



Мониторинг Docker



Логи и метрики

Логи — текст*

Метрики - не текст
(а похоже)

*Иногда бинарные, но там тоже текст

```
➔ declarative-lookup-rollup-summaries git:(master)
sfdx force:apex:log:tail --color | grep USER_DEBUG
20:20:54.0 (470304968)|USER_DEBUG|[220]IDEBUG|SOQL is SELECT AccountId, Sum(Amount) lre0 FROM Opportunity
20:20:55.52 (1244253913)|USER_DEBUG|[220]IDEBUG|SOQL is SELECT AccountId, Sum(Amount) lre0 FROM Opportuni
20:20:55.52 (1257140303)|USER_DEBUG|[254]INFO|New aggregarte value 500.0 for master 0016300000YI8MnAAL
20:20:55.52 (1261582516)|USER_DEBUG|[220]IDEBUG|SOQL is SELECT AccountId, Count(CloseDate) lre0, Sum(Amou
20:20:55.52 (1269702915)|USER_DEBUG|[254]INFO|New aggregarte value 4 for master 0016300000YI8MnAAL
20:20:55.52 (1270343744)|USER_DEBUG|[254]INFO|New aggregarte value 600.0 for master 0016300000YI8MnAAL
20:21:12.0 (241563308)|USER_DEBUG|[220]IDEBUG|SOQL is SELECT AccountId, Sum(Amount) lre0 FROM Opportunity
20:21:12.0 (249003313)|USER_DEBUG|[254]INFO|New aggregarte value 42512.0 for master 0016300000YHxYZAAL
20:21:19.0 (192238208)|USER_DEBUG|[220]IDEBUG|SOQL is SELECT AccountId, Sum(Amount) lre0 FROM Opportunity
20:21:19.0 (198444538)|USER_DEBUG|[254]INFO|New aggregarte value 47512.0 for master 0016300000YHxYZAAL

jvm_memory_pool_bytes_used{pool="CodeHeap 'non-nmethods'",} 1494912.0
jvm_memory_pool_bytes_used{pool="Metaspace",} 1.80054296E8
jvm_memory_pool_bytes_used{pool="CodeHeap 'profiled nmethods'",} 4.9295232E7
jvm_memory_pool_bytes_used{pool="Compressed Class Space",} 2.2233568E7
jvm_memory_pool_bytes_used{pool="G1 Eden Space",} 2.55852544E8
jvm_memory_pool_bytes_used{pool="G1 Old Gen",} 9.0902528E7
jvm_memory_pool_bytes_used{pool="G1 Survivor Space",} 1.1534336E7
jvm_memory_pool_bytes_used{pool="CodeHeap 'non-profiled nmethods'",} 1.7572224E7
# HELP jvm_memory_pool_bytes_committed Committed bytes of a given JVM memory pool.
# TYPE jvm_memory_pool_bytes_committed gauge
jvm_memory_pool_bytes_committed{pool="CodeHeap 'non-nmethods'",} 2555904.0
jvm_memory_pool_bytes_committed{pool="Metaspace",} 1.9529728E8
jvm_memory_pool_bytes_committed{pool="CodeHeap 'profiled nmethods'",} 5.1904512E7
jvm_memory_pool_bytes_committed{pool="Compressed Class Space",} 2.7262976E7
jvm_memory_pool_bytes_committed{pool="G1 Eden Space",} 6.46971392E8
jvm_memory_pool_bytes_committed{pool="G1 Old Gen",} 3.8797312E8
jvm_memory_pool_bytes_committed{pool="G1 Survivor Space",} 1.1534336E7
jvm_memory_pool_bytes_committed{pool="CodeHeap 'non-profiled nmethods'",} 1.7629184E7
# HELP jvm_memory_pool_bytes_max Max bytes of a given JVM memory pool.
# TYPE jvm_memory_pool_bytes_max gauge
```



Prometheus

Prometheus – timeseries база данных





Prometheus



Prometheus – timeseries база данных

А еще формат метрик для этой базы
данных



Prometheus + Grafana

Prometheus – timeseries база данных

А еще формат метрик для этой базы данных

Grafana – инструмент визуализации почти чего угодно.





