**Руководство пользователя сервера моделирования bipython2**

Оглавление

[1. Введение 3](#_Toc3793567)

[2. Ссылки 3](#_Toc3793568)

[3. Глоссарий 4](#_Toc3793569)

[4. Получение доступа к системе 4](#_Toc3793570)

[5. Руководство пользователя Jupyter Notebook 4](#_Toc3793571)

[5.1. Введение 4](#_Toc3793572)

[5.2. Начало работы 5](#_Toc3793573)

[5.3. Основная функциональность 6](#_Toc3793574)

[5.4. Перезагрузка пользовательского сервера jupyerhub 7](#_Toc3793575)

[6. Руководство пользователя докерезированного Jupyter Notebook 7](#_Toc3793576)

[6.1. Введение 7](#_Toc3793577)

[6.2. Начало работы 8](#_Toc3793578)

[6.3. Основная функциональность 8](#_Toc3793579)

[7. Использования Docker CLI 9](#_Toc3793580)

[7.1. Введение 9](#_Toc3793581)

[7.2. Матрица доступа к командам 9](#_Toc3793582)

[7.3. Порядок работы с Python API для docker 11](#_Toc3793583)

[7.4. Доступ к внутренним ресурсам банка (docker registry, PyPi) 13](#_Toc3793584)

**История внесения изменений в документ**

| **Дата** | **Версия** | **Изменения** | **Автор** |
| --- | --- | --- | --- |
| 12.09.2018 | 0.1 | Создание документа | Кабанов В.В. |
| 10.12.2018 | 0.2 | Доработаны разделы 3, 6 и 7.2, 7.3 | Трофимов А.В. |
| 18.03.2019 | 0.3 | Добавлен пункт 5.4 | Трофимов А.В. |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

# Введение

Настоящий документ является руководством пользователя сервера моделирования bipython2.

Документ описывает использование приложения Jupyter Notebbok, в том числе в docker контейнере. Документ также описывает использование docker команд через командную строку.

Целью документа является описание в полном объеме особенностей использования функционала, связанных с конкретными настройками ПО на сервере bipython2 и описание возможностей функционала в объеме достаточном для того, чтобы понять, годится ли он для решения того или иного класса задач. Целью документа не является описание возможностей, настроек и особенностей работы Jupyter, docker или другого ПО, развернутого на среде моделирования – для этих целей даются ссылки на официальную документацию ПО в соответствующих местах.

Стороны могут изменять и дополнять настоящий документ. Документ подлежит версионному контролю; выпуск новой версии документа отменяет действие всех предыдущих версий. Все изменения и дополнения описываются в Листе изменений в начале документа.

# Ссылки

|  |  |  |
| --- | --- | --- |
| **№ п/п** | **Ссылка** | **Описание** |
| 1 | [Admin Guide среды моделирования](http://confluence.moscow.alfaintra.net/display/ORBARB/AG+v0.2) | Руководство администратора сервера bipython2 |
| **2** | [Инструкция по настройке среды моделирования](http://confluence.moscow.alfaintra.net/pages/viewpage.action?pageId=202411482) | Инструкция по настройке севера bipython2 |
| **3** |  |  |
| **4** |  |  |
| **5** |  |  |

# Глоссарий

|  |  |  |
| --- | --- | --- |
| Понятие | Сокращение | Определение понятия |
| Руководство пользователя | РП | Данный документ |
| Сервер моделирования |  | Сервер, на котором осуществляется работа команд моделирования. Предполагается, что сервер заведен в кластере Хадуп, как edge node. Данное РП писалось по итогам настройки функционала на сервере bipython2. |
| Кластер, Hadoop |  | Кластер Hadoop, на котором выполняются Spark, Hive и Impala задания из ноутбуков jupyter. Сервер моделирования заведен в нем, как edge node. На момент написания РП, функционал был настроен на кластере BDA31. |
| Jupyterhub |  | Сервер для многопользовательского использования Jupyter Notebook. |
| Jupyter Notebook, ноутбук. |  | Приложение для интерактивной разработки. В JupyterHub для пользователя запускается личный Jupyter Notebook. |
| Docker, докер |  | Движок для контейнеризации приложений. |

# Получение доступа к системе

# Руководство пользователя Jupyter Notebook

## Введение

Jupyter Notebook является средой интерактивного программирования. Основной сценарий использования – протопирование и эксперементирование. Jupyter Notebook не привязан к конкретному языку программирования, но поддерживает разные ядра или интепретаторы.

