

DEVOPS/DEVSECOPS. БЕЗОПАСНАЯ РАЗРАБОТКА

ВВЕДЕНИЕ В DEVOPS/DEVSECOPS



АЛЕКСАНДР БАКИН

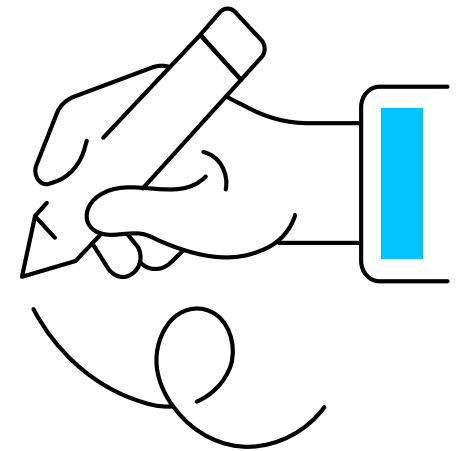
АО «Инфосистемы Джет»

Люблю шоссейный бег и безопасную разработку ПО в любых ее проявлениях.

Отвечаю за консалтинг в направлении безопасной разработки ПО в ЦИБ Джет

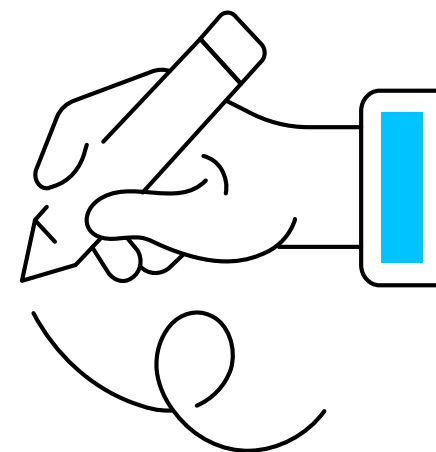
ЦЕЛИ ЗАНЯТИЯ

- ✓ Познакомить слушателей с основной терминологией DevOps/DevSecOps
- ✓ Сделать так, чтобы множество сокращений (SAST, IAST, CS, SM, SCM и т.д.) стали понятными терминами, а не набором букв
- ✓ Показать из чего могут состоять практики DevSecOps, что является важным



ЧЕГО **НЕ** БУДЕТ

- ✓ Конкретных систем защиты, их архитектур и способов внедрения
- ✓ Рекомендаций «Делай 1, 2 и 3» и получится DevSecOps
- ✓ Практики



ОБЩАЯ ИНФОРМАЦИЯ О DEVOPS

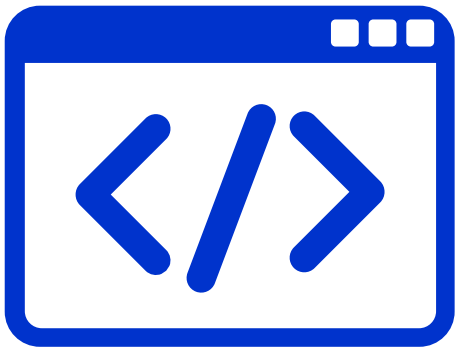
4TO TAKOE DEV(SEC)OPS?

“

It was good and bad not to have a definition [laughs]. People are really struggling with what DevOps is right now

”

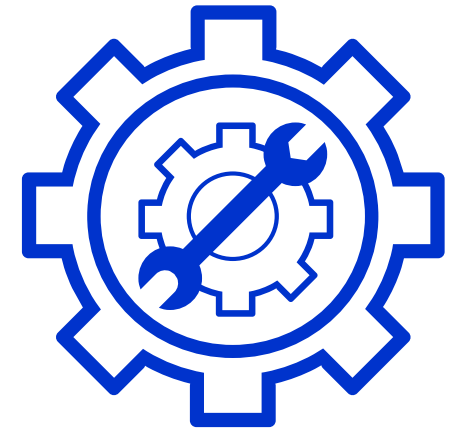
КОНФЛИКТ



РАЗРАБОТКА

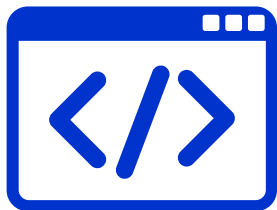


**ИНФОРМАЦИОННАЯ
БЕЗОПАСНОСТЬ**



СОПРОВОЖДЕНИЕ

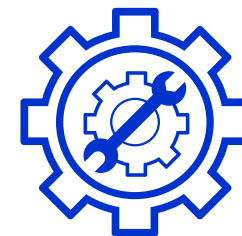
10+ DEPLOYS PER DAY: DEV AND OPS COOPERATION AT FLICKR, 2009



РАЗРАБОТКА

It's not my code, it's your machines!

It's not my machines, it's your code!



СОПРОВОЖДЕНИЕ



ИНФОРМАЦИОННАЯ
БЕЗОПАСНОСТЬ

Печатает...



Time-To- Market



ОТ...

Dev vs Ops

От противостояния...

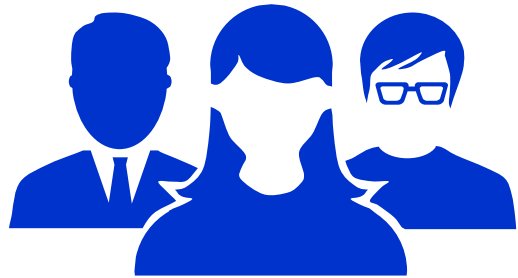


... К

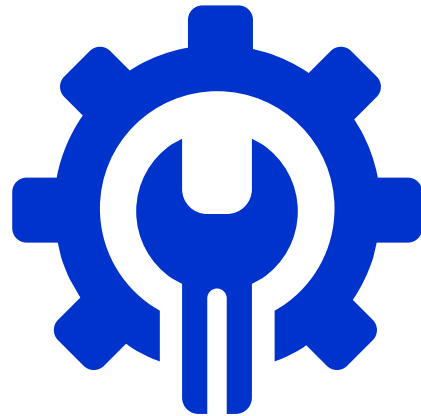
DevOps

... К созиданию!

ПРИНЦИПЫ DEVOPS – CAMS



CULTURE



AUTOMATION



MEASUREMENT



SHARING

DEV(SEC)OPS: ПРИМЕРЫ

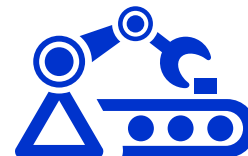
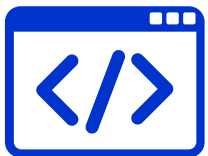
Ключевые характеристики		ELITE	HIGH	MEDIUM	LOW
1	ВРЕМЯ РАЗВОРАЧИВАНИЯ <i>Как часто производит разворачивание приложений в продуктивной среде?</i>	По запросу (несколько раз в день)	От 1 раза в день до 1 раза в неделю	От 1 раза в неделю до 1 раза в месяц	От 1 раза в неделю до 1 раза в месяц
2	ВРЕМЯ НА ИЗМЕНЕНИЕ <i>Время от commit'a кода до его «применения» в продуктивной среде?</i>	Менее дня	От дня до недели	От недели до месяца	От недели до месяца
3	ВРЕМЯ ВОССТАНОВЛЕНИЯ <i>Как быстро будет восстановлена работоспособность сервиса после сбоя?</i>	Менее часа	Менее дня	От дня до недели	От 1 до 6 месяцев
4	ДЕГРАДАЦИЯ СЕРВИСА <i>Какой % изменений приводит к деградации сервиса и/или его временной недоступности?</i>	5%	10%	15%	64%
5	% РЕСПОНДЕНТОВ	18%	31%	33%	17%



Небольшой, но крайне насыщенный на события роман о том, как менялось взаимодействие ИТ, ИБ и разработки в одной, казалось бы, не ИТ-компании

DEVOPS: КЛЮЧЕВЫЕ ТЕХНОЛОГИИ

КЛЮЧЕВАЯ ТЕРМИНОЛОГИЯ



Среда разработки (IDE)

Инструмент для написания исходного кода ПО, включающий, как минимум:

- редактор исходного кода с подсветкой синтаксиса
- отладчик для поиска ошибок, допущенных разработчиком
- Компилятор и/или интерпретатор
- Может поддерживать один или несколько языков
- Упрощает рефакторинг исходного кода

Система контроля версий

Инструмент для организации централизованное хранилище исходного кода. Значимые возможности:

- Возврат кода до рабочего состояния
- Возможность параллельной работы команды над одним проектом
- Контроль внесения изменений в исходный код ПО
- «Слияние» нескольких доработок в единый проект

Непрерывная интеграция и доставка (CI/CD)

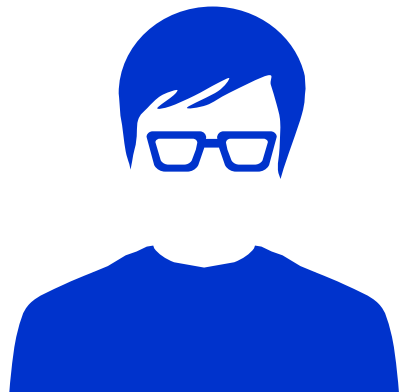
Решение для автоматизации двух основных задач:

- Continuous Integration (CI):
 - частое внесение небольших изменений в модули ПО (units)
 - отслеживание работоспособности ПО, собранного из модулей
- Continuous Delivery (CD):
 - подготовка необходимой инфраструктуры (web-серверы, БД и т.д.)
 - «разворачивание» ПО, собранного на стадии CI

Реестр артефактов

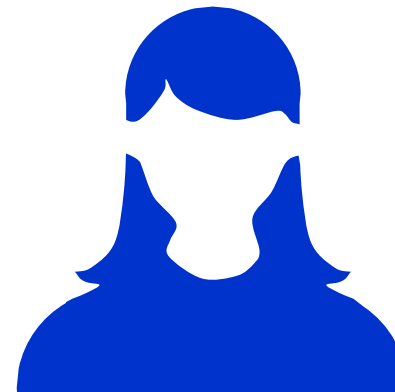
Инструмент для централизованного хранилища артефактов:

- Open source компоненты (библиотеки, зависимости)
- Результаты сборки ПО (например, jar файлы)
- Образы контейнеров



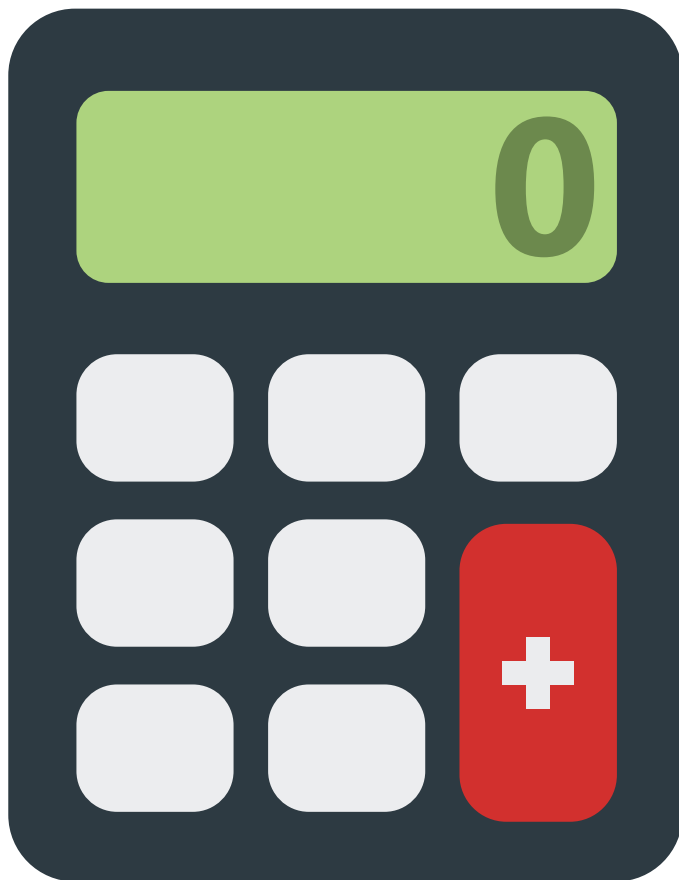
Разработчик ПО

Пишет исходный код, основная задача которого – реализация функциональных возможностей программного обеспечения



Специалист по тестированию

Тестирует приложение и/или разрабатывает автоматические тесты, которые позволяют проверить корректность работоспособности ПО (функций), отображения информации, поведения ПО в условиях высоких нагрузок и т.д.

