

Отчёт по установке систем taiga, gogs и jenkins

ПРЕДВАРИТЕЛЬНЫЕ ШАГИ

Установим sudo:

```
apt update
apt install sudo
```

Добавим пользователя в sudo:

```
usermod -aG sudo user
```

Перезапустим машину.

Установим Docker:

<https://docs.docker.com/engine/install/debian/>

```
sudo apt-get remove docker docker-engine docker.io containerd runc
sudo apt-get update
sudo apt-get install \
    apt-transport-https \
    ca-certificates \
    curl \
    gnupg \
    lsb-release

curl -fsSL https://download.docker.com/linux/debian/gpg | sudo gpg --dearmor -o
/usr/share/keyrings/docker-archive-keyring.gpg
echo \
    "deb [arch=amd64 signed-by=/usr/share/keyrings/docker-archive-keyring.gpg]
https://download.docker.com/linux/debian \
    $(lsb_release -cs) stable" | sudo tee /etc/apt/sources.list.d/docker.list > /dev/null
sudo apt-get update
sudo apt-get install docker-ce docker-ce-cli containerd.io
sudo usermod -aG docker user
```

Установим Docker-compose:

<https://www.digitalocean.com/community/tutorials/how-to-install-docker-compose-on-debian-10-ru>

```
sudo curl -L https://github.com/docker/compose/releases/download/1.25.3/docker-compose-`uname
-s`-`uname -m` -o /usr/local/bin/docker-compose
sudo chmod +x /usr/local/bin/docker-compose
```

Перезапустим машину.

ОПИСАНИЕ СЕРВИСОВ

Все сервисы получены с помощью официальных образов.

Taiga – Система управления задачами работает на порту 9000

Mysql – База данных (порт 3307)

Gogs – Система управления репозиториями (порты 3000 и 2222)

Jenkins – Система непрерывной интеграции (порты 8080 и 50000)

ОПИСАНИЯ DOCKERFILE

Все dockerfile получены с официальных сайтов

Dockerfile для mysql (открытые порты 3306, 33060)

```
Образ берём официальный debian:buster-slim
FROM debian:buster-slim
```

сначала добавляем нашего пользователя и группу, чтобы убедиться, что их идентификаторы назначаются последовательно, независимо от того, какие зависимости добавляются

```
RUN groupadd -r mysql && useradd -r -g mysql mysql
```

Устанавливаем нужные для работы приложения

```
RUN apt-get update && apt-get install -y --no-install-recommends gnupg dirmngr && rm -rf /var/lib/apt/lists/*
```

Записываем в переменную окружения версию gosu

```
ENV GOSU_VERSION 1.12
```

Устанавливаем приложения

```
RUN set -eux; \
    savedAptMark="$(apt-mark showmanual)"; \
    apt-get update; \
    apt-get install -y --no-install-recommends ca-certificates wget; \
    rm -rf /var/lib/apt/lists/*; \
    dpkgArch="$(dpkg --print-architecture | awk -F- '{ print $NF }')"; \
    wget -O /usr/local/bin/gosu \
"https://github.com/tianon/gosu/releases/download/$GOSU_VERSION/gosu-$dpkgArch"; \
    wget -O /usr/local/bin/gosu.asc \
"https://github.com/tianon/gosu/releases/download/$GOSU_VERSION/gosu-$dpkgArch.asc"; \
    export GNUPGHOME="$(mktemp -d)"; \
    gpg --batch --keyserver hkp://keys.openpgp.org --recv-keys \
B42F6819007F00F88E364FD4036A9C25BF357DD4; \
    gpg --batch --verify /usr/local/bin/gosu.asc /usr/local/bin/gosu; \
    gpgconf --kill all; \
    rm -rf "$GNUPGHOME" /usr/local/bin/gosu.asc; \
    apt-mark auto '.*' > /dev/null; \
    [ -z "$savedAptMark" ] || apt-mark manual $savedAptMark > /dev/null; \
    apt-get purge -y --auto-remove -o APT::AutoRemove::RecommendsImportant=false; \
    chmod +x /usr/local/bin/gosu; \
    gosu --version; \
    gosu nobody true
```

создаём директорию

```
RUN mkdir /docker-entrypoint-initdb.d
```

Устанавливаем дополнительные модули

```
RUN apt-get update && apt-get install -y --no-install-recommends \
# for MYSQL_RANDOM_ROOT_PASSWORD
    pwgen \
# for mysql_ssl_rsa_setup
    openssl \
# FATAL ERROR: please install the following Perl modules before executing
/usr/local/mysql/scripts/mysql_install_db:
# File::Basename
# File::Copy
# Sys::Hostname
# Data::Dumper
    perl \
```

```

# install "xz-utils" for .sql.xz docker-entrypoint-initdb.d files
    xz-utils \
    && rm -rf /var/lib/apt/lists/*

RUN set -ex; \
# gpg: key 5072E1F5: public key "MySQL Release Engineering <mysql-build@oss.oracle.com>"
imported
    key='A4A9406876FCBD3C456770C88C718D3B5072E1F5'; \
    export GNUPGHOME="$(mktemp -d)"; \
    gpg --batch --keyserver ha.pool.sks-keyservers.net --recv-keys "$key"; \
    gpg --batch --export "$key" > /etc/apt/trusted.gpg.d/mysql.gpg; \
    gpgconf --kill all; \
    rm -rf "$GNUPGHOME"; \
    apt-key list > /dev/null

Записываем переменные окружения версии
ENV MYSQL_MAJOR 8.0
ENV MYSQL_VERSION 8.0.25-1debian10

Формируем список пакетов
RUN echo 'deb http://repo.mysql.com/apt/debian/ buster mysql-8.0' >
/etc/apt/sources.list.d/mysql.list

# the "/var/lib/mysql" stuff here is because the mysql-server postinst doesn't have an
explicit way to disable the mysql_install_db codepath besides having a database already
"configured" (ie, stuff in /var/lib/mysql/mysql)
# also, we set debconf keys to make APT a little quieter
RUN { \
    echo mysql-community-server mysql-community-server/data-dir select ''; \
    echo mysql-community-server mysql-community-server/root-pass password ''; \
    echo mysql-community-server mysql-community-server/re-root-pass password ''; \
    echo mysql-community-server mysql-community-server/remove-test-db select false;
\
} | debconf-set-selections \
&& apt-get update \
&& apt-get install -y \
    mysql-community-client="${MYSQL_VERSION}" \
    mysql-community-server-core="${MYSQL_VERSION}" \
&& rm -rf /var/lib/apt/lists/* \
&& rm -rf /var/lib/mysql && mkdir -p /var/lib/mysql /var/run/mysqld \
&& chown -R mysql:mysql /var/lib/mysql /var/run/mysqld \
# ensure that /var/run/mysqld (used for socket and lock files) is writable regardless of the
UID our mysqld instance ends up having at runtime
&& chmod 1777 /var/run/mysqld /var/lib/mysql

Монтируем директорию для Mysql
VOLUME /var/lib/mysql

Копируем из config в /etc/mysql
COPY config/ /etc/mysql/
Копируем файл в контейнер
COPY docker-entrypoint.sh /usr/local/bin/
Для обратной совместимости выполняем команду
RUN ln -s usr/local/bin/docker-entrypoint.sh /entrypoint.sh
Открываем точку доступа
ENTRYPOINT ["docker-entrypoint.sh"]
Открываем порты 3306 и 33060
EXPOSE 3306 33060
Запускаемое приложение по умолчанию - mysqld
CMD ["mysqld"]

