BAB 4

IMPLEMENTASI DAN PEMBAHASAN

4.1 Implementasi

Pada bagian ini peneliti akan menjelaskan langkah-langkah membangun docker swarm *cluster* sebagai server hosting yang memiliki tingkat ketersediaan yang tinggi atau memiliki *High Availability* yang tinggi.

4.1.1 Docker Swarm

Gambar 4.1 merupakan konfigurasi file /etc/hosts yang digunakan untuk melakukan resolve hostname node ke ip address. Konfigurasi ini dilakukan pada ketiga node

```
127.0.0.1 localhost
10.10.10.75 manager
10.10.10.76 worker1
10.10.10.77 worker2
```

Gambar 4.1 Konfigurasi File /etc/hosts

Selanjutnya melakukan validasi apakah konfigurasi file /etc/hosts sudah berjalan dengan menggunakan perintah ping ke hostname seperti pada gambar 4.2

```
root@manager:/home/ghofarnugroho# ping worker1
PING worker1 (10.10.10.76) 56(84) bytes of data.
64 bytes from worker1 (10.10.10.76): icmp_seq=1
ttl=64 time=0.652 ms
64 bytes from worker1 (10.10.10.76): icmp_seq=2
ttl=64 time=0.310 ms
64 bytes from worker1 (10.10.10.76): icmp_seq=3
ttl=64 time=0.323 ms
64 bytes from worker1 (10.10.10.76): icmp_seq=4
ttl=64 time=0.361 ms
--- worker1 ping statistics ---
4 packets transmitted, 4 received, 0% packet loss,
time 3068ms
rtt min/avg/max/mdev = 0.310/0.411/0.652/0.141 ms
root@manager:/home/ghofarnugroho# ping worker2
PING worker2 (10.10.10.77) 56(84) bytes of data.
64 bytes from worker2 (10.10.10.77): icmp_seq=1
ttl=64 time=0.686 ms
64 bytes from worker2 (10.10.10.77): icmp_seq=2
ttl=64 time=0.238 ms
64 bytes from worker2 (10.10.10.77): icmp_seq=3
ttl=64 time=0.260 ms
--- worker2 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss,
time 2031ms
rtt min/avg/max/mdev = 0.238/0.394/0.686/0.207 ms
```

Gambar 4.2 Perintah Validasi Konfigurasi File /etc/hosts

Gambar 4.3 merupakan perintah untuk melakukan update package pada Ubuntu Server. Perintah ini dijalankan pada ketiga node

```
root@manager:/home/ghofarnugroho# apt update
Hit:1 http://id.archive.ubuntu.com/ubuntu bionic
InRelease
Hit:2 http://id.archive.ubuntu.com/ubuntu bionic-
updates InRelease
Hit:3 http://id.archive.ubuntu.com/ubuntu bionic-
backports InRelease
Hit:4 http://id.archive.ubuntu.com/ubuntu bionic-
security InRelease
Reading package lists... Done
Building dependency tree
Reading state information... Done
85 packages can be upgraded. Run 'apt list --
upgradable' to see them.
```

Gambar 4.3 Perintah untuk *Update Package* Ubuntu Server

Selanjutnya, melakukan instalasi docker pada ketiga node dengan perintah pada gambar 4.4

```
root@manager:/home/ghofarnugroho# apt install
docker.io
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following additional packages will be
installed:
  bridge-utils containerd pigz runc ubuntu-fan
Suggested packages:
  ifupdown aufs-tools cgroupfs-mount | cgroup-lite
debootstrap docker-doc rinse zfs-fuse
  | zfsutils
The following NEW packages will be installed:
  bridge-utils containerd docker.io pigz runc
ubuntu-fan
0 upgraded, 6 newly installed, 0 to remove and 85
not upgraded.
Need to get 74.2 MB of archives.
After this operation, 360 MB of additional disk
space will be used.
Do you want to continue? [Y/n] y
```

Gambar 4.4 Perintah untuk Instalasi Docker

Langkah selanjutnya adalah melakukan generate token pada node manager agar node worker dapat melakukan join ke node manager dengan menggunakan token yang degenerate tersebut. Perintah untuk generate token terdapat pada gambar 4.5

```
root@manager:/home/ghofarnugroho# docker swarm init
Swarm initialized: current node
(mlc6zpodvemfxggpi87f1xvft) is now a manager.

To add a worker to this swarm, run the following
command:

    docker swarm join --token SWMTKN-1-
1kx8rryt9ocbb7masu4k71t7dw39n2axhmdyal2ggwcxdjcirr-
104t4w7p251lbu8fy1gtyytoq 10.10.10.75:2377

To add a manager to this swarm, run 'docker swarm
join-token manager' and follow the instructions.
```

Gambar 4.5 Perintah untuk *Generate* Token

Selanjutnya peneliti melakukan join cluster pada kedua node worker agar dapat dimanage oleh node manager seperti pada gambar 4.6

```
root@worker1:/home/ghofarnugroho# vi /etc/hosts
root@worker1:/home/ghofarnugroho# docker swarm join
--token SWMTKN-1-
1kx8rryt9ocbb7masu4k71t7dw39n2axhmdyal2ggwcxdjcirr-
104t4w7p251lbu8fy1gtyytoq 10.10.10.75:2377
This node joined a swarm as a worker.
```

Gambar 4.6 Perintah untuk *Join Cluster Manager*

Selanjutnya peneliti melakukan check apakah kedua worker sudah terjoinkan. Perintahnya seperti pada gambar 4.7

```
root@manager:/home/ghofarnugroho# docker node ls
ID                            HOSTNAME   STATUS
AVAILABILITY   MANAGER STATUS   ENGINE VERSION
mlc6zpodvemfxggpi87f1xvft  *     manager       Ready
Active         Leader           20.10.7
laua81aonwp1w8nt8rchff21j        worker1       Ready
Active                          20.10.7
8kuxgelg7c1tvvya3ajnuquf0        worker2       Ready
Active                          20.10.7
```

Gambar 4.7 Perintah Verifikasi Node Worker

Selanjutnya peneliti melakukan instalasi docker-compose pada node manager yang berfungsi untuk melakukan deployment image docker. Perintah untuk melakukan instalasi docker-compose terdapat pada gambar 4.8

```
root@manager:~#                         curl          -L
"https://github.com/docker/compose/releases/download
/1.24.0/docker-compose-$(uname  -s)-$(uname  -m)"  -o
/usr/local/bin/docker-compose
  %  Total       %  Received  %  Xferd   Average  Speed
Time    Time     Time  Current
                                  Dload        Upload
Total   Spent    Left  Speed
  0     0    0     0    0     0      0      0 --:--
:-- --:--:-- --:--:--     0
100  15.4M   100  15.4M     0      0   4821k        0
0:00:03  0:00:03 --:--:-- 5638k
root@manager:~# ls /usr/local/bin
docker-compose
root@manager:~#   chmod   +x   /usr/local/bin/docker-
compose
```

Gambar 4.8 Perintah untuk Instalasi docker-compose

Selanjutnya peneliti melakukan check apakah docker-compose sudah bias digunakan atau belum. Dengan perintah docker-compose –version

```
root@manager:~# docker-compose -version
docker-compose version 1.24.0, build 0aa59064
```

Gambar 4.9 Perintah *Verifikasi docker-compose*

4.1.2 Deploy Local Registry

Docker registry merupakan repository yang digunakan untuk meyimpan image docker yang akan dijalankan pada docker swarm. Pada saat kita akan menjalankan service pada docker swarm image akan dibuild pada node manager kemudian image tersebut akan didistribusikan ke semua node worker melalui registry.

