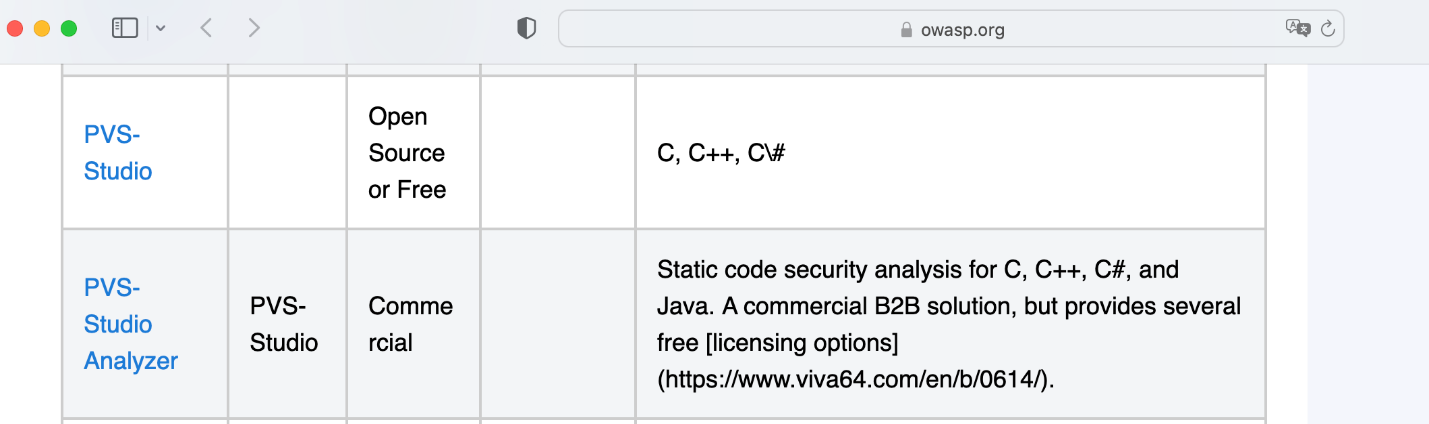
**1. Провести исследования доступных статических анализаторов для выбранного командой стека разработки (большой, но не всеобъемлющий [список](**[**https://owasp.org/www-community/Source\_Code\_Analysis\_Tools)**](https://owasp.org/www-community/Source_Code_Analysis_Tools))**)**

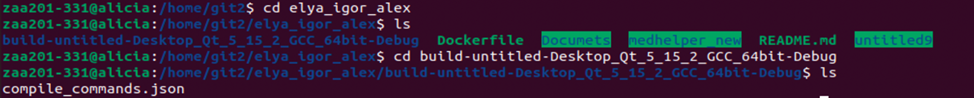


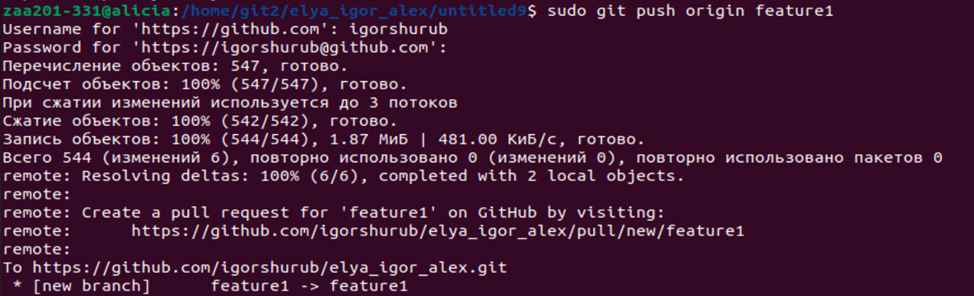
**2. Выбрать минимум два из следующих видов статических анализаторов:**

1. **анализатор качества кода (пример - Checkstyle)**
2. **анализатор безопасности кода (Пример - Gosec)**
3. **средство поиска учетных данных/секретов (Пример - Trufflehog).**