```

Dockerfile для gogs (порты 22, 3000)

<https://github.com/gogs/gogs/blob/v0.12.0/Dockerfile>

```
Образ golang:alpine3.11 назовём binarybuilder
FROM golang:alpine3.11 AS binarybuilder
Установим нужные для работы приложения
RUN apk --no-cache --no-progress add --virtual \
    build-deps \
    build-base \
    git \
    linux-pam-dev

Рабочая директория внутри контейнера: /gogs.io/gogs
WORKDIR /gogs.io/gogs
Копируем все из внешних файлов в рабочую текущую директорию
COPY . .
Собираем приложение
RUN make build-no-gen TAGS="cert pam"

FROM alpine:3.11
Копируем из репозитория в /usr/sbin/gosu
ADD https://github.com/tianon/gosu/releases/download/1.11/gosu-amd64 /usr/sbin/gosu
Создаём приложение и выдаём полномолчия
RUN chmod +x /usr/sbin/gosu \
    && echo http://dl-2.alpinelinux.org/alpine/edge/community/ >> /etc/apk/repositories \
    && apk --no-cache --no-progress add \
    bash \
    ca-certificates \
    curl \
    git \
    linux-pam \
    openssh \
    s6 \
    shadow \
    socat \
    tzdata \
    rsync

В переменную окружения добавляем значение:/data/gogs
ENV GOGS_CUSTOM /data/gogs

Копируем данные в контейнер
COPY docker/nsswitch.conf /etc/nsswitch.conf
В качестве рабочей директории /app/gogs
WORKDIR /app/gogs
Копируем данные в /app/gogs/docker
COPY docker ./docker
Используем прошлый образ по названию binarybuilder и из него
Копируем /gogs.io/gogs/gogs в текущую рабочую директорию
COPY --from=binarybuilder /gogs.io/gogs/gogs .

Запускаем finalize.sh
RUN ./docker/finalize.sh

Конфигурируем docker container
Монтируем директории
VOLUME ["/data", "/backup"]
Открываем 22 и 3000 порты
EXPOSE 22 3000

В качестве точки входа (обязательного запуска приложения): /app/gogs/docker/start.sh
ENTRYPOINT ["/app/gogs/docker/start.sh"]
Первоначальная команда для запуска контейнера (может быть опущена)
CMD ["/bin/s6-svscan", "/app/gogs/docker/s6/"]
```

Примечание: у jenkins и taiga Dockerfile скрыты - их нет в публичном доступе (только частично на dockerhub)

Часть Dockerfile для jenkins (Которая доступна в jenkins/jenkins:lts-alpine) (порты 8080, 50000)

<https://hub.docker.com/layers/jenkins/jenkins/lts-alpine/images/sha256-465b93777cd68a83992adc8fd4d1130fbb2db1fd7992abbd10013e2a11377c5?context=explore>

```
CMD ["/bin/sh"]
Пишем необходимые переменные окружения
ENV LANG=en_US.UTF-8 LANGUAGE=en_US:en LC_ALL=en_US.UTF-8
/bin/sh -c apk add --no-cache
ENV JAVA_VERSION=jdk8u292-b10
/bin/sh -c set -eux;
ENV JAVA_HOME=/opt/java/openjdk
PATH=/opt/java/openjdk/bin:/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin
/bin/sh -c apk add --no-cache
Прописываем аргументы
ARG user=jenkins
ARG group=jenkins
ARG uid=1000
ARG gid=1000
ARG http_port=8080
ARG agent_port=50000
ARG JENKINS_HOME=/var/jenkins_home
ARG REF=/usr/share/jenkins/ref
Прописываем переменные окружения
ENV JENKINS_HOME=/var/jenkins_home
ENV JENKINS_SLAVE_AGENT_PORT=50000
ENV REF=/usr/share/jenkins/ref
|6 agent_port=50000 gid=1000 group=jenkins http_port=8080
VOLUME [/var/jenkins_home]
|6 agent_port=50000 gid=1000 group=jenkins http_port=8080
ARG JENKINS_VERSION
ENV JENKINS_VERSION=2.289.1
ARG JENKINS_SHA=e5688a8f07cc3d79ba3afa3cab367d083dd90daab77cebd461ba8e83a1e3c177
ARG JENKINS_URL=https://repo.jenkins-ci.org/public/org/jenkins-ci/main/jenkins-war/2.289.1/jenkins-war-2.289.1.war
|8 JENKINS_SHA=70f9cc6ff1ac59aeeb831b980709a9ddb0ee70d216ee50625a8508b9840f75f2
JENKINS_URL=https://repo.jenkins-ci.org/public/org/jenkins-ci/main/jenkins-war/2.289.1/jenkins-war-2.289.1.war agent_port=50000 gid=1000
ENV JENKINS_UC=https://updates.jenkins.io
ENV JENKINS_UC_EXPERIMENTAL=https://updates.jenkins.io/experimental
ENV JENKINS_INCREMENTALS_REPO_MIRROR=https://repo.jenkins-ci.org/incrementals
|8 JENKINS_SHA=70f9cc6ff1ac59aeeb831b980709a9ddb0ee70d216ee50625a8508b9840f75f2
JENKINS_URL=https://repo.jenkins-ci.org/public/org/jenkins-ci/main/jenkins-war/2.289.1/jenkins-war-2.289.1.war agent_port=50000 gid=1000
ARG PLUGIN_CLI_URL=https://github.com/jenkinsci/plugin-installation-manager-tool/releases/download/2.9.0/jenkins-plugin-manager-2.9.0.jar
|9 JENKINS_SHA=70f9cc6ff1ac59aeeb831b980709a9ddb0ee70d216ee50625a8508b9840f75f2
JENKINS_URL=https://repo.jenkins-ci.org/public/org/jenkins-ci/main/jenkins-war/2.289.1/jenkins-war-2.289.1.war PLUGIN_CLI_URL=https://github.com/jenkinsci/plugin-installation-manager-tool/releases/download/2.9.0/jenkins-plugin-manager-2.9.0.jar
agent_port=50000
Открываем порты 8080 и 50000
EXPOSE 8080
EXPOSE 50000
ENV COPY_REFERENCE_FILE_LOG=/var/jenkins_home/copy_reference_file.log

USER jenkins
```

```
Копируем файлы в контейнер
COPY file:2a8e84f82e3646a38efbd5b89833d9be6e60188df8937ed38ab2f20901f5064d in
/usr/local/bin/jenkins-support
COPY file:b1d9bec18d388d962c78be035aa3bf9d207ff48dee56a41119723dca817df187 in
/usr/local/bin/jenkins.sh
COPY file:dc942ca949bb159f81bbc954773b3491e433d2d3e3ef90bac80ecf48a313c9c9 in /bin/tini
COPY file:5a7967a89c74c1d95eeabf80b4f38d19348425d2e418ac42b44ec9fb73dbb4c8 in /bin/jenkins-
plugin-cli
Открываем точка входа
ENTRYPOINT ["/sbin/tini" "--" "/usr/local/bin/jenkins.sh"]
Копируем файл
COPY file:e8cf7c918bffcfc8eafd97b7c3d3003143dd412c91b710b209d509af27cac7c27 in
/usr/local/bin/install-plugins.sh
```

Часть Dockerfile для taigaio/taiga-back (порты 8080, 50000)

