

Za one koji žele znati više!



Napredni modeli i baze podataka - upute za vježbu
(uz predavanje 10. Tokovi podataka):

Tokovi podataka i Apache Spark

Akadska godina 2015./2016.

1. Uvod

- Preporučljivo je da se student prije početka izvođenja vježbe upozna s osnovnim pojmovima vezanim uz tokove podataka i distribuirano procesiranje podataka sa sustavom Apache Spark (slajdovi uz predavanje „10. Tokovi podataka“)
- Na vašem virtualnom računalu već je instaliran sustav Apache Spark 1.5.1 u direktoriju /root/spark
- Pokrenite vaše virtualno računalo i radi lakšeg izvođenja vježbe spojite se na virtualno računalo putem terminal emulatora (npr. Putty, kako je objašnjeno u dokumentu „Upute za rad s virtualnim računalom iz predmeta Napredni modeli i baze podataka“)

Dva su osnovna načina korištenja Apache Spark platforme:

1. **Spark interaktivna ljuska (eng. *shell*)** – najjednostavniji način za učenje Spark API-ja i brzu, interaktivnu analizu podataka. Trenutno postoje implementacije u programskim jezicima Java i Python.
2. **Samostojeće aplikacije (eng. *standalone applications*)** – mogućnost razvoja samostojećih aplikacija povrh Apache Spark platforme. Koriste se za distribuiranu obradu podataka na većoj skali (velike količine podataka). Spark pruža podršku za samostojeće aplikacije pisane u programskim jezicima Java, Python i Scala.

U okviru ove vježbe upoznat ćemo se s oba načina korištenja Spark platforme. U prvom dijelu vježbe, kroz rad u interaktivnoj ljusci upoznat ćemo se s osnovnim pojmovima i konceptima u radu sa Spark platformom. Stečena znanja ćemo u drugom dijelu vježbe primijeniti na razvoj samostojeće, distribuirane Java aplikacije za stvarnovremensku analizu toka podataka s Twitter-a.

1. Apache Spark interaktivna ljuska

- U ovom dijelu vježbe upoznat ćemo se s osnovama obrade podataka u Spark-u, na primjeru analize log zapisa o razmjeni HTTP zahtjeva/odgovora
- Postavite potrebne varijable okruženja (Spark izvorni direktorij i putanja do potrebnih Python biblioteka):

```
export SPARK_HOME=/root/spark
export PYTHONPATH=$SPARK_HOME/python:$SPARK_HOME/python/build:$PYTHONPATH
```

- Ukoliko već niste, omogućite pristup vanjskoj mreži:

```
ifup eth1
```

- Pozicionirajte se u direktorij /root/spark te preuzmite i raspakirajte log datoteke za analizu u direktorij nmbplogs:

```
[root@localhost spark]# wget http://kent.zpr.fer.hr/nmbplogs.tar.gz && mkdir
./nmbplogs && tar xvfz nmbplogs.tar.gz -C nmbplogs && rm nmbplogs.tar.gz
```

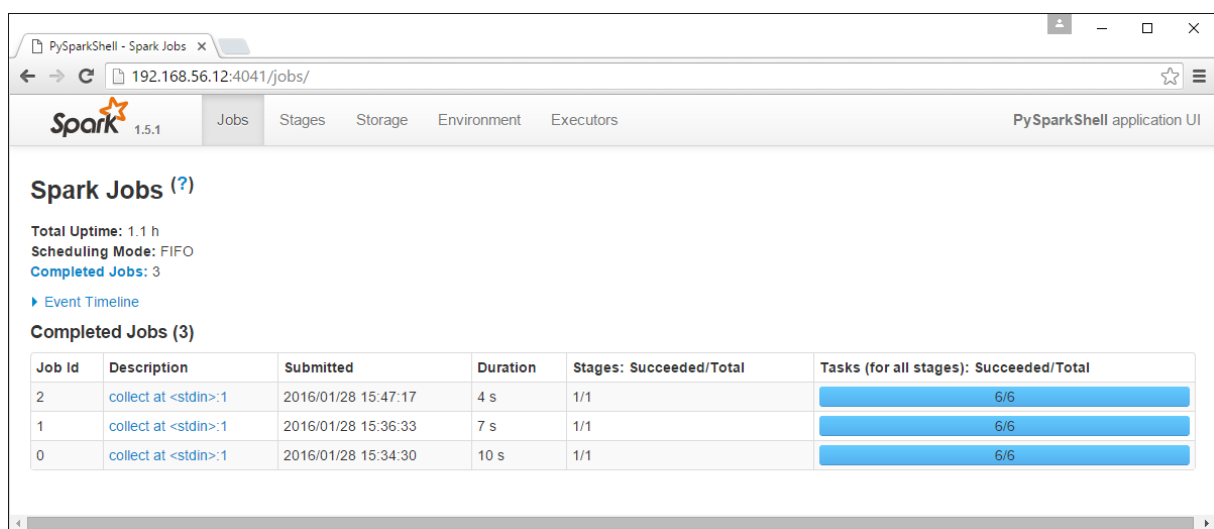
- Pozicionirajte se u direktorij /root/spark i pokrenite Spark interaktivnu ljusku (Python implementacija) s 3 lokalna čvora:

```
[root@localhost spark]# ./bin/pyspark -master local[3]
```

- Ljuskę u svakom trenutku možete napustiti zadavanjem kombinacije tipki 'CTRL+D':
- Tijekom pokretanja ljuske, u prikazanom tragu izvođenja dobili smo i informaciju o pristupnoj IP adresi i portu za Spark grafičko korisničko sučelje, pomoću kojeg tijekom rada možete pratiti status pojedinih čvorova, dodijeljene zadatke, itd.:

```
.....
16/01/28 14:01:01 INFO Utils: Successfully started service 'SparkUI' on port 4041.
16/01/28 14:01:01 INFO SparkUI: Started SparkUI at http://192.168.56.12:4041
16/01/28 14:01:01 WARN MetricsSystem: Using default name DAGScheduler for source
because spark.app.id is not set.
.....
```

- Na *host* računalu pokrenite preglednik, spojite se na Spark sučelje putem navedene adrese i porta, te tijekom daljnjeg rada pratite status vaših čvorova:



The screenshot shows the Spark UI web interface in a browser window. The address bar shows the URL `192.168.56.12:4041/jobs/`. The interface includes a navigation bar with tabs for Jobs, Stages, Storage, Environment, and Executors. The main content area is titled "Spark Jobs (?)" and displays summary statistics: Total Uptime: 1.1 h, Scheduling Mode: FIFO, and Completed Jobs: 3. Below this is a table of completed jobs.

