

Задача 3: Jenkins/TeamCity

Подготовка

Jenkins

Убедимся, что Jenkins установлен и функционирует:

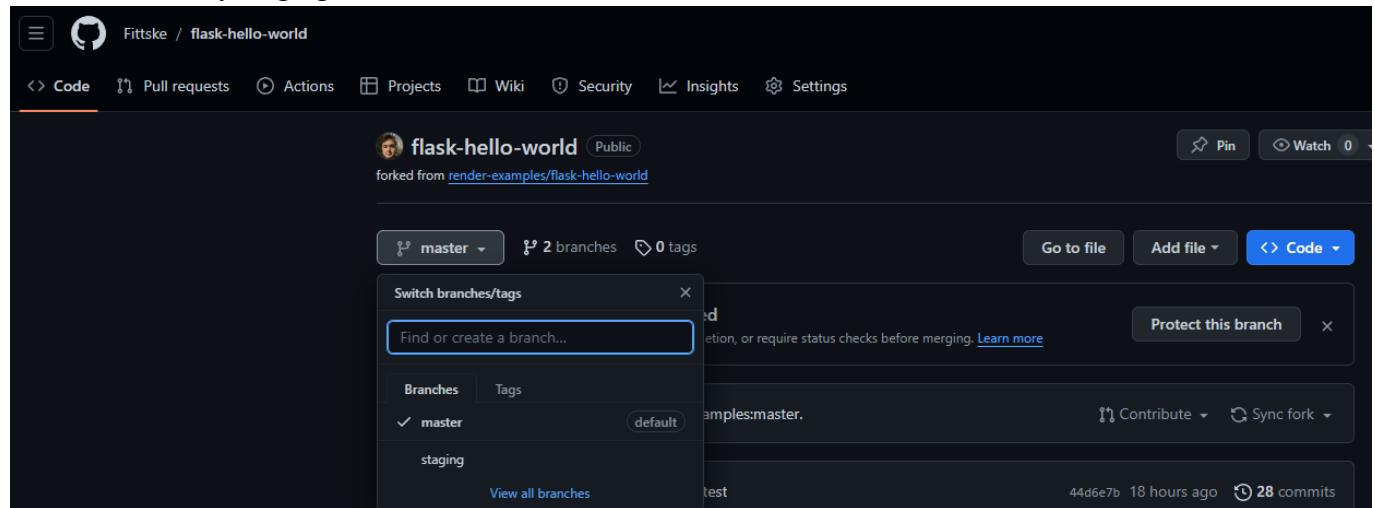
```
sudo systemctl status jenkins
```

```
ruslan@ruslan-Z690-UD:~/IT-2023/task_3$ sudo systemctl status jenkins
● jenkins.service - Jenkins Continuous Integration Server
   Loaded: loaded (/lib/systemd/system/jenkins.service; enabled; vendor preset: enabled)
     Active: active (running) since Sat 2023-11-11 18:52:40 +05; 5min ago
       Main PID: 26429 (java)
          Tasks: 63 (limit: 38128)
         Memory: 3.1G
            CPU: 36.987s
           CGroup: /system.slice/jenkins.service
                   └─26429 /usr/bin/java -Djava.awt.headless=true -jar /usr/share/java/jenkins.war --webroot=/var/cache/jenkins/war --httpPort=8080

ноя 11 18:52:39 ruslan-Z690-UD jenkins[26429]: 2023-11-11 13:52:39.976+0000 [id=47]      INFO      jenkins.InitReactorRunner$1#onAttained: Started all plugins
ноя 11 18:52:39 ruslan-Z690-UD jenkins[26429]: 2023-11-11 13:52:39.979+0000 [id=36]      INFO      jenkins.InitReactorRunner$1#onAttained: Augmented all extensions
ноя 11 18:52:40 ruslan-Z690-UD jenkins[26429]: 2023-11-11 13:52:40.202+0000 [id=47]      INFO      h.p.b.g.GlobalTimeoutConfiguration#load: global timeout not set
ноя 11 18:52:40 ruslan-Z690-UD jenkins[26429]: 2023-11-11 13:52:40.308+0000 [id=56]      INFO      jenkins.InitReactorRunner$1#onAttained: System config loaded
ноя 11 18:52:40 ruslan-Z690-UD jenkins[26429]: 2023-11-11 13:52:40.309+0000 [id=50]      INFO      jenkins.InitReactorRunner$1#onAttained: System config adapted
ноя 11 18:52:40 ruslan-Z690-UD jenkins[26429]: 2023-11-11 13:52:40.309+0000 [id=45]      INFO      jenkins.InitReactorRunner$1#onAttained: Loaded all jobs
ноя 11 18:52:40 ruslan-Z690-UD jenkins[26429]: 2023-11-11 13:52:40.315+0000 [id=52]      INFO      jenkins.InitReactorRunner$1#onAttained: Configuration for all jobs updated
ноя 11 18:52:40 ruslan-Z690-UD jenkins[26429]: 2023-11-11 13:52:40.333+0000 [id=46]      INFO      jenkins.InitReactorRunner$1#onAttained: Completed initialization
ноя 11 18:52:40 ruslan-Z690-UD jenkins[26429]: 2023-11-11 13:52:40.346+0000 [id=26]      INFO      hudson.lifecycle.Lifecycle#onReady: Jenkins is fully up and running
ноя 11 18:52:40 ruslan-Z690-UD systemd[1]: Started Jenkins Continuous Integration Server.
ruslan@ruslan-Z690-UD:~/IT-2023/task_3$
```

GitHub

В качестве веб-приложения возьмем готовый вариант с GitHub, сфоркаем в собственный репозиторий и создадим ветку staging:



Dockerfile

Подготовим Dockerfile для сборки нашего приложения и сохраним его в репозитории:

```
# Базовый образ
FROM python:3.9

# Определяем рабочую директорию контейнера
WORKDIR /webapp

# Копируем файл с необходимыми компонентами в созданную директорию
COPY requirements.txt /webapp
```

```
# Обновляем pip и устанавливаем компоненты
RUN pip install --upgrade pip && python -m pip install -r requirements.txt

# Указываем Docker порт для прослушивание (tcp)
EXPOSE 5000

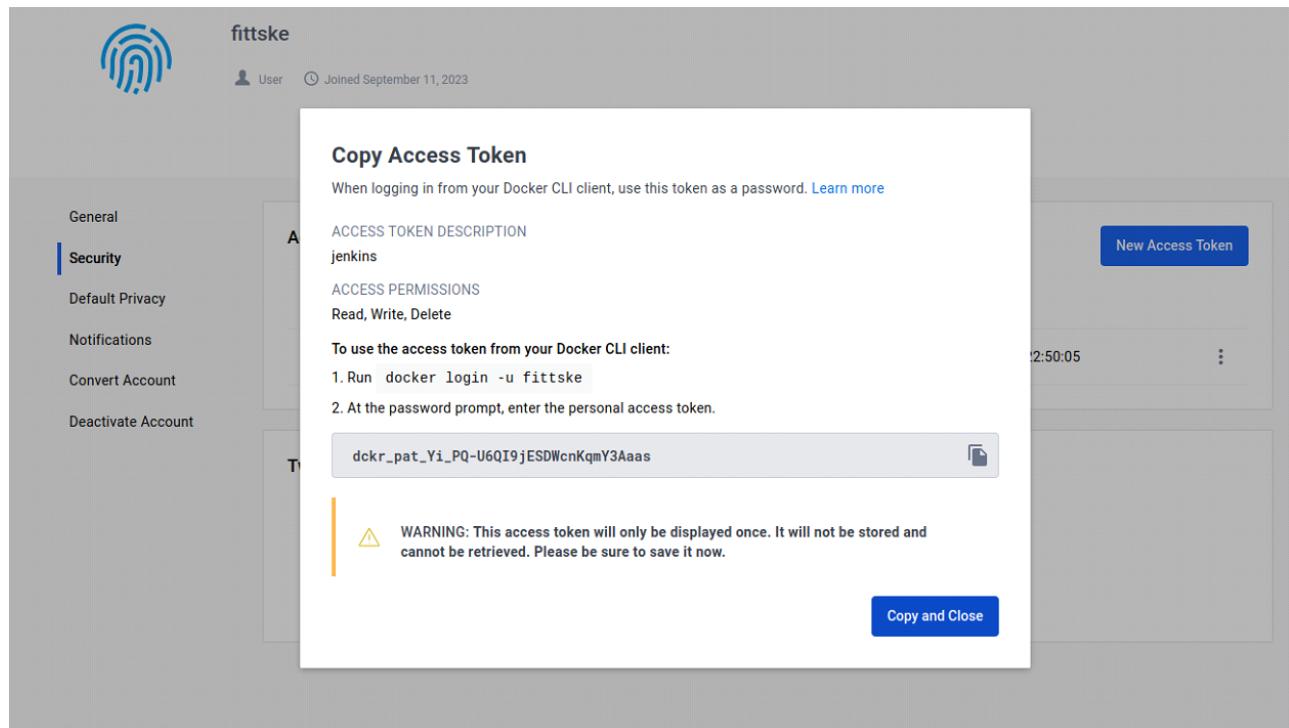
# Копируем наше приложение внутрь контейнера
COPY . /webapp

# Определяем команду для старта приложения
CMD ["gunicorn", "--bind", "0.0.0.0:5000", "app:app"]
```