Метрики Docker демона

Включаем:

```
{  
  "experimental": true,  
  "metrics-addr": "0.0.0.0:9100"  
}
```





Метрики Docker демона

Включаем:

```
{  
  "experimental": true,  
  "metrics-addr": "0.0.0.0:9100"  
}
```

Проверяем:

```
curl localhost:9100/metrics
```





Когда встроенных метрик недостаточно

Внешние экспортеры





Когда встроенных метрик недостаточно

Внешние экспортеры

Например:

<https://github.com/draganm/missing-container-metrics>





Когда встроенных метрик недостаточно

Внешние экспортеры

Например:

- <https://github.com/draganm/missing-container-metrics>

- <https://github.com/google/cadvisor>





Когда встроенных метрик недостаточно

Внешние экспортеры

Например:

- <https://github.com/draganm/missing-container-metrics>

- <https://github.com/google/cadvisor>

- Ну или пишем свой экспортер





Метрики приложений

Просто публикуем нужный порт

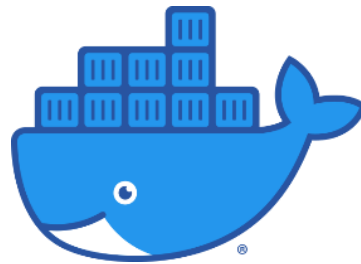
```
docker run -p 8080:8080 -p 9090:9090 my-app
```

Хорошая практика отдавать метрики на /metrics





Docker Stats



Показывает текущее потребление

CONTAINER ID	NAME	CPU %	MEM USAGE / LIMIT	MEM %	NET I/O	BLOCK I/O	PIDS
1248ce98fb7a	magical_dubinsky	0.00%	0B / 0B	0.00%	0B / 0B	0B / 0B	0
b38ec85d0dff	zealous_pasteur	0.00%	4.027MiB / 25.02GiB	0.02%	1.08kB / 0B	0B / 0B	2
6e186b4d80c8	strange_einstein	0.00%	4.684MiB / 25.02GiB	0.02%	1.08kB / 0B	0B / 0B	2
b381b3002eb4	sharp_heisenberg	0.00%	4.449MiB / 25.02GiB	0.02%	1.15kB / 0B	0B / 0B	2
21c1a80de44f	elastic_curran	0.00%	5.359MiB / 25.02GiB	0.02%	1.59kB / 0B	0B / 0B	2

Метрики в псевдофайлах:

/sys/fs/cgroup/memory/docker/<longid>/memory.stat

/sys/fs/cgroup/cpu/docker/<longid>/cpuacct.stat



Cadvisor

Отдаёт метрики контейнеров в удобной форме





Stop

Как top только для контейнеров

cTop - 10:04:06 AEDT 20 containers					
NAME	CID	CPU	MEM	NET RX/TX	
● luminous_lady	cf754eb3aa09	33%	81M / 2G	1K / 1K	
● ultimate_jennifer	9eb1e9a6cb91	30%	82M / 2G	1K / 1K	
● top_notch	7b5fd634a980	24%	103M / 2G	1K / 1K	
● neat_roulette	0677c5437698	14%	40M / 2G	400B / 452B	
● exquisite_jackpot	72372078a320	12%	38M / 2G	487B / 519B	
● neat_multiple	5fbe5928be03	10%	31M / 2G	645B / 626B	
● ace_void	292db4fa7c5c	8%	37M / 2G	496B / 559B	
● peachy_sakura	70bd0664ff8c	8%	36M / 2G	468B / 505B	
● legendary_korath	23a40fbba1cb	6%	36M / 2G	621B / 566B	
● astonishing_nikita	93de62c2b03f	-	-	-	
● cats_pajamas	ec19826bd862	-	-	-	
● fantabulous_titane...	8e23ebb7e05f	-	-	-	
● grand_bebop	c3a231ea8f49	-	-	-	
■ impressive_sentine...	e154df51f8e6	-	-	-	

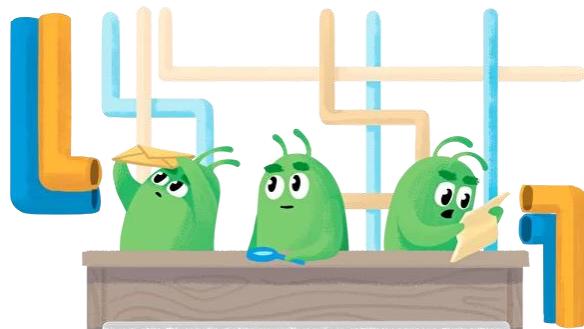
Давайте посмотрим,
как это работает!





