

# Оркестрация AppSec-сканирований с использованием Kubernetes

**Алена Жилина**

---

DevSecOps-инженер

# whoami

- В инфобезе более 5 лет, 3 из которых в качестве DevSecOps-инженера
- Близко знакома с Kubernetes с версии 1.16
- Аудиты на соответствие PCI DSS
- Проектная деятельность в качестве Data-инженера, разработчика, System-дизайнера
- Спикер DevOpsConf, SafeCode, CyberCamp
- Ищу пути решения нестандартных кейсов или создаю новые
- Люблю Союзмультфильм 🐝



# О чём поговорим

1. Оркестрация AppSec-сканирований
  2. ShiftLeft
  3. SecurityQualityGate
- 
4. Подходы к технической реализации через Kubernetes-оператор

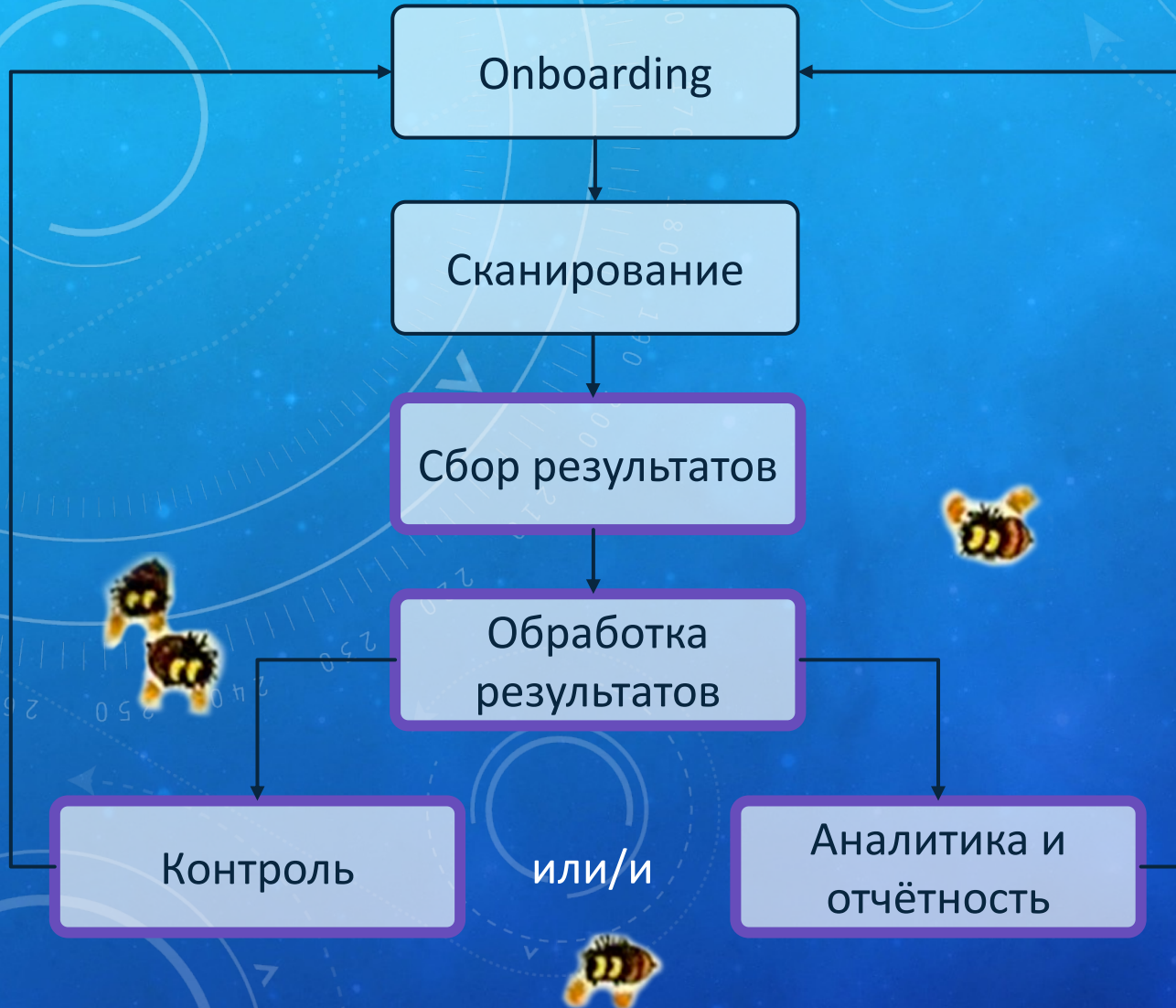


# Оркестрация сканирований



- ✓ есть пайплайн с несколькими инструментами
- есть ASOC-платформа
- нет ничего из этого

