

# Интеграция OPA/REGO проверок в Jenkins pipeline

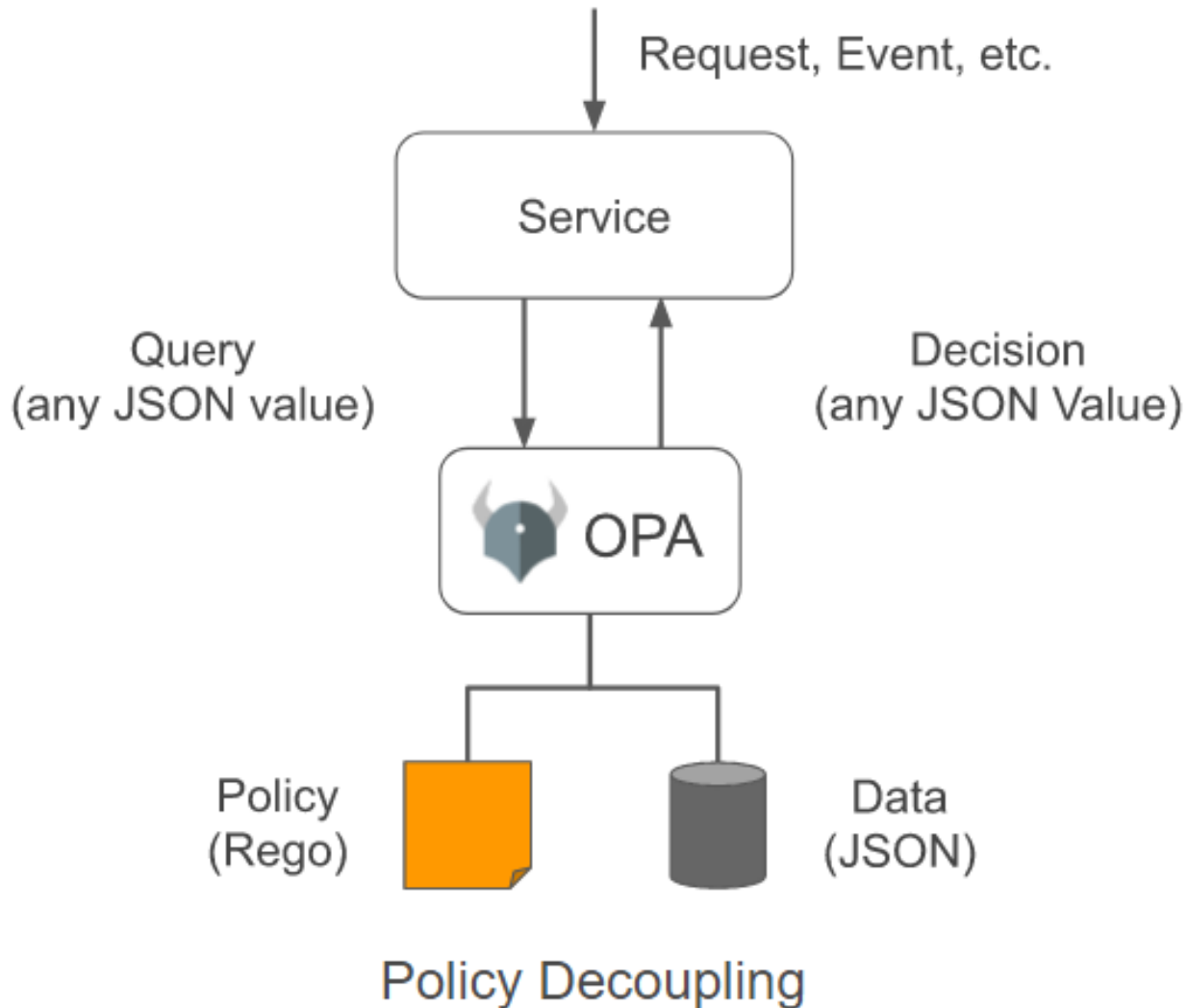
Евгений Злобин

Архитектор, ДКА

# Краткое содержание

- Что такое OPA/REGO
- Какие варианты интеграции в Jenkins pipeline
- Стенд для экспериментирования
- Демонстрация
- Полезные ресурсы

# Open Policy Agent



- Движок политик
- Написан на Go
- Быстрый, потому что inmemory
- Проверяет, что угодно
- Язык Rego декларативен
- Не сложнее SQL
- Внешние данные

