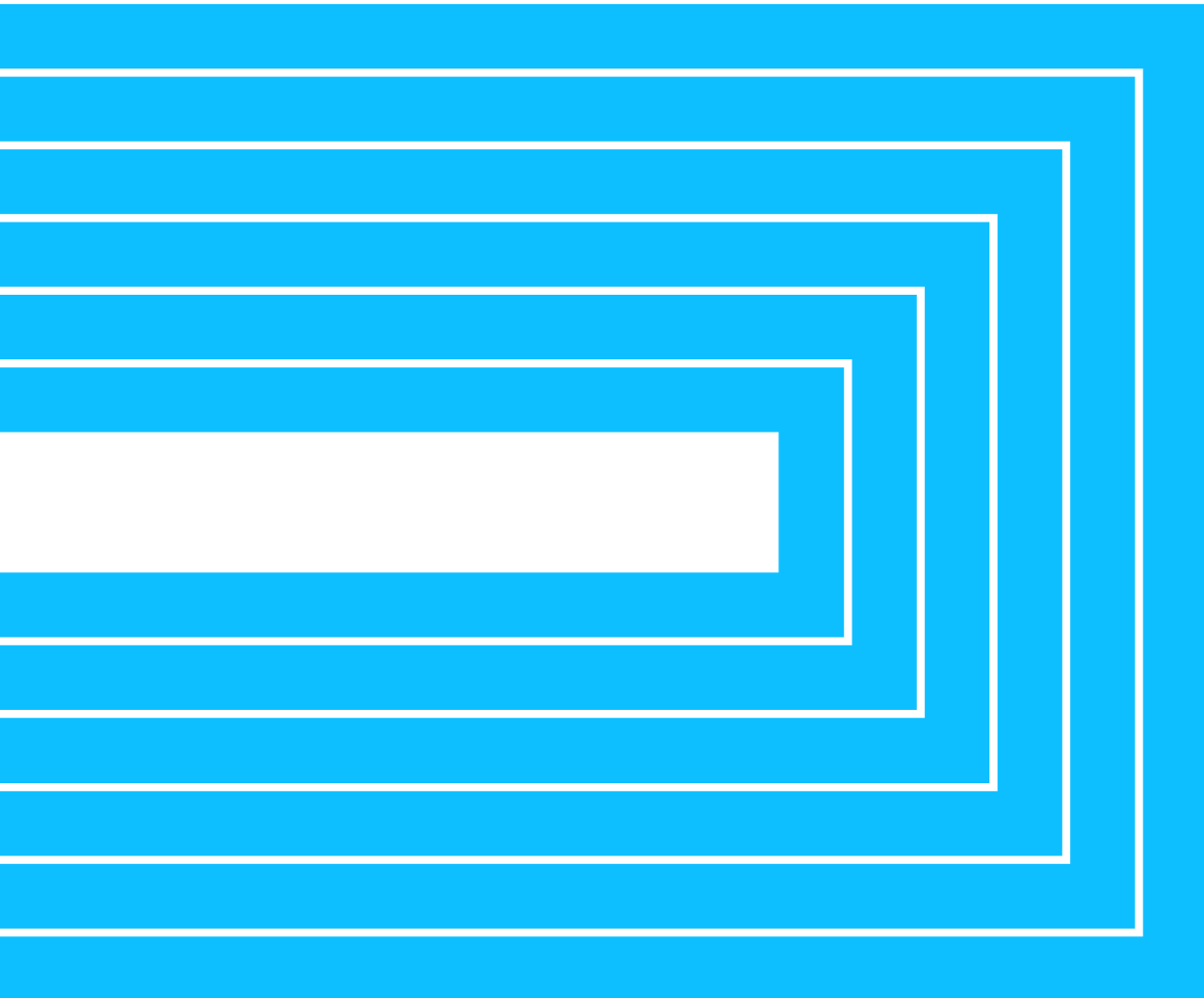


정찬혁

AWS 클라우드 환경에서 cicd파이프라인구현



AWS] CI/CD란



CI(Continuous Integration)/CD(Continuous Delivery/Continuous Deployment)란, 어플리케이션 개발에 필요한 여러 단계에 대한 자동화를 통해 어플리케이션을 보다 빠르고 짧은 주기로 고객에게 제공하는 방법입니다. CI/CD의 개념은 지속적인 통합(Continuous Integration), 지속적인 서비스 제공(Continuous Delivery) 및 지속적인 배포(Continuous Deployment)를 통해 새로운 코드의 통합, 테스트, 릴리스, 배포 등의 어플리케이션 라이프사이클 전체에 대한 자동화 과정을 모니터링 가능하도록 하는 것을 말합니다.

CI(Continuous Integration)



CI는 Continuous Integration의 약자로 지속적인 통합을 의미합니다. 개발자를 위한 자동화 프로세스를 통해 새로운 코드 개발과, 코드의 변경 사항이 정기적으로 빌드 및 테스트되고 공유 레포지토리에 병합되어 여러 명의 개발자가 동시에 어플리케이션 개발과 관련된 코드를 작업할 경우에도 서로 충돌없이 원하는 개발 작업을 수행하고 문제를 해결이 가능합니다.

다시 말해, 클래스와 기능에서부터 전체 어플리케이션을 구성하는 서로 다른 모듈에 이르기까지 모든 것에 대한 테스트를 수행합니다. 자동화된 테스트에서 기존 코드와 신규 코드 간의 충돌이 발견되면 CI를 통해 이러한 버그를 더욱 빠르게 확인할 수 있고, 이렇게 확인된 소스를 수정하여 원활하게 배포할 수 있습니다.

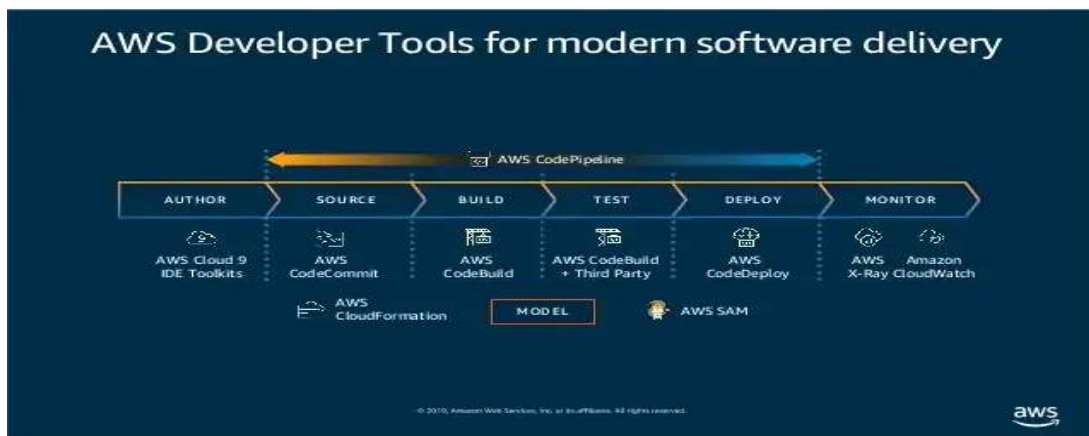
CD(Continuous Delivery/Continuous Deployment)



CD는 지속적인 전달(Continuous Delivery) 및 지속적인 배포(Continuous Deployment)의 의미를 가지고 있는데, 두가지 용어를 혼용하여 사용합니다. 개발자가 지속적인 서비스 전달과 배포를 통해 새로운 코드에 대한 배포를 자동화할 수 있으며, 이로써 신속한 어플리케이션을 제공할 수 있도록 도움을 주는 역할을 합니다.

CI 를 통한 빌드의 자동화 및 유닛의 통합 테스트 수행 이후 이어지는 CD 프로세스는 유효한 소스코드를 레포지토리에 자동으로 전달합니다. 따라서 효과적인 CD의 프로세스를 실현하기 위해서는 개발 파이프라인 CI가 먼저 구축되어야 합니다. 이러한 지속적 제공의 목표는 운영 환경으로 배포할 준비가 되어 있는 코드베이스를 확보하는 것에 있습니다.

이러한 CD 프로세스를 수행하면 보다 빠르고 손쉽게 어플리케이션을 운영 환경으로 배포할 수 있습니다.

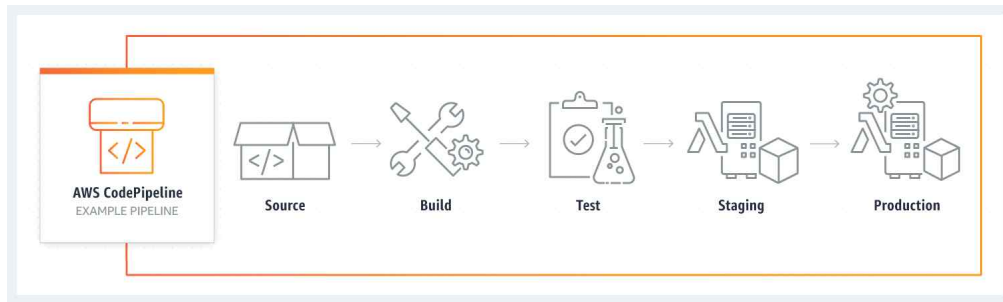


AWS DevOps Tools		
Service	Solution	Description
AWS CodeCommit	버전 관리	소스코드를 개인 Git 저장소에 안전하게 저장
AWS CodeBuild	CI/CD	연속적인 스케일링으로 코드 빌드 및 테스트에 사용
AWS CodeDeploy	CI/CD	소스코드의 자동 배포
AWS CodePipeline	CI/CD	지속적인 통합/지속적인 전달(CI/CD) 서비스
AWS CodeStar	CI/CD	템플릿 기반의 신속한 어플리케이션 개발, 구축 및 배포
AWS X-Ray	모니터링	어플리케이션의 디버그 및 분석, 모니터링 수행
AWS CLI	명령 인터페이스	Command를 기반으로 AWS 리소스 관리
AWS Cloud9	개발 툴	웹 브라우저를 기반으로 Cloud IDE를 활용한 개발 및 실행, 디버깅을 위한 개발 툴

AWS 클라우드 환경에서 CICD 파이프라인 구현

제일 중요한건 레포지토리를 생성해서 그안에 변경된 코드가 있으면 그거를 통해서 애플리케이션에 적용하는 구조를 만드는것이 중요하며 어떤서비스, 레포지토리를 사용하는건 자유이다.

aws codepipeline



CodeCommit, CodeBuild, CodeDeploy를 하나의 프로세스로 통합시켜주는 CI/CD도구 즉 코드 변경이 있을 때 마다 사용자가 사전에 정의한 릴리스 모델을 기반으로 빌드,테스트, 배포 단계를 자동화 한다.

1)

Codepipeline은 내가 작성한 코드가 릴리즈가 되면 새로운 Docker 이미지를 빌드해줍니다.

2)

새로이 빌드된 이미지는 ECR에 저장되어서 현재 우리가 유지하고 있는 ECS 클러스터 서버에 배포

3)

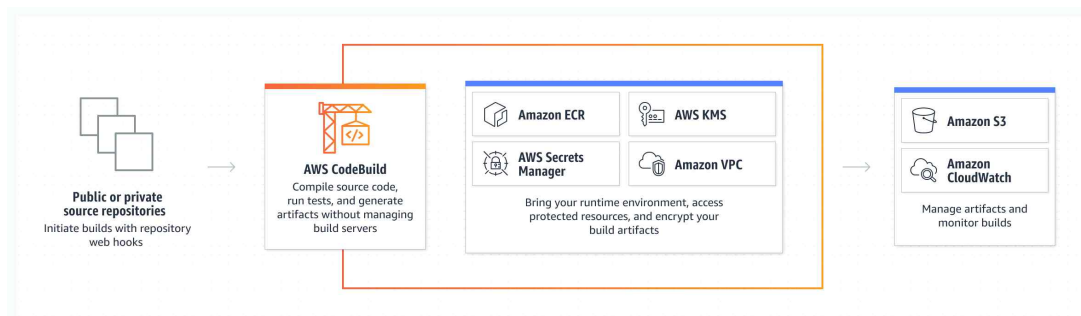
소스코드 저장소에 push만 하게 되면 빌드 - 테스트 - 배포가 AWS에 의해서 진행됩니다.

aws code commit

소스코드를 개인 git 저장소에 안전하게 저장

aws code build

소스코드를 컴파일하고 테스트를 실행한 다음 바로 배포 가능한 소프트웨어 패키지를 생성



aws code deploy

소스코드의 자동 배포

사용 서비스

- 1) EC2 인스턴스
 - 2) Amazon Elastic Container Registry
 - 3) Amazon Elastic Container Service
 - 4) AWS CodePipeline
(CodeCommit-CodeBuild-CodePipeline)
- *) IAM

실습에 앞서)

이 문서는 <https://www.youtube.com/watch?v=d7PTjQiahOQ>의 내용을 기반으로 따라 만든 것으로 영상과 함께 문서를 참고해 실습하는 것을 추천한다. 이 문서는 과정에 대한 이해와 개인 학습용도를 통해 만든것으로 추가적인 내용 및 실습과정에 대한 자세한 정보는 영상에 들어있다.

목표)

우선 무엇을 구현하는지 그것에 대한 이해를 해야한다. 결과를 말하자면 이 문서는 웹 사이트를 컨테이너를 통해 구현해 제공한뒤 codepipeline을 통해 개발자가 코드를 수정을 했을 때 자동으로 적용되는 CI/CD를 구현하는것이다. 즉 완성됐을 때 **개발자가 웹페이지의 내용을 바꾸면 컨테이너에서 제공하는 웹페이지의 내용이 자동으로 바뀌어야한다.**

<전체적인 그림>

시험용으로 쓰게 될 간단한 인스턴스 생성

실습에 쓰게될 간단한 웹페이지 배포 확인 혹은 코드 commit등 기본 베이스로 작업하며 쓰게 될 운영체제를 만들어둔다. (git bash EC2 인스턴스등 자유롭게 사용)실습은 EC2인스턴스

The screenshot shows the AWS 'New EC2 Experience' interface. On the left, a sidebar contains navigation links: 'EC2 대시보드', 'EC2 글로브 보기', '이벤트', '태그', '제한', '인스턴스' (highlighted), '인스턴스 유형', '시작 템플릿', '스팟 요청', 'Savings Plans', '예약 인스턴스', '전용 호스트', '용량 예약', '이미지', 'AMI', 'AMI 카탈로그', 'Elastic Block Store', '볼륨', '스냅샷', '수명 주기 관리자', '네트워크 및 보안', '보안 그룹', '탄력적 IP', '배치 그룹'. The main panel is titled '인스턴스 시작' (Start Instance) and includes a 'Name and tags' section with a text input 'e.g. My Web Server' and an 'Add additional tags' link. Below this is the 'Application and OS image (Amazon Machine Image)' section, which includes a search bar and a 'Quick Start' section displaying various AMI options: Amazon Linux, macOS, Ubuntu, Windows, Red Hat, and SUSI. The 'Amazon Linux 2023 AMI' is selected and highlighted. Below the AMI selection, there is a section for 'Amazon Machine Image(AMI)' showing details for 'Amazon Linux 2023 AMI' (ami-0a306845f8cfd41a) and 'Amazon Linux 2023 AMI' (ami-08ae25f781ed86943).

웹에서 EC2 인스턴스 연결, 혹은 putty같은 프로그램으로 SSH클라이언트 연결을 사용한다.

The screenshot shows the 'Connect to Instance' page in the AWS Management Console. The 'EC2 인스턴스 연결' (Connect to Instance) tab is selected. The page displays the instance ID 'i-0e34815fb72280ced (testchan)', the public IP address '13.209.41.76', and the user name 'ec2-user'. A warning message states: '참고: 대부분의 경우 기본 사용자 이름 ec2-user(는) 정확합니다. 하지만 AMI 사용 지침을 읽고 AMI 소유자가 기본 AMI 사용자 이름을 변경했는지 확인하십시오.' (Note: In most cases, the default user name ec2-user (is) correct. However, read the AMI usage instructions and confirm whether the AMI owner has changed the default AMI user name.) The page includes buttons for '취소' (Cancel) and '연결' (Connect).

