Bridging Security Infrastructure Between the Data Center and AWS Lambda

Michael Weissbacher
Square, Inc.
@mweissbacher

Black Hat USA 2021
August 4th

Overview

- Background & Context
- Goals
- Solution Pt 1: Enabling Lambda to call microservices in the DC + AWS EKS
- Solution Pt 2: Syncing application secrets
- Key Learnings

Bio

- Michael Weissbacher, PhD
- Infrastructure Security Team @ Square in NYC
- Subteam focusing on Cryptographic Identity and Secrets
- Previously: Security Research @ Northeastern University

Background & Context

Why do developers choose Lambda?

Benefits of serverless or Lambda specifically:

- **Focus**: business logic, rather than infrastructure
- **Speed:** it’s fast!
- **Scalability**: both up and down. No need to maintain idle servers
- **Compatibility**: can be triggered through close integration with AWS APIs

How does it work?

- Lambdas are code without permanent infrastructure
- Small VMs on Amazon Linux
- Firecracker MicroVM, Lambda Sandbox
- On demand, infrastructure is allocated: cold start
- Subsequent invocations: reuse state
- Demand fades: infrastructure is deallocated

The problem with serverless

Serverless architecture is not compatible with security infrastructure in the DC

- Reasons Lambda is not compatible:
  - Geared towards short lived workloads
  - Deploy process different
  - Immediate response core feature
  - Serverless scalability cannot be held back

Goal

Goal at Square

Square has been migrating to the cloud to achieve higher flexibility and scalability

We need Lambdas to be treated the same as other workloads

- DC: Kubernetes-like platform
- AWS: Kubernetes on EKS
- Connections: Envoy service mesh

What do we need for Lambda?

- Communicate securely
- Access application secrets

Lambda can't be an island!

System Goals

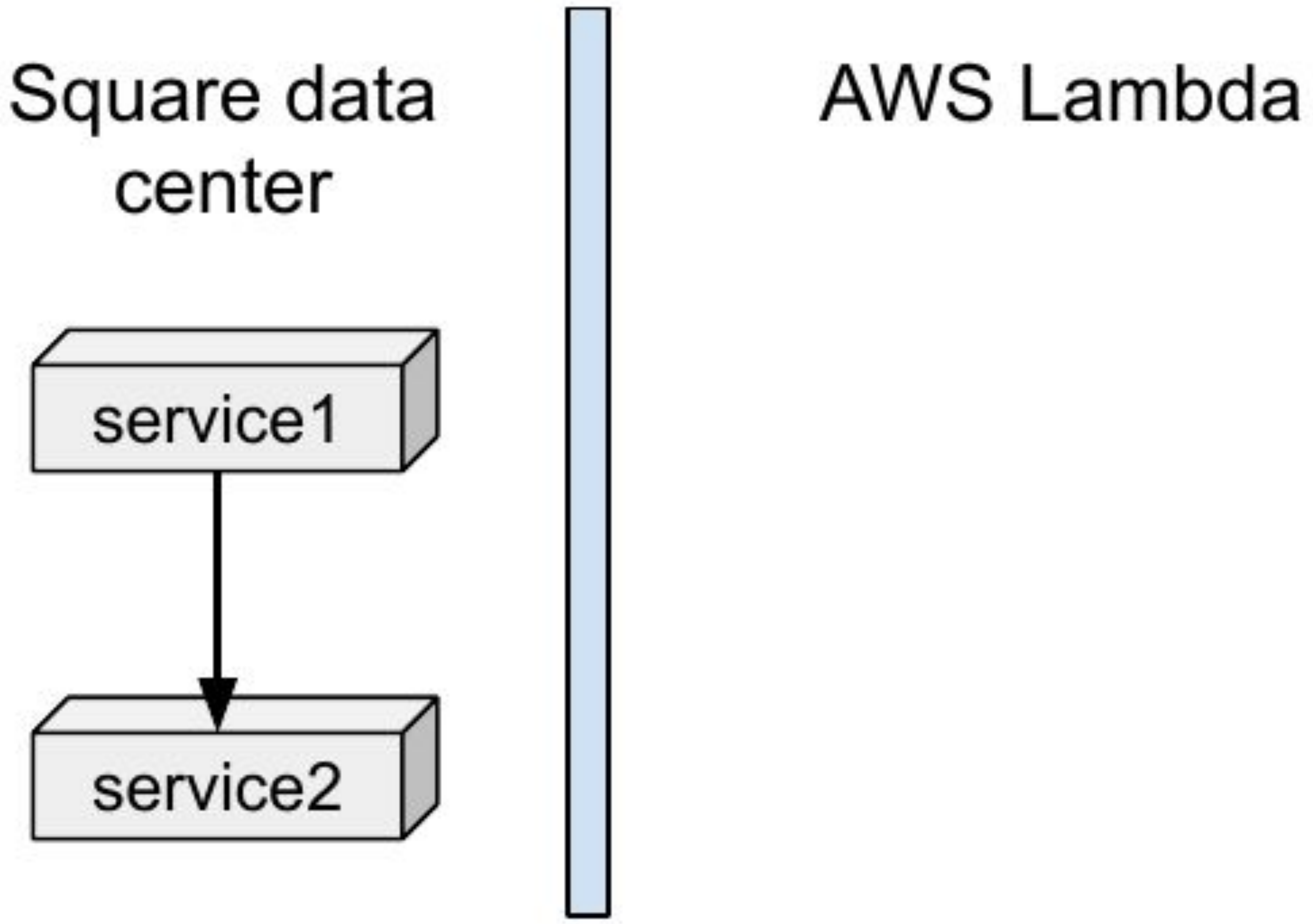
Equal footing with DC security infrastructure ...

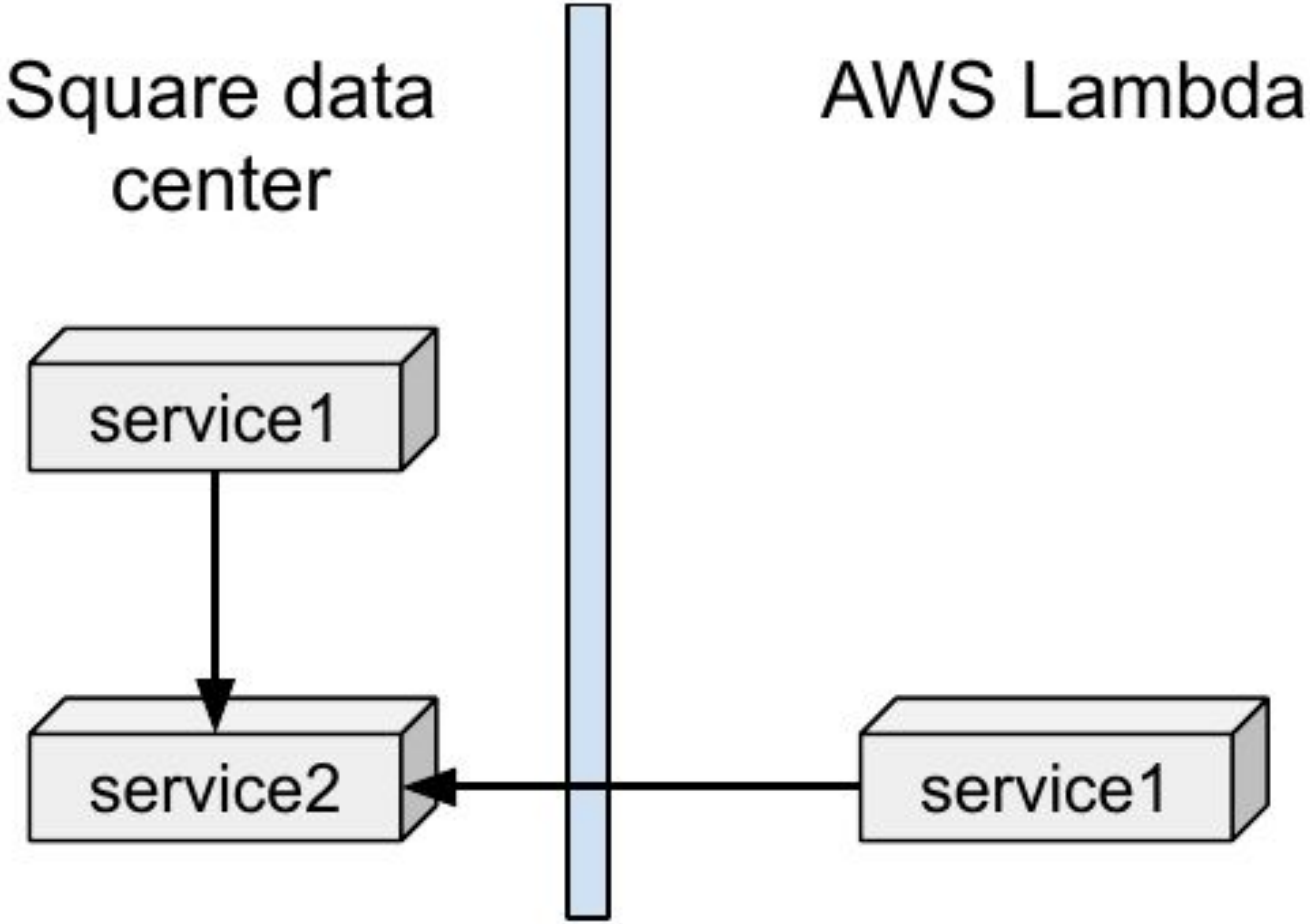
- **Communication:** Connect to DC/AWS EKS mesh
- **Application secrets**: Access to protected data

