Nama : Fiodhy Ardito Narawangsa

NRP : 05111940000218

Kelas : Progjar D

Tugas 2 Individu

Progjar5 merupakan implementasi sederhana dari protokol HTTP 1.0 (RFC 2616). Metode yang diimplementasikan adalah GET dan POST

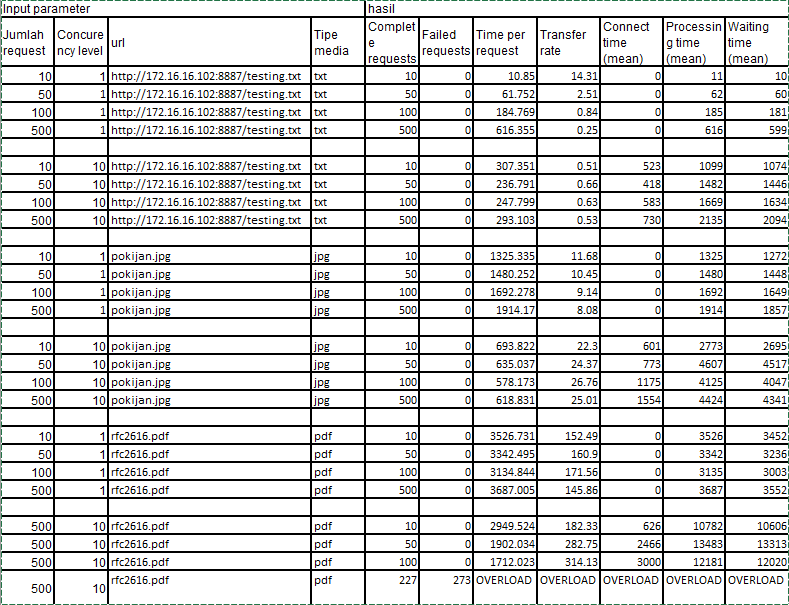
* File http.py merupakan implementasi http server
* File server\_thread\_http.py merupakan implementasi servernya, dijalankan di port 8889

Pendahuluan:

* Jalankan environment lab ([progjar/environment at master · rm77/progjar (github.com)](https://github.com/rm77/progjar/tree/master/environment))
* Jalankan server\_thread.py pada server 16.101
* Dengan menggunakan browser akseslah alamat
  + <http://172.16.16.101:8889/>
  + <http://172.16.16.101:8889/rfc2616.pdf>
  + [http://172.16.16.101:8889/](http://172.16.16.101:8889/rfc2616.pdf)testing.txt
  + [http://172.16.16.101:8889/](http://172.16.16.101:8889/rfc2616.pdf)pokijan.jpg

* Install apache-benchmark (ab) di console shell **172.16.16.102**
  + #sudo bash
  + #apt update -y && apt install -y apache2-utils
  + Jalankan perintah ab
    - Ab merupakan apache-benchmark, tool untuk melakukan performance test pada web server

Tugas

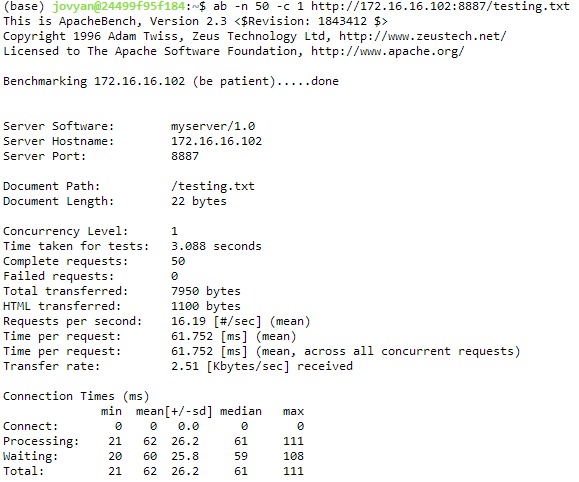
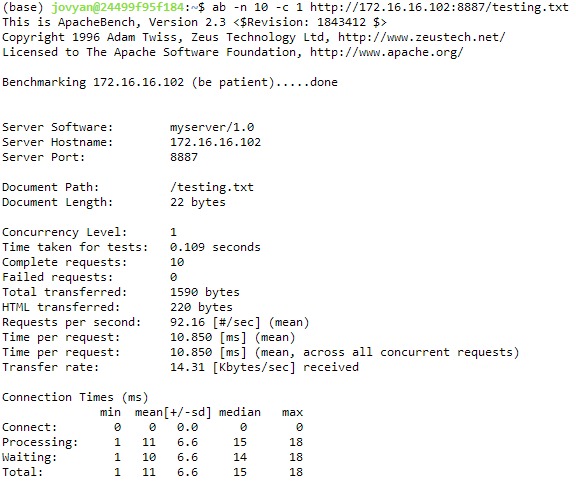