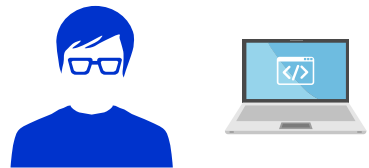


Разработка калькулятора

РАЗРАБОТКА КАЛЬКУЛЯТОРА: ФУНКЦИОНАЛЬНЫЕ МОДУЛИ

СЛОЖЕНИЕ

```
Calc.py x
1 def sum_dev(a,b):
2   return a+b
```



Никита

ВЫЧИТАНИЕ

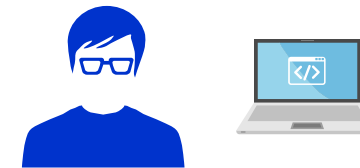
```
Calc.py x
1 def sub_dev(a,b):
2   return a-b
```



Андрей

УМНОЖЕНИЕ

```
Calc.py x
1 def mul_dev(a,b):
2   return a*b
```



Филипп

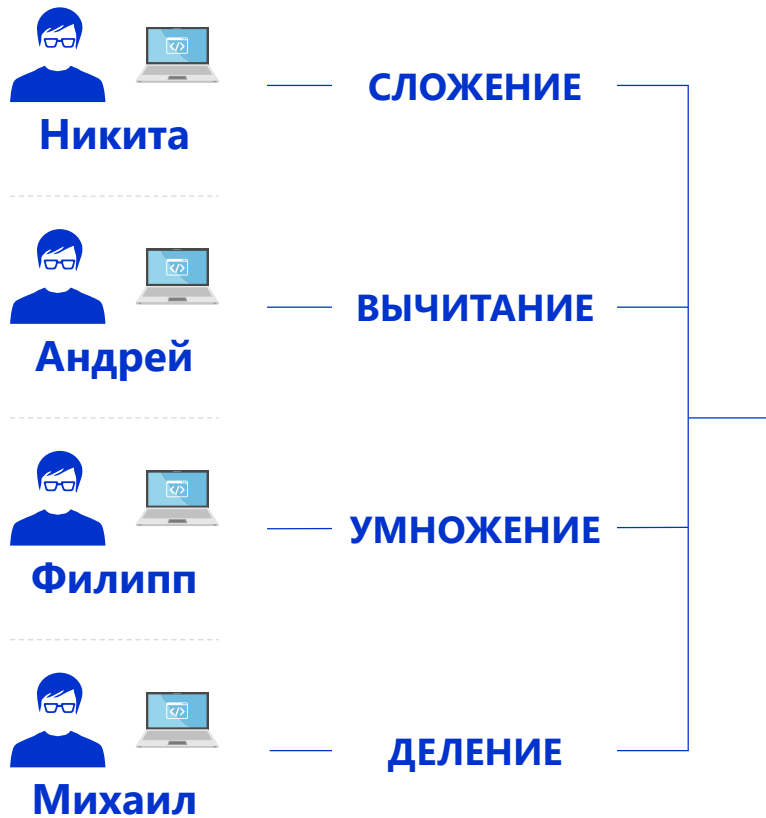
ДЕЛЕНИЕ

```
Calc.py x
1 def div_dev(a,b):
2   return a/b
```



Михаил

РАЗРАБОТКА КАЛЬКУЛЯТОРА: ИНТЕГРАЦИЯ



```
Calc.py x
1  def sum_dev(a,b):
2  return a+b
3
4  def sub_dev(a,b):
5  return a-b
6
7  def mul_dev(a,b):
8  return a*b
9
10 def div_dev(a,b):
11 return a/b
```

«Калькулятор» собран, теперь
необходимо протестировать, что
все работает корректно

РАЗРАБОТКА КАЛЬКУЛЯТОРА: ТЕСТИРОВАНИЕ



Екатерина

```
Calc_Test.py x
1  import pytest
2  import Calc
3
4  def test_sum():
5      assert Calc.sum_dev(2,4) == 6
6
7  def test_sub():
8      assert Calc.sub_dev(6,4) == 2
9
10 def test_mul():
11     assert Calc.mul_dev(10,10) == 100
12
13 def test_div():
14     assert Calc.div_dev(84,2) == 42
```

Успешное прохождение всех тестов! Можно двигаться дальше!

```
===== test session starts =====
platform win32 -- Python 3.8.1, pytest-3.0.6, py-1.8.1, pluggy-0.4.0
rootdir: C:\Users\ay.gavrilov\PycharmProjects\PrismaAPIUsage, inifile:
collected 4 items
```

Calc_Test.py

```
===== 4 passed in 0.10 seconds =====
```

Функциональный модуль СУММИРОВАНИЯ содержит ошибки.
Постановка задачи на исправление (заведение bug'a)

```
===== FAILURES =====
test_sum
```

```
def test_sum():
> assert Calc.sum_dev(2,4) == 6
E assert -2 == 6
E + where -2 = <function sum_dev at 0x037F24A8>(2, 4)
E + where <function sum_dev at 0x037F24A8> = Calc.sum_dev
```

Calc_Test.py:5: AssertionError

```
===== 1 failed, 3 passed in 0.25 seconds =====
```


РАЗРАБОТКА КАЛЬКУЛЯТОРА: ПОМЕЩЕНИЕ АРТЕФАКТА В РЕЕСТР, РАЗВОРАЧИВАНИЕ



ЖИЗНЕННЫЙ ЦИКЛ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ

Нам нужен калькулятор с функциями:

- Сложение
- Вычитание
- Деление
- Умножение



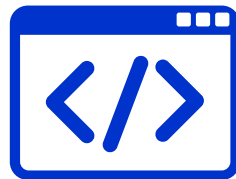
Сбор требований

Определяем подходящий стек технологий, архитектуру ПО, ставим задачи на разработку



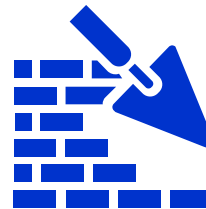
Проектирование

Пишем исходный код функциональных модулей



Разработка

Периодически «собираем ПО» для того, чтобы проверить как модули взаимодействуют друг с другом



Сборка

Проводим различные тесты:

- Функциональные
- Нагрузочные и т.д.



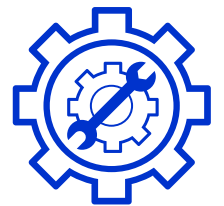
Тестирование

«Запускаем» ПО для эксплуатации пользователями



Развертывание

Решаем проблемы, которые возникают в ходе эксплуатации



Эксплуатация

ЖИЗНЕННЫЙ ЦИКЛ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ: АВТОМАТИЗАЦИЯ

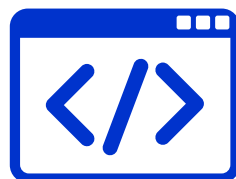
Автоматизация этапов при помощи **Continuous Integration/Continuous Deployment (CI/CD)**



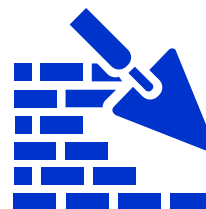
Сбор требований



Проектирование



Разработка



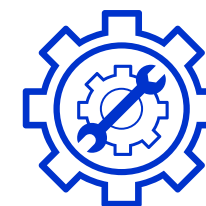
Сборка



Тестирование

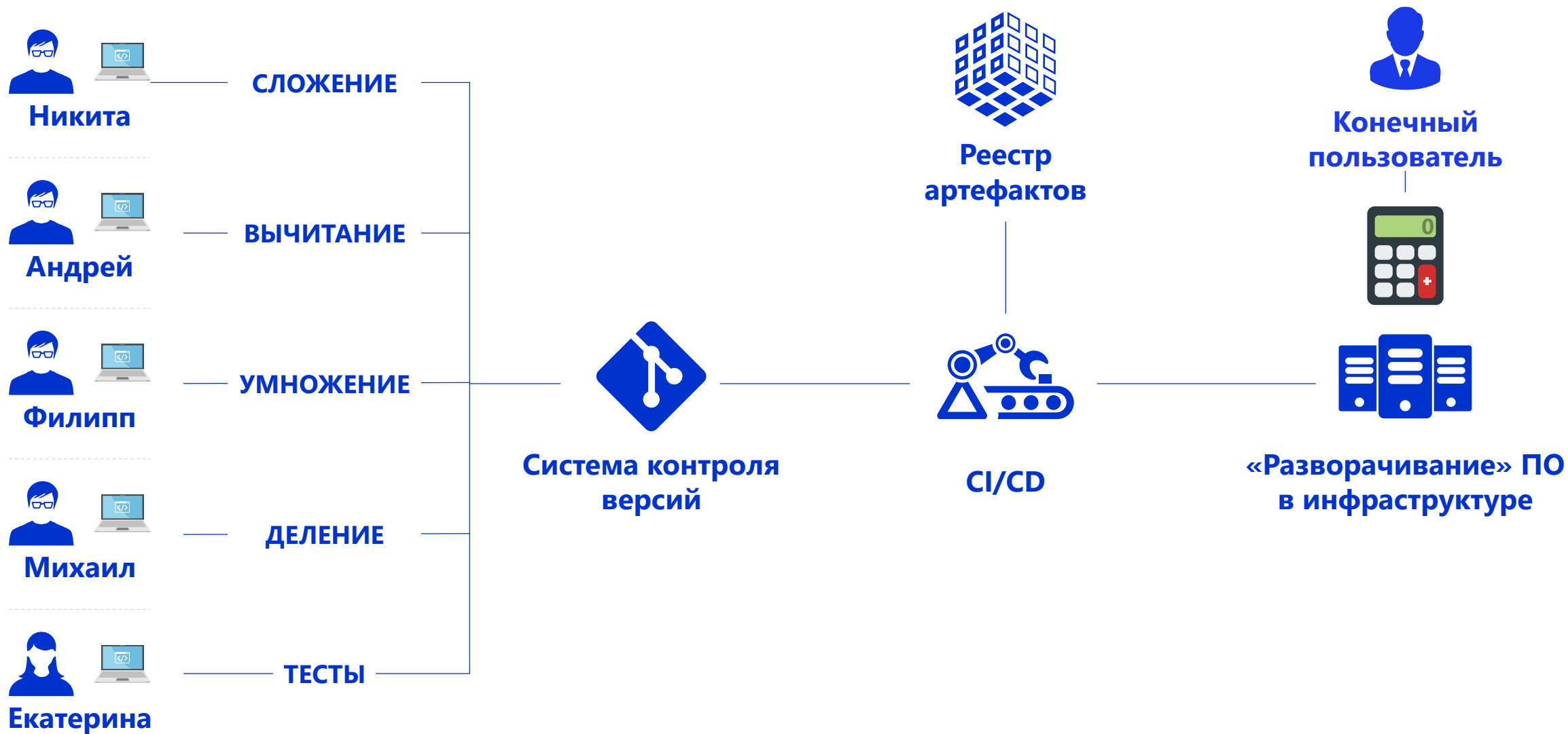


Развертывание



Эксплуатация

РАЗРАБОТКА КАЛЬКУЛЯТОРА: АВТОМАТИЗАЦИЯ



КАК ЭТО ВЫГЛЯДИТ?

Write Preview changes

master .gitlab-ci.yml .gitlab-ci.yml Apply a template

```
1 variables:
2   COMMIT: "0.19"
3   TEST_COMMIT: "0.24"
4
5 stages:
6   - pre-build checks
7   - build
8   - test
9   - deploy
10
11 build-image:
12   stage: build
13   tags:
14     - fvm233
15   script:
16     - docker login -u $CI_REGISTRY_USER -p $CI_REGISTRY_PASSWORD $CI_REGISTRY
17     - docker build -t $CI_REGISTRY_IMAGE:$COMMIT .
18     - docker push $CI_REGISTRY_IMAGE:$COMMIT
19
20 deploy:
21   stage: deploy
22   tags:
23     - fvm233
24   script:
25     - oc login --server=https://"$SERVER" -u "$OC_LOGIN" -p "$OC_PASS"
26     - oc project seminar-demo || true
27     - oc status
28     - oc create secret docker-registry gitlab-registry --docker-username=$OC_GITLAB_LOGIN --docker-password=$OC_GITLAB_PW --docker-server="$GITSERVER" || true
29     - oc create secret docker-registry gitlab --docker-username=$OC_GITLAB_LOGIN --docker-password=$OC_GITLAB_PW --docker-server="$GITSERVER" || true
30     - oc secrets link default gitlab gitlab-registry --for=pull || true
31     - oc apply -f openshift/
32   only:
33     refs:
34       - master
35
```


