Cloud Thinking

# Модуль 7, урок 2


В этом уроке мы обсудим IoT-сервисы Microsoft и реализацию CI/CD для Azure Iot Edge.

Возможности подключения и аналитика

### Центр Интернета вещей Azure

Подключайте, администрируйте и масштабируйте миллиарды устройств Интернета вещей, перенося их из пограничной среды в облако.

Возможности подключения и аналитика

### Azure IoT Central

Ускорьте создание решений Интернета вещей и сократите затраты времени и средств на управление решениями Интернетом вещей, их эксплуатацию и разработку.

Поддержка пограничных сред и устройств

### Azure Percept

Accelerate edge intelligence from silicon to service.

Возможности подключения и аналитика

### Azure Digital Twins

Создавайте решения нового поколения для пространственной аналитики Интернета вещей, реплицируя реальное физическое пространство и создавая подключенные среды.

Поддержка пограничных сред и устройств

### Azure IoT Edge

Расширение охвата средств облачной и традиционной аналитики за счет переноса рабочих нагрузок и бизнес-логики из облака на пограничные устройства.

Поддержка пограничных сред и устройств

### Azure Sphere

Безопасность подключения устройств с микроконтроллерами — от микросхемы до облака.

Поддержка пограничных сред и устройств

### Windows IoT

Долгосрочная поддержка ОС и службы для управления устройствами.

Поддержка пограничных сред и устройств

### Azure RTOS

Простота подключения и разработки внедренных решений Интернета вещей.

Сегодня бизнес видит выгоду в IoT-проектах, и все больше проектов переходят из пилотных стадий в коммерческую эксплуатацию.

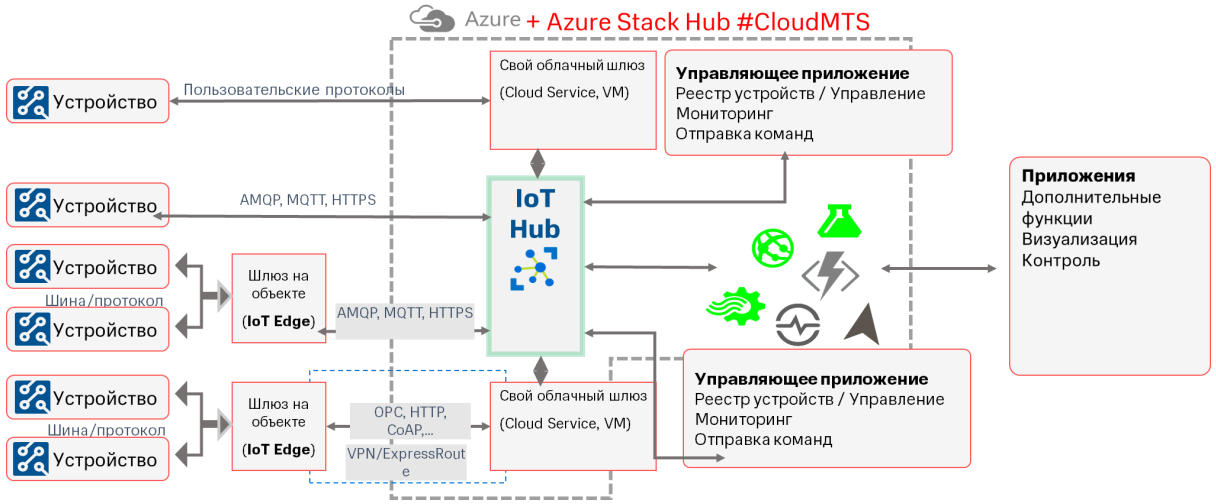
Компания Microsoft внимательно следит за этим трендом, выпускает и поддерживает полный стек продуктов, которые позволяют обеспечить как сбор информации с датчиков и сенсоров, так и ее последующую визуализацию. Также компания предоставляет множество сервисов, связанных с IoT: например, цифровые двойники или Windows IoT.

Сегодня мы разберем три сервиса:

- **Azure IoT Hub** — PaaS-сервис, который можно использовать из облака #CloudMTS.
- **Azure IoT Central** — облачный сервис, предоставляемый по модели SaaS.
- **Azure IoT Edge** — позволяет выполнять нагрузки на сайте заказчика на удаленных территориях.

Cloud Thinking

# Azure IoT Hub




На схеме видно, что Azure IoT Hub находится в центре архитектуры. Его задача — не только принимать и передавать данные. Рассмотрим его функции подробнее.

