|  |
| --- |
| ОАО "АЛЬФА-БАНК"РУКОВОДСТВО **Администратора**  **приложения JupyterHub на среде моделирования**  Версия 0.3  **Москва**  **2018** |

Оглавление

[История изменений 3](#_Toc523446319)

[1. Глоссарий 4](#_Toc523446320)

[2. Ссылки 4](#_Toc523446321)

[3. Введение 4](#_Toc523446322)

[4. Требования к программному и аппаратному обеспечению 5](#_Toc523446323)

[5. Установка приложения 5](#_Toc523446324)

[6. Конфигурация JupyterHub 6](#_Toc523446325)

[7. Административные действия с JupyterHub 8](#_Toc523446326)

[8. Docker контейнеры, используемые в приложени 9](#_Toc523446327)

[9. Административные действия с docker контейнерами 10](#_Toc523446328)

[10. Часто встречающиеся ошибки 12](#_Toc523446329)

[11. Восстановление после сбоя 12](#_Toc523446330)

# История изменений

|  |  |  |  |
| --- | --- | --- | --- |
| **Дата** | **Номер версии** | **ФИО вносившего корректировки** | **Описание корректировок** |
| 28.08.2018 | 0.1 | Кабанов В.В. | Документ создан |
| 31.08.2018 | 0.2 | Кабанов В.В. | Доработан раздел 8. Добавлены разделы 11, 12. |
| 12.10.2018 | 0.3 | Трофимов А.В. | Доработаны разделы 6, 7 и 9. |

1. Глоссарий

|  |  |
| --- | --- |
| Термин | Описание |
| Банк | ОАО «Альфа-Банк» |
| Сервер моделирования | Сервер, на котором осуществляется работа команд моделирования. Предполагается, что сервер заведен в кластере Хадуп, как edge node. Данный Admin Guide писался 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 Admin Guide, 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. |
| Hub, хаб | Компонента JupyterHub, которая осуществляет аутентификацию пользователей, сохранение информации о работе пользователей и запуск ноутбуков. |
| Proxy, прокси | Компонента JupyterHub, которая отправляет запросы пользователей хабу и ноутбукам. |
| Spawner | Компонента, которая запускат пользовательский ноутбук. |

1. Ссылки

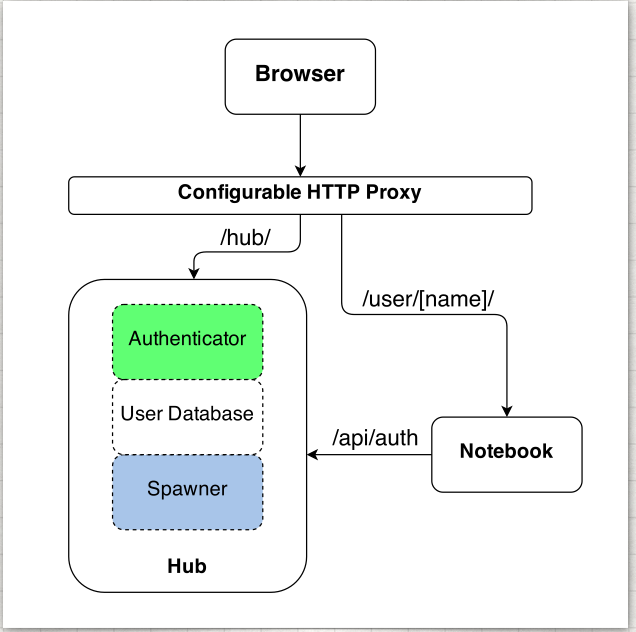
|  |  |
| --- | --- |
| Ссылка | Описание |
| [Инструкция по настройке среды моделирования](http://confluence.moscow.alfaintra.net/pages/viewpage.action?pageId=202411482) | В документе описана установка необходимых Python пакетов. |
| [Инструкция по настройке кластера BDA](http://confluence.moscow.alfaintra.net/pages/viewpage.action?pageId=202411467) | В документе описано заведение сервера моделирование в качестве edge node кластера Hadoop. |

1. Введение

В документе описана настройка, запуск и администрирование приложения JupyterHub. Описана специфика связи с кластером Hadoop. Описана настройка JupyterHub для запуска ноутбуков в docker контейнерах. Описаны наиболее часто встречающиеся ошибки пользователей и способы их решения.