<https://hub.docker.com/layers/taigaio/taiga-back/6.2.1/images/sha256-8d59f985fea3d0a0a261a68a0484345bf20db502543c94bde58e8db03c5a3353?context=explore>

```
Sh - инструкция по умолчанию
CMD ["/bin/sh"]
Записываем переменную окружения
ENV LANG=en_US.UTF-8 LANGUAGE=en_US:en LC_ALL=en_US.UTF-8
/bin/sh -c apk add --no-cache
ENV JAVA_VERSION=jdk8u292-b10
/bin/sh -c set -eux;
ENV JAVA_HOME=/opt/java/openjdk
PATH=/opt/java/openjdk/bin:/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin
/bin/sh -c apk add --no-cache
Пишем аргументы
ARG user=jenkins
ARG group=jenkins
ARG uid=1000
ARG gid=1000
ARG http_port=8080
ARG agent_port=50000
ARG JENKINS_HOME=/var/jenkins_home
ARG REF=/usr/share/jenkins/ref
ENV JENKINS_HOME=/var/jenkins_home
ENV JENKINS_SLAVE_AGENT_PORT=50000
ENV REF=/usr/share/jenkins/ref
agent_port=50000 gid=1000 group=jenkins http_port=8080
Монтируем директорию /var/jenkins_home
VOLUME ["/var/jenkins_home"]
|6 agent_port=50000 gid=1000 group=jenkins http_port=8080
ARG JENKINS_VERSION
ENV JENKINS_VERSION=2.289.1
ARG JENKINS_SHA=e5688a8f07cc3d79ba3afa3cab367d083dd90daab77cebd461ba8e83a1e3c177
ARG JENKINS_URL=https://repo.jenkins-ci.org/public/org/jenkins-ci/main/jenkins-
war/2.289.1/jenkins-war-2.289.1.war
|8 JENKINS_SHA=70f9cc6ff1ac59aeeb831b980709a9ddb0ee70d216ee50625a8508b9840f75f2
JENKINS_URL=https://repo.jenkins-ci.org/public/org/jenkins-ci/main/jenkins-
war/2.289.1/jenkins-war-2.289.1.war agent_port=50000 gid=1000
ENV JENKINS_UC=https://updates.jenkins.io
ENV JENKINS_UC_EXPERIMENTAL=https://updates.jenkins.io/experimental
ENV JENKINS_INCREMENTALS_REPO_MIRROR=https://repo.jenkins-ci.org/incrementals
|8 JENKINS_SHA=70f9cc6ff1ac59aeeb831b980709a9ddb0ee70d216ee50625a8508b9840f75f2
JENKINS_URL=https://repo.jenkins-ci.org/public/org/jenkins-ci/main/jenkins-
war/2.289.1/jenkins-war-2.289.1.war agent_port=50000 gid=1000
ARG PLUGIN_CLI_URL=https://github.com/jenkinsci/plugin-installation-manager-
tool/releases/download/2.9.0/jenkins-plugin-manager-2.9.0.jar
|9 JENKINS_SHA=70f9cc6ff1ac59aeeb831b980709a9ddb0ee70d216ee50625a8508b9840f75f2
JENKINS_URL=https://repo.jenkins-ci.org/public/org/jenkins-ci/main/jenkins-
```

```
war/2.289.1/jenkins-war-2.289.1.war PLUGIN_CLI_URL=https://github.com/jenkinsci/plugin-
installation-manager-tool/releases/download/2.9.0/jenkins-plugin-manager-2.9.0.jar
agent_port=50000
Открываем порты 8080 и 50000
EXPOSE 8080
EXPOSE 50000
ENV COPY_REFERENCE_FILE_LOG=/var/jenkins_home/copy_reference_file.log
USER jenkins
Копируем файлы в контейнер
COPY file:2a8e84f82e3646a38efbd5b89833d9be6e60188df8937ed38ab2f20901f5064d in
/usr/local/bin/jenkins-support
COPY file:b1d9bec18d388d962c78be035aa3bf9d207ff48dee56a41119723dca817df187 in
/usr/local/bin/jenkins.sh
COPY file:dc942ca949bb159f81bbc954773b3491e433d2d3e3ef90bac80ecf48a313c9c9 in /bin/tini
COPY file:5a7967a89c74c1d95eeabf80b4f38d19348425d2e418ac42b44ec9fb73dbb4c8 in /bin/jenkins-
plugin-cli
Устанавливаем входную точку
ENTRYPOINT ["/sbin/tini" "--" "/usr/local/bin/jenkins.sh"]
COPY file:e8cf7c918bffcfc8eafd97b7c3d3003143dd412c91b710b209d509af27cac7c27 in
/usr/local/bin/install-plugins.sh
```

Часть Dockerfile для taigaio/taiga-front (порт 80)

<https://hub.docker.com/layers/taigaio/taiga-front/6.2.1/images/sha256-8f1bf97aac7a27af9cb50022b641763ccd0589cc242fbcd8fafba5551d8193ae?context=explore>

```
Sh – инструкция по умолчанию
CMD ["/bin/sh"]
Пропишем метаданные по тому, кто мейнтейнер
LABEL maintainer=NGINX Docker Maintainers
Пропишем переменные окружения по версиям
ENV NGINX_VERSION=1.19.9
ENV NJS_VERSION=0.5.3
ENV PKG_RELEASE=1
/bin/sh -c set -x
Копируем файлы в контейнер
COPY file:65504f71f5855ca017fb64d502ce873a31b2e0dec75297a8fb0a287f97acf92 in /
COPY file:0b866ff3fc1ef5b03c4e6c8c513ae014f691fb05d530257dfff07035c1b75da in /docker-
entrypoint.d
COPY file:0fd5fca330dcd6a7de297435e32af634f29f7132ed0550d342cad9fd20158258 in /docker-
entrypoint.d
COPY file:09a214a3e07c919af2fb2d7c749ccbc446b8c10eb217366e5a65640ee9edcc25 in /docker-
entrypoint.d
Установим точку входа
ENTRYPOINT ["/docker-entrypoint.sh"]
Откроем 80 порт
EXPOSE 80
CMD ["nginx" "-g" "daemon"]
Добавим метаданные по мейнтейнеру
LABEL maintainer=support@taiga.io
COPY file:423c72d215f533f442f5114e23d6ca0632289d001a937595cbae0b92dc37c804 in
/etc/nginx/conf.d/default.conf
COPY file:c7f19a2492cdd915de54ed8e5046cfb75504e5e231b007af659aa6253568cb25 in /
COPY file:36de697e3cd5a0c99cf47a64fd45cf47ff8c69cada26f6eeee7672c026db8f66 in /docker-
entrypoint.d/30_config_env_subst.sh
/bin/sh -c set -eux;
```

Часть Dockerfile для taigaio/taiga-events (порт 8888)

