# Модуль 4. Непрерывная интеграция и развертывание (CI/CD).

## В этом модуле

Целью проекта разработки программного обеспечения является вывод результатов работы команды проекта в промышленное использование (production, продакшн, «прод»). Этот процесс состоит из множества отдельных рутинных повторяющихся операций, связанных с тестированием, сборкой, развертыванием и организацией мониторинга системы в эксплуатации. С целью повышения эффективности решения этой задачи в DevOps применяется автоматизация в тестировании и сборке решения (непрерывная интеграция, Continuous Integration, CI) и развертывании (непрерывная доставка, Continuous Delivery, CD). Все эти инструменты оказались очень востребованы и в MLOps, поскольку в проектах машинного обучения правильное развертывание и получение качественной обратной связи от эксплуатации имеют гораздо большую важность, чем для обычного программного обеспечения, поскольку напрямую очень сильно влияют на качество работы модели машинного обучения, а также на необходимость ее переобучения при изменении условий.

Существует множество типовых инструментов для решения задач CI/CD, например:

* Jenkins является наиболее популярным инструментом для автоматизации контроля и выполнения задач. Он позволяет собрать отдельные операции в конвейер и управлять выполнением этих операций по заранее подготовленным правилам: по расписанию, по получению сигнала (триггера) от внешнего источника или вручную. Например, Jenkins позволяет увидеть обновление в git репозитории, сделанное разработчиком, запустить автоматизированное тестирование обновленного ПО, в случае успеха собрать программное обеспечение из всех модулей проекта и запустить его на сервере эксплуатации.
* Настройка нужной конфигурации серверов не представляет сложности для опытного специалиста, однако является рутинной, часто повторяющейся операцией, в которой высока цена даже незначительной ошибки в синтаксисе или параметрах выполняемой команды. Поэтому в этом процессе также давно применяют методы автоматизации. Для автоматизированной подготовки инфраструктуры используют подход «Инфраструктура-как-Код» (Infrastructure-as-a-Code, IaaC), который означает, что все параметры создаваемой инфраструктуры описываются на некотором скриптовом языке, после чего специальное программное обеспечение автоматически создает и настраивает нужную инфраструктуру, следуя инструкциям в этом скрипте. Типовыми инструментами для этой задачи являются terraform и ansible. В практическом задании в этом модуле мы будем использовать ansible.
* Для автоматизированного тестирования используется множество различных утилит. Библиотека с открытым кодом pytest очень популярна для тестирования python скриптов.

Перечисленные инструменты вы используете в этом модуле для решения практической задачи. В реализуемом сценарии разработчик будет публиковать код в git репозиторий, который опрашивается Jenkins с определенной периодичностью. Если Jenkins обнаружит обновления в опубликованном коде, то он автоматически запускает сборку пакета на сервере stage и проводит тестирование с помощью pytest. При успешном завершении тестирования Jenkins запускает настройку сервера эксплуатации с помощью ansible и устанавливает на него обновленное протестированное программное обеспечение.

## Содержание юнитов

Последовательность тем, изучаемых в отдельных юнитах модуля:

|  |
| --- |
| 1. Общая постановка задачи, создание инфраструктуры и окружения |
| 1. Установка и настройка Jenkins. |
| 1. Установка и настройка ansible. |
| 1. Установка и настройка pytest. |
| 1. Подробная постановка задачи. |
| 1. Настройка конвейера операций в Jenkins, автоматизация взаимодействия с git. |
| 1. Настройка сервера тестирования с ansible и автоматизированное тестирование с pytest. |
| 1. Настройка сервера эксплуатации с ansible и автоматизированное развертывание с Jenkins. |
| 1. Задание для самостоятельной работы. |

# Модуль 4. Юнит 1. Общая постановка задачи, создание инфраструктуры и окружения.

*Введение*: В этом юните делается постановка решаемой в модуле задачи, описывается инфраструктура и инструменты, а также даются рекомендации по действиям, которые необходимо выполнить для решения задачи в следующих юнитах.

*Содержание:*

В этом модуле разбирается практический пример создания инфраструктуры для простого проекта машинного обучения. Несмотря на то, что рассматриваемый проект имеет очень простую структуру и решает учебную, а не производственную, задачу, возникающие при прохождения модуля ситуации часто возникают в реальных производственных проектах.

Обычно команда проекта машинного обучения состоит из разных участников, каждый из которых решает свои задачи и использует свои инструменты.

* Инженер и исследователь данных занимаются организацией сбора, преобразования, хранения данных, а также поиском в данных полезных признаков, созданием новых признаков.
* Инженер и исследователь ML создает модель машинного обучения, обучает ее на имеющихся данных, выкладывает программный код в git репозиторий, а обученную модель в формате pickle в репозиторий моделей.
* Тестировщик создает и выполняет различные тесты, чтобы убедится в качестве созданного программного кода, отсутствии негативного влияния на уже существующий код, возможности работы в рамках заданных ограничений на производительность процессора, размер жесткого диска, объем оперативной памяти
* Разработчик создает микросервис, предоставляющий возможность пользоваться моделью ML. Например, с использованием flask создается web сервис с API или простым графическим интерфейсом для того, чтобы с разработанной моделью могли пользователи, не разбирающиеся во внутреннем устройстве микросервиса, просто общаясь с ним командами через API.

Инфраструктура проекта машинного обучения, должна инструменты для всего конвейера проекта от сбора и обработки данных до вывода модели в эксплуатацию и мониторинга. В предыдущих модулях вы уже освоили некоторые инструменты для решения отдельных задач, например, управление данными и трекинг экспериментов с dvc (Модуль 1), управление экспериментом с ClearML (Модуль 2), организация конвейеров операций с airflow и их мониторинг с mlflow (Модуль 3). В этом модуле мы рассмотрим завершающий этап любого проекта разработки программного обеспечения: выполнение тестирования, сборку решения и вывод в эксплуатацию. Для этих целей вы используете: Jenkins для контроля выполнения операций, git для хранения и загрузки кода, ansible для автоматизации настройки элементов инфраструктуры и pytest для автоматического тестирования.

В итоге по результатам работы в модуле необходимо будет создать следующую инфраструктуру:

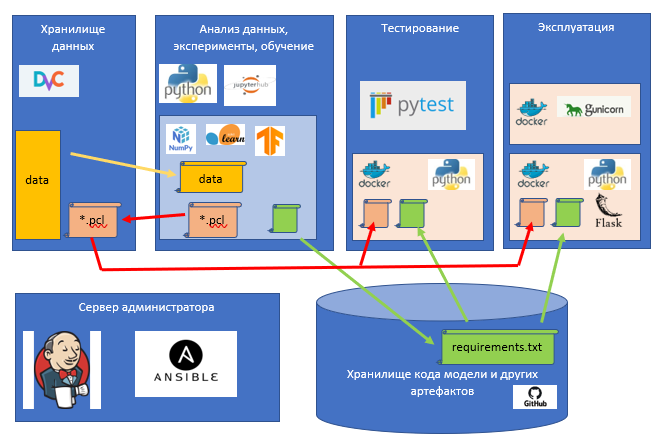


Рисунок «Целевая инфраструктура»

**«Хранилище данных»**

В проекте этого модуля вы будете работать с данными задачи“Titanic Disaster” с площадки соревнований по машинному обучению Kaggle.com. Ранее, в Модуле 1 вы уже настроили dvc хранилище в виде локального сервера с доступом по ssh. Также в Модуле 1 вы научились настраивать доступ к данным из репозитория проекта с помощью утилиты dvc. Также dvc хранилище используется для хранения больших файлов обученных моделей, например, в формате pickle, которую потом можно подгружать в окружение для тестирования и выводить в эксплуатацию.

