```
1
 2
   -SpringBootRestfulMembership.zip -> 교육 중 학생들에게 배포하고자 하는 Spring Boot Code
 3
 4
   1. Jenkins?
 5
      1)지속적으로 통합 서비스(CI)를 제공해 주는 툴.
 6
      2)원래 허드슨 프로젝트로 개발.
 7
      3)Continuous Integration(CI) Tool라고도 부른다.
 8
      4)역할은 Github 혹은 Bitbucket에 올라가있는 Source를 Build해주고, 해당 Source를 Compile 해주어서
 9
      5)오류를 감지하고, 테스트 자동화와 맞물려 사용이 가능.
10
      6)개발 서버나 운영 서버에 배포까지 해주는 절차 가능.
11
      7)500여가지의 Plugin을 쉽게 install 할 수 있는 환경들을 제공.
12
13
14
   2. Jenkins의 장점
15
      1)Java뿐만이 아니라 다양한 환경에서의 build/test 환경을 맞출 수 있다.
16
      2)다양한 언어들을 지원하고, 설치해서 해당 Compile 환경들을 만들 수 있다.
17
      3)정적 코드 분석를 통해서 코딩 규약을 어겼는지를 체크할 수 있다.
18
      4)기본적으로는 다양한 테스트 환경에 대하여 배포작업을 연결할 수 있다.
19
      5)Build time을 비롯해서, 실행 시간의 변화등 성능 변화 또한 감지할 수 있다.
20
21
22
   3. 설치 방법
23
      1)Server에 직접 설치(Docker 등 활용)
24
      2)Local host에 설치(개인 컴퓨터)
25
26
27
   1. Git 작업
28
      1)압축 풀기 SpringBootRestfulMembership.zip
29
      2)STS에서 Import
30
      3)Local Mysql에서 CRUD 테스트
31
      4)Github에 Repository 생성하기
32
         -Repository name: springboot-restful-memebership
33
         -Description : Spring Boot □| Restful Membership
34
         -Public
35
         -README.md check
36
         -Choose a license check > MIT
37
38
      5)STS와 연동하기
39
         -STS의 Git Perspective
40
         -In Git Repositories view
41
         -Clone Git Repository
42
           --URI: https://github.com/gitinstructor/springboot-restful-membership.git
43
           --Authentication
44
             ---User : gitinstructor
45
             ---Password:
46
                --gitinstructor > Settings > Developer settings > Personal access tokens
47
                --Generate new token
48
           --Next
           --main check > Next > Finish
49
50
         -In Spring Perspective
51
           -- In Package Explorer View
52
             --Project > right-click > Team > Share Project
             -- Configure Git Repository
53
54
                ---Repository: 목록에서 선택 > Finish
55
           --Commit 하기
56
             --Project > right-click > Team > Commit
57
             --All file add to index > Commit Message : "Create Project" > Commit and Push > Close
58
             --Github에서 확인
59
60
61 2. AWS에서
62
      1)EC2 생성
63
         -Name: lab-ec2-db-xx
64
           --Instance type : t2.micro
65
           --보안그룹: launch-sg-00
           ---Port: 22, 3306, 8080, 80
66
67
        -Name: lab-ec2-was-xx
```

```
68
             -Instance type: t2.medium
 69
             --보안그룹: launch-sg-00
 70
 71
        2)Xshell 연결
 72
        3)MySQL 설치하기
 73
        4)MySQL 설치후, 작업
 74
          -In Command,
 75
             >mysql -h {{AWS EC2 MySQL Address}} -u root -p
 76
          -test database 생성
 77
             mysql> create database test character set utf8;
 78
          -scott 계정 생성
 79
             mysql> create user 'scott'@'%' identified by 'tiger';
 80
             mysql> grant all privileges on test.* to 'scott'@'%';
 81
             mysql> flush privileges;
 82
          -HeidiSQL에서 연결 테스트
 83
 84
        5)STS에서 AWS Mysql 연결테스트
 85
           -application.properties 수정
 86
