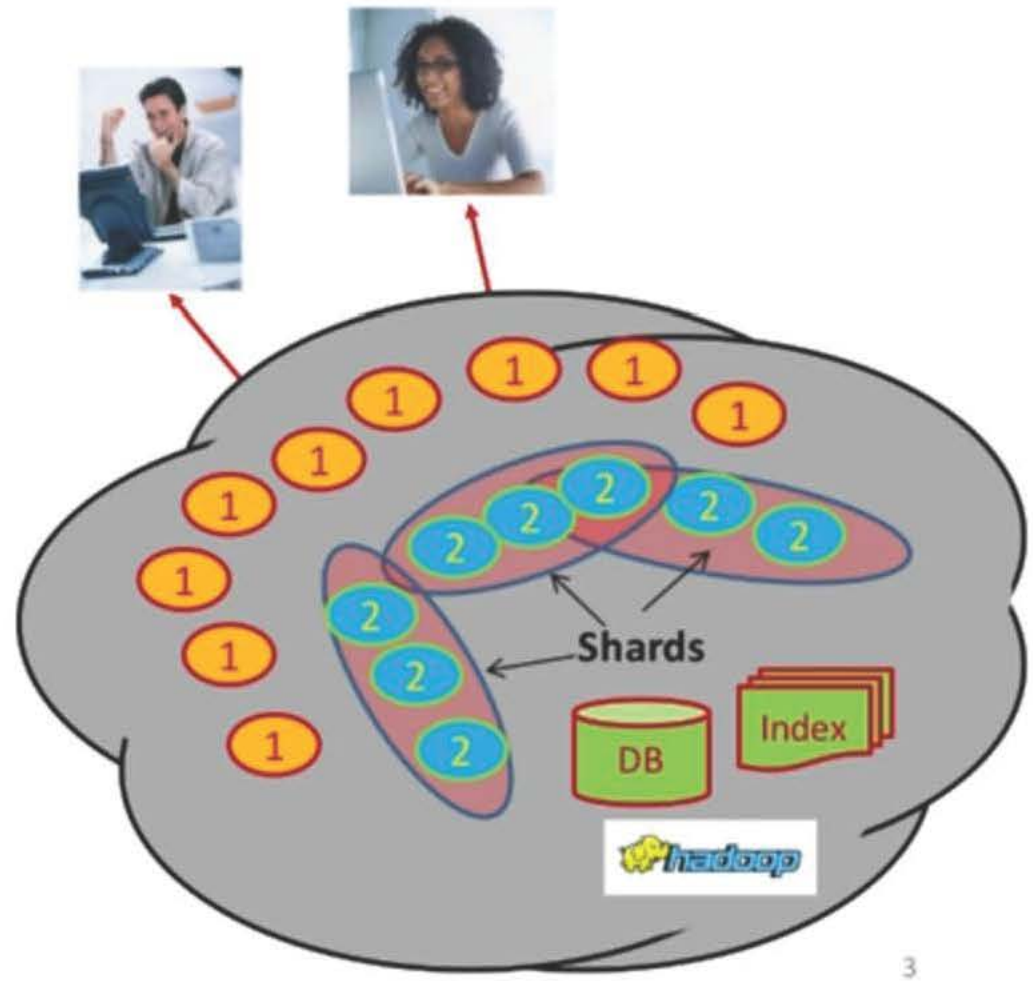
IoT: Cloud Services

Cloud Edge Services

IoT Apps talking to Cloud

- The modern cloud is comprised by a number of services called micro-services
  - many times in a multi-tiered architecture
- Client devices (IoT sensors, smartphones, etc) talk to the Edge of the Cloud
  - The **edge services** are simple, lightweight, and nimble



IoT Cloud Edge

- Cloud services for IoT must provide per-device authentication and access control.

- Cloud Edge must route messages to a microservice endpoint based on message properties.

- We will talk later about micro services and IoT messages.

- Routing rules give the flexibility to send messages where they need to go without the need to stand up additional services to process messages or to write additional code.

Edge services

- Near the edge of the cloud focus is on serving a vast number of clients in the fastest possible way
  - Caching content at different layers helps
  - Stateless

- Inside we find high volume services that operate in a pipelined manner, asynchronously

- Deep inside the cloud we see a world of virtual computer clusters that are scheduled to share resources and on which applications like MapReduce/Hadoop/Spark are very popular
  - Applications: A/B testing

- In the bottom of the Cloud is where the data are being stored

IoT Systems and Edge Services

- An IoT ecosystem is a complex array of intertwined systems that work together to seamlessly provide to the customers a great experience.

- The Cloud APIs is the front door to that system
  - Needs to support thousands of different devices
  - Handle multiple thousands of requests per second during peak hours.

Cloud Edge evolution

- The Cloud API is the way the IoT services interact with the cloud. Hence background upgrades/updates should not affect the communication.
- This can be achieved with a common well-defined and extensible Cloud API
- A robust edge service must enable
  - rapid development
  - great flexibility
  - expansive insights
  - resiliency

Cloud Edge Features to support IoT

- Authentication
- Insights
- Stress Testing
- Canary Testing
- Dynamic Routing
- Load Shedding
- Security
- Static Response handling
- Multi-Region Resiliency

Cloud Edge Insights

- A service that allows us to shed and prioritize traffic when issues occur.

- Detailed information into network performance and errors, as well as handles software load balancing for even load distribution.

- Fine-grained metrics in real-time so that we can quickly observe and react to problems.

