

Введение

В данной работе можно будет познакомиться со способом создания CI пайпланов в Jenkins на языке Groovy.

Несколько полезных ссылок:

1. [Для чего применяется Jenkins](#)
2. [Jenkins Pipeline. Что это и как использовать в тестировании](#)
3. [Учимся разворачивать микросервисы](#)

Gitlab позволяет использовать для реализации CI/CD не только внутреннюю систему, но и сторонние сервисы. Но следует помнить, **что в один и тот же момент может и должна работать только одна система CI.**

Поэтому, перед переключением пайплайна сборки для проекта с Gitlab CI на Jenkins необходимо сделать несколько шагов.

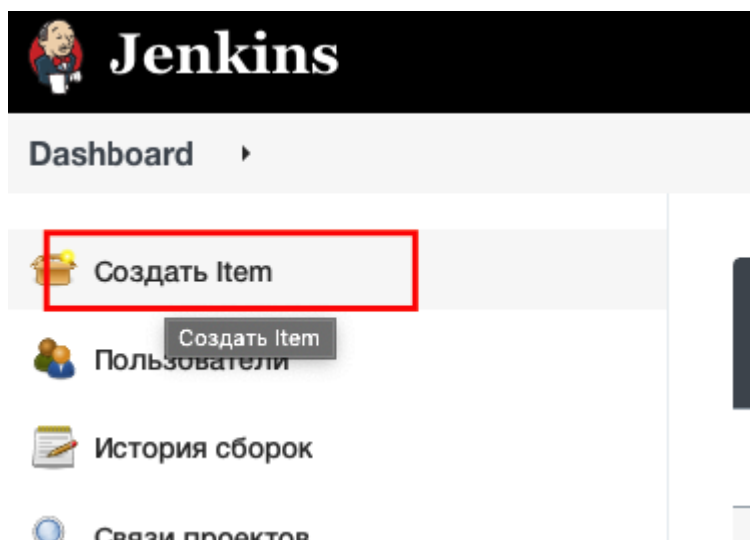
Во первых, и самое главное, перед началом работы с Jenkins pipeline **НЕОБХОДИМО** удалить файл `.gitlab-ci.yml`, чтобы отключить выполнение Gitlab CI.

Шаг 1. Создание Jenkins Item

В Jenkins все проекты CI/CD называются Item. Item может быть нескольких видов, на же интересует тип Pipeline (вы можете в свободное время изучить другие типы).

Так как имя Item в Jenkins должно быть уникальным, соблюдайте формат его названия: `lab4_{Группа}_{Фамилия}`!

Переходим по ссылке: <http://jenkins.devops.ru/>, далее создаем item.



Попадаем на следующую страницу и создаем пайнлайн

Введите имя Item'a

ИСПОЛЬЗУЕМ ФОРМАТ НАЗВАНИЯ, ПРИВЕДЕННЫЙ ВЫШЕ

» Обязательное поле

Создать задачу со свободной конфигурацией
 Это - основной и наиболее универсальный тип задач в Jenkins. Jenkins будет собирать ваш проект, комбинируя различные задачи, отличные от сборки ПО.

Pipeline
 Orchestrates long-running activities that can span multiple build agents. Suitable for building pipelines (formerly known as Jenkinsfile).

Шаг 2. Настройка Item

Теперь необходимо выставить все доступы для своего аккаунта

☒ Enable project-based security

Inheritance Strategy

Inherit permissions from parent ACL

This item will inherit its parent item's permissions (in addition to any permissions granted here). If this item is at the top level in Jenkins, it will inherit the [global security settings](#).

Пользователь/группа	Credentials		Задача										Запуск			Система контроля версий	
	Create	Delete	ManageScripts	Update	View	Build	Cancel	Configure	Delete	Discover	Move	Read	Workspace	Delete	Replay	Update	Tag
Аноним	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Authenticated Users	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Nikolay Vedernikov	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>

Add user... Add group...

