**틀라우드 클라우드 구축**

**장수민**

1 . VPC 구축

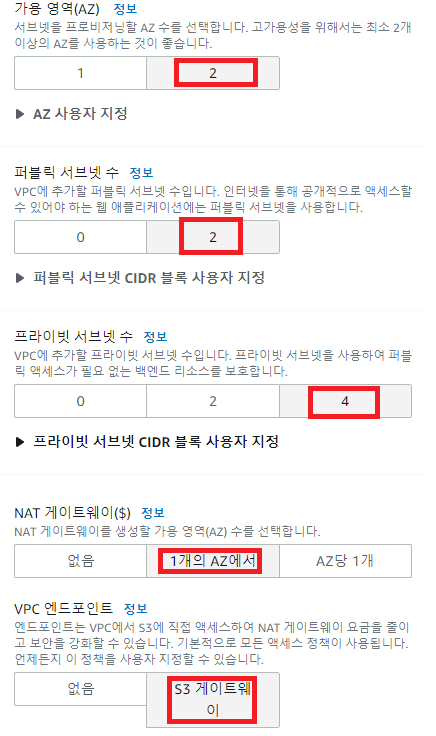
VPC 생성

-현재 프로젝트에서 VPC를 2개 만들고 VPC안의 인스턴스가 서로 통신할 수 있게 해주기위해 피러링을 사용할 것입니다



피어링을 하려면 VPC간에 겹치는 CICD 블록이 있으면 안되기 때문에 VPC를 만들 때 하나는 IPv4 CICD블록 설정을 10.0.0.0/16 다른 하나는 1.0.0.0/16 로 설정 했습니다





생성 확인

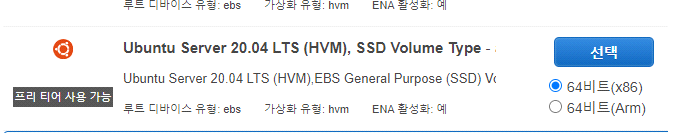


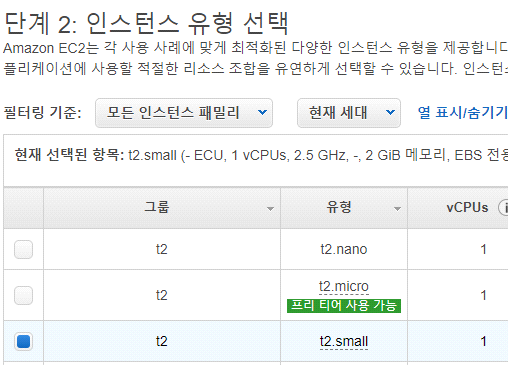
서브넷 확인 ( 각 서브넷을 어디에 쓸것인지 이름을 적어주면 좋다)



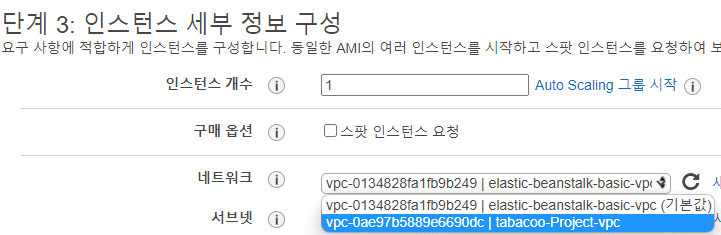
EC2 인스턴스 생성

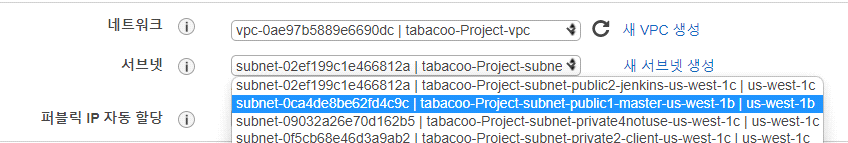
매뉴얼에는 인스턴스 유형을 t2.small로 하였지만 실제 프로젝트에서 돌고있는 젠킨스용 ec2는 t2.midium으로 설정했습니다. 후에 있을 안드로이드 CICD를 하기위해서는 t2.small유형의 용량이 부족하기 때문입니다



