



Jenkins 와 CI/CD

Jenkins 를 활용한 실무 CI/CD

강의 목표

- CI/CD 파이프라인의 기본 개념에 대해서 이해한다.
- 기본적인 운영환경(DEV, QA, PROD) 이 어떻게 구성되고 운영되는지 이해한다.
- Jenkins 의 기본 개념에 대해 이해한다.
- Jenkins 를 통해 기본적인 배포 파이프라인을 직접 구축할 수 있다.
- 실제 운영기에서 특히 AWS 기반의 클라우드 환경에서 Jenkins 가 어떻게 활용되는지 알 수 있다.

목차

1. CI/CD 란 무엇인가?
2. Jenkins 의 기본 개념과 동작 방식
3. 개발 환경 및 CI/CD 의 기본 동작 이해
4. Jenkins 및 플러그인 설치 실습
5. CI/CD 파이프라인 구축 및 QnA
6. 실제 운영 환경에서 Jenkins 사용 사례 알아보기(DW ETL Pipeline)

1. CI/CD 란 무엇인가?

CI/CD 란?

Continuous Integration => 뭘 통합한다는 거야?

여러 개발자들의 코드베이스를 계속해서 통합하는 것.

continuous integration (CI) is the practice of merging all developers' working copies to a shared mainline several times a day

Continuous Delivery => 무엇을 배달한다는 거야?

사용자에게 제품을 서비스를 지속적으로 배달한다!
코드베이스가 항상 배포가능한 상태를 유지하는 것.

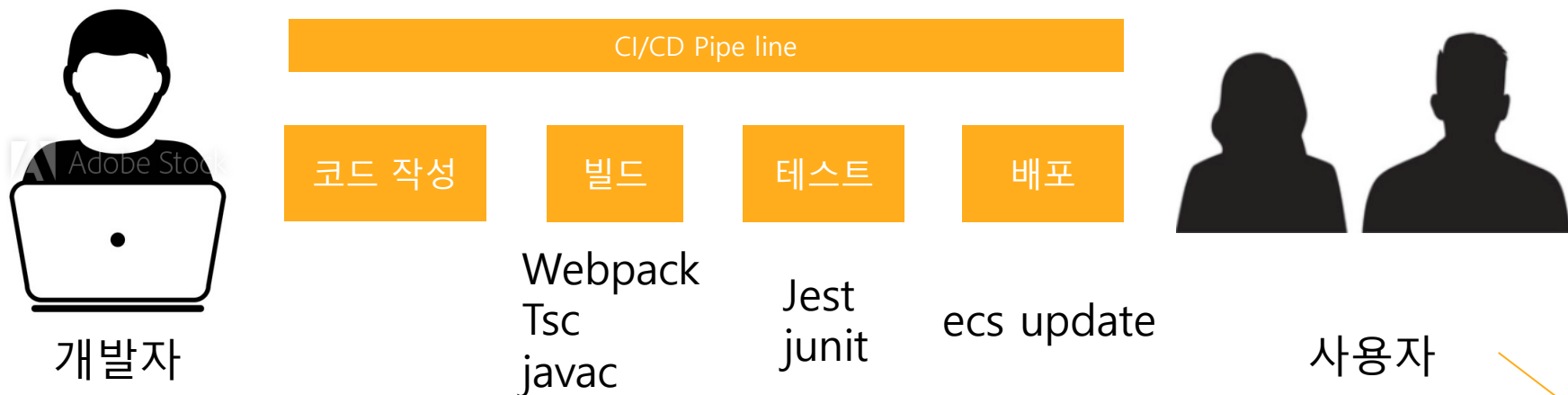
Continuous Deployment

코드베이스를 사용자가 사용가능한 환경에 배포하는 것을 자동화함.

1. CI/CD 란 무엇인가?

즉, CI/CD 란 각각의 개발자들이 개발을 하는 개발 환경을 사용자가 사용 가능한 서비스로 전달하는 모든 과정을 지속 가능한 형태로 또 가능하다면 자동으로 해서 개발자와 사용자 사이의 격차를 없애는 것이다!

이러한 과정에는 코드를 빌드하고, 테스트하고 배포하는 활동이 있다.



1. CI/CD 란 무엇인가?

CI 왜 필요한가요?

- 10명의 개발자가 열심히 개발
- 1 Week Later....
- Merge Hell....
- 10명의 개발자가 열심히 개발
- 1 Week Later....
- 마지막 커밋 누구야 내꺼 안되잖아!

모든 개발자들이 안심하고 개발을 하기 위해서 또, 내 코드와 멘탈의
평안을 위하여....

1. CI/CD 란 무엇인가?

CI 와 함께라면?

- 10명의 개발자가 열심히 개발
- 커밋!
- 로컬 테스트 통과 실패..
- 커밋!
- 코드베이스 머지
- 만족!

가능한 최대한 많이 빨리빨리 내 코드를 코드베이스에 안착시키자!