Далее выставляем отмеченные галочки и значения.

GitLab Connection

Gitlab

- ☐ Use alternative credential
- ☐ Pipeline speed/durability override
- ☐ Preserve stashes from completed builds
- ☐ Throttle builds
- ☒ Удалять устаревшие сборки

Strategy

Log Rotation

Сколько дней хранить результаты сборки

Если указано, информация о сборках будет храниться это количество дней.

Сколько последних сборок хранить

1

Если указано, будет храниться информация об этом количестве сборок.

Расширенные...

☐ Это - параметризованная сборка

Build Triggers

- ☐ Build after other projects are built
- ☐ Запускать периодически
- ☒ Build when a change is pushed to GitLab. GitLab webhook URL: http://jenkins.devops.ru/project/lab4_example

Enabled GitLab triggers

- ☒ Push Events
- ☐ Push Events in case of branch delete
- ☒ Opened Merge Request Events
- ☐ Build only if new commits were pushed to Merge Request
- ☒ Accepted Merge Request Events
- ☐ Closed Merge Request Events

Rebuild open Merge Requests

Never

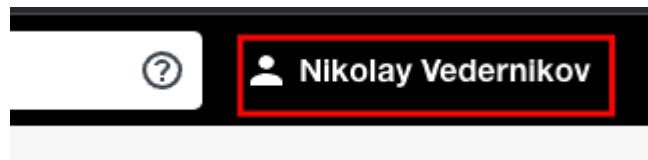
- ☒ Approved Merge Requests (EE-only)
- ☒ Comments

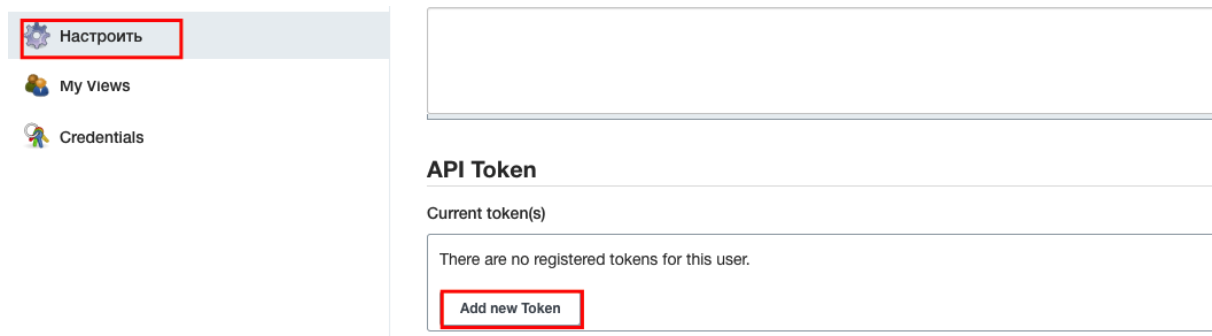
Comment (regex) for triggering a build

Jenkins please retrv a build

Далее нажимаем на кнопку сохранить.

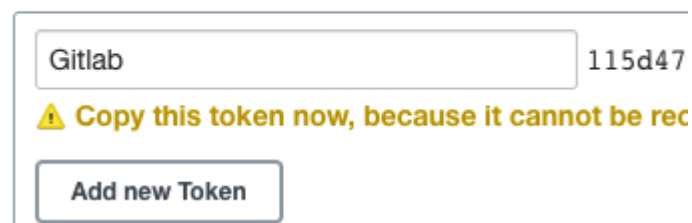
Шаг 3. Создание Jenkins API Token



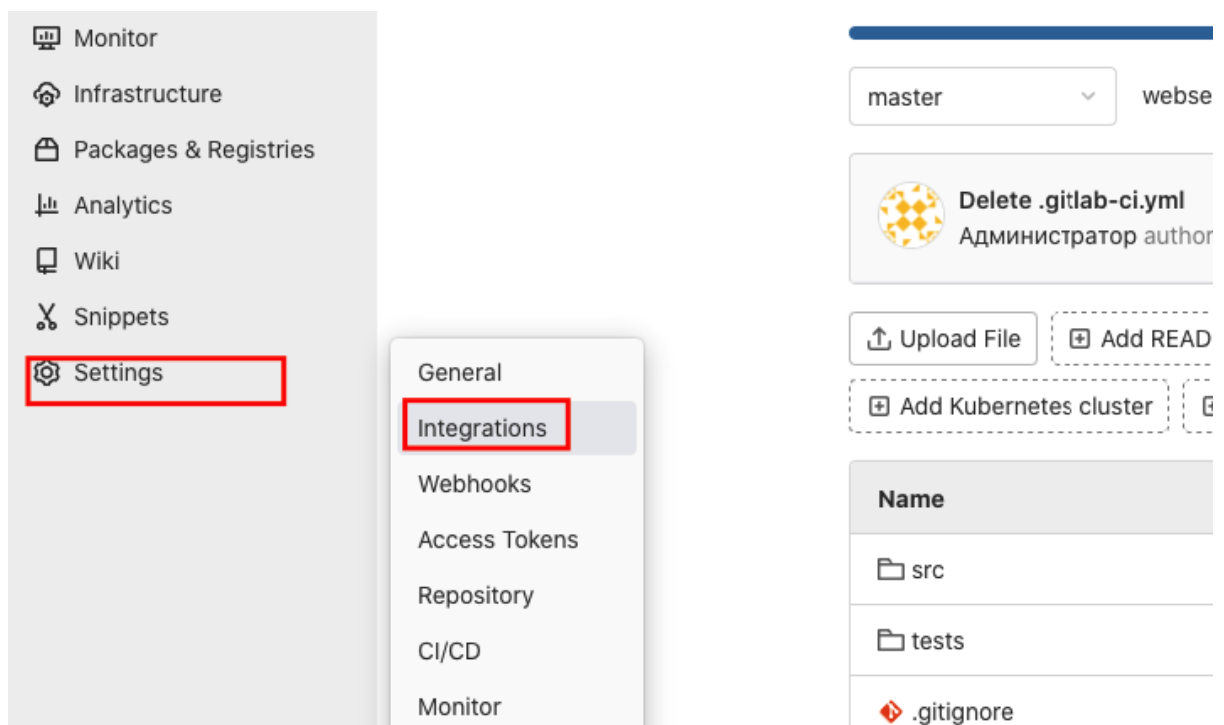


API Token

Current token(s)



Шаг 4. Включение интеграции в проекте Gitlab



EWM	Use IBM Engineering Workflow Management as this project's issue tracker.
Emails on push	Email the commits and diff of each push to a list of recipients.
External wiki	Link to an external wiki from the sidebar.
Flowdock	Send event notifications from GitLab to Flowdock flows.
Google Chat	Send notifications from GitLab to a room in Google Chat.
Jenkins	Run CI/CD pipelines with Jenkins.
JetBrains TeamCity	Run CI/CD pipelines with JetBrains TeamCity.
Jira	Use Jira as this project's issue tracker.
Mattermost notifications	Send notifications about project events to Mattermost channels.
Mattermost slash commands	Perform common tasks with slash commands.

В поле пароля вводим скопированный в Jenkins токен (см. стр. 4) и нажимаем на “Test settings”, **статус запроса должен быть 200**. Далее сохраняем изменения.

Enable integration

☒ Active

Trigger

☒ Push

Trigger event for pushes to the repository.

☒ Merge Request

Trigger event when a merge request is created, updated, or merged.

☒ Tag Push

Trigger event for new tags pushed to the repository.

Jenkins server URL

http://jenkins.devops.ru/

The URL of the Jenkins server.

SSL verification

☐ Enable SSL verification

Clear if using a self-signed certificate.

