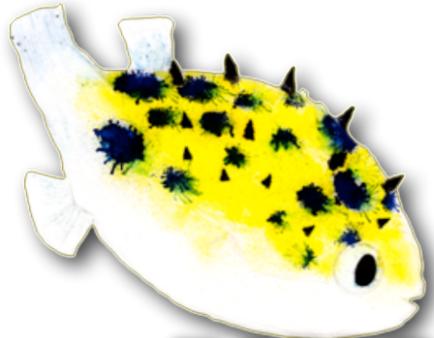


강의 노트 : [https://bit.ly/202504\\_Li\\_adv1](https://bit.ly/202504_Li_adv1)



# AWS 서비스 및 개발자 도구를 활용한 클라우드 애플리케이션 개발

nowage@gmail.com

# Agenda



- Cloud 이해
- AWS 소개
- EC2 (Elastic Compute Cloud)
- IAM (Identity and Access Management)
- Terraform (인프라스트럭처 코드 관리 도구)
- EBS (Elastic Block Store)
- 빅데이터 솔루션 프로비저닝

# 수업 진행



- 교재는 PDF로 제공됩니다. (종이 교재는 참고용)
- 강의 노트에서 질문도 가능하고 수업 중 제공되는 추가 정보와 스크립트도 확인할 수 있습니다.
- 강의 노트의 "강의 파일 구글 드라이브"에서 강의 **스크린샷** 및 강사 제공 파일을 다운로드할 수 있습니다.
- 강의 스크린샷을 통해 놓친 실습을 따라갈 수 있습니다.
- 각 실습이 끝나면 강의 노트의 "**Progress**" 탭에 완료 체크 해주시기 바랍니다.



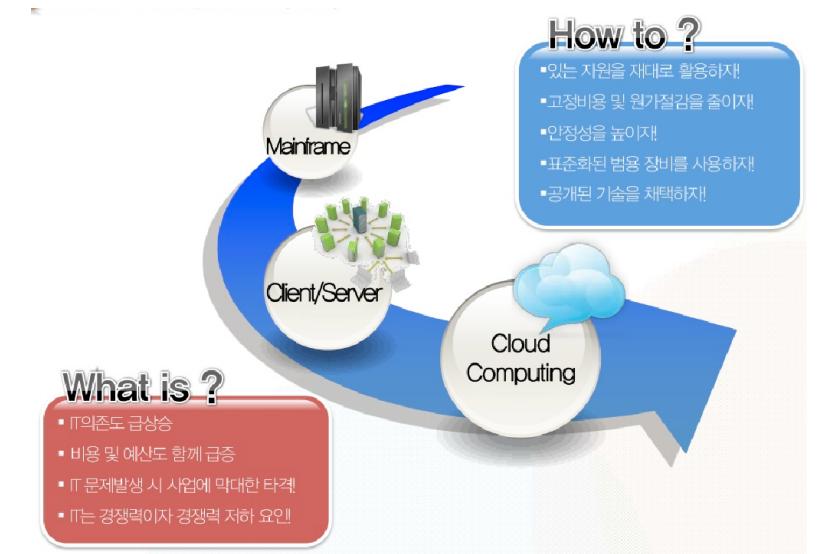
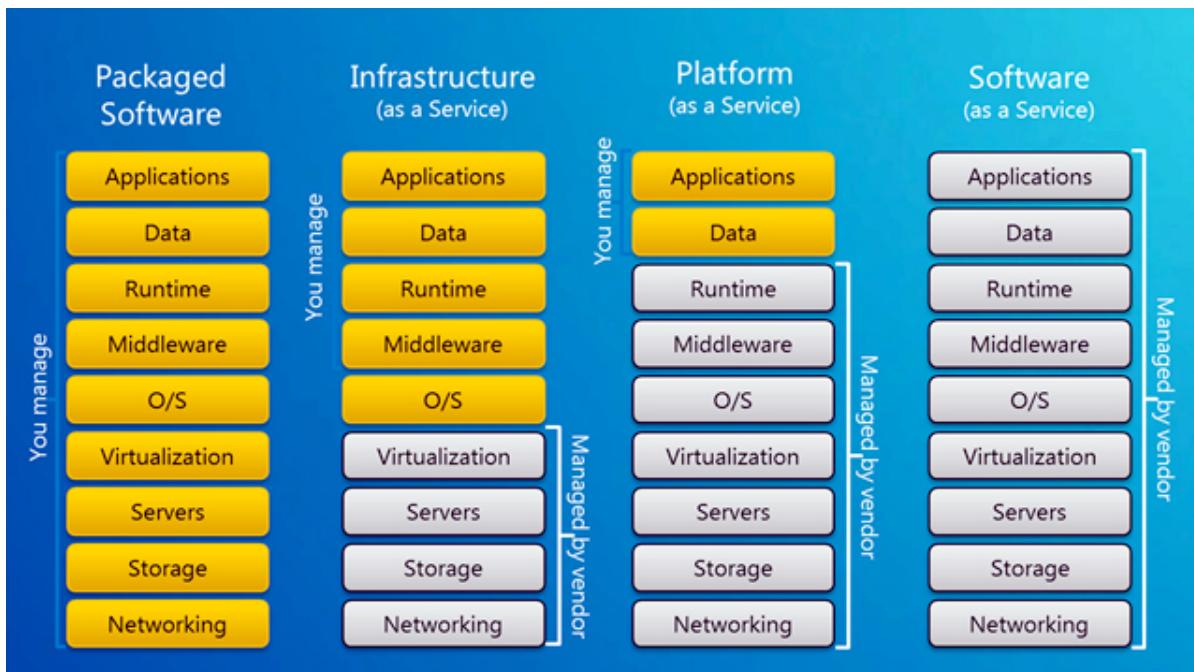
# Cloud 이해



# Cloud란?

- 클라우드는 기본적으로 멀티플랫폼 기반임.  
but, 어떤 플랫폼을 지원 하는지의 차이임.  
then, 클라우드란? 결국,  
“돈주고 컴퓨팅 자원을 사서 쓰자!”

[Software, Platform, Infrastructure] as a Service



Linode 1GB	Linode 2GB	Linode 4GB
\$5/mo (\$.0075/hr)	\$10/mo (\$.015/hr)	\$20/mo (\$.03/hr)
1 GB RAM	2 GB RAM	4 GB RAM
1 CPU Core	1 CPU Core	2 CPU Cores
20 GB SSD Storage	30 GB SSD Storage	48 GB SSD Storage
1 TB Transfer	2 TB Transfer	3 TB Transfer
40 Gbps Network In	40 Gbps Network In	40 Gbps Network In
1000 Mbps Network Out	1000 Mbps Network Out	1000 Mbps Network Out
<a href="#">Sign Up</a>	<a href="#">Sign Up</a>	<a href="#">Sign Up</a>



# 클라우드 컴퓨팅의 장점

## 비용 절감

- 물리적 하드웨어 구매 및 유지보수 비용 절감
- 사용한 만큼만 지불하는 종량제 모델

## 확장성 및 유연성

- 필요에 따라 리소스를 쉽게 확장하거나 축소 가능
- 서버 및 스토리지를 실시간으로 추가하거나 제거 가능

## 고가용성 및 신뢰성

- 다중 리전과 가용 영역을 통해 데이터 복구 및 서비스 안정성 보장
- 자동으로 장애를 감지하고 복구하는 기능 제공



# 클라우드 컴퓨팅이란?

“ 서버를 빌려 쓰는 것! ”

## 1. 인터넷을 통한 접근

어디서나 필요한 컴퓨팅 자원을 주문형으로  
제공받을 수 있음.

## 2. 하드웨어 구입 및 관리 부담 없음

사용자는 IT 자원을 신속하게 배포할 수 있음.



# 클라우드 분류

- 개인용 클라우드  
(Personal Cloud)
- 기업용 클라우드  
(Enterprise Cloud)



iCloud



Dropbox



Cloud Services



rackspace<sup>®</sup>  
HOSTING



# 클라우드 서비스 모델 (IaaS, PaaS, SaaS)



## IaaS(Infrastructure as a Service)

기본 컴퓨팅 인프라를 제공함. 이는 서버, 가상 머신, 스토리지, 네트워크 및 운영 체제를 포함함.



## PaaS(Platform as a Service)

소프트웨어 개발 및 배포 플랫폼을 제공함. 이는 개발 도구, 데이터베이스 관리, 비즈니스 인텔리전스(BI) 서비스 등을 포함함.



## SaaS(Software as a Service)

인터넷을 통해 애플리케이션 사용을 제공함. 이는 이메일, 칼렌더, CRM 및 다른 비즈니스 앱을 포함함.



# 클라우드 배포 모델 (공개, 사설, 하이브리드, 커뮤니티)

## 공개 클라우드

- 일반 대중이 사용 가능한 클라우드임. 클라우드 서비스 제공업체가 소유하고 운영함.

## 사설 클라우드

- 단일 기업이 전용으로 사용하는 클라우드임. 기업 내부 또는 호스팅 서비스 제공업체가 관리할 수 있음.

## 하이브리드 클라우드

- 공개 및 사설 클라우드의 혼합임. 이는 데이터 및 애플리케이션을 공유할 수 있음.

## 커뮤니티 클라우드

- 특정 커뮤니티가 공유하는 클라우드임. 이는 특정 조직 그룹이 공유하며, 보안, 규정 준수 및 거버넌스 요구 사항을 충족함.



# 클라우드의 장점 : 라우드 컴퓨팅은 다음과 같은 주요 이점을 제공함

## 비용 절감

- 물리적 인프라 구입 및 유지 관리 비용을 절감할 수 있음.

## 확장성

- 필요에 따라 자원을 쉽게 추가 또는 제거할 수 있음.

## 접근성

- 언제 어디서나 인터넷을 통해 클라우드 서비스에 접근할 수 있음.

## 백업 및 복구

- 클라우드는 데이터 백업 및 복구를 용이하게 함.



# 클라우드의 단점

## 보안 우려

외부 서비스 사용으로 인한 데이터 보안 문제가 있을 수 있음.

## 종속성

특정 클라우드 제 공업체의 서비스에 대한 의존도가 증가할 수 있음.

## 대역폭 제한

대용량 데이터 전송 시 비용이 증가하고 속도가 저하될 수 있음.



# 클라우드의 사용 사례

## 데이터 백업 및 재해 복구

- 클라우드는 데이터를 안전하게 저장하고, 재해 시 빠르게 복구할 수 있는 기능을 제공함.

## 가상 데스크톱 제공

- 클라우드는 사용자가 어디서나 자신의 데스크톱에 접근할 수 있게 함.

