Fault Tolerance - midterm

Nikhil, Jessica, Fabian

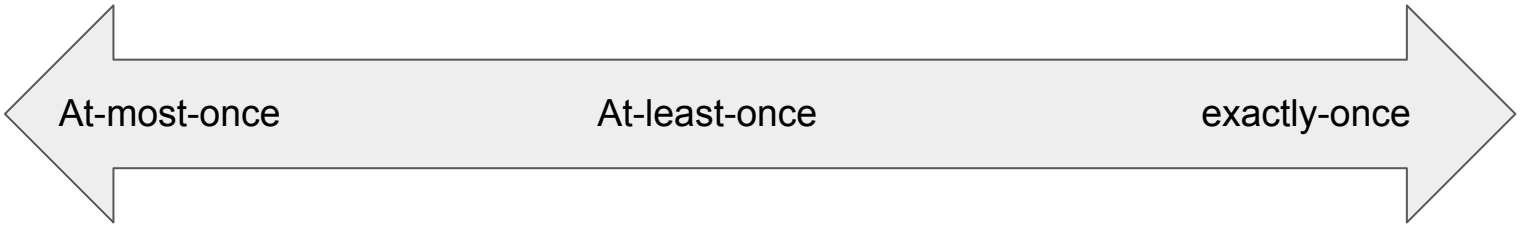
Motivation

- Fault Tolerance - ability to continue functioning in the event of failure.

