초격차 패키지 Online.

안녕하세요
백엔드 개발자를 위한
DevOps 입문 강의를 진행할
이성미, 양재현입니다.

Chap1 | 백엔드 아키텍처 맛보기 (멀티모듈, MSA, DDD)

Chap2 | 백엔드 개발자를 위한 클라우드 인프라 운영 - Amazon

Chap3 | 백엔드 개발자를 위한 컨테이너 서비스 - Docker

Chap4 | 클라우드 기반 컨테이너 서비스 운영- Amazon ECR, ECS, Fargate

Chap5 | 백엔드 개발자를 위한 빌드 자동화 - Jenkins

Chap6 | MVP 프로젝트: 채용사이트 구축하기

백엔드 개발자를 위한 빌드 자동화- Jenkins

1 Git을 이용한 소스코드 버전관리
2 Git Brunch를 통한 협업 개발하기
3 Jenkins 프로그램 소개 및 설치하기
4 Git 연동을 위한 Jenkins 환경설정
5 Jenkins 파이프라인을 이용한 컨테이너 빌드
6 Jenkins Slack Webhook 연동하기
7 실전 빌드하기
source push → container build → ecr에 업로드
8 LAB: CI 구축하기
git 코드 업로드 부터 컨테이너 빌드까지

백엔드 개발자를 위한 빌드 자동화- Jenkins

1 Git을 이용한 소스코드 버전관리

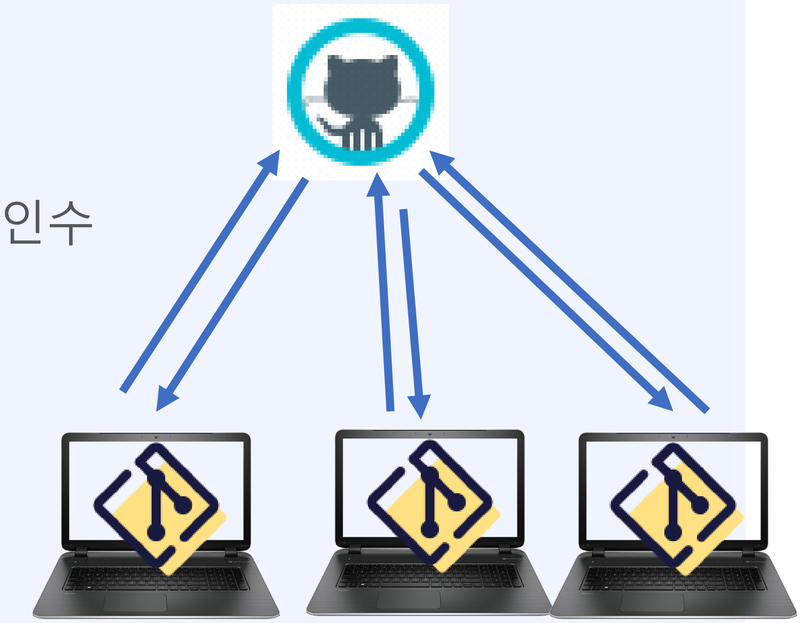
Git을 이용한 소스코드 버전관리

학습내용

- Git과 GitHub
- Git 설치
- Git 기본 명령어
- Git 브랜치로 버전관리

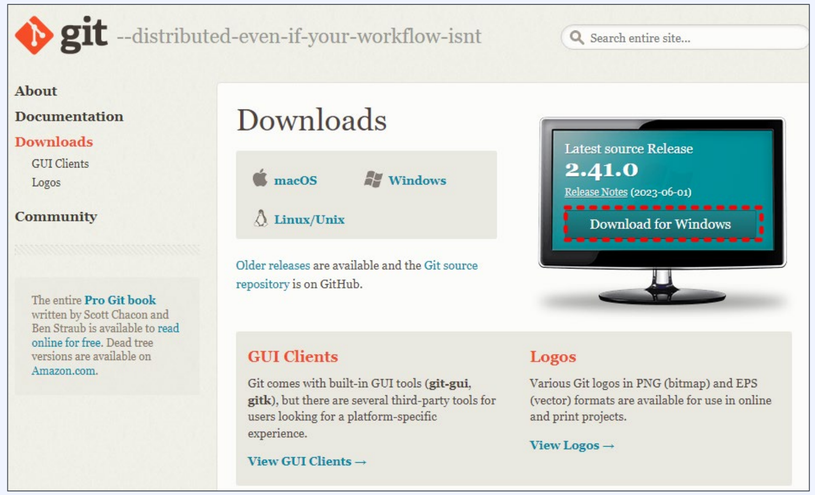
Git 과 GitHub

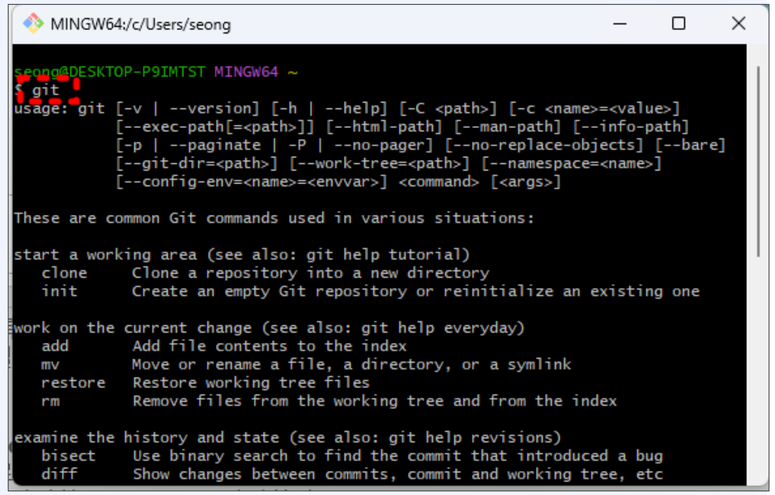
- **Git?**
  - 2005년 리누스 토발즈가 리눅스 커널 프로젝트 개발을 위해 만든 분산 버전관리 툴
  - 코드의 원본이나 변경 내역을 저장하는 형상관리(Configuration Management) 툴
  - 버전관리
    소프트웨어 개발과정에서 코드의 추가 변경과정을 모두 기록하여
    특정 시점으로 돌아거가나 문제가 생긴 파일을 복원하는 과정
- **GitHub**
  - Git 을 관리해주는 웹 호스팅 서비스로 2008년 설립. 2018년도에 MS 인수
  - 분산 버전 제어, 액세스 제어, 소스 코드 관리,
    버그 추적, 기능 요청 및 작업 관리를 제공
  - 개발자들 사이의 거대한 커뮤니티를 형성
- **참고 도서**
  - Pro Git(https://git-scm.com/book/ko/v2)
  - 알잘딱깔센GitHub



Git 설치

- **Windows**
  - https://git-scm.com/downloads -> Diownload for Windows -> latest 64bit
- **Mac**
  - 기본으로 설치되어 있음
- **Linux**
  - Ubuntu
    - apt update
    - apt install -y git
  - Redhat
    - yum install -y git





Git 사용자 정보 설정

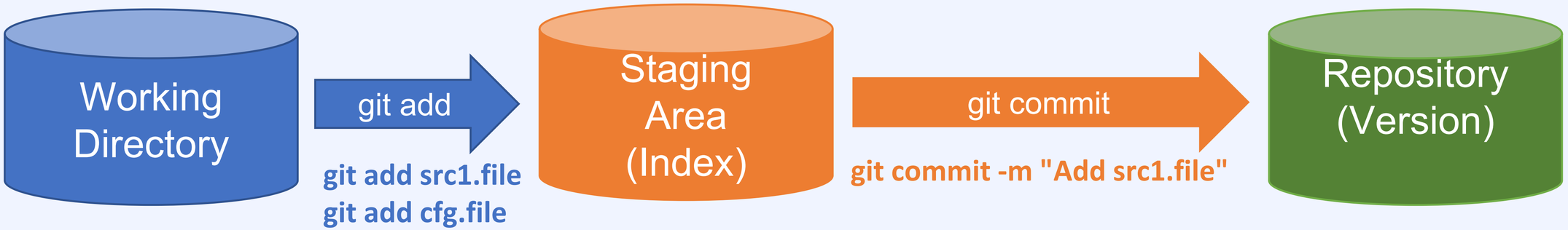
- git 은 누가 커밋했는지 확인하기 위해 사용자정보(email 주소와 이름)을 설정

- 설정파일의 위치
  - git 기본 환경 구성(/etc/gitconfig) ③
  - 유저별 설정 : HOME_DIR/.gitconfig ②
  - 작업 디렉토리별 구성 : /소스코드디렉토리/.git/config ①

- 유저별 설정

```
git config --global user.name "237summit"
git config --global user.email "seongmi.lee@gmail.com"
cat ~/.gitconfig
```

Git의 3가지 작업공간



git add src1.file
git add cfg.file

git commit -m "Add src1.file"

Git으로 관리하도록 지정된
작업 디렉토리

Repository에 반영될 수
있는 파일들의 공간

변경 이력들이 저장되는
디렉토리(버전이력저장)

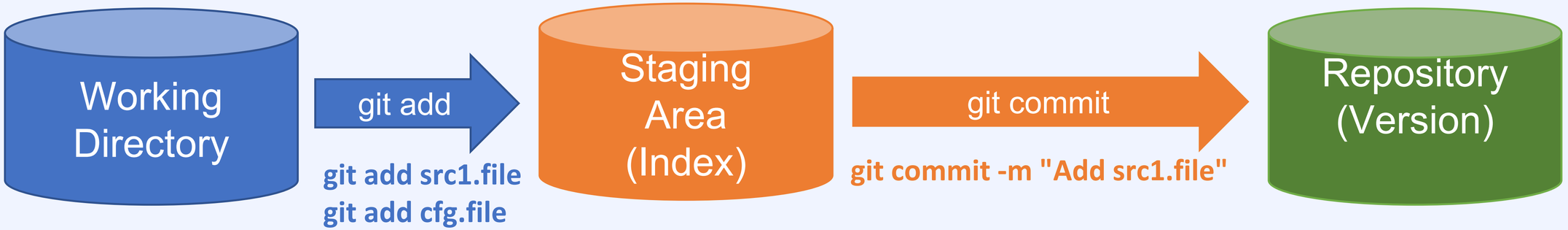
```
echo "code1" > src1.file
echo "data" > data.file
echo "config" > cfg.file
```

src1.file
cfg.file

src1.file

Git 기본 명령어

- 레포지토리 초기화: **git init**
- Git Repository나 Staging Area에 추가되지 않아야 하는 파일 정의: **.gitignore**
- Working directory에 있는 파일을 Staging Area로 옮기는 명령 **: git add**
- Git repository의 브랜치, 추적중인 파일과 변경된 파일 등의 현재 상태 확인: **git status**
- Staging Area에 있는 파일을 Repository 에 저장(변경 내용 기록) : **git commit**
- Repository의 commit된 로그 확인: **git log**
- 워킹 디렉토리, 스테이지 영역, 리포지토리 사이의 변경사항 확인 **: git diff**



Git으로 관리하도록 지정된 작업 디렉토리

Repository에 반영될 수 있는 파일들의 공간

변경 이력들이 저장되는 디렉토리(버전이력저장)

Git 브랜치 관리 명령어

브랜치(Branch)란 나뭇가지(brabch)의 의미로, 소프트웨어 개발시 원래 코드와 상관없이 독립적으로 개발해야할 경우 발생하는데 이때 브랜치를 사용하여 가지를 만들어 독립적으로 개발하거나 병합

- 브랜치 리스트 확인(현재 사용중인 브랜치 : *master) : **git branch**
- 브랜치 생성 : **git branch branch_name**
- 브랜치 삭제 : **git branch -d branch_name**
- 브랜치 이름 변경: **git branch -m branch_name**
- 브랜치 바꾸기: **git checkout 브랜치이름**
- 브랜치들의 커밋 상태 보기: **git log --oneline --decorate --graph --all**
- 브랜치 병합: **git merge other_branch**

