ЛЕКЦІЯ № 10
з навчальної дисципліни

Еталонна архітектура Microsoft Azure IoT

Питання лекції

1. Загальна структура
2. Основні принципи та концепції архітектури Azure IoT
3. Деталі підсистеми архітектури

1. Загальна структура

Підключені датчики, пристрої та інтелектуальні операції можуть перетворити підприємства та забезпечити нові можливості для зростання з послугами Microsoft Azure Internet of Things (IoT).

Метою лекції є надання огляду рекомендованої архітектури та реалізації вибір технологій для створення рішень Azure IoT. Ця архітектура описує термінологію, технологічні принципи, загальні середовища конфігурації, а також композиції служб Azure IoT, фізичних пристроїв та інтелектуального краю.

Застосування IoT може бути описано як речі (або пристрої), які передають дані або події, які використовуються для формування статистичних даних, які використовуються для створення дій, які допомагають покращити бізнес або процес. Прикладом є двигун (річ), що посилає тиск і дані про температуру, які використовуються для оцінки того, чи працює двигун, як і очікувалося (інсайт), який використовується проактивно визначати пріоритети графіка технічного обслуговування двигуна (дії). Сьогодні ми зосередимся на тому, як побудувати Рішення IoT, однак, важливо усвідомлювати кінцеву мету архітектури: вживати заходів щодо розуміння бізнесу, яке ми знаходимо через збір даних з активів.

Лекція містить три питання:

1) огляд - містить загальну рекомендовану архітектуру IoT рішення (розділені на підсистеми), короткий вступ до підсистем застосування IoT,

технологія за замовчуванням рекомендації для кожної підсистеми, а також обговорення міжгалузевих проблем для програм IoT,

2) основні поняття та принципи - в цьому описуються концепції та принципи, що є центральними для побудови масштабованих додатків IoT

3) подробиці підсистеми - для кожної підсистеми підрозділ присвячений опису відповідальності підсистеми і технологічні альтернативи для реалізації.

1.1 Огляд архітектури

Архітектура, яку ми рекомендуємо для IoT-рішень, - це хмара, мікросервіси і безсерверне базування. Рішення IoT - підсистеми повинні бути побудовані як дискретні послуги, які незалежно розгортаються і здатні масштабуватися незалежно.

Ці атрибути забезпечують більший масштаб, більше гнучкості в оновленні окремих підсистем і забезпечують гнучкість вибрати відповідну технологію на основі кожної підсистеми. Дуже важливо мати здатність контролювати елементи підсистеми, а також застосування IoT в цілому. Рекомендується підсистемам зв'язуватися за допомогою REST/HTTPS JSON (так, як це зрозуміло людині), хоча двійкові протоколи повинні використовуватися для високопродуктивних потреб. Архітектура також підтримує гібридну хмару і стратегію обчислення країв; певна обробка даних, як очікується, відбудеться на місці.

Ми рекомендуємо використовувати оркестратора (наприклад, служби Azure Kubernetes - AKS або Service Fabric) для масштабування окремих елементів підсистеми по горизонталі або служби PaaS (наприклад, служби Azure), які пропонують вбудовані можливості горизонтального масштабу.

Базовий додаток IoT складається з наступних підсистем:

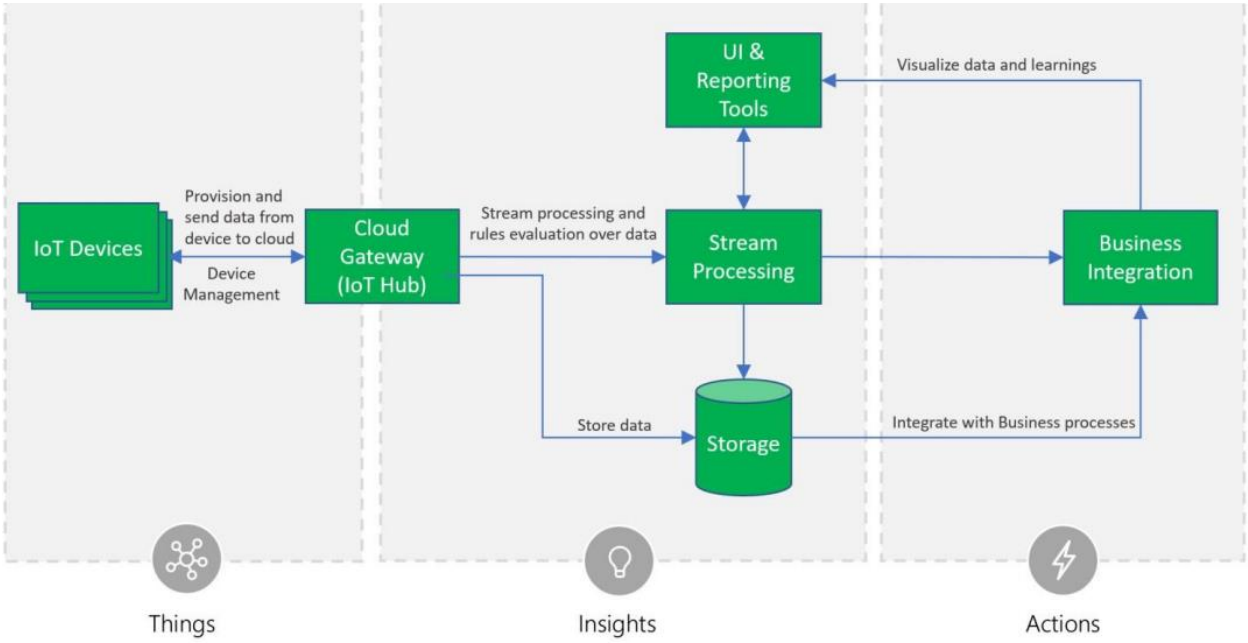
1) пристрої (і/або на пограничних шлюзах) повинні мати можливість безпечно реєструватися в хмарі, а також можливості підключення для надсилання й отримання даних за допомогою хмари,

2) служба хмарного шлюзу або IoT хаб, має безпечно прийняти ці дані та забезпечити можливості керування пристроями,

3) потокові процесори, які споживають ці дані, інтегруються з бізнес-процесами і розміщують дані в сховище;

4) інтерфейс користувача для візуалізації даних телеметрії та полегшення керування пристроєм.

Далі, ці підсистеми коротко описані рекомендаційні технології.



Cloud Gateway забезпечує хмарний хаб для безпечного підключення, телеметрії та прийому подій та керування пристроями (включаючи командний і контрольний) можливості. Ми рекомендуємо використовувати службу Hub для Azure як шлюз хмари.

Концентратор IoT забезпечує вбудовану безпеку підключення, телеметрію та прийом подій, а також двостороннє спілкування з пристроями, включаючи керування пристроями з можливістю керування командами та управління. Крім того, Hub IoT пропонує об'єкт сховище, яке можна використовувати для зберігання метаданих пристрою.

Для реєстрації та підключення великих наборів пристроїв ми рекомендуємо використовувати службу надання послуг концентратора пристроїв Azure IoT (DPS). DPS дозволяє призначати та реєструвати пристрої на конкретних кінцевих точках концентратора Azure IoT у масштабі. Ми рекомендуємо використання SDK для концентратора Azure IoT, щоб забезпечити можливість безпечного підключення пристроїв і відправлення телеметричних даних у хмару.

Обробка потоків обробляє великі потоки записів даних і оцінює правила для цих потоків. Для обробки потоку рекомендується використовувати Analytics Azure Stream Analytics для додатків IoT, які вимагають складної обробки правил у масштабі. Для простої обробки правил ми рекомендуємо маршрути концентраторів Azure IoT, що використовуються з функціями Azure.

Інтеграція бізнес-процесів полегшує виконання дій на основі даних, отриманих від даних телеметрії пристрою обробки потоку. Інтеграція може включати зберігання інформаційних повідомлень, нагадувань, надсилання електронної пошти або SMS, інтеграція з CRM тощо. Для бізнес-процесу рекомендується використовувати функції Azure і логічні програми інтеграції.

Зберігання можна розділити на теплу траєкторію (дані, які повинні бути доступними для звітності та візуалізації негайно від пристроїв) і холодну

траєкторію (дані, що зберігаються довше терміну і використовуються для пакетної обробки). Рекомендується використовувати Azure Cosmos DB для зберігання теплих траєкторій і сховища Azure Blob для холодного зберігання. Для додатків з часовими рядами певні звітні потреби ми рекомендуємо використовувати Azure Time Series Insights.

Інтерфейс користувача для програми IoT може бути доставлений на широкий спектр типів пристроїв, у власних програмах, і браузерів. Потреба в IoT системах для користувальницького інтерфейсу та звітності різноманітна і ми рекомендуємо використовувати Power BI, TSI провідник, власні програми та спеціальні програми веб-інтерфейсу.

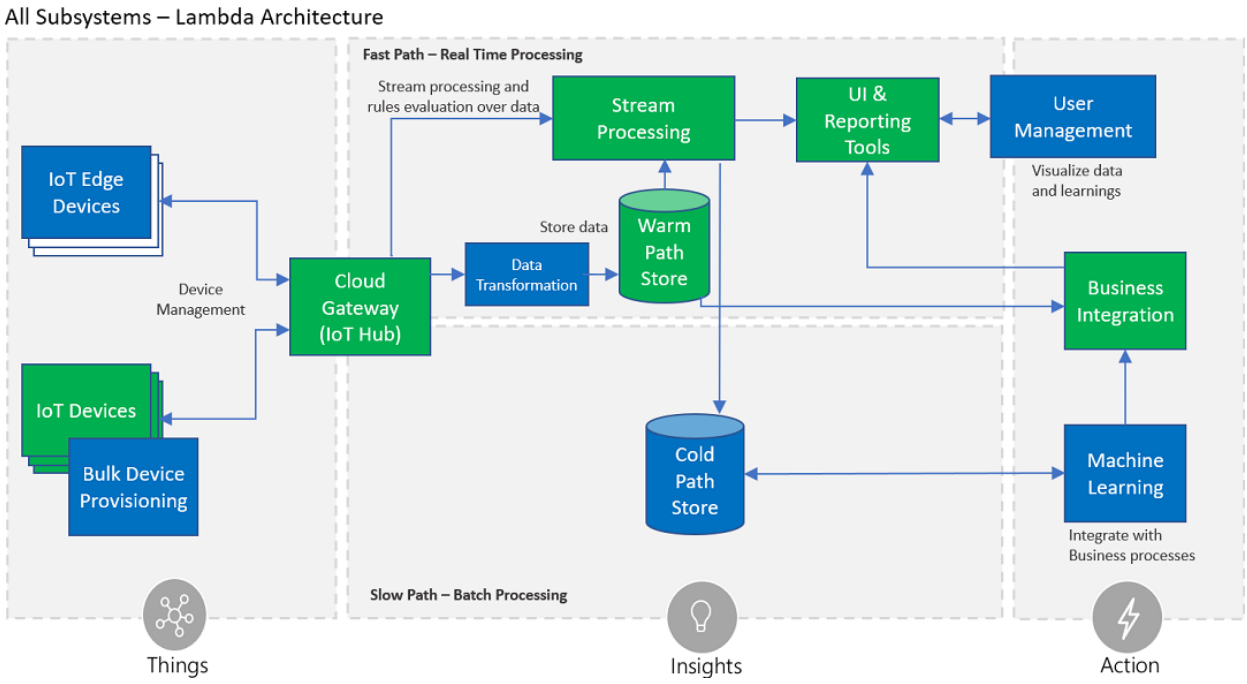
Крім основних підсистем, багато додатків IoT включатимуть підсистеми для:

5) інтелектуальних крайових пристроїв, які дозволяють агрегувати або трансформувати дані телеметрії та обробку приміщень,

6) обробку даних хмарної телеметрії, що дозволяє реструктуризувати, комбінувати або трансформувати дані телеметрії, що надсилаються з пристроїв,

7) машинне навчання, яке дозволяє виконувати прогностичні алгоритми над даними історичної телеметрії, дозволяючи сценарії

8) управління користувачами, що дозволяє розбивати функції між різними ролями та користувачів.



Інтелектуальні пристрої Edge виконують активну роль у управлінні доступом та інформаційним потоком. Вони можуть допомогти в забезпеченні фільтрації даних, дозуванні та агрегації, буферизації даних, перекладі протоколів, обробці правил подій. Рекомендується Azure IoT Edge

використовувати для цих потреб на приміщеннях. Azure IoT Edge також пропонує розширювану модель для включення користувацьких функцій за допомогою модулів Edge.

Перетворення даних включає в себе маніпулювання або агрегацію потоку телеметрії або до, або після її отримання службою хмарного шлюзу (Hub IoT). Маніпуляція може включати в себе перетворення протоколу (наприклад, перетворення двійкової системи потокових даних до JSON), об'єднання точок даних і багато іншого. Для перекладу телеметричних даних до його отримання за допомогою IoT Hub ми рекомендуємо використовувати шлюз протоколу. Для перекладу даних після його надходження IoT Концентратор рекомендується використовувати для інтеграції IoT Hub з функціями Azure.

Підсистема машинного навчання (ML) дозволяє системам вчитися на основі даних і досвіду і діяти, не будучи явно запрограмованою. Такі сценарії, як прогнозне обслуговування, включені через ML. Рекомендується використовувати службу Azure Машинне Навчання для потреб машинного навчання.

Підсистема керування користувачами дозволяє специфікувати різні можливості для користувачів і груп для виконання дій на пристроях (наприклад, команд і керування, таких як оновлення мікропрограми для пристрою) і можливостей для користувачів у програмах.

Вона додатково обговорюється як частина наскрізних вимог безпеки.

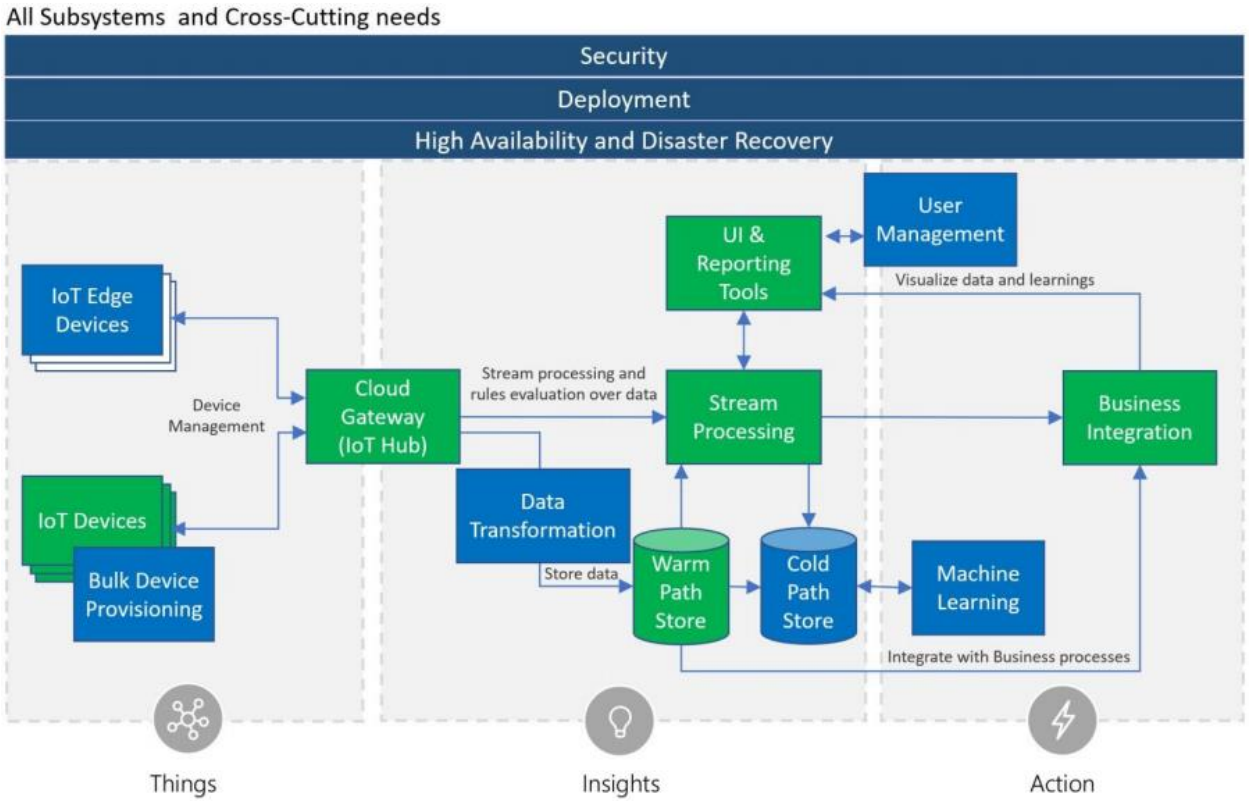
Існує безліч різнопланових потреб для додатків IoT, які мають вирішальне значення для успіху, включаючи:

8) вимоги безпеки

вимоги; включаючи керування та аудит користувачів, підключення пристроїв, телеметрію в дорозі та безпеку,

9) ведення журналу та моніторинг для додатків хмари IoT є критичним для визначення здоров'я та для усунення несправностей і для окремих підсистем, і для програми в цілому, і

10) висока доступність і аварійне відновлення, яка є використовуються для швидкого відновлення після системних невдач.



Безпека є критичним фактором у кожній з підсистем. Захист рішень IoT вимагає надійного забезпечення пристроїв, безпечне підключення між пристроями, крайовими пристроями і хмарою, безпечний доступ до серверних рішень, і захистити захист даних у хмарі під час обробки та зберігання (шифрування в стані спокою). Як було зазначено раніше, рекомендується використовувати Azure IoT Hub, який пропонує повністю керовану послугу, яка забезпечує надійну та безпечну двонаправлену функцію зв'язок між пристроями IoT і Azure, такими як Azure Machine Learning і Azure Stream Analytics використання облікових даних захисту та пристрою контролю доступу. Для технологій зберігання рекомендується використовувати Azure Cosmos DB для зберігання теплих шляхів і сховища Azure Blob для холодного зберігання, які підтримують спокійне шифрування. Для управління доступом користувача, такі як аутентифікація облікових даних користувача, авторизація можливостей користувальницького інтерфейсу користувача, звітність і управління інструментами, якими користуються користувачі, управління доступом до аудиторської діяльності й аудиту додатків, ми рекомендуємо Azure Active Directory. Azure Active Directory підтримує широко використовуваний протокол авторизації OAuth2, рівень аутентифікації OpenID Connect і надає записи журналу аудиту системної діяльності.

Журнали і моніторинг для програми IoT є критичним для визначення часу безвідмовності системи та у вилученні збоїв. Рекомендується використовувати операції Azure Management Management Suite (OMS), додаток Map і App Insights для операцій моніторингу, реєстрації та усунення несправностей.

Висока доступність та відновлення після аварії зосереджується на забезпеченні завжди доступної системи IoT, включаючи невдачі внаслідок катастроф. Технологія, що використовується в підсистемах IoT, має різні показники відмовостійкості і міжрегіональну підтримку. Для програм IoT це може призвести до необхідності розміщення дублікатів сервісів і дублювання програми.

2. Основні принципи та концепції архітектури Azure IoT

Принципи

Еталонна архітектура дозволяє збирати безпечні, комплексні рішення, що підтримують екстремальні масштаби, але дозволяють гнучкість щодо сценаріїв розв'язання. Це мотивує наступні керівні принципи в різних областях архітектури.

Гетерогенність. Ця еталонна архітектура повинна враховувати різні сценарії, середовища, пристрої, процеси обробки та стандарти. Вона повинна бути здатною обробляти велику неоднорідність апаратного та програмного забезпечення.

Безпека. Тому що рішення IoT являють собою потужний зв'язок між цифровим і фізичним світами, будівництво безпечних елементів є необхідною основою для побудови безпечних систем. Ця еталонна модель передбачає безпеку і заходи щодо конфіденційності в усіх областях, включаючи ідентифікацію пристроїв і користувачів, аутентифікацію та авторизацію, захист даних для даних у спокої та даних у русі, а також стратегії атестації даних.

Гіпер-масштабні розгортання. Запропонована архітектура підтримує мільйони підключених пристроїв. Це дозволить довести концепції і пілотні проекти, які починаються з невеликої кількості пристроїв, які будуть масштабовані до гіпермасштабних розмірів.

Гнучкість. Гетерогенні потреби ринку IoT вимагають відкритого складу послуг і компонентів. Еталонна архітектура побудована на принципі композитності, що дозволяє створювати рішення IoT шляхом поєднання ряду будівельних блоків і дозволяє використовувати різні технології першої або третьої сторони окремі концептуальні компоненти. Ряд точок розширення дозволяє інтегруватися з існуючими системами та програми. Високомасштабна, керована подіями архітектура з посередницькою комунікацією є основою вільно пов'язаної склад послуг і модулів обробки.

Поняття даних

Розуміння концепцій даних є важливим першим кроком для побудови, аналізу та контролю даних, орієнтованих на пристрої систем. Роль моделей пристроїв і даних, потоків даних і кодування детально описана в наступних питаннях.

Моделі пристроїв і даних

Ключовими для реалізації є інформаційні моделі, що описують пристрої, їх атрибути та асоційовану схему даних

рішення бізнес-логіки та обробки.

Існує багато різних моделей варіантів для моделювання пристроїв у різних галузях промисловості, а також до цієї архітектури займає нейтральну позицію для підтримки цих поточних спроб моделювання та схематизації.

Наприклад, у випадку промислового сценарію IoT, семантика даних і структура можуть базуватися на OPC UA. Інші реалізації, такі як домашня автоматизація та автомобільні додатки, можуть використовувати повністю різні галузеві моделі та стандарти.

Архітектура приймає фундаментальну абстракцію потоків даних, де не потрібні моделі пристроїв і даних, маршрут або зберігання інформації в основних компонентах платформи. На етапі рішення будуть збережені структуровані дані моделями даних і схемою, коли вона виробляється або споживається компонентами. Розробники мають можливість вибору використання схем для розробки пристрою-клієнта, аналітики бекенда або певної логіки обробки, як цього вимагає рішення.

Записи та потоки даних

Рішення IoT розроблені з урахуванням фундаментального аспекту: пристрої, що періодично передають записи даних, які представляються, аналізуються і зберігаються як множинні і безперервні потоки даних. Повідомлення, події, телеметрія, оповіщення та прийом є термінами, які зазвичай використовуються при описі потоків даних IoT.

Записи даних зазвичай виписуються за часом, сортуються за часом і асоціюються принаймні з одним джерелом. Наприклад, запис телеметрії може містити час вимірювання і час, коли дані приймаються, і може бути пов'язаний з ім'ям пристрою, де проводиться вимірювання, до шлюзу, де збирається телеметрія, концентратора, де телеметрія поглинається і т.д.

Поглинання - це процес завантаження записів даних у сховище через шлюз, наприклад, Azure IoT Edge і Azure IoT Hub. Записи даних можна приймати одночасно або масово. Зміст потоків може бути даними реального часу або минулим трафіком, що повторюється.

Повідомлення та події є взаємозамінними термінами, які використовуються при посиланні на записи даних, що генеруються підключеними пристроями. Термін телеметрія використовується спеціально для повідомлень, що несуть дані, повідомлені датчиками пристрою, наприклад, поточна температура, що надходить з датчика температури на пристрій. Записи телеметрії можуть містити одну або кілька точок даних.

Наприклад, пристрій з одним датчиком вологості і одним датчиком температури може посилати вологість і температуру вимірювання в одному повідомленні або в окремих повідомленнях.

Пристрої можуть мати декілька встановлених датчиків і можуть посилати записи з вимірами, повідомленими всіма датчиками або тільки значення, які змінилися з часу останньої телеметрії, що були надіслані, наприклад, для зменшення кількості переданих даних.

Значення точки даних у записі телеметрії стає останнім відомим станом. При відправленні тільки записів різниці, пристрої іноді можуть також надіслати повний знімок всіх значень сенсорів (так званих ключових кадрів), для узгодженості та цілей синхронізації.

Записи телеметрії зазвичай аналізуються локально або в хмарі, у відповідності до набору правил.

Формат записів даних

Записи даних не мають встановленого формату. Припущення для кожного потоку даних полягає в тому, що всі записи використовують сумісність структури і семантики. Формат, обраний виробниками пристроїв і рішеннями IoT, залежить від декількох факторів, наприклад програмне забезпечення, що працює на пристроях, ємність процесорів, пропускна здатність, безпека і т.д. Рекомендується в рішеннях IoT прийняти формат JSON, зважаючи на його читаність і відносно низький необхідний простір, однак існує декілька бінарних форматів, наприклад Avro, що дозволяє підвищити продуктивність і знизити витрати.

Для спрощення десеріалізації слід допускати нерозривні зміни та сегрегацію потоків за версією. Краща практика полягає в тому, щоб рішення IoT включали метадані в кожен запис, напр. використовуючи властивості повідомлення, вказуючи формат і версії. З наявністю моделі версій, розробники рішень можуть належним чином вирішити потенційні конфлікти полів записів в термінах семантики або типу; напр. якщо специфічне програмне забезпечення пристрою змінюється і після цього пристрій надсилає записи даних то різний формат версій дозволить розробнику розв'язати проблему між потоками даних.

Служби платформи Azure IoT є агностичним корисним навантаженням і не вимагають присутності певного поля в повідомленні.

Повнота повідомлень і сумісність є обов'язком розробників пристроїв і рішень.

Взаємодія пристрою

Еталонна модель приймає принципи комунікації для пристроїв, які потенційно розгортаються в ненадійному фізичному просторі. Застосовуються наступні принципи:

• Пристрої не приймають небажані мережні з'єднання. Всі з'єднання та маршрути встановлені при налаштуванні.

• Пристрої зазвичай підключаються або встановлюють маршрути до відомих шлюзів служби, з якими вони звертаються. У випадку, якщо їм необхідно подавати інформацію або отримувати команди з множини послуг, пристрої розглядаються з шлюзом, який опікується інформацією про маршрутизацію вниз і гарантує, що команди тільки дозволені від уповноважених сторін перед тим, як направити їх на пристрій.

• Шлях зв'язку між пристроєм і службою або пристроєм і шлюзом забезпечується на транспортному рівні та рівні протоколу додатків.