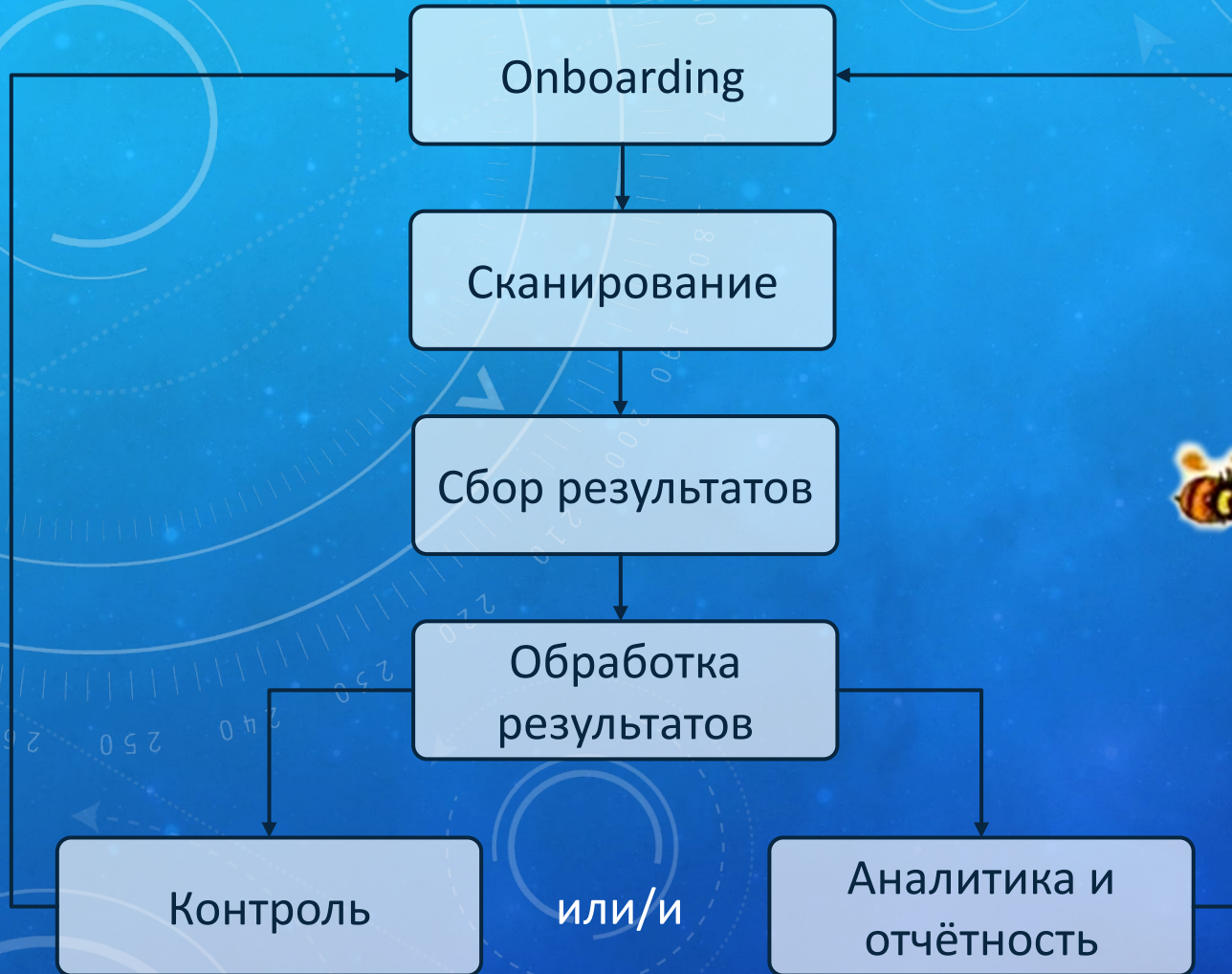
# Оркестрация сканирований



- есть пайплайн с несколькими инструментами
- ✓ есть ASOC\*-платформа
- нет ничего из этого

\* - *Application Security Orchestration and Correlation*

# Оркестрация сканирований

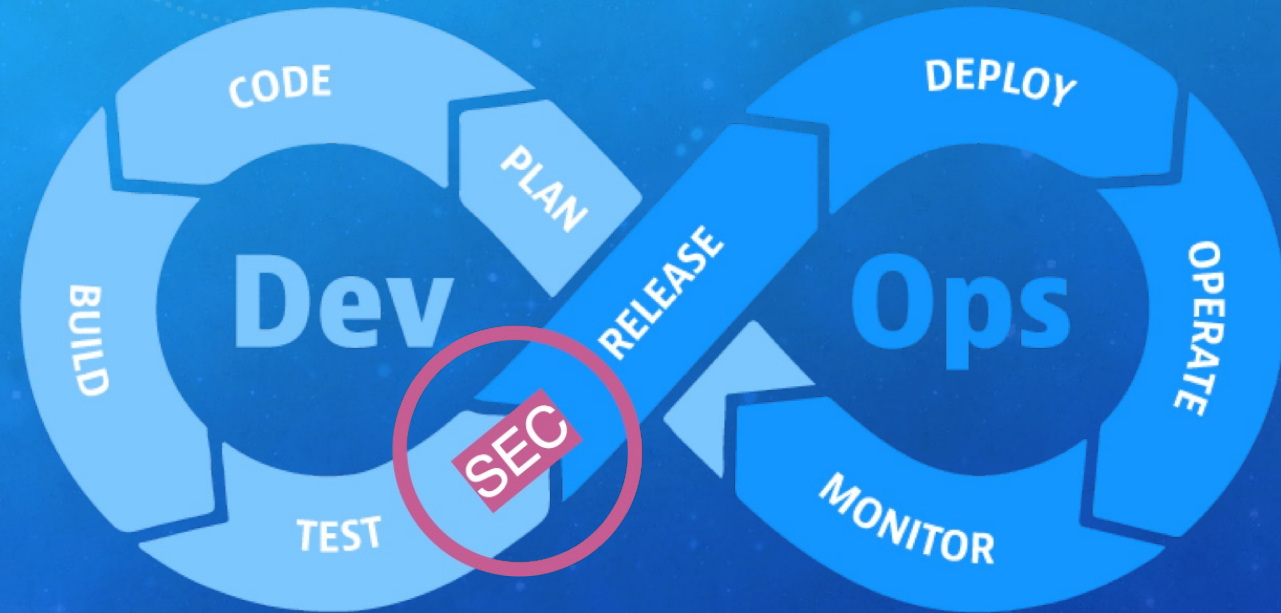


- есть пайплайн с несколькими инструментами
- есть ASOC-платформа
- ✓ нет ничего из этого

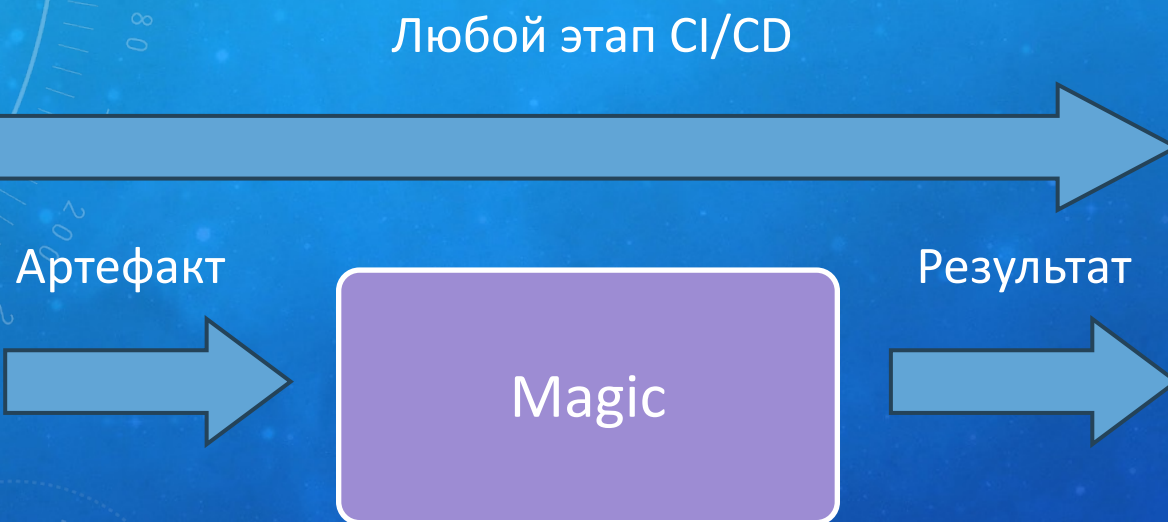


# ShiftLeft

- нехватка кадров
- низкая мотивация к безопасной разработке
- отсутствие автоматизации проверок и технических ресурсов



# SecurityQualityGate



# Для чего оркестрировать сканирования?

Оркестрация – согласованное выполнение всех стадий жизненного цикла сканирования.

Чтобы:

- ✓ снизить количество ручных действий и степень вовлечённости разных ролей
- ✓ увеличить скорость проверок
- ✓ повысить качество проверок



# При чём тут Kubernetes?



# AppSecTool

Хранит конфигурации инструментов

Минимально и достаточно:

...→ что может сканировать

- где расположен

Кастомизация:

- необходимые ресурсы
- доп информация о конфигурации
- типизация артефакта
- тегирование

```
apiVersion: cache.appsecauto.ru/v1alpha1
kind: AppSecTool
metadata:
  name: semgrep-sast
spec:
  name: semgrep-sast
  toolConfiguration:
    - state: inJob
      tag: "python-basic"
      artifactType: "code"
      languages: ["Python"]
      accuracy: "0.9"
      image: "semgrep/semgrep:1.79-nonroot"
      args: ["semgrep", "scan", "/workspace/repo", "--json"]
      reportPath: "/workspace/semgrep/report.json"
      cpuRequest: "150m"
      memoryRequest: "250Mi"
```

# AppSecTool

Хранит конфигурации инструментов

Минимально и достаточно:

- что может сканировать
- где расположен

Кастомизация:

- необходимые ресурсы
- доп информация о конфигурации
- типизация артефакта
- тегирование

```
apiVersion: cache.appsecauto.ru/v1alpha1
kind: AppSecTool
metadata:
  name: semgrep-sast
spec:
  name: semgrep-sast
  toolConfiguration:
    - state: inJob
      tag: "python-basic"
      artifactType: "code"
      languages: ["Python"]
      accuracy: "0.9"
      image: "semgrep/semgrep:1.79-nonroot"
      args: ["semgrep", "scan", "/workspace/repo", "--json"]
      reportPath: "/workspace/semgrep/report.json"
      cpuRequest: "150m"
      memoryRequest: "250Mi"
```

# AppSecTool

Хранит конфигурации инструментов

Минимально и достаточно:

- что может сканировать
- где расположен

Кастомизация:

...> необходимые ресурсы

- доп информация о конфигурации
- типизация артефакта
- тегирование

```
apiVersion: cache.appsecauto.ru/v1alpha1
kind: AppSecTool
metadata:
  name: semgrep-sast
spec:
  name: semgrep-sast
  toolConfiguration:
    - state: inJob
      tag: "python-basic"
      artifactType: "code"
      languages: ["Python"]
      accuracy: "0.9"
      image: "semgrep/semgrep:1.79-nonroot"
      args: ["semgrep", "scan", "/workspace/repo", "--json"]
      reportPath: "/workspace/semgrep/report.json"
      cpuRequest: "150m"
      memoryRequest: "250Mi"
```

# AppSecTool

Хранит конфигурации инструментов

Минимально и достаточно:

- что может сканировать
- где расположен

Кастомизация:

- необходимые ресурсы

доп информация о конфигурации

- типизация артефакта
- тегирование

```
apiVersion: cache.appsecauto.ru/v1alpha1
kind: AppSecTool
metadata:
  name: semgrep-sast
spec:
  name: semgrep-sast
  toolConfiguration:
    - state: inJob
      tag: "python-basic"
      artifactType: "code"
      languages: ["Python"]
      accuracy: "0.9"
      image: "semgrep/semgrep:1.79-nonroot"
      args: ["semgrep", "scan", "/workspace/repo", "--json"]
      reportPath: "/workspace/semgrep/report.json"
      cpuRequest: "150m"
      memoryRequest: "250Mi"
```

# AppSecTool

Хранит конфигурации инструментов

Минимально и достаточно:

- что может сканировать
- где расположен

Кастомизация:

- необходимые ресурсы
- доп информация о конфигурации
- ...► типизация артефакта
- тегирование

```
apiVersion: cache.appsecauto.ru/v1alpha1
kind: AppSecTool
metadata:
  name: semgrep-sast
spec:
  name: semgrep-sast
  toolConfiguration:
    - state: inJob
      tag: "python-basic"
      artifactType: "code"
      languages: ["Python"]
      accuracy: "0.9"
      image: "semgrep/semgrep:1.79-nonroot"
      args: ["semgrep", "scan", "/workspace/repo", "--json"]
      reportPath: "/workspace/semgrep/report.json"
      cpuRequest: "150m"
      memoryRequest: "250Mi"
```

# AppSecTool

Хранит конфигурации инструментов

Минимально и достаточно:

- что может сканировать
- где расположен

Кастомизация:

- необходимые ресурсы
- доп информация о конфигурации
- типизация артефакта
- ...► тегирование

```
apiVersion: cache.appsecauto.ru/v1alpha1
kind: AppSecTool
metadata:
  name: semgrep-sast
spec:
  name: semgrep-sast
  toolConfiguration:
    - state: inJob
      tag: "python-basic"
      artifactType: "code"
      languages: ["Python"]
      accuracy: "0.9"
      image: "semgrep/semgrep:1.79-nonroot"
      args: ["semgrep", "scan", "/workspace/repo", "--json"]
      reportPath: "/workspace/semgrep/report.json"
      cpuRequest: "150m"
      memoryRequest: "250Mi"
```

