Infrastructure

Сценарии в нагрузочной тестировании

Новые механизмы сценарной нагрузки
в Yandex Pandora

Сергей Бевзенко,
Старший разработчик
Группа нагрузочного тестирования

Сергей Бевзенко

- Yandex Cloud Loadtesting
- Команда инструментов нагрузочного тестирования
- Maintainer Yandex Pandora

Yandex Infrastructure

- Мы создаём надёжную и масштабируемую платформу для эффективной работы разработчиков Яндекса.
- Строим ЦОДы, обеспечиваем доступность по проводам и без, надежно храним данные, создаем удобные инструменты разработки и масштабируем внутреннее облако.
- Многие наши разработки можно попробовать в Yandex Cloud, а YDB и YTsaurus доступны в Open Source.

О чем доклад?

1

Про создание сценариев
в нагрузочном тестировании

2

Да.
Опять про сценарии.

Содержание

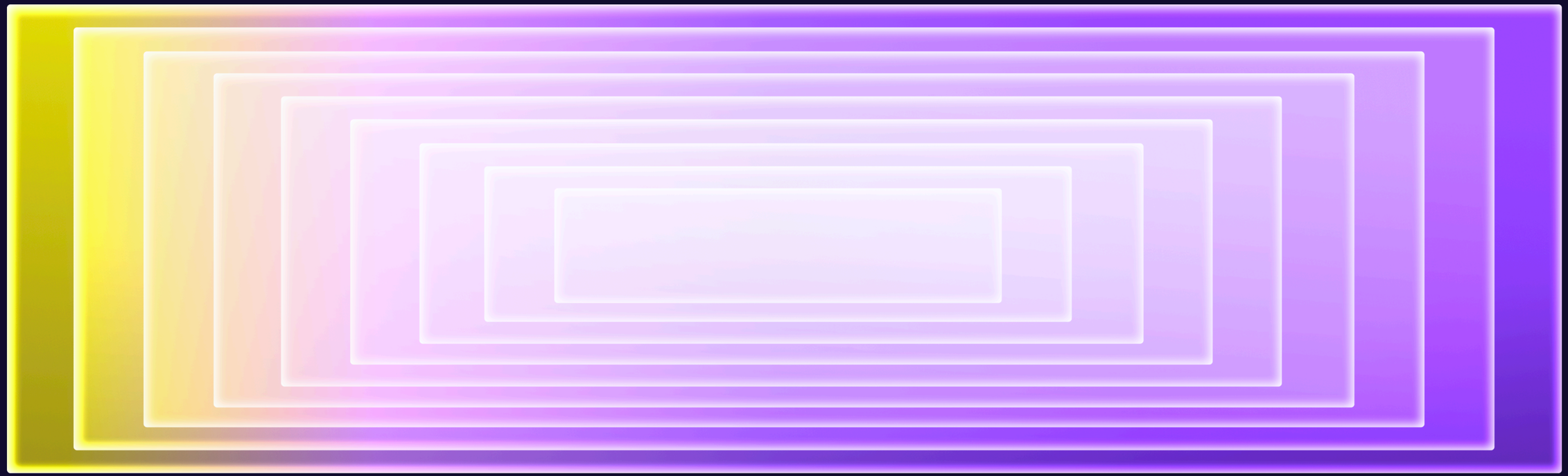
Yandex Pandora

- Go
- HTTP, HTTP2, gRPC
- uri, uripost, raw, json - payload
- custom load generator

Custom load generator

Алексей Лавренюк
Pandora: нагрузочные тесты в виде кода
https://www.youtube.com/watch?v=IkusMkIniq0

Дмитрий Кузнецов
Подготовка тестовых данных для нагрузочного тестирования протокола gRPC
https://www.youtube.com/watch?v=DKR6o6nEVLY



Custom load generator

Using Go

Спасибо.
Про сценарии всё сказано.

Типы payload

1. player: проигрыватель нагрузки
   - заранее подготовленный
   - сгенерированный
   - recorder
2. logic
   - сценарии

Пример подготовленных тестовых данных в Пандоре

uripost

```
[Host: example.com]
[User-Agent: Tank]
5 /route/?ll=50.262025%2C53.276083~50.056015
class
6 /route/?ll=50.262025%2C53.276083~50.056015
hello!
7 /route/?ll=37.565147%2C55.695758~37.412796
uripost
```

uri

```
[Host: example.com]
[User-Agent: Yandex-browser]
/?sleep=100
/?sleep=200
```

Пример подготовленных тестовых данных в Пандоре

raw

```
74 tag_1
GET /abra HTTP/1.0
Host: xxx.example.com
User-Agent: xxx (shell 1)

599 tag_2
POST /upload/2 HTTP/1.0
Content-Length: 496
Host: xxxxxxxxx.dev.example.com
User-Agent: xxx (shell 1)

^.^.1......W.j^1^.^.^.²..^^.i.^B.P..–!(.l/Y..V^.      ...L?...S'NR.^^vm...3Gg@s...d'.\^.5N.
$NF^,.Z^.aTE^.
._.[..k#L^ƨ'\RE.J.<.!,.q5.F^ʾiΔĬq..^6..P..ᴛH.'..i2
.".uuzs^^F2...Rh.&.U.^^.2J.P@.A......x..ləy^?.u.p{4..g...m.,..R^.^.^.3....].^^.^J...p.ifTF0<
.s.9V.o5<..%!6ĺS.ƐĀ..歡....C^&.....^.^y...v]^YT.1.#K.ibc...^.26...   ..7.
b.$...j6.ʌf...W.R7.^1.3....K'%.&^.4d..{{      l0..^\..^X.g.^.r.(!.^^.5.4.1.$\ .%.8$
(.n&..^^q.,.Q..^.D^.].^.R9.kE.^.$^.I..<..B^.6^.h^^C.^E.|....3o^.@..Z.^.s.$[v.
```

Пример подготовленных тестовых данных в Пандоре

json

```
{"host": "4bs65mu2kdulxmir.myt.yp-c.yandex.net", "method": "GET", "uri": "/?sleep=100",
"tag": “tag_1", "headers": {"User-agent": "Tank", "Connection": "close"}}
{"host": "4bs65mu2kdulxmir.myt.yp-c.yandex.net", "method": "POST", "uri": "/?sleep=200",
"tag": “tag_s”, "headers": {"User-agent": "Tank", "Connection": "close"}, "body":
«body_data"}
```

- host
- method
- uri
- headers
- body
- tag

Типы payload

1. Заранее подготовленные
2. **Сложная логика**

Реализация сценариев

1. Apache JMeter
   - GUI. На выходе XML или Kotlin
   - JMeterDSL. Сценарии на Java
     - https://github.com/abstracta/jmeter-java-dsl
2. Gatling — сценарии на Java, Kotlin
3. k6 — сценарии на Javascript, DSL json
4. Pandora — сценарии на Go

