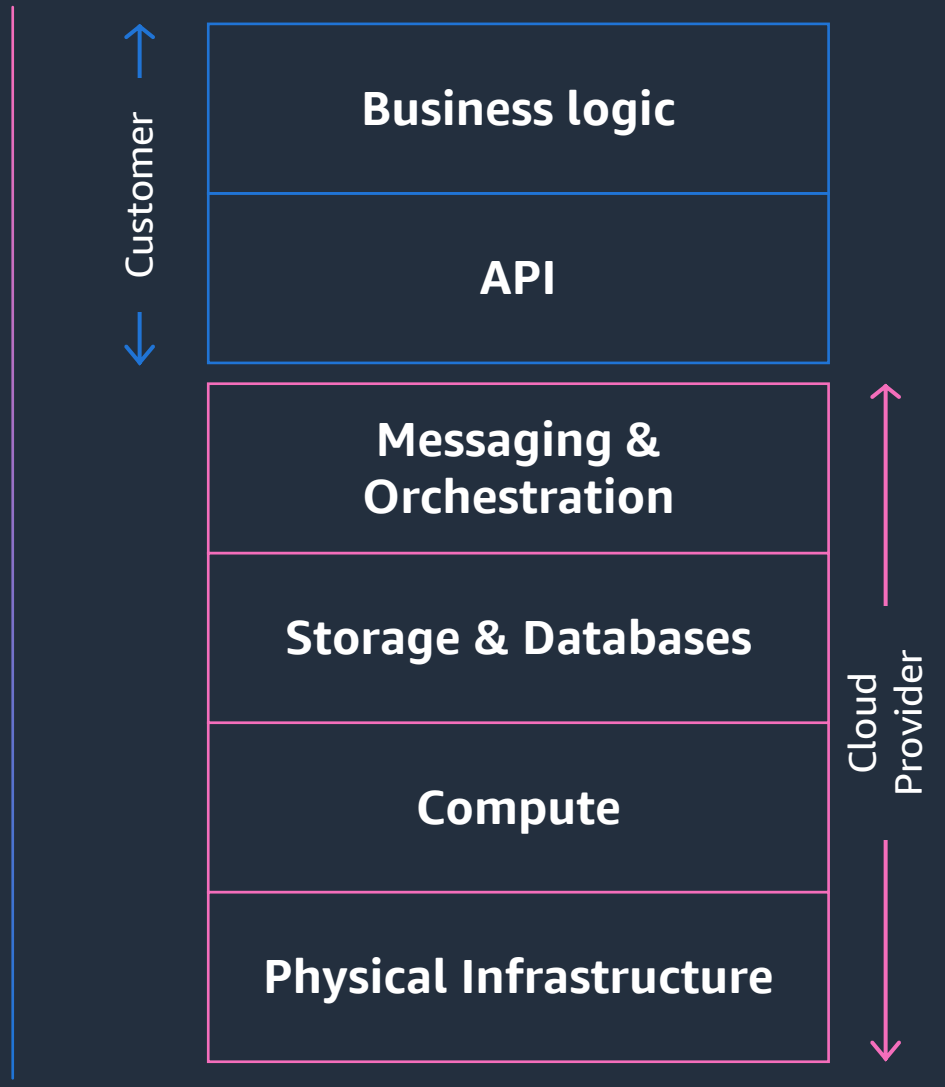
Securing Serverless

What Changes and What Doesn't

Pouya Ghotbi

What is Serverless?

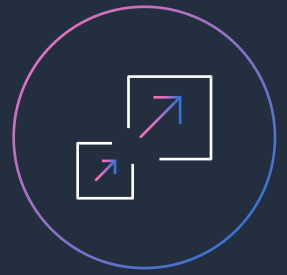
Serverless services simplify the management and scaling of cloud applications by shifting undifferentiated operational tasks to the cloud provider so **development teams can focus on writing code** that solve business problems



