AWS re:Invent

NOV. 29 – DEC. 3, 2021 | LAS VEGAS, NV

SVS309-R2

Architecting secure serverless applications

Josh Kahn (he/him)
Principal Solutions Architect
Tech Leader – Serverless
AWS

© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Security is everyone's responsibility

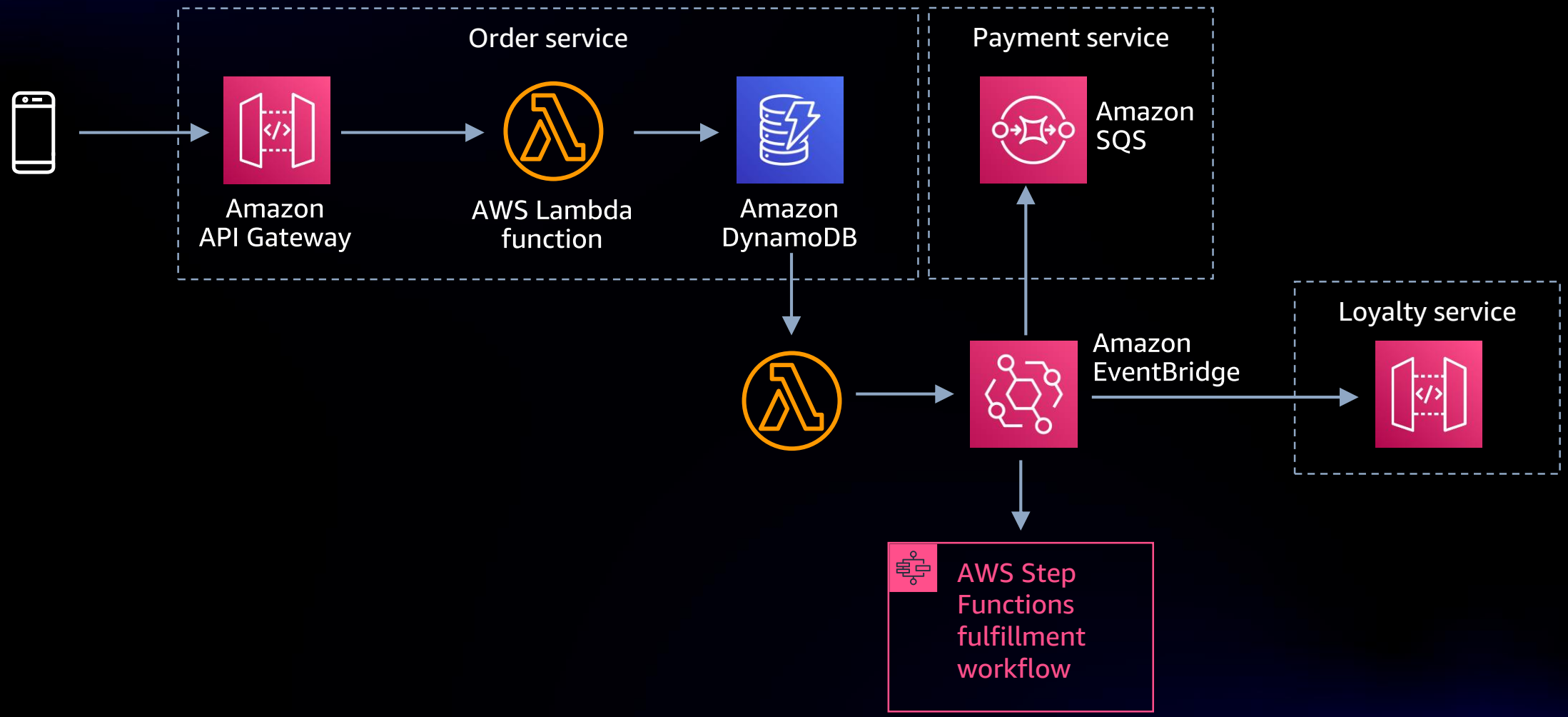
© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Four principles to securing serverless applications

1. Understand **shared responsibility model**
2. Grant **least privilege**
3. Implement **defense in depth**
4. Secure your **software supply chain**

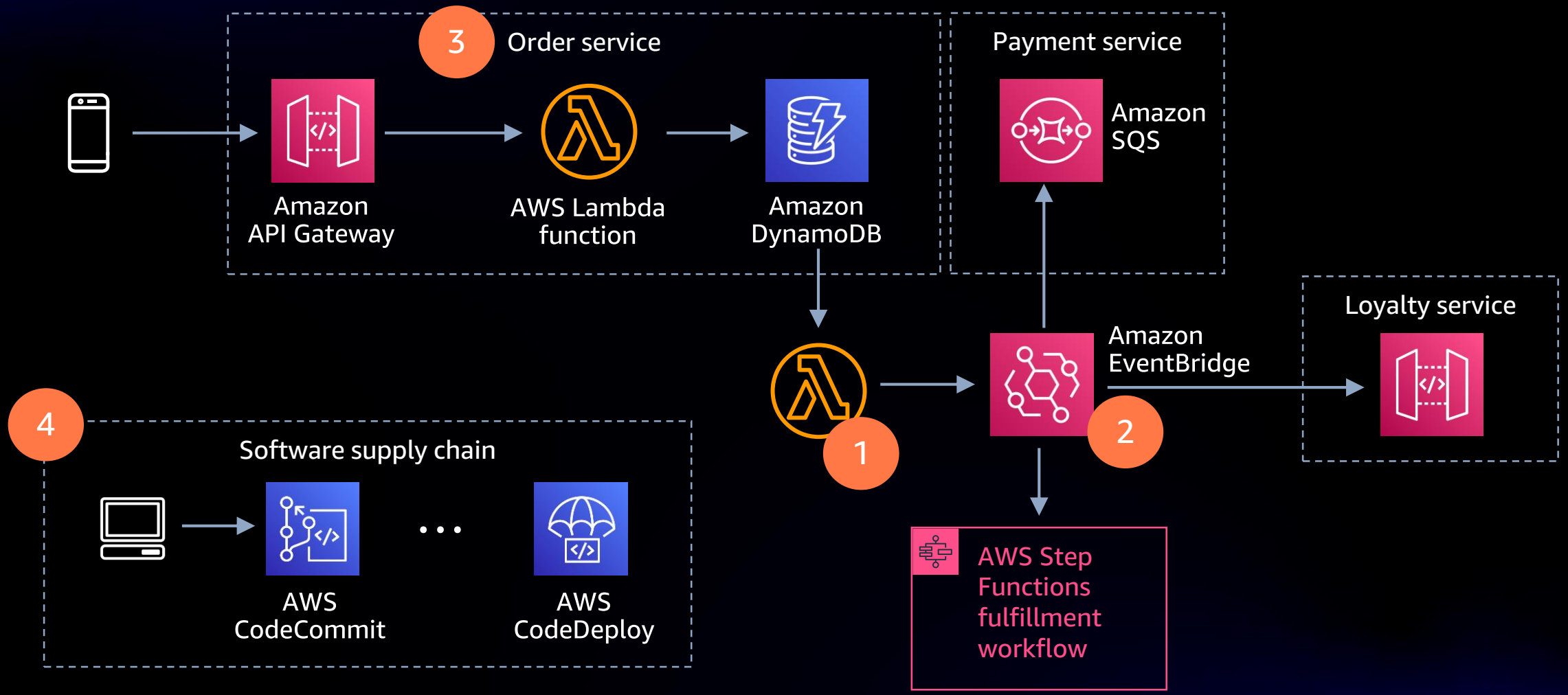
© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Let’s apply these principles to a sample architecture



© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Let’s apply these principles to a sample architecture



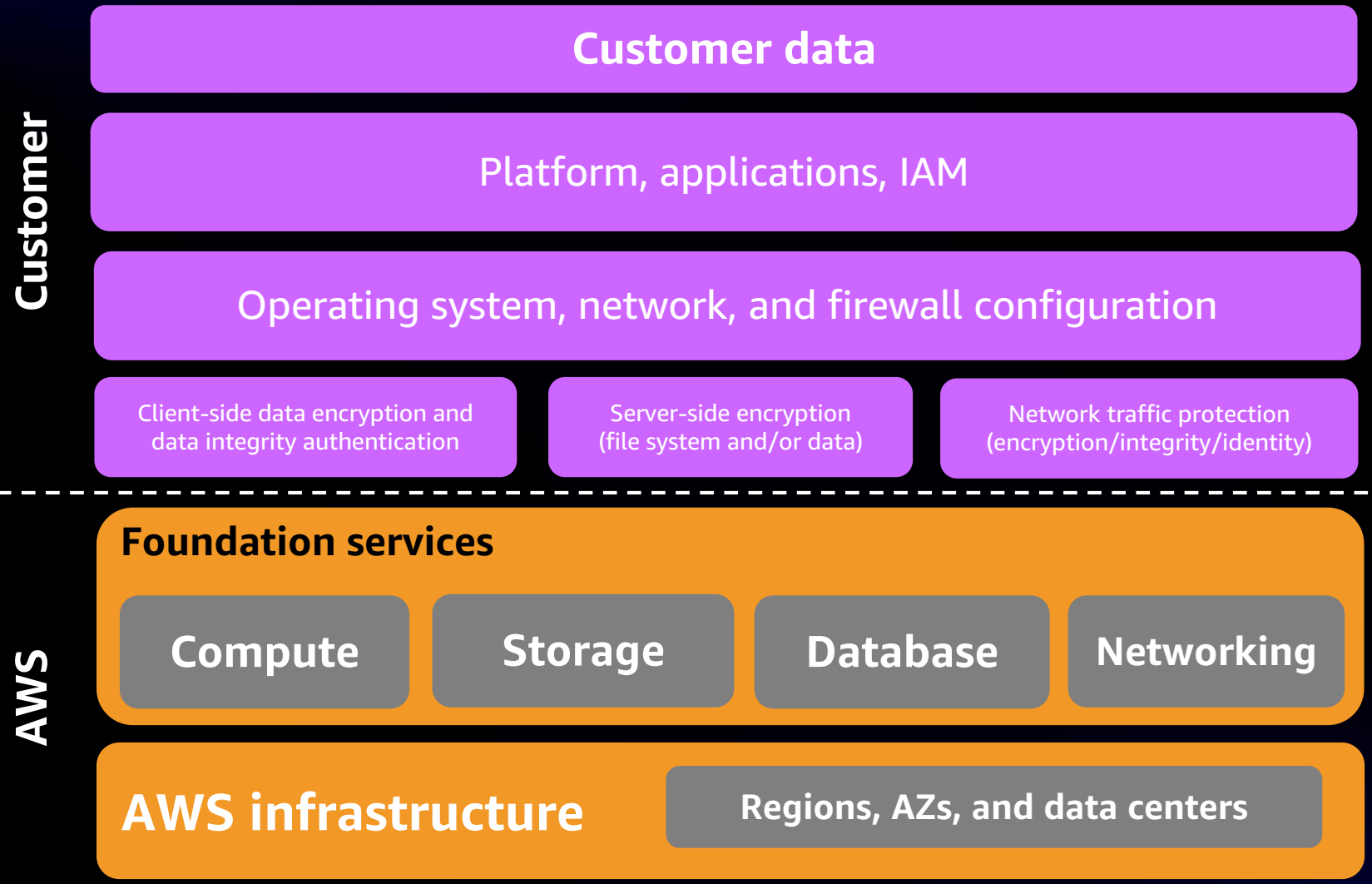
© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

SECURITY PRINCIPLE #1

Understand shared responsibility with AWS

© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Shared responsibility with AWS



