AWS / GCP Fundamental

스타트업 엔지니어를 위한 IaaS/PaaS 구성 설계, 테스트 및 배포관리 가이드

2025.08

aws

Google Cloud Platform

전체 목차

본 발표자료는 스타트업 엔지니어들이 AWS와 GCP 클라우드 환경에서 IaaS와 PaaS를 효과적으로 구성하고 운영하는 방법을 다루고 있습니다. 이론적 개념부터 실무 적용 사례까지, 클라우드 운영과 배포관리 전 과정에 대한 체계적 가이드를 제공합니다.

1장. 클라우드 컴퓨팅 개요

클라우드 컴퓨팅의 기본 개념, 아키텍처, 서비스 모델 및 배포 모델에 대한 이해

스타트업을 위한 클라우드 기본 원리

학습 목표 및 기대 성과

본 강의를 통해 스타트업 엔지니어들은 AWS와 GCP 클라우드 환경에서의 운영 및 배포 관리에 대한 실무 능력을 갖추게 됩니다. 특히 비용 효율성과 확장성을 고려한 실전 아키텍처 구성에 중점을 둡니다.

- 클라우드 서비스 모델(IaaS, PaaS, SaaS)의 핵심 개념과 적용 전략 이해
- AWS와 GCP의 주요 서비스 비교 분석 및 스타트업 환경에 맞는 선택 능력
- 비용 최적화와 자원 효율성을 고려한 클라우드 아키텍처 설계
- DevOps 파이프라인 구축 및 CI/CD 자동화 능력 확보
- 실제 넷플릭스, 에어비앤비 등 유명 스타트업의 클라우드 성공 사례 분석

클라우드 컴퓨팅 정의와 특징

클라우드 컴퓨팅은 인터넷을 통해 서버, 스토리지, 데이터베이스, 네트워킹, 소프트웨어 등의 컴퓨팅 리소스를 필요에 따라 제공하는 서비스입니다. 미국 국립표준기술연구소(NIST)의 정의에 따른 5가지 핵심 특성을 가집니다.

클라우드 컴퓨팅의 5대 핵심 특성은 온디맨드 셀프 서비스(필요할 때 즉시 사용), 광범위한 네트워크 접근(다양한 기기에서 접속), 리소스 풀링(다수 사용자에게 리소스 공유), 신속한 확장성(필요에 따라 신속하게 규모 조절), 그리고 측정 가능한 서비스(사용량 기반 과금)입니다.

이러한 특성들은 스타트업이 적은 초기 투자로 빠르게 확장 가능한 인프라를 구축할 수 있게 하며, AWS와 GCP는 이러한 특성을 충실히 구현한 대표적인 클라우드 서비스 제공업체입니다.

클라우드 컴퓨팅 핵심 5대 특성

미국 국립표준기술연구소(NIST)에서 정의한 클라우드 컴퓨팅의 핵심 특성은 스타트업이 IT 인프라를 유연하게 활용하고 비용을 효율화하는 기반이 됩니다. 이러한 특성들이 스타트업의 빠른 확장과 시장 대응력을 가능하게 합니다.

01

온디맨드 셀프 서비스

사용자가 인적 상호작용 없이 필요할 때마다 서버, 스토리지 등의 컴퓨팅 리소스를 자동으로 프로비저닝할 수 있습니다. AWS 콘솔이나 GCP 대시보드에서 클릭 몇 번으로 인프라 자원을 즉시 확보하고 필요 없을 때 해제할 수 있어 빠른 개발 주기를 지원합니다.

02

광범위한 네트워크 접근

표준 메커니즘을 통해 네트워크로 접근 가능하며, 다양한 클라이언트 플랫폼(PC, 모바일, 태블릿 등)에서 사용할 수 있습니다. REST API, 웹 콘솔, 모바일 앱 등 다양한 인터페이스를 통해 언제 어디서든 클라우드 리소스를 관리할 수 있어 원격 작업과 글로벌 협업이 용이합니다.

03

리소스 풀링

공급자의 컴퓨팅 리소스는 다중 테넌트 모델을 사용하여 풀링되며, 사용자 수요에 따라 물리적, 가상 자원이 동적으로 할당되고 재할당됩니다. 이를 통해 인프라 사용률을 극대화하고, 규모의 경제를 실현하여 개별 기업이 자체 인프라를 구축할 때보다 훨씬 저렴한 비용으로 고성능 인프라를 활용할 수 있습니다.

04

신속한 확장성

필요에 따라 리소스를 빠르게 확장하거나 축소할 수 있는 능력입니다. 사용자에게는 무제한의 리소스를 언제든지 원하는 양만큼 프로비저닝할 수 있는 것처럼 보입니다. Auto Scaling 기능을 통해 트래픽 급증 시 자동으로 인스턴스를 추가하고, 트래픽이 감소하면 자동으로 리소스를 줄여 비용을 최적화합니다.

05

측정 가능한 서비스

클라우드 시스템은 리소스 사용량을 자동으로 모니터링, 제어, 보고하여 투명성을 제공합니다. 사용한 만큼만 정확히 지불하는 종량제 방식으로, AWS Cost Explorer나 GCP Billing Dashboard를 통해 실시간으로 비용을 추적하고 예산을 관리할 수 있어 스타트업의 비용 예측성을 높여줍니다.

클라우드 컴퓨팅 3단계 계층 구조

01

사용자 계층 (User Layer)

클라우드 서비스를 이용하는 다양한 유형의 사용자들이 존재하며, 각 사용자마다 다른 목적과 권한을 갖습니다.

개발자 운영자 시스템 관리자 최종 사용자 비즈니스 의사결정자

02

서비스 계층 (Service Layer)

클라우드 제공 업체가 사용자에게 제공하는 다양한 수준의 서비스 모델로, 사용자의 책임 범위가 달라집니다.

SaaS (Software as a Service) PaaS (Platform as a Service) IaaS (Infrastructure as a Service)
CaaS (Container as a Service) FaaS (Function as a Service)

03

인프라 계층 (Infrastructure Layer)

클라우드 서비스를 지원하는 물리적/가상 컴퓨팅 자원들로 구성된 기반 계층입니다.

서버 스토리지 네트워크 데이터센터 가상화 보안 인프라

서비스 모델:

IaaS/PaaS/SaaS

클라우드 컴퓨팅은 제공하는 서비스 수준에 따라 세 가지 주요 서비스 모델로 구분됩니다. 각 모델은 사용자와 제공자 간 책임 범위가 다릅니다.

IaaS(Infrastructure as a Service): 가상 머신, 스토리지, 네트워크와 같은 기본 인프라를 제공합니다. 사용자는 OS부터 애플리케이션까지 관리해야 합니다.

예시: AWS EC2, GCP Compute Engine, 가상 서버

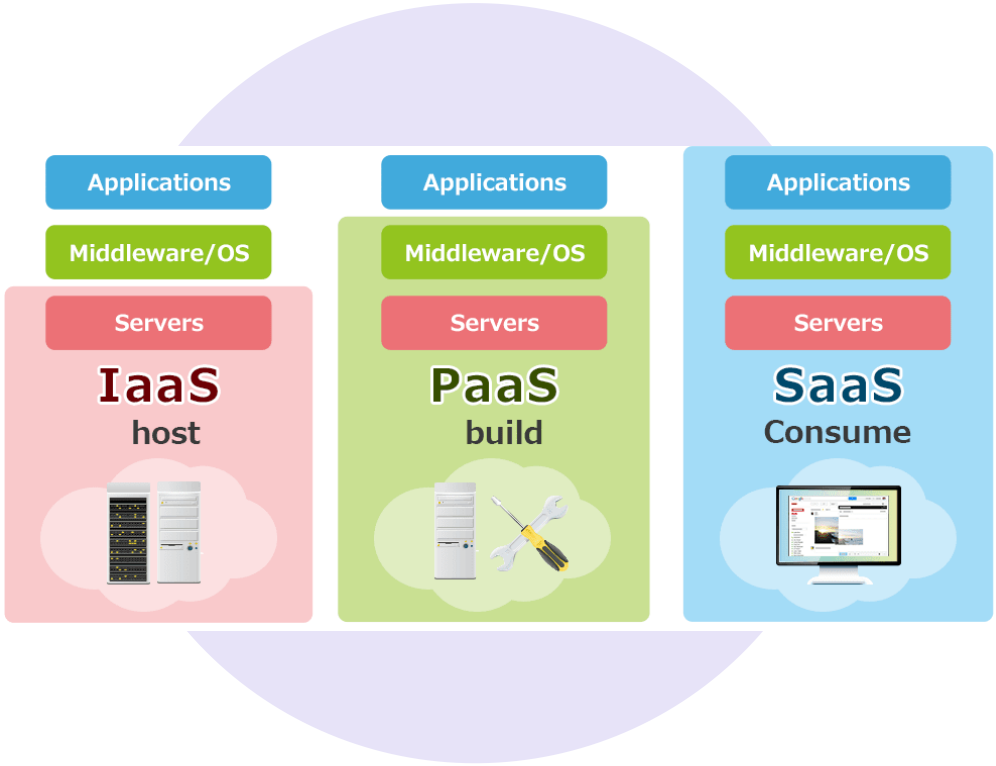
PaaS(Platform as a Service): 애플리케이션 개발 및 구축에 필요한 플랫폼을 제공합니다. 인프라는 제공자가 관리하고, 사용자는 애플리케이션과 데이터만 관리합니다.

예시: AWS Elastic Beanstalk, GCP App Engine, Heroku

SaaS(Software as a Service): 완전한 소프트웨어 솔루션을 인터넷을 통해 제공합니다. 사용자는 구성만 하고, 나머지는 모두 제공자가 관리합니다.

예시: Google Workspace, Slack, Salesforce

스타트업에서는 초기 개발 속도와 비용 효율성을 위해 **PaaS**와 **SaaS**를 주로 활용하고, 확장 단계에서 **IaaS**로 일부 마이그레이션하는 전략이 효과적입니다.



서비스 모델 시각화

IaaS (Infrastructure as a Service)

하드웨어 인프라를 서비스로 제공

스타트업 사례: *AWS EC2, Google Compute Engine,* 초기 인프라 구축 시 비용 절감

PaaS (Platform as a Service)

애플리케이션 개발/배포 플랫폼 제공

스타트업 사례: *AWS Beanstalk, Google App Engine,* 빠른 프로토타입 구축 및 시장 검증

SaaS (Software as a Service)

완전한 애플리케이션 서비스 제공

스타트업 사례: *Google Workspace, Salesforce,* 핵심 비즈니스에 집중 가능

클라우드 컴퓨팅의 세 가지 주요 서비스 모델(IaaS, PaaS, SaaS)은 **기업 책임 범위**와 **제공업체 관리 범위**에 따라 구분됩니다. 스타트업은 성장 단계와 기술 역량에 맞게 적절한 모델을 선택해야 합니다.

IaaS는 최대한의 제어가 필요한 초기 스타트업에 적합하며, **PaaS**는 빠른 개발과 배포 가 중요한 성장기 스타트업에 효과적입니다. **SaaS**는 핵심 비즈니스에 집중하며 IT 인 프라 관리 부담을 최소화하려는 경우 이상적입니다.

"스타트업의 성공은 적절한 시점에 올바른 클라우드 서비스 모델을 선택하는 것에서 시작됩니다."

배포 모델 비교

클라우드 배포 모델은 클라우드 리소스가 누구에 의해 소유되고, 어디에 위치하며, 어떻게 관리되는지를 결정합니다. 각 모델은 서로 다른 스타트업 요구사항과 비즈니스 목표에 맞게 선택할 수 있습니다.

01

퍼블릭 클라우드(Public Cloud): 인터넷을 통해 제3자 제공업체가 소유하고 관리하는 컴퓨팅 서비스입니다. 초기 비용이 없고 확장성이 높으며 관리 오버헤드가 낮은 장점이 있어 스타트업, SaaS 기업, 웹 애플리케이션에 이상적입니다. 단점으로는 제한된 커스터마이징과 규제가 엄격한 산업에서의 컴플라이언스 이슈가 있습니다.

02

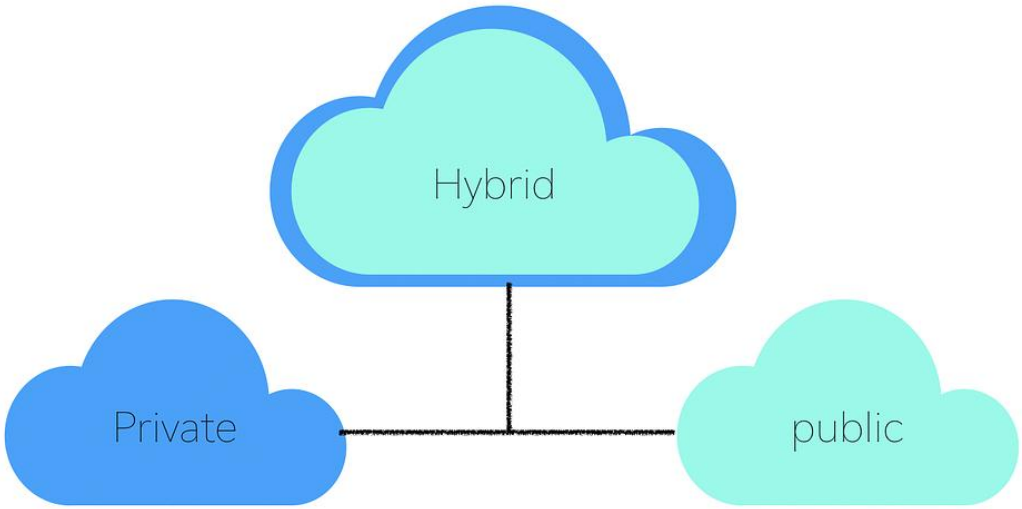
프라이빗 클라우드(Private Cloud): 단일 조직에서 독점적으로 사용하는 클라우드 환경으로, 온프레미스 또는 제3자 데이터 센터에 위치할 수 있습니다. 데이터 보안, 규정 준수, 커스터마이징이 필요한 금융, 의료, 정부 기관에 적합합니다. 높은 초기 투자와 유지보수 비용, 확장 제한이 단점입니다.

03

하이브리드 클라우드(Hybrid Cloud): 퍼블릭과 프라이빗 클라우드 환경을 통합한 모델로, 데이터와 애플리케이션이 두 환경 사이를 이동할 수 있습니다. 성장 중인 스타트업, 핀테크, 규제 대상 산업에 적합하며, 민감한 데이터는 프라이빗에, 확장이 필요한 서비스는 퍼블릭에 배치하는 유연성을 제공합니다. 복잡한 인프라 관리가 필요합니다.

04

멀티 클라우드(Multi-Cloud): 여러 클라우드 제공업체(AWS, GCP, Azure 등)의 서비스를 동시에 사용하는 전략입니다. 벤더 종속성 감소, 장애 대응력 강화, 최적 서비스 선택이 가능하지만, 복잡한 관리, 일관된 보안 정책 수립, 통합 모니터링 구축이 필요합니다. DevOps 성숙도가 높은 스타트업에 적합합니다.



배포 모델 시각화

클라우드 배포 모델은 클라우드 리소스가 누구에게 제공되고 어디에 위치하는지에 따라 구분됩니다. 스타트업의 성장 단계와 요구사항에 맞는 최적의 모델을 선택하는 것이 중요합니다.

- **퍼블릭 클라우드:** AWS, GCP와 같은 서비스 제공업체가 인터넷을 통해 다수의 고객에게 리소스를 제공. 초기 스타트업에 적합하며 초기 투자 비용이 낮고 빠른 확장성 제공.
- **프라이빗 클라우드:** 단일 조직만을 위한 전용 클라우드 환경. 데이터 주권이나 규제 준수가 중요한 금융, 의료 스타트업에 적합.
- **하이브리드 클라우드:** 퍼블릭과 프라이빗 클라우드의 조합. 성장 중인 스타트업에서 민감한 데이터와 일반 워크로드를 분리할 때 효과적.

스타트업 선택 기준: 비용 효율성, 확장성 요구사항, 데이터 보안 민감도, 법적 규제, 현재 인프라와의 호환성을 고려하세요. 초기 단계에서는 퍼블릭 클라우드로 시작하여 점진적으로 하이브리드 모델로 전환하는 전략이 일반적입니다.

"올바른 배포 모델 선택은 스타트업의 기술 부채를 줄이고 미래 성장을 준비하는 핵심 결정입니다." — 클라우드 아키텍처 전문가

31% vs 10%

AWS vs GCP 시장비교

2025년 글로벌 클라우드 시장에서 **AWS**는 **31%**의 점유율로 선두를 유지하고 있으며, **GCP**는 **10%**의 점유율로 3위를 차지하고 있습니다. AWS는 기업용 서비스와 광범위한 IaaS 영역에서 강점을 보이는 반면, GCP는 데이터 분석과 AI/ML 영역에서 빠르게 성장하고 있습니다.

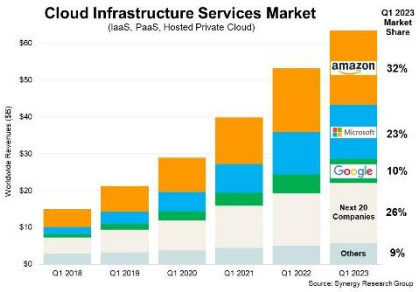
스타트업의 경우, 초기 단계에서는 **AWS** 프리티어와 **GCP** 무료 크레딧을 활용한 하이브리드 전략이 증가하고 있으며, 서비스 규모 확장에 따라 비용 효율성 및 특화된 기능에 맞춰 서비스를 선택하는 경향이 두드러지고 있습니다.

AWS

- 175+ 서비스 제공
- 다양한 산업별 솔루션
- 성숙한 시장과 에코시스템
- 월평균 비용 증가율: 9%

GCP

- 120+ 서비스 제공 데이터/AI 중심 서비스 강점
- 연간 성장률: 27.5%
- 대규모 데이터 처리 비용 우위



AWS vs GCP 서비스별 주요 차이

주요 서비스 카테고리별 핵심 비교 및 스타트업 선택 가이드

컴퓨팅 서비스

- **AWS EC2** vs **GCP Compute Engine**: AWS는 인스턴스 유형이 더 다양하고, GCP는 커스텀 머신 유형 지원
- **AWS Lambda** vs **GCP Cloud Functions**: AWS가 더 긴 실행 시간과 메모 리 옵션 제공

스토리지 서비스

- **AWS S3** vs **GCP Cloud Storage**: 비슷한 기능이나 GCP는 단일 스토리지 클래스, AWS는 다양한 스토리지 클래스 제공
- **AWS EBS** vs **GCP Persistent Disk**: GCP는 자동 확장 기능 내장
- **AWS EFS** vs **GCP Filestore**: AWS가 더 저렴하고 다양한 성능 옵션 제공

데이터베이스 서비스

- **AWS RDS** vs **GCP Cloud SQL**: AWS는 Oracle, Aurora 추가 지원, GCP 는 단순한 설정 제공
- **AWS DynamoDB** vs **GCP Firestore**: DynamoDB는 성능 우수, Firestore는 실시간 기능 우수

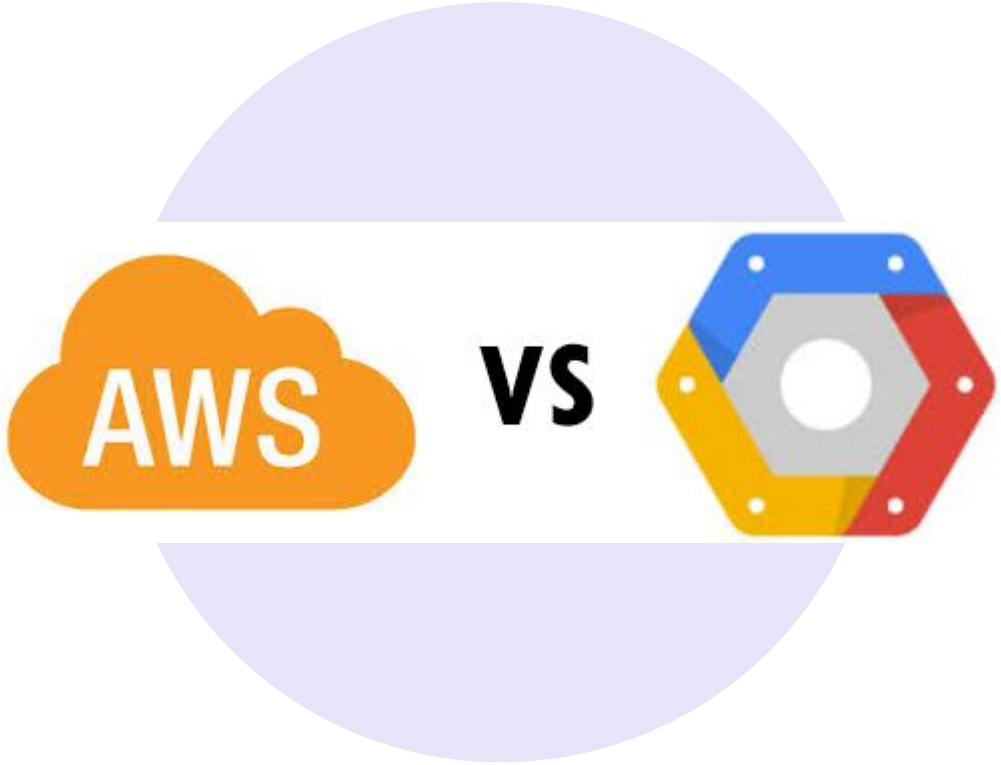
네트워킹 및 AI/ML

- **AWS Route 53** vs **GCP Cloud DNS**: AWS가 더 많은 라우팅 옵션 제공
- **AWS CloudFront** vs **GCP Cloud CDN**: AWS가 더 많은 엣지 로케이션 보 유
- **AWS SageMaker** vs **GCP Vertex AI**: GCP는 기본 모델과 AutoML 강점

AWS vs GCP
주요 아키텍처 차이

AWS와 GCP는 각각 고유한 아키텍처 방식과 네이티브 서비스를 제공합니다. 이러한 차이점을 이해하면 스타트업에 최적화된 클라우드 설계가 가능합니다.

기능	AWS	GCP
DNS 서비스	**Route 53** 글로벌 단일 서비스, 상태 확인 내장	**Cloud DNS** 단순 구조, API 중심 관리
로드밸런싱	**ELB** ALB, NLB, CLB 유형별 분리	**Cloud Load Balancing** 통합형 글로벌/리전 밸런싱
네트워크	**VPC** 리전 기반 구성, 서브넷 필수	**VPC** 글로벌 리소스, 서브넷은 리전별 관리
IAM	**IAM** 사용자/그룹/역할 기반	**IAM** 주 구성원-역할-리소스 모델



65%

클라우드 도입 장점

스타트업이 클라우드 인프라를 도입하면 평균 **65%**의 초기 인프라 비용을 절감할 수 있으며, 제품 출시 시간(Time to Market)을 **3**배 이상 단축할 수 있습니다. 클라우드는 스타트업에게 경제적 효율성, 신속한 확장성, 접근성 측면에서 핵심 경쟁력을 제공합니다.

47%

운영 비용 절감

클라우드 도입 스타트업은 인프라 유지보수, 전력, 인건비 등에서 평균 47%의 운영 비용을 절감했습니다.

78%

빠른 확장성

스타트업의 78%가 클라우드 덕분에 사용자 급증 시에도 다운타임 없이 즉시 서비스 확장이 가능했다고 보고했습니다.

