क्लाउड कंप्यूटिंग और बिग डेटा का परिचय

इस पाठ में निम्नलिखित विषय शामिल हैं:

- स्पार्क
- हडूप
- कुबेरनेट्स
- डॉकर
- फ्लिंक
- मोंगोडीबी

क्लाउड कंप्यूटिंग की बात करें तो ऐसा लगता है कि यह बहुत सारे टूल्स के बिना अधूरी है, जैसे □□□□□□, □□□□, □□□□□, □□□□□□□□□, □□□□□□, □□□□□□□□□□, □□□□□, □□□□□, □□□□□□□, □□□□□, □□□□□□, □□□□□□□ □□□□□□। क्या आपने इन सभी के बारे में सुना है? इनमें से कुछ टूल्स मैंने और की जॉब डिस्क्रिप्शन में देखे हैं। ये सभी उच्च वेतन वाली पद हैं। हम इन सभी को इंस्टॉल करने की कोशिश करेंगे और उन्हें थोड़ा सा समझने की कोशिश करेंगे। ## □□□□□ का प्रारंभिक परिचय

आधिकारिक वेबसाइट के अनुसार, `Spark` बड़े पैमाने पर डेटा के विश्लेषण के लिए एक इंजन है। `Spark` एक लाइब्रेरी का सेट है। यह `Redis` की तरह सर्वर और क्लाइंट में विभाजित नहीं होता है। `Spark` केवल क्लाइंट साइड पर उपयोग किया जाता है। आधिकारिक वेबसाइट से नवीनतम संस्करण डाउनलोड किया गया है, `spark-3.1.1-bin-hadoop3.2.tar`।

```
$ tree . -L 1
.
  LICENSE
  NOTICE
  R
  README.md
  RELEASE
  bin
  conf
  data
  examples
  jars
  kubernetes
  licenses
  python
  sbin
  yarn
```

11 डायरेक्टरीज़, 4 फ़ाइलें

                          (libraries)

    ,   ,   Python   (dependency library)   `pip install pyspark`

```shell
$ pip install pyspark
Collecting pyspark
  Downloading pyspark-3.1.1.tar.gz (212.3 MB)
     |                 | 212.3 MB 14 kB/s
Collecting py4j==0.10.9
  Downloading py4j-0.10.9-py2.py3-none-any.whl (198 kB)
     |                 | 198 kB 145 kB/s
Building wheels for collected packages: pyspark
  Building wheel for pyspark (setup.py) ... done
  Created wheel for pyspark: filename=pyspark-3.1.1-py2.py3-none-any.whl size=212767604 sha256=0b8079e8
  Stored in directory: /Users/lzw/Library/Caches/pip/wheels/23/bf/e9/9f3500437422e2ab82246f25a51ee480a4
Successfully built pyspark
Installing collected packages: py4j, pyspark
Successfully installed py4j-0.10.9 pyspark-3.1.1
```

इंस्टॉल हो गया है।

यह वेबसाइट पर कुछ उदाहरण देख रहा है।

```
./bin/run-example SparkPi 10
```

ओह, मैंने अभी डाउनलोड किए गए इंस्टॉलेशन पैकेज में मौजूद प्रोग्राम को चलाने की कोशिश की। लेकिन एक त्रुटि हुई।

```
$ ./bin/run-example SparkPi 10
21/03/11 00:06:15 WARN NativeCodeLoader:              -                ...       ,   -
21/03/11 00:06:16 INFO ResourceUtils: spark.driver
21/03/11 00:06:16 WARN Utils:    'sparkDriver'
```

> □□□□□ एक तेज़ और सामान्य प्रसंस्करण इंजन है जो □□□□□□ डेटा के साथ संगत है। यह □□□□ या □□□□□ के स्टैंडअलोन मोड के माध्यम से □□□□□□ क्लस्टर में चल सकता है, और यह □□□□, □□□□□, □□□□□□□□□, □□□□, और किसी भी □□□□□□ □□□□□□□□□□□ में डेटा को प्रोसेस कर सकता है। इसे बैच प्रोसेसिंग (□□□□□□□□□ के समान) और नए वर्कलोड जैसे स्ट्रीमिंग, इंटरएक्टिव क्वेरीज़, और मशीन लर्निंग को संचालित करने के लिए डिज़ाइन किया गया है।

hadoop कई बार आया है। spark depends hadoop को □□□□□□ करने के बाद, मुझे यह वाक्य मिला। ऐसा लगता है कि यह Hadoop फॉर्मेट के डेटा पर निर्भर करता है। आइए पहले Hadoop पर शोध करें।

□□□□□□

आधिकारिक वेबसाइट को संक्षेप में देखने के बाद, इसे इंस्टॉल करने का प्रयास करते हैं।

```
brew install hadoop
```

इंस्टॉलेशन के दौरान, आइए इसके बारे में थोड़ा जानें।

> □□□□□□ □□□□□□ सॉफ्टवेयर लाइब्रेरी एक ऐसा फ्रेमवर्क है जो सरल प्रोग्रामिंग मॉडल का उपयोग करके कंप्यूटरों के क्लस्टर पर बड़े डेटा सेट के वितरित प्रसंस्करण की अनुमति देता है। यह सिंगल सर्वर से हजारों मशीनों तक स्केल करने के लिए डिज़ाइन किया गया है, जहां प्रत्येक मशीन स्थानीय गणना और भंडारण प्रदान करती है। हार्डवेयर पर निर्भर होने के बजाय उच्च उपलब्धता प्रदान करने के लिए, यह लाइब्रेरी स्वयं एप्लिकेशन लेयर पर विफलताओं का पता लगाने और उन्हें संभालने के लिए डिज़ाइन की गई है, इस प्रकार कंप्यूटरों के एक क्लस्टर के ऊपर एक अत्यधिक उपलब्ध सेवा प्रदान करती है, जहां प्रत्येक मशीन विफलताओं के प्रति संवेदनशील हो सकती है।

□□□□□□ एक ऐसा फ्रेमवर्क है जो वितरित डेटासेट को प्रोसेस करने के लिए बनाया गया है। ये डेटासेट कई कंप्यूटरों पर वितरित हो सकते हैं। इसे एक बहुत ही सरल प्रोग्रामिंग मॉडल का उपयोग करके प्रोसेस किया जाता है। इसे एकल सर्वर से हजारों मशीनों तक स्केल करने के लिए डिज़ाइन किया गया है। हार्डवेयर की उच्च उपलब्धता पर निर्भर होने के बजाय, यह लाइब्रेरी एप्लिकेशन लेयर पर ही त्रुटियों की जांच और उन्हें संभालने के लिए डिज़ाइन की गई है। इस तरह, उच्च उपलब्धता वाली सेवाओं को क्लस्टर में तैनात किया जा सकता है, भले ही क्लस्टर में शामिल हर कंप्यूटर विफल हो सकता हो।

```
$ brew install hadoop
  :
  homebrew-core
  homebrew-cask
`brew update`       ,               :
  git -C /usr/local/Homebrew/Library/Taps/homebrew/homebrew-core fetch --unshallow
  git -C /usr/local/Homebrew/Library/Taps/homebrew/homebrew-cask fetch --unshallow

GitHub                                          Homebrew/homebrew-core    Homebrew/homebrew-cask
==> Downloading https://homebrew.bintray.com/bottles/openjdk-15.0.1.big_sur.bottle.tar.gz
              : /Users/lzw/Library/Caches/Homebrew/downloads/d1e3ece4af1d225bc2607eaa4ce85a873d2c6d43757
==> Downloading https://www.apache.org/dyn/closer.lua?path=hadoop/common/hadoop-3.3.0/hadoop-3.3.0.tar.g
              : /Users/lzw/Library/Caches/Homebrew/downloads/764c6a0ea7352bb8bb505989feee1b36dc628c2dcd6
```

```
==> hadoop                    : openjdk
==> hadoop                : openjdk
==> openjdk-15.0.1.big_sur.bottle.tar.gz
==>
    Java         JDK          ,                    :
  sudo ln -sfn /usr/local/opt/openjdk/libexec/openjdk.jdk /Library/Java/JavaVirtualMachines/openjdk.jdk
```

□□□□□□□□ □□□-□□□□□ है, जिसका अर्थ है कि इसे /□□□/□□□□□□ में □□□□□□□□ नहीं किया गया है,
क्योंकि यह □□□□□□ के java □□□□□□□□ को छिपाता है।

यदि आपको अपने □□□□□ में □□□□□□□□ को पहले रखने की आवश्यकता है, तो निम्नलिखित कमांड चलाएँ: bash echo 'export PATH="/usr/local/opt/openjdk/bin:$PATH"' >> /Users/lzw/.bash_profile

कंपाइलर्स को □□□□□□□□ ढूंढने के लिए आपको यह सेट करने की आवश्यकता हो सकती है: □□□□□□□ □□□□□□□□□□="-□/□□□/□□□□□□/□□□□/□□□□□□□□/□□□□□□□□"

==> सारांश □ /□□□/□□□□□□/□□□□□□□/□□□□□□□□/15.0.1: 614 फ़ाइलें, 324.9□□ ==> □□□□□□□ इंस्टॉल कर रहा है □ /□□□/□□□□□□/□□□□□□□/□□□□□□□/3.3.0: 21,819 फ़ाइलें, 954.7□□, 2 मिनट 15 सेकंड में बनाया गया ==> 1 निर्भर को अपग्रेड कर रहा है: □□□□□□ 3.3.3 -> 3.6.3_1 ==> □□□□□□ 3.3.3 -> 3.6.3_1 को अपग्रेड कर रहा है ==> डाउनलोड कर रहा है □□□□□□://□□□.□□□□□□□.□□□□/□□□□/□□□□□□□.□□□□?□□□□□=□□□□□□/□□□□□□-3/3.6.3/□□□□□□□□□/□□□□□□□-□□□□□□-3.6.3-□□□.□□□.□□ ==> डाउनलोड कर रहा है □□□□□□://□□□□□□□.□□□□□□□□□□.□□□□/□□□□/□□□□□□□/□□□□□□/□□□□□□-3/3.6.3/□□□□□□□□□/□□□□□□□-□□□□□□-3.6.3-□□□.□□□.□□ से ##
100.0% त्रुटि: brew link चरण सफलतापूर्वक पूरा नहीं हुआ फ़ॉर्मूला बनाया गया, लेकिन /□□□/□□□□□□ में सिमलिंक नहीं किया गया □□□/□□□ को सिमलिंक नहीं किया जा सका लक्ष्य /□□□/□□□□□□/□□□/□□□ □□□□□□ का एक सिमलिंक है। आप इसे अनलिंक कर सकते हैं: □□□□ □□□□□□□ □□□□□□

लिंक को बाध्य करने और सभी संघर्षरत फ़ाइलों को ओवरराइट करने के लिए: □□□□ □□□□ –□□□□□□□□□□ □□□□□□

सभी फ़ाइलों को सूचीबद्ध करने के लिए जो हटा दी जाएंगी: □□□□ □□□□ –□□□□□□□□□□ –□□□-□□□ □□□□□□

संभावित विरोधाभासी फ़ाइलें हैं: /□□□/□□□□□□/□□□/□□□ -> /□□□/□□□□□□/□□□□□□□/□□□□□□/3.3.3/□□□/□□□ /□□□/□□□□□□/□□□/□□□□□□□□□ -> /□□□/□□□□□□/□□□□□□□/□□□□□□/3.3.3/□□□/□□□□□□□□□ /□□□/□□□□□□/□□□/□□□□□□□ -> /□□□/□□□□□□/□□□□□□□/□□□□□□/3.3.3/□□□/□□□□□□□ ==> सारांश □ /□□□/□□□□□□/□□□□□□□/□□□□□□/3.6.3_1: 87 फ़ाइलें, 10.7□□, 7 सेकंड में बनाया गया हटाया जा रहा है: /□□□/□□□□□□/□□□□□□□/□□□□□□/3.3.3... (92 फ़ाइलें, 9□□) ==> अपग्रेड किए गए फ़ॉर्मूले के आश्रितों की जाँच कर रहा है... ==> कोई टूटे हुए आश्रित नहीं मिले! ==> चेतावनी ==> □□□□□□□□ सिस्टम □□□□ □□□□□□□□□ को इस □□□ को खोजने के लिए, इसे □□□□□□□□ करें □□□□ □□ -□□□ /□□□/□□□□□□/□□□/□□□□□□□□/□□□□□□□□/□□□□□□□□.□□□ /□□□□□□□□/□□□□/□□□□□□□□□□□□□□□□□□□□/□□□□□□□□.□□□