Работа с Jupyter Notebook на сервере моделирования осуществляется через JupyterHub. JupyterHub является серверным приложением, обеспечивающим многопользовательскую работу с Jupyter Notebook. Пользователь осуществляет логин в JupyterHub, JupyterHub создает для этого пользователя Jupyter Notebook и перенаправляет пользователя туда. Дальнейшая работа осуществляется в Jupyter Notebook.

На данный момент на bipython2 установлены следующие ядра для Jupyter Notebook:

|  |  |  |
| --- | --- | --- |
| Ядро | Описание | Комментарий |
| Python 3 | Интерпретатор Python3.6. | Список библиотек см. в документе 2. |
| Python3.6 + PySpark(Spark 2.2.0) 16GB | Интерпретатор PySpark. Автоматически созадется Spark session с подключением к кластеру BDA31. | Сессия Spark создается с параметарами 16GB driver memory, 4GB executor memory |
| Python3.6 + PySpark(Spark 2.2.0) 64GB | Интерпретатор PySpark. Автоматически созадется Spark session с подключением к кластеру BDA31. | 64GB driver memory, 8GB executor memory |
| Python3.6 + PySpark(Spark 2.2.0) 128GB | Интерпретатор PySpark. Автоматически созадется Spark session с подключением к кластеру BDA31.. | 128GB driver memory, 16GB executor memory |
| Spark -Scala | Интерпретатор Spark Scala. Автоматически созадется Spark session с подключением к кластеру BDA31. | Используется интерпретатор toree <https://github.com/apache/incubator-toree> |

Полный список доступных ядер: [https://github.com/jupyter/jupyter/wiki/Jupyter-kernels](https://github.com/jupyter/jupyter/wiki/Jupyter-kernels%20). Для установки дополнительных ядер обращаться к 3 линии поддержки (контактное лицо – Владимир Кабанов). Документация Jupyter <http://jupyter.org/documentation>. Также для Jupyter установлен пакет jupyter\_contrib\_nbextensions с пользовательскими плагинами для удобства разработки <https://github.com/ipython-contrib/jupyter_contrib_nbextensions>.

## Начало работы

После получения доступа к системе (см. параграф 4), приложение JupyterHub доступно по адресу <http://bipython2:8000>. Аутентификация осуществляется по учетной записи на сервере.

Важно: для работы с приложением рекомендуется использовать браузер Google Chrome и не рекомендуется использовать Internet Explorer. Другие браузеры (Yandex Browser, Mozilla) на момент написание документа не тестировались. Приложение не поддерживает одновременную работу под несколькими учетными записями с одного локального компьютера.

При успешной аутентификации в JupyterHub пользователь будет перенаправлен в созданный для него Jupyter Notebook и дальнейшая работа будет проходить в Jupyter Notebook.

## Основная функциональность

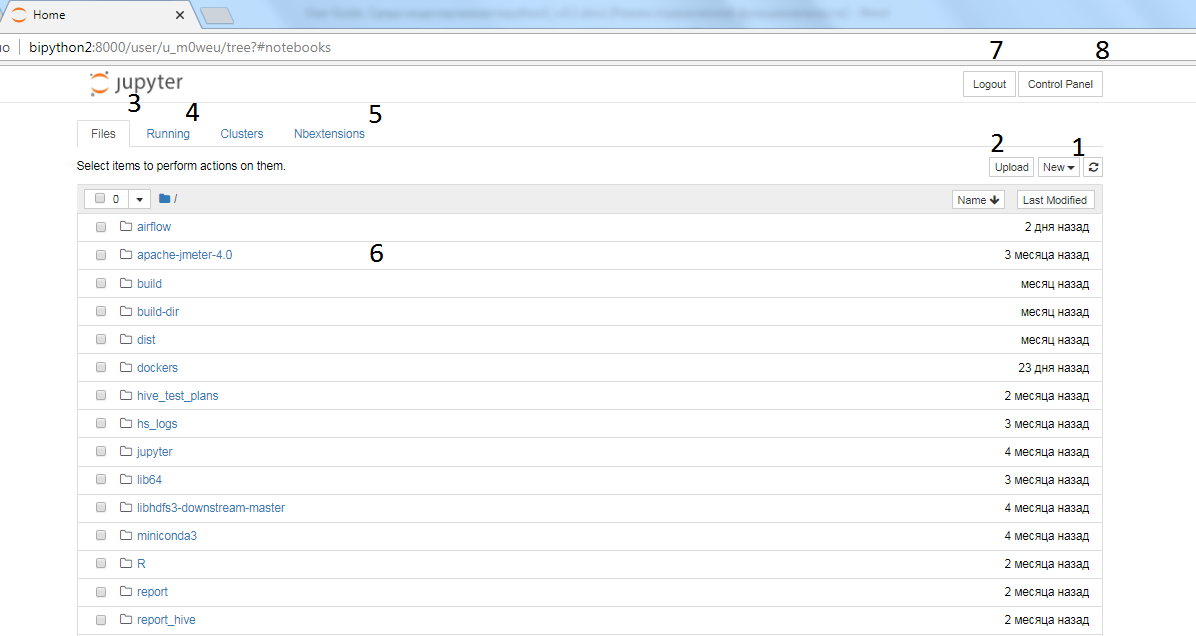


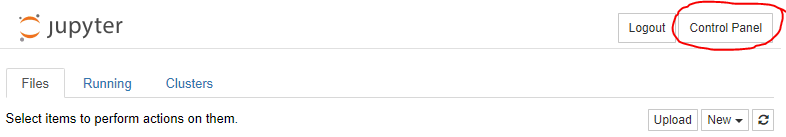
Рис.1 интерфейс Jupyter Notebook

Компоненты интерфейса

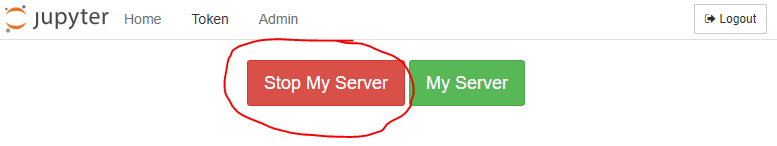
1. New – создание нового ноутбука. В выпадающем списке можно выбрать ядро или открыть терминал (командную строку).
2. Upload – загрузка файлов в рабочий каталог.
3. Вкладка Files – открывает рабочий каталог.
4. Вкладка Running – активные приложения (ноутбуки, терминалы) пользователя. **Не рекомендуется открывать большое количество Spark ноутбуков, т.к. даже в пассивном режиме они утилизируют ресурсы.**
5. Вкладка Nbextensions – управление расширениями (см. [параграф 5.2](#_Начало_работы)).
6. Рабочий каталог. Рабочим каталогом является директория /home/<username> в файловой системе сервера. Не рекомендуется хранить в ней большие файлы (см. [параграф 6.1](#_Введение)).
7. Logout – выход из приложения. При это Jupyter Notebook продолжает работать.
8. Control panel – остановка Jupyter Notebook пользователя. При повторном логине Jupyter Notebook будет снова запущен.

## Перезагрузка пользовательского сервера jupyerhub

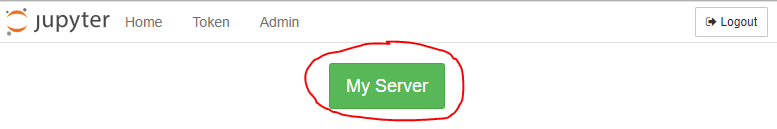
1. Зайти в Control Panel (правый верхний угол)



1. Остановить сервер (Красная кнопка Stop My Server)



1. Запустить сервер (зеленая кнопка My Server)



# Руководство пользователя докерезированного Jupyter Notebook

## Введение

Помимо основного JupyterHub, запускающего ноутбуки на «голом железе», на сервере работают хабы, запускающие ноутбуки в docker контейнерах. Docker в контексте Jupyter используется для управления (разграничения) вычислительных ресурсов сервера и для версионирования сред разработки, где под средой понимается конкретный набор пакетов Python, библиотек C, C++, Fortran и так далее.

Для пользователей работы в докерезированном ноутбуке в целом не отличается от работы на голом железе, за исключением некоторых нюансов. Докерезированным ноутбуком следует пользоваться в следующих случаях:

* Вам нужно версионировать среды разработки, например для того, чтобы запаковывать зависимости модели в docker контейнер (см. [параграф 7](#_Использования_Docker_CLI))
* Вам нужна библиотека, которую невозможно установить на голое железо
* В целях разграничения ресурсов сервера, группа поддержки инфраструктуры дала вам указание использовать докерезированный хаб

В остальных случаях, вам проще будет использовать обычный хаб.

Чтобы избежать дублирование, в данном параграфе не будут повторяться базовые указание по работе с Jupyter Notebook – будут описываться только дополнительные особенности в связи с докером. Предполагается, что перед чтением параграфа 6 пользователь ознакомился с параграфом 5.

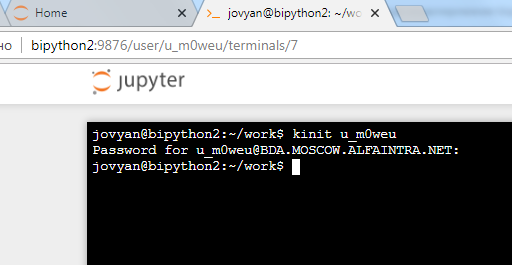
## Начало работы

Логин осуществляется также, как и для обычного хаба (см. [параграф 5](#_Начало_работы)). Приложения JupyterHub доступны по адресам:

* <http://bipython2:9876> – среда datamonetization отдела монетизации данных
* <http://bipython2:9877> – среда aim отдела аналитики и моделирования МСКБ
* <http://bipython2:9878> – среда aim2 отдела аналитики и моделирования МСКБ
* <http://bipython2:9879> – среда root отдела аналитики и моделирования МСКБ
* http://bipython2:9880 – среда sup отдела аналитики и моделирования МСКБ

**Перед** запуском ядра, осуществляющего подключение к кластеру, нужно создать керберос тикет. Для этого откройте терминал и выполните команду

kinit <username>



## Основная функциональность

Основное отличие в работе с докерезированным Jupyter, кроме необходимости делать kinit, состоит в наличии нужной вам версии среды разработки библиотеки или наличии нужной вам библиотеки, которую невозможно установить на голое железо. Вместо директории в /home/<username> для пользователей создается директория в /home/<username>/work. Файлы в ней принадлежат техническому пользователю контейнера, jovyan. Для работы с рабочей директорией (загрузка и скачивание файлов) рекомендуется использовать веб-интерфейс Jupyter Notebook.

Также можно использовать команду docker для просмотра логов контейнера. Для этого

Найдите ваш контейнер через команду

sudo docker ps

он будет называться jupyter[-<название среды>]-<username>. Дальше можно посмотреть логи контейнера командой

sudo docker logs <container name>

Подробнее об использовании docker cli см. [параграф 7](#_Использования_Docker_CLI).

# Использования Docker CLI

## Введение

Docker является средством контейнеризации. Подробнее см. официальную страницу проекта <https://docs.docker.com/>.

На сервере моделирования пользователям предоставляется доступ на запуск docker контейнеров через интерфейс командной строки. Доступ к командам предоставляется ограниченный, с учетом требований безопасности и стабильности работы сервера.

Целевое использование docker cli на сервере моделирование – регламентный запуск «запакованных» Python скриптов, например, для применения моделей. **Другие сценарии использования docker необходимо согласовывать с группой поддержки инфраструктуры (руководитель – Алексей Щербаков), даже если выдача дополнительных прав на docker cli для этого не требуется.**

## Матрица доступа к командам

Назначение команд см. <https://docs.docker.com/engine/reference/commandline/docker/> Предоставление доступа описано ниже:

|  |  |
| --- | --- |
| Команда | Вид доступа |
| docker diff | sudo |
| docker events | sudo |
| docker history | sudo |
| docker images | sudo |
| docker info | sudo |
| docker inspect | sudo |
| docker logs | sudo |
| docker port | sudo |
| docker ps | sudo |
| docker search | sudo |
| docker stats | sudo |
| docker top | sudo |
| docker version | sudo |
| docker build | sudo |
| docker commit | sudo |
| docker export | sudo |
| docker import | sudo |
| docker load | sudo |
| docker login | sudo |
| docker logout | sudo |
| docker pull | sudo |
| docker push | sudo |
| docker save | sudo |
| docker rmi | sudo |
| docker tag | sudo |
| docker exec | sudo |
| docker run | Python API |
| docker start | Python API |
| docker stop | Python API |
| docker rm | Python API |

О порядке работы с Python API см. [параграф 7.3](#_Порядок_работы_с). О порядке работы с docker registry (команды docker login, docker logout, docker push, docker pull) см. [параграф 7.4](#_Доступ_к_внутренним).

При сборке нового докер образа за основу рекомендуется брать один из образов для запуска докерезированных ноутбуков (см. [параграф 6](#_Руководство_пользователя_докерезиро)). Такая рекомендация вызвана тем, что, во-первых, если ваша задача заключается в запаковывании зависимостей вашего приложения, вы таким образом возьмете за основу среду, в которой вы приложение разрабатывали (а именно, докерезированный Jupyter Notebook), во-вторых, в образах для ноутбуков уже есть необходимые настройки для подключения к кластеру. Список доступных базовых образов:

|  |  |  |
| --- | --- | --- |
| Репозиторий | Описание | Комментарий |
| datamonetization/notebook | Jupyter Notebook для команды монетизации данных. | В JupyterHub используется тэг base. |
| mskb/aim-notebook | Jupyter Notebook со средой aim для команды аналитики и моделирования МСКБ | В JupyterHub используется тэг base. |
| mskb/aim2-notebook | Jupyter Notebook со средой aim2 для команды аналитики и моделирования МСКБ | В JupyterHub используется тэг base. |
| mskb/root-notebook | Jupyter Notebook со средой root для команды аналитики и моделирования МСКБ | В JupyterHub используется тэг base. |
| mskb/sup-notebook | Jupyter Notebook со средой sup для команды аналитики и моделирования МСКБ | В JupyterHub используется тэг base. |

В частности, из сказанного выше следует, что если вы хотите запаковывать зависимости ваших приложений в докер контейнеры, то разрабатывать приложения вам следует в докерезированном Jupyter Notebook или сразу в docker контейнерах.

Особо внимательно нужно относится к командам docker tag и docker build – это связано с тем, что JupyterHub находит нужный докер образ *по репозиторию и тегу*. Таким образом, если вы случайно протегаете ваш образ репозиторием и тегом, используемом в JupyterHub, новые ноутбуки соотв. хаба будут запускаться в вашем образе, что может привести к нарушению работы.

При наличии обоснованной проектной необходимости допустимо расширение привилегий на команды docker. По этому вопросу обращаться к группе поддержки инфраструктуры (руководитель – Алексей Щербаков).

## Порядок работы с Python API для docker

Как объясняется в [параграфе 7.2](#_Матрица_доступа_к), к ряду команд docker доступ осуществляется через Python API.

Через скрипты предоставляется доступ к следующим командам:

|  |  |  |
| --- | --- | --- |
| Скрипт | Команда docker | Комментарий |
| /opt/docker/docker\_run.py | docker run | Выполняется проверка на монтируемые каталоги. |
| /opt/docker/docker\_start.py | docker start | Выполняется проверка на то, что контейнер принадлежит пользователю. |
| /opt/docker/docker\_stop.py | docker stop | Выполняется проверка на то, что контейнер принадлежит пользователю. |
| /opt/docker/docker\_rm.py | docker rm | Выполняется проверка на то, что контейнер принадлежит пользователю. |

В данных скриптах не накладывается никакая дополнительная докер-специфичная логика, например, запуск определенного ядра в docker\_run.py. Все, что делает скрипт – *проверяет*, что у вас есть права *на запуск команды*, после чего ее запускает. При этом, если продолжить пример с docker\_run.py, в самом контейнере может запускаться что угодно – Spark, Python, C++ – это зависит только от пользователя.

