

Основы системного администрирования и установка Python

Гущин Александр, весна 2021

Операционные системы

Наиболее часто нам придется работать с Linux - на нем работает большинство серверов, на которых запускается продакшн код, research, платформы CI/CD.

Можно использовать разные дистрибутивы Linux. Одни из наиболее известных - Debian и его производные (Ubuntu, LinuxMint), CentOS (RHEL), Alpine (busybox). К счастью, принципы функционирования достаточно похожи.

Локально мы часто можем иметь дело также с MacOS и Windows.

Linux

bash - командная оболочка Linux, также командная строка или терминал.

Типичные команды:

```
1 cd myfolder
2 ls .
3 sudo apt-get install python
4 pip install -r requirements.txt
5 python app.py
6 mkdir tempfolder
7 rm -rf tempfolder
```

<https://linuxconfig.org/bash-scripting-tutorial-for-beginners>

Варианты для Linux/MacOs: sh, bash, zsh, ksh...

Менеджеры пакетов Linux

apt-get (Debian)

```
1 apt-get update
2 apt-get install nano -y
```

yum (CentOS)

```
1 yum update -y
2 yum install nano
```

MacOS

Менеджер пакетов Homebrew

```
1 /bin/bash -c "$(curl -fsSL https://raw.githubusercontent.com/Homebrew/install/HEAD/install.sh)"
2 brew update && brew upgrade
3 brew install git
```

Windows

???

Текстовые редакторы

- Удобно использовать, когда мы работаем на удаленном сервере через консоль: Nano, Vim. Следует изучить хотя бы один, чтобы не испытывать страданий :)
- Удобно для работы на своем ноутбуке: Sublime, Atom
- Тutorials:
 - <https://www.openvim.com>
 - <https://www.hostinger.com/tutorials/how-to-install-and-use-nano-text-editor>
- CheatSheets:
 - <https://devhints.io/vim>
 - <https://www.nano-editor.org/dist/latest/cheatsheet.html>

Среды разработки

Для анализа данных: Jupyter notebook, Jupyter lab.

Для подготовки кода к production: Pycharm, VSCode.