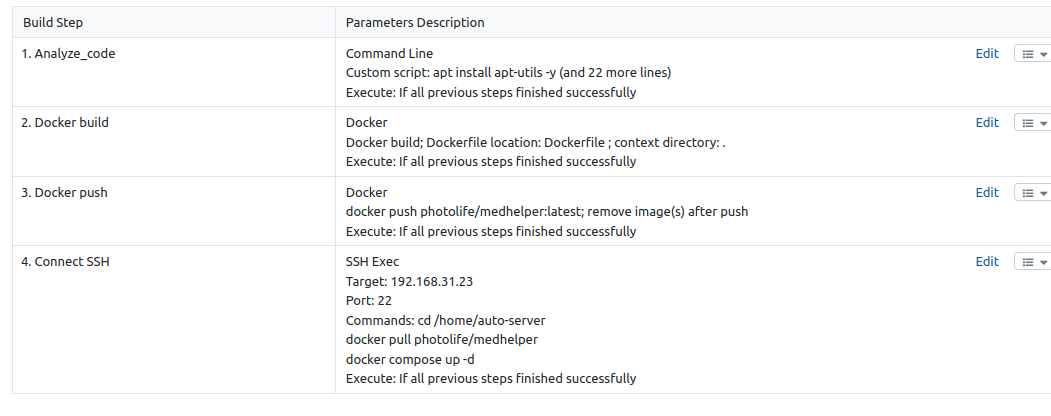
**Анализатор PVS‑Studio** выявляет не только опечатки, мёртвый код и другие ошибки, но и потенциальные уязвимости (Static Application Security Testing, SAST). Анализатор отображает предупреждения на Common Weakness Enumeration, SEI CERT Coding Standards, а также поддерживает стандарт MISRA.

3. Добавить в построенном пайплайне TeamCity шаги, на которых код проверяется статическими анализаторами и, в случае обнаружения проблем максимальной критичности для выбранного анализатора (для анализатора безопасности - CRITICAL, для анализатора качества кода - BLOCKER, для средства поиска секретов - любой обнаруженный пароль), билд завершается с ошибкой.

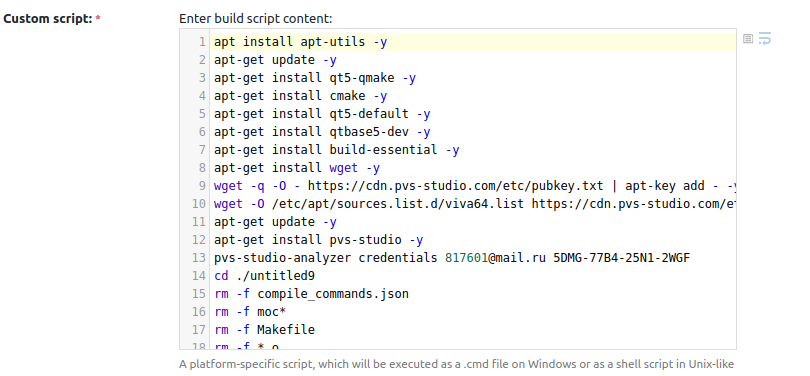




Добавим шаг, в котором будет происходить процесс анализа серверного кода:



Рассмотрим шаг «Analize\_code »:



Разберем подробнее custom script:

#Установим apt-utils и обновим индекс пакетов:

apt install apt-utils -y

apt-get update -y

#Установим Qt и компиляторы

apt-get install qt5-qmake -y

apt-get install cmake -y

apt-get install qt5-default -y

apt-get install qtbase5-dev -y

apt-get install build-essential -y

apt-get install wget -y

#Установим непосредственно pvs-studio

wget -q -O - https://cdn.pvs-studio.com/etc/pubkey.txt | apt-key add - -y

wget -O /etc/apt/sources.list.d/viva64.list https://cdn.pvs-studio.com/etc/viva64.list

apt-get update -y

apt-get install pvs-studio -y

#Добавим лицензию pvs-studio

pvs-studio-analyzer credentials 817601@mail.ru 5DMG-77B4-25N1-2WGF

#Перейдем в папку сервера:

cd ./untitled9

#Удалим старые сборки и старые версии файлов:

rm -f compile\_commands.json

rm -f moc\*

rm -f Makefile

rm -f \*.o

rm -f untitled9.pro

rm -f strace\_out

rm -f pvs.log

rm -f pvs\*.tasks

#Проведем трассировку компиляции (соберем проект):

qmake -o Makefile untitled.pro

make clean

pvs-studio-analyzer trace -- make

rm -f moc\*

#Удалим старую директорию для сохранения отчета и создадим новую:

rm -R ./report

mkdir ./report

#Запустим анализ нашего сервера:

pvs-studio-analyzer analyze -l /root/.config/PVS-Studio/PVS-Studio.lic -o pvs.log –disableLicenseExpirationCheck

