aws professional services

DevSecOps – Deployment Security

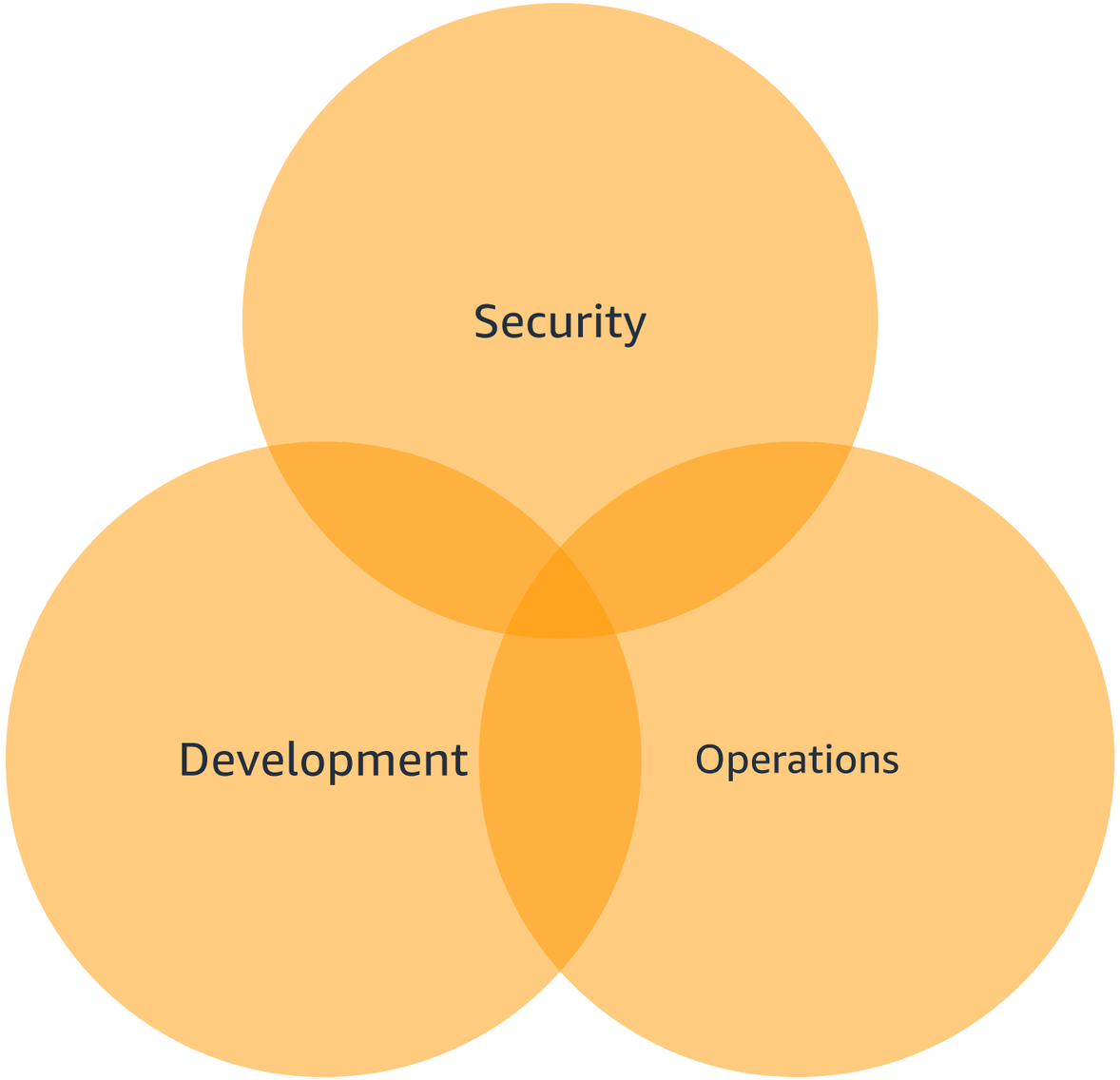
Agenda

- Security of the pipeline
- Security in the pipeline
- Static Analysis
- ECR & EC2 AMI’s
- AWS services

What is DevSecOps?

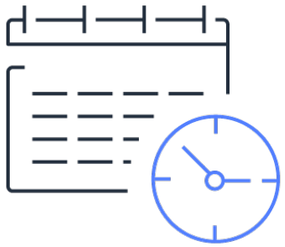
DevSecOps is the combination of cultural philosophies, practices, and tools that exploits the advances made in IT automation to achieve a state of production immutability, frequent delivery of business value, and automated enforcement of security policy

DevSecOps is achieved by integrating and automating the enforcement of preventive, detective, and responsive security controls into the pipeline



aws professional services

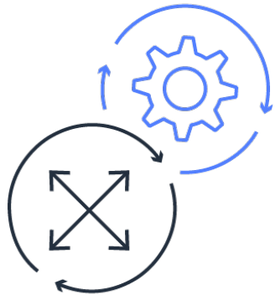
What is DevSecOps?



Test early

Prioritize preventative

Detective WITH responsive controls



Automate, automate, automate

Security of the pipeline

Security Governance & Strategy

aws professional services

AWS Cloud Adoption Framework

1 Business
- Align business and IT needs
- Map IT investments to business results

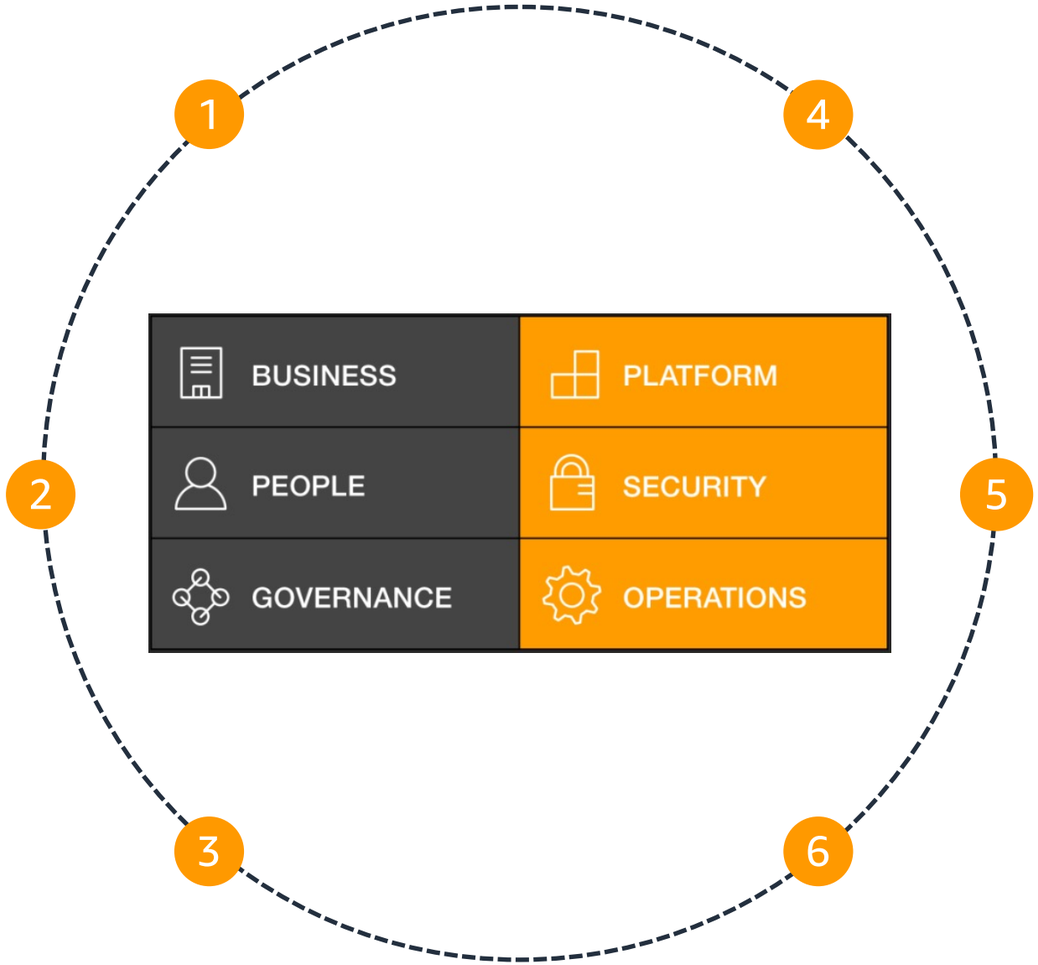
2 People
- Prioritize cloud-based competencies
- Drive organizational readiness