Jenkins credentials

Подготовим некоторые реквизиты для доступа Jenkins к DockerHub:

1. В DockerHub, куда будем складывать образ собранного приложения, добавим собственный токен и скопируем его:



This screenshot shows the same user profile and sidebar as the previous one. The main area now displays the 'Access Tokens' section. It has a table with columns: Description, Source, Scope, Last Used, and Created. There is one entry: 'jenkins' (Source: MANUAL, Scope: Read, Write, Delete, Last Used: Never, Created: Nov 11, 2023 22:50:05). A blue 'New Access Token' button is in the top right of this section. Below it is a 'Two-Factor Authentication' section with a note: 'Two-factor authentication is not enabled yet. Two-factor authentication adds an extra layer of security to your account by requiring more than just a password to sign in.' A blue 'Enable Two-Factor Authentication' button is at the bottom of this section.

2. Далее в перейдем на фронт Jenkins'a, в раздел Credentials->System->Global credentials и создадим новые реквизиты для dockerhub:

New credentials

Kind: Username with password

Scope: Global (Jenkins, nodes, items, all child items, etc)

Username: fitske

Treat username as secret

Password:

ID: dockerhub

Description: dockerhub

Create

Подготовим пару ssh ключей для доступа пользователя Jenkins к удаленной ВМ, где будем разворачивать приложение:

- Перейдем на удаленную ВМ в папку `.ssh/` для нашего пользователя и сгенерируем ключи:

`ssh-keygen`

```
ruslan@vm-jenkins-deploy:~$ cd .ssh/
ruslan@vm-jenkins-deploy:~/ssh$ ssh-keygen
Generating public/private rsa key pair.
Enter file in which to save the key (/home/ruslan/.ssh/id_rsa):
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/ruslan/.ssh/id_rsa
Your public key has been saved in /home/ruslan/.ssh/id_rsa.pub
The key fingerprint is:
SHA256:hXS/a5+g46dN76eEUAEHtP0EKeYQlMn4Xx4X3frzn6U ruslan@vm-jenkins-deploy
The key's randomart image is:
+---[RSA 3072]---+
|       .+=X* o|
|       .o..B+.+o|
|       . .0000.o|
|       . .00.oo|
|      S . . . +o|
|       . o..o|
|       * . +|
|       .=.= o=|
|       .++..E+o|
+---[SHA256]---+
ruslan@vm-jenkins-deploy:~/ssh$ 
```

- Добавим строку с полученным публичным ключом в файл "authorized_keys" в этой же папке:

`nano authorized_keys`

- В Jenkins перейдем в раздел Credentials->System->Global credentials и добавим приватный ключ для доступа к удаленной ВМ:

Kind: SSH Username with private key

Scope: Global (Jenkins, nodes, items, all child items, etc)

ID: vm-deploy-ssh-key

Description: vm-deploy-ssh-key

Username: ruslan

Treat username as secret

Private Key: Enter directly

Key:

```
-----BEGIN OPENSSH PRIVATE KEY-----
b3BlbnNzaC1rZXktdjEAAAABG5vbmljAAAEBm9uZDAAAAAAAABAAABlwAAAAdzc2gtcn
NhAAAAAwEAAQAAAYEA7Xaz6F4CbDUwlon02Elvcn35WmqYBj2XAbSJhTDnFluwnF67Z0n1
-----END OPENSSH PRIVATE KEY-----
```

Jenkins plugins

Установим плагин ssh-agent для удаленного подключения к целевой ВМ по ssh:

The screenshot shows the Jenkins Plugins page. In the top navigation bar, there is a search bar with the placeholder 'поиск (CTRL+K)'. Below it, the user 'Ruslan Zamaletdinov' is logged in. The main area has a sidebar with 'Updates', 'Available plugins' (which is selected), 'Installed plugins', and 'Advanced settings'. The 'Available plugins' section contains a search bar with 'ssh agent' typed in. Below the search bar is a table with one row. The row contains a checkbox, the plugin name 'SSH Agent 346.vda_a_c4f2c8e50', a 'Released' timestamp '11 дней ago', and two buttons: 'Install' and a 'View' icon.

| | Name | Released | Action |
|--------------------------|-------------------------------|-------------|--|
| <input type="checkbox"/> | SSH Agent 346.vda_a_c4f2c8e50 | 11 дней ago | <input type="button" value="Install"/> <input type="button" value="View"/> |

Jenkins scripts

Заранее подготовим наши groovy скрипты для сборки и деплоя нашего приложения:

- Подготовим скрипт, который будет собирать наше приложение в докер-контейнере и пушить его образ в dockerhub. В качестве параметров будем использовать:
 - Созданные для dockerhub реквизиты - USERNAME и PASSWORD
 - Наименование создаваемого образа - DOCKERIMAGENAME.

```
#!/groovy
// Run docker build

//Запрещаем одновременное выполнение билдов
properties([disableConcurrentBuilds()])

pipeline {
  agent any
  triggers { pollSCM('* * * * *') }
```

```

options {
    //Храним 10 последних артефактов и билдов и добавляем временные
метки
    buildDiscarder(logRotator(numToKeepStr: '10',
artifactNumToKeepStr: '10'))
    timestamps()
}
stages {
    //Авторизация
    stage("docker login") {
        steps {
            echo " ===== docker login ====="
            withCredentials([usernamePassword(credentialsId:
'dockerhub', usernameVariable: 'USERNAME', passwordVariable:
'PASSWORD')]) {
                sh 'docker login -u $USERNAME -p $PASSWORD'
            }
        }
    }
    //Сборка образа
    stage("create docker image") {
        steps {
            echo " ===== start building image
===="
            sh 'docker build -t $DOCKERIMAGENAME . '
        }
    }
    //Пуш в докерхаб
    stage("docker push") {
        steps {
            echo " ===== start pushing image
===="
            sh 'docker push $DOCKERIMAGENAME'
        }
    }
}
}

```

2. Подготовим скрипт, который будет получать образ нашего приложения из dockerhub и запускать его на удаленной ВМ . В качестве параметров будем использовать:

1. Наименование образа - DOCKERIMAGENAME.
2. Публичный IP адрес удаленной ВМ - REMOTEHOSTIP.

```

#!/groovy
// Deploy and start docker image

//Запрещаем одновременное выполнение билдов
properties([disableConcurrentBuilds()])

```

```

pipeline {
    agent any
    stages {
        // Подключение к удаленной ВМ с помощью SSH и запуск приложения из
образа
        stage("Connect to VM via SSH and deploy app") {
            environment{
                dockerRun = "sudo apt update && sudo docker run -d -p
5000:5000 $DOCKERIMAGENAME"
            }
            steps {
                echo " ====== Connect to VM via SSH and deploy
====="
                sshagent(credentials : ['vm-deploy-ssh-key']) {
                    sh "ssh -o StrictHostKeyChecking=no -l ruslan
$REMOTEHOSTIP '${env.dockerRun}' "
                }
            }
        }
    }
}

```

3. Запушим данные скрипты в наш GitHub репозиторий в ветку "Staging":