#Сохраним отчет в формате .html в созданной папке:

plog-converter -a GA:1,2 --renderTypes fullhtml -t tasklist -o ./report/ pvs.log

#Заменим «фальшивые» err просто на 1, чтобы корректно найти критические ошибки:

sed "s|pvs-studio.com/en/docs/warnings/ 1 err Help: The documentation for all analyzer warnings is available here: https://pvs-studio.com/en/docs/warnings/.|1|" ./report/pvs.tasks > ./report/pvs1.tasks

sed "s|err Renew|1|" ./report/pvs1.tasks > ./report/pvs2.tasks

#Найдем err в отчете (если err будет найден, то код – 0, иначе – 1):

grep -r 'err' ./report/pvs2.tasks

#Выведем в случае обнаружения err в отчете код 1 (если ошибок нет, то 0):

if [ $? -eq 0 ]

then

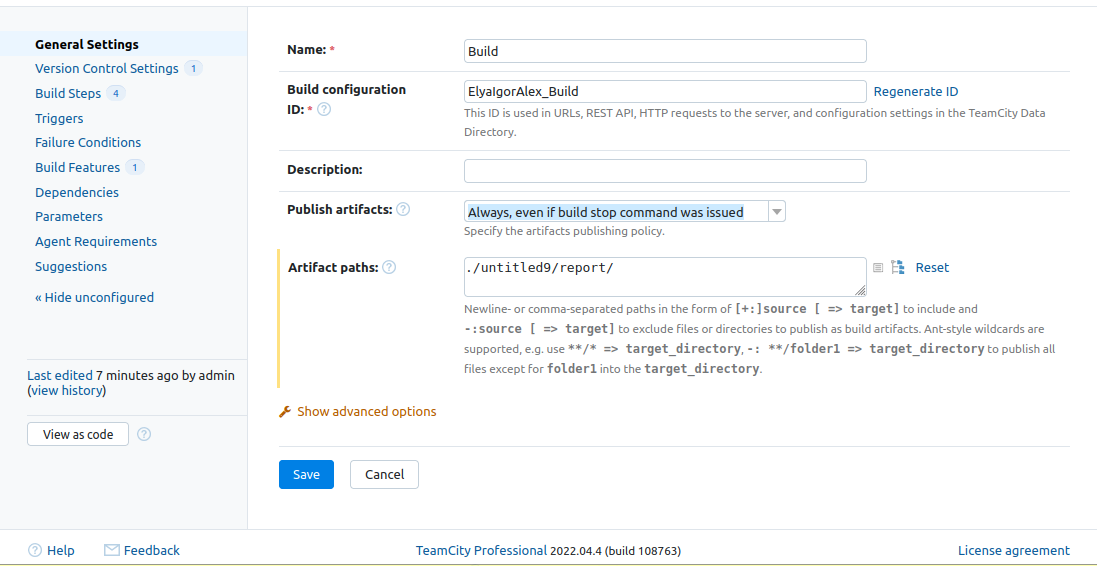
exit 1

else

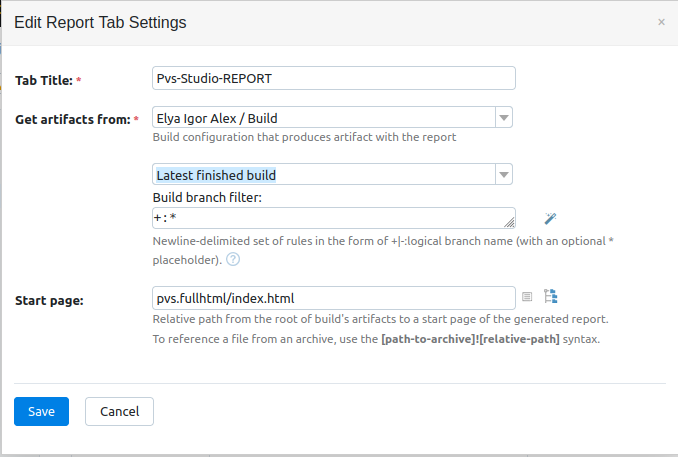
exit 0

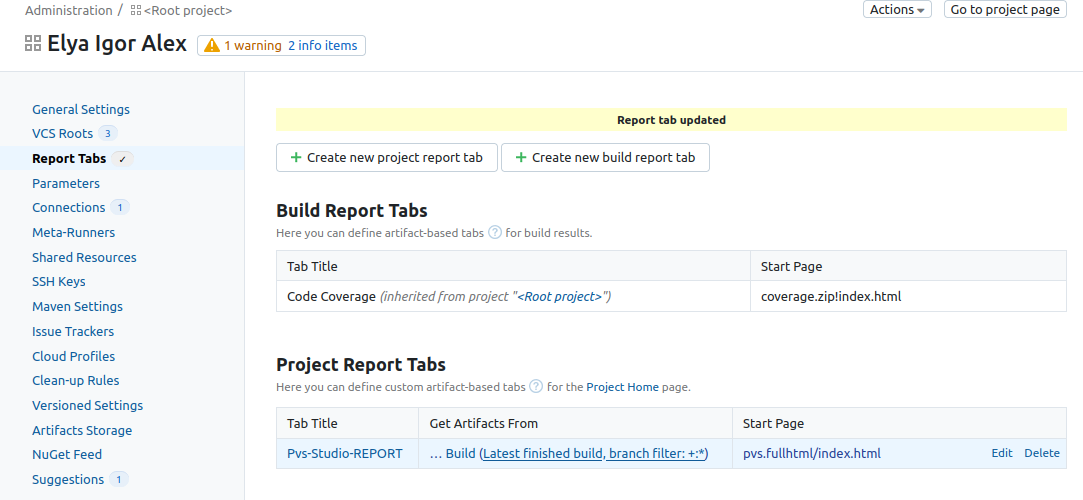
fi

После создания шага настроим пути Arthifact, то есть пропишем в них путь созданной папки report, чтобы после успешного выполнения сборки отчёт анализатора будет доступен в артефактах:

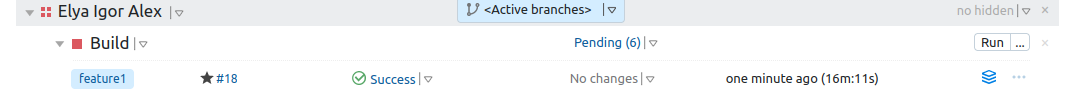


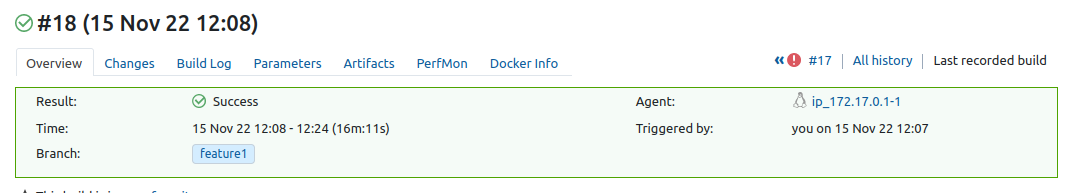
Также сделаем так, чтобы отчёт анализатора отображался на специальной вкладке отчёта о сеансе сборки. Для этого необходимо перейти в настройки проекта, открыть 'Report Tabs' и добавить новую вкладку отчёта сборки ('Create new build report tab').

