

AWS Essentials

13. 개발 및 운영 방안

CONTENTS

1

AWS 지속적 코드 배포 방법

2


AWS 운영 방안

3

AWS 운영 체크 리스트

학습 목표

- AWS에서 지속적 통합과 지속적 배포 고려한 구성을 이해할 수 있습니다.
- AWS의 운영 방안 정책에 대해 이해할 수 있습니다.
- AWS 상에서 운영 체크 리스트의 목록을 파악할 수 있습니다.

A person's hands are shown holding a smartphone with a white screen. The background is dark with out-of-focus, colorful bokeh lights in shades of yellow, orange, and blue. A semi-transparent dark banner is at the bottom, containing a yellow decorative element and the title text.

1. AWS 지속적 코드 배포 방법

지속적 통합(Continuous integration)

- 빌드 및 테스트가 수행된 후, 개발자가 **코드 변경**을 중앙 레포지토리에 **정기적으로 통합**하는 소프트웨어 개발 방식이다.
- 소프트웨어 릴리스 프로세스 중 **빌드** 또는 **통합** 단계를 주로 가리키며, **자동화 구성 요소**와 **문화적 구성 요소**를 포함한다.

버그 신속 해결

소프트웨어
업데이트 검증

지속적 통합의
핵심 목표

소프트웨어
품질 개선

릴리스 시간 단축

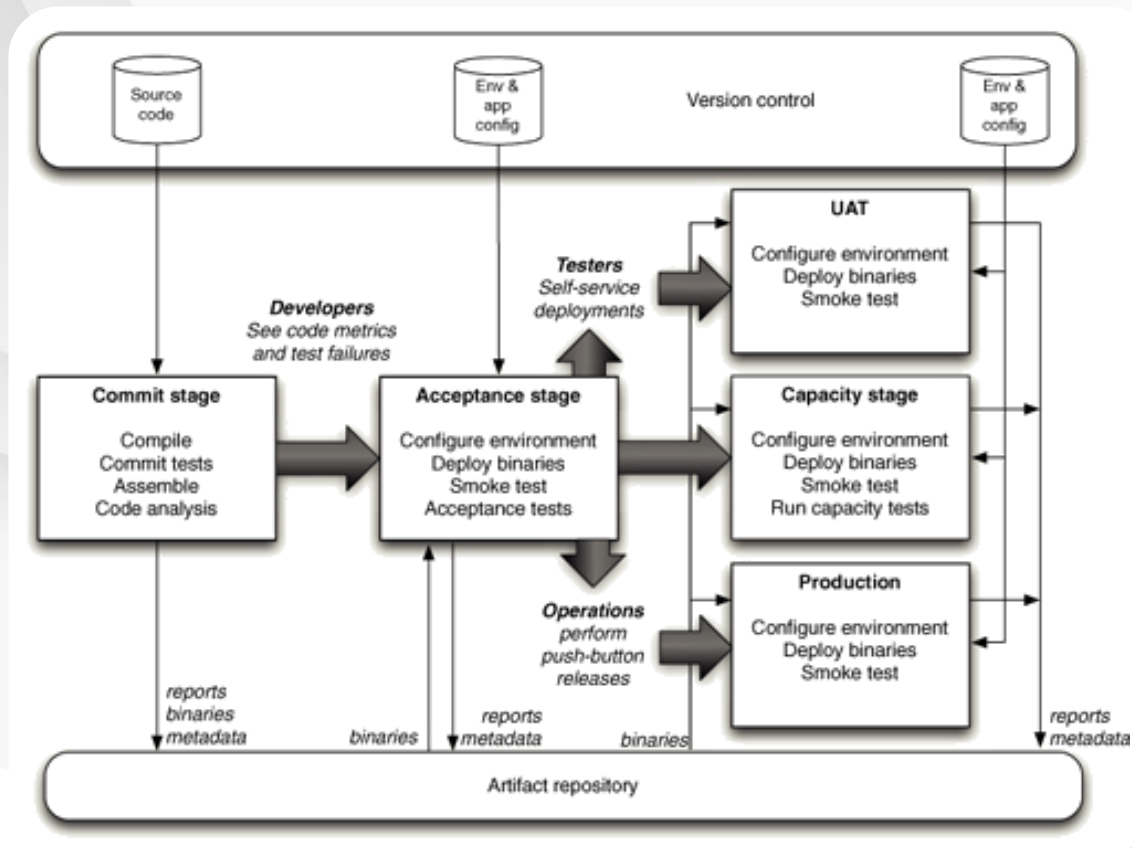
지속적 전달(Continuous delivery)