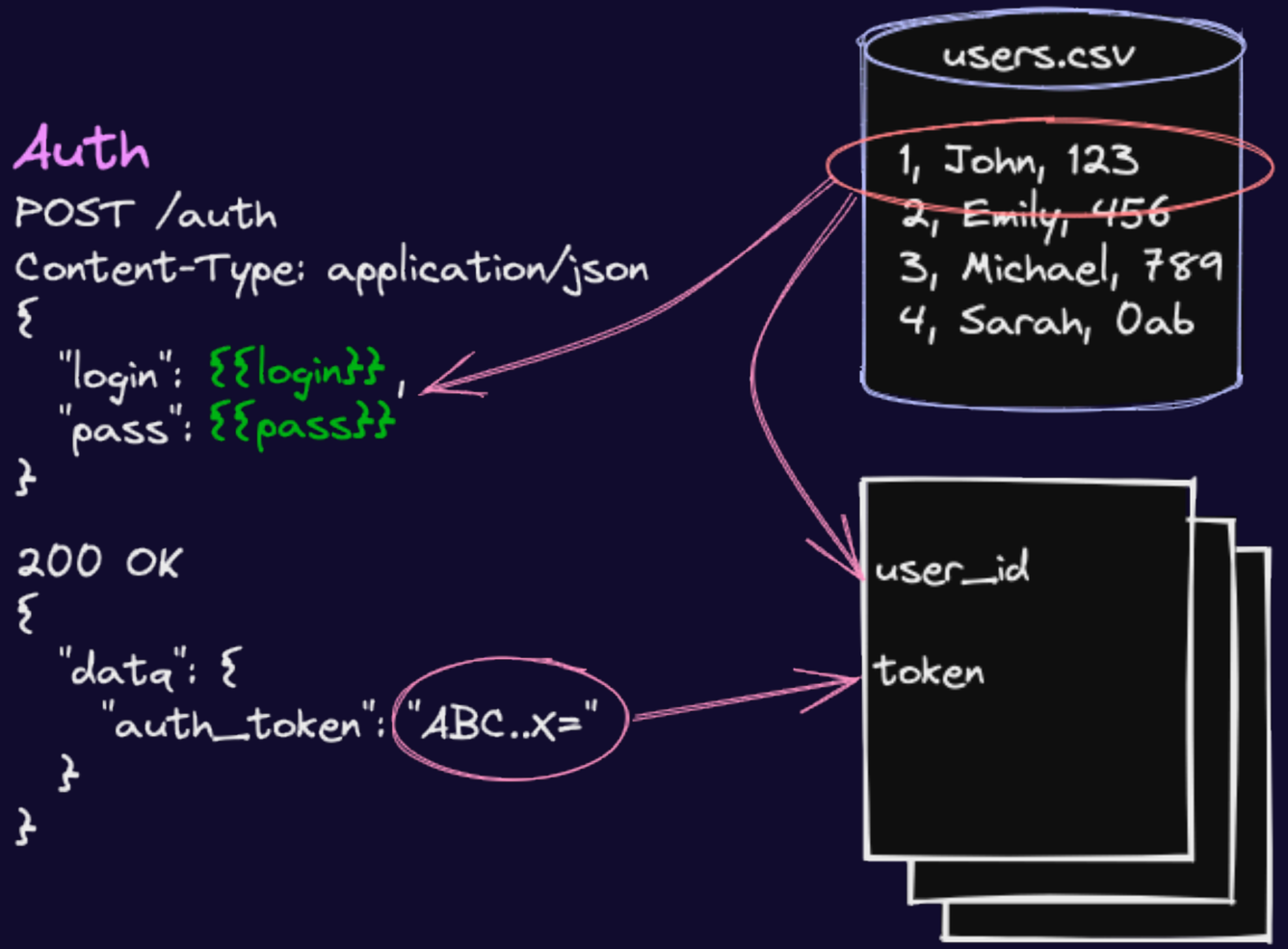

Запомни: широта использования инструмента
зависит от широты его возможностей

Пример сценария

1. Авторизоваться
2. Получить список товаров
3. Сделать заказ трёх товаров

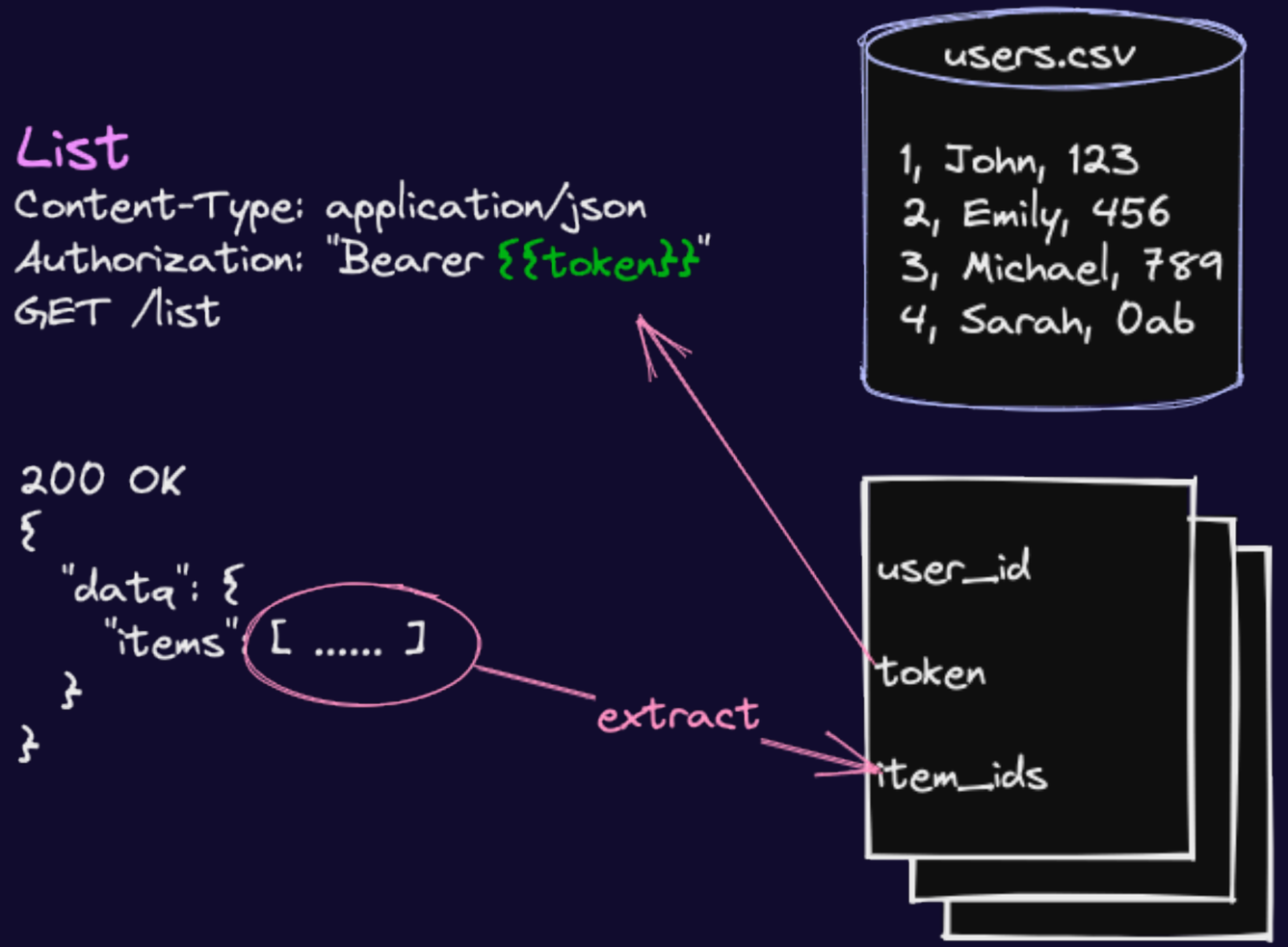
Пример сценария

1. Авторизоваться
2. Получить список товаров
3. Сделать заказ трёх товаров



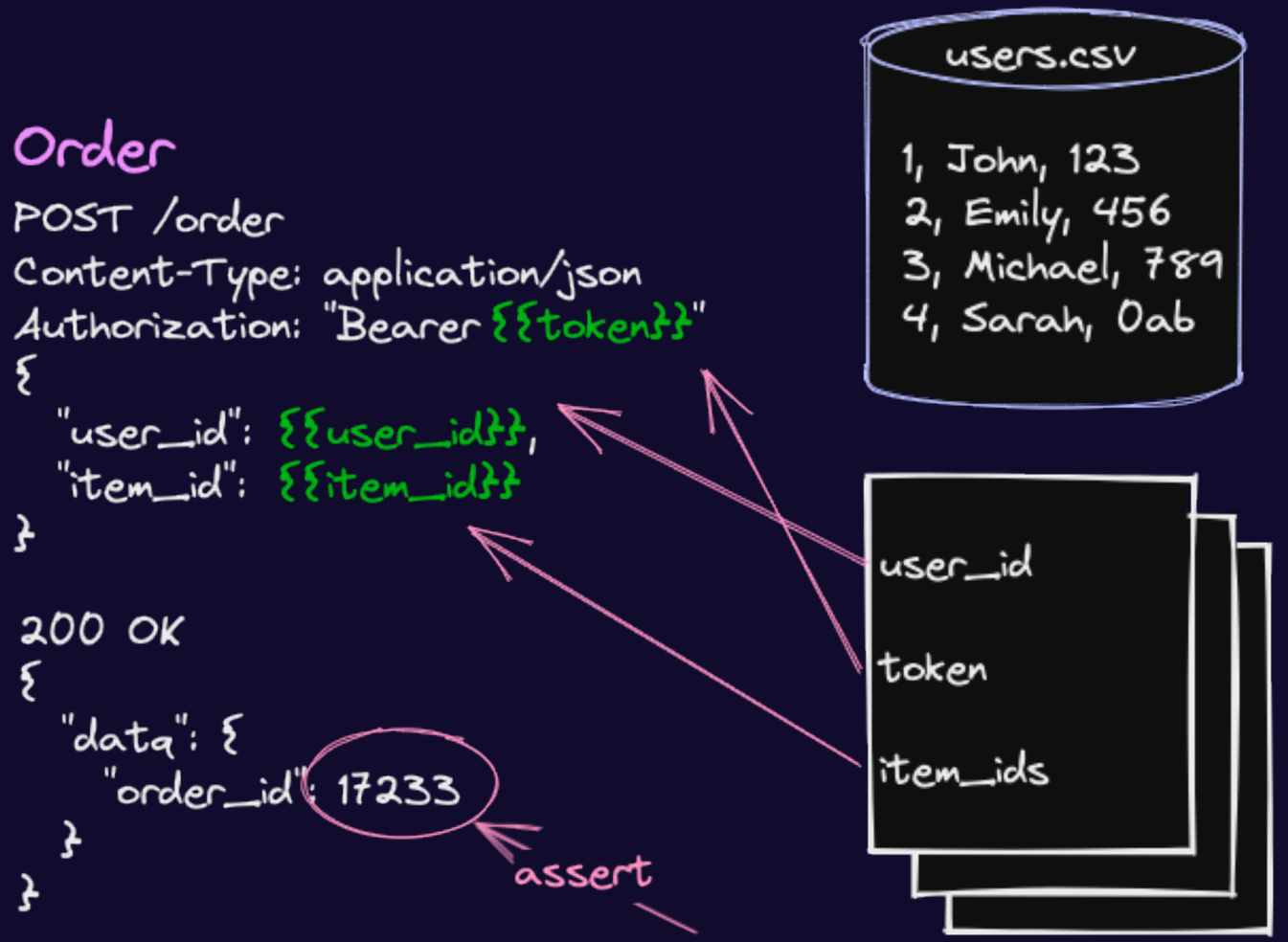
Пример сценария

1. Авторизоваться
2. Получить список товаров
3. Сделать заказ трёх товаров



Пример сценария

1. Авторизоваться
2. Получить список товаров
3. Сделать заказ трёх товаров



В чем отличие от подготовленных

- Переменные

- Где получить?
- Где использовать?

Работа с переменными

1. Как получать эти переменные?
   - Разные источники переменных
     - Файлы
     - База данных
     - Предыдущие запросы (response parsing)
2. Как использовать в запросы?
   - Плейсхолдеры
   - Шаблонизаторы

Как бы это выглядело


```
POST /auth
{
  "login": {{login}},
  "pass": {{pass}}
}
```

```
GET /list
Authorization: "Bearer {{token}}"
```

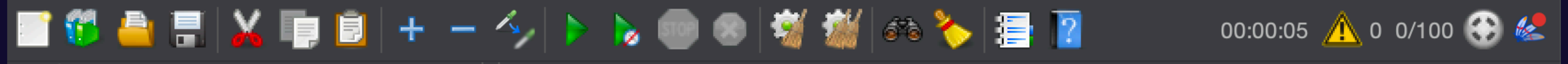
```
POST /order
Authorization: "Bearer {{token}}"
{
  "user_id": {{user_id}},
  "item_id": {{item_id}}
}
```

Код

Примеры для эксперимента можно посмотреть на GitHub.

- код тестового сервера
- конфиг для pandora
- конфиг для jmeter
- файлы с тестовыми данными
- Makefile со скриптами запуска

https://github.com/oke11o/pandora-presentaion



- Test Plan
  - Thread Group
    - CSV Data Set Config
    - User Defined Variables
    - HTTP Header Manager
    - scenario1.auth
      - JSON Extractor
      - Response Assertion
      - Constant Timer
    - scenario1.list
      - JSON Extractor
      - Constant Timer
    - scenario1.order1
      - Constant Timer
    - scenario1.order2
      - Constant Timer
    - scenario1.order3
      - Constant Timer
    - View Results Tree
    - Aggregate Report
  - Open Model Thread Group

Thread Group

Name: Thread Group

Comments:

Action to be taken after a Sampler error

(•) Continue ( ) Start Next Thread Loop ( ) Stop Thread ( ) Stop Test ( ) Stop Test Now

Thread Properties

Number of Threads (users): 100

Ramp-up period (seconds): 0

Loop Count: [x] Infinite

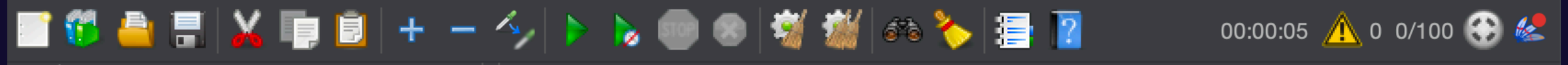
[x] Same user on each iteration

[ ] Delay Thread creation until needed

[x] Specify Thread lifetime

Duration (seconds): 5

Startup delay (seconds):



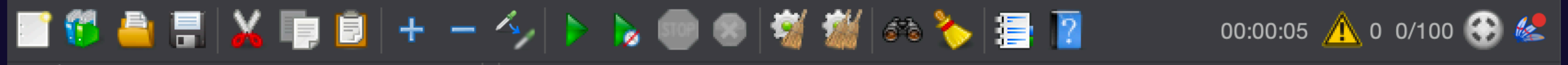
- Test Plan
  - Thread Group
    - CSV Data Set Config
    - User Defined Variables
    - HTTP Header Manager
    - scenario1.auth
      - JSON Extractor
      - Response Assertion
      - Constant Timer
    - scenario1.list
      - JSON Extractor
      - Constant Timer
    - scenario1.order1
      - Constant Timer
    - scenario1.order2
      - Constant Timer
    - scenario1.order3
      - Constant Timer
    - View Results Tree
    - Aggregate Report
  - Open Model Thread Group

CSV Data Set Config

Name:	CSV Data Set Config
Comments:	

Configure the CSV Data Source

Filename:	users.csv
File encoding:	
Variable Names (comma-delimited):	user_id
Ignore first line (only used if Variable Names is not empty):	False
Delimiter (use '\t' for tab):	,
Allow quoted data?:	False
Recycle on EOF ?:	True
Stop thread on EOF ?:	False
Sharing mode:	All threads



- Test Plan
  - Thread Group
    - CSV Data Set Config
    - User Defined Variables
    - HTTP Header Manager
    - scenario1.auth
      - JSON Extractor
      - Response Assertion
      - Constant Timer
    - scenario1.list
      - JSON Extractor
      - Constant Timer
    - scenario1.order1
      - Constant Timer
    - scenario1.order2
      - Constant Timer
    - scenario1.order3
      - Constant Timer
    - View Results Tree
    - Aggregate Report
  - Open Model Thread Group

User Defined Variables

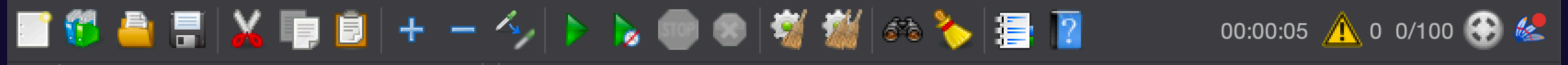
Name: User Defined Variables

Comments:

User Defined Variab

Name:	Value
auth_token	-
user_id	-
item_id	-
host	localhost
port	8092
sleep	100

Detail | Add | Add from Clipboard



- Test Plan
  - Thread Group
    - CSV Data Set Config
    - User Defined Variables
    - HTTP Header Manager
    - scenario1.auth
      - JSON Extractor
      - Response Assertion
      - Constant Timer
    - scenario1.list
      - JSON Extractor
      - Constant Timer
    - scenario1.order1
      - Constant Timer
    - scenario1.order2
      - Constant Timer
    - scenario1.order3
      - Constant Timer
    - View Results Tree
    - Aggregate Report
  - Open Model Thread Group

HTTP Header Manager

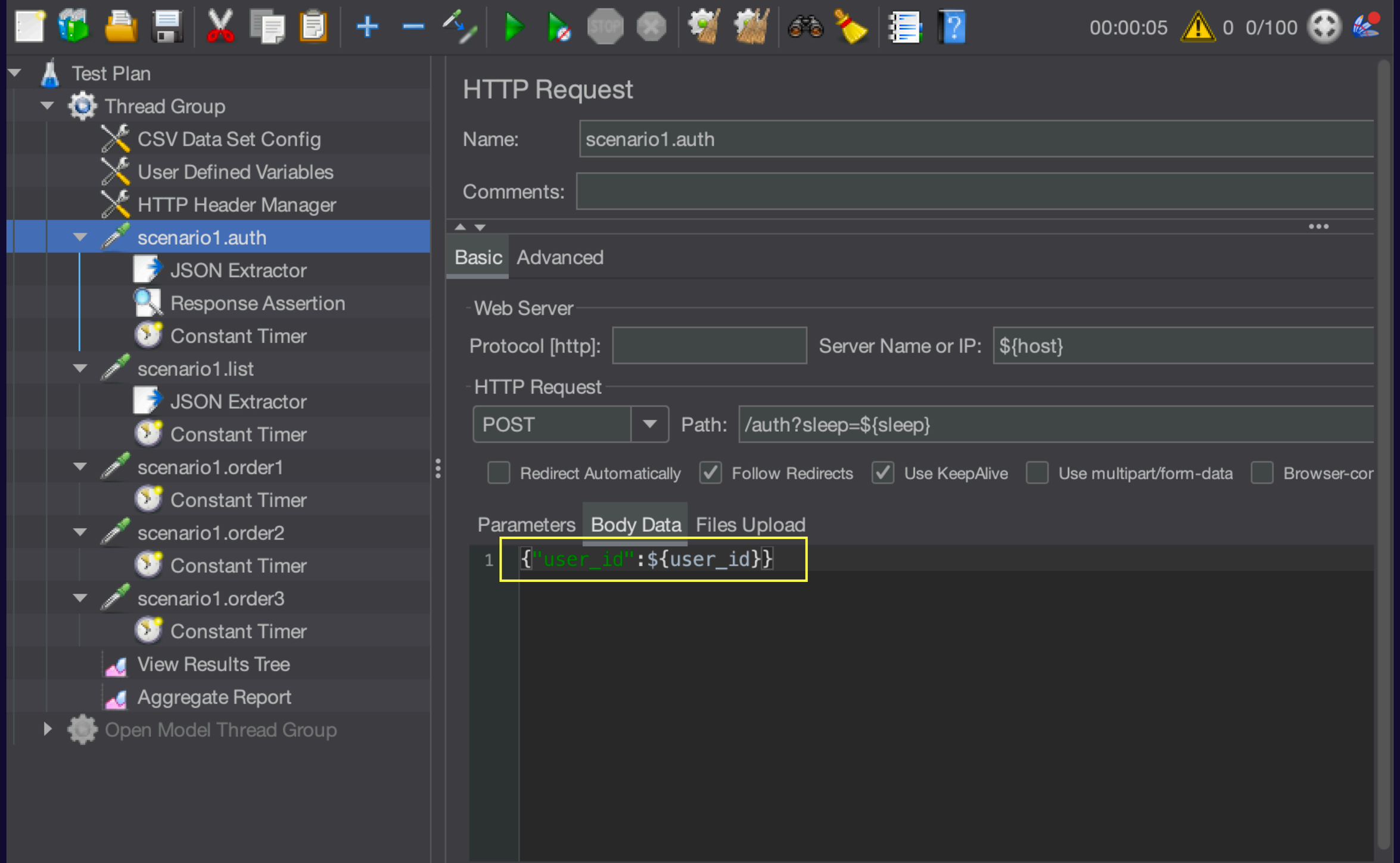
Name: HTTP Header Manager

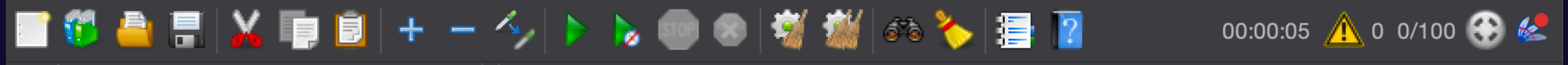
Comments:

Headers Stored in the Header Manager

Name:	
Content-Type	application/json
Authorization	Bearer ${auth_token}

Add | Add from Clipboard | Delet


00:00:05
0
0/100
Test Plan
Thread Group
CSV Data Set Config
User Defined Variables
HTTP Header Manager
scenario1.auth
JSON Extractor
Response Assertion
Constant Timer
scenario1.list
JSON Extractor
Constant Timer
scenario1.order1
Constant Timer
scenario1.order2
Constant Timer
scenario1.order3
Constant Timer
View Results Tree
Aggregate Report
Open Model Thread Group
HTTP Request
Name:
scenario1.auth
Comments:
Basic
Advanced
Web Server
Protocol [http]:
Server Name or IP:
${host}
HTTP Request
POST
Path:
/auth?sleep=${sleep}
Redirect Automatically
Follow Redirects
Use KeepAlive
Use multipart/form-data
Browser-cor
Parameters
Body Data
Files Upload
1
{"user_id":${user_id}}



- Test Plan
  - Thread Group
    - CSV Data Set Config
    - User Defined Variables
    - HTTP Header Manager
    - scenario1.auth
      - JSON Extractor
      - Response Assertion
      - Constant Timer
    - scenario1.list
      - JSON Extractor
      - Constant Timer
    - scenario1.order1
      - Constant Timer
    - scenario1.order2
      - Constant Timer
    - scenario1.order3
      - Constant Timer
    - View Results Tree
    - Aggregate Report
  - Open Model Thread Group

Response Assertion

Name: Response Assertion

Comments:

Apply to:

(○) Main sample and sub-samples (◉) Main sample only (○) Sub-samples only (○) JMeter Variable

