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
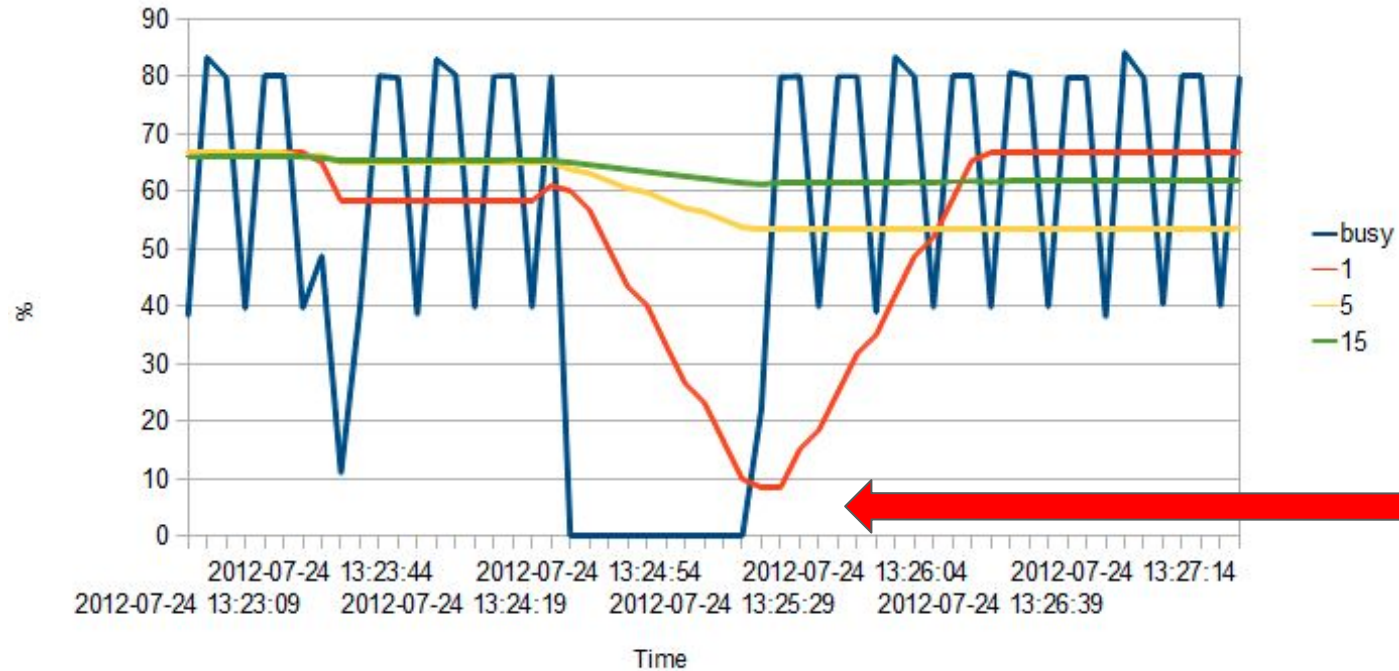 - Service Discovery (Consul) integration

..Quick Inventory..

Our scary movie "The Happy Developer"