Для каждого скрипт есть help, который описывает доступные опции (они повторяют смысл опций соответствующих docker команд).

Пример вызова help:

sudo /usr/local/bin/python3.6 /opt/docker/docker\_run.py --help

usage: docker\_run.py [-h] [-t] [-i] [-d] [--volumes\_from VOLUMES\_FROM] [-v V]

[-name NAME]

image [command [command ...]]

positional arguments:

image the image to run

command the command to run in the container

optional arguments:

-h, --help show this help message and exit

-t allocate a pseudo-tty

-i keep STDIN open

-d run a container in the background

--volumes\_from VOLUMES\_FROM

list of container names of IDs to get volumes from

-v V volumes to mount inside the container

-name NAME name for this container

Пример выполнения команды docker run:

sudo /usr/local/bin/python3.6 /opt/docker/docker\_run.py -d --volumes\_from configContainer -v /etc/krb5.conf:/etc/krb5.conf:ro -v /home/u\_m0weu/work:/home/u\_m0weu/work:rw –name myContainer datamonetization/notebook:base

Ключевые моменты:

* Для корректной работы sudo рекомендуется писать абсолютный путь к исполняемым файлам, т.е. **/usr/local/bin/python3.6 /opt/docker/<скрипт с командой докер>**
* Для доступа к кластеру необходимо подмонтировать директории из контейнера с конфигом и конфиг керберос, т.е. **–volumes\_from configContainer –v /etc/krb5.conf:/etc/krb5.conf:ro**
* Дополнительно можно подмонтировать пользовательский каталог из /home/<username>, т.е. **–v /home/<username>/work:<целевая директория в контейнере>:rw**
* Другие точки монтирования не разрешены
* Рекомендуется присваивать имя контейнеру, т.е. **–name myContainer**

На данный момент ограничением Python API является невозможность интерактивной работы с контейнером (флаги –i, -t). Для интерактивной работы рекомендуется использовать докерезированный Jupyter Notebook.

Для доступа к кластеру перед запуском любого функционала в контейнере, в частности, перед созданием Spark сессии, нужно сделать kinit. Для выполнения kinit из Python рекомендуется использовать код:

import os  
os.system('/bin/echo -n "XXXXXX " | /usr/bin/kinit u\_m0weu')

где XXXXXX нужно заменить на ваш пароль, u\_m0weu – на ваш логин. Также можно выполнять шелл скрипт перед запуском докер команды:

sudo /usr/local/bin/python3.6 /opt/docker/docker\_run.py -d --volumes\_from configContainer -v /etc/krb5.conf:/etc/krb5.conf:ro -v /home/u\_m0weu/work:/home/u\_m0weu/work:rw –name myContainer datamonetization/notebook:base **/bin/echo -n "XXXXXX " | /usr/bin/kinit u\_m0weu' && /usr/bin/Python3.6 /home/jovyan/script.py**

При запуске контейнера, к его имени автоматически конкатенируется логин запускающего пользователя. В дальнейшем этот признак используется для определения принадлежности контейнера, в частности, при запуске команд docker\_start.py, docker\_stop.py, docker\_rm.py.

## Доступ к внутренним ресурсам банка (docker registry, PyPi)

На сервере моделирования настроен доступ к репозиторию PyPi и к docker registry, развенутых на банковском Artifactory.

Также доступ к PyPi настроен в базовых докер образах (см. [параграф 7.2](#_Матрица_доступа_к)), то есть при создании образов вы можете запускать установку дополнительных пакетов через pip (это целевое использование pip, т.к. установкой библиотек на bare metal занимается группа сопровждения инфраструктуры).

Доступ к pip осуществляется стандартно, то есть

pip3 install django

Доступ к docker registry осуществляется командой

sudo docker login datalake.binary.alfabank.ru

**Важно:** для логина в docker registry используется **доменная УЗ** (пароль может отличаться от учетной записи на сервере моделирования).

После этого вы в docker registry, можно выполнять push и pull.