Методика описание микросервисов Kubernetes на архитектурной схеме ИС.

Список использованной литературы.

1. Kubernetes_in_Action.pdf
2. Kubernetes iznutri - Dzhiei V'ias, Kris Lav.pdf
3. Описание концепции Kubernetes.https://kubernetes.io/docs/concepts

Основные цели и способы описания

Цели

При описании микросервисной архитектуры мы должны ответить на четыре основных вопроса.

1. Какую бизнес-функциональность реализуют микросервисы?
2. Где хранится код реализации микросервисов?
3. Какова структура микросервисной сети кластера?
4. На каких ресурсах в ЦОД развернуты микросервисы?

Способы

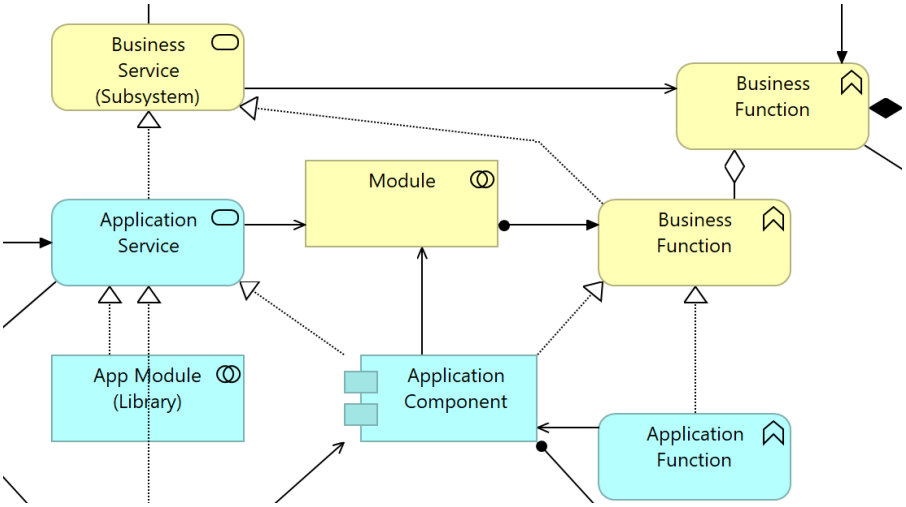
Для ответа на эти вопросы мы располагаем подмножеством нотации Archimate формализованной в документе - схема вертикальной связности Требования (основные) к вертикальной связанности архитектурной модели Продукта.

Базовыми элементами связности являются:

- Программный компонент - опосредовано связывающий исходный код с бизнесом и инфраструктурой.
- Узел - ВМ инфраструктуры ЦОД.
- Бизнес - функция, которая связывает основные требования заказчика со способом их реализации.

Шаблон проектирования в микросервисной парадигме в качестве атомарной единицы проектирования предлагает микросервис, реализующий бизнес функциональность которая имеет четко определенный перечень связей с другой частью функциональности. Таким образом шаблон проектирования накладывает определенные условия на декомпозицию бизнес слоя на котором должны быть выделены бизнес модули для бизнес-функций которые имеют четко определенные связи, определяющие интерфейс микросервиса. Таким образом модули являются постановкой задачи для микросервиса. Такой подход позволяет разделить бизнес задачи функционального заказчика инкапсулированные в бизнес функции от постановки задачу на реализацию микросервиса. Это позволяет избежать скрытых связей между модулями на этапе бизнес проектирования.

Рисунок 1.



Способы описания бизнес-слоя

На основании вышеизложенного можно сформулировать первое требование к бизнес-слою архитектуры, описывающей микросервисное решение.