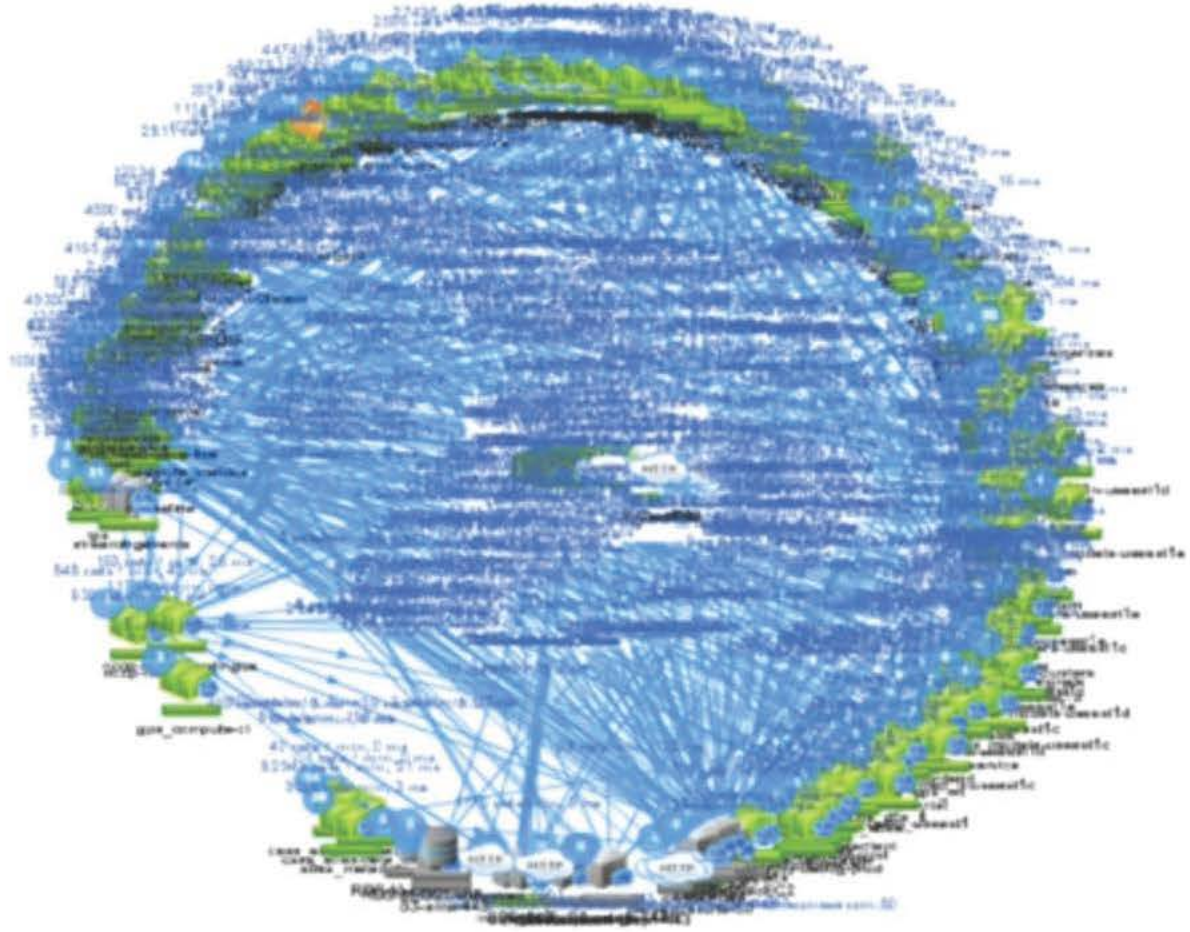
- Dynamically change properties

IoT: Cloud Services

IoT pushing the limits of SOA to microservices

IoT and Microservices

- IoT is gaining a lot of attention and the Cloud platform has more requirements.
- Big Data became commonplace and the world started moving towards the API economy.



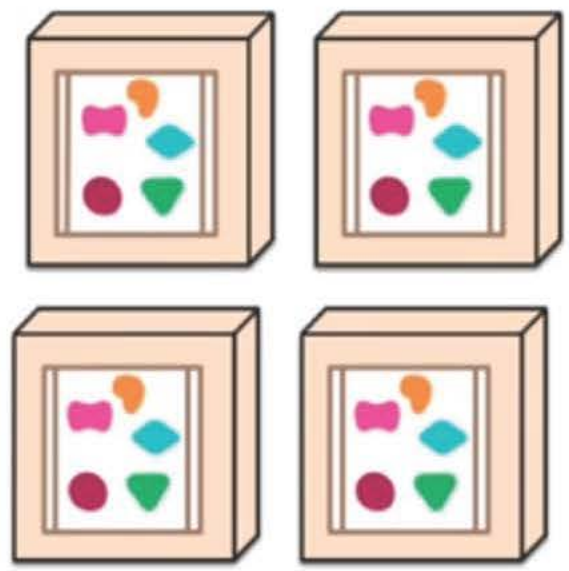
Classic SOA

- This is where the Classic SOA started showing problems.
    - too complicated, with hundreds of interfaces and impossible to define granularity.
- The Microservices architecture adds agility to SOA and brings the much needed speed and efficiency when it comes to deployment and modification of systems.
- Microservices can define the size of a service and the manner in which they talk to other services.
- Microservice architecture brings in smaller services and lightweight protocols.
- The principle of the Microservices architecture is akin to the Unix principle “Do one thing and do it well”.

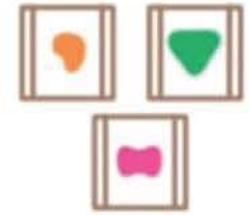
Microservice style vs Monolyth

A monolithic application puts all its functionality into a single process...

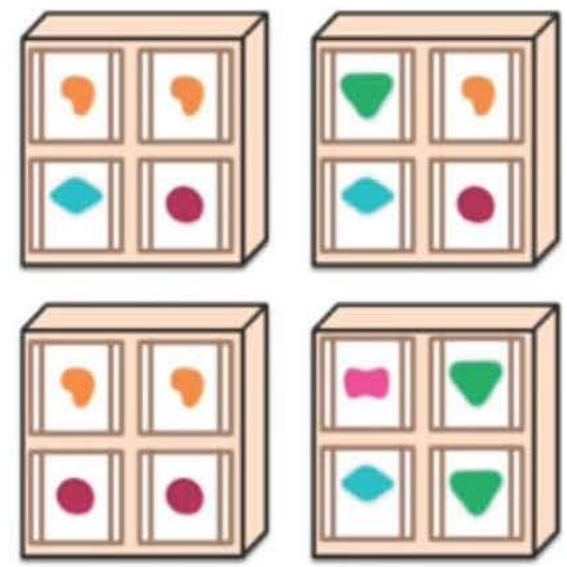
... and scales by replicating the monolith on multiple servers



A microservices architecture puts each element of functionality into a separate service...

