SAMSUNG SDS

Foresee

# Techtonic 2021

Partner

Disrupt

누구나 개발 할 수 있다

# Low Code Development Platform

서지원 프로

# AGENDA

IT 개발 환경의 급격한 변화

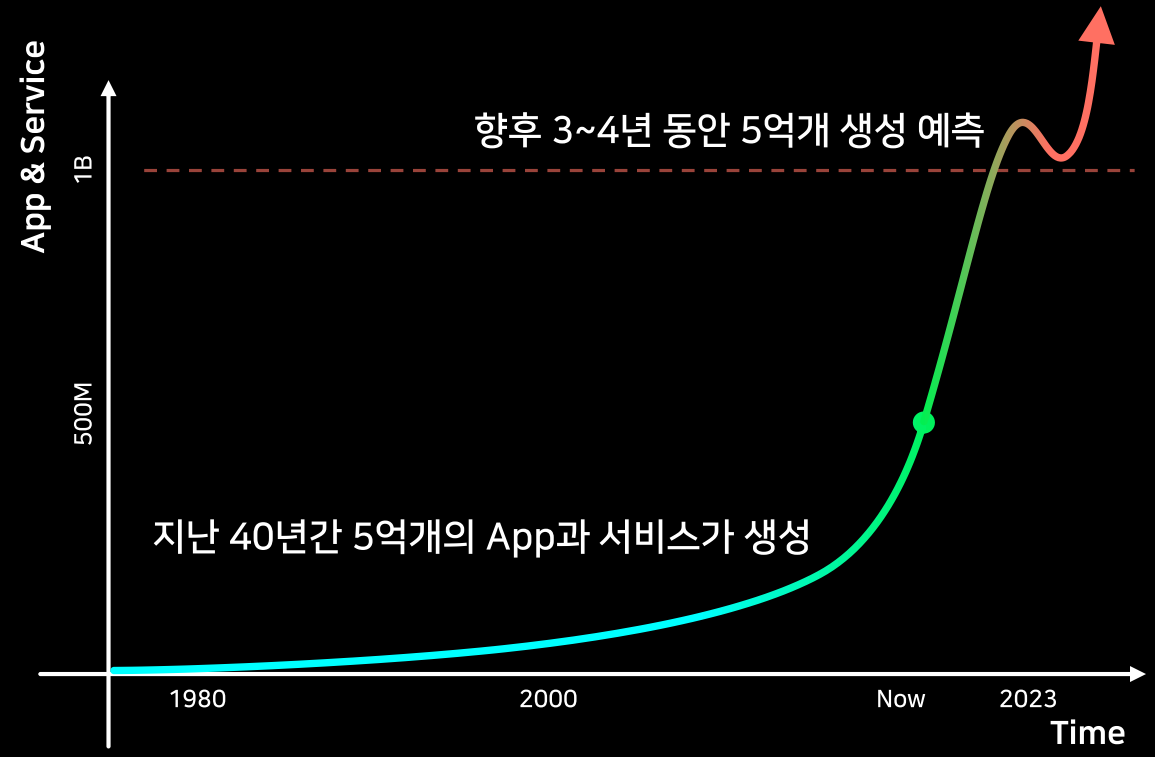
Citizen Developer

Low Code Development Platform

기술적 고민들

# IT 개발 환경의 급격한 변화 – 1st Impact : 개발자 부족

## Industry Application Explosion




## 수요에 비해 심각하게 부족한 개발자

개발자 부족 계속 됩니다...학위 없어도 문제 없어요
매일경제 (배윤경) : 21.04

올해만 1만명 부족...뺏기면 죽는 개발자 쟁탈전
머니투데이 (이진욱) : 21.03

다시 불붙은 개발자 확보戰
조선일보 (조유미) : 21.09

잇딴 버그 잡아야 하는데...코딩 인력은 부족한 삼성전자
중앙일보 (김영민) : 19.10

내년 IT인력 1.5만명 부족..."AI 인재 몸값 천정부지"
한경 (김동현) : 21.04

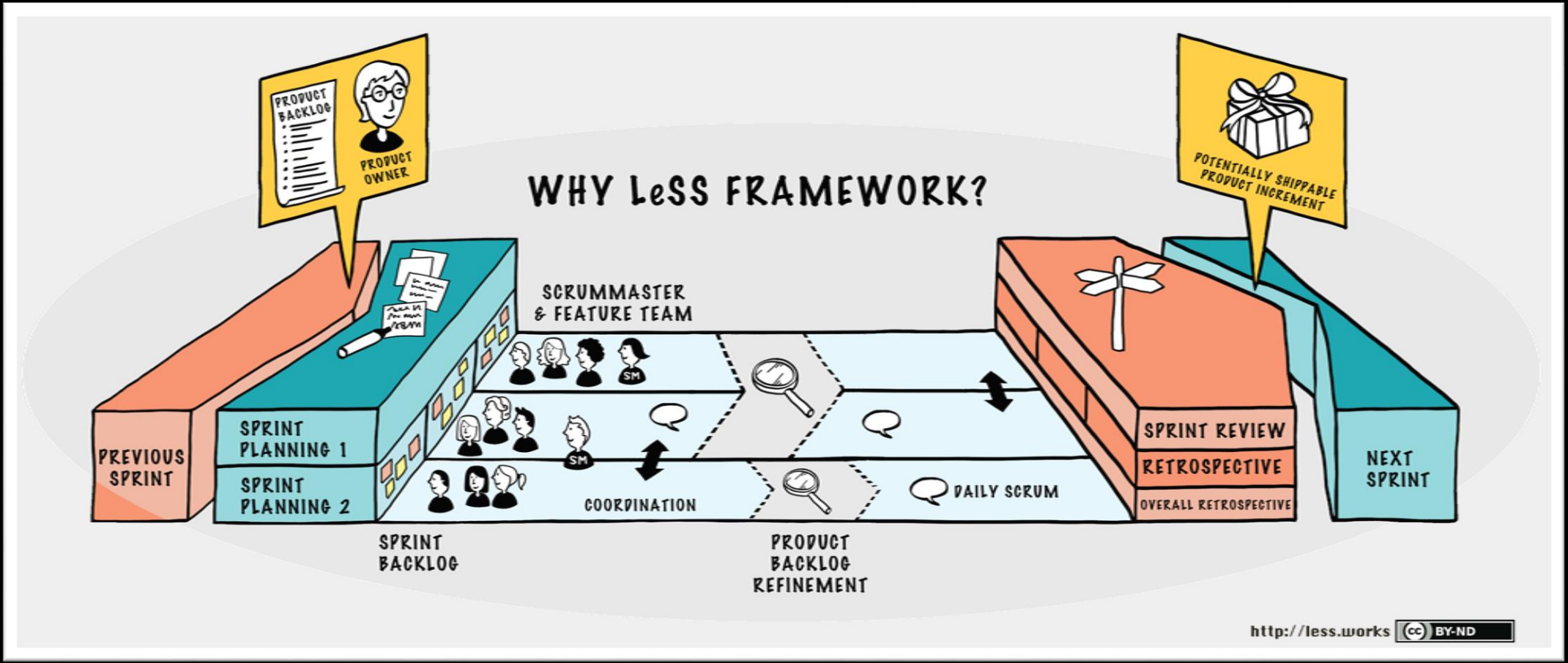
개발자 초봉 '6천'부터 모셔온다...IT 인력 공급 부족해
문화뉴스 (김종민) : 21.02

코로나가 부른 'IT 인력 전쟁'..."비전공자도 키운다"
KBS NEWS (김민경) : 21.04

개발자 부족시대, '시민 개발자'가 뜬다
Coding world News(이진영) : 21.09

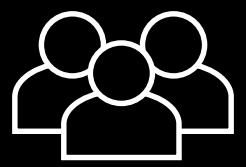
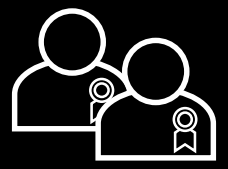
# IT 개발 환경의 급격한 변화 – 2nd Impact : Whole Team powered by Agile

Agile 기반 Whole Team



# Citizen Developer

## Low-code vs. Pro-code

| | Citizen (End User) | Citizen Developer | Pro Developer |
|---|---|---|---|
| |  |  |  |
| IT/개발 지식 | Low | Mid | High |
| 사용 도구 | No-Code | Low-Code | Pro-Code |
| App 규모 | Individual | Work Group/ Departmental | Enterprise |