Project name

lab4_example (НАЗВАНИЕ ВАШЕГО ITEM)

The name of the Jenkins project. Copy the name from the end of the URL to the project.

Username

imm0bilize (БАШ НИК НА GITLAB)

The username for the Jenkins server.

Enter new password.

Leave blank to use your current password.

Save changes

Test settings

Cancel

Status	Trigger	Elapsed time	Request time	
200	Push Hook	0.06 sec	just now	View details

Шаг 4. Создание Jenkins Pipeline скрипта

К этому шагу мы сделали все подготовительные этапы, добавим сам скрипт для CI пайплайна.

Dashboard

lab4_example

Back to Dashboard

Status

Changes

Собрать сейчас

Настройки


Удалить Pipeline

Full Stage View

Rename

Pipeline Syntax

Pipeline lab4_example

 Recent Changes

Stage View

This Pipeline has run successfully, but does not define any stages. Please use the `stage` step to define some stages in this Pipeline.

Постоянные ссылки

Pipeline

Definition

Pipeline script

Script

```
1 pipeline {
2   agent {
3     docker {
4       image 'python:3.8'
5     }
6   }
7   environment {
8     HOME = "${env.WORKSPACE}@tmp"
9     BIN_PATH = "${HOME}/.local/bin/"
10  }
11  stages {
12    stage('Git Clone') {
13      steps {
14        git changelog: false, url: 'http://gitlab.devops.ru/immobilize/webseviceexample.git'
15      }
16    }
17    stage('Prepare') {
18      steps {
19        sh 'python --version'
20        sh 'pip install virtualenv'
21        sh "${BIN_PATH}virtualenv venv"
22        sh 'bash -c "source venv/bin/activate"'
23        sh 'pip install -r requirements.txt'
24      }
25    }
26    stage('Test') {
27      steps {
28        sh 'python -m unittest discover -s "./tests" -p "*_test.py"'
29        sh "${BIN_PATH}flake8 ."
30        sh "${BIN_PATH}mypy ."
31      }
32    }
33  }
34 }
```

```

pipeline {
    agent {
        docker {
            image 'python:3.8'
        }
    }
    environment {
        HOME = "${env.WORKSPACE}@tmp"
        BIN_PATH = "${HOME}/.local/bin/"
    }
    stages {
        stage('Git Clone') {
            steps {
                git changelog: false, url: "ссылка на ваш репозиторий"
            }
        }
        stage('Prepare') {
            steps {
                /* аналогично подготовительному этапу во 3 л.р.*/
                sh 'python --version'
                sh 'pip install virtualenv'
                sh "${BIN_PATH}virtualenv venv"
                sh 'bash -c "source venv/bin/activate"'
                sh 'pip install -r requirements.txt'
            }
        }
        stage('Test') {
            steps{
                /* меняем запуск теста, который был в 3 л.р.*/
                sh 'python -m unittest discover -s "./tests" -p "*_test.py"'
                sh "${BIN_PATH}flake8 ."
                sh "${BIN_PATH}mypy ."
            }
        }
    }
}

```


После чего мы можем нажать на кнопку “Собрать сейчас”, которая запустит созданный пайплайн. Справа мы увидим таблицу, показывающую результат каждого из созданных этапов в сборке.

The screenshot shows the GitLab CI/CD interface. On the left sidebar, the 'Собрать сейчас' (Build now) button is highlighted with a red box. The main area displays the 'Stage View' for a pipeline. It includes a table with stage names and their durations, and a list of recent builds.

	Git Clone	Prepare	Test
Average stage times: (Average full run time: ~31s)	823ms	17s	5s
#2 Mar 29 16:13 No Changes	823ms	17s	5s
#1 Mar 29 15:47 No Changes			

Шаг 5. Перенос Jenkins Pipeline в Jenkinsfile

Рассмотренный ранее способ не самый популярный при работе с Jenkins. В отличие от gitlab-ci, разработчик не может при необходимости быстро внести изменения в описанный пайплайн. Для решения этой проблемы существует Jenkinsfile. Фактически в него мы просто переносим скрипт из Item. Из дополнительных плюсов такого подхода, мы можем удобно создать уникальный Jenkinsfile для каждой ветки нашего проекта, если на то есть необходимость.

