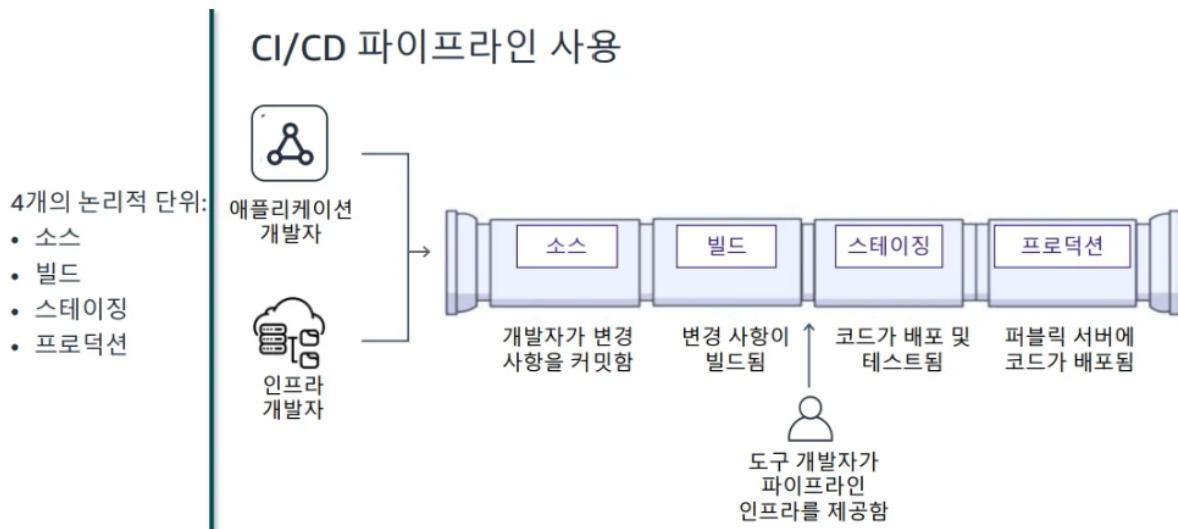


🎵 CI/CD

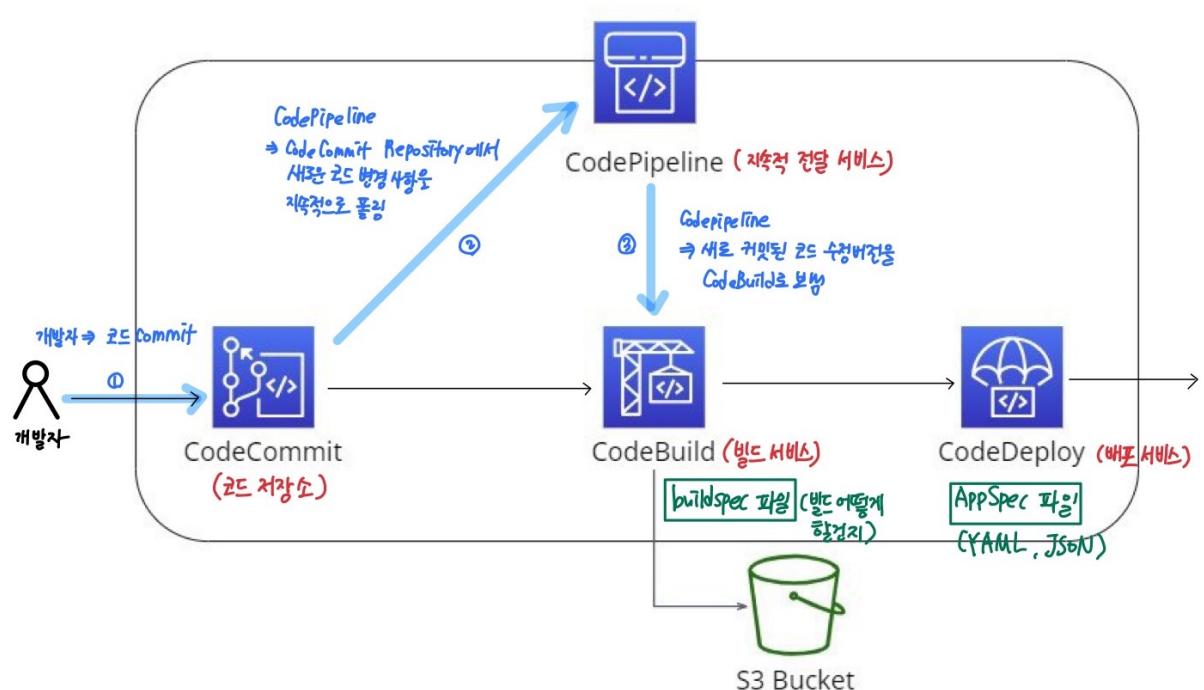
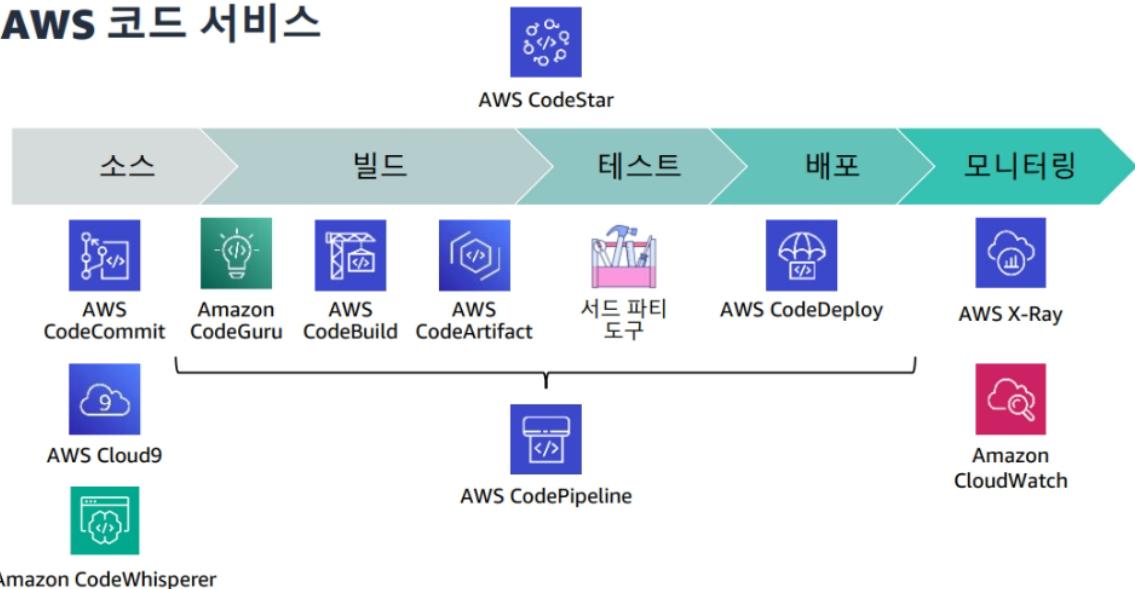
0. CI/CD

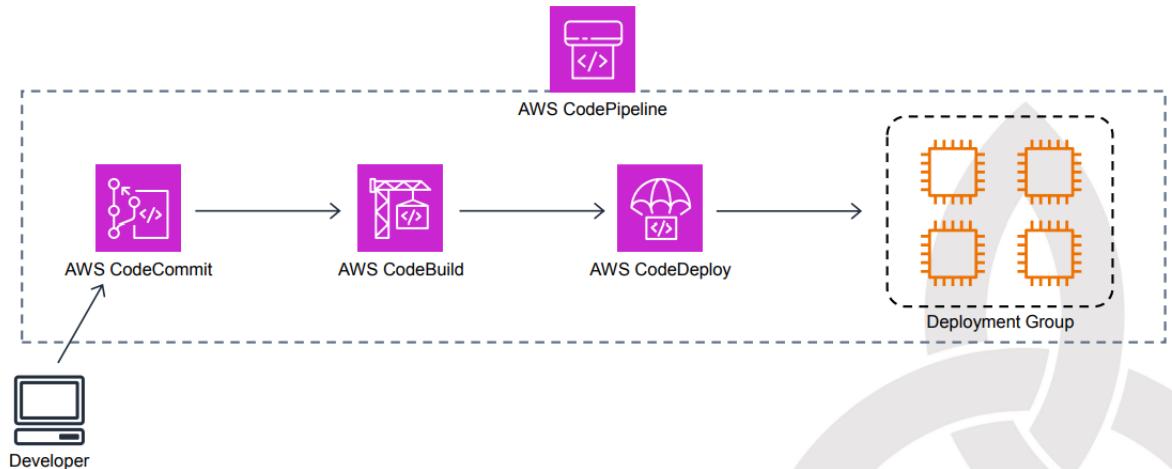


- Continous Integration/Continuous Delivery
- 애플리케이션 개발 단계를 자동화하여 애플리케이션을 보다 짧은 주기로 고객에게 제공하는 방법
- 자주 빌드하고 자주 배포하자 ⇒ 유저에게 빠르게 제품을 전달
 - 더 빠르게 버그가 수정되고 유저의 요구사항이 반영될 수 있는 시스템

1. AWS CI/CD 서비스

AWS 코드 서비스





1. CodePipeline

- SW 릴리스에 필요한 단계를 모델링, 시작화 및 자동화하는데 사용할 수 있는 **지속적 전달 서비스**
- 릴리스 프로세스 자동화

2. CodeCommit

- 코드를 안전하게 보관
- Git 같은 것
- 완전관리형
- 버전 관리 서비스

3. CodeBuild



참고)

https://www.youtube.com/watch?v=SjnwThCUXqA&list=PLfth0bK2MgIZsq9C7Cs6_Vp_D8o_V8Qoc&index=19

- 빌드 & 테스트를 서비스 제공
- 소스 코드를 컴파일하고, 유닛 테스트를 실행하고, 배포할 준비가 된 아티팩트를 생성함
- CodeBuild 사용 시 자체 빌드 서버를 프로비저닝, 관리 및 크기 조정할 필요가 x
- 완전관리형

3-1. buildspec.yml 파일

- 빌드를 어떻게 할 것인지에 대한 파일
- 루트 디렉토리에 위치해야 함
 - 경로 변경 가능
- 정의 가능한 내용
 - 빌드 환경
 - ex) node.js, python 등
 - 환경 변수
 - phases 별 명령어

install	빌드/테스트 환경을 구성하기 위한 패키지의 다운로드
pre_build	빌드/테스트 전 수행해야 하는 내용 ex) 외부 리소스 확보, 디펜던시 다운로드 등
build	실제 빌드/테스트 수행
post_build	빌드/테스트 이후 마무리 작업 ex) docker image 푸시, 슬랙 알람 등

- 아티팩트 설정
 - path 및 구성 방법
- 기타
 - 캐시, 프록시
- buildspec 구문 설명

3-2. artifact

- 빌드된 결과물
 - 보통 빌드된 결과물을 artifact로 만들어서 code pipeline으로 넘김

3-3. CodeBuild Caching

2가지 모드가 존재

1. Amazon S3 caching

- 서로 공유하는 S3 버킷에 캐시하는 것
- 주로 **소거나, 미리 빌드해두면 좋은 패키지** 등을 저장
- **network**을 사용함 → so 큰 파일의 경우 비효율적

2. Local Caching

- 하나의 빌드 호스트에 로컬로 저장하는 캐시 (공유X)
- 주로 크거나 바로 필요한 내용을 캐싱하는 것
- 빌드가 빈번한 경우
- 3가지 모드 존재

1) Source Cache

- Primary/Secondary Source의 Git Metadata를 캐시
- 캐시 생성 이후 부터는 커밋의 변화 부분만 가져옴
- git 소스코드 자체가 엄청 큰 상황일 때

2) Docker Layer Cache

- Docker의 Layer를 캐시
- 큰 도커 이미지를 빌드하거나 가져올때
- 리눅스 환경만 가능

3) Custom Cache Mode

- buildspec에서 명시한 디렉토리만 캐시
- 소스 다운로드 전에 세팅
 - 소스에 동일한 파일명이 있다면 덮어씌워짐(overwrite)

4. CodeDeploy



참고)

https://www.youtube.com/watch?v=_oOGYyKnaI0

• 자동화된 배포 서비스

- EC2, 온프레미스 인스턴스, 서비스 AWS Lambda 함수, ECS에 app을 자동으로 배포하는 서비스
- 새 기능을 신속하게 릴리스 가능
- 완전관리형

4-1. AppSpec 파일

- CodeDeploy에서 배포를 관리하는 데 사용하는 YAML 형식 또는 JSON 형식의 파일
- 예제

```

version: 0.0 ) 설정
os: linux
files:
  - source: /index.html      ) /var/www/html에 index.html 배포
    destination: /var/www/html
hooks: ⇒ Event hook
  BeforeInstall:
    - location: scripts/install_dependencies.sh ) location : 스크립트 파일의 경로
      timeout: 300 ) 기본값 1h(3600s), (필수 옵션)
    사용자( runas: root 사용자 지정 시간 논의 ⇒ 파일을 간주
  AfterInstall:
    - location: scripts/change_permissions.sh
      timeout: 300
      runas: root
  ApplicationStart:
    - location: scripts/start_server.sh
    - location: scripts/create_test_db.sh
      timeout: 300
      runas: root
  ApplicationStop:
    - location: scripts/stop_server.sh
      timeout: 300
      runas: root

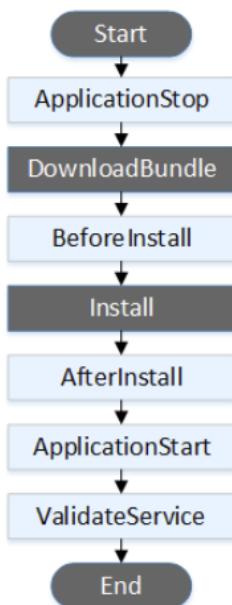
```

<https://yoo11052.tistory.com/113>

4-2. CodeDeploy Agent

- 인스턴스에 설치 및 구성할 때 해당 인스턴스를 배포에 **CodeDeploy** 사용할 수 있게 해주는 소프트웨어 패키지

4-3. CodeDeploy의 Event LifeCycle



- 이와 같은 순서대로 event 실행됨
- 이 순서는 CodeDeployment에서 이미 정해놓은 순서임
- 각각을 Event Hook라고 함
- 일부 Event Hook은 사용자가 직접 **스크립트 파일**을 생성해 사용할 수 있고, 일부는 이미 **예약되어** 있는 Event Hook이라 사용할 수 x
 - 왼쪽 그림에서 회색 ⇒ 이미 예약되어 있는 Event Hook

1. Start

- lifecycle의 첫 번째 이벤트
- CodeDeploy 에이전트를 자동으로 실행하고, 인스턴스 배포가 시작

2. ApplicationStop

- 이전 프로그램을 중지하는 스크립트를 실행하는 단계
- ex) 커머스 웹 app을 운영하고 있는데 새 버전(v.1)을 배포할 경우, 이 이벤트를 통해 구 버전(v.0)을 사용하지 않도록 설정하고, 새 버전(v.1)을 수신하도록 인스턴스를 준비 할 수 있음

3. DownloadBundle

- 이 이벤트 동안 CodeDeploy 에이전트는 새 버전을 인스턴스로 가져옴
- ex) CodeBuild에서 패키징된 zip 파일

4. BeforeInstall

- 이 이벤트를 통해 구버전의 설치 구성을 저장하고, 파일을 복호화하고, 현재 버전의 백업을 만들 수 있다.

5. Install

- DownloadBundle을 통해 가져온 Bundle의 압축을 해제하고, appspec.yml에 정의 된대로 파일을 지정한 경로로 복사함

6. AfterInstall

- 이 이벤트를 통해 **프로그램이 시작되기 전에 프로그램의 구성을 변경할 수 있음**

7. ApplicationStart

- 어플리케이션을 구 버전(v.0) 대신 새 버전(v.1)로 설정함

8. ValidateService

- 이 이벤트를 통해 **배포가 성공했는지 확인할 수 있는 검증 로직을 실행할 수 있음**

9. End

- lifecycle의 마지막 이벤트로, 인스턴스의 배포 성공유무를 중앙 서비스에 알림

5. CloudFormation

- IaC 서비스
- **VPC, EC2, Lambda** 등과 같은 리소스를 수동으로 생성할 필요x이 리소스들을 템플릿(코드)으로 구성하고, Stack을 생성하여 인프라를 구성

5-1. 구성

1. 템플릿

- aws 리소스를 프로비저닝 및 구성을 위해 필요한 파일
- yaml, json 형식

2. 스택

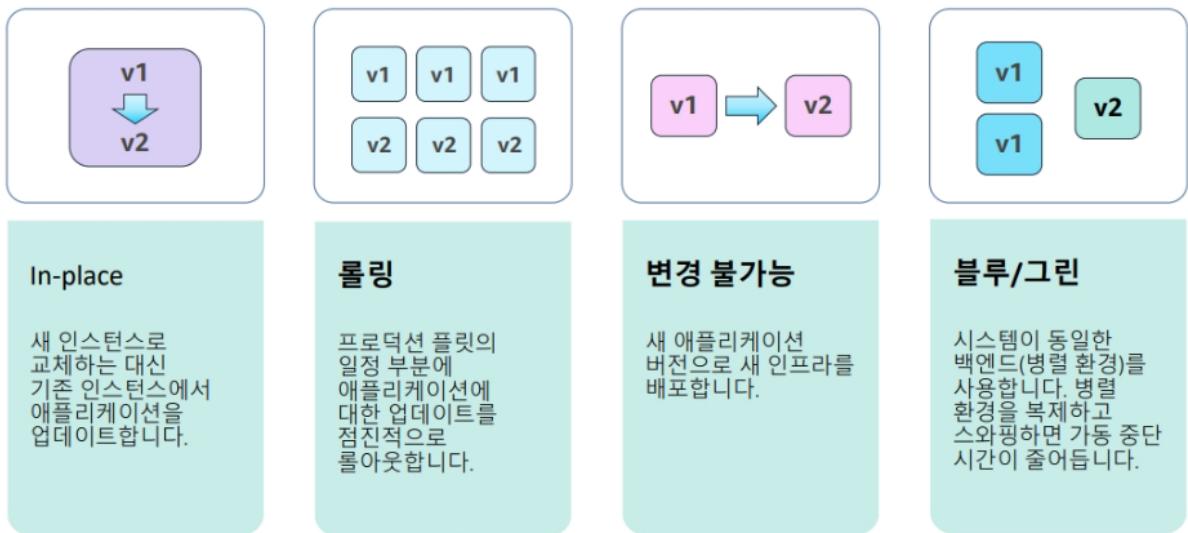
- 하나의 단위로 관리할 수 있는 aws 리소스 모음
- 스택을 통해 템플릿을 읽고, 실제 리소스를 생성하고, 인프라를 관리함
- 스택의 모든 리소스는 템플릿을 통해 정의함
- 스택을 삭제 시 → 스택이 관리하는 모든 리소스 삭제됨

<https://yoo11052.tistory.com/188>

6. CodeWhisperer

- 개발자가 IDE(통합 개발 환경)에서 작성하는 **주석과 코드를 분석**
- 단순히 코드 완성하는게 X닌 자연어 처리 기능을 사용해 코드의 주석을 이해함

2. 지속적 배포 전략



1. 롤링 배포 및 롤백

- 일반적으로 기존 리소스에 배포함
- 기존 코드를 덮어 쓰는 것이 효율적
- 배포 실패 시 이전 버전으로 롤백해야하는 경우에는 어려움

2. 변경 불가능한 업데이트

- 변경 불가능한 업데이트 실패할 경우
 - 롤백 프로세스에서 ASG 종료하기만 하면됨
 - BUT 롤링 업데이트 ⇒ 변경사항을 롤백하기 위한 추가 롤링 업데이트 수행해야 함

3. 블루/그린 배포

- 블루 ⇒ 기존 환경
- 그린 ⇒ 새 버전
- 새 버전 배포 시 위험 줄일 수 있음

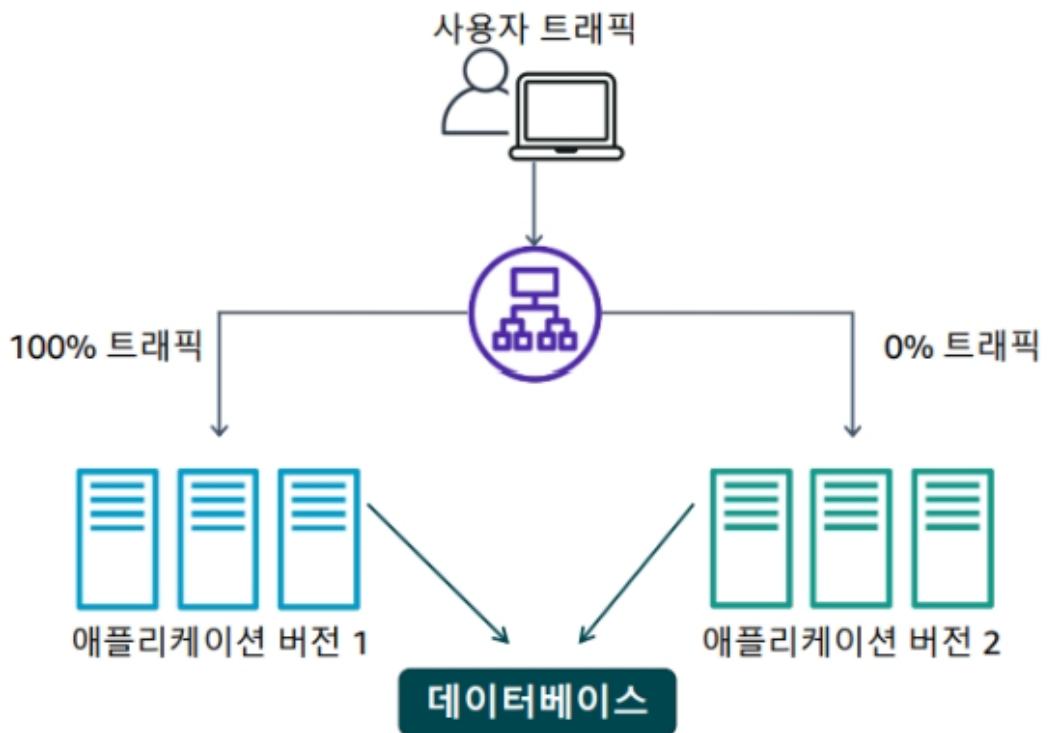
1) 방식

1. 카나리
 - 트래픽을 사전에 설정된 2가지 비율로 전환함
2. 선형
 - 트래픽을 동등한 증분 및 시간(분)으로 전환함

3. 한꺼번에

- 트래픽을 한 번에 완전히 전환함

2) 데이터베이스 블루/그린 배포



- 블루 버전 & 그린 버전 ⇒ 모두 동일한 DB 공유가능해야 함
- 그린 버전의 DB 변경사항은 블루버전(이전 버전)과 호환이 되어야 함

CI/CD	풍선에 개발환경 (환경)	코드 저장소	빌드	배포	지속적 통합 S
Microservice	eks	컨테이너	컨테이너 이미지 registry	컨테이너 배포·리스팅 플랫폼 앤진	Lambda
Infrastructure as Code	IaC				
Monitoring and Logging					



Skill Builder : Introduction to AWS CodeDeploy (Korean)

개요

이 실습에서는 AWS CodeDeploy를 소개합니다. 이 실습에서는 AWS CodeDeploy를 사용하여 애플리케이션을 Amazon EC2 인스턴스에 배포합니다.

목표

이 실습을 마치면 다음을 수행할 수 있습니다.

- CodeDeploy 에이전트가 설치되었는지 확인합니다.
- CodeDeploy에 배포할 애플리케이션 소스 콘텐츠를 구성합니다.
- Amazon S3 버킷을 생성한 다음 버킷에 WordPress 애플리케이션을 업로드합니다.
- Amazon EC2 인스턴스에 WordPress 애플리케이션을 배포합니다.
- WordPress 애플리케이션 배포를 모니터링합니다.
- WordPress 애플리케이션을 업데이트한 다음 재배포합니다.

개념

- **AWS CodeDeploy**
 - Amazon EC2 인스턴스, 온프레미스 인스턴스 또는 서비스 Lambda 함수로의 애플리케이션 배포를 자동화하는 배포 서비스
 - 코드, 서비스 AWS Lambda 함수, 웹 및 구성 파일, 실행 파일, 패키지, 스크립트, 멀티미디어 파일 등 거의 모든 애플리케이션 콘텐츠를 배포할 수 있음
 - AWS CodeDeploy는 서버에서 실행하고 Amazon S3 버킷, GitHub 리포지토리 또는 Bitbucket 리포지토리에 저장된 애플리케이션 콘텐츠를 배포할 수 있음
 - 서비스 Lambda 함수도 배포 가능
 - AWS CodeDeploy를 사용하기 전에 **기존 코드를 변경할 필요 x**
 - AWS CodeDeploy 사용 시 다음 작업을 보다 쉽게 수행 가능

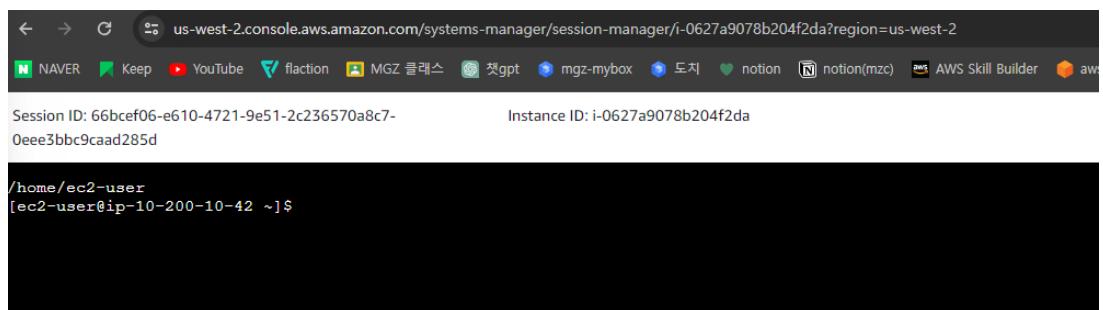
- 새로운 기능의 신속한 릴리스
- AWS Lambda 함수 버전 업데이트
- 애플리케이션 배포 시 가동 중지 시간 방지
- 오류가 발생하는 수동 배포와 관련된 다양한 위험 없이 애플리케이션 업데이트에 따른 복잡성 처리

태스크 1: Linux EC2 인스턴스에 연결

이 랩의 일부로, Linux EC2 인스턴스가 자동으로 생성되어 있습니다.

이 태스크에서는 **AWS Systems Manager - Session Manager**를 사용하여 EC2 인스턴스에 연결합니다.

1. 이 지침의 왼쪽에 있는 목록에서 **Ec2InstanceSessionUrl** 값을 복사하여 새 웹 브라우저 탭에 붙여넣기
 - 새 웹 브라우저 탭에서 EC2 인스턴스에 연결하는 AWS Systems Manager - Session Manager 콘솔이 열림
 - 인스턴스에 연결하면 일련의 명령이 자동으로 실행되어 사용자의 홈 디렉터리로 변경하고 다음과 같이 작업 디렉터리의 경로를 표시함



태스크 2: 환경 확인

다음을 확인하기

- EC2 인스턴스에서 CodeDeploy 에이전트가 실행 중입니다.
- EC2 보안 그룹이 올바르게 구성되었습니다.

1. CodeDeploy 에이전트가 실행 중인지 확인

1. 인스턴스에서 CodeDeploy 에이전트가 실행 중인지 확인하기

```
sudo service codedeploy-agent status
```

```
[ec2-user@ip-10-200-10-226 ~]$ sudo service codedeploy-agent status  
The AWS CodeDeploy agent is running as PID 2477
```



참고) AWS CodeDeploy 에이전트

- 인스턴스에 설치 및 구성한 경우 해당 인스턴스를 AWS CodeDeploy 배포에서 사용할 수 있게 해 주는 소프트웨어 패키지
- EC2/온프레미스 컴퓨팅 플랫폼에 배포하는 경우에만 필요**
- AWS Lambda 컴퓨팅 플랫폼을 사용하는 배포에는 필요X

2. 보안 그룹 확인

- EC2 > **Security Groups**를 클릭 > **CodeDeploySG**를 선택 > **Inbound Rules** 탭을 클릭
 - Source - 0.0.0.0/0에서 HTTP 액세스를 허용하는 규칙이 표시됨

The screenshot shows the AWS EC2 Security Groups page for the 'sg-015f9bd249a23e88f - CodeDeploySG' group. The 'Inbound rules' tab is selected, displaying two rules:

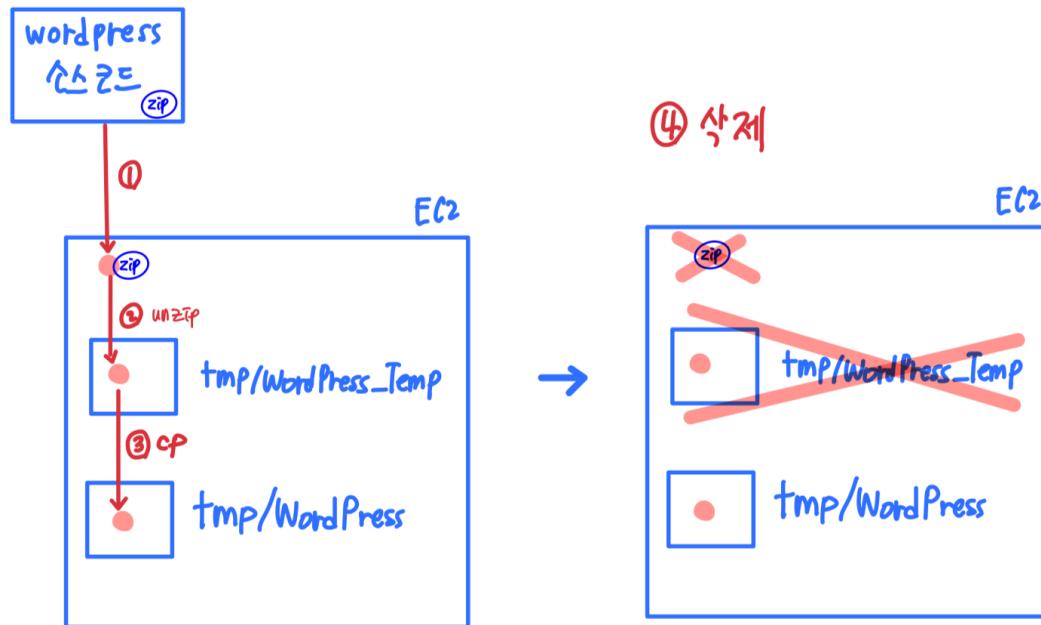
Name	Security group rule...	IP version	Type	Protocol	Port range	Source	Description
-	sgr-0939880258eef52f6	IPv4	HTTP	TCP	80	0.0.0.0/0	-
-	sgr-03ed7cd7ecbd0a28d	IPv4	HTTPS	TCP	443	0.0.0.0/0	-

태스크 3: 애플리케이션 소스 콘텐츠 구성

다음을 수행하기

- 배포할 소스 애플리케이션 콘텐츠를 구성합니다.
- 애플리케이션이 배포될 때 실행될 스크립트를 생성합니다.
- CodeDeploy AppSpec 파일을 생성합니다.