□□□□□□□□ □□□-□□□□□ है, जिसका अर्थ है कि इसे /□□□/□□□□□□ में □□□□□□□□ नहीं किया गया है, क्योंकि यह □□□□□□ के java □□□□□□□□ को छिपाता है।

यदि आपको अपने □□□□ में □□□□□□□□ को सबसे पहले रखने की आवश्यकता है, तो निम्नलिखित कमांड चलाएँ: bash echo 'export PATH="/usr/local/opt/openjdk/bin:$PATH"' >> /Users/lzw/.bash_profile

कंपाइलर को □□□□□□□□ ढूंढने के लिए आपको यह सेट करने की आवश्यकता हो सकती है: □□□□□□□ □□□□□□□□□=“-□/□□□/□□□□□□/□□□/□□□□□□□□/□□□□□□□□”

`brew` `maven` , `3.6.3_1`

```shell
  brew link --overwrite maven
```

(यह कमांड □□□□□ को लिंक करने और पहले से मौजूद फाइलों को ओवरराइट करने के लिए उपयोग की जाती है।)

Hadoop सफलतापूर्वक इंस्टॉल हो गया है।

मॉड्यूल

इस प्रोजेक्ट में निम्नलिखित मॉड्यूल शामिल हैं:

- □□□□□□□ □□□□□□□: अन्य □□□□□□□ मॉड्यूल का समर्थन करने वाले सामान्य उपयोगिताएँ।
- □□□□□□□ □□□□□□□□□□□□□ □□□□□ □□□□□□□ **(**□□□□□□**)**: एक वितरित फ़ाइल सिस्टम जो एप्लिकेशन डेटा तक उच्च-थ्रूपुट पहुंच प्रदान करता है।
- □□□□□□□ □□□□□: जॉब शेड्यूलिंग और क्लस्टर संसाधन प्रबंधन के लिए एक फ्रेमवर्क।
- □□□□□□□ □□□□□□□□□□□: बड़े डेटा सेट के समानांतर प्रसंस्करण के लिए एक □□□□□-आधारित सिस्टम।
- □□□□□□□ □□□□□□: □□□□□□□ के लिए एक ऑब्जेक्ट स्टोर।

यह कहा जा रहा है कि ये मॉड्यूल हैं। इसे टाइप करने पर hadoop दिखाई देगा:

```
$ hadoop
   : hadoop [OPTIONS] SUBCOMMAND [SUBCOMMAND OPTIONS]
      hadoop [OPTIONS] CLASSNAME [CLASSNAME OPTIONS]
     CLASSNAME        -    Java

OPTIONS                            :

--config dir                      Hadoop
--debug
--help
buildpaths
```

```
hostnames list[,of,host,names]
hosts filename
loglevel level                          log4j
workers

  SUBCOMMAND            :
           :

daemonlog                       /

          :

archive       Hadoop
checknative   Hadoop
classpath     Hadoop jar
conftest            XML
credential
distch
distcp
dtutil
envvars               Hadoop
fs
gridmix                         ,
jar <jar>       jar          :    YARN                "yarn jar"        ,
jnipath       java.library.path
kdiag         Kerberos
kerbname      auth_to_local
key           KeyProvider
rumenfolder     rumen
rumentrace          rumen
s3guard       S3
trace         Hadoop
version

     :
```

```
kms           KMS    ,
registrydns        DNS

SUBCOMMAND              -h
```

आधिकारिक वेबसाइट पर कुछ उदाहरण दिए गए हैं।

```
  $ mkdir input
  $ cp etc/hadoop/*.xml input
  $ bin/hadoop jar share/hadoop/mapreduce/hadoop-mapreduce-examples-3.2.2.jar grep input output 'dfs[a-z
  $ cat output/*
```

ध्यान दें कि `share/hadoop/mapreduce/hadoop-mapreduce-examples-3.2.2.jar` मौजूद है। इसका मतलब यह हो सकता है कि कुछ उदाहरण फ़ाइलें हमें नहीं मिली हैं। अनुमान है कि `Homebrew` के माध्यम से इंस्टॉल करने पर ये फ़ाइलें नहीं मिलती हैं। हमने आधिकारिक वेबसाइट से इंस्टॉलेशन पैकेज डाउनलोड किया है।

```
$ tree . -L 1
.
  LICENSE-binary
  LICENSE.txt
  NOTICE-binary
  NOTICE.txt
  README.txt
  bin
  etc
  include
  lib
  libexec
  licenses-binary
  sbin
  share
```

`share` डायरेक्टरी दिखाई दी है। हालांकि, क्या `Homebrew` वास्तव में इन अतिरिक्त फ़ाइलों को शामिल नहीं करता है? `Homebrew` के इंस्टॉलेशन डायरेक्टरी को ढूंढें।

```
$ type hadoop
hadoop   /usr/local/bin/hadoop
```

```
$ ls -alrt /usr/local/bin/hadoop
lrwxr-xr-x  1 lzw  admin  33 Mar 11 00:48 /usr/local/bin/hadoop -> ../Cellar/hadoop/3.3.0/bin/hadoop
$ cd /usr/local/Cellar/hadoop/3.3.0
```

यह /usr/local/Cellar/hadoop/3.3.0/libexec/share/hadoop के अंतर्गत प्रिंट की गई डायरेक्टरी ट्री है।

```
$ tree . -L 2
.
   client
      hadoop-client-api-3.3.0.jar
      hadoop-client-minicluster-3.3.0.jar
      hadoop-client-runtime-3.3.0.jar
   common
      hadoop-common-3.3.0-tests.jar
      hadoop-common-3.3.0.jar
      hadoop-kms-3.3.0.jar
      hadoop-nfs-3.3.0.jar
      hadoop-registry-3.3.0.jar
      jdiff
      lib
      sources
      webapps
   hdfs
      hadoop-hdfs-3.3.0-tests.jar
      hadoop-hdfs-3.3.0.jar
      hadoop-hdfs-client-3.3.0-tests.jar
      hadoop-hdfs-client-3.3.0.jar
      hadoop-hdfs-httpfs-3.3.0.jar
      hadoop-hdfs-native-client-3.3.0-tests.jar
      hadoop-hdfs-native-client-3.3.0.jar
      hadoop-hdfs-nfs-3.3.0.jar
      hadoop-hdfs-rbf-3.3.0-tests.jar
      hadoop-hdfs-rbf-3.3.0.jar
      jdiff
      lib
      sources
      webapps
```

```
mapreduce
    hadoop-mapreduce-client-app-3.3.0.jar
    hadoop-mapreduce-client-common-3.3.0.jar
    hadoop-mapreduce-client-core-3.3.0.jar
    hadoop-mapreduce-client-hs-3.3.0.jar
    hadoop-mapreduce-client-hs-plugins-3.3.0.jar
    hadoop-mapreduce-client-jobclient-3.3.0-tests.jar
    hadoop-mapreduce-client-jobclient-3.3.0.jar
    hadoop-mapreduce-client-nativetask-3.3.0.jar
    hadoop-mapreduce-client-shuffle-3.3.0.jar
    hadoop-mapreduce-client-uploader-3.3.0.jar
    hadoop-mapreduce-examples-3.3.0.jar
    jdiff
    lib-examples
    sources
tools
    dynamometer
    lib
    resourceestimator
    sls
    sources
yarn
    csi
    hadoop-yarn-api-3.3.0.jar
    hadoop-yarn-applications-catalog-webapp-3.3.0.war
    hadoop-yarn-applications-distributedshell-3.3.0.jar
    hadoop-yarn-applications-mawo-core-3.3.0.jar
    hadoop-yarn-applications-unmanaged-am-launcher-3.3.0.jar
    hadoop-yarn-client-3.3.0.jar
    hadoop-yarn-common-3.3.0.jar
    hadoop-yarn-registry-3.3.0.jar
    hadoop-yarn-server-applicationhistoryservice-3.3.0.jar
    hadoop-yarn-server-common-3.3.0.jar
    hadoop-yarn-server-nodemanager-3.3.0.jar
    hadoop-yarn-server-resourcemanager-3.3.0.jar
    hadoop-yarn-server-router-3.3.0.jar
```

```
hadoop-yarn-server-sharedcachemanager-3.3.0.jar
hadoop-yarn-server-tests-3.3.0.jar
hadoop-yarn-server-timeline-pluginstorage-3.3.0.jar
hadoop-yarn-server-web-proxy-3.3.0.jar
hadoop-yarn-services-api-3.3.0.jar
hadoop-yarn-services-core-3.3.0.jar
lib
sources
test
timelineservice
webapps
yarn-service-examples
```

आप देख सकते हैं कि बहुत सारे `jar` पैकेज हैं।

```
$ mkdir input
$ ls
bin         hadoop-config.sh    hdfs-config.sh      libexec         sbin            yarn-config.sh
etc         hadoop-functions.sh input           mapred-config.sh    share
$ cp etc/hadoop/*.xml input
$ cd input/
$ ls
capacity-scheduler.xml  hadoop-policy.xml   hdfs-site.xml       kms-acls.xml        mapred-site.xml
core-site.xml       hdfs-rbf-site.xml   httpfs-site.xml     kms-site.xml        yarn-site.xml
$ cd ..
$ bin/hadoop jar share/hadoop/mapreduce/hadoop-mapreduce-examples-3.2.2.jar grep input output 'dfs[a-z.]
JAR                         : /usr/local/Cellar/hadoop/3.3.0/libexec/share/hadoop/mapreduce/hadoop-mapr
$
$ bin/hadoop jar share/hadoop/mapreduce/hadoop-mapreduce-examples-3.3.0.jar grep input output 'dfs[a-z.]
2021-03-11 01:54:30,791 WARN util.NativeCodeLoader:               -                ...       ,   -
2021-03-11 01:54:31,115 INFO impl.MetricsConfig: hadoop-metrics2.properties
2021-03-11 01:54:31,232 INFO impl.MetricsSystemImpl:            10
...
```

वेबसाइट के उदाहरण के अनुसार टाइप करें। ध्यान दें कि `bin/hadoop jar share/hadoop/mapreduce/hadoop-mapreduce-examples-3.2.` `grep input` में, `jar` पैकेज से पहले वर्ज़न नंबर है। इसलिए हमें इसे अपने `3.3.0` से बदलना होगा।

लॉग का अंत:

```
2021-03-11 01:54:35,374 INFO mapreduce.Job:  map 100% reduce 100%
2021-03-11 01:54:35,374 INFO mapreduce.Job: Job job_local2087514596_0002
2021-03-11 01:54:35,377 INFO mapreduce.Job:    : 30

        FILE:             =1204316
        FILE:              =3565480
        FILE:             =0
        FILE:               =0
        FILE:              =0
    Map-Reduce
        Map        =1
        Map         =1
        Map        =17
        Map              =25
                  =141
                    =0
                     =0
                  =1
                  =25
                    =1
                     =1
                =2
             =1
            =0
               =1
        GC        (ms)=57
                     (   )=772800512

        BAD_ID=0
        CONNECTION=0
        IO_ERROR=0
        WRONG_LENGTH=0
        WRONG_MAP=0
        WRONG_REDUCE=0

               =123
```

=23

जारी रखें और देखें।

```
$ cat output/*
1   dfsadmin
```

यह वास्तव में क्या मतलब है? कोई बात नहीं, कुल मिलाकर हमने Hadoop को चला दिया है। और पहला स्टैंडअलोन कंप्यूटेशन उदाहरण भी चलाया है।

स्पार्क

□□□□□ पर वापस आते हैं। एक उदाहरण देखते हैं।

```
text_file = sc.textFile("hdfs://...")
counts = text_file.flatMap(lambda line: line.split(" ")) \
             .map(lambda word: (word, 1)) \
             .reduceByKey(lambda a, b: a + b)
counts.saveAsTextFile("hdfs://...")
```

(यह कोड ब्लॉक को अनुवादित नहीं किया गया है क्योंकि यह □□□□□□ कोड है और इसे हिंदी में अनुवादित करने की आवश्यकता नहीं है।)

यहां hdfs फ़ाइल दिखाई दी है। जांच करने के बाद, पता चला कि hdfs फ़ाइल इस तरह बनाई जा सकती है:

```
hdfs dfs -mkdir /test
```

hdfs कमांड को देखते हैं।

```
$ hdfs
   : hdfs [   ]     [       ]
```

□□□□□□□ कोई नहीं या निम्नलिखित में से कोई भी हो सकता है:

–□□□□□□□□□□ बिल्ड ट्री से क्लास फ़ाइलें जोड़ने का प्रयास करें –□□□□□□ □□□ □□□□□□ कॉन्फ़िग डायरेक्टरी –□□□□□□ (□□□□□|□□□□□□|□□□□) डेमन पर कार्रवाई करें –□□□□□ शेल स्क्रिप्ट डिबग मोड चालू करें –□□□□ उपयोग जानकारी –□□□□□□□□□ □□□□[,□□,□□□□,□□□□□] वर्कर मोड में उपयोग करने के लिए होस्ट्स –□□□□□ □□□□□□□□ वर्कर मोड में उपयोग करने के लिए होस्ट्स की सूची –□□□□□□□□ □□□□□ इस कमांड के लिए □□□4□ स्तर सेट करें –□□□□□□□ वर्कर मोड चालू करें

□□□□□□□□□□ निम्नलिखित में से एक हो सकता है: □□□□□ □□□□□□□□:

□□□□□□□□□□ □□□□ कैश को कॉन्फ़िगर करें □□□□□□ □□□□ एन्क्रिप्शन ज़ोन को कॉन्फ़िगर करें □□□□□ □□□□ डीबग कमांड्स को एक्ज़ीक्यूट करने के लिए डीबग एडमिन चलाएं □□□□□□□□ एक □□□ एडमिन क्लाइंट चलाएं □□□□□□□□□□□□□□□ राउटर-आधारित फेडरेशन का प्रबंधन करें □□ □□□□ इरेज़र कोडिंग □□□ चलाएं □□□□ □□□ फ़ाइलसिस्टम चेकिंग यूटिलिटी चलाएं □□□□□□□ □□□ □□ एडमिन क्लाइंट चलाएं □□□□□□ □□□□□□□□ या □□□□□□□□ से □□□ एक्सपोर्टेड वैल्यूज़ प्राप्त करें □□□ एडिट्स फ़ाइल पर ऑफ़लाइन एडिट्स व्यूअर लागू करें □□□ □□□□□□□ पर ऑफ़लाइन □□□□□□□ व्यूअर लागू करें □□□_□□□□□□ लेगेसी □□□□□□□ पर ऑफ़लाइन □□□□□□□ व्यूअर लागू करें □□□□□□□□□□□□□□□ ब्लॉक स्टोरेज पॉलिसीज़ को सूचीबद्ध/प्राप्त/सेट/संतुष्ट करें

:

□□□□□□□□□ □□□□□□ □□□ और आवश्यक लाइब्रेरीज़ प्राप्त करने के लिए आवश्यक क्लास पथ प्रिंट करता है □□□ फ़ाइल सिस्टम पर एक फ़ाइल सिस्टम कमांड चलाता है □□□□□□□ गणित □□□□□□ पर्यावरण चर प्रदर्शित करता है □□□□□□□ □□□□□□□□ से एक प्रतिनिधि टोकन प्राप्त करता है □□□□□□□ कॉन्फ़िगरेशन से कॉन्फ़िग मान प्राप्त करता है □□□□□□ उन समूहों को प्राप्त करता है जिनसे उपयोगकर्ता संबंधित हैं □□□□□□□□□□□□□□□□□□ वर्तमान उपयोगकर्ता द्वारा स्वामित्व वाले सभी स्नैपशॉटेबल डायरेक्टरीज़ को सूचीबद्ध करता है □□□□□□□□□□□□ एक डायरेक्टरी के दो स्नैपशॉट्स के बीच अंतर या वर्तमान डायरेक्टरी सामग्री और एक स्नैपशॉट के बीच अंतर दिखाता है □□□□□□□□ संस्करण प्रिंट करता है

:

□□□□□□□□□ क्लस्टर संतुलन उपयोगिता चलाएं □□□□□□□□□ □□□ डेटानोड चलाएं □□□□□□□□□□ □□□ राउटर चलाएं □□□□□□□□-□□□□□ दिए गए नोड पर डिस्क्स के बीच डेटा समान रूप से वितरित करें □□□□□□□ □□□□□□□ सर्वर चलाएं, □□□□ □□□□ गेटवे □□□□□□□□□□□□ □□□ जर्नलनोड चलाएं □□□□□ ब्लॉक प्रतिकृति को स्टोरेज प्रकारों के बीच ले जाने के लिए उपयोगिता चलाएं □□-□□□□□□ □□□ नेमनोड चलाएं □□□3 □□□ संस्करण 3 गेटवे चलाएं □□□□□□□ पोर्टमैप सेवा चलाएं □□□□□□□□□□□□□□□□□□ □□□ सेकेंडरी नेमनोड चलाएं □□□ बाहरी स्टोरेजपॉलिसीसंतुष्टि चलाएं □□□□ □□ फेलओवर कंट्रोलर डेमन चलाएं

□□□□□□□□□□, जब बिना पैरामीटर्स के या -□ के साथ इनवोक किया जाता है, तो मदद प्रिंट कर सकता है।

```python
from pyspark.sql import SparkSession

spark = SparkSession.builder.master("local[*]")\
          .config('spark.driver.bindAddress', '127.0.0.1')\
          .getOrCreate()
sc = spark.sparkContext
```

यह कोड एक □□□□□□□□□□□□□ बनाता है और उसे `spark` नाम से इनिशियलाइज़ करता है। `master("local[*]")` का उपयोग करके यह स्थानीय मशीन पर सभी उपलब्ध कोर का उपयोग करता है। `spark.driver.bindAddress` को `127.0.0.1` पर सेट

किया गया है, जो ड्राइवर को लोकलहोस्ट पर बाइंड करता है। अंत में, `getOrCreate()` मेथड का उपयोग करके □□□□□□□□□□□□□ को बनाया या पहले से मौजूद सत्र को प्राप्त किया जाता है। `sc` वेरिएबल □□□□□□□□□□□□□ को संदर्भित करता है, जो □□□□□ एप्लिकेशन का मुख्य प्रवेश बिंदु है।

```
text_file = sc.textFile("a.txt")
counts = text_file.flatMap(lambda line: line.split(" ")) \
             .map(lambda word: (word, 1)) \
             .reduceByKey(lambda a, b: a + b)
counts.saveAsTextFile("b.txt")
```

यह कोड □□□□□□ □□□□□ में एक सरल शब्द गणना (□□□□ □□□□□) प्रोग्राम है। यह `a.txt` फ़ाइल से पाठ पढ़ता है, प्रत्येक शब्द को गिनता है, और परिणाम को `b.txt` फ़ाइल में सहेजता है।

ध्यान दें कि `.config('spark.driver.bindAddress', '127.0.0.1')` सेट करना महत्वपूर्ण है। अन्यथा, आपको `Service 'sparkDriver' could not bind on a random free port. You may check whether configuring an appropriate binding address` जैसी त्रुटि मिल सकती है।

हालांकि, इस समय फिर से एक त्रुटि उत्पन्न हुई।

```
Caused by: org.apache.spark.api.python.PythonException: Traceback (most recent call last):
  File "/usr/local/lib/python3.9/site-packages/pyspark/python/lib/pyspark.zip/pyspark/worker.py", line 4
    raise Exception(("Python in worker has different version %s than that in " +
Exception:      Python       3.8              3.9      , PySpark   -                                  PYSP
```

यह दर्शाता है कि विभिन्न संस्करणों के `Python` चलाए गए हैं।

`.bash_profile` को संशोधित करें:

```
PYSPARK_PYTHON=/usr/local/Cellar/python@3.9/3.9.1_6/bin/python3
PYSPARK_DRIVER_PYTHON=/usr/local/Cellar/python@3.9/3.9.1_6/bin/python3
```

फिर भी वही त्रुटि आ रही है। कुछ जानकारी प्राप्त करने के बाद, यह संभव है कि `spark` चलाते समय यह पर्यावरण चर लोड नहीं हुआ हो, और टर्मिनल के डिफ़ॉल्ट पर्यावरण चर का उपयोग नहीं किया गया हो।

कोड में निम्नलिखित सेटिंग्स की आवश्यकता है:

```
import os
```

□□□□□ वातावरण सेट करें

□□.□□□□□□□□['□□□□□□□□_□□□□□□□'] = '/□□□/□□□□□□/□□□□□□□/□□□□□□□□3.9/3.9.1_6/□□□/□□□□□□□3'
□□.□□□□□□□□['□□□□□□□□_□□□□□□□_□□□□□□□'] = '/□□□/□□□□□□/□□□□□□□/□□□□□□□□3.9/3.9.1_6/□□□/□□□□□□□3'

```shell
$ python sc.py
21/03/11 02:54:52 WARN NativeCodeLoader:            native-hadoop               ...       ,    builtin-
Spark       log4j               : org/apache/spark/log4j-defaults.properties
           "WARN"
                   sc.setLogLevel(newLevel)         SparkR    , setLogLevel(newLevel)
PythonRDD[6] at RDD at PythonRDD.scala:53
```

इस समय `b.txt` फ़ाइल जनरेट हो गई है।

```
  b.txt
      _SUCCESS
      part-00000
      part-00001
```

खोलो।

```
$ cat b.txt/part-00000
('college', 1)
('two', 1)
('things', 2)
('worked', 1)
('on,', 1)
('of', 8)
('school,', 2)
('writing', 2)
('programming.', 1)
("didn't", 4)
('then,', 1)
('probably', 1)
```

```
('are:', 1)
('short', 1)
('awful.', 1)
('They', 1)
('plot,', 1)
('just', 1)
('characters', 1)
('them', 2)
...
```

सफलता! क्या यह परिचित नहीं लगता? यह ठीक वैसा ही है जैसा Hadoop उदाहरण में था।

```
$ cat output/*
1   dfsadmin
```

(यहाँ कोड ब्लॉक है, इसलिए इसे अनुवादित नहीं किया गया है।)

ये फ़ाइलें HDFS कहलाती हैं। यहाँ Spark का उपयोग शब्दों की गिनती करने के लिए किया गया है। कुछ ही पंक्तियों में, यह बहुत सुविधाजनक लगता है।

□□□□□□□□□□

अगला कदम Kubernetes को समझना है, जिसे k8s भी कहा जाता है, जहां बीच के 8 अक्षरों को 8 से संक्षिप्त किया गया है। यह एक ओपन-सोर्स सिस्टम है जो कंटेनर एप्लिकेशन्स को स्वचालित रूप से डिप्लॉय, स्केल और मैनेज करने के लिए उपयोग किया जाता है।

kubectl कमांड लाइन टूल का उपयोग □8□ क्लस्टर पर कुछ कमांड चलाने के लिए किया जाता है। इसका उपयोग एप्लिकेशन को डिप्लॉय करने, क्लस्टर संसाधनों को देखने और प्रबंधित करने, और लॉग्स देखने के लिए किया जा सकता है।

□□□□□□□□□ का उपयोग करके भी इंस्टॉल किया जा सकता है।

```
brew install kubectl
```

(नोट: यह एक कमांड है जिसे टर्मिनल में चलाया जाता है। इसे हिंदी में ट्रांसलेट करने की आवश्यकता नहीं है।)

लॉग आउटपुट:

```
==>            https://homebrew.bintray.com/bottles/kubernetes-cli-1.20.1.big_sur.bottle.tar.gz
==>            https://d29vzk4ow07wi7.cloudfront.net/0b4f08bd1d47cb913d7cd4571e3394c6747dfbad7ff114c558
######################################################################## 100.0%
==>          kubernetes-cli-1.20.1.big_sur.bottle.tar.gz
```

```
==>
Bash              :
  /usr/local/etc/bash_completion.d
==>
   /usr/local/Cellar/kubernetes-cli/1.20.1: 246    , 46.1MB
```

इंस्टॉल हो गया है।

```
$ kubectl version --client
Client Version: version.Info{Major:"1", Minor:"20", GitVersion:"v1.20.1", GitCommit:"c4d752765b3bbac223
```

```
$ kubectl
kubectl Kubernetes
```

अधिक जानकारी के लिए यहां जाएं: □□□□□://□□□□□□□□□□□.□□/□□□□/□□□□□□□□□□/□□□□□□□□/□□□□□□□□□/

बेसिक कमांड्स (शुरुआती): □□□□□□ फ़ाइल या □□□□□ से एक संसाधन बनाएं। □□□□□□ एक रिप्लिकेशन कंट्रोलर, सर्विस, डिप्लॉयमेंट या पॉड लें और इसे एक नई □□□□□□□□□□ सर्विस के रूप में एक्सपोज़ करें। □□□ क्लस्टर पर एक विशिष्ट इमेज चलाएं। □□□ ऑब्जेक्ट्स पर विशिष्ट फ़ीचर्स सेट करें।

बेसिक कमांड्स (इंटरमीडिएट): □□□□□□□ संसाधनों का दस्तावेज़ीकरण □□□ एक या अनेक संसाधनों को प्रदर्शित करें □□□□ सर्वर पर एक संसाधन को संपादित करें □□□□□□ फ़ाइलनाम, □□□□□, संसाधन और नाम, या संसाधन और लेबल चयनकर्ता द्वारा संसाधनों को हटाएं

□□□□□□ □□□□□□□□: □□□□□□□ किसी संसाधन के रोलआउट का प्रबंधन करें □□□□□ □□□□□□□□□□, □□□□□□□□□□ या □□□□□□□□□□ □□□□□□□□□□ के लिए एक नया आकार सेट करें □□□□□□□□ □□□□□□□□□□, □□□□□□□□□□, या □□□□□-□□□□□□□□□□□□□□□□ को स्वचालित रूप से स्केल करें

क्लस्टर प्रबंधन कमांड्स: □□□□□□□□□□ सर्टिफिकेट संसाधनों को संशोधित करें। □□□□□□□-□□□□ क्लस्टर की जानकारी प्रदर्शित करें। □□□ संसाधन (□□□/मेमोरी/स्टोरेज) उपयोग प्रदर्शित करें। □□□□□□ नोड को अनुसूचित नहीं करने योग्य के रूप में चिह्नित करें। □□□□□□□□ नोड को अनुसूचित करने योग्य के रूप में चिह्नित करें। □□□□□ रखरखाव की तैयारी में नोड को ड्रेन करें। □□□□□ एक या अधिक नोड्स पर टेंट्स को अपडेट करें।

समस्या निवारण और डिबगिंग कमांड: □□□□□□□□ किसी विशिष्ट संसाधन या संसाधनों के समूह का विवरण दिखाएं □□□□ पॉड में कंटेनर के लॉग प्रिंट करें □□□□□□ चल रहे कंटेनर से जुड़ें □□□□ कंटेनर में एक कमांड निष्पादित करें □□□□-□□□□□□□ एक या अधिक स्थानीय पोर्ट्स को पॉड में फॉरवर्ड करें □□□□□ □□□□□□□□□□ □□□ सर्वर के लिए एक प्रॉक्सी चलाएं □□ कंटेनर से और कंटेनर में फ़ाइलें और डायरेक्टरीज़ कॉपी करें □□□□ प्राधिकरण का निरीक्षण करें □□□□□ वर्कलोड और नोड्स के समस्या निवारण के लिए डिबगिंग सत्र बनाएं

उन्नत कमांड्स: □□□□ लाइव संस्करण और लागू किए जाने वाले संस्करण के बीच अंतर दिखाएं □□□□□ फ़ाइलनाम या □□□□□ के माध्यम से कॉन्फ़िगरेशन को संसाधन पर लागू करें □□□□□ संसाधन के फ़ील्ड को अपडेट करें □□□□□□□ फ़ाइलनाम या □□□□□ के माध्यम से

संसाधन को बदलें □□□□ प्रायोगिक: एक या कई संसाधनों पर विशिष्ट स्थिति की प्रतीक्षा करें। □□□□□□□□□□ एक निर्देशिका या दूरस्थ □□□ से एक □□□□□□□□□□□□□□□ लक्ष्य बनाएं।

सेटिंग्स कमांड्स: □□□□□ किसी संसाधन पर लेबल अपडेट करें □□□□□□□□□ किसी संसाधन पर एनोटेशन अपडेट करें □□□□□□□□□□□ निर्दिष्ट शेल (□□□□ या □□□) के लिए शेल पूर्णता कोड आउटपुट करें

अन्य कमांड्स: □□□-□□□□□□□□□□ सर्वर पर समर्थित □□□ संसाधनों को प्रिंट करें □□□-□□□□□□□□□ सर्वर पर समर्थित □□□ संस्करणों को "□□□□□/□□□□□□□□" के रूप में प्रिंट करें □□□□□□ □□□□□□□□□□□ फ़ाइलों को संशोधित करें □□□□□□ प्लगइन्स के साथ इंटरैक्ट करने के लिए उपयोगिताएँ प्रदान करें □□□□□□□□ क्लाइंट और सर्वर संस्करण जानकारी प्रिंट करें

उपयोग: □□□□□□□□ [फ्लैग्स] [विकल्प]

किसी दिए गए कमांड के बारे में अधिक जानकारी के लिए "□□□□□□□□ –□□□□" का उपयोग करें। सभी कमांड्स पर लागू होने वाले वैश्विक कमांड-लाइन विकल्पों की सूची के लिए "□□□□□□□□ □□□□□□□□" का उपयोग करें।

```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: nginx-deployment
spec:
  selector:
    matchLabels:
      app: nginx
  minReadySeconds: 5
  template:
    metadata:
      labels:
        app: nginx
    spec:
      containers:
      - name: nginx
        image: nginx:1.14.2
        ports:
        - containerPort: 80
```

यह □□□□ फ़ाइल एक □□□□□□□□□□□ □□□□□□□□□□□ को परिभाषित करती है जो `nginx` कंटेनर को डिप्लॉय करती है। यह □□-□□□□□□□□□ `nginx-deployment` नाम से जाना जाएगा और यह `nginx:1.14.2` इमेज का उपयोग करेगा। कंटेनर 80 पोर्ट पर

लिसन करेगा। minReadySeconds सेटिंग यह सुनिश्चित करती है कि कंटेनर कम से कम 5 सेकंड तक तैयार रहेगा।

```shell
$ kubectl apply -f simple_deployment.yaml
   localhost:8080                  -                          ?

$ kubectl cluster-info
```

क्लस्टर समस्याओं को और डीबग और डायग्नोस करने के लिए, '□□□□□□□□ □□□□□□□□-□□□□ □□□□' का उपयोग करें। सर्वर □□-□□□□□□□□:8080 से कनेक्शन अस्वीकार कर दिया गया - क्या आपने सही होस्ट या पोर्ट निर्दिष्ट किया है?

```shell
$ start.sh
Kubernetes         ...minikube    : v1.8.1
  : cbda04cf6bbe65e987ae52bb393c10099ab62014
* minikube v1.8.1 Ubuntu 18.04
*                   none
* localhost         (CPUs=2, Memory=2460MB, Disk=145651MB) ...
* OS     Ubuntu 18.04.4 LTS
```

□ □□□□□□□□□□□ □1.17.3 को □□□□□□□ 19.03.6 पर तैयार कर रहे हैं ...

- □□□□□□□□.□□□□□□□-□□□□□=/□□□/□□□□□□□□/□□□□□□□□/□□□□□□□.□□□□

□ □□□□□□□□□□□ लॉन्च कर रहे हैं ...
□ ऐडऑन्स सक्षम कर रहे हैं: □□□□□□□□-□□□□□□□□□□□□□□□, □□□□□□□□-□□□□□□□□□□□□□□
□ लोकल होस्ट वातावरण को कॉन्फ़िगर कर रहे हैं ...
□ हो गया! □□□□□□□□ अब "□□□□□□□□□" का उपयोग करने के लिए कॉन्फ़िगर है
□ '□□□□□□□□□□□' ऐडऑन सक्षम है □□□□□□□□□□□□ शुरू हो गया है

```shell
$ kubectl version --client
Client Version: version.Info{Major:"1", Minor:"20", GitVersion:"v1.20.1", GitCommit:"c4d752765b3bbac223
$ kubectl version
Client Version: version.Info{Major:"1", Minor:"20", GitVersion:"v1.20.1", GitCommit:"c4d752765b3bbac223
   localhost:8080                  -                          ?
```

दिलचस्प बात यह है कि `--client` विकल्प को जोड़ने पर कोई त्रुटि नहीं आई।

दस्तावेज़ के अनुसार, `Minikube` को पहले इंस्टॉल करना आवश्यक है।

```
$ brew install minikube
==> Downloading https://homebrew.bintray.com/bottles/minikube-1.16.0.big_sur.bottle.tar.gz
==> Downloading from https://d29vzk4ow07wi7.cloudfront.net/1b6d7d1b97b11b6b07e4fa531c2dc21770da290da9b28
######################################################################## 100.0%
==> Pouring minikube-1.16.0.big_sur.bottle.tar.gz
==> Caveats
Bash     (completion)          :
  /usr/local/etc/bash_completion.d
==> Summary
   /usr/local/Cellar/minikube/1.16.0: 8    , 64.6MB

$ minikube start
   minikube v1.16.0      11.2.2
   minikube 1.18.1      !         : https://github.com/kubernetes/minikube/releases/tag/v1.18.1
                    ,    : 'minikube config set WantUpdateNotification false'
```

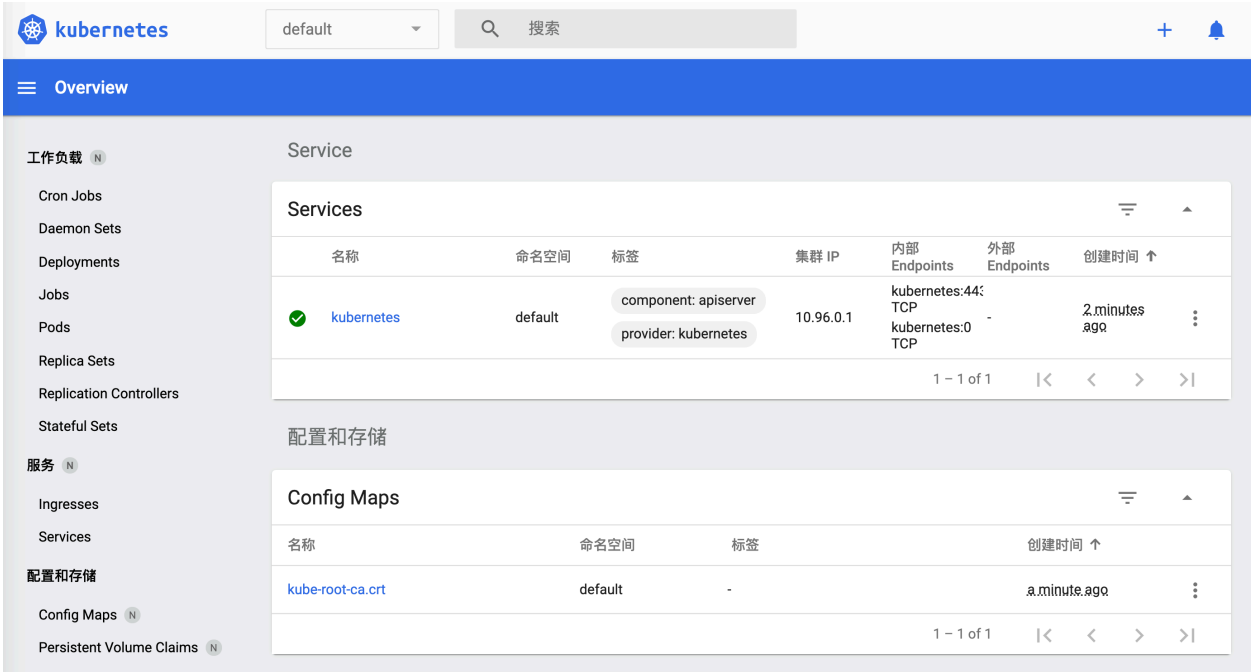
□ स्वचालित रूप से □□□□□□□□□□ ड्राइवर का चयन किया गया □ □□ बूट इमेज डाउनलोड हो रहा है ... > □□□□□□□□-□1.16.0.□□□.□□□256: 65 □ / 65 □ [————-] 100.00% ? □/□ 0□ > □□□□□□□□-□1.16.0.□□□: 212.62 □□□ / 212.62 □□□ [] 100.00% 5.32 □□□ □/□ 40□ □ कंट्रोल प्लेन नोड □□□□□□□□ को क्लस्टर □□□□□□□□ में शुरू किया जा रहा है □ □□□□□□□□□□ □1.20.0 प्रीलोड डाउनलोड हो रहा है ... > □□□□□□□□□-□□□□□□-□8□-□8-□1....: 491.00 □□□ / 491.00 □□□ 100.00% 7.52 □□□ □ □□□□□□□□□□ □□ बनाया जा रहा है (□□□□=2, □□□□□□=4000□□, □□□□=20000□□) ... □ इस □□ को □□□□□://□8□.□□□.□□ तक पहुंचने में समस्या हो रही है □ नई बाहरी इमेज को पुल करने के लिए, आपको एक प्रॉक्सी कॉन्फ़िगर करने की आवश्यकता हो सकती है: □□□□□://□□□□□□□□.□□□□.□8□.□□/□□□□/□□□□□□□□□/□□□□□□□□□/□□□□□/ □ □□□□□□□□□□ □1.20.0 को □□□□□□ 20.10.0 पर तैयार किया जा रहा है ... □ सर्टिफिकेट और कुंजियाँ जनरेट की जा रही हैं ... □ कंट्रोल प्लेन को बूट किया जा रहा है ... □ □□□□ नियम कॉन्फ़िगर किए जा रहे हैं ... □ □□□□□□□□□□ कंपोनेंट्स की जाँच की जा रही है... □ एडऑन्स सक्षम किए गए: □□□□□□□-□□□□□□□□□□, □□□□□□□-□□□□□□□□□□ □ हो गया! □□□□□□□ अब डिफ़ॉल्ट रूप से "□□□□□□□□" क्लस्टर और "□□□□□□□" नेमस्पेस का उपयोग करने के लिए कॉन्फ़िगर हो गया है

```shell
$ kubectl get po -A
NAMESPACE     NAME                               READY    STATUS     RESTARTS    AGE
kube-system   coredns-74ff55c5b-ndbcr            1/1              0           60s
```

```
kube-system   etcd-minikube                      0/1     0     74s
kube-system   kube-apiserver-minikube            1/1     0     74s
kube-system   kube-controller-manager-minikube   1/1     0     74s
kube-system   kube-proxy-g2296                   1/1     0     60s
kube-system   kube-scheduler-minikube            0/1     0     74s
kube-system   storage-provisioner                1/1     1     74s
```

minikube का डैशबोर्ड खोलने के लिए।

```
$ minikube dashboard
               ...
                    ...
               ...
                    ...
              http://127.0.0.1:50030/api/v1/namespaces/kubernetes-dashboard/services/http:kubernetes-d
```



□□□□□□ 1: □8□

कैसे बंद करें।

```
$ minikube
minikube                    Kubernetes
```

बेसिक कमांड्स: □□□□□ एक स्थानीय □□□□□□□□□□ क्लस्टर शुरू करता है □□□□□□ एक स्थानीय □□□□□□□□□□ क्लस्टर की स्थिति प्राप्त करता है □□□□ चल रहे स्थानीय □□□□□□□□□□ क्लस्टर को रोकता है □□□□□□ एक स्थानीय □□□□□□□□□□ क्लस्टर को हटाता है □□□□□□□□□□ □□□□□□□□□ क्लस्टर के अंदर चल रहे □□□□□□□□□□ डैशबोर्ड तक पहुंच प्रदान करता है □□□□□ □□-□□□□□□□□ को पॉज़ करता है □□□□□□□ □□□□□□□□□□ को अनपॉज़ करता है

□□□□□□ □□□□□□□□: □□□□□□-□□□ □□□□□□□□ के □□□□□□ डेमन का उपयोग करने के लिए वातावरण को कॉन्फ़िगर करें □□□□□□-□□□ □□□□□□□□ के □□□□□□ सेवा का उपयोग करने के लिए वातावरण को कॉन्फ़िगर करें □□□□□ □□□□□□□□ में एक स्थानीय छवि को जोड़ें, हटाएं, या पुश करें

कॉन्फ़िगरेशन और प्रबंधन कमांड: □□□□□□ मिनिक्यूब ऐडऑन को सक्षम या अक्षम करें □□□□□□ स्थायी कॉन्फ़िगरेशन मानों को संशोधित करें □□□□□□□ वर्तमान प्रोफाइल (क्लस्टर) प्राप्त करें या सूचीबद्ध करें □□□□□□-□□□□□□□ □□ या पोर्ट परिवर्तन के मामले में □□□□-□□□□□□ को अपडेट करें

नेटवर्किंग और कनेक्टिविटी कमांड्स: □□□□□□□ किसी सेवा से कनेक्ट करने के लिए एक □□□ रिटर्न करता है □□□□□□ □□□□□□□-□□□□□ सेवाओं से कनेक्ट करता है

उन्नत कमांड्स: □□□□□ निर्दिष्ट डायरेक्टरी को □□□□□□□□ में माउंट करता है □□□ □□□□□□□□ वातावरण में लॉग इन करें (डिबगिंग के लिए) □□□□□□□ क्लस्टर संस्करण से मेल खाने वाले □□□□□□□ बाइनरी को चलाएं □□□□ अतिरिक्त नोड्स को जोड़ें, हटाएं, या सूचीबद्ध करें

समस्या निवारण कमांड्स: □□□-□□□ निर्दिष्ट नोड की □□□ पहचान कुंजी पथ प्राप्त करें □□□-□□□□ निर्दिष्ट नोड की □□□ होस्ट कुंजी प्राप्त करें □□ निर्दिष्ट नोड का □□ पता प्राप्त करें □□□□ स्थानीय □□□□□□□□□□ क्लस्टर को डीबग करने के लिए लॉग्स लौटाएं □□□□□□-□□□□□ वर्तमान और नवीनतम संस्करण संख्या प्रिंट करें □□□□□□□ □□□□□□□□ का संस्करण प्रिंट करें

अन्य कमांड: □□□□□□□□□□ किसी शेल के लिए कमांड पूर्णता उत्पन्न करें

किसी दिए गए कमांड के बारे में अधिक जानकारी के लिए "□□□□□□□□ -□□□□" का उपयोग करें।

```
            `minikube stop`
```

`kubernetes`   ,

```shell
$ kubectl cluster-info
Kubernetes       https://192.168.99.100:8443
KubeDNS https://192.168.99.100:8443/api/v1/namespaces/kube-system/services/kube-dns:dns/proxy
```

क्लस्टर समस्याओं को और डीबग और निदान करने के लिए, '□□□□□□□ □□□□□□□-□□□□ □□□□' का उपयोग करें।

    `https://192.168.99.100:8443`   ,   :

```json
```

```
{
  "kind": "Status",
  "apiVersion": "v1",
  "metadata": {

  },
  "status": "Failure",
  "message": "    :       \"system:anonymous\"    \"/\"            ",
  "reason": "Forbidden",
  "details": {

  },
  "code": 403
}
```


https://192.168.99.100:8443/api/v1/namespaces/kube-system/services/kube-dns:dns/proxy पर जाएं:


```
{
  "kind": "Status",
  "apiVersion": "v1",
  "metadata": {

  },
  "status": "Failure",
  "message": "    \"kube-dns:dns\"      :       \"system:anonymous\" API    \"\"       \"kube-system\"
  "reason": "Forbidden",
  "details": {
    "name": "kube-dns:dns",
    "kind": "services"
  },
  "code": 403
}
```


आइए अभी उस कॉन्फ़िगरेशन को आज़माएं।

```
$ kubectl apply -f simple_deployment.yaml
deployment.apps/nginx-deployment
```

थोड़ी समस्या है। हालांकि, यहां तक हमने `kubernetes` को चला लिया है। पहले इसे समाप्त करते हैं। बाद में फिर खेलेंगे।

```
$ minikube stop
   "minikube"              ...
   1
```

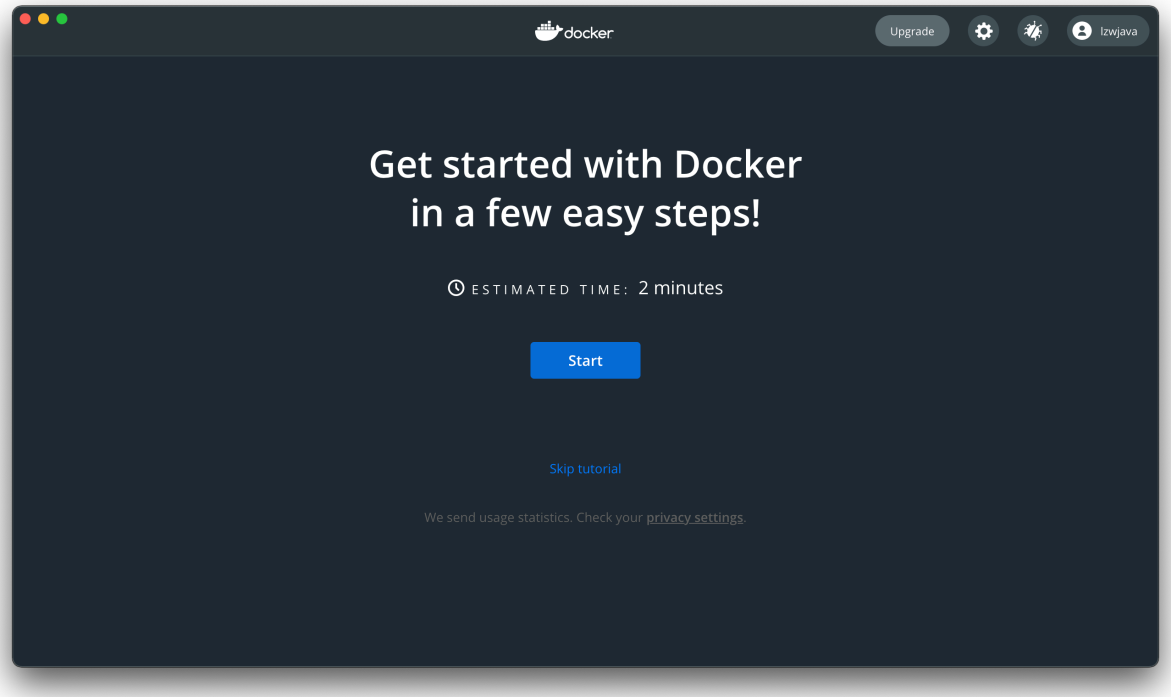
समाप्ति की जाँच करें।

```
w$ minikube dashboard

              ,    : "minikube start"
```

□□□□□□

`Docker` एक कंटेनर प्लेटफॉर्म भी है, जो आधुनिक एप्लिकेशन बनाने, साझा करने और चलाने में तेजी लाने में मदद करता है। आधिकारिक वेबसाइट से एप्लिकेशन डाउनलोड करें।



□□□□□□ 2: □□□□□□

क्लाइंट का उपयोग करते समय थोड़ा लैग हो रहा है। चलो कमांड लाइन का उपयोग करते हैं।

```
$ docker
```

उपयोग: □□□□□□ [विकल्प] कमांड

कंटेनरों के लिए एक स्वयं-पर्याप्त रनटाइम

विकल्प: –□□□□□□ □□□□□□ क्लाइंट कॉन्फ़िग फ़ाइलों का स्थान (डिफ़ॉल्ट "/□□□□□/□□□/.□□□□□□") -□, –□□□□□□□ □□□□□□ डेमन से कनेक्ट करने के लिए उपयोग करने वाले कॉन्टेक्स्ट का नाम (□□□□□□_□□□□ □□□ □□□ और "□□□□□□ □□□-□□□□ □□□" के साथ सेट डिफ़ॉल्ट कॉन्टेक्स्ट को ओवरराइड करता है) -□, –□□□□□ डिबग मोड सक्षम करें -□, –□□□□ □□□□ कनेक्ट करने के लिए डेमन सॉकेट(□) -□, –□□□-□□□□□ □□□□□□ लॉगिंग स्तर सेट करें ("□□□□□"|"□□□□"|"□□□□"|"□□□□□"|"□□□□□") (डिफ़ॉल्ट "□□□□") –□□□ □□□ का उपयोग करें; –□□□□□□□□□ द्वारा निहित –□□□□□□□□□ □□□□□□ केवल इस □□ द्वारा हस्ताक्षरित प्रमाणपत्रों पर भरोसा करें (डिफ़ॉल्ट "/□□□□□/□□□/.□□□□□□/□□.□□□") –□□□□□□□ □□□□□□ □□□ प्रमाणपत्र फ़ाइल का पथ (डिफ़ॉल्ट "/□□□□□/□□□/.□□□□□□/□□□□.□□□") –□□□□□□ □□□□□□ □□□ कुंजी फ़ाइल का पथ (डिफ़ॉल्ट "/□□□□□/□□□/.□□□□□□/□□□.□□□") –□□□□□□□□□ □□□ का उपयोग करें और रिमोट को सत्यापित करें -□, –□□□□□□□ संस्करण जानकारी प्रिंट करें और बाहर निकलें

प्रबंधन कमांड्स: □□□* □□□□□□ □□□ (□□□□□□ □□□., □0.9.1-□□□□3) □□□□□□□ बिल्ड्स प्रबंधित करें □□□□□□* □□□□□-□□□ के साथ बिल्ड करें (□□□□□□ □□□., □0.5.1-□□□□□□) □□□□□□ □□□□□□ कॉन्फ़िगरेशन प्रबंधित करें □□□□□□□□□ कंटेनर प्रबंधित करें □□□□□□□ कॉन्टेक्स्ट प्रबंधित करें □□□□□ इमेज प्रबंधित करें □□□□□□□□ □□□□□□ इमेज मैनिफेस्ट और मैनिफेस्ट सूचियाँ प्रबंधित करें □□□□□□□ नेटवर्क प्रबंधित करें □□□□ □□□□□ नोड्स प्रबंधित करें □□□□□□ प्लगइन्स प्रबंधित करें □□□□* □□□□□□ □□□□ (□□□□□□ □□□., □0.5.0) □□□□□□ □□□□□□ सीक्रेट्स प्रबंधित करें □□□□□□□ सर्विसेज प्रबंधित करें □□□□□ □□□□□□ स्टैक्स प्रबंधित करें □□□□□ □□□□□ प्रबंधित करें □□□□□□ □□□□□□ प्रबंधित करें □□□□□ □□□□□□ इमेज पर ट्रस्ट प्रबंधित करें □□□□□□ वॉल्यूम प्रबंधित करें

कमांड्स: □□□□□□ एक चल रहे कंटेनर से स्थानीय मानक इनपुट, आउटपुट और त्रुटि स्ट्रीम को जोड़ें □□□□□ □□□□□□□□□□ से एक इमेज बनाएं □□□□□□ कंटेनर में हुए परिवर्तनों से एक नई इमेज बनाएं □□ कंटेनर और स्थानीय फाइल सिस्टम के बीच फाइल/फोल्डर कॉपी करें □□□□□□ एक नया कंटेनर बनाएं □□□□ कंटेनर के फाइल सिस्टम पर फाइल या डायरेक्टरी में हुए परिवर्तनों का निरीक्षण करें □□□□□□ सर्वर से रियल टाइम इवेंट्स प्राप्त करें □□□□ एक चल रहे कंटेनर में कमांड चलाएं □□□□□□ कंटेनर के फाइल सिस्टम को □□□ आर्काइव के रूप में निर्यात करें □□□□□□□ एक इमेज का इतिहास दिखाएं □□□□□□ इमेज की सूची दिखाएं □□□□□□ एक □□□□□□□ से सामग्री आयात करके फाइल सिस्टम इमेज बनाएं □□□□ सिस्टम-व्यापी जानकारी प्रदर्शित करें □□□□□□□ □□□□□□ ऑब्जेक्ट्स पर निम्न-स्तरीय जानकारी दिखाएं □□□□ एक या अधिक चल रहे कंटेनर को समाप्त करें □□□□ □□□ आर्काइव या □□□□□ से एक इमेज लोड करें □□□□□ □□□□□□ रजिस्ट्री में लॉग इन करें □□□□□□ □□□□□□ रजिस्ट्री से लॉग आउट करें □□□□ कंटेनर के लॉग प्राप्त करें □□□□□ एक या अधिक कंटेनर के सभी प्रक्रियाओं को रोकें □□□□ कंटेनर के पोर्ट मैपिंग या एक विशिष्ट मैपिंग की सूची दिखाएं □□ कंटेनर की सूची दिखाएं □□□□ रजिस्ट्री से एक इमेज या रिपॉजिटरी डाउनलोड करें □□□□ रजिस्ट्री में एक इमेज या रिपॉजिटरी अपलोड करें □□□□□□ कंटेनर का नाम बदलें □□□□□□□ एक या अधिक कंटेनर को पुनरारंभ करें □□ एक या अधिक कंटेनर हटाएं □□□ एक या अधिक इमेज हटाएं □□□ एक नए कंटेनर में कमांड चलाएं □□□□ एक या अधिक इमेज को □□□ आर्काइव के रूप में सहेजें (डिफ़ॉल्ट रूप से □□□□□□ पर स्ट्रीम किया जाता है) □□□□□□ □□□□□□ □□□ पर इमेज खोजें □□□□□ एक या अधिक रुके हुए कंटेनर शुरू करें □□□□□ कंटेनर(र) के संसाधन उपयोग सांख्यिकी का लाइव स्ट्रीम प्रदर्शित करें □□□□ एक या अधिक चल रहे कंटेनर को रोकें □□□ एक टैग □□□□□□_□□□□□ बनाएं जो □□□□□□_□□□□□ को संदर्भित करता है □□□ कंटेनर की चल रही प्रक्रियाओं को प्रदर्शित करें □□□□□□□ एक या अधिक कंटेनर के सभी प्रक्रियाओं को अनपॉज़ करें □□□□□□ एक या अधिक कंटेनर की कॉन्फ़िगरेशन अपडेट करें □□□□□□□ □□□□□□ संस्करण जानकारी दिखाएं □□□□ एक या अधिक कंटेनर के रुकने तक ब्लॉक करें, फिर उनके एग्जिट कोड प्रिंट

करें

किसी कमांड के बारे में अधिक जानकारी के लिए '□□□□□□ □□□□□□□ –□□□□' चलाएं।

□□□□□□ के बारे में अधिक मदद पाने के लिए, हमारे गाइड्स देखें: □□□□□://□□□□.□□□□□□.□□□/□□/□□□□□□/

```shell
$ docker run -d -p 80:80 docker/getting-started
'docker/getting-started:latest'
latest: docker/getting-started
aad63a933944:
b14da7a62044:
343784d40d66:
6f617e610986:
Digest: sha256:d2c4fb0641519ea208f20ab03dc40ec2a5a53fdfbccca90bef14f870158ed577
Status: docker/getting-started
815f13fc8f99f6185257980f74c349e182842ca572fe60ff62cbb15641199eaf
docker:             :           : listen tcp 0.0.0.0:80: bind:
```

पोर्ट बदलें।

```
$ docker run -d -p 8080:80 docker/getting-started
45bb95fa1ae80adc05cc498a1f4f339c45c51f7a8ae1be17f5b704853a5513a5
```

ब्राउज़र खोलें, यह दर्शाता है कि हमने `docker` को सफलतापूर्वक चला दिया है।

कंटेनर को रोकें। पहले प्राप्त किए गए `ID` का उपयोग करें।

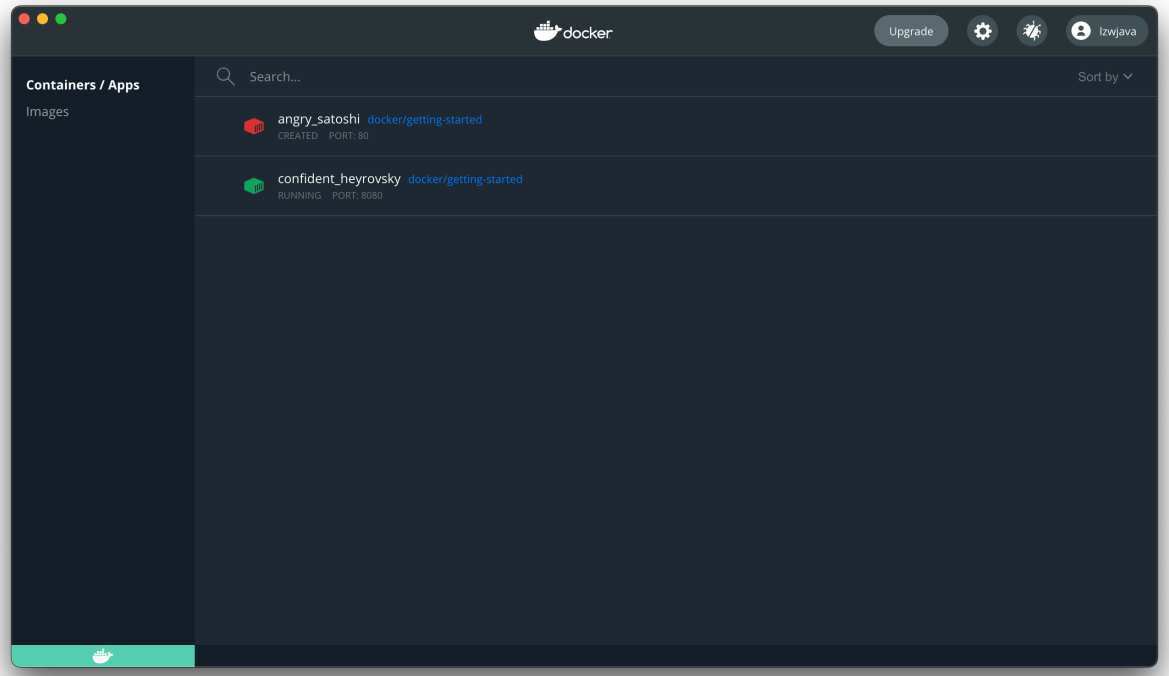
```
$ docker stop 45bb95fa1ae80adc05cc498a1f4f339c45c51f7a8ae1be17f5b704853a5513a5
45bb95fa1ae80adc05cc498a1f4f339c45c51f7a8ae1be17f5b704853a5513a5
```

इस समय वेबसाइट खोलना संभव नहीं था।

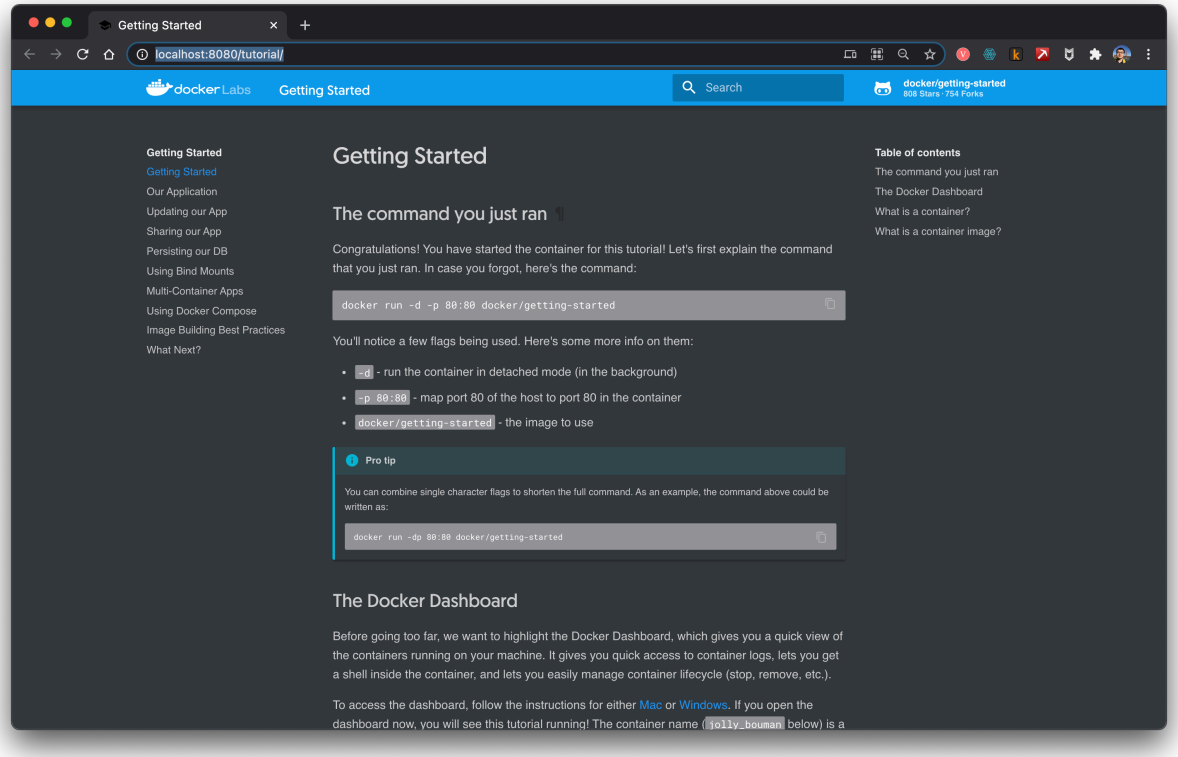
इससे पता चलता है कि `docker` एक वर्चुअल मशीन की तरह है।

□□□□□

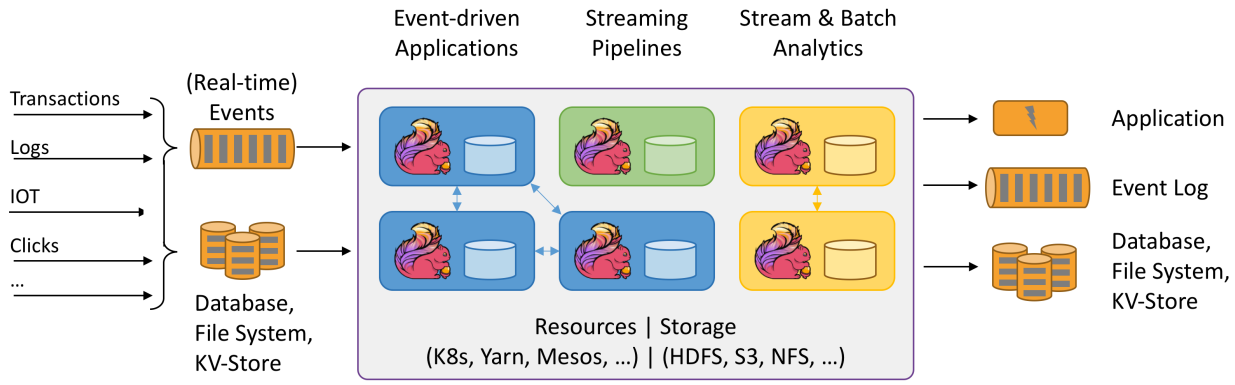
वेबसाइट खोलें।


docker
Upgrade
lzwjava
Containers / Apps
Images
Search...
Sort by
angry_satoshi docker/getting-started
CREATED PORT: 80
confident_heyrovsky docker/getting-started
RUNNING PORT: 8080

□□□□□□□ 3: □□□□□□□_□□□


Getting Started
localhost:8080/tutorial/
docker Labs
Getting Started
Search
docker/getting-started
808 Stars · 754 Forks
Getting Started
Getting Started
Our Application
Updating our App
Sharing our App
Persisting our DB
Using Bind Mounts
Multi-Container Apps
Using Docker Compose
Image Building Best Practices
What Next?
Getting Started
The command you just ran
Congratulations! You have started the container for this tutorial! Let's first explain the command that you just ran. In case you forgot, here's the command:
docker run -d -p 80:80 docker/getting-started
You'll notice a few flags being used. Here's some more info on them:
-d - run the container in detached mode (in the background)
-p 80:80 - map port 80 of the host to port 80 in the container
docker/getting-started - the image to use
Pro tip
You can combine single character flags to shorten the full command. As an example, the command above could be written as:
docker run -dp 80:80 docker/getting-started
The Docker Dashboard
Before going too far, we want to highlight the Docker Dashboard, which gives you a quick view of the containers running on your machine. It gives you quick access to container logs, lets you get a shell inside the container, and lets you easily manage container lifecycle (stop, remove, etc.).
To access the dashboard, follow the instructions for either Mac or Windows. If you open the dashboard now, you will see this tutorial running! The container name (jolly_bouman below) is a
Table of contents
The command you just ran
The Docker Dashboard
What is a container?
What is a container image?

□□□□□□□ 4: ब्राउज़र



□□□□□□□ 5: □□□□□-□□□□-□□□□□□□□

Flink को डेटा स्ट्रीम के Stateful कंप्यूटेशन के रूप में जाना जाता है। Stateful का क्या मतलब है? अभी तक समझ में नहीं आया। ऊपर दिया गया चित्र बहुत दिलचस्प है। आइए इसे आज़माएं।

यह कहा जाता है कि □□□□ वातावरण की आवश्यकता है।

```
$ java -version
java version "1.8.0_151"
Java(TM) SE Runtime Environment (build 1.8.0_151-b12)
Java HotSpot(TM) 64-Bit Server VM (build 25.151-b12, mixed mode)
```

आधिकारिक वेबसाइट से नवीनतम संस्करण flink-1.12.2-bin-scala_2.11.tar डाउनलोड करें।

```
$ ./bin/start-cluster.sh

   lzwjava
   lzwjava

$ ./bin/flink run examples/streaming/WordCount.jar
                WordCount
                  --input
      stdout                                --output
JobID 60f37647c20c2a6654359bd34edab807

JobID 60f37647c20c2a6654359bd34edab807
      : 757

$ tail log/flink-*-taskexecutor-*.out
```

```
(nymph,1)
(in,3)
(thy,1)
(orisons,1)
(be,4)
(all,2)
(my,1)
(sins,1)
(remember,1)
(d,4)

$ ./bin/stop-cluster.sh
          (pid: 41812)      lzwjava
```

हाँ, सफलतापूर्वक शुरुआत की। यह देखा जा सकता है कि यह `Spark` के समान है।

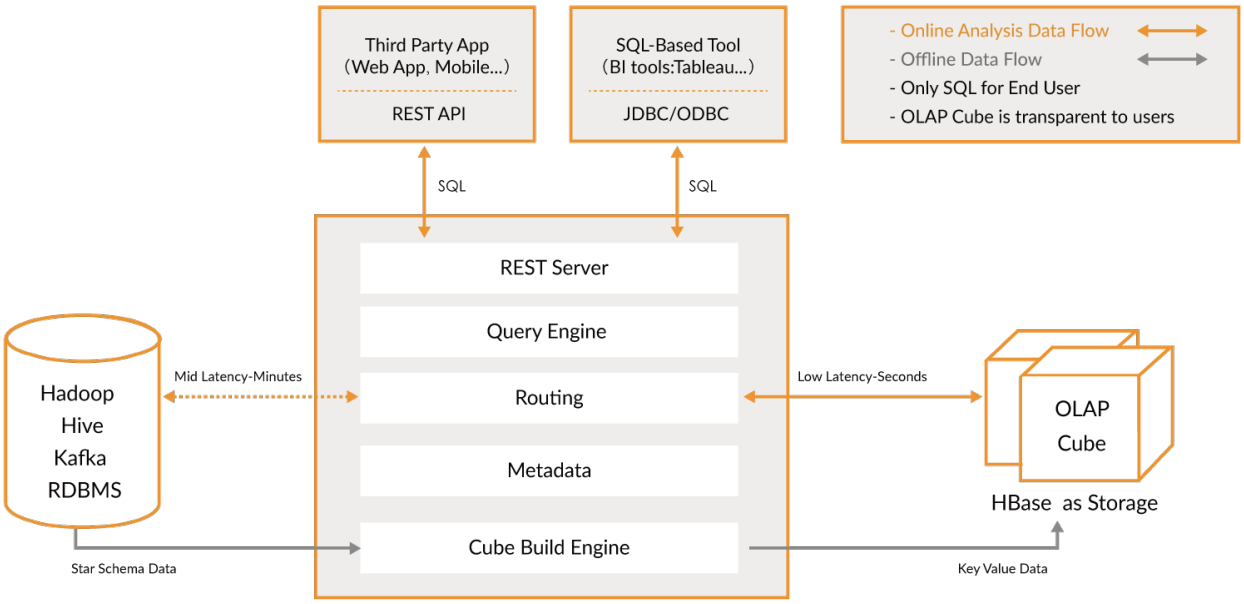
काइलिन

आधिकारिक वेबसाइट खोलने के लिए।

> □□□□□□ □□□□□□ एक ओपन सोर्स, वितरित विश्लेषणात्मक डेटा वेयरहाउस है जो बिग डेटा के लिए डिज़ाइन किया गया है; यह बिग डेटा युग में □□□□ (ऑनलाइन विश्लेषणात्मक प्रसंस्करण) क्षमता प्रदान करने के लिए बनाया गया था। □□□□□□ और □□□□□ पर मल्टी-डायमेंशनल क्यूब और प्रीकैलकुलेशन तकनीक को नवीनीकृत करके, □□□□□ लगातार बढ़ते डेटा वॉल्यूम के बावजूद लगभग स्थिर क्वेरी गति प्राप्त करने में सक्षम है। क्वेरी विलंबता को मिनटों से सब-सेकंड तक कम करके, □□□□□ ऑनलाइन एनालिटिक्स को बिग डेटा में वापस लाता है।
>
> □□□□□□ □□□□□□ आपको 3 चरणों में सब-सेकंड लेटेंसी पर अरबों पंक्तियों को क्वेरी करने की सुविधा देता है।
>
> 1. □□□□□□ पर एक □□□□/□□□□□□□□□ □□□□□□ की पहचान करें।
> 2. पहचाने गए टेबल्स से □□□□ बनाएं।
> 3. □□□□-□□□ का उपयोग करके क्वेरी करें और □□□□, □□□□ या □□□□□□□ □□□ के माध्यम से सब-सेकंड में परिणाम प्राप्त करें।

यह मूल रूप से बड़े डेटा के विश्लेषण की एक परत है। इसका उपयोग करके आप बहुत तेज़ी से खोज सकते हैं। यह एक पुल की तरह काम करता है।

दुर्भाग्य से, यह वर्तमान में केवल `Linux` वातावरण में उपयोग किया जा सकता है। बाद में इसे फिर से संभालूंगा।



□□□□□□□ 6: □□□□□_□□□□□□□□

□□□□□□□□

यह भी एक प्रकार का डेटाबेस है। इसे इंस्टॉल करके देखें।

```
$ brew tap mongodb/brew
==> mongodb/brew
'/usr/local/Homebrew/Library/Taps/mongodb/homebrew-brew'              ...
remote:                : 63,
remote:          : 100% (63/63),
remote:                 : 100% (62/62),
remote:   566 (   21),  :         6 (   1),  - :         503
             : 100% (566/566), 121.78 KiB | 335.00 KiB/s,
            : 100% (259/259),
11     (39    , 196.2KB)

$ brew install mongodb-community@4.4
==> mongodb/brew   mongodb-community
==> https://fastdl.mongodb.org/tools/db/mongodb-database-tools-macos-x86_64-100.3.0.zip
######################################################################## 100.0%
==> https://fastdl.mongodb.org/osx/mongodb-macos-x86_64-4.4.3.tgz
######################################################################## 100.0%
==> mongodb/brew/mongodb-community                    : mongodb-database-tools
```

```
==> mongodb/brew/mongodb-community     : mongodb-database-tools
  : `brew link`
        ,     /usr/local
bin/bsondump
    /usr/local/bin/bsondump
mongodb                          :
  brew unlink mongodb
```

लिंक को फोर्स करने और सभी संघर्ष करने वाली फ़ाइलों को ओवरराइट करने के लिए: □□□□ □□□□ –□□□□□□□□□□ □□□□□□□□-□□□□□□□□□-□□□□□□

सभी फ़ाइलों को सूचीबद्ध करने के लिए जिन्हें हटाया जाएगा: □□□□ □□□□ –□□□□□□□□□□ –□□□-□□□ □□□□□□□□-□□□□□□□□□-□□□□□□

संभावित टकराव वाली फ़ाइलें हैं: /□□□/□□□□□□/□□□□/□□□□□□□□□ -> /□□□/□□□□□□/□□□□□□□/□□□□□□□□/3.0.7/□□□/□□□□□□□□□ /□□□/□□□□□□/□□□□/□□□□□□□□□□ -> /□□□/□□□□□□/□□□□□□□/□□□□□□□□/3.0.7/□□□/□□□□□□□□□□ /□□□/□□□□□□/□□□□/□□□□□□□□□□□ -> /□□□/□□□□□□/□□□□□□□/□□□□□□□□/3.0.7/□□□/□□□□□□□□□□□□ /□□□/□□□□□□/□□□□/□□□□□□□□□□□ -> /□□□/□□□□□□/□□□□□□□/□□□□□□□□/3.0.7/□□□/□□□□□□□□□□□ /□□□/□□□□□□/□□□□/□□□□□□□□□□□□ -> /□□□/□□□□□□/□□□□□□□/□□□□□□□□/□□□□□□□□/3.0.7/□□□/□□□□□□□□□□□□ /□□□/□□□□□□/□□□□/□□□□□□□□□□ -> /□□□/□□□□□□/□□□□□□□/□□□□□□□□/3.0.7/□□□/□□□□□□□□□□ /□□□/□□□□□□/□□□□/□□□□□□□□□ -> /□□□/□□□□□□/□□□□□□□/□□□□□□□□/3.0.7/□□□/□□□□□□□□□ ==> सारांश □ /□□□/□□□□□□/□□□□□□□/□□□□□□□□-□□□□□□□□□-□□□□□□/100.3.0: 13 फ़ाइलें, 154□□, 11 सेकंड में बनाई गईं ==> □□□□□□□□/□□□□/□□□□□□□□-□□□□□□□□□□ स्थापित कर रहा है त्रुटि: `brew link` चरण सफलतापूर्वक पूरा नहीं हुआ फ़ॉर्मूला बन गया, लेकिन /□□□/□□□□□ में सिमलिंक नहीं किया गया है □□□/□□□□□ को सिमलिंक नहीं किया जा सका लक्ष्य /□□□/□□□□□/□□□/□□□□□ □□□□□□□□ का एक सिमलिंक है। आप इसे अनलिंक कर सकते हैं: □□□□ □□□□□□□ □□□□□□□□

लिंक को फोर्स करने और सभी संघर्ष करने वाली फ़ाइलों को ओवरराइट करने के लिए: □□□□ □□□□ –□□□□□□□□□□ □□□□□□□□-□□□□□□□□□□

सभी फ़ाइलों को सूचीबद्ध करने के लिए जो हटा दी जाएंगी: □□□□ □□□□ –□□□□□□□□□□ –□□□-□□□ □□□□□□□□-□□□□□□□□□□

संभावित संघर्ष करने वाली फ़ाइलें हैं: /□□□/□□□□□□/□□□/□□□□□□ -> /□□□/□□□□□□/□□□□□□□/□□□□□□□□/3.0.7/□□□/□□□□□□ /□□□/□□□□□□/□□□/□□□□□□□ -> /□□□/□□□□□□/□□□□□□□/□□□□□□□□/3.0.7/□□□/□□□□□□□ /□□□/□□□□□□/□□□/□□□□□□□ -> /□□□/□□□□□□/□□□□□□□/□□□□□□□□/3.0.7/□□□/□□□□□□□ ==> सावधानियाँ □□□□□□□□ को □□□□□□□□/□□□□/□□□□□□□□-□□□□□□□□□□ को अभी शुरू करने और लॉगिन पर पुनः आरंभ करने के लिए: □□□□ □□□□□□□□□ □□□□□□ □□□□□□□□/□□□□/□□□□□□□□-□□□□□□□□□□ या, यदि आपको बैकग्राउंड सेवा की आवश्यकता/इच्छा नहीं है तो आप सिर्फ चला सकते हैं: □□□□□□□ –□□□□□□□ /□□□□/□□□□□□/□□□□/□□□□□□□.□□□□ ==> सारांश □ /□□□□/□□□□□□/□□□□□□□/□□□□□□□□-□□□□□□□□□□/4.4.3: 11 फ़ाइलें, 156.8□□, 10 सेकंड में बनाया गया ==> सावधानियाँ ==> □□□□□□□□-□□□□□□□□□□ □□□□□□□□ को □□□□□□□□/□□□□/□□□□□□□□-□□□□□□□□□□ को अभी शुरू करने और लॉगिन पर पुनः आरंभ करने के लिए: □□□□ □□□□□□□□□ □□□□□□ □□□□□□□□/□□□□/□□□□□□□□-□□□□□□□□□□ या, यदि आपको बैकग्राउंड सेवा की आवश्यकता/इच्छा नहीं है तो आप सिर्फ चला सकते हैं: □□□□□□□ –□□□□□□□

/□□□/□□□□□/□□□/□□□□□□.□□□□

```shell
$ brew unlink mongodb
/usr/local/Cellar/mongodb/3.0.7                ... 11

$ mongod --version
db version v4.4.3
Build Info: {
    "version": "4.4.3",
    "gitVersion": "913d6b62acfbb344dde1b116f4161360acd8fd13",
    "modules": [],
    "allocator": "system",
    "environment": {
        "distarch": "x86_64",
        "target_arch": "x86_64"
    }
}
```

फिर mongod चलाकर □□□□□ डेटाबेस सर्वर शुरू करें। हालांकि, पहली बार शुरू करते समय यह कहा गया कि /data/db मौजूद नहीं है। हम एक डायरेक्टरी बनाते हैं, ~/mongodb, जहां डेटाबेस फाइलों को सहेजा जाएगा।

```
$ mongod --dbpath ~/mongodb
```

(यह कोड ब्लॉक है, इसे अनुवादित नहीं किया जाना चाहिए।)

आउटपुट होगा:

```
{"t":{"$date":"2021-03-11T18:17:32.838+08:00"},"s":"I",  "c":"CONTROL",  "id":23285,   "ctx":"main","ms
{"t":{"$date":"2021-03-11T18:17:32.842+08:00"},"s":"W",  "c":"ASIO",     "id":22601,   "ctx":"main","ms
{"t":{"$date":"2021-03-11T18:17:32.842+08:00"},"s":"I",  "c":"NETWORK",  "id":4648602, "ctx":"main","ms
{"t":{"$date":"2021-03-11T18:17:32.842+08:00"},"s":"I",  "c":"STORAGE",  "id":4615611, "ctx":"initandli
{"t":{"$date":"2021-03-11T18:17:32.842+08:00"},"s":"I",  "c":"CONTROL",  "id":23403,   "ctx":"initandli
{"t":{"$date":"2021-03-11T18:17:32.843+08:00"},"s":"I",  "c":"CONTROL",  "id":51765,   "ctx":"initandli
...
```

देखा जा सकता है कि सभी JSON प्रारूप में हैं। □□□□□□□□ में सभी डेटा फ़ाइलें JSON प्रारूप में सहेजी जाती हैं। इसके बाद, एक और टर्मिनल टैब खोलें।

```
$ mongo
MongoDB shell version v4.4.3
connecting to: mongodb://127.0.0.1:27017/?compressors=disabled&gssapiServiceName=mongodb
Implicit session: session { "id" : UUID("4f55c561-70d3-4289-938d-4b90a284891f") }
MongoDB server version: 4.4.3
---
                              :
        2021-03-11T18:17:33.743+08:00:
        2021-03-11T18:17:33.743+08:00:                                                --bind_ip <addr
        2021-03-11T18:17:33.743+08:00:    rlimits
        2021-03-11T18:17:33.743+08:00:         currentValue: 4864
        2021-03-11T18:17:33.743+08:00:         recommendedMinimum: 64000
---
---
        MongoDB        -                  ,                    (      , CPU,         ,   )

          MongoDB             ,          URL                   ,         URL         MongoDB

                 ,            : `db.enableFreeMonitoring()`
                          ,            : `db.disableFreeMonitoring()`
```

इसके बाद, आप डेटा इन्सर्ट करने और डेटा क्वेरी करने का प्रयास कर सकते हैं।

```
> db.inventory.insertOne(
...    { item: "canvas", qty: 100, tags: ["cotton"], size: { h: 28, w: 35.5, uom: "cm" } }
... )
{
    "acknowledged" : true,
    "insertedId" : ObjectId("6049ef91b653541cf355facb")
}
>
> db.inventory.find()
{ "_id" : ObjectId("6049ef91b653541cf355facb"), "item" : "canvas", "qty" : 100, "tags" : [ "cotton" ],
```

अंत में

यहाँ तक। बाद में हम अन्य टूल्स पर हाथ आज़माएंगे। हम ये सब कर रहे हैं, इसका क्या मतलब है। शायद पहले एक संरचना बनाना है। हर काम की शुरुआत मुश्किल होती है, और हमने शुरुआत में ही इन सभी को एक बार में कर लिया। इससे हमें आत्मविश्वास मिला है, और अब आगे, हम इन सॉफ़्टवेयर को और अधिक एक्सप्लोर करेंगे।

अभ्यास

- छात्र ऊपर दिए गए तरीके से समान रूप से एक्सप्लोर करें।