Bild: Shutterstock

Sonderdruck aus
BI-SPEKTRUM 1/2023

Der Weg zu einer multinationalen Datenplattform

Konzerndaten im Weltall

Ein Beitrag von
Lars Beumann

Stellen Sie sich vor, Sie möchten einen ganzen Stab an Satelliten ins Weltall schießen, um dort ein zentrales Satelliten-Netzwerk aufzubauen. Ein komplexes Unterfangen, das von vielen Faktoren beeinflusst wird, wie zum Beispiel der Kommandozentrale, dem Weltraumbahnhof, dem Wetter und so weiter. Mit ähnlichen Einflüssen hatten wir es beim Aufbau einer Datenplattform in einem multinationalen Unternehmensumfeld zu tun. Bevor es daran ging, die Satelliten der Plattform „ins Weltall" zu befördern, waren Technologien auszuwählen, die optimal zu den Anforderungen passen. Es galt stets äußere Einflüssen wie Architektur- und Security-Richtlinien oder Projektabhängigkeiten im Auge zu behalten. Wie wir diese „Mission" erlebten, zeigt dieser Artikel Schritt für Schritt.

Die Mission wird vorbereitet

Alles begann mit einem Projektsteckbrief, der die Vision für eine Datenplattform beschreiben sollte – eine Plattform, die zentral verwaltete und qualitätsgesicherte Unternehmensdaten für zentrale wie nationale Use-Cases bereitstellt. Was sich im ersten Moment einfach anhörte, entpuppte sich bei der genaueren Betrachtung als komplexes Unterfangen.

1. Wo liegen die Treiber für das Projekt?
Neben einem auf Oracle-Technologien basierenden klassischen Data-Warehouse-System gab es bereits seit mehreren Jahren ein On-Premises-Hadoop-System, das Hunderte Terabyte an Unternehmensdaten beherbergte. Im Laufe der Zeit implementierte die zentrale IT auf diesem System immer mehr analytische Use-Cases und der Datenbedarf für die Beantwortung fachlicher Fragestellungen wuchs rasant. Das hatte zur Folge, dass die Ladestrukturen bald nicht mehr zu den wachsenden Anforderungen passten, was wiederum die Analysemöglichkeiten einschränkte und damit die Arbeit der Data Engineers und Data Scientists blockierte. Auch die Skalierbarkeit des Systems stellte die IT-Abteilung vor immer größere Herausforderungen.

Zum Beispiel war es den dezentralen nationalen IT-Abteilungen und einigen Fachabteilungen wichtig, eigenverantwortlich an Analytics-Use-Cases arbeiten zu können, die nicht durch die zentrale IT betreut werden.

2. Welche Anforderungen muss die Datenplattform erfüllen?
Dank des vorhandenen Hadoop-Systems waren dem Team bereits einige Use-Cases bekannt. Teile davon waren bereits implementiert und bereit für die neue Plattform. Allerdings gab es viele Anforderungen, die noch offen waren – teils aus technischen, teils aus Kapazitätsgründen.

Um einen detaillierten Überblick zu bekommen, teilte das Team die Anforderungsaufnahme in zwei Teilprojekte: ein Teilprojekt für die Erhebung der fachlichen Anforderungen, ein Teilprojekt für die eher technisch orientierten Anforderungen der zentralen und nationalen Data-Science- und Data-Engineering-Teams. Das erste Zwischenergebnis wurde harmonisiert. So entstand eine Liste an Epics für das Product Backlog.

Nach der Anforderungsaufnahme wurde das Bild klarer. Zwei Anforderungen waren besonders deutlich:

- Die Datenplattform soll eine Vielzahl an unabhängigen Data-Analytics-Umgebungen bereitstellen, die neben individualisierbaren Konfigurationen auch unterschiedliche technische Anforderungen erfüllen.
- Es sollen Daten von hoher Qualität, aktuell und vollständig zur Verfügung stehen, die bei Bedarf durch spezifische Daten eines Use-Case angereichert werden.

3. Wie soll die Plattform technologisch aufgebaut werden?
Nachdem die groben Anforderungen nun bekannt waren, ging es an die Technologieauswahl. In die engere Auswahl kamen Alternativen, die eine Plattform schnell und in alle Richtungen skalieren lassen, ohne dabei an Flexibilität einzubüßen. Hier lohnte sich ein Blick in Richtung Cloud. Schließlich sind es Eigenschaften wie Skalierbarkeit, Verfügbarkeit und Flexibilität, mit denen die großen Cloud-Unternehmen für ihre Services werben [Azu23].