테스트 코드 없는 무서운 코드 버그 더미 코드를 애초에 코드베이스에서쫓아내자

1. CI/CD 란 무엇인가?

CD 왜 필요한가요?

- 백엔드 코드 개발
- 프론트와 협업해야하니 배포를 해볼까?
- 저기 배포좀 해주세요...
- 앳 버그...! 다시 배포좀 해주세요...
- 데브 서버에 누가 배포했나요? 제꺼 안되는데요;;
- 앳 죄송...
- QA 배포...
- 프로덕션 배포시 초긴장 유지...
- 열심히 배포스크립트 작성, AWS 콘솔 만지작.. 혹은 베어메탈?!

개발자가 코드만 짜면 되지 뭐이리 할게 많아 TT

1. CI/CD 란 무엇인가?

CD와 함께라면?

- 10명의 개발자가 열심히 개발
- 끝. (머지됐으니까 내 역할은 여기까지 peace...)

**QA 엔지니어와 같은 내부사용자 혹은 실제 production 환경의 사용자에게
지속적이고 안정적으로 서비스를 제공한다.**

2. 젠킨스의 기본 개념과 동작 방식

젠킨스 넌 누구냐!



Java Runtime 위에서 동작하는 자동화 서버!
빌드, 테스트, 배포 등 모든 것을 자동화 해주는 자동화 서버!

비서!
난 개발만 할테니까 귀찮은건 니가 다해라!

2. 젠킨스의 기본 개념과 동작 방식

기본 개념

- Java Runtime Environment 에서 동작
- 다양한 플러그인들을 활용해서 각종 자동화 작업을 처리할 수 있음
- 일련의 자동화 작업의 순서들의 집합인 Pipeline 을 통해 CI/CD 파이프라인을 구축함

2. 젠킨스의 기본 개념과 동작 방식

Plugin

- 정~~~말 많은 플러그인들이 존재
- 대표적인 플러그인들


Credentials Plugin

Gid Plugin

Pipeline (핵심 기능인 파이프라인 마저도 플러그인!)

- 처음에는 그냥 Recommend 해주는 것을 깔면 어지간한건 다 깔려있다!

2. 젠킨스의 기본 개념과 동작 방식

 **Jenkins**


검색


namhoon lee

로그아웃

Dashboard

Plugin Manager

 대시보드로 돌아가기

 Jenkins 관리

업데이트된 플러그인 목록

설치 가능

설치된 플러그인 목록

고급

사용가능	이름 ↓	버전	이전 설치 버전	설치 제거
<input checked="" type="checkbox"/>	Ant Plugin Adds Apache Ant support to Jenkins	1.11		설치 제거
<input checked="" type="checkbox"/>	Apache HttpComponents Client 4.x API Plugin Bundles Apache HttpComponents Client 4.x and allows it to be used by Jenkins plugins. This plugin is up for adoption! We are looking for new maintainers. Visit our Adopt a Plugin initiative for more information.	4.5.10-2.0		설치 제거
<input checked="" type="checkbox"/>	Bootstrap 4 API Plugin Provides Bootstrap 4 for Jenkins plugins.	4.5.2-1		설치 제거
<input checked="" type="checkbox"/>	bouncycastle API Plugin This plugin provides an stable API to Bouncy Castle related tasks.	2.18		설치 제거
<input checked="" type="checkbox"/>	Branch API Plugin This plugin provides an API for multiple branch based projects.	2.6.0		설치 제거
<input checked="" type="checkbox"/>	Build Timeout This plugin allows builds to be automatically terminated after the specified amount of time has elapsed. This plugin is up for adoption! We are looking for new maintainers. Visit our Adopt a Plugin initiative for more information.	1.20		설치 제거
<input checked="" type="checkbox"/>	Checks API plugin This plugin defines an API for Jenkins to publish checks to SCM platforms.	1.0.2		설치 제거
<input checked="" type="checkbox"/>	Command Agent Launcher Plugin Allows agents to be launched using a specified command.	1.4		설치 제거

2. 젠킨스의 기본 개념과 동작 방식

Plugin 살펴보기

Credentials Plugin

Jenkins 는 그냥 단지 서버일 뿐이기 때문에 배포에 필요한 각종 리소스(가령 클라우드 리소스 혹은 베어메탈에 ssh 접근 등) 에 접근하기 위해서는 여러가지 중요 정보들을 저장하고 있어야 한다.