> Этот сервис находится в глобальном Azure, но также его можно запросить в тест у #Cloud MTS.

Слева расположены устройства, передающие информацию, которую принимает IoT Hub. Они могут не знать новых протоколов, поэтому, чтобы преобразовать данные, мы будем использовать облачный шлюз.

Многие устройства уже поддерживают протоколы AMQP, MQTT, HTTPS, по которым работает IoT Hub. Также мы можем использовать компонент Azure IoT Edge. Устройства могут пересылать телеметрию, свои данные на шлюз Edge, откуда они по протоколам AMQP, MQTT, HTTPS уйдут в Azure IoT Hub. IoT Hub передает эти данные в хранилище, дополнительно можно подключить компоненты для аналитики этой информации.

Отличие от open-source-решений в том, что архитектуру IoT-решений можно собрать из разных кубиков, как LEGO, и подключить к этому потоку данных различные сервисы. Например, Stream Analytics. Он подойдет, если нам нужно в большом потоке информации сразу искать какие-то закономерности или, например, отслеживать какие-то данные.

**Пример.** По шоссе едет большой поток машин, из которого нам нужно выхватывать номера. Для этого мы будем использовать Stream Analytics.

Также можно собирать эти данные, хранить их и на базе собранной информации обучать модели и давать заказчику визуализацию того, что получилось, и аналитику.

Azure хорош тем, что все решения подстраиваются индивидуально под заказчика и состоят из различных компонентов. IoT Hub служит не просто центральным решением, MQTT-брокером. Он также отвечает и за другие задачи.

## Azure IoT Hub:

**Глобальное масштабирование и интеграция**

- Миллиарды сообщений
- Миллионы устройств
- Масштабирование вверх и вниз
- Многоязычный open source SDK
- Запросы и задачи
- Azure-монитор

**Двусторонняя связь с устройствами**

- Отправка данных и получение команд
- Управление устройствами
- Цифровые двойники устройств
- Загрузка файлов
- Приоритизация сообщений
- HTTPS/AMQP/MQTT

**End2end Security**

- Авторизация на уровне устройства с использованием сертификатов
- Возможно вкл/выкл каждое устройство индивидуально
- Поддержка TLS
- Поддержка X.509
- IP белые/черные листы
- Политики общего доступа
- Обновление прошивок и ПО
- Поддержка частных сетей
- Defender для IoT

В Azure есть более дешевый сервис — Event Hub. Он принимает все события и отправляет их, например, куда-то в хранилище. Azure IoT Hub стоит дороже. Вот с чем это связано:

- **Двусторонняя связь с устройствами**

Он позволяет не просто собирать и складывать информацию в хранилище, но и отдавать команды на устройства и управлять ими. Например, через IoT Hub можно дать команду, которая остановит электрическую помпу или уменьшит ее пропускную способность.

- **Цифровые двойники**

IoT Hub хранит в себе цифровые двойники всех устройств, к которым он подключен (порядка 1 000 000 устройств). Цифровой двойник — это JSON-файл, в котором записан ID устройства, статус, прошивка и другая информация. Иными словами, это цифровой слепок реального устройства. При изменении каких-то параметров цифровые двойники тоже будут меняться. Это позволяет, не залезая в устройство, через IoT Hub посмотреть, в какой оно сейчас конфигурации.

- **Масштабирование**

Представим, что мы реализовали пилот и он удовлетворил требованиям заказчика — пора запускать в продуктив. На этом этапе вы можете столкнуться с дилеммой. Сколько серверов нужно купить? Какова будет пропускная способность? Как за этим всем следить и как это обслуживать?

С IoT Hub такой операционной проблемы просто нет. После того как мы реализовали пилот с использованием IoT Hub и других компонентов, этот же инстанс можно перевести в продуктив и расширить пропускную способность IoT Hub до миллиардов сообщений. В самом начальном варианте тарифный план IoT Hub позволяет пропускать 400 тысяч сообщений в месяц, а в самом дорогом — 300 миллионов. Тарифные планы можно группировать и наращивать количество сообщений.

IoT Hub масштабируется как вверх, так и вниз, в зависимости от нагрузки.

- **Многоязычный open source SDK**

Можно разрабатывать собственные сервисы, подключать к IoT Hub свои приложения и датчики.

- **Безопасность**

IoT Hub обеспечивает авторизацию на уровне устройства с использованием сертификатов.