...while still maintaining Lambda benefits

- **Speed:** Must support fast response times
- **Scalability:** Must scale with Lambda demand
- **Compatibility:** Must plug into DC infrastructure
- **Availability:** Must be high

Solution Pt 1:
Enabling Lambda to call microservices in DC/EKS


Square data center
AWS Lambda
service1
service2


Square data center
AWS Lambda
service1
service2
service1

Workload identity at Square

- Everything is a micoservice
- mTLS at Square since 2012 a.k.a. “Zero trust networking” or “Identity is the new perimeter”
- Traffic via Envoy service mesh, sidecar handles connection
- Workloads in DC + AWS EKS
- Identity is tuple (service name, environment)
- E.g.: (service1, staging), or (service2, production)

SPIFFE

- Workload identity standard
- Influenced by industry usage of workload and service identity at Google, Square, ...
- SPIRE is reference implementation for SPIFFE

- Square started migrating to SPIRE 3 years ago
- Deprecate homemade identity issuance...
- ... onboard open source solution
- We use SPIFFE identity in all environments

Shape of identity for Lambda

- Multi-account architecture
- Mapping
  - (service1, staging) ⇔ (AWS Account 12345)
  - (service1, production) ⇔ (AWS Account 23456)
- Multiple Lambdas in one account: one service
- We use accounts as a security boundary per service and environment
- We have different roles within an account (read only, execution role, ...)

Identity issuance: what options were available?

	SPIRE	Bootstrap from Account Credentials
+	● DC: identity is solved problem ● Works well for long-lived services	● Externally signal account ownership ● Construct via KMS building bearer token ● Signed request
-	● No support for serverless ● Requires agent ● No sidecar support in Lambda	● Could carry cloud implementation details into DC ● Doesn't fit into architecture picture

Identity issuance decision

BUILD

Existing options didn’t meet our goals

Architecture for identity issuance

	Pull	vs.	Push
Identity Issuance	• Identity would be generated as a Lambda is invoked • Create agent that operates similar to a sidecar		• Issue identity and make sure it is readily available • Anti-pattern compared to SPIRE using pull
Security	• Security model equivalent to DC		• No security downsides • Identity controlled by IAM and SCP
Availability	• For Lambda, agent creates blocking dependency for invocation		• Flexible

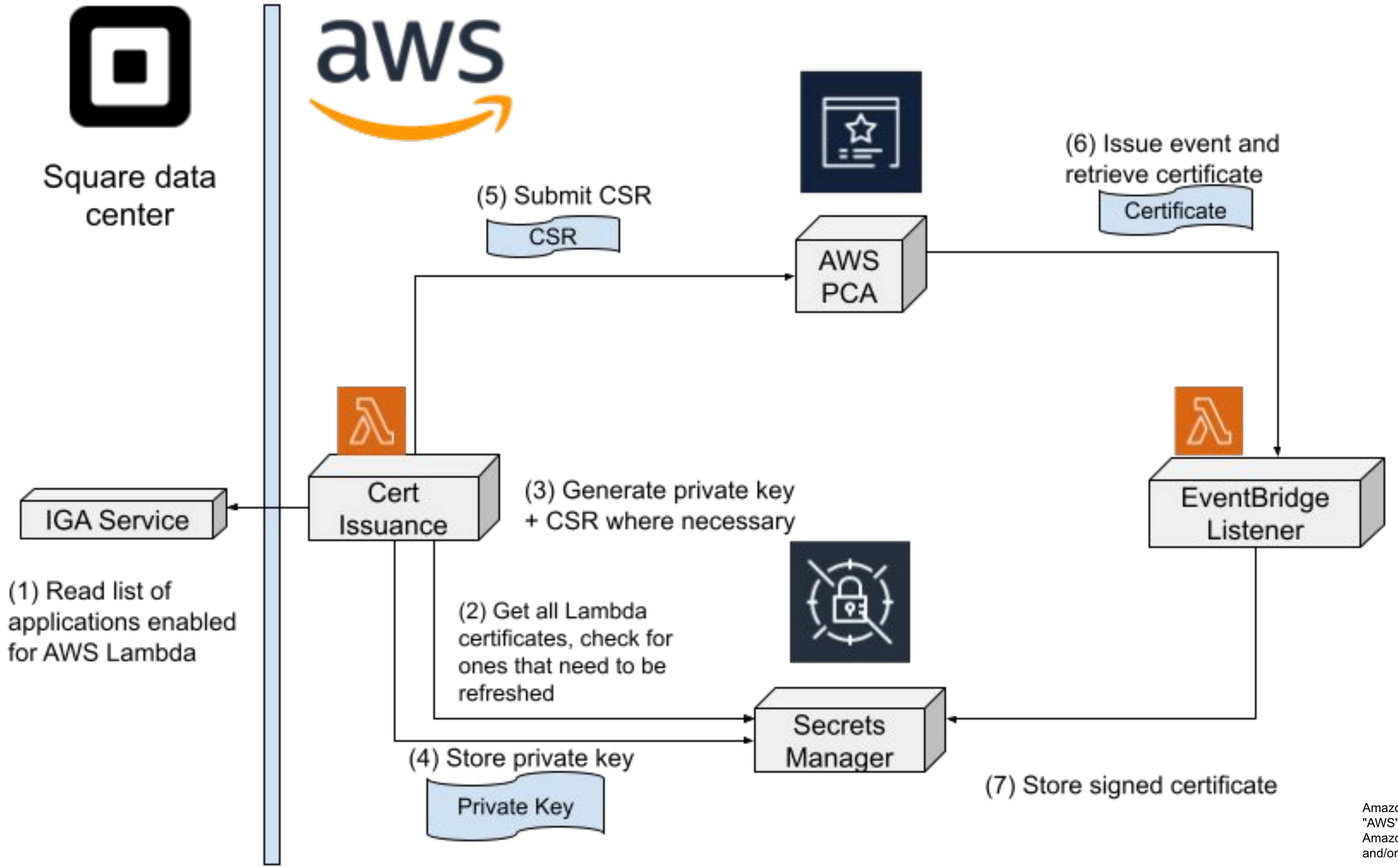
Architecture Decision

[✓ PUSH] ~~PULL~~

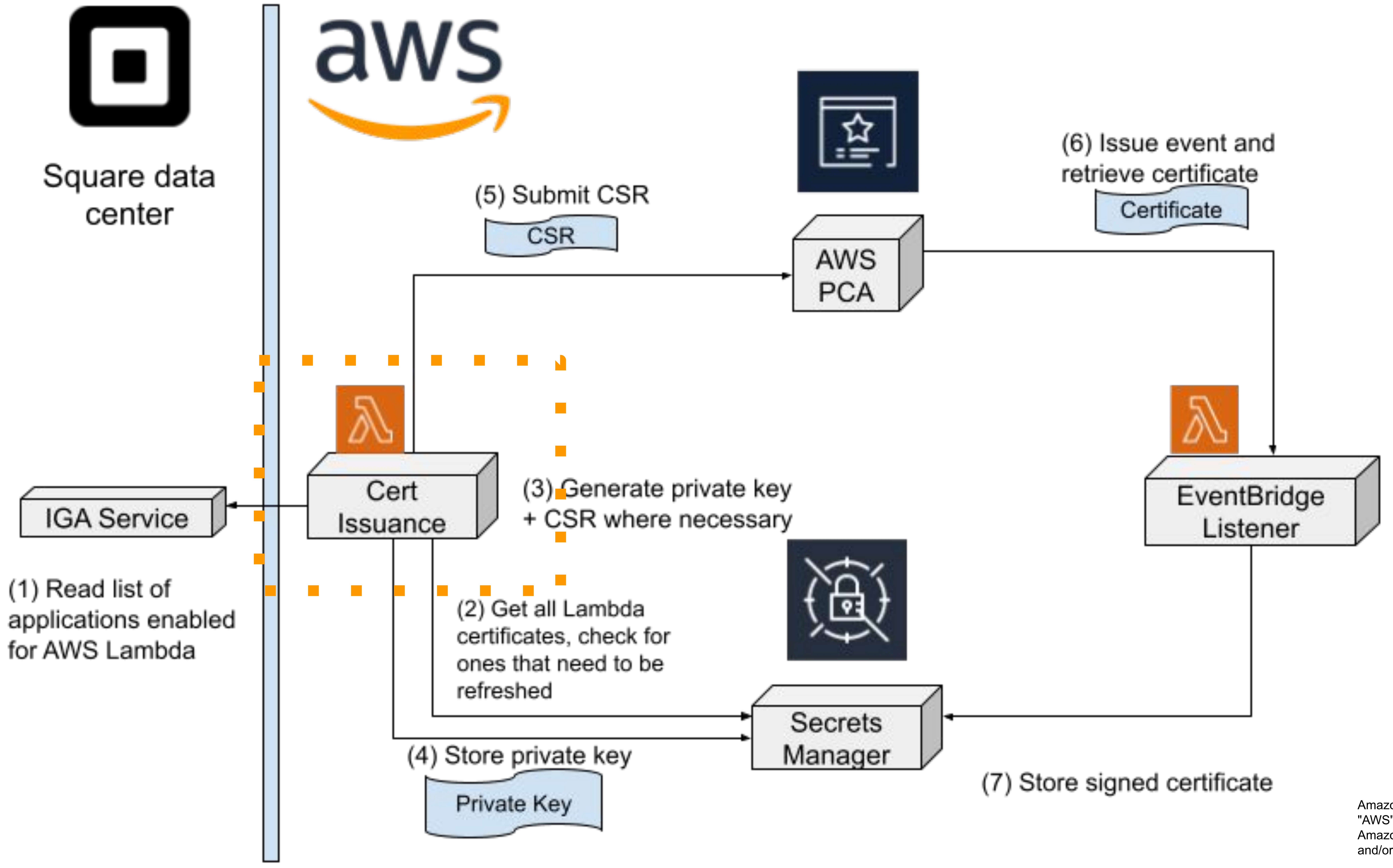
- No security downsides
- Enables us to be more flexible with SLA
- Lambda never has to wait for identity issuance

