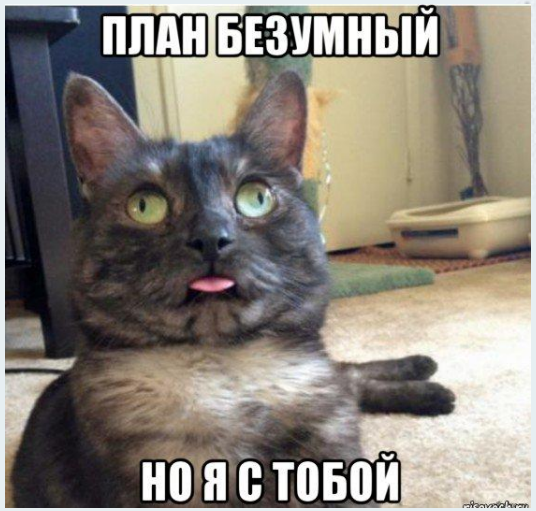
SIPssert в CI/CD Pipeline: Тестирование VoIP приложений

Обо мне:

- Образование - высшее, техническое (программная инженерия)
- Администратор сетей и инфраструктуры в Kolesa Group
- Опыт работы по направлению VoIP - 3 года
- Ключевые технологии: opensips, asterisk, rtpengine
- Хобби: тяжелая атлетика

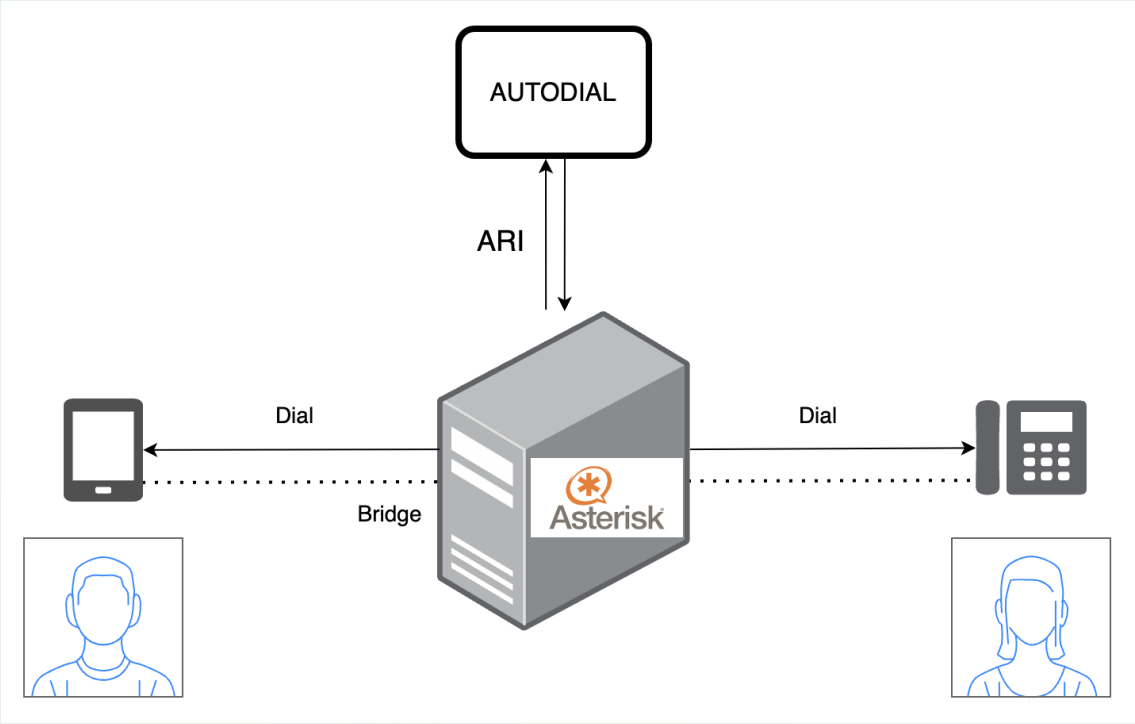
О чем доклад

- Интеграционное тестирование приложений
- Фреймворк SIPssert изнутри: от А до Я
- SIPssert на практике: как писать сценарии
- SIPssert как часть CI/CD Pipeline: запуск тестов
- Github Workflow: action-sipssert
- Расширение фреймворка SIPssert: generic tasks
- Интеграционное тестирование: демонстрация на реальном примере



Интеграционное тестирование

Простое приложение автодозвон



- Все работает?
- Так как задумано?
- Ничего не сломалось?
- А если пересоберу?

