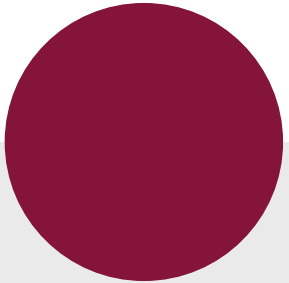
.msg .consulting .solutions .partnership



Ops for Developers
Monitoring with Prometheus for Java Developers

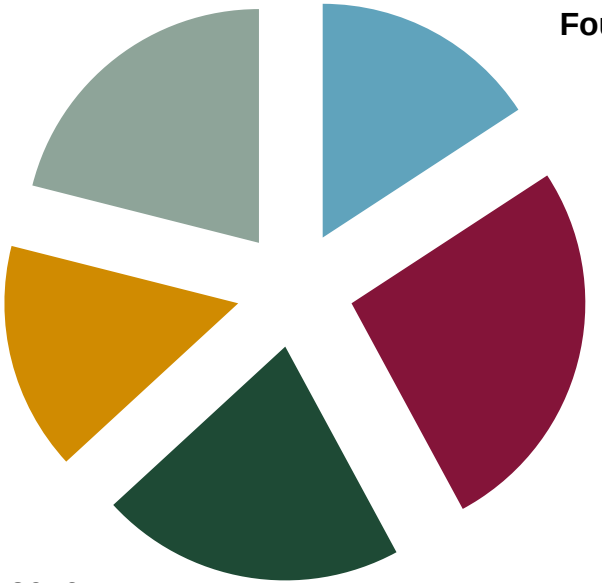
Alexander Schwartz, Principal IT Consultant

Berlin Expert Days / 21 September 2017

Ops for Developers – Monitoring with Prometheus for Java Developers

1. Prometheus Manifesto
2. Setup
3. How to...
4. Prometheus works for Developers (and Ops)

Sponsor and Employer – msg systems ag



Founded 1980

18 offices in Germany

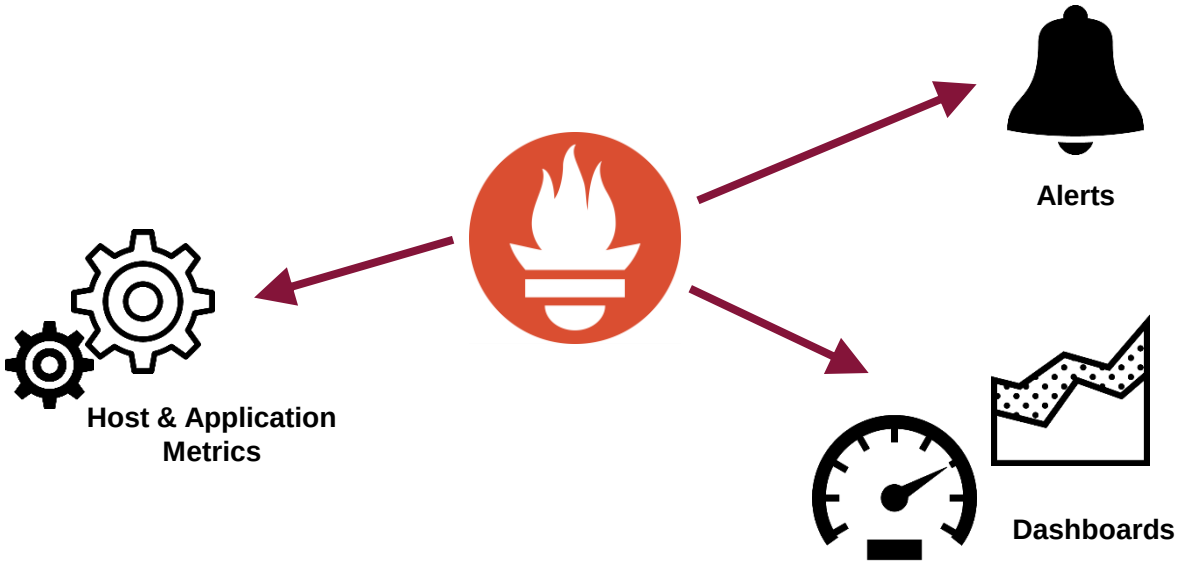
25 Countries

More than 6.000 Employees

812 Million € Turnover 2016

1. **Prometheus Manifesto**
2. Setup
3. How to...
4. Prometheus works for Developers (and Ops)

