tech

Сквозное логирование в автоматизации тестирования

Чаплашкин Олег

бизнес

Познакомимся?

- Инженер по автоматизации тестирования в Tarantool
- Старший преподаватель кафедры ИиВМ в Самарском университете
- Ментор на платформе GetMentor

@ochaplashkin




getmentor

Поговорим...

- о том, как тестируют Tarantool
- о проблеме логирования в автоматизации тестирования
- о поиске нашего «идеального» решения
- об итоговой реализации на Lua

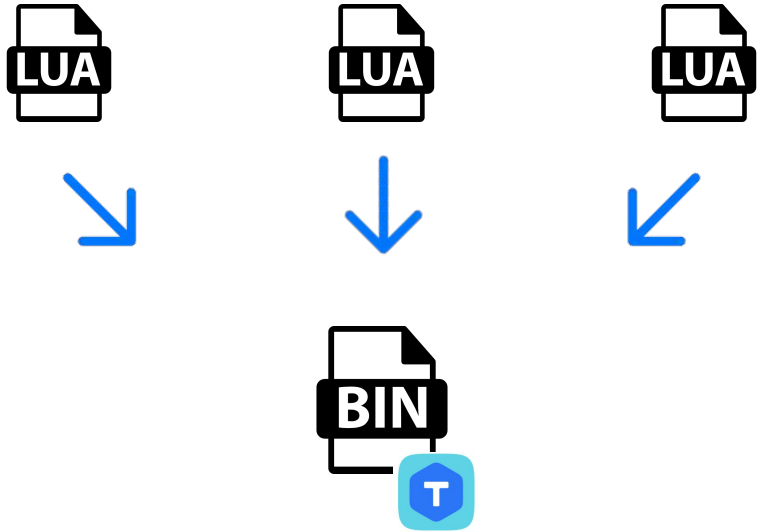
Тестирование Tarantool

Тестирование Tarantool

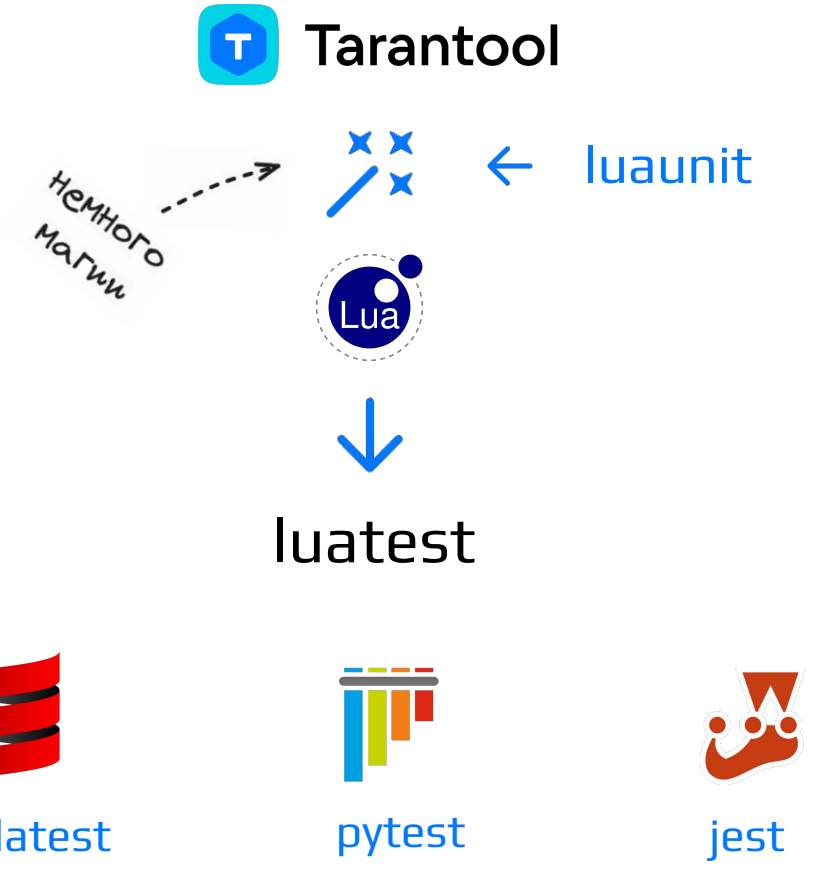
- модульные тесты на C
- diff-тестирование
- TAP13 спецификация
- bash / python / lua - скрипты

Тестирование СУБД: 10 лет опыта

Тестирование Tarantool: luatest



Тестирование Tarantool: luatest



Тестирование Tarantool: luatest

```
local t = require('luatest')
local server = require('luatest.server')
local g = t.group()
```

Тестирование Tarantool: luatest

```lua
local t = require('luatest')
local server = require('luatest.server')
local g = t.group()
```

Тестирование Tarantool: luatest

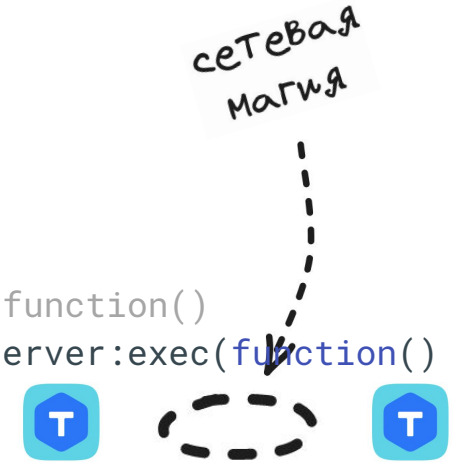
```
local t = require('luatest')
local server = require('luatest.server')
local g = t.group()

g.before_all(function()
    g.server = server:new()
    g.server:start()
end)

g.after_all(function()
    g.server:drop()
end)
```

Тестирование Tarantool: luatest

```
local t = require('luatest')
local server = require('luatest.server')
local g = t.group()

g.before_all(function()
    g.server = server:new()
    g.server:start()
end)

g.after_all(function()
    g.server:drop()
end)

g.test_one_plus_one = function()
    local actual = g.server:exec(function()
        return 1 + 1
    end)
    t.assert_equals(actual, 2)
end
```

Тестирование Tarantool: luatest

```lua
local t = require('luatest')
local server = require('luatest.server')
local g = t.group()

g.before_all(function()
    g.server = server:new()
    g.server:start()
end)

g.after_all(function()
    g.server:drop()
end)

g.test_one_plus_one = function()
    local actual = g.server:exec(function()
        return 1 + 1
    end)
    t.assert_equals(actual, 2)
end
```



Тестирование Tarantool

- Сервер приложений
- Конфигурация
- Алгоритмы репликации
- Метрики

Тестирование Tarantool

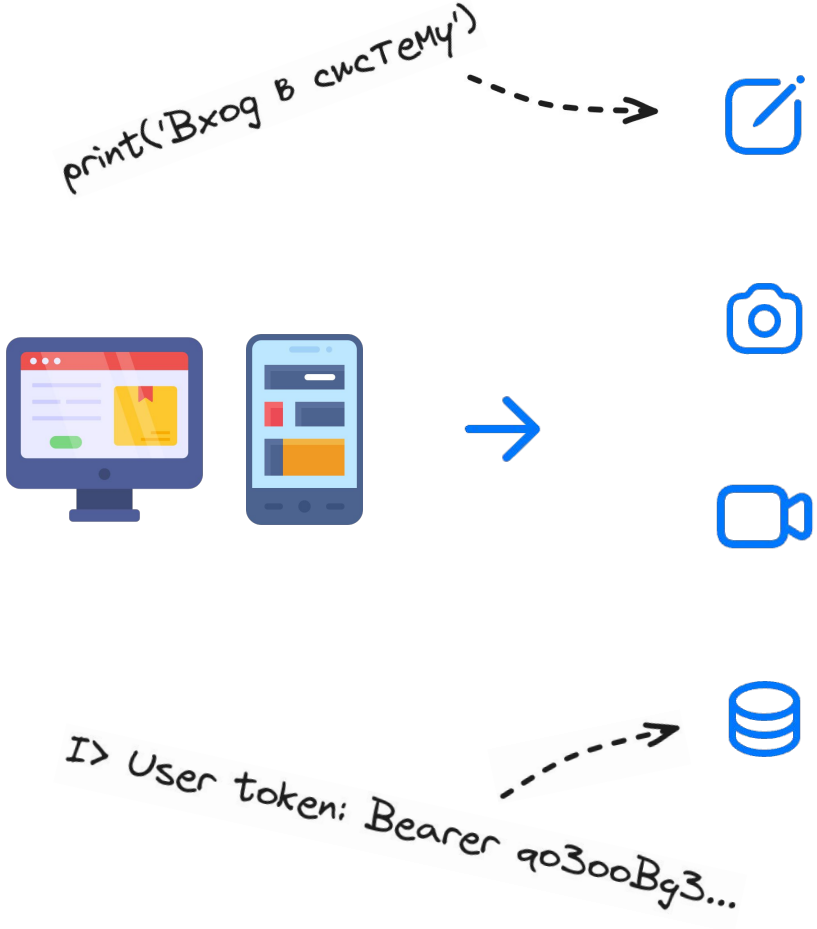
- Сервер приложений
- 🔥 Конфигурация
- 🔥 Алгоритмы репликации
- Метрики