Ручное тестирование ✕

- А если много проектов?
- А если много функционала?
- А если мало тестировщиков?
- А если частые сборки?
- А если разные версии?

Автоматизированное тестирование ✓

	Mock	Тестовое окружение	docker compose	SIPssert
Изоляция	✓	✗	✓	✓
Гибкость	✗	✗	✓	✓
Автоматизация	✓	✗	✓	✓
Простота	✓	✓	✗	✓

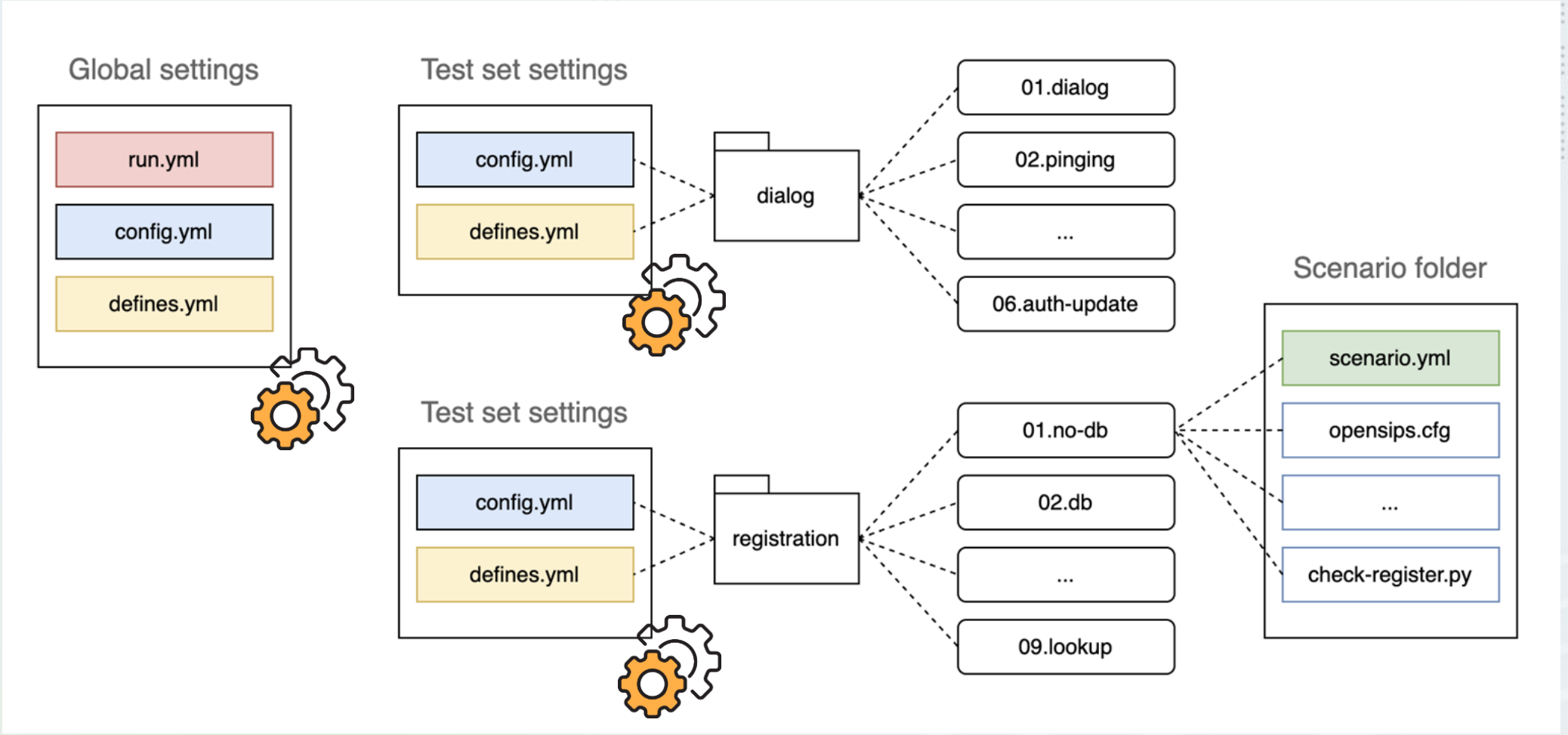
SIPssert - краткий обзор

- **SIPssert** - фреймворк для автоматизации тестирования VoIP приложений
- Анонсирован в апреле 2023 года на сайте blog.opensips.org
- Conformance testing, интеграционное тестирование
- Python3, YAML, Docker
- OpenSIPS/OpenSIPS CLI, SIPp/UAC/UAS, Asterisk, RTPProxy, MySQL/MySQL Client, PostgreSQL, Generic
- sipssert-opensips-tests: 130+ готовых сценариев

SIPssert - что предлагает?