- ◉ 상용에 릴리즈하기 위한 코드 변경이 **자동으로 빌드, 테스트 및 준비**되는 소프트웨어 개발 방식이다.
- ◉ 빌드 단계 이후의 모든 코드 변경을 테스트 환경 및 프로덕션 환경에 배포함으로써 지속적 통합을 확장한다.

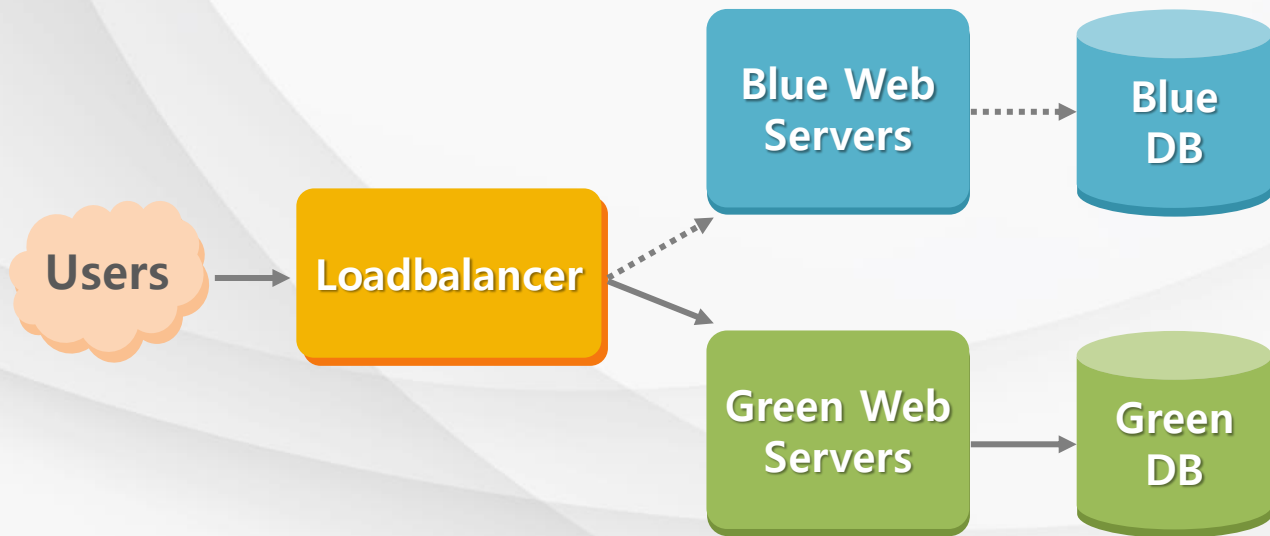
지속적 배포(Continuous deploy)

- ◉ 명시적 승인 없이 **자동으로 프로덕션 환경에 배포**되며 이를 통해 **전체 소프트웨어 릴리스 프로세스가 자동화**된다.

■ 기본 코드 배포 파이프라인



■ 효율적인 배포 정책(Blue/Green)



효율적인 배포 지원 시스템(Spinnaker)

The screenshot displays the Spinnaker Deck web interface. At the top, there's a navigation bar with tabs: PIPELINES (1), CLUSTERS, LOAD BALANCERS, SECURITY GROUPS, PROPERTIES, TASKS, and CONFIG. Below this, a header area shows 'deck' logo, a search bar, and buttons for 'Unpin', '+', '-', 'Group by Pipeline', 'Show 5 per group', 'Configure', and 'Start Manual Execution'.

On the left, a 'Filters' sidebar is visible with a 'Clear All' button. It contains sections for 'SEARCH', 'PIPELINES' (with a list of pipeline names like 'templation', 'wait forever on ci builds', etc., and 'demo' selected), and 'STATUS' (with options like 'Running', 'Failed', etc.).

The main content area shows a pipeline execution for 'demo'. It starts with a 'MANUAL START' section indicating the user 'chrisb@netflix.com' started it on '2015-11-09 21:41:16 PST'. The status is 'RUNNING' with a progress bar. Below this, a flow diagram shows the pipeline steps: 'Bake' (which branches into 'Verify Bake', 'Integration Tests', and 'Unit Tests'), followed by 'Deploy Canary', 'Manual Judgment: Go/No Go' (highlighted with a blue box), 'Deploy To Prod', 'Wait 4 hours', and 'Cleanup Old Server Groups'. A 'Hide Details' button is next to the flow diagram.

Below the flow diagram, there are two sections: 'MANUAL JUDGMENT: GO/NO GO DETAILS' and 'MANUAL JUDGMENT: GO/NO GO'. The first section contains a table with columns 'Step', 'Started', 'Duration', and 'Status'. The second section contains a 'Manual Judgment' and 'Task Status' area with instructions: 'Verify the canary scores are correct and you haven't received any emails or something.' and buttons for 'Continue' and 'Stop'.

At the bottom right, there's a 'Source | Permalink' link.

Step	Started	Duration	Status
Manual Judgment: Go/No Go	2015-11-09 21:42:10 PST	00:25	RUNNING

A person's hands are shown holding a smartphone with a white screen. The background is dark with out-of-focus, colorful bokeh lights in shades of yellow, orange, and blue. A semi-transparent dark blue banner is at the bottom, featuring a yellow diagonal bar on the left and the text '2. AWS 운영 방안' in yellow.

2. AWS 운영 방안

■ 계정 및 비용 관리

AWS를 운영하는 계정이 여러개 일 경우 **서로간에 접근 통제, 권한 부여 및 과금 관리** 등을 다양한 AWS의 tool, 서비스를 통해 효율적으로 관리 할 수 있도록 한다

- 보안 격리, 서비스 구분 회계 등과 같은 여러 사유로 인해 계정을 다수개로 운영 할 수 있으며 중앙 관리 부서 에서 통합 빌링(Consolidated Billing) 계정으로 과금 내역을 관리 한다.
- 통합 빌링(Consolidated Billing) 계정은 과금만 중앙에서 관리 가능하며 리소스에 대한 연결은 링크드 계정의 권한으로 사용 할 수 있다. 예를 들어 개발/QA/상용 처럼 환경의 목적에 맞게 계정을 분리 한다거나 비즈니스 군 별로 계정을 분리하여 운영 가능 하다

■ 자산 관리

배포된 자산에 대해 **추척 하거나 관리를 위해** 자산 관리 정책이 필요하며 효율적으로 자산 운영 할수 있도록 AWS 제공 기능 혹은 내부의 기술들을 활용 한다.

- 리소스에 대해 다양한 메타정보를 제공 하고 있으며 API, CLI를 통해 프로그래밍 가능하게 정보 수집이 가능 하다. 또한 커스텀 tag 기능을 통해 요구사항을 충족시킬수 있는 정보들을 직접 입력 및 관리 가능 하다.

■ 모니터링 및 사고 관리

어플리케이션(서비스)과 시스템 모니터링은 기본적인 운영 및 장애대응의 방법이며 비즈니스의 중요도에 준하여 알람 설정과 모니터링 정책들을 관리 한다

- CloudWatch 를 통해서 AWS의 리소스들의 모니터링 메트릭을 설정 가능하고 SNS, SMS를 통해 알람기능을 연동 하여 효과적인 운영을 가능 하게 한다.
- 서비스 중단 혹은 약속된 성능을 제공 하는데 실패한 경우 신속하게 정상 동작으로 서비스를 복구 해야 하며 서비스 성능의 저해 가능성이 있는 요소들을 제거 하고 방지 관리 한다.
- 인시던트 관리는 사고 인식, 로깅, 우선순위화, 초기 진단, 2선 에스컬레이션 와 같은 역할을 포함한다.

I 고가용성 및 탄력성 방안

효율적인 HA 정책에는 Multi AZ를 활용한 **인스턴스의 이중화 관리, 로드 밸런싱, 오토스케일링, 모니터링과 region 내 복구**를 포함하고 있다.

- 중요한 어플리케이션일수록 모든 SPOF의 사항을 확인하고, 어플리케이션의 가용 요구 사항과 위험 측면을 바탕으로 검토해야 한다.

I 고가용성 및 탄력성 방안

- ◉ EC2 인스턴스가 Multi AZ에 걸쳐서(이중화되어) 동작되고 있는가?
- ◉ Multi AZ에 걸쳐서 ELB를 활용하고 있는가?
- ◉ 자동화된 인스턴스의 확장과 복구를 위해 Autoscaling을 활용하고 있는가?
- ◉ CloudWatch 매트릭을 활용해 모니터링과 알람을 연동하고 있는가?
- ◉ Elastic IP의 Static 한 특성을 활용 하여 인스턴스에 매핑하였는가?

I 고가용성 및 탄력성 방안

- ◉ EBS 볼륨을 관리 하기 위해 Point-in-time snapshot을 활용하고 있는가?
- ◉ 오브젝트 저장을 위해 S3 혹은 콘텐츠 분배를 위해 CloudFront를 활용하고 있는가?
- ◉ Key/Value 형태의 데이터 저장을 위해 Simple DB나 Dynamo DB를 사용하고 있는가?
- ◉ 어플리케이션의 버전 관리와 환경 구성을 위해 Elastic Beanstalk를 활용하고 있는가?

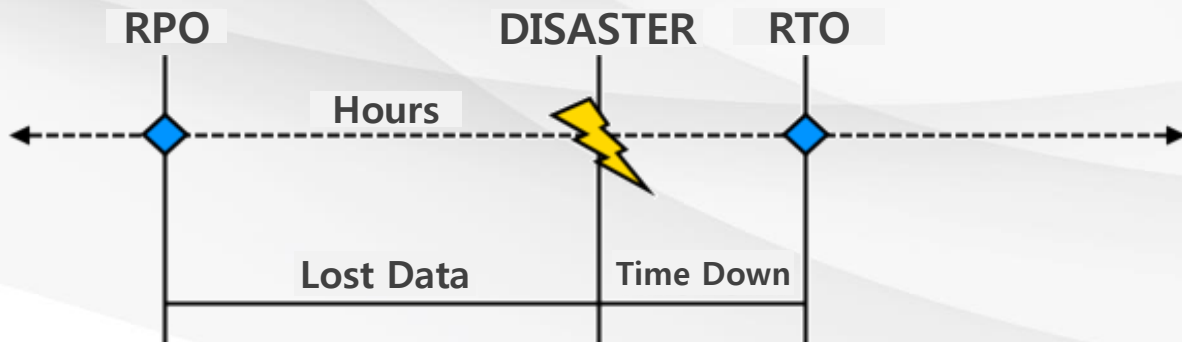
I 고가용성 및 탄력성 방안

- ◉ 복잡성과 확장성을 가진 어플리케이션의 관리를 위해 DevOps 솔루션인 OpsWorks를 활용하고 있는가?
- ◉ 데이터베이스 미러링과 같은 동기형 데이터 복제 기술을 사용하고 있는가?
- ◉ HA 리소스 용량까지 고려하여 AWS 리소스의 Quota를 추가하였는가?