- Objective: provide dependability

- Failure masking Techniques:
  - Replication
  - Checkpointing

Data handling guarantees

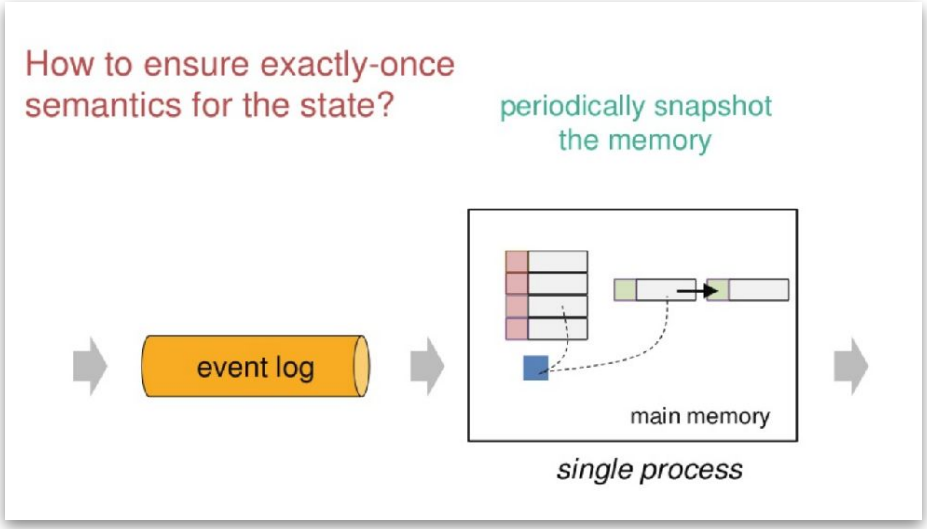


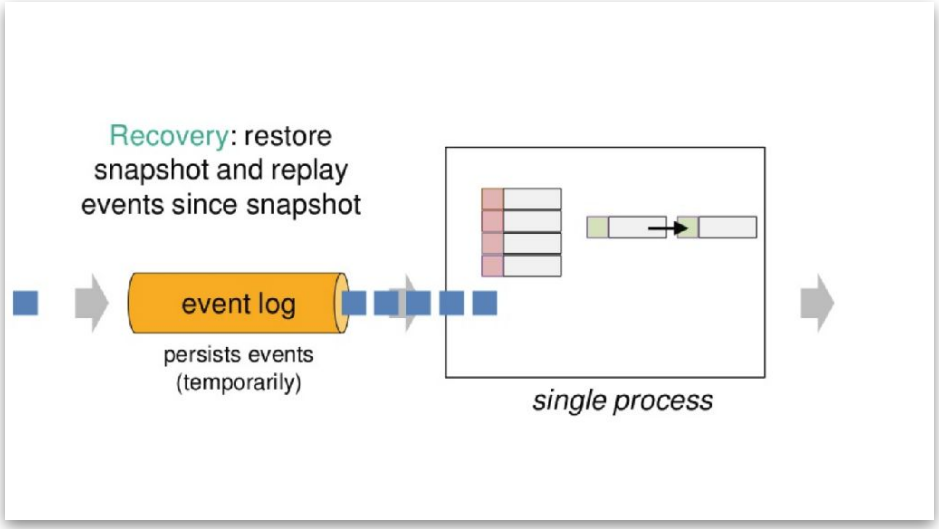
Messages may be lost

Messages may be duplicated but not lost

Messages neither lost nor duplicated

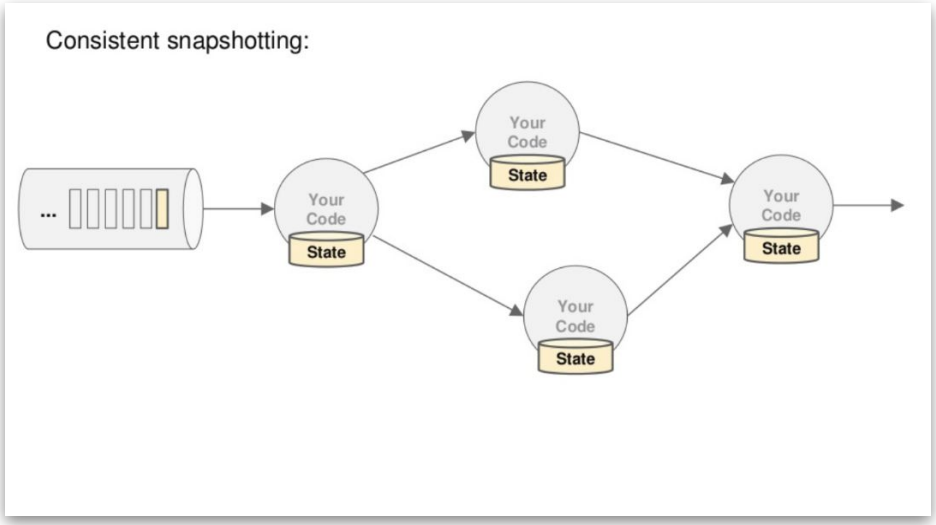
Flink fault tolerance

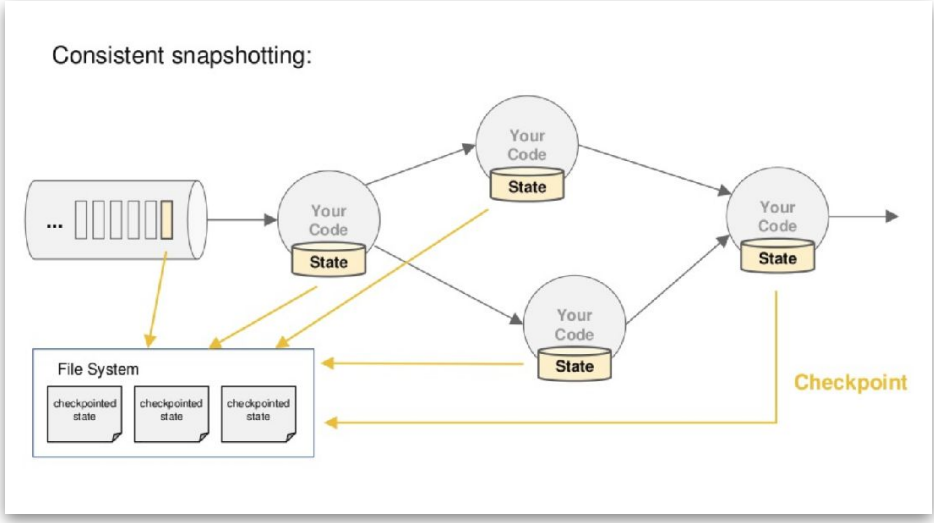




Flink checkpointing

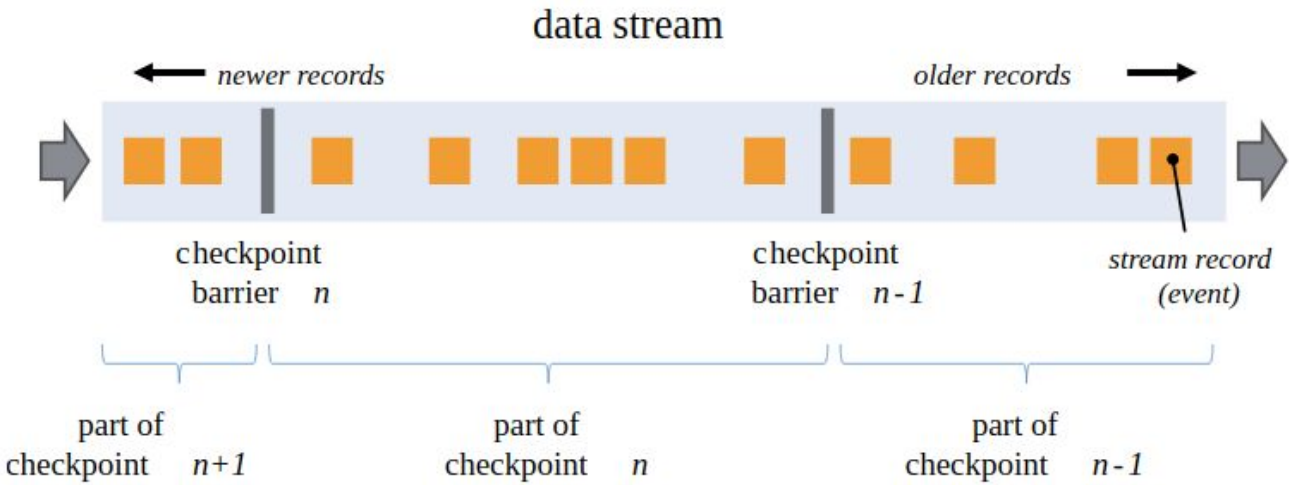
- Draws consistent snapshots of the distributed data stream and operator state.

