



Kubernetes and microservice monitoring

NMaaS Use case

Łukasz Łopatowski (PSNC)
Vojdan Kjorveziroski (UKIM)
GN4-3 WP6 Task 3

6th SIG-PVM Meeting, Dublin, 3 July 2018

www.geant.org

NMaaS: Network Management as a Service

- Enables management and monitoring of client networks through **on-demand deployment** of network management tools in the cloud infrastructure
- Who is NMaaS for?
 - Small and Emerging NRENs
 - Campuses
 - Small Organisations
 - Distributed research projects
- Work started in GN4-2 JRA2 T5, continued in GN4-3 WP6 T3

NMaaS: Network Management as a Service

- Web-based Network Management application market
- On-demand deployment of containerized applications in the cloud
- Current tools portfolio available via NMaaS

- Oxidized 
- LibreNMS 
- NAV 
- Prometheus 
- Grafana 
- OpenNTI 

NMaaS is deployed in K8s as well

- Underlying container orchestrator evolution



Kubernetes

- Kubernetes (K8s) is an open-source system for automating deployment, scaling, and management of containerized applications
- Groups containers that make up an application into logical units (**Pods**) for easy management and discovery

Service discovery and load balancing

No need to modify your application to use an unfamiliar service discovery mechanism. Kubernetes gives pods their own IP addresses and a single DNS name for a set of pods, and can load-balance across them.

Self-healing

Restarts containers that fail, replaces and reschedules containers when nodes die, kills containers that don't respond to your user-defined health check, and doesn't advertise them to clients until they are ready to serve.

Horizontal scaling

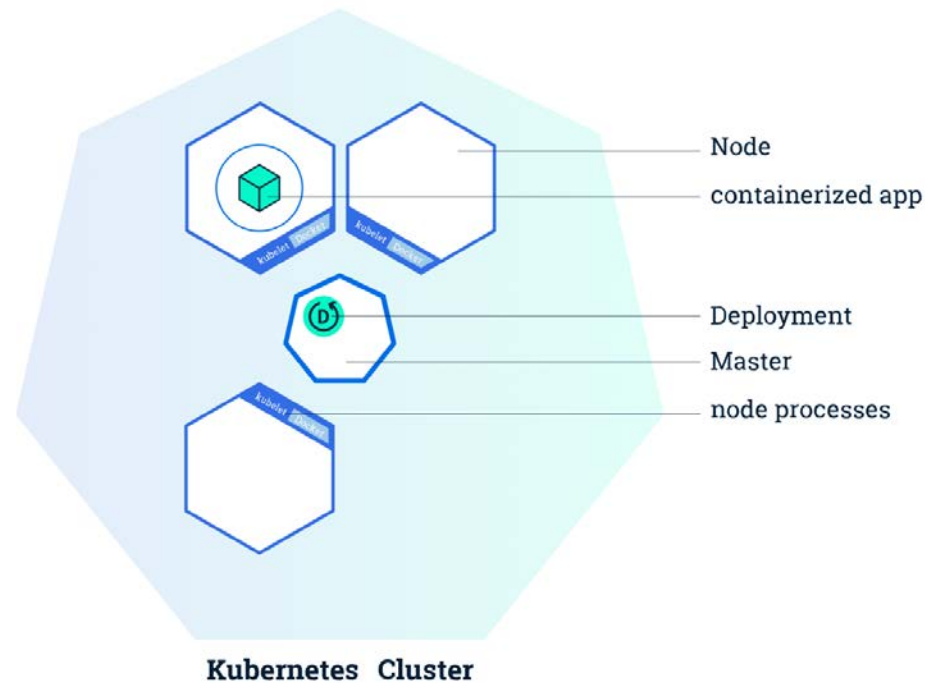
Scale your application up and down with a simple command, with a UI, or automatically based on CPU usage.

Storage orchestration

Automatically mount the storage system of your choice, whether from local storage, a public cloud provider such as [GCP](#) or [AWS](#), or a network storage system such as NFS, iSCSI, Gluster, Ceph, Cinder, or Flocker.

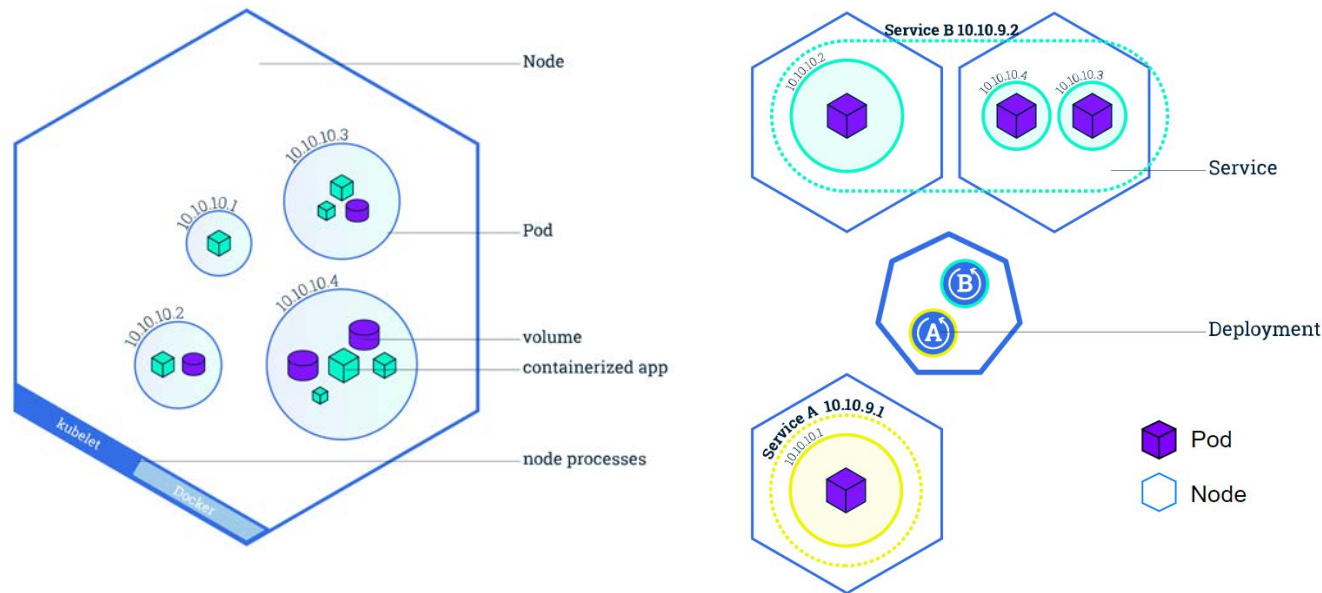
Kubernetes

- Deployment resource instructs Kubernetes how to create and update instances of application
- Kubernetes master schedules application instances onto individual Nodes in the cluster




Kubernetes

- If the Node hosting an instance goes down or is deleted, the Deployment controller replaces the instance with an instance on another Node in the cluster
- **kubectl** uses the Kubernetes API to interact with the cluster to create and manage Deployments



NMaaS deployment infrastructure

- Kubernetes cluster
 - 3 master nodes
 - 3 worker nodes
-  **ceph** cluster for persistence storage
 - 3 monitor nodes
 - 3 storage nodes
- OpenVPN server VM (one per client)

What do we need to monitor?

- Kubernetes cluster internal resource utilization (CPU and memory) and the attached CEPH cluster (available storage space)
- NMaaS components availability (service KPI)
- Client application instances availability

Prometheus-Operator project supports all of the above

Prometheus-Operator

- Prometheus-Operator is a **Kubernetes native** adaptation of the original open-source **Prometheus** monitoring framework, extended to include additional software in the installation, such as **Alertmanager** and **Grafana**
- The *stable/prometheus-operator* Helm chart combines the Prometheus Operator with a collection of manifests to help getting started with monitoring Kubernetes itself and applications running on top of it

Prometheus-Operator

- Configuration of the operator is done through CustomResourceDefinitions, custom YAML files that are sent to the Kubernetes API service, just like any other Kubernetes object, such as a Deployment, Service or Ingress
- Adding a new service that needs to be monitored is a matter of only creating a new ServiceMonitor resource

```
apiVersion: monitoring.coreos.com/v1
kind: ServiceMonitor
metadata:
  labels:
    app: ingress-nginx
    release: prometheus-operator
  name: ingress-nginx
  namespace: nmaas-system
spec:
  endpoints:
    - port: metrics
  jobLabel: ingress-nginx
  namespaceSelector:
    matchNames:
      - ingress-nginx
  selector:
    matchLabels:
      app: ingress-nginx-metrics
```



Prometheus-Operator: Prometheus

- Polls a given endpoint (e.g. `http://app.example.com/metrics`) at a predefined interval and parses the exposed metrics
- Metrics are stored either in local storage or a more robust, external, database, such as InfluxDB
- Metrics can be queried and later on visualized (e.g with Grafana)
- In order to query the metrics, the PromQL querying language is used, which has its own syntax, as well as built-in functions
- Prometheus web interface itself has a rough ability to visualize metrics in an ad hoc manner



Prometheus-Operator: Prometheus

- Monitored service is required to publish the metrics on a given endpoint which can be accessed with a simple HTTP GET request by Prometheus.
- An example of published metrics:

```
nginx_ingress_controller_response_size_count{controller_class="nginx",controller_namespace="ingress-nginx",controller_pod="ingress-nginx-controller-2bwvg",host=
# TYPE nginx_ingress_controller_ssl_expire_time_seconds gauge
nginx_ingress_controller_ssl_expire_time_seconds{class="nginx",host="access.local",namespace="ingress-nginx"} 1.568708703e+09
nginx_ingress_controller_ssl_expire_time_seconds{class="nginx",host="adminer.local",namespace="ingress-nginx"} 1.568701819e+09
nginx_ingress_controller_success{controller_class="nginx",controller_namespace="ingress-nginx",controller_pod="ingress-nginx-controller-2bwvg"} 53
process_cpu_seconds_total 42049.05
```



Prometheus-Operator: Grafana

- The Grafana instance comes preconfigured with multiple dashboards that aim to give insight into the current state of the cluster, for example:
 - cluster wide usage of memory and CPU across all the Kubernetes nodes,
 - individual resource usage for distinct pods,
 - percentage of the allocated resources used for the different applications
- All of these dashboards can be further extended by writing custom PromQL queries and adding new visualizations.



Prometheus-Operator: Alertmanager

- Alerts the cluster administrator when any given threshold is reached
- Alert conditions can be defined using custom PromQL queries that are periodically evaluated
- All alerts can be previewed from the web interface at all times
- If a suitable notification channel has been configured, an alert will be communicated to the cluster administrator
- Currently NMaaS related alerts are notified using dedicated Slack channel

Prometheus-Operator: Alertmanager

```

receivers:
- name: 'slack'
  slack_configs:
  - channel: '<SLACK_CHANNEL>'
    api_url: '<SLACK_WEBHOOK_URL>'
    username: '{{ template "slack.default.username" . }}'
    color: '{{ if eq .Status "firing" }}danger{{ else }}good{{ end }}'
    title: '{{ .Status | toUpper }}{{ if eq .Status "firing" }}:{{ .Alerts.Firing | len }}{{ end }} Prometheus Event Notification'
    title_link: '{{ template "slack.default.titlelink" . }}'
    pretext: '{{ .CommonAnnotations.summary }}'
    text: |-
      {{ range .Alerts }}
        {{- if .Annotations.summary }}*Alert:* {{ .Annotations.summary }} - `{{ .Labels.severity }}`{{- end }}
        *Description:* {{ .Annotations.description }}{{ .Annotations.message }}
        *Graph:* <{{ .GeneratorURL }}|:chart_with_upwards_trend:>{{ if or .Annotations.runbook .Annotations.runbook_url }} *Runbook:*
        *Details:*
          {{ range .Labels.SortedPairs }} • *{{ .Name }}*: `{{ .Value }}`
          {{ end }}
        {{ end }}
      fallback: '{{ template "slack.default.fallback" . }}'
      icon_emoji: '{{ template "slack.default.iconemoji" . }}'
      icon_url: '{{ template "slack.default.iconurl" . }}'
      send_resolved: true
  
```

APP 05:47

nmaas-platform has unavailable replicas

[QALAB2] [FIRING:1] Prometheus Event Notification

Alert: nmaas-platform has unavailable replicas - **critical**

Description: nmaas-platform in namespace nmaas-system, pod prometheusoperator-kube-state-metrics-6d6cbb848f-zr5fh has unavailable replicas

Graph:

Details:

- **alertname:** PodDown
- **deployment:** nmaas-platform
- **endpoint:** http
- **instance:** 10.233.70.105:8080
- **job:** kube-state-metrics
- **namespace:** nmaas-system
- **pod:** prometheusoperator-kube-state-metrics-6d6cbb848f-zr5fh
- **prometheus:** monitoring/prometheusoperator-prometh-prometheus
- **service:** prometheusoperator-kube-state-metrics
- **severity:** critical

[Show less](#)

APP 17:41

[QALAB1] [RESOLVED] Prometheus Event Notification

Description: 76% throttling of CPU in namespace pllabb for container nmaas-librenms in pod b32ad05b-f12d-458f-94d5-f046cb183314-547bb6f684-k2v2p.

Graph: **Runbook:**

Details:

- **alertname:** CPUThrottlingHigh
- **container_name:** nmaas-librenms
- **namespace:** pllabb
- **pod_name:** b32ad05b-f12d-458f-94d5-f046cb183314-547bb6f684-k2v2p
- **prometheus:** nmaas-system/prometheusoperator-prometh-prometheus
- **severity:** warning

[Show less](#)

What we actually monitor?

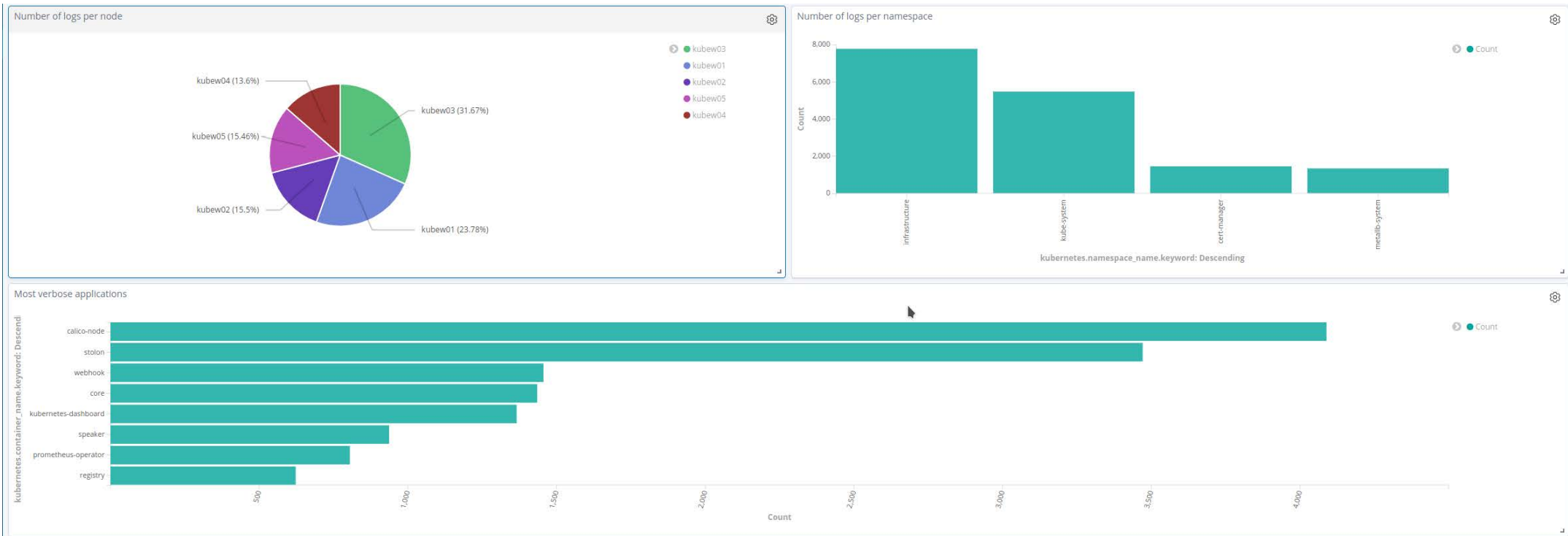
- Internal cluster resources utilization (using pre-configured Grafana dashboards)
 - **CPU and memory** on cluster, node, pod and container level
 - **Ingress Controller** (number of requests per second to different applications being hosted, the amount of bandwidth used, certificate expiry time, etc.)
- Pods availability using custom Grafana configuration
 - Based on output from Liveness and Readiness Probes exposed by **applications running within pods**
 - Custom calculation of availability level over time
- External services
 - **Ceph storage cluster** – native support for Prometheus with the ability to publish a Prometheus metrics endpoint
 - **OpenVPN server** – statistics file published by OpenVPN is parsed and outputted as a Prometheus endpoint by using the `openvpn_exporter` (https://github.com/kumina/openvpn_exporter).

