Сфера: Тестирование Веб

Руководство администратора

Содержание

1. Общие сведения

Данное руководство администратора является частью эксплуатационной документации для программного продукта «Сфера: Тестирование Веб».
Руководство администратора содержит описание общих системных требований и свойств продукта.

1.1. Назначение и краткая характеристика системы

Программный продукт «Сфера: Тестирование Веб» предназначен для организации ручного и автоматизированного тестирования веб-приложений в настольных и мобильных браузерах на основе протокола Selenium WebDriver, работающий в кластере Kubernetes.
«Сфера: Тестирование Веб» поставляется набором готовых образов контейнеров для тестирования в поддерживаемых браузерах и мобильных платформах: Google Chrome, Mozilla Firefox, Opera, Internet Explorer, Microsoft Edge, Android и других.
Продукт «Сфера: Тестирование Веб» реализовывается в виде веб-приложения, работа с которым осуществляется через браузер в случае ручного тестирования и с посредством интеграции с инструментами для запуска автотестов в случае автоматизированного тестирования.

1.2. Цели и задачи системы

Основная цель разработки продукта «Сфера: Тестирование Веб» – предоставление удобного и надёжного инструмента отечественной разработки, который обеспечивает эффективное тестирование веб-приложений в настольных и мобильных браузерах на основе протокола Selenium WebDriver.
Продукт «Сфера: Тестирование Веб» предназначен для пользователей, участвующих в тестировании и разработке программного обеспечения.
В задачи продукта входит:

- эмулирование различных браузеров, обеспечение их параллельного запуска и автоматического масштабирования кластера в зависимости от нагрузки;
- поддержка создания проектных областей, в которых пользователи смогут запускать браузеры независимо друг от друга, обеспечивая тем самым безопасность и конфиденциальность;
- поддержка ролевой модели для предоставления различных уровней доступа к приложению;
- поддержка возможности конфигурировать параметры окружений для каждой сессии;
- гибкий контроль за потреблением ресурсов, выделяемых каждому запускаемому браузеру;
- надежное хранение лог-файлов и видеозаписей выполняемых тестов;
- обеспечение отказоустойчивости за счёт одновременно запущенных нескольких копий приложения.

1.3. Требования к программному обеспечению

Для эксплуатации продукта «Сфера: Тестирование Веб» и корректной работы его функциональных компонентов необходимо следующее программное обеспечение:

- ОС, имеющая интерпретатор командной строки (Linux, MacOS, Windows);
- Kubernetes версии 1.16 или выше и установленная утилита `kubectl`, распространяемая под лицензией Apache 2.0;
- Хранилище S3 (если необходимо сохранять различные объекты тестирования).

1.4. Общие системные требования

Для корректной эксплуатации приложения «Тестирование Веб» убедитесь, что соблюдены требования к рабочим станциями пользователей и используемым веб-браузерам.

1.4.1. Рекомендуемые характеристики рабочих станций

Для работы одного клиентского места пользователя рекомендуется использовать персональный компьютер, оснащённый техническими средствами со следующими минимальными характеристиками:

- процессор Intel Core 2-ядерный с тактовой частотой 3 ГГц или аналогичный;
- оперативная память объёмом 4 Гб;
- цветной монитор, позволяющий использовать для отображения веб-приложения рабочую область экрана размером 1680x1050 px (но не менее 1024x768 px);

Примечание. Для корректного отображения элементов веб-приложения «Тестирование Веб» рекомендуется использовать 100% масштаб дисплея в настройках операционной системы ПК.

- устройства ввода (клавиатуру, «мышь»);
- пропускная способность канала связи между рабочей станцией и веб-сервером – 2 Мбит/с.

1.4.2. Требования по настройке веб-браузера

Работа с приложением «Тестирование Веб» выполняется с помощью веб-браузеров.

В качестве веб-браузера могут быть использованы MS EDGE, Google Chrome и другие браузеры последних устойчивых версий.