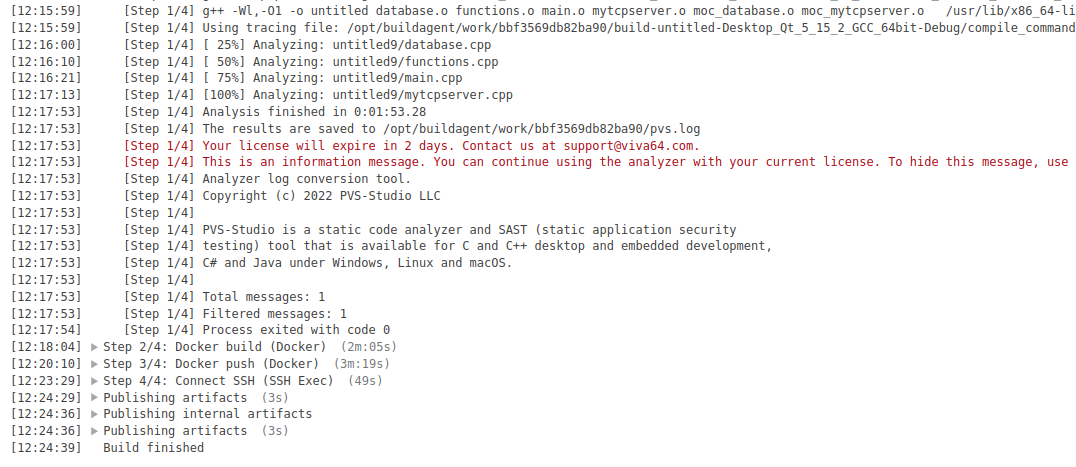




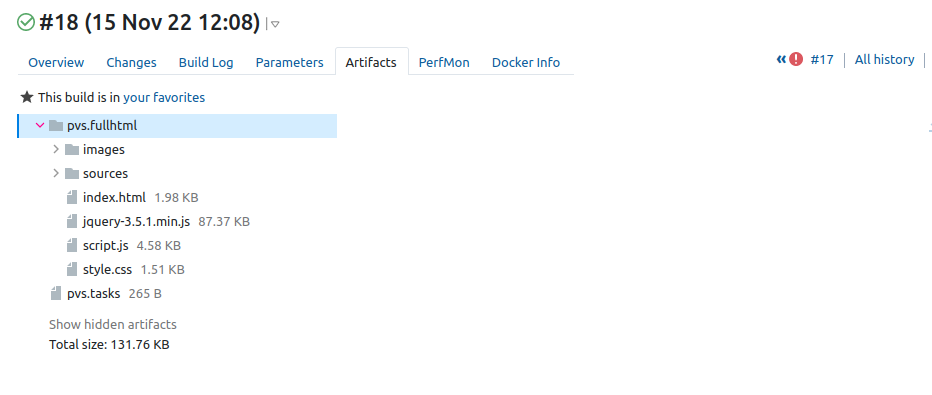
Зафиксируем успешную сборку из feature1:

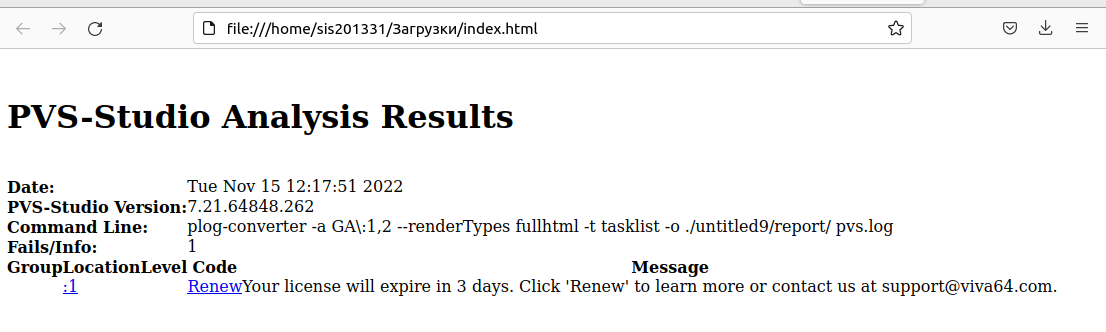






Посмотрим созданные отчеты с анализом кода сервера:







Как видим, ошибок не обнаружено, лишь предупреждение, что лицензия закончится через 3 дня

Далее сделаем код уязвимым и вновь запустим наш сборку с анализатором.

Теперь зафиксируем сборку с ошибками из ветки feature4:

