가용 영역을 사용한 정적 안정성

Becky Weiss, Mike Furr

Amazon 의 서비스는 매우 높은 가용성 목표에 따라 구축됩니다. 따라서 시스템의 종속 시스템에 각별히 주의를 기울여야 합니다. Amazon 은 그 종속 시스템에서 장애가 발생할 경우에도 복원력을 유지하도록 시스템을 설계합니다. 이 문서에서는 이처럼 높은 수준의 복원력을 실현하기 위한 소위 *정적 안정성*이라는 패턴을 정의하겠습니다. AWS 의 주요 인프라 빌딩 블록으로서 모든 서비스 구축에 있어 기반 종속 시스템이 되는 가용 영역에 이 개념을 적용하는 방법을 설명합니다.

정적 안정성 설계에서는 종속 시스템에서 장애가 발생하더라도 전체 시스템이 정상적으로 작동합니다. 종속 시스템에서 제공되어야 할 업데이트된 정보(예: 새로운 항목, 삭제된 항목 또는 수정된 항목)는 시스템에 표시되지 않을 수 있습니다. 하지만 종속 시스템에서 장애가 발생하기 전에 실행되던 기능은 종속 시스템이 손상되더라도 계속 정상적으로 작동합니다. Amazon Elastic Compute Cloud(EC2)가 어떻게 정적 안정성을 보장하도록 설계되는지 설명하겠습니다. 다음으로, 가용 영역을 기반으로 한 고가용성 리전별 시스템을 구축하는 데 유용한 2 개의 정적으로 안정적인 아키텍처 예제를 소개합니다.

마지막으로 소프트웨어 수준에서 가용 영역 종속성을 제공하도록 설계되는 방식을 비롯하여 Amazon EC2 이면의 설계 원리를 몇 가지 자세하게 살펴봅니다. 또한 이 아키텍처 옵션으로 서비스를 구축하는 것과 관련한 몇 가지 장단점도 설명합니다.

가용 영역의 역할

가용 영역은 논리적으로 격리된 AWS 리전의 섹션입니다. 각 AWS 리전에는 독립적으로 운영되도록 설계된 여러 개의 가용 영역이 있습니다. 가용 영역은 낙뢰, 토네이도, 지진 등 발생 가능한 문제로 인한 연쇄적 영향으로부터 보호하기 위해 물리적으로 유의미한 거리를 두고 설치됩니다. 전력이나 기타 인프라를 서로 공유하지 않지만, 애플리케이션이 중단 없이 신속하게 장애 조치될 수 있도록 빠르고 암호화된 프라이빗 광섬유 네트워크로 서로 연결되어 있습니다. 한마디로 가용 영역은 인프라 격리 환경에 일종의 추상화 계층을 제공합니다. 가용 영역을 필요로 하는 서비스는 이 같은 독립성의 이점을 활용할 수 있도록 호출자가 리전 내에서 인프라를 물리적으로 프로비저닝할 위치를 AWS 에 알릴 수 있도록 합니다. Amazon 은 이 영역별 독립성을 이용하여 자체적으로 고가용성 목표를 실현할 수 있도록 리전별 AWS 서비스를 구축했습니다. Amazon DynamoDB, Amazon Simple Queue Service(SQS), Amazon Simple Storage Service(S3) 등의 서비스가 이러한 리전별 서비스의 대표적인 예입니다.

이 같은 서비스의 경우 대부분 Amazon Virtual Private Cloud(VPC) 내에서 클라우드 인프라를 프로비저닝하는 AWS 서비스와 상호 작용할 때 호출자는 여러 서비스에서 리전뿐만 아니라 가용 영역도 지정해야 합니다. 가용 영역은 EC2 인스턴스를 시작하거나, Amazon Relational Database Service(RDS) 데이터베이스를 프로비저닝하거나, Amazon ElastiCache 클러스터를 생성하는 등의 경우에 필수 서브넷 인수를 통해 암시적으로 지정되는 경우가 많습니다. 가용 영역에 서브넷이 여러 개 있는 것이

일반적이지만, 단일 서브넷은 전적으로 단일 가용 영역 내에 상주하므로 호출자는 서브넷 인수를 제공함으로써 사용할 가용 영역을 암시적으로 지정하게 됩니다.

정적 안정성

가용 영역을 기반으로 시스템을 구축하면서 얻은 한 가지 교훈은 사전에 장애에 대비해야 한다는 것입니다. 특정 가용 영역 내에서 장애가 발생할 경우 서비스가 다른 가용 영역으로 확장(AWS Auto Scaling 을 사용하여)되어 완전히 정상 상태로 복원될 것이라는 기대로, 여러 가용 영역에 배포하는 방법은 효과적이지 않습니다. 이 방식은 장애가 발생하기 전에 대비하는 게 아니라 순전히 발생한 장애에 대응하는 데에만 의존하기 때문에 효과적이지 않습니다. 한마디로 정적 안정성이 부족합니다. 반면 가용 영역에서 장애가 발생하더라도 EC2 인스턴스를 새로 시작하지 않고도 계속 정상적으로 운영될 수 있을 정도로 인프라를 오버프로비저닝하는 서비스가 더 효과적이고 정적 안정성이 더 높습니다.

정적 안정성의 특성을 좀 더 잘 이해하기 위해 이 같은 원칙에 따라 설계된 Amazon EC2 를 살펴보겠습니다.

Amazon EC2 서비스는 제어 플레인과 데이터 플레인으로 구성됩니다. "제어 플레인"과 "데이터 플레인"은 네트워킹 분야에서 사용하는 기술 용어지만, Amazon 에서는 AWS 의 전반에 사용하고 있습니다. 제어 플레인은 리소스를 추가하거나 삭제하거나 수정하는 등 시스템을 변경하고, 그러한 변경 사항을 적용해야 할 모든 위치로 전파하는 데 관여하는 기계적 구성 요소를 말합니다. 반면 데이터 플레인은 그러한 리소스의 일상적인 작업, 즉 리소스가 작동하는 데 필요한 것을 나타냅니다.

Amazon EC2 에서는 EC2 에서 새 인스턴스가 시작될 때 발생하는 모든 작업이 제어 플레인에 해당합니다. 제어 플레인의 로직은 수많은 작업을 수행함으로써 새 EC2 인스턴스에 필요한 모든 요소가 유기적으로 작동하게 합니다. 다음은 그러한 작업의 몇 가지 예입니다.

- 배치 그룹 및 VPC 테넌시 요구 사항에 따라 컴퓨팅에 사용할 물리적 서버를 찾습니다.
- VPC 서브넷 외부에 네트워크 인터페이스를 할당합니다.
- Amazon Elastic Block Store(EBS) 볼륨을 준비합니다.
- AWS Identity and Access Management(IAM) 역할 자격 증명을 생성합니다.
- 보안 그룹 규칙을 설치합니다.
- 다양한 다운스트림 서비스의 데이터 스토어에 결과를 저장합니다.
- 필요한 구성을 VPC 의 서버와 네트워크 엣지에 적절하게 전파합니다.

반면 Amazon EC2 데이터 플레인은 기존 EC2 인스턴스의 정상적인 작동 상태를 유지하기 위해 다음과 같은 작업을 수행합니다.

- VPC 의 경로 테이블에 준하여 패킷을 라우팅합니다.
- Amazon EBS 볼륨에서 데이터를 읽고 씁니다.
- 기타 여러 가지 작업을 수행합니다.

일반적인 데이터 플레인 및 제어 플레인과 마찬가지로, Amazon EC2 데이터 플레인은 Amazon EC2 제어 플레인보다 훨씬 단순합니다. 상대적으로 단순하기 때문에 Amazon EC2 데이터 플레인은 Amazon EC2 제어 플레인보다 높은 가용성을 목표로 설계됩니다.

무엇보다, Amazon EC2 데이터 플레인은 제어 플레인의 가용성이 손상될 경우(예: EC2 인스턴스를 시작할 수 없는 경우)에도 정적 안정성을 유지하도록 세심하게 설계되었다는 점이 중요합니다. 예를 들어 네트워크 연결 장애를 방지하기 위해, Amazon EC2 데이터 플레인은 EC2 인스턴스를 실행하는 물리적 시스템이 VPC 내부와 외부의 지점으로 패킷을 라우팅하는 데 필요한 모든 정보에 로컬로 액세스할 수 있도록 설계되었습니다. Amazon EC2 제어 플레인에서 장애가 발생하면 해당 이벤트 동안 물리적 서버가 새 EC2 인스턴스가 VPC 에 추가되거나 새 보안 그룹 규칙이 생성되는 등의 업데이트를 감지하지 못하게 됩니다. 하지만 이벤트가 발생하기 전에 데이터를 전송하고 수신하던 트래픽은 계속 작동합니다.

제어 플레인, 데이터 플레인, 정적 안정성이라는 개념은 Amazon EC2 이외의 영역에도 광범위하게 적용할 수 있습니다. 시스템을 제어 플레인과 데이터 플레인으로 분리하는 것은 다음과 같은 여러 가지 이유로 고가용성 서비스를 설계하는 데 있어서 매우 유용한 개념적 도구가 될 수 있습니다.