Gambar 4.10 merupakan perintah yang digunakan untuk melakukan deploy registry pada docker swarm. --publish merupakan setting expose port agar registry dapat diakses dari luar docker seperti akses dari workstation melalui laptop. --name digunakan untuk memberi nama service dalam hal ini adalah registry. registry:2 merupakan base image yang digunakan.

```
root@manager:~# docker service create --name
registry --publish published=5000,target=5000
registry:2
glvzjxysr68ohlnqmcs6fvcn9
overall progress: 1 out of 1 tasks
1/1: running
verify: Service converged
```

Gambar 4.10 Perintah untuk *Deploy Registry*

Selanjutnya peneliti melakukan verifikasi apakah service registry sudah berjalan dengan menggunakan perintah docker service ls seperti pada gambar 4.11

```
root@manager:~# docker service ls
ID              NAME        MODE          REPLICAS
IMAGE        PORTS
glvzjxysr68o      registry      replicated      1/1
registry:2   *:5000->5000/tcp
```

Gambar 4.11 Perintah untuk *Verifikasi Service Registry*

4.1.3 Virtual IP

Virtual IP merupakan IP yang nantinya akan digunakan client untuk melakukan akses pada aplikasi – aplikasi yang dihosting pada docker swarm. Virtual IP ini bekerja dengan cara melakukan monitoring pada semua node, semisal virtual ip ini diinstall pada node manager dan sewaktu waktu node manager down maka virtual ip ini akan otomatis berpindah ke node worker. Sehingga aplikasi yang dihosting tetap bias diakses oleh client.

Langkah pertama peneliti melakukan install software cluster manager pada semua node.

```
root@manager:/home/ghofarnugroho#    apt    install
pacemaker corosync crmsh haveged
Reading package lists... Done
Building dependency tree
Reading state information... Done
The   following   additional   packages   will   be
installed:
  cluster-glue  ibverbs-providers  libcfg6  libcib4
libcmap4 libcorosync-common4 libcpg4
  libcrmcluster4    libcrmcommon3    libcrmservice3
libesmtp6 libhavege1 libibverbs1 liblrm2 liblrmd1
  libltdl7   libnet1   libnl-route-3-200   libnspr4
libnss3 libopenhpi3 libopenipmi0 libpe-rules2
  libpe-status10  libpengine10  libpils2  libplumb2
libplumbgpl2 libpython-stdlib
  libpython2.7-minimal  libpython2.7-stdlib  libqb-
dev libqb0 libquorum5 librdmacm1 libsensors4
  libsnmp-base libsnmp30 libstatgrab10 libstonith1
libstonithd2 libtimedate-perl libtotem-pg5
  libtransitioner2   libvotequorum8   libxml2-utils
openhpid pacemaker-cli-utils pacemaker-common
  pacemaker-resource-agents    python    python-bs4
python-chardet python-dateutil python-html5lib
```

Gambar 4.12 Perintah untuk *Install Software Cluster Manager*

Selanjutnya,peneliti melakukan stop cluster manager service agar bisa dikonfigurasi pada semua node.

```
root@manager:/home/ghofarnugroho# systemctl stop corosync pacemaker
```

Gambar 4.13 Perintah untuk *Stop Cluster Manager Service*

Selanjutnya peneliti membuat authorization key untuk cluster pada node manager

```
root@manager:/home/ghofarnugroho# corosync-keygen
Corosync Cluster Engine Authentication key generator.
Gathering 1024 bits for key from /dev/random.
Press keys on your keyboard to generate entropy.
Press keys on your keyboard to generate entropy (bits = 920).
Press keys on your keyboard to generate entropy (bits = 1000).
Writing corosync key to /etc/corosync/authkey.
```

Gambar 4.14 Perintah untuk Membuat *Authorization Key*

Selanjutnya peneliti melakukan konfigurasi file /etc/corosync/corosync.conf pada node manager

```
totem {
  version: 2
  cluster_name: docker-swarm
  transport: udpu

  interface {
    ringnumber: 0
    bindnetaddr: 10.10.10.0
    broadcast: yes
    mcastport: 5407
  }
}

nodelist {
  node {
    ring0_addr: manager
  }
  node {
    ring0_addr: worker1
  }
  node {
    ring0_addr: worker2
  }
}

quorum {
  provider: corosync_votequorum
}

logging {
  to_logfile: yes
  logfile: /var/log/corosync/corosync.log
  to_syslog: yes
  timestamp: on
}
service {
  name: pacemaker
  ver: 0
}
```

Gambar 4.15 Konfigurasi File corosync.conf

Selanjutnya copy file authorization dan konfigurasi corosync ke semua node worker

```
root@manager:/home/ghofarnugroho#      tar      -czvf
corosync.tar.gz /etc/corosync/
tar: Removing leading `/' from member names
/etc/corosync/
/etc/corosync/uidgid.d/
/etc/corosync/authkey
/etc/corosync/corosync.conf
root@manager:/home/ghofarnugroho#                 scp
corosync.tar.gz ghofarnugroho@worker1:~/
ghofarnugroho@worker1's password:
corosync.tar.gz
100%  666   975.6KB/s   00:00
root@manager:/home/ghofarnugroho#                 scp
corosync.tar.gz ghofarnugroho@worker2:~/
The  authenticity  of  host  'worker2  (10.10.10.77)'
can't be established.
ECDSA          key          fingerprint          is
SHA256:1yl2VL45WVv9psh7Qqcn009b9BHvjTf9c/xn690SBA4.
Warning:  Permanently  added  'worker2,10.10.10.77'
(ECDSA) to the list of known hosts.
ghofarnugroho@worker2's password:
corosync.tar.gz
100%  666     1.3MB/s   00:00
```

Gambar 4.16 Perintah Copy File Authorization dan Konfigurasi corosync

Selanjutnya peneliti melakukan replace file konfigurasi corosync pada semua node worker.

```
root@worker2:/home/ghofarnugroho#       rm       -rf
/etc/corosync/
root@worker2:/home/ghofarnugroho#                  mv
corosync.tar.gz /etc/
root@worker2:/home/ghofarnugroho# cd /etc/
root@worker2:/etc# tar -xzvf co
console-setup/   corosync.tar.gz
root@worker2:/etc# tar -xzvf co
console-setup/   corosync.tar.gz
root@worker2:/etc# tar -xzvf corosync.tar.gz
etc/corosync/
etc/corosync/uidgid.d/
etc/corosync/authkey
etc/corosync/corosync.conf
```

Gambar 4.17 Perintah untuk Replace File Konfigurasi corosync

Selanjutnya ubah owner pada folder /etc/corosync agar menjadi root

```
root@worker2:/etc# chown -R root:root /etc/corosync
```

Gambar 4.18 Perintah untuk Mengubah *Owner* folder /etc/corosync

Selanjutnya menjalankan *service corosync* dan *pacemaker* pada semua node

```
root@manager:/etc# systemctl start corosync
root@manager:/etc# systemctl start pacemaker
```