- Lets 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
```
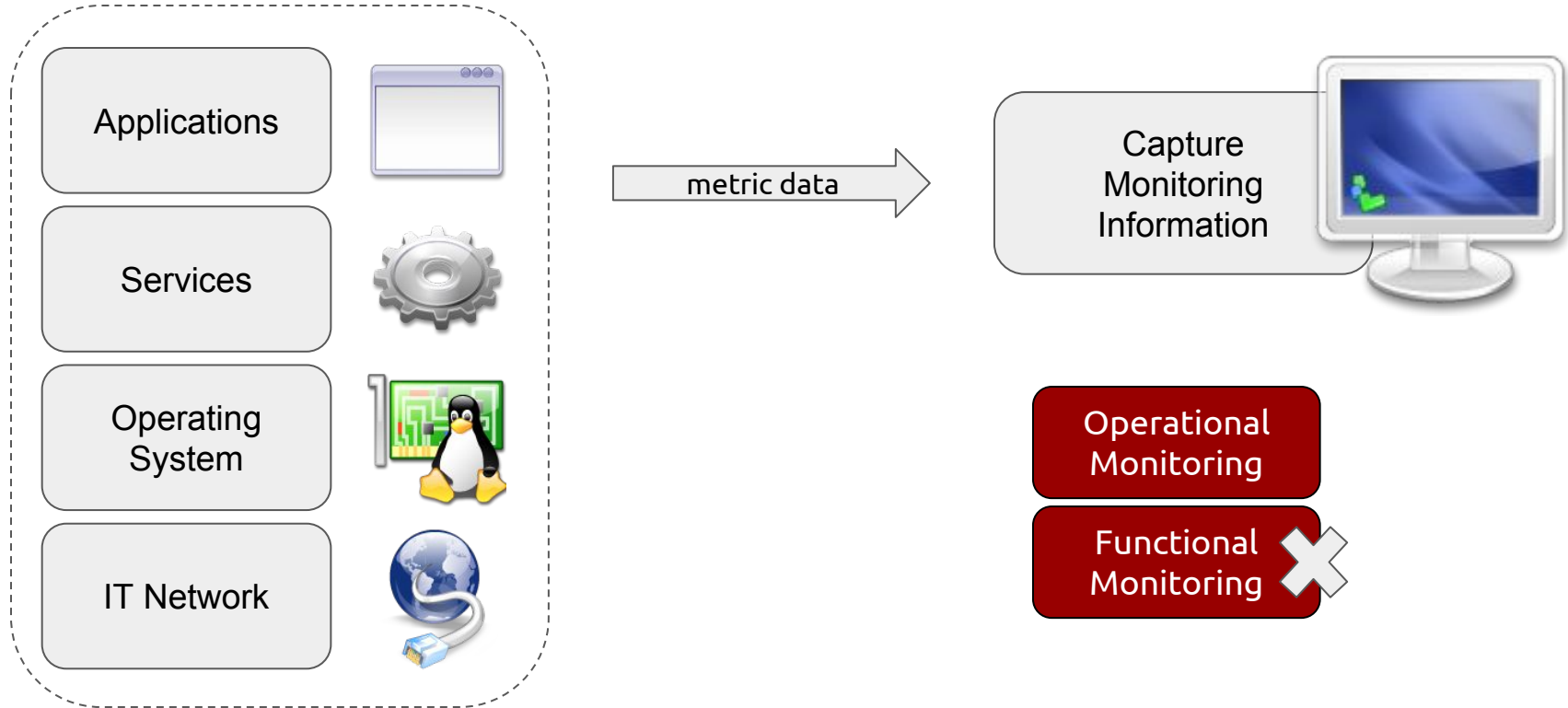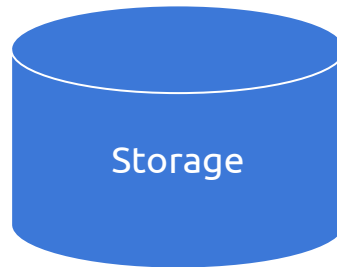
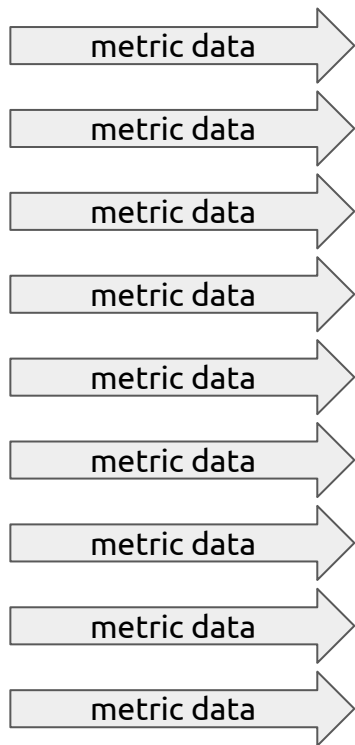Vital Signs

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?

Applications

Services

Operating System

IT Network

metric data →

Capture Monitoring Information

Operational Monitoring

Functional Monitoring ✖

Houston we have Storage problem!

metric data

metric data

metric data

metric data

metric data

metric data

metric data
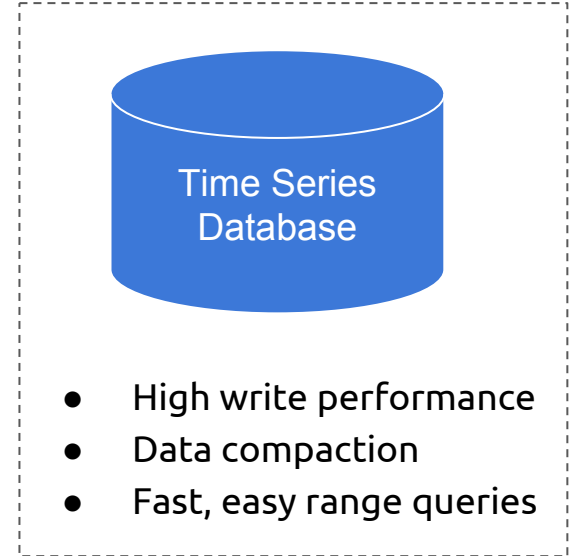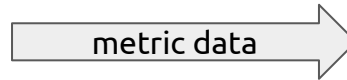
metric data

metric data

Storage

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

metric data →

Time Series Database

- High write performance
- Data compaction
- Fast, easy range queries

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

Rank			DBMS	Database Model	Score		
Apr 2017	Mar 2017	Apr 2016			Apr 2017	Mar 2017	Apr 2016