- 일반적으로 데이터 플레인의 가용성은 제어 플레인의 가용성보다 서비스 고객의 성공적인 비즈니스 운영에 있어 훨씬 더 중요합니다. 예를 들어 대부분의 AWS 고객에게 있어 EC2 인스턴스가 실행된 이후의 지속적인 가용성과 정상적인 작동은 새 EC2 인스턴스를 시작하는 기능보다 훨씬 더 중요합니다.
- 일반적으로 데이터 플레인은 제어 플레인보다 더 큰 볼륨(몇 배에 달하는 경우가 많음)으로 운영됩니다. 따라서 두 플레인이 각각 적절한 규모로 확장 또는 축소될 수 있도록 서로 분리하는 것이 바람직합니다.
- 지난 몇 년간 시스템의 제어 플레인의 작동 부품이 데이터 플레인보다 많다는 사실을 발견했습니다. 즉, 통계적으로 볼 때 이 이유만으로도 고장이 발생할 가능성이 더 높습니다.

이 같은 고려 사항을 모두 종합할 때, 제어 플레인과 데이터 플레인의 경계에 따라 시스템을 분리하는 것이 좋습니다.

실제 구축 시에는 이 같은 분리 상태를 구현하기 위해 정적 안정성이라는 원칙을 적용합니다. 일반적으로 데이터 플레인은 제어 플레인에서 전송되는 데이터에 의존합니다. 하지만 더 높은 가용성 목표를 실현하기 위해 데이터 플레인은 제어 플레인에서 장애가 발생하더라도 기존 상태를 유지하고

계속 정상적으로 작동합니다. 장애가 발생한 동안 데이터 플레인이 업데이트되지 않을 수 있지만, 이전에 잘 작동하던 부분은 계속 정상적으로 작동합니다.

앞서 가용 영역에서 장애가 발생했을 때 EC2 인스턴스를 교체하는 방식의 대응 체계는 효과적이지 않다고 언급했습니다. 새 EC2 인스턴스를 시작할 수 없기 때문에 효과적이지 않다는 말이 아닙니다. 장애에 대응하기 위해 시스템이 Amazon EC2 제어 플레인의 복구 경로에 대한 직접적인 종속성을 가지게 되고, 새 인스턴스에 필요한 모든 애플리케이션별 시스템이 필요한 작업을 수행하게 되기 때문입니다. 애플리케이션에 따라 이 같은 종속성에는 런타임 구성을 다운로드하거나, 검색 서비스에 인스턴스를 등록하거나, 자격 증명을 획득하는 등의 작업 단계가 포함될 수 있습니다. 제어 플레인 시스템은 데이터 플레인의 시스템보다 복잡할 수밖에 없고, 전체 시스템에서 장애가 발생할 때 정상적으로 작동하지 않을 가능성이 더 높습니다.

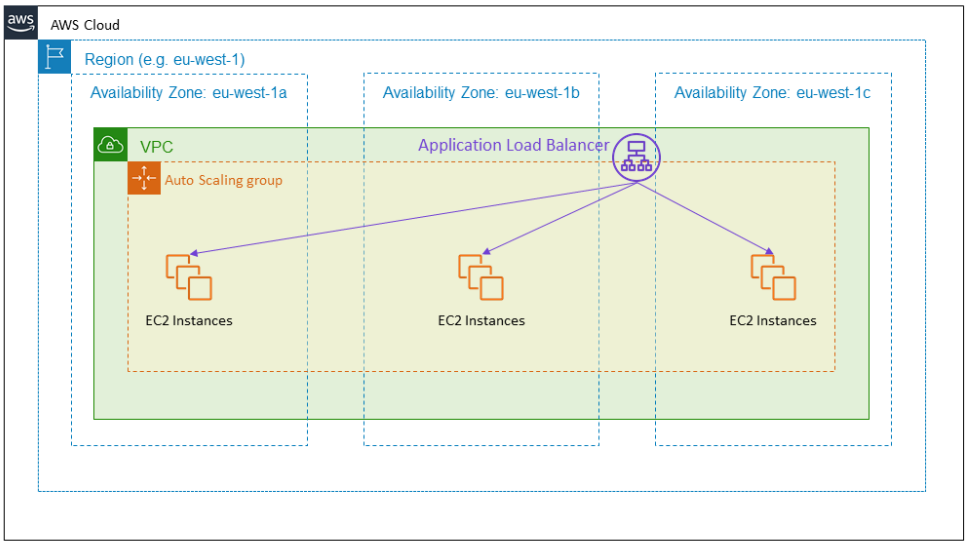
정적 안정성 패턴

이 섹션에서는 AWS 에서 정적 안정성을 활용하여 고가용성 시스템을 설계하기 위해 사용하는 두 가지 개략적인 패턴을 소개하겠습니다. 두 패턴은 적용되는 상황이 각기 다르지만 모두 가용 영역 추상화의 이점을 활용합니다.

