



Serverless Korea

THE SERVERLESS

사라진 떡국 레시피 되찾기





Serverless Korea

나이를 먹는 방법

아빠, 나이는
어떻게 먹어?

나이? 떡국을
먹으면 된단다

떡국 레시피가 어디 있더라...





Serverless Korea

사라진 떡국 레시피

전 세계 서버에서
완전히 사라진 떡국 레시피

Google

 NAVER

떡국 레시피가 없다고?



인류가 나이를 먹는 것을
방해하는 세력들



Serverless Korea

서버없는 레시피 구현

큰 일이야, 방법이 없어

더 이상 인류가 나이를 먹을
방법은 없는가...

마지막 희망, Serverless



목차



Serverless Korea

01

노아론

서버리스 장점
서버리스 특징
전체적인 구현 방식

02

김유진

떡국 레시피와 플로우 차트
세부적인 구현 방식
로직앱 과 파워앱

03

김홍민

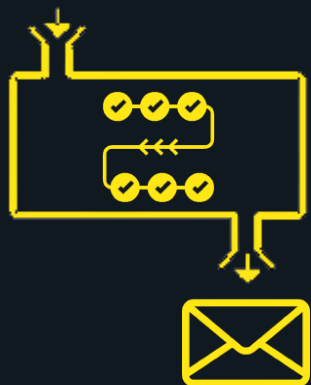
실제 결과
챌린지
활용방안

HACK THE PLANET

SW 요구사항 분석

기능

입력값







제약조건



목표

요구하는 기능들을
애저 서버리스 서비스를 이용하여
25일 내로 구현

시스템 명세

- 
입력값
- 떡 불리기
 - 파 썰기
 - 계란을 풀고 휘젓기
 - 재료를 끓이기
- 
-
- 
-
- 떡이 썰렸는지
 - 후추 토핑 여부
- 
-
- 이메일 주소

- 
처리내용
- 떡국 레시피
- 
출력
- 

 파워 오토메이트 만들기
#23 by justinyoo was closed on 23 Dec 2020
 파워 앱 만들기
#13 by justinyoo was closed on 23 Dec 2020
 레시피 스텝 9: 후추 뿌려 음식 내기
#12 by justinyoo was closed on 29 Nov 2020
 레시피 스텝 8: 계란 풀어 휘젓기 + 채 썬 파 넣기
#11 by justinyoo was closed on 24 Nov 2020
 레시피 스텝 7: 양념 추가 - 마늘, 간장, 소금
#10 by justinyoo was closed on 30 Nov 2020
 레시피 스텝 6: 계란 풀기
#9 by justinyoo was closed on 24 Nov 2020
 레시피 스텝 5: 끓는 동안 거품 제거
#8 by justinyoo was closed on 30 Nov 2020
 레시피 스텝 4: 재료 물에 넣고 끓이기
#7 by justinyoo was closed on 24 Nov 2020
 레시피 스텝 3: 소고기 볶기
#6 by justinyoo was closed on 22 Nov 2020
 레시피 스텝 2: 파 썰기
#5 by justinyoo was closed on 30 Nov 2020
 레시피 스텝 1: 가래떡 썰기 + 물에 불리기
#4 by justinyoo was closed on 22 Nov 2020
 전체 아키텍처 구성
#3 by justinyoo was closed on 30 Nov 2020
 순서도 작성
#1 by justinyoo was closed on 18 Nov 2020

그림 1. 프로젝트 이슈들

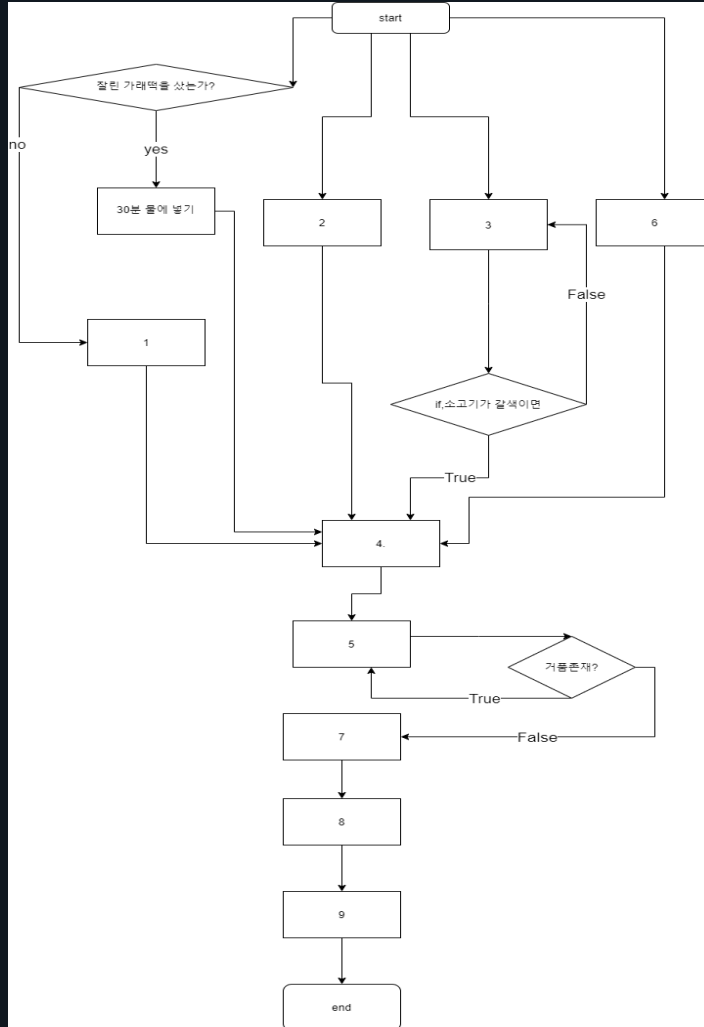
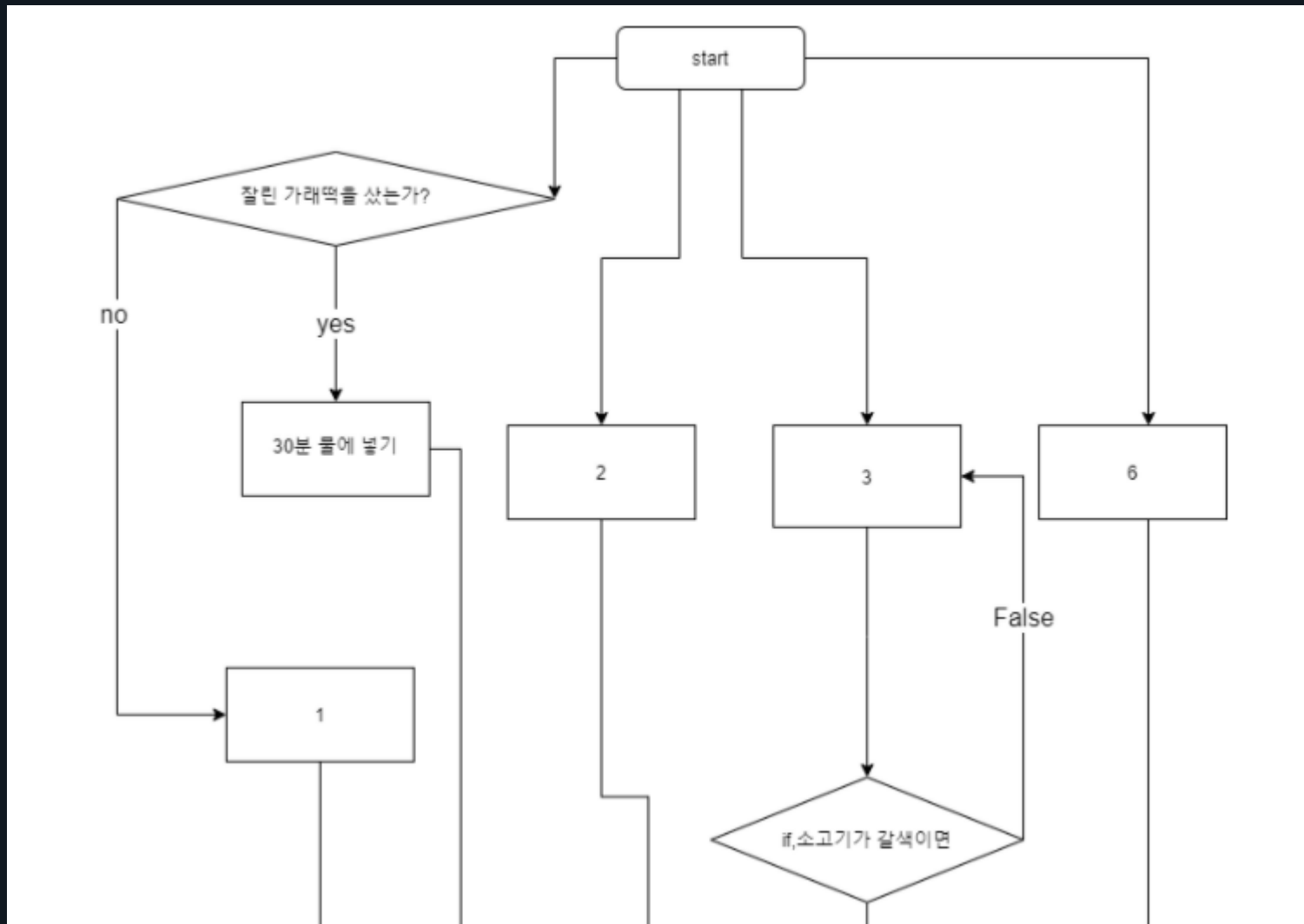