1.	1.	1.	InfluxDB ⊞	Time Series DBMS	7.24	+0.36	+3.28
2.	2.	2.	RRDtool	Time Series DBMS	2.87	+0.05	+0.34
3.	3.	3.	Graphite	Time Series DBMS	1.82	+0.04	+0.27
4.	4.	4.	OpenTSDB	Time Series DBMS	1.52	+0.02	+0.11
5.	5.	5.	Kdb+ ⊞	Multi-model ℹ	1.48	+0.05	+0.27
6.	6.	6.	Druid	Time Series DBMS	0.82	+0.02	+0.59
7.	7.	↑ 8.	Prometheus	Time Series DBMS	0.45	+0.05	+0.30
8.	8.	↓ 7.	KairosDB	Time Series DBMS	0.36	+0.06	+0.17
9.	9.	↑ 12.	Warp 10	Time Series DBMS	0.15	+0.04	+0.15
10.	10.	↓ 9.	Axibase	Time Series DBMS	0.14	+0.03	+0.02
11.	11.	↓ 10.	Riak TS ⊞	Time Series DBMS	0.13	+0.02	+0.11
12.	↑ 14.	↓ 11.	TempoIQ	Time Series DBMS	0.09	+0.05	+0.08
13.	↓ 12.		Heroic	Time Series DBMS	0.04	-0.03	
14.	↓ 13.	↓ 12.	Yanza	Time Series DBMS	0.04	-0.01	+0.04
15.	↑ 16.	↓ 12.	Blueflood	Time Series DBMS	0.01	+0.00	+0.01
16.	↓ 15.	↓ 12.	Newts	Time Series DBMS	0.01	-0.01	+0.01
17.	17.	↓ 12.	Hawkular Metrics	Time Series DBMS	0.00	±0.00	±0.00
17.	17.	↓ 12.	Infiniflux	Time Series DBMS	0.00	±0.00	±0.00