Job Id	Description	Submitted	Duration	Stages: Succeeded/Total	Tasks (for all stages): Succeeded/Total
2	collect at <stdin>:1	2016/01/28 15:47:17	4 s	1/1	6/6
1	collect at <stdin>:1	2016/01/28 15:36:33	7 s	1/1	6/6
0	collect at <stdin>:1	2016/01/28 15:34:30	10 s	1/1	6/6

- Svaki Spark program kao prvi korak kreira `SparkContext` objekt koji definira kako Spark pristupa raspodijeljenom skupu čvorova te služi kao ulazna točka prema Spark API-ju. Kod rada u interaktivnoj ljusci, `SparkContext` objekt je automatski kreiran i sadržan u varijabli `sc`:

```
>>> sc
<pyspark.context.SparkContext object at 0x9d12cac>
```

- Primarnu apstrakciju podataka u Spark-u čine RDD objekti (*Resilient Distributed Dataset*, pogledati predavanje). Jednom naredbom učitavamo sve podatke iz preuzetih log datoteka i spremamo ih u jedan RDD objekt `logData`:

```
>>> logData = sc.textFile("/root/spark/nmbplogs/*")
```

- Nad RDD objektima možemo obavljati:
 - **transformacije** - operacije bazirane na principu lijene evaluacije (eng. *lazy evaluation*), stvaraju novi RDD na temelju transformacije postojećeg. Evaluiraju se u trenutku kad se nad RDD-om obavi neka akcija.
 - **akcije** – vraćaju rezultat ili perzistiraju RDD objekt. Podrazumijevaju i obradu (evaluaciju) svih do tada obavljenih transformacija kroz koje je nastao RDD

RDD

- Obavimo transformaciju (`filter()`) kreiranog RDD objekta na način da dohvatimo sve retke sa zapisom „ERROR“:

```
>>>> errorLines = logData.filter(lambda line: "ERROR" in line)
```

- Pogledajte ([ovdje](#), poglavlje „Transformations“) i jednostavne primjere drugih često korištenih *transformacija* nad RDD objektima, kao što su `map()` i `flatMap()`

- Pogledajmo sada rezultat naše *transformacije* (filtriranja) pozivom odgovarajuće *akcije* (`collect()` za ispis sadržaja objekta) nad kreiranim RDD objektom `errorLines`:

```
>>>> errorLines.collect()
```

- Pogledajte ([ovdje](#), poglavlje „Actions“) i jednostavne primjere drugih često korištenih akcija nad RDD objektima, kao što su `take()`, `count()`, `saveAsTextFile()`, ...

- Pokušajmo sada rezultat filtriranja pospremiti u .CSV datoteku izvođenjem odgovarajuće *akcije*:

```
>>>> errorLines.saveAsTextFile("errorLines.txt")
```

Što se dogodilo? Gdje je rezultat? Zbog svoje distribuirane prirode, Spark je kreirao *direktorij* naziva `errorLines.txt` u kojem su smještene različite particije rezultata. Pregledajte sadržaj svih kreiranih `part-*` datoteka i pronađite vaš rezultat.

- Sada želimo doznati koliko su trajali pojedini HTTP zahtjevi. To ćemo izračunati kao razliku u vremenima (u sekundama) između redaka koji u sebi sadrže "Request received" i "Sending response"

- Pogledajte nekoliko redaka da se поближе upoznate sa strukturom podataka koje obrađujemo:

```
[root@localhost spark]# tail -n 100 ./nmbplogs/nmbp_webapi_log.txt
```

- Primijetite da svi zapisi jednog Request-Response slijeda imaju jedinstven ID koji će nam omogućiti da po njemu grupiramo, npr.

```
[root@localhost spark]# cat ./nmbplogs/nmbp_webapi_log.txt | grep 8132e370-e347-4711-9b5a-fe1c3d2da209
```

- Uključimo biblioteke za rad s vremenskim tipovima podataka te biblioteku za rad s regularnim izrazima:

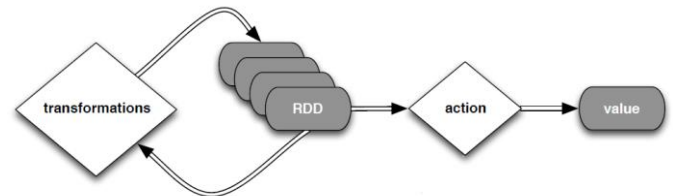
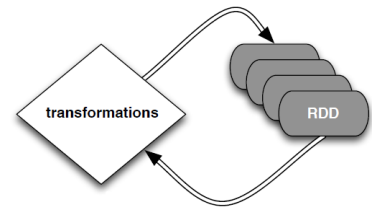
```
>>>> import time
>>>> import re
```

- Novom transformacijom početnog RDD objekta izdvojimo samo one retke koji sadrže „Request received“ i „Sending response“ stringove:

```
>>>> reqRes = logData.filter(lambda line: "Request received" in line or "Sending response" in line)
```