- **Индивидуальное включение/выключение устройств**

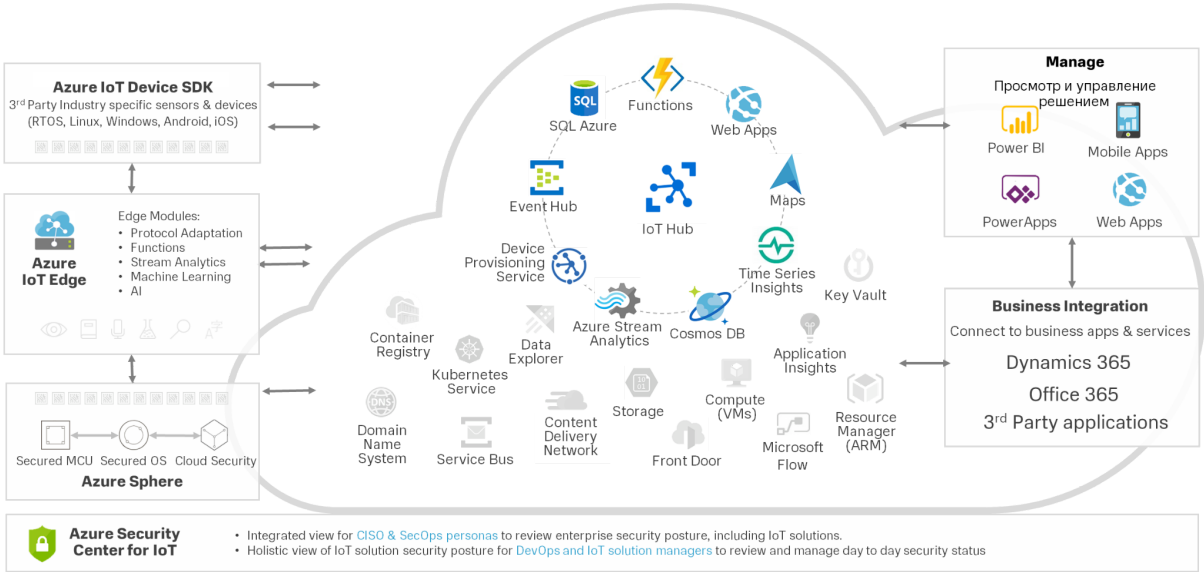
Через IoT Hub, поскольку двусторонняя связь уже есть, каждое подключенное устройство можно включать и выключать. Зачем это нужно? Допустим, к IoT Hub у нас подключено 1000 устройств и некоторые из них «пропали». Они могли исчезнуть из-за какого-то физического воздействия или их могли украсть, чтобы попасть в сеть. Устройство перестало посылать сообщения, и мы не можем проверить, что с ним. В такой ситуации можно отключить его от IoT Hub. Даже если злоумышленники перепрошьют его и захотят подключиться к сети, у них ничего не выйдет. Ведь IoT Hub блокирует доступ от этого устройства, и его невозможно подключить в IoT-архитектуру.

Для одного из наших (Microsoft) заказчиков это было очень важно, поэтому он перешел с другого решения на IoT Hub, потому что он давал возможность контролировать все удаленные устройства и повышать уровень безопасности.

- Обновление прошивок и ПО

Через IoT Hub можно загрузить новую прошивку массово, сразу на множество устройств.

# Azure IoT Central




**IoT Central** — это SaaS-сервис. Нам не нужно собирать единое решение из множества компонентов. С одной стороны, это плохо, ведь такой сервис может подойти не под все требования заказчика. С другой — хорошо, так как IoT-проекты сами по себе не простые, они состоят из многих компонентов,

которые начинаются от разработки прошивки под конечное устройство, разработки ПО (куда эти устройства будут пересылать информацию, где ее хранить, как обрабатывать и как дальше передавать). Иными словами, это достаточно трудоемкие и затратные проекты.

IoT Central состоит более чем из 30 сервисов: хранилище, Event Hub'ы, Stream Analytics, Cosmos DB, Time Series Insight, даже есть карты, которые можно использовать в приложениях по отслеживанию доставки грузов.

Да, IoT Central не позволяет делать тонкую настройку (например, «перекрасить» выбранные кнопки в зеленый).

Также IoT Central обладает возможностью двусторонней связи. С помощью IoT Central можно посылать на устройства управляющие сигналы (конечно, если устройство само поддерживает двустороннюю связь).

# Преимущества IoT Central:

- Все компоненты уже собраны в одном месте и находятся в IoT Central.
- Результат IoT Central — ссылка, по которой открывается дашборд или графики, которые мы ранее составили и где определили, какие параметры выводить и пр.
- IoT Central, в отличие от других решений, обладает конечной стоимостью. На момент составления урока цена одного датчика начинается от 6 рублей за 400 сообщений в месяц. Иными словами, стоимость можно рассчитать «на берегу», на этапе планирования проекта.
- IoT Central также позволяет делать небольшие проекты. Например, перед вами стоит задача по мониторингу оборудования (автотранспорта, холодильников, нагревателей и так далее). Если этих устройств у вас не десятки тысяч, IoT Central станет оптимальным решением с предсказуемой стоимостью.
- Кроме того, IoT Central — решение Enterprise-класса. Он гибко масштабируется, а поскольку сервис находится в «большом» Azure, обеспечивается его надежность и безопасность.


Не все IoT-решения и нагрузки можно разместить в облаке. Вы можете столкнуться с индустриальными требованиями, согласно которым данные должны находиться в периметре компании. Также бывают ситуации, когда нужно срочно принять управляющее воздействие (например, закрыть заслонку) и на ожидание нет даже миллисекунд, за которые сигнал уйдет в облако, вернется обратно, выполнится действие, — нужен мгновенный результат на площадке.

В этом случае можно использовать технологию Azure IoT Edge. Она работает в паре с IoT Hub или IoT Central — вы можете выбрать, с чем эта технология будет работать. Состоит IoT Edge из:

- среды выполнения IoT Edge Runtime — это management-софт, который управляет контейнерами, размещенными на Edge-устройстве, и определяет, какие должны быть контейнеры, какую информацию и куда нужно направить и т. д., то есть занимается управлением;
- слева вы можете снова увидеть датчики;
- справа — вся собранная информация, которую можно хранить в IoT Edge или отправлять в IoT Hub/IoT Central

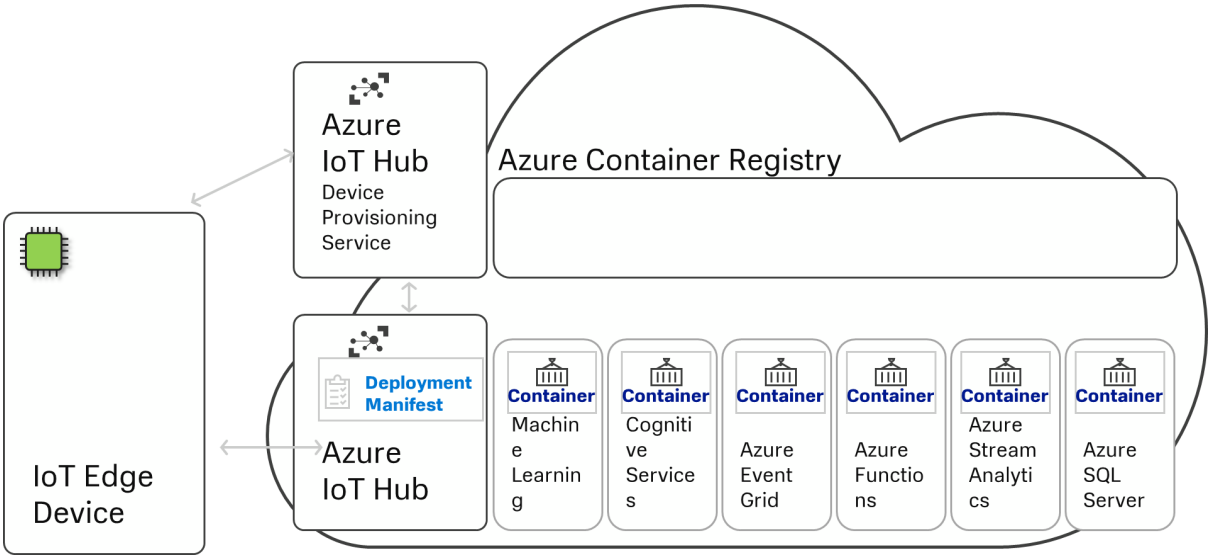
Требований к аппаратной части, на которой должен размещаться IoT Edge, нет — его можно разместить даже на RaspberryPi. IoT Edge поддерживает Linux и Windows ОС и построен по контейнерной (модульной) технологии. Эти контейнеры размещаются на Edge-устройстве и работают в соответствии с нашим планом.

Поддерживается многоязычность — C, NodeJS, Python и пр.

Может быть сконфигурировано как «устройство» (IoT Hub видит его как устройство и принимает с него поток информации) и как gateway.

IoT Edge может работать в офлайн-режиме — если интернет пропадет, все зависит от того, какого объема диск расположен на площадке, чтобы информация от IoT Edge там сохранялась. По документации, IoT Edge может находиться в офлайне 2 000 000 000 секунд, то есть около 60 лет. При восстановлении связи IoT Edge передает телеметрию в IoT Hub / IoT Central и продолжает функционировать.

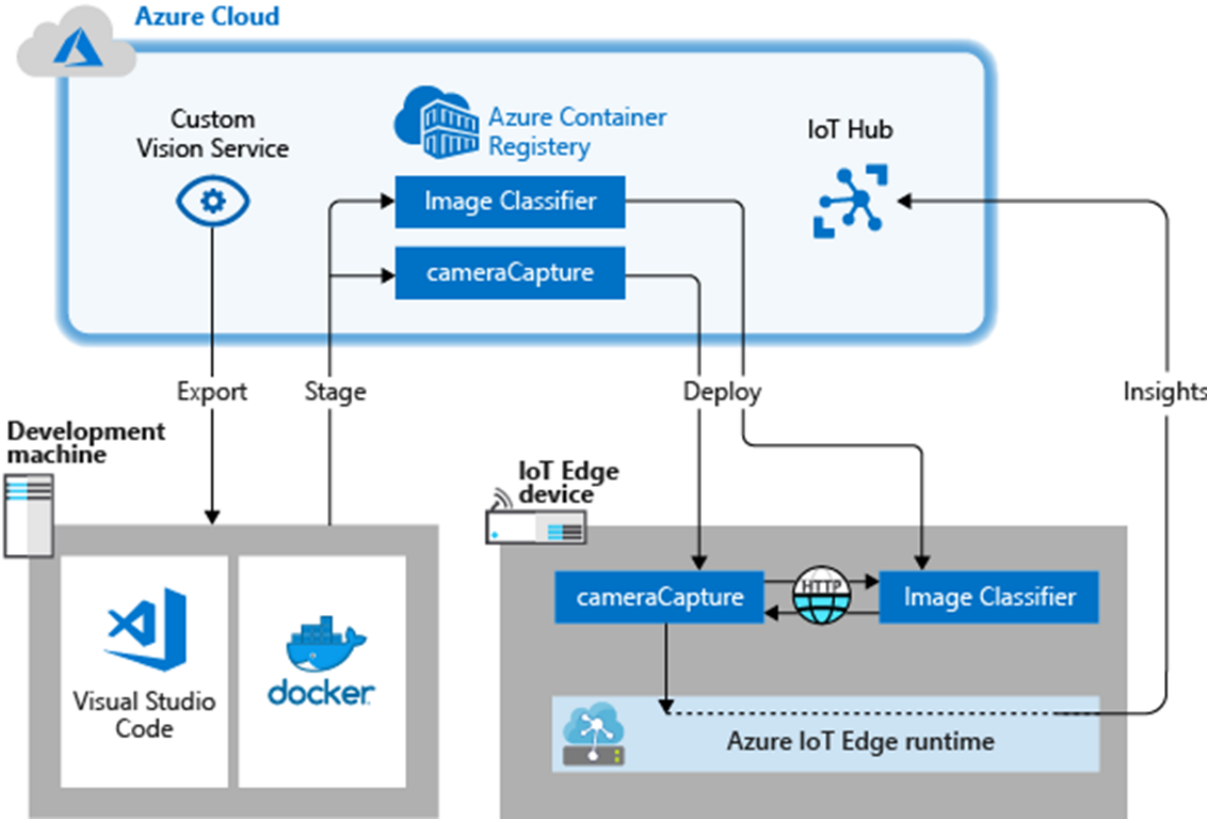
# Как развертывается Azure IoT Edge?



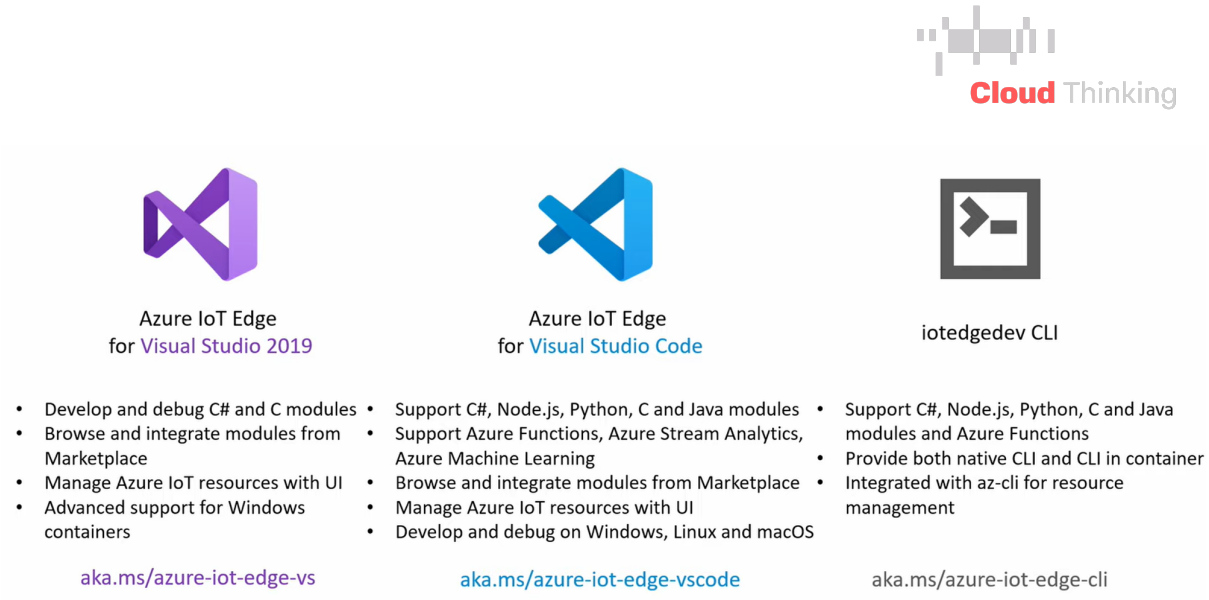

Как мы уже упомянули выше, IoT Edge работает по контейнерной технологии — можно взять контейнеры непосредственно из Microsoft или разработать собственный контейнер с аналитикой. Далее мы отправляем эти контейнеры в облачное хранилище Azure Container Registry — оно удобно тем, что хранит все последние копии контейнеров с их версионностью, поэтому в любой момент из облака можно эти контейнеры направить на любой сегмент IoT-решения.

Для развертывания контейнеров нужно создать файл Deployment Manifest. Он отправляется на Edge-устройство. Файл указывает, какие контейнеры должны быть установлены, какая у них должна быть версионность и по каким правилам все должно идти. То есть после того, как мы установили Deployment Manifest, разворачиваем контейнеры и запускаем нужные нам нагрузки.

С точки зрения разработки можно взять готовые модули от Microsoft — например, по машинному обучению или ИИ. Доступно большое количество модулей — от детектора аномалий до распознавания образов и речи. Их можно устанавливать на Edge.




Как мы говорили выше, есть возможность (и это основная функция) установить свои контейнеры. Для этого есть Azure IoT SDK, который позволяет разрабатывать собственные модули. Он поддерживает все языки, начиная от C, Java, NodeJS, и контейнерные технологии.




Инструменты, которыми можно воспользоваться для разработки модулей:

- **Visual Studio 2019**

Если вы разрабатываете в среде Windows и используете C, C#, C++, вы наверняка знакомы с Visual Studio. Для разработки модулей можно использовать Visual Studio 2019.

- **Visual Studio Code**

Бесплатный мультиязычный инструмент с поддержкой всех современных языков программирования.

- **iotedgedev CLI**

Позволяет разрабатывать модули с использованием командной строки.

Какие преимущества вы получаете, когда выстраиваете CI/CD в проектах с использованием IoT Edge и инструментов разработки:

- **Унификация процесса разработки**

Представим, что вы берете ноутбук, собираете команду и начинаете разрабатывать проект. Допустим, кто-то из команды работает над одной частью проекта, кто-то над другой, и все специалисты используют привычные им инструменты. Наша основная задача будет заключаться в деплое и дебаггинге. Огромное количество времени будет уходить на проверку кода и его исправление.

**Цель концепции CI/CD** — сэкономить время разработчикам, чтобы оно тратилось на программирование и релизы, а не на дебаг кода.

# Преимущества:

- Унификация процесса разработки для команды, работающей над различными сегментами кода
- Использование одних pipelines для тестирования облачной инфраструктуры и устройств
- Быстрая и эффективная сборка, тестирование и публикация приложений на базе контейнеров
- Более простое тестирование и развертывание IoT Edge-устройств, а также масштабирование решения
- Ведение учета сборок и релизов
- Переход решения от пилота в реализацию за короткий срок

Инструменты CI/CD позволяют унифицировать процесс разработки. Разнородные команды работают в одном ритме над своими «кусочками» кода, все это собирается в единую структуру, они используют одинаковые пайплайны (как с точки зрения облачных, так и с точки зрения физических устройств). Это позволяет быстрее переходить к тестированию и запуску решений в продакшен. CI/CD экономит время и позволяет рационально его использовать.

**Какие инструменты выбрать для встраивания CI/CD-процессов при работе с Azure IoT Edge?**

Azure IoT Edge tasks in
Azure Pipelines

aka.ms/azure-iot-edge-devops

Azure IoT Edge
Jenkins Plugin

aka.ms/azure-iot-edge-jenkins

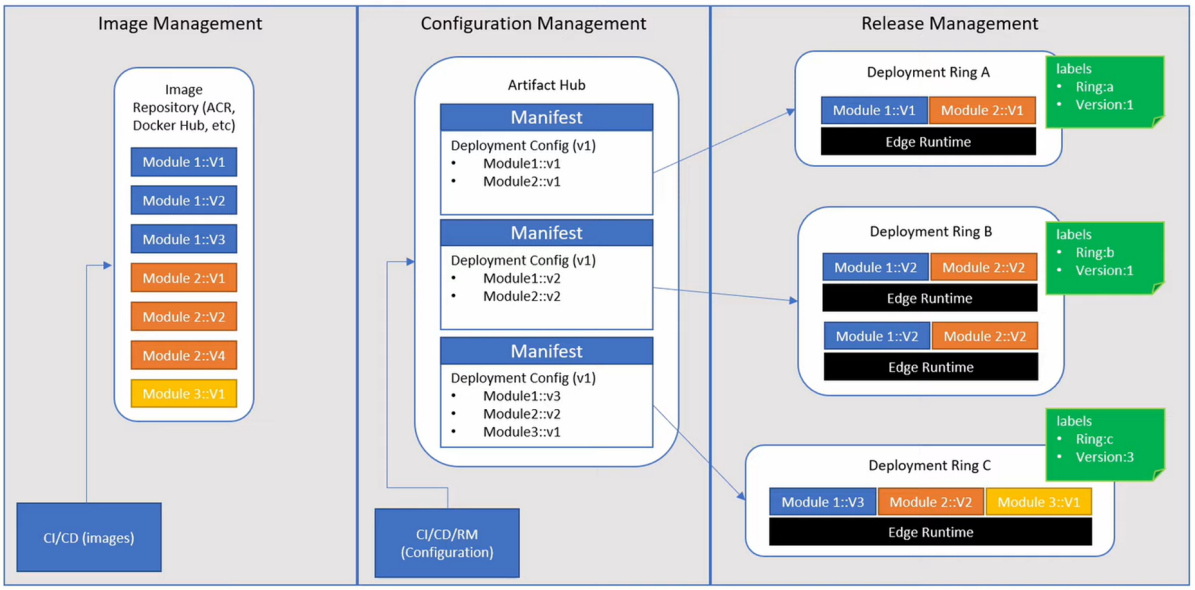
iotedgedev CLI with
Custom CI/CD system
(Travis CI, Circle CI, etc.)

aka.ms/azure-iot-edge-cli

- Тем, кто уже знаком с Azure, мы рекомендуем использовать Azure Pipelines. Это удобный инструмент, который позволяет легко настроить работу по CI/D.
- Специалисты из мира open source могут использовать Jenkins Plugin — его можно скачать и тоже работать в вашей удобной среде с Jenkins и использовать этот плагин для организации CI/CD.
- Если вы предпочитаете такие инструменты, как Travis CI, Circle CI и другие, — их тоже можно использовать в Azure IoT Edge.

# Как выстроить процесс пайплайнов?

На что обратить внимание, когда вы закончили проект и пришла пора запускать его в продуктив?




- **Image Management**

Количество разрабатываемых модулей может исчисляться десятками. У каждого модуля есть своя версионность. Когда вы отправляете модуль в Azure Container Registry или Docker Hub, важно обращать внимание на версию модуля. И наш инструмент эту задачу решает.

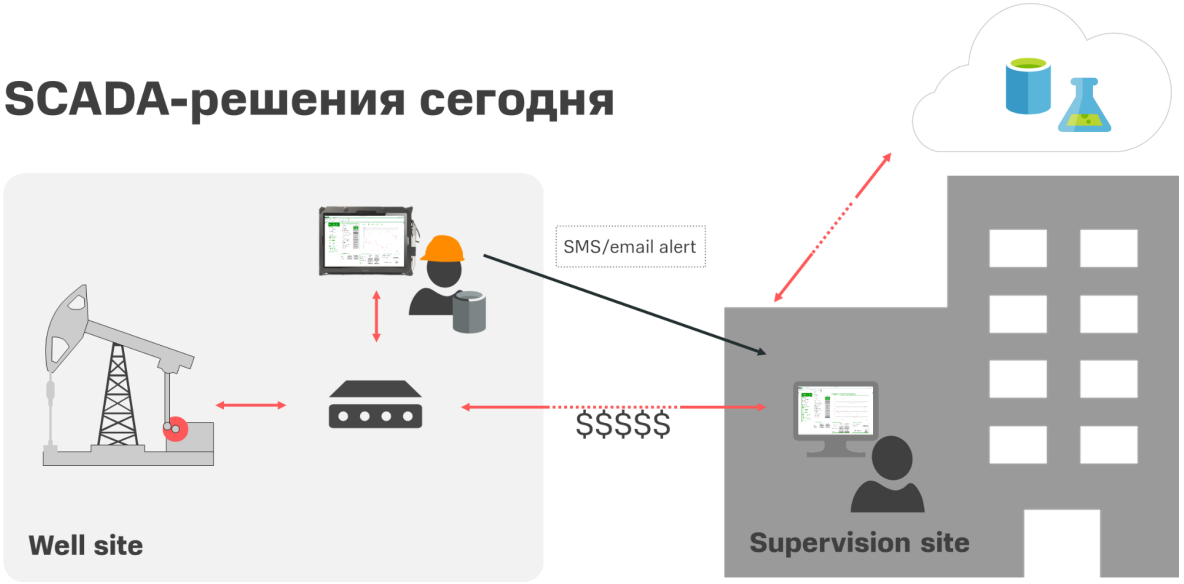
- **Configuration Management**

Для развертывания контейнера на Azure IoT Edge, как мы говорили ранее, есть Deployment Manifest, где мы указываем, какие модули, в какой последовательности должны быть развернуты. Важно следить за тем, чтобы соблюдались параметры и правила, по которым манифесты конфигурируются. За это отвечает Configuration Management.

- **Release Management**

Когда вы определили, какие модули и какие версии используются, эти модули нужно сопоставить с физическими устройствами, на которых они будут развернуты. У вас может быть широкая сеть и множество типов устройств. На разных устройствах могут работать разные версии. Release Management предусмотрен в этих инструментах и позволяет сопоставить физические устройства с модулями, которые вы разработали.

Чтобы получить практические знания по организации CI/CD на Azure IoT Edge, рекомендуем пройти курс на Microsoft Learn. Там вы познакомитесь с сервисами Azure Container Registry, Azure IoT Hub и поработаете в Azure Pipeline: соберете архитектуру, пройдете интеграционные и Smoke-тесты.




Разберем практический пример использования материалов, которые мы рассмотрели в сегодняшнем уроке.

Есть буровые станции. Обычно они находятся на удаленных площадках, куда нужно ехать на машине или лететь на вертолете. Данные со станций собираются в локальные системы. Далее процесс устроен следующим образом.

Когда что-то происходит, отправляется SMS или e-mail в центр и к управляющему, который следит за этими станциями. Исходя из того, что это за сообщение, он принимает решение: например, направить ли туда инженера, в какое время это сделать и так далее. Этот процесс достаточно трудоемкий.

**Что сделала компания?**
Она собрала данные с сотен своих станций (а именно — с насосов), эти данные прогнали через собственный алгоритм и получили предсказательную модель. Когда приходит определенный сигнал или превышается порог работы определенных элементов насоса, благодаря модели они уже знают, что может произойти: или это серьезная поломка и на станцию нужно направить специалиста, или это небольшая поломка (а может быть, и не поломка вовсе) и это подождет регулярного технического обслуживания.

Реализовать эту модель позволила Azure IoT Edge. Что сделали:

- после сбора всех данных их использовали для обучения и тренировки модели;
- модель поместили в контейнер;
- контейнер развернули на сайте, где находится насос.

Как это работает:

- насос передает данные по своему протоколу, поэтому для преобразования поставили контейнер с Modbus;
- далее эти данные уходят в модель;
- модель непосредственно на сайте понимает, в каком состоянии работает насос, какие у него нагрузки;
- при критической нагрузке высылается уведомление.