Ch07. Node Exporter (c)

Quick Summary

- Node Expoter는 제외할 수집을 다양한 Option으로 지정하여 수행할 수 있다.
- Node Exporter는 root권한이 아닌 채로, 수집을 하고 있음에 관심을 가져라.
- sysstat(mpstat, iostat, vmstat)와 /proc의 내용들이 Node Exporter의 주요 원천인 것은 당연하다.
- Metrics Sample & Key Metrics

```
avg without(cpu)(rate(node_cpu_seconds_total[1m]))
avg without(cpu)(rate(node_cpu_guest_seconds_total[1m]))
node_filesystem_size_bytes
node_filesystem_avail_bytes
node_filesystem_files~ /* inode */
node_disk_~
node_network_receive_bytes_total
node_network_transmit_bytes_total
node_forks_total
node_context_switches_total
node_procs_blocked
node_procs_running
node_uname_info /* info type 이다. */
... more, more, more ...
```

textfile 수집기

Node Exporter의 Metric대부분이 Kernel에서 얻어지는 반면, 이것은 우리가 만든 파일에서 가져온다. 이런 방법의 구성 개념과 일반적인 사용패턴은 다음과 같다.

- smartctl, iptables 같은 명령어(대부분 기존 Meric에 없거나, root가 필요한)를 cronjob으로 만들고, 거기에서 생성되는 출력을 프로메테우스 형식의 문서로 변환하여 특정 디렉터리에 기록한다. (파일을 생성할 때는 automic write가 필요하다)
- Node exporter는 데이터를 수집할 때, 해당 디렉터리의 파일을 일고, 그 결과를 Metric에 포함할 것이다. 이렇게 프로메테우스에게 제공된다.
- 예를 들어 Casandra node가 백업 같은 Batch Job을 처리하고 그 결과를 기록해 놓으면, 프로메테우스에서 그 기록을 Metric처럼 사용할 수 있다. textfile을 사용하는 것은 Kernel의 정보를 제공하는 OS를 위한 Node exporter가 아니라, 마치 Node Exporter를 Application을 위한 Exporter 처럼 사용하는 것과 유사하게 보일 수 있다.

간단하게 예제로 textfile 을 만들어 본다.

- 우선 Node exporter를 실행하기전에 textfile을 사용하기 위한 디렉터리를 만들고, 거기에 Metric(아래 예제에서는 `shadow_entries`)으로 사용할 textfile을 미리 만들어 두고, 그 이후에 Node exporter를 실행한다.

```
$ pwd
$ /svc
$ mkdir textfile_dir
$ echo "shadow_entries 1" > /svc/textfile_dir/example.prom
$ cd $NODE_EXPORTER_HOME
$ node_exporter --collector.textfile.directory="/svc/textfile_dir"
```

    - 파일명은 `.prom` 이라는 확장자로 파일을 만들어야 하며, 미리 만드는 이유는 Node exporter가 시작시 부터 해당 Metric이 있어야 Metric에 대한 기록과 처리가 부분적으로 처리되지 않기 때문이다.
    - 파일 내부에 Metric이름은 앞의 명명규칙에 따라야 하며, `#HELP`, `#TYPE` 은 없으면, Node exporter가 대충 만들어서 붙여주게 된다. 만약 동일 Metric을 여러 textfile에 넣는 경우, 동일한 `#HELP` 를 제공해야 한다.
- 다음으로 cronjob을 만든다. cronjob의 내용은 임시 파일을 만들어 기록할 Metric 내용을 우선 file write처리 하고, 최종 원하는 파일명으로 바꾸어 준다. (이렇게 하는 이유는 파일 자체에 대해 Automic상태가 유지되도록 하기 위함이며, 참고로, 동일 Volume의 filesystem내에서만 가능하다.)

```
$ more /etc/crontab
...
TEXTFILE=/svc/textfile_dir

# 아래 내용은 cron file 특성한 한 줄에 작성되어야 한다.
* * * * * root ( echo -n 'shadow_entries '; grep -c . /etc/shadow )
          > $TEXTFILE/example.prom.$$ &&
          mv $TEXTFILE/example.prom.$$ $TEXTFILE/example.prom
$
```

    - shell script에서 `&&` 은 앞의 명령이 성공하면 그 다음 수행하는 것이고, `$$` 는 current process의 `pid` 이다.
- 더 많은 textfile 수집기 예제는 다음을 참고한다. >

textfile관련된 Metric에 대한 timestamp가 필요한 경우, textfile 수집기에서 얻을 수 있는 timestamp의 사용은 권장되지 않는다. 차라리, 앞의 예제에서 처럼, 취급한 파일의 unix file system의 mtime을 사용하는 것이 권장된다. (대충 뭔말인지는 알겠는데, 책에서는 그 이유, 원인의 설명이 한국어가 아님. ㅋㅋ…따라서 시간날때 더 찾아 볼 기회가 있기를….)

```
# HELP node_textfile_mtime_seconds Unixtime mtime of textfiles successfully modify.
# TYPE node_textfile_mtime_seconds guage
node_textfile_mtime_seconds{file="example.prom"} 1.516205651e+09
```

(opt) `systemd` 기반에서 OS가 제공하는 node_exporter사용하기

- 누가 제공하는 node_exporter를 사용할 것인가?
    - Prometheus에서 제공하는 Linux를 위한 node_exporter는 Ubuntu환경에서 몇몇 Metric에서 문제가 있다. 아마도 다른 OS에서도 문제가 있을 수 있다. 그렇지만, Ubuntu도 그렇지만, OS 제공자들이 Prometheus를 위한 자신들에게 맞는 node_exporter를 별도로 제공한다. Ubuntu에서는 그것을 사용해 볼 수 있다.
- Ubuntu에서 node_exporter의 설치하고, 가동하기
    - 간단하게 apt를 사용하여 설치 한다.

```
$ sudo apt udpate
$ sudo apt upgrade
$ sudo apt install prometheus-node-exporter
```

    - 다음을 참고하여, 해당 서비스를 관리하고, 시작과 정지를 할 수 있다.

```
// enable, disable Service
$ sudo systemctl enable prometheus-node-exporter

// start, stop, status Service
$ sudo systemctl start prometheus-node-exporter
```

- Linux에서의 `systemd` 는 무엇인가? 말하자면 길어지지만…OS의 기본 Daemon관리자
  - 간단히 말하자면, Linux에는 SystemV부터 사용 되어 온 `init.d`와 cloud환경을 지원하는 immutable개념의 `systemd`가 존재한다. 2010년 초반, Cloud환경(특히, docker, k8s가 강조되면서)부터, Redhat, Ubuntu, CentOS 들 모두 OS의 기본 Daemon관리자는 `init.d`가 아니라, `systemd`가 되었다.
  - `init.d`는 부팅시에 절차적으로 Daemon들을 기동하고, 운영중에도 개별적으로 쉽게 통제는 기능을 제공한다. 해당 기능은 Shell Script로 작성되어 있다. (`service`라는 명령으로 사용)
  - `init.d`는 점점 단점이 많았졌는데, 주요 단점들은 다음과 같다: 부팅시 Daemon가동이 순차진행방식의 느림, 부팅 flow lock가능성, 각 Demon의 enable/disable 및 운용의 명령 비표준, 설정이나 환경변수의 수정시 해당 Shell Script의 직접 수정, Daemon들의 상태감시 및 Retry기능은 별도로 구현 필요
  - `systemd`는 기존 `init.d`의 역할을 완벽히 대체하며, `init.d`의 단점을 보완하고, Cloud환경에 적합한 방식으로 개선되었다. `systemd`는 `systemctl` 이라는 명령으로 사용한다.
  - `systemd`의 특징과 개선된 장점은 다음과 같다.
    - `systemd` 방식은 해당 기능에 대해 immutable하게 정의와 정보를 구성하고, `systemd`가 직접 가동을 하는 구조이다. 따라서, 더 가볍게 부팅과 가동이 상대적으로 빠르며, 필요시 Flow lock없이 동시 가동도 가능하다.
    - 설정 방식과 운용방식이 표준화되었으며, 수정이 필요할 경우 해당 설정을 변경하고 반영하는 개념이다. (설정이 변경되면, `systemctl daemon-reload` 명령이 필요한데, 이유는 분명하다)
    - `systemd`의 모든 것들은 모두 `systemctl`명령으로 모두 가능하고, immutable을 지향하므로 API처럼 사용할 수 있고, 분산환경 ㅁ및 cloud환경에 적합한 방법이다.
- 단일 OS내에서 `systemd`를 위한 환경은 어떠 한가?
  - 설정 파일 디렉터리들. 보통은 수정하거나 조작할 일은 별로 없다.

```
// 설정 파일 디렉터리들. 보통은 수정하거나 할 일은 별로 없다.
$ /etc/systemd
$ /etc/systemd/system    // 대부분은 여기에 있다.
$ /etc/systemd/system/multi-user.target.wants    // multi-user level의 것들은 여기에 있다.
$ /etc/systemd/user
```

  - 프로메테우스 node_exporter Service Daemon에 대한 내용은 다음과 같다. 설정 정보만이 있다.

```
$ more /etc/systemd/system/multi-user.target.wants/prometheus-node-exporter.service
[Unit]
Description=Prometheus exporter for UBUNTU  metrics
Documentation=https://github.com/prometheus/node_exporter

[Service]
Restart=on-failure
User=prometheus
EnvironmentFile=/etc/default/prometheus-node-exporter
ExecStart=/usr/bin/prometheus-node-exporter $ARGS
ExecReload=/bin/kill -HUP $MAINPID
TimeoutStopSec=20s
SendSIGKILL=no

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

  - 프로메테우스 node_exporter관련된 추가 설정 정보는 다음과 같다. 과거 Shell Script기반의 init.d와 차이점이 무엇인지 짐작이 된다.

```
$ more /etc/default/prometheus-node-exporter
# Set the command-line arguments to pass to the server.
# Due to shell scaping, to pass backslashes for regexes, you need to double
# them (\\d for \d). If running under systemd, you need to double them again
```

```
# (\\\\d to mean \d), and escape newlines too.
ARGS="--collector.textfile.directory=\"/svc/textfile_dir\""
$
```

- 마지막으로 `systemd`를 운용하는 `systemctl` 명령어의 사용 예시이다.

```
$ sudo systemctl list-units
$ sudo systemctl list-unit-files
$ sudo systemctl list-unit-files | grep cron
$ sudo systemctl list-units --state=enabled
$ sudo systemctl list-units --state=active
$ sudo systemctl list-units --all --state=inactive
$ sudo systemctl list-units --type=service --state=running

$ sudo systemctl is-active nginx    // nginx가 가동상태인지 확인
$ sudo systemctl start nginx

$ sudo systemctl is-enabled nginx   // nginx가 부팅시 구동되는 상태인지 확인
$ sudo systemctl enable nginx
$ sudo systemctl disable nginx
$ sudo systemctl daemon-reload      // 메모리기반 운용과 설정들의 안전장치 이다.

$ sudo systemctl mask ntpd       // ntpd를 symbolic link처리로 숨기는 방식으로 사용되지 않도록 함
$ sudo systemctl unmask ntpd
```

+==