## 빅 데이터 분석

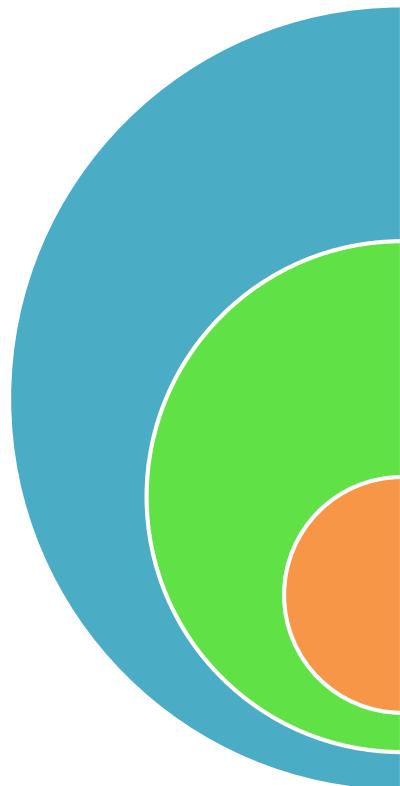
- 클라우드는 대량의 데이터를 저장하고 분석하는 데 필요한 처리 능력을 제공함.

## 소프트웨어 개발 및 테스트

- 클라우드는 개발자가 애플리케이션을 개발하고 테스트하는 데 필요한 환경을 제공함.



# 클라우드 보안 개요



## 멀티 테넌시

- 리소스 공유로 인한 데이터 유출 위험을 관리함.

## 데이터 암호화

- 저장 및 전송 중 데이터를 보호하기 위해 데이터를 암호화함.

## 접근 제어

- 사용자 인증 및 권한 관리를 통해 누가 데이터에 접근할 수 있는지 제어함.



# 클라우드 인프라 예

“ 클라우드 인프라의 배포 구성 단위는 Region > Zone > Pod > Cluster > Host ”  
 (Cloud Stack 기준)

**리전(Region)** – 국가/대륙 단위, DR(Disaster Recovery) 관점

**존(Zone)** – 데이터 센터 단위, HA(High Availability) 관점

**파드(Pod)** – 랙 단위

**클러스터(Cluster)** – 서버 묶음 단위

**호스트  
(Host)**

**프라머리  
스토리지**



**존(Zone)**

**동일한 물리적 단위  
(하이퍼바이저, 서버 스펙...)**

**가용존  
(Availability Zone)**



# 클라우드 공급자 선택 기준

서비스 안정성 및 가용성

- 공급자가 제공하는 서비스의 안정성과 가용성을 확인함.

데이터 보안 및 개인 정보 보호 정책

- 공급자의 데이터 보안 및 개인 정보 보호 정책을 검토함.

비용 구조 및 계약 조건

- 서비스의 비용 구조와 계약 조건을 이해하고 비교함.

기술 지원 및 서비스 수준 협약(SLA)

- 공급자가 제공하는 기술 지원 및 SLA를 검토함.

# 클라우드 채택에 따른 조직적 변화



기술 역량 강화

- 직원들에게 클라우드 기술에 대한 교육을 제공하고, 필요한 기술 지원을 받음.



업무 프로세스 재정립

- 클라우드 기반 작업 흐름을 채택하여 업무 프로세스를 재정립함.



문화적 적응

- 혁신적 기술을 수용하는 조직 문화를 적응시킴.

# On-Premises, Public Cloud, Private Cloud

항목	On-Premises	Private Cloud	Public Cloud
인프라 소유	자체 보유 (기업이 직접 소유)	기업 또는 제3자가 전용으로 소유	클라우드 사업자 소유
인프라 위치	내부 데이터센터	내부 또는 외부 전용 환경	클라우드 제공자 데이터센터
접근 방식	내부 네트워크만 접근 가능	기업 내부망 또는 전용 연결	인터넷을 통한 접근
초기 투자 비용	높음 (CapEx 중심)	중간 (CapEx + OpEx 혼합)	낮음 (OpEx 중심)
유지보수 책임	전적으로 기업 책임	기업 또는 관리 위탁업체가 담당	클라우드 제공자 책임
확장성	느림 (물리 자원 추가 필요)	제한적이나 유연성 확보 가능	매우 높음 (자동 확장 가능)
보안 및 통제	최고 수준의 통제 가능	높은 보안 통제 가능 (전용 환경)	통제 제한적 (공유 환경)
규제 및 컴플라이언스	직접 대응 필요	규제 요건 충족 용이	일부 산업 규제에 한계 있음
대표 기술/서비스	VMWare, Bare Metal 등	OpenStack, VMWare vCloud 등	AWS, Azure, GCP 등

# 업무 자동화 도입의 장애물과 극복 전략

## 장애물

- 기술적 한계: 복잡한 업무를 자동화 도구로 처리 어려움.
- 저항감: 변화에 대한 직원들의 저항.
- 비용: 과도한 초기 도입 비용.
- 교육 및 적응: 새 도구에 대한 직원 교육 및 적응 필요.

## 극복 전략

- 단계적 접근: 점진적 자동화 도입.
- 교육 및 지원: 충분한 교육 및 도구 사용의 이점 강조.
- ROI 분석: 초기 비용 대비 장기 이익 분석.
- 사용자 친화적 도구 선택: 사용 용이성을 갖춘 도구 선택.



# AWS 소개



<https://aws.amazon.com/>

# AWS 소개: 세계 최대의 클라우드 서비스 제공업체

## 세계적인 범위

- 전 세계 데이터 센터 네트워크를 통한 서비스 제공.

## 다양한 고객층

Type the Return Keystroke

- 스타트업부터 대기업까지 다양한 고객에 서비스 제공.

## 혁신적 리더

- 클라우드 컴퓨팅 분야에서 지속적인 혁신과 성장.



# AWS의 역사와 배경

## 역사

- 2006년에 Amazon이 최초로 AWS 서비스 시작
- 초기에는 단순한 스토리지(S3)와 컴퓨팅(EC2) 서비스로 시작하여 점점 더 다양한 서비스로 확장됨
- 현재는 수백 개의 클라우드 서비스 제공으로 클라우드 컴퓨팅의 선두주자 역할

## 배경

- Amazon이 자체 인프라스트럭처 최적화를 위해 시작
- 대규모 인프라 구축과 운영 경험을 바탕으로 클라우드 컴퓨팅 서비스로 개방



# 주요 서비스 개요

## 컴퓨팅 서비스

- EC2: 가상 서버 인스턴스를 제공하는 서비스
- Lambda: 서버리스 컴퓨팅으로 코드 실행만으로 인프라 관리 불필요

## 스토리지 서비스

- S3: 확장성 높은 객체 스토리지 서비스
- EBS: EC2 인스턴스에 연결하는 블록 스토리지 서비스

## 데이터베이스 서비스

- RDS: 관리형 관계형 데이터베이스 서비스
- DynamoDB: NoSQL 데이터베이스 서비스

## 네트워킹 서비스

- VPC: 사용자 지정 가상 네트워크 환경 제공
- CloudFront: 전 세계에 콘텐츠를 빠르게 배포하는 CDN 서비스



# 리전과 존의 이해

## 리전

- AWS 서비스를 제공하는 물리적 위치 단위로 각 리전은 독립적임

## 가용 영역(AZ)

- 하나의 리전 내에서 물리적으로 분리된 데이터 센터 그룹

## 엣지 로케이션

- 콘텐츠 배포를 위해 사용되는 위치로, CloudFront의 네트워크 엣지 캐시를 제공



# 클라우드 인프라 예

“ 클라우드 인프라의 배포 구성 단위는 Region > Zone > Pod > Cluster > Host ”  
(Cloud Stack 기준)

리전(Region) – 국가/대륙 단위, DR(Disaster Recovery) 관점

존(Zone) – 데이터 센터 단위, HA(High Availability) 관점

파드(Pod) – 랙 단위

Type the Return Keystroke

클러스터(Cluster) – 서버 묶음 단위

호스트  
(Host)

프라머리  
스토리지



존(Zone)

동일한 물리적 단위  
(하이퍼바이저, 서버 스펙...)

가용존  
(Availability Zone)



# AWS 계정 생성 및 관리

## 계정 생성 과정

- AWS 공식 웹사이트에서 회원 가입 후 계정 생성
- 신용카드 정보 및 본인 인증 필요

## IAM 사용자 및 역할 설정

- 보안을 위해 Root 계정 대신 IAM 사용자 생성
- 역할(Role)을 설정해 다양한 서비스 접근 권한 부여

## 멀티 팩터 인증(MFA)

- 보안 강화를 위해 MFA 설정 권장



# AWS 무료 티어 활용 방법

## 무료 티어

신규 사용자에게 12개월 동안 다양한 AWS 서비스 무료 제공

EC2, S3, RDS 등의 서비스 기본(?) 사용량 무료  
Type the Return Keystroke

## 활용 사례

실습 및 테스트 용도로 활용 가능

개인 프로젝트나 학습 목적으로 사용하여 클라우드 환경 경험



# 비용 관리와 예산 설정 방법

## 비용 관리 도구

- AWS Cost Explorer: 비용을 시각적으로 모니터링
- AWS Budgets: 예산을 설정하고 초과 시 알림 설정 가능

## 예산 설정

- 월별 또는 연간 예산을 설정해 클라우드 사용 비용 통제
- 서비스별 비용 분석을 통해 비용 절감 방안 모색

# AWS의 경쟁력: 대규모 운영의 효율성

AWS는 전 세계 26개 리전과 84개 가용존에 걸쳐 광범위한 인프라를 운영하여 높은 가용성, 확장성 및 성능을 제공함.

AWS INFRA은 어디서든 낮은 지연 시간과 안정적인 서비스 제공을 가능하게 함.  
Type the Return Keystroke

AWS는 탄력적인 리소스 프로비저닝을 통해 급증하는 트래픽에도 쉽게 대처할 수 있도록 지원함.

## 고객 이점

- 전 세계 어디에서든 사용자에게 일관된 경험을 제공할 수 있음.
- 비용 효율적인 방식으로 비즈니스를 확장할 수 있음.



# AWS의 주요 고객

스타트업부터 대기업까지 다양한 규모의 기업들이 AWS를 사용

- 금융 서비스: Capital One, Goldman Sachs, Bank of America
- 소매: Netflix, Airbnb, Amazon
- 제조: Boeing, Siemens, General Electric
- 의료: Johnson & Johnson, Novartis, Pfizer
- 정부: 미국 국방부, NASA, 국무부

## 고객 이점

- AWS는 다양한 산업 분야의 고객 요구 사항을 충족시킬 수 있는 입증된 플랫폼임.
- 업계 선두 기업들의 성공 사례를 참고하여 클라우드 도입의 효과를 확인할 수 있음.



# AWS의 적용 사례

## Netflix

- AWS를 사용하여 글로벌 스트리밍 서비스를 구축하고 수백만 명의 사용자에게 원활한 서비스를 제공함.
- <https://aws.amazon.com/solutions/case-studies/netflix-case-study/>

## Airbnb

### Type the Return Keystroke

- AWS를 사용하여 호스팅 플랫폼을 확장하고 전 세계 여행객들에게 안전하고 편리한 숙박 경험을 제공함.
- <https://aws.amazon.com/solutions/case-studies/innovators/airbnb/>

## NASA

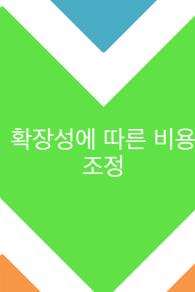
- AWS를 사용하여 과학 데이터를 분석하고 우주 탐사 임무를 지원함.
- <https://aws.amazon.com/solutions/case-studies/nasa-image-library/>



# AWS의 경제성: 비용 절감 효과



- AWS는 전통적인 데이터 센터에 비해 운영 비용을 크게 절감할 수 있는 인프라를 제공함. 이는 물리적 인프라의 구축 및 유지보수에 드는 비용을 줄이고, IT 자원을 보다 효율적으로 활용할 수 있게 함.



