Мониторинг кластера

Слайд 1 — Титул

Мониторинг кластера Kubernetes

Metrics Server, Prometheus, Grafana

Практика: kube-prometheus-stack + просмотр метрик Pod

Слайд 2 — Зачем нужен мониторинг

- Понимание нагрузки на Pod/Node
- Поиск аномалий
- Диагностика просадок производительности
- Аналитика для ресурсных лимитов
- Основной инструмент для SRE/DevOps, чтобы избежать «почему всё упало, блин?»

Слайд 3 — Metrics Server: что это

Назначение

- Собирает *текущие* (not historical) метрики ресурсов: CPU, RAM
- Работает через kubelet /metrics/resource
- Агрегирует и отдаёт API: `metrics.k8s.io`

Используется для:

- `kubectl top`
- Horizontal Pod Autoscaler
- Vertical Pod Autoscaler

Не умеет:

- История

Слайд 4 — Пример работы Metrics Server

Команды:

```
kubectl top nodes
kubectl top pods -n default
```

Вывод: мгновенное состояние, без ретроспективы.

Слайд 5 — Prometheus: основы

Назначение

- TSDB (time-series database)
- Сбор метрик со всех компонентов: kubelet, apiserver, node-exporter, cAdvisor, приложений

Что даёт:

- История
- Гибкие запросы PromQL
- Алерты (Alertmanager)

Слайд 6 — Как Prometheus собирает метрики

- Model: **pull**
- Таргеты (endpoints): `/metrics`
- Формат: текстовый exposition
- Пример: kubelet → Prometheus → Grafana

Слайд 7 — Grafana

Назначение

- Визуализация метрик
- Дашборды для Kubernetes (nodes, pods, requests/limits, network IO)

Что полезно студенту:

- Просмотр ресурсов Pod
- Диагностика перегруженного нода
- Дашборды из kube-prometheus-stack

Слайд 8 — kube-prometheus-stack

Состав чарта:

- Prometheus
- Alertmanager
- Grafana
- Node Exporter
- Kube State Metrics
- Сервисные дашборды

Это "всё-в-одном" стек. Устанавливается быстро, но жрёт ресурсы, так что не удивляйтесь, если кластер вздохнёт погромче обычного.

Слайд 9 — Практика: шаги занятия

1. Установка Helm
2. Добавление репо
3. Деплой kube-prometheus-stack
4. Проверка Pod
5. Порт-форвард Grafana
6. Просмотр метрик Pod через `kubectl top` и через Grafana

Слайд 10 — Установка Helm

```
curl https://raw.githubusercontent.com/helm/helm/master/scripts/get-helm-3 | bash
helm version
```

Слайд 11 — Добавление репозитория

```
helm repo add prometheus-community https://prometheus-community.github.io/helm-charts
helm repo update
```

Слайд 12 — Установка kube-prometheus-stack

```
helm install kps prometheus-community/kube-prometheus-stack -n monitoring --create-namespace
```

Проверка:

```
kubectl get pods -n monitoring
```

Ожидаемые компоненты:

`prometheus-*`, `grafana-*`, `alertmanager-*`, `kube-state-metrics`, `node-exporter`.

Слайд 13 — Доступ к Grafana

Порт-форвард:

```
kubectl port-forward -n monitoring svc/kps-grafana 3000:80
```

Открыть:

`http://localhost:3000`

Логин/пароль:

`admin / prom-operator`

Слайд 14 — Дашборды

В Grafana появляются готовые борды:

- Kubernetes / Compute Resources / Nodes
- Kubernetes / Compute Resources / Pods
- Kubelet metrics
- API Server etc.

На уроке смотрим:
Compute Resources / Pods

Слайд 15 — Просмотр метрик Pod

Проверка через Metrics Server:

```
kubectl top pod -n default
```

Проверка в Grafana:
Kubernetes → Pods

- CPU usage
- Memory usage
- Requests/limits
- Throttling

Слайд 16 — Отладка частых проблем

1. **Пустые графики** → Prometheus не может достучаться до target
2. **Pod Pending** → kubelet не отдаёт метрики, из-за чего Grafana показывает нули
3. **Высокие CPU spikes** → проверяйте throttling
4. **Prometheus OOMKilled** → запросы слишком тяжёлые (PromQL пожирает память)

Слайд 17 — Мини-резюме урока

Факты:

- Metrics Server = быстрые метрики без истории
- Prometheus = вся история метрик
- Grafana = удобные дашборды
- kube-prometheus-stack = готовый набор мониторинга
- Практика: установка стека + просмотр метрик Pod

Если в кластере что-то жрёт CPU как бешеное — теперь это видно.