Infrastructure & System Monitoring

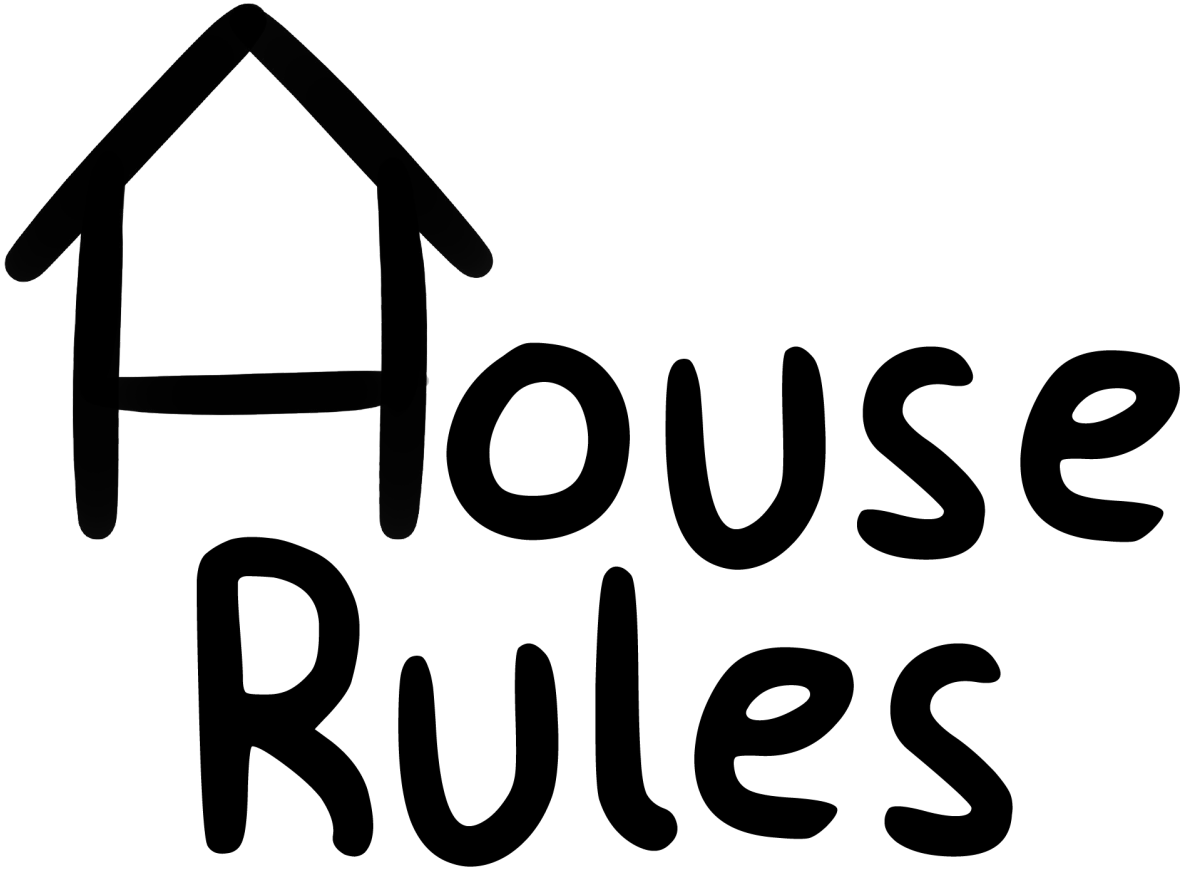
using Prometheus

Marco Pas

@marcopas

Goal

Learn how to monitor your infrastructure, systems, and applications.


House
Rules

Agenda

- Monitoring
  - Introducing you to a Scary Movie
- Prometheus overview (demo’s)
  - Running Prometheus
  - Gathering host metrics
  - Introducing Grafana
  - Monitoring Docker containers
  - Alerting
  - Instrumenting your own code
  - Service Discovery (Consul) integration

..Quick Inventory..

I am going to introduce
you to some bad movies


BEST
WORST
MOVIE
GOBLIN


NYFD


JURASSIC SHARK

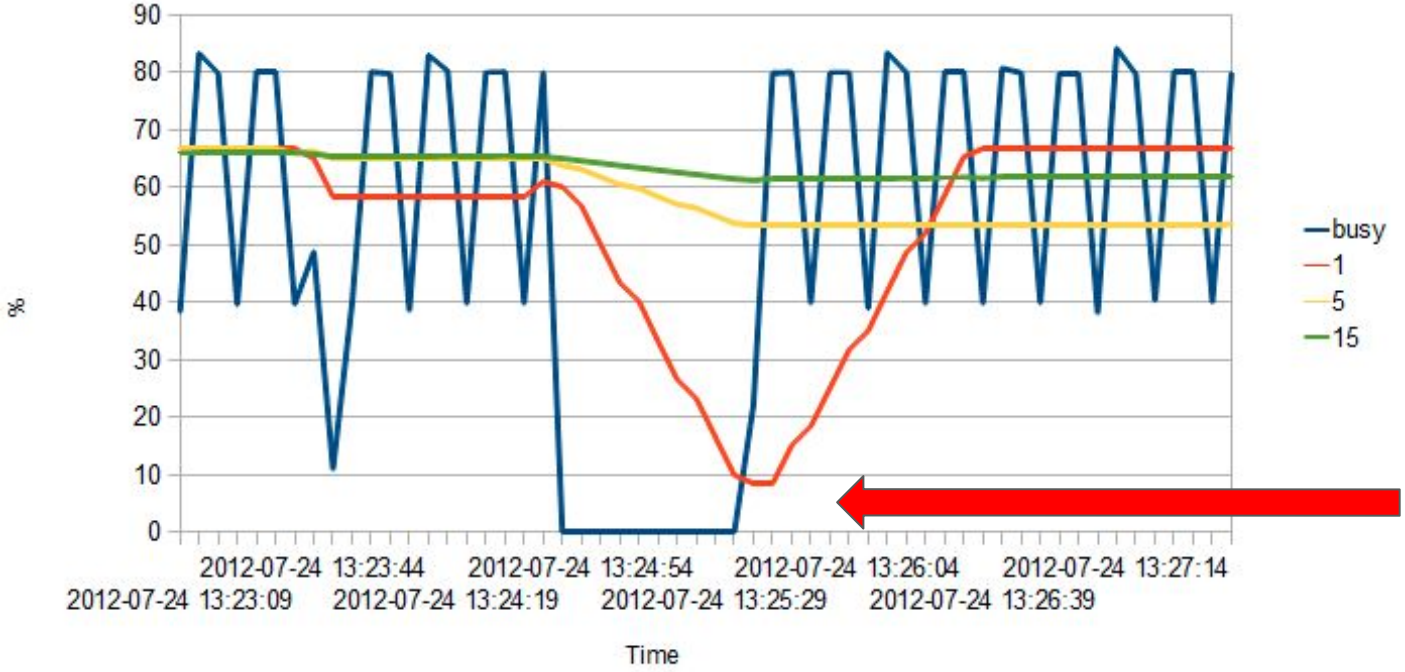

Commonality
between
these movies?

Monitoring

Our scary movie “The Happy Developer”

- Let's push out features
- I can demo so it works :)
- It works with 1 user, so it will work with multiple
- Don’t worry about performance we will just scale using multiple machines/processes
- Logging is into place

Disaster Strikes



Did anyone notice?