루트사용자로 변환 및 docker 설치

```
# sudo -i
```

```
# yum install docker
```

```
ec2-user@ip-172-31-33-37 ~]$ sudo -i
root@ip-172-31-33-37 ~]#
root@ip-172-31-33-37 ~]# yum update
Last metadata expiration check: 0:01:33 ago on Sun Apr 30 04:25:48 2023.
Dependencies resolved.
Nothing to do.
Complete!
root@ip-172-31-33-37 ~]# yum install docker
Last metadata expiration check: 0:02:23 ago on Sun Apr 30 04:25:48 2023.
Dependencies resolved.

=====
Package                                Architecture      Version
=====
Installing:
docker                                x86_64            20.10.17-1.amzn2023.0.6
Installing dependencies:
containerd                            x86_64            1.6.19-1.amzn2023.0.1
iptables-libs                         x86_64            1.8.8-3.amzn2023.0.2
iptables-nft                          x86_64            1.8.8-3.amzn2023.0.2
libcgroup                             x86_64            3.0-1.amzn2023.0.1
libnetfilter_conntrack                x86_64            1.0.8-2.amzn2023.0.2
libnftnl                             x86_64            1.0.1-19.amzn2023.0.2
libnftnl                             x86_64            1.2.2-2.amzn2023.0.2
pigz                                   x86_64            2.5-1.amzn2023.0.3
runc                                   x86_64            1.1.4-1.amzn2023.0.1
=====

Transaction Summary
-----
Install 10 Packages

Total download size: 74 M
Installed size: 287 M
Is this ok [y/N]: y
Downloading Packages:
(1/10): libnftnl-1.0.1-19.amzn2023.0.2.x86_64.rpm
(2/10): pigz-2.5-1.amzn2023.0.3.x86_64.rpm
(3/10): libnftnl-1.2.2-2.amzn2023.0.2.x86_64.rpm
(4/10): runc-1.1.4-1.amzn2023.0.1.x86_64.rpm
(5/10): libnetfilter_conntrack-1.0.8-2.amzn2023.0.2.x86_64.rpm
(6/10): libcgroup-3.0-1.amzn2023.0.1.x86_64.rpm
(7/10): iptables-libs-1.8.8-3.amzn2023.0.2.x86_64.rpm
(8/10): iptables-nft-1.8.8-3.amzn2023.0.2.x86_64.rpm
(9/10): containerd-1.6.19-1.amzn2023.0.1.x86_64.rpm
(10/10): docker-20.10.17-1.amzn2023.0.6.x86_64.rpm
-----
Total
Running transaction check
```

도커 데몬 실행 및 확인

```
# systemctl start docker (=service docker start)
```

```
# systemctl status docker (=service docker status)
```

```
[root@ip-172-31-33-37 ~]# systemctl status docker
● docker.service - Docker Application Container Engine
   Loaded: loaded (/usr/lib/systemd/system/docker.service; disabled; preset: disabled)
   Active: active (running) since Sun 2023-04-30 04:29:14 UTC; 34s ago
 TriggeredBy: ● docker.socket
    Docs: https://docs.docker.com
   Process: 26361 ExecStartPre=/bin/mkdir -p /run/docker (code=exited, status=0/SUCCESS)
   Process: 26362 ExecStartPre=/usr/libexec/docker/docker-setup-runtimes.sh (code=exited, status=0/SUCCESS)
  Main PID: 26364 (dockerd)
    Tasks: 7 (limit: 1112)
   Memory: 22.4M
      CPU: 275ms
   CGroup: /system.slice/docker.service
           └─26364 /usr/bin/dockerd -H fd:// --containerd=/run/containerd/containerd.sock --default-ulimit nofil

Apr 30 04:29:13 ip-172-31-33-37.ap-northeast-2.compute.internal dockerd[26364]: time="2023-04-30T04:29:13.84962366
Apr 30 04:29:13 ip-172-31-33-37.ap-northeast-2.compute.internal dockerd[26364]: time="2023-04-30T04:29:13.84970484
Apr 30 04:29:13 ip-172-31-33-37.ap-northeast-2.compute.internal dockerd[26364]: time="2023-04-30T04:29:13.85237310
Apr 30 04:29:13 ip-172-31-33-37.ap-northeast-2.compute.internal dockerd[26364]: time="2023-04-30T04:29:13.89976884
```

도커에서 이미지를 가져와서 컨테이너 제작해가며 확인

실습에 앞서 우선 docker에서 설치시 각종 에러 가 뜰 수도 있다.

```
$ Error: Failed to download metadata for repo 'appstream': Cannot prepare internal mirrorlist: No URLs in mirrorlist
```

--> CentOS를 위한 패키지 저장소의 주소가 잘못되었거나 주소에 접속할 수 없기 때문에 발생하는 문제로 근본적인 문제는 centos8의 EOL 즉 지원종료로 일어난 문제이다.

- CentOS 8 - 2021년 12월 31일
- CentOS 7 - 2024년 6월 30일

해결 방법으로는

- 1) Centos7 사용
- 2) CentOS Stream 사용
- 3) repository를 변경하여 사용

영상에서 centos6 버전을 사용하기 때문에 그에 관련한 repository 변경 과정은(repository 주소를 vault로 변경) 아래의 사이트 참고.

<https://hsunnystory.tistory.com/211>

```
$ Failed to get D-Bus connection: Operation not permitted
```

--> docker run을 통해 명령 안에서 systemctl 명령을 썼을 때 뜰 수 있는 오류인데

docker을 run 할 때 Privileged 모드 선언을 하거나 혹은

/sys/fs/cgroup:/sys/fs/cgroup:rw, sbin/init으로 실행 후 컨테이너 접속 등의 방법을 사용하면 된다.

(systemd는 /sys/fs/cgroup 아래에 cgroupfs들을 마운트 하는데 이리 선언해서 사용할 수있다.

혹은 systemd 가 깔려 설정되어있는 이미지를 사용해도 된다.)

*** 사실 dockerfile을 통해 이미지를 제작해 서비스를 제공할때는 그것에 필요한 것만 최소한으로 넣어 크기를 줄여야 하기 때문에(실습에 경우 웹과 같은) 필요한 패키지만 넣어 만들어야한다.** - 실습에서는 httpd를 설치하기위해 yum과 같은 작업만 필수적이지 사실 systemd와 clear등의 명령어는 편의성을 넣기위해 추가 한것으로 간단하게 httpd만 넣어도 무방하다.

centos 이미지 가져오기

```
# docker pull tgagor/centos-stream
```

(실행되었는지 확실하게 실행하기위해서 systemd가 설치된 이미지파일을 가져왔다.)

```
# docker images
```

```
[root@ip-172-31-33-37 ~]# docker pull centos:centos7
centos7: Pulling from library/centos
2d473b07cdd5: Pull complete
Digest: sha256:be65f488b7764ad3638f236b7b515b3678369a5124c47b8d32916d6487418
Status: Downloaded newer image for centos:centos7
docker.io/library/centos:centos7
[root@ip-172-31-33-37 ~]#
[root@ip-172-31-33-37 ~]# docker images
REPOSITORY      TAG          IMAGE ID      CREATED        SIZE
centos           centos7      eeb6ee3f44bd  19 months ago  204MB
[root@ip-172-31-33-37 ~]#
```

컨테이너 내부로 들어가 작업하면서 Docerfile에 들어갈 명령정리

```
# docker run -it -p 80:80 [이미지 이름]
```

```
(docker run --privileged --name test --cgroupns=host -v /sys/fs/cgroup:/sys/fs/cgroup:rw
-p 80:80 -d tgagor/centos-stream /sbin/init)
```

```
(docker exec -it test /bin/bash)
```

```
[root@ip-172-31-33-37 ~]# docker images
REPOSITORY      TAG          IMAGE ID      CREATED        SIZE
tgagor/centos-stream  latest      36c2bda5b9ae  6 days ago     220MB
centos           7           eeb6ee3f44bd  19 months ago  204MB
centos           centos7      eeb6ee3f44bd  19 months ago  204MB
centos/systemd    latest      05d3c1e2d0c1  4 years ago    202MB
[root@ip-172-31-33-37 ~]#
[root@ip-172-31-33-37 ~]# docker run --privileged --name test --cgroupns=host -v /sys/fs/cgroup:/sys/fs/cgroup:rw -p 80:80 -d tgagor/centos-stream /sbin/init
c36f6edec05bac132a2af4cb2fdb283b71994143b3344276c004783c81141284
[root@ip-172-31-33-37 ~]#
[root@ip-172-31-33-37 ~]# docker exec -it test /bin/bash
root@c36f6edec05b /#
root@c36f6edec05b /# ls
bin dev etc home lib lib64 lost+found media mnt opt proc root run sbin srv sys tmp usr var
root@c36f6edec05b /# systemctl status
```

[참고] 작업하는데 clear 명령어가 없다면

```
# yum -y install ncurses
```

httpd 패키지 설치

```
# yum install httpd
```

간단한 웹페이지 생성

```
# cd /var/www/html (index.html을 읽게될 기본 폴더위치)
```

```
# echo "test" >> index.html
```

데몬 재시작 및 상태 확인

```
# systemctl start OR restart httpd
```

```
# systemctl status httpd
```

```
[root@c36f6edec05b /]# cd /var/www/html
[root@c36f6edec05b html]#
[root@c36f6edec05b html]# ls
[root@c36f6edec05b html]# echo "test" >> index.html
[root@c36f6edec05b html]#
[root@c36f6edec05b html]# systemctl start httpd
[root@c36f6edec05b html]# systemctl status httpd
● httpd.service - The Apache HTTP Server
   Loaded: loaded (/usr/lib/systemd/system/httpd.service; disabled; vendor preset: disabled)
   Active: active (running) since Sun 2023-04-30 05:10:42 UTC; 5s ago
     Docs: man:httpd.service(8)
  Main PID: 195 (httpd)
   Status: "Started, listening on: port 80"
    Tasks: 213 (limit: 5933)
   Memory: 16.4M
    CGroup: /system.slice/docker-c36f6edec05bac132a2af4cb2fdb283b71994143b3344276c004783c81141284.scope/system.slice/httpd.service
            └─195 /usr/sbin/httpd -DFOREGROUND
              └─196 /usr/sbin/httpd -DFOREGROUND
                └─197 /usr/sbin/httpd -DFOREGROUND
                  └─198 /usr/sbin/httpd -DFOREGROUND
```

public ip주소로 요청해서 만든 웹페이지가 외부에서 보이냐 확인

The screenshot shows the AWS Management Console interface. At the top, there's a search bar with the text "인스턴스를 속성 또는 (case-sensitive) 태그로 찾기". Below it, a table lists instances, with "testchan" selected. The instance ID is "i-0e34815fb72280ced", status is "실행 중" (Running), and type is "t2.micro". Below the table, the "인스턴스: i-0e34815fb72280ced(testchan)" details are shown. The "세부 정보" (Details) tab is active, displaying various attributes. A red circle highlights the "퍼블릭 IPv4 주소" (Public IPv4 address) field, which shows "13.209.41.76" with a link to "개방 주소별" (Open address). Other fields include "인스턴스 ID", "IPv6 주소", "프라이빗 IPv4 주소", and "퍼블릭 IPv4 DNS".

주소창에 ip 주소 입력

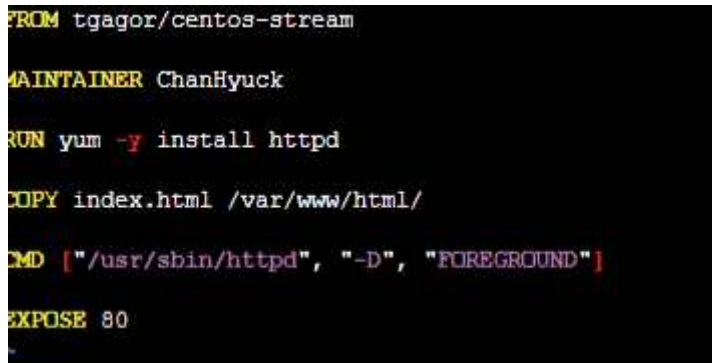
test

* 지금 여기 까지의 작업은 만든 컨테이너가 퍼블릭 ip 주소를 통해 웹 페이지가 잘 작동하
나 확인한 작업으로 이것을 토대로 Dockerfile을 제작해 이미지 파일을 만들 것이다.

Dockerfile 제작

vi Dockerfile

```
FROM tgagor/centos-stream
MAINTAINER ChanHyuck
RUN yum -y install httpd
COPY index.html /var/www/html/
CMD ["/usr/sbin/httpd", "-D", "FOREGROUND"]
EXPOSE 80
```



```
FROM tgagor/centos-stream
MAINTAINER ChanHyuck
RUN yum -y install httpd
COPY index.html /var/www/html/
CMD ["/usr/sbin/httpd", "-D", "FOREGROUND"]
EXPOSE 80
```

FROM 베이스 이미지

(앞에서 확실하게 돌아갔기 때문에 그냥 사용한 것으로 httpd만 돌아간다면 베이스로는 원하는 centos version을 아무거나 사용해도 괜찮다.)

MAINTAINER 이미지를 개발한 제작자 이름(필수는 아님)

RUN image가 올라갔을 때 실행되는 명령어들

COPY build 명령 중간에 호스트의 파일 또는 폴더를 이미지에 가져온다.

- index.html을 이미지안에 넣는 작업

CMD 컨테이너를 생성 및 실행 할 때 실행할 명령어

(보통 컨테이너 내부에서 항상 돌아가는 서버를 띄울 때 사용 - 지금은 웹)

실습에서 쓸 웹페이지 제작

vi index.html

```
<html>
  <body>
    <h1> CICD & Docker example ChanHyuck </h1>
    <p style="background-color:MediumSeaGreen;"> Version one (V1.0) in Green Color.
  </p>
  </body>
</html>
```

```
<html>
  <body>
    <h1> CICD & Docker example ChanHyuck </h1>
    <p style="background-color:MediumSeaGreen;"> Version one (V1.0) in Green Color. </p>

  </body>
</html>
```

위에서 요청한것처럼 컨테이너쪽으로 다시 웹 페이지 요청

CICD & Docker example ChanHyuck

Version one (V1.0) in Green Color.

Dockerfile을 통해 image 제작

```
# docker build -t chanwebserver .
```

```
[root@ip-172-31-33-37 docker]# ls
Dockerfile index.html
[root@ip-172-31-33-37 docker]# docker ps -a
CONTAINER ID   IMAGE          COMMAND                  CREATED        STATUS        PORTS          NAMES
c36f6edec05b   tgagor/centos-stream   "/sbin/init"          51 minutes ago   Exited (137) 7 seconds ago
[root@ip-172-31-33-37 docker]# docker images
REPOSITORY      TAG          IMAGE ID      CREATED        SIZE
tgagor/centos-stream   latest      36c2bda5b9ae   6 days ago    220MB
centos           7           eeb6ee3f44bd   19 months ago 204MB
centos           centos7     eeb6ee3f44bd   19 months ago 204MB
centos/systemd     latest     05d3cle2d0c1   4 years ago   202MB
[root@ip-172-31-33-37 docker]#
[root@ip-172-31-33-37 docker]# docker build -t chanwebserver .
Sending build context to Docker daemon 3.072kB
Step 1/6 : FROM tgagor/centos-stream
--> 36c2bda5b9ae
Step 2/6 : MAINTAINER ChanHyuck
--> Running in 75fa281f5dd2
Removing intermediate container 75fa281f5dd2
--> aa90008cecff
Step 3/6 : RUN yum -y install httpd
--> Running in b1b87ea7c71f
CentOS Stream 8 - AppStream          24 MB/s | 29 MB    00:01
CentOS Stream 8 - BaseOS            29 MB/s | 33 MB    00:01
CentOS Stream 8 - Extras             43 kB/s | 18 kB    00:00
CentOS Stream 8 - Extras common packages 11 kB/s | 6.2 kB  00:00
Dependencies resolved.
```

만들어진 이미지 확인하고 만들어진 이미지를 활용해 실행한 컨테이너로 정상적으로 작동하나 확인

```
[root@ip-172-31-33-37 docker]#
[root@ip-172-31-33-37 docker]# docker images
REPOSITORY      TAG          IMAGE ID      CREATED        SIZE
chanwebserver    latest      be58a729f15a   20 seconds ago 360MB
tgagor/centos-stream   latest      36c2bda5b9ae   6 days ago    220MB
centos           7           eeb6ee3f44bd   19 months ago 204MB
centos           centos7     eeb6ee3f44bd   19 months ago 204MB
centos/systemd     latest     05d3cle2d0c1   4 years ago   202MB
[root@ip-172-31-33-37 docker]# docker ps -a
CONTAINER ID   IMAGE          COMMAND                  CREATED        STATUS        PORTS          NAMES
c36f6edec05b   tgagor/centos-stream   "/sbin/init"          59 minutes ago   Exited (137) 7 minutes ago
[root@ip-172-31-33-37 docker]# docker run -it -d -p 80:80 chanwebserver
9809d3c3ec47e2alc6a122f1c9e2be4014e78dff5b0c4d4199ff3baf3589137
[root@ip-172-31-33-37 docker]#
[root@ip-172-31-33-37 docker]# docker ps -a
CONTAINER ID   IMAGE          COMMAND                  CREATED        STATUS        PORTS          NAMES
9809d3c3ec47   chanwebserver   "/usr/sbin/httpd -D _"   5 seconds ago   Up 3 seconds   0.0.0.0:80->80/tcp, :::80->80/tcp   eloquent_tharp
c36f6edec05b   tgagor/centos-stream   "/sbin/init"          About an hour ago   Exited (137) 8 minutes ago
[root@ip-172-31-33-37 docker]#
```

CICD & Docker example ChanHyuck

Version one (V1.0) in Green Color.

Amazon ECR repository 생성

리포지토리 생성

일반 설정

표시 여부 설정 **정보**
리포지토리에 대한 가시성 설정을 선택합니다.

☒ 프라이빗
액세스는 IAM 및 리포지토리 정책 권한에 의해 관리됩니다.

☐ 퍼블릭
이미지들에 대해 공개적으로 표시되고 액세스할 수 있습니다.

리포지토리 이름
간결한 이름을 제공합니다. 개발자는 이름으로 리포지토리 콘텐츠를 식별할 수 있어야 합니다.

767406634679.dkr.ecr.ap-northeast-2.amazonaws.com/

최대 256자 중 9자(최소 2자 이상) The name must start with a letter and can only contain lowercase letters, numbers, hyphens, underscores, periods and forward slashes.

태그 변경 불가능 **정보**
동일한 태그를 사용하는 후속 이미지 푸시가 이미지 태그를 덮어쓰지 않도록 방지하려면 [태그 변경 불가능]을 활성화합니다. 이미지 태그를 덮어쓰려면 (태그 변경 불가능)을 비활성화합니다.

☒ 비활성화됨

이 리포지토리가 생성되면 해당 리포지토리의 가시성 설정을 변경할 수 없습니다.

푸쉬 명령어들 확인

aws에서 잘 알려주기 때문에 순서를 따라서 잘 push하면 된다.

chancloud 리포지토리로 설정 완료

푸시 명령 보기

chancloud에 대한 푸시 명령

macOS / Linux Windows

최신 버전의 AWS CLI 및 Docker가 설치되어 있는지 확인합니다. 자세한 내용은 [Amazon ECR 시작하기](#)를 참조하세요.

다음 단계를 사용하여 이미지를 인증하고 리포지토리에 푸시합니다. Amazon ECR 자격 증명 helpers를 비롯한 추가 레지스트리 인증 방법은 [레지스트리 인증](#)을 참조하십시오.