이런 중요한 정보 (AWS token, Git access token, etc...) 들을 저장해 주는 플러그인

Pipeline Plugin

젠킨스의 핵심 기능인 Pipeline 을 관리할 수 있게 해주는 플러그인

2. 젠킨스의 기본 개념과 동작 방식

Plugin 살펴보기

Docker plugin and Docker Pipeline

Docker agent 를 사용하고 jenkins 에서 도커를 사용하기 위함

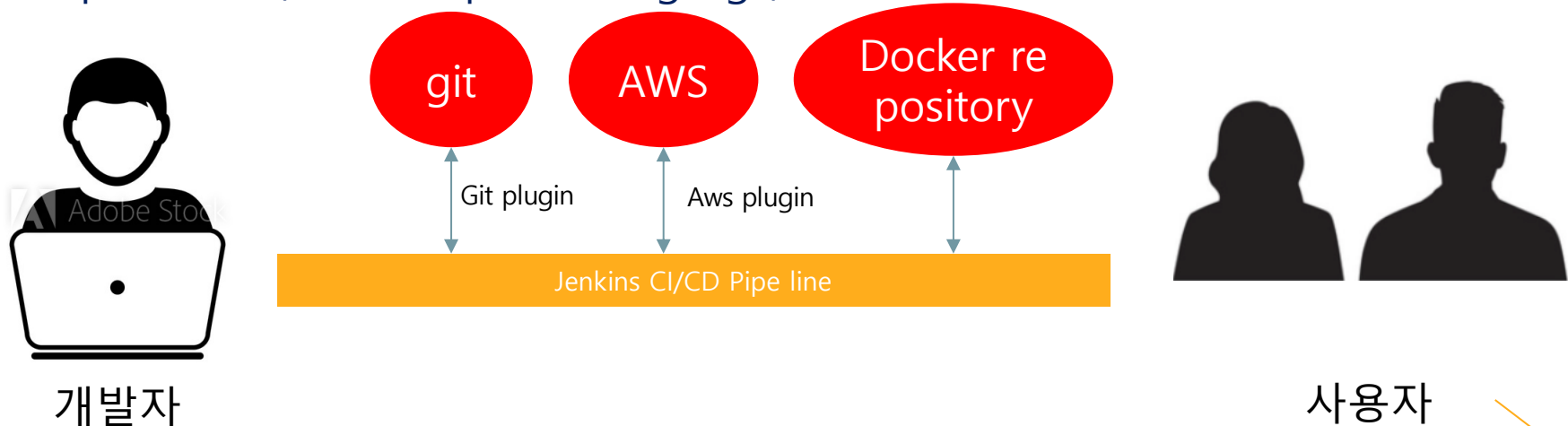
2. 젠킨스의 기본 개념과 동작 방식

Pipeline

파이프라인이란 CI/CD 파이프라인을 젠킨스에 구현하기 위한 일련의 플러그인들의 집합이자 구성.

즉 여러 플러그인들을 이 파이프라인에서 용도에 맞게 사용하고 정의함으로써 파이프라인을 통해 서비스가 배포됨

Pipeline DSL(Domain Specific Language) 로 작성



2. 젠킨스의 기본 개념과 동작 방식

Pipeline 을 구성하는 요소

파이프라인이란 CI/CD 파이프라인을 젠킨스에 구현하기 위한 일련의 플러그인들의 집합이자 구성.

즉 여러 플러그인들을 이 파이프라인에서 용도에 맞게 사용하고 정의함으로써 파이프라인을 통해 서비스가 배포됨

두가지 형태의 Pipeline syntax가 존재(Declarative, Scripted Pipeline)
우리는 최신이자 더 가독성 좋은 문법을 가진 Declarative Pipeline syntax를 사용

2. 젠킨스의 기본 개념과 동작 방식

Pipeline Syntax

Sections

- Agent section
- Post section
- Stages section
- Steps section

으로 구성

2. 젠킨스의 기본 개념과 동작 방식

Pipeline Syntax

Agent Section

젠킨스는 많은 일들을 해야 하기 때문에 혼자 하기 버겁다.

여러 slave node 를 두고 일을 시킬 수 있는데, 이처럼 어떤 젠킨스가 일을 하게 할 것인지를 지정한다.

젠킨스 노드 관리에서 새로 노드를 띄우거나 혹은 docker 이미지등을 통해서 처리할 수 있음

2. 젠킨스의 기본 개념과 동작 방식

Pipeline Syntax

Post section

스테이지가 끝난 이후의 결과에 따라서 후속 조치를 취할 수 있다.

Ex) success, failure, always, cleanup

Ex) 성공시에 성공 이메일, 실패하면 중단 혹은 건너뛰기 등등...

```
post {  
    // If Maven was able to run the tests, even if some of the test  
    // failed, record the test results and archive the jar file.  
    success {  
        echo 'success'  
    }  
}
```

2. 젠킨스의 기본 개념과 동작 방식

Pipeline Syntax

Stages Section

어떤 일들을 처리할 건지 일련의 stage 를 정의함

Steps Section

한 스테이지 안에서의 단계로 일련의 스텝을 보여줌

```
stages {  
    stage('Prepare') {  
        steps {  
            git url: 'https://github.com/frontalnh/pet-insurance.git',  
                branch: 'master',  
                credentialsId: 'jenkinsgit'  
            sh 'ls'  
            dir ('./docs'){  
                sh '''  
                    aws s3 sync ./ s3://namhoontest  
                    '''  
            }  
        }  
    }  
    post {  
        // If Maven was able to run the tests, even if some of the test  
        // failed, record the test results and archive the jar file.  
        success {  
            echo 'success'  
        }  
    }  
}  
  
stage('Build') {  
    steps {  
        echo 'Building...'  
    }  
}
```