91%

글로벌 접근성

클라우드 활용 스타트업의 91%가 초기부터 글로벌 서비스 진출이 용이해져 시장 확장 기회가 증가했다고 응답했습니다.

클라우드 도입 고려사항

클라우드 컴퓨팅의 수많은 이점에도 불구하고, 스타트업이 클라우드를 도입할 때 고려해야 할 여러 가지 잠재적 위험 요소가 존재합니다. 이러한 단점을 인지하고 대응 전략을 마련하는 것이 중요합니다.

- 보안 및 규정 준수: 데이터가 외부 공급자에게 저장되므로 데이터 유출, 규제 위반 위험이 증가합니다. 특히 금융, 의료 분야의 스타트업은 규제 준수 여부를 면밀히 검토해야 합니다.
- 인터넷 의존성: 인터넷 연결 장애 시 서비스 접근이 불가능합니다. 대역폭 제한은 대용량 데이터 처리가 필요한 애플리케이션에 병목 현상을 일으킬 수 있습니다.
- 제어권 제한: 물리적 인프라에 대한 직접 제어가 불가능하며, 클라우드 공급자의 정책이나 서비스 변경에 영향을 받을 수 있습니다.
- 예상치 못한 비용: 초기 비용은 낮지만, 잘못된 구성이나 자원 관리 부재로 장기적으로 예상보다 높은 비용이 발생할 수 있습니다.

Public Cloud		Private Cloud
유지관리 용이 사용량에 따른 비용 지불 가능 유동적으로 리소스 활용 가능	장점	강력한 보안 비즈니스에 맞는 클라우드 환경 조성 가능
데이터가 외부에 존재한다는 리스크	단점	클라우드 관리를 위한 자체 전문 인력 필요
메일, 일정관리 등 개인용 및 중소형 웹 비즈니스 서비스	주요 적용 분야	ERP 등의 기업 핵심 애플리케이션 및 중대형 웹 비즈니스 서비스

스타트업 클라우드 실전 트렌드

현대 스타트업들은 클라우드 기술을 통해 **빠른 시장 검증**과 **확장 가능한 인프라**를 구현하고 있습니다. 특히 2025년 이후 스타트업들은 기존 리소스 프로비저닝 개념에서 벗어나 서버리스, 컨테이너, 관리형 서비스를 중심으로 인프라를 구축하는 추세입니다.

초기 단계에서는 **PaaS와 서버리스** 아키텍처로 빠른 MVP를 구현하고, 성장기에 들어서면 **멀티 클라우드**나 **하이브리드 클라우드**로 글로벌 시장을 공략합니다. 이를 통해 초기 투자비용 최소화와 빠른 시장 검증, 이후 탄력적인 확장이 가능합니다.

"스타트업의 핵심 경쟁력은 속도와 유연성에 있습니다. 클라우드는 이 두 가지를 모두 제공합니다."

빠른 MVP 구축

서버리스 아키텍처와 관리형 서비스를 활용해 최소 기능 제품(MVP)을 몇 주 만에 시장에 출시합니다.

사례: 핀테크 스타트업 *A*사는 *AWS Lambda*와 *API Gateway*로 결제 시스템을 *8*주 만에 구축하여 초기 투자자 확보

글로벌 인프라 확장

CDN, 멀티 리전 배포, 글로벌 로드밸런서를 활용해 초기부터 글로벌 서비스를 위한 확장성 있는 인프라를 구축합니다.

사례: *B*커머스는 *GCP*의 *Cloud CDN*과 리전별 *Cloud SQL*로 북미/유럽/아시아 시장을 동시 진출

시장 확장과 비용 최적화

자동 스케일링과 서버리스, 스팟 인스턴스를 통해 트래픽 증가에도 비용 효율적인 운영이 가능합니다.

사례: 동영상 플랫폼 *C*사는 *AWS Spot Instance*와 *S3 Intelligent-Tiering*으로 운영비 *62%* 절감 달성

실습 준비사항 체크리스트

AWS와 GCP 클라우드 실습을 위해 필요한 필수 도구와 계정 준비 사항을 체크리스트로 정리했습니다. 스타트업 환경에서도 즉시 실습할 수 있도록 모든 필요 요소를 포함했으며, 각 항목은 실무에서도 자주 활용되는 기본 도구입니다.

01 기본 개발 환경

- ✓ 최신 웹 브라우저 (Chrome/Firefox 권장) - 개발자 도구 활용 가능
- ✓ 터미널 에뮬레이터 (Windows: PowerShell/WSL, macOS: Terminal, Linux: Bash/Zsh)
- ✓ 텍스트 에디터 또는 IDE (VSCode 권장, AWS/GCP 확장 기능 설치)

02 네트워크 도구

- ✓ SSH 클라이언트 (OpenSSH, PuTTY 등) - 원격 서버 접속용
- ✓ SSH 키 생성 준비 (RSA/ED25519)
- ✓ SFTP 클라이언트 (필요 시) - FileZilla, WinSCP 등

03 API 테스트 도구

- ✓ Postman 또는 Insomnia - API 테스트 및 문서화
- ✓ curl/wget 명령어 기본 사용법 숙지
- ✓ jq 유틸리티 (선택) - JSON 처리 효율화

04 AWS 계정 준비

- ✓ 유효한 이메일 주소 (계정 생성 및 알림용)
- ✓ 신용카드 또는 체크카드 (인증용, 실습은 무료 티어 범위 내)
- ✓ 전화번호 (SMS 인증용)
- ✓ MFA 인증 앱 (Google Authenticator, Authy 등) - 권장

05 GCP 계정 준비

- ✓ Google 계정 (Gmail 등)
- ✓ 신용카드 또는 체크카드 (인증용, $300 무료 크레딧 활용)
- ✓ 전화번호 (인증용)
- ✓ 프로젝트 기획 아이디어 (GCP 프로젝트 구성용)

AWS/GCP 계정 준비 체크리스트

클라우드 계정을 안전하게 생성하고 관리하기 위한 필수 체크리스트입니다. 스타트업 환경에서 시작할 때 발생할 수 있는 보안 위험과 예상치 못한 비용 발생을 방지하기 위한 실전 팁을 제공합니다. AWS와 GCP 모두 유사한 절차가 필요하지만, 서비스별 특징을 반영한 설정이 중요합니다.

01 신용카드 및 결제 수단 준비

AWS 국제 결제 가능 신용카드 필수, 체크카드도 가능하나 일부 제한

GCP 신용카드, 체크카드, 일부 국가 직불카드 지원, 300달러 무료 크레딧 활용

신용카드 도난/부정사용 알림 설정, 전용 클라우드 결제 카드 사용 권장

02 본인 인증 및 계정 보안

AWS 전화번호 인증, 이메일 확인, 계정 생성 후 즉시 MFA 설정

GCP Google 계정 연동, 조직 계정 생성 시 도메인 소유권 인증

루트/소유자 계정에 하드웨어 보안키(Yubikey 등) 사용 적극 권장

03 결제 알림 및 예산 설정

AWS Budgets 서비스로 임계값 설정, SNS 알림 연동, Cost Explorer 활성화

GCP Billing 메뉴에서 예산 및 알림 설정, 예산별 알림 임계값 설정(50%, 90%, 100%)

월별/프로젝트별 예산 설정, 과금 이상징후 자동 알림, 일일 사용량 리포트 설정

04 비용 효율화 초기 설정

AWS 리소스 태깅 전략 수립, 비용 할당 태그 활성화, 무료 티어 사용량 알림 설정

GCP 라벨 전략 구현, 비용 내보내기 BigQuery 연동, 할인 프로그램 검토

실수로 인한 과다 사용 방지를 위한 리소스 생성 권한 제한 및 리전별 한도 설정

05 액세스 관리 및 계정 구조화

AWS 루트 계정 MFA 필수, 관리자 IAM 계정 별도 생성, 액세스 키 관리

GCP 프로젝트 구조 설계, 서비스 계정 생성 및 키 관리, 최소 권한 원칙 적용

개발/스테이징/프로덕션 환경 분리, 비상시 권한 상승 프로세스 마련

실전 설계 사례: 스타트업 MVP

스타트업 초기 단계에서는 빠른 출시와 비용 효율성이 핵심입니다. 이를 위해 IaaS와 PaaS를 전략적으로 혼합한 MVP 아키텍처를 구성하는 것이 중요합니다.

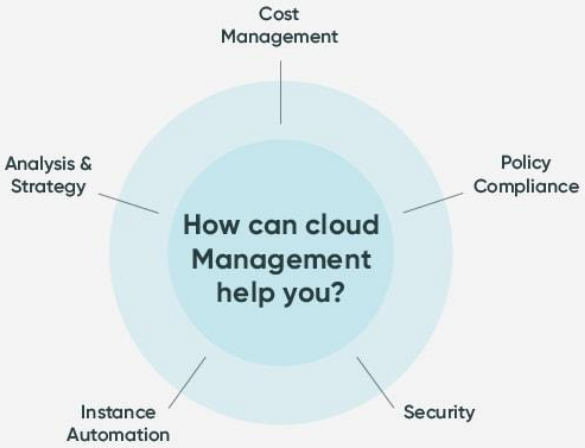
- 코어 인프라 - **IaaS**: AWS EC2/GCP VM으로 핵심 애플리케이션 호스팅 (커스텀 설정 가능)
- 개발 가속화 - **PaaS**: AWS Elastic Beanstalk/GCP App Engine으로 애플리케이션 배포 자동화
- 데이터 관리: AWS RDS/GCP Cloud SQL로 관리형 DB 서비스 활용
- 비용 최적화: AWS Auto Scaling/GCP Instance Groups로 수요에 따른 자동 확장
- 개발자 효율성: 자체 관리 인프라를 최소화하고 관리형 서비스 활용 극대화

문제 해결 사례: 예산 초과 방지와 리스크 관리

스타트업에게 클라우드 비용 관리는 매우 중요합니다. 예산 초과 방지와 효과적인 리스크 관리 전략으로 예상치 못한 비용 발생을 방지할 수 있습니다.

- 비용 모니터링 자동화 - AWS Budget 및 GCP Budget Alert으로 일일/주간/월간 사용량 임계치 설정 및 알림 구성
- 리소스 태깅 전략 - 비용 할당을 위한 체계적인 태그 정책 수립 (부서/기능/프로젝트별)
- 자동 종료 정책 - 개발/테스트 환경의 야간/주말 자동 종료 스크립트 구현 (약 45% 비용 절감)
- 장애 대응 **Flow** - 예상치 못한 리소스 증가, 네트워크 장애, 보안 이슈 발생 시 단계별 대응 체계 구축

특히 **CloudWatch**와 **Cloud Monitoring**을 연동한 실시간 대시보드를 통해 이상 징후를 조기에 포착하고, Slack/이메일 알림과 자동화된 대응 람다 함수를 구성하세요.



클라우드 도입 로드맵

스타트업 성장 단계별 클라우드 구축 모델 가이드

01 PoC 단계

초기 스타트업은 최소한의 비용으로 아이디어 검증에 집중해야 합니다. 서버리스 아키텍처와 관리형 서비스를 활용하여 인프라 관리 부담을 최소화하세요.

- AWS Lambda/GCP Functions
- S3/Cloud Storage
- DynamoDB/Firestore
- Free Tier 활용

02 베타 단계

제품-시장 적합성(PMF)을 확인하는 단계로, 확장 가능성을 고려한 아키텍처 설계가 필요합니다. 컨테이너 기반 서비스와 자동 스케일링을 도입하세요.

- ECS/GKE
- Auto Scaling
- RDS/Cloud SQL
- CDN 도입

03 프로덕션 단계

안정적인 서비스 운영을 위한 고가용성 아키텍처와 모니터링 체계를 구축합니다. 다중 가용영역 배포와 자동화된 CI/CD 파이프라인을 구현하세요.

