Краткий анализ услуг вида «Managed Kubernetes» или «Container as a Service (CaaS)»

1. Disclaimer – вводные данные

Согласно договоренностям, во всех описанных кейсах услуга заказана у сервис-провайдера (без детализации - демо доступ или коммерческий, по умолчанию подразумевается, что коммерческий):

- Все базово необходимые ресурсы выделены и оплачены (вычислительные ресурсы BM, IP-адреса, подсети, и др.);
- На ресурсах каждого провайдера настроен кластер K8S в базовой конфигурации;
- Настроены все базово необходимые доступы и утилиты (например, токены для доступа к Control Plane, настройка `kubeconfig/kubectl` на машинах разработчиков и администраторов);
- Формат образов и среда исполнения контейнеров, для простоты – Docker/Docker engine, соответственно. В рамках данного документа дополнительные вводные, например, CR-IO, не рассматриваются.
- Цены не сравнивались в основном из-за слишком разного подхода к ценообразованию у разных сервис-провайдеров;
- Технические нюансы, преимущества и недостатки в основном описывались с точки личного опыта эксплуатации сервисов (поэтому, в частности, больше описано про AWS EKS).

2. Общая схема/принципы деплоя приложения

Детализация и графическая схема последовательности развертывания представлена на слайдах.

Первый слайд - общие подготовительные процедуры, ОБЩИЕ для любого провайдера (шаги, не являющиеся provider-specific). Относится ТОЛЬКО к самому приложению либо к универсальным необходимым шагам. Шаги с пометкой (ИЛИ) на схеме отражены как значки принятия решений

- Подготовить репозиторий с исходным кодом и статическими файлами для сборки приложения;
- Определить базовый образ и написать Dockerfile для докеризации приложения;
- (ИЛИ) Определить реестр для хранения образов контейнеров - не привязанный к ресурсам провайдера либо привязанный (связанный провайдерский сервис);
- (ИЛИ) Определиться с доступом к приложению - откуда должен быть доступ и по каким каналам (публичный из Internet либо ограниченный по VPN из корп. сети);
- (ИЛИ) Для доступа из Интернет – служба DNS существующая или от сервис-провайдера в составе пакета услуг?
- (ИЛИ) Нужен ли специфичный провайдерский сервис по балансировке нагрузки?
- Для доступа из Интернет - определить, нужен ли доступ по SSL, и, если да - какой требуется сертификат - коммерческого УЦ или публичного (например, LetsEncrypt)?
- (ИЛИ) Собрать контейнер с помощью сервиса провайдера либо собрать его локально и запушить в любой доступный извне Registry. Доступ к чужому Registry - через `kubectl`

`create secret` и затем через декларацию `imagePullSecrets` в deployment-манифесте приложения.

- Создать необходимый набор декларативных YAML-файлов для развертывания приложения - как минимум это Deployment.yml и Service.yml.
    - Deployment описывает само приложение, Service - точку доступа к набору pod-ов.
    - В Service type: LoadBalancer создает L4 балансировщик (без терминирования SSL), kind: Ingress создает L7-балансировщик.
- (ИЛИ) Определить набор переменных окружения (в том числе специфичные для провайдера)
- Собрать `config-map`. После этого при изменении любой переменной достаточно пересобрать `config-map` вместо пересборки всего Deployment-а.
- `kubectl apply -f /.../directory_with_yamls`
- проверка развертывания: `kubectl get service/deployment ...` и проверка доступа к сервису

Второй слайд - специфические куски воркфлоу для каждого провайдера - стрелка сверху с цифрой пункта и описание действий в пункте.

Amazon EKS

1. Краткое описание услуги

Полностью управляемая среда для деплоя, управления и эксплуатации контейнеризированных приложений.

2. Технические параметры и нюансы

EKS - это реализация Managed Kubernetes от AWS, где Control Plane находится полностью под контролем сервис-провайдера, а Data/Worker Plane - на стороне пользователей.

При развертывании по умолчанию создается минимум 2 узла с ролью API Server и минимум 3 узла `etcd`, распределенные по 3 зонам доступности (Availability Zone, AZ)

Для возможности использования `kubectl` требуется установить и настроить специфичное ПО (aws-iam-authenticator)

В каждой AZ по умолчанию разворачивается по одному Bastion Host для доступа по SSH на worker nodes снаружи. Напрямую по умолчанию нельзя!

По умолчанию, рекомендуемая модель развертывания и использования услуги следующая:

- 1 DNS-домен на кластер
- Подключается 1 экземпляр балансировщика ELB - Elastic Load Balancer
- на внешний адрес ELB настраивается Nginx Ingress
- маршрутизация по сервисам настраивается на Nginx Ingress и по путям от основного домена (http://domain.ext/service1/ и так далее)

3. Основные преимущества

Прозрачная интеграция с другими сервисами AWS и единая веб-консоль настройки базовых параметров:

- Amazon ECR в качестве реестра образов для контейнеров
- ELB для балансировки нагрузки (но каждый экземпляр ELB стоит денег и требует минимум 1 белого IP, который тоже стоит денег)
- IAM для аутентификации, авторизации и доступа к ресурсам
- VPC для изоляции на сетевом уровне

Не нужно самим разворачивать и настраивать балансировщики - можно воспользоваться предустановленными в Амазоне с рекомендованными настройками.

Отказоустойчивость by design – kube master и экземпляры базы ETCD размазаны по 3 зонам доступности сразу.

В целом упрощены управление конфигурациями, кластеризация ETCD - автоматически, отказоустойчивость kube master - автоматически.

Пользователи AWS управляют в явном виде только worker-ами.

Control plane находится на стороне и под управлением AWS и ее настройки для пользователей сильно упрощены (что скорее хорошо)

Есть возможность заказать Baremetal-машины в роли worker node.

- Теоретически есть возможность приобрести сервисы AWS с русскоязычной поддержкой на 2 и 3 уровне (от компании OCS), но неизвестно, распространяется ли она в полной мере на EKS.

4. Выявленные недостатки/недоработки и риски

Доступ к control plane (команды kubectl) - по одноразовым токенам. По умолчанию доступ к control plane есть только на шлюзе (bastion host) в терминологии AWS. Это может быть неудобно с точки зрения работы с сервисом.

Для того, чтобы получить доступ к kubectl из любой точки Интернета - нужно проделать ряд действий (не всегда работающих согласно инструкциям).

Проблемы с шаблонами AWS Cloud Formation:

- Шаблоны AWS Cloud Formation для развертывания кластера по умолчанию задействуют 3 AZ почти для всех ресурсов = нехорошо с точки зрения стоимости, если развертывается DEV-среда без жестких требований к доступности и SLA.
- Скорее всего без заказанной услуги S3 шаблоны не заработают - поскольку в S3 хранятся промежуточные темплейты и временные файлы. Образы, использующиеся в Cloud Formation - используют специфичные для Amazon AMI-flavour, работают в ограниченном количестве регионов (в нашем примере - только в Огайо).
- Один темплейт при развертывании подтягивает за собой еще минимум 7! Но без них крайне сложно развернуть EKS руками - много ошибок и несоответствий с инструкциями.

В EKS есть лимиты по количеству инстансов виртуальных машин EC2 (worker nodes), по умолчанию это по 20 на регион каждого типа.

EKS поддерживается полностью не во всех регионах

Минимальный тип инстанса для развертывания ноды кластера - это t3.medium, даже если это DEV-среда и там нулевая нагрузка. На 1ГБ памяти даже пустой кластер рассыпается примерно через сутки.

EKS стоит денег просто за работающий инстанс кластера. Поэтому для небольших сред это невыгодно, выгоднее держать на AWS крупные развертывания, чтобы размазать цену за услугу на общую цену аренды ресурсов.

У AWS есть ограничение по количеству ENI (Elastic Network Interface), которые можно вешать на виртуальный сетевой адаптер машины в EC2. Из этого вытекает ограничение на кол-во подов, которые можно запустить на одной ноде.

Бизнес-поддержка для PROM-развертываний стоит достаточно дорого (и рассчитывается в виде процента от суммарной стоимости услуг AWS)

Для возможности вешать на сервисы собственные сертификаты SSL из коммерческих УЦ (не LetsEncrypt) - нужны дополнительные действия для загрузки своих сертификатов в IAM.

Внешний трафик терминируется на AWS ELB (Elastic Load Balancer) - внешний IP-адрес нельзя прокинуть прямо на контейнер. Для этого есть отдельный продукт AWS NLB, но он пока не предназначен для production-сред.

Нет веб-консоли на узлы кластера, только доступ по SSH - это ограничивает диагностику и затрудняет работу, если worker-ы по каким-то причинам зависли и не отвечают.

5. SLA (основные ключевые параметры)

Нет сквозного SLA. SLA на EKS - покрывает только доступность Control Plane. Все, что относится к доступности worker nodes, ELB, хранилищ и так далее - покрывается отдельными SLA по соответствующим сервисам.

Под недоступностью Control Plane понимается отказ в обработке ВСЕХ запросов на подключение к endpoint-у для соответствующего кластера в течение 5 минут. Расчет фактической доступности = вычет из 100% времени работы кластера в месяц кол-ва 5-минутных интервалов.

Заявленная доступность Control Plane - 99.9%:

- при уровне деградации до 99.0% в месяц - возврат 10% стоимости сервиса в следующий месяц
- при уровне деградации от 99.0 до 95% в месяц - возврат 25%
- при уровне деградации ниже 95% в месяц - возврат 100%

6. Принципы тарификации

Отдельно тарифицируются почти все компоненты, задействованные в сервисе:

- кластеры EKS (просто за факт их создания!)
- внешний балансировщик трафика ELB или ALB плюс параметры (кол-во правил, объем трафика итд)
- инстансы VPC, IAM (сетевая изоляция плюс изоляция по проектам в части auth/autz)
- EC2-машины
- EBS-тома
- EFS-тома, если нужны persistent volumes с режимом ReadWriteMany (это NFS от Amazon)
- внешние IP
- DNS (если пользоваться сервисом Route 53)

Google Kubernetes Engine (GKE)

1. Краткое описание услуги

Полностью управляемая среда для деплоя, управления и эксплуатации контейнеризированных приложений.

2. Технические параметры и нюансы

Сервис от создателей Kubernetes – значит, by design приготовлен правильно (мы, пользователи, на это по крайней мере, надеемся 😊). Старейшее публичное облако для контейнеров (с 2014 года)

Много типов балансировки, в том числе глобальная балансировка трафика для кейсов, где пользователи сервисов в разных точках мира. (https://cloud.google.com/load-balancing/docs/choosing-load-balancer)

В том числе HTTPS Load Balancing - заточенная под HTTP-трафик балансировка, интегрирующаяся с Cloud CDN

3. Основные преимущества

Простой и понятный калькулятор

Нет специфичных провайдерских закладок и vendor lock-in: приложения можно изымать из облака и запускать локально на своем кластере Kubernetes.

Для начала работы требует только учетку Google (есть у всех владельцев Android) - ниже порог входа чем даже у Amazon.

Больше количество поддерживаемых регионов.

Быстрое время создания кластера (по утверждениям Google - 3 минуты, не проверялось).

Можно управлять worker-нодами, в отличие от AWS.

Control Plane ничего не стоит, в отличие от AWS!

Cross-region networking - прозрачно без VPN сразу доступна плоская сеть на все регионы кластера.

Автоматический апгрейд control plane - можно занести как в плюс, так и в минус, но с учетом отсутствия прерывания сервиса - скорее плюс. Широкие возможности по выбору балансировщиков нагрузки.

4. Выявленные недостатки/недоработки и риски

Автоматический апгрейд control plane - можно занести как в плюс, так и в минус, но с учетом отсутствия прерывания сервиса - скорее плюс.

Отсутствие сквозного SLA.

Информация по коммерческой поддержке неясна и неочевидна.

5. SLA (основные ключевые параметры)

Так как GKE сам по себе является бесплатным сервисом, то он не попадает под SLA.

SLA узлов (kubemaster и workers) соответствует параметрам для Google Compute Engine.

Целевой аптайм - 99.5% для сервера Kubernetes API для кластеров внутри зоны доступности (зональные кластера) и 99.95% для растянутых по нескольким зонам доступности (региональные кластеры)

Заявленная доступность BM в GCE - 99.99%

- при уровне деградации до 99.0% в месяц - возврат 10% стоимости сервиса в следующий месяц
- при уровне деградации от 99.0 до 95% в месяц - возврат 25%
- при уровне деградации ниже 95% в месяц - возврат 50%

Процент доступности вычисляется поминутно. Даунтайм охватывает как недоступность внешних подключений либо недоступность диска самой BM, так и недоступность внешних подключений к внешним балансировщикам нагрузки в составе услуги GCE.

6. Принципы тарификации

Тарифицируются:

- Ресурсы Google Compute Engine, необходимые для работы Kubernetes
- При необходимости – сетевые продукты: CDN, Cloud DNS, трафик и правила маршрутизации/балансировки

Dataline Kuberline

1. Краткое описание услуги

1) Интервью, сбор потребностей
2) Договор на комплексную услугу
3) Предоставление вычислительных, сетевых, storage-ресурсов в облаке провайдера
4) Установка кластера Kubernetes и оптимизация его под задачи заказчика
5) Перевод услуги в эксплуатацию

2. Технические параметры и нюансы

До 3 площадок / 3 зон доступности

Persistent Storage - NFS (NetApp FAS/AFF)

Платформа для работы Kubernetes - VMware vSphere + NSX

Параметры compute node:

- vCPU — 2 до 32; RAM — 2 до 512 ГБ.

Параметры по дискам:

- SSD — 2000 IOPS/1000GB; SAS — 250 IOPS/500GB; SATA — 50 IOPS/500GB.

Бекап etcd:

- Периодичность — 1 раз в 6 часов; Глубина хранения — 7 дней.

Возможность использования любых контейнерных бэкендов и сетевых плагинов (нет вендор-лока и искусственных ограничений со стороны провайдера);

Провайдер предоставляет:

- Сертификат для доступа к кластеру и ссылка для скачивания Kubectl;
- Сертификат для HELM;
- Ссылки на документацию по сборке и официальную документацию Kubernetes;
- Ссылку на Dashboard для управления кластером;
- Доступ и ссылка на резервные копии etcd*.

3. Основные преимущества

Коммерческая версия позволяет вынести на уровень провайдера:

- Развертывание кластера
- Обновление Kubernetes в рамках минорных версий
- Решение инцидентов
- Выявление проблем и консультирование по работе с Kubernetes
- Гарантия доступности кластера

- Управление масштабированием ресурсов
- Поддержка актуальности сертификатов
- Управление пользователями
- Резервное копирование ETCD и пользовательских данных
- Установка базового Storage-сервера

Сетевая часть облака Kubernetes реализована на базе VMware NSX-T. Благодаря этому с одной стороны мы имеем промышленное решение, с другой - у разработчиков и девопсов нет оверхеда на настройку NSX: он просто прозрачно работает через NSX Container Plugin - специальный pod, который смотрит на control plane NSX и транслирует туда сетевые команды kubectl.

Промышленные решения для платформы виртуализации, андерлей-сети, для persistent storage (NFS одно из лучших решений для работы с Kubernetes в режиме ReadWriteMany, особенно если бэкенд на NetApp)

Русскоязычная поддержка.

Услуга бекапа.

4. Выявленные недостатки/недоработки и риски

2 AZ (ЦОДа) из трех в Москве (близко географически)

Новая услуга для сервис-провайдера (запуск в 2018) - скорее всего компетенции еще не наработаны в полной мере и на устранение сложных проблем понадобится больше времени.

Нельзя, как у Google Cloud - просто взять и потестировать сервис. Нужно:

- официальный запрос
- интервью для выявления потребностей
- получить ТКП
- сделать заказ
- заключить договор (в т.ч. на тестирование)

Нет публичной оферты на услуги и публичного калькулятора. Расчет идет под конкретного заказчика (как и у большинства отечественных интеграторов).

Нет связанного сервиса по хранению и управлению образами контейнеров (как ECR, GCR) для реализации более комплексной и полной услуги.

5. SLA (основные ключевые параметры)

Есть заявленный сквозной SLA - 99,95%;

Заявленные расчетные сроки создания объектов в K8S:

- Среднее время создания и удаления Pod — до 2 минут;
- Среднее время создания и удаления Service — до 2 минут;

Максимальная недоступность кластера в месяц — не более 22 минут

Среднее время создания и удаления Pod — не более 2 минут

Среднее время создания и удаления Service — не более 2 минут

Целостность данных etcd после восстановления — RPO не более 6 часов, глубина хранения 7 дней

Среднее время доступа к дискам на виртуальных машинах — не более 3 мс для SSD, 20 мс для SAS, 30 мс для SATA

Время реакции на инцидент — до 15 минут

Максимальное время решения критического инцидента — до 120 минут

6. Принципы тарификации

Не исследовались, т.к. нет публичной оферты и калькулятора.

Yandex Managed Service for Kubernetes

1. Краткое описание услуги

Сервис Yandex Managed Service for Kubernetes находится на стадии Preview и не тарифицируется.

Сервис управляет kubemaster-ом и мониторит состояние worker nodes.

2. Технические параметры и нюансы

Для начала работы нужно самостоятельно настроить минимум 1 ВМ в качестве NAT Gateway.

Максимум 3 зоны доступности, все из которой локализованы в европейской части России:

- Московская область
- Рязанская область
- Владимирская область

Ограничения:

- макс 2 кластера, 20 worker nodes, 96GB RAM, 24vCPU на облако (на тенант)
- макс 10 worker nodes, 6 групп узлов на кластер

Собственная утилита для создания и управления кластерами K8S (`yc`), а также API.

3. Основные преимущества

Есть служба Yandex Container Registry и возможность создавать там приватные репозитории контейнеров, которые подключать к Kubernetes.

Собственный сервис IAM (аутентификация + авторизация по токенам, некий аналог AWS IAM)

Большие ресурсы Яндекса по разработке и сопровождению (но только когда сервис выйдет из стадии Preview в стадию GA)

Русскоязычная поддержка

4. Выявленные недостатки/недоработки и риски

Сервис как Container Registry, так и Managed Kubernetes на стадии preview (отсутствие SLA, со всеми вытекающими).

Куцая документация.

Сервис не предназначен для PROD-сервисов

ВМ для worker nodes с более чем 4 vCPU и более чем с 8GB памяти требуют заказа ядер без переподписки, что существенно дороже.

Нет распределенного persistent storage, только локальные диски BM или пространство в объектном хранилище.

Есть публичная оферта на услуги в стадии GA (публичный доступ, а не preview), но не на поддержку.

5. SLA (основные ключевые параметры)

Для preview-сервисов SLA не действует.

Для остальных компонентов услуги сквозного SLA нет, есть составной SLA.

SLA для Compute:

Заявленная доступность - 99.95%

- при уровне деградации до 99.0% в месяц - возврат 10% стоимости сервиса в следующий месяц
- при уровне деградации от 99.0 до 95% в месяц - возврат 15%
- при уровне деградации ниже 95% в месяц - возврат 30%
- Под недоступностью понимается потеря внешней связности по вине Яндекса либо потеря загрузочного диска BM.

SLA для Container Registry:

- Заявленная доступность - 99.95%
- при уровне деградации до 99.0% в месяц - возврат 10% стоимости сервиса в следующий месяц
- при уровне деградации от 99.0 до 95% в месяц - возврат 15%
- при уровне деградации ниже 95% в месяц - возврат 30%
- Под недоступностью понимается сетевая недоступность endpoint-ов либо HTTP-коды ответа 50x на более чем 10% запросов в период наблюдения (не менее 5 минут).

6. Принципы тарификации

Тарифицируются:

- Ресурсы Compute, необходимые для работы Kubernetes
- При необходимости – сетевые продукты: CDN, DNS

Выводы

1. Выбор провайдера

Зависимость от среды:

- если Greenfield-развертывание - то смотреть организационные и политические ограничения. Если таких нет - то лучше Google, если речь идет именно о контейнерах и Kubernetes.
- если legacy - то смотреть другие зависимости и ставить приоритеты. Если провайдер (отечественный) готов помогать в миграции приложений на свою платформу и обладает хорошей компетенцией, не боится публично анонсировать данные по SLA и деградации сервиса - можно рассмотреть его при наличии бюджета.

Зависимость от провайдера:

- Если уже есть DEV и тем более PROD-среды на AWS или Azure - то только ради Kubernetes переходить в Google Cloud смысла нет.
- Если привязки к провайдеру нет - то на стороне Google скорость сети, скорость развертывания ресурсов и в принципе сетевые возможности
- если инфраструктура в компании завязана на сервисах MS (Windows Server, Hyper-V, Active Directory, Exchange, Sharepoint etc...) - то проще всего будет уехать в Azure, как минимум из-за возможности построения федеративной среды.

Зависимость от других факторов:

- Основной посыл для использования отечественных провайдеров в рамках услуги CaaS - это русскоязычная поддержка и санкционные риски.
- Законодательные обязательства, например по хранению ПДн граждан РФ на территории РФ согласно 152-ФЗ.

2. Дальнейшие действия с точки зрения проработки описаний сервисов

- расписать ЗО и RACI-матрицу (в основном если отечественный провайдер либо коммерческая поддержка)
- детализация функциональных возможностей услуги с учетом уточненных требований к рабочим нагрузкам клиентов
- мониторинг параметров доступности и SLA
- детализация показателей эффективности и качества услуги
- расписать детально change management/incident management, особенно действия со стороны провайдера.

3. Дальнейшие действия с точки зрения эксплуатации приложения (не обязательно в указанном порядке)

- продумать масштабирование приложения, спланировать рост нагрузки

- CI/CD, выбор инструментов исходя из многих факторов, не в первую очередь исходя из выбора целевой платформы и провайдера
- прогноз необходимости апгрейда и масштабирования самого кластера K8S
- развитие самого приложения, интеграция с другими приложениями, как stateless так и stateful
- мониторинг/логирование
- безопасность, разграничение и контроль доступа, выполнение требований регуляторов (ПДн, PCI-DSS и не только)
- рассмотреть использование Helm - с помощью helm charts намного удобнее управлять разными средами и конфигурациями в одних и тех же манифестах (банальный пример - среды master и develop) и жизненным циклом деплоя