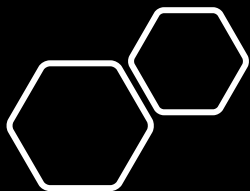


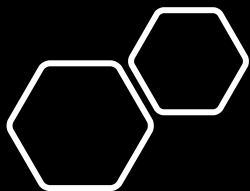
# Monitoring à l'ère du DevOps

Ulrich MONJI



# Plan

- Présentation de la formation
- Rappels sur les microservices
- Les enjeux du monitoring
- Prometheus et Grafana
- Monitoring des hôtes / OS
- Monitoring du Cluster
- Monitoring des Conteneurs
- Monitoring des applications
- Alerting
- Operators
- Cloud Solution
- Miniprojet
- Conclusion



# Plan

- **Présentation de la formation**
- Rappels sur les microservices
- Les enjeux du monitoring
- Prometheus et Grafana
- Monitoring des hôtes / OS
- Monitoring du Cluster
- Monitoring des Conteneurs
- Monitoring des applications
- Alerting
- Operators
- Cloud Solution
- Miniprojet
- Conclusion

# Présentation de la formation (1/3) : Ulrich MONJI

---

- Atos-Worldline - Ingénieur Système
    - Build et Run de plateforme Cloud
    - Virtualisation - Conteneurisation - Automatisation
    - Comptes clients: Carrefour, Auchan, ARJEL, SAMU
  - Adneom - Consultant IT
  - Groupe SII - Consultant IT (Cloud/Devops)
    - Consultant chez Orange France
    - Migration d'une application monolithique en microservice
  - Formateur et blogueur chez eazytraining
- 



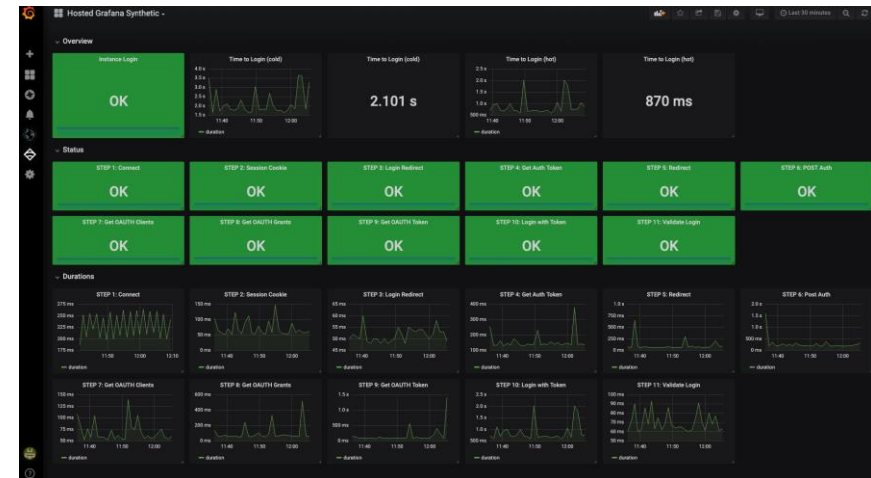
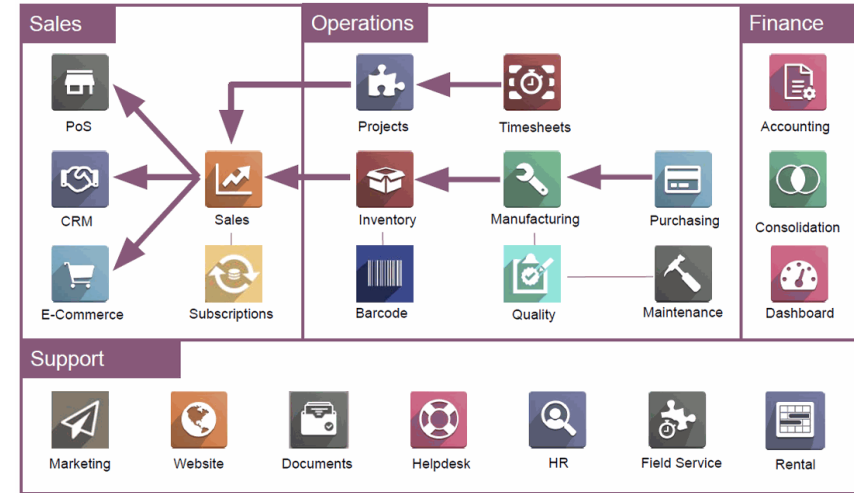


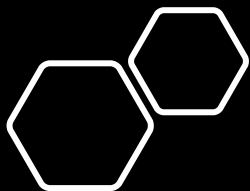
# Présentation de la formation (2/3): Prérequis



# Présentation de la formation (3/3): Projet fil Rouge

- De monolithique vers micro-service
- Monitoring infra et applicative
- Vers le Cloud ?

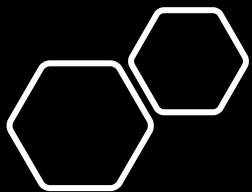




# Plan

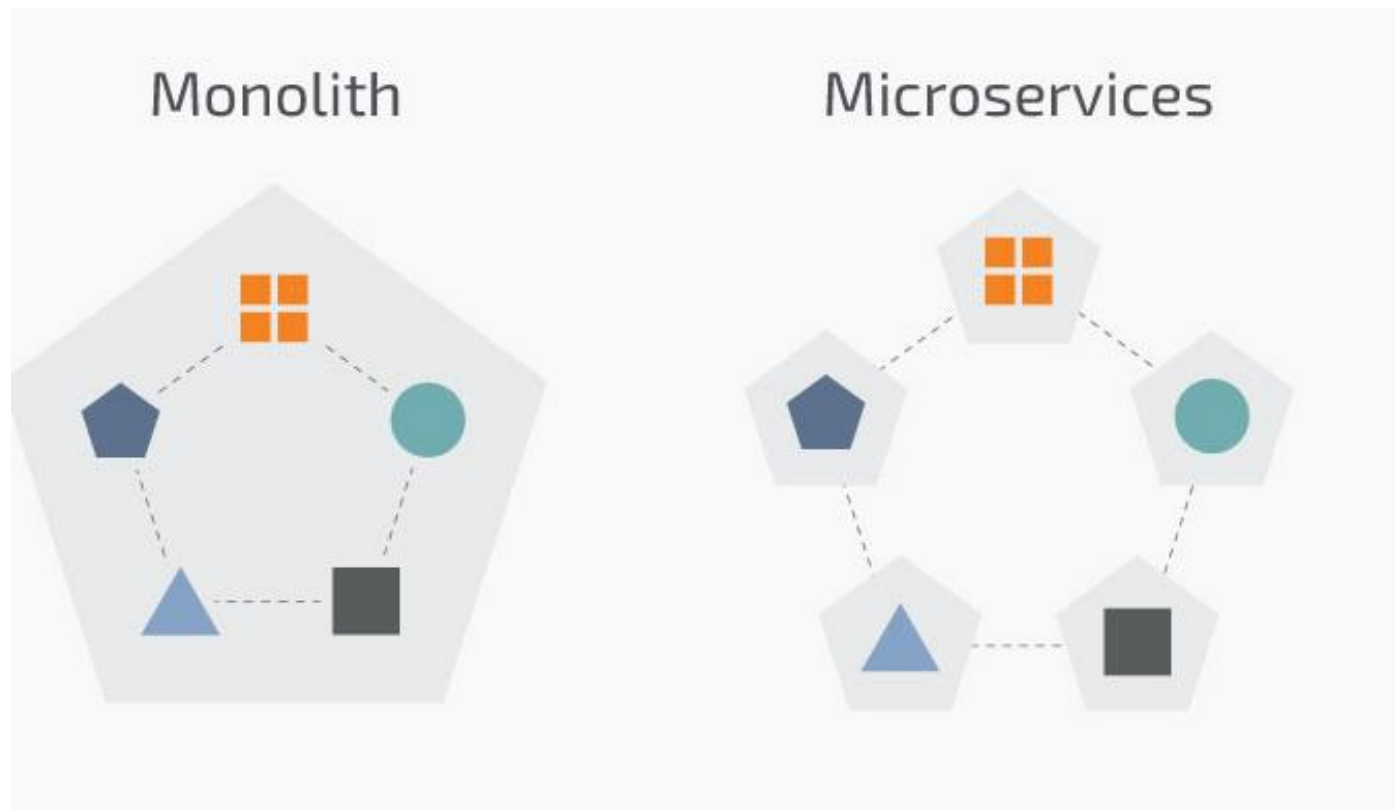
- Présentation de la formation
- **Rappels sur les microservices**
- Les enjeux du monitoring
- Prometheus et Grafana
- Monitoring des hôtes / OS
- Monitoring du Cluster
- Monitoring des Conteneurs
- Monitoring des applications
- Alerting
- Operators
- Cloud Solution
- Miniprojet
- Conclusion



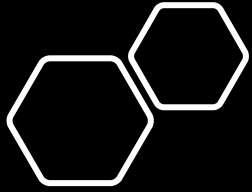


## Rappels sur les microservices (1/5): Docker (1/2)

- Elasticity
- Availability
- Agility

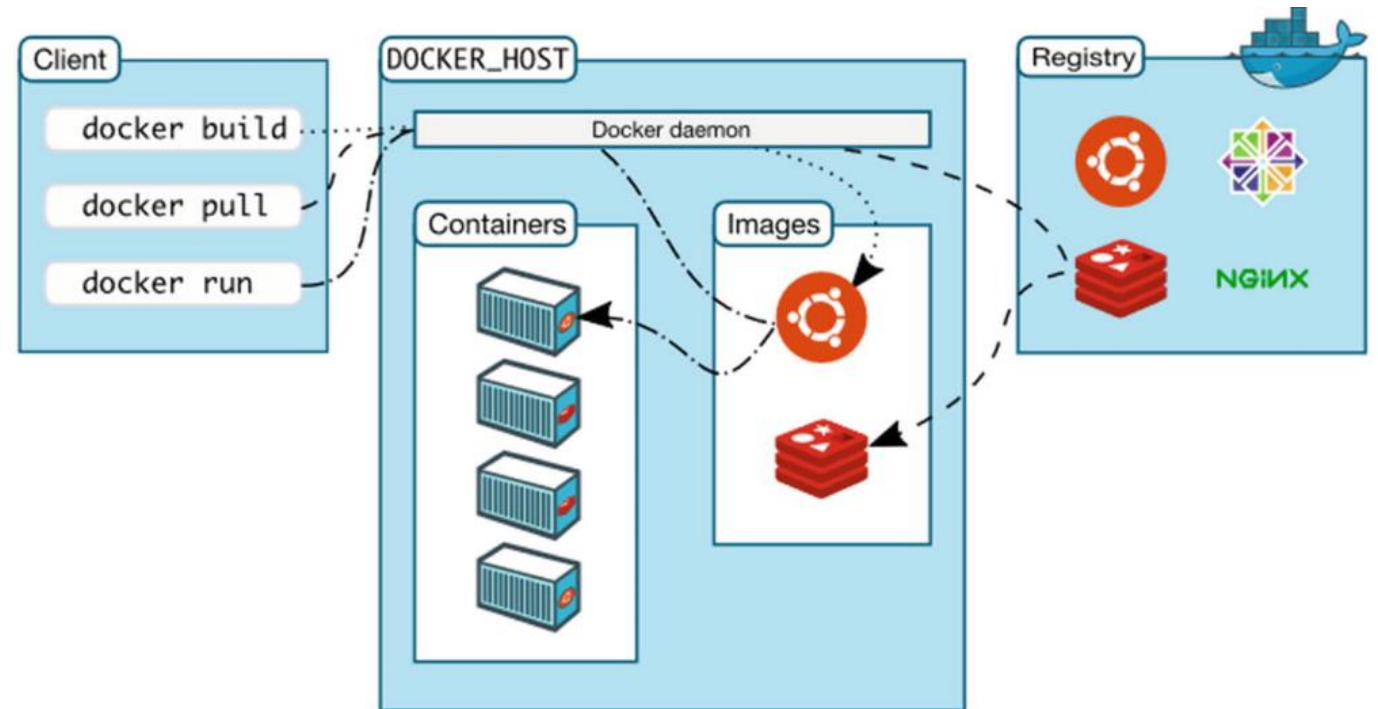


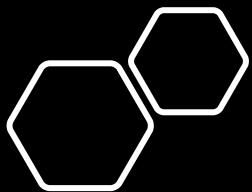




## Rappels sur les microservices (2/5): Docker (2/2)

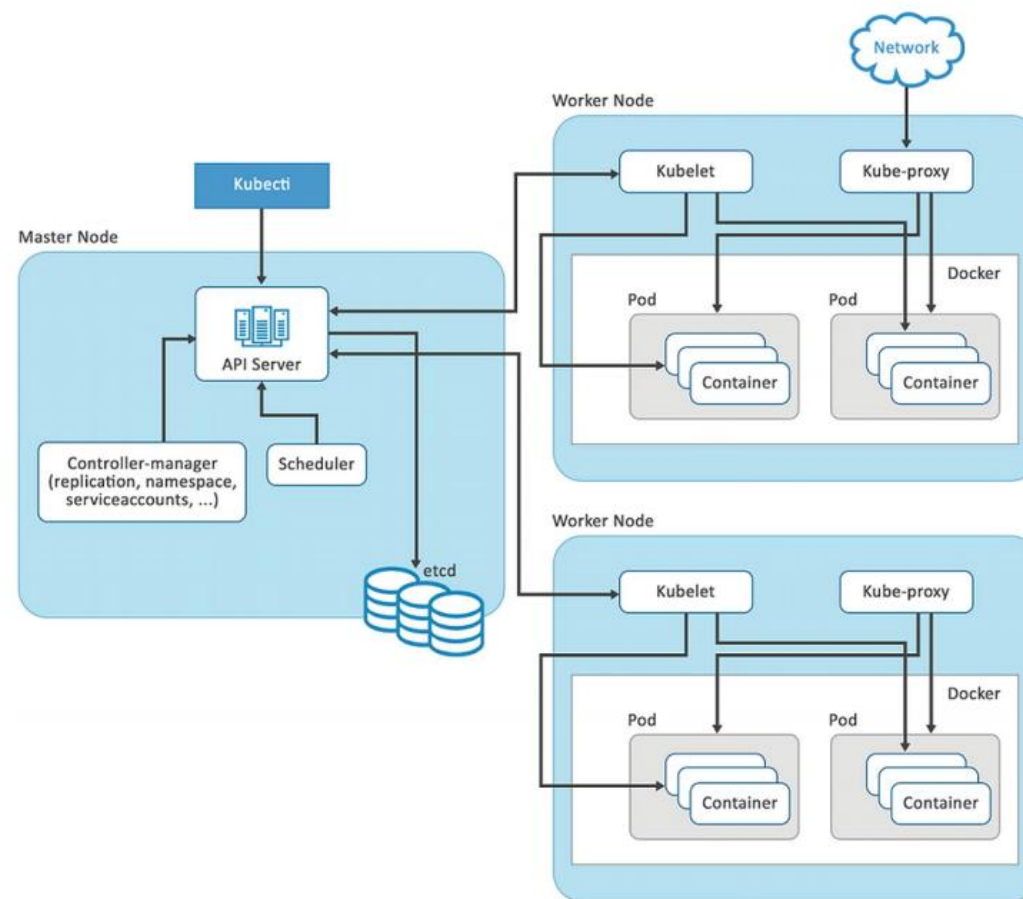
- Build
- Run
- reuse

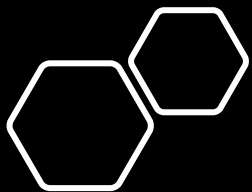




## Rappels sur les microservices (3/5): Kubernetes

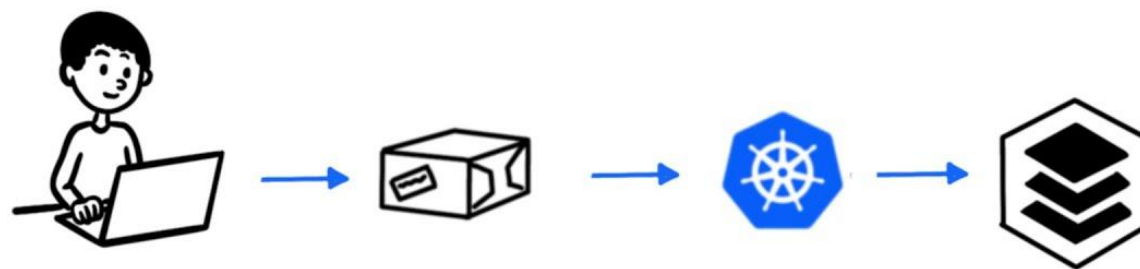
- Scale
- Update
- HA



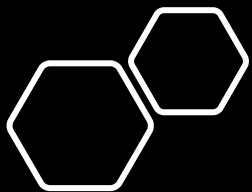


## Rappels sur les microservices (4/5): Helm (1/2)

- Package
- Template
- Share
- Deploy
- Revision

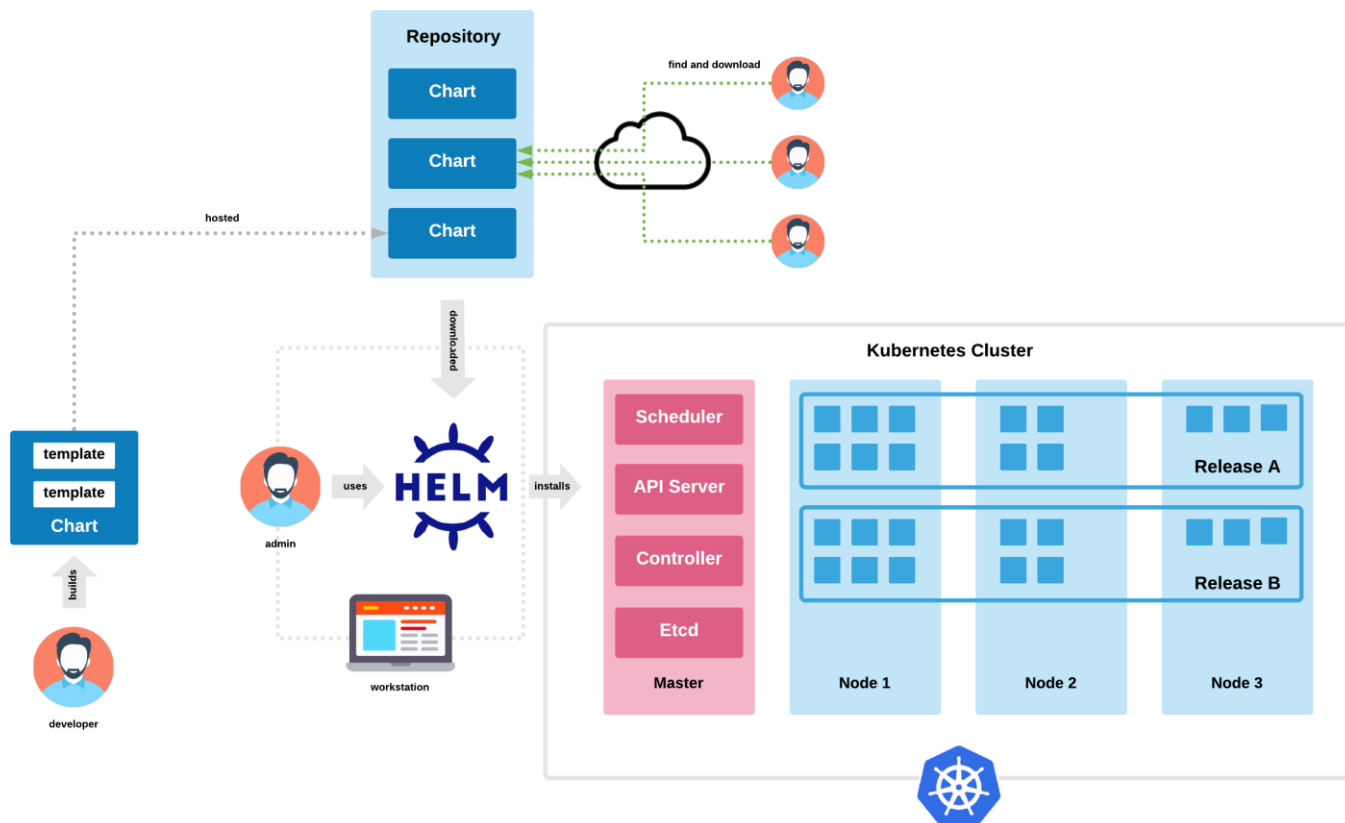


# HELM



## Rappels sur les microservices (5/5): Helm (2/2)

- DevOps
- GitOps
- CI/CD
- Reuse and overwrite

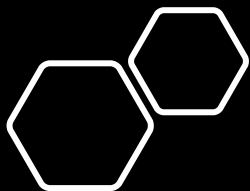


# Lab-0: Environnements de travail

- [EAZYTraining](#) k8s platform (2h)
- [Katakoda](#) 1 master 1 worker
- [Katakoda](#) minikube (1h)
- Votre propre cluster 1 master 1 worker : recommandé
- Votre propre minikube

# Lab-1: Déploiement de l'application

- Installez helm 3 sur votre cluster
- Utilisez le [chart helm](#) de l'application odoo pour le deployer
- Désactivez toutes les options de persistance de données (odoo et postgres)
- Exposez l'application via un service de type nodeport (30069)
- Créez un fichier values.yaml contenant toutes les variables que vous avez surchargées et poussez le sur un git dans un dossier que vous appellerez lab-1
- Vérifiez que l'application est bien accessible via le service créé



# Plan

- Présentation de la formation
- Rappels sur les microservices
- **Les enjeux du monitoring**
- Prometheus et Grafana
- Monitoring des hôtes / OS
- Monitoring du Cluster
- Monitoring des Conteneurs
- Monitoring des applications
- Alerting
- Operators
- Cloud Solution
- Miniprojet
- Conclusion



# Les enjeux du monitoring

## (1/4): Le besoin

### (1/2)

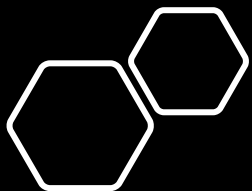
Architecture	Metric Selection Decision Logic	Sample Metrics
<b>Microservice</b> In general, there is one process to track per container.	Where are the new services deployed? What percentage of time is the service reachable? How many requests are enqueued?	Average percentage of time a request-servicing thread is busy. Number of enqueued requests. Percentage of time a service is reachable
<b>Application</b> Multiple microservices running simultaneously constitute an application	Does the database respond quickly? Are the message queues fast enough? How does heap memory usage change over time? Are application services responsive?	Query execution frequency, response time, and failure rate. Response time, failure rate
<b>Container</b> Separate from the underlying process being run within it, containers are also monitored	How responsive are the processes within container? Which images have been deployed? Are specific containers associated with over-utilization of host?	CPU throttle time. Container disk I/O. Memory usage. Network (volume, dropped packets)

# Les enjeux du monitoring

## (2/4): Le besoin

### (2/2)

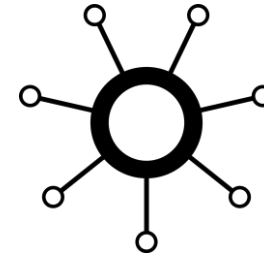
<b>Container Cluster</b> Multiple containers deployed to run as group. Many of the metrics of individual containers can also be summarized.	Are your clusters healthy and properly sized? Can applications be effectively run on fewer nodes?	Percentage of clusters remaining operational compared to those originally deployed
<b>Host</b> Also called a node, multiple hosts can support a cluster of containers	Do changes in utilization indicate a problem with a process or application?	Percentage of total memory capacity in use. Percentage of time CPUs are utilized
<b>Infrastructure</b> Cloud in which hosts are running	How much does it cost to run each service or deployment? What is the ratio of microservices and/or containers per instance?	Network traffic Utilization of databases, storage, and other shared services
<b>End user</b> The users using the application or other applications using APIs.	What is the average web/transaction response time experienced by users or by target application?	Response time. Number and percentage of failed user actions/transactions



## Les enjeux du monitoring (3/4): les micro-services

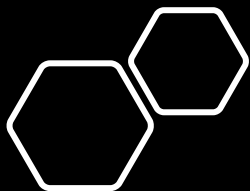
- Cycle de vie court des conteneurs
- Utilisation d'un même micro-service dans différents contextes
- La consommation de l'infrastructure fluctue lors du provisioning
- Plusieurs couches du SI doivent être surveillées
- Multiple Container Frameworks (ECS, ACS, Google ...)

Per Host Metrics Explosion			
Component	# of Metrics for a Traditional Stack	for 10 Container Cluster with 1 Underlying Host	for 10 Container Cluster with 1 Underlying Host
Operating System	100	100	200
Orchestrator	n/a	50	50
Container	n/a	500 (50 per container)	5,000 (50 per container)
Application	50	500 (50 per container)	5,000 (50 per container)
Total # of Metrics	150	1,150	10,250



Les enjeux du monitoring (4/4): Le marché

---



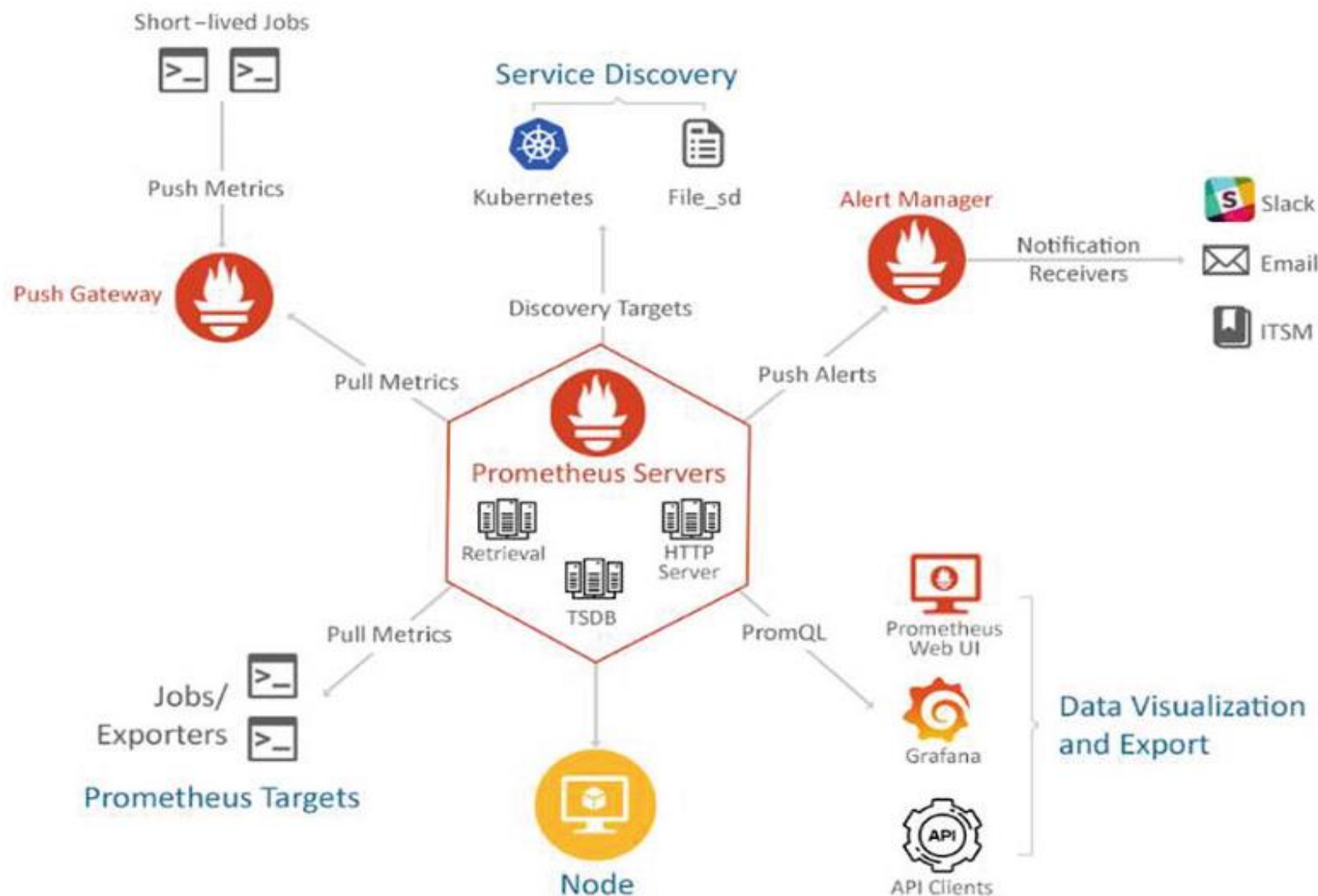
# Plan

- Présentation de la formation
- Rappels sur les microservices
- Les enjeux du monitoring
- **Prometheus et Grafana**
- Monitoring des hôtes / OS
- Monitoring du Cluster
- Monitoring des Conteneurs
- Monitoring des applications
- Alerting
- Operators
- Cloud Solution
- Miniprojet
- Conclusion

# Prometheus et Grafana (1/3): Prometheus (1/2)

- Container monitoring tools
- Développé initialement par [SoundCloud](#)
- Ecrit en grande grande partie en Go
- Fait désormais partie des projet [CNCF](#)
- Key-Value data model
- PromQL pour requêter les données

# Prometheus et Grafana (2/3): Prometheus (2/2)





# Prometheus et Grafana (3/3): Grafana

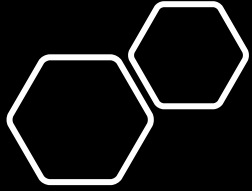
- UI-based Dashboard and reporting tool
- Il permet également de faire l'alerting sur des outils tels que
  - Slack
  - PagerDuty
  - VictorOps
  - OpsGenie
- Il supporte les Time-series database tels que
  - Prometheus
  - Influxdb
  - CloudWatch
  - Graphite
  - OpenTSDB
  - ElasticSearch

# Lab-2: Installation de prometheus

- Créez un namespace **monitoring**
- Déployez Prometheus à l'aide des [sources fournies](#)
- Etudiez les fichiers et déduisez comment accéder à l'application via le nodeport
- Découvrez l'interface
- Allez dans les targets et vérifiez que le target Prometheus est bien présente et up
- Félicitations ! Vous pouvez passer au lab-3

# Lab-3: Installation de grafana

- Déployez Grafana à l'aide du [chart Grafana](#) en version 3.12.1 disponible sur le [helm Communautaire](#) Surchargez les variables du chart en utilisant le fichier values.yaml [fournies dans les sources](#)
- A l'aide du fichier deduisez comment accéder au dashboard et récupérer le mot de passe comme précisé après le déploiement du chart
- Configurez la datasource pour qu'elle pointe sur la target de Prometheus
- Importez le dashboard Prometheus d'id 3662
- Que constatez-vous ?
- Si vous avez des métriques qui s'affichent, alors félicitation, votre installation de Prometheus et Grafana est terminée !



# Plan

- Présentation de la formation
- Rappels sur les microservices
- Les enjeux du monitoring
- Prometheus et Grafana
- **Monitoring des hôtes / OS**
- Monitoring du Cluster
- Monitoring des Conteneurs
- Monitoring des applications
- Alerting
- Operators
- Cloud Solution
- Miniprojet
- Conclusion

# Monitoring des hôtes / OS (1/3): Exporter

## Databases %

- Aerospike exporter
- ClickHouse exporter
- Consul exporter (official)
- Couchbase exporter
- CouchDB exporter
- Druid Exporter
- Elasticsearch exporter
- EventStore exporter
- IoTDB exporter
- KDB+ exporter
- Memcached exporter (official)
- MongoDB exporter
- MongoDB query exporter
- MSSQL server exporter
- MySQL router exporter
- MySQL server exporter (official)
- OpenTSDB Exporter
- Oracle DB Exporter
- PgBouncer exporter
- PostgreSQL exporter
- Presto exporter
- ProxySQL exporter
- RavenDB exporter
- Redis exporter
- RethinkDB exporter
- SQL exporter
- Tarantool metric library
- Twemproxy

## Hardware related

- apcupsd exporter
- BIG-IP exporter
- Bosch Sensortec BMP/BME exporter
- Collins exporter
- Dell Hardware OMSA exporter
- Fortigate exporter
- IBM Z HMC exporter
- IoT Edison exporter
- IPMI exporter
- knxd exporter
- Modbus exporter
- Netgear Cable Modem Exporter
- Netgear Router exporter
- Network UPS Tools (NUT) exporter
- Node/system metrics exporter (official)
- NVIDIA GPU exporter
- ProSAFE exporter
- Ubiquiti UniFi exporter
- Waveplus Radon Sensor Exporter
- Weathergoose Climate Monitor Exporter
- Windows exporter

## APIs

- AWS ECS exporter
- AWS Health exporter
- AWS SQS exporter
- Azure Health exporter
- BigBlueButton
- Cloudflare exporter
- Cryptowat exporter
- DigitalOcean exporter
- Docker Cloud exporter
- Docker Hub exporter
- GitHub exporter
- Gmail exporter
- InstaClustr exporter
- Mozilla Observatory exporter
- OpenWeatherMap exporter
- Pagespeed exporter
- Rancher exporter
- Speedtest exporter
- Tankerkönig API Exporter

## Logging

- Fluentd exporter
- Google's mtail log data extractor
- Grok exporter

## Issue trackers and continuous integration

- Bamboo exporter
- Bitbucket exporter
- Confluence exporter
- Jenkins exporter
- JIRA exporter

## Messaging systems

- Beanstalkd exporter
- EMQ exporter
- Gearman exporter
- IBM MQ exporter
- Kafka exporter
- NATS exporter
- NSQ exporter
- Mirth Connect exporter
- MQTT blackbox exporter
- MQTT2Prometheus
- RabbitMQ exporter
- RabbitMQ Management Plugin exporter
- RocketMQ exporter
- Solace exporter

## Storage

- Ceph exporter
- Ceph RADOSGW exporter
- Gluster exporter
- GPFS exporter
- Hadoop HDFS FSImage exporter
- Lustre exporter
- NetApp E-Series exporter
- ScaleIO exporter
- Tivoli Storage Manager/IBM Spectrum

## HTTP

- Apache exporter
- HAProxy exporter (official)
- Nginx metric library
- Nginx VTS exporter
- Passenger exporter
- Squid exporter
- Tinyproxy exporter
- Varnish exporter
- WebDriver exporter

# Monitoring des hôtes / OS (2/3): Node Exporter (1/2)

- Prometheus exporter
- Il est écrit en Go
- Il permet de récupérer des données/métriques exposées par le noyau Unix/Linux concernant
  - l'OS
  - le hardware

# Monitoring des hôtes / OS (3/3): node exporter (2/2)

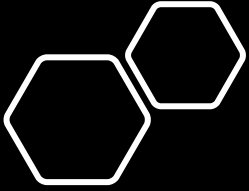
Arp	Exposes ARP statistics from <code>/proc/net/arp</code>	Linux
Boottime	Exposes system boot time derived from <code>kern.boottime sysctl</code>	Darwin, Dragonfly, FreeBSD, NetBSD, OpenBSD, Solaris
Cpu	Exposes CPU statistics	Darwin, Dragonfly, FreeBSD, Linux, Solaris
Cpufreq	Exposes CPU frequency statistics	Linux, Solaris
Diskstats	Exposes disk I/O statistics	Darwin, Linux, OpenBSD
Filesystem	Exposes filesystem statistics, such as disk space used	Darwin, Dragonfly, FreeBSD, Linux, OpenBSD
Hwmon	Exposes hardware monitoring and sensor data from <code>/sys/class/hwmon/</code>	Linux
Meminfo	Exposes memory statistics	Darwin, Dragonfly, FreeBSD, Linux, OpenBSD

Netclass	Exposes network interface info from <code>/sys/class/net/</code>	Linux
netdev	Exposes network interface statistics such as bytes transferred	Darwin, Dragonfly, FreeBSD, Linux, OpenBSD
netstat	Exposes network statistics from <code>/proc/net/netstat</code> . This is the same information as <code>netstat -s</code> .	Linux
Nfs	Exposes NFS client statistics from <code>/proc/net/rpc/nfs</code> . This is the same information as <code>nfsstat -c</code> .	Linux
Nfsd	Exposes NFS kernel server statistics from <code>/proc/net/rpc/nfsd</code> . This is the same information as <code>nfsstat -s</code> .	Linux
uname	Exposes system information as provided by the <code>uname</code> system call	Darwin, FreeBSD, Linux, OpenBSD



# Lab-4: Déploiement du node exporter

- Installez le node exporter Prometheus à l'aide du [chart helm](#) disponible sur le repo [Prometheus-community](#)
- Modifier la fichier config-map.yaml afin d'intégrer le node-exporter (un job si vous voulez)
- Vous devez supprimer et recréer le configmap ainsi que le deployment de Prometheus pour appliquer les modifications
- Vérifiez sur l'interface de Prometheus que la target node-exporter est bien présente et up
- Toujours sur l'interface de Prometheus vous pouvez vous assurez que les métriques **node\_load15**, **node\_cpu\_seconds\_total** renvoient des resultats
- Pour terminer, importer le dashboard [Grafana par excellence](#) pour le node-exporter
- Vérifiez que le dashboard nouvellement importé affiche des données



# Plan

- Présentation de la formation
- Rappels sur les microservices
- Les enjeux du monitoring
- Prometheus et Grafana
- Monitoring des hôtes / OS
- **Monitoring du Cluster**
- Monitoring des Conteneurs
- Monitoring des applications
- Alerting
- Operators
- Cloud Solution
- Miniprojet
- Conclusion

## Monitoring du Cluster (1/2): Kubernetes API health and metrics

- Kubernetes API Server
- Controller Manager
- Scheduler
- Etcd

# Monitoring du Cluster (2/2): kube-state-metrics

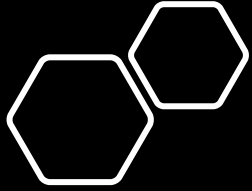
- C'est un service qui permet d'évaluer
  - Le nombre de pods démarrés / arrêtés / terminées
  - Le nombre de fois qu'un pod a été redémarré
- Analyse le temps de réponse des services kubernetes afin de
  - Déterminer les endpoints les plus adressés par les utilisateurs
  - Déterminer le endpoint http le plus lent
  - Déterminer les requêtes qui ont renvoyées un code erreur

# Lab-5: Mise en place des métriques K8S API

- Modifiez le configmap prometheus afin d'y intégrer le endpoint de l'API Kubernetes (n'hésitez pas à regarder la Doc)
- Vous devez supprimer et recréer le configmap ainsi que le deployment de Prometheus pour appliquer les modifications
- Vérifiez sur l'interface de Prometheus que la target apiserver est bien présente et up
- Pour terminer, importer le dashboard permettant de visualiser les [métriques des API k8s](#)
- Vérifiez que le dashboard nouvellement importé affiche des données

# Lab-6: Mise en place des métriques K8S State-metrics

- Déployez kube-state-metrics à l'aide de la [documentation officielle](#) en déployement l'ensemble des manifests présent
- Modifiez le configmap prometheus afin d'y intégrer le endpoint de kube-state-metrics précédement déployé (n'hésitez pas à regarder la Doc)
- Vous devez supprimer et recréer le configmap ainsi que le deployment de Prometheus pour appliquer les modifications
- Vérifiez sur l'interface de Prometheus que la target kube state est bien présente et up
- Toujours sur l'interface de Prometheus vous pouvez vous assurez que la métrique **kube\_deployment\_status\_replicas** renvoie un resultat
- Pour terminer, importer le dashboard permettant de visualiser les [métriques de kube-state](#)
- Vérifiez que le dashboard nouvellement importé affiche des données

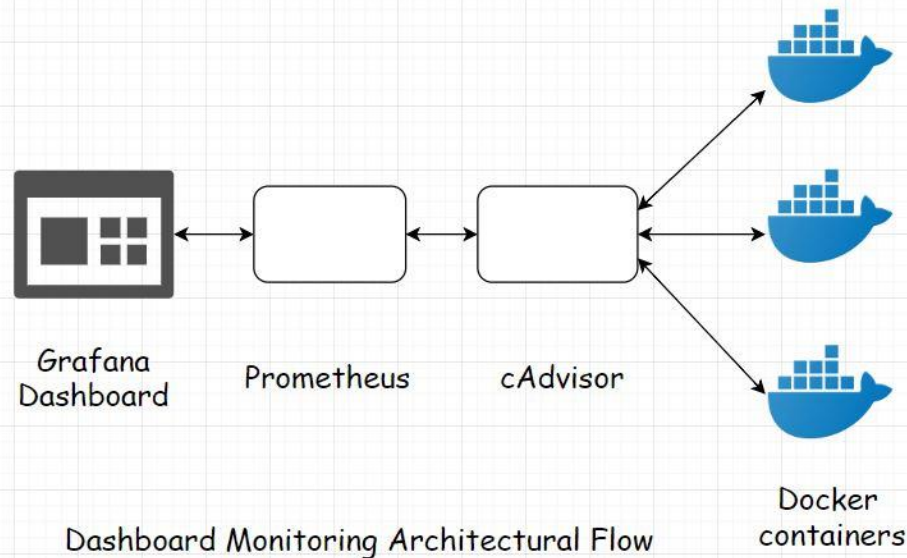


# Plan

- Présentation de la formation
- Rappels sur les microservices
- Les enjeux du monitoring
- Prometheus et Grafana
- Monitoring des hôtes / OS
- Monitoring du Cluster
- **Monitoring des Conteneurs**
- Monitoring des applications
- Alerting
- Operators
- Cloud Solution
- Miniprojet
- Conclusion



# Monitoring des Conteneurs (1/2): cAdvisor

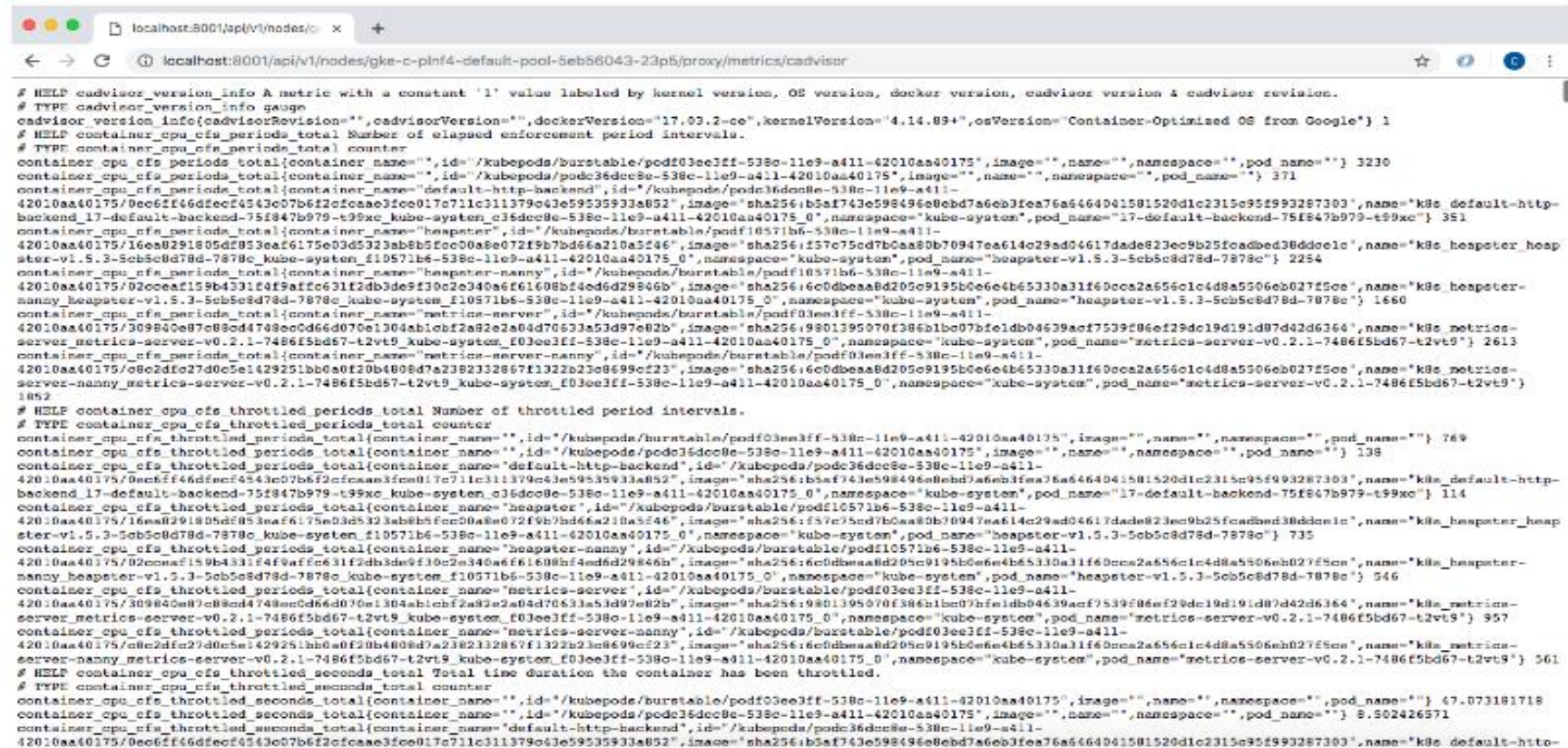


8bitmen.com

- Collecte les données des hôtes docker
- Collecte les données des conteneurs
- Il peut stocker ces données dans Prometheus ou Influxdb
- La visualization peut se faire via grafana

# Monitoring des Conteneurs (2/2): Kubelet metrics

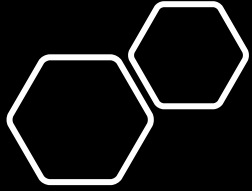
<http://localhost:8001/api/v1/nodes/gke-c-plnf4-default-pool-5eb56043-23p5/proxy/metrics/cadvisor>



```
# HELP cadvisor_version_info A metric with a constant '1' value labeled by kernel version, OS version, docker version, cadvisor version & cadvisor revision.
# TYPE cadvisor_version_info gauge
cadvisor_version_info{cadvisorRevision="",cadvisorVersion="",dockerVersion="17.03.2-ce",kernelVersion="4.14.83+",osVersion="Container-Optimized OS from Google"} 1
# HELP container_cpu_cfs_periods_total Number of elapsed enforcement period intervals.
# TYPE container_cpu_cfs_periods_total counter
container_cpu_cfs_periods_total{container_name="",id="/kubepods/burstable/podf03ee3ff-538c-11e9-a411-42010aa40175",image="",name="",namespace="",pod_name=""} 3230
container_cpu_cfs_periods_total{container_name="",id="/kubepods/podc36dce8e-538c-11e9-a411-42010aa40175",image="",name="",namespace="",pod_name=""} 371
container_cpu_cfs_periods_total{container_name="default-http-backend",id="/kubepods/podc36dce8e-538c-11e9-a411-42010aa40175/0ec6ff46dfe0f4543c07b6f2cfcaae3fce017c11311379c43e59535933a852",image="sha256:b5af743e598496e8b7a6eb3fea76a6464041581520d1c2315c95f993287303",name="k8s_default-http-backend_17-default-backend-75f847b979-t99xc_kube-system_c36dce8e-538c-11e9-a411-42010aa40175_0",namespace="kube-system",pod_name="17-default-backend-75f847b979-t99xc"} 351
container_cpu_cfs_periods_total{container_name="heapster",id="/kubepods/burstable/podf10571b6-538c-11e9-a411-42010aa40175/16ea8291805df853caef175e03d5323ab8b5fcc00a8e072f9b7bd66a210a5f46",image="sha256:f57c75cd7b0aa80b70947ae614c29ad04617dade823ec9b25fcaadbed38ddc1c",name="k8s_heapster_heapster-v1.5.3-5eb5c8d78d-7878c_kube-system_f10571b6-538c-11e9-a411-42010aa40175_0",namespace="kube-system",pod_name="heapster-v1.5.3-5eb5c8d78d-7878c"} 2254
container_cpu_cfs_periods_total{container_name="heapster-nanny",id="/kubepods/burstable/podf10571b6-538c-11e9-a411-42010aa40175/02cccef159b4331f4f9affc631f2db3de9f39c2e340ae6f61608bf4ed6d29846b",image="sha256:6c0dbaa8d205c9195b6e6e4b65330a31f60cca2a656c1c4d8a5506eb027f5cc",name="k8s_heapster-nanny_heapster-v1.5.3-5eb5c8d78d-7878c_kube-system_f10571b6-538c-11e9-a411-42010aa40175_0",namespace="kube-system",pod_name="heapster-nanny_heapster-v1.5.3-5eb5c8d78d-7878c"} 1660
container_cpu_cfs_periods_total{container_name="metrics-server",id="/kubepods/burstable/podf03ee3ff-538c-11e9-a411-42010aa40175/309840e07c88d478e0c0d66d070e1304ab1cbf2a82e2a8d470631a53d97e82b",image="sha256:9801395070f386d1b07b1cfd1db04639acf7539f86ef29dc19d191d87d42d6364",name="k8s_metrics-server_metrics-server-v0.2.1-7486f5bd67-t2vt9_kube-system_f03ee3ff-538c-11e9-a411-42010aa40175_0",namespace="kube-system",pod_name="metrics-server-v0.2.1-7486f5bd67-t2vt9"} 2613
container_cpu_cfs_periods_total{container_name="metrics-server-nanny",id="/kubepods/burstable/podf03ee3ff-538c-11e9-a411-42010aa40175/c8c2dfc27d0c5e1429251bb0a0f20b4808d7a2382132867f1322e2308699cf23",image="sha256:6c0dbaa8d205c9195b6e6e4b65330a31f60cca2a656c1c4d8a5506eb027f5cc",name="k8s_metrics-server-nanny_metrics-server-v0.2.1-7486f5bd67-t2vt9_kube-system_f03ee3ff-538c-11e9-a411-42010aa40175_0",namespace="kube-system",pod_name="metrics-server-v0.2.1-7486f5bd67-t2vt9"} 1852
# HELP container_cpu_cfs_throttled_periods_total Number of throttled period intervals.
# TYPE container_cpu_cfs_throttled_periods_total counter
container_cpu_cfs_throttled_periods_total{container_name="",id="/kubepods/burstable/podf03ee3ff-538c-11e9-a411-42010aa40175",image="",name="",namespace="",pod_name=""} 769
container_cpu_cfs_throttled_periods_total{container_name="",id="/kubepods/podc36dce8e-538c-11e9-a411-42010aa40175",image="",name="",namespace="",pod_name=""} 138
container_cpu_cfs_throttled_periods_total{container_name="default-http-backend",id="/kubepods/podc36dce8e-538c-11e9-a411-42010aa40175/0ec6ff46dfe0f4543c07b6f2cfcaae3fce017c11311379c43e59535933a852",image="sha256:b5af743e598496e8b7a6eb3fea76a6464041581520d1c2315c95f993287303",name="k8s_default-http-backend_17-default-backend-75f847b979-t99xc_kube-system_c36dce8e-538c-11e9-a411-42010aa40175_0",namespace="kube-system",pod_name="17-default-backend-75f847b979-t99xc"} 114
container_cpu_cfs_throttled_periods_total{container_name="heapster",id="/kubepods/burstable/podf10571b6-538c-11e9-a411-42010aa40175/16ea8291805df853caef175e03d5323ab8b5fcc00a8e072f9b7bd66a210a5f46",image="sha256:f57c75cd7b0aa80b70947ae614c29ad04617dade823ec9b25fcaadbed38ddc1c",name="k8s_heapster_heapster-v1.5.3-5eb5c8d78d-7878c_kube-system_f10571b6-538c-11e9-a411-42010aa40175_0",namespace="kube-system",pod_name="heapster-v1.5.3-5eb5c8d78d-7878c"} 735
container_cpu_cfs_throttled_periods_total{container_name="heapster-nanny",id="/kubepods/burstable/podf10571b6-538c-11e9-a411-42010aa40175/02cccef159b4331f4f9affc631f2db3de9f39c2e340ae6f61608bf4ed6d29846b",image="sha256:6c0dbaa8d205c9195b6e6e4b65330a31f60cca2a656c1c4d8a5506eb027f5cc",name="k8s_heapster-nanny_heapster-v1.5.3-5eb5c8d78d-7878c_kube-system_f10571b6-538c-11e9-a411-42010aa40175_0",namespace="kube-system",pod_name="heapster-nanny_heapster-v1.5.3-5eb5c8d78d-7878c"} 546
container_cpu_cfs_throttled_periods_total{container_name="metrics-server",id="/kubepods/burstable/podf03ee3ff-538c-11e9-a411-42010aa40175/309840e07c88d478e0c0d66d070e1304ab1cbf2a82e2a8d470631a53d97e82b",image="sha256:9801395070f386d1b07b1cfd1db04639acf7539f86ef29dc19d191d87d42d6364",name="k8s_metrics-server_metrics-server-v0.2.1-7486f5bd67-t2vt9_kube-system_f03ee3ff-538c-11e9-a411-42010aa40175_0",namespace="kube-system",pod_name="metrics-server-v0.2.1-7486f5bd67-t2vt9"} 957
container_cpu_cfs_throttled_periods_total{container_name="metrics-server-nanny",id="/kubepods/burstable/podf03ee3ff-538c-11e9-a411-42010aa40175/c8c2dfc27d0c5e1429251bb0a0f20b4808d7a2382132867f1322e2308699cf23",image="sha256:6c0dbaa8d205c9195b6e6e4b65330a31f60cca2a656c1c4d8a5506eb027f5cc",name="k8s_metrics-server-nanny_metrics-server-v0.2.1-7486f5bd67-t2vt9_kube-system_f03ee3ff-538c-11e9-a411-42010aa40175_0",namespace="kube-system",pod_name="metrics-server-v0.2.1-7486f5bd67-t2vt9"} 561
# HELP container_cpu_cfs_throttled_seconds_total Total time duration the container has been throttled.
# TYPE container_cpu_cfs_throttled_seconds_total counter
container_cpu_cfs_throttled_seconds_total{container_name="",id="/kubepods/burstable/podf03ee3ff-538c-11e9-a411-42010aa40175",image="",name="",namespace="",pod_name=""} 47.073181718
container_cpu_cfs_throttled_seconds_total{container_name="",id="/kubepods/podc36dce8e-538c-11e9-a411-42010aa40175",image="",name="",namespace="",pod_name=""} 8.502426571
container_cpu_cfs_throttled_seconds_total{container_name="default-http-backend",id="/kubepods/podc36dce8e-538c-11e9-a411-42010aa40175/0ec6ff46dfe0f4543c07b6f2cfcaae3fce017c11311379c43e59535933a852",image="sha256:b5af743e598496e8b7a6eb3fea76a6464041581520d1c2315c95f993287303",name="k8s_default-http-
```

# Lab-7: Visualisation des métriques docker

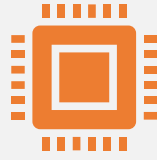
- Modifiez le configmap prometheus afin d'y intégrer le endpoint de Kubernetes-cadvisor (n'hésitez pas à regarder la Doc)
- Vous devez supprimer et recréer le configmap ainsi que le deployment de Prometheus pour appliquer les modifications
- Vérifiez sur l'interface de Prometheus que la target Kubernetes-cadvisor est bien présente et up
- Pour terminer, importer le dashboard permettant de visualizer [les métriques cadvisor](#)
- Vérifiez que le dashboard nouvellement importé affiche des données



# Plan

- Présentation de la formation
- Rappels sur les microservices
- Les enjeux du monitoring
- Prometheus et Grafana
- Monitoring des hôtes / OS
- Monitoring du Cluster
- Monitoring des Conteneurs
- **Monitoring des applications**
- Alerting
- Operators
- Cloud Solution
- Miniprojet
- Conclusion

# Monitoring des applications (1/3): Exemples



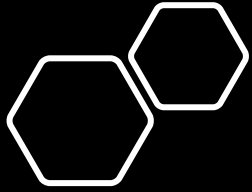
HTTP



Database



API



# Monitoring des applications (2/3): Blackbox

- Go Prometheus exporter
- Permet de tester la disponibilité d'un endpoint via les protocols
  - HTTP
  - HTTPS
  - DNS
  - TCP
  - ICMP



# Monitoring des applications (3/3): Postgres- exporter

- Go Prometheus exporter
- Permet d'exporter les métriques d'une base de données postgresql tels que:
  - La durée des requêtes
  - La RAM et le CPU consommées par cette dernière

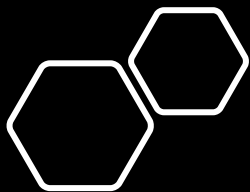
# Lab-8: Déploiement de blackbox

- Installez le blackbox exporter Prometheus à l'aide du [chart helm](#) disponible sur le repo [Prometheus-community](#)
- Surchargez les variables du chart avec le fichier [values.yaml](#)
- Modifier la fichier config-map.yaml afin d'intégrer le blackbox-exporter (un job si vous voulez)
- Vous devez supprimer et recréer le configmap ainsi que le deployment de Prometheus pour appliquer les modifications
- Vérifiez sur l'interface de Prometheus que la target blackbox-exporter est bien présente et up
- Pour terminer, importer le dashboard [blackbox](#)
- Vérifiez que le dashboard nouvellement importé affiche des données



# Lab-9: Déploiement de postgres exporter

- Installez le postgresql-exporter Prometheus à l'aide du [chart helm](#) disponible sur le repo [Prometheus-community](#)
- Surchargez les variables du chart avec le fichier [values.yaml](#)
- Modifier la fichier config-map.yaml afin d'intégrer le postgres-exporter (un job si vous voulez)
- Vous devez supprimer et recréer le configmap ainsi que le deployment de Prometheus pour appliquer les modifications
- Vérifiez sur l'interface de Prometheus que la target postgres-exporter est bien présente et up
- Pour terminer, importer le dashboard [blackbox](#)
- Vérifiez que le dashboard nouvellement importé affiche des données

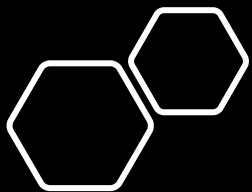


# Plan

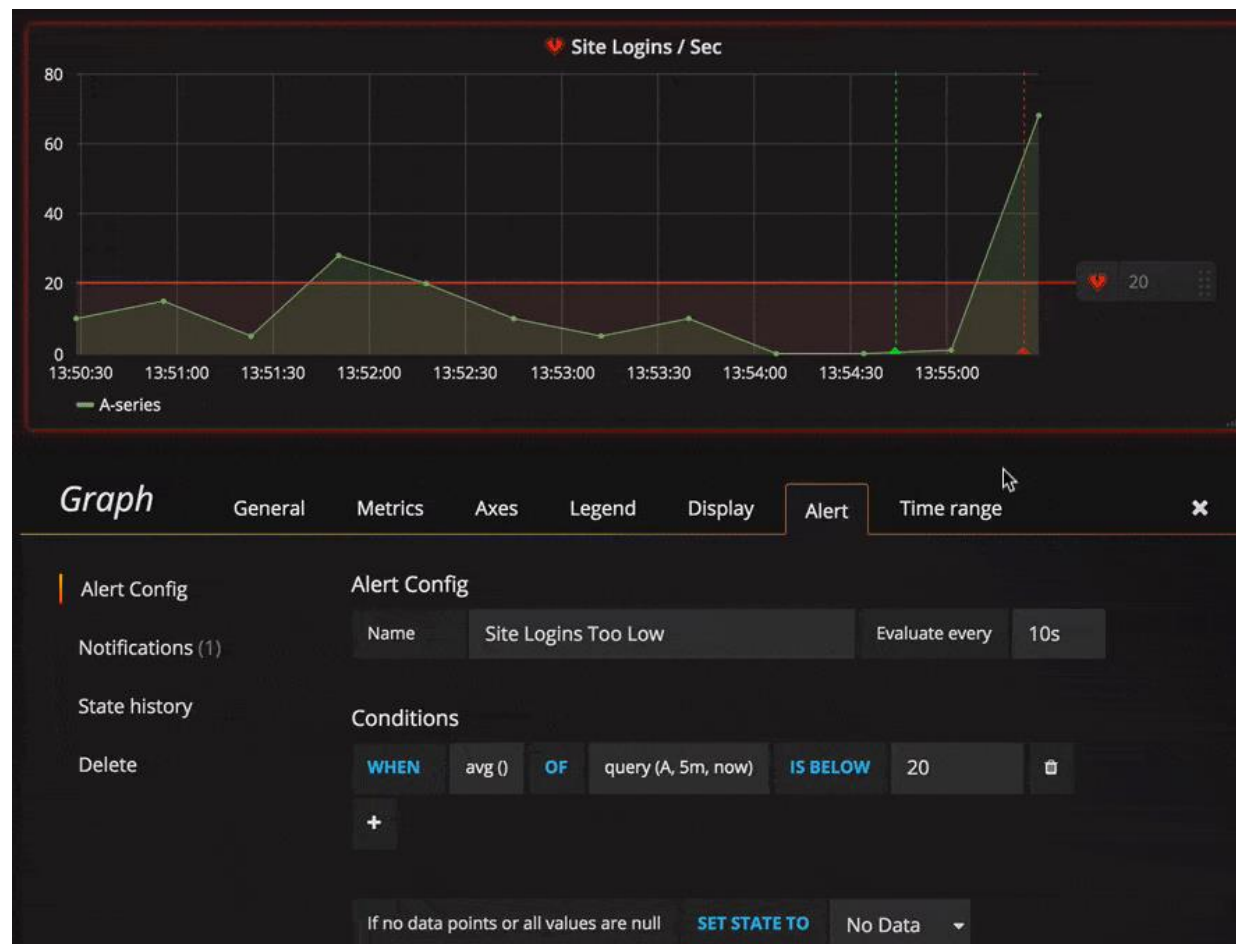
- Présentation de la formation
- Rappels sur les microservices
- Les enjeux du monitoring
- Prometheus et Grafana
- Monitoring des hôtes / OS
- Monitoring du Cluster
- Monitoring des Conteneurs
- Monitoring des applications
- **Alerting**
- Operators
- Cloud Solution
- Miniprojet
- Conclusion

# Alerting (1/2): Prometheus

```
prometheus.rules: |-
  groups:
  - name: devopscube demo alert
    rules:
    - alert: High Pod Memory
      expr: sum(container_memory_usage_bytes) > 1
      for: 1m
      labels:
        severity: slack
      annotations:
        summary: High Memory Usage
prometheus.yml: |-
  global:
    scrape_interval: 20s
    evaluation_interval: 20s
  rule_files:
  - /etc/prometheus/prometheus.rules
  alerting:
    alertmanagers:
    - scheme: http
      static_configs:
      - targets:
        - "10.1.150.150:32000"
```



# Alerting (2/2): Grafana alerting

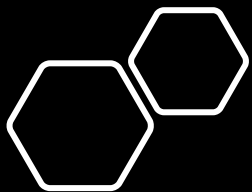


# Lab-10: Connexion Prometheus - alertmanager

- Déployez l'alertmanager à l'aide des [sources fournies](#)
- Accédez à l'interface de l'alertmanager via le service de type nodeport
- Modifiez le configmap afin d'y insérer les règles présentées dans les slides précédents
- Vous devez supprimer et recréer le configmap ainsi que le deployment de Prometheus pour appliquer les modifications
- Vérifiez sur l'interface de Prometheus que l'alerte a bien été envoyée à l'alertmanager

# Lab-11: Odoo health alert dans grafana

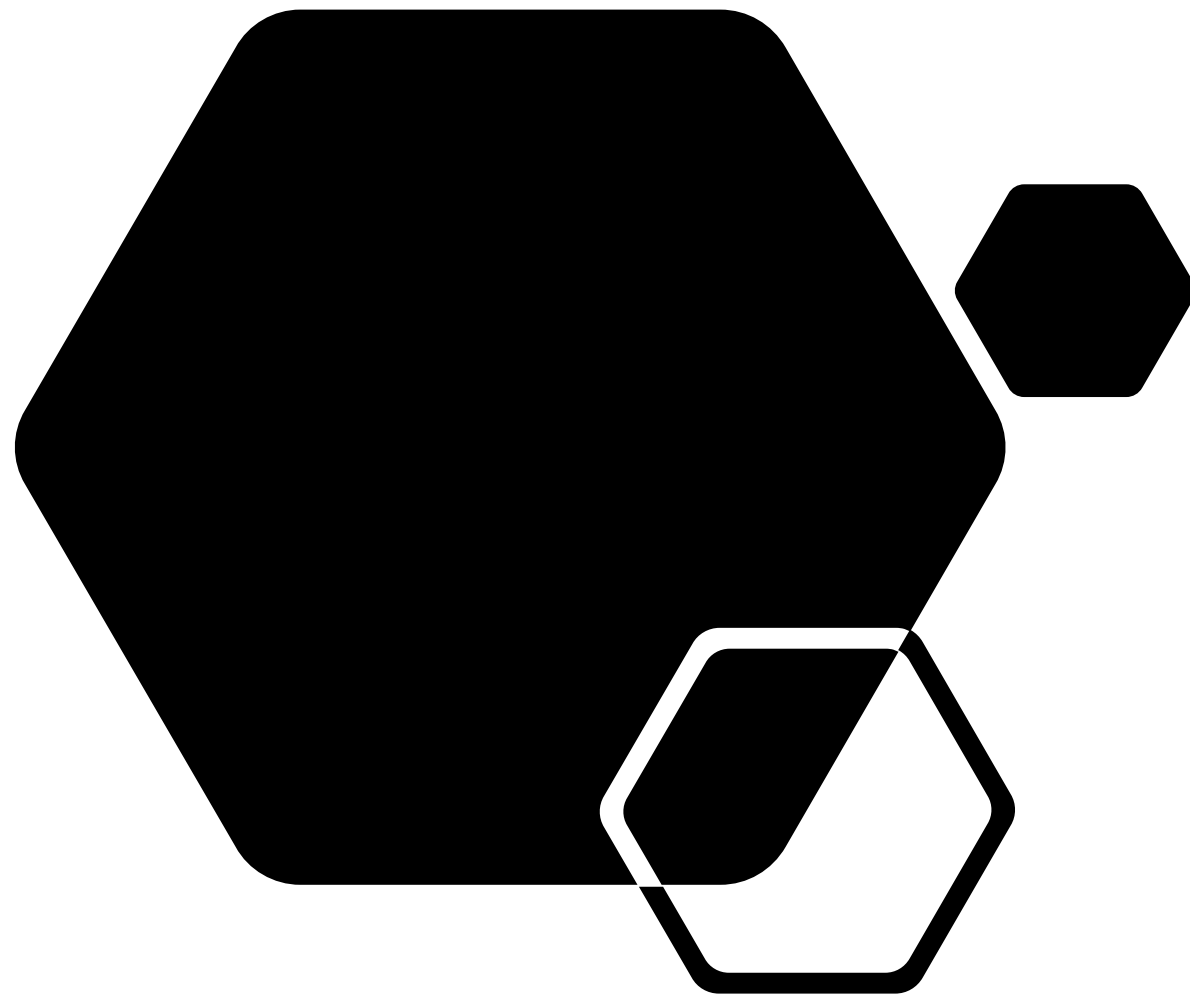
- Mettez en place une alerte qui se déclenche lorsque le service odoo est down (en utilisant le cadrant du dashboard Grafana lié au blackbox qui marque up/down). En effet cet indicateur depend de la valeur de **probe\_success** pour chacunes des targets qu'il surveille
- Mettez en place un notification slack qui doit être déclenchée quand l'alerte est observée
- Tentez de scaler le nombre de pod odoo à 0 et verifiez que vous recevez bien l'alerte sur slack



# Plan

- Présentation de la formation
- Rappels sur les microservices
- Les enjeux du monitoring
- Prometheus et Grafana
- Monitoring des hôtes / OS
- Monitoring du Cluster
- Monitoring des Conteneurs
- Monitoring des applications
- Alerting
- **Operators**
- Cloud Solution
- Miniprojet
- Conclusion

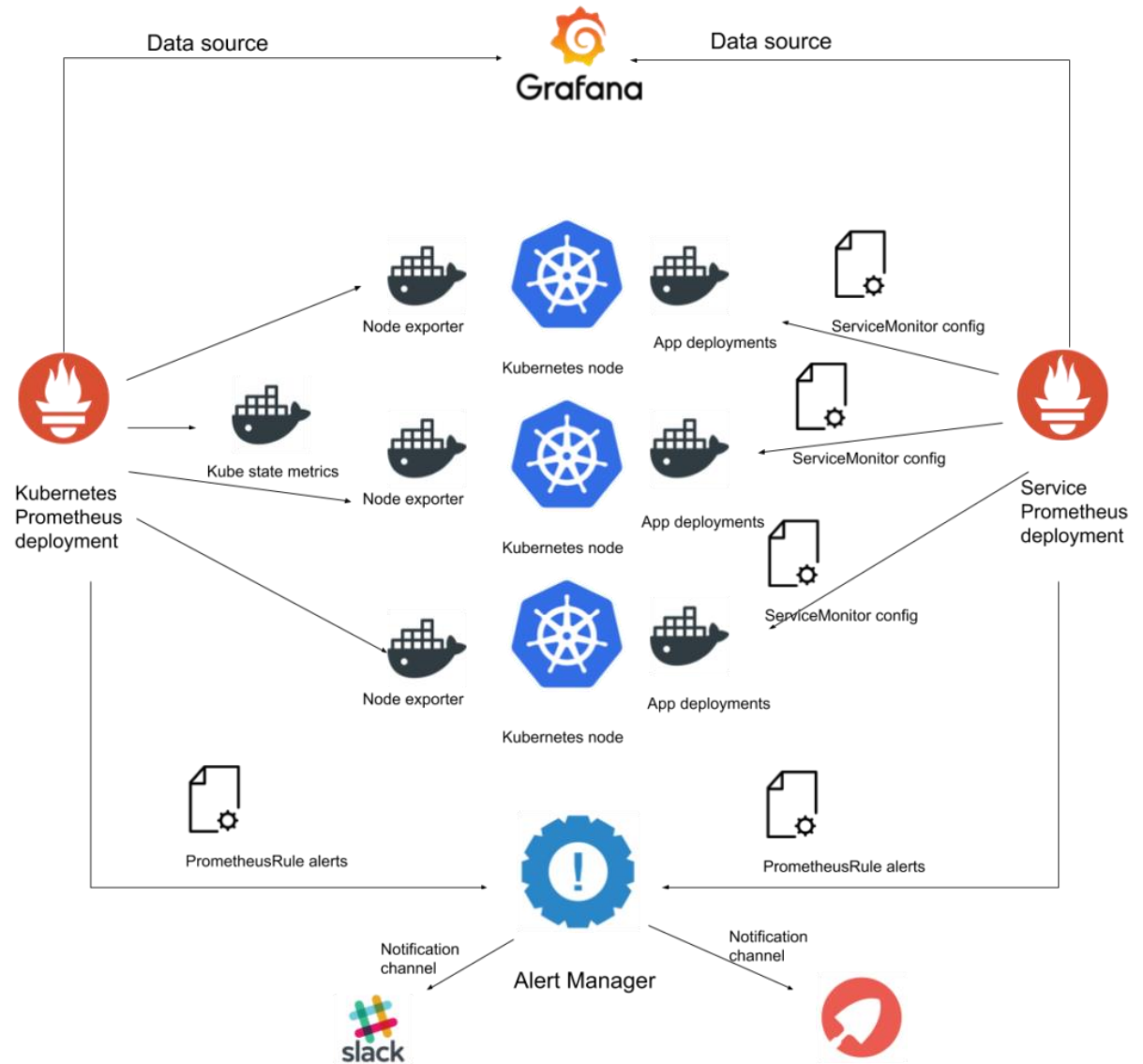
# Operators (1/2): Principe





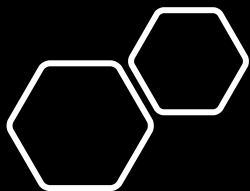
# Operators (2/2): Prometheus

- Création et Suppression d'instance Prometheus
- Simple à configurer
- Target Services via labels
- CRD
  - Alertmanagers
  - Podmonitors
  - Prometheus
  - Prometheusrules
  - servicemonitors



# Lab-12: Déploiement du Prometheus operator

- Conservez les scripts et fichiers sur un git
- Déployez un nouveau cluster Kubernetes vierge
- Installez helm 3
- Utilisez le [chart helm officiel](#) pour deployer Prometheus
- Parcourez les sources créer et faites des comparaisons avec le travail que nous avons réalisé manuellement
- Tirez-en des conclusions



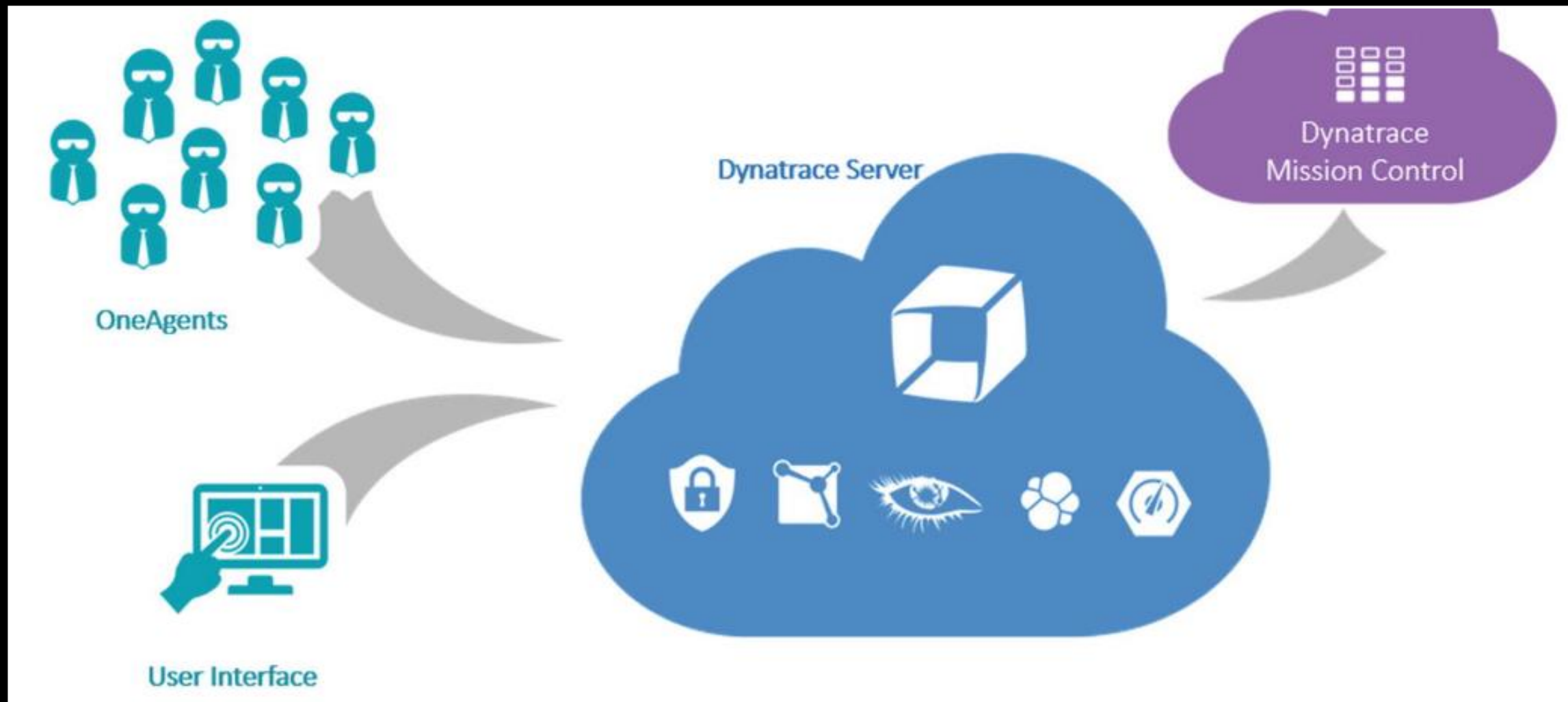
# Plan

- Présentation de la formation
- Rappels sur les microservices
- Les enjeux du monitoring
- Prometheus et Grafana
- Monitoring des hôtes / OS
- Monitoring du Cluster
- Monitoring des Conteneurs
- Monitoring des applications
- Alerting
- Operators
- **Cloud Solution**
- Miniprojet
- Conclusion

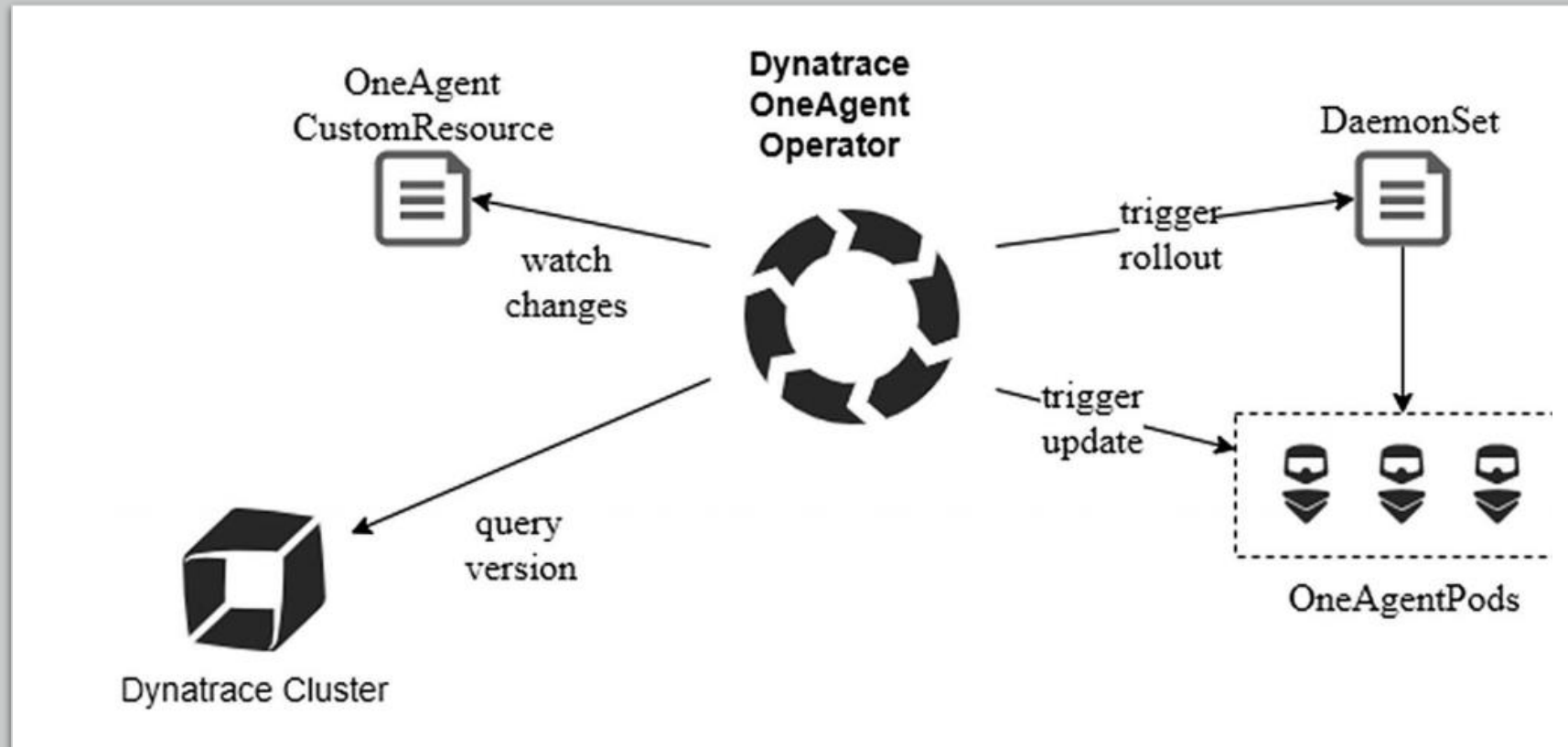
# Cloud Solutions (1/5): Dynatrace (1/3)

- Software-intelligence monitoring platform
- deployable on-premise (Dynatrace Managed) ou dans le Cloud (Dynatrace SaaS)
- Il apporte une solution pour le monitoring et la performance de nos infrastructures et nos applications on-premise/cloud à l'aide de Davis
- Fonctions clés:
  - Real User Monitoring
  - Server-side Service Monitoring
  - Network, Process et Host Monitoring
  - Container Monitoring





## Cloud Solutions (2/5): Dynatrace (2/3)

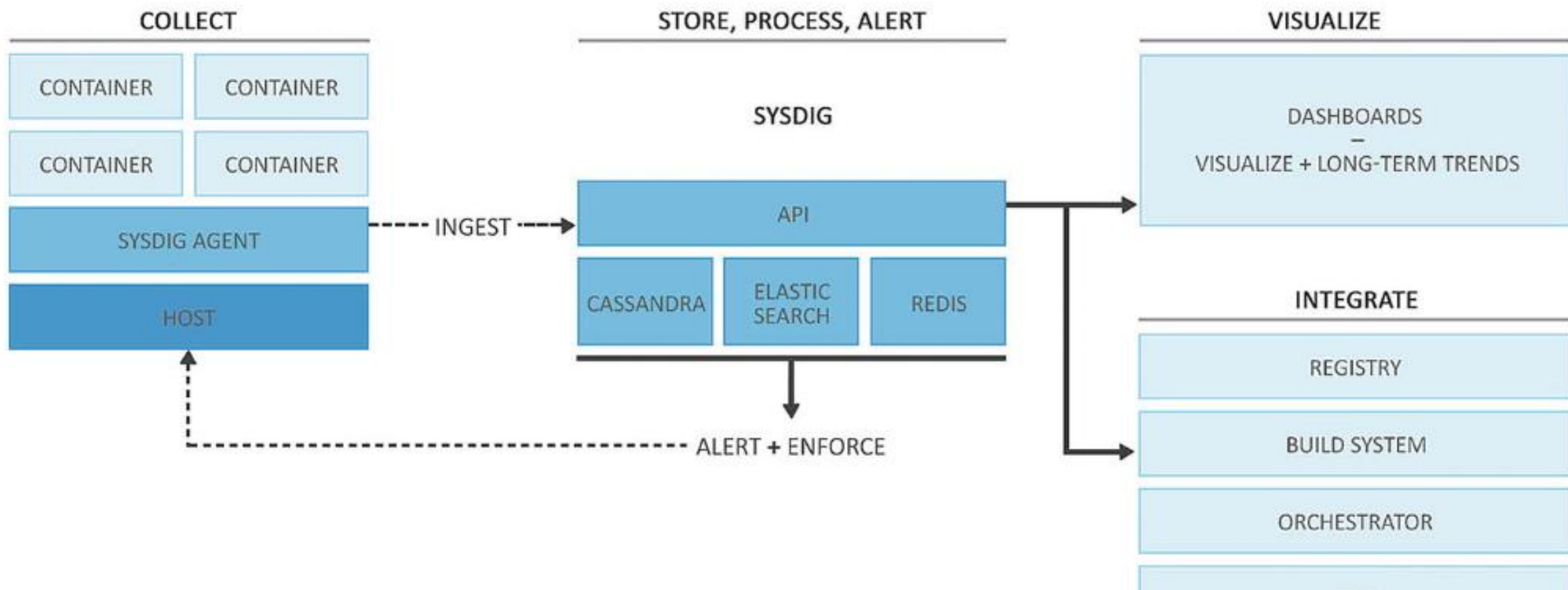


## Cloud Solutions (3/5): Dynatrace (3/3)

# Cloud Solutions (4/5): Sysdig (1/2)

- Solution de monitoring et troubleshooting Container-Native
- Il s'intègre parfaitement les orchestrateurs Kubernetes, Docker Swarm, AWS EKS, Azure AKS et Google GKE
- Il est disponible en version on-premise et saas
- Fonctionnalités clés:
  - Auto-discovery des applications, conteneurs, hôtes et réseaux
  - Visualizes service reliability
  - Monitoring de l'infra et des applications
  - Custom Dashboard
  - Simplifies and scales Prometheus monitoring
  - Exploration visuel de votre infrastructure
  - Alertes proactives
  - Accélère le troubleshooting



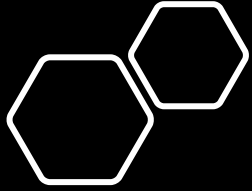


## Cloud Solutions (5/5): Sysdig (2/2)



# Lab-13: Implémentation de Dynatrace

- Créez un [compte gratuit](#)
- Déployez un cluster vierge (minikube delete -> minikube start)
- Déployez une fois de plus l'application odoo en utilisant helm 3
- Utilisez la [documentation officielle](#) pour mettre en place le monitoring de vos microservices à l'aide de Dynatrace en utilisant l'opérateur Dynatrace + Helm
- Parcourez la [liste](#) des applications supportées par dynatrace
- Parcourez le dashboard et amusez-vous
- Parcourez la [tarification](#)



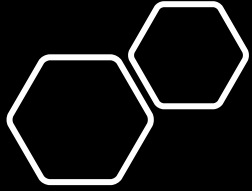
# Plan

- Présentation de la formation
- Rappels sur les microservices
- Les enjeux du monitoring
- Prometheus et Grafana
- Monitoring des hôtes / OS
- Monitoring du Cluster
- Monitoring des Conteneurs
- Monitoring des applications
- Alerting
- Operators
- Cloud Solution
- **Miniprojet**
- Conclusion

# Miniprojet: Wordpress



- [Chart wordpress](#) disponible
- Monitoring applicatif (http) et backend (mysql) à l'aide des exporter correspondant en utilisant helm
- Monitoring OS
- Monitoring Kubernetes
- Monitoring Container
- BONUS : utiliser l'opérateur Prometheus si vous pouvez, sinon faites le manuellement



# Plan

- Présentation de la formation
- Rappels sur les microservices
- Les enjeux du monitoring
- Prometheus et Grafana
- Monitoring des hôtes / OS
- Monitoring du Cluster
- Monitoring des Conteneurs
- Monitoring des applications
- Alerting
- Operators
- Cloud Solution
- Miniprojet
- **Conclusion**

# Conclusion