«УПАКОВКА» ПРИЛОЖЕНИЯ: КОНТЕЙНЕРЫ

```
FROM python:alpine3.14

COPY /app .

RUN pip install -r
requirements.txt

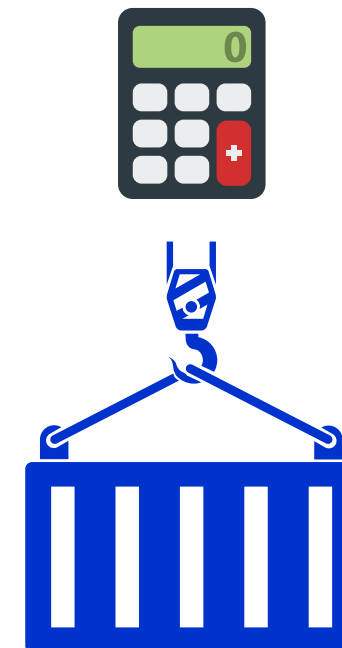
COPY . /code/

EXPOSE 8888
CMD ["python", "./app/app.py"]
```

DOCKERFILE



ОБРАЗ КОНТЕЙНЕРА (IMAGE)

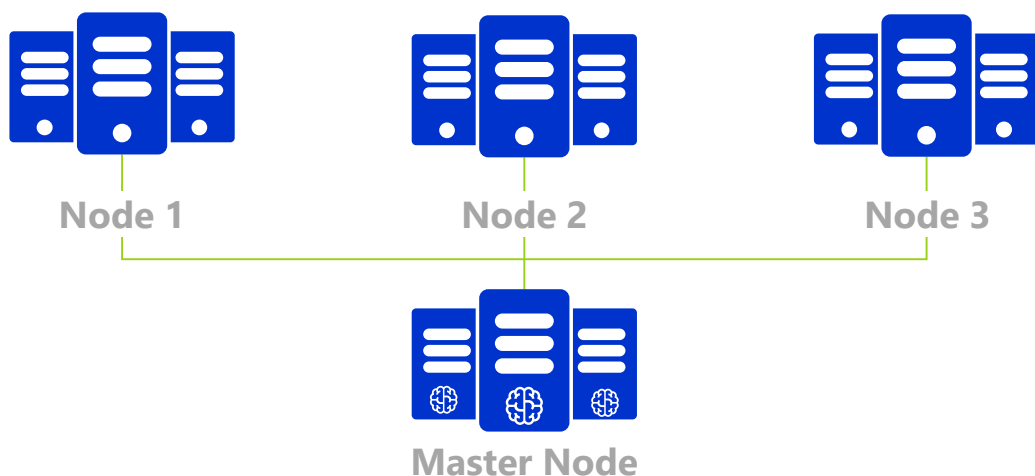


КОНТЕЙНЕР

ЧТО ТАКОЕ КОНТЕЙНЕРНАЯ ОРКЕСТРАЦИЯ?

Оркестратор

Класс решений, который позволяет автоматизировать управление жизненным циклом контейнеров. Особенно удобны при большом количестве контейнеров

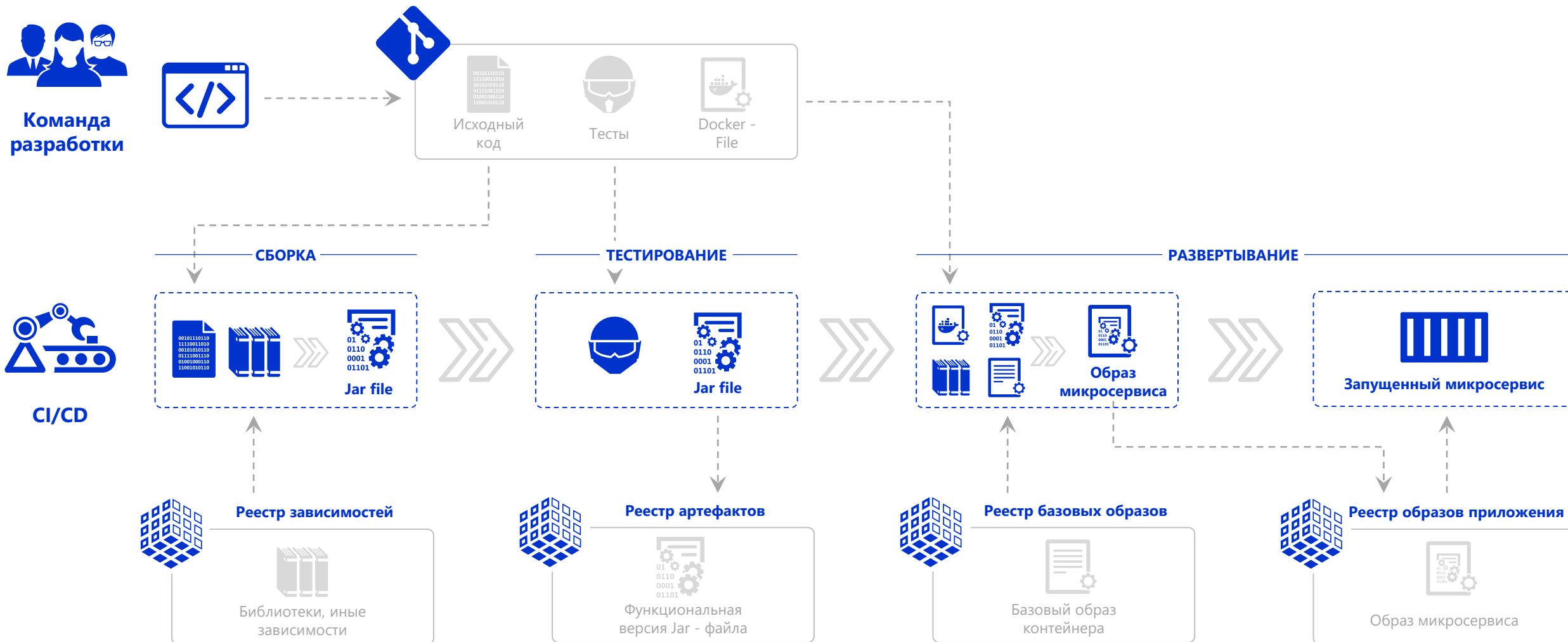


Какие задачи он решает?

- Управление deployment контейнерных приложений
- Помогает в организации сетевого взаимодействия
- Помогает при масштабировании (управлении количеством) контейнеров
- Распределяет ресурсы между контейнерами
- Несет ответственность за балансировку нагрузки между контейнерами
- Управляет перемещением контейнеров между nodes
- ...

DEVSEC: ВОЗМОЖНОСТИ ИБ

ЧТО МОЖЕТ ПОЙТИ НЕ ТАК?



ЧТО МОЖЕТ ПОЙТИ НЕ ТАК?



КОМПОНЕНТНЫЙ АНАЛИЗ ИСХОДНОГО КОДА

SCA

*Software Composition
Analysis*

ЧТО АНАЛИЗИРУЕТ?

Сторонние зависимости разрабатываемого ПО

ПРИНЦИП РАБОТЫ (ОБОБЩЕННО)

Собирает информацию по hash значениям зависимостей, анализирует и предоставляет результат по уязвимостям и лицензиям



ПРЕИМУЩЕСТВА

- Относительно небольшое количество false-positive
- Позволяет анализировать транзитивные зависимости (зависимости зависимостей)
- Может быть использовано при формировании Software Bill of Materials, SBOM
- Простота интерпретации результатов для устранения недостатков



НЕДОСТАТКИ

- Не все идентифицированные уязвимости могут существовать на самом деле
- Для работоспособности скорее всего потребуется доступ в интернет
- Процесс разрешения зависимостей для повышения уровня ИБ может быть не тривиальным

СТАТИЧЕСКИЙ АНАЛИЗ ИСХОДНОГО КОДА

SAST

*Static Application
Security Testing*

ЧТО АНАЛИЗИРУЕТ?

Исходный код приложения

ПРИНЦИП РАБОТЫ (ОБОБЩЕННО)

Построение абстрактного синтаксического дерева (AST) с последующими запросами к нему и анализом результатов



ПРЕИМУЩЕСТВА

- Технология, давно существующая на рынке
- Большое количество решений, из которых можно выбрать
- Несколько видов анализа – от поиска по тексту до taint-анализа
- Может применяться к разным «аспектам» исходного кода, в том числе к анализу конфигураций (linting)
- Позволяет идентифицировать строки в исходном коде, которые являются причиной уязвимости



НЕДОСТАТКИ

- Большое количество false-positive по умолчанию
- Сложность интерпретации результатов работы
- Некоторые решения не могут анализировать НЕ собранный артефакт
- Может занимать продолжительное время (зависит от размеров кодовой базы и сложности кода)
- Не предоставляет информации о возможности эксплуатации уязвимости

ДИНАМИЧЕСКИЙ АНАЛИЗ ИСХОДНОГО КОДА

DAST

*Dynamic Application
Security Testing*

ЧТО АНАЛИЗИРУЕТ?

Запущенное приложение

ПРИНЦИП РАБОТЫ (ОБОБЩЕННО)

Отправка видоизмененных запросов к приложению, анализ успешности реализуемых действий



ПРЕИМУЩЕСТВА

- Меньшее количество false-positive в сравнении с SAST
- Интерпретировать результаты проще, чем в случае с SAST
- Может использоваться вместе с WAF для реализации виртуального патчинга
- Не требуется исходный код для проведения анализа безопасности ПО
- Позволяет идентифицировать, в том числе, некорректные настройки приложения
- Предоставляет информацию о возможности эксплуатации уязвимостей



НЕДОСТАТКИ

- Не указывает место в исходном коде, которое является причиной уязвимости
- Обладает «слепыми пятнами» – может распознать не все формы на web-ресурсе
- Полноценный DAST-анализ может занимать продолжительное время

ИНТЕРАКТИВНЫЙ АНАЛИЗ ИСХОДНОГО КОДА

IAST

*Interactive Application
Security Testing*

ЧТО АНАЛИЗИРУЕТ?

Исходный код и запущенное приложение

ПРИНЦИП РАБОТЫ (ОБОБЩЕННО)

Инструментация – встраивание «сенсоров» внутрь приложения для сбора информации о его работе и последующего анализа



ПРЕИМУЩЕСТВА

- False-positive практически отсутствуют
- Позволяет идентифицировать строки кода приложения, которые являются уязвимыми
- Быстрое предоставление результатов, как правило в режиме около-реального времени
- Предоставляет информацию о возможности эксплуатации уязвимостей



НЕДОСТАТКИ

- Многие решения являются «пассивными», им необходимы данные для анализа (например, тесты, автотесты)
- Сильные ограничения по технологическому стеку
- Используемый подход к тестированию встречается не часто

Secret Finding

ЧТО АНАЛИЗИРУЕТ?

Исходный код

ПРИНЦИП РАБОТЫ (ОБОБЩЕННО)

Идентифицирует секреты на основе pattern-matching и/или с использованием энтропии



ПРЕИМУЩЕСТВА
















- Небольшое количество ложных срабатываний
- Возможность сканирования отдельных commits, branches и т.д.



НЕДОСТАТКИ

- Небольшое количество доступных решений
- Как правило, отсутствуют графические интерфейсы, но есть исключения

СВОДНАЯ ИНФОРМАЦИЯ

ТИП АНАЛИЗА		ОБЪЕКТ ОЦЕНКИ			ЗНАЧИМЫЕ КРИТЕРИИ	
		Исходный код	Зависимости	Запущенное приложение	False positive	Скорость
1	SCA					
2	SAST					
3	DAST					
4	IAST					
5	Secret Finding					

ОБОБЩЕННЫЕ РЕКОМЕНДАЦИИ ПРИ ВЫБОРЕ ЦЕЛЕВЫХ РЕШЕНИЙ

1

Обладать возможностью доступа по API и/или Command Line (для встраивания в CI/CD pipeline)

2

Время осуществления проверки не должно превышать 15 минут. Инкрементальное сканирование (анализ изменений) будет плюсом

3

Возможность предоставления в виде контейнеров в значительной степени упрощает возможность использования

