Лабораторная работа

Тема: Работа с Jenkins.

Теоретическая часть

Jenkins — программная система с открытым исходным кодом на Java, предназначенная для обеспечения процесса непрерывной интеграции программного обеспечения. Ответвлена в 2008 году от проекта Hudson, принадлежащего компании Oracle. Позволяет автоматизировать часть процесса разработки программного обеспечения, в котором не обязательно участие человека, обеспечивая функции непрерывной интеграции. Работает в сервлет-контейнере, например, Apache Tomcat. Поддерживает инструменты системы управления версиями, включая AccuRev, CVS, Subversion, Git, Mercurial, Perforce, Clearcase и RTC. Может собирать проекты с использованием Apache Ant и Apache Maven, а также выполнять произвольные сценарии оболочки и пакетные файлы Windows. Сборка может быть запущена разными способами, например, по событию фиксации изменений в системе управления версиями, по расписанию, по запросу на определённый URL, после завершения другой сборки в очереди.

Непрерывная интеграция (CI, англ. Continuous Integration) — практика разработки программного обеспечения, которая заключается в постоянном слиянии рабочих копий в общую основную ветвь разработки (до нескольких раз в день) и выполнении частых автоматизированных сборок проекта для скорейшего выявления потенциальных дефектов и решения интеграционных проблем. В обычном проекте, где над разными частями системы разработчики трудятся независимо, стадия интеграции является заключительной. Она может непредсказуемо задержать окончание работ. Переход к непрерывной интеграции позволяет снизить трудоёмкость интеграции и сделать её более предсказуемой за счёт наиболее раннего обнаружения и устранения ошибок и противоречий, но основным преимуществом является сокращение стоимости исправления дефекта, за счёт раннего его выявления.

Непрерывная доставка (англ. Continuous delivery или CD, или CDE) — это подход к разработке программного обеспечения, при котором программное обеспечение производится короткими итерациями, гарантируя, что ПО является стабильным и может быть передано в эксплуатацию в любое время, а передача его происходит вручную. Целью является сборка, тестирование и релиз программного обеспечения с большей скоростью и частотой. Подход позволяет уменьшить стоимость, время и риски внесения изменений путём более частных мелких обновлений в продакшн-приложение.

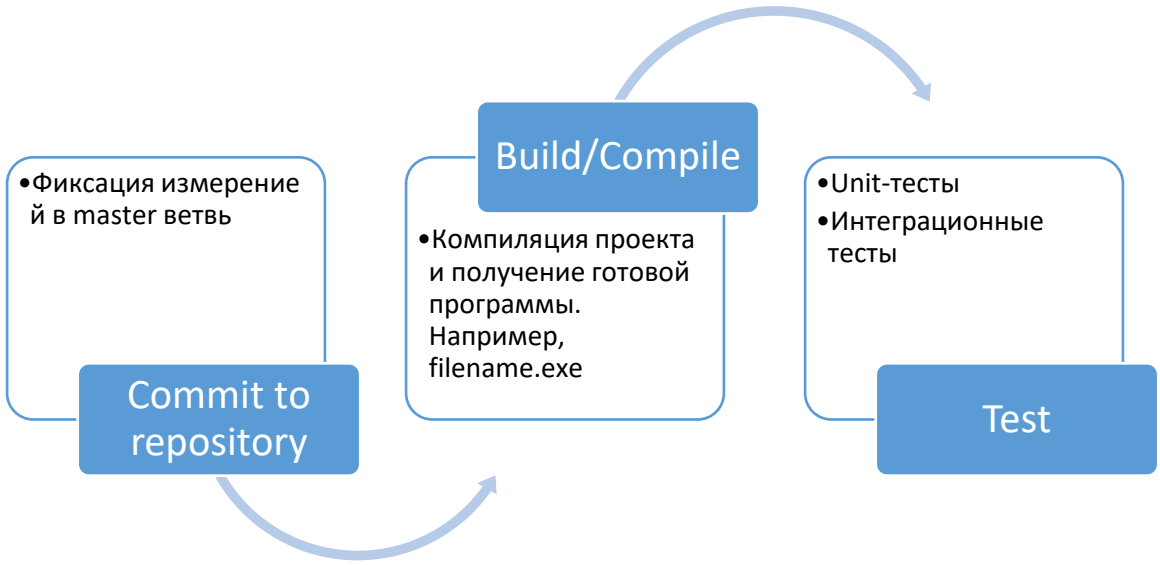

Рисунок 1 - Непрерывная интеграция (CI, англ. Continuous Integration)

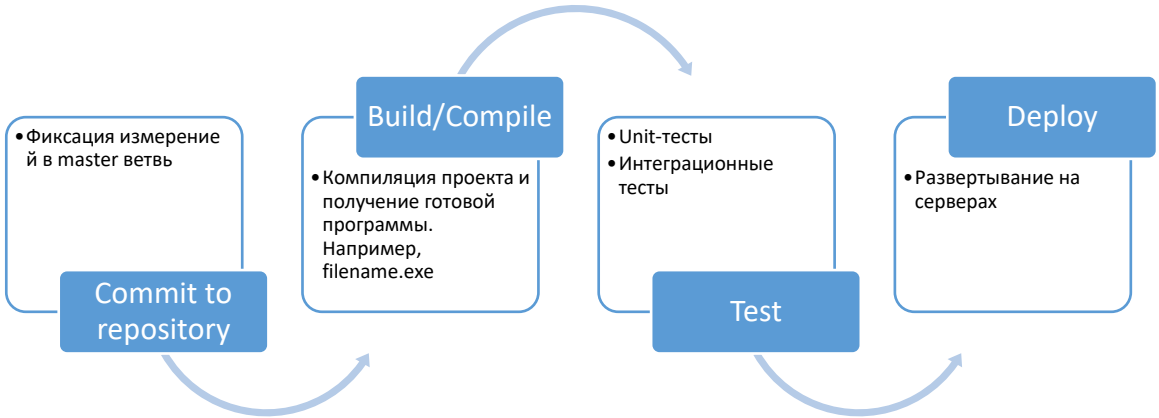

Рисунок 2 - Непрерывная доставка (англ. Continuous delivery или CD, или CDE)

Практическая часть

Первый запуск Jenkins

Необходимо активировать Jenkins. Для этого необходимо ввести пароль из файла «C:\Windows\system32\config\systemprofile\AppData\Local\Jenkins.jenkins\secrets\initialAdminPassword» (Рисунок 1).

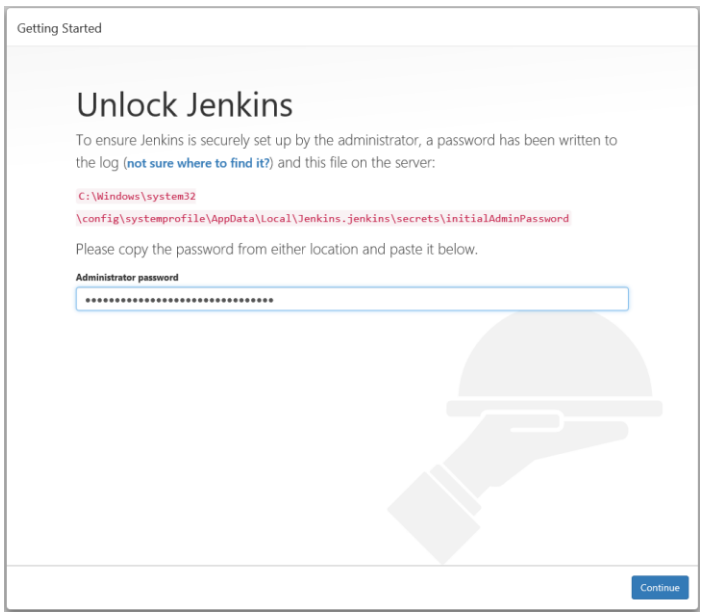

Рисунок 1 – главная страница процесса установки

Плагины расширяют Jenkins дополнительными функциями для поддержки множества различных потребностей (Рисунок 2).

