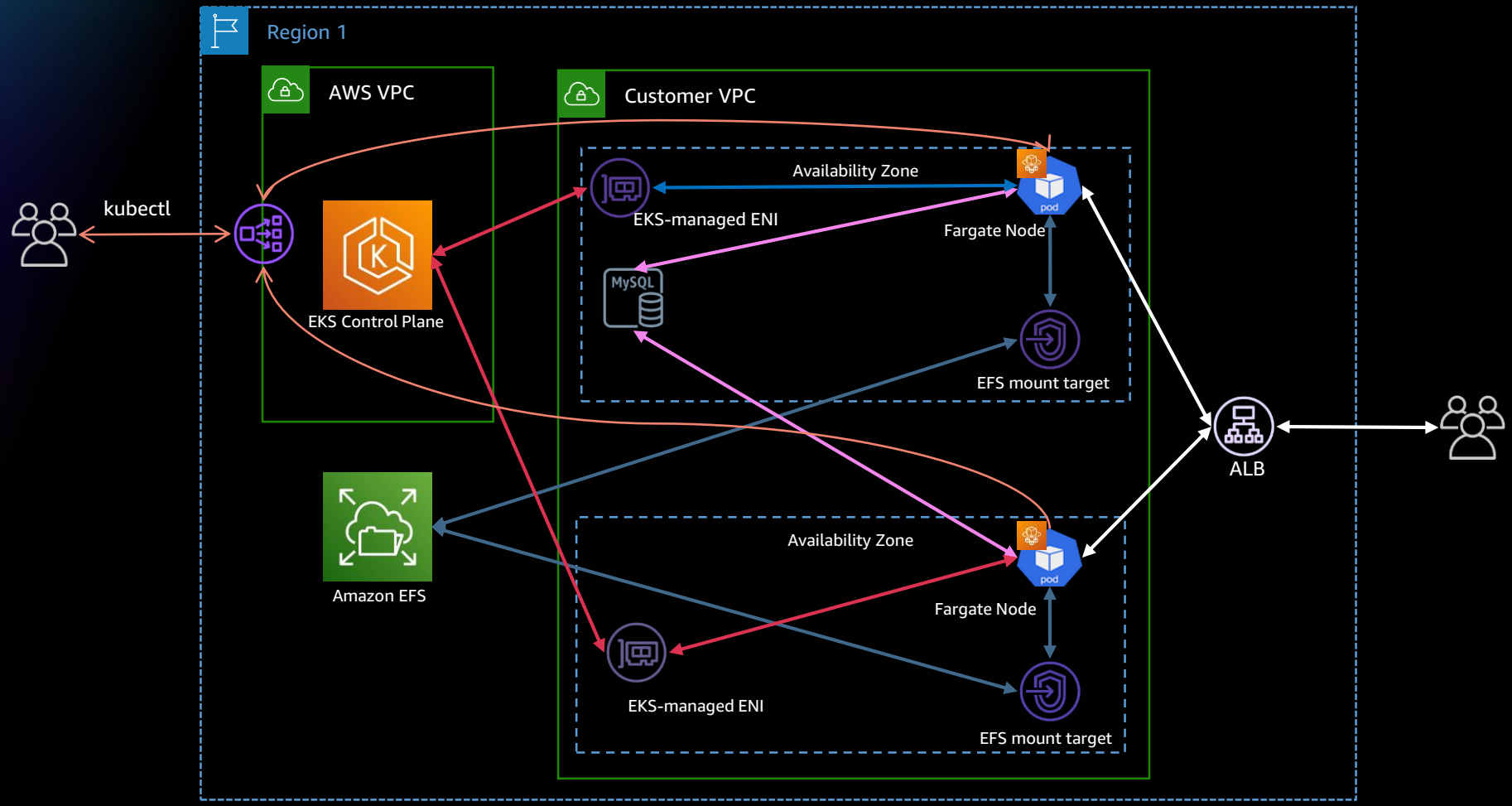
- I don't need shared storage (e.g., database)
- I need point-in-time snapshots

Note: Amazon FSx for Lustre can be used for containers that require ultra-high throughput and very low latency file sharing