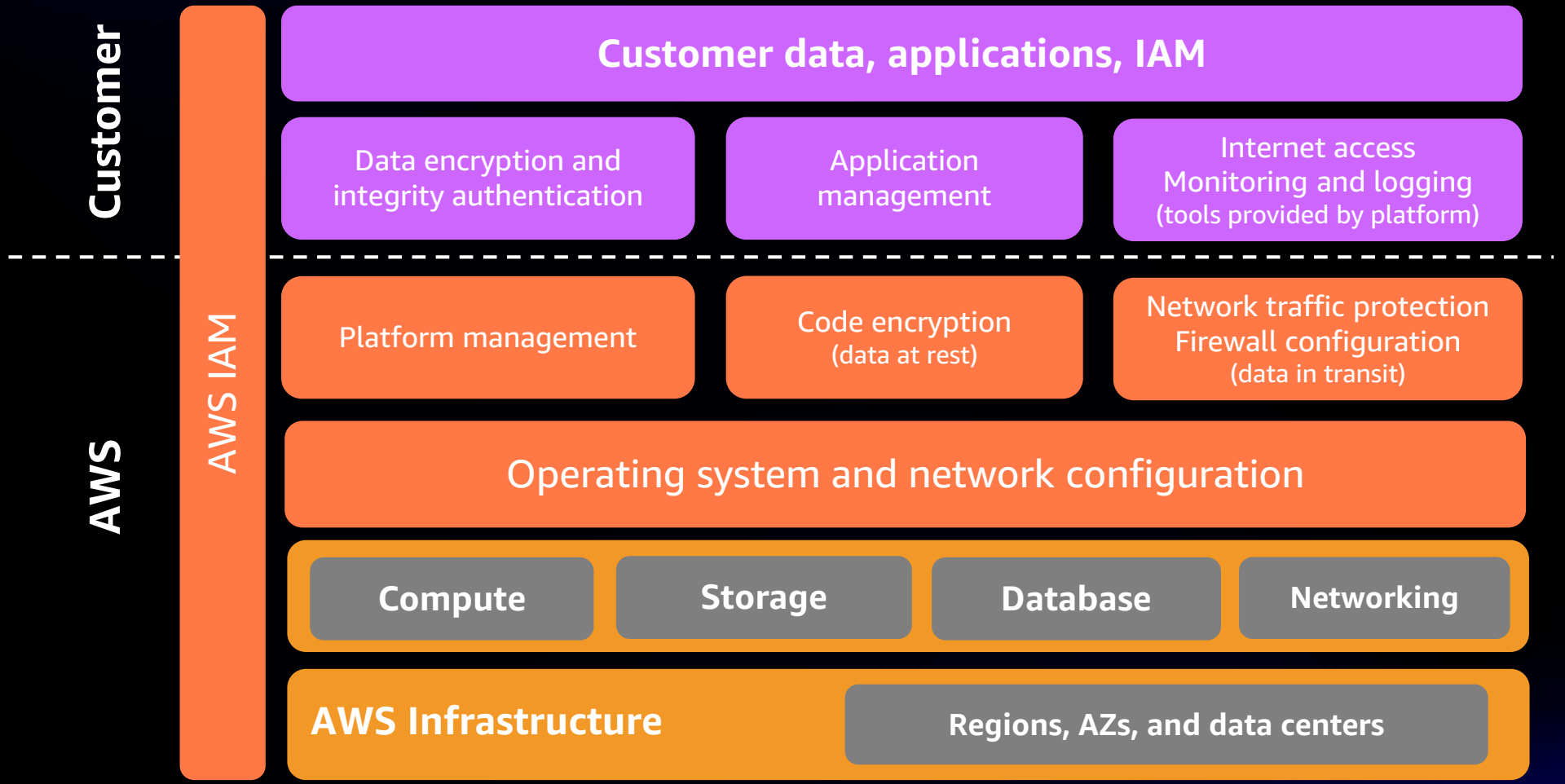
Customers are responsible for their security and compliance **IN** the cloud

AWS is responsible for the security **OF** the cloud

aws

© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

With serverless, AWS takes a greater share of responsibility



© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

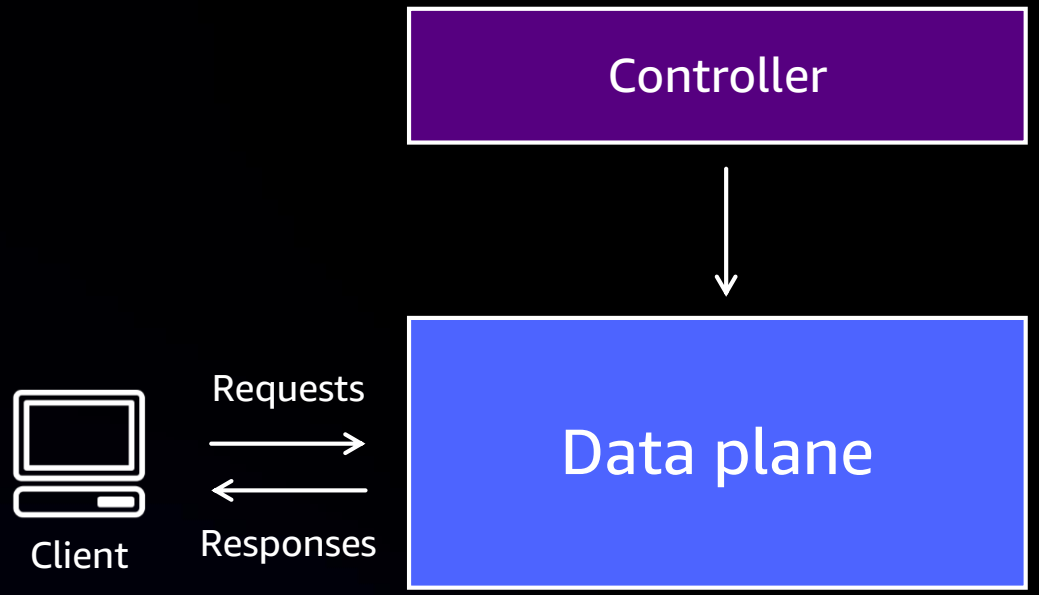
Lambda service composed of control and data planes

