DIA

(Directing Assistance)

빌드 및 배포

: 2022.10.11 ~ 2022.11.21

대전 자율 3반

권혁림 강승훈 김진산 윤영훈 이슬기 조경민

1. 기술스택

|  |  |  |  |
| --- | --- | --- | --- |
| 구분 | 기술스택 | 상세내용 | 버전 |
| 공통 |  |  |  |
|  | 형상관리 | GitLab |  |
|  | 이슈관리 | Jira |  |
|  | 커뮤니케이션 | Mattemost |  |
|  | 커뮤니케이션 | Notion |  |
| 서버 |  |  |  |
|  | 서버 | AWS EC2 |  |
|  | 플랫폼 | Ubuntu | 20.04 |
|  | 배포 | Docker | 20.10.20 |
|  | 배포 | Docker Compose | 1.29.2 |
|  | 배포 | Jenkins | 2.361.2 |
|  | 배포 | Nginx | 1.18.0 |
| BackEnd |  |  |  |
|  | DB | MySQL | 8..0.30 |
|  | Cache Storage | Redis | 7-alpine |
|  | Spring boot |  | 2.7.4 |
|  | QueryDSL |  | 5.0.0 |
| Local |  |  |  |
|  | Python |  | 3.10 |
|  | Numpy |  | 1.23.3 |
|  | Pandas |  | - |
|  | Anaconda |  | - |
|  | Pytorch |  | - |
| FrontEnd | HTML5 |  |  |
|  | CSS3 |  |  |
|  | JavaScript(ES6) |  |  |
|  | React |  | 18.2.0 |
|  | Build | Node | 16.15.0 |
| Android, WearOS | Android Studio |  | - |
|  | Kotlin |  | 1.7.20 |
| IDE | Visual Studio Code | 1.70.0 |  |

1. 포트번호

|  |  |
| --- | --- |
| 구분 | 포트번호 |
| Jenkins | 8080 |
| Spring boot | 8081 |
| React | 8082 |
| MySQL | 3306 |
| Redis | 6373 |

1. DokerFile
2. Spring

|  |
| --- |
| FROM openjdk:8-jdk-alpine  RUN addgroup -S seungh1024 && adduser -S seungh1024 -G seungh1024  USER seungh1024:seungh1024  ARG JAR\_FILE=build/libs/\*.jar  COPY ${JAR\_FILE} app.jar  ENTRYPOINT ["java","-jar","/app.jar"] |

1. React

|  |
| --- |
| FROM node:16.15.0 as build-stage  WORKDIR /app  COPY package\*.json ./  COPY package-lock.json ./  RUN npm install  COPY . .  RUN npm run build  FROM nginx:stable-alpine as production-stage  RUN rm -rf /etc/nginx/conf.d/default.conf  COPY ./nginx/default.conf /etc/nginx/conf.d/default.conf  RUN rm -rf /usr/share/nginx/html/\*  COPY --from=build-stage /app/build /usr/share/nginx/html  EXPOSE 8082  CMD ["nginx", "-g","daemon off;"] |

1. Docker-Compose File

|  |
| --- |
| 1. version: "3.8" 2. services: 3. react: 4. container\_name: dia-react 5. build: ./frontend/ 6. restart: on-failure 7. volumes: 8. - ./frontend/nginx/:/etc/nginx/conf.d/ 9. ports: 10. - 8082:8082 11. redis: 12. container\_name: redis 13. image: redis:7-alpine 14. ports: 15. - 6379:6379 16. # command: redis-server --save 20 1 --loglevel warning --requirepass eYVX7EwVmmxKPCDmwMtyKVge8oLd2t81 17. volumes: 18. - /redis/data:/data 19. - /redis/conf/redis.conf:/usr/local/conf/redis.conf 20. mysqldb: 21. container\_name: mysql 22. image: mysql:latest 23. restart: unless-stopped 24. environment: 25. - MYSQL\_ROOT\_PASSWORD=rnjsgurfla 26. - MYSQL\_DATABASE=ssafy 27. - MYSQL\_USER=ssafy 28. - MYSQL\_PASSWORD=rnjsgurfla 29. ports: 30. - 3306:3306 31. command: --default-authentication-plugin=mysql\_native\_password 32. # - --lower\_case\_table\_names=1 33. volumes: 34. - ./mysqldata:/var/lib/mysql 35. spring: 36. container\_name: dia-spring 37. depends\_on: |

1. Jenkins file

|  |
| --- |
| 1. pipeline { 2. agent any 3. stages { 4. stage('Prepare') { 5. steps { 6. sh 'echo "Clonning Repository"' 7. git branch: 'release', 8. url: 'https://lab.ssafy.com/s07-final/S07P31B307.git', 9. credentialsId: 'DIA' 10. } 11. post { 12. success { 13. sh 'echo "Successfully Cloned Repository"' 14. } 15. failure { 16. sh 'echo "Fail Cloned Repository"' 17. } 18. } 19. } 20. stage('Docker stop'){ 21. steps { 22. // sh 'echo "root ubuntu" | sudo -S ls' 23. // sh 'ls' 24. sh 'sudo chmod -R 777 /usr/local/bin' 25. sh 'sudo chmod +x /usr/local/bin/docker-compose' 26. sh 'echo "Docker Container Stop"' 27. // 도커 컴포즈 다운 28. // sh 'curl -L https://github.com/docker/compose/releases/download/1.29.2/docker-compose-$(uname -s)-$(uname -m) -o /usr/local/bin/docker-compose' 29. // 해당 도커 컴포즈 다운한 경로로 권한 설정 30. // sh 'chmod -R 777 /usr/local/bin' 31. // sh 'chmod +x /usr/local/bin/docker-compose' 32. // 기존 백그라운드에 돌아가던 컨테이너 중지 33. sh 'sudo docker-compose stop' 34. } 35. post { 36. failure { 37. sh 'echo "Docker Fail"' 38. } 39. } 40. } 41. stage('RM Docker'){ 42. steps { 44. sh 'echo "Remove Docker"' 45. //정지된 도커 컨테이너 찾아서 컨테이너 ID로 삭제함 46. sh ''' 47. result=$(sudo docker container ls -a --filter "name=dia\*" -q ) 48. if [ -n "$result" ] 49. then 50. sudo docker rm $(sudo docker container ls -a --filter "name=dia\*" -q) 51. else 52. echo "No such containers" 53. fi 54. ''' 55. // dia로 시작하는 이미지 찾아서 삭제함 56. sh ''' 57. result=$(sudo docker images -f "reference=dia\*" -q ) 58. if [ -n "$result" ] 59. then 60. sudo docker rmi -f $(sudo docker images -f "reference=dia\*" -q) 61. else 62. echo "No such container images" 63. fi 64. ''' 65. // 안쓰는이미지 -> <none> 태그 이미지 찾아서 삭제함 66. sh ''' 67. result=$(sudo docker images -f "dangling=true" -q) 68. if [ -n "$result" ] 69. then 70. sudo docker rmi -f $(sudo docker images -f "dangling=true" -q) 71. else 72. echo "No such container images" 73. fi 74. ''' 75. } 76. post { 77. failure { 78. sh 'echo "Remove Fail"' 79. } 80. } 81. } 82. stage('Build Gradle'){ 83. steps{ 84. dir('backend') { 85. sh "sudo chmod +x gradlew" 86. sh """ 87. sudo ./gradlew clean build --exclude-task test 88. """ 89. } 90. } 91. post{ 92. failure{ 93. sh 'echo "Build Gradle Fail"' 94. } 95. } 96. } 97. stage('Bulid & Run') { 98. steps { 99. sh 'echo " Image Bulid Start"' 100. script { 101. // 업데이트된 코드로 빌드 및 실행 102. sh 'sudo docker-compose up -d' 103. sh 'sudo chmod 777 -R /home/ubuntu/profile' 104. sh 'sudo chmod 777 -R /var/lib/docker' 105. sh 'sudo chmod 777 -R /var/lib/docker/containers/\*' 106. } 108. } 109. post { 110. failure { 111. sh 'echo "Bulid Docker Fail"' 112. } 113. } 114. } 116. } 117. } |

1. 프로퍼티 정의

Spring src/main/resources/properties/env.properties

|  |
| --- |
| # mysql  mysql.url=jdbc:mysql://localhost:3306/test?serverTimezone=UTC&characterEncoding=UTF-8  mysql.username=root  mysql.password=root  # jwt secret  env.jwt.secret=c2lsdmVybmluZS10ZWNoLXNwcmluZy1ib290LWp3dC10dXRvcmlhbC1zZWNyZXQtc2lsdmVybmluZS10ZWNoLXNwcmluZy1ib290LWp3dC10dXRvcmlhbC1zZWNyZXQK  # redis  redis.host = 127.0.0.1  # mail  mail.host = smtp.naver.com  mail.username = 네이버아이디  mail.userpassword= 네이버비밀번호  mail.port= 465  mail.useremail = 네이버이메일 |