1. WORDPRESS 코드 다운로드



1번 실습 과정 그림

1. WordPress 소스 코드를 Linux 인스턴스에 복사하기

```
 wget https://s3-us-west-2.amazonaws.com/us-west-2-aws-training/courses/spl-82/v1.4.5.prod-2bbeba30/scripts/WordPress-master.zip
```

```
[ec2-user@ip-10-200-10-42 ~]$ wget https://s3-us-west-2.amazonaws.com/us-west-2-aws-training/courses/spl-82/v1.4.5.prod-2bbeba30/scripts/WordPress-master.zip
--2024-04-01:14:55:55 https://s3-us-west-2.amazonaws.com/us-west-2-aws-training/courses/spl-82/v1.4.5.prod-2bbeba30/scripts/WordPress-master.zip
Resolving s3-us-west-2.amazonaws.com (s3-us-west-2.amazonaws.com)... 52.218.234.152, 52.218.236.160, 52.92.201.24, ...
Connecting to s3-us-west-2.amazonaws.com (s3-us-west-2.amazonaws.com)|52.218.234.152|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 13373728 (113M) [application/xzip]
Saving to: 'WordPress-master.zip'

100%[=====] 13,373,728 44.2MB/s   in 0.3s

2024-04-01:14:56:36 (44.2 MB/s) - 'WordPress-master.zip' saved [13373728/13373728]
[ec2-user@ip-10-200-10-42 ~]$
```

2. 마스터 .zip 파일의 압축을 /tmp/WordPress_Temp 폴더에 풀기

```
unzip WordPress-master.zip -d /tmp/WordPress_Temp
```

3. WordPress 대상 폴더 생성하기

```
mkdir -p /tmp/WordPress
```

4. 압축을 푸니 콘텐츠를 /tmp/WordPress 대상 폴더에 복사하기

```
cp -paf /tmp/WordPress_Temp/WordPress-master/* /tmp/WordPress
```



명령어 해석) -paf

- **p** : 파일의 소유권, 권한, 타임스탬프 등을 보존하여 파일을 복사합니다.
- **a** : "archive" 모드로 파일을 복사합니다. 소유권, 권한, 심볼릭 링크 등 의 속성을 보존하고, 디렉토리 구조를 재귀적으로 복사합니다.
- **f** : 목적지에 이미 동일한 이름의 파일이 있어도 덮어쓰기를 강제합니다.

5. 임시 /tmp/WordPress_Temp 폴더와 마스터 파일 삭제하기

```
rm -rf /tmp/WordPress_Temp  
rm -f WordPress-master.zip
```

2. 애플리케이션을 실행하기 위한 스크립트 작성

EC2

tmp/WordPress

Scripts

- `install_dependencies.sh` (apache, MariaDB, PHP 설치)
MySQL 자원 추가 ↑
- `start_server.sh` (apache, MariaDB 시작)
- `stop_server.sh` (apache, MariaDB 중지)
- `create_server.sh` (MariaDB 사용할
WordPress의 테스트 DB 만들기)
- `change_permission.sh` (apache 폴더 권한 변경)

2번 실습 과정 그림

1. WordPress 소스 코드의 사본에 스크립트 디렉터리를 만들기

```
mkdir -p /tmp/WordPress/scripts
```

2. 다음을 실행하여 install_dependencies.sh 스크립트를 생성하기

```
nano /tmp/WordPress/scripts/install_dependencies.sh  
#!/bin/bash  
yum install -y httpd php mariadb-server php-mysqlnd
```

The screenshot shows a terminal window on an AWS Systems Manager session. The URL is `us-west-2.console.aws.amazon.com/systems-manager/session-manager/i-0627a9078b204f2da?region=us-west-2`. The terminal title is "GNU nano 2.9.8". The script content is:

```
#!/bin/bash  
yum install -y httpd php mariadb-server php-mysqlnd
```

- `install_dependencies.sh` 스크립트는 Apache, MariaDB, PHP를 설치함
- 또한 PHP에 MySQL 지원을 추가

3. 다음을 실행하여 start_server.sh 스크립트를 만들기

```
nano /tmp/WordPress/scripts/start_server.sh  
#!/bin/bash  
service httpd start  
service mariadb start
```

The screenshot shows a terminal window on an AWS Systems Manager session. The URL is `us-west-2.console.aws.amazon.com/systems-manager/session-manager/i-0627a9078b204f2da?region=us-west-2`. The terminal title is "GNU nano 2.9.8". The script content is:

```
#!/bin/bash  
service httpd start  
service mariadb start
```

- 이 `start_server.sh` 스크립트는 Apache와 MariaDB를 시작함

4. 다음을 실행하여 stop_server.sh 스크립트 만들기

```

nano /tmp/WordPress/scripts/stop_server.sh
#!/bin/bash
isExistApp=`pgrep httpd`
if [[ -n $isExistApp ]]; then
    service httpd stop
fi
isExistApp=`pgrep sql`
if [[ -n $isExistApp ]]; then
    service mariadb stop
fi

```

Session ID: 66bcef06-e610-4721-9e51-2c236570a8c7 Instance ID: i-0627a9078b204f2da
0eee3bbc9caad285d

```

GNU nano 2.9.8
#!/bin/bash
isExistApp=`pgrep httpd`
if [[ -n $isExistApp ]]; then
    service httpd stop
fi
isExistApp=`pgrep sql`
if [[ -n $isExistApp ]]; then
    service mariadb stop
fi

```

- stop_server.sh 스크립트는 Apache와 MariaDB를 중지함

5. 다음을 실행하여 create_test_db.sh 스크립트를 만들기

```

nano /tmp/WordPress/scripts/create_test_db.sh
#!/bin/bash
mysql -uroot <<CREATE_TEST_DB
CREATE DATABASE IF NOT EXISTS test;
CREATE_TEST_DB

```

Session ID: 66bcef06-e610-4721-9e51-2c236570a8c7-
0eee3bbc9caad285d

Instance ID: i-0627a9078b204f2da

```
GNU nano 2.9.8

#!/bin/bash
mysql -uroot <<CREATE_TEST_DB
CREATE DATABASE IF NOT EXISTS test;
CREATE_TEST_DB
```

- 이 create_test_db.sh 스크립트는 MariaDB를 사용할 WordPress의 테스트 데이터베이스를 만듭니다.

6. 다음을 실행하여 change_permissions.sh 스크립트 만들기

```
nano /tmp/WordPress/scripts/change_permissions.sh
#!/bin/bash
chmod -R 777 /var/www/html/WordPress
```

Session ID: 66bcef06-e610-4721-9e51-2c236570a8c7-
0eee3bbc9caad285d

Instance ID: i-0627a9078b204f2da

```
GNU nano 2.9.8

#!/bin/bash
chmod -R 777 /var/www/html/WordPress
```

- 이 스크립트는 Apache의 폴더 권한을 변경하는 데 사용됩니다.

7. 모든 스크립트를 실행 가능하게 만들기

```
chmod +x /tmp/WordPress/scripts/*
```

8. 스크립트를 나열하기

```
ls -la /tmp/WordPress/scripts
```

```
[ec2-user@ip-10-200-10-42 ~]$ ls -la /tmp/WordPress/scripts
total 24
drwxrwxr-x 2 ec2-user ec2-user 136 Apr  4 02:10 .
drwxrwxr-x 6 ec2-user ec2-user 4096 Apr  4 01:53 ..
-rwxrwxr-x 1 ec2-user ec2-user 50 Apr  4 02:10 change_permissions.sh
-rwxrwxr-x 1 ec2-user ec2-user 188 Apr  4 02:09 create_test_db.sh
-rwxrwxr-x 1 ec2-user ec2-user 65 Apr  4 01:55 install_dependencies.sh
-rwxrwxr-x 1 ec2-user ec2-user 55 Apr  4 01:57 start_server.sh
-rwxrwxr-x 1 ec2-user ec2-user 176 Apr  4 02:00 stop_server.sh
[ec2-user@ip-10-200-10-42 ~]$
```

3. CodeDeploy appsec 파일 생성



참고) AppSpec 파일

- AWS CodeDeploy는 appspec.yml 파일을 사용하여 다음을 수행합니다.
 - 애플리케이션 설정 버전의 소스 파일을 대상 Amazon EC2 인스턴스로 매핑합니다.
 - 배포된 파일에 대한 사용자 지정 권한을 지정합니다.
 - 배포 중 대상 Amazon EC2 인스턴스에서 실행할 스크립트를 지정합니다.
- AppSpec 파일은 AWS CodeDeploy에 대해 고유합니다. 이 파일은 **AWS CodeDeploy**가 실행할 배포 작업을 정의합니다. 배포 가능한 콘텐츠 및 **AppSpec 파일을 아카이브 파일로 번들링(묶는 것)**한 다음 **Amazon S3 버킷** 또는 **GitHub 리포지토리**에 업로드합니다. 이러한 아카이브 파일을 **애플리케이션 설정 버전**(또는 간단하게 설정 버전)이라고 합니다.

AppSpec 파일의 이름은 appspec.yml이어야 합니다. 이 파일은 애플리케이션 소스 코드의 루트 디렉터리(/tmp/WordPress)에 저장해야 합니다.