Для общего понимания функционала полезно ознакомиться с титульной страницей документации <https://jupyterhub.readthedocs.io/en/stable/index.html>.

Взаимодействие компонент jupyterhub



1. Требования к программному и аппаратному обеспечению
2. Установка приложения

Установка приложения JupyterHub в объеме функциональности, описанном в текущем

документе, состоит из следующих пунктов

* Установка пакетов Python, необходимых для запуска JupyterHub – описано в документе «Инструкция по настройке среды моделирования»
* Заведения сервера моделирования, как edge node в кластере BDA. Это нужно для возможности запускать задания на Spark. Описано в документе «Инструкция по настройке кластера BDA»
* Для использования docker – установка docker engine

Установка docker engine выполняется командой Linux

yum install docker-engine

После установки необходимо выполнить рестарт сервера. Проверка установки выполняется командой

docker –version

1. Конфигурация JupyterHub

Конфигурация JupyterHub осуществляется через файл конфига, jupyterhub\_config.py. В данном документе описывается конфигурация, которая была разработа в ходе внедрения на bipython2. Для изучения всех возможностей, нужно обратиться к [официальной документации](https://jupyterhub.readthedocs.io/en/stable/getting-started/config-basics.html).

По умолчанию, JupyterHub использует конфиг, находящийся в том каталоге ОС, где была выполнена команда jupyterhub. Таким образом, параллельный запуск нескольких инстансов JupyterHub (например, для разных команд из разных отделов), предлагается такая организация на уровне ОС:

* /opt/jupyterhub
  + /jhub-datamonet
    - jupyterhub\_config.py –конфиг для отдела монетизации данных
  + /jhub-mskb-aim
    - jupyterhub\_config.py –конфиг со средой aim для отдела аналитики и моделирования МСКБ
  + /jhub-mskb-aim2
    - jupyterhub\_config.py –конфиг со средой aim2 для отдела аналитики и моделирования МСКБ
  + /jhub-mskb-root
    - jupyterhub\_config.py –конфиг со средой root для отдела аналитики и моделирования МСКБ
  + /jhub-mskb-sup
    - jupyterhub\_config.py –конфиг со средой sup для отдела аналитики и моделирования МСКБ

Тогда для запуска JupyterHub для каждого из инстансов нужно будет выполнить команду jupyterhub из соответствующего каталога. Подробней о запуске JupyterHub см. параграф 7.

Используемый конфиг:



Описание базовых настроек конфига:

|  |  |  |
| --- | --- | --- |
| Название параметра | Значение | Комментарий |
| c.JupyterHub.port | 9876 | Порт, на котором доступен jupyterhub для пользователей. Нужно менять при добавлении нового инстанса. |
| c.ConfigurableHTTPProxy.api\_url | 'http://bipython2:9001' | URL, по которому хаб будет взаимодействовать с прокси. Нужно менять при добавлении нового инстанса. |
| c.JupyterHub.hub\_port | 9080 | Порт, на котором хаб будет слушать внешние сервисы. Нужно менять при добавлении нового инстанса. |
| c.JupyterHub.debug\_proxy | true | Вывод сообщений отладки прокси в лог JupyterHub. |

Предполагается, что администратору JupyterHub придется изменять только 4 настройки выше. Для полноты ниже приводится описание остальных настроек.

Описание дополнительных настроек, которые нужно сделать для работы ноутбуков в контейнерах:

|  |  |  |
| --- | --- | --- |
| import dockerspawner, os |  | Импорт пакетов, нужных для настройки докера. |
| c.DockerSpawner.container\_prefix | ('jupyter-aim') | Изменение стандартного префикса в именовании контейнера (добавление наименования среды). |
| c.JupyterHub.hub\_ip | ‘0.0.0.0’ | IP, на котором хаб будет слушать другие компоненты JupyterHub. Выставляется в значение ‘0.0.0.0’ (слушать на всех интерфейсах). |
| c.JupyterHub.spawner\_class | 'dockerspawner.DockerSpawner' | Класс Python, который выполняет запуск ноутбуков. Класс dockerspawner.DockerSpawner выполняет запуск ноутбуков в контейнерах. |
| c.DockerSpawner.image | 'datamonetization:base-notebook' | Докер образ, в котором запускаются ноутбуки. Может меняться администратором по требованию пользователей JupyterHub. |
| c.DockerSpawner.hub\_ip\_connect | '172.17.0.1' | IP, по которому котейнеризированные ноутбуки будут искать хаб. |
| c.DockerSpawner.use\_internal\_ip | ‘false’ |  |
| c.DockerSpawner.network\_name | ‘host’ | Сеть, в которой запускаются докер контейнеры. Значение ‘host’ означает запуск без сетевой изоляции. |
| c.DockerSpawner.extra\_host\_config | { 'network\_mode': ‘host’} |  |
| c.DockerSpawner.notebook\_dir | ‘/home/jovyan/work’ | Задание рабочей директории внутри контейнера. |
| c.DockerSpawner.volumes | {'/app/share/{username}': {'bind': notebook\_dir, 'mode': 'rw'}, '/etc/krb5.conf': {'bind': '/etc/krb5.conf', 'mode': 'ro'}} | Задание монтирования пользовательской директории из /app/share и керберос конфига krb5.conf. |
| c.DockerSpawner.extra\_host\_config.update({ 'volumes\_from': 'configContainer' }) |  | Монтирование конфига Hadoop из контейнера. Подробнее о конфигурационном контейнере см. параграф 8. |
| c.DockerSpawner.remove\_containers | true | Контейнеры запускаются с опцией «удалить после остановки контейнера». |
| c.DockerSpawner.debug | true | Расширенное логирование dockerspawner. |
| c.Authenticator.admin\_users | {'u\_m0weu', 'u\_m0je7', 'u\_m064s'} | Список учетных записий пользователей jupyterhub с правами администратора. |
| def prespawn\_hook(spawner): … |  | Код, который создает пользовательскую директорию в /app/share/, если ее там нет. |
| c.DockerSpawner.pre\_spawn\_hook | prespawn\_hook | pre\_spawn\_hook – код, который запускается перед стартом контейнера. В данном случае перед стартом контейнера создается директория в /app/share (см. выше) |

1. Административные действия с JupyterHub

Запуск JupyterHub осуществлется командой

nohup jupyterhub >> /var/logs/jhub/jhub-datamonet.log &

выполняемой из директории, где находится конфиг. Запуск необходимо выполнять под root. Файл /var/logs/jhub/jhub-datamonet.log является файлом с логами JupyterHub. Название файла выбирается администратором. Для инстанса отдела монетищации данных файл с логами предлагается называть jhub-datamonet.log. Для инстансов отдела аналитики и моделирования МСКБ предлагается называть файл с логами jhub-mskb-<название среды>.log

Остановка JupyterHub осуществляется серией команд

ps alx | grep configurable | grep "port < номер порта на котором доступен jupyterhub для пользователей >"

kill –term <номер процесса нужного JupyterHub>

нужный процесс предлагается искать по PPID configurable http proxy с портом, на котором доступен JupyterHub.

Для деплоя нового инстанса JupyterHub нужно

* Создать подкаталог /opt/jupyterhub/<название папки>
* Скопировать в него jupyterhub\_config.py
* Задать новые значения для параметров
  + c.JupyterHub.port
  + c.ConfigurableHTTPProxy.api\_url
  + c.JupyterHub.hub\_port
* Выполнить запуск согласно пункту «Запуск JupyterHub»

Просмотр логов JupyterHub

1. Для просмотра логов всего приложения используется файл с логами, определнный при запуске.
2. В случае докерезированного JupyterHub можно дополнительно посмотреть логи контейнера, в котором работает ноутбук конкретного пользователя. Для этого
   * Командой docker ps посмотреть работающие контейнеры. К названию контейнера конкатенируется имя пользователя ОС, таким образом можно определить нужный контейнер.
   * Командой docker logs <название контейнера> посмотреть логи контейнера.
3. Docker контейнеры, используемые в приложени

В рамках приложения используются docker образы 2 типов – основной контейнер, в котором запускается ноутбук, и контейнер с конфигурацией Hadoop.

Для образа с ноутбуком за основу берется официальный образ проекта Jupyter: <https://github.com/jupyter/docker-stacks/tree/master/scipy-notebook>

Дальше в образ устанавливаются дополнительные пакеты Python, библиотеки и приложения типа Kerberos и gssapi и Java. Примеры Dockerfile:



Контейнер с конфигом Hadoop создается для удобства монтирования директорий, содержащих настроечные файлы Хадуп в контейнер с ноутбуком. За основу берется образ busybox <https://hub.docker.com/_/busybox/>

Дальше в нем создаются точки монтирования

* /opt/cloudera
* /etc/spark2
* /etc/hadoop
* /etc/alternatives/spark2-conf
* /etc/alternatives/hadoop-conf

После этого контейнер с конфигом испольутеся при запуске контейнеров с ноутбуками (опция c.DockerSpawner.extra\_host\_config.update({ 'volumes\_from': 'configContainer' }) в jupyterhub\_config.py).

На случай сбоя системы, docker образы хранятся в двух местах: в каталоге /root/Dockers/ в виде .tar архивов (файлы notebook.tar и datacontainer.tar) и в docker registry.

1. Административные действия с docker контейнерами

**Порядок изменения образа, в котором запускаются ноутбуки**

1. Оповестить пользователей о времнеи недоступности JupyterHub
2. Остановить JupyterHub (см. параграф 7)
3. Поменять в файле jupyterhub\_config.py значение параметра c.DockerSpawner.image на строку с нужным названием образа
4. Командой docker ps –а найти все контейнеры с ноутбуками. Такие контейнеры имеют название вида jupyter[-<название среды>]<username>. Уничтожить данные контейнеры командой docker rm –f <имя контейнера>
5. Выполнить запуск JupyterHub (см. параграф 7)

**Подключение к контейнеру пользователя с целью диагностики ошибок**

Иногда бывает ползено, с целью разбора инцидента, подключиться к контейнеру пользователя и выполнить ряд команд Linux. Для этого

1. Командой docker ps найти нужный контейнер
2. Выполнить команду docker exec –it <имя контейнера> /bin/bash
3. Вы в контейнере

**Установка новых библиотек в образ**

На данный момент в образе datamonetization/notebook:base настроена связь с репозиторием pip, развернутом внтури контура банка. Для установки библиотек из этого репозитория нужно

1. Создать в /root/Dockers поддиректорию, например, newnotebook
2. Создать в этой директории Dockerfile вида
   * FROM datamonetization/notebook:base
   * RUN pip3 install django –t /opt/conda/lib/python3.6/site-packages
   * CMD [“start-notebook.sh”]
3. Выполнить комануд docker build /root/Dockers/newnotebook –t datamonetization/notebook:20180905
4. Так как одна из главных целей при использовании докера – версионирование сред, важно новые версии образа тегать именем, отличным от datamonetization/notebook:base. Пока предлагается в качестве лейбла использовать дату, на которую был собран образ, можно также применять семантические метки, например datamonetization/notebook:libhdfs3.
5. Сразу после создания нового образа рекомендуется сделать push в docker registry (см. ниже)
6. При необходимости выполнить действия из пункта «изменение образа, который запускается в JupyterHub»

**Диагоностика ошибок образа**

Если есть предположение, что в самом докер образе содержатся ошибки, диагностику можно провести следующими способами

1. Выполнить команду docker run -it <имя контейнера> /bin/bash – в окажетесь в командной строке контейнера
2. Если вдобавок нужен доступ к керберос: docker run -it --network host -v /etc/krb5.conf:/etc/krb5.conf:ro <имя контейнера> /bin/bash
3. Если вдобавок нужен доступ к Hadoop: docker run -it --network host -v /etc/krb5.conf:/etc/krb5.conf:ro --volumes-from configContainer <имя контейнера> /bin/bash
4. Вместо /bin/bash можно написать любую другую команду. При отсутствии команды, выполняется команда из блока CMD докерфайла, в случае datamonetization/notebook:base это запуск однопользовательского jupyter notebook.
5. Если даже запуск контейнера оказался не успешным, все равно можно найти контейнер через docker ps -a и посмотреть логи.

**Взаимодействие с внутренним docker registry**

* Логин под доменной УЗ: docker login datalake.binary.alfabank.ru
* Push
  1. Нужно протегать образ, который вы хотите запушшить, тегом, который содержит в себе адрес внутреннего registry: docker tag datamonetization/notebook:base datalake.binary.alfabank.ru /datamonetization/notebook:base
  2. По префиксу тега докер понимает, куда выполнять push: push datalake.binary.alfabank.ru /datamonetization/notebook:base
  3. docker pull datalake.binary.alfabank.ru /datamonetization/notebook:base

**Локальное сохранение и локальная загрузка образа**

* docker save --output /root/Dockers/image.tar some\_docker\_image
* docker load --input /root/Dockers/image.tar

10. Часто встречающиеся ошибки

Общее замечание: приложение стабильно работает только в браузере Google Chrome. Если у пользователя возникает ошибка, которая предположительно связана с интерфейсом (не работает логин, не работает ноутбук у одного пользователя, но работает у всех остальных), нужно первым делом поинтересоваться, каким браузером он пользуется и, если это не Google Chrome, посоветовать ему использовать этот браузер.  
  
Ошибка: у пользователя не получается залогиниться.  
Решение: посоветовать пользователю почистить историю браузера, в частности, удалить cookies. Также нужно донести до пользователя, что нельзя логиниться с одной машины под двумя аккаунтами.  
  
Ошибка: у пользователя не запускается Ядро PySpark.  
Решение: убедиться, что пользователь сделал kinit. После этого убедиться, что в файле /opt/cloudera/parcels/SPARK2/lib/spark2/conf/yarn-conf/yarn-site.xml для ключа yarn.resourcemanager.principal указано значение yarn/bda31node04.moscow.alfaintra.net@BDA.MOSCOW.ALFAINTRA.NET а не yarn/\_HOST@BDA.MOSCOW.ALFAINTRA.NET. После этого посмотреть логи контейнера пользователя (см. параграф 7). Если таким образом не удается определить проблему, передать анализ инцидента на 3 линию поддержки.  
  
Ошибка: у пользователя не работает вызов команды ОС через командую строку или через пакет subprocess Python (пример: subprocess.run(["hadoop", "fs", "-get", hdfs\_path, local\_path],stderr=subprocess.PIPE, stdout=subprocess.PIPE) )  
Решение: вероятно, данный бинарный файл есть в файловой системе образа, но не находится в переменной $PATH образа. Такое часто случается, когда файл на хост машине прилинкован в /usr/bin из /opt/cloudera (приложения Hadoop типа hdfs dfs). Тогда нужно найти бинарный файл и попросить пользователя запускать команду из этой директории. После этого сделать запрос 3 линии поддержки на добавления данной директории в $PATH образа.

11. Восстановление после сбоя

Для того, чтобы восстановить работу приложений bipython2 после сбоя:

1. Проверить конфиги spark, hadoop и hive (см параграф 12)
2. Сделать kinit под пользователем u\_m0weu (или иной личной учеткой с правами на чтение из какого-нибудь каталога HDFS)
3. Запустить обычный jupyterhub:  JupyterHub --ip="172.25.215.190"
   * Для проверки обычного JupyterHub
   * Залогиниться на bipython2:8000
   * Открыть ноутбук Python + PySpark
   * Написать в ячейке слово «spark» и выполнить код – должна появиться информация о запущенной сессии Spark
4. Запустить zeppelin: /opt/zeppelin/bin/zeppelin.sh
   * Для проверки zeppelin залогиниться на bipython:8123
5. Проверить, что работает R Studio Server: залогиниться на bipython2:8787
   * Если не работает, запустить командой rstudio-server start
6. Проверить работу докер демона
   * docker version
   * docker ps –a
   * docker run –it --network host –v /etc/krb5.conf:/etc/krb5.conf:ro –volumes-from configContainer datamonetization/notebook:base /bin/bash
   * в открывшемся контейнере сделать kinit и зайти в pyspark
7. Если докер демон не работает, например, команды висят
   * Попытаться сохранить все образы, которые еще не сохранены или не запушенны
   * Очистить /var/run/docker, /var/docker
   * Выполнить команду yum uninstall docker
   * Переустановить docker, либо из rpm файла /root/docker...<номер версии>.rpm, либо скачав соотв. rpm из интернета.
   * Загрузить сохраненные докер образы (либо локально, либо из registry)
   * Запустить configContainer
8. После этого запустить докерезированный JupyterHub (см. параграф 7)

Залогиниться на bipython2:9876 и выполнить такую же проверку, как для обычного JupyterHub (см. выше)

1. Проверка взаимодействия между BDA и bipython2

После проведения любых настроек на кластере BDA31, а также в случае Kerberos ошибок при попытке запустить Spark, а также ошибок вида «висит sqoop или pysqoop» или «не работает beeline» нужно сделать следующие действия на bipython2:

1. В файлах /opt/cloudera/parcels/SPARK2/lib/spark2/conf/yarn-conf/yarn-site.xml, /etc/hadoop/conf.cloudera.yarn/yarn-site.xml, /etc/hive/conf.cloudera.hive/yarn-site.xml значение ключа yarn.resourcemanager.principal с yarn/\_HOST@BDA.MOSCOW.ALFAINTRA.NET на yarn/bda31node04.moscow.alfaintra.net@BDA.MOSCOW.ALFAINTRA.NET
2. Для проверки корректности запустить команду pyspark и дождаться создания Spark сессии
3. Также для проверки корректности нужно сделать kinit под пользователем u\_m0weu и запустить команду

sqoop import --connect 'jdbc:oracle:thin:@(DESCRIPTION=(CONNECT\_TIMEOUT=1)(TRANSPORT\_CONNECT\_TIMEOUT=1)(ADDRESS\_LIST=(ADDRESS=(PROTOCOL=TCP)(HOST=exa2-scan)(PORT=1521))(ADDRESS=(PROTOCOL=TCP)(HOST=exa1-scan)(PORT=1521)))(CONNECT\_DATA=(SERVER=dedicated)(SERVICE\_NAME=SANDBOX\_TAF)))' \

--username bmbpilot \

--password bmb255pilot \

--target-dir /user/u\_m0weu/db/SEGMENT\_TYPE -m 1 \

--table SEGMENT\_TYPE \

--hive-import \

--hive-table u\_m0weu.dict\_camp\_test\_staging \

--delete-target-dir \

--hive-drop-import-delims \

--compress \

--hive-overwrite \

--verbose