Data Architecture Basics:
An Illustrated Guide for Non-Technical Readers

A WHITE PAPER BY DATAIKU

www.dataiku.com

INTRODUCTION

WHY DATA ARCHITECTURE MATTERS

Data architecture is the foundation of every organization's data strategy. Just like in a house or office, the structure determines whether the inhabitants are able to perform their tasks. Hopefully, it's with efficiency and joy; but when it comes to data architecture, poor use of space, inconvenient pathways, security, and clutter can make life difficult.

Designing data architecture is a valuable skill, but it's not something that everyone is equipped with, and that's OK - not everyone needs to be an expert. But data architecture is not just something for CIOs and data architects either; everyone at data-powered organizations can benefit from understanding the ways data moves between teams and flows into data projects to yield insights (as well as where it might get stuck along the way).

In order to ultimately democratize the use of data through a company, each team and role (whether data architect, analyst, or data scientist) needs to work together. That becomes infinitely easier when each role understands what the other is doing and how their work affects the whole of the data machine.

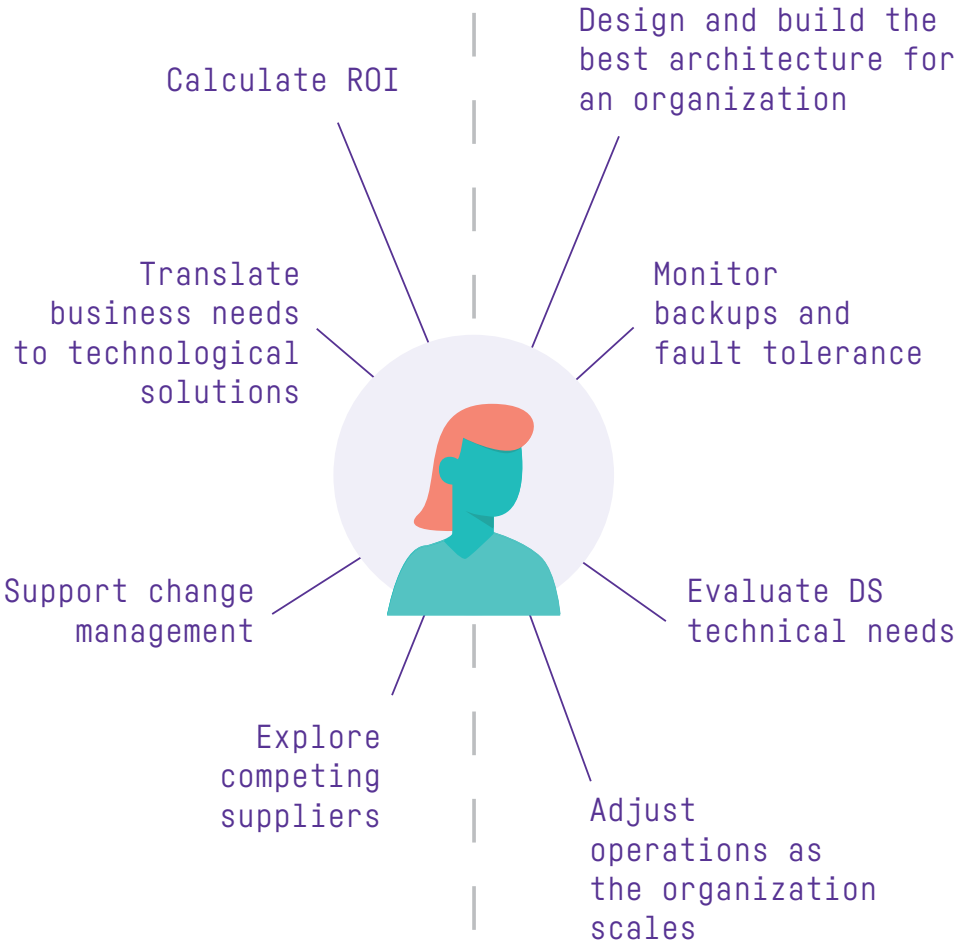
WHAT IS DATA ARCHITECTURE, AND WHO IS A DATA ARCHITECT?

Data architecture is a big umbrella term that encapsulates data storage, computational resources, and everything in between. All the technology that supports the collection, processing, and dashboarding of data is included in the architecture.

However, data architecture doesn't just include objects; it also includes the operational discipline surrounding data usage and compliance. Data architects can be said to act as data physicians, diagnosing both the physical maladies in the data pipeline and the organizational habits that are detrimental to the health of the system.

There is no typical day-in-the-life of a data architect, as their role is always evolving to the needs of the organization they are embedded within. Architects are often embedded in the IT team, but are multidisciplinary agents. They balance on the fine line between engineering acumen and business savvy, ensuring that communication, expectations, and KPIs between these teams are as unified as possible.

The Data Architect



This can manifest as monitoring backups one day or translating jargon the next. Some examples of data architects' jobs include:

- Collaborate with a variety of teams (hardware, machine learning, robotics, software, user experience, etc) to design and maintain an ideal data architecture.
- Exploring where and how process automation can best improve workflows.
- Getting proactive about data and service security by seeking out potential failure points and attempting to mitigate risks.

ABOUT THE GUIDEBOOK

Data architecture can be daunting, but the goal of this guidebook is to break down the basics in an easy-to-understand and non-technical manner. It is divided into two simple sections, which are the cornerstones of designing and maintaining data architecture:

1. Scalability, that is, the ability to handle the storage of more and more data without sacrificing performance of a system (and without causing costs to skyrocket).
2. Security, or the standards and technologies that protect data from being destroyed or modified (whether intentionally or not).

GLOSSARY

ASYNCHRONOUS REPLICATION

[eɪˈsɪŋkɹənəs ˌɹɛpləˈkeɪʃən]

The process by which data is copied to a replica or backup after it has been written to the primary storage location.

COMPUTE RESOURCES

[kəmˈpjut ˈɹisɔɹsɪz]

The processing capabilities of a system that are available to perform computational work (e.g. execute programs, carry out analysis, etc.).

DATA PARTITIONING

[ˈdætə pɑrˈtɪʃənɪŋ]

The act of splitting data into segments that are more easily maintained or accessed.

For example, distributed systems partition data in order to improve scalability and optimize performance.

DATA REPLICATION

[ˈdeɪtə ˌɹɛpləˈkeɪʃən]

The act of copying the same data multiple times to different systems.

KEY TERMS TO UNDERSTANDING SCALABILITY IN DATA ARCHITECTURE

DISTRIBUTED SYSTEMS

[dɪs'tɹɪbjutɪd 'sɪstəmz]

Segment the storage and compute resources of a system onto different machines that are able to run in parallel, thus speeding up work and minimizing the risk of single points of failure. The building blocks (storage and computation unites) of distributed systems are Nodes. A Cluster is a collection of multiple nodes.

FAULT TOLERANT

[fɔlt 'tɑlərənt]

The ability of a system to maintain usage during the failure of a component. A key benefit of distributed systems.

HADOOP

[eɪtʃ-eɪ-di-oʊ-oʊ-pi]

A popular cluster-based open-source framework for distributed storage. Different data processing engines or frameworks that can be used on top of Hadoop include (but are not limited to) Hadoop MapReduce and Apache Spark.

HIGH AVAILABILITY

[haɪ əˌveɪlə'bɪlətɪ]

The ability of a system to function continually without failure for a long period of time. A key benefit of distributed systems.

STORAGE RESOURCES

['stɔrədʒ 'risɔrsɪz]

The data storage capabilities of an architecture and a main requirement for all data science operations. Along with compute resources, these are a good indicator of an architecture's ability to scale.

SYNCHRONOUS REPLICATION

['sɪŋkɹənəs ˌɹɛplə'keɪʃən]

The process by which data is written to the primary storage location and a replica or backup location simultaneously.

PART I
SCALABILITY IN DATA ARCHITECTURE

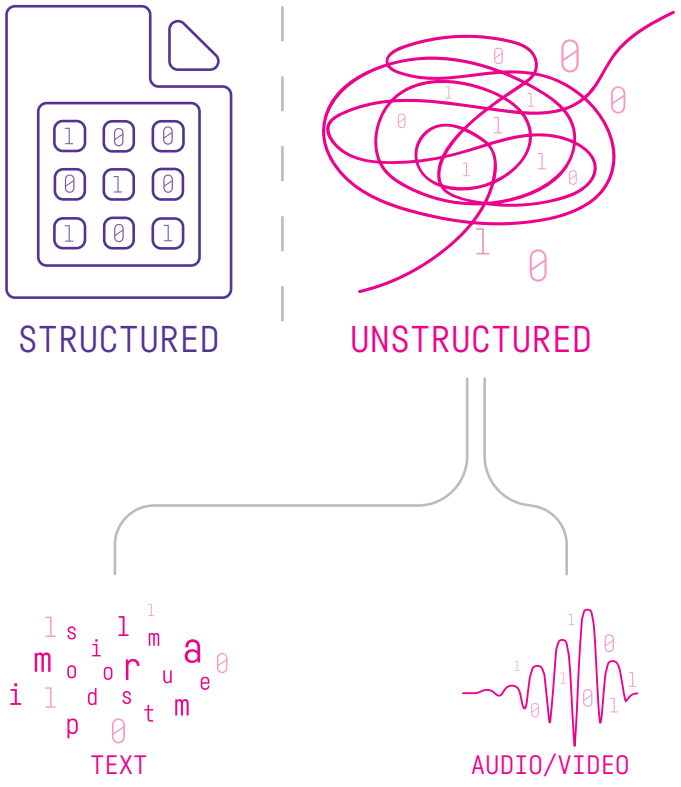
Scalability has two facets: scalability of computational power and scalability of data storage. When someone asks “Is it scalable?” in the world of data architecture, what they’re asking is whether the system could handle additional users with ease. Users can mean both externally (i.e., those producing data the company will need to store) or internally (i.e., those working with or processing the data).