3 Governance
- Manage cloud investments
- Measure business outcomes

4 Platform
- Provision cloud applications and infrastructure
- Improve cloud services and solutions

5 Security
- Align security and compliance with current requirements
- Manage access and authorization

6 Operations
- Monitor and maintain system health and reliability
- Observe cloud best practices



Align with Cloud Adoption Framework

Identity and
access management

Detective controls

Infrastructure controls

Data protection

Incident response

aws

Some IAM risks for pipelines

- Anyone can run build jobs
- Consistent user management across build servers
- Pipeline role is too permissive
- Slave node adverse affects on masters

Enforcing least privilege between pipelines

- Pipeline can perform a specific job
- E.g., Jenkins/Spinnaker/AWS CodePipeline is a pipeline factory
- Pipeline role is too permissive
- Pipelines can be limited to blast-radius-based functions
    - Pipeline factory
    - AMI factory
    - Artifact factory

Align with Cloud Adoption Framework

Identity and
access management

Detective controls

Infrastructure controls

Data protection

Incident response

aws

Detective controls for pipelines

- Who logged in?
- What code was committed and by whom?
- What jobs did they run?
- Did the jobs succeed/fail?
- Was static/dynamic analysis enforced?
- What were the results of the static/dynamic analysis?

Align with Cloud Adoption Framework

Identity and
access management

Detective controls

Infrastructure controls

Data protection

Incident response

Infrastructure security risks to pipelines

- Who has access to underlying infrastructure resources?
- How are pipelines patched and updated?
- How is least privilege between pipelines enforced?
- Are my pipelines deploying into approved AWS accounts?
- Does the pipeline align with organizational responsibility?

Align with Cloud Adoption Framework

Identity and
access management

Detective controls

Infrastructure controls

Data protection

Incident response

Data protection risks for pipelines

- Who can change/commit code?
- How is production data prevented from being introduced into non-prod environments?
- How is artifact integrity maintained?

Top data protection best practices

- Control access and permissions to the code repository
- Trigger builds automatically (time based or event based)
- Use tokenization or dummy data in non-production environments
- Categorize data and enforce restrictions through pipeline
    - For example, pipeline is configured to build the Dev environment but is not allowed to pull production data from repo

Data protection for pipelines wrap-up

- Control access and permissions to source repository: artifacts are critical data for your pipeline
- Build pipelines that are environment aware (e.g., prod vs. non-prod)
- Build artifact handlers to validate integrity across pipelines and environments

Align with Cloud Adoption Framework

Identity and
access management

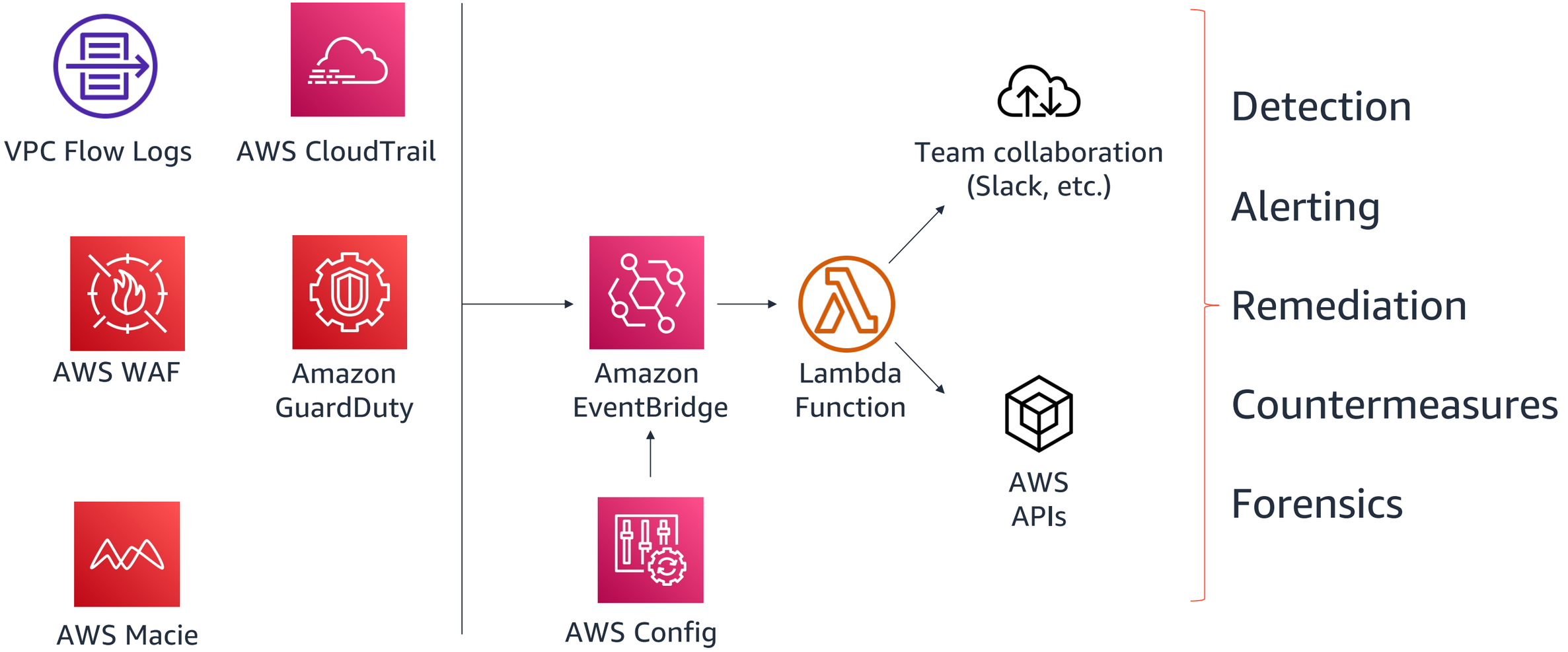
Detective controls

Infrastructure controls

Data protection

Incident response

Pattern for automated remediation



Security in the pipeline

Security Governance & Strategy

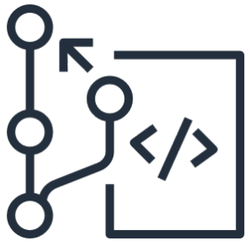
aws professional services

Pipeline as a workload

- Securing the application starts with securing the pipeline
- The CI/CD pipeline is a workload
- Its purpose is to integrate and deliver other workloads
- It has users, supporting infrastructure, application, data components, etc.
- Those components are typically managed as code

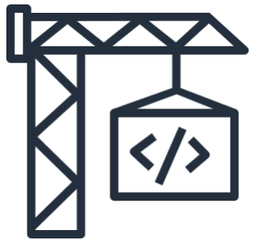
Security IN the pipeline

Code analysis



Code

Dependencies



Build

Vulnerabilities

Test

Hash verification

Deploy

Automated

Monitor

Pipeline as a workload

- **Static analysis**
    - **Infrastructure as code**
    - Security as code
- Dynamic analysis
    - Unit tests
    - Integration tests
    - System tests

CloudFormation

- Predictable, repeatable, version-controlled infrastructure
- Perform controlled upgrades, rollbacks across accounts, regions
- Define templates to create logically grouped resources
- Custom resources with Lambda

Cloud Development Kit (CDK)

