Санкт-Петербургский государственный электротехнический университет им. В.И. Ульянова (Ленина)

Анализ механизмов отказоустойчивости веб-приложений в платформах оркестрации контейнеров Kubernetes и Docker Swarm

Выполнил: Басин Даниил Дмитриевич, гр. 5303
Руководитель: Лавров Андрей Александрович, к.т.н., ассистент

Актуальность

Актуальность: приложения с микросервисной архитектурой включают множество контейнеров, управлять которыми вручную не эффективно. Для управления используют системы оркестрации контейнерами, такие как Kubernetes, Docker Swarm и др.

Проблема: такие системы из-за разной реализации по разному реагируют на отказы, которые сказываются на времени простоя приложения и его экономической эффективности.

Цель и задачи

Цель: проанализировать реакции систем оркестрации контейнерами Kubernetes и Docker Swarm на отказы разного уровня, предложить улучшения механизмов отказоустойчивости и определить какая из систем предпочтительнее

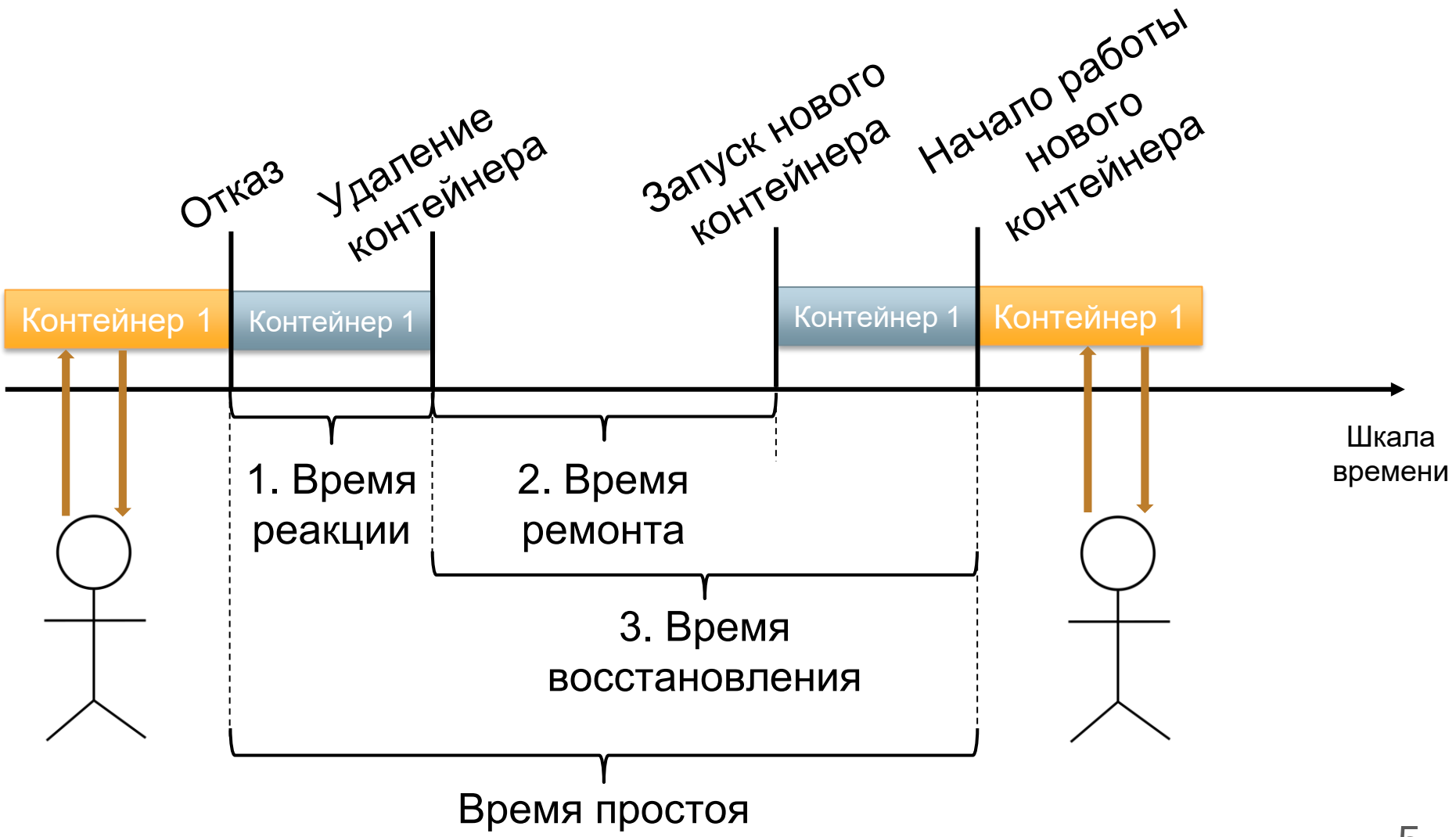
Задачи:

1. Развернуть кластера с максимально близкими конфигурациями с использованием систем
2. Выбрать и запустить контейнеризованное веб-приложение в существующих кластерах
3. Сымитировать отказы уровня узла кластера и уровня объекта, управляющего контейнером
4. Проанализировать результаты и предложить улучшения систем

Конфигурации кластеров

	Docker Swarm	**Kubernetes**
Количество узлов	3 управляющих узла + 2 рабочих узла	
Инструмент для имитации кластера	Docker-Machine	K-in-D
Абстракция узла кластера	Виртуальная машина	Docker-контейнер
Объект управления контейнером приложения	“task” (управляется через “service”)	“Pod” (управляется напрямую и через родительские объекты)

Исследуемые метрики



Сценарии отказов

Уровень	Сигнал	Docker Swarm	Kubernetes
Узел кластера	Внешний	Отключение ВМ	
	Внутренний	docker swarm leave	Kubectl drain node <NODE-ID>
Объект управления контейнером	Внешний	kill <CONTAINER-PID>	
	Внутренний	docker stop <CONTAINER-ID>	kubectl delete pod test-app

Таблица сигналов для имитации отказов

Результаты выполнения сценариев со стандартными настройками

Система	Сценарий		Критерий оценки			
	Воздействие	Объект	Время реакции, сек	Время ремонта, сек	Время восстановления, сек	Время простоя, сек
Docker Swarm	Внутренне	Контейнер	0,043±0,035	0,173±0,306	5,480±1,004	**5,523**±1,039
	Внешнее		0,037±0,005	0,246±0,085	5,365±0,115	**5,401**±0,119
	Внутренне	Узел	12±1	0,19±0,07	5,5±0,8	**17**±3
	Внешнее		11±1	32±2	37±2	**48**±3
Kubernets	Внутренне	Контейнер	0,041±0,008	0,982±0,005	1,547±0,009	**1,59**±0,01
	Внешнее		0,496±0,005	31,5±0,7	32,5±0,8	**33**±1
	Внутренне	Узел	0,031±0,002	1,01±0,03	1,50±0,08	**1,53**±0,09
	Внешнее		38±2	291±15	292±15	**330**±17

Анализ результатов Kubernetes

Причины:

- Долгое удаление отказавшего пода
- Редкий опрос статусов узлов кластера

Улучшения:

- Добавить проверку на работоспособность приложения в контейнере
- Изменить настройки благоприятного завершения пода
- Изменить настройки системы по обновлению статусов узлов

Ожидаемый результат: снижения времени простоя до ~10 сек и до ~20 сек для отказов уровня контейнера и

Анализ результатов Docker Swarm

Улучшения:

- Использование Docker Compose для более точной настройки критериев отказа приложения в контейнере, времени опроса приложения

Ожидаемый результат: положительно скажется на точности определения статуса работоспособности приложения в контейнере

Заключение

Результат: Docker Swarm обеспечивает **меньшее время простоя**, но также **менее гибкий функционал** для управления контейнерами. Поэтому его лучше выбрать **для небольшого приложения** и система позволит затратить меньшее количество ресурсов на настройку.

Планы дальнейшей работы

- Проверка предложенных улучшений на практике
- Аналогичный эксперимент для распределенной высоконагруженной программной системы в качестве полезной нагрузки

Спасибо за внимание!