- Definirajmo Map funkciju:

```
>>>> def emit(line):
```



```
key = line.split(", ")[4 if "Sending response" in line else 3]
url = re.search('Url=(.*?),', line).group(1)
mtime = time.strptime(line.split(", ")[0].split()[4][1:][:3], "%Y-%m-%dT%H:%M:%S.%f")
return (key, (url, mtime))
```

- Probajte ju obaviti nad jednim retkom iz loga da vidite što vraća (npr. možete dohvatiti redak s `reqRes.take(1)`)
- Reduce funkciju ćemo ostaviti inline, tj. putem `lambda`-e. Reduce samo računa razliku u sekundama (zašto ima `abs`?)

```
result = reqres.map(emit).reduceByKey(lambda a, b: (a[0], abs( time.mktime(a[1]) - time.mktime(b[1]))))
```

- Pogledajmo rezultat:

```
result.collect()
```

- Vidimo da rezultat sadrži previše podataka i nisu sortirani - uzmimo prvih 100 zapisa sortiranih silazno po broju sekundi:

```
result.takeOrdered(100, lambda(key, val): -1*val[1]).collect()
```

2. Samostojeće Spark aplikacije

- U ovom dijelu vježbe pogledati ćemo jednostavni gotovi primjer samostojeće aplikacije pisane u programskom jeziku Java koju ćemo pokrenuti na Spark instalaciji u vašem virtualnom stroju. Po analognom principu ćemo u zadnjem dijelu vježbe razvijati vlastitu Java aplikaciju koju ćemo pokrenuti na Spark clusteru postavljenom na Amazon EC2 servisu.
- Aplikacija treba učitati sadržaj README.md datoteke smještene u direktoriju /root/spark te prebrojati sve retke koji sadrže slovo „a“.
- U direktoriju /root/spark kreirajmo direktorij SimpleApp u kojeg ćemo smjestiti izvorni kod i ostale prateće datoteke aplikacije. Budući da ćemo za prevođenje koristiti [Apache Maven](#), sve datoteke unutar direktorija SimpleApp potrebno je posložiti prema sljedećoj standardiziranoj strukturi:

```
./pom.xml
./src
./src/main
./src/main/java
./src/main/java/SimpleApp.java
```

- Izvorni kod aplikacije (sadržaj datoteke SimpleApp.java):

```
import org.apache.spark.api.java.*;
import org.apache.spark.SparkConf;
import org.apache.spark.api.java.function.Function;

public class SimpleApp {
    public static void main(String[] args) {
        String logFile = "/root/spark/README.md";
        SparkConf conf = new SparkConf().setAppName("Simple Application");
        JavaSparkContext sc = new JavaSparkContext(conf);
        JavaRDD<String> logData = sc.textFile(logFile).cache();

        long numAs = logData.filter(new Function<String, Boolean>() {
            public Boolean call(String s) { return s.contains("a"); }
        }).count();

        System.out.println("Lines with a: " + numAs);
    }
}
```

- Sadržaj datoteke pom.xml (eng. *POM – Project Object Model*, XML datoteka koja sadrži informacije o projektu, konfiguracijske parametre, ovisnosti o drugim bibliotekama itd.):

```
<project>
  <groupId>edu.berkeley</groupId>
  <artifactId>simple-project</artifactId>
  <modelVersion>4.0.0</modelVersion>
```

```
<name>Simple Project</name>
<packaging>jar</packaging>
<version>1.0</version>
<dependencies>
  <dependency> <!-- Spark dependency -->
    <groupId>org.apache.spark</groupId>
    <artifactId>spark-core_2.10</artifactId>
    <version>1.5.1</version>
  </dependency>
</dependencies>
</project>
```

- Za prevođenje aplikacije koristimo Apache Maven – za jednostavnije pokretanje Maven-a dodajte bin direktorij u varijablu okruženja PATH:

```
[root@localhost SimpleApp]# export PATH=$PATH:/opt/apache-maven-3.3.3/bin
```

- Za uspješno prevođenje potreban je i pristup Internetu iz vašeg virtualnog računala:

```
[root@localhost SimpleApp]# ifup eth1
```

- Pokrenimo prevođenje iz izvornog direktorija aplikacije (/root/SimpleApp)

```
[root@localhost SimpleApp]# mvn package
```

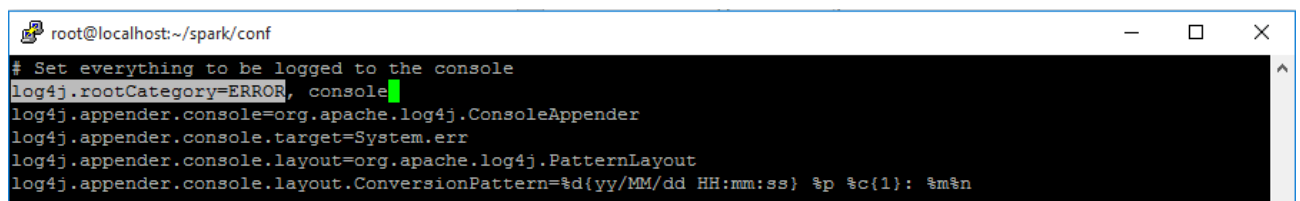
- Prevedena i zapakirana izvršna aplikacija (simple-project-1.0.jar) smještena je u direktorij target unutar izvornog direktorija aplikacije. Pokrenimo aplikaciju (za pokretanje samostojjećih aplikacija u Sparku koristi se skripta [spark-submit](#)).

```
[root@localhost SimpleApp]# ../bin/spark-submit --class "SimpleApp" --master
local[3] target/simple-project-1.0.jar
```