<https://hub.docker.com/layers/taigaio/taiga-events/6.2.1/images/sha256-b6cc51ebb0da6307f46012995a8c31cc570415f977d26d29cad58a038382846f?context=explore>

```
Инструкция по умолчанию - cmd
CMD ["/bin/sh"]
Запишем в переменные окружения версии
ENV NODE_VERSION=12.22.1
/bin/sh -c addgroup -g 1000
ENV YARN_VERSION=1.22.5
/bin/sh -c apk add --no-cache
Копируем файлы в контейнер
COPY file:238737301d47304174e4d24f4def935b29b3069c03c72ae8de97d94624382fce in /usr/local/bin/
Выставим точка входа
ENTRYPOINT ["docker-entrypoint.sh"]
CMD ["node"]
Добавим метаданные по мэнтейнеру
LABEL maintainer=support@taiga.io
COPY dir:ebf4953db5402cb53502a7817fec324d42b476f31e8102fff9ce1ea380dab208 in /taiga-events
Рабочий каталог теперь /taiga-events
WORKDIR /taiga-events
/bin/sh -c set -eux;
Открываем 8888 порт
EXPOSE 8888
ENTRYPOINT ["/docker/entrypoint.sh"]
```

Часть Dockerfile для taigaio/taiga-protected (порт 8003)

<https://hub.docker.com/layers/taigaio/taiga-protected/6.2.1/images/sha256-7f37d338dd05769bf468dfa0b3415f744306054687ce3dc9a5afd79aba168015?context=explore>

```
Инструкция по умолчанию - cmd
CMD ["bash"]
Записываем переменные окружения
ENV PATH=/usr/local/bin:/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin
ENV LANG=C.UTF-8
/bin/sh -c set -eux; apt-get
ENV GPG_KEY=0D96DF4D4110E5C43FBFB17F2D347EA6AA65421D
ENV PYTHON_VERSION=3.7.10
/bin/sh -c set -ex &&
/bin/sh -c cd /usr/local/bin &&
ENV PYTHON_PIP_VERSION=21.0.1
ENV PYTHON_GET_PIP_URL=https://github.com/pypa/get-pip/raw/29f37dbe6b3842ccd52d61816a3044173962ebef/public/get-pip.py
ENV PYTHON_GET_PIP_SHA256=e03eb8a33d3b441ff484c56a436ff10680479d4bd14e59268e67977ed40904de
/bin/sh -c set -ex; savedAptMark="$(apt-mark
Инструкция по умолчанию - Python3
CMD ["python3"]
Добавим метаданные по мэнтейнеру
LABEL maintainer=support@taiga.io
Переменные окружения
ENV DEBIAN_FRONTEND=noninteractive
ENV PYTHONUNBUFFERED=1
ENV PYTHONDONTWRITEBYTECODE=1
ENV PYTHONFAULTHANDLER=1
/bin/sh -c python -m venv
ENV
PATH=/opt/venv/bin:/usr/local/bin:/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin
```



```
Теперь рабочий каталог /taiga-protected
WORKDIR /taiga-protected
Запишем версию gosu в переменную окружения
ENV GOSU_VERSION=1.12
/bin/sh -c set -eux;
Открываем порт 8003
EXPOSE 8003
ENTRYPOINT ["/docker/entrypoint.sh"]
```

ОПИСАНИЕ DOCKER-COMPOSE.YML

Примечание: вместо localhost подставьте адрес вашей машины.

```
version: "3.6"
Переменные
x-environment:
  &default-back-environment
  # Database settings
  POSTGRES_DB: taiga
  POSTGRES_USER: taiga
  POSTGRES_PASSWORD: taiga
  POSTGRES_HOST: taiga-db
  TAIGA_ENABLE_WEBHOOKS: "True"
  WEBHOOKS_ENABLED: "True"

  # Taiga settings
  TAIGA_SECRET_KEY: "taiga-back-secret-key"
  TAIGA_SITES_DOMAIN: "localhost:9000"
  TAIGA_SITES_SCHEME: "http"

  # Rabbitmq settings
  # Should be the same as in taiga-async-rabbitmq and taiga-events-rabbitmq
  RABBITMQ_USER: taiga
  RABBITMQ_PASS: taiga

  # Telemetry settings
  ENABLE_TELEMETRY: "False"

Переменные для монтирования директории
x-volumes:
  &default-back-volumes
  - taiga-static-data:/taiga-back/static
  - taiga-media-data:/taiga-back/media

services:
  mysql:
    image: mysql:8
    container_name: mysql
    environment:
      MYSQL_DATABASE: gogs
      MYSQL_USER: gogs
      MYSQL_PASSWORD: gogs
      MYSQL_ROOT_PASSWORD: root
    ports:
      - "3307:3306"
      - "33070:33060"
Данные будут храниться в директории mysql
  volumes:
    - ./mysql:/var/lib/mysql
В случае сбоя – перезапускать сервис всегда
  restart: always
```

```
gogs:
  image: gogs/gogs:0.12
  container_name: gogs
  ports:
    - "3000:3000"
    - "2222:22"
```

Данные будут храниться в директории gogs

```
volumes:
  - ./gogs:/data
```

Обязателен mysql для него

```
links:
  - mysql
```

```
jenkins:
  image: jenkins/jenkins:lts-alpine
  container_name: jenkins
```

Jenkins требует привилегированного пользователя

```
user: root
privileged: true
ports:
  - "50000:50000"
  - "8080:8080"
```

Данные будут храниться в jenkins директории

```
volumes:
  - ./jenkins:/var/jenkins_home
```

Сервис БД для Taiga

```
taiga-db:
  image: postgres:12.3
  environment:
    POSTGRES_DB: taiga
    POSTGRES_USER: taiga
    POSTGRES_PASSWORD: taiga
  volumes:
    - taiga-db-data:/var/lib/postgresql/data
  networks:
    - taiga
```

Сервис backend для taiga

```
taiga-back:
  image: taigaio/taiga-back:latest
```

В качестве переменных окружения: переменные, которые указывали раньше

```
environment: *default-back-environment
volumes: *default-back-volumes
```

Зависимости

```
networks:
  - taiga
depends_on:
  - taiga-db
  - taiga-events-rabbitmq
  - taiga-async-rabbitmq
```

Сервис асинхронных запросов taiga

```
taiga-async:
  image: taigaio/taiga-back:latest
  entrypoint: ["/taiga-back/docker/async_entrypoint.sh"]
  environment: *default-back-environment
  volumes: *default-back-volumes
  networks:
    - taiga
  depends_on:
    - taiga-db
    - taiga-back
    - taiga-async-rabbitmq
```

Сервис брокер сообщений rabbitmq для taiga

```
taiga-async-rabbitmq:
  image: rabbitmq:3-management-alpine
```

```
environment:
  RABBITMQ_ERLANG_COOKIE: secret-erlang-cookie
  RABBITMQ_DEFAULT_USER: taiga
  RABBITMQ_DEFAULT_PASS: taiga
  RABBITMQ_DEFAULT_VHOST: taiga
volumes:
  - taiga-async-rabbitmq-data:/var/lib/rabbitmq
networks:
  - taiga
```

Сервис frontend для taiga

```
taiga-front:
  image: taigaio/taiga-front:latest
  environment:
    TAIGA_URL: "http://localhost:9000"
    TAIGA_WEBSOCKETS_URL: "ws://localhost:9000"
  networks:
    - taiga
```

Сервис событий для taiga

```
taiga-events:
  image: taigaio/taiga-events:latest
  environment:
    RABBITMQ_USER: taiga
    RABBITMQ_PASS: taiga
    TAIGA_SECRET_KEY: "taiga-back-secret-key"
  networks:
    - taiga
  depends_on:
    - taiga-events-rabbitmq
```