4

Минимальные лицензионные ограничения (например, на параллельное выполнение задач)

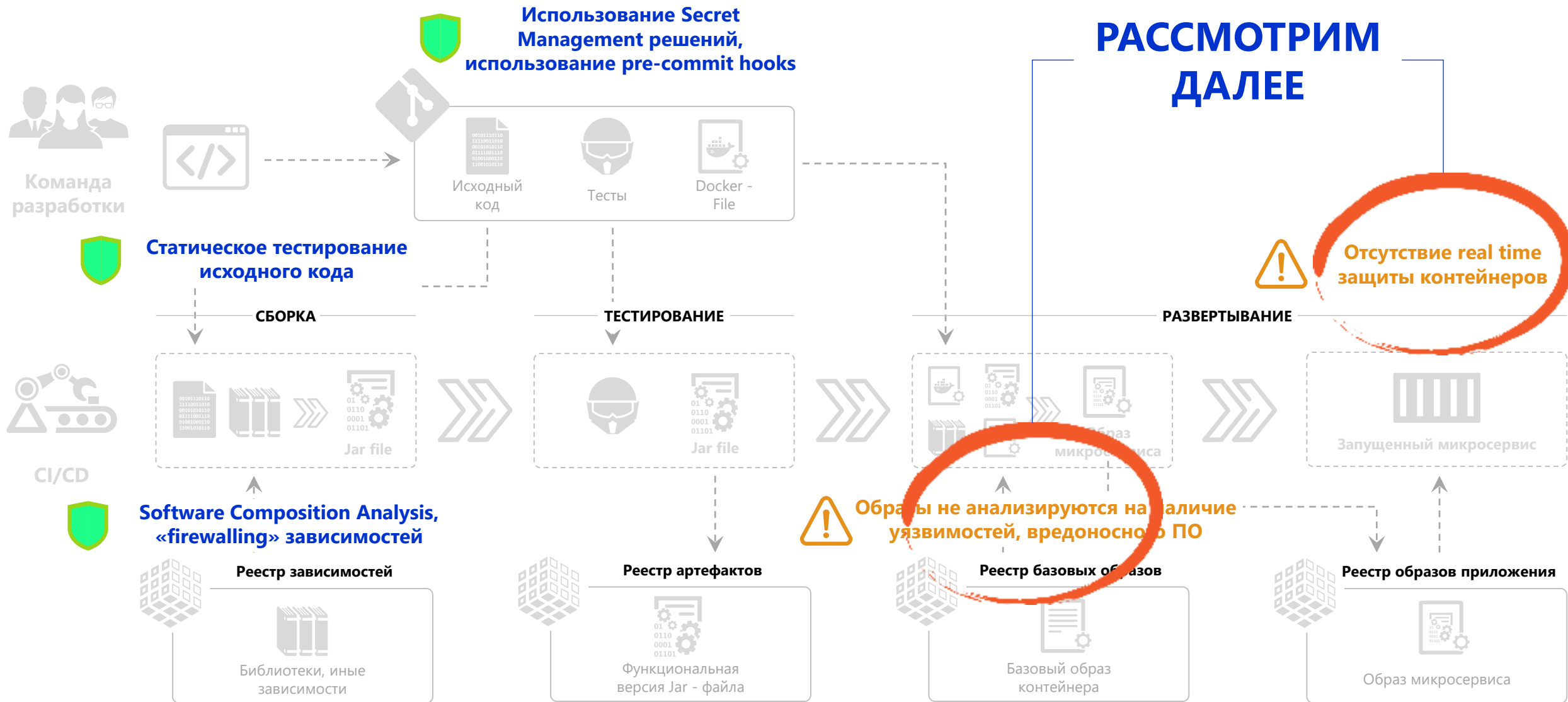
5

Легко «разбираемый» результат (parsing), например, xml и/или json

6

Должна быть реализована возможность подсчета false positive / false negative

ЧТО МОЖЕТ ПОЛУЧИТЬСЯ В ИТОГЕ?



SECOPS : ВОЗМОЖНОСТИ ИБ

ЧТО НУЖНО ЗАЩИЩАТЬ



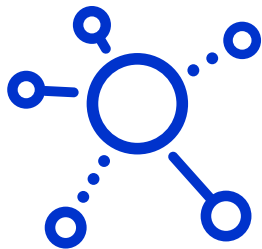
ДОСТУП

Control plane

Node

Контейнеры

...



СЕТЬ

North-South

East-West

Аутентификация

...



КОНТЕЙНЕР

Образы контейнеров

Запуск контейнеров

Возможности контейнера

...



ХОСТ

Сетевой доступ

Уязвимости, ошибки в конфигурации

Контроль запускаемого ПО на хосте

...

ВЫВОД

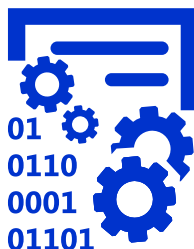
Необходимо **обеспечивать защиту всех уровней «абстракции»**, предлагаемой технологией среды контейнеризации

КОГДА НАДО ЗАЩИЩАТЬ



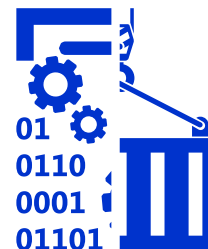
СБОР ТРЕБОВАНИЙ

- Учет требований ИБ



СОЗДАНИЕ ОБРАЗА

- Соответствие рекомендациям ИБ по написанию образа
- Минимально допустимое количество уязвимостей
- Отсутствие вредоносного ПО в образе
- Отсутствие hardcoded секретов
- ...



ЗАПУСК КОНТЕЙНЕРА

- Запуск контейнеров из образов:
 - Из доверенных реестров
 - Не являющихся привилегированными
 - Не содержащих «избыточные» Linux Capabilities
 - Не содержащих hostPathMount и иные недопустимые параметры конфигурации по ИБ
 - ...



РАБОТА КОНТЕЙНЕРА

- Контроль запускаемого ПО внутри контейнера
- Контроль сетевых соединений
- Контроль использования файловых ресурсов
- Контроль потребляемых ресурсов node
- Возможность блокировки нежелательных действий
- Идентификация событий ИБ и инцидентов ИБ
- ...