Content Management Systems (CMS) generic architecture

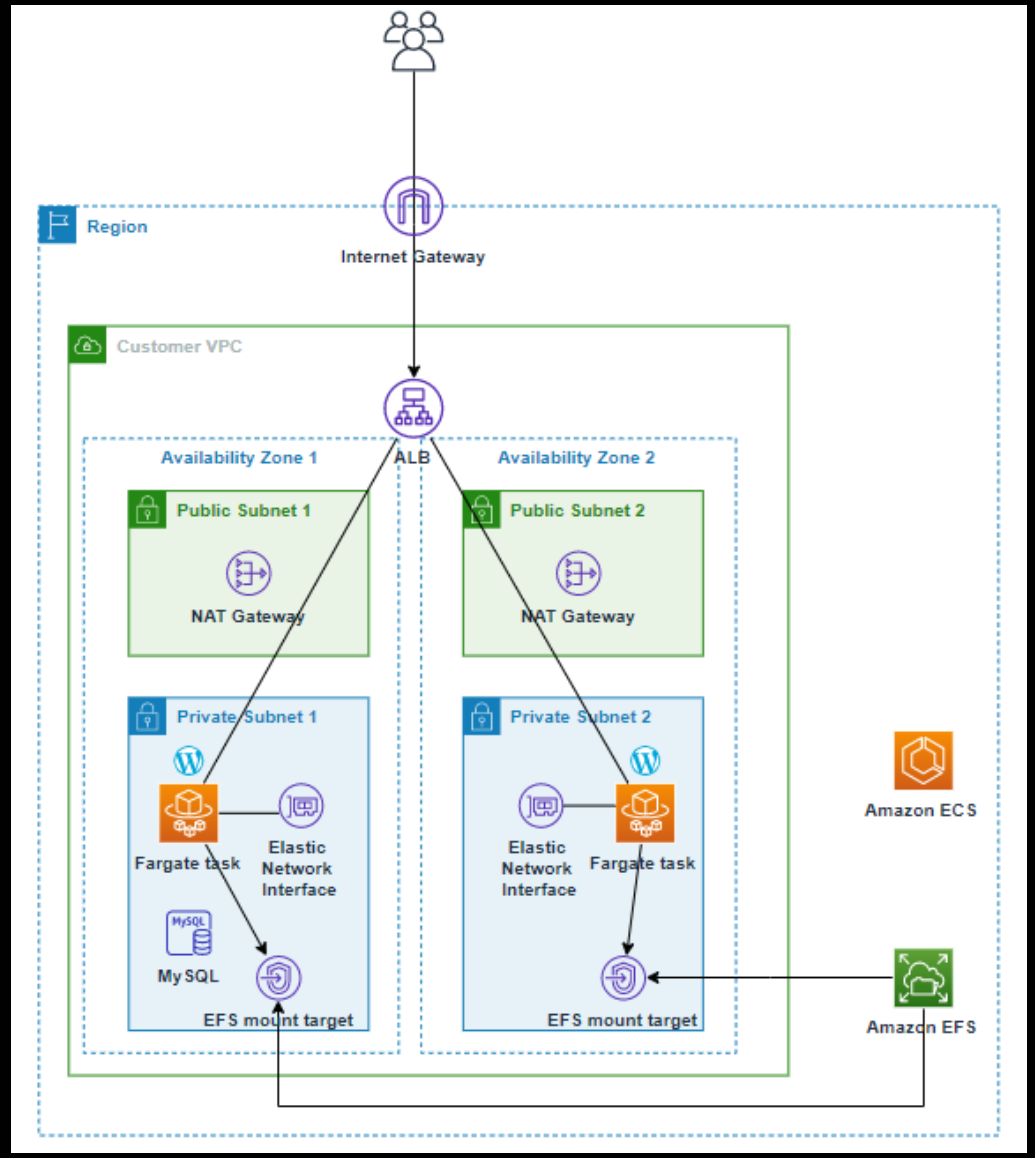


Demo - Amazon EKS on AWS Fargate using Amazon EFS



Demo

Demo - Amazon ECS on AWS Fargate using Amazon EFS



Demo

Key takeaways

- Many modern apps on containers require sharing of data
- Amazon EFS works with all container services for variety of apps
- Security
- Demo and references

Visit the Modern Applications Resource Hub for more resources

Dive deeper with these resources to help you develop an effective plan for your modernization journey.

- Build modern applications on AWS e-book
- Build mobile and web apps faster e-book
- Modernize today with containers on AWS e-book
- Adopting a modern Dev+Ops model e-book
- Modern apps need modern ops e-book
- Determining the total cost of ownership: Comparing Serverless and Server-based technologies paper
- Continuous learning, continuous modernization e-book
- … and more!

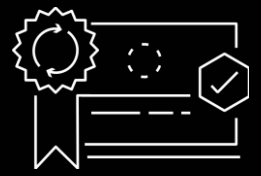
https://bit.ly/3yfOvbK

Visit resource hub »

AWS Training and Certification

Accelerate modernization with continuous learning

Free digital courses, including:
Architecting serverless solutions
Getting started with DevOps on AWS



Earn an industry-recognized credential:
AWS Certified Developer – Associate
AWS Certified DevOps – Professional

Hands-on classroom training (available virtually) including:
Running containers on Amazon Elastic Kubernetes Service (Amazon EKS)
Advanced developing on AWS

Create a self-paced learning roadmap
AWS ramp-up guide - Developer
AWS ramp-up guide - DevOps

Take Developer and DevOps training today

Learn more about Modernization training for you and your team

Thank you for attending AWS Innovate Modern Applications Edition

We hope you found it interesting! A kind reminder to **complete the survey.**
Let us know what you thought of today's event and how we can improve the event experience for you in the future.


aws-apj-marketing@amazon.com


twitter.com/AWSCloud


facebook.com/AmazonWebServices


youtube.com/user/AmazonWebServices


slideshare.net/AmazonWebServices


twitch.tv/aws

Thank you!