Ну, и начнём мы с краткого описания самого Kubernetes-а. Мы в прошлом занятии работали с Docker SDK, мы знаем, что такое за Docker платформа, что вы в ней можете собрать какой-то легковесный контейнер. И этот контейнер загрузить с вашим работающим приложением. Таких контейнеров у вас может быть несколько. Кроме приложения у вас могут быть так называемые supplementary контейнеры, там базы данных, какие-то, может быть, сетевые сервисы, какие-то сервисы нагрузки, что-то ещё. В общем, это такая, как бы, зоопарк из приложений, маленький кластер из приложений в вашей системе. Естественно, для production использования этого недостаточно. Потому что серверов у вас может быть несколько и надо как-то организовать взаимодействие с этими приложениями.

У Docker-а есть нативный механизм, называется он Docker Swarm. Вы организуете так называемую сеть из Docker-контейнеров с контроллером, которые друг с другом общаются. И, соответственно, они позволяют вам объединить несколько серверов в такую виртуальную Docker-сеть. У вас там на одном, значит, Docker-хосте, на одном сервере, лежит три Instance-а, три контейнера с вашим приложением. На другом Docker-сервисе у вас лежит база данных. На третьем Docker-сервисе у вас лежит балансировщик нагрузки, который принимает все входящие запросы. Но все вместе они друг друга видят, как единую сеть, и могут общаться, как приложения в одной сети.

Kubernetes берёт эту идею и развивает её ещё дальше. Ребята говорят: «Блин, а что, если мы, значит, сделаем ещё полноценный API, позволим декларативно и очень гибко описывать конфигурации нашей сети. Приделаем к этому API, чтобы с этим взаимодействовать». И всё вместе у нас получился, соответственно, Kubernetes.

Теперь давайте разберёмся с его внутренней структурой. Если мы будем разбирать Kubernetes архитектурно, то мы начнём с так называемой

master-node-ы Кубернетеса. Это его мозговой центр. Это node-а, которая управляет node-ами. Node-а – это как раз наши сервера, на которых уже грузятся контейнеры.

То есть, что здесь есть? Здесь есть server API. То есть, сервер, который предоставляет какие-то команды для управления node-и. Для того, чтобы сам Kubernetes смог взаимодействовать. И в том числе, node-ы могли общаться с так называемой контрольной, контрольным Plane-ом, с master-node-ой.

Kubernetes хранит свои настройки специальным key-value хранилищем, etcd - я его уже упоминал, когда рассказывал про базу данных. Соответственно, это специальное такое высоконагруженное хранилище, которое позволяет вам очень быстро обмениваться данными. Быстро читать и не очень быстро писать. Что, в принципе, идеальный вариант как раз для Kubernetes-а.

Соответственно, есть scheduler, который решает, согласно вашим ресурсам, используемым вашим контейнером, на какой node-е они должны быть загружены. И есть controller manager, который управляет разными типами загрузки этих контейнеров. Мы как раз на него посмотрим чуть-чуть попозже сегодня. Соответственно, node у нас может быть несколько. Должно быть несколько. В этом-то и смысл вообще Kubernetes-а. Если у вас Kubernetes крутится на одной node-е не с тестовыми целями, вам он, на самом деле, не нужен, достаточно будет Docker-сети.

Соответственно, pod-ы – это несколько, один или несколько контейнеров, в которых уже крутятся какие-то ваши приложения, и специальный сервис, называемый kubelet, который общается с самим севером API. И как раз обеспечивает нахождение этих node сети Kubernetes. То есть, это, вот это и есть связующее звено. Соответственно, снаружи на эти node-ы может заходить

какой-то балансировщик нагрузки. Он тоже может быть размещен в Kubernetes-е, а может не быть размещен Kubernetes-е. Как правило, это остаётся за вами. Большинство облачных провайдеров предоставляют свои какие-то решения для балансировщика нагрузки. Ну, и соответственно, после балансировщика нагрузки у вас идут конечные пользователи.

