

8 attributes of successful IoT solutions

Contents

Scale	3
Device management	4
Big data management	5
Analytics, insights, and actions	5
High availability and disaster recovery	6
Security and compliance	7
Managing IoT solutions with DevOps	8
Understanding total cost of ownership (TCO)	9
Determining your IoT solution approach	9
Get started today	10






The Internet of Things (IoT) is helping companies across industries improve their products, develop new revenue streams, and improve customer experiences. At its core, IoT is about creating solutions that achieve business outcomes. According to the recent [Microsoft IoT Signals report](#), which draws from a survey of over 3,000 organizations, businesses that have adopted IoT estimate they will see a 30% ROI on business investments in IoT, and 88% see IoT as critical to business success.¹ Despite the substantial value IoT can provide, the high degree of complexity creates pitfalls and roadblocks that cause many projects to fail.

Many IoT initiatives fail because of unforeseen costs, pricing challenges, security concerns, or general lack of knowledge. Among those who have had projects fail in the POC stage, 32% said the main reason for failure was the cost of scaling.¹ Among IoT adopters, 47% say they don't have enough skilled workers, and 38% say that technical complexities are the key barrier preventing them from investing more in IoT.¹ These obstacles raise the question: how can organizations avoid these issues to develop scalable, secure, efficient, and profitable IoT solutions?

At Microsoft, we've identified 8 key attributes of IoT solutions that must be addressed to reduce the risk of failure. Throughout this paper, we will take a detailed look at each of these attributes and provide concrete guidance for putting your IoT solution on a path to success.

Over 90% of companies experience challenges in the proof of concept stage.¹

Among businesses that have adopted IoT:

-  **30%** estimate they will see a ROI on business investments in IoT
-  **88%** see IoT as critical to business success
-  **32%** said the main reason for failure was the high cost of scaling
-  **47%** say they don't have enough skilled workers
-  **38%** say that technical complexities are the key barrier preventing them from investing more in IoT

1 Scale



The architecture of an IoT solution is complex. It typically consists of connected devices, stream processors, storage services, integrated business applications, and user interfaces for managing the solution and analyzing data. Thoughtful planning that accounts for long-term objectives will ensure your solution can organically and seamlessly grow to meet your business needs.

Keep your end goal in mind and the architecture that is needed to get you there. Many IoT solutions fail because architecture and solution design are based on a proof of concept (POC) and neglect to determine what the architecture should look like when fully rolled out, or in 'steady state.' Those efforts run into issues when trying to scale from a small number of devices and sensors in POC to full scale solutions.

¹ <https://azure.microsoft.com/en-us/iot/signals/>

For example, scaling from hundreds to millions of devices will require a corresponding increase in data processing and storage requirements. Organizations also have difficulty scaling from a single-tenant to a multi-tenant solution if they did not plan for that evolution in advance.

Next, you must decide what service capacities are right for your solution. A typical IoT solution architecture includes several services: storage, cloud computing, networking, data, device management, and more. Frequently, companies choose to pay for high resource capacity up front to ensure they have room to grow, but it is difficult to estimate how much capacity is needed.

The alternative option is to pay for lower resource capacity up front and increase it incrementally, but this increases the risk of running into resource throttling. If you choose this approach, it's imperative to create a process for monitoring resource utilization across the board to ensure you don't exceed your service limits.

Finally, it is necessary to consider advanced growth strategies that will allow you to scale more seamlessly. One such strategy is to group devices into larger geographically distributed deployment 'stamps' of your IoT solution that can be easily and repeatedly deployed in multiple regions across the globe. Creating stamps allows you to surpass the maximum device caps set by individual cloud resources and it prevents issues in one stamp, like a regional outage, from affecting the rest of your resources.

2 Device management



With any IoT solution designed to operate at scale, a well-structured approach to device management is paramount. You must not only consider how to manage the devices themselves but also how to manage the data they are sending.

First, consider how you might provision, deprovision, manage, monitor and update your IoT devices remotely. Over time, you will need to reassign devices to different technicians or locations, remove devices from production, update devices with new settings or firmware, and monitor device health. You will also likely need to track and manage the status of each device during bulk provisioning and other bulk operations. For example, imagine you deploy an update to 7,000 connected devices but only 5,000 devices received the update. You would need a way to know exactly which devices were not updated so you can complete the updates without wasting time and resources.

Device data transmission patterns require equal attention. A key step is determining what data your devices will send to the cloud and how often. Some business scenarios require frequent data collection, while others don't. For example, you do not need to send device temperature every minute if the temperature only changes a few times a day. It is challenging to estimate how "chatty" your devices should be, but it's easy to find yourself collecting more data than needed. The goal is to send the right amount of data, which not only reduces data processing costs and security risks but also makes it easier for you to gain insights from your data down the line.

Finally, there are several important data security considerations to keep in mind as you think about your device management. For more information on device security, follow the guidance in another Microsoft paper, [**The Seven Properties of Highly Secure Devices.**](#)



IoT applications capture, store, analyze, and act on a huge volume and variety of data. How you choose to store, retain, and organize your data sets up a foundation with long-term implications.

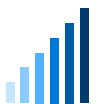
Begin by determining which kinds of data points will go through a hot, warm, or cold path:

- **Hot path:** Data is kept in-memory and analyzed in near-real-time, typically using a stream processing engine. The output may trigger an alert or be written to a structured format that can be queried immediately using analytical tools.
- **Warm path:** Recent data (e.g. the last day, week, or month) is kept in a storage service that can be queried immediately.
- **Cold path:** Historical data is kept in lower-cost storage to be queried in large batches

In terms of storage options, consider which storage services are best suited to different types of data. Each storage service is optimized for different transactional consistency and querying a specific type of data in its own unique syntax. For example, a specialized timeseries database is well suited to store a steady stream of fast-moving data coming from a sensor, while a NoSQL database is well suited for data that doesn't change frequently, like app and device configuration data (e.g. user roles, role assignments, device metadata, etc.).

It is also important to establish distinct retention policies for different types of data coming in. For example, telemetry data may be retained indefinitely, but user data must be deleted permanently if that user is removed. If your solution stores and handles user-identifiable data, it needs to be managed in compliance with data protection regulations like the General Data Protection Regulation (learn more at: [Microsoft.com/GDPR](https://www.microsoft.com/gdpr)).

Lastly, you'll need a strategy to ensure your data schema remains dynamic over time because it will evolve along with your IoT solution. For example, you may start out by capturing a limited number of data fields, such as device temperature, pressure, and humidity. If you want to add or delete fields in the future, your data system needs to be able to understand both your previous and current schema. This is a challenging undertaking but it's incredibly beneficial to remain flexible and enable your solution to evolve.



When people think of IoT insights, they often only think of real-time device health and performance monitoring. However, it's even more valuable to monitor device data in aggregate and over time, establish rules that automate actions based on device data, and set up integrations with your broader business systems. Evaluating device data on aggregate and over time can provide priceless insights. To accomplish this, you will need to build different kinds of dashboards and visualization tools for analyzing broad trends and drilling down into individual device telemetry. You may also want to leverage machine learning models to analyze historical data and predict when device maintenance is needed (a scenario known as predictive maintenance). While static analytics and insights are essential, it is even more powerful to automate rules based on IoT data rather than relying on individuals to identify and initiate actions.

For example, you could establish that your IoT solution sends you an email or SMS notification when there is an issue or forces a device to shut down when it's above a certain temperature. Your solution could also automatically create a maintenance ticket in a field service application. To make your rules and actions work correctly, you will need to combine the constant stream of telemetry data with other metadata in the cloud (such as CRM data that associates a device with a customer record).

Remember that your IoT solution is part of your broader business, and therefore it must be able to integrate with other systems. For example, you may need a solution like Microsoft Dynamics 365 to evaluate business results and present insights to decision-makers. You will also likely need to integrate your IoT solution with business applications like ERP, CRM, or industry-specific applications for tasks like user management, billing, service tickets, and customer support. When determining integration needs, keep in mind that you may need to use APIs to tailor user experiences for different roles, such as field technicians, equipment operators, business decision-makers, and other stakeholders.

5 High availability and disaster recovery



Ensure your mission-critical applications and data are built for resiliency and built correctly the first time. While each individual cloud service in your IoT solution architecture may deliver 99.9% availability or better, when you combine many services together you are compounding risks that will lower your solution's overall availability percentage. To counteract that, architect your solution with fault tolerance and disaster recovery in mind from day one. By doing so, you will be proactive in protecting yourself against bad code, intermittent connectivity failures, data corruption, and accidental deletion. If you start planning for failures after your solution is built, recovery processes will be much more difficult to retrofit.

There are three common types of resource failures to keep in mind: network, compute, and data.