1. Установите предлагаемые плагины.
2. Выберите плагины для установки.

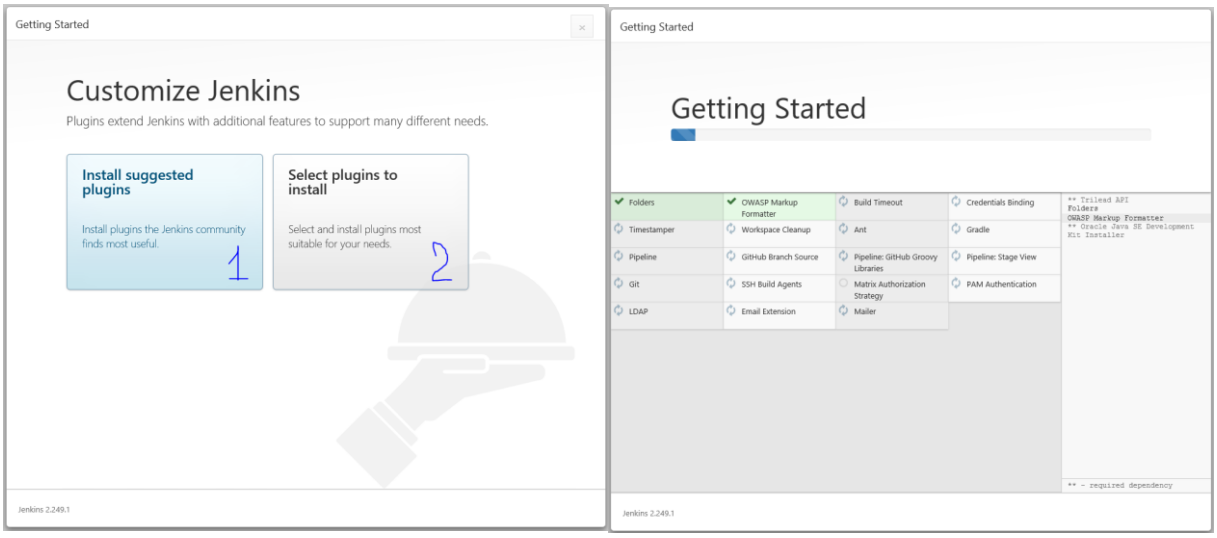

Рисунок 2 – настройка плагинов

Далее производится первоначальная настройка, включающая в себя создание нового пользователя (рисунок 3).

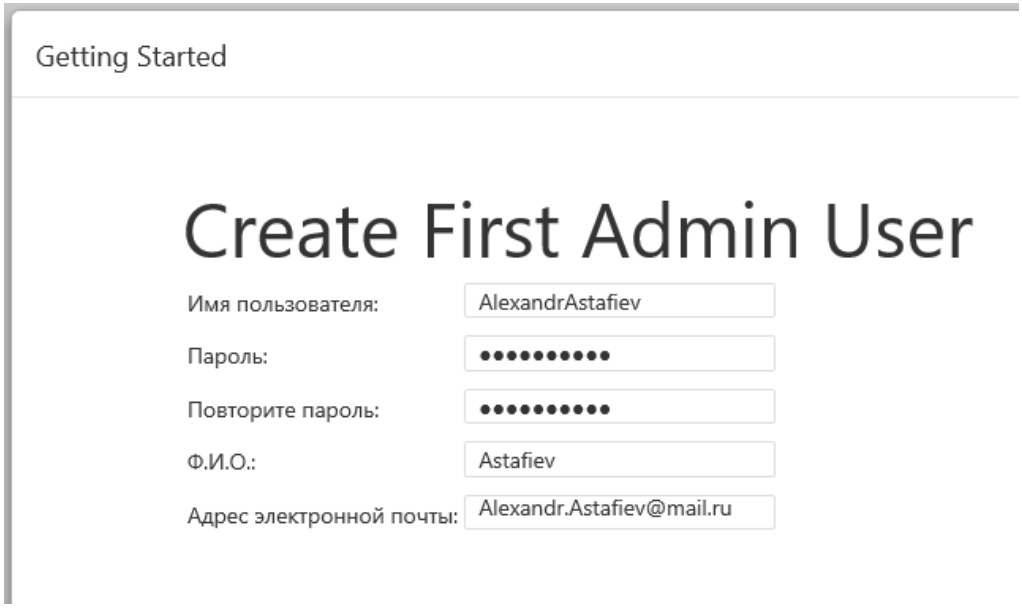

Рисунок 3 – регистрация нового пользователя

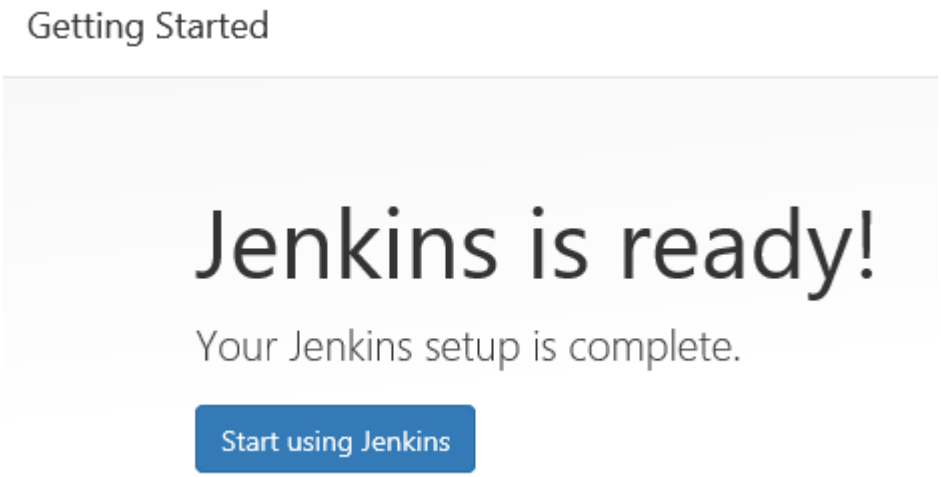

Рисунок 4 – настройка завершена

После завершения настройки (рисунок 4) главное меню выглядит как показано на рисунке 5.

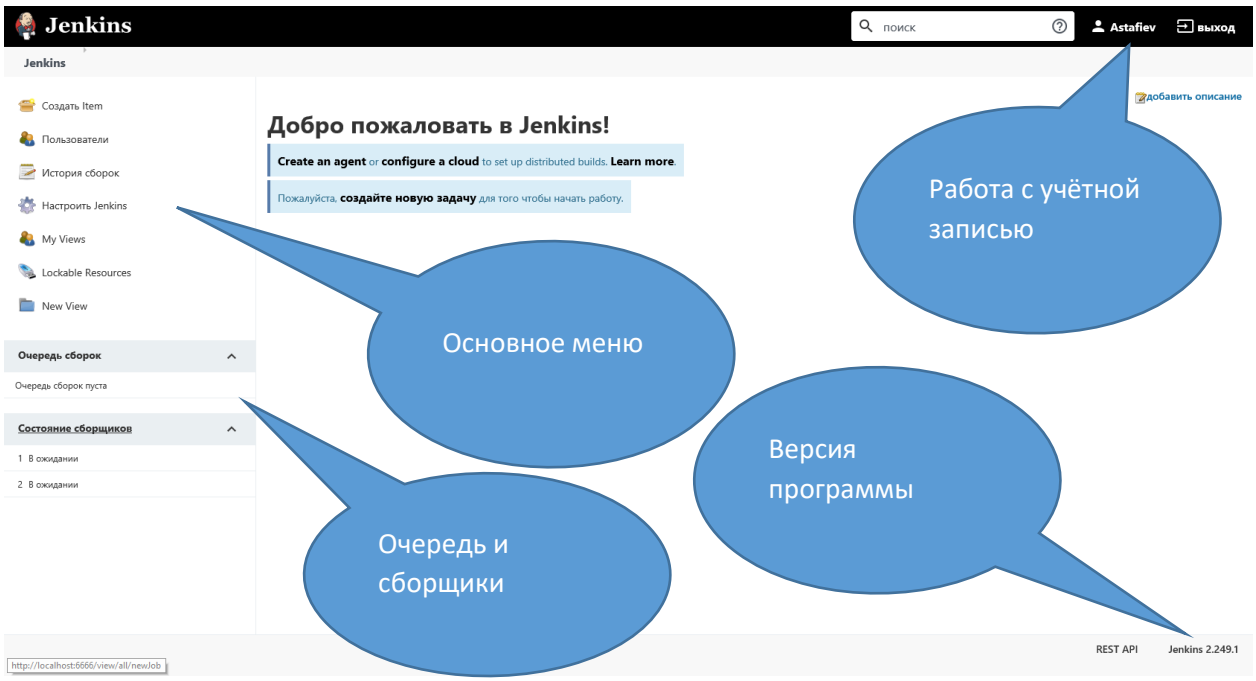

Рисунок 5 – главное меню системы Jenkins

Создание простейшей задачи

Создание задачи

Для создания задачи необходимо выбрать пункт «Создать Item» как показано на рисунке 6.

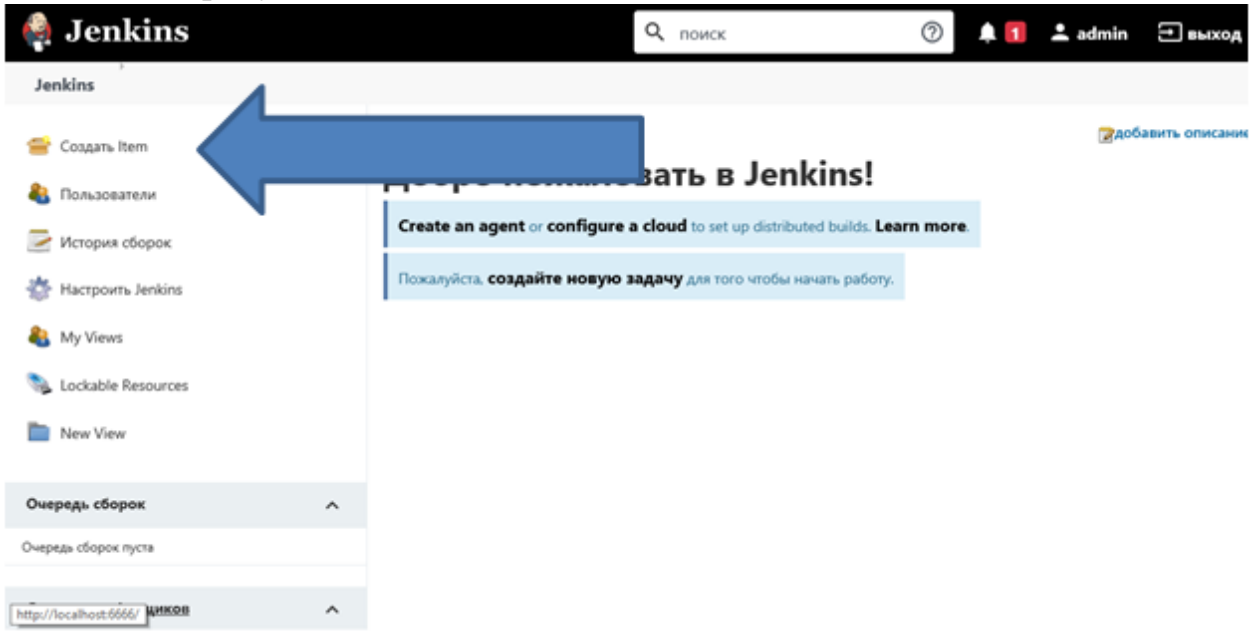

Рисунок 6 – создание новой задачи

Далее необходимо выбрать имя и тип проекта.

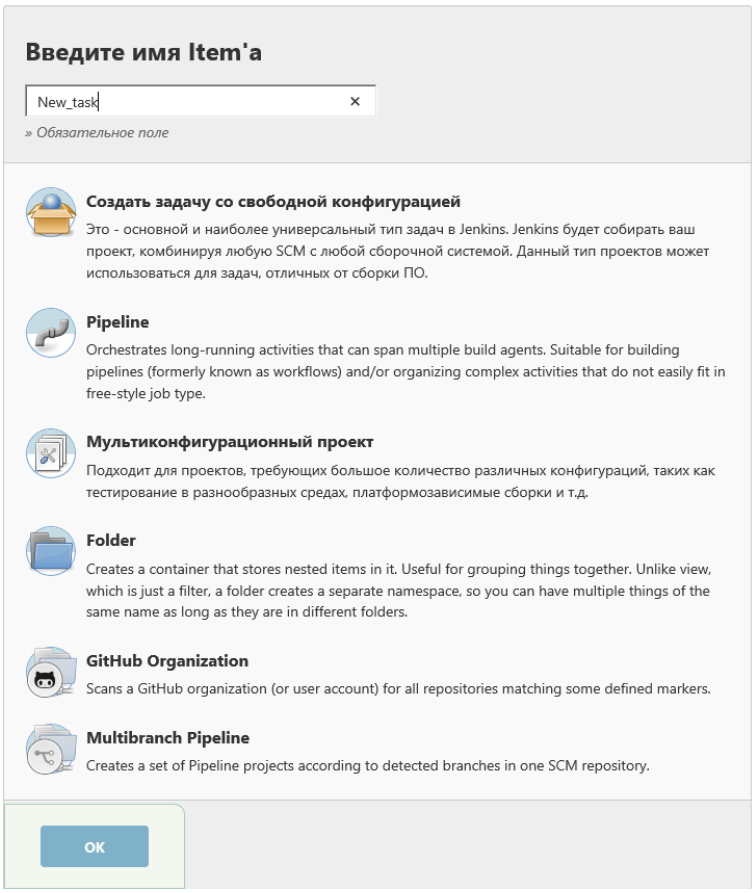

Рисунок 7 – выбор имени и типа проекта

Выберем задачу со свободной конфигурацией. В появившемся окне нас интересуют пункты «Сборка» и «Послесборочные операции». Это инструменты, которые позволяют выполнять необходимые операции в автоматическом режиме. Работа производится путем добавления инструкций.