- 다중 AZ 배포
- CI/CD 파이프라인
- IAM/Security
- 모니터링/알림

04 글로벌 확장

전 세계 사용자를 대상으로 서비스를 제공하기 위한 다중 리전 아키텍처와 글로벌 네트워크 최적화가 필수입니다. 비용 최적화와 규제 준수도 고려하세요.

- 다중 리전 배포
- 글로벌 로드밸런싱
- 데이터 주권 준수
- 비용 최적화

클라우드 네이티브 문법

인프라 as 코드(IaC)는 클라우드 네이티브 환경에서 인프라를 코드로 정의하고 관리하는 방법론입니다. 스타트업 엔지니어는 IaC를 통해 인프라 구성을 자동화하고 반복 가능하게 만들어 휴먼 에러를 줄이고 배포 속도를 높일 수 있습니다.

AWS CloudFormation, GCP Deployment Manager, terr 등의 도구를 활용하여 전체 인프라 스택을 코드로 관리함으로써 버전 관리, 협업, 환경 복제가 간소화됩니다.

01 IaC 도구 선택 가이드: AWS 전용 환경에는 CloudFormation, GCP 전용 환경에는 Deployment Manager를 사용하고, 멀티 클라우드 환경이나 하이브리드 환경에서는 Terraform을 사용하는 것이 효율적입니다. 스타트업 초기 단계에서는 학습 곡선이 낮은 Pulumi(Python, TypeScript 지원)도 고려해볼 수 있습니다.

02 모듈화 및 재사용성 확보: IaC 코드를 모듈화하여 재사용 가능한 컴포넌트로 구성하세요. AWS CDK 또는 Terraform 모듈을 활용하면 개발팀이 공통 패턴을 표준화하고 일관성 있게 적용할 수 있습니다. 마이크로서비스 단위로 인프라 코드를 분리하여 관리하면 서비스별 독립적인 배포가 가능해집니다.

03 환경 분리 전략: 개발(Dev), 스테이징(Staging), 프로덕션(Prod) 환경을 코드에서 명확히 분리하되 동일한 템플릿을 사용하세요. 환경별 변수만 다르게 적용하여 일관성을 유지하고, AWS Organization 또는 GCP Folders를 통해 계정/프로젝트 단위로 환경을 격리하는 것이 안전합니다.

04 상태 관리 최적화: Terraform 사용 시 원격 상태 저장소(AWS S3+DynamoDB 또는 GCP Cloud Storage)를 구성하여 팀 협업을 지원하세요. 상태 파일은 민감 정보를 포함하므로 암호화와 액세스 제한이 필수입니다. 상태 잠금(State Locking) 메커니즘을 사용하여 동시 변경을 방지하세요.

05 CI/CD 파이프라인 통합: IaC 코드 변경을 Git으로 관리하고, GitHub Actions, AWS CodePipeline, GCP Cloud Build와 같은 CI/CD 도구와 통합하세요. Pull Request 단계에서 자동 검증(terraform plan, cfn-lint)을 수행하고, 승인 후 자동 배포하는 GitOps 모델을 적용하면 안전하고 일관된 배포가 가능합니다.

06 드리프트 감지 및 관리: 코드로 정의된 상태와 실제 인프라 간 차이(드리프트)를 정기적으로 확인하세요. AWS Config, GCP Config Management 또는 Terraform의 plan 명령어를 스케줄링하여 드리프트를 자동 감지하고, 수정 조치를 취하는 자동화 워크플로우를 구축하면 일관성을 유지할 수 있습니다.

보안 이슈와 대응 방안

클라우드 환경에서 스타트업이 직면하는 주요 보안 위협은 **자격 증명 관리 취약점**, **분산 서비스 거부 공격(DDoS)**, 그리고 **단일 장애점** 위험입니다. 이러한 위협에 체계적으로 대응하면서도 비용 효율적인 전략이 필요합니다.

보안 자동화는 스타트업의 제한된 인력으로도 **엔터프라이즈급 보안 체계**를 구축할 수 있게 해줍니다. IAM 정책 자동화, 이벤트 기반 보안 대응, 인프라의 지역적 분산은 비즈니스 연속성과 데이터 보호를 동시에 달성하는 핵심 전략입니다.

"클라우드 보안은 제품이 아닌 지속적인 과정입니다. 자동화된 방어 체계를 구축하는 것이 성공의 열쇠입니다."

IAM/키 관리

- **최소 권한 원칙**: 필요한 권한만 부여하여 보안 위험 최소화
- **MFA 활성화**: 관리자 및 파워 유저 계정에 다중 인증 필수 적용
- **액세스 키 교체**: 정기적인 키 순환으로 장기 노출 위험 감소
- **비밀 저장소 활용**: AWS Secrets Manager, GCP Secret Manager로 API 키 중앙화 관리

DDoS/Bad Bot 자동화 대응

- **클라우드 방화벽**: AWS Shield, GCP Cloud Armor로 공격 자동 차단
- **WAF 규칙**: 악성 트래픽 패턴 식별 및 필터링 자동화
- **자동 스케일링**: 트래픽 급증 시 자원 자동 확장으로 가용성 보장
- **봇 관리**: reCAPTCHA, 행동 분석으로 악성 봇 차단

리전/가용영역 믹스 전략

- **다중 리전 배포**: 지역 장애에도 서비스 가용성 유지
- **가용영역 분산**: 단일 DC 장애 발생해도 서비스 지속
- **데이터 복제**: 리전 간 자동 데이터 복제로 재해 복구 계획 구현
- **글로벌 엣지**: CDN 활용으로 성능 향상 및 DDoS 방어력 강화

Q&A - 클라우드 도입의 모든 궁금증

Q. 스타트업에게 가장 적합한 클라우드 서비스 모델은 무엇인가요?

A. 초기 스타트업은 **PaaS**로 시작하여 개발 속도를 높이고, 성장 단계에 따라 **IaaS**를 혼합하는 하이브리드 접근이 효과적입니다. 제품 검증 단계에서는 서버리스로 비용 효율성을 높일 수 있습니다.

Q. AWS와 **GCP** 중 어떤 클라우드를 선택해야 할까요?

A. 기술 스택, 예산, 필요 서비스에 따라 다릅니다. **AWS**는 서비스 다양성과 성숙도가 높고, **GCP**는 데이터 분석과 ML 서비스가 강점입니다. 멀티클라우드 전략도 고려해볼 수 있습니다.

Q. 클라우드 비용을 효과적으로 관리하는 방법은?

A. 예산 알림 설정, 태그 기반 비용 추적, 자동 스케일링, 비용 최적화 도구(AWS Cost Explorer, GCP Cost Management) 활용이 중요합니다. 또한 리저브드 *커밋* 인스턴스로 장기 사용 시 **30-70%** 비용 절감이 가능합니다.

클라우드 트렌드와 앞으로의 발전

스타트업이 주목해야 할 클라우드 산업의 미래 발전 방향

AI/ML 융합 전망

- **내장형 AI 서비스:** 2025년까지 90% 이상의 클라우드 서비스에 AI 기능 내장 전망
- **스타트업 기회:** AWS Bedrock, GCP Gemini 등 생성형 AI API 활용 확대
- **MLOps 자동화:** 모델 개발부터 배포까지 전 과정 자동화 플랫폼 등장

BigData 서비스 발전

- **실시간 분석 표준화:** AWS Kinesis, GCP Dataflow 등 실시간 처리 비용 70% 절감 전망
- **서버리스 데이터 웨어하우스:** 인프라 관리 없는 분석 환경(BigQuery, Redshift Serverless)

차세대 서버리스 발전

- **서버리스 컨테이너:** AWS Fargate, GCP Cloud Run 중심 아키텍처로 전환
- **WASM 런타임:** 2025년부터 언어 제약 없는 새로운 서버리스 런타임 표준화
- **eBPF 기반 관측성:** 서버리스 환경에서 성능 및 보안 모니터링 고도화 **이벤트**
- **메시:** 마이크로서비스 간 복잡한 이벤트 처리 추상화 레이어 등장

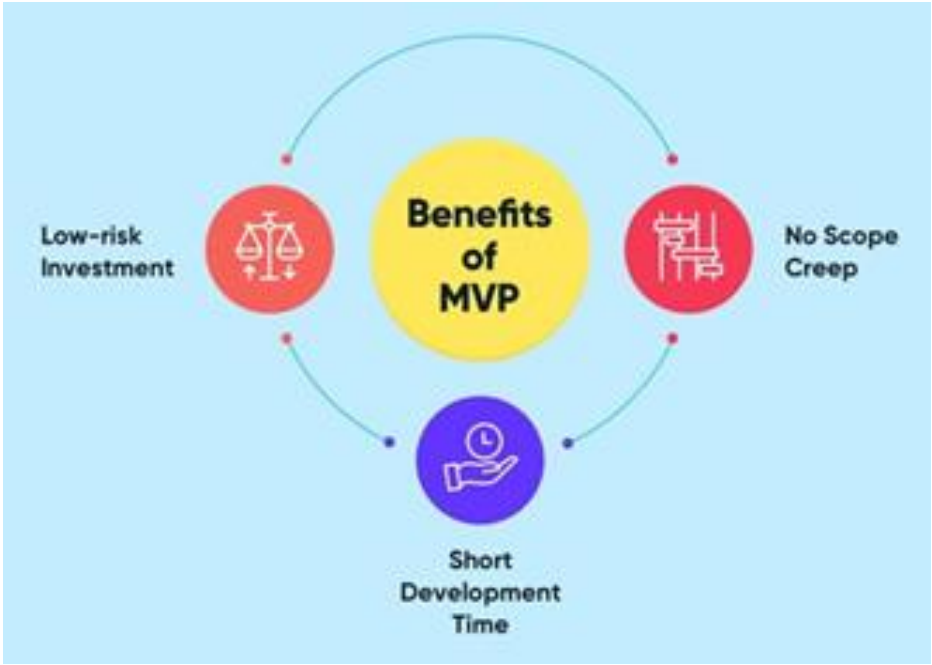
운영 패러다임 변화

- **플랫폼 엔지니어링:** DevOps에서 내부 개발자 플랫폼(IDP) 중심으로 전환
- **FinOps 표준화:** AWS/GCP 비용 최적화를 위한 자동화 도구 및 모범사례 확산
- **다중 클라우드 통합:** AWS/GCP 서비스를 단일 인터페이스로 관리하는 추상

1장 요약 및 실전 메시지

클라우드 컴퓨팅은 온디맨드 셀프서비스, 광범위한 네트워크 접근, 리소스 풀링, 신속한 확장성, 측정 가능한 서비스라는 핵심 특성을 갖습니다. 스타트업에게 클라우드는 초기 비용 없이 필요에 따라 확장 가능한 인프라를 구축할 수 있는 혁신적인 방법 입 니 다.

IaaS, PaaS, SaaS 서비스 모델과 **Public, Private, Hybrid** 배포 모델을 기업의 상황에 맞게 선택해야 합니다. AWS와 GCP의 선택은 서비스 구성, 비용 구조, 향후 확장성을 고려하여 결정해야 합니 다.



스타트업 실전 팁

- 초기 단계에서는 관리형 서비스(PaaS)로 시작해 빠르게 MVP 구축
- 비용 모니터링과 알림 설정은 서비스 론칭 전 필수 구성
- 지역 규제와 데이터 접근성을 고려한 리전 선택
- 초기부터 인프라as 코드(IaC) 방식 도입으로 반복 가능한 배포 구현

2장. AWS/GCP 계정 생성 및 초기 설정

클라우드 계정 가입, 보안 설정, 비용 관리 및 실습 환경 구축을 위한 스타트업 가이드

안전한 클라우드 환경 구축의 시작

학습 목표 및 기본 안내

이번 장에서는 스타트업 엔지니어가 **AWS**와 **GCP** 계정을 안전하게 생성하고 관리하는 방법을 학습합니다. 클라우드 계정 생성부터 비용 관리, 초기 리소스 설정까지 성공적인 클라우드 여정의 첫 단계를 다룹니다.

