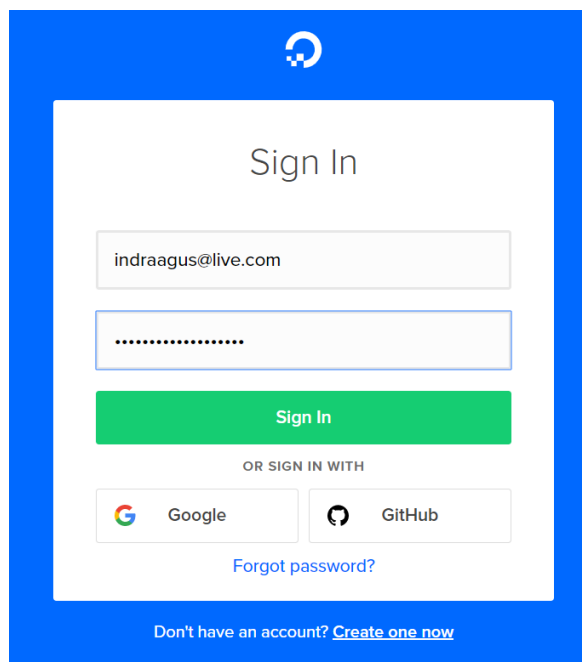


CARA MENJALANKAN PROGRAM

SISTEM OTOMASI PROVISIONING STORAGE AS A SERVICE

A. Menjalankan Cluster Kubernetes

Untuk membuat cluster Kubernetes login terlebih dahulu untuk masuk ke dashboard console Digital Ocean pada URL <https://cloud.digitalocean.com/login>.



Sign In

indraagus@live.com

.....

Sign In

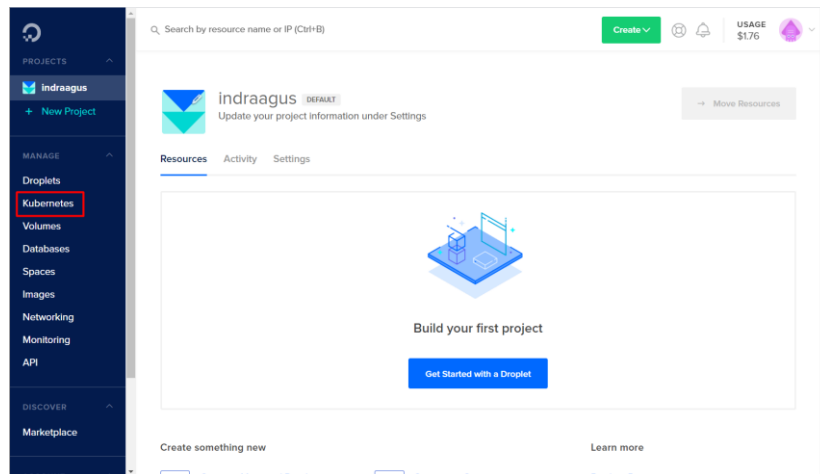
OR SIGN IN WITH

Google GitHub

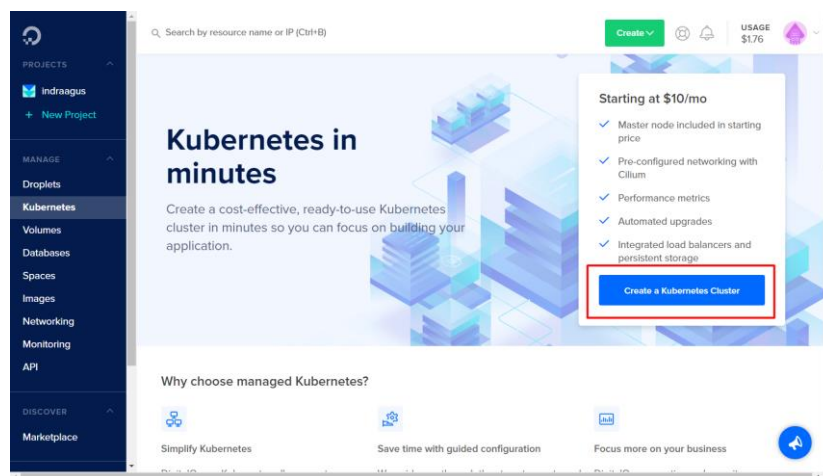
[Forgot password?](#)

Don't have an account? [Create one now](#)

Lalu pada dashboard pilih menu Kubernetes (bagian yang ditandai merah).