• Авторизація та аутентифікація на рівні системи повинні ґрунтуватися на ідентифікаціях кожного пристрою та облікових даних доступу. Дозволи повинні бути негайно відкликаними у випадку зловживання пристроєм.

• Двоспрямований зв'язок для пристроїв, які періодично підключені через проблеми з електроживленням або підключенням можуть бути полегшені за допомогою зберігання команд і сповіщень до пристроїв, поки вони не підключатимуться до цих пристроїв вгору.

• Дані корисного навантаження програми можуть бути окремо захищені для захищеного транзиту через шлюзи до конкретного обслуговування.

Загальний шаблон для інформування пристроїв із обмеженим енергоспоживанням про важливі команди під час відключення є за допомогою використання каналу зв'язку поза смугою, такого як протоколи стільникових мереж і послуги. Так наприклад, SMS-повідомлення може бути використано для "пробудження" пристрою і доручення йому встановити вихідну мережу підключення до свого «домашнього» шлюзу. Після підключення пристрій отримає команди та повідомлення.

Комунікаційні протоколи

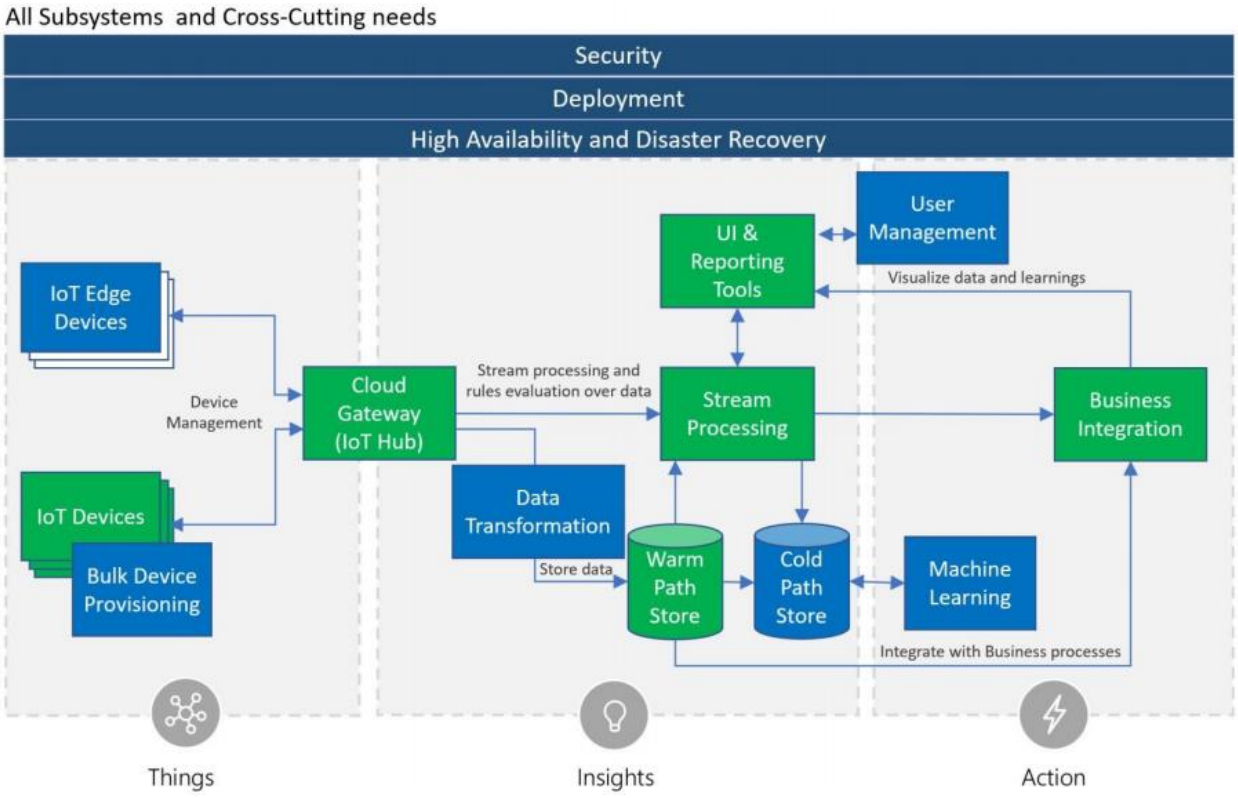
Сьогодні для сценаріїв пристроїв доступно багато протоколів зв'язку, кількість яких зростає. Вибір з тих, що використовуються в системах гіпермасштабу, щоб забезпечити безпечні операції, забезпечуючи при цьому можливості та гарантії, обіцяні обраними протоколами, вимагають значної експертизи у побудові розподілених систем. Проте існує величезна кількість існуючих пристроїв, для яких вже було зроблено вибір протоколу, і ці пристрої повинні бути інтегровані в рішення.

У цій еталонній моделі обговорюються переважні варіанти вибору протоколу зв'язку, пояснюються потенційні компроміси з ними виборів, а також явно дозволяє розширюваність, адаптацію та локальну обробку протоколів на місцевому шлюзі (IoT Edge), шлюз протоколу, заснований на хмарі, або під час обробки потоку.

Зверніть увагу, що протокол зв'язку визначає, як переміщуються корисні навантаження та передає метадані про корисне навантаження які можуть бути використані для диспетчеризації / маршрутизації і декодування, але зазвичай не визначають форму або формат корисного навантаження. Так, наприклад, зв'язок може бути включений за протоколом AMQP, але кодування даних може бути Apache Avro, або JSON або рідне кодування AMQP.

3. Деталі підсистеми архітектури

У цьому підрозділі кожна архітектурна підсистема детально обговорюється, включаючи призначення підсистеми, технології варіанти реалізації та рекомендовані варіанти впровадження.



3.1 Пристрої, підключення пристроїв, польовий шлюз (Edge Devices), Cloud Gateway

Пристрої можуть бути підключені безпосередньо або опосередковано через шлюз поля (IoT edge device). І пристрої, і польові шлюзи можуть реалізовувати функції розвідки краю. Це дає можливість агрегування та скорочення даних необробленої телеметрії раніше транспортування до бекенда та можливості прийняття рішень на місцевому рівні.

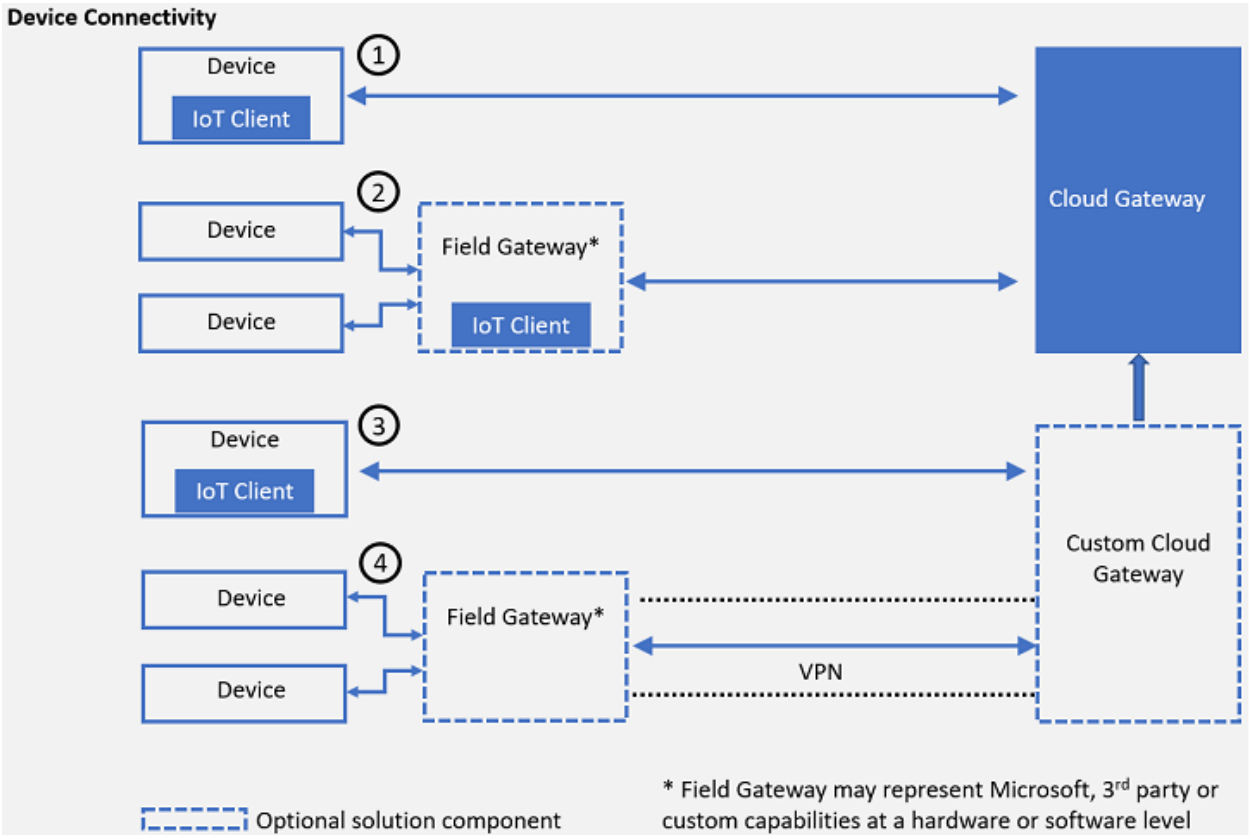
Нижче наведено концептуальне представлення різних варіантів підключення пристроїв до рішень IoT. Позначена цифра відповідає чотирьом основним схемам підключення, визначеним таким чином:

1. Пряме підключення пристрою до хмарного шлюзу: для пристроїв, що підтримують IP, які можуть встановлювати безпечні з'єднання за допомогою Інтернет.

2. З'єднання через польовий шлюз (IoT Edge Device): для пристроїв, що використовують специфічні для промисловості стандарти (наприклад, протокол обмеженого застосування [CoAP8] або OPC UA), технології комунікації малого діапазону (такі як Bluetooth або ZigBee), а також для пристроїв, обмежених ресурсами, які не можуть розміщувати стек TLS/SSL, або пристрої, які не підключаються до Інтернету. Цей варіант також корисний, коли виконується агрегація потоків і даних на польовому шлюзі перед переходом до хмари.

3. З'єднання через користувальницький хмарний шлюз: для пристроїв, які вимагають перекладу протоколу або певної форми користувача обробку до досягнення кінцевої точки зв'язку шлюзу хмари.

4. Зв'язок через польовий шлюз та користувальницький шлюз хмари: Подібно до попереднього шаблону, сценарії підключення через польовий шлюз можуть вимагати адаптації протоколів або налаштувань на стороні хмари і тому можуть вибирати для підключення до користувацького шлюзу, запущеного у хмарі. Деякі сценарії вимагають інтеграції поля та хмари шлюзи з використанням ізольованих мережевих тунелів, використовуючи технологію VPN або використовуючи службу ретрансляції на рівні програми.



Концептуальне представлення підключення пристроїв

Пряма комунікація між пристроями і пристроєм дозволяє здійснювати керування локальною мережею та інформаційний потік, або спільні операції, коли кілька пристроїв виконують скоординовані дії. Чисто місцеві взаємодії виходять за рамки цієї архітектури і охоплені такими галузевими стандартами, як AllJoyn, UPnP / DLNA та інші.

Важливо розуміти термінологію та ключові компоненти, які використовуються для опису підключення пристроїв до Azure IoT.

Пристрої

Гетерогенна підтримка пристрою. Мета полягає в тому, щоб забезпечити безпечну, ефективну і надійну комунікацію між майже будь-якими типами пристрою та хмарним шлюзом. Це можна зробити як безпосередньо, так і через шлюзи.

Цільові пристрої. Орієнтовані на нього пристрої - це лінійка бізнесу, від простих температурних датчиків до складних заводських виробничі лінії з сотнями компонентів з датчиками всередині них.

Мета цих пристроїв буде диктувати їх технічний дизайн, а також кількість ресурсів, необхідних для їх виробничої та планованої експлуатації. Поєднання цих двох факторів визначатиме наявні можливості зберігання, обчислення та безпеки. Еталонна архітектура в цілому нейтральна по відношенню до середовища виконання, платформи, операційної системи і виконуваної функції пристрою.

Польовий Шлюз (Edge Devices)

Польовий шлюз, або крайовий пристрій, є спеціалізованим пристроєм або програмним забезпеченням загального призначення, що діє як комунікаційний агент і, потенційно, як локальний пристрій управління системою і пристроєм обробки даних концентратора. Польовий шлюз може виконувати локальну обробку і управління функціями для пристроїв і може фільтрувати або агрегувати телеметрію пристрою і таким чином зменшують обсяг даних, що передаються до бекенда в хмарі.

Область Польового шлюзу поля включає в себе сам шлюз і всі приєднані до нього пристрої. Як випливає з назви, Польові шлюзи діють поза призначеними засобами обробки даних і зазвичай розташовуються між пристроями.

Польовий шлюз відрізняється від простого маршрутизатора трафіку тим, що шлюзи поля можуть мати активну роль в управлінні доступом вони є "розумними" пристроями. Польові шлюзи можуть допомогти у забезпеченні пристроїв, фільтрації даних, дозування і агрегації, буферизації даних, переклад протоколів і обробці правил подій. Рекомендується Azure IoT Edge використовувати для шлюзів поля в рішеннях IoT; Azure IoT Edge діє як розумний край і полегшує підключення, реалізація маршрутизації, перекладу протоколів, машинного навчання, штучного інтелекту, обробки потоків тощо.

Azure IoT Edge пропонує розширювану модель для розширення функціональності за допомогою модулів Edge.

Шлюз хмари (хмарний шлюз)

Шлюз хмари дає можливість віддаленого зв'язку з пристроями, або з крайовими пристроями, які потенційно перебувають у декількох різних сайтах. Шлюз хмари буде або доступним через загальнодоступний Інтернет, або накладання мережевої віртуалізації (VPN) або приватні мережні підключення до центрів обробки даних Azure, щоб ізолювати шлюз хмари та всі його прикріплені пристроїв або крайніх пристроїв з іншого мережевого трафіку.

Він, як правило, керує всіма аспектами зв'язку, включаючи управління з'єднанням на рівні транспортного протоколу, захист шляху зв'язку, аутентифікації пристрою та авторизації до системи. Шлюз дотримується збору та пропускної квоти, а також збирає дані, які використовуються для виставлення рахунків, діагностики та інших завдань моніторингу.

Для того, щоб підтримувати архітектури, керовані подіями, хмарний шлюз зазвичай пропонує модель комунікації з посередниками.

Телеметричні та інші повідомлення від пристроїв вводяться в хмару, а обмін повідомленнями здійснюється за допомогою шлюзу. Дані стійко буферизуються, що не тільки розв'язує відправник від приймача, але і дозволяє множину споживачів даних. Зазвичай трафік із сервісу сервера на пристрої (наприклад, сповіщення та команди) реалізовано за допомогою шаблону "Вхідні". Навіть коли пристрій перебуває в автономному режимі, повідомлення, надіслані до нього, будуть довго зберігатися в черзі (папка "Вхідні" для пристрою) і доставлятися після підключення пристрою.

Важливим є врахування можливого часу затримки подій, особливо для чутливих до часу команд, таких як "відкрити автомобіль або домашні двері" або "запустити автомобіль або машину". Шаблон папки "Вхідні" зберігатиме повідомлення в тривалий час (для заданої тривалості TTL), після чого термін дії повідомлень закінчується.

Брокерська комунікація через описані структури дозволяє розв'язувати край з компонентами хмари у залежності від часу виконання, швидкості обробки та поведінкових контрактів. Це також дає можливість композиції видавцям і споживачам, необхідним для побудови ефективних, масштабних, керованих подіями рішень.

Концентратори подій Azure. Концентратори подій Azure представляють собою високоякісну службу, призначену лише для прийому, для збору даних телеметрії з джерел паралельного доступу з дуже високою пропускною спроможністю. Концентратори подій також можуть використовуватися в сценаріях IoT, крім Hub IoT, для вторинні телеметричні потоки (тобто телеметричність без пристрою) або збір даних з інших системних джерел (таких як подачі погоди або соціальні потоки). Концентратори подій не пропонують ідентифікацію пристрою або можливості команд і керування, тому це може бути придатним тільки для додаткових потоків даних, які можуть корелювати з телеметрією пристрою на сервері, але ні як основний шлюз для підключення пристроїв. Концентратори подій Azure підтримують підтримку AMQP 1.0 з додатковими параметрами Підтримка WebSocket, а також протоколи HTTPS.

Підтримка додаткових протоколів поза AMQP, MQTT і HTTP може бути реалізована за допомогою протокольного шлюзу модель адаптації. CoAP є прикладом протоколу, який може використовувати цю модель.

Користувацький шлюз хмари

Користувальницький хмарний шлюз дозволяє адаптувати протокол і/або певну форму користувацької обробки, перш ніж дістатися до кінцевих точок зв'язку шлюзу. Це може включати відповідну реалізацію протоколу, необхідну для пристроїв (або поля шлюзів) під час пересилання повідомлень на хмарний шлюз для подальшої обробки і передачі команди і керувати повідомленнями від хмарного шлюзу до пристроїв. Крім того, спеціальна обробка, така як повідомлення перетворення або стиснення/декомпресії також можуть бути реалізовані як частина користувацького шлюзу. Однак це

необхідно ретельно оцінювати, оскільки, в цілому, корисно вносити дані до шлюзу хмари так швидко, як можливо, а потім виконувати перетворення на бекенд хмари, відокремлено від прийому.

Спеціальні шлюзи допомагають підключати різноманітні пристрої з користувацькими або власними вимогами і нормалізувати край трафіку на кінці хмари. Спеціальні користувальницькі шлюзи, що відповідають конкретним рішенням, зазвичай виступатимуть як пропускний пристрій і може реалізувати спеціальну аутентифікацію або покластися на можливості аутентифікації та авторизації шлюзу хмари.

Зауважте, що користувацькі шлюзи також можуть бути розгорнуті на краю. У деяких випадках може бути декілька шлюзів між пристроєм і шлюзом хмари.

Шлюз протоколу Azure IoT. Шлюз протоколу Azure IoT - це платформа з відкритим вихідним кодом для користувацьких шлюзів і адаптація протоколу. Шлюз протоколу Azure IoT сприяє високому масштабу, двосторонньому зв'язку між ними пристроїв і концентратора Azure IoT. Вона включає в себе адаптер протоколу для MQTT, який демонструє методи реалізації користувальницьких протоколів і дозволяє налаштувати поведінку протоколу MQTT, якщо потрібно. Шлюз протоколу також дозволяє здійснювати додаткову обробку, наприклад, спеціальну аутентифікацію, перетворення повідомлень, стиснення / розпакування, або шифрування / дешифрування.

Клієнт IoT

Хмара-комунікація з пристроями або крайовими пристроями повинна здійснюватися через захищені канали до кінцевих точок шлюзу хмари (або спеціальні шлюзи, розміщені в хмарі).

На додаток до захищеного каналу зв'язку, пристрій зазвичай повинен доставляти дані телеметрії до хмарного шлюзу і дозволяють приймати повідомлення і виконувати дії або відправляти їх до відповідних обробників клієнта. Як зазначено раніше, всі з'єднання пристроїв і шлюзів повинні бути встановлені тільки в вихідному режимі.

Існує *три ключові моделі для підключення клієнтів*, які використовуються в системах IoT:

• Пряме підключення з рівня пристрою / програмного забезпечення

• Зв'язок через агентів

• Використання компонентів клієнта, інтегрованих у прикладний/програмний рівень пристрою або шлюзу

Пряме підключення. У цьому випадку зв'язок з кінцевою точкою шлюзу хмари зазвичай кодується в пристрої або полі програмного рівня шлюзу з використанням бажаних протоколів. Це вимагає знання необхідних протоколів і повідомлень обмінюються шаблонами, але забезпечує повний контроль над реалізацією, включаючи формат даних на дроті.

Агенти. Агент - це компонент програмного забезпечення, встановлений на пристрої або польовому шлюзі, який виконує дії від імені іншої програми або компоненти управління. У IoT агенти зазвичай контролюються і діють на компоненти працює на хмарі backend. Наприклад, у випадку команди, надісланої пристрою, агент отримає можна виконати безпосередньо на пристрої.

Агенти можуть бути власними агентами, спеціально написаними для конкретного програмного рішення, або агентами на основі стандартів реалізації окремих стандартів, таких як OMA LWM2M. В обох випадках для розробників пристроїв зручно інтегрувати і покладатися на інкапсульовані можливості агентів; однак існують певні обмеження. Як правило, агенти представляють собою замкнуту систему, обмежену можливостями, наданими агентом для набору підтримуваних платформ.

Переносимість на інші платформи або налаштування та розширення за межі наданої функціональності зазвичай не є можливо.

Компоненти клієнта. Клієнтські компоненти надають набір можливостей, які можна інтегрувати в запущений програмний код на пристрої, щоб спростити підключення до сервера. Вони зазвичай надаються у вигляді бібліотек або SDK, які можна зв'язати або скомпільовані в програмний шар пристрою. Наприклад, якщо бекенда посилає команду на пристрій, клієнтські компоненти спрощують прийом команди, хоча виконання буде виконуватися в області дії програмний/програмний рівень.

У порівнянні з агентами, клієнтські компоненти вимагають інтеграційних зусиль у програмне забезпечення пристрою, але вони надають найбільшу гнучкість для розширюваності та портативності.

Варіанти технологій

SDK пристрої Azure IoT. SDK пристрої Azure IoT являють собою набір компонентів клієнта, які можна використовувати на пристроях або шлюзи для спрощення підключення до концентратора Azure IoT. Пристрої SDK можна використовувати для реалізації клієнта IoT, який полегшує підключення до хмари. Вони забезпечують послідовний досвід розвитку клієнта на різних платформах і допомагають абстрактно складність систем обміну повідомленнями від розробників пристроїв. Ці бібліотеки включають підключення гетерогенного діапазону пристроїв і польових шлюзів до рішення на основі IoT на основі Azure. Вони спрощують загальні завдання підключення шляхом абстрагування деталей базових протоколів і шаблонів обробки повідомлень. Бібліотеки можуть бути використані безпосередньо в додатку пристрою або для створення окремого агента, запущеного на пристрої, який встановлюється підключення з хмарним шлюзом і полегшує зв'язок між пристроєм і рішенням IoT backend.

SDK пристрої Azure IoT - це платформа з відкритим вихідним кодом, яка узгоджується з можливостями платформи Azure IoT. Поки ці бібліотеки спрощують підключення до концентратора Azure IoT, вони є необов'язковими і не потрібні, якщо розробники вибирають пристрої для

підключення до кінцевих точок IoT Hub за допомогою існуючих фреймворків і підтримуваних стандартів протоколу.

3.2 Сховище ідентифікації пристрою

Сховище ідентифікаційних даних пристрою є повноваженням для всіх даних ідентифікації пристрою. Воно також зберігає і дозволяє перевіряти криптографічні секрети для цілей автентифікації клієнта пристрою. Типове сховище ідентичності не надає жодного засобу індексації або пошуку, крім прямого пошуку за ідентифікатором пристрою; функціональна роль прийнятий іншим магазином, який зберігає модель домену конкретного додатку. Ці магазини в основному розділені з міркувань безпеки; пошук на пристроях не повинен дозволяти розкриття криптографічного матеріалу.

Крім того, обмеження сховища ідентифікації мінімальним набором атрибутів, керованих системою, допомагає забезпечити швидке реагування операції, а з іншого боку, схема сховища моделі домену визначається вимогами рішення.

Шлюз хмари покладається на інформацію в сховищі ідентифікаторів для цілей аутентифікації пристрою та управління. Сховище ідентичності може міститися в шлюзі хмари, або альтернативно, хмарний шлюз може окремо ідентифікувати пристрої.

Варіанти технологій

Рекомендується використовувати концентратор Azure IoT, який містить вбудоване сховище ідентифікації пристрою, який є повноваженням для зареєстрованих пристроїв і надає облікові дані захисту для кожного пристрою.

Коли використовується користувацький шлюз хмари, він також може покладатися на сховище ідентифікації IoT Hub і його аутентифікацію і можливості авторизації. Магазин ідентифікаційних даних повинен дозволяти доступ тільки до привілейованих частин системи, якщо це необхідно; користувацькі шлюзи будуть шукати необхідний матеріал аутентифікації з цього магазину.

Якщо не використовувати концентратор Azure IoT, зовнішні реалізації можуть бути реалізовані за допомогою Azure Cosmos DB, Azure Tables, Azure SQL.

3.3 Топологія та сховище об'єктів

Моделі пристроїв і додатків є фундаментальними для побудови бізнес-логіки додатків. Приклади включають можливість визначення та налаштування бізнес-правил, виконання пошуку підмножини пристроїв або додатків об'єктів, побудувати інтерфейс користувача та панелі інструментів, а також забезпечити узгодженість між різними компонентами рішення та інші резервні системи.

Моделі пристроїв часто описують:

• Схемою для метаданих про пристрій, включаючи характеристики та/або можливості пристрою. Метадані схема і значення змінюються рідко.

Прикладами метаданих пристрою є тип пристрою, марка, модель, серійний номер, ємність тощо.

• Схемою даних для даних, що видаються пристроєм, які визначають атрибути телеметрії разом з їх типами даних і дозволені діапазони.

Наприклад, пристрій моніторингу навколишнього середовища буде вимірювати температуру, яка визначається як назва атрибута: temp, data type: decimal, одиниця виміру: по Фаренгейту, а діапазон даних [10 - 110], а вологість визначається як атрибут ім'я: вологість, тип даних: десятковий, одиниця виміру: відсоток, діапазон даних [0-100].

• Схемою для параметрів конфігурації, які контролюють поведінку пристрою.

Наприклад, деяка поведінка пристрою моніторингу навколишнього середовища може контролюватися такими параметрами, як частота дискретизації, інтервал передачі телеметрії та режим роботи.

• Операції та параметри для керуючих дій, які може виконувати пристрій.

Наприклад, пристрій з підключеним приводом може виставляти віддалені операції, такі як поворот вліво (градуси), turn_right (градуси) і flash_warning_light (number_of_times).