Основные требования по настройке веб-браузеров:

- убедитесь, что установлены последние обновления браузеров;
- отключите все сторонние дополнения и расширения (Addons), которые могут блокировать веб-элементы приложения «Тестирование Веб»;
- используйте 100% масштаб браузера.

Примечание. Если персональные настройки ОС Windows на ПК пользователя приводят к некорректному отображению веб-форм (например, нестандартный масштаб шрифтов и пр.), то для устранения проблемы выполните сброс настроек темы ОС.

2. Особенности работы системы

Для эксплуатации продукта «Сфера: Тестирование Веб» подходит любое серверное оборудование или виртуальные машины, на которых возможна эксплуатация кластера Kubernetes.

Оборудование заказчика должно работать круглосуточно.

В штатном режиме «Сфера: Тестирование Веб» отвечает на запросы круглосуточно, автоматически и без перерывов. Для штатной работы необходимо держать постоянно запущенной как минимум один экземпляр ИС за сетевым балансировщиком и обеспечивать сетевую связность между пользователями и запущенным экземпляром ИС средствами Kubernetes.

Для работы продукта достаточно одного его запущенного экземпляра, но для отказоустойчивости рекомендуется иметь не менее двух экземпляров.

3. Настройка приложения

Раздел содержит описание конфигурирования разных компонентов продукта, необходимых для работы с приложением «Сфера: Тестирование Веб».

3.1. Конфигурирование списка пользователей

Продукт «Сфера: Тестирование Веб» является многопользовательским приложением, позволяющим разным пользователям получать доступ к разным версиям браузеров.

Для хранения списка пользователей по умолчанию используется конфигурационный файл формата `htpasswd`, содержащий имена пользователей и зашифрованные пароли, разделенные двоеточием, например:

```
$ cat users.htpasswd
test:$apr1$.dZyHlKN$jdoZkin/kPviFNArx/cVL1 # User is test, password is encrypted
alice:$apr1$mLYJAC4y$VYeJstWjWP/4iVlH/TNcD.
bob:$apr1$gyqzbSpt$RBNcxrsQaolPZCQZW0VQW1
```

Этот файл сохраняется в виде секрета Kubernetes и подключается в приложение при помощи встроенных возможностей Kubernetes (`volumes`).

Для добавления или удаления пользователей выполните следующие действия:

1. Измените файл `users.htpasswd` при помощи команды `htpasswd`:

```
$ htpasswd -Bbn new-user new-user-password >> users.htpasswd # Adding new user
$ htpasswd -Bb users.htpasswd some-user new-password # Updating password
$ htpasswd -D users.htpasswd test-user # Deleting existing user
```

2. Обновите секрет `users`:

```
$ kubectl replace secret users --from-file=./users.htpasswd -n web-test
```

Изменения применяются сразу же, перезагрузка приложения не требуется.

3.2. Конфигурирование списка браузеров

Приложение «Сфера: Тестирование Веб» использует файлы формата JSON для хранения списка доступных браузеров (`browsers.json`). По умолчанию структура файла выглядит следующим образом:

```
{
    "browserName": {
        "default": "x.y",
        "versions": {
            "x.y": {
                "image": "",
                "port": "",
```

```
                    "path": "/wd/hub",
                    "resources": {
                        "limits": {
                            "cpu": "2",
                            "memory": "2Gi"
                        },
                        "requests": {
                            "cpu": "200m",
                            "memory": "1Gi"
                        }
                    },
                    "privileged": false,
                    "nodeSelector": {
                        "node-type": "hardware"
                    },
                    "env": ["TZ=Europe/Moscow", "LANG=ru"],
                    "hosts": ["example.com:192.168.0.1"],
                }
            }
        }
    }
```

где:

- `browserName` – имя браузера;
  - `default` – версия браузера по умолчанию;
  - `versions` – список доступных версий браузера;
    - `x.y` – версия браузера, для которой задаются настройки;
      - `image` – образ для запуска браузера;
      - `port` – сетевой порт для перенаправления соединений (в большинстве случаев должен быть равен `4444`);
      - `path` – полный путь до папки, в которую должен отправляться запрос на создание Selenium сессии;
      - `resources` – конфигурация вычислительных ресурсов, доступных браузеру (память и процессор);
      - `privileged` – признак запуска подов в привилегированном режиме;
        Значение по умолчанию - false (выключено).
      - `nodeSelector` – Kubernetes node selector, настройка запуска подов с браузерами на строго определенных хостах Kubernetes;
      - `env` – переменные окружения (задаются в контейнере с браузером);
      - `hosts` – пользовательские записи в файле `/etc/hosts` в формате `hostname:ip`.

Пример заполнения файла:

```
{
    "firefox": {
        "default": "62.0",
        "versions": {
            "62.0": {
                "image": "selenoid/firefox:62.0",
                "port": "4444",
                "path": "/wd/hub",
                "resources": {
                    "limits": {
                        "cpu": "2",
                        "memory": "2Gi"
                    },
                    "requests": {
                        "cpu": "200m",
                        "memory": "1Gi"
                    }
                },
                "privileged": true,
                "nodeSelector": {
                    "node-type": "hardware"
                },
                "env": ["TZ=Europe/Moscow", "LANG=ru"],
                "hosts": ["example.com:192.168.0.1"],
            },
            "60.0": {
                //...
            }
        }
    },
    "chrome": {
        //...
    },
    "opera": {
        "default": "56.0",
        "versions": {
            //...
        }
```

```
    }
}
```

3.2.1. Списки браузеров для отдельных пользователей

Для назначения разных доступных версий браузеров разным пользователям создайте для каждого пользователя файл с именем `<имя-пользователя>.json`. Например, для пользователя `worker1` из файла `users.htpasswd` создайте файл `worker1.json`.

Все JSON-файлы должны храниться в директории, отмеченной флагом `-quota-dir`, следующим образом:

```
\---quota
        |---- worker1.json
        |---- browsers.json
        |---- worker2.json
        |---- testUser.json
```

Эта директория загружается в `Kubernetes ConfigMap` и монтируется в контейнер приложения «Сфера: Тестирование Веб» средствами Kubernetes `volumes`.

3.2.2. Гостевая квота

Для анонимного доступа в Selenium (без указания имени пользователя и пароля) в приложении «Сфера: Тестирование Веб» используется *гостевая квота*, которая настраивается при помощи флага `-guest-user` (значение по умолчанию – `browsers`).

Версии браузеров, указанные в файле `<guest-user>.json`, будут доступны без указания имени пользователя и пароля:

```
-guest-user browsers ====> browsers.json # Значение по умолчанию
-guest-user guest-user ====> guest-user.json
```

где `<guest-user>` – имя гостевого пользователя, переданное флагом `-guest-user`.

3.2.3. Обновление списка браузеров

Для обновления списка браузеров выполните следующие действия:

1. Внесите необходимые изменения в файл с квотами, хранящийся в директории `quota`;
2. Обновите `ConfigMap` с именем `quota`, используя содержимое директории `quota`:

```
$ kubectl replace configmap quota --from-file=quota -n web-test
```

Изменения применяются сразу же, перезагрузка приложения не требуется. Работа ранее запущенных браузерных сессий не прерывается.