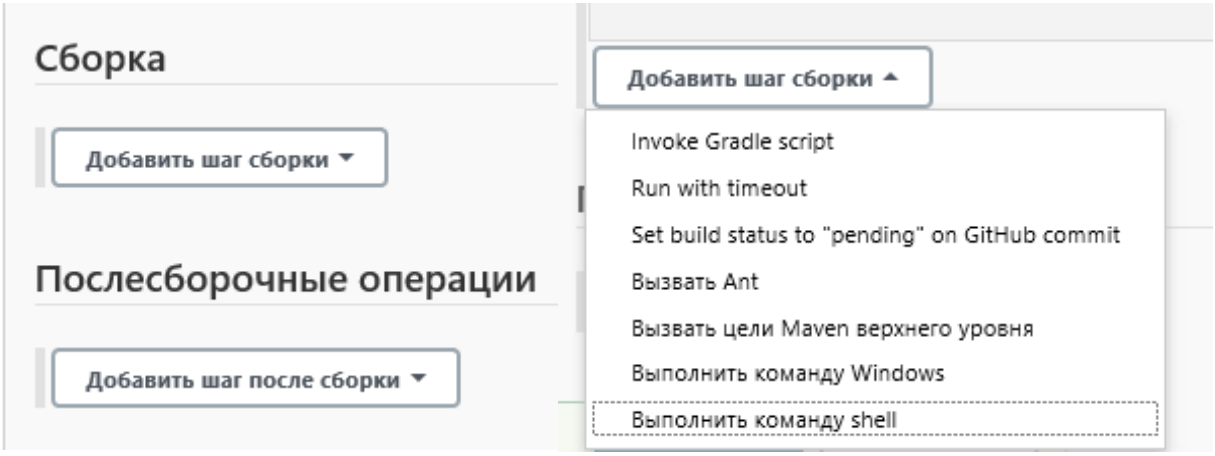

Рисунок 8 – добавление инструкций

Для добавления инструкций необходимо нажать «Добавить шаг сборки». В выпадающем меню будут предложены основные возможные шаги. Самыми «простыми» является «Выполнить команду Windows» и «Выполнить команду shell». Это классические команды командных строк ОС Windows и Linux.

Для примера добавим команду Windows. Результатом этого этапа будет появления окна ввода команд, как показано на рисунке 9.

Рисунок 9 – добавление команды Windows

Выполним простую команду по выводу сообщения «Hello world!». Для этого добавим команду echo Hello world!.

Рисунок 10 – Описание инструкции

После внесения и сохранения изменений попробуем выполнить нашу задачу. Для этого необходимо выбрать пункт «Собрать сейчас».

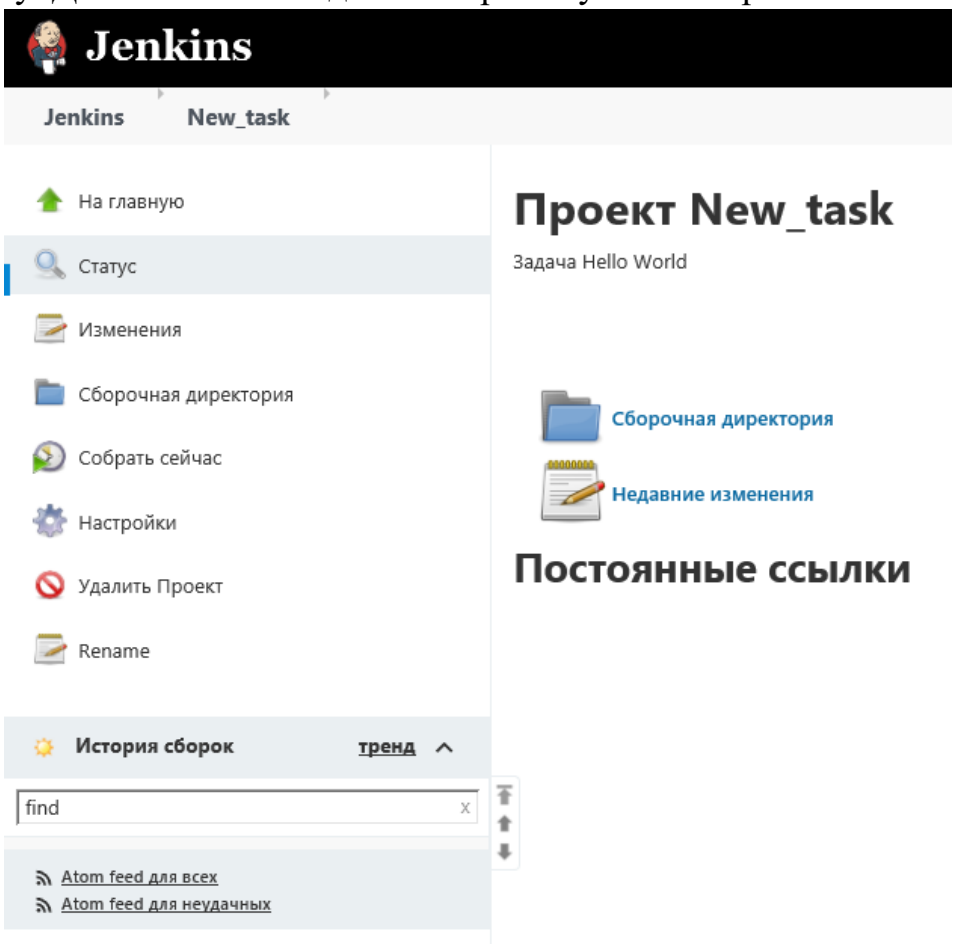

Рисунок 11 – сборка проекта

Задача встанет в очередь на выполнение, выполнится и будет отображаться в истории сборок.

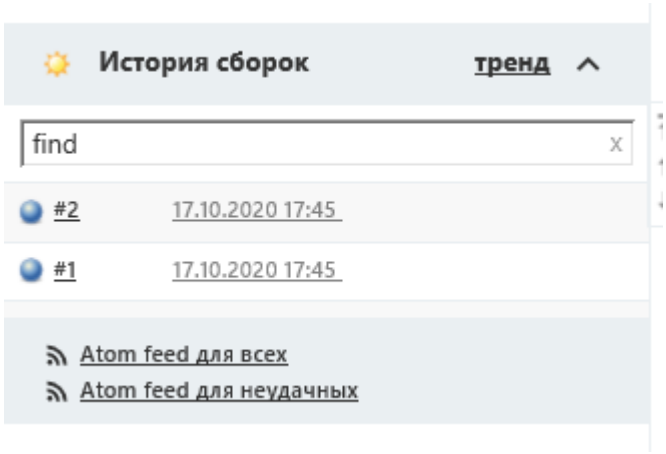

Рисунок 12 – история сборок

Чтобы посмотреть результат выполнения необходимо открыть режим консоли в контекстном меню. Результатом будет являться текст, показанный на рисунке 13.

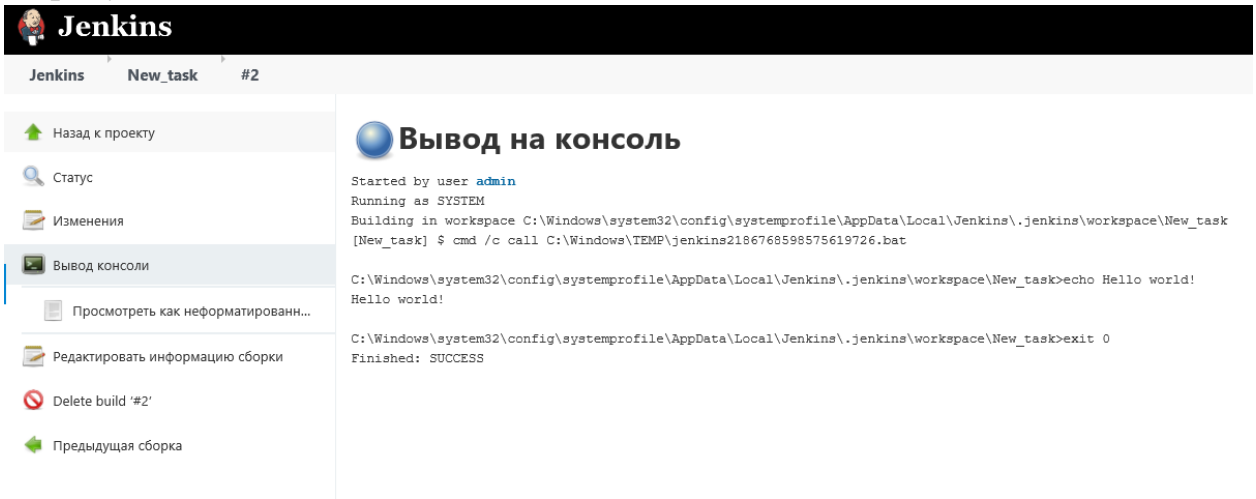

Рисунок 13 – результат работы задачи

В результате получено сообщение:

```
Started by user admin
Running as SYSTEM
Building in workspace
C:\Windows\system32\config\systemprofile\AppData\Local\Jenkins\.jen
kins\workspace\New_task
[New_task] $ cmd /c call
C:\Windows\TEMP\jenkins2186768598575619726.bat

C:\Windows\system32\config\systemprofile\AppData\Local\Jenkins\.jen
kins\workspace\New_task>echo Hello world!
Hello world!

C:\Windows\system32\config\systemprofile\AppData\Local\Jenkins\.jen
kins\workspace\New_task>exit 0
Finished: SUCCESS
```

Основными частями является та часть, где выведено сообщение Hello world! и Finished: SUCCESS, что означает успешное выполнение инструкции.

В инструкциях Jenkins имеются внутренние команды. Узнать о них можно перейдя по ссылке «See the list of available environment variables» в

описаниях шагов сборки. Полный список доступных переменных окружения приведен в приложении 1 на языке оригинала и в приложении 2 на русском языке (Яндекс Передодчик). Для использования переменных окружения в ОС Linux (команды shell) необходимо использовать

$ИМЯ_ПЕРЕМЕННОЙ_ОКРУЖЕНИЯ

В ОС Windows используется конструкция:

%ИМЯ_ПЕРЕМЕННОЙ_ОКРУЖЕНИЯ%

Для проверки добавим в имеющийся шаг сборки информацию о самой инструкции:

```
echo "BUILD_NUMBER" :: %BUILD_NUMBER%
echo "BUILD_ID" :: %BUILD_ID%
echo "BUILD_DISPLAY_NAME" :: %BUILD_DISPLAY_NAME%
echo "JOB_NAME" :: %JOB_NAME%
echo "JOB_BASE_NAME" :: %JOB_BASE_NAME%
echo "BUILD_TAG" :: %BUILD_TAG%
echo "EXECUTOR_NUMBER" :: %EXECUTOR_NUMBER%
echo "NODE_NAME" :: %NODE_NAME%
echo "NODE_LABELS" :: %NODE_LABELS%
echo "WORKSPACE" :: %WORKSPACE%
echo "JENKINS_HOME" :: %JENKINS_HOME%
echo "JENKINS_URL" :: %JENKINS_URL%
echo "BUILD_URL" ::%BUILD_URL%
echo "JOB_URL" :: %JOB_URL%
```