Da sich das Unternehmen im Rahmen seiner IT-Strategie bereits für die Microsoft Azure Cloud entschieden hatte, war die Richtung hier klar und wir konnten früh den Kontakt zu Microsoft aufnehmen, um kritische Fragen zu klären, wie:

- Wie lassen sich die Zugriffe auf Azure-Storage-Accounts steuern?
- Welche Dinge sind in Bezug auf das Netzwerk zu beachten?
- Wo könnte es zu Performance-Einschränkungen kommen?

So kam heraus, dass bei vergleichbar großen Datenmengen die Schreib- und Lesegeschwindigkeit einzelner Storage-Accounts ebenso wie die Netzwerkbandbreite zum Flaschenhals werden konnten. Eine Aufteilung in verschiedene Container würde hier nicht ausreichen. Stattdessen wurde geraten, die Daten auf verschiedene Storage-Accounts zu verteilen [Mic22-1].

4. Welche Daten werden in der Plattform gebraucht?
Alle Daten aus dem Hadoop-System wurden benötigt. Dabei gab es zwei Dinge, die etwas knifflig waren: Erstens mussten Hunderte Terabyte an Daten aus dem Unternehmensrechenzentrum in die Cloud transferiert werden. Und zweitens mussten die Ladestränge migriert werden, um die Daten auch in der Cloud weiterhin zur Verfügung zu haben.

Darüber hinaus ergab sich aus den Anforderungen, dass künftig viele verschiedene SAP-Systeme, Datenbanken und Dateien angebunden werden würden, aber auch nationale oder Use-Case-bezogene Daten einzelner Data-Sciences-Teams.

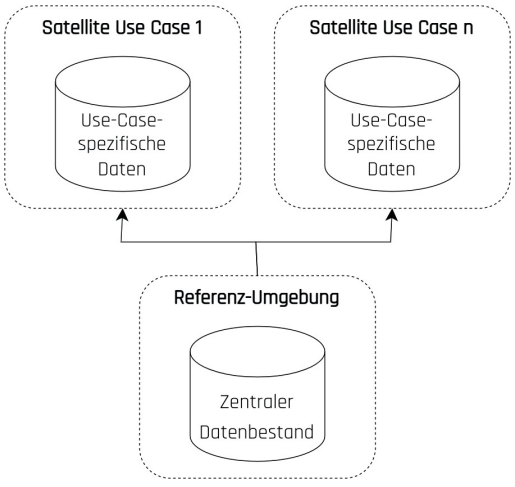

Abb. 1: Plattformstruktur

Die Mission beginnt

Unsere vier Eingangsfragen waren nun geklärt. Damit konnten wir das Ziel unserer „Weltraummission“ folgendermaßen formulieren:

Es wird eine Microsoft-Azure-Cloud-Plattform benötigt, die

- das vorhandene Hadoop-System vollständig ablöst,
- eine hohe Flexibilität bei der Anbindung von zusätzlichen Datenquellen bietet
- und gleichzeitig den nationalen Data-Science-Teams möglichst viel Freiheit in der Datenanalyse bietet.

Was tun, wenn die Use-Cases mehr werden?
Eine wichtige Frage, denn ein Szenario wie beim aktuellen Hadoop-System, bei dem die Anzahl der potenziellen Use-Cases schon zu Beginn im hohen zweistelligen Bereich lag – Tendenz steigend –, musste verhindert werden, wobei ein Data-Analytics-Projekt von mehreren Monaten als ein Use-Case galt.

Reicht bei dieser Fülle an unterschiedlichen Anforderungen eine einzelne Datenplattform noch aus?

Abbildung 1 zeigt ein Konzept, das solche Abhängigkeiten teilweise auflöst: durch eine zentral verwaltete Referenzumgebung, bei der Use-Case-bezogene Umgebungen wie Satelliten um ihren Referenzplaneten kreisen.

Die Referenzumgebung stellt alle zentral verwalteten Daten bereit und kümmert sich um die Aufbereitung und Qualitätssicherung. Sie diente also zunächst als Umgebung für die Migration der Hadoop-Umgebung und ist auch inhaltlich Bestandteil der zentralen IT.