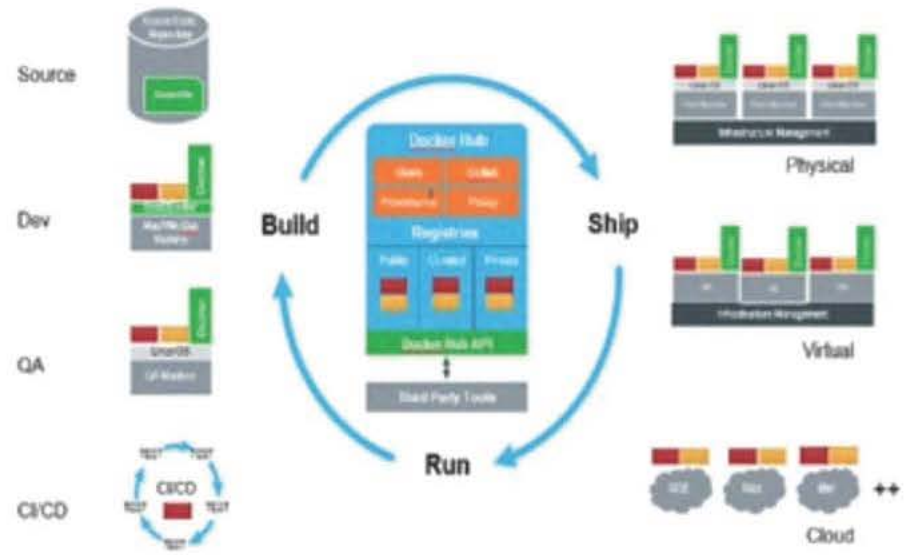


... and scales by distributing these services across servers, replicating as needed.

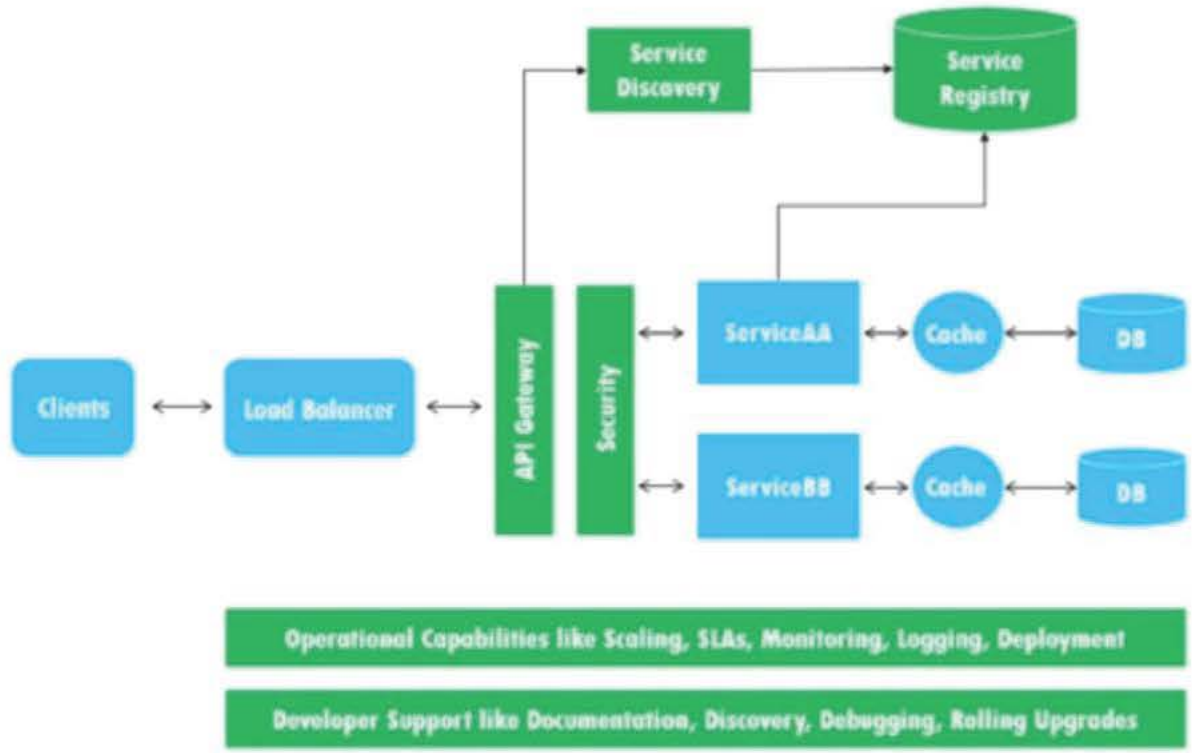


IoT & Microservices

- **Microservice architectural style:** developing a single application as a suite of small services, each running in its own process and communicating with lightweight mechanisms, e.g. REST API.
- Microservices also offer a way of scaling the infrastructure both horizontally and vertically giving long term benefits to the IoT deployments. Each of the services can scale based on the needs.
- Given the dynamism of deployment and scalability expectations which comes with IoT, Microservices need to become an important part of the overall IoT Strategy.



Microservices Challenges



- **Distributed application logic:** with microservices the logic is spread across the services and, more importantly, embedded in the control and data flow between those services.
- **Diverse technology stack**: the system may be comprised of in house developed services, open source software, and external libraries.
- **Hard to test and debug**: there might be thousands of interactions between the constituent services
  - Equivalent to the “butterfly effect” (a minor change of service, could potentially be catastrophic)

IoT: Cloud Services

Device to Cloud (D2C)

How do IoT communication with Cloud

- Core properties of an IoT messaging functionality are the reliability and durability of messages.
    - These properties enable resilience
        - intermittent connectivity on the device side
        - load spikes in event processing on the cloud side.
- IoT Cloud app must exposes the messages/events built-in endpoint for your back-end services to read the device-to-cloud received messages.