The screenshot shows the Jenkins web interface. At the top, there's a dropdown menu for 'master' and a text field for 'webserviceexample /'. Below this, there's a section for 'Delete .gitlab-ci.yml' with a button 'Администратор authored 1 day ago'. In the center, there are buttons for 'Upload File', 'Add README', 'Add LICENSE', 'Add CHANGELOG', and 'Add C'. At the bottom, there are buttons for 'Add Kubernetes cluster', 'Set up CI/CD', and 'Configure Integrations'. On the right side, there's a 'Clone' dropdown menu with options for 'Clone with SSH', 'Clone with HTTP', and 'Open in your IDE'. The 'Clone with HTTP' option is highlighted with a blue box, showing the URL 'http://gitlab.devops.ru/immobilize'.

Изменим параметры Item для работы с Jenkinsfile.

Pipeline script from SCM

SCM ?

Git

Repositories ?

Repository URL ?

http://gitlab.devops.ru/imm0bilize/webserviceexample.git

Credentials ?

- none - Add

Расширенные...

Add Repository

Branches to build ?

Branch Specifier (blank for 'any') ?

*/master

Add Branch

Просмотрщик репозитория ?

(Автоматически)

Additional Behaviours

Добавить

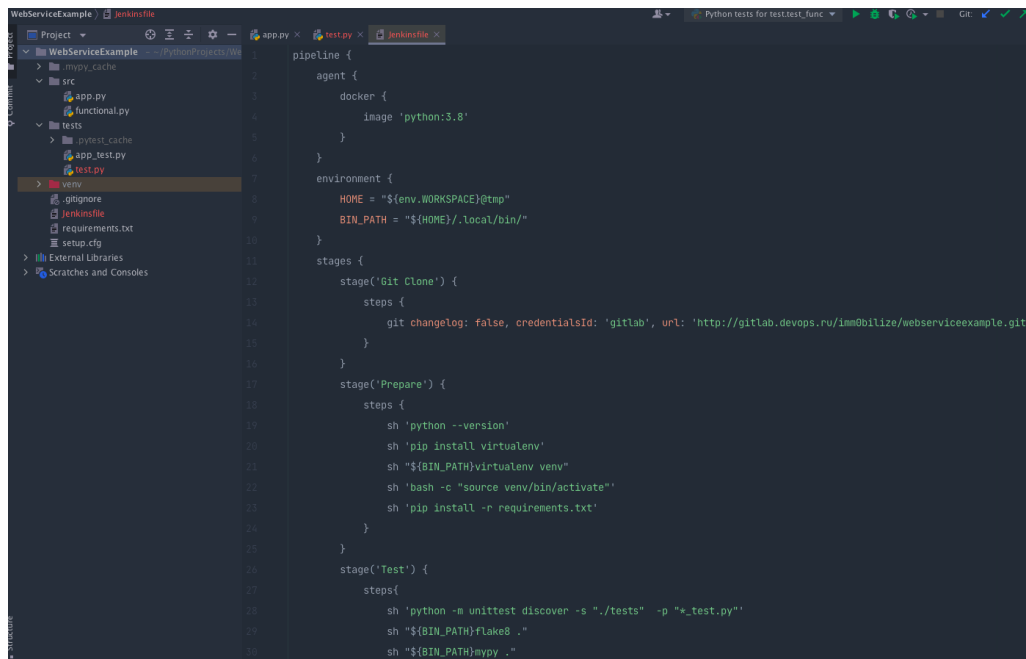
Script Path ?

Jenkinsfile

☒ Lightweight checkout ?

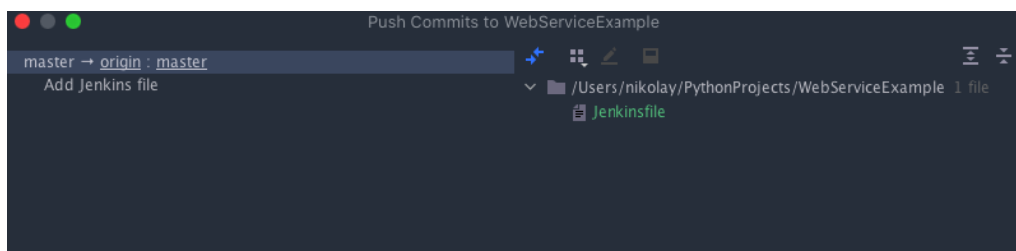
[Pipeline Syntax](#)

Создаем файл “Jenkinsfile” и переносим в него содержимое нашего скрипта из Item.

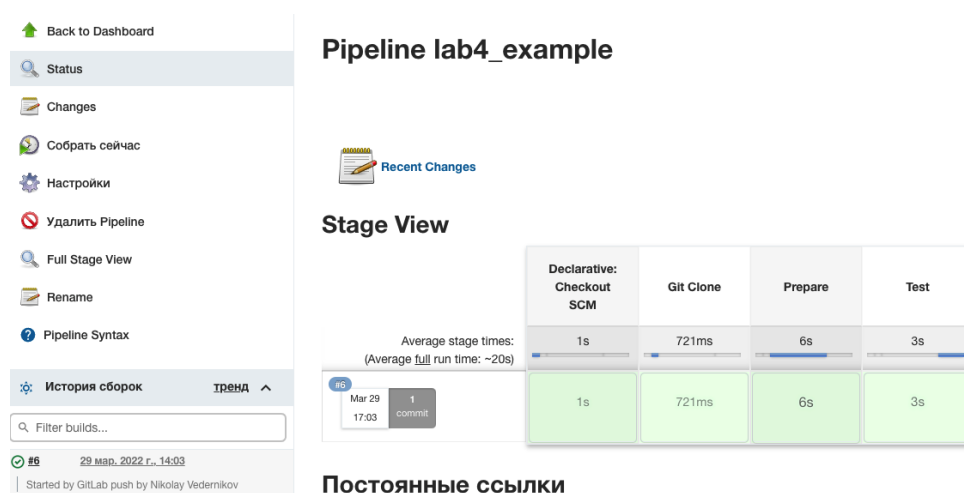


```
1 pipeline {
2   agent {
3     docker {
4       image 'python:3.8'
5     }
6   }
7   environment {
8     HOME = "${env.WORKSPACE}@tmp"
9     BIN_PATH = "${HOME}/.local/bin/"
10  }
11  stages {
12    stage('Git Clone') {
13      steps {
14        git changelog: false, credentialsId: 'gitlab', url: 'http://gitlab.devops.ru/immobilize/web-serviceexample.git'
15      }
16    }
17    stage('Prepare') {
18      steps {
19        sh 'python --version'
20        sh 'pip install virtualenv'
21        sh "${BIN_PATH}virtualenv venv"
22        sh 'bash -c "source venv/bin/activate"'
23        sh 'pip install -r requirements.txt'
24      }
25    }
26    stage('Test') {
27      steps {
28        sh 'python -m unittest discover -s "./tests" -p "*_test.py"'
29        sh "${BIN_PATH}flake8 ."
30        sh "${BIN_PATH}mypy ."
```

Делаем коммит и пуш в наш репозиторий.



Jenkins успешно запустил пайплайн по сценарию из Jenkinsfile, и он завершился успешно.



Pipeline lab4_example

Recent Changes

Stage View

Declarative: Checkout SCM	Git Clone	Prepare	Test
1s	721ms	6s	3s

Average stage times: (Average full run time: ~20s)

1 commit

Постоянные ссылки