Field to Test

(◉) Text Response (○) Response Code (○) Resp

(○) Request Headers (○) URL Sampled (○) Doc

(○) Request Data

Pattern Matching Rules

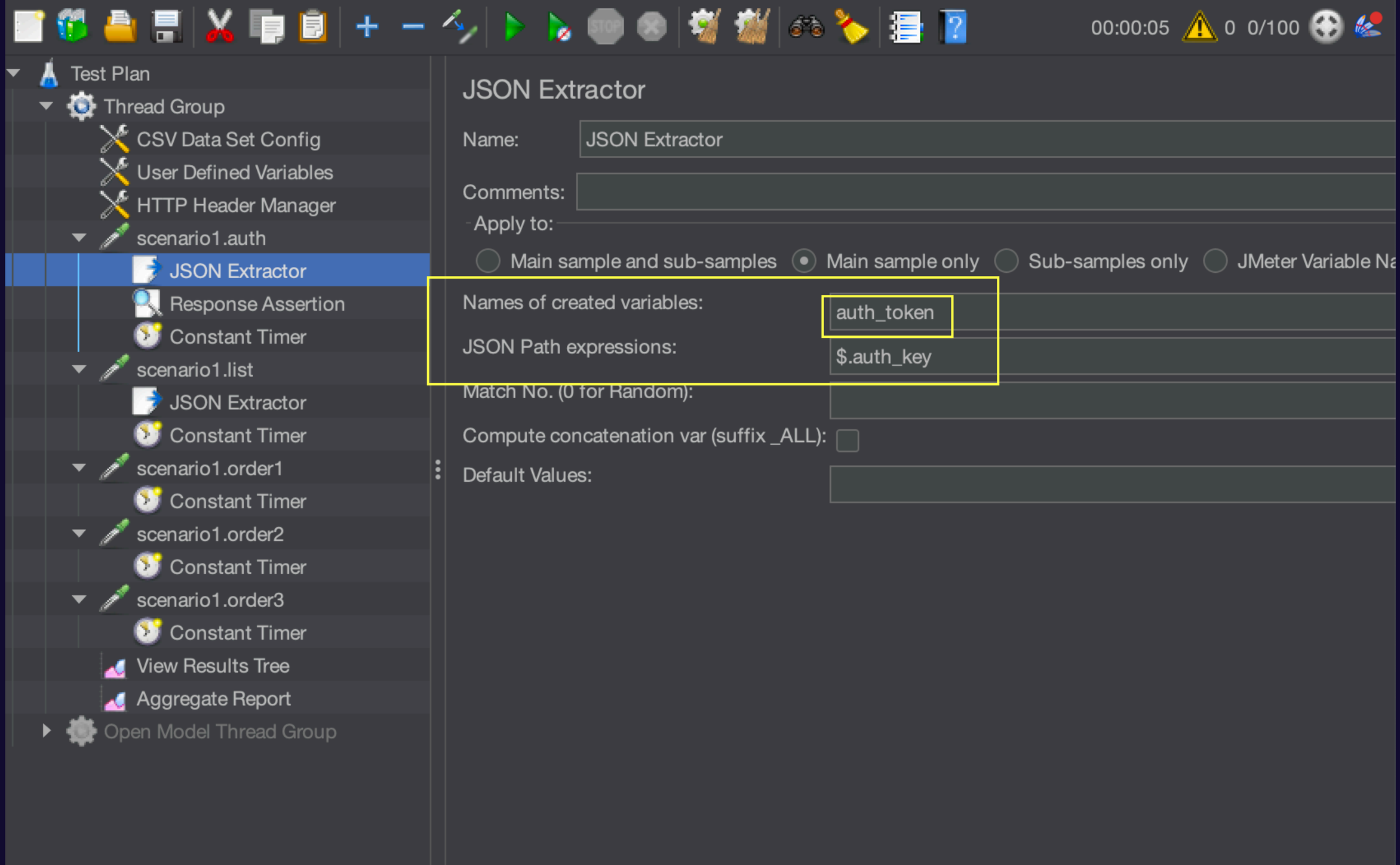
(○) Contains (○) Matches (○) Equals

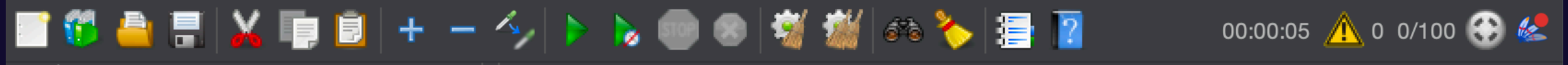
Patterns to Test

	Patterns to Tes
1	auth_key

Add | Add from Clipboa

Custom failure message


00:00:05
0
0/100
Test Plan
Thread Group
CSV Data Set Config
User Defined Variables
HTTP Header Manager
scenario1.auth
JSON Extractor
Response Assertion
Constant Timer
scenario1.list
JSON Extractor
Constant Timer
scenario1.order1
Constant Timer
scenario1.order2
Constant Timer
scenario1.order3
Constant Timer
View Results Tree
Aggregate Report
Open Model Thread Group
JSON Extractor
Name:
JSON Extractor
Comments:
Apply to:
Main sample and sub-samples
Main sample only
Sub-samples only
JMeter Variable Na
Names of created variables:
auth_token
JSON Path expressions:
$.auth_key
Match No. (0 for Random):
Compute concatenation var (suffix _ALL):
Default Values:



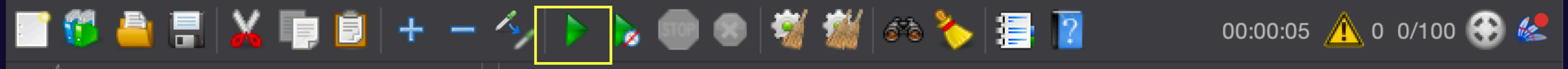
- Test Plan
  - Thread Group
    - CSV Data Set Config
    - User Defined Variables
    - HTTP Header Manager
    - scenario1.auth
      - JSON Extractor
      - Response Assertion
      - Constant Timer
    - scenario1.list
      - JSON Extractor
      - Constant Timer
    - scenario1.order1
      - Constant Timer
    - scenario1.order2
      - Constant Timer
    - scenario1.order3
      - Constant Timer
    - View Results Tree
    - Aggregate Report
  - Open Model Thread Group

Constant Timer

Name: Constant Timer

Comments:

Thread Delay (in milliseconds): 100



- Test Plan
  - Thread Group
    - CSV Data Set Config
    - User Defined Variables
    - HTTP Header Manager
    - scenario1.auth
      - JSON Extractor
      - Response Assertion
      - Constant Timer
    - scenario1.list
      - JSON Extractor
      - Constant Timer
    - scenario1.order1
      - Constant Timer
    - scenario1.order2
      - Constant Timer
    - scenario1.order3
      - Constant Timer
    - View Results Tree
    - Aggregate Report
  - Open Model Thread Group

Aggregate Report

Name: Aggregate Report

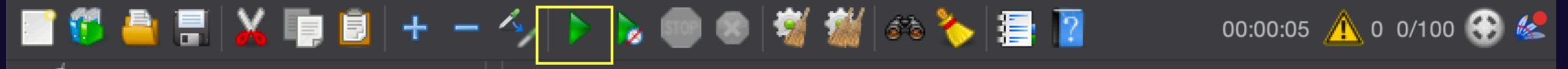
Comments:

Write results to file / Read from file

Filename

Label	# Samples	Average	Median	90% Line	95% Line	99% Lin
scenario1.auth	500	102	102	104	104	1
scenario1.list	500	102	102	103	104	1
scenario1.order1	500	101	102	103	103	1
scenario1.order2	500	102	102	103	104	1
scenario1.order3	400	102	102	104	104	1
TOTAL	2400	102	102	103	104	1

Include group name in label? Save Table

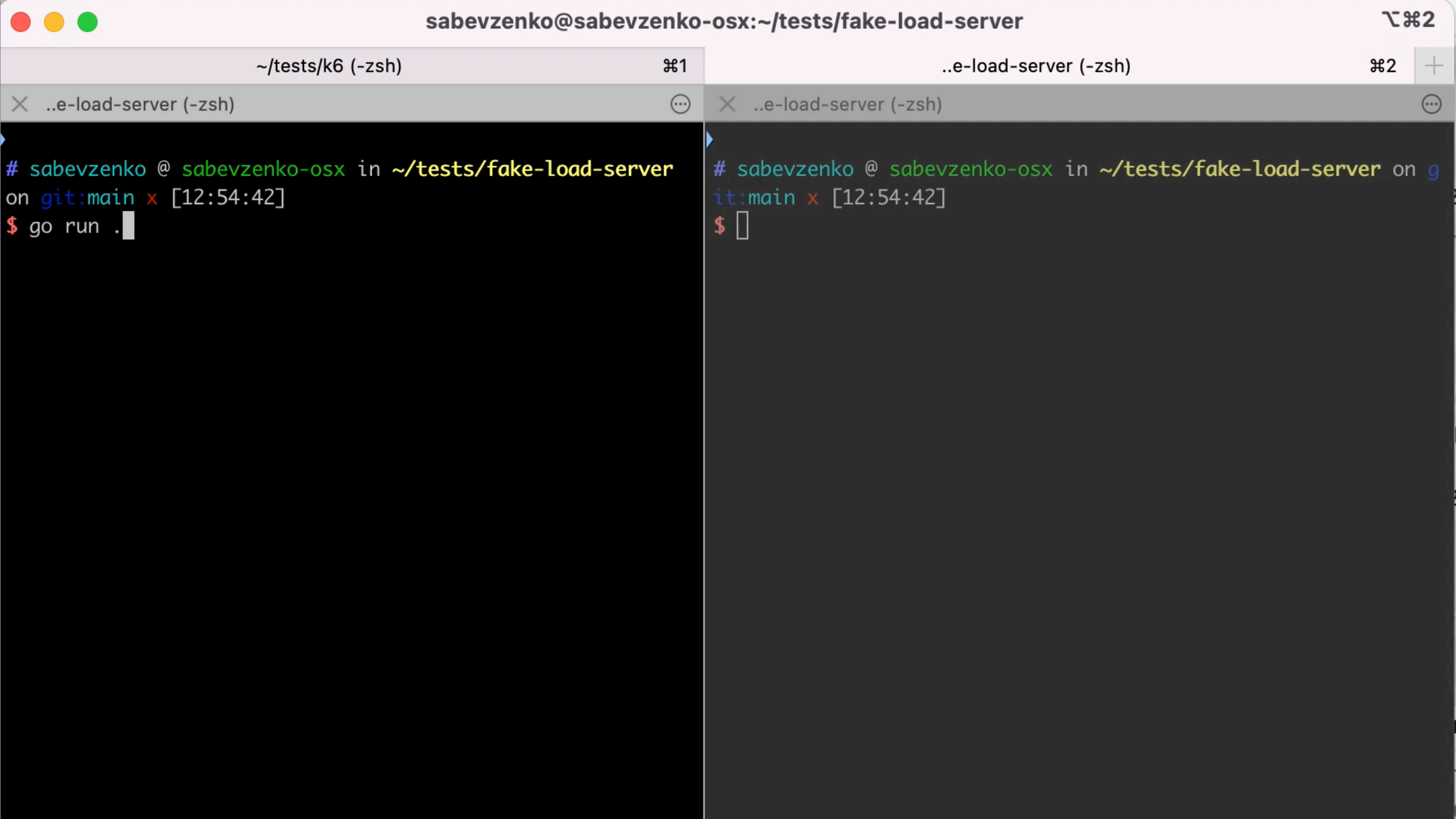


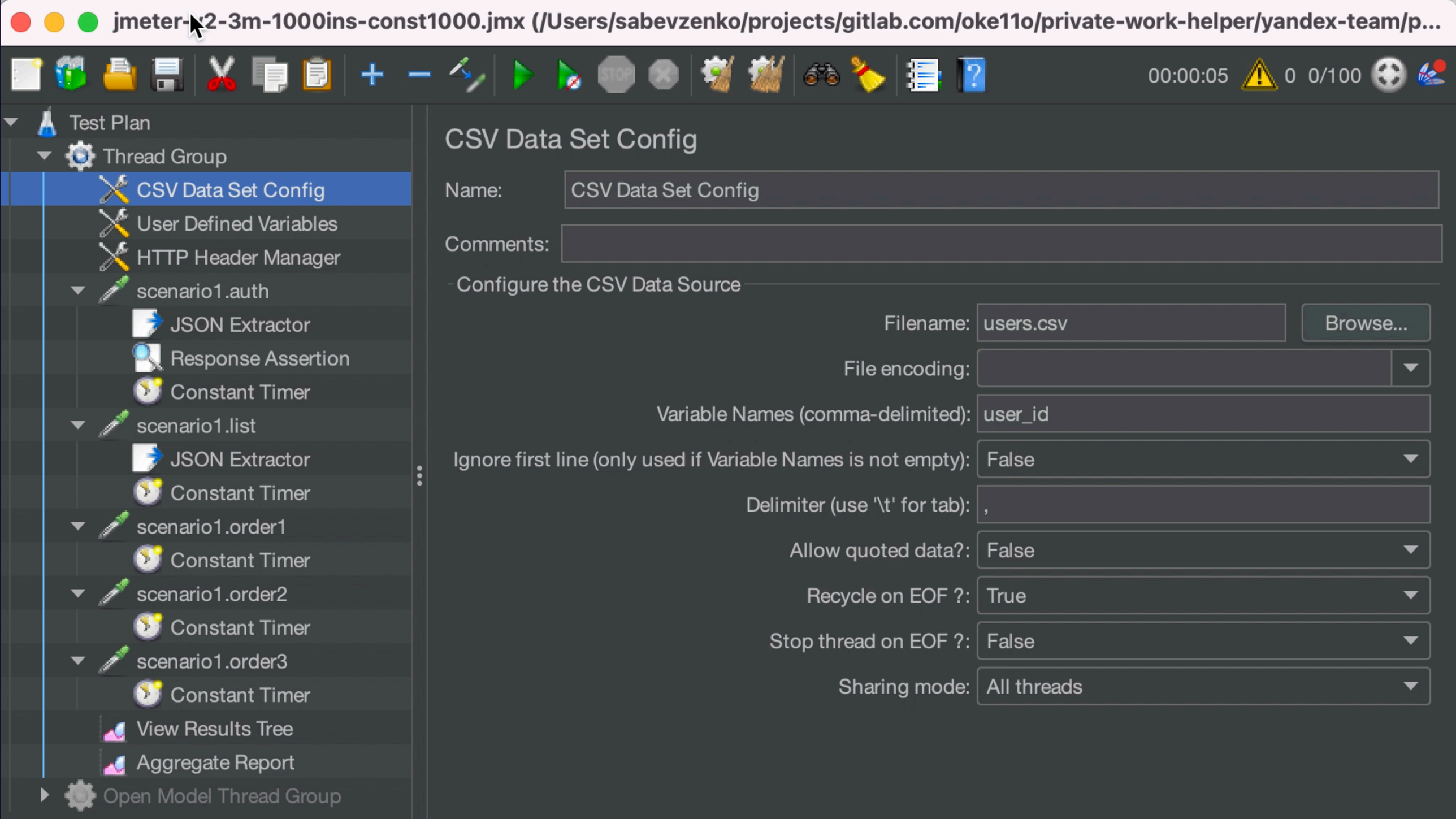
- Test Plan
  - Thread Group
    - CSV Data Set Config
    - User Defined Variables
    - HTTP Header Manager
    - scenario1.auth
      - JSON Extractor
      - Response Assertion
      - Constant Timer
    - scenario1.list
      - JSON Extractor
      - Constant Timer
    - scenario1.order1
      - Constant Timer
    - scenario1.order2
      - Constant Timer
    - scenario1.order3
      - Constant Timer
    - View Results Tree
    - Aggregate Report
  - Open Model Thread Group