System components

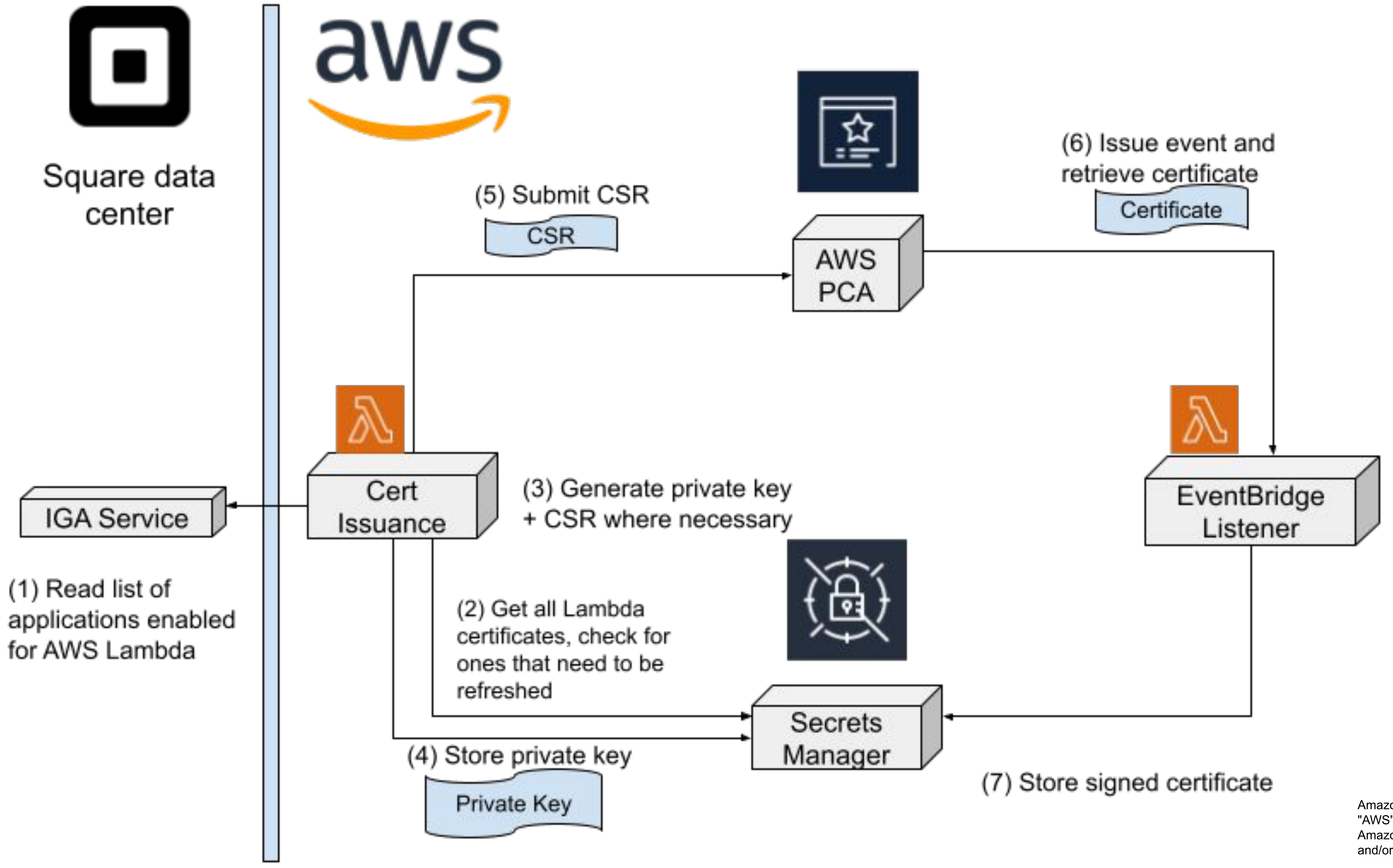
- Identity Governance and Administration (IGA)
  - Square internal service
  - Provides (service, environment, account ID) with enabled Lambda identity
- Issuance
  - Generate Certificate with SPIFFE URI per-service
  - Short lived, 24h
  - Implemented as Lambda
- AWS Private CA (PCA)
  - Used by issuance
  - HSM backed CA service
  - Audit capabilities
- AWS Secrets Manager
  - Centralized resource
  - Access via IAM and SCP



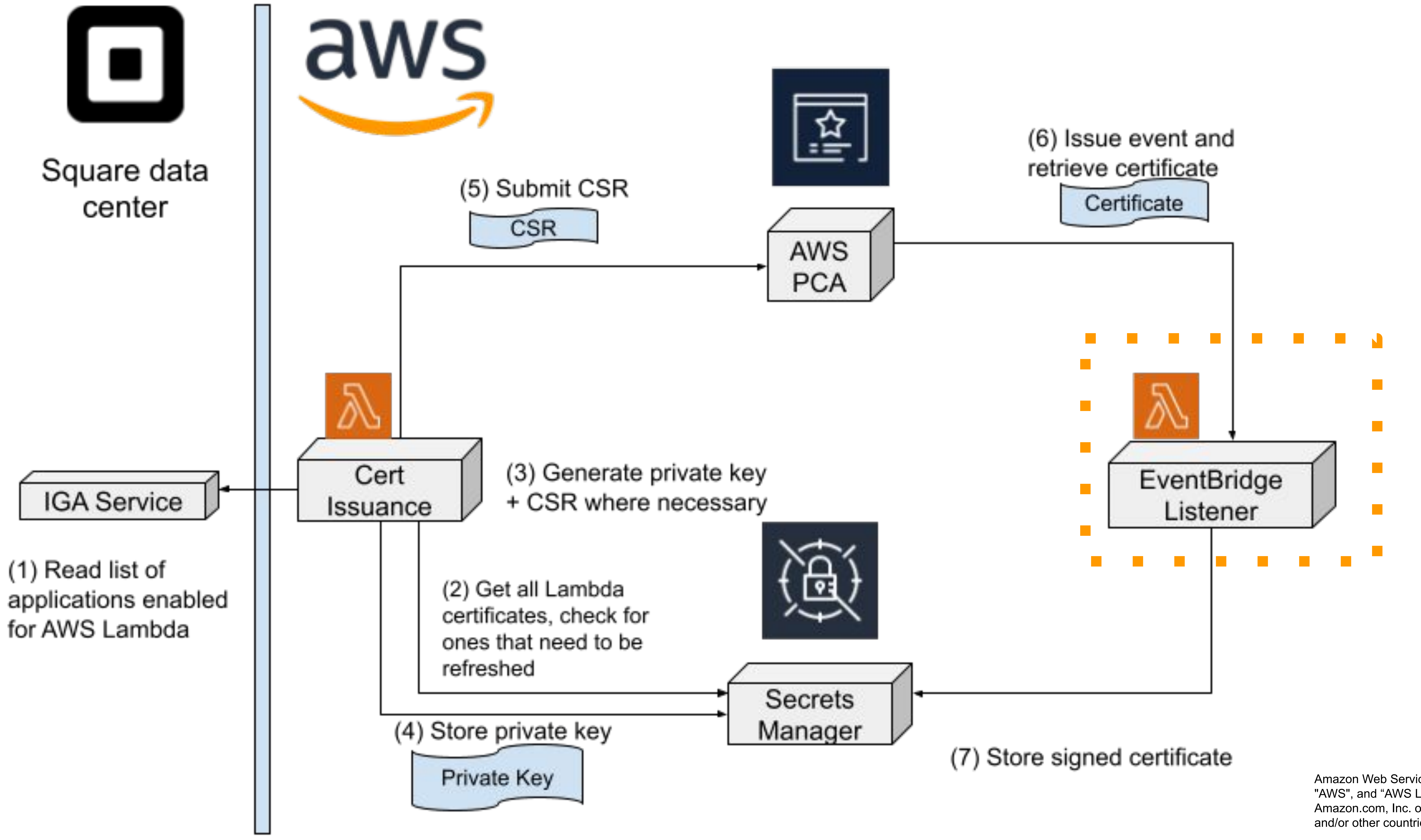
Square data center
AWS
(5) Submit CSR
CSR
AWS PCA
(6) Issue event and retrieve certificate
Certificate
IGA Service
Cert Issuance
(3) Generate private key + CSR where necessary
EventBridge Listener
(1) Read list of applications enabled for AWS Lambda
(2) Get all Lambda certificates, check for ones that need to be refreshed
Secrets Manager
(4) Store private key
Private Key
(7) Store signed certificate