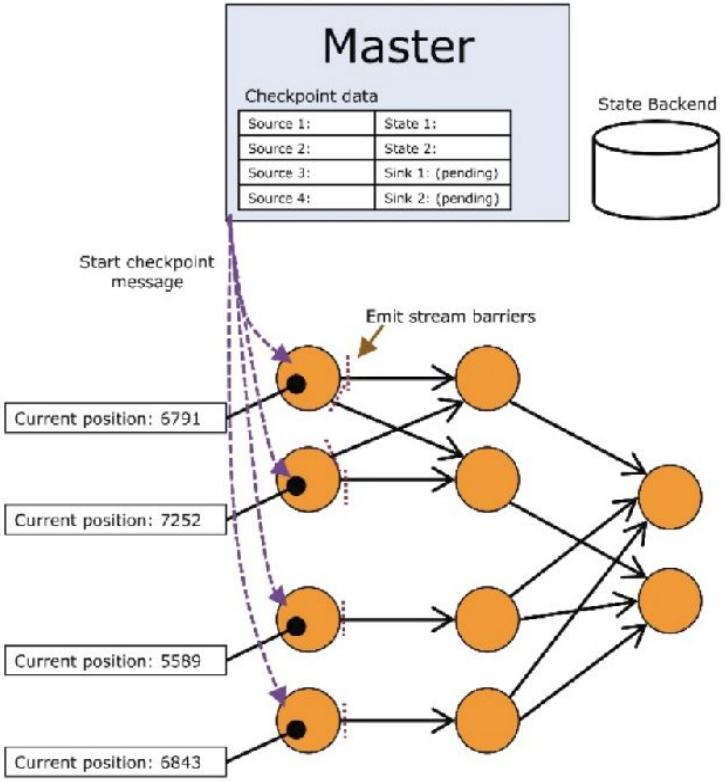




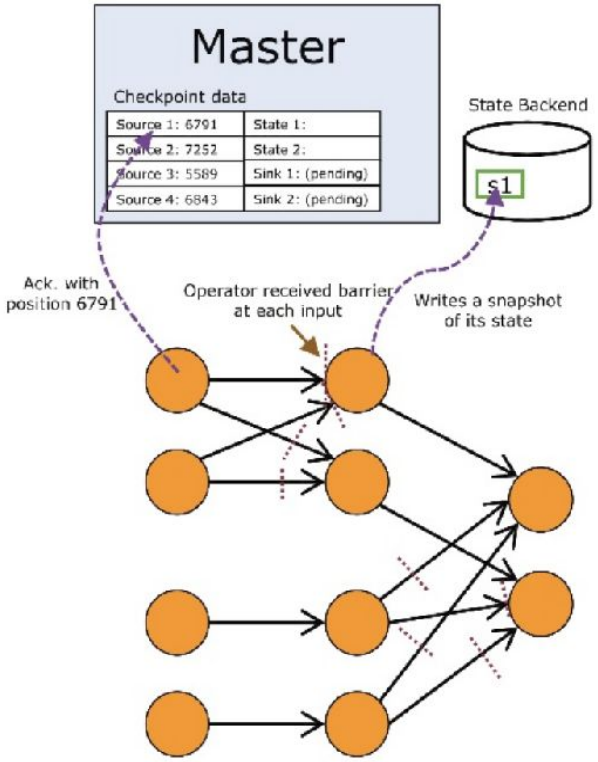
Checkpoint barriers

- Markers for checkpoints
- Injected into Datastream

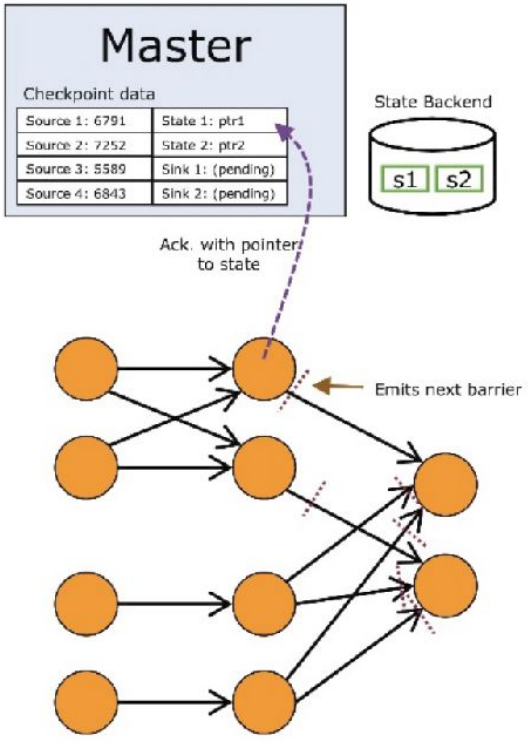




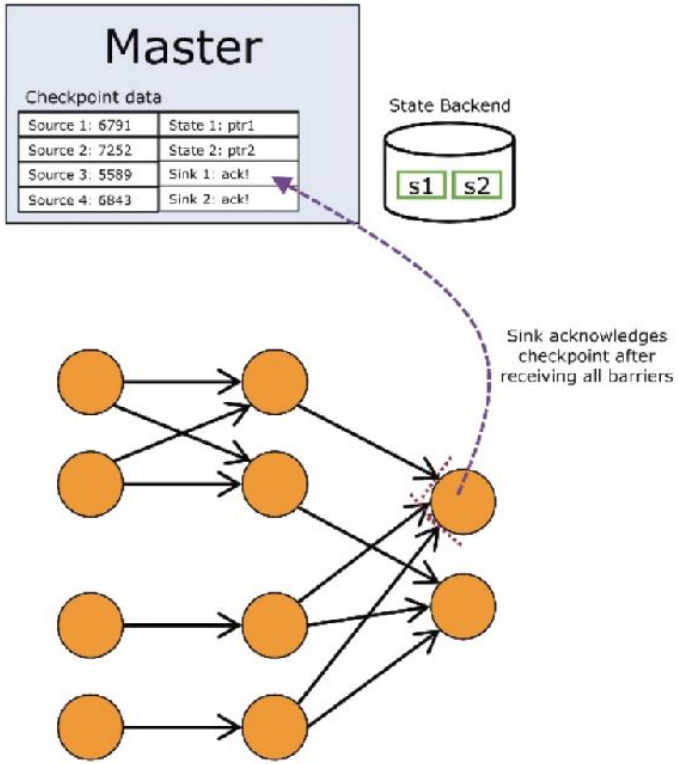
Starting Checkpoint