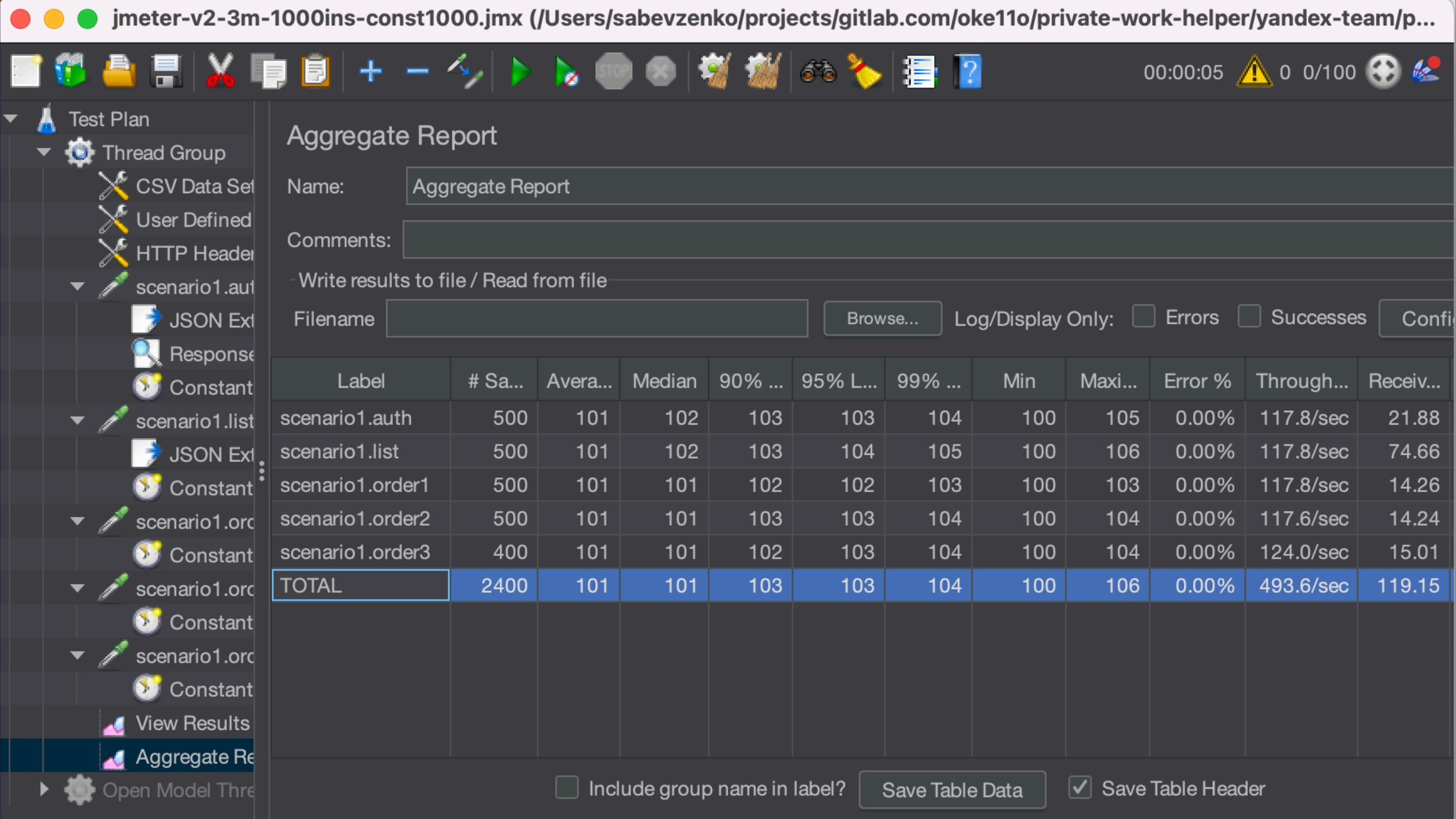
Browse... Log/Display Only: Errors Successes Configure

Line	99% Line	Min	Maximum	Error %	Throughput	Received K...	Sent KB/sec
104	105	100	105	0.00%	118.2/sec	21.94	31.53
104	105	100	105	0.00%	118.3/sec	75.00	29.13
103	103	100	103	0.00%	118.3/sec	14.32	34.21
104	105	100	105	0.00%	118.2/sec	14.32	34.43
104	104	101	104	0.00%	125.0/sec	15.14	36.40
104	105	100	105	0.00%	495.2/sec	119.52	136.83

Save Table Data Save Table Header


sabevzenko@sabevzenko-osx:~/tests/fake-load-server
⌥⌘2
~/tests/k6 (-zsh)
⌘1
..e-load-server (-zsh)
⌘2
..e-load-server (-zsh)
..e-load-server (-zsh)
sabevzenko @ sabevzenko-osx in ~/tests/fake-load-server
on git:main x [12:54:42]
$ go run .
# sabevzenko @ sabevzenko-osx in ~/tests/fake-load-server on g
it:main x [12:54:42]
$


jmeter-x2-3m-1000ins-const1000.jmx (/Users/sabevzenko/projects/gitlab.com/oke11o/private-work-helper/yandex-team/p...
00:00:05
0
0/100
Test Plan
Thread Group
CSV Data Set Config
User Defined Variables
HTTP Header Manager
scenario1.auth
JSON Extractor
Response Assertion
Constant Timer
scenario1.list
JSON Extractor
Constant Timer
scenario1.order1
Constant Timer
scenario1.order2
Constant Timer
scenario1.order3
Constant Timer
View Results Tree
Aggregate Report
Open Model Thread Group
CSV Data Set Config
Name:
CSV Data Set Config
Comments:
Configure the CSV Data Source
Filename:
users.csv
Browse...
File encoding:
Variable Names (comma-delimited):
user_id
Ignore first line (only used if Variable Names is not empty):
False
Delimiter (use '\t' for tab):
,
Allow quoted data?:
False
Recycle on EOF ?:
True
Stop thread on EOF ?:
False
Sharing mode:
All threads


jmeter-v2-3m-1000ins-const1000.jmx (/Users/sabevzenko/projects/gitlab.com/oke11o/private-work-helper/yandex-team/p...
00:00:05
0
0/100
Test Plan
Thread Group
CSV Data Set
User Defined
HTTP Header
scenario1.aut
JSON Ext
Response
Constant
scenario1.list
JSON Ext
Constant
scenario1.orc
Constant
scenario1.orc
Constant
scenario1.orc
Constant
View Results
Aggregate Re
Open Model Thre
Aggregate Report
Name:
Aggregate Report
Comments:
Write results to file / Read from file
Filename
Browse...
Log/Display Only:
Errors
Successes
Confi
Label
Sa...
Avera...
Median
90% ...
95% L...
99% ...
Min
Maxi...
Error %
Through...
Receiv...
scenario1.auth 500 101 102 103 103 104 100 105 0.00% 117.8/sec 21.88
scenario1.list 500 101 102 103 104 105 100 106 0.00% 117.8/sec 74.66
scenario1.order1 500 101 101 102 102 103 100 103 0.00% 117.8/sec 14.26
scenario1.order2 500 101 101 103 103 104 100 104 0.00% 117.6/sec 14.24
scenario1.order3 400 101 101 102 103 104 100 104 0.00% 124.0/sec 15.01
TOTAL 2400 101 101 103 103 104 100 106 0.00% 493.6/sec 119.15
Include group name in label?
Save Table Data
Save Table Header

Как бы это выглядело


```
POST /auth
{
  "login": {{login}},
  "pass": {{pass}}
}

GET /list
Authorization: "Bearer {{token}}"

POST /order
Authorization: "Bearer {{token}}"
{
  "user_id": {{user_id}},
  "item_id": {{item_id}}
}
```

k6 / Javascript

```
import http from 'k6/http';
import {check, sleep} from 'k6';
import {SharedArray} from 'k6/data';

export let options = {
    scenarios: {
        scenario_name: {
            executor: 'constant-vus',
            vus: 100,
            duration: '5s',
        },
    },
};

const users = new SharedArray('users', function () {
    return JSON.parse(open('./users.json'));
});


export default function () { … }
```

k6 / Javascript

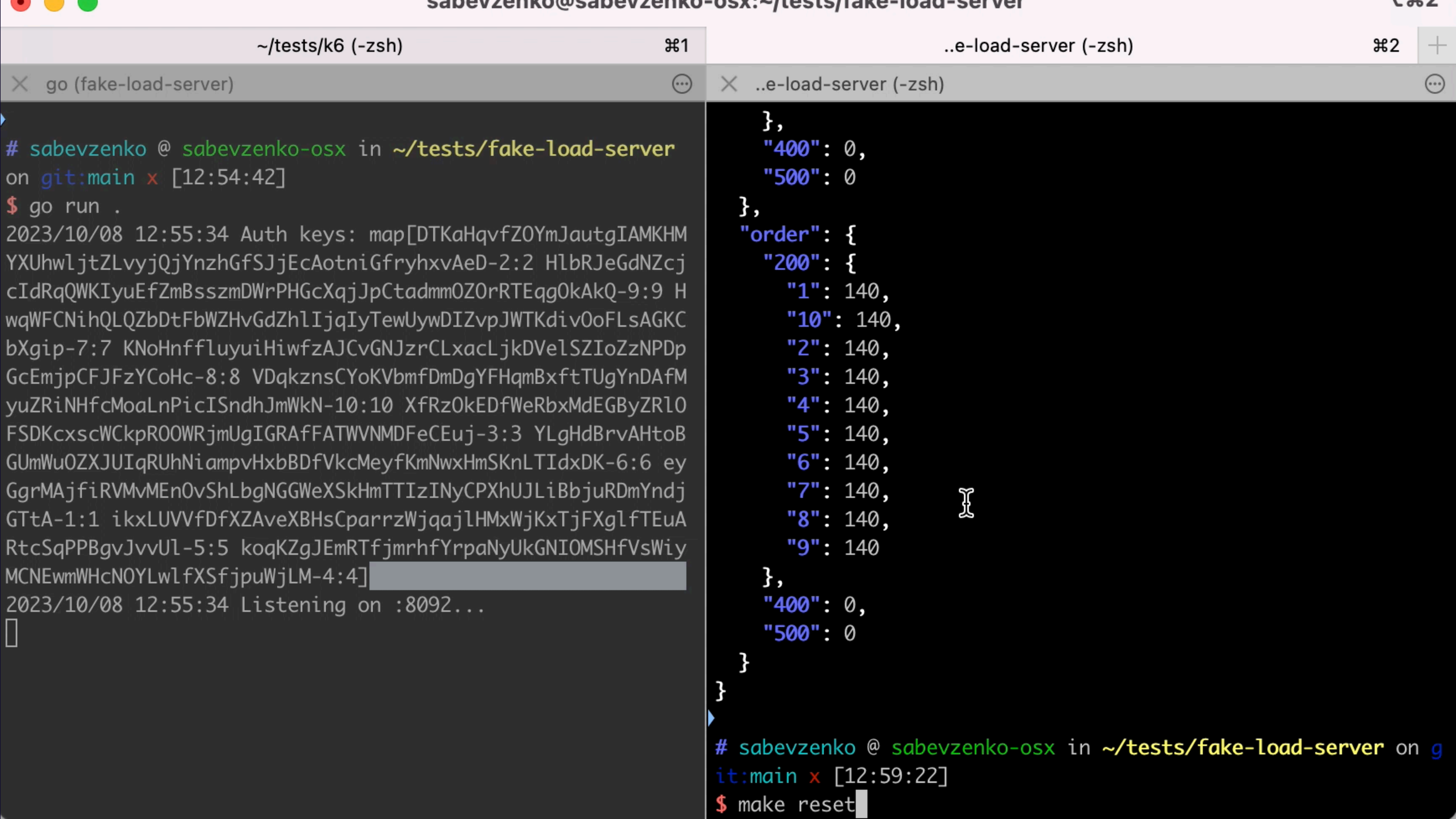
```
// Step 1: POST /auth
const user = users[Math.floor(Math.random() * users.length)];
let authPayload = JSON.stringify({"user_id": user.user_id});
let authHeaders = {'Content-Type': 'application/json'};
let authResponse = http.post(
    'http://localhost:8092/auth?sleep=100',
    authPayload, {headers: authHeaders}
);
check(authResponse,
    {'Auth Request Successful': (r) => r.status === 200}
);
let authToken = JSON.parse(authResponse.body).auth_key;

sleep(0.1)
```

k6 / Javascript

```
// Step 2: GET /list
let listHeaders = {
    'Content-Type': 'application/json',
    'Authorization': `Bearer ${authToken}`,
};
let listResponse = http.get('http://localhost:8092/list?sleep=100',
    {headers: listHeaders}
);
check(listResponse,
    {'List Request Successful': (r) => r.status === 200}
);

let itemIds = JSON.parse(listResponse.body);
itemIds = itemIds.items

sleep(0.1)
```

k6 / Javascript

```
// Step 3: POST /order
for (let n = 0; n < 3; n++) {
    let randomItemId = itemIds[Math.floor(Math.random() * itemIds.length)];
    let orderPayload = JSON.stringify(
        {"item_id": randomItemId}
    );
    let orderHeaders = {
        'Content-Type': 'application/json',
        'Authorization': `Bearer ${authToken}`,
    };
    let orderResponse = http.post(
        'http://localhost:8092/order?sleep=100',
        orderPayload,
        {headers: orderHeaders}
    );
    check(orderResponse,
        {'Order Request Successful': (r) => r.status === 200}
    );
    sleep(0.1)
}
```


sabevzenko@sabevzenko-osx:~/tests/fake-load-server
~/tests/k6 (-zsh) ⌘1
..e-load-server (-zsh) ⌘2
go (fake-load-server)
..e-load-server (-zsh)
sabevzenko @ sabevzenko-osx in ~/tests/fake-load-server
on git:main x [12:54:42]
$ go run .
2023/10/08 12:55:34 Auth keys: map[DTKaHqvfZOYmJautgIAMKHM
YXUhwljtZLvyjQjYnzhGfSJjEcAotniGfryhxvAeD-2:2 HlbRJeGdNZcj
cIdRqQWKIyuEfZmBsszmDWrPHGcXqjJpCtadmmOZOrRTEqgOkAkQ-9:9 H
wqWFCNihQLQZbDtFbWZHvGdZhlIjqIyTewUywDIZvpJWTKdivOoFLsAGKC
bXgip-7:7 KNoHnffluyuiHiwfzAJCvGNJzrCLxacLjkDVelSZIoZzNPDp
GcEmjpCFJFzYCoHc-8:8 VDqkznsCYoKVbmfDmDgYFHqmBxftTUgYnDAfM
yuZRiNHfcMoaLnPicISndhJmWkN-10:10 XfRzOkEDfWeRbxMdEGByZRlO
FSDKcxscWCkpROOWRjmUgIGRAfFATWVNMDFeCEuj-3:3 YLgHdBrvAHtoB
GUmWuOZXJUIqRUhNiampvHxbBDfVkcMeyfKmNwxHmSKnLTIdxDK-6:6 ey
GgrMAjfiRVMvMEnOvShLbgNGGWeXSkHmTTIzINyCPXhUJLiBbjuRDmYndj
GTtA-1:1 ikxLUVVfDfXZAveXBHsCparrzWjqajlHMxWjKxTjFXglfTEuA
RtcSqPPBgvJvvUl-5:5 koqKZgJEmRTfjmrhfYrpaNyUkGNIOMSHfVsWiy
MCNEwmWHcNOYLwlfXSfjpuWjLM-4:4]
2023/10/08 12:55:34 Listening on :8092...
},
"400": 0,
"500": 0
},
"order": {
"200": {
"1": 140,
"10": 140,
"2": 140,
"3": 140,
"4": 140,
"5": 140,
"6": 140,
"7": 140,
"8": 140,
"9": 140
},
"400": 0,
"500": 0
}
}
# sabevzenko @ sabevzenko-osx in ~/tests/fake-load-server on g
it:main x [12:59:22]
$ make reset

```
    ✓ Order Request Successful

    checks.....................: 100.00% ✓ 2500       x 0
    data_received..............: 606 kB  120 kB/s
    data_sent..................: 569 kB  113 kB/s
    http_req_blocked...........: avg=85.55µs  min=1µs      med=3µs      max=6.15ms   p(90)=7µs      p(95)=29µs
    http_req_connecting........: avg=75.4µs   min=0s       med=0s       max=3.98ms   p(90)=0s       p(95)=0s
    http_req_duration..........: avg=101.62ms min=100.14ms med=101.41ms max=105.91ms p(90)=103.01ms p(95)=103.48ms
      { expected_response:true }...: avg=101.62ms min=100.14ms med=101.41ms max=105.91ms p(90)=103.01ms p(95)=103.48ms
    http_req_failed............: 0.00%   ✓ 0          x 2500
    http_req_receiving.........: avg=35.11µs  min=14µs     med=28µs     max=875µs    p(90)=55µs     p(95)=74µs
    http_req_sending...........: avg=101.32µs min=6µs      med=16µs     max=3.67ms   p(90)=267.09µs p(95)=525µs
    http_req_tls_handshaking...: avg=0s       min=0s       med=0s       max=0s       p(90)=0s       p(95)=0s
    http_req_waiting...........: avg=101.48ms min=100.11ms med=101.28ms max=105.23ms p(90)=102.84ms p(95)=103.3ms
    http_reqs..................: 2500    493.793217/s
    iteration_duration.........: avg=1.01s    min=1s       med=1.01s    max=1.01s    p(90)=1.01s    p(95)=1.01s
    iterations.................: 500     98.758643/s
    vus........................: 100     min=100      max=100
    vus_max....................: 100     min=100      max=100


running (05.1s), 000/100 VUs, 500 complete and 0 interrupted iterations
scenario_name ✓ [======================================] 100 VUs  5s

# sabevzenko @ sabevzenko-osx in ~/tests/k6 on git:main x [13:02:01]
$
```

Как бы это выглядело

Pandora

custom generator

```
POST /auth
{
  "login": {{login}},
  "pass": {{pass}}
}
```

```
GET /list
Authorization: "Bearer {{token}}"
```

```
POST /order
Authorization: "Bearer {{token}}"
{
  "user_id": {{user_id}},
  "item_id": {{item_id}}
}
```

Custom Pandora Generator

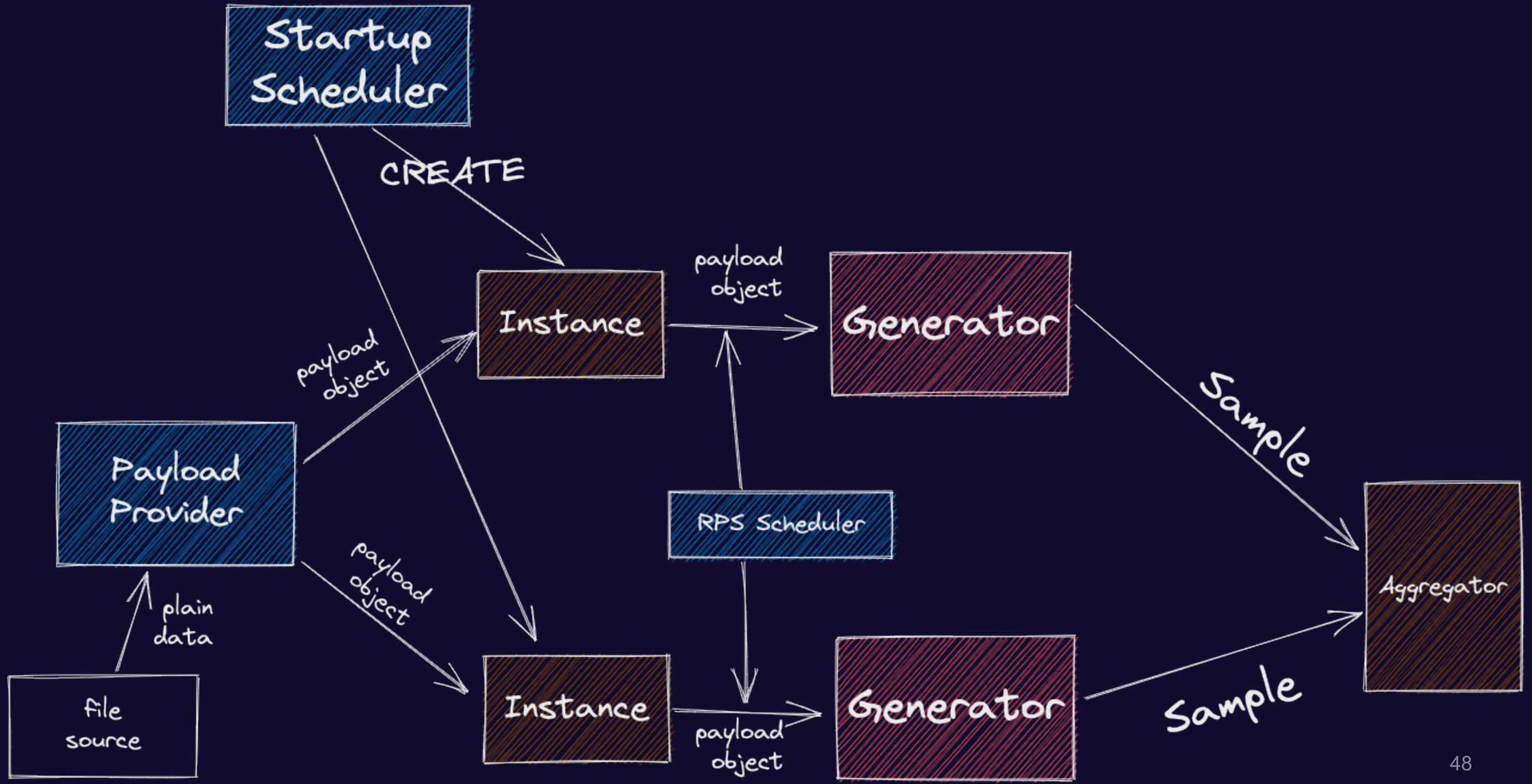
Components

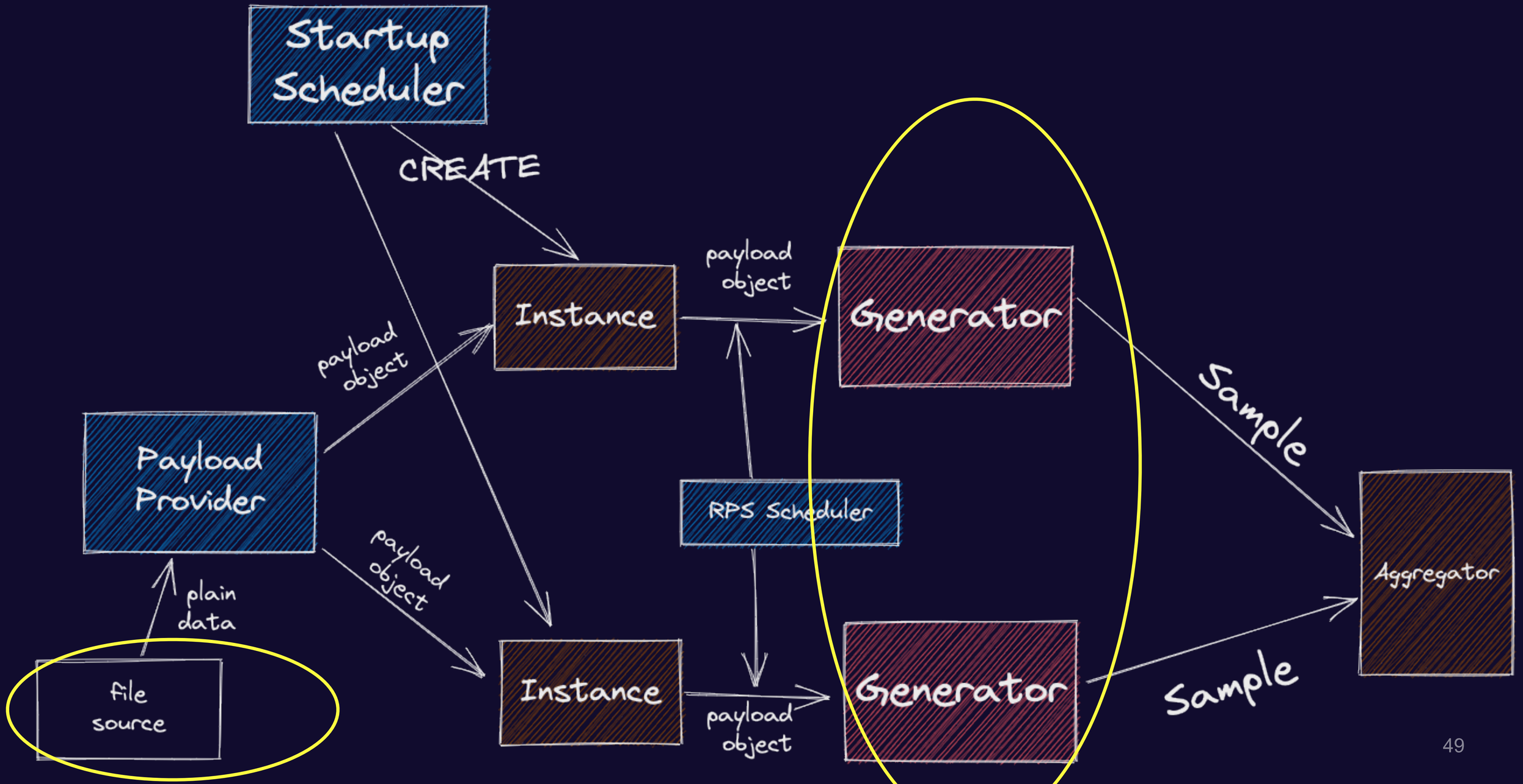
- Payload
- Provider
- Generator

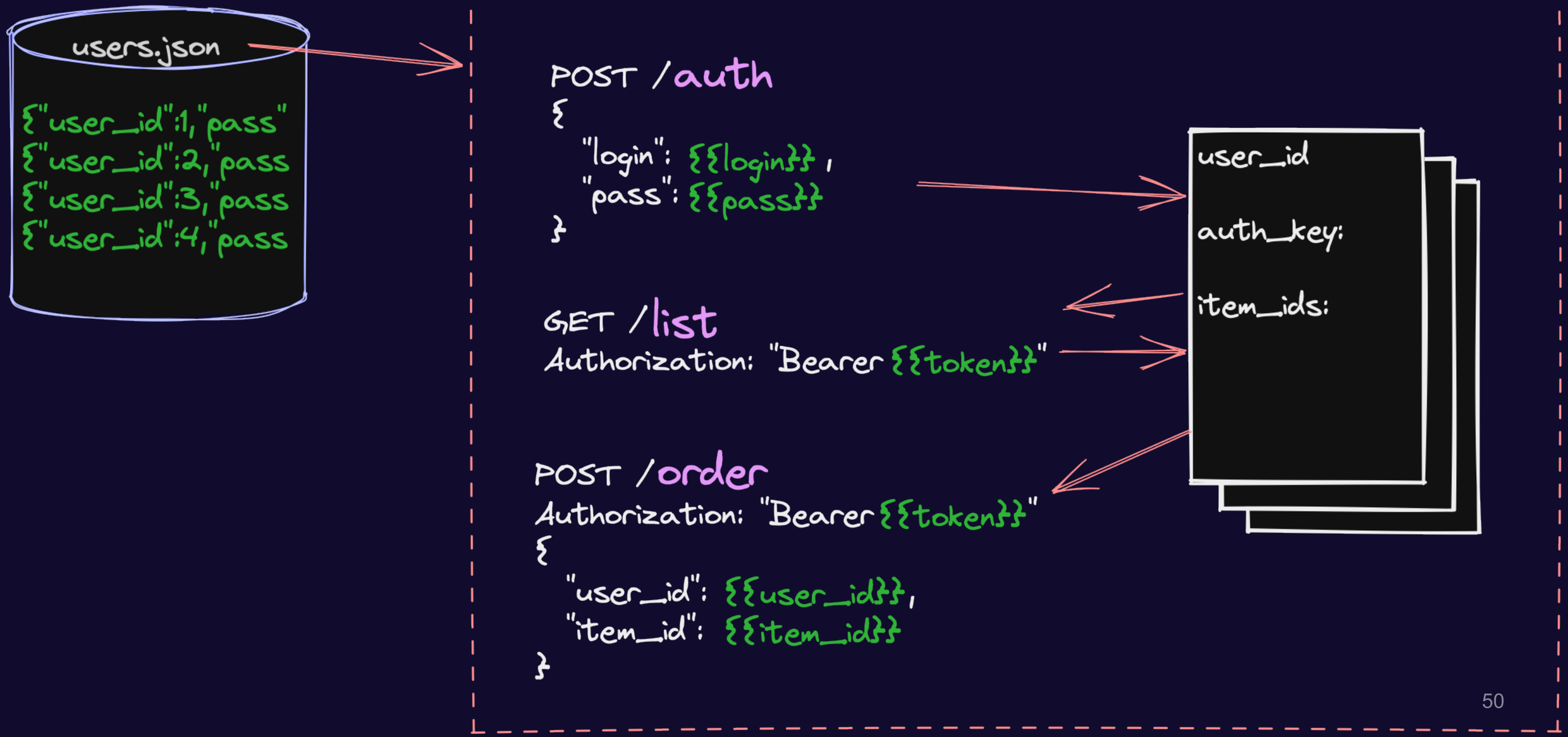
Custom Pandora Generator

Tips & Errors

- sample aggregator
- errors
- body.Close()
- nil pointer exception
- debug


Startup Scheduler
CREATE
Instance
payload object
Generator
Sample
payload object
Payload Provider
RPS Scheduler
Aggregator
payload object
plain data
file source
Instance
payload object
Generator
Sample


Startup Scheduler
CREATE
Instance
payload object
Generator
payload object
Payload Provider
RPS Scheduler
Sample
Aggregator
payload object
plain data
file source
Instance
payload object
Generator
Sample

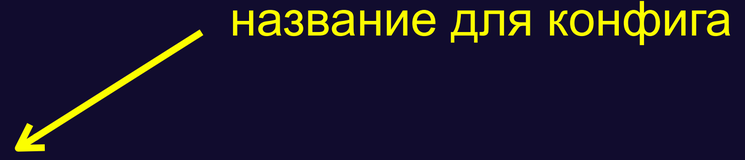

users.json
{"user_id":1,"pass"
{"user_id":2,"pass
{"user_id":3,"pass
{"user_id":4,"pass
POST /auth
{
"login": {{login}},
"pass": {{pass}}
}
GET /list
Authorization: "Bearer {{token}}"
POST /order
Authorization: "Bearer {{token}}"
{
"user_id": {{user_id}},
"item_id": {{item_id}}
}
user_id
auth_key:
item_ids:

Скопируем main.go

```go
package main

import (
  "github.com/spf13/afero"
  "github.com/yandex/pandora/cli"
  grpc "github.com/yandex/pandora/components/grpc/import"
  http "github.com/yandex/pandora/components/phttp/import"
  coreimport "github.com/yandex/pandora/core/import"
)

func main() {
  fs := afero.NewOsFs()
  coreimport.Import(fs)
  http.Import(fs)
  grpc.Import(fs)

  Import(fs)

  cli.Run()
}
```

Опишем формат Payload

```
type Payload struct {
  UserID   int64  `json:"user_id"`
  Name     string `json:"name"`
  Password string `json:"password"`
}

func Import(fs afero.Fs) {
  coreimport.RegisterCustomJSONProvider(“custom_provider",
    func() core.Ammo {
      return &Payload{}
    },
  )
  // TODO: register generator
}
```



Создадим Generator

```go
type Generator struct {
  conf   GeneratorConfig
  aggr   netsample.Aggregator
  deps   core.GunDeps
  client http.Client
}

func (g *Generator) Bind(aggr core.Aggregator, deps core.GunDeps) error {
  //TODO: implement me
  return nil
}

func (g *Generator) Shoot(payload core.Ammo) {
  //TODO: implement me
}
```

Укажем для него конфиги

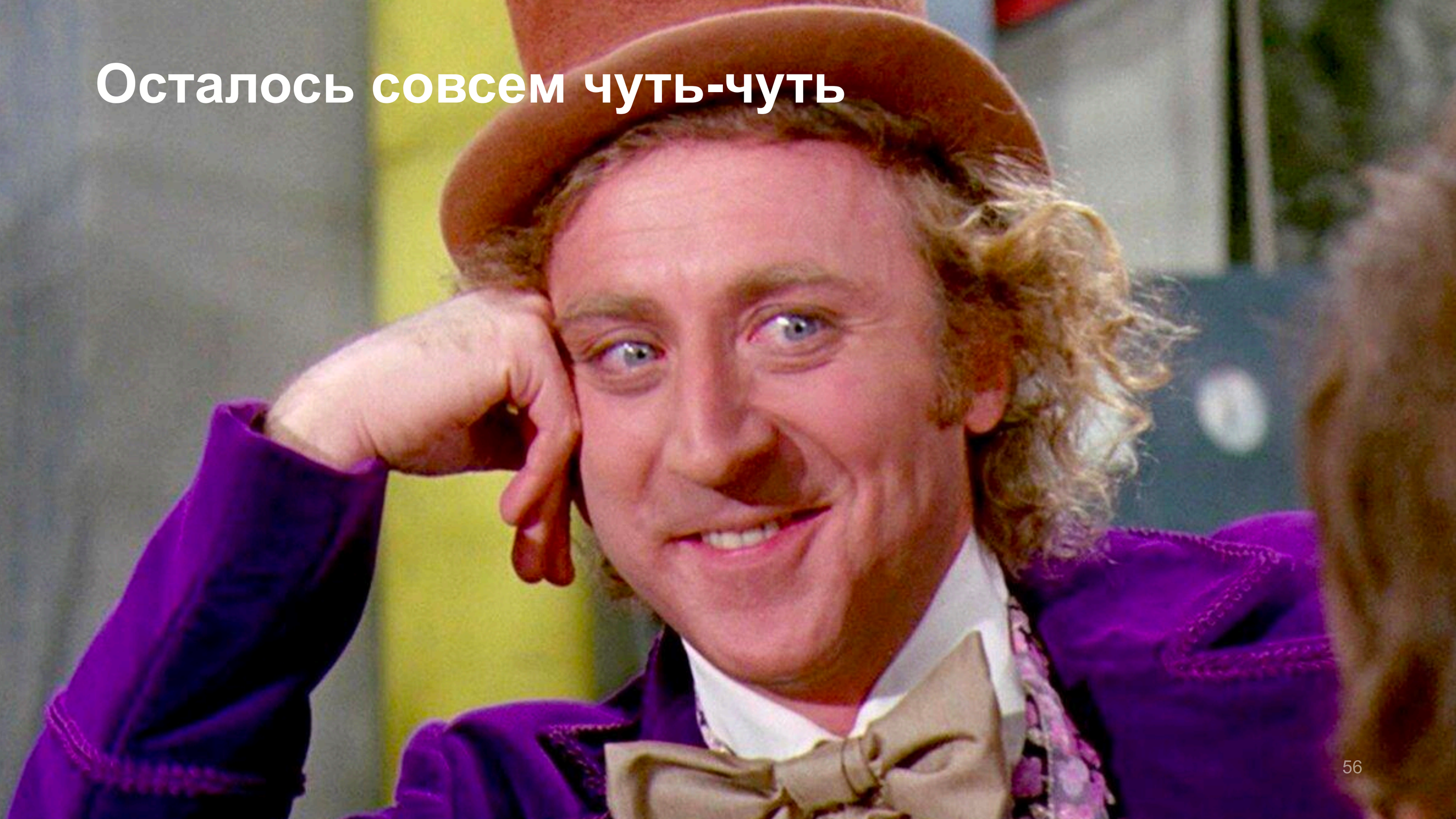
```
type GeneratorConfig struct {
  Target    string `validate:"required"`
  Transport TransportConfig
  Sleep     time.Duration
  ReqSleep  time.Duration
}
type TransportConfig struct {
  IdleConnTimeout time.Duration
}

func defaultConfig() GeneratorConfig {
  return GeneratorConfig{
    Transport: TransportConfig{
      IdleConnTimeout: 1,
    },
    ReqSleep: 100 * time.Millisecond,
  }
}
```

Зарегистрируем Generator

```
func Import(fs afero.Fs) {
  coreimport.RegisterCustomJSONProvider("custom_provider",
    func() core.Ammo {
      return &Payload{}
    },
  )

  register.Gun("custom_generator", func(conf GeneratorConfig) core.Gun {
    return &Generator{
      conf: conf,
    }
  }, defaultConfig)
}
```

название для конфига

Осталось совсем чуть-чуть



Сделаем последние приготовления

```
func (g *Generator) Bind(aggr core.Aggregator, deps core.GunDeps) error {
  g.aggr = netsample.UnwrapAggregator(aggr)
  g.deps = deps
  tr := &http.Transport{
    IdleConnTimeout: g.conf.Transport.IdleConnTimeout,
  }
  g.client = http.Client{Transport: tr}
  return nil
}

func (g *Generator) Shoot(payload core.Ammo) {
  p, ok := payload.(*Payload)
  if !ok {
    g.deps.Log.Error("unexpected payload type", zap.Any("payload", payload))
    return
  }
  g.shoot(p)
}
```

И наконец пишем логику

```go
func (g *Generator) shoot(payload *Payload) {
  ctx := context.Background()
  authToken, err := g.auth(ctx, payload.UserID)
  if err != nil {
    g.deps.Log.Error("cant get auth token", zap.Error(err))
    return
  }
  time.Sleep(g.conf.Sleep)

  itemIDs, err := g.list(ctx, payload.UserID, authToken)
  if err != nil {
    g.deps.Log.Error("cant get item list", zap.Error(err))
    return
  }
  time.Sleep(g.conf.Sleep)

  for i := 0; i < 3; i++ {
    itemID := itemIDs[rand.Intn(len(itemIDs))]
    err := g.order(ctx, itemID, payload.UserID, authToken)
    if err != nil {
      g.deps.Log.Error("cant get item list", zap.Error(err))
      return
    }
    time.Sleep(g.conf.Sleep)
  }
}
```

И наконец пишем логику

```go
func (g *Generator) shoot(payload *Payload) {
  ctx := context.Background()
  authToken, err := g.auth(ctx, payload.UserID)
  if err != nil {
    g.deps.Log.Error("cant get auth token", zap.Error(err))
    return
  }
  time.Sleep(g.conf.Sleep)

  itemIDs, err := g.list(ctx, payload.UserID, authToken)
  if err != nil {
    g.deps.Log.Error("cant get item list", zap.Error(err))
    return
  }
  time.Sleep(g.conf.Sleep)

  for i := 0; i < 3; i++ {
    itemID := itemIDs[rand.Intn(len(itemIDs))]
    err := g.order(ctx, itemID, payload.UserID, authToken)
    if err != nil {
      g.deps.Log.Error("cant get item list", zap.Error(err))
      return
    }
    time.Sleep(g.conf.Sleep)
  }
}
```

И наконец пишем логику

```go
func (g *Generator) shoot(payload *Payload) {
  ctx := context.Background()
  authToken, err := g.auth(ctx, payload.UserID)
  if err != nil {
    g.deps.Log.Error("cant get auth token", zap.Error(err))
    return
  }
  time.Sleep(g.conf.Sleep)

  itemIDs, err := g.list(ctx, payload.UserID, authToken)
  if err != nil {
    g.deps.Log.Error("cant get item list", zap.Error(err))
    return
  }
  time.Sleep(g.conf.Sleep)

  for i := 0; i < 3; i++ {
    itemID := itemIDs[rand.Intn(len(itemIDs))]
    err := g.order(ctx, itemID, payload.UserID, authToken)
    if err != nil {
      g.deps.Log.Error("cant get item list", zap.Error(err))
      return
    }
    time.Sleep(g.conf.Sleep)
  }
}
```

И наконец пишем логику

```go
func (g *Generator) shoot(payload *Payload) {
  ctx := context.Background()
  authToken, err := g.auth(ctx, payload.UserID)
  if err != nil {
    g.deps.Log.Error("cant get auth token", zap.Error(err))
    return
  }
  time.Sleep(g.conf.Sleep)

  itemIDs, err := g.list(ctx, payload.UserID, authToken)
  if err != nil {
    g.deps.Log.Error("cant get item list", zap.Error(err))
    return
  }
  time.Sleep(g.conf.Sleep)

  for i := 0; i < 3; i++ {
    itemID := itemIDs[rand.Intn(len(itemIDs))]
    err := g.order(ctx, itemID, payload.UserID, authToken)
    if err != nil {
      g.deps.Log.Error("cant get item list", zap.Error(err))
      return
    }
    time.Sleep(g.conf.Sleep)
  }
}
```

И наконец пишем логику

```go
func (g *Generator) shoot(payload *Payload) {
  ctx := context.Background()
  authToken, err := g.auth(ctx, payload.UserID)
  if err != nil {
    g.deps.Log.Error("cant get auth token", zap.Error(err))
    return
  }
  time.Sleep(g.conf.Sleep)

  itemIDs, err := g.list(ctx, payload.UserID, authToken)
  if err != nil {
    g.deps.Log.Error("cant get item list", zap.Error(err))
    return
  }
  time.Sleep(g.conf.Sleep)

  for i := 0; i < 3; i++ {
    itemID := itemIDs[rand.Intn(len(itemIDs))]
    err := g.order(ctx, itemID, payload.UserID, authToken)
    if err != nil {
      g.deps.Log.Error("cant get item list", zap.Error(err))
      return
    }
    time.Sleep(g.conf.Sleep)
  }
}
```

И пример реализации одного метода

```
func (g *Generator) auth(ctx context.Context, userID int64) (token string, err error) {
  sample := netsample.Acquire("auth")
  sampleCode := 0
  defer func() {
    sample.SetProtoCode(sampleCode)
    g.aggr.Report(sample)
  }()


  // TODO:
}
```

И пример реализации одного метода

```
func (g *Generator) auth(ctx context.Context, userID int64) (token string, err error) {
  // Sample

  addr := fmt.Sprintf("http://%s/auth?sleep=%d",
                      g.conf.Target, g.conf.Sleep.Milliseconds())
  body := strings.NewReader(fmt.Sprintf(`{"user_id": %d}`, userID))
  req, err := http.NewRequestWithContext(ctx, http.MethodPost, addr, body)
  if err != nil {
    sampleCode = http.StatusBadRequest
    return "", fmt.Errorf("failed to create request: %w", err)
  }
  req.Header.Set("Content-Type", "application/json")

  resp, err := g.client.Do(req)
  sampleCode = resp.StatusCode
  if err != nil {
    return "", fmt.Errorf("failed to do request: %w", err)
  }
  defer resp.Body.Close()

  // TODO: Parsing body
}
```

И пример реализации одного метода

```
func (g *Generator) auth(ctx context.Context, userID int64) (token string, err error) {
  // Sample
  // Request


  result := struct {
    AuthKey string `json:"auth_key"`
  }{}
  err = json.NewDecoder(resp.Body).Decode(&result)
  if err != nil {
    sampleCode = http.StatusBadRequest
    return "", fmt.Errorf("failed to decode response: %w", err)
  }

  return result.AuthKey, nil
}
```

Я железный человек

Параметры запуска

```
pools:
  - id: Just-pool-name
    gun:
      type: custom_generator
      target: localhost:8092
      sleep: 100ms
      reqSleep: 100ms
    ammo:
      type: custom_provider
      source:
        type: file
        path: ./users.json
    result:
      type: phout
      destination: phout.log
    startup:
      type: once
      times: 100
    rps:
      - type: unlimited
        duration: 5s
log:
  level: info
```

```
  - id: Just-pool-name
    gun:
      type: custom_generator
      target: localhost:8092
      sleep: 100ms
      reqSleep: 100ms
    ammo:
      type: users/json
      source:
        type: file
        path: ./users.json
    result:
      type: phout
      destination: phout.log
    startup:
      type: once
      times: 100
    rps:
      - type: unlimited
        duration: 5s
log:
  level: info%

# sabevzenko @ sabevzenko-osx in ~/tests/custom_pandora on git:main x [13:13:14]
$ go build -o pandora
```

Custom Pandora Generator

Using Go

- Powerful
- Flexible
- Все GO библиотеки вам в помощь

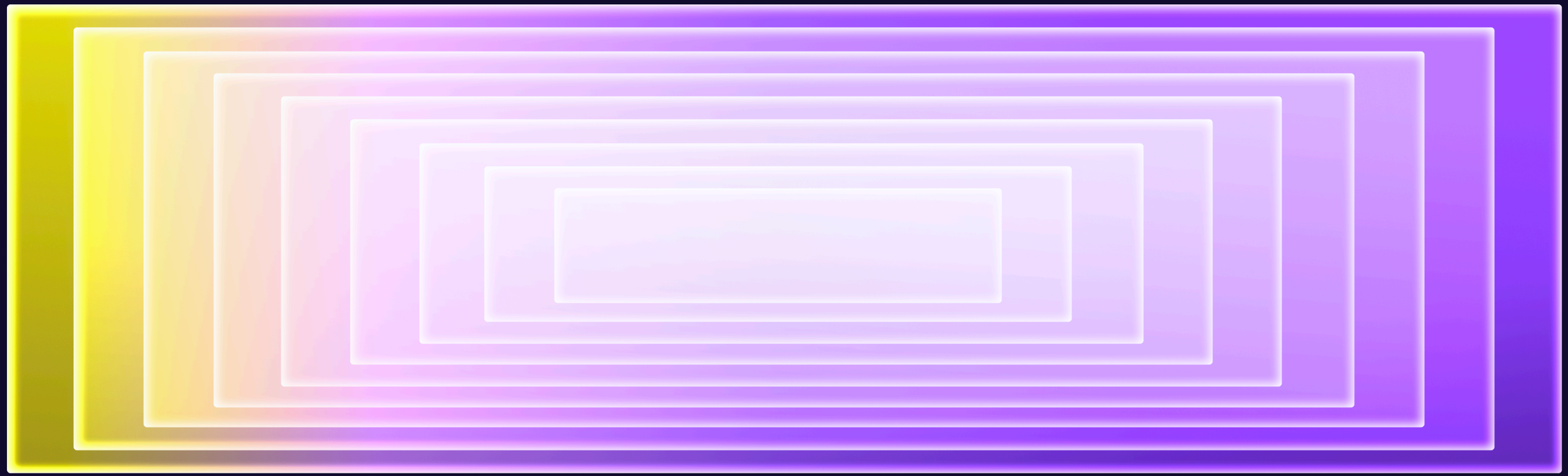
Custom Pandora Generator

Reasons

- By design
- Direction to framework

Давайте для сложных вещей использовать сложные инструменты, а для простых – простые.

Ганди



Pandora HTTP Scenario

Pandora HTTP Scenario Generator

Возможности

1. Описать сценарии в разных форматах
   - yaml, json, hcl (terraform)
2. Получать переменные из файлов
   - csv, json
3. Парсить ответ предыдущего запроса
   - jsonpath, xpath, headers
4. Использовать шаблонизатор запроса
   - go templates (text, html)

Описание файла сценариев

1. yaml
2. json
3. hcl (terraform)

Описание файла сценариев

HLC

1. scenarios
2. requests
3. variable_sources

Простой сценарий — 1 запрос

List

```
Content-Type: application/json
Authorization: "Bearer {{token}}"
GET /list
```

```
200 OK
{
  "data": {
    "items": [ ..... ]
  }
}
```

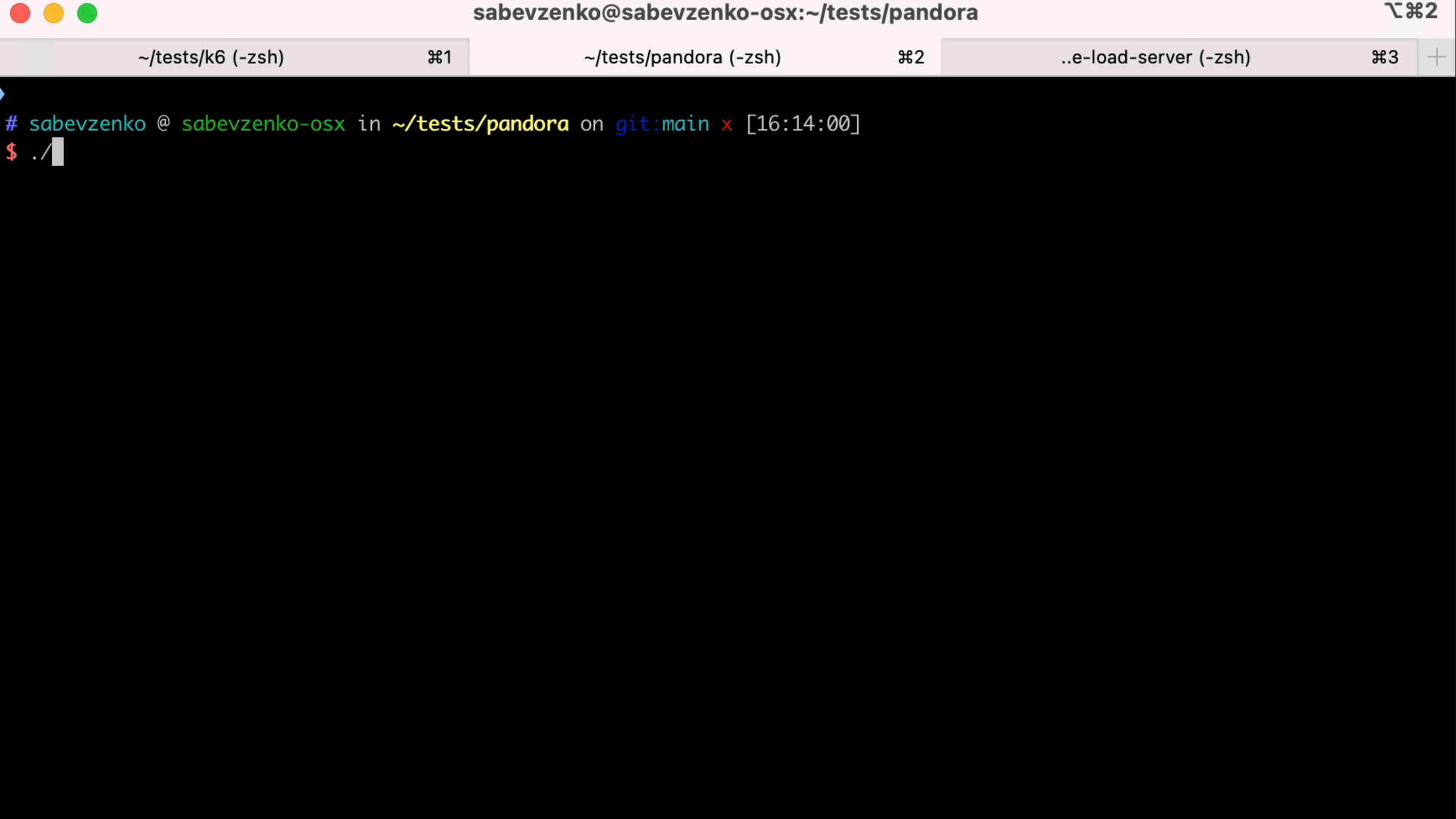
Описание файла сценариев

```
request "list_req" {
  method  = “GET"
  uri = "/list?sleep=100"
  headers = {
    Authorization = "Bearer
HCcbSueVAYbwYjYNxhhNDrWEioaVZSJuqFeVOgZdlKeQeQmWTTzpRdmIYtWmqKMS-4"
    Content-Type  = "application/json"
    Useragent     = "Yandex"
  }
  tag = "list"
}

scenario "scenario_name" {
  requests = [
    "list_req",
  ]
}
```

payload-1.hcl

Конфиг для запуска

```
pools:
  - id: Simple Scenario
    gun:
      type: http/scenario
      target: localhost:8092
    ammo:
      type: http/scenario
      file: payload.hcl
    result:
      type: phout
      destination: out/phout.log
    startup:
      type: once
      times: 100
    rps:
      - type: unlimited
        duration: 5s
```


sabevzenko@sabevzenko-osx:~/tests/pandora
⌥⌘2
~/tests/k6 (-zsh) ⌘1
~/tests/pandora (-zsh) ⌘2
..e-load-server (-zsh) ⌘3
sabevzenko @ sabevzenko-osx in ~/tests/pandora on git:main x [16:14:00]
$ ./

Добавим 2-й запрос и переменные

List

```
Content-Type: application/json
Authorization: "Bearer {{token}}"
GET /list
```

```
200 OK
{
  "data": {
    "items": [ ..... ]
  }
}
```

Order

```
POST /order
Content-Type: application/json
Authorization: "Bearer {{token}}"
{
  "item_id": {{item_id}}
}
```

```
200 OK
{
  "data": {
    "order_id": 17233
  }
}
```

```
request "list_req" {
  …
  postprocessor "var/jsonpath" {
    mapping = {
      first_item_id = "$.items[0]"
    }
  }
}

request "order_req" {
  method = "POST"
  uri    = "/order?sleep=100"
  headers = {
    Authorization = "Bearer
HCcbSueVAYbwYjYNxhhNDrWEioaVZSJuqFeVOgZdlKeQeQmWTTzpRdmIYtWmqKMS-4"
    Content-Type  = "application/json"
    Useragent     = "Yandex"
  }
  body = <<EOF
{"item_id": {{.request.list_req.postprocessor.first_item_id}}}
EOF
  tag  = "order"
}
```

Добавим 2-й запрос

```
scenario "scenario_name" {
  requests = [
    "list_req",
    "order_req",
    "order_req",
    "order_req"
  ]
}
```

Добавим 2-й запрос

```
scenario "scenario_name" {
  requests = [
    "list_req",
    "order_req",
    "order_req",
    "order_req"
  ]
}

scenario "scenario_name" {
  requests = [
    "list_req",
    "order_req(3)"
  ]
}
```

Добавим 2-й запрос и thinking time

```
scenario "scenario_name" {
  requests = [
    "list_req",
    "order_req",
    "order_req",
    "order_req"
  ]
}

scenario "scenario_name" {
  requests = [
    "list_req",
    "order_req(3)"
  ]
}
```

```
scenario "scenario_name" {
  requests = [
    "list_req",
    "sleep(100)",
    "order_req(3)"
  ]
}
```

Добавим 2-й запрос и thinking time

```
scenario "scenario_name" {
  requests = [
    "list_req",
    "order_req",
    "order_req",
    "order_req"
  ]
}
```

```
scenario "scenario_name" {
  requests = [
    "list_req",
    "sleep(100)",
    "order_req(3)"
  ]
}
```

```
scenario "scenario_name" {
  requests = [
    "list_req",
    "order_req(3)"
  ]
}
```

```
scenario "scenario_name" {
  requests = [
    "list_req",
    "sleep(100)",
    "order_req(3, 100)"
  ]
}
```

Добавим 2-й запрос и thinking time

```
scenario "scenario_name" {
  requests = [
    "list_req",
    "order_req",
    "order_req",
    "order_req"
  ]
}
```

```
scenario "scenario_name" {
  requests = [
    "list_req",
    "order_req(3)"
  ]
}
```

```
scenario "scenario_name" {
  requests = [
    "list_req",
    "sleep(100)",
    "order_req(3)"
  ]
}
```

```
scenario "scenario_name" {
  min_waiting_time = 1000
  requests = [
    "list_req",
    "sleep(100)",
    "order_req(3, 100)"
  ]
}
```

```
request "list_req" {
  …
  postprocessor "var/jsonpath" {
    mapping = {
      first_item_id = "$.items[0]"
    }
  }
}

request "order_req" {
  method = "POST"
  uri    = "/order?sleep=100"
  headers = {
    Authorization = "Bearer
HCcbSueVAYbwYjYNxhhNDrWEioaVZSJuqFeVOgZdlKeQeQmWTTzpRdmIYtWmqKMS-4"
    Content-Type  = "application/json"
    Useragent     = "Yandex"
  }
  body = <<EOF
{"item_id": {{.request.list_req.postprocessor.first_item_id}}}
EOF
  tag  = "order"
}
```

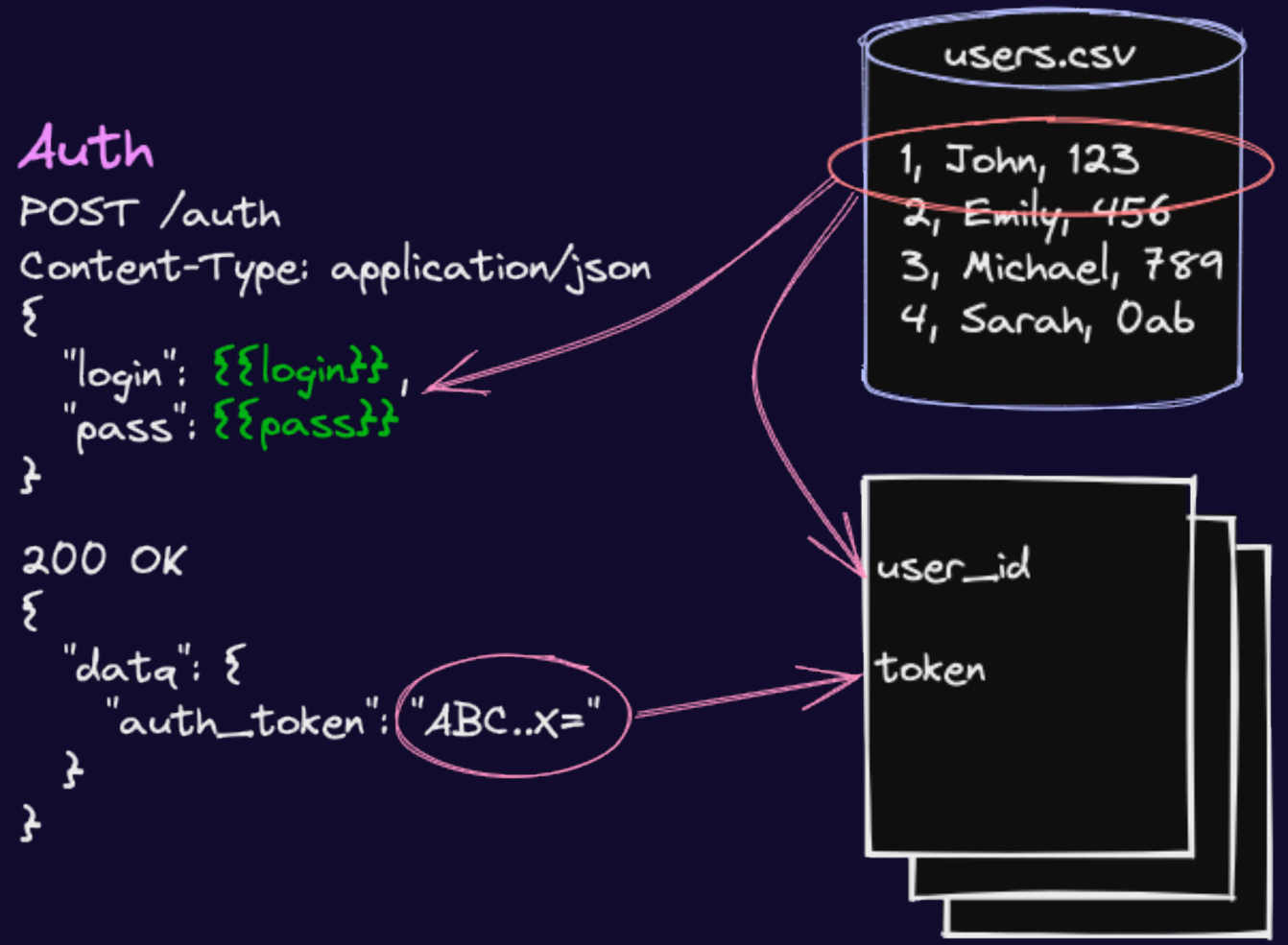
```
request "list_req" {
  …
  postprocessor "var/jsonpath" {
    mapping = {
      items   = "$.items"
    }
  }
}

request "order_req" {
  …
  body = <<EOF
{"item_id": {{.request.order_req.preprocessor.item}}}
EOF
  tag  = "order"

  preprocessor {
    mapping = {
      item = "request.list_req.postprocessor.items[next]"
    }
  }
}
```

- next
- rand
- last


./pandora scenario-4.yaml
⌥⌘2
~ (-zsh) ⌘1
./pandora (-zsh) ⌘2
..e-load-server (-zsh) ⌘3
sabevzenko @ sabevzenko-osx in ~/tests/pandora on git:main x [17:22:03]
$ ./pandora scenario-4.yaml
2023-10-08T17:22:17.768+0200 INFO cli/cli.go:229 Reading config {"file": "scenario-4.yaml"}
2023-10-08T17:22:17.796+0200 INFO engine/engine.go:138 Pool run started {"pool": "Simple Scenario"}

