Metrics mit Prometheus

Übung 10

Setup

- https://prometheus.io/docs/prometheus/latest/getting_started/
- Prometheus läuft im Docker
  - Siehe auch prometheus.yaml → default config + geänderte targets
- Project Metrics
  - 127.0.0.1:8080 → von dort **pulled** prometheus die Metriken
  - NuGet: prometheus-net
  - http://127.0.0.1:8080/metrics zeigt was geholt wird
- Unter http://localhost:9090/targets sieht man ob die .NET Applikation erreicht werden kann
- Unter http://localhost:9090/graph den Counter („sampleapp_ticks_total“) eingeben

Setup

- Grafana ist unter http://localhost:3000 erreichbar
- Login:
  - User: admin
  - Passwort: admin

Aufgabe 1

- Erzeugen von 30 Threads
- Endlosschleife in jedem Thread:
  - Jeder Thread macht eine 1 Sekunden Aufgabe `(Thread.Sleep(TimeSpan.FromSeconds(1)))`
  - Dann schläft er randomisiert 0-10 Sekunden `(Thread.Sleep(TimeSpan.FromSeconds(X)))`
- Erzeugen eines Graphs (Dashboard) in Grafana: Wieviel Aufgaben sind zum Zeitpunkt X immer aktiv?
- Hint:
  - `Prometheus.Metrics.CreateGauge`
    - Name: `tasks_active`
  - `using(X.TrackInProgress())`
- **Screenshot von Grafana in die Abgabe (selber Ordner wie *.sln File)**

Aufgabe 2: Messen von Request-Dauer

- Erstellen eines **Histogramm** „create_stock_duration_seconds“ (mit 10 Buckets in 0.1 sec Abständen → siehe HistogramConfiguration Param)
- Wieder 30 Threads in Endlosschleife
- Pro Thread: Befüllen mit Random Werten
  - `using (LoginDuration.NewTimer())` mit
  - `Thread.Sleep(random.Next(300, 1000));`
- Was sieht man im Graph?
  - http://localhost:9090/graph
- Erstellen eine Heatmap (Dashboard) in Grafana → Screenshot in Abgabe
  - Metrics: das _bucket
  - Rechtes > Axes > Data Format > Format > Time Series buckets
- Was sehen Sie? (Fragen.txt + Abgabe Screenshot)