

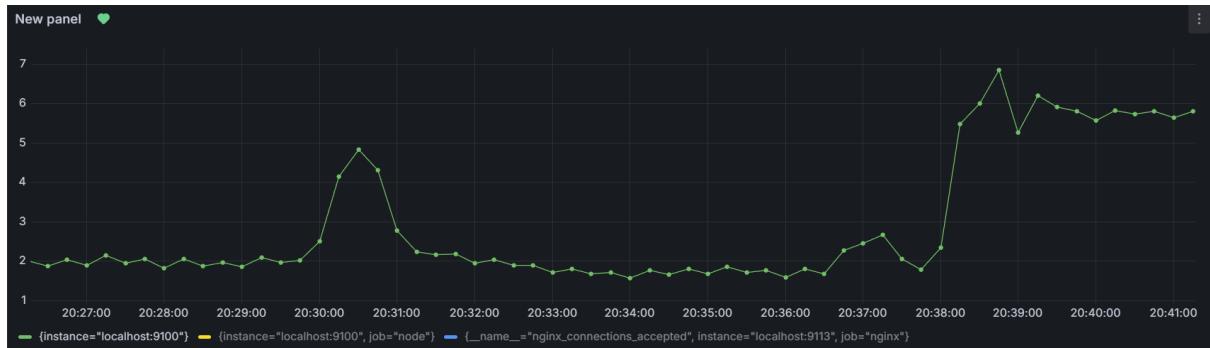
# Cloud computing assignment 01

2023320302

데이터과학과

이유찬

## 1. Dashboard screenshots/query/explanations

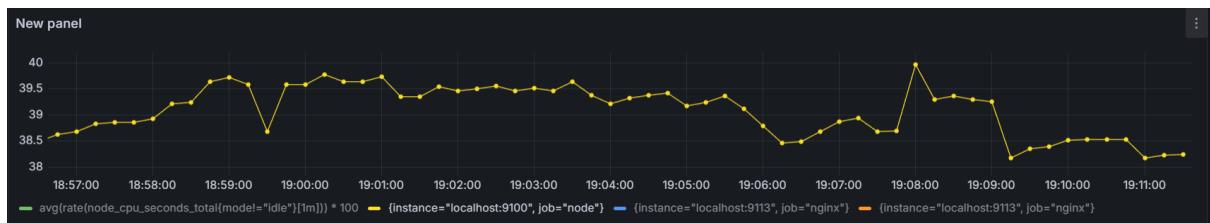


### CPU usage

```
100 * sum by (instance) (rate(node_cpu_seconds_total{mode!="idle"}[1m])) / sum by (instance) (rate(node_cpu_seconds_total[1m]))
```

Gets the CPU's non-idle time, then divide it with total time passed for all cores for 1 minute window.

Then multiply by 100 to get percentage.



### Memory usage

```
(1 - (node_memory_MemAvailable_bytes / node_memory_MemTotal_bytes)) * 100
```

Subtracts the ratio of (available memory / total memory) (= unused memory) from 1, to get used memory ratio.

Then multiply by 100 to get percentage.



**Nginx accepted connections (blue)**

**nginx\_connections\_accepted**

**Nginx HTTP Requests (Orange)**

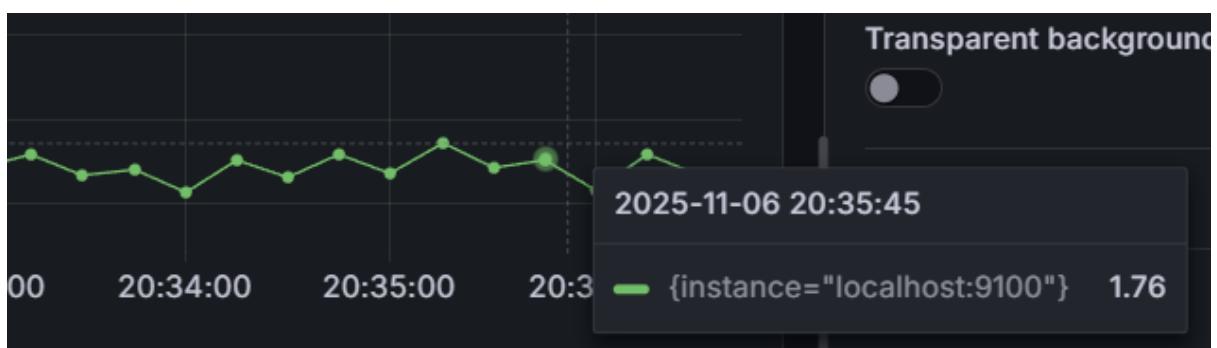
**nginx\_http\_requests\_total**

Both used single metric that is scraped from nginx exporter.

(Tried using delta operation to get differences, but didn't matter that much.)

## 2. ApacheBench load testing

**Small load**

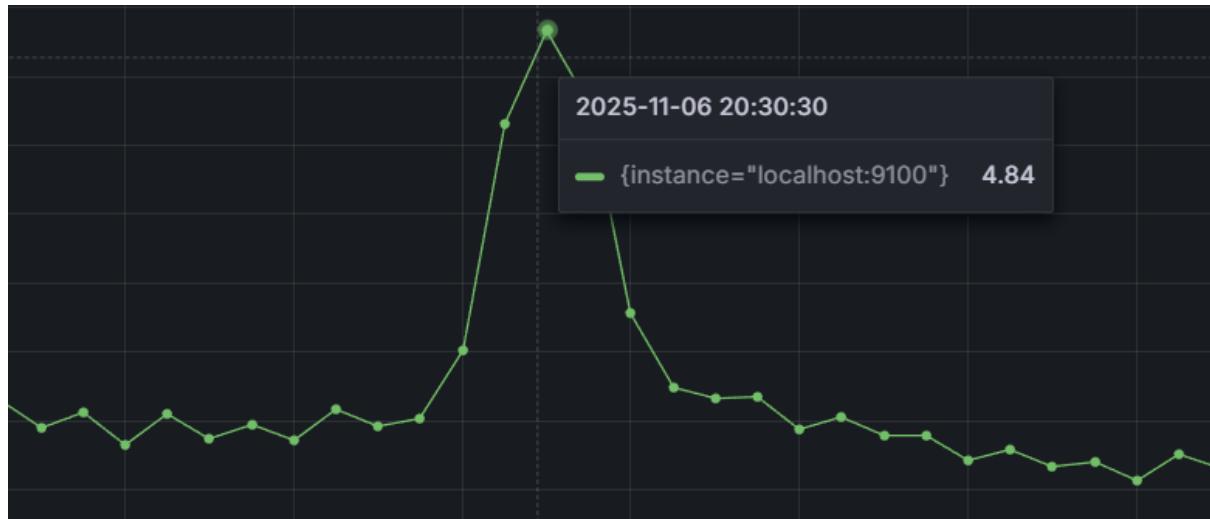


CPU usage almost had no spike, moving around 1.8%.

Memory usage had small spikes, moving around 38%.

Both accepted connections and requests were growing linearly, as more and more requests were made.