Logging != Monitoring

Logging

“recording to diagnose a system”

```
127.0.0.1 - frank [10/Oct/2000:13:55:36 -0700] "GET /apache_pb.gif HTTP/1.0" 200 2326
```

Monitoring

“observation, checking and recording”

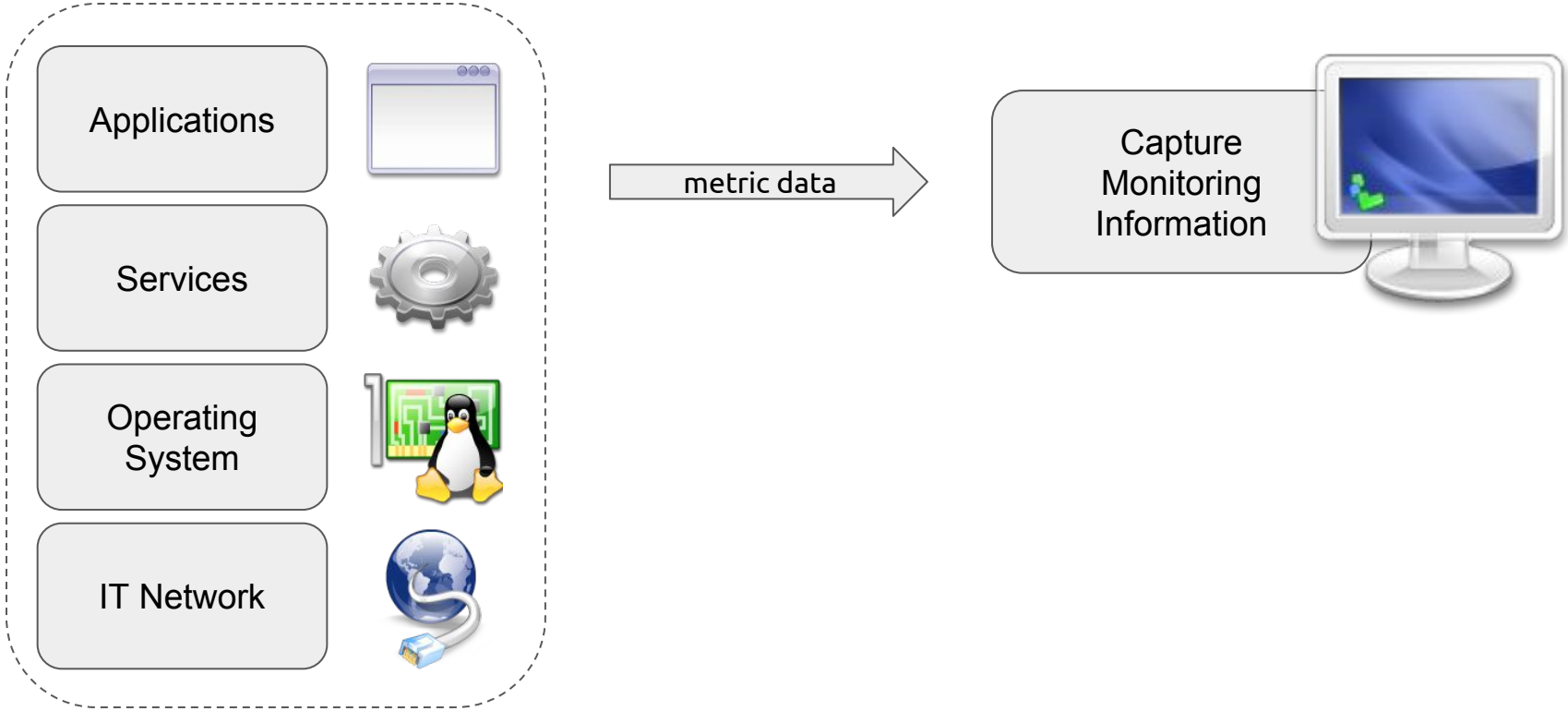
```
http_requests_total{method="post",code="200"} 1027 1395066363000
```

Vital Signs

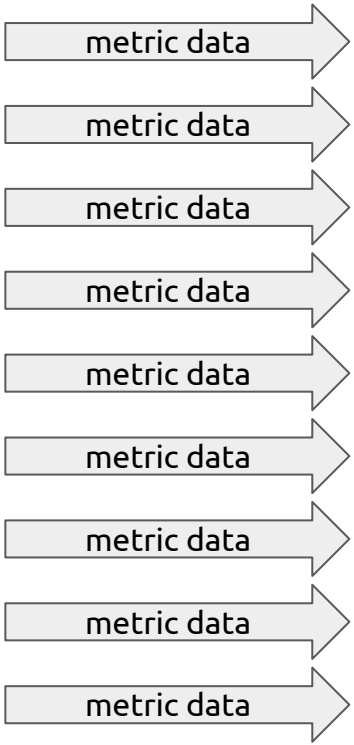
Why Monitoring?

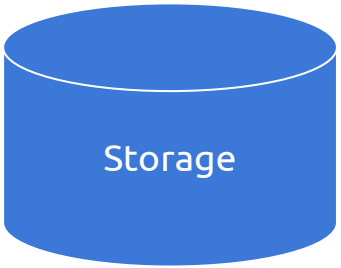
- Know when things go wrong
  - Detection & Alerting
- Be able to debug and gain insight
- Detect changes over time and drive technical/business decisions
- Feed into other systems/processes (e.g. security, automation)

What to monitor?



Houston we have Storage problem!

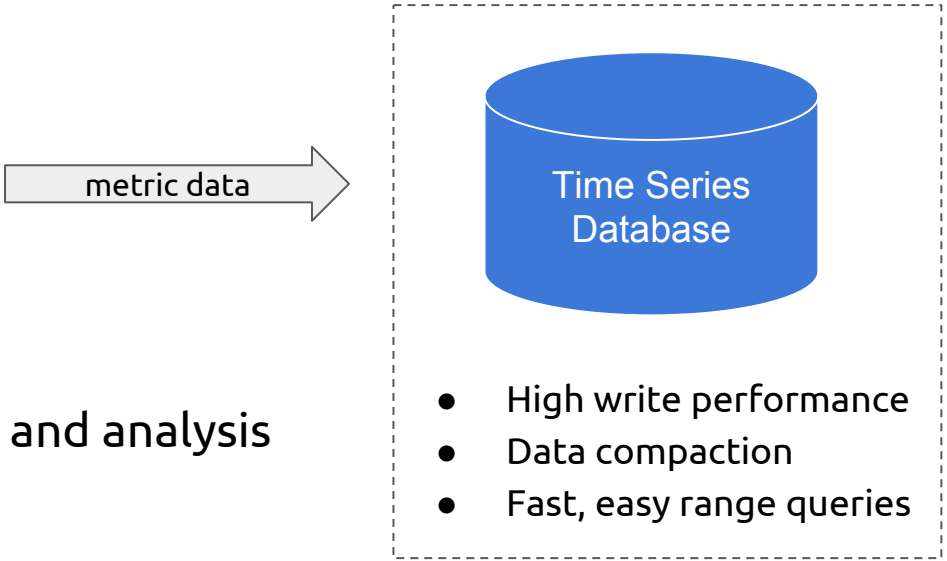




How to store the mass amount of metrics and also making them easy to query?

Time Series - Database

- Time series data is a sequence of data points collected at regular intervals over a period of time. (metrics)
  - Examples:
    - Device data
    - Weather data
    - Stock prices
    - Tide measurements
    - Solar flare tracking
- The data requires aggregation and analysis



Time Series - Data format

```
metric name and a set of key-value pairs, also known as labels
```

```
<metric name>{<label name>=<label value>, ...} value [ timestamp ]
```

```
http_requests_total{method="post",code="200"} 1027 1395066363000
```

23 systems in ranking, May 2018

Rank May 2018	Rank Apr 2018	Rank May 2017	DBMS	Database Model	Score May 2018	Score Apr 2018	Score May 2017
1.	1.	1.	InfluxDB	Time Series DBMS	11.00	+0.24	+3.05
2.	2.	↑ 5.	Kdb+	Multi-model	3.07	-0.01	+1.50
3.	3.	↓ 2.	RRDtool	Time Series DBMS	2.68	-0.07	-0.34
4.	4.	↓ 3.	Graphite	Time Series DBMS	2.26	+0.07	+0.25
5.	5.	↓ 4.	OpenTSDB	Time Series DBMS	1.62	-0.08	-0.05
6.	↑ 7.	↑ 8.	Prometheus	Time Series DBMS	1.12	+0.07	+0.64
7.	↓ 6.	↓ 6.	Druid	Time Series DBMS	1.01	-0.05	+0.07
8.	8.	↓ 7.	KairosDB	Time Series DBMS	0.43	-0.01	-0.08
9.	9.	9.	eXtremeDB	Multi-model	0.31	-0.01	-0.02
10.	10.	↑ 11.	Riak TS	Time Series DBMS	0.27	-0.00	+0.06
11.	11.	↑ 19.	Hawkular Metrics	Time Series DBMS	0.11	+0.00	+0.11
12.	12.	↑ 18.	Blueflood	Time Series DBMS	0.10	+0.00	+0.07
13.	↑ 15.	↓ 10.	Axibase	Time Series DBMS	0.05	+0.02	-0.16
14.	↑ 18.	↓ 12.	Warp 10	Time Series DBMS	0.04	+0.01	-0.15
15.	↑ 16.		TimescaleDB	Time Series DBMS	0.04	+0.01	
16.	↑ 17.	↓ 13.	TempoIQ	Time Series DBMS	0.02	-0.00	-0.13
17.	↓ 13.	↓ 15.	Machbase	Time Series DBMS	0.01	-0.07	-0.04
18.	↑ 19.	↓ 17.	Heroic	Time Series DBMS	0.00	±0.00	-0.03
18.	↑ 19.		IRONdb	Time Series DBMS	0.00	±0.00	
18.	↑ 19.	↑ 19.	Newts	Time Series DBMS	0.00	±0.00	±0.00

http://db-engines.com/en/ranking/time+series+dbms

Prometheus Overview

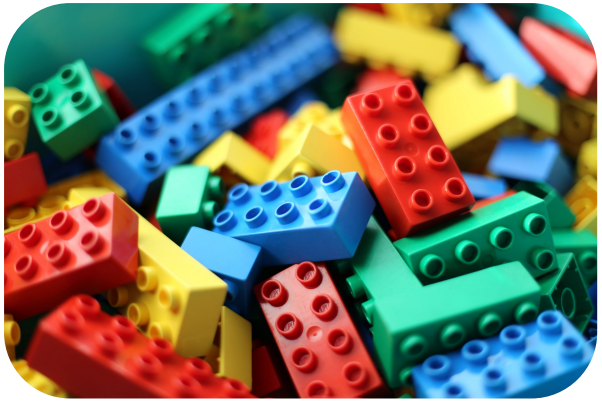
Prometheus

Prometheus is an open-source systems monitoring and alerting toolkit originally built at SoundCloud. It is now a standalone open source project and maintained independently of any company.

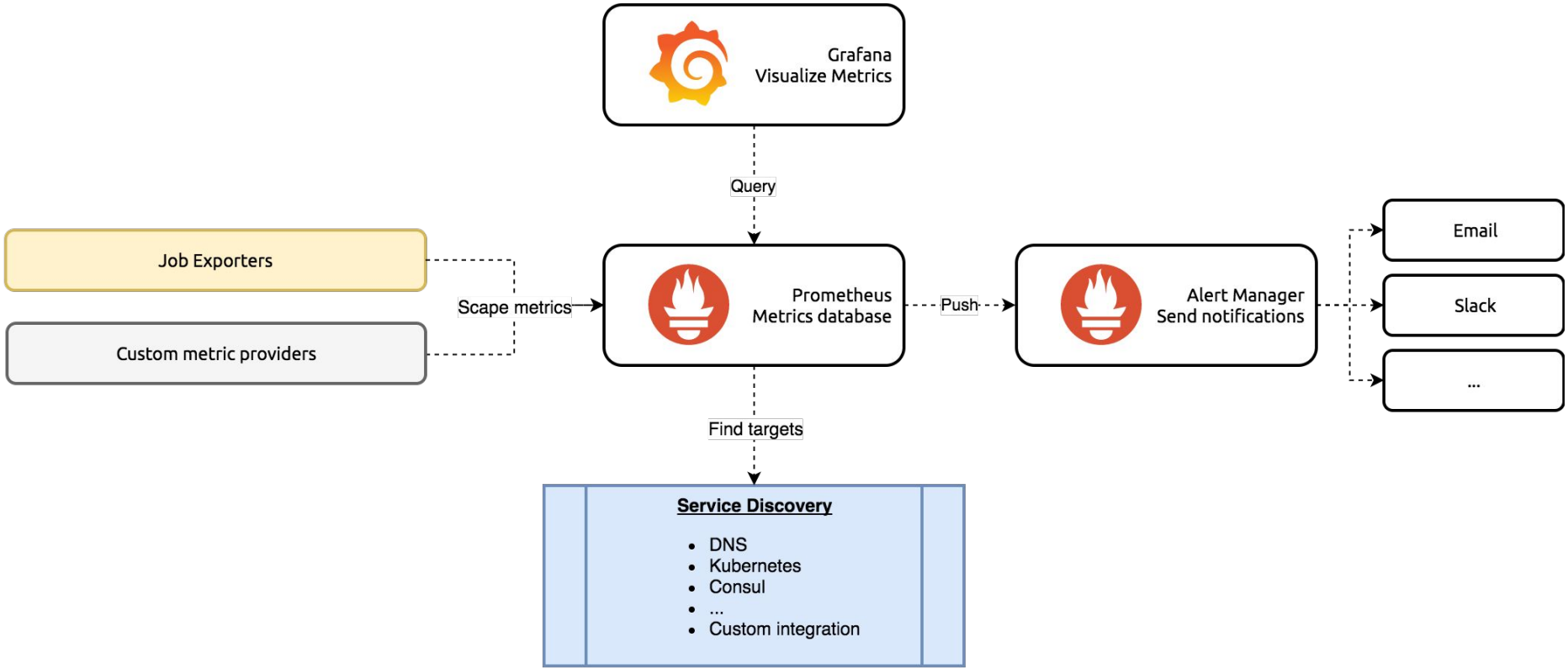
https://prometheus.io

Prometheus Components

- The main Prometheus server which scrapes and stores time series data
- Client libraries for instrumenting application code
- A push gateway for supporting short-lived jobs
- Special-purpose exporters (for HAProxy, StatsD, Graphite, etc.)
- An alertmanager
- Various support tools



Prometheus Overview

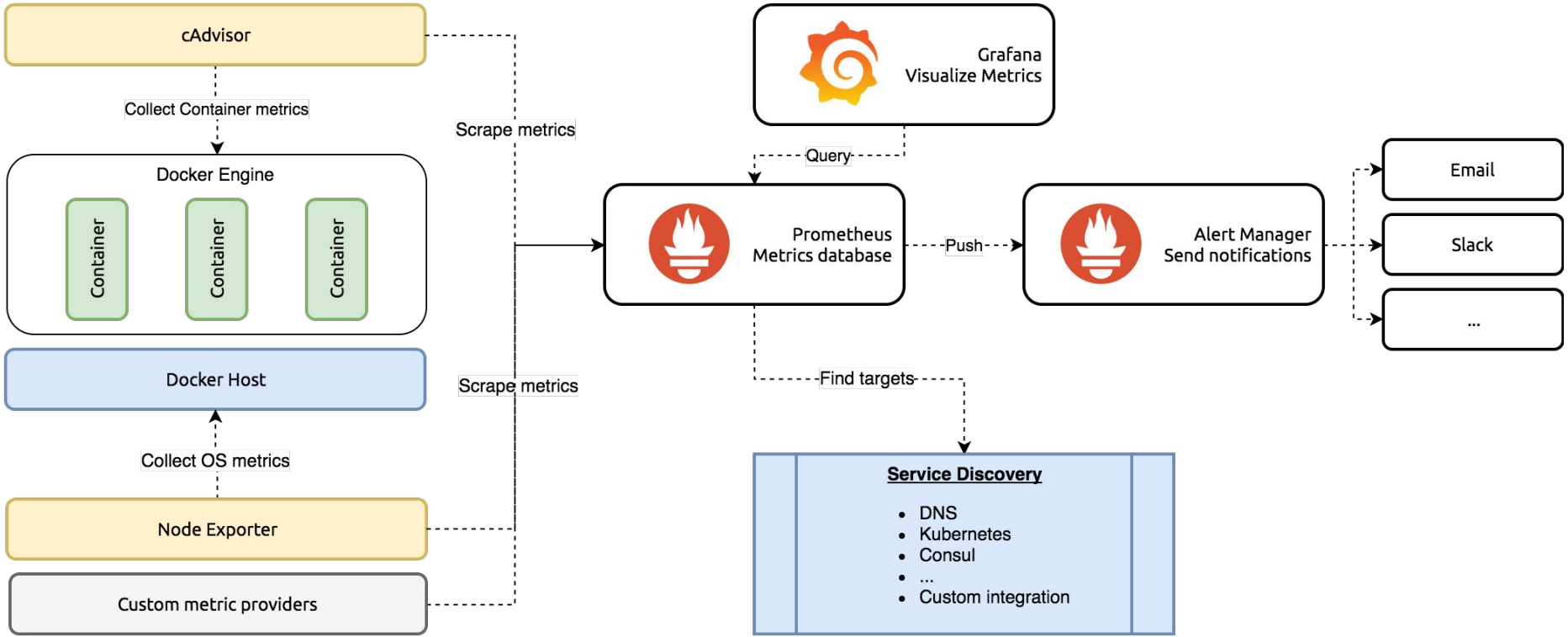


List of Job Exporters

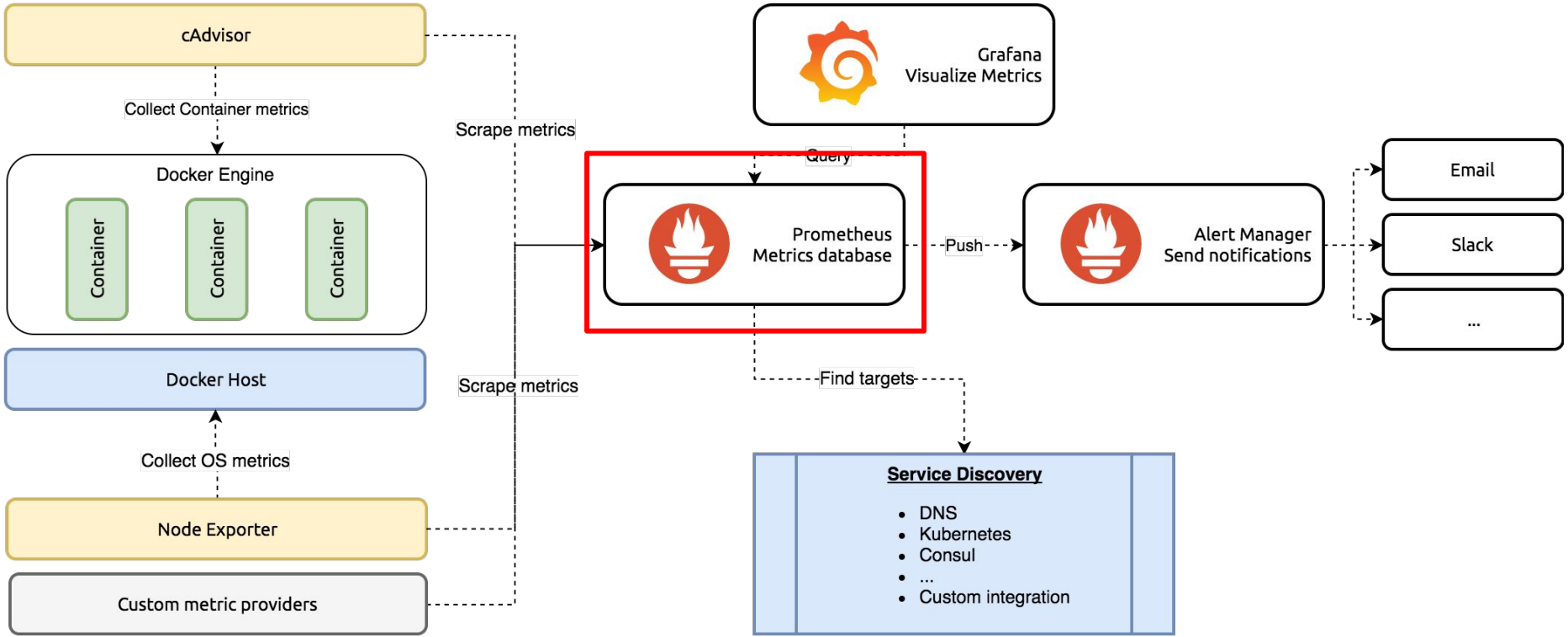
- Prometheus managed:
  - JMX
  - Node
  - Graphite
  - Blackbox
  - SNMP
  - HAProxy
  - Consul
  - Memcached
  - AWS Cloudwatch
  - InfluxDB
  - StatsD
  - ...
- Custom ones:
  - Database
  - Hardware related
  - Messaging systems
  - Storage
  - HTTP
  - APIs
  - Logging
  - …

https://prometheus.io/docs/instrumenting/exporters/

Demo Structure



Demo: Run Prometheus (native)

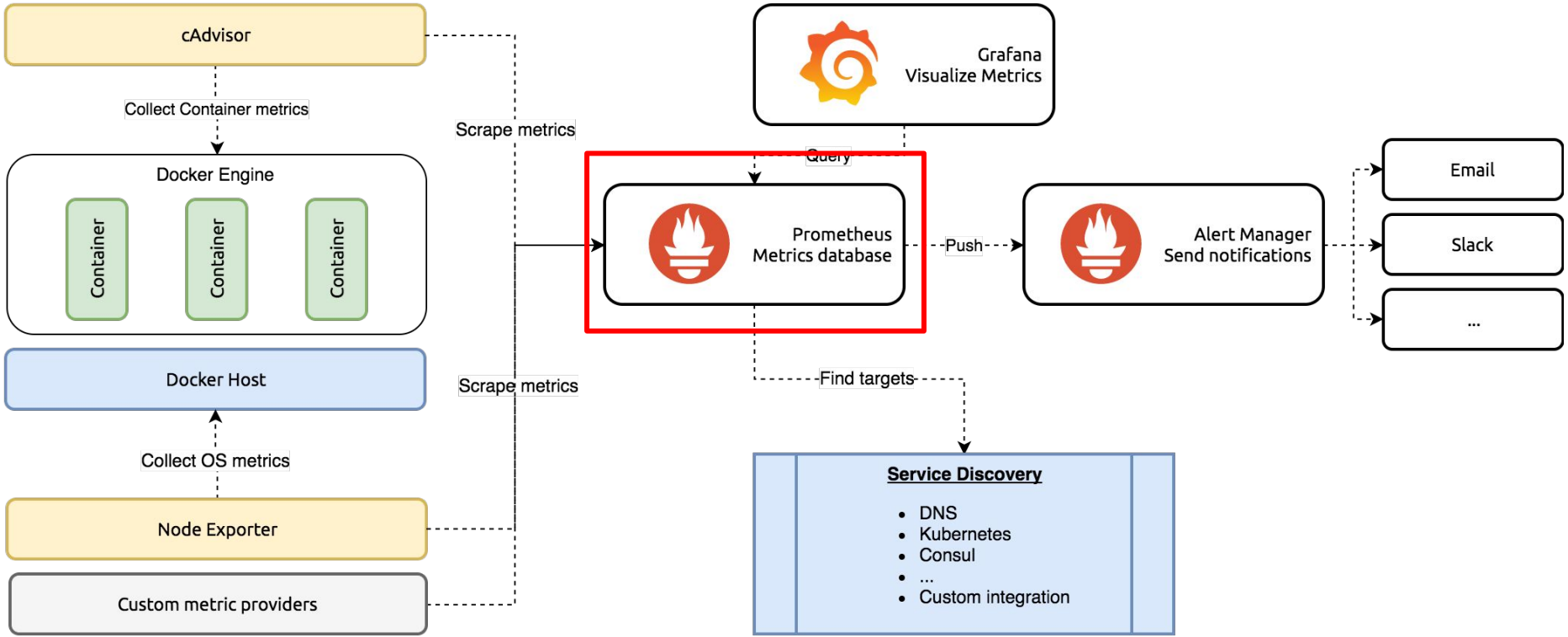


```yaml
# file: prometheus.yml
global:
  scrape_interval:     15s # Set the scrape interval to every 15 seconds. Default is every 1 minute.

# some settings intentionally removed!!

# A scrape configuration containing exactly one endpoint to scrape:
# Here it's Prometheus itself.
scrape_configs:
  # The job name is added as a label `job=<job_name>` to any timeseries scraped from this config.
  - job_name: 'prometheus'
    static_configs:
      - targets: ['localhost:9090']
```

Code Demo

“Running Prometheus Native”

Demo: Run Prometheus using Docker

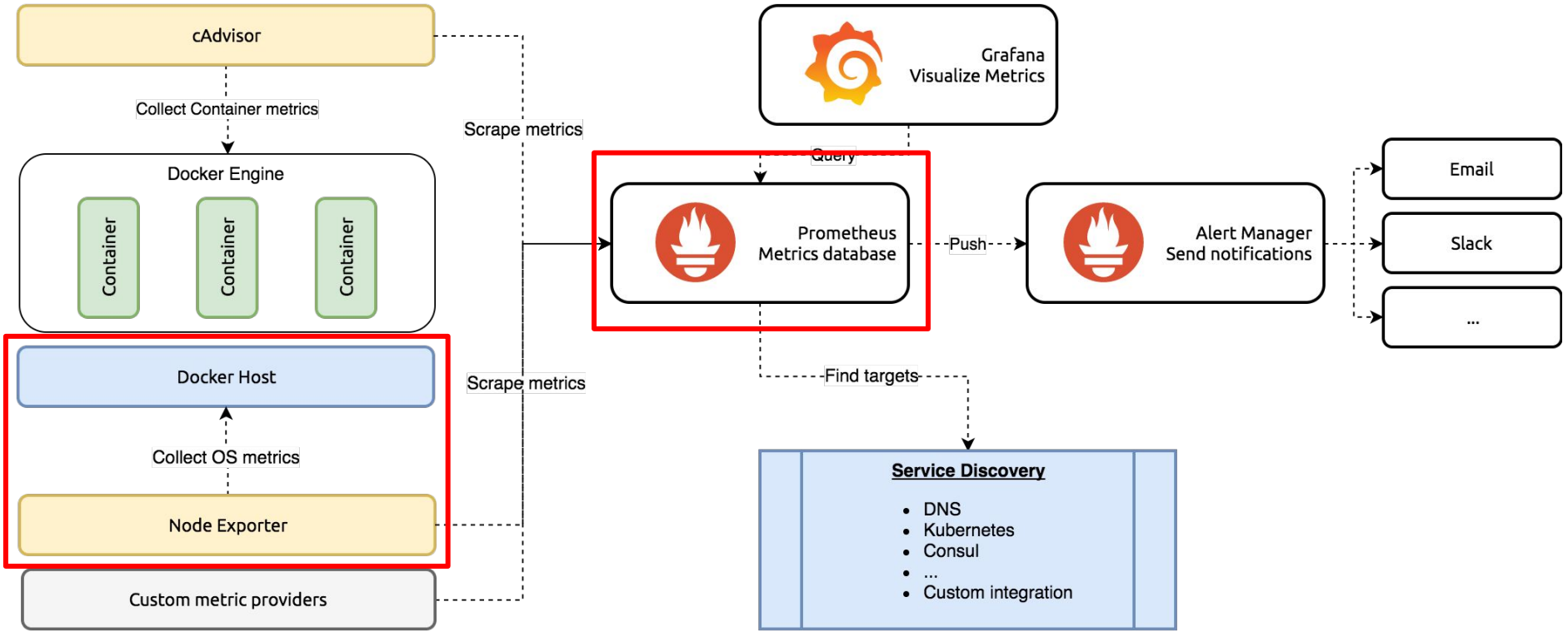


```yaml
# file: docker-compose.yml
version: '3'
services:
    prometheus:
        image: prom/prometheus:latest          → Using official prometheus container
        container_name: prometheus             → The container name
        ports:
            - "9090:9090"                      → Port mapping used for this container host:container
        volumes:
            - $PWD:/etc/prometheus/            → Mount local directory used for config
```

Code Demo

“Running Prometheus Dockerized”

Demo: Add host metrics



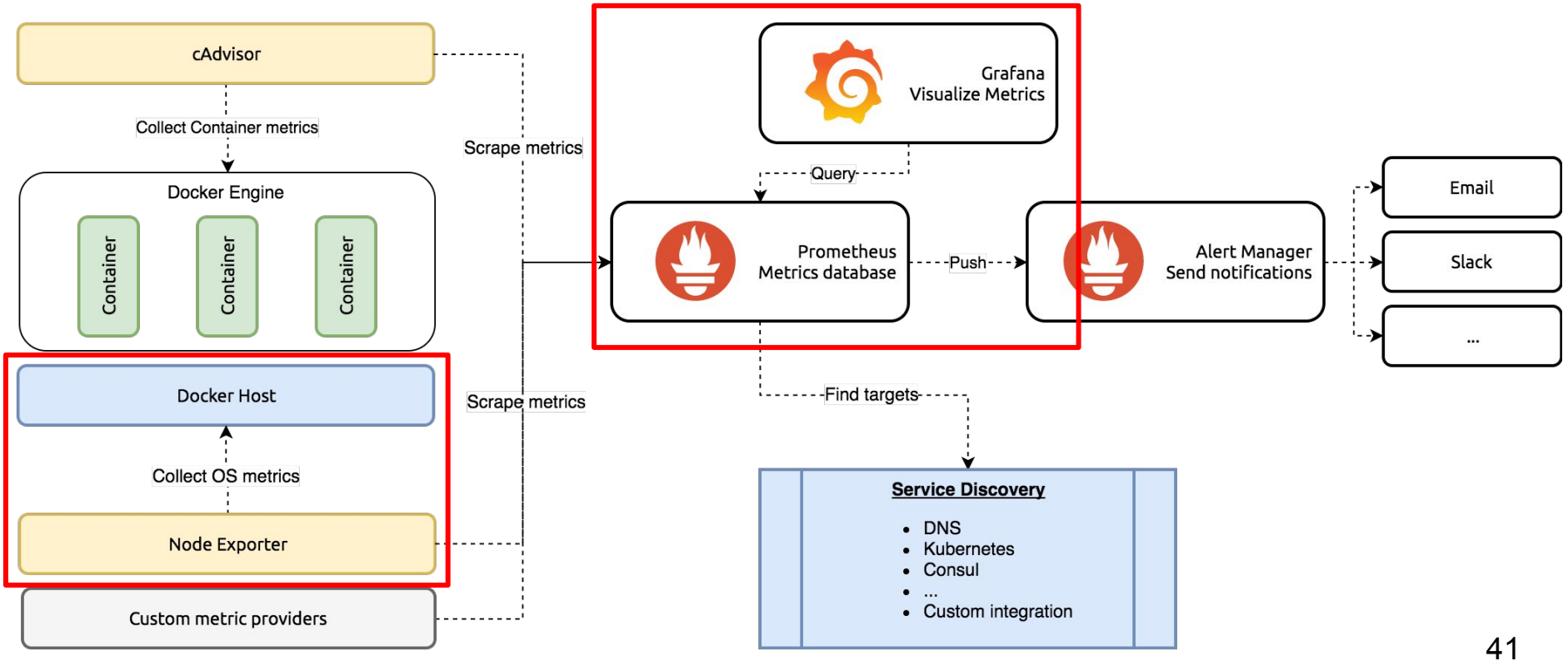
```yaml
# file: docker-compose.yml
version: '3'
services:

   prometheus:
       image: prom/prometheus:latest           → Using official prometheus container
       container_name: prometheus              → The container name
       ports:
           - "9090:9090"                       → Port mapping used for this container host:container
       volumes:
           - $PWD:/etc/prometheus/             → Mount local directory used for config

   node-exporter:
       image: prom/node-exporter:latest        → Using node exporter as an additional container
       container_name: node-exporter           → The container name
       ports:
         - '9100:9100'                         → Port mapping used for this container host:container
```

```yaml
# file: prometheus.yml
global:
  scrape_interval:     15s # Set the scrape interval to every 15 seconds. Default is every 1 minute.

# some settings intentionally removed!!

# A scrape configuration containing exactly one endpoint to scrape:
# Here it's Prometheus itself.
scrape_configs:
  # The job name is added as a label `job=<job_name>` to any timeseries scraped from this config.
  - job_name: 'prometheus'
    static_configs:
      - targets: ['localhost:9090']

  - job_name: 'node-exporter'
    static_configs:
      - targets: ['node-exporter:9100']
```

Code Demo

“Add host metrics”