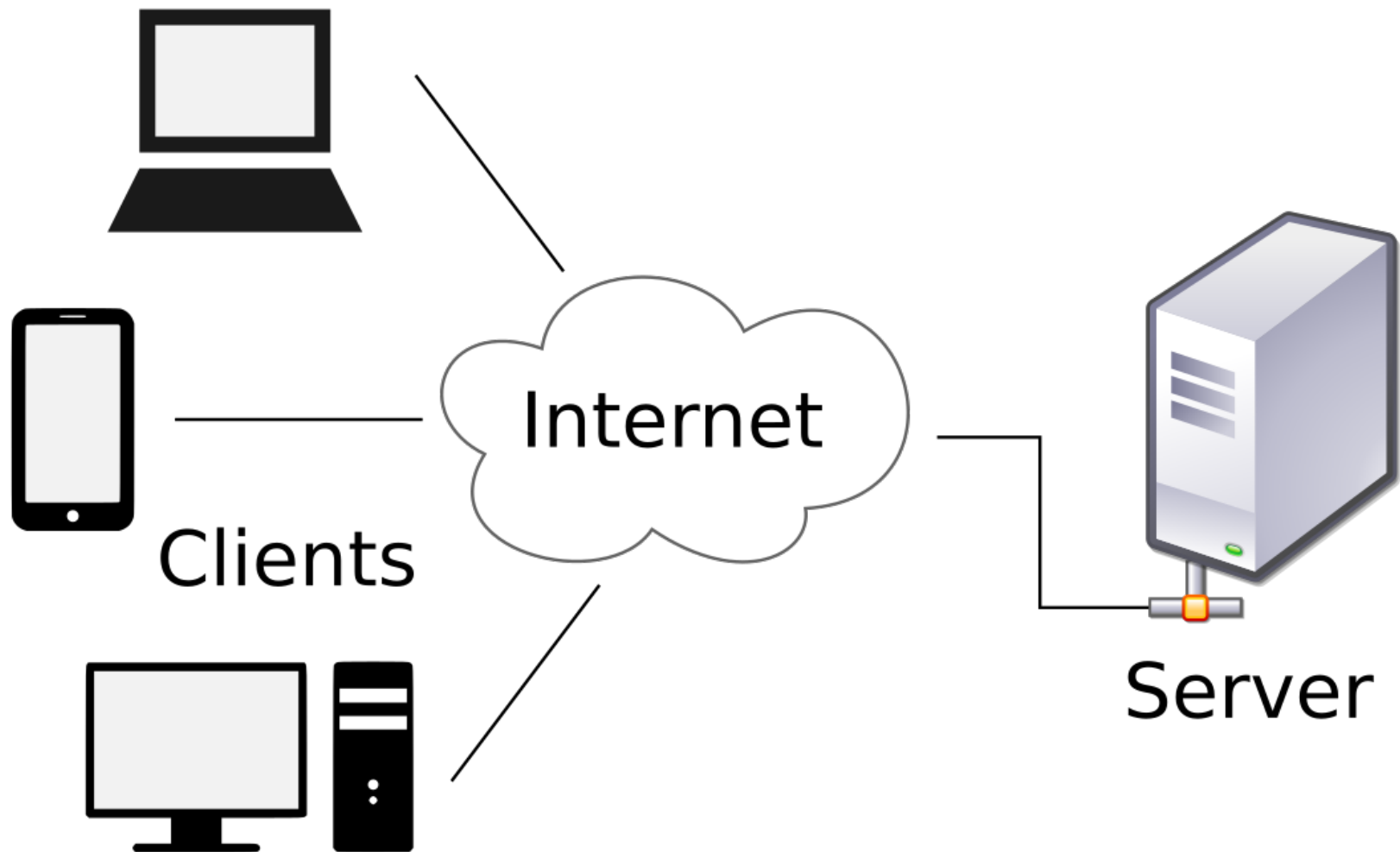
Jupyter

Запуск: `jupyter notebook`, `jupyter lab`

Для использования виртуального окружения в Jupyter

```
1 | pipenv install ipykernel --dev
2 | pipenv shell # заходим в виртуальное окружение
3 | python -m ipykernel install --user --name=py38
4 | jupyter lab # or notebook
```

Компьютерные сети



```
1 → ~ ping yandex.ru
2 PING yandex.ru (77.88.55.55): 56 data bytes
3 64 bytes from 77.88.55.55: icmp_seq=0 ttl=54 time=15.778 ms
4 ^C
5 --- yandex.ru ping statistics ---
6 1 packets transmitted, 1 packets received, 0.0% packet loss
7 round-trip min/avg/max/stddev = 15.778/15.778/15.778/0.000 ms
```

SSH - локально

```
1 ~ ssh -i ~/.ssh/europe-north.pem ubuntu@13.53.39.46
2 Welcome to Ubuntu 18.04.4 LTS (GNU/Linux 4.15.0-20-generic x86_64)
3 ...
4 Last login: Fri Dec 18 16:05:46 2020 from 88.255.91.110
5 laguschin@ec2-server:~$
```

<https://www.debian.org/devel/passwordlessssh> (утилита ssh-keygen)

<https://scotch.io/tutorials/how-to-create-an-ssh-shortcut>

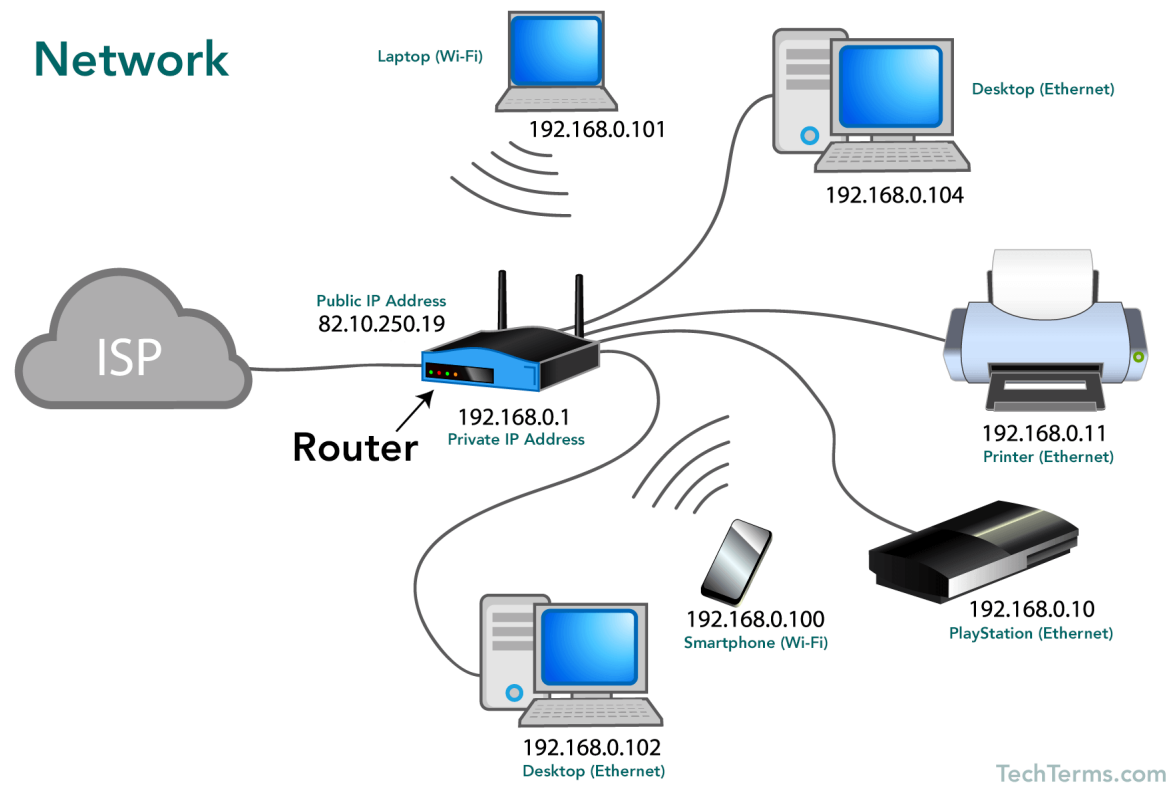
Если у вас windows, пригодится <http://winscp.net>

SSH - на сервере

Публичные ключи нужно сложить в `.ssh/authorized_keys`

```
1 cd
2 mkdir .ssh
3 touch .ssh/authorized_keys
4 chmod 700 .ssh
5 chmod 600 .ssh/authorized_keys
6 nano .ssh/authorized_keys
```

Network

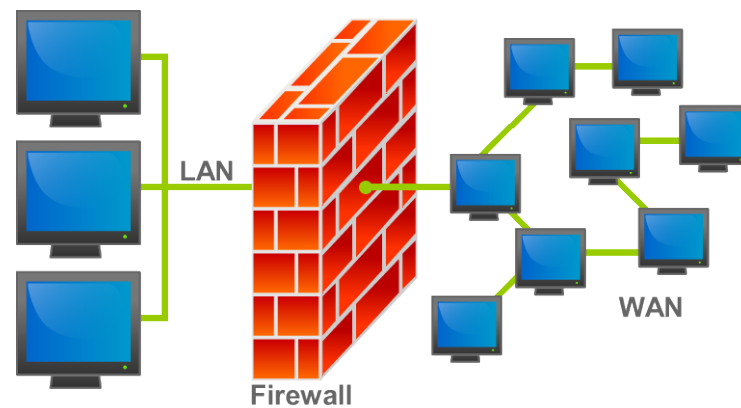


<https://yzrkiller.home.blog/2019/06/22/computer-network/>

Порты

- 22: Secure Shell (SSH)
- 80: Hypertext Transfer Protocol (HTTP) used in World Wide Web
- 443: HTTP Secure (HTTPS) HTTP over TLS/SSL

```
1 ~ jupyter lab --port 8888
2 The port 8888 is already in use, trying another port.
3 Jupyter Notebook 6.1.1 is running at:
4 http://localhost:8889/
5 Use Control-C to stop this server and shut down all kernels (twice to skip confirmation)
```



Пример из админки на aws ec2:

Edit inbound rules ✕

Type ⓘ	Protocol ⓘ	Port Range ⓘ	Source ⓘ	
SSH ⌵	TCP	22	Anywhere ⌵ 0.0.0.0/0	✕
HTTP ⌵	TCP	80	Anywhere ⌵ 0.0.0.0/0	✕
HTTPS ⌵	TCP	443	Anywhere ⌵ 0.0.0.0/0	✕

Add RuleCancelSave

Port forwarding

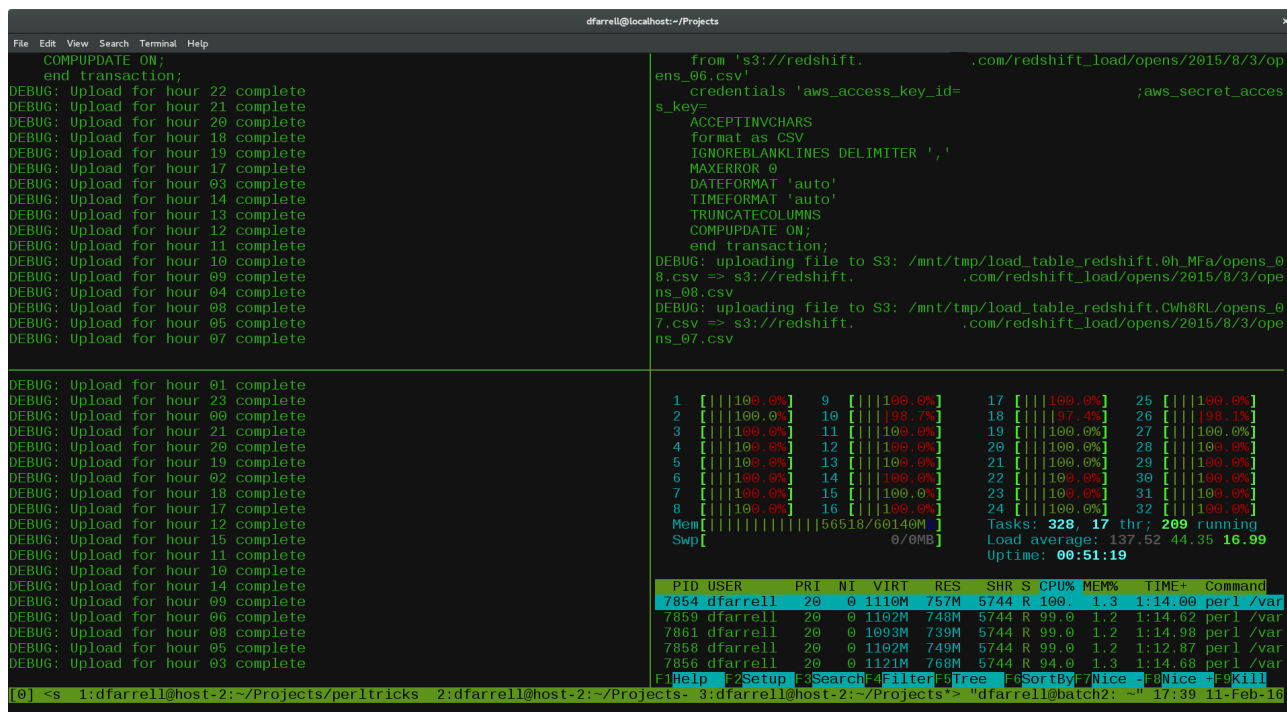
Как подключиться к Jupyter notebook на сервере, если закрыты все порты, кроме 22 (ssh)?

```
1  ssh -L $LOCAL_PORT:$MY_SERVER_IP:$REMOTE_PORT $MY_SERVER_IP
2  ssh -L 10001:myserver:8888 myserver
```

Открываем в браузере <http://localhost:10001> и работаем с Jupyter notebook

Запуск процессов на сервере

Когда мы закрываем терминал и прерываем работу с удаленным сервером, процесс в терминале также прерывается. Как избежать этого? **Tmux, screen**



```
File Edit View Search Terminal Help
dfarrell@localhost:~/Projects

COMPUTE ON;
end transaction;
DEBUG: Upload for hour 22 complete
DEBUG: Upload for hour 21 complete
DEBUG: Upload for hour 20 complete
DEBUG: Upload for hour 18 complete
DEBUG: Upload for hour 19 complete
DEBUG: Upload for hour 17 complete
DEBUG: Upload for hour 03 complete
DEBUG: Upload for hour 14 complete
DEBUG: Upload for hour 13 complete
DEBUG: Upload for hour 12 complete
DEBUG: Upload for hour 11 complete
DEBUG: Upload for hour 10 complete
DEBUG: Upload for hour 09 complete
DEBUG: Upload for hour 04 complete
DEBUG: Upload for hour 08 complete
DEBUG: Upload for hour 06 complete
DEBUG: Upload for hour 07 complete

DEBUG: Upload for hour 01 complete
DEBUG: Upload for hour 23 complete
DEBUG: Upload for hour 00 complete
DEBUG: Upload for hour 21 complete
DEBUG: Upload for hour 20 complete
DEBUG: Upload for hour 19 complete
DEBUG: Upload for hour 02 complete
DEBUG: Upload for hour 18 complete
DEBUG: Upload for hour 17 complete
DEBUG: Upload for hour 12 complete
DEBUG: Upload for hour 11 complete
DEBUG: Upload for hour 10 complete
DEBUG: Upload for hour 14 complete
DEBUG: Upload for hour 09 complete
DEBUG: Upload for hour 06 complete
DEBUG: Upload for hour 08 complete
DEBUG: Upload for hour 06 complete
DEBUG: Upload for hour 03 complete

from 's3://redshift.
ens_06.csv'
credentials 'aws_access_key_id=
s_key=
ACCEPTIMVCHARS
format as CSV
IGNOREBLANKLINES DELIMITER ','
MAXERROR 0
DATEFORMAT 'auto'
TIMEFORMAT 'auto'
TRUNCATECOLUMNS
COMPUTE ON;
end transaction;
DEBUG: uploading file to S3: /mnt/tmp/load_table_redshift.0h.MFa/opens_0
8.csv => s3://redshift.
.com/redshift_load/opens/2015/8/3/ope
ns_08.csv
DEBUG: uploading file to S3: /mnt/tmp/load_table_redshift.CWh8RL/opens_0
7.csv => s3://redshift.
.com/redshift_load/opens/2015/8/3/ope
ns_07.csv

1 [||||100.0%] 9 [||||100.0%] 17 [||||100.0%] 25 [||||100.0%]
2 [||||100.0%] 10 [||||98.7%] 18 [||||97.4%] 26 [||||98.1%]
3 [||||100.0%] 11 [||||100.0%] 19 [||||100.0%] 27 [||||100.0%]
4 [||||100.0%] 12 [||||100.0%] 20 [||||100.0%] 28 [||||100.0%]
5 [||||100.0%] 13 [||||100.0%] 21 [||||100.0%] 29 [||||100.0%]
6 [||||100.0%] 14 [||||100.0%] 22 [||||100.0%] 30 [||||100.0%]
7 [||||100.0%] 15 [||||100.0%] 23 [||||100.0%] 31 [||||100.0%]
8 [||||100.0%] 16 [||||100.0%] 24 [||||100.0%] 32 [||||100.0%]
Mem[|||||||||56618/60140M]
Swp[|||||||||0/0MB]
Tasks: 328, 17 thr; 209 running
Load average: 137.52 44.36 16.99
Uptime: 00:51:19

PID USER PRI NI VIRT RES SHR S CPU% MEM% TIME+ Command
7854 dfarrell 20 0 1110M 757M 5744 R 100 1.3 1:14.00 perl /var
7859 dfarrell 20 0 1102M 748M 5744 R 99 0 1.2 1:14.02 perl /var
7861 dfarrell 20 0 1093M 739M 5744 R 99 0 1.2 1:14.08 perl /var
7858 dfarrell 20 0 1102M 749M 5744 R 99 0 1.2 1:12.87 perl /var
7856 dfarrell 20 0 1121M 768M 5744 R 94 0 1.3 1:14.68 perl /var

[0] <S 1:dfarrell@host-2: ~/Projects/perltricks 2:dfarrell@host-2: ~/Projects- 3:dfarrell@host-2: ~/Projects> "dfarrell@batch2: ~" 17:39 11-Feb-16
```

Ищите cheatsheets. Придется выучить один из них :)

Загрузка машинки

top, atop, htop помогут следить за потреблением RAM, CPU

```
Terminal - [screen 1: bash] jo68huc@meggy:~
File Edit View Terminal Go Help

1 [|||||] 100.0% 13 [|||||] 100.0% 25 [|||||] 0.0% 37 [|||||] 100.0%
2 [|||||] 100.0% 14 [|||||] 100.0% 26 [|||||] 0.0% 38 [|||||] 17.2%
3 [|||||] 100.0% 15 [|||||] 100.0% 27 [|||||] 0.0% 39 [|||||] 0.0%
4 [|||||] 100.0% 16 [|||||] 100.0% 28 [|||||] 0.0% 40 [|||||] 0.0%
5 [|||||] 8.6% 17 [|||||] 100.0% 29 [|||||] 100.0% 41 [|||||] 0.0%
6 [|||||] 0.6% 18 [|||||] 100.0% 30 [|||||] 0.0% 42 [|||||] 0.0%
7 [|||||] 100.0% 19 [|||||] 0.0% 31 [|||||] 0.0% 43 [|||||] 6.7%
8 [|||||] 100.0% 20 [|||||] 0.6% 32 [|||||] 0.0% 44 [|||||] 100.0%
9 [|||||] 0.6% 21 [|||||] 82.8% 33 [|||||] 100.0% 45 [|||||] 100.0%
10 [|||||] 100.0% 22 [|||||] 100.0% 34 [|||||] 0.0% 46 [|||||] 100.0%
11 [|||||] 100.0% 23 [|||||] 100.0% 35 [|||||] 0.0% 47 [|||||] 100.0%
12 [|||||] 100.0% 24 [|||||] 0.0% 36 [|||||] 100.0% 48 [|||||] 100.0%
Mem [|||||] 38453/258325MB Tasks: 359, 910 thr; 28 running
Swp [|||||] 407/4095MB Load average: 27.17 27.10 27.02
Uptime: 166 days(!), 17:02:16

PID USER PRI NI VIRT RES SHR S CPU% MEM% TIME+ Command
24175 ha62gad 20 0 6354M 2220M 70908 S 101. 0.9 176h /opt/matlab12a/bin/glnxa64/MATLAB -nodisplay -r run
26059 neb 39 19 15816 8152 1868 R 100. 0.0 5h01:27 dnetc -n -1
42787 to78men 20 0 16.0G 5548M 71092 S 100. 2.1 23h35:27 /opt/matlab12a/bin/glnxa64/MATLAB -nodisplay -r mainCEPscan34
21110 neb 39 19 15816 8148 1868 R 100. 0.0 6h07:19 dnetc -n -1
43251 to78men 20 0 14.5G 3353M 71096 S 100. 1.3 23h35:09 /opt/matlab12a/bin/glnxa64/MATLAB -nodisplay -r mainCEPscan78
8095 neb 39 19 15816 8148 1868 R 100. 0.0 30:20:09 dnetc -n -1
40647 neb 39 19 15816 8152 1868 R 100. 0.0 2h19:37 dnetc -n -1
43783 to78men 20 0 12.6G 2303M 71092 S 100. 0.9 23h32:46 /opt/matlab12a/bin/glnxa64/MATLAB -nodisplay -r mainCEPscan56
44650 neb 39 19 15816 8148 1868 R 100. 0.0 1h33:53 dnetc -n -1
43253 to78men 20 0 17.0G 5809M 71088 S 100. 2.2 23h34:52 /opt/matlab12a/bin/glnxa64/MATLAB -nodisplay -r mainCEPscan910
7682 neb 39 19 15816 8152 1868 R 100. 0.0 33:57:49 dnetc -n -1
42712 neb 39 19 15816 8152 1868 R 100. 0.0 1h51:09 dnetc -n -1
46816 neb 39 19 15816 8152 1868 R 100. 0.0 1h03:36 dnetc -n -1
12766 neb 39 19 15816 8152 1868 R 100. 0.0 8h13:10 dnetc -n -1
7956 neb 39 19 15816 8152 1868 R 100. 0.0 32:58:00 dnetc -n -1
43377 to78men 20 0 14.5G 3353M 71096 R 100. 1.3 23h32:50 /opt/matlab12a/bin/glnxa64/MATLAB -nodisplay -r mainCEPscan78
42910 to78men 20 0 16.0G 5548M 71092 R 100. 2.1 23h33:22 /opt/matlab12a/bin/glnxa64/MATLAB -nodisplay -r mainCEPscan34
45076 neb 39 19 15816 8148 1868 R 100. 0.0 1h27:13 dnetc -n -1
33588 neb 39 19 15816 8148 1868 R 100. 0.0 3h38:03 dnetc -n -1
45340 neb 39 19 15816 8148 1868 R 100. 0.0 1h20:58 dnetc -n -1
41565 mi28tup 20 0 12.7G 12.2G 13008 R 100. 4.8 112h /opt/Mathematica/8.0/SystemFiles/Kernel/Binaries/Linux-x86-64/MathK
42321 neb 39 19 15816 8148 1868 R 100. 0.0 1h56:46 dnetc -n -1
41370 neb 39 19 15816 8152 1868 R 100. 0.0 2h14:06 dnetc -n -1
43843 to78men 20 0 12.6G 2303M 71092 R 100. 0.9 23h30:39 /opt/matlab12a/bin/glnxa64/MATLAB -nodisplay -r mainCEPscan56
43372 to78men 20 0 17.0G 5809M 71088 R 100. 2.2 23h32:52 /opt/matlab12a/bin/glnxa64/MATLAB -nodisplay -r mainCEPscan910
42905 to78men 20 0 12.3G 2234M 71100 R 100. 0.9 23h33:31 /opt/matlab12a/bin/glnxa64/MATLAB -nodisplay -r mainCEPscan12
F1Help F2Setup F3Search F4Filter F5Tree F6SortBy F7Nice F8Nice F9Kill F10Quit
```