2. 젠킨스의 기본 개념과 동작 방식

Pipeline Syntax

Declaratives

Environment, stage, options, parameters, triggers, when 등의 Declarative 가 있음

Environment -> 어떤 pipeline 이나 stage scope 의 환경 변수 설정

Parameter -> 파이프라인 실행시 파라미터 받음

Triggers -> 어떤 형태로 트리거 되는가

When -> 언제 실행되는가

2. 젠킨스의 기본 개념과 동작 방식

Pipeline Syntax

```
triggers {  
  | pollSCM('*/*3 * * * *')  
}
```

2. 젠킨스의 기본 개념과 동작 방식

Pipeline Syntax

```
stage('Only for production') {  
  when {  
    branch 'production'  
    environment name: 'APP_ENV', value: 'prod'  
    anyOf {  
      environment name: 'DEPLOY_TO', value: 'production'  
      environment name: 'DEPLOY_TO', value: 'staging'  
    }  
  }  
}
```

You, 4 minutes ago • Uncommitted changes



2. 젠킨스의 기본 개념과 동작 방식

Pipeline Syntax

```
environment {  
    AWS_ACCESS_KEY_ID = credentials('awsAccessKeyId')  
    AWS_SECRET_ACCESS_KEY = credentials('awsSecretAccessKey')  
    AWS_DEFAULT_REGION = 'ap-northeast-2'  
    HOME = '.' // Avoid npm root owned  
}
```

2. 젠킨스의 기본 개념과 동작 방식

Steps

- Steps 내부는 여러가지 스텝들로 구성
- 여러 작업들을 실행가능
- 플러그인을 깔면 사용할 수 있는 스텝들이 생겨남

플러그인별 스텝 종류들

<https://www.jenkins.io/doc/pipeline/steps/>

2. 젠킨스

Pipeline

파라미터

작업



개

```
pipeline {
    agent any

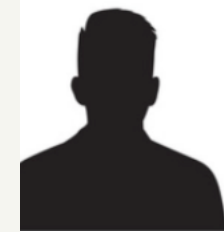
    stages {
        stage('Prepare') {
            steps {
                git url: 'https://github.com/frontalnh/pet-insurance.git',
                    branch: 'master',
                    credentialsId: 'jenkinsgit'
                sh 'ls'
                dir ('./docs'){
                    sh '''
                        aws s3 sync ./ s3://namhoontest
                    '''
                }
            }
        }

        post {
            // If Maven was able to run the tests, even if some of the test
            // failed, record the test results and archive the jar file.
            success {
                echo 'success'
            }
        }
    }

    stage('Build') {
        steps {
            echo 'Building...'
        }
    }
}
```

련의 플

정의함



자

2. 젠킨스의 기본 개념과 동작 방식

젠킨스 설치하기

```
yum update -y
```

```
# Jenkins 패키지 추가
```

```
sudo wget -O /etc/yum.repos.d/jenkins.repo http://pkg.jenkins.io/redhat/jenkins.repo  
&&
```

```
sudo rpm --import https://pkg.jenkins.io/redhat/jenkins.io.key
```

```
# Install java, docker, git
```

```
sudo yum install -y java-1.8.0-openjdk jenkins git docker
```

```
# 자바 버전 8 로 설정
```

```
alternatives --config java
```

```
service jenkins start
```

3. 개발 환경 및 CI/CD 의 기본 동작 이해

개발 환경의 종류에는 무엇이 있을까?

- 개발자가 개발을 하는 Local 환경(자신의 workspace 안에서 일함)
- 개발자들끼리 개발 내용에 대한 통합 테스트를 하는 Development 환경
- 개발이 끝나고 QA 엔지니어 및 내부 사용자들이 사용해 보기 위한 QA 환경
- 실제 유저가 사용하는 Production 환경
- 쉽게 DEV, QA, PROD 환경으로 부르자!

3. 개발 환경 및 CI/CD 의 기본 동작 이해

개발 프로세스

1. 개발자가 자신의 PC 에서 개발을 진행한다.
2. 다른 개발자가 작성한 코드와 차이가 발생하지 않는지 내부 테스트를 진행한다.
3. 진행한 내용을 다른 개발자들과 공유하기 위해 git 과 같은 SCM에 올린다.
=> 흔히 dev 브랜치
4. Dev브랜치의 내용을 개발 환경에 배포하기 전에 테스트와 Lint 등 코드 포매팅을 한다.
5. 배포하기 위한 빌드과정을 거친다.
6. 코드를 배포한다.
7. 테스트를 진행한다.
8. 위 모든 과정을 DEV, QA, PROD 환경에서 모두 하고 각각에 맞는 환경에 배포한다.

3. 개발 환경 및 CI/CD 의 기본 동작 이해

여러 배포 환경의 관리

여러 배포환경의 관리에서 핵심은

인프라를 모듈화 하여 어떤것이 변수인지 잘 설정하고 이를 잘 설계하는것!

가령 APP_ENV 처럼 현재 배포하고자 하는 것이 무슨 환경인지 설정하고
앱 내에서 사용하는 다양한 변수들을 APP_ENV 에 맞게 잘 가져다 쓰는것이 핵심.

서비스 내부의 변수 뿐만 아니라 클라우드 리소스를 많이 활용해서 개발하는 요즘에는 클라우드 리소스 내에서 인프라별 키관리가 매우 중요해서 aws system manager 의 parameter store 와 같은 키 관리 서비스를 쓰는것을 강추!

3. 개발 환경 및 CI/CD 의 기본 동작 이해

예시 배포 환경

1. 웹사이트 코드를 작성한다.
2. 웹사이트 코드를 린트, 웹팩 빌드 해서 AWS S3 bucket 에 html 파일을 업로드 한다
3. Node.js 백엔드 코드를 typescript 작성한다.
4. 위 코드를 javascript compile 하고, 테스트 코드를 돌려서 도커 이미지를 만들어 ECR 에 올린다.
5. 업로드한 ECR 이미지로 ECS 서비스를 재시작한다(rolling deploy) => continuouse deploy

3. 개발 환경 및 CI/CD 의 기본 동작 이해

AWS 리소스 간단 리뷰 – S3

- Simple Storage Service 의 약자로 그냥 클라우드 스토리지
- 정적 웹사이트 코드배포에 용이
- 정적 웹사이트 호스팅에 필요한 다양한 기능 제공
- AWS Cloudfront와 함께 사용해서 최적화 가능하고 DNS 관리도 가능

3. 개발 환경 및 CI/CD 의 기본 동작 이해

AWS 리소스 간단 리뷰 – ECR

- Elastic Container Registry
- 도커 이미지를 저장하는 프라이빗 레포지토리
- 실제 프로덕션 환경에서는 container 기반의 배포(ECS 등을 활용) 할 것이기 때문에 반드시 repository 가 있어야 함.

3. 개발 환경 및 CI/CD 의 기본 동작 이해

AWS 리소스 간단 리뷰 – ECS

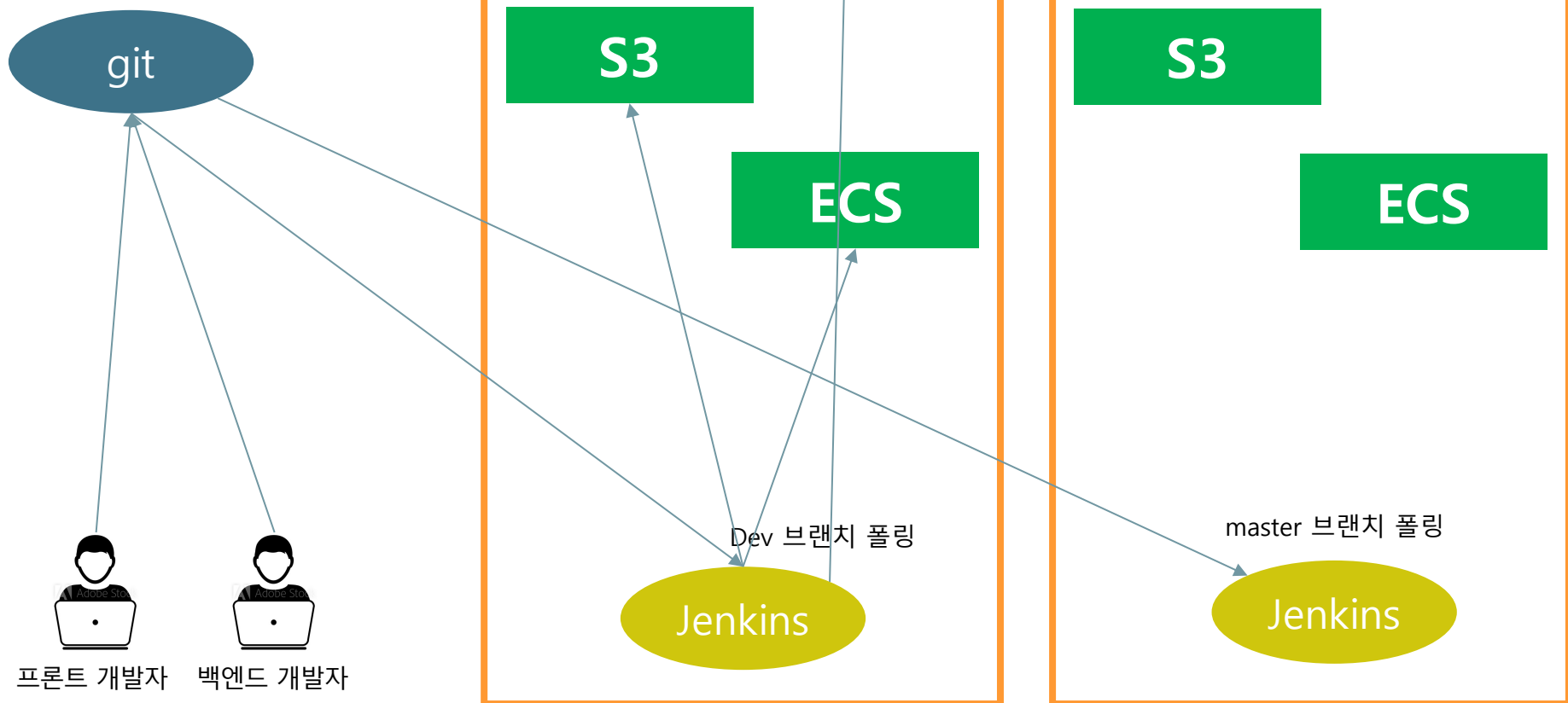
- Elastic Container Service
- 도커 컨테이너 기반으로 서비스 운용을 가능하게 해주는 간단한 서비스
- 무중단 배포(rolling update)를 제공하며 scale up 이 가능한 특징
- 백엔드 서비스를 스케일업 가능한 형태로 배포하는데 최적화
- 수많은 도커 컨테이너 서버를 띄우고 LB 가 이들 사이에 밸런싱을 해줌.
- fargate, ec2 모드가 있어서 docker container 리소스만 띄우거나 혹은 물리적인 EC2 instance 클러스터로 구성 가능)

