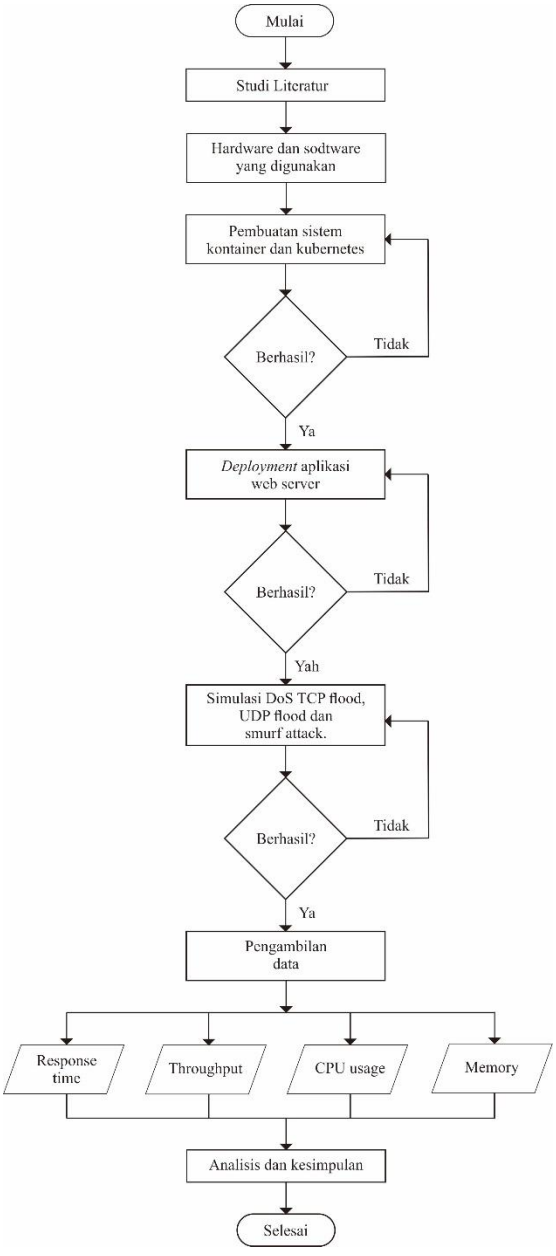
BAB 3
METODE PENELITIAN

3.1 Alur Penelitian

Penelitian analisis performansi kontainer *berplatform* Kubernetes terhadap serangan *Denial of Service* yang akan dilakukan terdiri dari beberapa tahapan, di mulai dari studi literatur sampai tahap penyusunan laporan. Gambar 3.1 adalah diagram alur yang digunakan pada penelitian ini.



Gambar 3.1 Alur Penelitian

Pada gambar 3.1 Alur Penelitian dimulai dari studi literatur, dimana pada studi literatur yaitu mencari referensi mengenai penelitian sebelumnya, yang berkaitan dengan materi analisis performansi *container berplatform* Kubernetes terhadap serangan *Denial of Service* (DoS). Langkah kedua yaitu persiapan *hardware* dan *software* yang akan digunakan untuk penelitian. Dilanjutkan dengan pembuatan *system container* dan Kubernetes pada *Google Cloud Platform*, mulai dari *create hardware virtual* seperti CPU dan *memory* lalu setelah terbentuk akan dilakukan *setting* IP atau *deployment container*. *Deployment* dikatakan berhasil jika sudah tidak ada *error* dan container bisa digunakan, setelah *deployment container* berhasil akan dilanjutkan dengan *create* aplikasi *Nginx web server* pada masing-masing *container* yang terdapat pada *worker* node. *Deploy* aplikasi *web server* dikatakan berhasil jika *web server* sudah dapat diakses oleh *user* dan *website* tertampil dengan semestinya. Setelah *website* sudah dapat diakses, selanjutnya akan dilakukan simulasi penyerangan menggunakan *scenario* penyerangan DoS TCP *flood*, UDP *flood* dan *Smurft Attack* dengan aplikasi penyerang Hping3. Setelah penyerangn berhasil ditandai dengen *script* penyerangan *running* dan CPU *usage* pada *master node* naik, akan dilakukan pengambilan data berupa *response time*, *throughput*, CPU *usage* dan *memory usage*. Dimana untuk pengambilan *response time* dan *throughput* menggunakan aplikasi *benchmark Siege* dari sisi penyerang, serta untuk pengambilan data CPU *usage* dan *memory usage* menggunakan aplikasi Htop yang diambil melalui *master node*. Setelah selesai dilakukan simulasi dan pengambilan data, selanjutnya yaitu akan dianalisis dan diambil kesimpulan dari seluruh hasil data.