Logging driver

Настраивается в daemon.json

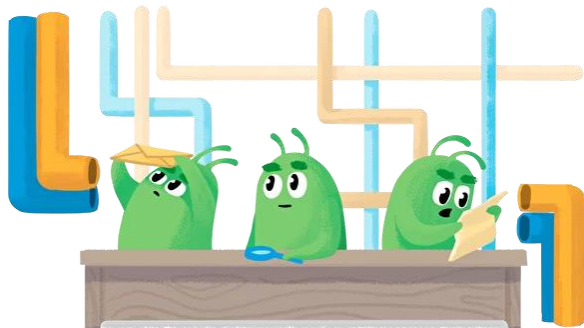




Logging driver

Настраивается в daemon.json

Внутренние: **local**, **json-file**



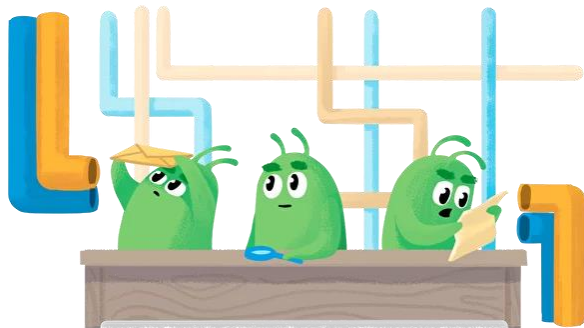


Logging driver

- Настраивается в daemon.json

- Внутренние: **local**, **json-file**

- Внешние: **syslog**, journald, fluentd, итд





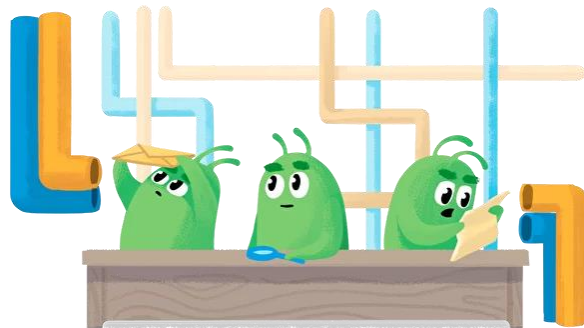
Logging driver

- Настраивается в daemon.json

- Внутренние: **local**, **json-file**

- Внешние: **syslog**, journald, fluentd, итд

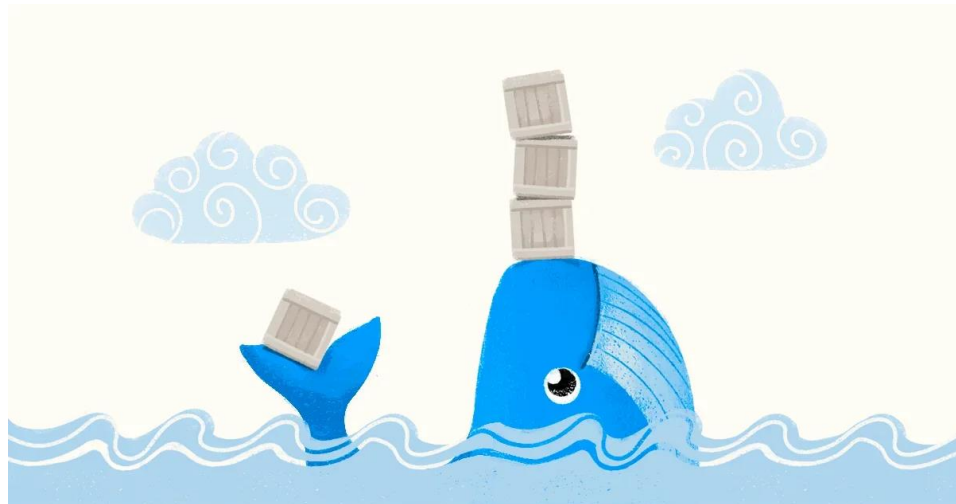
Для **оранжевых** будет работать docker logs





Docker inspect

Показывает состояние контейнера:





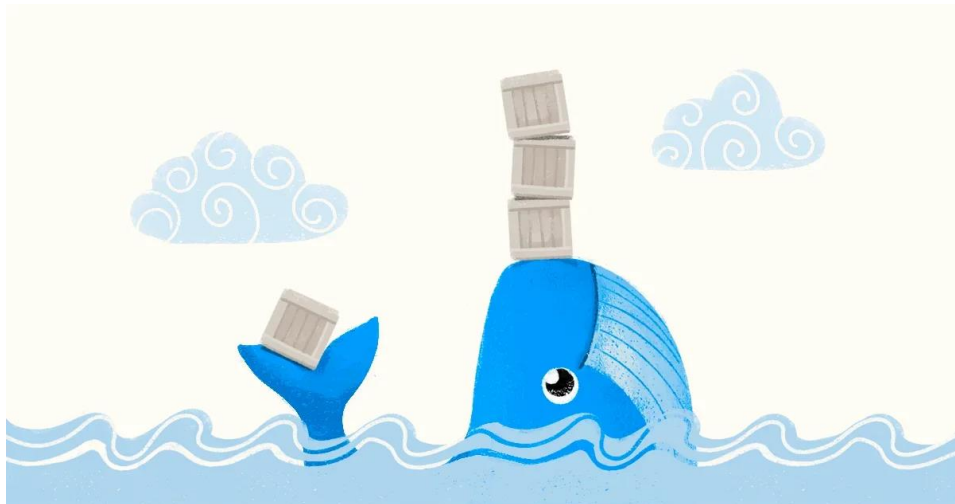
Docker inspect

Показывает состояние контейнера:

- OOMkilled

- ExitCode

- Error





Docker inspect

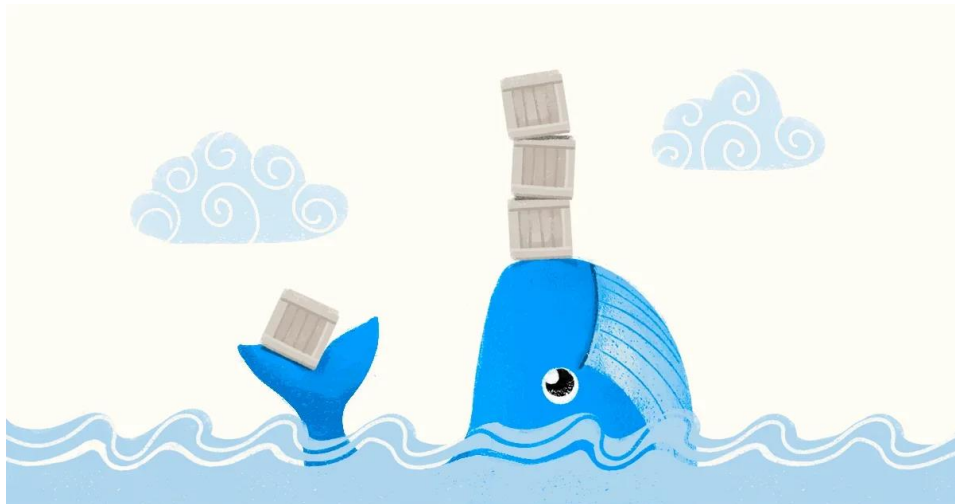
Показывает состояние контейнера:

- OOMkilled

- ExitCode

- Error

Jq ваш друг!





Healthcheck



Контейнер жив, пока жив процесс 1





Healthcheck

- Контейнер жив, пока жив процесс 1
- Процесс жив != приложение работает





Healthcheck

- Контейнер жив, пока жив процесс 1
- Процесс жив != приложение работает
- Настраивается в dockerfile
HEALTHCHECK CMD curl 127.0.0.1





Healthcheck

- Контейнер жив, пока жив процесс 1
- Процесс жив != приложение работает
- Настраивается в dockerfile
HEALTHCHECK CMD curl 127.0.0.1
- Практически бесполезны в голом Docker



```
$ docker ps
```

CONTAINER ID	IMAGE	COMMAND	CREATED	STATUS
a8d4a6cea27a	docker-health	"nginx -g 'daemon of...'"	About a minute ago	Up About a minute (unhealthy)



Enterprise решения

— **Логи:** ELK, Loki, Splunk





Enterprise решения

Логи: ELK, Loki, Splunk

Метрики: Prometheus, InfluxDB, graphite





Enterprise решения

— **Логи:** ELK, Loki, Splunk

— **Метрики:** Prometheus, InfluxDB, graphite

— **Tracing:** jaeger



Практика!





Практика

Выполните

```
docker run --name test_nginx -v  
$(pwd)/conf.d:/etc/nginx/conf.d/ nginx
```

И поймите почему не работает





Практика

Выполните

```
docker run --name test_nginx -v  
$(pwd)/conf.d:/etc/nginx/conf.d/ nginx
```

И поймите почему не работает

Подсказка 1

```
Docker logs test_nginx
```





Практика

Выполните

```
docker run --name test_nginx -v  
$(pwd)/conf.d:/etc/nginx/conf.d/ nginx
```

И поймите почему не работает

Решение

Исправить опечатку в конфигурационном файле





southbridge.io

Спасибо!

СЛЁРМ

slurm.io