Setelah masuk ke menu Kubernetes klik “Create a Kubernetes Cluster” pada bagian yang ditandai seperti gambar berikut.



Setelah itu pilih versi Kubernetes yang akan digunakan. Dalam penelitian ini versi Kubernetes yang akan digunakan adalah versi 1.14.8-do.2.

Create a cluster

Select a Kubernetes version

Select the Kubernetes version. The newest available version is selected by default.

116.2-do.1 (latest) Tip: We generally recommend the latest version unless your team has a specific need. See the DigitalOcean Kubernetes release notes.









116.2-do.1 (latest) ✓

115.5-do.2 ✓

114.8-do.2

Enter region









be located in a single datacenter.

 New York 1 2 3	 Amsterdam 1 2 3	 San Francisco 1 2	 Singapore 1	 London 1	 Frankfurt 1
 Toronto 1	 Bangalore 1				

Setelah itu pilih lokasi *data center* yang diinginkan. Pilih negara yang paling dekat untuk meminimalkan latensi akses ke *cluster*.

Choose a datacenter region

Your Kubernetes cluster will be located in a single datacenter.

 New York 1 2 3	 Amsterdam 1 2 3	 San Francisco 1 2	 Singapore 1	 London 1	 Frankfurt 1
 Toronto 1	 Bangalore 1				

Dalam penelitian ini lokasi *data center* yang digunakan adalah Singapore yang merupakan negara dengan jarak terdekat dari lokasi penelitian yaitu Indonesia.

Setelah memilih *data center*, pilih spesifikasi server yang akan dijadikan *node* Kubernetes.

Choose cluster capacity ?

Increasing the number of nodes in a pool lets you run more instances of the scheduled services. Adding more node pools allows you to schedule pods to different node pools so each pod has the RAM, CPU, and storage it requires. You can add and remove nodes and node pools at any time.

NODE POOL NAME	MACHINE TYPE (DROPLET)	NODE PLAN	NUMBER NODES
<input type="text" value="pool-skripsi"/>	<input type="text" value="Standard nodes"/> Balanced with a healthy amount of m...	<input type="text" value="\$10/Month per node (\$0.015/hr)"/> Includes: 2 GB Memory / 1 vCPU	<input type="text" value="2"/>
<input type="button" value="Add Additional Node Pool"/>			

MONTHLY RATE **\$20.00/month** \$0.03/hour

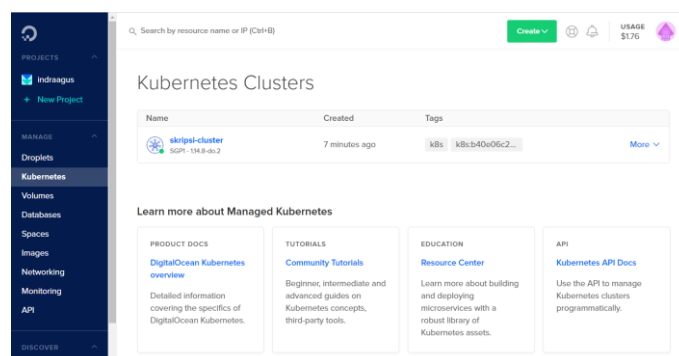
Dalam penelitian spesifikasi server *node* yang dipilih sesuai dengan gambar di atas yaitu 2GB RAM dan 1 vCPU dengan jumlah unit sebanyak 2 unit.

Setelah itu lakukan klik “Create Cluster” di bagian paling bawah halaman.