Checkpoint in Progress

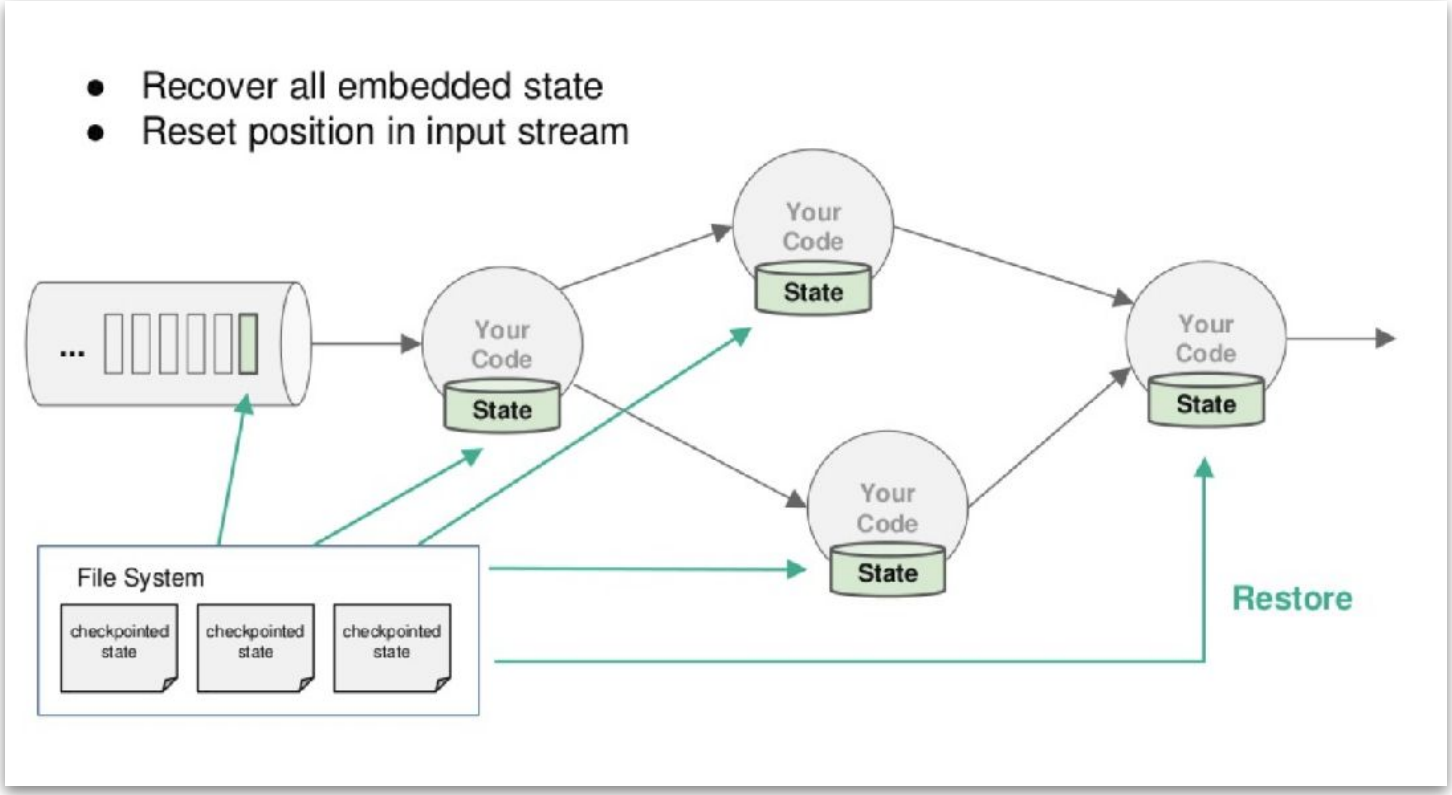


Checkpoint in Progress

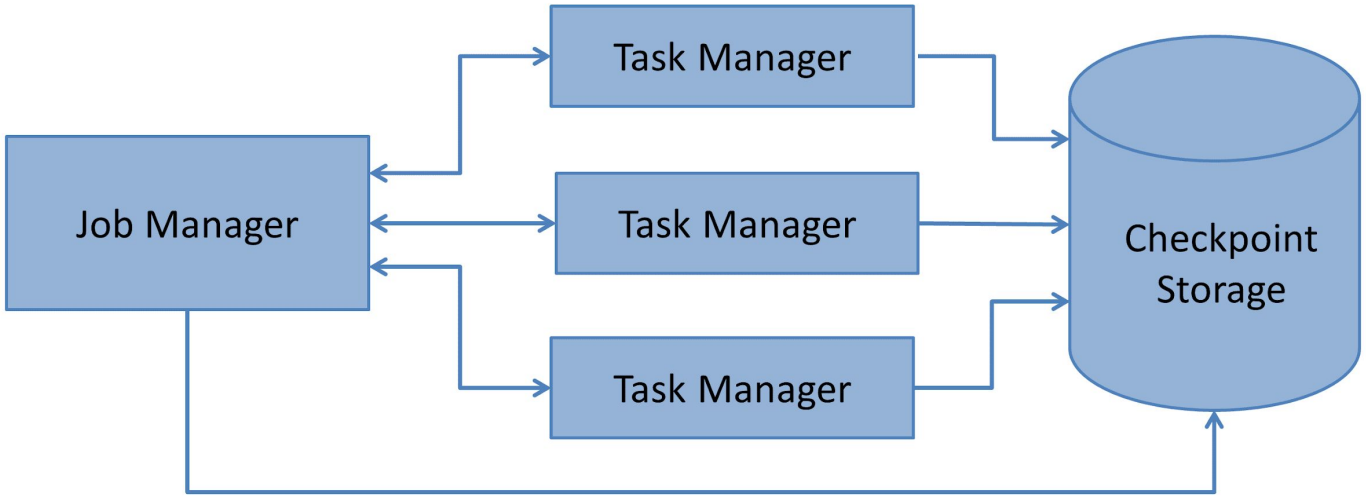


Checkpoint Completed

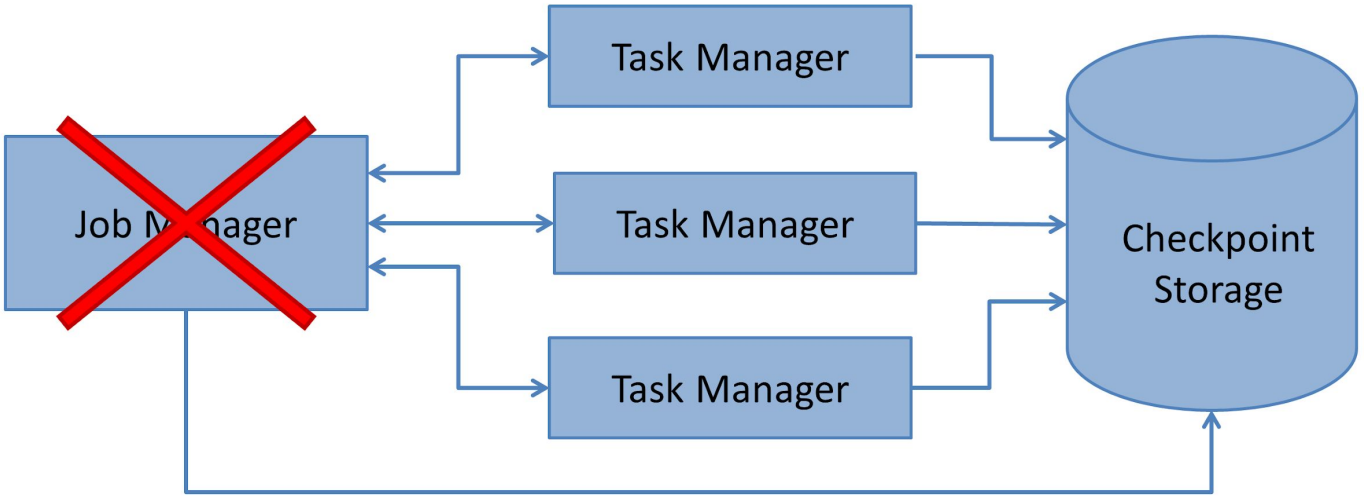
Flink Recovery



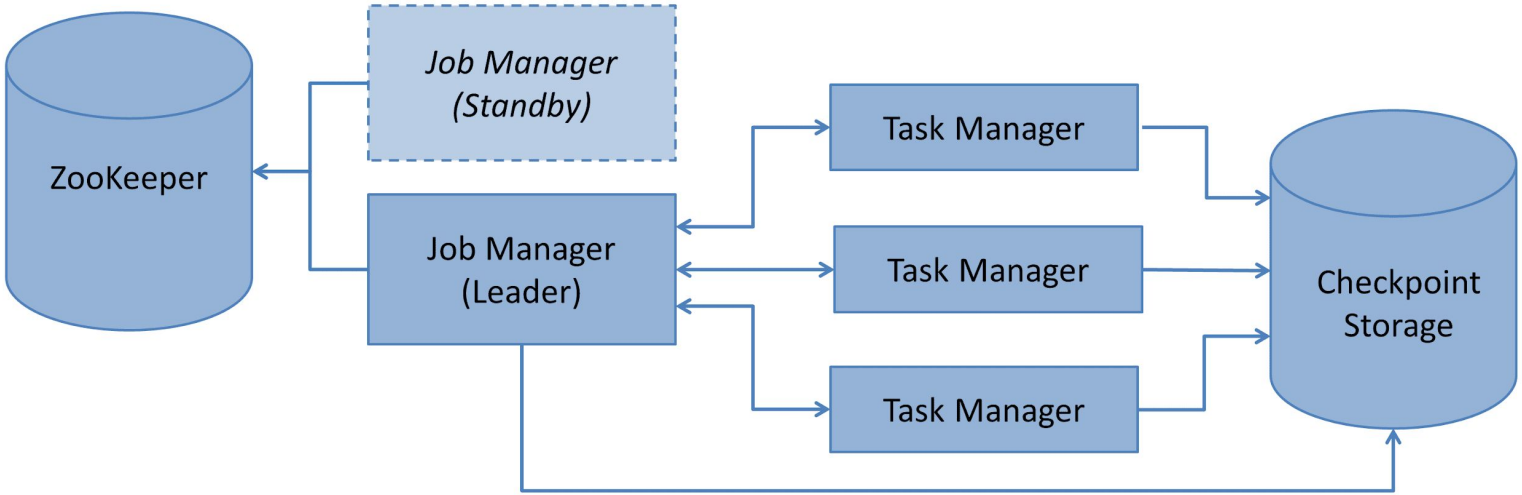