1. /tmp/WordPress에서 애플리케이션 특정 파일을 만들기

```
nano /tmp/WordPress/appspec.yml
```

2. appspec.yml 파일에 다음 텍스트를 추가하고 저장하기

```
version: 0.0
os: linux
files:
  - source: /
    destination: /var/www/html/WordPress
hooks:
```

```
BeforeInstall:
  - location: scripts/install_dependencies.sh
    timeout: 300
    runas: root
AfterInstall:
  - location: scripts/change_permissions.sh
    timeout: 300
    runas: root
ApplicationStart:
  - location: scripts/start_server.sh
  - location: scripts/create_test_db.sh
    timeout: 300
    runas: root
ApplicationStop:
  - location: scripts/stop_server.sh
    timeout: 300
    runas: root
```

- 애플리케이션이 배포될 때 실행됨

태스크 4: WordPress 애플리케이션 배포

이 과제에서는 다음을 수행합니다.

- 애플리케이션 소스 콘텐츠를 호스팅할 Amazon S3 버킷을 생성합니다.
- 애플리케이션 소스 콘텐츠를 Amazon S3 버킷에 업로드합니다.
- *CodeDeploy*라는 환경에서 실행 중인 인스턴스에 **WordPress 애플리케이션을 배포**합니다.



참고) AWS CodeDeploy

- AWS CodeDeploy에서 배포는 하나 이상의 인스턴스에 콘텐츠를 설치하는 프로세스이자 해당 프로세스와 관련된 구성 요소입니다. 이러한 콘텐츠는 코드, 웹 및 구성 파일, 실행 파일, 패키지, 스크립트 등으로 구성될 수 있습니다. AWS CodeDeploy는 사용자가 지정한 구성 규칙에 따라 소스 리포지토리에 저장된 콘텐츠를 배포합니다.

1. CodeDeploy 애플리케이션 생성

```
aws deploy create-application --application-name WordPress_App
```

```
[ec2-user@ip-10-200-10-42 ~]$ aws deploy create-application --application-name WordPress_App
{
    "applicationId": "05a33e1e-4b12-4f9f-9743-0e98bef5dcbd"
}[ec2-user@ip-10-200-10-42 ~]$ █
```

2. Amazon S3 버킷 프로비저닝



참고) Amazon S3

- Amazon S3는 인터넷 어디서나 원하는 양의 데이터를 저장하고 검색할 수 있도록 구축된 객체 스토리지입니다. S3는 **안정성**이 매우 뛰어나고 **가용성**이 높으며 **무제한으로 확장 가능한** 데이터 스토리지 인프라를 **매우 저렴한 비용**으로 제공하는 간단한 **스토리지 서비스**입니다.

1. AWS Systems Manager - Session manager 탭에서 다음과 같은 방법으로 S3 버킷 생성하기

```
aws s3 mb s3://codedeploybucketNUMBER
```

- **NUMBER**를 무작위 번호로 교체

```
[ec2-user@ip-10-200-10-42 ~]$ aws s3 mb s3://codedeploybucket0811
make_bucket: codedeploybucket0811
[ec2-user@ip-10-200-10-42 ~]$ █
```

2. S3 버킷의 이름을 텍스트 편집기에 복사하기

- `codedeploybucket000811`

3. WordPress 파일이 들어 있는 폴더로 이동하기

```
cd /tmp/WordPress
```

3. Amazon S3에 Wordpress 애플리케이션 업로드

1. 다음 push 명령을 텍스트 편집기에 붙여넣기

```
aws deploy push --application-name WordPress_App --description "This is a revision  
for the application WordPress_App" --ignore-hidden-files --s3-location s3://{BUCKET 이름}/WordPressApp.zip --source .
```

```
[ec2-user@ip-10-200-10-42 WordPress]$ aws deploy push --application-name WordPress_App --description "This is a revision for the application WordPressApp.zip --source .  
To deploy with this revision, run:  
aws deploy create-deployment --application-name WordPress_App --s3-location bucket=codedeploybucket0811,key=WordPressApp.zip,bundleType=zip,eTag=9858  
[ec2-user@ip-10-200-10-42 WordPress]$ ^C  
[ec2-user@ip-10-200-10-42 WordPress]$
```

이 명령은 다음 작업을 수행합니다.

- 번들링된 파일을 WordPress_App이라는 애플리케이션과 연결합니다.
- 수정 버전에 설명을 첨부합니다.
- 숨겨진 파일을 무시합니다.
- 수정 버전의 이름을 WordPressApp.zip으로 지정하고 버킷에 푸시합니다.
- 루트 디렉터리의 모든 파일을 수정 버전으로 번들링합니다.

4. 애플리케이션 배포

1. 다음 create-deployment-group 명령을 텍스트 편집기에 붙여넣기

```
aws deploy create-deployment-group --application-name WordPress_App --deployment-  
config-name CodeDeployDefault.AllAtOnce --deployment-group-name WordPress_DG --ec2-  
tag-filters Key=Name,Value=CodeDeploy,Type=KEY_AND_VALUE --service-role-arm  
{CodeDeployRoleARN}
```



명령어 해석)

이 명령어를 실행하면 "WordPress_App" 응용 프로그램에 "WordPress_DG"라는 새로운 배포 그룹이 생성되며, 이 그룹은 "CodeDeployDefault.AllAtOnce" 배포 구성을 사용하여 EC2 인스턴스를 필터링하여 "CodeDeploy"라는 이름의 태그가 지정된 인스턴스에 배포됩니다.

- `--application-name WordPress_App` : 배포 그룹이 속할 응용 프로그램의 이름을 지정합니다.
- `--deployment-config-name CodeDeployDefault.AllAtOnce` : 배포 구성을 지정합니다. 여기서는 "CodeDeployDefault.AllAtOnce"를 사용하여 한번에 모든 인스턴스에 배포하는 설정을 사용합니다.
- `--deployment-group-name WordPress_DG` : 새로운 배포 그룹의 이름을 지정합니다.
- `--ec2-tag-filters Key=Name,Value=CodeDeploy,Type=KEY_AND_VALUE` : 배포 그룹에 속한 EC2 인스턴스를 필터링하기 위해 사용되는 EC2 태그 필터를 지정합니다. 이 예에서는 "Name"이 "CodeDeploy"인 인스턴스만을 선택합니다.
- `-service-role-arn {CodeDeployRoleARN}` : CodeDeploy가 배포 작업을 실행하는 데 사용할 IAM 역할의 ARN(Amazon Resource Name)을 지정합니다.

```
[ec2-user@ip-10-200-10-42 WordPress]$ aws deploy create-deployment-group --application-name WordPress_App --deployment-config-name CodeDeployDefault.AllAtOnce --deployment-group-name WordPress_DG --ec2-tag-filters Key=Name,Value=CodeDeploy,Type=KEY_AND_VALUE --service-role-arn arn:aws:iam::402689298468:role/codedeploy-service-role
{
    "deploymentGroupId": "0752300d-7e48-45b5-b0e6-4fb26E92efda"
}
[ec2-user@ip-10-200-10-42 WordPress]$
```

- CodeDeploy 배포 그룹이 생성됨



참고) CodeDeploy 배포 그룹

- EC2/온프레미스 배포에서 배포 그룹은 **배포 대상인 개별 인스턴스의 집합입니다.** 배포 그룹에는 개별적으로 태그가 지정된 인스턴스 또는 Amazon EC2 Auto Scaling 그룹의 Amazon EC2 인스턴스 또는 둘 다가 포함됩니다. AWS Lambda 배포에서 배포 그룹은 향후 서비스 Lambda 배포를 위한 AWS CodeDeploy 구성 세트를 정의합니다.

2. 다음 create-deployment 명령을 터미널 세션에 붙여넣기

```
aws deploy create-deployment --application-name WordPress_App --s3-location bucket={BUCKET 이름},key=WordPressApp.zip,bundleType=zip --deployment-group-name WordPress_DG --description "This is a revision for the application WordPress_App"
```



명령어 해석)

이 명령어를 실행하면 "WordPress_App" 응용 프로그램의 "WordPress_DG" 배포 그룹에 대한 새로운 배포가 생성됩니다. 이 배포는 지정된 S3 버킷에서 "WordPressApp.zip" 파일을 가져와서 배포 그룹에 배포됩니다.