17.	17.		SiriDB	Time Series DBMS	0.00	±0.00	
17.	17.	↓ 12.	SiteWhere	Time Series DBMS	0.00	±0.00	±0.00

20 systems in ranking, April 2017

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

Prometheus Overview

Prometheus

Prometheus is an <u>open-source</u> 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

Implemented using
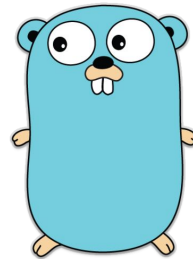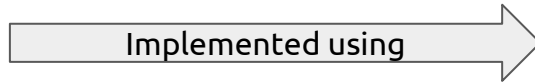
Prometheus Components

- 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

- WhiteBox Monitoring instead of probing [aka BlackBox Monitoring]
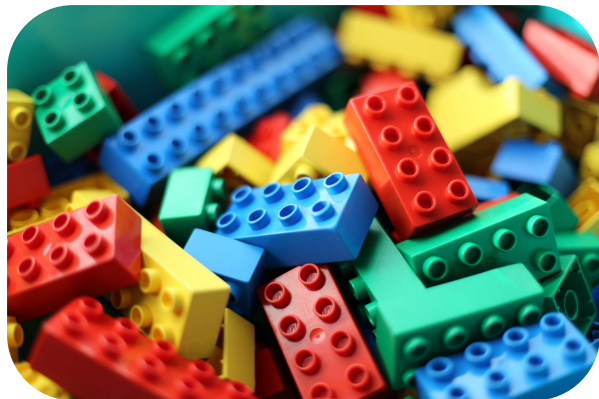
Prometheus Overview
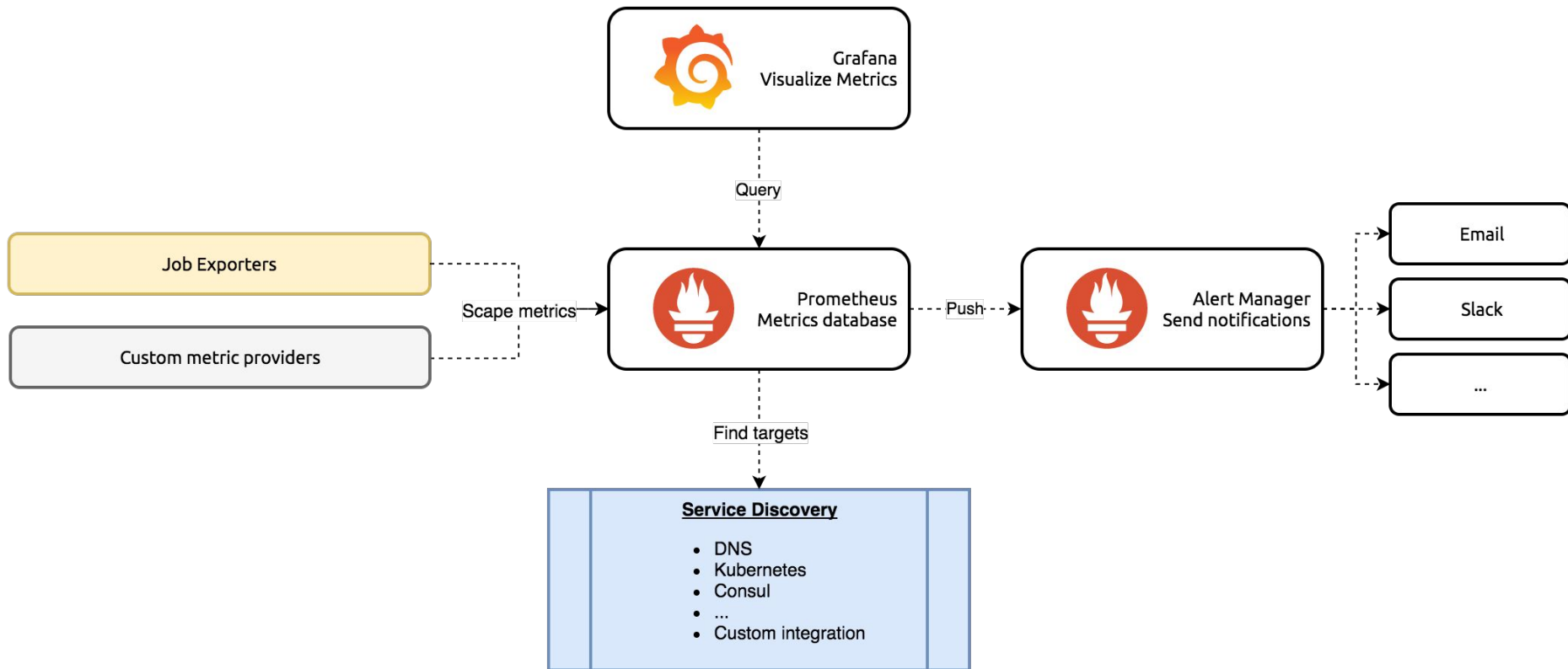
List of Job Exporters

- 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
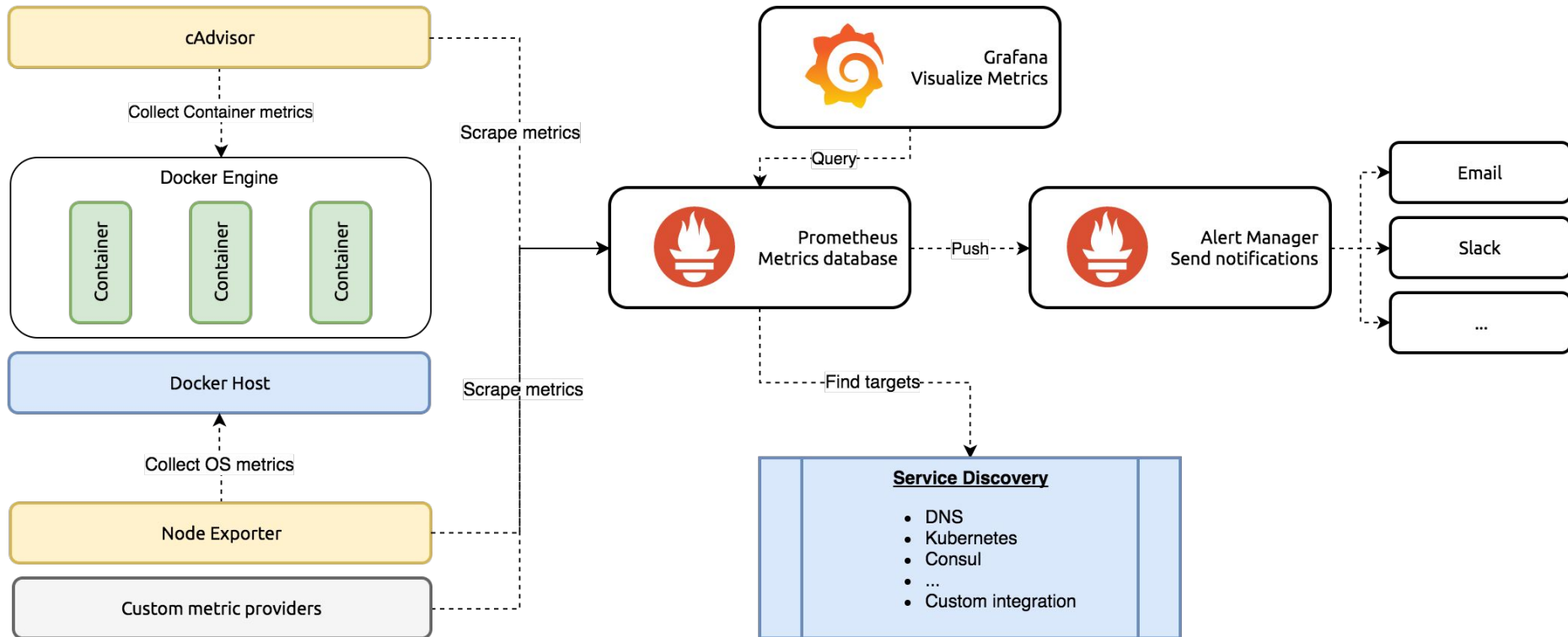
Demo: Run Prometheus (native)
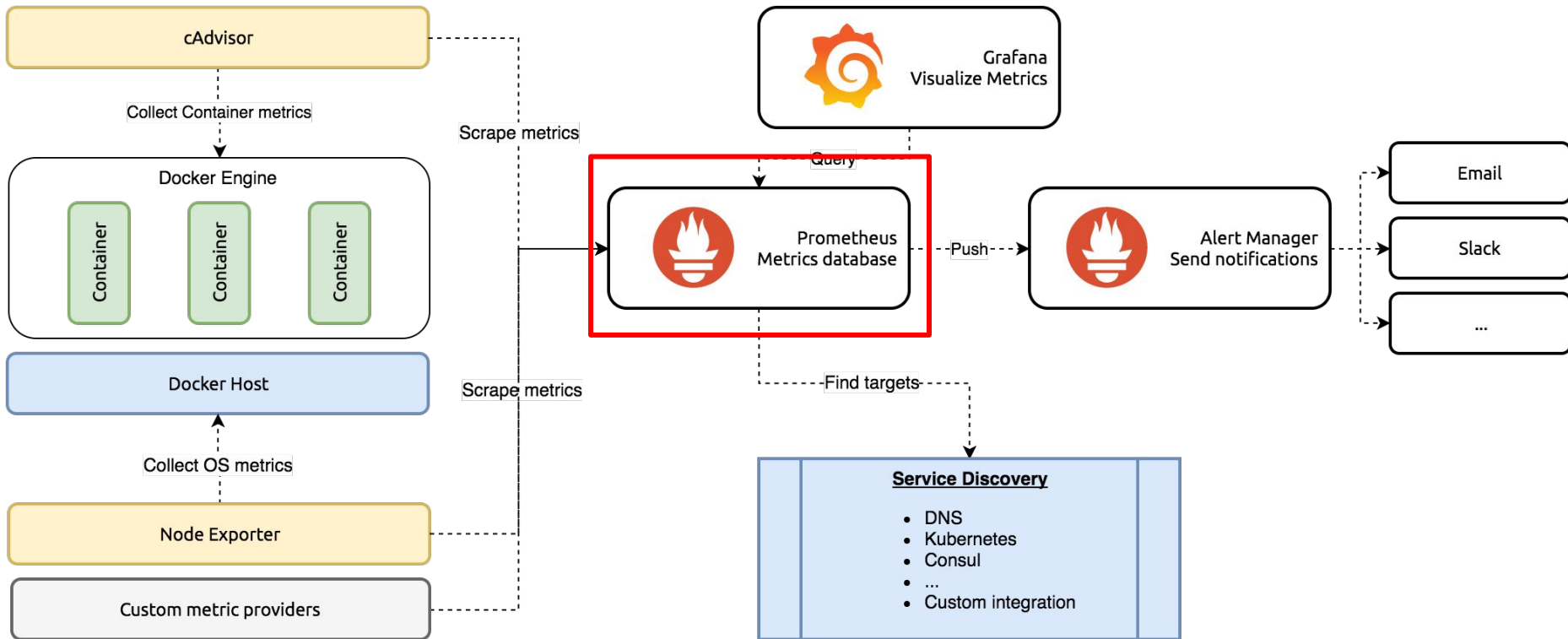
```yaml
# file: prometheus.yml
global:
  scrape_interval:     15s # Set the scrape interval to every 15 seconds. Default is every 1 minute.