- 인증 토큰을 검색하고 레지스트리에 대해 Docker 클라이언트를 인증합니다.
AWS CLI 사용:

```
aws ecr get-login-password --region ap-northeast-2 | docker login --username AWS --password-stdin 767406634679.dkr.ecr.ap-northeast-2.amazonaws.com
```


참고: AWS CLI(를) 사용하는 등 오류가 발생하면 최신 버전의 AWS CLI 및 Docker가 설치되어 있는지 확인하세요.
- 다음 명령을 사용하여 도커 이미지를 빌드합니다. 도커 파일을 처음부터 새로 빌드하는 방법에 대한 자세한 내용은 [여기](#) 지침을 참조하십시오. 이미지를 이미 빌드한 경우에는 이 단계를 건너뛸 수 있습니다.

```
docker build -t chancloud.
```
- 빌드가 완료되면 이미지에 태그를 지정하여 이 리포지토리에 푸시할 수 있습니다.

```
docker tag chancloud:latest 767406634679.dkr.ecr.ap-northeast-2.amazonaws.com/chancloud:latest
```
- 다음 명령을 실행하여 이 이미지를 새로 생성한 AWS 리포지토리로 푸시합니다.

```
docker push 767406634679.dkr.ecr.ap-northeast-2.amazonaws.com/chancloud:latest
```

닫기

ECR 권한 추가

권한 추가 [정보](#)

권한 정책 (선택됨 1/847) [정보](#) [정책 생성](#)

새 역할에 연결할 정책을 하나 이상 선택합니다.

Q 속성 또는 정책 이름을 기준으로 정책을 필터링하고 Enter를 누릅니다. 35 개 일치

"admin" X [필터 지우기](#)

	정책 이름	유형	설명
<input type="checkbox"/>	AWSSSODirectoryAdministrator	AWS 관...	Administrator access for SSO Directory
<input type="checkbox"/>	CloudWatchAgentAdminPolicy	AWS 관...	Full permissions required to use AmazonCloudWatchAgent.
<input type="checkbox"/>	DatabaseAdministrator	AWS 관...	Grants full access permissions to AWS services and actions required to set up and configure AWS d...
<input type="checkbox"/>	AWSSSOMasterAccountAdministrator	AWS 관...	Provides access within AWS SSO to manage AWS Organizations master and member accounts and...
<input type="checkbox"/>	AWSCloud9Administrator	AWS 관...	Provides administrator access to AWS Cloud9.
<input type="checkbox"/>	AWSSSOMemberAccountAdministrator	AWS 관...	Provides access within AWS SSO to manage AWS Organizations member accounts and cloud appli...
<input type="checkbox"/>	SystemAdministrator	AWS 관...	Grants full access permissions necessary for resources required for application and development op...
<input checked="" type="checkbox"/>	AdministratorAccess	AWS 관...	Provides full access to AWS services and resources.

실습에서는 편의상 Administrator에 대한 권한 설정을 부여한 것으로 실제 ecr push에 대한 권한을 주기위해선 ecr push 정책에대한것만 부여한다. (일반사용자에게 push 권한등만 주면되는데 관리자급의 권한을 주면 너무 과하고 보안상 좋지않다.)

ECR에 올리기위해 만들어진 정책에 해당하는 역할 부여

인스턴스 (1/1) [정보](#) [연결](#) [인스턴스 상태](#) [작업](#) [인스턴스 시작](#)

Q 인스턴스 속성 또는 (case-sensitive) 태그 찾기

	Name	인스턴스 ID	인스턴스 상태	인스턴스 유형	상태 검사	경보 상태	가...
<input checked="" type="checkbox"/>	testchan	i-0e34815fb72280ced	실행 중	t2.micro	2/2개 검사 통과...	경보 없음	ap...

- 연결
- 세부 정보 보기
- 인스턴스 상태 관리
- 인스턴스 설정
- 네트워킹
- 보안 그룹 변경
- 보안
- 이미지 및 템플릿
- 모니터링 및 문제 해결
- Windows 알로 가져오기
- IAM 역할 수정

EC2 > 인스턴스 > i-0e34815fb72280ced > IAM 역할 수정

IAM 역할 수정 [정보](#)

IAM 역할을 인스턴스에 연결합니다.

인스턴스 ID

[i-0e34815fb72280ced](#) (testchan)

IAM 역할

인스턴스에 연결할 IAM 역할을 선택하거나 역할이 생성되어 있지 않다면 새 역할을 생성합니다. 선택한 역할이 현재 인스턴스에 연결된 모든 역할을 대체합니다.

[EC2_access_for_test](#) [새 IAM 역할 생성](#)

[취소](#) [IAM 역할 업데이트](#)

IAM 역할이 부여된 걸 확인

인스턴스: i-Oe34815fb72280ced(testchan)

세부 정보

보안

네트워킹

스토리지

상태 검사

모니터링

태그

▼ 인스턴스 요약 정보

인스턴스 ID

i-Oe34815fb72280ced (testchan)

IPv6 주소

~

호스트 이름 유틸

IP 이름: ip-172-31-33-37.ap-northeast-2.compute.internal

프라이빗 리소스 DNS 이름 응답

IPv4(A)

자동 할당된 IP 주소

13.209.41.76 (파블릭 IP)

IAM 역할

EC2_access_for_test

퍼블릭 IPv4 주소

13.209.41.76 | [개방 주소법](#)

인스턴스 상태

실행 중

프라이빗 IP DNS 이름 (IPv4만 해당)

ip-172-31-33-37.ap-northeast-2.compute.internal

인스턴스 유틸

t2.micro

VPC ID

vpc-025ae1c0d14cdd5cd

서브넷 ID

subnet-090c51aa4fd4d801

프라이빗 IPv4 주소

172.31.33.37

퍼블릭 IPv4 DNS

ec2-13-209-41-76.ap-northeast-2.compute.amazonaws.com | [개방 주소법](#)

탄력적 IP 주소

~

AWS Compute Optimizer 찾기

환경 사항을 위해 AWS Compute Optimizer에 솔루션을 자세히 알아보기

Auto Scaling 그룹 이름

~

```
[root@ip-172-31-33-37 docker]# aws ecr get-login-password --region ap-northeast-2 | docker login --username AWS --password-stdin 767406634679.dkr.ecr.ap-northeast-2.amazonaws.com
WARNING! Your password will be stored unencrypted in /root/.docker/config.json.
Configure a credential helper to remove this warning. See
https://docs.docker.com/engine/reference/commandline/login/#credentials-store

Login Succeeded
[root@ip-172-31-33-37 docker]#
```

안내과정 3)에 해당하는 이미지에 태그를 붙여 리포지토리에 푸쉬할 수 있게 생성

```
[root@ip-172-31-33-37 docker]# docker tag chanwebserver:latest 767406634679.dkr.ecr.ap-northeast-2.amazonaws.com/chancloud:latest
[root@ip-172-31-33-37 docker]#
[root@ip-172-31-33-37 docker]# docker images
REPOSITORY                                TAG                IMAGE ID            CREATED             SIZE
767406634679.dkr.ecr.ap-northeast-2.amazonaws.com/chancloud    latest             be58a729f15a       42 minutes ago     360MB
chanwebserver                                                    latest             be58a729f15a       42 minutes ago     360MB
tgagor/centos-stream                                             latest             36c2bda5b9ae       6 days ago         220MB
centos                                                            7                 eeb6ee3f44bd       19 months ago      204MB
centos                                                            centos7            eeb6ee3f44bd       19 months ago      204MB
centos/systemd                                                   latest             05d3c1e2d0c1       4 years ago         202MB
[root@ip-172-31-33-37 docker]#
```

안내과정 4)에 해당하는 ECR에 내가만든 이미지 올리기

```
[root@ip-172-31-33-37 docker]# docker push 767406634679.dkr.ecr.ap-northeast-2.amazonaws.com/chancloud:latest
The push refers to repository [767406634679.dkr.ecr.ap-northeast-2.amazonaws.com/chancloud]
0e5f844d747f: Pushed
56c7304e30ab: Pushed
fe9ae4acca4e: Pushed
latest: digest: sha256:a25c594eab1933d5d66f986ba404662a7a9c94ec87d540f3e7a2a9411600a475 size: 948
[root@ip-172-31-33-37 docker]#
```

ECR에 올라간 이미지파일 확인

chancloud

푸시 명령 보기

편집

이미지 (1)

🔄

삭제

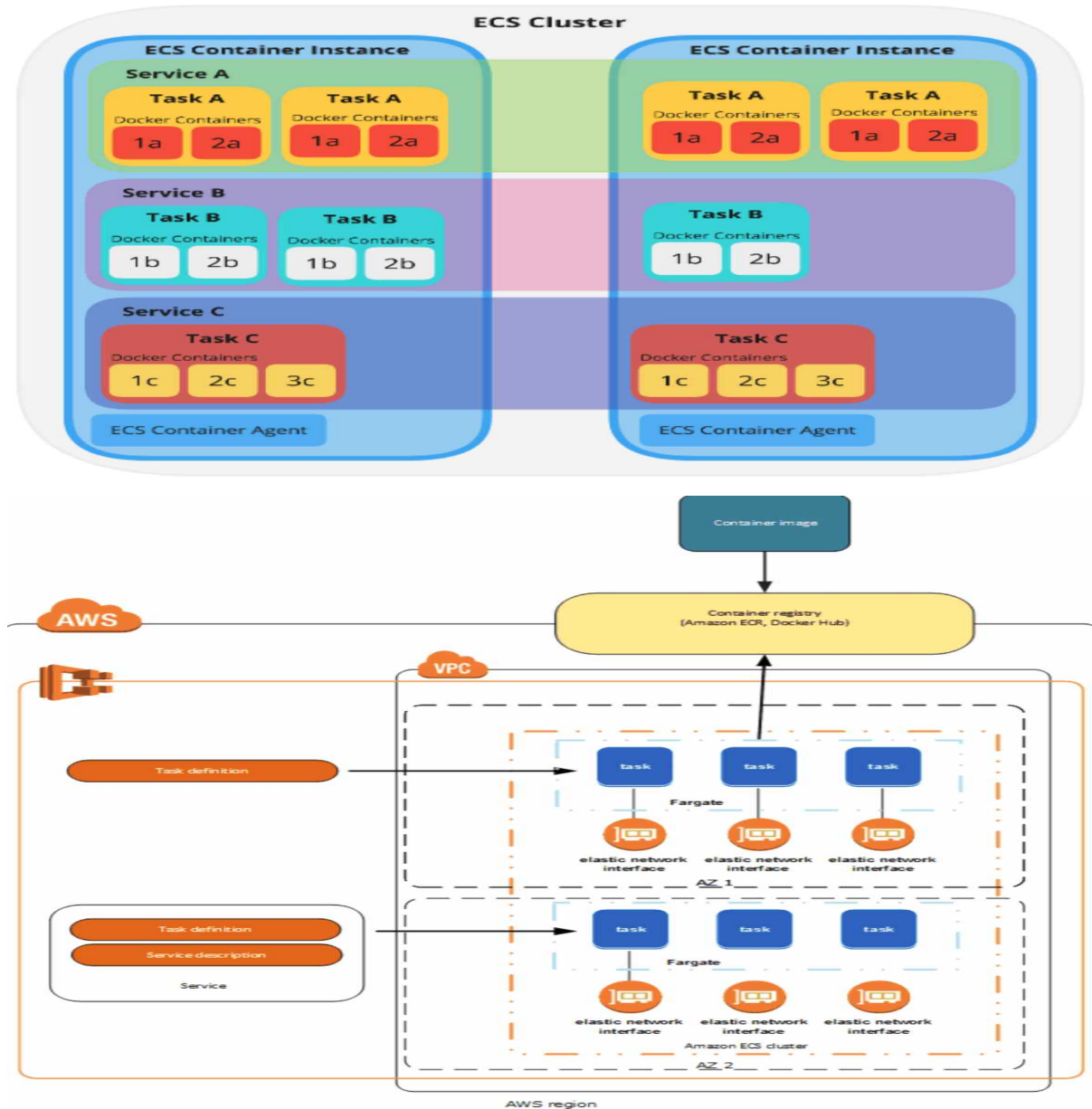
세부 정보

스캔

<input type="checkbox"/>	이미지 태그	아티팩트 유형	푸시 위치	크기 (MB)	이미지 URI	다이제스트	스캔 상태	위약성
<input type="checkbox"/>	latest	Image	2025년 4월 30일, 15:51:15 (UTC+09)	180.01	URI 복사	sha256:a25c594eab1933...	-	-

Amazon Elastic Container Service

AWS에서 제공하는 완전관리형 컨테이너 오케스트레이션 서비스로 EKS는 직접 관리해야하는 부분이 많고 비싸서 ECS사용도 상황에 맞게 사용하면 좋다.



ECS 구성 요소

- **Cluster**: 관리할 컨테이너 리소스들의 논리적인 그룹 단위
- **Task Definition**: 사용할 컨테이너들에 대한 작업 정의, 컨테이너 이미지/CPU/Memory/네트워킹모드/로깅구성/호스팅인프라 등을 정의해준다.
- **Task**: Task Definition을 인스턴스화 한 개념(K8S의 pod와 비슷한 개념)
- **Service**: Cluster내의 지정된 수의 Task를 실행하고 관리하는 개념, 원하는 Task의 수를 유지하고 **선택적으로 LB와 연결**(K8S의 Deployment처럼 원하는 만큼 배포하는기능 존재)
- **Container Agent**: Cluster의 각 호스팅 인스턴스마다 실행, 리소스들에 대한 정보를 ECS에 전송하고, ECS의 요청을 받아서 수행(K8S의 kubelet와 비슷한 개념)

ESC > 클러스터 > 클러스터 생성

어려울 것 없이 기본으로 제공되는 vpc를 사용하면된다.

클러스터 생성 정보

Amazon ECS 클러스터는 태스크와 서비스를 함께 그룹화하고 용량과 공통적인 구성을 공유합니다. 모든 태스크, 서비스 및 용량은 클러스터에 속해야 합니다.

클러스터 구성

클러스터 이름

cicd-chan-cluster

최대 255자까지 가능합니다. 유효한 문자는 문자(대문자 및 소문자), 숫자, 하이픈 및 밑줄입니다.

▼ 네트워킹 정보

기본적으로 태스크 및 서비스는 기본 VPC의 기본 서브넷에서 실행됩니다. 기본값이 아닌 VPC를 사용하려면 VPC와 서브넷을 지정하세요.

VPC

퍼블릭 및 프라이빗 서브넷이 있는 VPC를 사용합니다. 기본적으로 VPC는 AWS 계정에 대해 생성됩니다. 새 VPC를 생성하려면 [VPC 콘솔](#) 링크를 클릭하세요.

vpc-025ae1e0d14cdd5cd

기본값

서브넷

태스크가 실행되는 서브넷을 선택합니다. 프로젝트에는 서브넷 3개를 사용하는 것이 좋습니다.

서브넷 선택

subnet-0faf5c2fc470f5d14 X
ap-northeast-2d

subnet-0fe36fbcfc9660e82 X
ap-northeast-2a

subnet-0c9d50d4950e3b9b3 X
ap-northeast-2b

subnet-090e51aa4fcd4d801 X
ap-northeast-2c

기본 네임스페이스 - 선택 사항

네임스페이스를 선택하여 애플리케이션을 구성하는 서비스 그룹을 지정합니다. 서비스 수준에서 이 값을 덮어쓸 수 있습니다.

Q cicd-chan-cluster

task 생성

Amazon Elastic Container Service > 태스크 정의

태스크 정의 (0) 정보

속성 또는 값으로 태스크 정의 필터링

No matches

< 1 >

마지막 개정 상태 = ACTIVE

필터 지우기

태스크 정의

마지막 개정 상태

태스크 정의 없음

표시할 태스크 정의가 없습니다.

새 태스크 정의 생성

레포지토리 URI 복사

Private Public

프라이빗 레포지토리 (1)

리포지토리 찾기

< 1 >

<input type="checkbox"/>	리포지토리 이름	URI	생성 날짜	태그 변경 불가능	스캔 빈도	암호화 유형	플스루 캐시
<input type="checkbox"/>	chancloud	767406634679.dkr.ecr.ap-northeast-2.amazonaws.com/chancloud	2023년 4월 30일, 15:11:27 (UTC+09)	비활성화됨	수동	AES-256	비활성

task 이름과 컨테이너이름, ECR URI 지정

태스크 정의 구성

태스크 정의 패밀리 정보

고유한 태스크 정의 패밀리 이름을 지정합니다.

cicd-chan-task

최대 255개의 문자(대문자 및 소문자, 숫자, 하이픈 및 밑줄이 허용됩니다).

컨테이너 - 1 정보

필수 컨테이너

제거

컨테이너 세부 정보

이름과 컨테이너 이미지를 지정하고 컨테이너를 필수로 표시할지 지정합니다. 각 태스크 정의에는 필수 컨테이너가 하나 이상 있어야 합니다.

이름

web1-chan

이미지 URI

767406634679.dkr.ecr.ap-northeast-2.amazonaws.com

필수 컨테이너

예

Private registry 정보

Store credentials in Secrets Manager, and then use the credentials to reference images in private registries.

☒ Private registry authentication

포트 매핑 정보

포트 매핑을 추가하여 컨테이너가 호스트의 포트에 액세스하여 트래픽을 송수신하도록 허용합니다. 포트 매핑 구성을 변경하면 연결된 서비스 연결 설정에 영향을 미칩니다.

컨테이너 포트

80

프로토콜

TCP

포트 이름

web1-chan-80-tcp

앱 프로토콜

HTTP

제거

포트 매핑 더 추가

컨테이너 최소 사양으로 설정

간단하게 실습으로 사용하는것이니 그리 비쌀 필요가 없다.

▼ 환경

태스크 정의에 대한 인프라 요구 사항을 지정합니다.

앱 환경 정보

태스크 정의에 대한 인프라를 지정합니다.

옵션 추가 ▼

AWS Fargate(서버리스) X

운영 체제/아키텍처 정보

Linux/X86_64 ▼

태스크 크기 정보

태스크를 위해 예약할 CPU 및 메모리의 양을 지정합니다.

CPU

.25 vCPU ▼

메모리

.5 GB ▼

태스크 정의 생성 완료

cicd-chan-task:1(7) 성공적으로 생성되었습니다. 이 태스크 정의를 사용하여 서비스를 배포하거나 태스크를 실행할 수 있습니다.

백도 ▼

Amazon Elastic Container Service > 태스크 정의 > cicd-chan-task > 개정 1개 > 컨테이너

cicd-chan-task:1

백도 ▼ 작업 ▼ 새 개정 생성 ▼

개요 정보

ARN arn:aws:ecs:ap-northeast-2:767406634679:task-definition/cicd-chan-task:1	상태 ACTIVE	생성된 시간 2023. 4. 30. 8시 17분 6초 UTC	앱 환경 FARGATE
태스크 역할 -	태스크 실행 역할 ecsTaskExecutionRole	운영 체제/아키텍처 Linux/X86_64	네트워크 모드 awsvpc

사진에는 없지만 생성과정에서 보안선택부분이 있다면 80포트로 접속 할 수 있도록

“보안그룹을 생성“ 해서 80포트 인바운드 규칙 설정

”기존 보안그룹 선택“해서 80포트 인바운드 규칙이 설정되어 있는 보안그룹을 사용하면된다.

서비스 생성

유형은 선택

패밀리 선택

서비스 이름 생성

태스크는 service가 task의 갯수를 조정하는걸 보려고 2로 바꾼것으로 1으로 설정해도 무방

배포 구성

애플리케이션 유형 [정보](#)

실행할 애플리케이션 유형을 지정합니다.



서비스

중지 및 다시 시작할 수 있는 장기 실행
런타임 작업(예: 웹 애플리케이션)을 처리
하는 태스크 그룹을 시작합니다.



태스크

실행하고 종료하는 독립 실행형 태스크
(예: 배치 작업)를 시작합니다.

태스크 정의

기존 태스크 정의를 선택합니다. 새 태스크 정의를 생성하려면 [태스크 정의](#) 페이지로 이동합니다.

☐ 수동으로 개정 지정하기

선택한 작업 정의 패밀리의 가장 최근 개정 100개 중에서 선택하는 대신 수동으로 개정을 입력합니다.

패밀리

cicd-chan-task

개정

1 (최신)

서비스 이름

이 서비스에 대한 고유한 이름을 지정합니다.

cicdd-chan-service

서비스 유형 [정보](#)

서비스 스케줄러가 따를 서비스 유형을 지정합니다.



복제본

클러스터 전체에 원하는 작업 수를 배치
하고 유지 관리합니다.



대몬(Daemon)

각 컨테이너 인스턴스에 작업 사본 하나
를 배치하고 유지 관리합니다.

원하는 태스크

시작할 태스크 수를 지정합니다.

2

▶ 배포 옵션

▶ 배포 실패 감지 [정보](#)

네트워크는 기본 vpn으로

보안그룹- 권한을 웹페이지를 요청하면 볼수있도록 http 즉 80포트를 열어줘야한다.

(이런 하나하나 선택을 안해서 연결이 되었는데 안된것처럼 착각을 할 수있어서 하나 하나 잘 보면서 설정해주는게 좋다. - 이 문서를 만들때 보안그룹을 설정해주지않아 애먼곳에서 찾다가 2시간정도 날렸다 그만큼 설정 하나하나 잘보고 이해하며 선택하는게 좋다.)

▼ 네트워킹

VPC 정보

사용할 가상 사설 클라우드를 선택합니다.

vpc-025ae1e0d14cdd5cd
기본값

서브넷

태스크 스케줄러가 배치 시 고려해야 하는 VPC 내 서브넷을 선택합니다.

서브넷 선택

subnet-0faf5c2fc470f5d14 X
ap-northeast-2d

subnet-0fe36fbfc9660e82 X
ap-northeast-2a

subnet-0c9d50d4950e3b9b3 X
ap-northeast-2b

subnet-090e51aa4fcd4d801 X
ap-northeast-2c

보안 그룹 정보

기존 보안 그룹을 선택하거나 새 보안 그룹을 생성합니다.

☒ 기존 보안 그룹 선택

☐ 새 보안 그룹 생성

보안 그룹 이름

기존 보안 그룹을 선택합니다.

sg-065196a4477042667 X
chan-GW

sg-0ba045ab79010d020 X
default

퍼블릭 IP 정보

태스크의 탄력적 네트워크 인터페이스(ENI)에 퍼블릭 IP를 자동 지정할지 여부를 선택합니다.

☒ 켜짐

LB 생성은 ALB로 구성

(EC2 인스턴스에서 ALB를 만들고 타겟 그룹을 생성해서 여기서 "기존 로드 밸런서 사용", "기존 대상 그룹 사용"으로 설정해서 사용해도 무방하다.)

로드 밸런서 유형 **정보**

서비스에서 실행 중인 태스크에서 수신 트래픽을 분산하도록 로드 밸런서를 구성합니다.

Application Load Balancer

Application Load Balancer

새 로드 밸런서를 생성할지, 아니면 기존 로드 밸런서를 선택할지 지정합니다.

☒ 새 로드 밸런서 생성

☐ 기존 로드 밸런서 사용

로드 밸런서 이름

로드 밸런서에 대한 고유한 이름을 지정합니다.

ALB-chan

로드 밸런싱할 컨테이너 선택

web1-chan 80:80

리스너 **정보**

로드 밸런서가 연결 요청을 수신 대기할 포트 및 프로토콜을 지정합니다.

☒ 새 리스너 생성

☐ 기존 리스너 사용

기존 로드 밸런서를 선택해야 합니다.

포트

80

프로토콜

HTTP

대상 그룹 **정보**

새 대상 그룹을 생성할지, 아니면 로드 밸런서가 요청을 서비스의 태스크에 라우팅하는 데 사용할 기존 대상 그룹을 선택할지 지정합니다.

☒ 새 대상 그룹 생성

☐ 기존 대상 그룹 사용

기존 로드 밸런서를 선택해야 합니다.

대상 그룹 이름

ALB-chan-tg

프로토콜

HTTP

상태 확인 경로 **정보**

/

상태 확인 프로토콜

HTTP

상태 검사 유예 기간 **정보**

초

생성된 클러스터에 해당하는 서비스 확인 하고 동작 확인

cicdd-chan-service **정보**

↻

서비스 업데이트

서비스 삭제

상태 및 지표

로그

구성 및 태스크

배포 및 이벤트

네트워킹

태그

상태 **정보**

ARN	상태	태스크	배포 현재 상태
cicdd-chan-cluster/cicdd-chan-service	🟢 활성	<div><div></div>0개 대기 중, 2개 실행 중 / 2개 필요</div>	🟢 2개 완료

▼ 로드 밸런서 상태

로드 밸런서 이름	대상 합계	정상 대상	비정상 대상
ALB-chan	2	2	0

로드밸런서로 들어가 DNS이름으로 웹요청 시도

EC2 > 로드 밸런서

로드 밸런서 (1/1)
Elastic Load Balancing은 수신 트래픽의 변화에 따라 자동으로 로드 밸런서 용량을 확장합니다.

🔄

작업 ▼

로드 밸런서 생성

🔍 속성 또는 값을 기준으로 필터링

< 1 >

<input checked="" type="checkbox"/>	이름 ▼	DNS 이름 ▼	상태 ▼
<input checked="" type="checkbox"/>	ALB-chan	ALB-chan-1905723908.ap-northeast-2.elb.amazonaws.com	🟢 Active

Load balancer: ALB-chan

세부 정보 | 리스너 | 네트워크 매핑 | 보안 | Monitoring | 통합 | 속성 | 태그

세부 정보

📄 `arn:aws:elasticloadbalancing:ap-northeast-2:767406634679:loadbalancer/app/ALB-chan/f3b2b865be7364fe`

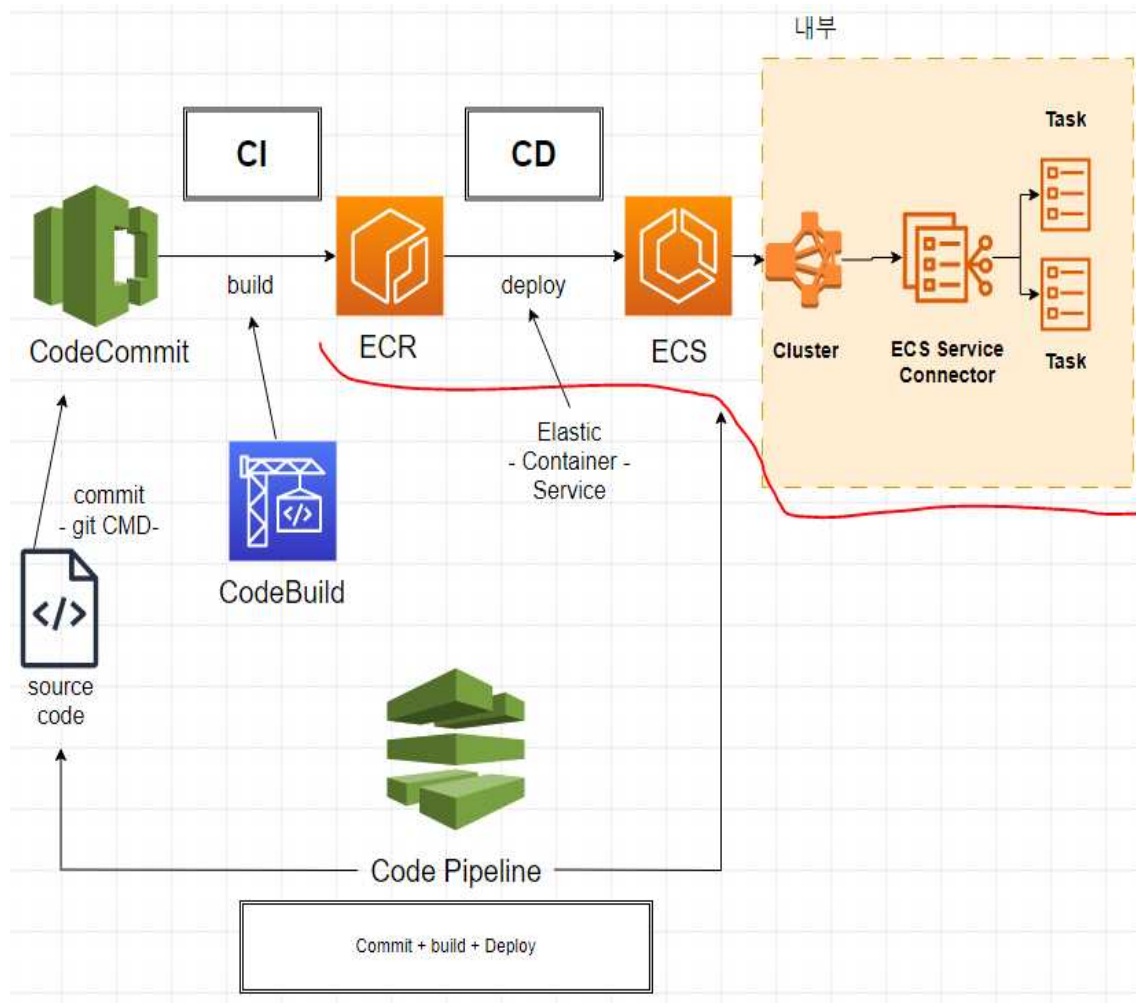
로드 밸런서 유형 Application	DNS 이름 ALB-chan-1905723908.ap-northeast-2.elb.amazonaws.com (A 레코드)	상태 🟢 Active	VPC vpc-025ae1e0d14cdd5cd
--------------------------	---	----------------	--

웹 요청시 화면이 보임

CICD & Docker example ChanHyuck

Version one (V1.0) in Green Color.

현재까지 설정한 상황



위 그림은 전체적인 그림이고 실질적으로 우리가 해놓은 설정은 밑줄 친 부분으로 사용자 운영체제에서 ECR로 이미지파일을 만들어 넣은 후 그것에 해당하는 이미지를 ECS를 통해 컨테이너를 만든 상황이다. 이 후는 이제 pipeline 구축 즉 Commit + build + Deploy 작업을 자동으로 하도록 연동해 주어야 한다.

code commit

repository 생성

repository 이름 code를 올릴 저장소로 원하는 이름으로 생성

개발자 도구 > CodeCommit > 리포지토리 > 리포지토리 생성

리포지토리 생성

코드를 저장하고 공유할 안전한 리포지토리를 만듭니다. 먼저 리포지토리 이름과 리포지토리 설명을 입력합니다. 리포지토리 이름은 해당 리포지토리의 URL에 포함됩니다.

리포지토리 설정

리포지토리 이름

최대 100자입니다. 기타 제한이 적용됩니다.

설명 - 선택 사항

최대 1,000자입니다.

태그

추가

[취소](#) [생성](#)

레포지토리 우측상단의 "push 명령 보기"를 통해 절차대로 진행
(이 사진은 임시로 찍은것으로 실습에선 쓴 레포지토리와 다름)

▼ 연결 단계

[HTTPS](#) | [SSH](#) | [HTTPS\(GRC\)](#)

1단계: 사전 요구 사항

Git 버전 1.7.9 이상을 지원하는 Git 클라이언트를 사용하여 AWS CodeCommit 리포지토리에 연결해야 합니다. Git 클라이언트가 없으면 Git 다운로드에서 설치할 수 있습니다. [Git 다운로드 페이지 보기](#)

IAM 사용자에게 연결된 AWS CodeCommit 관리 정책, CodeStar 프로젝트 팀에 속한 권한 또는 동등한 권한이 있어야 합니다. [AWS CodeCommit에 액세스하기 위한 IAM 사용자를 만들고 구성하는 방법을 익히십시오.](#) | [팀원을 AWS CodeStar 프로젝트에 추가하는 방법에 대해 알아보십시오.](#)

2단계: Git 자격 증명

IAM 사용자를 위한 Git 자격 증명을 아직 만들지 않은 경우 만드십시오. 자격 증명을 다운로드하여 안전한 위치에 저장합니다. [Git 자격 증명 생성](#)

3단계: 리포지토리 복제

리포지토리를 로컬 컴퓨터에 복제하고 코드 작업을 시작합니다. 다음 명령을 실행합니다.

[복사](#)

추가 세부 정보

자세한 지침은 설명서에서 찾을 수 있습니다. [설명서 보기](#)

[참고] https, ssh 총 두 단계가 있으며 문서대로 진행할 생각이라면 https 단계는 건너뛰고 ssh 단계를 따라하는 걸 추천한다.

(공용) EC2 인스턴스에서 git 다운로드

```
[root@ip-172-31-33-37 docker]# yum install git
Last metadata expiration check: 1 day, 0:49:54 ago on Sun Apr 30 04:25:48 2023.
Dependencies resolved.

=====
Package                               Size                               Architecture
Repository                             Size
=====
Installing:
git                                     68 k                               x86_64
Installing dependencies:
git-core                               4.2 M                               x86_64
```

방법1) HTTPS를 통한 연결

iam 자격증명

2단계: Git 자격 증명

IAM 사용자를 위한 Git 자격 증명을 아직 만들지 않은 경우 만드십시오. 자격 증명을 다운로드하여 안전한 위치에 저장합니다. [Git 자격 증명 생성](#)

AWS CodeCommit에 대한 HTTPS Git 자격 증명 (0)

AWS CodeCommit 리포지토리에 대한 HTTPS 연결을 인증하는 데 사용할 수 있는 사용자 이름과 암호를 생성합니다. 한 번에 최대 2개의 자격 증명 세트(활성 또는 비활성)를 보유할 수 있습니다. [자세히 알아보기](#)

작업 ▼

자격 증명 생성

사용자 이름

생성됨

상태

자격 증명 없음

자격 증명 생성

3단계: 리포지토리 복제

리포지토리를 로컬 컴퓨터에 복제하고 코드 작업을 시작합니다. 다음 명령을 실행합니다.

```
git clone https://git-codecommit.ap-northeast-2.amazonaws.com/v1/repos/docker-chan-repo
```

✓ 복사됨

복사

자격증명으로 생성된 아이디와 비밀번호를 복사해서 입력

자격 증명 생성



✔ 새 자격 증명을 사용할 수 있습니다.

사용자 이름과 암호를 저장하거나 자격 증명 파일을 다운로드합니다.

지금 아니면 암호를 확인하거나 다운로드할 수 없습니다. 나중에 복구할 수 없습니다. 하지만 언제든지 암호를 재설정할 수 있습니다.

로컬 컴퓨터나 정적 사용자 이름과 암호가 필요한 도구에서 연결할 때 이러한 자격 증명을 사용할 수 있습니다. [자세히 알아보기](#)

사용자 이름

aws.user01-at-767406634679

암호

Ntt7u2WVu4sL8le0PEG+1u/V+gVPaOxliZ71+j9L4Tk= [숨기기](#)

자격 증명 다운로드

닫기

```
[root@ip-172-31-33-37 ~]$ docker run --rm --network host --env-file docker-creds git
[root@ip-172-31-33-37 docker]# git clone https://git-codecommit.ap-northeast-2.amazonaws.com/v1/repos/docker-chan-repo
Cloning into 'docker-chan-repo'...
Username for 'https://git-codecommit.ap-northeast-2.amazonaws.com': aws.user01-at-767406634679
Password for 'https://aws.user01-at-767406634679@git-codecommit.ap-northeast-2.amazonaws.com':
warning: You appear to have cloned an empty repository.
[root@ip-172-31-33-37 docker]#
```

방법2) SSH를 통한 연결



AWS에서 제공하는 절차

단계별로 순서대로 따라하면 된다.

다운로드에서 설치할 수 있습니다. [Git 다운로드 페이지 보기](#)

IAM 사용자에게 연결된 AWS CodeCommit 관리 정책, CodeStar 프로젝트 팀에 속한 권한 또는 동등한 권한이 있어야 합니다. [AWS CodeCommit에 액세스하기 위한 IAM 사용자를 만들고 구성하는 방법을 익히십시오.](#) | [팀원을 AWS CodeStar 프로젝트에 추가하는 데 대해 알아보십시오.](#)

SSH 퍼블릭-프라이빗 키 쌍이 있어야 합니다. Bash 에뮬레이터를 열고 ssh-keygen을 사용하여 퍼블릭-프라이빗 키 쌍을 만듭니다. [퍼블릭 키 쌍 생성 방법 알아보기](#)

2단계: SSH 퍼블릭 키 등록

SSH 퍼블릭 키를 IAM 사용자에게 업로드합니다. [SSH 퍼블릭 키 업로드 방법 알아보기](#)

SSH 퍼블릭 키를 업로드한 후 SSH 키 ID를 복사합니다. 다음 단계에서 이 정보가 필요합니다.