Single Point of Failure



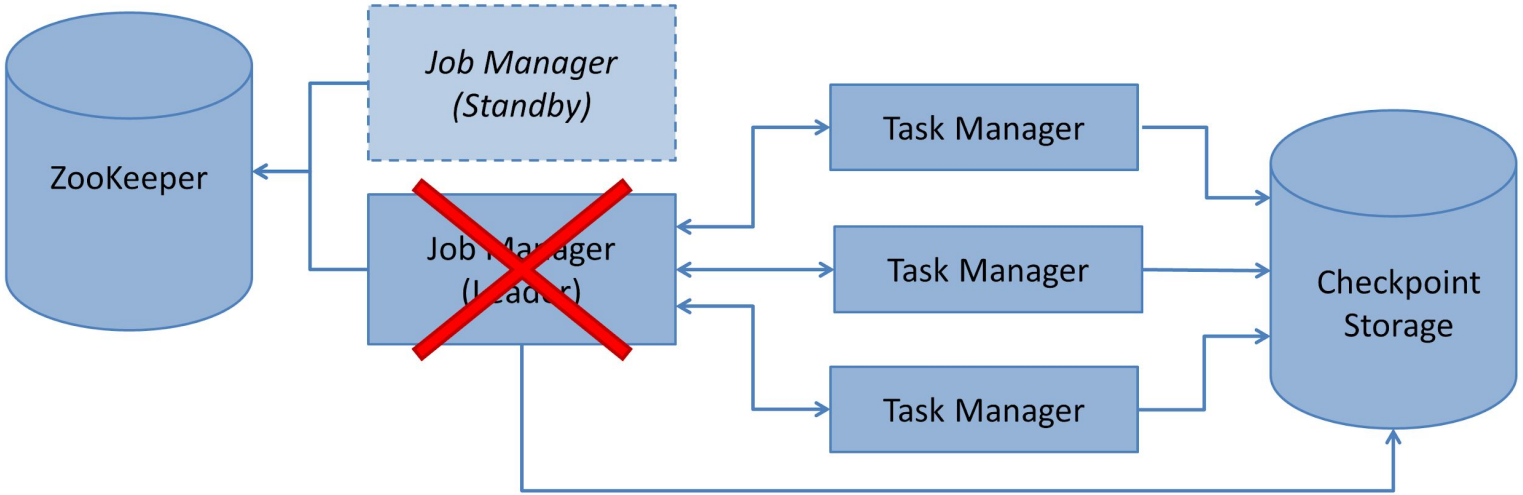
Single Point of Failure



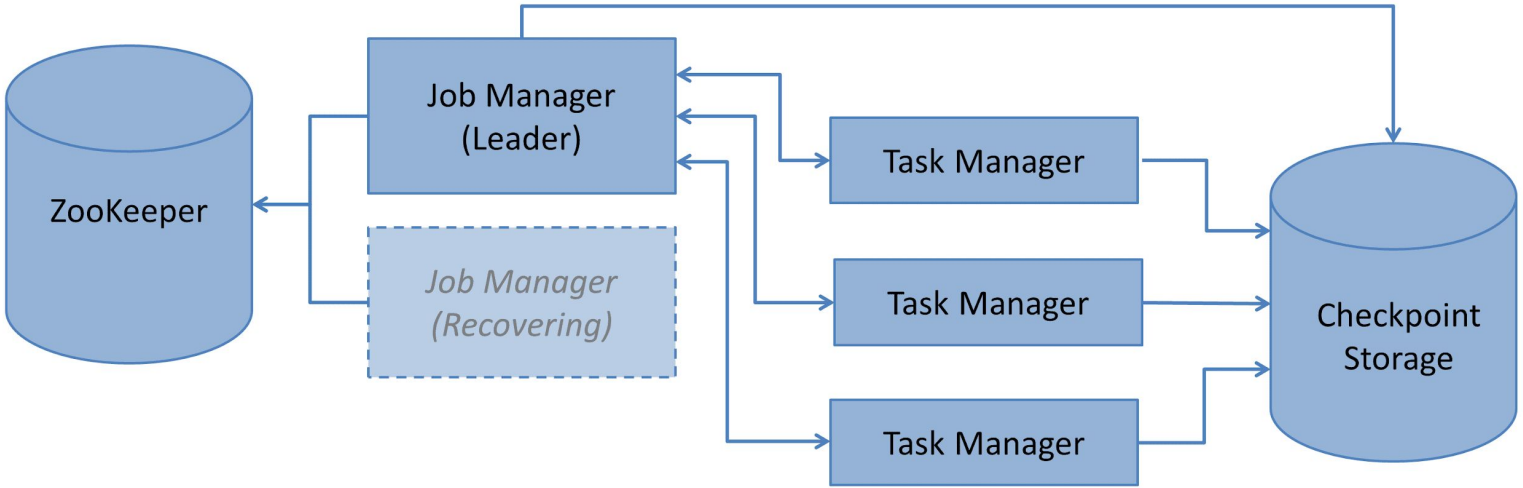
High Availability Mode (Standalone)



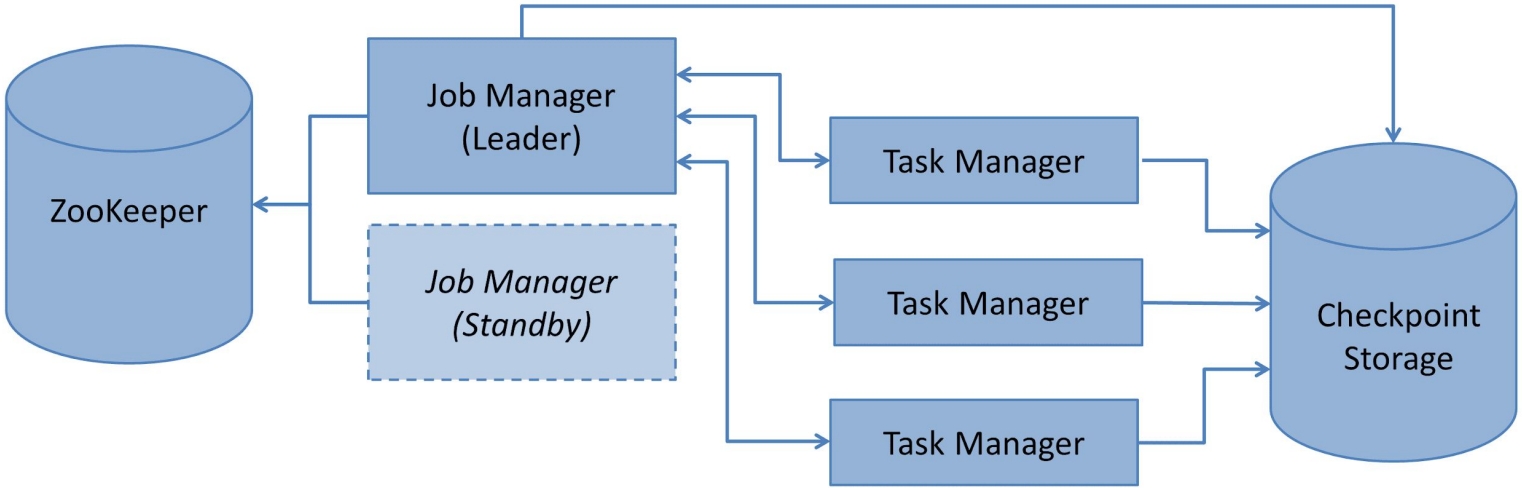
High Availability Mode (Standalone)



