

Автоматизация тестирования консольных приложений Linux на Python

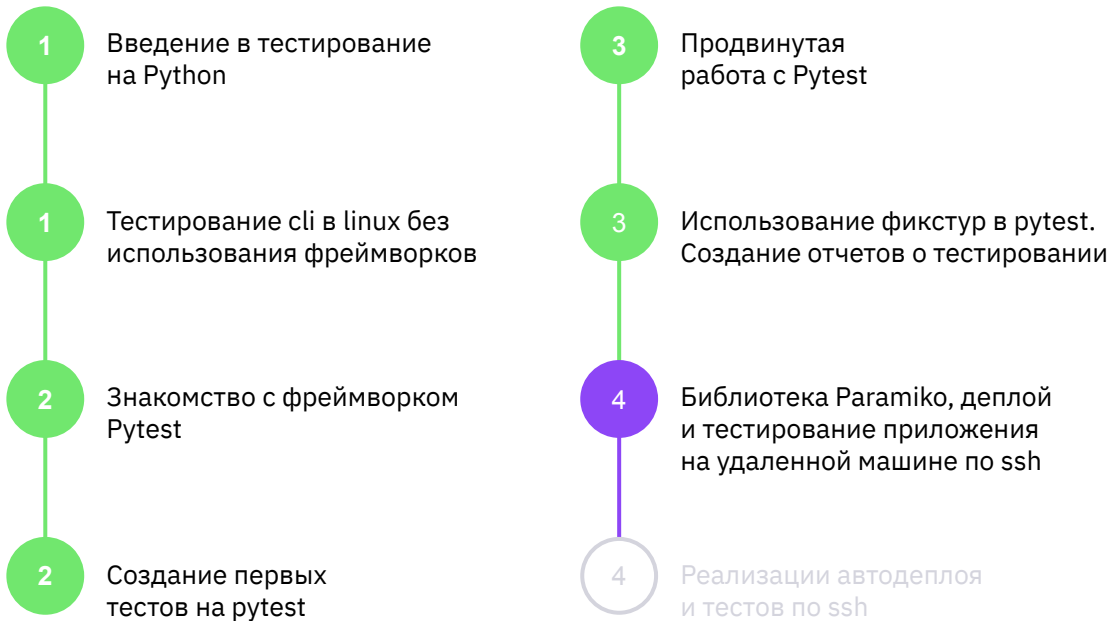
Лекция 4

Библиотека Paramiko, деплой и тестирование
приложения на удаленной машине по ssh.





План курса





Библиотека Paramiko, деплой
и тестирование приложения
на удаленной машине по ssh



Что будет на уроке сегодня

- 📌 Как настроить ssh-сервер в Linux
- 📌 Как работать с библиотекой paramiko
- 📌 Как устанавливать и удалять программы в Ubuntu Linux
- 📌 Как реализовать автодеплой
- 📌 Как работать с журналами Linux
- 📌 Как настроить автозапуск тестов в Jenkins





Поехали!



Включение
и настройка ssh и sftp





SSH

SSH — защищенный протокол для удаленного доступа к компьютерам.

Через SSH можно выполнять операции в командной строке компьютера, который физически находится в другом месте.

```
ssh username@remote_host -p port
```

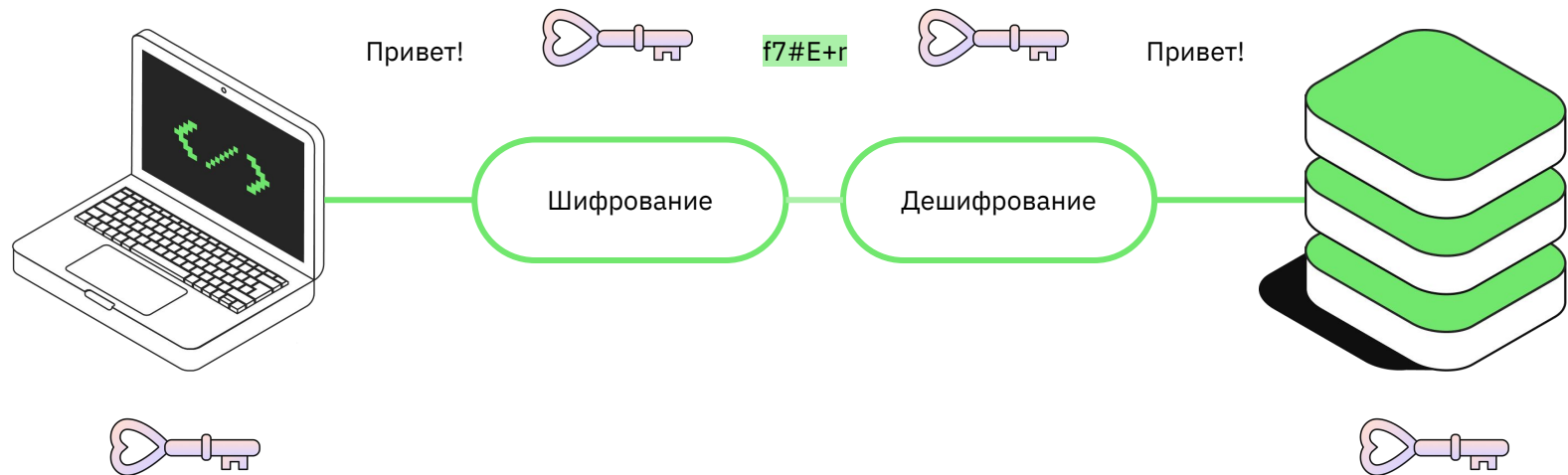




SSH

SSH-клиент

SSH-сервер





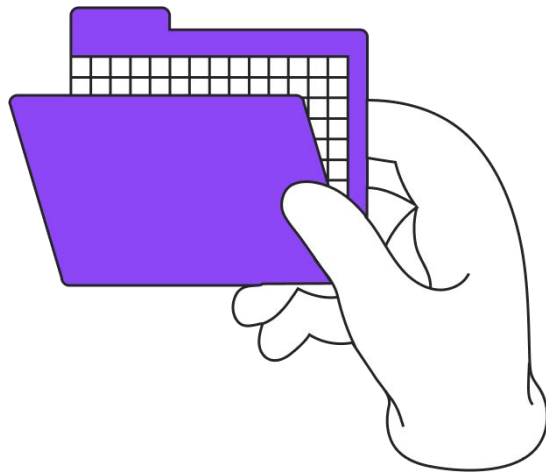
SFTP

SFTP, протокол передачи файлов по SSH, или безопасный протокол передачи файлов, — это отдельный протокол, поддерживающий SSH.

Его преимуществом является возможность использования защищенного подключения для передачи файлов и просмотра файловой системы как на локальной, так и на удаленной системе.

`get remoteFile`

`put localFile`





Включение ssh в Ubuntu





Библиотека paramiko

Paramiko — это модуль для python, который реализует ssh-протокол для защищенного соединения с удаленным компьютером.

При подключении предоставляется высокоуровневое API для работы с ssh — создается объект SSHClient.

Для большего контроля можно передать сокет (или подобный объект) классу Transport и работать с удаленным хостом в качестве сервера или клиента.

Клиенту для аутентификации можно использовать пароль или приватный ключ и проверку ключа сервера.















Реализация `ssh_checkout`
и `upload_files`





Управление пакетами Ubuntu

| | | |
|---|-------------------------|---|
|  | установить пакет | <code>dpkg -i package_name.deb</code> |
|  | показать список | <code>dpkg -l search_pattern</code> |
|  | удаление пакета | <code>dpkg -r package_name.deb</code> |
|  | отображение содержимого | <code>dpkg --contents package_name.deb</code> |
|  | проверка установки | <code>dpkg -s package_name.deb</code> |
|  | проверка пути | <code>dpkg -L package_name.deb</code> |
|  | подробная информация | <code>dpkg -p package_name.deb</code> |
|  | рекурсивная установка | <code>dpkg -R --install /deb-files-location/</code> |
|  | распаковка | <code>dpkg --unpack package_name.deb</code> |
|  | пересборка | <code>dpkg --configure package_name</code> |



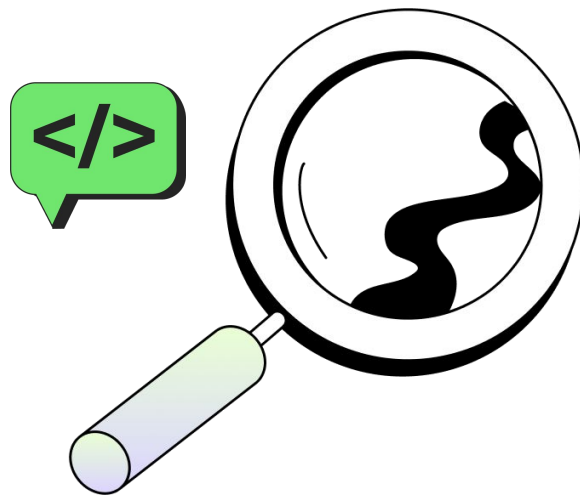
Автодеплой



Развертывание при помощи автоматизированных решений и называется автодеплоем.

Основные преимущества автодепоя:

- ✓ развертывание сразу в нескольких средах;
- ✓ повышение надежности и предсказуемости развертываемого ПО;
- ✓ минимизация ошибок, связанных с человеческим фактором;
- ✓ снижение затрат на развертывание;
- ✓ упрощение рабочего процесса





Вопрос

Подумайте, что должен проверять автодеплой?





Ответ

Код возврата, сообщение установщика,
наличие в списке пакетов.



Написание
автодеплойа





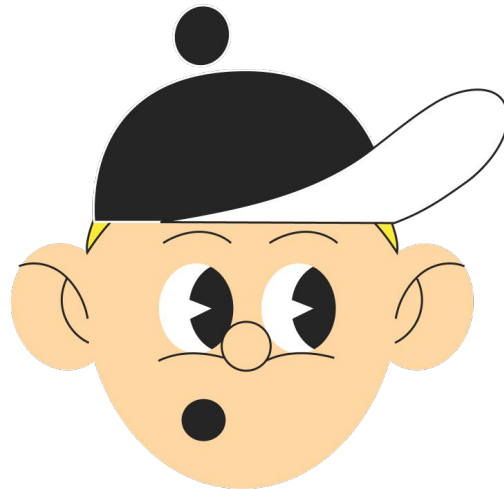
Проверка журналов (логов)



В современных дистрибутивах Linux для просмотра логов определенного сервиса или загрузки системы необходимо использовать утилиту `journalctl`.

Примеры команд:

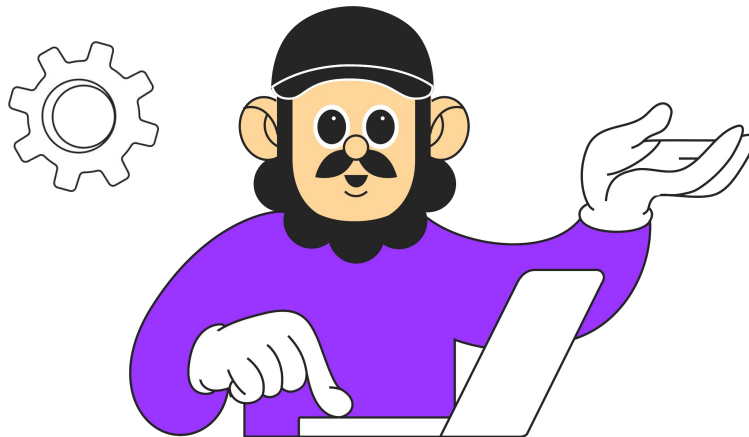
- ✓ `journalctl`
- ✓ `sudo journalctl --since "2023-01-20 15:10:10"`
- ✓ `sudo journalctl --until "2023-01-20 15:15:50"`
- ✓ `journalctl -p 0`





Уровни важности:

- ✓ **0: emergency** (неработоспособность системы)
- ✓ **1: alerts** (предупреждения, требующие немедленного вмешательства)
- ✓ **2: critical** (критическое состояние)
- ✓ **3: errors** (ошибки)
- ✓ **4: warning** (предупреждения)
- ✓ **5: notice** (уведомления)
- ✓ **6: info** (информационные сообщения)
- ✓ **7: debug** (отладочные сообщения)





Интеграция с Jenkins



Добавление шага запуска тестов

⋮

Execute shell ?

×

×

Command

See [the list of available environment variables](#)

```
bash -e startests.sh
```

Advanced...



Добавление шага выгрузки результатов

Post-build Actions

☰ Publish JUnit test result report ?

×

Test report XMLs

Fileset **'includes'** setting that specifies the generated raw XML report files, such as 'myproject/target/test-reports/*.xml'.

Basedir of the fileset is **the workspace root**.

results.xml

?

☐ Retain long standard output/error

Health report amplification factor ?

1,0

1% failing tests scores as 99% health. 5% failing tests scores as 95% health

Allow empty results ?



Что мы узнали сегодня

- 📌 Как настроить ssh-сервер в Linux
- 📌 Как работать с библиотекой paramiko
- 📌 Как устанавливать и удалять программы в Ubuntu Linux
- 📌 Как реализовать автодеплой
- 📌 Как работать с журналами Linux
- 📌 Как настроить автозапуск тестов в Jenkins





Спасибо за внимание!