우리가 배운 내용

- git 설치후 git-bash 실행
- git 기본 명령어 사용
- git 브랜치 관리 명령어 사용
- 개발 환경에서의 브랜치

백엔드 개발자를 위한 빌드 자동화- Jenkins

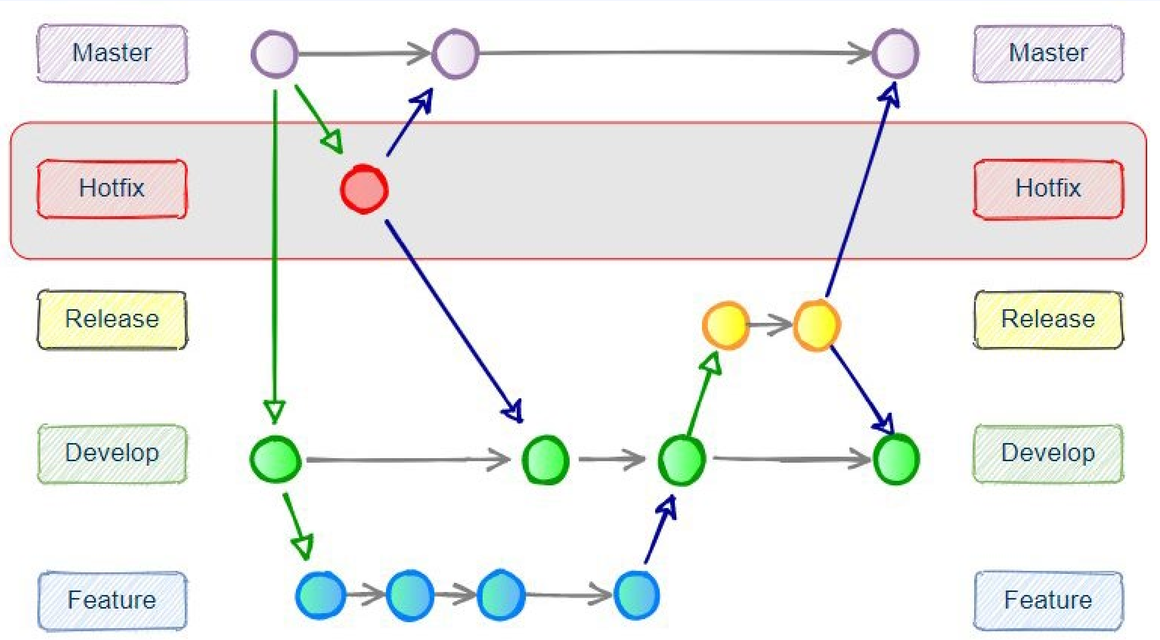
2 Git Brunch를 통한 협업 개발하기

학습내용

- Git 브랜치 Model
- GitHub 가입
- GitHub Repository 관리

Git 브랜치 Model

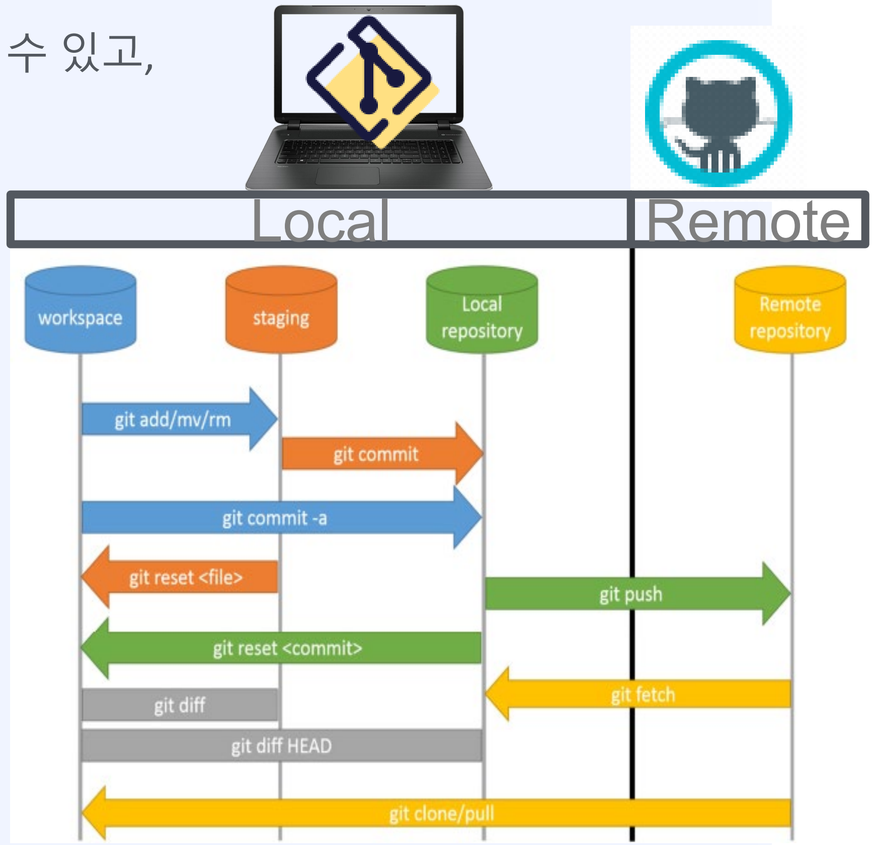
- 운용 모델에서는 크게 나눠 4가지 종류의 브랜치를 이용하여 개발을 진행
  - 메인 브랜치(Main branch)
    - **master** : 배포 가능한 상태만을 관리.
    - **develop** : 통합 브랜치의 역할을 하며, 평소에는 이 브랜치에서 개발을 진행
  - 피처 브랜치(Feature branch)
    - 새로운 기능 개발 및 버그 수정이 필요할 때에 'develop' 브랜치로부터 분기
  - 릴리스 브랜치(Release branch)
    - 버그를 수정하거나 새로운 기능을 포함한 상태로 모든 기능이 정상적으로 동작하는지 확인
  - 핫픽스 브랜치(Hotfix branch)
    - 배포한 버전에 긴급하게 수정을 해야 할 필요가 있을 경우, 'master' 브랜치에서 분기하는 브랜치



https://www.theserverside.com/blog/Coffee-Talk-Java-News-Stories-and-Opinions/GitFlow-Hotfix-Branch-Example-Start-Finish

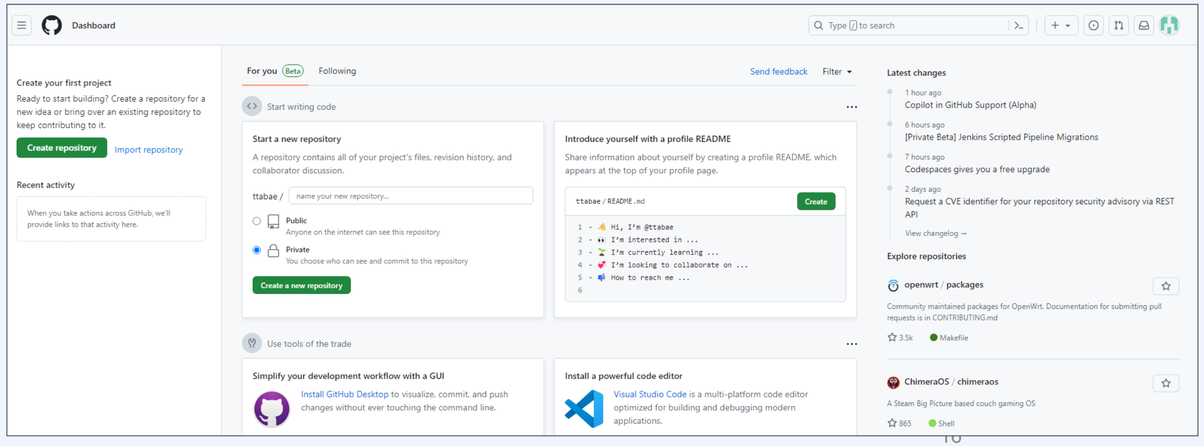
GitHub

- Git 저장소를 관리해주는 웹 호스팅 서비스
- 한 프로젝트에 여러 명의 사람이 참여하여 버전 제어 및 공동 작업이 가능
- 자신의 프로젝트는 물론 다른 개발자들의 인기 있는 프로젝트의 코드를 볼 수 있고,
  이슈(issue)를 제기할 수 있으며
  원한다면 다른 이의 프로젝트를 수정하고 발전시키는 데에 참여
- 개발자들 사이의 거대한 커뮤니티를 형성
- 여러 명의 사람이 참여하여 버전 제어 및 공동 작업이 가능



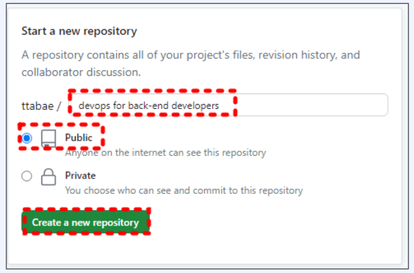
GitHub 계정 만들기(github.com)

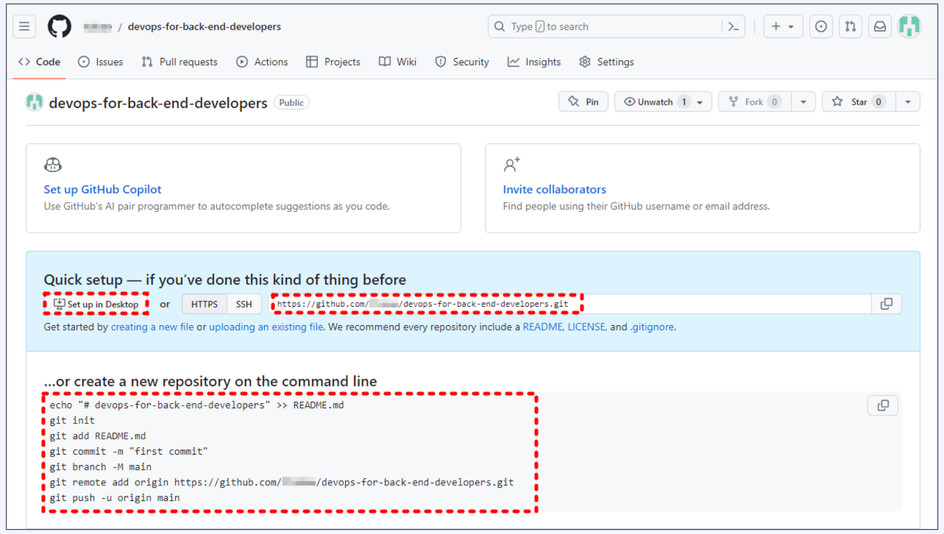
- github.com -> Sign up ->
- email 주소, 패스워드 입력 후 [Continue]
- 계정(username)이름(fastcampusdeveloper) 입력 후 [Continue]
- 제품 업데이트 정보 메일통해 받을지? <n>
- 계정 검증 후[제출하십시요] 클릭(2회)
- [Create Account] 버튼 클릭 -> Email 로 전달된 코드 입력
- 설문조사 - 프로젝트 도구 추천해줌
- 이용금액 안내 [Continue for free]
- Git Hub Dashboard 화면이 표시되고 Repository 생성가능



GitHub 신규 Repository 만들기

- Start a new repository





- GitHub Desktop 다운로드 후 설치
- 로그인 토큰 만들기
  - 오른쪽 계정 > Settings
  - [Developer settings] - [Personal access tokens]
    Tokens(classic) > [Generate new token]
    - NOTE : 토큰 설명
    - Expiration : 토큰 만료일
    - Select scopes : 토큰 적용 범위 선택
      > repo - command line에서 repository access
      > user - command line 에서 계정 정보 수정
    [Generate token]
  - 토큰 생성되면 복사해서 저장.

READMI.md

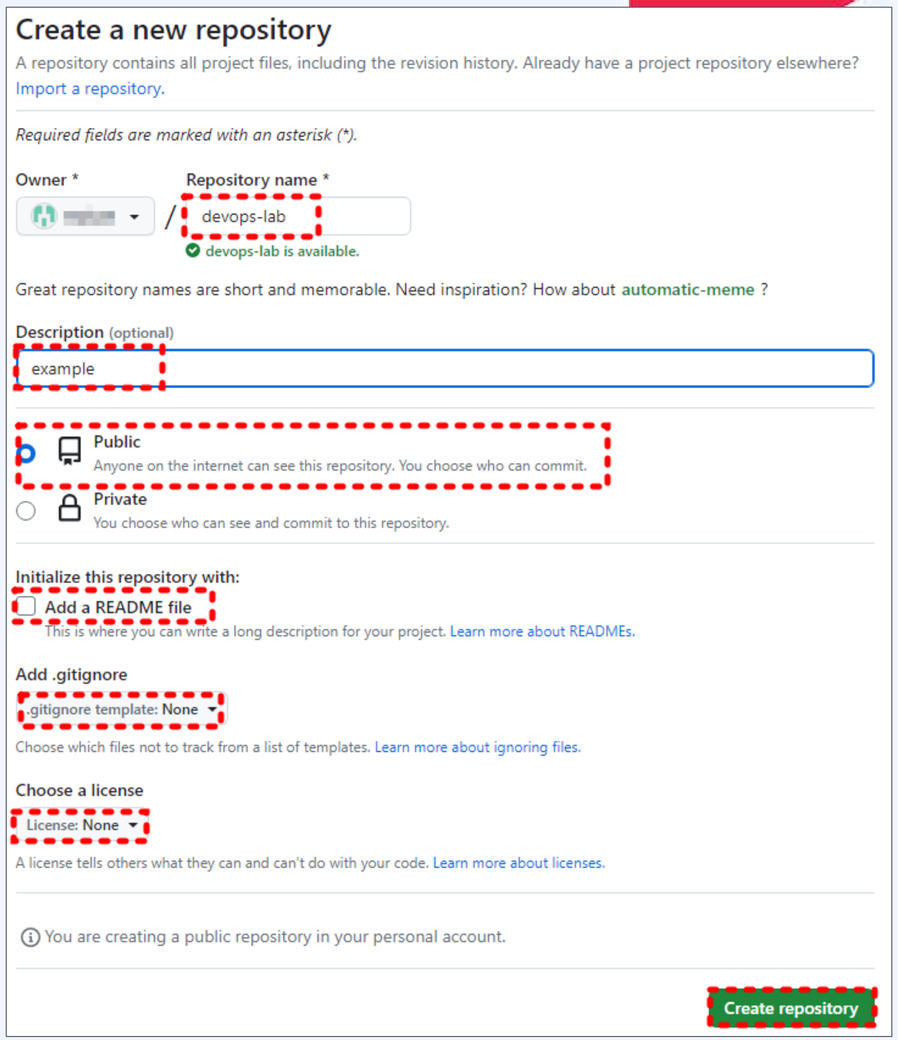
- Git Repository의 최상위 페이지로 Markdown 문법으로 작성
- 치트시트 : https://www.markdownguide.org/cheat-sheet/

GitHub 관리 명령어

- remote repository를 로컬 저장소와 연결
  git remote add [alias-name] [github_url]

- 마스터 브랜치 업로드
  git push -u [alias-name] branch

- GitHub Repository master 브랜치에서 다운로드
  git clone [github_url]



우리가 배운 내용

- 개발 환경에서의 브랜치 모델
- GitHub 계정 등록
- GitHub 레파지토리 연결 및 사용

백엔드 개발자를 위한 빌드 자동화- Jenkins

3 Jenkins 프로그램 소개 및 설치하기

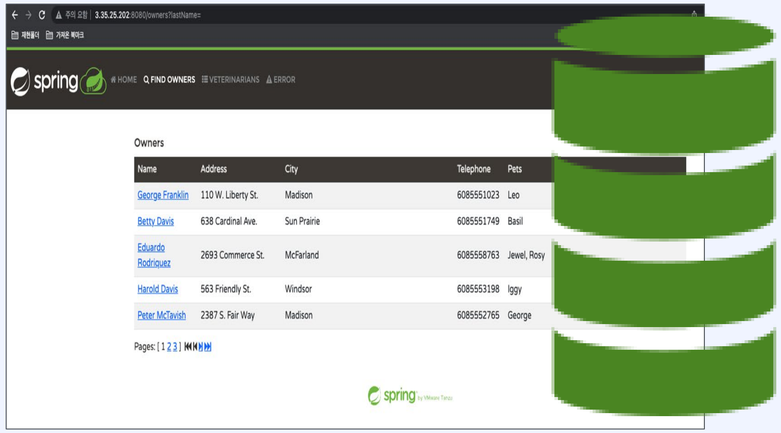
Jenkins 프로그램 소개 및 설치하기

학습 내용

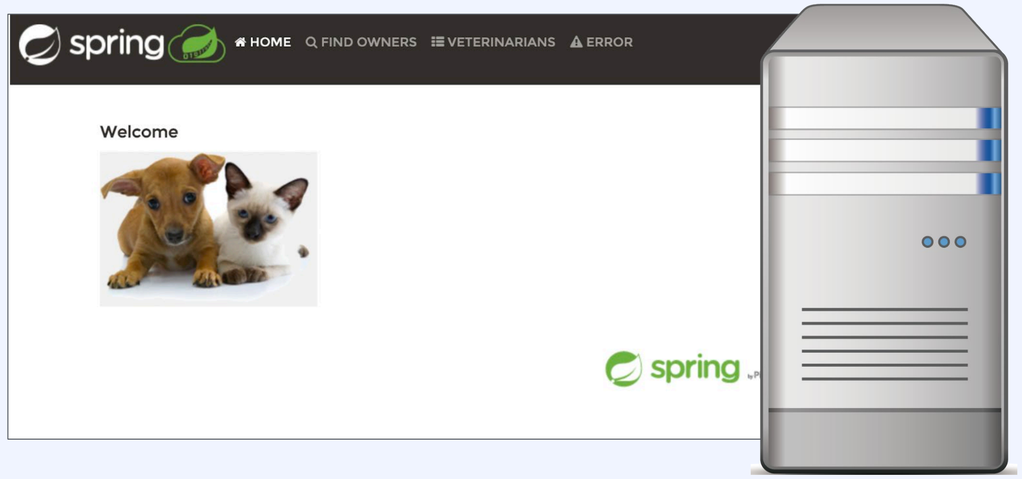
- CICD
- Jenkins 설치

클라우드 인스턴스를 이용한 애플리케이션 운영

Amazon RDS(MySQL 8.x) 구성

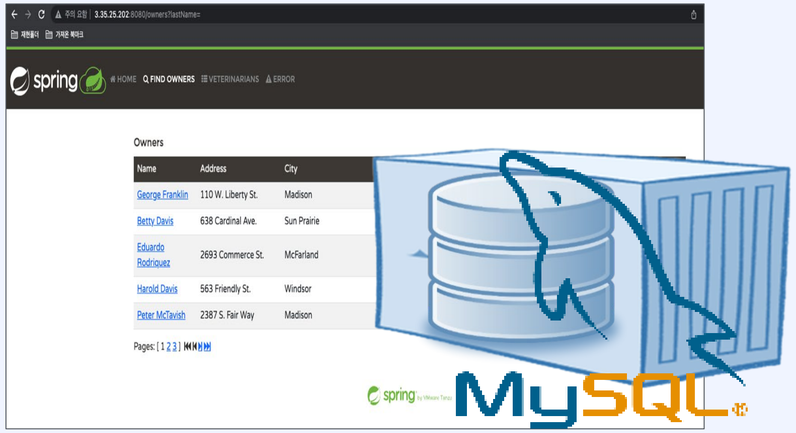


PetClinic 애프리케이션 서버 구성 및 실행



컨테이너 기반의 애플리케이션을 빌드하고 운영하기

MySQL 8.x 데이터베이스 실행



애플리케이션 컨테이너 빌드 및 운영

자동 빌드와 자동 배포 서비스 운영

MySQL 8.x 데이터베이스 실행

애플리케이션 컨테이너 빌드 및 운영

Jenkins 개요

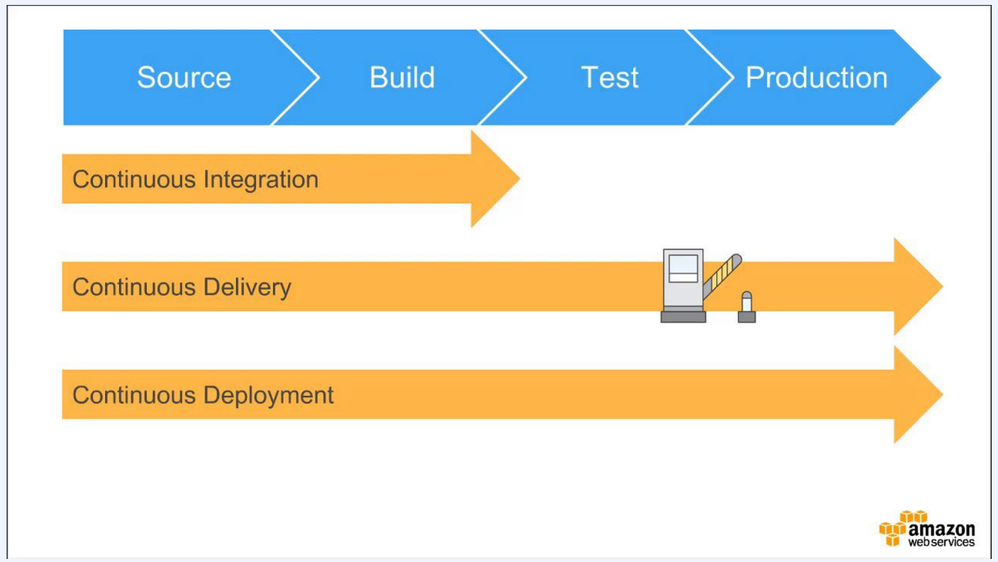
Jenkins 란?

- 소프트웨어 구축, 테스트, 전달 및 배포와 관련된 모든 종류의 작업을 자동화하는데 사용할 수 있는 오픈 소스 자동화 서버로 시스템 패키지, Docker 또는 JRE(Java Runtime Environment)가 설치된 모든 환경에서 실행할 수 있습니다. 젠킨스는 다양한 플러그인들을 조합하여 일을 처리하는 Pipeline을 통해 CI/CD Pipeline을 구축합니다.

CICD

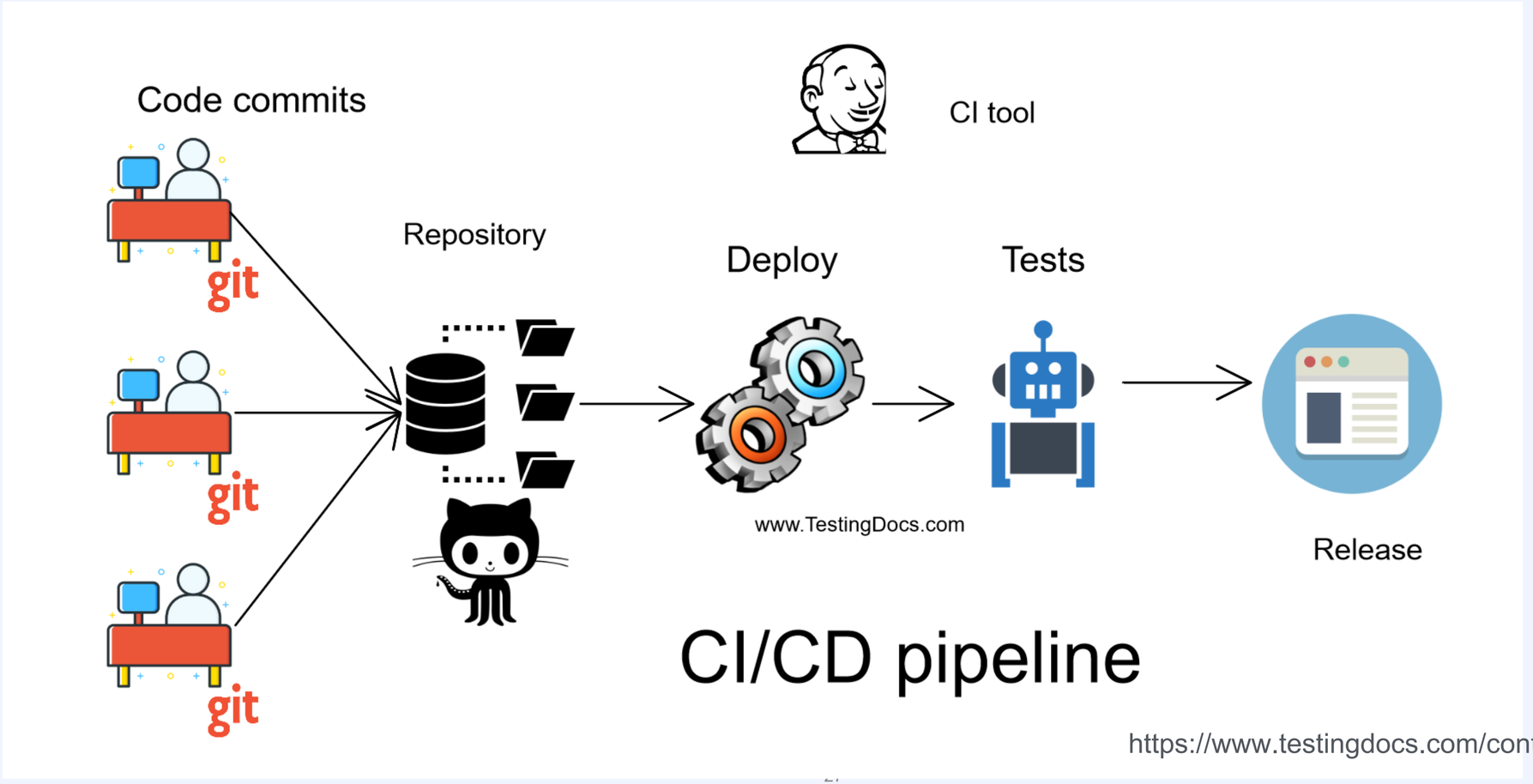
- CI(Continuous Integration, 지속적인 통합)
  - 어플리케이션의 새로운 코드 변경 사항이 정기적으로 빌드 및 테스트 되어 공유 레포지토리에 통합하는 것