# AppSecTool

Для AppSec-инженера:

Централизованное управление  
инструментами сканирования

Апробация новых инструментов  
сканирования



```
apiVersion: cache.appsecauto.ru/v1alpha1
kind: AppSecTool
metadata:
  name: semgrep-sast
spec:
  name: semgrep-sast
  toolConfiguration:
    - state: inJob
      tag: "python-basic"
      artifactType: "code"
      languages: ["Python"]
      accuracy: "0.9"
      image: "semgrep/semgrep:1.79-nonroot"
      args: ["semgrep", "scan", "/workspace/repo", "--json"]
      reportPath: "/workspace/semgrep/report.json"
      cpuRequest: "150m"
      memoryRequest: "250Mi"
```

# SecurityQualityGate

Устанавливает пороги SQG

...▶ уязвимости

- метрики кода
- тегирование



```
apiVersion: cache.appsecauto.ru/v1alpha1
kind: SecurityQualityGate
metadata:
  name: default-security-gate
  namespace: default
spec:
  # Пороги уязвимостей по степени критичности
  vulnerabilityThresholds:
    critical: 0      # Не допускаются критические уязвимости
    high: 2          # Максимум 2 высокие уязвимости
    medium: 10       # Максимум 10 средних
    low: 20          # Максимум 20 низких

  # Пороги по OWASP Top 10
  owaspVulnerabilityThresholds:
    brokenAccessControl: 1      # Макс. 1 нарушение контроля доступа
    injection: 0               # Инъекции запрещены
    insecureDesign: 3          # Макс. 3 проблемы дизайна
    serverSideRequestForgery: 1 # Макс. 1 SSRF

  # Пороги качества кода
  codeQualityThresholds:
    maxVulnerabilityDensity: "10" # Макс. 10 уязвимостей на 1000 строк кода
    maxUnsafeLanguages: 1         # Макс. 1 небезопасный язык
    maxDirectoryDepth: 8         # Макс. глубина вложенности папок
    maxBinaryFiles: 5            # Макс. 5 бинарных файлов в проекте

  # Пороги security-метрик
  securityMetricsThresholds:
    minSecurityScore: "85"       # Минимальный security score (0-100)
    maxLanguageRiskScore: "30"   # Макс. риск из-за используемых языков
```

# SecurityQualityGate

Устанавливает пороги SQG

- уязвимости
- ...▶ метрики кода
- тегирование



```
apiVersion: cache.appsecauto.ru/v1alpha1
kind: SecurityQualityGate
metadata:
  name: default-security-gate
  namespace: default
spec:
  # Пороги уязвимостей по степени критичности
  vulnerabilityThresholds:
    critical: 0      # Не допускаются критические уязвимости
    high: 2          # Максимум 2 высокие уязвимости
    medium: 10       # Максимум 10 средних
    low: 20          # Максимум 20 низких

  # Пороги по OWASP Top 10
  owaspVulnerabilityThresholds:
    brokenAccessControl: 1      # Макс. 1 нарушение контроля доступа
    injection: 0               # Инъекции запрещены
    insecureDesign: 3          # Макс. 3 проблемы дизайна
    serverSideRequestForgery: 1 # Макс. 1 SSRF

  # Пороги качества кода
  codeQualityThresholds:
    maxVulnerabilityDensity: "10" # Макс. 10 уязвимостей на 1000 строк кода
    maxUnsafeLanguages: 1         # Макс. 1 небезопасный язык
    maxDirectoryDepth: 8          # Макс. глубина вложенности папок
    maxBinaryFiles: 5             # Макс. 5 бинарных файлов в проекте

  # Пороги security-метрик
  securityMetricsThresholds:
    minSecurityScore: "85"        # Минимальный security score (0-100)
    maxLanguageRiskScore: "30"    # Макс. риск из-за используемых языков
```

# SecurityQualityGate

Устанавливает пороги SQG

- уязвимости
- метрики кода
- ...► тегирование



```
apiVersion: cache.appsecauto.ru/v1alpha1
kind: SecurityQualityGate
metadata:
  name: default-security-gate
  namespace: default
spec:
  # Пороги уязвимостей по степени критичности
  vulnerabilityThresholds:
    critical: 0      # Не допускаются критические уязвимости
    high: 2          # Максимум 2 высокие уязвимости
    medium: 10       # Максимум 10 средних
    low: 20          # Максимум 20 низких

  # Пороги по OWASP Top 10
  owaspVulnerabilityThresholds:
    brokenAccessControl: 1      # Макс. 1 нарушение контроля доступа
    injection: 0               # Инъекции запрещены
    insecureDesign: 3          # Макс. 3 проблемы дизайна
    serverSideRequestForgery: 1 # Макс. 1 SSRF

  # Пороги качества кода
  codeQualityThresholds:
    maxVulnerabilityDensity: "10" # Макс. 10 уязвимостей на 1000 строк кода
    maxUnsafeLanguages: 1         # Макс. 1 небезопасный язык
    maxDirectoryDepth: 8         # Макс. глубина вложенности папок
    maxBinaryFiles: 5            # Макс. 5 бинарных файлов в проекте

  # Пороги security-метрик
  securityMetricsThresholds:
    minSecurityScore: "85"       # Минимальный security score (0-100)
    maxLanguageRiskScore: "30"   # Макс. риск из-за используемых языков
```

# SecurityQualityGate

Для аналитика:

Анализ трендов безопасности по проектам

Внедрение адаптивных порогов безопасности в командах разработки



```
apiVersion: cache.appsecauto.ru/v1alpha1
kind: SecurityQualityGate
metadata:
  name: default-security-gate
  namespace: default
spec:
  # Пороги уязвимостей по степени критичности
  vulnerabilityThresholds:
    critical: 0      # Не допускаются критические уязвимости
    high: 2         # Максимум 2 высокие уязвимости
    medium: 10      # Максимум 10 средних
    low: 20         # Максимум 20 низких

  # Пороги по OWASP Top 10
  owaspVulnerabilityThresholds:
    brokenAccessControl: 1      # Макс. 1 нарушение контроля доступа
    injection: 0               # Инъекции запрещены
    insecureDesign: 3          # Макс. 3 проблемы дизайна
    serverSideRequestForgery: 1 # Макс. 1 SSRF

  # Пороги качества кода
  codeQualityThresholds:
    maxVulnerabilityDensity: "10" # Макс. 10 уязвимостей на 1000 строк кода
    maxUnsafeLanguages: 1         # Макс. 1 небезопасный язык
    maxDirectoryDepth: 8         # Макс. глубина вложенности папок
    maxBinaryFiles: 5           # Макс. 5 бинарных файлов в проекте

  # Пороги security-метрик
  securityMetricsThresholds:
    minSecurityScore: "85"      # Минимальный security score (0-100)
    maxLanguageRiskScore: "30"  # Макс. риск из-за используемых языков
```