Additional feature: Application logs monitoring

- Monitoring of the logs per pod should be implemented using either the ELK (Elasticsearch, Logstash / Fluentd, Kibana) stack
- Raw logs from all pods / microservices will be persisted to Elasticsearch from where they can later be previewed and visualized using Kibana (text oriented visualization tool)
- Individual log files of the Docker containers on each of the Kubernetes nodes are enriched with additional information, such as the node where the container is run, the deployment that it is part of, etc.
- Possibility to trigger alerts based on the content of the logs (additional tools required)

Additional feature: Application logs monitoring

Example visualization of log files statistics



More Information

Main GÉANT web page:

- geant.org/NMaaS

Contact e-mail address of NMaaS team:

- nmaas@lists.geant.org

NMaaS instance:

- <https://nmaas.geant.org/>





Thank you

Any questions?

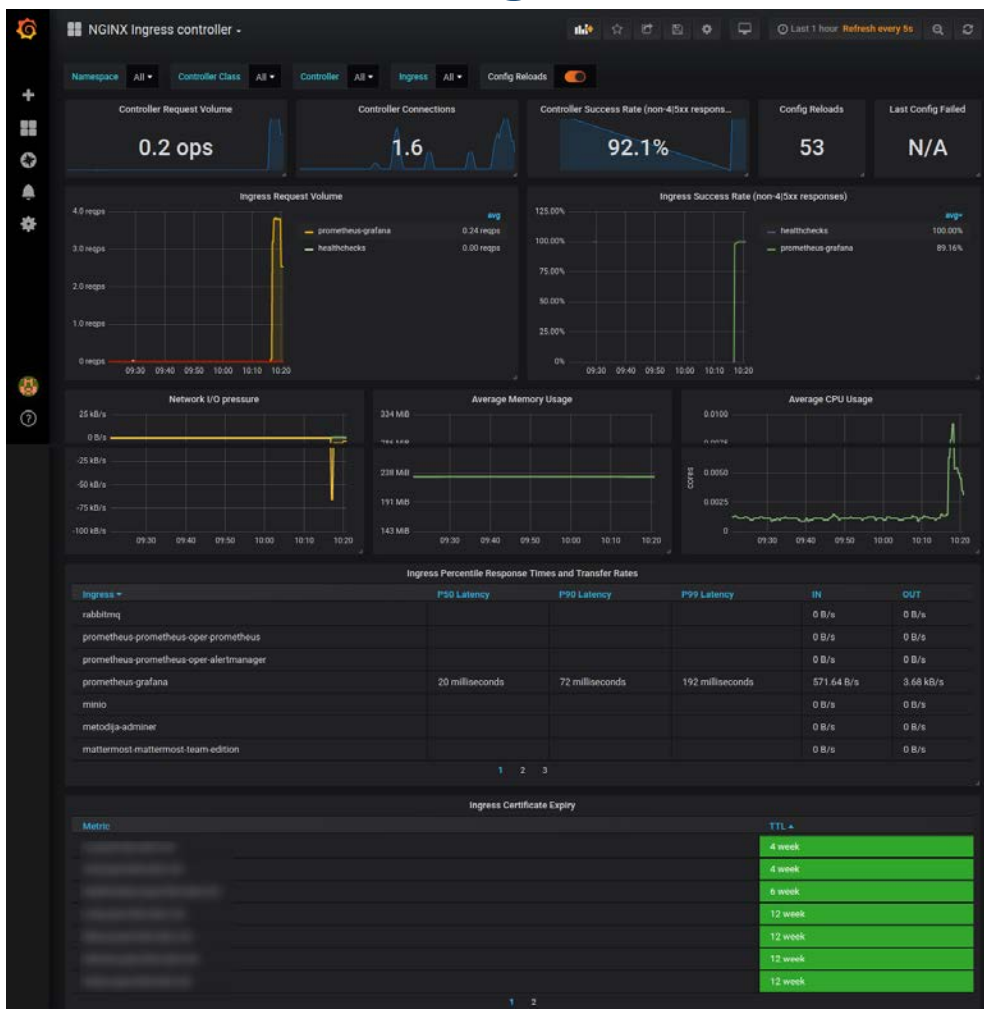
www.geant.org



© GÉANT Association on behalf of the GN4 Phase 3 project (GN4-3).
The research leading to these results has received funding from
the European Union's Horizon 2020 research and innovation
programme under Grant Agreement No. 856726 (GN4-3).

Backup slides: Example Grafana screenshots

Ingress Controller monitoring dashboard view



Ceph storage cluster monitoring dashboard view

