

Jenkins

- [Установка](#)
- [Jenkins -> k8s](#)
- [Разное](#)
- [Интеграция Jenkins и GitVerse](#)
- [Ssh агент](#)

Установка

Jenkins server, docker compose

Источник

- Создать директорию jenkins_compose

```
mkdir jenkins_compose && cd jenkins_compose
```

- Создать compose файл

```
nano docker-compose.yaml
```

- Скопировать в файл текст, сохранить и выйти

```
services:
  jenkins:
    image: jenkins/jenkins:lts
    privileged: true
    user: root
    ports:
      - 8080:8080
      - 50000:50000
    container_name: jenkins
    volumes:
      - /home/${myname}/jenkins_compose/jenkins_configuration:/var/jenkins_home
      - /var/run/docker.sock:/var/run/docker.sock
```

- Запустить образ

```
docker compose up
```

- С данной конфигурацией по <http://localhost:8080> будет находиться web интерфейс. При первом старте в консоль будет выдан ключ инициализации, который нужно скопировать в web форму.
- Jenkins для обучения работает!

Для обучения на этом можно остановиться.

Jenkins agent, docker compose

- В директории jenkins_compose создаем ключи

```
ssh-keygen -t rsa -f jenkins_agent
```

- Перейти

Настройка Jenkins

Есть подозрение, что настройки вашего обратного прокси некорректны.

Выполнять сборки на встроенном узле может быть небезопасно. Рекомендуется настроить распределенные сборки. См. документацию.

System Configuration

- System**: Конфигурация глобальных настроек и путей.
- Tools**: Конфигурация инструментов, их расположение и автоматическая установка.
- Plugins**: Добавить, удалить, включить плагины функционала Jenkins.
- Clouds**: Add, remove, and configure cloud instances to provision agents on-demand.
- Appearance**: Configure the look and feel of Jenkins

Security

- Security**: Настройка безопасности Jenkins, конфигурация прав для доступа и использования системы.
- Credentials**: Configure credentials (highlighted with a red arrow)
- Credential Prov**: Configure the credential types

- Перейти в системное хранилище - глобальные параметры

Stores scoped to Jenkins

Store	Domains
System	(global)

System

Domain

Global credentials (unrestricted)

- Добавить способ авторизации со следующими настройками (ключ ввести вручную)

Scope ?

System (Jenkins and nodes only)

ID ?

jenkins_agent

Description ?

For agen conn

Username


jenkins

☐ Treat username as secret ?

Private Key

☒ Enter directly

Key

 Concealed for Confidentiality

Passphrase

- Теперь добавить в compose файл образ агента с открытым ключом

```
services:
  jenkins:
    image: jenkins/jenkins:lts
    privileged: true
    user: root
    ports:
      - 8080:8080
      - 50000:50000
    container_name: jenkins
    volumes:
      - /home/${myname}/jenkins_compose/jenkins_configuration:/var/jenkins_home
      - /var/run/docker.sock:/var/run/docker.sock
  agent:
```

```
image: jenkins/ssh-agent
privileged: true
user: root
container_name: agent
expose:
  - 22
environment:
  - JENKINS_AGENT_SSH_PUBKEY=ssh-rsa...
```

- Почти все прошло по инструкции, кроме путей java и версии java. Второе было исправлено в compose (сейчас актуальная версия). А для установки корректного пути к JAVA нужно было найти в контейнере путь java

```
sergey@sergey-VirtualBox:~$ docker exec -it 2b5036c4a35b /bin/bash
root@2b5036c4a35b:/home/jenkins# env | grep JAVA
JAVA_HOME=/opt/java/openjdk
```

И установить следующий путь в настройках агента:

Расширенные ^  Edited

Port ?

22

JavaPath ?

/opt/java/openjdk/bin/java

- Агент поднялся!

Jenkins -> k8s

[Видео по настройке взаимодействия](#)

[Описание плагина Kubernetes](#)

[Описание плагина Credentials](#)

[Детальное описание Pod](#)

[Шпаргалка по k8s](#)

- Есть Jenkins вне кластера. Дополнительно установить плагин Kubernetes. Остальные нужные плагины подтянутся сами (Kubernetes client API, ...).

Kubernetes plugin 4313.va_9b_4fe2a_0e34

This plugin integrates Jenkins with Kubernetes

[Report an issue with this plugin](#)

- Есть кластер k8s с доступным для Jenkins внешним адресом. Внутри кластера есть одна нода с доступом в Интернет. Доступ с ноды для приложенного pipeline критичен из-за скачивания исполняемых файлов, если эти файлы есть внутри кластера - все еще лучше.
- На кластере создаем namespace, пользователя jenkins, токен

```
kubectl create namespace jenkins
```

```
kubectl create sa jenkins -n jenkins
```

```
kubectl create token jenkins -n jenkins --duration=8760h
```

#здесь отобразится токен, его нужно сохранить, затем добавим в настройки Cloud

```
kubectl create rolebinding jenkins-admin-binding --clusterrole=admin --
```

```
serviceaccount=jenkins:jenkins --namespace=jenkins
```

- //еще не полностью понял// Под термином Cloud в Jenkins понимается поддерживаемые типы виртуализации/контейнеризации для создания динамических агентов. Т е авторизация и т д, настраиваемая в Clouds, используется только для создания динамического агента в соответствующем типе виртуализации и затем удаления его. Все. Агент создается пустой, все остальное нужно доделывать. При установке соответствующих плагинов (Docker, Kubernetes, Virtualbox) в списке появляются соответствующие типы облаков.

- Для использования в динамическом агенте настроим авторизацию. Настроить Jenkins - Credentials - System
- Выбрать тип Secret text, в обычно одинаковый.

New credentials

ый токен, ID и Description

Kind

Secret text

Scope ?

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

Secret

ID ?

forkuber

Description ?

forkuber

Create

- Добавляем Cloud. Name это имя, любое. Kubernetes URL - адрес кластера, по которому Jenkins может подключиться. Credentials - созданный на предыдущем шаге.

Name ?

Kuber

Kubernetes URL ?

https://192.168.1.185:6443

☐ Use Jenkins Proxy ?

Kubernetes server certificate key ?

☒ Disable https certificate check ?

Kubernetes Namespace

jenkins

Agent Docker Registry ?

☐ Inject restricted PSS security context in agent container

Credentials

kubescli

+ Add

- Справа есть кнопка Test connection Обязательно проверить соединение.
- Добавить стоит только адрес Jenkins сервера и Web Socket. Все остальное по умолчанию.

☒ WebSocket ?

☐ Direct Connection ?

Jenkins URL ?

http://192.168.1.194:8080/

Jenkins tunnel ?

Connection Timeout ?

5

- Настройка основы для динамического Jenkins клиента завершена.

Для работы с kubernetes нужен файл авторизации. Формат файла и настройки админа (для примера) можно посмотреть на кластере Kubernetes в

```
/etc/kubernetes/admin.conf
```

Сохраняем его в Credentials Jenkins и тип - файл. В данном случае ID textauth.



textauth

fuck.txt (textauth)

Secret file

textauth



Важно, что рабочая директория Jenkins может (и скорее всего будет) отличаться от homedir пользователя

Pipeline:

```
pipeline {
  agent {
    kubernetes (kubernetesAgent(name: 'mini'))
  }

  stages {
    stage('Integrate Remote k8s with Jenkins ') {
      steps {
        sh "mkdir ~/.kube"
        withCredentials([file(credentialsId: 'textauth', variable: 'FILE')]) {
          sh 'cp $FILE ~/.kube/config'
        }
        sh "curl -LO 'https://dl.k8s.io/release/v1.32.2/bin/linux/amd64/kubectl'"
        sh "chmod +x kubectl"
        sh "./kubectl get nodes"
      }
    }
  }
}
```

Разное

```
node('docker-agent-01') {
    stage('Pulling repo and filling constants'){
        sh "rm *"
        git 'https://gitverse.ru/bobrobot/liteconnect_docker.git'
        withEnv(readFile('env.txt').split("\n") as List) {
            sh 'chmod +x replace_env.sh && ./replace_env.sh'
        }
        sh "cp env.txt .env"
    }
}
```

```
node() {
    stage('first') {
        withEnv(["PATH+HUIWAM=${HOME}/.local/bin"])
        {
            sh "printenv PATH"
        }
        sh "printenv PATH"
    }
}
```

```
withEnv([
    "PATH+EXTRA=/home/jenkins/.local/bin"])
{
    node() {
        stage('first') {
            sh "printenv PATH"
        }
    }
}
```

```
pipeline {
    agent {
```

```

    kubernetes (kubernetesAgent(name: 'mini'))
}
environment {
    PATH="${HOME}/.local/bin:${PATH}"
}
stages {
    stage('Integrate Remote k8s with Jenkins ') {
        steps {
            sh "printenv PATH"
            withCredentials([file(credentialsId: 'mysrc', variable: 'MYFILE')]) {
                sh 'cp $MYFILE new.sh'
            }
        }
    }
}
}
}