# 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
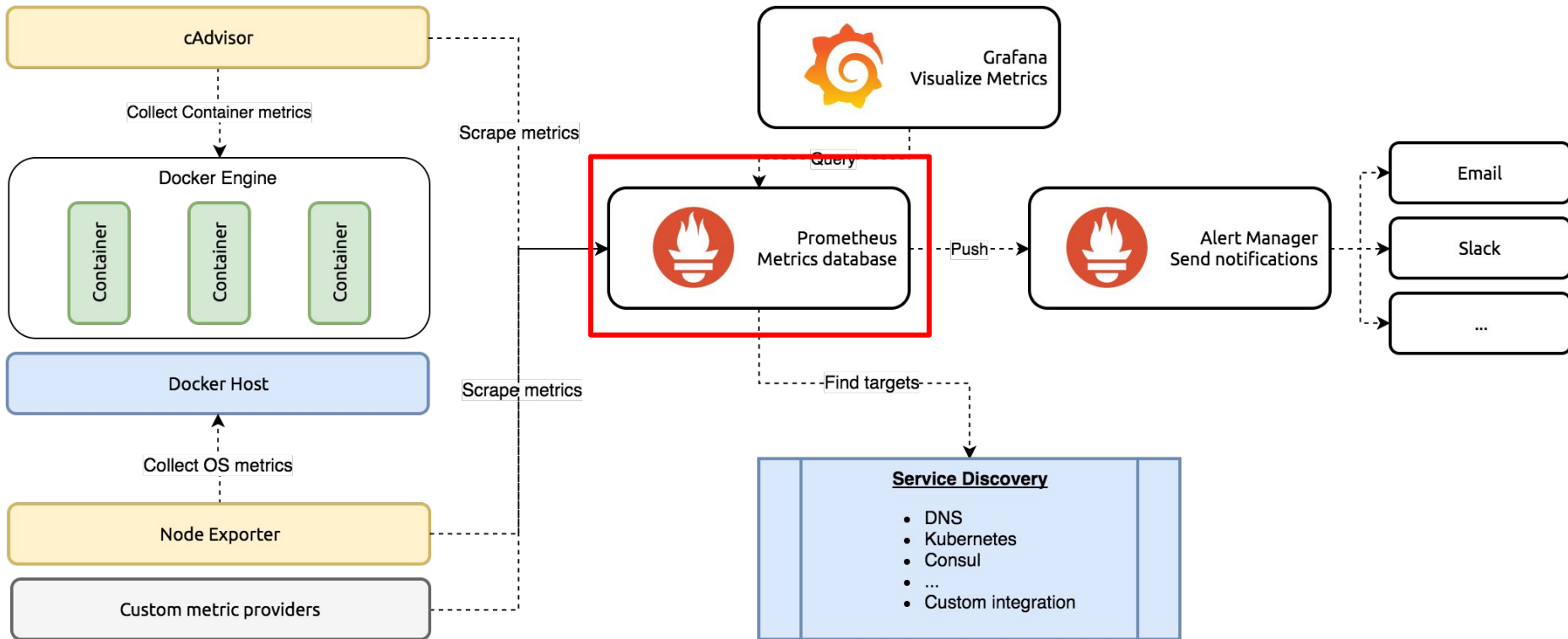
```yaml
# file: docker-compose.yml
version: '2'
services:

    prometheus:
        image: prom/prometheus:latest          → Using official prometheus container
        volumes:
            - $PWD:/etc/prometheus             → Mount local directory used for config + data
        ports:
            - "9090:9090"                      → Port mapping used for this container host:container
        command:
            - "-config.file=/etc/prometheus/prometheus.yml"  → Prometheus configuration
```