Control plane

- Management APIs, such as
  - `CreateFunction`
  - `UpdateFunctionCode`
- Requires IAM permission to access

Data plane

- Invoke Lambda function via `Invoke`
- IAM permission or resource policy
- When invoked, executes code on
  - Existing execution environment, if exists
  - New environment, after allocation



© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Secure Lambda functions with AWS IAM

Function policy

- Defines how function can be invoked
- Supports cross-account access
- Used for synchronous and asynchronous invocations

“Actions on API Gateway A can invoke Lambda function B”

Execution role

- Defines which AWS resources can access via IAM
- Used for poll-based invocations (Lambda polling)

“Lambda function A can write data to DynamoDB Table B”



© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Secure other resources with AWS IAM

Resource policy

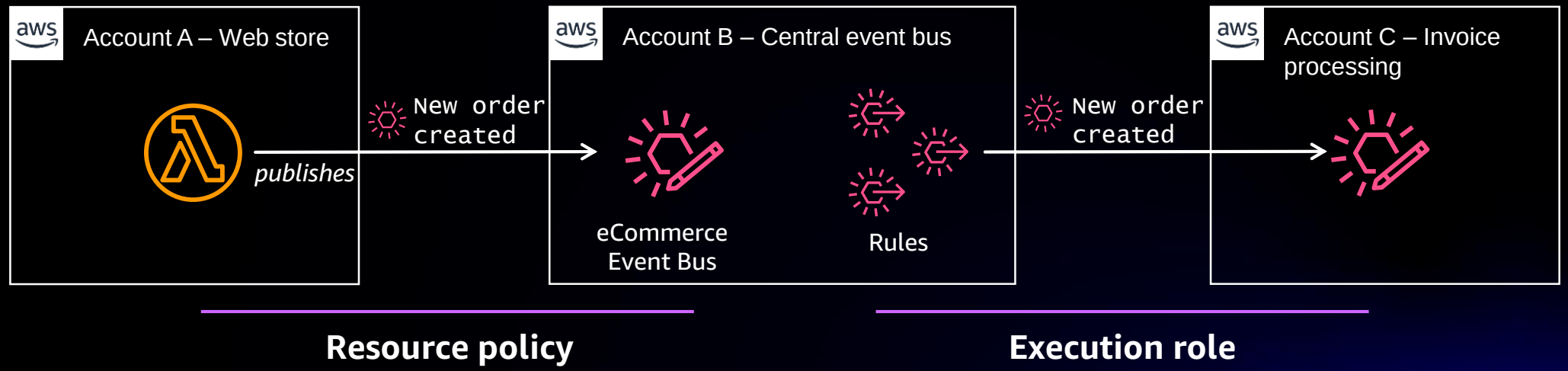
- Defines access to event bus, for example `PutEvent`
- Use conditions to scope permissions
- Allows cross-account access

"Principals in Account A can publish new order events to the event bus in Account B"

Execution role

- Assumed by event bus to determine access to AWS resources
- Allows cross-account access

"Event Bus A can publish events to Event Bus B"



© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

SECURITY PRINCIPLE #2

Grant least privilege

© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

What does it mean to grant least privilege?

- Grant only the **essential privileges** needed to perform intended work
- Attach to function via AWS IAM **execution role**
  - Prefer **unique role** per function
  - Assign **fine-grained permissions**
  - Enforce permission boundaries
- Be specific: Identify limited set of resources and actions allowed
  - Scrutinize use of “*”



```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Action": [
                "xray:PutTraceSegments",
                "xray:PutTelemetryRecords"
            ],
            "Resource": "*",
            "Effect": "Allow"
        },
        {
            "Action": "s3:PutObject",
            "Resource": [
                "arn:aws:s3:::my-bucket",
                "arn:aws:s3:::my-bucket/*"
            ],
            "Effect": "Allow"
        }
    ]
}
```

© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

AWS SAM accelerates policy, role development

```yaml
SaveToDynamoDBFunction:
  Type: AWS::Serverless::Function
  Properties:
    Handler: app.handler
    Runtime: nodejs14.x
    Policies:
      - DynamoDBWritePolicy:
          TableName: !Ref DDBTable
    Events:
      GetItem:
        Type: HttpApi
        Properties:
          Path: "/todo/{id}"
          Method: GET
```




```json
{
  "Version": "2012-10-17",
  "Statement": [{
    "Effect": "Allow",
    "Action": [
      "dynamodb:PutItem",
      "dynamodb:UpdateItem",
      "dynamodb:BatchWriteItem"
    ],
    "Resource": "<DYNAMODB_TABLE_ARN>”
}
```

© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

SECURITY PRINCIPLE #3

Practice defense in depth

© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Common vectors of attack



App vulnerabilities

- SQL injection
- Cross-site scripting (XSS)
- OWASP Top 10
- Common vulnerabilities and exposures (CVE)