⇒ ECS 혹은 k8s 등을 통해 rolling deploy가 처리되기 때문에 jenkins 의 역할은 배포 명령만 내려주면 된다!

Ex) aws ecs update-service '서비스 이름'

3. 개발 환경 및 CI/CD 의 기본 동작 이해

개발환경 예시1



3. 개발 환경 및 CI/CD 의 기본 동작 이해

개발환경 예시1



4. Jenkins 및 플러그인 설치 실습

젠킨스 설치하기

```
yum update -y
```

```
# Jenkins 패키지 추가
```

```
sudo wget -O /etc/yum.repos.d/jenkins.repo http://pkg.jenkins.io/redhat/jenkins.repo &&
```

```
sudo rpm --import https://pkg.jenkins.io/redhat/jenkins.io.key
```

```
# Install java, docker, git
```

```
sudo yum install -y java-1.8.0-openjdk jenkins git docker
```

```
# 자바 버전 8 로 설정
```

```
alternatives --config java
```

```
service jenkins start
```

4. Jenkins 및 플러그인 설치 실습

소스코드 다운받기

현장 강의에서는 GIT 을 통해 소스코드를 공유드렸으나

인터넷으로 수강하시는 분들은 첨부한 zip 파일을 다운로드 받아서 사용 부탁드립니다.

4. Jenkins 및 플러그인 설치 실습

젠킨스 접속하기 – MAC

```
ssh -i <AWS pem file 경로> ec2-user@<Jenkins 서버 IP>
```


4. Jenkins 및 플러그인 설치 실습

젠킨스 접속하기 – WINDOW

1. "선택적 기능 관리" 실행

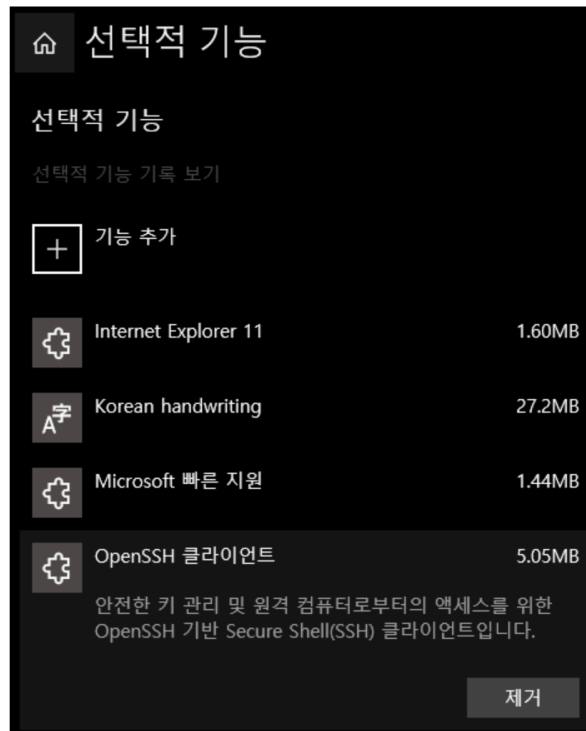


선택적 기능 관리

4. Jenkins 및 플러그인 설치 실습

젠킨스 접속하기 – WINDOW

2. "OpenSSH 클라이언트" 기능 추가



OpenSSH 클라이언트

위 절차가 끝나고 CMD 에서 아래 명령어로 접속하시면 됩니다.

```
ssh -i <AWS pem file 경로> ec2-user@<Jenkins 서버 IP>
```