3단계: 로컬 SSH 구성 편집

로컬 ~/.ssh 디렉터리에서 "config"라는 SSH 구성 파일을 편집합니다. 파일에 다음 줄을 추가하십시오. 여기서 User의 값은 2단계에서 한 SSH 키 ID입니다.

```
Host git-codecommit.*.amazonaws.com
User Your-IAM-SSH-Key-ID-Here
IdentityFile ~/.ssh/Your-Private-Key-File-Name-Here
```

파일을 저장했으면 ~/.ssh 디렉터리에서 다음 명령을 실행하여 올바른 권한이 있는지 확인하십시오.

```
chmod 600 config
```

4단계: 리포지토리 복제

리포지토리를 로컬 컴퓨터에 복제하고 코드 작업을 시작합니다. 다음 명령을 실행합니다.

```
git clone ssh://git-codecommit.ap-northeast-2.amazonaws.com/v1/repos/docker-chan-repo
```

복사

시험용 인스턴스에서 keygen 확인

```
[root@ip-172-31-33-37 docker]# ssh-keygen
Generating public/private rsa key pair.
Enter file in which to save the key (/root/.ssh/id_rsa):
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /root/.ssh/id_rsa
Your public key has been saved in /root/.ssh/id_rsa.pub
The key fingerprint is:
SHA256:KI6nNaUgK1KzprghuPvnKh0tXCTb+VNqoBqLIA8sYfM root@ip-172-31-33-37
The key's randomart image is:
+---[RSA 3072]-----+
|
|  . .
|  = .
| .o. = ..
|ooo+.ooooS
|B.BE.+==
|OO.=*..
|@o==..
|XX=+.
+----[SHA256]-----+
[root@ip-172-31-33-37 docker]# cd ~/.ssh/
[root@ip-172-31-33-37 .ssh]# ll
total 12
-rw-----. 1 root root 557 Apr 30 04:25 authorized_keys
-rw-----. 1 root root 2655 May 1 05:31 id_rsa
-rw-r--r--. 1 root root 606 May 1 05:31 id_rsa.pub
[root@ip-172-31-33-37 .ssh]#
```

퍼블릭 키 복사

```
[root@ip-172-31-33-37 .ssh]#
[root@ip-172-31-33-37 .ssh]# cat id_rsa.pub
ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAQGCxBOpNcE0+rHO1qePZwg/LZUV70IrAJOX1/rL1S74
we5tBqkIO42WRemL5s0rdnSre6wCTZcbDVdkIF50X7eRUznKTdbIE/K8ijtEnXZZIuttIZEaVJn2W3j
12i5Zts6KaMqs+g99tgAARfOcD9n+GKWnwACTvQ4GYv39uXQgP9oja/oBOxwR2RCpfssP/yd3KQusCS
zyGhzXJDcrDp+TdQ2pMLWtvfz1sBwBJila5dvcBvGTCFOqSm6pxnG0sHiQ1q77dx5tDx5XjShwBiCbS
theast-2.compute.internal
[root@ip-172-31-33-37 .ssh]#
```

2)단계인 public key 업로드

SSH 퍼블릭 키 업로드

SSH 퍼블릭 키의 내용을 다음 필드에 붙여 넣습니다.

ssh-rsa
AAAAB3NzaC1yc2EAAAADAQABAAQGCxBOpNcE0+rHO1qePZwg/LZUV70IrAJOX1/
rL1S74r+OT7jWiRuE05pN2vV9/F1C+5NI1NPahLwjY2/6uKChx0vyFkhKpXwKkDVN63
bY5Stwe5tBqkIO42WRemL5s0rdnSre6wCTZcbDVdkIF50X7eRUznKTdbIE/K8ijtEnXZZI
uttIZEaVJn2W3jozgcWrAKXYReso4wJ9INmN6db26u6cOBjkP8AS+KvWNBga+rJ9VRD
lzhWjRbh9rDmp12i5Zts6KaMqs+g99tgAARfOcD9n+GKWnwACTvQ4GYv39uXQgP9oj
a/oBOxwR2RCpfssP/yd3KQusCSsn/OGxSU4DxHPvG6lZM3dNkl6ttsz2u6K8Vf5rLLpd7
spPVfExCikoMZ1iRgFX3axbzyGhzXJDcrDp+TdQ2pMLWtvfz1sBwBJila5dvcBvGTCFOqS
m6pxnG0sHiQ1q77dx5tDx5XjShwBiCbSew2E7jl9qMad2/gGIMnH2zaegVO8YDVK2Y
OaU= root@ip-172-31-33-37.ap-northeast-2.compute.internal

취소

SSH 퍼블릭 키 업로드

SSH 구성 파일 편집

vi config

```
Host git-codecommit.*amazonaws.com
User <엑세스 키 이름>
IdentityFile <./.ssh에 들어있는 키 = id_rsa>
```

엑세스 키 (1)

엑세스 키를 사용하여 AWS CLI, AWS Tools for PowerShell, AWS SDK 또는 직접 AWS API 호출을 통해 AWS에 프로그래밍 방식 호출을 전송합니다. 한 번에 최대 두 개의 엑세스 키(활성 또는 비활성)를 가질 수 있습니다. 자세히 알아보기

엑세스 키 만들기

AKIA3FLH7623WGZ6GA6D

설명
myec2-s3-accesskey

마지막 사용
6일 전

마지막으로 사용한 리전
ap-northeast-2

상태
Active

생성된
7일 전

마지막으로 사용한 서비스
s3

user이름은 IAM서비스의 SSH에 해당하는 ID를 추가하고 Identifile은 .ssh폴더에 있는 id_rsa를 넣으면 된다.

```
Host git-codecommit.*.amazonaws.com
User APKA3FLH76233RJNCDAD
IdentityFile ~/.ssh/id_rsa
```

설정 파일 권한 주기

chmod 600 config

파일을 저장했으면 ~/.ssh 디렉터리에서 다음 명령을 실행하여 올바른 권한이 있는지 확인하십시오.

chmod 600 config

4단계: 리포지토리 복제

리포지토리를 로컬 컴퓨터에 복제하고 코드 작업을 시작합니다. 다음 명령을 실행합니다.

git clone ssh://git-codecommit.ap-northeast-2.amazonaws.com/v1/repos/docker-chan-repo

복사됨

복사

레포지토리를 로컬 컴퓨터에 복제

```
[root@ip-172-31-33-37 ~]# ls
docker
[root@ip-172-31-33-37 ~]# git clone ssh://git-codecommit.ap-northeast-2.amazonaws.com/v1/repos/docker-chan-repo
Cloning into 'docker-chan-repo'...
The authenticity of host 'git-codecommit.ap-northeast-2.amazonaws.com (52.95.194.97)' can't be established.
RSA key fingerprint is SHA256:eegAPQrWY9YsYo9ZHlK0mxtfXBHZAzd8Eya53Qcwko.
This key is not known by any other names
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added 'git-codecommit.ap-northeast-2.amazonaws.com' (RSA) to the list of known hosts.
warning: You appear to have cloned an empty repository.
[root@ip-172-31-33-37 ~]# ls
docker  docker-chan-repo
[root@ip-172-31-33-37 ~]#
```

Dockerfile과 index.html을 로컬 repository (ex: docker-chan-repo)로 이동

```
[root@ip-172-31-33-37 docker]# ll
total 8
-rw-r--r--. 1 root root 162 Apr 30 05:44 Dockerfile
drwxr-xr-x. 3 root root 18 May 1 05:24 docker-chan-repo
-rw-r--r--. 1 root root 163 Apr 30 06:05 index.html
[root@ip-172-31-33-37 docker]#
[root@ip-172-31-33-37 docker]# mv Dockerfile index.html docker-chan-repo/
[root@ip-172-31-33-37 docker]# ll
total 0
drwxr-xr-x. 3 root root 54 May 1 06:05 docker-chan-repo
[root@ip-172-31-33-37 docker]# cd docker-chan-repo/
[root@ip-172-31-33-37 docker-chan-repo]#
[root@ip-172-31-33-37 docker-chan-repo]# ll
total 8
-rw-r--r--. 1 root root 162 Apr 30 05:44 Dockerfile
-rw-r--r--. 1 root root 163 Apr 30 06:05 index.html
[root@ip-172-31-33-37 docker-chan-repo]#
```

파일들을 확인 후 add 하고 commit 하고 push함

로컬 repository에 있는 Dockerfile과 index.html을 CodeCommit repository로 이동

```
[root@ip-172-31-33-37 docker-chan-repo]# ll
total 8
-rw-r--r--. 1 root root 162 Apr 30 05:44 Dockerfile
-rw-r--r--. 1 root root 163 Apr 30 06:05 index.html
[root@ip-172-31-33-37 docker-chan-repo]#
[root@ip-172-31-33-37 docker-chan-repo]# git add .
[root@ip-172-31-33-37 docker-chan-repo]# git commit -am "my 1st commit"
[master (root-commit) 8e07088] my 1st commit
Committer: root <root@ip-172-31-33-37.ap-northeast-2.compute.internal>
Your name and email address were configured automatically based
on your username and hostname. Please check that they are accurate.
You can suppress this message by setting them explicitly. Run the
following command and follow the instructions in your editor to edit
your configuration file:

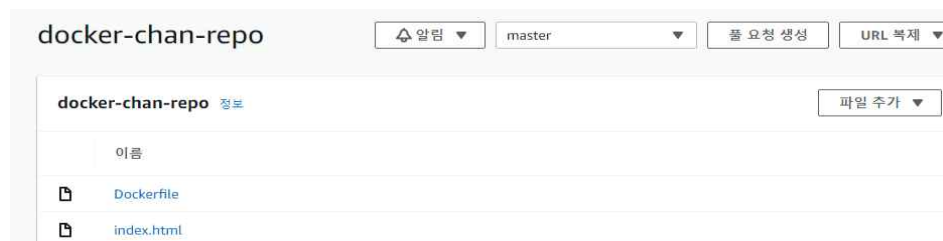
    git config --global --edit

After doing this, you may fix the identity used for this commit with:

    git commit --amend --reset-author

2 files changed, 18 insertions(+)
create mode 100644 Dockerfile
create mode 100644 index.html
[root@ip-172-31-33-37 docker-chan-repo]#
[root@ip-172-31-33-37 docker-chan-repo]# git push
Enumerating objects: 4, done.
Counting objects: 100% (4/4), done.
Compressing objects: 100% (4/4), done.
Writing objects: 100% (4/4), 559 bytes | 559.00 KiB/s, done.
Total 4 (delta 0), reused 0 (delta 0), pack-reused 0
remote: Validating objects: 100%
To ssh://git-codecommit.ap-northeast-2.amazonaws.com/v1/repos/docker-chan-repo
 * [new branch] master -> master
[root@ip-172-31-33-37 docker-chan-repo]#
```

push하자 repository에 올라간 걸 볼 수 있다.



CodeBuild 를 선택하고 프로젝트 실행

CodeBuild > 빌드 프로젝트 생성

개발자 도구 > CodeBuild > 빌드 프로젝트

빌드 프로젝트 정보

  알림 ▼  빌드 시작 ▼ 세부 정보 보기 편집 ▼ 빌드 프로젝트 삭제

빌드 프로젝트 생성

사용자 프로젝트 ▼ < 1 > ⚙

이름 ▼	소스 공급자	리포지토리	최신 빌드 상태	설명	마지막 수정
결과 없음 표시할 결과가 없습니다.					

code build에 사용할 이름 생성

개발자 도구 > CodeBuild > 빌드 프로젝트 > 빌드 프로젝트 생성

빌드 프로젝트 생성

프로젝트 구성

프로젝트 이름

프로젝트 이름은 2~255자여야 합니다. 글자(A-Z 및 a-z), 숫자(0-9) 및 특수 문자(- 및 _)를 포함할 수 있습니다.

설명 - 선택 사항

빌드 배치 - 선택 사항

☐ 빌드 배치 활성화

동시 빌드 제한 활성화 - 선택 사항

이 프로젝트에 허용되는 동시 빌드 수를 제한합니다.

☐ 이 프로젝트가 시작할 수 있는 동시 빌드 수 제한

▶ 추가 구성

태그

(이미지를 잘 선택해야한다 필자의 경우 잘못 선택해서 뒤에 있는 codepipeline 과정에서 version 오류를 설정하는 등 애먼 오류 처리를 진행했다.)

소스 공급자- AWS CodeCommit
레포지토리 - 내가 저장해놓은 repository 선택
브랜치- master
역할이름 - 후에 이 역할에 해당하는 정책을 설정해야 하니 기억해야한다.

아티팩트 및 로그 체크

[참고] 후에 이 아티팩트를 설정하게 되긴 하지만 이 부분은 후에 기술하기 때문에 지금은 넘어간다.

아티팩트

아티팩트 추가

아티팩트 1 - 기본

유형

아티팩트 없음

텍스트를 실행하거나 도커 이미지를 Amazon ECR에 넣는 경우 [아티팩트 없음]을 선택할 수 있습니다.

▶ 추가 구성

캐시, 암호화 키

로그

CloudWatch

☒ CloudWatch 로그 - 선택 사항

이 옵션을 선택하면 빌드 출력 로그가 CloudWatch에 업로드됩니다.

그룹 이름

스트림 이름

S3

☐ S3 로그 - 선택 사항

이 옵션을 선택하면 빌드 출력 로그가 S3에 업로드됩니다.

cloudwatch는 오류 발생시 로그를 확인해가면서 볼 수있어 원하면 그룹과 스트림 이름등을 생성해 로그를 보는 방법도 괜찮다.

아래의 “생성“ 클릭

build start

개발자 도구 > CodeBuild > 빌드 프로젝트 > docker-chan-build

docker-chan-build

알림 ▼ 공유 편집 ▼ 빌드 프로젝트 삭제 재정의로 빌드 시작 빌드 시작

구성

소스 공급자 AWS CodeCommit	기본 리포지토리 docker-chan-repo	아티팩트 업로드 위치 -	빌드 배지 비활성
퍼블릭 빌드 비활성			

에러가 발생 할 수 있다.

codebuild는 기본적으로 yaml형식의 파일인 buildspec을 기반으로 만들기때문에 그것에대한 파일이 없기 때문에 발생한 오류이다. (.yaml OR .yml)

❌ 실패함

시작 시간: 2분 전

현재 단계: COMPLETED

```
1 [Container] 2023/05/01 06:36:36 Waiting for agent ping
2 [Container] 2023/05/01 06:36:37 Waiting for DOWNLOAD_SOURCE
3 [Container] 2023/05/01 06:36:45 Phase is DOWNLOAD_SOURCE
4 [Container] 2023/05/01 06:36:45 CODEBUILD_SRC_DIR=/codebuild/output/src987863467/src/git-codecommit.ap-northeast-2.amazonaws.com/v1/repos/docker-chan-repo
5 [Container] 2023/05/01 06:36:48 Phase complete: DOWNLOAD_SOURCE State: FAILED
6 [Container] 2023/05/01 06:36:48 Phase context status code: YAML_FILE_ERROR Message: YAML file does not exist
7
```

codebuild가 필요로하는 .yml 파일 제작

vi buildspec.yml

```
version: 0.2
phases:
  pre_build:
    commands:
      - echo Logging in to Amazon ECR...
      - $(aws ecr get-login --no-include-email --region $AWS_DEFAULT_REGION)
  build:
    commands:
      - echo Build started on `date`
      - echo Building the Docker image...
      - docker build -t web:1 .
      - docker tag web:1 <자신의 ECR 리포지토리 URL>
  post_build:
    commands:
      - echo Build completed on `date`
      - echo Pushing the Docker image...
      - docker push <자신의 ECR 리포지토리 URL>
```

ECR에 해당하는 url 복사

Amazon ECR > 리포지토리

Private Public

프라이빗 리포지토리 (1)

리포지토리 찾기

리포지토리 이름	URI	생성 날짜	태그 변경 불가능	스캔 빈도	암호화 유형	풀스루 캐시
chancloud	767406634679.dkr.ecr.ap-northeast-2.amazonaws.com/chancloud	2023년 4월 30일, 15:11:27 (UTC+09)	비활성화됨	수동	AES-256	비활성

파일 제작 및 해당하는 부분에 경로 집어넣기

```
version: 0.2
phases:
  pre_build:
    commands:
      - echo Logging in to Amazon ECR...
      - $(aws ecr get-login --no-include-email --region $AWS_DEFAULT_REGION)

  build:
    commands:
      - echo Build started on `date`
      - echo Building the Docker image...
      - docker build -t web:1 .
      - docker tag web:1 767406634679.dkr.ecr.ap-northeast-2.amazonaws.com/chanccloud

  post_build:
    commands:
      - echo Build completed on `date`
      - echo Pushing the Docker image...
      - docker push 767406634679.dkr.ecr.ap-northeast-2.amazonaws.com/chanccloud
```

add하고 commit 하고 push

commit 바뀔때마다 구분하기위해 이름과 명칭을 다르게 하는게 좋다.

```
[root@ip-172-31-33-37 docker-chan-repo]# vi buildspec.yaml
[root@ip-172-31-33-37 docker-chan-repo]# ll
total 12
-rw-r--r--. 1 root root 162 Apr 30 05:44 Dockerfile
-rw-r--r--. 1 root root 563 May  1 06:48 buildspec.yaml
-rw-r--r--. 1 root root 163 Apr 30 06:05 index.html
[root@ip-172-31-33-37 docker-chan-repo]#
[root@ip-172-31-33-37 docker-chan-repo]# git add .
[root@ip-172-31-33-37 docker-chan-repo]#
[root@ip-172-31-33-37 docker-chan-repo]# git commit -am "my 2st commit"
[master 577ed21] my 2st commit
Committer: root <root@ip-172-31-33-37.ap-northeast-2.compute.internal>
Your name and email address were configured automatically based
on your username and hostname. Please check that they are accurate.
You can suppress this message by setting them explicitly. Run the
following command and follow the instructions in your editor to edit
your configuration file:

    git config --global --edit

After doing this, you may fix the identity used for this commit with:

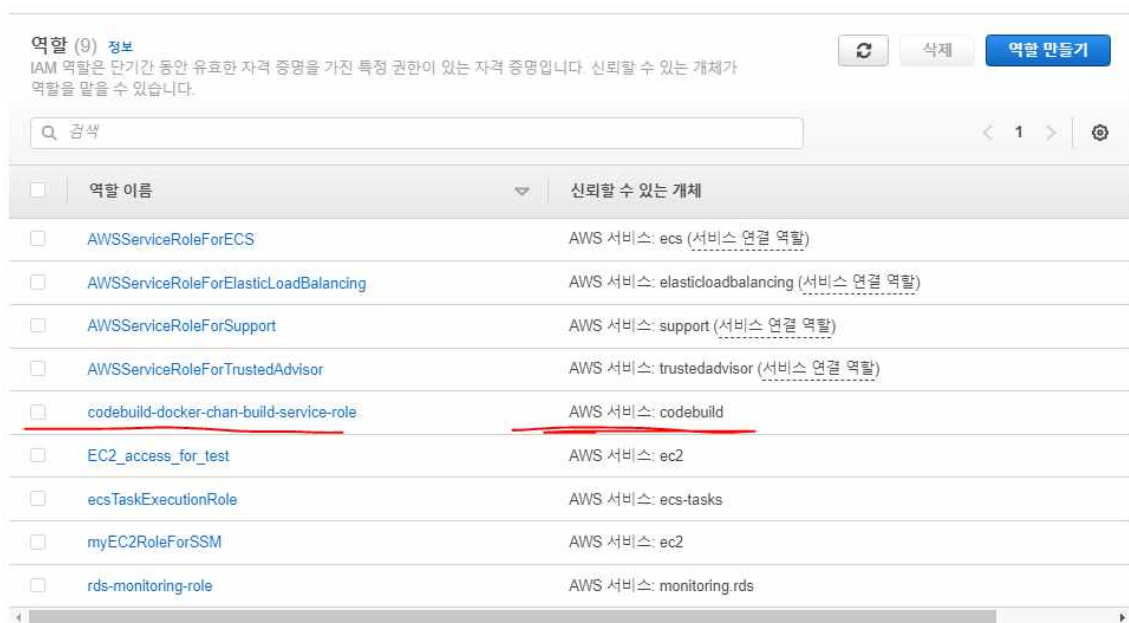
    git commit --amend --reset-author

1 file changed, 17 insertions(+)
create mode 100644 buildspec.yaml
[root@ip-172-31-33-37 docker-chan-repo]#
[root@ip-172-31-33-37 docker-chan-repo]# git push
Enumerating objects: 4, done.
Counting objects: 100% (4/4), done.
Compressing objects: 100% (3/3), done.
Writing objects: 100% (3/3), 616 bytes | 616.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
remote: Validating objects: 100%
To ssh://git-codecommit.ap-northeast-2.amazonaws.com/v1/repos/docker-chan-repo
   8e07088..577ed21  master -> master
[root@ip-172-31-33-37 docker-chan-repo]#
```

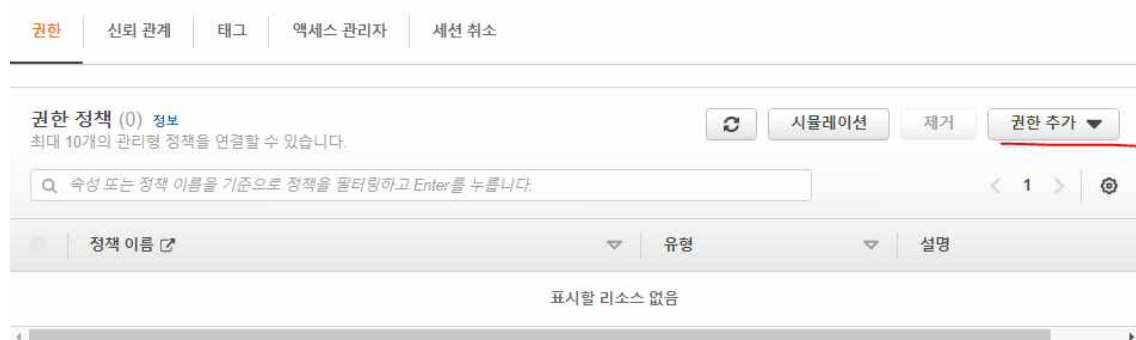
codebuild -> 왼쪽 선택란의 "코드" -> 레포지토리 선택 -> 파일이 올라와진걸 확인



코드빌드에 해당하는 역할 선택



권한추가 -> 정책연결



registry 검색 -> FullAccess -> 권한추가

codebuild를 통해 만들어진 이미지 파일이 ECR에 들어가도록 권한을 주어야한다. 이 실습에서는 편의상 FullAccess를 주었지만 앞에서 전 과정에서 설명한 것처럼 필요한 정책만 주는 것이 보안상 좋다.

기타 권한 정책 (선택됨 1/847) 정책 생성

Q 속성 또는 정책 이름을 기준으로 정책을 필터링하고 Enter를 누릅니다. 19 개 일치

"regis" X 필터 지우기

	정책 이름	유형	설명
<input type="checkbox"/>	AWSIoTThingsRegistration	AWS 관리형	This policy allows users to register things at bulk using AWS IoT StartThingRegistratio...
<input type="checkbox"/>	AWSCloudMapRegisterInstanceAccess	AWS 관리형	Provides registrant level access to AWS Cloud Map actions.
<input checked="" type="checkbox"/>	AmazonEC2ContainerRegistryFullAccess	AWS 관리형	Provides administrative access to Amazon ECR resources
<input type="checkbox"/>	AmazonEC2ContainerRegistryReadOnly	AWS 관리형	Provides read-only access to Amazon EC2 Container Registry repositories.
<input type="checkbox"/>	AmazonEC2ContainerRegistryPowerUser	AWS 관리형	Provides full access to Amazon EC2 Container Registry repositories, but does not allo...
<input type="checkbox"/>	AWSOpsWorksInstanceRegistration	AWS 관리형	Provides access for an Amazon EC2 instance to register with an AWS OpsWorks stack.

build 재시도시 실행된걸 볼 수 있다.

개발자 도구 > CodeBuild > 빌드 프로젝트 > docker-chan-build > docker-chan-build:8f3319a4-126e-4b92-ba85-e4adb4dd8751

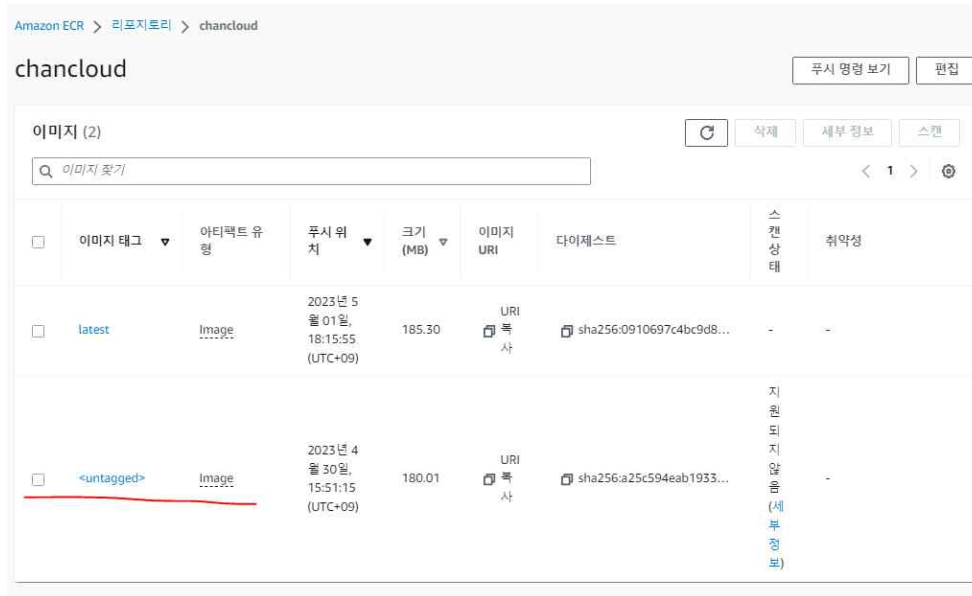
docker-chan-build:8f3319a4-126e-4b92-ba85-e4adb4dd8751 빌드 중지 빌드 재시도

빌드 상태

상태	시작한 사용자	빌드 ARN	해결된 소스 버전
🟢 성공함	aws.user01	arn:aws:codebuild:ap-northeast-2:767406634679:build/docker-chan-build:8f3319a4-126e-4b92-ba85-e4adb4dd8751	cd40abfe90ba645f485af035f8fc05a90b470b8f
시작 시간	종료 시간	빌드 번호	
5월 1, 2023 6:14 오후 (UTC+9:00)	5월 1, 2023 6:15 오후 (UTC+9:00)	2	

[빌드 로그](#) | [단계 세부 정보](#) | [보고서](#) | [환경 변수](#) | [빌드 세부 정보](#) | [리소스 사용률](#)

ECS repository에 들어가 새로 생성된 이미지를 확인 할 수 있다.



```
162 ---> Running in 0f9a80711484
163 Removing intermediate container 0f9a80711484
164 ---> 51ba96d1bc4b
165 Successfully built 51ba96d1bc4b
166 Successfully tagged web:1
167
168 [Container] 2023/05/01 09:15:42 Running command docker tag web:1 767406634679.dkr.ecr.ap-northeast-2.amazonaws.com/chancloud
169
170 [Container] 2023/05/01 09:15:42 Phase complete: BUILD State: SUCCEEDED
171 [Container] 2023/05/01 09:15:42 Phase context status code: Message:
172 [Container] 2023/05/01 09:15:42 Entering phase POST_BUILD
173 [Container] 2023/05/01 09:15:42 Running command echo Build completed on 'date'
174 Build completed on date
175
176 [Container] 2023/05/01 09:15:42 Running command echo Pushing the Docker image...
177 Pushing the Docker image...
178
179 [Container] 2023/05/01 09:15:42 Running command docker push 767406634679.dkr.ecr.ap-northeast-2.amazonaws.com/chancloud
180 Using default tag: latest
181 The push refers to repository [767406634679.dkr.ecr.ap-northeast-2.amazonaws.com/chancloud]
182 13724ca3892c: Preparing
183 7d227d7057b5: Preparing
184 9e87d03243db: Preparing
185 13724ca3892c: Pushed
186 7d227d7057b5: Pushed
187 9e87d03243db: Pushed
188 latest: digest: sha256:0910697c4bc9d8f0c3d2982c338766fe89777f7763475d43e3dde016c8fe0a1 size: 949
189
190 [Container] 2023/05/01 09:15:54 Phase complete: POST_BUILD State: SUCCEEDED
191 [Container] 2023/05/01 09:15:54 Phase context status code: Message:
192
```

LOG를 확인해 보자 buildspec.yml 을 통해 빌드된 내용이 ECS repository에 들어가는 걸 알 수있다 그렇다면 웹 페이지의 내용을 바꿔 새로운내용을 commit하면 그것을 토대로 build해서 ECS repository에들어가는 지 확인해 봐야한다.

즉, Code commit -> CodeBuild-> ECR

V1.0 -> V2.0으로 변경, version two로 변경

```
<html>
  <body>
    <h1> CI/CD & Docker example ChanHyuck </h1>
    <p style="background-color:MediumSeaGreen;"> Version two (V2.0) in Green Color. </p>
  </body>
</html>
```

git add-commit-push 작업

```
[root@ip-172-31-33-37 docker-chan-repo]# vi index.html
[root@ip-172-31-33-37 docker-chan-repo]# cat index.html
<html>
  <body>
    <h1> CI/CD & Docker example ChanHyuck </h1>
    <p style="background-color:MediumSeaGreen;"> Version two (V2.0) in Green Color. </p>
  </body>
</html>
[root@ip-172-31-33-37 docker-chan-repo]# git add .
[root@ip-172-31-33-37 docker-chan-repo]# git commit -am "version 2.0"
[master 1b7be22] version 2.0
Committer: root <root@ip-172-31-33-37.ap-northeast-2.compute.internal>
Your name and email address were configured automatically based
on your username and hostname. Please check that they are accurate.
You can suppress this message by setting them explicitly. Run the
following command and follow the instructions in your editor to edit
your configuration file:

    git config --global --edit

After doing this, you may fix the identity used for this commit with:

    git commit --amend --reset-author

1 file changed, 1 insertion(+), 1 deletion(-)
[root@ip-172-31-33-37 docker-chan-repo]#
[root@ip-172-31-33-37 docker-chan-repo]# git push
Enumerating objects: 5, done.
Counting objects: 100% (5/5), done.
Compressing objects: 100% (3/3), done.
Writing objects: 100% (3/3), 304 bytes | 304.00 KiB/s, done.
Total 3 (delta 2), reused 0 (delta 0), pack-reused 0
remote: Validating objects: 100%
To ssh://git-codecommit.ap-northeast-2.amazonaws.com/v1/repos/docker-chan-repo
cd40abf..1b7be22 master -> master
```

Code commit의 repository에 index.html 내용이 바뀐걸 알 수 있다.

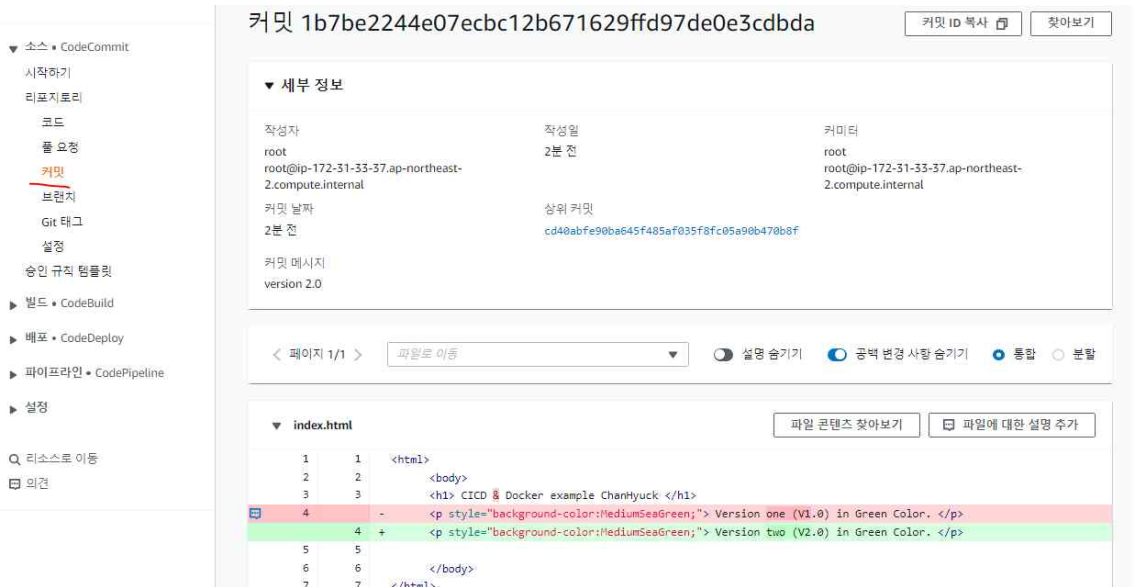
docker-chan-repo

알림 master 플 요청 생성 URL 복제

docker-chan-repo / index.html 정보 편집

```
1 <html>
2   <body>
3     <h1> CI/CD & Docker example ChanHyuck </h1>
4     <p style="background-color:MediumSeaGreen;"> Version two (V2.0) in Green Color. </p>
5   </body>
6 </html>
7
```

CodeCommit의 commit란의 바뀐버전을 클릭하면 어떤부분이 바뀌었는지 직관적으로 확인 가능하다.



code project로 들어가 project 재실행



프로젝트 빌드에서 그냥 실행해도 괜찮고 "재정의로 시작"에 들어가 위 사진처럼 바뀐 버전으로 수정해 build를 실행해도 된다.

성공된 모습을 확인 후 다시 ECS registories로 들어가 확인

docker-chan-build:fb20812e-0695-40e7-863e-0e6f3474c8a2

빌드 중지

빌드 재시도

빌드 상태

상태 🟢 성공함	시작한 사용자 aws.user01	빌드 ARN arn:aws:codebuild:ap-northeast-2:767406634679:build/docker-chan-build:fb20812e-0695-40e7-863e-0e6f3474c8a2	해결된 소스 버전 1b7be2244e07ecbc12b671629ffd97de0e3cdbda
시작 시간 5월 1, 2023 6:33 오후 (UTC+9:00)	종료 시간 5월 1, 2023 6:35 오후 (UTC+9:00)	빌드 번호 3	

Amazon ECR > 리포지토리 > chancloud

chancloud

푸시 명령 보기

편집

이미지 (3)

🔍 이미지 찾기

< 1 > ⚙️

<input type="checkbox"/>	이미지 태그 ▼	아티팩트 유형	푸시 위치 ▼	크기 (MB) ▼	이미지 URI	다이제스트	스캔 상태	취약성
<input type="checkbox"/>	latest	Image	2023년 5월 01일, 18:35:44 (UTC+09)	185.30	URI 복사	sha256:3daedef8c755f6b...	-	-
<input type="checkbox"/>	<untagged>	Image	2023년 5월 01일, 18:15:55 (UTC+09)	185.30	URI 복사	sha256:0910697c4bc9d8...	-	-
<input type="checkbox"/>	<untagged>	Image	2023년 4월 30일, 15:51:15 (UTC+09)	180.01	URI 복사	sha256:a25c594eab1933...	지원되지 않음 (세부 정보)	-

이렇게 만들어진 이미지 파일을 통해 컨테이너를 구현했을때 그 바뀐 버전이 잘 보이냐 확인 해봐야한다. (무턱대고 파이프라인으로 연결후 확인하면 나중에갔을때 오류가 발생했을 때 골치아파지는경우가 많다 그렇기 때문에 하나하나씩 차근차근 진행한다.)

cluster의 새 계정 생성에 들어가 새로 생성

새로운 ECS 환경
의견을 알려주세요.

Amazon Elastic
Container Service

클러스터
네임스페이스 [신규](#)
[태스크 정의](#)
[계정 설정](#) [신규](#)

AWS Copilot 설치

Amazon Elastic Container Service > 태스크 정의 > cicc-chan-task > 개정 1개 > 컨테이너

cicc-chan-task:1

배포

작업

새 개정 생성

개요 정보

ARN arn:aws:ecs:ap-northeast-2:7674016634679:task-definition/cicc-chan-task:1	상태 ACTIVE	생성된 시간 2023. 4. 30. 8시 17분 6초 UTC	업 환경 FARGATE
태스크 역할 -	태스크 실행 역할 ecsTaskExecutionRole	운영 체제/아키텍처 Linux/X86_64	네트워크 모드 awsvpc

바뀐 URI 를 통해 새로운 컨테이너로 제작해야한다.

chancloud

푸시 명령 보기

편집

이미지 (3)

이미지 찾기

삭제

세부 정보

스캔

<input type="checkbox"/>	이미지 태그	아티팩트 유형	푸시 위치	크기 (MB)	이미지 URI	다이제스트	스캔 상태	취약성
<input type="checkbox"/>	latest	Image	2023년 5월 01일, 18:35:44 (UTC+09)	185.30	URI 복사	sha256:3daedef8c755f6b...	-	-
<input type="checkbox"/>	<untagged>	Image	2023년 5월 01일, 18:15:55 (UTC+09)	185.30	URI 복사	sha256:0910697c4bc9d8...	-	-
<input type="checkbox"/>	<untagged>	Image	2023년 4월 30일, 15:51:15 (UTC+09)	180.01	URI 복사	sha256:a25c594eab1933...	지원되지 않음 (세부 정보)	-

생성 후 새로 생성된 task:2의 해당하는 서비스 생성(업데이트)

cicc-chan-task (1/2) 정보

값을 기준으로 작업 정의 개정 필터링

활성 작업 정의

서비스 생성

태스크 실행

태스크 정의: 개정

상태

<input checked="" type="checkbox"/>	cicc-chan-task:2	ACTIVE
<input type="checkbox"/>	cicc-chan-task:1	ACTIVE

직관적이지 않아서 옛날버전을 수정 후 진행
복사한 이미지 경로를 새로 넣어준다.

▼ 표준

컨테이너 이름* web1-chan ⓘ

이미지* 767406634679.dkr.ecr.ap-northeast-2.amazonaws.com/chancloud:lates ⓘ

프라이빗 레지스트리 인 증* ☐ ⓘ

메모리 제한(MiB) 소프트 제한 ▼ 128 ⓘ

+ 하드 제한 추가

컨테이너에 MiB 단위로 하드 및/또는 소프트 메모리 제한을 정의합니다. 하드 및 소프트 제한은 작업 정의에서 각각 "memory" 및 "memoryReservation" 파라미터에 해당합니다. ECS는 웹 애플리케이션 시작점으로 300-500MiB를 권장합니다.

포트 매핑 컨테이너 포트 프로토콜 ⓘ

80 tcp ⓘ

+ 포트 매핑 추가

task 역할 추가

태스크 정의 이름* cicd-chan-task ⓘ

태스크 역할 ecsTaskExecutionRole ⓘ

~~한층 높은 AWS 서비스에 API 요청을 할 때 작업이 사용할 수 있는 IAM 역할 옵션입니다. IAM 콘솔에서 Amazon Elastic Container Service 태스크 역할을 생성합니다. [🔗](#)~~

네트워크 모드 awsvpc ⓘ

을(를) 선택할 경우 ECS는 Docker의 기본 네트워킹 모드를 사용하여 컨테이너를 시작합니다. Docker의 기본 네트워킹 모드는 Windows의 Linux 브리지(Bridge) 및 NAT 브리지(Bridge)입니다. Windows 태스크는 및 awsvpc 네트워크 모드를 지원합니다.

ALB로 로드밸런싱 되는지 꼭 확인하자

상태 검사 유예 기간

로드 밸런싱

서비스 생성 시에 로드 밸런싱 설정을 설정할 수 있습니다.

Load Balancer 이름 ALB-chan

컨테이너 이름: web1-chan

컨테이너 포트: 80

대상 그룹: arn:aws:elasticloadbalancing:ap-northeast-2:767406634679:targetgroup/ALB-chan-tg/34c5cd43fd4ec197

*필수 항목

취소

이전

다음 단계

작업태그의 새로운 task에 해당하는 컨테이너들이 실행되는걸 확인

세부 정보

작업

이벤트

Auto Scaling

배포

측정치

태그

로그

5월 1, 2023 7:24:36 오후에 최종 업데이트(0분 전)

작업 상태:

Running

 Stopped

이 페이지의 필터

< 1~4 > 페이지 크기 50

작업	작업 정의	마지막 상태	원하는 상태	그룹	시작 유형
29054038186f4347976ff0...	cicd-chan-task: 1	RUNNING	RUNNING	service:cicdcd-chan-service	FARGATE
dfe34a01d6ce44659815c...	cicd-chan-task: 1	RUNNING	RUNNING	service:cicdcd-chan-service	FARGATE
6255e4651a5e47efabf8b...	cicd-chan-task: 3	PROVISIONING	RUNNING	service:cicdcd-chan-service	FARGATE
6abac34954954f998b0f8...	cicd-chan-task: 3	PROVISIONING	RUNNING	service:cicdcd-chan-service	FARGATE

새로운 버전으로 컨테이너가 실행된 것이 보인다.

세부 정보

작업

이벤트

Auto Scaling

배포

측정치

태그

로그

5월 1, 2023 7:26:08 오후에 최종 업데이트(0분 전)

작업 상태: Running Stopped

이 페이지의 필터

< 1~2 > 페이지 크기 50

작업	작업 정의	마지막 상태	원하는 상태	그룹	시작 유형
6255e4651a5e47efabf8b...	cicd-chan-task:3	RUNNING	RUNNING	service:cicdcd-chan-service	FARGATE
6abac34954954f998b0f8...	cicd-chan-task:3	RUNNING	RUNNING	service:cicdcd-chan-service	FARGATE

웹 페이지로 요청을해 바뀐 버전 확인

CICD & Docker example ChanHyuck

Version two (V2.0) in Green Color.

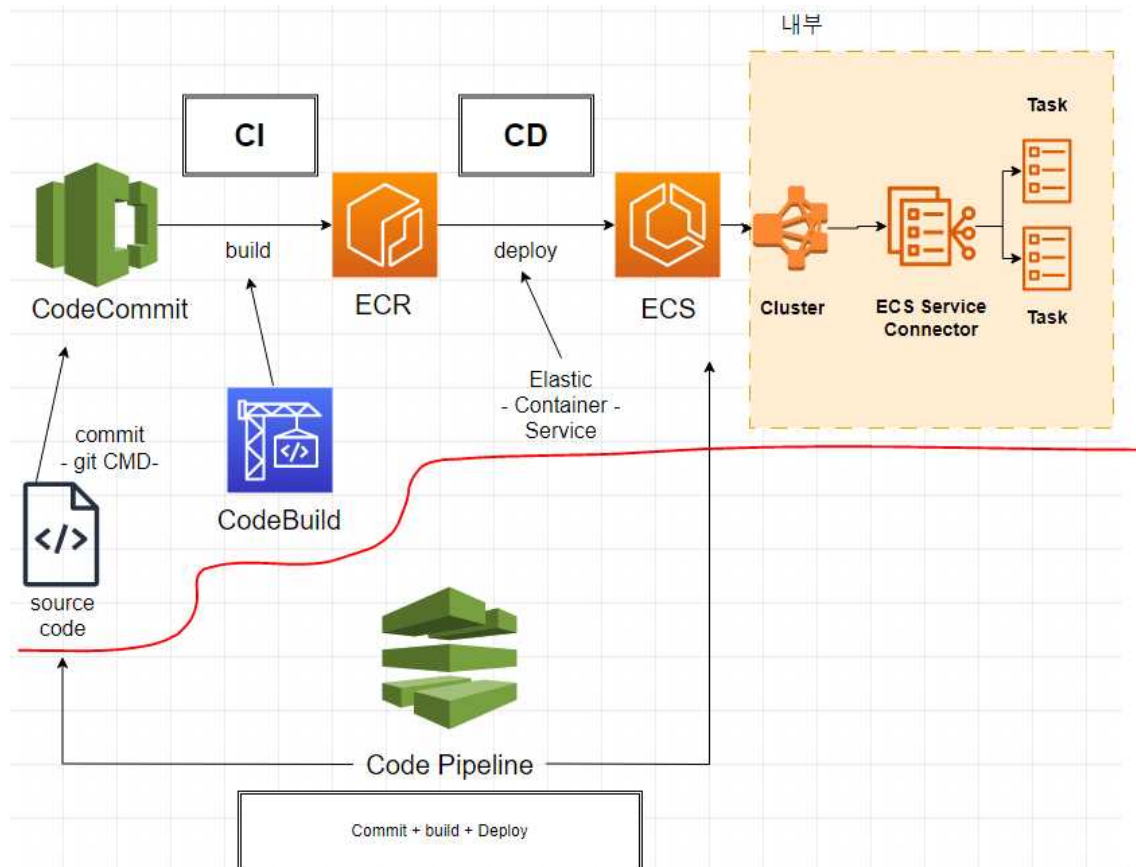
현재 까지는 코드를 commit 해서 바뀐 코드를 통해 codebuild를 한 후 수동으로 바뀐 이미지 버전을 deploy해서 컨테이너를 작업하는 과정을 했으며 간단하게 정리하자면

commit - CodeCommit

build - CodeBuild

deploy - ECS

를 사용한 것이다.



소스 코드를 수정해서 올린후 그것에따른 이미지 build 작업을 수행한후 ECS를 통해 배포하는 작업을 했으며 바로 파이프라인으로 연결하지 않고 이 작업을 수동한 이유는 하나하나 해 보가며 오작동이있나 체크하고 연동이 잘된걸 확인하고 pipeline으로 제공하기 위해서이다.

파이프라인 구축

서비스에서 CodePipeline 검색후 이름 설정

개발자 도구 > CodePipeline > 파이프라인 > 새 파이프라인 생성

Step 1
파이프라인 설정 선택

Step 2
소스 스테이지 추가

Step 3
빌드 스테이지 추가

Step 4
배포 스테이지 추가

Step 5
검토

파이프라인 설정 선택 정보

파이프라인 설정

파이프라인 이름
파이프라인 이름을 입력합니다. 생성된 후에는 파이프라인 이름을 편집할 수 없습니다.

cicd-pipeline-chan

100자를 초과할 수 없음

서비스 역할

☒ 새 서비스 역할
계정에 서비스 역할 생성

☐ 기존 서비스 역할
계정에서 기존 서비스 역할 선택

역할 이름

AWSCodePipelineServiceRole-ap-northeast-2-cicd-pipeline-chan

서비스 역할 이름 입력

☒ AWS CodePipeline이 이 새 파이프라인에 사용할 서비스 역할을 생성하도록 허용

1) commit 이름 설정

소스 스테이지 추가 정보

소스

소스 공급자
파이프라인의 입력 아티팩트를 저장한 위치입니다. 공급자를 선택한 후 연결 세부 정보를 제공하십시오.

AWS CodeCommit

리포지토리 이름
이미 생성하여 소스 코드를 푸시한 리포지토리를 선택합니다.

Q docker-chan-repo X

브랜치 이름
리포지토리의 브랜치 선택

Q master X

변경 감지 옵션
소스 코드에서 변경이 발생하면 파이프라인을 자동으로 시작하는 감지 모드를 선택합니다.

☒ Amazon CloudWatch Events(권장)
변경이 발생할 때 Amazon CloudWatch Events를 사용하여 파이프라인을 자동으로 시작합니다.

☐ AWS CodePipeline
AWS CodePipeline을 사용하여 정기적으로 변경 사항을 확인합니다.

출력 아티팩트 형식
출력 아티팩트 형식을 선택합니다.

☒ CodePipeline 기본값
AWS CodePipeline은 파이프라인의 아티팩트에 대해 기본 zip 형식을 사용합니다. 리포지토리에 대한 git 메타데이터는 포함하지 않습니다.

☐ 전체 복제
AWS CodePipeline은 후속 작업에서 전체 git 복제를 수할할 수 있도록 리포지토리에 대한 메타데이터를 전달합니다. AWS CodeBuild 작업에 대해서만 지원됩니다.

취소 이전 다음

2) build 스테이지 추가

앞에 설정해준 code build 이름을 추가해주면 된다.

빌드 스테이지 추가 정보

빌드 - 선택 사항

빌드 공급자
빌드 프로젝트의 도구입니다. 운영 체제, 빌드 사양 파일, 출력 파일 이름 등 빌드 아티팩트 세부 정보를 제공하십시오.

AWS CodeBuild ▼

리전

아시아 태평양 (서울) ▼

프로젝트 이름
AWS CodeBuild 콘솔에서 이미 생성한 빌드 프로젝트를 선택합니다. 또는 AWS CodeBuild 콘솔에서 빌드 프로젝트를 생성한 후 이 작업으로 돌아옵니다.

Q docker-chan-build X

 또는

프로젝트 생성

환경 변수 - 선택 사항
CodeBuild 환경 변수의 키, 값 및 유형을 선택합니다. 값 필드에서 CodePipeline에서 생성된 변수를 참조할 수 있습니다. [자세히 알아보기](#)

환경 변수 추가

빌드 유형

☒ 단일 빌드
단일 빌드를 트리거합니다.

☐ 배치 빌드
여러 빌드를 단일 실행으로 트리거합니다.

취소

이전

빌드 스테이지 건너뛰기

다음

3) 배포 스테이지 추가

배포 공급자는 Amazon ECS로 설정

배포 스테이지 추가 [정보](#)

배포 - 선택 사항

배포 공급자
인스턴스에 배포하는 방법을 선택합니다. 공급자를 선택한 후 해당 공급자에 대한 구성 세부 정보를 제공하십시오.

Amazon ECS ▼

리전

아시아 태평양 (서울) ▼

클러스터 이름
Amazon ECS 콘솔에서 이미 생성한 클러스터를 선택합니다. 또는 Amazon ECS 콘솔에서 클러스터를 생성한 후 이 작업으로 돌아옵니다.

Q cicc-cha-cluster X

서비스 이름
Amazon ECS 콘솔에서 클러스터에 대해 이미 생성한 서비스를 선택합니다. 또는 Amazon ECS 콘솔에서 새 서비스를 생성한 후 이 작업으로 돌아옵니다.

Q cicc-cha-service X

이미지 정의 파일 - 선택 사항
서비스의 컨테이너 이름을 설명하는 JSON 파일과 이미지, 태그를 입력합니다.

MyFilename.json

배포 제한 시간 - 선택 사항
배포 작업에 대한 제한 시간(분)을 입력합니다.

취소

이전

배포 스테이지 건너뛰기

다음

4) 파이프라인 연동 확인

아래에 deploy부분을 보면 권한부족 오류가 뜰걸 볼 수 있다 아마 정책 추가 같은데 자세히 들어가 확인해서 수정한다.

The screenshot displays the AWS CodePipeline console for a pipeline named 'cicd-pipeline'. The 'Source' stage (AWS CodeCommit) is successful. The 'Build' stage (AWS CodeBuild) is also successful. The 'Deploy' stage (Amazon ECS) has failed with the error '권한 부족' (Insufficient permissions). Below the stages, a summary table shows the pipeline status as '실패' (Failed). A detailed error message at the bottom states: 'Unable to access the artifact with Amazon S3 object key 'cicd-pipeline-chan/BuildArtif/LOhtSfQ' located in the Amazon S3 artifact bucket 'codepipeline-ap-northeast-2-657734784216'. The provided role does not have sufficient permissions.'

Source (AWS CodeCommit) - 성공 - 2분 전

Build (AWS CodeBuild) - 성공 - 방금

Deploy (Amazon ECS) - 실패 - 방금

실행 요약

상태	시작됨	완료됨	기간
실패	4분 전	2분 전	2분 9초

트리거: CreatePipeline - user/aws.user01

최신 작업 실행 메시지
Unable to access the artifact with Amazon S3 object key 'cicd-pipeline-chan/BuildArtif/LOhtSfQ' located in the Amazon S3 artifact bucket 'codepipeline-ap-northeast-2-657734784216'. The provided role does not have sufficient permissions.

--> 제공된 역할에 충분한 권한이 없다는 걸 알 수 있다. 앞서 했던 작업처럼 iam 서비스에 들어가 권한을 추가해주자.

4)-1 파이프라인 역할 정책 추가

IAM > 역할

역할 (11) 정보
IAM 역할은 단기간 동안 유효한 자격 증명을 가진 특정 권한이 있는 자격 증명입니다. 신뢰할 수 있는 개체가 역할을 맡을 수 있습니다.

2 개 일치

역할 이름	신뢰할 수 있는 개체	마지막 활동
AWSCodePipelineServiceRole-ap-northeast-2-cicd-pipelir	AWS 서비스: codepipeline	-
cwe-role-ap-northeast-2-cicd-pipeline-chan	AWS 서비스: events	-

S3에 대한 정책 추가

기타 권한 정책 (선택됨 1/848)

9 개 일치

"s3" 필터 지우기

정책 이름	유형	설명
<input type="checkbox"/> AmazonDMSRedshiftS3Role	AWS 관리형	Provides access to manage S3 settings for Redshift endpoints for DMS.
<input checked="" type="checkbox"/> AmazonS3FullAccess	AWS 관리형	Provides full access to all buckets via the AWS Management Console.
<input type="checkbox"/> QuickSightAccessForS3StorageManagementAnalyticsReadO...	AWS 관리형	Policy used by QuickSight team to access customer data produced by S3 Storage Ma...
<input type="checkbox"/> AmazonS3ReadOnlyAccess	AWS 관리형	Provides read only access to all buckets via the AWS Management Console.
<input type="checkbox"/> AmazonS3OutpostsFullAccess	AWS 관리형	Provides full access to Amazon S3 on Outposts via the AWS Management Console.
<input type="checkbox"/> AWSBackupServiceRolePolicyForS3Backup	AWS 관리형	Policy containing permissions necessary for AWS Backup to backup data in any S3 bu...
<input type="checkbox"/> AWSBackupServiceRolePolicyForS3Restore	AWS 관리형	Policy containing permissions necessary for AWS Backup to restore a S3 backup to a ...
<input type="checkbox"/> AmazonS3ObjectLambdaExecutionRolePolicy	AWS 관리형	Provides AWS Lambda functions permissions to interact with Amazon S3 Object Lamb...
<input type="checkbox"/> AmazonS3OutpostsReadOnlyAccess	AWS 관리형	Provides read only access to Amazon S3 on Outposts via the AWS Management Cons...

권한 정책 (2) 정보
최대 10개의 관리형 정책을 연결할 수 있습니다.

시뮬레이션 제거 권한 추가

9 개 일치

정책 이름	유형	설명
<input type="checkbox"/> AWSCodePipelineServiceRole-ap-northeast-2-cicd-pipeline-chan	고객 관리형	Policy used in trust relationship w
<input type="checkbox"/> AmazonS3FullAccess	AWS 관리형	Provides full access to all buckets

pipeline 연결 재시도

오류발생 - 아마 다른 문제가 요인인것같다

Deploy 실패
파이프라인 실패 ID: ffe8745e-1b06-4b04-9e6e-8d0ed91a5ee3

재시도

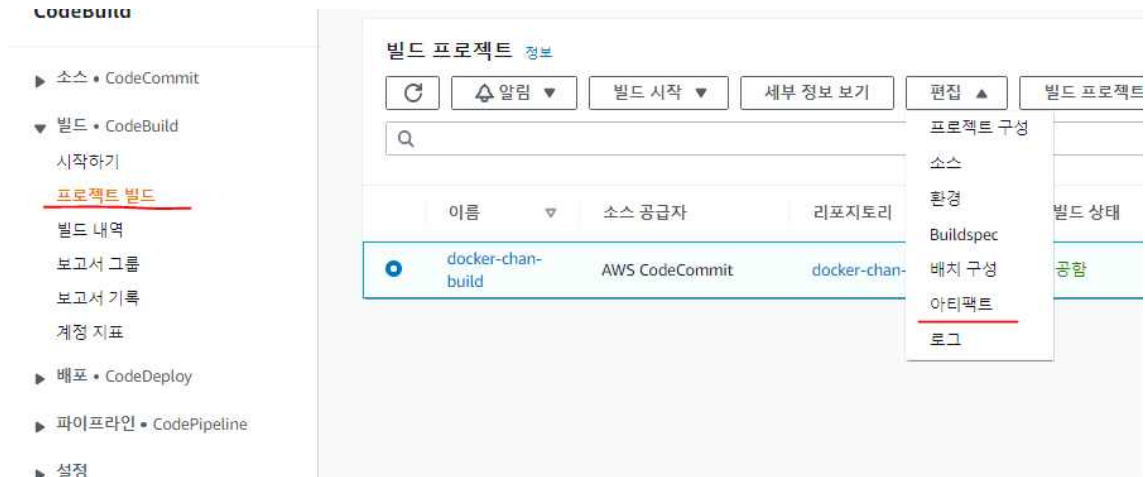
Deploy
Amazon ECS

실패 - 10분 전
권한 부족

1b7be224 Source: version 2.0

4)-2 CodeBuild 부분의 아티팩트 부분 수정

(꼭 읽기) Codebuild에서 [아티팩트]에서는 빌드 후 생성되는 아티팩트에 대한 설정을 하는 부분인데 우선 이 아티팩트 패키징을 왜 설정하는지 알아야하는데 codedeploy를 할 때 zip이나 tar등으로 패키징 된 아티팩트 파일을 서버로 가져와서 배포를 하기 때문에 S3에 아티팩트를 저장시켜주면서 deploy가 이 파일을 보고 deploy해줄도록 설정해 준것이다.



아티팩트 설정 후 생성

아티팩트

아티팩트 추가

아티팩트 1 - 기본

유형

Amazon S3

테스트를 실행하거나 도커 이미지를 Amazon ECR에 넣는 경우 [아티팩트 없음]을 선택할 수 있습니다.

버킷 이름

codepipeline-ap-northeast-2-657734784216

이름

output

출력 아티팩트를 포함할 버킷에 있는 폴더 또는 압축 파일의 이름입니다. [추가 구성]에서 [아티팩트 패키징]을 사용하여 폴더를 사용할지 아니면 압축 파일을 사용할지 선택합니다. 이름을 제공하지 않으면 기본적으로 프로젝트 이름으로 설정됩니다.

☐ 의미 체계 버전 관리 활성화

buildspec 파일에 지정된 아티팩트 이름 사용

경로 - 선택 사항

빌드 출력 ZIP 파일 또는 폴더의 경로입니다.

/

예: MyPath/MyArtifact.zip

네임스페이스 유형 - 선택 사항

없음

[빌드 ID]를 선택하여 빌드 출력 ZIP 파일 또는 폴더의 경로(예: MyPath/MyBuildID/MyArtifact.zip)에 빌드 ID를 삽입합니다. 또는 [없음]을 선택합니다.

아티팩트 패키징

☐ 없음
아티팩트 파일이 버킷에 업로드됩니다.

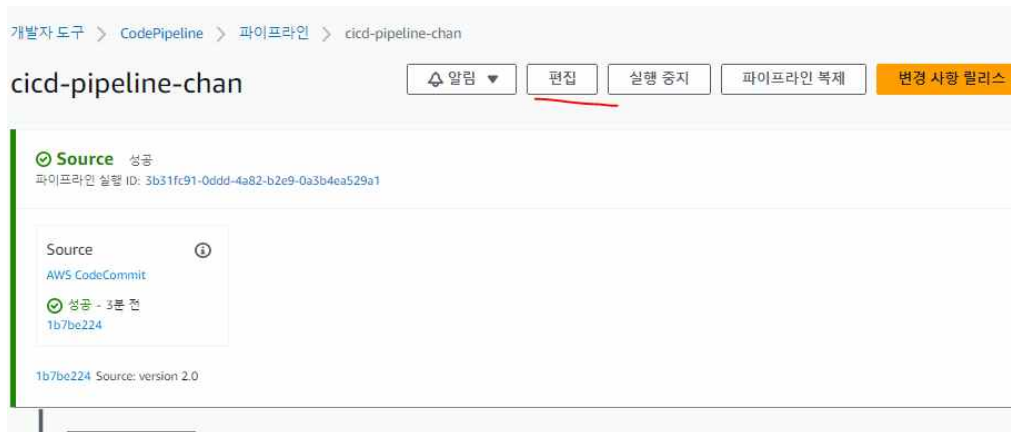
☒ Zip
AWS CodeBuild가 아티팩트를 압축 파일로 업로드합니다(이 압축 파일은 지정된 버킷에 놓음).

[아티팩트] 빌드 후 생성되는 아티팩트에 대한 설정을 하는 부분으로 여기서는 S3에 넣음
[버킷 이름] 아티팩트가 들어갈 버킷 이름을 넣는다.
[경로] 버킷 다음 디렉토리 명들을 적는다.

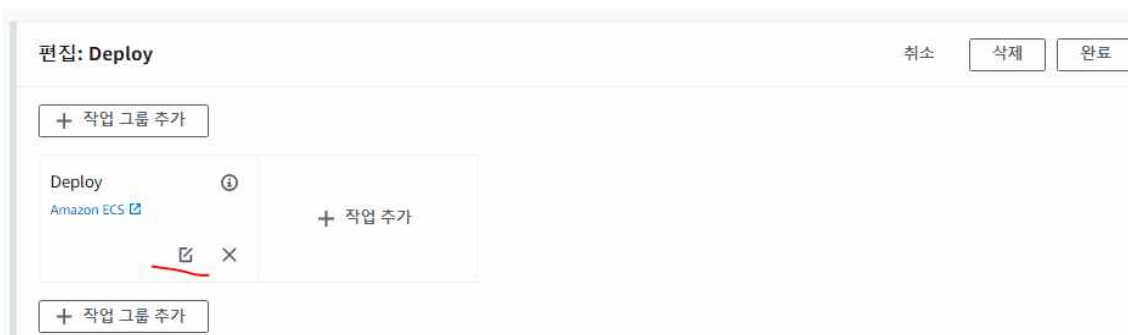
변경사항 릴리즈



pipeline의 편집으로 들어가 설정 변경



deploy 우측 상단의 "편집"에 들어간뒤 안에 편집 그림 클릭



deploy 편집 작업 수행

deploy에서 ECS 표준 배포에는 배포 작업에 대한 입력으로 imagedefinitions.json파일이 필요합니다 그렇기때문에 그거에대한 설정을 해주되 인스턴스에서 ~.json을 만들어서 commit 해 제공해 줄 예정이다.

작업 이름

작업의 이름을 선택합니다.

Deploy

100자 이하

작업 공급자

Amazon ECS

리전

아시아 태평양 (서울)

입력 아티팩트

이 작업의 입력 아티팩트를 선택합니다. 자세히 알아보기

SourceArtifact

100자 이하

클러스터 이름

Amazon ECS 콘솔에서 이미 생성한 클러스터를 선택합니다. 또는 Amazon ECS 콘솔에서 클러스터를 생성한 후 이 작업으로 돌아옵니다.

cidcd-chan-cluster

X

↺

서비스 이름

Amazon ECS 콘솔에서 클러스터에 대해 이미 생성한 서비스를 선택합니다. 또는 Amazon ECS 콘솔에서 새 서비스를 생성한 후 이 작업으로 돌아옵니다.

cidcd-chan-service

X

↺

이미지 정의 파일 - 선택 사항

서비스의 컨테이너 이름, 이미지 및 태그를 설명하는 JSON 파일을 입력합니다.

imagedefinitions.json

배포 제한 시간 - 선택 사항

배포 작업에 대한 제한 시간(분)을 입력합니다.

변수 네임스페이스 - 선택 사항

이 작업의 출력 변수에 대해 네임스페이스를 선택합니다. 이 작업에서 생성된 변수를 구성에 사용하려면 네임스페이스를 선택해야 합니다. 자세히 알아보기

DeployVariables

설정 저장

g: cicd-pipeline-chan

역재

취소

저장

Source

스태이지 편집

파이프라인 변경 사항 저장

X

변경 사항을 저장하면 실행 취소할 수 없습니다. 파이프라인이 실행되는 동안 변경 사항을 저장하면 실행이 완료되지 않습니다.

취소

저장

Build

스태이지 편집

vi imagedefinitions.json

```
{
  {
    "name": "ECS 컨테이너 이름",
    "imageUri": "ECS Repositories URI"
  }
}
```



```
{
  {
    "name": "web1-chan",
    "imageUri": "767406634679.dkr.ecr.ap-northeast-2.amazonaws.com/chanccloud:latest"
  }
}
```

파이프라인 구현 완료 및 테스트

🟢 **Source** 성공

파이프라인 실행 ID: [bee1b8d6-6658-42a7-9e6d-85c63a4cd243](#)

Source ⓘ

AWS CodeCommit

🟢 성공 - 5분 전

[735ac333](#)

[735ac333](#) Source: chan-imagedefinitions.json



전환 비활성화

🟢 **Build** 성공

파이프라인 실행 ID: [bee1b8d6-6658-42a7-9e6d-85c63a4cd243](#)

Build ⓘ

AWS CodeBuild

🟢 성공 - 3분 전

[세부 정보](#)

[735ac333](#) Source: chan-imagedefinitions.json



전환 비활성화

🟢 **Deploy** 성공

파이프라인 실행 ID: [bee1b8d6-6658-42a7-9e6d-85c63a4cd243](#)

Deploy ⓘ

Amazon ECS [🔗](#)

index.html 편집

vi index.html

```
<html>
  <body>
    <h1> CI/CD & Docker example Completion By ChanHyuck </h1>
    <p style="background-color:Purple;"> Version three (V3.0) in Purple Color. </p>
  </body>
</html>
```

원하는 설정으로 변경후 V.3로 배포

```
[root@ip-172-31-33-37 docker-chan-repo]# ll
total 16
-rw-r--r--. 1 root root 162 May  1 08:44 Dockerfile
-rw-r--r--. 1 root root 574 May  1 08:44 buildspec.yml
-rw-r--r--. 1 root root 159 May  1 12:07 imagedefinitions.json
-rw-r--r--. 1 root root 163 May  1 09:28 index.html
[root@ip-172-31-33-37 docker-chan-repo]#
[root@ip-172-31-33-37 docker-chan-repo]# vi index.html
[root@ip-172-31-33-37 docker-chan-repo]#
[root@ip-172-31-33-37 docker-chan-repo]# git add .
[root@ip-172-31-33-37 docker-chan-repo]#
[root@ip-172-31-33-37 docker-chan-repo]# git commit -am "version 3.0"
[master 9353ea5] version 3.0
  Committer: root <root@ip-172-31-33-37.ap-northeast-2.compute.internal>
  Your name and email address were configured automatically based
  on your username and hostname. Please check that they are accurate.
  You can suppress this message by setting them explicitly. Run the
  following command and follow the instructions in your editor to edit
  your configuration file:

    git config --global --edit

  After doing this, you may fix the identity used for this commit with:

    git commit --amend --reset-author

1 file changed, 2 insertions(+), 2 deletions(-)
[root@ip-172-31-33-37 docker-chan-repo]#
[root@ip-172-31-33-37 docker-chan-repo]# git push
Enumerating objects: 5, done.
Counting objects: 100% (5/5), done.
Compressing objects: 100% (3/3), done.
Writing objects: 100% (3/3), 410 bytes | 410.00 KiB/s, done.
Total 3 (delta 1), reused 0 (delta 0), pack-reused 0
remote: Validating objects: 100%
To ssh://git-codecommit.ap-northeast-2.amazonaws.com/v1/repos/docker-chan-repo
  735ac33..9353ea5 master -> master
[root@ip-172-31-33-37 docker-chan-repo]#
```

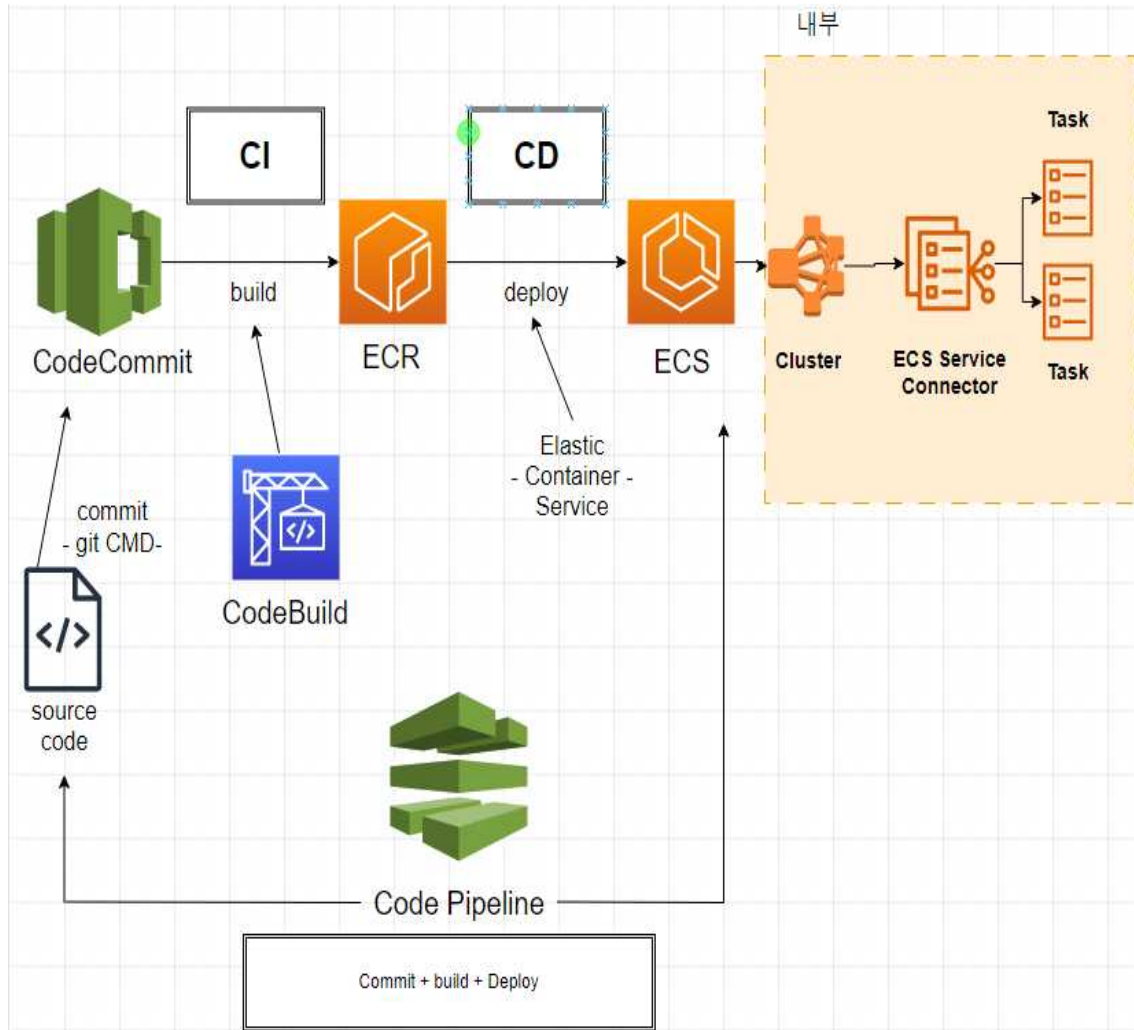
웹페이지 다시 요청

변경한 코드를 받아서 설정된 걸 볼 수 있다.

CI/CD & Docker example Completion By ChanHyuck

Version three (V3.0) in Purple Color.

<전체적인 완성된 그림>



다음 그림과같이 AWS를통한 CI/CD 구축을 완료 했으며 서비스를 무엇을 쓰느냐에 따라 다르게 구현할 수있다 예를 들어 CodeCommit repository에는 GitHub를 쓰거나 ECR 대신 Dockerhub 사용 하던 혹은 deploy를 EKS로 제공하던 다른 다양한 서비스로 제공 할 수 있다. 결국 어떤 서비스를 사용하는가는 중요하지 않고 Commit + Build + Deploy 과정을 자동화 할 수 있도록 구현하는 것이 중요하다.

[참고]

<https://hsunnystory.tistory.com/211>

<https://catalog.workshops.aws/cicdonaws/ko-KR/lab04/2-codedeploy-stage>

<https://yoo11052.tistory.com/m/101>

<https://woono.tistory.com/123>

<https://www.youtube.com/watch?v=d7PTjQiahOQ>

<https://a-half-human-half-developer.tistory.com/161>