Why serverless

TAKE FULL ADVANTAGE OF THE CLOUD TO MODERNIZE APPLICATIONS AND ACCELERATE INNOVATION

No infrastructure provisioning, no management



Automatic scaling

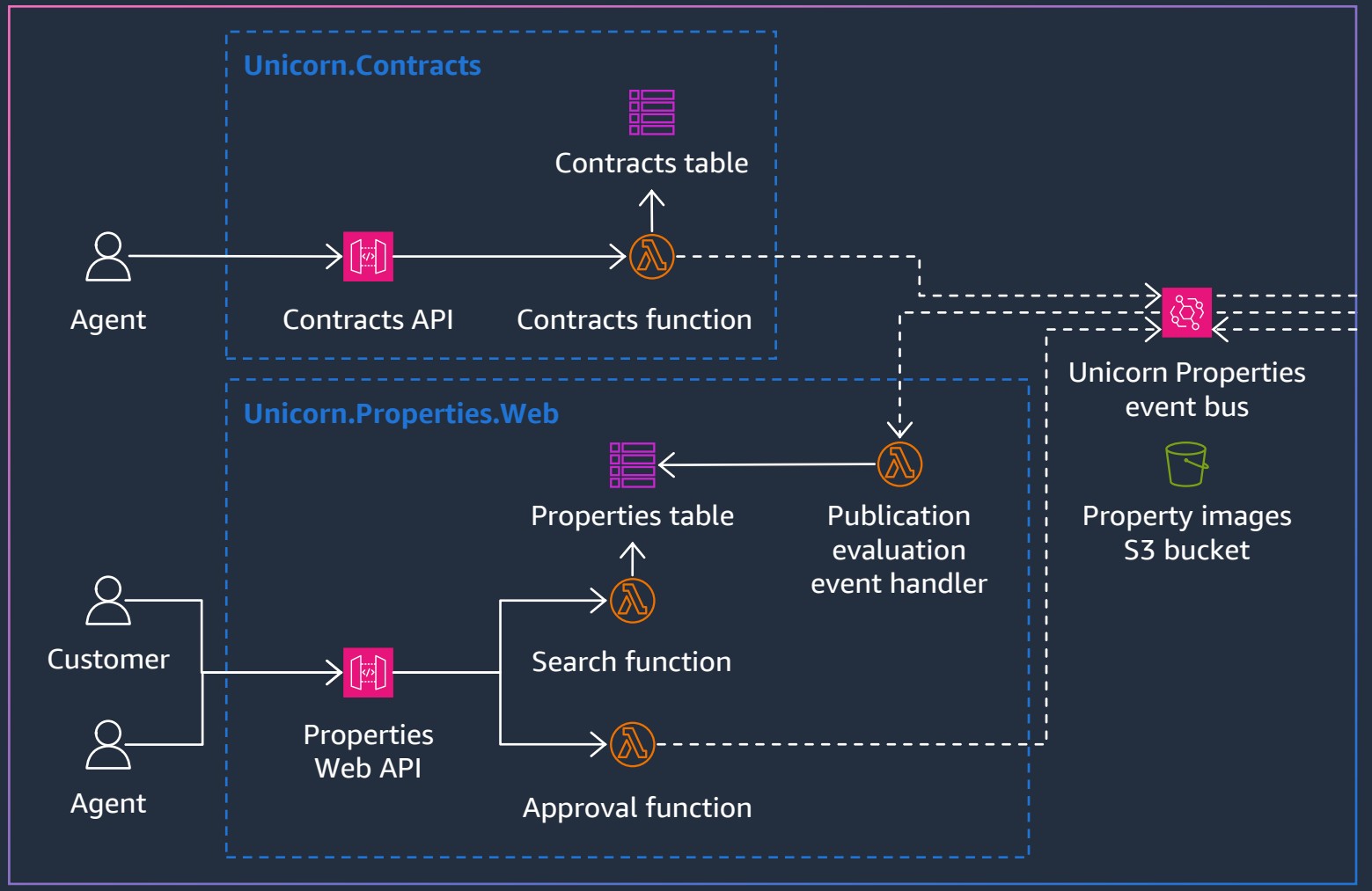
Pay for value

Highly available and secure

Serverless is the fastest way to build modern apps

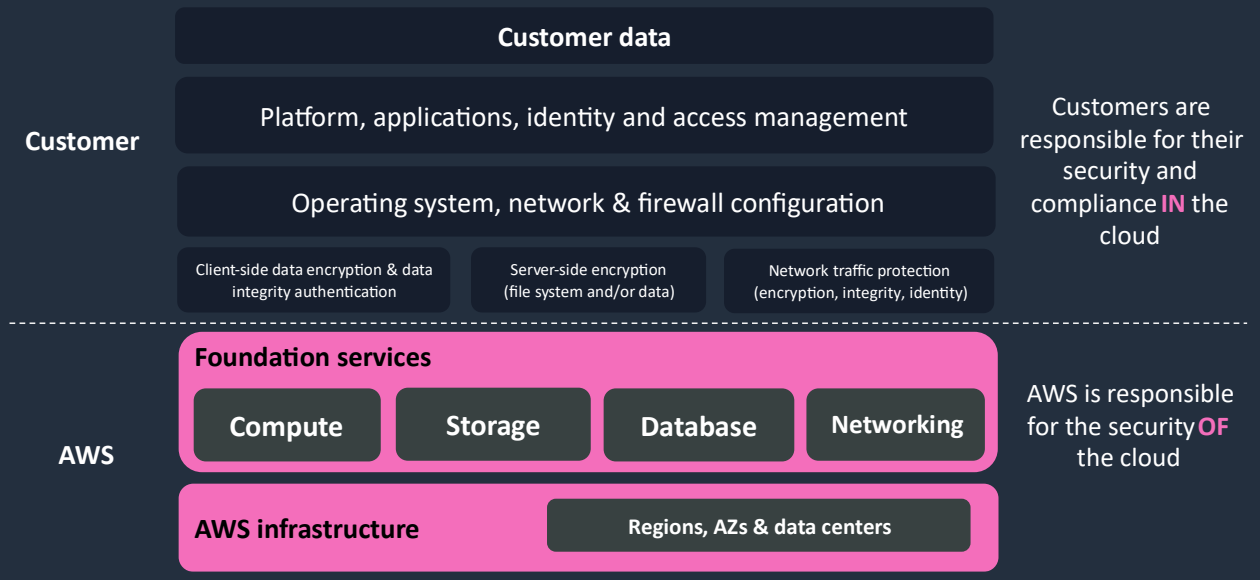
SMALL APP COMPONENTS LOOSELY COUPLED THROUGH EVENTS

Highly responsive, fault tolerant asynchronous applications

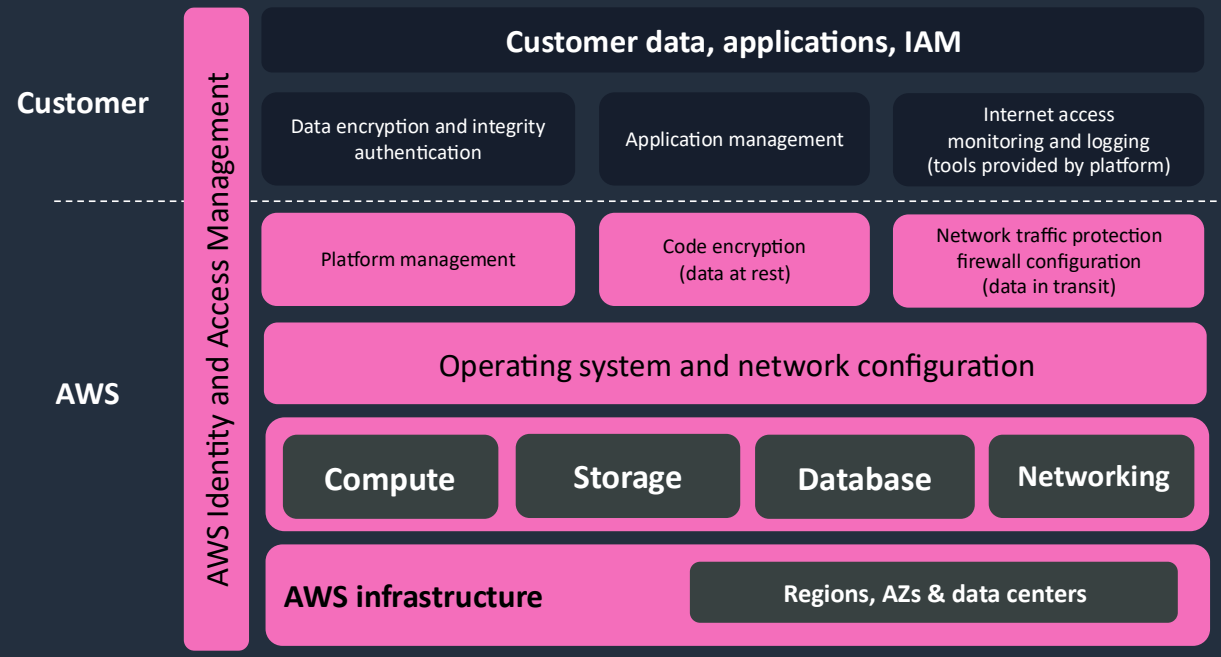


Shared responsibility model

IaaS



Serverless



Types of serverless services

Compute

- Functions-as-a-Service (FaaS)
  - AWS Lambda, Azure Functions
- Serverless containers
  - AWS Fargate, Google Cloud Run
- Edge functions
  - Cloudflare Workers, AWS Lambda@Edge

Storage & Databases

- Object storage
  - Amazon S3, Azure Blob Storage
- Serverless databases
  - Amazon DynamoDB, Azure Cosmos DB
  - Firebase Realtime Database