# Примеры проверок

```
package k8s.probes.readiness

apply_readiness_probe = {"msg": msg, "status": status, "type": type, "name": name }{
  deployment:=lower(input.kind)
  deployment == "deployment"
  t:=input.spec.template.spec.containers[_]
  t1:=t.readinessProbe.httpGet.path
  startswith(t1, "/") == true
  count(t1)>2
#   msg:=sprintf("Проверка на ReadinessProbe пройдена: %s", [t1])
  msg:="Проверка на ReadinessProbe пройдена"
  status:= "1"
  type:="Проверка на наличие ReadinessProbe endpoint"
  name:=input.metadata.name
}

else = {"msg": msg, "status": status, "type": type, "name": name }{
  deployment:=lower(input.kind)
  deployment == "deployment"
  t:=input.spec.template.spec.containers[_]
  t1:=t.readinessProbe.tcpSocket.port
  count(t1)>0
  msg:="Проверка на ReadinessProbe пройдена"
  status:= "1"
  type:="Проверка на наличие ReadinessProbe endpoint"
  name:=input.metadata.name
}
```

# Как работать с OPA & Rego

- Использовать через командную строку

```
# Evaluate a trivial expression.  
./opa eval '1*2+3'
```

- Interactive shell

```
./opa run
```

- Как сервер

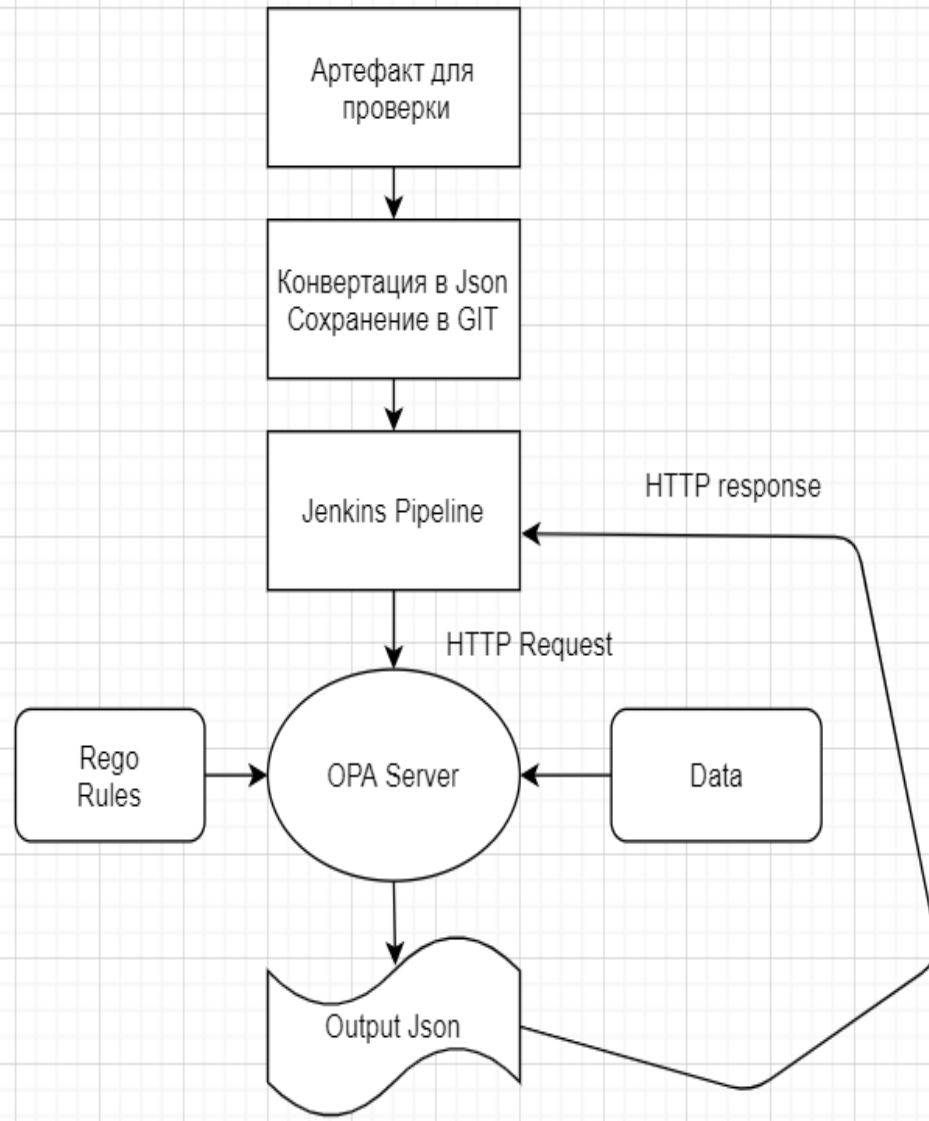
```
./opa run --server ./example.rego
```