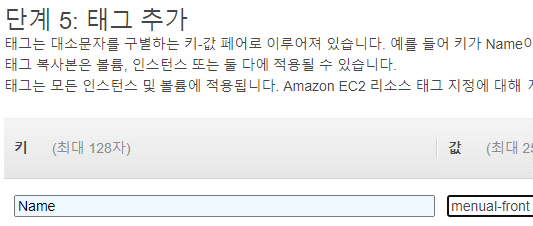


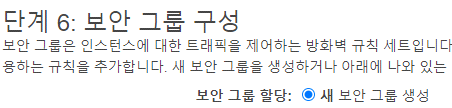
네트워크에 만들었던 vpc 선택



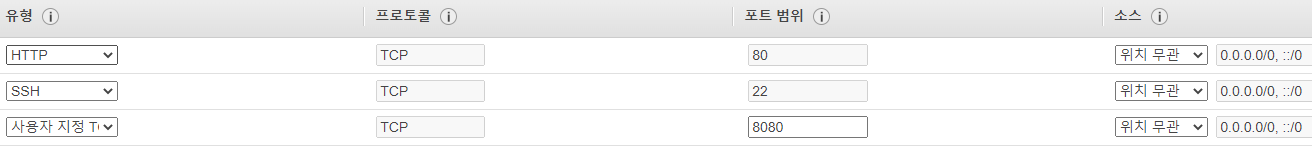


퍼블릭 ip는 만들고자하는 ec2의 용도에 맞게 public이면 활성화 private이면 비활성화를 선택하면 됩니다

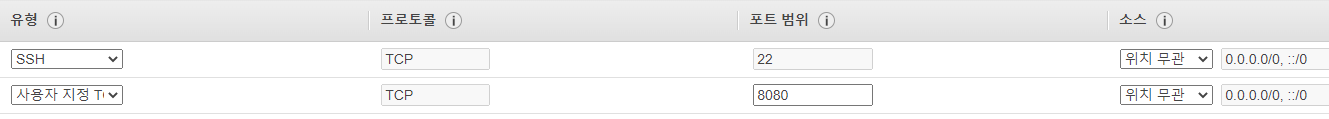




보안그룹 public 용



보안그룹 private 용



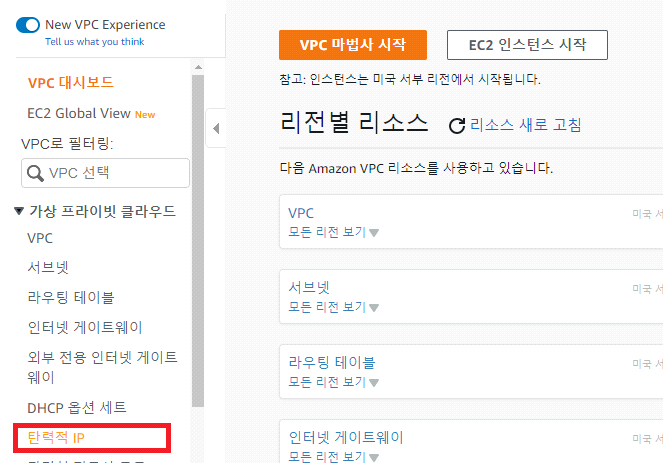
키페어 생성후 다운로드하고 인스턴스 시작

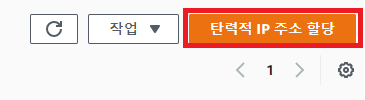




탄력적 ip 생성

먼저 탄력적 IP를 생성한다 (ec2가 중단이 된 후 ip가 자동으로 바뀌게 되는이 이것을 탄력적 ip가 고정시켜준다





할당 설정 값은 기본값으로 생성

탄력적 ip 연결

EC2에 할당된 탄력적 ip주소 연결

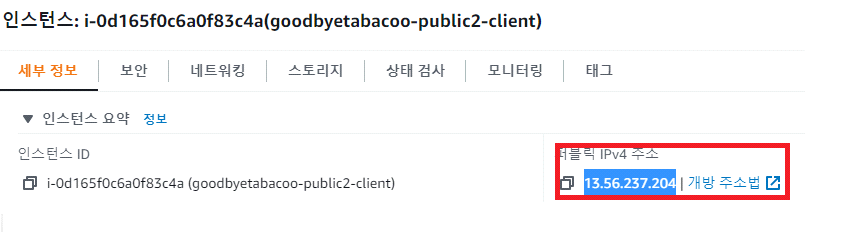


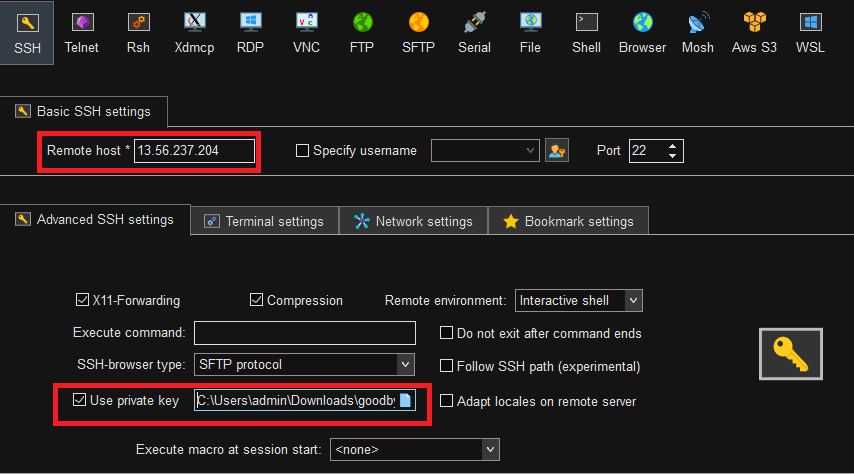
연결할 ec2 선택후 연결하면 ip가 고정이 된다



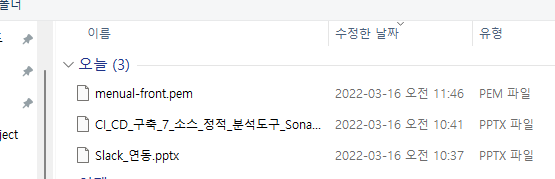
**EC2 public 서버 접속**

MobaXterm의 ssh로 접속

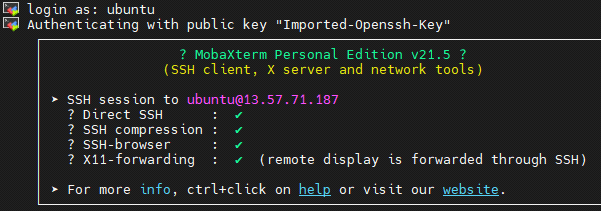
접속할 인스턴스의 퍼블릭ip 주소를 입력



인스턴스에 접속하기 위해 발급받은 키 파일(.pem) 선택



Ubuntu로 접속해준다



jenkins는 java버전 8을 쓰기 때문에 jdk버전 8을 다운받는다

먼저 apt update를 한후에



jdk버전 11을 다운받는다



패스 경로 설정

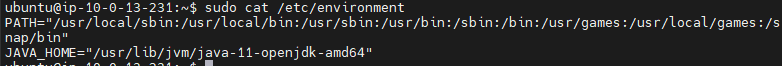
$ sudo nano /etc/environment

JAVA\_HOME="/usr/lib/jvm/java-11-openjdk-amd64"



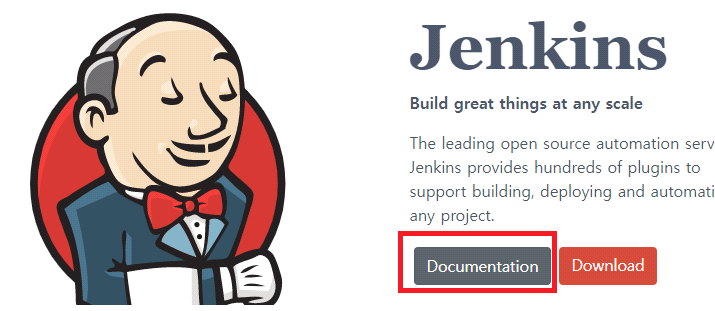
패스 경로 확인

$ sudo cat /etc/environment



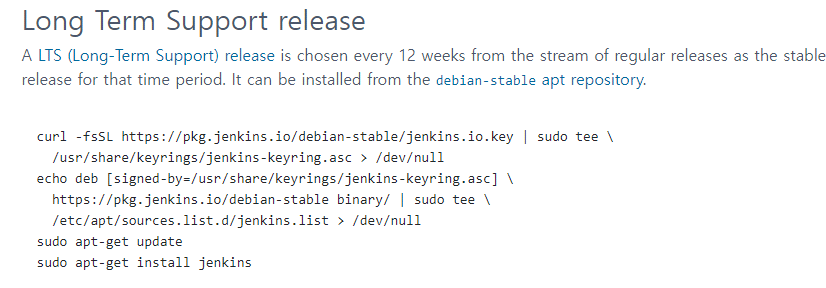
젠킨스 설치

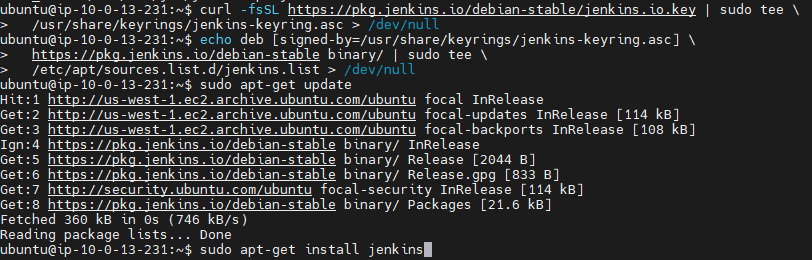
<https://www.jenkins.io/> 홈페이지 들어가서 Documentation 클릭



Installing Jenkins 에 Linux 클릭 하면 나오는 코드로 Linux에 jenkins를 설치한다

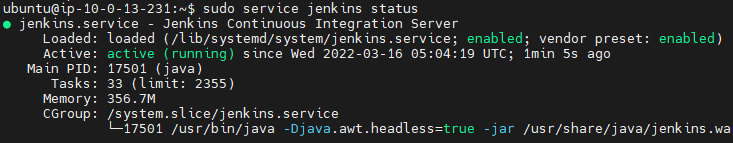






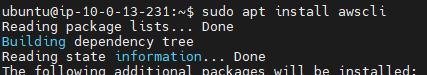
젠킨스 설치확인

sudo service jenkins status



aws명령어 사용을 위한 awscli 설치

Sudo apt install awscli

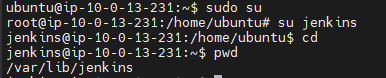


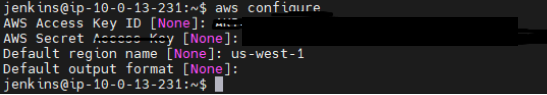
Aws 설치 확인

aws —version



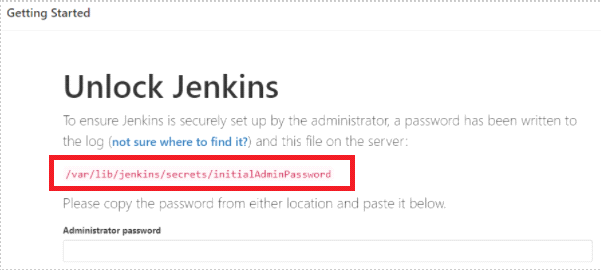
Aws 계정 설정





설치 후 ec2의 {ip주소}:8080을 주소창에 입력하면 아래처럼 뜹니다

아래의 빨간 박스 안에는 최초 실행시 비밀번호 경로 확인하는 코드입니다



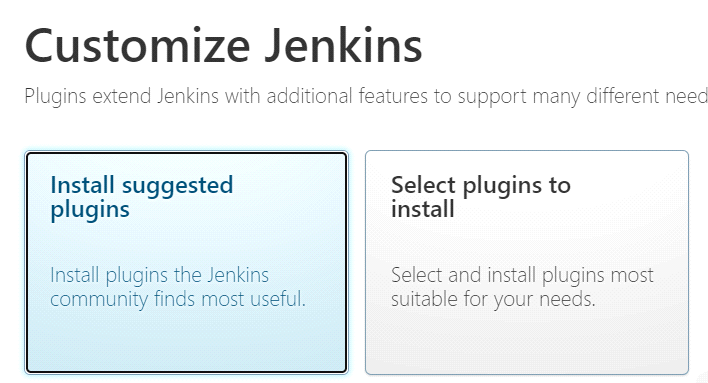
Jenkins 에서 ubuntu로 나온후

최초실행시 비밀번호 경로를 확인하기위해

Sudo cat /var/lib/jenkins/secrets/initialAdminPassword를 입력후 나오는 값을 복사하여 젠킨스 페이지의 Administrator password 칸에 넣어줍니다



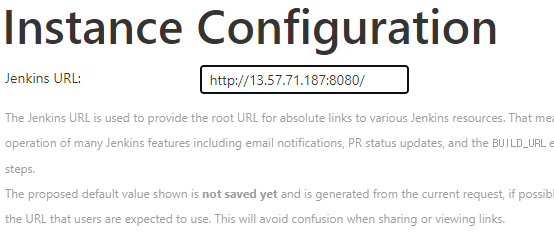
Install suggested plugins 선택

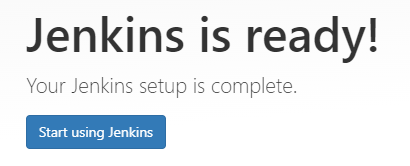


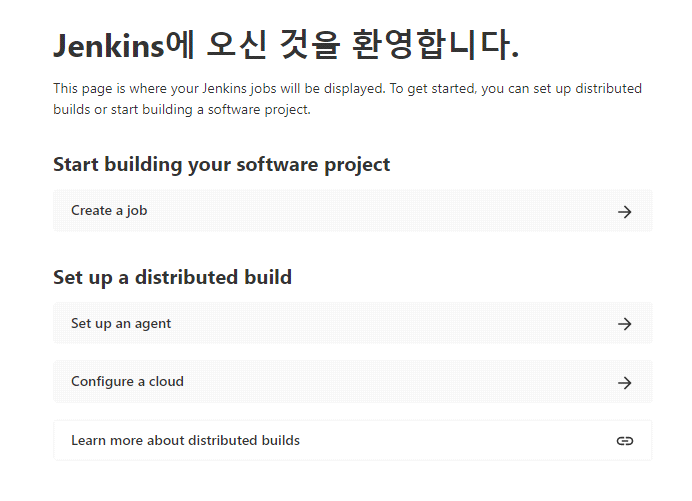
설치후 계정을 생성하는 페이지가 나오는데 작성해줍니다



젠킨스 주소 확인







젠킨스와 Git연동

EC2에 git이 설치되어있는지 먼저 확인(만약 없다면 sudo apt install git)

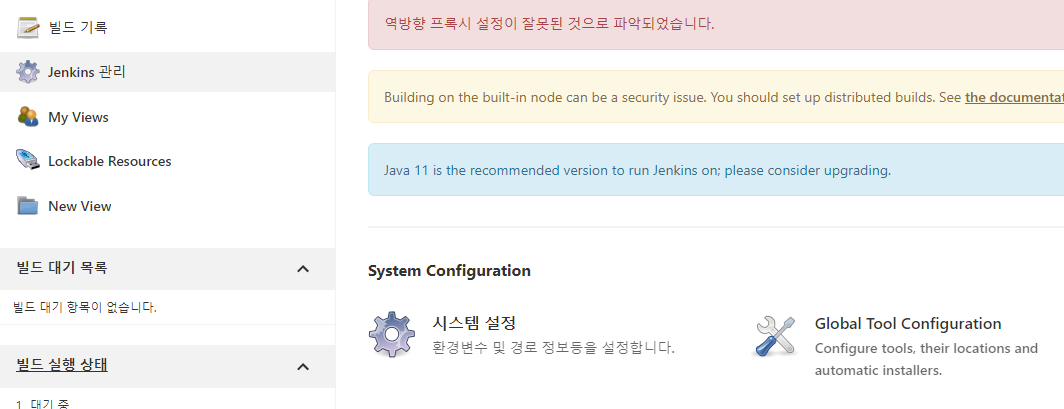
$ sudo git –version



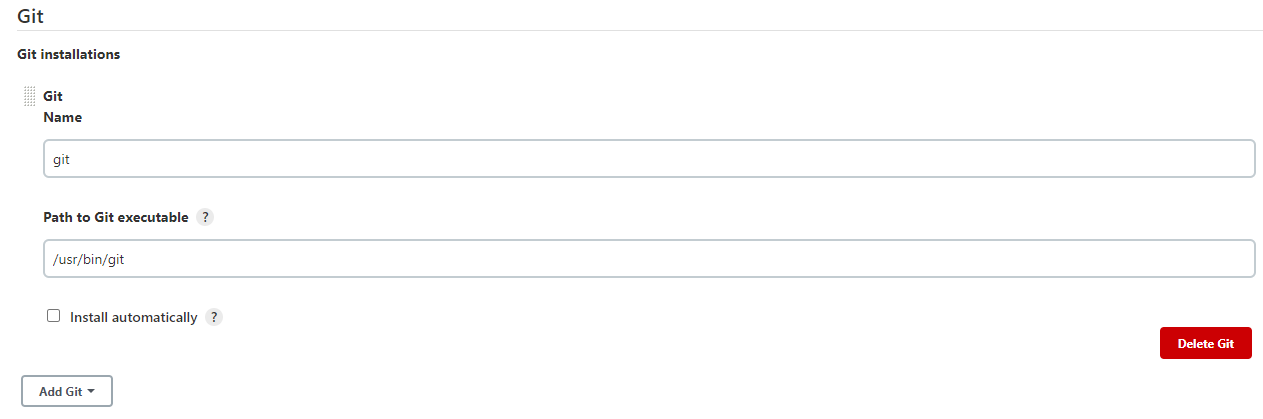
sudo which git으로 git의 설치 경로 확인



젠킨스 관리에서 Global Tool Configuration 클릭



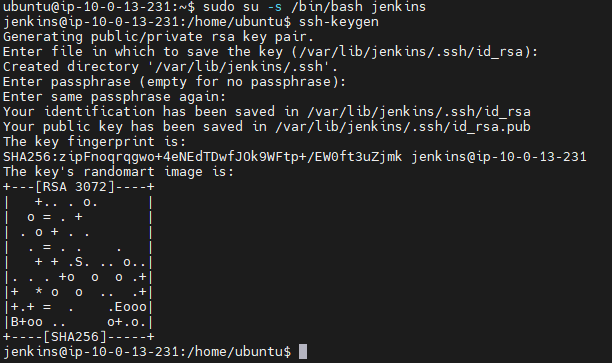
Git의 경로를 확인했던 경로로 수정 후 저장



sudo su 수퍼유저권한으로 젠킨스를 bash형태로 접속하여 keygen 생성

$ sudo su -s /bin/bash Jenkins

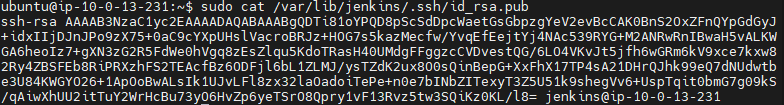
$ ssh-keygen



exit명령어로 jenkins에서 빠져나온 이후 위에서 생성한 퍼블릭 키를 cat 명령어로 조회한다.

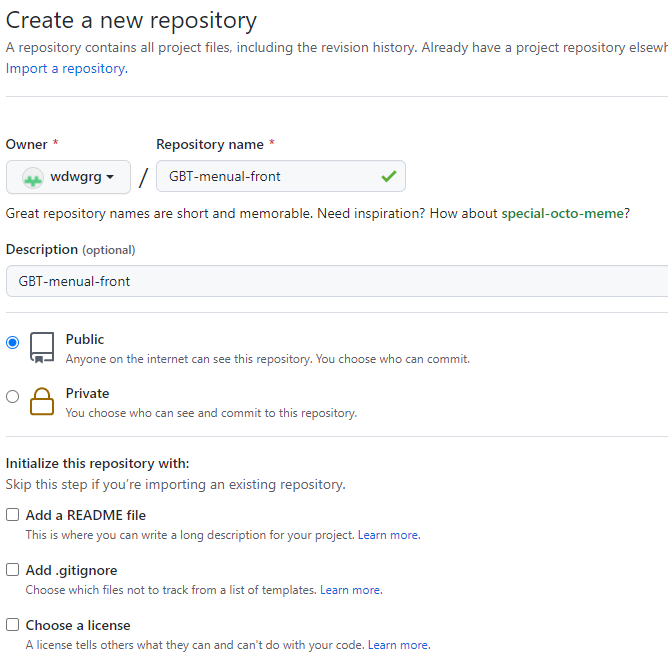
$ sudo cat /var/lib/jenkins/.ssh/id\_rsa.pub



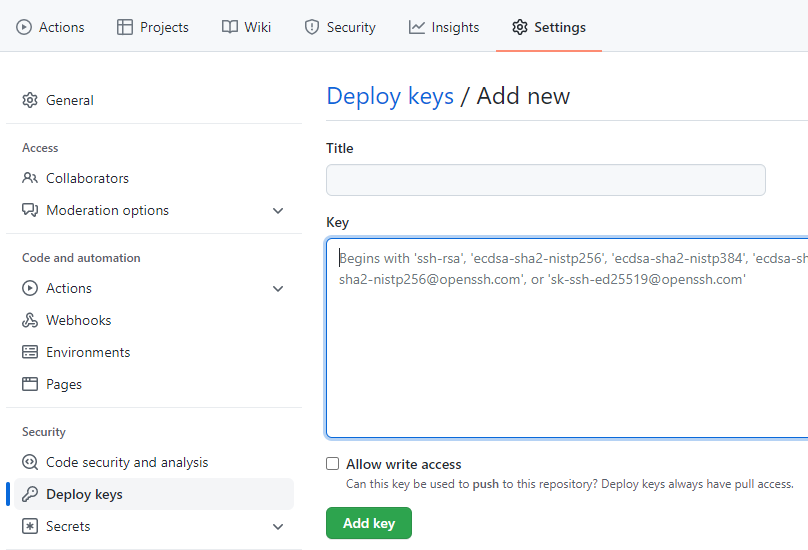


jenkins와 연동할 레포지토리 생성

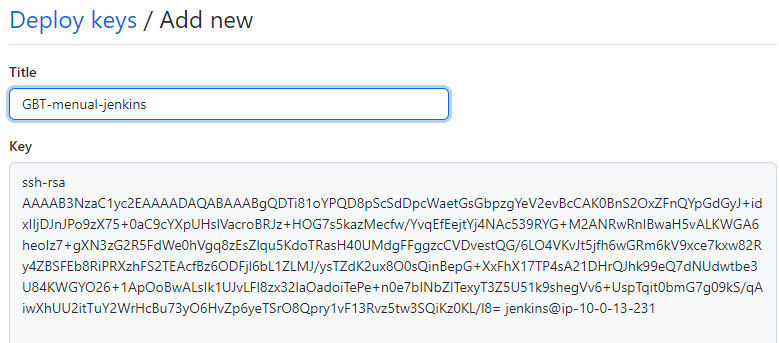
github 에서 젠킨스와 연동할 레포지토리를 생성한다



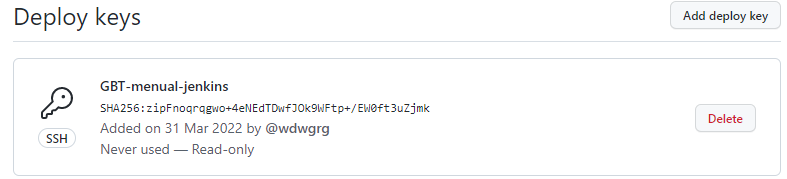
생성한 레포지토리의 설정에서 Deploy keys에서 add deploy key 선택



key 부분에 위에서 cat 명령어로 조회했던 jenkins 키 입력후 add key 버튼을 누른다

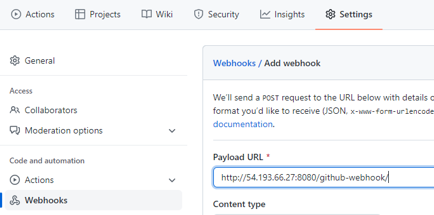
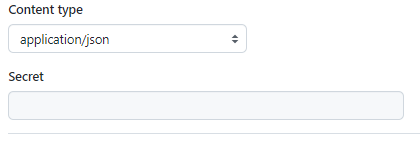