Events & Messaging

- Event buses & queues
  - Amazon EventBridge, Azure Event Grid, Google Eventarc
- Message queues
  - Amazon SQS, Google Pub/Sub, Azure Service Bus

Identity & Security

- Identity providers
  - Amazon Cognito, Firebase Auth
- Secrets management
  - AWS Secrets Manager, Google Secret Manager

Application Orchestration

- Workflow engines
  - AWS Step Functions, Azure Durable Functions
- Backend-as-a-Service
  - Firebase, Supabase

What else is different?

Ephemerality

Diffused perimeter

Fine-grained control

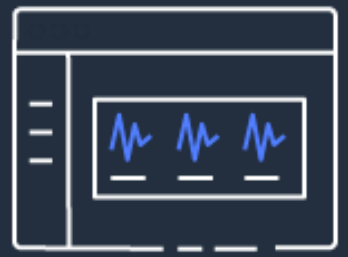
Tooling

What stays the same?

Securing data

Quality code

Least privilege



Monitoring

What Changes	What Stays the Same
You don’t manage infrastructure	You still need secure coding practices
IAM and event triggers are the attack surface	Least privilege is still critical
Logs come from services, not OS agents	You still need to monitor and alert

OWASP Top 10 – Serverless interpretation

- S1-Function Event Data Injection
- S2-Broken Authentication
- S3-Insecure Serverless Deployment
- S4-Over-Privileged Function Permissions
- S5-Inadequate Function Monitoring and Logging
- S6-Insecure Third-Party Dependencies
- S7-Improper Exception Handling
- S8-Misconfigured CORS
- S9-Insecure CI/CD Configuration
- S10-Lack of Security

Example: Node.js Preinstall Attack in Serverless Deployments

Context

- **What it is:**
  - Malicious `preinstall` scripts in NPM packages execute during function deployment.
- **Why it's dangerous:**
  - Runs before runtime monitoring kicks in, often undetected in CI/CD pipelines.
- **Real-world impact:**
  - Attackers exfiltrate secrets, scan environments, or embed persistent payloads.
- **Why serverless is at risk:**
  - Serverless platforms often install dependencies during deployment with limited visibility.

Mitigations

- Audit and pin all dependencies

- Use package-lock.json to lock dependency trees

- Scan builds with tools like Snyk or npm audit

- Disable lifecycle scripts in deployment (e.g. --ignore-scripts)

Serverless security best practices

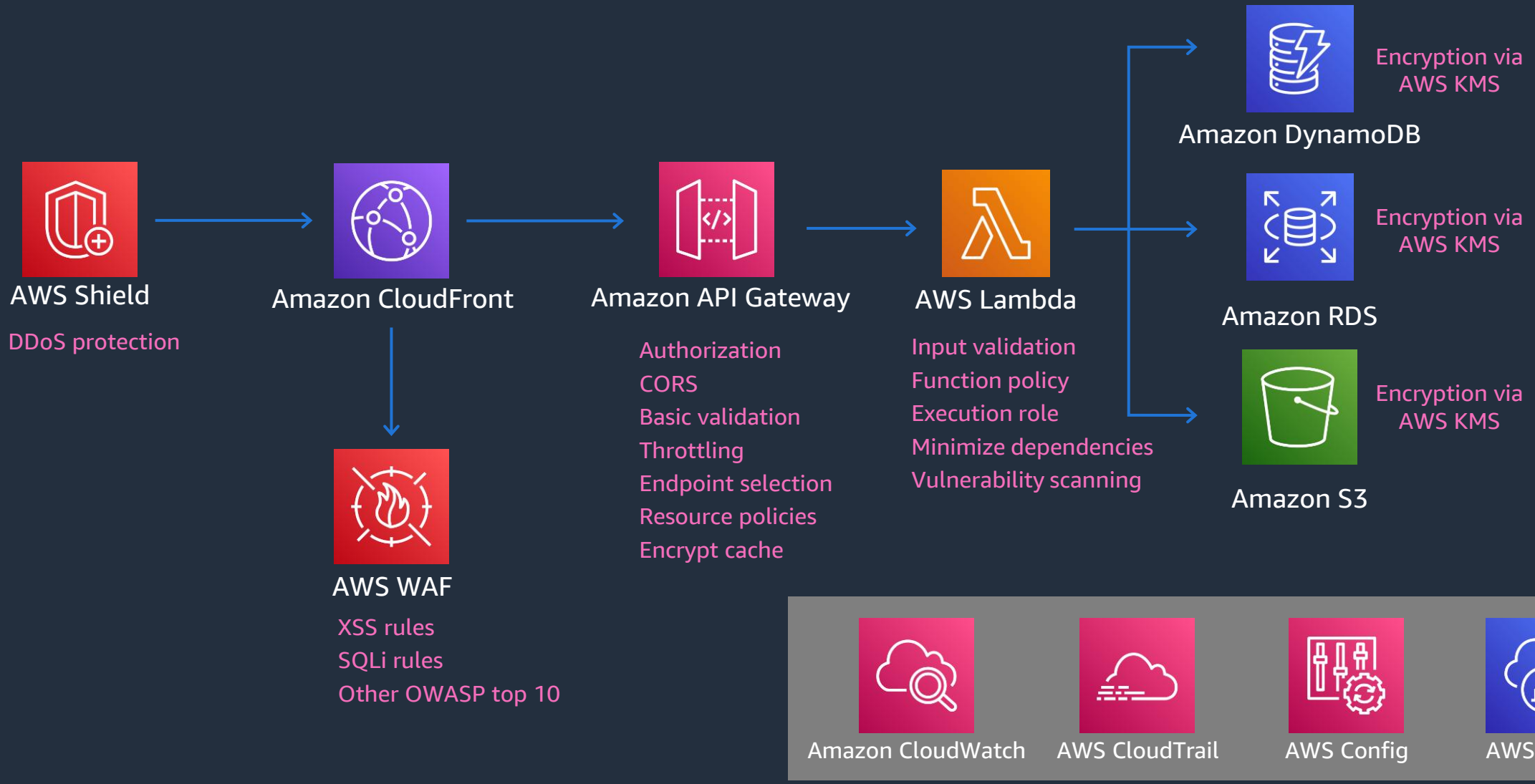
Use authentication and authorization mechanisms

Data encryption and integrity

Monitoring, logging, and configuration management

Denial of service and infrastructure protection

A sample use case...



Key takeaways

Serverless shifts the attack surface, not the responsibility

Secure your code, identity, and configuration

Automate and monitor everything

Build security into every event trigger and permission model

Thank you!

Pouya Ghotbi

linkedin.com/in/pouyaghotbi