- Synthesize CloudFormation using code
- Supports Javascripts, Typescript, Python, Java, C#
- Best practices:
    - Model through constructs, not stacks
    - Use secrets manager
    - Let CDK handle roles and security groups
    - Take advantage of the ecosystem

https://docs.aws.amazon.com/cdk/latest/guide/best-practices.html

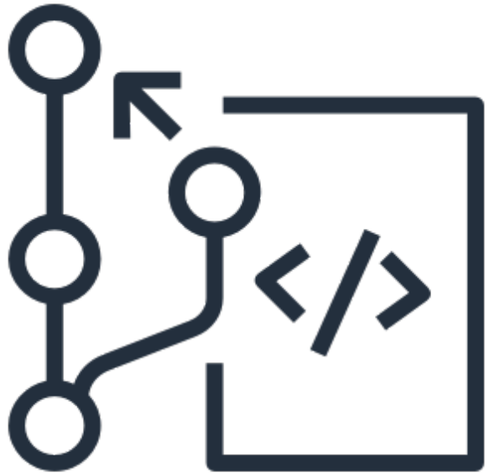
CDK Aspects

```
class BucketVersioningChecker implements IAspect {
  public visit(node: IConstruct): void {
    // See that we're dealing with a CfnBucket
    if (node instanceof s3.CfnBucket) {

      // Check for versioning property, exclude the case where the property
      // can be a token (IResolvable).
      if (!node.versioningConfiguration
        || (!Tokenization.isResolvable(node.versioningConfiguration)
            && node.versioningConfiguration.status !== 'Enabled')) {
        Annotations.of(node).addError('Bucket versioning is not enabled');
      }
    }
  }
}

// Later, apply to the stack
Aspects.of(stack).add(new BucketVersioningChecker());
```

https://docs.aws.amazon.com/cdk/latest/guide/aspects.html

Static Code Analysis

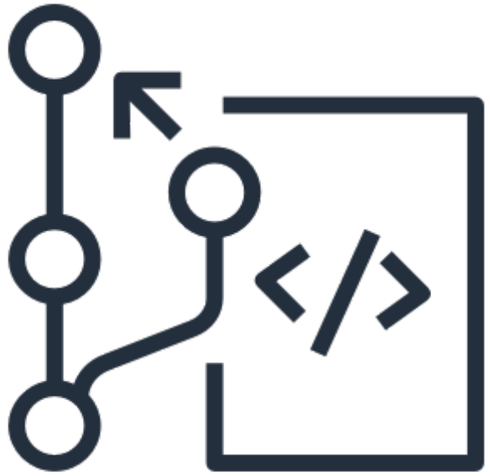
- **git secrets**
- cfn-lint, cfn-nag
- **CloudFormation Guard**
- **CodeGuru (Reviewer)**



Git Secrets

- Pre-commit hook(!)
- Scan for AWS secrets in commits, commit messages, and history.

```
$ cd /path/to/my/repo
$ git secrets —install
$ git secrets —register-aws

# Global hook
$ git secrets –register-aws --global

# Scan repo including all revisions
$ git secrets --scan-history
```



CloudFormation Guard

- Open-source Policy-as-code evaluation tool
- Scans CloudFormation templates (JSON/YAML)
- Detect non-compliance prior to deployment
- Rulegen – generate a set of rules from a compliant template

https://github.com/aws-cloudformation/cloudformation-guard
https://aws.amazon.com/blogs/mt/introducing-aws-cloudformation-guard-2-0/

CloudFormation Guard – writing rules

```
##
## S3
##
let s3_buckets = Resources.*[ Type == 'AWS::S3::Bucket' ]

# S3 Bucket encryption
rule s3_bucket_encryption_check when %s3_buckets !empty {
  %s3_buckets {
    let metadata = Metadata."cfn-guard".SuppressedRules[
      some this == "s3_bucket_encryption_check"
    ]

    when %metadata empty {
      Properties {
        # encryption must be configured
        BucketEncryption.ServerSideEncryptionConfiguration[*] {
          ServerSideEncryptionByDefault.SSEAlgorithm !empty
        }
      }
    }
  }
}
```

https://github.com/aws-cloudformation/cloudformation-guard/tree/main/guard-examples

CloudFormation Guard – suppressing rules

CloudFormation

```
Bucket83908E77:
   Type: AWS::S3::Bucket
   UpdateReplacePolicy: Retain
   DeletionPolicy: Retain
   Metadata:
     cfn-guard:
       SuppressedRules:
         - s3_bucket_encryption_check
```

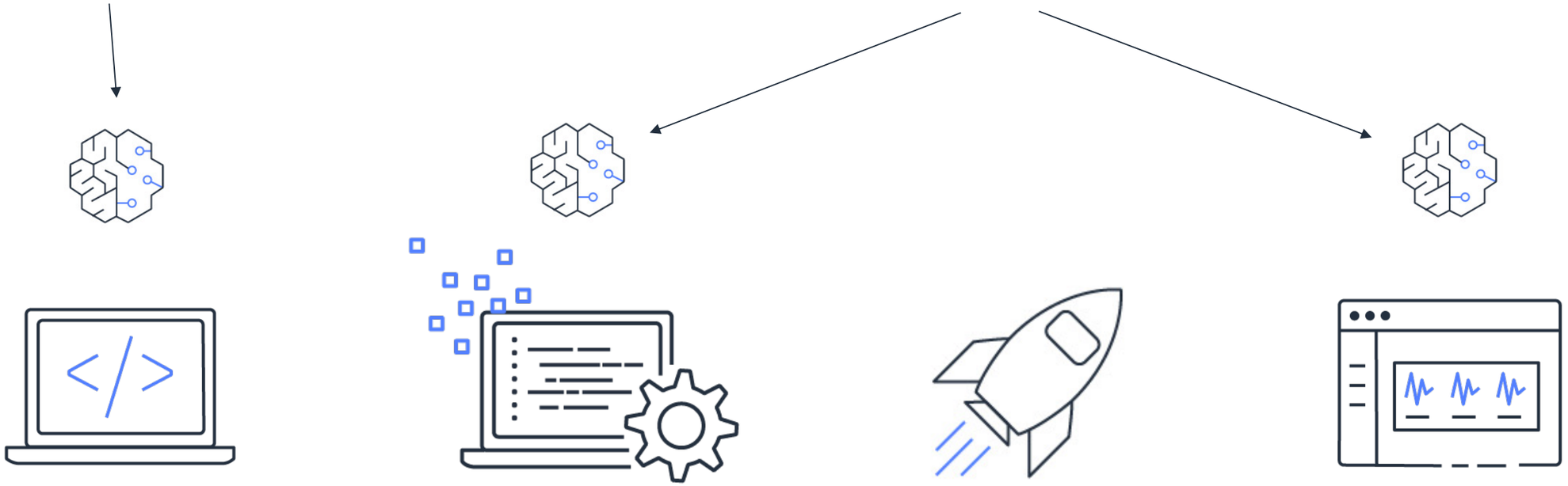
CDK (TS)

```
const bkt = new s3.Bucket(this, 'Bucket');
const cfnBkt = bucket.node.defaultChild as s3.CfnBucket;
cfnBucket.cfnOptions.metadata = {
  'cfn-guard': {
    'SuppressedRules': ['s3_bucket_encryption_check'],
  },
};
```

CodeGuru

CodeGuru Reviewer

CodeGuru Profiler



Coding:

Built-in code reviews with actionable recommendations

Build & Test:

Detect and optimize the expensive lines of code pre-production

Deploy

Measure:

Easily identify performance and cost improvements in production environment

aws professional services

CodeGuru Reviewer

- **Reviewer**:
    - Detect deviations from best practices
- **Reviewer Security Detector**:
    - Identify security best practices
- Github, codecommit, bitbucket cloud,
- Scans code, provides a report
- Java (8..11) and Python (3) support
- Free 90 day trial

Example: Detecting sensitive information leaks

Code

```
try {
    updateJobStatus(context.getAwsRequestId(),
                    request.getAwsAccountId(),
                    request.getPredictorName(),
                    request.getInternalStatus());
} catch (validationException e) {
    log.error(NON_RETRIABLE_LIST_ERROR_MESSAGE, e);
    throw e;
} catch (internalServerException e) {
    log.warn(RETRIABLE_LIST_ERROR_MESSAGE, e);
    retries++;
    continue;
}
```

Fix

```
try {
    updateJobStatus(context.getAwsRequestId(),
                    request.getAwsAccountId(),
                    request.getPredictorName(),
                    request.getInternalStatus());
} catch (validationException e) {
    log.error(NON_RETRIABLE_LIST_ERROR_MESSAGE, redact(e));
    throw readact(e);
} catch (internalServerException e) {
    log.warn(RETRIABLE_LIST_ERROR_MESSAGE, redact(e));
    retries++;
    continue;
}
```

Recommendation

This code contains potential unintended disclosure of information in the error handling for the following call: 'getAwsAccountId()'. You are handling the error with catch classes. There are methods available that could be added to handle sensitive data, like masking and redaction.

