Recitation 10

Monitoring

Monitoring

- Responsibility of a development team doesn't end when their code is deployed to production
- **Use cases** (after deployment)
  - Track how our software is performing in production
  - Make decisions based on live metrics
  - Quickly inform developers / operations team of undesirable situations
  - Quickly respond to situations in production

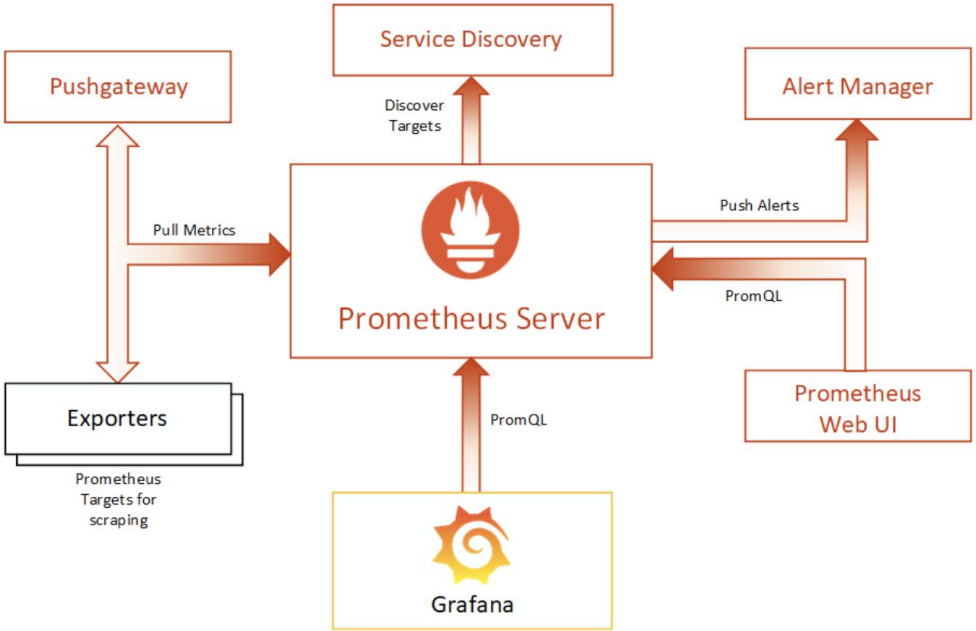
- **Goal for the Recitation**
  - Give an introduction to Prometheus and Grafana
  - Save some time for you in setting up the monitoring stack, and linking your application to it
  - Hopefully, the demo / starter code will help you progress faster

Prometheus + Grafana

- **Prometheus**
  - A time series database that stores metrics from your application
  - It has client libraries that let you create and expose metrics
  - Metrics can be pulled by Prometheus from your app, or.. Your app can push it to Prometheus

- **Grafana**
  - A visualization tool (dashboards - charts, etc.)
  - Where to get the data to visualize - data sources (Prometheus, PostgreSQL, etc.)
  - What do we want to visualize - write PromQL queries to configure

Prometheus + Grafana



Source: https://neilkillen.com/2020/05/30/monitoring-sitecore-container-environment-with-prometheus/

Demo

- GitHub repo: https://github.com/vaithya94/Monitoring_Demo
- Sample app
  - Running as a Docker container
  - Exposes some API endpoints
  - Exports metrics using Prometheus client (Pull model)
- Run Prometheus and Grafana as containers
- Access Prometheus UI
- Build a sample dashboard in Grafana by connecting to Prometheus

Links:

- Prometheus client for Python - https://github.com/prometheus/client_python
- Prometheus configuration - https://prometheus.io/docs/prometheus/latest/configuration
- Prometheus & Grafana integration - https://prometheus.io/docs/visualization/grafana/
- https://neilkillen.com/2020/05/30/monitoring-sitecore-container-environment-with-prometheus/

Push vs Pull

- **Pull**
  - Expose metrics from your application via an API, and let Prometheus poll it
  - Typically used when your application is going to be is a long running process

- **Push**
  - Push metrics from your application / CI tool to a “push gateway” app
  - You need to run push gateway as a separate container in your VM
  - Prometheus will poll the push gateway to fetch metrics
  - Typically used when you have
    - a short-lived process
    - when the metrics aren’t gonna change that often

Common Metric Types

- **Counter** - Values that increase, or get reset to zero; a cumulative metric; rate of increase is valid
  - Example: Number of incoming HTTP requests (grouped by status codes)
  - Example: It's not appropriate to use a counter for number of running containers

- **Gauge** - Single numerical value that can arbitrarily increase or decrease; represents a state
  - Example: Model quality, Telemetry metric value, etc.

More on metric types

- https://prometheus.io/docs/concepts/metric_types/
- https://tomgregory.com/the-four-types-of-prometheus-metrics/

PromQL - Examples

- request_count_total{http_status="500", method="GET"}
  - Shows us the count of requests that have the status 500

- rate(request_latency_seconds_count[1h])
  - Shows us the request latency over time

- More resources
  - https://prometheus.io/docs/prometheus/latest/querying/basics/
  - https://prometheus.io/docs/prometheus/latest/querying/examples/
  - https://prometheus.io/docs/prometheus/latest/querying/functions/

Thank You!