Сервис событий с брокером сообщений rabbitmq

```
taiga-events-rabbitmq:
  image: rabbitmq:3-management-alpine
  environment:
    RABBITMQ_ERLANG_COOKIE: secret-erlang-cookie
    RABBITMQ_DEFAULT_USER: taiga
    RABBITMQ_DEFAULT_PASS: taiga
    RABBITMQ_DEFAULT_VHOST: taiga
  volumes:
    - taiga-events-rabbitmq-data:/var/lib/rabbitmq
  networks:
    - taiga
```

Сервис по безопасности taiga

```
taiga-protected:
  image: taigaio/taiga-protected:latest
  environment:
    MAX_AGE: 360
    SECRET_KEY: "taiga-back-secret-key"
  networks:
    - taiga
```

Сервис по проксированию для taiga

```
taiga-gateway:
  image: nginx:1.19-alpine
  ports:
    - "9000:80"
  volumes:
    - ./config/taiga-gateway/taiga.conf:/etc/nginx/conf.d/default.conf
    - taiga-static-data:/taiga/static
    - taiga-media-data:/taiga/media
  networks:
    - taiga
  depends_on:
    - taiga-front
    - taiga-back
    - taiga-events
```

Монтируем точки, указанные выше для taiga

```
volumes:
  taiga-static-data:
  taiga-media-data:
  taiga-db-data:
  taiga-async-rabbitmq-data:
  taiga-events-rabbitmq-data:
```

Taiga сервисы должны работать в одной сети

```
networks:
  taiga:
```

Для taiga также нужен docker-compose-inits.yml

```
version: "3.6"

x-environment:
  &default-back-environment
  POSTGRES_DB: taiga
  POSTGRES_USER: taiga
  POSTGRES_PASSWORD: taiga
  POSTGRES_HOST: taiga-db
  TAIGA_SECRET_KEY: "taiga-back-secret-key"
  # these rabbitmq settings should be the same as
  # in taiga-rabbitmq and taiga-events services
  RABBITMQ_USER: taiga
  RABBITMQ_PASS: taiga
  CELERY_ENABLED: "False"

x-volumes:
  &default-back-volumes
  - taiga-static-data:/taiga-back/static
  - taiga-media-data:/taiga-back/media

services:
  taiga-manage:
    image: taigaio/taiga-back:latest
    environment: *default-back-environment
    depends_on:
      - taiga-db
    entrypoint: "python manage.py"
    volumes: *default-back-volumes
    networks:
      - taiga
```

РАЗВОРАЧИВАНИЕ СИСТЕМ

Запустите docker-compose:

```
docker-compose up
```

Настроим gogs, перейдём по адресу <http://localhost:3000>:

Gogs requires MySQL, PostgreSQL, SQLite3, MSSQL or TiDB.

Database Type *	MySQL ▼
Host *	mysql:3307
User *	gogs
Password *	••••
Database Name *	gogs

Please use INNODB engine with utf8_general_ci charset fo

Application General Settings

Datbase Type: MySQL

Host: mysql:3307 (либо вместо mysql прописываем хост машины)

User: gogs

Password: gogs

Database Name: gogs

Общие параметры Gogs

Имя приложения *	Gogs	Укажите здесь название вашей потрясающей организации!
Путь корня репозитория *	/data/git/gogs-repositories	Все сетевые репозитории Git будут сохранены в этой директории.
Пользователь *	git	У пользователя должен быть доступ к пути к корню репозитория и к запуску Gogs.
Домен *	localhost	Влияет на URL-адреса для клонирования по SSH.
SSH порт *	22	Номер порта, который использует SSH сервер. Оставьте пустым, чтобы отключить SSH. <input type="checkbox"/> Использовать встроенный SSH сервер
Порт HTTP *	3000	Номер порта, который приложение будет слушать.
URL приложения *	http://localhost:3000/	Этот параметр влияет на URL для клонирования по HTTP/HTTPS и на адреса в электронной почте.
Путь к журналу *	/app/gogs/log	

И добавим админа

Имя пользователя: root

Пароль: root

Расширенные настройки

Настройки службы электронной почты

Сервер и другие настройки служб

Настройки учётной записи администратора

Вам не обязательно создавать учётную запись администратора прямо сейчас, пользователь с ID = 1 получит доступ с правами администратора автоматически.

Имя пользователя

root

Пароль

....

Подтвердить пароль

....

Электронная почта администратора

nick_over@inbox.ru

Установить Gogs

Настроим Jenkins, перейдём по адресу <http://localhost:8080>

Получим ключ для администратора, выполним команду:

```
docker exec jenkins cat /var/jenkins_home/secrets/initialAdminPassword
```

И вставим его:

Getting Started

Unlock Jenkins

To ensure Jenkins is securely set up by the administrator, a password has been written to the log (**not sure where to find it?**) and this file on the server:

`/var/jenkins_home/secrets/initialAdminPassword`

Please copy the password from either location and paste it below.

Administrator password

.....

Continue

Устанавливаем suggested plugins:

Getting Started

Customize Jenkins

Plugins extend Jenkins with additional features to support many different needs.

Install suggested plugins

Install plugins the Jenkins community finds most useful.

Select plugins to install

Select and install plugins most suitable for your needs.

Создаём админа:

Getting Started

Create First Admin User

Имя пользователя:	<input type="text" value="root"/>
Пароль:	<input type="password" value="...."/>
Повторите пароль:	<input type="password" value="...."/>
Ф.И.О.:	<input type="text" value="root"/>
Адрес электронной почты:	<input type="text" value="nick_over@inbox.ru"/>

Имя пользователя: root

Пароль: root

Jenkins будет доступен по localhost:8080

Instance Configuration

Jenkins URL:

The Jenkins URL is used to provide the root URL for absolute links to various Jenkins resources. That means this value is required for proper operation of many Jenkins features including email notifications, PR status updates, and the BUILD_URL environment variable provided to build steps.

The proposed default value shown is **not saved yet** and is generated from the current request, if possible. The best practice is to set this value to the URL that users are expected to use. This will avoid confusion when sharing or viewing links.

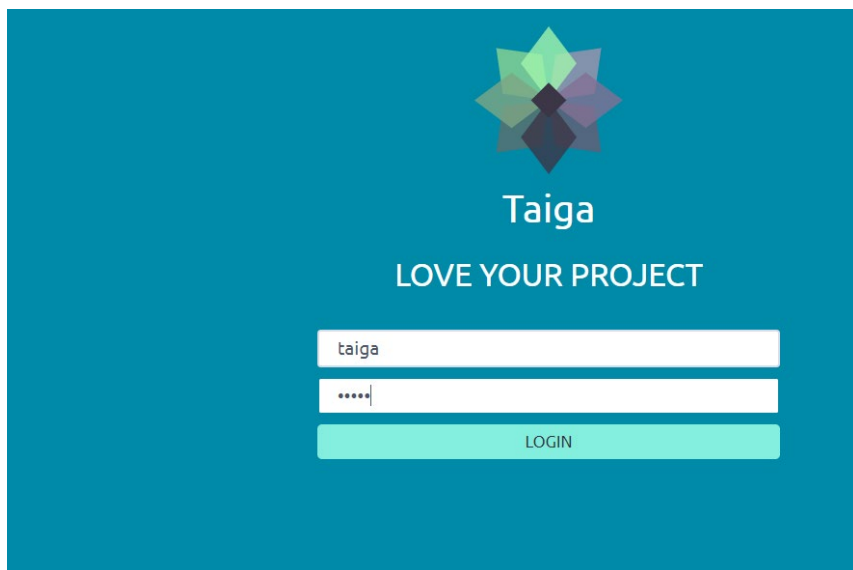
Настроим taiga, перейдём по адресу <http://localhost:9000>

Создадим админа, для этого запустим скрипт /config/taiga/create-super-user.sh и введём данные:

Username: taiga

Password: taiga

Теперь залогинимся с помощью этого аккаунта:




Пример настройки систем:

Создание репозитория

Новый репозиторий

Владелец *

 root

Имя репозитория *

demo-rep

Лучшие названия репозитория коротки, запоминаемы и уникальны.