- AWS는 사용량 기반의 비용 모델을 제공하여, 고객의 비즈니스 규모와 요구에 따라 비용을 조정할 수 있음. 이는 불필요한 비용을 최소화하고, 비즈니스의 성장에 따라 서비스를 쉽게 확장할 수 있게 함.



- AWS를 통한 기술 투자는 높은 수익률을 제공함. 클라우드 기반 솔루션은 빠른 시간 내에 비즈니스 가치를 창출하며, 기업의 경쟁력을 향상시킴.

# AWS와 비즈니스 연속성: 재해 복구 및 고가용성

## 데이터 센터와 리전 간의 재해 복구

- AWS는 자연 재해나 시스템 장애가 발생했을 때 빠른 복구를 가능하게 함. 이는 비즈니스 연속성을 보장하며, 고객의 신뢰를 유지하는 데 중요함.

## 다중 리전 설정

- AWS는 다중 리전 설정을 통해 고객 데이터의 높은可用성과 보호를 제공함. 이는 데이터 손실의 위험을 최소화하고, 서비스 중단 시간을 줄임.

## 자동화된 백업 솔루션

- AWS는 자동화된 백업 솔루션을 제공하여, 데이터 손실 위험을 최소화함. 이는 비즈니스 데이터의 안전성을 보장하며, 복구 시간을 단축시킴.



# AWS의 보안 및 컴플라이언스 특징



## 포괄적인 보안 아키텍처

AWS는 데이터 보호를 위한 철저한 보안 정책 및 프로토콜을 제공함. 이는 고객 데이터의 안전성을 보장하며, 비즈니스의 신뢰성을 높임.



## 업계 표준 준수

AWS는 글로벌 보안 인증과 컴플라이언스 기준을 충족함. 이는 고객이 AWS를 안심하고 사용할 수 있게 함.



## 지속적인 보안 감사

AWS는 외부 감사를 통해 보안 상태를 지속적으로 검토하고 강화함. 이는 보안 위협에 대응하고, 시스템의 안전성을 유지하는 데 중요함.



# AWS의 전략적 파트너십 및 생태계



## 광범위한 파트너 네트워크

AWS는 전세계적인 파트너들과 협력하여, 고객에게 가장 적합한 솔루션을 제공함. 이는 고객의 비즈니스 요구에 맞는 최적의 서비스를 찾는데 도움이 됨.



## 개발자 및 혁신 생태계 지원

AWS는 개발자 리소스와 커뮤니티를 통해 지원을 제공함. 이는 고객이 AWS 서비스를 최대한 활용하고, 비즈니스를 혁신하는데 도움이 됨.



## 기술 통합

AWS는 다양한 업계 솔루션과 원활하게 통합되도록 지원함. 이는 고객이 자신의 비즈니스 환경에 AWS를 쉽게 통합할 수 있게 함.



# AWS 계정 생성 과정

## 기본 절차

- AWS 공식 웹사이트 접속 후 'Complete Sign up' 클릭.
- 이메일 주소 및 비밀번호 입력.
- 계정 유형 선택(개인 또는 비즈니스).

## 세부 절차

- 결제 정보 입력: 신용카드 또는 직불카드 정보.
- 전화 인증: 보안 확인을 위한 전화번호 인증.
- 지원 플랜 선택: 무료 기본 지원 또는 유료 지원 플랜 선택.

## 완료 확인

- 계정 생성 완료 이메일 수신.
- AWS 관리 콘솔 로그인 확인.

# AWS (Instance) 설정: Create Account

<https://aws.amazon.com/free>

The screenshot illustrates the process of creating an AWS account, divided into two main sections:

- Left Side (AWS Free Tier Landing Page):** Shows the "Create a Free Account" button highlighted with a red box. A red arrow points from this button to the "Create a new AWS account" button on the right side of the sign-in page.
- Right Side (Sign In Page):** Shows the "Create a new AWS account" button highlighted with a red box. A red arrow points from this button back to the "Create a Free Account" button on the left.

**Top Bar:** AWS Free Tier | https://aws.amazon.com/free/ | Complete Sign Up

**AWS Free Tier Page Content:**

- Header: AWS Free Tier, Overview, FAQs, Terms and Conditions
- Section: AWS Free Tier
- Text: Gain free, hands-on experience with the AWS platform, products, and services
- Button: Create a Free Account

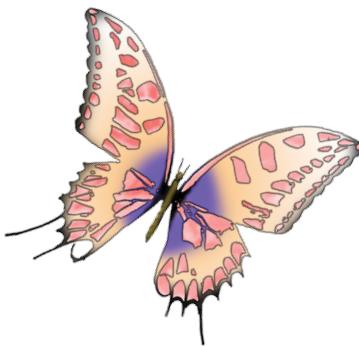
**Sign In Page Content:**

- Header: Sign in
- Text: Email address of your AWS account  
Or to sign in as an IAM user, enter your account ID or account alias instead.
- Form: Email input field
- Button: Next
- Text: New to AWS?
- Button: Create a new AWS account

**Annotations:**

- 1. AWS계정 없으신 분들은 먼저 만드세요.
- 2. AWS Free Tier계정을 만들어요.
- 신용카드 필요. 1\$ 결제 됩니다.

# 실습: AWS 콘솔에서 EC2 계정 생성



- 실습 목표

AWS계정 생성

- 준비

계정 등록시 사용할 Email

신용카드(해외 결제 가능)

- 실습 단계

<https://aws.amazon.com/> → Complete Sign up

A screenshot of a web browser displaying the AWS sign-up page. At the top, there is a navigation bar with links like 'About AWS', 'Contact Us', 'Support', 'English', 'My Account', and 'Sign In'. Below the navigation bar, there is a search bar and several other links: 'Partner Network', 'AWS Marketplace', 'Customer Enablement', 'Events', 'Explore More', and a magnifying glass icon. A large red rectangular box highlights the 'Complete Sign Up' button, which is located at the bottom right of the page. The rest of the page contains form fields for email, password, and other account details.





# 실습: 서비스 탐색



- 실습 목표

AWS 관리 콘솔에서 다양한 서비스 탐색

- 실습 단계

콘솔 로그인 후 주요 서비스(EC2, S3, RDS 등) 탐색

각 서비스의 기본 정보와 기능 확인

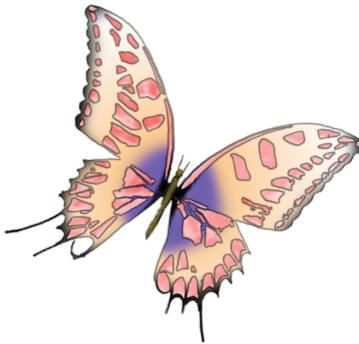
- 결과 확인

각 서비스의 용도와 사용법 간략히 이해





# 실습: 리전 확인과 이동



- 실습 목표

AWS 콘솔에서 리전을 변경하고 서비스 가용성 확인

- 실습 단계

콘솔 우측 상단에서 리전 선택 후 변경

리전별 서비스 가용 여부 탐색

- 결과 확인

리전 변경에 따른 서비스 목록 변화 확인





# **EC2** **(Elastic Compute Cloud)**



Amazon  
**EC2**

<https://aws.amazon.com/ko/ec2/>



# AWS EC2: Elastic Compute Cloud의 기초

## 주요 개념

가상 서버를 제공하는 AWS의 핵심 컴퓨팅 서비스.

다양한 용도의 컴퓨팅 파워 제공.

## 특징

다양한 인스턴스 유형: 사용 목적에 맞는 선택 가능.

Auto Scaling: 트래픽 변화에 따른 자동 확장/축소.

보안 그룹: 인바운드 및 아웃바운드 트래픽 제어.

## 사용 사례

웹 애플리케이션 호스팅.

데이터 분석 작업.

개발 및 테스트 환경 구축.



# EC2 인스턴스 유형 설명

## 인스턴스 패밀리

- **범용(GP, General Purpose)**: 균형 잡힌 CPU와 메모리 (ex. t4g, m5)
- **컴퓨팅 최적화(Compute Optimized)**: 높은 성능의 CPU 제공 (ex. c5)
- **메모리 최적화(Memory Optimized)**: 메모리 집약적 워크로드에 적합 (ex. r5)
- **스토리지 최적화(Storage Optimized)**: 고속 로컬 스토리지 필요 시 (ex. i3)

## 스팟 인스턴스, 예약 인스턴스

- 비용을 절감하기 위해 스팟 인스턴스 또는 예약 인스턴스 사용 가능



# EC2 보안 및 네트워크 구성 옵션

## 보안 그룹 설정

- 인바운드 및 아웃바운드 트래픽 규칙 설정.
- IP 주소 및 포트 기반 접근 제어.

## 네트워크 ACL 구성

- 서브넷 레벨에서의 트래픽 제어.
- 상태 비저장 필터링 규칙 적용.

## 키 페어 사용

- SSH 액세스를 위한 퍼블릭/프라이빗 키 페어 생성.
- 인스턴스 접근 보안 강화.

## VPC 설정

- 프라이빗 및 퍼블릭 서브넷 구성.
- 인터넷 게이트웨이 및 NAT 게이트웨이 설정.



# AMI(Amazon Machine Image) 이해하기

## AMI란?

- EC2 인스턴스를 생성하기 위한 템플릿 역할을 하는 이미지
- 운영체제, 애플리케이션, 설정 등이 포함된 상태로 인스턴스 생성 가능

## 사용자 정의 AMI

- 기존 인스턴스를 커스터마이징하여 새로운 AMI로 저장 가능
- 동일한 설정으로 여러 인스턴스 생성 시 유용



# 인스턴스 시작 및 중지 과정

## 인스턴스 생성 과정

AMI 선택 후 인스턴스  
유형, 키 페어, 보안 그  
룹 등 설정

시작 후 퍼블릭 IP 할  
당 및 접속 가능

## 인스턴스 중지 및 종료

중지(Stop): 인스턴스  
를 종료하지만 데이터  
는 유지

종료(Terminate): 인  
스턴스와 모든 데이터  
삭제



# 보안 그룹 및 네트워크 구성

## 보안 그룹(Security Group)

- 인스턴스에 대한 트래픽을 제어하는 방화벽 역할
- 인바운드/아웃바운드 규칙 설정을 통해 접근 제어

## 네트워크 인터페이스 및 VPC

- VPC 내에서 서브넷과 라우팅을 통해 네트워크 구성
- EC2는 특정 서브넷에 연결되어 통신 가능



# EC2 비용 절감 방법 [예약 인스턴스, 스팟 인스턴스]

## 예약 인스턴스

- 장기 사용 시 할인된 요금으로 인스턴스 예약 가능 (1년, 3년 단위)
- 안정적인 워크로드에 적합

## 스팟 인스턴스

- AWS의 여유 컴퓨팅 자원을 활용해 저렴하게 사용 가능
- 중단 가능성이 있어 일시적인 작업이나 분산 처리에 적합



# EC2 확장성 (Auto Scaling)

## Auto Scaling

- 수요에 따라 인스턴스를 자동으로 추가하거나 제거하여 비용 효율성 극 대화
- 최소/최대 인스턴스 수와 조건을 설정하여 유연하게 확장 가능

## 로드 밸런서와의 연계

- ELB(Elastic Load Balancer)와 연계하여 트래픽을 여러 인스턴스로 분산

# AWS 비용 관리 및 예산 설정



## 비용 탐색기 사용

월별, 일별, 서비스별 비용 분석.  
필터를 사용한 상세 비용 조회.



## 예산 설정

예산 유형 선택: 비용, 사용량,  
RI(Reserved Instance) 예산.  
알림 임계값 설정: 예산 초과 시 알  
림 수신.



## 비용 절감 전략

예약 인스턴스 및 스팟 인스턴스 활용.  
비용 분석을 통한 불필요한 리소스 식  
별 및 제거.



## 청구서 검토

청구서 세부 항목 확인.  
서비스별 비용 분석을 통한 최적화  
기회 식별.

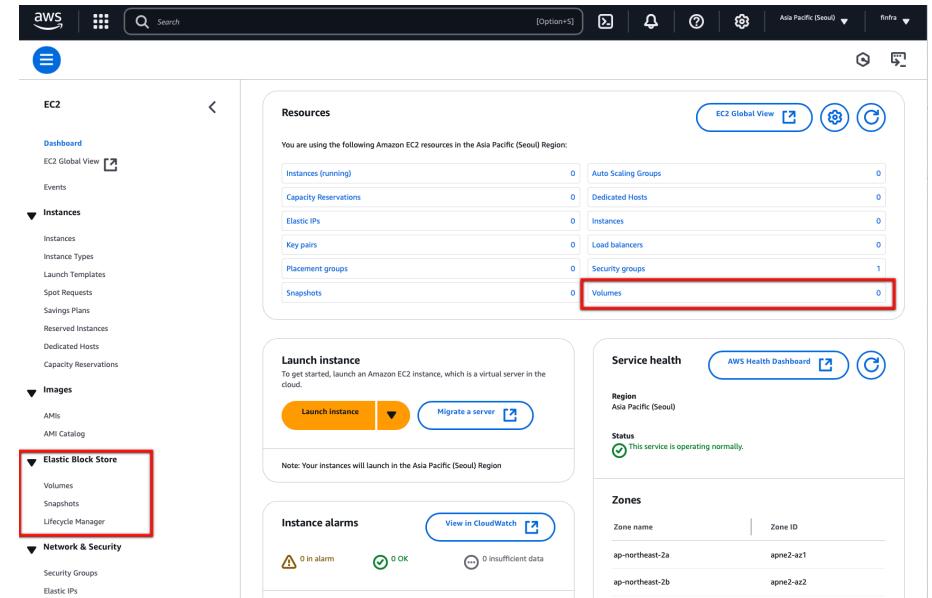


# EIP(Elastic IP address)

- 고정된 공인 IPv4 주소! 요금 발생
- 특징
  - 계정에 귀속
  - 인스턴스 종료/시작과 무관하게 유지됨
- 사용 가능 대상 : EC2 인스턴스 또는 NAT Gateway 등
- 기본 제공 여부 : 기본은 동적 공인 IP(무료)이고, EIP는 별도로 할당해야 함

# EBS(Amazon Elastic Block Store)

- EC2의 하드 디스크 !
- 정의 : Amazon EC2 인스턴스에 연결 가능한 고성능 블록 스토리지 서비스
- 특징 : 내구성 보장, 크기 조정 가능, 스냅샷 지원, 다양한 성능 옵션 제공
- 사용 방식 : EC2 인스턴스에 디스크 드라이브처럼 마운트하여 사용함
  - 루트 디바이스 또는 추가 데이터 볼륨



(EBS 챕터 참고)



# 모니터링과 로깅 (CloudWatch)



- EC2 인스턴스의 상태, CPU 사용률, 네트워크 트래픽 등을 모니터링하는 서비스
  - 알람 설정을 통해 임계치 초과 시 알림 발송
- 
- CloudWatch Logs를 통해 애플리케이션 로그와 시스템 로그 수집 및 분석 가능



# 모니터링과 로깅 (CloudWatch)

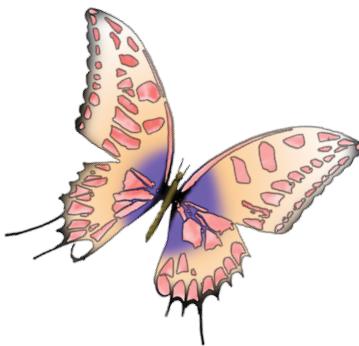
## CloudWatch

## 로그 관리

EC2 인스턴스의 상태, CPU 사용률, 네트워크 트래픽 등을 모니터링하는 서비스알람 설정을 통해 임계치 초과 시 알림 발송

CloudWatch Logs를 통해 애플리케이션 로그와 시스템 로그 수집 및 분석 가능

# 실습: 수동으로 EC2 인스턴스 생성



- 실습 목표

AWS 콘솔을 사용하여 EC2 인스턴스를 수동으로 생성하고 설정

- 실습 단계

AWS 콘솔에 접속 후 EC2 서비스 선택

AMI, 인스턴스 유형, 보안 그룹 등을 설정하고 인스턴스 생성

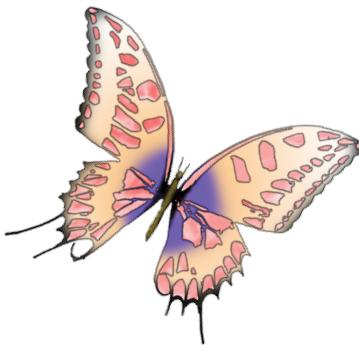
- 결과 확인

생성된 인스턴스에 퍼블릭 IP로 접속 확인





# 실습: EC2 보안 그룹 설정 및 연결



- 실습 목표

보안 그룹을 생성하고 인스턴스에 연결하여 접근 제어 설정

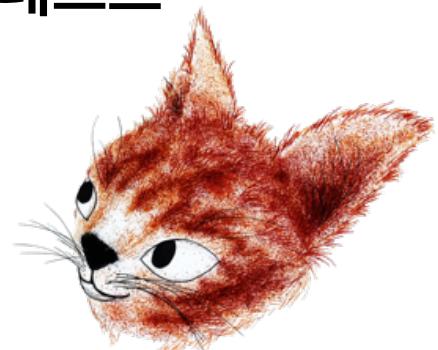
- 실습 단계

새 보안 그룹 생성 및 인바운드 규칙 추가 (ex. SSH, HTTP)

기존 EC2 인스턴스에 보안 그룹 연결

- 결과 확인

보안 그룹 설정에 따라 접근 가능 여부 테스트





# 실습: SSH Client 연결



- 실습 목표

SSH 클라이언트를 사용해 EC2 인스턴스에 접속

- 실습 단계

로컬 머신에서 SSH 키 페어를 사용해 인스턴스에 접속

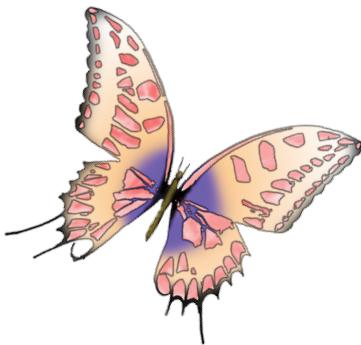
퍼블릭 IP 또는 퍼블릭 DNS를 통해 연결 시도

- 결과 확인

정상적으로 EC2 인스턴스에 접속되어 명령어 실행 가능



# 실습: EC2 인스턴스 상태 모니터링



- 실습 목표

EC2 인스턴스의 상태를 모니터링하고 성능 확인

- 실습 단계

AWS 콘솔에서 EC2 대시보드 접근

인스턴스 선택 후 ‘모니터링’ 탭 클릭

주요 지표(CPU 사용률, 네트워크 트래픽 등) 확인

CloudWatch를 통한 경보 설정 및 알림 구성





# 실습: EC2 인스턴스 중지 및 재시작 처리



- 실습 목표

EC2 인스턴스를 안전하게 중지하고 재시작하는 방법 이해

- 실습 단계

AWS 콘솔에서 인스턴스 선택

‘인스턴스 상태’에서 ‘중지’ 선택

중지 상태 확인 후 ‘인스턴스 상태’에서 ‘시작’ 선택

인스턴스의 상태 변화 확인





# 실습: EIP 생성



- 실습 목표

Elastic IP를 생성하고 고정 IP 주소 관리 이해

- 실습 단계

AWS 콘솔에서 EC2 대시보드 접근

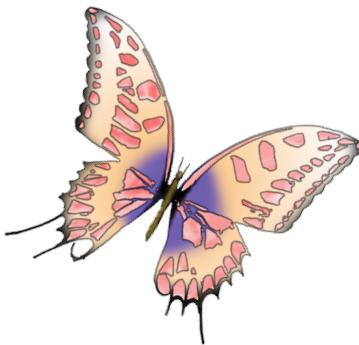
‘네트워크 및 보안’에서 ‘Elastic IP’ 선택

‘Elastic IP 주소 할당’ 클릭

VPC 선택 및 EIP 생성 완료



# 실습: EC2 인스턴스에 EIP 연결



- 실습 목표

Elastic IP를 EC2 인스턴스에 연결하여 고정 IP 설정

- 실습 단계

AWS 콘솔에서 ‘Elastic IP’ 선택

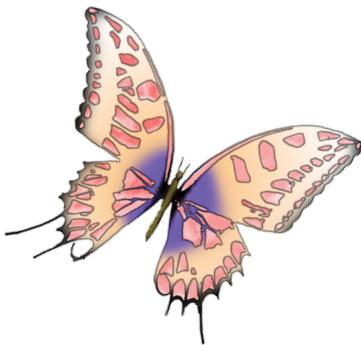
생성된 EIP 선택 후 ‘작업’ 클릭

‘인스턴스에 연결’ 선택 후 대상 인스턴스 지정

연결 상태 확인 및 IP 변경 사항 적용 확인



# 실습: EC2 인스턴스 종료 및 삭제



- 실습 목표

EC2 인스턴스를 안전하게 종료 및 삭제

- 실습 단계

AWS 콘솔에서 인스턴스 선택

‘인스턴스 상태’에서 ‘종료’ 선택

종료 상태 확인 후 인스턴스가 삭제되는지 확인

필요에 따라 연결된 리소스(EIP, 볼륨)도 삭제





# IAM (Identity and Access Management)



<https://aws.amazon.com/ko/iam/>



# IAM 개요 및 중요성

## IAM이란?

- AWS 리소스에 대한 접근을 제어하는 서비스로, 사용자의 권한을 관리
- AWS 계정을 안전하게 보호하고, 최소 권한 원칙을 적용하여 보안 강화

## 중요성

- 모든 AWS 리소스에 대한 접근 제어의 핵심 역할을 수행
- 사용자마다 필요에 맞는 권한을 설정하여 불필요한 권한 남용 방지



# 사용자와 그룹 생성 및 관리

사용자(User)

- 개별 IAM 사용자 계정을 생성해 각 사용자에게 특정 권한 부여 가능
- 프로그램matic 접근과 AWS Management Console 접근을 제공

그룹(Group)

- 동일한 권한을 필요로 하는 사용자들을 그룹으로 묶어 효율적으로 관리
- 권한을 그룹에 부여함으로써 사용자 추가 시 자동으로 권한 부여



# IAM 정책과 권한 부여 이해하기

## 정책 (Policy)

- JSON 형식으로 작성된 문서로, AWS 리소스에 대한 접근 권한 정의
- 사용자, 그룹, 역할에 연결하여 권한을 설정함

## 정책 유형

- 관리형 정책: AWS에서 제공하는 표준 정책
- 고객 관리 정책: 사용자가 직접 정의한 커스텀 정책



# MFA 설정 및 보안 관리

## 멀티 팩터 인증(MFA)

- AWS 계정에 대한 이중 보안 설정
- 계정 접근 시 비밀번호 외에 추가 인증이 필요하여 보안 성 강화

## 보안 관리 모범 사례

- 사례
  - Root 계정에 MFA 필수 설정
  - IAM 사용자에게 최소 권한 원칙 적용



# 역할(Role)과 권한 위임

## 역할(Role)

- 특정 AWS 리소스를 사용하게 하는 권한을 부여할 수 있는 엔터티
- 한 사용자가 아닌 여러 AWS 서비스 또는 애플리케이션에 권한 위임 가능

## 권한 위임

- Cross-account 접근을 허용해 여러 AWS 계정 간에 자원 공유 가능



# IAM 인증 토큰과 API 접근

## IAM 인증 토큰

- API 호출 시 사용자 인증을 위해 사용되는 임시 인증 정보
- AWS STS(Security Token Service)를 통해 발급됨

## API 접근 제어

- 각 IAM 사용자에게 API 호출을 허용하거나 제한하는 정책 설정 가능
- 서비스별 API 접근 권한을 세분화하여 부여



# 키 관리 및 로테이션

## 액세스 키 관리

- 프로그램matic 접근을 위해 생성된 액세스 키를 주기적으로 회전하며 보안 유지

## 키 로테이션

- 오래된 키를 주기적으로 삭제하고 새로운 키 생성하여 보안 강화



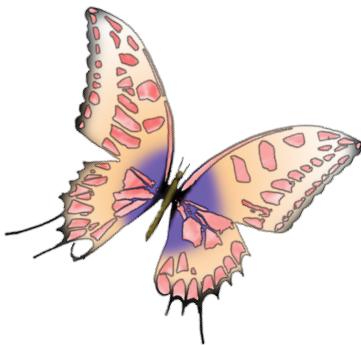
# AWS (Instance) 설정: IAM Setting

The screenshot shows the AWS IAM 'Add user' interface. At the top, there's a navigation bar with the AWS logo, 'Services', 'Resource Groups', a search bar, and account information ('nowageFree', 'Global', 'Support'). Below the navigation, the title 'Add user' is displayed, along with three numbered steps (1, 2, 3). Step 2 is currently selected. The main area is titled 'Set permissions' and contains three options: 'Add user to group' (selected), 'Copy permissions from existing user', and 'Attach existing policies directly' (highlighted with a red box). Below these options, there's a section titled 'Get started with groups' with a note about creating groups for managing permissions. A 'Create group' button is also present.

- 생성한 유저의 secret\_key와 access\_key를 잘 저장해 둘것.
- SystemAdministrator 권한 필요



# 실습: Terraform을 위한 IAM 사용자 생성



- 실습 목표

Terraform으로 AWS 리소스를 관리하기 위한 IAM 사용자 생성

- 실습 단계

AWS 콘솔에서 IAM 사용자 생성 및 프로그램matic 접근 설정

사용자에게 Terraform을 사용할 수 있는 권한 부여

- 결과 확인

사용자 액세스 키 생성 후 Terraform에서 인증 확인





# 실습: Role 생성 및 적용



- 실습 목표

특정 서비스에 역할(Role)을 생성하고 적용하는 방법 이해

- 실습 단계

IAM 콘솔에서 새로운 역할 생성 (ex. EC2용 역할)

EC2 인스턴스에 역할을 연결하여 S3 접근 권한 부여

- 결과 확인

EC2에서 S3 접근 시도하여 역할 작동 여부 확인



# 실습: MFA 설정 및 키 생성



- 실습 목표

IAM 사용자에 대한 MFA 설정과 액세스 키 생성 이해

- 실습 단계

IAM 사용자에 MFA 설정 추가

인증 앱을 통해 MFA 설정 완료

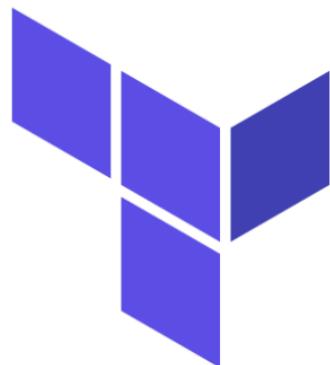
- 결과 확인

MFA 설정 후 로그인 과정에서 MFA 인증 요구 확인





# Terraform (인프라스트럭처 코드 관리 도구)



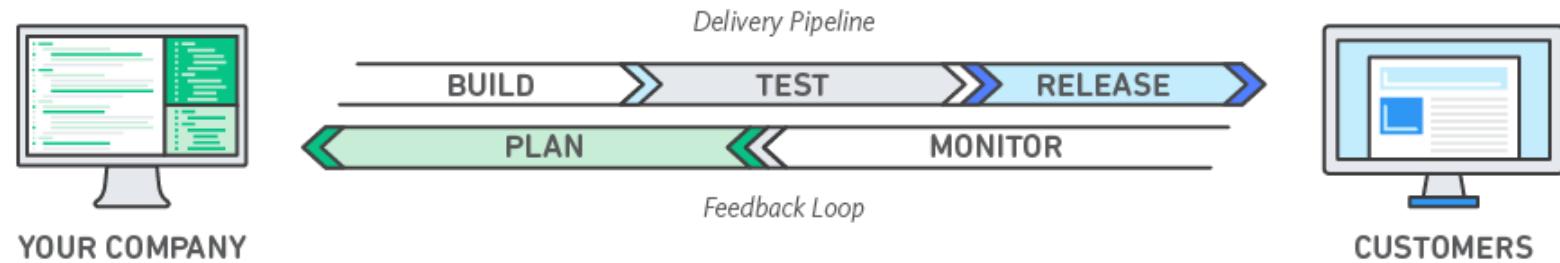
<https://www.terraform.io/>

HashiCorp  
**Terraform**



# DevOps란?

- 데브옵스는 애플리케이션과 서비스를 빠른 속도로 제공할 수 있도록 조직의 역량을 향상시키는 문화 철학, 방식 및 도구의 조합



- 데브옵스 방식(모범사례)
  - CICD : 지속적 통합과 지속적 전달 (Jenkins)
  - MSA : 마이크로 서비스 아키텍쳐(Docker&Kubernetes or SpringBoot)
  - Infrastructure as Code : ex) 코드형 인프라스트럭처(Terraform)
  - 모니터링 및 로깅(Prometheus)
  - 커뮤니케이션 및 협업 ( Git )

\* 읽어보면 좋은 자료 : DevOps!! 도데체 왜,  
어떻게 할까?? <https://www.slideshare.net/chingu94/devops-62712241>



# Terraform 개요와 기능

## Terraform이란?

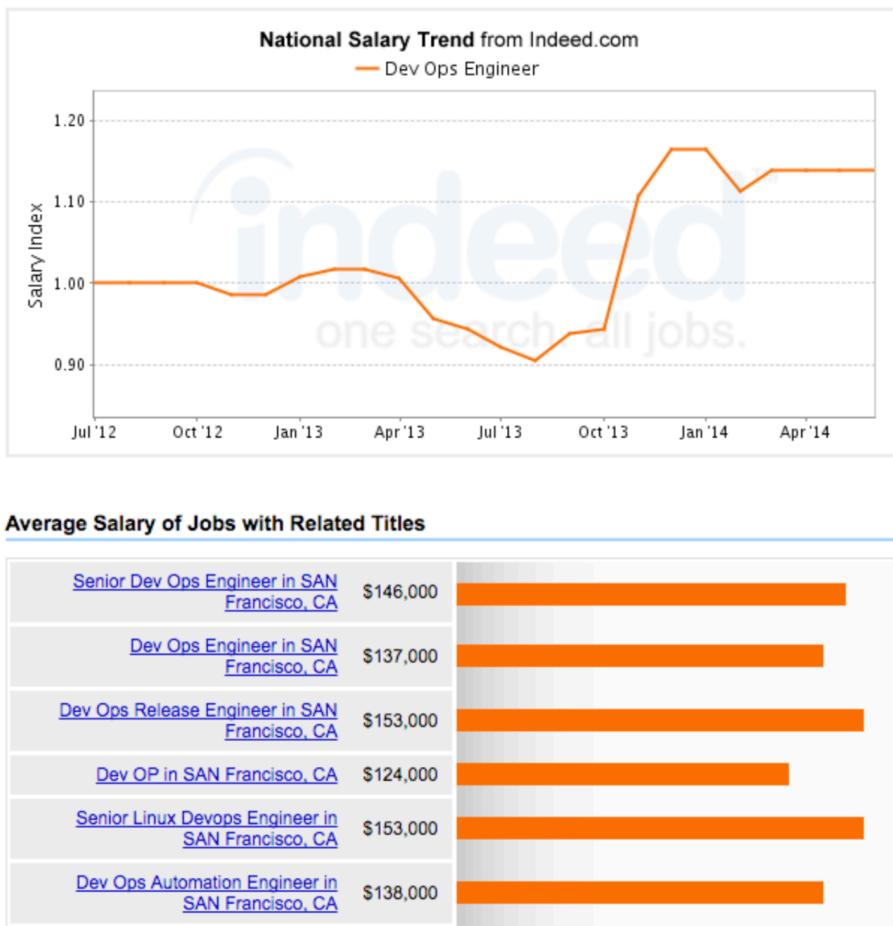
- 인프라스트럭처를 코드로 관리할 수 있는 오픈 소스 도구
- 클라우드 환경의 리소스를 코드 기반으로 생성, 수정, 삭제 가능

## 주요 기능

- 프로비저닝:** 클라우드 리소스를 자동으로 설정하고 배포
- 상태 파일 관리:** 현재 인프라의 상태를 파일로 관리하여 추적 가능



# Why Terraform



- DevOps / Automation 엔지니어의 평균 급여는 San Francisco에서 최대 \$ 146,000
- Terraform은 지난 몇 년 동안 많은 인기
- 이제 이 분야에서 가장 많이 찾는 기술 중 하나
- 클라우드 환경을 프로비저닝 할 때 숙달 해야하는 도구



# Terraform 설치 및 설정

## 설치 과정

- Terraform 공식 사이트에서 다운로드 및 설치
- 환경 변수 설정을 통해 CLI에서 Terraform 명령어 사용 가능

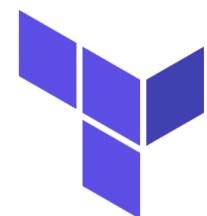
## 설정파일 생성

- Terraform 설정 파일(.tf)을 통해 리소스 정의 및 설정



# IaC (Infrastructure as Code)

- IaC란 시스템을 수동으로 관리하는 대신 스크립트를 사용하여 컴퓨팅 인프라를 구성하는 방식이다.
- 대표적으로 hashcorp에서 오픈소스로도 제공되는 terraform, AWS의 Cloud formation등 이 있다.
- cf) 소프트웨어 기반 관리 툴 : ansible, chef, puppet



HashiCorp  
**Terraform**



# Terraform이란?

- 테라폼(Terraform)은 Hashicorp에서 오픈소스로 개발한 인프라스트럭처 관리 도구
  - 안전하고 효율적인 인프라 구축, 변경 및 버전 관리 도구
  - Terraform은 기존의 인기있는 서비스 제공 업체와 맞춤형 사내 솔루션을 관리
- 즉, 코드로 IT Infra 구성
- 예, Terraform코드 실행하여 AWS Ec2 및 VPC구성



# Terraform이 좋은 점

- 코드에서 인프라를 자동화 할 수 있습니다
- 클라우드 인프라를 프로비저닝합니다. 예) AWS
- 인프라 변경 내역을 유지할 수 있습니다
- 팀 내에서 협업하여 인프라 구축  
→ SRE를 구현하기 위해서 필요



# Terraform

- DevOps 역할을 수행하는 경우와 인프라를 자동화 하려는 경우
- Terraform은 이를 위한 훌륭한 도구
  - 코드에서 인프라를 자동화 가능(IaC)
  - 클라우드 인프라를 프로비저닝 함, Ex: AWS, GCP
  - 인프라 변경 내역을 유지할 수 있습니다
  - 팀 내에서 협업하여 인프라 구축



# Terraform

- Infrastructure As Code
- "Automation of your infrastructure"
- 인프라를 특정 상태 (호환)로 유지
  - 예 : 2 개의 볼륨이있는 2 개의 웹 인스턴스와 1 개의로드 밸런서.
- 인프라 감사 가능
  - GIT와 같은 버전 제어 시스템에서 인프라 변경 기록을 유지



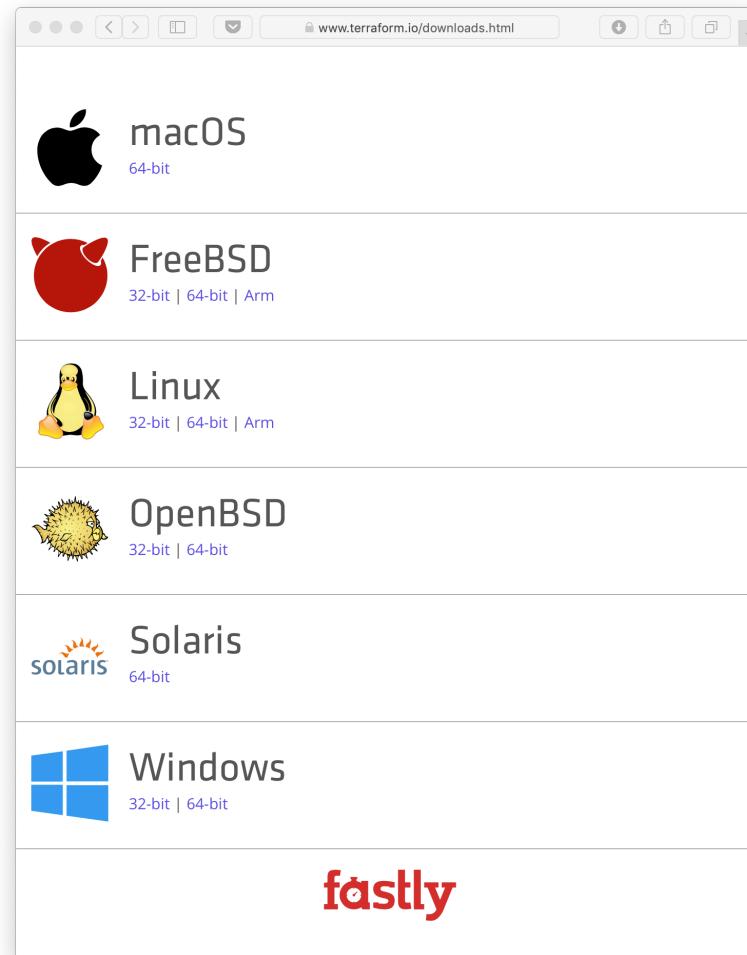
# Terraform

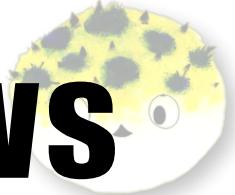
- Ansible, Chef, Puppet, Saltstack은 소프트웨어 설치 및 구성 자동화에 중점을 두고 만들어짐(운영용).
  - 특정 상태에서 Machine을 지정된 상태로 유지
- Terraform은 인프라 자체의 프로비저닝을 자동화.
  - 예. AWS, DigitalOcean, Azure API 사용
  - Terraform이 프로비저닝하고 난 후 Ansible과 같은 자동화 소프트웨어가 추가 설치할 때도 잘 작동



# Terraform 설치 - Download

- Download URL: <https://www.terraform.io/downloads.html>
- 지원 OS :
  - macOS
  - FreeBSD
  - Linux
  - OpenBSD
  - Solaris
  - Windows





# Terraform 설치 - Windows

- Path 추가하고 실행하면 됨
- Putty-gen 사용 키 생성할 것
  - 1. 메뉴 : Key → Generate key pair → (마우스 이동)
  - 2. Save Public key 버튼 눌러서 저장
  - 3. 메뉴 : Conversions → Export OpenSSH key
  - 4. Save Private키 버튼 눌러서 저장
- Putty 세션에 키 등록
  - 1. Connection → SSH → Auth → Browse...
  - 2. KeyGen으로 생성한 ppk파일 선택
  - 3. Session저장 (ip는 나중에 지정)



# Terraform 설치 예 - Linux

```
wget https://releases.hashicorp.com/terraform/1.8.3/terraform_1.8.3_darwin_amd64.zip
```

```
unzip terraform_1.8.3_darwin_amd64.zip
```

```
mkdir ~/bin
```

```
mv terraform ~/bin
```

```
x=`echo $PATH|grep ~/bin`
```

```
if [ ${#x} -eq 0 ];then
```

```
echo "PATH=$$PATH:~/bin" >>~/.profile
```

```
./.profile
```

```
fi
```

# 적당한 패쓰에 복사해서 쓰세요.

apple ~ \$ terraform version  
Terraform v1.8.3



# Provider와 설정 파일 이해하기

## ● Provider

- 클라우드 서비스(AWS, Azure, GCP 등)에 접근하기 위한 플러그인 역할
- `provider "aws"` 구문을 사용해 AWS와의 연동 설정

## ● 설정 파일

- `.tf` 파일을 통해 리소스, 변수, 모듈 등을 정의
- 코드 기반으로 인프라 설정을 자동화하여 관리 가능



# 리소스 정의와 관리 방법

- 리소스(Resource)

- AWS의 EC2, S3 등 클라우드 리소스를 Terraform 코드로 정의
  - `resource "aws_instance"` 등으로 리소스를 선언하고 필요한 설정 지정

- 관리 방법

- Terraform 명령어(`apply`, `destroy` 등)를 통해 정의된 리소스를 생성, 삭제

주의 : 예제임. 실제 셋팅은 `~/.bashrc`에서 환경 변수로 설정합니다.

# AWS (Instance) 설정: terraform으로 인스턴스 생성

`git clone https://github.com/finfra/awsTerraform-course`

`cd awsTerraform-course/01-FirstSteps`

`vi instance.tf`

`terraform init`

`terraform plan`

`terraform apply`

```
provider "aws" {
  access_key = "AKIA4RDJMLVWBT5WZV7C"
  secret_key = "UiMHjt+ZqTE/bLI6Oc0YC63GPCvEGMg+gduDyxU7"
  region     = "eu-west-1"
}

resource "aws_instance" "example" {
  ami        = "ami-0dad359ff462124ca"
  instance_type = "t2.micro"
}
```

key는 유저 생성시 저장해 둔 것 사용

ami는 <http://cloud-images.ubuntu.com/locator/ec2>에서 검색

\* 플러그인 재사용

`cp -r .terraform/ ~/`

```
01-FirstSteps (master)$ terraform apply -auto-approve
aws_instance.example: Creating...
aws_instance.example: Still creating... [10s elapsed]
aws_instance.example: Still creating... [20s elapsed]
aws_instance.example: Still creating... [30s elapsed]
aws_instance.example: Creation complete after 37s [id=i-0003b7f1307159fb3]

Apply complete! Resources: 1 added, 0 changed, 0 destroyed.
```



# Terraform Variables이 필요한 이유

- terraform file 하나로 작업 것은 좋지 않음.
- 보안 정보를 숨기기 위해 필요
  - git 저장소에 secret\_key가 저장되지 않게
- 경우에 따라 변하는 정보도 있음.
  - ex) AMIs는 region마다 다름
- terraform file의 재사용성도 높임.



# Terraform 상태 관리 및 버전 관리

- 상태 파일(State File)
  - 현재 인프라의 상태를 저장하는 파일 (`terraform.tfstate`)
  - 협업 시 상태 파일을 원격 저장소에 저장하여 동기화 필요
- 버전 관리
  - 각 리소스 정의의 변경 사항을 Git 등의 도구를 통해 관리
  - 코드 변경 이력과 인프라 변경 이력을 함께 추적 가능



# 변수와 출력 값 설정

- **변수(Variables)**
  - 인프라 설정 시 동적 값을 사용하기 위해 변수 선언 (`variable "name"`)
  - 여러 환경에서 동일한 코드를 사용할 수 있도록 재사용성 향상
- **출력 값(Outputs)**
  - 리소스 생성 후 중요한 정보를 출력하여 이후 과정에 활용 가능 (`output "ip"`)



# 모듈과 코드 재사용성 향상

- 모듈(Module)
  - 반복적으로 사용되는 리소스를 모듈화하여 코드 재사용성 증가
  - `module` 구문을 통해 여러 번 사용 가능
- 모듈 활용 사례
  - VPC, 네트워크 설정 등 여러 프로젝트에서 공통으로 사용되는 설정을 모듈로 정의



# Terraform 계획 및 적용 과정

- 계획(Plan)
  - `terraform plan` 명령을 사용해 코드 변경 사항을 미리 검토
  - 실제 적용 전에 예상되는 변경 사항을 확인하여 오류 방지
- 적용(Apply)
  - `terraform apply` 명령으로 코드에 정의된 리소스를 생성 또는 수정
  - 승인 후 인프라 생성 또는 업데이트 진행



# 리소스 삭제와 변경 관리

- 리소스 삭제
  - 코드에서 리소스를 제거한 후 `terraform apply`를 실행하여 삭제
  - 특정 리소스만 삭제할 때는 `terraform destroy -target` 사용
- 변경 관리
  - 코드 수정 후 `plan` 명령으로 변경 사항 확인
  - 기존 리소스를 안전하게 업데이트하거나 교체



# AWS [Instance] 설정: instance 삭제

`terraform plan -out /tmp/p.out`

`terraform apply out.terraform`

`terraform destroy`

The screenshot shows the AWS EC2 Dashboard with the 'Resources' tab selected. A modal window is open, displaying the output of a `terraform destroy` command. The command and its output are as follows:

```
01-FirstSteps (master)$ terraform destroy -auto-approve
aws_instance.example: Refreshing state... [id=i-0003b7f1307159fb3]
aws_instance.example: Destroying... [id=i-0003b7f1307159fb3]
aws_instance.example: Still destroying... [id=i-0003b7f1307159fb3, 10s elapsed]
aws_instance.example: Still destroying... [id=i-0003b7f1307159fb3, 20s elapsed]
aws_instance.example: Still destroying... [id=i-0003b7f1307159fb3, 30s elapsed]
aws_instance.example: Destruction complete after 37s

Destroy complete! Resources: 1 destroyed.
```



# Software Provisioning : key 적용

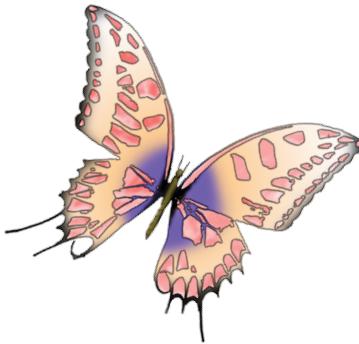
- 원격에서 Provisioning된 OS에 원격 로그인이 가능하려면 키파일 생성이 필요함.
  - 비번 방식 로그인은 권장되지 않음.
- key 생성

```
$ ssh-keygen -f ~/mykey
```

- VPC에서 Security Groups에 작업 Terraform 작동하는 OS의 IP 등록
  1. AWS → Services → VPC → Security Groups
  2. default 선택 → Edit Rules → Add Rule
  3. Source== MyIP → Save rules



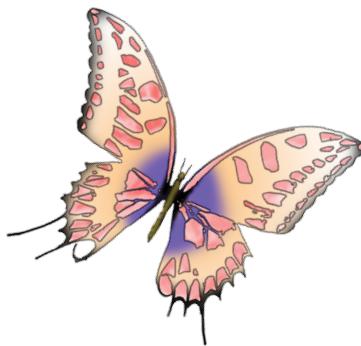
# 실습: Terraform 콘솔 설치



- 실습 목표
  - Terraform을 사용실행되는 콘솔 설치
- 실습 단계
  - 수동으로 인스턴스 생성
  - Terraform설치
- 결과 확인
  - "terraform -version"



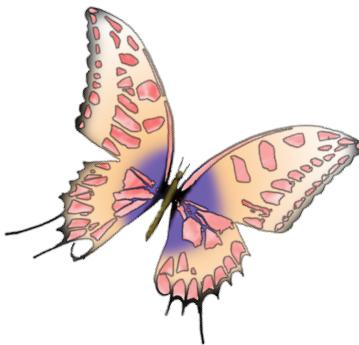
# 실습: Terraform으로 EC2 인스턴스 생성



- 실습 목표
  - Terraform을 사용하여 EC2 인스턴스를 자동으로 생성
- 실습 단계
  - AWS Provider 설정 및 EC2 리소스 정의
  - `terraform apply` 명령을 통해 인스턴스 생성
- 결과 확인
  - EC2 인스턴스가 AWS 콘솔에 생성되었는지 확인



# 실습: Terraform 상태 파일 관리 및 백업



- 실습 목표
  - Terraform 상태 파일을 안전하게 관리하고 백업
- 실습 단계
  - 상태 파일을 원격 저장소(S3 등)에 저장하여 협업 환경 구성
  - `terraform state` 명령으로 상태 파일 확인 및 관리
- 결과 확인
  - 원격 저장소에서 상태 파일 확인 및 백업 여부 점검





# **EBS (Elastic Block Store)**



Amazon **EBS**

<https://aws.amazon.com/ko/ebs/>



# EBS 개요와 작동 방식

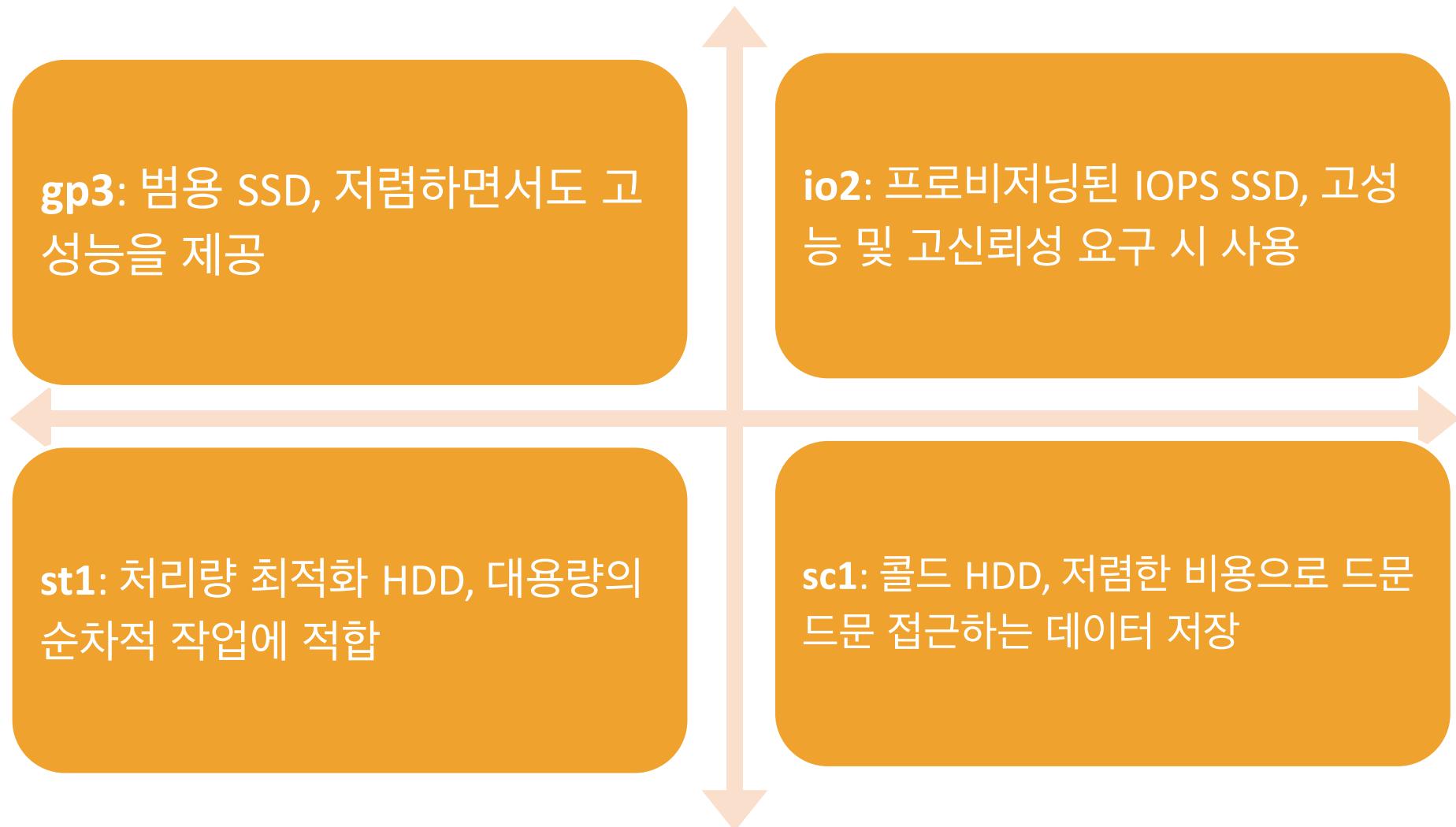
## EBS란?

- AWS에서 제공하는 블록 스토리지 서비스로, EC2 인스턴스에 연결하여 사용
- 지속적인 데이터 저장이 필요한 애플리케이션에 적합

## 작동 방식

- EC2 인스턴스에 연결된 상태에서 데이터를 블록 단위로 저장
- 인스턴스 중지/시작에도 데이터 유지

# EBS 볼륨 유형 (gp3, io2 등) 설명





# 스냅샷 및 백업 전략

## 스냅샷

- EBS 볼륨의 현재 상태를 백업하여 저장하는 기능
- 스냅샷을 이용해 새로운 볼륨 생성 또는 기존 데이터 복구 가능

## 백업 전략

- 정기적인 스냅샷 생성으로 데이터 보호
- AWS Backup 서비스를 사용해 백업 정책 자동화



# 볼륨 크기와 성능 조정 방법

## 크기 조정

- 필요에 따라 EBS 볼륨의 크기를 확장 가능
- 기존 데이터를 유지하면서 용량을 증가시킬 수 있음

## 성능 조정

- IOPS를 프로비저닝하여 성능 최적화 가능 (ex. io2 볼륨)
- 성능이 중요한 애플리케이션에 적합하게 설정



# EBS와 EC2 인스턴스 연결

볼륨 연결

- EC2 인스턴스 생성 시 기본 볼륨 연결 또는 추가 볼륨 연결 가능
- attach volume 명령으로 추가 볼륨을 인스턴스에 연결

데이터  
유지

- 연결된 EBS 볼륨은 인스턴스가 중지되더라도 데이터 유지 가능
- 인스턴스 종료 시 볼륨을 분리하거나 삭제하여 데이터 관리 가능



# 볼륨 마운트와 파일 시스템 설정

## 볼륨 마운트

- EC2 인스턴스에 연결된 EBS 볼륨을 특정 경로에 마운트하여 사용
- Linux 기반 인스턴스에서 mount 명령을 사용해 디스크를 파일 시스템에 연결



## 파일 시스템 생성

- 새로 연결된 볼륨에 파일 시스템 생성 (mkfs 명령 사용)
- 필요에 따라 ext4, xfs 등 파일 시스템 유형을 선택 가능



# 데이터 암호화 및 보안 설정

- 데이터 암호화
  - EBS 볼륨 생성 시 암호화를 활성화하여 데이터 보호
  - AWS KMS(Key Management Service)를 이용해 암호화 키 관리 가능
- 보안 설정
  - IAM 정책을 통해 특정 사용자나 서비스가 EBS 볼륨에 접근하는 것을 제어
  - 스냅샷 또한 암호화하여 안전하게 백업 가능



# 비용 관리 및 최적화

- 비용 절감 방법
  - 필요에 따라 스냅샷을 정리하여 저장 비용 절감
  - 사용하지 않는 EBS 볼륨은 삭제하여 비용 최적화
- 효율적인 스토리지 사용
  - 데이터 액세스 패턴에 따라 볼륨 유형 선택 (gp3, sc1 등)
  - IOPS 설정을 적절히 조절하여 필요 이상의 비용 발생 방지

# EBS의 내구성과 가용성 이해



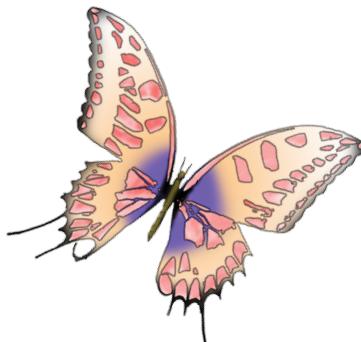
- **내구성**

- EBS는 고가용성과 내구성을 제공하며, 데이터는 다중 물리적 시설에 복제됨
- 스냅샷을 통해 추가적인 데이터 복구 방법 제공

- **가용성**

- AZ 단위의 가용성을 보장하므로 동일 리전 내 다른 AZ에 스냅샷을 이용해 복구 가능
- EC2와 결합해 고가용성 아키텍처 구축 가능

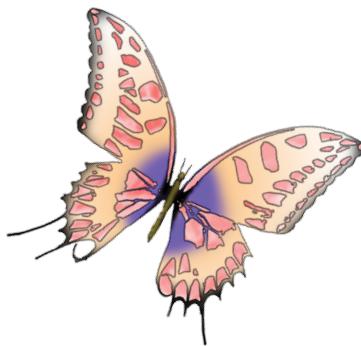
# 실습: 수동으로 볼륨 연결 및 생성



- 실습 목표
  - AWS 콘솔을 사용해 EBS 볼륨을 수동으로 생성하고 EC2에 연결
- 실습 단계
  - EBS 볼륨 생성 후 특정 EC2 인스턴스에 연결
  - EC2 인스턴스에 SSH 접속 후 볼륨 인식 확인 (`lsblk` 명령)
- 결과 확인
  - 연결된 볼륨이 EC2에서 인식되는지 확인



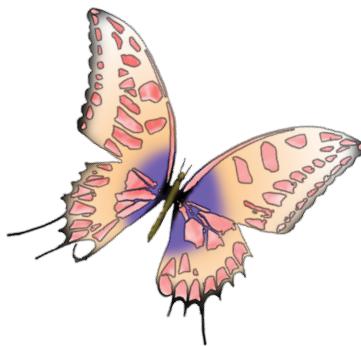
# 실습: 수동으로 EBS 볼륨 연결 및 포맷



- 실습 목표
  - 수동으로 연결된 EBS 볼륨을 파일 시스템으로 포맷하고 마운트
- 실습 단계
  - 연결된 볼륨에 파일 시스템 생성 (`mkfs -t ext4` 등)
  - 특정 디렉터리에 볼륨 마운트 후 사용 가능하게 설정
- 결과 확인
  - 마운트된 볼륨에 데이터 생성 및 읽기/쓰기 테스트



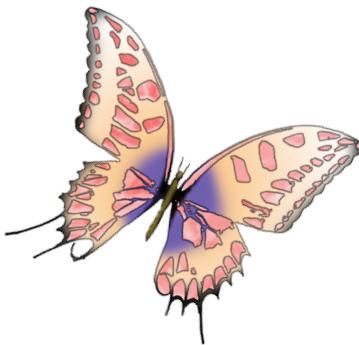
# 실습: Terraform으로 EBS 볼륨 생성



- 실습 목표
  - Terraform을 사용하여 EBS 볼륨을 자동으로 생성하고 관리
- 실습 단계
  - Terraform 설정 파일에 EBS 리소스 정의 (`resource "aws_ebs_volume"`)
  - `terraform apply` 명령을 통해 EBS 볼륨 생성
- 결과 확인
  - AWS 콘솔에서 생성된 볼륨 확인 및 Terraform 상태 파일 확인



# 실습: Terraform으로 EBS 볼륨 연결 및 포맷



- 실습 목표
  - Terraform으로 생성한 EBS 볼륨을 EC2 인스턴스에 연결하고 포맷
- 실습 단계
  - Terraform으로 EC2와 EBS 볼륨을 생성하고 연결 설정
  - 연결된 볼륨을 SSH 접속하여 포맷 및 마운트 설정
- 결과 확인
  - EC2 인스턴스에서 볼륨이 정상적으로 마운트되고 사용 가능한지 확인

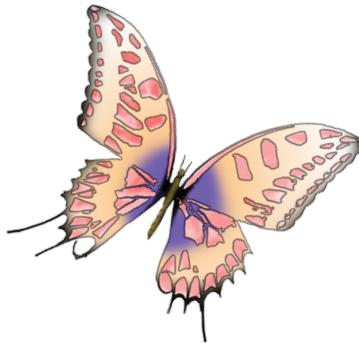




# **빅데이터 솔루션 프로비저닝**



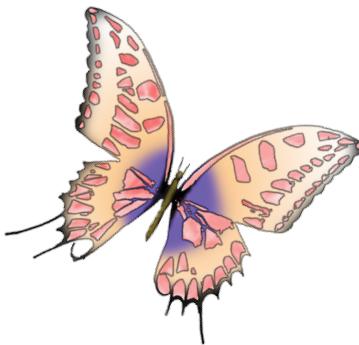
# 실습: 콘솔 인스턴스 프로비저닝



- 실습 목표
  - Terraform과 Ansible을 실행 시킬 ec2인스턴스를 생성하고 설정
- 실습 단계
  - i1 instance 생성(수동)
  - Terraform과 Ansible 설치
- 추가 실습 : Docker 기반으로 설정
- 결과 확인
  - Terraform과 Ansible의 버전 확인



# 실습: Terraform을 통한 인스턴스 프로비저닝



- 실습 목표
  - Hadoop과 Spark을 설치할 ec2 instance를 Terraform으로 설정한다
- 실습 단계
  - i1 instance에서 terraform 스크립트 실행
  - tfstate파일 확인
  - hosts파일 설정
- 결과 확인
  - ssh 접속 테스트(s1, s2, s3)
  - Ansible ad-hoc명령으로 ansible 작동여부 테스트



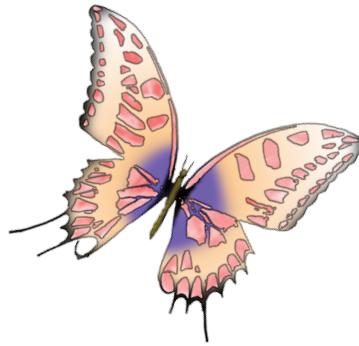


# 실습: hadoop cluster 설치



- 실습 목표
  - ec2인스턴스에 hadoop cluster를 설치한다.
- 실습 단계
  - <https://github.com/Finfra/hadoopInstall> 클론
  - 하둡 설치파일 전송
  - ansible을 통한 hadoop 설치
- 결과 확인
  - s1 인스턴스에 접속해서 hadoop 클러스터 작동여부 확인(ps명령)
  - hdfs 작동 확인
  - MapReduce작동 확인





# 실습: spark cluster 설치

- 실습 목표
  - ec2인스턴스에 spark cluster를 설치한다.
- 실습 단계
  - <https://github.com/Finfra/hadoopInstall> 클론(기존 소스 활용)
  - Spark 설치파일 전송
  - ansible을 통한 Spark 설치
- 결과 확인
  - s1 인스턴스에 접속해서 Spark 클러스터 작동여부 확인(ps명령)
  - pyspark를 통한 테스트 코드 실행

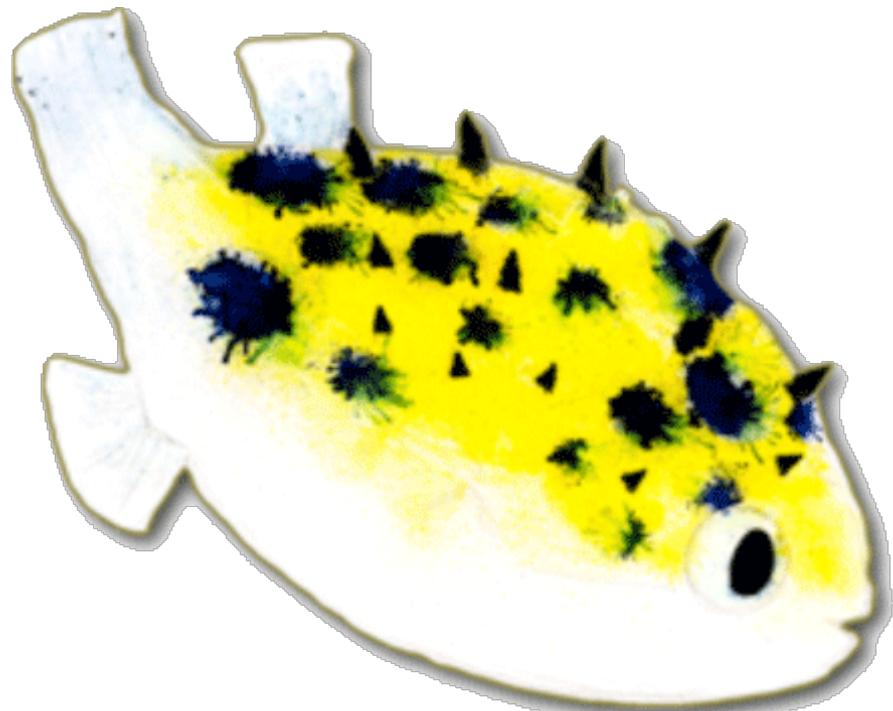




# Cf) AWS Bigdata 서비스

Hadoop 구성요소	AWS 대체 서비스	콘솔 링크
HDFS	Amazon S3	<a href="https://console.aws.amazon.com/s3/">https://console.aws.amazon.com/s3/</a>
Spark	AWS Glue EMR (Spark)	<a href="https://console.aws.amazon.com/glue">https://console.aws.amazon.com/glue</a> <a href="https://console.aws.amazon.com/elasticmapreduce">https://console.aws.amazon.com/elasticmapreduce</a>
Kafka	Amazon Kinesis MSK	<a href="https://console.aws.amazon.com/kinesis">https://console.aws.amazon.com/kinesis</a> <a href="https://console.aws.amazon.com/msk/home">https://console.aws.amazon.com/msk/home</a>
MapReduce	AWS Glue EMR (Spark)	<a href="https://console.aws.amazon.com/glue">https://console.aws.amazon.com/glue</a> <a href="https://console.aws.amazon.com/elasticmapreduce">https://console.aws.amazon.com/elasticmapreduce</a>
YARN	Glue Job Scheduler Step Functions	<a href="https://console.aws.amazon.com/glue">https://console.aws.amazon.com/glue</a> <a href="https://console.aws.amazon.com/states">https://console.aws.amazon.com/states</a>
Hive	Amazon Athena Redshift	<a href="https://console.aws.amazon.com/athena">https://console.aws.amazon.com/athena</a> <a href="https://console.aws.amazon.com/redshiftv2">https://console.aws.amazon.com/redshiftv2</a>
Oozie	Step Functions MWAA	<a href="https://console.aws.amazon.com/states">https://console.aws.amazon.com/states</a> <a href="https://console.aws.amazon.com/mwaa/home">https://console.aws.amazon.com/mwaa/home</a>
HBase	Amazon DynamoDB OpenSearch	<a href="https://console.aws.amazon.com/dynamodb">https://console.aws.amazon.com/dynamodb</a> <a href="https://console.aws.amazon.com/aos/home">https://console.aws.amazon.com/aos/home</a>

# Thank You !!



Question!!

Solution!!

The End