- **AWS/GCP** 계정 생성 - 스타트업에 최적화된 계정 등록 프로세스와 유의사항
- **IAM** 및 권한 관리 - 최소 권한 원칙을 적용한 안전한 계정 구조 설계
- 결제 설정 및 알림 - 예산 초과 방지를 위한 비용 모니터링 시스템 구축
- 다중 인증(**MFA**) - 루트 계정 및 IAM 사용자 보안 강화 전략
- 클라우드 리소스 태깅 - 효율적인 리소스 관리 및 비용 추적 방법

AWS 계정 생성 단계별 안내

AWS 계정을 처음 생성하는 스타트업 엔지니어들을 위한 단계별 가이드입니다. 계정 생성부터 최종 확인까지의 과정을 따라가며 주의해야 할 점과 Best Practice를 함께 소개합니다. 안전하고 효율적인 AWS 환경 구축의 첫 걸음을 시작해 보세요.

01 계정 정보 입력

- 업무용 이메일 주소 사용 (개인 메일 지양)
- 계정 이름은 명확한 비즈니스 목적 반영 (예: company-prod, company-dev)
- 강력한 비밀번호 설정 (최소 14자, 특수문자 포함)

02 연락처 정보 등록

- 회사 주소 정보 정확하게 입력 (결제 및 세금 관련)
- 비상 연락처로 접근 가능한 전화번호 사용 (보안 확인용)
- 가능하면 회사 대표 또는 공식 연락처 정보 사용

03 결제 정보 등록

- 회사 법인카드 사용 권장 (개인카드 지양) 비
- 용 알림 threshold 설정 (기본 80%, 100%)
- 청구서 수신 이메일 추가 설정 (재무/경영진 포함)
- 프리티어 사용 한도 사전 파악 (과금 방지)

04 신원 확인 및 계정 활성화

- SMS 또는 전화 인증 코드 확인
- 필요시 사업자등록증/신분증 제출 (특정 서비스 이용 시)
- 계정 생성 직후 MFA 활성화 필수
- 루트 계정 보안 체크리스트 확인

05 Best Practice 적용

- 루트 계정과 별도의 관리자 IAM 사용자 즉시 생성
- 루트 계정 액세스 키 생성 금지 (삭제 권장)
- 비용 예산 설정 및 AWS Budgets 활성화
- 리소스 태깅 정책 수립 (비용 할당 및 관리용)

powered by aws

AWS Root 계정 보안 설정

멀티 팩터 인증(MFA) 필수 활성화

Root 계정에 물리적 또는 가상 MFA 디바이스를 연결하여 계정 접근 시 추가 인증 요구

●●● 높은 위험도

Root 액세스 키 삭제

Root 사용자의 액세스 키를 생성하지 않거나, 이미 있다면 즉시 삭제하고 IAM 사용자 사용

●●● 높은 위험도

강력한 비밀번호 정책

최소 14자 이상, 특수문자, 대소문자, 숫자 조합 및 주기적 변경 정책 적용

●●○ 중간 위험도

최소 사용 원칙 적용

Root 계정은 극히 제한된 작업에만 사용하고, 일상적인 작업은 IAM 사용자로 수행

●●○ 중간 위험도

AWS Root 계정은 **절대적인 권한**을 가진 계정으로, 해킹 시 치명적인 피해가 발생합니다. 스타트업에서는 처음부터 철저한 보안 관리가 필수적입니다.

MFA(Multi-Factor Authentication)는 비밀번호 유출 시에도 추가 인증을 요구해 무단 액세스를 차단합니다. Root 사용자의 **액세스 키는 즉시 삭제**하고, 일상 작업은 적절한 권한이 부여된 IAM 사용자로 전환하세요.

AWS는 Root 계정 사용을 필요한 몇 가지 경우(계정 삭제, 결제 변경 등)로만 제한할 것을 권장합니다. 스타트업에서는 **계정 별칭 설정** 및 **결제 알림**을 함께 구성하여 안전한 클라우드 환경을 구축하세요.

"잠금장치는 정직한 사람들을 정직하게 유지합니다. *Root* 계정 보안은 스타트업의 첫 번째 보안 과제입니다."

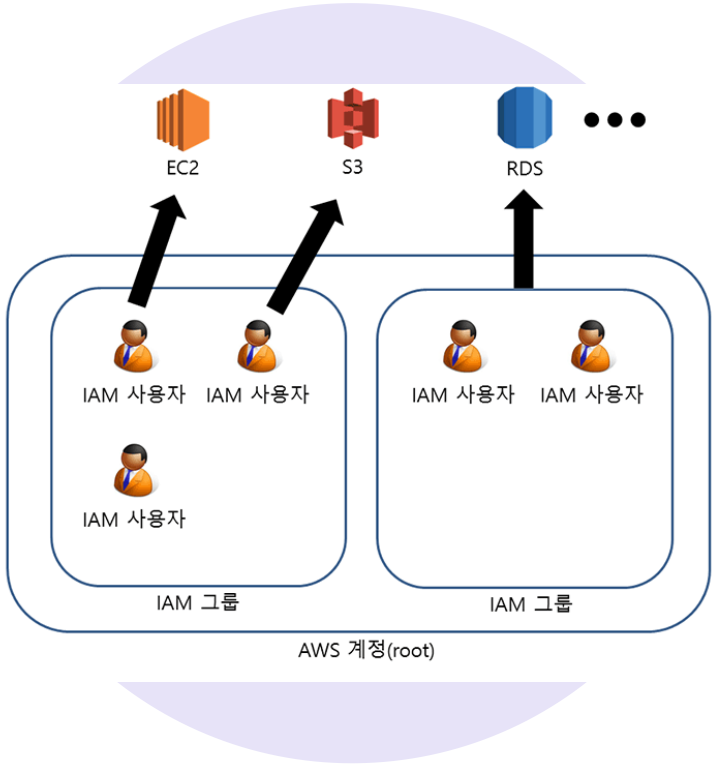
IAM 사용자 생성 및 관리

- 계정 분리 권장: 루트 계정은 초기 설정과 관리 목적으로만 사용하고, 일상적인 작업은 IAM 사용자 계정을 통해 진행해야 합니다. AWS와 GCP 모두 최소 권한 원칙(Principle of Least Privilege)에 따른 계정 생성을 권장합니다.
- 권한 할당 기준: 역할에 따른 권한 그룹화를 통해 효율적인 관리가 가능합니다.

역할	AWS	GCP
관리자	AdministratorAccess	Owner/Editor
개발자	PowerUserAccess	Editor + 제한
읽기전용	ReadOnlyAccess	Viewer

- 액세스 키 관리: API 호출 시 필요한 액세스 키는 노출되지 않도록 철저히 관리하고, 정기적으로 교체해야 합니다. 특히 GitHub 등 공개 저장소에 키를 업로드하지 않도록 주의해야 하며, 사용하지 않는 키는 즉시 비활성화하세요.

스타트업에서는 초기부터 체계적인 IAM 관리 정책을 수립하고, 프로젝트나 팀별 권한 분리를 통해 보안과 효율성을 동시에 확보하는 것이 중요합니다.



GCP 계정 생성 실전

Google Cloud Platform(GCP)은 Gmail 계정으로 쉽게 시작할 수 있습니다. 스타트업이 GCP를 처음 시작할 때는 계정 생성부터 프로젝트 설정, 그리고 무엇보다 중요한 결제 및 예산 관리까지 체계적으로 접근해야 합니다. 여기서는 안전하고 효율적인 GCP 초기 설정 방법을 단계별로 안내합니다.

스타트업 팁

GCP는 신규 가입 시 $300 무료 크레딧을 제공하며, 90일 동안 사용 가능합니다. 이 크레딧으로 실제 프로덕션 환경을 테스트해보세요.

01

Google 계정 로그인: 기존 Gmail 계정으로 로그인하거나 새 계정을 생성합니다. 가능하면 회사 도메인의 G Suite 계정 사용을 권장합니다. 개인 계정보다는 기업용 계정으로 시작하면 이후 관리가 용이합니다. Google Cloud Console에 처음 접속하면 이용약관에 동의하는 과정이 필요합니다.

02

GCP 프로젝트 생성: 모든 GCP 리소스는 프로젝트 내에서 관리됩니다. 프로젝트 ID는 글로벌하게 유일해야 하며 나중에 변경할 수 없습니다. 개발, 스테이징, 프로덕션 환경을 명확히 구분하기 위해 별도의 프로젝트를 생성하는 것이 좋습니다. 프로젝트 이름은 나중에 변경 가능하므로 명확한 네이밍 규칙을 적용하세요.

03

결제 계정 연결: GCP를 사용하려면 유효한 신용카드 정보가 필요합니다. 프로젝트에 결제 계정을 연결해야 하며, 하나의 결제 계정으로 여러 프로젝트를 관리할 수 있습니다. 스타트업은 특히 '결제 관리자' 역할을 별도 계정에 부여하는 것이 좋습니다. 결제 계정 설정 시 세금 정보와 실제 사업장 정보를 정확히 입력해야 합니다.

04

예산 설정 및 알림 구성: GCP Billing 메뉴에서 프로젝트별 예산을 설정할 수 있습니다. 예산 금액의 50%, 90%, 100%에 도달했을 때 이메일 알림을 받도록 구성하세요. 또한 특정 서비스나 리소스 라벨별로 세부 예산을 설정하면 비용 관리가 더 효율적입니다. Cloud Billing Reports를 활용하면 일별/월별 비용 추이를 그래프로 확인할 수 있어 비용 급증을 조기에 발견할 수 있습니다.

GCP IAM 및 서비스 계정

GCP의 **Identity and Access Management (IAM)**는 스타트업이 클라우드 리소스 에 대한 접근을 세밀하게 제어할 수 있도록 합니다. 기본적으로 세 가지 주요 역할(소유 자, 편집 자, 뷰어)과 함께 수백 개의 사전 정의된 역할을 제공하여 최소 권한 원칙을 구 현할 수 있습니다.

서비스 계정은 사용자가 아닌 애플리케이션과 워크로드를 위한 특별한 계정으로, 스타 트업의 자동화된 프로세스와 배포 파이프라인에 필수적입니다. 서비스 계정에 필요한 **JSON 키 파일**은 최고 수준의 보안으로 관리해야 하며, 가능한 경우 GCP의 키 없는 인 증 방식을 활용하는 것이 좋습니다.

스타트업은 성장 단계에 따라 팀 구성원의 역할을 명확히 구분하고, 필요 이상의 권한을 부여하지 않는 **최소 권한 원칙**을 적용해야 합니다. 또한 Cloud Audit Logs를 활성화하 여 누가 언제 어떤 작업을 수행했는지 추적하는 것이 중요합니다.

"적절한 IAM 설정은 스타트업의 보안 기반이자 효율적인 협업의 핵심입니다."

GCP IAM 역할 기반 사용자

소유자 (Owner)

- 모든 리소스에 대한 완전한 제어 권한
- 결제 관리 및 프로젝트 삭제 권 한
- IAM 권한 할당 및 변경

편집자 (Editor)

- 대부분의 리소스 생성 및 수정 가능
- 애플리케이션 배포 및 관리 권 한
- IAM 권한 변경은 불가

뷰어 (Viewer)

- 리소스 상태 및 구성 확인만 가능
- 데이터 읽기 전용 접 근
- 변경 작업 불가 (모니터링 팀에 적합)

서비스 계정 및 키 관리

애플리케이션이 GCP 리소스에 안전하게 접근하기 위한 특별 계 정

JSON 키 파일
서비스 계정 인증을 위한 비공개 키로,
안전한 보관과 관리가 필 수적

보안 주의사항: JSON 키는 절대 버전 관리 시스템이나 공개 저장소에 포함시키지 마세요. 키가 노출되면 즉시 폐기하고 새로운 키를 발급해야 합니 다.