Правила для балансировщика нагрузок, правила, какие, в какой контейнер у вас пойдёт какой запрос, правила, как размещены эти контейнеры, как правило, можно гибко настроить. Иногда с помощью самого Kubernetes, иногда с помощью каких-то там, соответственно, сторонних инструментов, которые в этот Kubernetes легко интегрируются.

Следующий вопрос – это где нам встретить Kubernetes. Но тут ответ простой. Kubernetes можно встретить в так называемых облачных провайдерах. Варианты, когда вам придется поднимать с нуля, настраивать Kubernetes в вашей системе – они имеют место быть, но это довольно-таки редкий случай. Как правило, если вы слышите Kubernetes, это связка с AWS (Amazon Web Services), это Google Cloud, это Azure, это IBM, это Яндекс облако, это Mail облако. В общем, этих провайдеров реально тысячи. Многие из них используют систему, так называемую, OpenShift, которая позволяет вам накатить Kubernetes кластер просто на любой набор серверов, его настроить и продавать как своё собственное решение.

И в принципе, эти сервисы, они – на глубинном уровне я не вижу особых отличий – они отличаются операционными системами там, у них могут быть там свои обвязки вокруг контейнеров, репозиториев. Может быть, свой API для запуска контейнеров. Но по факту там всё одно и то же.

В принципе, они могут использовать даже разные движки для запуска самих контейнеров. То есть, я наверно никого не удивлю, если скажу, что несмотря на то,

что Kubernetes всё ещё поддерживает Docker, как бы полностью, и поддерживает формат Docker-а, под капотом у Kubernetes на самом деле не Docker, под капотом у них штука для запуска контейнеров, называемая containerd, container Daemon. И они планируют, что vendor-ы будут производить ещё какие-то, соответственно, Daemon управление контейнерами, используя Kubernetes, Kubernetes-овский API. Насколько это там близко сейчас к истине, я не знаю, но по факту можно и этого ожидать.

Как правило, разные облачные провайдеры отличаются дополнительными сервисами. Они предоставляют свои блокировщики нагрузок, помимо железа, да. Они предоставляют свои сервисы там, всяких очередей, предоставляют сервисы мониторинга. И это всё у них, как правило, разное. Это, как правило, их собственный vendoring и прочее.

Отличаются ценами. На данный момент, на момент записи видео, насколько мне известно, тут надо быть трижды осторожным с Disclaimer-ами, Google Cloud, AWS и AZURE – вот эта вот большая тройка отличается ценами в следующую сторону. Google – самый дешёвый, дальше идёт AZURE, и AWS самый дорогой. Соответственно, насколько там оно будет в будущем я не знаю.

И заканчивая тем, с чего должен был начать: как правило, если вы покупаете у облачного провайдера именно систему контейнеризации, систему управления контейнерами, то это будет, как правило, Kubernetes. Вы всегда можете откатиться на уровень назад, взять какую-то систему, просто виртуальные сервера и на них поднять Kubernetes, но, как правило, уже это всё встроено. Там есть elastic кластер в Amazon-е, в Google-е это Kubernetes Engine, ну, и так далее. И, как правило, это всё Kubernetes, со своим API и всё там вам будет знакомо.

Я на предыдущих слайдах показывал вот этот самый, сейчас покажу, я показывал API server. Соответственно, API server назван так, потому что он предоставляет API.

И на самом деле, ничего сверхъестественного и мистического в этом API нету. Я, когда только начинал работать с Kubernetes-ом, после там первоначальной настройки, надо было там, что-то там добавить, соответственно, в API. Я думаю: «Блин, что, ну, там же умные дяди сидят, короче, не то, что вам там эти JSON перекладывать». Ну, и соответственно, на самом деле, практически то же самое. Это просто frontend для вашего Kubernetes кластера, спроектированный определённым образом.

На самом деле, это всё равно JSON. Это всё равно обычные запросы, обычное HTTP API с, соответственно, разделением доступов, доступное для контейнера внутри и снаружи. Соответственно, доступно как изнутри, так и для взаимодействия извне, позволяет управлять загруженными pod-ами и вообще загруженными ресурсами Kubernetes-а так называемыми, какие-то настройки добавлять и так далее.

И, что нас интересует, как именно DevOps-ов, оно ещё и умеет расширяться. Первый вопрос, который будет перед нами стоять, для того чтобы нам поиграться с Kubernetes-ом. Для того, чтобы нам посмотреть паттерны вживую, нам нужно этот Kubernetes где-то иметь. Если вы не системный администратор в компании, в которой уже настроен, установлен Kubernetes кластер, либо которая пользуется услугами облачного провайдера, вам нужно где-то этот Kubernetes получить, как-то с ним поиграться и, желательно, бесплатно. Вариантов у нас довольно много. Давайте на них посмотрим. И для себя вы определите подходящий вариант. Я кратенько их разберу и поясню, какой стоит, когда брать.

Начнём мы с моего любимого облачного провайдера, как ни странно. Принято считать Google корпорацией зла, но я считаю, что Google Cloud довольно-таки успешен, как облачный провайдер, предоставляет довольно много сервисов и услуг. Не хватает ему немножко привязки там по зонам, по регионам, маловато их присутствия. Но, тем не менее, я считаю это гораздо более удачным выбором, чем default name, AWS (Amazon Web Services).

Итак, Google позволяет нам бесплатно получить 300 долларов в виде free credits, то есть, в виде такого некого кредита на использование вашего кластера. И получает 20, позволяет получить 20+ бесплатных продуктов.

В развитии Kubernetes-а нас интересует только Kubernetes кластер, как бесплатный продукт. Соответственно, Google даже даёт некое такое вот сравнение, что, если у вас, значит, бесплатный кластер, так называемый Автопилот single-зона у кластера. То есть, кластер нераспределённый географически, который сам себя регулирует по ресурсам. Ну, то есть, на самом деле мы понимаем, да, что это минимальные какие-то ресурсы, то за 74.40 из этих бесплатных кредитов вы будете платить 0 долларов в месяц в течение, получается, трех месяцев использования этого кластера. Что довольно-таки неплохо.

Соответственно, если вы хотите там уже зонировать и распределять эти кластеры дальше, вам начнут приходить счета. Опять же, здесь можно посмотреть на разные зоны этих кластеров. Они отличаются периодически в большую или меньшую сторону, в зависимости от нагрузки самого кластера. Вам это, думаю, неважно. Таким, default-ной зоной для Google-а является либо Europe West, Лондон или Франкфурт, поскольку у них довольно низкие цены, либо просто выбираете зону ближе всего к вам. К России, соответственно, ближе всего, естественно, Лондон, Франкфурт или Цюрих. Но в Цюрихе, ну, мы сами понимаем,

жизнь в Швейцарии дорогая. Облака не исключение. В общем рекомендую, в принципе, этот вариант, если вы хотите просто зайти в консоль, создать себе Kubernetes кластер. Он там прям так и называется Kubernetes кластер и не морочиться, просто попробовать Kubernetes. Из минусов этого кластера могу назвать гугловский toolchain, который довольно сложно авторизируется. Но, если, в принципе, с ним разобрались один раз… Вернее, он сложно авторизировался – насколько я помню они там что-то поменяли. То есть, теперь это просто single sign-on через браузер. Открывается браузер, вы подтверждаете, и всё у вас работает дальше. Мы пользовались гугловским кластером довольно долго. Нам всё нравилось, бы дороговато, но для ваших целей он точно подойдёт.

Следующее, что я могу предложить это IBM Cloud, такой непопулярный довольно вариант, но довольно дружелюбный к новичкам. IBM очень хочет выйти на рынок облачных технологий, поэтому довольно активно пиарит своей продукт. И пиарит, в том числе, давая бесплатный доступ. Соответственно, они в бесплатном тире не ограничивают вас по кредитам и просто ограничивают вас по ресурсам. То есть, дают один кластер в Kubernetes на 30 дней, дают 25GB Storage-а. Соответственно, дают какой-то доступ к их системе, значит, машинного обучения. Этот самый известный IBM Watson. Соответственно, довольно удобно.

Из минусов: кластер стартует, ну, очень долго. То есть, по сравнению с Google-ом, который там поднимает вам виртуальные машины и накатывает на них Kubernetes. Ну, то есть, не накатывает, естественно, поднимает вам, в общем, node-ы Kubernetes-а за там буквально 2-3 минуты. С IBM можно ждать и 10, и 20, довольно долго, но зато бесплатно. Вас никто никуда не гонит, и, соответственно, сколько хотите, столько играетесь.

Опять же, кроме того, что кластеры поднимаются долго, у IBM есть, опять же, свой toolchain. Я давно с ним не работал. Когда я с ним работал, он был довольно

ужасен. То есть, там надо было очень много операций руками делать. Чуть ли не копировать вот, ключи доступа. Они там выдают, надо их скопировать и вставить, значит, в другую тулзу, чтобы получить доступ к кластеру. Не очень было удобно. Наверное, сейчас несколько более удобно.

Требует, соответственно, аккаунта на IBM и вам периодически будет приходить их спам, он ненавязчивый, но он есть. Поэтому здесь всё неплохо. Могу рекомендовать, если вы хотите там спокойно, не торопясь, разобраться в облаках.

Для людей, живущих в России, соответственно, попадающих под действие Российских законов, наверное, есть смысл рассмотреть ещё и российских облачных провайдеров. Например, Yandex.Cloud. Yandex.Cloud предоставляет бесплатный триал, с ограничениями, но Yandex.Cloud в триале довольно-таки много ресурсов предоставляется вам для экспериментов.

Удобно, если у вас есть уже какая-то там, типа, инфраструктура, с которой вы просто хотите поэкспериментировать вообще, взлетит или нет. И, если взлетит, то сколько будет ресурсов потреблять. Опять же, на IBM Cloud и Google Cloud у вас скорее всего не хватит просто системных мощностей, чтобы там всё это запустить. На Yandex.Cloud довольно-таки внушительный, внушительный объем виртуальных машин и процессоров. На Cloud можно что-то уже такое внушительное запустить. Даже будет работать.

И, опять же, если вы в России, то скорее всего у вас есть требование хранить данные пользователей в России. Yandex.Cloud это предоставляет из коробки. Это их такое главное преимущество. Они даже на различных веб-сайтах, поясняют, что, если вы хотите, значит, IT в России, то Yandex.Cloud ваш лучший выбор. Потому что мы всё храним в России, вы никакие законы нарушать не будете.

Ни разу не работал, если честно, с Yandex.Cloud. На бумаге это выглядит очень внушительно. Мои коллеги, которые работали с Yandex.Cloud, говорят, что это довольно удобно. Естественно, после доработки напильником некоторых вещей там, связанных, опять же, со сторонними сервисами Yandex-а самими, которые там надо прокидывать в облако. Не всё там идеально. Но trial-ка жестко ограничена 60-тью днями. Это короткая довольно trial-ка. После этого вас либо попробуют перевести на paid, либо, соответственно, выкинут с облака. Сохранят наши данные на 60 дней, потом всё удалят. Опять же, с нашими, с учебными целями довольно-таки неважно, что вы используете. Но, опять же, если у вас цель работать в Российской компании, которая будет хоститься в облаках, Yandex.Cloud, наверное, будет лучшим выбором.

Естественно, нельзя пройти мимо Amazon-а. Соответственно, у Amazon-а тоже существует бесплатный тир – 12 месяцев, 750 часов. То есть, один месяц бесплатного использования, соответственно, одного instance-а, 5 GB в их storage-е. И опять же, те же 750 часов для базы данных маленького объема.

Ну, я могу порекомендовать Amazon из всех этих сервисов, только если у вас есть задача именно познакомиться с AWS, посмотреть на их, так называемый, vendor provided решение, на их экоструктуру, экосистему. Экоструктуру, классное слово. Экосистему. И, ну, не поехать, соответственно, головой, когда вы будете, когда вы будете уже, непосредственно, деплоить и разворачивать ваше окружение в реальных условиях, в реальных production условиях. В принципе, позволяет познакомиться со всеми фичами AWS и подготовить вашу инфраструктуру к переезду в облако, если вы ставите своей целью именно Amazon.

Для активных триалистов существуют всякие там business discount-ы. Это всё обсуждается с техподдержкой. Можно получить после бесплатного периода довольно классные оферы на Amazon. Если вы какой-то open source проект или

какой-то прорывной проект, вы можете ещё годик пожить бесплатно. Для ваших учебных целей, опять же, я бы брал Amazon, только если есть цель зайти именно в Amazon. В других случаях я бы брал Google Cloud или IBM решения.

Из сервисов, которые можно установить теперь на вашу машину и бесплатно пользоваться, сколько вам угодно, сколько вам нравится, экспериментировать, рушить, ломать, не париться, что вы вдруг получите счёт за то, что вы нечаянно активировали какую-то фичу – у нас существует два решения. Это, соответственно, minikube и Docker Desktop.

Соответственно, начнем с minikube. Minikube требует довольно много ресурсов на машине. То есть, если у вас какой-то старый ноут или какая-то не очень мощная машина, вам будет сложно на нее установить minikube. Уже эти самые node-ы, control plane Kubernetes-а, про который я говорил, требует, как минимум, этих ресурсов.

Соответственно, вам понадобится ещё установленный Docker и какая-то виртуальная машина. Соответственно… Или какая-то, вернее, виртуальная машина. Соответственно, как правило, Kubernetes ставят через hyperkit либо virtualbox. Ну, соответственно, довольно простое, простая установка здесь. Просто по порядку запускаете команды. И здесь даже приведены примеры, как этим самым minikube-ом начать пользоваться. Рекомендуется пользователям Mac-а, потому что, я знаю, что у пользователей Mac-а существуют проблемы с производительностью Docker for Desktop. На Винде я использую следующий вариант, о котором сейчас поговорим. Вариант для ленивых людей, то есть, для людей как я, если у вас уже установлен Docker for Desktop, дальше ходить вам не надо. Вы, особенно если он у вас не на Mac-е, на Mac-е я знаю, что включенный Kubernetes на Docker for Desktop ест прямо очень много ресурсов. Вы можете, соответственно, зайти во вкладочку Kubernetes, просто тыкнуть Enable Kubernetes,

и кластер у вас поднимется самостоятельно. Он там попросит вас что-то скачать, что-то будет качать и так далее. Соответственно, для меня это самый удобный способ, на Винде Kubernetes не ест почти ничего. Вот мои там потребления ресурсов Docker Desktop-а на дефолтном кубернетесовском control plane, где ничего не происходит в данный момент.

Соответственно, я рекомендую этот способ, если вы хотите просто познакомиться с Kubernetes-ом, у вас есть Docker for Desktop, есть какая-то машина, не надо никуда далеко ходить, где-то там регистрироваться, какие-то свои данные оставлять, карты привязывать, вообще ничего не надо. Включайте Kubernetes, экспериментируйте. Рано или поздно у вас кончатся ресурсы на компьютере, вот тогда и подумаете об облачных решениях.

Соответственно, этот способ я буду использовать для показывания примеров в данной лекции. Вы вольны зарегистрироваться в облаках, использовать, что угодно. Особо отличаться оно не будет. Как бы, принципиально Kubernetes, как я и говорил, не особо отличается. Отличаются только его различные обвязки, которые нас сейчас не очень интересуют.