- Gdje nam je rezultat? Nalazi se negdje u šumi ispisanih tragova izvođenja ...
- Kao što ste mogli primijetiti, prema standardnim postavkama, Spark na standardni izlaz ispisuje detaljan trag izvođenja (eng. *trace*). Radi bolje preglednosti, promijenimo postavke na način da onemogućimo ispis *INFO* traga izvođenja, u većini slučajeva zanimaju nas isključivo *ERROR* zapisi. Za potrebe logiranja i praćenja traga izvođenja Spark koristi biblioteku [Log4j](#). Pozicionirajte se u direktorij /root/spark/conf i kreirajte novu konfiguracijsku datoteku prema već dostupnom predlošku log4j.properties.template:

```
[root@localhost conf]# cp log4j.properties.template log4j.properties
```

- Ažurirajte novostvorenu datoteku (korištenjem ugrađenog editora vi ili korištenjem nekog vanjskog (host) editora koji ima mogućnost direktnog editiranja i snimanja podataka preko FTP-a) – postavite opciju log4j.rootCategory na vrijednost ERROR:



```
root@localhost:~/spark/conf
# Set everything to be logged to the console
log4j.rootCategory=ERROR, console
log4j.appender.console=org.apache.log4j.ConsoleAppender
log4j.appender.console.target=System.err
log4j.appender.console.layout=org.apache.log4j.PatternLayout
log4j.appender.console.layout.ConversionPattern=%d{yy/MM/dd HH:mm:ss} %p %c{1}: %m%n
```

- Spremite datoteku i ponovno pokrenite program – uočite da sada kao izlaz dobivamo samo rezultat izvođenja programa.

3. Analiza toka podataka s Twitter-a

U ovom dijelu vježbe fokusirat ćemo se na Apache Spark Streaming - komponentu sustava Apache Spark koja je namijenjena za efikasnu distribuiranu analizu tokova podataka (eng. *data streams*). Razvijati ćemo Java aplikaciju koja koristi Apache Spark Streaming za procesiranje toka objava na Twitter-u u stvarnom vremenu. Aplikaciju ćemo umjesto pokretanja u okviru virtualne mašine pokrenuti na Apache Spark clusteru postavljenom na Amazon EC2 Cloud servisu.

3.1. Amazon EC2 Cloud

Za potrebe izvođenja ove vježbe koristit ćemo Apache Spark instalaciju postavljenu u cluster modu na Amazon EC2 Cloud servisu. Prije nastavka vježbe potrebno se upoznati s osnovnim pojmovima vezanim uz Amazon EC2 Cloud servis i Apache Spark u cluster modu:

- Pogledajte uvodni video o Amazon EC2 Cloud servisu dostupan [ovdje](#)
- Pročitajte kratki pregled komponenti Apache Spark sustava u cluster modu [ovdje](#)

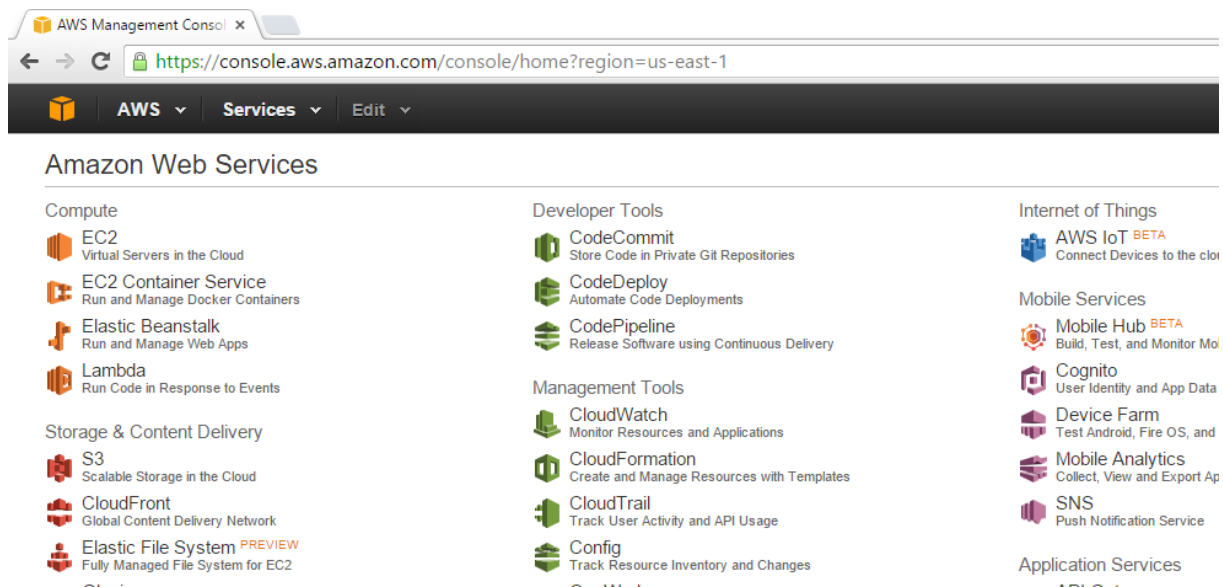
Na Amazon-u je već postavljen i pokrenut Apache Spark cluster s jednim *master* čvorom i pet *slave* čvorova.

NAPOMENA: Radi uštede resursa, cluster će studentima biti dostupan za rad tijekom cijele veljače, **svakim danom u terminu 19:00 – 00:00 h.**

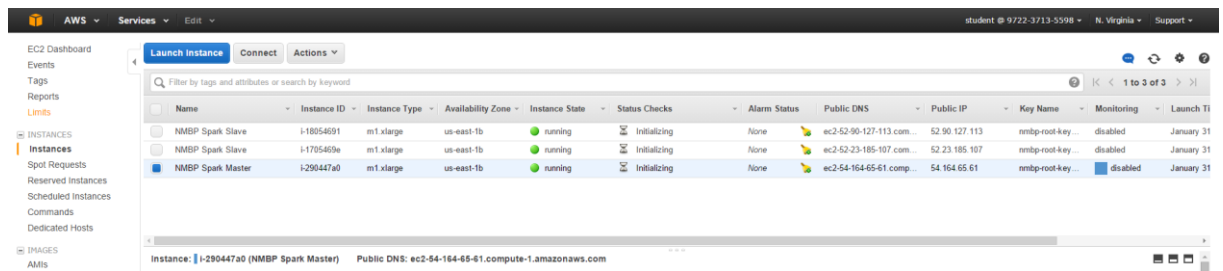
Spojimo se na kontrolnu konzolu Amazon EC2 servisa i uvjerimo se da je naš cluster u pogonu:

- URL za logiranje na konzolu: <https://972237135598.signin.aws.amazon.com/console>
 - **Username:** student
 - **Password:** nbmp-streaming-lab

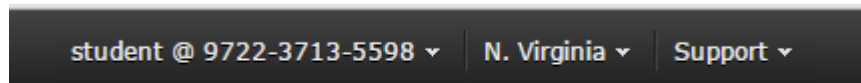
Nakon uspješnog logiranja otvara se početna stranica Amazon kontrolne konzole:



Odaberimo *EC2 Virtual Servers in the Cloud*. Na početnoj stranici možemo vidjeti pregled resursa koji su trenutno u upotrebi. U izborniku s lijeve strane odaberimo stavku *Instances* i pogledajmo stanje instanci na kojima se vrti naš Apache Spark cluster.



Ukoliko instance nisu vidljive, potrebno se prebaciti (padajući izbornik gore-desno) na odgovarajuću zonu (US East: N. Virginia)



Možemo vidjeti ukupno tri instance: jedna Spark master instanca (*NMBP Spark Master*) i dvije Spark slave instanci (*NMBP Spark Slave*). Za uspješan rad clustera sve instance moraju biti pokrenute (stanje *running*). Označite master instancu i proučite dodatne informacije o odabranoj instanci:

Name	Instance ID	Instance Type	Availability Zone	Instance State	Status Checks	Alarm Status	Public DNS	Public IP	Key Name	Monitoring	Launch Time
NMBP Spark Slave	i-18054691	m1.xlarge	us-east-1b	running	2/2 checks passed	None	ec2-52-90-127-113.com...	52.90.127.113	nmbp-root-key...	disabled	January 31
NMBP Spark Slave	i-1705469e	m1.xlarge	us-east-1b	running	2/2 checks passed	None	ec2-52-23-185-107.com...	52.23.185.107	nmbp-root-key...	disabled	January 31
NMBP Spark Master	i-290447a0	m1.xlarge	us-east-1b	running	2/2 checks passed	None	ec2-54-164-65-61.comp...	54.164.65.61	nmbp-root-key...	disabled	January 31

Instance: **i-290447a0 (NMBP Spark Master)** Public DNS: **ec2-54-164-65-61.compute-1.amazonaws.com**

Description | Status Checks | Monitoring | Tags

Instance ID	i-290447a0	Public DNS	ec2-54-164-65-61.compute-1.amazonaws.com
Instance state	running	Public IP	54.164.65.61
Instance type	m1.xlarge	Elastic IP	-
Private DNS	ip-172-31-15-150.ec2.internal	Availability zone	us-east-1b
Private IPs	172.31.15.150	Security groups	ampcamp3-master view rules
Secondary private IPs	-	Scheduled events	No scheduled events
VPC ID	vpc-b717a1d3	AMI ID	ampcamp4-4 (ami-19474270)
Subnet ID	subnet-f7f1f781	Platform	-
Network interfaces	eth0	IAM role	-
Source/dest. check	True	Key pair name	nmbp-root-keypair
EBS-optimized	False	Owner	972237135598
Root device type	ebs	Launch time	January 31, 2016 at 3:04:43 PM UTC+1 (less than one hour)
Root device	/dev/sda1	Termination protection	False
Block devices	/dev/sda1	Lifecycle	normal
	/dev/sdf	Monitoring	basic
		Alarm status	None
		Kernel ID	aki-427d952b

Trenutno nam je najvažniji podatak u vanjskoj IP adresi/nazivu računala preko kojeg ćemo se kasnije moći spojiti na cluster putem SSH/FTP konekcije:

Public DNS: `ec2-54-164-65-61.compute-1.amazonaws.com` (moguće je da se hostname između uzastopnih paljenja/gašenja instanci promijeni, stoga obavezno uvijek prvojerite hostname vašeg master čvora prije nastavka)

3.2. Amazon EC2 Cloud- SSH konekcija

Iako se putem SSH konekcije moguće spojiti direktno iz terminala (npr. Putty) sa vašeg host računala, u ostatku vježbe ćemo komunikaciju prema master čvoru na Amazonu ostvariti iz našeg virtualnog računala. Za direktno spajanje iz Putty-ja pogledajte upute u Dodatku B: *Konfiguracija FTP i SSH klijenata za spajanje na Amazon*.

- Ukoliko već niste, pokrenite vaše virtualno računalo i omogućite vezu prema vanjskoj mreži (startati mrežno sučelje eth1)
- Za uspješno spajanje unaprijed su generirani potrebni pristupni ključevi koje je potrebno dodati u odgovarajuće varijable okruženja:

```
[root@localhost ~]# export AWS_ACCESS_KEY_ID=AKIAJBIIIPXL7J4XJ2ALQ
[root@localhost ~]# export
AWS_SECRET_ACCESS_KEY=u4i5r08xNpWP5nF9nKQtSbQwVQ0fA4yurhbOb37d
```

- Uvjerimo se da su varijable okruženja uspješno postavljene:

```
[root@localhost ~]# printenv
```

- Iz home (~) direktorija obavite sljedeću naredbu koja će skinuti ključeve i spremiti ih u .ssh poddirektorij:

```
[root@localhost ~]# wget http://kent.zpr.fer.hr/nmbp-keypair.tar.gz && mkdir
~/ssh && tar xvfz nmbp-keypair.tar.gz -C ~/.ssh && rm nmbp-keypair.tar.gz
```

- Nakon uspješnog kopiranja, promijenite dozvole nad datotekom na način da samo vi možete u datoteku pisati i iz nje čitati:

```
[root@localhost ~]# chmod 600 ~/.ssh/nmbp-root-keypair.pem
```

- Ažurirajte datoteku `/etc/ssh/ssh_config` na način da obrišete zadnju liniju datoteke sa zapisom `'UseDNS no'`
- Konačno, otvorimo SSH konekciju prema master čvoru na Amazon Cloud-u, koristeći ime računala koje smo ranije doznali iz Amazon kontrolne konzole (provjeriti hostname Spark master instance kroz AWS konzolu):

```
[root@localhost ~]# ssh -i ~/.ssh/nmbp-root-keypair.pem <spark master public DNS>
```

- Za login koristite korisničko ime `root`

3.2. Implementacija programa

- Prije nastavka, poželjno je upoznati se s osnovama Apache Spark Streaming komponente (do poglavlja *Input Dstreams and receivers*) dostupnima [ovdje](#). (nije potrebno implementirati prateći primjer na navedenom linku)
- Nakon uspješnog spajanja na master čvor, u početnom (`root`) direktoriju možete pronaći direktorij `nmbp-streaming-lab`
- Unutar ovog direktorija kreirajte novi direktorij naziva koji odgovara vašem JMBAG-u (**NAPOMENA:** ostatak vježbe odradite isključivo u vašem direktoriju – molimo da ne mijenjate sadržaj ostalih direktorija/datoteka)
- Kopirajte unaprijed pripremljeni predložak za implementaciju programa dostupan u direktoriju `/root/nmbp-streaming-lab/project-twitter-template` u vaš novokreirani direktorij. Npr. za studenta s JMBAG-om 1122334455 potrebno je obaviti sljedeće:

```
[root@ip-172-31-49-35 ~]# mkdir /root/nmbp-streaming-lab/1122334455
[root@ip-172-31-49-35 ~]# cp -r /root/nmbp-streaming-lab/project-twitter-template/* /root/nmbp-streaming-lab/1122334455/
```

- Nakon uspješno obavljenog prethodnog koraka, struktura vašeg direktorija treba izgledati ovako:

```
1122334455/
├── project_twitter
│   ├── java
│   │   ├── build.sbt
│   │   ├── sbt
│   │   │   ├── sbt
│   │   │   └── sbt-launch-0.13.1.jar
│   │   ├── ScalaHelper.scala
│   │   ├── TutorialHelper.java
│   │   └── Tutorial.java
│   └── twitter.txt
```

- Samostalno analizirajte strukturu projekta – najbitnije su nam sljedeće datoteke:
 - `twitter.txt` – datoteka u koju ćemo zapisati ključeve potrebne za uspješnu autentifikaciju prema Twitter API-ju
 - `Tutorial.java` – datoteka koja sadrži predložak za glavni Java program, ovu datoteku ćemo ažurirati, kompajlirati i pokrenuti
 - `TutorialHelper.java` – datoteka koja sadrži nekoliko pomoćnih funkcija (samostalno

pogledajte koje su to funkcije i čemu služe)

- Za uspješan programski pristup na Twitter API (eng. *Application Programming Interface*) radi dohvata toka podataka, potrebno je kroz vaš Twitter korisnički račun osigurati privremenu autentifikaciju prema uputama u Dodatku A: *Konfiguracija Twitter korisničkog računa*
- Nakon uspješno odrađenog prethodnog koraka imamo generirane **API (Consumer) Key**, **API (Consumer) Secret**, **Access Token** i **Access Token Secret** koje je potrebno zapisati u datoteku `twitter.txt` u našem projektu. Primjer kako bi datoteka trebala izgledati nakon ažuriranja (iz sigurnosnih razloga u ovom primjeru nisu navedeni cjeloviti zapisi ključeva):

```
consumerKey = z25xt02zcaadf12...
consumerSecret = gqc9uAkjla13...
accessToken = 8mitfTqDrgAzasd...
accessTokenSecret = 479920148...
```

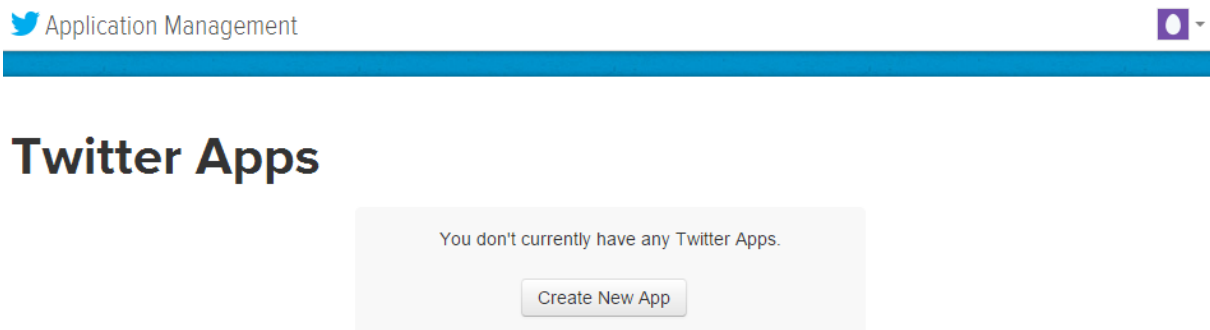
- Ostatak vježbe obavite u vašem novokreiranom direktoriju, prateći korake dostupne [ovdje](#) (poglavlje „2. First Spark Streaming Program“).
 - Za jednostavniji rad s datotekama možete se na Amazon Spark master čvor spojiti i putem FTP-a, prema uputama u Dodatku B: *Konfiguracija FTP i SSH klijenata za spajanje na Amazon*
 - **NAPOMENA:** u uputama na navedenoj poveznici stoji da se za uspješno prevođenje/pokretanje programa potrebno pozicionirati u direktorij `/root/streaming/java` – u našem slučaju potrebno je biti pozicioniran u direktorij `/root/nmbp-streaming-lab/<JMBAG>/project_twitter/java`
- Nakon uspješno implementiranog i pokrenutog programa isprobajte i dodatne vježbe dostupne na istoj poveznici, poglavlje „3. Further exercises“

Dodatak A: Konfiguracija Twitter korisničkog računa

- potrebno je konfigurirati privremenu autentifikaciju preko Twitter korisničkog računa (key/secret par i access token/secret par)
- u nijednom trenutku iz programa nije potrebno davati Twitter username / password – korisnički račun neće biti kompromitiran tijekom vježbe

Twitter Application Settings stranica: <https://apps.twitter.com/>

- Lista postojećih aplikacija za koje su prethodno kreirani i generirani pristupni ključevi i tokeni
- Ukoliko je ovo prvi puta, lista je prazna. Kreirajmo novu aplikaciju klikom na „Create New App“



Otvora se obrazac za kreiranje nove aplikacije kao na sljedećoj slici:


The screenshot shows the 'Create an application' form. It has a header with the Twitter logo and 'Application Management'. The main heading is 'Create an application'. Below this, there's a section titled 'Application Details' with four fields: 'Name' (containing 'nkatanic-nmbp-streaming'), 'Description' (containing 'Laboratorijska vježba iz predmeta Napredni modeli i baze podataka'), 'Website' (containing 'http://www.nmbp-streaming-lab.com'), and 'Callback URL' (empty). Each field has a small asterisk indicating it's required. Below the fields, there's a note about OAuth 1.0a applications and a warning about restricting the application from using callbacks.

- Unesite tražene podatke:
 - **Name:** mora biti globalno jedinstveni identifikator – možemo osigurati korištenjem username-a kao prefixa u sljedećem obliku: <username>-nmbp-streaming
 - **Description:** proizvoljno
 - **Website:** u okviru vježbe nam nije potrebno, ali je obavezno polje za uspješno kreiranje aplikacije. U smislu sintakse mora biti ispravan URL iako ne mora trenutno postojati. Za ovu potrebu upišite primjerice *http://www.nmbp-streaming-lab.com*
 - **Callback URL:** ostaviti prazno
 - Označiti „Yes, I agree“ za „Developer Agreement“ na dnu stranice i kliknuti na gumb „Create your Twitter application“
- Potrebno unijeti broj mobitela (ubaciti ranije u opisu)

Create an application

- Na početnoj stranici Twittera klik na ikonu koja reprezentira vaš account -> Settings -> Mobile
- Izaberite Country/region „Croatia“ i unesite vaš broj mobitela bez prateće 0 (npr. 98111222 za „098111222“ ili 91333444 za „091333444“ i sl.)
- Klikom na Continue od vas će se zatražiti unos verifikacijskog koda kojeg će te primiti SMS-om

Home Notifications Messages Search Twitter Tweet



Nenad Katanić
@nkatanic

- Account
- Security and privacy
- Password
- Cards and shipping
- Order history
- Mobile**

Mobile
Expand your experience, get closer, and stay current.

Check your phone.
We sent a code to +38598761016. Enter it below to verify your number.

Verification code

[Activate phone](#)

[Request a new confirmation code](#)

[Cancel phone activation](#)

- Unesite verifikacijski kod i dovršite unos broja klikom na „Activate Phone“
- Nakon obavljene vježbe dodani broj možete obrisati

1.1. Twitter Keys and Access Tokens


- Nakon uspješnog kreiranja otvara se stranica prikazana ispod

Application Management

nkatanic-nmbp-streaming

Test OAuth

- Details
- Settings
- Keys and Access Tokens
- Permissions



Laboratorijska vježba iz predmeta Napredni modeli i baze podataka
<http://www.nmbp-streaming-lab.com>

Organization

Information about the organization or company associated with your application. This information is optional.



Organization	None
Organization website	None

Application Settings

Your application's Consumer Key and Secret are used to [authenticate](#) requests to the Twitter Platform.

Access level	Read and write (modify app permissions)
Consumer Key (API Key)	zAXdK41YVHXAbODzm0O6CKn2M (manage keys and access tokens)
Callback URL	None
Callback URL Locked	No

- Prebacite se na tab „Keys and Access Tokens“
- Na otvorenoj stranici trebali biste vidjeti već generirani API Key i API Secret kao što je prikazano na slici ispod (cenzurirani dio 😊)

 Application Management
 

nkatanic-nmbp-streaming

Test OAuth

[Details](#)
[Settings](#)
[Keys and Access Tokens](#)
[Permissions](#)

Application Settings

Keep the "Consumer Secret" a secret. This key should never be human-readable in your application.

Consumer Key (API Key)	[REDACTED]
Consumer Secret (API Secret)	[REDACTED]
Access Level	Read and write (modify app permissions)
Owner	[REDACTED]
Owner ID	[REDACTED]

Application Actions

Regenerate Consumer Key and Secret

Change App Permissions

Your Access Token

You haven't authorized this application for your own account yet.

By creating your access token here, you will have everything you need to make API calls right away. The access token generated will be assigned your application's current permission level.

Token Actions

Create my access token

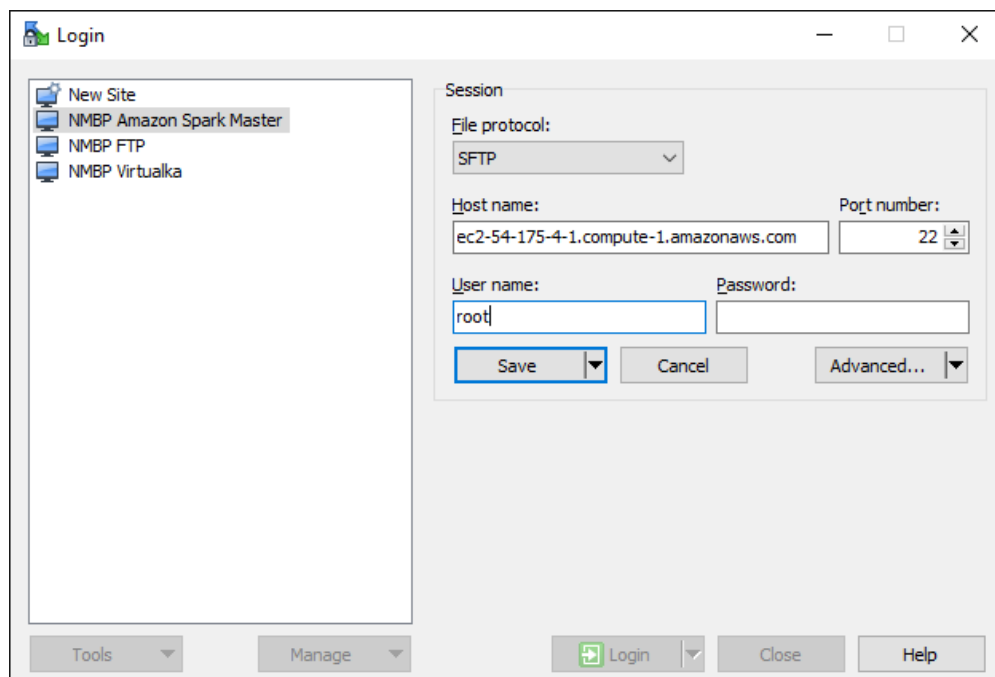
- Da biste generirali novi Access Token i Access Token Secret kliknite na gumb „Create my access token“ na dnu stranice (označeno crveno na prethodnoj slici)
- U ovom trenutku imamo generirane **API (Consumer) Key**, **API (Consumer) Secret**, **Access Token** i **Access Token Secret**. Ova četiri ključa koristit ćemo za programski pristup Twitter API-ju u nastavku ove vježbe.
- Nakon odrađene vježbe možete izbrisati generirane ključeve (Tab „Keys and Access Tokens“ -> Revoke Token Access), odnosno možete obrisati i cijelu aplikaciju (tab „Details“ -> Delete Application).

Dodatak B: Konfiguracija FTP i SSH klijenata za spajanje na Amazon

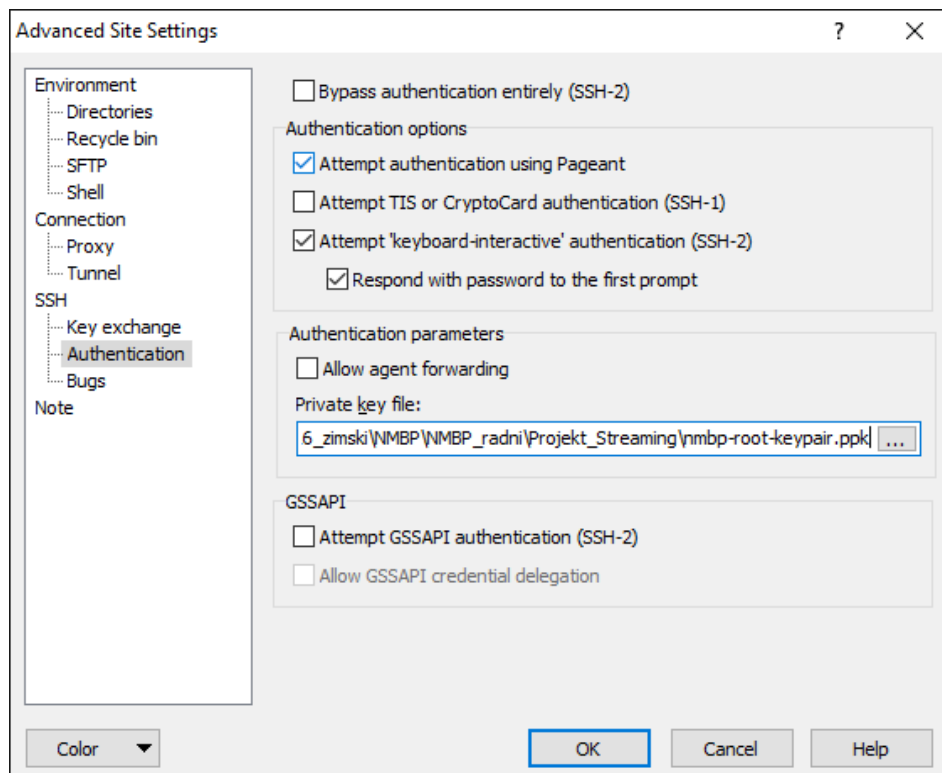
- U nastavku je prikazan primjer konfiguracije FTP klijenta WinSCP i SSH klijenta Putty
- Radi uspješne SSH autentifikacije, potrebno je prije spajanja dodati odgovarajuću *private key* datoteku (.ppk – *Putty Private Key File*)
- Već pripremljenu datoteku preuzmite ovdje (<http://kent.zpr.fer.hr/nmbp-root-keypair.zip>)

FTP klijent (WinSCP)

- Kreirajmo novu sesiju (New Session)
 - **Hostname:** <public DNS Spark master čvora na amazonu>
 - **Username:** root
 - **Password:** <prazno>



- Odabrati *Advanced* -> *SSH* -> *Authentication*
- Pod „*Private key file*“ učitati .ppk datoteku koju ste prethodno preuzeli



SSH klijent (Putty)

- **Hostname:** <public DNS Spark master čvora na amazonu>
- Iz izbornika s lijeve strane odabrati *SSH -> Auth*
- Pod „Private key file for authentication“ učitati .ppk datoteku koju ste prethodno preuzeli