40%

비용 관리 및 알림 설정

스타트업의 약 **40%**가 클라우드 비용 초과를 경험하며, 효과적인 비용 관리는 클라우드 도입의 핵심 과제입니다. AWS Cost Explorer와 GCP Billing은 비용 추적, 예산 설정, 알림 자동화를 통해 예상치 못한 비용 증가를 방지하고 최적화 기회를 식별하는 데 도움을 줍니다.

비용 알림 자동화는 실시간 모니터링과 사전 대응을 가능하게 하여, 스타트업이 제한된 예산 내에서 효율적으로 클라우드 리소스를 활용할 수 있도록 합니다. 적절한 비용 관리 도구를 활용하면 클라우드 지출을 최대 25% 절감할 수 있습니다.

AWS Cost Explorer

- AWS 예산 설정 및 임계값 알림
- 태그 기반 비용 추적
- 예측 기능으로 월별 지출 예상
- 비용 이상 탐지 기능 SMS/이
- 메일 알림 설정

GCP Billing

- 예산 알림 및 프로그래매틱 알림
- BigQuery 연동 분석
- 라벨 기반 비용 분석
- Pub/Sub 통합으로 자동화 설정
- Cloud Functions 연동 자동 대응

무료 티어 제한사항과 주의점

클라우드 서비스 제공업체들은 신규 사용자들이 서비스를 경험할 수 있도록 무료 티어를 제공합니다. 하지만 스타트업이 이를 활용할 때는 각 서비스별 제한사항을 명확히 이해해야 예상치 못한 비용 발생을 방지할 수 있습니다.

AWS 주요 무료 티어

- **EC2**: 750시간/월 t2.micro (Linux/Windows), 12개월 한정
- **S3**: 5GB 스토리지, 20,000 Get 요청, 2,000 Put 요청
- **RDS**: 750시간/월 db.t2.micro, 20GB 스토리지
- **Lambda**: 매월 100만 요청, 400,000GB-초 컴퓨팅 시간

주의: 무료 티어 만료 또는 초과 시 자동으로 유료 전환

GCP 주요 무료 티어

- **Compute Engine**: e2-micro 인스턴스 1개 (North America 리전)
- **Cloud Storage**: 5GB 표준 스토리지, 일정 수준의 작업
- **BigQuery**: 매월 1TB 쿼리 처리, 10GB 스토리지
- **Cloud Run**: 매월 2백만 요청, 360,000 vCPU-초, 180,000 GiB-초

주의: 일부 서비스는 무기한 무료, 일부는 **$300** 크레딧 소진까지

실무 팁: 예산 알림을 필수로 설정하고, 리소스 태깅을 통한 비용 추적, 자동 종료 스크립트 사용으로 의도치 않은 과금을 예방하세요.

리전/가용영역 실무 선택 가이드

비용, 가용성, 레이턴시, 규제 등 클라우드 리전 선정 기준 비교

비용 최적화

- **리전별 차이**: 북미/유럽 지역보다 아시아 리전이 최대 20% 비싸며, 특히 서울/도쿄/홍콩이 고비용
- **AWS**: 미국 버지니아(us-east-1)가 가장 저렴하고 신규 서비스 출시 우선
- **GCP**: 아이오와(us-central1)가 가장 저렴하고 안정적

가용성 및 복원력

- **멀티 AZ 배포**: AWS는 리전당 평균 3-6개 AZ, GCP는 리전당 3개 영역 제공
- **AWS**: 버지니아(us-east-1)는 가장 많은 6개 AZ 보유하나 장애 빈도도 높음
- **GCP**: 리전 간 네트워크 연결이 더 효율적으로 설계됨
- 스타트업 전략: 주요 워크로드는 최소 2개 AZ에 배포, 중요 데이터는 리전 간

레이턴시 및 성능

- **지리적 근접성**: 사용자와 가까운 리전 선택이 응답 속도 30-70ms 개선
- **AWS**: 글로벌 엣지 로케이션 300개 이상, CloudFront로 전 세계 최적화
- **GCP**: Premium Tier 네트워크로 리전 간 우수한 연결성 제공
- 스타트업 전략: 주 사용자층 기준 리전 선택, CDN 활용, 글로벌 확장 시 다중

규제 및 데이터 주권

- **데이터 상주 요구사항**: GDPR(EU), PIPA(한국), CCPA(캘리포니아) 등 지역별 규제 고려
- **AWS**: 더 많은 국가별 리전과 규제 인증 보유(IRAP, C5, FINMA 등)
- **GCP**: 더 투명한 데이터 처리 정책과 Assured Workloads 제공

실습 환경 체크리스트

클라우드 기술 학습과 실습을 위해서는 적절한 개발 환경이 필요합니다. 로컬 머신에 다음과 같은 도구와 환경을 구성하면 AWS/GCP 실습을 효율적으로 진행할 수 있습니다. 본 체크리스트를 통해 필요한 도구를 확인하고 설치하세요.

01

로컬 개발환경

- 명령줄 도구 (CLI): AWS CLI, gcloud CLI, Azure CLI
- 인프라 자동화: Terraform, Ansible
- 컨테이너화: Docker, Docker Compose, Kubernetes CLI
- 버전 관리: Git, GitHub CLI, GitLab CLI

02

IDE 및 편집기

- Visual Studio Code: AWS Toolkit, Cloud Code, HashiCorp Terraform
- JetBrains IDE: IntelliJ AWS Toolkit, PyCharm Google Cloud
- AWS Cloud9: 클라우드 기반 개발 환경
- Google Cloud Shell: 브라우저 기반 개발 환경

03

브라우저 확장 및 도구

- Chrome 확장: AWS Console Enhancer, GCP Console Helper
- API 테스트: Postman, Insomnia, cURL
- 모니터링: AWS CloudWatch, Google Cloud Monitoring
- 보안 점검: IAM Access Analyzer, Security Command Center

실전 보안 체크리스트

스타트업이 클라우드 환경에서 **보안 인시던트 위험을 최소화**하기 위한 필수 보안 체크리스트입니다. 비즈니스 초기 단계부터 이러한 보안 체계를 구축하면 향후 확장 과정에서 발생할 수 있는 대규모 보안 사고를 예방할 수 있습니다.

특히 **MFA 적용**과 **키 관리 자동화**는 최우선 순위로 구현하고, **리소스 태깅과 비용 모니터링**은 보안과 함께 비용 최적화에도 기여합니다. 모든 체크리스트 항목은 가능한 자동화하여 지속적인 보안 관리 부담을 줄이는 것이 중요합니다.

"보안은 개별 기능이 아닌 전체 개발 및 운영 문화의 일부가 되어야 합니다." — *AWS* 보안 모범 사례

다중 인증(MFA) 필수 적용 필수

모든 관리자 계정과 IAM 사용자에게 MFA 강제 적용. AWS의 경우 가상 MFA, 하드웨어 U2F, GCP의 경우 Google Authenticator 또는 타사 TOTP 앱 활용. 비활성화된 MFA를 정기적으로 감사하고 자동 알림 설정.

접근 키 관리 시스템화 중요

AWS 액세스 키와 GCP 서비스 계정 키 정기 교체(90일 주기), 미사용 키 자동 탐지 및 제거, 보안 저장소(Secret Manager/KMS)를 통한 중앙 관리, GitLab/GitHub 저장소에 키 업로드 방지 필터링 적용.

비용/리소스 모니터링 체계화 권장

모든 리소스에 소유자/프로젝트/환경 태그 필수화, AWS Budget/GCP Budget Alert 설정, 이상 징후 실시간 알림, CloudWatch/Cloud Monitoring 대시보드 구성, 미사용 리소스 자동 감지 및 종료 자동화.

네트워크 보안 강화 중요

VPC 환경 내 보안 그룹/방화벽 규칙 최소화, 퍼블릭 접근 포트 제한, WAF 적용, 외부 통신 엔드포인트 암호화(TLS 1.2+), API 게이트웨이 인증 필수화, 정기적인 취약점 스캔 자동화.

실무 팁: 계정 분리와 운영 정책

개인/업무/시스템 계정 분할

스타트업에서도 계정 분리는 필수입니다. 개발자 개인 계정, 공용 업무 계정, 시스템 자동화용 계정을 분리하여 책임 소재를 명확히 하고 보안 위험을 최소화하세요. AWS Organizations와 GCP 프로젝트 구조를 활용하면 효과적입니다.

최소 권한 원칙 적용

모든 계정에 필요한 최소한의 권한만 부여하세요. AWS IAM 정책과 GCP IAM 역할을 세분화하여 설정하고, 정기적으로 미사용 권한을 제거합니다. 개발/테스트/운영 환경별로 권한을 분리하면 실수로 인한 장애를 방지할 수 있습니다.

감사 로그 병행 설정

모든 클라우드 활동을 기록하고 모니터링하세요. AWS CloudTrail과 GCP Cloud Audit Logs를 활성화하고, 중요 활동에 대한 알림을 설정합니다. 로그는 최소 90일 이상 보관하고, 보안 이벤트 분석을 위한 대시보드를 구축하면 위협에 신속하게 대응할 수 있습니다.

문제 해결: 계정 인증 오류와 리커버리

클라우드 계정 액세스 문제는 스타트업에게 심각한 비즈니스 중단을 초래할 수 있습니다. **계정이 잠기거나**, **결제 이슈가 발생하거나**, **보안 인증 정보를 분실했을 때** 체계적인 해결 방법을 알아두는 것이 중요합니다.

AWS와 GCP 모두 **24/7 복구 지원 시스템**을 갖추고 있지만, 대응 시간을 줄이기 위해서는 적절한 문의 채널과 필요한 정보를 사전에 준비해야 합니다. 특히 **MFA 관련 이슈**는 별도의 검증 절차가 필요하므로 백업 코드를 안전하게 보관하는 습관이 중요합니다.

결제 이슈로 인한 계정 정지는 **선결제 크레딧**이나 **백업 결제 수단 등록**으로 예방할 수 있으며, 중요 서비스를 운영 중인 경우 비즈니스 지원 계획 업그레이드도 고려해볼 수 있습니다.

"클라우드 계정은 디지털 비즈니스의 열쇠입니다. 항상 2개 이상의 관리자 계정을 안전하게 유지하세요."

1. 문제 식별

정확한 에러 메시지와 코드를 기록하세요. 몇 가지 일반적인 케이스:

- 인증 실패: 비밀번호 5회 이상 오류
- 계정 정지: 결제 실패 또는 정책 위반
- MFA 장치 분실 또는 액세스 불가

AWS GCP

2. 본인 인증 준비

계정 복구를 위해 다음 정보를 준비하세요:

- 계정 생성 시 사용한 이메일
- 최근 결제 정보 및 결제 카드 세부정보
- 계정과 연결된 전화번호
- 최근에 생성한 리소스 목록

3. 지원팀 연락

적절한 채널을 통해 연락하세요:

- **AWS:** 지원 센터 > 계정 및 결제 > 사례 생성
- **GCP:** 지원 > 새 사례 > 계정 & 결제
- 비즈니스 고객은 기술지원 계정관리자 연락

4. 복구 후 조치

계정 복구 후 필수 보안 단계:

- 강력한 새 비밀번호 설정
- MFA 재설정 또는 신규 등록
- 비정상 활동이 없는지 활동 로그 검토
- 결제 정보 및 알림 확인 및 재설정

실전 사례 소개: 유망스타트업의 Secure Onboarding

핀테크 스타트업 **SecurePayKorea**의 클라우드 인프라 구축 사례를 통해 알아보는 안전하고 체계적인 클라우드 온보딩 과정입니다. 이 스타트업은 결제 서비스를 제공하므로 높은 수준의 보안이 필요했지만, 초기 단계에서는 제한된 리소스로 효율적인 설정이 필요했습니다.