Тестирование Tarantool

- Сервер приложений
- 🔥 Конфигурация
- 🔥 Алгоритмы репликации
- Метрики

Это точно проблема?

Это точно проблема?



Пытаемся найти подходящий инструмент

ELK

Реакция на событие

Сбор и отправка

Обработка и хранение

Выгрузка логов для автотестов

Медленно

это можно ускорить

Пытаемся найти подходящий инструмент

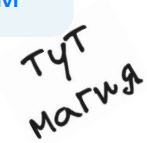

```
box.cfg{log = 'syslog:identity=tarantool'}
-- или
box.cfg{log = 'syslog:facility=user'}
-- или
box.cfg{log = 'syslog:server=unix:/dev/log'}
```

Пытаемся найти подходящий инструмент


```
box.cfg{log = 'syslog:identity=tarantool'}
-- или
box.cfg{log = 'syslog:facility=user'}
-- или
box.cfg{log = 'syslog:server=unix:/dev/log'}
```

Поддержка конфигурации

Разработчик следит за окружением

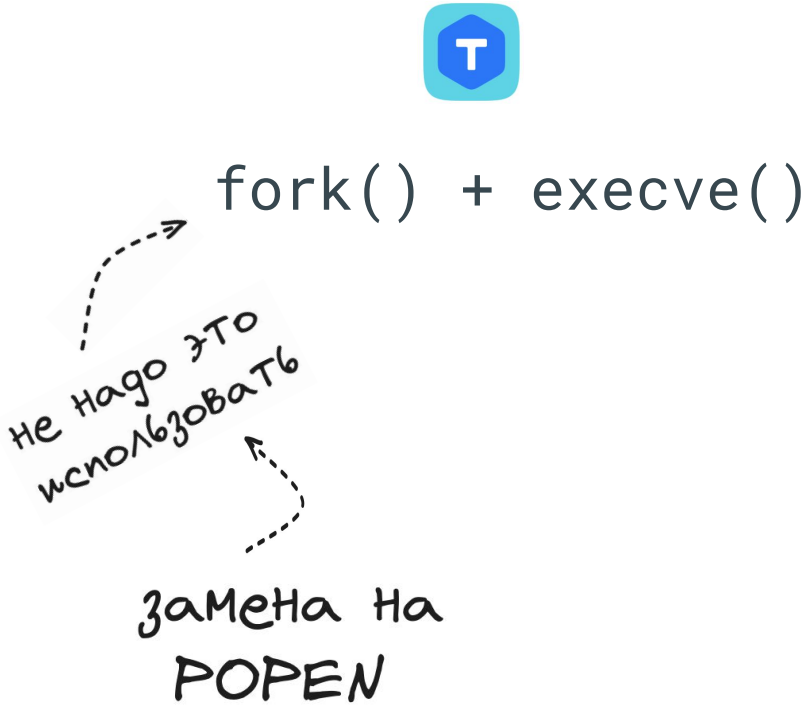


Изобретаем свое решение

Изобретаем свое решение

- **Запускать сервер Tarantool и записывать логи**
- **Собирать логи из стандартного вывода**
- Выполнять обработку логов

Изобретаем свое решение: запуск сервера



Изобретаем свое решение: запуск сервера

fork() + execve()

```
log.cfg{log = 'file: server.log'}

log.cfg{log = 'pipe: tee server.log'}
```

логи должны быть в stderr !

Изобретаем свое решение: запуск сервера

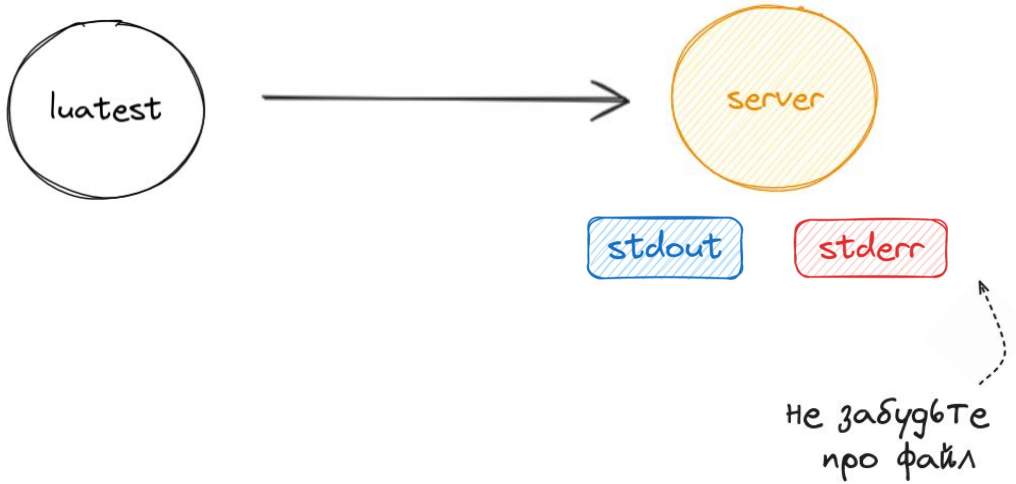
fork() + execve()

```
log.cfg{log = 'file: server.log'}

log.cfg{log = 'pipe: tee server.log'}

log.cfg{log = '| tee server.log > /dev/fd/2')
```