Свободное место на диске можно узнать командой `df -h`

Установка python

- Мы будем использовать python 3.8.5
- Нам потребуется устанавливать и иметь дело с другими версиями python (например, 3.8.7 на heroku)
- Нам потребуется работать с виртуальными окружениями. Мы будем делать это с помощью `pipenv` в рамках разрабатываемых нами пакетов.
- Мы будем делать это как на вашей системе, так и на серверах и в docker образах

Для этого нам нужно управлять **несколькими** версиями питона и **разными** виртуальными окружениями.

Anaconda



Готовые дистрибутивы с python. Внутри: предустановленные библиотеки, ipython, jupyter notebook, etc

- Плюсы: легко установить, даже на windows
- Минусы: не подходит для использования в production

<https://www.anaconda.com/products/individual>

Miniconda

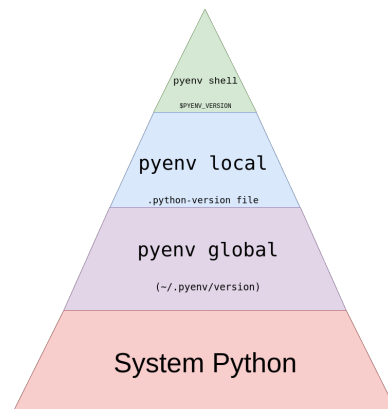
```
1 curl -O https://repo.anaconda.com/miniconda/Miniconda3-latest-Linux-x86_64.sh
2 chmod +x Miniconda3-latest-Linux-x86_64.sh
3 ./Miniconda3-latest-Linux-x86_64.sh -b
4
5 # создадим виртуальное окружение для нужной версии
6 echo 'PATH="$HOME/miniconda3/bin:$PATH"' >> ~/.bashrc
7 source ~/.bashrc
8 conda create -n py38 python==3.8
9 source activate py38
```

- Плюсы: позволяет более гибкую настройку окружений
- Минусы: поддерживаются не все версии python

pyenv

```
1 curl https://pyenv.run | bash
2 export PATH="/root/.pyenv/bin:$PATH"
3 eval "$(pyenv init -)"
4 eval "$(pyenv virtualenv-init -)"
5 mkdir -p $(pyenv root)/cache
6 pyenv install 3.8.5
7 # pyenv global 3.8.5
8 # pyenv local 3.8.5
9 # pyenv shell 3.8.5
10 # виртуальное окружение будем создавать с помощью pipenv
```

- Плюсы: наиболее гибок
- Минусы: проблемы при работе на windows (форк pyenv-win)



Интерактивные среды выполнения python

- `python` - непосредственно интерпретатор языка, всегда в комплекте
- `ipython` - "interactive python" - немного удобнее, но нужно устанавливать отдельно `pip install ipython`

Пригождаются, когда мы хотим проверить окружение или отдебажить работу нашего скрипта в нем.




Типичные сетапы рабочей среды для Python

- Локальный
- Удаленный через ssh -L
- Удаленный через VSCode/Pycharm

Локальный

- Ставим `pyenv/Miniconda/Anaconda`
- Заводим общее окружение, создаем для него `kernel`. Можем использовать его там, где не требуется воспроизводимость и точный список библиотек.
- В других местах используем виртуальные окружения и инструменты для управления ими, например, `pipenv`. Виртуальными окружениями можно пользоваться из IDE (`VSCode`, `PyCharm`).