Dynamic Code Analysis

- **Reviewer**:
    - Detect deviations from best practices
- **Reviewer Security Detector**:
    - Identify security best practices
- Github, codecommit, bitbucket cloud,
- Scans code, provides a report
- Java (8..11) and Python (3) support
- Free 90 day trial

Amazon Elastic Container Registry (ECR)

- Managed registry for container images (docker/OCI)
- Data encrypted in transit and at rest (kms)
- Managed lifecycle (tagging, versioning)
- Vulnerability scanning on-push or ad-hoc
- ECS, EKS, on-premises, …
- Access control
- Artifact trust
- CloudWatch metrics and events

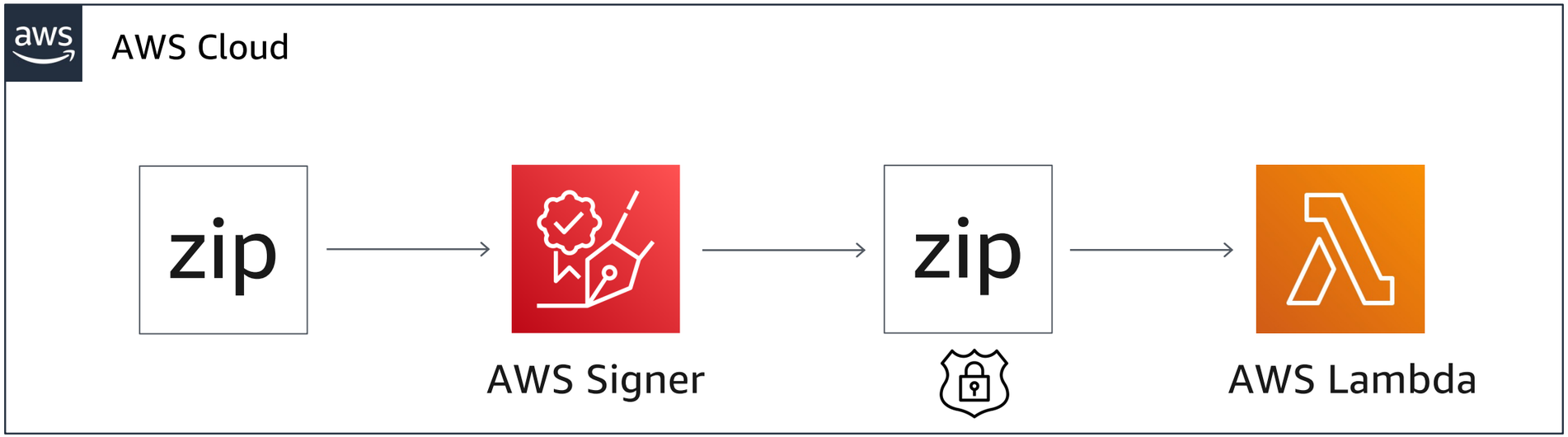
Amazon ECR: best practices

- Apply least-privilege w/ periodic reviews
- Process alerts (critical & high!)
    - Use a lambda to deny instantiation*
- Use immutable tags
- Enable scan on push
- Implemented scheduled scans for (at least) production images

https://aws.amazon.com/blogs/containers/automating-image-compliance-for-amazon-eks-using-amazon-elastic-container-registry-and-aws-security-hub/

AWS Signer

- Fully Managed Code Signing Service
- Signing for Lambda Functions and Layers
- Signing for IOT supported by Amazon FreeTOS

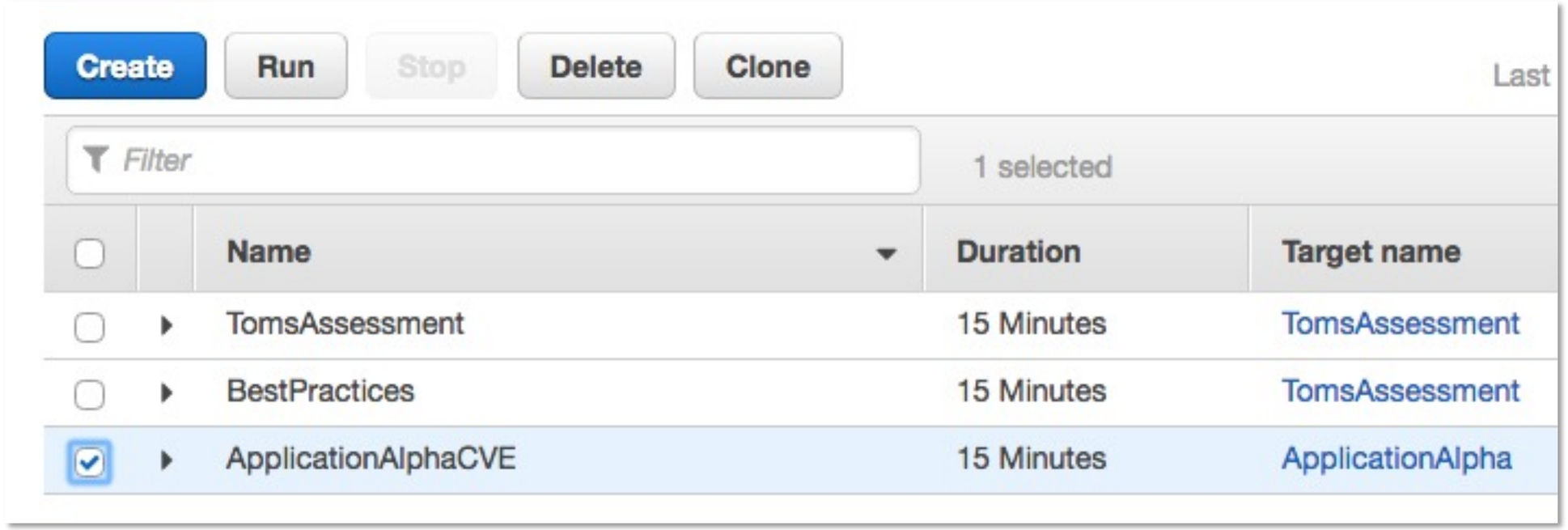


https://aws.amazon.com/blogs/aws/new-code-signing-a-trust-and-integrity-control-for-aws-lambda/

Amazon Inspector

- Vulnerability Assessment Service
  - Built from the ground up to support DevSecOps
  - Automatable via APIs
  - Integrates with CI/CD tools
  - On-Demand Pricing model
  - Generates Findings
  - Multiple Static & Dynamic Rules Packages
  - Generate Finding based Action using EventBridge integration

Amazon Inspector - Assessments and results



Amazon Inspector - Finding details

Finding	Instance i-57dd8990 is using insecure protocol(s) smtp (port 25, Simple Mail Transfer).
Severity	Informational
Description	This rule helps determine whether your EC2 instances allow support for insecure and unencrypted ports/services such as FTP, Telnet, HTTP, IMAP, POP version 3, SMTP, SNMP versions 1 and 2, rsh, and rlogin.
Recommendation	We recommend you disable insecure protocols in your application and replace them with secure alternatives as listed below: • Disable telnet, rsh, and rlogin and replace them with SSH. Where this is not possible, you should ensure that the insecure service is protected by appropriate network access controls such as VPC network ACLs and EC2 security groups. • Replace FTP with SCP or SFTP where possible. Where this is not possible, you should ensure that the FTP server is protected by appropriate network access controls such as VPC network ACLs and EC2 security groups. • Replace HTTP with HTTPS where possible. For more information specific to the web server in question see http://nginx.org/en/docs/http/configuring*https*servers.html and https://httpd.apache.org/docs/2.4/ssl/ssl_howto.html. • Disable IMAP, POP3, SMTP services if not required. If required, it's recommended these email protocols should be used with encrypted protocols such as TLS. • Disable SNMP service if not required. If required, replace SNMP v1, v2 with more secure SNMP v3 which uses encrypted communication.

