Measurements in AWS VM2VM (#4a)

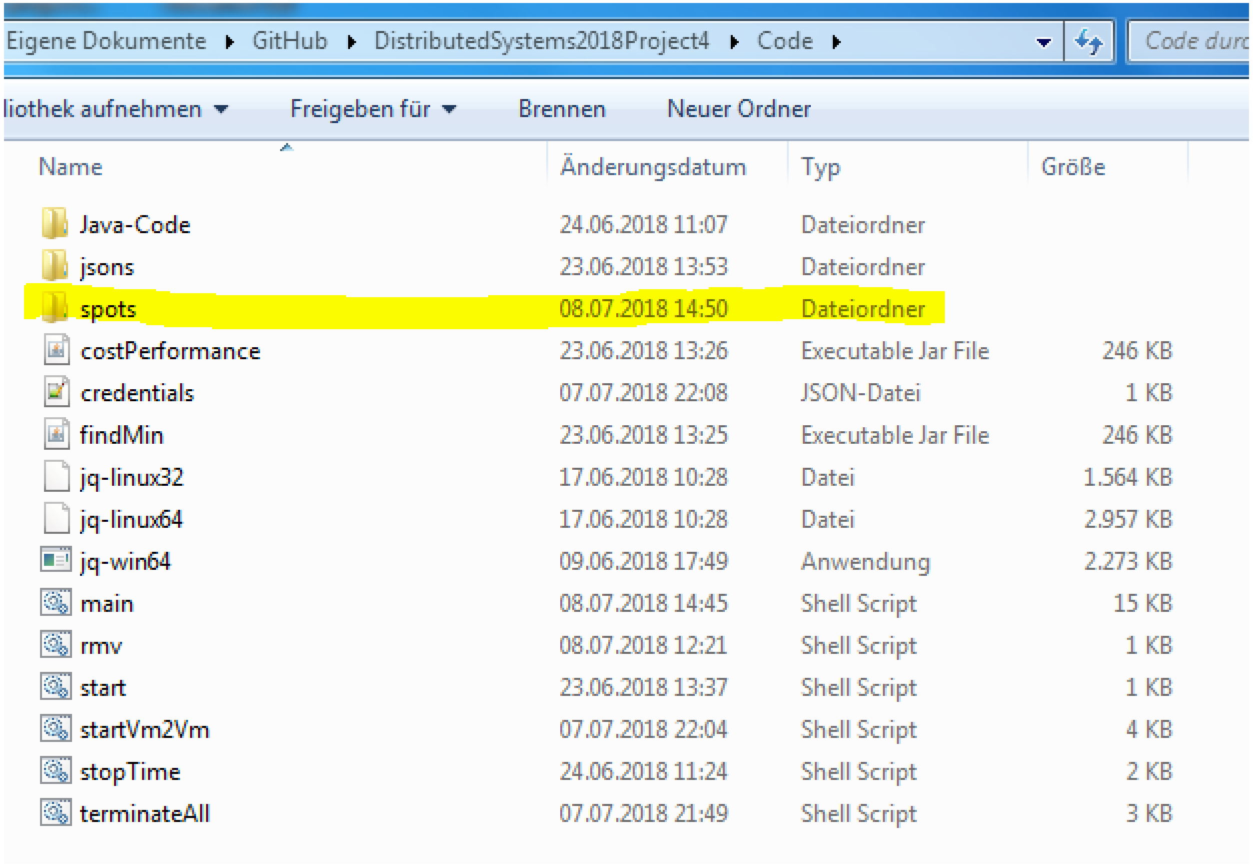
Ahmet Aspir

Mark Nardi

Martin Pfeifhofer

main.sh

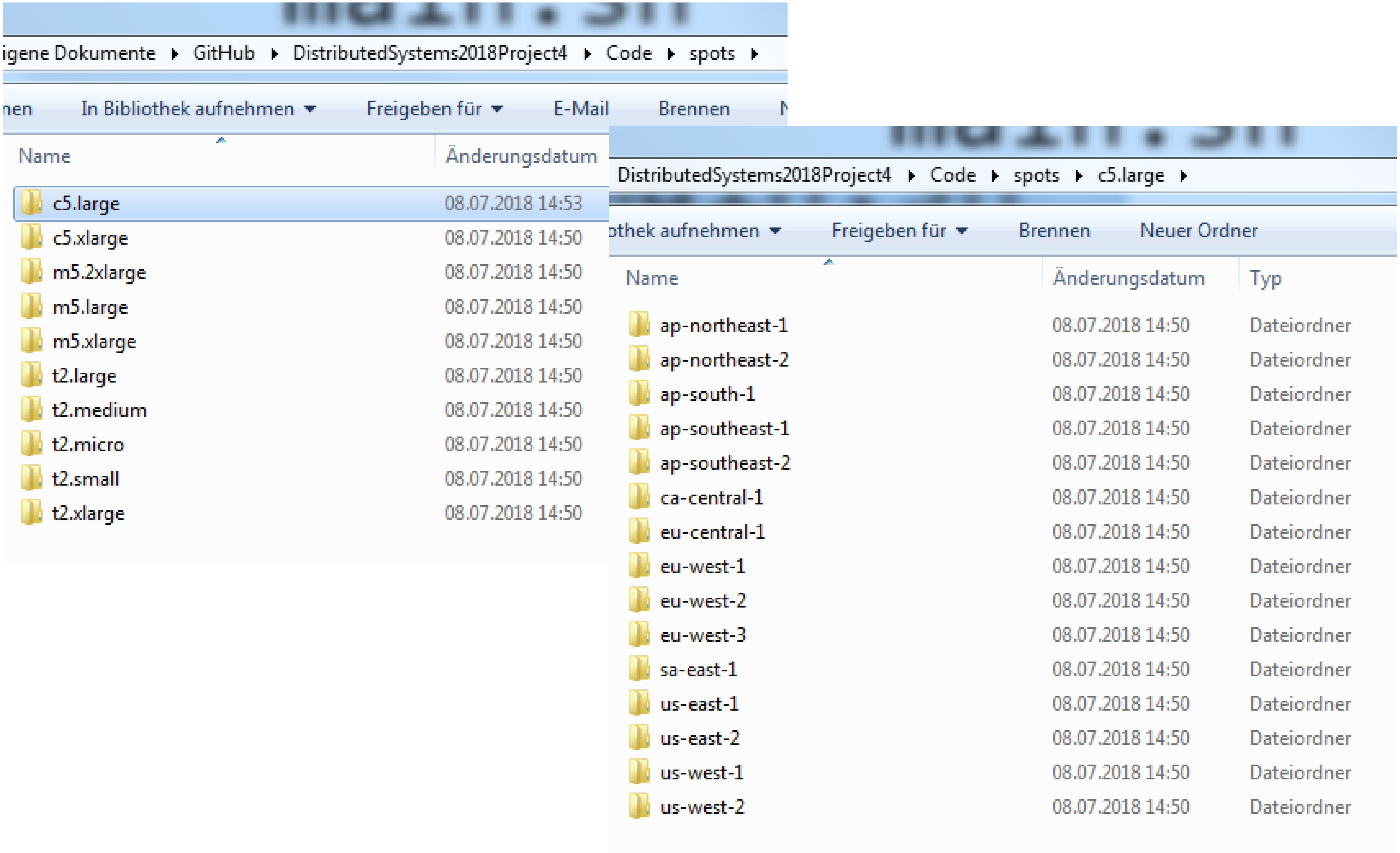
- 1. Generate folders for every instance-type and its regions



Eigene Dokumente ▸ GitHub ▸ DistributedSystems2018Project4 ▸ Code ▸
Code durc
liothek aufnehmen ▾
Freigeben für ▾
Brennen
Neuer Ordner

Name	Änderungsdatum	Typ	Größe
Java-Code	24.06.2018 11:07	Dateiordner	
jsons	23.06.2018 13:53	Dateiordner	
spots	08.07.2018 14:50	Dateiordner	
costPerformance	23.06.2018 13:26	Executable Jar File	246 KB
credentials	07.07.2018 22:08	JSON-Datei	1 KB
findMin	23.06.2018 13:25	Executable Jar File	246 KB
jq-linux32	17.06.2018 10:28	Datei	1.564 KB
jq-linux64	17.06.2018 10:28	Datei	2.957 KB
jq-win64	09.06.2018 17:49	Anwendung	2.273 KB
main	08.07.2018 14:45	Shell Script	15 KB
rmv	08.07.2018 12:21	Shell Script	1 KB
start	23.06.2018 13:37	Shell Script	1 KB
startVm2Vm	07.07.2018 22:04	Shell Script	4 KB
stopTime	24.06.2018 11:24	Shell Script	2 KB
terminateAll	07.07.2018 21:49	Shell Script	3 KB

main.sh


igene Dokumente ▸ GitHub ▸ DistributedSystems2018Project4 ▸ Code ▸ spots ▸