# AppSecJob

Описывает детали запуска сканирования

Минимально и достаточно:

...» объект сканирования

Кастомизация :

- явный выбор инструментов сканирования и/или практики
- режим сканирования
- явный выбор SQG
- обработка и выгрузка результата

```
apiVersion: cache.appsecauto.ru/v1alpha1
kind: AppSecJob
metadata:
  name: appsecjob-python-basic
spec:
  level: development
  complexity: balanced
  practice: sast
  tools:
    - "semgrep-sast:python-basic"
  artifact:
    code:
      repository:
        url: https://github.com/fportantier/vulpy.git
        port: 443
        protocol: HTTP
        commithash: 5249cc8b05a1c37f6b2f757b1cf16a509c327122
  resultDestination:
    s3:
      host:
        url: http://go-minio-service.go-system
        port: 9000
        protocol: HTTP
        bucket: dev-results
  reports:
    parserImage: localhost:5000/sast-parser
```

# AppSecJob

Описывает детали запуска сканирования

Минимально и достаточно:

- объект сканирования

Кастомизация :

...▶ явный выбор инструментов  
сканирования и/или практики

- режим сканирования
- явный выбор SQG
- обработка и выгрузка результата

```
apiVersion: cache.appsecauto.ru/v1alpha1
kind: AppSecJob
metadata:
  name: appsecjob-python-basic
spec:
  level: development
  complexity: balanced
  practice: sast
  tools:
    - "semgrep-sast:python-basic"
  artifact:
    code:
      repository:
        url: https://github.com/fportantier/vulpy.git
        port: 443
        protocol: HTTP
        commithash: 5249cc8b05a1c37f6b2f757b1cf16a509c327122
  resultDestination:
    s3:
      host:
        url: http://go-minio-service.go-system
        port: 9000
        protocol: HTTP
        bucket: dev-results
  reports:
    parserImage: localhost:5000/sast-parser
```

# AppSecJob

Описывает детали запуска сканирования

Минимально и достаточно:

- объект сканирования

Кастомизация :

- явный выбор инструментов сканирования и/или практики

...► режим сканирования

- явный выбор SQG
- обработка и выгрузка результата

```
apiVersion: cache.appsecauto.ru/v1alpha1
kind: AppSecJob
metadata:
  name: appsecjob-python-basic
spec:
  level: development
  complexity: balanced
  practice: sast
  tools:
    - "semgrep-sast:python-basic"
  artifact:
    code:
      repository:
        url: https://github.com/fportantier/vulpy.git
        port: 443
        protocol: HTTP
        commithash: 5249cc8b05a1c37f6b2f757b1cf16a509c327122
  resultDestination:
    s3:
      host:
        url: http://go-minio-service.go-system
        port: 9000
        protocol: HTTP
        bucket: dev-results
  reports:
    parserImage: localhost:5000/sast-parser
```

# AppSecJob

Описывает детали запуска сканирования

Минимально и достаточно:

- объект сканирования

Кастомизация :

- явный выбор инструментов сканирования и/или практики
- режим сканирования
- ...▶ явный выбор SQG
- обработка и выгрузка результата

```
apiVersion: cache.appsecauto.ru/v1alpha1
kind: AppSecJob
metadata:
  name: appsecjob-python-basic
spec:
  level: development
  complexity: balanced
  practice: sast
  tools:
    - "semgrep-sast:python-basic"
  artifact:
    code:
      repository:
        url: https://github.com/fportantier/vulpy.git
        port: 443
        protocol: HTTP
        commithash: 5249cc8b05a1c37f6b2f757b1cf16a509c327122
  resultDestination:
    s3:
      host:
        url: http://go-minio-service.go-system
        port: 9000
        protocol: HTTP
        bucket: dev-results
  reports:
    parserImage: localhost:5000/sast-parser
```

# AppSecJob

Описывает детали запуска сканирования

Минимально и достаточно:

- объект сканирования

Кастомизация :

- явный выбор инструментов сканирования и/или практики
- режим сканирования
- явный выбор SQG
- ...▶ обработка и выгрузка результата

```
apiVersion: cache.appsecauto.ru/v1alpha1
kind: AppSecJob
metadata:
  name: appsecjob-python-basic
spec:
  level: development
  complexity: balanced
  practice: sast
  tools:
    - "semgrep-sast:python-basic"
  artifact:
    code:
      repository:
        url: https://github.com/fportantier/vulpy.git
        port: 443
        protocol: HTTP
        commithash: 5249cc8b05a1c37f6b2f757b1cf16a509c327122
  resultDestination:
    s3:
      host:
        url: http://go-minio-service.go-system
        port: 9000
        protocol: HTTP
        bucket: dev-results
  reports:
    parserImage: localhost:5000/sast-parser
```

# AppSecJob

Для разработчика:

Проверки безопасности на ранних  
стадиях процесса разработки

Для AppSec-инженера:

Запуск AppSec-сканирований  
на любом этапе



```
apiVersion: cache.appsecauto.ru/v1alpha1
kind: AppSecJob
metadata:
  name: appsecjob-python-basic
spec:
  level: development
  complexity: balanced
  practice: sast
  tools:
    - "semgrep-sast:python-basic"
  artifact:
    code:
      repository:
        url: https://github.com/fportantier/vulpy.git
        port: 443
        protocol: HTTP
        commithash: 5249cc8b05a1c37f6b2f757b1cf16a509c327122
  resultDestination:
    s3:
      host:
        url: http://go-minio-service.go-system
        port: 9000
        protocol: HTTP
        bucket: dev-results
  reports:
    parserImage: localhost:5000/sast-parser
```

# Оркестрация на уровне оператора

Анализ кода

Выбор инструментов

Создание джобы

Обработка отчётов

Security Quality Gate

Сбор статистики

- клонирование репозитория
- определение языкового состава
- определение кодовой статистики
- формирование метрик

## Artifact Analysis:

### Code Analysis Result:

Blank Lines: 975  
Clone Path: /tmp/artifacts/tmp/artifact/repo  
Code Lines: 57296  
Comment Lines: 87  
Complexity Score: 32  
File Count: 108

### Languages:

CSS: 4  
CSV: 1  
HTML: 28  
Ignore List: 3  
Java Script: 6  
Pip Requirements: 1  
Python: 57  
Shell: 1  
Text: 6  
Re Structured Text: 1

### Lines Of Code:

CSS:  
Blanks: 0  
Code: 470  
Comments: 0  
Total: 470  
CSV:  
Blanks: 0

# Оркестрация на уровне оператора

Анализ кода

Выбор инструментов

Создание джобы

Обработка отчётов

Security Quality Gate

Сбор статистики

- фильтрация AppSecTools по параметрам из AppSecJob и характеристикам артефакта
- сортировка по точности и ресурсозатратности
- выбор количества, определяемого режимом проверки

# Оркестрация на уровне оператора

Анализ кода

Выбор инструментов

Создание джобы

Обработка отчётов

Security Quality Gate

Сбор статистики

- клонирующий init-контейнер
- контейнеры с инструментами сканирования
- контейнер с парсером отчётов
- контейнер выгрузки результатов
- тома для совместного использования артефакта, отчётов от сканеров и итоговых результатов

# Оркестрация на уровне оператора

Анализ кода

Выбор инструментов

Создание джобы

Обработка отчётов

Security Quality Gate

Сбор статистики

- извлечение findings
- нормализация
- дообогащение
- маппинг
- дедупликация
- генерация и сохранение результатов

```
{  
  "type": "python.sqlalchemy.security.sqlalchemy-execute-raw-q  
  "description": "Avoiding SQL string concatenation: untrusted  
  "file_path": "/workspace/repo/bad/db.py",  
  "line_number": 19,  
  "column_start": 9,  
  "column_end": 103,  
  "cwe": [  
    "CWE-89: Improper Neutralization of Special Elements used :  
  ],  
  "owasp": [  
    {  
      "code": "A03:2021",  
      "description": "Injection"  
    }  
  ],  
  "severity": "high",  
  "confidence": 2.0,  
  "tool": "Semgrep OSS",  
  "logical_location": "",  
  "code_snippet": "c.execute(\"INSERT INTO users (user  
  "related_locations": [],  
  "code_flows": [],  
  "taxa": []  
}
```

# Оркестрация на уровне оператора

Анализ кода

Выбор инструментов

Создание джобы

Обработка отчётов

Security Quality Gate

Сбор статистики

- извлечение findings
- нормализация
- дообогащение
- маппинг
- дедупликация
- генерация и сохранение результатов

```
{  
  "overall_status": "Critical",  
  "severity_statistics": {  
    "low": 4,  
    "medium": 33,  
    "high": 247,  
    "critical": 18  
  },  
  "owasp_statistics": {  
    "A02:2021": 18,  
    "A03:2021": 45,  
    "A08:2021": 2,  
    "A01:2021": 19,  
    "A04:2021": 2,  
    "A07:2021": 2,  
    "A05:2021": 12,  
    "A10:2021": 6  
  }  
}
```

# Оркестрация на уровне оператора

Анализ кода

Выбор инструментов

Создание джобы

Обработка отчётов

Security Quality Gate

Сбор статистики

```
Status:
  SQG Failed Reason: Failed thresholds: High: 11 > 2, Medium: 30 > 10, CryptographicFailures: 4 > 1, Injection: 12 > 0,
: 18 > 1
  SQG Status: Fail
Overall Status: Feel bad
Security Metrics:
  Binary Files Count: 102
  Comment Coverage: 0.15
  Kloc: 57.30
  Language Risk Score: 10.00
  Maintainability Index: 50.38
  Unsafe Languages:
    JavaScript (6 files)
    Python (57 files)
  Vulnerability Density: 0.79
Severity Statistics:
  Critical: 0
  High: 11
  Low: 4
  Medium: 30
Top CW Es Formatted: CWE-522: Insufficiently Protected Credentials | CWE-89: Improper Neutralization of Special Eleme
CWE-489: Active Debug Code
```

- выбор ресурса SQG для оценки результатов джобы
- загрузка результатов и оценка на соответствие порогам
- фиксация результата в поле Status ресурса AppSecJob.

# Оркестрация на уровне оператора

Анализ кода

Выбор инструментов

Создание джобы

Обработка отчётов

Security Quality Gate

Сбор статистики

```
~/Doc/k/o/g/appsec-operator/tests v1.0 !25 ?20 > kubectl get sqg -o wide
```

| NAME                  | STATUS    | PASS RATE | PASSED | FAILED | AGE  | CRITICAL | HIGH | MEDIUM | LOW |
|-----------------------|-----------|-----------|--------|--------|------|----------|------|--------|-----|
| default-security-gate | Active    | 33%       | 1      | 2      | 27h  | 0        | 2    | 10     | 20  |
| duble-security-gate   | Duplicate | 0%        | 0      | 0      | 116s | 0        | 2    | 10     | 20  |

```
Failed Jobs: 2
```

```
Job Statuses:
```

```
Job Name: appsecjob-sqgpassed-2
```

```
Message: Job passed security quality gate
```

```
Passed: true
```

```
Job Name: appsecjob-toolselection-1
```

```
Message: Job failed: Failed thresholds: High: 11 > 2, Medium: 30 > 10, UnsafeLanguages: 2 > 1 (Python (57 files), JavaScript (6 files)), CryptographicFailures: 4 > 0, Injection: 12 > 0, IntegrityFailures: 2 > 0, BrokenAccessControl: 18 > 1
```

```
Passed: false
```

```
Job Name: appsecjob-toolselection-2
```

```
Message: Job failed: Failed thresholds: High: 11 > 2, Medium: 30 > 10, UnsafeLanguages: 2 > 1 (JavaScript (6 files), Python (57 files)), CryptographicFailures: 4 > 0, Injection: 12 > 0, IntegrityFailures: 2 > 0, BrokenAccessControl: 18 > 1
```

```
Passed: false
```

