Автоматическое масштабирование

Слайд 1. Задачи урока

- Понять механизмы автоматического масштабирования.
- Разобрать HPA, VPA, Cluster Autoscaler.
- Настроить HorizontalPodAutoscaler на практике.

Слайд 2. Зачем нужно автоскейлинг

- Уменьшает расходы: меньше Pod'ов — меньше ресурсов.
- Устраняет деградацию при росте нагрузки.
- Делает систему предсказуемой по SLA: Pod'ы появляются автоматически.

Слайд 3. HorizontalPodAutoscaler (HPA)

Функция: регулирует *количество Pod'ов*.

Основы работы:

- Следит за метриками (CPU, RAM или кастомные метрики).
- Если метрика превышает target — увеличивает реплики.
- Если падает — уменьшает.

Требования:

- Metrics Server.
- Deployment / ReplicaSet как объект-таргет.

Слайд 4. Как HPA принимает решение

1. Считывает текущее значение метрики с Pod'ов.
2. Рассчитывает «насколько» планка превышена.
3. Выбирает нужное количество реплик.
4. Обновляет `spec.replicas` у Deployment.

Формула упрощённая:

```
desiredReplicas = ceil(currentMetric / targetMetric * currentReplicas)
```

Слайд 5. VerticalPodAutoscaler (VPA)

Функция: меняет *ресурсы Pod'а* (CPU/Memory requests/limits).

Особенности:

- Полезен для сервисов с непредсказуемым потреблением RAM.
- Перезапускает Pod для применения новых значений.
- Нельзя одновременно включать VPA и HPA на одну метрику запросов CPU — конфликт логики.

Слайд 6. Cluster Autoscaler

Функция: масштабирует *узлы кластера*.

Работает так:

- Если Pod не может запуститься из-за нехватки ресурсов — добавляет узлы.
- Если узел простаивает — удаляет его.

Условия:

- Поддерживаемый облачный провайдер (AWS, GCP, Azure и т.д.).
- Правильная настройка групп узлов.

Слайд 7. Как все три работают вместе

- HPA → регулирует количество Pod'ов.
- VPA → регулирует ресурсы каждого Pod'а.
- Cluster Autoscaler → регулирует количество узлов.

Типичный сценарий:
Рост нагрузки → HPA увеличивает Pod'ы → узлы заполняются → Cluster Autoscaler добавляет ноды.

Слайд 8. Практика: настройка HPA

Пример Deployment:

```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: demo
spec:
  replicas: 1
  selector:
    matchLabels:
      app: demo
  template:
    metadata:
      labels:
        app: demo
    spec:
      containers:
      - name: demo
        image: nginx
        resources:
          requests:
```

Слайд 9. Практика: манифест HPA

```yaml
apiVersion: autoscaling/v2
kind: HorizontalPodAutoscaler
metadata:
  name: demo-hpa
spec:
  scaleTargetRef:
    apiVersion: apps/v1
    kind: Deployment
    name: demo
  minReplicas: 1
  maxReplicas: 5
  metrics:
  - type: Resource
    resource:
      name: cpu
      target:
        type: Utilization
        averageUtilization: 70
```

Слайд 10. Проверка работы HPA

Команды:

```
kubectl get hpa
kubectl describe hpa demo-hpa
kubectl top pods
```

Нагрузка для теста:

```
kubectl run load --image=busybox -- sh -c "while true; do wget -q -O- http://demo; done"
```

Слайд 11. Типичные ошибки

- Нет установленного Metrics Server → метрики не собираются.
- Слишком низкие лимиты CPU — Pod быстро уходит в throttle.
- Неправильные targets → HPA дёргается вверх-вниз.
- Конфликт HPA + VPA по CPU requests.