Code Demo

"Running Prometheus Dockerized"

Demo: Add host metrics
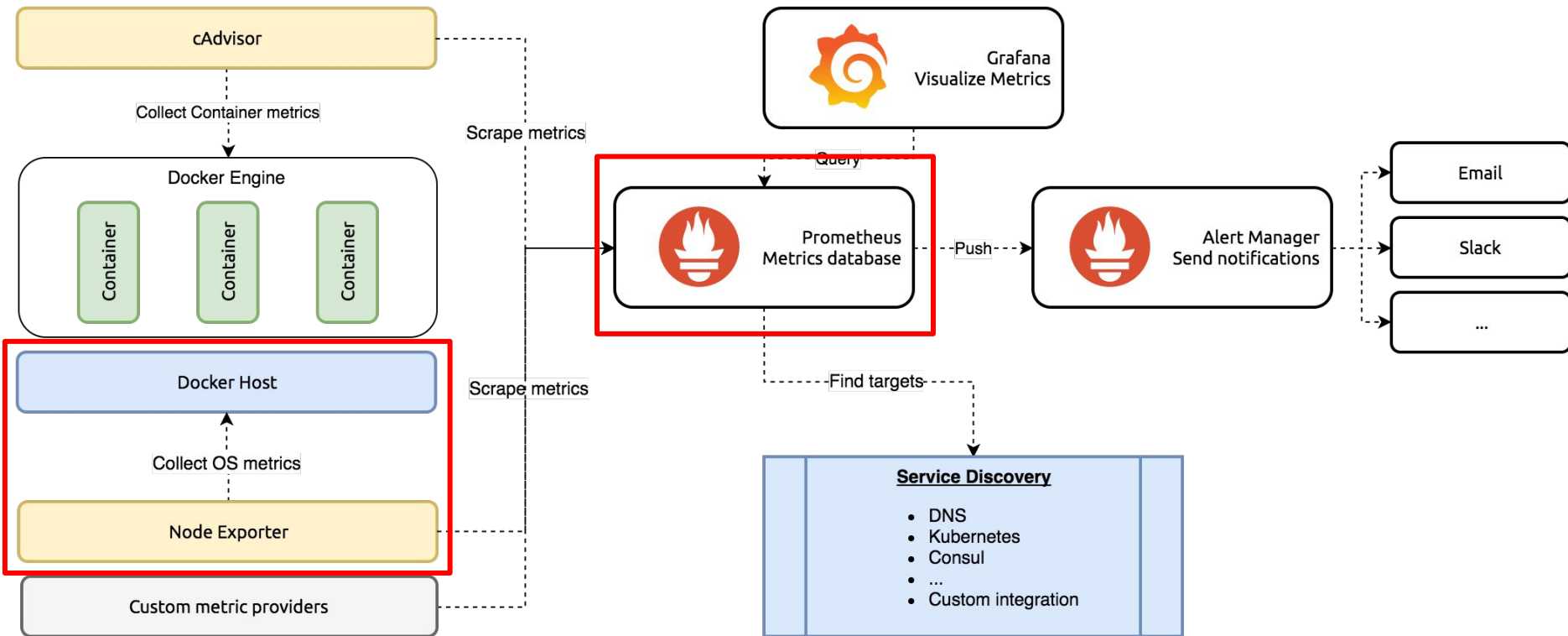
```yaml
# file: docker-compose.yml
version: '2'
services:

    prometheus:                                          → Runnning prometheus as Docker container
        image: prom/prometheus:latest                    → Using official prometheus container
        volumes:
            - $PWD:/etc/prometheus                        → Mount local directory used for config + data
        ports:
            - "9090:9090"                                 → Port mapping used for this container host:container
        command:
            - "-config.file=/etc/prometheus/prometheus.yml"  → Prometheus configuration

    node-exporter:                                       → Using node exporter as an additional container
        image: prom/node-exporter:latest                 → Using node exporter as an additional container
        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
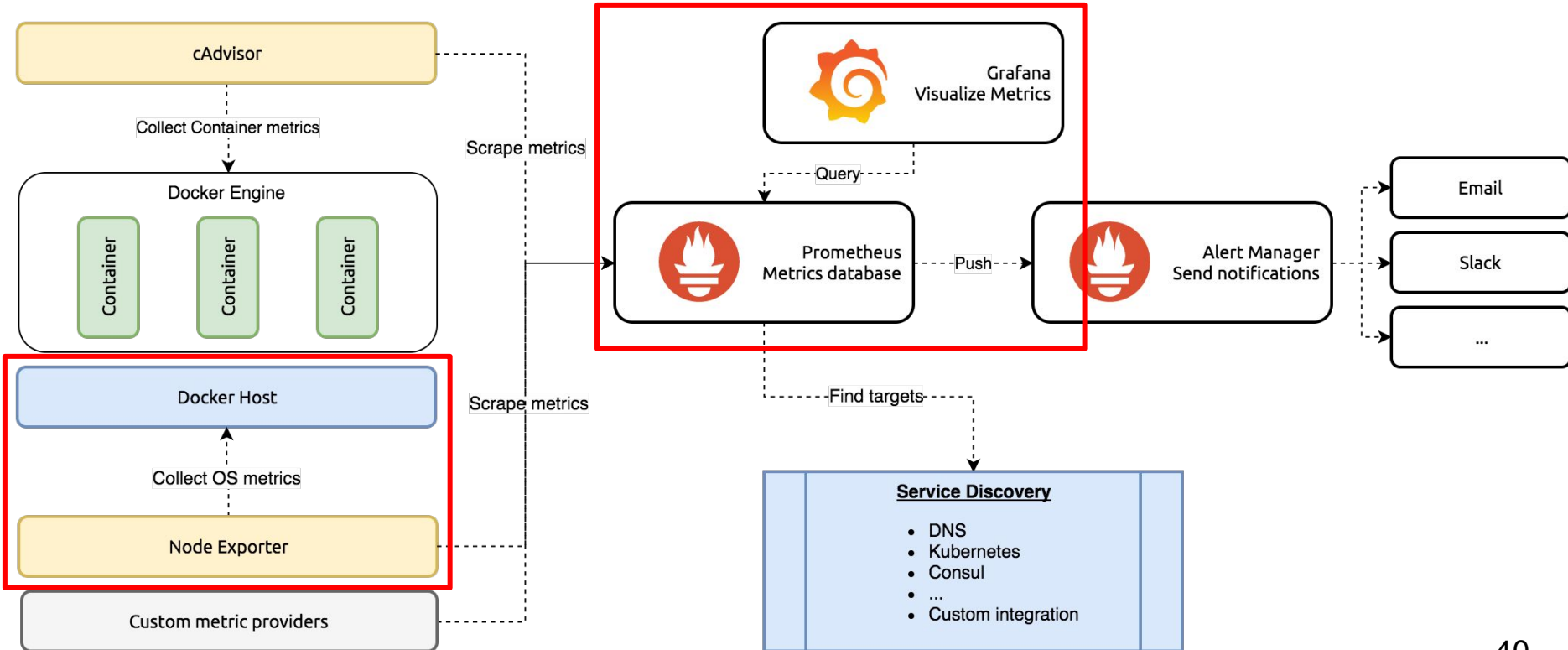
You get the idea :)