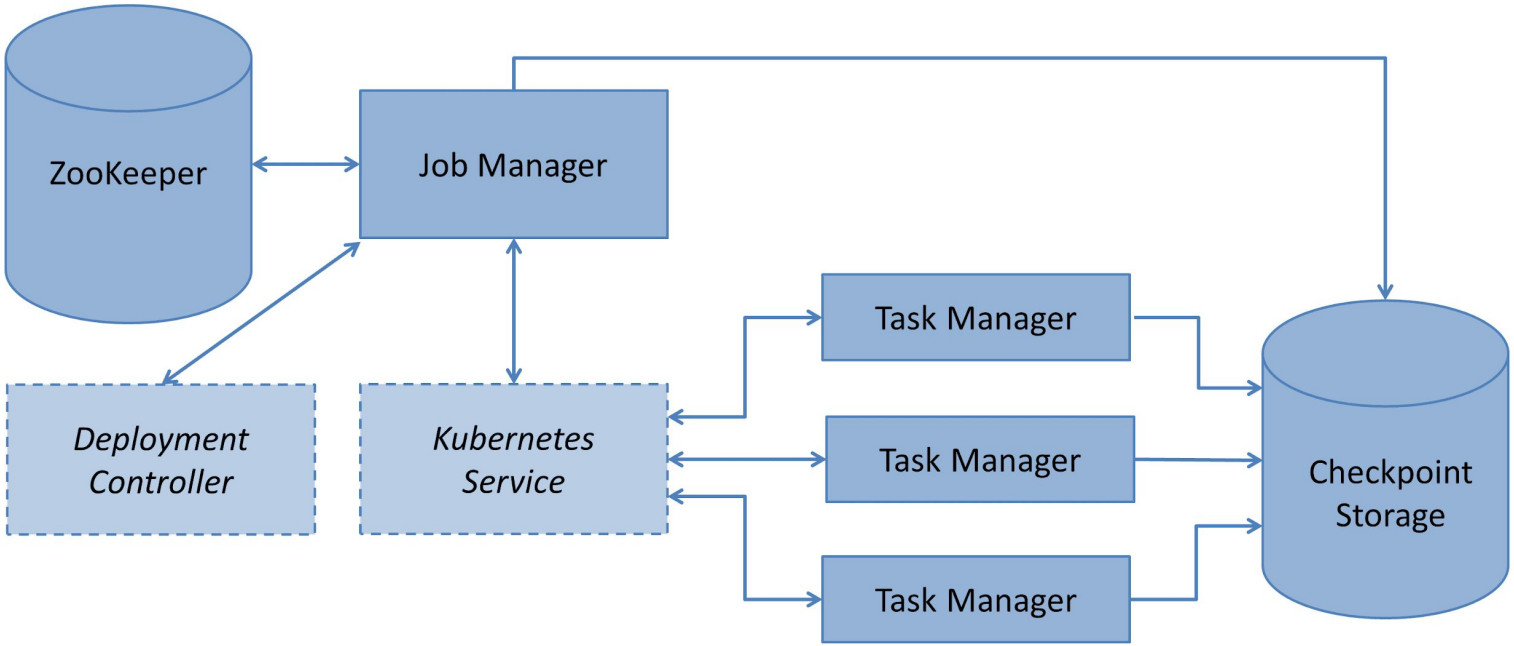
High Availability Mode (Standalone)



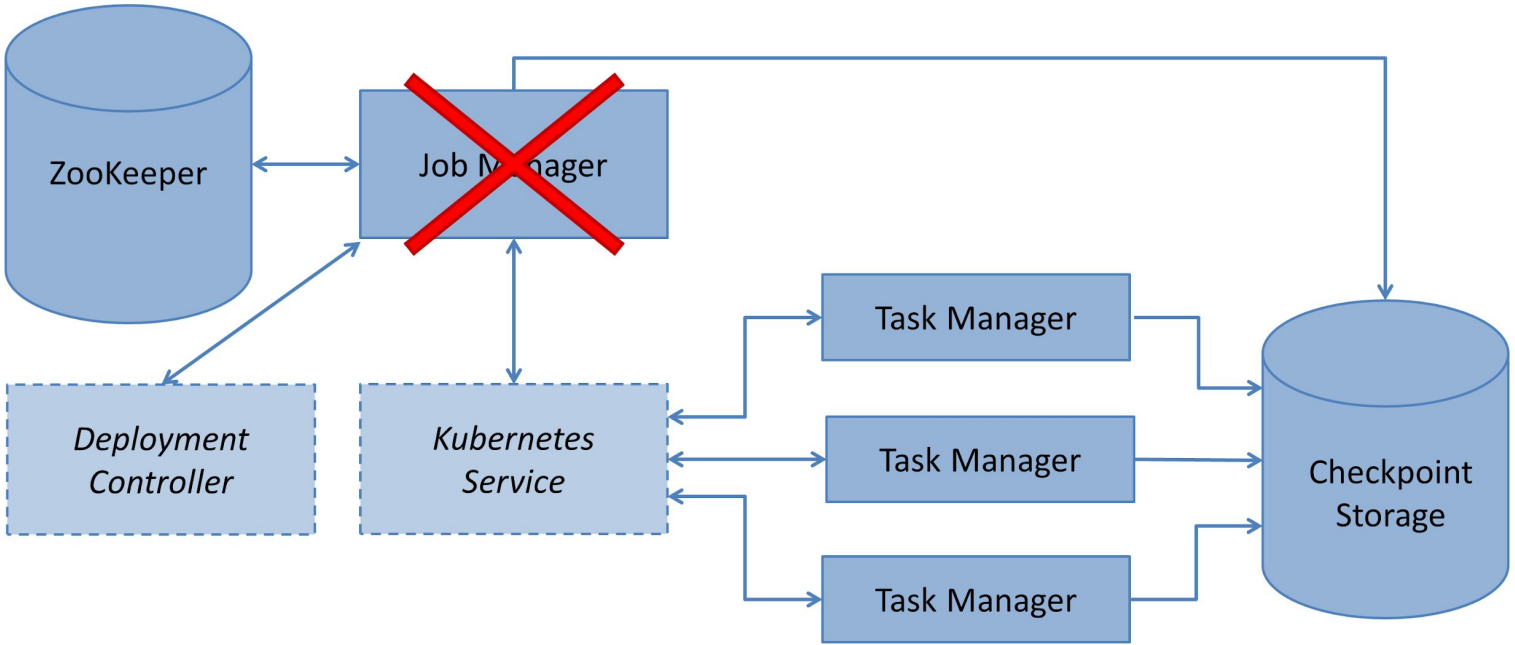
High Availability Mode (Standalone)