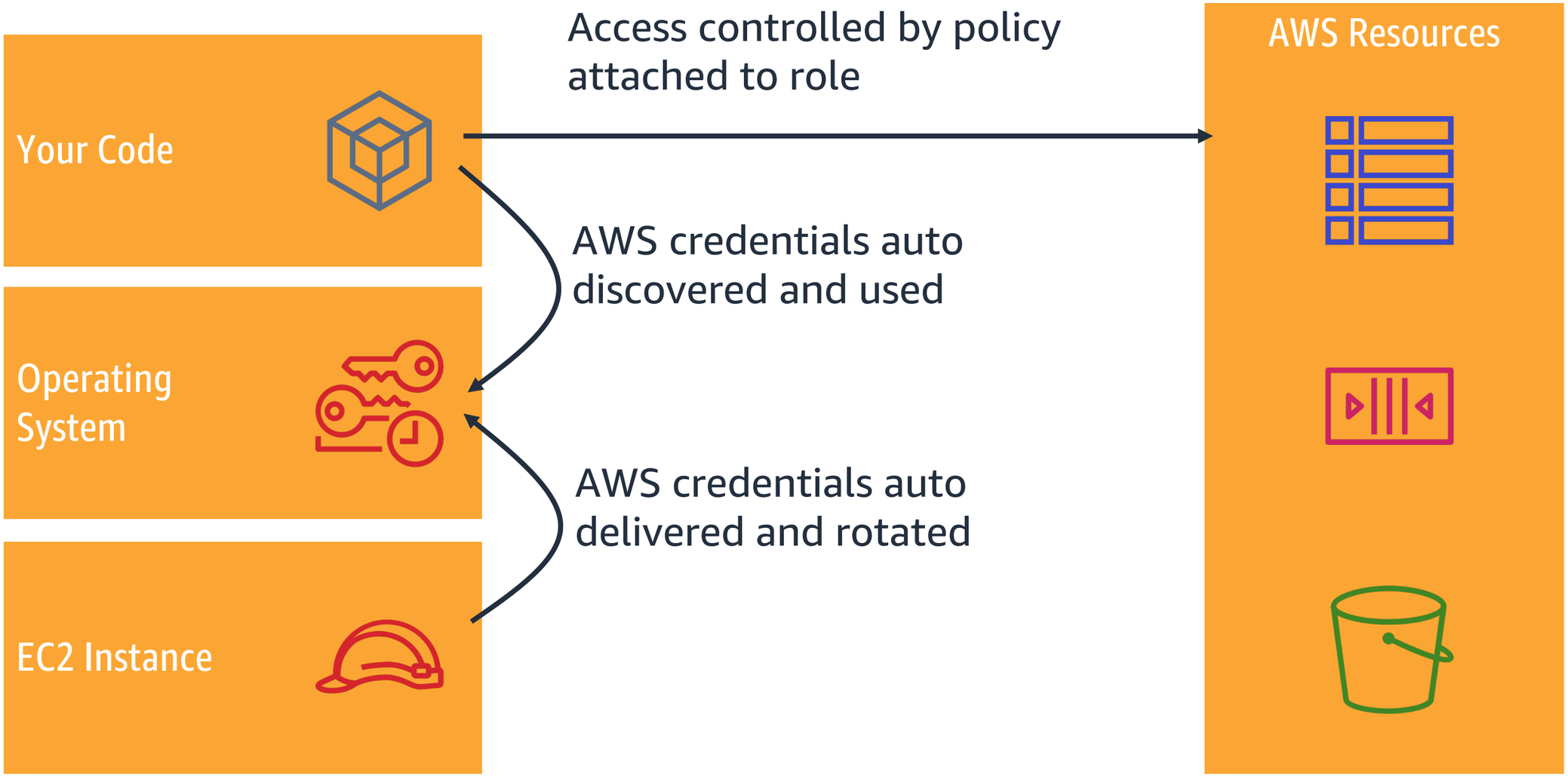
Amazon Inspector – Rules Packages

- Common Vulnerabilities & Exposures
- Center for Internet Security (CIS) Benchmarks
- Security Best Practices
- Runtime Behavior Analysis
- Network Reachability

AWS Secrets Manager

- Safe rotation of secrets
- Built-in integrations, extensible with Lambda
- On-demand or automatic rotation with versioning
- Fine-grained access policies
- Encrypted storage
- Logging and auditability

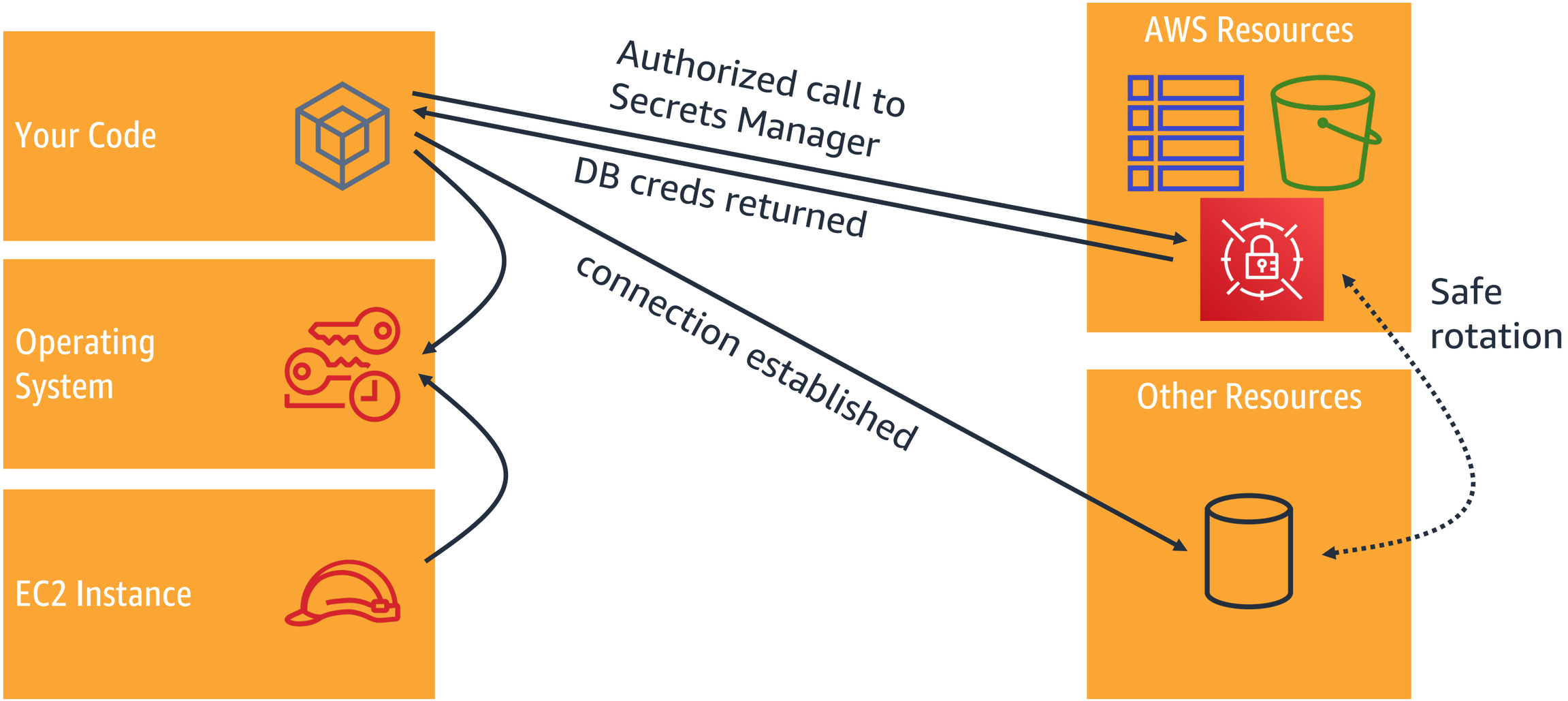
AWS Secrets Manager: IAM Roles



Also works with AWS Lambda & Amazon ECS

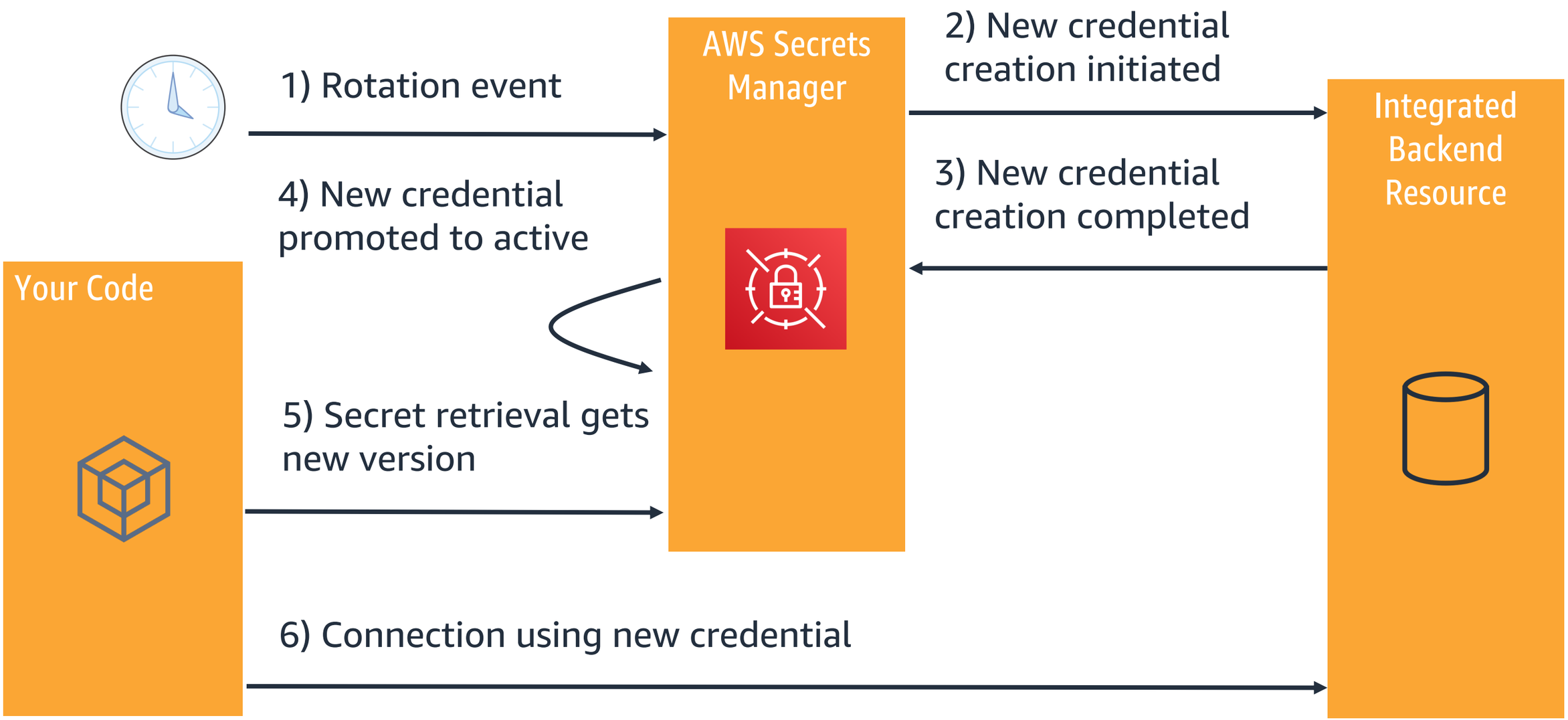
aws professional services

AWS Secrets Manager: Retrieve Secret



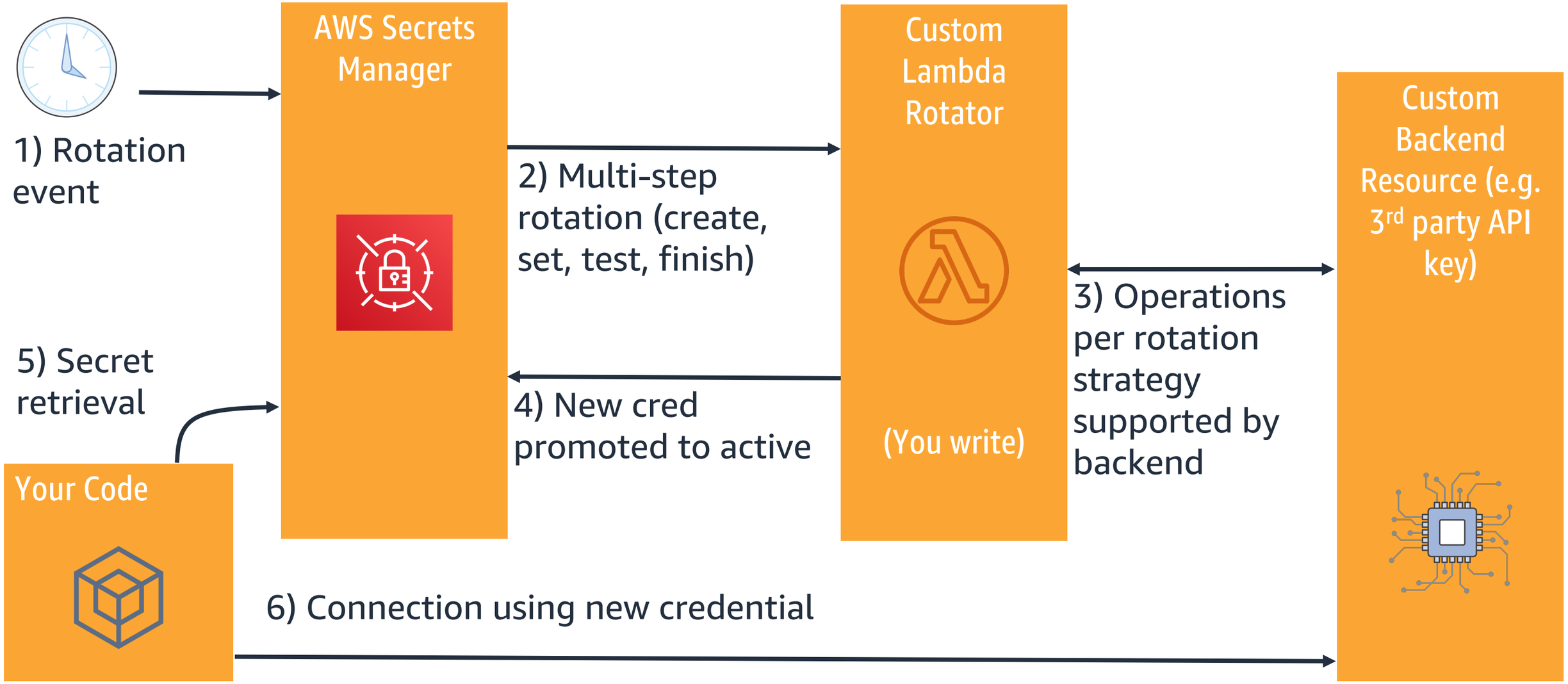
aws professional services

AWS Secrets Manager: Rotate Secret (Integrated)



aws professional services

Rotate Secret (Custom)



aws professional services

Encryption

All secrets protected at-rest and in-transit

At-rest

Secrets encrypted at rest using AWS Key Management Service (KMS).

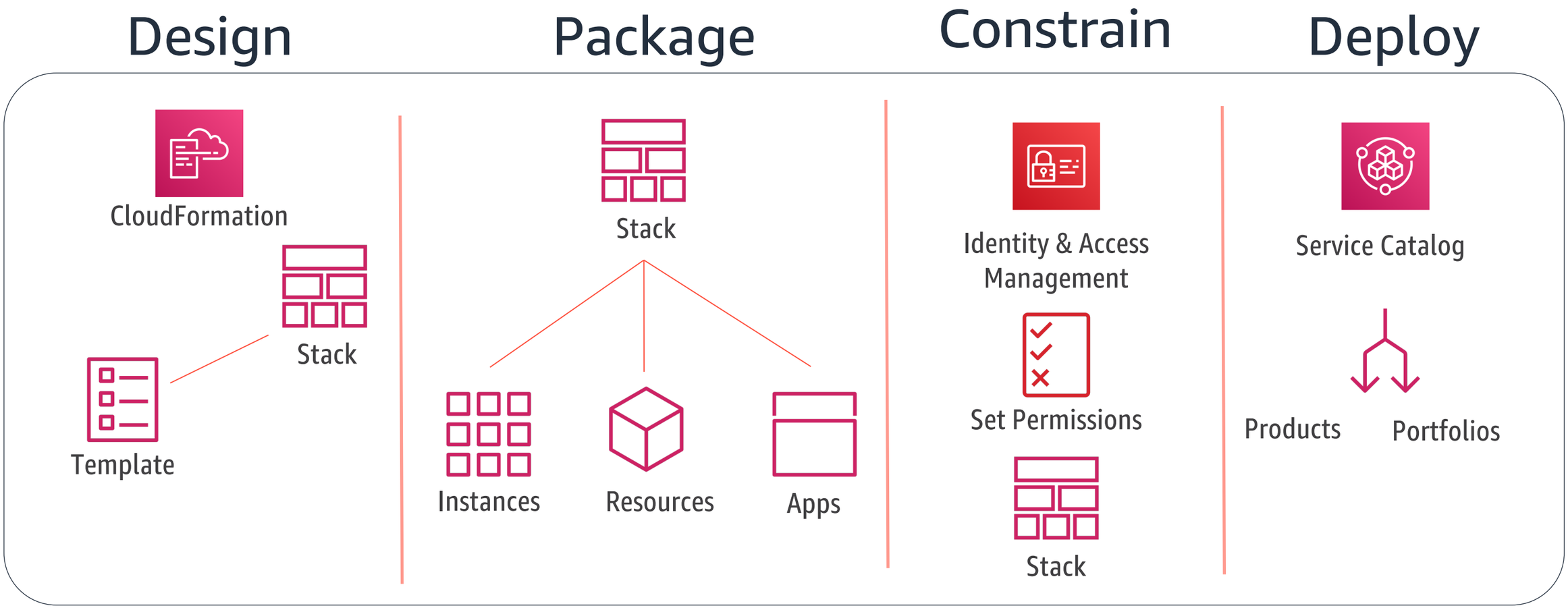
Choose your desired Customer Master Key (CMK) or AWS managed default encryption key.

In-transit

Secrets encrypted in transit using Transport Layer Security (TLS).

All API calls authenticated by SigV4 verification.

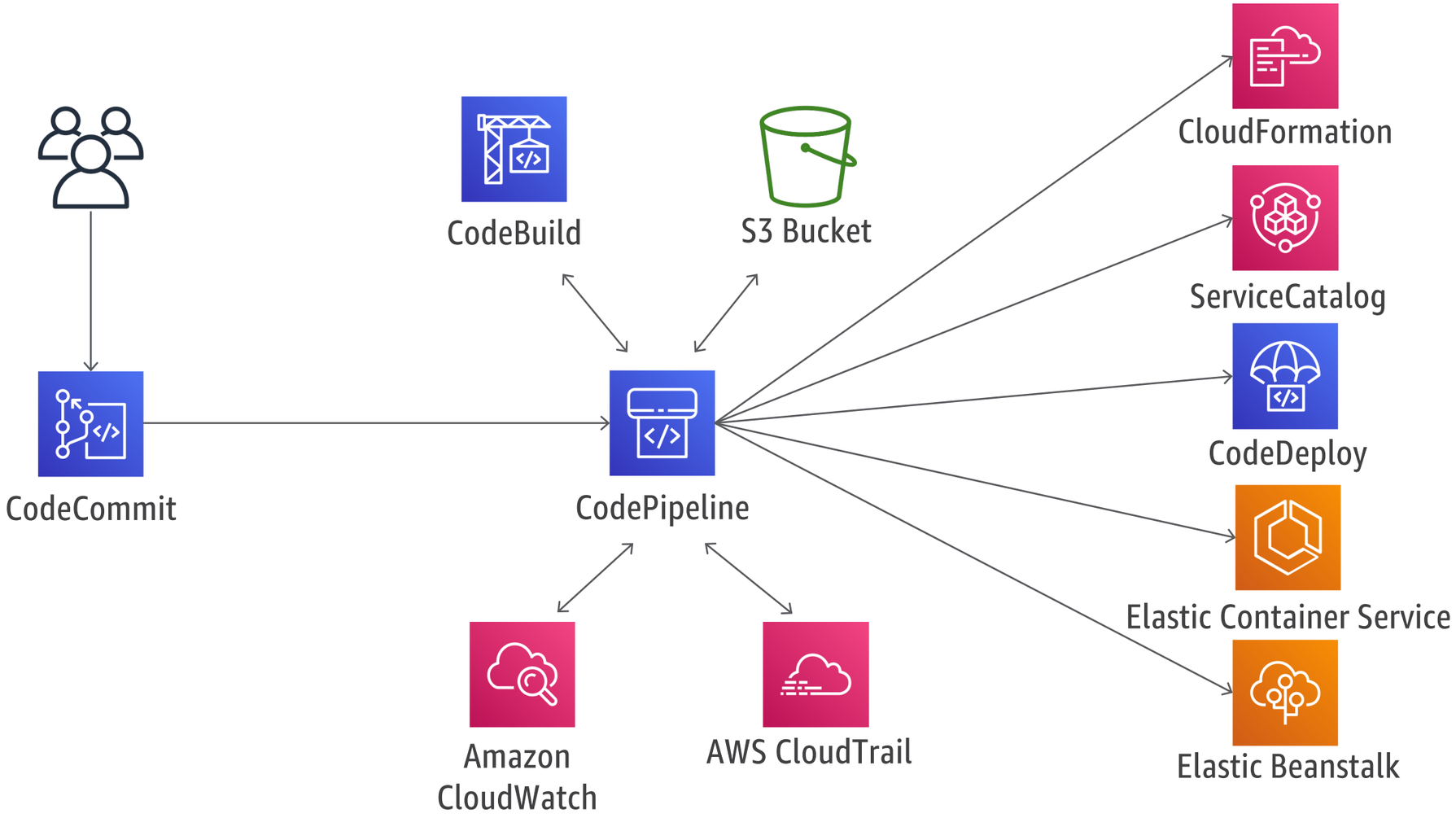
SbD – Automated Deployments



Automate deployments, provisioning, and configurations of the AWS customer environments

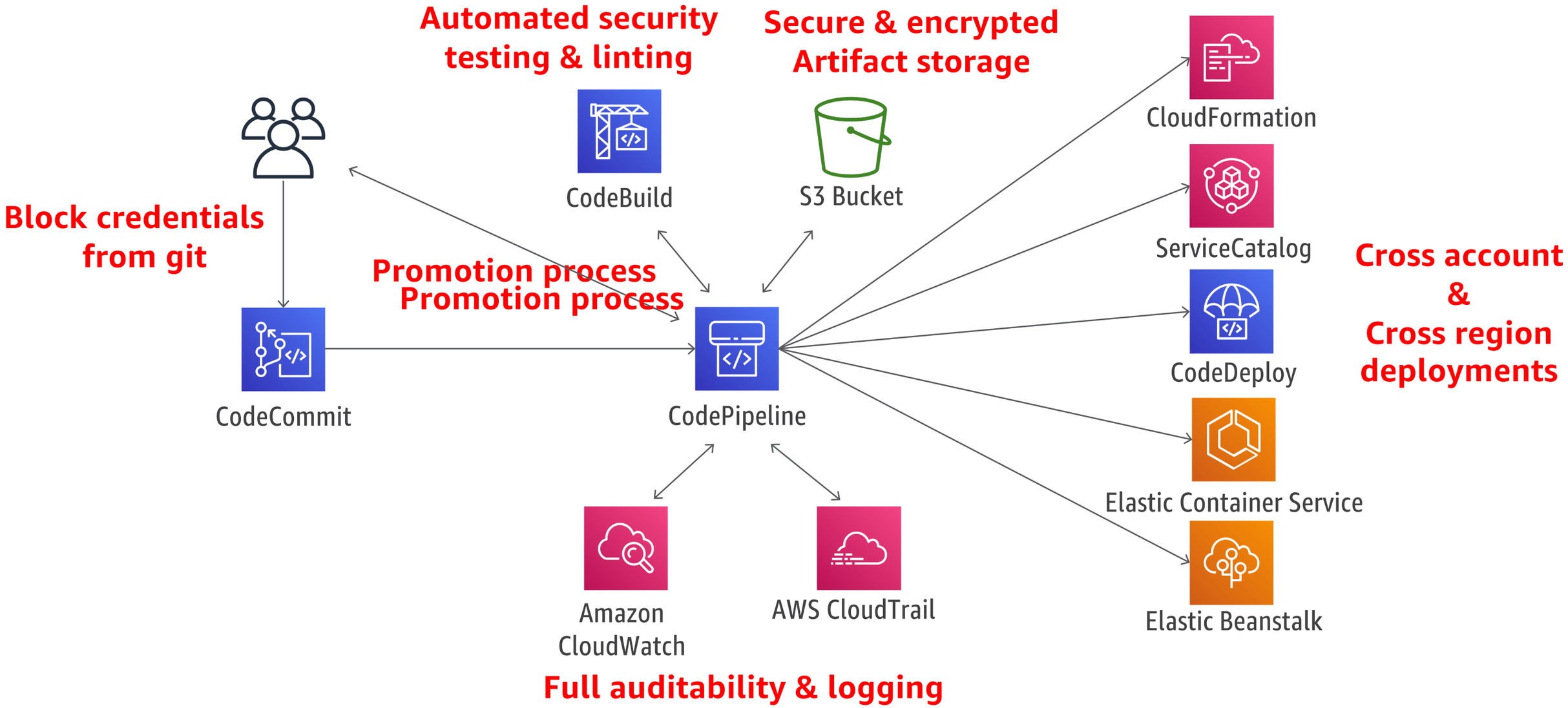
aws professional services

SbD – Continuous Deployment



aws professional services

SbD – Continuous Deployment



aws professional services

Resources

Adding Security into DevOps workshop

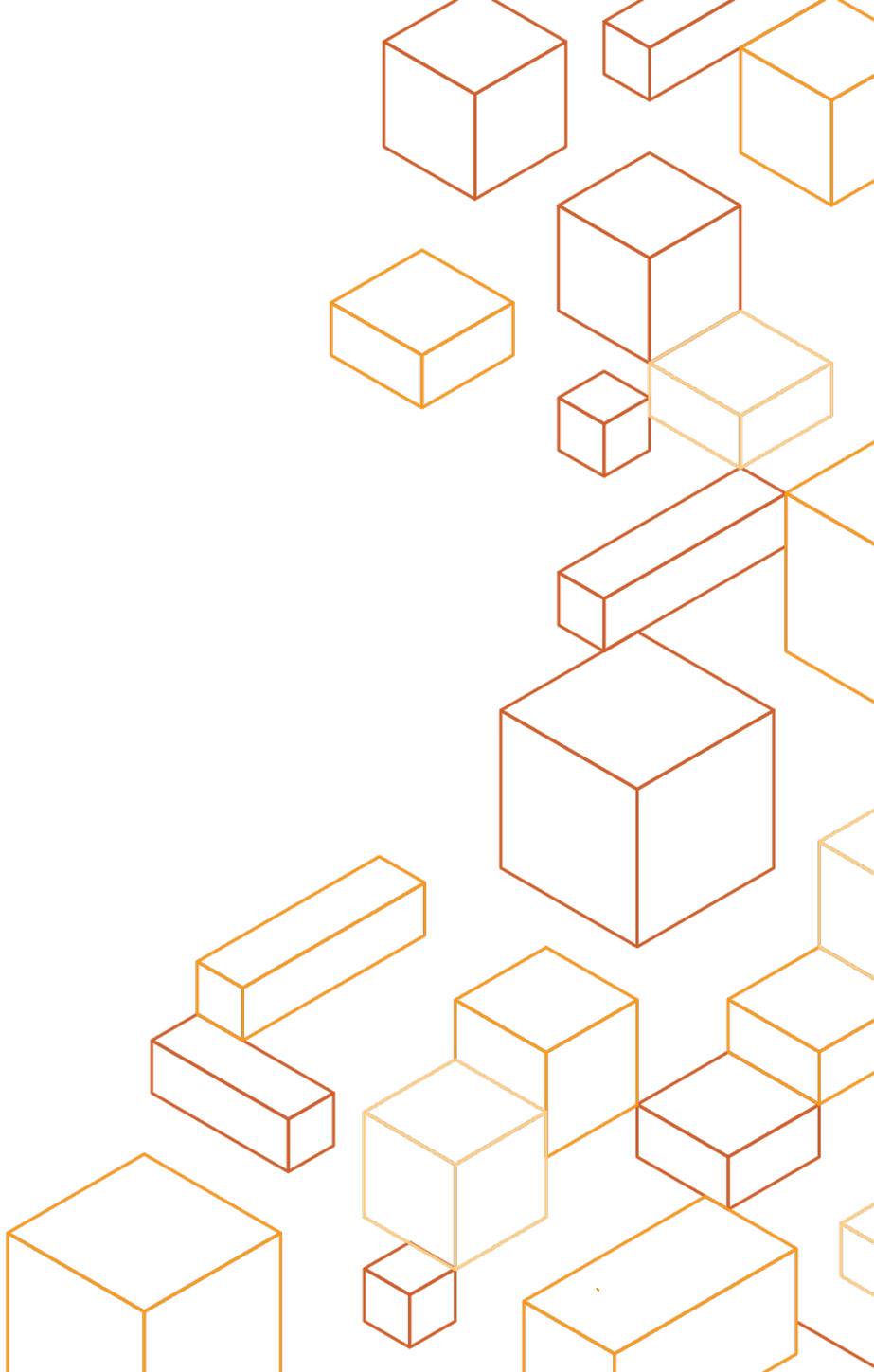
- https://devops.awssecworkshops.com/

Building end-to-end AWS DevSecOps …

- https://aws.amazon.com/blogs/devops/building-end-to-end-aws-devsecops-ci-cd-pipeline-with-open-source-sca-sast-and-dast-tools/

aws professional services

Questions



Appendix A - Threat Modelling

"… a form of risk assessment that models aspects of the attack and defense sides of a particular logical entity, such as a piece of data, an application, a host, a system, or an environment."

- https://csrc.nist.gov/CSRC/media/Publications/sp/800-154/draft/documents/sp800_154_draft.pdf

Appendix A - Threat Modelling

What?

- Record our worries
- Record assumptions
- Identify potential threats
- Actions to be taken for each threat
- Validation of success

Why?

- Build a secure design
- Understand compliance requirements and risk evaluation
- Prioritise our efforts
- Identify test cases

Appendix A - Threat Modelling

How?

- Repeatable process
- Frameworks
    - OWASP Testing Framework
    - **DREAD**
    - STRIDE

Dread Example:

Category	Rating	Information Disclosure
Damage Potential	7	All data in the database
Reproducibility	7	Once identified, is repeatable
Exploitability	5	Easier than XSS
Affected Users	2	Unnoticed, unless data is damaged
Discoverability	6	Easier to identify, than to exploit
DREAD Average	5,4	

Appendix B – Security as Code – KMS Rotation

```
const ruleKmsRotation = new config.ManagedRule(this, 'RuleKmsRotation', {
  identifier: config.ManagedRuleIdentifiers.CMK_BACKING_KEY_ROTATION_ENABLED,
});

new config.CfnRemediationConfiguration(this, 'RuleKmsRotationRemediation', {
  configRuleName: ruleKmsRotation.configRuleName,
  targetId: 'AWSConfigRemediation-EnableKeyRotation',
  targetType: 'SSM_DOCUMENT',
  targetVersion: '1',
  automatic: true,
  maximumAutomaticAttempts: 5,
  retryAttemptSeconds: 60,
  parameters: {
    AutomationAssumeRole: {
      StaticValue: { Values: [autoRemediationRole.roleArn] },
    },
    KeyId: {
      ResourceValue: { Value: 'RESOURCE_ID’ },
    },
  },
});
```

Appendix B – Security as Code – EventBridge Rule

```
const bus = new events.EventBus(this, 'Bus');

new events.Rule(this, 'RuleCloudTrailChange', {
  description: 'Detect CloudTrail change’,
  eventBus: this.bus,
  eventPattern: {
    source: ['aws.ec2'],
    detailType: ['AWS API Call via CloudTrail'],
    detail: {
      eventSource: ['cloudtrail.amazonaws.com'],
      eventName: ['CreateTrail', 'UpdateTrail', 'DeleteTrail', 'StartLogging', 'StopLogging'],
    },
  },
});
```

Appendix B – Security as Code – Ebs Encryption

```
const enableEbsEncryptionByDefault = new AwsCustomResource(this, 'EbsEncryptionByDefault', {
  onCreate: {
    service: 'EC2',
    action: 'enableEbsEncryptionByDefault',
    physicalResourceId: PhysicalResourceId.of('ebsEncryption'),
  },
  onDelete: {
    service: 'EC2',
    action: 'disableEbsEncryptionByDefault',
  },
  policy: AwsCustomResourcePolicy.fromSdkCalls({ resources: AwsCustomResourcePolicy.ANY_RESOURCE }),
});
```