After all, if changes in volume or traffic of data cannot be effectively handled, the system cannot possibly serve its users. In the best case scenario, the ability to work with data is slowed; in the worst case, data could be lost if a system isn’t flexible enough to handle the influx.

MORE DATA, MORE DOLLARS

Of course, the lifeblood of technical architecture and of scalability is the data. Without it, no one would need to worry about the complexities of architecture. However, it’s important to recognize as a non-data architect that data storage is not free (far from it). So when data architects think about data, one very important consideration that goes hand-in-hand with scalability is cost. And the types of data being stored have a big impact on storage, and, therefore, you guessed it - cost. While pricing and technological constraints may be limiting, architecture should ideally be constructed to accommodate every data aspiration, even if they are a few iterations down the line. That’s the essence of scalability in data architecture.

Types of Data



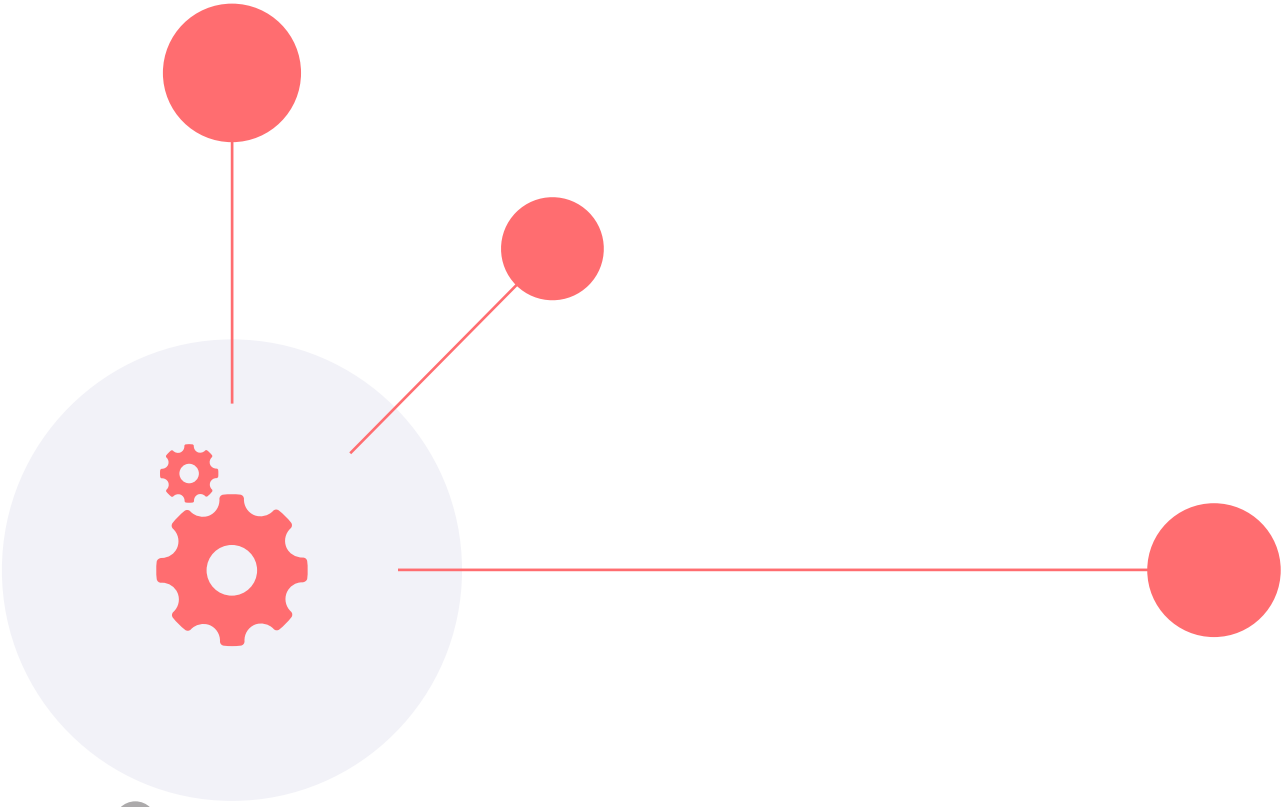
DATA ARCHITECTURE, CONTEXTUALIZED

To take an example that illustrates why scalability matters in context, let's say Bob's Flower Company starts collecting data on what customers purchase, when, and what online content they engage with. Bob creates a recommendation engine with this information and offers targeted coupons to his customers based on their preferences.

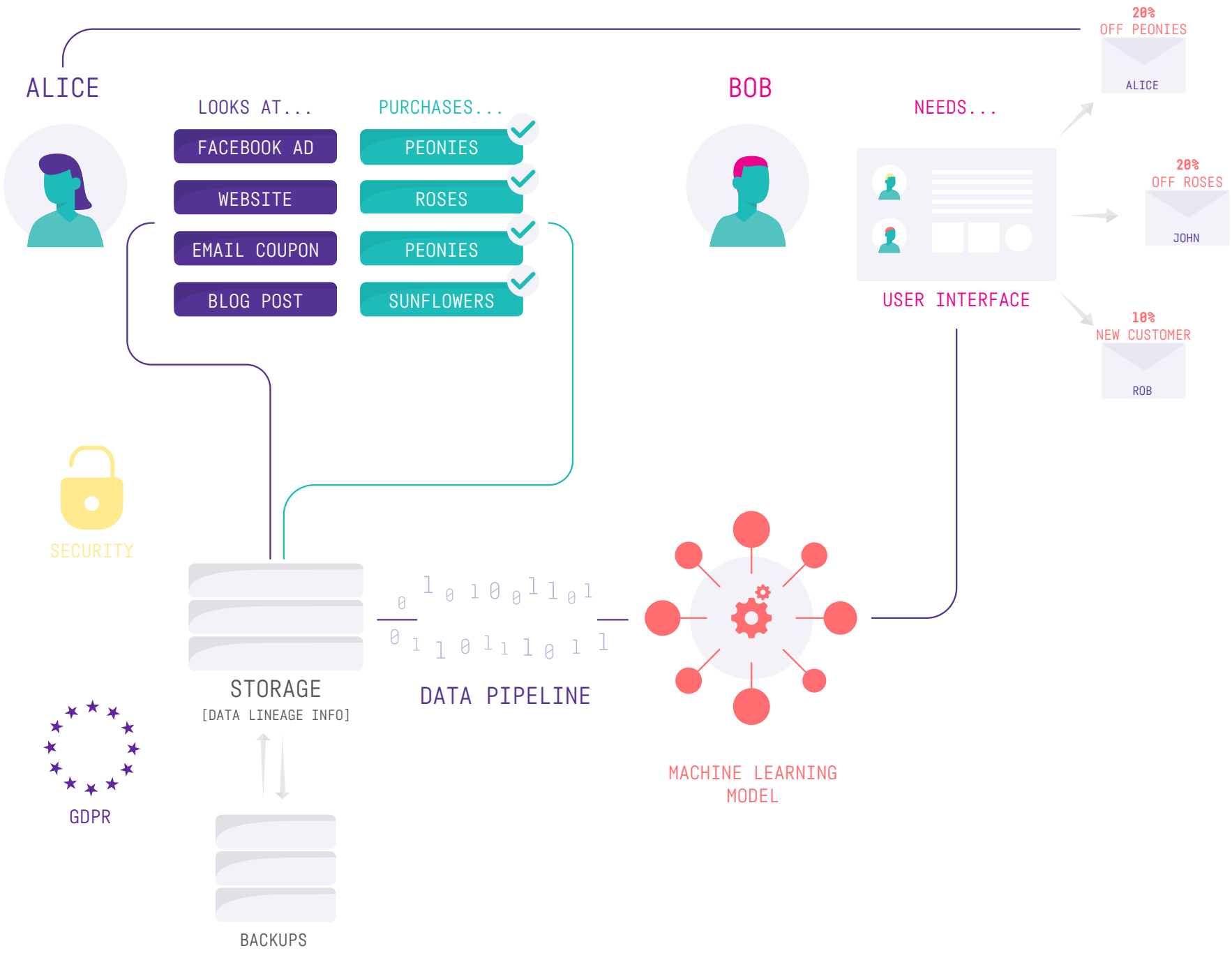
He stores this data on a local server and runs the machine learning model from a desktop. Engagement with the coupons works very well, and these customers, in turn, are more likely to log onto Bob's for spur-of-the-moment floral delivery needs.