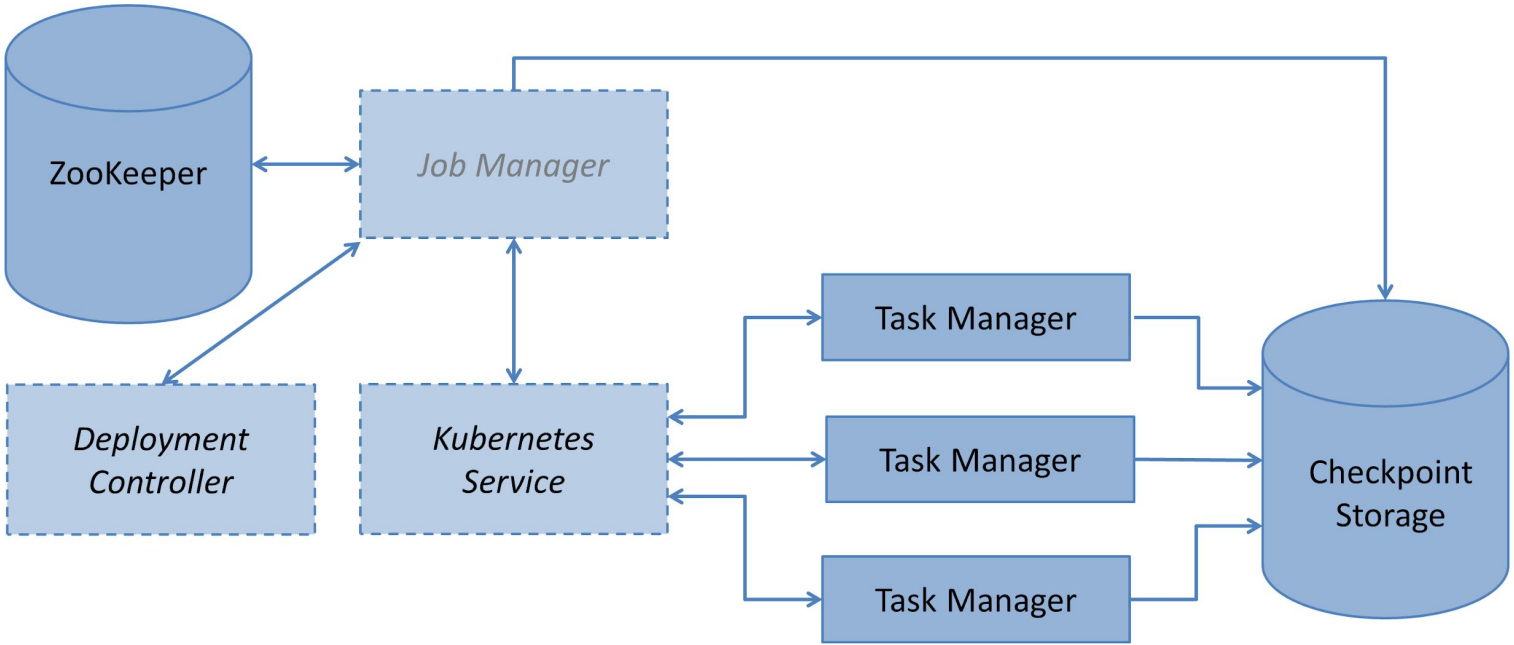
High Availability Mode (w/ Kubernetes or Yarn)



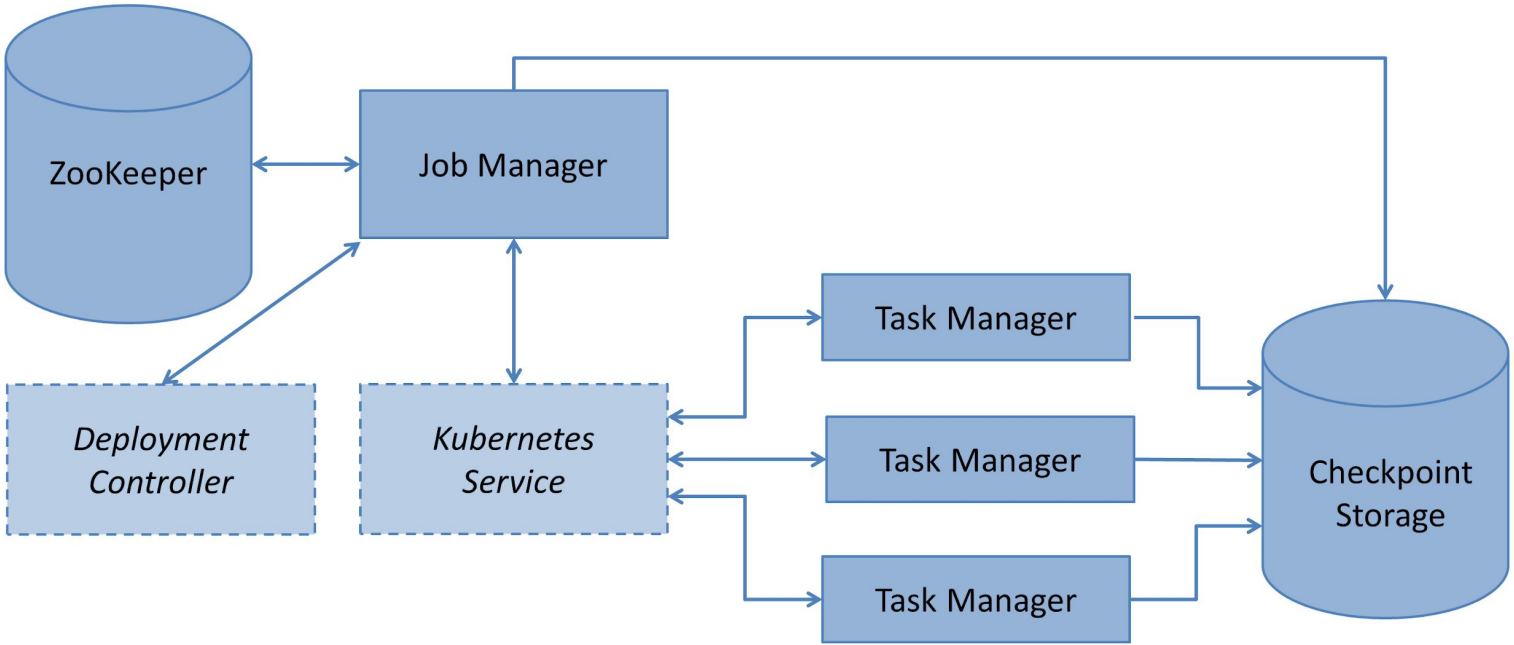
High Availability Mode (w/ Kubernetes or Yarn)



High Availability Mode (w/ Kubernetes or Yarn)



High Availability Mode (w/ Kubernetes or Yarn)



Our approach

- Making transformations stateful, enabling checkpointing in the applications and configuring restart strategies.
- Focus is on heterogeneous env, Multiple Job managers for high availability where one as leader and others as standby. Zookeeper can be used in case of Master(Job manager) failure to re-elect the Manager from Standby,
- To reduce latency we want the checkpoints/snapshots to be stored as close as possible to the node that fails,
- If possible add machine learning for adaptive fault tolerance.

Initial planning (next steps)

Cluster of Docker containers. First Single host for homogeneous env then multi host for heterogeneous env

Running flink cluster on those containers and run jobs.

Chaos testing using Pumba to check how application behaves in case of node or network failures.