Видимость

☐ Личный репозиторий

Описание

Описание репозитория. Максимальная длина 512 символов.

Доступные символы: 512

.gitignore

Выберите шаблоны .gitignore

Лицензия

Выберите файл лицензии

Readme ?

Default

☐ Инициализировать этот репозиторий выбранными файлами и шаблоном

Создать репозиторий

Отмена

Создаем локальные репозиторий и запустим в только что созданный demo-rep:

```
touch README.md
git init
git config --global user.name "Иван Иванов"
git config --global user.email nick\_over@inbox.ru
git add README.md
git commit -m "Первый коммит"
git remote add origin http://localhost:3000/root/demo-rep.git
git push -u origin master
```



Панель управления

Задачи

Запросы на слияние

Обзор



root / demo-rep

👁 Перестать следить

1

☆ В избранное

0

💬 Ответить

0

📁 Файлы

📌 Задачи 0

🔗 Запросы на слияние 0

📖 Вики

⚙ Настройки

Нет описания

📁 1 Коммиты

🌿 1 Ветки

📦 0 Релизы



Ветка: master ▾

demo-rep

Новый файл

Загрузить файл

HTTP

SSH

http://localhost:3000/root/



Иван Иванов

9275267ff2

Первый коммит

56 секунд назад



README.md

9275267ff2

Первый коммит

56 секунд назад



README.md



📁 Projects •

Create your first project

You don't have any projects yet. Create one now, invite your team and start using Taiga to the fullest.

NEW PROJECT

Create Project

Which template fits your project better?



SCRUM

Prioritize and solve your tasks in short time cycles.



KANBAN

Keep a constant workflow on independent tasks



DUPLICATE PROJECT

Start clean and keep your configuration



IMPORT PROJECT

Import your project from multiple platforms into Taiga

Kanban

Keep a constant workflow on independent tasks

New project details

demo

demo taiga project



This value is required.



PUBLIC PROJECT



PRIVATE PROJECT

BACK

CREATE PROJECT

Свяжем gogs и taiga:

<https://docs.taiga.io/changing-elements-status-via-commit-message.html>

копируем данные из settings/integrations/gogs

demo

Kanban

Search

Team

Settings

collapse menu

PROJECT

ATTRIBUTES

MEMBERS

PERMISSIONS

INTEGRATIONS

WEBHOOKS

GITHUB

GITLAB

BITBUCKET

GOGS

Gogs

Secret key

63c273178b0340c3a69e55b05bf3972e


Payload URL

http://localhost:9000/api/v1/gogs-hook?project=1

SAVE

Do you need help? Check out our support page!

Переходим в gogs репозиторий и его настройки

 root / demo-rep

👁 Следить 0

☆ В избранное 0

💬 Ответить 0

📁 Файлы

📌 Задачи 0

🔗 Запросы на слияние 0

📖 Вики

⚙ Настройки

Настройки

Опции

Сотрудничество

Ветки

Автоматическое обновление

Git хуки

Ключи развертывания

Автоматическое обновление

Webhooks позволяют внешним службам получать уведомления при возникновении определенных событий на Gogs. При возникновении указанных событий мы отправим запрос POST на каждый заданный вами URL. Узнать больше можно в нашем [Руководстве по Webhooks](#).

Add a new webhook:

Choose a type...

- Gogs
- Slack
- Discord
- Dingtalk

Выбираем gogs.

Далее вставляем скопированные данные:

Настройки

Опции

Сотрудничество

Ветки

Автоматическое обновление

Git хуки

Ключи развертывания

Добавить Webhook

Мы отправим запрос POST на указанный ниже URL с информацией о событиях. Можно также указать формат, в котором вы бы хотели получить данные (JSON, x-www-form-urlencoded, и т.д.). Дополнительную информацию можно найти в [Руководстве по Webhook](#).

URL обработчика *

http://localhost:9000/api/v1/gogs-hook?project=1

Тип содержимого

application/json

Secret

.....

Секрет будет отправлен как SHA256 HMAC контента в шестнадцатеричном виде в заголовке X-Gogs-Signature.

На какие события этот webhook должен срабатывать?

☒ Просто push событие.

☐ Мне нужно все.

☐ Позвольте мне выбрать то, что нужно.

☒ Активен

Подробности о событии, вызвавшем срабатывание хука, также будут предоставлены.

Добавить Webhook

Создаём новую задачу в taiga:

demo-task

Add tag +

task 1

0 Attachments +

Drop attachments here!

CREATE

In progress

Assign or Assign to me

POINTS

UX	0
Design	0
Front	0
Back	0
total points	0

Теперь чтобы перевести задачу 1 в состояние done, достаточно в коммите указать следующее
TG-1 #done

root / demo-rep

Следить 0

В избранное 0

Ответвить 0

Файлы

Задачи 0

Запросы на слияние 0

Вики

Настройки

Нет описания

9 Коммиты

1 Ветки

0 Релизы

Ветка: master

demo-rep

Новый файл

Загрузить файл

HTTP

SSH

http://localhost:3000/root/

user

d562042f27

TG-1 #done

11 минут назад

README.md

e416112cfb

first commit

4 часов назад

done.txt

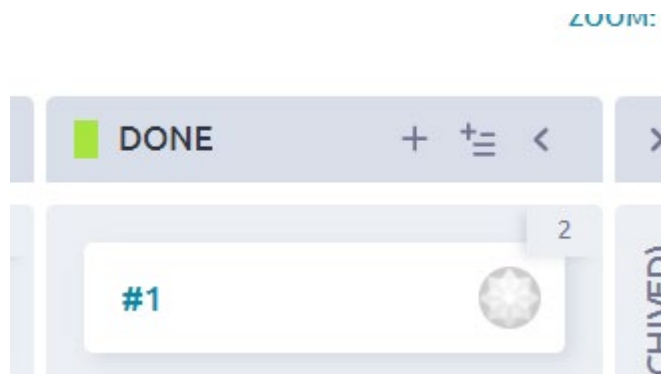
d562042f27

TG-1 #done

11 минут назад

README.md

Taiga переведёт задачу в состояние done




Создадим задачу по тестированию кода с помощью jenkins:

Возьмём проект laser-warmup-controller из интернета (симуляция лазера – его нагрев и охлаждение):

<https://github.com/overcomzi/laser-warmup-controller>

Запустим его в наш репозиторий

 **root / demo-rep**

Следить 0

В избранное 0

Ответвить 0

Файлы

Задачи 0

Запросы на слияние 0

Вики

Настройки

Нет описания

15 Коммиты

1 Ветки

0 Релизы

Ветка: master

demo-rep











Новый файл

Загрузить файл

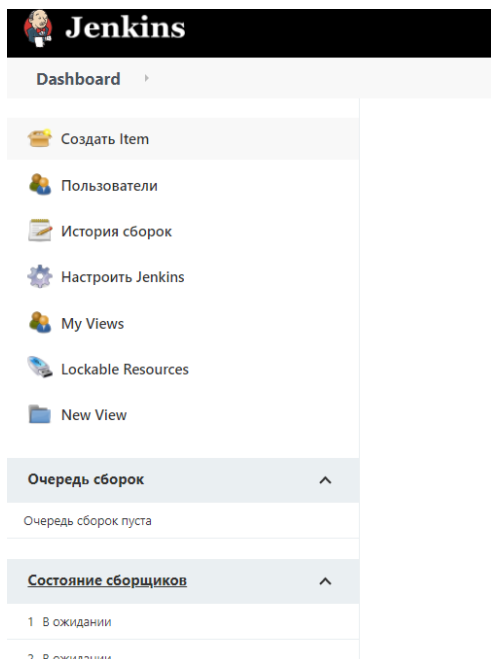
HTTP

SSH

http://localhost:3000/root/

 user	29fdbba0c68	Проект laser	23 секунд назад
 .gradle	29fdbba0c68	Проект laser	23 секунд назад
 .idea	29fdbba0c68	Проект laser	23 секунд назад
 build	29fdbba0c68	Проект laser	23 секунд назад
 gradle	29fdbba0c68	Проект laser	23 секунд назад
 src	29fdbba0c68	Проект laser	23 секунд назад
 build.gradle	29fdbba0c68	Проект laser	23 секунд назад
 gradlew	29fdbba0c68	Проект laser	23 секунд назад
 gradlew.bat	29fdbba0c68	Проект laser	23 секунд назад
 settings.gradle	29fdbba0c68	Проект laser	23 секунд назад

Добавим задачу по запуску gradle тестов в jenkins. Создаем item:




Item со свободной конфигурацией:

Введите имя Item'a

Manual-test

» Обязательное поле

 **Создать задачу со свободной конфигурацией**
Это - основной и наиболее универсальный тип задач в Jenkins. Jenkins будет собирать ваш проект, комбинируя любую SCM с любой сборочной системой. Данный тип проектов может использоваться для задач, отличных от сборки ПО.

Пропишем данные как на скриншотах:

General | **Gogs Webhook** | Управление исходным кодом | Триггеры сборки | Среда сборки | Сборка | Послесборочные операции

☐ Использовать другую директорию

Отображаемое имя

☐ Сохранять журнал сборки зависимостей

Управление исходным кодом

☐ Mer
☒ Git

Repositories

Repository URL

http://192.168.0.16:3000/root/demo-rep

Credentials

- none -

Расширенные...
Add Repository

Branches to build

Branch Specifier (blank for 'any')

*/master

Сборка

Invoke Gradle script

☐ Invoke Gradle

☒ Use Gradle Wrapper

☒ Make gradlew executable

Wrapper location ?

Tasks ?

test

Расширенные...

Добавить шаг сборки ▾

Послесборочные операции

Aggregate downstream test results

☒ Автоматически объединять результаты всех нисходящих (downstream) тестов ?

☒ Include failed builds in results

Добавить шаг после сборки ▾

Чтобы запустить тесты, достаточно собрать его и посмотреть в консоли результат:

Dashboard ▸ **Manual-test** ▸

- На главную
- Статус
- Изменения
- Сборочная директория
- Собрать сейчас**
- Сборка запланирована
- Удалить Проект
- Rename

История сборок **тренд** ^

find x

#1 24.06.2021 7:23


```
[Gradle] - Launching build.  
[Manual-test] $ /var/jenkins_home/workspace/Manual-test/gradlew test  
Starting a Gradle Daemon (subsequent builds will be faster)  
> Task :compileJava  
> Task :processResources NO-SOURCE  
> Task :classes  
> Task :compileTestJava  
> Task :processTestResources NO-SOURCE  
> Task :testClasses  
> Task :test  
  
BUILD SUCCESSFUL in 15s  
3 actionable tasks: 3 executed  
Build step 'Invoke Gradle script' changed build result to SUCCESS  
Finished: SUCCESS
```

Сборка #1 (24.06.2021 7:23:52)



No changes.



Создана пользователем [Иван](#)



Revision: b03fb68bf7c6f36a0bfc9dfc80499246a2d151c6

Repository: <http://192.168.0.16:3000/root/demo-rep>

- [refs/remotes/origin/master](#)



[Aggregated Test Result](#) (Нет тестов)

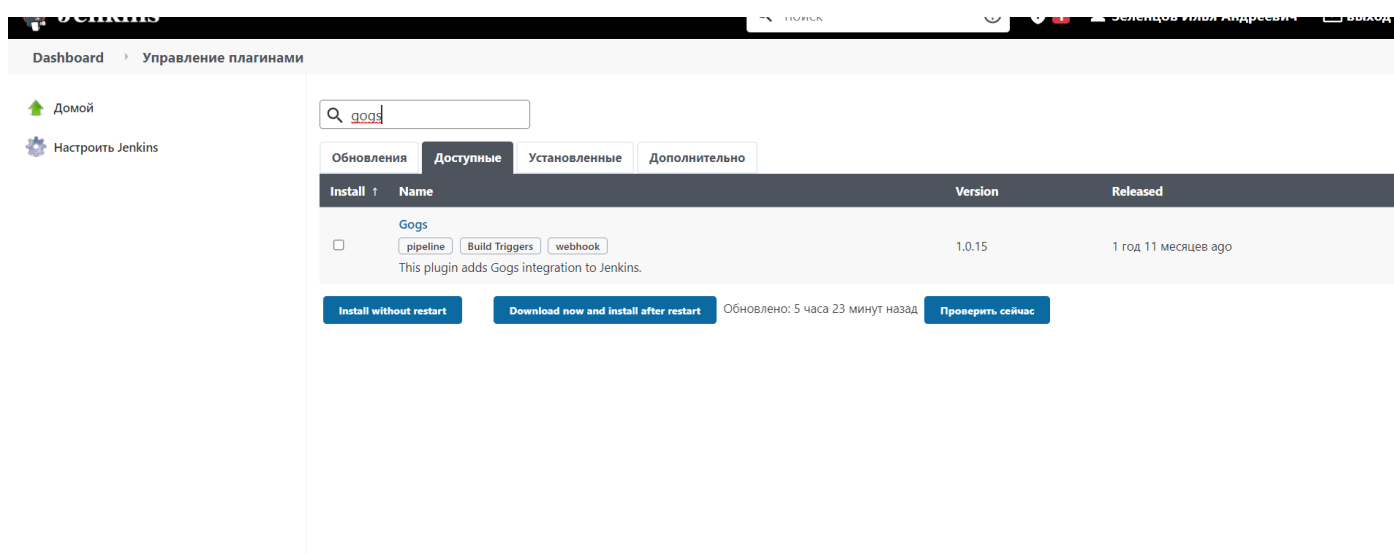
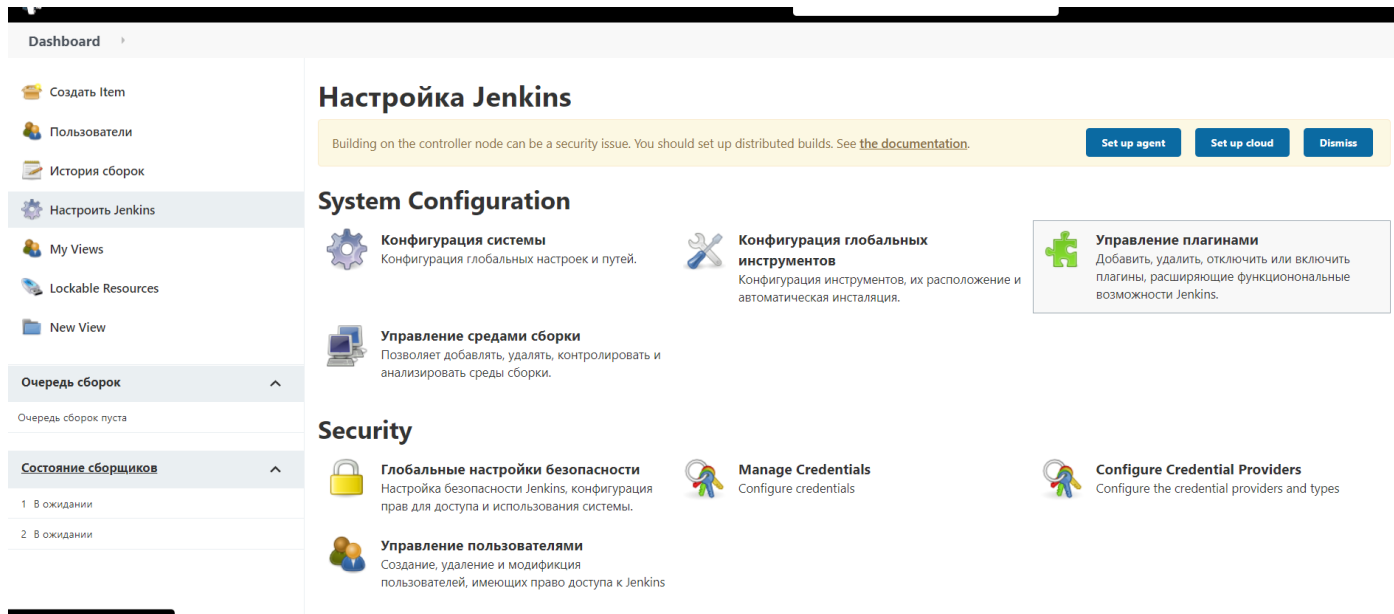
Свяжем gogs и jenkins:

<https://jamalshahverdiev.wordpress.com/2018/02/09/jenkins-gogs-integration-with-webhook/>

либо

<https://medium.com/@salohyprivat/getting-gogs-and-jenkins-working-together-5e0f21377bcd>

Добавим плагин gogs для Jenkins:



Устанавливаем с перезапуском

(либо вручную Перезапустим jenkins:

```
systemctl restart jenkins
```

)

сформируем ssh:

```
ssh-keygen -t rsa -b 4096 -C nick\_over@inbox.ru
```

Даём полномочия jenkins:

http://localhost:8080/credentials/store/system/domain/_/newCredentials

Прописываем данные как на скриншоте:

The screenshot shows the 'New Credentials' form in Jenkins. The 'Kind' is set to 'SSH Username with private key'. The 'Scope' is 'Global (Jenkins, nodes, items, all child items, etc)'. The 'ID' is '1'. The 'Description' is '1'. The 'Username' is 'root'. The 'Treat username as secret' checkbox is unchecked. The 'Private Key' section has 'Enter directly' selected. The 'Key' field contains a long base64-encoded string. The 'Passphrase' field is empty. An 'OK' button is at the bottom left.

Переходим к репозиторию gogs и прописываем ключ:

Добавляем публичный ключ, созданный ранее

Настройки
Профиль
Аватар
Пароль
Адреса электронной почты
SSH ключи
Безопасность
Репозитории
Организации
Приложения
Удалить аккаунт

Был успешно добавлен новый ключ SSH «jenkins ключ»!

Управление SSH ключами [Добавить ключ](#)
Это список ключей SSH связанных с вашей учетной записью. Удалите любые неизвестные вам ключи.

jenkins ключ
SHA256:gZqN14auTGVqcePN0M1HLA/N7H/gZBB1iLbbo7tmqus
Добавлено Jun 23, 2021 — [Еще не применялся](#)


[Удалить](#)

Нужна помощь? Ознакомьтесь с нашим путеводителем по [созданию SSH-ключей](#) или посмотрите решения [частых проблем, связанных с SSH](#).

Создаем новый item Jenkins на основе уже существующего (manual-test)

multibranch pipeline
Creates a set of Pipeline projects according to detected branches in one SCM repository.

Выберите эту опцию, если вы хотите создать Item из существующего

 Копировать из

OK

Не забудьте прописать secret (любой, какой захотите). Запомните его

Use Gogs secret
☒ Use Gogs secret
Secret
 Change Password

☐ Branch Filter
☐ This build requires lockable resources
☐ Throttle builds
☐ Удалять устаревшие сборки
☐ Это - параметризованная сборка
☐ Приостановить сборки
☐ Разрешить параллельный запуск задачи

Управление исходным кодом

Управление исходным кодом

☐ Hg
☒ Git

Repositories

Repository URL

Credentials
 Add

Branches to build

Branch Specifier (blank for 'any')

Add Repository

Среда сборки

☒ Delete workspace before build starts
Расширенные...

☐ Use secret text(s) or file(s)
☐ Abort the build if it's stuck
☐ Add timestamps to the Console Output
☐ With Ant

Сборка

Invoke Gradle script

☐ Invoke Gradle
☒ Use Gradle Wrapper
☒ Make gradlew executable
Wrapper location

Tasks

Расширенные...

Добавить шаг сборки ▾

Послесборочные операции

Aggregate downstream test results

☒ Автоматически объединять результаты всех нисходящих (downstream) тестов ⓘ
☒ Include failed builds in results

Добавить шаг после сборки ▾

Сохранить

Применить

В gogs Добавим вебхуки: (secrets должны совпадать)

URL обработчика, пример: <http://192.168.0.16:8080/gogs-webhook/?job=auto-test>

И вставим secret, который запомнили

Обновление Webhook

Мы отправим запрос POST на указанный ниже URL с информацией о событиях. Можно также указать формат, в котором вы бы хотели получить данные (JSON, x-www-form-urlencoded, и т.д.). Дополнительную информацию можно найти в [Руководстве по Webhook](#).

URL обработчика *

Тип содержимого

Secret

Секрет будет отправлен как SHA256 HMAC контента в шестнадцатеричном виде в заголовке X-Gogs-Signature.

На какие события этот webhook должен срабатывать?

☒ Просто push событие.
☐ Мне нужно **все**.
☐ Позвольте мне выбрать то, что нужно.

☒ **Активен**

Подробности о событии, вызвавшем срабатывание хука, также будут предоставлены.

Обновление Webhook

Удалить автоматическое обновление

Недавние рассылки

Проверить доставку

Теперь при коммите в репозиторий будут выполняться автоматически тесты.

Dashboard

auto-test

На главную

Статус

Изменения

Сборочная директория

Собрать сейчас

Настройки

Удалить Проект

Rename

История сборок

Тренд

find

#3

24.06.2021 7:39

#2

24.06.2021 7:38

#1

24.06.2021 7:36

Проект auto-test

123r23fjp34ifjgp143i9f2

Сборочная директория

Недавние изменения

Latest Aggregated Test Result (Нет тестов)

Постоянные ссылки

Последняя сборка (#3), 13 секунд назад

Последняя стабильная сборка (#3), 13 секунд назад

Последняя успешная сборка (#3), 13 секунд назад

Last completed build (#3), 13 секунд назад