Нужна авторизация

```
user_id,name,pass
1,John,123
2,Jack,456
3,Jim,789
4,Joe,0asdf
5,Jane,12dasdf
6,Jill,asdf
7,Joy,zxcv
8,Joy,zxcv
9,Joy,zxcv
10,Joy,zxcv
```



Нужна авторизация

```
variable_source "users" "file/csv" {
  file              = "users.csv"
  fields            = ["user_id", "name", "pass"]
  ignore_first_line = true
  delimiter         = ","
}
```

```
user_id,name,pass
1,John,123
2,Jack,456
3,Jim,789
4,Joe,0asdf
5,Jane,12dasdf
6,Jill,asdf
```

```
{{ .source.users[0].name }}
```

Нужна авторизация

```
request "auth_req" {
  method  = "POST"
  uri     = “/auth?sleep=100”
  headers = {
    Content-Type = "application/json"
    Useragent    = "Yandex"
  }
  tag  = "auth"
  body = <<EOF
{"user_id":  {{ .TODO }}}
EOF
  preprocessor {
    …
  }
  postprocessor "var/jsonpath" {
    …
  }
}
```

Нужна авторизация

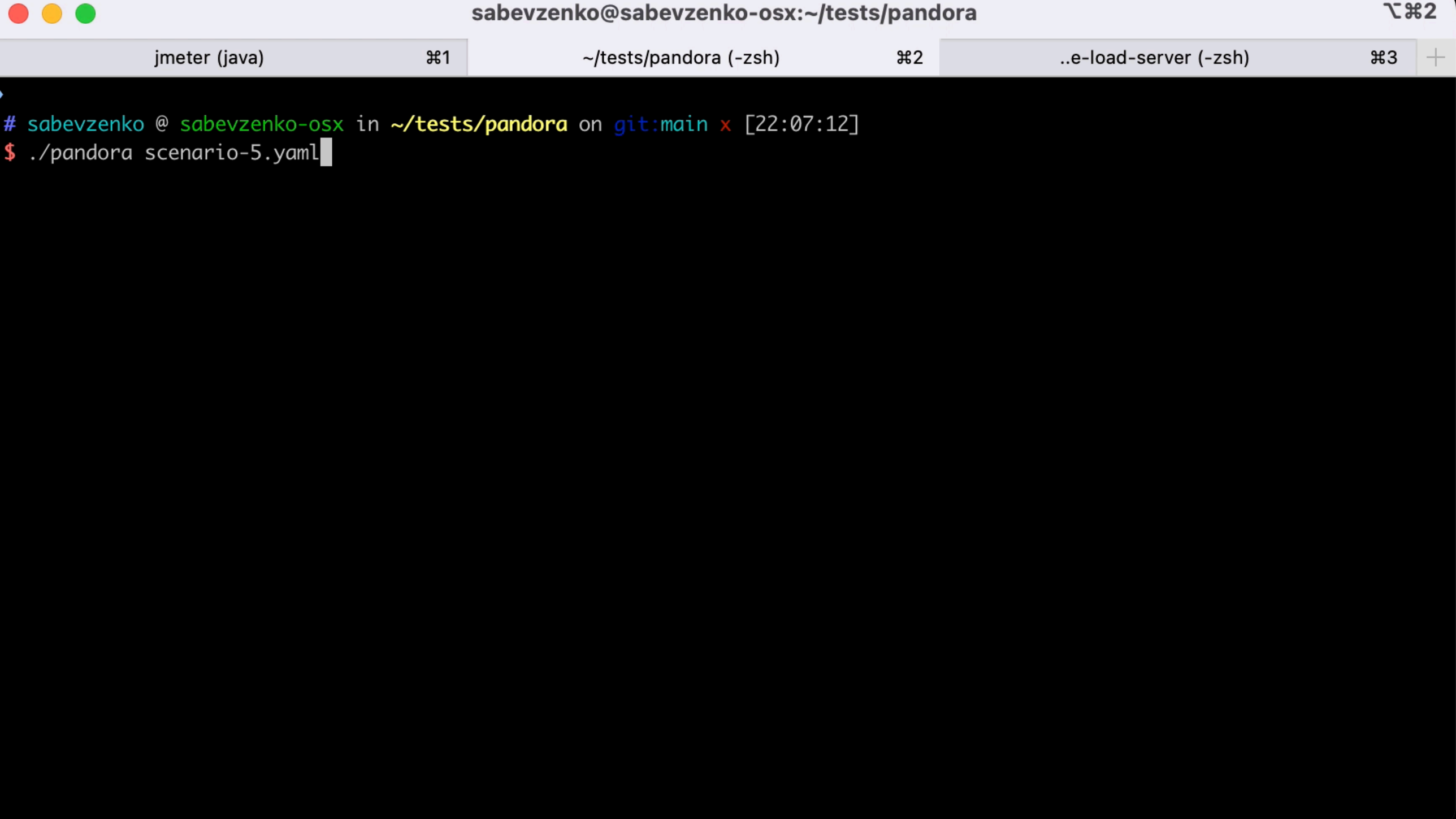
```
request "auth_req" {
  …
  body = <<EOF
{"user_id":  {{.request.auth_req.preprocessor.user_id}}}
EOF

  preprocessor {
    mapping = {
      user_id = "source.users[next].user_id"
    }
  }
  postprocessor "var/jsonpath" {
    mapping = {
      token = "$.auth_key"
    }
  }
}
```

Нужна авторизация

```
request "list_req" {
  …
  headers = {
    Authorization = "Bearer {{.request.auth_req.postprocessor.token}}”
    …
  }
  …
}

request "order_req" {
  …
  headers = {
    Authorization = "Bearer {{.request.auth_req.postprocessor.token}}”
    …
  }
  …
}
```

Нужна авторизация

```
scenario "scenario_name" {
  requests = [
    "auth_req",
    "sleep(100)",
    "list_req",
    "sleep(100)",
    "order_req(3, 100)"
  ]
}
```


sabevzenko@sabevzenko-osx:~/tests/pandora
⌥⌘2
jmeter (java) ⌘1
~/tests/pandora (-zsh) ⌘2
..e-load-server (-zsh) ⌘3
sabevzenko @ sabevzenko-osx in ~/tests/pandora on git:main x [22:07:12]
$ ./pandora scenario-5.yaml

More postprocessors

xpath

```
postprocessor "var/xpath" {
  mapping = {
    data = "//div[@class='data']"
  }
}
```

More postprocessors

asserts

```
postprocessor "assert/response" {
  headers = {
    Content-Type = "json"
  }
  body = ["key"]
  size {
    val = 40
    op  = ">"
  }
}
```

Много сценариев

```
scenario "scenario_first" {
  weight = 1
  requests = [
    "auth_req(1, 100)",
    "list_req(1, 100)",
    "order_req(3, 100)"
  ]
}

scenario "scenario_second" {
  weight = 50
  requests = [
    "mainpage",
  ]
}
```

Профиль нагрузки

```yaml
rps:
  - duration: 2m
    type: line
    from: 100
    to: 1000
  - duration: 10m
    type: const
    ops: 1000
```

https://cloud.yandex.ru/docs/load-testing/concepts/load-profile

https://cloud.yandex.ru/docs/load-testing/concepts/testing-stream

Phout

```
1696576823.143 scenario_name.auth  1080  285 41  207     24  0 107 190 0 200
1696576823.245 scenario_name.list  102018  4 7   101501  23  0 190 639 0 200
1696576823.448 scenario_name.order 101920  9 29  101011  28  0 210 124 0 200
1696576823.643 scenario_name.auth  1035  326 53  220     17  0 107 190 0 200
1696576823.651 scenario_name.order 101563  4 10  101066  30  0 210 124 0 200
1696576823.745 scenario_name.list  102601  4 18  101746  42  0 190 639 0 200
1696576823.853 scenario_name.order 101297  3 12  100880  21  0 210 124 0 200
1696576823.948 scenario_name.order 101952  3 11  101474  26  0 210 124 0 200
                                   1       2  3        4   5  6  7   8   9  10
```

1. RTTMicro
2. ConnectMicro
3. SendMicro
4. LatencyMicro
5. ReceiveMicro
6. IntervalEventMicro
7. RequestBytes
8. ResponseBytes
9. Errno
10. ProtoCode

Log level: debug

```
pools:
  …

log:
  level: debug
```

Log level: debug


```
2023-10-06T09:20:27.550+0200    DEBUG   http_scenario/gun.go:240    Postprocessor variables {"pool":
"Simple Scenario", "instance": 1, ".resuest.order_req.postprocessor": {}}
2023-10-06T09:20:27.644+0200    DEBUG   http_scenario/gun.go:146    Preprocessor variables  {"pool":
"Simple Scenario", "instance": 0, ".resuest.auth_req.preprocessor": {"user_id":"10"}}
2023-10-06T09:20:27.645+0200    DEBUG   http_scenario/gun.go:331    Request debug info      {"pool":
"Simple Scenario", "instance": 0, "URL": "http://[::1]:8092/auth", "Host": "localhost", "Headers":
{"Content-Type":["application/json"],"Useragent":["Yandex"]}, "Body": "POST /auth HTTP/1.1\r\nHost:
localhost\r\nUser-Agent: Go-http-client/1.1\r\nContent-Length: 17\r\nContent-Type: application/
json\r\nUseragent: Yandex\r\nAccept-Encoding: gzip\r\n\r\n{\"user_id\":  10}\n"}
2023-10-06T09:20:27.645+0200    DEBUG   http_scenario/gun.go:340    Response debug info     {"pool":
"Simple Scenario", "instance": 0, "Status Code": 200, "Status": "200 OK", "Headers": {"Content-Length":
["83"],"Content-Type":["application/json"],"Date":["Fri, 06 Oct 2023 07:20:27 GMT"]}, "Body":
"{\"auth_key\": \"SRIcuhZKKiCqrlaeuNRHPpfBUClzFyBpgxoPTGwJtOesTGQmacvMwERgSsoGOseO-10\"}"}
```

Debug

```
answlog:
  enabled: true
  path: out/answ.log
  filter: all
```

Debug

```
2023-10-06T09:20:23.144+0200  DEBUG REQUEST[scenario_name.auth.7691227975663582509]:
POST /auth HTTP/1.1
Host: localhost
User-Agent: Go-http-client/1.1
Content-Length: 16
Content-Type: application/json
Useragent: Yandex
Accept-Encoding: gzip

{"user_id":  1}

2023-10-06T09:20:23.144+0200  DEBUG RESPONSE[scenario_name.auth.7691227975663582509]:
HTTP/1.1 200 OK
Content-Length: 82
Content-Type: application/json
Date: Fri, 06 Oct 2023 07:20:23 GMT

{"auth_key": "oqwkQrDueVUVlOZDLRghFTQFdcjmMlEehuQqxhRrBYLEVHgcnLXJzfQIYkgkQewg-1"}
```

Код

Примеры для эксперимента можно посмотреть на GitHub.

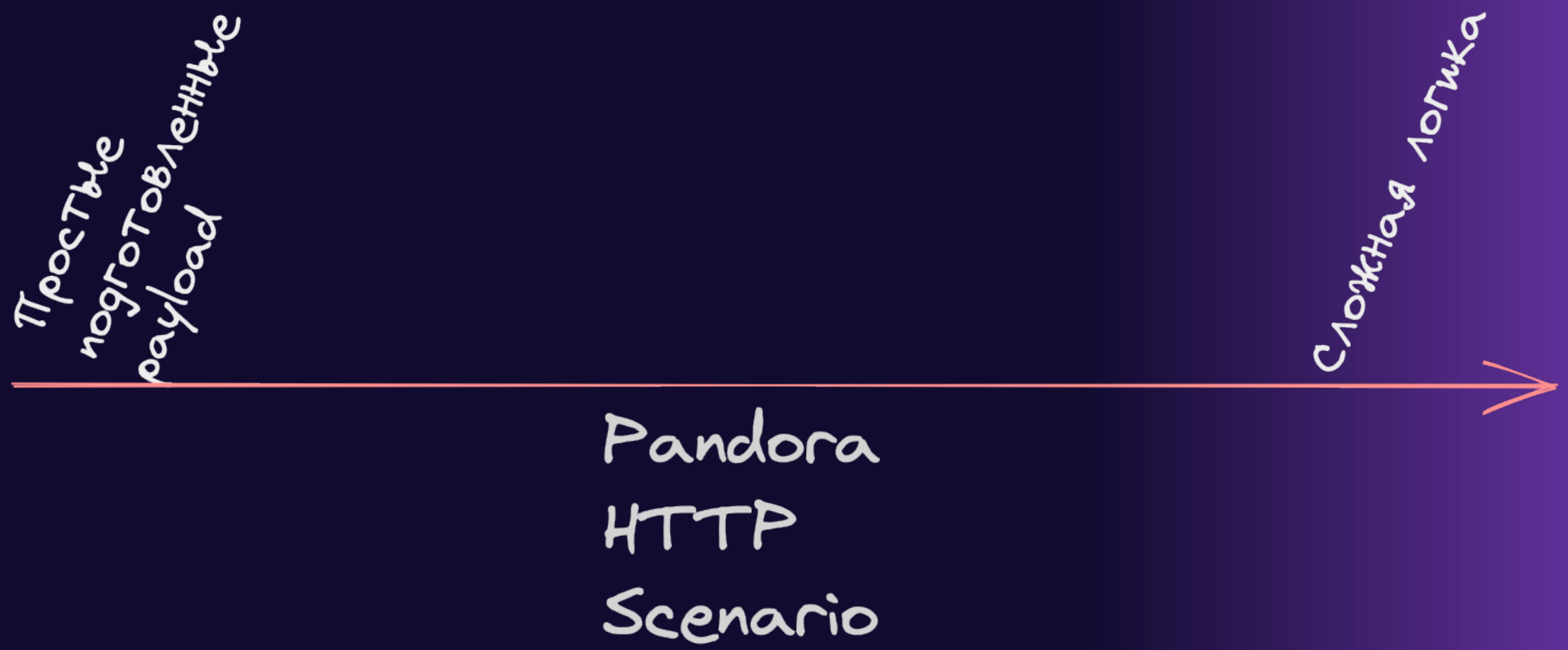
- код тестового сервера
- конфиг для pandora
- конфиг для jmeter
- файлы с тестовыми данными
- Makefile со скриптами запуска

https://github.com/oke11o/pandora-presentaion

Заключение

- JMeter
  - 18 компонентов
  - 317 строк Kotlin
  - 648 строк of XML
- k6
  - 71 строк Javascript
- Custom Pandora
  - 220 строк Go
- Scenario Pandora
  - 71 строк HCL

Заключение
Где использовать



Заключение
Где использовать

- Вы хорошо пишете кастомные генераторы на языках программирования
  - Go
  - Java
  - Kotlin
  - Javascript
- У вас есть много сценариев на JMeter и вам комфортно их поддерживать

Заключение
Когда?

- Это добавлено в версии 0.5.9

Заключение
Планы

- документация
- asserts
- новые источники
- шаблонизаторы
- gRPC

Заключение

https://github.com/yandex/pandora

https://t.me/oke11o

Присоединяйтесь к Yandex Infrastructure в телеграмм

Канал о том, как мы развиваем
и поддерживаем IT-инфраструктуру,
на которой работает весь Яндекс.

https://t.me/
yandex_infrastructure

Спасибо!
Можно задать вопросы

Сергей Бевзенко
Yandex Infrastructure

https://t.me/oke11o