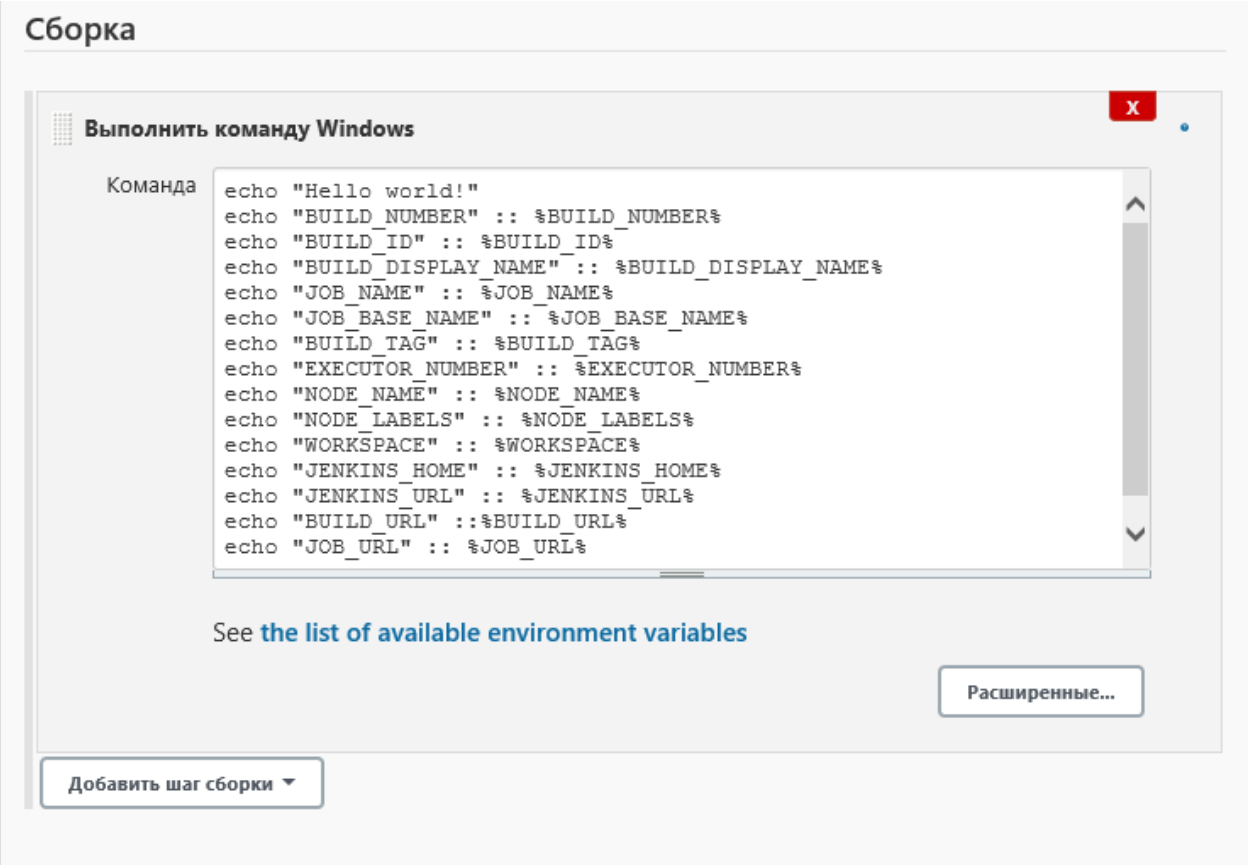

Рисунок 14 – использование переменных окружения

После выполнения результат будет следующим:


```
Started by user admin
Running as SYSTEM
Building in workspace C:\Windows\system32
\config\systemprofile\AppData\Local\Jenkins\.jenkins\workspace\New_task
[New_task] $ cmd /c call C:\Windows\TEMP\jenkins2937211691255023384.bat

C:\Windows\system32
\config\systemprofile\AppData\Local\Jenkins\.jenkins\workspace\New_task>echo "Hello
world!"
"Hello world!"

C:\Windows\system32
\config\systemprofile\AppData\Local\Jenkins\.jenkins\workspace\New_task>echo
"BUILD_NUMBER" :: 9
"BUILD_NUMBER" :: 9

C:\Windows\system32
\config\systemprofile\AppData\Local\Jenkins\.jenkins\workspace\New_task>echo
"BUILD_ID" :: 9
"BUILD_ID" :: 9

C:\Windows\system32
\config\systemprofile\AppData\Local\Jenkins\.jenkins\workspace\New_task>echo
"BUILD_DISPLAY_NAME" :: #9
"BUILD_DISPLAY_NAME" :: #9

C:\Windows\system32
\config\systemprofile\AppData\Local\Jenkins\.jenkins\workspace\New_task>echo
"JOB_NAME" :: New_task
"JOB_NAME" :: New_task

C:\Windows\system32
\config\systemprofile\AppData\Local\Jenkins\.jenkins\workspace\New_task>echo
"JOB_BASE_NAME" :: New_task
"JOB_BASE_NAME" :: New_task

C:\Windows\system32
\config\systemprofile\AppData\Local\Jenkins\.jenkins\workspace\New_task>echo
"BUILD_TAG" :: jenkins-New_task-9
"BUILD_TAG" :: jenkins-New_task-9
```

Рисунок 15 – результат выполнения шага сборки с использованием окружающих переменных

Создание задачи по развертыванию проекта на OpenServer

При установке OpenServer самым первым проектом создается сайт по адресу **http://localhost** как показано на рисунке 16.

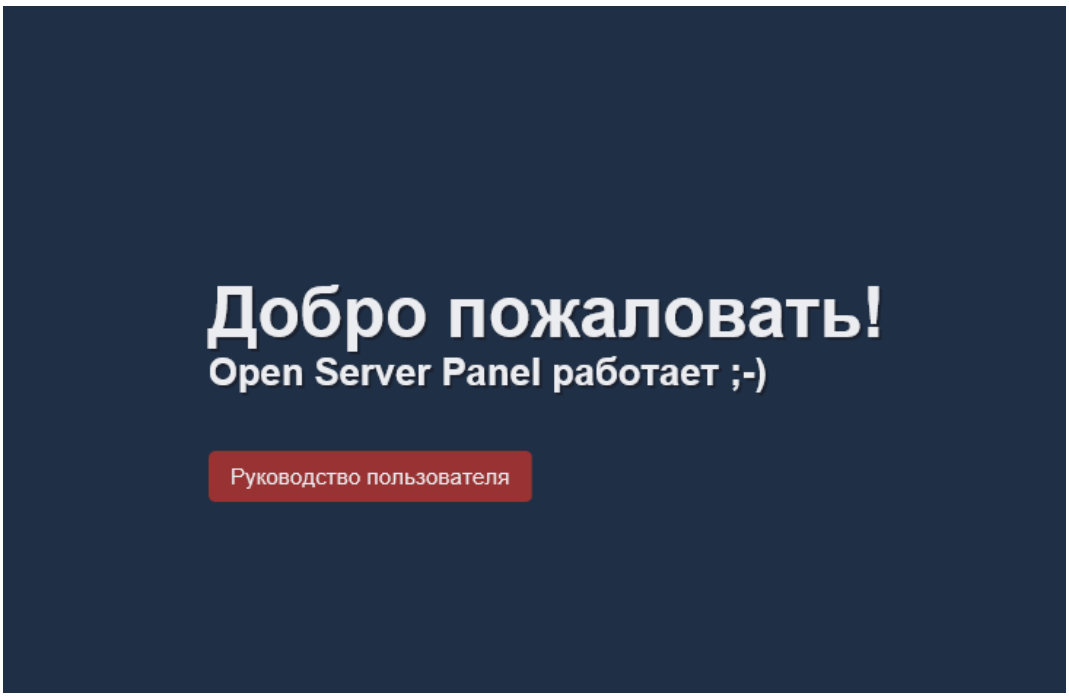

Рисунок 16 – первый проект OpenServer

Допустим, что программист работает в рабочей директории D:/Code. Задача заключается в автоматическом развертывании файла index.html на рабочем сервере. Формально задача заключается в копировании файла D:/Code/index.html на сервере по адресу D:/OpenServer/Domains/localhost/. Для этого создадим новую задачу, которая будет копировать файл D:/Code/index.html, заменяя D:/OpenServer/Domains/localhost/index.html.

Создадим страницу с содержимым:

```
<html>
<head>
<title>Сайт для развертывания проекта через Jenkins</title>
</head>
<body>
<center><p><h1>Здравствуйте! Это привет от Астафьева Александра. Развертываем проект через Jenkins.</h1></p></center>
</body>
</html>
```

Результатом этого будет страница следующего вида:

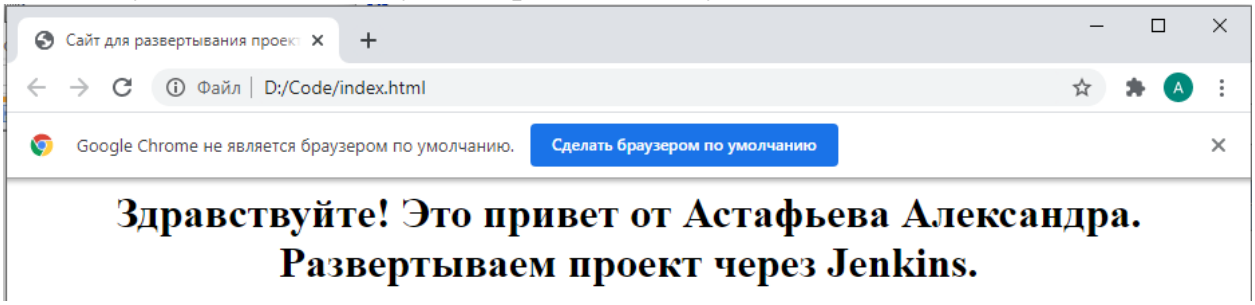

Рисунок 17 – пример созданной страницы

Теперь создадим инструкцию в Jenkins для развертывания проекта, состоящего из одного файла index.html. Для этого необходимо:

1. Создадим новую задачу.
2. В сборке добавим новую инструкцию «Выполнить команду Windows».
3. Запишем в нее код: copy D:\\Code\index.html D:\\OpenServer\Domains\localhost\index.html.
4. Запустим исполнение задачи.
5. После исполнения задачи, если зайти на сайт **http://localhost** можно увидеть, что код из рабочей директории был скопирован на рабочий сервер.

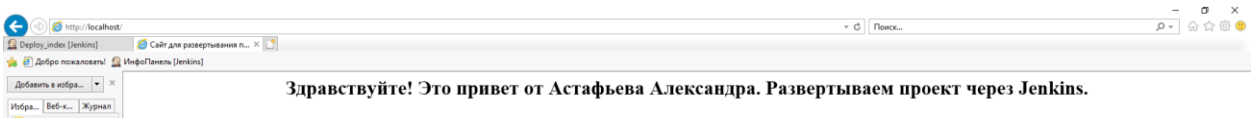

Рисунок 18 – результат развертывания кода.