			SIPssert	
Docker Engine API	Docker CLI		Docker SDK Python	-
Конфиги	YAML		YAML	-
Переменные	.env		defines.yml	Наследование
Наборы сценариев	✗		✓	Test sets
Статус выполнения	✗		✓	PASS/FAIL/TOUT
Управления зависимостями	depends_on		require, ready	daemon/non-daemon
Логирование	✗		✓	logs
Дампы	✗		✓	.pcap
Автомонтирование конфигов	✗		✓	*.xml, *.py, *.cfg

SIPssert - организация тестов



SIPssert - организация тестов

SIPssert - тесты в проекте

Тесты в репозитории проекта хранятся в директории *_sippsert*

```
.
├── CODEOWNERS
├── Dockerfile
├── Dockerfile.sipssert
├── README.md
├── _example
├── _sipssert
├── catalog-info.yaml
├── config
├── deploy
├── go.mod
├── go.sum
├── internal
├── main.go
├── makefile
└── microscope.yml
```

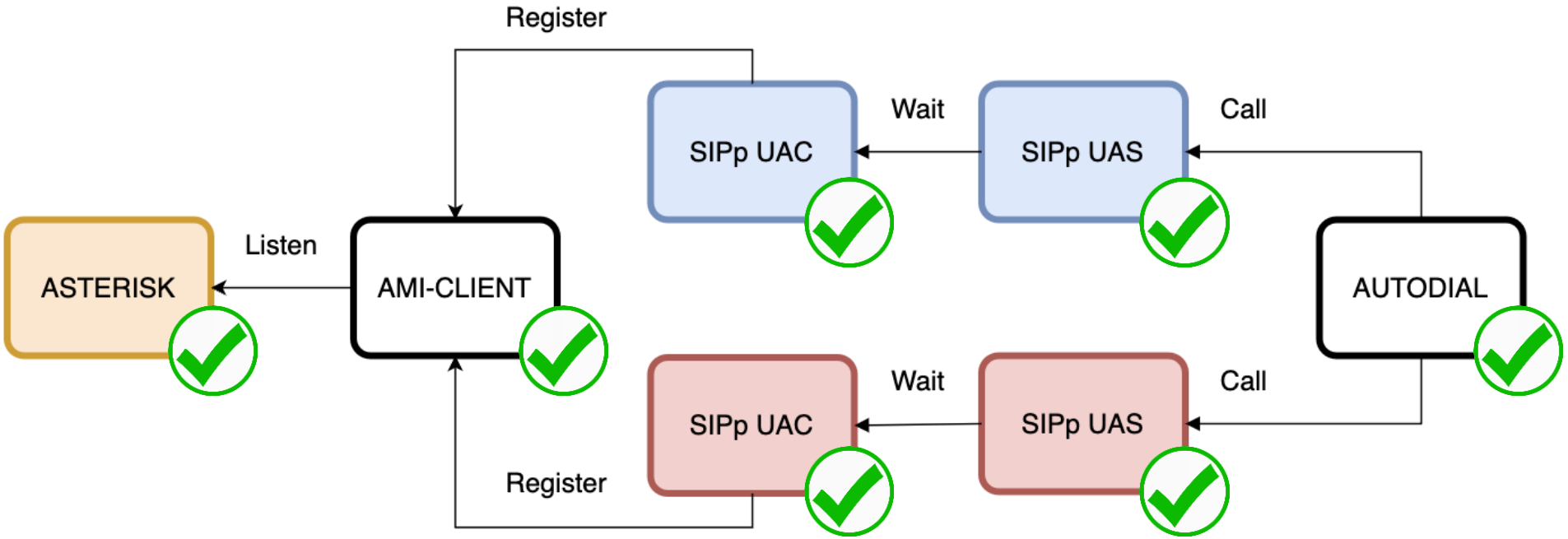
```
_sipssert
├── README.md
├── autodial
│   ├── README.md
│   ├── asterisk
│   │   ├── ari.conf
│   │   ├── extensions.lua
│   │   ├── http.conf
│   │   ├── keys
│   │   ├── logger.conf
│   │   ├── manager.conf
│   │   ├── modules.conf
│   │   └── pjsip.conf
│   ├── config.yml
│   ├── defines.yml
│   └── regular_call
│       ├── README.md
│       ├── asterisk -> ../asterisk
│       ├── scenario.yml
│       └── scripts
└── global.yml
```