**«Анализ данных, эксперименты и обучение»**

На этом сервере работают исследователи данных и инженеры ML, поэтому на нем необходим python и необходимые библиотеки для анализа данных и машинного обучения: numpy, pandas, matplotlib, scikit-learn, tensorflow и другие, в зависимости от проекта. Для анализа данных и исследований эффективно использовать инструменты jupyter, для многопользовательской работы используется jupyterhub. Для изоляции проекта и возможности сохранить и точно передать все настройки среды выполнения программы используются виртуальные окружения, например venv, poetry, conda. В результате обучения получается модель ML, которая сохраняется в формате pickle в dvc хранилище. На github сохраняются другие артефакты, полученные на этом этапе: рабочие скрипты, описание параметров виртуального окружения, метаинформацию по датасету. Такой виртуальный сервер вы также уже создали в Модуле 1.

**«Тестирование модели»**

Перед запуском модели в эксплуатацию необходимо провести тщательное тестирование модели, чтобы убедиться в надежности, проверить производительность. Для этого вы будете использовать сервер тестирования, который создадите в этом модуле в юните 7. Тестировать систему необходимо в условиях, максимально приближенных к условиям обучения модели и будущей эксплуатации, поэтому для повторяемости эксперимента при создании тестовой среды используются виртуальные окружения и настройки из файла requirements.txt.

**«Эксплуатация модели»**

Для запуска модели в эксплуатации вам понадобится собрать весь проект целиком, включая backend и frontend проекта. Для удобства проект будет представлять собой микросервис, который запускается в docker. Для простоты в качестве инструмента для бэкенда вы будете использовать flask. Настройки среды внутри контейнера берутся из файла requirements.txt.

**«Сервер администратора»**

Обычно системные администраторы или DevOps/MLOps инженеры используют отдельные сервера для запуска своих рабочих программ или утилит. Это связано к с безопасностью, так и с удобством работы администратора. Также и в нашем примере вы создадите такой отдельный сервер для запуска на нем прикладного программного обеспечения для MLOps; сервера Jenkins, ansible.

С использованием этой инфраструктуры вы с помощью Jenkins настроите регулярный опрос git репозитория проекта и, при наличии изменений, автоматически создадите нужную инфраструктуру для тестирования на сервере тестирования. Затем, если выполненные тесты завершились успешно, Jenkins настроить среду эксплуатации программы и осуществит вывод разработанной и протестированной системы в эксплуатацию. В качестве данных и конвейера обработки и обучения вы используете данные и модель из задачи «Titanic Disaster», которые настраивали в Модуле 1. Системой в эксплуатации будет простой web сервер, который предсказывает выжил ли пассажир по получаемым от пользователя данным.

## Тест

1. Какой инструмент для трекинга обучения модели будет использован в модуле? (0.25)
   1. clearml
   2. mlflow
   3. bash
   4. **dvc**
2. Какие инструменты мы установим на сервер тестирования модели? (0.25)
   1. **pytest**
   2. grafana
   3. flask
   4. jupyter notebook
3. Какое программное обеспечение используется на сервере эксплуатации? (0.25)
   1. openssh-server
   2. **docker**
   3. **flask**
   4. jupyter notebook
4. Сколько виртуальных серверов планируется использовать в проекте? (0.25)
   1. 4
   2. 3
   3. **5**
   4. 6

## Итоги/выводы

В этом юните сформулирована задача модуля, описаны инструменты для ее решения, описано что будем делать в следующих юнитах модуля.

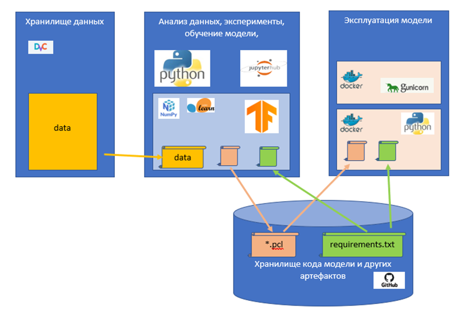
# Модуль 4. Юнит 2. Установка и настройка Jenkins.

## Введение:

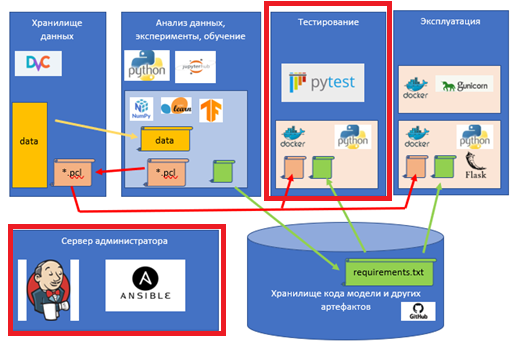
В этом юните вы создадите виртуальный сервер для работы инженера DevOps/MLOps, установите на него Jenkins и осуществите его настройку для дальнейшего использования.

## Содержание:

В следующих юнитах 2-4 вы осуществите дополнительную настройку уже созданной вами в предыдущих модулях инфраструктуры проекта машинного обучения. Например, в Модуле 1 вы создали такую инфраструктуру с использованием виртуальных машин virtualbox

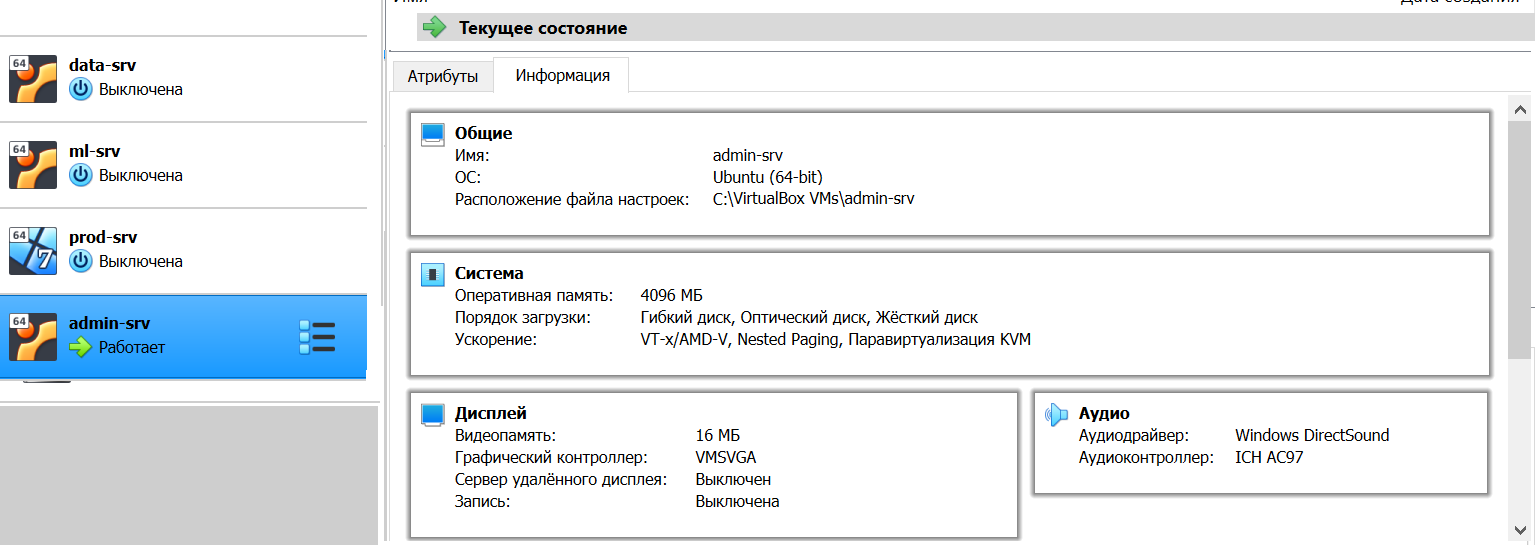


Для запланированной нами целевой инфраструктуры не хватает «Сервера тестирования» и «Сервера администратора», чтобы получилась такая схема



В этом юните вы создадите виртуальный сервер «Сервер администратора» с именем admin-srv и установите на него Jenkins.

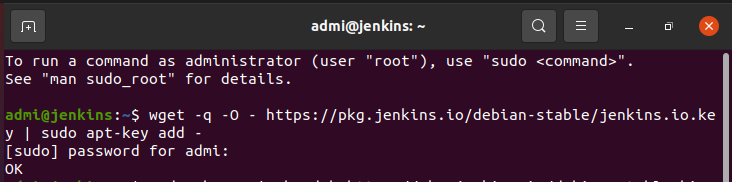
Создание и настройка виртуального сервера admin-srv в virtualbox осуществляется точно так же, как это было сделано в Модуле 1 для создания и настройки других виртуальных машин. Рекомендуемые параметры для настройки:



После установки и настройки операционной системы вы можете залогироваться на новом виртуальном сервере admin-srv и выполнить установку и настройку Jenkins. Описание установки Jenkins можно найти на официальном сайте, выбрав подходящий вариант для используемой вами операционной системы: https://www.jenkins.io/doc/book/installing.

Далее осуществляется установка для используемой нами операционной системы Ubuntu 20.4 Сначала надо подключить ключи для установки пакета Jenkins и установить адрес в Интернет, с которого будет загружаться пакет. Это делается командой

**wget -q -O - https://pkg.jenkins.io/debian-stable/jenkins.io.key | sudo apt-key add -**

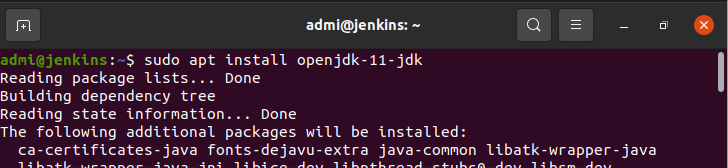


**sudo sh -c 'echo deb https://pkg.jenkins.io/debian-stable binary/ > \ /etc/apt/sources.list.d/jenkins.list'**



Далее для запуска Jenkins нам потребуется установить OpenJDK, это делается командой

**sudo apt install openjdk-11-jdk**



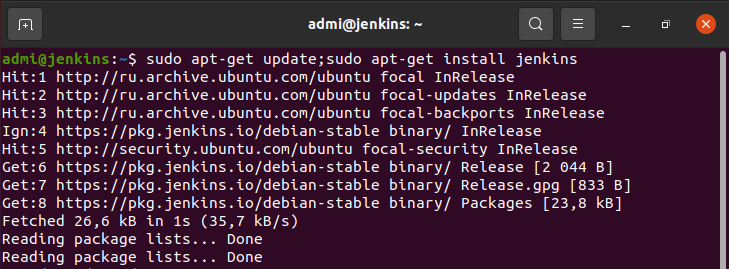
Как правило ssh и git клиенты не установлены по умолчанию, поэтому понадобится их установить командой

**sudo apt-get install ssh git**

После предварительной установки требуемых всех компонентов вы можете установить пакет Jenkins командами

**sudo apt-get update**

**sudo apt-get install jenkins**



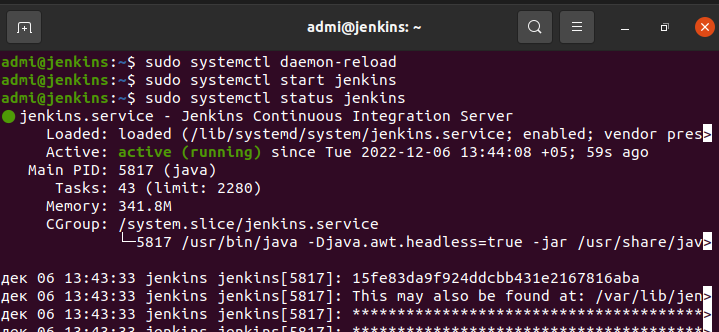
Проверьте, что все установка завершилась корректно

Теперь можно добавить Jenkins в автозагрузку и запустить сервис, после чего проверить, что сервис Jenkins запустился без ошибок. Это можно сделать командами

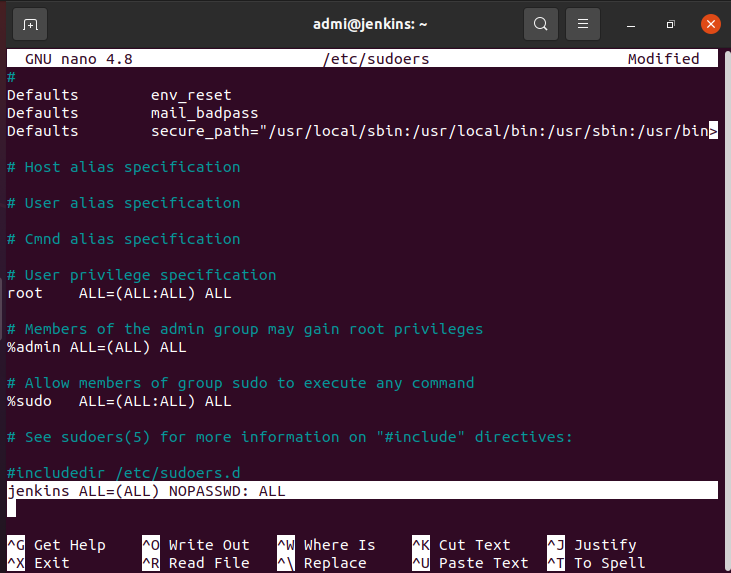
**sudo systemctl daemon-reload**

**sudo systemctl start jenkins**

**sudo systemctl status jenkins**

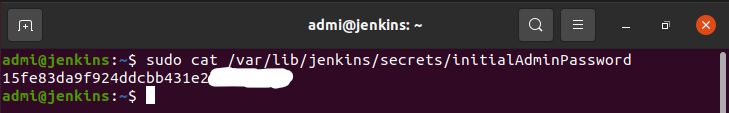


Полезным шагом в настройке является добавление пользователя jenkins в группу sudoers, чтобы он мог исполнять различные команды с root правами. Для этого в файл «/etc/sudoers» необходимо добавить следующую строку «jenkins ALL=(ALL) NOPASSWD: ALL»



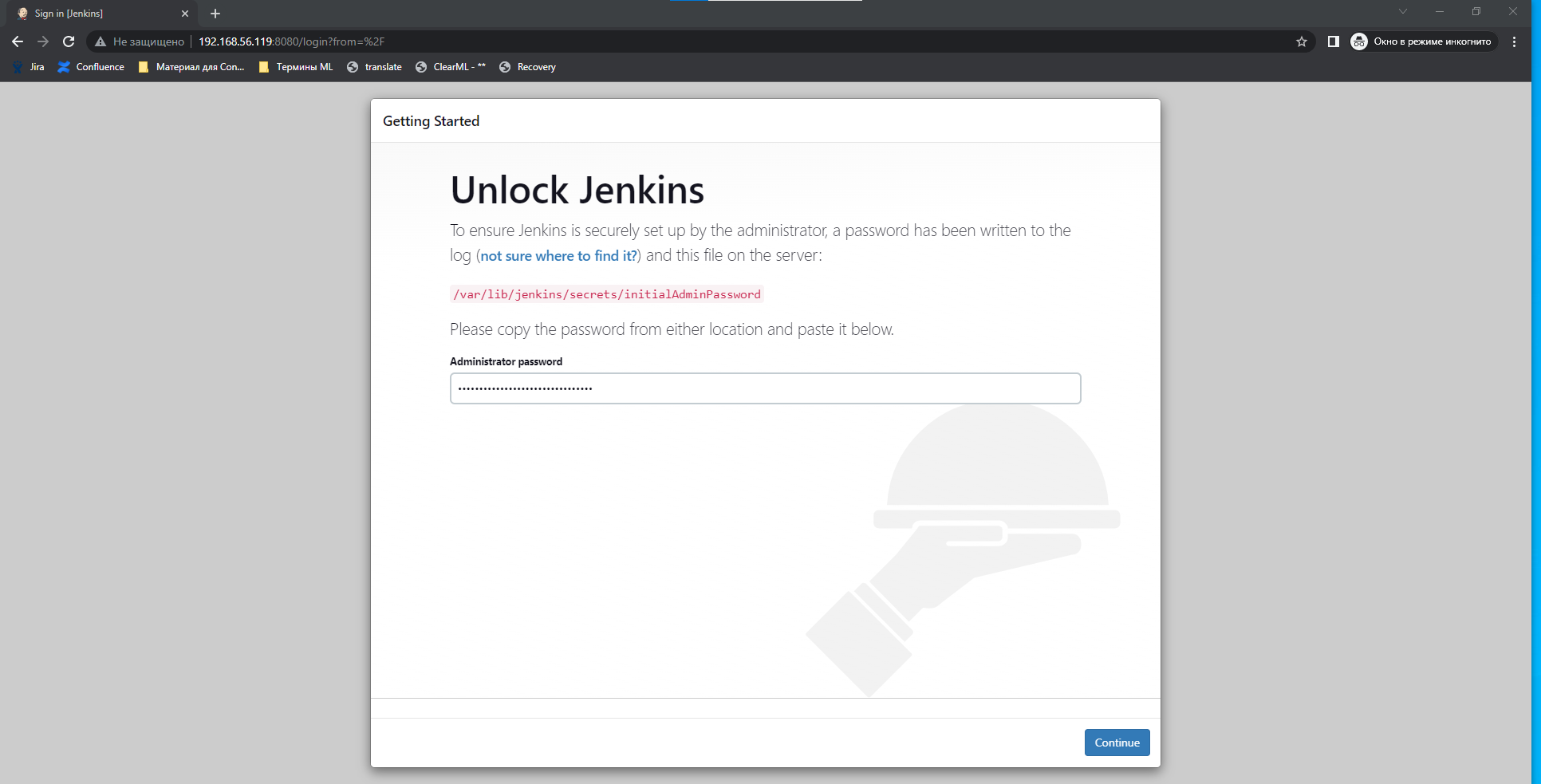
Также потребуется узнать токен для авторизации на сайте, который хранится в специально созданном файле.

**sudo cat /var/lib/jenkins/secrets/initialAdminPassword**

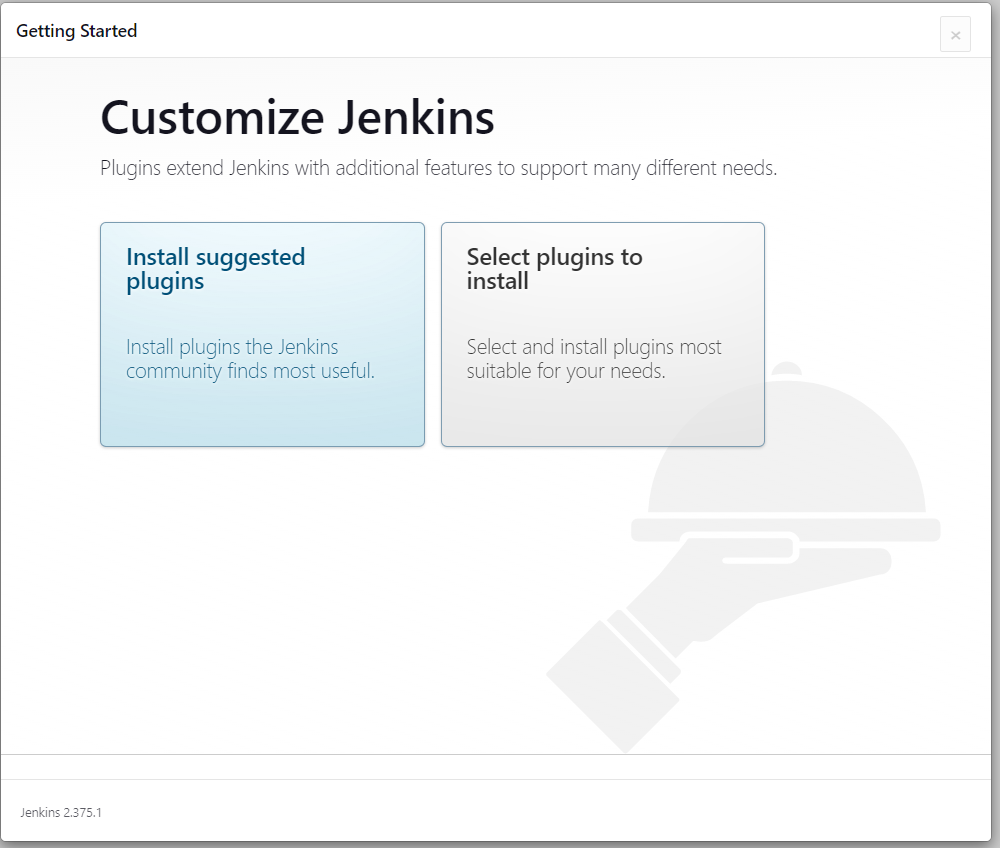


Дальше переходим на сайт «http:IP-Address:8080», где IP-Address – это адрес вашей машины.