- CD(Continuous Delivery/Continuous Depolyment, 지속적인 서비스 제공 혹은 지속적인 배포)
  - 개발자의 변경 사항이 레포지토리를 넘어, 고객의 프로덕션(Production) 환경까지 릴리즈 되는 것



CICD 기반의 자동화 배포및 운영

CICD



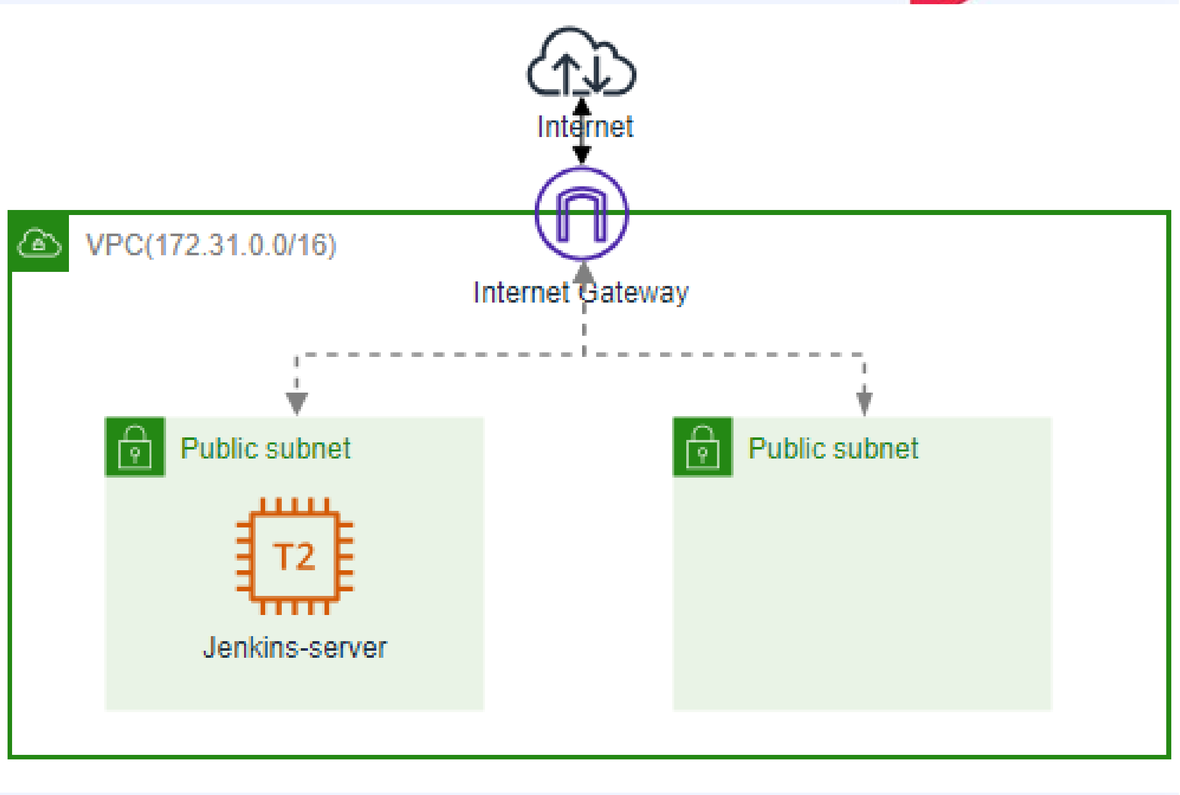
https://www.testingdocs.com/continuous-integration-tools/

Jenkins server 구성 - 인프라

- EC2
  - name: jenkins-server
  - AMI: Amazon Linux(Kernel 5.10)
  - Instance : **t2.medium**
  - ssh Login Access Key

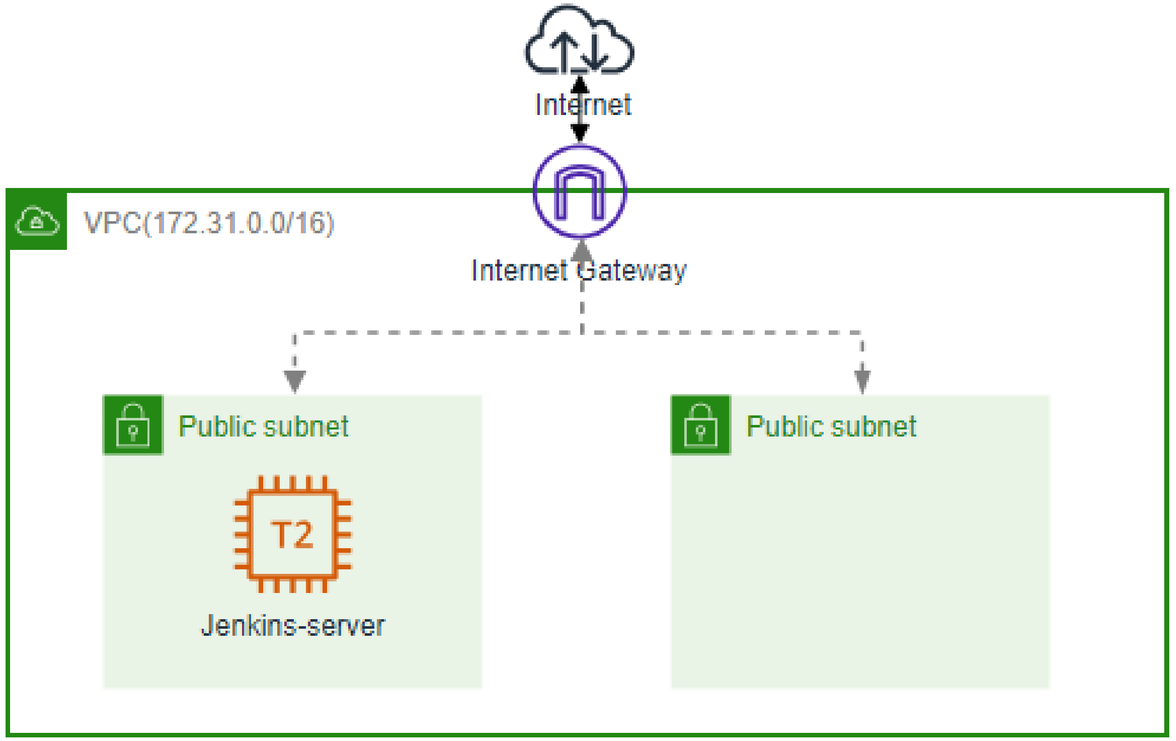
- VPC
  - default 2a, 퍼블릭 IP 할당
  - 보안그룹: jenkins-all - 모든트래픽허용

- 스토리지
  - 범용 SSD(gp3) : 50GiB



Jenkins server 구성 - 서비스 환경

- docker 설치
  - ec2-user 계정 로그인 후 docker install
- Jenkins
  - 컨테이너 기반 Jenkins 설치
- 스토리지
  - 범용 SSD(gp3) : 50GiB



우리가 배운 내용

- 클라우드 환경에서 Jenkins 인프라 구성
- Jenkins 컨테이너 빌드
- Jenkins 운영

백엔드 개발자를 위한 빌드 자동화- Jenkins

4 Git 연동을 위한 Jenkins 환경설정

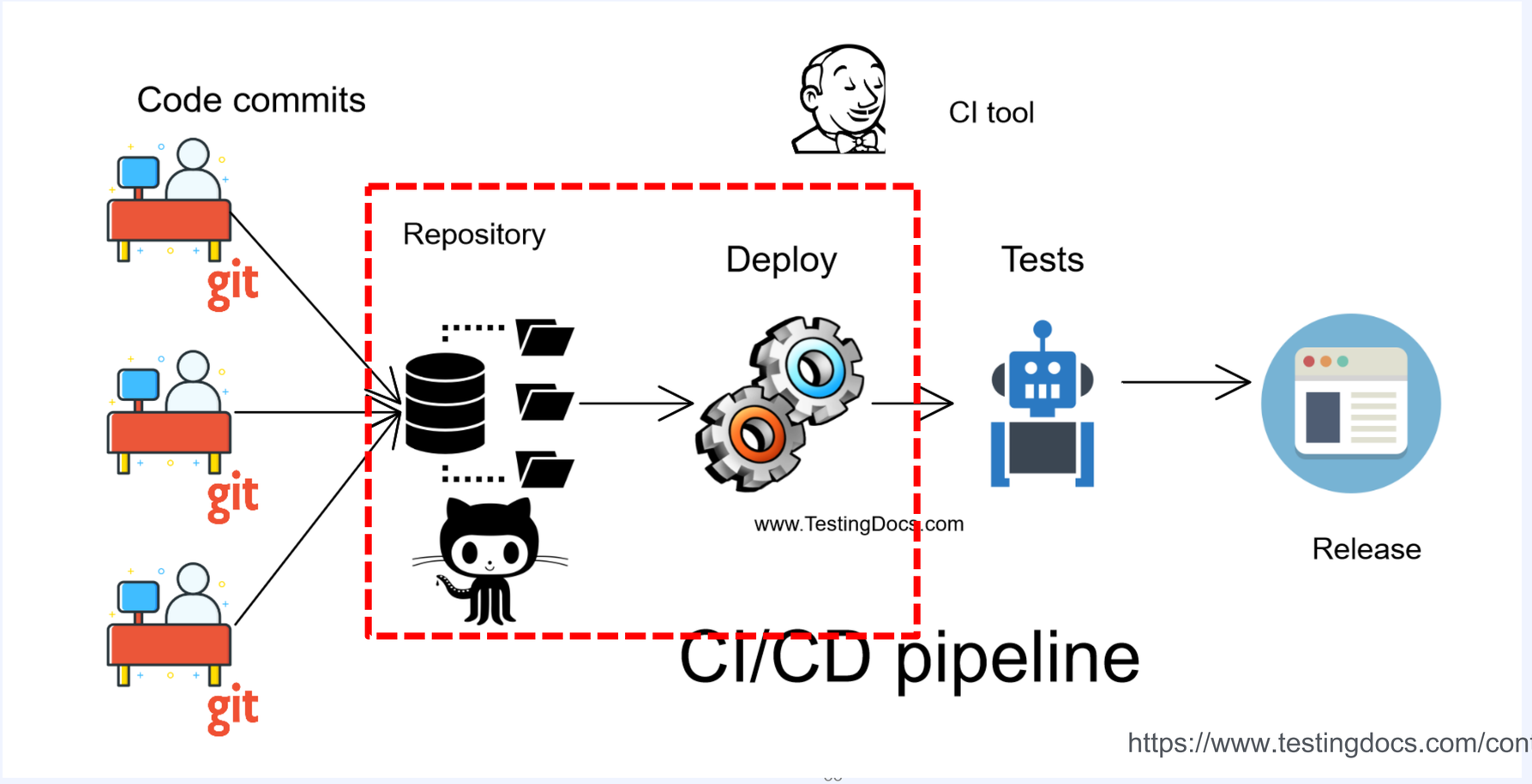
Git 연동을 위한 Jenkins 환경설정

학습 내용

Jenkins pipeline 이란?