The screenshot shows the GitHub interface for the `flask-hello-world` repository. The left sidebar shows the `staging` branch selected. The main area displays a list of 18 commits from the Jenkins pipeline. The commits are:

- `docker_build_jenkins`: Commit for jenkins pipeline groovy script v5 (2 days ago)
- `docker_deploy_and_start_jenkins`: Commit for deploy and start script (yesterday)
- `test_file_for_commit.txt`: Test commit №6 for auto deploy test (20 hours ago)

4. Подготовим аналогичные скрипты, но для deploy укажем порт 8000, после чего запушим их в GitHub репозиторий в ветку "Master":

`sudo docker run -d -p 8000:5000 $DOCKERIMAGENAME`

The screenshot shows the GitHub interface for the `flask-hello-world` repository. The left sidebar shows the `master` branch selected. The main area displays a list of 21 commits from the Jenkins pipeline. The commits are:

- `docker_build_jenkins`: Commit for jenkins pipeline groovy script v5 (2 days ago)
- `docker_deploy_and_start_jenkins`: Commit for production - port change on docker run (yesterday)
- `test_file_for_commit.txt`: Test commit №7 for auto deploy test (20 hours ago)

Настройка Jenkins

Настройка проекта "Staging"

1. Создадим папку для проекта "Staging":

Введите имя Item'a

Staging
» 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.

Pipeline
Orchestrates long-running activities that can span multiple build agents. Suitable for building pipelines (formerly known as workflows) and/or organizing complex activities that do not easily fit in free-style job type.

Мультиконфигурационный проект
Подходит для проектов, требующих большое количество различных конфигураций, таких как тестирование в разнообразных средах, платформозависимые сборки и т.д.

Folder
Creates a container that stores nested items in it. Useful for grouping things together. Unlike view, which is just a filter, a folder creates a separate namespace, so you can have multiple things of the same name as long as they are in different folders.

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

Organization Folder
Creates a set of multibranch project subfolders by scanning for repositories.

OK

Выберите эту опцию, если вы хотите создать Item из существующего

Dashboard > Staging > Configuration

Configuration

General

Display Name ?
Staging

Description
Staging project

Plain text: Предпросмотр

Health metrics

Health metrics ^

Add metric ▾

Properties

Pipeline Libraries

Sharable libraries available to any Pipeline jobs inside this folder. These libraries will be untrusted, meaning their code runs in the Groovy sandbox.

Добавить

Save Применить

2. Внутри проекта создадим новый item для задачи сборки с типом "Pipeline":

Введите имя Item'a

Build and push
» 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.

 **Pipeline**
Orchestrates long-running activities that can span multiple build agents. Suitable for building pipelines (formerly known as workflows) and/or organizing complex activities that do not easily fit in free-style job type.

 **Мультиконфигурационный проект**
Подходит для проектов, требующих большое количество различных конфигураций, таких как тестирование в разнообразных средах, платформозависимые сборки и т.д.

 **Folder**
Creates a container that stores nested items in it. Useful for grouping things together. Unlike view, which is just a filter, a folder creates a separate namespace, so you can have multiple things of the same name as long as they are in different folders.

 **Multibranch Pipeline**
Creates a set of Pipeline projects according to detected branches in one SCM repository.

 **Organization Folder**
Creates a set of multibranch project subfolders by scanning for repositories.

OK

Выберите эту опцию, если вы хотите создать Item из существующего

Общие настройки

Enabled

Описание

Build docker image and push to repo

Plain text [Предпросмотр](#)

Do not allow concurrent builds

Do not allow the pipeline to resume if the controller restarts

GitHub project

Pipeline speed/durability override ?

3. Укажем флаг "Это параметризированная сборка" и создадим параметр для наименования docker образа:



4. Укажем Git в качестве SCM для пайплайна и укажем ссылку на наш репозиторий:

The screenshot shows a Jenkins Pipeline configuration page. Under 'Definition', it says 'Pipeline script from SCM'. Under 'SCM', it is set to 'Git'. In the 'Repositories' section, the URL is git@github.com:Fittske/flask-hello-world.git. A red error message indicates a failed connection: 'Failed to connect to repository : Command "git ls-remote -h - git@github.com:Fittske/flask-hello-world.git HEAD" returned status code 128: stdout: stderr: git@github.com: Permission denied (publickey). fatal: Не удалось прочитать из внешнего репозитория.' Below the error message, there is a note: 'Убедитесь, что у вас есть необходимые права доступа и репозиторий существует.' Under 'Credentials', it says '-none-'.

5. Как видно из скрина, Jenkins не может подключиться к репозиторию на данный момент. Поэтому - добавим credentials для GitHub репозитория:

1. Сгенерируем пару ключей для git (<https://docs.github.com/en/authentication/connecting-to-github-with-ssh/generating-a-new-ssh-key-and-adding-it-to-the-ssh-agent>):

```
ssh-keygen -t ed25519 -C r3cordsman@mail.ru
```

```
ruslan@ruslan-Z690-UD:~/ssh$ ssh-keygen -t ed25519 -C "r3cordsman@mail.ru"
Generating public/private ed25519 key pair.
Enter file in which to save the key (/home/ruslan/.ssh/id_ed25519):
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/ruslan/.ssh/id_ed25519
Your public key has been saved in /home/ruslan/.ssh/id_ed25519.pub
The key fingerprint is:
SHA256:xigPzw44pCecEXwUeERg4x7GA29PTmNTbfNFZkzVChg r3cordsman@mail.ru
The key's randomart image is:
+--[ED25519 256]--+
| .+=. .. *=o.o |
| *oo. . + .oE . .
| 0+. * . o o |
| o.=* o o . |
| o. = . S |
| .oo. * . |
| ooo+ . + |
| o . o |
| . |
+---[SHA256]---+
ruslan@ruslan-Z690-UD:~/ssh$
```

2. Публичный ключ добавим в GitHub в настройки аккаунта:

Ruslan Zamaletdinov (Fittske)
Your personal account

SSH keys

This is a list of SSH keys associated with your account. Remove any keys that you do not recognize.

Authentication Keys

| Key | SHA256 | Added on | Last used | Permissions | Action |
|----------------|---|--------------|--------------------------|-------------|--------|
| MyKeyFromLinux | SHA256:FZ9bmAV/81R5f/sjIZVuytPwzoqVGUDxEZ1bYkcKksLY | Aug 17, 2023 | Within the last 3 months | Read/write | Delete |
| Jenkins | SHA256:xiGPzw4pCecExwJeERg4x7GA29PTmNTb+NFZkzVChg | Nov 11, 2023 | Within the last week | Read/write | Delete |

Check out our guide to [generating SSH keys](#) or troubleshoot common SSH problems.

3. Приватный ключ добавим в Jenkins credentials:

Kind: SSH Username with private key

Scope: Global (Jenkins, nodes, items, all child items, etc)

ID: github-ssh-key

Description: github-ssh-key

Username: Fittske

Treat username as secret

Private Key: Enter directly
Key:

6. После добавления ключей, выбираем нужный и ошибка пропадает, Jenkins успешно подключается:

Pipeline

Definition: Pipeline script from SCM

SCM: Git

Repositories: git@github.com:Fittske/flask-hello-world.git

Credentials: Fittske (github-ssh-key)

Add Repository

7. Укажем ветку "Staging" и определим путь к скрипту для сборки:

Branches to build ?

Branch Specifier (blank for 'any') ?

*/staging

Add Branch

Просмотрщик репозитория ?

(Автоматически)

Additional Behaviours

Добавить ▾

Script Path ?

Jenkins/docker_build.jenkins

Lightweight checkout ?

Pipeline Syntax

Сохранить Применить

8. Пайплайн для сборки ГОТОВ:

Jenkins

Dashboard > Staging > Build and push >

поиск (CTRL+K) Rustan Zamaletdinov выход

Pipeline Build and push

Status

Changes

Собрать с параметрами

Настройки

Удалить Pipeline

Move

Full Stage View

Переименовать

Pipeline Syntax

Stage View

No data available. This Pipeline has not yet run.

изменить описание

Выключить проект

История сборок ТРОЛЯ ▾

Нет сборок

Atom feed для RSS Atom feed для недавних

9. Далее, запустим наш пайплайн. Jenkins сразу просит указать значение параметра для наименования образа нашего приложения:

Dashboard > Staging > Build and push >

Pipeline Build and push

Status

Changes

Собрать с параметрами

Настройки

Удалить Pipeline

Move

Full Stage View

Переименовать

Pipeline Syntax

Для этой сборки необходимы следующие параметры:

DOCKERIMAGENAME

Please, enter docker image name

fittske/jenkins-docker-hub

Собрать Cancel

10. Убеждаемся в успешности выполнения пайплайна:

The screenshot shows the Jenkins Pipeline Build and push stage view. On the left, there is a sidebar with various options: Status, Changes, Собрать с параметрами (Build with parameters), Настройки (Configure), Удалить Pipeline (Delete Pipeline), Move, Full Stage View, Переименовать (Rename), Pipeline Syntax, История сборок (Build History) with a trend dropdown, and a search bar for filtering builds. Below the sidebar, it says "Полное название проекта: Staging/Build and push" and "Build docker image and push to repo". The main area is titled "Stage View" and displays a timeline of stages: Declarative: Checkout SCM (2s), docker login (2s), create docker image (1s), and docker push (14s). A summary at the top states "Average stage times: (Average full run time: ~23s)". Below the timeline, a specific build entry for #1 is shown with the date "нояб. 11" and time "23:51", noting "No Changes". The "Постоянные ссылки" (Permanent links) section includes a link to the "Последняя сборка (#1), 7 ms назад".

The screenshot shows the Jenkins Build #1 details page. The top navigation bar includes "Dashboard", "Staging", "Build and push", and "#1". The main content area features a large green checkmark icon followed by the text "Build #1 (11 нояб. 2023 г., 23:51:38)". To the left is a sidebar with options: Status, Changes, Console Output, Edit Build Information, Delete build '#1', Параметры (Parameters), Git Build Data, Restart from Stage, Replay, Pipeline Steps, and Workspaces. On the right, there is information about the build: "Создана пользователем Ruslan Zamaletdinov", "Revision: 2887af14544152e6218dfc3c2429066c0d4d059d", "Repository: git@github.com:Fittske/flask-hello-world.git", and a ref list: "refs/remotes/origin/staging".

11. Проверим вывод консоли:

```
Dashboard > Staging > Build and push > #1
-----
23:51:47 29e49b59edda: Preparing
23:51:47 70866f64c03e: Waiting
23:51:47 1777ac7d307b: Preparing
23:51:47 2d788bc47240: Waiting
23:51:47 12b956927ba2: Waiting
23:51:47 86e50e0709ee: Waiting
23:51:47 29e49b59edda: Waiting
23:51:47 266def75d28e: Waiting
23:51:47 1777ac7d307b: Waiting
23:51:49 fa83a371445d: Mounted from library/python
23:51:50 ed5e0a89a0fd: Pushed
23:51:51 8c6e19de4630: Pushed
23:51:51 0cee309e6bb0: Pushed
23:51:52 70866f64c03e: Mounted from library/python
23:51:53 d316dc65e7a6: Pushed
23:51:53 2d788bc47240: Mounted from library/python
23:51:54 86e50e0709ee: Mounted from library/python
23:51:54 12b956927ba2: Mounted from library/python
23:51:55 29e49b59edda: Mounted from library/python
23:51:55 266def75d28e: Mounted from library/python
23:51:56 1777ac7d307b: Mounted from library/python
23:52:01 latest: digest: sha256:d9bbc992bb56cc5a4975d718a5d44649c6027c099a4de29ff35443a5315622b size: 2840
[Pipeline]
[Pipeline] // stage
[Pipeline]
[Pipeline] // timestamps
[Pipeline]
[Pipeline] // withEnv
[Pipeline]
[Pipeline] // node
[Pipeline] End of Pipeline
Finished: SUCCESS
```

12. Проверим, выгруился ли образ в dockerhub:

The screenshot shows the Docker Hub interface. At the top, there's a search bar with 'Search Docker Hub' and a 'ctrl+K' keyboard shortcut. Below the search bar are navigation links for 'Explore', 'Repositories', 'Organizations', 'Help', and an 'Upgrade' button. On the right, there's a user profile icon for 'fittske'.

In the main area, there's a search bar with 'fittske' and a dropdown menu. Below it, there are three repository cards:

- fittske / jenkins-docker-hub**
Contains: Image | Last pushed: a minute ago
- fittske / k8sapache**
Contains: Image | Last pushed: 2 months ago
- fittske / k8sphp**
Contains: Image | Last pushed: 2 months ago

Each repository card includes a star icon (0 stars), a fork icon (0 forks), and a 'Public' label.

13. Далее переходим к созданию нового item для задачи деплоя приложения:

Введите имя Item'a

Deploy and start
» 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.

 **Pipeline**
Orchestrates long-running activities that can span multiple build agents. Suitable for building pipelines (formerly known as workflows) and/or organizing complex activities that do not easily fit in free-style job type.

 **Мультиконфигурационный проект**
Подходит для проектов, требующих большое количество различных конфигураций, таких как тестирование в разнообразных средах, платформозависимые сборки и т.д.

 **Folder**
Creates a container that stores nested items in it. Useful for grouping things together. Unlike view, which is just a filter, a folder creates a separate namespace, so you can have multiple things of the same name as long as they are in different folders.

 **Multibranch Pipeline**
Creates a set of Pipeline projects according to detected branches in one SCM repository.

 **Organization Folder**
Creates a set of multibranch project subfolders by scanning for repositories.

Выберите эту опцию, если вы хотите создать Item из существующего

OK Старт из

14. В качестве параметров указываем наименование образа для деплоя, а также IP адрес удаленной VM, где он будет производиться:

Это - параметризованная сборка ?

≡ String Parameter ?

Имя ?
DOCKERIMAGENAME

Значение по умолчанию ?
fittske/jenkins-docker-hub

Описание ?
Please enter docker image to deploy and start

Plain text: Предпросмотр

Обрезать пробелы ?

[Добавить Параметр ▾](#)

≡ String Parameter ?

Имя ?
REMOTEHOSTIP

Значение по умолчанию ?
158.160.115.211

Описание ?
Please enter public ip address of remote host on which to deploy and start image.

Plain text: Предпросмотр

Обрезать пробелы ?

[Добавить Параметр ▾](#)

15. Указываем зависимость задачи деплоя от задачи сборки:

Запустить по окончанию сборки других проектов ?

Устанавливает триgger таким образом, что когда другой проект завершает сборку, запускается сборка этого проекта. Это полезно для запуска дополнительных тестов после завершения сборки, например.

Эта опция противоположна опции "Собрать другой проект" в секции "Послесборочные действия". Изменив это значение, второе изменится автоматически.

Наблюдаемые проекты:
Build and push

Срабатывать, только если сборка стабильна
 Срабатывать, даже если сборка нестабильна
 Срабатывать, даже если сборка провалилась
 Срабатывать в любом случае, даже если сборка была прервана

Опрашивать SCM об изменениях ?
 Пауза перед сборкой ?
 Trigger builds remotely (e.g., from scripts) ?

[Сохранить](#) [Применить](#)

16. Производим настройки по SCM аналогично пайплайну для сборки, указываем ветку и путь к скрипту в нашем github репозитории:

Pipeline

Definition

Pipeline script from SCM

SCM

Git

Repositories

Repository URL: git@github.com:Fittske/flask-hello-world.git

Credentials: Fittske (github-ssh-key)

Add

Расширенные

Branches to build

Branch Specifier (blank for 'any'): */staging

Add Branch

Просмотрщик репозитория: (Автоматически)

Additional Behaviours

Добавить

Script Path

Jenkins/docker_deploy_and_start.jenkins

Lightweight checkout

17. Настроим опрос SCM на каждую минуту, чтобы деплой автоматом запускался при commit в ветке staging:

Dashboard > Staging > Deploy and start > Настройка

Configure

- Общие настройки
- Advanced Project Options
- Pipeline

запустить по окончанию сборки других проектов

Наблюдаемые проекты: Build and push

Срабатывать, только если сборка стабильна

Срабатывать, даже если сборка нестабильна

Срабатывать, даже если сборка провалилась

Срабатывать в любом случае, даже если сборка была прервана

Опрашивать SCM об изменениях

Расписание: ****

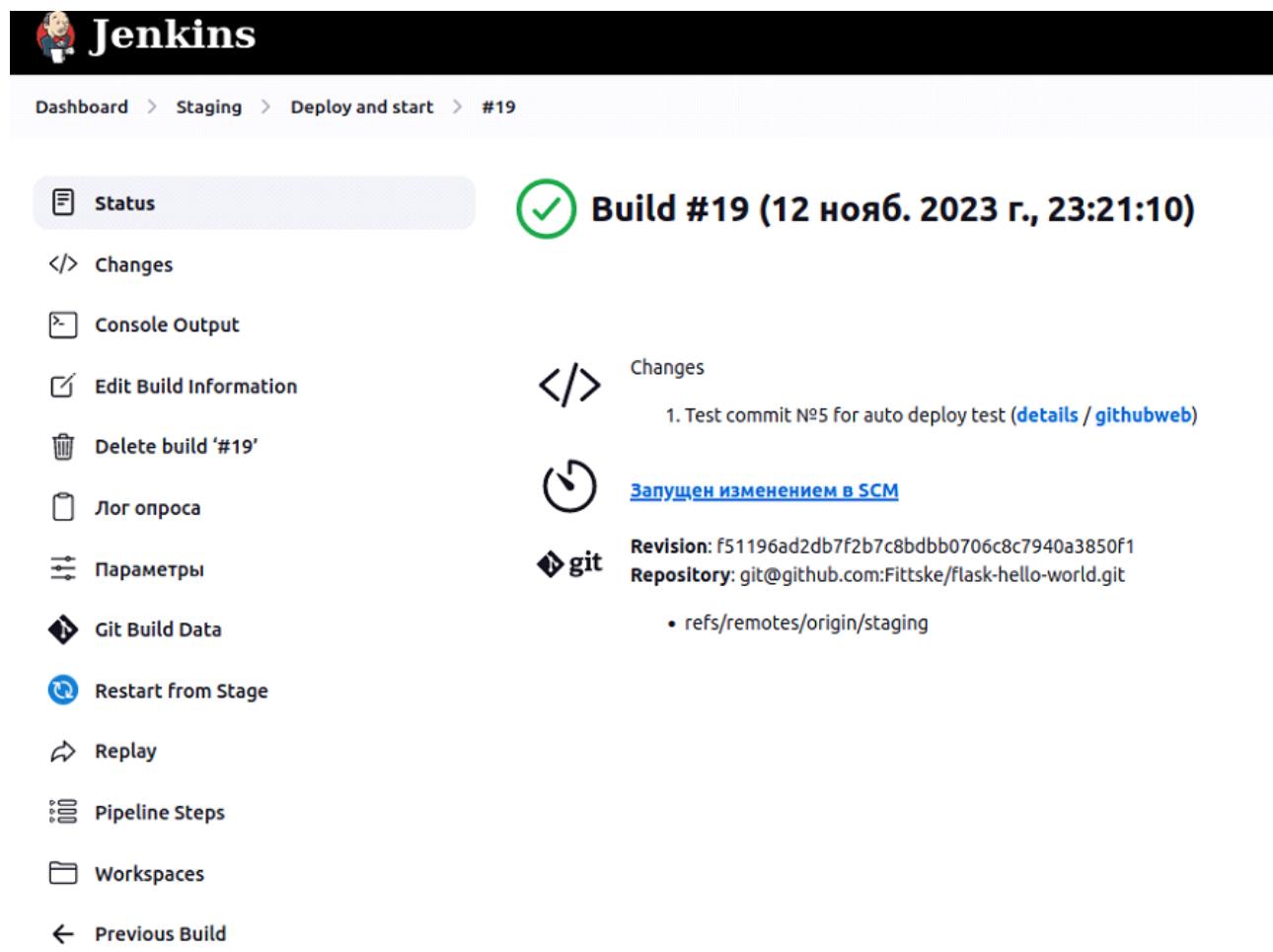
⚠️ Do you really mean "every minute" when you say "****"? Perhaps you meant "H * * * *" to poll once per hour
Последний запланированный запуск: воскресенье, 12 ноября 2023 г., 23:03:30 Екатеринбург, стандартное время; следующий запланированный запуск: воскресенье, 12 ноября 2023 г., 23:03:30 Екатеринбург, стандартное время.

18. Сделаем тестовый commit в ветку staging и убедимся в запуске пайплайна:

The screenshot shows the Jenkins interface for a pipeline named 'Staging'. The top navigation bar includes 'Dashboard', 'Staging', 'Deploy and start', and 'Лог опроса Git'. The left sidebar has options like 'Status', 'Changes', 'Собрать с параметрами', 'Настройки', 'Удалить Pipeline', 'Move', 'Full Stage View', 'Переименовать', 'Pipeline Syntax', and 'Лог опроса Git'. The main content area is titled 'Журнал последнего опроса SCM' and displays the following log output:

```
Started on 12 нояб. 2023 г., 23:21:00
Using strategy: Default
[poll] Last Built Revision: Revision 3616949d4e1f3ee79b4727206d180339ca34afe5 (refs/remotes/origin/staging)
Selected Git installation does not exist. Using Default
The recommended git tool is: NONE
using credential github-ssh-key
> git --version # timeout=10
> git --version # 'git version 2.34.1'
using GIT_SSH to set credentials github-ssh-key
> git ls-remote -h -- git@github.com:Fittske/flask-hello-world.git # timeout=10
Found 2 remote heads on git@github.com:Fittske/flask-hello-world.git
[poll] Latest remote head revision on refs/heads/staging is: f51196ad2db7f2b7c8bdbb0706c8c7940a3850f1
Done. Took 1,8 секунды
Changes found
```

Below the log, there's a summary section with 'История сборок' (Build History) and a 'Фильтровать сборки...' (Filter builds...) search bar. A specific build is highlighted: '#19 12 нояб. 2023 г., 23:21'.



The screenshot shows the Jenkins interface for Build #19. The main header says "Build #19 (12 нояб. 2023 г., 23:21:10)". On the left, there's a sidebar with various build-related links like Status, Changes, Console Output, and Git Build Data. The "Changes" section lists a single commit: "Test commit №5 for auto deploy test". The "Git" section shows the revision and repository: "f51196ad2db7f2b7c8bdbb0706c8c7940a3850f1" and "git@github.com:Fittske/flask-hello-world.git". Below these, there's a "Pipeline Steps" section with a single step named "Deploy and start". The "Console Output" tab is open, displaying the terminal logs for the build process.

```

[Pipeline] sh
+ ssh -o StrictHostKeyChecking=no -l ruslan 158.160.115.211 sudo apt update && sudo docker run -d -p 5000:5000 fittske/jenkins-docker-hub

WARNING: apt does not have a stable CLI interface. Use with caution in scripts.

Hit:1 http://mirror.yandex.ru/ubuntu jammy InRelease
Hit:2 http://mirror.yandex.ru/ubuntu jammy-updates InRelease
Hit:3 http://mirror.yandex.ru/ubuntu jammy-backports InRelease
Hit:4 http://security.ubuntu.com/ubuntu jammy-security InRelease
Hit:5 https://download.docker.com/linux/ubuntu jammy InRelease
Reading package lists...
Building dependency tree...
Reading state information...
7 packages can be upgraded. Run 'apt list --upgradable' to see them.
5bedf69c4785b343b253ed7e5cb17c245a9e3243dc601d8fc955fcf4a20c1
[Pipeline] }
$ ssh-agent -k
unset SSH_AUTH_SOCK;
unset SSH_AGENT_PID;
echo Agent pid 52505 killed;
[ssh-agent] Stopped.
[Pipeline] // sshagent
[Pipeline] }
[Pipeline] // withEnv
[Pipeline] }
[Pipeline] // stage
[Pipeline] }
[Pipeline] // withEnv
[Pipeline] }
[Pipeline] // node
[Pipeline] End of Pipeline
Finished: SUCCESS

```

19. Зайдем на удаленную ВМ и проверим состояние нашего контейнера:

```

sudo docker ps
ruslan@vm-jenkins-deploy: $ sudo docker ps
CONTAINER ID   IMAGE          COMMAND       CREATED      STATUS      PORTS          NAMES
5b6df9c4785   fittske/jenkins-docker-hub   "gunicorn --bind 0.0..."   About a minute ago   Up About a minute   0.0.0.0:5000->5000/tcp, :::5000->5000/tcp   eloquent_chebyshev
ruslan@vm-jenkins-deploy: $ 

```

20. Проверим в браузере - приложение запустилось на порту 5000:



21. Далее в проекте "staging" создадим пайплайн для создания бэкапа на удаленном хосте:

Введите имя Item'a

Backup image
» 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.

Pipeline
Orchestrates long-running activities that can span multiple build agents. Suitable for building pipelines (formerly known as workflows) and/or organizing complex activities that do not easily fit in free-style job type.

Мультиконфигурационный проект
Подходит для проектов, требующих большое количество различных конфигураций, таких как тестирование в разнообразных средах, платформозависимые сборки и т.д.

Folder
Creates a container that stores nested items in it. Useful for grouping things together. Unlike view, which is just a filter, a folder creates a separate namespace, so you can have multiple things of the same name as long as they are in different folders.

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

Organization Folder
Creates a set of multibranch project subfolders by scanning for repositories.

Выберите эту опцию, если вы хотите создать Item из существующего

OK

22. Укажем параметры аналогичные задаче деплоя:

The screenshot shows the Jenkins configuration interface for defining parameters. There are two entries in a list:

- String Parameter** (checkbox checked):
 - Имя**: DOCKERIMAGENAME
 - Значение по умолчанию**: fitske/jenkins-docker-hub
 - Описание**: Please enter which docker image to backup on server
 - Plain text**: Предпросмотр
 - Обрезать пробелы**: checkbox unchecked
- String Parameter** (checkbox unchecked):
 - Имя**: REMOTENHOSTIP
 - Значение по умолчанию**: 158.160.115.211
 - Описание**: Please enter ip address of host where to save image
 - Plain text**: Предпросмотр
 - Обрезать пробелы**: checkbox unchecked

At the bottom of each section is a "Добавить Параметр" (Add Parameter) button.

23. Настроим зависимость от задачи деплоя:

Build Triggers

- GitHub hook trigger for GITScm polling
- Запускать периодически
- Запустить по окончанию сборки других проектов
 - Наблюдаемые проекты: Deploy and start
 - Срабатывать, только если сборка стабильна
 - Срабатывать, даже если сборка нестабильна
 - Срабатывать, даже если сборка провалилась
 - Срабатывать в любом случае, даже если сборка была прервана

24. Укажем скрипт для пайплайна прямо внутри Jenkins - будем складывать tar архив в папку backups нашего пользователя:

Pipeline

Definition

Pipeline script

```

1 #!/groovy
2 // Deploy and start docker image
3
4 //Запрещаем одновременное выполнение билдов
5 properties(disableConcurrentBuilds())
6
7+ pipeline {
8     agent any
9     stages {
10         stage("Connect to VM via SSH and deploy app") {
11             environment{
12                 Backup = "sudo docker pull $DOCKERIMAGENAME && mkdir -p backups && sudo docker image save -o ./backups/image-$(date +\"%d-%m-%Y-%H%M%S\").tar $DOCKERIMAGENAME"
13             }
14             steps {
15                 echo " ===== Connect to VM via SSH and deploy ====="
16                 sshagent(credentials : ['vm-deploy-ssh-key']) {
17                     sh "ssh -o StrictHostKeyChecking=no -t ruslan $REMOTEHOSTIP '$env.Backup'"
18                 }
19             }
20         }
21     }
22 }
23 }
```

Use Groovy Sandbox

25. Проверим работу пайплайна, запустим его:

Pipeline Backup image

Для этой сборки необходимы следующие параметры:

DOCKERIMAGENAME

Please enter which docker image to backup on server

fittske/jenkins-docker-hub

REMOTEHOSTIP

Please enter ip address of host where to save image

158.160.115.211

Собрать

Cancel

26. Проверяем вывод консоли:

```

[Pipeline] 
[Pipeline] echo
===== Connect to VM via SSH and deploy =====
[Pipeline] sshagent
[ssh-agent] Using credentials ruslan (vm-deploy-ssh-key)
[ssh-agent] Looking for ssh-agent implementation...
[ssh-agent]   Exec ssh-agent (binary ssh-agent on a remote machine)
$ ssh-agent
SSH_AUTH_SOCK=/tmp/ssh-XXXXXXAwqQIG/agent.35575
SSH_AGENT_PID=35578
Running ssh-add (command line suppressed)
Identity added: /var/lib/jenkins/workspace/Staging/Backup image@tmp/private_key_14217926555055064449.key (ruslan@vm-jenkins-deploy)
[ssh-agent] Started.
[Pipeline] {
[Pipeline] sh
+ ssh -o StrictHostKeyChecking=no -l ruslan 158.160.115.211 sudo docker pull fittske/jenkins-docker-hub && mkdir -p backups && sudo docker image save -o ./backups/image-$(date + "%d-%m-%Y-%H%M%S").tar fittske/jenkins-docker-hub
Using default tag: latest
latest: Pulling from fittske/jenkins-docker-hub
Digest: sha256:d9bbec992bb56cc5a4975d718a5d44649c6027c099a4de29ff35443a5315622b
Status: Image is up to date for fittske/jenkins-docker-hub:latest
docker.io/fittske/jenkins-docker-hub:latest
[Pipeline] }
$ ssh-agent -k
unset SSH_AUTH_SOCK;
unset SSH_AGENT_PID;
echo Agent pid 35578 killed;
[ssh-agent] Stopped.
[Pipeline] // sshagent
[Pipeline] }
[Pipeline] // withEnv
[Pipeline] }
[Pipeline] // stage
[Pipeline] }
[Pipeline] // node
[Pipeline] End of Pipeline
Finished: SUCCESS
```

27. Зайдем на ВМ и проверим папку backups:

```
ruslan@vm-jenkins-deploy:~/backups$ ll
total 1019360
drwxrwxr-x 2 ruslan ruslan      4096 Nov 12 15:58 .
drwxr-x--- 6 ruslan ruslan      4096 Nov 12 15:56 ..
-rw----- 1 root   root    1043811840 Nov 12 15:58 image-12-11-2023-155652.tar
ruslan@vm-jenkins-deploy:~/backups$ 
```

28. Настроим пайплайн внутри проекта "staging", где будут включены все наши шаги:

Введите имя Item'a

Main pipeline
» 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.

Pipeline
Orchestrates long-running activities that can span multiple build agents. Suitable for building pipelines (formerly known as workflows) and/or organizing complex activities that do not easily fit in free-style job type.

Мультиконфигурационный проект
Подходит для проектов, требующих большое количество различных конфигураций, таких как тестирование в разнообразных средах, платформозависимые сборки и т.д.

Folder
Creates a container that stores nested items in it. Useful for grouping things together. Unlike view, which is just a filter, a folder creates a separate namespace, so you can have multiple things of the same name as long as they are in different folders.

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

Organization Folder
Creates a set of multibranch project subfolders by scanning for repositories.

OK

Выберите эту опцию, если вы хотите создать Item из существующего

29. Добавим скрипт по запуску наших дочерних пайплайнов:

Pipeline

Definition

Pipeline script

Script ?

```
1 #!/groovy
2
3 //Запрещаем одновременное выполнение билдов
4 properties([disableConcurrentBuilds()])
5
6 pipeline {
7     agent any
8     stages {
9         // Запуск пайплайна по сборке, деплою и бэкапу
10        stage("Start pipeline") {
11            steps {
12                echo " ===== Starting pipeline ====="
13                build job: "Build and push", parameters: [{"$class: 'StringParameterValue', name: 'DOCKERIMAGENAME', value: 'fittske/jenkins-docker-hub'}]
14                //Deploy job runs automatically on trigger from build and push job
15                build job: "Backup image", parameters: [{"$class: 'StringParameterValue', name: 'DOCKERIMAGENAME', value: 'fittske/jenkins-docker-hub'}, {"$class: 'StringParameterValue', name: 'REMOTEHOSTIP', value: '158.160.115.211'}]
16            }
17        }
18    }
19 }
20 }
```

Use Groovy Sandbox ?

Pipeline Syntax

30. Запустим и проверим:

The screenshot shows the Jenkins interface for a 'Main pipeline' build #3. The left sidebar includes options like Status, Changes, Console Output (which is selected), View as plain text, Edit Build Information, Delete build '#3', Restart from Stage, Replay, Pipeline Steps, Workspaces, and Previous Build. The main content area is titled 'Выход на консоль' (Console Output) with a green checkmark icon. It displays the log output of the pipeline execution, which starts with 'Started by user Ruslan Zamaletdinov' and ends with 'Finished: SUCCESS'. The log also shows steps for starting the pipeline, building stages, and pushing images.

```

Started by user Ruslan Zamaletdinov
[Pipeline] Start of Pipeline
[Pipeline] properties
[Pipeline] node
Running on Jenkins in /var/lib/jenkins/workspace/Staging/Main pipeline
[Pipeline] {
[Pipeline] stage
[Pipeline] { (Start pipeline)
[Pipeline] echo
===== Starting pipeline =====
[Pipeline] build (Building Staging » Build and push)
Scheduling project: Staging » Build and push
Starting building: Staging » Build and push #3
Build Staging » Build and push #3 completed: SUCCESS
[Pipeline] build (Building Staging » Backup image)
Scheduling project: Staging » Backup image
Starting building: Staging » Backup image #7
Build Staging » Backup image #7 completed: SUCCESS
[Pipeline] }
[Pipeline] // stage
[Pipeline] }
[Pipeline] // node
[Pipeline] End of Pipeline
Finished: SUCCESS

```

31. Проверим dockerhub:

The screenshot shows the Docker Hub website. In the search bar, the repository 'fittske/jenkins-docker-hub' is selected. Below the search bar, there's a 'Search by repository name' field and a 'Create repository' button. The main content area shows the repository details: it's public, has 3 stars, and was last pushed a few seconds ago. It contains one image.

32. Проверим приложение в браузере:

The screenshot shows a Firefox browser window. The address bar shows the URL '158.160.115.211:5000'. The page content displays the text 'Hello, World!'. The browser status bar indicates the date and time: 'Вс, 12 ноября 21:41'.

33. Проверим папку backups:

The screenshot shows a terminal session on a VM named 'jenkins-deploy'. The user runs the command 'll' in the directory '~/backups'. The output shows several files and directories, including two tar archives: 'image-12-11-2023-155652.tar' and 'image-12-11-2023-163611.tar'. The user is identified as 'ruslan'.

```

ruslan@vm-jenkins-deploy:~/backups$ ll
total 2038712
drwxrwxr-x 2 ruslan ruslan 4096 Nov 12 16:37 .
drwxr-x--- 6 ruslan ruslan 4096 Nov 12 15:56 ..
-rw----- 1 root  root 1043811840 Nov 12 15:58 image-12-11-2023-155652.tar
-rw----- 1 root  root 1043811840 Nov 12 16:37 image-12-11-2023-163611.tar
ruslan@vm-jenkins-deploy:~/backups$ 

```

34. Проверим состояние контейнера на ВМ:

The screenshot shows a terminal session on a VM named 'jenkins-deploy'. The user runs the command 'sudo docker ps'. The output lists a single container with ID '290be165b2c7', which is associated with the image 'fittske/jenkins-docker-hub'. The container is running a 'unicorn' command with port mappings and is up for 6 minutes.

```

ruslan@vm-jenkins-deploy:~/backups$ sudo docker ps
CONTAINER ID IMAGE COMMAND CREATED STATUS PORTS NAMES
290be165b2c7 fittske/jenkins-docker-hub "unicorn --bind 0.0..." 6 minutes ago Up 6 minutes 0.0.0.0:5000->5000/tcp, :::5000->5000/tcp great_noether
ruslan@vm-jenkins-deploy:~/backups$ 

```

1. Создадим новую папку для проекта "Production":

Введите имя Item'a

Production
» 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.

 **Pipeline**
Orchestrates long-running activities that can span multiple build agents. Suitable for building pipelines (formerly known as workflows) and/or organizing complex activities that do not easily fit in free-style job type.

 **Мультиконфигурационный проект**
Подходит для проектов, требующих большое количество различных конфигураций, таких как тестирование в разнообразных средах, платформозависимые сборки и т.д.

 **Folder**
Creates a container that stores nested items in it. Useful for grouping things together. Unlike view, which is just a filter, a folder creates a separate namespace, so you can have multiple things of the same name as long as they are in different folders.

 **Multibranch Pipeline**
Creates a set of Pipeline projects according to detected branches in one SCM repository.

 **Organization Folder**
Creates a set of multibranch project subfolders by scanning for repositories.

OK

Выберите эту опцию, если вы хотите создать Item из существующего

2. Для данного проекта создадим пайплайны с настройками, аналогичными проекту "Staging", но везде укажем в качестве ветки - "master":

Branches to build ?

Branch Specifier (blank for 'any') ?

*/master

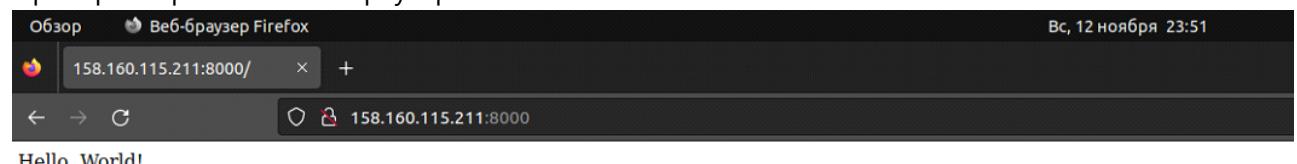
3. Для примера - проверим работу задачи деплоя для проекта "Production" и ветки "master":

The screenshot shows the Jenkins Pipeline Deploy and start configuration page. On the left, there's a sidebar with various options like Status, Changes, and Pipeline Syntax. The main area is titled "Pipeline Deploy and start" and contains a "Stage View" section. This section displays a timeline with two stages: "Declarative: Checkout SCM" (2s) and "Connect to VM via SSH and deploy app" (4s). Below the timeline, a summary says "Average stage times: (Average full run time: ~9s)". To the right of the timeline, there's a "Постоянные ссылки" (Permanent links) section with a link to the last build (#1, 19 ms ago).

4. Проверим состояние контейнеров - продакшен приложение должно запуститься на порту 8000:

```
rustan@vm-jenkins-deploy: $ sudo docker ps
CONTAINER ID IMAGE COMMAND CREATED STATUS PORTS NAMES
9e7477fec2af flitske/jenkins-docker-hub "guncorn --bind 0.0..." 12 seconds ago Up 9 seconds 0.0.0.0:8000->5000/tcp, :::8000->5000/tcp eloquent_feynman
5b6df69c4785 flitske/jenkins-docker-hub "guncorn --bind 0.0..." 29 minutes ago Up 29 minutes 0.0.0.0:5000->5000/tcp, :::5000->5000/tcp eloquent_chebyshev
rustan@vm-jenkins-deploy: $
```

5. Проверим приложение в браузере:



6. Настроим автоматический запуск проекта при подтверждении pull request в ветке "master" - для этого в параметре "Refspec" настроек SCM укажем следующее значение:

Credentials ?

Fittske (github-ssh-key)

Add ▾

Расширенные ^ Edited

Name ?

Refspec ?

+refs/pull/*:refs/remotes/origin/pr/*

A refspec controls the remote refs to be retrieved and how they map to local refs. If left blank, it will default to the normal behaviour of git fetch, which retrieves all the branch heads as remotes/REPOSITORYNAME/BRANCHNAME. This default behaviour is OK for most cases. In other words, the default refspec is "+refs/heads/*:refs/remotes/REPOSITORYNAME/*" where REPOSITORYNAME is the value you specify in the above "name of repository" textbox.

When do you want to modify this value? A good example is when you want to just retrieve one branch. For example, +refs/heads/master:refs/remotes/origin/master would only retrieve the master branch and nothing else.

The plugin uses a default refspec for its initial fetch, unless the "Advanced Clone Option" is set to honor refspec. This keeps compatibility with previous behavior, and allows the job definition to decide if the refspec should be honored on initial clone.

Multiple refspecs can be entered by separating them with a space character. +refs/heads/master:refs/remotes/origin/master +refs/heads/develop:refs/remotes/origin/develop retrieves the master branch and the develop branch and nothing else.

See [the refspec definition in Git user manual](#) for more details.

(from [Git plugin](#))

7. Не забудем указать опрос SCM:

Опрашивать SCM об изменениях ?

Расписание ?

Игнорировать хуки post-commit ?

8. Далее создадим тестовый pull request в ветке "master" и сделаем его merge:

Test commit №6 for auto deploy test #1

[Open](#) Fittske wants to merge 1 commit into `master` from `staging`

Conversation 0 Commits 1 Checks 0 Files changed 1

Fittske commented 5 minutes ago
No description provided.

Test commit №6 for auto deploy test 9799ee3

Add more commits by pushing to the `staging` branch on [Fittske/flask-hello-world](#).

Require approval from specific reviewers before merging
[Branch protection rules](#) ensure specific people approve pull requests before they're merged.
Add rule X

A build service has not been set up
We have detected a top-level `dockerfile`. Pick from [apps](#) that can perform automatic builds.

This branch has no conflicts with the base branch
Merging can be performed automatically.

Merge pull request or view command line instructions.

Reviewers
No reviews
Still in progress? Convert to draft.

Assignees
No one—assign yourself!

Labels
None yet

Projects
None yet

Milestone
No milestone

Notifications
Customize
Unsubscribe
You're receiving notifications because you authored the thread.

The screenshots show the GitHub interface for a pull request titled "Test commit №6 for auto deploy test #1". In the first screenshot, a user named Fittske is merging the pull request from the "staging" branch into the "master" branch. The commit message is "Test commit №6 for auto deploy test". In the second screenshot, the pull request has been successfully merged, and a message indicates that the staging branch can be safely deleted.

9. В Jenkins перейдем в журнал опроса, где увидим, что нашлись изменения:

The Jenkins dashboard shows the "Log опроса Git" section. The "Журнал последнего опроса SCM" (SCM poll log) displays the following output:

```

Started on 13 нояб. 2023 г., 00:47:00
Using strategy: Default
[polll] Last Built Revision: Revision 496b29e2095731262184d2c34668b8140bebc9ca (refs/remotes/origin/master)
Selected Git installation does not exist. Using Default
The recommended git tool is: NONE
using credential github-ssh-key
> git --version # timeout=10
> git --version # 'git version 2.34.1'
using GIT_SSH to set credentials github-ssh-key
> git ls-remote -h -- git@github.com:Fittske/flask-hello-world.git # timeout=10
Found 2 remote heads on git@github.com:Fittske/flask-hello-world.git
[polll] Latest remote head revision on refs/heads/master is: 09153759077a6330615b5a3dc019f90dfdeb8e26
Done. Took 1,9 секунды
Changes found
  
```

The screenshot shows the Jenkins interface for Build #2. The main title is "Build #2 (13 ноября 2023 г., 00:47:12)". A green checkmark icon indicates the build was triggered by a change in the SCM. Below it, there's a message in Russian: "Запущен изменением в SCM". To the right, it shows the revision and repository information: "Revision: 496b29e2095731262184d2c34668b8140bebc9ca" and "Repository: git@github.com:Fittske/flask-hello-world.git". A bullet point lists "refs/remotes/origin/master". On the left sidebar, there are several options: Status, Changes, Console Output, Edit Build Information, Delete build '#2', Log опроса, Параметры, Git Build Data, Restart from Stage, Replay, Pipeline Steps, Workspaces, and Previous Build.

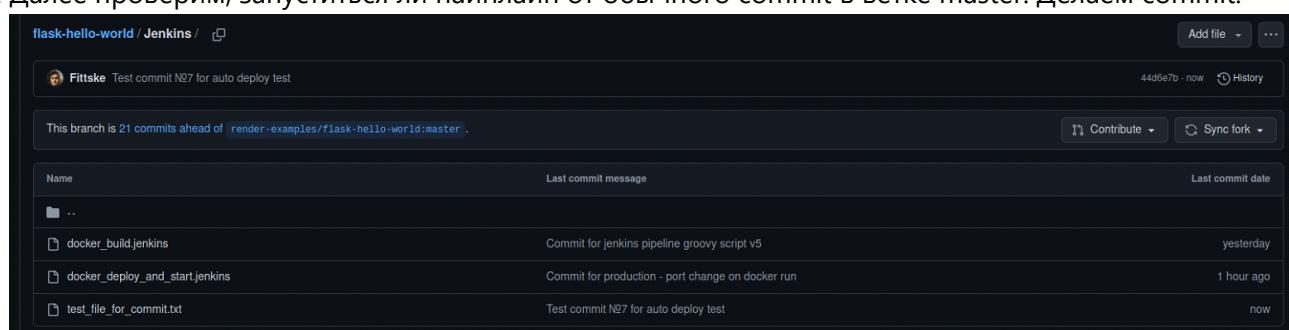
10. Пайп успешно запустился и выполнился, проверяем состояние контейнеров:

```
ruslan@vn-jenkins-deploy:~$ sudo docker ps
CONTAINER ID IMAGE COMMAND CREATED STATUS PORTS NAMES
5406e24bd8f6 fittske/jenkins-docker-hub "gunicorn --bind 0.0..." 2 minutes ago Up 2 minutes 0.0.0.0:8000->5000/tcp, :::8000->5000/tcp blissful_germain
643660a493ab fittske/jenkins-docker-hub "gunicorn --bind 0.0..." 8 minutes ago Up 8 minutes 0.0.0.0:5000->5000/tcp, :::5000->5000/tcp focused_burnell
ruslan@vn-jenkins-deploy:~$
```

11. Проверим в браузере:



12. Далее проверим, запуститься ли пайплайн от обычного commit в ветке master. Делаем commit:



13. Заходим в журнал опроса - Jenkins видит изменения в репозитории, но так как это было не подтверждение pull request, пайплайн не запустился:

Jenkins

Dashboard > Production > Deploy and start > Лог опроса Git

Журнал последнего опроса SCM

Started on 13 нояб. 2023 г., 00:51:00
Using strategy: Default
[poll] Last Built Revision: Revision 496b29e2095731262184d2c34668b8140bebc9ca (refs/remotes/origin/master)
Selected Git installation does not exist. Using Default
The recommended git tool is: NONE
using credential github-ssh-key
> git --version # timeout=10
> git --version # 'git version 2.34.1'
using GIT_SSH to set credentials github-ssh-key
> git ls-remote -h -- git@github.com:Fittske/flask-hello-world.git # timeout=10
Found 2 remote heads on git@github.com:Fittske/flask-hello-world.git
Ignoring refs/heads/master as it doesn't match any of the configured refspecs
Ignoring refs/heads/staging as it doesn't match any of the configured refspecs
Done. Took 1,9 секунды
No changes

Лог опроса Git

История сборок **тренд**

Фильтровать сборки... /

#2 13 нояб. 2023 г., 00:47
#1 12 нояб. 2023 г., 23:50

Jenkins

Dashboard > Production > Deploy and start > Лог опроса Git

Журнал последнего опроса SCM

Started on 13 нояб. 2023 г., 00:52:00
Using strategy: Default
[poll] Last Built Revision: Revision 496b29e2095731262184d2c34668b8140bebc9ca (refs/remotes/origin/master)
Selected Git installation does not exist. Using Default
The recommended git tool is: NONE
using credential github-ssh-key
> git --version # timeout=10
> git --version # 'git version 2.34.1'
using GIT_SSH to set credentials github-ssh-key
> git ls-remote -h -- git@github.com:Fittske/flask-hello-world.git # timeout=10
Found 2 remote heads on git@github.com:Fittske/flask-hello-world.git
Ignoring refs/heads/master as it doesn't match any of the configured refspecs
Ignoring refs/heads/staging as it doesn't match any of the configured refspecs
Done. Took 1,9 секунды
No changes

Лог опроса Git

История сборок **тренд**

Фильтровать сборки... /

#2 13 нояб. 2023 г., 00:47
#1 12 нояб. 2023 г., 23:50