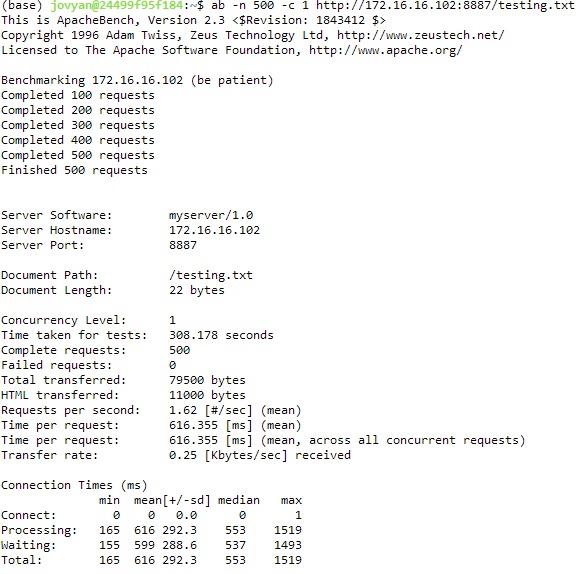
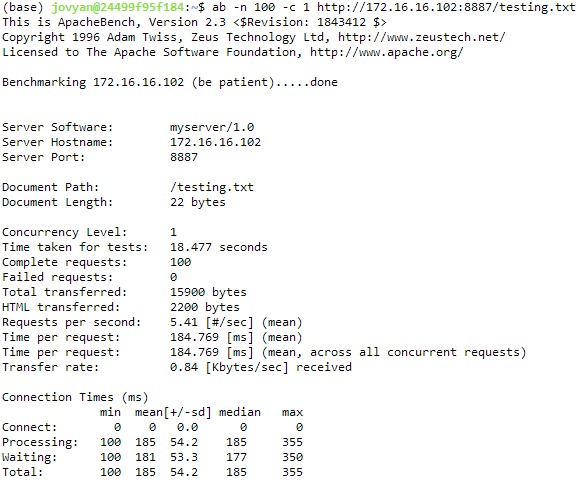
Dari tabel metrik diatas, terlihat bahwa semakin besar jumlah request maka semakin besar pula total time nya. Hal tersebut juga dapat dioptimasi dengan menambah jumlah konkurensi. Pada jumlah request dibawah 100, dengan menaikkan jumlah konkurensi, maka dapat memperkecil total time. Namun pada jumlah request diatas 100, menaikkan jumlah konkurensi tidak dapat menghasilkan perubahan yang berarti. Tipe media tidak berpengaruh pada request time, namun panjang dari data itu lah yang mempengaruhi total time. Namun pada kasus nyata, tipe txt biasanya memiliki panjang data yang lebih kecil daripada jpg, begitu juga jpg lebih kecil daripada pdf. Sehingga pada pengukuran diatas rata-rata total time menghasilkan txt yang paling cepat, disusul jpg, lalu pdf

**Solusi Overload**

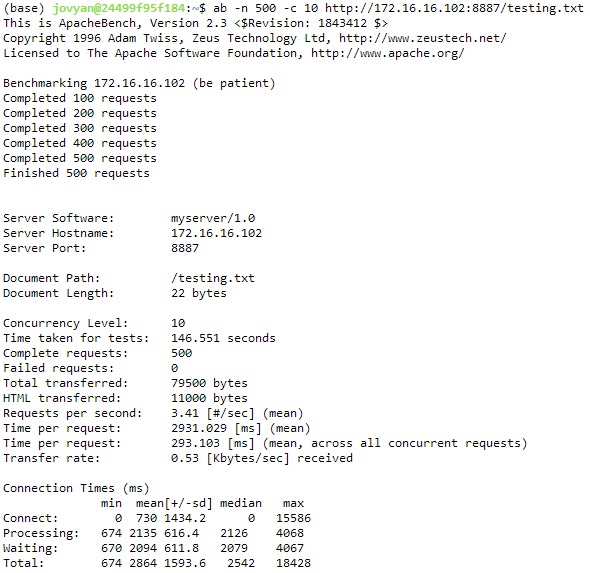
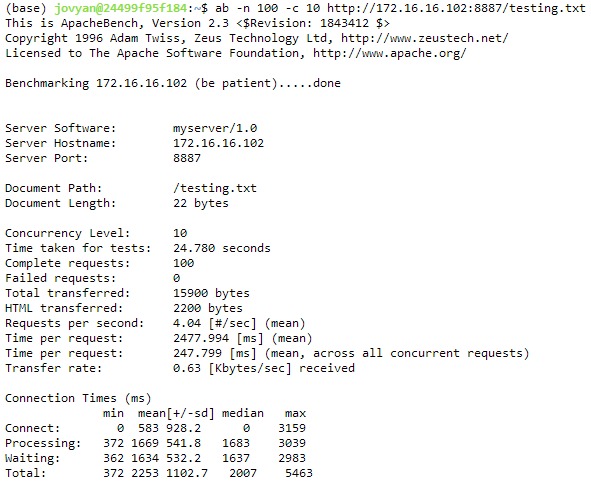
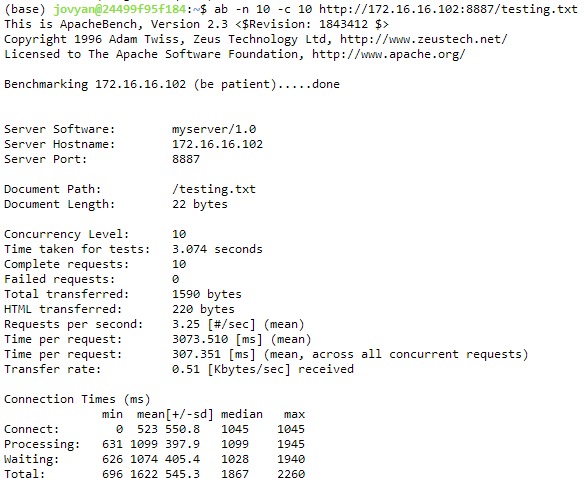
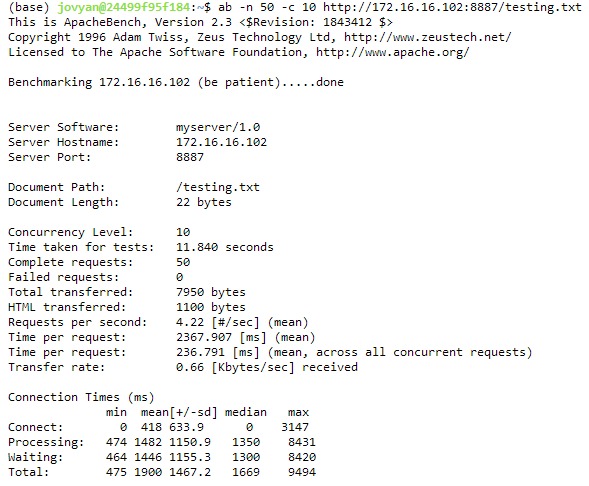
Untuk mengatasi OVERLOAD adalah dengan mendisable logging pada server. Sehingga penggunaan resources pada server akan berkurang dan akan lebih berfokus untuk melayani request dari client. Pada kasus ini logging menghambat waktu proses pada server, atau lebih sering dikenal dengan I/O Bottleneck.

**File tipe txt dengan Konkurensi = 1**

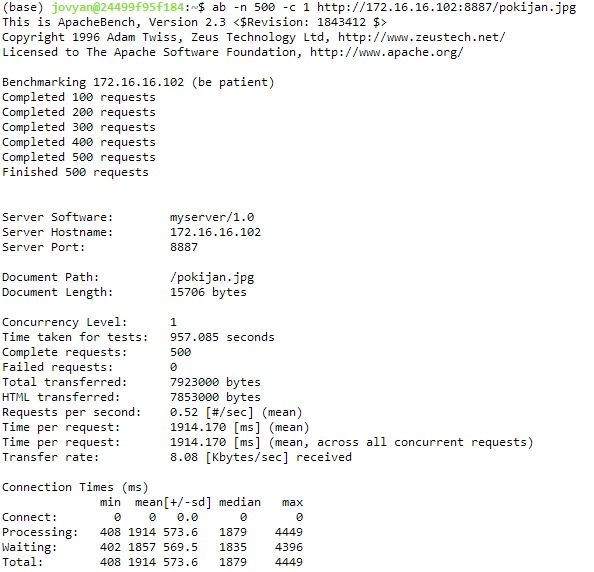
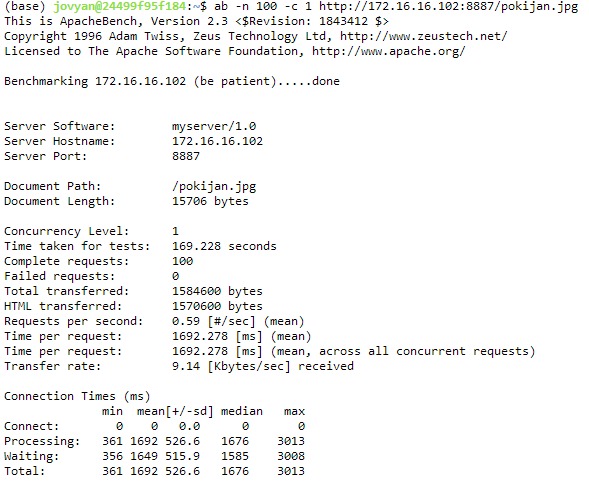
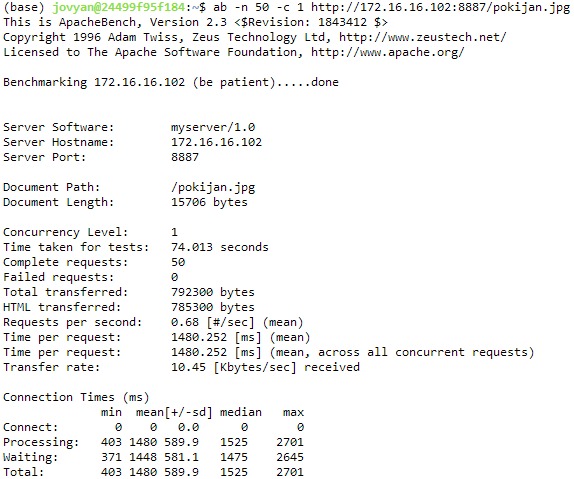
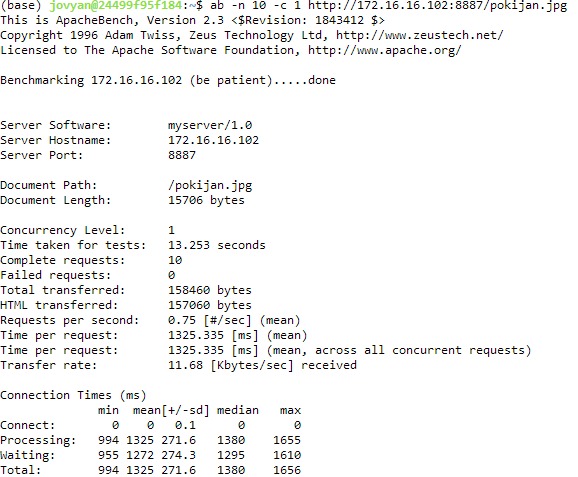
****

****

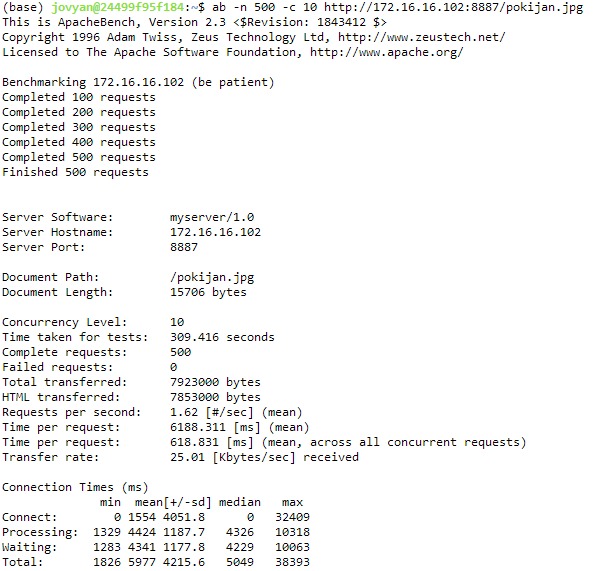
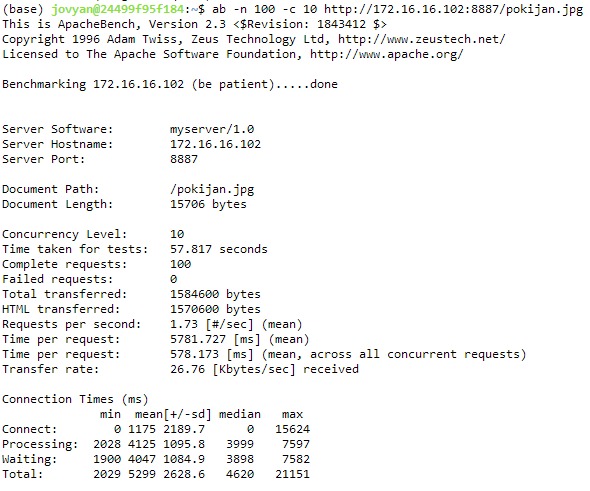
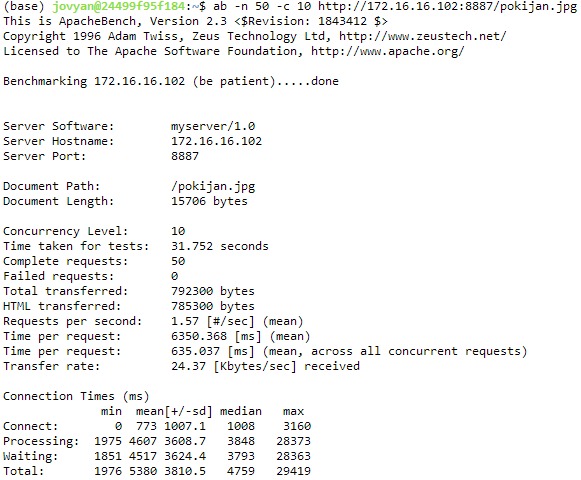
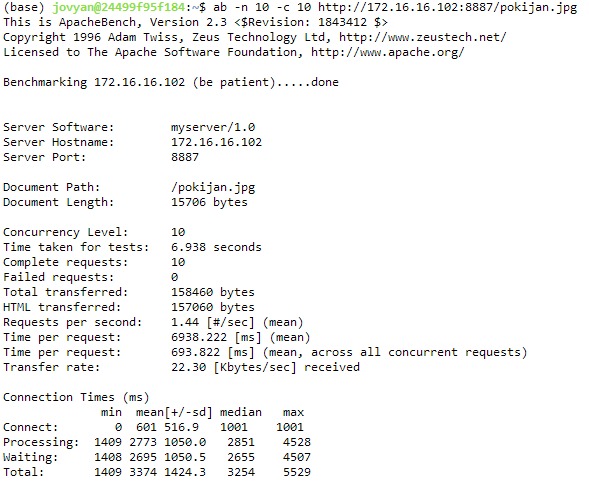
**File tipe txt dengan konkurensi = 10**

****

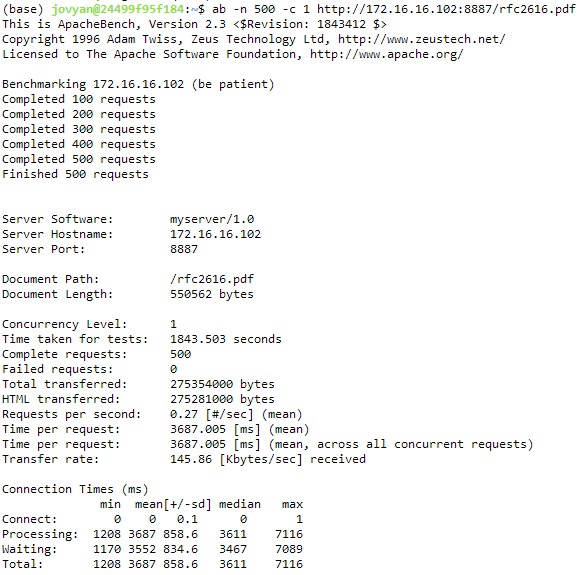
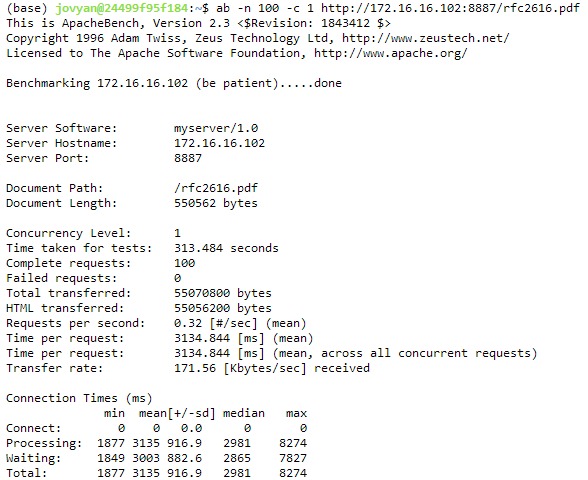
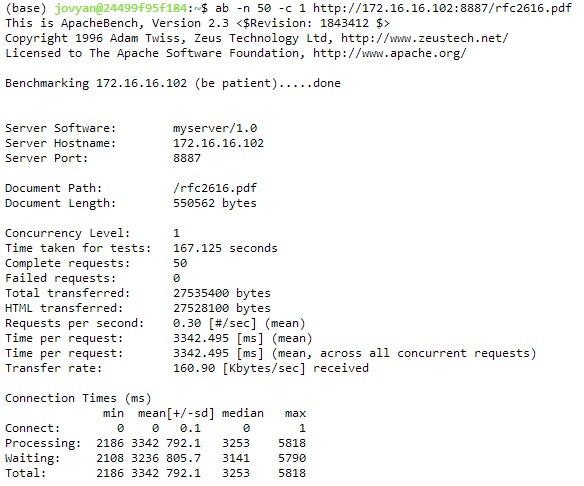
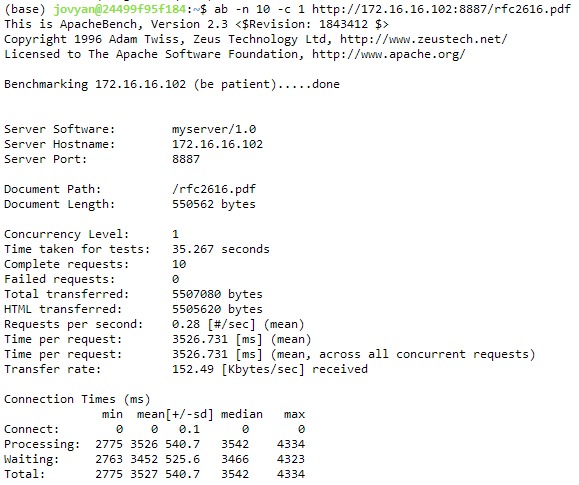
**File tipe jpg dengan konkurensi = 1**

****

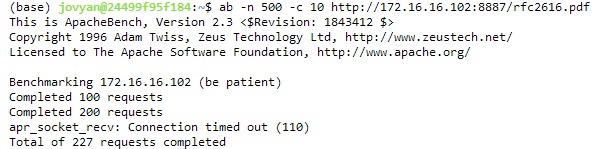
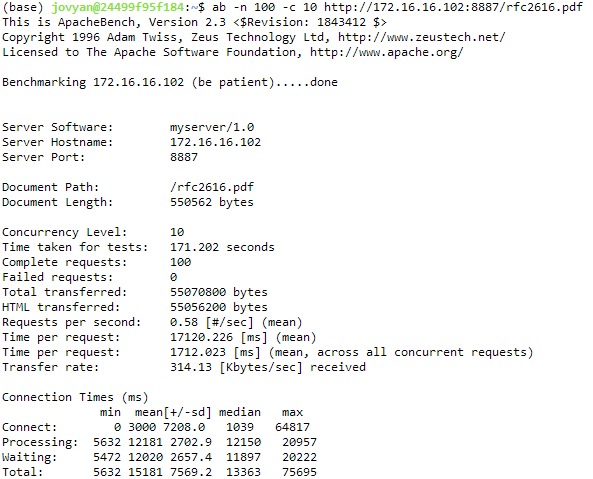
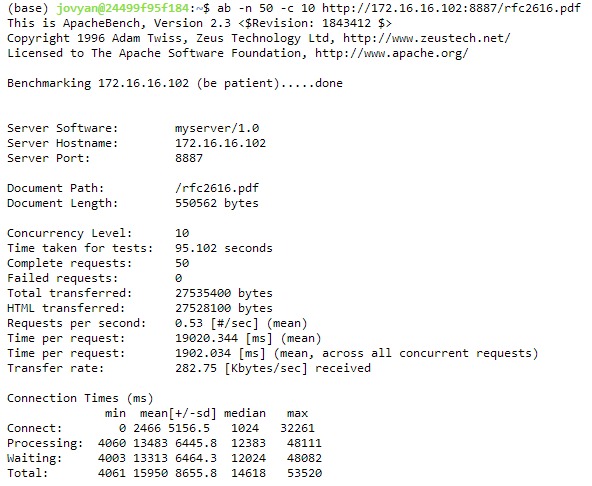
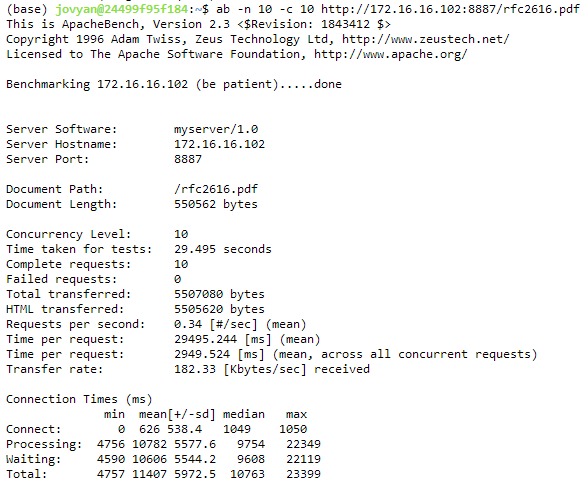
**File tipe jpg dengan konkurensi = 10**

****

**File tipe pdf dengan konkurensi = 1**

****

**File tipe pdf dengan konkurensi = 10**

****