4. Jenkins 및 플러그인 설치 실습

젠킨스 대시보드 확인하기

Curl 52.79.118.181:**8080**

```
sudo cat /var/lib/jenkins/secrets/initialAdminPassword
```

4. Jenkins 및 플러그인 설치 실습

Credential 관리하기

- Git 에서 Access Token 발급

The screenshot shows the Jenkins web interface. At the top is a black header with the Jenkins logo, a search bar, and the user 'namhoon lee' with a logout button. Below the header is a light gray navigation bar with 'Dashboard'. On the left is a sidebar with various links: '새로운 Item', '사람', '빌드 기록', '프로젝트 연관 관계', '파일 핑거프린트 확인', 'Jenkins 관리' (highlighted), 'My Views', 'Lockable Resources', 'New View', '빌드 대기 목록', and '빌드 실행 상태'. The main content area is titled 'Jenkins 관리' and contains several sections. The 'System Configuration' section includes '시스템 설정', 'Global Tool Configuration', and '노드 관리'. The 'Security' section is highlighted with a red box and includes 'Configure Global Security' (which is also highlighted with a red box), 'Manage Users', 'Manage Credentials', and 'Configure Credential Providers'. The 'Status Information' section includes '시스템 정보', 'System Log', and 'About Jenkins'. The '부하 통계' section is also visible at the bottom right.

Jenkins 관리

System Configuration

- 시스템 설정**
환경변수 및 경로 정보등을 설정합니다.
- Global Tool Configuration**
Configure tools, their locations and automatic installers.
- 플러그인 관리**
Jenkins의 기능을 확장하기 위한 플러그인을 추가, 제거, 사용, 미사용으로 설정할 수 있습니다.

Security


- Configure Global Security**
Secure Jenkins; define who is allowed to access/use the system.
- Manage Users**
Create/delete/modify users that can log in to this Jenkins
- Manage Credentials**
Configure credentials
- Configure Credential Providers**
Configure the credential providers and types

Status Information

- 시스템 정보**
문제 해결을 돕기위한 다양한 환경 정보를 보여줍니다.
- System Log**
System log captures output from java.util.logging output related to Jenkins.
- 부하 통계**
Check your resource utilization and see if you need more computers for your builds.

About Jenkins
See the version and license information.

4. Jenkins 및 플러그인 설치 실습


 **Jenkins**


검색


namhoon lee


로그아웃


Dashboard > Credentials


 새로운 Item


 사람


 빌드 기록


 프로젝트 연관 관계

 파일 링거프린트 확인

 Jenkins 관리

 My Views

 Lockable Resources

 New View

빌드 대기 목록 ^



빌드 대기 항목이 없습니다.

빌드 실행 상태 ^

1 대기 중



2 대기 중

Credentials


T	P	Store ↓	Domain	ID	Name
		Jenkins	(global)	jenkinsgit	jenkinsuser/*****

아이콘: S M L

Stores scoped to Jenkins

P	Store ↓	Domains
	Jenkins ▾	 (global)

4. Jenkins 및 플러그인 설치 실습


 **Jenkins**


검색


namhoon lee


로그아웃


Dashboard > Credentials > System >


 새로운 Item


 사람


 빌드 기록


 프로젝트 연관 관계


 파일 핑거프린트 확인


 Jenkins 관리

 My Views

 Lockable Resources

 New View

 **System**

Domain	Description
 Global credentials (unrestricted) ▼	Credentials that should be available irrespective of domain specification to requirements matching.

아이콘: S M L

빌드 대기 목록 ^


빌드 대기 항목이 없습니다.

빌드 실행 상태 ^

1 대기 중

2 대기 중

4. Jenkins 및 플러그인 설치 실습


 **Jenkins**


검색


namhoon lee

로그아웃



Dashboard > Credentials > System > Global credentials (unrestricted) >

 Back to credential domains

 Add Credentials

 **Global credentials (unrestricted)**

Credentials that should be available irrespective of domain specification to requirements matching.

ID	Name	Kind	Description
 jenkinsgit	jenkinsuser/*****	Username with password	

아이콘: S M L

4. Jenkins 및 플러그인 설치 실습

Credential 관리하기

- Git 에서 Access Token 발급

Kind

Username with password

Scope

Global (Jenkins, nodes, items, all child items, etc)