1단계: 분리된 계정 구조 설계 (1주 차)

운영/개발/테스트 환경을 위한 별도 AWS Organizations 구성, 멀티 계정 전략 도입

2단계: 최소 권한 IAM 정책 구현 (2주 차)

직무별 권한 세분화, 서비스 계정 분리, 임시 보안 인증 정보 활용

3단계: 자동화된 보안 모니터링 (3주 차)

AWS CloudTrail, Config, GuardDuty를 Slack 알림과 연동, 이상 행동 실시간 탐지

4단계: 환경 승인 프로세스 (4주 차)

CISO 주도 보안 검토회의, 자동화된 보안 스캔, 통과 후 프로덕션 배포 승인

3장. CLI 설치 및 인증

AWS CLI, gcloud 등 자동화 환경 구축 및 인증 프로세스 통합 가이드

클라우드 운영 자동화의 핵심

학습 목표 & CLI 도입 혜택

명령줄 인터페이스(CLI)는 클라우드 자동화와 효율적인 리소스 관리의 핵심 도구입니다. CLI를 통해 스타트업 엔지니어들은 반복 작업을 자동화하고, 인프라를 코드로 관리하며, 협업 워크플로우를 최적화할 수 있습니다.

- 자동화 역량 강화: 스크립트를 활용한 AWS와 GCP 리소스 생성, 배포 및 모니터링 자동화
- 비용 절감 효과: CLI 자동화로 평균 운영 비용 43% 절감 및 업무 효율 37% 향상 (2025 DevOps 보고서)
- 협업 및 버전 관리: 인프라 코드로 관리하여 팀 협업과 버전 제어 용이
- 멀티 클라우드 일관성: 동일한 스크립트와 패턴으로 AWS/GCP 환경 동시 제어
- 복잡한 배포 단순화: 배포 파이프라인 구축과 연동으로 반복 가능한 인프라 관리

AWS CLI 설치 단계별 설명

AWS CLI(Command Line Interface)는 명령줄 쉘에서 AWS 서비스와 상호 작용할 수 있는 오픈 소스 도구입니다. 모든 주요 운영 체제에 설치가 가능하며, 자동화 스크립트 및 배포 파이프라인에 필수적입니다. 최신 버전(v2)을 설치하여 모든 기능과 보안 업데이트를 활용하세요.

01

Windows 설치 방법

1. AWS CLI MSI 설치 프로그램을 다운로드합니다.
2. 다운로드한 MSI 파일을 실행하고 설치 마법사의 안내에 따릅니다.
3. 설치 확인: `aws --version` 명령어로 확인합니다.
4. PowerShell에서 명령어 완성 기능을 활성화하려면: `Install-Module -Name AWS.Tools.Installer`

02

macOS 설치 방법

1. Homebrew 사용 시: `brew install awscli`

2. 수동 설치 시: macOS pkg 파일 다운로드 후 설치

```
curl "https://awscli.amazonaws.com/AWSCLIV2.pkg" -o "AWSCLIV2.pkg" sudo installer -pkg AWSCLIV2.pkg -target /
```

3. 설치 확인: `aws --version` 명령어로 확인합니다.

03

Linux 설치 방법

1. ZIP 파일 다운로드 및 압축 해제:

```
curl "https://awscli.amazonaws.com/awscli-exe-linux-x86_64.zip" -o "awscliv2.zip"
unzip awscliv2.zip
```

2. 설치 스크립트 실행:

```
sudo ./aws/install
```

3. 패키지 관리자 사용 시 **(Ubuntu):**

```
sudo apt-get update && sudo apt-get install awscli -y
```

04

설치 후 구성 및 유의사항

1. 환경 변수 확인: `echo $PATH`로 실행 경로 확인
2. 최신 버전 업데이트: 각 OS별 업데이트 명령어로 최신 상태 유지
3. 오류 발생 시: 관리자 권한(sudo) 필요 여부 확인
4. **AWS** 지원 페이지: 문제 발생 시 AWS CLI 공식 문서 참조

환경 변수 및 인증 설정

AWS 구성 파일 예시 ~/.aws/

 # ~/.aws/credentials 파일

```
[default]
aws_access_key_id = AKIAIOSFODNN7EXAMPLE
aws_secret_access_key =
wJalrXUtnFEMI/K7MDENG/bPxRfiCYEXAMPLEKEY

[dev-profile]
aws_access_key_id = AKIAI44QH8DHBEXAMPLE
aws_secret_access_key =
je7MtGbClwBF/2Zp9Utk/h3yCo8nvbEXAMPLEKEY
```

 # ~/.aws/config 파일

```
[default]
region = ap-northeast-2 output =
json

[profile dev-profile] region =
us-west-2 output = text
role_arn = arn:aws:iam::123456789012:role/DevRole source_profile =
default
```

1 `aws configure` 명령어로 **기본 설정**을 구성합니다

AWS CLI 구성은 **credentials**와 **config** 두 파일을 통해 관리됩니다. 이는 ~/.aws/ 디렉토리에 저장되며 사용자 인증 정보와 기본 설정을 담고 있습니다.

4 `aws --profile dev-profile s3 ls` 와 같이 **프로필별 명령 실행**이 가능합니다

여러 AWS 계정을 관리해야 하는 스타트업 개발자라면 **다중 프로필** 설정이 필수입니다. 개발, 테스트, 프로덕션 환경을 분리하여 안전하게 관리할 수 있으며, 환경 변수 **AWS_PROFILE**로 기본 프로필을 전환할 수 있습니다.

"효율적인 *AWS CLI* 설정은 복잡한 클라우드 환경에서의 자동화와 보안의 기본이 됩니다."

Google Cloud CLI는 GCP 리소스를 관리하는 명령줄 도구로, 다양한 운영체제에서 설치 가능합니다. 기본 명령어인 **gcloud**를 통해 대부분의 Google Cloud 서비스를 제어할 수 있습니다.

Windows 설치

Google Cloud SDK 설치 프로그램을 다운로드하여 실행하거나, PowerShell에서 다음 명령어로 설치:

```
(New-Object
Net.WebClient).DownloadFile("https://dl.google.com/dl/cloudsdk/channels/rap
"$env:Temp\GoogleCloudSDKInstaller.exe") &
$env:Temp\GoogleCloudSDKInstaller.exe
```

macOS 설치

Homebrew를 통한 설치 방법:

```
brew install --cask google-cloud-sdk
```

Linux 설치

Debian/Ubuntu에서 apt를 통한 설치:

```
echo "deb [signed-by=/usr/share/keyrings/cloud.google.gpg]
https://packages.cloud.google.com/apt cloud-sdk main" | sudo tee -a
/etc/apt/sources.list.d/google-cloud-sdk.list && curl
https://packages.cloud.google.com/apt/doc/apt-key.gpg | sudo apt-key -- keyring
/usr/share/keyrings/cloud.google.gpg add - && sudo apt-get update && sudo apt-get
install google-cloud-cli
```

초기화 자동화 팁: gcloud init --console-only 명령어를 스크립트에 포함하여 브라우저 인증 과정 없이 초기화하거나, --skip-diagnostics 옵션을 추가하여 진단 과정을 생략할 수 있습니다. 서비스 계정 키를 활용한 gcloud auth activate-service-account -- key-file=KEY_FILE 명령으로 인증을 자동화할 수 있습니다.

GCP CLI 인증 방식

Google Cloud Platform(GCP)의 커맨드 라인 도구인 gcloud CLI는 여러 인증 방식을 지원합니다. 각 인증 방식은 다른 사용 사례와 보안 요구 사항에 맞게 설계되어 있어, 스타트업의 상황과 목적에 맞는 인증 방식을 선택하는 것이 중요합니다. 이 슬라이드에서는 주요 인증 방식의 차이점과 실전 활용법을 알아봅니다.

01

사용자 계정 인증: 개발자 및 관리자가 직접 사용하는 방식으로, Google 계정으로 인증합니다. `gcloud auth login` 명령어를 실행하면 브라우저에서 Google 로그인 페이지가 열리고, 로그인 성공 시 로컬 시스템에 인증 토큰이 저장됩니다. 개인 개발 환경이나 임시 작업에 적합하지만, 자동화 프로세스에는 부적합합니다.

02

서비스 계정 인증: 애플리케이션 및 자동화 프로세스를 위해 설계된 비인간 계정입니다. 특정 역할과 권한이 할당되며, 사람이 아닌 시스템에서 GCP 리소스에 접근할 때 사용합니다. `gcloud auth activate-service-account` 명령으로 활성화하며, 자동화된 CI/CD 파이프라인, 서버리스 함수 등에 이상적입니다.

03

JSON 키 파일 활용: 서비스 계정을 위한 인증 방법으로, 키 파일에는 민감한 인증 정보가 포함됩니다. `gcloud auth activate-service-account --key-file=[KEY_FILE]`로 사용합니다. 보안을 위해 키 파일은 안전하게 저장하고, 주기적으로 교체해야 합니다. 키 관리 모범 사례로는 비밀 관리 서비스 (Secret Manager)나 환경 변수 사용을 권장합니다.

04

인증 관리 명령어: `gcloud config list`는 현재 구성된 설정 전체를 보여주며, 프로젝트 ID, 리전, 계정 등의 정보를 확인할 수 있습니다. `gcloud auth list`는 현재 시스템에 구성된 모든 인증된 계정을 보여주고 활성 계정을 표시합니다. `gcloud auth revoke`로 인증을 취소하고, `gcloud config configurations create [NAME]`으로 별도의 설정 프로필을 만들 수 있습니다.

AWS CLI 주요 명령어

S3

```
$ aws s3 ls
```

모든 S3 버킷 나열

```
$ aws s3 cp file.txt s3://bucket-name/
```

파일 업로드

EC2

```
$ aws ec2 describe-instances
```

모든 EC2 인스턴스 정보 조회

IAM

```
$ aws iam list-users
```

사용자 목록 조회

CloudFormation

```
$ aws cloudformation deploy \ --template-file template.yaml \
--stack-name my-stack
```

스택 배포

CloudWatch Logs

```
$ aws logs describe-log-groups
```

로그 그룹 조회

CLI 주요 명령어 요약

클라우드 CLI는 클라우드 리소스를 효율적으로 관리하고 자동화하기 위한 필수 도구입니다. AWS CLI와 GCP의 gcloud CLI는 각각 서로 다른 명령 구조를 가지고 있지만, 기본 사용법은 유사합니다.

AWS CLI는 'aws' 뒤에 서비스명과 명령을 입력하는 구조입니다. 예를 들어 'aws s3 ls'는 S3 버킷을 나열합니다. **GCP CLI**는 'gcloud' 뒤에 서비스 그룹과 명령을 입력하거나, 특수 서비스('gsutil')를 사용합니다.

스타트업이 효율적으로 클라우드 환경을 관리하려면 기본 명령어들을 숙지하고, 반복 작업은 스크립트로 자동화하는 것이 좋습니다. 또한 '--help' 플래그를 활용해 CLI 사용법을 빠르게 참조할 수 있습니다.

"CLI 활용 능력은 클라우드 엔지니어의 생산성을 결정짓는 가장 중요한 요소 중 하나입니다."

GCP CLI 주요 명령어

Compute Engine

```
$ gcloud compute instances list
```

모든 VM 인스턴스 나열

Cloud Storage

```
$ gsutil ls
```

모든 버킷 나열

Cloud SQL

```
$ gcloud sql instances list
```

SQL 인스턴스 나열

IAM

```
$ gcloud iam roles list
```

IAM 역할 목록 조회

Logging

```
$ gcloud logging logs list
```

로그 목록 조회

CLI 주요 명령어 요약

클라우드 CLI는 클라우드 리소스를 효율적으로 관리하고 자동화하기 위한 필수 도구입니다. AWS CLI와 GCP의 gcloud CLI는 각각 서로 다른 명령 구조를 가지고 있지만, 기본 사용법은 유사합니다.

AWS CLI는 'aws' 뒤에 서비스명과 명령을 입력하는 구조입니다. 예를 들어 'aws s3 ls'는 S3 버킷을 나열합니다. **GCP CLI**는 'gcloud' 뒤에 서비스 그룹과 명령을 입력하거나, 특수 서비스('gsutil')를 사용합니다.