■ DR 및 백업 관리

효율적인 DR 정책은 **단일 컴포넌트의 복구**뿐만이 아니고 애플리케이션 단에서의 **예상된 DR 시나리오의 수행** 등을 포함하고 있다.

- ◎ DR 정책은 지역 간 이중화, 글로벌 트래픽 관리 혹은 로드 밸런싱, 모니터링, 지역 대 지역 간 복구들을 포함해야 하며 아래와 같은 사항 및 기술들이 DR 정책에 고려되어야 한다.



I DR 및 백업 관리

- ◉ 데이터 저장, AMI 혹은 타 지역(region)에 추가 인스턴스가 있는지 확인하였는가?
- ◉ Route 53을 통해 DNS에 기반한 지역간 fail-over를 사용하고 있는가?
- ◉ Glacier를 통해서 데이터 보관을 하고 있는가?
- ◉ 데이터베이스 log shipping과 같은 비동기형의 데이터 복제 기술을 사용하고 있는가?
- ◉ EBS Snapshot 복제 와 혹은 AMI 복제를 지역 간 하고 있는가?

I DR 및 백업 관리

- ◉ S3의 Versioning을 활용하여 데이터 저장이나 보호를 하고 있는가?
- ◉ 데이터 손실로부터 빠른 복구를 위해 주기적인 EBS Snapshot 혹은 3rd party 툴을 활용하고 있는가?
- ◉ S3의 생명주기 정책(object lifecycle policies)을 활용하여 Glacier로의 보관을 하고 있는가?
- ◉ DR 리소스 용량까지 고려하여 DR 리존의 리소스의 Quota를 추가하였는가?

A person's hands are shown holding a smartphone, with the screen glowing. The background is dark with out-of-focus, colorful bokeh lights in shades of yellow, orange, and blue. A semi-transparent dark banner is at the bottom, featuring a yellow decorative element and the section title.

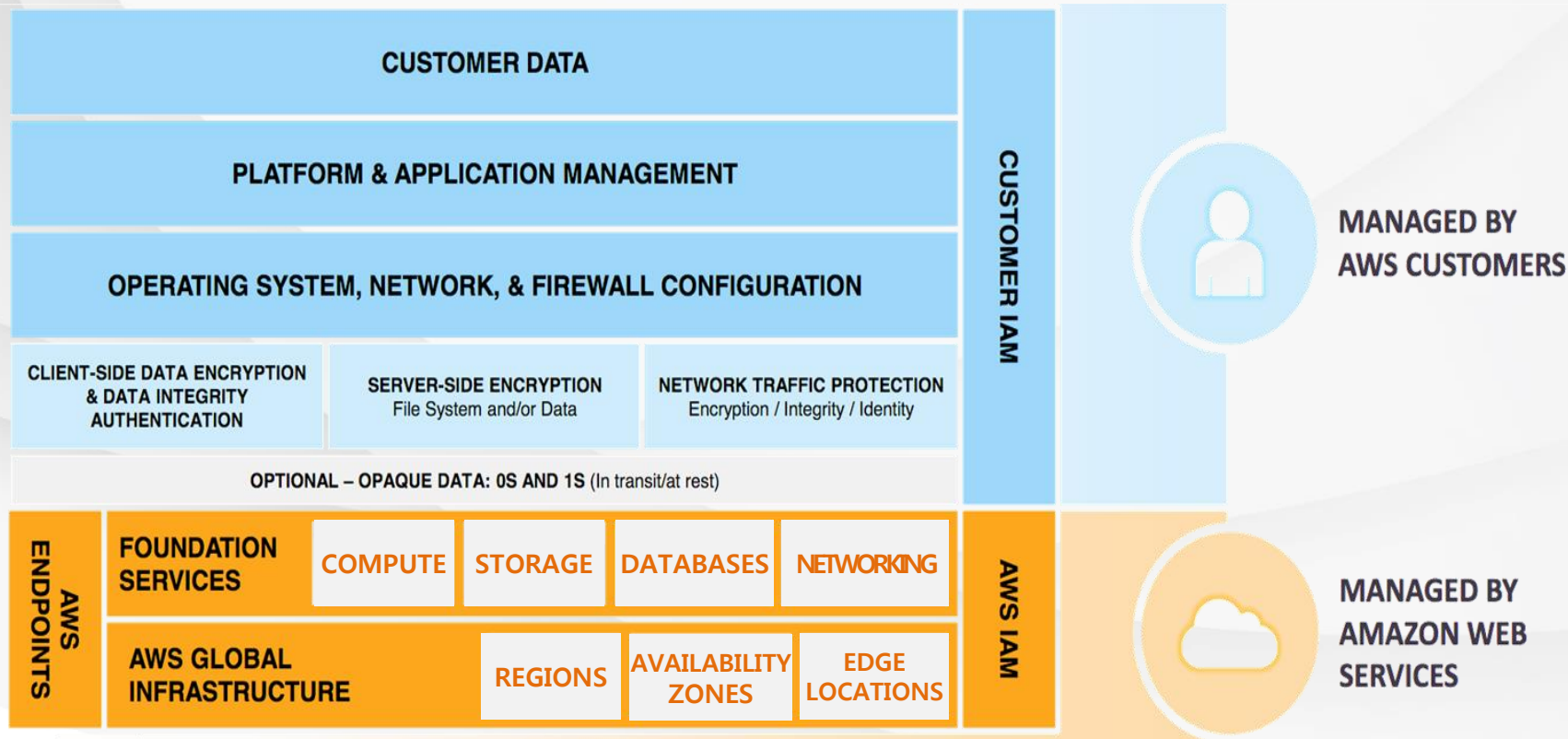
3. AWS 운영 체크리스트

I 기본 체크리스트

AWS를 통해 상용 레벨의 애플리케이션을 런칭하기 전에 검토해야 할 **체크리스트**이며 **잠재적인 위험요소와 통상적인 인프라 go-live에** 관련한 사항이다.