3.2 Studi Literatur

Studi literatur menjadi tahap pertama dalam proses penelitian yang berfungsi untuk mencari referensi yang dapat dijadikan sebagai acuan dalam pelaksanaan penelitian. Sumber yang digunakan oleh penulis berupa jurnal, buku, dan *website* yang berhubungan dengan topik penelitian.

3.3 *Hardware* dan *Software* yang Digunakan

Perangkat-perangkat yang digunakan untuk membuat simulasi dalam penelitian ini terdiri atas perangkat keras (*hardware*) dan perangkat lunak (*software*) untuk *Kubernetes cluster*, aplikasi *deployment webserver* dan *tools benchmark*. Spesifikasi *hardware* dan *software* yang akan digunakan ditunjukkan pada Tabel 3.1.

Tabel 3.1 Spesifikasi *Hardware* dan *Software* yang Digunakan

Perangkat	Spesifikasi	Keterangan
Kubernetes Cluster	OS	*Ubuntu Server* 20.04
	CPU	2 vCPU
	RAM	8 GB
	Tool dan Aplikasi	1. *Kubelet version* 1.22.1 2. *Kubectl version* 1.22.1 3. *Kubeadm version* 1.22.1
Web Server	*Apps*	*Nginx version* 1.18.0
Benchmark tools	*Apps*	*Siege* v4.1.6 *Htop* v2.2.0
DoS Tools	*Apps*	*Hping3*

3.4 Perancangan Skenario

Skenario yang digunakan pada penelitian ini menggunakan jenis skenario serangan *Denial of Service* yang berbeda yaitu TCP *flood*, UDP *flood*, dan *smurf attack.* Jenis komunikasi yang digunakan yaitu komunikasi bertipe *loadbalancer* pada aplikasi *web server* di Kubernetes. Skenario penelitian dimaksudkan untuk mengetahui pengaruh serangan *Denial of Service* pada kinerja kontainer *berplatform* Kubernetes.

Tabel 3.2 Skenario Pengujian

Skenario Penelitian	Jenis DoS	Parameter Pengujian
Mengalami serangan DoS	TCP *Flood*	1. *response time,* 2. *throughput,* 3. CPU *usage,* 4. *memory*
	UDP *Flood*	
	Smurf Attack	

Skenario Penelitian	Jenis DoS	Parameter Pengujian
Tidak Mengalami Serangan DoS	(keadaan normal)	1. *response time*, 2. *throughput*, 3. CPU *usage*, 4. *memory*

3.5 Perancangan Parameter

Parameter *benchmark* yang akan dianalisis pada penelitian ini yaitu *response time*, *throughput*, CPU *usage*, dan *memory*. Hasil dari parameter tersebut didapatkan menggunakan *tools Siege* dan *Htop*. Pengujian ini dilakukan dengan menguji beban *web server* berbasis *container* Kubernetes yang dapat ditampung dan tidak menyebakan *server* menjadi *down*/tidak tersedia. Tabel 3.2 adalah tabel parameter yang akan diamati pada penelitian.

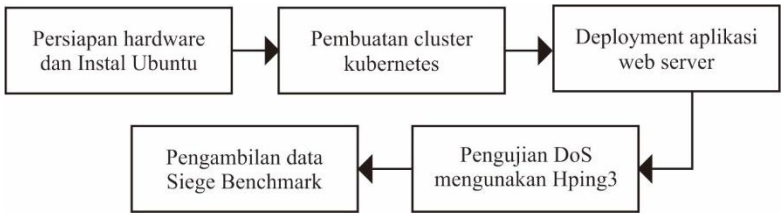
Tabel 3.3 Parameter Pengujian

Parameter Pengujian	Satuan
Response Time	*second*
Throughput	Mbps
CPU *Usage*	%
Memory	MB

3.6 Proses Simulasi

Pada tahap ini, akan dilakukan pembuatan simulasi yang dijalankan dengan skenario yang dirancang seperti pada Gambar 3.2. Langkah awal yaitu mempersiapkan *hardware* dan *software* yang digunakan untuk proses simulasi. Pada proses simulasi pertama menginstal sistem operasi *Ubuntu Server* 20.04 yang digunakan sebagai *server*. Selanjutnya membuat *cluster* Kubernetes dengan menggunakan satu *master node* dan dua *worker node*. Membuat *deployment* aplikasi *webserver* menggunakan *Nginx*. Ketika pembuatan *system* selesai dilanjutkan dengan persiapan pembuatan *tools* pengujian menggunakan aplikasi *Hping3* yang dijalankan pada sisi *client*, di mana ini dapat menjadi gambaran ketika

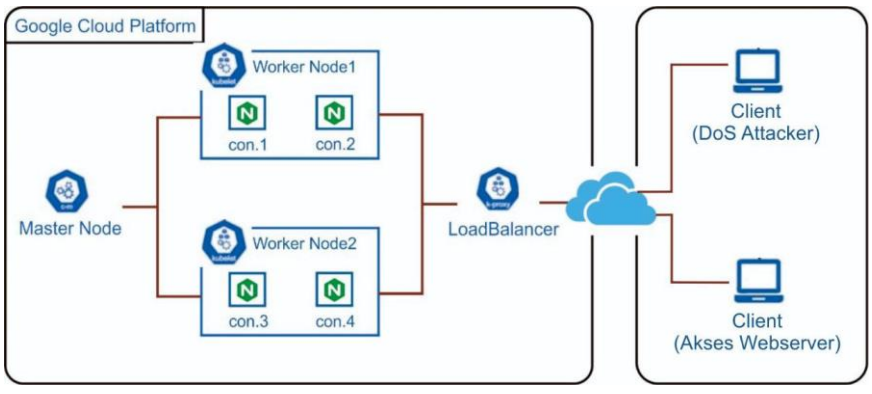
server mengalami penyerangan DoS dari luar. Pada proses pengambilan data yaitu menggunakan *tools* Siege dan *Htop* untuk mengetahui kinerja *server* Ketika mengalami serangan DoS ataupun saat tidak mengalami serangan DoS.



Gambar 3.2 Diagram Blok Sistem

Pada gambar 3.3 merupakan gambaran *cluster* Kubernetes yang akan menggunakan satu *master node* sebagai *control plane* dan dua *worker node*. Komunikasi antara *cluster* Kubernetes dengan *client* menggunakan jenis komunikasi *loadbalancer* yang di pasangkan dengan *web server Nginx*.

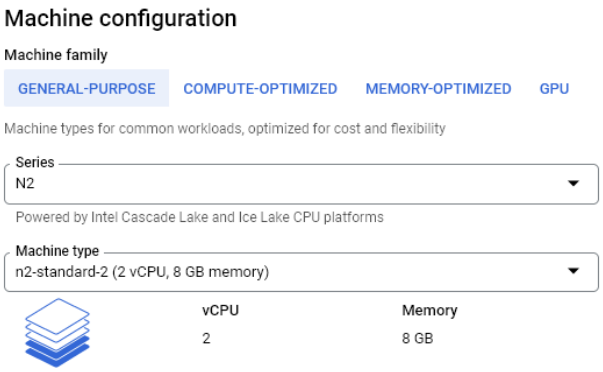
Dimana pada gambaran 3.3 *cluster* Kubernetes secara total terdapat empat worker node. Dan pada masing-masing worker node akan diinstal aplikasi *web server Nginx*, yang masing-masing aplikasi *web server Nginx* tersebut terhubung dengan *load balancer*. Dengan proses penyerangan yaitu user attacker akan melakukan pengerangan menggunakan suatu perangkat, selanjutnya yang akan menerima serangan tersebut adalah load balancer. Lalu load balancer akan membagi serangan atau request masuk tersebut kepada masing-masing container secara merata. Sementara user penguna akan mengakses pada *web server Nginx* untuk melakukan pengujian *web server Nginx* tersebut masih dapat di access dengan baik.



Gambar 3.3 Topologi Jaringan

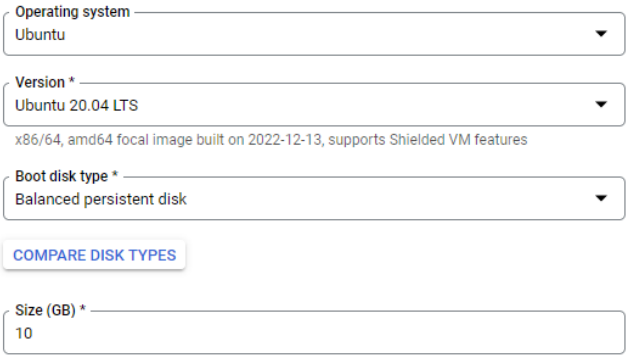
3.6.1 *Google Cloud Platform*

Google Cloud Platform (GCP) adalah kumpulan layanan komputasi awan yang ditawarkan oleh *Google*. Fitur yangdigunakan pada GCP ini *compute engine*, fitur ini digunakan untuk membuat sebuah *host server* pada *platform* GCP. *Hardware* yang sudah ditentukan pada pembahasan sebelumnya akan di implementasikan pada GCP.



Gambar 3.4 Konfigurasi CPU dan *Memory*

Gambar 3.4 merupakan konfigurasi CPU yang digunakan yaitu 2vCPD dan *memory* yang digunakan sebesar 8GB. Gambar 3.5 merupakan konfigurasi untuk jenis sistem operasi yang digunakan yaitu Linux Ubuntu 20.04.



Gambar 3.5 Konfigurasi CPU dan *Memory*

Setelah proses konfigurasi *server* yang digunakan selasai, dimana pada penelitian ini akan membuat 3 *host server* yang berfungsi sebagai 1 *master node* dan 2 *worker node* menggunakan konfigurasi *hardware* yang sama untuk ketiganya.

3.6.2 *Instalasi Kubernetes*

Instalasi Kubernetes dilakukan dengan beberapa tahapan dimulai dari instalasi *containerd, kubectl, kubelet, kubeadm, container network interface,*

inisialisasi *cluster* dan menggabungkan *worker node* ke dalam *cluster* Kubernetes. Serta konfigurasi Kubernetes dibagi menjadi 2 segmen yaitu konfigurasi pada *master node* dan juga *worker node*.

3.6.2.1 Konfigurasi *Master Node, Worker Node* 1, *Worker Node* 2

Konfigurasi yang pertama yaitu membuat *maaping* nama-nama *host* yang akan digunakan. Pada penelitian ini menggunakan 3 *host server* dengan dengan masing masing nama yaitu kube-masternode, kube-workernode01 dan kube-workernode02 seperti pada gambar 3.6.

```
kiki@kube-masternode:~$ cat /etc/hosts
10.128.0.24 kube-masternode
10.128.0.25 kube-workernode01
10.128.0.26 kube-workernode02
127.0.0.1 localhost
```

Gambar 3.6 Mengatur *mapping hostname*

Menginstal *packaged containerd* yang digunakan untuk menjalankan Kubernetes sebagai pembuat kontainer dan *image container* pada *node* Kubernetes. Muat modul kernel overlay dan br_netfilter yang digunakan untuk mengaktifkan modul pada kernel Linux Ubuntu yang berfungsi untuk komunikasi pada *cluster* Kubernetes.

```
cat <<EOF | sudo tee /etc/modules-load.d/containerd.conf
overlay
br_netfilter
EOF
```

```
sudo modprobe overlay
sudo modprobe br_netfilter
```

Tetapkan konfigurasi sistem untuk jaringan Kubernetes untuk parameter net bridge ip table, ipv4 forward dan bridge ip6table.

```
cat <<EOF | sudo tee /etc/sysctl.d/99-kubernetes-cri.conf
net.bridge.bridge-nf-call-iptables = 1
net.ipv4.ip_forward = 1
net.bridge.bridge-nf-call-ip6tables = 1
EOF
```

Melakukan *apply* pada sistem dengan cara membaca dan memodifikasi atribut kernel sistem.

```
sudo sysctl --system
```

Melakukan *update* dan menginstal *containerd*, dan dilanjutkan dengan menambahkan direktori khusus untuk *containerd*. Dilanjutkan dengan beberapa konfigurasi untuk mengatur entri sistem sehingga dapat memulainya secara otomatis pada saat *boot*. *Containerd* memiliki perintah praktis untuk menghasilkan konfigurasi *default*.

```
sudo apt-get update && sudo apt-get install -y containerd
sudo mkdir -p /etc/containerd
sudo containerd config default | sudo tee
/etc/containerd/config.toml
sudo systemctl restart containerd
sudo systemctl enable containerd
```