그림 1. 떡국 레시피 자동 프로세스 순서도

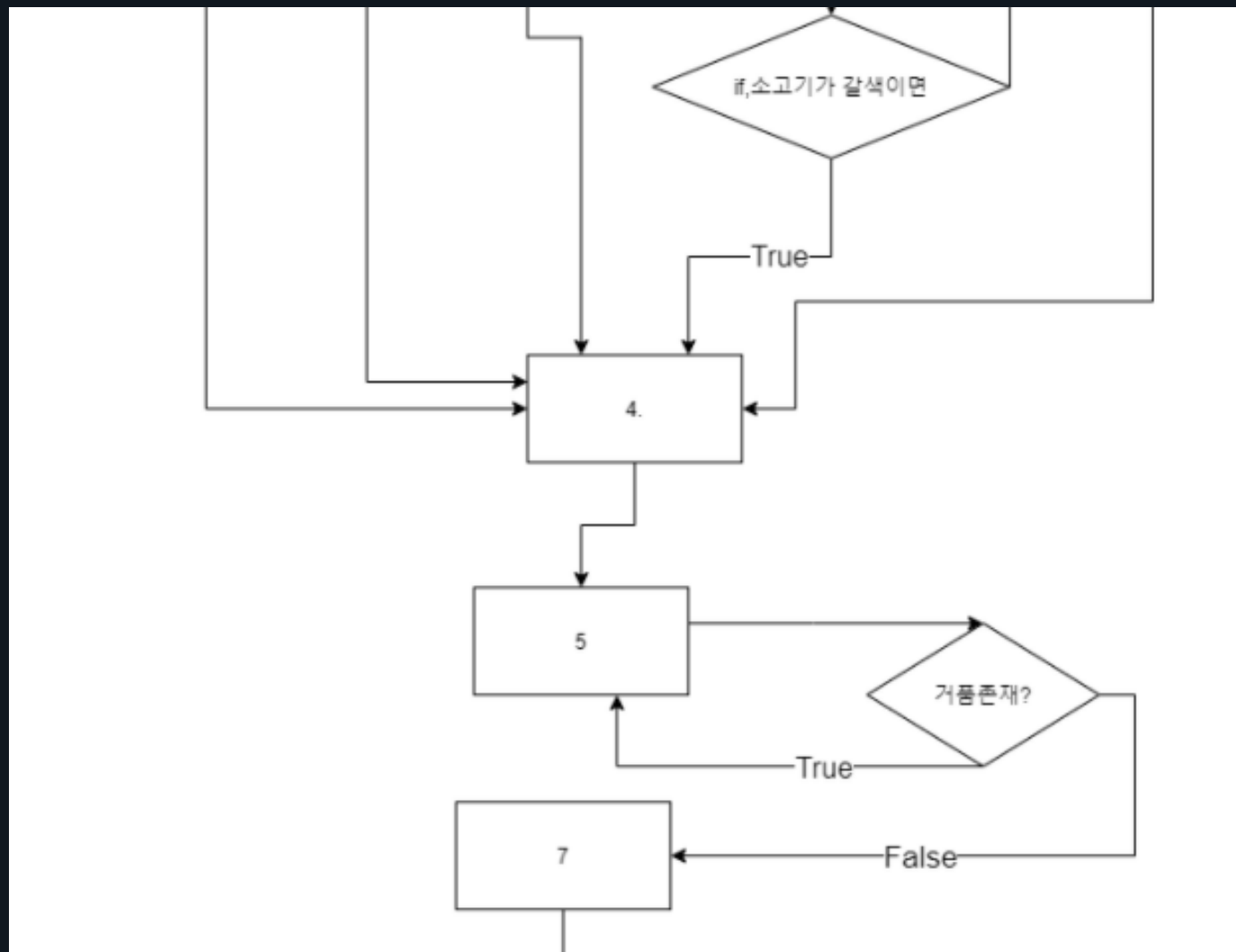
순서도

떡국 레시피의 절차에 따라 프로그램의 알고리즘을 작성

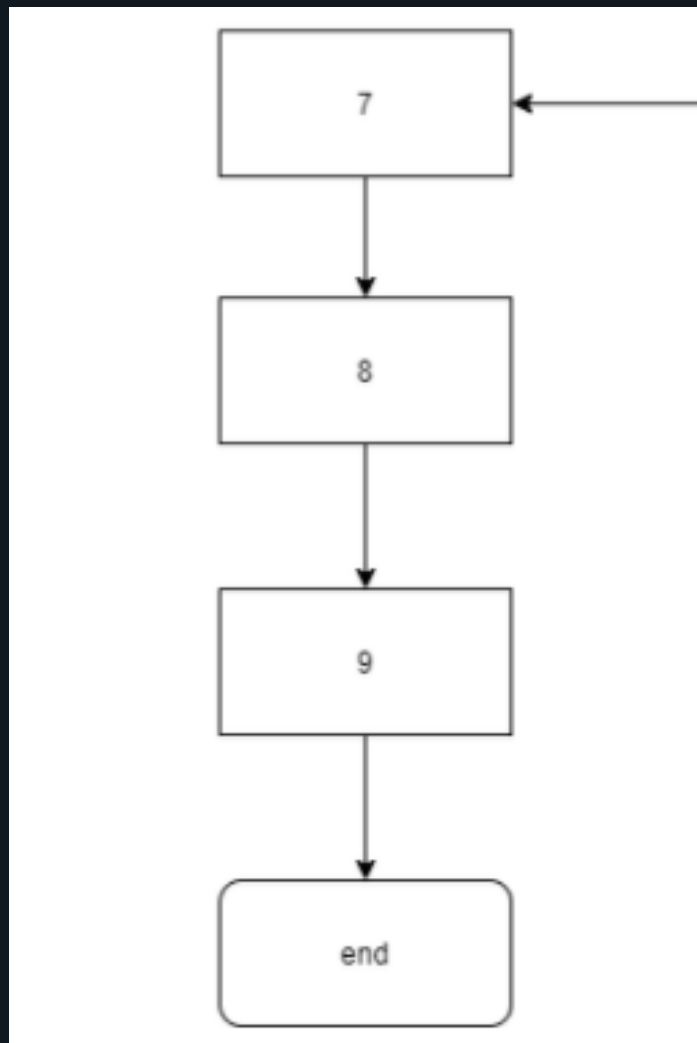
순서도 – 재료손질



순서도 - 조리



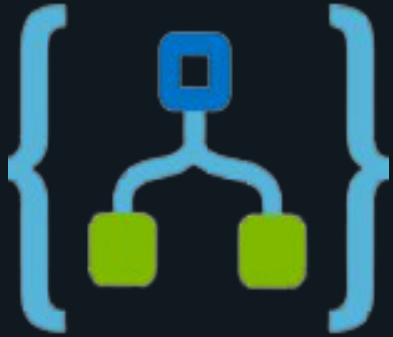
순서도 - 마무리





평션앱

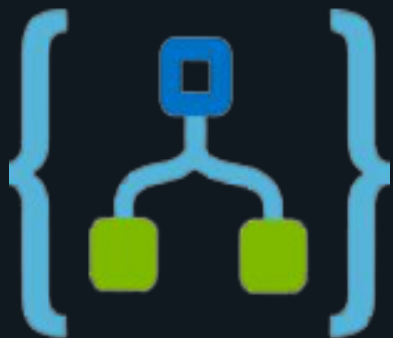
- 오케스트레이터 평션(Orchestrator Function)
- 액티비티 평션(Activity Function)
- 클라이언트 평션(Client Function)



로직앱

- 이벤트 발생, 트리거로 작동
- 작업 흐름(Work Flow)

웹훅 URL

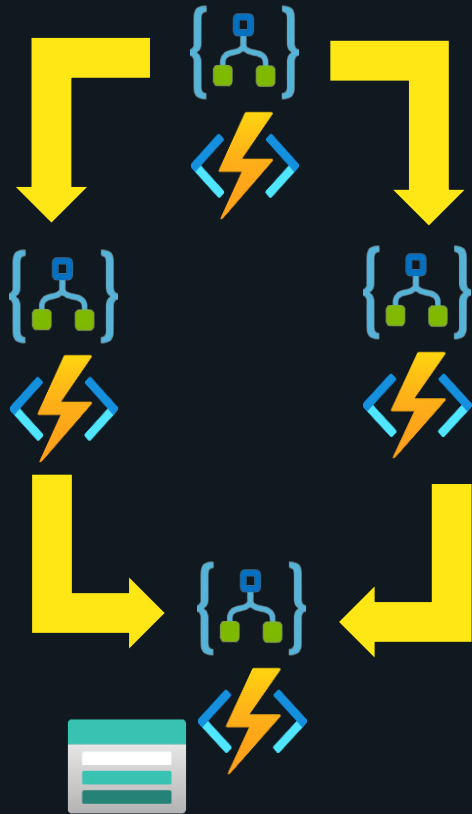


- 듀얼러블 펄션의 시간 제한 문제(2분)



- 웹훅 URL을 이용

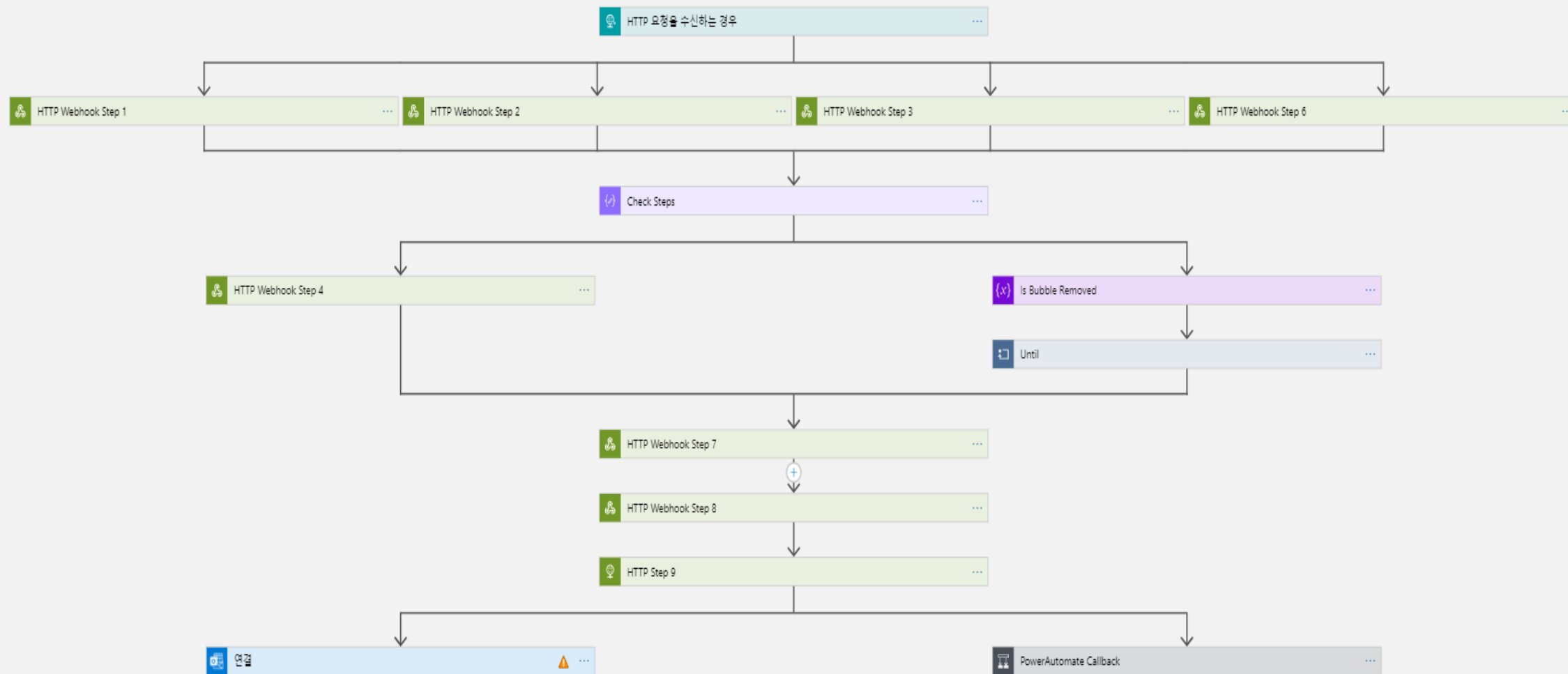
로직앱 작업 흐름 연결



블롭 저장소

- 알고리즘에 따른 로직앱 간의 유기적 연결
- 마지막 로직앱에서 블롭 저장소의 이미지 호출

로직앱 구성



로직앱 입력 값

HTTP Webhook Step 1	HTTP Webhook Step 2	HTTP Webhook Step 3
<p>*구독 - 메서드</p> <p>POST</p>	<p>*구독 - 메서드</p> <p>POST</p>	<p>*구독 - 메서드</p> <p>POST</p>
<p>*구독 - URI</p> <p>step01Endpoint x</p>	<p>*구독 - URI</p> <p>step02Endpoint x</p>	<p>*구독 - URI</p> <p>step03Endpoint x</p>
<p>구독 - 본문</p> <pre>{ "boughtSlicedGaraetteok": boughtSlicedGaraetteok x, "callbackUrl": listCallbackUrl() x, "timeToSoakInMinutes": timeToSoakInMinutes x }</pre>	<p>구독 - 본문</p> <pre>{ "callbackUrl": listCallbackUrl() x, "timeToSliceInMinutes": timeToSliceInMinutes x }</pre>	<p>구독 - 본문</p> <pre>{ "callbackUrl": listCallbackUrl() x, "timeToStirFryInMinutes": timeToStirFryInMinutes x }</pre>
<p>구독 취소 - 메서드</p> <p></p>	<p>구독 취소 - 메서드</p> <p></p>	<p>구독 취소 - 메서드</p> <p></p>
<p>구독 취소 - URI</p> <p></p>	<p>구독 취소 - URI</p> <p></p>	<p>구독 취소 - URI</p> <p></p>
<p>구독 취소 - 본문</p> <p></p>	<p>구독 취소 - 본문</p> <p></p>	<p>구독 취소 - 본문</p> <p></p>
<p>Add new parameter</p>	<p>Add new parameter</p>	<p>Add new parameter</p>

파워앱 / 파워 오토메이트



파워앱



파워 오토메이트

- 파워앱에서 파워 오토메이트를 호출
- 파워 오토메이트에서 로직앱을 호출
- 최종적으로 파워 오토메이트에서 파워앱으로 노티



Time to soak in mins

Time to slice in mins

Time to stir-fry in mins

Time to boil in mins

youjin@studentambassadors.com

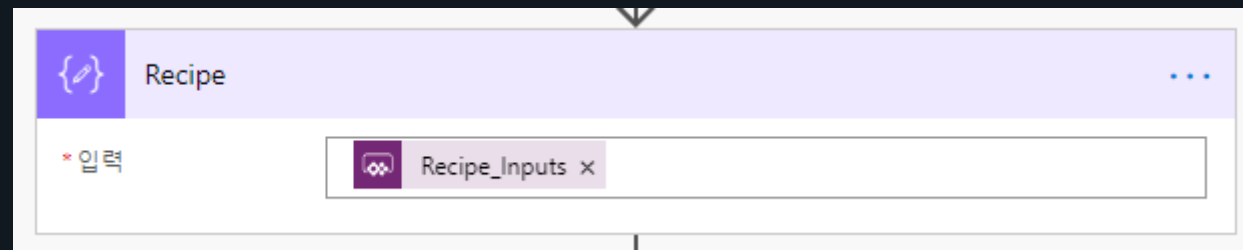
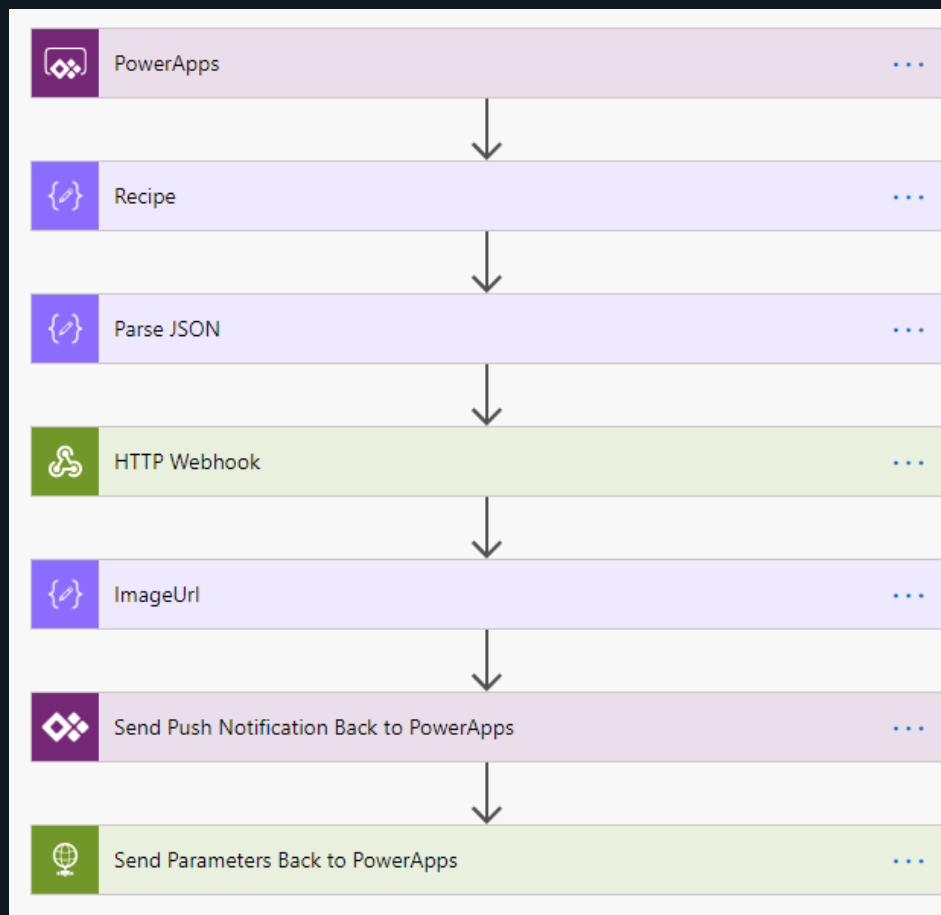
☒ I bought sliced garae-tteok!

☒ I want to add pepper!

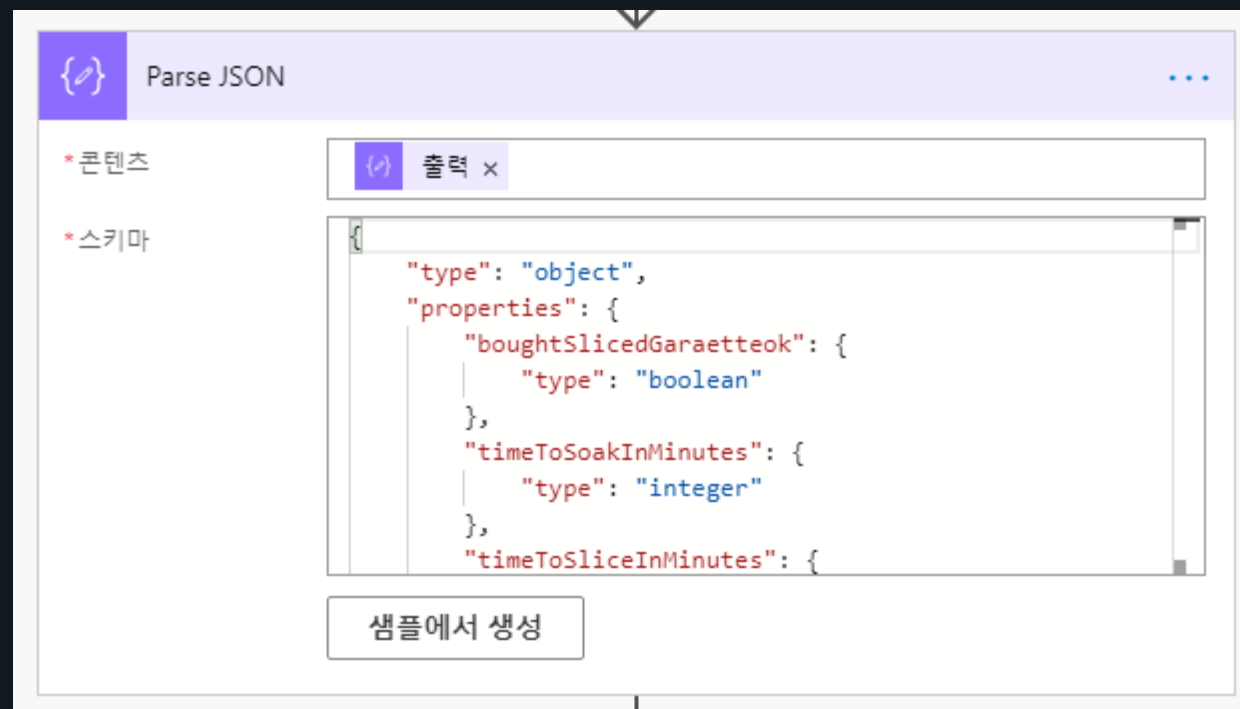
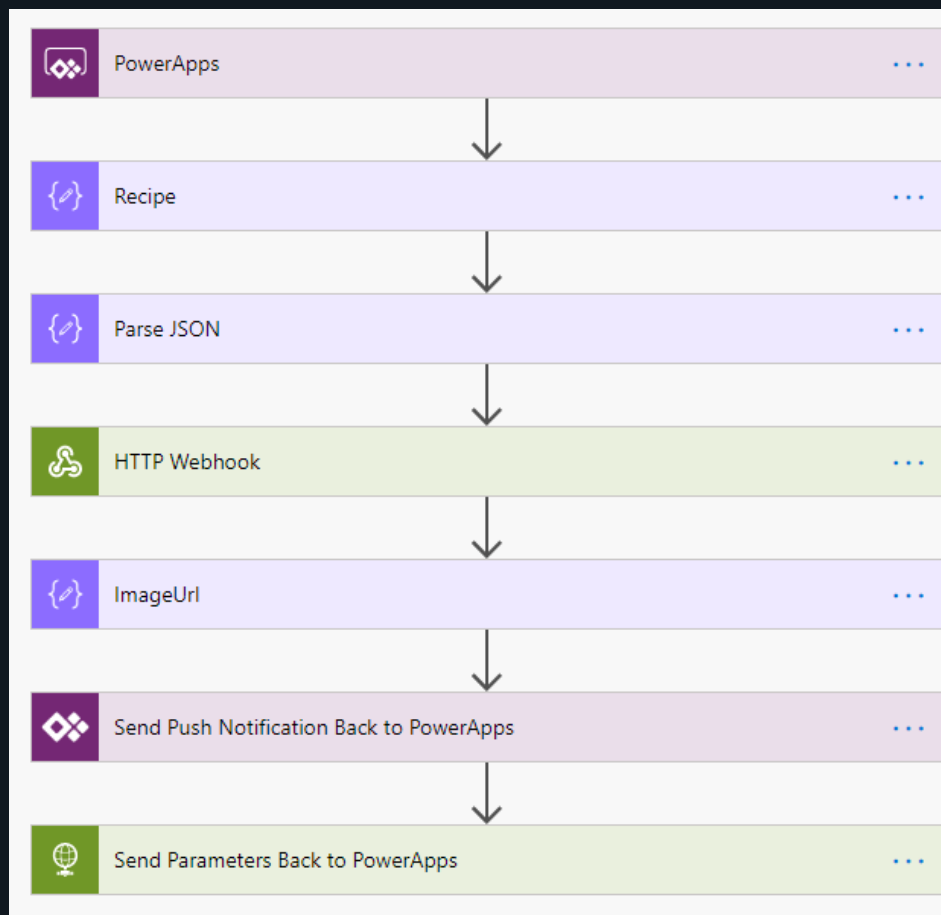
Cook for me!

- 정수값 입력 필드 (시간)
- 문자열 입력 필드 (이메일)
- 불리언 체크박스 (불리언)
- 실행 버튼

파워 오토메이트



파워오토메이트



The screenshot shows the configuration for the 'Parse JSON' action. The 'Content' field is set to '출력 x' (Output x). The 'Schema' field is set to a JSON schema defining the structure of the data being parsed.

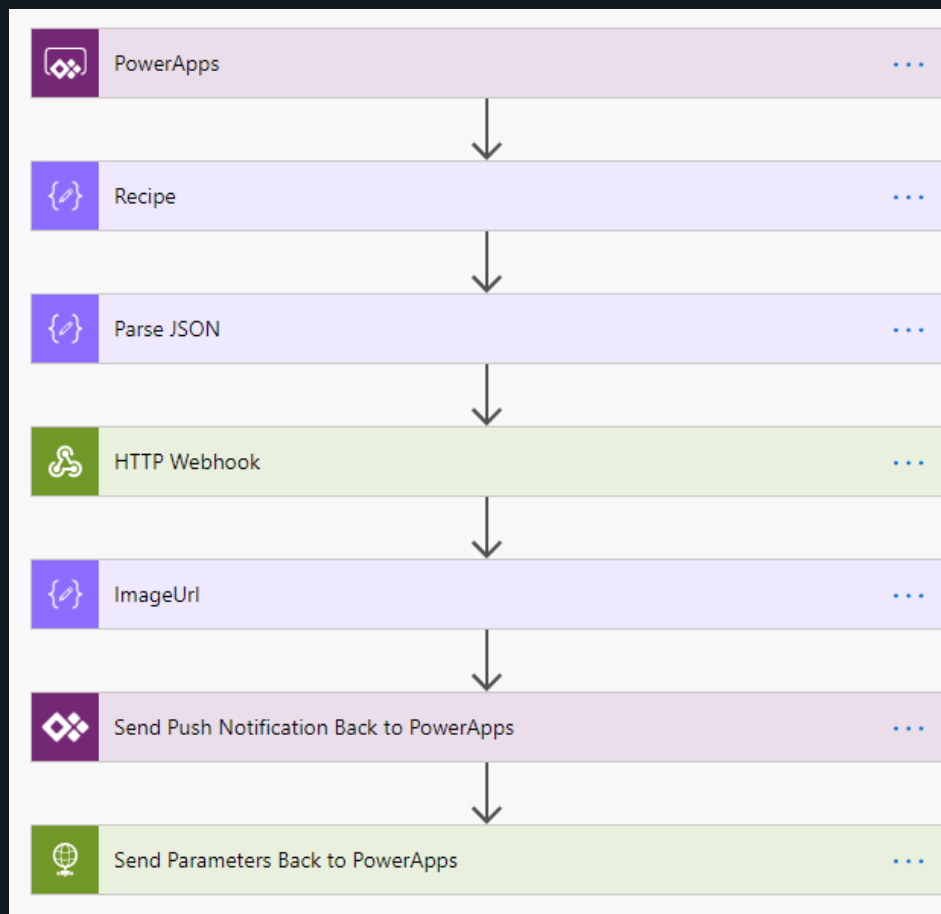
*콘텐츠: 출력 x

*스키마

```
{  
  "type": "object",  
  "properties": {  
    "boughtSlicedGaraetteok": {  
      "type": "boolean"  
    },  
    "timeToSoakInMinutes": {  
      "type": "integer"  
    },  
    "timeToSliceInMinutes": {  
      "type": "integer"  
    }  
  }  
}
```

샘플에서 생성

파워 오토메이트



HTTP Webhook

* 구독 - 메서드: POST

* 구독 - URI: `https://prod-71.eastus.logic.azure.com:443/workflows/037eeae7361f4ff8a7d12e783b0df01e/triggers/manual/paths/invoke?api-version=2016-10-01&sp=%2Ftriggers%2Fmanual%2Frun&sv=1.0&sig=2B81EI26-AiQ5MTZ36WKF8Ax1TDPa0n6HHRl0khfJ4`

구독 - 본문

```
{
  "boughtSlicedGaraetteok": boughtSlicedG... x,
  "timeToSoakInMinutes": timeToSoakIn... x,
  "timeToSliceInMinutes": timeToSliceInM... x,
  "timeToStirFryInMinutes": timeToStirFryIn... x,
  "timeToBoilInMinutes": timeToBoilInMi... x,
  "pepper": pepper x,
  "email": email x,
  "callbackUrl": listCallbackUrl() x
}
```

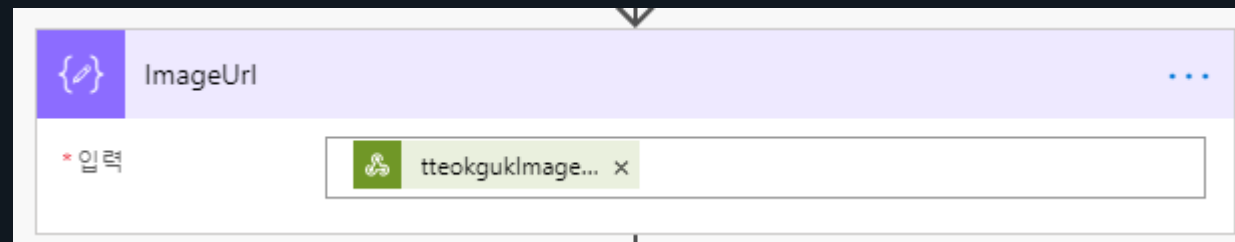
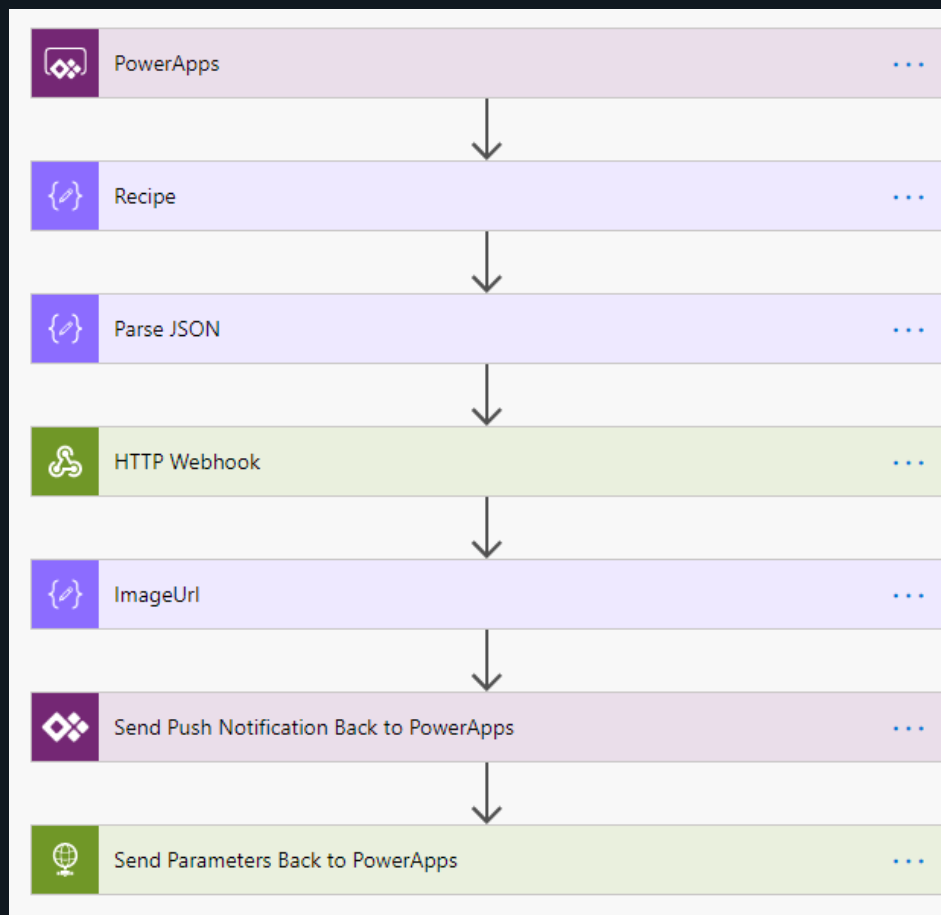
구독 취소 - 메서드:

구독 취소 - URI:

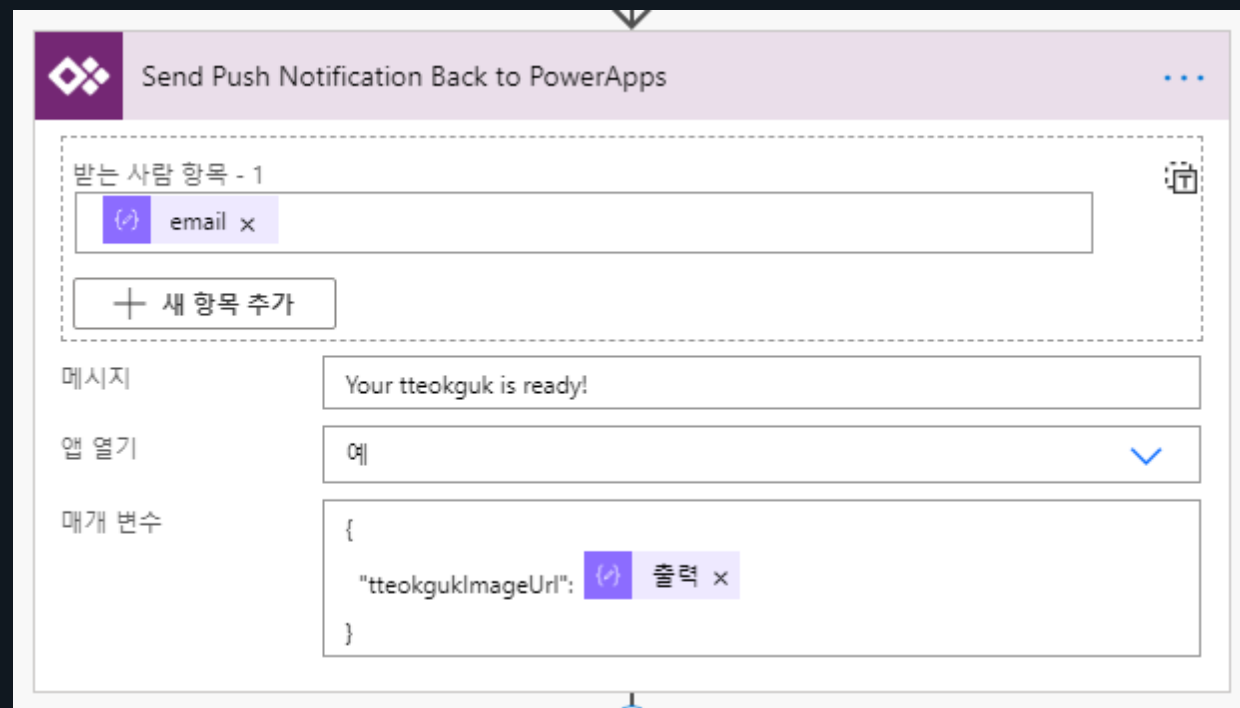
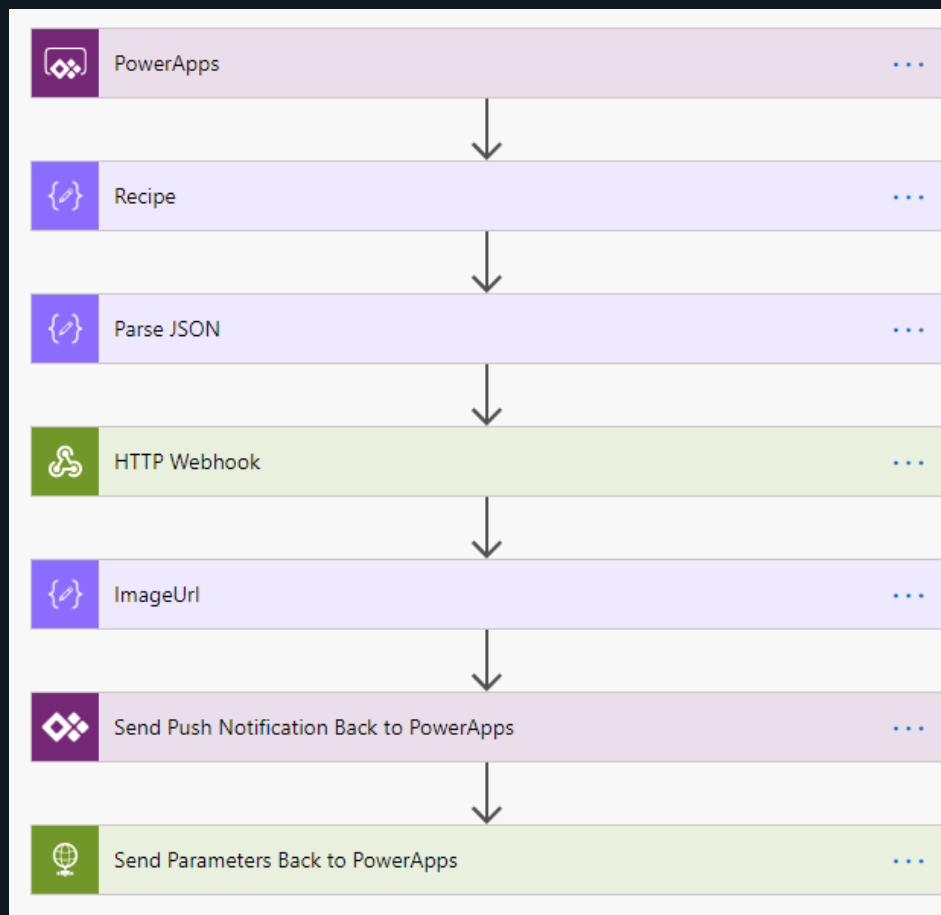
구독 취소 - 본문:

[고급 옵션 표시](#)

파워 오토메이트



파워 오토메이트



Send Push Notification Back to PowerApps

받는 사람 항목 - 1

email x

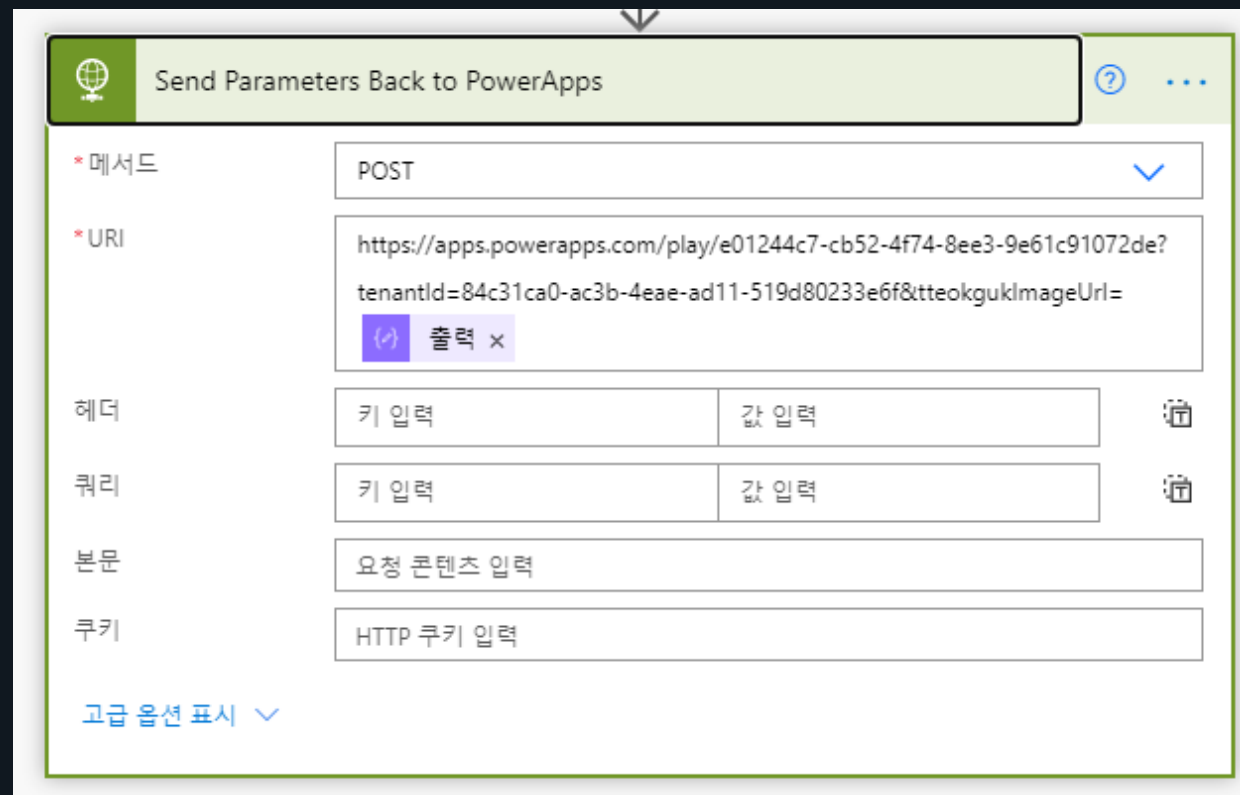
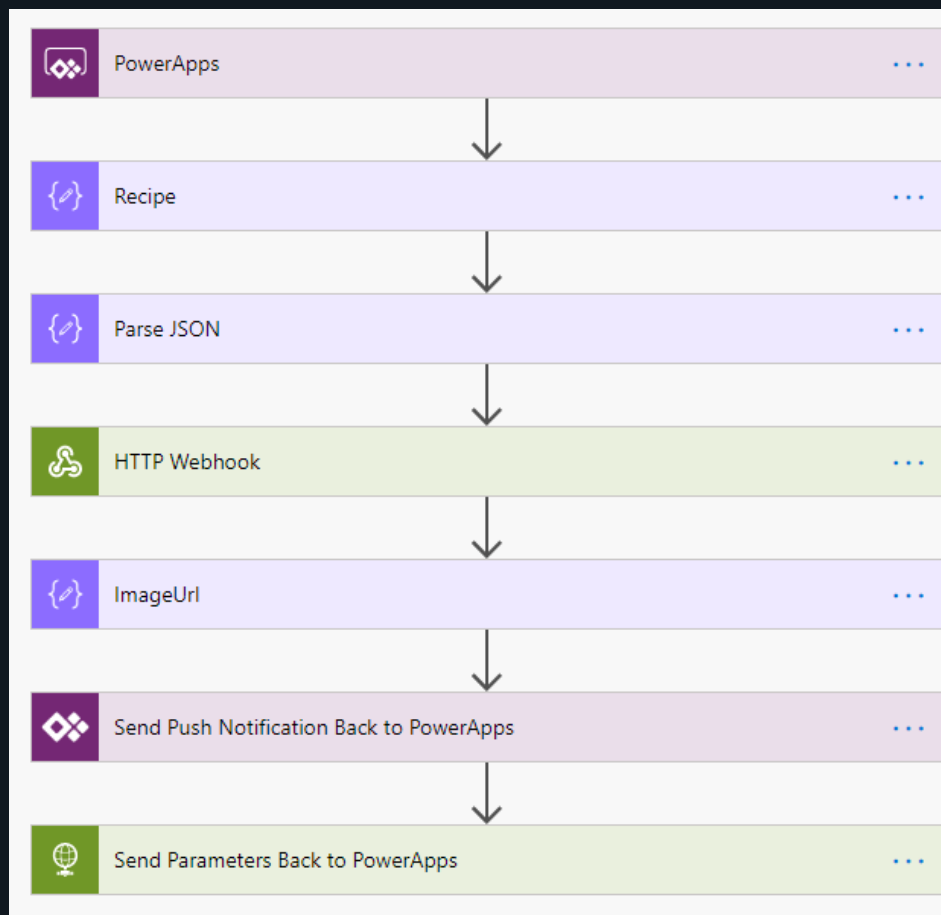
+ 새 항목 추가

메시지: Your tteokguk is ready!

앱 열기: 예

매개 변수: { "tteokgukImageUrl": 출력 x }

파워 오토메이트



The screenshot shows the configuration for the 'Send Parameters Back to PowerApps' action. The method is set to POST, and the URI is a PowerApps URL with a dynamic content placeholder for the tenantId. The headers, query, body, and cookies sections are visible, each with input fields for key and value.

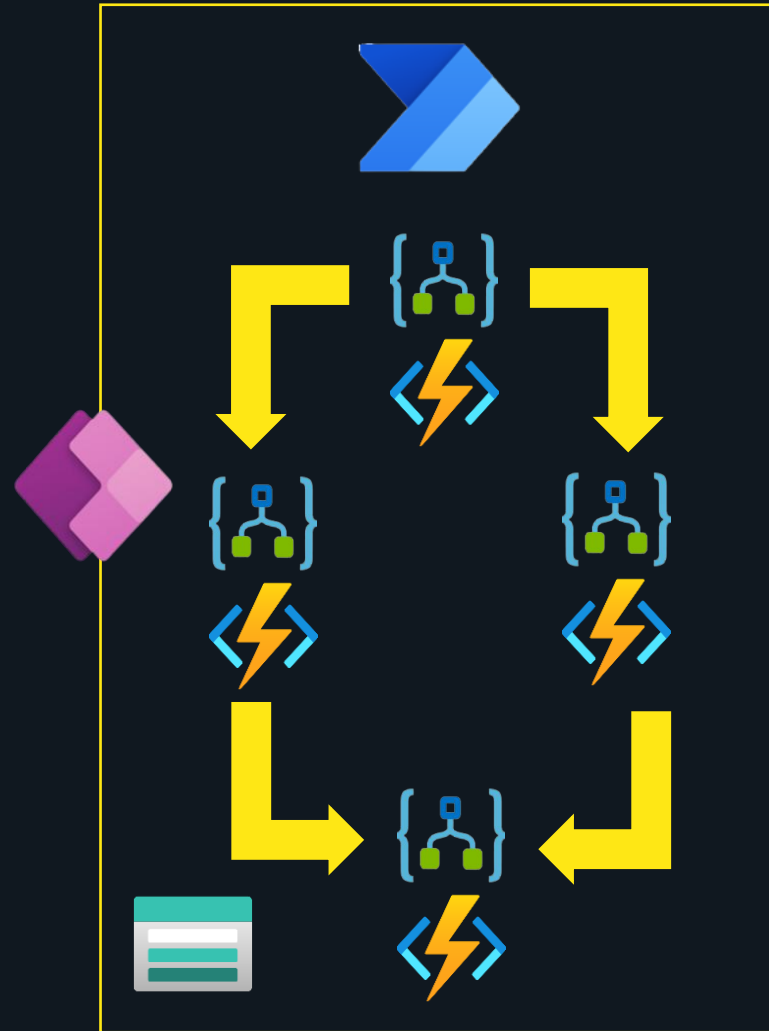
키 입력	값 입력	🗑️
키 입력	값 입력	🗑️

요청 콘텐츠 입력

HTTP 쿠키 입력

고급 옵션 표시 ▾

시스템 구조 설계



파이썬의 클라이언트 평션

```
async def main(req: func.HttpRequest, starter: str) -> func.HttpResponse:
    client = df.DurableOrchestrationClient(starter)

    instance_id = await client.start_new(req.route_params["functionName"], None, client_input=req.get_json())

    return client.create_check_status_response(req, instance_id)
```

- 오케스트레이터 평션 트리거
- 임의의 인스턴스 ID 생성
- 상태값 반환

파이썬의 오케스트레이터 평션

```
while not is_fried:
    time = context.current_utc_datetime
    if is_first:
        # First time to fry
        delay_time = time + timedelta(minutes=TIME_TO_STIR)
        is_first = False

    else:
        # Not fried yet
        delay_time = time + timedelta(minutes=1)

    yield context.create_timer(delay_time)
    result = yield context.call_activity('Fry', CALLBACK_URL)

    is_fried = result
```

- 현재 시간 값 + timedelta = 타이머 시간
- 작업 흐름에 따른 조건 분기
- 액티비티 평션 호출 및 결과 값을 변수에 할당

파이썬의 액티비티 평션

```
def main(req: func.HttpRequest) -> dict:
    bool_data = [True, False]
    pick_data = choice(bool_data)

    if pick_data:
        # Must execute when pick_data is True
        logging.info(req)
        requests.post(
            req,
            headers={"Content-Type": "application/json"},
            data=json.dumps({
                "completed": pick_data
            })
        )

    return pick_data
```

- 임의로 정해지는 불리언 값
- 조건이 충족되면 웹훅 URL 반환

폴리그랏(Polyglot)



```
module.exports = async function (context, req) {  
  const client = df.getClient(context);
```



```
async def main(req: func.HttpRequest, starter: str) -> func.HttpResponse:  
  client = df.DurableOrchestrationClient(starter)
```



```
public static class Step2 {  
  private static HttpClient httpClient = new HttpClient();
```


감사합니다