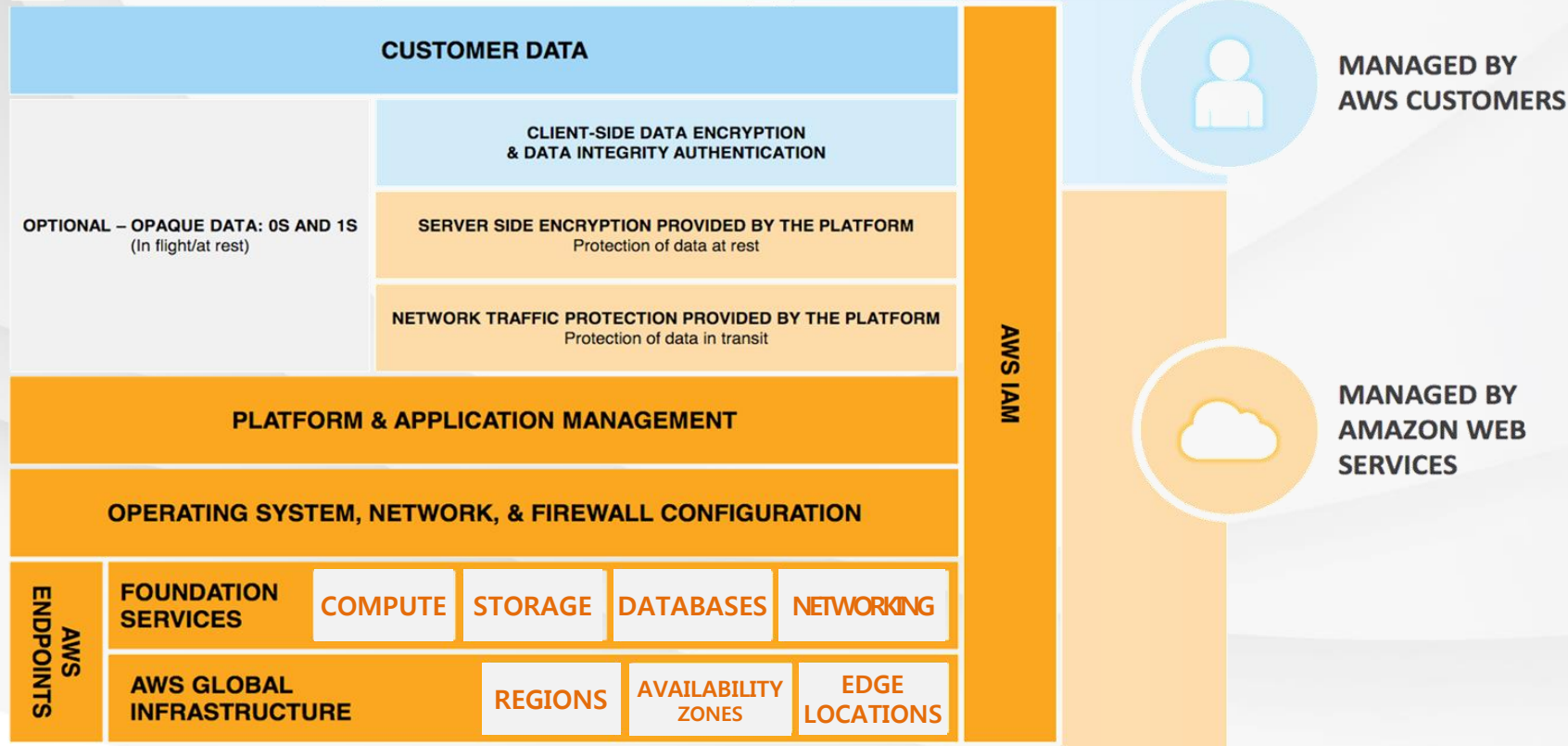
- 비용 관리 및 계정 관리를 위한 관리 시스템을 개발하였는가?
조직 안에 다수의 계정 과금에 대한 통합 관리 방안은 결정하였는가?
- AWS에서 사용 중인 리소스들의 확인 및 변경사항 추적을 위한 정책을 가지고 있는가?
- AWS API, 콘솔, OS, 네트워크와 데이터의 보안과 접근을 위한 정책을 개발하였는가?
- AWS 리소스와 연동하여 장애 관리 방법과 적합한 모니터링 툴을 산정하였는가?

■ Infrastructure 서비스 모델



※ 참고 : AWS Security Best 백서

■ Abstracted 서비스 모델



※ 참고 : AWS Security Best 백서



학습정리

지금까지 **[개발 및 운영 방안]**에 대해서 살펴보았습니다.

AWS 지속적 코드 배포 방법

지속적 통합, 지속적 전달, 기본 배포 파이프라인과 배포 방안의 개념을 토대로 **프로세스를 자동화** 시키고 **신속한 서비스 개발과 품질**을 올려 비즈니스의 가치를 높인다.

AWS 운영 방안

고가용성 및 탄력성, DR 및 백업 관리에 연관된 요소들을 이해하고 적용하여 높은 수준의 운영 정책을 수립한다.

AWS 운영 체크리스트

상용 레벨의 애플리케이션을 런칭하기 전에 **체크리스트를 검토**해 잠재적인 위험요소를 제거하며 **AWS 책임 분담 모델**에 의거하여 운영을 효과적으로 수행한다.