КАК МОЖНО ЗАЩИЩАТЬ

**ШТАТНЫЕ ИБ
МЕХАНИЗМЫ K8S**

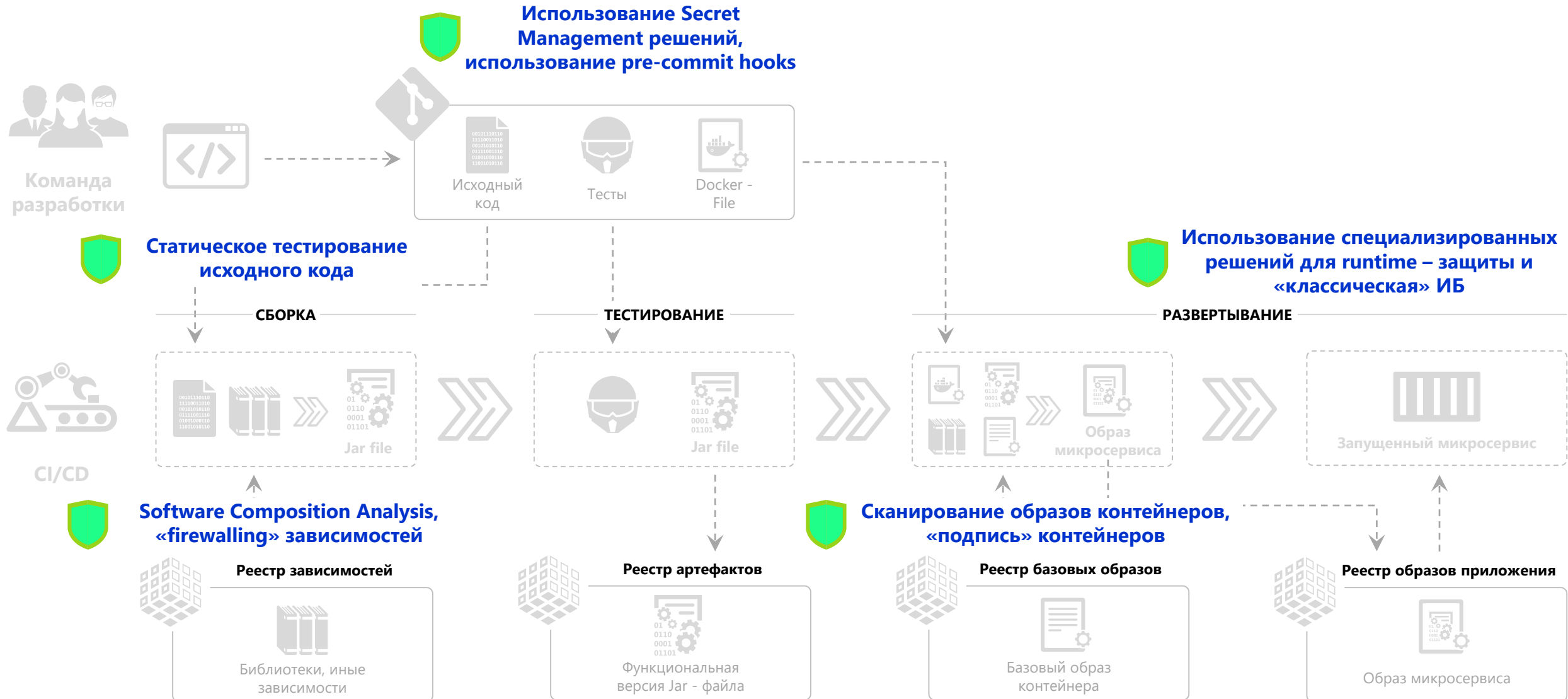


**КОМБИНИРОВАННЫЙ
ПОДХОД**



**НАВЕСНЫЕ
СРЕДСТВА ЗАЩИТЫ**

ЧТО МОЖЕТ ПОЛУЧИТЬСЯ В ИТОГЕ



DEVSECOPS

ЧТО У НАС УЖЕ ЕСТЬ

DevOps



Среда разработки
(IDE)



Система контроля
версий



Непрерывная интеграция
и доставка (CI/CD)



Контейнеры



Реестра артефактов

Sec

Software Composition Analysis, **SCA**

Static Application Security Testing,
SAST

Dynamic Application Security
Testing, **DAST**

Interactive Application Security
Testing, **IAST**

Secret Finding

Container Security

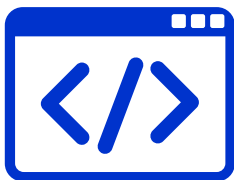
ГДЕ ЭТО МОЖНО ИСПОЛЬЗОВАТЬ



Сбор требований



Проектирование



Разработка



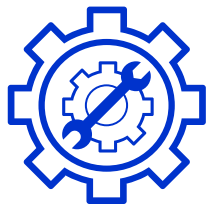
Сборка



Тестирование



Развертывание



Эксплуатация

SCA

SAST

DAST



IAST



Secret Finding



Container Security

Container Security



Pre-build checks



 sast 

 sca 



 secretfinder 



Build



 build-image 

 imagescan 

Test

 dast 

 testdeploy 

 tests 

Сегодня мы поговорили о...

- Что такое DevOps, какие были предпосылки его создания/развития и чем он может быть полезен компаниям
- Инструменты, используемы при разработке ПО – какие они могут быть и зачем нужны
- Как можно соединить несколько решений вместе для автоматизации процесса разработки ПО
- Какие могут быть проблемы информационной безопасности, на каких этапах они могут проявляться
- Как можно анализировать приложение для идентификации ИБ-недостатков, какие бывают способы и когда их можно/целесообразно применять
- Что такое контейнеры, каким угрозам подвержены оркестраторы и какие функциональные возможности требуются для их защиты
- Как создать единый конвейер – безопасная автоматизация процесса разработки ПО, начало DevSecOps

О ЧЕМ МЫ СЕГОДНЯ НЕ ГОВОРИЛИ

Сегодня мы не затронули...

- Управление требованиями ИБ в DevSecOps
- Моделирование угроз и анализ рисков ИБ
- Обучение разработчиков, использование стандартов безопасной разработки
- Управление секретами при помощи Secret Management решений
- Bug bounty программы
- Процессы, которые необходимо создавать и развивать при внедрении DevSecOps
- И многое другое!

СПАСИБО ЗА
ВНИМАНИЕ !