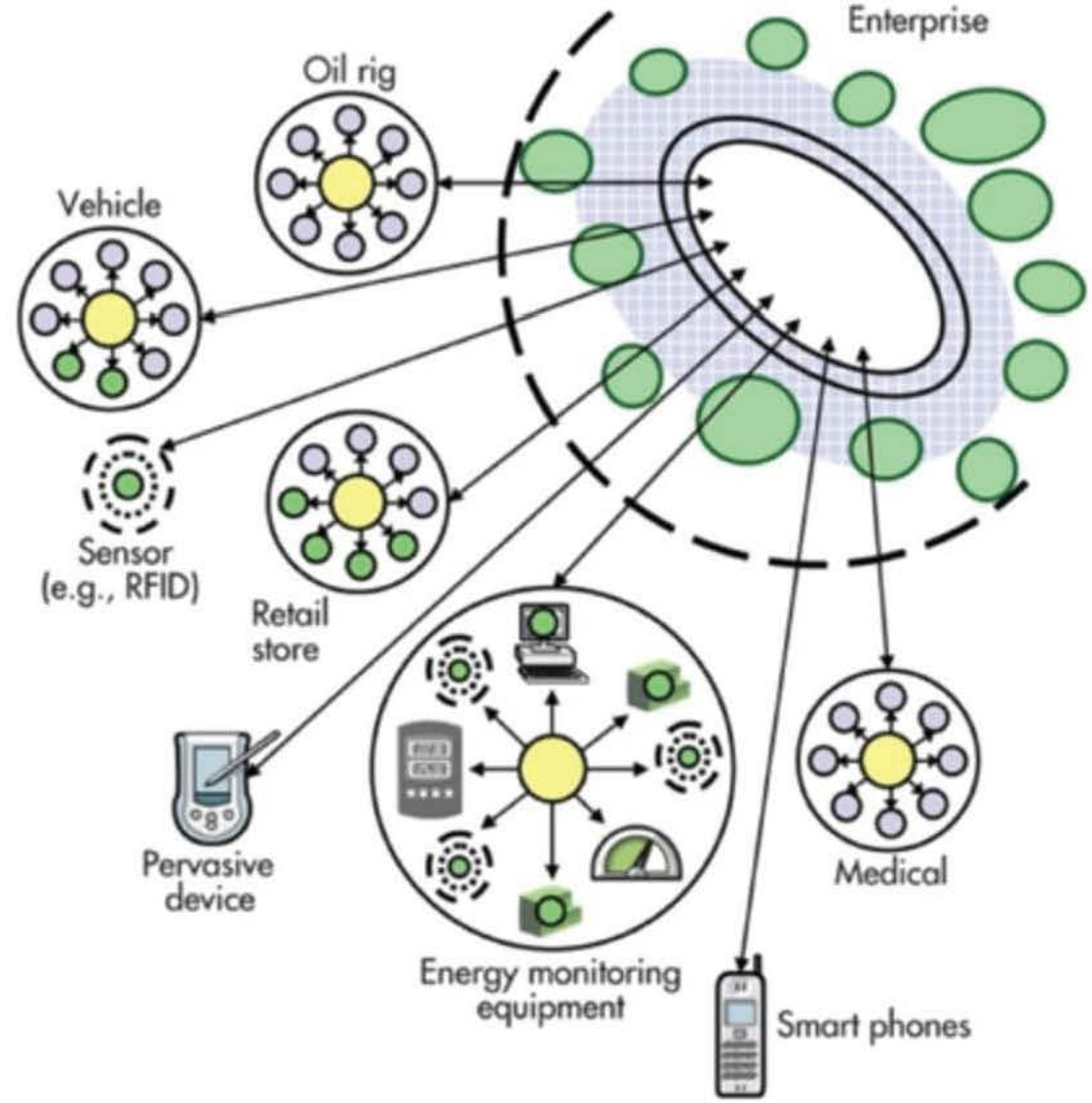
Device facing protocols

- MQTT
- AMPQ
- XMPP
- HTTP

MQTT

- MQ Telemetry Transport (MQTT) from IBM is the bad boy of small device messaging.
    - It uses TCP, is lightweight, and has features beneficial to IoT devices.
    - It is mature and there are a lot of client and server implementations, making it easier to develop upon.

- MQTT works on a pub/sub architecture. A client subscribes to a channel on a server, and when a server receives new information for that channel, it pushes it out to that device.



MQTT (cnt'd)

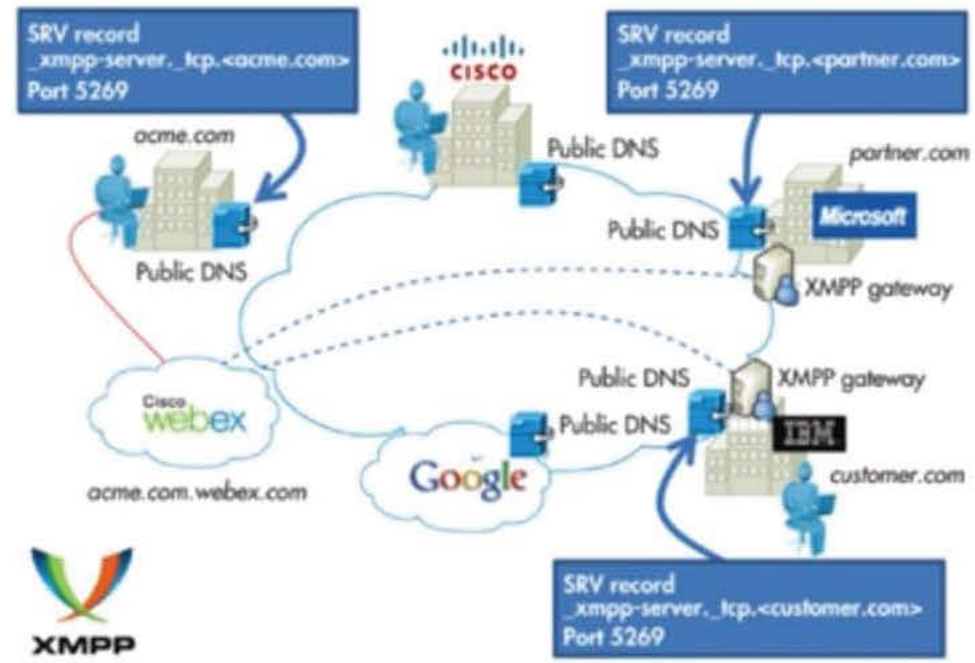
- Another great feature of MQTT is that you can set a priority or Quality of Service (QoS):
  - 0: The client/server will deliver the message once, with no confirmation required.
  - 1: The client/server will deliver the message at least once, confirmation required.
  - 2: The client/server will deliver the message exactly once by using a handshake process.
- Each level uses more bandwidth but will give you varying assurance of deliverability.
- With MQTT your application must have a library to talk to the MQTT server and handle publish/subscribe methods.

XMPP

- XMPP: a protocol best for connecting devices to people, a special case of the D2C pattern, since people are connected to the servers.

- XMPP was originally called “Jabber.” It was developed for instant messaging (IM) to connect people to other people via text messages.

- XMPP stands for Extensible Messaging and Presence Protocol. Again, the name belies the targeted use: presence, meaning people are intimately involved.

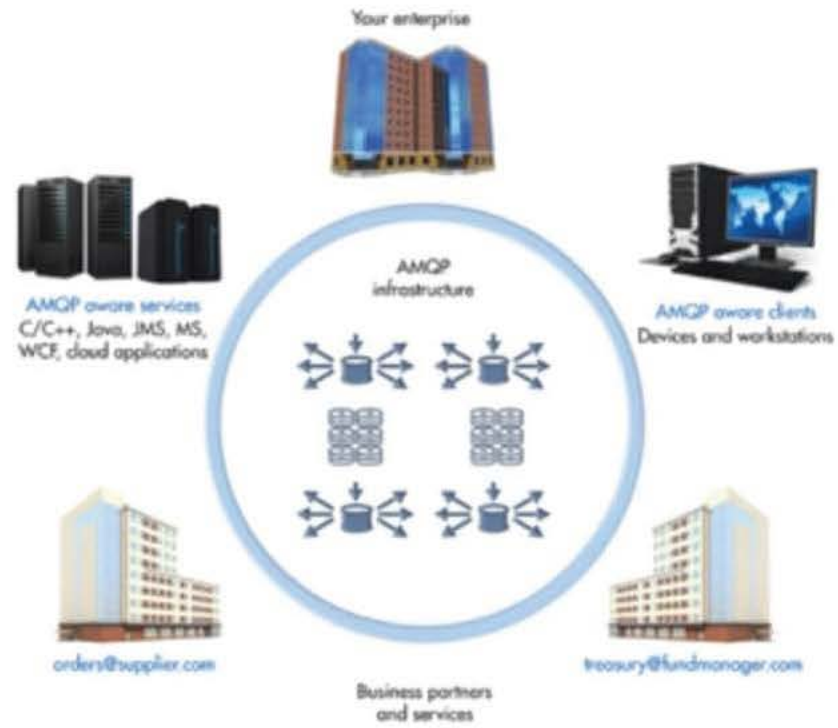


XMPP (cnt'd)

- XMPP uses the XML text format as its native type, making person-to-person communications natural.
    - Like MQTT, it runs over TCP, or perhaps over HTTP on top of TCP.
    - Its key strength is a name@domain.com addressing scheme that helps connect the needles in the huge Internet haystack.
- IoT world:
    - XMPP offers an easy way to address a device. This is especially handy if that data is going between distant, mostly unrelated points, just like the person-to-person case.
    - It's not designed to be fast. In fact, most implementations use polling, or checking for updates only on demand.
    - A protocol called BOSH (Bidirectional streams over Synchronous HTTP) lets severs push messages. But "real time" to XMPP is on human scales, measured in seconds.

AMPQ

- Advanced Message Queuing Protocol (AMQP) is sometimes considered an IoT protocol.
- It sends transactional messages between cloud services.
- As a message-centric middleware that arose from the banking industry, it can process thousands of reliable queued transactions.



AMPQ (cnt'd)

- AMQP is focused on not losing messages.
  - Communications from the publishers to exchanges and from queues to subscribers use TCP, which provides strictly reliable point-to-point connection.
  - Further, endpoints must acknowledge acceptance of each message.
    - The standard also describes an optional transaction mode with a formal multiphase commit sequence.
    - True to its origins in the banking industry, AMQP middleware focuses on tracking all messages and ensuring each is delivered as intended, regardless of failures or reboots.
- AMQP is mostly used in business messaging. It usually defines "devices" as mobile handsets communicating with back-office data centers. In the IoT context, AMQP is most appropriate for the control plane or server-based analysis functions.

IoT: Cloud Services

Device to Cloud (D2C) - part #2

WebSockets

- Now that browser websockets are standardized, they make a very compelling option for reading and writing data to IoT devices.

- Websockets are the best way to achieve full push/pull communications to a web server, which is not possible over basic HTTP protocol.

- Websockets are great if you have a full web client! However, with many IoT devices, it is a lot of overhead which might not even be an option.

- Another downside of using Websockets is that you would need to come up with your own protocol for the transmission of data. This isn't too difficult (for example, using JSON, XML) but does create a non-standard protocol.

Protocol over WebSocket

- A frequent use of Websockets is to actually use MQTT/XMPP/AMPQ over Websockets.
- WebSocket defines how can you raise an HTTP connection into a bidirectional channel,
    - the problem WebSocket want to resolve is HTTP is unidirectional and it dominated the web.
- In a higher layer, MQTT/XMPP/AMPQ defined how several endpoint exchange data when they can talk with the same broker.
- Metaphor:
    - MQTT works like FedEx, you sign addresses, it deliver your express.
    - WebSocket works like road if TCP is the roadbed, it enables you transport your express from starting point to the endpoint.
    - FedEx walks on the road, MQTT works on WebSocket, TCP, SCTP also.
    - So if we build a real-time IoT system, the probably architecture is we use MQTT over TCP in the connected devices, and we use MQTT over WebSocket in the web pages.

IoT protocols

- The IoT needs many protocols.
- The ones outlined here differ markedly. Perhaps it's easiest to categorize them along a few key dimensions:
  - QoS
  - Addressing
  - Application

QoS of the transport protocol

- Quality of Service (QoS) control is a much better metric than the overloaded “real-time” term.
- QoS control refers to the flexibility of data delivery.
- A system with complex QoS control may be harder to understand and program, but it can build much more demanding applications.
- Example:
  - Consider the reliability QoS. Most protocols run on top of TCP, which delivers strict, simple reliability.
  - Every byte put into the pipe must be delivered to the other end, even if it takes many retries.
  - This is simple and handles many common cases, but it doesn’t allow timing control. TCP’s single-lane traffic backs up if there’s a slow consumer.

Discoverability

- Finding the data needle in the huge IoT haystack is a fundamental challenge.
- XMPP shines here for “single item” discovery.
  - Its “user@domain” addressing leverages the Internet’s well-established conventions.
  - XMPP doesn’t easily handle large data sets connected to one server.
- With its collection-to-a-server design, MQTT handles that case well. If you can connect to the server, you’re on the network.
- AMQP queues act similarly to Cloud services, but for C2C microservices.

Intended Applications

- Inter-device data use is a fundamentally different use case from device data collection.

- Example: For example, turning on your light switch (best for XMPP) is worlds apart from generating that power, monitoring the transmission lines (MQTT), or analyzing the power usage back at the data center (AMQP).

IoT: Cloud Services

D2C and C2D

Considerations

- Consider the following points when you choose your protocol for device-side communications:
    - HTTP does not have an efficient way to implement server push.
    - As such, when you are using HTTP, devices poll the cloud for cloud-to-device messages. This approach is inefficient for both the device and the Cloud Edge.
    - Under current HTTP guidelines, each device should poll for messages every 25 minutes or more.
    - On the other hand, MQTT and AMQP support server push when receiving cloud-to-device messages.
        - They enable immediate pushes of messages from IoT Hub to the device. If delivery latency is a concern, MQTT or AMQP are the best protocols to use.

Field Gateways

- In an IoT solution, a field gateway sits between your devices and your IoT Hub endpoints.
- It is typically located close to your devices.
- The IoT devices communicate directly with the field gateway by using a protocol supported by the devices.
- The field gateway connects to an IoT Cloud endpoint using a protocol that is supported by the Cloud Edge.
- A field gateway can be highly specialized hardware or a low-power computer running software that accomplishes the end-to-end scenario for which the gateway is intended.
- Other benefits: ability to multiplex the communication from multiple devices onto the same Cloud connection.

MQTT with Field Gateways

- When using MQTT and HTTP, you cannot connect multiple devices (each with its own per-device credentials) using the same TLS connection.

- Thus, for Field gateway scenarios, these protocols are suboptimal because they require one TLS connection between the field gateway and Cloud Edge for each device connected to the field gateway.

Low Resource Devices

- The MQTT and HTTP libraries have a smaller footprint than the AMQP libraries.

- As such, if the device has limited resources (for example, less than 1 MB RAM), these protocols might be the only protocol implementation available.

Network Traversal

- The standard AMQP protocol uses port 5671, while MQTT listens on port 8883, which could cause problems in networks that are closed to non-HTTP protocols.

- MQTT over WebSockets, AMQP over WebSockets, and HTTP are available to be used in this scenario.

Payload Size

- MQTT and AMQP are binary protocols, which result in more compact payloads than HTTP.

Cloud to Device Communication

- **Direct methods** for communications that require immediate confirmation of the result.
    - Direct methods are often used for interactive control of devices such as turning on a fan.
- **Twin's** desired properties for long-running commands intended to put the device into a certain desired state.
    - For example, set the telemetry send interval to 30 minutes.
- **Cloud-to-device messages** for one-way notifications to the device app.

Examples

	direct methods	Twin's desired properties	cloud to device messages
Scenario	Commands that require immediate confirmation, such as turning on a fan.	Long-running commands intended to put the device into a certain desired state. For example, set the telemetry send interval to 30 minutes.	One-way notifications to the device app.
Data flow	Two-way. The device app can respond to the method right away. The solution back end receives the outcome contextually to the request.	One-way. The device app receives a notification with the property change.	One-way. The device app receives the message
Durability	Disconnected devices are not contacted. The solution back end is notified that the device is not connected.	Property values are preserved in the device twin. Device will read it at next reconnection.	Messages can be retained by Cloud provider

IoT: Cloud Services

Cloud security

Introduction

- Cloud is a massive concentration of resources
- Also a massive concentration of risk expected loss from a single breach
- It can be significantly larger concentration of "users" represents a concentration of threats
  - "*Ultimately, you can outsource responsibility but you can't outsource accountability.*" From [2] John McDermott, ACSAC 09

Cloud Security

Security is one of the most difficult task to implement in cloud computing.

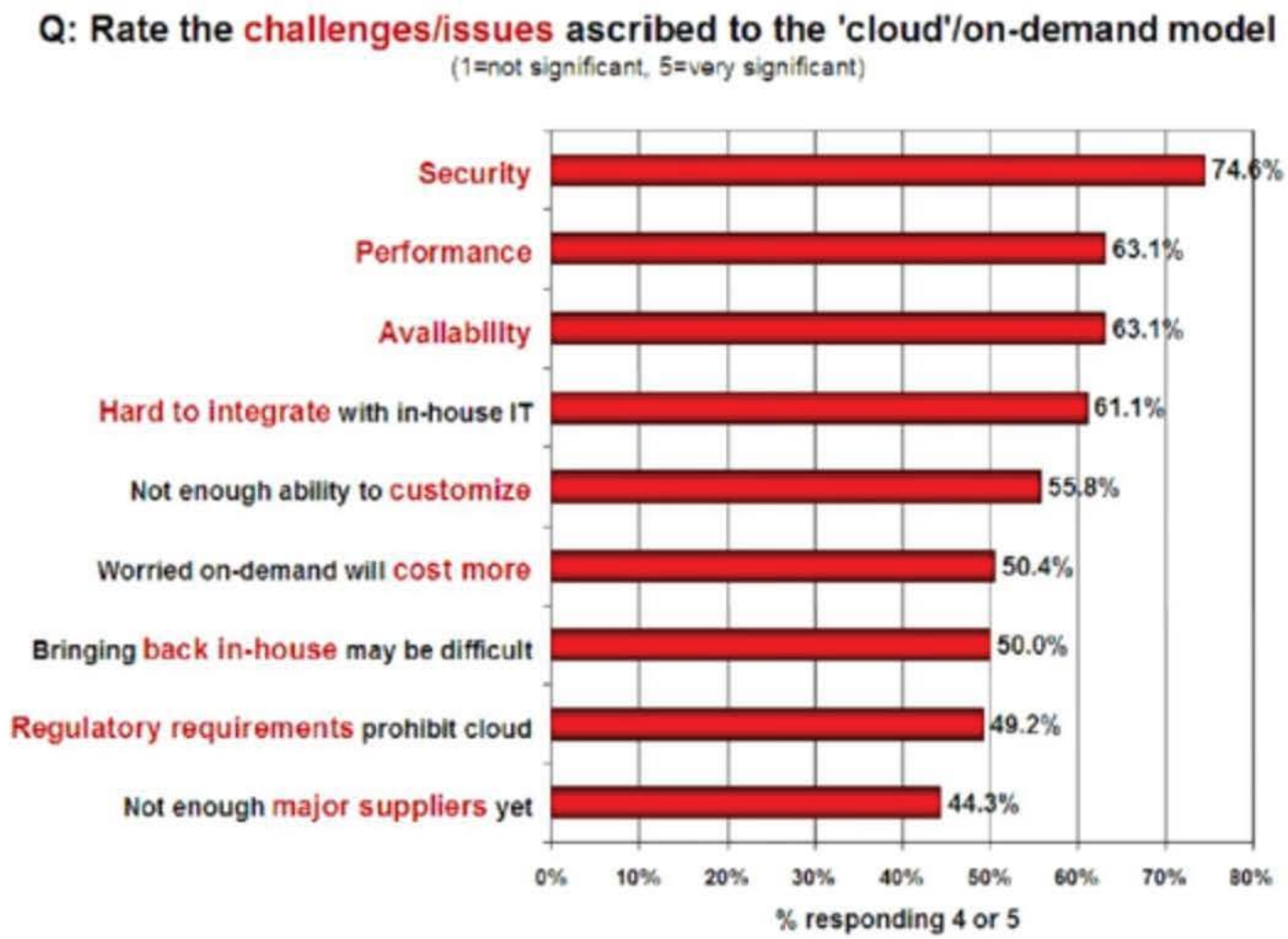
- Different forms of attacks in the application side and in the hardware components

Attacks with catastrophic effects only needs one security flaw

Why is everybody not moving to the Cloud?

- The cloud acts as a big black box, nothing inside the cloud is visible to the clients
- Clients have no idea or control over what happens inside a cloud
- Even if the cloud provider is honest, it can have malicious system admins who can tamper with the VMs and violate confidentiality and integrity
- Clouds are still subject to traditional data confidentiality, integrity, availability, and privacy issues, plus some additional attacks

Security is a paramount challenge



Source: IDC Enterprise Panel, August 2008 n=244

IoT: Cloud Services

Security Problems

Cause of Cloud security problems

- Most security problems stem from:
  - Loss of control
  - Lack of trust (mechanisms)
  - Multi-tenancy
- These problems exist mainly in 3rd party management models
  - Self-managed clouds still have security issues, but not related to above

Loss of Control

- Consumer's loss of control
  - Data, applications, resources are located with provider
  - User identity management is handled by the cloud
  - User access control rules, security policies and enforcement are managed by the cloud provider
  - Consumer relies on provider to ensure
    - Data security and privacy
    - Resource availability
    - Monitoring and repairing of services/resources

Lack of Trust

- **Trusting a third party requires taking risks**
- Defining trust and risk
  - Opposite sides of the same coin (J. Camp)
  - People only trust when it pays (Economist's view)
  - Need for trust arises only in risky situations
- Defunct third party management schemes
  - Hard to balance trust and risk
  - e.g. Key Escrow (Clipper chip)
  - Is the cloud headed towards the same path?

Lack of Trust

- What if the Cloud provider has a conflicting business?
  - For example Amazon sells products, streams movies and music, offers an online grocery and many others
  - How do companies like Netflix, Lyft etc use AWS although they have a competitive product?
  - How does Amazon, as a Cloud provider benefit from having these companies run their infrastructure on AWS?

Multi-Tenancy

- Conflict between tenants' opposing goals
  - Tenants share a pool of resources and have opposing goals
- How does multi-tenancy deal with conflict of interest?
  - Can tenants get along together and 'play nicely' ?
  - If they can't, can we isolate them?
- How to provide separation between tenants?

Multi-Tenancy

- Cloud Computing brings new threats
- Multiple independent users share the same physical infrastructure
- Thus an attacker can legitimately be in the same physical machine as the target

Core Issue

- The core issue here is the levels of trust
  - Many cloud computing providers trust their customers
  - Each customer is physically commingling its data with data from anybody else using the cloud while logically and virtually you have your own space
  - The way that the cloud provider implements security is typically focused on they fact that those outside of their cloud are evil, and those inside are good.
- But what if those inside are also evil?

IoT: Cloud Services

Taxonomy of Fear

Confidentiality & Integrity

- Confidentiality
  - Fear of loss of control over data
    - Will the sensitive data stored on a cloud remain confidential?
    - Will cloud compromises leak confidential client data
  - Will the cloud provider itself be honest and won't peek into the data?
- Integrity
  - How do I know that the cloud provider is doing the computations correctly?
  - How do I ensure that the cloud provider really stored my data without tampering with it?

From [5] www.cs.jhu.edu/~ragib/sp10/cs412

Availability

- Will critical systems go down at the client, if the provider is attacked in a Denial of Service attack?
- What happens if cloud provider goes out of business?
- Would cloud scale well-enough?
- Often-voiced concern
  - Although cloud providers argue their downtime compares well with cloud user's own data centers

More fears…

- Privacy issues raised via massive data mining
  - Cloud now stores data from a lot of clients, and can run data mining algorithms to get large amounts of information on clients
- Increased attack surface
  - Entity outside the organization now stores and computes data, and so
  - Attackers can now target the communication link between cloud provider and client
  - Cloud provider employees can be phished

Auditability and Forensics

- Auditability and forensics (out of control of data)
  - Difficult to audit data held outside organization in a cloud
  - Forensics also made difficult since now clients don't maintain data locally
- Legal quagmire and transitive trust issues
  - Who is responsible for complying with regulations?
    - e.g., SOX, HIPAA, GLBA ?
  - If cloud provider subcontracts to third party clouds, will the data still be secure?

IoT: Cloud Services

Threat Model

What is a threat model?

- A threat model helps in analyzing a security problem, design mitigation strategies, and evaluate solutions
- Steps:
  - Identify attackers, assets, threats and other components
  - Rank the threats
  - Choose mitigation strategies
  - Build solutions based on the strategies

Why thread model is useful?

- Threat modeling is useful in any software context, but is particularly valuable in cloud computing due to the widespread preoccupation with security.
- It's also useful because technical and non-technical people alike can follow the diagrams easily.

Threat Model components

- Basic components
  - Attacker modeling
    - Choose what attacker to consider
      - insider vs. outsider?
      - single vs. collaborator?
    - Attacker motivation and capabilities
  - Attacker goals
  - Vulnerabilities / threats

Malicious Insider

- At client
  - Learn passwords/authentication information
  - Gain control of the VMs
- At cloud provider
  - Log client communication
  - Can read unencrypted data
  - Can possibly peek into VMs, or make copies of VMs
  - Can monitor network communication, application patterns
  - Why?
    - Gain information about client data
    - Gain information on client behavior
    - Sell the information or use itself

Outside Attacker

- What?
  - Listen to network traffic (passive)
  - Insert malicious traffic (active)
  - Probe cloud structure (active)
  - Launch DoS
- Goal?
  - Intrusion
  - Network analysis
  - Man in the middle
  - Cartography

Thread modeling in Cloud Computing

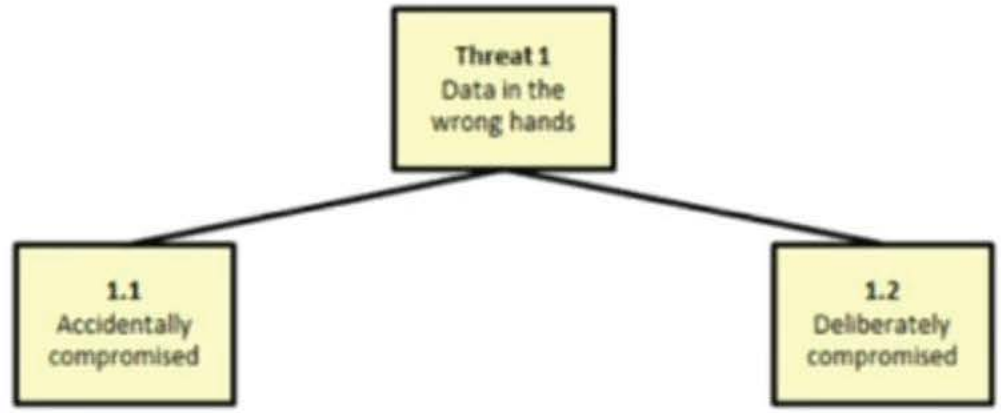
- A common concern is that the use of shared resources in the cloud might compromise the security of your data by allowing it to fall into the wrong hands—what we call *Data Isolation Failure*.

- A data isolation failure is one of the primary risks organizations considering cloud computing worry about.

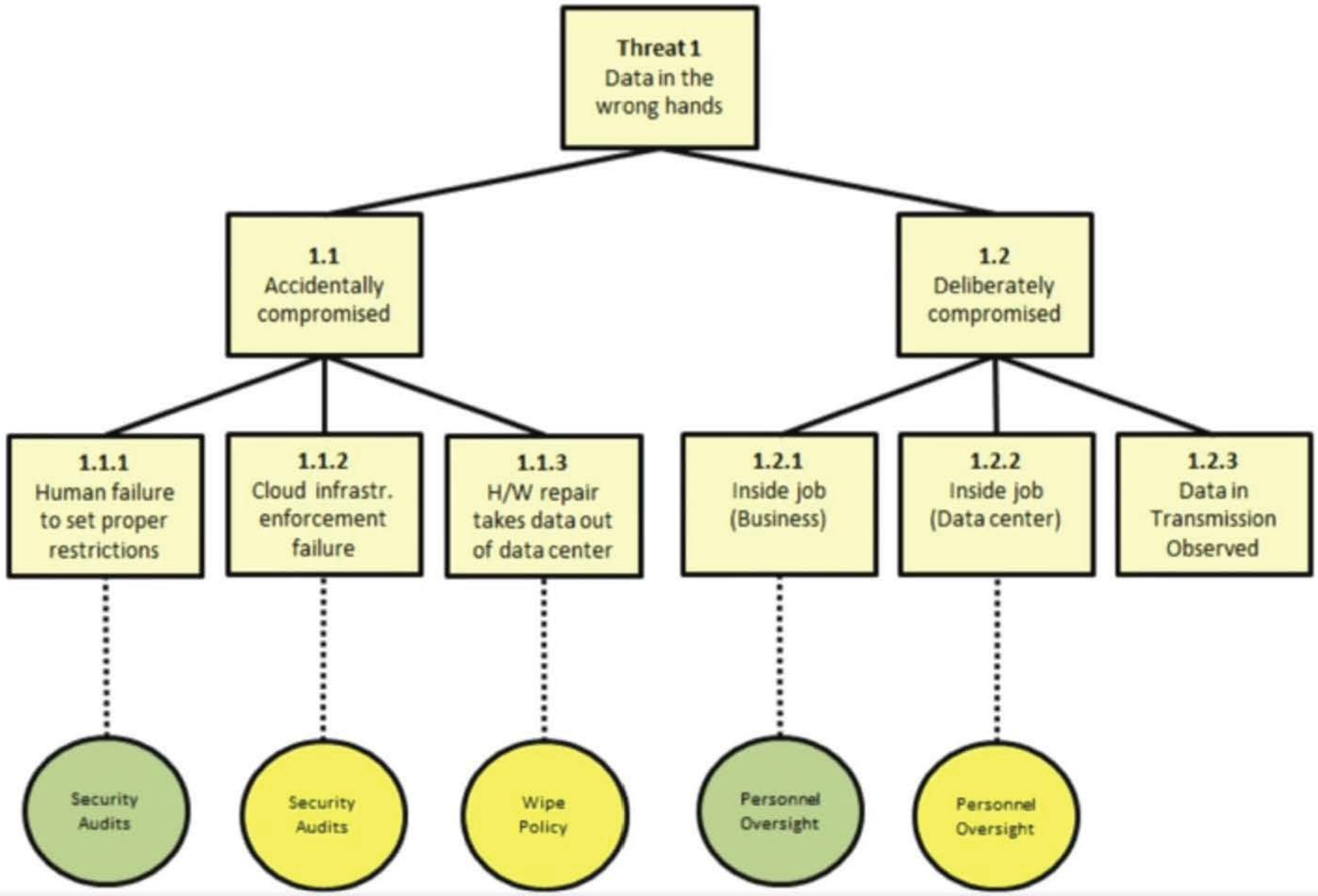
- To create our threat model, we'll start with the end result we're trying to avoid: data in the wrong hands.

What can lead to this result?

Either one of these conditions is sufficient to cause data to be in the wrong hands, so this is an OR condition. We'll see later on how to show an AND condition.



Threat Modeling and Mitigations



Challenges for Attacker

- How to find out where the target is located?
- How to be co-located with the target in the same (physical) machine?
- How to gather information about the target?