Dependencies

- Libraries
- Distributions
- Base images



Host/network

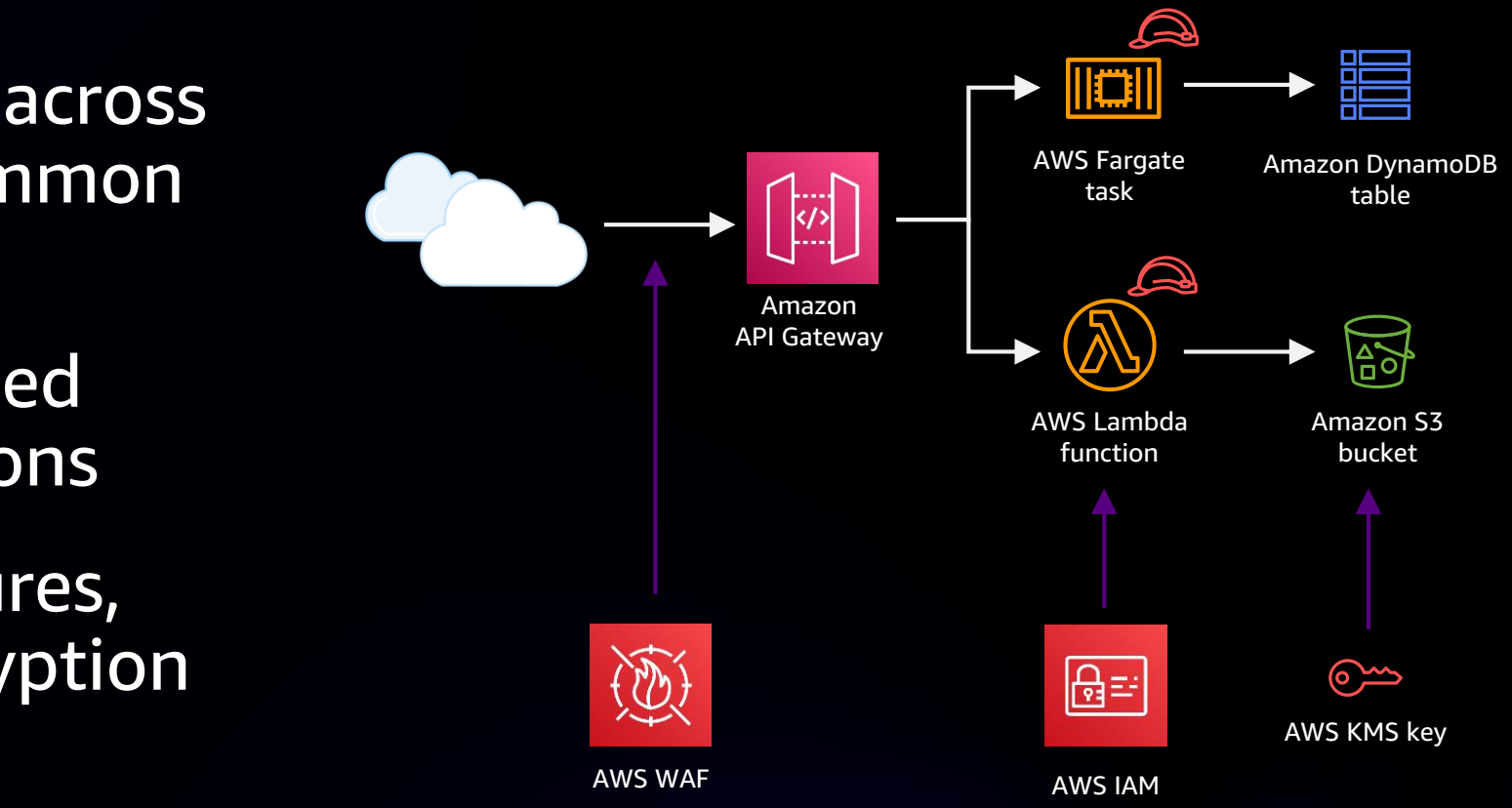
- Patching
- Network segmentation

aws

© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

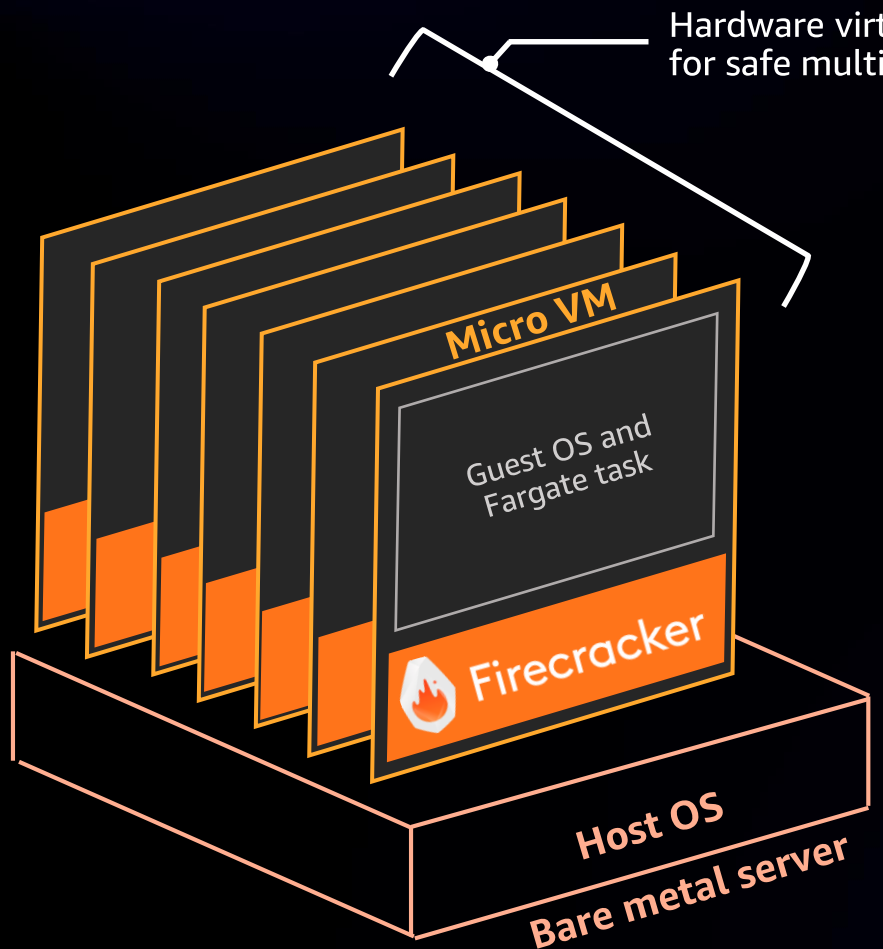
How to practice defense in depth?