Menonaktifkan *swap* pada *host* yang akan digunakan

```
sudo swapoff -a
sudo sed -i '/ swap / s/^\(.*\)$/#\1/g' /etc/fstab
```

Melakukan *update system* pada *host server* dan melakukan instalasi apt-transport-https dan juga menambahkan paket dependensi dari *Google* untuk menjalankan Kubernetes.

```
sudo apt update && sudo apt-get install -y apt-transport-https
curl
curl -s https://packages.cloud.google.com/apt/doc/apt-key.gpg |
sudo apt-key add -
```

Menambahkan repositori (kumpulan direktori) dari Kubernetes.io dan melakukan perintah *update system*.

```
cat <<EOF | sudo tee /etc/apt/sources.list.d/kubernetes.list
deb https://apt.kubernetes.io/ kubernetes-xenial main
EOF
sudo apt update
```

Menginstal kubelet (*service* Kubernetes), kubeadm (menghubungkan cluster), dan kubectl (berisi *command* perintah) dengan versi 1.22 untuk semuanya dan melakukan perintah penandaan untuk terus berjalan pada aplikasi kubelet, kubeadm, dan kubectl.

```
sudo apt-get install -y kubelet=1.22.1-00 kubeadm=1.22.1-00
kubectl=1.22.1-00
sudo apt-mark hold kubelet kubeadm kubectl
```

3.6.2.2 Konfigurasi untuk *Master Node*

Inisialisasi *cluster* Kubernetes dengan membuat *file* konfigurasi yaml yang berisikan versi API dan IP *subnet* yang akan digunakan oleh *pod* nantinya.

```
sudo nano kubeadm-config.yaml
```

```
apiVersion: kubeadm.k8s.io/v1beta2
kind: ClusterConfiguration
kubernetesVersion: stable
networking:
    podSubnet: "10.244.0.0/16"
```

Menjalankan perintah *kubeadm init* untuk membuat *cluster* dengan konfigurasi dari *file* yaml yang telah dibuat dan membuat *certificate* untuk *worker node join* kedalam *cluster* Kubernetes.

```
kubeadm init --config=kubeadm-config.yaml --upload-certs
```

Membuat direktori *kube* pada *home* dan melakukan perintah *copy* untuk konfigurasi Kubernetes. Merubah pengaturan *owner* untuk *directori config*.

```
mkdir -p $HOME/.kube
sudo cp -i /etc/kubernetes/admin.conf $HOME/.kube/config
sudo chown $(id -u):$(id -g) $HOME/.kube/config
```

Menginstal Kubernetes CNI *Calico* untuk komunikasi antar *host* pada Kubernetes *cluster*.

```
kubectl apply -f
https://docs.projectcalico.org/manifests/calico.yaml
kubectl get pods -n kube-system
```

3.6.2.3 Konfigurasi untuk *Worker Node1 dan Worker Node 2*

Proses selanjutnya yaitu menghubungkan worker node 1 dan worker node 2 ke dalam cluster Kubernetes dengan menggunakan perintah sebagai berikut.

```
kubeadm join 10.128.0.24:6443 --token jtddqd.ywyv3qfffi3g3dsk --
discovery-token-ca-cert-hash
sha256:0f023ee1bd0341f5985be3c20cc108072193fe20f45491f46a6ff4b35
56df861
```

3.6.3 *Instalasi Web Server*

Instalasi *web server* menggunakan *file* konfigurasi yaml. Aplikasi *web server* yang digunakan yaitu *Nginx*, ada beberapa tahapan penginstalan *web server Nginx* pada Kubernetes. Tahapan pertama yaitu membuat *configmap*, configurai aplikasi *Nginx*, dan membuat *service* untuk *Nginx*. Tahapan pertama yaitu membuat file configmap.yaml untuk melakukan konfigurasi *directory index* html aplikasi *Nginx*.

```
sudo nano index-html-configmap.yaml
```

```
apiVersion: v1
kind: ConfigMap
metadata:
  name: index-html-configmap
  namespace: default
data:
  index.html: |
    <html>
    <h1>Welcome</h1>
    </br>
    <h1>Hi! This is a configmap Index file </h1>
    </html>
```

Tahapan selanjutnya yaitu membuat *file* nginx-deployment.yaml untuk melakukan konfigurasi aplikasi *Nginx* yang akan di instal.

```
sudo nano nginx-deployment.yaml
```

Konfigurasi yang digunakan pada aplikasi *Nginx* yaitu menggunakan versi 1.18. Aplikasi *nginx* akan di instal pada kedua *worker node* yaitu *kube-wokernode02* dan *kube-wokernode02* dengan masing-masing memiliki dua kontainer berisi aplikasi *nginx* didalamnya. Total kontainer yang digunakan yaitu berjumlah 4 kontainer.

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: nginx-deployment
  namespace: default
spec:
  selector:
    matchLabels:
```

```
      app: nginx
  replicas: 4
  template:
    metadata:
      labels:
        app: nginx
    spec:
      containers:
      - name: nginx
        image: nginx:1.18.0
        ports:
        - containerPort: 80
        volumeMounts:
            - name: nginx-index-file
              mountPath: /usr/share/nginx/html/
      volumes:
      - name: nginx-index-file
        configMap:
          name: index-html-configmap
```

Tahapan terakhir dalam menginstal *Nginx* yaitu membuat *file* nginx-service.yaml yang berfungsi agar *service* aplikasi *nginx* dapat bisa diakses dari luar.

```
sudo nano nginx-service.yaml
```

Konfigurasi *Nginx service* yaitu berisikan tipe *service* yang digunakan yaitu *load balancer* dengan alamat eksternal ip *load balancer*nya yaitu 34.27.7.253 dan menggunakan *port* 30062.

```
apiVersion: v1
kind: Service
metadata:
  name: nginx-service
  namespace: default
spec:
  selector:
    app: nginx
  type: LoadBalancer
  externalIPs:
  - 34.27.7.253
  ports:
    - name: http
      port: 80
      nodePort: 30062
```

3.7 Pengambilan Data Penelitian

Data yang diamati untuk dianalisis yaitu parameter *benchmark* yang terdiri dari *response time*, *throughput*, CPU *usage*, dan *memory* dari tiga jenis serangan DoS yaitu TCP *flood*, UDP *flood*, dan *smurf attack* sebanyak 30 kali percobaan menggunkan *Hping3* di install pada *client*. Data tersebut diambil dengan *tools Siege* dan *Htop* yang diinstal pada *master node*.

3.7.1 Tidak Mengalami Serangan DoS

Pengambilan data ketika tidak mengalami serangan DoS dilakukan dengan mengakses alamat ip nginx *service* yang dapat di lihat pada *master node*. Alamat *service* nginx yang di gunakan yaitu http://34.170.24.55:30062/ yang merupakan gabungan dari alamat ip *loadbalancer* dan nomor *port* yang digunakan seperti pada gambar 3.7.

```
kiki@kube-masternode:~$ kubectl get service
NAME            TYPE           CLUSTER-IP     EXTERNAL-IP    PORT(S)        AGE
kubernetes      ClusterIP      10.96.0.1      <none>         443/TCP        2d20h
nginx-service   LoadBalancer   10.97.25.105   34.170.24.55   80:30062/TCP   3h8m
```

Gambar 3.7 IP nginx *service.*

Setelah mendapatkan alamat nginx *service* selanjutnya alamat tersebut akan diakses menggunakan aplikasi siege yang di *install* pada *client*. Pengambilan data menggunakan aplikasi siege berfungsi untuk mengambil data *response time* dan *throughput* seperti pada gambar 3.8.

```
kiki@user-attacker:~$ sudo siege http://34.170.24.55:30062/ -c10 -t10s
** SIEGE 4.0.4
** Preparing 10 concurrent users for battle.
The server is now under siege...
Lifting the server siege...
Transactions:                   1658 hits
Availability:                 100.00 %
Elapsed time:                   9.90 secs
Data transferred:              54.55 MB
Response time:                  0.06 secs
Transaction rate:             167.47 trans/sec
Throughput:                     5.51 MB/sec
Concurrency:                    9.97
Successful transactions:        1658
Failed transactions:               0
Longest transaction:            1.01
Shortest transaction:           0.00
```

Gambar 3.8 Pengambilan data siege tidak mengalami DoS.

Untuk pengambilan data CPU *usage* dan *Memory* mengguakan aplikasi Htop yang di *install* pada *master* node, pengambilan data dengan memantau angka persentasi tertinggi yang dicapai oleh CPU dan juga *Memory* seperti pada gambar 3.9.

Gambar 3.9 Pengambilan data htop tidak mengalami DoS.

3.7.2 Serangan DoS TCP Flood

Pengambilan data ketika mengalami serangan DOS TCP Flood dilakukan ketika webserver dalam keaadaan mengalami serangan oleh aplikasi hping3 yang di jalankan oleh *client*. Selama proses pengambilan data untuk seluruh parameter penyerangan TCP Flood terus dilakukan. Adapun perintah yang digunakan yaitu "sudo hping3 -S --flood -V -p 30062 34.170.24.55" yang berarti menjalankan *flooding* TCP menggunakan aplikasi hping3 dengan tujuan webserver dengan alamat ip 34.27.7.253 dan nomor port 30062 seperti pada gambar 3.10.

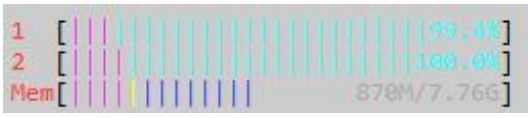
```
kiki@user-attacker:~$ sudo hping3 -S --flood -V -p 30062 34.170.24.55
using ens4, addr: 10.128.0.17, MTU: 1460
HPING 34.170.24.55 (ens4 34.170.24.55): S set, 40 headers + 0 data bytes
hping in flood mode, no replies will be shown
```

Gambar 3.10 Proses penyerangan TCP flood.

Pengambilan data untuk parameter *response time* dan *throughput* menggunakan siege pada *master node*. Dengan menggunakan perintah "sudo siege http://34.170.24.55:30062/ -c10 -t10s" yang berarti menjalakan program siege dengan alamat *url* tujuan dan juga menggunakan jumlah koneksi sebesar 10 dengan waktu 10 detik seperti pada gambar 3.11. dilanjutkan dengan pengambilan data untuk parameter CPU *usage* dan *Memory* menggunakan htop seperti pada gambar 3.12.

```
Lifting the server siege...
Transactions:                     48 hits
Availability:                  56.47 %
Elapsed time:                   9.54 secs
Data transferred:               1.58 MB
Response time:                  0.06 secs
Transaction rate:               5.03 trans/sec
Throughput:                     0.17 MB/sec
Concurrency:                    0.31
Successful transactions:          48
Failed transactions:              37
Longest transaction:            0.13
Shortest transaction:           0.00
```

Gambar 3.11 Pengambilan data siege mengalami DoS TCP Flood.



Gambar 3.12 Pengambilan data htop mengalami DoS TCP Flood.

3.7.3 Serangan DoS UDP *Flood*

Pengambilan data ketika mengalami serangan DOS UDP *Flood* dilakukan ketika *webserver* dalam keaadaan mengalami serangan oleh aplikasi hping3 yang di jalankan oleh *client*. Selama proses pengambilan data untuk seluruh parameter penyerangan UDP *Flood* terus dilakukan. Adapun perintah yang digunakan yaitu "sudo hping3 -S --udp --flood -V -p 30062 34.170.24.55" yang berarti menjalankan *flooding* UDP menggunakan aplikasi hping3 dengan tujuan *webserver* dengan alamat ip 34.170.24.55 dan nomor port 30062 seperti pada gambar 3.13.

```
kiki@user-attacker:~$ sudo hping3 -S --udp --flood -V -p 30062 34.170.24.55
using ens4, addr: 10.128.0.17, MTU: 1460
HPING 34.170.24.55 (ens4 34.170.24.55): udp mode set, 28 headers + 0 data bytes
hping in flood mode, no replies will be shown
```

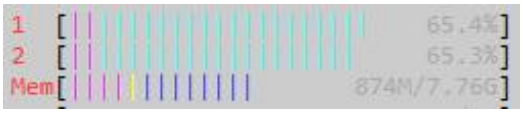
Gambar 3.13 Proses penyerangan UDP flood.

Pengambilan data untuk parameter *response time* dan *throughput* menggunakan siege pada *master node*. Dengan menggunakan perintah "sudo siege http://34.170.24.55:30062/ -c10 -t10s" yang berarti menjalakan program siege dengan alamat *url* tujuan dan juga menggunakan jumlah koneksi sebesar 10 dengan waktu 10 detik seperti pada gambar 3.14. dilanjutkan dengan pengambilan data

untuk parameter CPU *usage* dan *Memory* menggunakan htop seperti pada gambar 3.15.

```
kiki@user-attacker:~$ sudo siege http://34.170.24.55:30062/ -c10 -t10s
** SIEGE 4.0.4
** Preparing 10 concurrent users for battle.
The server is now under siege...
Lifting the server siege...
Transactions:                   1605 hits
Availability:                 100.00 %
Elapsed time:                   9.20 secs
Data transferred:              53.53 MB
Response time:                  0.06 secs
Transaction rate:             174.46 trans/sec
Throughput:                     5.82 MB/sec
Concurrency:                    9.95
Successful transactions:        1605
Failed transactions:               0
Longest transaction:            0.12
Shortest transaction:           0.00
```

Gambar 3.14 Pengambilan data siege mengalami DoS UDP Flood.



Gambar 3.15 Pengambilan data htop mengalami DoS UDP Flood.

3.7.4 Serangan DoS Smurf Attack

Pengambilan data ketika mengalami serangan DOS *Smurf Attack* dilakukan ketika webserver dalam keaadaan mengalami serangan oleh aplikasi hping3 yang di jalankan oleh *client*. Selama proses pengambilan data untuk seluruh parameter penyerangan *Smurf Attack* terus dilakukan. Adapun perintah yang digunakan yaitu "sudo hping3 -S --icmp --flood -V -p 30062 34.170.24.55" yang berarti menjalankan *flooding* icmp menggunakan aplikasi hping3 dengan tujuan webserver dengan alamat ip 34.170.24.55 dan nomor port 30062 seperti pada gambar 3.16.

```
kiki@user-attacker:~$ sudo hping3 -S --icmp --flood -V -p 30062 34.170.24.55
using ens4, addr: 10.128.0.17, MTU: 1460
HPING 34.170.24.55 (ens4 34.170.24.55): icmp mode set, 28 headers + 0 data bytes
hping in flood mode, no replies will be shown
```

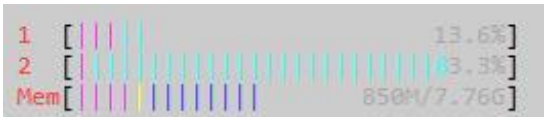
Gambar 3.16 Proses penyerangan Smurf Attack.

Pengambilan data untuk parameter *response time* dan *throughput* menggunakan siege pada *master node*. Dengan menggunakan perintah "sudo siege http://34.170.24.55:30062/ -c10 -t10s" yang berarti menjalakan program siege

dengan alamat *url* tujuan dan juga menggunakan jumlah koneksi sebesar 10 dengan waktu 10 detik seperti pada gambar 3.17. dilanjutkan dengan pengambilan data untuk parameter CPU *usage* dan *Memory* menggunakan htop seperti pada gambar 3.18.

```
kiki@user-attacker:~$ sudo siege http://34.170.24.55:30062/ -c10 -t10s
** SIEGE 4.0.4
** Preparing 10 concurrent users for battle.
The server is now under siege...
Lifting the server siege...
Transactions:                   1390 hits
Availability:                 100.00 %
Elapsed time:                   9.10 secs
Data transferred:              46.26 MB
Response time:                  0.06 secs
Transaction rate:             152.75 trans/sec
Throughput:                     5.08 MB/sec
Concurrency:                    9.92
Successful transactions:        1390
Failed transactions:               0
Longest transaction:            1.02
Shortest transaction:           0.00
```

Gambar 3.17 Pengambilan data siege mengalami DoS Smurf Attack.



Gambar 3.18 Pengambilan data htop mengalami DoS Smurf Attack.

3.8 Analisa Data

Tahap ini dilakukan ketika pengambilan data dari hasil simulasi selesai dan dapat menghasilkan keluaran untuk dianalisis. Analisis yang dilakukan yaitu dengan membandingkan hasil *response time*, *throughput*, CPU *usage*, dan *memory* dari tiga jenis serangan DoS yaitu TCP *flood*, UDP *flood*, dan *smurf attack* serta kertika server tidak mengalami serangan.