# RandPwGen Service





* CPU usage (blue) rises gradually, peaks, then falls.
* Latency (red) follows a smoother, less spiky curve compared to the hash generator.
* Unlike hash generation, latency here increases proportionally with load, not abruptly.
* This indicates the password generator is likely less CPU-intensive per request and can handle small bursts without immediately saturating CPU. Latency degradation is more graceful, suggesting:
* Lightweight password generation logic (random sampling)
* Efficient concurrency (e.g., async)
* No blocking computation (unlike hashing)
* No correlation between memory and latency (unlike the hash generator).
* Latency is **purely CPU-bound** here, not memory-bound. You’re not dealing with:
* GC pressure
* Fragmentation
* Memory-related queuing
* Like the hash generator, latency drops fast after CPU usage declines.
* Indicates:
  + No queued backlog
  + Stateless request handling
  + Possibly short-lived connections