Monitoring



Prometheus is a Monitoring System and Time Series Database

Prometheus is an opinionated solution

for

instrumentation, collection, storage
querying, alerting, dashboards, trending

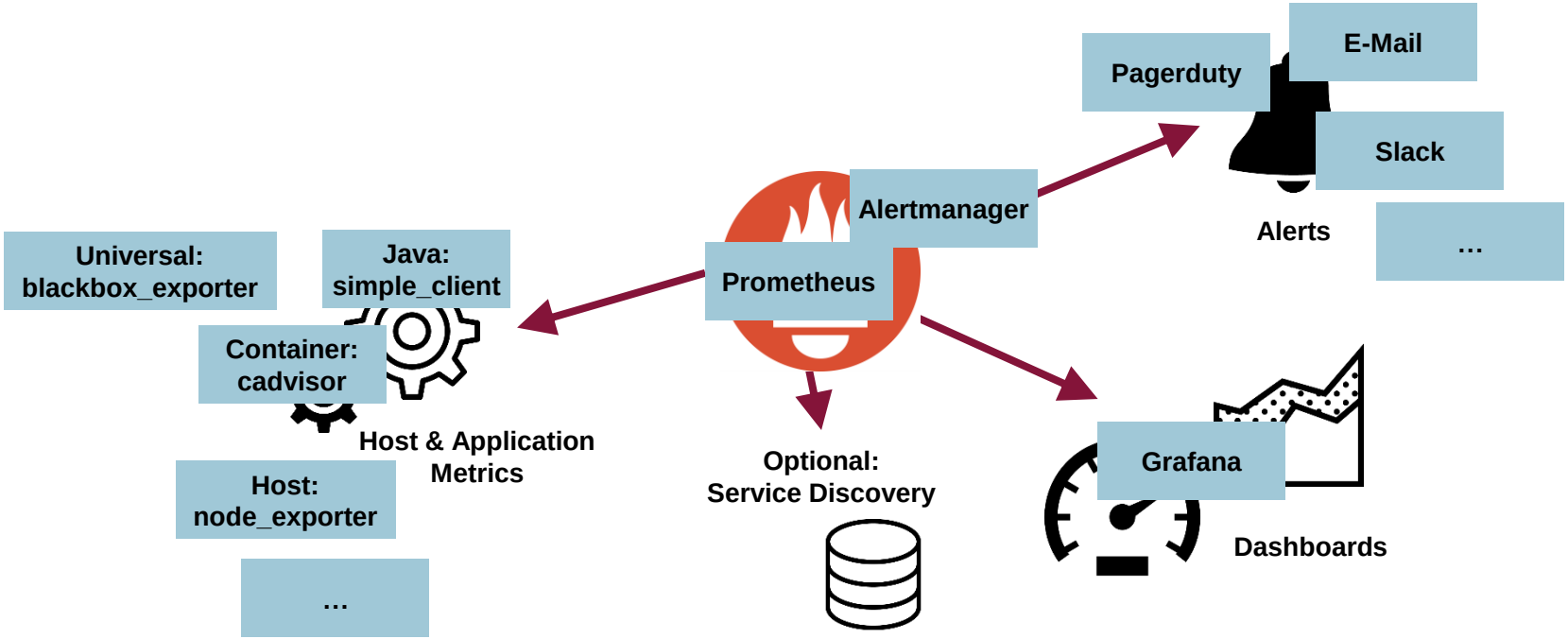
Prometheus values …

operational systems monitoring **(not only) for the cloud**	over	raw logs and events, tracing of requests, magic anomaly detection, accounting, SLA reporting
simple single node **w/ local storage for a few weeks**	over	horizontal scaling, clustering, multitenancy
configuration files	over	Web UI, user management
pulling data from single processes	over	pushing data from processes aggregation on nodes
NoSQL query & data massaging **multidimensional data** **everything as float64**	over	point-and-click configurations data silos complex data types

