클라우드 플랫폼에서 Docker Container의 효율적인 CI/CD Pipeline 설계

CI/CD 오픈소스 장단점 및 사용법 조사

- Jenkins/GitLab 과 AWS/GCP를 중심으로 -

기업명 : 맨텍

팀장 : 강민경

조원 : 김정식, 양승호, 이재성, 전예림

목차

Ⅰ. 개요

기존에 사용하던 Waterfall 개발 방식은 병행해서 작업을 하기 어렵고 중간에 발생하는 사용자 요구사항에 대해 대처하기 힘들다는 단점이 존재했다. 이러한 단점을 보완한 개발 방식이 DevOps이다. DevOps는 개발(development)과 운영(operation)을 결합한 혼성어로 개발 담당자와 운영 담당자가 연계하여 협력하는 개발 방법론이다. DevOps는 CI/CD 개념과 관련이 있는데, CI는 Continuous Integration, CD는 Continuous Delivery로 지속적인 통합, 지속적인 배포를 의미한다. 이 CI/CD를 이용하게 되면 아래와 같은 장점을 가지게 된다.

1. 빌드와 테스트 프로세스가 자동화되어 코드 작성에 전념할 수 있다.
2. 자동화를 통해 변경사항을 수시로 통합할 수 있으며, 이를 통해 문제를 조기에 발견하고 조치할 수 있다.
3. 빌드와 테스트를 개인 환경과 독립적으로 구성할 수 있다.
4. 빌드한 결과를 톰캣이나 제티 등 기타 환경에 배포하는 업무를 자동화함으로써 프로세스를 간소화 할 수 있다.
5. 다른 개발자가 수정한 내용을 자동으로 빌드하고 통합 테스트를 진행할 수 있다.

이러한 강력한 장점을 가진 CI/CD를 이용하기 위해 현재 시장에는 다양한 CI/CD 도구가 오픈소스 기반으로 제공되어 있다. 이제 그 중에서도 많이 사용되는 jenkins, gitlab, gcp, aws의 간단한 사용법, 특징, 제공되는 서비스, 도구 그리고 장단점 조사를 통해서 어떤 CI/CD 오픈소스가 더 효율적인지 알아볼 것이다.

Ⅱ. 프로젝트 의의

1. 프로젝트의 목적

요즘 DevOps가 굉장히 뜨고 있다. DevOps란 소프트웨어 개발팀과 IT 팀이 더 빠르고 안정적으로 소프트웨어를 빌드, 테스트 및 릴리스할 수 있도록 두 팀 간의 프로세스를 자동화하는 일련의 과정이다. 이러한 장점 때문에 많은 업무가 CI/CD를 포함 하고 있다. 따라서 앞으로 개발자들도 CI/CD를 잘 알아야 하기 때문에 이에 필요한 jenkins, gitlab, gcp, aws 같은 CI/CD Open Source 자료조사 및 장단점 분석을 진행 중이다.

2. 기대효과

이 프로젝트를 진행함으로써 CI/CD의 개념에 대해 이해하고, CI/CD 오픈소스에서 CI 환경을 구축하기 위해 필요한 기초 설정 및 다운로드하는 법, 스크립트를 작성하는 법 등을 배우고, 이를 통해 각각의 오픈소스에서 CI/CD 환경을 구축할 수 있게 되고, 구축한 환경을 통해서 CPU 사용량, RAM 사용 등을 모니터링 함으로써 각 오픈소스의 성능, 메모리 사용량, 가격, UI, 편의성 등 여러 가지 정보를 얻고, 최종적으로 이를 비교함으로써 어떤 상황에 어떤 오픈소스를 사용하는 것이 효율적인지, 주의할 점은 무엇인지에 대한 정보를 얻음으로써 각 오픈소스를 상황에 맞게 적재적소에 사용할 수 있게 될 것이다.

Ⅲ. CI/CD 오픈소스

1. Jenkins

◇ 개요

- 허드슨의 주요 개발자가 나와서 만든 프로젝트로, 오픈소스 CI 솔루션 가운데 압도적인 사용자를 보유하고 있다.
- 사용자 및 플러그인 개발자가 많아서 다양한 기능이 제공되며, 관련 자료도 많은 편이다.
- 거의 모든 언어의 조합과 소스코드 리포지토리에 대한 지속적인 통합과 지속적인 전달 환경을 구축하기 위한 간단한 방법을 제공한다.

◇ 특징

1. 자동화

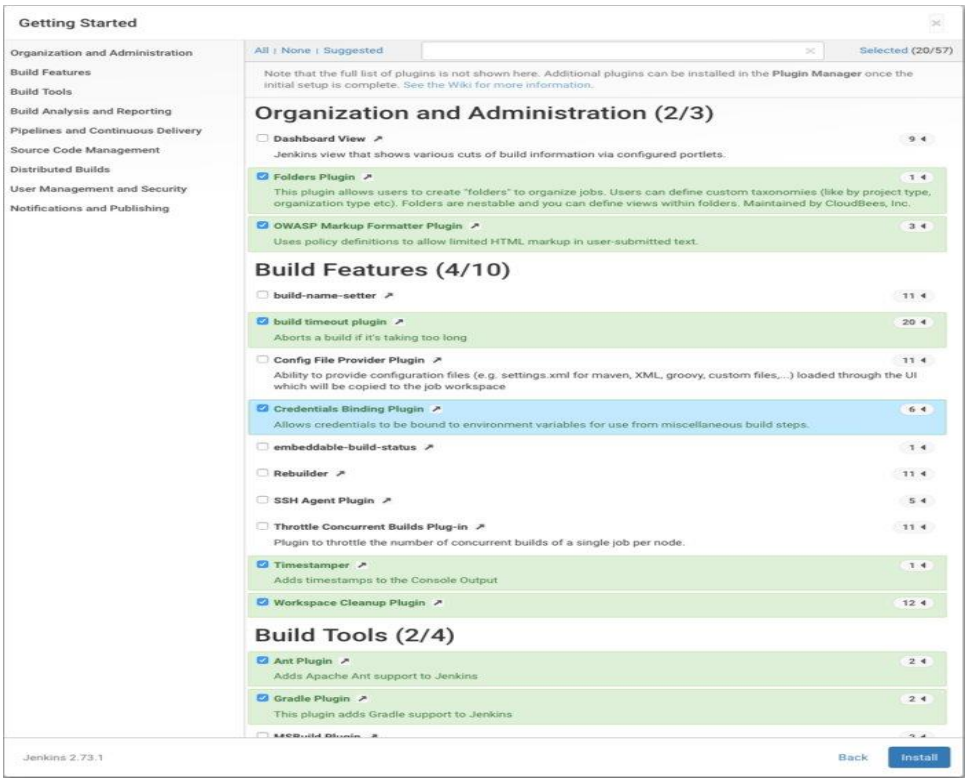
젠킨스는 약 1400가지의 플러그인을 가지고 있는 오픈소스 자동화 서버이다. 지속적인 통합, 지속적인 자바코드 전달 (프로젝트 빌드), 테스트 실행, 정적 코드 분석 시행, 배포 작업에 더불어 많은 프로세스들이 제공된다. 1400개의 플러그인은 5가지 영역을 포괄하고 있다. (플랫폼, UI, 관리, 소스코드 관리, 빌드 관리)

2. 동작 방식

젠킨스는 운영체제용 자바 8 WAR 아카이브와 설치 패키지, 홈브루 패키지, 도커 이미지, 소스코드 형태로 사용할 수 있다. 소스코드는 대부분 자바이며, 그루비, 루비, 앤틀러 파일이 들어 있다. 젠킨스 WAR를 단독으로 또는 톰캣 같은 자바 애플리케이션 서버에서 서버렛으로 실행할 수 있다. 둘 중 어느 경우든, UI(웹 사용자 인터페이스)를 생성하며 REST API에 대한 호출을 받아들인다.

3. 젠킨스 플러그인

플러그인은 웹 브라우저의 일부로서 쉽게 설치되고 사용될 수 있는 프로그램을 말한다. 젠킨스는 수많은 플러그인을 가지고 있으며 사용자가 기본 플러그인 목록을 수용하거나 원하는 플러그인을 선택하여 설치할 수 있다.



4. 젠킨스 파이프라인

Pipeline script를 생성해 리포지토리에 체크인한다. Pipeline script로써 Jenkinsfile을 생성하여 빌드할 수 있다. 단계별 실행을 통해 어느 단계에 문제가 있는지 확인할 수 있다는 장점을 가진다.

파이프라인 문법에는 두가지 방식이 있다. 처음에는 Scripted 방식이었고 후에 Declarative 방식이 추가되었다.

```
pipeline {
agent { docker 'node:6.3' }
stages {
stage('build') {
steps {
sh 'npm ?version'
}
```

[Declarative pipeline]

```
node('docker') {
checkout scm
stage('Build') {
docker.image('node:6.3').inside {
sh 'npm ?version'
}
```

[Scripted pipeline]

◇ 장점

1. 배포의 간편성

Jenkins는 AWS EC2 서비스에 비해 배포가 빠르고 간편하다. 소스 저장소에 push하거나, 목록에서 배포할 서버를 찾아 클릭하는 것만으로도 배포를 완료할 수 있다.

2. 무료

무료로 제공되고 Reference 및 사용자가 많고 정보가 많은 편이다. (최근 오픈소스 클라우드 컴퓨팅 플랫폼인 .openstack의 CI 솔루션으로 채택되기도 했다)

3. 방대한 양의 플러그인 지원

Hudson core 개발자가 jenkins 를 시작했고 주요 플러그인 개발자도 jenkins 로 전환해서 개발 속도가 빠르고 플러그인 지원이 좋은 편이다.

4. 사용 편의성

설치 및 사용이 간단하다. 실제로 maven으로 build가 구성되어 있다면 jenkins 설치후 project를 만드는데 얼마 걸리지 않는다.

5. Remote Access API 제공

Remote Access API를 제공하므로 다른 솔루션에서 연계하여 기능 확장이 가능하다.

◇ 단점

1. 오토 스케일링

오토 스케일링이 필요하지 않은 서버라면 상관없지만, 오토스케일링이 필요한 사람에게는 늘어난 서버를 수대로, 또 수동으로 세팅해줘야 한다는 것은 매우 번거로운 문제이다.

2. 설정

bamboo에 비해 프로젝트 별 보안 및 권한 설정이 불편하다.

3. 연계의 불편

JIRA나 redmine 등 Issue tracking과 연계가 불편하거나 완벽하지 않다.

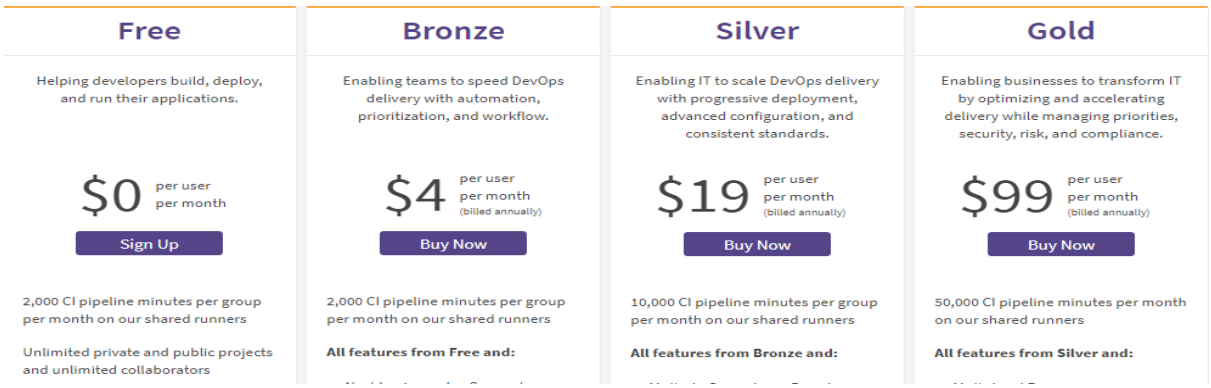
4. 미제공 플러그인

외부 라이브러리를 사용했을 때 젠킨스가 이것을 플러그인으로 지원해 주지 않는다면 자동빌드를 할 수 없다.

2. Gitlab

◇ 개요

마이크로소프트사가 오픈 소스 커뮤니티의 대표주자인 Gitlab을 인수하면서 많은 사람들이 우려를 표하고 대체제를 찾기 시작했다. 그 중 하나인 gitlab은 이슈 트래커, 코드를 저장하는 원격 저장소, 코드를 공유하고 리뷰하는 커뮤니티, 코드의 CI/CD 등의 기능을 통합하여 제공하는 오픈 소스 서비스이다. Github과 유사한 UI를 지니고 있고 clone, push, pull 등의 git 명령을 통해 관리할 수 있다. Gitlab은 자신의 서버에 직접 설치하여 사용할 수도 있고, 공식사이트 (https://gitlab.com /gitlab-com)의 서버를 사용할 수 도 있다. 기본적인 기능은 모두 무료이고 추가 비용을 통해 더 다양한 기능을 제공받을 수 있다.



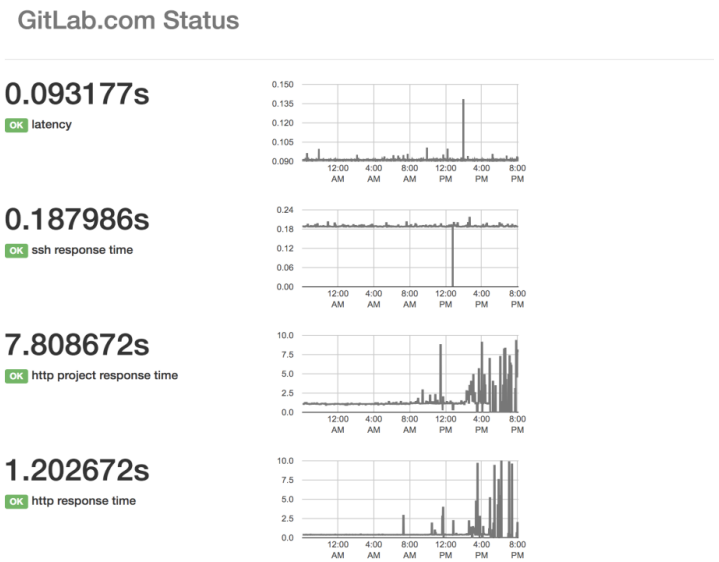
Gitlab의 CI/CD는 gitlab-runner가 gitlab 프로젝트 최상위 디렉토리에 위치한 .gitlab-ci.yml 파일의 코드를 실행하여 이뤄진다. 배포 혹은 빌드를 원하는 서버에 gitlab-runner를 설치하고 프로젝트에 등록하면, 그 프로젝트에 대해 CI/CD가 가능해진다. 이 때, gitlab-runner의 executor는 shell, docker, ssh 등 다양하므로 상황에 맞는 executor 를 선택하면 된다.

◇ 장점

1. Github과 유사한 UI를 지녀 익숙하게 사용할 수 있다.
2. 설치와 설정이 간단하다.
3. Gitlab 서버는 한 곳에만 설치하고, gitlab-runner 을 여러 위치에 설치하여 다양한 서버에서 가볍게 CI/CD 를 구축할 수 있다.
4. 공식 모바일 앱이 존재한다.
5. pull request 등 코드 review 가 쉽다.

◇ 단점

- UI 가 다소 느리다.



◇ 사용법

1. 서버에 맞는 설치법에 따라 gitlab 서버를 설치하거나, 공식사이트를 이용한다.

2. Git 커맨드 혹은 서버 자체의 UI를 활용해 여러 소스를 생성, 수정해 커밋한다.

3. CI/CD 기능을 원한다면, 빌드 및 배포를 할 서버에 gitlab-runner를 설치하고 등록을 원하는 프로젝트의 토큰을 받아와 gitlab-runner register 명령을 통해 runner를 등록한다. 이 때, executor는 shell이 가장 간단하고 대표적이지만, docker 등을 이용하면 해당 프로젝트에 필요한 라이브러리 등의 환경을 간편하게 마련할 수 있다.

4. 프로젝트 repository 최상단에 .gitlab-ci.yml 파일을 push 하면 등록&활성화된 gitlab-runner 에 의해 자동으로 CI/CD 가 진행된다.

3. GCP(Google Cloud Platform)

◇ 개요

- 구글 클라우드 플랫폼(Google Cloud Platform)은 구글에서 개발한 클라우드 컴퓨팅 플랫폼이다. 여기에 아파치 HTTP 서버나 nginx 등의 웹 서버 소프트웨어를 설치해 웹 사이트를 만들 수 있다.
- 리소스는 전 세계 곳곳의 Google Data Center에 위치한 컴퓨터와 하드 디스크 드라이브와 같은 물리적 자산과 가상 머신(VM)과 같은 가상 리소스로 구성.

◇ 서비스

- 항목별로 다수의 서비스가 존재함 (클라우드 컴퓨팅/ 애널리스틱스 및 머신러닝/ ID 및 보안/ 공동작업 및 생산성/ Google Maps Platform/ 브라우저, 하드웨어, OS/ 전문 서비스/ 마켓플레이스)

- 이용할 서비스가 주로 클라우드 컴퓨팅 분야이므로 해당 카테고리를 집중적으로 분석

● 컴퓨팅

탄력성을 지닌 글로벌 부하 분산 서비스부터 유연한 단일 인스턴스 VM까지 제공.

- Google Compute Engine : 확장 가능한 고성능 VM
- Google App Engine : 확장성이 뛰어난 웹 애플리케이션과 모바일 및 IoT 백엔드를 구축하기 위한 플랫폼
- Google Kubernetes Engine : Docker 컨테이너 실행을 위한 강력한 클러스터 관리자이자 조정 시스템
- Google Cloud Container Registry : 널리 사용되는 지속적 배포 시스템과 호환되는 비공개 Docker 저장소
- Google Cloud Function : 어디서든 수신된 이벤트에 따라 로직이 요청 시 가동되는 서버리스 컴퓨팅 모델 제공

● Cloud SDK

리소스를 쉽게 만들고 관리할 수 있는 클라이언트 라이브러리가 포함되어 있음 (앱 API : 서비스에 대한 액세스를 제공/ 관리자 API : 리소스 관리를 위한 기능을 제공. 자동화된 도구를 빌드하려는 경우에 사용 가능)

- Cloud Platform에 사용되는 도구 모음. Gcloud, gsutil, bq를 사용하여 Command line을 통해 Google Compute Engine,

Google Cloud Storage, Google BigQuery 등의 제품과 서비스에 액세스할 수 있음.

- gcloud로 Compute Engine의 가상 머신 집단을 손쉽게 관리 가능
- Cloud Platform Console을 거치지 않고도 Compute Engine 네트워크, 방화벽, 디스크 저장소 등을 관리 가능
- 패키지 관리자에서 자바, Python, Node.js, Ruby, Go, .NET, PHP용 클라이언트 라이브러리를 설치 가능
- Google Cloud Pub/Sub 및 Google Cloud Datastore용 Cloud SDK 에뮬레이터를 사용하여 로컬 개발, 테스트, 검증 목적으로 자체 환경에서 시뮬레이션 가능

◇ 도구

1. Cloud Build

지속적 빌드, 테스트, 배포(CI/CD)를 수행

- Commit to deploy in minutes

PR부터 빌드, 테스트, 배포에 이르기까지 모든 작업들이 매우 간단하게 진행. GitHub, Cloud Source Repositories, Bitbucket 저장소로 변경사항을 푸시할 때 자동으로 소스 코드를 빌드, 테스트, 배포하도록 트리거를 설정 가능

- 빌드 대상 선택

Maven, Gradle, Go와 같은 빌드 도구를 사용하여 소스를 Docker 컨테이너 또는 컨테이너가 아닌 아티팩트로 패키징 가능

- Extremely fast builds

Google의 글로벌 네트워크를 통해 연결된 머신에 액세스하여 빌드 시간을 크게 단축 가능

- 배포 자동화

빌드 단계의 일부로 파이프라인을 만들어 배포를 자동화 가능.

- 커스텀 워크플로

빌드, 테스트, 배포의 일부로 수행할 단계를 완벽하게 제어 가능. 이미지를 빌드, 패키징, 푸시하거나 여러 작업의 커스텀 빌드 단계와 동시 실행을 설정 가능

2. Container Registry

한 곳에서 Docker 이미지를 관리하고, 취약성 분석을 수행하고, 세분화된 액세스 관리 기능으로 액세스 허용 수준을 결정 가능. 기존 CI/CD 통합을 사용하면 완전히 자동화된 Docker 파이프라인을 설정하여 빠른 피드백을 얻을 수 있음

- 안전한 비공개 Docker 레지스트리

Google Cloud Platform의 비공개 Docker 이미지 저장소에 빠르고 안전하게 액세스 가능. 또한 이미지 액세스, 보기, 다운로드를 허용할 사용자를 제어 가능

- **자동으로 빌드 및 배포**

Google Source Repositories, GitHub 등에 코드를 커밋할 때 이미지를 비공개 레지스트리에 자동으로 빌드하고 푸시 가능. Cloud Build에 통합하여 CI/CD 파이프라인을 쉽게 설정하거나 Kubernetes Engine, App Engine, Cloud Function, Firebase에 직접 배포 가능

- **심층적인 취약점 스캔**

소프트웨어 배포 주기의 초기 단계에서 취약점을 감지 가능

- **위험한 이미지 잠금**

기본 통합된 바이너리 승인을 사용해 정책을 정의하여 설저오딘 정책에 위배되는 이미지 배포를 차단 가능

- **기본 Docker 지원**

필요에 따라 여러 개의 레지스트리를 정의. 표준 Docker Command line 인터페이스를 사용하여 Docker 이미지를 비공개 Container Registry로 내보내고 가져올 수 있음

- **빠르고 가용성이 높은 액세스**

전 세계 각 리전에 있는 비공개 저장소를 사용하여 최적의 응답시간을 확보 가능

3. Cloud Source Repository

확장이 가능하며 모든 기능을 갖춘 비공개 Git 저장소에서 코드 공동작업 가능.

- **무제한 비공개 Git 저장소**

GitHub 또는 Bitbucket 저장소의 코드를 미러링하여 효과적인 방식으로 처리 가능

- **개발자 생산성 향상**

지속적 통합을 위해 내장된 통합 기능을 사용하여 코드 변경에 따른 빠른 피드백 가능.

- **빠른 코드 검색**

강력한 정규 표현식을 사용하여 검색 범위를 좁히고 프로젝트, 파일, 코드 저장소에서 한 번에 타겟팅된 검색 실행 가능

◇ 장점

1. 인터페이스가 직관적 깔끔한 편
2. 웹 브라우저에서 SSH 접속 가능함
3. 유료계정 업그레이드 안하면 과금이 되지않는다.
4. 안정성 (라이브 마이그레이션)
    - 물리적 서버에 문제가 발생해도 심각한 경우가 아니라면 서버가 꺼지기 전에 다른 물리적 호스트로 자동으로 마이그레이션 됨.
5. 확정성 : 구글 데이터센터급의 확장성.
6. 프로젝트 계층이 IAM 권한으로 관리되어 보안과 액세스 관리가 편리
7. 가상머신의 인스턴스 타입이 단순.
8. 네트워킹 속도가 2Gbps/core~16Gbps/instance까지 지원 가능
9. Document가 제품별로 자세히 제공되고 있음
10. 무료로 300$ 크레딧을 제공

◇ 단점

1. 한국 리전이 없다(2020년 서울 리전 설립 확정) - 지연 속도 발생
2. 크레딧과 무관한 제품(micro)군이 약하다(컴퓨팅 파워가 약하고 US 리전에만 혜택이 있다.)
3. 트래픽 요금이 아마존보다 약간 비싼 편
4. BigQuery, Spanner, Datastore과 같은 Core GCP 제품들은 제한된 커스텀때문에 변경시 문제가 발생할 수 있음.
5. 아직 beta 단계의 제품들이 다수 남아있음.
6. Google Cloud Storage에서의 데이터 다운로드가 비쌈 (0.12$/GB)
7. SDK API가 AWS보다 안정적이지 못함

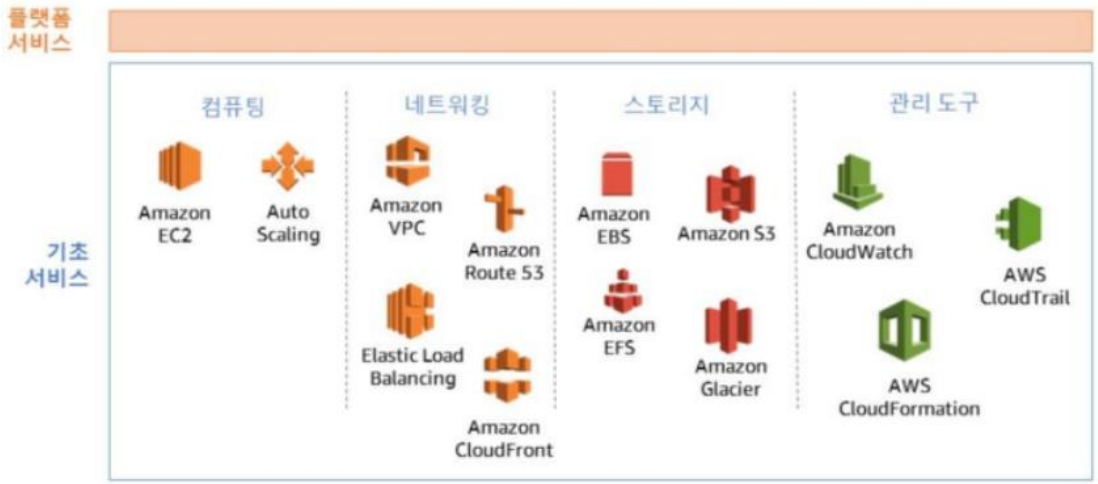
4. AWS (Amazon Web Services)

◇ 개요

- Amazon(아마존)에서 개발한 클라우드 컴퓨팅 플랫폼
- 네트워킹을 기반으로 하는 가상 컴퓨터와 스토리지, 네트워크 인프라 등 다양한 서비스 제공
- 비즈니스와 개발자가 웹 서비스를 이용하여 확장 가능하고 정교한 어플리케이션을 구축할 수 있도록 지원

◇ 기초 서비스 (참고 : https://goddaehee.tistory.com/174)

- GCP와 마찬가지로 AWS도 제공하는 서비스가 다수 존재함.
- 그 중 컴퓨팅, 네트워킹, 스토리지, 관리도구가 기초 서비스에 해당.



(이미지 출처 : https://goddaehee.tistory.com/174)

● 컴퓨팅

Amazon Elastic Compute Cloud(EC2)가 핵심.

- 새로운 서버를 부팅하는데 필요한 시간 단축
- 컴퓨터 요구사항의 변화에 따라 신속하게 파워 확장/축소 가능

● 네트워킹

- DNS 서비스 제공
- 사용자에게 AWS상의 가상 네트워킹 환경을 직접 제어할 수 있는 기능 제공
- 기존 네트워크 내에 있는 것과 같이 EC2 인스턴스와 상호작용 가능

● 스토리지

- 데이터 및 사용 유형에 따라 여러 스토리지 옵션 제공

● 관리 도구

Identity and Access Management(IAM)을 통해 사용자 인증과 권한 부여 기능 관리

- Amazon CloudWatch 및 AWS CloudTrail을 사용하여 성능 지표를 모니터링하고 서비스에 수행된 호출을 기록할 수 있음

◇ 도구 (참고 : https://d1.awsstatic.com/whitepapers/ko_KR/AWS_DevOps.pdf)

- DevOps를 위한 도구를 설명한다

- DevOps를 구현하는 전략으로는 5가지가 있음

(코드형 인프라, 지속적인 개발, 자동화, 모니터링, 보안)

- CI/CD 는 DevOps를 통해 이루어 내는 것.

1. 코드형 인프라

설명 및 선언적인 방식으로 인프라 구축, 배포 및 유지 관리 지원

- AWS CloudFormation

AWS CloudFormation 템플릿을 사용하면 생성 및 업데이트 할 수 있는 AWS 리소스를 정의하고 모델링 할 수 있다. AWS CloudFormation을 사용하면 AWS 리소스 모음을 쉽게 구성 및 배포가 가능하고, 스택이 구성될 때 특수 파라미터 전달, 종속성을 설명할 수 있다.

- AWS AMI

Amazon EC2 인스턴스를 시작(provisioning)할 수 있는 일종의 디지털 템플릿이다. 인스턴스에서 시작 시 어플리케이션 소프트웨어를 설치할 수 있도록 하는 부팅 스크립트 및 소프트웨어를 포함할 수 있다.

부팅 시 추가 소프트웨어를 설치할 필요가 없으므로 인스턴스를 빨리 시작할 수 있다는 장점이 있지만 어플리케이션 수준의 소프트웨어가 변경될 때마다 새 AMI를 생성해야 할 수도 있는 단점이 있다.

2. 지속적인 개발

프로덕션 가능한 어플리케이션 코드의 자동화된 개발을 실현

- AWS CodeDeploy

  관리를 중앙화하고, 기존 소프트웨어 또는 연속 전송 프로세스와 통합하여 최소 중단 시간으로 EC2 집합 전체에 어플리케이션을 배포할 수 있는 기능을 제공한다..

- AWS CodePipeline

  원활한 배포를 돕는 연속 전송 및 방출 자동화 서비스이다. 코드를 체크인하여 구축하고 어플리케이션을 준비 단계로 배포하고 테스트하며 출시하는 개발 워크 플로우를 설계할 수 있다.

- AWS CodeCommit

  Private Git 저장소를 호스팅하는 안전하고 확장 가능한 소스 관리형 서비스이다. 코드부터 바이너리까지 저장 가능하며, Git의 표준 기능을 지원하여 기존 Git 기반 도구와 연동이 가능하다.

- AWS Elastic Beanstalk 및 AWS OpsWorks

어플리케이션 코드 변경 사항 및 인프라 수정 사항의 연속 배포(CD)를 지원한다.

- Blue-Green 배포

DNS를 사용하여 어플리케이션을 배포하는 DevOps의 배포 사례이다. 기존 환경(blue)로 시작하면서 동시에 새 환경(green)을 테스트한다. 새 환경이 모든 필수 테스트를 통과하고 가동 준비가 완료되면 간단히 DS를 통해 이전 환경에서 새 환경으로 트래픽을 redirection하면 된다.

3. 자동화

인프라와 인프라에서 실행되는 어플리케이션의 설정, 구성, 배포 및 지원에 중점을 둠. 자동화를 이용하여 표준화되고 반복 가능한 방식으로 환경을 매우 신속하게 설정

- AWS Elastic Beanstalk

개발자가 일반적으로 사용되는 기술 스택에 어플리케이션을 쉽고 생산적으로 배포할 수 있게 해주는 서비스이다. 자동화는 물론, 자동화된 어플리케이션 배포, 모니터링, 인프라 구성 및 버전 관리 등을 지원한다.

- AWS OpsWorks

구성 관리 소프트웨어 및 어플리케이션 수명 주기 관리와의 통합 같은 추가 기능과 함께 다양한 자동화 수준을 제공한다. 어플리케이션 수명 주기 관리를 사용하여 리소스가 설정, 구성, 배포, 배포 해제 또는 종료되는 시점을 정의할 수 있음.

4. 모니터링

DevOps의 핵심은 커뮤니케이션과 협업이고, 이를 지원하기 위해서는 피드백이 중요. 인프라에 대한 강력한 모니터링, 경고 및 감사 기능을 제공하여 개발자와 운영 팀이 협력할 수 있도록 도움

- Amazon CloudWatch

모든 AWS 리소스와 이러한 리소스에서 실행되는 어플리케이션을 실시간으로 모니터링함. 피드백 제공 뿐 아니라 Auto Scaling 같은 AWS 서비스를 제공하기 위해 사용되기도 함.

- AWS CloudTrail

협력, 커뮤니케이션 및 투명성을 위해서는 누가 인프라를 변경하는지 알아야 하는데 이를 제공해준다. 모든 AWS 상호작용은 AWS CloudTrail에서 모니터링하고 기록하는 AWS API 호출을 통해 처리된다.

.

5. 보안

인프라와 회사 자산을 보호, 문제 발생시 반복적이고 효과적으로 문제를 해결해야함.

- Identity and Access Management (IAM)

보안 인프라 구성 요소 중 하나이다. 사용자가 어떤 AWS 서비스와 리소스에 액세스할 수 있는지를 제어하는 암호, 액세스 키 및 사용 권한 정책과 같은 보안 자격 증명과 사용자를 중앙에서 관리할 수 있다.

◇ 사용법

1. Amazon Web Service(AWS) 10분 자습서
   https://aws.amazon.com/ko/getting-started/tutorials/
2. AWS 단계별 학습 링크
   https://blog.wisen.co.kr/?p=3517

◇ 장점

1. 저렴한 비용

   자본 비용을 가변 비용으로 대체하여 시간대별로도 자원을 끌 수 있음. 최근에는 비용에 관한 장점이 줄어든 편 – 필요한 기능을 추가해서 이용하다 보면 각각 부과되는 비용 방식이 예상보다 지출을 크게 만들기도 함
2. 자체적으로 지원하는 서비스 기능이 다양해 원하는 옵션만 선택해서 이용가능
3. 다양한 고객 경험이 기반이 된 서비스 제공

다양한 고객이 요구하는 기능을 개발해 최대한 추가하는 전략을 구사, 지속적인 서비스 추가를 지원함

4. 창업을 막 시작하는 사람들이 사용하기에도 최적의 서비스
   - 혼자서 인프라를 구축하고 관리하는 것이 가능
   - 초기 비용이 거의 없음
   - 실패하더라도 해당하는 서비스를 제거하면 비용 문제가 없음
   - 서버가 증설 혹은 축소되어야 하는 경우 손쉽게 대응
5. 오토 스케일링을 통한 확장성

   순간적인 접속 과부하를 해결 가능. 엄청난 양의 트래픽이 발생하더라도 접속자를 분석하여 자동으로 서버를 증설
6. 플랫폼 설정 옵션, 모니터링 및 정책 기능, 보안, 안정성 측면에서 높이 평가됨

◇ 단점

1. 장애 발생시 빠른 대처가 이루어지지 않음

2. AWS의 장점인 다양한 서비스와 기능을 사용하기 위해서 숙련된 개발자가 필요

시작은 쉽지만, 효과적으로 사용하기에는 복잡한 기능과 서비스로 인해 비용과 기술 지원 측면에서 어려움이 있음

3. 요금 체계가 복잡함 (가능하다면 컨설팅 받는 것을 추천)

4. 무료제공 트래픽이 없음.

5. EC2 인스턴스가 restart될 때마다 IP가 매번 바뀜에 유의

IP가 바뀌지 않게 하려면 EIP 사용 – 비용이 큼

6. 하이브리드 클라우드 전략 측면에서는 뒤처지는 모습을 보임

(On-Premise 사설 클라우드의 장점을 경시하는 경향을 가지고 있음, 중요 데이터는 자체 데이터 센터에 보관하고 공용 클라우드는 다른 목적으로 활용하려는 기관이 꽤 있는 것에 반해 이에 관한 지원은 떨어지는 편)

Ⅳ. 결론 및 분석

1. Jenkins와 GitLab의 비교

Jenkins와 다르게 Gitlab은 다음과 같은 특징을 가지고 있다.

1. 코드 보안을 유지하는 세분화 된 액세스 제어로 git 저장소를 관리한다.
2. Merge request로 코드 검토를 실행하고, 공동 작업의 효율을 향상시킬 수 있다.
3. 각 프로젝트는 issue tracker와 wiki를 가지고 있어 연계가 편하다.

Gitlab과 다르게 Jenkins는 다음과 같은 특징을 가지고 있다.

1. 설치가 간편하다.
2. 구성이 알아보기 쉽게 되어 있다.
3. 세트 지원 변경이 용이하다.

Jenkins가 Gitlab보다 배우기 쉽고 사용하기 좋다. Jenkins의 GUI가 Gitlab과는 다르게 속도도 빠르고 보기에 편하다. 그리고 자신이 필요한 플러그인을 찾아서 다운로드하면 쉽게 사용할 수 있기 때문에 처음에 사용하기에는 jenkins과 편할 수 있다. 하지만 깊게 들어가면 Jenkins의 수동으로 오토스케일링을 해줘야하는 문제, 보안 및 권한 설정이 불편, issue tracking과 연계가 불편하다는 점, 필요한 플러그인을 일일이 다 다운받아야 하는 점도 존재하기 때문에 자신이 진행할 프로젝트의 특성, 자

신이 각 CI/CD 오픈소스에 대한 이해정도에 따라 적절히 선택하는 것이 좋을 것 같다.

2. GCP와 AWS의 비교

1. 유료 서비스
    - 현재 국내 대기업, 공공기관에서 사용하는 거의 대부분의 클라우드는 AWS이다. (독점적인 위치에 존재)
    - 다양한 레퍼런스 존재

2. 무료 프리티어
    - 과금 체계 : 둘다 해외 결제가 가능한 카드가 필요하다. 하지만 AWS는 프리티어 무료 서비스를 초과해서 사용하면 카드에 자동으로 비용이 청구되므로 부담감이 존재한다. 이에 비해 GCP는 유료계정으로 전환하지 않는한 절대로 자동 결제가 되지 않는다.
    - 무료 사용 크레딧 : GCP가 AWS보다 크레딧을 더 많이 지원해준다.

3. 클라우드 서비스(가상서버) 운영 및 대시보드 등 모니터링
    - 가상서버(VM) 생성 : AWS 프리티어 서비스는 설치 가능한 플랫폼이 상당히 제한적이지만 구글 클라우드는 거의 모든 클라우드 서비스를 이용가능하다.
    - 가상서버 운영 및 보안키 : GCP는 크롬에서 바로 운영이 가능하지만, AWS는 플랫폼을 따로 설치해야 하는 번거로움이 존재한다.

- 프리티어 사용량 모니터링 : GCP는 자동 결제가 될 걱정이 없기에 결제관련 알람설정도 필요없고, 무료 사용량 조회도 간단하다.

여러 가지 측면에서 가상서버 및 클라우드 서비스를 경험해보기에는 구글 클라우드 플랫폼이 훨씬 사용하기 좋다. 그럼에도 불구하고, 향후 클라우드 전문가를 생각하거나 취업 시 클라우드 경험을 생각하면 시장에서 가장 많이 사용하고 있는 AWS를 사용하는 것이 좋을 수 있다.