Тесты всех проектов в одной репе

Тесты проекта в репе проекта

SIPssert - пишем сценарий

Простой сценарий тестирования звонка в приложении автодозвона



SIPssert - пишем сценарий

Типы Task

	Тип	*Параметры*	*Точка монтирования*	*Образ*
1	opensips	socket, config_file	/etc/opensips	opensips/opensips
2	asterisk	config_files	/home	yaroslavonline/asterisk
3	sipp	username, password, port, keys, calls, duration, proxy, service, scenario	/home	ctaloi/sipp
4	uac-sipp	proxy, remote, caller, destination, config_file	/home	ctaloi/sipp
5	uas-sipp	config_file	/home	ctaloi/sipp
6	opensips-cli	script, mi_ip, mi_port, mi_path, config_file	/home	opensips/opensips-cli

SIPssert - пишем сценарий

Смотри не перепутай!

- Общие настройки - в config.yml
  - настройки сетей
  - настройки типов
- Переменные - в defines.yml
- Таски и зависимости - в scenario.yml
- Конфиги тасков - в директорию сценария

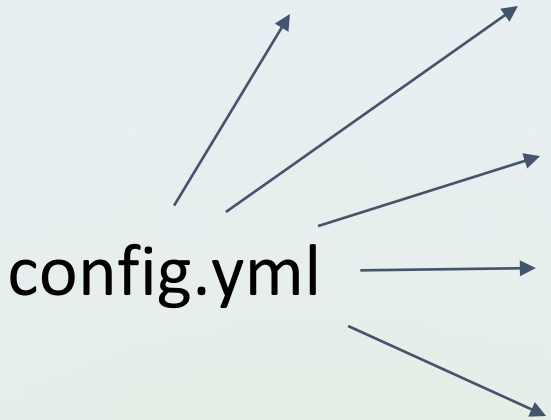
SIPssert - пишем сценарий

Настройки сетей

1

```
bridge_networks:
  - name: osbr0
    subnet: {{ network_range }}
    gateway: {{ network_gateway }}

network: osbr0
```



Настройки типов

2

```
sipp:
  ip: {{ uac_ip }}
  port: {{ uac_port }}
  image: {{ sipp_image }}
  service: '{{ uac_username }}'
  proxy: {{ asterisk_ip }}:{{ asterisk_port }}
  username: '{{ uac_username }}'
  password: {{ uac_password }}
```

3

```
uas-sipp:
  ip: '{{ uas_ip }}'
  port: {{ uas_port }}
  image: {{ sipp_image }}
  service: '{{ uas_username }}'
  proxy: {{ asterisk_ip }}:{{ asterisk_port }}
  username: '{{ uas_username }}'
  password: {{ uas_password }}
```

4

```
ami-client:
  ip: {{ ami_client_ip }}
  image: {{ ami_image }}
  login: '{{ ami_login }}'
  secret: '{{ ami_secret }}'
  asterisk_ip: {{ asterisk_ip }}
```

5

```
asterisk:
  port: {{ asterisk_port }}
  ip: {{ asterisk_ip }}
  image: {{ asterisk_image }}
  stop_timeout: 5
  config_files:
    - "modules.conf:asterisk/modules.conf"
    - "pjsip.conf:asterisk/pjsip.conf"
    - "logger.conf:asterisk/logger.conf"
    - "extensions.lua:asterisk/extensions.lua"
    - "http.conf:asterisk/http.conf"
    - "ari.conf:asterisk/ari.conf"
    - "manager.conf:asterisk/manager.conf"
    - "keys/cert.pem:asterisk/keys/cert.pem"
    - "keys/key.pem:asterisk/keys/key.pem"
  ready:
    wait: {{ asterisk_ready_timeout }}
```