Die Use-Case-Satelliten können lesend auf den Referenzdatenbestand zugreifen und die Daten konsumieren. Außerdem können sie eigene spezifische Daten in ihre Satelliten laden. Für deren Verfügbarkeit, Qualität und Aktualität sind sie dann selbst verantwortlich.

Dieser Ansatz hat zwei Vorteile:

- Zum einen sind die Use-Case-Satelliten sehr flexibel, da sie nahezu unabhängig voneinander agieren können.
- Auf der anderen Seite werden Redundanzen bei der Datenhaltung vermieden.

Wie sieht das Basis-Set-up aus?
Jede Umgebung und jeder Satellit bekam eine eigene Azure Subscription, was die Flexibilität erhöhte. Außerdem konnten die Kosten, die jeder Use-Case für die Datenplattform verursacht, zielgerichtet zugeordnet und Rechte zielgerichtet vergeben werden.

Die Bereitstellung der Azure Subscriptions übernahm das im Unternehmen bereits vorhandene Cloud-Infrastruktur-Team. Neben der Subscription und einem zugehörigen Service Principal kümmerte es sich um allgemeine Netzwerkkomponenten, zum Beispiel um ein eigenes, gepeertes virtuelles Netzwerk pro Umgebung, das Zugriffe auf das Unternehmensnetzwerk ermöglicht.

Welche Technologie für den Storage-Service?
Wie in der Cloud üblich, wurden die Bereiche Storage und Compute getrennt voneinander betrachtet. Auch wegen der großen Datenmengen setzte das Team für die Datenspeicherung auf Azure-Storage-Accounts. Abbildung 2 zeigt die Unterteilung in vier Schichten:

1. Im **Raw Layer** befinden sich die unveränderten Daten aus den Quellsystemen.
2. Diese werden im **Stage Layer** in das einheitliche Datenformat Apache Parquet überführt.
3. Im **Business Storage** werden sie systemübergreifend miteinander verknüpft.
4. In der letzten Schicht, dem **Serve Layer,** werden Teile der Daten Use-Case-spezifisch aufbereitet.

Für jede Schicht wurde ein eigener Data Lifecycle definiert, der beschreibt, wann Daten in den unterschiedlichen Access Tiers verfügbar sind. Das ist wichtig, um die Speicherkosten zu minimieren. Im Raw Layer werden Daten beispielsweise nach 90 Tagen ohne Zugriff vom Hot Tier in den Cold Tier überführt und weitere 30 Tage später in den Archive Tier.

Abbildung 3 zeigt zusätzlich einen weiteren Broker Storage in der Referenzumgebung. Damit lassen sich in einzelnen Satelliten-Umgebungen eigene, externe Daten bereitstellen.

Mit Hilfe des Azure Storage Explorer oder des Azure Storage Browser können Daten dort abgelegt und später in die entsprechenden Satelliten-Umgebungen transferiert werden. Um die Abhängigkeiten zwischen einzelnen Satelliten-Umgebungen zu minimieren, wurde im Broker Storage für jeden Satelliten ein eigener Container angelegt.

Warum werden externe Daten nicht direkt in den Raw Storage eines Satelliten geladen, fragen Sie sich jetzt vielleicht. Die Antwort lautet: Aus Sicherheitsgründen. Der Broker Storage ist das einzige Eingangstor für externe Daten, und das kann zentral überwacht werden.

Welche Technologie für den Compute-Service?
Die Auswahl der passenden Compute-Services gestaltete sich etwas schwieriger.

Aus den technischen Anforderungen ergaben sich bestimmte Notwendigkeiten: Es brauchte ein grafisches ETL-Tool und eine ganzheitliche Data-Science-Oberfläche. Die Oberfläche sollte sich möglichst für alle Disziplinen in Data Engineering und Data Science eignen. Dafür waren zwei Gründe maßgeblich:

- Einzelne Entwicklerteams benötigten eine einheitliche Umgebung mit einem zentralen Einstieg.
- Aus Security-Gründen sollte der direkte Zugriff auf das Azure-Portal vermieden werden.