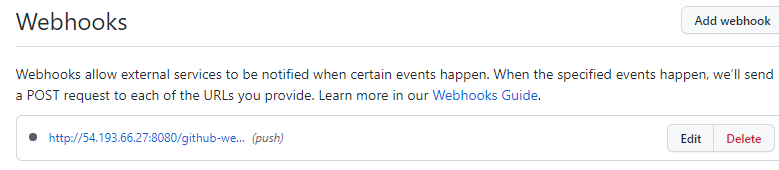
발급된 키를 확인할 수 있다



push를 했을 때 jenkins에서 인식할 수 있도록 webhook 설정을 합니다

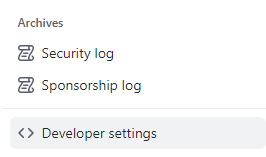
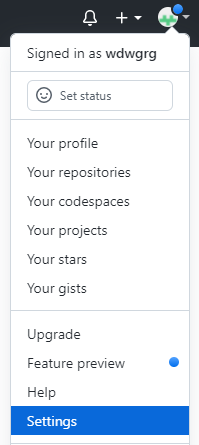
EC2 주소와 젠킨스 포트를 입력하고 github-webhook/ 을 씁니다

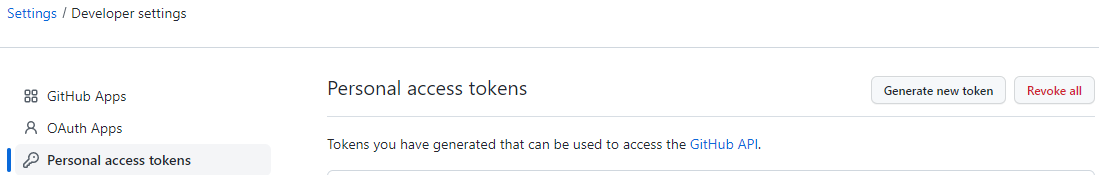
  




jenkins에서 github로 접근할 수 있도록 토큰을 생성합니다

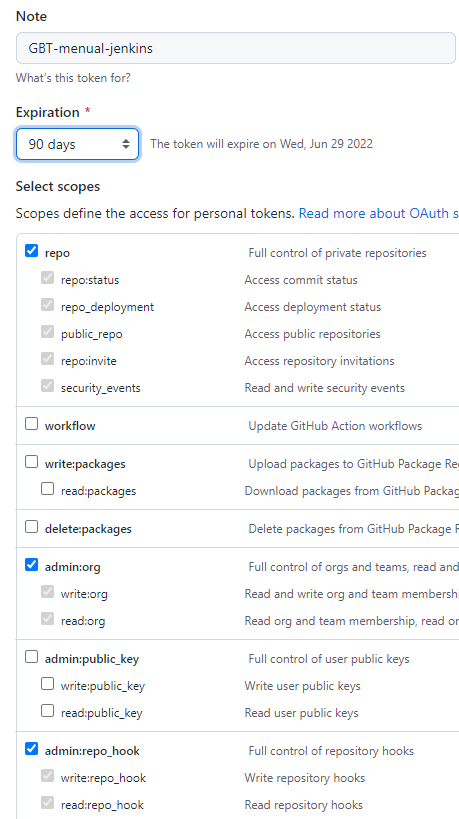
자신의 계정의 setting에 Develop setting에 personal access token 클릭





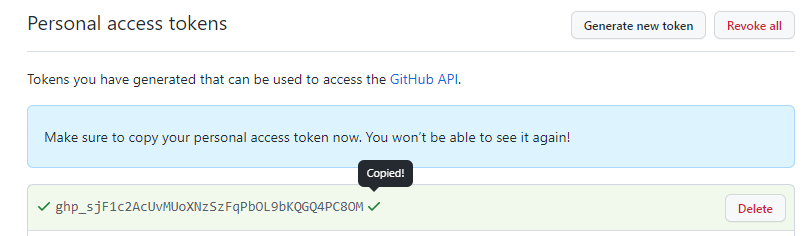
Generate new token 클릭을 한 후

토큰 이름을 설정해주고 아래의 권한을 허용해 줍니다



토큰을 만들면 토큰 키가 나오는데 토큰 키는 지금이 아니면 볼 수 없으므로 다른곳에 저장을 하는 것이 좋습니다

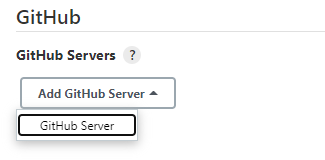
토큰 : ghp\_QtN8EDjNodRI9ceyKHvxksUOSg01dx4JuEtg



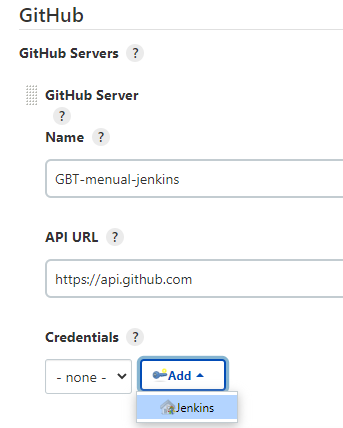
토큰을 발급 받은 후 젠킨스로 돌아와 젠킨스 설정에서 시스템 설정으로 들어갑니다



Github를 찾아서 add github를 해줍니다

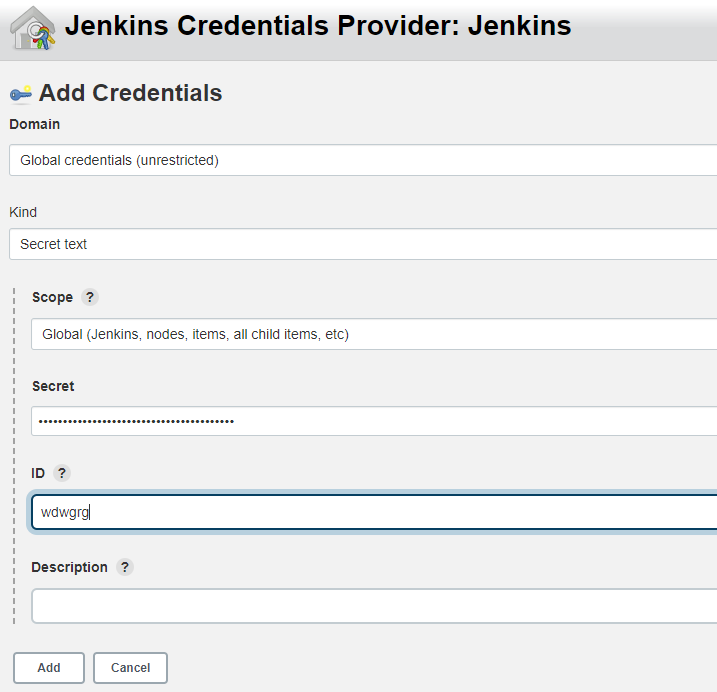


Credential을 추가하는데 Credential 항목에 Add부분에 jenkins클릭합니다



Kind 부분을 secret text로 바꾸고 Secret 부분에는 아까 발급 받은 key값을 넣습니다

ID부분에는 github Id를 적으면됩니다



**Credentials 에서 위에서 만든 Credentials을 선택 후 test connection을 실행**

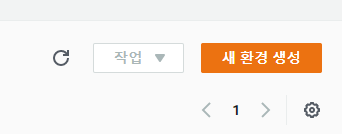
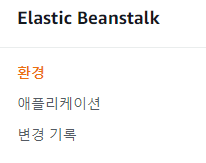
Credentials verified for user YeongSeon1, rate limit: 4999 이렇게 뜨면 성공한것이기 때문에 성공한 것을 확인하고 저장을 해줍니다



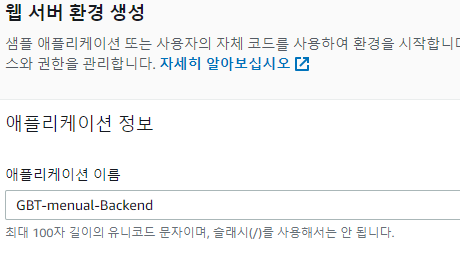
**---Elastic Beanstalk를 이용한 Backend CICD---**

**Elastic Beanstalk은 어플리케이션을 신속하게 배포하고 관리할 수 있고 배포를 하면 자체적으로 로드밸런싱, 오토 스케일링을 하기 때문에 개발하는데 여러 장점이 있습니다**

**엘라스틱 빈스톡에서 새 환경을 만들어 줍니다**

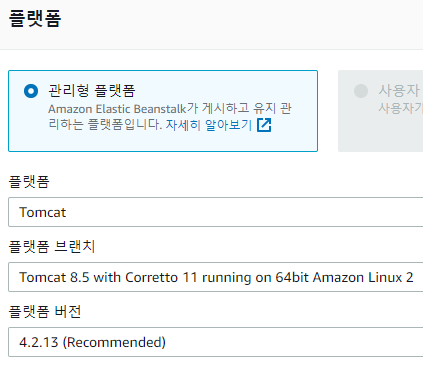
****

**애플리케이션 이름을 설정 합니다**

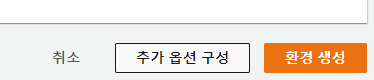
****

****

**플랫폼에는 빌드를 war로 할 것 이기 때문에 Tomcat을 선택해 줍니다**

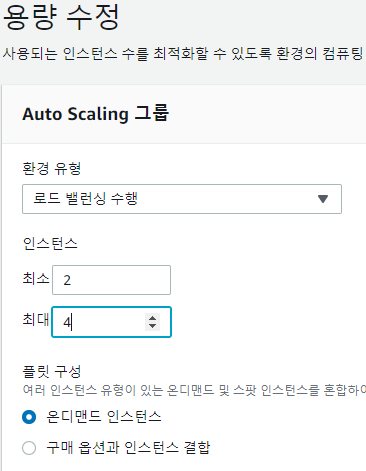
****

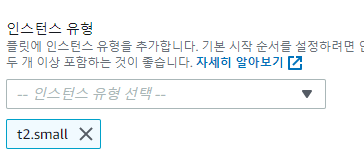
**또한 추가적으로 구성할것이 있기 때문에 추가 옵션 구성을 클릭합니다**

****

**Elastic beanstalk으로 로드 밸런싱을 수행할 것이기 때문에 환경 유형을 로드 밸런싱 수행으로 설정 합니다**

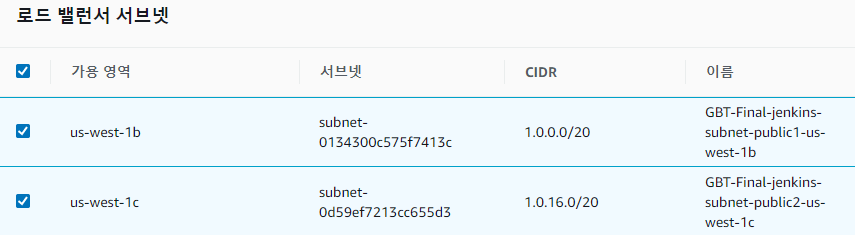
**로드밸런싱을 하려면 인스턴스가 최소 2개가 필요하기 때문에 인스턴스를 최소 2개로 설정하고 이 환경을 만들면 ec2 인스턴스가 자동으로 2개 생길 겁니다**

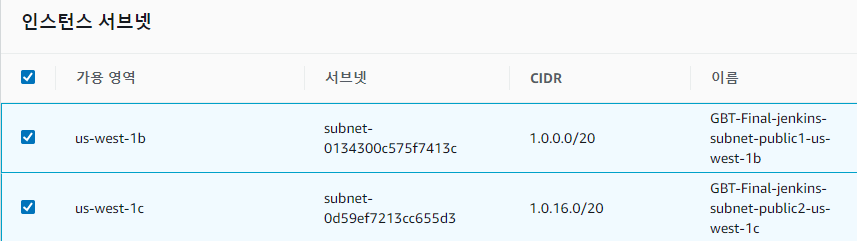
****

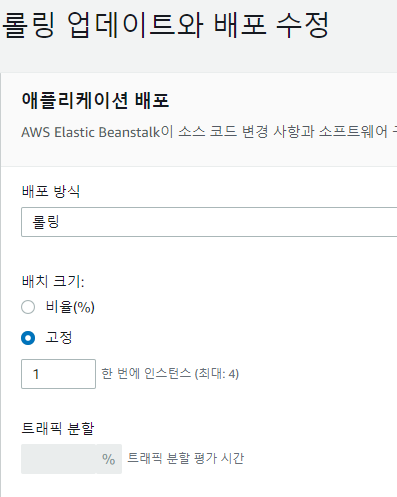
****

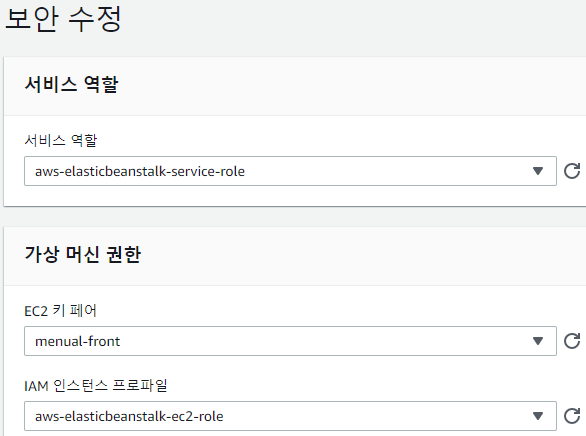
**VPC는 앞에서 만들었던 젠킨스 VPC와는 다른 VPC로 설정합니다**

****

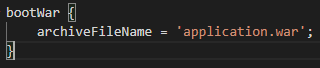
****

****

****

****

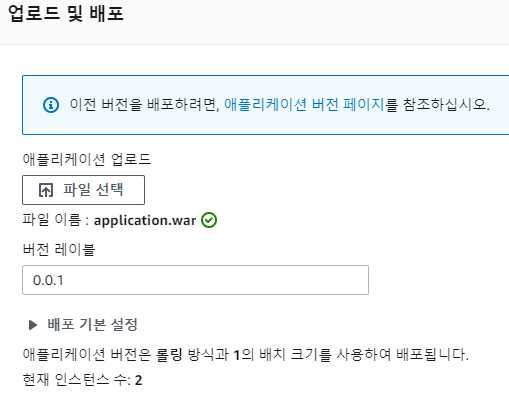
**환경을 구성한후 배포할 backend의 build.gradle에서 war파일 name지정**

****

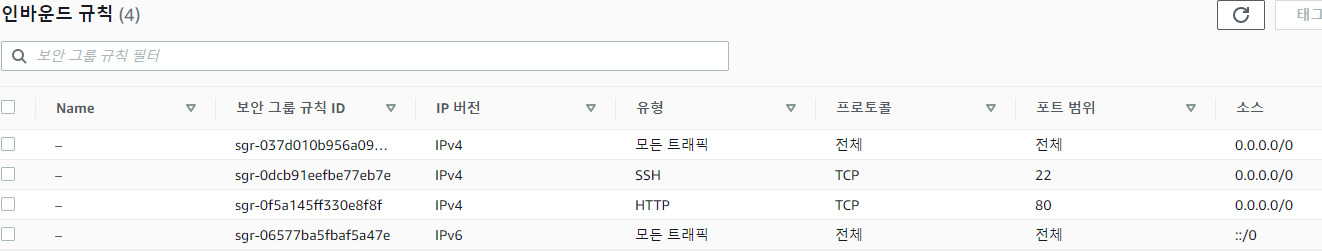
**Elastic beanstalk 에서 업로드 및 배포 클릭**

****

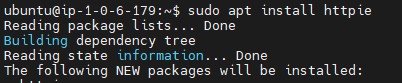
**로컬에서 만든 war 파일 업로드 하여 배포합니다**

****

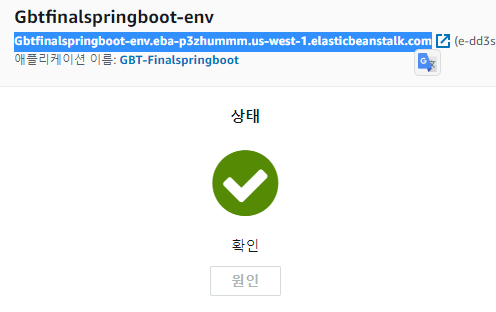
**현재 프로젝트에서 배포한 war파일이 private ec2에 올려져있기 때문에 잘 배포가 되었는지 확인하기 위해선 빈스톡 private ec2에 ssh에서 접속이 가능하도록 보안그룹에 ssh에 대한 인바운드를 준뒤에 ssh에서 rest api로 요청하여 배포를 확인 해야합니다**

