# Как стримить данные из Snowflake в Couchbase / плагин для Redpand/Connect

Александр Ванюшкин

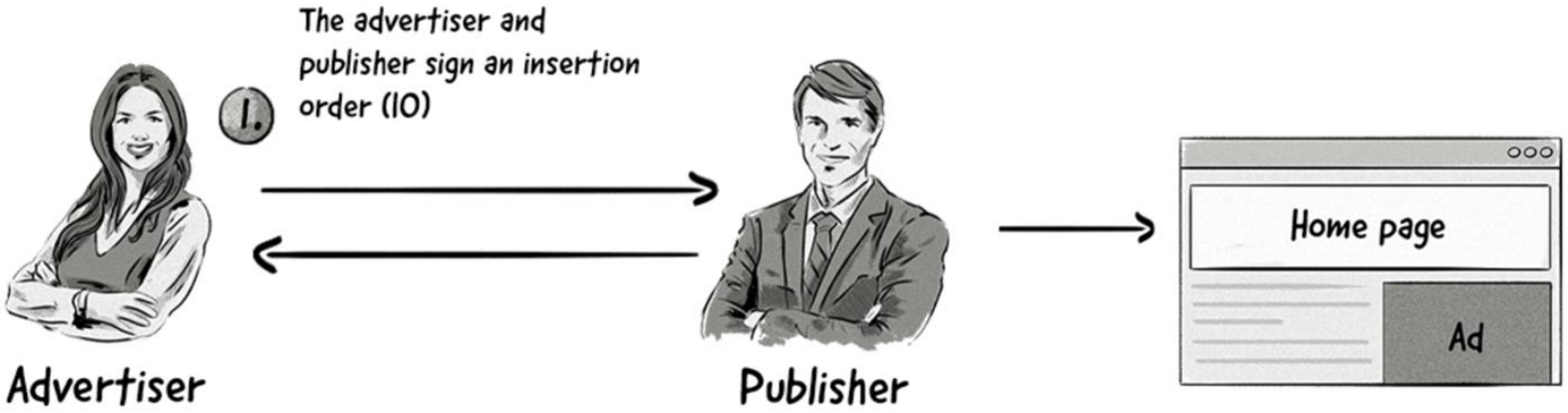
# Обо мне

- Разрабатывал на C#
- Разработчик в Weborama (Ad Tech)
- Автор курса по Go в Яндекс Лицее

# О чём поговорим

- Media buying and advertising
- Real-time bidding (RTB)
- Snowflake
- Redpanda/Connect (ex Benthos)

# Manual media buying



# Programmatic media buying

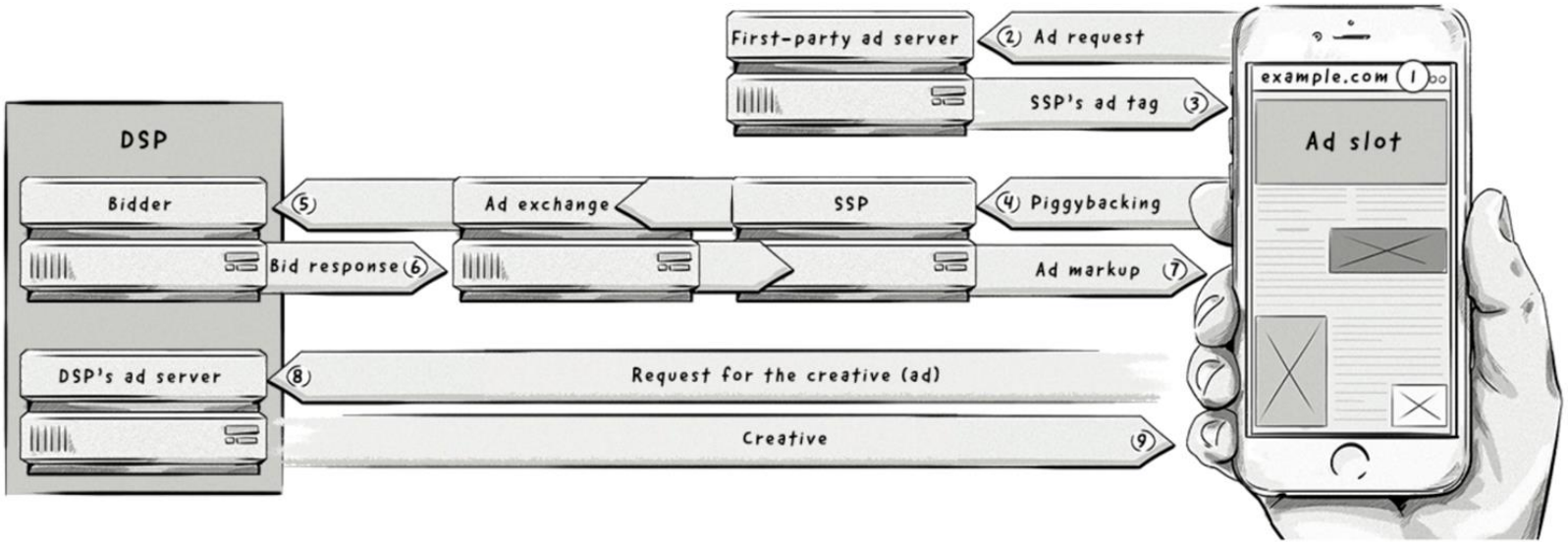
# RTB - аукцион

# RTB


Publisher
Real-time bidding
Advertisers
AD
Ad request
Bid request
BID
BID
BID
$1.15
$1
$1.25
Ad delivery
$1.25
Winner!

# Bid request example

```
"site": {
    "id": "102855",
    "cat": ["IAB3-1"],
    "domain": "www.foobar.com",
    "page": "http://www.foobar.com/1234.html ",
    "publisher": {
        "id": "8953",
        "name": "foobar.com",
        "cat": ["IAB3-1"],
        "domain": "foobar.com"
    }
},
"device": {
    "ua": "Mozilla/5.0 (Macintosh; Intel Mac OS X 10_6_8) AppleWebKit/537.13
    Version/5.1.7 Safari/534.57.2",
    "make": "Apple",
    "model": "iPhone",
    "ip": "123.145.167.10"
},
```




**DSP settings**

- Max. CPM bid
- Frequency capping
- Targeting criteria
- Pacing

**Bid response**

- Bid price
- Ad markup
- Seat ID

**Bid request**

- Ad format
- IAB category
- Browser & OS
- Geolocation
- IP address

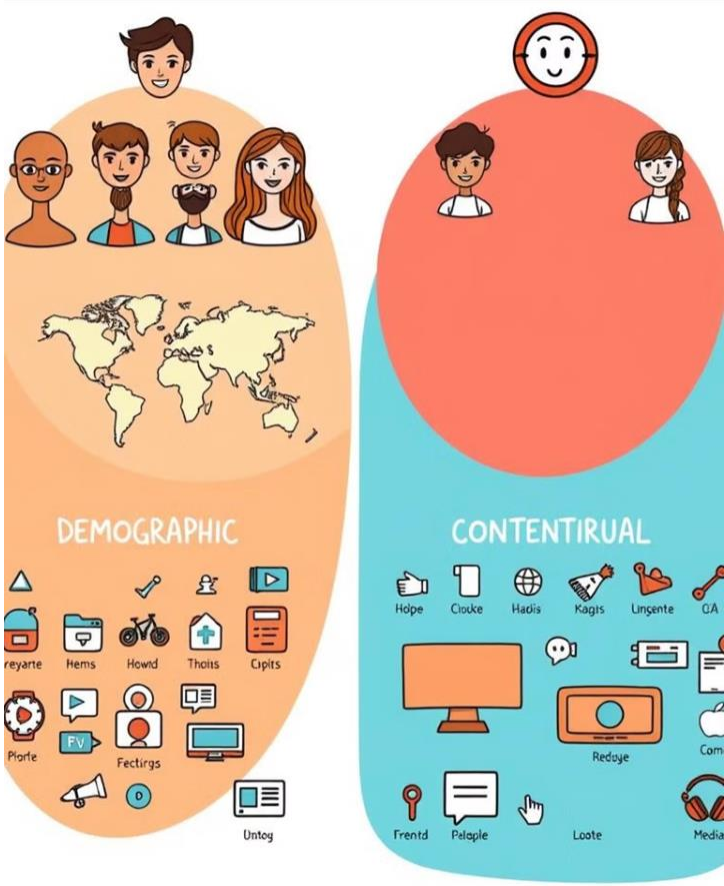
**SSP settings**

- Bid floor
- Placement settings
- Placement ad tag

# Ad Targeting

# Types of Ad Targeting

1. Demographic
2. Geographic
3. Behavioral
4. Contextual



# Contextual Targeting in Action



1. Content Scan
2. Ad Request
3. AdTech Platforms
4. Demand-Side Platforms
5. Ad Display

https://api.com?url=https://weird-site.com


```
"categories": [
        "category_1",
        "category_2"
 ],
```

# Snowflake storage

| | URL | REACH | LANGUAGE | SEGMENT_ID | SEGMENT_NAME | JOB_ID | SCORE |
|---|---|---|---|---|---|---|---|
| 1 | www.ac-pa | 4 | fr | 333173 | [Preprod-Test]- Infos | 302999 | 2 |
| 2 | pia.ac-pari | 2 | fr | 333173 | [Preprod-Test]- Infos | 302999 | 4 |
| 3 | www.ac-pa | 2 | fr | 333173 | [Preprod-Test]- Infos | 302999 | 2 |
| 4 | www.ac-pa | 2 | fr | 333173 | [Preprod-Test]- Infos | 302999 | 1 |
| 5 | www.ac-pa | 2 | fr | 333173 | [Preprod-Test]- Infos | 302999 | 3 |
| 6 | www.ac-pa | 2 | fr | 333173 | [Preprod-Test]- Infos | 302999 | 3 |

# Prepared data

Key: `weird-website.com`

Value:

```
{
  "categories": [
    "wr_gtctx_21074",
    "wr_gtctx_20974",
    "wr_gtctx_20975"
  ]
}
```

# Prepared data

Key: `weird-website.com`

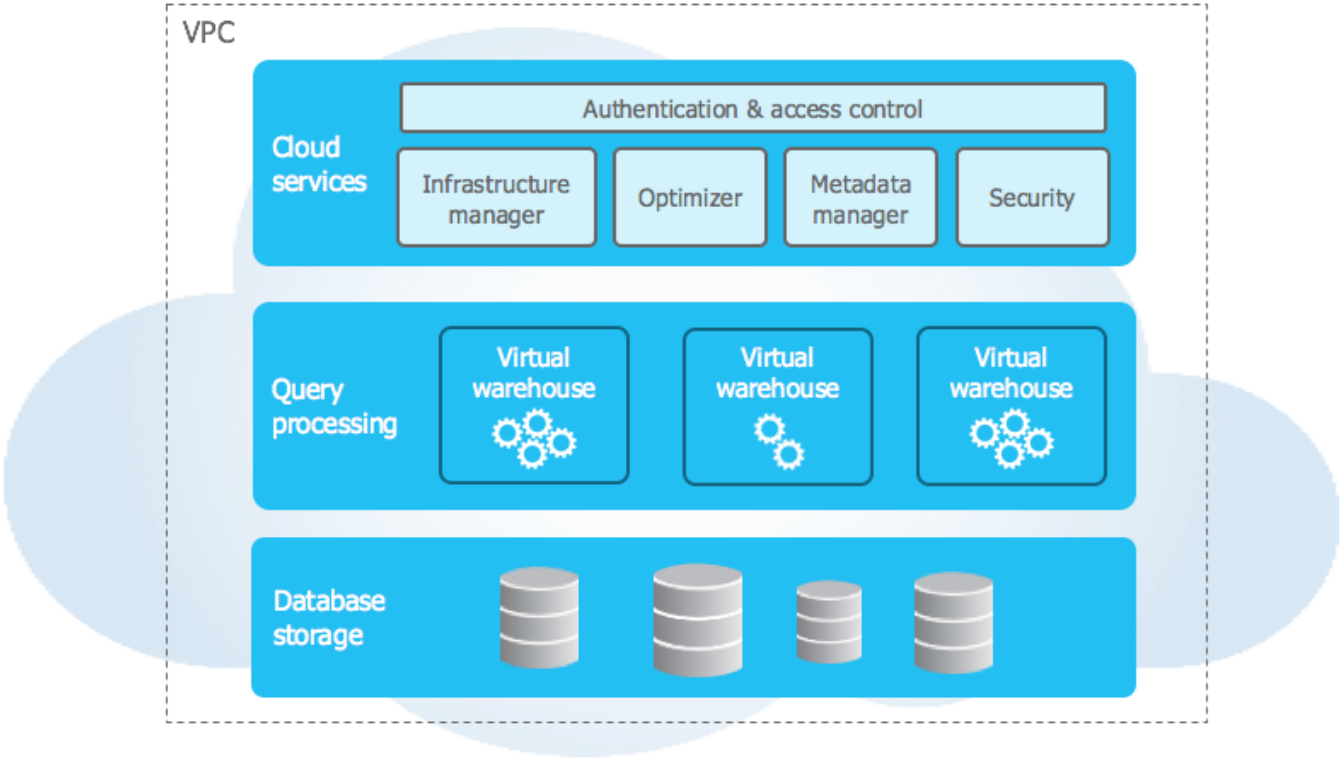
Value:

```
{
  "categories": [
    "wr_gtctx_21074",
    "wr_gtctx_20974",
    "wr_gtctx_20975"
  ]
}
```

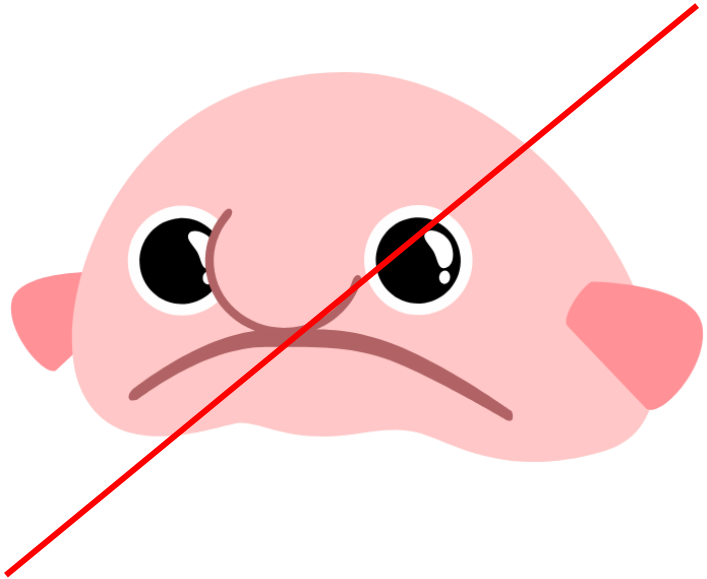
# API

- 40 000 QPS
- Response time up to 200 ms
- Satisfy format

# Tooling - Snowflake

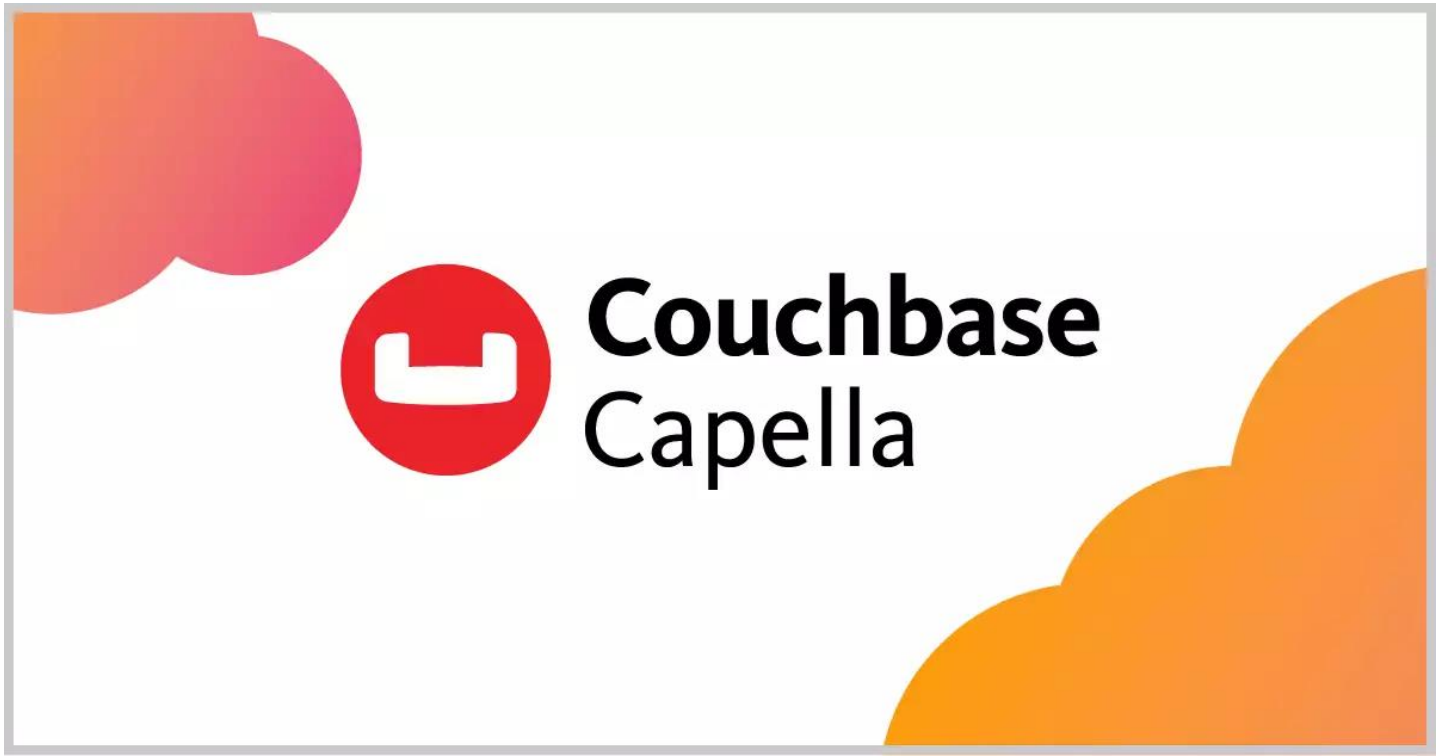


# Tooling - Redpanda/Connect (ex Benthos)



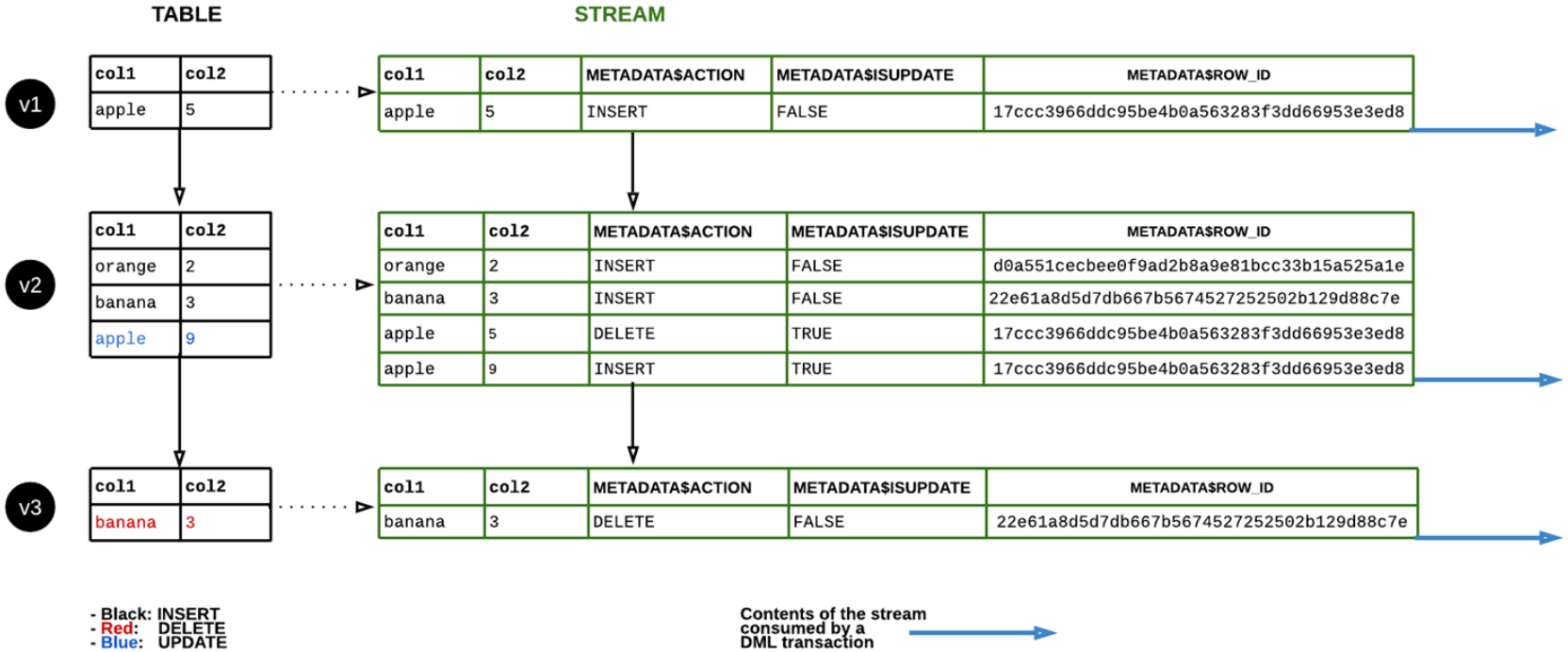
RedPanda

# Tooling Couchbase



# Tooling Golang

# Snowflake stream



# Snowflake stream

| | URL | LANGUAGE_CODE | SEGMENT_ID | METADATA$ACTION | METADATA$ISUPDATE |
|---|---|---|---|---|---|
| 1 | www.nouvel | fr | 333173 | INSERT | FALSE |
| 2 | www.lemonc | fr | 333173 | INSERT | FALSE |
| 3 | www.bfmtv.c | fr | 333173 | INSERT | FALSE |
| 4 | www.purepe | fr | 333173 | INSERT | FALSE |
| 5 | www.sudoue | fr | 333173 | INSERT | FALSE |
| 6 | www.lunion.f | fr | 333173 | INSERT | FALSE |
| 7 | www.challen | fr | 333173 | INSERT | FALSE |
| 8 | dict.leo.org/f | fr | 333173 | INSERT | FALSE |
| 9 | www.celibat | fr | 333173 | INSERT | FALSE |
| 10 | www.lemonc | fr | 333173 | INSERT | FALSE |

# Snowflake task

```
CREATE TASK URL_TASK
    SCHEDULE='USING CRON 0 * * * * UTC'
    AS COPY INTO "STAGE"
FROM
      (SELECT OBJECT_CONSTRUCT(*) FROM URL_STREAM)
      MAX_FILE_SIZE = 1048576
      FILE_FORMAT = (TYPE = JSON COMPRESSION = NONE);
```

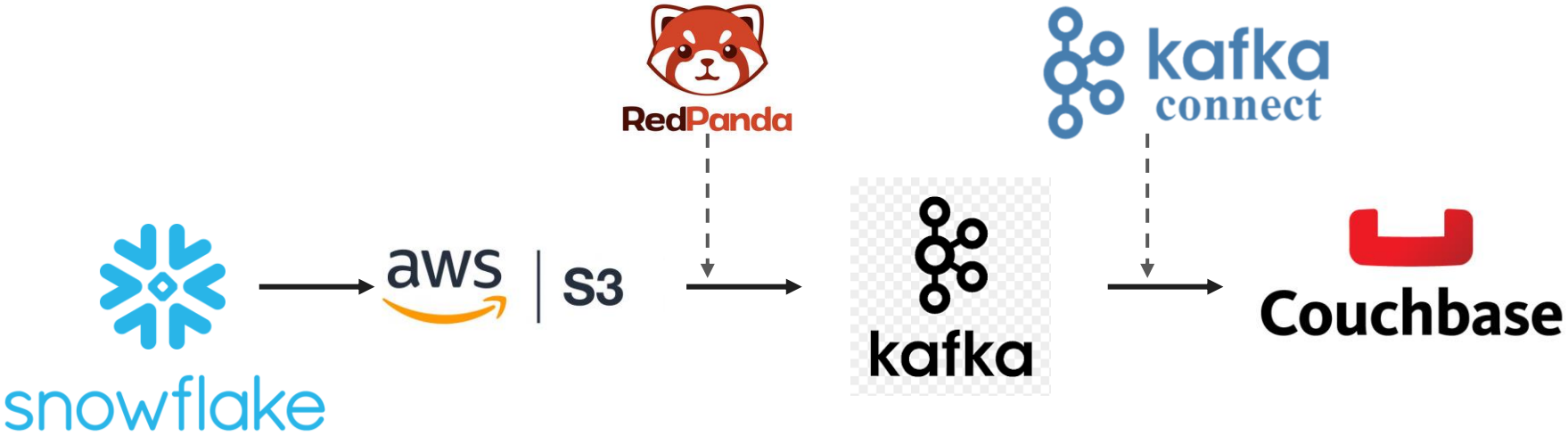
# Snowflake task

```
CREATE TASK URL_TASK
    SCHEDULE='USING CRON 0 * * * * UTC'
    AS COPY INTO "STAGE"
FROM
      (SELECT OBJECT_CONSTRUCT(*) FROM URL_STREAM)
      MAX_FILE_SIZE = 1048576
      FILE_FORMAT = (TYPE = JSON COMPRESSION = NONE);
```

# Snowflake task

```
CREATE TASK URL_TASK
    SCHEDULE='USING CRON 0 * * * * UTC'
    AS COPY INTO "STAGE"
FROM
      (SELECT OBJECT_CONSTRUCT(*) FROM URL_STREAM)
      MAX_FILE_SIZE = 1048576
      FILE_FORMAT = (TYPE = JSON COMPRESSION = NONE);
```

# Solution one



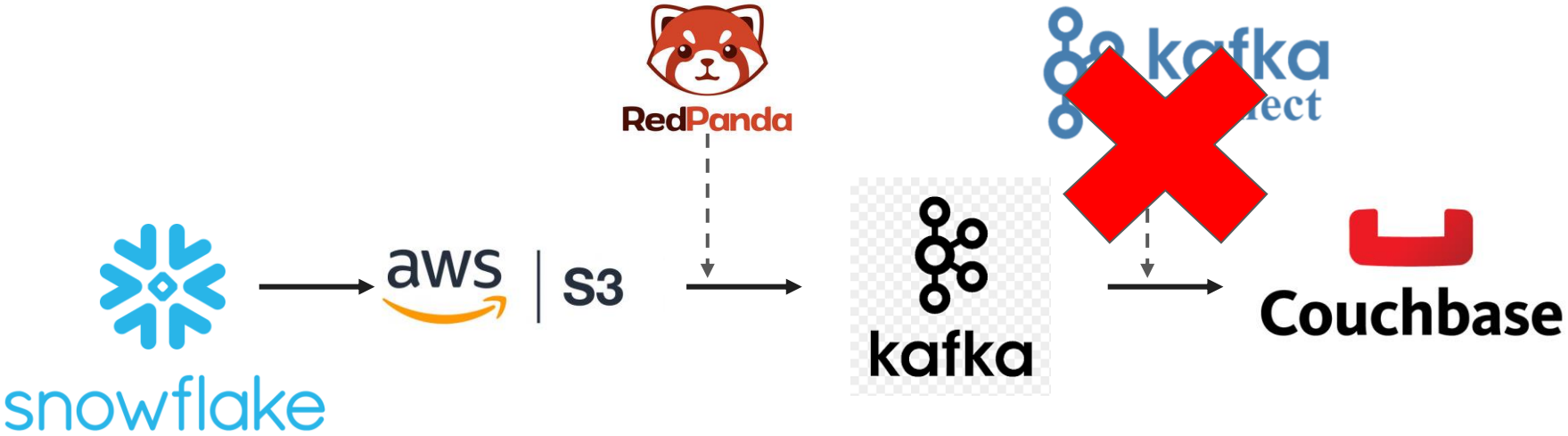
# Prepared data

Key: `weird-website.com`

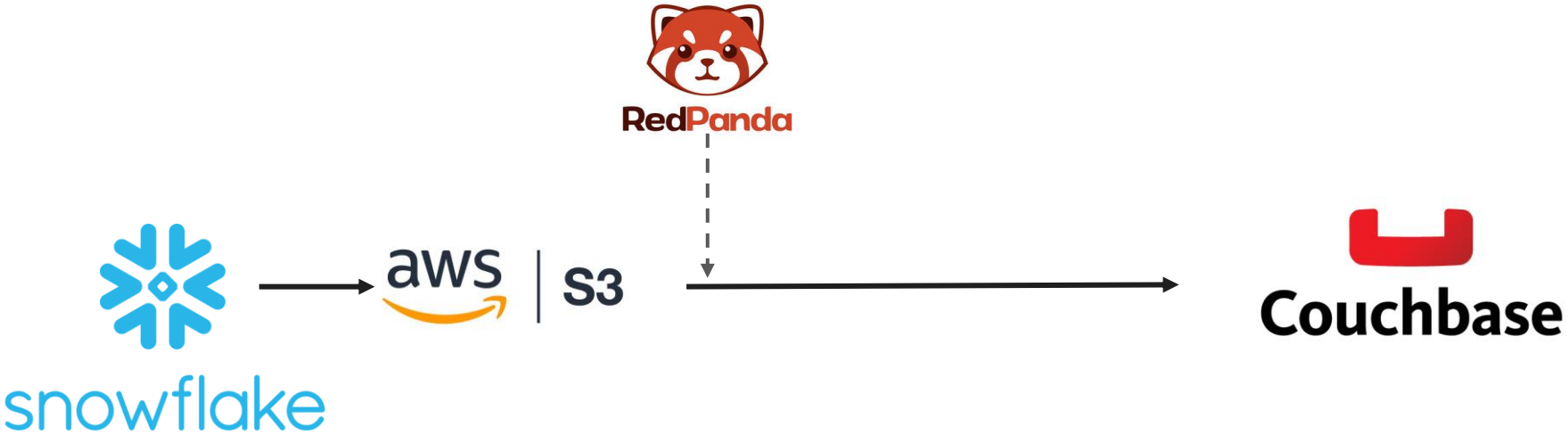
Value:

```
{
  "categories": [
    "wr_gtctx_21074",
    "wr_gtctx_20974",
    "wr_gtctx_20975"
  ]
}
```

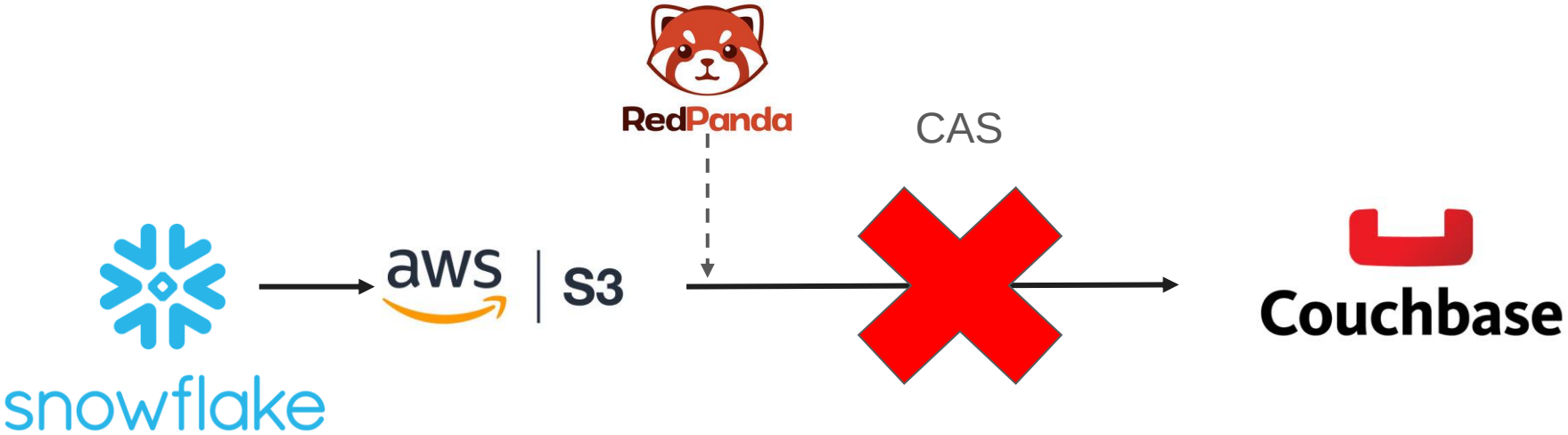
# Solution one



# Solution two



# Solution two



https://api.com?url=https://weird-site.com


```
"topics": [
        "topic_1",
        "topic_2"
 ],
```

```yaml
input:
  http_server:
    address: 0.0.0.0:4195
    path: /
    allowed_verbs: [POST]
output:
  sync_response: {}
  processors:
    - mapping: "root = content().uppercase()"
```

```yaml
input:
  http_server:
    address: 0.0.0.0:4195
    path: /
    allowed_verbs: [POST]
output:
  sync_response: {}
  processors:
    - mapping: "root = content().uppercase()"
```

```
input:
  http_server:
    address: 0.0.0.0:4195
    path: /
    allowed_verbs: [POST]
output:
  sync_response: {}
  processors:
    - mapping: "root = content().uppercase()"
```

```
input:
  http_server:
    address: 0.0.0.0:4195
    path: /
    allowed_verbs: [POST]
output:
  sync_response: {}
  processors:
    - mapping: "root = content().uppercase()"
```

```
input:
  http_server:
    address: 0.0.0.0:4195
    path: /
    allowed_verbs: [POST]
output:
  sync_response: {}
  processors:
    - mapping: "root = content().uppercase()"
```

```
curl -d "https://ria.ru" -X POST http://localhost:4444
```

```
HTTPS://RIA.RU
```

```go
package main

import "github.com/redpanda-data/benthos/v4/public/service"


func main() {
    service.RunCLI(context.Background())
}
```

```
func UrlContent(url string) string {
    resp, _ := http.Get(url)
    defer resp.Body.Close()

    body, _ := io.ReadAll(resp.Body)

    return html2text.HTML2Text(string(body))
}
```

```
func UrlContent(url string) string {
    resp, _ := http.Get(url)
    defer resp.Body.Close()

    body, _ := io.ReadAll(resp.Body)

    return html2text.HTML2Text(string(body))
}
```

# Bloblang



```
pSpec := bloblang.NewPluginSpec()

urlContentMethod := bloblang.StringMethod(
    func(url string) (interface{}, error) {
        return UrlContent(url)
    })

bloblang.RegisterMethodV2("url_content", pSpec,
 func(args *bloblang.ParsedParams) (bloblang.Method, error) {
                return urlContentMethod, nil
    })
```

# Bloblang



```
pSpec := bloblang.NewPluginSpec()

urlContentMethod := bloblang.StringMethod(
    func(url string) (interface{}, error) {
        return UrlContent(url)
    })

bloblang.RegisterMethodV2("url_content", pSpec,
 func(args *bloblang.ParsedParams) (bloblang.Method, error) {
                return urlContentMethod, nil
    })
```

# Bloblang



```
pSpec := bloblang.NewPluginSpec()

urlContentMethod := bloblang.StringMethod(
    func(url string) (interface{}, error) {
        return UrlContent(url)
    })

bloblang.RegisterMethodV2("url_content", pSpec,
 func(args *bloblang.ParsedParams) (bloblang.Method, error) {
                return urlContentMethod, nil
    })
```

# Bloblang



```
pSpec := bloblang.NewPluginSpec()

urlContentMethod := bloblang.StringMethod(
    func(url string) (interface{}, error) {
        return UrlContent(url)
    })

bloblang.RegisterMethodV2("url_content", pSpec,
 func(args *bloblang.ParsedParams) (bloblang.Method, error) {
                return urlContentMethod, nil
    })
```

```
input:
  http_server:
    address: 0.0.0.0:4444
    path: /
    allowed_verbs: [POST]
output:
  sync_response: {}
  processors:
    - mapping: "root = content().url_content()"
```

```
curl -d "https://www.culture.ru/s/nizhnij-novgorod/"
-X POST http://localhost:4444
```

Гуляем по древнему кремлю, заглядываем в современные музеи и смотрим закаты над ВолгойНИЖНИЙ НОВГОРОД:800 ЛЕТ ИСТОРИИГуляем по древнему кремлю, заглядываем в современные музеи и смотрим закаты над Волгой В 2021 году свое 800-летие отмечает Нижний Новгород – один из древнейших российских городов. В Средневековье он был крепостью, которая защищала границы Руси от набегов татар. В XVI веке здесь появилась одна из крупнейших ярмарок страны, а к XIX столетию город стал одним из главных промышленных центров России. Портал «Культура.РФ» представляет гид по Нижнему: познакомьтесь с историей города и узнайте, как интересно провести в нем время.В 2021 году свое 800-летие отмечает Нижний Новгород – один из древнейших российских городов. В Средневековье он был крепостью, которая защищала границы Руси от набегов татар. В XVI веке здесь появилась одна из крупнейших ярмарок страны, а к XIX столетию город стал одним из главных промышленных центров России. Портал «Культура.РФ» представляет гид по Нижнему: познакомьтесь с историей города и узнайте, как интересно провести в нем время.В 2021 году свое 800-летие отмечает Нижний Новгород – …

```
type Processor interface {
    Process(context.Context,*Message)(MessageBatch, error)
    Closer
}
```

# 🦜🔗 LangChain Go

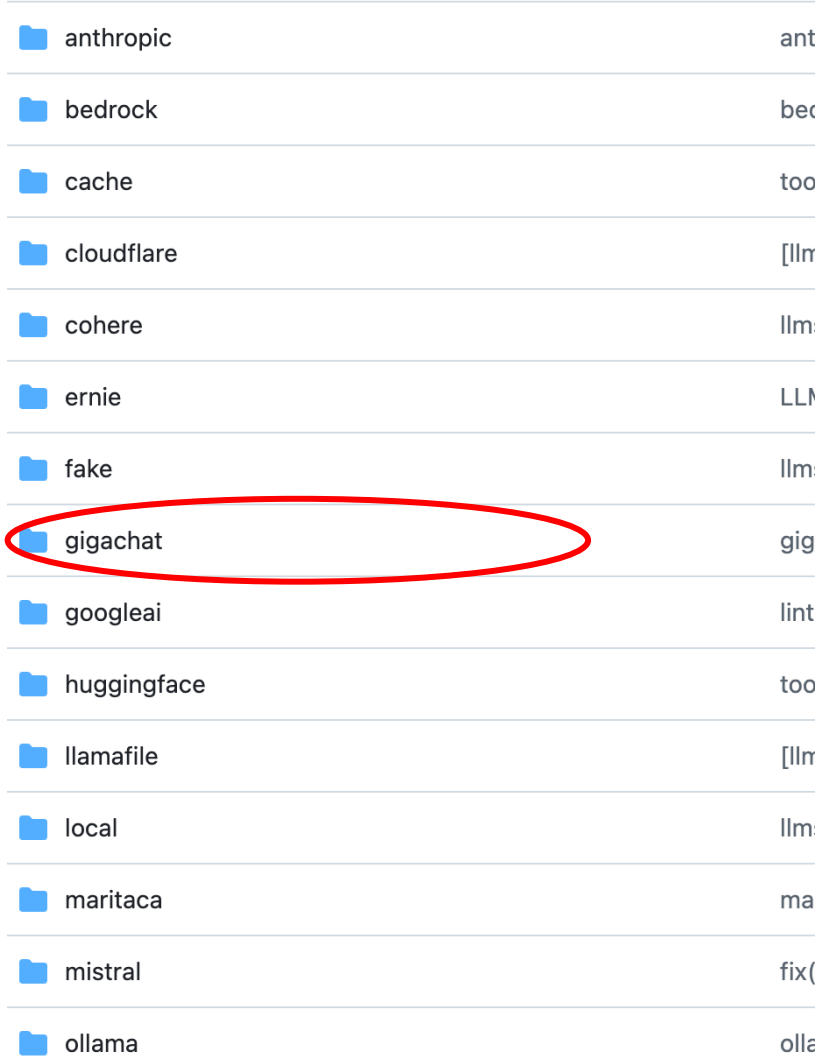

⚡ Building applications with LLMs through composability, with Go! ⚡

## 🤔 What is this?

This is the Go language implementation of LangChain.

# https://github.com/tmc/langchaingo

```
type Model interface {
    GenerateContent(
        ctx context.Context,
        messages []MessageContent,
        options ...CallOption,
    ) (*ContentResponse, error)
}
```


anthropic ant
bedrock bed
cache too
cloudflare [llm
cohere llm
ernie LLM
fake llm
gigachat gig
googleai lint
huggingface too
llamafile [llm
local llm
maritaca ma
mistral fix(
ollama olla

```
type AI struct {
    llm    llms.Model
    prompt string
}
```

```
configSpec := service.NewConfigSpec().
        Field(service.NewStringField("prompt"))


processor := func(conf *service.ParsedConfig, mgr *service.Resources)
(service.Processor, error) {
    prompt, _ := conf.FieldString("prompt")
    llm, _ := gigachat.New()

    return &AI{llm: llm, prompt: prompt}, nil
}


service.RegisterProcessor("ai", configSpec, processor)
```

```
configSpec := service.NewConfigSpec().
        Field(service.NewStringField("prompt"))

processor := func(conf *service.ParsedConfig, mgr *service.Resources)
(service.Processor, error) {
    prompt, _ := conf.FieldString("prompt")
    llm, _ := gigachat.New()

    return &AI{llm: llm, prompt: prompt}, nil
}

service.RegisterProcessor("ai", configSpec, processor)
```

```
configSpec := service.NewConfigSpec().
        Field(service.NewStringField("prompt"))

processor := func(conf *service.ParsedConfig, mgr *service.Resources)
(service.Processor, error) {
    prompt, _ := conf.FieldString("prompt")
    llm, _ := gigachat.New()

    return &AI{llm: llm, prompt: prompt}, nil
}

service.RegisterProcessor("ai", configSpec, processor)
```

```
configSpec := service.NewConfigSpec().
        Field(service.NewStringField("prompt"))


processor := func(conf *service.ParsedConfig, mgr *service.Resources)
(service.Processor, error) {
    prompt, _ := conf.FieldString("prompt")
    llm, _ := gigachat.New()

    return &AI{llm: llm, prompt: prompt}, nil
}


service.RegisterProcessor("ai", configSpec, processor)
```

```
func (ai *AI) Process(ctx context.Context, m *service.Message)
(service.MessageBatch, error) {

    llmChat := []llms.MessageContent{
        llms.TextParts(llms.ChatMessageTypeSystem, ai.prompt),
        llms.TextParts(llms.ChatMessageTypeHuman, string(m.AsBytes()))}

    resp, _ := ai.llm.GenerateContent(ctx, llmChat)

    m.SetBytes([]byte(resp.Choices[0].Content))

    return service.MessageBatch{m}, nil

}
```

```
func (ai *AI) Process(ctx context.Context, m *service.Message)
(service.MessageBatch, error) {

    llmChat := []llms.MessageContent{
        llms.TextParts(llms.ChatMessageTypeSystem, ai.prompt),
        llms.TextParts(llms.ChatMessageTypeHuman, string(m.AsBytes()))}

    resp, _ := ai.llm.GenerateContent(ctx, llmChat)

    m.SetBytes([]byte(resp.Choices[0].Content))

    return service.MessageBatch{m}, nil

}
```

```
func (ai *AI) Process(ctx context.Context, m *service.Message)
(service.MessageBatch, error) {

    llmChat := []llms.MessageContent{
        llms.TextParts(llms.ChatMessageTypeSystem, ai.prompt),
        llms.TextParts(llms.ChatMessageTypeHuman, string(m.AsBytes()))}

    resp, _ := ai.llm.GenerateContent(ctx, llmChat)

    m.SetBytes([]byte(resp.Choices[0].Content))

    return service.MessageBatch{m}, nil

}
```

```
func (ai *AI) Process(ctx context.Context, m *service.Message)
(service.MessageBatch, error) {

    llmChat := []llms.MessageContent{
        llms.TextParts(llms.ChatMessageTypeSystem, ai.prompt),
        llms.TextParts(llms.ChatMessageTypeHuman, string(m.AsBytes()))}

    resp, _ := ai.llm.GenerateContent(ctx, llmChat)

    m.SetBytes([]byte(resp.Choices[0].Content))

    return service.MessageBatch{m}, nil

}
```

```
func (ai *AI) Process(ctx context.Context, m *service.Message)
(service.MessageBatch, error) {

    llmChat := []llms.MessageContent{
        llms.TextParts(llms.ChatMessageTypeSystem, ai.prompt),
        llms.TextParts(llms.ChatMessageTypeHuman, string(m.AsBytes()))}

    resp, _ := ai.llm.GenerateContent(ctx, llmChat)

    m.SetBytes([]byte(resp.Choices[0].Content))

    return service.MessageBatch{m}, nil

}
```

```
func (ai *AI) Process(ctx context.Context, m *service.Message)
(service.MessageBatch, error) {

    llmChat := []llms.MessageContent{
        llms.TextParts(llms.ChatMessageTypeSystem, ai.prompt),
        llms.TextParts(llms.ChatMessageTypeHuman, string(m.AsBytes()))}

    resp, _ := ai.llm.GenerateContent(ctx, llmChat)

    m.SetBytes([]byte(resp.Choices[0].Content))

    return service.MessageBatch{m}, nil

}
```

```
pipeline:
  processors:
    - ai:
        prompt: >
          Ты система,
          которая определяет не менее трех основных тем текста.
          Верни ответ в виде json. Пример ответа для трех тем:
          {"topics": "погода, здоровье, зима"}
```

```
pipeline:
  processors:
    - ai:
        prompt: >
          Ты система,
          которая определяет не менее трех основных тем текста.
          Верни ответ в виде json. Пример ответа для трех тем:
          {"topics": "погода, здоровье, зима"}
```

```
pipeline:
  processors:
    - ai:
        prompt: >
          Ты система,
          которая определяет не менее трех основных тем текста.
          Верни ответ в виде json. Пример ответа для трех тем:
          {"topics": "погода, здоровье, зима"}
```

```yaml
input:
  http_server:
    address: 0.0.0.0:4444
    path: /
    allowed_verbs: [POST]
pipeline:
  processors:
    - ai:
        prompt: >
          Ты система, которая определяет не менее трех основных тем текста.
          Верни ответ в виде json. Пример ответа для трех тем:
          {"topics": "погода, здоровье, зима"}
output:
  sync_response: {}
  processors:
    - mapping: "root = content()"
```

```yaml
input:
  http_server:
    address: 0.0.0.0:4444
    path: /
    allowed_verbs: [POST]
pipeline:
  processors:
    - ai:
        prompt: >
          Ты система, которая определяет не менее трех основных тем текста.
          Верни ответ в виде json. Пример ответа для трех тем:
          {"topics": "погода, здоровье, зима"}
output:
  sync_response: {}
  processors:
    - mapping: "root = content()"
```

```
curl -d "https://www.culture.ru/s/nizhnij-novgorod/" -X POST
http://localhost:4444
```

```
{"topics": "Нижний Новгород, культура, туризм"}
```

# Solution three

# Выводы

- No-code решения заслуживают внимания, но…
- Попробуйте Redpanda/Connect

Александр Ванюшкин
dialog2312@gmail.com