1. PromCon 2016: Prometheus Design and Philosophy - Why It Is the Way It Is - Julius Volz
   https://youtu.be/4DzoajMs4DM / https://goo.gl/1oNaZV

1. Prometheus Manifesto
2. **Setup**
3. How to...
4. Prometheus works for Developers (and Ops)

Technical Building Blocks



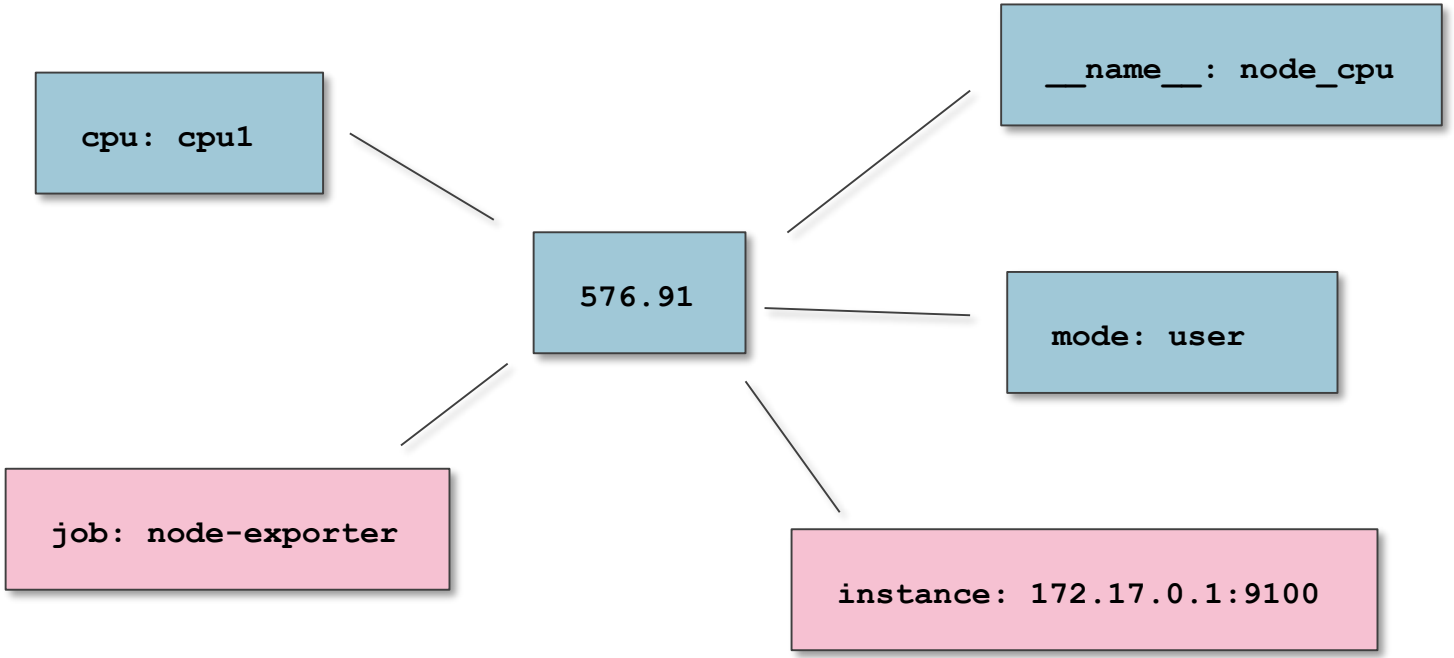
Targets as configured in Prometheus Configuration

```
scrape_configs:
 - job_name: 'node-exporter'
    scrape_interval: 5s
    static_configs:
      - targets: ['172.17.0.1:9100']
```

CPU Metric as exported by the Node Exporter

```
# HELP node_cpu Seconds the cpus spent in each mode.
# TYPE node_cpu counter
node_cpu{cpu="cpu0",mode="guest"} 0
node_cpu{cpu="cpu0",mode="idle"} 4533.86
node_cpu{cpu="cpu0",mode="iowait"} 7.36
...
node_cpu{cpu="cpu0",mode="user"} 445.51
node_cpu{cpu="cpu1",mode="guest"} 0
node_cpu{cpu="cpu1",mode="idle"} 4734.47
...
node_cpu{cpu="cpu1",mode="iowait"} 7.41
node_cpu{cpu="cpu1",mode="user"} 576.91
...
```

Multidimensional Metric as stored by Prometheus



Calculations based on metrics

Metric:

node_cpu: Seconds the CPUs spent in each mode (Type: Counter).