Git 연동을 위한 Jenkins 환경설정

Jenkins Pipeline 첫단계 : 깃 연동



https://www.testingdocs.com/continuous-integration-tools/

Jenkins pipeline script

- Pipeline DSL(Domain Specific Language) 코드로 워크로드 정의
- Job을 수동으로 설치할때 반복되는 생성, 버전관리 등의 유지보수의 어려움 대응 - 코드로 운영

pipeline { ... } : 파이프라인 구문의 시작/종료

section

agent : 블록의 최상단에 기록
jenkins 실행자를 전체 파이프라인에 사용(any), 특정 stage에 사용 (none)할지 설정.

stage : pipeline에는 하나 이상의 stage를 포함.
steps 하위 항목이 Jenkins 플러그인(sh: Shell Script)으로 실행되어 명령어 실행

```
stage('stage명') {
            steps{
               할 일
            }
        }
```

Directives : 파이프라인의 config 설정 값

environment : 파이프라인 혹은 스테이지 내부에서 쓰일 변수값을 정의
tool: 자동 설치나 Path에 추가할 도구를 정의
parameter: 파라미터 정의

```
pipeline {
    agent any
    stages {
        stage('sh 플러그인 실습') {
            steps {
                sh '''
                    id
                    echo $$
                    sleep 30
                    pwd
                    cat /proc/$(echo $$)/cmdline
                    env | sort
                '''
            }
        }
    }
}
```

git 연동을 위한 Jenkins 파이프라인 구성: 수동빌드

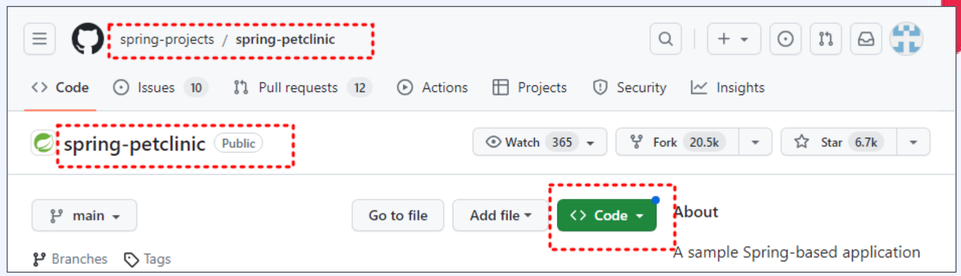
수동빌드 예:
소스 코드 다운로드 후 github에 binary 업로드 - 구현

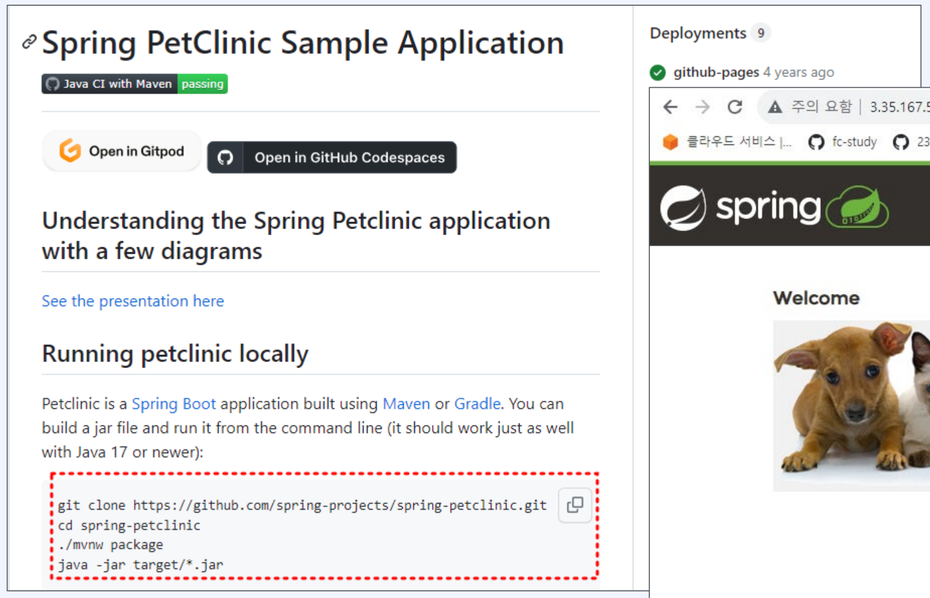
git clone https://github.com/spring-projects/spring-petclinic.git
cd spring-petclinic
./mvnw package

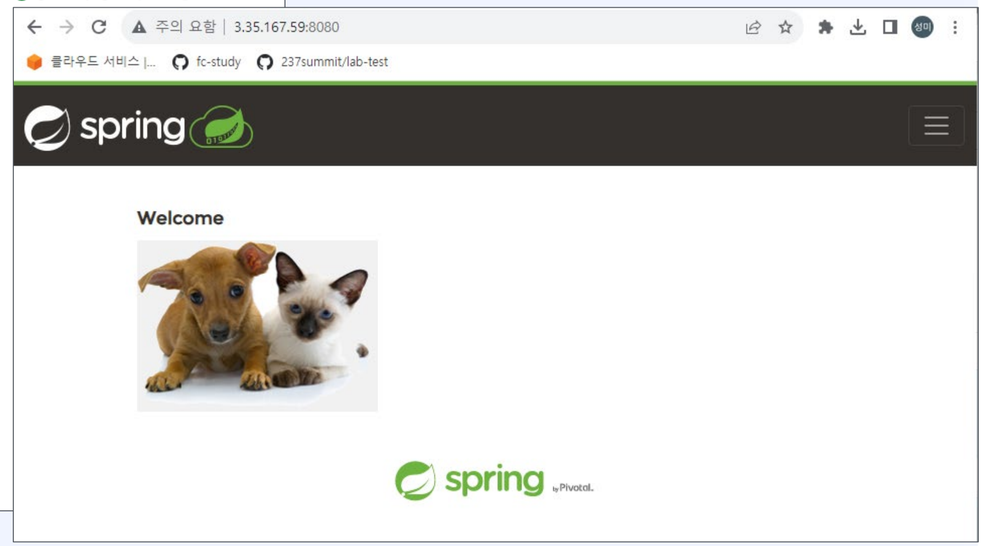
target/petclinic-2.7.0-SNAPSHOT.jar

동작 TEST
java -jar target/*.jar

웹브라우저 접속 TEST
http://jenkins-server's_EIP:8080/



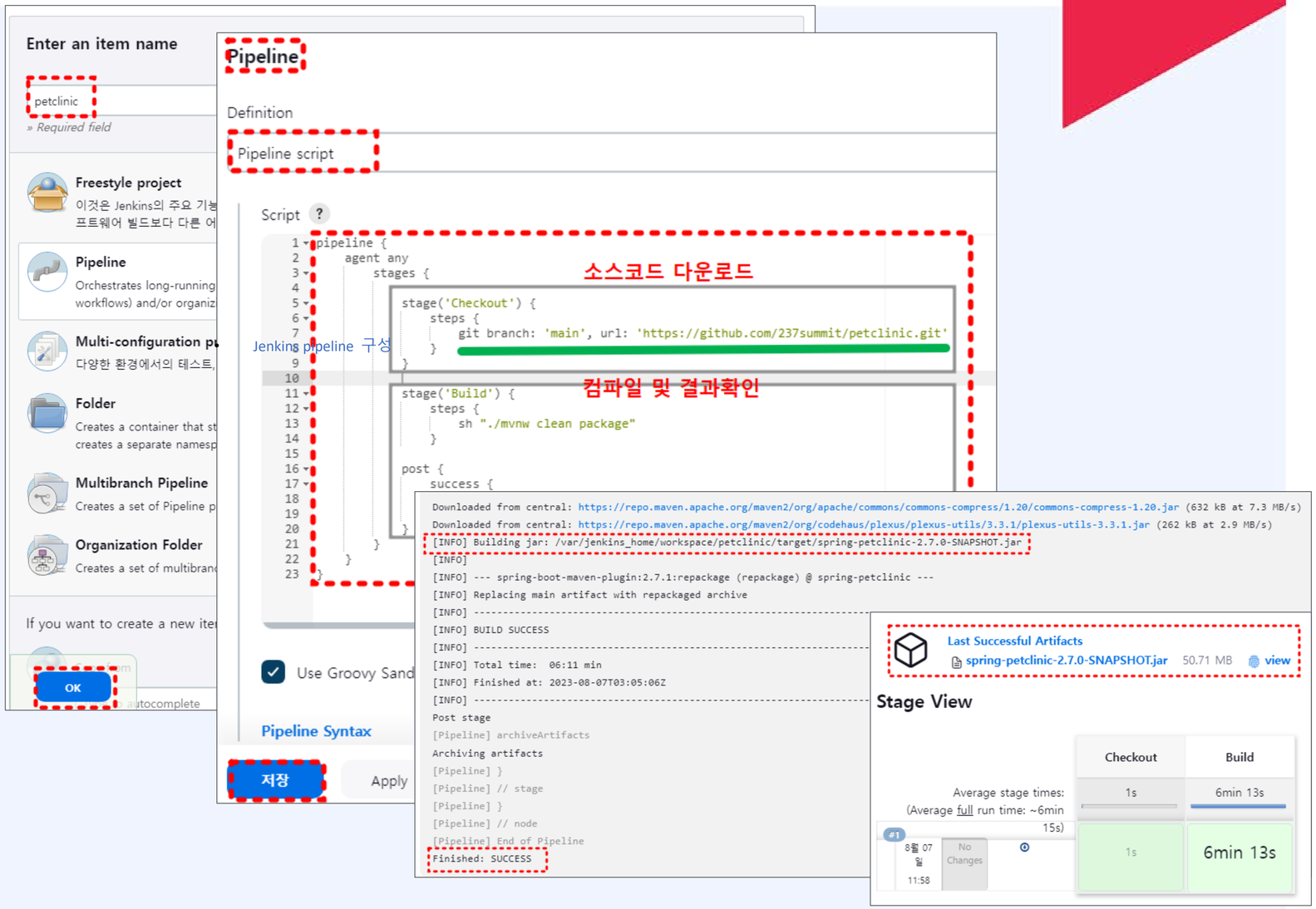




git 연동을 위한 Jenkins 파이프라인 구성: 자동 빌드

Jenkins Project 만들기

- 새로운 Item 만들기
  - item name: petclinic
  - Pipeline
- Configure
  - 설명 : 1. GIT: petclinic build
  - Pipeline : pipeline script
    - 스크립트 넣고 [저장]
- 빌드 진행
- 콘솔 로그보기
- 아카이브 링크 확인



우리가 배운 내용

- DSL코드를 이용한 Jenkins Pipeline 구성
- Git 연동으로 소스코드 다운로드 및 컴파일

백엔드 개발자를 위한 빌드 자동화- Jenkins

5 Jenkins 파이프라인을 이용한 컨테이너 빌드

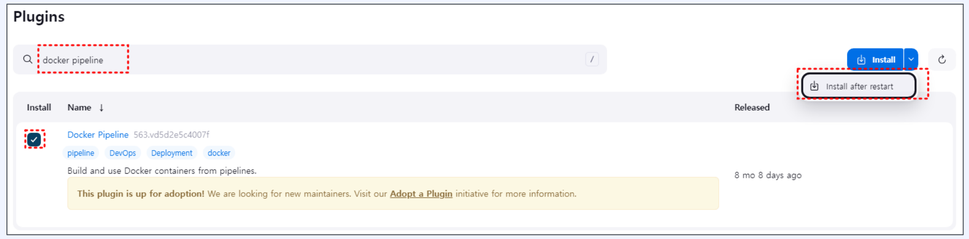
Jenkins 파이프라인을 이용한 컨테이너 빌드

학습 내용

- 도커 플러그인 설치
- Pipeline을 이용한 컨테이너 빌드

Jenkins Pipeline을 이용한 Docker 연동

- **플러그인 설치**
  - Jenkins 관리 -> [System Configuration] - Plugins -> [Available package] - 'docker pipeline' 검색후 선택 > Install