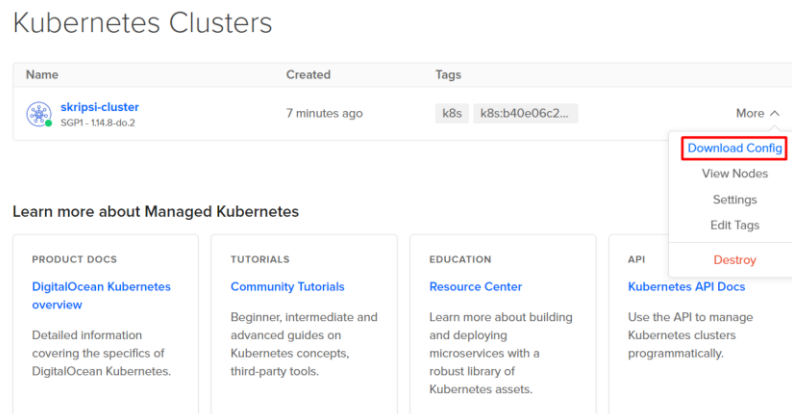
Add Tags
Add optional tags to your cluster.

Choose a name
You can edit the default name to something meaningful to you.

Setelah itu tunggu beberapa menit sampai status cluster *ready* seperti pada gambar berikut.



Setelah *cluster* siap digunakan *download* konfigurasi cluster dengan cara seperti pada gambar berikut.



Klik “Download Config” pada menu “More” pada daftar cluster yang ditampilkan. Kemudian browser akan melakukan *download* file dalam format “.yaml” yang berisi format konfigurasi akses ke *cluster* yang telah dibuat.

Konfigurasi Metric Server API

Metric server API pada Kubernetes merupakan fitur yang dibuat oleh para developer Kubernetes. Akan tetapi *metric server API* tidak dijalankan secara default pada saat *cluster* pertama kali dijalankan.

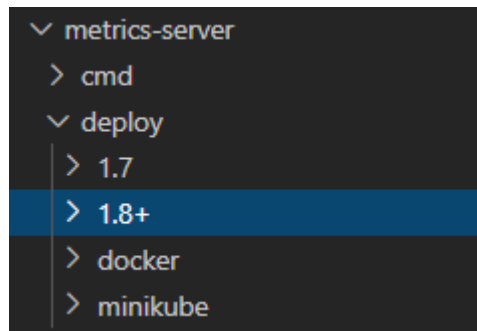
Untuk melakukan konfigurasi *metric server API* dibutuhkan file konfigurasi yang telah di-*download* sebelumnya untuk dijadikan konfigurasi *tool* kubectl yang digunakan untuk melakukan konfigurasi *cluster* secara *remote*. Untuk instalasi *tool* kubectl bisa dilakukan dengan cara yang dijelaskan pada halaman web <https://kubernetes.io/docs/tasks/tools/install-kubectl/> . Setelah *tool* kubectl siap digunakan *copy* file konfigurasi yang telah di-*download* ke folder “.kube” pada *home directory* dengan nama file “config”. Untuk letak *home directory* tergantung

pada sistem operasi yang digunakan untuk melakukan konfigurasi *remote*. Dalam penelitian ini sistem operasi yang digunakan adalah Ubuntu. *Command* yang digunakan untuk meng-*copy* file konfigurasi tersebut adalah seperti berikut.

```
cp skripsi-cluster-kubeconfig.yaml $HOME/.kube/config
```

Langkah pertama untuk menjalankan fitur *metric server API* adalah melakukan *clone repository* github berikut: <https://github.com/kubernetes-sigs/metrics-server.git>. Pada *repository* tersebut terdapat code program untuk fitur *metric server API* Kubernetes serta file konfigurasi Kubernetes dalam format “.yaml” pada folder “deployment”.

Pada folder “deployment” terdapat beberapa sub-folder. Dimana nama folder-folder ini mewakili kompatibilitas versi *cluster* Kubernetes yang sesuai dengan file konfigurasi yang ada di dalamnya.



Karena dalam penelitian ini versi Kubernetes yang digunakan adalah versi 1.14.8 maka file konfigurasi yang kompatibel dengan versi tersebut adalah file-file yang terdapat dalam folder “1.8+”.

Sebelum dilakukan penerapan konfigurasi yang ada pada folder “deploy/1.8+” berdasarkan dari temuan yang dijelaskan pada artikel web <https://medium.com/@cagri.ersen/kubernetes-metrics-server-installation-d93380de008>, harus dilakukan penambahan atribut “command” pada file konfigurasi “metrics-server-deployment.yaml” dengan *value* seperti pada gambar berikut.

```
31  - name: metrics-server
32  image: k8s.gcr.io/metrics-server-amd64:v0.3.6
33  command:
34  - /metrics-server
35  - --kubelet-insecure-tls
36  - --kubelet-preferred-address-types=InternalIP
37  args:
38  - --cert-dir=/tmp
39  - --secure-port=4443
40  ports:
41  - name: main-port
42    containerPort: 4443
43    protocol: TCP
```

Tambahan atribut “command” beserta nilainya seperti pada bagian yang ditandai pada gambar dibawah *line* nomor 32 pada file “metrics-server-deployment.yaml” akan mematikan konfigurasi TLS pada akses *metric server API* terhadap data *load* setiap *node* dan mengarahkan pengambilan data *load node* ke ip internal setiap *node* dimana pada beberapa provider cloud prosedur keamanan akses ip internal lebih terbuka daripada ip eksternal (public).

Setelah itu dengan menggunakan *tool* kubectl dilakukan penerapan file-file konfigurasi yang ada dengan menggunakan *command* seperti berikut.

```
kubectl apply -f deploy/1.8+/
```

Command tersebut menghasilkan output seperti berikut.

```
clusterrole.rbac.authorization.k8s.io/system:aggregated-metrics-reader created
clusterrolebinding.rbac.authorization.k8s.io/metrics-server:system:auth-delegator created
rolebinding.rbac.authorization.k8s.io/metrics-server-auth-reader created
apiservice.apiregistration.k8s.io/v1beta1.metrics.k8s.io created
serviceaccount/metrics-server created
deployment.apps/metrics-server created
service/metrics-server created
clusterrole.rbac.authorization.k8s.io/system:metrics-server created
clusterrolebinding.rbac.authorization.k8s.io/system:metrics-server created
```

Setelah itu dilakukan pengecekan konfigurasi *metric server API* apakah sudah siap digunakan dengan menggunakan *command* berikut.

```
kubectl top node
```

Command tersebut menghasilkan output tampilan data load node. Dengan demikian maka *metric server API* sudah siap digunakan.

```
root@DESKTOP-RGCIAKR:~# kubectl top node
NAME                CPU(cores)   CPU%   MEMORY(bytes)   MEMORY%
pool-skripsi-sd7j   56m          5%     789Mi           39%
pool-skripsi-sd7o   128m         12%     813Mi           40%
root@DESKTOP-RGCIAKR:~#
```

Konfigurasi Ingress Nginx

Konfigurasi *Ingress Nginx* dibutuhkan agar dapat dilakukan konfigurasi *unique endpoint* melalui Kubernetes API untuk akses owncloud dari setiap akun yang terdaftar berdasarkan data *username* masing-masing akun.

Konfigurasi awal yang dibutuhkan adalah membuat objek-objek yang dibutuhkan agar *ingress nginx* dapat berjalan dengan menggunakan file konfigurasi yang sudah disediakan oleh *developer* dari Nginx pada url

[https://raw.githubusercontent.com/kubernetes/ingress-](https://raw.githubusercontent.com/kubernetes/ingress-nginx/master/deploy/static/mandatory.yaml)

[nginx/master/deploy/static/mandatory.yaml](https://raw.githubusercontent.com/kubernetes/ingress-nginx/master/deploy/static/mandatory.yaml). Konfigurasi awal ini dilakukan

dengan cara mengeksekusi *command* berikut.


```
kubectl apply -f
```

```
https://raw.githubusercontent.com/kubernetes/ingress-  
nginx/master/deploy/static/mandatory.yaml
```

```
namespace/ingress-nginx created  
configmap/nginx-configuration created  
configmap/tcp-services created  
configmap/udp-services created  
serviceaccount/nginx-ingress-serviceaccount created  
clusterrole.rbac.authorization.k8s.io/nginx-ingress-clusterrole created  
role.rbac.authorization.k8s.io/nginx-ingress-role created  
rolebinding.rbac.authorization.k8s.io/nginx-ingress-role-nisa-binding created  
clusterrolebinding.rbac.authorization.k8s.io/nginx-ingress-clusterrole-nisa-binding created  
deployment.apps/nginx-ingress-controller created
```

Setelah itu dilakukan pengecekan *pod* pada namespace “ingress nginx” untuk mengetahui apakah *deployment pod* server *ingress* sudah berjalan dengan *command*.

```
kubectl get pod -n ingress-nginx
```

Command di atas menghasilkan output seperti berikut.

```
root@DESKTOP-RGCIAKR:~# kubectl get pod -n ingress-nginx  
NAME                                READY   STATUS    RESTARTS   AGE  
nginx-ingress-controller-7dcc95dfbf-1l4sx  1/1     Running   0           4m47s  
root@DESKTOP-RGCIAKR:~#
```

Output tersebut menandakan bahwa *pod* server *ingress* sudah berjalan.

Setelah itu dilakukan pembuatan *service* dengan tipe *LoadBalancer* dengan *command* berikut.

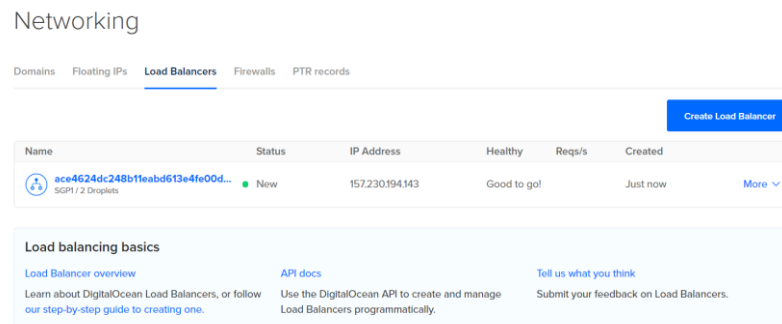
```
kubectl apply -f  
https://raw.githubusercontent.com/kubernetes/ingress-  
nginx/master/deploy/static/provider/cloud-generic.yaml
```

Command tersebut menghasilkan output seperti berikut.

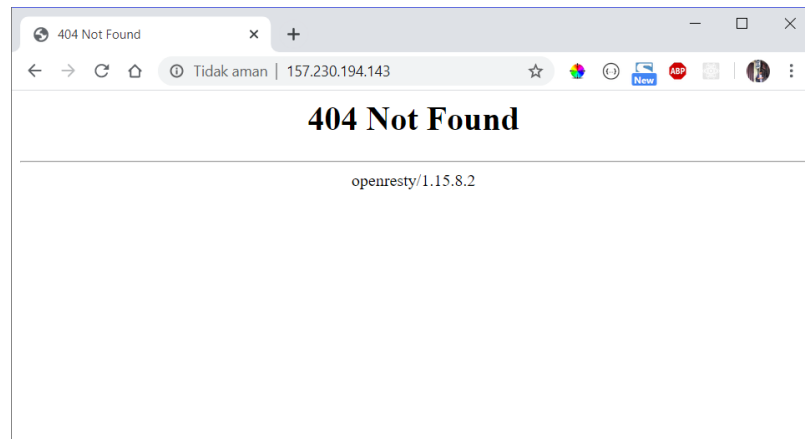
```
service/ingress-nginx created
```

Command tersebut menerapkan konfigurasi *service* yang ada pada file “cloud-generic.yaml” yang disediakan oleh *developer* Nginx. File konfigurasi “cloud-generic.yaml” yang disediakan ditujukan untuk konfigurasi Kubernetes yang dijalankan pada *cloud provider* secara umum. Akan tetapi untuk *cloud provider* Microsoft Azure dan AWS memiliki file konfigurasi dan prosedur yang berbeda.

Pada *dashboard* Digital Ocean pada menu “Networking” pada *tab* “Load Balancers” terdapat *load balancer instance* baru yang terhubung dengan *ingress nginx service* pada cluster Kubernetes yang baru saja dibuat.



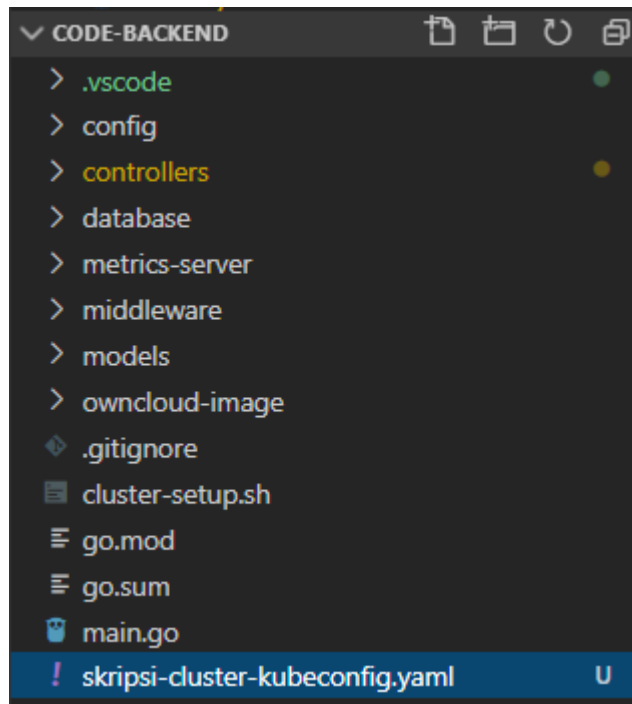
Lalu dilakukan pengecekan *endpoint* dengan mengakses port 80 pada ip *load balancer instance* yang ditampilkan pada *dashboard* Digital Ocean. Hasilnya adalah seperti pada gambar berikut.



Didapatkan *response* 404 yang disebabkan karena belum ada konfigurasi *ingress* yang diterapkan pada *ingress server* yang telah dikonfigurasi pada *cluster*.

B. Menjalankan Program Backend

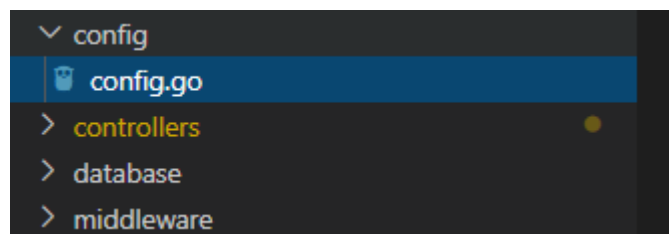
Pastikan bahasa pemrograman go versi 1.12.6 (atau lebih baru) sudah terinstall pada sistem operasi. Untuk menjalankan program backend pastikan file konfigurasi akses cluster Kubernetes yang telah dibuat sudah didownload. Ubah nama filenya menjadi “skripsi-cluster-kubeconfig.yaml” dan letakan dalam root directory file code program backend seperti berikut.



Setelah itu jalankan command :

```
go mod init
```

Command tersebut akan melakukan download library yang dibutuhkan oleh code program. Setelah itu import data sql ke server database yang digunakan lalu sesuaikan konfigurasi program dengan mengedit file config.go pada folder config.



Edit konfigurasi program dengan mengubah nilai-nilai variabel pada fungsi SetConfig() dengan contoh seperti pada gambar berikut.

```

func SetConfig() Config {
    var config Config

    //set configuration here
    config.Host = "localhost"
    config.Port = "3306"
    config.DbUser = "root"
    config.DbName = "provisioning_owncloud"
    config.DbPass = ""
    config.HttpPort = "1235"
    config.SrvKey = "Aw4s_g414k"
    return config
}

```

Setelah itu jalankan program dengan menjalankan script main.go pada root directory folder seperti berikut.

```

PS C:\Users\Indra Agus Setiawan\Documents\code-backend> go run .\main.go
LOADED DB CONNECTION : root:@tcp(127.0.0.1:3306)/provisioning_owncloud?charset=utf8&parseTime=True&loc=Local
Creating tables...
[GIN-debug] [WARNING] Creating an Engine instance with the Logger and Recovery middleware already attached.

[GIN-debug] [WARNING] Running in "debug" mode. Switch to "release" mode in production.
- using env:   export GIN_MODE=release
- using code:  gin.SetMode(gin.ReleaseMode)

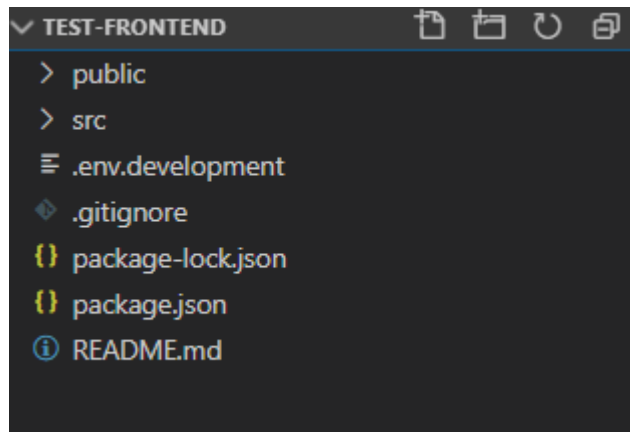
[GIN-debug] POST   /api/login/          --> github.com/seregant/golang_k8s_provisioning/controllers.(*AuthController).GenerateToken-fm (4 handlers)
[GIN-debug] GET    /api/pengguna/       --> github.com/seregant/golang_k8s_provisioning/controllers.(*Pengguna).GetAll-fm (5 handlers)
[GIN-debug] POST   /api/pengguna/add    --> github.com/seregant/golang_k8s_provisioning/controllers.(*Pengguna).Add-fm (5 handlers)
[GIN-debug] GET    /api/pengguna/u      --> github.com/seregant/golang_k8s_provisioning/controllers.(*Pengguna).GetDataPengguna-fm (6 handlers)
[GIN-debug] GET    /api/clusters/nodes --> github.com/seregant/golang_k8s_provisioning/controllers.(*Nodes).GetNodesData-fm (5 handlers)
[GIN-debug] Listening and serving HTTP on :1235

```

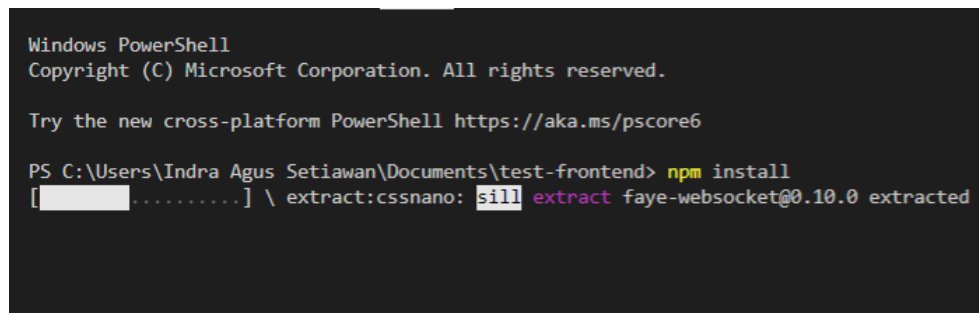
Program dijalankan dengan command “**go run main.go**” dan berjalan pada port 1235 sesuai dengan konfigurasi pada file config.go pada nilai variabel config.HttpPort.

C. Menjalankan program frontend

Pastikan sistem sudah terinstal bahasa pemrograman NodeJS dengan versi v10.16.3 ke atas. Untuk menjalankan program frontend terlebih dahulu masuk ke document root dari program seperti berikut.



Setelah itu jalankan command “npm install” untuk mendownload library yang dibutuhkan.



Setelah proses selesai jalankan program frontend dengan command “**npm start**”. Program frontend akan berjalan pada port 3008.

Untuk membuat user admin lakukan registrasi biasa lalu edit data table “is_admin” ganti nilainya menjadi 1 pada data user yang ingin dijadikan admin.