3.3. Дополнительные настройки приложения

«Сфера: Тестирование Веб» хранит дополнительные конфигурационные настройки (например, настройки S3) в конфигурационном файле `service.json`, например, настройки хранилища S3:

```
{
    "s3": {
```

```
            "endpoint": "https://storage.googleapis.com",
            "bucketName": "web-test",
            "version": "S3v2",
            "keyPattern": "$quota/$date"
        },
        "images": {
            "logger": {
                "image": "my-reg.com/web-test/logger:1.2.0",
                "cpu": "0.3",
                "mem": "1024Mi"
            }
        },
        "annotations": {
            "key1": "value1",
            "key2": "value2"
        },
        "labels": {
            "key1": "value1",
            "key2": "value2"
        }
    }
```

где:

- `s3` – раздел настроек хранилища S3;
  - `endpoint` – адрес API S3;
  - `bucketName` – имя контейнера S3;
  - `version` – версия подписи S3 (`S3v2` или `S3v4`);
  - `keyPattern` – шаблон сохранения файлов в S3;
- `images` – раздел для настройки пользовательских образов системных компонентов приложения (для использования приложения с собственным Docker-репозиторием);
  - `logger` – настройки системного компонента приложения;

    Допустимые значения:
    - `logger` – лог-файлы;
    - `defender` – защитник;
    - `videoRecorder` – видеозаписи;
  - `image` – образ системного компонента приложения;
  - `cpu` – количество процессорных ядер, доступных системному компоненту приложения;
  - `mem` – количество памяти, доступное системному компоненту приложения;
- `annotations` – пользовательские аннотации подов (применяется ко всем подам с браузерами);
  - `key1 / key2` – ключ и значение пользовательских аннотаций подов;

- `labels` – пользовательские метки подов (применяется ко всем подам с браузерами);
  - `key1 / key2` – ключ и значение пользовательских меток подов.

Файл `service.json` монтируется в контейнер приложения в виде обычного файла с помощью `Kubernetes ConfigMap`. Путь до смонтированного конфигурационного файла указывается с помощью флага `-config-file`.

Пример добавления файла service.json в приложение:

```
apiVersion: v1
kind: ConfigMap
metadata:
  name: config
  namespace: web-test
data:
  service.json:  |
    {
      "s3": {
        "endpoint": "https://storage.googleapis.com",
        "bucketName": "web-test",
        "version": "S3v2"
    }
  }
---
apiVersion: apps / v1beta1
kind: Deployment
metadata:
  name: web-test
  namespace: web-test
spec:
  template:
    metadata:
      labels:
        app: web-test
    spec:
      containers:
      - name: web-test
        image: webtest / web: latest - release
        args: ["-config-file", "/config/service.json"]
        volumeMounts:
        - name: config
```

```
            mountPath: /config
            readOnly: true
      volumes:
        - name: config
          configMap:
          name: config
```

Изменения применяются сразу же, перезагрузка приложения не требуется.

3.4. Хранение внешних учетных записей

Приложение «Сфера: Тестирование Веб» считывает все учетные записи (например, данные для доступа к хранилищу S3) из дополнительного секрета Kubernetes с именем `credentials`. Этот секрет монтируется в контейнер приложения, при этом каждый ключ хранится в виде отдельного файла. Путь до каталога с учетными записями указывается флагом `-credentials-dir`.
Изменения в секрете `credentials` применяются сразу же, перезагрузка приложения не требуется.

Пример добавления секрета `credentials` в приложение:

```
apiVersion: v1
kind: Secret
metadata:
  name: credentials
  namespace: web-test
stringData:
  s3.accessKey: "access-key-value"
  s3.secretKey: "secret-key-value"
---
apiVersion: apps/v1beta1
kind: Deployment
metadata:
  name: web-test
  namespace: web-test
spec:
  template:
    metadata:
      labels:
        app: web-test
    spec:
      containers:
      - name: web-test
        image: web-test/web:latest-release
```

```
            args: ["-credentials-dir", "/credentials"]
            volumeMounts:
            - name: credentials
              mountPath: /credentials
              readOnly: true
        volumes:
        - name: credentials
          secret:
            secretName: credentials
```

Поддерживается следующие ключи:

Ключ	Значение
`s3.accessKey`	Имя пользователя хранилища S3 (`access key`)
`s3.secretKey`	Пароль пользователя хранилища S3 (`secret key`)

3.5. Загрузка файлов в хранилище S3

В приложении доступна настройка автоматического сохранения всех журналов (лог-файлов) браузерных сессий и записанных видеофайлов в S3-совместимое хранилище. Такой тип хранилища поддерживается большинством публичных облачных платформ.

Для создания собственного S3-совместимого хранилища поддерживается использование инструмента Minio. Для включения поддержки S3 выполните следующие действия:

1. Создайте S3-контейнер согласно инструкции для выбранной облачной платформы.

   В примере в п.2. имя контейнера – `web-test`. При создании контейнера вы также получите пару ключей: `access key` и `secret key` для доступа к контейнеру.

   Также потребуется информация о поддерживаемой версии S3-протокола (обычно указано в документации).

2. Создайте конфигурационный файл `service.json` со следующим содержимым:

```
$ cat service.json{
    "s3": {
        "endpoint": "https://storage.googleapis.com",
        "bucketName": "web-test",
        "version": "S3v2"
    }
}
```

   где: `bucketName` – имя контейнера.

3. Сохраните файл `service.json` в Kubernetes ConfigMap и смонтируйте его в виде файла в контейнер приложения «Сфера: Тестирование Веб», а затем укажите путь до файла с помощью флага `-config-file`, как показано в разделе «Дополнительные настройки приложения».

4. Создайте Kubernetes секрет для хранения пары S3-ключей, смонтируйте его как каталог в контейнер приложения «Сфера: Тестирование Веб», а затем укажите путь до этого каталога при помощи флага `–credentials-dir`, как показано в разделе «Хранение внешних учетных записей».

   Секрет с S3:

```
apiVersion: v1
kind: Secret
  metadata:
    name: credentials
    namespace: web-test
  stringData:
    s3.accessKey: "access-key-value"
    s3.secretKey: "secret-key-value"
```

По умолчанию файлы сохраняются в S3-контейнер в следующей последовательности:

```
\---my-bucket
    \---- <session-id>
          |---- video.mp4
          |---- session.log
```

«Сфера: Тестирование Веб» также позволяет сохранять файлы в S3-контейнер в виде произвольной иерархии, используя шаблон S3-ключа и подставляя в него текущие значения, относящиеся к Selenium-сессии. Полный шаблон S3-ключа выглядит следующим образом:

```
$quota/$browserName/$browserVersion/$platformName/$sessionId
```

Шаблон S3-ключа по умолчанию `$sessionId`.

Переменные, поддерживаемые в шаблоне S3-ключа:

Переменная	Значение
`$sessionId`	Идентификатор Selenium-сессии
`$browserName`	Имя браузера
`$browserVersion`	Версия браузера
`$platformName`	Имя платформы (операционной системы)
`$date`	Текущая дата в формате `ГГГГ-ММ-ДД`
`$quota`	Имя пользователя, переданное в Selenium URL

Полученный S3-ключ будет использоваться как «каталог» для сохранения записанных лог-файлов и видеофайлов в S3-контейнер.

Значение шаблона S3-ключа можно изменить с помощью конфигурационного файла `service.json`:

```
{
    "s3": {
        // Остальные настройки S3...
        "keyPattern":
"$quota/$browserName/$browserVersion/$platformName/$sessionId"
  // Пользовательский шаблон S3 ключа
```

```
        },
        "images": {
            //...
        },
        "annotations": {
            //...
        }
    }
```

Для определения индивидуального шаблона S3-ключа для каждой браузерной сессии используйте функцию `s3KeyPattern` (подробнее см. Руководство пользователя для продукта «Сфера: Тестирование Веб»).

После внесений изменений в ConfigMap, содержащий файл `service.json`, или секрет, содержащий пару ключей для S3, все настройки применяются автоматически, перезагрузка приложения не требуется.

3.6. Использование внутреннего хранилища Docker-образов

По умолчанию приложение скачивает системные Docker-образы (webtest/defender, webtest/logger и так далее) из общедоступного хранилища Docker-образов из сети Интернет.

Если требуется скачивать Docker-образы из внутреннего корпоративного или персонального хранилища (private Docker registry), выполните следующие действия:

1. Настройте Kubernetes для работы с внутренним хранилищем:

```
$ kubectl create secret docker-registry my-registry.example.com --
docker-server=my
-registry.example.com --docker-username=some-user --docker-
password=registry
-password --docker-email=some-user@example.com
$ kubectl patch serviceaccount default -p '{"imagePullSecrets":
[{"name": "myregistry.example.com"}]}' # Use correct service account
name here
```

2. Скопируйте требуемые образы браузеров во внутреннее хранилище, например:

```
selenoid/chrome:73.0 => my-registry.example.com/web-test/chrome:73.0
```

3. Обновите список браузеров, чтобы использовались новые образы, например:

```
{
  "chrome": {
      "default": "73.0",
      "versions": {
          "73.0": {
              "image": "my-registry.example.com/webtest/chrome:73.0",
              "port": "4444"
          },
          "72.0": {
```

```
                "image": "my-registry.example.com/webtest/chrome:72.0",
                "port": "4444"
            }
        }
    }
}
```

4. Скопируйте требуемую версию служебных образов приложения «Сфера: Тестирование Веб» во внутреннее хранилище:

```
webtest/webtest-video-recorder:1.3.4 => my-registry.example.com/
webtest/webtest-videorecorder:1.3.4
webtest/defender:1.3.4 => my-
registry.example.com/webtest/defender:1.3.4
webtest/logger => my-registry.example.com/webtest/logger:1.3.4
```

5. Переопределите используемые системные образы приложения в файле `service.json`:

```
$ cat service.json{
    "images": {
        "videoRecorder": {
            "image": "my-registry.example.com/webtest/video-
recorder:latest-release"
        },
        "defender": {
            "image": "my-registry.example.com/webtest/defender:latest-
release"
        },
        "logger": {
            "image": "my-registry.example.com/webtest/logger:latest-
release"
        }
    }
}
```

Если в этом файле уже присутствует конфигурация S3 – допишите в этот файл новый ключ `images`.

6. Сохраните `service.json` в Kubernetes ConfigMap, смонтируйте его как файл в контейнер приложения и укажите путь до файла с помощью флага `–config-file`.

   При внесении изменений в `service.json` в уже смонтированный ConfigMap изменения применяются автоматически, перезагрузка приложения не требуется.

7. Скопируйте основные образы приложения во внутреннее хранилище:

```
webtest/web:1.3.4 => my-registry.example.com/webtest/web:1.3.4
webtest/webtest-api:1.3.4 => my-registry.example.com/webtest/wettest-
api:1.3.4
```

```
webtest/selenoid-ui:1.6.5 => my-registry.example.com/webtest/selenoid-ui:1.6.5
```

8. Используйте основные образы приложения в Kubernetes YAML-манифестах, предназначенных для старта приложения.

Глоссарий

В документе используются следующие термины и сокращения:

Термин / Аббревиатура	Определение
ИС	Информационная система
Контейнер	Программный процесс, изолированный от других программных процессов средствами контейнеризации
Контейнеризация	Легковесная технология изоляции программных процессов, построенная на низкоуровневых возможностях ядра операционной системы Linux Позволяет изолировать процессы по процессорным ядрам, оперативной памяти, сетевым подключениям, файловой системе
Образ контейнера	Архив специального формата, содержащий исполняемую программу, а также полный набор зависимостей (операционную систему, системные утилиты, библиотеки и другие данные), необходимых для корректной работы
ОС	Операционная система
ПО	Программное обеспечение
Секрет (Secret)	Сущность Kubernetes , позволяющая хранить набор ключей и значений в закрытом виде и подключать их к подам В основном используется для безопасного хранения данных учетных записей

Вернуться в начало документа