****

**Rest api를 사용하기 위해서 httpie를 설치해줍니다**

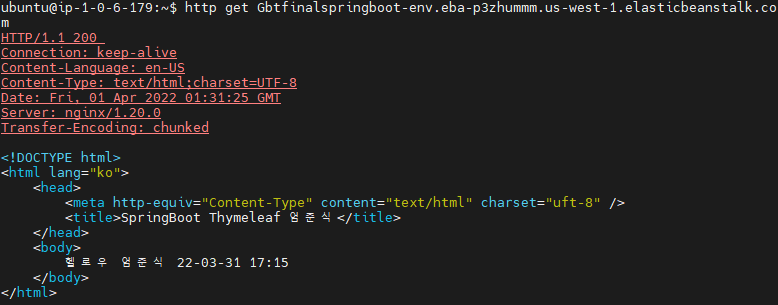
****

**엘라스틱 빈스톡에 표시한 저 부분을 복사한후**

****

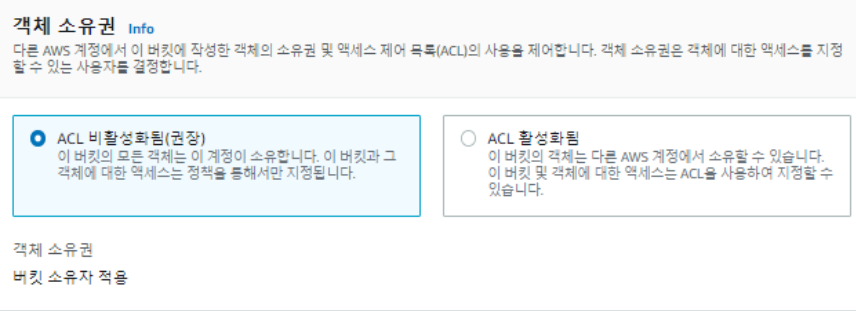
**우분투에서 rest api 요청하는 명령문 뒤에 붙여 줍니다**

**$ sudo http get 표시한주소**

****

**잘 배포되었는지 확인 할 수 있습니다**

**S3 버킷을 생성합니다**

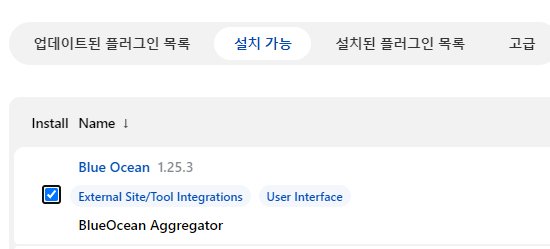
****

****

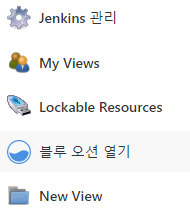
**젠킨스에서 블루오션을 설치 합니다**

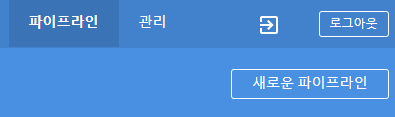
**블루 오션은 기존보다 대시보드가 좀더 시각화 되어있고 파이프 라인이 시각화 되어 있어 문제가 발생시 어느 스테이지 또는 스텝이 문제가 있는지 더 쉽게 알 수 있습니다**

**젠킨스 관리의 플러그인 관리에서 blue ocean을 설치합니다**

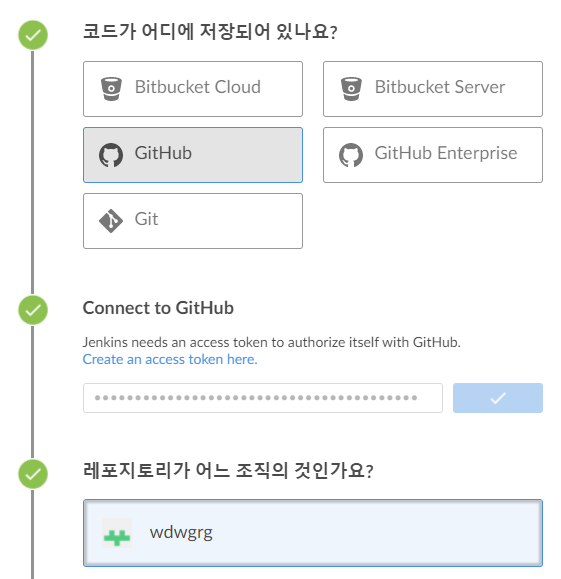
****

**설치후 블루오션으로 들어갑니다**

****

****

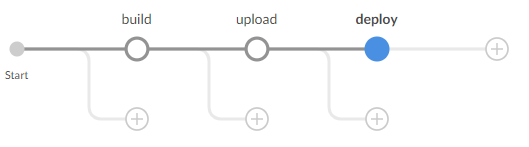
**Connect to Github란에는 이전에 만든 access 토큰값을 넣어 줍니다**

****

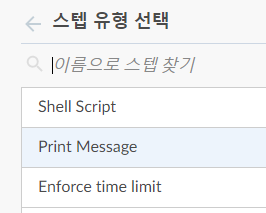
**연결하려는 레포지토리 선택후 파이프 라인 생성**

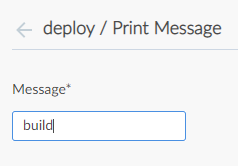
****

**블루 오션의 파이프 라인입니다 ( + 버튼을 눌러 stage를 추가하는 형태입니다)**

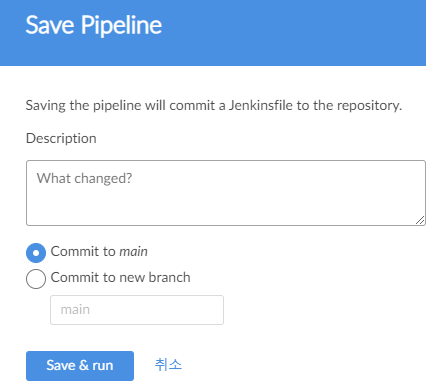
****

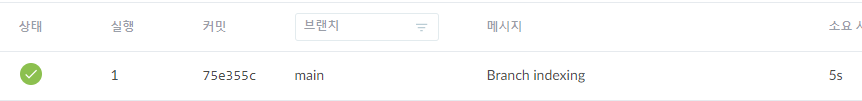
**각 스테이지 별로 이름을 넣고 스텝을 추가합니다 (print message를 추가)**

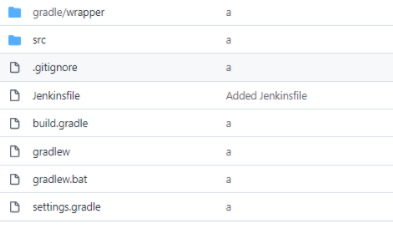
****

****

**설정을 완료하고 저장을 하면 레포지토리에 Jenkinsfile 이 생깁니다**

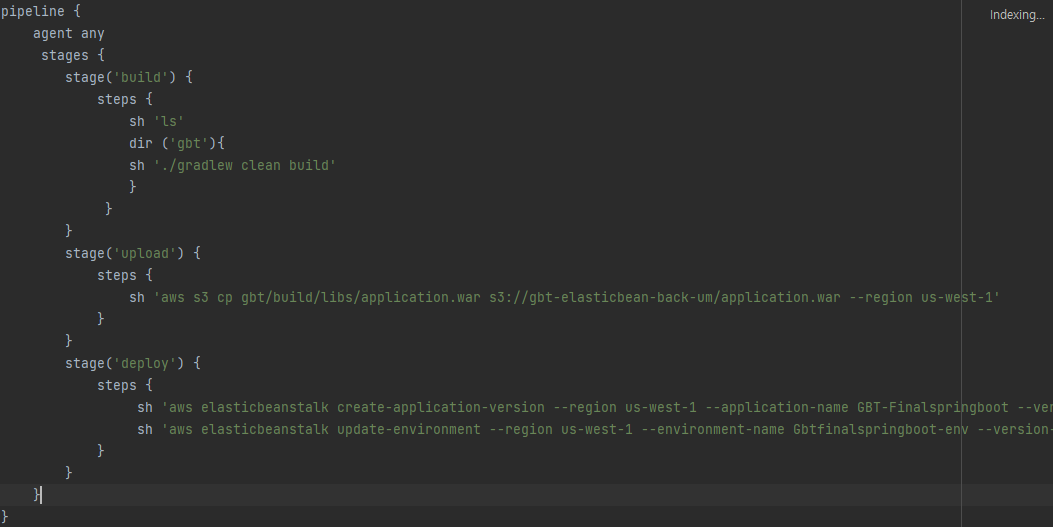
****

****

****

**생성된 jenkins파일을 레포지도리에서 pull해서 가져온 후**

**jenkins파일에서 파이프 라인을 작성하면 됩니다**

****

**--build stage 설명--**

**gradle wapper를 이용하여 build 후 war 파일을 생성합니다.**

**./gradlew clean build**

**--upload stage 설명--**

**AWS CLI를 이용하여 Jenkins 서버에 생성된 war 파일을 S3로 복사합니다.**

**aws s3 cp gbt/build/libs/application.war s3://gbt-elasticbean-back-um/application.war --region us-west-1**

**build/lib/application.war**

**gradle로 build 후 생성되는 war 파일의 위치를 지정합니다.**

**저는 application.war로 생성되도록 설정하였기 때문에 위와 같이 지정하였습니다.**

**s3://gbt-elasticbean-back-um/application.war**

**생성 한 S3의 버킷이름(gbt-elasticbean-back-um)과 복사될 파일명을 입력합니다.**

**--region us-west-1**

**Elastic Beanstalk이 등록된 region을 입력합니다.**

**--deploy stage 설명—**

**deploy은 두 개의 step으로 나눠서 진행합니다.**

**1) create-application-version (Elastic Beanstalk에 새로운 애플리케이션 버전을 등록합니다.)**

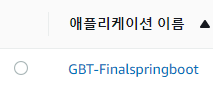
**aws elasticbeanstalk create-application-version --region us-west-1 --application-name GBT-Finalspringboot --version-label ${BUILD\_TAG} --source-bundle S3Bucket="gbt-elasticbean-back-um",S3Key="application.war**

**--region us-west-1**

**Elastic Beanstalk이 등록된 region을 입력합니다.**

**--application-name GBT-Finalspringboot**

**Elastic Beanstalk에 등록 된 애플리케이션명을 입력합니다**

****

**--source-bundle S3Bucket="gbt-elasticbean-back-um",S3Key="application.war**

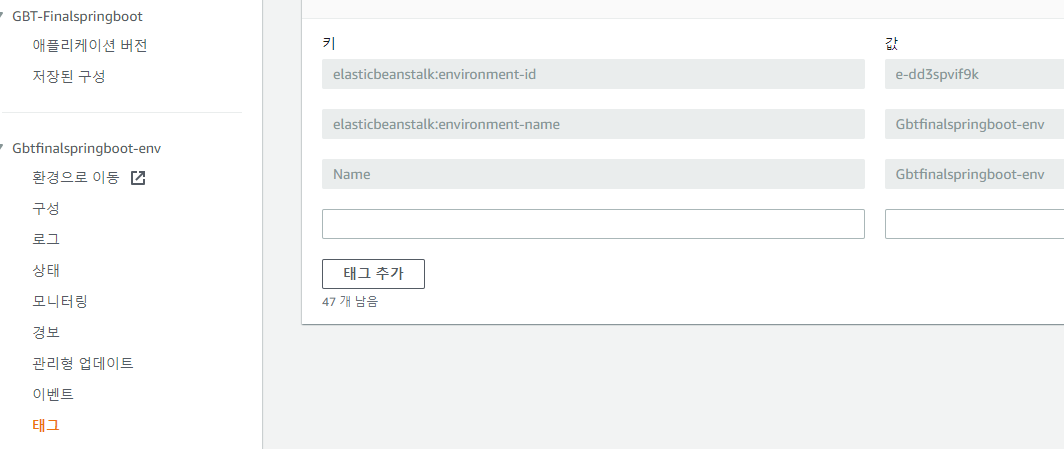
**버전에 등록할 S3 버킷 이름과 파일명을 입력합니다.**

**2) update-environment (새로 등록된 버전으로 배포를 시작합니다.)**

**aws elasticbeanstalk update-environment --region us-west-1 --environment-name Gbtfinalspringboot-env --version-label ${BUILD\_TAG}**

**--environment-name Gbtfinalspringboot-env**

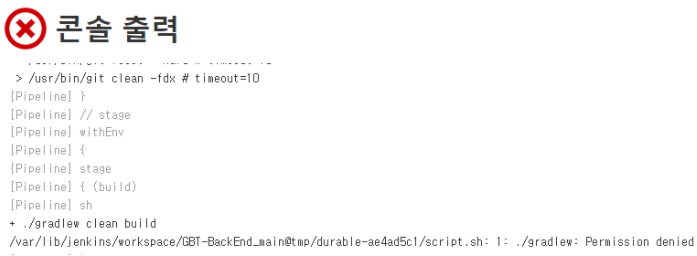
**Elastic Beanstalk 환경 이름을 입력합니다(엘라스틱 빈스톡의 어플리케이션 테그항목에서 확인 가능합니다)**

****

**여기까지가 Backend 파이프 라인에대한 설명입니다**

**---gradlew를 사용하여 build 시 발생하는 permission 오류 ---**

**파이프 라인 생성후 gradlew 를 사용하여 build 시 permission deny 오류가 뜰 수 있습니다**

****

**이유는 window 환경에서 작업해서 소스를 push 했기 때문에 window 환경에서 파일 생성시 기본권한이 644로 생성되어 있기 때문입니다**

**해결 방법 : 일단 build 하여 war파일을 생성하면 workspace가 생기는데**

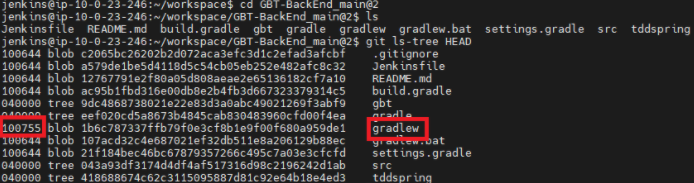
**우분투에서 jenkins의 workspace까지 들어갑니다**

****

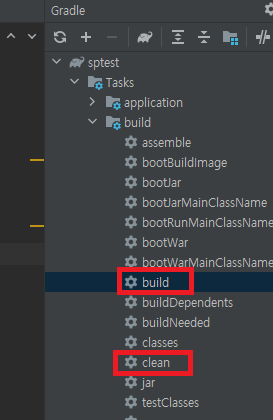
**그다음 ls로 해당하는 파이프 라인 이름을 확인한후**

**해당하는 파이프라인에 들어가서 ls로 gradlew 권한 확인(아래의 사진은 권한설정한 후의 사진이라 10755로 나타나 있지만 고치기전에는 10644로 되어있습니다)**

**$ git ls-tree HEAD (권한 확인)**

****

**우선 해당파일을 clean build를 하고 ( 1 - clean, 2 - build)**

****

**권한을 변경 후 커밋,푸쉬를 해준뒤에 다시 권한을 확인합니다**

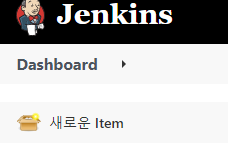
**$ git update-index --add --chmod=+x gradlew (권한 변경 코드)**

**권한이 644에서 755로 바뀐 것을 확인할 수 있습니다**

****

**--프론트 CICD--**

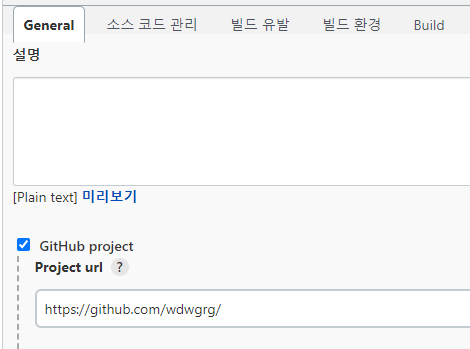
**Credentials 설정 후 새로운 Item 선택**



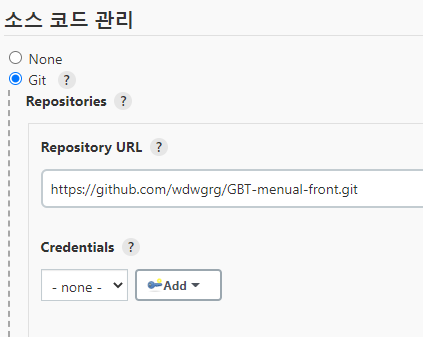
이름을 입력하고 Freestyle project 선택



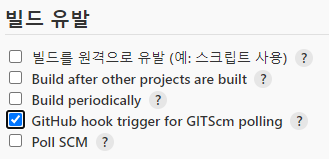
Github project를 선택후 github주소와 자신의 닉네임을 입력하고 끝에 “ / “ 를 입력합시다