nen
In Bibliothek aufnehmen ▾
Freigeben für ▾
E-Mail
Brennen
N

Name	Änderungsdatum
c5.large	08.07.2018 14:53
c5.xlarge	08.07.2018 14:50
m5.2xlarge	08.07.2018 14:50
m5.large	08.07.2018 14:50
m5.xlarge	08.07.2018 14:50
t2.large	08.07.2018 14:50
t2.medium	08.07.2018 14:50
t2.micro	08.07.2018 14:50
t2.small	08.07.2018 14:50
t2.xlarge	08.07.2018 14:50

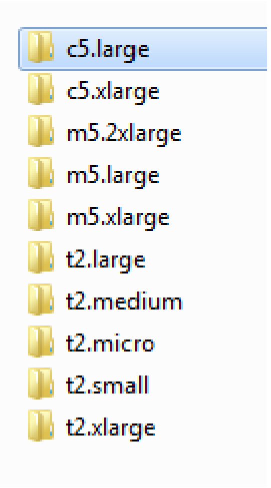
DistributedSystems2018Project4 ▸ Code ▸ spots ▸ c5.large ▸
othek aufnehmen ▾
Freigeben für ▾
Brennen
Neuer Ordner

Name	Änderungsdatum	Typ
ap-northeast-1	08.07.2018 14:50	Dateiordner
ap-northeast-2	08.07.2018 14:50	Dateiordner
ap-south-1	08.07.2018 14:50	Dateiordner
ap-southeast-1	08.07.2018 14:50	Dateiordner
ap-southeast-2	08.07.2018 14:50	Dateiordner
ca-central-1	08.07.2018 14:50	Dateiordner
eu-central-1	08.07.2018 14:50	Dateiordner
eu-west-1	08.07.2018 14:50	Dateiordner
eu-west-2	08.07.2018 14:50	Dateiordner
eu-west-3	08.07.2018 14:50	Dateiordner
sa-east-1	08.07.2018 14:50	Dateiordner
us-east-1	08.07.2018 14:50	Dateiordner
us-east-2	08.07.2018 14:50	Dateiordner
us-west-1	08.07.2018 14:50	Dateiordner
us-west-2	08.07.2018 14:50	Dateiordner

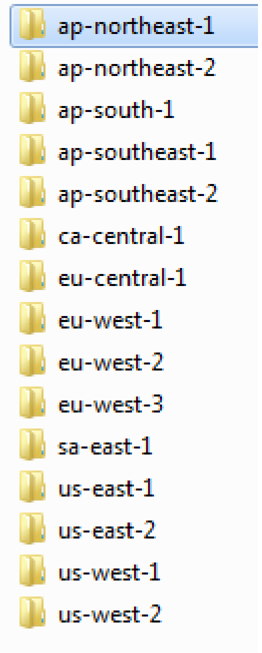
main.sh

- 2. Gather prices for all instance-types in each region and store them in every region folder of each instance-type

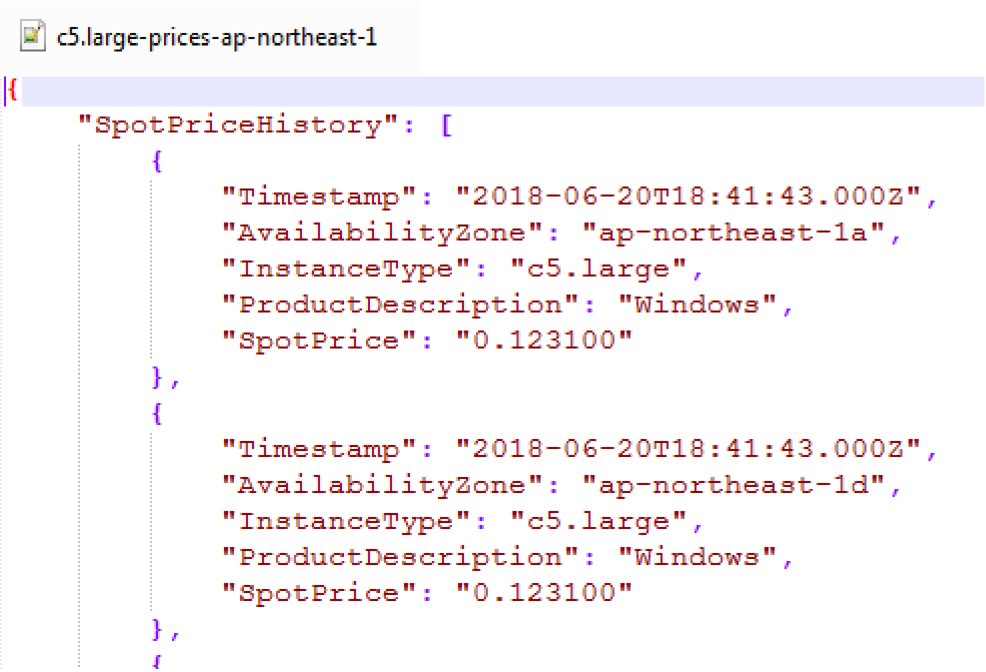
main.sh



c5.large
c5.xlarge
m5.2xlarge
m5.large
m5.xlarge
t2.large
t2.medium
t2.micro
t2.small
t2.xlarge



ap-northeast-1
ap-northeast-2
ap-south-1
ap-southeast-1
ap-southeast-2
ca-central-1
eu-central-1
eu-west-1
eu-west-2
eu-west-3
sa-east-1
us-east-1
us-east-2
us-west-1
us-west-2



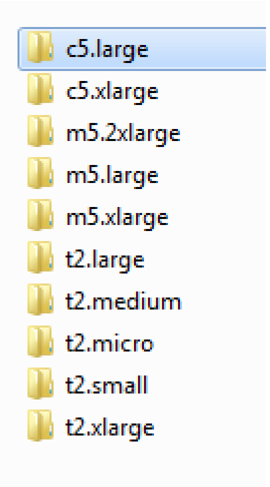
c5.large-prices-ap-northeast-1

```
{
    "SpotPriceHistory": [
        {
            "Timestamp": "2018-06-20T18:41:43.000Z",
            "AvailabilityZone": "ap-northeast-1a",
            "InstanceType": "c5.large",
            "ProductDescription": "Windows",
            "SpotPrice": "0.123100"
        },
        {
            "Timestamp": "2018-06-20T18:41:43.000Z",
            "AvailabilityZone": "ap-northeast-1d",
            "InstanceType": "c5.large",
            "ProductDescription": "Windows",
            "SpotPrice": "0.123100"
        },
        {
```

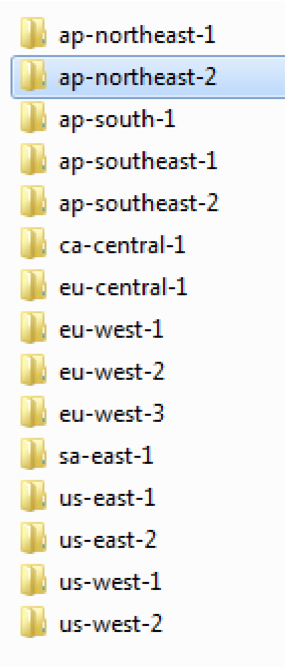
3. findMin.jar

- 3.1 Find the best price for all regions of each instance-type
- 3.2 Extract the 2 cheapest spot-instance entries from above and save it in the instance-type folder

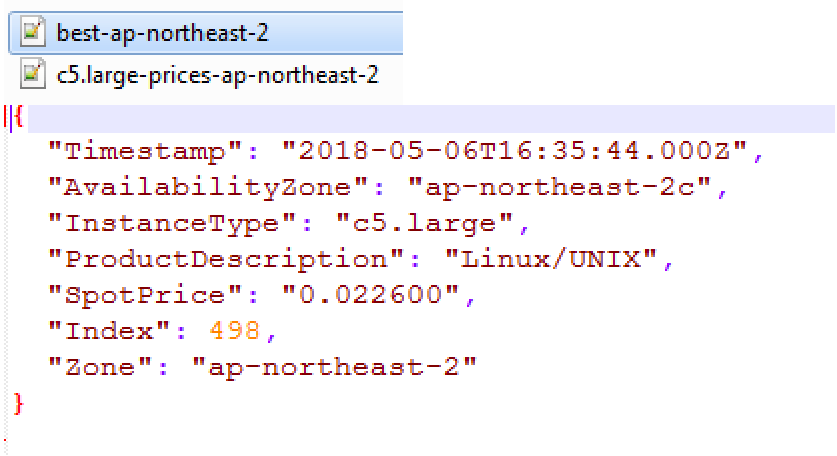
findMin.jar



c5.large
c5.xlarge
m5.2xlarge
m5.large
m5.xlarge
t2.large
t2.medium
t2.micro
t2.small
t2.xlarge



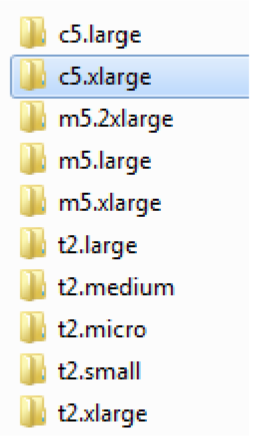
ap-northeast-1
ap-northeast-2
ap-south-1
ap-southeast-1
ap-southeast-2
ca-central-1
eu-central-1
eu-west-1
eu-west-2
eu-west-3
sa-east-1
us-east-1
us-east-2
us-west-1
us-west-2



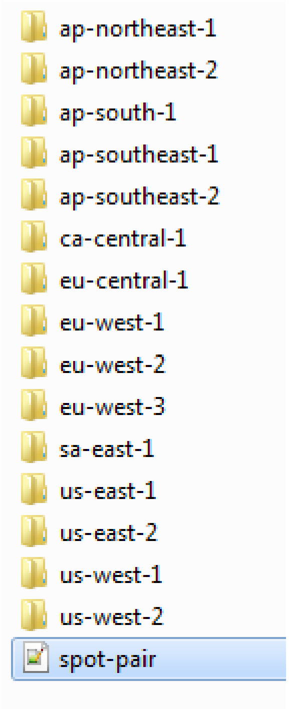
best-ap-northeast-2
c5.large-prices-ap-northeast-2

```
{
  "Timestamp": "2018-05-06T16:35:44.000Z",
  "AvailabilityZone": "ap-northeast-2c",
  "InstanceType": "c5.large",
  "ProductDescription": "Linux/UNIX",
  "SpotPrice": "0.022600",
  "Index": 498,
  "Zone": "ap-northeast-2"
}
```

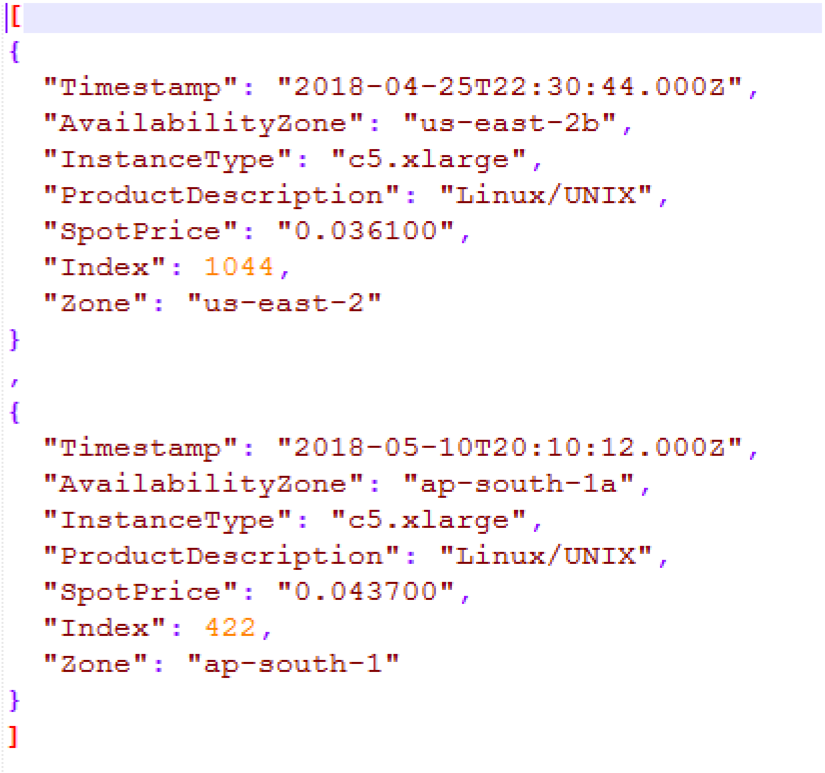
findMin.jar



c5.large
c5.xlarge
m5.2xlarge
m5.large
m5.xlarge
t2.large
t2.medium
t2.micro
t2.small
t2.xlarge



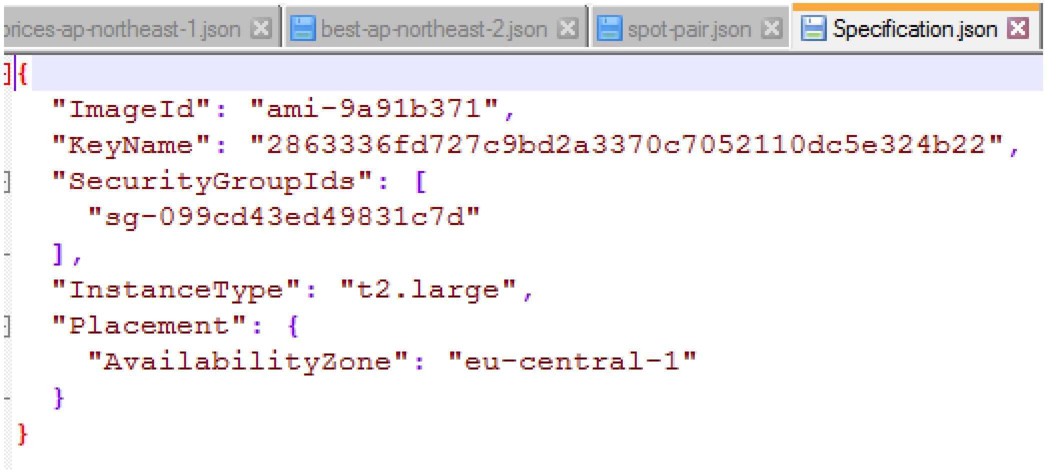
ap-northeast-1
ap-northeast-2
ap-south-1
ap-southeast-1
ap-southeast-2
ca-central-1
eu-central-1
eu-west-1
eu-west-2
eu-west-3
sa-east-1
us-east-1
us-east-2
us-west-1
us-west-2
spot-pair



```
[
{
  "Timestamp": "2018-04-25T22:30:44.000Z",
  "AvailabilityZone": "us-east-2b",
  "InstanceType": "c5.xlarge",
  "ProductDescription": "Linux/UNIX",
  "SpotPrice": "0.036100",
  "Index": 1044,
  "Zone": "us-east-2"
}
,
{
  "Timestamp": "2018-05-10T20:10:12.000Z",
  "AvailabilityZone": "ap-south-1a",
  "InstanceType": "c5.xlarge",
  "ProductDescription": "Linux/UNIX",
  "SpotPrice": "0.043700",
  "Index": 422,
  "Zone": "ap-south-1"
}
]
```

main.sh

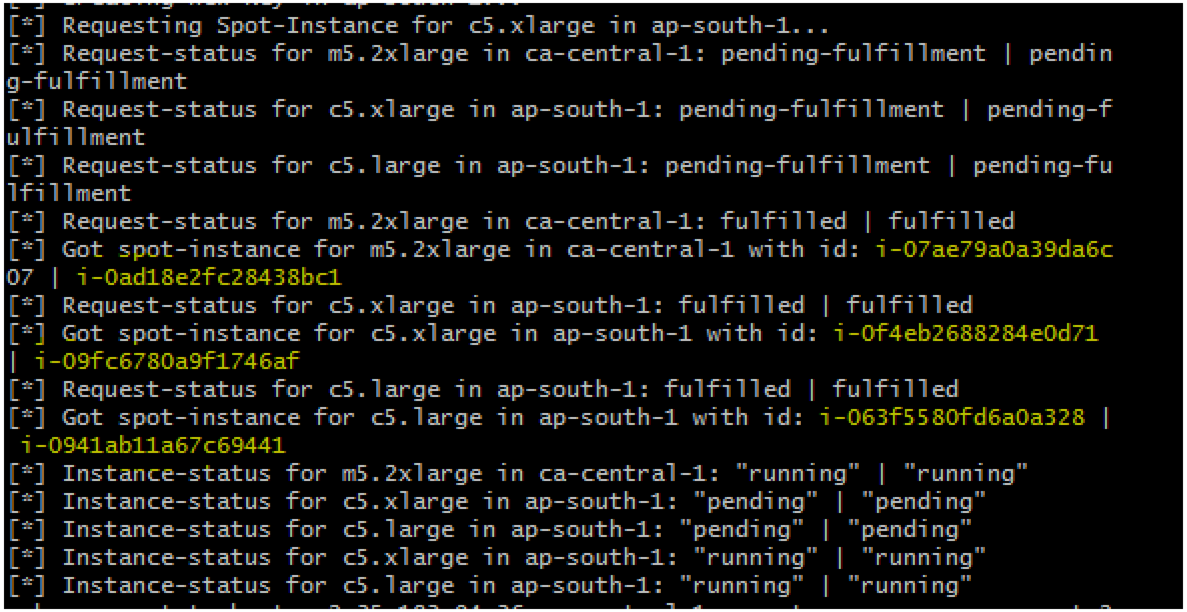
- 4. Prepare Specification file for spot-request (Sec-group, Key, Image-Id, ect...)



```
{
  "ImageId": "ami-9a91b371",
  "KeyName": "2863336fd727c9bd2a3370c7052110dc5e324b22",
  "SecurityGroupIds": [
    "sg-099cd43ed49831c7d"
  ],
  "InstanceType": "t2.large",
  "Placement": {
    "AvailabilityZone": "eu-central-1"
  }
}
```

main.sh

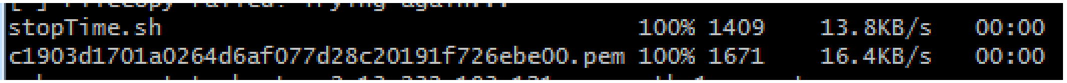
- 5. Send a request for 2 instances in same region and try to acquire them (get running instance ids)



```
[*] Requesting Spot-Instance for c5.xlarge in ap-south-1...
[*] Request-status for m5.2xlarge in ca-central-1: pending-fulfillment | pendin
g-fulfillment
[*] Request-status for c5.xlarge in ap-south-1: pending-fulfillment | pending-f
ulfillment
[*] Request-status for c5.large in ap-south-1: pending-fulfillment | pending-fu
lfillment
[*] Request-status for m5.2xlarge in ca-central-1: fulfilled | fulfilled
[*] Got spot-instance for m5.2xlarge in ca-central-1 with id: i-07ae79a0a39da6c
07 | i-0ad18e2fc28438bc1
[*] Request-status for c5.xlarge in ap-south-1: fulfilled | fulfilled
[*] Got spot-instance for c5.xlarge in ap-south-1 with id: i-0f4eb2688284e0d71
| i-09fc6780a9f1746af
[*] Request-status for c5.large in ap-south-1: fulfilled | fulfilled
[*] Got spot-instance for c5.large in ap-south-1 with id: i-063f5580fd6a0a328 |
 i-0941ab11a67c69441
[*] Instance-status for m5.2xlarge in ca-central-1: "running" | "running"
[*] Instance-status for c5.xlarge in ap-south-1: "pending" | "pending"
[*] Instance-status for c5.large in ap-south-1: "pending" | "pending"
[*] Instance-status for c5.xlarge in ap-south-1: "running" | "running"
[*] Instance-status for c5.large in ap-south-1: "running" | "running"
```

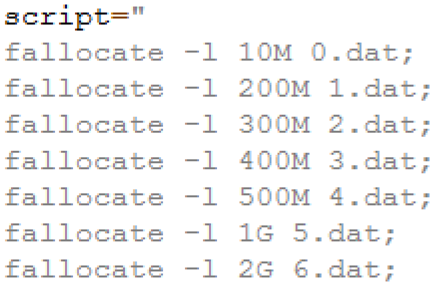
6. startVm2Vm.sh

- 6.1 Setup aws config
- 6.2 Connect to one instance and transfer key and the measurement script



```
stopTime.sh                                      100% 1409     13.8KB/s   00:00
c1903d1701a0264d6af077d28c20191f726ebe00.pem 100% 1671     16.4KB/s   00:00
```

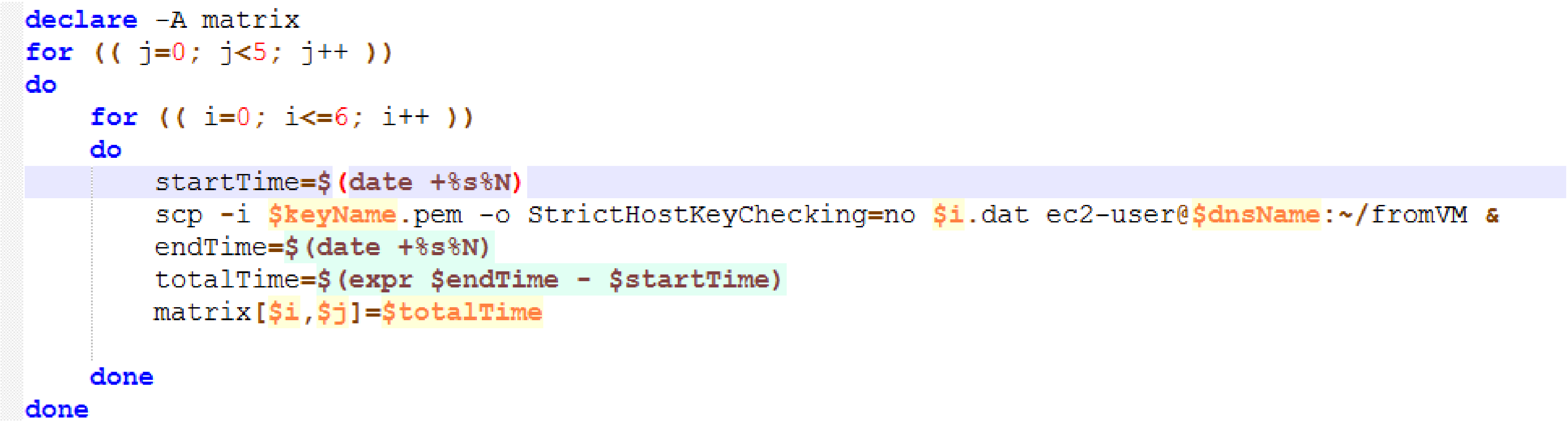
- 6.3 Once connected generate files and start measurement script



```
script="
fallocate -l 10M 0.dat;
fallocate -l 200M 1.dat;
fallocate -l 300M 2.dat;
fallocate -l 400M 3.dat;
fallocate -l 500M 4.dat;
fallocate -l 1G 5.dat;
fallocate -l 2G 6.dat;
```

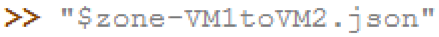
6.4 stopTime.sh

- 6.4.1 Transfer 7 files 5 times and take average time for each transmission of a file



```
declare -A matrix
for (( j=0; j<5; j++ ))
do
    for (( i=0; i<=6; i++ ))
    do
        startTime=$(date +%s%N)
        scp -i $keyName.pem -o StrictHostKeyChecking=no $i.dat ec2-user@$dnsName:~/fromVM &
        endTime=$(date +%s%N)
        totalTime=$(expr $endTime - $startTime)
        matrix[$i,$j]=$totalTime

    done
done
```

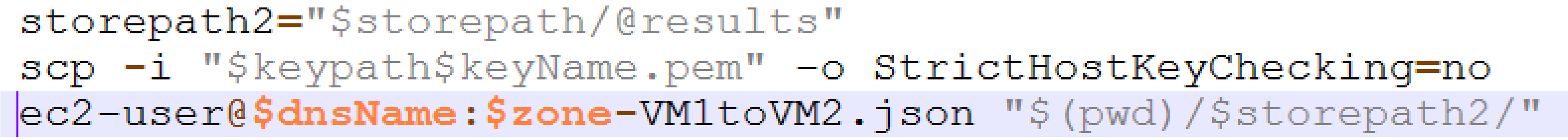
- 6.4.2 Save the result as a file



```
>> "$zone-VM1toVM2.json"
```

startVm2Vm.sh

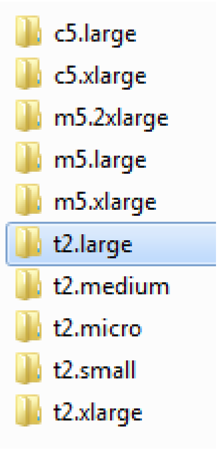
- 6.5 download the result file and fill it with necessary information



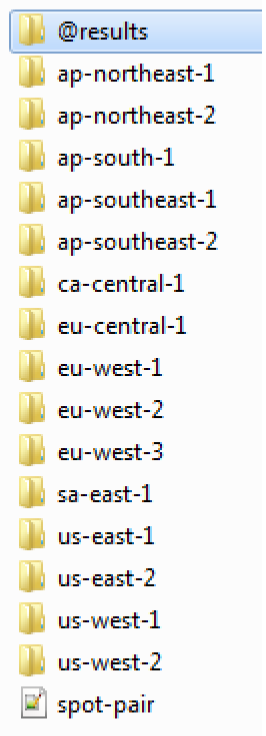
```
storepath2="$storepath/@results"
scp -i "$keypath$keyName.pem" -o StrictHostKeyChecking=no
ec2-user@$dnsName:$zone-VM1toVM2.json "$(pwd)/$storepath2/"
```

7. cost-performance.jar

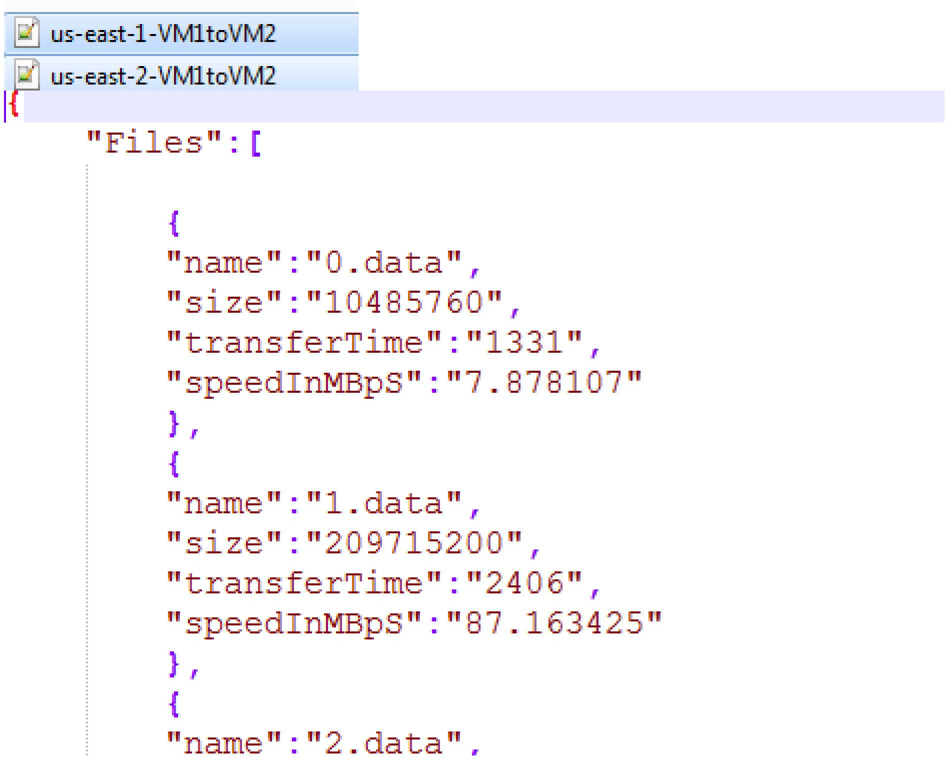
- 7.1 crawl through every instance-type folder and check the results



c5.large
c5.xlarge
m5.2xlarge
m5.large
m5.xlarge
t2.large
t2.medium
t2.micro
t2.small
t2.xlarge



@results
ap-northeast-1
ap-northeast-2
ap-south-1
ap-southeast-1
ap-southeast-2
ca-central-1
eu-central-1
eu-west-1
eu-west-2
eu-west-3
sa-east-1
us-east-1
us-east-2
us-west-1
us-west-2
spot-pair

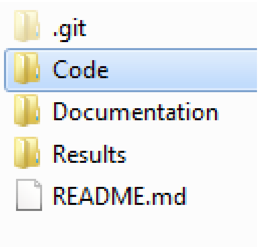


us-east-1-VM1toVM2
us-east-2-VM1toVM2

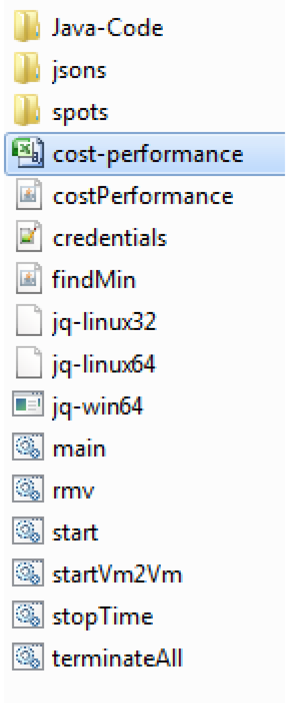
```
{
    "Files":[

        {
        "name":"0.data",
        "size":"10485760",
        "transferTime":"1331",
        "speedInMBpS":"7.878107"
        },
        {
        "name":"1.data",
        "size":"209715200",
        "transferTime":"2406",
        "speedInMBpS":"87.163425"
        },
        {
        "name":"2.data",
```

cost-performance.jar

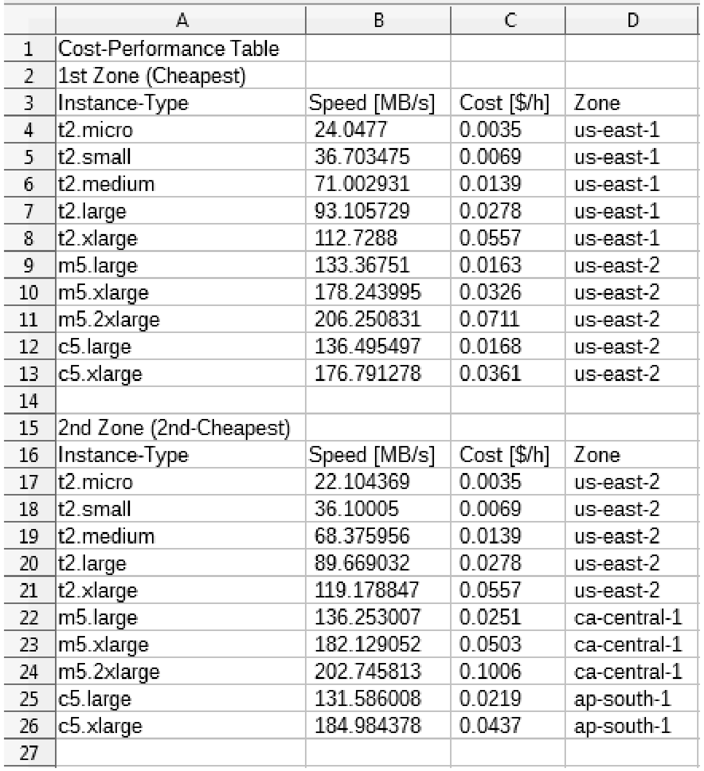
- 7.2 create one single CSV-File for all results with the 2 zones of each instance-type



.git
Code
Documentation
Results
README.md



Java-Code
jsons
spots
cost-performance
costPerformance
credentials
findMin
jq-linux32
jq-linux64
jq-win64
main
rmv
start
startVm2Vm
stopTime
terminateAll



	A	B	C	D
1	Cost-Performance Table			
2	1st Zone (Cheapest)			
3	Instance-Type	Speed [MB/s]	Cost [$/h]	Zone
4	t2.micro	24.0477	0.0035	us-east-1
5	t2.small	36.703475	0.0069	us-east-1
6	t2.medium	71.002931	0.0139	us-east-1
7	t2.large	93.105729	0.0278	us-east-1
8	t2.xlarge	112.7288	0.0557	us-east-1
9	m5.large	133.36751	0.0163	us-east-2
10	m5.xlarge	178.243995	0.0326	us-east-2
11	m5.2xlarge	206.250831	0.0711	us-east-2
12	c5.large	136.495497	0.0168	us-east-2
13	c5.xlarge	176.791278	0.0361	us-east-2
14				
15	2nd Zone (2nd-Cheapest)			
16	Instance-Type	Speed [MB/s]	Cost [$/h]	Zone
17	t2.micro	22.104369	0.0035	us-east-2
18	t2.small	36.10005	0.0069	us-east-2
19	t2.medium	68.375956	0.0139	us-east-2
20	t2.large	89.669032	0.0278	us-east-2
21	t2.xlarge	119.178847	0.0557	us-east-2
22	m5.large	136.253007	0.0251	ca-central-1
23	m5.xlarge	182.129052	0.0503	ca-central-1
24	m5.2xlarge	202.745813	0.1006	ca-central-1
25	c5.large	131.586008	0.0219	ap-south-1
26	c5.xlarge	184.984378	0.0437	ap-south-1
27				