```

```

// для груви скрипта нужно упаковывать в withenv чтобы были видны определенные в скрипте
параметры,
//параметры из jenkins видны
//param_in_main - текстовый параметр,
//build_type - второй параметр.
withEnv([
    'ENV1=It work ' + param_in_main,
    'ENV2=ooo',
    'missingfname=thefilereallymissing.txt'])
{
    node('slave') {
        stage('first') {
            sh "mkdir ~/bin"
            sh "printenv PATH"
            sh "echo $param_in_main" //обращение к параметру из параметров в jenkins
            sh "echo $ENV1" //обращение к составному параметру, определенному внутри скрипта
            sh "echo $ENV2" //обращение к независимому параметру внутри скрипта
        }

        stage('check type of pipeline'){
            if (build_type == 'prod') { //если проверяем вне sh - доллар не ставится
                sh "echo 'This is a prod'"
            }
        }
    }
}

```

```

}

stage('test exeptions'){
    try {
        sh 'rm $missingfname'
    }
    catch (exc) {
        echo 'Попытались удалить несуществующий файл ' + missingfname
    }
}

stage('check cycle'){
    sh 'mkdir "firstdir"'
    def fileNames = sh(returnStdout: true, script: 'ls').trim().split()
    for (item in fileNames) {
        echo "Processing item: ${item}"
    }
}
}
}

```

```

def buildPod = {
    def podYaml = """
kind: Pod
metadata:
  name: jenkins-agent
spec:
  containers:
  - name: my-jenkins-agent
    image: jenkins/inbound-agent:latest
    command:
    - cat
    tty: true
    volumeMounts:
    - mountPath: /var/run/docker.sock
      name: docker-sock
  volumes:
  - name: docker-sock
    hostPath:
      path: /var/run/docker.sock

```

```
""""

    return podYaml
}

def label = "jenkins-agent-${UUID.randomUUID().toString()}"

podTemplate(label: label, yaml: buildPod()) {
    node(label) {
        stage('Build') {
            container('my-jenkins-agent') {
                echo 'Start stage build'
                sh 'ls -al'
                echo 'End stage build'
            }
        }
        stage('Deploy') {
            container('my-jenkins-agent') {
                echo 'Start stage deploy'
                try {
                    sh "rm my.txt"
                }
                catch(exc) {
                    echo "The ERROR!"
                    echo "${exc}"
                }
            }
        }
    }
}
```

Интеграция Jenkins и GitVerse

Настройка доступа в GitVerse

Профиль -> Управление токенами, создаем токен и сохраняем его

Создать новый токен

Jenkins integration for CI/CD

Доступ к функционалу

Чтение

Запись

Удаление

☒ Пакеты

☒

☒

☒

☒ Репозитории

☐

☐

☐

Генерировать токен

Добавляем токен доступ в Jenkins

Home -> Настроить Jenkins -> Credentials Затем System -> global credentials -> Add credential

Kind

Secret text

Scope ?

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

Secret

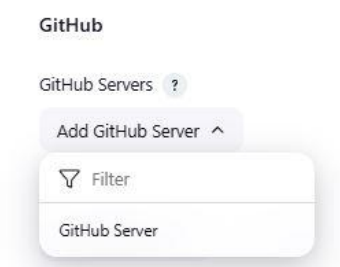
ID ?

Description ?

Create

Настройка доступа к GitVerse

Home -> Настроить Jenkins -> System



Создаем в проекте Jenkinsfile

Создаем новый pipeline в Jenkins

Новый Item

Введите имя Item'a

Docker images creation

Select an item type



Создать задачу со свободной конфигурацией

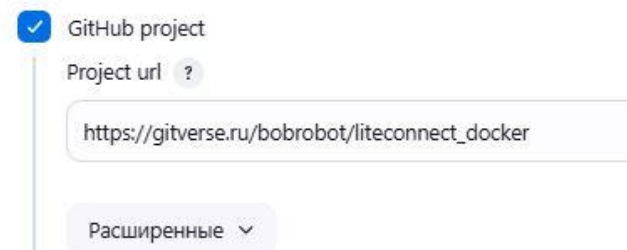
Classic, general-purpose job type that checks out from up to one SCM steps like archiving artifacts and sending email notifications.



Pipeline

Orchestrates long-running activities that can span multiple build agent workflows) and/or organizing complex activities that do not easily fit i

Устанавливаем github project



В разделе Triggers установить GitHub hook trigger

Triggers

Set up automated actions that start your build based on specific events

- ☐ Запустить по окончании сборки других проектов ?
- ☐ Запускать периодически ?
- ☒ GitHub hook trigger for GITScm polling ?

Устанавливаем тип Pipeline from SCM

Pipeline

Define your Pipeline using Groovy directly or pull it from source control.

Definition

Pipeline script from SCM

SCM ?

Git

Repositories ?

Repository URL ?

https://gitverse.ru/bobrobot/liteconnect_docker

Credentials ?

- none -

+ Add

В первый раз сделать сборку вручную. После успешной сборки перейти в Settings пайплайна и нажать Save

Получилось собирать из Jenkinsfile в проекте, но webhook не поехал. Сервер снаружи виден. Поэтому пока оставим ручной запуск на исполнение.

Ssh агент

SSH Agent plugin нахер в данном случае не нужен. Он необходим только для подключения внутри stage с существующего агента к другому серверу.

Сервер, на котором будем выполнять команды:

Добавить пользователя jenkins и добавить нового пользователя в группу sudo, docker (в случае моей задачи)

```
sudo adduser jenkins  
sudo usermod -aG sudo jenkins  
sudo usermod -aG docker jenkins
```

Авторизоваться на сервере под пользователем jenkins

Установить java

```
sudo apt update  
sudo apt install fontconfig openjdk-17-jre
```

Создать приватный и публичный ключи для доступа к серверу по ssh пользователя jenkins.

```
mkdir ~/.ssh && cd ~/.ssh
```

```
ssh-keygen -t rsa -C "Access key for Jenkins slaves"
```

```
cat id_rsa.pub >> ~/.ssh/authorized_keys
```

Ключ из файла id_rsa потребуется для настройки доступа Jenkins

Jenkins web интерфейс:

Настройки Jenkins -> Credentials -> System -> Global credentials -> Add credential

New credentials

Kind

SSH Username with private key

Scope ?

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

ID ?

Jenkins

Description ?

For access from jenkins username

Username

jenkins

☐ Treat username as secret ?

Private Key

☒ Enter directly

Key

```
|-----BEGIN OPENSSH PRIVATE KEY-----  
b3B1bnNzaC1rZXktdjEAAAABAG5vbmUAAAABbm9uZQAAAAAAAAABAAABlwAAAAdzc2gtcn  
NhAAAAAwEAAQAAAYEakvJS4Ld7K4kiISUVo8fBDoxUxx+Y7pvOt5k6mp0e100sZDJu04gP
```

В раздел ключ вставить текст из файла id_rsa целиком (включая -----BEGIN... -----END...)

Перейти в Настроить Jenkins -> Nodes и создать узел

Название узла

jenkins-debdocker-agent

Type

- ☒ Постоянный агент
- Добавляет в Jenkins простой, п
этим агентами, например, ди
физический компьютер, вирту:

Создать

Параметры агента:

Параметр	Значение
Удаленная корневая директория	/home/jenkins/jenkins-agent
Метки	docker-agent-01
Host key verification strategy	Manually trusted key verification

Имя ?

jenkins-debdocker-agent

Описание ?

Plain text [Предпросмотр](#)

Количество процессов-исполнителей ?

1

Удалённая корневая директория ?

/home/jenkins/jenkins-agent

Метки ?

docker-agent-01

Использование ?

Загружать этот узел настолько, насколько возможно

Способ запуска ?

Launch agents via SSH

Host ?

192.168.1.10

Credentials ?

jenkins (For access from jenkins username)

+ Add

После настройки в узлах агент должен быть активен.

Тестовый pipeline

```
node('docker-agent-01') {  
    stage ('second'){  
        sh 'hostname'  
    }  
}
```

```
}
```

должен вывести хост агента.

В Jenkins текущая рабочая директория -

```
/home/jenkins/jenkins-agent/workspace/<имя pipeline>
```