가용 영역의 액티브-액티브 사례: 로드 밸런싱된 서비스

몇몇 AWS 서비스는 내부적으로, EC2 인스턴스 또는 Amazon Elastic Container Service(ECS) 컨테이너의 수평 확장이 가능한 상태 비저장 플릿으로 구성됩니다. 이러한 서비스는 3 개 이상의 가용 영역에 걸쳐 구성되는 Auto Scaling 그룹에서 실행됩니다. 또한 이들 서비스는 특정 가용 영역 전체에서 장애가 발생하더라도 나머지 가용 영역의 서버가 로드를 처리할 수 있도록 용량을 오버프로비저닝합니다. 예를 들어 3 개의 가용 영역을 사용할 경우 50% 오버프로비저닝합니다. 다시 말해, 각 가용 영역이 테스트된 로드 수준의 66%로만 작동하도록 오버프로비저닝합니다.

가장 일반적인 예는 로드 밸런싱된 HTTPS 서비스입니다. 다음 다이어그램은 HTTPS 서비스를 제공하는 공용 Application Load Balancer 를 보여 줍니다. 이 로드 밸런서의 대상은 eu-west-1 리전의 가용 영역 3 개에 걸쳐 구성된 Auto Scaling 그룹입니다. 이는 가용 영역을 사용한 액티브-액티브 고가용성 구축 사례입니다.



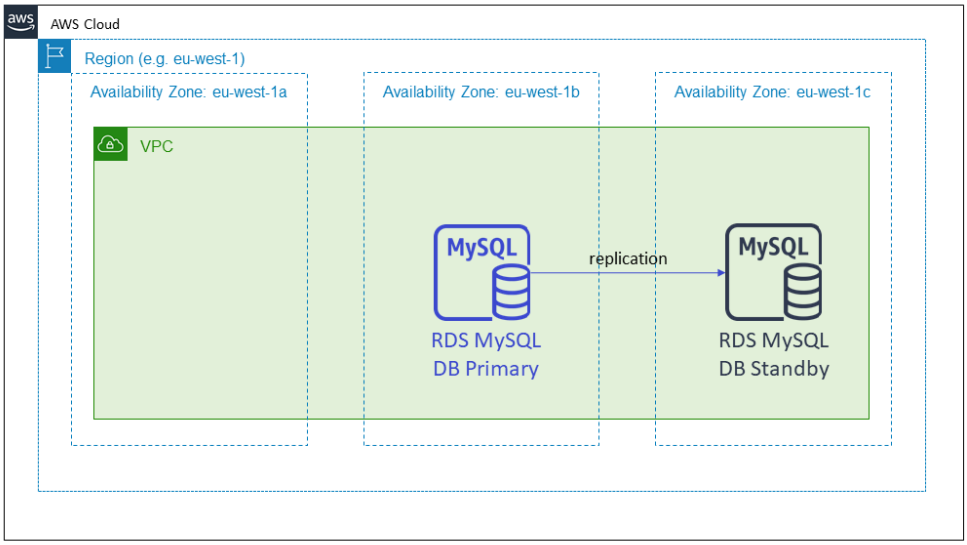
앞서 살펴본 다이어그램의 아키텍처에서는 가용 영역 중 하나에서 장애가 발생할 경우 어떤 조치도 취할 필요가 없습니다. 장애가 발생한 가용 영역의 EC2 인스턴스의 상태 점검이 실패하기 시작하고 Application Load Balancer 가 트래픽을 다른 곳으로 보냅니다. 사실 Elastic Load Balancing 서비스가 이 원칙에 따라 설계되었습니다. 이 서비스에는 가용 영역 장애가 발생하더라도 확장 없이 정상 작동하는 데 충분한 로드 밸런싱 용량이 프로비저닝되어 있습니다.

로드 밸런서나 HTTPS 서비스를 사용하지 않는 경우에도 이 패턴을 적용합니다. 일례로, Amazon Simple Queue Service(SQS) 대기열에서 받은 메시지를 처리하는 EC2 인스턴스 플릿도 이 패턴을 따를 수 있습니다. 이 같은 인스턴스는 여러 가용 영역에 걸쳐 구성된 Auto Scaling 그룹에 배포되어 적절하게 오버프로비저닝됩니다. 가용 영역 중 하나에서 장애가 발생하더라도 서비스에서는 아무 조치도 취하지 않습니다. 장애가 발생한 인스턴스는 작업을 중단하고 다른 인스턴스가 대신 해당 작업을 처리합니다.

가용 영역의 액티브-스탠바이 사례: 관계형 데이터베이스

Amazon 에서 구축하는 일부 서비스는 상태 저장 방식으로, 작업을 조정하는 단일 기본 노드 또는 리더 노드를 필요로 합니다. MySQL 또는 Postgres 데이터베이스 엔진을 사용한 Amazon RDS 와 같이 관계형 데이터베이스를 사용하는 서비스가 대표적인 예입니다. 이 같은 관계형 데이터베이스의 일반적인 고가용성 구성에는 모든 쓰기 작업이 반드시 전송되어야 하는 기본 인스턴스가 있고, 대기 인스턴스도 있습니다. 다음 다이어그램에는 나와 있지 않지만 추가 읽기 복제본이 생성될 수도 있습니다. 이 같은 상태 저장 인프라를 사용할 때는 기본 노드와 다른 가용 영역에 웜 대기 노드를 구축하게 됩니다.