             spring.datasource.url=jdbc:log4jdbc:mysql://{{AWS EC2 MySQL Address}}:3306/test
             spring.datasource.hikari.driver-class-name=net.sf.log4jdbc.sql.jdbcapi.DriverSpy
 87
             spring.datasource.hikari.jdbc-url=jdbc:log4jdbc:mysql://{{AWS EC2 MySQL
 88
             Address}}:3306/test
 89
             spring.datasource.hikari.username=scott
             spring.datasource.hikari.password=tiger
 90
          -add > commit -m "Modify application.properties file." > Command & Push
 91
 92
          -AWS와 연결된 HeidiSQL에서 membership.sql 실행할 것
 93
          -STS에서 Web Application 실행 및 테스트
 94
             --Tomcat 환경설정 변경
 95
                --port: 80
                --Web Modules : /
 96
 97
             --Maven Update
             --Pom.xml > Maven Install
 98
 99
100
101
     Task1. Localhost에 Jenkins 설치하기
102
     1. Jenkins Local Host에 설치하기
103
        1)https://www.jenkins.io/
        2)JDK 설치, Git 설치, Maven 설치
104
105
        3)Port: 8080
106
        4)services.msc > jenkins service Start 확인
        5)http://localhost:8080
107
108
        6)administrator password --> C:\ProgramData\Jenkins\.jenkins\secrets\initialAdminPassword
        7)Customize Jenkins -> Install suggested plugins
109
110
111
112
     2. Jenkins Global Tool Configuration
113
        1)설치 후 좌측 메뉴의 Jenkins 관리
        2)System Configuration > Global Tool Configuration 클릭
114
115
        3) Maven Configuration
116
          -Default settings provider
117
             --Settings file in filesystem 선택
                File path: C:\Program Files\apache-maven-3.8.3\conf\settings.xml
118
119
          -Default global settings provide
             ----Settings fi,le in filesystem 선택
120
121
                File path: C:\Program Files\apache-maven-3.8.3\conf\settings.xml
122
          -Apply Click
123
124
        4)JDK > JDK installations > Add JDK
125
          -Name: jdk11
126
           -Install automatically uncheck
127
           -JAVA HOME: C:\Program Files\Java\jdk-11.0.12
128
          -Apply click
129
130
        5)Git
131
          -Name: Default
           -Path to Git executable : C:\Program Files\Git\bin\git.exe
132
133
          -Apply click
```

```
134
135
       6)Maven
136
          -Maven installations > Add Maven
137
            --Name: maven
138
            --Install automatically uncheck
            --MAVEN_HOME: C:\Program Files\apache-maven-3.8.3
139
140
          -Apply click
141
       6)Save
142
143
144
     3. 플러그인 관리
145
       1)Jenkins 관리
146
       2)System Configuration > 플러그인 관리
147
       3)설치된 플러그인 목록 탭
148
          -git으로 검색하여 Git plugin과 GitHub plugin이 있는지 확인
149
       4)
150
          -설치 가능 탭으로 이동
151
          -"deploy to container plugin"으로 검색
152
          -[deploy to container plugin] 체크하고 [Install without restart] click
153
154
155
     4. Jenkins job이 GitHub의 Push Event 받기
156
        1)Github plugin은 jenkins가 Github의 Webhook을 받을 수 있도록 수동 모드와 자동 모드를 제공한다.
157
       2)수동 모드는 사용자가 Git Repository setting에서 Hook & Services를 직접 등록해 사용하는 것이고,
158
       3)자동 모드는 사용자의 GitHub OAuth 토큰을 이용해 Jenkins가 자동으로 Hooks & Service를 등록한다.
       4)자동으로 Webhook이 만들어지지 않는 경우 수동으로 Webhook을 만든다.
159
160
       5)payload URL: Jenkins 주소/github-webhook/
161
       6)GitHub OAuth 등록
162
          -GitHub에서 Settings 선택
          -Settings > Developer settings > Personal access tokens > Generate new token
163
164
          -Note: jenkins access token
165
          -Expiration: 30 days
166
          -Select scopes
167
            --repo
168
            --admin:repo_hook
169
            --delete repo
170
            --{{GitHub에서 발급받은 personal_access_token}}
171
172
       7)Jenkins > Jenkins 관리 > 시스템 설정
173
          -GitHub
            -GitHub Servers > Add GitHub Server
174
175
            -GitHub Server
176
               -- Name: My GitHub Server
177
               --API URL: 그대로
178
               --Credentials
179
                 --Add > Jenkins
180
                    ---Domain: Global crredentials
                    ---Kind : Secret text 선택
181
182
                    ---Scope : Global(Jenkins...)
183
                    ---Secret : {{GitHub에서 발급받은 personal_access_token}}
184
                    ---ID: accss-token
185
                    ---Add Click
                      --[Test connection] 클릭해서 연결 테스트
186
187
                         ---Credentials verified for user gitinstructor, rate limit: 4999
188
          -저장 클릭
189
190
191
     5. 새로운 Item 생성하기
192
       1)좌측 메뉴에서 새로운 Item 클릭
193
       2)Enter an item name: deployment demo
194
       3)Freestyle project 클릭 > OK 클릭
195
       4) GitHub project check
196
          Project url: https://github.com/gitinstructor/springboot-restful-membership/
197
       5)소스 코드 관리 > Git 선택
          -Repository URL: https://github.com/gitinstructor/springboot-restful-membership.git
198
          -Credentials > Add > Jenkins
199
            -- Domain: Global crredentials
200
```

```
201
             --Kind : SSH Usename with private key
202
             --ID: ssh-key
203
             --Username : ssh-key
204
             --Private Key > Enter directly check
205
               ---C:\Users\devex\.ssh\id_rsa의 내용을 에디터로 열어서 내용 복사 후
206
               ---[Add] 버튼 클릭하고 그 아래에 붙여넣는다.
207
             --Add click
208
          -방금 생성한 ssh-key에 맞춘다.
209
       6)Branches to build
210
          -Branch Specifier
211
             --*/main
       7)저장 버튼 클릭
212
213
       8)Jenkins 관리 > 시스템 설정
214
          -Jenkins Location
215
             -Jenkins URL: http://192.168.0.14:8080/
216
          -[저장] 클릭
217
218
       9)GitHub에 Deploy Key 등록하기
219
          -GitHub의 해당 Repository의 Settings > Deploy keys
220
          -Add deploy key click
221
             -Title: jenkins deploy key
222
               --C:\Users\devex\.ssh\id_rsa.pub의 내용을 에디터로 열어서 그 내용을 복사해서 붙여 넣는다.
223
224
             -Add key click
225
226
        10)GitHub의 해당 Repository의 Settings
227
          -Webhooks가 자동으로 등록되어 있는지 확인
228
          -만일 자동등록이 안되어 있으면 수동으로 해야 함.
229
          -[Add webhook] click
230
             --Payload URL: http://192.168.0.14:8080/github-webhook/
231
             -Content type : application/json
232
             -[Add webhook] click
233
234
        11)Jenkins > Dashboard > item 이름 > 구성 > 빌드 유발
235
          -[GitHub hook trigger for GITScm polling] check
236
          -Build section > Add build step 클릭
            --Invoke top-level Maven targets 선택
237
238
             --Maven Version : maven
239
             --Goals : clean package
240
             --[고급] 클릭
241
               ---POM: SpringBootRestfulMembership/pom.xml
242
          -[저장] 클릭
243
        12)Build Now click
244
245
246
247
     Task2. Docker Container로 설치하기
248 1. AWS에 Docker 설치
249
       1)$ sudo docker pull jenkins/jenkins
250
       2)$ sudo docker run -dp 8080:8080 --name jenkins -v /home/jenkins:/var/jenkins home -v
       /var/run/docker.sock:/var/run/docker.sock -u root jenkins/jenkins
251
252
253
     2. 필수 설치
254
        1)JDK 설치 확인
255
          $ which java
256
             --/opt/java/openjdk/bin/java
257
       2)Git 설치 확인
258
          $ which git
259
             --/usr/bin/git
260
261
       3)Maven 설치
262
          $ apt update
263
          $ apt install maven
264
          $ mvn -version
265
             Maven home: /usr/share/maven
266
             Java version: 11.0.13, vendor: Eclipse Adoptium, runtime: /opt/java/openjdk
```

```
Default locale: en, platform encoding: UTF-8
268
             OS name: "linux", version: "5.11.0-1022-aws", arch: "amd64", family: "unix"
269
          $ which mvn
270
             /usr/bin/mvn
271
272
273
     3. http://{{AWS EC2 Jenkins IP}}:8080
274
275
276
     4. Jenkins Global Tool Configuration
277
        1)설치 후 좌측 메뉴의 Jenkins 관리
278
        2)System Configuration > Global Tool Configuration 클릭
279
        3) Maven Configuration
280
          -기본값 그대로
281
282
        4)JDK > JDK installations > Add JDK
283
          -Name: jdk11
284
          -Install automatically uncheck
285
          -JAVA_HOME: /opt/java/openjdk
286
          -Apply click
287
288
        5)Git
289
          -Name: Default
290
          -Path to Git executable : /usr/bin/git
291
          -Apply click
292
293
        6)Maven
294
          -Maven installations > Add Maven
295
             --Name: maven
296
             --Install automatically uncheck
297
             --MAVEN_HOME : /usr/share/maven
298
          -Apply click
299
        7)Save
300
301
302
     5. 새로운 Item 생성하기
303
        1)좌측 메뉴에서 [새로운 Item] 클릭
304
        2)[Enter an item name] : demo
305
        3)[Freestyle project] 클릭 > [OK] 클릭
306
307
        4)소스 코드 관리 > Git 선택
308
          -Repository URL: https://github.com/gitinstructor/springboot-restful-membership.git
309
          -Credentials > Add > Jenkins
310
             -- Domain: Global credentials
             --Kind: Username with password
311
312
             --Scope : Global(Jenkins...)
313
             --Username : {{ Github ID}}
314
             --Password : {{GitHub에서 발급받은 Personal access token}}
315
             --ID: accss-token
316
             --Add click
          -방금 생성한 gitinstructor/****에 맞춘다.
317
318
319
        5)Branches to build
320
          -Branch Specifier
321
             --*/main
322
323
        6)[Apply] 클릭
324
325
        7)Build
326
          -Add build step 클릭
327
             --[Invoke top-level Maven targets] 선택
328
             --Goals : clean package
329
             --[고급] 클릭
330
                ---POM : SpringBootRestfulMembership/pom.xml
331
          -[저장] 클릭
332
333
        8)Build Now click
```

267

```
334
          Finished: SUCCESS 확인
335
336
       9)jenkins container에서 확인
337
          /var/jenkins_home/workspace/demo/SpringBootRestfulMembership/target에서 war 확인
338
339
340
     6. EC2 WAS에 Tomcat 설치하기
       1)openjdk 설치하기
341
342
          $ sudo apt update
343
          $ sudo apt install openjdk-11-jdk
344
          $ iava -version
          openjdk version "11.0.13" 2021-10-19
345
346
          OpenJDK Runtime Environment (build 11.0.13+8-Ubuntu-0ubuntu1.20.04)
347
          OpenJDK 64-Bit Server VM (build 11.0.13+8-Ubuntu-0ubuntu1.20.04, mixed mode, sharing)
348
349
       2)Tomcat 설치
350
          $ wget https://dlcdn.apache.org/tomcat/tomcat-9/v9.0.56/bin/apache-tomcat-9.0.56.tar.gz -P
          /tmp
351
          $ sudo mkdir /opt/tomcat
352
          $ sudo tar -xf /tmp/apache-tomcat-9.0.56.tar.gz -C /opt/tomcat/
353
          $ sudo In -s /opt/tomcat/apache-tomcat-9.0.56 /opt/tomcat/latest
354
355
          $ sudo nano /etc/systemd/system/tomcat.service
356
            [Unit]
357
            Description=Tomcat 9 servlet container
358
            After=network.target
359
360
            [Service]
361
            Type=forking
362
363
            Environment="JAVA HOME=/usr/lib/jvm/java-11-openjdk-amd64"
364
            Environment="JAVA OPTS=-Djava.security.eqd=file:///dev/urandom
            -Djava.awt.headless=true"
365
366
            Environment="CATALINA_BASE=/opt/tomcat/latest"
367
            Environment="CATALINA HOME=/opt/tomcat/latest"
368
            Environment="CATALINA_PID=/opt/tomcat/latest/temp/tomcat.pid"
            Environment="CATALINA_OPTS=-Xms512M -Xmx1024M -server -XX:+UseParallelGC"
369
370
371
            ExecStart=/opt/tomcat/latest/bin/startup.sh
372
            ExecStop=/opt/tomcat/latest/bin/shutdown.sh
373
374
            [Install]
375
            WantedBy=multi-user.target
376
377
          $ sudo systemctl daemon-reload
378
          $ sudo -i
379
          # nano /opt/tomcat/latest/conf/server.xml
380
            Port를 80으로 변경
381
          #exit
          $ sudo systemctl enable --now tomcat
382
383
          $ sudo systemctl status tomcat
384
          -EC2 보안그룹에서 port 80 오픈
385
          -브라우저에서 확인
386
            http://{{ AWS EC2 Tomcat Domain Name}}
387
388
389
     7. 필요한 Plugin 설치
390
       1)Jenkins 관리 > System Configuration > 플러그인 관리 > 설치 가능 탭
391
          -"deploy to container plugin"으로 검색
392
          -[deploy to container plugin] 체크하고 [Install without restart] click
393
       2)설치 페이지에서 [성공] 확인
394
       3)다시 Jenkins 관리 > Tools and Actions > Reload Configuration from Disk > 확실합니까? > 확인
395
396
397
     8. 빌드 후 조치
398
       1)maven을 통해 빌드 후 패키징됐다면 패키징된 .war 파일을 톰캣이 구동중인 원격 서버에 배포되어야한다. 아래와 같이
```

```
399
          -해당 프로젝트 > 구성 > 빌드 후 조치 > 빌드 후 조치 추가 > Deploy war/ear to a container 선택
400
401
       2)WAR/EAR files
402
          -실제 빌드시 /var/lib/jenkins/workspace 디렉토리에 .war 파일이 생성되는데 job 실행시 해당 .war를 가져올 수
          있도록 경로를 입력한다.
          -**/*.war
403
404
405
       3)Context path
406
          -배포시 사용할 컨텍스트를 지정한다.
407
408
409
       5)Containers
410
          -Tomcat 9.x Remote 선택
411
          -Tomcat URL:
412
            -배포되는 원격 서버 URL을 입력한다. http://{host}:{tomcat port}
413
            http://{{ AWS EC2 Tomcat Domain Name}}
414
415
          -Credentials: jenkins 서버가 원격서버에 배포할 수 있게 원격서버측에서 jenkins 서버 접근을 허용해줘야 한다.
416
            1)[Jenkins 서버 접속 허용]
417
               -톰캣이 구축되어있는 원격서버에서 아래 경로의 파일 내용을 수정한다.
418
              -/opt/tomcat/latest/webapps/manager/META-INF/context.xml
419
                 <Valve className="org.apache.catalina.valves.RemoteAddrValve"
420
                   allow="127..d+..d+..d+|::1|0:0:0:0:0:0:0:1|{Jenkins Container's IP}"/>
421
422
            2)[톰캣 원격 서버 접속 계정 설정]
423
               -톰캣이 구축되어있는 원격서버에서 아래 경로의 파일 내용을 수정하여 jenkins 서버에서 접속시 사용되는
              username, password를 설정해준다.
424
               -{tomcat path}/conf/tomcat-users.xml
425
                 <role rolename="manager-script" />
426
                 <user username="admin" password="javatomcat" roles="manager-script" />
427
428
            3)다시 jenkins 설정으로 돌아와 Containers > Credentials > Add > Jenkins
429
               -Kind: Username with password
430
              -Username: admin
431
              -Password: javatomcat
432
              -Add click
433
434
          -Credentials : admin/****로 선택
435
436
       6)저장 클릭
437
       7)Build Now 클릭 후 Build Success 확인하고 웹 브라우저에서 확인할 것
438
439
440
    Task3. Pipeline 만들기
441
     1. 새로운 Item
442
       1)Enter an item name: pipeline demo
443
       2)Pipeline 선택 후 OK
444
445
     2. Pipeline tab
446
       1)Definition: Pipeline script 선택
447
       2)[Use Groovy Sandbox] uncheck
448
       3)Script에 다음과 같이 코딩
449
          pipeline {
450
            agent any
451
452
            tools {
453
              maven "maven"
454
            }
455
456
            stages {
              stage("git Pull") {
457
                 steps {
458
459
460
                 }
461
              }
462
            }
```

[빌드 후 조치] > [Deploy war/ear to a container]를 선택한다.

```
}
463
464
465
        4)git Pull Stage
466
           -[Pipeline Syntax] click
467
             -Steps
468
                --Sample Step > git: Git 선택
469
470
                --Repository URL: https://github.com/gitinstructor/demo.git
471
                --Branch: main
472
                --Credentials : gitinstructor/**** 선택
473
                --두개 체크박스 모두 uncheck
474
                --[Generate Pipeline Script] 클릭
475
                --아래 코드를 steps 사이에 넣고 저장
476
                  git branch: 'main', changelog: false, credentialsId: 'access-token1', poll: false, url:
                  'https://github.com/gitinstructor/demo.git'
477
             -Build Now로 테스트
478
479
        5)Build Stage
480
          stage('Build') {
481
             steps {
482
                sh "mvn -Dmaven.test.failure.ignore=true -N -f SpringBootRestfulMembership/pom.xml
                clean package"
483
             }
484
485
           -저장 후 Build Now로 테스트
486
487
        6)Deploy Stage
488
          -[Pipeline Syntax] click
489
490
                --Sample Step > deploy: Deploy war/ear to a container
491
                --deploy
492
                  ---WAR/EAR files: **/*.war
                  ---Context path : /
493
494
                  ---Containers
495
                     Tomcat 9.x Remote
496
                     Tomcat URL : {{ AWS EC2 Domain Name}}
497
                     Credentials: admin/*****로 선택
498
                -[Deploy on failure] uncheck
499
          -[Generate Pipeline Script] 클릭
500
             deploy adapters: [tomcat9(credentialsId: '844eae6f-190e-4d14-a7cb-f34e9dcf35d1', path: ",
             url: '{{ AWS EC2 Domain Name}}')], contextPath: '/', onFailure: false, war: '**/*.war'
501
502
        7)Restart Stage
503
           -[Pipeline Syntax] click
504
             -Steps
505
                --Sample Step > sh: Shell Script 선택
506
                --Shell Script
507
                  다음 2줄을 넣는다.
508
                  여기서 admin은 아이디, javatomcat은 패스워드
509
                  curl -u admin:javatomcat {{ AWS EC2 Domain Name}}/host-manager/text/stop
                  curl -u admin:javatomcat h{{ AWS EC2 Domain Name}}/host-manager/text/start
510
511
             --[Generate Pipeline Script] 클릭
512
                sh "'curl -u admin:javatomcat {{ AWS EC2 Domain Name}}/host-manager/text/stop
513
                  curl -u admin:javatomcat {{ AWS EC2 Domain Name}}/host-manager/text/start'''
514
515
        8)전체 코드
516
           pipeline {
517
             agent any
518
519
             tools {
520
                maven "maven"
521
             }
522
523
             stages {
                stage("git Pull") {
524
525
526
                     git branch: 'main', changelog: false, credentialsId: 'access-token1', poll: false, url:
```

```
'https://github.com/gitinstructor/demo.git'
527
                  }
528
               }
529
530
               stage('Build') {
531
                  steps {
532
                    sh "mvn -Dmaven.test.failure.ignore=true -N -f
                    SpringBootRestfulMembership/pom.xml clean package"
533
                  }
534
               }
535
536
               stage("Deploy") {
537
                  steps {
                    deploy adapters: [tomcat9(credentialsId: '844eae6f-190e-4d14-a7cb-f34e9dcf35d1',
538
                    path: ", url: '{{ AWS EC2 Domain Name}}')], contextPath: '/', onFailure: false, war:
                    '**/*.war'
539
                  }
               }
540
541
542
               stage("RESTART") {
543
                  steps {
544
                    sh "'curl -u admin:javatomcat {{ AWS EC2 Domain Name}}/host-manager/text/stop
545
          curl -u admin:javatomcat {{ AWS EC2 Domain Name}}/host-manager/text/start'''
546
547
               }
548
             }
549
          }
550
551
       9)저장 후 Build Now 클릭
552
        10)테스트
553
          -src/main/webapp/view.html을 /static/으로 이동 후 commit
554
          -index.html에 image 추가
555
556
557
     3. Jenkins job이 GitHub의 Push Event 받기
558
        1)Github plugin은 jenkins가 Github의 Webhook을 받을 수 있도록 수동 모드와 자동 모드를 제공한다.
559
        2)수동 모드는 사용자가 Git Repository setting에서 Hook & Services를 직접 등록해 사용하는 것이고,
560
       3)자동 모드는 사용자의 GitHub OAuth 토큰을 이용해 Jenkins가 자동으로 Hooks & Service를 등록한다.
561
       4)자동으로 Webhook이 만들어지지 않는 경우 수동으로 Webhook을 만든다.
562
       5)payload URL: Jenkins 주소/github-webhook/
563
       6)GitHub OAuth 등록
564
          -GitHub에서 Settings 선택
565
          -Settings > Developer settings > Personal access tokens > Generate new token
566
          -Note : jenkins_access_token
567
          -Expiration: 30 days
568
          -Select scopes
569
             --repo
570
             --admin:repo_hook
571
             --delete_repo
572
             --{{GitHub에서 발급받은 personal access token}}
573
       7)Jenkins > Jenkins 관리 > 시스템 설정
574
575
          -GitHub
576
             -GitHub Servers > Add GitHub Server
577
             -GitHub Server
               --Name: My GitHub Server
578
               --API URL: 그대로
579
580
               --Credentials
581
                  --Add > Jenkins
582
                    --- Domain: Global crredentials
583
                    ---Kind : Secret text 선택
584
                    ---Scope : Global(Jenkins...)
585
                    ---Secret : {{GitHub에서 발급받은 personal_access_token}}
586
                    ---ID: accss-token
587
                    ---Add Click
588
                       --[Test connection] 클릭해서 연결 테스트
589
                          ---Credentials verified for user gitinstructor, rate limit: 4999
```

590 591	-저장 클릭
592	8)GitHub의 해당 Repository의 Settings
593	-Webhooks가 자동으로 등록되어 있는지 확인
594	-만일 자동등록이 안되어 있으면 수동으로 해야 함.
595	-[Add webhook] click
596	Payload URL: http://192.168.0.14:8080/github-webhook/
597	-Content type: application/json
598	-[Add webhook] click
599	
600	9)GitHub의 해당 Repository의 Settings
601	-Webhooks가 자동으로 등록되어 있는지 확인
602	-만일 자동등록이 안되어 있으면 수동으로 해야 함.
603	-[Add webhook] click
604	Payload URL : {{ AWS EC2 Domain Name }}/github-webhook/
605	-Content type : application/json
606	-[Add webhook] click