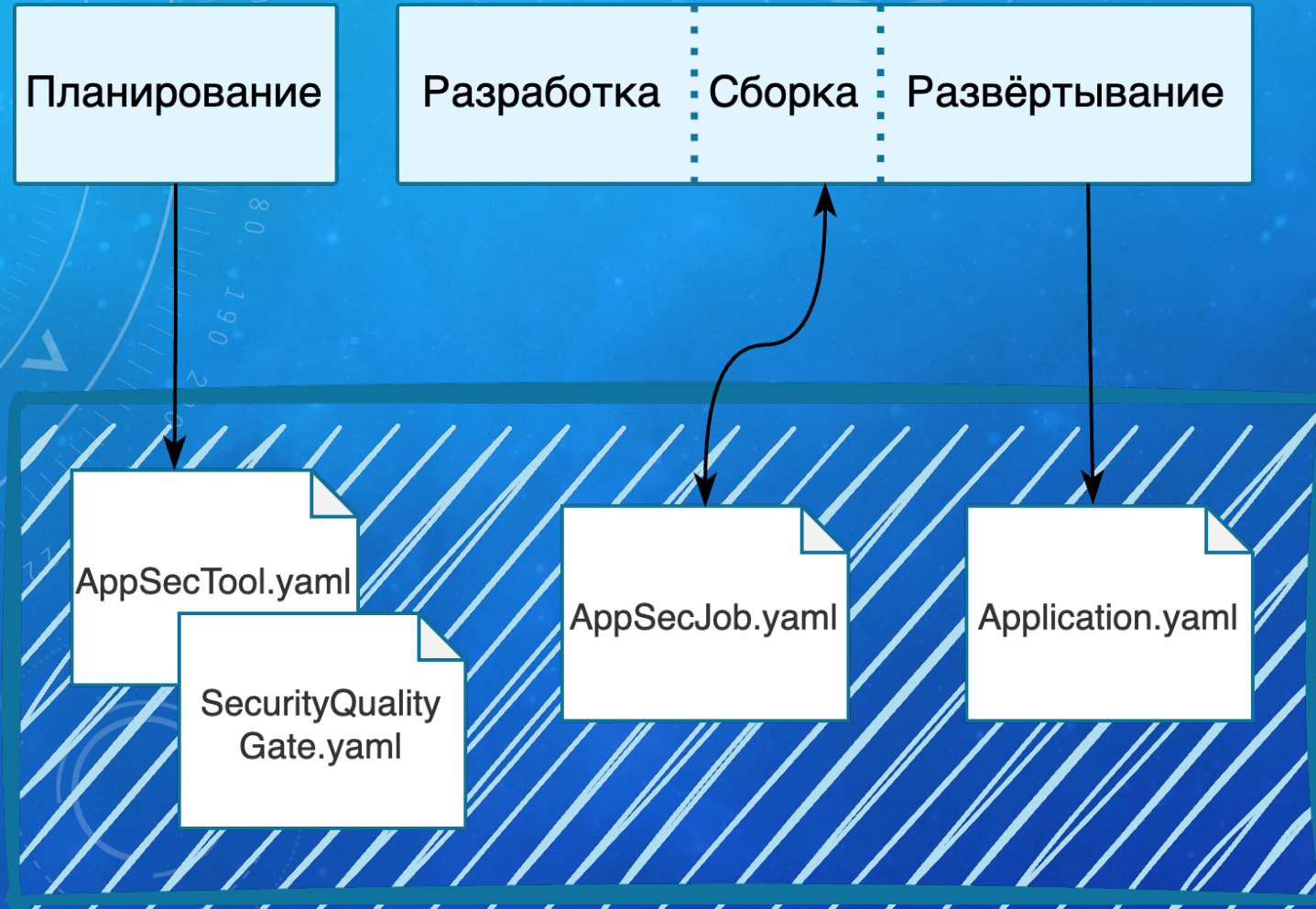
```
Last Evaluation: 2025-05-02T18:55:12Z
```

```
Pass Rate: 33%
```

```
Passed Jobs: 1
```

- просмотр всех AppSecJob
- сбор и сохранение статистики по прохождению оценки в статусе SQG

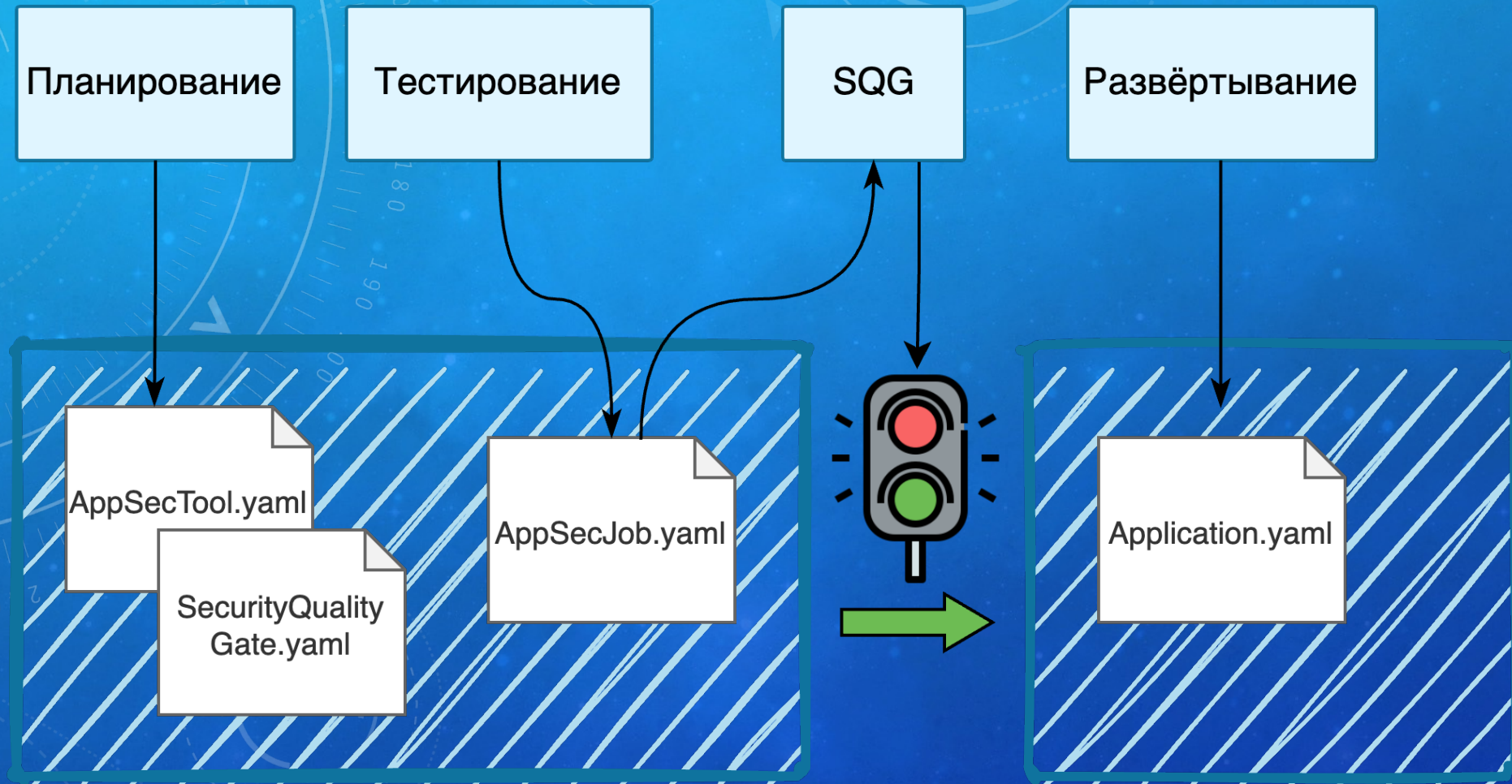
# Как включить проверки в пайп?