Username

Password

ID

Description

OK

5. CI/CD 파이프라인 구축 실습 및 QnA

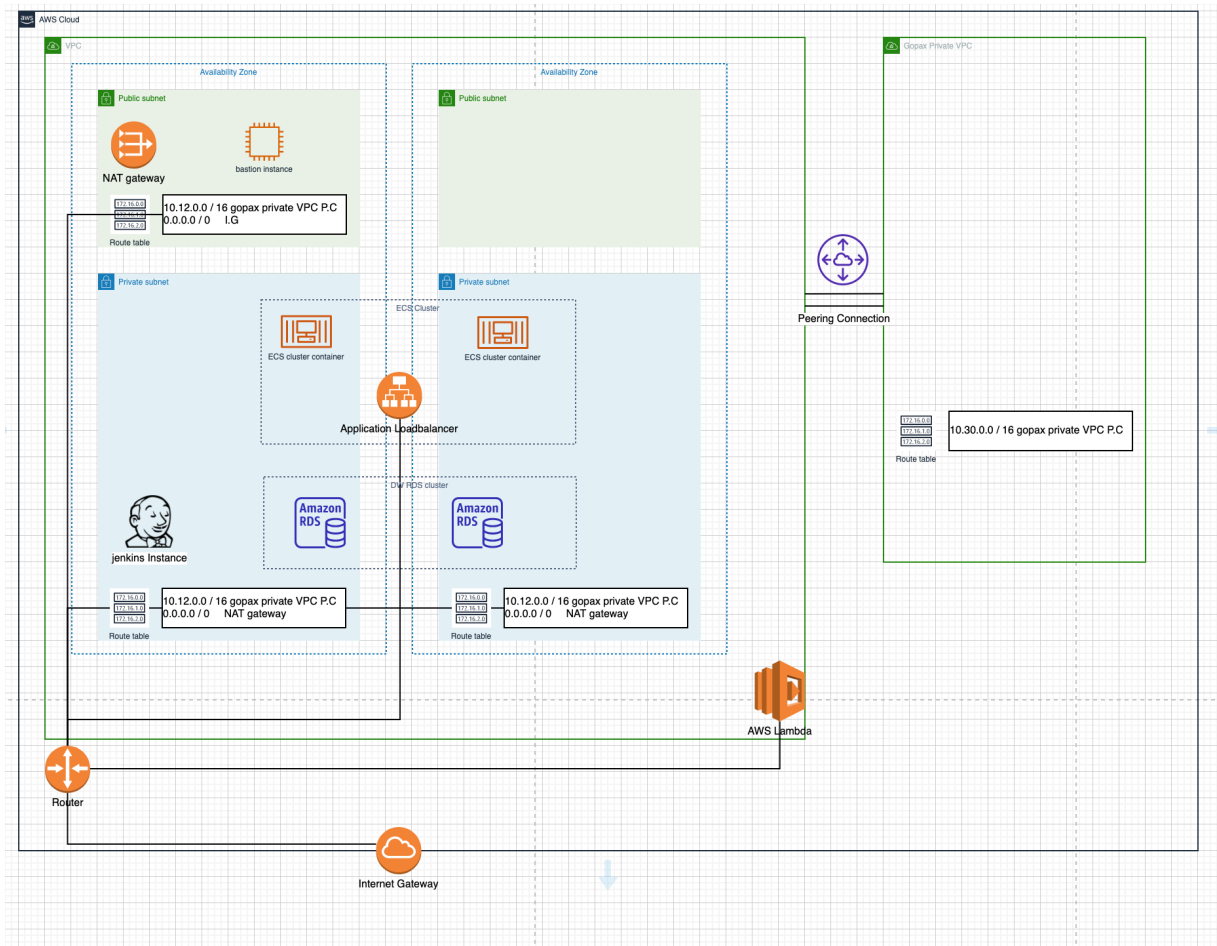
목표: git 에 코드가 머지되면 jenkins 에서 코드를 받아와 테스트 와 빌드를 거쳐서 docker 로 서버를 띄우고, s3 에 웹사이트를 배포한다.

1. S3 버킷 만들기
2. HTML 파일 작성하기
3. AWS credential 발급하기
4. Jenkins 에 aws, git credential 등록하기
5. Pipeline 작성하기

[최종 jenkinsfile](#)

6. 실제 사용 사례

ETL 파이프라인과 모든 리소스를 Cloud에 구축 했기 때문에 각종 클라우드 리소스까지 배포 자동화를 위해 IaaS 툴인 Terraform 사용 및 Dev, Qa, Prod 환경에서의 인프라 와 서비스 자동 배포 파이프라인
* 서버 등 운영 서비스뿐만 아니라 클라우드 리소스도 자동배포.



7. QnA

Q: 젠킨스 파이프 라인의 각 스테이지는 병렬로 처리되나요?

A: 젠킨스 파이프라인은 기본적으로 직렬로 처리되지만 개발자가 병렬로 처리가 필요하다고 생각되는 경우 병렬로 처리하도록 할 수 있습니다. 예를 들어 여러 젠킨스 노드가 있다면 각 노드에게 서로 다른 일을 시켜 일을 분산시킬 수 있습니다.

Q: 바스천이 있는 환경에서 Jenkins 를 어떻게 활용하나요?

A: 바스천을 통해 proxy ssh 를 활용해서 접근할 수 있습니다.

Thank's for
listening