Практическая часть

1. Ознакомиться с теоретическим материалом.
2. Выполнить установки Jenkins и OpenServer.
3. Повторить примеры из методических указаний.
4. Разработать собственный сайт в соответствии с вариантом из таблицы 1 в рабочей директории и создать задачу автоматического развертывания на локальном сервере. В состав сайта должно входить не менее 3 страниц.
5. Создать расписание запуска сборки. Например, 1 раз в 3 минуты.
6. Реализовать механизм запуска сборки по токену из командной строки с использованием автоматической авторизации. Формат: **http://username:token@jenkins_address**.....
7. Добавить в сборку информацию с использованием переменных окружения.
8. По мере возможности, реализовать задачу сборки при наличии изменений в репозитории Git. Например, вывести список файлов и результаты консоли сборки.
9. Оформить отчёт по проделанной работе.

Таблица 1 – варианты заданий

№ Варианта	Тема
1	Сайт отеля
2	Сайт ресторана
3	Сайт администрации
4	Новостной сайт

5	Сайт института
6	Сайт школы
7	Сайт обувного магазина
8	Сайт автомойки
9	Сайт цветочного магазина
10	Сайт охранного агентства
11	Сайт турфирмы
12	Сайт города
13	Сайт налоговой службы
14	Сайт полиции
15	Сайт скорой помощи
16	Сайт автовокзала
17	Сайт знакомств
18	Сайт интернет провайдера
19	Сайт автосервиса
20	Сайт объявлений

Приложение 1. Список доступных переменных окружения

The following variables are available to shell scripts

BRANCH_NAME

For a multibranch project, this will be set to the name of the branch being built, for example in case you wish to deploy to production from master but not from feature branches; if corresponding to some kind of change request, the name is generally arbitrary (refer to CHANGE_ID and CHANGE_TARGET).

CHANGE_ID

For a multibranch project corresponding to some kind of change request, this will be set to the change ID, such as a pull request number, if supported; else unset.

CHANGE_URL

For a multibranch project corresponding to some kind of change request, this will be set to the change URL, if supported; else unset.

CHANGE_TITLE

For a multibranch project corresponding to some kind of change request, this will be set to the title of the change, if supported; else unset.

CHANGE_AUTHOR

For a multibranch project corresponding to some kind of change request, this will be set to the username of the author of the proposed change, if supported; else unset.

CHANGE_AUTHOR_DISPLAY_NAME

For a multibranch project corresponding to some kind of change request, this will be set to the human name of the author, if supported; else unset.

CHANGE_AUTHOR_EMAIL

For a multibranch project corresponding to some kind of change request, this will be set to the email address of the author, if supported; else unset.

CHANGE_TARGET

For a multibranch project corresponding to some kind of change request, this will be set to the target or base branch to which the change could be merged, if supported; else unset.

CHANGE_BRANCH

For a multibranch project corresponding to some kind of change request, this will be set to the name of the actual head on the source control system which may or may not be different from BRANCH_NAME. For example in GitHub or Bitbucket this would have the name of the origin branch whereas BRANCH_NAME would be something like PR-24.

CHANGE_FORK

For a multibranch project corresponding to some kind of change request, this will be set to the name of the forked repo if the change originates from one; else unset.

TAG_NAME

For a multibranch project corresponding to some kind of tag, this will be set to the name of the tag being built, if supported; else unset.

TAG_TIMESTAMP

For a multibranch project corresponding to some kind of tag, this will be set to a timestamp of the tag in milliseconds since Unix epoch, if supported; else unset.

TAG_UNIXTIME

For a multibranch project corresponding to some kind of tag, this will be set to a timestamp of the tag in seconds since Unix epoch, if supported; else unset.

TAG_DATE

For a multibranch project corresponding to some kind of tag, this will be set to a timestamp in the format as defined by **java.util.Date#toString()** (e.g., Wed Jan 1 00:00:00 UTC 2020), if supported; else unset.

BUILD_NUMBER

The current build number, such as "153"

BUILD_ID

The current build ID, identical to BUILD_NUMBER for builds created in 1.597+, but a YYYY-MM-DD_hh-mm-ss timestamp for older builds

BUILD_DISPLAY_NAME

The display name of the current build, which is something like "#153" by default.

JOB_NAME

Name of the project of this build, such as "foo" or "foo/bar".

JOB_BASE_NAME

Short Name of the project of this build stripping off folder paths, such as "foo" for "bar/foo".

BUILD_TAG

String of "jenkins-*${JOB_NAME}-${BUILD_NUMBER}*". All forward slashes ("/") in the JOB_NAME are replaced with dashes ("-"). Convenient to put into a resource file, a jar file, etc for easier identification.

EXECUTOR_NUMBER

The unique number that identifies the current executor (among executors of the same machine) that’s carrying out this build. This is the number you see in the "build executor status", except that the number starts from 0, not 1.

NODE_NAME

Name of the agent if the build is on an agent, or "master" if run on master

NODE_LABELS

Whitespace-separated list of labels that the node is assigned.

WORKSPACE

The absolute path of the directory assigned to the build as a workspace.

WORKSPACE_TMP

A temporary directory near the workspace that will not be browsable and will not interfere with SCM checkouts. May not initially exist, so be sure to create the directory as needed (e.g., mkdir -p on Linux). Not defined when the regular workspace is a drive root.

JENKINS_HOME

The absolute path of the directory assigned on the master node for Jenkins to store data.

JENKINS_URL

Full URL of Jenkins, like http://server:port/jenkins/ (note: only available if *Jenkins URL* set in system configuration)

BUILD_URL

Full URL of this build, like http://server:port/jenkins/job/foo/15/ (*Jenkins URL* must be set)

JOB_URL

Full URL of this job, like http://server:port/jenkins/job/foo/ (*Jenkins URL* must be set)

GIT_COMMIT

The commit hash being checked out.

GIT_PREVIOUS_COMMIT

The hash of the commit last built on this branch, if any.

GIT_PREVIOUS_SUCCESSFUL_COMMIT

The hash of the commit last successfully built on this branch, if any.

GIT_BRANCH

The remote branch name, if any.

GIT_LOCAL_BRANCH

The local branch name being checked out, if applicable.

GIT_CHECKOUT_DIR

The directory that the repository will be checked out to. This contains the value set in Checkout to a sub-directory, if used.

GIT_URL

The remote URL. If there are multiple, will be GIT_URL_1, GIT_URL_2, etc.

GIT_COMMITTER_NAME

The configured Git committer name, if any, that will be used for FUTURE commits from the current workspace. It is read from the **Global Config user.name Value** field of the Jenkins **Configure System** page.

GIT_AUTHOR_NAME

The configured Git author name, if any, that will be used for FUTURE commits from the current workspace. It is read from the **Global Config user.name Value** field of the Jenkins **Configure System** page.

GIT_COMMITTER_EMAIL

The configured Git committer email, if any, that will be used for FUTURE commits from the current workspace. It is read from the **Global Config user.email Value** field of the Jenkins **Configure System** page.

GIT_AUTHOR_EMAIL

The configured Git author email, if any, that will be used for FUTURE commits from the current workspace. It is read from the **Global Config user.email Value** field of the Jenkins **Configure System** page.

Приложение 2. Список доступных переменных окружения на русском языке.

