Membuat dan Menjalankan aplikasi di Linux SystemD

```
$ sudo systemctl start useless.service
```

wuriyan.to

Apa itu Linux SystemD

systemd adalah *system* dan *service manager* untuk sistem operasi varian **Linux**. Sesuai pada akhiran namanya **“d”** yang berarti ***“daemon”***, **systemd** digunakan oleh **Linux** untuk menjalankan sebuah *service* atau aplikasi, misalnya **http server**, **database server**, dan lain-lain pada **mode background**, *start-stop*, *tracking*, dan *monitoring*. *Tool* utama dari **systemd** adalah **systemctl**. **Systemctl** bisa kita gunakan untuk *start-stop* sebuah *service* ataupun hanya sekedar kita gunakan untuk monitoring.

```
$ sudo systemctl status
```

Penggunaan

Penggunaan paling sederhana yaitu untuk melihat status sistem yang sedang berjalan.

```
$ sudo systemctl status
```

```
● ubuntu-bionic
    State: running
     Jobs: 0 queued
   Failed: 0 units
    Since: Sat 2020-07-18 05:16:07 UTC; 2 days ago
   CGroup: /
           ├─user.slice
           │ └─user-1000.slice
           │   ├─user@1000.service
```

Atau untuk me-restart dan melihat status dari **nginx** yang ada di **server** kita.

```
$ sudo systemctl restart nginx
```

```
$ sudo systemctl status nginx
```

```
● nginx.service - A high performance web server and a reverse proxy server
   Loaded: loaded (/lib/systemd/system/nginx.service; enabled; vendor preset: enabled)
   Active: active (running) since Mon 2020-07-20 05:59:18 UTC; 2min 0s ago
     Docs: man:nginx(8)
  Process: 25208 ExecStop=/sbin/start-stop-daemon --quiet --stop --retry QUIT/5 --pidfile /run/nginx.pid (code=exited, status=0/SUCCESS)
  Process: 25223 ExecStart=/usr/sbin/nginx -g daemon on; master_process on; (code=exited, status=0/SUCCESS)
  Process: 25210 ExecStartPre=/usr/sbin/nginx -t -q -g daemon on; master_process on; (code=exited, status=0/SUCCESS)
 Main PID: 25225 (nginx)
    Tasks: 5 (limit: 4702)
   CGroup: /system.slice/nginx.service
           ├─25225 nginx: master process /usr/sbin/nginx -g daemon on; master_process on;
           ├─25227 nginx: worker process
           ├─25228 nginx: worker process
           ├─25229 nginx: worker process
           └─25230 nginx: worker process
```

Menjalankan service tanpa systemd

Aplikasi, *service* atau apapun yang sedang kita jalankan di *terminal* utama kita sebenarnya sedang berjalan di **main process**. Jadi ketika kita menekan **CTRL+C** atau kita menutup *terminal* pada *server* kita, maka otomatis aplikasi atau *service* tersebut juga akan ikut di *terminate/shut down*.

Misalnya saya menjalankan sebuah *script* **python** pada sebuah *terminal*.

```
$ python server.py
```

Kemudian kita menekan **CTRL+C** dan kemudian *logout* dari *server*, maka proses yang berjalan di **server.py** akan otomatis mati. Dengan menggunakan **systemd**, hal ini dapat kita solusikan.

Untuk contoh penggunaan **systemd** kali ini, saya akan menggunakan **Poppy (TCP Echo server written in Kotlin)**, **Poppy** ini adalah contoh **TCP echo server** yang saya buat dengan **Kotlin**. Jadi jika anda ingin mengikuti tutorial ini, maka *requirement* yang perlu anda penuhi dan dipasang di *server* anda adalah:

- **Java 8 atau lebih tinggi**
- **Maven**
- **git**
- **Ubuntu Server**

Silahkan *clone repository code*nya

```
$ git clone https://github.com/telkomdev/Poppy.git
```

Build dengan **Maven**.

```
$ cd Poppy
```

```
$ mvn clean package
```

Kemudian kita coba jalankan secara manual terlebih dahulu tanpa menggunakan **systemd**.

```
$ PORT=9000 QUEUE_SIZE=100 java -jar target/Poppy-1.0-SNAPSHOT-jar-with-dependencies.jar
```

Untuk menjalankan Poppy kita perlu dua *environment variable*, yaitu **PORT** dan **QUEUE_SIZE**.

```
vagrant@ubuntu-bionic:~/Poppy$ PORT=9000 QUEUE_SIZE=100 java -jar target/Poppy-1.0-SNAPSHOT-jar-with-dependencies.jar
waiting for client connection on port 9000
```

Tapi apa yang terjadi jika kita meninggalkan atau *logout* dari *server* kita?

Yang terjadi adalah **Poppy** *server* yang sudah kita jalankan tadi akan *terminate*. Seperti yang sudah saya sampaikan sebelumnya, karena **Poppy** berjalan di **main process**.

Nah disinilah kita membutuhkan systemd untuk membantu menjalankan *server* **Poppy** kita pada *daemon mode*.

Membuat dan menjalankan service dengan systemd

Untuk menjalankan Poppy pada systemd, kita perlu membuat *Unit file service* secara manual pada *folder* **/etc/systemd/system** kemudian *service* tersebut kita beri nama **poppy.service**:

```
$ sudo vi /etc/systemd/system/poppy.service
```

Tambahkan *script* berikut pada *file* **poppy.service**:

```
[Unit]
Description=TCP Echo server written in Kotlin
Documentation=https://github.com/telkomdev/Poppy/blob/master/README.md
After=network.target

[Service]
Environment=PORT=9000
Environment=QUEUE_SIZE=100
Type=simple
User=vagrant
ExecStart=/usr/bin/java -jar /home/vagrant/Poppy/target/Poppy-1.0-SNAPSHOT-jar-with-dependencies.jar
Restart=on-failure

[Install]
WantedBy=multi-user.target
```

Kita *breakdown* tiap *section* dari *unit service* yang kita buat.

Pada section **[Unit]** ada *description, documentation,* dan *After*. *Description* dan *documentation* tidak perlu saya jelaskan. *After* ini kita gunakan untuk memberitahu mesin kita ketika *boot up* supaya menjalankan aplikasi atau service kita ketika komponen networknya sudah siap digunakan. Mengingat **Poppy** adalah sebuah **TCP server** yang memerlukan komponen *networking*.

Pada section **[Service]** ada *Environment*, *Type*, *User*, *ExecStart*, dan *Restart*. *Environment* kita gunakan untuk memasukan *environment variable* ke aplikasi atau *service* kita. Pada contoh **Poppy**, kita memasukan 2 *environment variable*, yaitu **PORT** dan **QUEUE_SIZE**. *Type* kita isi simple, karena kita hanya ingin service kita jalan, tanpa membutuhkan pengaturan khusus. Untuk lebih jelas tentang *Type* bisa anda bukan disini https://wiki.archlinux.org/index.php/systemd#Service_types. *ExecStart* digunakan untuk memberitahu systemd bagaimana aplikasi kita akan dijalankan. Untuk contoh **Poppy**, kita membutuhan **Java Virtual Machine** yang berada di **/usr/bin/java**.

Untuk **Python** dan **Nodejs** misalnya, anda bisa menyesuaikannya pada *ExecStart* dengan **/usr/bin/python myscript.py** dan **/usr/bin/node myscript.js**. User digunakan untuk memberitahu systemd supaya aplikasi kita menggunakan *user* khusus dan tidak menggunakan ***user root***. Karena menggunakan ***user root*** memang tidak dianjurkan. *Restart* kita isi dengan on-failure, supaya aplikasi atau *service* yang kita jalankan akan *restart* secara otomatis ketika mati atau gagal berjalan.

Menjalankan service

Ketika ada *unit service* baru, **systemd** tidak akan otomatis menjalankan *service* yang kita buat. Kita perlu memberitahu **systemd** bahwa ada *service* baru.

systemd harus kita *reload* terlebih dahulu.

```
$ sudo systemctl daemon-reload
```

Command di atas juga harus kita jalankan jika ada perubahan pada file service kita.

Kemudian kita jalankan *unit service* Poppy

```
$ sudo systemctl start poppy
```

Selanjutnya kita cek status **Poppy**.

```
$ sudo systemctl status poppy
```

```
vagrant@ubuntu-bionic:~$ sudo systemctl status poppy
● poppy.service - TCP Echo server written in Kotlin
   Loaded: loaded (/etc/systemd/system/poppy.service; disabled; vendor preset: enabled)
   Active: active (running) since Mon 2020-07-20 06:39:25 UTC; 52min ago
     Docs: https://github.com/telkomdev/Poppy/blob/master/README.md
 Main PID: 25850 (java)
    Tasks: 19 (limit: 4702)
   CGroup: /system.slice/poppy.service
           └─25850 /usr/bin/java -jar /home/vagrant/Poppy/target/Poppy-1.0-SNAPSHOT-jar-with-dependencies.jar

Jul 20 06:39:25 ubuntu-bionic systemd[1]: Started TCP Echo server written in Kotlin.
Jul 20 06:39:25 ubuntu-bionic java[25850]: waiting for client connection on port 9000
Jul 20 06:39:47 ubuntu-bionic java[25850]: new client connected /127.0.0.1:39798
Jul 20 06:39:47 ubuntu-bionic java[25850]: waiting for client connection on port 9000
Jul 20 06:39:53 ubuntu-bionic java[25850]: hello
Jul 20 06:39:55 ubuntu-bionic java[25850]: client /127.0.0.1:39798 disconnected
```

Status **Poppy** *active (running)*, menandakan *Server* **Poppy** berjalan dengan baik.

Selanjutnya kita akan *test server* **Poppy** dengan menggunakan **nc**.

`$ nc localhost 9000` ganti **localhost** dengan **IP** jika diakses dari komputer lain.

```
vagrant@ubuntu-bionic:~/Poppy/target$ nc localhost 9000
12345
unauthenticated
authenticated
Hello
OK
Poppy
OK
```

Menghentikan *service* **Poppy** dengan *command stop*.

```
$ sudo systemctl stop poppy
```

Kemudian cek status **Poppy**

```
$ sudo systemctl status poppy
```

```
● poppy.service - TCP Echo server written in Kotlin
   Loaded: loaded (/etc/systemd/system/poppy.service; disabled; vendor preset: enabled)
   Active: failed (Result: exit-code) since Mon 2020-07-20 07:43:45 UTC; 3s ago
     Docs: https://github.com/telkomdev/Poppy/blob/master/README.md
  Process: 25850 ExecStart=/usr/bin/java -jar /home/vagrant/Poppy/target/Poppy-1.0-SNAPSHOT-jar-with-dependencies.jar (code=exited, status=143)
 Main PID: 25850 (code=exited, status=143)

Jul 20 06:39:55 ubuntu-bionic java[25850]: client /127.0.0.1:39798 disconnected
Jul 20 07:32:30 ubuntu-bionic java[25850]: new client connected /127.0.0.1:39800
Jul 20 07:32:30 ubuntu-bionic java[25850]: waiting for client connection on port 9000
Jul 20 07:32:48 ubuntu-bionic java[25850]: Hello
Jul 20 07:32:52 ubuntu-bionic java[25850]: Poppy
Jul 20 07:43:38 ubuntu-bionic java[25850]: client /127.0.0.1:39800 disconnected
Jul 20 07:43:44 ubuntu-bionic systemd[1]: Stopping TCP Echo server written in Kotlin...
Jul 20 07:43:45 ubuntu-bionic systemd[1]: poppy.service: Main process exited, code=exited, status=143/n/a
Jul 20 07:43:45 ubuntu-bionic systemd[1]: poppy.service: Failed with result 'exit-code'.
```

Terima kasih

Hari ini segitu dulu, besok-besok lagi :D

Semoga bermanfaat...