ручной запуск проверок в кластере разработки



# Как включить проверки в пайп?



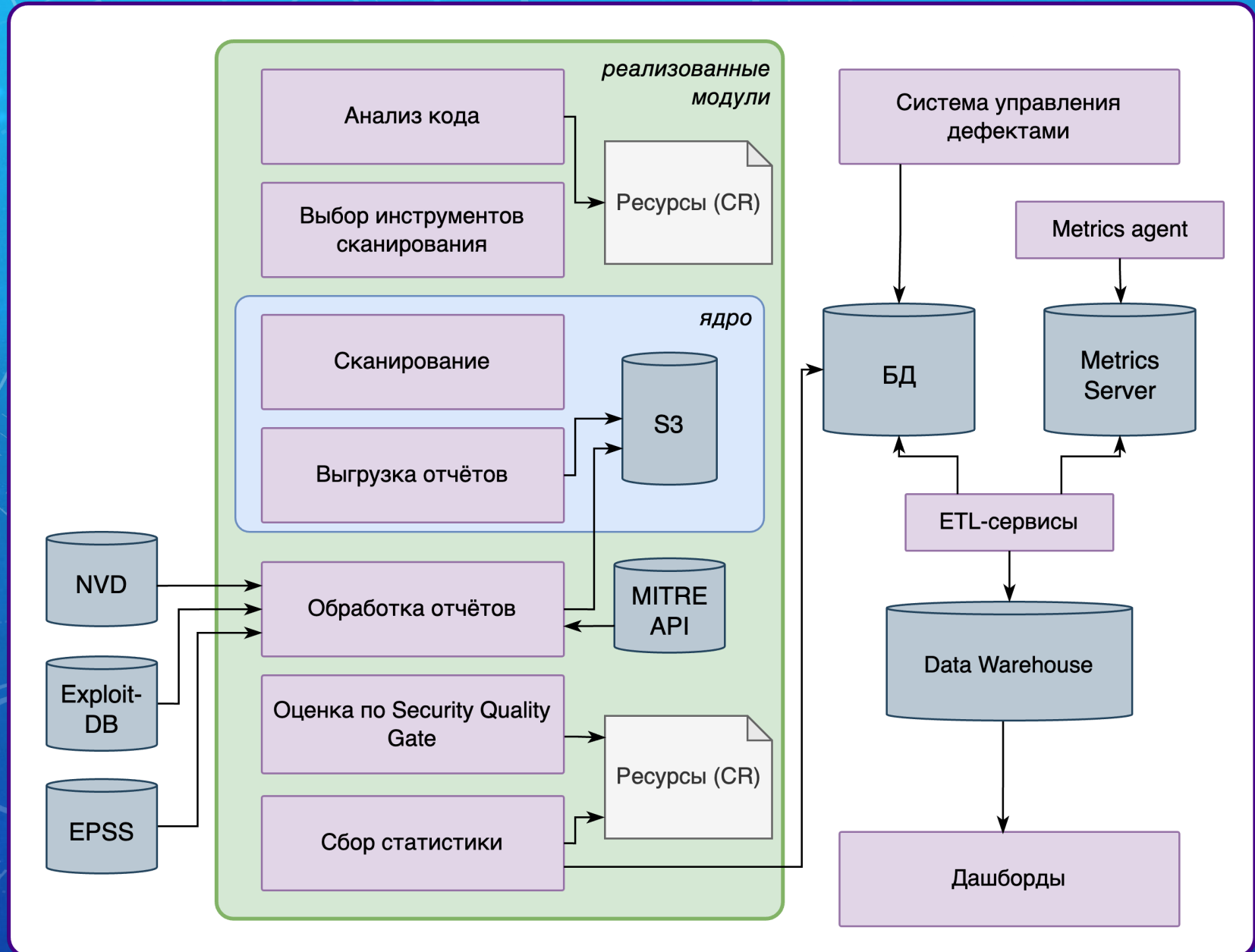
# Как включить проверки в пайп?



периодические проверки и проверки по событию



# Overview



# Сегодня рассмотрели

- Этапы AppSec-проверки и что следует учесть
- Для сканирования нужны инструмент, пороги контроля и джоба (+ тот, кто этим управляет)
- ShiftLeft – хорошо, но на собственных ресурсах – ещё лучше
- Для Security Quality Gate необходима унификация результатов
- Управление на уровне Kubernetes-абстракций может быть удобно



# Оркестрация AppSec-сканирований с использованием Kubernetes

Алена Жилина

 @coalyonysh