```yaml
# file: docker-compose.yml
version: '2'
services:

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

Code Demo

"Grafana"

Demo: Monitor Docker containers

Code Demo

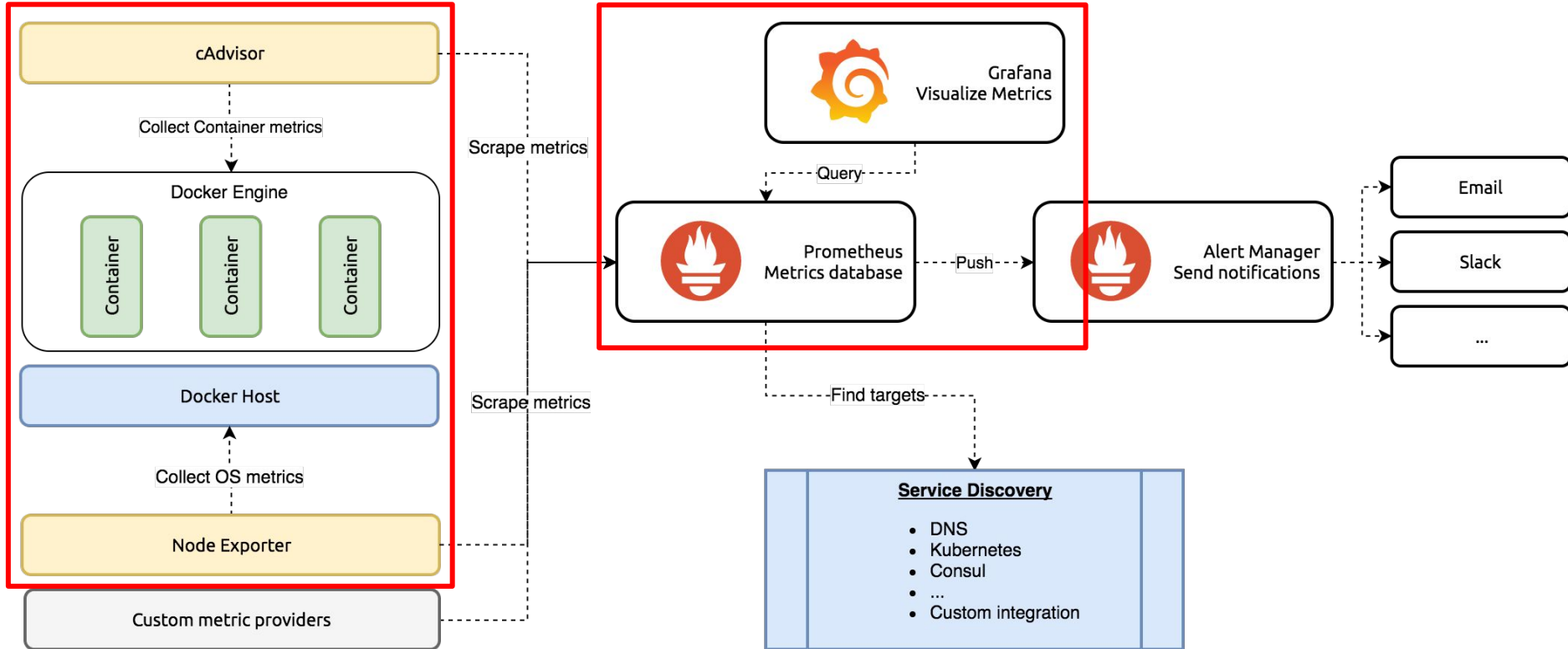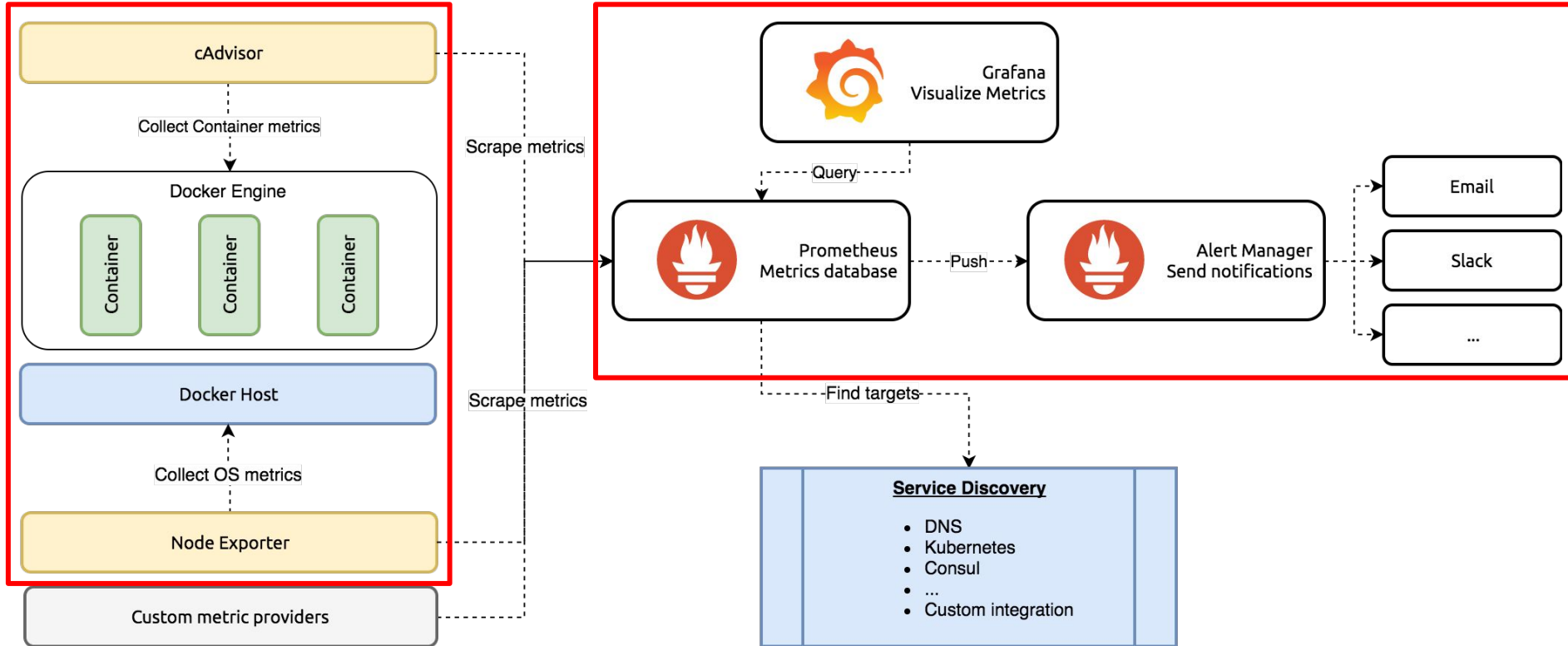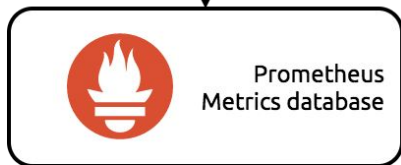# "cAdvisor"

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?
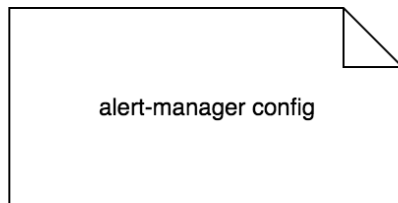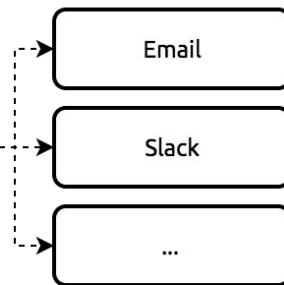
```
# file: alert.rules
ALERT serviceDownAlert
    IF absent(((time() - container_last_seen{name="<service_name>"}) < 5))
    FOR 5s
    LABELS {
        severity = "critical",                    → setting the labels so we can use them in the AlertManager
        service = "backend"
    }
    ANNOTATIONS {                                 → information used in the alert event
      SUMMARY = "Container Instance down",
      DESCRIPTION = "Container Instance is down for more than 15 sec."
    }
```

```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:
    - to: 'team-X+alerts@example.org'
- 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

```java
@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!";

    }

}
```
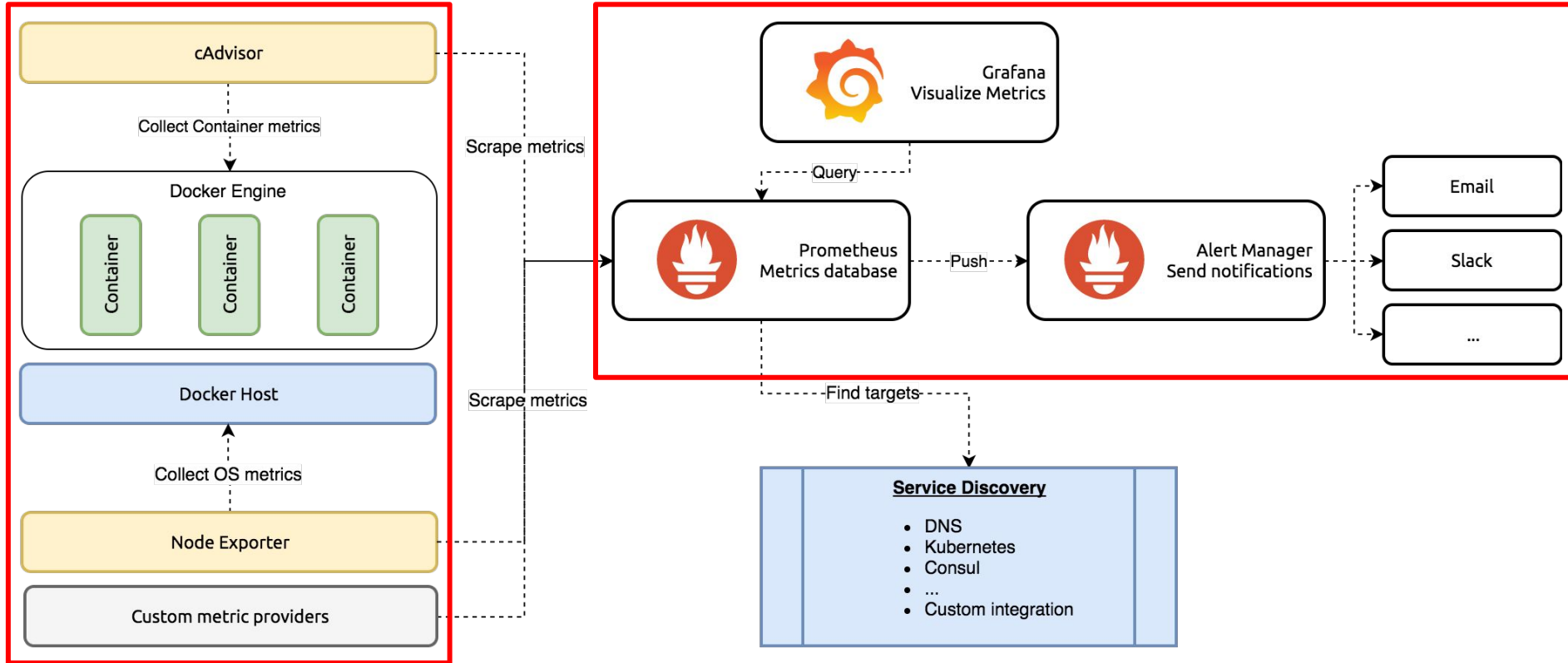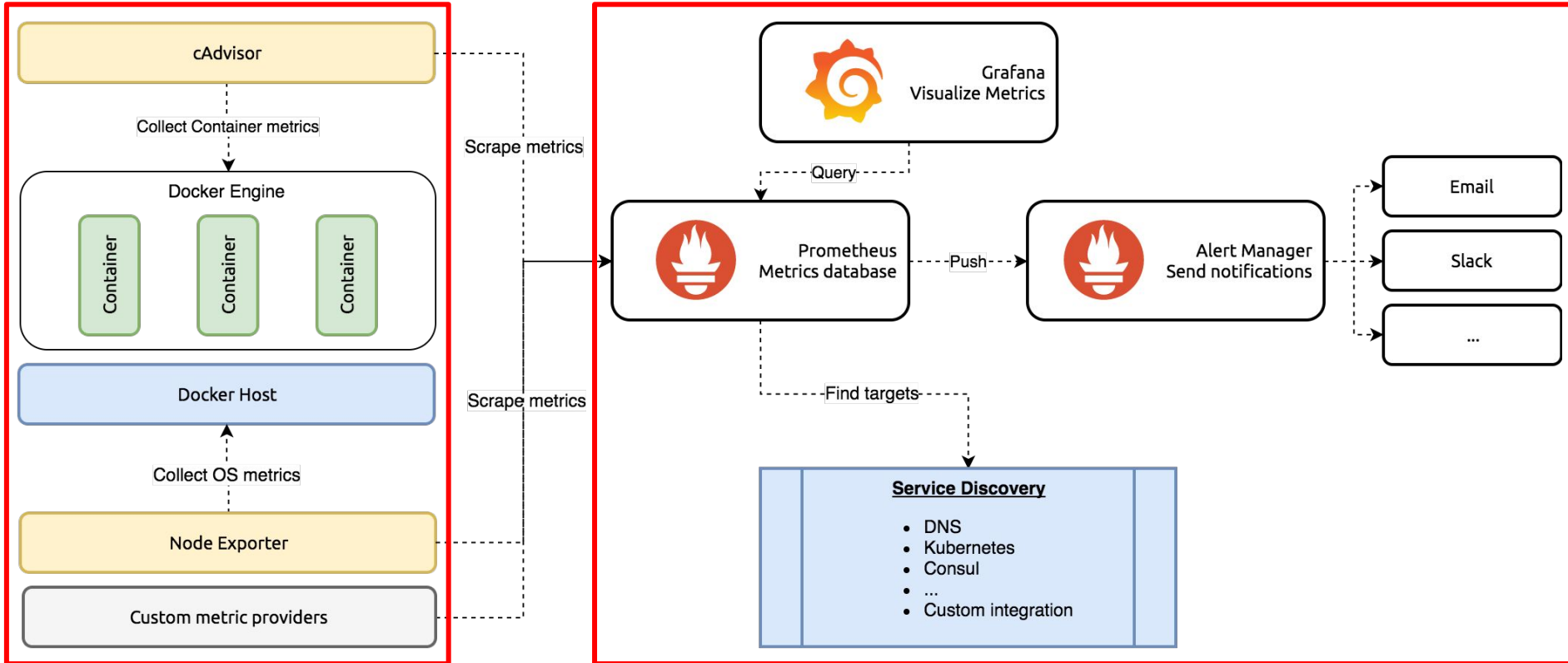
Demo: Application metrics

Code Demo

"Application metrics"

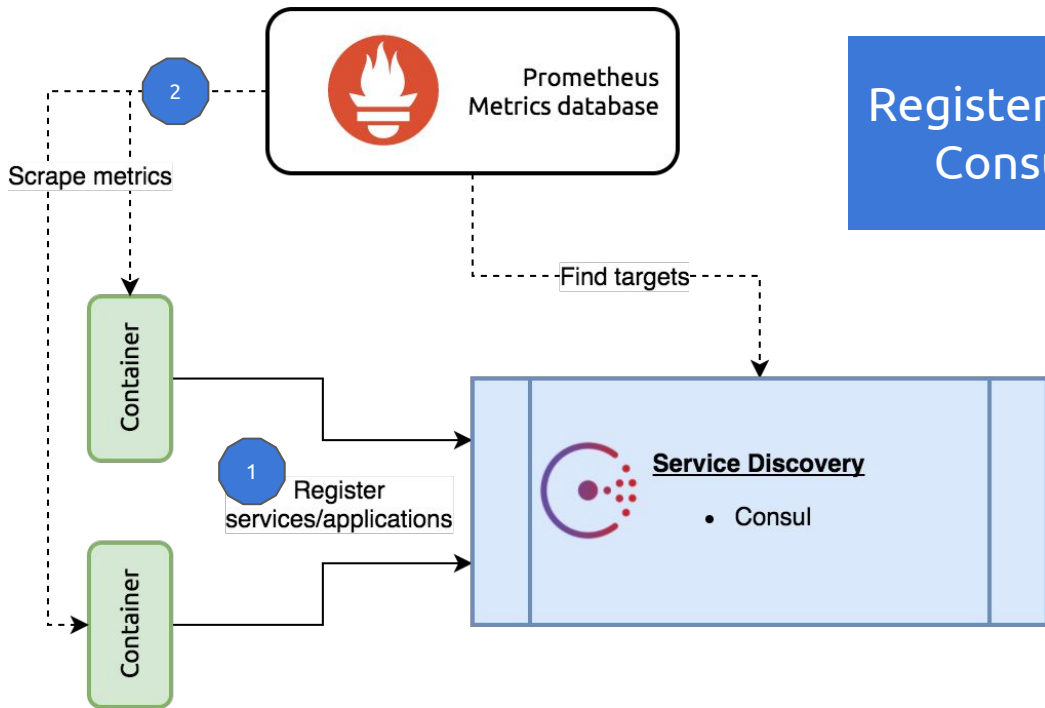# Service Discovery (Consul) Integration

Demo: Consul Integration

Service Discovery

Demo: Consul integration

Register the services with Consul and Monitor

Code Demo

"Consul to the rescue"