- **Network failures** may happen within a cluster, within a region, or across regions. Consider DNS, Front Door, and load balancing techniques to recover.
- **Compute failures** may result in cloud services going down. You may need to write code to re-direct traffic to another service, and you need to consider stateless and stateful compute failover.
- **Data failures** may result in data loss or corruption. You need to have different disaster recovery systems in place for different types of data. For example, you need more resiliency for master data than you do for ephemeral or transitory data. Maintain reliability through geo-replicated storage and simplify your processes with automation.

When it comes to disaster recovery, you have a choice between two architectures. One option is an active/active system, also known as "always on" architecture. With "always on" architecture, multiple locations are processing the same data simultaneously, so if an outage occurs in one region traffic can be easily balanced across other regions. An active/active system will reduce downtime, but it is also more expensive and harder to implement. The second option is an active/passive system, in which data must be migrated to a backup, or "passive," location if one region goes down. This is easier to implement and less expensive but could result in significantly greater downtime. When choosing between the two architectures, consider the purpose of your IoT solution and how much downtime and expense your company or your customers can afford.

6 Security and compliance



Security and compliance measures are vital for every digital initiative — and especially in IoT scenarios. Because of the unique requirements and risks involved with IoT solutions, it's essential to establish an overarching strategy for maintaining security and ensuring regulatory compliance. Key security requirements include secure connectivity between devices and the cloud, secure integration with other applications, and secure data protection in the cloud. Approach each of these security components as layers that need to be accounted for.

Cloud service providers will provide multi-layered security across physical data centers and infrastructure. This is a good foundation but it's not sufficient. When you're combining individual cloud services, the solution as a whole is not secure simply because each individual service is secure. You are still responsible for ensuring data security across the entire IoT solution.

Below are best practices we recommend to support end-to-end security:

- Ensure that data is encrypted at rest (data stored on disk, services, databases, etc.) and in transit (calls between microservices and any other service calls).
- Use Transport Security Layer (TLS) 1.2+ – a cryptographic protocol designed to provide communication security over a network.
- Control what actions users can take and what data they can see using role-based access control (RBAC)
- Secure your devices using the guidance in [The Seven Properties of Highly Secure Devices](#), which takes a deeper look at considerations specific to device security.
- Ensure you have an audit trail across each IoT cloud service to track user activity. For example, you should be able to trace unauthorized access and determine who updated a device and when.
- For multi-tenant solutions, customize your cloud services with an app security layer to ensure that each organization can only access its own data.
- Conduct the following activities on a regular basis: security audits, drills for hypothetical disaster recovery and security threat situations, threat modeling to find vulnerabilities in code or design, and penetration testing by a hired third party.

Having end-to-end security helps ensure regulatory compliance, but compliance doesn't end there. Many companies don't realize that initial compliance or certifications quickly become obsolete. It's important to plan ahead for annual or bi-annual assessments to maintain compliance with accessibility requirements, data sovereignty laws, data privacy requirements like GDPR, and certifications such as ISO, HIPPA and FedRAMP. If you are making SLA guarantees to your customers, implement processes to ensure you are meeting those guarantees and keeping them up to date.



Managing IoT solutions with DevOps



To manage IoT solutions effectively, it's important to have a healthy DevOps pipeline. DevOps is a set of practices that combines software development (dev) and IT operations (ops) in order to build, operate, and maintain cloud services.

DevOps processes for IoT projects are complex. Moving code safely from a developer's desk into production without interruption to customers is already a daunting challenge for many organizations. IoT is particularly challenging due to the variety and volume of devices and cloud services involved, some of which may be outsourced to vendors (e.g. embedded device hardware, firmware, software, internet connectivity, cloud services, mobile applications, and big data services).

DevOps practices provide overarching governance for planning, testing, deploying, documenting, and managing the different components of your solution in a secure, coordinated way. Below we'll outline key processes that most IoT deployments require, but keep in mind that these are just a sample of the processes that might be needed, and all of them require some cloud development expertise.

- **Manage, provision, and deploy code:**
 - Source code management (e.g. GitHub, etc.)
 - Automated code deployments through continuous integration (CI) and continuous deployment (CD)
 - Automated provisioning of cloud resources (e.g. provisioning a new test environment)
 - Handling multiple versions of your IoT service
 - Environment management – development, integration testing, scale testing, pre-production, production
- **Prevent and manage security issues:**
 - Code scanning for content / virus, digital signing
 - Scanning for known software (including open source) vulnerabilities
 - Secret management (e.g. cluster keys, database access keys, certificates in KeyVault)
 - Certificate management and auto-rotation
 - Audit trail of DevOps user actions
 - Secure access to production and customer data
- **Test your solution before product deployment:**
 - Functional testing: Test different versions of device firmware that works in conjunction with the IoT solution in the cloud
 - Stress testing: Push the solution to its known breaking limits to identify bottlenecks and weakness
 - Long haul testing: Run the solution at production scale for a long period of time to accumulate more data and realistic device call patterns
 - Chaos testing: Simulate failures by turning off certain services randomly and measure the resiliency of the solution

- **Manage information related to customer support:**

- Service-level agreement (SLA) and service-level objective (SLO) management: you will need to track many metrics that roll up to your SLAs and SLOs
- Billing and metering
- Service incident management
 - Monitoring of LiveSite and alerting when something does not go well as planned
 - 24x7 Livesite DRI (Designated Responsible Individual) who is on call to address issues

8

Understanding total cost of ownership (TCO)



Many IoT solutions fail because of the difficulty of estimating long-term aggregated costs of various cloud services. It's critical to understand how much you're spending to run and operate all services involved. Without this understanding, it's common for budgets to run into the red, leaving you with ballooning costs.

When evaluating infrastructure costs, account for the basics first: storage, compute and network. Beyond that, you must account for all services needed to ingest, egress, and prepare data to be used in business decisions. Make sure to estimate costs based on the architecture of the solution when it is running at scale, not your POC architecture. Architecture and costs will evolve rapidly after the POC. Furthermore, do not overlook long-term operational costs that will increase in parallel with infrastructure costs, such as employing technicians to operate the solution, vendors to manage non-public clouds, and customer support teams.

Your costs will depend greatly on the "chattiness" of your devices and size of messages devices send. Some devices are "chatty," sending many messages to the cloud every minute, while others are relatively quiet, only needing to send data every hour or more. Getting clear on device chattiness and message size helps reduce the likelihood of over-provisioning, which leads to unused cloud capacity, or under-provisioning, which leads to elastic scale challenges. As you plan your IoT solution, consider carefully the size and frequency of the message payloads to ensure your infrastructure is "right-sized" for where you are today and ready to scale with you.

Determining your IoT solution approach



In terms of approach, you have many options for building your IoT solution. If you need a high degree of control and customization, platform-as-a-service (PaaS) cloud services like those from Azure are available for you to combine in a way that meets your unique needs. Typical PaaS services found in an Azure IoT Solution include:

- [Azure IoT Hub](#) for device connectivity and management
- [Azure IoT Hub](#) Device Provisioning Service for automated device provisioning to IoT Hub in your solution at scale
- [Azure Time Series Insights](#) for storing and analyzing warm and cold path time series data from IoT devices
- [Azure Stream Analytics](#) for analyzing hot path data from IoT devices
- [Azure IoT Edge](#) for running AI, 3rd party service or your business logic on the edge devices

We provide a [reference architecture](#) that covers the best practices for assembling these services into an IoT solution. This approach generally requires a high level of investment and cloud expertise. Alternatively, you can use a hosted platform that streamlines the complexities of building an IoT solution by handling all of the key considerations mentioned above.

For example, [Azure IoT Central](#) is a hosted, extensible, fully managed app development platform that reduces the burden and cost associated with managing the IoT solution attributes covered in this paper. It is built on the same Azure PaaS services available to everyone, but it takes the guesswork and complexity out of building reliable, scalable and secure IoT applications. The easy-to-use interface makes it simple to monitor device conditions, create rules, and manage millions of devices and their data remotely throughout their lifecycle. Furthermore, it enables you to take action on device insights by extending IoT intelligence into line of business applications. Azure IoT Central offers built-in disaster recovery, multi-tenancy, global availability, and a predictable cost structure. Choosing to build with Azure IoT Central gives you the opportunity to focus time and money on transforming your business and designing innovative offerings, rather than maintaining and updating a complex and continually evolving IoT infrastructure.



Get started today

With these 8 attributes in mind, discuss next steps with your team and evaluate what approach makes the most sense for your IoT solution.

To learn more, explore these resources

[Azure IoT Central](#)

[Azure IoT](#)

[IoT Signals report](#)

2019 Microsoft. All rights reserved. This white paper is for informational purposes only. MICROSOFT MAKES NO WARRANTIES, EXPRESS OR IMPLIED, IN THIS DOCUMENT.

This document is provided "as is." Information and views expressed in this document, including URL and other Internet website references, may change without notice. You bear the risk of using it. This document does not provide you with any legal rights to any intellectual property in any Microsoft product. You may copy and use this document for your internal, reference purposes.