- Implement **multiple, redundant measures** across system to address common attack vectors
- Leverage AWS managed services and integrations
- Consider service features, e.g., backup and encryption



© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Brief aside on Firecracker



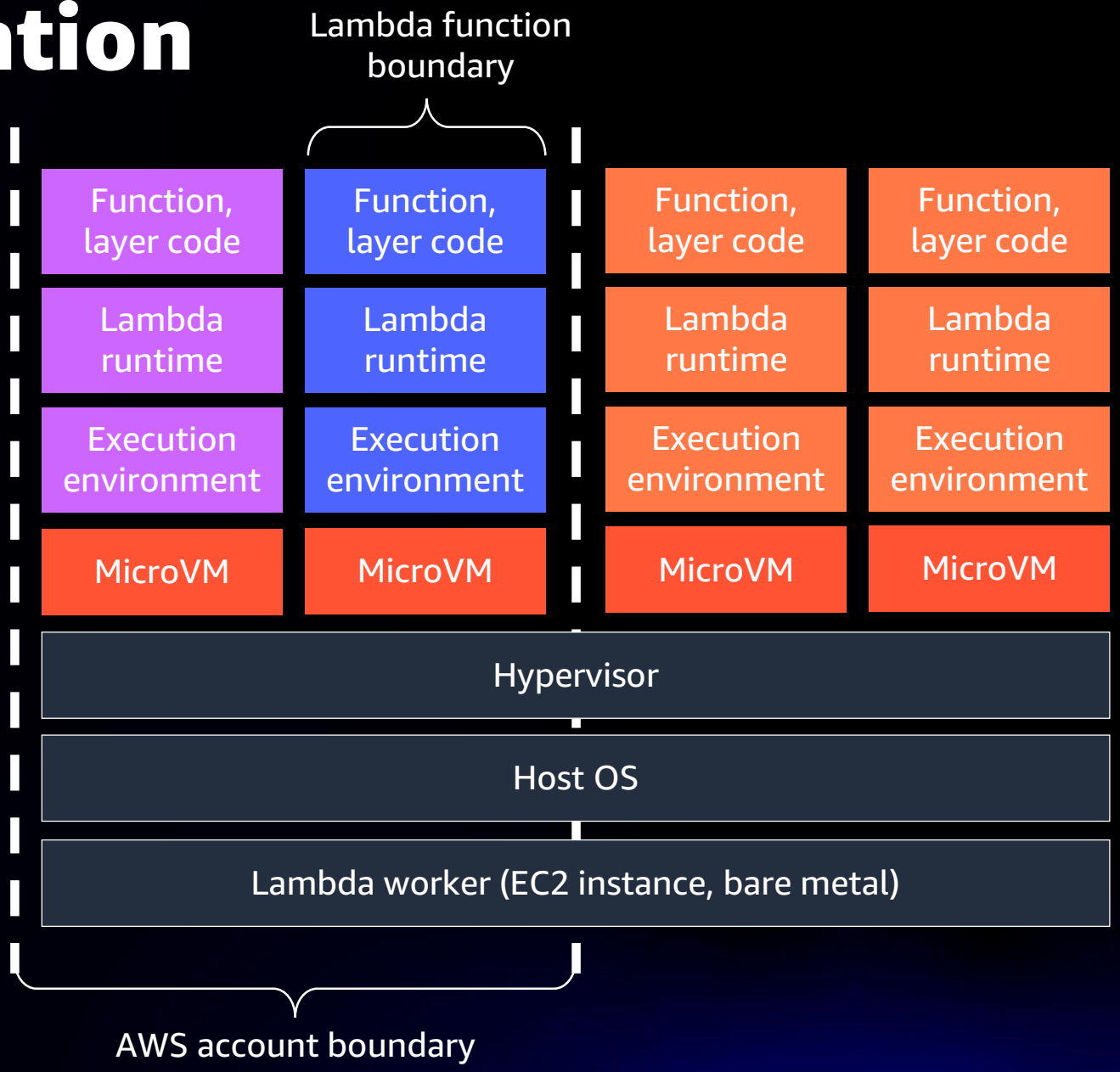
Firecracker is built on KVM, the same hypervisor that EC2 Nitro instances are built on.

Hardware virtualization ensures that tasks from different customers can run safely on the same physical machine.

© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

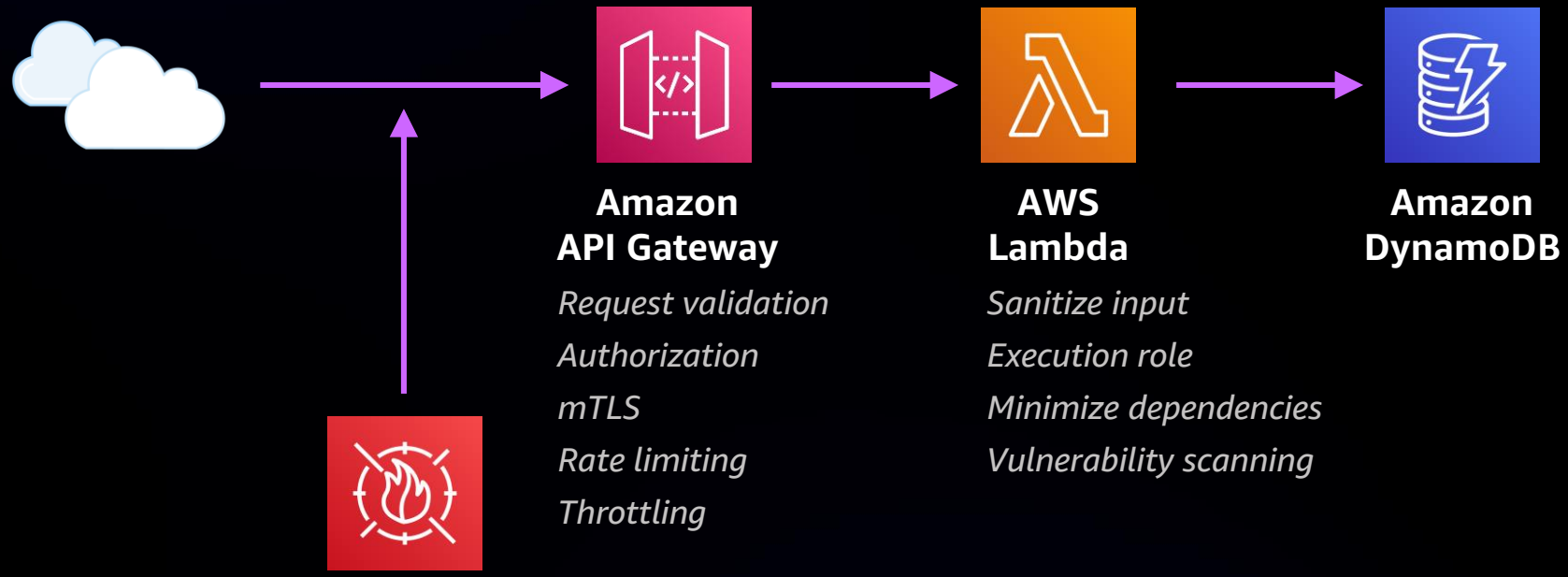
Lambda function isolation

- Each function runs in a **dedicated** execution environment
  - Each execution environment handles **one concurrent invocation**
- Execution environment **may be reused** between invocations
  - Use caution when storing sensitive data in memory or `/tmp`
- AWS maintains runtime and execution environment
  - Patching, etc.
  - Does **not** apply to container packaging or customer runtimes



© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Securing a serverless web service



Amazon API Gateway

Request validation
Authorization
mTLS
Rate limiting
Throttling

AWS Lambda

Sanitize input
Execution role
Minimize dependencies
Vulnerability scanning

Amazon DynamoDB

AWS WAF

XSS rules
SQLi rules
OWASP Top 10

© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

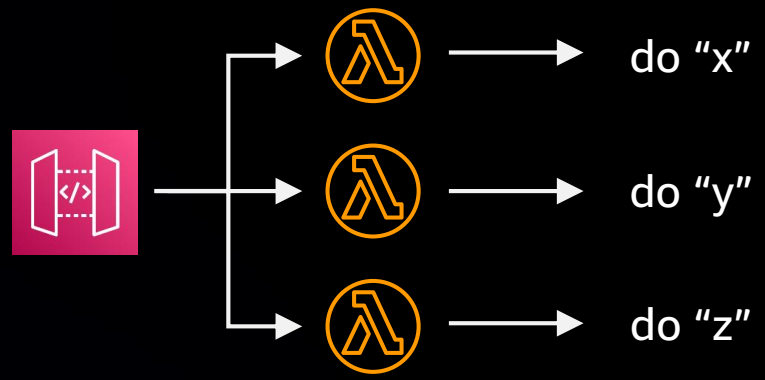
SECURITY PRINCIPLE #4

Secure your software supply chain

© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Secure your software supply chain

- Keep it **simple**
  - Prefer single responsibility
  - Easier to debug; cleaner IAM privileges
- Again, **never hardcode secrets** in code
- Leverage **code and vulnerability scanning**
  - Don’t forget dependencies



© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Managing dependencies is key

- **Understand** you dependencies: https://deps.dev/
- **Minimize** dependencies
- Keep dependencies **up-to-date** to reduce risk and effort

- Leverage dependency check tools, such as
  - OWASP
  - Protego
  - Snyk
  - Twistlock
  - Puresec



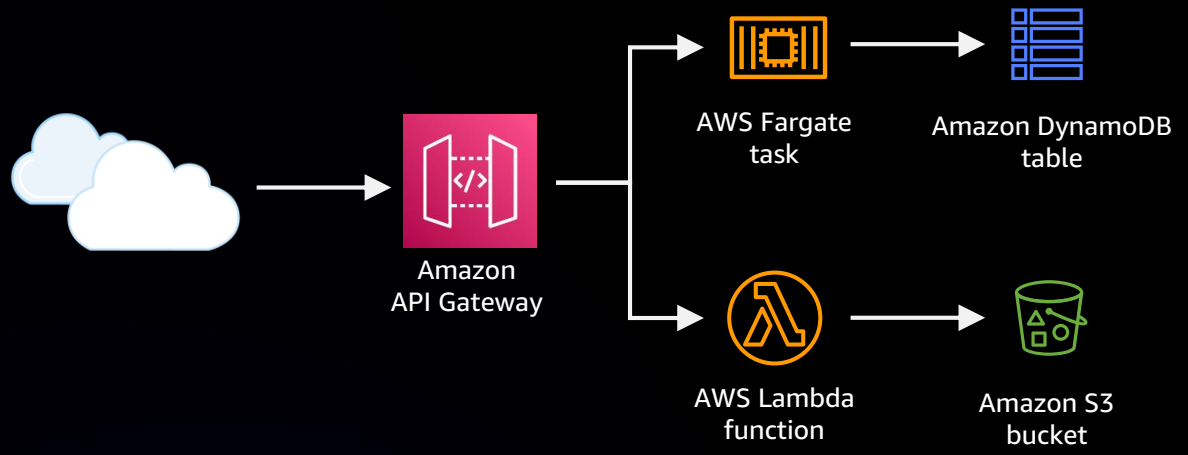
© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Common security topics

© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

How do I secure access to my API? Are API keys good enough?

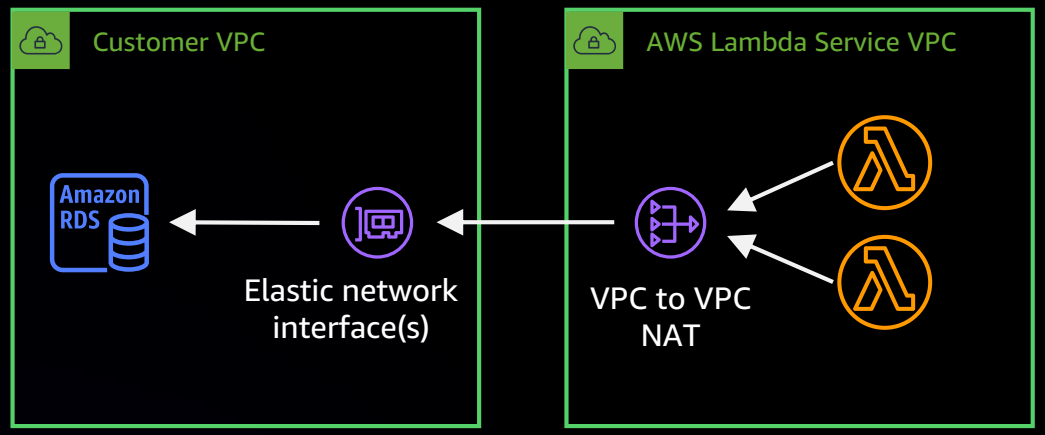
- Options for authorization:
  - IAM
  - Cognito user pool/JWT
  - Lambda authorizer
- Can be used with:
  - AWS WAF
  - Resource policies
  - Mutual TLS (mTLS)



© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Should my Lambda function be VPC-enabled?

- Lambda functions **always run in VPCs owned by the Lambda service team**
  - When VPC is enabled, configured with **access to your VPC** via an ENI
- Lambda functions are **always invoked** via `Invoke` action
  - Access controlled by AWS IAM
- Answer: Only if your function **needs access to resources in the VPC**



© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

How should I handle secrets?

- Function requires static, sensitive data
  - For example, API key or password
- While convenient, do **not** use Lambda environment variables
  - Accessible to anyone with access to the function
- Answer: Use **purpose-built services**, such as AWS Secrets Manager secured with IAM permissions



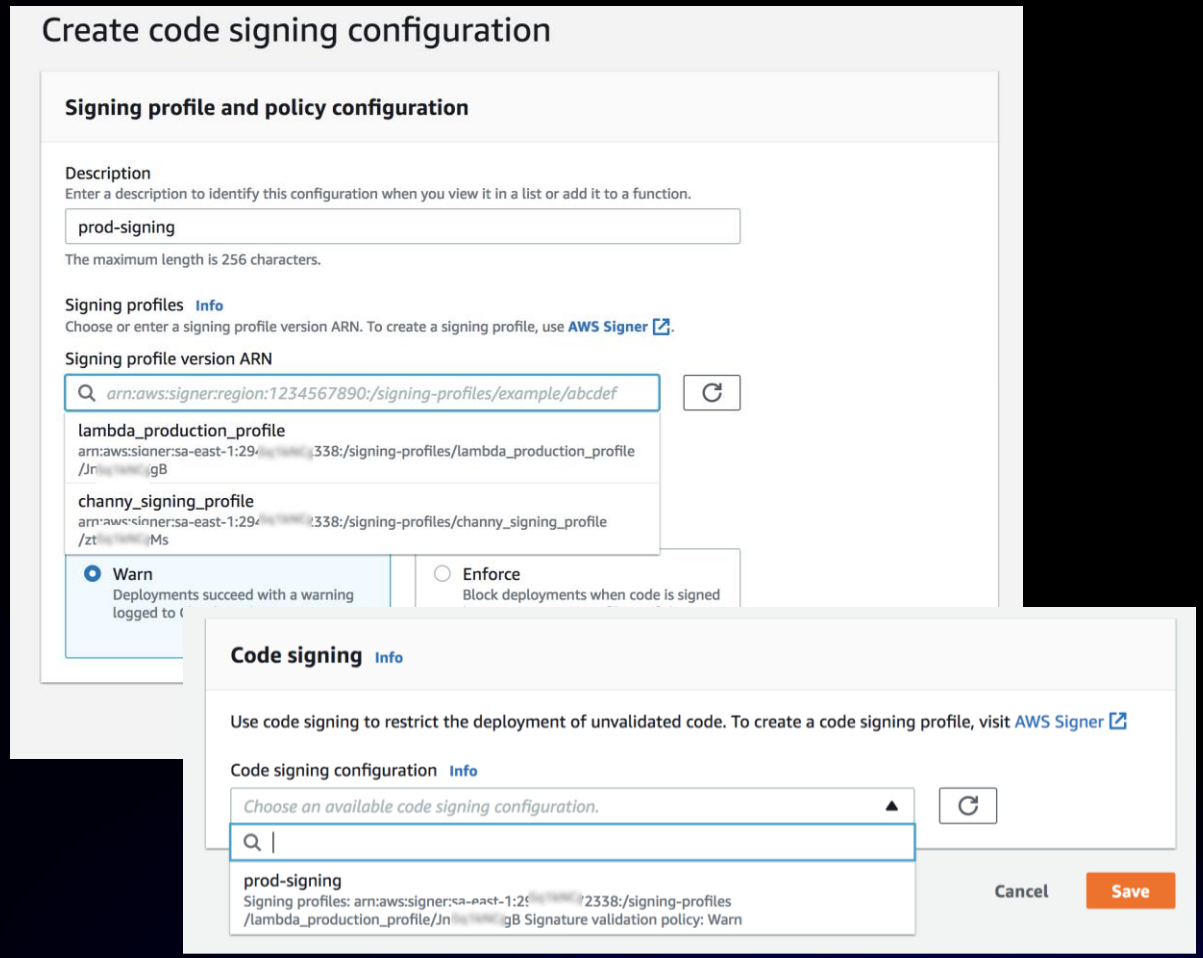
AWS Secrets Manager

aws

© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

How do I know my code has not been changed?

- Use IAM permissions to **limit create/update actions**
- Monitor with AWS Config, AWS CloudTrail
- Leverage **code signing** to cryptographically sign code
  - Allow deployment of functions only from trusted publishers



© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Thank you!

Josh Kahn

@joshuaakahn

© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.