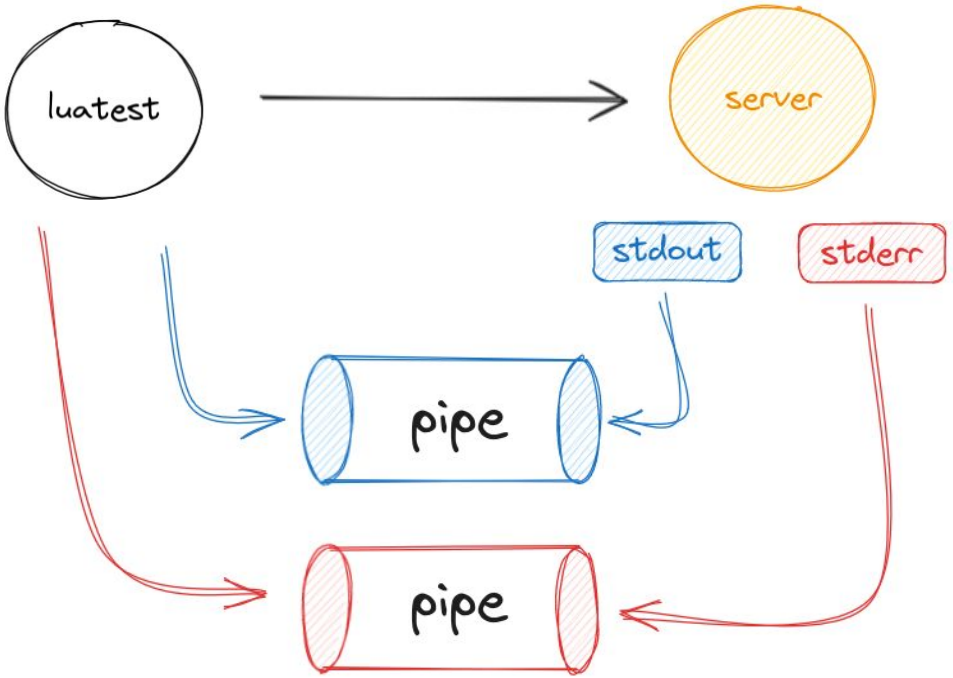
stderr

file

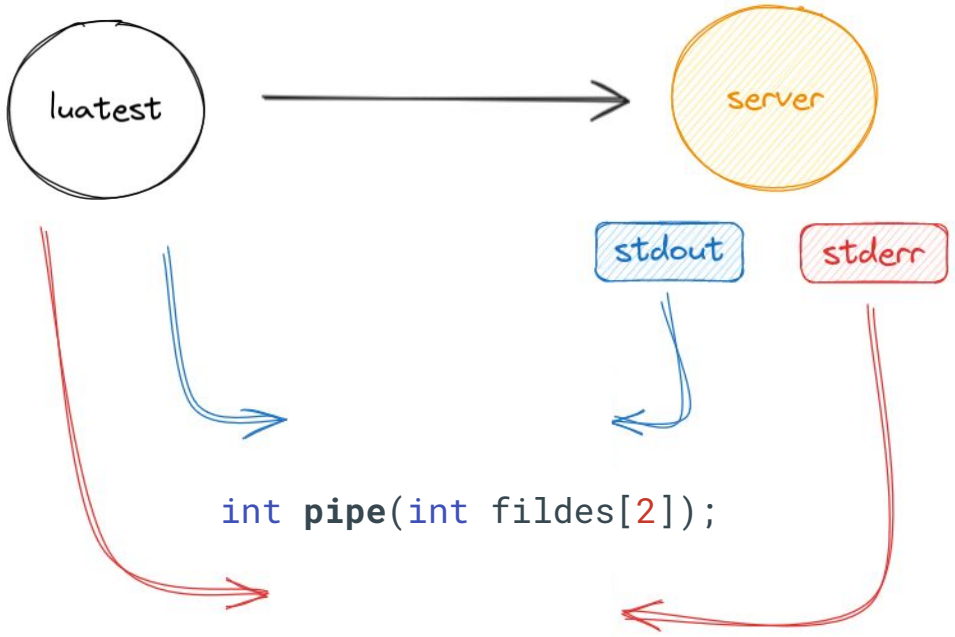
Изобретаем свое решение: сбор логов



Изобретаем свое решение: сбор логов



Изобретаем свое решение: сбор логов



Изобретаем свое решение: сбор логов

```
function read_fd(fd)
    local buffer_size = 4096
    local timeout = 1
    chunks = {}
    local buffer
    while socket.iowait(fd, 'R', timeout) ~= '' do
        buffer = ffi.new('char[?]', buffer_size)
        local count = ffi.C.read(fd, buffer, buffer_size)
        table.insert(chunks, ffi.string(buffer, count))
    end
    return chunks
end
```

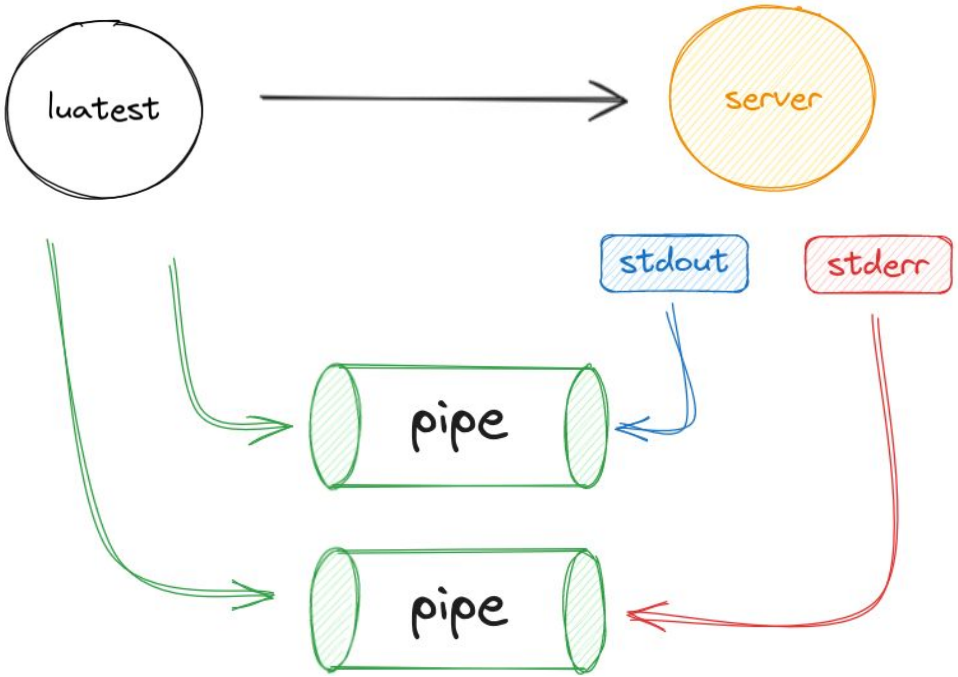
Изобретаем свое решение: сбор логов

```
function read_fd(fd)
    local buffer_size = 4096
    local timeout = 1
    chunks = {}
    local buffer
    while socket.iowait(fd, 'R', timeout) ~= '' do
        buffer = ffi.new('char[?]', buffer_size)
        local count = ffi.C.read(fd, buffer, buffer_size)
        table.insert(chunks, ffi.string(buffer, count))
    end
    return chunks
end
```

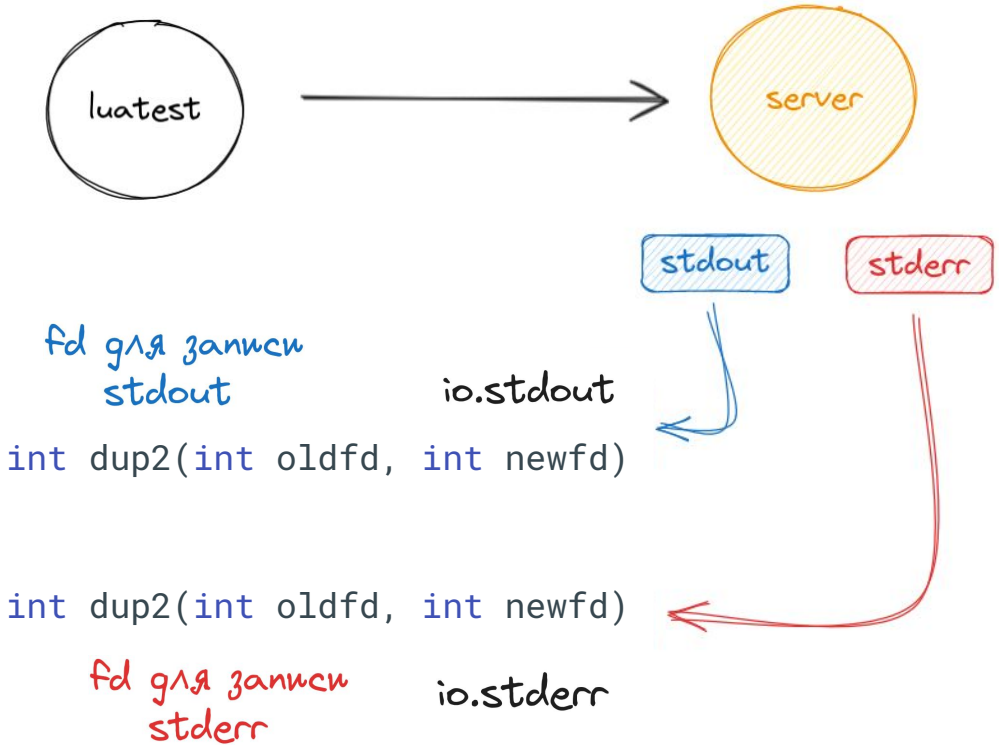
Изобретаем свое решение: сбор логов

```
local fiber = require('fiber')

fiber.new(function(fd)
    while fiber.testcancel() or true do
            read_fd(fd)
    end
    fiber.yield()
    end,
fd)
```

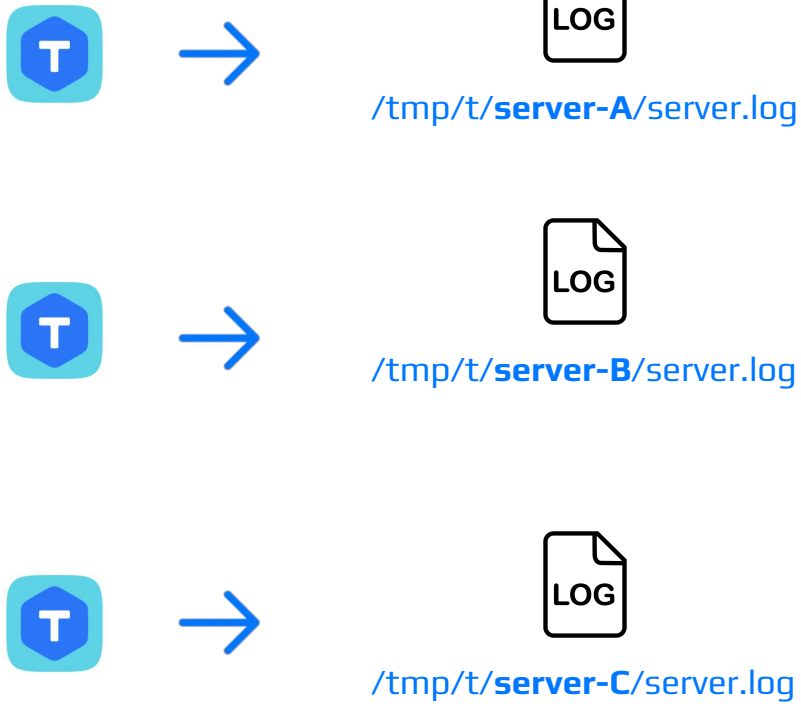
Изобретаем свое решение: сбор логов



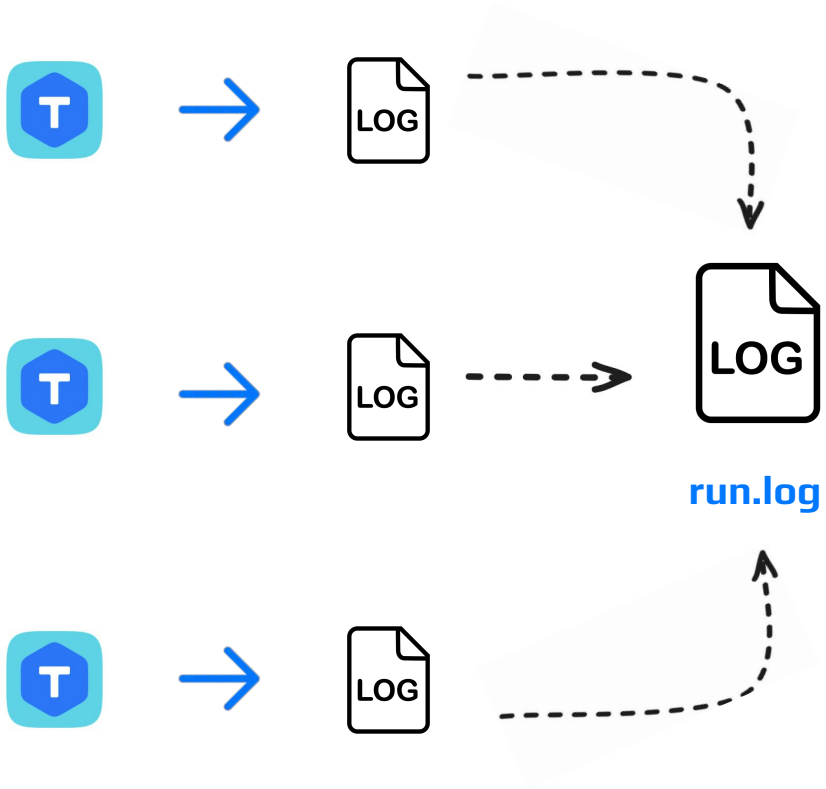
Изобретаем свое решение: сбор логов



Результаты:
что было



Результаты:
как стало

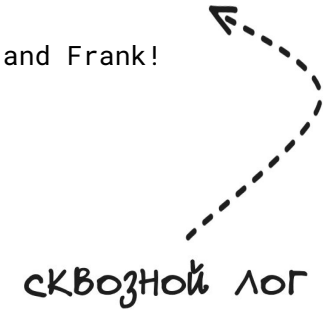


Результаты: как стало

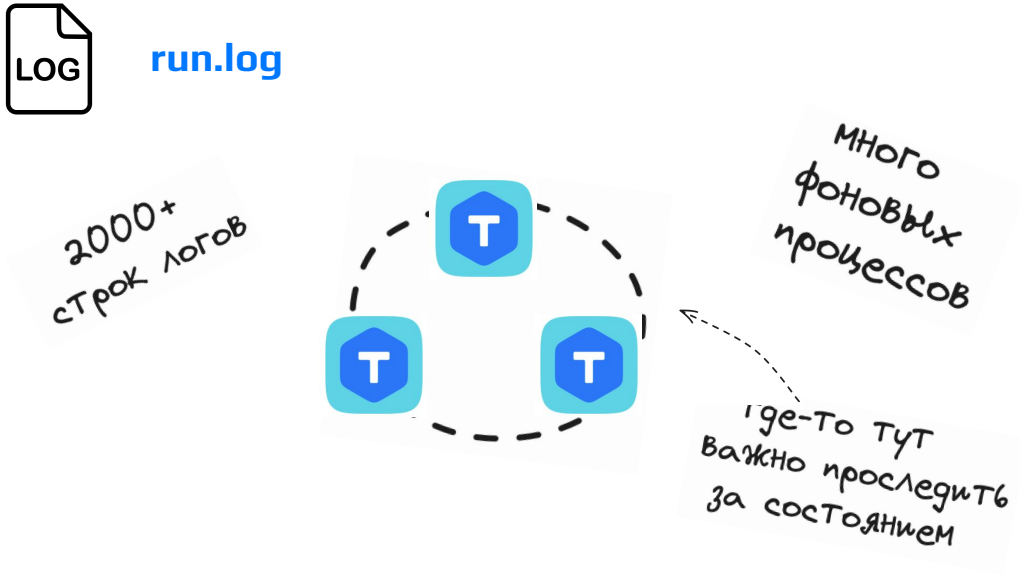
run.log

```
g.test_bob_and_frank = function()
    g.bob:exec(function() log.info('Hi, Frank!') end)
    g.frank:exec(function() log.info('Hi, Bob!') end)

    log.info('Hi Bob and Frank!')
end
```

```
bob     | main/109/main I> Hi, Frank!
frank   | main/109/main I> Hi, Bob!
luatest | main/109/main I> Hi, Bob and Frank!
```



Результаты: как стало



Raft в Tarantool. Как это работает и как этим пользоваться

Результаты: как стало

run.log

cat + grep "RAFT:"

```
bob   | 2024-04-18 16:55:55.398 I> RAFT: fencing paused
frank | 2024-04-18 16:55:55.401 I> RAFT: fencing paused
john  | 2024-04-18 16:55:55.405 I> RAFT: fencing paused
bob   | 2024-04-18 16:55:55.544 I> RAFT: start state machine
bob   | 2024-04-18 16:55:55.544 I> RAFT: bump term to 2, follow
bob   | 2024-04-18 16:55:55.544 I> RAFT: vote for 1, follow
bob   | 2024-04-18 16:55:55.545 I> RAFT: persisted state {term: 2, vote: 1}
bob   | 2024-04-18 16:55:55.545 I> RAFT: enter leader state with quorum 1