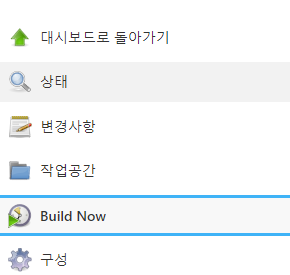
소스 코드에는 git을 선택한후 연결할 레포지토리의 주소를 입력합니다



webhook에서 이벤트가 발생하면 빌드를 할것이기 때문에 빌드 유발에서 GitHub hook 체크하고 저장을 합니다

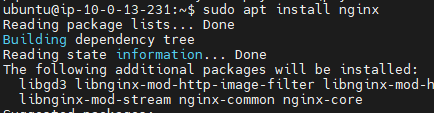


저장한후 build now 를 눌러 workspace가 만들어 지도록 합니다



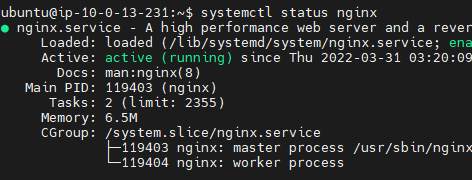
Ec2에 Nginx 설치

$ sudo apt install nginx



Nginx 설치후 상태 확인

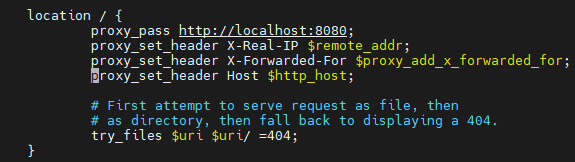
$ systemctl status nginx



경로 추가

$sudo nano /etc/nginx/sites-available/default



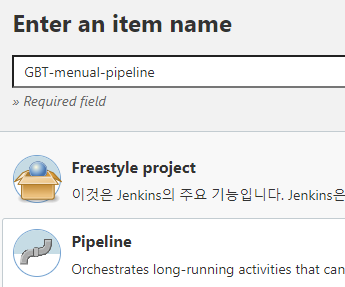


설정을 다했으면 nginx를 재시작 해줍니다

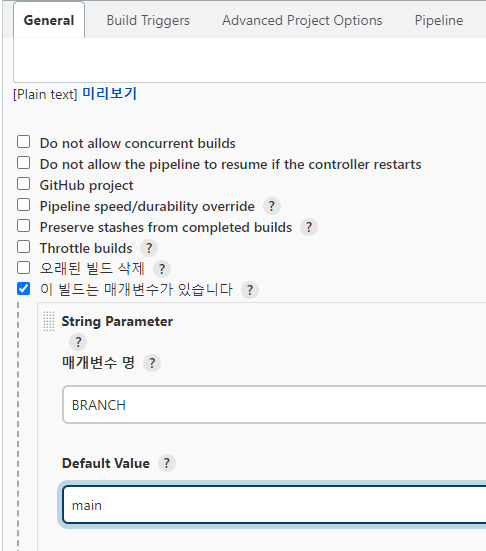
$ sudo systemctl restart nginx

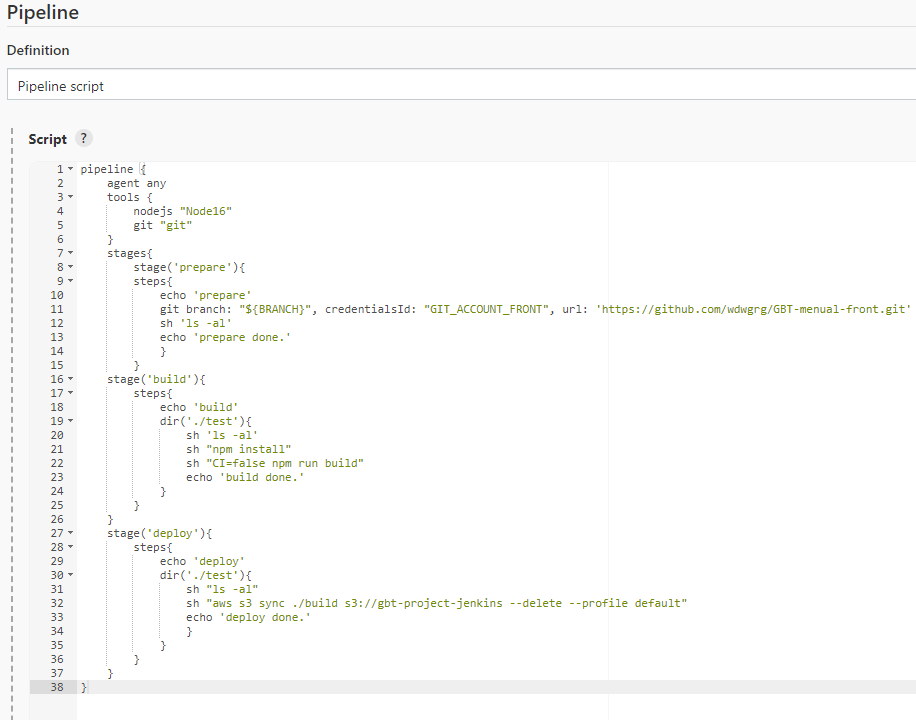


다음은 젠킨스에서 파이프 라인을 생성 합니다



매개변수를 체크하고 String Parameter를 선택한뒤 매개변수 명과 Default Value에는 branch이름을 적어줍니다





S3를 생성 합니다

먼저 버킷 엑세스 차단 설정에 차단을 풀어줍니다



정적 웹 사이트 호스팅을 활성해주고

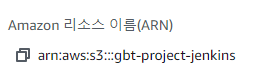
인덱스 문서와 오류 문서를 둘다 index.html로 설정해줍니다



S3를 생성한후 권한의 버킷 정책에 이렇게 입력합니다

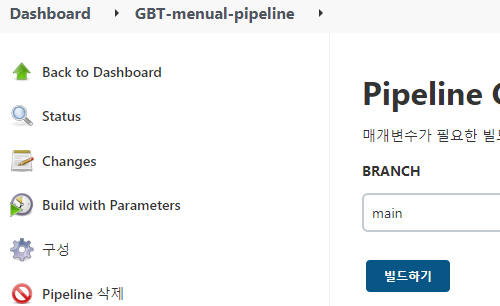


S3 속성에 Amazon 리소스 이름이 있는데 이부분을 복사한후 pipeline에 수정합니다

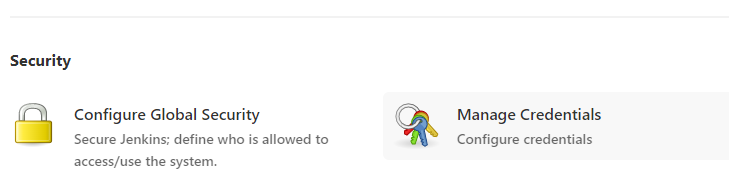


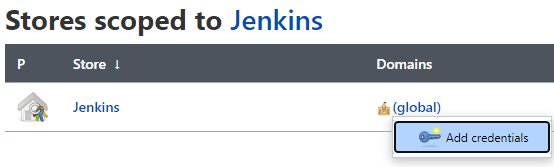


파이프 라인 실행

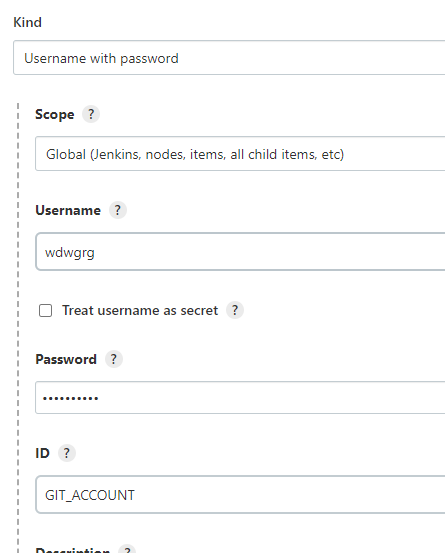


실행후 Jenkins 관리에 Manage Credentials클릭

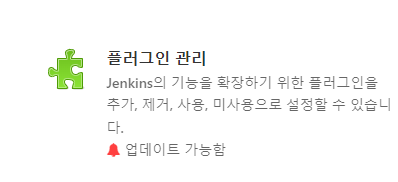


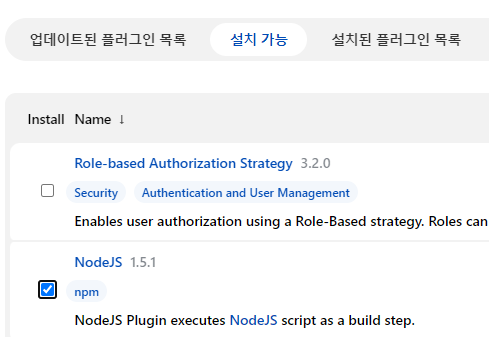


username에는 깃허브 아이디 password에는 깃허브 비밀번호, ID에는 GIT\_COUNT를 넣는다

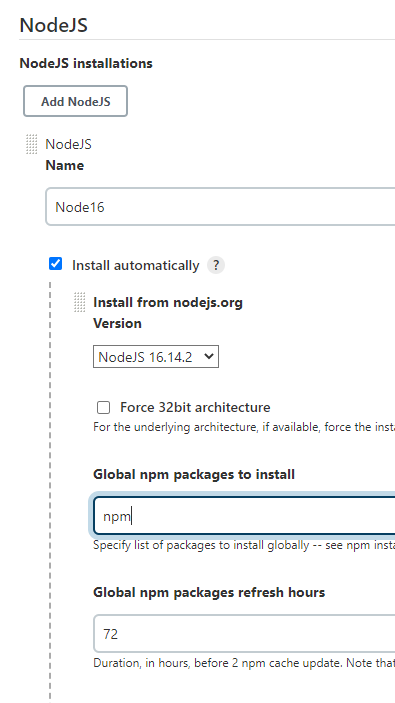


jenkins관리에 플러그 인에서 node검색후 설치





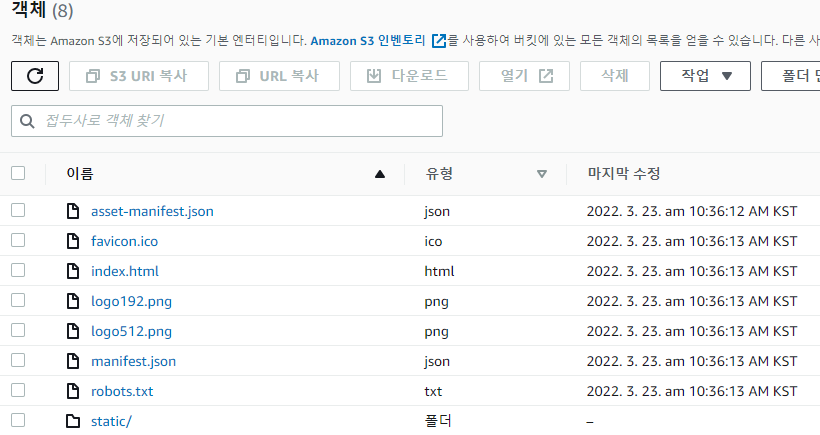
설치후 Global Tool Configuration에서 Add NodeJS 클릭



Name칸에 입력한 Node16 기억할것

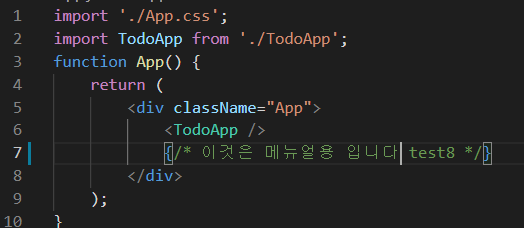
S3에 파일 추가

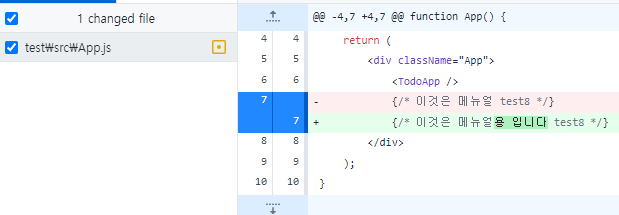
로컬의 build폴더 안에 있는 파일 및 폴더를 넣어준다

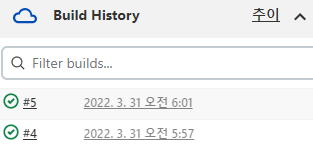


그런 다음 올린파일에 해당하는 것을 test용으로 주석을 달고 github에 commit, push를 하면

젠킨스가 github의 웹훅을 감지 하여 파이프라인으로 인해 자동빌드가되고 S3에 다시 배포하게됩니다







엘라스틱 빈스톡을 이용한 Backend 구축