LARS BEUMANN, Datenarchitekt bei der OPITZ CONSULTING Deutschland GmbH, berät seit mehr als zehn Jahren Unternehmen vorwiegend rund um datengetriebene Themen. Mit dem Blick für moderne Datenlandschaften findet er gemeinsam mit seinen Kunden maßgeschneiderte Lösungen in den Bereichen Business Intelligence, Analytics und Process Intelligence. Seit August letzten Jahres ist er auch als Dozent an der Fachhochschule der Wirtschaft tätig.
E-Mail: Lars.Beumann@opitz-consulting.com

Als grafisches ETL-Tool wählte das Team die Azure Data Factory aus. Dafür sprachen vor allem die gute Integration in die Microsoft-Azure-Cloud-Welt sowie die große Auswahl an Konnektoren [Mic22-2]. Bei der ganzheitlichen Data-Science-Oberfläche schwankte das Team zunächst zwischen Azure Synapse und Databricks.

Azure Synapse punktete mit der Möglichkeit, verschiedene Azure-Services wie Pipelines, Machine Learning Workbench, Azure Functions etc. miteinander zu verbinden. Diese starke Integration in die Azure-Cloud ist aber gleichzeitig auch eines der Gegenargumente: Dadurch entstünde eine hohe Abhängigkeit von Microsoft. Das würde die Migration zu einem anderen Cloud-Anbieter in der Zukunft nahezu unmöglich machen.