What percentage of a CPU is used per core?

```
1 - rate(node_cpu{mode='idle'} [5m])
```

What percentage of a CPU is used per instance?

```
avg by (instance) (1 - rate(node_cpu{mode='idle'} [5m]))
```

1. Prometheus Manifesto
2. Setup
3. **How to...**
4. Prometheus works for Developers (and Ops)

Information about your node

Presented by: node_exporter

Free disk space:

Variable: node_filesystem_free

Expression: node_filesystem_free{fstype =~ '(xfs|vboxsf)', device !~ '/dev/mapper/.*' }

Additional Options: Axis / Left Y -> Unit bytes

Percent free:

Variables: node_filesystem_free, node_filesystem_size

Expression: node_filesystem_free / node_filesystem_size {fstype =~ '(xfs|vboxsf)'}

Information about your JVM

Presented by: Java simple_client

RAM Usage of Java VM:

Variable: jvm_memory_bytes_used

Expressions:
```
irate(container_cpu_usage_seconds_total [30s])
sum by (instance, job) (jvm_memory_bytes_used)
sum by (instance, job) (jvm_memory_bytes_committed)
```

CPU seconds used by Garbage Collection:

Variable: jvm_gc_collection_seconds_sum

Expression: sum by (job, instance) (irate(jvm_gc_collection_seconds_sum [10s]))

Test: ab -n 100000 -c 10 http://192.168.23.1:8080/manage/metrics

Information about your JVM

Add a Configuration to Spring Boot to serve standard JVM metrics using a custom URL.

```
@Configuration
public class MetricsApplicationConfig {

    @Bean
    public synchronized ServletRegistrationBean metrics() {
        DefaultExports.initialize();
        return new ServletRegistrationBean(new MetricsServlet(),
                "/manage/metrics");
    }
}
```

Information about your Application Metrics

Presented by: Java simple_client, Dropwizard Metrics/Spring

Timings of a method call:

Java Annotation:	@Timed
Variables:	countedCallExample_snapshot_mean countedCallExample_snapshot_75thPercentile countedCallExample_snapshot_98thPercentile
Test:	ab -n 10000 -c 10 http://192.168.23.1:8080/api/countedCall

Information about your Application Metrics

Add a Configuration to Spring Boot to serve standard JVM metrics using a custom URL.

```
@Configuration
@EnableMetrics(proxyTargetClass = true)
public class MetricsApplicationConfig extends MetricsConfigurerAdapter {

      /* ... */
}
```

Information about your Application Metrics

Add @Timed annotations to any method of any Bean to collect metrics

```
@Component
public class RestEndpoint {

    @Path("countedCall")
    @GET
    @Timed(absolute = true, name = "countedCallExample")
    public Response countedCall() throws InterruptedException {
        /* ... */
        return Response.ok("ok").build();
    }

}
```

Information about your External Interfaces

Presented by: Java simple_client, Hystrix/Spring

Hystrix Metrics:

Java Annotation:	@HystrixCommand
Test:	ab -n 10000 -c 10 http://192.168.23.1:8080/api/externalCall
Variables:	hystrix_command_event_total{event="success", …} hystrix_command_latency_total_seconds_*{command_name="…", …}
Expressions:	irate(hystrix_command_event_total{event="success"} [15s]) irate(hystrix_command_event_total{event="exception_thrown"} [15s]) hystrix_command_latency_total_mean hystrix_command_latency_total_percentile_90 hystrix_command_latency_total_percentile_99

Information about your External Interfaces – Hystrix Metrics

Register the Hystrix Publisher and add @HystrixCommand for resilience and timing of external calls.

```
HystrixPrometheusMetricsPublisher.register();
```

```
@Component
public class ExternalInterfaceAdapter {

    @HystrixCommand(commandKey = "externalCall", groupKey = "interfaceOne")
    public String call() {
        /* ... */
    }
}
```

Information about your Spring Servlet Container

Presented by: your own Java metric provider

Tomcat Connector:

Java Class:	Write your own: TomcatStatisticsCollector
Variables:	tomcat_thread_pool_current_thread_count tomcat_thread_pool_current_threads_busy

Tomcat DB Connection Pool:

Java Class:	Write your own: DatasourceStatisticsCollector
Variables:	tomcat_datasource_active tomcat_datasource_idle tomcat_datasource_max_idle

Information about your Spring Servlet Container

```
public class DatasourceStatisticsCollector extends Collector {

    /* ... */

    @Override
    public List<MetricFamilySamples> collect() {
        /* ... */
        result.add(buildGauge("active", "number of connections in use",
            labelNames, labelValues, tomcatDS.getActive()));
        return result;
    }

}
```

```
new DatasourceStatisticsCollector(dataSource).register();
```

Information about your Vert.x application

Presented by: Java Simple Client for Vert.x

Internal Event Bus:

Variables: vertx_eventbus_messages_sent_total
vertx_eventbus_messages_pending
vertx_eventbus_messages_delivered_total
vertx_eventbus_messages_reply_failures_total

HTTP Server metrics:

Variables: vertx_http_servers_requests_count
vertx_http_servers_open_netsockets

Test: ab -n 100000 -c 100 http://192.168.23.1:8081/manage/metrics

Information about your Vert.x application

```
// During Setup
vertx = Vertx.vertx(new VertxOptions().setMetricsOptions(
            new DropwizardMetricsOptions()
                .setRegistryName("vertx")
                .addMonitoredHttpClientEndpoint(
                    new Match().setValue(".*").setType(MatchType.REGEX))
                .setEnabled(true)
        ));

DefaultExports.initialize();
new DropwizardExports(SharedMetricRegistries.getOrCreate("vertx")).register();

// When starting up Routes and a HTTP Server
final Router router = Router.router(vertx);
router.route("/metrics").handler(new MetricsHandler());
```

Federation of Prometheus

Any Metric can be exported to other Prometheus instances

http://localhost/prometheus/federate?match[]={job=%22prometheus%22}

Alerting with Prometheus

Any expression can be used for alerting

```
ALERT HDD_Alert_warning
  IF (1 - node_filesystem_free{mountpoint=~".*"} / node_filesystem_size{mountpoint=~".*"}) * 100 > 70
  FOR 5m
  LABELS {severity="warning"}
  ANNOTATIONS  {summary=“High disk usage on {{ $labels.instance }}: filesystem {{$labels.mountpoint}}
                        more than 70 % full."}
```

1. Prometheus Manifesto
2. Setup
3. How to...
4. **Prometheus works for Developers (and Ops)**

Prometheus is “friendly tech” in your environment

Team friendly

- Every team can run its own Prometheus instance to monitor their own and neighboring systems
- Flexible to collect and aggregate the information that is needed

Coder and Continuous Delivery friendly

- All configurations (except dashboard) are kept as code and are guarded by version control
- Changes can be tested locally and easily staged to the next environment

Simple Setup

- Go binaries for *prometheus* and *alertmanager* available for major operating systems
- Client libraries for several languages available (also adapters to existing metrics libraries)
- Several existing exporters for various needs

Links

Prometheus:
https://prometheus.io

Java Simple Client
https://github.com/prometheus/client_java

Hystrix
https://github.com/Netflix/Hystrix

Prometheus Hystrix Metrics Publisher
https://github.com/ahus1/prometheus-hystrix

Dropwizard Metrics
http://metrics.dropwizard.io

Prometheus on Kubernetes @ fabric8
https://github.com/fabric8io/fabric8-devops

Julius Volz @ PromCon 2016
Prometheus Design and Philosophy
- Why It Is the Way It Is
https://youtu.be/4DzoajMs4DM
https://goo.gl/1oNaZV

CAdvisor
https://github.com/google/cadvisor

@ahus1de

Alexander Schwartz
Principal IT Consultant

+49 171 5625767
alexander.schwartz@msg-systems.com

@ahus1de

msg systems ag (Headquarters)
Robert-Buerkle-Str. 1, 85737 Ismaning
Germany

www.msg-systems.com

.msg .consulting .solutions .partnership