Let's say the model works so well that Bob earns enough capital to open a brick-and-mortar location. However, Bob's data is stored locally, and his model is still run from the desktop, so he can't access the information in his new store. Instead, he has to download the new customer information and add it manually to his local model. Obviously, this is not an ideal situation and one that is prone to user error.

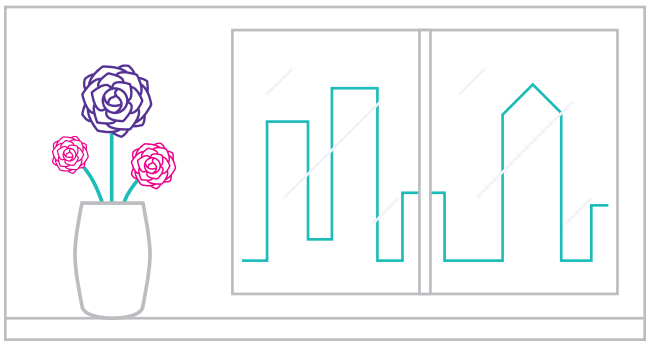
This may seem like an exaggeration, but enterprises follow Bob's architecture on a grander scale all the time. Unless organizations are able to anticipate their future data needs, their data will become a hindrance, not a boon. Thus, collaboration across teams and workstreams is critical when designing data architecture to help reveal as many areas for improvement or threats as possible.



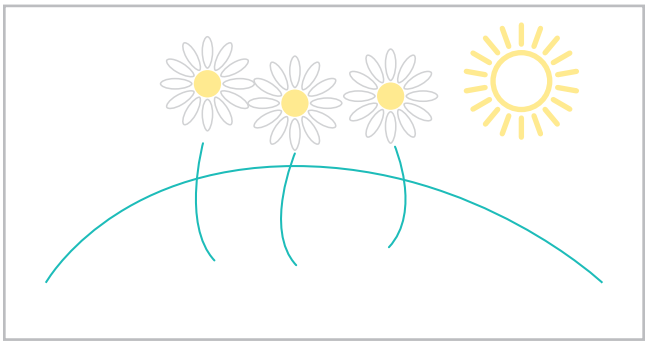
Data Architecture Contextualized



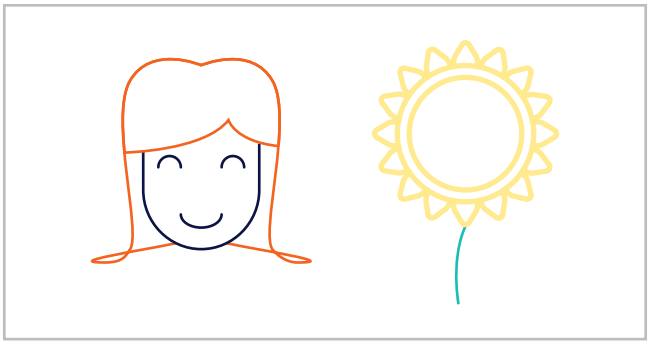
To contextualize with another example, let's say that Bob wants to more deeply understand what ads his user Alice responds to, so that he can target her better in the future. We know that she generally likes peonies, so Bob starts with three ads:



··1



··2



··3

At this scale, it's easy for Bob to say "the first option has peonies, the second daisies," but with even a hundred customers, human analysis at this level has a high time cost. It also ignores some of the image analysis deep learning can perform to extract context clues. So when Alice clicks on the third ad the most, Bob could better understand that she might not like sunflowers, but that she prefers the human element of that particular ad.

Looking at the architecture from the result, in order to analyze the images, Bob would need to have access to structured data, in the form of tags associated with each image (e.g., "peony," or "city"). These would be generated from deep learning image classification, which, depending on the size of the image database, can require an architecture with a lot of computational power.

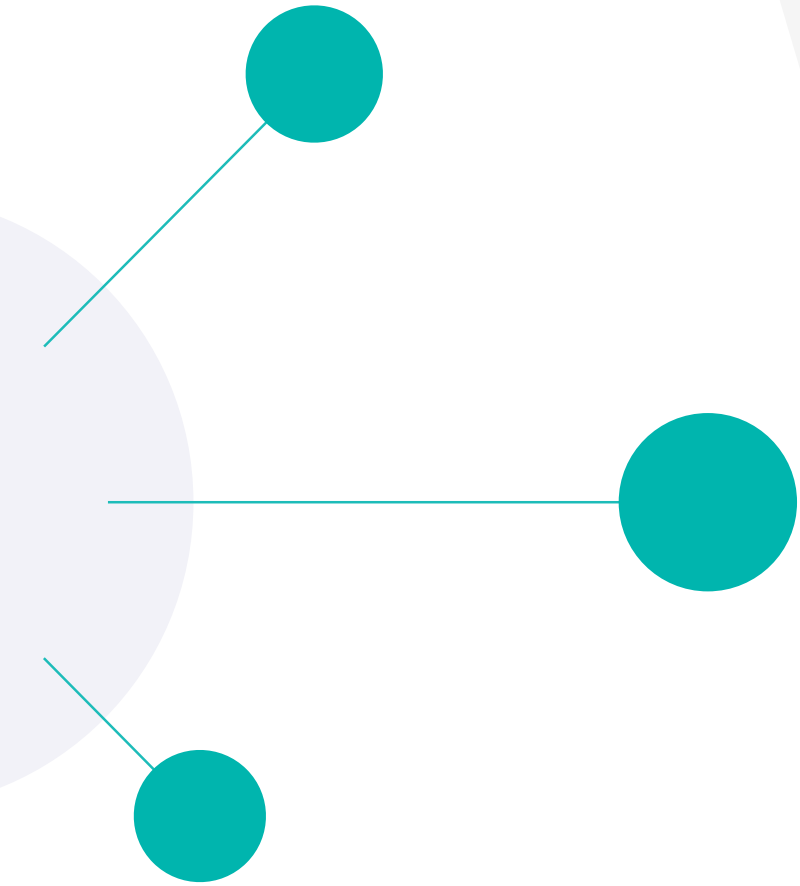
Ideally, this classification would be advanced enough to retain relationship information, (e.g. “peony” and “city” works better than “daisy” and city”), but capturing relationship information would (at least) double the data storage requirement. But before any tags can be calculated, Bob needs to collect the images to send and analyze.

Since high resolution visual files (as would be used in an ad) are relatively large, Bob may need to check with his architects to see if they can customize the data storage to efficiently accommodate compressed image files. It is at this point where Bob and his employees will really need to start to think about scale.

COMMUNICATION GUIDE

Business and architecture teams need to be on the same page when it comes to expectations about data scaling.

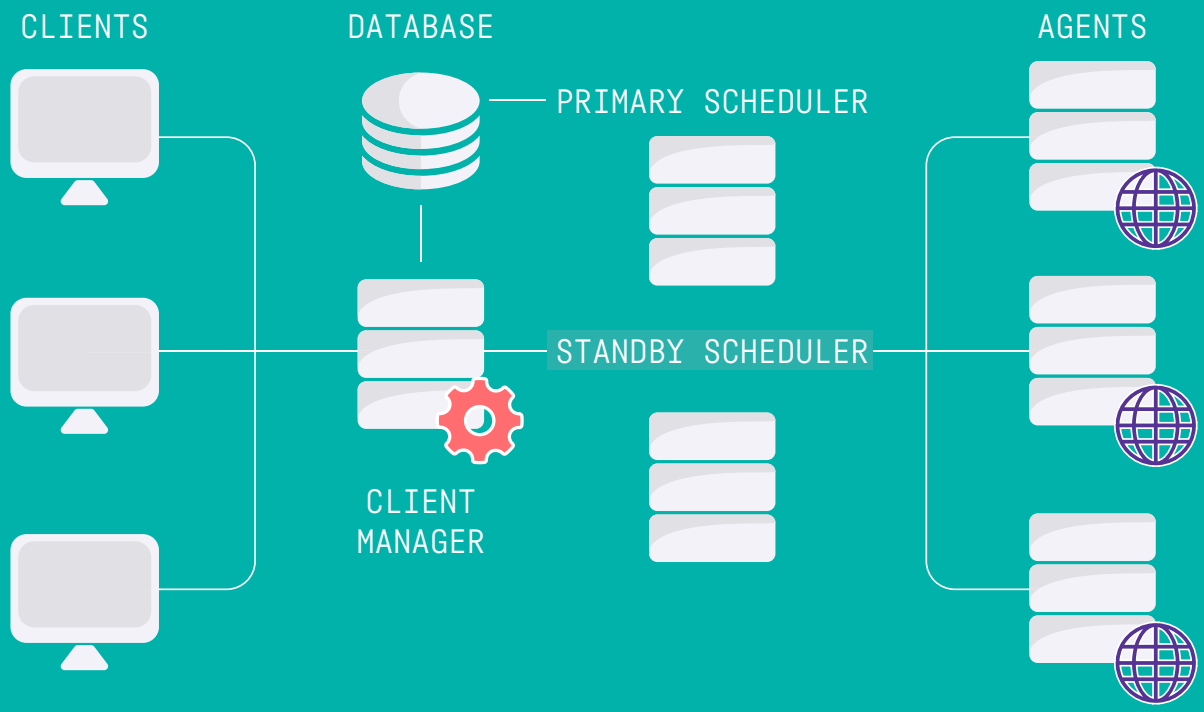
If an organization produces 1 TB of data a day but each new customer and model represents drastic increases to that total, an architecture must be built to accommodate these future needs.