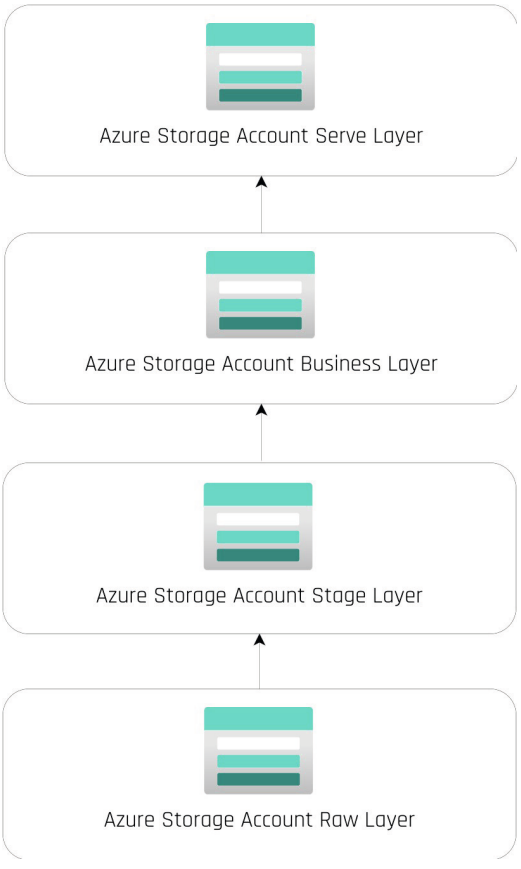

Abb. 2: Storage-Account-Struktur

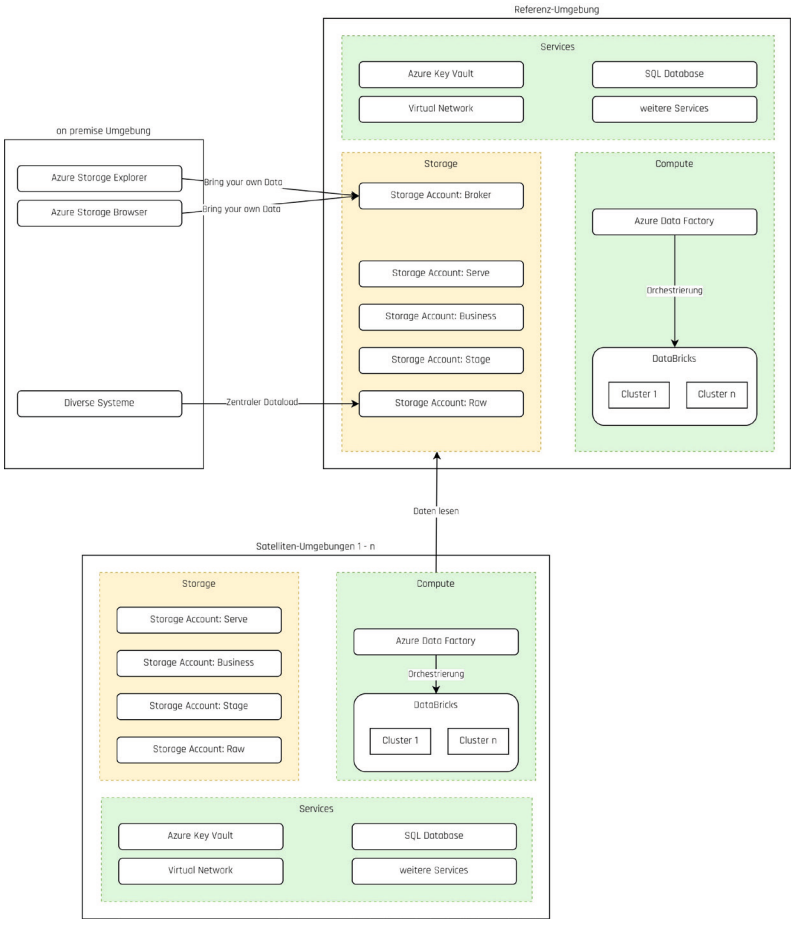

Abb. 3: Umgebungsdetails

Die Funktionalitäten von Databricks standen denen von Azure Synapse in nichts nach. Hier stehen einige Funktionen, zum Beispiel neue Spark-Features, teilweise sogar früher zur Verfügung, und theoretisch könnte Databricks auch bei einem anderen Cloud-Anbieter betrieben werden [Mic23; Dat23]. Daher fiel die Entscheidung am Ende auf Databricks.

Natürlich wurden neben den zentralen Ressourcen wie Storage-Accounts, Azure Data Factory und Databricks noch viele weitere benötigt. Hierzu gehörten unter anderem Azure Key Vault, Virtual Networks, MS SQL Server. Welche Ressourcen jeweils benötigt werden, ist abhängig vom Use-Case und somit für jede Satelliten-Umgebung anders.

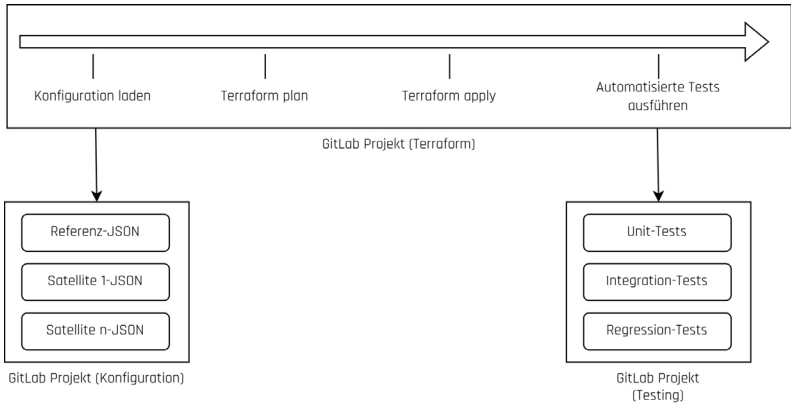

Abb. 4: Deployment-Pipeline

Erste Satelliten heben ab

Nachdem die zentralen technischen Entscheidungen getroffen waren, konnte das Team damit beginnen, die Plattform sprintbasiert Schritt für Schritt aufzubauen.

Eine skalierbare Infrastruktur

Für eine möglichste effiziente Skalierung sorgte der Einsatz von Infrastructure as Code mittels Terraform. Dafür wurden unternehmensspezifische Terraform-Module entwickelt, die einzelne Azure-Ressourcen standardisiert und wiederholbar zur Verfügung stellen. Über umgebungsspezifische Konfigurationsdateien wurden die Referenzumgebung sowie erste Satelliten-Umgebungen definiert.

Neben einigen Metainformationen wie Umgebungsnamen oder IP-Adressräumen beinhalten die Konfigurationsdateien auch die Azure-Services. Somit kann für jede Umgebung frei entschieden werden, welche Komponenten in welcher Konfiguration deployt werden.

Abbildung 4 zeigt vereinfacht die verwendete Gitlab-Deployment-Pipeline. Die Terraform-Module sowie die Deployment-Pipeline befinden sich in einem eigenen Repository und sind getrennt von den Konfigurationsdateien, um eine Entkopplung der Umgebungskonfiguration und der Modulentwicklung sicherzustellen.

- Im ersten Schritt wird die passende Konfiguration geladen und in Terraform-Variablen übersetzt. Diese dienen als Input für den Terraform plan und apply.
- Für die Qualitätssicherung der Deployments wurde ein Test-Automation-Framework entwickelt, das auf Pytest basiert.
- Nach jeder Ausführung der Pipeline finden Tests statt, die beispielsweise die korrekte Konfiguration der Azure-Ressourcen überprüfen. Für das Bereitstellen einer neuen Satelliten-Umgebung musste demnach nur eine neue Konfigurationsdatei erstellt und die Deployment-Pipeline ausgeführt werden.

Frontrunner-Satelliten

Um möglichst zeitnah erste Erfahrungen bei der Verwendung der Plattform zu sammeln, wurden schon früh im Projekt erste Frontrunner-Satelliten bereitgestellt. Für jeden Frontrunner gab es ein eigenes Projektteam, das erste Use-Cases umsetzte und die Plattform damit auf Herz und Nieren prüfte.

Neben anderen Kleinigkeiten stellte auch das Redeployment der Plattform das Projektteam immer wieder vor Herausforderungen. So musste sichergestellt werden, dass keine Storage-Accounts gelöscht werden und damit Daten verloren gehen. Dazu wurden beispielsweise für bestimmte Releases Terraform-Skripte und Powershell bereitgestellt. Sie können alte Storage-Accounts im Terraform State bekannt machen oder vor dem Deployment ein Backup des Storage-Accounts erstellen.

Die frühe produktive Nutzung hatte zur Folge, dass sich ebenfalls früh neue Anforderungen ergaben. Das konnten kleine, spezifische Anforderungen

sein, wie das automatisierte Bereitstellen von Python-Bibliotheken in einzelnen Databricks-Umgebungen, aber auch komplexe Vorhaben, die zu mehr Flexibilität bei der Konfiguration führen, oder eine Kommandozentrale, die für mehr Transparenz sorgen sollte.

Mission geglückt?

Als dieser Artikel geschrieben wurde, war die Plattform teilweise schon produktiv. Das heißt, ein Großteil der Hadoop-Daten befand sich nach tagelangen Kopieraktivitäten in der Cloud, und auch Teile der initialen Ladestränge wurden mittels eines teilautomatisierten Frameworks von Hadoop nach Databricks und Azure Data Factory migriert. Mehr als 20 Satelliten waren zu der Zeit im Gebrauch. Darauf wurden also bereits Data-Science-Use-Cases entwickelt.

Bis die multinationale Datenplattform mit allen Funktionen bereitsteht, wird es aber noch etwas Zeit brauchen. Denn neben der Weiterentwicklung müssen auch noch grundlegende Weichen gestellt werden, um die Plattform aus einem Projektmodus in einen Betriebsmodus zu überführen.

Dennoch – unser Team ist zufrieden: Ein spannendes Projekt zieht seine Kreise. Die ersten Satelliten befinden sich in der Umlaufbahn und die Nachfrage nach weiteren, fortschrittlicheren Umgebungen ist groß. Alle sind sich einig: Die Mission ist geglückt!

Quellen

[Azu23] Microsoft Azure: Mit Clouddiensten aktuelle Herausforderungen meistern. https://azure.microsoft.com/de-de/explore/why-azure/, abgerufen am 31.1.2023

[Dat23] Databricks: Databricks runtime releases. 18.1.2023, https://docs.Databricks.com/release-notes/runtime/releases.html, abgerufen am 31.1.2023

[Mic22-1] Microsoft: Overview of Azure Data Lake Storage for cloud-scale analytics. 22.4.2022, https://learn.microsoft.com/en-us/azure/cloud-adoption-framework/scenarios/cloud-scale-analytics/best-practices/data-lake-overview, abgerufen am 31.1.2023

[Mic22-2] Microsoft: Übersicht über die Connectors in Azure Data Factory und Azure Synapse Analytics. 29.11.2022, https://learn.microsoft.com/de-de/azure/data-factory/connector-overview, abgerufen am 31.1.2023

[Mic23] Microsoft: Azure Synapse-Runtimes. 26.1.2023, https://learn.microsoft.com/de-de/azure/synapse-analytics/spark/apache-spark-version-support, abgerufen am 31.1.2023