Docker (incl. NETEM , cgroups (het. env))

Zookeeper, Kubernetes

Demo (Homogeneous Environment)

```
d071242@C02T427GHF1R ~> docker network ls
NETWORK ID          NAME                DRIVER              SCOPE
488a762ee4e8        bridge              bridge              local
c0e7560dc8d2        flink               bridge              local
267bb5c387c2        host                host                local
d071242@C02T427GHF1R ~/Flink> docker-compose up --scale taskmanager=3
flink_jobmanager_1 is up-to-date
Starting flink_taskmanager_1 ... done
Creating flink_taskmanager_2 ... done
Creating flink_taskmanager_3 ... done
Attaching to flink_jobmanager_1, flink_taskmanager_1, flink_taskmanager_3, flink_taskmanager_2
```

```
d071242@C02T427GHF1R ~> docker network ls
NETWORK ID          NAME                DRIVER              SCOPE
488a762ee4e8        bridge              bridge              local
c0e7560dc8d2        flink               bridge              local
dadd2c02367e        flink_default       bridge              local
267bb5c387c2        host                host                local
d071242@C02T427GHF1R ~> docker exec -t -i flink_jobmanager_1 flink run /opt/flink/examples/streaming/SocketWindowWordCount.jar --hostname
 10.183.176.71 --port 9000
Starting execution of program
Program execution finished
Job with JobID a5003bad6cf9efe11efe153de4727f2c has finished.
Job Runtime: 1371036 ms
```

```
"Containers": {
    "2e8caef8ca37dbff59011bc714549842432b991f9bdfbf9c22c50d823628dff8": {
        "Name": "flink_taskmanager_1",
        "EndpointID": "d8f3c8484c70dae0065d7c6a9d8357a7505134b33a2027062d96ba17dfceff18",
        "MacAddress": "02:42:ac:1f:00:03",
        "IPv4Address": "172.31.0.3/16",
        "IPv6Address": ""
    },
    "53b8cfecb5c8db25953d0d2b29ad818dae9559831f92fc5f352bddf8e92a3f2f": {
        "Name": "flink_taskmanager_2",
        "EndpointID": "036af5632cce8c3892357e64fcd933a6091399f98ae9531760c86f24429d8088",
        "MacAddress": "02:42:ac:1f:00:05",
        "IPv4Address": "172.31.0.5/16",
        "IPv6Address": ""
    },
    "84d14c4c2485b44584140773e9a8a121a3c93ce484b670432289ec0f892c648d": {
        "Name": "flink_jobmanager_1",
        "EndpointID": "cd36e9970c6dc9b66ab5b73437059dee42918030d671f20e7546c04b8090c115",
        "MacAddress": "02:42:ac:1f:00:02",
        "IPv4Address": "172.31.0.2/16",
        "IPv6Address": ""
    },
    "cc90bcfa728f956f2cf2f030baca45c6e0c46257e796200bc03db9c5cbae8d0f": {
        "Name": "flink_taskmanager_3",
        "EndpointID": "04fcbc5bdebca1259109de725f5c20a4cb610e2be7655dcaa5ab9b137c1755a2",
        "MacAddress": "02:42:ac:1f:00:04",
        "IPv4Address": "172.31.0.4/16",
        "IPv6Address": ""
    }
```

```
d071242@C02T427GHF1R ~> nc -l 9000
This is to test if wordcount is working properly
More texts to test
test test
d071242@C02T427GHF1R ~> nc -l 9000
Hello keep testing please
d071242@C02T427GHF1R ~> nc -l 9000
Some more text to test
We would like to check how it behaves when we stop the containers using Pumba
d071242@C02T427GHF1R ~> nc -l 9000
Now we will write more stuff to check how it executes the job when TM2 has been stopped
Keep writing
hello hello
```

```
d071242@C02T427GHF1R ~> pumba stop flink_taskmanager_1
d071242@C02T427GHF1R ~> pumba stop flink_taskmanager_2
```

```
taskmanager_1  | This : 1
taskmanager_1  | properly : 1
taskmanager_1  | working : 1
taskmanager_1  | wordcount : 1
taskmanager_1  | if : 1
taskmanager_1  | test : 1
taskmanager_1  | to : 1
taskmanager_1  | is : 2
taskmanager_1  | More : 1
taskmanager_1  | test : 3
taskmanager_1  | to : 1
taskmanager_1  | texts : 1
```

```
flink_taskmanager_1 exited with code 143
```

```
taskmanager_2  | Hello : 1
taskmanager_2  | please : 1
taskmanager_2  | testing : 1
taskmanager_2  | keep : 1
```

```
taskmanager_2  | Some : 1
taskmanager_2  | test : 1
taskmanager_2  | to : 1
taskmanager_2  | text : 1
taskmanager_2  | more : 1
taskmanager_2  | We : 1
taskmanager_2  | Pumba : 1
taskmanager_2  | using : 1
taskmanager_2  | containers : 1
taskmanager_2  | the : 1
taskmanager_2  | stop : 1
taskmanager_2  | we : 1
taskmanager_2  | when : 1
taskmanager_2  | behaves : 1
taskmanager_2  | it : 1
taskmanager_2  | how : 1
taskmanager_2  | check : 1
taskmanager_2  | to : 1
taskmanager_2  | like : 1
taskmanager_2  | would : 1
```

```
flink_taskmanager_2 exited with code 143
```

```
taskmanager_3  | Now : 1
taskmanager_3  | writing : 1
taskmanager_3  | Keep : 1
taskmanager_3  | stopped : 1
taskmanager_3  | been : 1
taskmanager_3  | has : 1
taskmanager_3  | TM2 : 1
taskmanager_3  | when : 1
taskmanager_3  | job : 1
taskmanager_3  | the : 1
taskmanager_3  | executes : 1
taskmanager_3  | it : 1
taskmanager_3  | how : 1
taskmanager_3  | check : 1
taskmanager_3  | to : 1
taskmanager_3  | stuff : 1
taskmanager_3  | more : 1
taskmanager_3  | write : 1
taskmanager_3  | will : 1
taskmanager_3  | we : 1
taskmanager_3  | hello : 2
```

```
"Containers": {
    "84d14c4c2485b44584140773e9a8a121a3c93ce484b670432289ec0f892c648d": {
        "Name": "flink_jobmanager_1",
        "EndpointID": "cd36e9970c6dc9b66ab5b73437059dee42918030d671f20e7546c04b8090c115",
        "MacAddress": "02:42:ac:1f:00:02",
        "IPv4Address": "172.31.0.2/16",
        "IPv6Address": ""
    },
    "cc90bcfa728f956f2cf2f030baca45c6e0c46257e796200bc03db9c5cbae8d0f": {
        "Name": "flink_taskmanager_3",
        "EndpointID": "04fcbc5bdebca1259109de725f5c20a4cb610e2be7655dcaa5ab9b137c1755a2",
        "MacAddress": "02:42:ac:1f:00:04",
        "IPv4Address": "172.31.0.4/16",
        "IPv6Address": ""
    }
},
```

Interesting Points

When Pumba stops the container netcat listening on port 9000 exits automatically.

Job manager shows “Association failed” Caused by: [No route to host] which ultimately switches the job from RUNNING to FAILED.

When we re-run nc -l 9000 and write something. We don’t see anything in the logs. However, the moment we re-run the job Job manager assigns the task to another available task manager which spits this word count output.

Questions?