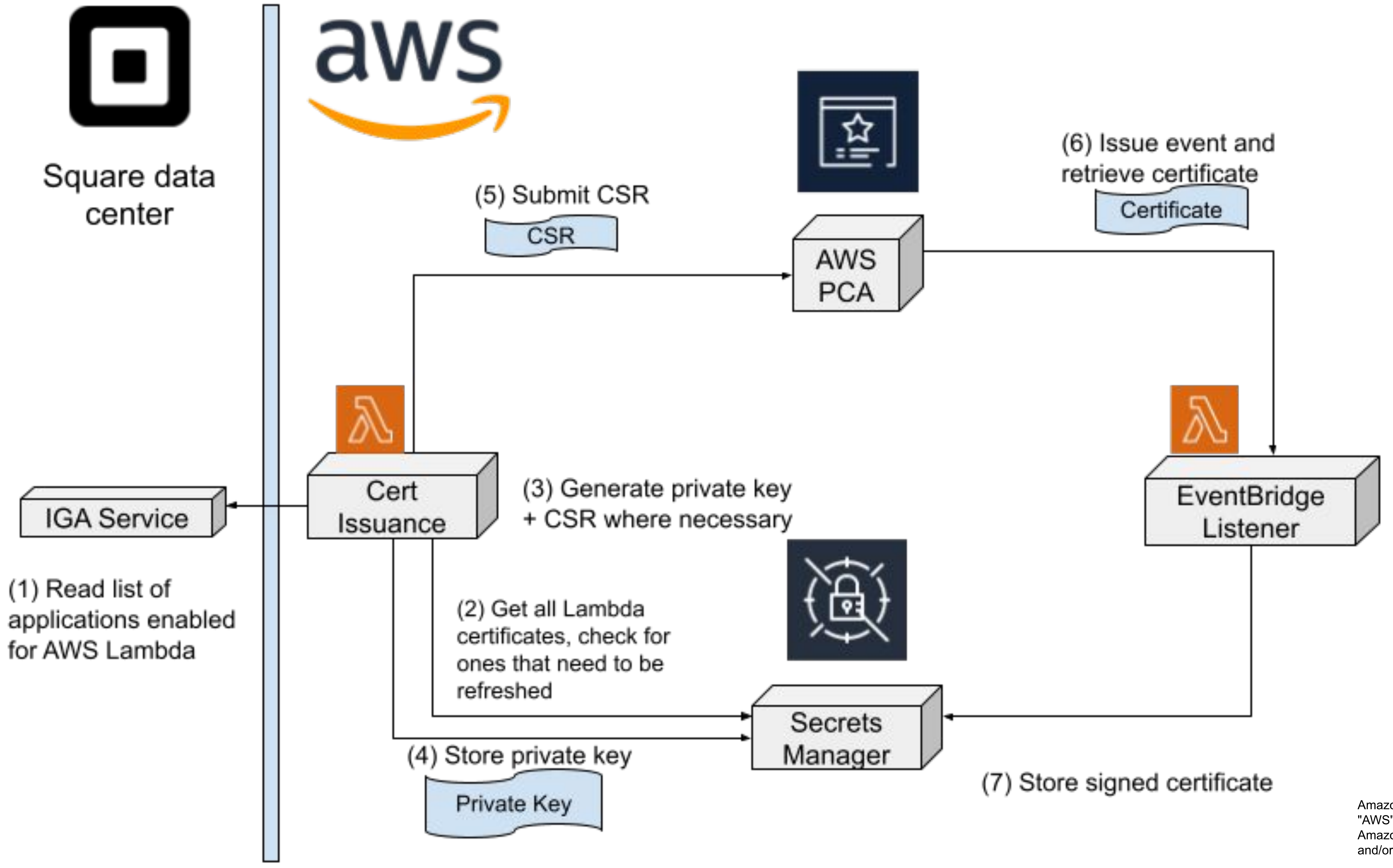


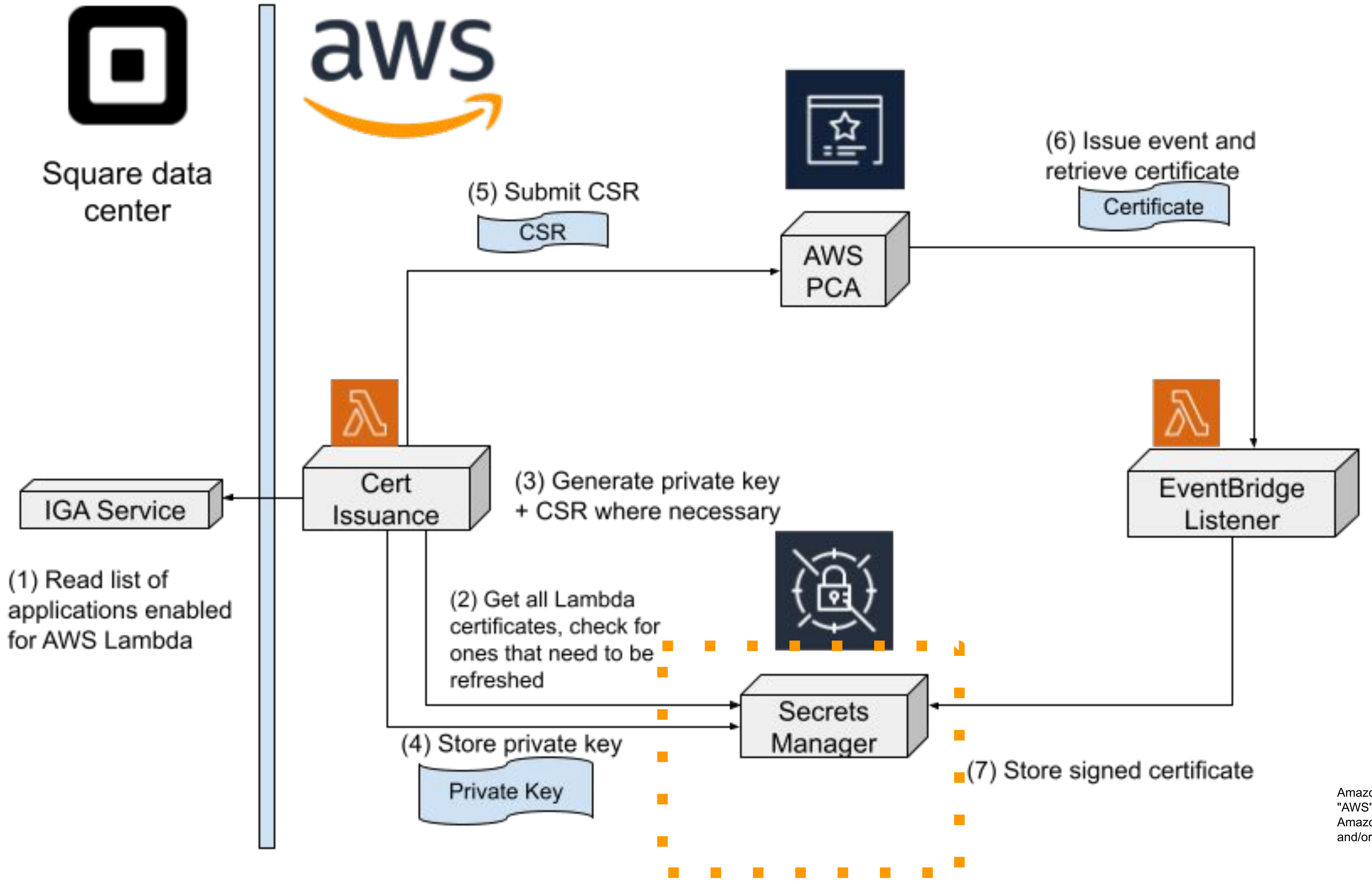
Square data center
aws
(5) Submit CSR
CSR
AWS PCA
(6) Issue event and retrieve certificate
Certificate
IGA Service
Cert Issuance
(3) Generate private key + CSR where necessary
EventBridge Listener
(1) Read list of applications enabled for AWS Lambda
(2) Get all Lambda certificates, check for ones that need to be refreshed
Secrets Manager
(4) Store private key
Private Key
(7) Store signed certificate

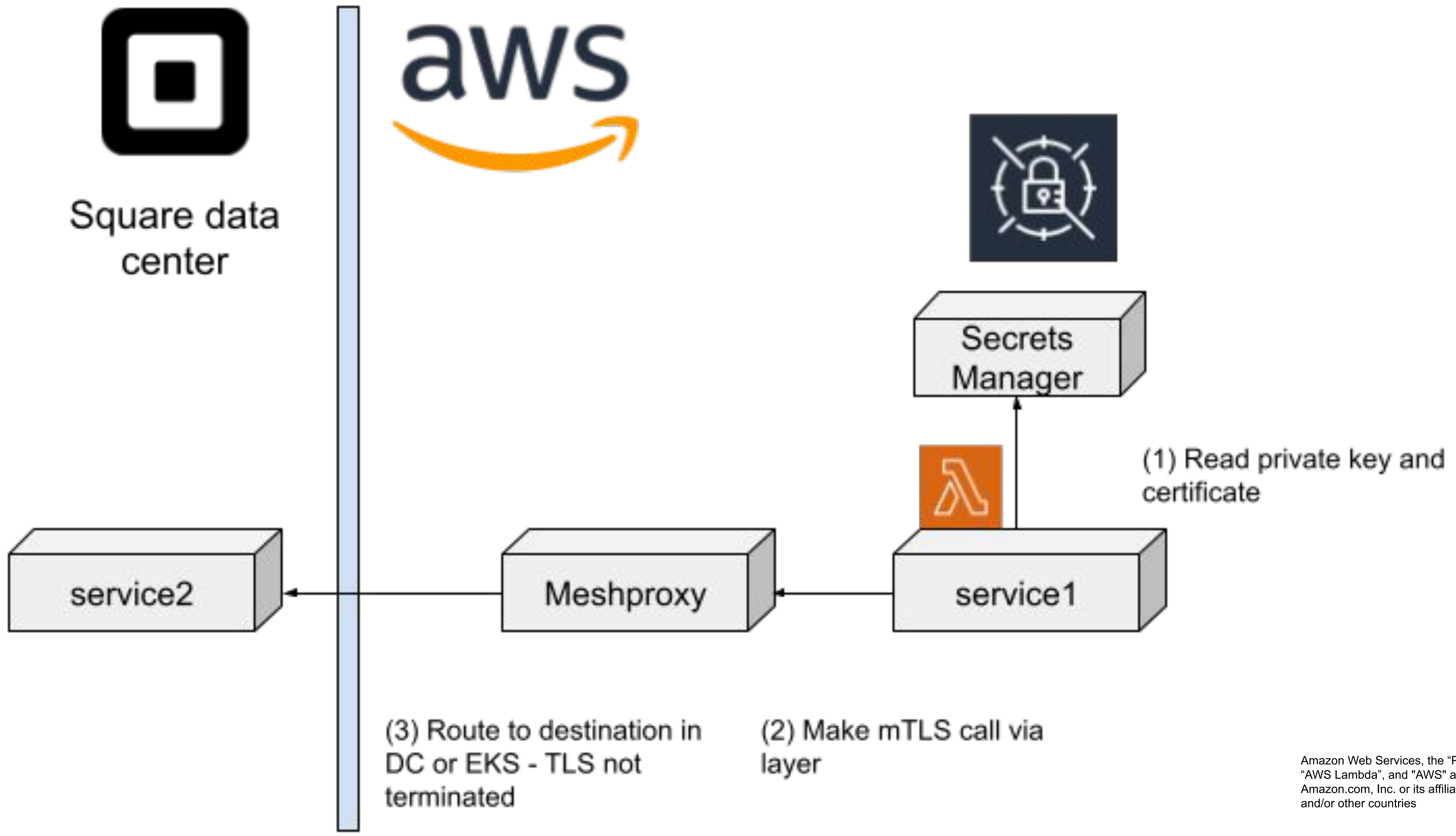


Square data center
IGA Service
(1) Read list of applications enabled for AWS Lambda
Cert Issuance
(2) Get all Lambda certificates, check for ones that need to be refreshed
(3) Generate private key + CSR where necessary
(4) Store private key
Private Key
Secrets Manager
(5) Submit CSR
CSR
AWS PCA
(6) Issue event and retrieve certificate
Certificate
EventBridge Listener
(7) Store signed certificate


Square data center
aws
(5) Submit CSR
CSR
AWS PCA
(6) Issue event and retrieve certificate
Certificate
IGA Service
Cert Issuance
(3) Generate private key + CSR where necessary
EventBridge Listener
(1) Read list of applications enabled for AWS Lambda
(2) Get all Lambda certificates, check for ones that need to be refreshed
Secrets Manager
(4) Store private key
Private Key
(7) Store signed certificate


Square data center
(1) Read list of applications enabled for AWS Lambda
IGA Service
Cert Issuance
(2) Get all Lambda certificates, check for ones that need to be refreshed
(3) Generate private key + CSR where necessary
(4) Store private key
Private Key
(5) Submit CSR
CSR
AWS PCA
(6) Issue event and retrieve certificate
Certificate
EventBridge Listener
Secrets Manager
(7) Store signed certificate


Square data center
(1) Read list of applications enabled for AWS Lambda
IGA Service
Cert Issuance
(2) Get all Lambda certificates, check for ones that need to be refreshed
(3) Generate private key + CSR where necessary
(4) Store private key
Private Key
(5) Submit CSR
CSR
AWS PCA
(6) Issue event and retrieve certificate
Certificate
EventBridge Listener
Secrets Manager
(7) Store signed certificate


Square data center
Secrets Manager
(1) Read private key and certificate
service2
Meshproxy
service1
(3) Route to destination in DC or EKS - TLS not terminated
(2) Make mTLS call via layer

Hello! Lambda calling

- Lambda layer
  - Pulls identity from Secrets Manager
  - Golang process listening on localhost
  - Overloads VerifyPeerCertificate in TLS to perform SPIFFE URI validation
  - Entire process transparent to developers

- Meshproxy
  - Modified version of envoy
  - Pass-through TLS
  - Routes by SNI between DC and EKS

- DC/EKS services
  - ACL check against SPIFFE URI

System Goals

Equal footing with DC security infrastructure …

- **Communication:** Connect to DC/AWS EKS mesh ✅
- **Application secrets:** Access to protected data

…while still maintaining Lambda benefits

- **Speed:** Must support fast response times ✅
- **Scalability:** Must scale with Lambda demand ✅
- **Compatibility:** Must plug into DC infrastructure ✅
- **Availability:** Must be high ✅

Risk Mitigation

Stealing the root

- Attacker would be able to issue certs offline
- Root contained in AWS Private CA
- No intermediates used, leaf issuance off of root

Attacking issuance

- Account locked down
- Audit trail

Stealing identity from a Lambda

- Blast radius limited to 24h, ACL system limits damage
- IAM and SCP

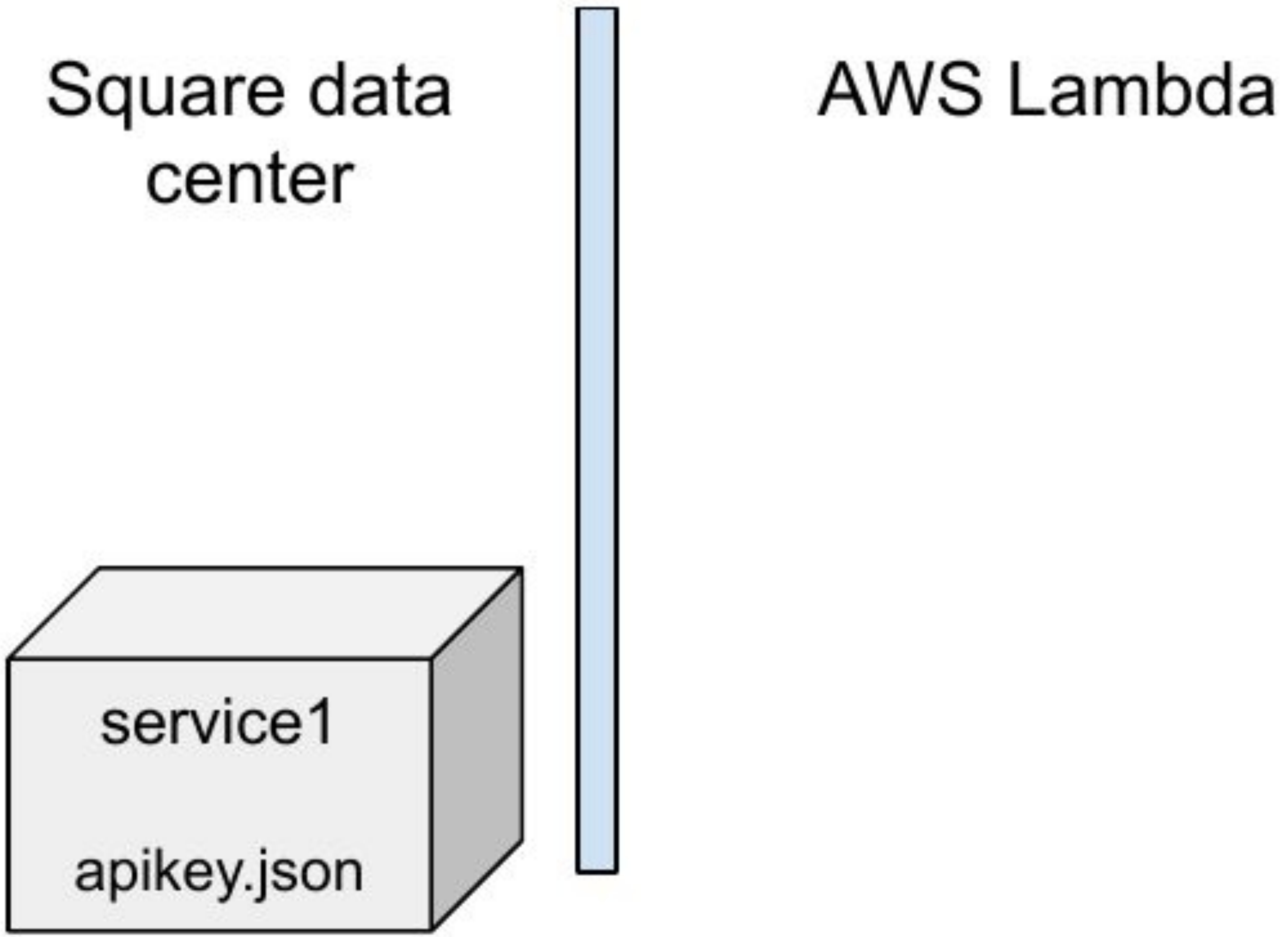
Influenced SPIRE serverless architecture

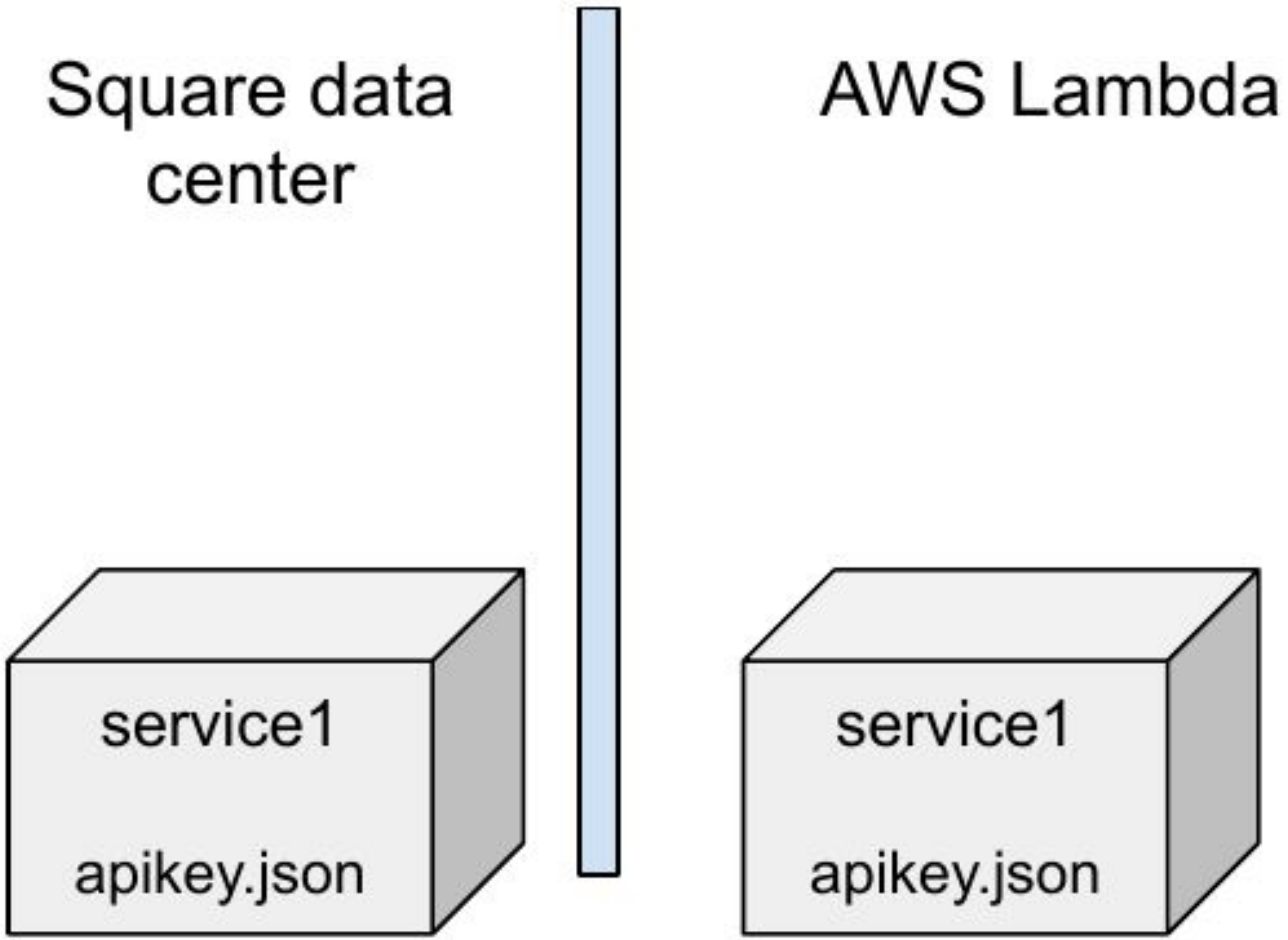
RFC posted after we published our system description

RFC initially pull-style issuance, but has **adopted push-style** based on our implementation

- Implementation of serverless issuance in progress, target release: SPIRE v1.1 September/October 2021
- Square looking to migrate to open source implementation

Solution Pt 2: Syncing application secrets


Square data center
AWS Lambda
service1
apikey.json


Square data center
AWS Lambda
service1
apikey.json
service1
apikey.json

Application Secrets: Keywhiz

- What are secrets? API keys, GPG keys, ...
- Open source: github.com/square/keywhiz
- Secret ownership mapped to microservices
- DC
  - Parallel PKI
  - Syncer daemonset on each server node
  - Syncer has access to all applications' secrets deployed on node
- Integrated web tooling in "Square Console"
  - Self-serve adding secrets
  - Tracking of expiration

Full decentralization?

Evaluating using Secrets Manager directly

- We decided against it - Why?
  - Security teams have expertise in handling centralized secrets
  - Conflicting versions of secrets, e.g.: in multi-cloud + DC scenario
  - No centralized expiration tracking
  - Centralized tooling, such as GPG integration
- Bottom line: too risky, and wanted to do better

Full centralization?

Evaluating DC equivalent

- No deploy moment
- Can’t block on invoke
- Bottom line: not compatible

Application secrets decision

BUILD

DC benefits + cloud native features

Security Boundaries

- DC: Node syncers with wide ranging access
- Lambda
  - An opportunity to reduce blast radius
  - No concept of "node"

An Idea

- SPIFFE identity work unlocked infrastructure capabilities (!)
- Added SPIFFE support to Keywhiz
- Client-side syncer uses service identity

Reduce exposure

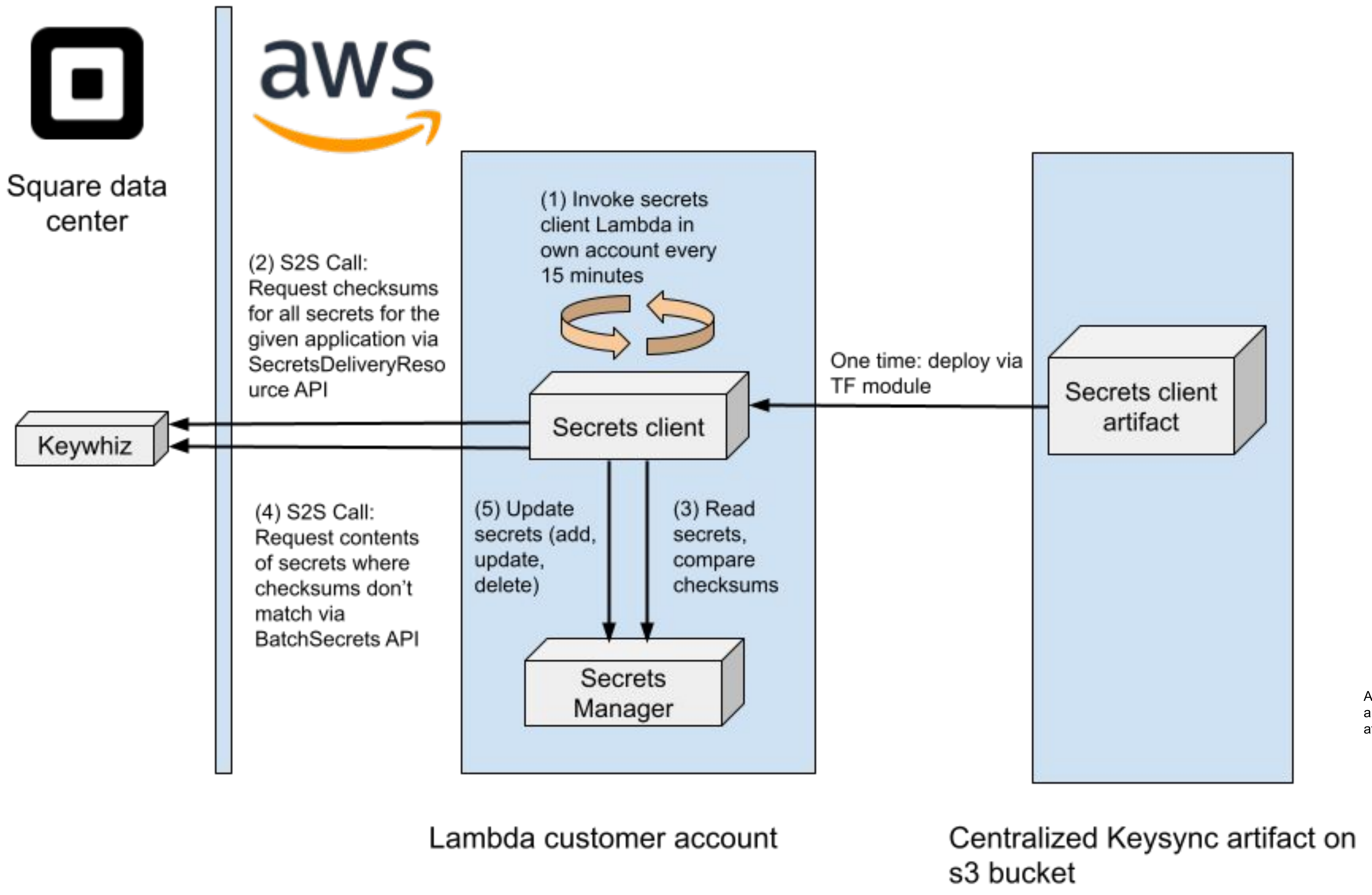
- Opt-in to secrets vs. opt-out
- Action required via Square Console

Secrets Availability

- Observation: Secrets are updated rarely
- Majority of syncing operations: no-op
- Reliable cache > blocking on updates
- Unscheduled update: Trigger syncer

Storage

- Fast reads: Secrets Manager
- No DC dependencies
- Default encryption key enforces account boundaries


Square data center
aws
(1) Invoke secrets client Lambda in own account every 15 minutes
(2) S2S Call: Request checksums for all secrets for the given application via SecretsDeliveryResource API
One time: deploy via TF module
Keywhiz
Secrets client
Secrets client artifact
(4) S2S Call: Request contents of secrets where checksums don't match via BatchSecrets API
(5) Update secrets (add, update, delete)
(3) Read secrets, compare checksums
Secrets Manager
Lambda customer account
Centralized Keysync artifact on s3 bucket

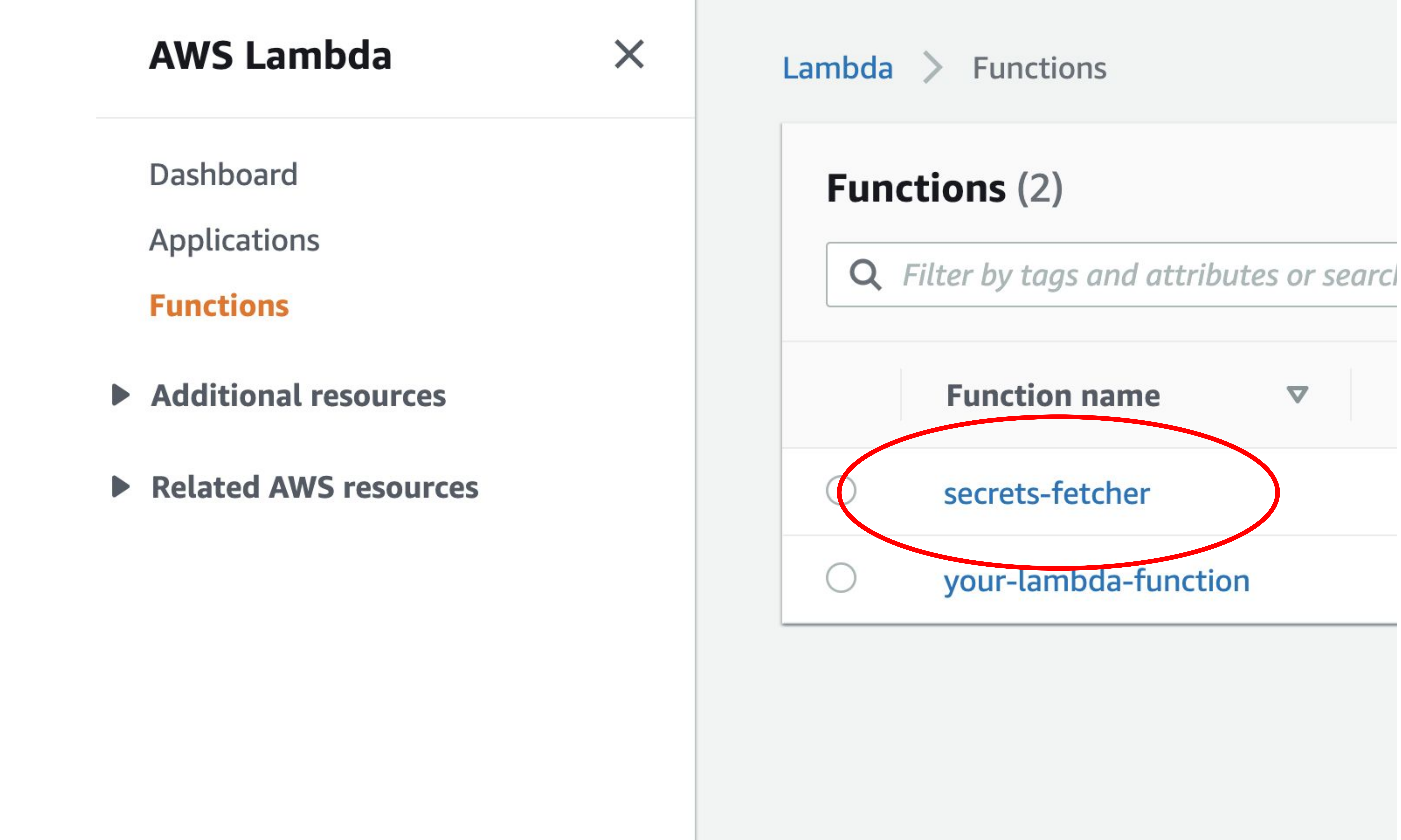
How to onboard

- Terraform module
- To apply: 24 lines
- Secrets Manager: direct access

- Implemented as Lambda
- Uses Lambda workload identity
- Client operates in each enabled account

Production
Assign Common Secret
Update infras...
Add
1
2
test-secret-1
DC
Info
test-secret-2
DC
Lambda
Info

Update secret infra availability
Action
Add secret to infra...
Infra
Lambda
3
test-secret-1
4
DC
Lambda
Update Assignments


AWS Lambda

Dashboard

Applications

Functions

Additional resources

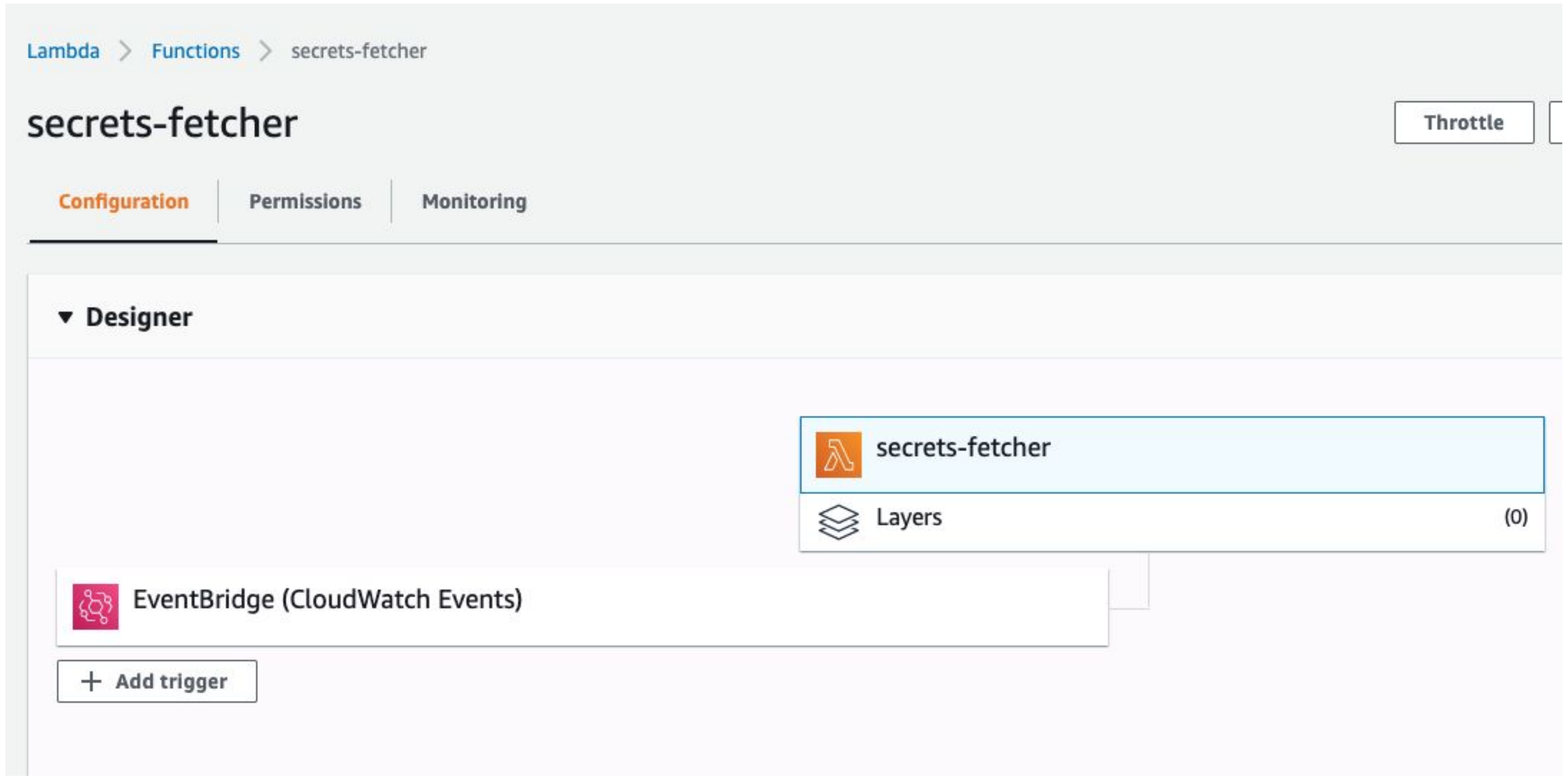
Related AWS resources

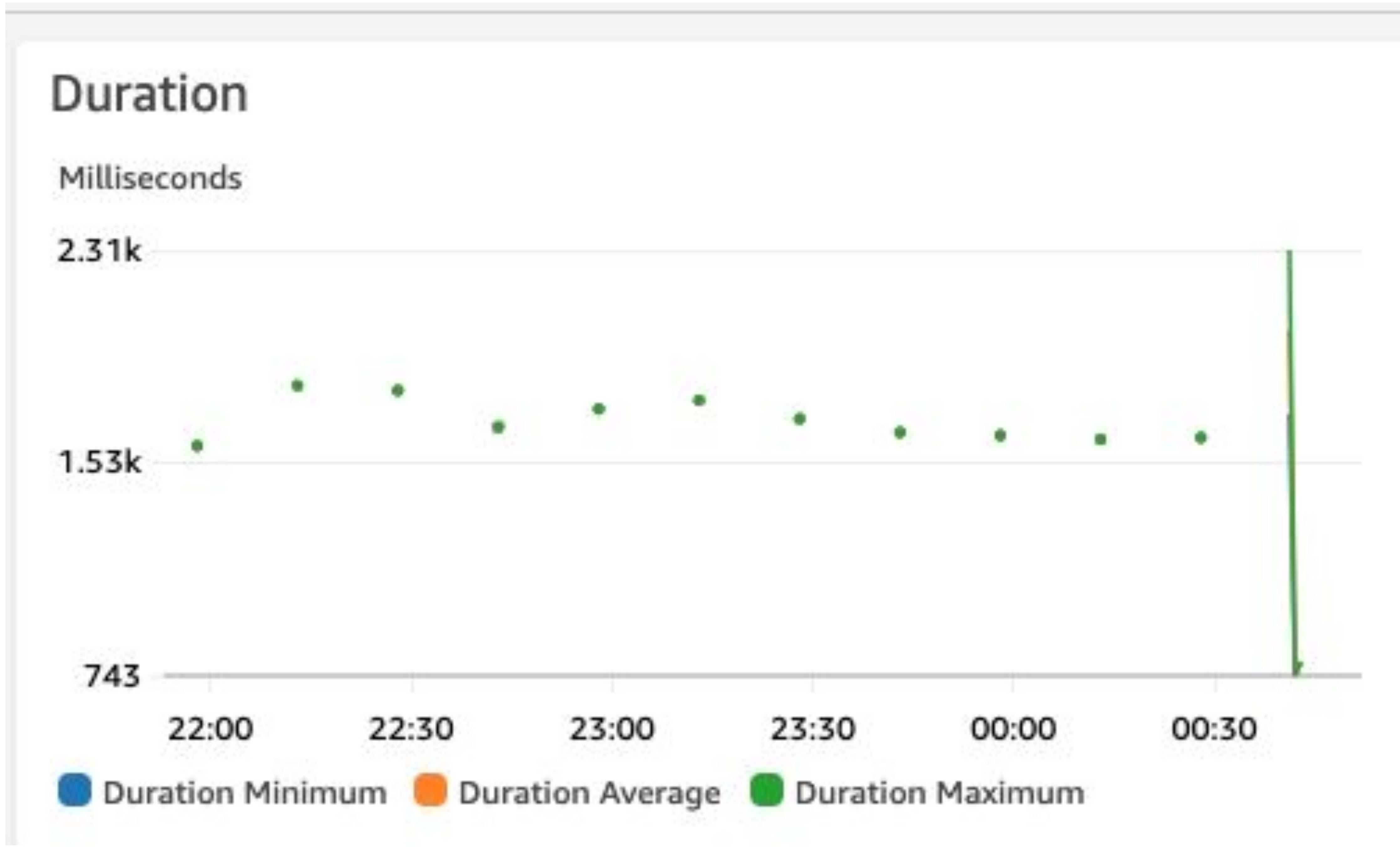
Lambda > Functions

Functions (2)

Filter by tags and attributes or searc

Function name
secrets-fetcher
your-lambda-function


Lambda > Functions > secrets-fetcher
secrets-fetcher
Throttle
Configuration
Permissions
Monitoring
Designer
secrets-fetcher
Layers
(0)
EventBridge (CloudWatch Events)
+ Add trigger


Duration
Milliseconds
2.31k
1.53k
743
22:00
22:30
23:00
23:30
00:00
00:30
Duration Minimum
Duration Average
Duration Maximum

System Goals

Equal footing with DC security infrastructure …

- **Communication:** Connect to DC/AWS EKS mesh ✅
- **Application secrets:** Access to protected data ✅

…while still maintaining Lambda benefits

- **Speed:** Must support fast response times ✅
- **Scalability:** Must scale with Lambda demand ✅
- **Compatibility:** Must plug into DC infrastructure ✅
- **Availability:** Must be high ✅

Risk Mitigation

Access all secrets

- Attack Keywhiz
- ACL system blocks access

Compromise syncer

- CI/CD pipeline
- S3 object version

Access individual secrets

- Secrets tied to identity
- Lambda secret access $\subseteq$ service's secrets

Present and Future

- Secrets in sync between DC and Lambda in production
- SPIFFE support in Keywhiz offers interoperability with new environments

Key Learnings

How we knew it worked

- Identity issuance and secrets used in production
- SPIRE is implementing serverless support following our model

Summary

- Developers want Lambdas, whether you're ready or not
- Support your developers with infrastructure they already know
  - mTLS Envoy service mesh
  - Keep secrets in sync between multiple environments
- Hybrid environments are hard
  - “Moving” to the cloud means operating in two environments
  - This challenging interim state can last years
  - Services in the cloud will rely on DC
  - Use best of both: environments should support each other, do not block

Thank you

Michael Weissbacher
@mweissbacher

References

Square blog posts covering this presentation

- Providing mTLS Identities to Lambdas
- Expanding Secrets Infrastructure to AWS Lambda

Related Square blog posts

- Enabling Serverless Applications at Square
- Using AWS Lambda Extensions to Accelerate AWS Secrets Manager Access
- Adopting AWS VPC Endpoints at Square

References

Service identity

- Envoy
- SPIFFE
- SPIRE RFC for serverless architecture
- AWS Certificate Manager Private Certificate Authority
- What is mutual TLS?

Application secrets

- What is AWS Secrets Manager?
- Keywhiz

References

Lambda

- Understanding Container Reuse in AWS Lambda
- Behind the scenes, AWS Lambda
- Firecracker - AWS Blog