Demo: Grafana



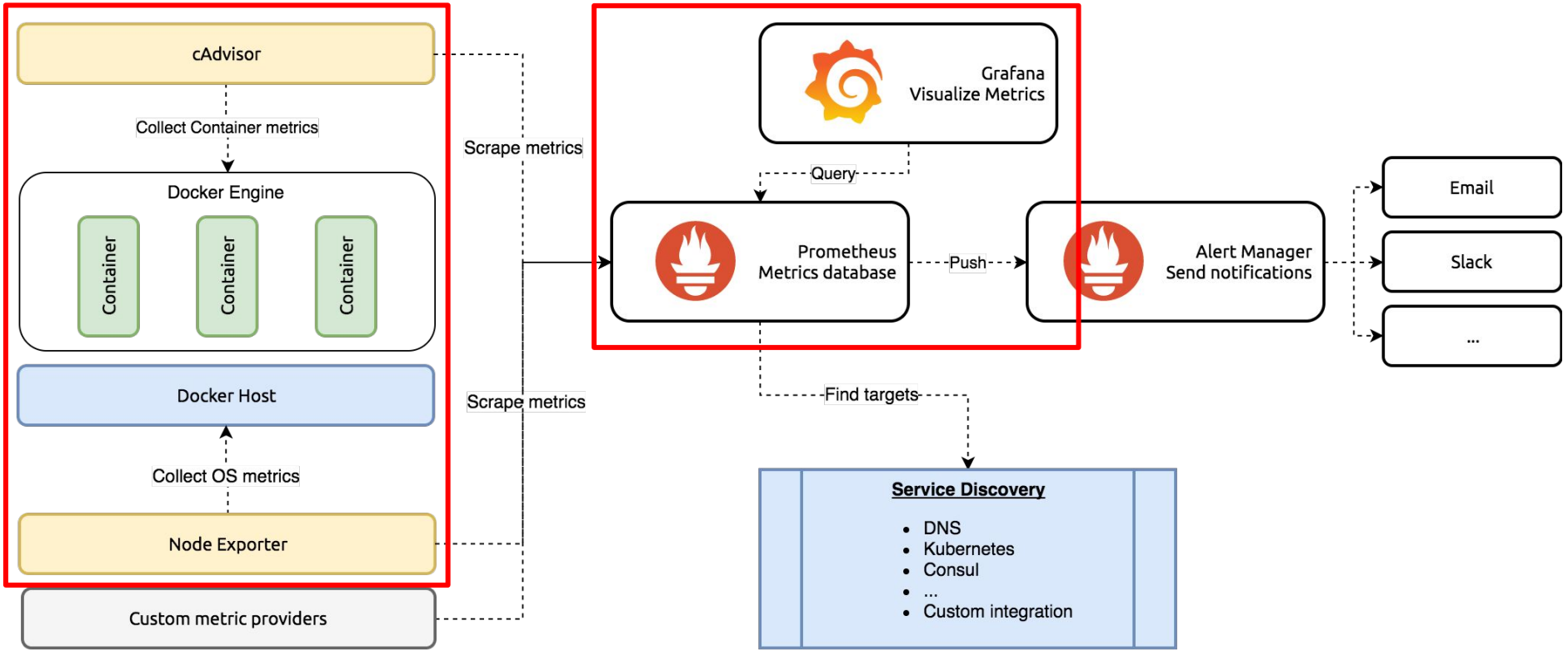
You get the idea :)

```yaml
# file: docker-compose.yml
version: '3'
services:

    # some code intentionally removed!!
    grafana:
        image: grafana/grafana:latest          → Using official prometheus container
        container_name: grafana                → The container name
        ports:
          - "3000:3000"                        → Port mapping used for this container host:container
```

Code Demo

“Grafana”

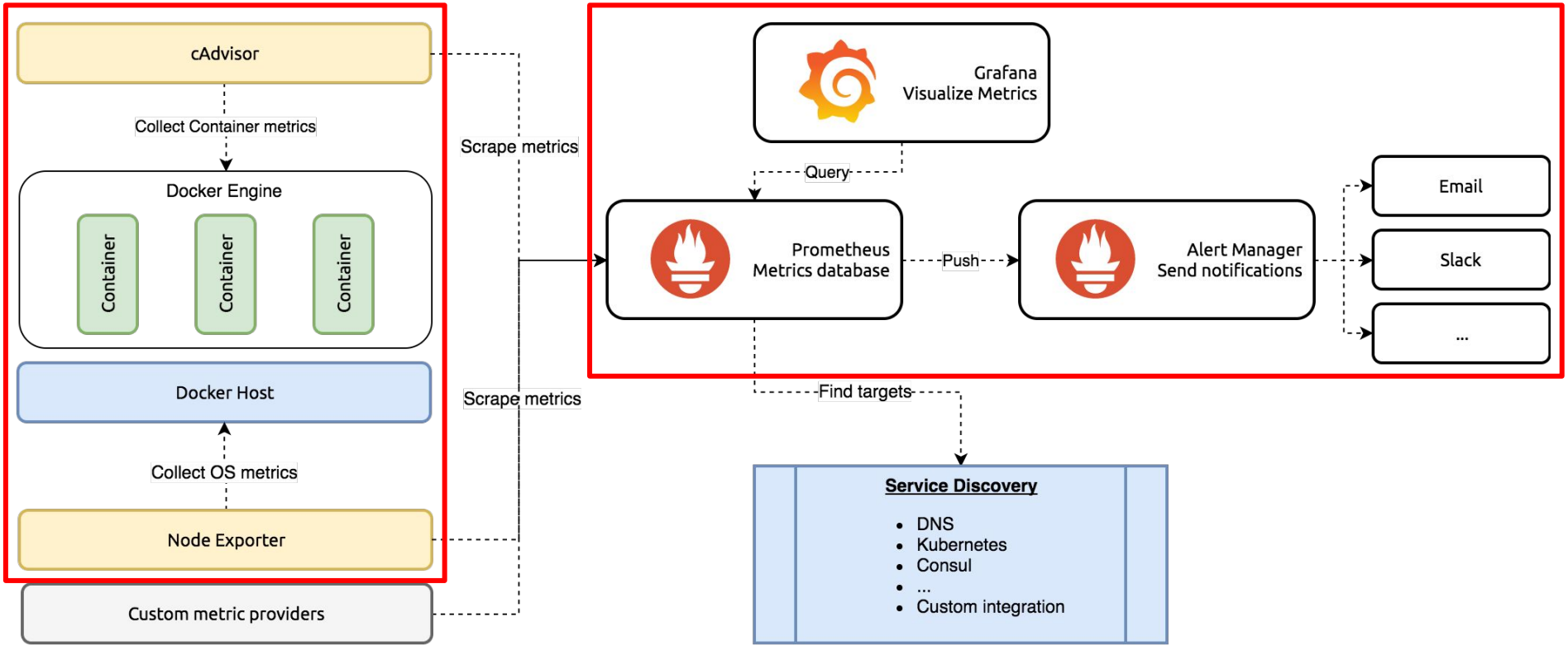
Demo: Monitor Docker containers



Code Demo

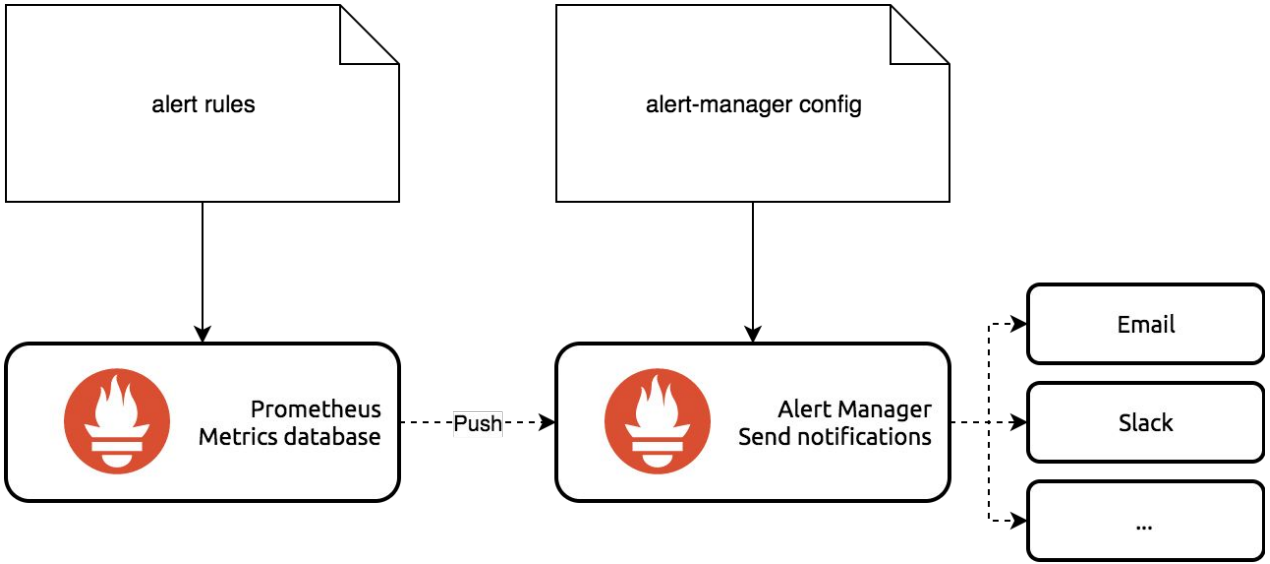
“cAdvisor”

Demo: Alerting



Alerting Configuration

- Alert Rules
  - What are the settings where we need to alert upon?
- Alert Manager
  - Where do we need to send the alert to?



```
# file: alert.rules

# Alert for <service> instance that is unreachable for 5 seconds
- alert: <service>_instance_down
expr: absent(container_memory_usage_bytes{name="<service"})
for: 5s
labels:
    severity: critical
annotations:
    summary: "Instance {{ $labels.instance }} down"
    description: "{{ $labels.instance }} of job {{ $labels.job }} has been down for more than 5 seconds."
```

```yaml
# file: alert-manager.yml
global:                                          → Global settings
 smtp_smarthost: 'mailslurper:2500'
 smtp_from: 'alertmanager@example.org'
 smtp_require_tls: false

route:                                           → Routing
 receiver: mail # Fallback                       → Fallback is there is no match
 routes:
 - match:
     severity: critical                          → Match on label!
   continue: true                                → Continue with other receivers if there is a match
   receiver: mail                                → Determine the receiver
 - match:
     severity: critical
   receiver: slack
```

```yaml
# file: alert-manager.yml (continued)
receivers:
- name: mail                                    → mail receiver
  email_configs:
    - send_resolved: true
      from: 'foo@bar.acme'
      to: 'please-help@bar.acme'
- name: slack                                   → slack receiver
  slack_configs:
    - send_resolved: true
      username: 'AlertManager'
      channel: '#alert'
      api_url: 'THIS IS A VERY SECRET URL :)’
```

```yaml
# file: prometheus.yml
global:
  scrape_interval:     15s # Set the scrape interval to every 15 seconds. Default is every 1 minute.

# Load rules once and periodically evaluate them according to the global 'evaluation_interval'.
rule_files:
  - "alert.rules"

# some settings intentionally removed!!
```

Code Demo

“Alerting -> The Alert Manager”

Instrumenting your own code!

- Counter
  - A cumulative metric that represents a single numerical value that only ever goes up
- Gauge
  - Single numerical value that can arbitrarily go up and down
- Histogram
  - Samples observations (usually things like request durations or response sizes) and counts them in configurable buckets. It also provides a sum of all observed values
- Summary
  - Histogram + total count of observations + sum of all observed values, it calculates configurable quantiles over a sliding time window

Available Languages

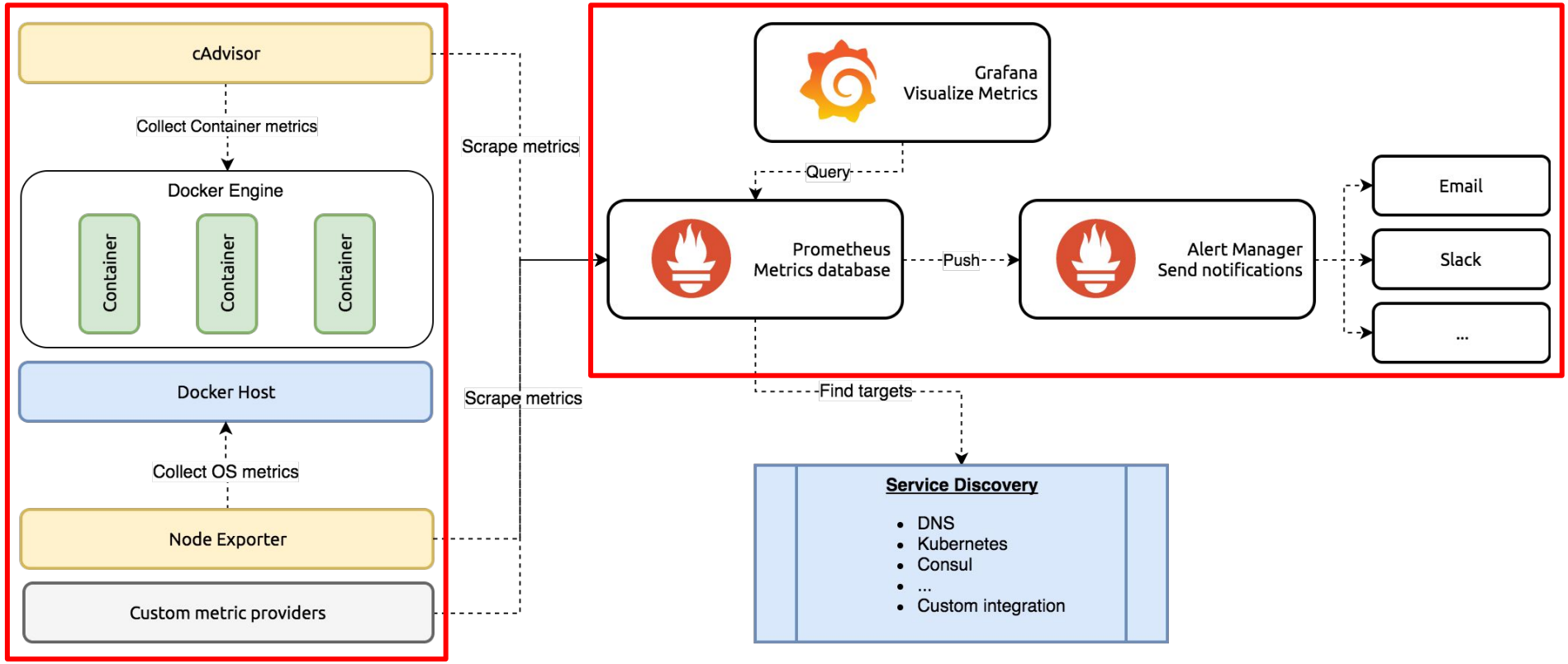
- Official
  - Go, Java or Scala, Python, Ruby
- Unofficial
  - Bash, C++, Common Lisp, Elixir, Erlang, Haskell, Lua for Nginx, Lua for Tarantool, .NET / C#, Node.js, PHP, Rust

```
// Spring Boot example -> file: build.gradle
dependencies {
    compile('org.springframework.boot:spring-boot-starter-web')
    testCompile('org.springframework.boot:spring-boot-starter-test')

    compile('io.prometheus:simpleclient_spring_boot:0.0.21')          → Add dependency
}
```

Prometheus Client Libaries: SpringBoot Example

```
@EnablePrometheusEndpoint
@EnableSpringBootMetricsCollector
@RestController
@SpringBootApplication
public class DemoApplication {
    public static void main(String[] args) { SpringApplication.run(DemoApplication.class, args); }
    static final Counter requests = Counter.build()              → create metric type counter
      .name("helloworld_requests_total")                         → set metric name
      .help("HelloWorld Total requests.").register();            → register the metric

    @RequestMapping("/helloworld")
    String home() {
        requests.inc();                  → increment the counter with 1 (helloworld_requests_total)
        return "Hello World!";
    }
}
```

Demo: Application metrics

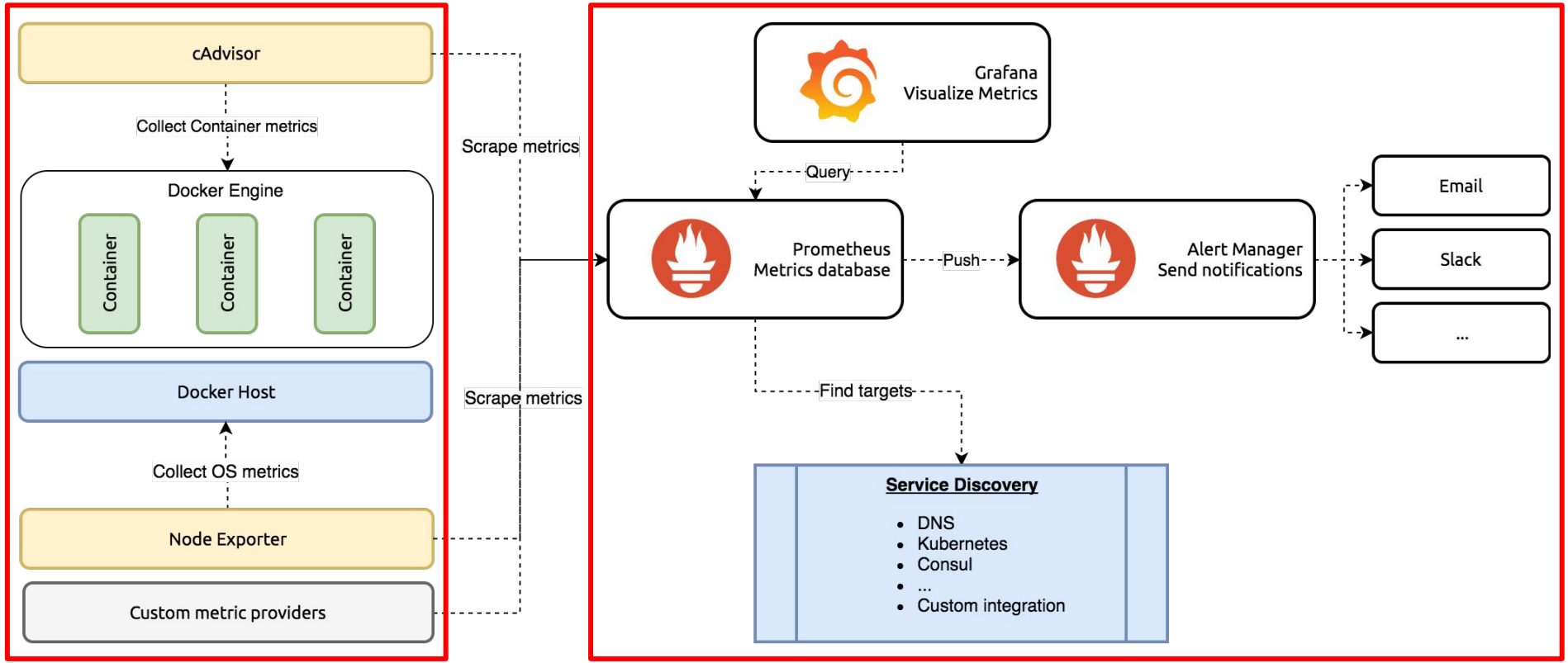


Code Demo

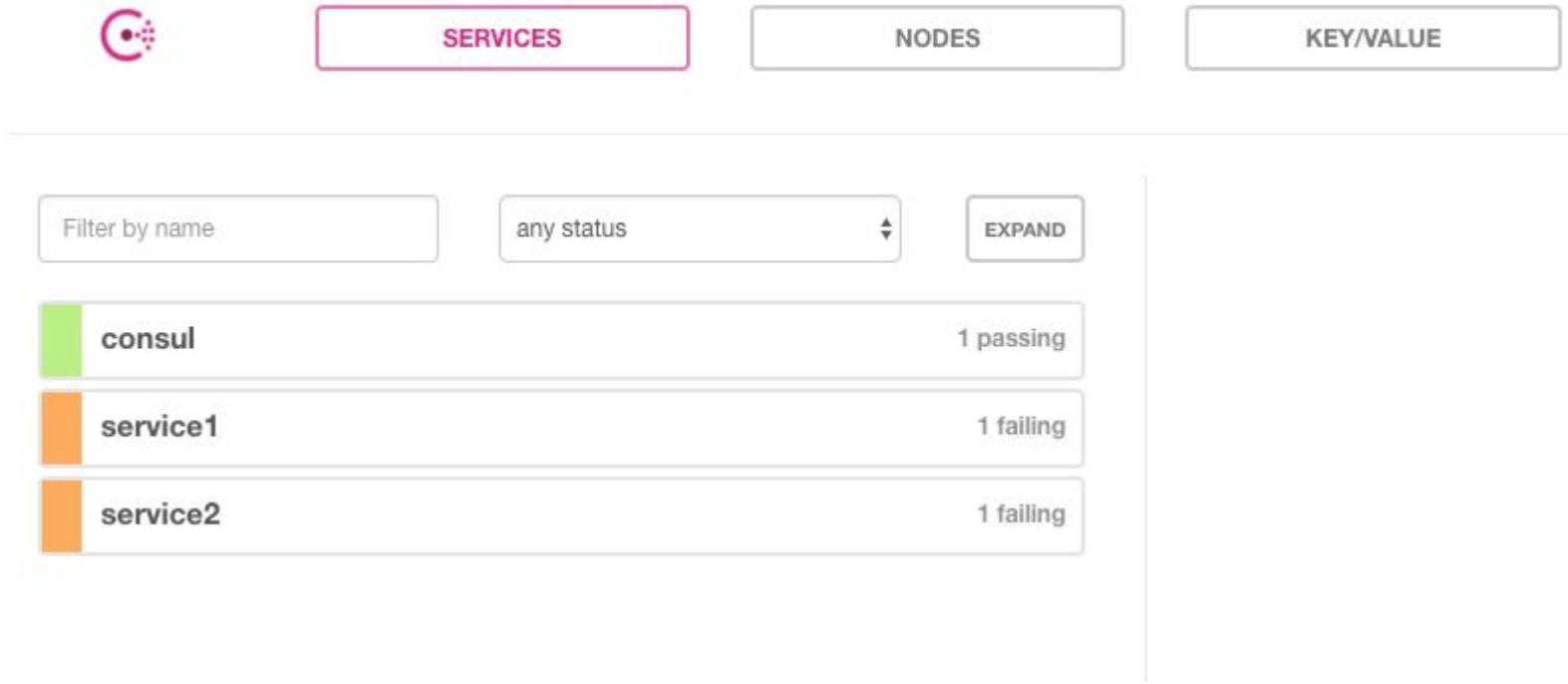
“Application metrics”

Service Discovery (Consul) Integration

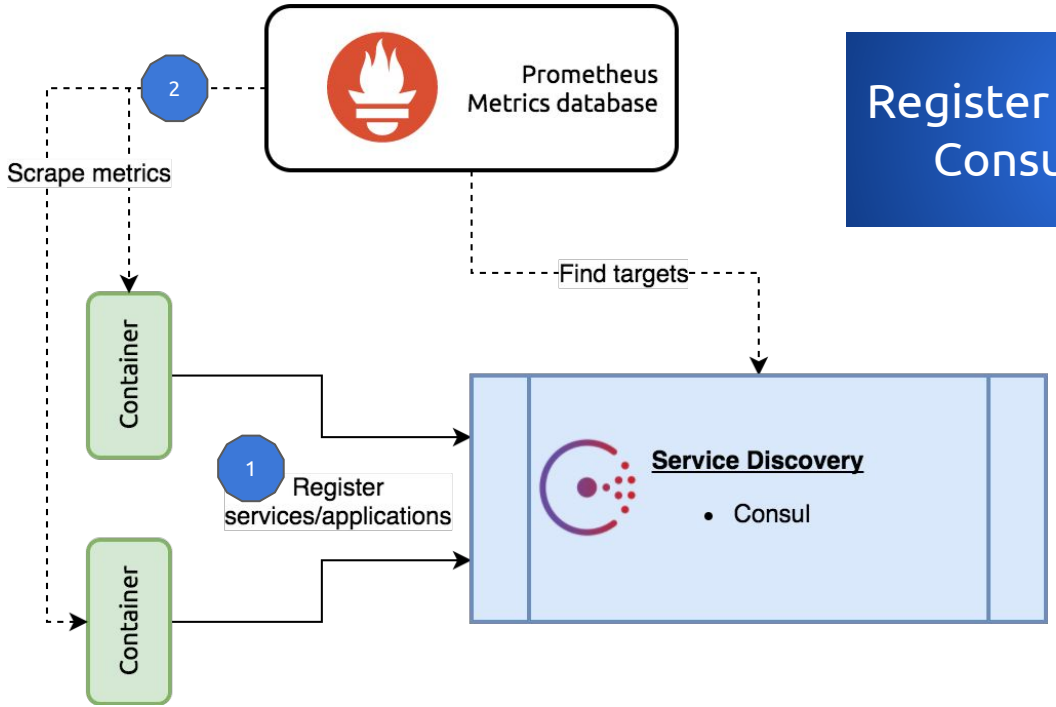
Demo: Consul Integration



Service Discovery



Demo: Consul integration



Register the services with Consul and Monitor

Code Demo

“Consul to the rescue”


I LOVE BAD MOVIES
"The Food Issue"
Vol. 6
You stood in line all day to get clean water. Enjoy some delicious Soylent Green with it!
Try serving tasty Soylent Green on a plate—like how food used to be!
INGREDIENTS
Plankton...
totally
delicious blend of essays, comics &
creations made from people like you!

https://github.com/mpas/infrastructure-and-system-monitoring-using-prometheus

That’s a wrap!

Question?