Для сценариев оболочки доступны следующие переменные

BRANCH_NAME

Для проекта с несколькими ветвями это будет установлено в качестве имени создаваемой ветви, например, если вы хотите развернуть ее в производство из главной ветви, но не из ветвей объектов; если это соответствует какому-то запросу на изменение, то имя обычно является произвольным (см. CHANGE_ID и CHANGE_TARGET).

CHANGE_ID

Для многоотраслевого проекта, соответствующего какому-либо запросу на изменение, этот параметр будет установлен в идентификатор изменения, например номер запроса на вытягивание, если он поддерживается; в противном случае он не будет установлен.

CHANGE_URL

Для многоотраслевого проекта, соответствующего какому-либо запросу на изменение, этот параметр будет установлен в URL-адрес изменения, если он поддерживается; в противном случае он не будет установлен.

CHANGE_TITLE

Для многоотраслевого проекта, соответствующего какому-либо запросу на изменение, этот параметр будет установлен в заголовок изменения, если он поддерживается; в противном случае он не будет установлен.

CHANGE_AUTHOR

Для многоотраслевого проекта, соответствующего какому-либо запросу на изменение, это будет установлено на имя пользователя автора предлагаемого изменения, если оно поддерживается; в противном случае оно не будет установлено.

CHANGE_AUTHOR_DISPLAY_NAME

Для многоотраслевого проекта, соответствующего какому-либо запросу на изменение, это будет установлено на человеческое имя автора, если оно поддерживается; в противном случае оно не будет установлено.

CHANGE_AUTHOR_EMAIL

Для многоотраслевого проекта, соответствующего какому-либо запросу на изменение, этот параметр будет установлен на адрес электронной почты автора, если он поддерживается; в противном случае он не будет установлен.

CHANGE_TARGET

Для многоотраслевого проекта, соответствующего какому-либо запросу на изменение, этот параметр будет установлен в целевую или базовую ветвь, с которой изменение может быть объединено, если оно поддерживается; в противном случае он не будет установлен.

CHANGE_BRANCH

Для многоотраслевого проекта, соответствующего какому-то запросу на изменение, это будет установлено на имя фактического руководителя в системе управления версиями, которое может отличаться или не отличаться от BRANCH_NAME. Например, в GitHub или Bitbucket это будет иметь имя исходной ветви, тогда как BRANCH_NAME будет чем-то вроде PR-24.

CHANGE_FORK

Для многоотраслевого проекта, соответствующего какому-либо запросу на изменение, это будет установлено в качестве имени разветвленного РЕПО, если изменение происходит от одного из них; иначе не установлено.

TAG_NAME

Для многоотраслевого проекта, соответствующего какому-либо тегу, это будет установлено в имя создаваемого тега, если оно поддерживается; в противном случае оно не будет установлено.

TAG_TIMESTAMP

Для проекта разветвленную соответствует какой-то тег, это будет установлено для отметки метка в миллисекундах с начала эпохи Unix, если поддерживается; еще неопределенные.

TAG_UNIXTIME

Для мультиответвленного проекта, соответствующего какому-либо тегу, это будет установлено в метку времени тега в секундах с эпохи Unix, если она поддерживается; в противном случае она не будет установлена.

TAG_DATE

Для мультиответвленного проекта, соответствующего какому-то тегу, он будет установлен в метку времени в формате, определенном java.util.Date#toString() (например, Wed Jan 1 00:00:00 UTC 2020), если поддерживается; else unset.

BUILD_NUMBER

Текущий номер сборки, например "153"

BUILD_ID

Текущий идентификатор сборки, идентичный номеру сборки для сборок, созданных в версии 1.597+, но временная метка гггг-мм-DD_hh-мм-ss для более старых сборок

BUILD_DISPLAY_NAME

Отображаемое имя текущей сборки, которое по умолчанию является чем-то вроде "#153".

ИМЯ_ЗАДАНИЯ

Название проекта этой сборки, например "foo" или "foo/bar".

JOB_BASE_NAME

Краткое название проекта этой сборки, удаляющее пути к папкам, например "foo" для "bar/foo".

BUILD_TAG

Строка "jenkins-${JOB_NAME}-${BUILD_NUMBER}". Все косые черты ("/") в имени задания заменяются тире ("-"). Удобно помещать в файл ресурсов, файл jar и т. д. Для более легкой идентификации.

EXECUTOR_NUMBER

Уникальный номер, идентифицирующий текущего исполнителя (среди исполнителей одной и той же машины), выполняющего эту сборку. Это число вы видите в разделе "статус исполнителя сборки", за исключением того, что оно начинается с 0, а не с 1.

ИМЯ_УЗЛА

Имя агента, если сборка выполняется на агенте, или "master", если выполняется на master

NODE_LABELS

Разделенный пробелами список меток, назначенных узлу.

Рабочее пространство

Абсолютный путь к каталогу, назначенному сборке в качестве рабочей области.

WORKSPACE_TMP

Временный каталог рядом с рабочей областью, который не будет доступен для просмотра и не будет мешать проверкам SCM. Возможно, изначально он не существует, поэтому обязательно создайте каталог по мере необходимости (например, mkdir-p в Linux). Не определено, если обычная рабочая область является корневым диском.

JENKINS_HOME

Абсолютный путь к каталогу, назначенному на главном узле для хранения данных Дженкинсом.

JENKINS_URL

Полный URL-адрес Дженкинс, как http://server:port/jenkins/ (Примечание: доступно только если Дженкинс набору URL-адресов в настройки системы)

BUILD_URL

Полный URL-адрес этой сборки, например http://server:port/jenkins/job/foo/15/ (должен быть установлен URL-адрес Дженкинса)

JOB_URL

Полный URL этой работы, например http://server:port/jenkins/job/foo/ (должен быть установлен URL-адрес Дженкинса)

GIT_COMMIT

Хэш фиксации проверяется.

GIT_PREVIOUS_COMMIT

Хэш последнего коммита, построенного на этой ветке, если таковой имеется.

GIT_PREVIOUS_SUCCESSFUL_COMMIT

Хэш последнего успешно построенного коммита на этой ветке, если таковой имеется.

GIT_BRANCH

Имя удаленного филиала, если таковое имеется.

GIT_LOCAL_BRANCH

Проверяемое имя местного филиала, если это применимо.

GIT_CHECKOUT_DIR

Каталог, в который будет извлечен репозиторий. Он содержит значение, установленное в Checkout для подкаталога, если оно используется.

GIT_URL

Удаленный URL-адрес. Если их несколько, то будут GIT_URL_1, GIT_URL_2 и т. д.

GIT_COMMITTER_NAME

Настроенное имя Коммиттера Git, если таковое имеется, которое будет использоваться для будущих коммитов из текущей рабочей области. Он считывается из глобальной конфигурации user.name поле Значение страницы Jenkins Configure System.

GIT_AUTHOR_NAME

Настроенное имя автора Git, если таковое имеется, которое будет использоваться для будущих коммитов из текущей рабочей области. Он считывается из глобальной конфигурации user.name поле Значение страницы Jenkins Configure System.

GIT_COMMITTER_EMAIL

Настроенного мерзавца электронной почты исполнителя, если таковые имеются, которые будут использоваться для будущих совершает из текущей рабочей области. Это чтение из глобальной конфигурации пользователя.поле "электронная почта" стоимость Дженкинс странице настройки системы.

GIT_AUTHOR_EMAIL

Настроенный адрес электронной почты автора Git, если таковой имеется, который будет использоваться для будущих коммитов из текущей рабочей области. Это чтение из глобальной конфигурации пользователя.поле "электронная почта" стоимость Дженкинс странице настройки системы.