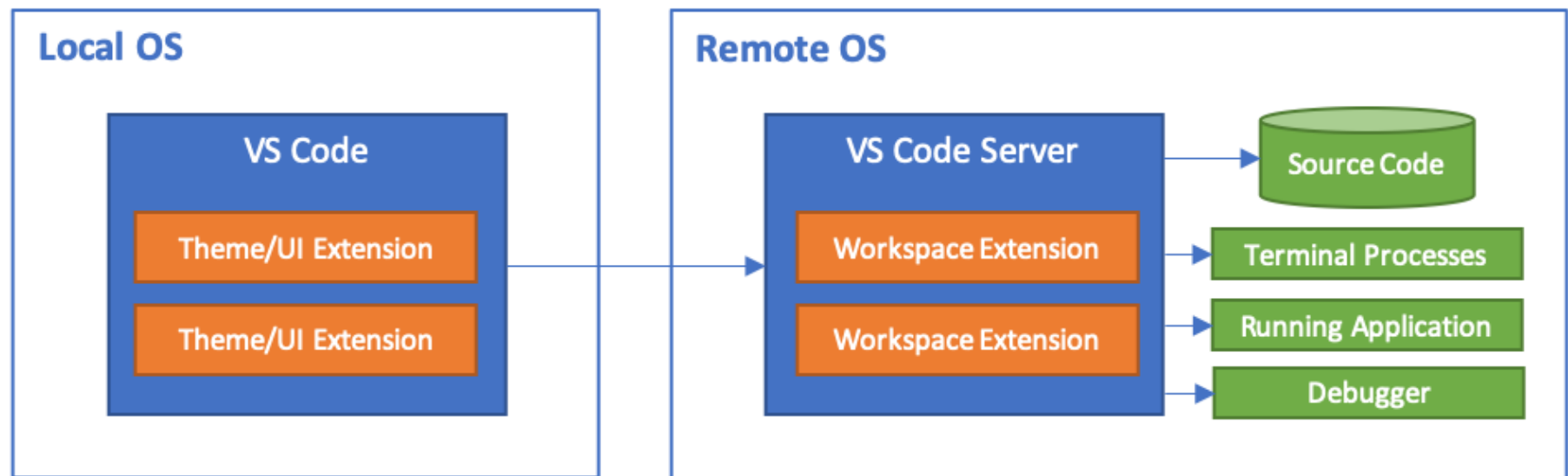
Удаленный через ssh -L

- Заходим по ssh -L \$LOCAL_PORT:...
-  Ставим pyenv/Miniconda
-  Заводим общее окружение, создаем для него kernel
-  Заходим в tmux, поднимаем там jupyter notebook/lab на \$LOCAL_PORT
- Открываем в браузере localhost:\$LOCAL_PORT
- Держим ssh-соединение открытым

*Этапы с  происходят на сервере

Удаленный через IDE

- Заходим по ssh
- ➡ Ставим pyenv/Miniconda
- ➡ Заводим общее окружение
- Настраиваем соединение в IDE
- Запускаем код в IDE на выполнение на удаленной машинке



<https://code.visualstudio.com/docs/remote/remote-overview>

Саммари

1. Для начала установите Python на локальном компьютере, поставьте себе IDE, запустите Jupyter сервер
2. Когда начнете работать с удаленным Linux-сервером, вернитесь к слайдам этой лекции, чтобы освежить в памяти нужные вещи
3. Если большинство инструментов вам не знакомы, имеет смысл запомнить как они называются и зачем нужны, а потом – изучить, когда они потребуются (это произойдет очень скоро :)

Семинар:

1. Генерируем ключи для ssh, закидываем их на сервер
2. Заходим по ssh на сервер
3. обновляем пакеты apt-get
4. Ставим git, nano, vim
5. Ставим pyenv и нужную версию питона
6. Ставим jupyter, добавляем kernel, запускаем jupyter
7. по ssh -L пробрасываем порт и открываем jupyter локально
8. Открываем VSCode и настраиваем коннекшн, выполняем bash и python скрипт на удаленной машинке (uname -a, например)