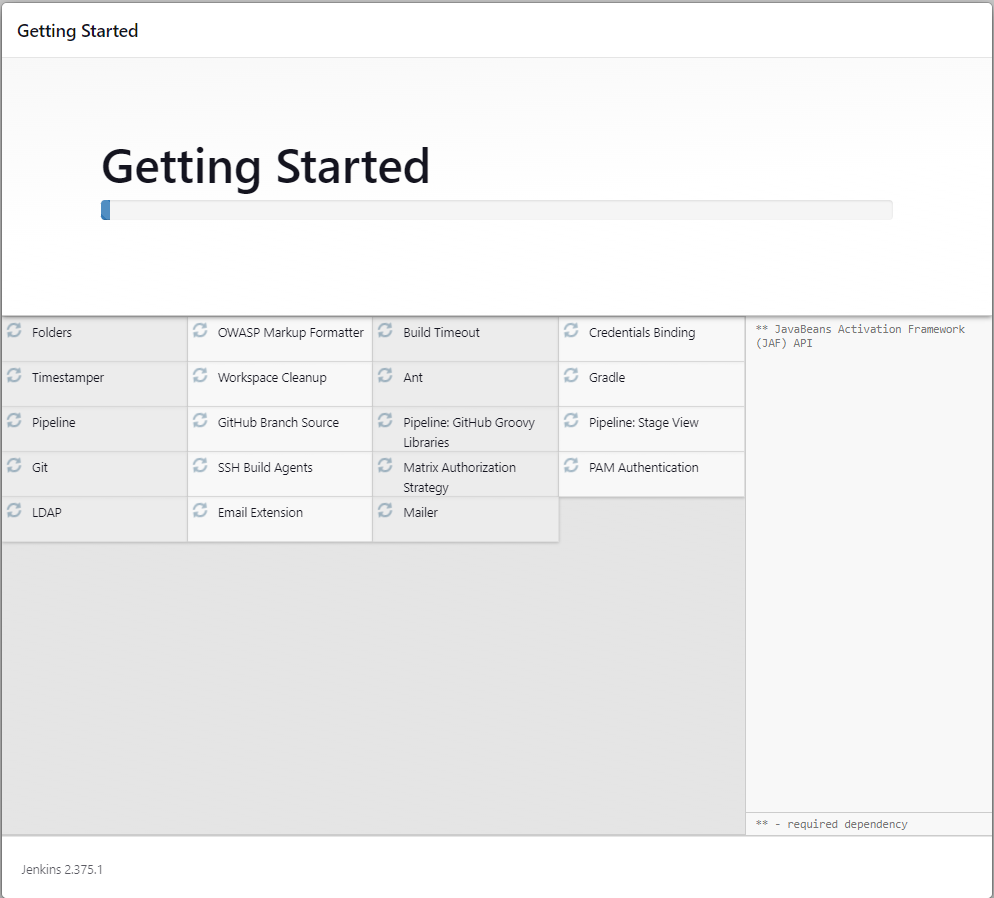
Далее для продолжения нам необходимо будет ввести сохраненный токен, который был получен ранее.



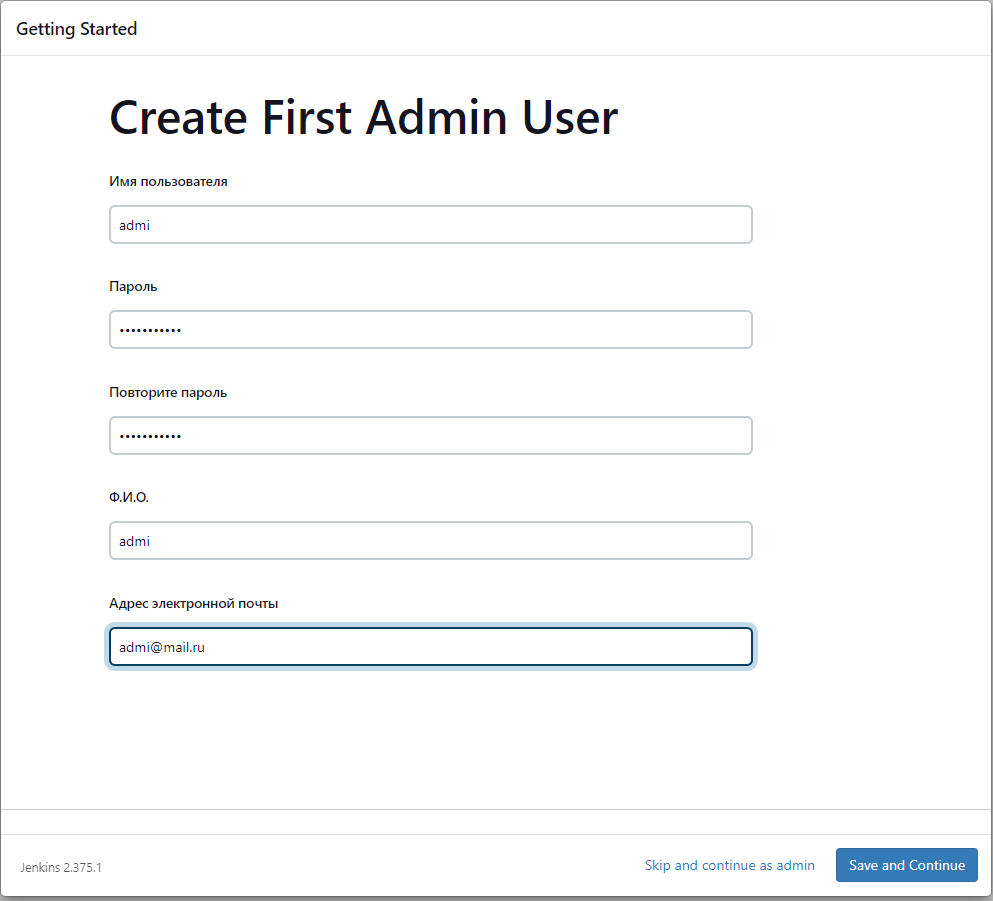
Далее необходимо установить плагины.



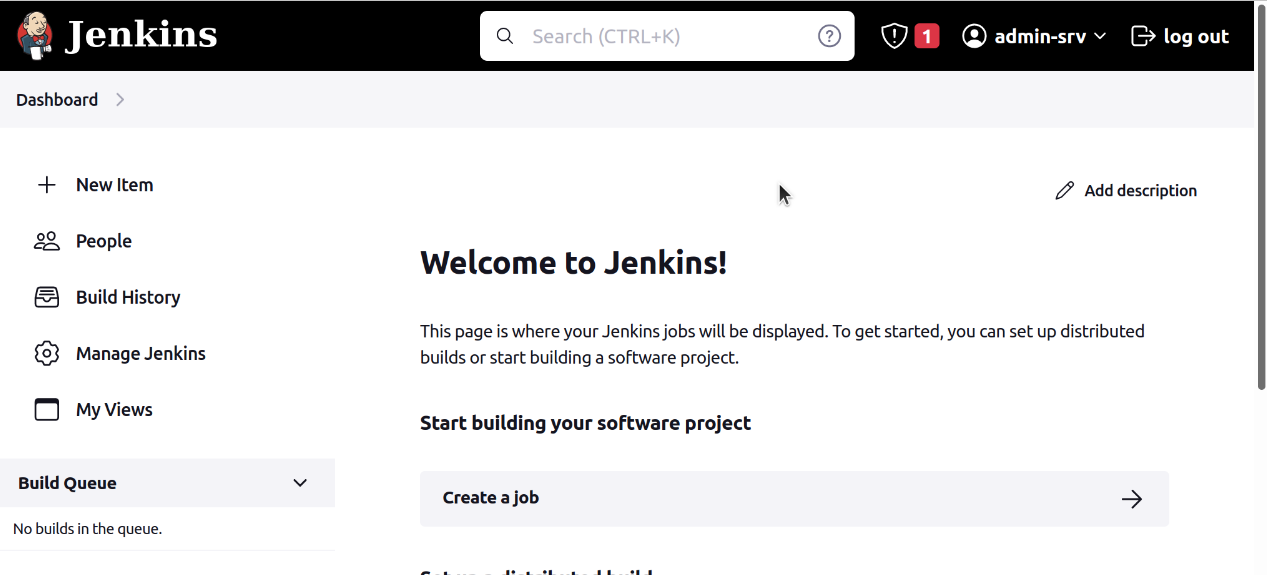
Можно выбрать предлагаемые плагины “Install suggested plugins”, либо выбрать плагины для установки самостоятельно «Select plugins to install»



Затем вы будете перенаправлены на страницу, где необходимо ввести данные для пользователя внутри Web-интерфейса.



После выполнения всех настроек и установки требуемых значений параметров (имя пользователя, пароль и пр.) откроется главное графическое меню для взаимодействия с Jenkins.



На этом установка Jenkins завершена.

## Тест

1. На какой виртуальный сервер была выполнена установка jenkins? (0.25)
   1. data-srv
   2. ml-srv
   3. **admin-srv**
   4. jenkins-srv
2. Без какого дополнительно устанавливаемого программного обеспечения не будет работать jenkins? (0.25)
   1. dvc
   2. clearml
   3. **java**
   4. sql
3. Как проверить, что сервис Jenkins успешно запущен? (0.25)
   1. sudo jenkins status
   2. **sudo systemctl status jenkins**
   3. **открыть в браузере страницу по адресу http://«IP сервера»:8080**
   4. sudo Jenkins info
4. Какой номер порта используется для web сервера jenkins по умолчанию? (0.25)
   1. 8000
   2. **8080**
   3. 8008
   4. 8888

## Итоги/выводы

В этом юните вы установили и настроили Jenkins для использования в следующих юнитах для решения задачи.

# Модуль 2. Юнит 3. Установка и настройка ansible.

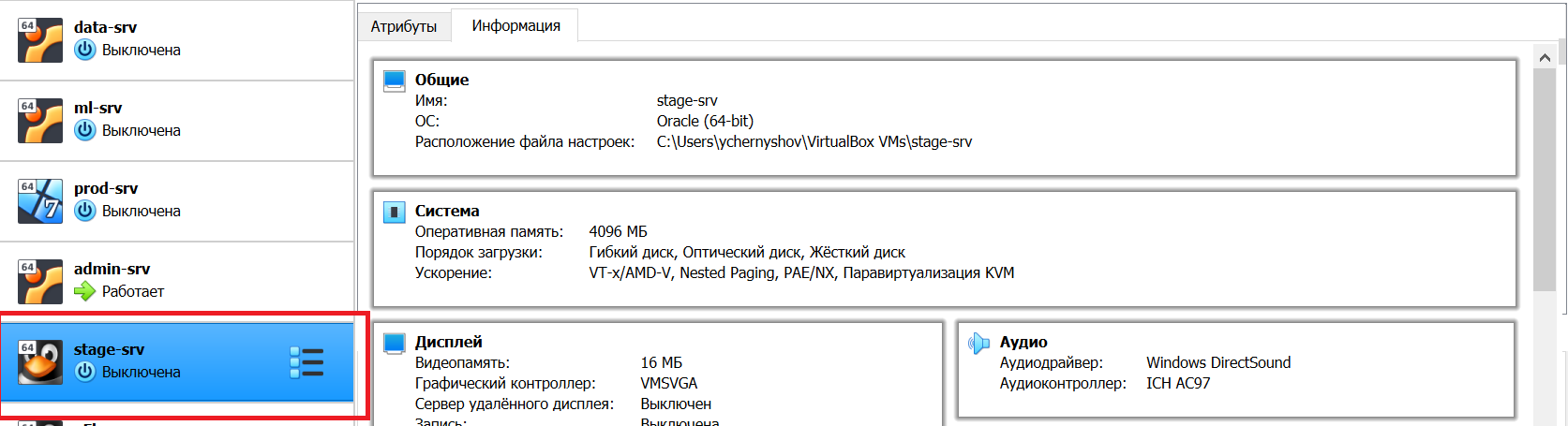
## Введение

Программное обеспечение ansible предназначено для автоматизации настройки инфраструктуры в соответствии с концепцией «Инфраструктура-как-Код» (Infrastructure-as-a-Code, IaaC). В этом юните вы установите ansible и настроите его для работы.

## Содержание:

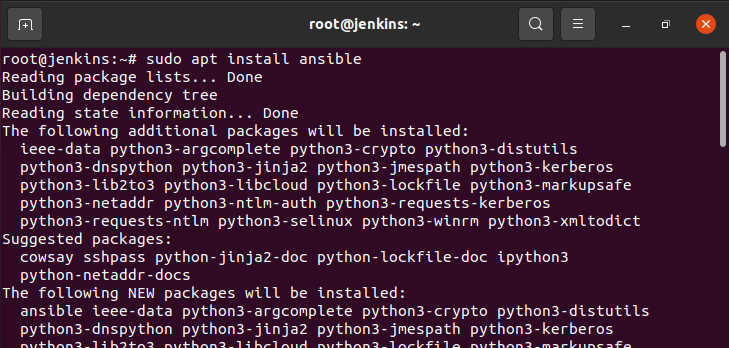
Ansible предназначен для автоматизированной настройки элементов инфраструктуры (серверов, пользовательских компьютеров, специализированных устройств, сети передачи данных), позволяет существенно снизить нагрузку на системных администраторов, ИТ-инженеров и DevOps/MLOps инженеров. Ansible неприхотлив в требованиях к своей работе, необходимо обеспечить ssh соединение между управляющим сервером и управляемыми устройствами, а также наличие python3 (и даже это требование не для всех задач). Для выполнения задач этого модуля ansible вы используете для автоматизированной настройки сервера тестирования stage и сервера эксплуатации prod.

Для целей тестирования вам осталось создать в инфраструктуре виртуальных серверов под управлением virtualbox последний сервер stage-srv, процедура установки аналогична уже выполнявшимся вами ранее действиям для создания других виртуальных серверов. Рекомендованный набор параметров следующий



Установка пакета Ansible должна быть выполнена на сервере «Сервер администратора» admin-srv с помощью команды

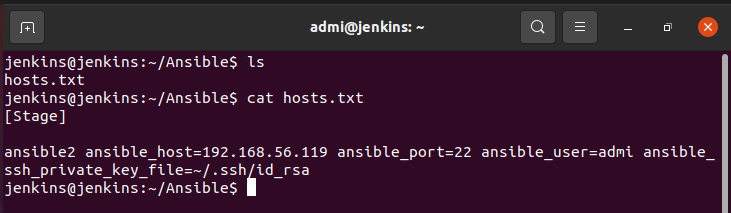
**sudo apt install ansible**



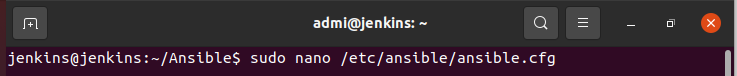
Дальше создайте файл «hosts.txt» и внутри этого файла укажите название виртуального сервера «Stage» и данные для подключения к этой машине. Синтаксис команды для создания подключения к удаленной машине:

***[Stage]***

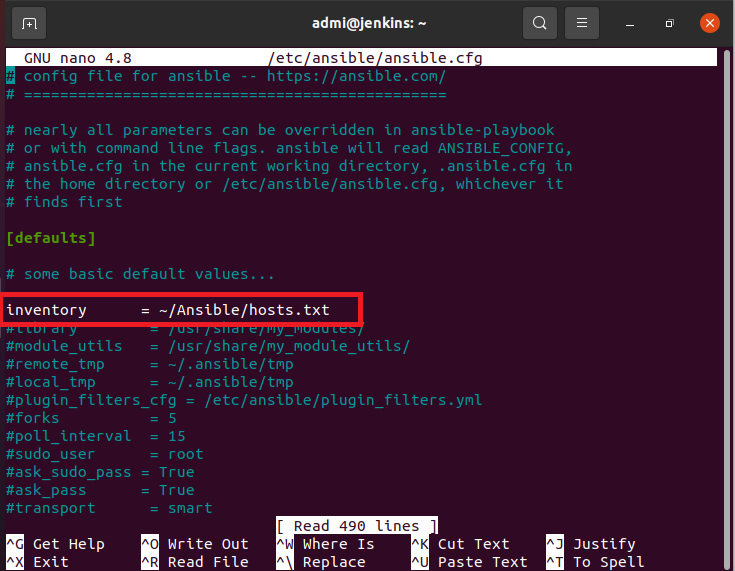
***ansible2 ansible\_host=192.168.56.119 ansible\_port=22 ansible\_user=stage-srv ansible\_ssh\_private\_key\_file=~/.ssh/id\_rsa***



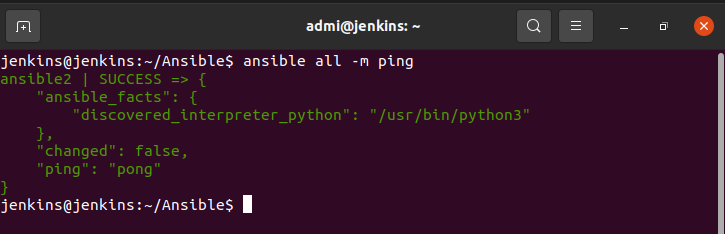
Далее измените конфигурационный файл Ansible, для подключения ранее созданного файла.



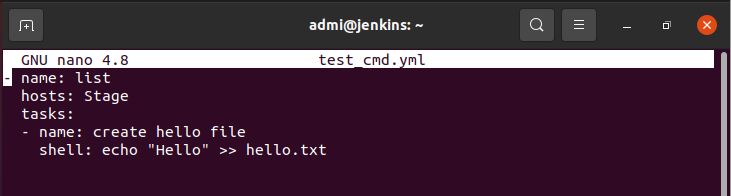
В конфигурационном файле необходимо раскоментировать строку «inventory» и указать путь до созданного файла. Из даного файла, будут браться устройства к которым можно будет подключаться.



Проверьте, что все данные корректно введены.

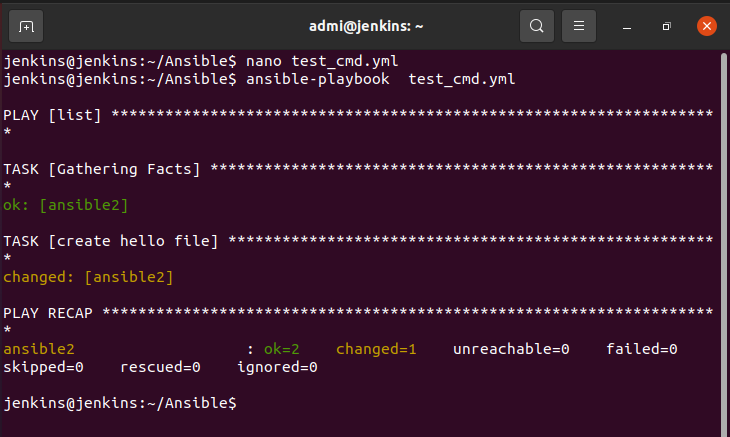


Далее создайте файл «test\_cmd.yml», в котором укажем шаги, которые будут исполняться на удаленном устройстве. Например, в данном случае создается файл, «hello.txt», в который запишется текст «Hello».

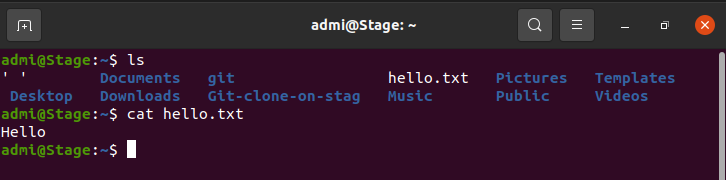


После создания ansible плейбука (ansible playbook) запустите его с использованием команды

**ansible-playbook test\_cmd.yml**



после чего на удаленной машине можно увидеть созданный файл, с установленным текстом, который вы задали в playbook.



Теперь ansible установлен и настроен для работы.

## Тест

1. Какие инструменты нужны для корректной работы ansible? (0.25)
   1. **python3**
   2. ftp
   3. **ssh**
   4. telnet
2. На какой сервер устанавливается управляющая часть ansible? (0.25)
   1. data-srv
   2. ansible-srv
   3. **admin-srv**
   4. ml-srv
3. Как называется файл, содержащий перечень команд, используемый ansible для настройки удаленных серверов? (0.25)
   1. инструкция
   2. **плейбук**
   3. commands
   4. actions
4. Какая команда ansible опрашивает все удаленные сервера? (0.25)
   1. ansible echo all
   2. ansible get reply
   3. **ansible all -m ping**
   4. ansible all -m ask servers

## Итоги/выводы

В этом юните вы создали виртуальные сервер для тестирования stage-srv, а также установили на управляющий сервер admin-srv программное обеспечение ansible для автоматизированной настройки элементов инфраструктуры.

# Модуль 4. Юнит 4. Установка и настройка pytest для тестирования.

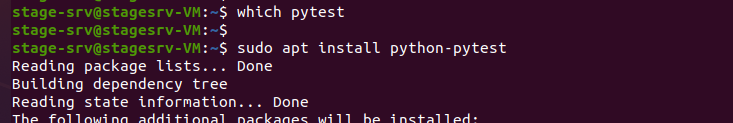
## Введение

Назначение тестирования в любом проекте разработки программного обеспечения одинаковое и очень важное: с помощью тестов проверяют, что после внесенных изменений в программном коде и настройках разрабатываемая система продолжает работать в соответствии с функциональными и техническими требованиями, по которым эта система разрабатывается. Существует множество различных видов тестирования и инструментов. В этом юните настроим pytest для использования в следующих юнитах.

## Содержание

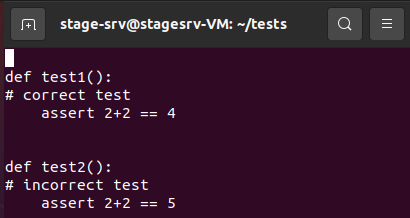
Pytest это python библиотека для написания небольших удобно читаемых тестов, а также для их масштабирования на большие проекты для функционального тестирования приложений или библиотек.

Библиотека pytest не входит в стандартный набор программ, устанавливаемый вместе с python, поэтому ее необходимо устанавливать отдельно.

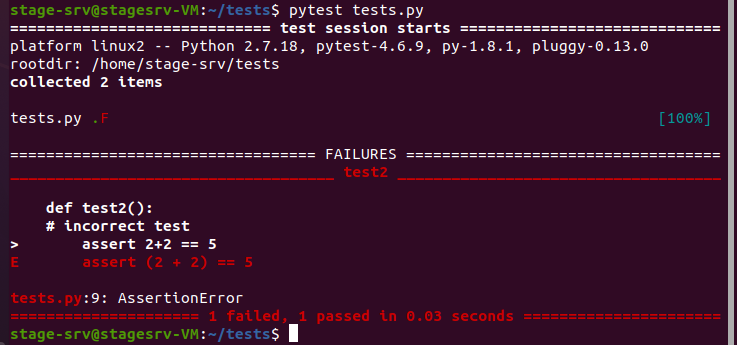




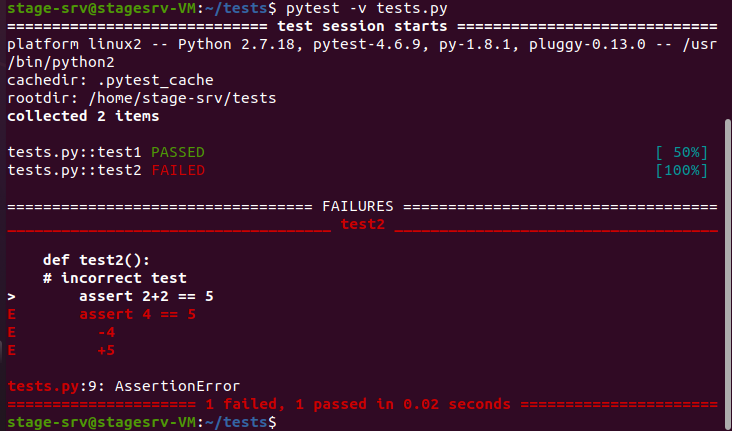
Для проверки и демонстрации работоспособности pytest создадим две функции в файле tests.py, одну корректную, а другую с логической ошибкой



При записке pytest tests.py вы увидите результат проверки двух функций.

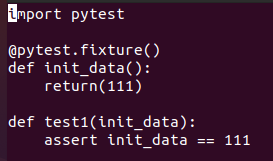


Если выполнить pytest с флагом -v, то можно увидеть еще более подробное описание, указывающее на причину ошибки



И, в заключение знакомства с pytest давайте познакомимся с понятием фикстур (fixture). Фикстуры необходимы для структурирования тестирующего кода, давая возможность добавлять стандартные части кода, необходимые для конкретной программной среды. Это функции, выполняемые pytest до или после тестовых функций для инициализации или завершения теста. Код в фикстуре может выполнять любые операции, например, вы можете использовать фикстуры, чтобы загрузить датасет. Это позволяет правильно инициализировать тестовое окружение, а также корректно освободить ресурсы после завершения работы.

Вот простой пример фикстуры:



Теперь pytest готов к использованию в проекте.

## Тест

1. На каком сервере выполняется установка и последующее использование pytest в проекте? (0.25)
   1. data-srv
   2. ml-srv
   3. **stage-srv**
   4. prod-srv
2. Какой флаг позволяет сделать вывод pytest более подробным? (0.25)
   1. -a
   2. -b
   3. **-v**
   4. -z
3. Что такое фикстуры в pytest? (0.25)
   1. Специальные константы
   2. **Специальные функции-декораторы**
   3. Специальные функции агрегаторы
   4. **Функции для выполнения типовых операций в начале или конце теста**
4. Какой стандартный метод в python и pytest используется для проверки утверждений? (0.25)
   1. check
   2. **assert**
   3. case
   4. statement

## Итоги/выводы

В этом юните вы установили и настроили для работы программное обеспечение pytest на сервере тестирования stage-srv.

# Итоги/выводы по модулю

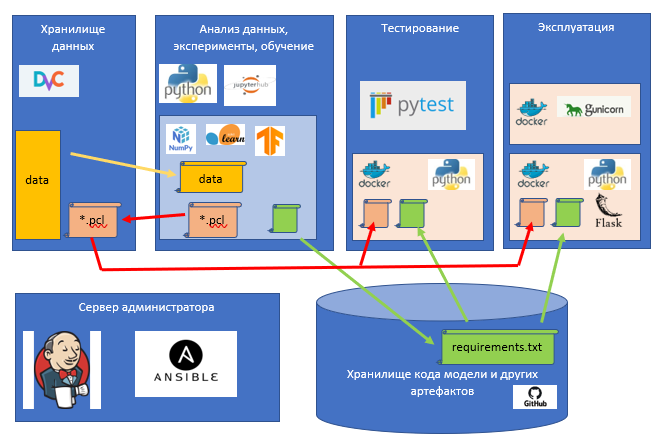
В этом модуле вы решали задачу: создать инфраструктуру и установить необходимое программное обеспечение для небольшого проекта машинного обучения. Причем часть этой инфраструктуры вами уже была создана ранее в Модуле 1, в котором вы уже создали и настроили серверы data-srv для работы с данными и ml-srv для проведения исследований и обучения модели.

В этом модуле вы создали и настроили виртуальные сервера admin-srv для работы системного администратора и stage-srv для тестирования.

Эта инфраструктура будет использоваться вами далее.

# Задание для самостоятельной работы

Вам необходимо, с применением полученных знаний, выполнить настройку всей инфраструктуры для дальнейшей работы



В качестве результата необходимо предъявить конфигурационный файл ansible и результат выполнения ansible плейбука, который выполняет функцию “uname -a” на всех серверах инфраструктуры.