스타트업이 효율적으로 클라우드 환경을 관리하려면 기본 명령어들을 숙지하고, 반복 작업은 스크립트로 자동화하는 것이 좋습니다. 또한 '--help' 플래그를 활용해 CLI 사용법을 빠르게 참조할 수 있습니다.

"CLI 활용 능력은 클라우드 엔지니어의 생산성을 결정짓는 가장 중요한 요소 중 하나입니다."

고급 CLI 기능 & 자동화

클라우드 인프라 자동화와 효율적인 관리를 위해 **AWS CLI**와 **GCP gcloud**의 고급 기능을 활용할 수 있습니다. 이러한 기능들은 스크립트 작성과 데이터 처리에 필수적인 요소입니다.

AWS CLI 고급 기능

- **JMESPath**: `--query` 옵션으로 복잡한 JSON 데이터 필터링
- 출력 포맷: `--output json|yaml|text|table` 다양한 형식 지원
- 페이지네이션: `--page-size`, `--max-items` 대용량 데이터 효율적 처리

GCP gcloud 고급 기능

- 필터링: `--filter="name=value"` 복잡한 조건식 지원
- 정렬: `--sort-by="~createTime"` 원하는 필드로 정렬
- 자동완성: `gcloud completion` 으로 쉘 자동완성 설정

이러한 고급 기능들을 결합하면 배포 자동화, 리소스 모니터링, 비용 최적화 등 스타트업의 DevOps 워크플로우를 크게 개선할 수 있습니다.

IAM 정책 및 샘플정책

AWS JSON 정책 효율화

AWS 정책은 최소 권한 원칙을 따르되, 태그 기반 정책(ABAC)을 활용해 관리 효율성을 높일 수 있습니다. S3 버킷별 정책 대신 태그로 분류하면 정책 수를 크게 줄일 수 있습니다.

```
{ "Effect": "Allow", "Action": "s3:*", "Resource": "*", "Condition": {"StringEquals": {"aws:ResourceTag/Environment": "${aws:PrincipalTag/Environment}"}} }
```

GCP IAM 역할 최적화

Google Cloud의 커스텀 역할을 사용하면 세밀한 권한 제어가 가능합니다. 기본 역할(Owner, Editor, Viewer)은 과도한 권한을 부여할 수 있으므로, 프로젝트별로 필요한 최소 권한만 설정하세요.

```
title: "Custom DevOps Role" description: "최소 권한을 가진 CI/CD 배포 전용 역할" permissions: - compute.instances.get - container.deployments.create
```

CLI 활용 스크립트 예제

AWS CLI와 GCP gcloud CLI를 활용한 자동화 스크립트 예제들을 통해 실무에서 바로 적용 가능한 자동화 방법을 알아보겠습니다. 실제 스타트업 환경에서 유용하게 활용될 수 있는 스크립트를 중심으로 소개 합니 다.

Bash로 AWS 리소스 백업

```
#!/bin/bash
# S3 버킷 백업 스크립트
BUCKET="my-startup-backup"
DATE=$(date +%Y-%m-%d)
aws s3 sync /data s3://$BUCKET/$DATE/data
```

PowerShell로 모니터링 자동화

```
# EC2 인스턴스 상태 모니터링
$instances = aws ec2 describe-instances
| ConvertFrom-Json
$instances.Reservations.Instances
| Select-Object InstanceId, State
```

Python/boto3 간단 연계

```
import boto3
# DynamoDB 데이터 조회
dynamodb = boto3.resource('dynamodb')
table = dynamodb.Table('users') response =
table.scan()
for item in response['Items']: print(item)
```

문제 해결: CLI 인증/디버깅

1. 문제 식별

AWS --debug 플래그 사용
GCP --verbosity=debug 사용

2. 구성 확인

AWS ~/.aws/credentials, ~/.aws/config 확인
GCP gcloud config list, gcloud auth list

3. 권한 확인

AWS aws sts get-caller-identity
GCP gcloud auth print-identity-token

4. 인증 갱신

AWS 새 액세스 키 생성 또는 IAM 역할 재설정
GCP gcloud auth login 또는 서비스 계정 키 재발급

CLI 작업 시 발생하는 인증 및 권한 문제는 스타트업의 개발/운영 흐름을 저해할 수 있습니다. **체계적인 문제 진단**과 **효율적인 해결 방법**을 통해 이러한 장애를 최소화할 수 있습니다.

AWS CLI | GCP gcloud

AWS CLI 인증 문제 해결:

- **AccessDenied 오류:** IAM 권한 확인 → aws iam get-user 명령으로 현재 사용자 검증
- **토큰 만료:** aws sts get-session-token으로 새 토큰 발급
- **MFA 인증 필요:** aws sts get-session-token --serial-number MFA_DEVICE --token-code CODE
- **프로파일 문제:** --profile 파라미터로 올바른 프로필 지정

```
$ export AWS_DEFAULT_REGION=ap-northeast-2
$ aws configure set default.output json
$ aws s3 ls --debug 2> debug.log
```

CLI 실습 시나리오

계정 로그인부터 리소스 관리까지 단계별 실습 가이드

01

환경 설정 및 로그인

계정 인증 정보를 설정하고 CLI로 로그인하는 기초 과정을 실습합니다.

- AWS
- aws configure
- GCP
- gcloud auth login

02

리소스 조회 및 탐색

기존 클라우드 리소스 목록을 조회하고 상세 정보를 확인하는 방법을 실습합니다.

- aws ec2 describe-instances
- gcloud compute instances list
- 필터링

03

리소스 생성 및 관리

CLI 명령으로 새 리소스를 프로비저닝하고 관리하는 방법을 학습합니다.

- aws s3 mb s3://bucket-name
- gcloud storage buckets create
- JSON/YAML 템플릿

04

정보 저장 및 자동화

CLI 출력을 파일로 저장하고 간단한 스크립트로 작업을 자동화하는 방법을 학습합니다.

- 리다이렉션 > file.json
- --output formats
- jq
- Bash/PowerShell

사용자별 CLI 환경 분리/보안 팁

클라우드 운영 시 CLI 도구는 강력한 권한을 가지므로, 적절한 환경 분리와 보안 설정이 필수적입니다. 다양한 팀원들이 안전하게 CLI를 사용할 수 있도록 프로필 관리, 인증 토큰 보호, 접근 제어를 체계적으로 구성해야 합니다.

01

프로필 별도 관리: AWS CLI에서는 `--profile` 옵션으로 여러 계정 프로필을 분리하세요. 개발, 테스트, 프로덕션 환경별로 프로필을 구분하고, credentials 파일에 [profile-name] 형식으로 지정합니다. GCP에서는 `gcloud config configurations create` 명령어로 여러 구성을 분리하고 `gcloud config configurations activate` 로 전환하세요.

02

토큰/키 노출 예방: AWS 액세스 키와 GCP 서비스 계정 키는 절대 소스 코드나 공개 저장소에 포함하지 마세요. 대신 `AWS Secrets Manager` 나 `Google Secret Manager` 를 활용하세요. Git에서는 `.gitignore` 에 키 파일을 추가하고, 실수로 커밋된 키는 즉시 삭제하고 새 키로 교체해야 합니다. CI/CD 파이프라인에서는 환경 변수를 통해 안전하게 주입하세요.

03

MFA 연동 사례: AWS CLI에서 `aws sts get-session-token` 으로 MFA 토큰을 사용한 임시 자격 증명을 발급받아 사용하세요. 자동화 스크립트에서는 `assume-role` 로 시간 제한이 있는 권한을 활용합니다. GCP에서는 `gcloud auth login` 시 자동으로 2단계 인증이 적용됩니다. 서비스 계정 키 대신 `Workload Identity Federation` 을 사용해 외부 ID와 연동하는 것이 더 안전합니다.

04

권한 유효 시간 제한: 장기 유효한 액세스 키 대신, AWS에서는 `STS(Security Token Service)` 를 사용해 최대 12시간의 임시 토큰을 사용하세요. GCP에서는 `gcloud auth application-default login` 으로 얻는 토큰이 1시간 후 만료됩니다. 민감한 작업을 위해 권한 상승이 필요한 경우에는 명시적 승인 프로세스를 구현하고 권한 사용 후 즉시 해제하는 시스템을 도입하세요.

3장 요약 및 실전 활용법

CLI로 자동화 효율성 극대화

- 재현 가능한 인프라 - 수동 작업 대비 일관성과 반복성 확보 배
- 치 처리 - 대량의 클라우드 리소스를 스크립트로 한번에 관리
- 버전 관리 - 인프라 변경 사항을 Git과 통합하여 추적 및 롤백

DevOps 협업 기반 구축

- **CI/CD** 파이프라인 - AWS CLI/gcloud를 GitHub Actions, Jenkins와 통합
- **Infrastructure as Code** - CLI 스크립트를 Terraform, CloudFormation과 결합
- 비용 모니터링 - 자원 사용량을 자동으로 추적하여 낭비 방지

스타트업을 위한 CLI 실전 전략

- 시작은 필수 명령어 10개로 간단하게, 점진적으로 확장 환
- 경별 프로필 분리로 개발/테스트/프로덕션 안전하게 관리
- 팀 전체의 CLI 활용 가이드 문서화로 지식 공유

추가 학습 가이드

지금까지 클라우드 컴퓨팅 기본 개념, AWS/GCP 계정 설정, CLI 도구 활용에 대해 살펴 보았습니다. 다음 장에서는 보다 심화된 내용을 다루게 됩니다.

4장. 클라우드 핵심 서비스 개념

VPC, 객체 스토리지(S3/Cloud Storage), 가상머신(EC2/Compute Engine) 등 핵심 서비스의 아키텍처와 설계 원칙을 이해합니다.

5장. 컴퓨팅 서비스 비교

EC2 vs Compute Engine, Lambda vs Cloud Functions 성능 및 확장성, 비용 최적화 전략을 비교 분석합니다.

6장. 스토리지 서비스 비교

S3 vs Cloud Storage의 스토리지 클래스, 가격 구조, 성능 및 보안 기능을 심도 있게 학습합니다.

7장. 데이터베이스 서비스 비교

RDS vs Cloud SQL의 지원 엔진, 성능, 가용성 및 마이그레이션 전략을 탐구합니다.

8장. DevOps 및 운영 관리

IaC, CI/CD 파이프라인, 컨테이너 서비스 및 모니터링 도구를 활용한 스타트업 환경의 클라우드 운영 모범 사례를 소개합니다.

전체 요약 및 실전 메시지

1. 클라우드 컴퓨팅 기초

- **IaaS, PaaS, SaaS** 모델을 비즈니스 요구사항에 맞게 선택하고, 초기 스타트업은 서버리스 아키텍처로 비용 효율성을 높이세요.
- 클라우드 도입은 단계적 접근이 중요합니다. PoC → 베타 → 프로덕션 → 글로벌 확장 로드맵을 따르세요.

2. 계정 및 보안 관리

- 루트 계정 보호는 필수 - MFA 설정, 액세스키 삭제, 일상 작업은 IAM 계정 사용을 철저히 준수하세요.
- 리전/가용영역 선택 시 비용, 지연시간, 규제 세 요소를 함께 고려하세요.

3. 자동화와 효율성

- AWS CLI와 GCP gcloud로 반복 작업 자동화하고, 인프라를 코드로 관리(IaC)하여 일관성을 유지하세요.
- 비용 모니터링 알림과 예산 설정은 클라우드 운영의 기본입니다. 자동 스케일링으로 비용 최적화하세요.
- 스타트업 성장 단계에 따라 관리형 서비스에서 시작하여 점차 맞춤형 인프라로 발전시키는 전략이 효과적입니다.

감사합니다!

http://www.xxx.xxx.xxx

lnhwan.jung@gmail.com
github.com/jungfrau70
정인환 강사/ 클라우드 아키텍트 & 컨설턴트