|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **□ 수행평가 - 빅데이터를 활용한 IoT 시스템 개발(feat.커넥티드카)** | | | | | | |
|  |  |  | |  | |  |
| **과정명** | | 빅데이터를 활용한 IoT 시스템 개발(feat.커넥티드카) | | | | |
| **교과목명** | | 빅데이터저장및분석시스템구축기술 | | **훈련교사** | | 이진만 |
| **과정명** | | Linux OS 이해  빅데이터 수집시스템 개발  빅데이터 저장시스템 개발  빅데이터 분석시스템 개발 | | | | |
| **수행날짜** | | 2019.08.09 | 훈련생명 | | 이경헌 | |
| **과제개요** | | | | | | |
| 1. Linux OS 를 설치 하고 빅데이터 시스템 환경을 구축 하시오 2. 빅데이터 수집 시스템을 구축 하시오 (log4j 이용) 3. 빅데이터 저장 시스템을 구축 하시오 (MariaDB, Hadoop 이용) 4. 빅데이터 분석 시스템을 구축 하시오 (Hive 이용) 5. 기존 Spring MVC와 빅데이터 시스템을 통합 하시오 | | | | | | |
| 시스템 구성도 | | | | | | |
| 작성 내용  **1. 가상머신 (VMware Workstation 15 Player)에 CentOS 7 을 설치**  VMware의 Virtual Network Editor를 이용해서 VMnet8(type이 NAT)의 서브넷 마스크를 자신의 환경에 맞게 설정하고, VMware 설정에서 CD롬에 준비해 놓은 CentOS 7 시디 이미지파일을 마운트 시킨다.  부팅하면 GUI로 되어있는 설치환경이 실행된다. 소프트웨어 선택에서 ‘개발 및 창조를 위한 워크스테이션’을 선택한다. 네트워크 설정에 들어가서 랜을 ON 한다, 키보드에서 한국어와 영어가 잘 선택되어 있는지 확인한다. 설치대상에서 드라이브를 고른 후, 표준파티션에서 swap을 설정하는데, 보통 RAM의 2배 정도되는 크기를 설정한다. 그리고 나머지는 마운트 지점을 /로 표시하고 확인을 선택하고 설치를 진행한다.  설치 진행 중 사용자 생성에서 root 암호를 설정하고, 사용자 생성에서 원하는 사용자를 생성한다.  시간이 지나면 설치가 완료되고, 로그인 할 때는 root계정으로 로그인한다.    설치완료.  **2. 빅데이터 수집시스템 구축 → Log4j를 이용** 기존의 웹서비스 서버에서 log를 생성하도록 수정한다.  web.xml 에 listener 를 추가  ```xml  <listener>  <listener-class>org.springframework.web.util.Log4jConfigListener</listener-class>  </listener>  <context-param>  <param-name>log4jConfigLocation</param-name>  <param-value>/WEB-INF/config/log4j.properties</param-value>  </context-param>  ```  - pom.xml 에서 org.apache.log4j 패키지 다운  ```xml  <dependency>  <groupId>log4j</groupId>  <artifactId>log4j</artifactId>  <version>1.2.17</version>  <exclusions>  <exclusion>  <groupId>javax.mail</groupId>  <artifactId>mail</artifactId>  </exclusion>  <exclusion>  <groupId>javax.jms</groupId>  <artifactId>jms</artifactId>  </exclusion>  <exclusion>  <groupId>com.sun.jdmk</groupId>  <artifactId>jmxtools</artifactId>  </exclusion>  <exclusion>  <groupId>com.sun.jmx</groupId>  <artifactId>jmxri</artifactId>  </exclusion>  </exclusions>  </dependency>  ```  - Log 속성을 위한 properties 생성  ```properties  ###############################################################################  #  # log4j  #  ###############################################################################  log4j.logger.user = DEBUG, console, user  log4j.logger.work = DEBUG, console, work  log4j.logger.data = DEBUG, console, data  # Console output...  log4j.appender.console= org.apache.log4j.ConsoleAppender  log4j.appender.console.layout = org.apache.log4j.PatternLayout  log4j.appender.console.layout.ConversionPattern = [%d] %-5p %L %m%n  # user  log4j.appender.user.Threadhold=DEBUG  log4j.appender.user = org.apache.log4j.DailyRollingFileAppender  log4j.appender.user.DatePattern = '.'yyyy-MM-dd  log4j.appender.user.layout = org.apache.log4j.PatternLayout  log4j.appender.user.layout.ConversionPattern = %-5p %L [%d] - %m%n  log4j.appender.user.File = c:/logs/user.log  # work  log4j.appender.work.Threadhold=DEBUG  log4j.appender.work = org.apache.log4j.DailyRollingFileAppender  log4j.appender.work.DatePattern = '.'yyyy-MM-dd  log4j.appender.work.layout = org.apache.log4j.PatternLayout  log4j.appender.work.layout.ConversionPattern = %-5p , %L , %d , %m%n  log4j.appender.work.File = c:/logs/work.log  # data  log4j.appender.data.Threadhold=DEBUG  log4j.appender.data = org.apache.log4j.DailyRollingFileAppender  log4j.appender.data.DatePattern = '.'yyyy-MM-dd  log4j.appender.data.layout = org.apache.log4j.PatternLayout  log4j.appender.data.layout.ConversionPattern = %5p [%d{MMdd HHmmss}] %F:%L:%M - %m%n  log4j.appender.data.File = c:/logs/data.log  ```  - AOP 가 적용된 java 파일 추가  ```java  @Service  @Aspect  public class Loggers {  private Logger work\_log =  Logger.getLogger("work");  private Logger user\_log =  Logger.getLogger("user");  private Logger data\_log =  Logger.getLogger("data");  // before  @Before("execution(\* com.hw.controller..\*Controller.\*(..))")  public void logging(JoinPoint jp) {  work\_log.debug(jp.getSignature().getName());  //work\_log.debug(jp.getArgs()[0].toString());  }  ```  - spring.xml 에 aop 추가!!!!!  ```xml  <aop:aspectj-autoproxy></aop:aspectj-autoproxy>  ```  ## IV. Listener  - 요청이 들어올 때의 처리는 어떻게 하는걸까?  - web.xml 에 필요한 listener 추가  ```xml  <listener>  <listener-class>org.springframework.web.context.ContextLoaderListener</listener-class>  </listener>  <context-param>  <param-name>contextConfigLocation</param-name>  <param-value>/WEB-INF/config/springex.xml</param-value>  </context-param>  ```  - springex.xml 파일 추가  ```xml  <bean id="exeptionResolver"  class="org.springframework.web.servlet.handler.SimpleMappingExceptionResolver">  <property name="exceptionMappings">  <props>  <prop key="org.springframework.dao.DuplicateKeyException">  error  // error.jsp 로 간다.  </prop>  // 여기에 exception 들을 써주면서 처리할 수 있다.  </props>  </property>  </bean>  ```  3. Hadoop 설치  **3. Hadoop 설치**  Linux 서버 준비 및 필요 파일(Java, Tomcat, Eclipse, mysql-MariaDB, Hadoop) 다운  **3-1. Hadoop 설치**   * setting에서 network는 'nat'으로 설정하기 → IP 할당받기 위함 * firewall 설정 변경   systemctl stop firewalld  systemctl disable firewalld   * Hadoop 파일 다운 및 압축 풀기 : [hadoop-1.2.1.tar.gz](https://archive.apache.org/dist/hadoop/common/hadoop-1.2.1/%5Bhadoop-1.2.1.tar.gz)   tar xvf hadoop-1.2.1.tar.gz → file 디렉토리에 있는 Hadoop 파일 압축 풀기  cp -r hadoop-1.2.1 /etc → 압축 푼 파일을 /etc로 복사   * /etc/pforile 파일에 환경변수 등록 → Tomcat 아래에 추가   HADOOP\_HOME=/etc/hadoop-1.2.1  export JAVA\_HOME CLASSPATH TOMCAT\_HOME HADOOP\_HOME  PATH=.:$JAVA\_HOME/bin:$TOMCAT\_HOME/bin:$HADOOP\_HOME/bin:$PATH   * hostname 변경   hostnamectl set-hostname hadoopserver1   * /etc/hosts : hadoopserver1만 있도록 함 * /etc/sysconfig/network-scripts/ifcfg-ens33 에서 IP 주소 확인   **3-2. SSH 설정**  computer SSH(Secure Shell)란? 네트워크 상 다른 컴퓨터에 로그인하거나 원격 시스템에서 명령을 실행할 수 있게 함   * private key와 public key 만듦 (두 개의 키가 쌍으로 생성)   ssh-keygen -t dsa -P '' -f ~/.ssh/id\_dsa  heavy_check_mark ​ls -a를 통해 '.ssh'가 생성되었는지 확인  heavy_check_mark ​id\_dsa (private key) / id\_dsa.pub (public key)   * '.ssh'에 접속하여 public key 등록   cat id\_dsa.pub >> authorized\_keys   * 비밀번호 없이 로그인이 가능한지 확인   ssh hadoopserver1 → 바로 'login' 메시지가 떠야 됨 / exit로 로그아웃  **2-3. Hadoop conf 설정**   * /etc/hadoop-1.2.1/conf/core-site.xml   HDFS와 MapReduce에서 공통적으로 사용할 환경정보를 설정  <configuration>  <property>  <name>fs.default.name</name>  <!-- HDFS의 기본 이름을 의미함 / URL 형태로 사용 -->  <value>hdfs://70.12.114.206:9000</value>  </property>  <property>  <name>hadoop.tmp.dir</name>  <!-- 임시 데이터 저장을 위한 공간 -->  <value>/etc/hadoop-1.2.1/tmp</value>  </property>  </configuration>   * /etc/hadoop-1.2.1/conf/hdfs-site.xml   HDFS에서 사용할 환경 정보를 설정  <configuration>  <property>  <name>dfs.permissions</name>  <value>false</value>  </property>  <property>  <name>dfs.replication</name>  <value>2</value>  <!-- 복제본 두 개씩 -->  </property>  <property>  <name>dfs.http.address</name>  <value>70.12.114.206:50070</value>  <!-- namenode의 IP 주소 -->  </property>  <property>  <name>dfs.secondary.http.address</name>  <value>70.12.114.197:50090</value>  <!-- secondary namenode의 IP 주소 -->  </property>  <property>  <name>dfs.name.dir</name>  <value>/etc/hadoop-1.2.1/name</value>  </property>  <property>  <name>dfs.data.dir</name>  <value>/etc/hadoop-1.2.1/data</value>  </property>  </configuration>   * /etc/hadoop-1.2.1/conf/mapred-site.xml   MapReduce 환경 정보를 설정  <configuration>  <property>  <name>mapred.job.tracker</name>  <value>master:9001</value>  </property>  </configuration>   * /etc/hadoop-1.2.1/conf/hadoop-env.sh → 9번 째 줄 수정   export JAVA\_HOME=/etc/jdk1.8  export HADOOP\_HOME\_WARN\_SUPPRESS="TRUE" // 오류메세지 발생 방지   * /etc/bashrc → 맨 아래에 추가   . /etc/hadoop-1.2.1/conf/hadoop-env.sh  서버를 켤 때마다 env 실행되도록 설정   * hadoop 포맷 후 reboot   hadoop namenode -format  /etc/hadoop-1.2.1 아래에 name, data, temp 디렉토리가 생겼으면 제대로 포맷된 것  4. MariaDB 세팅 후 Hive 이용을 위한 세팅  ##mysql admin 권한으로 로그인 후 ,  ## hive 에게 권한을 주고, 사용할 database를 생성한다.  grant all privileges on \*.\* to 'hive'@'localhost' identified by '111111';  flush privileges;  create database hive\_db;  grant all privileges on hive\_db.\* to 'hive'@'%' identified by '111111' with grant option;  grant all privileges on hive\_db.\* to 'hive'@'localhost' identified by '111111' with grant option;  flush privileges;  commit;  Hive에 연결할 DB를 설정하기 위해 hive/conf/hive-sitexml 생성 및 세팅  <?xml version="1.0"?>  <?xml-stylesheet type="text/xsl" href="configuration.xsl"?>  <configuration>  <property>  <name>hive.metastore.local</name>  <value>true</value>  <description>controls whether to connect to remove metastore server or open a new metastore server in Hive Client JVM</description>  </property>      <!-- Maria DB 의 테이블과 jdbc driver, 계정 id 와 pwd 지정-->  <property>  <name>javax.jdo.option.ConnectionURL</name>  <value>jdbc:mariadb://localhost:3306/hive\_db?createDatabaseIfNotExist=true</value>  <description>JDBC connect string for a JDBC metastore</description>  </property>  <property>  <name>javax.jdo.option.ConnectionDriverName</name>  <value>org.mariadb.jdbc.Driver</value>  <description>Driver class name for a JDBC metastore</description>  </property>  <property>  <name>javax.jdo.option.ConnectionUserName</name>  <value>hive</value>  <description>username to use against metastore database</description>  </property>  <property>  <name>javax.jdo.option.ConnectionPassword</name>  <value>111111</value>  <description>password to use against metastore database</description>  </property>  <property>  <!--data를 넣을때 hadoop 의 어느 디렉토리에 저장할지 지정.  밑의 값은 디폴트와 동일 -->  <name>hive.metastore.warehouse.dir</name>  <value>/user/hive/warehouse</value>  </property>  <property>  <!--hive 내에서 header를 출력-->  <name>hive.cli.print.header</name>  <value>true</value>  </property>  </configuration>  Hadoop 내의 hive를 사용할 디렉토리 생성 및 설정  $ hadoop fs -mkdir /tmp  $ hadoop fs -mkdir /user/root/warehouse  $ hadoop fs -chmod g+w /tmp  $ hadoop fs -chmod g+w /user/root/warehouse  $ hive  -- error ## /tmp/hive 디렉토리가 생성된다.  $ hadoop fs -chmod 777 /tmp/hive  Hive 테이블 생성  CREATE TABLE post\_log(  year STRING,  month STRING,  day STRING,  cmt STRING,  id STRING,  postid STRING  )ROW FORMAT DELIMITED  FIELDS TERMINATED BY ','  LINES TERMINATED BY '\n'  STORED AS TEXTFILE;  빅데이터를 Hive 테이블에 입력  load data local inpath '/etc/tomcat/webapps/log/shower/post\_log.19-0809’ overwrite into table post\_log partition (date=’190809'); | | | | | | |
| 5. 빅데이터 시스템과 통합된 Spring MVC    5-2. hive 시스템 연동을 위한 코드 작성  ① hive와 java를 연동하기 위한 listener 실행 → hive –service hiveserver2  ② post log출력  @RequestMapping("/postlog.sh")  @ResponseBody  public void getlog(HttpServletResponse hsr) throws Exception{  Class.forName("org.apache.hive.jdbc.HiveDriver");  HiveDriver a;  Connection conn = DriverManager.getConnection("jdbc:hive2://70.12.114.206:10000/default", "", "");  Statement stmt = conn.createStatement();  ResultSet rs = stmt.executeQuery("SELECT postid ,count(\*) FROM post\_log group by postid");  JSONArray ja = new JSONArray();  while (rs.next()) {  int year = Integer.parseInt(rs.getString(2));  JSONObject jo = new JSONObject();  jo.put("name", rs.getString(1));  jo.put("y", Integer.parseInt(rs.getString(2)));  ja.add(jo);  }  conn.close();  PrintWriter out = null;  try {  hsr.setCharacterEncoding("EUC-KR");  hsr.setContentType("text/json;charset=UTF-8");  out = hsr.getWriter();  } catch (IOException e) {  e.printStackTrace();  }  out.print(ja.toJSONString());  }  ③ airline log 출력  @RequestMapping("/getchart.sh")  @ResponseBody  public void getchart(HttpServletResponse hsr) throws Exception{  Class.forName("org.apache.hive.jdbc.HiveDriver");  Connection conn = DriverManager.getConnection("jdbc:hive2://70.12.114.206:10000/default", "root", "111111");  Statement stmt = conn.createStatement();  ResultSet rs = stmt.executeQuery("SELECT month , count(\*) FROM airline\_delay where delayyear='2006' group by month order by month");  JSONArray ja = new JSONArray();  int m =1;  while (rs.next()) {  int year = Integer.parseInt(rs.getString(2));  JSONObject jo = new JSONObject();  ja.add(year);  }  conn.close();  PrintWriter out = null;  try {  hsr.setCharacterEncoding("EUC-KR");  hsr.setContentType("text/json;charset=UTF-8");  out = hsr.getWriter();  } catch (IOException e) {  e.printStackTrace();  }  out.print(ja.toJSONString());  }  ▶ 실행결과 | | | | | | |