- **Pipeline script 구성후 컨테이너 빌드**

```
pipeline {
    agent any
    stages {
        stage('Checkout') {
            steps {
                git branch: 'main', url: 'https://github.com/237summit/petclinic.git'
            }
        }
        stage('Build') {
            steps {
                sh './mvnw clean package'
            }
        }
        stage('Docker Image Build') {
            steps {
                script {
                    docker.build("petclinic:v${BUILD_ID}")
                }
            }
        }
    }
}
```

우리가 배운 내용

- Jenkins Plugin 설치
- Jenkins pipeline을 이용한 컨테이너 빌드

백엔드 개발자를 위한 빌드 자동화- Jenkins

6 Jenkins Slack Webhook 연동하기

Jenkins Slack Webhook 연동하기

학습 내용

- Slack의 워크스페이스와 채널 생성
- 젠킨스에 슬랙 플러그인 설치 및 구성
- Jenkins Pipeline 이용해 Slack 연동

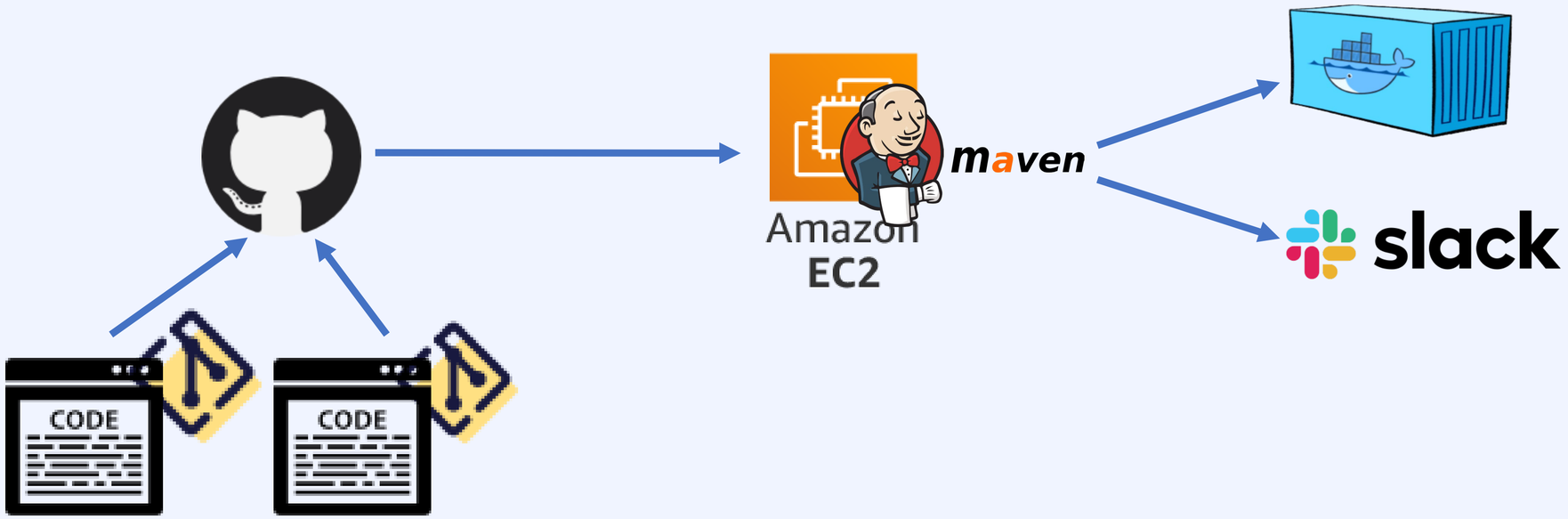
Jenkins Slack Webhook 연동하기

Slack 이란?

- Slack은 채널 기반 메시징 플랫폼입니다. Slack 워크스페이스는 사람들이 협업하고 모든 소프트웨어 도구와 서비스를 연결하며 최고의 작업을 수행하는 데 필요한 정보를 찾을 수 있는 장소입니다. - slack.com

Jenkins Slack 연동

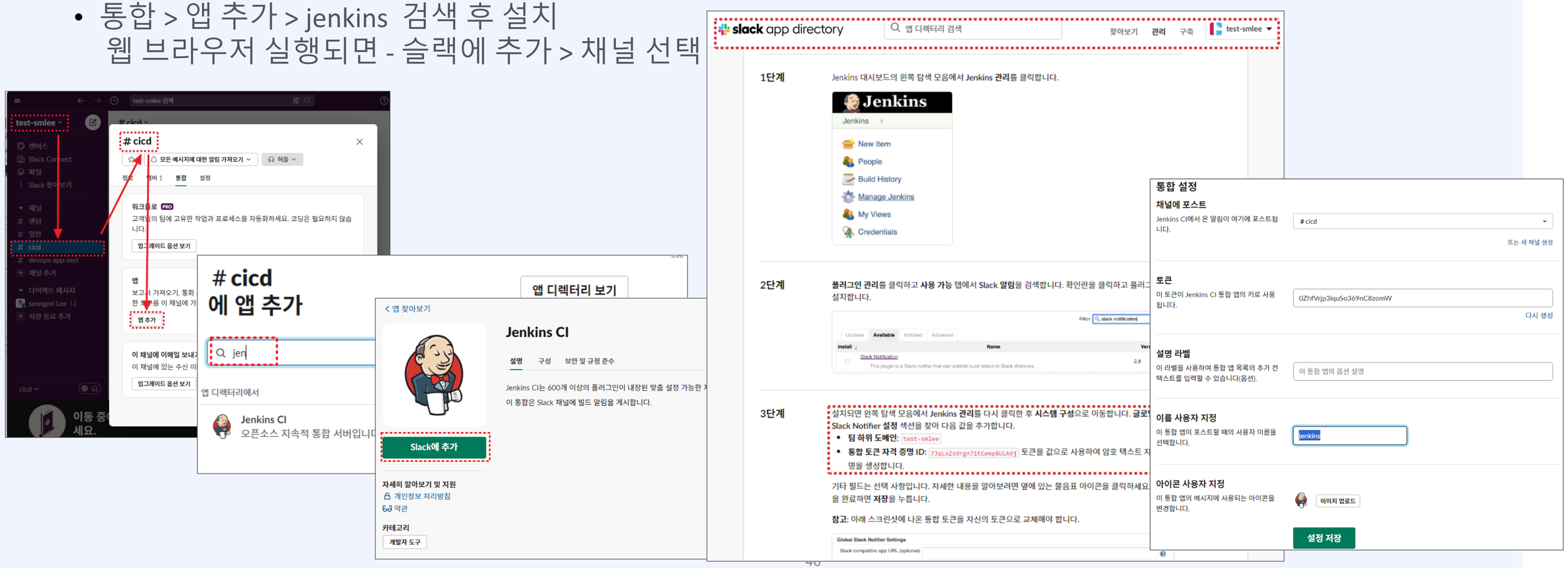
- Jenkins 에서 Pipeline script를 동작할때 성공 및 실패 여부를 슬랙을 통해 노티 할수 있다.



Jenkins Slack Webhook 연동하기

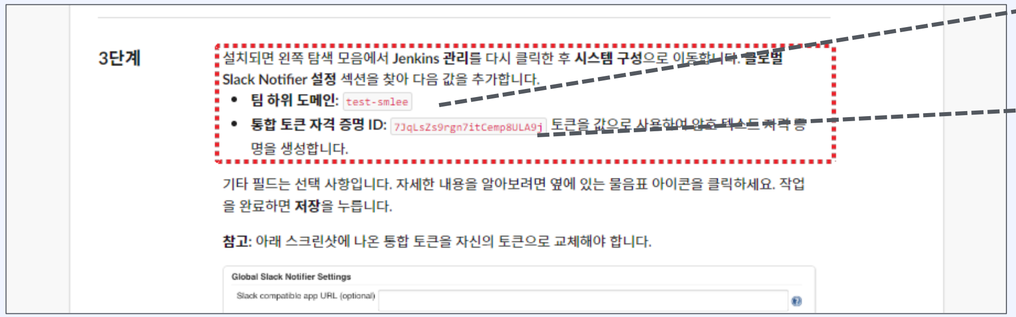
Slack과 Jenkins 연동하기

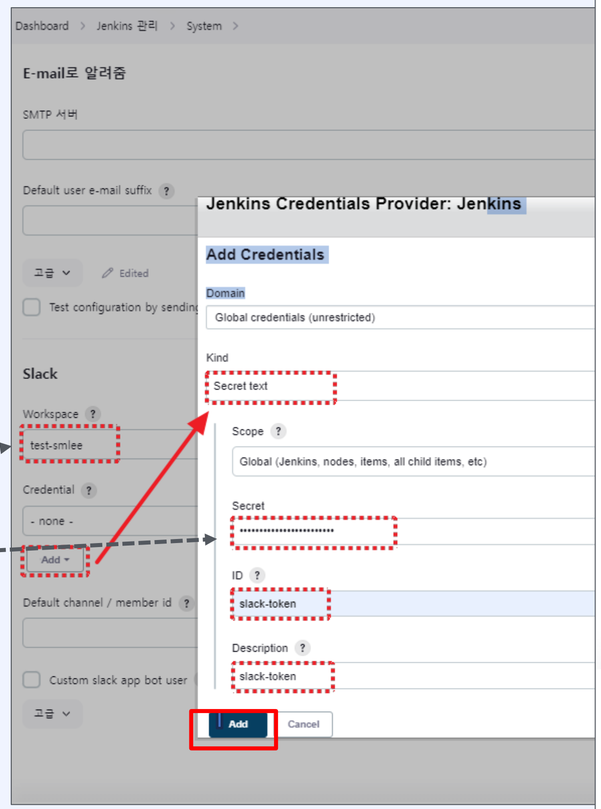
- 슬랙 워크스페이스 생성 -> 채널 생성
- 채널에 Jenkins 연동 : Jenkins에서 slack 채널에 연결할 수 있도록 권한 할당
  - 통합 > 앱 추가 > jenkins 검색 후 설치
    웹 브라우저 실행되면 - 슬랙에 추가 > 채널 선택

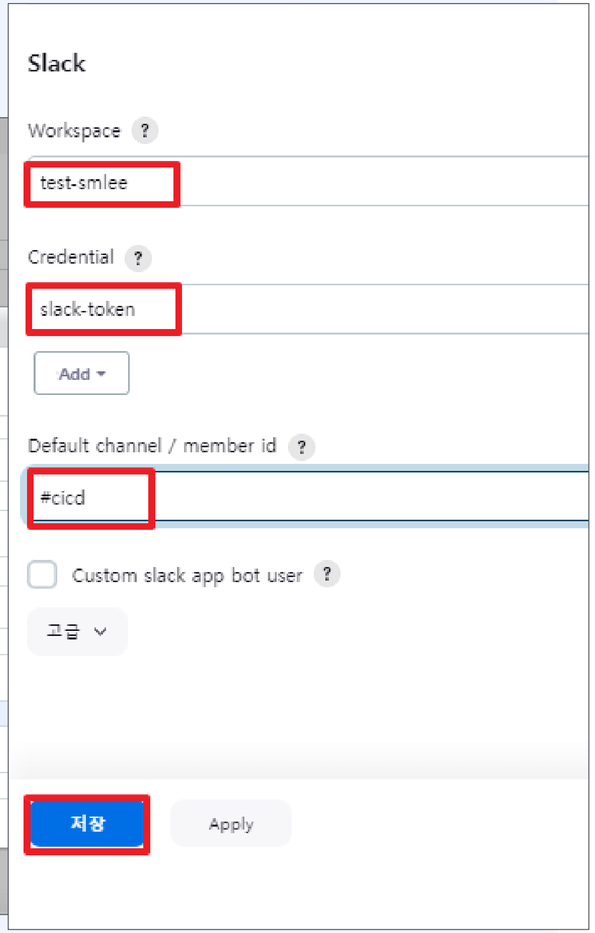


Slack과 Jenkins 연동하기

- 슬랙 워크스페이스 생 성 -> 채널 생성
- 채널에 Jenkins 연동 : Jenkins에서 slack 채널에 연결할 수 있도록 권한 할당
  - 통합 > 앱 추가 > jenkins 검색 후 설치 >
    웹 브라우저 실행되면 – 슬랙에 추가 > 채널 선택
- Jenkins 관리
  - 플러그인 설치: ‘slack notification’ 플러그인 설치
  - 시스템 구성 – slack (Credential – [Add] 클릭)
  - petclinic 프로젝트에 pipeline script 로 slack 노티 구성



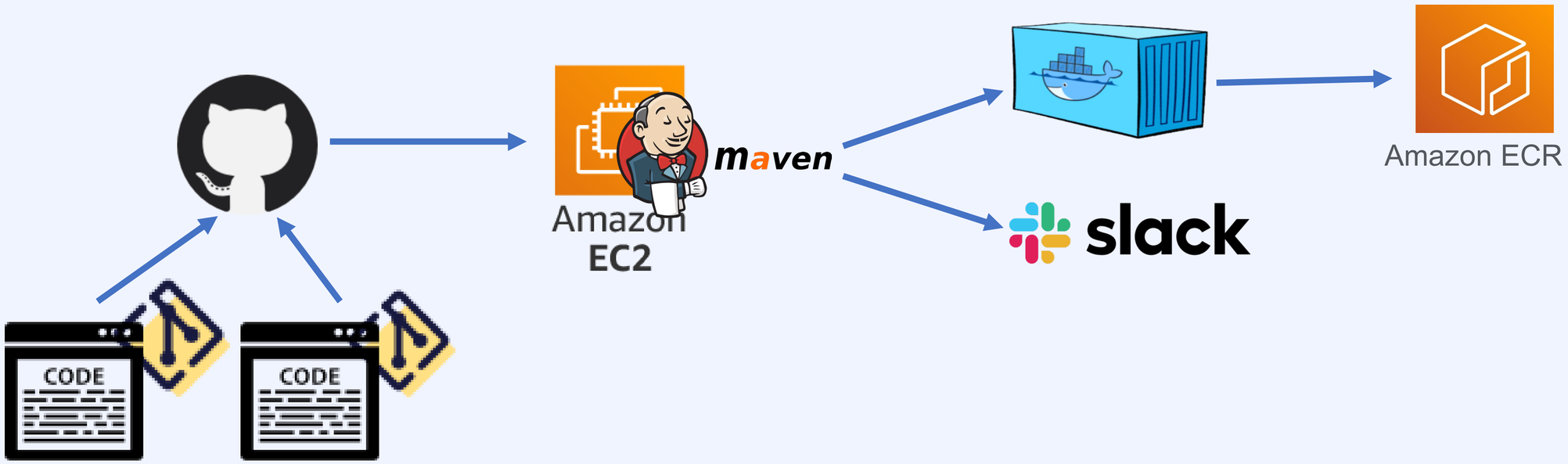




백엔드 개발자를 위한 빌드 자동화- Jenkins

6 Jenkins Slack Webhook 연동하기(2)

Amazon ECR 구성(1)



Jenkins Slack Webhook 연동하기(2)

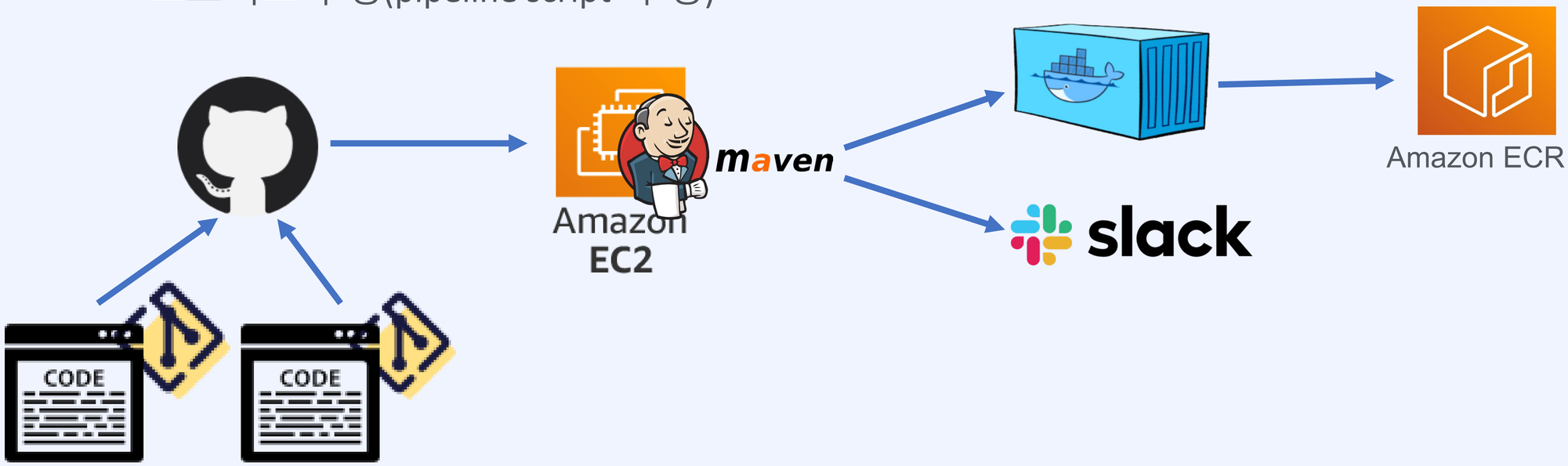
학습 내용

AWS :
- ECR 정책 생성
- 사용자 콘솔 액세스키 생성
- Amazon ECR 생성

Jenkins:
- Plugin : AWS Credentials, Pipeline: AWS Steps
- Credentials : 사용자 콘솔 액세스키 등록
- 프로젝트 구성 : pipeline script 구성

Amazon ECR 구성(1)

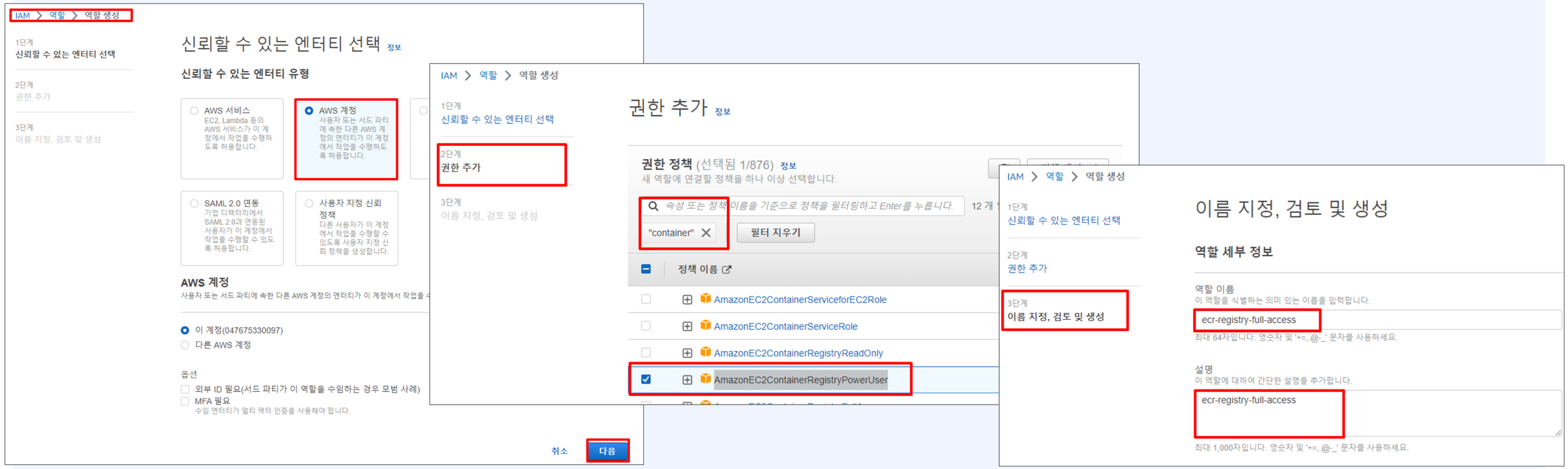
- **빌드 된 컨테이너 이미지가 로컬이 아닌 Amazon ECR에 저장**
- **AWS** : ECR 정책(role) 생성 -> 사용자 콘솔 액세스 키 생성 -> Amazon ECR 생성
- **Jenkins**: Plugin(AWS Credentials, Pipeline: AWS Steps)설치 -> Jenkins-Server: aws cli 설치
  Jenkins 관리: AWS Credentials 등록
  프로젝트 구성(pipeline script 구성)



Amazon ECR 구성(1)

AWS 구성

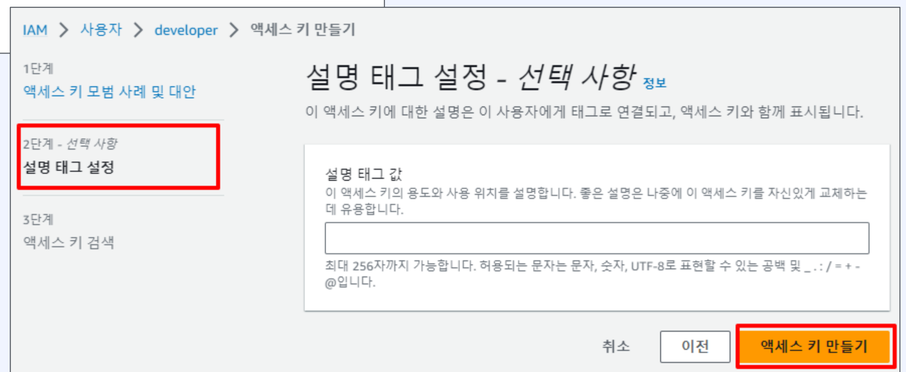
- ECR 역할(role) 생성 : **ecr-registry-full-access**(AmazonEC2ContainerRegistryPowerUser)
- 사용자 콘솔 액세스키 생성
- Amazon ECR 생성



Amazon ECR 구성(2)

AWS 구성

- ECR 권한 생성
- 사용자 콘솔 액세스키 생성 : user > 계정선택 > 액세스키, 비밀 액세스키 복사
- Amazon ECR 생성



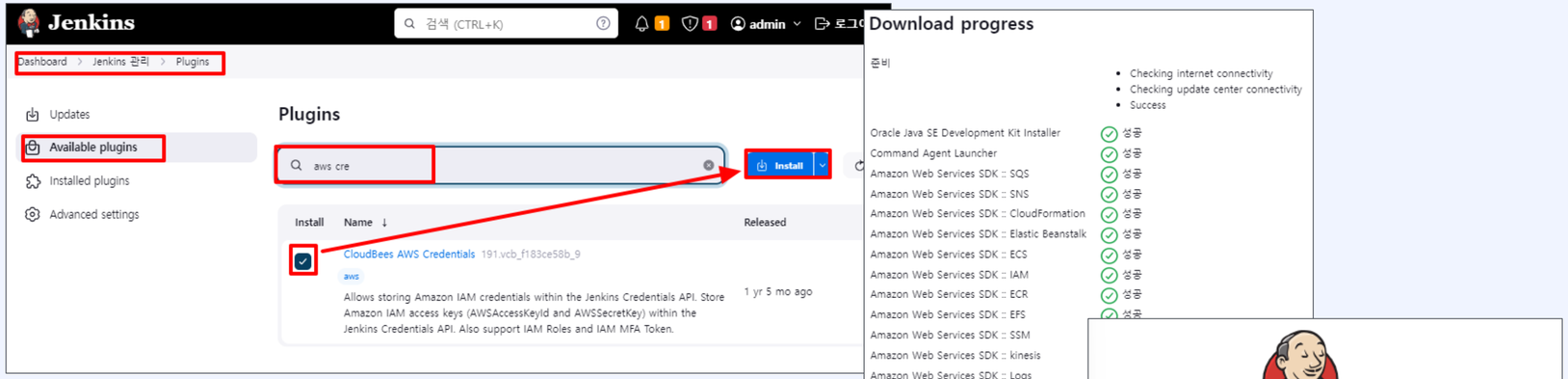
Amazon ECR 구성(3)

AWS 구성

- ECR 권한 생성
- 사용자 콘솔 액세스키 생성 : user > 계정선택 > 액세스키, 비밀 액세스키 복사
- Amazon ECR 생성

Jenkins 구성

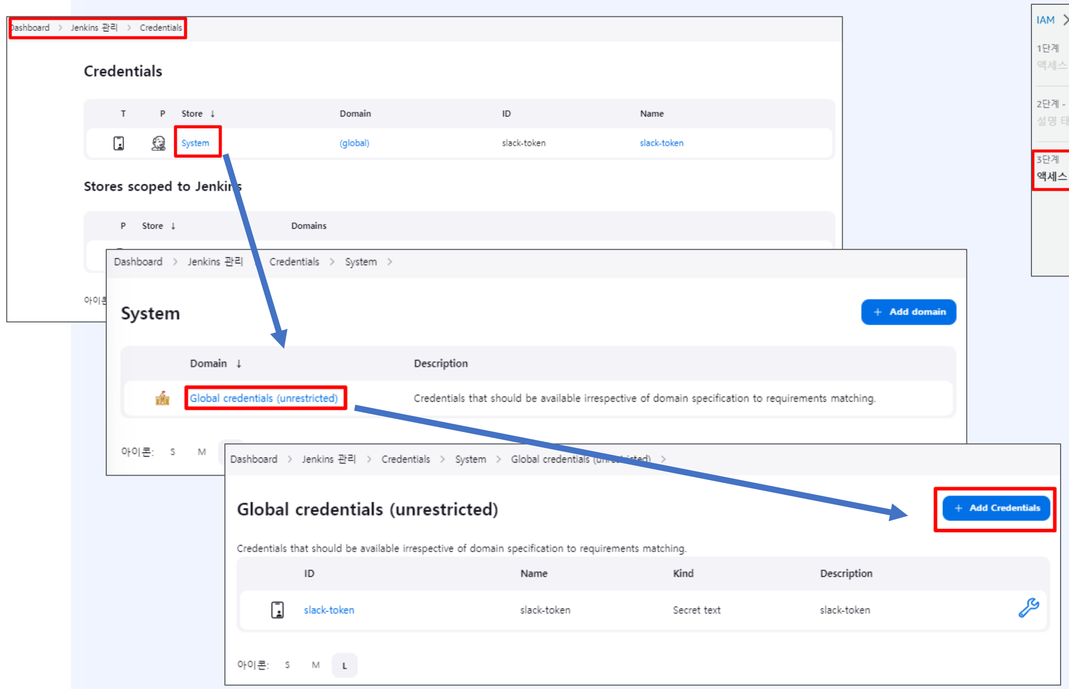
- Plugin 설치 : AWS Credentials, Pipeline: AWS Steps
  Jenkins-Server: aws cli 설치
- Credentials 등록 : AWS 사용자 콘솔 액세스키 등록
- Jenkins 프로젝트 구성 : pipeline script 구성

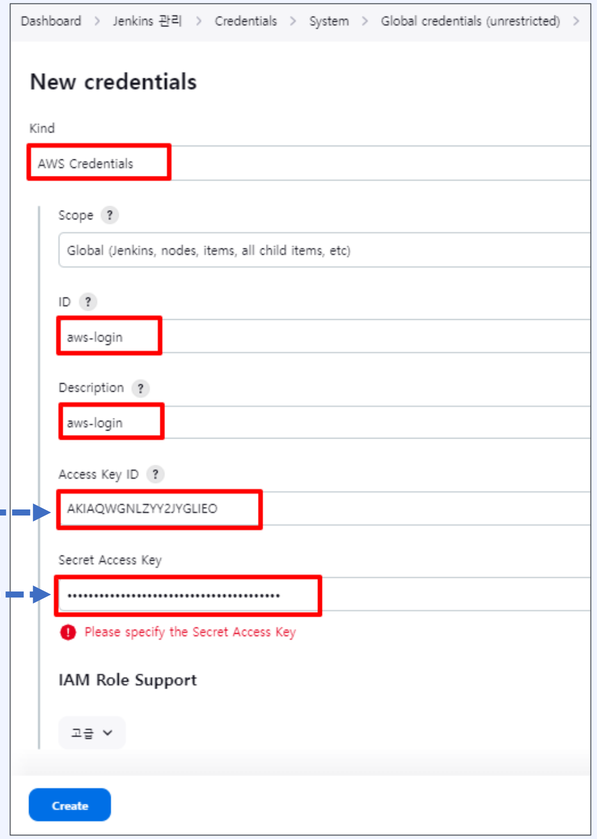


Jenkins 구성(2)

Jenkins 구성

- Plugin 설치 : AWS Credentials
- Credentials 등록 : AWS 사용자 콘솔 액세스키 등록(aws-login)
- Jenkins 프로젝트 구성 : pipeline script 구성





Jenkins 구성(3)

Jenkins 구성

- Plugin 설치 : AWS Credentials
- Credentials 등록 : AWS 사용자 콘솔 액세스키 등록
- Jenkins 프로젝트 구성 : pipeline script 구성

우리가 배운 내용

- AWS 관리콘솔에서 ECR 접근권한/액세스키 생성,컨테이너 저장소(Amazon ECR)만들기
- Jenkins Plugin(AWS Credentials) 등록을 통해 Jenkins 에서 컨테이너 빌드 후 ECR에 업로드

백엔드 개발자를 위한 빌드 자동화- Jenkins

7 실전 빌드하기

source push → container build → ecr에 업로드

실전 빌드하기: source push → container build → ecr에 업로드

- Jenkins Pipeline Flow

Code Push

Jenkins PipeLine Code

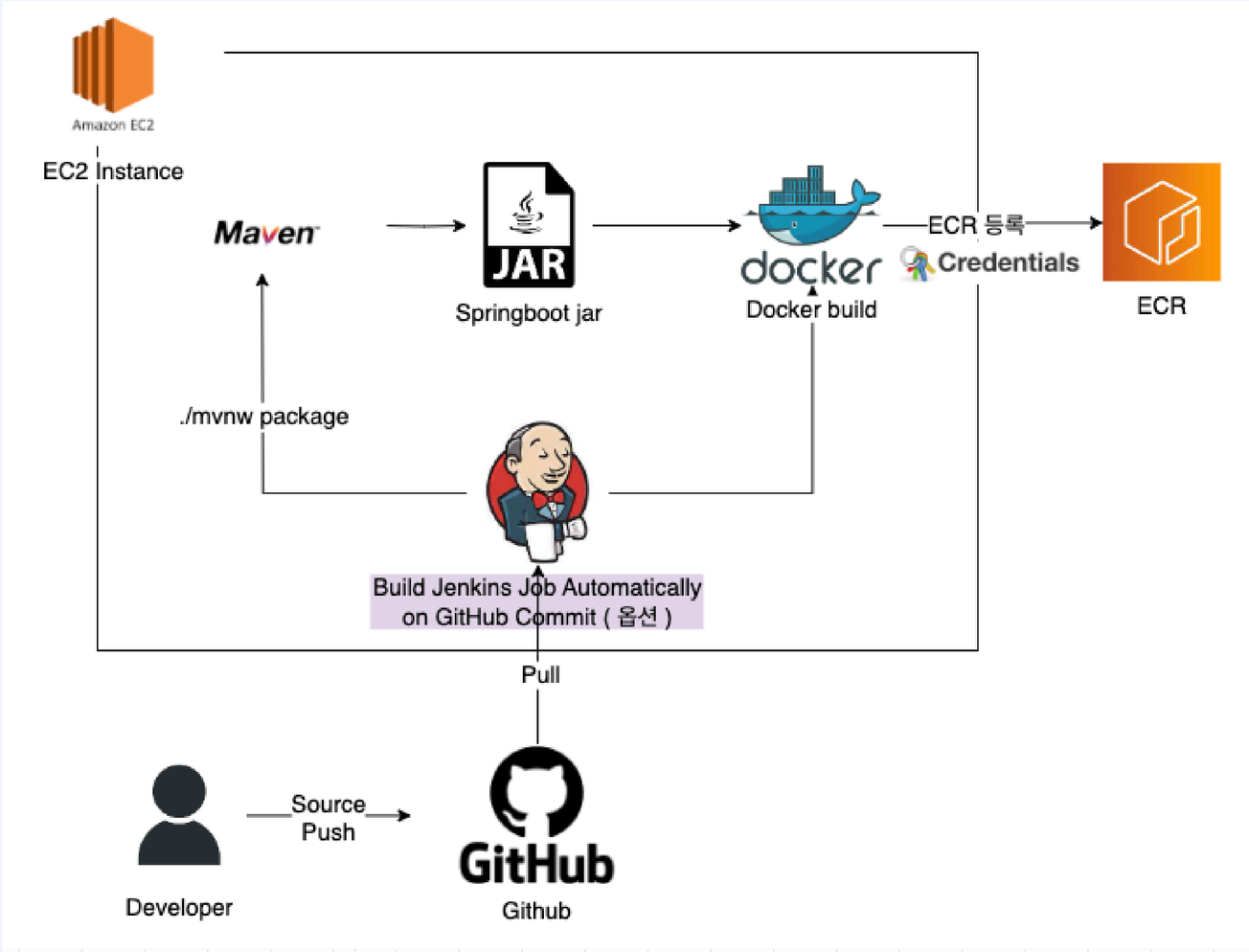
Jenkins WebHook (option)

Jenkins Build Start

Jenkins Docker build

Jenkins Docker image Tag
ECR repo add

docker image ECR upload



실전 빌드하기: source push → container build → ecr에 업로드

Jenkins Pipeline 작업순서

- Iam – Policy , Rule 생성 (ECR 업로드 하기 위해서 사용합니다)
  - Policy 생성
  - Rule 생성
- ECR repo 생성 - Name : spring-petclinic
- Aws Credential 등록
  - 보안자격증명 > 액세스 키 생성 > Accesskey , SecretKey 생성
  - jenkins credential 등록
- pipeline 코드 리뷰 - Jenkins Pipeline공식 가이드 문서
- pipeline job 등록
- Jekins Webhook 등록 - Auto Job Execute (선택)
  - LB 생성 – github webhook url 등록시 필요 (주의 - public ip는 사용불가)

우리가 배운 내용

- IAM 생성 - ECR 업로드시 필요한 Rule
- aws credential 등록
- jenkins pipeline 코드 리뷰 - 키워드 withAWS , dir , slackSend

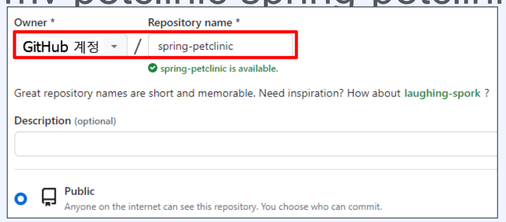
백엔드 개발자를 위한 빌드 자동화- Jenkins

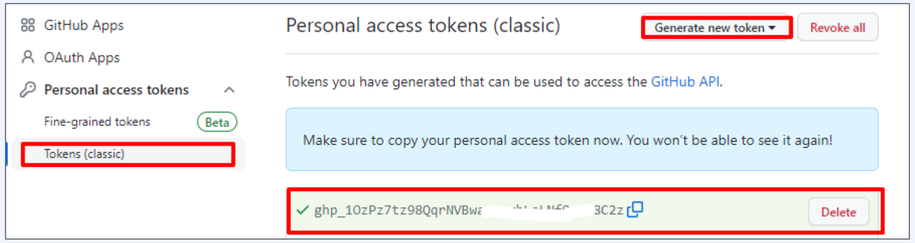
8 LAB: CI 구축하기
git 코드 업로드 부터 컨테이너 빌드까지

문제 : 빌드한 젠킨스 컨테이너로 git 코드 업로드부터 컨테이너 빌드 구성

1. **Jenkins 컨테이너 이미지 빌드** : Jenkins 플러그인을 포함한 컨테이너 이미지 빌드 후 jenkins 실행
   chapter-5/build 디렉토리에 있는 Dockerfile을 이용해 jenkins 컨테이너 빌드
   docker build -t jenkins-plugin:2.419 .

2. **Petclinic 소스코드 clone 후 자신의 Github에 소스코드 push**. 이때 Github 로그인 토큰 생성
   git clone https://github.com/237summit/petclinic.git
   mv petclinic spring-petclinic





   git push -u origin master

3. **Jenkins 통해 pet-clinic 컨테이너 이미지 빌드 후 Amazon ECR에 Push**
   AWS 구성: 액세스키 생성, role생성 (ecr-full-access: AmazonEC2ContainerRegistryPowerUser),
   Amazon ECR private repository : spring-petclinic
   젠킨스 구성 – slack-token, aws credendtials 등록(aws-login-token), 새로운 아이템 생성: spring-petclinic

LAB: git 코드 업로드 부터 컨테이너 빌드까지 - CI 구축하기

우리가 배운 내용

- Jenkins를 플러그인을 포함한 컨테이너 이미지 빌드
- Jenkins 파이프라인 구성을 통해 애플리케이션 이미지 빌드 자동화

미래가 오늘이다.