## 업무중요도 vs. IT전문지식

- IT 전문지식 없이 개별 업무데이터를 활용하여 스스로 단순기능을 개발/조작
- IT 전문지식 갖춘 사람의 “일부” 도움을 받아 업종전문 기능을 구현
- 명확한 IT 기준 및 프로세스에 따라 회사차원의 복합적인 업무 구현

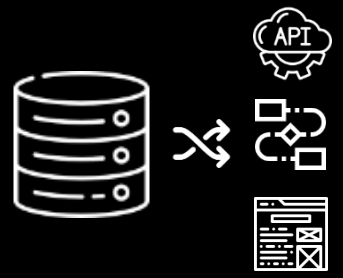
# Low Code Development Platform – 정의

A low-code development platform (LCDP) provides a development environment used to create application software through a graphical user interface instead of traditional hand-coded computer programming. A low-coded platform may produce entirely operational applications, or require additional coding for specific situations. Low-code development platforms reduce the amount of traditional hand coding, enabling accelerated delivery of business applications. A common benefit is that a wider range of people can contribute to the application's development—not only those with coding skills. LCDPs can also lower the initial cost of setup, training, deployment and maintenance.

Low-Code Development Platform는 전통적인 hand coding 대신에 GUI 통해 애플리케이션 SW를 만들 수 있게 개발환경을 제공한다. LCDP는 애플리케이션 전체를 만들 수 있지만, 특정 상황에서는 추가 코딩이 필요하다.  이들은 비즈니스 애플리케이션을 재빨리 전달할 수 있도록 전통적인 hand coding 양을 줄여준다. 공통적인 이점은 코딩 스킬을 가진 사람들뿐만 아니라 더 폭넓은 범위의 여러 사람들이 애플리케이션 개발에 참여할 수 있다는 것이다. 또한 LCDP는 초기 setup, 학습, 개발 및 유지보수에 대한 비용을 낮출 수 있다.

[ 출처 ] en.wikipedia.org/wiki/Low-code_development_platform

# LCDP – Methodology



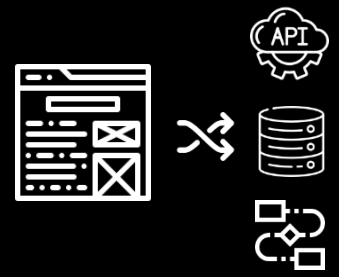

## Model Driven

**Pros**

모델을 정의하고 모델에서 화면, API등을 자동 생성하므로 자동화율이 높음

**Cons**

관계형 모델은 처리하기 어려움




## Canvas Driven

**Pros**

화면 저작에서 시작하므로 사용자 접근이 가장 편함

**Cons**

화면 개별 컴포넌트에 대한 변수 처리, Evnet 연결 등 추가로 해야 하는 일이 많음

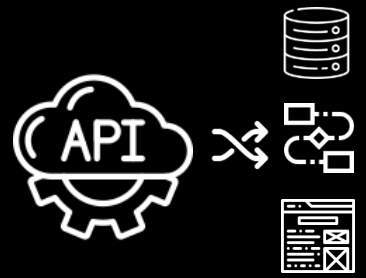

## Process Driven

**Pros**

업무 프로세스를 정의하고 각 프로세스에 대한 구현을 GUI로 수행함. 업무흐름이 명확히 표현

**Cons**

중소규모 Application의 경우 적합하지 않음




## API Driven

**Pros**

API를 먼저 정의하고 이를 기준으로 화면, 로직, 프로세스가 연결됨. 협업 유리

**Cons**

API를 잘 만들기 위해서는 IT 지식이 필요함

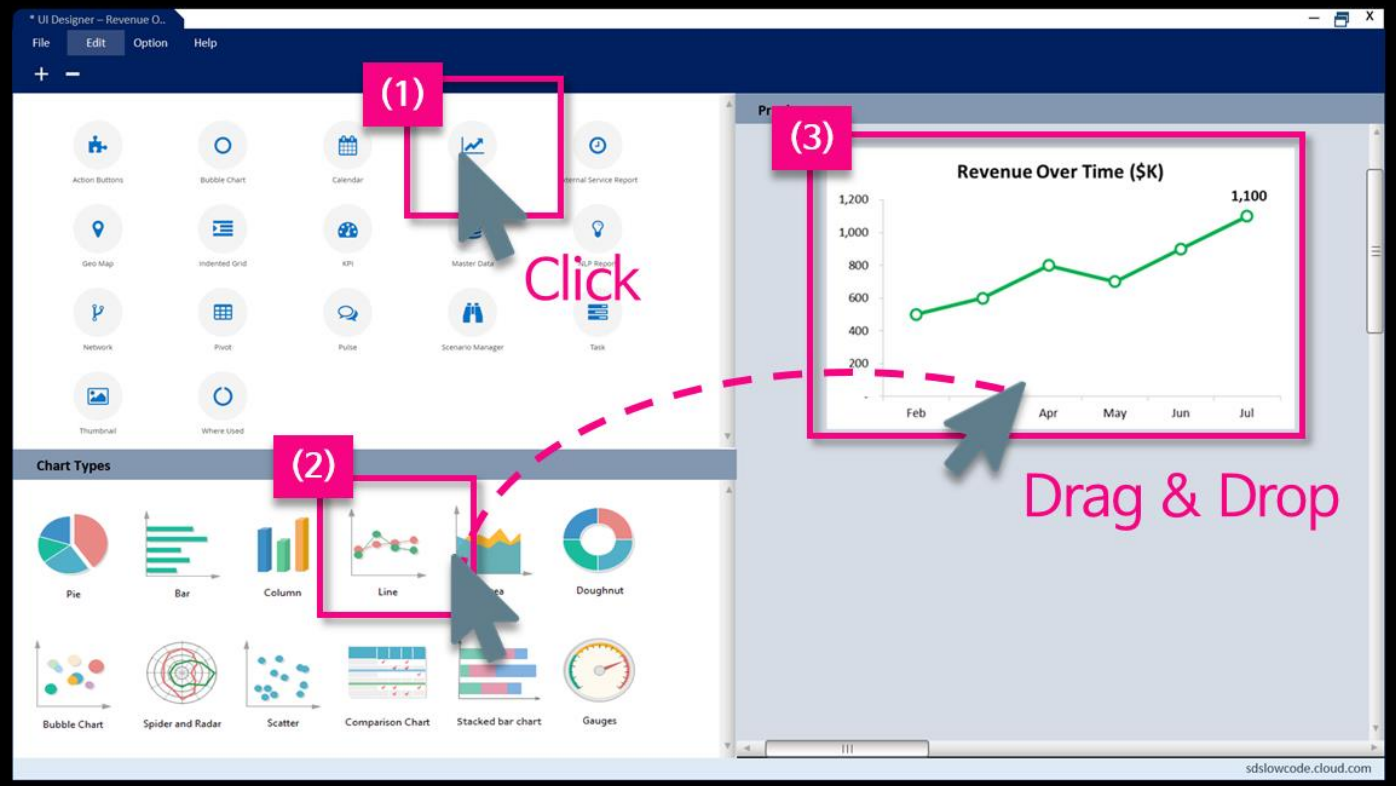
# LCDP – 화면 저작

화면

로직

Cloud

## Step 1. Low Code UX 화면에서 추가할 component를 선택하여 drag & drop



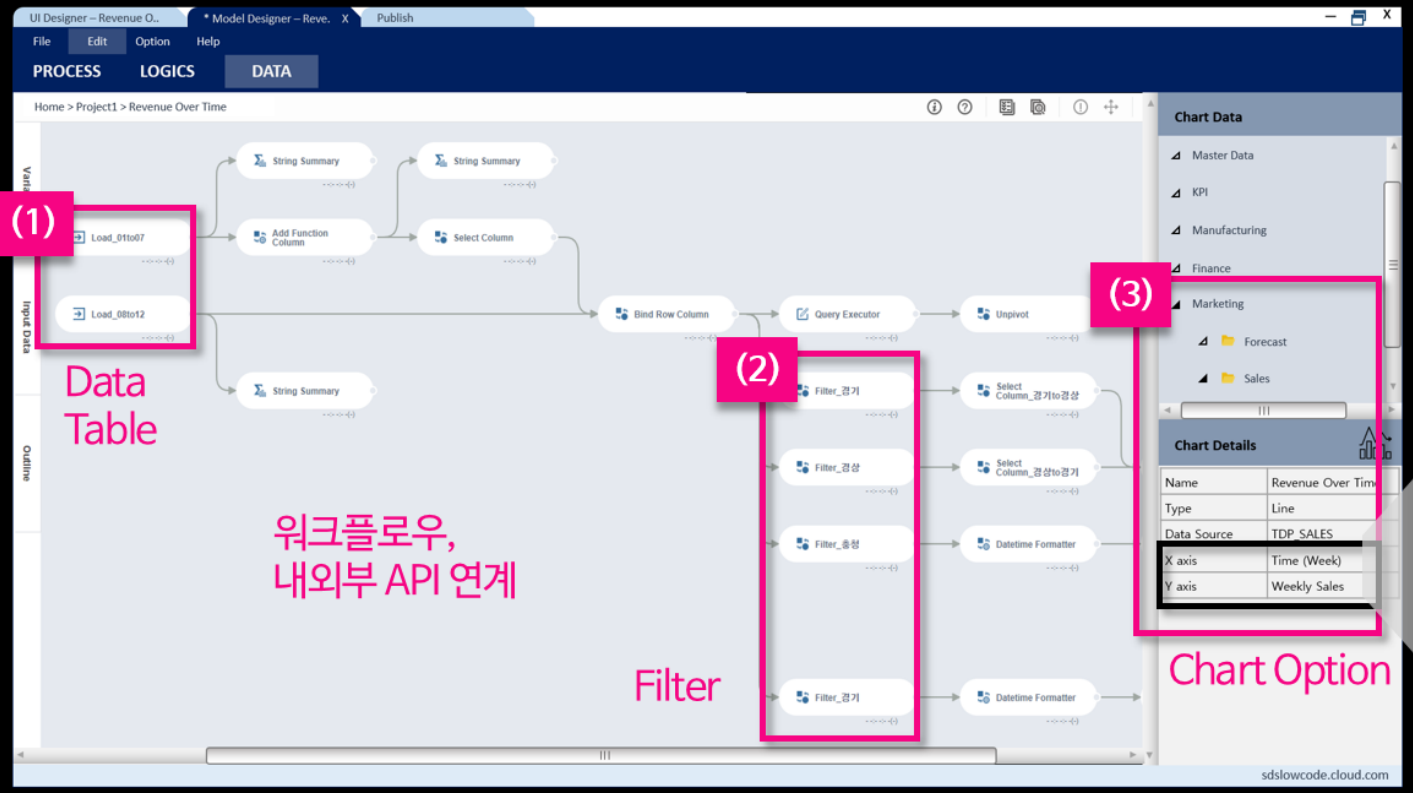
# LCDP – 로직 저작

화면

로직

Cloud

## Step 2. 신규 Component의 내/외부 인터페이스 및 워크플로우 자동 연계




Click-based
데이터 커스터마이징

Data Source
: Weekly Sales

X : time
Y : Sales

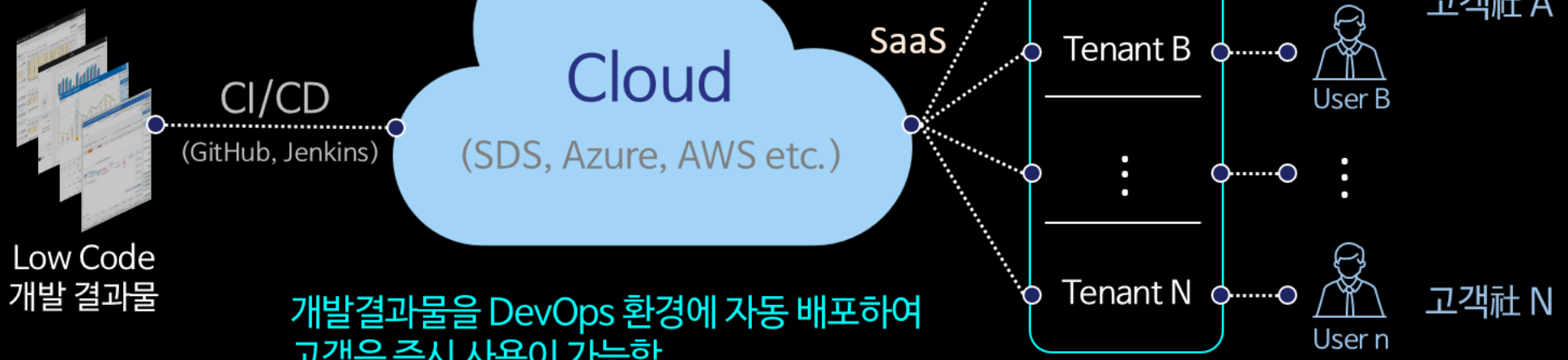
# LCDP – Cloud 배포

화면

로직

Cloud

Step 3. 배포할 Component를 자동 빌드 및 클라우드 환경에 1-click 배포




개발결과물을 DevOps 환경에 자동 배포하여
고객은 즉시 사용이 가능함

# 기술적 고민#1 - Floating Point Arithmetic(Machine epsilon)

**Excel**

| | A | B |
|---|---|---|
| 1 | 0.1 | =IF(SUM(A1:A10) = 1, TRUE,FALSE) |
| 2 | 0.1 | TRUE |
| 3 | 0.1 | |
| 4 | 0.1 | =IF(SUM(A1:A5) = 1, TRUE,FALSE) |
| 5 | 0.1 | FALSE |
| 6 | 0.1 | |
| 7 | 0.1 | |
| 8 | 0.1 | |
| 9 | 0.1 | |
| 10 | 0.1 | |

**Code(Java)**

Calculate.java

```
5 double result=0.0;
6 for(int i=0; i<10 ; i++) {
7     result += 0.1;
8 }
9 if(result == 1.0)
10     System.out.println(result+" == "+1.0);
11 else
12     System.out.println(result+" != "+1.0);
13 System.out.println("result : "+result);
```

Markers | Console | Progress | Search | Git Staging

<terminated> Calculate [Java Application] C:₩Program Files₩Zulu₩zulu-12₩bin₩javaw.exe (202

```
0.9999999999999999 != 1.0
result : 0.9999999999999999
```

2진수로는 실수의 정확한 값을 표현할 수 없음. 개발자가 상황에 맞게 처리해왔던 부분을 Low Code에서는 Apache Precision이나 Big Decimal 등을 사용하여 변형해야 함

※ IEEE 754(Standard for Floating-point Arithmetic)

# 기술적 고민#2 – Date Type

우리가 날짜에 대해서 "**일반적**"으로 고민해왔던 것들

2021-11-15

00:00:00

GMT+9

Nov 15, 2021

"2021년 11월 15일"

"15-11-2021"

"21' 11/15"

...

..

.

and **LowCode**

## UNIX time : getTime() 1636934400000

- UTC 기준(UTC+0) 1970년 1월 1일 0시 0분 0초부터 seconds 단위로 기록된 숫자
- 대부분의 시스템에서 표준으로 사용

## Time Zone : GMT+9(Korea)

- 그리니치 평균시와 협정 세계시, 그리고 각 상황 별 Time zone
- Client(사용자 접속위치)의 Time zone, Server/DataBase위치에 따른 플랫폼 별 TimeZone 동작, 데이터에 종속된 TimeZone(ex, 하역일시), 등... 각각의 상황에 맞는 TimeZone 처리

## Date format : YYYY-MM-DD

- 날짜 Format에 대해서 국가(문화)별 상황 별로 다른 기준을 적용해야 할 필요가 있음
- 때로는 사용자 별로/상황 별로 다른 Formatting 필요

## 2011년 Cloud

아무나 클라우드 구성 할 수 없습니다.

모든 App에 대한
인프라를 지원하지 못합니다

2021년
한국에서도
Cloud가 당연한 세상입니다.

## 2021년 Low Code

아무나 개발할 수 있지 않습니다.

모든 App을 Low code로
개발 할 수 없습니다.

2021년
한국에서도
첫발을 내딛고 있습니다.

# Thank you

SAMSUNG SDS