다음 다이어그램에는 Amazon RDS 데이터베이스가 나와 있습니다. Amazon RDS 를 사용하여 데이터베이스를 프로비저닝하려면 서브넷 그룹이 필요합니다. *서브넷 그룹*은 데이터베이스 인스턴스가 프로비저닝되는 여러 가용 영역에 걸쳐 있는 일련의 서브넷입니다. Amazon RDS 는 대기 노드를 기본 노드와 다른 가용 영역에 배치합니다. 이는 가용 영역을 사용한 액티브-스탠바이 고가용성 구축 사례입니다.



상태 비저장, 액티브-액티브 사례와 마찬가지로 기본 노드가 있는 가용 영역에서 장애가 발생하더라도 상태 저장 서비스는 인프라에 대해 아무런 조치도 취하지 않습니다. Amazon RDS 를 사용하는 서비스의 경우, RDS 는 장애 조치를 관리하고 DNS 이름을 정상 작동하는 가용 영역의 새 기본값으로 다시 지정합니다. 이 패턴은 관계형 데이터베이스를 사용하지 않는 다른 액티브-스탠바이 설정에도 적용됩니다. 특히 리더 노드가 있는 클러스터 아키텍처를 사용한 시스템에 이 기능을 적용합니다. 해당 클러스터를 여러 가용 영역에 걸쳐 구축하고, "적시에" 대체 클러스터를 시작하는 것이 아니라 대기 클러스터에서 새 리더 노드를 선택합니다.

이 두 가지 패턴의 공통점은 가용 영역 중 하나에서 장애가 발생할 경우에 필요한 용량을 실제 장애가 발생하기 전에 이미 사전 프로비저닝한다는 점입니다. 두 패턴 모두에서 서비스는 가용 영역 장애에 대응하여 새 인프라를 프로비저닝하거나 수정 사항을 적용하는 등의 의도적인 제어 플레인 종속성을 취하지 않습니다.

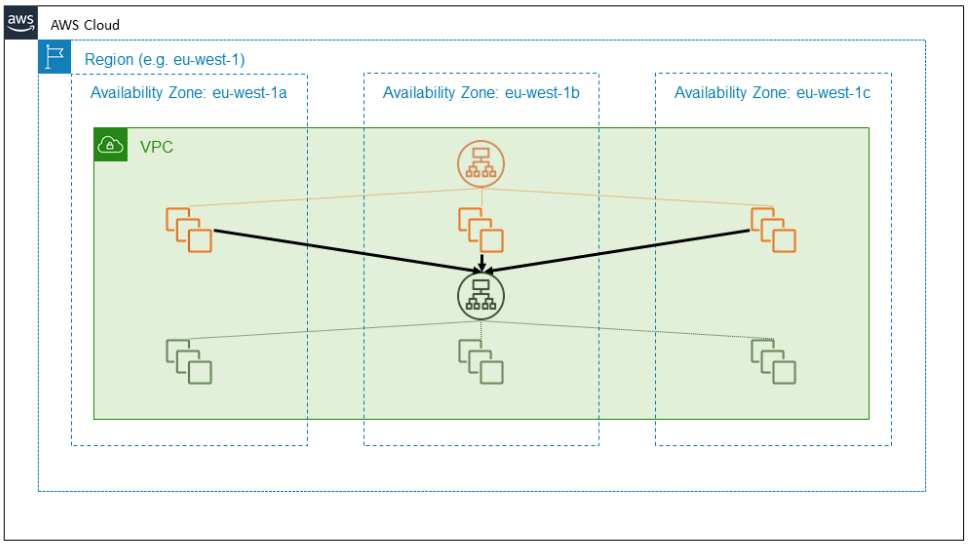
세부 설계: Amazon EC2 의 정적 안정성

문서의 이 마지막 섹션에서는 복원력이 뛰어난 가용 영역 아키텍처를 좀 더 자세히 살펴보면서 Amazon EC2 에서 가용 영역 독립성 원칙을 어떻게 따랐는지 몇 가지 측면에서 설명하도록 하겠습니다. 이러한 개념을 이해하면 자체적으로 고가용성을 유지해야 할 뿐만 아니라 다른 구성 요소의 고가용성을 보장하는 인프라까지 제공해야 하는 서비스를 구축할 때 도움이 됩니다. 하위 수준 AWS 인프라를 제공하는 서비스로서 Amazon EC2 는 고가용성을 보장하기 위해 애플리케이션에 사용되는 인프라입니다. 그런데 다른 시스템에 이 같은 전략을 적용해야 할 경우도 있습니다.

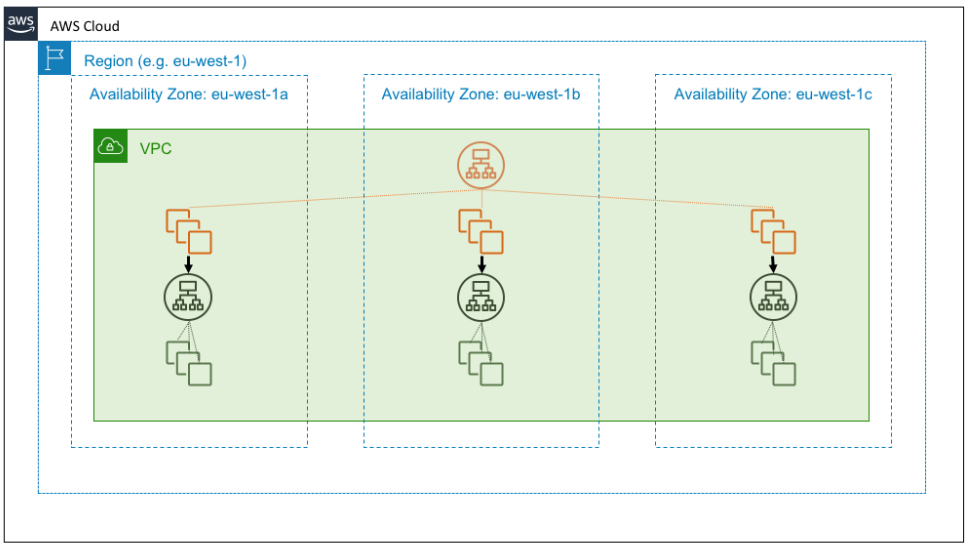
Amazon EC2 의 경우 배포 방식에서 가용 영역 독립성 원칙을 따릅니다. Amazon EC2 에서는 EC2 인스턴스, 엣지 디바이스, DNS 확인자, EC2 인스턴스 시작 경로의 제어 플레인 구성 요소 및 기타 EC2 인스턴스에 사용되는 여러 구성 요소를 호스팅하는 물리적 서버에 소프트웨어가 배포됩니다. 이 배포 방식에서는 영역별 배포 일정을 따릅니다. 즉, 같은 리전에 있는 2 개의 가용 영역에 서로 다른 날에 배포합니다. AWS 전반에서 배포는 단계별로 이루어집니다. 예를 들어 (배포하는 대상 서비스의 유형에 관계없이) 먼저 시스템을 하나 배포한 후 한 번에 1/N 개의 서버를 배포하는 식의 모범 사례를 따릅니다. 하지만 Amazon EC2 의 서비스와 같은 특정 서비스의 경우 이 같은 배포 방식을 한 단계 발전시켜 가용 영역 경계에 의도적으로 맞춥니다. 이렇게 하면 배포에서 발생하는 문제가 하나의 가용 영역에만 영향을 미치고, 롤백되어 해결됩니다. 다른 가용 영역은 영향을 받지 않으므로 계속 정상적으로 작동합니다.

Amazon EC2 에서 서비스를 구축할 때 독립적인 가용 영역의 원칙을 적용하는 또 다른 방법은 경계를 넘지 않고 가용 영역 내 머물도록 모든 패킷 흐름을 설계하는 것입니다. 네트워크 트래픽이 가용 영역 내에 로컬로 유지되도록 하는 이 두 번째 방식은 좀 더 자세히 살펴볼 필요가 있습니다. 두 번째 방식은 독립적인 가용 영역의 소비자(즉, 가용 영역 독립성 보장을 고가용성 서비스 구축의 기초로 사용)로서 리전별 고가용성 시스템을 구축할 때, 고가용성을 구축할 수 있도록 가용 영역 독립적 인프라를 제공할 때와는 다르게 어떻게 발상을 전환해야 하는지 잘 보여줍니다.

다음 다이어그램은 다른 내부 서비스(녹색)에 의존하는 고가용성 외부 서비스(주황색)를 보여줍니다. 단순한 설계 방식에서는 두 서비스를 모두 독립적인 EC2 가용 영역의 소비자로 취급합니다. 주황색 서비스와 녹색 서비스의 프런트엔드에는 각각 Application Load Balancer 가 배치되고,각 서비스의 충분히 용량이 프로비저닝된 백엔드 호스트 플릿은 3 개의 가용 영역에 걸쳐 분산됩니다. 하나의 고가용성 리전별 서비스가 다른 고가용성 리전별 서비스를 호출하는 것입니다. 이는 단순한 설계 방식으로, 대부분의 AWS 서비스에 효과적으로 적용됩니다.



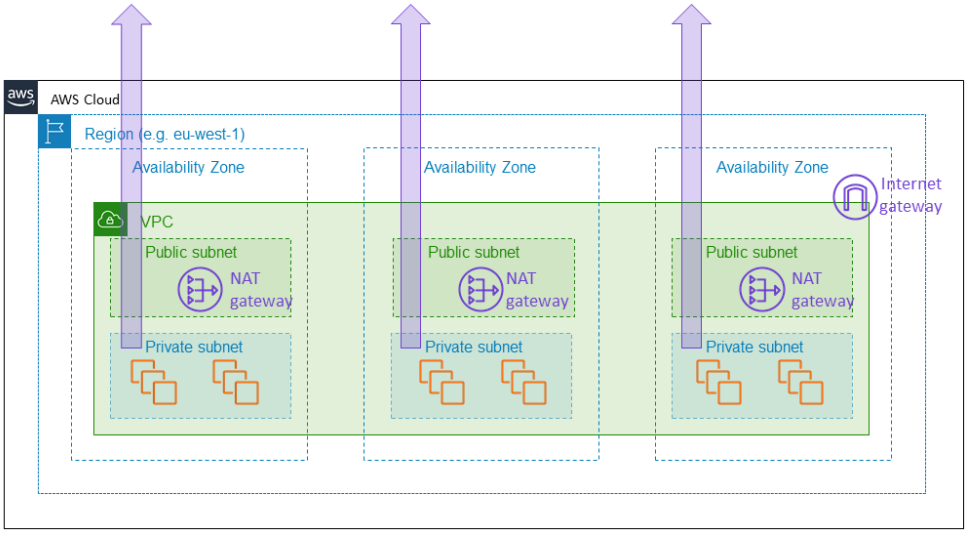
하지만 녹색 서비스가 기반 서비스라면 어떨까요? 즉, 고가용성을 유지해야 할 뿐만 아니라 그 자체가 가용 영역 독립성을 제공하기 위한 하나의 빌딩 블록 역할을 해야 하는 것입니다. 이 경우 가용 영역 인식 배포 방식에 따라 이 서비스를 영역-로컬 서비스의 3 개 인스턴스로 설계할 수 있습니다. 다음 다이어그램은 고가용성 리전별 서비스가 고가용성 영역별 서비스를 호출하는 설계를 보여줍니다.



빌딩 블록 서비스를 가용 영역 독립적으로 설계하는 이유는 단순한 계산으로 설명할 수 있습니다. 가용 영역 중 하나에서 장애가 발생했다고 가정해보겠습니다. 블랙 장애와 화이트 장애가 발생하면 Application Load Balancer 는 영향을 받은 노드로부터 자동으로 장애 조치합니다. 그러나 모든 장애가 명백하게 나타나는 것은 아닙니다. 소프트웨어의 버그와 같은 그레이 장애도 있습니다. 이 같은 장애는 로드 밸런서가 상태 점검에서 감지하지 못하고 완벽하게 처리하지 못합니다.

고가용성 리전별 서비스가 다른 고가용성 리전별 서비스를 호출하는 앞서 언급한 예에서, 시스템을 통해 요청이 전송되면, 단순하게 가정할 경우 요청이 장애가 발생한 가용 영역을 피할 확률은 2/3 * 2/3 = 4/9 가 됩니다. 즉, 요청이 장애 이벤트를 피할 확률이 50 대 50 보다 낮은 것입니다. 반면 현재 예에서와 같이 녹색 서비스를 영역별 서비스로 구축하면 주황색 서비스의 호스트가 같은 가용 영역의 녹색 엔드포인트를 호출할 수 있습니다. 이 아키텍처에서는 장애가 발생한 가용 영역을 피할 확률이 2/3 입니다. 이 호출 경로가 N 개의 서비스로 구성된다면 이 수치는 N 개의 리전별 서비스에 대해 (2/3)^N 이 됩니다. N 개의 영역별 서비스로 구성될 경우에는 2/3 의 상수로 남게 되는 것과는 대조적입니다.

Amazon EC2 NAT 게이트웨이를 영역별 서비스로 구축한 이유가 여기에 있습니다. NAT 게이트웨이는 프라이빗 서브넷에서 발생하는 아웃바운드 인터넷 트래픽을 허용하는 Amazon EC2 기능으로, VPC 전체에 걸친 리전별 게이트웨이가 아니라 영역별 리소스로 표시되며 다음 다이어그램과 같이 고객이 가용 영역에 따라 별도로 인스턴스화합니다. NAT 게이트웨이는 VPC 의 인터넷 연결 경로상에 위치하므로 VPC 내 모든 EC2 인스턴스의 데이터 플레인에 속합니다. 가용 영역 중 하나에서 연결 장애가 발생할 경우 장애가 다른 영역으로 확산되지 않도록 장애의 범위를 해당 가용 영역 내부로 한정해야 합니다. 결국, 이 문서의 앞부분에서 언급한 것과 유사한 아키텍처(2 개의 가용 영역으로 전체 로드를 처리할 수 있도록 충분한 용량으로 3 개의 가용 영역에 걸쳐 플릿 프로비저닝)를 구축하는 고객에게 강조하고 싶은 점은 장애가 발생한 가용 영역의 문제가 다른 가용 영역에 아무런 영향도 미치지 않는다는 사실입니다. 이렇게 할 수 있는 유일한 방법은 NAT 게이트웨이와 같은 모든 기반 구성 요소가 가용 영역 내에 머물도록 하는 것입니다.



이렇게 할 경우 복잡성이 가중됩니다. Amazon EC2 에서는 리전별 서비스 환경이 아니라 영역별 서비스를 관리해야 한다는 점에서 복잡성이 가중됩니다. NAT 게이트웨이의 고객 입장에서는 VPC 의 여러 가용 영역에 사용할 여러 개의 NAT 게이트웨이와 라우팅 테이블을 구성해야 한다는 점에서 복잡성이 가중됩니다. NAT 게이트웨이 자체가 영역별 가용성을 보장해야 하는 Amazon EC2 데이터 플레인을 구성하는 기반 서비스이기 때문에 이 같은 복잡성은 타당하다고 할 수 있습니다.

가용 영역이 독립적이고 데이터 내구성이 높은 서비스를 구축할 때 고려해야 할 사항이 하나 더 있습니다. 앞서 설명한 각각의 영역별 아키텍처에서는 전체 스택이 단일 가용 영역 내 구축되어 있지만, 실제로는 재해 복구를 목적으로 여러 가용 영역에 걸쳐 하드 상태를 복제합니다. 예를 들어 일반적으로 Amazon S3 에 주기적으로 데이터베이스 백업을 저장하고 데이터 스토어의 읽기 복제본을 가용 영역 경계 밖에 유지합니다. 이들 복제본은 기본 가용 영역이 작동하는 데 꼭 필요한 것은 아닙니다. 대신 고객 데이터 또는 비즈니스 크리티컬 데이터가 여러 장소에 저장되도록 보장하는 역할을 합니다.

AWS 에서 실행되는 서비스 지향 아키텍처를 설계하면서 이 두 가지 패턴 중 하나 또는 두 가지 모두를 사용하는 방법을 배웠습니다.

- 상대적으로 단순한 패턴: 리전별 서비스가 다른 리전별 서비스 호출. 외부용 서비스에는 이 패턴이 가장 적합한 경우가 많으며, 대부분의 내부 서비스에도 적합합니다. 예를 들어 AWS 에서 Amazon API Gateway 및 AWS 서버리스 기술과 같은 상위 수준 애플리케이션 서비스를 구축하는 경우 이 패턴을 사용하여 가용 영역 장애가 발생한 경우에도 고가용성을 보장합니다.
- 상대적으로 복잡한 패턴: 리전별 서비스가 영역별 서비스 호출 또는 영역별 서비스가 다른 영역별 서비스 호출. Amazon EC2 내에 내부(일부 경우에 외부) 데이터 플레인 구성 요소를 설계하는 경우(예: 중요 데이터 경로상에 바로 위치한 네트워크 애플리케이션 또는 기타 인프라) 가용 영역 독립성 패턴을 따르고 가용 영역에 사일로로 구축되는 인스턴스를 사용하여 네트워크 트래픽이 해당 가용 영역을 벗어나지 않도록 합니다. 이 패턴은 장애 범위가 특정 가용 영역으로 국한되도록 할 뿐만 아니라 AWS 에서 네트워크 트래픽 비용의 측면에서도 유리합니다.

결론

이 문서에서는 가용 영역에 대한 종속성을 성공적으로 극복하기 위해 AWS 가 사용하는 몇 가지 간단한 전략을 살펴보았습니다. 정적 안정성의 핵심은 발생하기 전에 장애를 예측하는 것이라는 점을 배웠습니다. 수평 확장 가능한 액티브-액티브 플릿을 기반으로 시스템을 실행하든, 상태 저장 액티브-스탠바이 패턴을 기반으로 하든 관계없이 가용 영역을 사용하여 높은 수준의 가용성 목표를 실현할 수 있습니다. AWS 는 장애 발생 시에 필요한 모든 용량이 사전에 완벽하게 프로비저닝되어 즉시 사용될 수 있도록 시스템을 배포합니다. 마지막으로, Amazon EC2 자체에서 가용 영역 간의 독립성을 유지하는 데 정적 안정성 개념이 어떻게 사용되는지 자세히 살펴보았습니다.