A FEW TERMS OFTEN USED IN CONJUNCTION WITH DISTRIBUTED SYSTEMS ARE FAULT TOLERANCE AND HIGH AVAILABILITY:

Fault Tolerance is the ability to maintain usage of a system during a component failure. A system that has **High Availability** is one that has long periods of uptime (and so, obviously, very little downtime).

Fault Tolerance/High Availability Environment

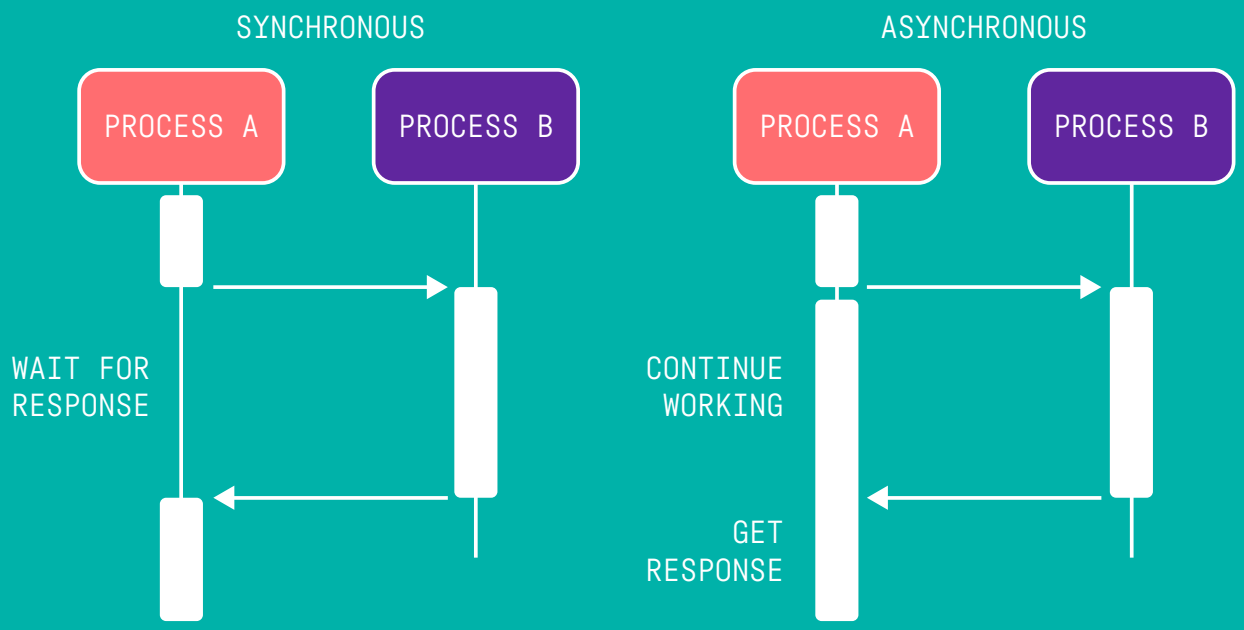


If an organization's operations are critical to the health and wellbeing of users, it is obvious that there is no room for error when it comes to either high availability or fault tolerance. But operations for consumer and media products face incredible scrutiny, too. The threat of hemorrhaging users is aways business-critical, and a good defense is driving user trust by creating a highly available, fault-tolerant system.

Data replication can help ensure high availability, and it can also help with fault tolerance. However, running multiple databases with the exact same information can not only be expensive, but - perhaps more critically problematic - complicated.

In a paired system, every time a new piece of data is added to the primary database, the same piece of data needs to be placed in the backup; thus, each byte must be double stored (running at least double the cost).

More important than the added cost is the added intricacy, which must be carefully considered as it can add some risk. Synchronous replication happens when writing data to primary storage and the backup or replica simultaneously. Asynchronous replication means that data is copied to the backup or replica after it is already written to the primary storage location. Choosing between the two means balancing riskiness and speed (with synchronous replication being lower risk but slower, and asynchronous replication being higher risk, but faster).

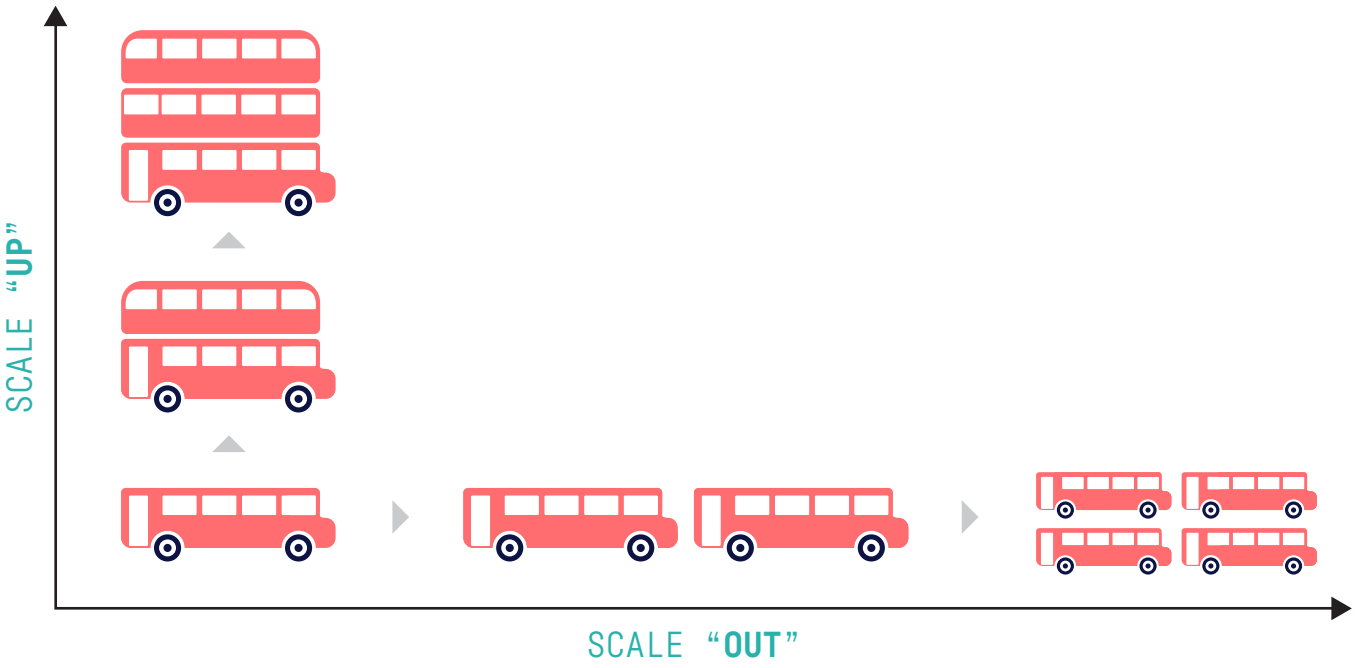


The risk stems from the fact that the system (or mapper) that copies data represents a dangerous single point of failure. If the mapper goes offline temporarily, the backup data will become outdated, and it will be challenging to determine which data needs to be updated.

As an example, let's come back to Bob's ecommerce site - let's say he has implemented asynchronous replication as a way to ensure high availability and fault tolerance. A user is trying to place an order, and at that very moment, a bad thunderstorm has caused a failure in Bob's system. The order has thus been recorded in the primary storage system, but has not been replicated to the backup. The order thus does not get placed, yet the customer's card is charged as a result of the data writing to primary storage.

DISTRIBUTED SYSTEMS

There are two basic ways to scale: either by scaling up, in other words, adding more resources to an existing system. Or scaling out - and that's where distributed systems come in. At some point, it's not possible to rely on one really big machine for data storage; instead, it makes more sense to have multiple machines connected together.



In a distributed system framework, data is:

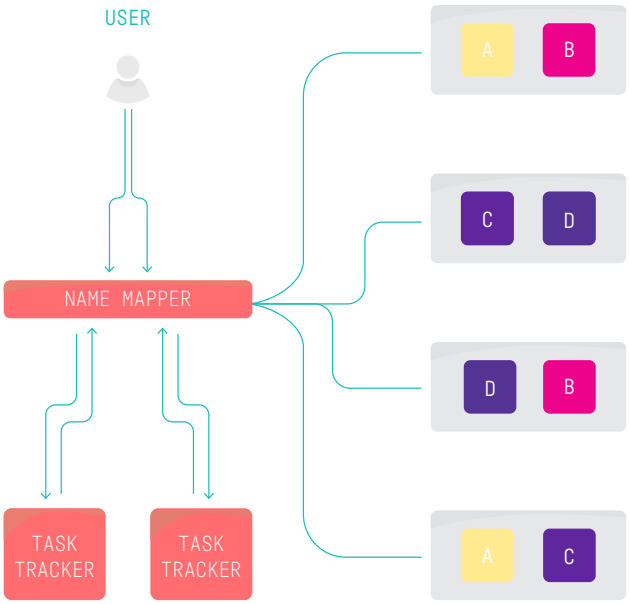
1. **Partitioned**, meaning split apart into smaller pieces and stored on different servers.
2. But it can also be **Replicated**, meaning that those partitioned bits are also copied several times. Read more about the intricacies of replication in the sidebar on high availability and fault tolerance.

Since data is broken into small chunks, computation is performed in small parts on each impacted server, thus "sharing the load" and improving overall computation accessibility and speed. Distributed systems are often faster than centralized ones (though it's important to note that they are not always faster, as speed also depends on algorithms and types of calculations being performed on the data), and they are less prone to single points of failure. They also scale well with rapidly growing databases.

Where data is stored can also determine a lot about how it can be used and ultimately how scalable the system is. Thus, as always, it's critical to select a storage system that works with an organization's business needs. Data that must cross the Atlantic ocean with each transaction in order to be stored will add some latency, so the question of response

time comes into play and must be weighed against other priorities. For example a ride-sharing company needs to provide fast service globally and can usually benefit from more isolated geographic units (i.e., data generated in Europe stored in Europe, data generated in North America stored in North America, etc.).

Distributed System



There are many different types of distributed systems that can be implemented in an architecture. **Hadoop** is very common, as it is cost-effective, incorporates redundancy to help with fault tolerance, can handle a variety of data types, and is (relatively) fast.

Hadoop was designed to handle massive amounts of information efficiently. It is also relatively inexpensive because the idea is that companies can leverage lots of cheap servers instead of one giant, expensive server. However, it might not be right for a particular organization's needs if:

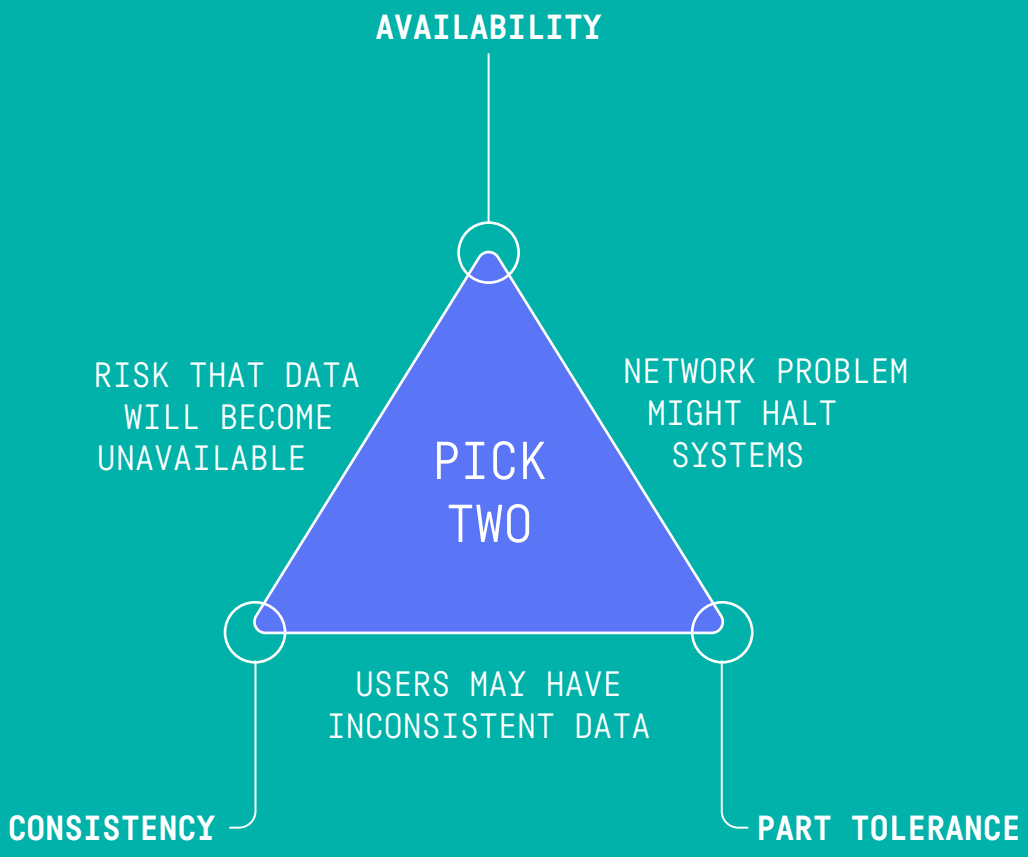
- The technical team does not have Hadoop experience (and cannot hire for that experience)
  - Performing any adjustments or maintenance on this type of system will be a challenge without the requisite experience
- There is a relatively small amount of data, and it's growing slowly
  - There are easier, smaller-scale solutions
- Real real-time analytics are needed
  - Hadoop MapReduce is much slower than Spark or Impala. For speed at scale, a combination of tools may be needed

FEATURE: CAP THEOREM

The CAP Theorem helps set expectations when it comes to distributed systems and can hopefully help prioritize what each architecture needs based on the systems being built by the business.

The theorem states that no distributed data storage system can have more than two of the following features:

1. Consistency - every read request returns the most recently written information (or error).
2. Availability - every read request receives a non-error response, but the information returned may be old.
3. Partition Tolerance - even if messages between nodes fail, the system will sustain operations.

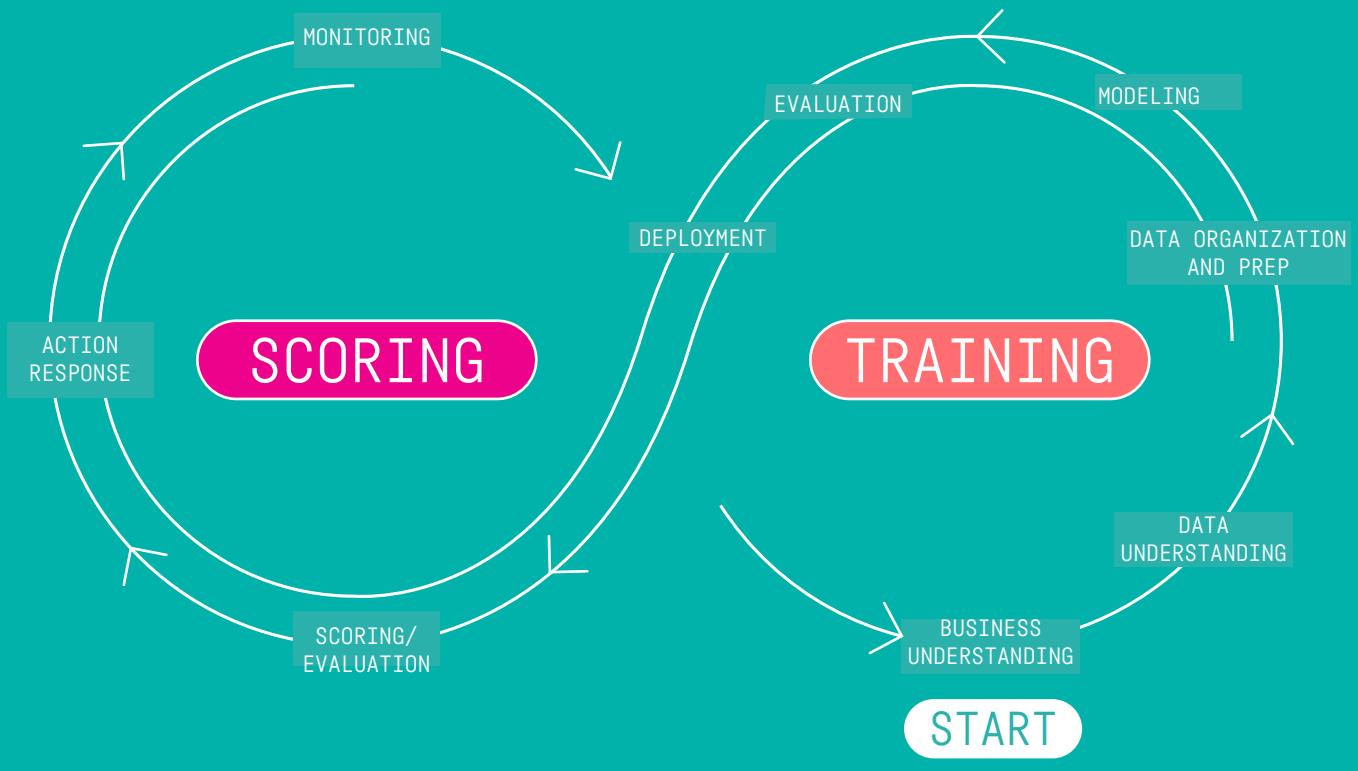


Whichever two functionalities are prioritized will depend on the output, users, and data. For example, if the output is a content recommendation engine, availability and partition tolerance matter more than consistency; if advice is a few minutes stale, it will likely still hold. On the other hand, if users are doctors analyzing medical records, consistency and availability are likely more important.

A WORD ON "REAL TIME" DATA

When users want real-time data, architects think distributed systems. But depending on the use-case, operations occurring in "real-time" can mean vastly different things to different people, so the key as a non-data architect is to be clear. If dashboards need to be updated every minute as opposed to refreshing overnight for the next day, this places divergent constraints on what the architecture needs to support.

Nothing beyond physical sensors is actually able to function in real-time, but so long as communication about the time-sensitive needs of the business is clear, then architectural priorities can respond.



THE CLOUD

Cloud computing is definitely not some next-step distributed computing; it's just a different way to provision and use infrastructure and managed services. For a high-level discussion on security and scalability, being on-premise or in the cloud should not matter too much when you define the basic principles.

The cloud is also not innately more or less affordable; however, the cloud can enable elastic resource allocation, which can result in less wasted storage. If managed incorrectly, it can still become too expensive. A self-generated cloud is unlikely outside a super-massive organization; this means that cloud services will come from a third-party provider (e.g., AWS).

While the cloud seems like an obvious choice, when determining if it is right for an organization, the costs and the value need to be compared for each organization.

Cloud providers are only as good as their security, so this remains a top priority. While no system can be completely fault-tolerant and secure in all cases, the security measures and access permission restraints are severe. However, the CAP Theorem still holds, so no system will be able to solve every data concern with absolute certainty.

Cloud Architecture

PROS	CONS
• Rapid scalability means the architecture can keep up with the business • Global accessibility enables comparisons between different markets • Fewer up front IT charges • No hardware requirements • The cloud service is responsible for fault tolerance • Fewer up-front IT charges	• Forfeit of data custody means businesses must trust their cloud provider to protect their data

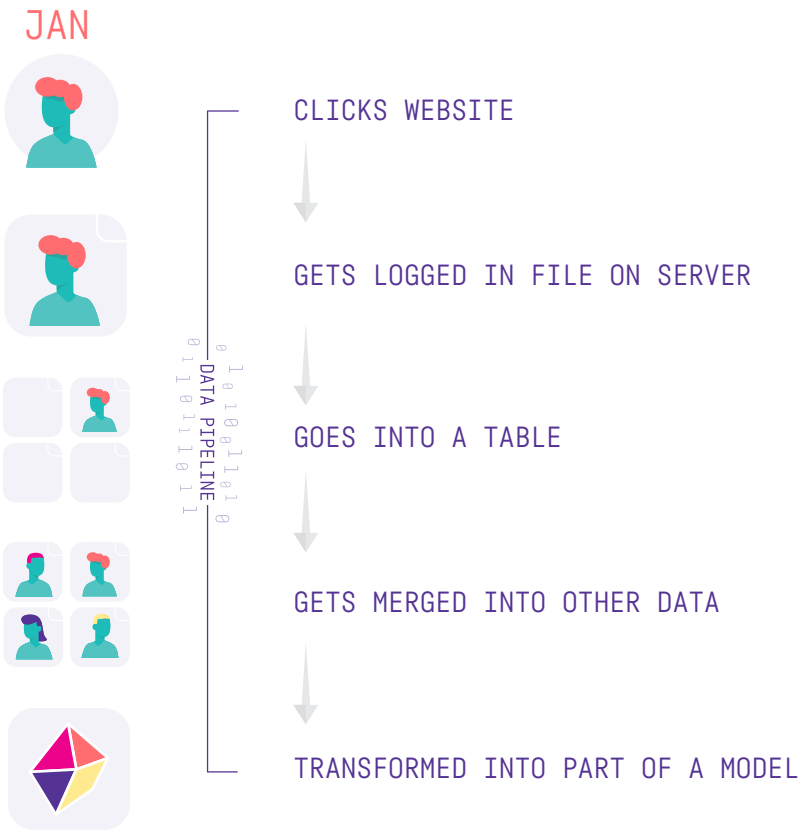
SCALABILITY AND THE END-USER

One critical part to scaling data architecture is making sure that end users (whether they are data scientists, analysts, or line-of-business people) can interact successfully with the data pipeline.

WHAT'S THE DATA PIPELINE? HERE ARE THE BASICS:

- Data pipelines are the paths data takes through the architecture once it's been created.
- The data pipeline follows the flow of processes and systems that enable the data to be used by analysts, data scientists, etc., in the context of a data project.
- Everything from data collection, storage, access, cleaning, analysis, and presentation are included in the data pipeline.
- The pipeline is often thought of in terms of plumbing; ideally, the computational and storage resources (i.e., the pipes) are wide enough to support the transmission of the necessary information, but not oversized and inefficient.
- Building a pipeline is never a one-off task; continued maintenance and optimization are a necessary part of any infrastructure, including data architecture.

Data Lineage



But ultimately, data architecture and data pipelines are just infrastructure; they doesn't inherently provide a way in which others can interact with it. This is something that must be considered separately.

Clearly, there are some pretty fundamental differences between data consumption for coders vs. non-coders:

Persona	Non-Coders	Coders
Priorities	- Legibility	- Flexibility
Needs	- Automatic updates - Sleek user experience	- Adaptability of coding environment

For example, if Bob is the one that currently governs his data analysis insights, he might care more about an easy dashboarding user experience to make sure he can't break anything and can clearly understand his insights. But as he grows, Bob hires a data scientist, who wants much more flexibility and control permissions. The end of the pipeline may still terminate with a dashboard for Bob's understanding, but it's stretched to incorporate the needs of technically savvy staff members.

This is where tools like data science, machine learning, and AI platforms (Dataiku is one example) come in. They sit on top of the underlying data architecture and provide end users - whether technical or not - an easy way to build and interact with data pipelines.

In addition, data science, machine learning, and AI platforms like Dataiku are useful because they offload computation, meaning that the tool inherently scales automatically as infrastructure scales. Having to worry about scaling the front-end tools as well as the back end can make questions about data architecture much more complex.

"Expert data scientists, citizen data scientists and application developers require professional capabilities for building, deploying and managing analytical models."

The January 2019 Gartner Magic Quadrant for Data Science and Machine Learning Platforms.[1]

GLOSSARY: KEY TERMS TO UNDERSTANDING DATA ARCHITECTURE SECURITY

AAA

[ə ə eɪ]

Stands for authentication, authorization, and audit, which are the three pillars of data architecture security.

AUDIT

['ɔdɪt]

The ability to trace and review everything that's been done within the system.

AUTHENTICATION

[ɔˌθɛntə'keɪʃən]

The security process by which a user or process confirms its identity. This can occur in one location (as is the case with SSO) or through multi-factor authentication.

DATA LINEAGE

['deɪtə 'lɪniədʒ]

Covers the entire path data takes from creation to storage to analysis. It also incorporates who "owns" what data at a given time, which is critical transparency for compliance regulations.

AUTHORIZATION

[ˌɔθərə'zeɪʃən]

The security process by which the system gives a user or process the ability to read and write data or execute programs within certain parts of a system. Depending on the user's clearance level, this may represent a small section of data relevant to their processes, or could include the ability to act as an admin and authorize other users.

USER PERMISSIONS

['juzər pər'mɪʃənz]

Enable users to work with a certain set of data. They come in three stages: Read, where a user can consume the data stored in the system; Write, where a user can modify data; and Execute, where a user can execute commands and programs that can impact the database structure.

PART II
SECURITY IN DATA ARCHITECTURE

Security includes technologies, techniques, processes, and best practices that guarantee the integrity, availability, and the enforcement of data and information system governance. It encompasses technical protocols and user behavior and usually centers on what behavior to normalize.

While business users may primarily care about having access to the data they need, daily analysis doesn't occur in a vacuum. It's informed by the rights of users, designed to protect the system's function, and governed by compliance regulations.

Malicious actors could try to compromise the system, or more likely, an unknowing user might do so inadvertently, which could result in temporary loss of service or more permanent destruction. Another risk is that employees in the organization see data or projects that they shouldn't have access to - especially in the age of data privacy, it's critical to protect against internal data leakage. Security protocols are not put in place to be annoying, but rather to protect business users and the broader organization from large data failures or leaks.

Going back to Bob's Flower Company and its delicate architecture - let's say that he doesn't have solid data governance or policies surrounding **AAA** (that is, authentication, authorization, and audit) in place. As a result of this, a nefarious employee gets access to credit card numbers, which (s)he sells on the black market. Bob has now put his customers at risk as a result of his lack of attention to the security of his data architecture. This is just one example of why data architecture needs to not only be scalable, but secure.

The three main questions to ask about security are:

AUTHENTICATION: IS THE USER WHO (S)HE SAYS (S)HE IS?

Authentication is the equivalent of ID verification at the airport using a driver's license or passport. The purpose is to ensure that the user is who (s)he is declaring to be. In data architecture, authentication happens through something the user knows (i.e., a password) plus sometimes for even more security, something (s)he has.

The latter is known as **Multi-Factor Authentication**. Using this protocol, users must provide at least two proofs that they are who they claim to be when

logging in. This often manifests as a login and supplemental sign-on from a trusted device like a cell phone, physical device or dongle, etc. While this takes a bit longer to sign in, the proof that users are who they claim to be is much stronger.

Another term worth being familiar with related to authentication is **Single Sign-On (SSO)**, which is a system that allows a user to authenticate once and then access a variety of systems based on that authentication. It's like the bracelets given out in bars once ID has already been checked by the bouncer - it proves that the user (or in this case, the customer) does not need to show ID for each purchase since (s)he has already been verified. **SAML**, or Security Assertion Markup Language, is one common protocol that enables single sign-on (SSO).

AUTHORIZATION: TO WHAT DATA DOES THE USER HAVE ACCESS?

Authorization is distinctly different from authentication. Considering again the example of the airport, while authentication is the ID or passport check, authorization is like a boarding pass - it gives you access to only board a certain airplane.

In the world of data architecture, whether or not a user can access a particular set of data is often a legal decision, not a business one. Compliance with privacy restrictions may limit right out of the gate what data users leverage. The European Union's General Data Protection Regulation (GDPR), for example, limits the use of personal information to only that which the person has agreed to allow.

But business reasons matter as well when it comes to authorization. Looking back at the example with Bob in the previous section, it's clear why having tight policies around authorization is crucial to protecting internal data from employees (whether their intentions are malicious or not).

Given restrictions for legal and business reasons, allowing employees to work with test data more freely in early stages of data projects is often helpful, which is where development (or sandbox) and production environments come in. In addition to allowing more agile use of data, a sandbox environment enables technical users to test out new features and catch bugs without pushing them to the entire system.

From an architectural standpoint, this can become expensive, as the additional computing power required for a sandbox environment is never enough (according to developers). However, this sort of system mirroring is both beneficial to services and critical for security so that developers cannot inadvertently modify user data.

AUDITIBILITY: CAN WE SEE LATER WHO ACCESSED WHAT, WHEN?

Aside from maintaining development environments to ensure data doesn't get modified unintentionally, knowing who modified what data is also critical when it comes to collaboration on data projects.

PERMISSIONS

From a data architecture perspective, there are three basic ways of accessing and modifying data, each of which represents a different level of permissions granted to the user or applications: Read, Write, and Execute.

These permissions are clustered, so often a user will have read and write permissions, but not execute, or read and execute, but not write. Understanding at this granular level the ways users engage with data is important because it highlights the balance that needs to be struck between ease and security.

If every user had every permission and eternal access, there would be no governance, and processes might run smoothly. But in the same situation, inept or malignant users could corrupt a system or files, irreparably harming data projects. Priorities will need to be clearly outlined in advance to help establish when to swing one way or the other and how to strike the correct balance for an organization's needs.

As the trend toward data analysts and data scientists working together in the same environment grows, being able to maintaining clear records of who adjusts what data is crucial.

In addition, the ability to perform data usage audits may be a necessary part of system requirements depending on the type of data used and the compliance standards to which an organization is held. But even if it's not a legal requirement, it can be helpful as part of good data governance practice to consider implementing audit systems to monitor usage and help ensure security standards are being met.

However, like any fun perk, audibility can be very computationally expensive from a data architecture standpoint. While in theory, of course, everyone should know the access histories for their data, this must be prioritized against speed and computation spend

Who is accessing the data	What data are they accessing	When are they accessing it
PROS		
• Helps evaluate authorizations and can be used to trim permissions wherever possible • Useful safeguard against malign internal actors or for tracking system errors	• Establish data lineage for GDPR/ HIPAA • Determine what data to put in archival storage and what to keep in memory • Can help target compression	• Accentuates deviations and unsanctioned behavior
CONS		
• Requires unique IDs for each user • Logs can take up exorbitant space depending on the frequency of access and number of users	• May result in confirmation bias if users always turn to the same source of information when others are available • Not particularly useful without other audit metrics	• Not particularly useful without other audit metrics

COMPLIANCE

If an organization accesses or creates any government data, there are likely additional stipulations regarding its storage and use. Data sovereignty restrictions often limit the availability for cloud structures to be considered.

Compliance with regulations are critical considerations when dealing with personal data, but organizations can (and should wherever feasible) go beyond the bare minimum when it comes to safeguarding their users' data. Prioritizing user rights can help improve confidence and loyalty to an organization, providing greater benefits in the long term.

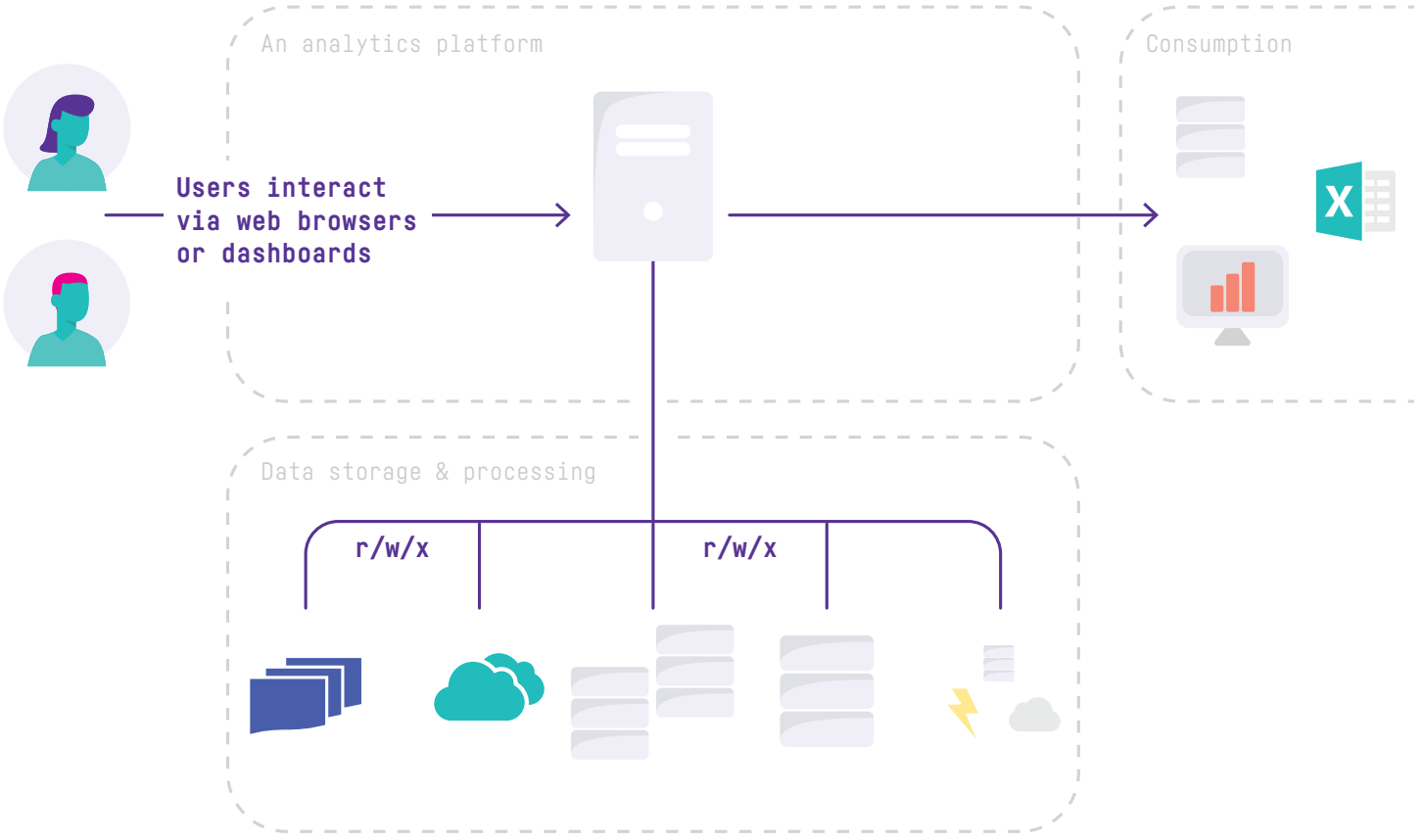
It's worth noting that whatever audit data is being captured, it is best practice to put it into a separate system immediately. That way, any bad actors can't modify the evidence that they were making changes to a system.

CONCLUSION

ARCHITECTURE EXAMPLE

By definition, every data architecture is customized to the organization it is embedded into. If the communication between business users and architects is open, then the data architecture will be able to empower users instead of hampering them.

As with any complex system, what works for one company doesn't necessarily work for another - cookie-cutter layouts cannot support organizational goals. That said, it's useful to visualize what a data architecture might look like:



ARCHITECTURE TRANSITIONS

Any data architecture transition is not something to undertake lightly. This represents a major immediate cost in addition to the process changes that come with new foundations. However, for an organization to achieve its full potential, data must be a boon, not a hindrance. Only by taking advantage of data can innovation occur at scale and at speed.

When an organization transitions to an architecture that enables better analytics, more robust security, or failure-resistant storage, it will be better able to serve its clients and achieve its business goals. However, just like any major change, operational adjustments will lead to (at least) short term inefficiencies and pain points.

In their 2019 Planning Guide for Data and Analytics, Gartner recommends to "shift your focus from collecting data to embedding analytical functionality in existing applications and integrating that functionality into custom product offerings," and to "combine architectural styles into a portfolio-based approach to building end-to-end data and analytics architectures. Use the architecture outlined in this document as a baseline."

With a holistic data vision, organizations will be set up for success. By the same measure, a piecemeal architecture transition may be doomed to inefficiencies and errors.

There is no quick-and-easy answer to the data architecture for each organization. Distributed computing is fault tolerant, but slow and difficult to maintain. The cloud takes a lot of this labor away, but is expensive and may not jive with data compliance needs.

This guide stresses communication and collaboration for a reason; if the architectural and business teams can get on the same page regarding priorities and expectations, then the resulting data architecture will equip that organization for the future.

"The complex and forever-changing requirements of analytics have created a scenario in which no one architectural style is sufficient to execute all required analytical use cases. To meet future demand, a more proactive and coordinated data and analytics strategy founded on a repository of architectural styles will be required."

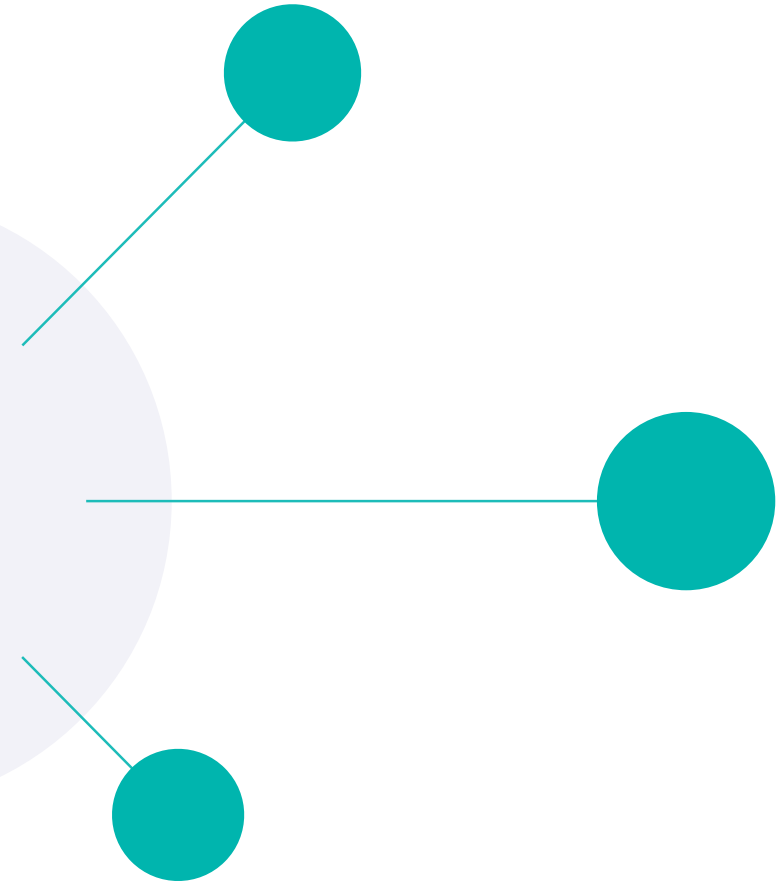
Gartner, 2019 Planning Guide
for Data and Analytics.
October 2018.

Subscribers may access
https://gtnr.it/32Ixg4q

Taking time for education up front is critical, in all teams that engage with an organization's data. Everyone needs to understand both how to make the shift to their processes and why such a shift might help them and the organization in the long run, thus enabling a happier (and thus, smoother) architectural transition.

COMMUNICATION GUIDE

Just like in any other business change situation, during data architecture transitions, make sure that communication between the technical and business teams is clear and frequent will help set realistic expectations and clear priorities.



FINAL WORDS AND NEXT STEPS

Now that you have a better understanding of what data architecture entails, you'll be equipped to communicate with your organization's architecture team to ensure that business needs are able to be met. The most important things to remember when sharing business priorities with the architecture team are:

Establish shared definitions:

Ensure that business users and architects are on the same page about key terms.

Take the goals of each team into account:

See if collaboration and overlap is feasible so teams can inform each other's best practice development.

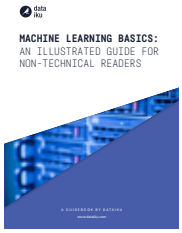
Keep the end client or customer in mind:

An organization's service must work beautifully when and where a client needs it, without breaching their trust or expectations for data usage.

Data transformations are messy, painful processes, but if you can come out the other side with a fine-tuned architecture, your organization will be equipped to thrive.

FURTHER READING

- Technoslavia
  http://bit.ly/Technoslavia



- Machine Learning Basics
  bit.ly/MLBASICS

data iku

Your Path to **Enterprise AI**

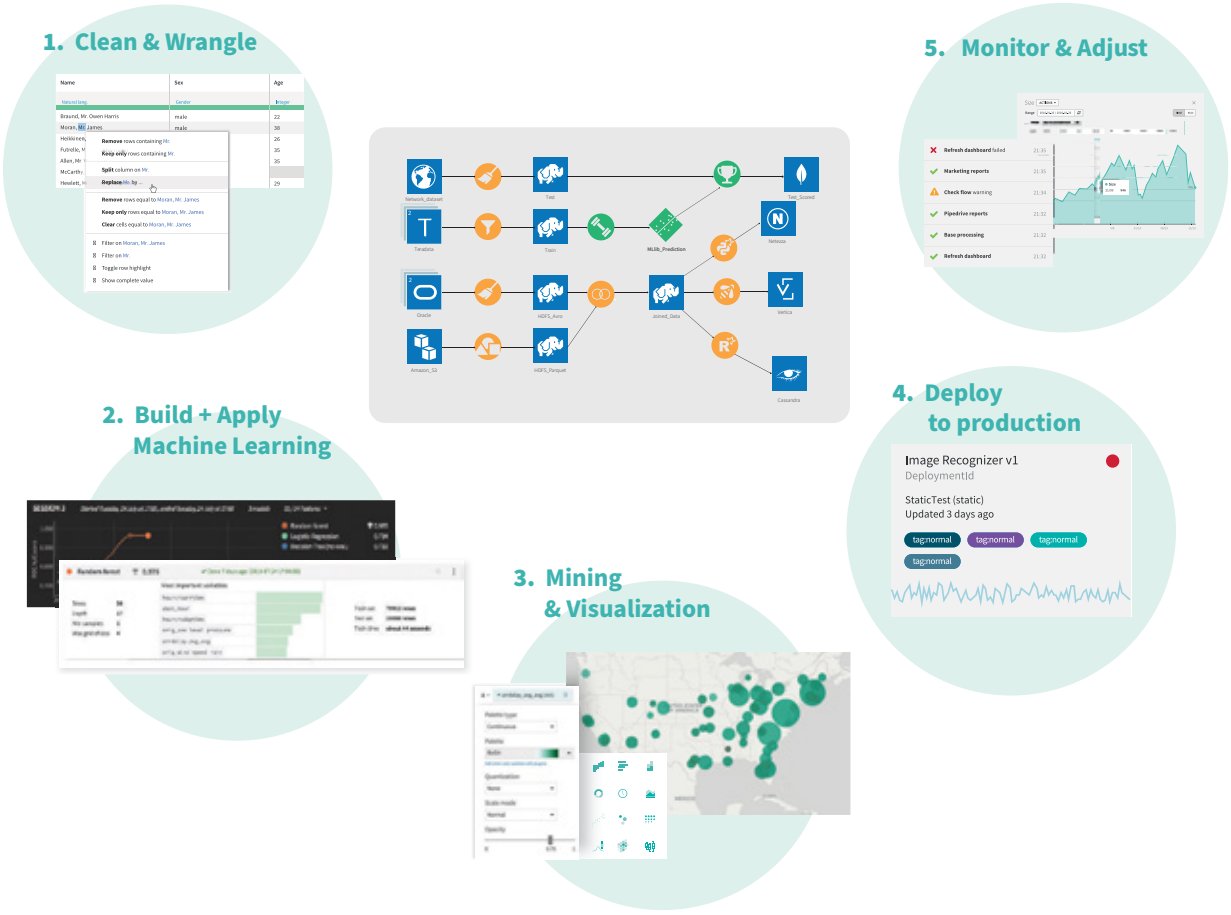
Dataiku is the platform democratizing access to data and enabling enterprises to build their own path to AI. To make this vision of Enterprise AI a reality, Dataiku is the only platform on the market that provides one simple UI for data wrangling, mining, visualization, machine learning, and deployment based on a collaborative and team-based user interface accessible to anyone on a data team, from data scientist to beginner analyst.

300+
CUSTOMERS

30,000+
ACTIVE USERS

*data scientists, analysts, engineers, & more

GE · Santander · FOX NETWORKS GROUP · Unilever · KUKA · BNP PARIBAS · paloalto NETWORKS · PREMERA BLUE CROSS · NXP · SEPHORA



WHITE PAPER

www.dataiku.com