Gambar 4.19 Perintah untuk Menjalankan *Service corosync* dan *pacemaker*

Selanjutnya peneliti melakukan setting agar *cluster manager* dapat dijalankan otomatis ketika booting pada semua node

```
root@manager:/etc# systemctl enable corosync
Synchronizing state of corosync.service with SysV
service script with /lib/systemd/systemd-sysv-
install.
Executing: /lib/systemd/systemd-sysv-install enable
corosync
root@manager:/etc# update-rc.d pacemaker defaults
20 01
root@manager:/etc# systemctl enable pacemaker
Synchronizing state of pacemaker.service with SysV
service script with /lib/systemd/systemd-sysv-
install.
Executing: /lib/systemd/systemd-sysv-install enable
pacemaker
```

Gambar 4.20 Perintah untuk *Enable corosync*

Selanjutnya cek status cluster, pastikan semua node online

```
root@manager:/etc# crm status
Stack: corosync
Current DC: worker1 (version 1.1.18-2b07d5c5a9) -
partition with quorum
Last updated: Fri May 13 16:48:27 2022
Last change: Fri May 13 16:43:04 2022 by hacluster
via crmd on worker1

3 nodes configured
0 resources configured

Online: [ manager worker1 worker2 ]

No resources
```

Gambar 4.21 Perintah Pengecekan Status Cluster

Selanjutnya peneliti melakukan disable STONITH dan Quorum Policy agar tidak terjadi konflik pada konfigurasi pada node manager

```
root@manager:/etc# crm configure property stonith-
enabled=false
root@manager:/etc# crm configure property no-
quorum-policy=ignore
```

Gambar 4.22 Perintah untuk *Disable STONITH* dan *Quorum Policy*

Selanjutnya peneliti melakukan konfigurasi virtual ip. IP: adalah virtual ip yang digunakan, nic: adalah interface network yang digunakan, interfal: adalah lama waktu pengecekan untuk monitoring status node apakah up atau down. Migration threshold: adalah batas toleransi node down untuk memenuhi syarat agar virtual ip dapat berpindah ke node yang lain.

```
root@manager:/etc# crm configure primitive
virtual_ip ocf:heartbeat:IPaddr2 params
ip="10.10.10.78" cidr_netmask="32" nic="ens160" op
monitor interval="3s" meta migration-threshold="5"
```

Gambar 4.23 Perintah untuk *Konfigurasi Virtual IP*

Selanjutnya peneliti melakukan cek apakah konfigurasi cluster manager sudah benar

```
root@manager:/etc# crm resource show
 virtual_ip                (ocf::heartbeat:IPaddr2):
Started
```

Gambar 4.24 Perintah Pengecekan *Status Node*

Selanjutnya cek virtual ip address

```
root@manager:/etc# ip addr | grep 10.10.10.78
    inet  10.10.10.78/32  brd  10.10.10.255  scope
global ens160
```

Gambar 4.25 Verifikasi *Virtual IP Address*

4.2 Pengujian

Setelah proses implementasi selesai, selanjutnya dilakukan pengujian untuk membuktikan sistem yang dibuat memenuhi tujuan penelitian. Pengujian dilakukan sebanyak 4 kali. Pengujian pertama dilakukan dengan mematikan salah satu node worker, pengujian kedua dilakukan dengan mematikan node manager dan worker1, pengujian ketiga dilakukan dengan mematikan node manager, pengujian keempat dilakukan dengan mematikan semua node. Pengujian ini dilakukan untuk mengetahui tingkat ketersediaan system yang dibangun apakah tinggi atau rendah.

4.2.1 Deploy Service Web Dummy

Untuk dapat melakukan pengujian dibutuhkan sebuah service web dummy. Gambar 4.26 merupakan file docker-compose.yml yang berisi nama image yang akan dibuild. Nama image tersebut adalah nginx.

```
version: '3'
services:
  nginx:
    image: 127.0.0.1:5000/nginx
    build: .
    ports:
      - "80:80"
```

Gambar 4.26 Konfigurasi File *docker-compose.yml*

Membuat Dockerfile yang akan digunakan untuk melakukan build image nginx

```
FROM nginx
```

Gambar 4.27 Konfigurasi *Dockerfile*

Selanjutnya build dan push image nginx ke local registry menggunakan perintah pada gambar 4.28

```
root@manager:/home/ghofarnugroho/nginx# docker-
compose build --no-cache
Building nginx
Step 1/1 : FROM nginx
latest: Pulling from library/nginx
214ca5fb9032: Pull complete
f0156b83954c: Pull complete
5c4340f87b72: Pull complete
9de84a6a72f5: Pull complete
63f91b232fe3: Pull complete
860d24db679a: Pull complete
Digest:
sha256:2c72b42c3679c1c819d46296c4e79e69b2616fa28bea92
e61d358980e18c9751
Status: Downloaded newer image for nginx:latest
 ---> 7425d3a7c478
Successfully built 7425d3a7c478
Successfully tagged 127.0.0.1:5000/nginx:latest
root@manager:/home/ghofarnugroho/nginx# docker-
compose push
Pushing nginx (127.0.0.1:5000/nginx:latest)...
The push refers to repository [127.0.0.1:5000/nginx]
feb57d363211: Pushed
98c84706d0f7: Pushed
4311f0ea1a86: Pushed
6d049f642241: Pushed
3158f7304641: Pushed
fd95118eade9: Pushed
latest: digest:
sha256:787480bfb4297dc887f8655dbc51074ef87f16ea359bae
ea3af0a4dd92948124 size: 1570
```

Gambar 4.28 Perintah untuk *Build* dan *Push Image Docker*

Setelah image berhasil dipush ke registry selanjutnya membuat docker service nginx dengan jumlah replika 3. Penulis menentukan jumlah replika sebanyak 3 dikarenakan dengan 3 replika sudah dapat untuk mengatasi request dari user sebanyak 50.000 request. Perintah pada gambar 4.29 akan membuat service nginx dengan nama nginx dan melakukan publish port 80 ke network eksternal agar dapat diakses oleh client.

```
root@manager:/home/ghofarnugroho/nginx#       docker
```

```
service create --name nginx --replicas 3 -p 80:80
127.0.0.1:5000/nginx
lmaaqhwqt31zl6lup9f4xn8fr
overall progress: 3 out of 3 tasks
1/3: running
2/3: running
3/3: running
verify: Service converged
```

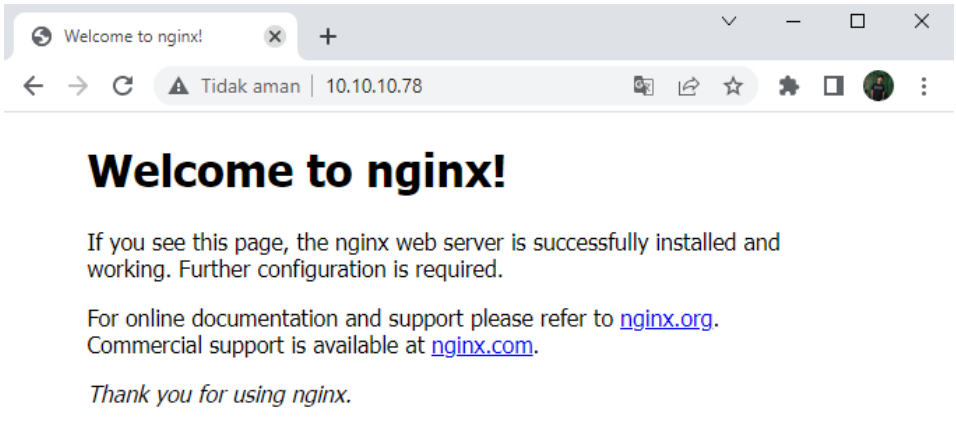
Gambar 4.29 Perintah untuk Membuat *Docker Service Nginx*

Untuk melakukan verifikasi cluster yang sudah dibuat dapat menggunakan perintah docker service ls seperti pada gambar 4.30

```
root@manager:/home/ghofarnugroho/nginx#        docker
service ls
ID                  NAME         MODE           REPLICAS
IMAGE                            PORTS
lmaaqhwqt31z      nginx            replicated      3/3
127.0.0.1:5000/nginx:latest    *:80->80/tcp
```

Gambar 4.30 Perintah untuk Verifikasi Cluster

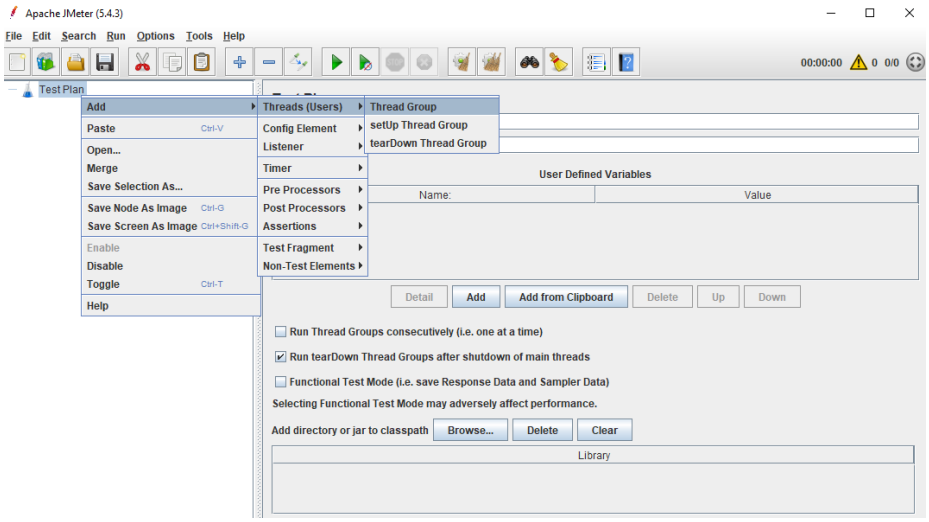
Selanjutnya peneliti melakukan pengujian service web menggunakan laptop workstation dengan mengakses virtual ip yang telah dibuat tadi, yaitu 10.10.10.78



Gambar 4.31 *Website Dummy* Dengan *Virtual IP*

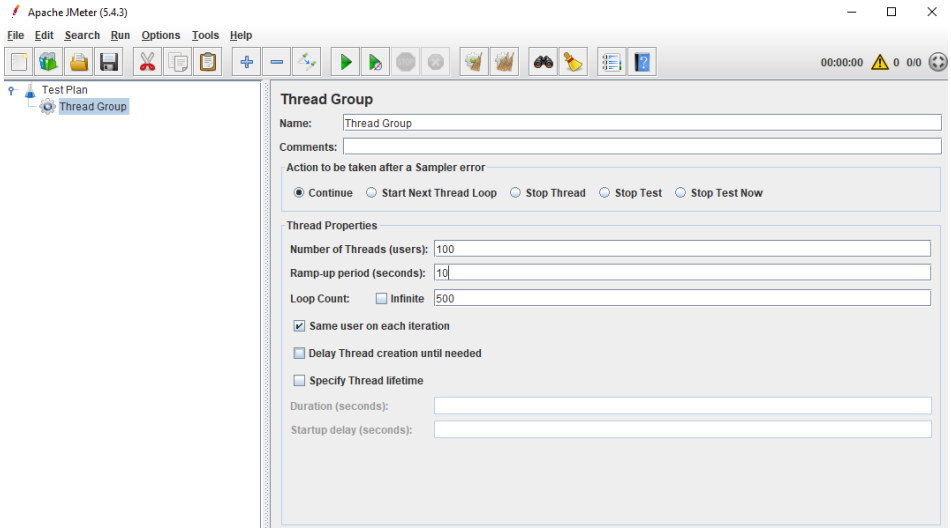
4.2.2 Pengujian High Availability

Untuk melakukan pengujian diperlukan sebuah tool yaitu JMeter. Membuat test plan pada JMeter dengan klik kanan Test Plan → Add → Threads (User) → Thread Grup



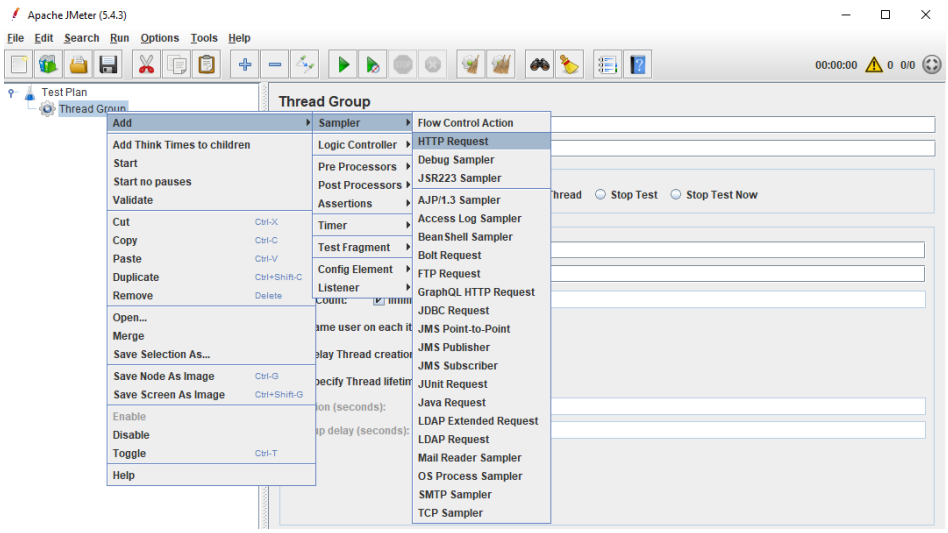
Gambar 4.32 Step Pembuatan *Thread Group*

Number thread of user adalah banyaknya jumlah user yang akan melakukan http request ke website dummy, sedangkan loop count adalah banyaknya request yang dilakukan oleh 1 user. Dari gambar 4.33 maka website dummy akan diload sebanyak 50.000 kali

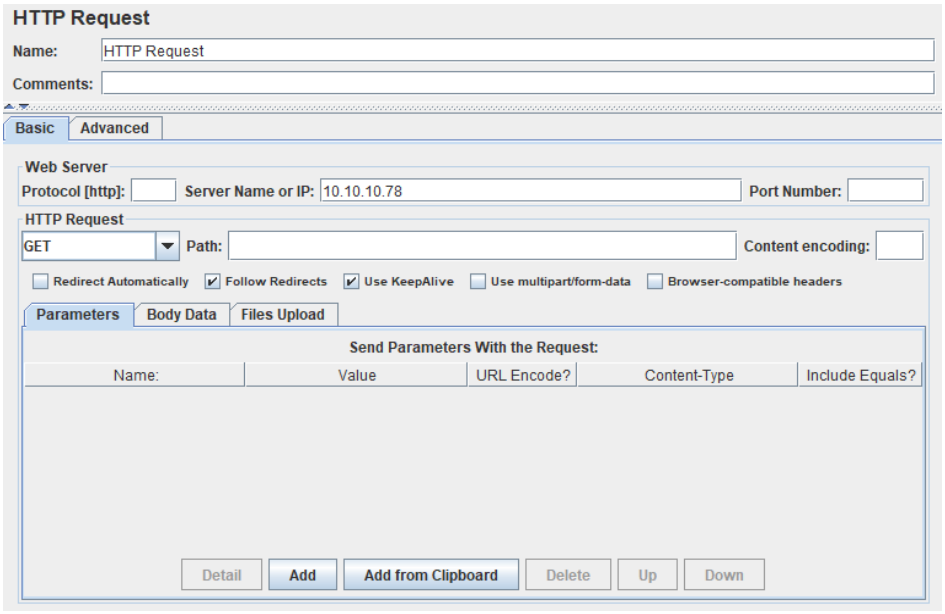


Gambar 4.33 *Step Input Loop Count* pada *Thread Group*

Selanjutnya tambahkan HTTP Request dengan klik kanan pada Thread Grup → Add → Sampler → HTTP Request. Isikan Virtual IP (10.10.10.78) pada kolom Server Name or IP

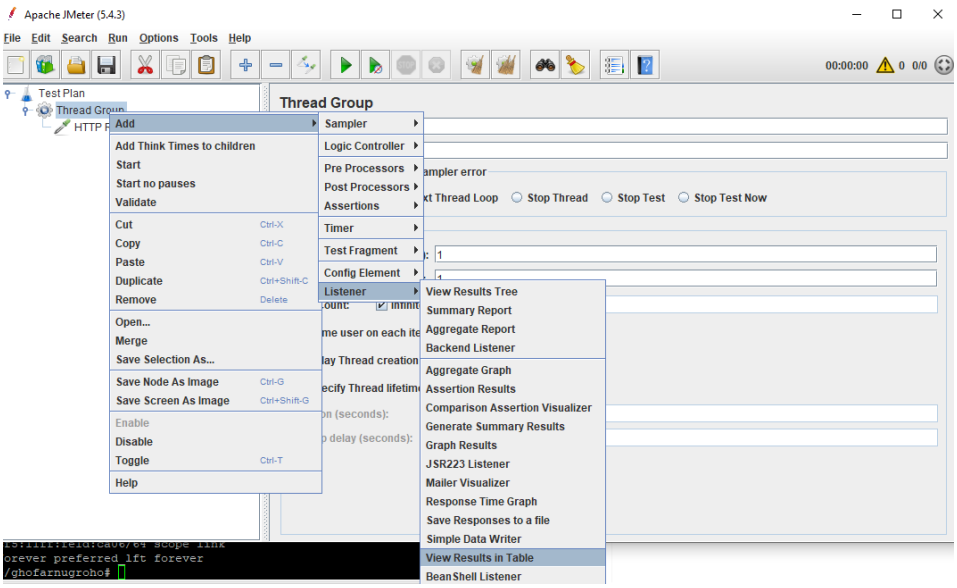


Gambar 4.34 Step Pembuatan HTTP Request



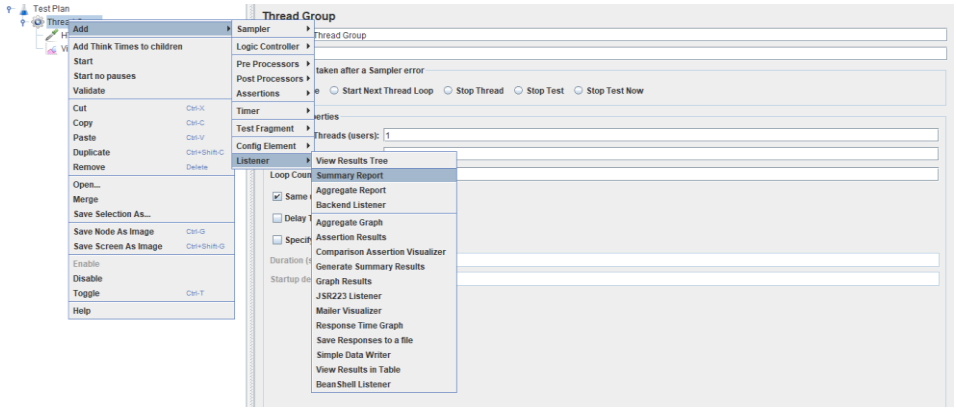
Gambar 4.35 *Step Input Virtual IP* pada *HTTP Request*

Selanjutnya tambahkan Listener agar dapat melihat hasil testing yang telah dilakukan. Klik kanan pada Thread Grup → Add → Listener → View Result in Table



Gambar 4.36 *Step* Pembuatan *View Result in Table*

Tambahkan Summary Report agar jumlah error dapat terdeteksi. Klik kanan Thread Group → Add → Listener → Summary Report



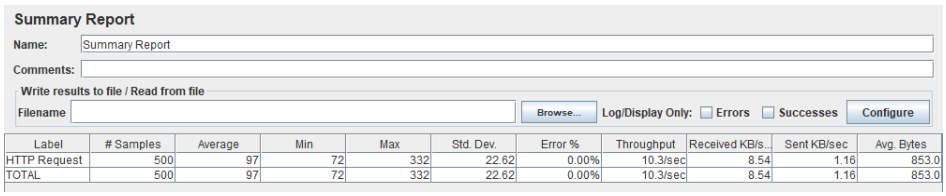
Gambar 4.37 *Step* Pembuatan *Summary Report*

Masing – masing pengujian akan dilakukan sebanyak 3 kali. Pengujian pertama yang dilakukan adalah mematikan satu node worker.

```
root@manager:/home/ghofarnugroho# crm status
Stack: corosync
Current DC: manager (version 1.1.18-2b07d5c5a9) - partition with quorum
Last updated: Fri May 13 20:45:21 2022
Last change: Fri May 13 16:54:31 2022 by root via cibadmin on manager

3 nodes configured
1 resource configured

Online: [ manager worker2 ]
OFFLINE: [ worker1 ]

Full list of resources:

 virtual_ip     (ocf::heartbeat:IPaddr2):       Started manager
```

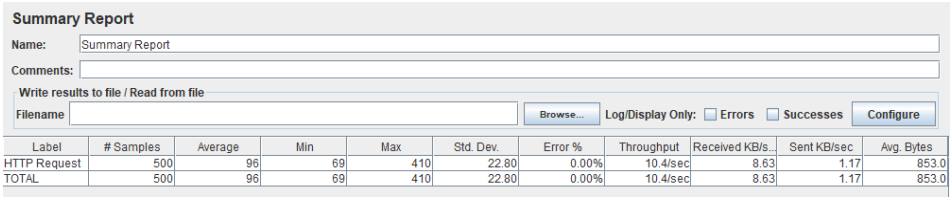
Gambar 4.38 Perintah Pengecekan *Status Node*

Hasil yang diperoleh sebagai berikut :

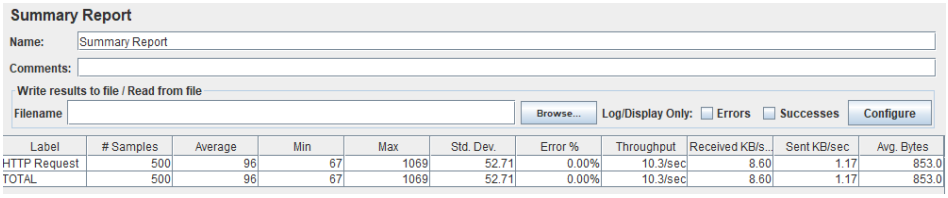


Label	# Samples	Average	Min	Max	Std. Dev.	Error %	Throughput	Received KB/s..	Sent KB/sec	Avg. Bytes
HTTP Request	500	97	72	332	22.62	0.00%	10.3/sec	8.54	1.16	853.0
TOTAL	500	97	72	332	22.62	0.00%	10.3/sec	8.54	1.16	853.0

Gambar 4.39 Hasil *Testing* Ke-1 *Node Worker1 Down*



Gambar 4.40 Hasil *Testing* Ke-2 *Node Worker1 Down*



Gambar 4.41 Hasil *Testing* Ke-3 *Node Worker1 Down*

Tabel 4.1 Hasil Pengujian High Availability Ketika Node Worker1 Down

Pengujian Ke :	Waktu Failover
Pertama	0
Kedua	0
Ketiga	0
Rata - rata	0

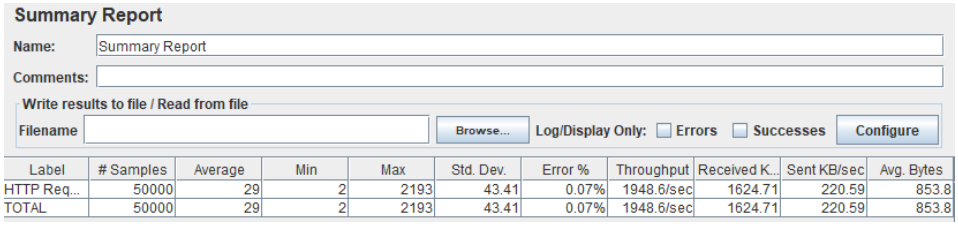
Berdasarkan pengujian diatas presentasi error yang didapatkan adalah 0% karena server tidak membutuhkan waktu untuk *failover* untuk sebuah *error*.

Pengujian kedua dilakukan dengan mematikan semua node worker.

```
root@worker2:~# crm status
Stack: corosync
Current DC: worker2 (version 1.1.18-2b07d5c5a9) - partition WITHOUT quorum
Last updated: Mon Jul 11 08:08:35 2022
Last change: Thu Jun 16 07:46:56 2022 by root via cibadmin on manager

3 nodes configured
1 resource configured

Online: [ worker2 ]
OFFLINE: [ manager worker1 ]

Full list of resources:

 virtual_ip     (ocf::heartbeat:IPaddr2):       Started worker2
```

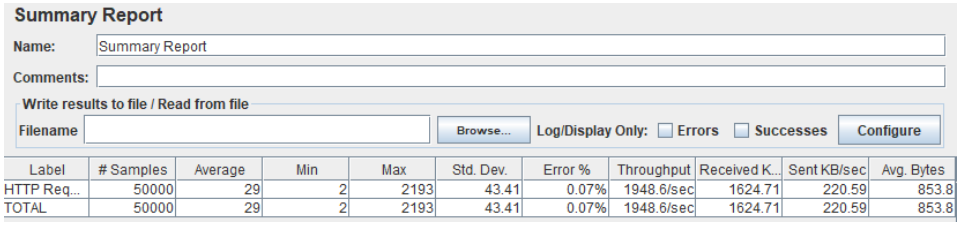
Gambar 4.42 Perintah Pengecekan *Status Node*

Hasil yang diperoleh sebagai berikut



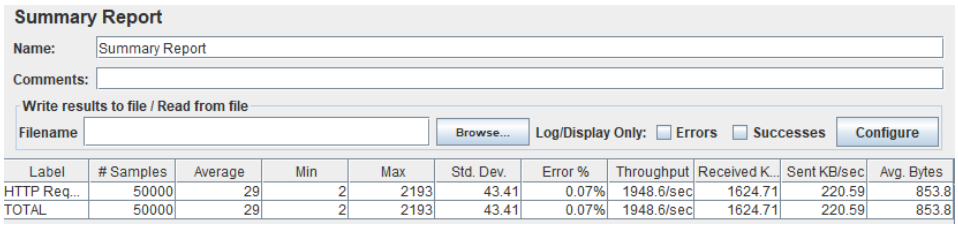
Label	# Samples	Average	Min	Max	Std. Dev.	Error %	Throughput	Received K...	Sent KB/sec	Avg. Bytes
HTTP Req...	50000	29	2	2193	43.41	0.07%	1948.6/sec	1624.71	220.59	853.8
TOTAL	50000	29	2	2193	43.41	0.07%	1948.6/sec	1624.71	220.59	853.8

Gambar 4.43 Hasil Testing Ke-1 Manager dan Worker1 Down



Label	# Samples	Average	Min	Max	Std. Dev.	Error %	Throughput	Received K...	Sent KB/sec	Avg. Bytes
HTTP Req...	50000	29	2	2193	43.41	0.07%	1948.6/sec	1624.71	220.59	853.8
TOTAL	50000	29	2	2193	43.41	0.07%	1948.6/sec	1624.71	220.59	853.8

Gambar 4.44 Hasil *Testing* Ke-2 Manager dan Worker1 *Down*



Label	# Samples	Average	Min	Max	Std. Dev.	Error %	Throughput	Received K...	Sent KB/sec	Avg. Bytes
HTTP Req...	50000	29	2	2193	43.41	0.07%	1948.6/sec	1624.71	220.59	853.8
TOTAL	50000	29	2	2193	43.41	0.07%	1948.6/sec	1624.71	220.59	853.8

Gambar 4.45 Hasil *Testing* Ke-3 Manager dan Worker1 *Down*

Tabel 4.2 Hasil Pengujian *High Availability* Node Manager & Worker1 Down

Pengujian Ke :	Waktu Failover
Pertama	0.07
Kedua	0.07
Ketiga	0.07
Rata - rata	0.07

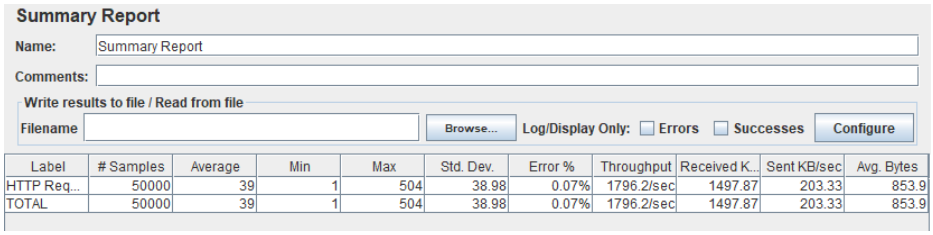
Berdasarkan pengujian diatas presentasi error yang didapatkan adalah 0.07% karena server tidak membutuhkan waktu untuk failover untuk sebuah error.

Pengujian ketiga dilakukan dengan mematikan node manager.

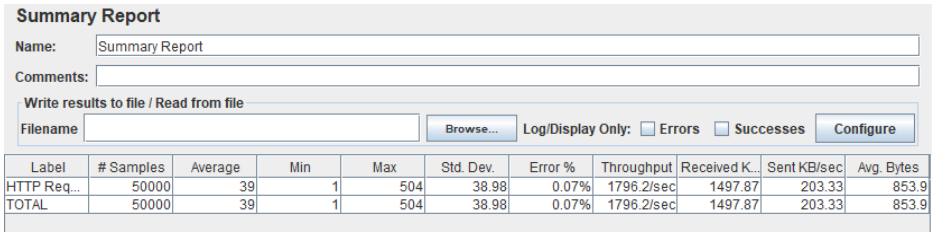
```
root@worker2:/home/ghofarnugroho# crm status
Stack: corosync
Current DC: worker2 (version 1.1.18-2b07d5c5a9) - partition with quorum
Last updated: Fri May 13 21:21:47 2022
Last change: Fri May 13 16:54:31 2022 by root via cibadmin on manager

3 nodes configured
1 resource configured

Online: [ worker1 worker2 ]
OFFLINE: [ manager ]

Full list of resources:

 virtual_ip     (ocf::heartbeat:IPaddr2):       Started worker1
```

Gambar 4.46 Perintah Pengecekan *Status Node*

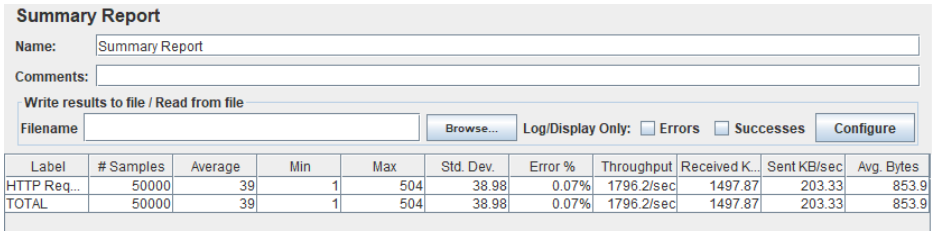
Hasil yang diperoleh sebagai berikut :



Gambar 4.47 Hasil Ke-1 Testing *Node Manager Down*



Gambar 4.48 Hasil Testing Ke-2 *Node Manager Down*



Gambar 4.49 Hasil Testing Ke-3 *Node Manager Down*

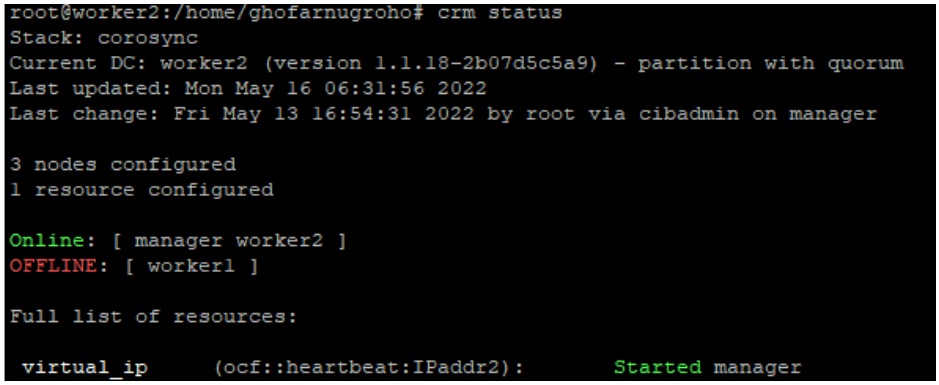
Tabel 4.3 Hasil Pengujian High Availability Ketika Node Manager Down

Pengujian Ke :	Waktu Failover
Pertama	0.07
Kedua	0.07
Ketiga	0.07
Rata - rata	0.07

Berdasarkan pengujian diatas waktu yang diperlukan sistem untuk melakukan failover error selama 0.07 detik. Sistem juga dapat melakukan failover virtual ip ke node yang lain ketika node manager mengalami downtime. Sistem yang dibangun tetap dapat beroperasi ketika 2/3 mengalami crash. Koneksi client akan secara otomatis dialihkan ke node yang up, sehingga sistem tidak mengalami downtime.

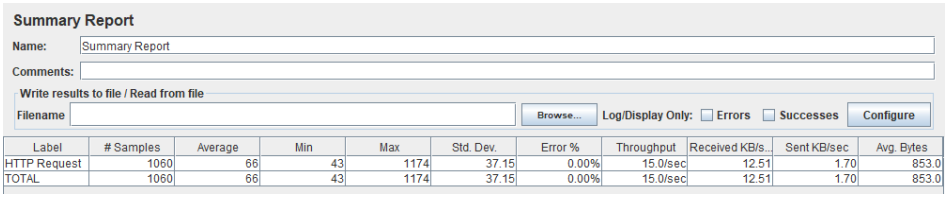
Pengujian keempat dilakukan dengan mematikan semua node, baik itu node manager maupun node worker

Pertama – tama peneliti mematikan node worker1 terlebih dahulu.



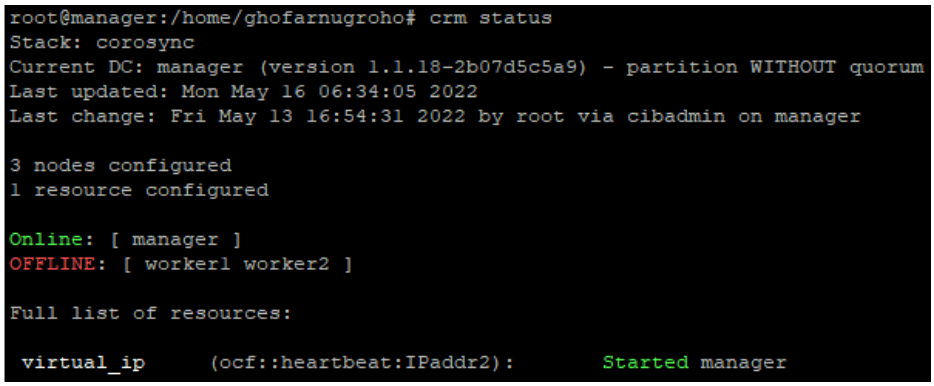
Gambar 4.50 Perintah Pengecekan *Status node*

Hasil setelah node worker1 dimatikan



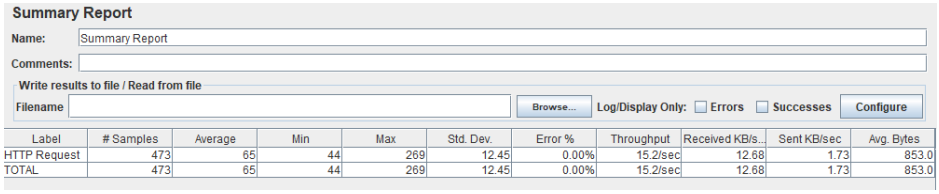
Gambar 4..51 Hasil Testing Setelah Node Worker1 Dimatikan

Selanjutnya peneliti mematikan node worker2.



Gambar 4.52 Perintah Pengecekan *Status node*

Hasil setelah node worker1 dan node worker2 dimatikan

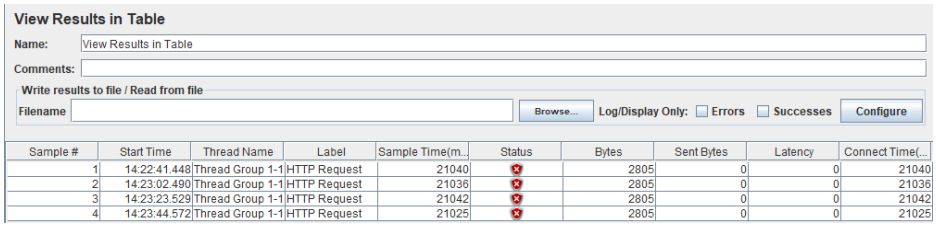


Gambar 4.53 Hasil *Testing Node Worker1* dan *Worker2* Dimatikan

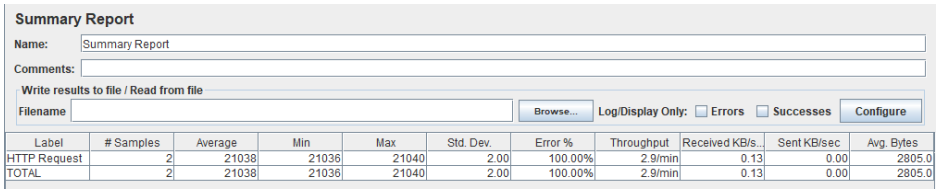
Yang terakhir peneliti mematikan node manager

Gambar 4.54 Semua Node Dimatikan

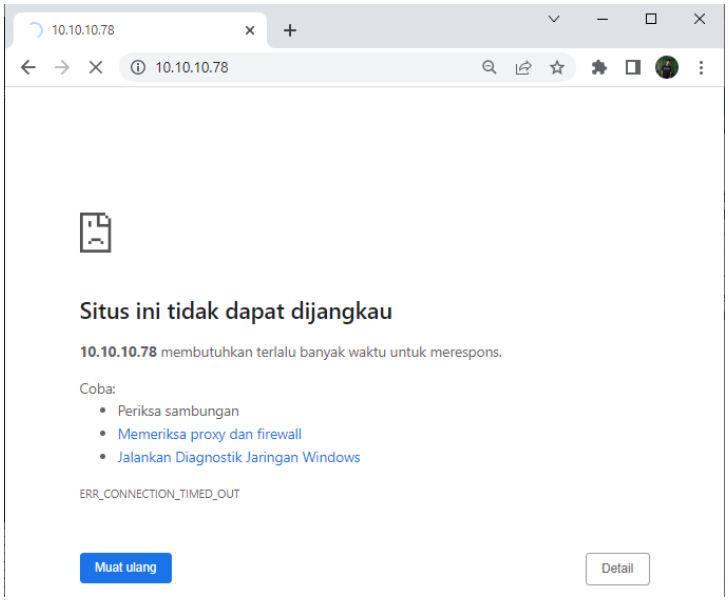
Hasil setelah semua node dimatikan



Gambar 4.55 Hasil *View Result in Table* Setelah Semua *Node* Dimatikan



Gambar 4.56 Hasil *Summary Report* Setelah Semua *Node* Dimatikan



Gambar 4.57 *Website Down* Setelah Semua *Node* Dimatikan

Berdasarkan pengujian diatas cluster tidak dapat melakukan failover dikarenakan semua node mati. untuk mengatasi hal itu maka sebaiknya node diletakkan pada environment yang berbeda untuk memperoleh *redundancy* sehingga kemungkinan semua node down rendah.

Summary Hasil testing High Avaibility :

Tabel 4.4 *Summary* Hasil Pengujian

Pengujian Ke :	Waktu Failover saat Worker1 Down	Waktu Failover saat Manager Down	Waktu Failover saat Manager dan Worker1 Down
Pertama	0	0.07	0.07
Kedua	0	0.07	0.07
Ketiga	0	0.07	0.07
Rata - rata	0	0.07	0.07

Berdasarkan pengujian yang dilakukan, dapat dikatakan sistem yang dibangun memiliki tingkat ketersediaan yang tinggi. Sistem tetap dapat melayani request user walaupun 2 node down. Error yang terjadi dikarenakan adanya perpindahan replika image container dari node yang down ke node yang up. Error yang terjadi masih bisa ditolerir karena sangat kecil sekali yaitu 0.07 detik, sedangkan website dummy tetap bisa diakses oleh user ketika 2 node mengalami down

4.2.3 Pengujian Replika Service

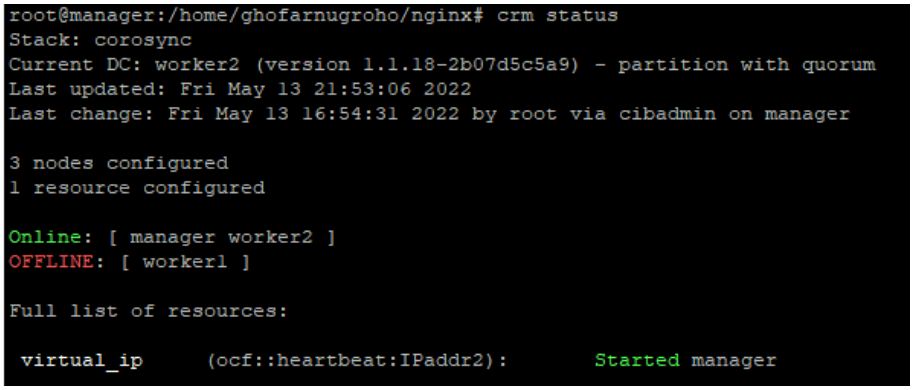
Pengujian ini dilakukan untuk mengetahui apakah docker swarm dapat menjaga agar replica tetap sesuai dengan yang telah kita tentukan. Pengujian ini dilakukan dengan membuat service dengan jumlah replika 5, kemudian dua node worker akan dimatikan.

Pada awalnya service akan dibagi secara merata kepada seluruh node yang tergabung pada docker swarm cluster seperti pada gambar 4.58

```
root@manager:/home/ghofarnugroho/nginx# docker service ps nginx
ID             NAME      IMAGE                          NODE      DESIRED STATE
 CURRENT STATE           ERROR     PORTS
fxolw93kmpja   nginx.1   127.0.0.1:5000/nginx:latest   worker2   Running
 Running 20 seconds ago
04ix97hb4kdl   nginx.2   127.0.0.1:5000/nginx:latest   manager   Running
 Running 20 seconds ago
ylr6f668rrkb   nginx.3   127.0.0.1:5000/nginx:latest   worker1   Running
 Running 20 seconds ago
buu3skbu7ecz   nginx.4   127.0.0.1:5000/nginx:latest   worker2   Running
 Running 20 seconds ago
w2bcu2r2eac6   nginx.5   127.0.0.1:5000/nginx:latest   worker1   Running
 Running 20 seconds ago
```

Gambar 4.58 Perintah Pengecekan *Cluster*

Kemudian jika salah satu node worker crash seperti pada gambar 4.59



Gambar 4.59 Perintah Pengecekan *Status Node*

Maka node manager akan menjaga agar jumlah replika yang running tetap sesuai dengan yang kita tentukan dengan membuat container baru pada node yang up seperti pada gambar 4.60

Gambar 4.60 Perintah Pengecekan Jumlah Replika

Dengan metode *auto replicated* maka jumlah client yang dapat ditangani oleh sistem akan tetap sama meskipun ada salah satu node yang *crash* karena *docker swarm manager* menjaga agar jumlah replika tetap sama.