
AWS
re:Invent

STG312

Modernize your applications with containers using Amazon EFS

Will Ochandarena
Principal Product Mgr., Amazon EFS
AWS

Jake Farrell
Senior Director of Engineering
Acquia

AWS re:Invent

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

Agenda

Modernization basics

Setting up Amazon EFS for containers

Connecting your container

Case study: Acquia



© 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



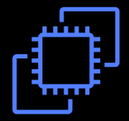
3. Improved reliability, simpler operations, and lower cost

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 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

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

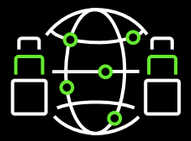
99.99% availability SLA

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

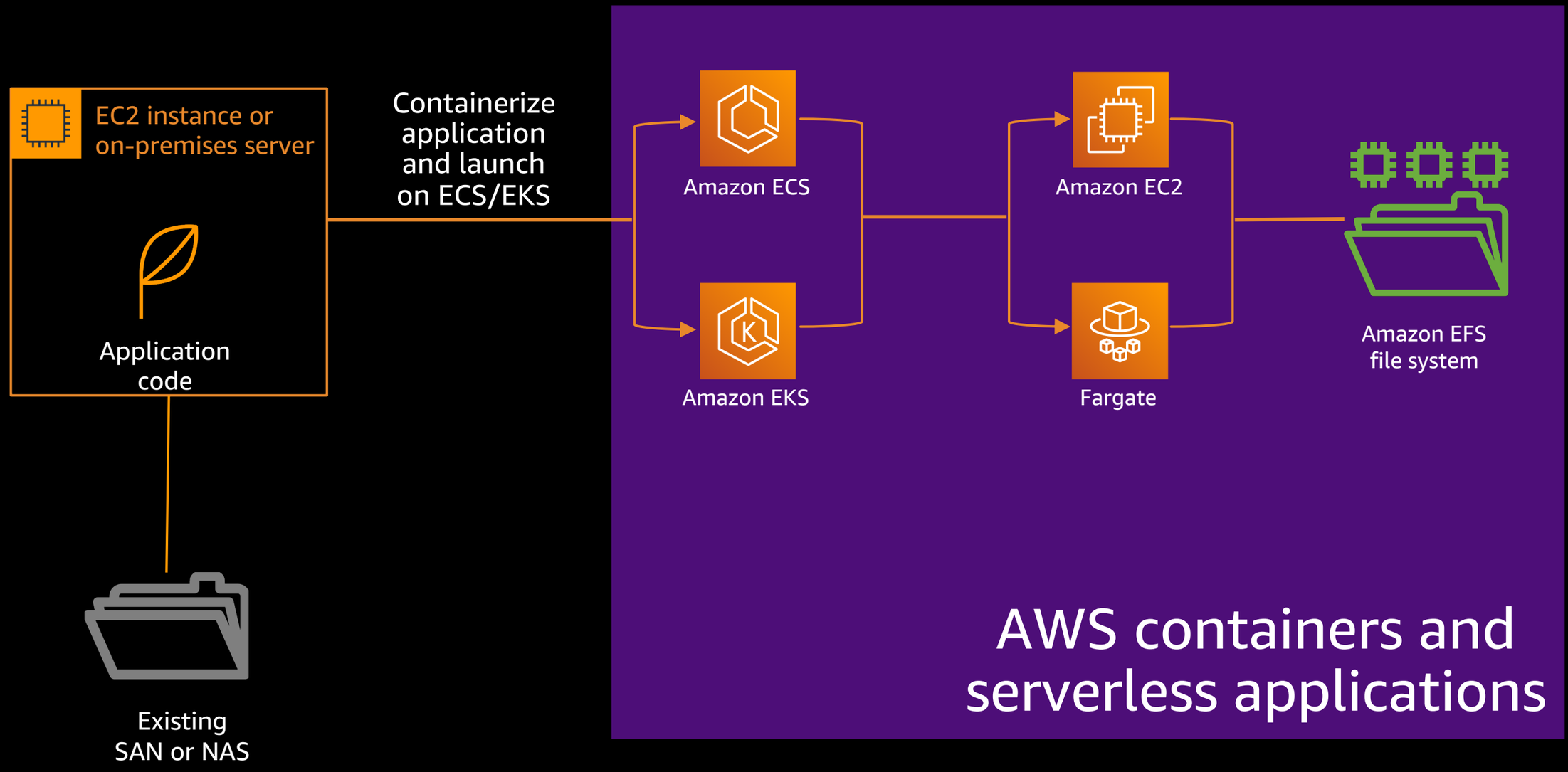
10s of GB/s of and 500,000+ IOPS



Two storage classes

Lifecycle-based cost optimization

Application modernization with containers



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

Benefits of Amazon EFS for containers

Simple

- Amazon EFS, Amazon ECS, and Amazon EKS and Fargate are fully serverless, so no instances to manage
- Amazon EFS configuration is inside ECS / EKS / Lambda application definition
- Developers can focus on their applications, not infrastructure



Available and durable

- Amazon ECS, Amazon EKS, Fargate, and Amazon EFS are regional services
- Customers can build applications that span multiple Availability Zones, with automatic failover

Elastic

- Amazon ECS, Amazon EKS, Fargate, 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 the IAM role of the Amazon ECS task
- Amazon EFS access points can enforce file system permissions when multiple applications share a file system

Setting up Amazon EFS for containers



AWS re:Invent

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

Checklist

Security and identity

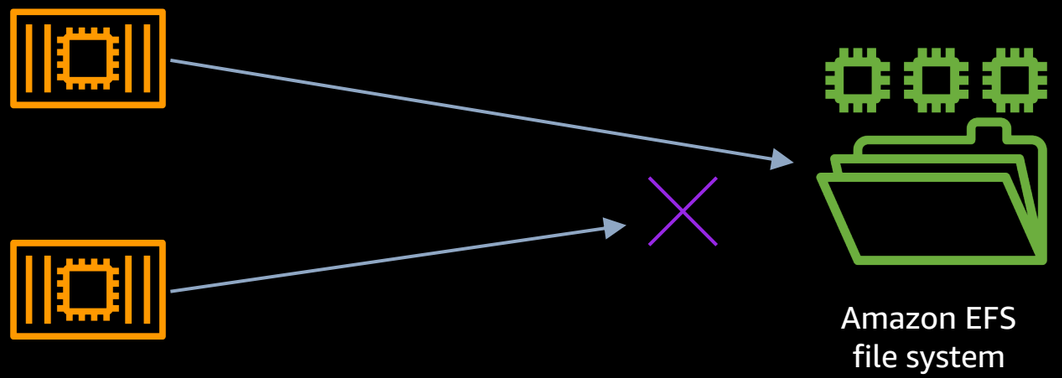
Performance optimization

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

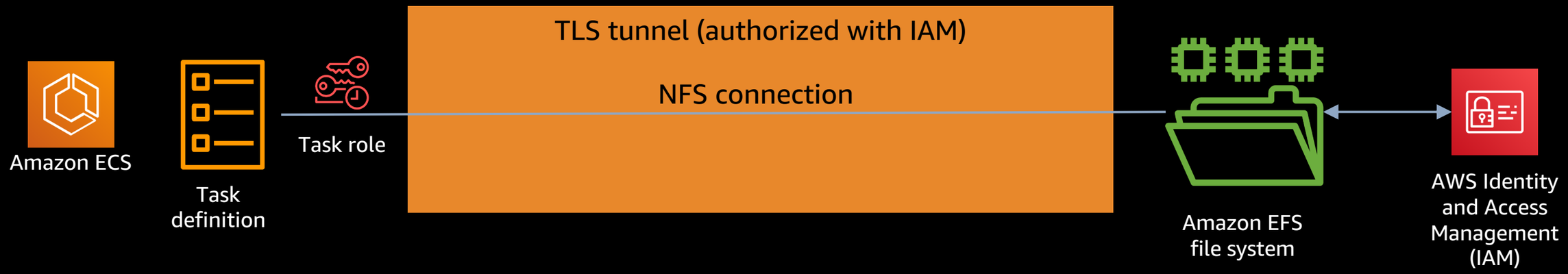



```
$ cat /my_app/data
### SUCCESS THIS IS MY FILE ###

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

Using IAM for file system access

ECS ✓
EKS ✗



Identity policy attached to IAM role

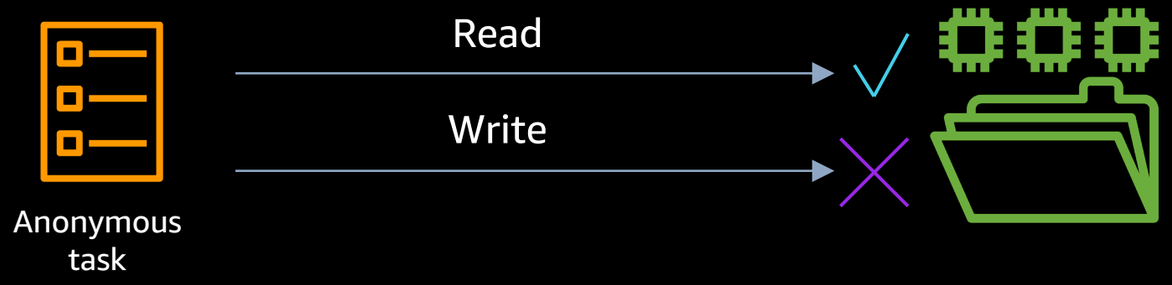
```
{
  “Statement” : {
    “Effect” : “allow”,
    “Action” : “elasticfilesystem:Client*”,
    "Resource": ”fs-feedfeed"
  }
}
```

File system resource policy

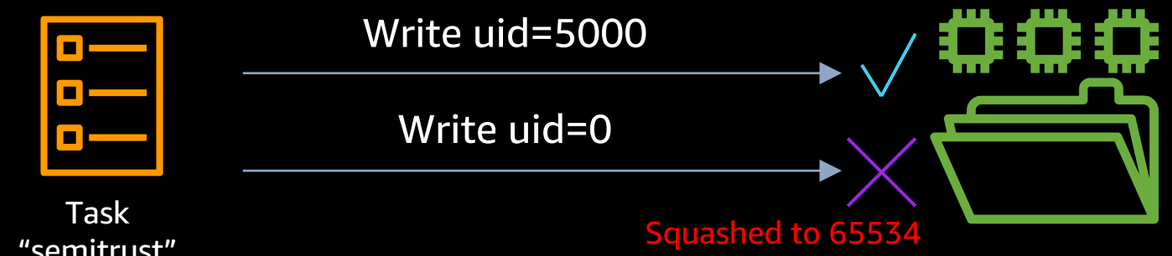
```
{
  “Statement” : {
    “Effect” : “allow”,
    “Action” : “elasticfilesystem:Client*”,
    “Principal” : { “AWS”: “FargateRole” }
  }
}
```

Handling authorization using IAM

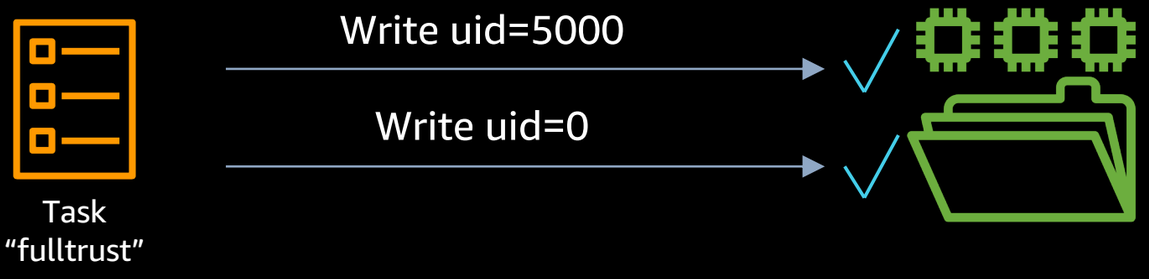
ECS ✓
EKS ✗




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



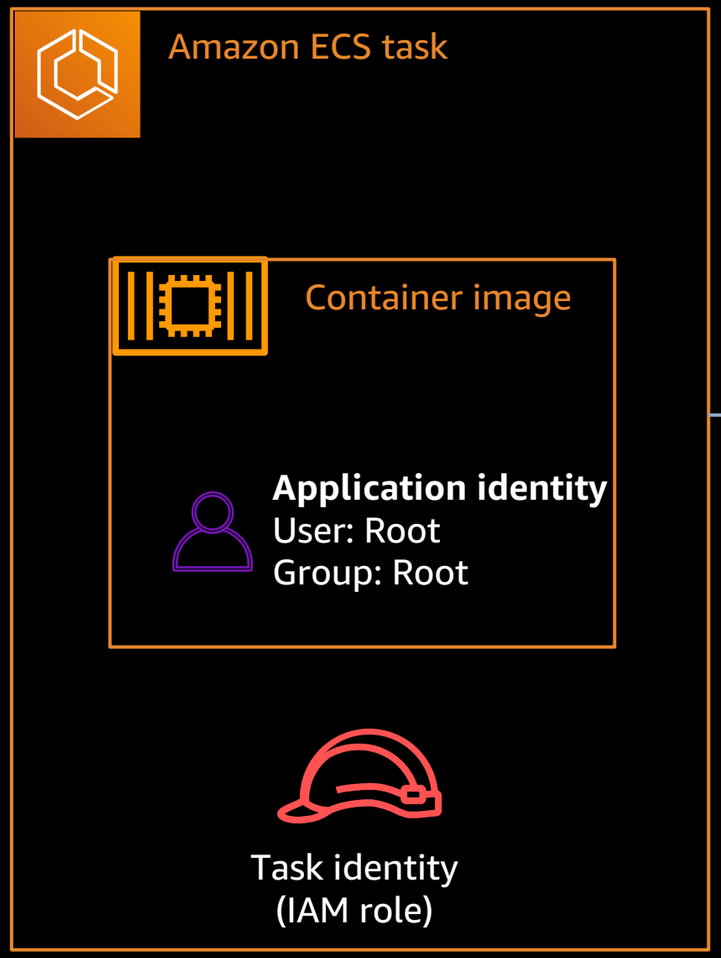

```
"Effect" : "allow",
"Action" : ["elasticfilesystem:ClientMount",
            "elasticfilesystem:ClientWrite"],
"Principal" : { "AWS": "semitrust" }
```




```
"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



READ
/ efs/home/bob

uid=0 (root)





IAM

```
$ ls -l /efs/home

drwx------ bob        .          BobHome

drwx------ maria      .          MariaHome

drwxrwx--- .          biusers    BI_Shared
```

Application-specific access with Amazon EFS access points

ECS	✓
EKS	✓

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

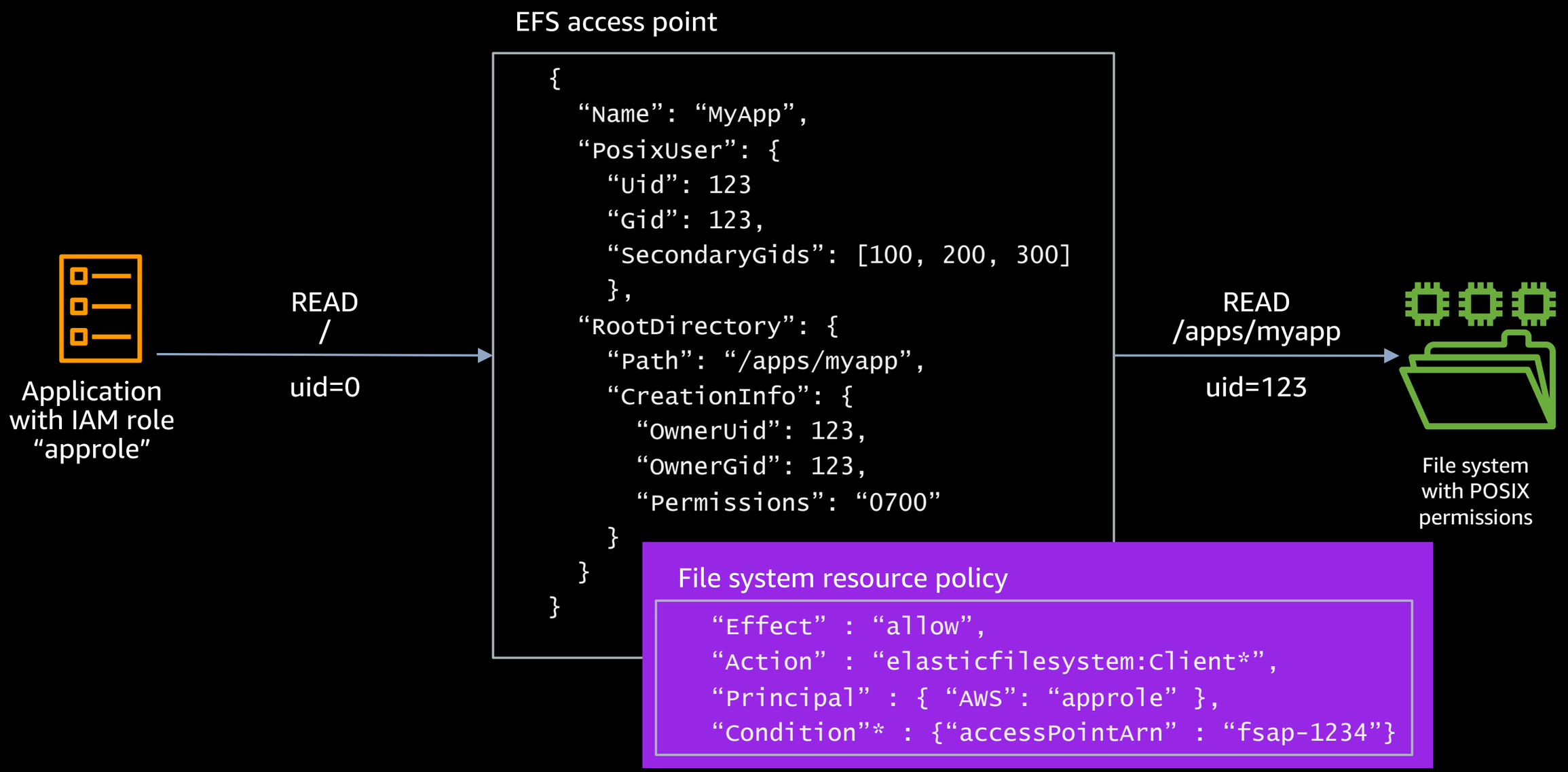
- Root containers can’t escalate access
- Arbitrary users aren’t locked out

Creates application-specific directory and permissions

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

How Amazon EFS access points work

ECS ✓
EKS ✓

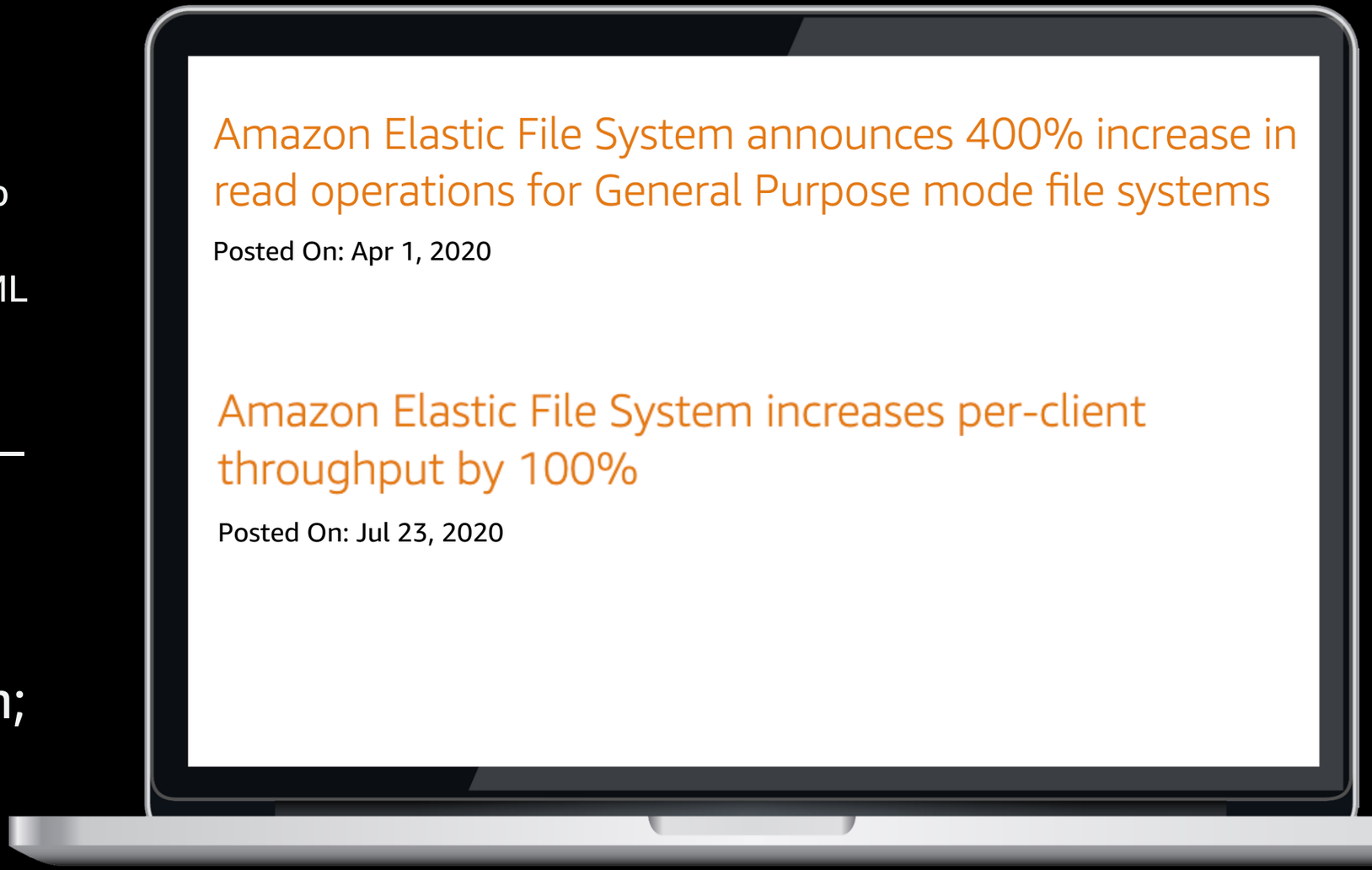


Best practices for security

- 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

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
- I'd like to take advantage of spot pricing



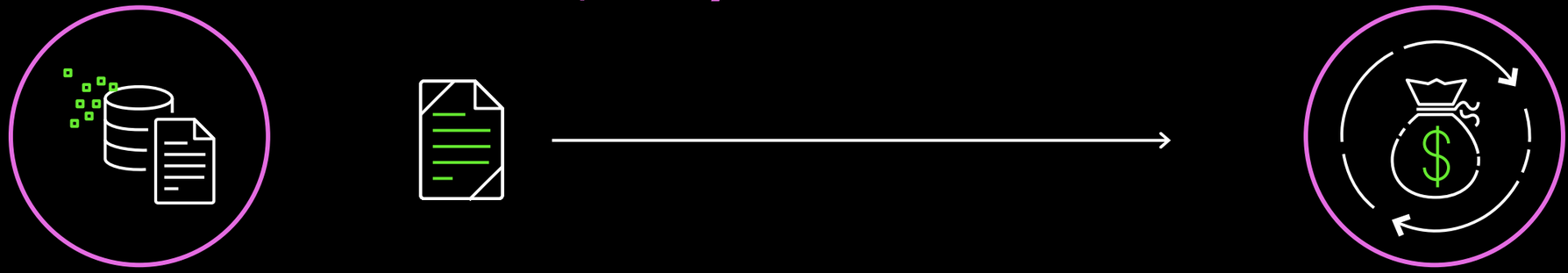
Amazon EBS

- 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

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

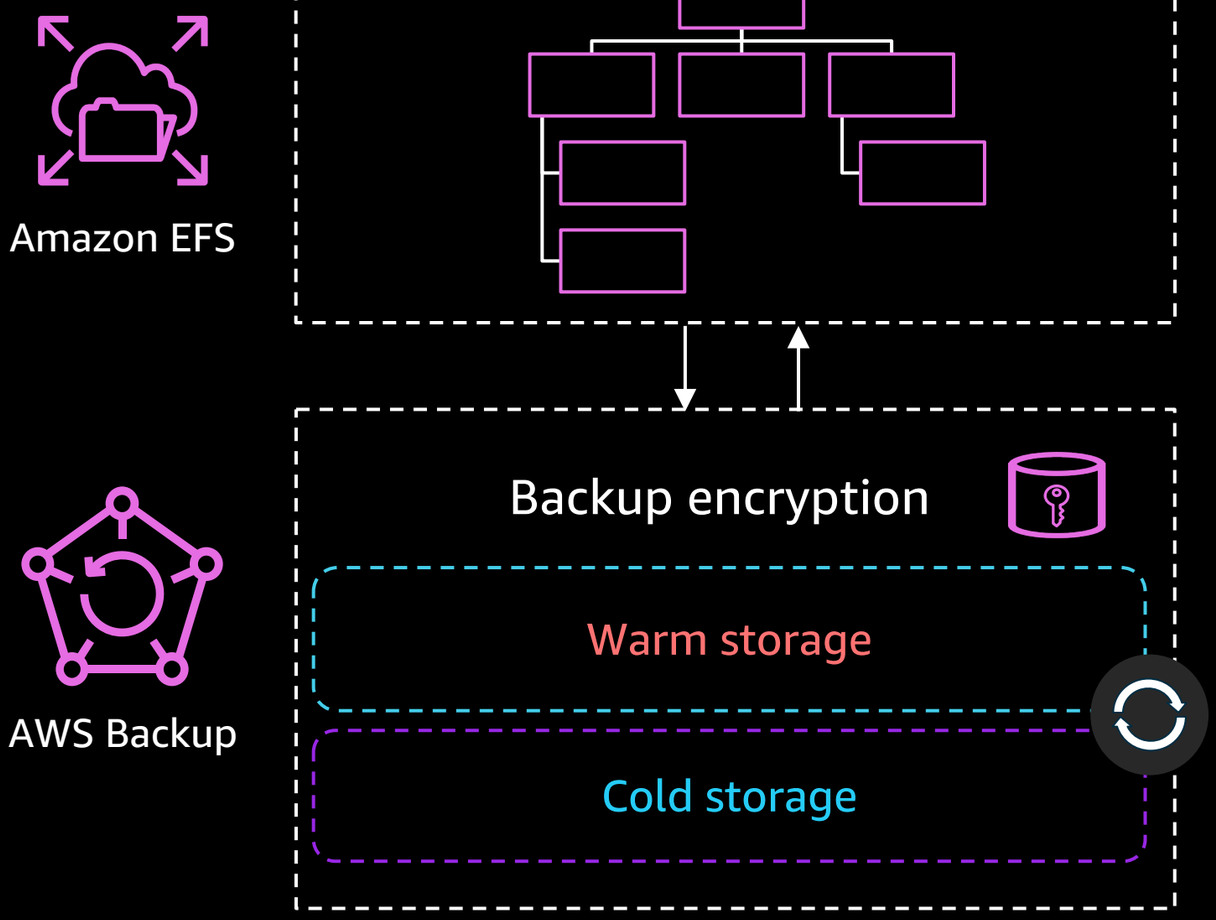
Cost-optimized for files not accessed every day

$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

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

Connecting your container



AWS re:Invent

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

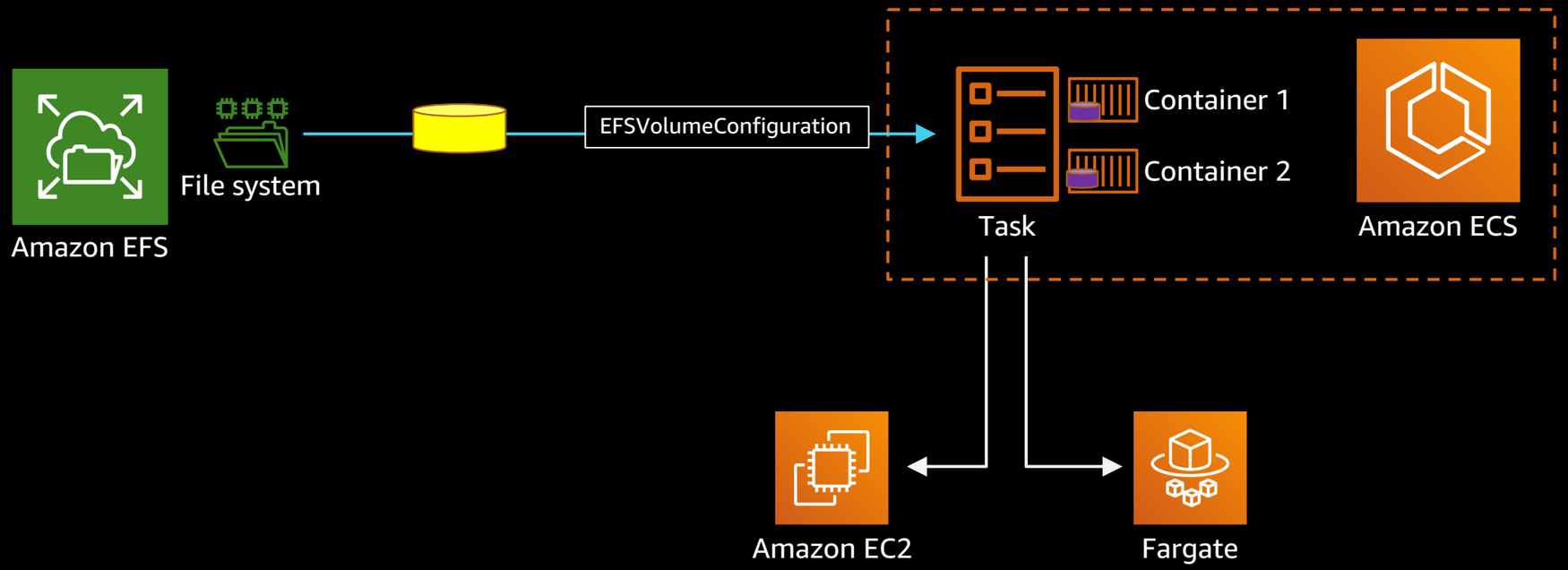
Amazon ECS



AWS re:Invent

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

How it works


Amazon EFS
File system
EFSVolumeConfiguration
Container 1
Container 2
Task
Amazon ECS
Amazon EC2
Fargate



Add volume

Name: NotebookServer

Volume type: EFS

File system ID: sourcecode | fs-8…

sourcecode | fs-8239c3fa

trainingdata | fs-dc0d2aa5

System in

Access point ID: None

Create an access point for your file system in the Amazon EFS Console

Root directory: /

Encryption in transit: Enable Transit Encryption

EFS IAM authorization: Enable IAM Authorization

*Required

Cancel

Add



Amazon ECS

Container Definitions

Add container

Container Name	Image
FileBrowser	2648900(

Service Integration

AWS App Mesh is a service mesh bas
giving you end-to-end visibility and he
will auto-configure the proxy configura

Edit container

Mount points

Source volume FargateDemoEFS

Container path /data

Read only

Add mount point



Amazon ECS

```
"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",
```

Amazon EKS



AWS re:Invent

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

Amazon EFS and Amazon EKS: Concepts



- Container Storage Interface (CSI)
  - Industry standard interface for connecting storage providers (block or file) to a container
- Amazon EFS CSI driver
  - Implementation of CSI for connecting file systems to containers
- Storage class (SC)
  - Administrator-defined class of storage that persistent volumes can be created from
- Persistent volume (PV)
  - Administrator-created unit of storage that can be attached to a container; has its own lifecycle
- Persistent volume claim (PVC)
  - Request to allocate an available PV from a SC to a container

Amazon EKS storage: Installation



Amazon EKS on Amazon EC2

```
# kubectl apply -k \
"github.com/kubernetes-sigs/aws-efs-csi-driver/deploy/kubernetes/overlays/stable/ecr/?ref=release-1.0”
#
```

Amazon EKS on Fargate

```
#
```

Attaching an Amazon EFS file system to a pod (admin)



Create storage class

```
kind: StorageClass
apiVersion: storage.k8s.io/v1
metadata:
 name: efs-sc
provisioner: efs.csi.aws.com
```

Create persistent volume

```
apiVersion: v1
kind: PersistentVolume
metadata:
 name: efs-pv
spec:
 capacity:
 storage: 5Gi
 volumeMode: Filesystem
 accessModes:
  - ReadWriteMany
  persistentVolumeReclaimPolicy: Retain
 storageClassName: efs-sc
 csi:
  driver: efs.csi.aws.com
  volumeHandle: fs-feedfeed::fsap-feedfeedfeed
```

VolumeHandle path and access point



New

Support access points on the same file system ✓ approved cncf-cla: yes lgtm ok-to-test size/L

#185 was merged on Jun 18

volumeHandle: {fsid}:{subpath}:{apid}

Examples

Mounting a subpath: fs-feedfeed:/myapppath/:

Mounting an AP: fs-feedfeed::fsap-feedfeedfeedfeed

Submounting an AP: fs-feedfeed:/subpath:fsap-feedfeedfeedfeed

Attaching an Amazon EFS file system to a pod (user)



Create persistent volume claim

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
 name: efs-claim
spec:
 accessModes:
  - ReadWriteMany
 storageClassName: efs-sc
 resources:
  requests:
   storage: 5Gi
```

Launch pod

```
apiVersion: v1
kind: Pod
metadata:
name: efs-app
spec:
containers:
-  name: web-container
   image: httpd
   ports:
   - containerPort: 80
     name: “http-server”
   volumeMounts:
   - name: persistent-storage
     mountPath: /mnt-efs
volumes:
-  name: persistent-storage
   persistentVolumeClaim:
     claimName: efs-claim
```

Case study: Acquia



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

WHO WE ARE

THE OPEN DIGITAL EXPERIENCE PLATFORM FOR DRUPAL

We provide the world's most ambitious brands with technology that allows them to embrace innovation and create customer moments that matter. At Acquia, we believe in the power of community — giving our customers the freedom to build tomorrow on their terms.

94%
Customer Satisfaction Rating

Leveraging **Drupal**, the Largest Open Source Community

4000+ customers, including 1,800 enterprises

30+
Fortune 100 customers

1000+
Employees

Global
Partner Program

Why Customers Build Their Digital Experiences on Acquia

NBC

Risk mitigation

30,000 sporting events

–

Platform scaled to deliver 3.3 billion total streamed minutes

–

100 million unique visitors on a single event, NBCOlympics.com

Wendy's

Digital transformation

Robust content and personalization strategy helped Wendy's migrate to Drupal with Acquia Cloud

–

140x reduction in content update times

–

137% improvement in speed

–

20% YoY increase in organic users (to 40M)

lululemon

Grow audience

Facilitate customer relationships across channels

–

Increased site visits by up to 50%

–

Achieved a 10–15% increase in baseline revenue from its digital marketing campaigns

Acquia
EXPERIENCE DIGITAL FREEDOM



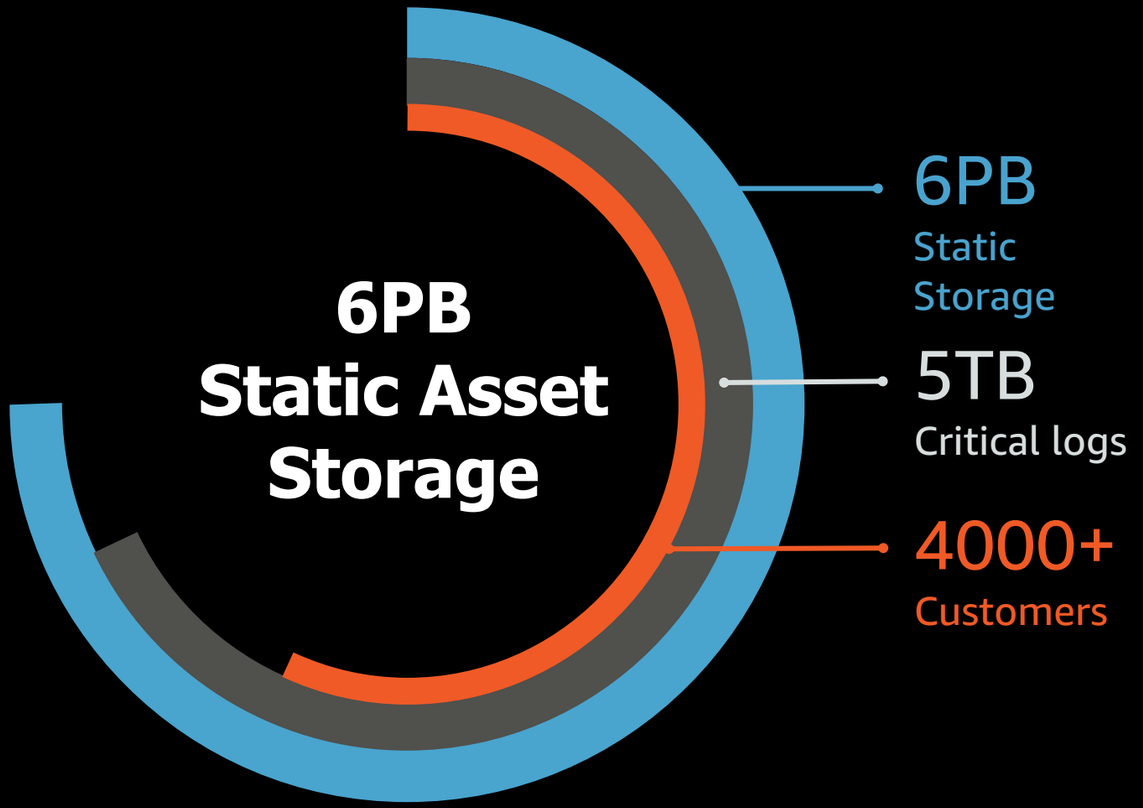
Acquia Cloud Platform

The Acquia Cloud Platform is a comprehensive solution for building, launching, and managing Drupal-based digital experiences. Built with security, performance, and infinite scalability in mind, and SLA uptime of 99.95%, the Acquia Cloud Platform is the foundation your business needs in an ever-changing world.

KEY BENEFITS

- Increased customer conversion and acquisition
- The flexibility and freedom to build what you want
- A reliable platform that meets internal governance and external compliance standards

By the Numbers



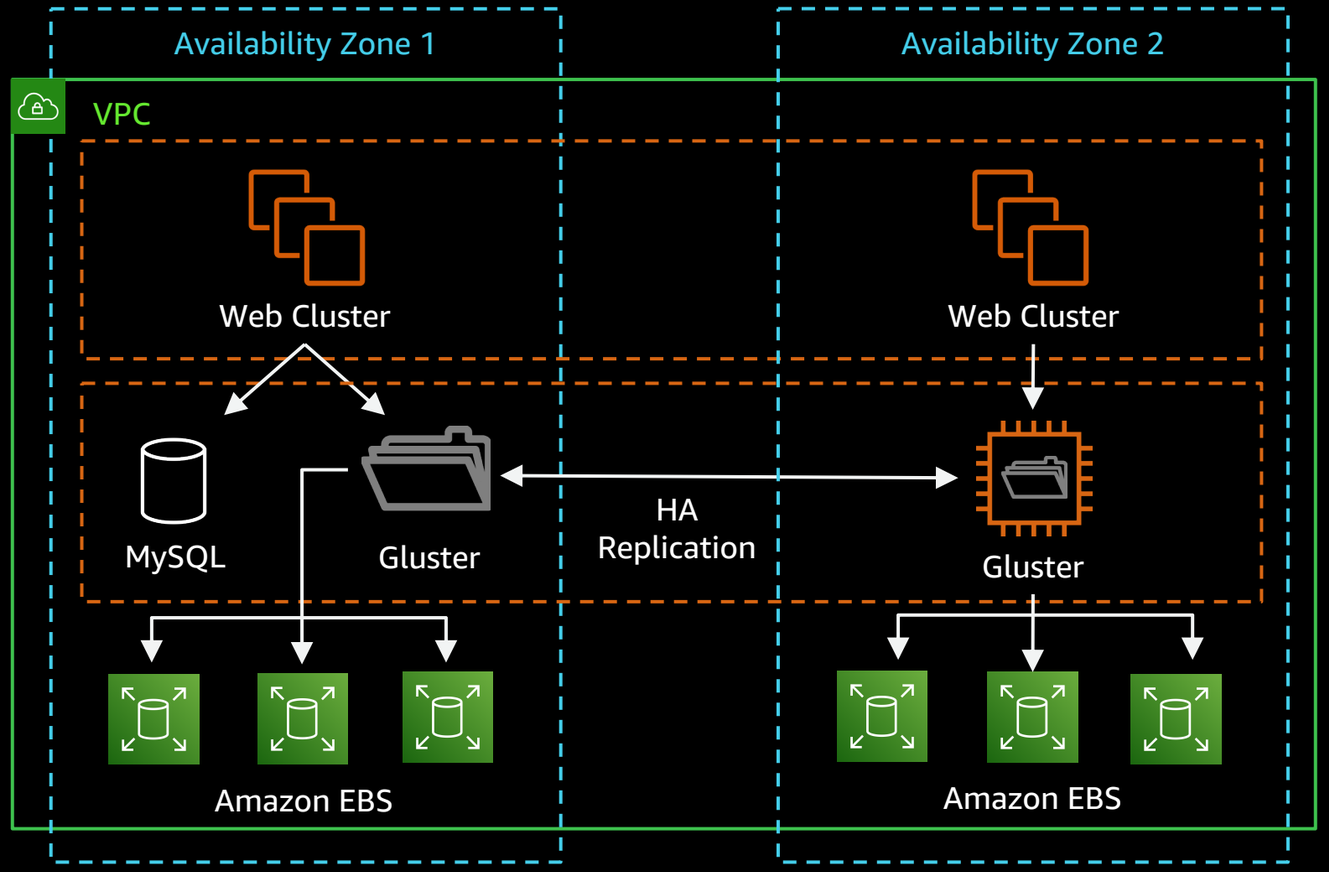
6 petabytes of customer static asset storage served across 10 regions from tens of thousands of elastic web servers

Acquia Cloud platform generates, sends and archives 5TB+ of mission critical service, application and system logs daily

Customers trust Acquia for the biggest, toughest most can’t-fail situations and digital events on the internet today. 4000+ customers are running thousands of sites on Acquia Cloud

Acquia
EXPERIENCE DIGITAL FREEDOM

Previous Storage Architecture



Acquia
EXPERIENCE DIGITAL FREEDOM

Setup and Challenges

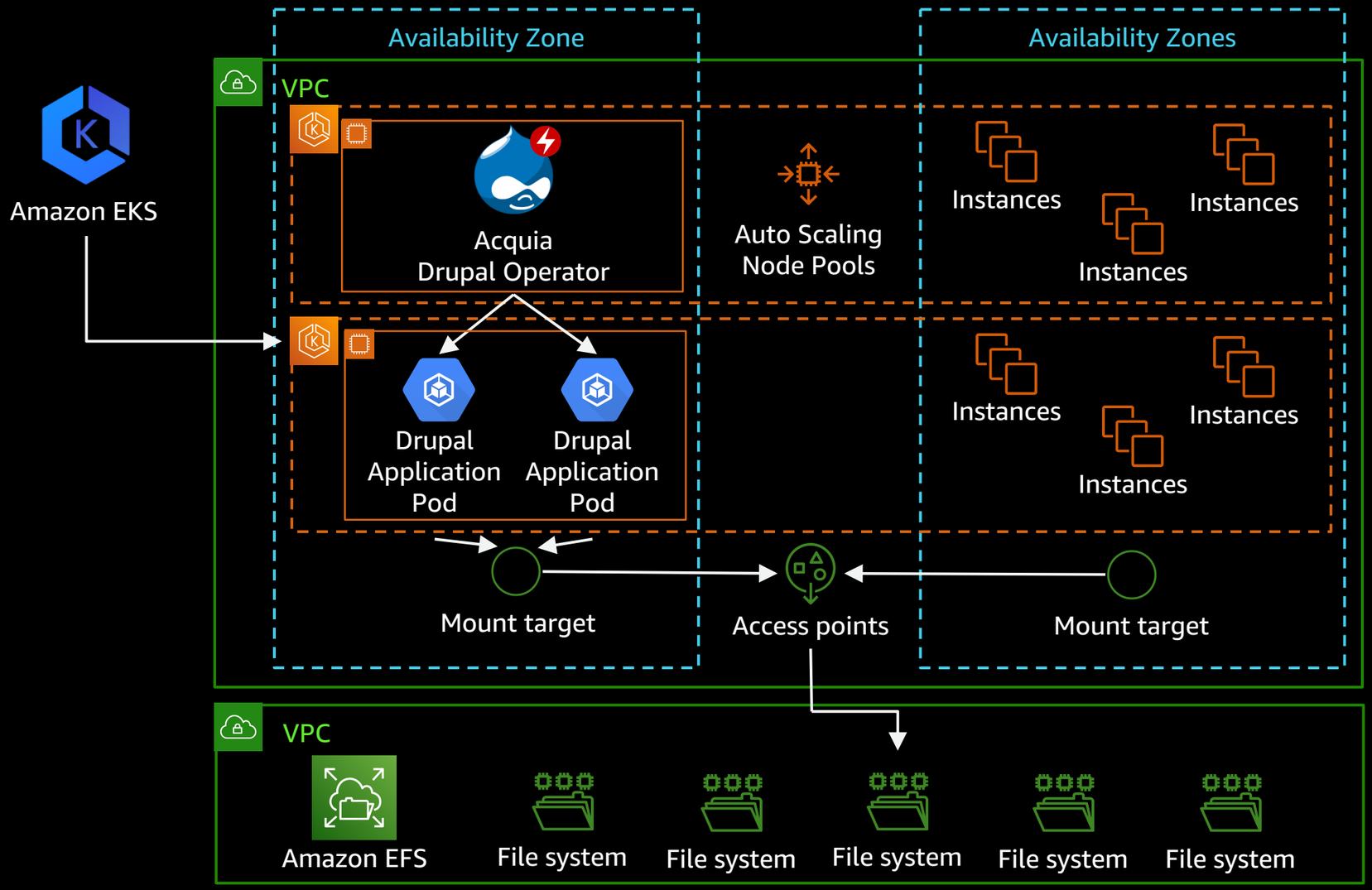
Self-managed Gluster on Amazon EC2

- Configuration via Puppet
- Mix of tasked-based and manual disk management
- Custom backup and DR strategy
- Data servers turn into pets quickly

PAIN POINTS

- Balancing act to not overprovision
- Service patching and upkeep
- Operational server maintenance overhead
- Customer data migration can be complex
- Recovery difficult and time consuming

Continuing to Evolve



Acquia
EXPERIENCE DIGITAL FREEDOM

Amazon EFS Container Storage Interface (CSI) driver

All the benefits of Amazon EFS integration utilizing standard Kubernetes interfaces

- Encryption in transit
- Access points integration
- NodeGetVolumeStats



Amazon EKS + Container Storage Interface + Amazon EFS

https://github.com/kubernetes-sigs/aws-efs-csi-driver

Acquia

By containerizing our hosting applications and running them on Amazon EKS with Amazon EFS we have improved our customer experience while considerably reducing our infrastructure and operational maintenance overhead

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.