1. **Cloud**
   1. **Cloud 종류**
      1. **퍼블릭 클라우드**
         1. **클라우드 사업자가 시스템을 구축하고, 인터넷망 등의 네트워크를 통해 불특정 다수의 기업과 개인에게 서비스를 제공하는 형태.**
         2. **필요한 컴퓨팅 자원을 단기간에 저비용으로 마련할 수 있고 운용 관리 부담이 적다.**
      2. **프라이빗 클라우드**
         1. **클라우드 서비스의 사용자 또는 사업자의 데이터 센터에 클라우드 관련 기술이 활용된 자사 전용 환경을 구축.**
         2. **가상화, 자동화와 같은 클라우드 관련 기술의 활용으로 인해 시스템의 성능과 비용이 최적화, 유연한 사용자 정의가 가능**
      3. **커뮤니티 클라우드**
         1. **공통의 목적을 가진 특정 기업들이 클라우드 시스템을 형성하여 데이터 센터에서 공동 운영하는 형태**
      4. **하이브리드 클라우드**
         1. **각종 클라우드 서비스들과 온프레미스 시스템을 연계시켜 활용하는 시스템**
   2. **Cloud 서비스의 종류**
      1. **SaaS(Software as a Service)**
         1. **하나의 서버를 여러 기업에서 공유하는 것을 전제한 멀티 테넌트 방식, 그러나 데이터는 기업 사용자별로 분리되도록 설계하여 보안성을 확보**
         2. **사용자 계정을 마련하면 서비스를 바로 이용할 수 있고, 이동 중에도 스마트폰이나 태블릿 등의 다양한 기기로 접속 가능**
      2. **PaaS(Platform as a Service)**
         1. **애플리케이션 실행 환경이나 데이터베이스 등이 미리 마련되어 있어서 단기간에 응용 프로그램을 개발하여 서비스를 제공할 수 있음**
         2. **애플리케이션 활용 자유도가 높지만, PaaS 의존도가 높아지면 다른 환경으로 마이그레이션이 힘듬**
         3. **Cloud Foundry, Openshift, Heroku 등…**
      3. **IaaS(Infrastructure as a Serivce)**
         1. **가상 서버 및 온라인 스토리지 등, 하드웨어 자원을 소프트웨어적으로 나누어 사용자에게 제공**
         2. **가상 서버 리소스는 필요에 따라 스케일 업, 다운이 가능하며 데이터베이스, 미들웨어, 응용 프로그램 등의 소프트웨어를 자유롭게 운영할 수 있지만, 그만큼 기업 사용자의 책임이 많아진다.**
         3. **사용량에 따른 종량제 혹은 월 정액제를 채택**
         4. **일시적으로 컴퓨터 리소스를 많이 빌려서 사용하고, 프로모션이 종료되면 리소스를 감축시키는 등 안정적인 구축 비용과 운용 비용을 절감 가능**
         5. **Amazon Elastic Compute Cloud(EC2)**
   3. **Cloud 서비스 특징, 용어**
      1. **가상 서버**
         1. **물리적 서버의 CPU와 메모리, 스토리지 같은 하드웨어 자원을 소프트웨어로 논리적으로 나누어 사용하는 것**
      2. **로드 밸런서**
         1. **외부의 접속을 통신 데이터양 등의 조건에 따라 가상 서버에 전달**
         2. **통신의 부하를 분산하여 대량의 접속에도 대응할 수 있는 시스템을 손쉽게 구축**
      3. **오토 스케일**
         1. **접속이 집중되는 때와 같이 외부의 통신 데이터 양에 따른 부하에 따라 자동으로 가상 서버의 대수를 증감시키는 기능**
         2. **스케일 업은 성능을 증가, 스케일 아웃은 동일한 성능의 가상 서버 개수를 증가, 스케일 다운은 성능을 감소, 스케일 인은 서버 가상 서버 개수를 감소**
      4. **스냅샷**
         1. **수동 또는 이용자가 설정한 시간에 가상 서버 디스크의 백업을 자동으로 생성**
   4. **Cloud 서비스의 구성**
      1. **스토리지 서비스**
         1. **S3(Amazon Simple Storage Service), 파일 단위로 저장/검색/삭제하는 저장소이며, 다수의 데이터 센터에 데이터를 분산 저장하여 데이터의 손실을 예방, 사용한 용량만큼 요금이 부과**
         2. **Amazon S3 Glacier, 데이터의 읽기에 시간이 오래 걸리기 때문에 빈번하게 읽을 필요가 없는 장기 보존용**
      2. **네트워크 서비스**
         1. **Amazon VPC(Virtual Private Cloud), 가상 네트워크를 만들어서, 개인 클라우드처럼 사용할 수 있는 서비스**
         2. **프라이빗 IP 주소가 할당된 네트워크에 가상 VPN 게이트웨이를 제공, 거점과 거점을 인터넷 VPN으로 연결**
         3. **필터링을 통해 불필요한 통신을 차단하는 기능을 제공**
      3. **데이터베이스 서비스**
         1. **RDBMS(Relational Database Management System : 관계형 데이터베이스 관리 시스템), MySQL, Oracle Database 등이 있다.**
         2. **RDB는 여러 개의 데이터를 행과 열이 있는 표 형식으로 표현하여, 복잡한 데이터의 관계를 처리할 수 있도록 만든 데이터베이스**
         3. **NoSQL, RDB와 같은 관계형 데이터베이스가 아닌 데이터베이스**
         4. **대량의 데이터를 분산시켜 고속으로 처리하는 분산 데이터베이스**
2. **Linux & Docker**
   1. **리눅스 명령어**
      1. **ls -a, -l, -A(디렉토리 내용 확인)**
      2. **cd(디렉토리 이동)**
      3. **man(매뉴얼 보기)**
      4. **mkdir -p(디렉토리 생성)**
      5. **rm -rf(파일 지우기)**
      6. **sudo(일시적으로 관리자 권한 부여)**
      7. **history !실행라인(사용했던 명령어 목록 확인)**
      8. **cat(화면상에서 파일 내용 확인)**
      9. **head(파일 앞부분 출력)**
      10. **tail(파일 끝부분 출력)**
      11. **cp(파일 복사하기)**
      12. **mv(파일 옮기기)**
      13. **touch(빈 파일 생성)**
      14. **grep(특정 조건 검색)**
      15. **find(파일 검색하기)**
      16. **ps -ef(프로세스 상태 확인하기)**
      17. **kill -2, -9(프로세스 강제 종료)**
      18. **rpm -iv, -Uv, -e, -l (패키지 관리, 설치, 업그레이드, 삭제, 목록 순)**
      19. **useradd(사용자 계정 만들기)**
      20. **groupadd(그룹 생성)**
      21. **chmod(파일 권한 변경)**
      22. **chown(파일 소유자 변경)**
   2. **리눅스 네트워크**
      1. **hostname**
         1. **서버에 접속된 수많은 컴퓨터들이 자신을 구별하기 위해 가지고 있는 것이 IP주소**
         2. **이러한 IP주소를 기억하기 쉽게 사용하는 것이 hostname**
         3. **hostname 명령어로 확인하거나 일시적으로 변경 가능**
         4. **/etc/sysconfig/network 에서 변경하면 영구적으로 적용**
      2. **ip address**
         1. **ip 명령어에 object인 address를 추가한 것으로 ip주소를 조회하고 나아가 추가 혹은 삭제도 가능한 명령어**
      3. **DNS**
         1. **네트워크에서 도메인이나 호스트 이름을 숫자로 된 IP 주소로 해석해주는 TCP/IP 네트워크 서비스**
         2. **hostname의 기능과 유사**
         3. **간단하게 설명하면 특정 사이트의 IP주소가 특정 도메인 이름(예시 :** [**www.google.com**](http://www.google.com)**)으로 되어있을 때, DNS 서버에서 해당 도메인의 IP 주소를 연결해주는 교환국 같은 역할**
   3. **Vi 에디터의 명령어**
      1. **명령모드 🡪 입력모드 전환**
         1. **i(현재 커서 위치에서부터 입력)**
         2. **a(현재 커서 다음 위치에서부터 입력)**
         3. **o(현재 커서의 다음 행에서부터 입력)**
      2. **입력모드 🡪 명령모드 전환**
         1. **Esc**
      3. **명령모드에서 단축키**
         1. **h, j, k, l(커서 이동)**
         2. **w(단어 단위로 이동)**
         3. **cw(단어 단위로 삭제)**
         4. **x(한 글자씩 삭제)**
         5. **dd(한 줄 삭제)**
         6. **/(검색 기능 == ctrl + f)**
         7. **G(문서 제일 마지막으로 이동)**
         8. **:w(저장)**
         9. **:wq!(저장, 종료)**
         10. **:q**
         11. **:q!(저장x, 강제종료)**
         12. **:set no(행 번호 표시)**
         13. **:set nonu(번호 표시 삭제)**
         14. **:라인넘버(해당 라인으로 이동)**
         15. **:sh(일시적으로 쉘 명령어로 이동, exit로 복귀 가능(**
   4. **Docker 컨테이너 가상화의 특징**
      1. **Hypervisor**
         1. **기존 OS의 가상환경에 또 다른 OS를 설치하여 완전히 독립적인 OS로 운영하는 것**
         2. **OS에 필요한 만큼 리소스가 할당이 되기 때문에 효율적으로 사용 가능**
         3. **별도로 운영되는 독자적 OS 덕분에 하나의 가상환경이 공격당해도 다른 가상환경에 영향을 끼치지 않음**
         4. **각기 다른 OS를 사용하기에는 부팅 시간이 길고 리소스를 많이 차지**
      2. **Docker container**
         1. **기존 OS의 커널 위의 공간에서 실행되어 하나의 호스트 서버에서 여러 독립적인 가상 환경을 생성**
         2. **리소스적으로 가볍고 빠르기에 가상환경을 빠르게 실행할 수 있고 일반적인 시스템 API interface를 사용**
         3. **독립적인 OS가 아니기에 기존 OS와 다른 운영체제 실행 불가**
         4. **호스트 서버에서 직접적으로 운영되기에 컨테이너가 공격당하면 다른 컨테이너도 공격당할 위험이 있음**
   5. **Docker 명령어**
      1. **docker image ls, pull, push, rm, build, tag**
      2. **docker container ls, create, start, stop, run, rm**
   6. **Docker container 정보를 조회 명령어들**
      1. **docker inspect를 통해 컨테이너 정보를 조회 가능**
   7. **Docker 실행 옵션**
      1. **docker run -d -p -e -it -rm**
   8. **Dockerfile 명령어**
      1. **FROM, ADD, COPY, ENV, ARG, CMD, ENTRYPOINT**
   9. **Docker port forwarding**
      1. **컨테이너 내부에서의 접속이 아닌 외부에서 접속할 때, 필요한 포트 경로를 지정해주는 개념이다.**
3. **Python**
   1. **프로그램 결과 예측**
   2. **객체지향언어 🡪 override 🡪 재정의할 수 있는 메소드**
      1. **재정의(override)는 기반 형식에 정의한 멤버 메서드와 같은 이름으로 파생 형식에서 정의하는 것을 말합니다. 이 때 해당 이름으로 접근하면 새롭게 정의한 파생 형식의 멤버를 접근합니다**
      2. **메소드 오버라이딩은 부모 클래스의 메소드를 자식 클래스에서 재정의 하는 것입니다.**
   3. **생성자 메소드**
      1. **파이썬 클래스에는 인스턴스 생성과 동시에 자동으로 호출되는 메서드인 생성자가 존재**
      2. **def.\_\_init\_\_(self, 필요 변수(생략 가능))**
   4. **에러 코드 정리**
   5. **Collection의 종류와 정의 방법, 사용방법, 특징**
      1. **list**
         1. **정의 방법 : list(), []**
         2. **특징 : 인덱싱, 슬라이싱, 추가, 삭제, 갱신**
      2. **tuple** 
         1. **정의 방법 : tuple(), ()**
         2. **특징 : 인덱싱, 슬라이싱, 병합, 삭제 & 갱신 불가**
      3. **set** 
         1. **정의 방법 : set()**
         2. **특징 : 순서가 없는 형식이기에 인덱싱, 슬라이싱 불가, 추가, 삭제는 가능, union(교집합), intersection(합집합), difference(차집합)**
      4. **dictionary** 
         1. **정의방법 : dict(), {}**
         2. **특징 : 키:값 의 형식을 지닌 자료형, 순서는 없으므로 인덱싱, 슬라이싱 불가능, 추가, 삭제, 갱신 가능**
   6. **Python 패턴 사용법 import re, 정의, 사용법**
      1. **정규표현식(Regular expression)을 이용한 문자열 매칭에 사용되는 파이썬 라이브러리이며 파이썬 정규표현식을 이용하여 문자열에서 특정 패턴의 문자열을 매칭하는 기능을 제공한다. 정규표현식은 '.', '\*', '$' 등의 특수한 메타문자를 이용한다.**
      2. **import re**
      3. **a = re.compile(‘정규식’)**
      4. **m = a.match(str), m = a.search(str)**
         1. **match의 경우 처음부터 검색해서 조건을 완전히 만족해야 반환**
         2. **search의 경우 전부 검색해서 조건이 존재하면 반환**
      5. **group(), start(), end(), span()을 이용하여 반환된 객체의 결과값을 정리할 수 있다.**