- Через REST API

```
GET /v1/policies/example1 HTTP/1.1
```

- Клиентская библиотека для GO, для Node JS (не официальная)

## Интеграции OPA - Jenkins



- ✓ Конвертация артефакта в json (например pom.xml)
- ✓ Сохранение его в GIT
- ✓ Jenkins pipeline загружает json и отправляет его через HTTP POST в OPA server
- ✓ OPA сервер содержит в себе все нужные rego правила для проверок и необходимые дополнительные данные
- ✓ OPA сервер возвращает json с результатом проверки
- ✓ Jenkins pipeline принимает решение о дальнейших шагах

# Стенд OPA-Jenkins

- Использует docker образы стандартного воркшопа для java + OPA образ
- Rego политики содержаться в репозитории проекта и загружаются в OPA сервер из Jenkins pipeline
- Продемонстрированы некоторые примеры работы с OPA сервер API
  - Получение списка политик
  - Загрузка rego политик в OPA сервер
  - Применение правила (запрос в OPA сервер)

# Rego политика

```
1  package j2opa
2
3  apply_maven = {"msg": msg, "status": status, "type": type, "name": name } {
4      dependency := input.project.dependencies.dependency
5      t := dependency.groupId
6      t == "junit"
7      t1 := to_number(dependency.version)
8      t1 >= 4.11
9      status := 1
10     type := "junit check"
11     msg := "junit check done"
12     name := sprintf("%s.%s:%s", [dependency.groupId, dependency.artifactId, dependency.version])
13 }
14 else = {"msg": msg, "status": status, "type": type, "name": name } {
15     dependency := input.project.dependencies.dependency
16     status := 0
17     type := "junit check"
18     msg := "junit check fail"
19     name := sprintf("%s.%s:%s", [dependency.groupId, dependency.artifactId, dependency.version])
20 }
21
```

# Пример Jenkins pipeline

- 1) Загрузить файл с rego проверками
- 2) Загрузить проверяемый артефакт в формате json (в данном случае, pom.xml)
- 3) Проверить какие политики загружены (не обязательный шаг)
- 4) Послать файл проверок в OPA сервер
- 5) Сделать проверку в OPA сервер для тестируемого артефакта
- 6) Преобразовать ответ OPA сервера в json формат
- 7) Сделать финальную проверку на соответствие правилу

```
stage('Junit version check'){  
    steps {  
        script {  
            1  
            def testrego = readFile(file: 'test.rego')  
            println(testrego)  
            2  
            def testjson_ = readFile(file: 'input.json')  
            println(testjson_)  
            3  
            def response = httpRequest "http://172.22.0.5:8181/v1/policies"  
            println('Status: '+response.status)  
            println('Response: '+response.content)  
            4  
            httpRequest(url: 'http://172.22.0.5:8181/v1/policies/test1',  
                acceptType: 'TEXT_PLAIN',  
                contentType: 'TEXT_PLAIN',  
                httpMode: 'PUT',  
                timeout: 1000,  
                requestBody: "${testrego}",  
                responseHandle: 'STRING',  
                validResponseCodes: '200')  
            5  
            def res1 = httpRequest(url: 'http://172.22.0.5:8181/v1/data/j2opa/apply_maven',  
                acceptType: 'APPLICATION_JSON',  
                contentType: 'APPLICATION_JSON',  
                httpMode: 'POST',  
                requestBody: "${testjson_}",  
                responseHandle: 'STRING',  
                validResponseCodes: '200')  
            6  
            println('Status: '+res1.status)  
            println('Response: '+res1.content)  
            7  
            def props = readJSON text: res1.content  
            println(props)  
            println(props['result'].status)  
            if (props['result'].status == 0)  
                error("Don't pass Junit version check")  
        }  
    }  
}
```

Демонстрация

# Полезные ресурсы

- <https://www.openpolicyagent.org/docs/latest/>
- <https://play.openpolicyagent.org/>
- <https://marketplace.visualstudio.com/items?itemName=tsandall.opa>
- [OPA & REGO deep dive](#)
  - <https://www.slideshare.net/TorinSandall/rego-deep-dive>
- <https://sbtatlas.sigma.sbrf.ru/stash/users/18158458/repos/opa-jenkins-sandbox>