- `--application-name WordPress_App` : 배포할 응용 프로그램의 이름을 지정 합니다.
- `--s3-location bucket={BUCKET 이름},key=WordPressApp.zip,bundleType=zip` : 배포할 애플리케이션의 코드가 포함된 S3 버킷과 파일 경로를 지정합니다. 이 경우 "WordPressApp.zip" 파일이 지정된 버킷에서 사용됩니다.
- `--deployment-group-name WordPress_DG` : 새로운 배포를 위해 사용할 배포 그룹의 이름을 지정합니다.
- `--description "This is a revision for the application WordPress_App"` : 배포에 대한 설명을 추가합니다.

```
[ec2-user@ip-10-200-10-42 WordPress]$ aws deploy create-deployment --application-name WordPress_App --s3-location bucket=codedeploybucket0811,key=WordPressApp.zip,bundleType=zip --deployment-group-name WordPress_DG --description "This is a revision for the application WordPress_App"
{
    "deploymentId": "d-C70BE10M4"
}
[ec2-user@ip-10-200-10-42 WordPress]$
```

- 이렇게 하면 WordPress 애플리케이션이 배포됨

태스크 5: 배포 모니터링

이 태스크에서는 AWS CodeDeploy 관리 콘솔을 사용하여 **CodeDeploy 배포를 모니터링** 합니다.

AWS는 다양한 도구를 제공하는데, **자동으로 모니터링을 수행하도록 구성할 수 있는 도구와 수동 작업이 필요한 도구가 있습니다. 모니터링 태스크를 최대한 자동화하는 것이 좋습니다.**

- **자동 모니터링 도구**를 사용하여 AWS CodeDeploy를 관찰하고 문제 발생 시 보고할 수도 있습니다.
 - **Amazon CloudWatch 경보** - 지정한 기간 동안 단일 지표를 감시하고 여러 기간에 걸쳐 지정된 임계값 대비 지표 값을 기준으로 하나 이상의 작업을 수행합니다. 이 작업은 Amazon SNS) 주제 또는 Amazon EC2 Auto Scaling 정책에 전송되는 알림입니다. CloudWatch 경보는 특정 상태에 있다는 이유만으로는 작업을 호출하지 않습니다. 상태가 변경되고 지정한 기간 동안 유지되어야 합니다.
 - **Amazon CloudWatch Events** – 이벤트를 일치시키고 하나 이상의 대상 함수 또는 스트림으로 라우팅하여 값을 변경하거나 상태 정보를 캡처하거나 수정 작업을 수행합니다.
 - **AWS CloudTrail Log Monitoring** - 계정 간에 로그 파일을 공유하고, CloudTrail 로그 파일을 CloudWatch Logs에 전송하여 실시간으로 모니터링하며, Java에서 로그 처리 애플리케이션을 작성하고, CloudTrail에서 전송한 후 로그 파일이 변경되지 않았는지 확인합니다.
 - **Amazon Simple Notification Service** - 배포 및 인스턴스 이벤트(예: 성공 또는 실패)에 대한 SMS 또는 이메일 알림을 수신하도록 이벤트 중심 트리거를 구성합니다.

1. AWS 관리 콘솔 상단의 검색 창에서 CodeDeploy를 검색하여 선택합니다.
2. 왼쪽 배포 창 > 생성되어 있는 배포 ID 선택 > **Deployment lifecycle events** 페이지 까지 아래로 스크롤합니다.
 - 여기에서 애플리케이션을 배포하는 데 사용된 인스턴스 ID를 볼 수 있습니다.

배포 수명 주기 이벤트						
인스턴스 ID	기간	상태	가장 최근 이벤트	이벤트	시작 시간	종료 시간
i-0291539fb0cf04cf5	19초	성공	ValidateService	View events	4월 6, 2024 4:52 오후 (UTC+9:00)	4월 6, 2024 4:52 오후 (UTC+9:00)

3. **Instance ID** 링크를 클릭합니다.

- 그러면 애플리케이션을 배포하는 데 사용된 인스턴스로 이동합니다. 보시다시피 애플리케이션이 CodeDeploy 인스턴스에 배포되었습니다.

인스턴스 (1) 정보

인스턴스 ID = i-0291539fb0cf04cf5

모든 상태 ▾

필터 지우기

Name	인스턴스 ID	인스턴스 상태	인스턴스 유형	상태 검사	경보 상태	가용 영역
CodeDeploy	i-0291539fb0cf04cf5	실행 중	t3.micro	2/2개 검사 통과...	경보 보기	us-west-2a

4. EC2 인스턴스의 탭을 닫습니다.

5. CodeDeploy 탭에서 **View events** 링크를 클릭합니다.

- 이 페이지에서 배포의 모든 이벤트를 모니터링할 수 있습니다.

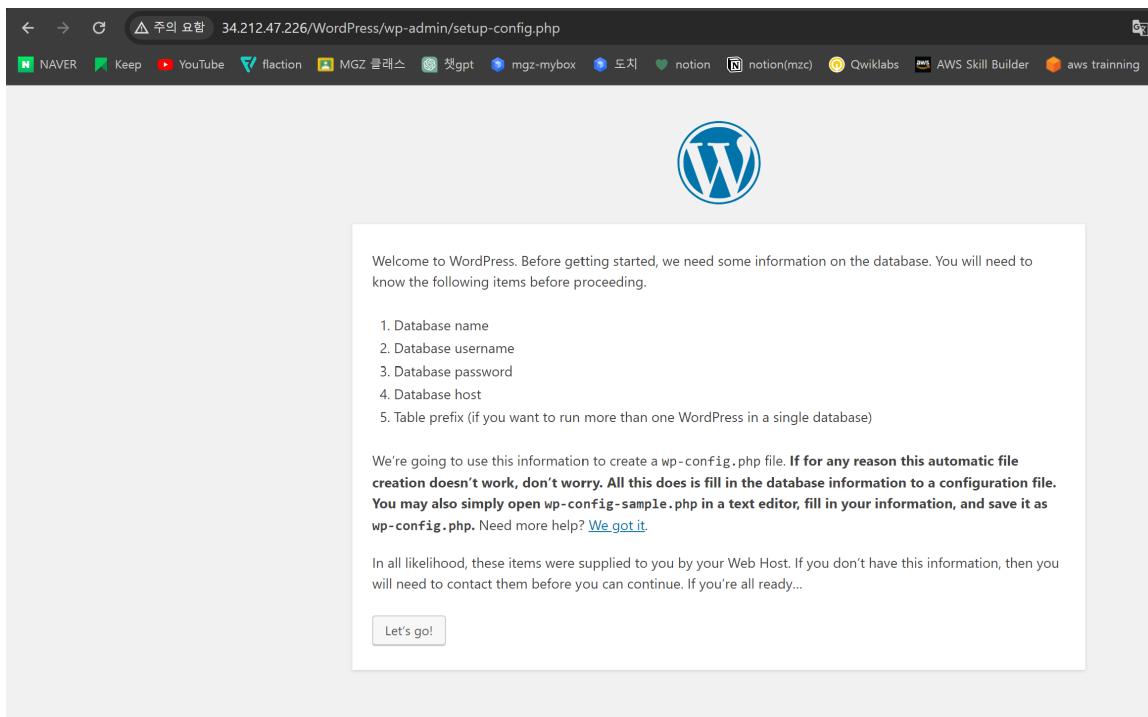
이벤트	기간	상태	오류 코드	시작 시간	종료 시간
ApplicationStop	1초 미만	성공	-	4월 6, 2024 4:52 오후 (UTC+9:00)	4월 6, 2024 4:52 오후 (UTC+9:00)
DownloadBundle	3초	성공	-	4월 6, 2024 4:52 오후 (UTC+9:00)	4월 6, 2024 4:52 오후 (UTC+9:00)
BeforeInstall	8초	성공	-	4월 6, 2024 4:52 오후 (UTC+9:00)	4월 6, 2024 4:52 오후 (UTC+9:00)
Install	1초 미만	성공	-	4월 6, 2024 4:52 오후 (UTC+9:00)	4월 6, 2024 4:52 오후 (UTC+9:00)
AfterInstall	1초 미만	성공	-	4월 6, 2024 4:52 오후 (UTC+9:00)	4월 6, 2024 4:52 오후 (UTC+9:00)
ApplicationStart	3초	성공	-	4월 6, 2024 4:52 오후 (UTC+9:00)	4월 6, 2024 4:52 오후 (UTC+9:00)
ValidateService	1초 미만	성공	-	4월 6, 2024 4:52 오후 (UTC+9:00)	4월 6, 2024 4:52 오후 (UTC+9:00)

6. 왼쪽 탐색 창에서 **Deployment**을 클릭합니다.

7. Deployment status에 **Succeeded**가 표시될 때까지 기다립니다.

8. 다음을 수행하여 WordPress 애플리케이션에 액세스할 수 있는지 확인합니다.

- 이 지침의 왼쪽에 있는 **EC2PublicIP** 값 복사
- /WordPress 추가 해서 새 브라우저 탭에 붙여넣기



태스크 6: WordPress 애플리케이션 업데이트 및 배포

애플리케이션 수정 버전의 배포를 마쳤으므로 이제 개발 머신에서 WordPress 코드를 업데이트한 다음 AWS CodeDeploy를 사용하여 사이트를 재배포합니다. 이후에는 Amazon EC2 인스턴스에서 코드 변경이 표시되어야 합니다.

1. WORDPRESS 사이트 설정 완료

1. **WordPress** 사이트에서 **Let's go!**를 클릭하고 다음을 구성합니다.

- *Database Name:* test
- *Username:* root
- *Password:* 암호를 삭제하여 필드를 비웁니다.
- **Submit**을 클릭합니다.

2. **Run the installation**을 클릭합니다.

3. **Welcome** 페이지에서 다음을 구성합니다.

- **Site Title:** mySite
- **username:** user1
- **password:** 기억할 수 있는 암호를 입력합니다.

- **Your Email:** 이메일 주소를 입력합니다.
- **Install WordPress**를 클릭합니다.

Welcome

Welcome to the famous five-minute WordPress installation process! Just fill in the information below and you'll be on your way to using the most extendable and powerful personal publishing platform in the world.

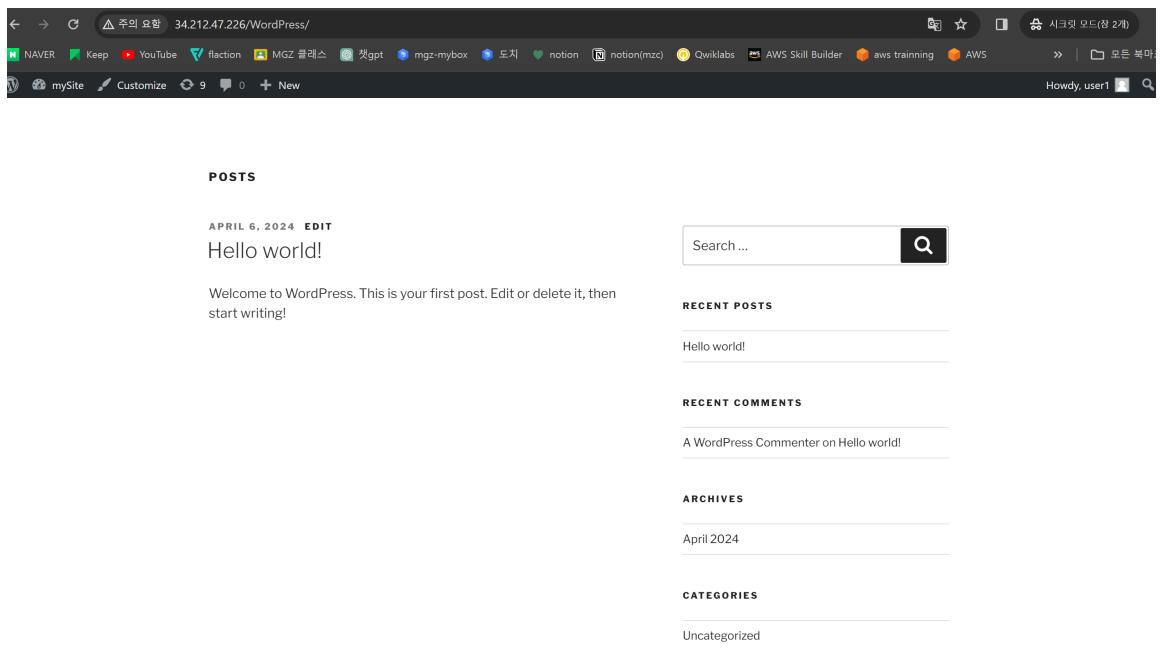
Information needed

Please provide the following information. Don't worry, you can always change these settings later.

Site Title	mySite
Username	user1
Usernames can have only alphanumeric characters, spaces, underscores, hyphens, periods, and the @ symbol.	
Password	1234 Very weak
Important: You will need this password to log in. Please store it in a secure location.	
Confirm Password	<input checked="" type="checkbox"/> Confirm use of weak password
Your Email	kmg4love@naver.com
Double-check your email address before continuing.	
Search Engine Visibility	<input type="checkbox"/> Discourage search engines from indexing this site It is up to search engines to honor this request.

Install WordPress

4. WordPress가 설치되면 WordPress 사이트에 로그인합니다.
5. 화면 왼쪽 상단에서 커서로 mySite를 가리킨 다음 **Visit Site**를 클릭합니다.



- 흰색 배경임

2. WORDPRESS 사이트 수정 및 다시 배포

이 과제에서는 WordPress 사이트를 수정한 다음 CodeDeploy를 사용하여 재배포합니다.

WordPress 애플리케이션 배포 중에 `change_permissions.sh` 스크립트는 누구나 `/tmp/WordPress` 폴더에 쓸 수 있도록 이 폴더의 권한을 업데이트했습니다.

1. AWS Systems Manager - Session manager 탭에서 다음 명령을 실행하여 소유자인 자신만이 쓸 수 있도록 권한을 제한합니다.

```
sudo chmod -R 755 /var/www/html/WordPress
```

2. WordPress 디렉터리로 이동합니다.

```
cd /tmp/WordPress
```

3. `wp-content/themes/twentyfifteen/style.css` 파일에서 사이트 색상 일부를 수정합니다.

```
# sed 명령은 이 색상(#fff)을 #768331로 변경함
sed -i 's/#fff/#768331/g' wp-content/themes/twentyseventee
```

4. AWS CodeDeploy 푸시 명령을 호출하여 업데이트된 파일을 함께 묶고, 수정 버전을 Amazon S3에 업로드하고, 다음을 통해 업로드된 수정 버전에 대한 정보를 AWS

CodeDeploy에 등록합니다.

```
aws deploy push --application-name WordPress_App --s3-location s3://{{BUCKET 이름}}/WordPressApp.zip --ignore-hidden-files
```

5. AWS 관리 콘솔 상단의 검색 창에서 CodeDeploy를 검색하여 선택합니다.
6. 왼쪽 탐색 창에서 **Applications**를 클릭합니다.
7. **WordPress_App**을 클릭합니다.
8. **Revisions** 탭을 클릭합니다.
9. 아직 배포되지 않은 수정 버전을 선택합니다. 이것이 최신 버전입니다.

The screenshot shows the AWS CodeDeploy console for the 'WordPress_App' application. The 'Revisions' tab is selected. A table lists three revisions:

개정 위치	처음 배포 날짜	마지막 배포 날짜	생성됨
s3://codedeploy...	-	-	4월 6, 2024 5...
s3://codedeploy...	4월 6, 2024 4:51 ...	4월 6, 2024 4:51 ...	4월 6, 2024 4:...
s3://codedeploy...	-	-	4월 6, 2024 4:...

10. **Deploy application**을 클릭합니다.

11. **Create deployment** 창에서 다음을 구성합니다.

- *Deployment group: WordPress_DG*
- 화면 맨 아래로 스크롤한 다음 **Create deployment**를 클릭합니다.

12. **Deployment status**에 **Succeeded**가 표시될 때까지 기다립니다.

The screenshot shows the AWS CodeDeploy console under the 'Deployment history' tab. A deployment entry is highlighted with a red box:

배포 ID	상태	배포 유형	컴퓨팅 플랫폼	애플리케이션
d-Q7JFGCIN4	성공	현재 위치	EC2/온프레미스	WordPress_App
d-3ALLIHHN4	성공	현재 위치	EC2/온프레미스	WordPress_App

13. WordPress 탭으로 돌아가서 화면을 새로 고칩니다.

- 이제 사이트의 배경이 녹색이어야 합니다.

⇒ 페이지가 녹색으로 변하지 않음 ✗