frank | 2024-04-18 16:55:55.690 I> RAFT: start state machine
frank | 2024-04-18 16:55:55.691 I> RAFT: message {term: 2, vote: 1, leader: 1, state: leader} from 1
frank | 2024-04-18 16:55:55.691 I> RAFT: received a newer term from 1
frank | 2024-04-18 16:55:55.691 I> RAFT: bump term to 2, follow
frank | 2024-04-18 16:55:55.691 I> RAFT: leader is 1, follow
frank | 2024-04-18 16:55:55.691 I> RAFT: message {term: 2, vote: 1, leader: 1, state: leader} from 1
frank | 2024-04-18 16:55:55.691 I> RAFT: vote request is skipped - the leader is already known - 1
frank | 2024-04-18 16:55:55.692 I> RAFT: persisted state {term: 2}
john  | 2024-04-18 16:55:55.695 I> RAFT: start state machine
john  | 2024-04-18 16:55:55.697 I> RAFT: message {term: 2, vote: 1, leader: 1, state: leader} from 1
john  | 2024-04-18 16:55:55.697 I> RAFT: received a newer term from 1
john  | 2024-04-18 16:55:55.697 I> RAFT: bump term to 2, follow
john  | 2024-04-18 16:55:55.697 I> RAFT: leader is 1, follow
```

Вместо итогов

- расширение тестового покрытия
- 🔥 **(r)syslog** для большинства систем
- 🔥 работа с логами может быть **быстрой** и удобной
- решение на поверхности спрятано глубоко внутри?

tech

Спасибо за внимание

tech.vk.com

бизнес