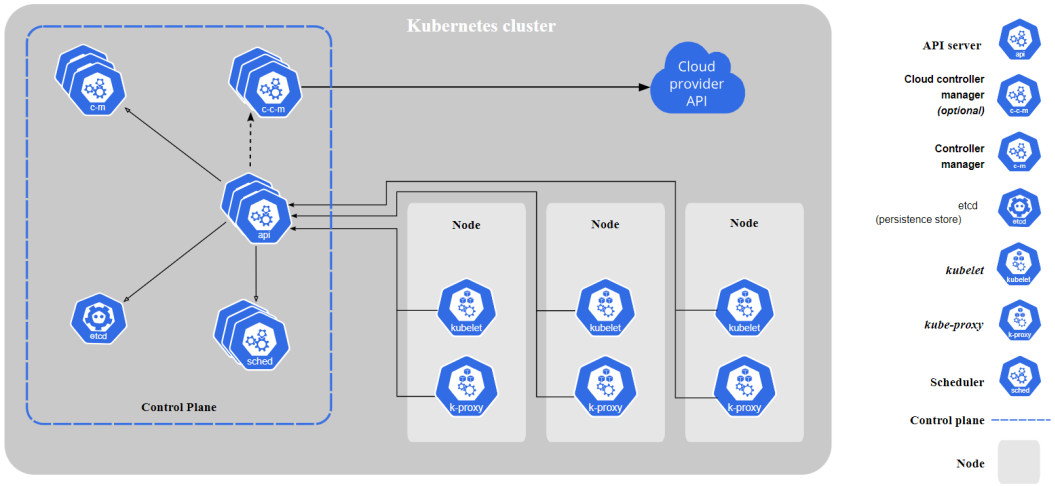
Каждый бизнес-функция должна быть состоять не менее чем из одного бизнес модуля.

Таким образом мы постулируем необходимость дополнительного проектирование бизнес слоя при переходе к микросервисной архитектуре, чтобы избежать ошибок связанных как с излишним дроблением функциональности на микросервисы так и с излишним укрупнением и превращением микросервисов в монолиты. Это дает нам возможность сформулировать критерий целесообразности перехода на микросервисную архитектуру в виде следующего утверждения - бизнес функции системы должны иметь возможность быть декомпозированы на слабосвязанные независимые друг от друга компоненты. Под слабосвязанностью здесь понимается возможность полного или частичного функционирования одного компонента при полной недоступности другого компонента и/или компонентов. Если система этому критерия с точки зрения бизнес-слоя не соответствует и является сильносвязанной, то ее перевод на микросервисную архитектуру нецелесообразен.

Способы описания программного слоя архитектуры

В качестве базового графического представления СПО Kubernetes будем использовать следующую схему https://kubernetes.io/images/docs/components-of-kubernetes.svg.

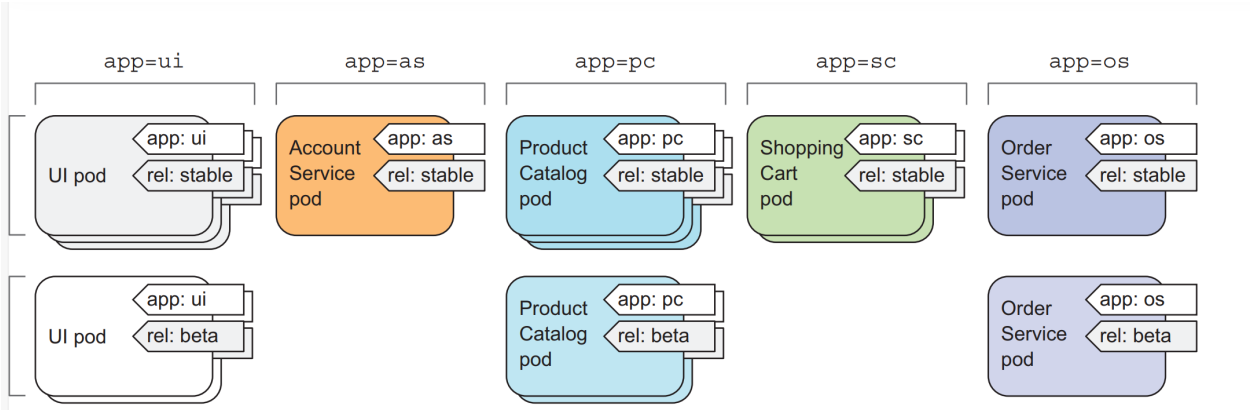
Рисунок 2.



Эта схема группирует все сущности в рамках кластера K8s. Связывает ВМ ЦОД с исполняемыми элементами через Worker Node. Дальнейшую детализацию признаем нецелесообразной т.к. она будет являться общей для каждого кластера Kubernetes, а следовательно не является предметом рассмотрения для программной архитектуры конкретной ИС.

Следующим интересующим уровнем описания кластера исходя из поставленных целей примем сервисы сгруппированные по пространству имен. Здесь в качестве схематического описания сегментации сервиса по пространствам имен примем не официальную документацию, а методические указания Kubernetes in Action.pdf (vk.com).

Рисунок 3



Где app это пространство имен.

Из вышеприведенных положений следует следующее требование к программному слою архитектуры.

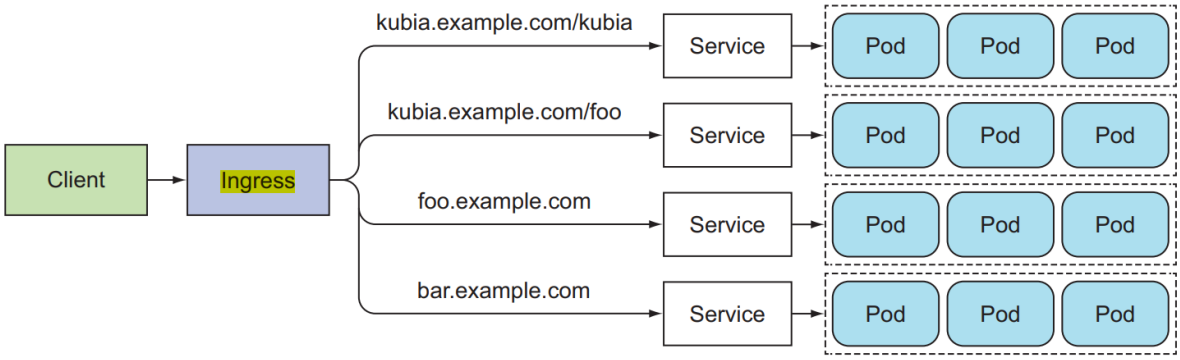
В архитектуре кластера не может быть ни одного сервиса не связанного с одним и только одним пространством имен.

Не может быть ни одного Pod не относящемуся к одному и только к одному сервису.

Более формализовано эти требования будут сформулированы после сопоставления нотации и элементов кластера.

После определения группировки сервисов по пространству имен определим правила формирования и отображения в архитектуре структуры сервисной сети кластера. В качестве методических материалов продолжим использовать Kubernetes in Action.pdf (vk.com)

Рисунок 4

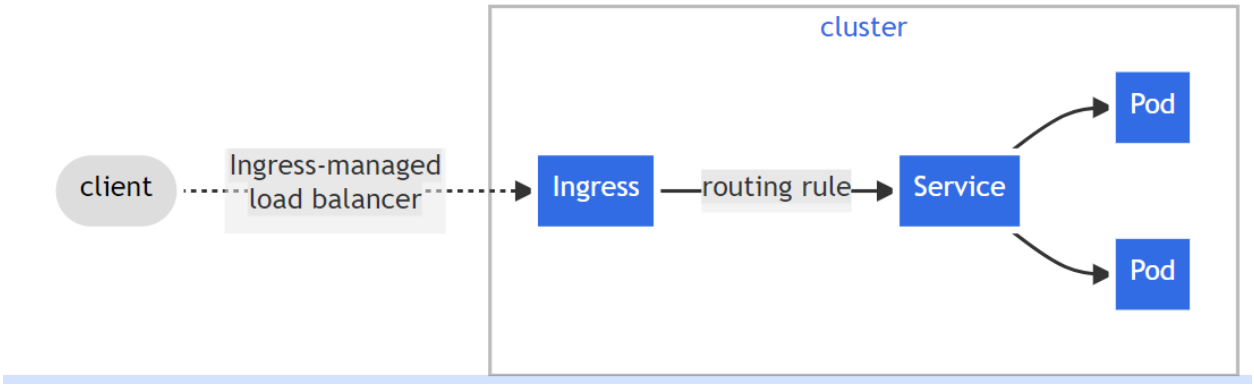


В соответствии с этой схемой примем что единственным способов реализации сети сервисов принимается вышеописанный, поэтому публикация портов с узлов (NodePort Service), задание статических точек доступа (EndPoint) должны быть запрещены.

В соответствии с документом по развертыванию Kubernetes Rancher как услуги ЦОД Начало работы с Kubernetes Rancher в качестве ingress используется Nginx, а в качестве средства мониторинга prometheus передающий метрики на mon-dc.mos.ru. Этот

сервис реализует спецификацию https://github.com/prometheus-operator/prometheus-operator/blob/main/Documentation/api.md#monitoring.coreos.com/v1.PodMonitor). Тогда выверка архитектурной схемы с фактическим развертыванием проводится путем сверки метаданных для POD и архитектурной схемой по таблице соответствия - Таблица 1.

Целевая схема развертываемая как услуга соответствует официальной документации - https://kubernetes.io/docs/concepts/services-networking/ingress/



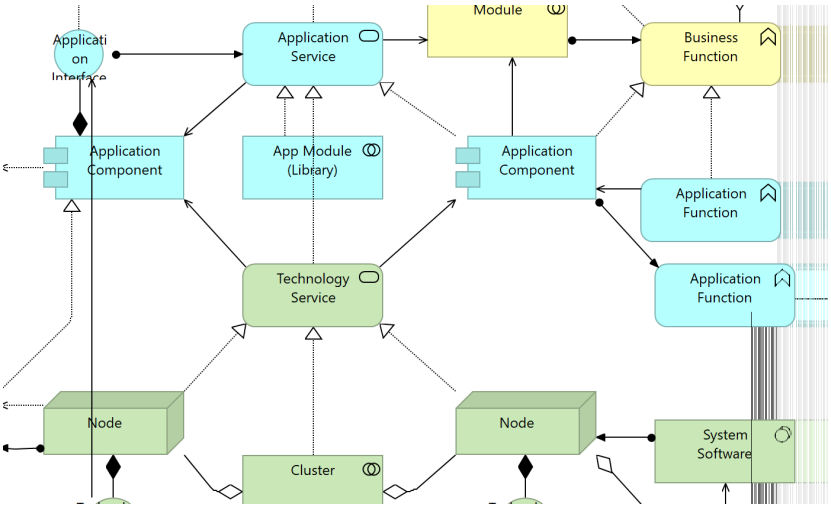
В соответствии с этой схемой выполняется настройка балансировки по инструкции ЦОД Nginx. Постановка ресурса (сайта) на балансировку нагрузки. Пояснения

Сопоставим сущности Kubernetes с элементами вертикальной связности.

Таблица 1.

Сущность Kubernetes	**Сущность Archimate**	**Ссылка на определение Kubernetes**	**Обоснование и ссылка на метрику для выверки фактического состояния и арх. схемы.**
Service	Application interface	kubectl get services	Реализует сетевой доступ к группе исполняемых элементов.https://kubernetes.io/docs/reference/generated/kubernetes-api/v1.24/#labelselector-v1-meta serviceName is the name of the service that governs this StatefulSet. This service must exist before the StatefulSet, and is responsible for the network identity of the set. Pods get DNS/hostnames that follow the pattern: pod-specific-string.serviceName.default.svc.cluster.local where "pod-specific-string" is managed by the StatefulSet controller.
Pod	Application Component	kubectl get pods	Под понятием pod здесь понимается не элемент исполнения, сколько файл задающий конфигурацию ресурса с типом Pod, включающий в себя ссылки на код из СКВ который развертывается в контейнере. Таким образом в этом элементе раскрываются связи напрямую отсутствующие в описании ресурса.
Ingress	Application service	kubectl get services	Сервис с типом ingress через свои характеристики определяет сетевые параметры для группы сервисов **https://kubernetes.io/docs/reference/generated/kubernetes-api/v1.24/#labelselector-v1-meta IngressList v1 networking** https://kubernetes.io/docs/reference/generated/kubernetes-api/v1.24/#ingresslist-v1-networking-k8s-io
Node Worker	Node	*kubectl get nodes*	https://github.com/prometheus-operator/prometheus-operator/blob/main/Documentation/api.md#monitoring.coreos.com/v1.PodMonitor

			NodeSelector
Namespace	Technology Service	kubectl get namespace	https://github.com/prometheus-operator/prometheus-operator/blob/main/Documentation/api.md#monitoring.coreos.com/v1.NamespaceSelector
Persistent Volumes	Data object	kubectl get pv	https://github.com/prometheus-operator/prometheus-operator/blob/main/Documentation/api.md#monitoring.coreos.com/v1.EmbeddedPersistentVolumeClaim
Cluster K8s	Technology Cluster		



Применение методики на примере арх. схемы для выборки и трансформации данных внутри кластера на основе Apache NiFI

Описание бизнес-задачи

В рамках описания структур данных собраны схемы базы данных. Чтобы обеспечить возможность реализации выборки и передачи данных между системами без привлечения подрядчика, необходимо реализовать сервис, позволяющий выбирать и передавать данные с минимальным кодированием.

{}