SIPssert - пишем сценарий

Переменные

1

```
---
network_range: 192.168.1.0/24
network_gateway: 192.168.1.254
```

2

```
# operator
uac_ip: 192.168.1.3
uac_port: 5080
uac_username: 9000
uac_password: secret
```

3

```
# trunk
uas_ip: 192.168.1.4
uas_port: 5090
uas_username: 'provider-trunk'
uas_number: '77777777777'
uas_password: secret
```

4

```
# asterisk
asterisk_ip: 192.168.1.5
asterisk_port: 5060
asterisk_ready_timeout: 3
```

5

```
# autodial
autodial_ip: 192.168.1.17

# ami-client
ami_client_ip: 192.168.1.120
ami_login: 'developer'
ami_secret: 'secret'
```

SIPssert - автодозвон

- Запускаем asterisk

1

```
- name: asterisk
  type: asterisk
```

- Запускаем client ami

2

```
- name: ami-client
  type: ami-client
  config_file: scripts/ami.xml
```

- Регистрируемся

3

```
- name: operator register
  type: sipp
  config_file: scripts/sipp-register.xml
```

```
- name: trunk register
  type: uas-sipp
  config_file: scripts/sipp-register.xml
```

- Ждем звонка

4

```
- name: operator
  type: sipp
  config_file: scripts/uas.xml
```

```
- name: trunk
  type: uas-sipp
  config_file: scripts/uas.xml
```

- Звоним

5

```
- name: autodial
  image: {{ autodial_image }}
  ip: {{ autodial_ip }}
  env:
    ASTERISK_HOST: "{{ asterisk_ip }}:8088"
```

scenario.yml

SIPssert - входящий звонок

- Запускаем asterisk

1

```
- name: asterisk
  type: asterisk
```

- Запускаем client ami

2

```
- name: ami-client
  type: ami-client
  config_file: scripts/ami.xml
```

- Регистрируемся

3

```
- name: operator register
  type: sipp
  config_file: scripts/sipp-register.xml
```

```
- name: trunk register
  type: uas-sipp
  config_file: scripts/sipp-register.xml
```

- Звоним

4

```
- name: trunk
  type: uas-sipp
  config_file: scripts/uac.xml
```

- Ждем звонка

5

```
- name: operator
  type: sipp
  config_file: scripts/uas.xml
```

- Обрабатываем звонок

6

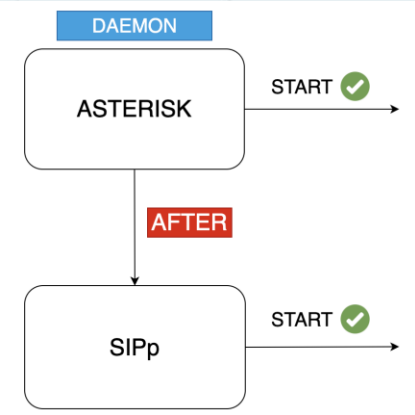
```
- name: autodial
  image: {{ autodial_image }}
  ip: {{ autodial_ip }}
  env:
    ASTERISK_HOST: "{{ asterisk_ip }}:8088"
```

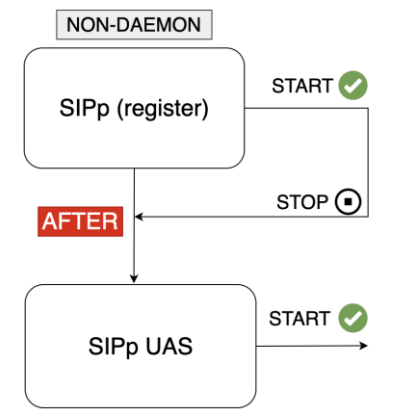
scenario.yml

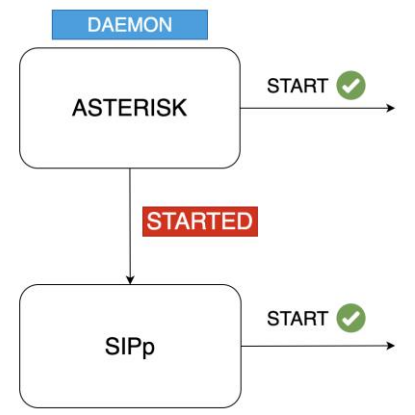
SIPssert - управление зависимостями

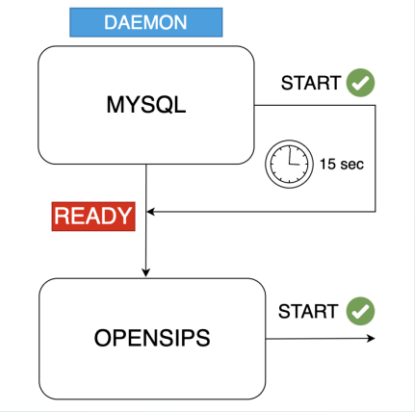
DAEMON
NON-DAEMON

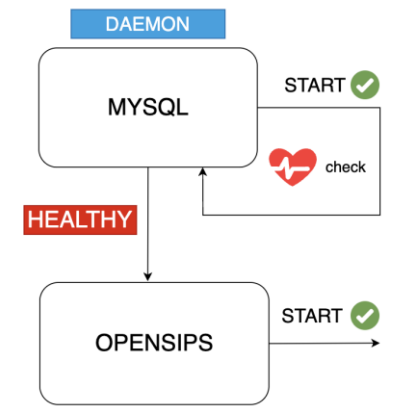
AFTER
STARTED
READY
HEALTHY
DELAY

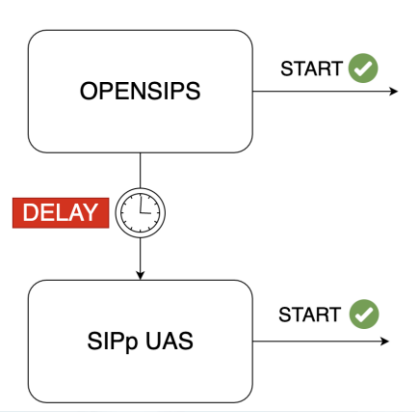












SIPssert - пишем сценарий

- Запускаем asterisk

1

```
- name: asterisk
  type: asterisk
```

- Запускаем client ami

2

```
- name: ami-client
  type: ami-client
  config_file: scripts/ami.xml
  require:
    Ready: asterisk
```

- Регистрируемся

3

```
- name: operator register
  type: sipp
  config_file: scripts/sipp-register.xml
  require:
    Ready: asterisk
```

- Ждем звонка

4

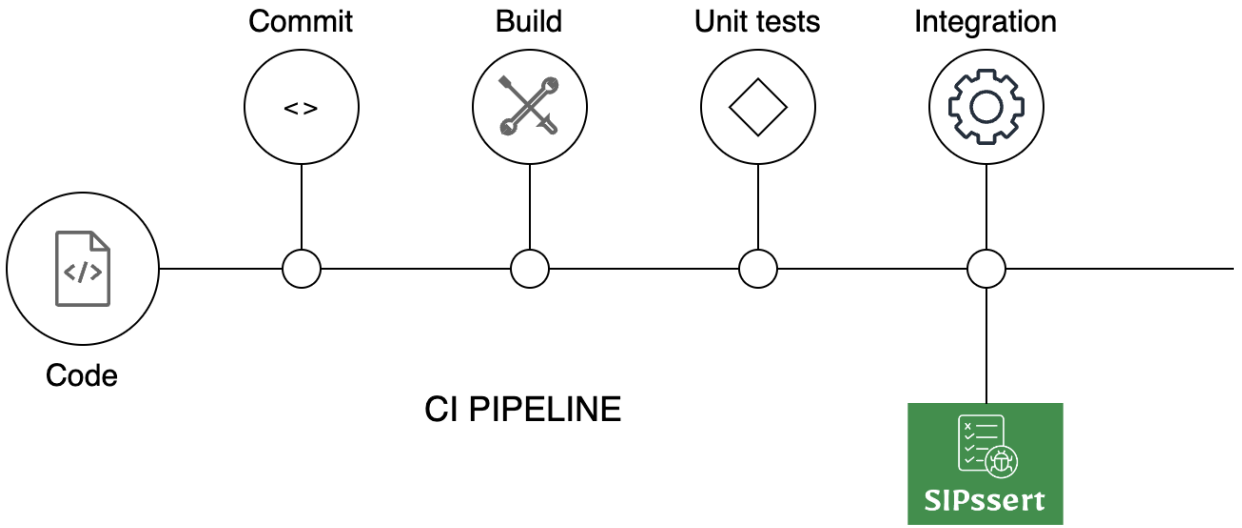
```
- name: operator
  type: sipp
  config_file: scripts/uas.xml
  require:
    After: operator register
```

- Звоним

5

```
- name: autodial
  image: {{ autodial_image }}
  ip: {{ autodial_ip }}
  env:
    ASTERISK_HOST: "{{ asterisk_ip }}:8088"
  require:
    - after:
        task: trunk register
        wait: 2
    - after:
        task: operator register
        wait: 2
```

CI/CD Pipeline - тестирование



- Checkout
- Сборка
- Установка
- Запуск тестов
- Вывод результатов

Github Workflows - action-sipssert

- Простота
- Функционал
- Переиспользование
- Версионность

Параметры	*Требуется*	*По-умолчанию*	*Описание*
test-path	-	_sipssert	Путь к директории с тестами
sipssert-path	-	.sipssert	Путь для установки sipssert
test-set	**!**	-	Название набора тестов
tests	-	-	Список тестов
username	**!**	-	Логин для Docker Registry
password	**!**	-	Пароль для Docker Registry
token	**!**	-	Токен доступа
extra-vars	-	-	Список переменных

Github Workflows - extra-vars

Проблема

1. Как передавать тэг образа?
2. Как менять настройки из воркфлоу?

Решение

- extra-vars !
- ключ=значение

```
sipssert-test:
  name: Sipssert тесты
  runs-on: [self-hosted, medium, Linux]
  needs:
    - build-sipssert
    - config
  steps:
    - uses: actions/checkout@v4
    - uses: telephony/action-sipssert@v2
      with:
        token: ${{ secrets.GITHUB_TOKEN }}
        username: ${{ secrets.DOCKER_REGISTRY_USERNAME }}
        password: ${{ secrets.DOCKER_REGISTRY_PASSWORD }}
        test-set: autodial
        extra-vars: |
          autodial_image=${{ needs.config.outputs.fqin_sipssert }}
          autodial_timeout_seconds=120
```

```
- name: autodial
  image: {{ autodial_image }}
  ip: {{ autodial_ip }}
  env:
    ASTERISK_HOST: "{{ asterisk_ip }}:8088"
```

kolesa group

KAMAILIO CONF'24

SIPssert - расширение фреймворка

Как?	***Когда?***	***Где?***
• Клон	• Проприетарные	• class Task()
• Контрибьют	• Опенсорс	• class Task()
• Generic Task	• Нужно быстро • Нужно просто	• scenario.yml
• asterisk	• sipexer miconda/**sipexer**	• ami-client

SIPssert - Generic Task

Тип generic задан *по-умолчанию*

Компонент Asterisk

```
- name: Asterisk
  daemon: true
  image: asterisk
  port: {{ asterisk_port }}
  ip: {{ asterisk_ip }}
  stop_timeout: 5
  args:
     -C /home/asterisk/asterisk.conf -c -vvvv -g
  ready:
    wait: {{ asterisk_ready_timeout }}
```

Компонент Autodial

```
- name: autodial
  image: {{ autodial_image }}
  ip: {{ autodial_ip }}
  env:
    ASTERISK_HOST: "{{ asterisk_ip }}:8088"
```

Параметры

- image
- mount_point
- daemon
- args

Демонстрация - видео

Итоги

- **Интеграционное тестирование:** средства автоматизации.
- **SIPssert:** фреймворк для тестирования VoIP-компонентов.
- **Приложение autodial:** тестируем базовые кейсы - автодозвон и входящий звонок.
- **Как написать сценарий:** ключевые моменты, на что обратить внимание.
- **Github Workflows:** интеграционное тестирование в CI/CD Pipeline.
- **Github Actions:** action-sipssert для запуска тестов в workflow.
- **Расширение фреймворка:** сборка и интеграционное тестирование.
- **Демонстрация на реальном примере:** сборка сборка и интеграционное тестирование.

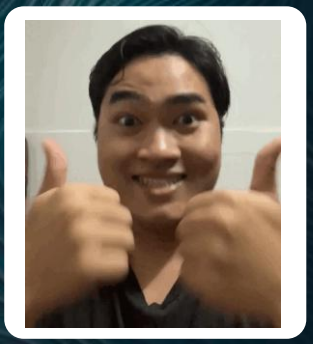
Спасибо за внимание!

Буду рад ответить на все ваши вопросы
сейчас или свяжитесь со мной в будущем:

kolesa group

Ярослав Нападайло

tg://YaroslavOnline



KAMAILIO CONF'24