• Топології пристроїв, що представляють модель домену, наприклад, багаті відносини між пристроями та іншими об'єктами, та семантичні зв'язки для ділового контексту бізнесу.

Наприклад, система управління будинком може використовувати модель домену, включаючи такі об'єкти, як кампус (або будівельний комплекс), будівля, підлога, приміщення, ресурс, пристрій і датчик. Модель топології визначає сутність атрибути (такі як властивості, операції тощо), а також відносини між сутностями.

Визначення та функції. Топологія і сховище сутностей - це база даних, яка містить об'єкти та відносини програми серед суб'єктів господарювання. Вона також містить метадані пристрою та атрибути для забезпечених пристроїв (представлені об'єктом пристрою в загальній топології).

Топологія і сховище об'єктів містять представлення моделі виконання. Посилання на Azure IoT архітектура не нав'язує будь-якої конкретної сутності або моделі пристрою, схеми або структури для метаданих пристрою. Це припускає, що вони визначені для конкретного рішення IoT в "проектний час", тобто під час розробки або динамічно під час конфігурації системи за допомогою відповідних засобів моделювання. Можна визначити власну модель програми або обрати вертикальну галузеву стандартну модель.

Під час надання послуг кожен пристрій реєструється з записом метаданих (екземпляр об'єкта пристрою) у топології та сховищі сутностей,

яке може містити структуровані та / або неструктуровані метадані, засновані на визначеній моделі (на етапі розробки).

Реєстр ідентичності пристроїв у порівнянні з топологією та сховищем об'єктів. Хоча сховище ідентифікаторів пристрою містить лише системні контрольовані атрибути і криптографічний матеріал, топологія і сховище об'єктів мають повне представлення пристрою, включаючи його відношення до інших юридичних осіб, таких як продукти, активи або машини. Запис у сховищі ідентичності визначає чи є пристрій зареєстрованим і може автентифікувати з системою. З міркувань безпеки слід дотримуватись належної практики відомості, пов'язані з безпекою, відокремлені від об'єкта пристрою. Магазин об'єктів не повинен зберігати будь-яку клавішу або іншу криптографічну інформацію інформацію, що стосується пристрою.

Зберігач ідентифікаційних даних пристрою є авторитетним списком ідентифікаторів пристроїв (насамперед для цілей автентифікації). И навпаки, топологія та сховище об'єктів має повний набір метаданих пристрою (атрибути пристроїв, властивості, операції тощо) серед відносин до інших прикладних об'єктів, необхідних для застосування для виконання своїх бізнес-функцій. Цей магазин є той, який використовується для виявлення пристрою, а також виявлення інших об'єктів прикладних програм і забезпечує індексацію досяжності і потужні можливості пошуку.

Топологія та сховище об'єктів є авторитетним сховищем для об'єктів та їх відносин для рішення IoT, що забезпечує

узгоджений вигляд у системі. З технічних причин проекції міститься інформації можуть бути збережені або

кешуються в інших компонентах для швидкого доступу. Однак джерелом істини для суб'єктів та їхніх відносин є цей магазин.

Зміни в ньому можуть бути необхідними для синхронізації або поширення до інших компонентів. Наприклад, деякий пристрій

Атрибути метаданих можуть знадобитися в близнюку пристрою концентратора IoT для оркестрування пристроями керування пристроями. В цьому випадку, зміни цих атрибутів одиниці пристрою повинні бути застосовані до близнюка пристрою IoT Hub. І навпаки, якщо атрибут Значення змінюється на пристрої близнюка (що надходить з пристрою), ця зміна буде поширюватися на пристрій об'єкта в топології та сховища сутностей. В інших випадках для певних завдань розширеної аналітики може знадобитися копія посилання на пристрій даних у певному компоненті або форматі зберігання.

Метадані. Відмінність між метаданими, що описують сам пристрій і оперативні дані, відображають стан пристрою або його робочого середовища важливо, оскільки він безпосередньо впливає на використання інформації про пристрій, кешуються і розподіляються по всій системі. Метадані, як правило, повільно змінюють дані, в той час як операційні дані очікується, що вони будуть швидко змінюватися.

Наприклад, геопозиція полюса світлофора є метаданими, але поточна геопозиція транспортного засобу розглядається оперативні дані. Ідентифікаційний номер, модель і марка транспортного засобу будуть метаданими. Виявлення всіх світлофорів певний ділянку дороги може бути виконаний як запит проти топології та сховища об'єктів, знаходячись на всіх транспортних засобах

В даний час водіння на певній ділянці дороги буде завданням аналізу над оперативними даними. Метадані в Топологія і сховище об'єктів можуть допомогти в якості довідкових даних для пошуку всіх транспортних засобів конкретної моделі на дорозі.

3.4 Забезпечення пристрою

Визначення. *Забезпечення* являє собою етап життєвого циклу пристрою, коли пристрій має бути відомий системі.

Провайдерський API - це загальний зовнішній інтерфейс для внесення змін до внутрішніх компонентів сервер, зокрема, сховище ідентифікаторів пристрою та сховище топології та об'єктної сутності. Він забезпечує абстрактний інтерфейс з загальні жести, і є реалізація цього абстрактного інтерфейсу для ідентифікації пристрою і топології і юридичних осіб. Реалізація може бути розширена до інших компонентів і систем.

Надання пристроїв, як правило, ініціюється на сервері, реєструючи пристрої в системі, перш ніж вони стануть оперативним. У деяких випадках це може статися під час виготовлення пристроїв (включаючи спалювання ідентифікатора пристрою та облікові дані, необхідні для підключення до інтерфейсу IoT). В інших випадках надання може бути виконано негайно перед увімкненням пристрою для використання, наприклад, під час встановлення пристрою. При першому спробі пристрою встановлюють підключення до бекенда, можуть бути виконані додаткові кроки для завершення його конфігурації (або "Bootstrap") для використання.

Робочий процес забезпечення. Робочий процес надання рішень вирішує питання обробки індивідуальних та об'ємних запитів реєстрація нових пристроїв та оновлення чи видалення існуючих пристроїв. Він також буде обробляти активацію і, можливо, тимчасове припинення доступу та можливе відновлення доступу. Це може також включати взаємодію з зовнішніми системами наприклад, M2M API оператора мобільного зв'язку для вмикання або вимикання SIM-карт або бізнес-систем, таких як платіж, підтримка або рішення для управління відносинами з клієнтами. Робочий процес надання послуг гарантує, що пристрій зареєстровано з усіма бекенд-системи, які повинні знати про його ідентичність і додаткові атрибути метаданих у разі потреби.

Процес завантаження. Коли пристрій хоче підключитися до системи вперше, додаткові кроки можуть бути виконується для завершення його конфігурації. Це може включати в себе конфігурацію або оновлення програмного забезпечення, які будуть застосовані раніше використання

пристрою. Під час етапу завантаження пристрій може бути призначений до нової "домашньої" кінцевої точки з новим облікові дані для неї. Це особливо важливо для багатопрофільних систем або глобально розподілених розгортань. Інформація про те, хто збирається використовувати пристрій (який орендар) або де буде використовуватися пристрій (географічне розташування), впливає на рішення про те, куди «додому» пристрій (тобто який хмарний шлюз буде відповідати за зв'язок з пристроєм). У багатьох випадках компонент резервування реалізується як "глобальна" послуга, яка делегує реєстрацію пристроїв у регіональних розгортаннях, а також в оркестрованих пристроях до їх «домашніх» кінцевих точок під час завантаження. В цьому випадку, глобальна служба може реалізувати робочий процес вищого рівня для надання, використовуючи ті ж або подібні зовнішні інтерфейси як регіональні компоненти, і делегувати операції регіональним компонентам забезпечення відповідно.

Варіанти технологій

Рекомендується використовувати службу забезпечення постачання пристроїв концентратора Azure IoT (DPS) для забезпечення пристроєм. DPS є глобальним послуга надання, яка підтримує реєстрацію та налаштування (тобто завантаження) пристроїв у декількох концентраторах IoT.

DPS спрощує автоматизацію надання пристроїв у сховище ідентифікаційних даних пристрою (частину концентратора IoT), забезпечуючи при цьому гнучкість управління розподілом пристроїв. DPS також пропонує API для резервних систем для реєстрації пристроїв як API для конфігурації пристрою (завантаження). DPS може використовуватися в робочому процесі для автоматизації розподіл пристроїв через IoT Hubs, поряд з іншими етапами реєстрації пристроїв в інших компонентах бекенда та системи (такі як сховище топології та об'єктна сутність або третій партійного провайдера). Для глобально розподілених розгортань, кожен з цих кроків може потребувати реєстрації в глобальній або регіональній системі.

Прикладні програми Azure API може використовуватися для реалізації зовнішнього API надання. API Apps надає платформу для створення, розміщення та розповсюдження API у хмарі та локальних мережах. API Apps легко інтегрується з Azure Logic

Програми, які можуть бути використані для реалізації всеохоплюючого робочого процесу забезпечення всім рішенням IoT компонентів і зовнішніх бізнес-систем.

Інтерфейс надання є простим набором жестів для керування життєвим циклом пристрою. Інтерфейс надання (API) повинні бути реалізовані як основний API через ідентифікацію пристрою і топологію і реєстр об'єктів і при необхідності додаткові компоненти внутрішнього розчину. Він використовується не тільки з інтерфейсу рішення (наприклад, портал адміністрування пристроїв), але також може служити інтерфейсом для робочих процесів більш високого рівня, з якими також можна взаємодіяти

зовнішніми системами, такими як M2M API оператора мобільного зв'язку для керування SIM-картами або для серверної бізнес-системи активацію рахунку за послугу.

Ключі безпеки можуть генеруватися за межами API і передаватися як параметри, або можуть бути створені і присвоєні служби як частина виклику API надання.

Створення маркера безпеки може бути виконано в API Provisioning з використанням необхідного ключа підпису. Маркер виданий пристрій буде обмежено за обсягом конкретною кінцевою точкою (наприклад, кінцева точка пристрою у випадку концентратора або події IoT Політика видавця хаба). Дані, що повертаються операцією Register і ResetCredentials, містять необхідну безпеку маркери, які повинні бути передані на пристрої. Крім того, маркери безпеки можуть генеруватися на пристрої або зовні і передається на пристрої.

Для користувацьких шлюзів необхідні облікові дані можуть бути згенеровані зовні і передані в API для зберігання або API можна розширити для створення ключів.

3.5 Зберігання

Визначення. Рішення IoT можуть генерувати значні обсяги даних залежно від кількості пристроїв у рішенні, як часто вони посилають дані, і розмір корисного навантаження в записах даних, відправлених з пристроїв. Дані часто є даними часових рядів і потрібно зберігати там, де його можна використовувати для візуалізації та звітування, а також для подальшого звернення до нього обробки. Зазвичай дані розділяються на "теплі" та "холодні" сховища даних. У сховищі даних зберігаються останні дані доступ до якого потрібно мати з низькою затримкою. Дані, що зберігаються в холодному сховищі, зазвичай є історичними даними. Найчастіше холодне вибране рішення для бази даних зберігання буде дешевше, але пропонує менше функцій запитів і звітування, ніж теплі рішення бази даних.

Загальноприйнятою для зберігання є збереження недавнього діапазону (наприклад, останнього дня, тижня або місяця) даних телеметрії у теплий накопичувач і зберігати історичні дані в холодному сховищі. За допомогою цієї реалізації програма має доступ до найостанніші дані і може швидко спостерігати останні дані і тенденції телеметрії. Отримання історичних даних для пристроїв може бути виконано за допомогою холодного зберігання, зазвичай з більш високою затримкою, ніж якщо б дані були в теплому сховищі. Для загального сценарію цілей рекомендується для Azure Cosmos DB для теплого зберігання та сховища Azure Blob для холодного зберігання. Якщо рішення вимагає частих запитів, які включають агрегацію на багатьох подіях і на багатьох пристроях, які ми рекомендуємо Часові ряди Дослідження для теплого зберігання. Нижче наведено докладний опис теплого та холодного зберігання та глибокого занурення в оцінювання технології, що проводиться для технології зберігання в кожній категорії.

Тепле зберігання. База даних теплого сховища зберігає стан пристрою за заздалегідь визначений останній інтервал і може також легко зберігати стан доступне останнє відоме стан на пристрій. Ці дані повинні бути доступними в базі даних швидко (в ідеалі в межах питання секунд, коли дані потрапляють у хмарний шлюз з пристрою) і легко запитуються для простих сценаріїв такі як візуалізація поточних значень датчика пристрою або візуалізація значень протягом останнього часу. Загальні шаблони запитів включають: дані для пристрою за останнім датами та часовим діапазоном, агреговані дані для одного або багатьох пристроїв, а також останні відоме значення для точки телеметрії для конкретного пристрою. Дані, що зберігаються в теплій базі даних, можуть бути необробленими даними, агреговані дані, або обидва вили.

Критерії оцінки

Теплі рішення для зберігання були оцінені на основі наступних критеріїв. Ці критерії не застосовуватимуться до кожного IoT рішення, але були розроблені, щоб бути найбільш загальноприйнятими у всіх рішеннях IoT.

1. Безпека. Рішення пропонує зрілі та надійні функції, такі як шифрування в стані спокою, аутентифікація та авторизації та мережевої безпеки.

2. Простота. Рішення добре документоване і має чітко визначену архітектуру. Завданнями розвитку є задокументовані, підтримуються наборами для розробки програмного забезпечення ("SDK") і можуть бути перевірені в локальному розвитку навколишнє середовище. Розгортання та операційні завдання підтримуються документацією, інструментами та інтерфейсами користувача.

3. Продуктивність. Читання і запис в базу даних відбувається швидко і масштабується до багатьох одночасних читання і записів. Запит продуктивність також швидко.

4. Масштабованість. База даних підтримує зберігання гігабайт до терабайт даних. Масштабування не вимагає простою. Ідеальне рішення автоматично пристосовує вартість та обчислювальну потужність до навантаження.

5. Можливість запиту. База даних має можливості запиту, необхідні для загального рішення.

6. Ціна. База даних доступна як для потужності зберігання, так і для пропускної здатності.

Рекомендується Azure Cosmos DB як тепле рішення загального призначення. Azure Cosmos DB - це безпечна, масштабована (без обмежень на зберігання або пропускної здатності), низька затримка бази даних NoSQL. це є найкраще для наборів даних, які можуть скористатися гнучкою схемою-агностиком, автоматичним індексуванням і інтерфейсами з багатими запитами.

Холодне зберігання.

Замість того, щоб зберігати всі дані в теплому сховищі даних з низькою затримкою, високою пропускною спроможністю і повноцінними можливостями запиту, дані можуть бути розділені на теплі та холодні шляхи зберігання. Це може забезпечити менші витрати на зберігання, зберігаючи при цьому історичні дані. А база даних холодного зберігання зберігає дані, які не потрібні так швидко і/або часто, як тепле сховище, але все ще може бути у майбутньому для доступу до звітності, аналізу, використання машинного навчання тощо

Критерії оцінки

Рішення для зберігання в холоді оцінювали на основі наступних критеріїв. Ці критерії не застосовуються до кожного рішення, але були розроблені як найбільш загальноприйняті для рішення IoT.

1. Безпека. Рішення пропонує зрілі та надійні функції, такі як шифрування в стані спокою, аутентифікація та авторизації та мережевої безпеки.

2. Простота. Рішення добре документоване і має чітко визначену архітектуру. Завданнями розвитку є Документовані, підтримуються наборами для розробки програмного забезпечення ("SDK"), і до певної міри можуть бути перевірені на локальному робоча станція. Розгортання та операційні завдання підтримуються документацією, інструментами та інтерфейсів користувача.

3. Масштабованість. База даних підтримує зберігання великої кількості даних. Масштабування не вимагає простою. База даних має дуже тривалий (на порядок років) або необмежений термін зберігання. Ідеальне рішення автоматично пристосовує вартість та обчислювальну потужність до навантаження.

4. Ціна. База даних доступна для великих обсягів даних.

Найкраща база даних холодного зберігання для рішення дуже залежить від того, якою метою буде обслуговувати базу даних. Два дані Рішення для зберігання даних нижче розроблені для високих масштабів за низькою ціною, але кожен має сильні сторони для різних сценаріїв. Ми рекомендувати Azure Blob Storage для загального випадку, оскільки це дешевше, ніж Azure Data Lake, особливо з точки зору запису наразі доступний у більшості регіонів і має краще відновлення після аварії. Однак якщо розчин вимагає холоду аналітичні дані зберігання (з Hadoop, аналітикою даних Azure тощо) або вимагає запитів за допомогою U-SQL, розроблено Data Lake з урахуванням цього сценарію і може бути кращим вибором.

Рекомендовано: сховище блоків Azure. База даних Azure Blob - це проста, недорога база даних зберігання файлів. Блоки можна використовувати зберігати необроблені дані пристрою. Використовуючи краплі сторінок замість блоку або додавати краплі, слід враховувати частота операцій запису

Лазурне озеро даних. Azure Data Lake - це розподілений сховище даних, яке може зберігати великі обсяги реляційних і нереляційні дані без перетворення або визначення схеми. Це хороший вибір для бази даних

зберігання, якщо великі дані необхідні аналітичні та / або необмежені місця для зберігання. Це трохи дорожче, ніж Azure Blob Storage (зокрема в терміни операцій запису), але він оптимізований для великих навантажень аналітики даних. Доступ до бази даних можна отримати з Hadoop через WebHDFS-сумісні REST API або з використанням мови U-SQL. Він має місцевий резервний зберігання і є доступні в деяких регіонах США, а також у Північній Європі. Ми рекомендуємо Blob Storage над даними озера через незначна премія в ціні, менша регіональна доступність і відсутність гео-резервного зберігання.

3.6 Потік даних і обробка потоків

Оскільки дані потрапляють в інтерфейс IoT, важливо зрозуміти, як може змінюватися потік обробки даних.

Залежно від сценаріїв та застосувань, записи даних можуть проходити через різні етапи, об'єднані в іншому порядку, і часто обробляються паралельними паралельними завданнями.

Ці етапи можна класифікувати за чотирма категоріями: зберігання, маршрутизація, аналіз і дія/відображення:

• Зберігання включає в себе кеші пам'яті, тимчасові черги та постійні архіви.

• Маршрутизація дозволяє відправляти записи даних до однієї або більше кінцевих точок зберігання, процесів аналізу та дій.

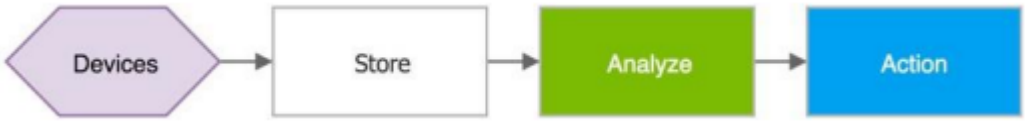
• Аналіз використовується для запуску записів вхідних даних через набір умов і може виробляти різні вихідні дані записів. Наприклад, вхідна телеметрія, закодована в Avro, може повернути вихідну телеметрію, закодовану у форматі JSON.

• Записи оригінальних вхідних даних і записи вихідного аналізу зазвичай зберігаються і доступні для відображення, а також можуть ініціювати такі дії, як електронні листи, миттєві повідомлення, квитки на випадок, завдання CRM, команди пристроїв тощо Ці процеси можуть бути об'єднані в прості графіки, наприклад, для відображення необробленої телеметрії, отриманої в реальному часі, або більше складні графіки, які виконують кілька розширених завдань, наприклад, оновлення інформаційних панелей, ініціювання тривог, і почати бізнес-інтеграційні процеси і т.д.

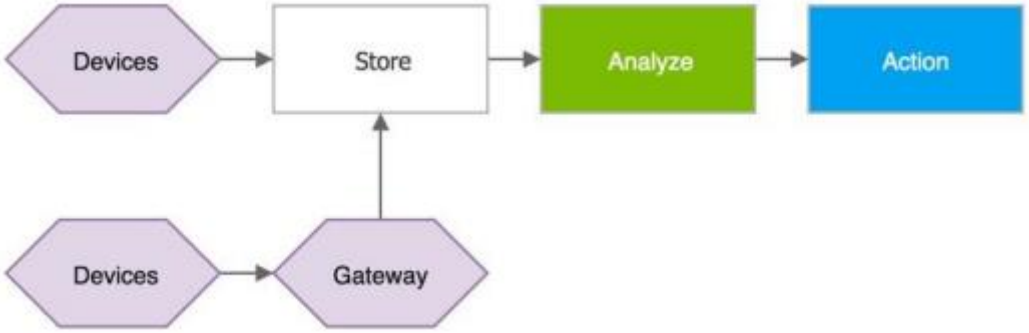
Наприклад, наступний графік являє собою простий сценарій, в якому пристрої відправляють записи телеметрії, які тимчасово зберігається в концентраторі Azure IoT, а потім негайно відображається на графіку на екрані для візуалізації:

Наступний графік представляє інший поширений сценарій, в якому пристрої відправляють телеметрію, зберігають її в короткостроковій

перспективі в Azure IoT Hub, незабаром після аналізу даних для виявлення аномалій, викликають такі дії, як електронна пошта, текст SMS, миттєве повідомлення повідомлення тощо:



Архітектура IoT може також складатися з декількох точок прийому. Наприклад, деяке зберігання та / або аналіз телеметрії може відбуваються на місці, в межах пристроїв і польових / крайових шлюзів; для підключення можуть знадобитися переклади протоколів обмежених пристроїв до хмари. Хоча отриманий графік є більш складним, логічні будівельні блоки є однаковими:



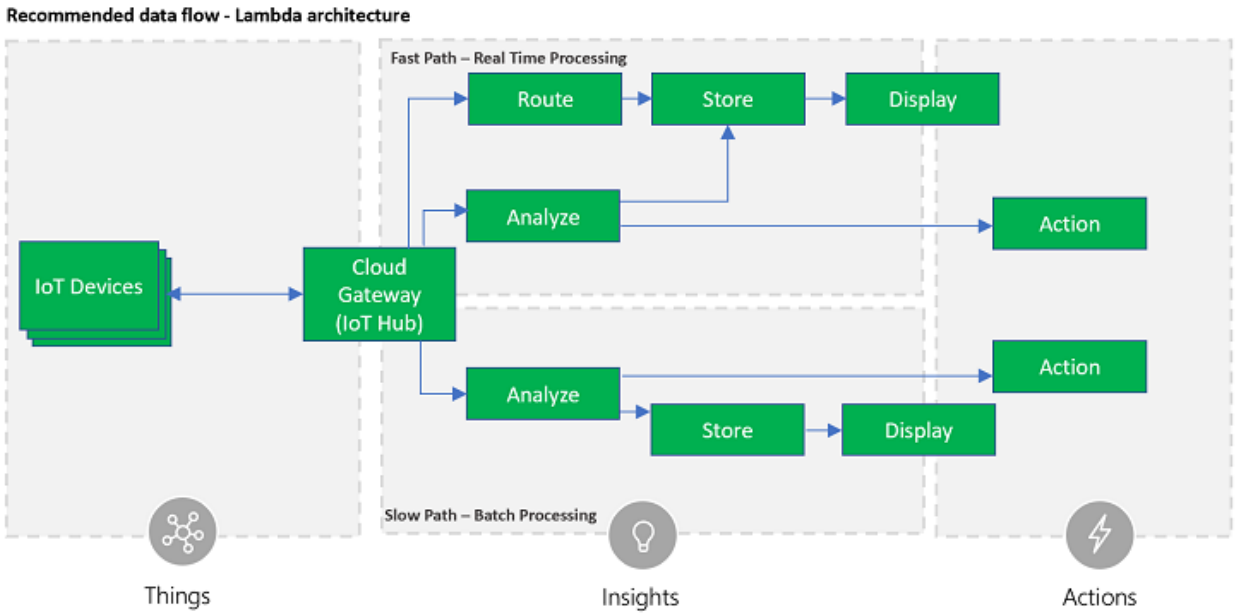
Рекомендований потік даних

Ця довідкова архітектура припускає, що бізнес запускає кілька паралельних потокових процесорів, або шляхом секціонування вхідний потік, або шляхом пересилання записів даних до декількох конвеєрів. Рекомендується розділити на основі властивості повідомлень (наприклад, ідентифікатор пристрою, розташування пристрою, формат корисного навантаження тощо), щоб уникнути де-серіалізації корисного навантаження перед маршрутизацією але документ містить рішення, здатні здійснювати маршрутизацію на основі вмісту повідомлень JSON.

Наступний графік, також відомий як Lambda архітектура, показує рекомендований потік повідомлень від пристрою до хмари і події в рішенні IoT. Записи даних проходять через два різних шляхи:

1. Швидкий процес архівування та відображення вхідних повідомлень, а також аналіз цих записів термін критичної інформації і дій, таких як тривоги.

2. Повільний конвеєр обробки, що виконує складний аналіз, наприклад, поєднуючи дані з декількох джерел і протягом більш тривалого періоду часу (наприклад, годин або днів) і створення нової інформації, наприклад, звітів, машин моделей навчання тощо.



У лямбда-архітектурі швидкий потік даних обмежений вимогами затримки, тому існує обмеження на складність аналізу можлива. Часто це вимагає компромісу певного рівня точності на користь даних та аналізу що готовий якомога швидше. Наприклад, функції усереднення і аналіз тренда можуть бути виконані тільки на а обмежений обсяг даних, як правило, порядку декількох секунд.

З іншого боку, дані, що потрапляють у повільний шлях, не підпадають під дію тих самих вимог затримки і дозволяють високі Обчислення точності на великих наборах даних, що може бути дуже інтенсивним. Також помітно, що повільний аналіз шляхів результати можуть бути використані шляхом швидкої аналітики шляхів; напр. для вирішення проблеми може знадобитися розрахунок середнього показника надходжень за тиждень даних і забезпечують, що середнє значення для використання в якості довідкових даних для швидких обчислень шляху.

Варіанти технологій

Існує декілька служб Azure та сторонніх розробників, які можна використовувати та комбінувати для створення надійного та масштабованого IoT архітектури, однак, при виборі служби для розгортання, деякі аспекти слід розглядати першими:

• Безстатистичний vs stateful: де це можливо, рішення повинно реалізувати процесори без стаціонару, щоб зменшити експлуатацію вартості та збільшення доступності. З іншого боку, обробка даних, що здійснюється за допомогою штату, дозволяє проводити більш різноманітний аналіз і часто є для реалізації функцій більш високого рівня.

• Статичні проти динамічних правил: якщо правила аналізу не змінюються і не посилаються на зовнішні дані, які змінюються, то це можна вибрати більш прості технології за нижчою ціною. У сценаріях, де потрібно більше гнучкості

підтримують змінні навантаження, часті зміни в логіці обробки потоків і змінні зовнішні довідкові дані

доступні технології є більш складними і дорогими для розгортання.

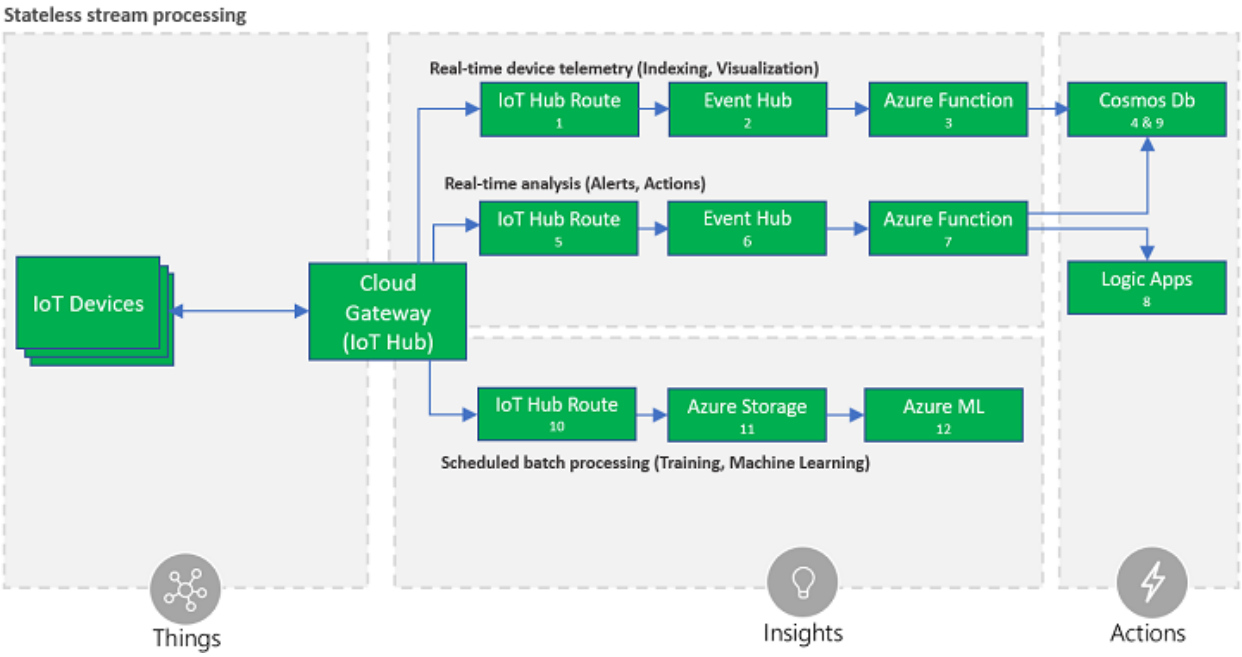
Документ представляє два варіанти, один для розв'язання простих сценаріїв з процесорами без статусу та досить статичними правилами,

і один складний сценарій, наприклад, процесори, що підтримують стан, з логікою динамічного аналізу та довідковими даними.

Наступні рішення представлені з припущенням, що керовані Azure послуги збільшують загальну систему безпеки та зниження витрат на установку та обслуговування. З іншого боку, розробники можуть створювати рішення гетерогенні системи, що поєднують керовані послуги з власними, сторонніми або відкритими компонентами, такими як Spark та Cassandra, використовуючи інші пропозиції Azure, такі як віртуальні машини Azure, послуги контейнерів Azure, та Azure HDInsight.

Обробка потоку без статусу

Наступна архітектура надає рішення для швидкого та масштабованого аналізу записів даних в реальному часі, в сценарії, де необхідний лише аналіз без участі громадян, використовуючи невеликий набір простих логічних правил. Також включено повільний шлях, дозволяє виконувати більш складний аналіз, наприклад, завдання на машинне навчання, без обмежень швидкості швидкий шлях.



Ця архітектура рекомендується для сценаріїв, де записи вхідних даних серіалізуються в JSON, і правила обробки беруть вхідне повідомлення за один раз, не враховуючи історичних даних. Архітектура використовує можливість визначення умов на корисному навантаженні (# 5) в Azure IoT Hub, щоб переслати лише певні повідомлення та дії тригера в послуги, підключені через логічні програми (№ 8).

Один маршрут концентратора Azure IoT також використовується для пересилання всієї телеметрії (# 1) до функції Azure (# 3), яка може перетворити її в іншого формату, наприклад, приєднуючись до зовнішньої інформації та зберігаючи її в Azure Cosmos DB (# 4) для теплого зберігання споживання, напр. відображення на інформаційній панелі.

Інший маршрут-концентратор Azure IoT (# 10) використовується для копіювання всіх вхідних записів даних до блоків зберігання Azure (# 11) для холодного зберігання, де воно може бути заархівоване на невизначений час за низькою ціною, і легко доступне для пакетної обробки, наприклад Azure

Завдання з наукової роботи з даними машинного навчання (№ 12).

Переваги архітектури:

1. Висока доступність завдяки географічній надмірності та швидким можливостям відновлення після аварії служб Azure.

2. Низька вартість: більшість компонентів автоматично масштабуються, пристосовуючись до змінної навантаження, мінімізуючи витрати коли немає даних для обробки.

3. Мінімальні експлуатаційні витрати, оскільки всі компоненти керуються службами Azure.

4. Гнучкість: функції Azure і Azure Cosmos DB дозволяють трансформувати дані в будь-яку бажану схему,

підтримка декількох шаблонів доступу та API, таких як API MongoDB, Cassandra і Graph.

5. Дії та бізнес-інтеграція: широкий вибір інтеграцій доступний через програми Logic Apps та Azure ML.

Коли використовувати цю архітектуру:

1. Записи вхідних даних серіалізуються у форматі JSON.

2. Необхідно невелика кількість правил. В даний час концентратор Azure IoT підтримує до 100 маршрутів.

3. Записи даних можна аналізувати по черзі; тобто немає необхідності агрегувати дані по безлічі точок даних

(наприклад, усереднення) або потоки даних (наприклад, об'єднання даних з декількох пристроїв).

Обробка потокового потоку

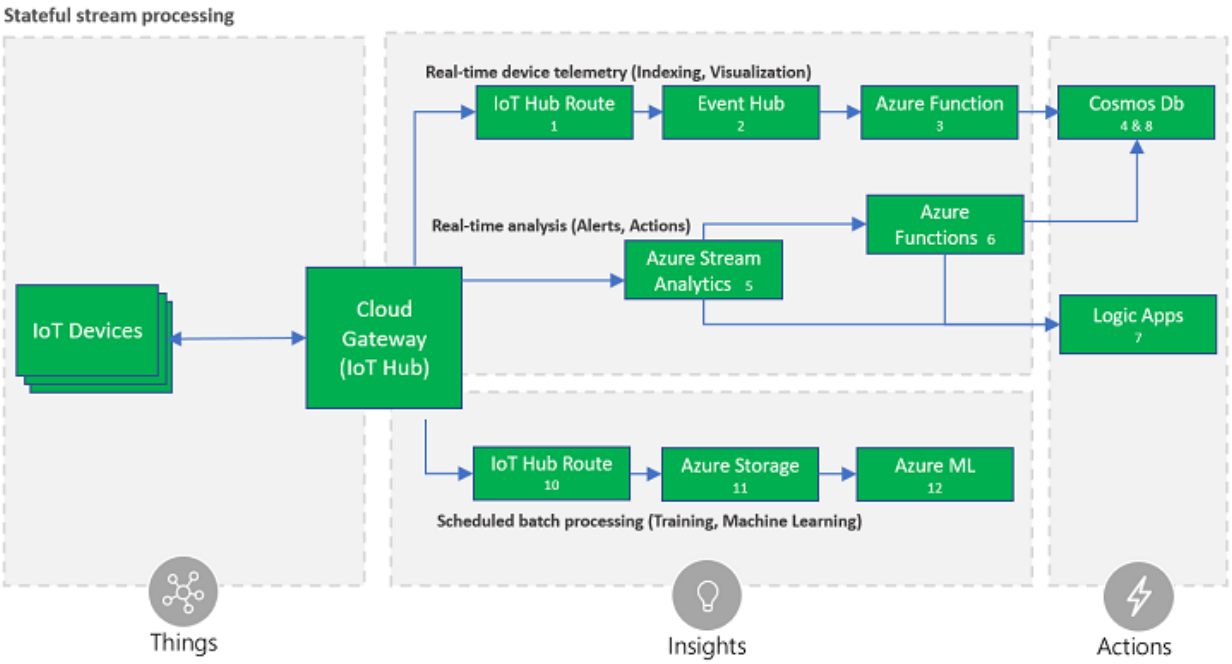
Наведена нижче архітектура описує швидке, гнучке і масштабоване рішення для аналізу реальних даних в реальному часі

записи в декількох форматах, з можливістю посилатися на зовнішні дані, без обмежень попереднього

архітектури, за рахунок більших експлуатаційних витрат.

Архітектура включає в себе той же самий повільний шлях, який спостерігається в попередній архітектурі для використання з машинним навчанням та іншим

Комплексний аналіз неможливий на швидкому шляху.



Архітектура схожа з рішенням, рекомендованим для обробки без тривалості, тільки шлях заміни аналізу

Аналітика Azure Stream (ASA) (# 5).

ASA призначений для гіпер-масштабного аналізу і маршрутизації записів даних, що є в стані, з можливістю застосування

складні запити протягом періодів часу і декількох потоків. Запити визначаються за допомогою SQL-схожої мови, що дозволяє

перетворення і обчислення. Служба підтримує запізнілі (до 21 дня) та позачергові (до однієї години) події,

при обробці за часом застосування35

Таким чином, висновок затримується на різницю в часі.

ASA також гарантує точно після доставки на підтримувані пункти призначення, з невеликою кількістю документів36

винятки, які можуть

генерувати дублікати. Мова запитів дозволяє оптимізувати аналіз продуктивності за допомогою розпаралелювання і розриву

запити.

ASA також підтримує записи даних у форматі Avro - компактний двійковий формат, який використовується для зменшення затримки та витрат на пропускну здатність.

Окрім обробки потоку ASA, в цій архітектурі використовується один маршрут маршрутизації Azure IoT Hub

телеметрії (# 1) до функції Azure (# 3), яка може перетворити її в інший формат, наприклад, приєднання до зовнішньої інформації,

і зберігати його в Cosmos DB (# 4) для споживання, наприклад, відображення на інформаційній панелі.

Окремий маршрут концентратора Azure IoT (# 10) використовується для копіювання всіх вхідних записів даних до блоків зберігання Azure (# 11), де

можуть бути архівовані на невизначений термін за низькою ціною і легко доступні для пакетної обробки, такі як Azure Machine Learning

завдання з наукової інформації (№ 12).

Переваги архітектури:

1. Висока доступність завдяки географічній надмірності та швидкому відновленню аварійних функцій служб Azure.

2. Мінімальні експлуатаційні витрати, оскільки всі компоненти керуються службами Azure.

3. Здатність Azure Stream Analytics виконувати комплексний аналіз в масштабі, наприклад, використання обкатки / ковзання / стрибків

вікна, агрегації потоків і зовнішні джерела даних.

4. Гнучкість: функції Azure і Cosmos DB дозволяють трансформувати дані в будь-яку бажану схему,

підтримка декількох шаблонів доступу та API, таких як API MongoDB, Cassandra і Graph.

5. Дії та бізнес-інтеграція: широкий вибір інтеграцій доступний через програми Logic Apps та Azure ML.

6. Продуктивність: підтримка двійкових потоків даних, щоб зменшити затримку.

Коли для реалізації цієї архітектури:

1. Записи вхідних даних вимагають складного аналізу, такого як часові вікна, агрегація потоків або об'єднання з зовнішніми

джерел даних, що неможливо з архітектурою без громадянства.

2. Логіка обробки складається з декількох правил або логічних одиниць, які можуть зростати з часом.

3. Вхідна телеметрія ріалізується у двійковому форматі, як Avro.

3.7 Рішення для користувача інтерфейсу

Інтерфейс користувача рішення (UI) зазвичай включає веб-сайт і звітність, але також може включати веб-служби і мобільний

або настільний додаток.

Рішення UI може забезпечити доступ і візуалізацію даних пристроїв і результатів аналізу, через виявлення пристроїв

можливостей реєстру, команд та керування та робочих процесів забезпечення. У багатьох випадках кінцеві користувачі будуть повідомлені про це

попередження, умови тривоги або необхідні дії, які необхідно здійснити через повідомлення поштовху.

Інтерфейс рішення також може забезпечувати або інтегрувати з живими та інтерактивними панелями, які є придатною формою

візуалізації сценаріїв IoT з великою кількістю пристроїв.

Рішення IoT часто включають в себе гео-розташування та гео-орієнтовані послуги, а інтерфейс користувача повинен забезпечити відповідний контроль та

можливостей.

Як зазначено на початку цього документу, безпека є критичною, а рішення інтерфейсу користувача, що забезпечує контроль над системою і

пристрої повинні бути захищені відповідним чином за допомогою контролю доступу, диференційованого за ролями користувачів і залежно від

авторизації.

Варіанти технологій

Служба прикладних програм Azure - це керована платформа з потужними можливостями для створення веб-і мобільних додатків для багатьох платформ і мобільних пристроїв. Веб-додатки та програми для мобільних пристроїв дозволяють розробникам створювати веб-програми та мобільні програми мови, такі як .NET, Java, NodeJS, PHP або Python. Крім того, Azure API Apps дозволяють легко експортувати та керувати API, до яких можуть звертатися мобільні або веб-клієнти.

Azure з часовими рядами Azure (TSI) включає оптимізований для користувача часовий ряд, включаючи графіки, теплову карту, а перспективний перегляд для порівняння візуалізації та статистики бізнес-аналітики, пов'язаної з базовими даними.

TSI також пропонує бібліотеку керування JavaScript, що полегшує інтеграцію візуалізації даних, що зберігаються в TSI, у звичайному режимі програми.

Бібліотека управління TSI JavaScript дозволяє розробникам з існуючими додатками вбудовувати діаграми для їх візуалізації Дані IoT. Якщо користувач хоче створити нову програму, починаючи з прискорювачів рішень Azure IoT (віддалений Моніторинг або підключена фабрика) є рекомендованим підходом. Ці відкриті джерела опорної архітектури реалізації використовують бібліотеку керування JavaScript TSI для візуалізації з коробки і полегшують необхідність створення користувацький веб-додаток з нуля. Зауважте, що ці прискорювачі рішень є почерговими, готовими до виробництва прикладами IoT рішення; вони включають реалізації для обробки потоку, зберігання тощо Концентратори повідомлень Azure дозволяють надсилати push-повідомлення на персональні мобільні пристрої (смартфони та планшети). Це підтримує платформи iOS, Android, Windows і Kindle, а також абстрагує деталі різних платформ системи оповіщення (PNS). За допомогою одного виклику API сповіщення може націлюватися на окремого користувача або на сегмент аудиторії велика кількість користувачів.

На додаток до традиційного інтерфейсу, панелі інструментів дуже важливі для сценаріїв IoT, оскільки вони забезпечують природний шлях

агреговані види та допомагають візуалізувати величезну кількість пристроїв. Power BI - це хмарна служба, яка надає легкий доступ

спосіб створення багатих, інтерактивних панелей для візуалізації та аналізу. Power BI також пропонує живі панелі, які

дозволяють користувачам стежити за змінами в даних і показниках. Power BI містить власні програми для настільних і мобільних пристроїв.

Іншою зручною технологією для візуалізації IoT є Maps Azure.

37

API служб "Карти Azure" включає елементи керування картами та

послуги, які можна використовувати для включення карт Azure в програми та веб-сайти. Крім інтерактивних і статичних

Карти, API надають доступ до геопросторових функцій, таких як геокодування, дані про маршрут і трафік, і джерела просторових даних

які можуть бути використані для зберігання та запиту даних, які мають просторовий компонент, наприклад розташування пристроїв.

Веб-і мобільні програми можуть бути інтегровані з Azure Active Directory (AAD) для контролю автентифікації та авторизації.

Програми залежатимуть від управління ідентифікаційними даними користувача в AAD і можуть забезпечувати контроль доступу на основі ролей для застосування

функціональність. У багатьох випадках між пристроями IoT і користувачами (або між групами. \ T

пристроїв і груп користувачів). Наприклад, пристрій може перебувати у власності когось, використовуватися кимось іншим і встановлюватися або

ремонтується іншим користувачем. Подібні приклади можуть бути вірними для груп пристроїв і користувачів. Дозволи та на основі ролей

Управління доступом може управлятися як частина матриці асоціації між ідентифікаціями пристроїв (зберігається в пристрої

ідентифікаційного сховища) та ідентифікацій користувача, якими керує AAD. Конкретна конструкція цієї матриці, деталізація дозволів, і

Рівень контролю буде залежати від конкретних вимог рішення. Ця матриця може бути реалізована зверху пристрою

або використовувати окремий магазин, використовуючи різні технології. Наприклад, реєстр пристроїв може бути реалізований

за допомогою БД Cosmos, в той час як асоціація і матриця дозволів можуть бути побудовані з використанням реляційної бази даних SQL. Будь ласка, запиши

що ця тема обговорюється в цьому розділі, оскільки аутентифікація та авторизація користувачів відображаються як частина UX;

однак фактична реалізація буде поширена на декілька базових компонентів, включаючи реєстр пристроїв

і бекенда програми, розглянута в наступному розділі.

3.8 Моніторинг та ведення журналу

Системи реєстрації та моніторингу IoT-рішень використовуються для визначення того, чи функціонує рішення, як очікувалося, і щоб допомогти усунути неполадки у вирішенні. Системи моніторингу та реєстрації допомагають відповісти на наступне оперативні питання:

• Чи існують помилки в пристроях або системах?
• Чи правильно налаштовані пристрої або системи?
• Чи виробляють пристрої або системи точні дані?
• Чи задовольняють системи очікування як для бізнесу, так і для кінцевих клієнтів?

Системи моніторингу та реєстрації допомагають відповідати на ці питання, а коли відповідь «ні», вони виходять на поверхню відповідну інформацію для операційних груп, щоб допомогти пом'якшити проблеми.

Системи реєстрації та моніторингу IoT-рішень часто є більш складними, ніж стандартні види бізнесу програми. Складність виникає через те, що рішення IoT охоплюють:

• Фізичні датчики, що взаємодіють із середовищем.
• Програми на інтелектуальних технологіях, що забезпечують формування даних, переклад протоколів тощо.
• Інфраструктурні компоненти, такі як шлюзи на місці, брандмауери та комутатори.
• Послуги прийому та обміну повідомленнями.
• Механізми наполегливості.
• Застосування інсайту та звітності.
• Підсистеми, які працюють і масштабуються незалежно в хмарі.

Наступний компонент складності рішення IoT виникає через те, що існує різноманітний набір зацікавлених сторін, у тому числі:

Internal Stakeholders
IT & Operations
Security Teams
Field Technicians & Service Personnel
Application Developers
Data Scientists
Business Teams & Executives

External Stakeholders
Suppliers & Partners
Customers
Compliance and Audit Specialists

Рішення з моніторингу та реєстрації може включати численні спеціальні прикладні програми та бібліотеки, націлені на кожну з них підсистеми рішення IoT. Засоби реєстрації та моніторингу зазвичай складаються з наступних чотирьох компонентів:

• Інструменти для візуалізації продуктивності системи та часової шкали - для моніторингу системи та для основного усунення несправностей.
• Буферизація даних із буферизацією - на дані журналу буферів (які можуть бути багатозначними).

• Стійкість зберігати - зберігати дані журналу.

• Можливості пошуку та запиту - для перегляду даних журналу для використання при детальному усуненні несправностей.

Широкомасштабні рішення IoT можуть складатися з багатьох менших підсистем. Часто розумно розгортати кілька екземпляри компонентів реєстрації та моніторингу для кожної з цих систем, причому екземпляри вищого рівня агрегують дані та аналізи з систем нижнього рівня. Наприклад, в дистанційному моніторингу рішення прискорювача кратне підсистеми (концентратор IoT, Cosmos Db, Azure Stream Analytics, користувацькі мікросервіси та багато іншого) використовуються для забезпечення оператора можливостей для пристроїв IoT. Ведення журналів для підсистем здійснюється на індивідуальному рівні, а потім може бути агреговано надають кінцевий вигляд рішення.

Зовнішні зацікавлені сторони

Постачальники та партнери

Клієнти

Спеціалісти з відповідності та аудиту

Широкомасштабні рішення IoT можуть складатися з багатьох менших підсистем. Часто розумно розгортати кілька екземпляри компонентів реєстрації та моніторингу для кожної з цих систем, причому екземпляри вищого рівня агрегують дані та аналізи з систем нижнього рівня. Наприклад, в дистанційному моніторингу рішення прискорювача кратне підсистеми (концентратор IoT, Cosmos Db, Azure Stream Analytics, користувацькі мікросервіси та багато іншого) використовуються для забезпечення оператора можливостей для пристроїв IoT. Ведення журналів для підсистем здійснюється на індивідуальному рівні, а потім може бути агреговано надають кінцевий вигляд рішення.

При проектуванні систем реєстрації та моніторингу необхідно враховувати стійкість та надмірність. Збільшилася кількість журналів і ефективність моніторингу може бути отримана шляхом спільного розміщення рішення поряд з системами; однак, стратегія приходить ризик системного відключення і ризик несанкціонованого впливу на систему реєстрації / моніторингу Масштаб самого базового рішення IoT. Раннє оцінювання "ризику невдачі" системи моніторингу забезпечить розробка надійних рішень.

Моніторинг і візуалізація

Системи моніторингу забезпечують розуміння стану здоров'я, безпеки та стабільності, а також ефективності рішення IoT. На високій Системи моніторингу забезпечують швидкий огляд того, чи функціонує рішення з кінця до кінця, як очікувалося.

Системи моніторингу також можуть надавати більш детальний перегляд, запис змін конфігурації компонентів і забезпечення витягнуті дані реєстрації, які можуть призвести до потенційних уразливостей безпеки, підвищити

процес управління інцидентами, і допомогти власникові системи усунути неполадки. Комплексні рішення моніторингу включають в себе можливість інформація запитів для конкретних підсистем або агрегування в декількох підсистемах.

Розробка системи моніторингу повинна починатися з визначення здорової роботи, відповідності нормативним вимогам та аудиту вимоги. Зібрані показники можуть включати:

• Фізичні пристрої, крайні пристрої та компоненти інфраструктури, що повідомляють про зміни в конфігурації; напр. відкритої мережі портів, виправлень, послуг та користувачів (для аудиту та відповідності), а також загальні операційні параметри, такі як споживання енергії, процесор, пам'ять і використання диска.

• Програми, що повідомляють про зміни в конфігурації, журнали аудиту безпеки, ставки запитів, час відповіді, частоту помилок і статистику збору сміття для керованих мов.

• Бази даних, зберігання записів і кеш-звітність запитів і продуктивності запису, зміни схеми, аудит безпеки журнал, замки або замикання, продуктивність індексу, використання ЦП, пам'яті та диска.

• Керовані служби (IaaS, PaaS, SaaS та FaaS), які повідомляють про показники здоров'я та зміни конфігурації, які впливають залежність системи від працездатності і продуктивності.

Необхідно дбати про збалансування витрат на продуктивність збору та зберігання показників у порівнянні зі значенням результатів дослідження надаються.

Стратегії збору показників часто ускладнюються контекстами розширеної безпеки, характерними для рішень IoT.

Програми на пристроях можуть бути розміщені на базі операційної системи хоста, що перешкоджає збору метрик пристрою через додатки, розгорнуті до них, вимагаючи додаткових програмних рішень для полегшення збору. Додатково тиск на проектування безпеки системи виникає внаслідок вимог аудиту та відповідності. Моніторинг SaaS, PaaS та IaaS компоненти часто мають сертифікати що дозволить звузити межі аудиту для задоволення відповідності нормам вимоги.

Візуалізація метрик моніторингу сповіщає зацікавлених сторін про нестабільність системи та сприяє реагуванню на інциденти.

Візуалізація повинна бути пристосована до ролей зацікавлених сторін і забезпечити розширення, щоб пристосувати зростання рішень і дозрівання. Дані, що подаються операторам, повинні обмежуватись діяльними показниками, які можуть бути пов'язані з ними змінюється стан системи, забезпечуючи при цьому можливість глибокого занурення в конкретні проблеми за потреби. Строки візуалізації повинні надавати корельовані дані, такі як розгортання, зміни конфігурації або попередні інциденти. Доданий контекст ці корельовані дані покращують прийняття рішень і прискорюють час відповіді.

Трасування телеметрії

Телеметрия трасування дозволяє оператору слідувати по шляху телеметрії від створення через систему.

Трасування важливо для налагодження та усунення несправностей. Для рішень IoT, які використовують концентратор Azure IoT і Azure Пристрої SDK, відстеження дейтаграм можна створювати спеціально, як Cloud-to-Device повідомлення і включені в телеметрію потік. Ідентифікатори трасування та прапори повідомлень дозволяють відслідковувати дейтаграми через потоковий ланцюжок обробки системи пропонуючи операційні ідеї на цьому шляху.

У рішеннях IoT, де тільки диференційна телеметрія (наприклад, тільки зміна температури охолодження) передається до послуга прийому, аналогічний підхід для відстеження повідомлень також може бути використаний для побудови схем серцебиття. Серцебиття схеми використовуються для забезпечення того, щоб пристрої з низькою швидкістю обміну повідомленнями залишалися активними, рідко використовувалися повідомленнями з'єднання залишаються живими, а також вказують на прогрес уперед під час тривалих завдань. Телеметрия серцебиття може бути використовуються для візуалізації продуктивності будівлі, звіти про рівень обслуговування (SLA) і можуть бути складені в IoT інструменту моніторингу рішення.