### Medium load

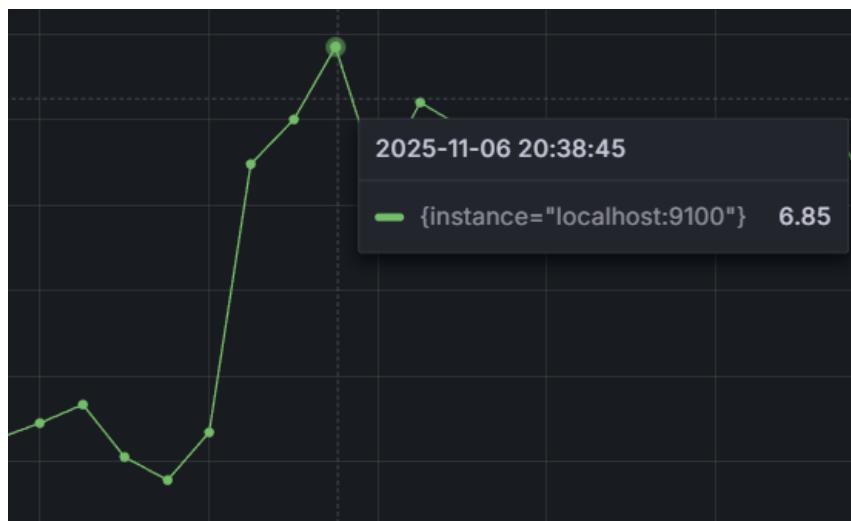


CPU usage had a spike, up to 4.84%.

Memory usage also had small spikes, moving around 39%.

Both accepted connections and requests were growing linearly, as more and more requests were made.

### Large load



CPU usage had a big spike, up to 6.85%.

Memory usage also had spikes, moving around 41%.

Both accepted connections and requests were growing linearly, as more and more requests were made.

For all of them, 'Nginx HTTP requests' were slightly bigger than 'Nginx accepted connections'.

### 3. Alerts

Query: same with above's memory query

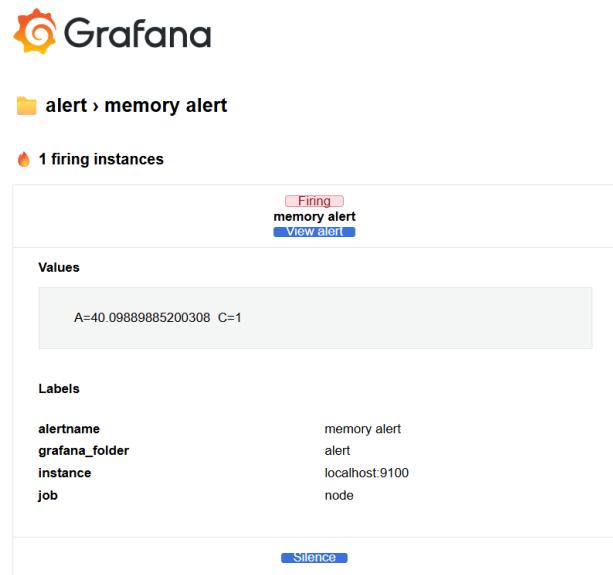
The screenshot shows the 'New alert rule' configuration page in Grafana. It consists of two main sections: '1. Enter alert rule name' and '2. Define query and alert condition'.  
Section 1: 'Enter alert rule name'  
- Sub-section: '1. Enter alert rule name'  
- Description: 'Enter a name to identify your alert rule.'  
- Input field: 'Name' containing 'memory alert'.  
Section 2: 'Define query and alert condition'  
- Sub-section: '2. Define query and alert condition'  
- Description: 'Define query and alert condition [Need help?](#)'  
- Advanced options toggle: Off.  
- Prometheus query builder:  
 - Target: 'prometheus'  
 - Time range: 'Options' set to '10 minutes'  
 - Query: '(1 - (node\_memory\_MemAvailable\_bytes / node\_memory\_MemTotal\_bytes)) \* 100'  
 - Buttons: 'Run queries', 'Builder', 'Code'  
- Alert condition:  
 - WHEN QUERY: 'IS ABOVE'  
 - Value: '40'  
 - Preview button: 'Preview alert rule condition'

If memory usage is over 40%, fire alert.

I used apache bench (ab -n 100000 -c 1000) to make it fire.

The alert:

☆✉️ [받은메일함] [FIRING:1] memory alert alert (localhost:9100 node) ↗  
2025. 11. 6. (목) 20:08  
✓ 보낸사람 ⓘ Grafana<udyann@korea.ac.kr>  
받는사람 <udyann@korea.ac.kr>



The dashboard:

